Universidad Autónoma de Madrid

Departamento de Ingeniería Informática

Estimación Estadística de Consumo en FPGAs

TESIS DOCTORAL

Autor: Elías Todorovich

Director: Eduardo Boemo Scalvinoni

Escuela Politécnica Superior

Julio de 2006

Resumen

Varios motivos impulsan a la industria y a las universidades a estudiar el consumo de energía de los circuitos digitales actuales: movilidad de los equipos electrónicos, coste de los accesorios de refrigeración y encapsulados, impacto medioambiental, coste de la energía, tasa de fallos, y restricciones a la densidad de integración VLSI.

Diseñar circuitos VLSI para bajo consumo requiere de una metodología en cada etapa del proceso de desarrollo. Los principales componentes de tal metodología son estimación y optimización de consumo. A pesar de las muchas técnicas para estimación de consumo publicadas recientemente en la literatura especializada, el problema todavía no está resuelto completamente, ni siquiera al nivel de puertas lógicas. Debido a la complejidad computacional que requiere la estimación de consumo, no puede obtenerse precisión y velocidad al mismo tiempo. Este problema se observa en la estimación de consumo medio para nodos individuales; y para el consumo promedio total en los circuitos secuenciales grandes.

Esta tesis tiene como objetivo estimar el consumo de potencia media a nivel de nodos individuales en FPGAs. Para ello, se desarrolló una plataforma de estimación de consumo con herramientas y estructuras de datos comunes y reusables dentro de una familia de aplicaciones EDA de diseño para bajo consumo. El software desarrollado se integra con las herramientas de Xilinx. Adicionalmente, se ha incorporado el simulador Modelsim dentro del bucle que implementa la técnica propuesta. Igualmente, al haberse usado formatos estándares para el intercambio de datos, es posible integrar la herramienta desarrollada en esta tesis en ambientes de diseño para FPGA de otros fabricantes, y lo mismo para otros simuladores que soporten formatos estándares.

Para validar los resultados, se realizaron más de 1500 simulaciones y experimentos sobre dispositivos Virtex, Virtex-E y Virtex-II cubriendo 10 años de evolución de la tecnología de dispositivos lógicos programables. Las medidas físicas obtenidas permitieron orientar la investigación y a la vez facilitar la evaluación, calibración y depuración del software. El resultado es una herramienta precisa de estimación

desarrollada sobre una plataforma general para el diseño de circuitos de bajo consumo.

Abstract

There are several reasons that strongly lead the industry and the researchers to study the power consumption of the current digital circuits: mobile electronic devices, refrigeration accessories and packaging costs, environmental impact, energy cost, reliability, and restrictions to the VLSI integration density.

Designing VLSI for low power requires a design methodology at every level of the design process. The main components of such a methodology are estimation and optimization. Despite the several techniques recently proposed for power estimation, the problem is not completely solved yet, even at the gate level. Due to the computing complexity of power estimation, accuracy and speed can not be met together. This problem is observed for individual gate average power, and for total average power estimation in large sequential circuits.

The goal in this thesis is the average power estimation at the individual nodes level on FPGAs. In order to do it, a power estimation framework was developed. This general framework has common tools and data structures that can be reused within a family of EDA tools for low power design. At this time this software is integrated with that provided by Xilinx, and its operation has been evaluated with the Modelsim simulator. However, due to the use of standard formats, an easy integration is expected in other design environments for FPGA, and the same feature for simulators that support standard formats.

More than 1500 simulation runs and experiments were performed with Virtex, Virtex-E and Virtex-II devices on available development boards covering 10 years of PLD evolution. These physical measurements allowed the evaluation, tuning and debugging of the developed software. As a result an accurate tool was developed over the mentioned power estimation framework.

Agradecimientos

A Eduardo Boemo, director y amigo, fundamentalmente por haberme dado una oportunidad; por hacer posible este trabajo no sólo en el plano académico, sino también en otros aspectos más mundanos pero vitales en la supervivencia del doctorando, y por tantísimas actitudes como invitarnos los domingos a su casa estando recién llegados a España: gracias. Maite: muchas gracias.

Esta tesis tampoco sería posible sin la confianza de Javier Garrido y Francisco Gómez Arribas, la solidaridad más allá del compañerismo de Sergio López Buedo, la amistad de los compañeros del laboratorio: Gustavo Sutter, Iván González, Juan González Gómez, Guillermo González, Estanislao Aguayo, Alberto Regadío, Alberto Martín-Ortega Rico, y muchos otros becarios y profesores, que durante estos años pusieron la mejor predisposición para el desarrollo de los trabajos en un ambiente de camaradería. Gracias también a Martín Gilabert por su trabajo y entusiasmo en las primeras experiencias vinculadas con esta tesis. Mi reconocimiento especial es para el Prof. Javier Martínez.

Sin Juana Calle no sólo me hubiera sido mucho más difícil cumplir con todo el formalismo que se requiere en un doctorado, sino que no habría mantenido el ritmo académico que hace que estos estudios tengan un final a su debido tiempo. Muchas gracias Juana por tus e-mails y por toda tu ayuda, y tu responsabilidad y seriedad dentro del afecto que pones en tu trabajo. Gracias también a Pablo y a todo el personal de la EPS. Espero ser capaz de mostrarles a todos mi gratitud.

También debo agradecer a las siguientes instituciones argentinas y españolas por su apoyo en esta investigación, por orden de aparición en el desarrollo de las tareas:

- Facultad de Ciencias Exactas de la Universidad Nacional del Centro de la Provincia de Buenos Aires
- CONICET (Consejo Nacional de Investigaciones Científicas y Técnicas) de la República Argentina
- Ministerio de Ciencia y Tecnología de España
- Escuela Politécnica Superior de la Universidad Autónoma de Madrid

- Escuela Politécnica Superior de Gandía, de la Universidad Politécnica de Valencia

En lo que concierne a la revisión del texto en inglés, le agradezco especialmente a Walter Routley, y también a Gery Bioul, Hernán Techeiro y naturalmente a Eduardo por su trabajo y paciencia.

Es bueno hacer memoria y este balance al final de un trabajo tan largo. Le doy las gracias de todo corazón a mi familia, amigos y a Dios.

Table of Contents

1 INTRODUCTION 1

1.1 AVERAGE POWER CONSUMPTION 2
1.1.1 PORTABLE ELECTRONIC PRODUCTS 2
1.1.2 ENVIRONMENTAL IMPACT 3
1.2 MAXIMUM POWER CONSUMPTION 4
1.3 POWER CONSUMPTION IN FPGAS 6
1.4 RESEARCH OBJECTIVES 7
1.5 THESIS ORGANIZATION 8
REFERENCES 9

2 VLSI POWER CONSUMPTION 11

2.1 ANALYSIS OF POWER CONSUMPTION 12
2.1.1 THERMODYNAMICS OF COMPUTATION 12
2.1.2 SOURCES OF POWER DISSIPATION 14
2.2 POWER CONSUMPTION IN FPGAS 18
2.2.1 PROGRAMMABLE ROUTING 19
2.2.2 PHYSICAL CAPACITANCE 20
2.2.3 SWITCHING ACTIVITY 21
2.3 SWITCHING ACTIVITY COMPUTATION 22
2.3.1 DEPENDENCE ON THE INPUT PATTERNS 22
2.3.2 DELAY MODEL 25
2.3.3 LOGIC FUNCTION 27
2.3.4 CIRCUIT STRUCTURE 28
2.3.5 TECHNOLOGY-DEPENDANT FACTORS 29
2.4 CONCLUSIONS 29
REFERENCES 30

3 POWER ESTIMATION TECHNIQUES 33

3.1 POWER ESTIMATION HISTORY –OR SIMULATIVE APPROACHES 35
3.1.1 SPICE-LIKE CIRCUIT SIMULATION 35
3.2 STATISTICAL APPROACHES 37
3.2.1 MONTE CARLO SIMULATION 37
3.2.2 TOTAL POWER (MCPOWER) 38
3.2.3 POWER OF INDIVIDUAL GATES (MED) 40
3.2.4 IMPROVEMENTS IN STATISTICAL METHODS 41
3.3 PROBABILISTIC APPROACHES 42
3.3.1 SOME IMPORTANT DEFINITIONS 43
3.3.2 PROBABILISTIC POWER ESTIMATION TECHNIQUES 44
3.4 SEQUENTIAL CIRCUITS 48
3.4.1 STATISTICAL APPROACHES 48
3.5 POWER ESTIMATION METHODS APPLIED ON FPGAS 58
3.5.1 RELATED WORKS AT THE UAM 60
3.6 CONCLUSIONS 61
REFERENCES 61

4 A-DYP: A TOOL FOR AVERAGE POWER ESTIMATION IN FPGAS 65

4.1 A-DYP MAIN STRUCTURE 67
4.2 THE PREPARATION PHASE 68
4.2.1 USER INTERFACE 70
4.2.2 POWER ESTIMATION SET-UP PHASE 73
4.3 ACTIVITY ESTIMATION SUB-SYSTEM 74
4.3.1 INPUT PATTERNS FOR THE PATTERN GENERATOR 77
4.3.2 THE POWER ESTIMATION PLATFORM 78
4.4 POWER COMPUTATION SUB-SYSTEM 78
4.5 THE POWER DATABASE 82
4.6 CONCLUSIONS 84
REFERENCES 85

5 ACTIVITY ESTIMATION SUB-SYSTEM 87

5.1 THE PATTERN GENERATOR 88
5.2 SIMULATING THE INPUT PATTERNS AND SAVING THE SIMULATION RESULTS 91
5.3 ANALYZING THE GENERATED ACTIVITY 93
5.3.1 THE SET-UP PERIOD 93
5.3.2 COUNTING THE EFFECTIVE TRANSITION NUMBER 94
5.4 UPDATING NODE STATISTICS 95
5.5 CHECKING THE STOPPING CRITERIA 97
5.6 CONCLUSIONS 97
REFERENCES 97

6 POWER COMPUTATION SUB-SYSTEM 99

6.1 PARSING THE VHDL SIMULATION MODEL 100
6.1.1 WHY PARSING THE VHDL MODEL? 101
6.1.2 OBTAINING THE IDENTIFIERS 103
6.1.3 OPTIMIZATION 105
6.2 PARSING THE XILINX DESIGN XDL FILE 105
6.2.1 ANALYZING AN FPGA SLICE DEFINITION 106
6.2.2 ANALYZING A NET DEFINITION 109
6.3 GENERATING THE XML SETTINGS FILE 111
6.3.1 USING A PACKAGE TO GENERATE XML 112
6.4 EXTRACTING THE CAPACITANCES 113
6.4.1 PARSING THE XILINX CAPACITANCE REPORT FILE (PWA) 114
6.5 CALCULATING THE POWER CONSUMPTION AND WRITING A REPORT 115
6.6 GENERATING THE POWER MAPS 116
6.7 CONCLUSIONS 116
REFERENCES 117

7 TEST CASES AND ANALYSIS 119

7.1. TEST CIRCUITS 119
7.1.1 QUADRATURE DIRECT DIGITAL FREQUENCY SYNTHESIZERS (QDDFS) 119
7.1.2 DISTRIBUTED-ARITHMETIC FIR FILTER (FIRDA) 122

7.1.3 FAST FOURIER TRANSFORM (FFT) 123
7.1.4 ARITHMETIC CIRCUITS 123
7.2 ANALYSIS OF THE RESULTS 124
7.2.1 TECHNIQUE CHARACTERIZATION 124
7.2.2 SOFTWARE EVALUATION 125
7.2.3 GRAPHICAL REPRESENTATION OF THE RESULTS 126
7.3 POWER MEASUREMENT 126
REFERENCES 129

8 EXPERIMENTAL RESULTS 131

8.1 A-DYP PRELIMINARY EVALUATION 132
8.1.1 TOTAL DYNAMIC POWER ESTIMATION 133
8.1.2 IMPACT OF THE INPUT PATTERNS DEFINITION 134
8.1.3 TOOL'S EVALUATION 135
8.1.4 POWER MAPS 136
8.2 A FIRST COMPLETE TEST CASE: FIRDA FILTERS 139
8.2.1 TOTAL DYNAMIC POWER ESTIMATION 139
8.2.2 ESTIMATING POWER FOR INDIVIDUAL NODES 140
8.2.3 ACCURACY VS. EXECUTION TIME TRADEOFF 147
8.2.4 TOOL'S EVALUATION 152
8.2.5 POWER MAPS 154
8.2.6 ENERGY ANALYSIS OR ENERGY OF THE COMPUTATION 158
8.3 IMPACT OF THE INPUT PATTERNS DEFINITION 160
8.3.1 TOTAL DYNAMIC POWER ESTIMATION 162
8.3.2 DYNAMIC POWER ESTIMATION FOR INDIVIDUAL NODES 166
8.3.3 INPUT PATTERNS FROM REAL SCENARIOS 168
8.4 ADDITIONAL EXPERIMENTS ON VIRTEX-II 171
8.4.1 IMPACT OF THE INPUT PATTER DEFINITION ON TOTAL POWER 172
8.4.2 DYNAMIC POWER ESTIMATION FOR INDIVIDUAL NODES 173
8.4.3 ACCURACY VS. EXECUTION TIME TRADEOFF 175
8.4.4 TOOL'S EVALUATION 178
8.4.5 POWER MAPS 179

8.5 CONCLUSIONS **181**
REFERENCES **182**

9 CONCLUSIONS AND FUTURE WORKS **183**

9.1 MAIN CONTRIBUTIONS OF THIS THESIS **183**
9.1.1 THE POWER PLATFORM FRAMEWORK AND A-DYP 184
9.1.2 SHORT-PULSE FILTERING AS A CALIBRATION RESOURCE 185
9.1.3 A-B NODES CLASSIFICATION 186
9.1.4 ENERGY OF THE COMPUTATION 186
9.2 REVERSE ENGINEERING **186**
9.3 PUBLICATIONS **187**
9.4 FUTURE TASKS **189**
9.4.1 HIGH-LEVEL POWER ESTIMATION 191
9.5 HOW TO ESTIMATE POWER CONSUMPTION **192**
9.6 HOW TO BUILD A POWER ESTIMATOR **193**
REFERENCES **193**

COMPLETE REFERENCES LIST **195**

TCL/TK SCRIPT FOR THE A-DYP POWER ESTIMATION TOOL **201**

INPUT PATTERNS FILE (.DO) **213**

THE CONFIGURATION .INI FILE **215**

POWER REPORT FILE **219**

Acronyms

API	Application Program Interface
ASIC	Application Specific IC
ATP	Area Time Power
CAD	Computer Aided Design
CLB	Configurable Logic Block
CMOS	Complementary Metal Oxide Semiconductor
DSP	Digital Signal Processing
DUT	Design Under Test
EDA	Electronic Design Automation
FF	Flip-Flop
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
HDL	Hardware Description Language
IC	Integrated Circuit
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronic Engineers
IEEE 1076	IEEE VHDL standard
IEEE 1364	IEEE Verilog HDL standard
iid	Independent and identically distributed
IOB	Input/Output Block
ITRS	International Technology Roadmap for Semiconductors
LUT	Look-Up Table
LSB	Less Significant Bit
NC Set	Near Closed Set

MSB	Most Significant Bit
MPU	Microprocessor Unit
OVI	Open Verilog International
PAR	Place and Route
PCB	Printed Circuit Board
PPR	Partitioning, Place and Route
P&R	Place and Route
RTL	Register Transfer Level
QAM	Quadrature Amplitude Modulation
QDDFS	Quadrature Direct Digital Frequency Synthesizer
QPSK	Quadrature Phase Shift Keying
SDF	Standard Delay Format
TWG	Technical Working Group
UUT	Unit Under Test
VITAL	VHDL Initiative Towards ASIC Libraries
VHDL	VHSIC (Very High-Speed IC) HDL
VI	VHDL International

Chapter 1.

"If performance per watt is to remain constant over the next few years, power costs could easily overtake hardware costs, possibly by a large margin." and "…one could envision bizarre business models in which the power company will provide you with free hardware if you sign a long-term power contract." Luiz André Barroso, a principal engineer at Google [Bar05]

1 Introduction

Revising the history of the technology [Rab96], power consumption is a recurring problem in an area like digital circuit design where complexity grows according to Moore's law. When CMOS technology was introduced, it was believed that the power consumption problem was solved for digital circuits. In this technology, the static power consumption is very low and the electric current only flows in the circuit when some computing is done. In that time, the major concerns of the VLSI designers were area and performance, and power consumption was mostly of secondary importance. Nevertheless, the number of transistors per unit of area, and the number of transistors in a single die have reached a limit where the power consumption is a problem. Now power is as important as area and speed. But several other factors have contributed to this trend as is briefly discussed below. This is particularly true for FPGAs, where the power consumption noticeably rises due to the increase in the clock frequency, chip area (capacitance), and the ability to be programmed.

Perhaps the primary driving factor that made power consumption as important as area and speed has been the remarkable success and growth of portable electronic products. In these applications, average power consumption is a critical design

concern. In addition, maximum power consumption is also a constraint in the contemporary electronic industry.

In this Chapter, a motivation for low power design is presented together with the research objectives of this thesis,

1.1 Average Power Consumption

1.1.1 Portable Electronic Products

MP3 players, pagers, mobile phones, portable CD players, notebooks, etc., demand high-speed computation and complex functionality with low power consumption. In these applications average power consumption is a critical design concern.

Example 1.1: A portable multimedia terminal, when implemented using off-the-shelf components not optimized for low-power operation, consumes about 40 W. For 10 hours of operation between recharges, with a Lithium-Ion battery with 100 Wh/kg of energy density[1], it will require 4 kg of batteries. Energy density and weight is displayed at Table 1.1 for several battery technologies, including the Lithium-Polymer-Potential technology not yet available in the market.

Technology	**Energy Density** [Wh/kg]	**Weight** [Kg]
Nickel-Metal-Hydride	80	5.00
Lithium-Ion	100	4.00
Lithium-Polymer-Potential	400	1.00

Table 1.1: Battery Sizing

Example 1.2: A more realistic example can be outlined looking at an actual portable computer running a multimedia application. It consumes 24.8 Watts [Kol01]. In the scenario as in example 1.1 (10 hours of operation between recharges, lithium-ion

[1] Energy density is the amount of energy stored per unit of weight expressed in watt-hour per kilogram.

battery) this computer will require 2.48 kilograms of battery. Battery sizing for other technologies is shown in Table 1.2.

Technology	**Energy Density** Wh/kg	**Weight** Kg
Nickel-Metal-Hydride	80	3.10
Lithium-Ion	100	2.48
Lithium-Polymer-Potential	400	0.62

Table 1.2: Battery Sizing

From the examples, we can see that without low-power design techniques portable devices will suffer usability problems.

1.1.2 Environmental Impact

For consumer electronics power savings means significant money saving. It could also be viewed as a long term objective for low power design. The smaller the power dissipation of electronic systems, the lower the heat pumped into the rooms, the lower the electricity consumed and hence the lower the impact on global environment, and less the office noise (e.g., due to elimination of a fan from the desktop), and less the office heat removal requirements [Ped97].

The United States Environmental Protection Agency (EPA) publishes Energy Star guidelines that suggest ways to reduce power consumption and to save money by eliminating unnecessary energy use. Office equipment, led by computers, is the fastest growing electrical load in the business world. In fact, office equipment accounts for 7-20% of all commercial sector electricity usage [Kaw01] [Lai01].

Some works studied the electric energy used by computing equipment ([Ang01] [Kaw00] [Hay01]). [Kaw01] and [Lai01] report that total power use by office and network equipment in the U.S. is about 2% of total electricity use. They also say that power management currently saves 23 % of the energy consumption; and complete saturation with proper power management and night shut down will save 38 %. On a monthly or annual basis, owners could save millions of dollars in electricity costs and the pollution associated with electricity use can be reduced.

1.2 Maximum Power Consumption

Table 1.3 shows maximum power consumption trends for MPUs and high performance ASICs according to the ITRS forecast [ITRS05]. These trends are presented in three categories:

1) high-performance desktop applications, for which a heat sink on the package is permitted;

2) cost-performance, where economical power management solutions of the highest performance are most important; and

3) portable battery operations (now designated as the “Harsh” application category by the Assembly and Packaging TWG).

Total power consumption continues to increase, despite the use of a lower supply voltage. The increased power consumption is driven by higher chip operating frequencies; the higher interconnect overall capacitance and resistance, the increasing gate leakage of exponentially growing, and scaled on-chip transistors.

Fig. 1.1 shows maximum power predictions for high-performance and cost performance MPUs and high performance ASICs according to different ITRS updates since 2001. Note that, for high-performance MPU, power consumption significantly exceeds the high-performance single-chip package power limits, even with allowed power densities in excess of 250 W/cm^2 [ITRS05]. Thus, power will be limited more by system level cooling and test constraints than packaging. This fact is considered from the 2003 forecast.

[Gun01] reports that power consumption of processors produced by Intel almost doubles every 4 years[2]. The cost associated with packaging and cooling of such devices could reach relatively high values.

[2] In this moment, there is a controversy about the maximum power dissipation reported by the industry. An independent analyst, [Smi01], reports 72 W of maximum power consumption for the 1.4 GHz Athlon; and 87-88 W for the 1.8 GHz Pentium 4. The same article refer to an interesting concept: the programs used by the vendors to run that tests are safeguarded as restricted information in order to prevent the development of thermal viruses.

	2007	2010	2012	2014	2016	2018	2020
High-performance with heatsink (W)	189	198	198	198	189	198	198
Cost-performance (W)	104	119	125	137	151	151	157
Battery (W) – (low cost/hand Held)	3.0	3.0	3.0	3.0	3.0	3.0	3.0

Table 1.3: Allowable maximum power for the next years

Since core power consumption must be dissipated through the packaging, increasingly expensive packaging, cooling strategies and protection mechanisms are required as chip power consumption increases. Consequently, there is a clear financial advantage to reducing the power consumed in high performance systems (See Fig. 1.2).

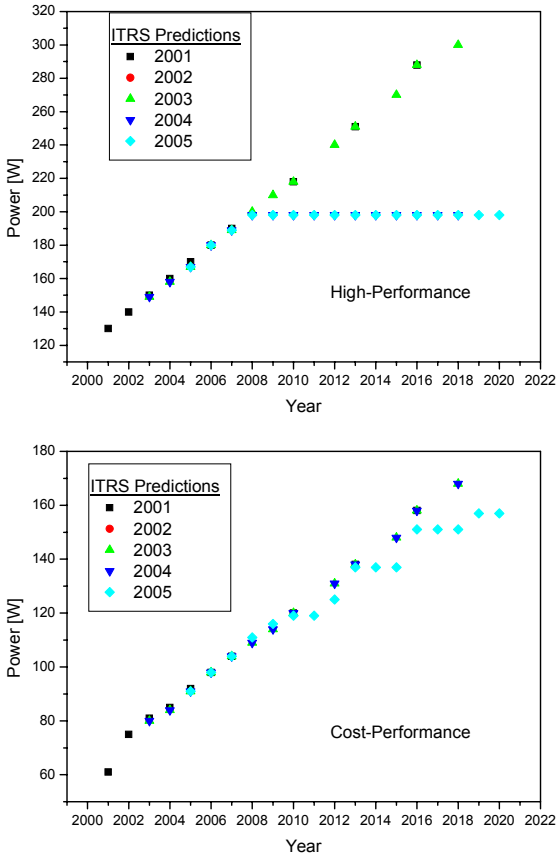


Fig. 1.1: Trends in maximum power consumption according to ITRS

Fig 1.2 shows that, as power increases, the relationship between the dissipated power and the cooling solution cost that should be adopted is non-linear.

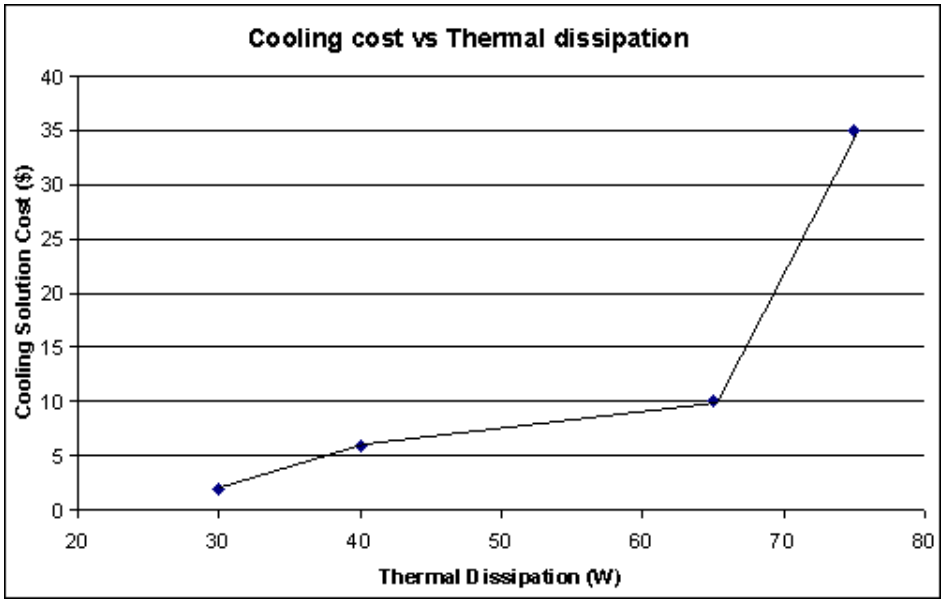


Fig. 1.2: Cost Associated with Packaging and Cooling Systems. Reproduced from [Gun01]

Reliability is another technological issue. The systems must be designed to ensure that the chip does not exceed the maximum specified operation temperature, even when it is dissipating the maximum power. High power systems often run hot and high temperature trends to increase occurrences of silicon failure mechanisms [Lal97]. For example it has been shown that the switching activity is a valuable parameter to estimate the available working time before failure [Iye86].

Another crucial driving factor is that excessive power consumption is becoming the limiting scale factor in integrating more transistors on a single chip or on a multiple-chip module.

Power supply rail design is a specific topic directly related to the maximum dissipated power.

1.3 Power Consumption in FPGAs

Although the background, techniques, research literature, theory, and motivations presented in this thesis are mostly aimed at CMOS circuits, the implemented software is integrated with the Xilinx design flow and experiments are performed on SRAM-

based FPGAs of the same vendor. The ability of these programmable logical devices for prototyping, digital electronics teaching and training, adaptation to modern systems with evolving specifications, economical viability for specific markets like industrial automation, control, aerospace, or high-end medical imaging, and the availability of state-of-the-art digital technology for end users all around the world, have spread FPGAs quickly since their commercial launch in 1985 [Wil99]. Furthermore, FPGAs are low cost devices in these contexts and vendors offer friendly development software, tutorials, teaching resources, and tools that enable accessing the technology in a few weeks with a reasonable level of expertise after a relatively short training period. Beyond these applications, FPGAs are also competing with ASICs and DSPs in the digital consumer market.

The specific nature of FPGAs leads to particular design techniques and the EDA tools for them also require specific considerations from their conception (i.e. analysis) to their implementation. The abundant routing resources and programmability significantly affect the device's power consumption. Furthermore, state-of-the-art VLSI technologies are applied to current FPGA production, e.g. 90 nm to Virtex-4 [XDS06] and Altera Stratix-II [Alt05]. This leads to unprecedented integration levels and transistor counts on a single chip which converts FPGA to high-performance ICs with the power consumption problems reported by ITRS [ITRS05]. In this way, it is interesting to evaluate techniques proposed to ASICs and adapt or develop new ones according to FPGA technologies.

1.4 Research Objectives

In this work, FPGAs are more than a technological framework in order to verify theoretical research; they are the core of this study. The main goal in this thesis is the average power estimation at the individual nodes level on FPGAs. In order to do it, a power estimation framework is developed. This general framework has common tools and data structures that can be reused within a family of EDA tools for low power design.

The instance of this framework presented in the following chapters is a gate-level statistics-based power estimation tool for both the total and individual nodes power. This tool must produce accurate estimation values, i.e. with less than 10% of error in

the total power in relation to physical measurements over real devices, in a reasonable run time when the required accuracy is moderate, e.g. with 90% confidence that error is less than 10%.

1.5 Thesis Organization

This thesis is organized in nine chapters. This chapter has presented a motivation for and introduction to the low-power design problems that will be presented in the next chapters.

Chapter 2 analyzes CMOS and FPGA power consumption and presents the problems that must be overcome in order to estimate power or energy dissipation in these technologies. There is also an interesting summary about thermodynamics of computation in this chapter.

Chapter 3 describes the state-of-the-art in gate-level power estimation with focus on the statistical techniques. These first three chapters make up the introductory part of the thesis.

The lack of modern power estimation tools for FPGAs within commercial design flows, leads us to propose a power estimation framework in Chapter 4 that can support the statistical tool whose activity estimation and power computation parts are detailed in Chapters 5 and 6 respectively.

Chapter 7 presents test circuits used in the power estimation experiments and tool evaluation. This chapter also describes the measurement methodology necessary to generate values against which the estimations are compared. In this way the developed tool can be calibrated and debugged. The development boards and devices employed in the measurements are also briefly presented.

Chapter 8 shows the experimental results in a chronological order where the circuits, devices and tests are joined together in order to evaluate the development and produce a robust EDA tool.

Finally, Chapter 9 presents the main conclusions and topics for future research in the power-aware EDA tools and low-power design areas.

References

[Alt05] Altera Corp., “Stratix-II Device Handbook, 2005”, available at http://www.altera.com

[Ang01] J. Angel, “Emerging Technology: Energy Consumption and the New Economy”, Network Magazine, January 5, 2001.

[Bar05] Luiz André Barroso, “The Price of performance”, ACM Queue, pp. 49-53, September 2005.

[Gun01] S.H. Gunther, F. Binns, D.M. Carmean and J.C. Hall, “Managing the Impact of Increasing Microprocessor Power Consumption”, Intel Technology Journal Q1, 2001.

[Hay01] Brian Hayes, “The Computer and the Dynamo”, American Scientist, Vol 89, No 5, September-October, 2001. pp. 390-394.

[ITRS05] International Technology Roadmap for Semiconductors, 2005 Edition, available at http://public.itrs.net

[Iye86] R. Iye, D. Rossetti and M. Hsueh, “Measurement and Modelling of computer reliability as affected by system activity”, ACM Trans. On Computer Systems, 4(3):214-237, Aug. 1986.

[Kaw00] K. Kawamoto, J.G. Koomey, B. Nordman, R.E. Brown, M.A. Piette and A.K. Meier, “Electricity Used by Office Equipment and Network Equipment in the U.S.”, Proceedings of the 2000 ACEEE Summer Study on Energy Efficiency in Buildings. Asilomar, CA. August 2000.

[Kaw01] K. Kawamoto, J.G. Koomey, B. Nordman, R.E. Brown, M.A. Piette, M. Ting and A.K. Meier, “Electricity Used by Office Equipment and Network Equipment in the U.S.: Detailed Report and Appendices”, Lawrence Berkeley National Laboratory Internal Report LBNL-45917, University of California. February 2001.

[Kol01] J. Koliski, R. Chary, A. Henroid and B. Press, “Building the Power-Efficient PC”, Intel Press, 2001.

[Lai01] John A. Laitner, Jonathan Koomey, Ernst Worrell, Etan Gumerman. “Re-estimating the Annual Energy Outlook 2000 Forecast Using Updated Assumptions about the Information Economy”. Presented at the American Economic Association Conference. New Orleans, LA. January 7 2001. (Also LBNL-46418).

[Lal97] Pradeep Lall, “Influence of Temperature on Microelectronics and System Reliability”, CRC Press, 1997

[Ped97] M. Pedram, Design technologies for Low Power VLSI, In Encyclopaedia of Computer Science and Technology, Vo. 36, Marcel Dekker, Inc., 1997, pp. 73-96.

[Rab96] Rabaey, Jan M. “Digital integrated circuits: a design perspective”. Upper Saddle River: Prentice-Hall International, 1996.

[Smi01] Van Smith, “Pentium 4 Thermal Throttling”, available at http://www.vanshardware.com.

[Will99] Craig Willert, “The Evolution of Programmable Logic Design Technology”, XCell Journal, Issue 32, 2nd quarter 1999, pp. 5-8.

[XDS06] Xilinx Inc. “Virtex-4 Data Sheet: DC and Switching Characteristics”, 2006, available at http://www.xilinx.com

Chapter 2.

"When we come to design the Ultimate Computers of the far future, which might have "transistors" that are atom-sized, we will want to know how the fundamental physical laws will limit us. When you get down to that sort of scale, you really have to ask about the energies involved in computation, and the answer is that there is no reason why you shouldn't operate below kT". From "Lectures on Computation" by Richard P. Feynman [Fey96].

2 VLSI Power Consumption

This Chapter explains the sources of power dissipation for the CMOS technology. CMOS is (and will remain) the industry workhorse up to and beyond the year 2020 according to ITRS predictions [ITRS2005]. From 2020 it is anticipated new nanoscale devices representing alternatives to CMOS. These new devices will be introduced utilizing different and new ways of processing and storing information. Most of the proposed devices rely on new materials and properties not well studied yet.

The details about the power consumption in FPGAs are particularly specified. Finally, how the activity must be calculated has been carefully studied.

As power optimization is not the topic in this thesis, it is treated briefly in this chapter. A parallel work at our laboratory that presents optimization techniques applicable to SRAM-based FPGAs is [Sut05].

2.1 Analysis of Power Consumption

It has been shown that designing VLSI for low power requires a design methodology at every level of the design hierarchy. The main components of such a methodology are estimation and optimization [Lan94], the classical analysis and synthesis pair.

In order to estimate and optimize the power consumption of a digital circuit it is necessary to know how energy is dissipated. The way each factor interacts with the others will also clarify the effects these elements have on every VLSI design stage. This analysis will determine which elements can be overlooked within a specific design environment. Indeed, designing digital circuits with FPGAs requires specific assumptions, as it will be pointed out later on, after a brief discussion on power dissipation sources in CMOS circuits.

2.1.1 Thermodynamics of Computation

Beyond the technological frenzy in the electronic industry nowadays, it is important to stop a moment in order to study the fundamental laws about the power consumption and thermodynamics of computation. In Feynman's book "Lectures on Computation", Chapter 5 [Fey96], two essential questions are studied. The first one is: "How much energy must be used in carrying out a computation?" This thesis explores how much energy *will* be used in carrying out a computation within a particular technological context: SRAM-based FPGAs, its goal is the estimation of this amount of energy in advance. Nevertheless, the second question is more fundamental: "What is the *minimum* energy required to carry out a computation?" In this section this second question is considered. Although they are much more efficient than earlier computers, the existing ones dissipate enormous amounts of energy, $10^8 kT$, compared with the theoretical lower bound kT (T is the temperature and k is the Boltzman's constant). The main reason for this waste is the use of macroscopic components with relatively huge inertia which require macroscopic amounts of energy to switch quickly. On the other hand, a microscopic device such as DNA replication has relatively high energy efficiency: 20-100kT per operation.

In [Fey96] a physical definition of the information content of a message is studied. In general when we develop an algorithm, we do not think about this but

No computing can be done without the participation of the physical world.

Rolf Landauer, in his classic 1961 paper pioneered applying thermodynamics to computation [Lan61]. In that paper, it is claimed that any logically irreversible computation, such as the erasure of a bit, *must* be accompanied by a corresponding entropy increase; and any logically reversible computation *can* be executed by a thermodynamically reversible device. This is also known as the basic principle of the thermodynamics of information processing or Landauer's principle. His work and other contributions are summarized in [Fey96], where the first conclusion is that the amount of information in a message is proportional to the free energy required to reset the tape to zero. In this way, some energy is necessary to reset a tape with "surprise" bits, but it is interesting to realize that a reset tape also contains energy. Bennet [Ben82] designed a machine that uses such tapes with information as fuel. The tape after that is randomized, full of information and again, some work must be done to reset it. For a detailed study of all these topics, please see the references mentioned in this section. Also interesting is the work in [Ben03], where Landauer's principle is revised and the historic arguments against it are refuted.

The second conclusion in [Fey96] is that ideally, it is possible to operate a computer without any loss of energy. This computation should be done in a reversible computer infinitesimally slowly. The only entropy loss comes in the resetting process for the next operation and does not depend on the complexity of the computation but on the number of output bits *N*:

$$NkT\ln 2 \qquad \text{(Eq. 2.1)}$$

$kT\ln 2$ is about 3 x 10^{-21}J at room temperature. Unfortunately, the price we must pay for this is that we will never know when the computation is finished.

There is not a minimum amount of energy required to carry out a computation, but there is a limit when the computation is done at a certain speed. In this way, the third point studied in [Fey96] is the amount of free energy required to carry out a computation in a finite time. If we have a reversible computer that goes forward at a rate *r* –it is *r* times more likely to make a forward computation than a backwards one- then the minimum energy that must be expended per computational step is

$$E_{s,\min} = kT \ln r \qquad \text{(Eq. 2.2)}$$

The smaller is *r*, the lower the energy. With some mathematical development, time can be the variable:

$$energy\_loss\_per\_step = kT \frac{\min time\_taken\_per\_step}{time\_per\_step\_actually\_taken} \qquad \text{(Eq. 2.3)}$$

Again, if the computation is infinitesimally slow, there is no loss of energy.

2.1.2 Sources of Power Dissipation

Beyond the thermodynamic arguments in the previous sections, it is clear that an efficient technology for digital circuit materialization from the power consumption point of view must dissipate the lowest energy possible when some computation is actually performed, and no energy in any other case. This occurs in CMOS circuits (with slight differences with the ideal case) and other modern technologies. Older technologies, such as vacuum tubes and relays dissipate relatively huge amounts of energy –even compared with the CMOS technology, that dissipates relatively enormous amounts of energy compared with the thermodynamic lower bounds- doing some computation or not. Power dissipation in CMOS circuits is caused by three main sources [Ped97]:

1. **Leakage** current which is primarily determined by the technology used in its construction, and consists of:
    - Reverse bias current in the parasitic diodes formed between source and drain diffusions and the bulk region in a MOS transistor.
    - Sub-threshold currents that arise from the inversion charges that exists at the gate voltages below the threshold voltage.

    This is also known as static power consumption. In older technologies, with minimum feature size of 0.15 μm or larger, adequate design decisions at the physical level may reduce this first source of power dissipation to very low values. However, recent work like [Kao02] suggest that it may represent over 40% of total power at the 70 nm technology. It is also true that, leakage power is proportional to the number of transistors in the off state and FPGAs requires more transistors to implement a logic function than ASICs.

Nevertheless, all these forecasts about power consumption can be interpreted more as a problem statement than a possible future prediction. For example, using triple-oxide technology [Kle05], the overall static power in Virtex-4 devices with 90 nm process is reduced compared to Virtex-II Pro devices with 130 nm process.

2. **Short-circuit** current which is due to the DC path between the supply rails during output transitions,
3. **Switching** current: it is dissipated when capacitive loads are charged and discharged during logic changes.

Ideally, a CMOS circuit dissipates no static power since in the steady state there is no direct path from V_{dd} to ground. Nevertheless, the MOS transistor is not a perfect switch and there will always be parasitic currents. Until now the static current had little effect on the overall power consumption. However, [Li03] found FPGA architectures (with more than 4 inputs in the LUTs) where leakage power emerges as a mayor source of power dissipation in devices using the projected 0.10 um technology.

The short-circuit power consumption, for example in an inverter gate, depends on the gain of the inverter, the supply voltage, the device threshold, the input rise/fall time and the operating frequency. The maximum short-circuit current flows when there is no load; this current decreases with the load.

From Xilinx and Altera datasheets, short-circuit power is 10% of dynamic power. If, however, design for high performance is taken to the extreme where gates with large fanout are used to drive relatively small loads, then there will be an excessive penalty in terms of short-circuit power consumption.

The dominant source of power dissipation is the switching power dissipation and is given, for a circuit node, by:

$$P_i = 0.5 \cdot C \cdot V_{dd}^2 \cdot E(sw) \cdot f_{clk} \qquad \text{(Eq. 2.4)}$$

Where:

C is the physical capacitance seen by the gate under consideration,

V_{dd} is the supply voltage,

$E(sw)$ (referred as the switching activity) is the average number of transitions in the circuit per $1/f_{clk}$ time, and

f_{clk} is the clock frequency.

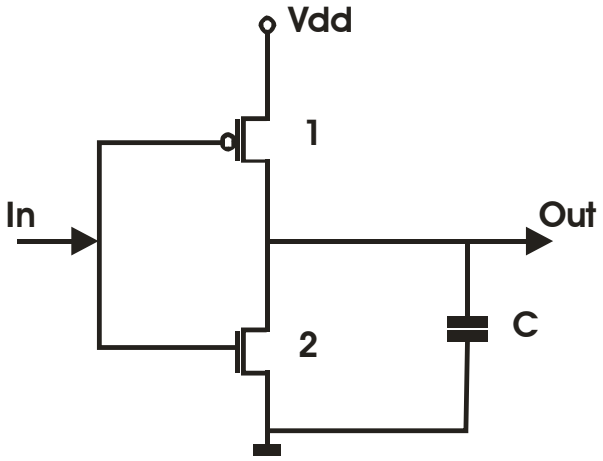


Fig. 2.1. Dynamic power consumption in a CMOS inverter

For a $0 \rightarrow V_{dd}$ transition, switch 1 is closed (Fig. 2.1), an energy $E_{0\rightarrow 1} = C * V_{dd}^2$ is drawn from the power supply V_{dd}, and the energy $E_C = ½\ C * V_{dd}^2$ is saved in the capacitance C. The other $½\ C * V_{dd}^2$ is dissipated in transistor 1.

For a $V_{dd} \rightarrow 0$ transition, switch 2 is closed, no energy is drawn from V_{dd}, but the energy previously stored in C is dissipated [Guy98] in transistor 2.

2.1.2.1 Extending the Dynamic Power Formula

Firstly, in [Li03] a simple model is proposed in order to consider the short-circuit power within Eq. 2.2. This component also depends on the switching activity. It can be assumed that the ratio between short-circuits and switching power, R_{sc} is a constant. In this way, an effective capacitance is defined as follows:

$$\hat{C} = C(1 + R_{sc}) \qquad \text{(Eq. 2.5)}$$

$\hat{C}$ is the total equivalent capacitance connected to the output of the gate under consideration. In this way, the short-circuit component can be integrated together with the charging and discharging of the node capacitances. These two power components are referred to as dynamic power dissipation.

Another point to consider is that Eq. 2.2 is obtained for a CMOS inverter, but the same results can be dragged for other logic gates and MOSFET based circuits. The only difference between the inverter and other CMOS gates, in order to calculate the load capacitance, is the number of transistors in each complementary part (Fig. 2.2).

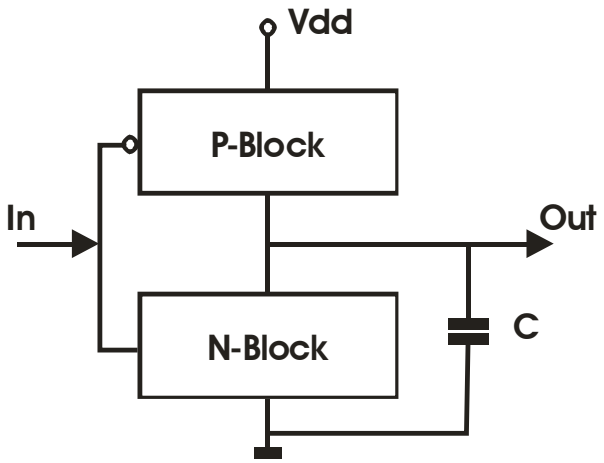


Fig. 2.2: Dynamic power consumption in a generic CMOS gate

For the whole circuit, the power can be calculated adding up all the contributions:

$$P = \frac{1}{2} V_{dd}^2 f_{clk} \sum_i \hat{C}_i E(sw)_i \qquad \text{(Eq. 2.6)}$$

It should be noted that in some work in this area, the 0.5 factor does not appear in the formula. In these cases the switching activity is replaced by the effective frequency. The effective number of signal cycles doubles the number of signal transitions.

The last point studied in this section, related to Eq. 2.3, is that it only considers full swings between V_{dd} and GND. Short glitches have partial swings and are considered by $\hat{E}(sw)_i$, the effective switching activity.

$$P = \frac{1}{2} V_{dd}^2 f_{clk} \sum_i \hat{C}_i \hat{E}(sw)_i \qquad \text{(Eq. 2.7)}$$

Details of $\hat{E}(sw)_i$ and $\hat{C}_i$ estimation are explained in Chapter 5.

2.2 Power Consumption in FPGAs

The previous section exposes the three variables, and degrees of freedom, inherent in the low-power design space: voltage, physical capacitance, and data activity. Because of the quadratic relationship to the power, voltage reduction offers the most effective means to minimize power consumption. Furthermore, this power reduction has a global effect, experienced not only in one gate or circuit node, but throughout the sub-circuit or device supplied with the same voltage. However, programmable logic devices are studied in this work. Once a specific commercially available device is selected, the nominal power supply voltages are given in the data sheets and only capacitance and switching activity need to be estimated (and optimized).

FPGAs consume much more power than ASICs because they have a large number of transistors per logic function in order to program the device. Nevertheless, programmability is the essence of this technology and this overhead must be assumed. In this section the different electronic components of a SRAM-based FPGA are analyzed in order to determine whether or not Eq. 2.4 can be applied to all the nodes in any design.

Most of the models used to explain the power consumption behavior of SRAM-based FPGAs are based on the equations derived from the analysis of the CMOS inverter. As it was said before, an efficient technology would dissipate the lowest energy when some computing is actually performed, while no energy is dissipated in any other case. SRAM-based FPGAs, like the ones used in this work as technological framework, have pure CMOS circuits but also pass-transistor structures, SRAM, buffers, input and output circuits [Gar00].

As it is presented in [Rab96b] (See chapter 3 by C. Svensson and D. Liu), the combinational CMOS static logic is the selected technology for low power. Though, for timing control in synchronous circuits, simple, non-precharged, dynamic flip-flops, or static gate based flip-flops appear to be the best suited techniques. It is important to note that, in the case of flip-flops, there is a component of the dynamic power consumption that does not depend on the input activity and thus behaves like static power consumption. This is the power consumed by transistors clocked at their gates. The power consumption for a non-precharged TSPC flip-flop is:

$$P_d = (8C_i + 4C_o + 4(\alpha/2)C_i + 8\alpha C_o)\, fV_{dd}^2 \qquad \text{(Eq. 2.8)}$$

The first two terms do not depend on the input activity. C_i and C_o are respectively the input and output capacitances at the transistors and α is the data activity.

Another problem found in logic circuits and in particular in FPGAs, comes from the high capacitance nodes where drivers are used to decrease the delay and the short-circuit power consumption due to long rise and fall times in the following stages. As shown in [Rab96b], using a tapered inverter chain, and minimizing the delay, the driver causes an excess power consumption of 80% over the load.

2.2.1 Programmable Routing

[Bet99] describes two important circuits in the design of FPGA routing switches: pass transistors and tri-state buffers. Routing switches are either pass transistors or pairs of tri-state buffers (one in each direction), and allow routing wire segments to be joined to form longer connections (Fig 2.3). Multiplexers allow routing wires to be connected to the input pins of logic blocks, while demultiplexers (a set of pass transistors) allow routing wires to be driven by output pins of logic blocks (Fig. 2.4).

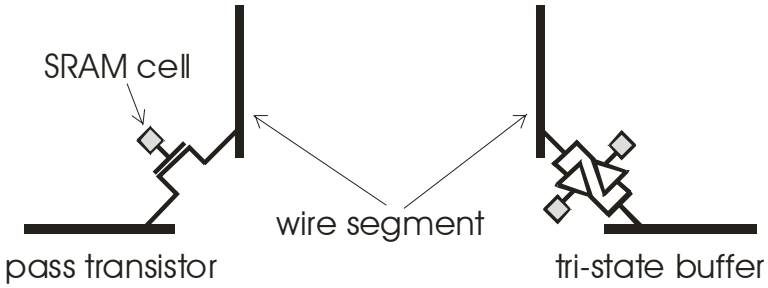


Fig. 2.3: Routing Switch

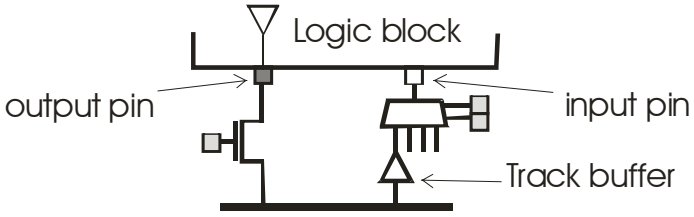


Fig. 2.4: Logic Block Routing

Pass transistors connecting different wire segments can be modeled by equivalent resistances and capacitances. In this way, it is possible to lump together the capacitances of wire segments and pass transistors in a net or node. In other words, these transistors are considered part of the wire. Buffers can be treated as logic cells and the wires, including pass transistors, are driven by these buffers. For example, Fig. 2.5 shows a net composed by several wire segments and pass transistors from buffer A to buffer B.

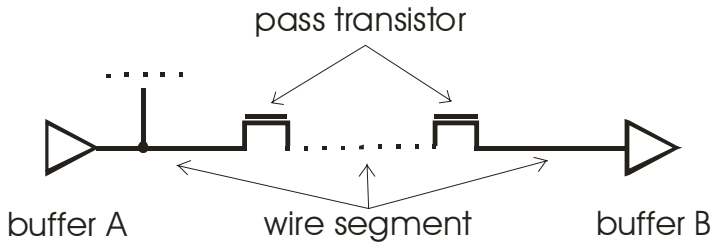


Fig. 2.5: Net or node model

2.2.2 Physical Capacitance

Interconnection plays a prominent role in determining the total chip area, delay and power, and hence, must be accounted for as early as possible during the design process. In the particular case of FPGAs, the long routing tracks, with significant capacitance, consumes relatively a lot of power for every transition. For example, [Poo02] [Poo05] found for theoretical models that 57% of the total energy consumption is due to connections between the logic clusters.

Power dissipation is linearly dependent on the physical capacitances driven by individual gates. So, once a design is mapped, placed and routed in a specific technology, capacitance calculation could be easily done using information from the target library. Unfortunately, this is not the case for commercial FPGAs: often, manufacturers do not provide the information about internal nodes capacitance or at least they do not give it directly. This makes mandatory the development of a solution in this thesis for the capacitance retrieval problem and it is presented in Chapter 6.

2.2.3 Switching Activity

In addition to voltage and physical capacitance, switching activity is the third factor that determines the dynamic power consumption. A chip may contain a high amount of physical capacitance, but if there is no switching in the circuit, then no dynamic power will be consumed. In a combinational circuit, if two consecutive and identical vectors are presented at the circuit inputs, no power is dissipated. The data activity determines how often this switching occurs. There are two components to the switching activity:

1. f_{clk} which determines the average periodicity of data arrivals, and
2. *E(sw)* which determines how many transitions each data arrival will generate.

F_{clk} and *E(sw)* are strongly related. F_{clk} can not be unlimitedly increased. The corresponding signal must have enough time, $1/f_{clk}$, to reach the steady state before the arrival of the new input vector.

For circuits that do not experience glitching, *E(sw)* can be interpreted as the probability that a power consuming transition will occur during a single data period. Even for these circuits, calculation of *E(sw)* is difficult as it depends not only on the switching activities of the circuit inputs and the logic function computed by the circuit, but also on the spatial and temporal correlations among the circuit inputs.

For certain design styles, glitching can be an important source of signal activity. Glitching refers to spurious and unwanted transitions that occur before a node settles down to its final steady-state value. Glitching often arises when paths with unbalanced propagation delays converge at the same point in a circuit. Since glitching can cause a node to make several unnecessary power consuming transitions, it should be avoided whenever possible [Boe95].

The data activity *E(sw)* can be combined with the physical capacitance *C* to obtain switched capacitance, $C_{sw} = C.E(sw)$, which describes the average capacitance charged during each data period $1/f_{clk}$. It is a useful magnitude for comparing implementations running at different clock frequencies and with different voltages.

2.3 Switching Activity Computation

The computing of switching activity in a logic circuit is difficult because it depends on a number of parameters. Some of these parameters are technology-dependent factors and will be treated below. The input pattern dependence, the delay model at each design stage, the circuit logic function and, for some techniques, the circuit structure, are not technology-dependent factors. The impact of these factors on the circuit node activity will be illustrated in the following sections.

2.3.1 Dependence on the Input Patterns

N	Input I	Input J	Output	Tr
1	0-0	0-0	0-0	N
2	0-0	0-1	0-0	N
3	0-0	1-0	0-0	N
4	0-0	1-1	0-0	N
5	0-1	0-0	0-0	N
6	0-1	0-1	0-1	Y
7	0-1	1-0	0-0	N
8	0-1	1-1	0-1	Y
9	1-0	0-0	0-0	N
10	1-0	0-1	0-0	N
11	1-0	1-0	1-0	Y
12	1-0	1-1	1-0	Y
13	1-1	0-0	0-0	N
14	1-1	0-1	0-1	Y
15	1-1	1-0	1-0	Y
16	1-1	1-1	1-1	N

Table 2.1: Activity for an AND gate with independent inputs

For example, consider a two-input AND gate *g* with independent inputs *I* and *J* whose signal probabilities are ½, then *Eg (sw)* = 3/8. This holds because in 6 out of 16 possible input transitions, the output of the two-input AND gate makes a transition as is shown in Table 2.1.

Now suppose that it is known that only patterns 00 and 11 can be applied to the gate inputs and that both patterns are equally probable, then $E_g(sw)$=1/2 (Table 2.2).

N	Input I	Input J	Output	Tr
1	0-0	0-0	0-0	N
2	0-1	0-1	0-1	Y
3	1-0	1-0	1-0	Y
4	0-0	1-1	0-0	N

Table 2.2: Activity for an AND gate with spatial dependence among the inputs

Alternatively, if one assumes that it is known that every 0 applied to input *I* is immediately followed by a 1, while every 1 applied to input *J* is immediately followed by a 0, then $E_g(sw)$ = 4/9 (Table 2.3).

N	Input I	Input J	Output	Tr
1	0-1	0-0	0-0	N
2	0-1	0-1	0-1	Y
3	0-1	1-0	0-0	N
4	1-0	0-0	0-0	N
5	1-0	0-1	0-0	N
6	1-0	1-0	1-0	Y
7	1-1	0-0	0-0	N
8	1-1	0-1	0-1	Y
9	1-1	1-0	1-0	Y

Table 2.3: Activity for an AND gate with temporal dependence among the inputs

Finally, if one assumes that it is known that *I* changes whenever *J* changes its value, then $E_g(sw)$ = ¼ (see Table 2.4).

N	Input I	Input J	Output	Tr
1	0-0	0-0	0-0	N
2	0-0	1-1	0-0	N
3	0-1	0-1	0-1	Y
4	0-1	1-0	0-0	N
5	1-0	0-1	0-0	N
6	1-0	1-0	1-0	Y
7	1-1	0-0	0-0	N
8	1-1	1-1	1-1	N

Table 2.4: Activity for an AND gate with spatial-temporal dependence among the inputs

The first case is an example of spatial correlations between gate inputs; the second case illustrates temporal correlations; while the third case describes an instance of spatial-temporal correlations.

In general there are first order and higher order temporal correlations. In the first case the next value of a signal depends on its current value. In the second case it also depends on the *n* previous values.

There are also special names for some types of correlations for internal signals. Spatial, temporal and spatial-temporal correlations at state lines, induced by a finite state machine, are known as sequential correlations. Even if primary inputs are uncorrelated, the state lines can be strongly correlated. Another interesting case of spatial correlation in internal signals is due to reconvergent fanout known as structural correlations. Reconvergent nodes are explained below in this Chapter. A very interesting study of the effects of correlations on power estimation methods is presented in [Sch96a].

With the previous examples, it is clear that the straightforward approach of estimating power just by using a simulator and applying a big but arbitrary set of input

patterns may give erroneous results due to this pattern-dependence problem. Experiments that quantify this fact are presented in this thesis.

It is clearly unfeasible to estimate the power consumption by exhaustive simulation of the circuit. Even for a combinational circuit with *n* inputs, it is not enough to apply the 2^n combinations because the activity depends on the node state after the last applied vector. In the restrictive case of uniform distribution, the number of combinations is 2^{2n}. Some techniques have been proposed to overcome this difficulty by using probabilities that describe the set of possible logic values at the circuit inputs. Some mechanisms to calculate these probabilities for gates inside the circuit have also been proposed. Alternatively, exhaustive simulation may be replaced by Monte-Carlo simulation with well-defined stopping criterion for specified relative or absolute error in power estimates and a given confidence level [Naj98]. A survey of activity estimation techniques will be presented in Chapter 3.

2.3.2 Delay Model

Any power estimation techniques must account for **s**teady-state transitions (which consume power and are necessary to perform a computational task). Based on the used delay model also the glitches could be considered (which dissipate power without doing any useful computation). Sometimes, the first component of power consumption is referred to as the functional activity while the latter is referred to as the spurious activity. It is shown in Chapter 5 that the average number of transitions per clock cycle in a combinational multiplier reaches high values in some nodes. The spurious power dissipation may be more significant in FPGAs than in ASICs because of the relative importance of the nets [Sha02].

Current power estimation techniques often handle both zero-delay (non-glitch) and real delay models. In the first model, it is assumed that all changes at the circuit inputs propagate through the internal gates of the circuits instantaneously. The latter model assigns a finite delay to each gate in the circuit and can thus account for the hazards in the circuit. A real delay model, post P&R, increases the computational requirements of the power estimation techniques while improving the accuracy of the estimates. On the other hand, support for the zero-delay models is useful for power estimation in early stages of the design process. Furthermore, between these two simulation models,

there are others coming from different points in the design flow (post synthesis, technology mapping, and place). The closer the simulation model is to the post P&R version, the more accurate could be the estimation.

The computing of spurious activity requires careful logic and circuit level characterization of the gates in a library as well as detailed knowledge of the circuit structure. This means that different results will be obtained if the estimation is done using a model generated before the technology mapping, when no technological data may be taken into account and no timing information is available; or after the technology mapping, when timing information is available just for the logic but not for the nets; or after the place and route, when a complete timing information is available.

VHDL users know how to write abstract, technology independent descriptions, but now it is necessary to simulate the actual hardware. How can such a simulation be done? The answer is VITAL (IEEE 1076.4 standard) [VIT01]. The VITAL (VHDL Initiative Towards ASIC Libraries) is a modeling specification that defines a methodology which promotes the development of highly accurate, efficient simulation models for ASIC components in VHDL.

2.3.2.1 The IEEE VITAL Standard

The way to describe “physical” hardware in VHDL is to write VHDL models of those components. This is supported in VHDL through the use of instantiation. Historically, gate-level simulation using VHDL has been notoriously slow. This led to the creation of the 1076.4 working group to provide a mechanism to allow fast gate-level simulation using VHDL. Their effort became known as the VITAL standard. VITAL is not an issue for VHDL designers, but an EDA vendor/ASIC supplier issue. A simulator is VITAL compliant if it implements the VITAL package in its kernel.

The FPGA vendor’s library elements need to be implemented entirely in VITAL primitives. They also provide tools that generate these VHDL models from post map, P&R, etc. proprietary files. Also note that, with the VHDL model, a SDF (Standard Delay Format) file [SDF01] is generated. The SDF file contains timing data and the VITAL compliant simulator, having implemented an SDF reader, directly imports it into the simulator. The naming conventions and types of VITAL generics provide the placeholders to load timing data via back-annotation.

Although an SDF file specifies delays as min:typ:max values, only one of these values will be used for back-annotation. The selection of the specific delay values (min, typ or max) could be done by the back-annotation program under a user controlled option.

2.3.3 Logic Function

In the first place, switching activity at the output of a logic gate depends on the Boolean function of the gate itself. For example, under the assumption that the input signals are uncorrelated, switching activity at the output of a two-input NAND or NOR gate is 3/8 and at the output of a two-input XOR gate is ½ (see Table 2.5).

N	Input I	Input J	Output	NAND	NOR	XOR
1	0-0	0-0	0-0	N	N	N
2	0-0	0-1	0-0	N	Y	Y
3	0-0	1-0	0-0	N	Y	Y
4	0-0	1-1	0-0	N	N	N
5	0-1	0-0	0-0	N	Y	Y
6	0-1	0-1	0-1	Y	Y	N
7	0-1	1-0	0-0	N	N	N
8	0-1	1-1	0-1	Y	N	Y
9	1-0	0-0	0-0	N	Y	Y
10	1-0	0-1	0-0	N	N	N
11	1-0	1-0	1-0	Y	Y	N
12	1-0	1-1	1-0	Y	N	Y
13	1-1	0-0	0-0	N	N	N
14	1-1	0-1	0-1	Y	N	Y
15	1-1	1-0	1-0	Y	N	Y
16	1-1	1-1	1-1	N	N	N

Table 2.5: Activity for different logic gates

Indeed, switching activity at the output of a *K*-input NAND or NOR gate approaches $½^{K-1}$ for large *K* whereas that for a *K*-input XOR gate remains at ½. The proposition for a *K*-input NAND gate can be demonstrated as follows.

As mentioned, the number of input vector combinations, when activity is studied at a gate or circuit output, is 2^{2K}, being *K* the number of primary inputs. In order to analyze a *K*-input NAND gate, all the combinations can be arranged in groups. In each group the first *K*-input vector is kept fixed, and for the second *k*-input vector has 2^K combinations. In all but one group there is just one case where a 1 to 0 transition is generated, when the second vector is formed by all 1's. The exceptional group is the one with the fixed vector with all 1's, where the possible transition is from 0 to 1. This happens in all the cases in the group except when the second vector is also the one formed by all 1's, keeping the gate output at logic 0.

Then, there are 2^K–1 groups with one transition, and one group with 2^K–1 transitions. The transition probability for the NAND gate where the inputs are independent is:

$$P(NAND_K) = \frac{(2^K - 1) + (2^K - 1)}{2^{2K}} = \frac{2^{K+1} - 2}{2^{2K}} \qquad \text{(Eq. 2.9)}$$

If *K* is big enough the second constant term can be neglected, then:

$$P(NAND_K) \cong \frac{2^{K+1}}{2^{2K}} = 2^{K+1-2K} = 2^{-K+1} = \frac{1}{2^{K-1}} \qquad \text{(Eq. 2.10)}$$

The demonstration for the *K*-input NOR gate can be developed in the same way.

2.3.4 Circuit Structure

If probabilistic techniques are used to estimate the switching activity, probabilities are calculated and propagated from primary inputs to the inner nodes and finally, to the circuit outputs. But dependencies among the inputs complicate probability calculations. Although primary inputs were supposed uncorrelated other dependencies originated on the circuit structure remain: the reconvergent nodes, circuit nodes that receive inputs from two paths connected to some gate output (Fig. 2.6). If a network consists of simple gates and has no reconvergent fan out nodes, then the exact switching activities can be computed during a single post-order traversal of the network [Ped94]. For

networks with reconvergent nodes, the problem is much more challenging, as internal signals may become strongly correlated and exact consideration of these correlations cannot be performed with reasonable computational effort or memory usage. Current power estimation techniques either ignore these correlations or approximate them, thereby improving the accuracy at the expense of longer run times. Exact methods (i.e., symbolic simulation) have also been proposed, but are impractical due to excessive time and memory requirements.

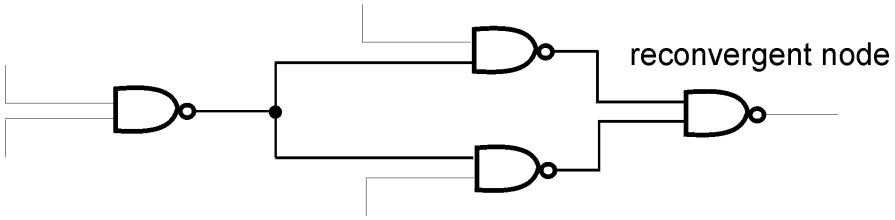


Fig. 2.6: Example of a reconvergent node

2.3.5 Technology-dependant Factors

In actual networks, statistical perturbations of circuit parameters may change the propagation delays and produce changes in the number of transitions because of the appearance or disappearance of glitches. For that reason it is useful to determine the change in the signal transition count as a function of these statistical fluctuations.

Variation of gate delay parameters may change the number of glitches occurring during a transition as well as their duration. In this way, the spurious component of power dissipation is sensitive to IC parameter fluctuations [Ben94].

2.4 Conclusions

The need for lower power systems is crucial in electronic applications from portable devices to high-end computers. Nevertheless, designing for low power adds another dimension to the already complex VLSI design problem: the design has to be optimized for power as well as for performance and area.

Optimizing these three axes necessitates a new generation of EDA tools at all design phases. These power aware tools and methodologies include power estimation tools. Behavioral synthesis, logic synthesis and layout optimization tools require

accurate and efficient estimation of the power consumption of alternative implementations.

There are several sources of power consumption in CMOS circuits (Fig. 2.7) but the dynamic power is the main component. In order to estimate the dynamic power consumption, both activity and capacitance must be gauged. Activity is hard to estimate because its dependence on the input patterns (known as the pattern-dependence problem). Nevertheless, the capacitance recovery is a specific design problem for commercial FPGAs because of the lack of these data or any direct information about how to calculate the capacitance at each circuit node.

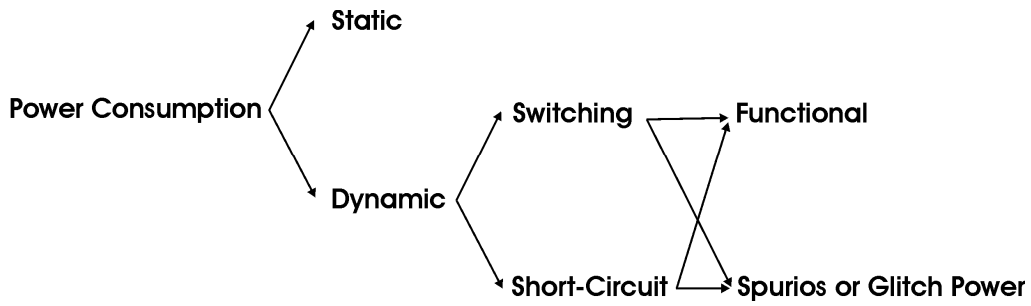


Fig. 2.7. Sources of power consumption in CMOS circuits and FPGAs

References

[Ben03] Charles H. Bennett, “Notes on Landauer’s Principle, Reversible Computation and Maxwell’s Demon”, Studies in History and Philosophy of Modern Physics, v. 34, pp. 501-510, 2003.

[Ben82] C.H. Bennett, “The Thermodynamics of Computation – a Review” Internat. J. Theoret. Phys. 21, pp. 905-940 (1982).

[Ben94] L. Benini, M. Favalli, and B. Ricco, “Analysis of hazard contribution to power dissipation in CMOS IC’s”. In Proceedings of the 1994 International Workshop on Low Power Design, pp 27-32, April 1994.

[Bet99] Vaughn Betz and Jonathan Rose, “Circuit Design, Transistor Sizing and Wire Layout of FPGA Interconnect”, IEEE Custom Integrated Circuits Conference, 1999.

[Boe95] Boemo, E., Gonzalez de Rivera, G., Lopez-Buedo, S., Meneses, J., “Some Notes on Power Management on FPGAs”, LNCS, No. 975, Springer-Verlag, Berlin (1995) 149-157.

[Fey96] Richard P. Feynman, “Feynman Lectures on Computation”, Ed. A.J.G. Hey and

R.W. Allen. Addison-Wesley, 1996.

[Gar00] Andrés David García García, “Etude sur l’Estimation et l’Optimisation de la consommation de puissance”, PhD Thesis, l’Ecole Nationale Supérieure des Télécommunications, Paris, 2000.

[Guy98] Alain Guyot and Sélim Abou-Samra, “Low Power CMOS Digital Design”, In proc. Of International Conference on Microelectronics 1998 (ICM’98), Monastir, Tunisia, December 1998.

[ITRS04] ITRS Technology Working Group, “Overall Roadmap Technology Characteristics (ORTC)”, from the International Technology Roadmap for Semiconductors (ITRS). 2004 Upgrade. Available at http://public.itrs.net

[Kao02] James Kao, Siva Narendra, Anantha Chandrakasan, “Subthreshold leakage modeling and reduction techniques”, In proc. of the 2002 IEEE/ACM international conference on Computer-Aided Design, pp. 141-148, 2002

[Kle05] M. Klein, “The Virtex-4 Power Play”, Xcell Journal, Spring 05

[Lan61] R. Landauer, “Irreversibility and Heat Generation in the Computing Process”, IBM Journal of Research and Development, Vol 5, N 3, pp. 261-269, 1961.

[Lan94] P. Landman, Low-Power Architectural Design Methodologies, Ph. D. Thesis, Electronic Research Laboratory, University of California, Berkeley, August 1994.

[Li03] Fei Li, Deming Chen, Lei He, Jason Cong: “Architecture evaluation for power-efficient FPGAs”, Proc. Of Int. Symp on Field Programmable Gate Arrays, 2003, pp. 175–184

[Naj98] F. N. Najm and M. G. Xakellis, “Statistical estimation of the switching activity in VLSI circuits”, VLSI Design, vol. 7, no. 3, pp. 243-254, 1998.

[Ped94] M. Pedram, "Power estimation and optimization at the logic level," Int'l Journal of High Speed Electronics and Systems, Vol. 5, No. 2 (1994), pp. 179-202.

[Ped97] M. Pedram, “Design technologies for Low Power VLSI”, In Encyclopaedia of Computer Science and Technology, Vo. 36, Marcel Dekker, Inc., 1997, pp. 73-96.

[Poo02] Kara K.W. Poon, Andy Yan, Steven J.E. Wilton, “A Flexible Power Model for FPGAs”, LNCS, Volume 2438, Jan 2002, pp. 312-321.

[Poo05] Kara K.W. Poon, Steven J.E. Wilton, and A. Yan, “A Detailed Power Model for Field-Programmable Gate Arrays,” ACM Transactions on Design Automation of Electronic Systems (TODAES), vol. 10, issue 2, pp. 279-302, April 2005.

[Rab96b] Jan M. Rabaey and Massoud Pedram. “Low power design methodologies”. Boston, Kluwer Academic, 1996.

[Sch96a] P. Schneider and S. Krishnamoorthy. “Effects of correlations on accuracy of power analysis - an experimental study”, International Symposium on Low Power Electronics and Design, Monterey, California, United States, 1996, pp. 113-116.

[SDF01] IEEE Std 1497-1999, IEEE Standard for Standard Delay Format (SDF) for the Electronic Design Process. The Institute of Electrical and Electronics Engineers, Inc. 3 Park Avenue, New York, NY 10016-5997, USA, 2001.

[Sha02] L. Shang, A. S. Kaviani, K. Bathala, “Dynamic Power Consumption in Virtex-II FPGA Family”, FPGA 2002 Monterey, California, USA, February 24-26, 2002,

pp. 157-164.

[Sut05] Gustavo Sutter, “Aportes a la Reducción de Consumo en FPFAs”, Ph. D. Thesis, Departamento de Ingeniería Informática, Escuela Politécnica Superior, Universidad Autónoma de Madrid, April 2005.

[VIT01] IEEE Std 1076.4-2000, IEEE Standard for VITAL ASIC (Application Specific Integrated Circuit) Modelling Specification. The Institute of Electrical and Electronics Engineers, Inc. 3 Park Avenue, New York, NY 10016-5997, USA, 2001.

Chapter 3.

"If it (the author refers to questionable estimations about power consumption of computer equipment in the USA) were correct , we are approaching a notable inflection point in human affairs, where we expend as much effort in moving information as we do in moving matter." From "The Computer and the Dynamo" by Brian Hayes [Hay01]

3 Power Estimation Techniques

The problem of design for low power cannot be achieved without accurate power prediction and optimization tools. Power dissipation must be estimated as soon as possible and during all the design process –in particular within an optimization loop- to meet the power budget without having to go through a costly redesign effort. When designing the corresponding PCB, the power consumed by the devices needs to be estimated as accurately as possible to design the power supplies, voltage regulators, heat sink and cooling system. Since FPGAs provide short design cycles and a fast time-to-market, the PCB is usually designed at the same time as the logic for the FPGAs. It means power estimation should de done as soon as possible for FPGA.

Several techniques for VLSI power estimation that deal with the pattern-dependence problem were proposed some years ago [Gho92] [Naj94]. Nevertheless, the problem is not completely solved yet, even at the gate level. Due to the computational complexity of power estimation, high accuracy and short run-time cannot be met together. This problem is observed for average power estimation in individual gate power estimation; and for total average power consumption in large sequential circuits [Koz01].

There are some issues to be considered together with accuracy, in order to evaluate the techniques in practice:

- As with all the design restrictions, the power budget must be considered in all the design stages, especially in the earlier ones. In this way, it must be provided with the design specifications. In other words, there should be estimation tools from the beginning of the design stages. If a power consumption problem is detected later, all the known drawbacks found in engineering could happen when deviations from the specification are discovered later.
- If the technique is not useful for the large circuits found in today and future practice, it is not effective because these circuits are generally the ones that have the highest power consumption.
- The switching activity is the most difficult factor to obtain in the power consumption equation. The strong dependency on input patterns could be a problem considering the lack of this information within the design process, especially in the first stages.

In this Chapter, various techniques for power estimation at the circuit, and logic level are reviewed. These techniques are classified according to the approach they are based on: probability or statistics. The design level and the type of circuits they could be applied to (combinational, sequential) are clearly mentioned when necessary. In addition, the first section groups together earlier techniques that produce power values but do not consider the pattern-dependence problem and could not be applied to current (with more than a few hundred gates) designs.

In the FPGA world, the techniques that operate over logic-level descriptions are very useful because these descriptions can be easily obtained from the RTL ones through synthesis. Other approaches try the power estimation problem starting from high-level descriptions [Buy05] in order to use them within power-aware EDA tools. However, this work is mainly focused on techniques based on logic-level descriptions due to two reasons. Techniques based on high-level descriptions also present high levels of error because these techniques will never have the necessary technological details to obtain precision and; as mentioned above, techniques developed for logic-level descriptions

can be used starting from synthesizable RTL descriptions that could be considered a high-enough level of abstraction. In Chapter 4, the implementation of a statistical-based approach at the logic-level is presented, and its details are given in Chapters 5 and 6.

3.1 Power Estimation History –or Simulative Approaches

3.1.1 SPICE-like Circuit Simulation

The simplest techniques for power estimation are based on circuit simulation, where the circuit is evaluated with a representative set of input vectors. They are accurate and capable of handling various technologies, different circuit design styles, single and multi-phase clocking methodologies, tri-state drives, etc. However, this technique experiences two serious problems. The model size for large, cell-based designs leads to efficiency problems: memory and execution time became strong constraints for its applicability. However, the most important problem is the size of the stimulus vector set necessary to calculate accurately the activity. As was mentioned in Chapter 2, the number of possible combinations at primary inputs is 2^{2n}. Still running this huge simulation, the real distribution of the stimuli and its correlations should be considered. This is known as the pattern-dependence problem.

The first problem, the efficiency, can be tackled simplifying the model as is shown below. On the other hand, to solve the pattern-dependence problem two approaches have been developed. The first approach is statistical and the second is based on probabilities.

In the next sub-sections, several simulation-based techniques are presented sorted in an increasing order of efficiency (but decreasing order of accuracy).

3.1.1.1 Timing Simulators

The efficiency can be improved based on simplified table-driven device models, circuit partitioning and single-step nonlinear iteration but with some inaccuracy in modeling leakage effects. PowerMill of Epic Design Technology [Den94] is a transistor-level power simulator and analyzer, which applies an event-driven timing simulation algorithm to increase the speed by two to three orders of magnitude over SPICE.

3.1.1.2 Switch Simulators

The transistor model can be further simplified to a simple resistive switch using a discrete data representation (0, 1, X, for example). Switch-level simulation techniques are in general much faster than circuit-level simulation techniques, but are not as accurate. Examples of switch-level simulators are IRSIM or the IRSIM-CAP simulator [Lan94], which is a modification of IRSIM.

3.1.1.3 Gate Level Simulators

These simulation programs rely on the accuracy of the macromodels built for the gates in the ASIC library as well as gate-level timing analysis. The accuracy depends heavily on the quality of the macromodels, the glitch filtering scheme used and the accuracy of physical capacitances provided at the gate level. The speed is 3-4 orders of magnitude faster than SPICE.

In [Büh00], using these simulators, a gate level switching activity estimation is presented based on bit-parallel simulation first published in [Sch95]. The first optimization is made executing several bitwise operations in parallel within the same processor instruction. This algorithm also offers some improvements in memory management and data structures.

3.1.1.4 Hierarchical Simulation

The idea is to use a hierarchy of power simulators (for example, at architectural, gate-level and circuit-level) to achieve a reasonable accuracy and efficiency tradeoff. For example, see the Entice-Aspen tool [Geo94]. This power analysis system consists of two components: Aspen, which computes the circuit activity information, and Entice which computes the power characterization data. A stimulus file must be supplied to Entice where power and timing delay vectors are specified. The set of power vectors discretizes all possible events in which power can be dissipated by the cell. With the relevant parameters set according to the user's specifications, a SPICE circuit simulation is invoked to accurately obtain the power dissipation of each vector. During logic simulation, Aspen monitors the transition count of each cell and computes the total power consumption as the sum of the power dissipation for all cells in the power vector path.

3.2 Statistical Approaches

3.2.1 Monte Carlo Simulation

Although the techniques shown in this section make use of standard simulators, the difference to the previous simulative approaches is that in this case the techniques are based on statistics. The first statistical technique for power estimation is a Monte Carlo simulation that minimizes the pattern dependence problem.

The first paper in this direction, for total average power estimation over combinational circuits, is [Bur93]. Other work extended its scope in order to estimate the average power for individual gates ([Xak94] [Naj98]).

The technique proposed in [Bur93] is based on the assumption that

the power consumed by the circuit over a long enough period T has a normal distribution.

It is made up of applying randomly generated input patterns at the circuit primary inputs and monitoring the power dissipation per time interval *T* using a standard simulator. The required number of power samples is expected to be a small fraction of the total number of possible vector combinations. It is calculated in function of a tolerated relative error at a given confidence level which enables the user to tune the accuracy of the measurement. This accuracy requires a given computational effort.

The first point to be noted is that existing simulators can be used in the inner loop of the Monte-Carlo program. It makes the technique easier to implement than the probabilistic ones as is shown below, which require the development of specialized simulators. In addition, the convergence time for this approach could be fast enough to compete with the probabilistic techniques whose main advantage is its speed. This is due to the dimension independence property of Monte Carlo techniques: the number of samples required to make an estimate is independent of the problem size ([Bur93]). In any case, despite the sample size, big designs require more time because a longer simulation time is required to evaluate each single input vector. Another factor that contributes to the speed is the distribution of the overall circuit power: that is generally very nearly Gaussian and very narrow around its mean. Naturally, it will be shown that the lower the required accuracy, the faster the power estimation.

A weakness of the technique presented in [Bur93] is that when the power consumption values on individual nodes are required, the convergence is very slow. In addition, the method does not handle spatial correlations at the circuit inputs. Finally, this statistical technique is directly applicable only to combinational circuits.

The power consumption for individual nodes is solved in [Xak94] [Naj98]. The original approach is extended for sequential circuits in [Sax02], but other statistical techniques are also developed for sequential designs ([Koz01]). The limitation of the spatial correlations at the circuit inputs is not so severe compared with the probabilistic approach, where they are hard to manage even at the internal nodes. For total power, [Sch96a] reports 10-25% of additional error for several circuits when spatial correlations at internal signals are ignored.

3.2.2 Total Power (McPower)

Code 3.1 shows the main loop of the statistical estimation as it was presented in the first version ([Bur93]). Two stages compose each iteration, and both are critical in order to guarantee the correctness of the measurement.

```
1  Repeat
2     // guarantees that typical power is measured
3  Setup;
4  // ensures the correctness of the stopping criterion
5     Sample;
6  // average power is measured
7  Until (Stop (mean_pow, stdev_pow));
```

Code 3.1: Basic Monte Carlo power estimation algorithm

The setup phase serves to guarantee that power values observed at the end of successive intervals are samples of independent random variables as required by the method. The power is not measured in the setup phase. At the end of this phase, the circuit is as if it were operating in a random point of time. The exact application of this phase depends on whether the circuit is purely combinational, or being combinational, if it has registered inputs.

The power values observed during the sample phase are noted down and used to decide when to stop. The duration of this phase is specified by the user, allowing a number of transitions per sampling interval on each input. For every input signal x_i, a

random number generator sets its value with probability $P(x_i)$. Once x_i has switched, another random number generator decides the duration of the new state following the distributions $F_{xi}^{1}(t)$ and $F_{xi}^{0}(t)$, for the high and low values respectively. The probability $P(x_i)$ and distributions $F_{xi}^{1}(t)$ and $F_{xi}^{0}(t)$ are supplied by the user, but it can be simplified asking the user just the average time the input is high and low: μ_{xi}^{1} and μ_{xi}^{0}. For combinational circuits with registered inputs, the duration of this phase would be a discrete multiple of the clock period.

If the power consumed by a circuit over a period T has a distribution that is very close to normal and the successive input patterns are independently generated, then the following stopping criterion can be derived. It can be shown that the number, N, of measurements (sample size) is:

$$N \geq \left(\frac{t_{\alpha/2} s}{p \varepsilon} \right)^2 \qquad \text{(Ec. 3.1)}$$

Where p is the measured power average of the random sample over a period T, s is the standard deviation of the power random sample, (1-α)*100% is the confidence that relative error, ε, in the measurement is less than a specified value. $T_{\alpha/2}$ is obtained from the t-distribution with (N–1) degrees of freedom.

Nevertheless, the major disadvantage of this approach is that it is very slow to provide the power consumed by individual gates or small groups of gates. These estimations are useful to diagnose high consumption problems, find the circuit parts that consume more energy, and reliability estimation. Another important use of this information is to optimize a design for low power at the technology mapping, placement or routing stages. It would take a bigger sample to estimate (with the same accuracy) the power of individual gates, because some gates may switch very infrequently, and as p decreases in Eq. 3.1, N increases. This problem is known as the slow convergence problem.

3.2.3 Power of Individual Gates (MED)

This statistical technique is a direct extension of McPower. It provides both the total and individual-gate power estimates [Naj98], with the same benefits shown above. Preliminary results of this extension were presented in [Xak94].

In order to solve the slow convergence problem for individual gates it is proposed a new stopping criterion. At each iteration, the number of transitions at every node, n_i, is written down and its average n and standard deviation s are calculated. N_i/T is an estimation of the transition density, $D(x_i)$, at node i.

According to the Central Limit Theorem, the average $n = \sum n_i/N$ has a distribution which is close to normal for large N. For $N > 30$, that typically is the minimum number to satisfy near-normality, ε_1 is a sample upper bound of the percentage error,

$$\left|\frac{\eta - \bar{n}}{\bar{n}}\right| \leq \frac{z_{\alpha/2}\, s}{\bar{n}\sqrt{N}} \leq \varepsilon_1$$

so,

$$N \geq \left(\frac{z_{\alpha/2}\, s}{\bar{n}\varepsilon_1}\right)^2 \qquad \text{(Ec. 3.2)}$$

This may also be expressed as the percent deviation from the population mean η:

$$\left|\frac{\eta - \bar{n}}{\bar{n}}\right| \leq \varepsilon_1 \Rightarrow \left|\frac{\bar{n} - \eta}{\eta}\right| \leq \frac{\varepsilon_1}{1 - \varepsilon_1} = \varepsilon \qquad \text{(Ec. 3.3)}$$

Where ε is the user specified error tolerance. Thus Ec. 3.2 provides a stopping criterion to yield the user specified accuracy (Ec. 3.3), with confidence (1 - α) x 100%.

However, the slow convergence problem has not been solved yet. In order to overcome it, a threshold for the average transition count, n_{min}, is defined. This limits the maximum iteration number tolerated by the algorithm. The following modified stopping criterion is proposed for nodes with $n < n_{min}$:

$$N \geq \left(\frac{z_{\alpha/2}\, s}{n_{\min}\varepsilon}\right)^2 \qquad \text{(Ec 3.4)}$$

It is shown in [Naj98] that $n_{min}.\varepsilon$ becomes an absolute error bound characterizing the accuracy for low-density nodes. With confidence (1 - α) x 100%:

$$\left|\bar{n} - \eta\right| \leq \frac{z_{\alpha/2} S}{\sqrt{N}} \leq n_{\min}\varepsilon$$

In short, the circuit nodes have been divided in two sets. For the regular nodes, where $n > n_{min}$, Ec. 3.2 is used as stopping criterion. For low-density nodes, where $n < n_{min}$, it is used Ec. 3.4. In both cases, the stopping criterion is tested after $N > 30$.

Low-density nodes have the least effect on circuit power. Therefore, the above strategy reduces the execution time, with little or no penalty. All the results presented in [Naj98] are also verified in this thesis for circuits implemented on FPGA.

Another point clarified in this second version of the technique is the use of two operation modes, synchronous and asynchronous mode. In the synchronous mode the inputs are generated as if they were part of a synchronous circuit with its inputs coming from registers.

In this way, the specific setup and sample routines are selected according to the operation mode. In the synchronous mode the input pulse widths are multiples of the clock period, *Tc* and a clock cycle is enough for setup. Instead of this, in the asynchronous mode the time between transitions and setup are as explained in [Bur93].

3.2.4 Improvements in Statistical Methods

The problem with the classical statistical estimation method is the execution time. Current big designs could require unacceptable run times when the user specifies medium or high accuracy requirements. In this thesis an improvement for the classical Monte Carlo power estimation method for individual nodes is presented. Equally, some tasks that presented results in this subject are briefly commented in this section

The first approach to solve this slow convergence problem was presented in [Xak94]. As is explained in the previous section, the nodes with less activity than a threshold η_{min} are considered low-activity nodes. For these nodes, an absolute error bound $\eta_{min}.\varepsilon$ is obtained. Even with this improvement, high execution times are observed while the accuracy is exceeded for regular nodes. In [Kwa98], circuit nodes

are partitioned in *M* groups according to their contribution to the total power dissipation, gradually decreasing the error to the high power groups. This error-to-group assignment is computed using a quadratic programming formulation. In [Din00] the authors present efficient sampling techniques for estimating the total power consumption of large hierarchical circuits.

3.3 Probabilistic Approaches

The other strategy to deal with the pattern dependence problem is probabilistic. In the statistics-based strategies the circuit under test is simulated with a number of patterns and *after* that, the resulting waveforms are processed as shown in Fig. 3.1.

On the other hand, if an appropriate probability characterization for circuit inputs is provided, the circuit can be simulated just once. In this way, some processing must be done somehow *before* the simulation run to compute the required probability values at the inputs. Thus, a single run of a probability analysis tool replaces a number of conventional circuit simulation runs. The issues to be defined for each specific technique in the following paragraphs are:

- what probability measures are required,
- how they must be obtained, and
- what type of analysis or simulation must be done.

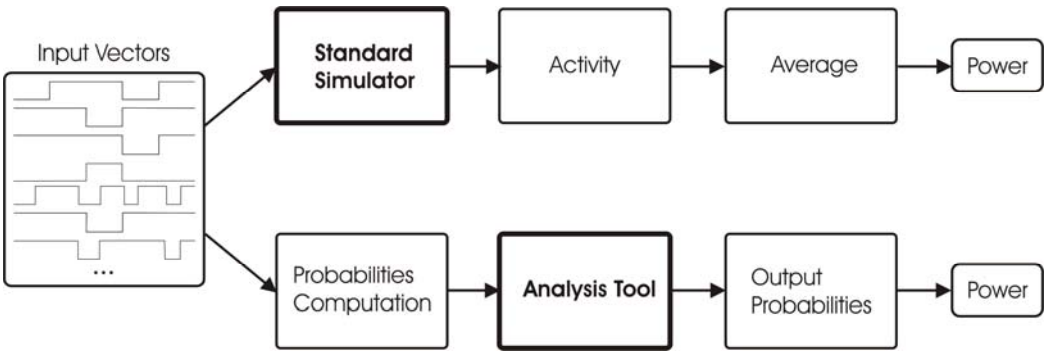


Fig. 3.1: Probabilistic (bottom) vs. Statistic approach (top)

As the results of the analysis still depend on the supplied probabilities –because the user must specify the typical behavior at the circuit inputs- it is said that techniques

based on probabilistic approaches are weakly pattern dependent, like the statistical ones.

3.3.1 Some Important Definitions

In this sub-section, the main probability definitions used in these power estimation techniques are briefly presented.

3.3.1.1 Definition 3.1: Signal Probability

The signal probability at a node *x*, $P_s(x)$, is defined as the average fraction of clock cycles in which the steady state value of *x* is a logic high.

It's important to observe that this measure is not affected by the circuit internal delays because steady state values are taken into account.

3.3.1.2 Definition 3.2: Transition Probability

The transition probability at a node *x*, $P_t(x)$, is defined as the average fraction of clock cycles in which the steady state value of *x* is different from its initial value.

As for signal probability, transition probability is not affected by the circuit internal delays. If these measurements are used to estimate power consumption in internal circuit nodes, toggle power (and spurious activity) is immediately excluded. But if it is used at circuit primary inputs, generally registered, there is no possibility of glitches, thus there is no lack of precision. In this way, signal probability and transition probability are good candidates to specify signal characteristics at circuit inputs.

Assuming a zero delay model, power consumption can be computed as:

$$P_{av} = \frac{1}{2T_C} V_{dd}^2 \sum_{i=1}^{n} C_i P_t(x_i) \qquad \text{(Eq. 3.5)}$$

Eq.3.5 gives a lower bound for P_{av}, compared with the general definition (Eq. 2.6) since Eq. 3.5 assumes at most one transition within a clock cycle.

The transition density, takes into account gate delays. It was introduced after the above definitions in [Naj91] and [Naj93]. The model for logic signals *x(t)* do not take

into account waveform details as the rise/fall times, over/under-shoots, etc. being just a function of the time that takes the values 0 or 1.

3.3.1.3 Definition 3.3: Transition Density

If a logical signal *x(t)* makes $n_x(T)$ transitions in a time interval *T*, then the transition density of *x(t)* is defined as:

$$D(x) = \lim_{T \to \infty} \frac{n_x(T)}{T} \qquad \text{(Eq. 3.6)}$$

The transition density takes into account gate delays, so average power dissipation can be accurately computed as:

$$P_{av} = \frac{1}{2} V_{dd}^2 \sum_{i=1}^{n} C_i D(x_i) \qquad \text{(Eq. 3.7)}$$

Instead of using *D(x)*, expressed in transitions per time unit, it is useful in synchronous circuits, to employ the transition number per clock cycle.

3.3.1.4 Definition 4: Equilibrium Probability

If *x(t)* is a logic signal, then its equilibrium probability is defined as:

$$P(x) \cong \lim_{T \to \infty} \frac{1}{T} \int_{-T/2}^{+T/2} x(t)dt \qquad \text{(Eq. 3.8)}$$

Equilibrium probability is the signal probability version taking into account gate delays. If a zero-delay model is assumed, $P(x) = P_s(x)$.

3.3.2 Probabilistic Power Estimation Techniques

The first paper found applying probabilistic techniques for VLSI power estimation is [Cir87] where zero-delay model and spatial and temporal independence is assumed.

Required probability measurement at circuit inputs: Signal probabilities, $P_s(x_i)$.

Analysis algorithm: Signal probabilities are propagated from primary inputs using basic probability theory.

Example: Let $y = AND\ (x_1, x_2)$

Given x_1, x_2 spatially independent; both signals with probability $P_s(x_1)$ *and* $P_s(x_2)$:

$$P_s(y) = P_s(x_1).P_s(x_2)$$

3.3.2.1 Probabilistic Simulation

In [Naj90], the reliability was studied more than the power, where the **shape** of the current waveform and not only its average current is important. The values at each time of the *expected current waveform* are calculated as the weighted average of all possible current values.

Restrictions: Spatial independence at primary inputs is assumed but temporal independence is not.

Required probability measures at circuit inputs: Probability waveforms. They are sequences of values indicating repeatedly the probability that the signal is high for a time interval and the probability that it makes a transition from low to high at the end of the interval. The transition times separating these intervals are deterministic. Fig. 3.2 is an example where the probability waveform is computed from four probable logical waveforms of a signal.

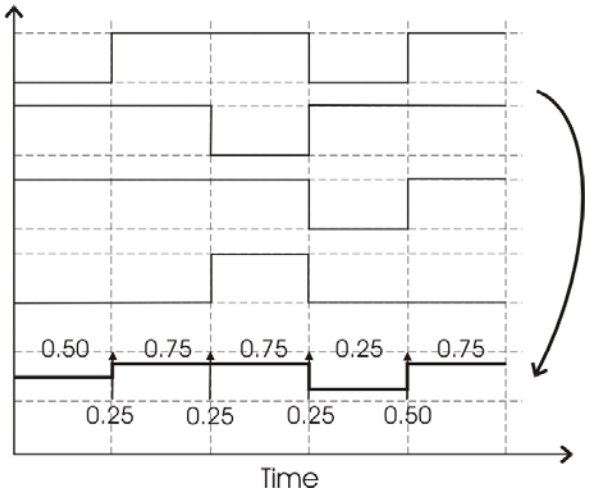


Fig. 3.2: Probability waveform computed from four logical waveforms

The drawback is the way to obtain the probability waveform. There are two (impractical) options: direct writing of the real number sequence, and obtaining it from a representative set of logical input waveforms. At least what should be the waveform length is not a trivial decision at any of the digital circuit design steps. Therefore, any

extension of [Naj90] should give a method to obtain a statistical-proved representative waveform.

Analysis algorithm: It is said that the simulator is probabilistic since it operates on probability waveforms but the simulation algorithm itself is deterministic. The spatial independence on internal nodes is managed through supergates, built on a partition of the circuit. But it is observed that it is computationally expensive to manage large supergates.

Probability waveforms are propagated from primary inputs computing the corresponding probability waveforms at all nodes. The propagation algorithm is similar to event driven logic simulation with a delay model. Indeed, the technique enables specifying logical waveforms at circuit inputs, so logical and probability waveforms can coexist in this algorithm. When an event occurs at a gate input, the gate generates an output event scheduled after a determined time delay.

The reported results are within 20% for peak currents and 10% for average current, which can be used to estimate average power according to:

$$P_{av} = V_{dd} . I_{av} \qquad \text{(Eq. 3.9)}$$

3.3.2.2 Transition Density Propagation

As in the technique explained above, in [Naj93] the activity is studied as causing stress failures. In addition to average power estimation, this technique can be used to estimate the average power and ground currents, the susceptibility to electromigration failures, and the extent of hot-electron degradation.

Restrictions: Spatial independence at primary inputs is assumed.

Required probability measurements at circuit inputs: For each input, equilibrium probability and transition density must be specified.

Obtained measurements: Immediate.

Analysis algorithm: as a digital circuit maps the logic signals from the primary inputs to every internal node, also the statistics of every internal node are determined by those at primary inputs. In fact, this technique gives an algorithm to compute the

transition density and equilibrium probability at every node from those given at primary inputs.

The circuit is considered as an interconnection of logic modules, each representing a combinational logic function with certain delay characteristics. The propagation of the transition density and equilibrium probability is done module by module and is called density simulation. [Naj93] shows that propagation can be reduced to the propagation problem in a zero-delay logic module. It is also important to understand how the partition of the circuit under test in logic modules affects the accuracy. Exact measurements can be obtained for a partition without reconvergent fanouts. Nevertheless, they were reported as poor as 32% error for the lowest level partitioning on an ISCAS-85 benchmark circuit.

Before explaining the density propagation problem, it is important to review the Boolean Difference definition ($\oplus$ denotes the exclusive-or operation):

$$\frac{\partial y}{\partial x} = y\big|_{x=1} \oplus y\big|_{x=0} \qquad \text{(Eq. 3.10),}$$

Being *x* an input and *y* an output of the considered logic module, observe that only if $\delta y/\delta x$ is 1, then a transition at *x* will cause a transition at *y*. Then, in [Naj93] it is demonstrated how the density can be calculated at a logic module output given the transition densities at the *n* (spatially independent) primary inputs:

$$D(y) = \sum_{i=1}^{n} P\left(\frac{\partial y}{\partial x_i}\right) D(x_i) \qquad \text{(Ec. 3.11)}$$

Intuitively, each input signal contributes to the total density by the average rate of transmitted transitions times the transition density of such input signal. If the transitions are transmitted from an input x_i to *y* 50% of the times and the transition density of x_i is 2 x 10^6 transitions per second, then the contribution of x_i to the transition density of *y* is 10^6 transitions per second.

There are other techniques to evaluate $P(\delta y/\delta x)$ probabilities for use in Eq. 3.10, but in [Naj93], Binary Decision Diagrams (BDDs) were used. With this, the average power consumption can be calculated using Ec.3.7.

If logic modules are chosen so that all reconvergent nodes are inside them, the calculated transition density is exact. But the main drawback of this technique appears when the size of the modules with this property becomes bigger and bigger. In such cases the speed-accuracy trade-off acquires importance. The partition into logic modules affects the calculation of *D(y)*.

3.4 Sequential Circuits

The methods described in section 3 have been developed for combinational logic circuits. Accurate average switching activity estimation for finite state machines (FSMs) is considerably more difficult for two reasons:

1. The probability of the circuit being in each of its possible states has to be calculated, maybe indirectly;
2. The present state line inputs of the FSM are strongly correlated
    - Temporally correlated due to the machine behavior, as represented in its state transition graph and,
    - Spatially correlated because of the given state encoding.

As for combinational circuits there are probabilistic and a statistical techniques. Nevertheless, in this work just the contributions among the statistical techniques are studied in the next section.

3.4.1 Statistical Approaches

The first work in this section estimates the power for the flip-flops and the combinational block of a sequential circuit separately extending the technique presented in [Naj98]. The rest of the presented papers assume that the FSM is embedded in the circuit and estimate the power of the circuit as a whole.

3.4.1.1 Power Consumption of the Sequential Circuit Combinational Part

In [Sax02] (the idea was firstly presented in [Naj95]) it is assumed that the studied sequential circuits have the synchronous design style. The power consumption for a combinational block of a sequential circuit could be estimated but no technique can be

applied if there is no information about the input patterns at all primary inputs including the inputs coming from state register outputs. Specifically, this information is not given for the state lines in a sequential circuit. Therefore, it is important to differentiate the two main stages in this technique for power estimation:

1. The necessary statistics acquisitions at the state register outputs.
2. With these statistics, any technique for combinational circuits can be applied over the combinational part of the sequential circuit.

In [Sax02] a statistical technique is proposed for the first stage and just this part of the estimation technique will be explained. The sequential circuit is simulated under a zero delay model because only steady state values are important to characterize the activities at the state lines. It means that fast, functional simulations over a RTL or higher-level circuit descriptions should be used. In addition, while computing these statistics, the power consumption must be estimated just for the registers.

The applied technique is based on two assumptions:

1. The sequential circuit implements a non-decomposable FSM.(i.e. all the states in the FSM are reachable from all the others in a finite number of clock cycles), and
2. The state of the FSM at cycle k becomes independent of its initial state as $k \rightarrow \infty$. This means that the FSM must be aperiodic.

The required statistics for the later power estimation stage on the state lines, $x_i(t)$, are the signal probability, $P(x)$, and transition density, $D(x)$. But it can be demonstrated that

$D(x) = P(t_x)$ where

$$t_x(k) = \begin{cases} 1, & x(k) \neq x(k-1) \\ 0, & otherwise \end{cases} \qquad \text{(Eq. 3.12)}$$

So, it is sufficient to describe an algorithm to estimate $P(x_i)$ and the same can be used to estimate $P(t_{xi}) = D(x_i)$.

The goal is to estimate $P(x)$ for all state lines. The probability of an event *A is denoted* by $P\{A\}$. It immediately gives

$$\lim_{k\to\infty} P\{x_i(k)=1 | X(0)=X_0\} = \lim_{k\to\infty} P\{x_i(k)=1\} = P(x_i) \quad \text{(Eq. 3.13)}$$

It means that the probability that the machine is in state x_i after a long time is independent of the state X_0, at which it was initialized. Therefore, the solution requires solving two sub-problems:

1. Estimating the left side of Eq.3.13 for some *k* and then
2. Obtaining an adequate *k* that guarantees convergence.

To solve the first problem the technique consists of running *N* simulations. Each run *j*, generates a waveform for each state line x_i

$x_i^{(j)}(k)$ with *k* = 0, 1, 2, ...

An estimate for the signal probability at every time *k* is obtained by

$$p_i^{(N)}(k) = \frac{1}{N}\sum_{j=1}^{N} x_i^{(j)}(k)$$

From the law of large numbers

$$\lim_{k\to\infty} p_i^{(N)}(k) = P_k(x_i | X_0)$$

Using a technique for the estimation of proportions, the required number of simulation runs *N* can be obtained with a given error and confidence level. If the confidence level is (1-α)x100% that:

$$\left| p_i^{(N)}(k) - P_k(x_i | X_0) \right| < \varepsilon \quad \text{(Eq. 3.14)}$$

Then $N > \max(N_1^2, N_2^2, N_3^2)$

where

$$N_1 = \frac{z_{\alpha/2}}{2\varepsilon}, \quad N_2 = \frac{z_{\alpha/2}\sqrt{2\varepsilon+0.1} + \sqrt{(\varepsilon+0.1)z_{\alpha/2}^2 + 3\varepsilon}}{2\varepsilon}, \quad N_3 = \frac{\sqrt{63} + z_{\alpha/2}}{2\sqrt{\varepsilon}} \quad \text{(Eq. 3.15)}$$

$z_{\alpha/2}$ is such that the probability the standard normal random variable is greater than $\alpha/2$.

To solve 2) two sets of N simulations are run in parallel starting at different initial states X_0 and X_1. As $P(x_i|X_0)$ and $P(x_i|X_1)$ must converge to $P(x_i)$, both their difference and average are monitored. When $P(x_i|X_0)$ and $P(x_i|X_1)$ are within $\pm\varepsilon$ for L consecutive cycles, the node convergence is declared.

To accelerate the convergence the waveforms are filtered before checking their difference and average. A 100 points FIR filter with a cutoff frequency of 0.02 Hz is used. When all nodes converge, the simulation is terminated and the last average for each node is reported as its signal probability.

For the primary input vector generation two considerations must be observed:

1. Periodic sequences must be avoided, and
2. The different N sequences for the N runs must be independently generated.

3.4.1.2 Sequential Circuits with Multimodal Distributions in Power Consumption

In the statistical techniques the simulation must start at a given initial state of the sequential circuit, but bias in the convergence should be checked. As reported for combinational circuits, multimodal distributions in power consumption can be found in sequential circuits. [Cho96] explains it by the Near-Closed (NC) sets. Intuitively, a NC set is a set of states that is unlike to get out or get into the set.

In this paper, two methods are proposed to estimate power consumption for sequential circuits with, possibly, NC sets:

1. If the STG modeling the sequential circuit is given, a STG based statistical method is used.
2. If it is not the case, the circuit is simulated during a warm up stage before any data sampling.

In this paper, it is shown how the method proposed in [Naj98] gives biased results when there are NC sets.

When the STG is given, and the NC sets (G_i) are identified, the average normalized activity (transition probability) at node y can be estimated, using the conditional activities for each set:

$$a(y) = \sum_i a(y|G_i) \times P(G_i) \qquad \text{(Eq. 3.16)}$$

Taking into account this equation, a modified Monte Carlo technique can be applied. In the modified Monte Carlo method, the initial states are generated according to the probabilities of NC sets. Now the problem is how to compute the state probabilities. A general solution is presented in [Cho96].

In the second case, the STG is not available for the sequential circuit under study. As it could be the common case for a general power estimation tool, it is studied here in detail. Conservatively, let us assume that there are two NC sets G_1 and G_2:

$$a(y) = a(y|G_1) \times P(G_1) + a(y|G_2) \times P(G_2)$$

In this case we do not have the probabilities but it can be shown that $P^k_{warm-up}(G_i)$ is very close to $P(G_i)$ as k is big enough.

$$P^k_{warm-up}(G_i) = \frac{1}{|E|} \sum_{S_i \in E} \sum_{S_j \in G_i} P^k(i, j) \qquad \text{(Eq. 3.17)}$$

where E and $|E|$ are the state space and its cardinality, and $P^k(i, j)$. Intuitively, if the Markov chain starts from a state uniformly selected, at the k^{th} clock cycle, the probability of arriving at any state in G_i is $P^k_{warm-up}(G_i)$.

It can also be shown that, if transition matrix P is aperiodic and irreducible, for all i, j:

$$\lim_{k \to \infty} P^k(i, j) = \pi(j) > 0 \qquad \text{(Eq. 3.18)}$$

$\pi(j)$, the steady state probability of state j is independent of i. From (Eq. 3.14) and (Eq. 3.15) $P^k_{warm-up}(G_i)$ converges to $P(G_i)$ as k approaches to infinity:

$$\lim_{k \to \infty} P^k_{warm-up}(G_i) = \frac{1}{|E|} \sum_{S_i \in E} \sum_{S_j \in G_i} \left(\lim_{k \to \infty} P^k(i, j) \right) = \frac{1}{|E|} \sum_{S_i \in E} \sum_{S_j \in G_i} \pi(j) = \frac{1}{|E|} \sum_{S_i \in E} P(G_i) = P(G_i)$$

If P is also diagonalizable, its eigenvalues: $\lambda_1 = 1 \geq |\lambda_2| \geq |\lambda_3| \geq ... \geq |\lambda_{|E|}|$, and there are two NC sets G_1 and G_2,

$$\sum_{S_j \in G_1} P(l, j) = p_1 \quad \text{for } S_l \in G_1$$

$$\sum_{S_j \in G_2} P(l, j) = p_2 \quad \text{for } S_2 \in G_2$$

where p_1 and p_2 such that $0 \le p_1, p_2 \approx 1$. It can always be found

$$\sum_{S_l \in G_i} \sum_{S_j \in G_i} P(l, j) = |G_i| \times p_i \qquad \text{for i = 1, 2} \qquad \text{(Eq. 3.19)}$$

Experimentally it has been found that $\lambda_2 \approx p_1 + p_2 - 1$ and

$$\left| P^k_{warm-up}(G_i) - P(G_i) \right| \le |E| \left| \lambda_2 \right|^k \qquad \text{(Eq. 3.20)}$$

If the allowed error is specified, the minimum k can be calculated. K is the warm-up period. Starting from a random initial state, and after the warm-up period, a new initial state is generated conforming to the probabilities of NC sets.

As it was assumed no information on STG, λ_2 and $|E|$ are specified conservatively. $|E|$ is assumed to be 2^{n+m} considering temporally correlated primary inputs. N is the number of primary inputs and m is the number of state bits.[3] Λ_2 must be specified by the user, under but close to one.

With this, k may be computed given the tolerated error, e, n, m, and λ_2. For example, in [Cho96] some experiments are run with:

$\lambda_2 = 0.999$

$P(G_i)=0.1$; $e=1\%$

$n=3$; $m=3$; $2^{n+m} = 64$

Then $0.1 \times 0.01 \le 2^6 \times 0.999^k$; $k = \log_{0.999} \frac{0.001}{64} \cong 11061$

This is a rather small test circuit but with values for a long run. The next example can be considered with operational values.

[3] The STG must be a Markov chain: If primary inputs were time independent this condition holds and 2^n could be used for $|E|$, but as it can not be the case the Extended ETG is a Markov chain and it can have until 2^{n+m} states.

$\lambda_2 = 0.9$

$P(G_i)=0.1$; $e=5\%$

$n=17$; $m=74$; $2^{n+m}=64$

Then $0.1 \times 0.05 \le 2^{91} \times 0.9^k$; $k = \log_{0.9} \frac{0.005}{2{,}4759\mathrm{E}+27} \cong 649$

This warm-up period requires a simulation time for each sample blindly assuming NC sets. In this way, this technique is considered conservative and this could lead to computational inefficiency. [Yua96] proposes a solution for this problem that is explained below.

3.4.1.3 A Technique to Generate a Random Sample in Sequential Circuits

Statistical estimation techniques require samples of independent and identically distributed (iid) data. However, sequential circuits present strong correlations among state lines. As it was said in section 3.4.1.1, a mayor drawback in [Sax02] trying to solve this problem is that spatial and high order temporal correlations among state signals are not considered. This could yield to poor accuracy as reported in [Sch96a].

[Cho96] works with the conservative assumption that the FSM has two NC sets. In the case that an actual circuit under test does not have NC sets, this assumption leads to a warm-up period longer than necessary.

In [Yua96], the problem with the correlations in sequential circuits is overcome using a randomness test to determine an independence interval over which the circuit should be simulated between two power sampling cycles. The independence interval is incremented until the hypothesis that the sequence is composed of iid's is statistically accepted with a user-defined significance level. After this a power random sample can be obtained. A distribution-independent stopping criterion is then used to analyze the power data until the desired accuracy is achieved.

More formally, the circuit under test dissipate $P_1, P_2, \ldots, P_n$ power values in n consecutive clock cycles. The sequence can be viewed as a random process $\{ \boldsymbol{P}_j \}$. Since mean estimation require random samples and the consecutive values are correlated, $P_1, P_2, \ldots, P_n$ cannot be directly used.

The task here is extracting an iid sequence from this original time series. If we assume that { $\boldsymbol{P}_j$ } is Φ-mixing[4] and stationary with finite variance, there exists an interval of m clock cycles, such as P_k, P_{k+m}, P_{k+2m} … which are independent. If we find such independent interval m, a random sample can be constructed by recording the power dissipation once every m clock cycles.

[Yua96] propose the use of a randomness test, the ordinary run test, belonging to the category of a non-parametric hypothesis test, to examine the statistical independence of the data in the power sequence and then to choose a proper independence interval; which is used to generate a random sample.

The ordinary run test works on an ordered sequence of two-symbol data. In these sequences, a run is defined as a succession of one or more identical symbols limited by the other symbol. The hypothesis is that the sequence is randomly generated. If it is true, the number of runs has a normal distribution.

Let an ordered sequence contain m and n symbols of each type. The total number of elements is $N=m+n$. Let U be the number of runs in the sequence. If the hypothesis is true, U has an asymptotic normal distribution.

The z statistic is:

$$z = \begin{cases} -\dfrac{U + 0.5 - 1 - 2mn/N}{\sqrt{\dfrac{2mn(2mn-N)}{N^2(N-1)}}} & if \quad U < 1 + 2mn/N \\ \dfrac{U - 0.5 - 1 - 2mn/N}{\sqrt{\dfrac{2mn(2mn-N)}{N^2(N-1)}}} & if \quad U > 1 + 2mn/N \end{cases}$$

(Eq. 3.21)

A small z in absolute value statistically indicates that the hypothesis H is true.

H: The sequence is random.

A: The sequence is not random.

(Eq. 3.22)

[4] Φ-mixing refers to the property that the future behavior of { $\boldsymbol{P}_j$ } becomes more and more independent of its past as the distance in time increases.

Let a value c > 0 such that H is accepted if

$$|z| < c \qquad \text{(Eq. 3.23)}$$

The user specifies a value α, called the significance level, and c is calculated by:

$$c = N^{-1}\left(1 - \frac{\alpha}{2}\right) \qquad \text{(Eq. 3.24)}$$

In short, to evaluate the sequence randomness, count *m*, *n*, and *U* and calculate the value of *z* with Eq. 3.18. Finally, accept or reject *H* using Eq. 3.20.

In the specific power estimation problem, in order to obtain a two-symbol sequence, in [Yua96] the median of the power sequence is calculated. With this, a symbol A is assigned to the values smaller than the median, and B, to the others.

To obtain the independence interval, [Yua96] proposes to evaluate try-values from cero, and collect the sequence using these values until the hypothesis is accepted. Once an independence interval is statistically obtained, the random sample sequence is efficiently obtained.

The random sample sequence acquisition is optimized running a zero delay simulation during the independence interval while a real delay model is used to sample the power. To measure the sample convergence, in [Yua96] a non-parametric criterion is employed [Yua98].

3.4.1.4 Block Sampling in large Sequential Circuits

[Koz01] observed that several techniques for average power estimation use signal probability and transition density as inputs. However, for large sequential circuits, with several operation modes, it is possible to define two input sets with the same signal probabilities and transition densities at every input, obtaining different power consumption figures. To show it, several sequential circuits were tested with two such input sets. The first vector set have spatial and temporal correlations, in the other vector set input signals are generated randomly, with the same signal probabilities and transition densities as in the first set. The reported average error in respect to the first vector set was about 30% but there was a circuit with an error of 120%.

The authors claim that an accurate method for power estimation in sequential circuits must simulate the circuit for realistic and typical vector sets, referred to as power vector sets. Nevertheless, in practice it is almost impossible to obtain a short enough power vector that considers the typical operation of the circuit with all its operation modes.

The proposed solution is based on a block sampling approach. Randomly selected blocks are simulated from a potentially huge power vector set. For each block, a lower bound and an upper bound are obtained for the power consumption. The average of these two values estimates the power dissipation taking into account all possible initial states, but in an original way, as is explained below. Using a Monte Carlo mean estimation technique, both bounds can be obtained with a specified error and confidence level.

This approach requires the solution of some sub problems. To perform a well defined estimation, a well defined initial state should be specified. However, some initial states, and input vectors could lead to unfeasible operations in the FSM. To solve this problem, the state bits are all set to X state (the unknown state, in this context) at the beginning of the block simulation. During the three valued simulation the upper and lower bounds are computed assuming at every transition from X to a 0 or 1, the 0 and 1 values for X to compute the upper and lower bounds. For example if the output of a gate makes an X->1 transition, assuming ‘0’ for the unknown state, a power consuming transition would happen, participating to the upper bound. If ‘1’ were assumed for the unknown state (1->1) no power would be consumed, participating to the lower bound. In that way the true power consumption for the simulated block is guaranteed to be between the upper and lower bounds, and the initial state is well-defined.

The second sub problem is the choice of the block size. It can affect the tightness of the bounds. As it becomes larger, more X’s will become defined, but from some block sizes and larger ones, there will be little reduction in the number of X’s. In [Koz01] the block size was set empirically to 500 cycles.

Finally, within a Monte Carlo mean estimation technique, the number of samples to estimate the average upper bound may be different from that required for the average lower bound, so the sampling is continued until the stopping criterion have been met for both means.

The stopping criterion is:

$$N \geq \left(\frac{z_{\alpha/2} s_N}{\mu_N \varepsilon_1} \right)^2$$

where :

$Z_{\alpha/2}$ is defined so that the area under the standard normal distribution is equal to $\alpha/2$.

μ_N is the sample mean

S_N is the sample standard deviation.

ε_1 is a small positive number, such that if ε is the user specified tolerated error in the estimation, $\varepsilon = \frac{\varepsilon_1}{1-\varepsilon_1}$.

3.5 Power Estimation Methods Applied on FPGAs

In the FPGA arena, the first approaches to power estimation, was a set of equations where the inputs are the number of logic blocks, I/O blocks, etc. In addition, a rough value for the activity must be specified. For example, Xilinx presents such equations in [Faw97], [Xil97h], [Tan99], [Xil00]. Just to mention one case, [Xil97h] presents a simple method to calculate the power dissipation in the XC4000 family. Power was estimated based on the number of logic cells and the percentage of them toggling every clock cycle. Later, these operations were automated in spreadsheets and tools in the web [XilPow]. The estimation, although more sophisticated, is based on similar data and equations as in [Xil97h] and with the same drawback: low accuracy. Altera has a similar approach and set of tools [Alt04]. This approach, although inaccurate, is useful for early power estimates.

[Gar99] presents a power consumption model for FPGA based on incremental measurements. Starting from a simple design, one of the internal resources is increased keeping the rest as it is. The difference in the current consumption is annotated to measure the power consumed by the selected element. With the power values for every resource (flip-flop, LUT, interconnect lines, etc.), and the number of occurrences of each type in the design, total and individual power can be computed.

Nevertheless, the user must provide the activity and the pattern-dependence problem is not considered.

[Osm98] employs the Xilinx's 4000 family as technological framework. This paper presents an activity estimation technique. It is based on the propagation of probabilistic parameters (signal probability and activity) from primary inputs to all the internal circuit nodes.

[Shan02] analyzes the dynamic power consumption of the Virtex-II family and reports where the power is consumed in these devices. Furthermore, it reports static and short-circuit power values for these devices. The main goal of that work is to detect in which part of FPGA the power is consumed in order to help actual designers to optimize power, and develop future power-efficient FPGAs. This information is also useful to engineer accurate EDA tools for low power design. Although the reported results are valuable, it should be noted that the authors use non-public information for the capacitances used in power computation: capacitances are obtained using Spice simulations from detailed schematics of the FPGA circuits. The sources of dynamic power were separated in logic, clock and the different routing resources. The steps to estimate the power consumption have been mentioned, but they have not been integrated within an EDA tool.

[Poo02] applied the probabilistic approach and evaluated different FPGA architectures for power efficiency. However, this implementation did not consider glitch power, or spatial and temporal signal correlations. Physical measurements were not provided because theoretical FPGA models were analyzed. [Li03] also analyzed FPGA architectures but glitch power and signal correlations were considered and important conclusions were reached. It is significant to note that the goal in [Poo02] and [Li03] is different to the one in this thesis. Here, it is important to measure the power dissipated in actual devices with real designs in order to develop a robust and accurate power estimation platform, while present and future FPGA architectures are studied.

Finally, in the Xilinx Integrated Software Environment (ISE), a power estimation tool, called XPower [XilUser], [Xil01] is provided since 5.1i version. It is a software tool that calculates the power consumption based on the physical implementation on a specific device and a timing simulation file. With a proper vector set used in the simulation, it can provide acceptable estimations of power usage. It is important to emphasize the

word “proper”. The user can provide arbitrary input vector sets. Therefore, the tool cannot guarantee that simulated activity really converges with the average values. Input vector generation is a user responsibility. The user should provide these input vectors by a specialized software tool but this program does not exist yet. In practice, XPower ignores the effect of data statistics on power consumption.

[And04] presents novel techniques for early activity and capacitance prediction on FPGAs. Activity estimation is based on zero-delay simulation models and a prediction function for glitches. On the other hand, the capacitance estimation is based on both technology-independent parameters and the specific interconnect architecture of the current FPGA where the design is implemented. These techniques are useful within optimization loops of power-aware synthesis, placement and routing tools, and early-power estimation.

Even considering all this work, it is clear that power estimation is not up to date in today’s commercial or academic FPGA environments. This thesis contributes to the previous research lines by the development of a new FPGA-oriented power estimation platform, where the accuracy is statistically guaranteed and main modules can be selected for specific applications.

3.5.1 Related Works at the UAM[5]

Low Power Design has two fundamental pillars: estimation and optimization. [Sut05] presents a methodology for power optimization at the topological, architectural, and algorithmic levels for FPGAs. The application of pipeline, block disabling, and other general purpose techniques are characterized for several FPGA families. In addition, a number of techniques are studied in order to reduce the power consumption in finite state machines and arithmetic circuits.

Power consumption produces temperature increases in the die and the entire chip. [Lop03] proposes the use of ring oscillators in the FPGA at positions that can be selected by the user. The technique permits the verification and detection of hot spots in the FPGA device.

[5] Universidad Autónoma de Madrid, School of Engineering

3.6 Conclusions

Even at the gate level, the problem of power estimation is not completely solved yet. Due to its computational complexity, accuracy and speed cannot both be met. This problem is observed for average power estimation, working with individual gates; and for total average power consumption, in large sequential circuits [Koz01].

Although both probabilistic and statistical techniques are studied in this work, the later ones are selected for the platform developed in this thesis and experiments due to the tunable accuracy properties they present, easier implementation, and general application, particularly in the FPGA environments.

Even considering all the work developed for FPGAs, commercial and academic tools for FPGA environments are not up to date for estimating power consumption. This thesis contributes to the previous work by developing a new FPGA-oriented power estimation platform, where the accuracy is statistically guaranteed. The main modules in this platform could be selected for specific applications. Also, the software pieces can be easily updated and improved making the platform flexible enough for today's fast changing FPGA technologies.

References

[Alt04] Altera Corp. "Power Calculator User Guide", March 2004, available at http://www.altera.com/support/devices/estimator/pow-powerplay.html

[And04] Anderson, J.H.; Najm, F.N., "Power estimation techniques for FPGAs", IEEE Trans. on VLSI Systems, Volume 12, Issue 10, pp. 1015-1027, Oct. 2004

[Büh00] M. Bühler, M. Papesch and U.G. Baitinger, "Accurate and Approximate Methods for Speeding up Signal Activity Estimation on Gate Level", PATMOS 2000, pp. 179-188.

[Bur93] R. Burch, F. N. Najm, P. Yang, and T. Trick. "A Monte Carlo approach for power estimation" IEEE Transactions on VLSI Systems, 1(1):63–71, March 1993.

[Buy05] K.M. Büyükşahin and F.N. Najm, "Early Power Estimation for VLSI Circuits", IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, 1(7):1076–1088, July 2005.

[Cho96] T. Chou y K. Roy, "Accurate Power Estimation of CMOS Sequential Circuits", IEEE Trans. On VLSI Systems, Vol.4, nº3, pp.369-380. September 1996.

[Cir87] M. A. Cirit, "Estimating Dynamic Power Consumption of CMOS Circuits", Proc. ICCAD, pp. 534-537, November 1987.

[Den94] C. Deng. "Power analysis for CMOS/BiCMOS circuits. " In Proceedings of the

1994 International Workshop on Low Power Design, pages 3–8, April 1994.

[Din00] Ding, C-S., C-T. Hsieh and M. Pedram, “Improving efficiency of the Monte Carlo power estimation”, IEEE Trans. on VLSI Systems, Vol. 8, No. 5, (2000) pp. 584-593.

[Faw97] Fawcett, B.: FPGAs, Power and Packages. XCELL (1997)

[Gar00] Andrés David García García, “Etude sur l’estimation et l’optimisation de la consommation de puissance des circuits logiques programmables du type FPGA”, Ph. D. Thesis, Ecole Nationale Supérieure des Télécommunications, paris, 2000.

[Geo94] B. George et al, “Power Analysis for Semi-Custom Design”, IEEE 1994 Custom Integrated Circuits Conf., pp.249-252, New York: IEEE Press 1994.

[Gho92] Abhijit Ghosh, Srinivas Devadas, Kurt Keutzer, Jacob White, “Estimation of Average Switching Activity in Combinational and Sequential Circuits”, In Procs. of the 29th ACM/IEEE Conference on Design Automation, pp. 253-259, 1992

[Hay01] Brian Hayes, “The Computer and the Dynamo”, American Scientist, Vol 89, No 5, September-October, 2001. pp. 390-394.

[Koz01] J. Kozhaya and F. N. Najm, “Power estimation for large sequential circuits”, IEEE Transactions on VLSI, vol. 9, no. 2, pp. 400-407, April 2001.

[Kwa98] B. Kwak, and E.S. Park, “An Optimization-Based Error Calculation for Statistical Power Estimation of CMOS Logic Circuits,” in Procs. of the Design Automation Conference, San Francisco, California, USA, pp. 690-693, 1998.

[Lan94] P. Landman, Low-Power Architectural Design Methodologies, Ph. D. Thesis, Electronic Research Laboratory, University of California, Berkeley, August 1994.

[Li03] Fei Li, Deming Chen, Lei He, Jason Cong: “Architecture evaluation for power-efficient FPGAs”, Proc. Of Int. Symp on Field Programmable Gate Arrays, 2003, pp. 175–184.

[Lop03] Sergio López Buedo, “Técnicas de Verificación Térmica para Arquitecturas Dinámicamente Reconfigurables”, Ph. D. Thesis, Departamento de Ingeniería Informática, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Julio 2003.

[Naj90] F. Najm, R. Burch, P. Yang, and I. Hajj, “Probabilistic simulation for reliability Analysis of CMOS VLSI circuits,” IEEE Transactions on Computer-Aided Design, vol. 9, no. 4, pp. 439-450, April 1990 (Errata in July 1990).

[Naj91] F. Najm, “Transition density, a stochastic measure of activity in digital circuits,” 28th ACM/IEEE Design Automation Conference, San Francisco, CA, pp. 644-649, June 17-21, 1991.

[Naj93] F. Najm, “Transition density: a new measure of activity in digital circuits,” IEEE Transactions on Computer-Aided Design, vol. 12, no. 2, pp. 310-323, February 1993.

[Naj94] F. Najm, "A survey of power estimation techniques in VLSI circuits," IEEE Transactions on VLSI Systems, vol. 2, no. 4, pp. 446-455, Dec. 1994.

[Naj95] F. N. Najm, S. Goel, and I. N. Hajj, "Power estimation in sequential circuits" ACM/IEEE Design Automation Conference, pp. 635-640, 1995.

[Naj98] F. N. Najm and M. G. Xakellis, “Statistical estimation of the switching activity in

VLSI circuits", VLSI Design, vol. 7, no. 3, pp. 243-254, 1998.

[Osm98] Timothy A. Osmulski, Implementation and Evaluation of a Power Prediction Model for a Field Programmable Gate Array, Master's Thesis, Department of Computer Science, Texas Tech University, Lubbock, TX, May 1998.

[Poo02] Kara K.W. Poon, Andy Yan, Steven J.E. Wilton, "A Flexible Power Model for FPGAs", LNCS, Volume 2438, Jan 2002, pp. 312-321.

[Sax02] V. Saxena, F. N. Najm, and I. N. Hajj, "Estimation of state line statistics in sequential circuits," *ACM Transactions on the Design Automation of Electronic Systems*, Vol. 7, No. 3, pp. 455-473, July 2002.

[Sch95] P.H. Schneider, "PAPSAS: A Fast Switching Activity Simulator", PATMOS'95, 1995, pp. 351-360.

[Sch96a] P. Schneider and S. Krishnamoorthy. "Effects of correlations on accuracy of power analysis - an experimental study", International Symposium on Low Power Electronics and Design, Monterey, California, United States, 1996, pp. 113-116.

[Sha02] L. Shang, A. S. Kaviani, K. Bathala, "Dynamic Power Consumption in Virtex-II FPGA Family", FPGA 2002 Monterey, California, USA, February 24-26, 2002, pp. 157-164.

[Sut05] Gustavo Sutter, "Aportes a la Reducción de Consumo en FPFAs", Ph. D. Thesis, Departamento de Ingeniería Informática, Escuela Politécnica Superior, Universidad Autónoma de Madrid, April 2005.

[Tan99] Tan, J: Virtex Power Estimator User Guide. XAPP 152 (1999)

[Xac94] M. Xakellis and F. Najm, "Statistical Estimation of the Switching Activity in Digital Circuits", 31st ACM/IEEE Design Automation Conference, San Diego, CA, pp. 728-733, 1994.

[Xil00] Xilinx Inc.: XC4000XL Power Calculation. XCELL, Nº27 (2000) pp 29

[Xil01] Xilinx Inc.: XPower Tutorial: FPGA Design, XPower (v1.1). (2001) Available at http://www.xilinx.com.

[Xil97h] Xilinx Press, "A Simple Method of Estimating Power in XC4000XL/EX/E FPGAs", Application Brief, XBRF 014 June 30, 1997.

[XilPow] Xilinx Inc. "Power Central", available at http://www.xilinx.com/products/design_resources/power_central/

[XilUser] Xilinx Inc.: "ISE 7 User Guide". Available at http://www.xilinx.com

[Yua97] L. Yuan, C. Teng, S. Kang, " Statistical estimation of average power dissipation in CMOS VLSI circuits using nonparametric techniques", Procs. of the 1996 international symposium on Low power electronics and design, Monterey, California, United States, pp. 73 – 78, August 1996.

[Yua98] L. Yuan, C. Teng, S. Kang, "Statistical estimation of average power dissipation using nonparametric techniques", IEEE Trans. on VLSI Systems, Vol 6, No 1, pp. 65-73, Mar 1998.

Chapter 4.

"It was Ross Freeman, really, who had the radical notion that transistors are free".
By Xilinx Staff, "Celebrating 20 Years of innovation", XCell Journal, Spring 2004.

4 A-DyP: A Tool for Average Power Estimation in FPGAs

In this Chapter, the main structure and characteristics of the power estimation platform for FPGA environments are presented together with a power estimation tool: A-DyP (Average Dynamic Power estimator). This tool is able to estimate average power for both the whole design and individual nodes. A-DyP is statistical-based, allowing the user to specify the tolerated error at a given confidence level.

Several techniques have been developed to estimate the power consumption of digital circuits (See Chapter 3). The present work tries to contribute to the previous research lines by the development of a new FPGA-oriented power platform and A-DyP that is implemented over this platform. The power platform includes:

- A data structure known as the Common Power Database.
- An abstraction layer to access the Power Database independently of the database engine currently used.
- Use of accepted standards formats in order to make straightforward the integration with current EDA software. These files are used for information interchange between the commercial applications and the different programs

in this power estimation system. The platform provides parsers for these formats.

- Parsers for the standard VCD format and VHDL.
- Tcl/Tk automation in order to integrate the common and the specific programs within a power estimation tool.
- A common approach to manage configuration data and an API to access this information from several programs in a power estimation system.
- An input generator program that is able to produce stimulus for both a simulator and a pattern generator that typically can be found together with a logic analyzer.
- Parsers for Xilinx proprietary formats useful for power estimation in that environment.
- Software components to read the Common Power Database; and write common power reports, write files compatible with scientific graphing tools to produce power, capacitance, activity maps, etc.
- Integration within available design flows.

A-DyP is developed on the power estimation platform. It can execute all the estimation process automatically, from the generation of input vectors (according to the user specifications), to the correlation of the physical node positions with their individual power consumptions. The last task allows the designers to create power maps that can help to detect hot spots in the die. The current version of the tool can be integrated into the latest Xilinx ISE suite and it has been tested with the Modelsim simulator.

A-DyP has two main sub-systems: the first one is responsible for the average activity estimation of the individual nodes in the DUT, while the second one calculates the power multiplying the estimated activities by the corresponding node capacitances. This Chapter presents an overview of the estimation tool implementation and the use of the power platform. The development details of the different tasks composing each subsystem are treated in depth in the next chapters.

4.1 A-DyP Main Structure

A-DyP is composed by several pieces of software and some of its programs interact with third-party tools. The main external application that forms part of the A-DyP core is the simulator. Nevertheless, synthesis and compilation tools are necessary in order to generate a design layout (from the layout, a VHDL model is obtained for the power estimation). Finally, a scientific graphing tool helps to draw power, capacitance and activity maps.

Fig. 4.1 shows the main structure of the proposed tool. It is composed of two main sub-systems that share a data repository: The Power Database. The Activity Estimation Sub-system estimates the average activity and standard deviation for all the individual nodes in the DUT. The estimated values are stored in the Power Database. On the other hand, the Power Computation sub-system multiplies these estimated activities by the corresponding node capacitances according to Eq. 2.7. The way the FPGA vendors provide node capacitances and other physical information is complex enough to implement this functionality in this separate piece of software.

The user interface is not strictly part or the Activity Estimation Sub-system as suggested in the Fig.4.1. It is the user friendly way provided by the power estimation toolkit to write the parameters (ini) file. Likewise, the ini file is a plain text file.

Before starting the activity estimation, compile and elaborate the VHDL model for the simulator is needed to be able to start the simulation. This is called the Set-Up phase and is the first task of the Activity Simulation Sub-system. The Set-up phase is detailed below.

It is important to note that the power estimation is part of a design flow and the simulation model is obtained after the design was synthesized and implemented for a particular FPGA device. In the context of the estimation process, the part of the design flow necessary to generate the simulation model and the other input files is considered the Preparation Phase.

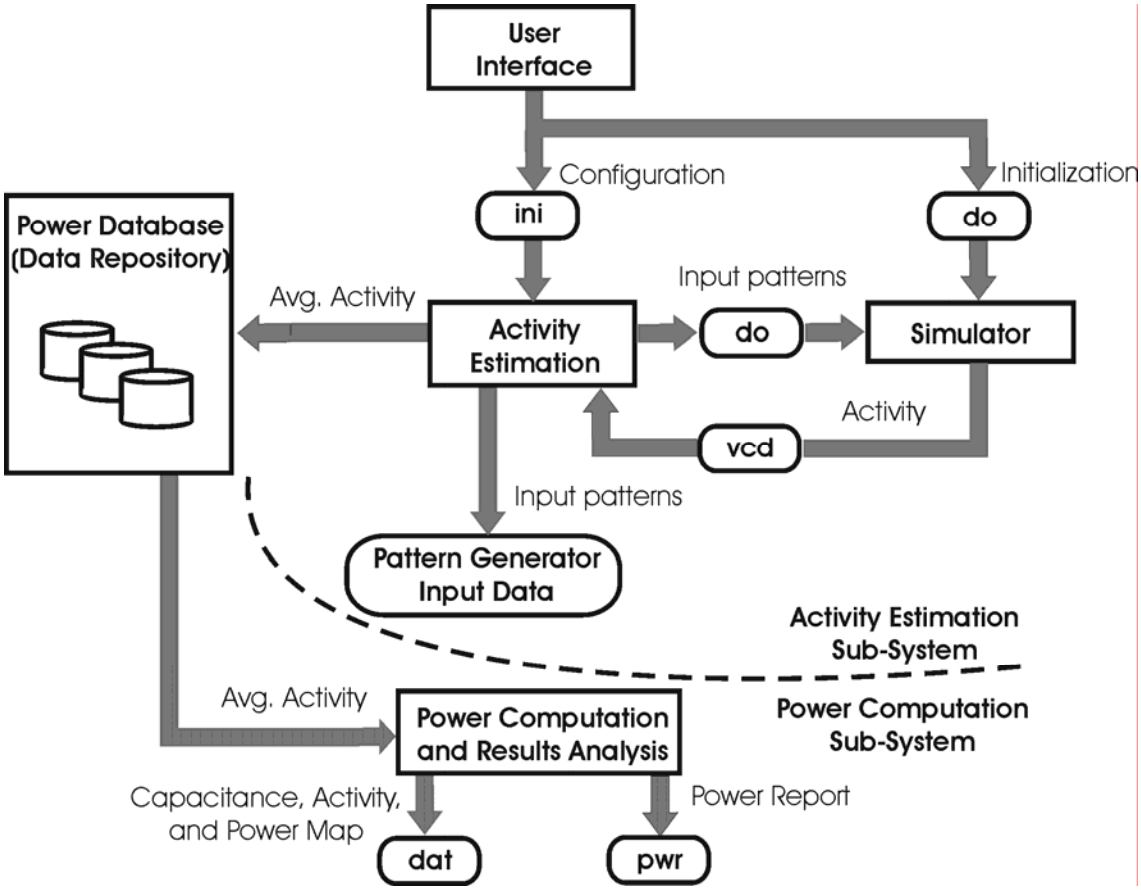


Fig. 4.1: Average dynamic power estimation tool (A-DyP). Main structure

In the next sections all the work is presented to give the reader a global view of the platform, the development of the tool, and underlying methodology. The preparation, the tool itself, and the post processing steps are explained. The user interface, the set-up phase, the Activity and Power Computation sub-systems, and the Power Database that compose A-DyP are also briefly set out.

4.2 The Preparation Phase

The goal here is to obtain a model of the DUT for which the power can be estimated.

Before running the proposed power estimation tool, the DUT needs to be synthesized and compiled. The synthesis can be done with any tool compatible with the Xilinx ISE design flow and the compilation must be done with the Xilinx ISE [Xil].

The power estimation tool could operate with designs specified in any HDL supported by the simulator, or mixed designs, and at any level in the development process. However, in order to consider the impact of the technology mapping, and the PAR, both a design description and an accurate delay model should be provided.

ISE produces a layout file after the PAR in a proprietary format (NCD). This file is translated with the `netgen` command to a VHDL simulation model and the corresponding SDF delay model (Standard Delay Format) [SDF01]. This command is included in the Xilinx ISE distribution. It can be executed from the ISE graphical interface, the command line, or automatically from the power estimation tool when it detects these files do not exist in the working directory. For example, the following line can be found in the set-up script:

```
netgen –ofmt vhdl –sim –w –aka –pcf design.pcf design_name
```

where

`design_name` is the NCD design file

`–ofmt vhdl` specify vhdl or verilog output format for simulation.

`–sim` generate a netlist compatible with a simulation tool.

`–w` overwrites the output file

`–aka` write “Also-Know-As” names as comments. This is a very important option in this work and its use will be explained in detail in Chapter 6.

`–pcf design.pcf` is an input constraint file.

The VHDL simulation model obtained uses a structural style where VITAL cells are instantiated [VIT01]. VHDL VITAL timing data is annotated from the SDF file using the simulator’s built-in SDF annotator. VHDL SDF annotation works on VITAL cells only. The IEEE 1076.4 VITAL ASIC Modelling Specification describes how cells must be written to support SDF annotation, but Xilinx, as other FPGA and ASIC vendors, naturally has already written the VITAL cells and provide tools that create compatible SDF files (`netgen`).

Note that in some cases the VITAL library could not have been compiled and be ready to use, but it can be built with the source files and tools provided by Xilinx. For specific information about how to do it, you can refer to the vendor documentation [XilSav] (Chapter 6: “Simulating Your Design”).

4.2.1 User Interface

The proposed power estimation tool can be executed in several ways:

1. From the simulator prompt
2. From the operating system command line
3. From the web interface
4. From a remote application using the power estimation web service

Whatever the execution method, three files are needed:

1. The Xilinx design file (NCD)
2. The Xilinx physical constraints file (PCF) (optional)
3. The tool parameters file (INI)

The first two files are generated in the preparation phase. Nevertheless, a user-friendly wizard (Fig. 4.2) is provided in order to help write correct INI files. This wizard is called the User Interface in this documentation.

Using the wizard, the user must first specify the necessary input files for the power estimation process (Fig 4.3). Next, the wizard parses the VHDL simulation model and offers the user the list of found entities. The user selects the top-level entity, and specifies the activity characteristics for every input port (Fig 4.4). In short, the possible input ports types are:

1. Periodic
2. Constant Value
3. Random

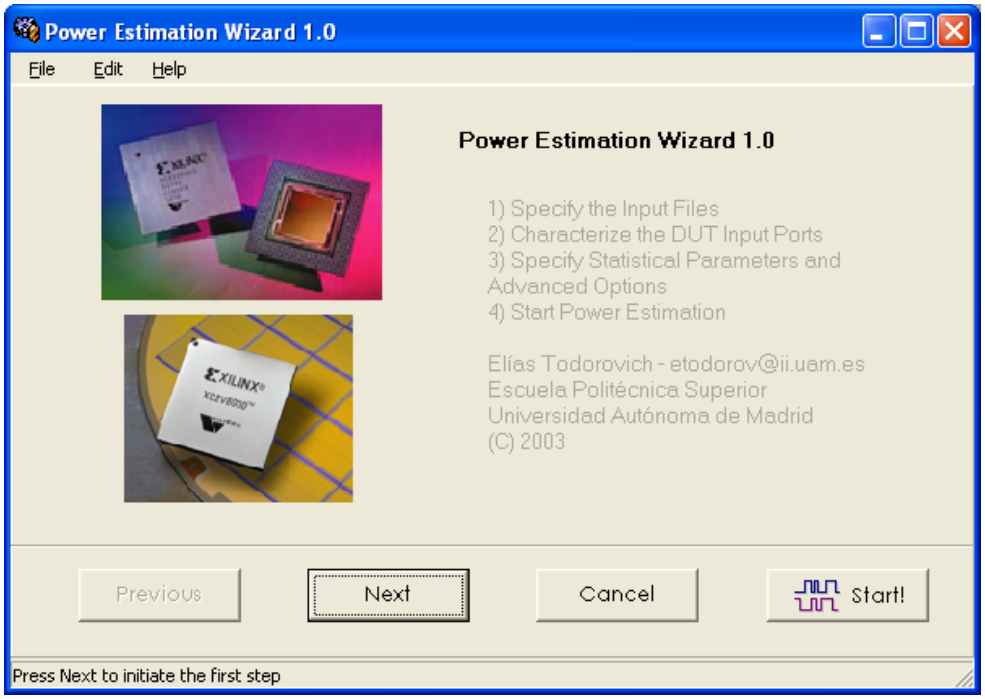


Fig 4.2: Power Estimation Wizard. Presentation step

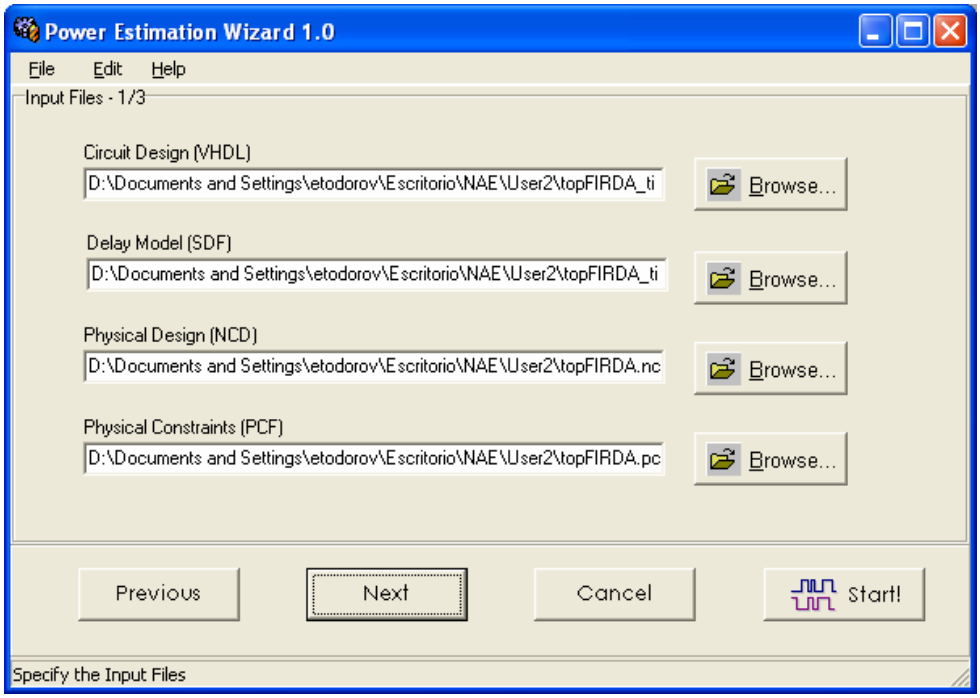


Fig 4.3: Power Estimation Wizard. Input file specification step

In the last step, the statistical parameters needed to run the selected power estimation technique must be specified:

- Tolerated error
- Confidence level
- Minimal activity mean, that divides the nodes in regular and low density ones

In addition, other advanced parameters can be selected although default values are provided for all the input boxes:

- Minimal glitch length
- Startup cycles
- Which information is stored in the Power Database

Once the user specifies all the parameters necessary for the power estimation tool, the wizard generates a correct-by-construction INI file. An example INI file can be found in Appendix C.

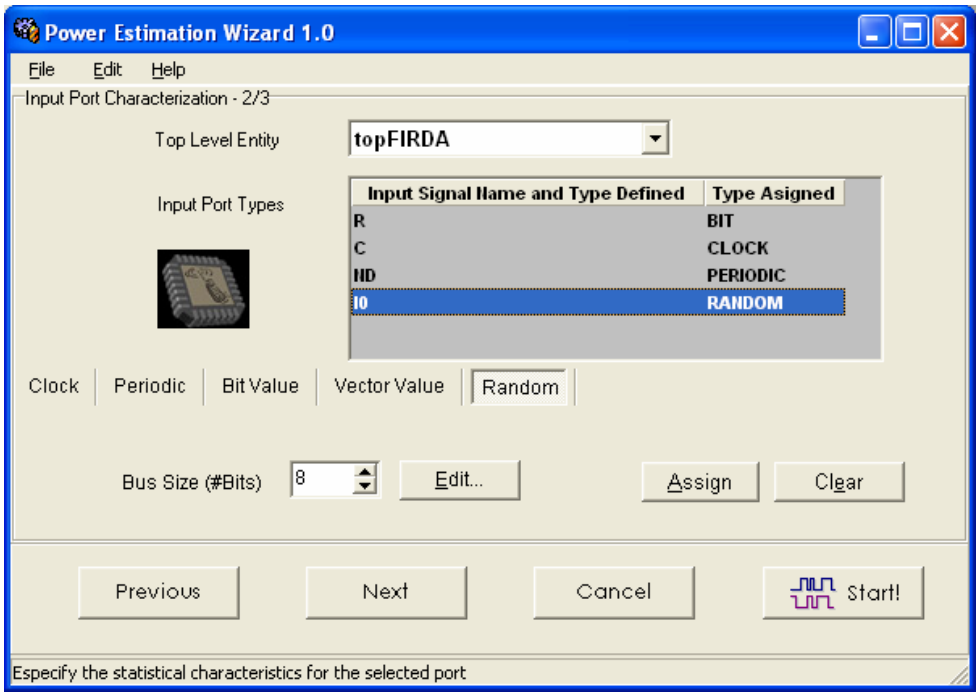


Fig 4.4: Power Estimation Wizard. Input port characterization step

In the current version, the following INI file sections are read in various power estimation programs:

- `FILES`
- `CIRCUIT FEATURES`
- `CIRCUIT CONFIGURATION`
- `STAT PARAMETERS`
- `OPTIMIZATION`
- `CLOCK`
- `PORTS`

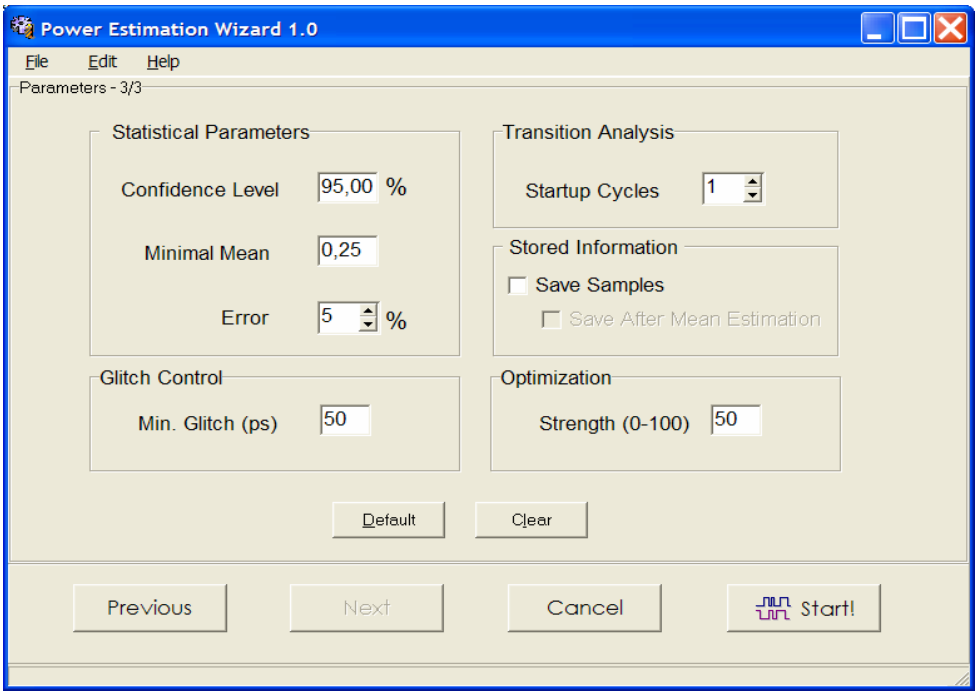


Fig 4.5: Power Estimation Wizard. Parameter specification step

4.2.2 Power Estimation Set-Up Phase

Besides de INI file, the power estimation wizard also generates other two files needed in the setup phase: `first_ini_msim.do` and `ini_msim.do`. These files must exist when the estimation Tcl/Tk script is interpreted and like the INI file, they are saved as plain text. Code 4.1 is a `first_ini_msim` example.

```
1  #
5  # FIRST_INI_MSIM.DO
6  #
7  # Activity Estimation using Statistics with Modelsim
8  #
9  # Changes the current directory.
10 cd D:/Users/PE
11 # Convert NCD to XDL (ncd2xdl)
12 catch "exec xdl -ncd2xdl topQDDFS_CORDIC.ncd
   topQDDFS_CORDIC.xdl" MESG
13 # Creates a new design library.
14 vlib work
15 vmap work work
16 #
17 # Compiles the VHDL file into the work library
18 vcom topQDDFS_CORDIC_timesim.vhd
19 # -sdftyp Annotates VITAL in the specified SDF file typical
   timing.
20 # -t Specifies the simulation time resolution.
21 # -noglitch Disables VITAL glitch generation.
22 vsim -sdftyp topQDDFS_CORDIC_timesim.sdf
   work.topQDDFS_CORDIC -t 1ps -noglitch
```

Code 4.1: Simulator macro file that sets-up the power estimation tool

These `first_ini_msim.do` and `ini_msim.do` are macro files for the simulator that compile (line 15) the VHDL model and start the simulation (line 19). In addition, a XDL file is generated with the `xdl` command (line 16). This file is necessary in the power computation sub system (see Chapter 6).

The difference between `first_ini_msim`.do and `ini_msim.do` is that the second is interpreted in the case estimation process is suspended for any reason or when the same design is studied with other parameters. In this case, the simulation must be started without compiling the design because it has been done before.

22.1 Activity Estimation Sub-system

The main purpose of the Activity Estimation Sub-system is the activity estimation of the individual nodes in the DUT. Also in this section, the generation of input vector files for the pattern generator will be introduced. The pattern generator equipment enables the physical measurement of the device current consumption with the same stimulus as in the simulation.

The activity estimator interacts with a commercial simulator in its inner loop and calls several programs to generate circuit stimuli and evaluate the simulator results until the stopping criterion is reached for every node as is shown in Fig 4.6. In fact, this Sub-system can be seen as a wrapper for the simulator program.

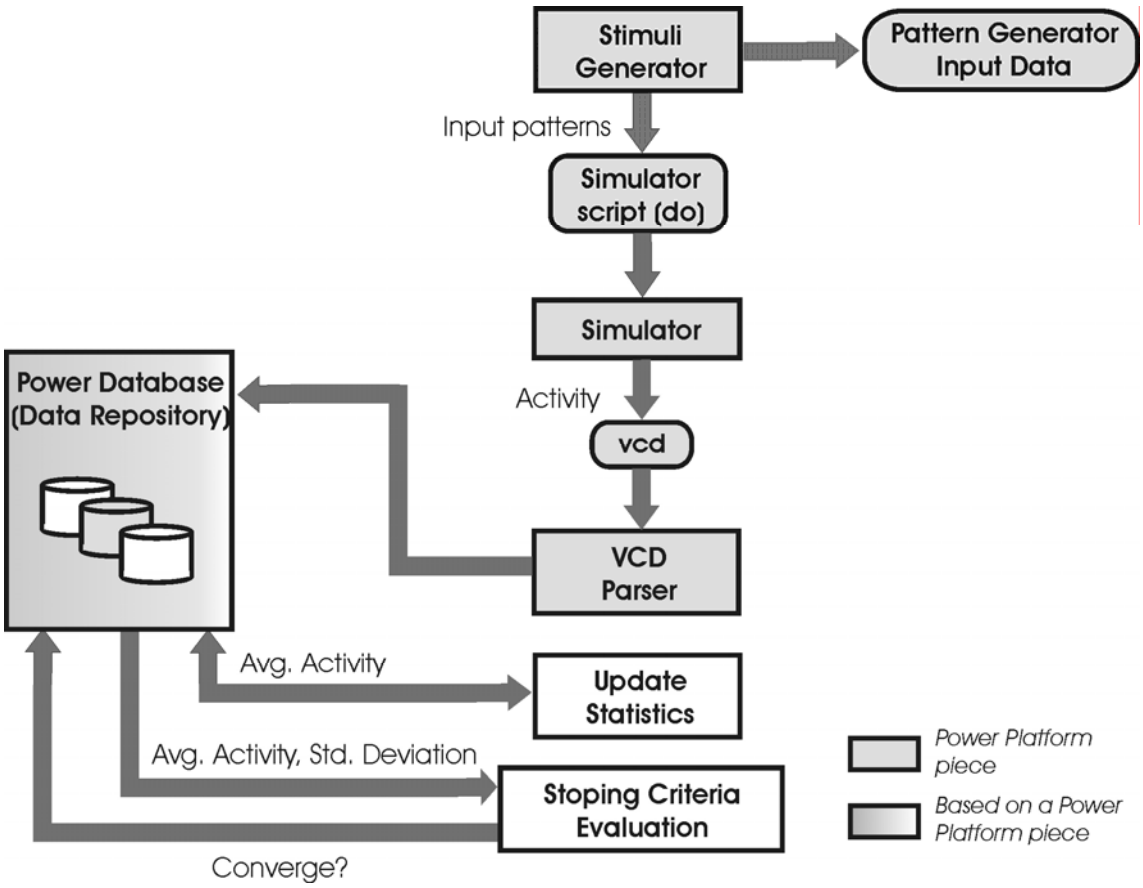


Fig. 4.6: Activity Estimation Sub-system

Each box in Fig. 4.6 represents a program with a specific purpose. All these programs are integrated or glued within a Tcl/Tk script in order to implement the statistical estimation technique. A simplified version of the Tcl procedure implementing this technique is Code 4.2. The complete Tcl/Tk script used in this work is printed in Appendix A.

```
1  ######################################################
2  #
3  # Estimate Average Transition Number for all Nodes
4  #
5  proc activityEstim {} {
6    upvar 1 blkNr blkNr
7    set END_SIM false
8    global sampleNum
9
10   while { $END_SIM == false } {
11
12     puts "Generating..."
13     exec generator.exe -pg tla -d [pwd]
14
15     puts "Simulating..."
16     do simulate.do
17
18     puts "Saving..."
19     saveVec
20
21     puts "Analizing transitions.vcd..."
22     catch "exec Transitions.exe -d [pwd] transitions[expr
   $blkNr - 1].vcd" sampleNum
23     puts "$sampleNum clock cycles analized..."
24
25     puts "Updating..."
26     exec Update.exe -d [pwd]
27
28     catch { exec Cuter.exe -d [pwd]} END_SIM
29   }
30
31   quit -sim
32   puts "END OF ACTIVITY ESTIMATION!"
33
34 }
```

Code 4.2: Tcl procedure implementing the statistical estimation technique

The loop between lines 10 and 29 repeats the core tasks of the activity estimation until the stopping criterion is reached.

Each iteration starts with the generation of a new block of input vectors. `Generator.exe`, at line 13, writes a macro file (`simulate.do`) with the vectors in an appropriate format for the simulator according to the user definitions. This program also produces the same vectors for the specified pattern generator equipment. The second output enables physical power measurements to further verification and tuning of the tool (section 4.3.1).

As the model was compiled and the simulation was initiated with the Set Up phase, the .do file can be interpreted by the simulator using the `do` command directly as shown at line 16. A complete .do file with a stimuli block is reproduced in Appendix B.

Once the current input pattern block is simulated, it is necessary to save the corresponding simulation results. It is done by the Tcl procedure call at line 19 (Code 5.2 or Appendix A). The activity results are stored using the VCD standard format [VCD01]. This format is widely accepted in the industry and in several commercial and open source simulators and EDA tools.

At line 22, with `transitions.exe`, the VCD simulation results are parsed. The signal transitions are counted within each clock cycle for all the circuit nodes, and the pulse durations are calculated in order to decide if it is long enough or it should be filtered. This program returns the current number of clock cycles analyzed in order to give the user some feedback about the estimation progress.

At line 26, `update.exe` computes the average and standard deviation for all the nodes in the circuit. This is done using the new data extracted from the last VCD results file. With these new results, the program updates the Power Database

Finally, the current iteration ends with `Cuter.exe`, which tests if all the nodes in the DUT have reached the stopping criteria defined in Eq. 3.2 and 3.4.

Implementation details of these programs and scripts are explained in Chapter 5.

4.3.1 Input Patterns for the Pattern Generator

It is useful to measure physically the power consumption of as many designs as possible in order to develop an accurate power estimation tool. Furthermore, it is necessary to tune, adjust and correct the tool for future devices and designs. In this way, the same input patterns generated for the software tool are generated for a pattern generator equipment. In the current development, a Tektronix logic analyzer and pattern generator are used. So, in this implementation, just files for Tektronix equipment can be generated.

4.3.2 The Power Estimation Platform

Several pieces of the Activity Estimation Sub-system can be re-used in other power estimation tools. The simulator is naturally part of any power estimation tool based on the statistical approach. It is an external development and in this way is considered the first component of the Power Estimation Platform. Nevertheless, the simulator must satisfy the VCD, VITAL and SDF standard formats.

In Fig. 4.6, the re-usable components are represented by light gray boxes. Some components are not used “as is” but as the base development for a new one with extended behavior. This is the case in the Power Database, where there are common data structures and information that are independent of the specific application, but the data necessary for the current technique can not be present.

4.4 Power Computation Sub-System

At the end, the estimated activity for all the DUT nodes is multiplied by the corresponding node capacitances. Unfortunately, those capacitances are not provided directly by the FPGA vendor. So, they are obtained with the Power Computation Sub-system. The basic operation of this sub-system is depicted in Fig. 4.7.

Clearly, the Power Computation Sub-system depends on the vendor tools and the proprietary formats they use, but for the current IDE with its special features, all the development can be reused in other power estimation tools. For example, a maximum power estimation tool can use all these programs as they are. It is just possible that the order in which they are used is different.

The representation of the layout, the way capacitances are retrieved, etc. are examples of the dependency on the vendor software and proprietary formats. The tool that Xilinx provides for the power computation, XPower, is essential in this particular estimation flow. Why is it called power computation and not a power estimation tool? The reason is that users can provide XPower with an arbitrary input vector set, and in this way, the pattern dependence problem is not treated at all. This problem is described in Chapter 3. XPower calculates the power with the activity provided by the user and capacitance values stored in proprietary databases.

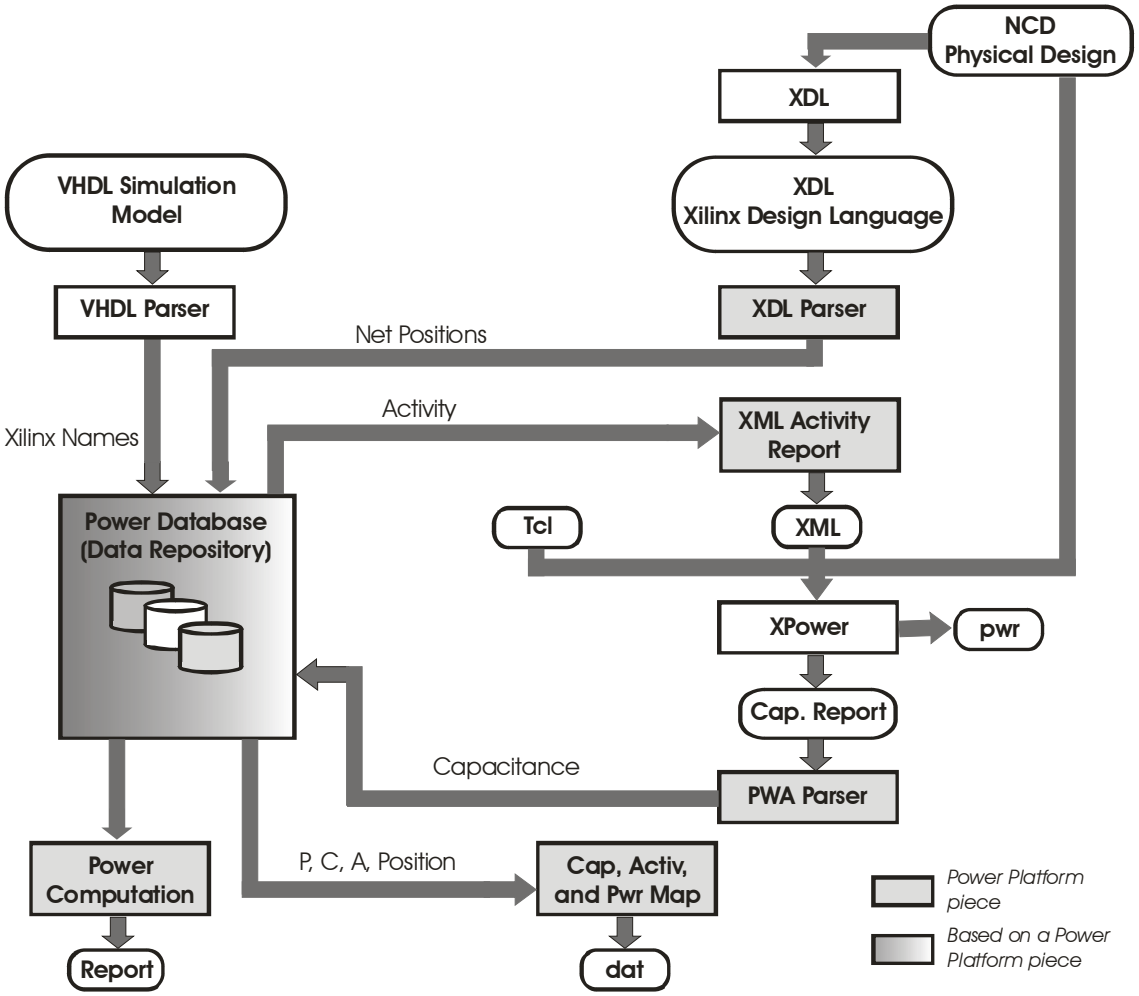


Fig. 4.7: Power Computation Sub-System

The main goal of the Power Computation Sub-system is to obtain capacitance through XPower reports. This can be done providing XPower with the activities estimated according to statistical considerations. With this information, XPower reports the capacitances. Finally, parsing this report the capacitances can be stored in the Power Database. Retrieving the capacitances, A-DyP can compute the power for every node, and the whole circuit. Also, now the power can be grouped in logic, nets and clock or other classifications depending on the additional information obtained from other sources about the nodes.

In the Power Computation sub system, a problem related to the different ways the same objects are identified within the vendor tools, must be solved. The first time the problem appeared was while trying to generate the activities for XPower with a so-

called setting file. This is an XML file where the activities are specified for individual nodes. Code 4.3 shows a fragment of a settings file for XPower.

```
<Power_Net name=”CORDIC/DATAPATH_Z/result_32(2)”>
  <Power_Activity freq=”159.009009Mhz” src=”Simulation”
   duty=”0.000%” />
 </Power_Net>
 <Power_Net name=”CORDIC/ DATAPATH_Z/result_32(3)”>
  <Power_Activity freq=”170.945946Mhz” src=”Simulation”
   duty=”0.000%” />
 </Power_Net>
```

Code 4.3: XPower setting file fragment

It should be easy to generate the XML file from the Power Database, but that is not the case because the identifiers stored in the Power Database, and obtained from VCD files, are different. The names for the XML file are built according to different alphabets, hierarchical separators and rules. This problem leads to a heavy processing as shown in the upper part of Fig 4.7. It is called in this work Multiple Identifiers' Problem.

The following examples can help to see the problem. They were extracted from test cases, where their VCD, XDL and XML names are shown in the first, second and third places respectively.

Example 4.1:

`lfirda_gen0_lfir_prt1_gen0_bqrom_bqrom_6_rom_gen1_1_rom_outrom_0`

`lFIRDA/GEN0.LFIR/PRT1/GEN0.BQROM.BQROM.6.ROM/GEN1.1.ROM/GEN0.3.ROM`

`lFIRDA/GEN0.LFIR/PRT1/GEN0.BQROM.BQROM.6.ROM/GEN1.1.ROM/outrom_0`

Example 4.2:

`lfirda_gen0_lfir_prt1_gen0_bqrom_bqrom_4_rom_gen3_3_gen31_gen31_0_add_outxor2_4`

`lFIRDA/GEN0.LFIR/PRT1/GEN0.BQROM.BQROM.4.ROM/GEN3.3.GEN31.GEN31.0.ADD/GEN0.2.lXOR2`

`lFIRDA/GEN0.LFIR/PRT1/GEN0.BQROM.BQROM.4.ROM/GEN3.3.GEN31.GEN31.0.ADD/outxor2_4`

Example 4.3:

`lfirda_gen0_lfir_prt1_gen0_bqrom_bqrom_1_rom_inoutphases_64_cymuxg`

`lFIRDA/GEN0.LFIR/PRT1/GEN0.BQROM.BQROM.1.ROM/GEN3.0.GEN31.GEN31.6.ADD/GEN0.1.MUXCY.lMUXCY`

`lFIRDA/GEN0.LFIR/PRT1/GEN0.BQROM.BQROM.1.ROM/GEN3.0.GEN31.GEN31.6.ADD/GEN0.1.MUXCY.lMUXCY/O`

To solve the multiple identifiers problem and generate the XML file, the VHDL simulation model and the XDL design files are parsed. During this process, an identifiers dictionary is built in the Power Database.

The Activity Estimation Sub-system deals with VHDL, SDF and VCD files, all using the first set of identifiers. The second set is what must be generated for XPower. Chapter 6 explains the details of the Power Computation Sub-system and its solution to the described problem.

```
1  #
2  # Capacitance Estimation
3  #
4  proc capacitanceEstim {} {
5
6    puts “Starting POWER Computation…”
7    puts “”
8
9    # VHDL Parsing
10   puts naestate “Analizing vhdl file…”
11   exec parserVHD.exe –d [pwd]
12
13   # XDL Parsing
14   puts naestate “Analizing phisical info. (xdl file)”
15   exec xdlParser.exe –d [pwd]
16
17   # XML Activity Report
18   puts naestate “Gen. Activity Rep. in XML format…”
19   exec XMLRep.exe –d [pwd]
20
21   # Connection with XPower
22   puts naestate “Gen. Power Report with XPower…”
23   do Connect2Xpower.do
24
25   # PWA Parsing
26   puts naestate “ Analizing Capacitance Report…”
27   exec PWAparser.exe –d [pwd]
28
29   # Calculates Power and write a Report
30   puts “Writing Power Report…”
31   exec report.exe –d [pwd] –v
32
33   # Maps
34   puts “Activity, Cap., and Power Maps…”
35   exec activityMap.exe –d [pwd] –r 1
36   exec capacitanceMap.exe –d [pwd] –r 1
37   exec powerMap.exe –d [pwd] –r 1
38 }
```

Code 4.4: Tcl procedure that implements the Power Computation Sub-system

The same way as in the Activity Estimation Sub-system, all the Power Computation processing is integrated within a Tcl/Tk script. A simplified version of the Tcl procedure implementing this technique is Code 4.4. The complete Tcl/Tk script for power estimation is reproduced in Appendix A.

At lines 11 and 15, the VHDL and XDL parser are executed. From the XDL file, as depicted in Fig. 4.7, important information is extracted: the node positions in the layout. With this information, power maps can be drawn where the resolution could be selected by the user.

Once identifiers for the XML file are obtained, it is generated (see line 19).

At line 24, other script file (.do) is interpreted and from it, XPower is run as a command line application. Chapter 6 explains the details related to this script.

Within the step described above, a capacitance report is generated. At line 27, a parser for this report generated by XPower is executed. The analyzed PWA file has capacitance information for every internal node in the DUT.

At line 32, `report.exe` calculates the power for the individual nodes and writes a report with detailed information. In addition, it obtains the power consumption of the different FPGA resources.

Lines 35 to 37, are examples of the different information that can be related to the physical position of the nodes in order to draw maps. In this case a resolution 1 is specified which means that the specific magnitude for the nodes in the same CLB are added. The files with the information relating the FPGA positions and the specific magnitude can be imported using any scientific graphing tool.

4.5 The Power Database

In Fig. 4.8, light gray boxes represent tables that can be used in any power estimation tool. In other words, they belong to the Power Platform. `Node` contains information independent of the current run for a specific circuit node. For example, the activity depends on the current run but the node capacitance does not. FPGA`Resource` contains slices, pads, and embedded cores with its position in the

die. These resources are part of a `Circuit`, the circuit is implemented in an `FPGADevice`, and this device belongs to an `FPGAFamily`.

The main fields in theses tables are:

`FPGAFamily`

`Name, Vdd_core, Vdd_pads`

`FPGADevice`

`Name, Size_X, Size_Y, Min_Glitch`

`Circuit`

`Name, VCD_Nodes, Latency`

`FPGAResource`

`Name, Type, Pos_X, Pos_Y`

`Node`

`Name_VCD, Name_AKA, Capacitance, Low_level_res_type`

The application specific tables are `Signal`, containing information related with a node for a particular run (`EstimParams`), and `Sample` with the number of transitions in a node and clock cycle.

The main fields in theses tables are:

`EstimParams`

`Error, Confidence, Min_Act`

`ImputPort`

`Name, Port_Type`

`ImputLine`

`Position, Params`

`Signal`

`Activity_Avg, Activity_StdDev, Saples_at_Conv`

```
Sample

   Clk_cycle, trn_num
```

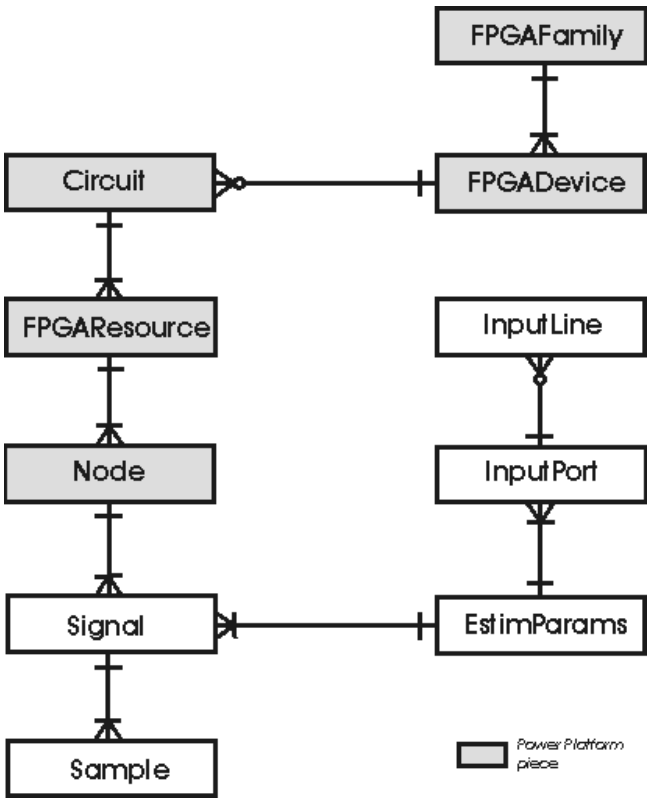


Fig. 4.8: Power Database

An abstraction layer is convenient to access the Power Database independently of the database engine currently used. In this way, the database engine can be changed with a bounded impact in the system implementation.

4.6 Conclusions

The architecture of the system is carefully designed and a Power Estimation Platform is developed. This feature and the use of standard formats enable the current development to be re-used in other power estimation and optimization tools.

A very important difficulty found in this work is introduced in this chapter: the Multiple Identifiers' Problem. It comes from the lack of integration with third-party tools as a vendor goal, at least in the power estimation area. In spite of the effort to solve this

problem, there are cases without a solution. For example, for some FPGA families and software versions, there are nodes where the Multiple Identifiers' Problem persists. Furthermore, a new flavour of this problem is introduced with each new version, making it even more difficult to solve. Nevertheless, it is reasonable to believe that in the near future, the vendor will clearly specify formats for data interchange and integration with third party developers in the power estimation area.

References

[SDF01] IEEE Std 1497-1999, IEEE Standard for Standard Delay Format (SDF) for the Electronic Design Process. The Institute of Electrical and Electronics Engineers, Inc. 3 Park Avenue, New York, NY 10016-5997, USA, 2001.

[VCD01] IEEE Std 1364-2001 (Revision of IEEE Std 1364-1995), IEEE Standard Verilog Hardware Description Language. The Institute of Electrical and Electronics Engineers, Inc. 3 Park Avenue, New York, NY 10016-5997, USA, 2001.

[VIT01] IEEE Std 1076.4-2000, IEEE Standard for VITAL ASIC (Application Specific Integrated Circuit) Modelling Specification. The Institute of Electrical and Electronics Engineers, Inc. 3 Park Avenue, New York, NY 10016-5997, USA, 2001.

[Xil] Xilinx Inc. at http://www.xilinx.com

[XilSaV] Xilinx Inc., "Synthesis and Verification Design Guide", available at http://www.xilinx.com

Chapter 5.

"Computers may be thought of as engines for transforming free energy into waste heat and mathematical work", Charles H. Bennett [Ben82]

5 Activity Estimation Sub-system

This chapter describes the development details of the Activity Estimation Sub-system. This is one of the two sub-systems in the A-DyP estimation tool. It is responsible for the estimation of the individual nodes average activity and standard deviation in the DUT.

As the preparation phase, user interface, and set-up phase for A-DyP were described in Chapter 4, here just the activity estimation core (see Fig. 4.6) is explained in detail. The Activity Estimation Sub-system interacts with a commercial simulator in its inner loop and calls several programs and scripts. This sub-system can be seen as a wrapper for the simulator program.

All the programs and scripts composing the Activity Estimation Sub-system are integrated within the Tcl/Tk script shown in Code 4.1 (The complete Tcl/Tk script used in this work can be found in Appendix A) and they are:

1. The pattern generator program
2. The script that runs the patterns in the simulator
3. The script that saves the simulator results
4. The program that analyzes the simulation results

5. The program that updates the average and standard deviation for all the nodes in the DUT
6. The program that checks if the stopping criteria for all the nodes in the DUT was reached

5.1 The Pattern Generator

The goal of this program is generating a macro file for the simulator with stimuli produced according to the user specifications. In Appendix B there is an example of this macro file. `generator.exe` is a console application with more than 750 lines of source code. These applications typically don't require user interaction. As with all the programs in this work, `generator.exe` is coded in Object Pascal, which is the language in the Delphi IDE. The Pattern generator, like the other programs in this system, read parameters from an INI file. The parameters obtained this way are called indirect parameters.

`generator.exe` also accept command line parameters:

```
generator -pg tla -d dir
```

`-d dir` specifies the working directory

`-pg` enables the specification of target pattern generation equipment. In the current version this parameter just accepts the `tla` value for a Tektronix pattern generator.

From the INI file, this program reads the PORT section to obtain the complete input port characterization of the DUT:

1. Number of ports
2. Input port names and types (clock, connected to a constant or random value)
3. Input port parameters according to its type

Also, the program reads the statistical parameters. These data belong to the Power Database. It is accessed through the abstraction layer and is currently implemented by a simple INI file.

The produced stimuli can be random, vector or bit constants, or periodic digital values. The same vectors generated for the simulator are formatted to the pattern generator in order to verify the power estimation system.

It should be noted that all the designs given as input to this tool are *synchronous* circuits. So the input stimuli are generated in synchrony with the system clock. As pointed in [Xak94] the input pulse widths are discrete multiples of the clock period. In that paper it is shown that the probability that a low (high) signal will transition high (low) on the clock edge is:

$$P(1/0) = \frac{T_C}{\mu_0} \qquad \text{(Eq. 5.1.)}$$

$$P(0/1) = \frac{T_C}{\mu_1} \qquad \text{(Eq. 5.2.)}$$

Where T_C is the clock period and μ_0 and μ_1 are the mean low and high pulse widths. Both probabilities are computed in the User Interface program and are retrieved here from the INI file. The main algorithm in this program uses these probabilities to generate the input patterns for every clock cycle and a simplified version is shown in Code 5.1.

```
1  if InputsArray[iPort].ranValues[j-1][i] = '1' then
2  begin
3    if random <= InputsArray[iPort].detail[i].param2 then
4      InputsArray[iPort].ranValues[j][i] := '0'
5  else InputsArray[iPort].ranValues[j][i] := '1';
6  end
7  else begin // '0'
8    if random <= InputsArray[iPort].detail[i].param1 then
9  InputsArray[iPort].ranValues[j][i] := '1'
10 else InputsArray[iPort].ranValues[j][i] := '0';
11 end;
```

Code 5.1: Program fragment for the random vectors generation

In `InputsArray`, all the information about the input ports is stored. `ranValues` stores a random input stimuli block where the first index points to a whole vector and the second one accesses to a specific bit position. At line 1, the last bit value, randomly generated, is tested. In the case it is ‘1’, it is decided if the next value will change to logic ‘0’ or will stay at ‘1’. The probability of transition from high to low is stored in

`InputsArray[iPort].detail[i].param2`. In the lines from 7 to 11 the case where the current bit is logic '0' is considered.

Another point in this program is the computation of the number of vectors to stimulate the circuit in a single estimation cycle. Eq. 3.1 and 3.4 determine the sample size needed in the statistical technique to converge. Nevertheless, when long enough input sets are applied, the amount of activity in a relatively big design could produce huge files. For example, a simulation of the FIRDA(8) test circuit (see section 7.1.2), with 9495 nodes, generates a 7.9MB vcd file applying 70 input vectors. In order to reduce it to tractable file sizes, samples are bounded and the activity estimation is computed in an iterative fashion. A reasonable maximum number of vectors is determined empirically to be generated in a single iteration with the function `GetMaxRun` (Code 5.2). This function basically depends on the circuit size, `NodesCount`.

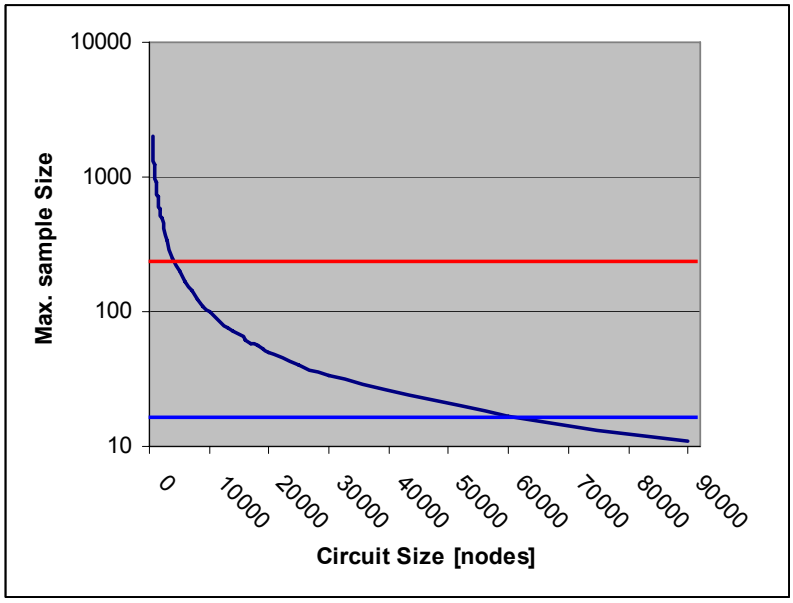


Fig. 5.1: Maximum sample size computation

`K_SAMPLE/NodesCount` is depicted in Fig 5.1 where the current value of `K_SAMPLE`, the sample size tuning constant, is one million. `SAMPLE_BLK_SIZE` and `MAX_SAMPLE_SIZE` are the minimum and maximum sample sizes and their current values are 16 and 250 respectively. It can be observed that for small designs, as many

as MAX_SAMPLE_SIZE vectors can be generated in one iteration, and as few as SAMPLE_BLK_SIZE for a big circuit.

```
1  function GetMaxRun: Integer;
2  begin
3   NodesCount := getNodesCount;
4   result := MIN(MAX_SAMPLE_SIZE, MAX(SAMPLE_BLK_SIZE,
   Trunc(K_SAMPLE /NodesCount)));
5  end;
```

Code 5.2: Function for the sample size bounds determination

The actual vector number must be less than the computed upper bound and not greater than the calculated according Eq. 3.1 and 3.4. Sample sizes (Eq. 3.1 and 3.4) are computed with the Code 5.3 for every node in the circuit.

```
1  if(media < umbral)then
2    N := sqr((ZalfaDiv2*desvio)/(umbral*errorUser))
3  else N:= sqr((ZalfaDiv2*desvio)/(media*errorMuestra));
```

Code 5.3: Function for the sample size bounds determination

5.2 Simulating the Input Patterns and Saving the Simulation Results

As the input vectors are produced using the macro language of the simulator, they can be directly interpreted (See Code 4.2, line 16), the simulated time advances, and a (huge) number of transitions are produced for the DUT.

In order to further analyze this activity, it is necessary to save the simulation results. It is done by the Tcl procedure shown in Code 5.4. The activity results are stored in the standard VCD format which is an IEEE standard [VCD01]. This standard defines the Verilog Hardware Description Language (HDL). However, the VCD file format is also specified in this standard. A VCD file is an ASCII file containing header information, variable definitions, and value changes in these VCD variables. ModelSim, the selected simulator in the current implementation, provides simulator command equivalents for VCD system tasks and extends VCD support to VHDL designs. So, the ModelSim commands can be used on either VHDL or Verilog designs.

In [Mod03], ModelSim VCD commands and VCD tasks, creating a VCD file, and example VCD outputs, can be studied.

```
1  ########################################
2  #
3  # Save a VCD file for further analysis
4  #
5  proc saveVec {} {
6    upvar 2 blkNr blkNr
7
8    # Flushes VCD file buffer to the last VCD file
9    vcd flush transitions$blkNr.vcd
10
11   # Turns off VCD dumping
12   vcd off transitions$blkNr.vcd
13   vcd flush transitions$blkNr.vcd
14
15   # Increments the number of simulation blocks
16   incr blkNr
17
18   # Adds all VHDL signals to the next VCD file
19   vcd add -file transitions$blkNr.vcd *
20 }
```

Code 5.4: Tcl procedure that saves simulation VCD results

`blkNr` just enables generating different .vcd file names. `vcd off` turns off VCD dumping over the specified file. `vcd flush` flushes the VCD file buffer to the last VCD file. It is not possible to close these files in the usual way during a simulation, so different files are generated. `vcd add` adds all VHDL signals to the next VCD file preparing it for the next estimation iteration.

As the estimation process moves forward, some nodes converge and it could be important to exclude them from the VCD file. It could significantly reduce the VCD file size and the corresponding file parsing time. Nevertheless, there is not an effective way to do it. The missing option could be `–nodes` and the following command

```
vcd add -file transitions$blkNr.vcd *
```

could be replaced with

```
vcd add -file transitions$blkNr.vcd –nodes node_list.txt
```

where the `node_list.txt` file contains the list of the nodes that have not yet reached the stopping criteria.

5.3 Analyzing the Generated Activity

The goal of this program is the analysis of VCD files generated by the simulator. It should count the effective number of transitions for every node in each clock cycle. It is also a console application. `transitions.exe` has more than 800 lines of source code and the most important fragments are explained in this section.

`transitions.exe` accept command line parameters:

`transitions -d dir` *`vcd_file_name`*

`-d dir` specifies the working directory

`vcd_file_name` is the current VCD file that will be analyzed in this iteration of the estimation process.

From the INI file, this program reads several indirect parameters where the most important are related to the clock, the duration of the minimum glitch pulse and the circuit latency. For example, in a pipelined circuit the program should wait a number of cycles before collecting samples of the circuit activity. The discarded activity is, in fact, generated outside the normal operation of the DUT.

The parser for the VCD files (like the other parsers in A-DyP) is based on the techniques proposed in [Aho86].

5.3.1 The Set-up Period

In order to consider the set up period, Code 5.5 is executed (It is a simplified version). `interpDecl` (line 1) is a Boolean variable that, when it is true, the variable definitions part in the VCD file is parsed using the `analizeDeclVCD` procedure.

firstCycles (line 3) is another Boolean variable that avoids considering the activity in the first simulated clock cycle because the nodes are not initialized (unknown logic state). Other cycles are ignored by user request according to the INI file parameter `SetupCycles` (line 4). This feature, as mentioned above, is useful for pipelined circuits. Although in the current implementation, this data is retrieved from the INI file, it is obtained using an abstraction layer,

`PowerEstimationManager`, in order to make it easy to implement any change. For example, this information could be stored using any database engine.

From line 8 to 15 it is assumed that the VCD declarations were parsed in a previous estimation cycle. When the number of set up cycles is bigger than the number of the simulated ones in the actual iteration, nothing must be considered in the statistical sample for the current estimation sample.

`countTrnVCD`, at line 17, is the procedure that parses the value changes section in the VCD file.

```
1  if interpDecl then
2  begin
3    firstCycles := true;
4    SetupCycles :=
   PowerEstimationManager.GetLatency;
5
6    analizeDeclVCD;
7  end // interpDecl is true
8  else begin
9    SetupCycles :=
   PowerEstimationManager.GetRemainingSetUpCycles;
10   if SetupCycles = 0 then
11     firstCycles := false
12   else firstCycles := true;
13
14   salteaDeclVCD;
15 end;
16
17 countTrnVCD;
```

Code 5.5: Program fragment that consider the set up period

5.3.2 Counting the Effective Transition Number

In [Lan94], rather than counting a short glitch as a full rail-to-rail transition, it is modeled as a swing from ground to Vdd/2 and back to ground. Here another approach is used to take this effect into account.

Code 5.6 is a simplified version of the algorithm that counts the effective number of transitions for each node and clock cycle.

```
1  if (tsim - Nodes[id-1].lastTr) > minGlitch then
2  begin
3    Nodes[id-1].trn := Nodes[id-1].trn + 1;
4    Nodes[id-1].lastTr := tsim;
5  end
6  else begin
7    if Nodes[id-1].trn > 0 then
8      Nodes[id-1].trn := Nodes[id-1].trn - 1;
9    NodesLine[id-1].lastTr := 0;
10 end;
```

Code 5.6: Program fragment that counts effective transitions

`Nodes` is a data structure in the main memory. It stores, for every node, the number of transitions in the current clock cycle and the simulated time of the last transition. At line 1, the pulse duration is checked. If it is long enough, the transition is considered as an effective one (lines 3 to 4). In the case the glitch was too short, the pulse is ignored and also the transition that initiated it is taken away (lines 7 to 9).

It should be noted that this data structure in memory (Nodes) leads to an important optimization. Dealing directly with the Power Database could require an unfeasible execution time. In the current implementation, just at the end of a clock cycle, the Power Database is accessed to store the analysis results.

5.4 Updating Node Statistics

The goal of this simple program is to update the statistics for all the nodes according to the new samples collected in the current iteration of the activity estimation process. Also it is decided if the stopping criterion is reached for every node. `update.exe` has just 77 lines of source code and a simplified version of the main procedure is explained in this section (Code 5.7).

Besides the indirect parameter read from the INI file, `update.exe` accepts command line parameters:

`update -d dir`

`-d dir` specifies the working directory

```
1  procedure UpdateCut;
2  var
3    media, desvio, N: Real;
4  begin
5    if (muestra >= minSamples) then
6    with NodeT do begin
7       Open;
8       while (not eof) do   // Visit every node
9       begin
10        if not (FieldByName('Cut_Cond').asBoolean) then
11        begin // no converged
12          media := FieldByName('average').asFloat;
13          desvio :=
14            sqrt((FieldByName('SumatCuad').asInteger
15            /muestra)-sqr(media));
16          if(media < umbral)then
17             N:= sqr((ZalfaDiv2*desvio)/
18                     (umbral*errorUser))
19          else
20             N:= sqr((ZalfaDiv2*desvio)/
21                     (media*errorMuestra));
22          if(N < muestra) then
23          Begin // has converged
24            Edit;
25            FieldByName('Cut_Cond').asBoolean := True;
26            FieldByName('SampleAtConv').AsInteger :=
27                    muestra;
28            Post;
29          end;
30        end; //del if
31        Next;
32     end; //del while
33     Close;
34   end; // del with
35 end;
```

Code 5.7: Program fragment that updates node statistics

At line 5 it is checked that the effective number of clock cycles is at least a minimum number, currently 30. This is required by the statistical technique. Just when it is true, the stopping criterion can be evaluated. At lines 12 and 13 the mean and standard deviation are calculated. Lines from 16 to 21 compute the sample size for the current node depending on it is a regular or a low density one. Line 22 decides the node convergence and, in this case the node information is edited in the Power Database to reflect this fact. When a node converges, the sample size at the present time is stored for further analysis.

5.5 Checking the Stopping Criteria

`Cuter.exe` is the simpler program in the power estimation system with just 37 lines of source code. Its goal is to determine if all the nodes are marked as reaching the stopping criteria or not. In the first case it outputs “true”. In any other case the output is “false”. This output is captured in the activity estimation Tcl procedure to decide if other estimation iteration is necessary or not.

`cuter.exe` accept command line parameters:

```
cuter -d dir
```

`-d dir` specifies the working directory

5.6 Conclusions

In this Chapter, the Activity Estimation Sub-system is explained in detail, completing the general description in the previous Chapter. It can be recognized that some software pieces could be used in other power estimation tools, for example, a maximum power estimation tool. Being independent programs, they can be coordinated with a Tcl script in another order or with another algorithm. In this way these programs form the Power Platform framework.

References

[Aho86] Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, Compilers: Principles, Techniques, and Tools, Addison-Wesley, 1986.

[Ben82] C.H. Bennett, “The Thermodynamics of Computation – a Review” Internat. J. Theoret. Phys. 21, pp. 905-940 (1982).

[Lan94] P. Landman, Low-Power Architectural Design Methodologies, Ph. D. Thesis, Electronic Research Laboratory, University of California, Berkeley, August 1994.

[Mod03] Model Technology, ModelSim SE User’s Manual, 2003.

[VCD01] IEEE Std 1364-2001 (Revision of IEEE Std 1364-1995), IEEE Standard Verilog Hardware Description Language. The Institute of Electrical and Electronics Engineers, Inc. 3 Park Avenue, New York, NY 10016-5997, USA, 2001.

Chapter 6.

"While power consumption is an urgent challenge, its leakage or static component will become a major industry crisis in the long term, threatening the survival of CMOS technology itself, just as bipolar technology was threatened and eventually disposed of decades ago. Leakage power varies exponentially with key process parameters such as gate length, oxide thickness, and threshold voltage; this presents severe challenges in light of both scaling and variability. Off currents in low-power devices increase by a factor of 10 per technology cycle. Therefore design technology must be the key contributor to maintain constant static power." [ITRS05] (pp.17)

6 Power Computation Sub-system

This chapter describes the development details of the Power Computation Sub-system. This is the second of the two sub-systems in the A-DyP power estimation tool. It is responsible for calculating the average power of the whole circuit and individual nodes in the DUT.

In order to obtain a power or energy estimation, the activity for all the DUT nodes must be multiplied by the corresponding node capacitances (Eq. 2.7). Unfortunately, those capacitances are not provided by some FPGA vendors, or not in a direct way in the best case. Currently, they are not thinking on the connectivity with third party tools at this point. Therefore, capacitances are obtained with the Power Computation Sub-system as shown in Fig. 4.7. In this way, the Power Computation Sub-system depends on the specific vendor flow, and could not be ported to other FPGA design flow but must be re-implemented. However, capacitances are not the only data collected here.

Node positions and other relevant information is obtained and stored in the Power Database. On the other hand, this development could be used in other power estimation tools of the same vendor, for example, for peak power estimation.

Now it is appropriate to mention that Xilinx provides a power computation tool: XPower. In this thesis, it is called a power computation tool because users can apply the activity generated from an arbitrary input vector set as its input, and in this way the pattern dependence problem is not treated at all. That problem is described and analyzed in Chapter 3. XPower calculates the power with the activity provided by the user and capacitance values stored in proprietary databases.

The main goal of the Power Computation Sub-system is to retrieve node capacitances considering the particular way the specific FPGA vendor could provide them and compute the individual and total power.

A possible method to retrieve capacitance values is by XPower. The activities obtained with the Activity Estimation Sub-system, and stored in the Power Database, are given to XPower by a so-called settings file. This is an XML file where the activities can be specified for individual nodes. It should be easy to generate the XML file from the Power Database, but that is not the case because the identifiers used by the activity estimation sub-system and the simulator, and the different files related with the power estimation process are built according to different alphabets and lexical rules. This problem leads to heavy processing as shown in the upper part of Fig 4.7 and is called in this work the *Multiple Identifiers' Problem*.

To solve the multiple identifiers problem and finally obtain the capacitances for the individual nodes, the VHDL simulation model and the proprietary XDL design file are parsed. During this process, an identifiers dictionary is built in the Power Database.

6.1 Parsing the VHDL Simulation Model

The goal of this program is the analysis of VHDL models generated by the `netgen` Xilinx tool from a post PAR layout. The result of this analysis is a dictionary of identifiers, stored in the Power Database, which solves the multiple identifiers problem. It is also, as all the programs in this work, a console application. `parserVHD.exe`

has more than 600 lines of source code and the most important fragments are explained in this section.

`parserVHD.exe` accept command line parameters:

```
parserVHD.exe -d dir
```

`-d dir` specifies the working directory

6.1.1 Why Parsing the VHDL Model?

From the first studies done to solve the multiple identifiers problem, the VHDL model was identified as a vehicle to relate the names in the different parts of the system. For example, Code 6.1 shows a slice's flip-flop instantiation. The VHDL label at line 1 can be used to find the identifier generated in other formats in the Xilinx design flow. The relationship can be understood observing this label and the output port of the instantiated component. The name of the signal connected to the output port is linked to the Xilinx's identifier through the label as shown in Code 6.2, line 6. Replacing the ".", "/", etc with "_" we have a complete match. Unfortunately, it was found that the rule explained in this paragraph has a number of exceptions.

```
1  QDDFSC_CORDICR_PIPELINE_PIPELINE_10_DPTH0_DPTH0_DPATHX_GEN0_
   GEN0_1_REGC_REGC_REG : X_FF
2    generic map(
3      INIT => '0'
4    )
5    port map (
6      I =>
   QDDFSC_CORDICR_PIPELINE_PIPELINE_10_DPTH0_DPTH0_DPATHX_outxo
   rcy,
7      CE => ce_s,
8      CLK => c_s,
9      SET => GND,
10     RST => QDDFSC_CORDICR_x_datapath_11_0_FFY_RST,
11     O => QDDFSC_CORDICR_x_datapath_11(1)
12   );
```

Code 6.1: Example instantiation of a Slice flip-flop in VHDL

```
1  inst "QDDFSC/CORDICR/x_datapath_11(0)" "SLICE" , placed
   R17C39 CLB_R17C39.S1 , module "hset" "hset"
   "QDDFSC/CORDICR/x_datapath_11(0)" ,
2   cfg
   "XORF:QDDFSC/CORDICR/PIPELINE.PIPELINE.10.DPTH0_DPTH0.DPATHX
   /GEN0.GEN0.0.lXORCY:
3
4  ...
5  FFX:QDDFSC/CORDICR/PIPELINE.PIPELINE.10.DPTH0_DPTH0.DPATHX/G
   EN0.GEN0.0.REGC_REGC.REG:#FF
6  FFY:QDDFSC/CORDICR/PIPELINE.PIPELINE.10.DPTH0_DPTH0.DPATHX/G
   EN0.GEN0.1.REGC_REGC.REG:#FF
7  ...
8  "
9  ;
```

Code 6.2: Example of a Slice definition in XDL

Nevertheless, a mature solution was obtained generating the VHDL model with a specific option that adds, as comments, the node names as known in other programs of the Xilinx design flow. This option is –aka: “Write Also-Know-As names as comments”. This is the only explanation found in the documentation [Xil05]. Code 6.3 shows the same component as Code 6.1 but with the –aka comment at line 1.

```
1  QDDFSC_CORDICR_PIPELINE_PIPELINE_10_DPTH0_DPTH0_DPATHX_GEN0_
   GEN0_1_REGC_REGC_REG : X_FF --
   AKA:QDDFSC/CORDICR/PIPELINE.PIPELINE.10.DPTH0_DPTH0.DPATHX/G
   EN0.GEN0.1.REGC_REGC.REG
2      generic map(
3        INIT => '0'
4      )
5      port map (
6        I =>
   QDDFSC_CORDICR_PIPELINE_PIPELINE_10_DPTH0_DPTH0_DPATHX_outxo
   rcy,
7        CE => ce_s,
8        CLK => c_s,
9        SET => GND,
10       RST => QDDFSC_CORDICR_x_datapath_11_0_FFY_RST,
11       O => QDDFSC_CORDICR_x_datapath_11(1)
12     );
```

Code 6.3: Example instantiation of a Slice flip-flop in VHDL with the –aka option

6.1.2 Obtaining the Identifiers

The VHDL model generated from the post PAR layout is based on the instantiation of components representing the physical resources within the Slices: LUTs, XOR gates, FFs, etc. All this VHDL code must be parsed in order to obtain the required identifiers.

Code 6.4 is a simplified version of the procedure that parses each component instantiation within the architecture of the VHDL model.

```
1  procedure componentInstantiation;
2  var
3    componentIdentifier: String;
4    portIdentifier: String;
5    netIdentifier: String;
6    AKAIdentifier: String;
7    Index: String;
8  begin
9    parea(PAL); // Label
10   parea (Ord(':'));
11   componentIdentifier := valcompl.s;
12   parea(PAL); // Component Identifier
13   AKAIdentifier := '';
14
15   // Correlate simulator and XPower Names
16   parea(AKA);
17   parea (Ord(':'));
18   AKAIdentifier := valcompl.s;
19   parea (PAL);
20   if preanalisis = Ord('(') then
21   begin
22     parea (Ord('('));
23     Index := valcompl.s;
24     parea(PAL); // Port Identifier
25     //Id with parenthesis
26     AKAIdentifier := AKAIdentifier + '(' + Index + ')';
27     parea (Ord(')'));
28   end;
29
30   // There could be a postfix after the parenthesis
31   if preanalisis = PAL then
32   begin
33     AKAIdentifier := AKAIdentifier + valcompl.s;
34     parea (PAL);
35   end;
36
37   // Generic Map
38   ...
39
40   // Port Map
41   if preanalisis = PRT then
```

```
42   begin
43     parea (PRT);
44     parea (MAP);
45     parea (Ord('('));
46
47     // Lista de inicializaciones
48     while preanalisis <> Ord(')') do
49     begin
50       portIdentifier := valcompl.s;
51
52       parea(PAL); // Port Identifier
53       parea(ASC);
54       netIdentifier := valcompl.s;
55       parea(PAL); // Net Identifier
56       if preanalisis = Ord('(') then
57       begin
58         parea (Ord('('));
59         Index := valcompl.s;
60         parea(PAL); // Port Identifier
61         //para poder hacer la comparación con vdc que pone
   paréntesis:
62         netIdentifier := netIdentifier + '[' + Index + ']';
63         parea (Ord(')'));
64       end;
65       if (portIdentifier = 'O') or (portIdentifier = 'Q')
   then
66         addVHDLName (AKAIdentifier, netIdentifier,
   componentIdentifier, true)
67       if preanalisis <> Ord(')')then
68         parea (Ord(','));
69     end;
70
71     parea (Ord(')'));
72     parea (Ord(';'));
73   end
74 end;
```

Code 6.4: Procedure that parses VHDL instantiations

The type of component instantiated is obtained at line 11 and is also stored in the Power Database. Afterwards, the power consumption could be grouped by FPGA resources (flip-flops, OR gates, LUTs, etc.). From line 16 to 35, the “aka” name is obtained. These names are written within comments, so the lexical analyzer was modified to avoid ignoring them. In this way, comments with the AKA token are considered special comments.

From line 41, the port map section of the current instantiation is analyzed, but just the output port is explored to relate with the “aka” name as explained above. At line 66,

the `addVHDLName` procedure is called to store the relevant collected information in the Power Database, but through the abstraction layer in order to write a code independent of a specific database engine.

6.1.3 Optimization

For some users it could be useful to study different scenarios changing the primary input activity rates. It could also be useful to estimate the power consumption with several accuracies in different stages of the design process. The VHDL model of a large design could be a several-MB file and analyzing it is a heavy time consuming task. As the information extracted from the VHDL model is independent from input pattern characteristics and the selected accuracy, it should not be analyzed more that once for a design. The program described in this section takes it into account and detects if it was previously executed with the current design, saving a significant execution time in subsequent power estimations.

6.2 Parsing the Xilinx Design XDL file

The XDL files are text files that can contain all the information necessary to generate the binary configuration of an FPGA device. In fact, XDL files are generated from circuit layouts in proprietary binary format (NCD).

The file described in this section obtains the position of the nets in a post PAR design. Also it finishes the uncompleted names dictionary started with the VHDL parser explained above.

Although the code is structured as a parser, the grammar is undocumented and was discovered studying test cases and some comments found in the XDL files.

The results of the analysis are stored in the Power Database. As all the programs in this work, the XDL parser is a console application. `xdlParser.exe` has more than 850 lines of source code and the most important fragments are explained in this section.

`xdlParser.exe` accept command line parameters:

```
xdlParser.exe -d dir
```

`-d dir` specifies the working directory

The XDL file contains two parts: The first one with slice definitions and the second one with net specifications. The treatment of these specific file parts are explained in the following sub-sections.

6.2.1 Analyzing an FPGA Slice Definition

The syntax rule for an FPGA slice definition (instances, in the XDL vocabulary) is:

```
instance <name> <sitedef>, placed <tile> <site>, cfg <string>;
```

or:

```
instance <name> <sitedef>, unplaced, cfg <string>;
```

As in this work just full PAR designs have been studied, the first is the only used form. Code 6.5 is a simplified version of the procedure that parses these Slice definitions:

```
1  procedure DInstance;
2  var
3    row, column, sliceNum: Integer;
4    p, q: Integer;
5    sliceName: string;
6    IOB: Boolean;
7    CLB: Boolean;
8  begin
9    CLB := false;
10   row      := -1;
11   column   := -1;
12   sliceNum := -1;
13
14   parea (INS);
15
16   parea(Ord('"'));
17   sliceName := valcompl.s;
18   parea(PAL);
19   parea(Ord('"'));
20
21   parea(Ord('"'));
22   parea(PAL);
23   parea(Ord('"'));
24
25   parea(Ord(','));
26
27   parea(PLC);
```

```
28   parea(PAL);
29
30   if Pos('CLB', valcompl.s) > 0 then // It's a CLB
31   begin
32     CLB := true;
33     p := pos ('R', valcompl.s);
34     valcompl.s[1] := ' ';
35     q := pos ('C', valcompl.s);
36     // Row in the FPGA where the CLB (Slice) is placed
37     row := StrToInt(Copy (valcompl.s, p+1, q-p-1));
38     p := pos ('.', valcompl.s);
39     // Column in the FPGA where the CLB(Slice) is placed
40     column := StrToInt(Copy (valcompl.s, q+1, p-q-1));
41     q := Length (valcompl.s);
42     // Specific Slice within the CLB
43     sliceNum := StrToInt(Copy (valcompl.s, p+2, q-p-1));
44
45     addSliceInSlicesT(SliceName, row, column, sliceNum);
46   end
47   else ...
48
49   parea(PAL); parea(Ord(','));
50
51   while preanalisis <> CFG do
52     parea(preanalisis);
53   parea(CFG);
54
55   parea(Ord('"'));
56
57   if CLB then DConfCLB (row, column, sliceNum)
58
59   else ...
60
61   parea(Ord('"'));
62   parea(Ord(';'));
63 end;
```

Code 6.5: Procedure that parses Slice definitions in an XDL file

Once a slice definition is found, from line 30 to 46, the slice position (<site>) is obtained. At line 45, the slice position is stored in the Power Database. It will be utilized later to build power maps, as explained below.

The slice configuration is complex enough to write a separated procedure as shown at line 47. Code 6.6 is a complete example of a Slice definition. Fig. 6.1 shows the same slice but as shown in the Xilinx FPGA Editor.

```
1  inst "round4/mult4/Mult/ADD/suma2_reg(14)" "SLICE" , placed
   R17C35 CLB_R17C35.S0 ,
2   cfg "XORF:round4/mult4/Mult/ADD/un4_suma2_s_14:
3        XORG:round4/mult4/Mult/ADD/un4_suma2_s_15:
4        CYMUXF:round4/mult4/Mult/ADD/un4_suma2_cry_14:
5        CYMUXG:round4/mult4/Mult/ADD/un4_suma2_cry_15:
   CYSELF::F
6        CYSELG::G CKINV::1 COUTUSED::0 YUSED::#OFF XUSED::#OFF
7        XBUSED::#OFF F5USED::#OFF YBMUX::#OFF CYINIT::CIN
   DYMUX::1
8        DXMUX::1 CY0F::F1 CY0G::G1
9
   F:round4/mult4/Mult/ADD/un4_suma2_axb_14:#LUT:D=(A1@A3)
10
   G:round4/mult4/Mult/ADD/un4_suma2_axb_15:#LUT:D=(A1@A4)
11       RAMCONFIG::#OFF REVUSED::#OFF BYMUX::#OFF BXMUX::#OFF
12       CEMUX::#OFF SRMUX::#OFF GYMUX::GXOR FXMUX::FXOR
13       SYNC_ATTR::ASYNC SRFFMUX::#OFF INITY::LOW
14       FFX:round4/mult4/Mult/ADD/suma2_reg[14]:#FF
15       FFY:round4/mult4/Mult/ADD/suma2_reg[15]:#FF INITX::LOW
16
   _PINMAP:24:0,1,2,3,4,5,6,8,7,9,10,11,12,15,14,13,16,17,18,19
   ,20,21,22,23"
17  ;
```

Code 6.6: Example of a Slice definition

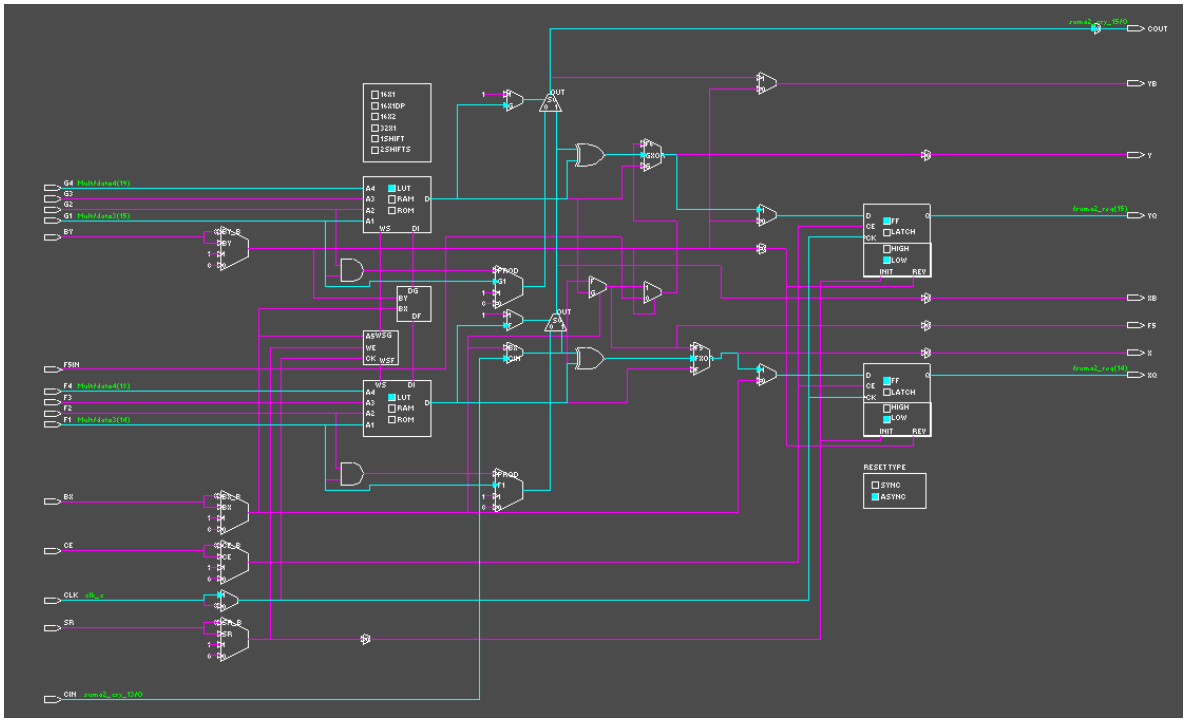


Fig. 6.1: Example of a Slice as shown in the Xilinx FPGA Editor

When the FPGA slice definition of Code 6.6 is analyzed, it should be detected all the internal nets: LUT outputs (Lines 9 and 10), XOR outputs (Lines 2 and 3), multiplexers (Lines 4 and 5), flip-flops (Lines 14 and 15), etc. Code 6.7 analyzes Slice configurations.

```
1  procedure DConfCLB (row, column, sliceNum: Integer);
2  var
3    p, q: Integer;
4    fieldXName: string;
5  begin
6    while preanalisis <> Ord('"') do
7    begin
8       if isSliceRes (valcompl.s) then
9       begin
10        // Resource type separator
11        p:= Pos(':',valcompl.s);
12        fieldXName:= Copy (valcompl.s, p + 1 ,
13                           Length(valcompl.s)- p);
14        // Id separator
15        q:= Pos(':', fieldXName);
16        fieldXName:= Copy (fieldXName, 1, q - 1); // Id
   extraction
17        if fieldXName <> ':' then
18          if AKAFound then
19            updatePos(fieldXName, row, column, sliceNum)
20          else updateXName(fieldXName, row, column,
   sliceNum);
21      end;
22
23      parea(preanalisis);
24    end;
25 end;
```

Code 6.7: Procedure that parses Slice configurations in an XDL file

Once a new name is obtained (line 16), its position is stored in the Power Database (from line 18 to 20). This is done until the end of the configuration is detected.

At the end of the analysis, the positions of the internal nodes are in the Power Database for future uses like drawing power maps. These maps are explained below in this chapter. Note that the Power Database is accessed through an abstraction layer as explained in Chapter 4.

6.2.2 Analyzing a Net Definition

The syntax for a Net definition is:

```
net <name> <type>,
  outpin <inst_name> <inst_pin>,
  ...
  inpin <inst_name> <inst_pin>,
  ...
  pip <tile> <wire0> <dir> <wire1> , # [<rt>]
  ...
;
```

The net location, from the power consumption point of view, is determined by the position of the output pin that drives the net. This output pin is situated in the <inst_name> slice. Slice names and their positions have been previously stored when the slice definitions were parsed. With Code 6.8, these nets are parsed:

```
1  procedure DNet;
2  var
3    netName, sliceName: string;
4  begin
5    parea (NET);
6
7    sliceName := '';
8
9    parea(Ord('"'));
10   netName := valcompl.s;
11   parea(PAL);
12   parea(Ord('"'));
13
14   if preanalisis = PAL then
15     parea (PAL);
16   parea(Ord(','));
17
18   // Skip the config
19   if preanalisis = CFG then
20   begin
21     parea(CFG);
22     parea(Ord('"'));
23     while preanalisis <> Ord('"') do
24       parea(preanalisis);
25     parea(Ord('"'));
26     parea(Ord(','));
27   end;
28
29   // Look for the net driver
30   while (preanalisis <> OPN) and (preanalisis <> Ord(';'))
   do
31     parea(preanalisis);
```

```
32
33   if preanalisis = OPN then
34   begin // This is the net driver
35     parea(OPN);
36     parea(Ord('"'));
37     // The Slice where the driver is positioned
38     sliceName := valcompl.s;
39     parea(PAL);
40     parea(Ord('"'));
41     parea(PAL);
42     parea(Ord(','));
43   end;
44
45   // Update the net position
46   addNet (netName, SliceName);
47
48   // Skip all other text within the net definition
49   while preanalisis <> Ord(';') do
50     parea(preanalisis);
51   parea(Ord(';'));
52 end;
```

Code 6.8: Procedure that parses net definitions in an XDL file

At line 10 the net name is obtained. At line 38 the Slice where the driver is positioned is retrieved from the definition. Finally, at line 46 the net position is updated in the Power Database through an abstraction layer. Now, all the net positions have been determined.

6.3 Generating the XML Settings File

The Xilinx's power computation tool, XPower, can store and retrieve the activity of the individual nodes (and other user settings) in XML format. However, XML files can be edited and also generated from scratch with a third party tool. In this way, XPower could be used for report generation.

The program described in this section generates an XML settings file for XPower in order to compare and debug the A-DyP results. Although the generation is relatively fast, it could be skipped when the estimation time budget is critical.

As all the programs in this work, the XML generator is a console application. `XMLRep.exe` has approximately 500 lines of source code and the most important fragments are explained in this section.

`XMLRep.exe` accept command line parameters:

```
XMLRep.exe -d dir
```

`-d dir` specifies the working directory

6.3.1 Using a Package to Generate XML

To make the XML generation easier, a public package was used: XDOM. The 'Extended Document Object Model' Package for Delphi and Kylix contains several functions, classes, and components which support the processing of XML documents. It allows representing an XML document by Delphi objects which reproduce the structure and content of the XML document in an object tree [XDOM].

An XML setting file is composed by a head and a body. The head contains information related to the whole circuit (ambient temperature, voltage, etc.), and the report body contains specific information of individual nodes. Code 6.9 shows an example for one circuit node.

```
1  <Power_Net name="clk">
2    <Power_Activity freq="20.000000Mhz" />
3  </Power_Net>
```

Code 6.9: Example of a net description in a XPower XML setting file

Code 6.10 is a simplified version of the procedure that shows how such information is retrieved from the Power Database in order to generate the XML report body.

```
1  procedure CreateXMLPowerBody (var powerNets: TdomNode);
2  var
3    lname: String;
4    media, desvio, netAct: Real;
5    muestra   : Integer;
6    CkFrec    : Real; // Clock frequency
7    net, act: TDOMElement;
8  begin
9    // Computes the clock frequency
10   ...
11
12   // SQL query to retrieve node information
13   ...
14
15   // XML body generation
16   while not eof do
17   begin
```

```
18     lname := FieldByName('X_Name').asString;
19     netAct := FieldByName('average').asFloat;
20     netAct := netAct * CkFrec/2;
21
22     net := Doc.createElement('Power_Net');
23     powerNets.appendChild(net);
24     net.setAttribute('name', lname);
25     act := Doc.createElement('Power_Activity');
26     net.appendChild(act);
27     act.setAttribute('freq', formatFloat('0.00000',
   netAct)+'MHz');
28     powerNets.appendChild(Doc.createTextNode(indent));
29
30     Next;
31   end;
32 end;
```

Code 6.10: Procedure that generates the net descriptions in a XPower XML setting file

The activity stored in the Power Database is used to generate the body of the XML file. Although all the internal nodes are included in the XML report body, some are ignored by XPower because they are output-only fields or other reasons. From line 22 to 28, the XDOM package is used to produce the description of an individual node activity.

6.4 Extracting the Capacitances

XPower can be used to produce a capacitance report file (PWA) for all the nodes in the DUT. With the generated XML settings file, XPower can be invoked by command line:

```
xpwr design[.ncd] [constraint[.pcf]] [options]
```

Where `design` is the name of the physical design file, `constraint` specifies the name of a physical constraints file. `options` is one or more of the XPower options (see the XPower documentation [Xil05]).

In this work XPower is executed with the following parameters:

```
-l limit

-x test_xpwr.xml

-o test.pwr
```

`-t script.tcl`

`-l limit` imposes a limit in the number of lines on the verbose report. An integer value must be specified as an argument.

`-x test_xpwr.xml` instructs XPower to use the generated XML settings file as explained in section 6.3.

`-o test.pwr` changes the name of the report file.

`-t script.tcl` enables the use of a Tcl commands file. Currently, this option is not present in the Xilinx documentation. In this work just one command is of the main interest, `annotateDesign`, which generates a detailed capacitance report, in fF (femtofarads), for all the internal nodes in the input physical design file. Unfortunately, this command is not found in the documentation at all.

6.4.1 Parsing the Xilinx Capacitance Report File (PWA)

The file generated with the `annotateDesign` command by XPower, is analyzed to extract the node capacitances and store them in the Power Database. As with all the programs in this work, the PWA parser is a console application. `PWAparser.exe` has more than 450 lines of source code and the most important fragments are explained in this section.

`PWAparser.exe` accepts command line parameters:

`PWAparser.exe -d dir`

`-d dir` specifies the working directory

Code 6.11 is a simplified version of the procedure that extracts and stores individual capacitances.

```
1  procedure PWAline;
2  var
3    c: integer;
4    signal: string;
5  begin
6    signal := valcompl.s; // node name
7    parea(PAL);
8
9    parea (Ord(','));
10
```

```
11   c := StrToInt(valcompl.s); // capacitance
12   parea(PAL);
13
14   parea (Ord(','));
15
16   parea(PAL); // activity
17
18   SaveCapacitance (signal, c);
19 end;
```

Code 6.11: Procedure that extracts node capacitances from a PWA file

At line 6, the node name is obtained, and at line 11, its capacitance in fF. Node activity is also reported but this is ignored because it has been already stored in the Power Database.

6.5 Calculating the Power Consumption and Writing a Report

With the activities and capacitances of each node in the DUT, now, the individual power consumptions can be computed. Also, using the information collected by parsing the XDL and PWA files, the power consumption can be grouped by general circuit type: logic, signals, clocks and I/Os. Furthermore, there is enough information to differentiate the power of the different logic resources within the slices. It means that can be produced a report that also shows the power consumed by the LUTs, XORs, MUXs, etc. within the slices in the physical design.

`report.exe` is the command line program that produces the power consumption reports. An example report is shown in Appendix D.

`report.exe` has approximately 550 lines of source code and accepts the following parameters:

```
report.exe -d dir -v
```

`-d dir` specifies the working directory

`-v` if present, indicates that the report will contain the power estimation for all the individual nodes, in other cases this information is excluded.

6.6 Generating the Power Maps

With the information collected during the estimation process, several maps can be drawn. On one hand, node positions are obtained parsing the XDL file. On the other hand activity, capacitance and individual power consumption are also available in the Power Database. Relating positions and node property values, the maps can be drawn with a user specified resolution.

`activityMap.exe`, `capacitanceMap.exe` and `powerMap.exe` are command line programs that produce the information for the activity, capacitance and average power consumption maps respectively. Several example maps are shown for the test cases in Chapter 8.

Each one of these programs has approximately 130 lines of source code and accepts the following parameters:

`{activityMap.exe, capacitanceMap.exe, powerMap.exe} -d dir –r resolution`

`-d dir` specifies the working directory

`-r resolution` defines the grain for the map drawing. A resolution `res` means that the property values of the nodes in `res` x `res` CLBs are added together.

The information is written to a text file where each line is in the form:

`column row property_value`

Thus, this information can be used to plot 2D and 3D figures using scientific graphing software.

6.7 Conclusions

In this Chapter, the Power Computation Sub-system is explained in detail, completing the Chapter 4 general description. Although this system has been developed taking into account proprietary formats, it can be recognized that some software pieces could be used in other power estimation tools for the same vendor. Being independent console applications, they can be coordinated within another Tcl script. In this way these programs belong to the Power Platform.

The Multiple Identifiers' Problem comes from the lack of integration with the vendor software in the power estimation area. This problem is not completely solved in this work because new modifications are introduced with each new version, making it very difficult to find a definitive solution. Nevertheless, it is reasonable to believe that in the near future, FPGA vendors will specify formats for data interchange and integration with third-party developers in the power estimation area.

References

[ITRS05] International Technology Roadmap for Semiconductors, 2005 Edition, available at http://public.itrs.net

[Xil05] Xilinx Inc., "Development System Reference Guide", 2005.

[XDOM] Dieter Köhler, "Extended Document Object Model", available at http://www.philo.de/xml/dom/

Chapter 7.

"How much energy must be used in carrying out a computation? This doesn't sound all that academic. After all, a feature of most modern machines is that their energy consumption when they run very fast is quite considerable." Richard P. Feynman in Lectures on Computation[Fey96].

7 Test Cases and Analysis

In this Chapter, a number of test circuits are briefly explained. These circuits are implemented in several FPGA devices. The possible tests to study the power estimation problem over these circuits, and to evaluate the power estimation tool are also enumerated. Using these circuits and running the test described below, a set of experiments and its results are presented in Chapter 8.

As it is necessary to physically measure the average core power for the test circuits, the experimental setups are also briefly described in this Chapter.

7.1. Test Circuits

Table 7.1 and 7.2 show the circuits studied in this work and their resource utilization in the specific devices where they were implemented. In the subsections each circuit is explained in some detail.

7.1.1 Quadrature Direct Digital Frequency Synthesizers (QDDFS)

Circuit	Code	Description	Device
1	QDDFS-CORDIC(RTL)	Quadrature Direct Digital Frequency Synthesizer based on CORDIC. Portable RTL HDL.	XCV300E-8-PQ240
2	QDDFS-CORDIC(RTL-A)	Quadrature Direct Digital Frequency Synthesizer based on CORDIC. Portable RTL HDL with area restriction.	XCV300E-8-PQ240
3	FIRDA(1)	Distributed Arithmetic FIR Filter. Completely serial (Digit 1)	XCV400E-8-PQ240
4	FIRDA(2)	Distributed Arithmetic FIR Filter. 2-bit serial (Digit 2)	XCV400E-8-PQ240
5	FIRDA(3)	Distributed Arithmetic FIR Filter. 3-bit serial (Digit 3)	XCV400E-8-PQ240
6	FIRDA(4)	Distributed Arithmetic FIR Filter. 4-bit serial (Digit 4)	XCV400E-8-PQ240
7	FIRDA(8)	Distributed Arithmetic FIR Filter. Completely combinational (Digit 8)	XCV400E-8-PQ240
8	FFT_A	Fast Fourier Transform, version A	XCV800-HQ240-4
9	FFT_B	Fast Fourier Transform, version B	XCV800-HQ240-4
10	FFT_C	Fast Fourier Transform, version C	XCV800-HQ240-4
11	FFT_D	Fast Fourier Transform, version D	XCV800-HQ240-4
12	MULT32-C	Unsigned Combinational 32-bit Multiplier	XCV50PQ240-4
13	ADDER32-C	Unsigned Combinational 32-bit Adder	XCV50PQ240-4
14	MULT16-P	Unsigned Pipelined 16-bit Multiplier	XCV50PQ240-4
15	DIV16-P	Unsigned Pipelined 16-bit Divider	XCV50PQ240-4
16	10MULT16-C	Ten Unsigned Combinational 16-bit Multiplier	XC2V3000FG676-6

Table 7.1: Studied Circuits

Two implementations of (COordinate Rotation Digital Computer) CORDIC-based QDDFS circuits are described in this sub section. Measurements for these implementations have been done over a Virtex-E FPGA (XCV300E-8PQ240 device) [XDS02]. The QDDFS circuits have been driven by a 100 MHz clock. Nevertheless, at the outputs, digital sine and cosine waveforms are generated with the specified period.

Power has been measured for several output frequencies (f_{out}): 1, 10, 20 and 30 MHz. These f_{out} are computed according to:

$$f_{out} = \frac{fcw \cdot f_{clk}}{2^M} \qquad \text{(Eq. 7.1)}$$

Circuit	# Slices	Slice FF	#Nodes	Min. Period (ns)
1	484 (15%)	773 (12%)	5411	8.591
2	484 (15%)	773 (12%)	5407	9.220
3	159 (3%)	307 (3%)	1486	5.781
4	303 (6%)	597 (6%)	2774	7.305
5	456 (9%)	897 (9%)	4092	6.276
6	595 (12%)	1177 (12%)	5245	6.484
7	1163 (24%)	2305 (24%)	9495	5.903
8	3424 (36%)	6364 (33%)	32622	12.803
9	3384 (35%)	6364 (33%)	32760	11.767
10	3424 (36%)	6364 (33%)	32242	11.731
11	3424 (36%)	6364 (33%)	32708	10.457
12	640 (83%)	193 (12%)	6627	40.377
13	49 (6%)	97 (6%)	656	11.354
14	172 (22%)	341 (22%)	2194	9.638
15	425 (55%)	831 (54%)	3257	9.899
16	1654 (11%)	586 (2%)	27471	14.928

Table 7.2: Resource utilization of the Test Circuits

Where *fcw* (frequency control word) is a circuit primary input that determines the output frequency; f_{clk} is the clock frequency; and M is the internal accumulator width (18 for these test cases). Table 7.3 shows the *fcw* values used in the experiments.

Fout [MHz]	Fcw (decimal)	fcw (hexadecimal)
1	2621	A3D
10	26214	6666
20	52429	CCCD
30	78643	13333

Table 7.3: QDDFS output frequencies

CORDIC based QDDFS is implemented with a portable HDL. The second circuit is implemented with this portable implementation but with an area restriction. For a detailed description of these circuits and their implementation in FPGA, see [Car03].

From the point of view of the power estimation these circuits are useful to test the results accuracy. However the Monte-Carlo technique cannot be tested due to the fixed inputs and the periodic outputs. Simulating one whole QDDFS cycle is enough for average power estimation.

7.1.2 Distributed-Arithmetic FIR Filter (FIRDA)

These are different implementations of a FIR filter. For all these circuits distributed arithmetic [May01] and the relative placement technique [XST03] are applied. The filters use 64 6-bit coefficients, 8-bit input and output words, 12.5 MHz fixed sampling frequencies, and a 2/3 cut-off frequency. Measurements for these implementations have been done over a Virtex-E FPGA (XCV400E-8PQ240 device) [XDS02].

Implementation	Internal Digit	Clk Frequency [MHz]
FIRDA(1)	1	100
FIRDA(2)	2	50
FIRDA(3)	3	33.3
FIRDA(4)	4	25
FIRDA(8)	8	12.5

Table 7.4: Clock frequencies for FIRDA implementations

The difference among these implementations is the internal digit size from bit serial to completely combinational. As the sampling frequency is fixed, the clock must be adjusted to compute each sample before the next is available. Table 7.4 summarizes this specific information.

For a detailed description of these circuits and their implementation in FPGA, see [Ang03].

7.1.3 Fast Fourier Transform (FFT)

FFT_A, B, C and D are 64-point pipelined FFT implementations that fulfill the Hiperlan/2 and IEEE 802.11a-g standards. This implementation uses the Radix-4 algorithm. In order to fulfill the mentioned standards the design is pipelined and R4MDC (Radix-4 Multi-path Delay Commutator) is the selected architecture due to its area, speed and power figure. Measurements for these implementations have been done over a Virtex FPGA (XCV800HQ240-4 device) [XDS01]. The core is carefully designed and optimized for Virtex and Virtex-E devices by means of the relative placement technique [XST03]. There are small differences among versions A-D. Each one try an optimization in the implementation of one component in the FFT core.

In [San03] a detailed description of these circuits and their implementation in FPGA can be found.

7.1.4 Arithmetic Circuits

Four fundamental arithmetic circuits have been implemented to test A-DyP: A combinational 32-bit multiplier, a combinational 32-bit adder, a pipelined 16-bit multiplier and a pipelined 16-bit divider. All these circuits operate with unsigned integers. The 32-bit adder and multiplier were specified using a simple behavioral VHDL description. For the pipelined multiplier and divider, the corresponding cores were generated with Core Generator [XCG03]. Measurements for these implementations have been done over a Virtex XCV50PQ240-4 device [XDS01].

Another arithmetic circuit has been implemented over a Virtex-II XC2V3000FG676-6 device [XDS15]. It is made up of ten 16-bit combinational multipliers with registered input and outputs.

7.2 Analysis of the Results

The test circuits described in the previous section are utilized in the experiments. They are measured, and simulated with several input pattern sets. These sets are characterized in such a way that several spatial and temporal correlations are defined. Furthermore, there are several parameters that users must specify for each A-DyP run. In this way, a tool characterization for these parameters is necessary.

The main goal in this work, from a practical point of view, is to obtain an accurate estimation for total and individual node power. Thus, in this section several tests are described in order to characterize the statistical technique and evaluate the power estimation tool as a software piece.

7.2.1 Technique Characterization

7.2.1.1 Testing the Total Power Estimation

The goal here is testing the tool accuracy. To reach it, it is necessary to measure physically the power consumption in order to compare them with the total average dynamic power estimations. This enables the tool calibration.

The input vectors are the same for both the estimations and the physical measurements. As explained in Chapter 5, generate.exe produces vectors in an adequate format for the simulator, and also translates these stimuli for the pattern generator. A-DyP results can also be compared with XPower [Xde03] Results. The activity generated for the same input vectors is written as an XML settings file as described in Chapter 6.

7.2.1.2 Testing the Power Estimation for Individual Nodes

It is not possible a physical power measurement for the individual nodes within the chip. Nevertheless, long run simulations are executed in order to have values to compare with the average power estimations for the nodes. Of course, the total power in these experiments is as accurate as possible compared with physical measurements.

The correctness of the implementation is tested specifying different values for the tolerated error for the individual nodes. Several simulations are run with a threshold for the minimal mean of 0.35, and tolerated errors varying from 30% to 3%.

7.2.1.3 Importance of the User Defined Accuracy

In this test, the accuracy improvement is analyzed against the execution time necessary to complete the simulations. This is also known as the accuracy vs. run-time tradeoff. The run time is represented by the number of samples necessary to obtain the required accuracy. This approach gives a value independent of the platform: computer, operating system, etc.

Accuracy is given by the confidence level and error pair but the minimal activity mean, that divides the normal from the low activity nodes, must also be studied.

7.2.1.4 Importance of the User Defined Minimal Mean Threshold

This is another test to study the accuracy vs. execution time tradeoff. In this case, the tolerated error and confidence are fixed and the minimal mean vary from 0.05 to 1 transitions per clock cycle. The percentage of regular and low activity nodes is studied and interesting conclusions can be obtained.

7.2.1.5 Importance of the Input Pattern Definitions

To test the importance of the input pattern descriptions, several specifications are defined for the primary inputs and for each circuit, varying from statistically independent inputs to other figures including connecting some inputs to a counter or even fixed values. This introduces test cases where primary inputs have spatial or temporal dependencies.

7.2.2 Software Evaluation

Beyond the correctness and accuracy, the execution time is essential in a practical artifact like this power estimation EDA tool.

Although the total execution time is studied, the run time of every sub program and script in the system is also analyzed. This is very important in order to focus the

optimization effort over the specific part of the system with the highest computational complexity and execution time.

7.2.3 Graphical Representation of the Results

Activity, capacitance, and power consumption maps are drawn for the circuits running in typical operating conditions. This enables the hot spots in the die to be discovered and which parts of the design should be optimized from the power consumption point of view. These figures also induce the study of correspondence between power consumption and temperature [Lop03].

7.3 Power Measurement

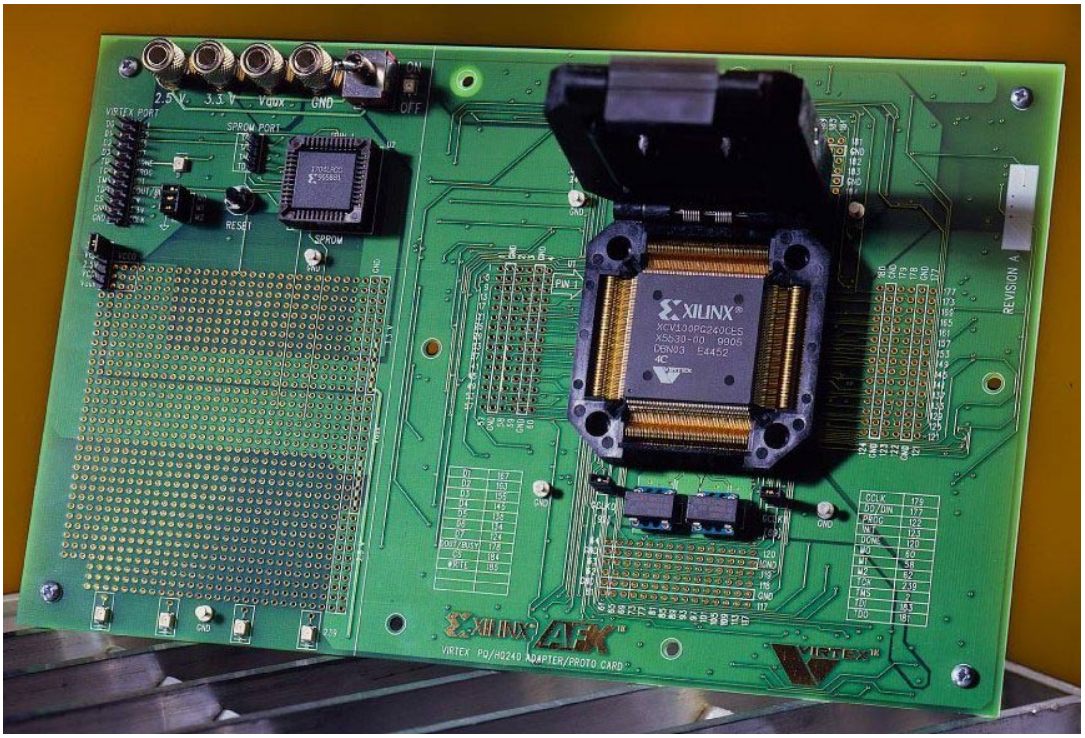


Fig. 7.1: Xilinx HW-AFX-PQ240-100 development board

As it was presented above, it is necessary to measure the test circuit's average power under the user specified conditions in order to calibrate, debug, and even develop an accurate power estimation tool. Several development boards were used for that purpose. The first one, employed in the preliminary experiments was developed by

Lopez-Buedo [Lop97] and those results were [Tod00], [Tod01] and [Sut01] but they are not presented in this work.

An important feature of a development board for average power measurement is to have special connectors for the logic analyzer and pattern generator. Other special characteristic of these boards is that power supply jacks are separated for the FPGA core, FPGA I/O, and general circuitry.

In the experiments with Virtex and Virtex-E devices a Xilinx HW-AFX-PQ240-100 development board as the one shown in Fig. 7.1 [APX99] was used. The socket in this board accepts 2.5V Virtex and Virtex-E devices in PQ240 and HQ240 packages. Another development board with similar characteristics for power measurement was used for Virtex-II devices. This is the Xilinx Virtex II FG676 development board as the one shown in Fig. 7.2 [APX03]. The socket in this board accepts 1.5V Virtex-II devices in FG676 packages.

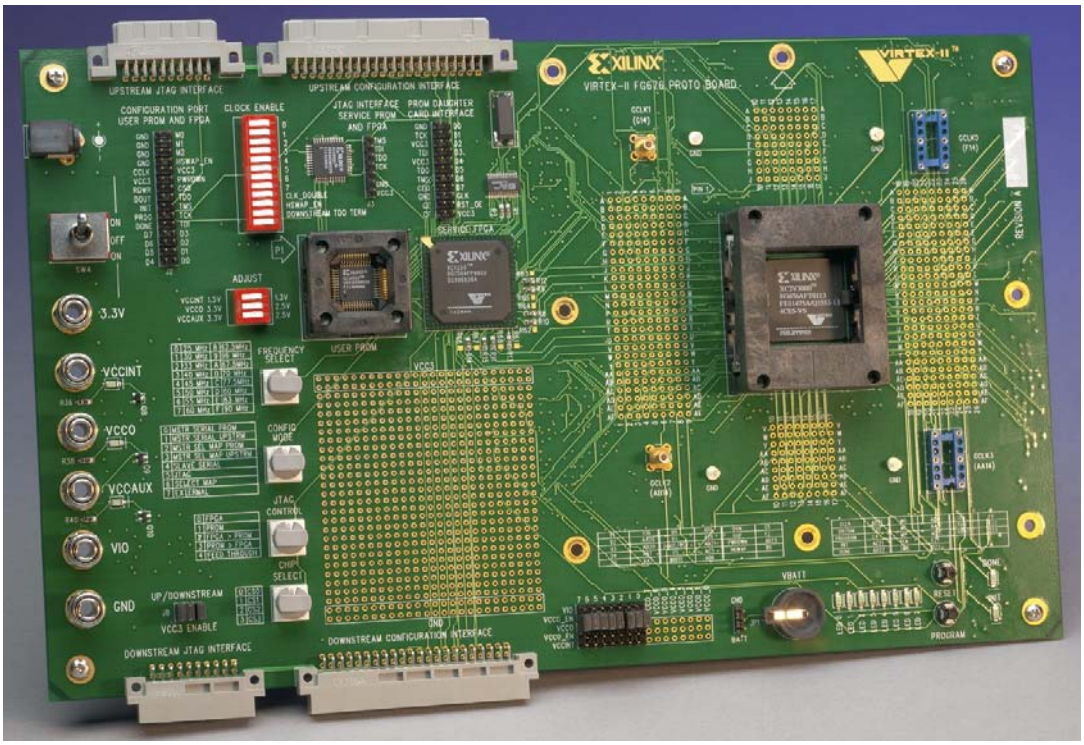


Fig. 7.2: Xilinx Virtex II FG676 development board

The most important characteristics of the devices used in this work are listed in Table 7.5.

A Tektronix Pattern Generator TLA7PG2 [Tek02] produces the stimuli as described in Chapter 5. Circuit outputs are also verified by means of a Tektronix Logic Analyzer TLA07A [Tek01]. An ammeter is used to measure average core currents maintaining the core voltage, *Vccint,* at the nominal values as shown in Fig. 7.3.

Family	Device	Size (CLB)	Slices	Distributed RAM	BlockRAM
Virtex	XCV50PQ240-4	16x24	768	24,576 bits	32,768 bits
Virtex	XCV800HQ240-4	56x84	9408	301,056 bits	114,688 bits
Virtex-E	XCV300EPQ240-8	32x48	3072	98,304 bits	131,072 bits
Virtex-E	XCV400EPQ240-8	40x60	4800	153,600 bits	163,840 bits
Virtex-II	XC2V3000FG676-6	64x56	14336	448 Kb	1728 Kb

Table 7.5: FPGAs members used in the experiments

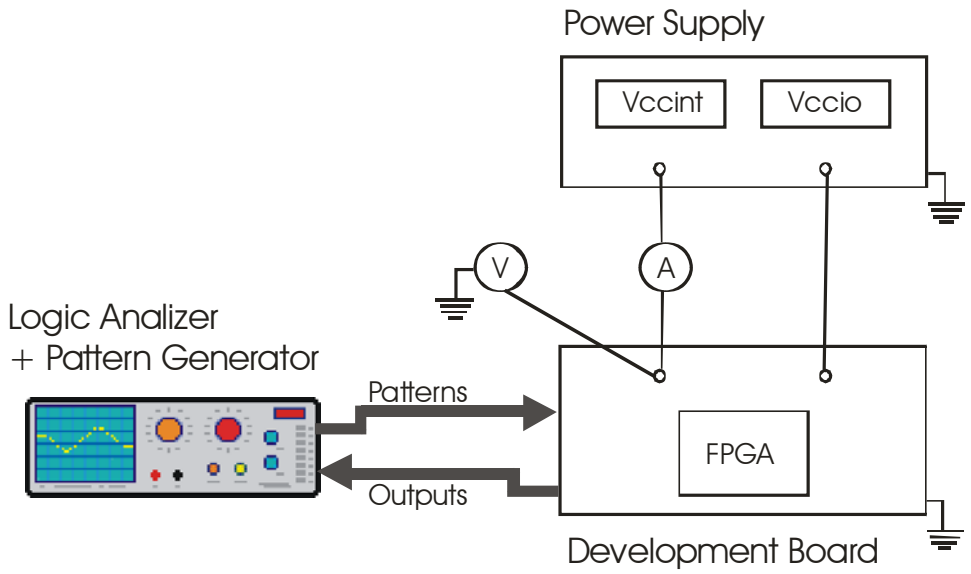


Fig. 7.3: Experimental setup

It is important to note that just the core current and voltage are measured. In this way I/O –or off-chip- power is excluded from the study. Core power can be divided into three components: dynamic, static, and synchronization power. The static, and static plus synchronization power measurements enables the division of these three components by subtraction. To measure the static power, the FPGA is configured with the DUT but neither stimuli nor clock are applied. Static plus synchronization power is

measured similarly but this time the clock signal is applied at the specified frequency. In [Sut05] a detailed description of all the experimental setups described in this section can be found.

References

[Ang03] Angarita F.E., Canet M.J., Valls J., □ardéi F: Implementación de un Core-IP "Filtro FIR Basado en Aritmética Distribuida", III Jornadas de Computación Reconfigurable y sus Aplicaciones – JCRA 2003. Madrid, 2003.

[APX03] Xilinx Inc. "Virtex II Prototype Platforms User Guide", UG015 (v1.1) Jan 2003, available at http://www.xilinx.com

[APX99] Xilinx Inc. "Xilinx Prototype Platforms User Guide for Virtex and Virtex-E Series FPGAs", Data Sheet DS020 (v1.1) Dec 1999, available at http://www.xilinx.com

[Car03] F. Cardéis, J. Valls, "Area-Optimized Implementation of Quadrature Direct Digital Frequency Synthesizers on LUT-based FPGAs", IEEE Trans. On Circuits and Systems II, vol. 50, no. 3, march 2003.

[Fey96] Richard P. Feynman, "Feynman Lectures on Computation", Ed. A.J.G. Hey and R.W. Allen. Addison-Wesley, 1996.

[Lop03] Sergio López Buedo, "Técnicas de Verificación Térmica para Arquitecturas Dinámicamente Reconfigurables", Ph. D. Thesis, Departamento de Ingeniería Informática, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Julio 2003.

[Lop97] S. López-Buedo, "Técnicas de Diseño de Alta Velocidad y Bajo Consumo" Memoria del Proyecto de Fin de Carrera, ETSI Telecomunicación, Universidad Politécnica de Madrid, Septiembre 1997.

[May01] Uwe Meyer-Baese, "Digital signal processing with field programmable gate arrays", Springer, 2001.

[San03] T. Sansaloni, A. Pérez-Pascual, J. Valls, "Area-efficient FPGA-based FFT processor", Electronic Letters, Vol.39, N.41, September 2003

[Sut01] G. Sutter, E. Todorovich, S. López-Buedo y E. Boemo "Propiedad Conmutativa y Diseño de Bajo Consumo: Algunos Ejemplos en FPGAs", VII WorkShop de IBERCHIP, Montevideo, Uruguay, 21-23 de Marzo, 2001.

[Sut05] Gustavo Sutter, "Aportes a la Reducción de Consumo en FPGAs", Ph. D. Thesis, Departamento de Ingeniería Informática, Escuela Politécnica Superior, Universidad Autónoma de Madrid, April 2005.

[Tek01] Tektronix Inc., "TLA 704 Logic Analyzer User Guide" available at http://www.tektronix.com

[Tek02] Tektronix Inc. "TLA7PG2 Pattern Generator Module" available at http://www.tektronix.com

[Tod00] E. Todorovich, G. Sutter, N. Acosta and E. Boemo, "End-user low-power alternatives at topological and physical levels. Some examples on FPGAs", XV Conference on Design of Circuits and Integrated Systems (DCIS2000), Le Corum, Montpellier, France, November 21-24, 2000.

[Tod01] E. Todorovich, G. Sutter, N. Acosta, S. López-Buedo y E. Boemo "Relación

entre Velocidad y Consumo en FPGAs", VII Workshop IBERCHIP, Montevideo, Uruguay, 21-23 de Marzo, 2001.

[XCG03] Xilinx Inc. "CORE Generator Guide", an Xilinx ISE 6 Software Manual, 2003, available at http://www.xilinx.com

[Xde03] Xilinx Inc. "Development System Reference Guide", an Xilinx ISE 6 Software Manual, 2003, available at http://www.xilinx.com

[XDS01] Xilinx Inc. "Virtex Data Sheets: Virtex 2.5 V FPGAs", 2001, available at http://www.xilinx.com

[XDS02] Xilinx Inc. "Virtex-E Data Sheets: Virtex-E 1.8 V FPGAs", 2002, available at http://www.xilinx.com

[XDS05] Xilinx Inc. "Virtex-II Platform FPGAs: Complete Data Sheet", 2005, available at http://www.xilinx.com

[XST03] Xilinx Inc. "XST User Guide", an Xilinx ISE 6 Software Manual, 2003, available at http://www.xilinx.com

Chapter 8.

"Every experiment proves something. If it doesn't prove what you wanted it to prove, it proves something else" Anonimous

8 Experimental Results

In the previous Chapter, a number of test circuits were briefly explained. These circuits are implemented in several FPGA devices. Also in Chapter 7, the possible tests to be executed in order to gauge these circuits and evaluate the estimation tool were enumerated. These elements (devices, designs, and tests), generate an important number of combinations and the results could be presented in several ways. In this chapter the experimental results are presented as they were obtained. Therefore, preceding results induce more ideas and experiments, and at the end a complete evaluation of the development gives a comprehensive picture to the reader.

First, QDDFS-CORDIC circuits are exercised. In these designs, there is a primary input that determines the period of the digitally synthesized sinusoidal waveform. With these circuits, the first evaluations of the current development are run in order to test accuracy, run time, software correctness, etc. The statistical approach is not evaluated with these designs because the inputs are precisely fixed.

Next, FIRDA filters evaluate the main characteristics of the implemented statistics-based power estimation technique. With these results, besides the accuracy, efficiency will be studied and a significant optimization is proposed. This saving in execution time is done without any loss of accuracy.

In the third place, the impact of the input pattern definitions is verified. Several synthetic pattern definitions are applied to arithmetical circuits. On the other hand, real world patterns are used over FFT implementations.

Finally, some of the experiments mentioned in the above paragraphs are repeated over circuits implemented on Virtex-II devices.

As the power is linear with the operating frequency, it is better to express the measurements by means of a normalized, frequency-independent unit. For this reason, in this work the measurements for synchronous circuits will be expressed in power per unit of frequency. With the current technology, and for a wide range of applications, mW/MHz is the preferred unit of energy. It could also be useful to consider the mW/MHz per node: two circuits with similar measured mW/MHz may have a different number of nodes, suggesting that the smaller one may have hot spots. Note that mW/MHz is a unit of energy equivalent to nJ. The energy consumption is the main function to optimize in low power design.

For all the test cases just the dynamic power is considered and the static, I/O and clocking power have been subtracted. In all these experiments post PAR VHDL models with routed delays are used in the simulations.

8.1 A-DyP Preliminary Evaluation

Over QDDFS-CORDIC circuits, a special set of test cases is executed. These circuits are implemented using a Virtex-E XCV300E-8PQ240 device. The statistical method is not tested but other programs within A-DyP are because all the primary inputs are fixed during the simulation. At the outputs, digital sine and cosine waveforms are generated within a specified period. With these tests, the computed power value can be compared with physical measurements. In this way, the estimations accuracy and the software tool correctness are studied. Almost all the A-DyP software participates in this computation. Also, the viability to draw power maps from the collected data is verified. Another point to note with these QDDFS-CORDIC circuits is that the clock power is considered within the estimations.

8.1.1 Total Dynamic Power Estimation

Table 8.1 shows the results for two QDDFS-CORDIC implementations. For each implementation four output frequencies (F_{out}) are synthesized. The measured total energy is compared with two estimations. In the first one, all the activity reported by the simulator is considered (T_g=*0*). In the second one, short glitches are filtered in order to avoid the overestimation observed in the first case. T_g represents the glitch pulse width where the time is so short that transitions are not rail to rail but smaller. In several papers like [And04], partial glitches are filtered. They define a partial glitch on a net as a pulse with duration shorter than the driver's delay.

It is important to note that there is another reason which leads to these inaccuracies. As pointed out in [Bae02]: "Logic simulators are neither precise nor reliable at measuring switching activity. It is due to the fact they are not accurate at simulating glitch propagation". In any case, without these short pulses, less power than that calculated with Eq. 2.7 is dissipated. Note that, filtering these short pulses, the error is less than 10% for all the cases. However, when no glitches are filtered, the error can reach more than 80%, always in excess.

	F_{out}	Measured	T_g = 0		Opt. T_g	
			Estimated	Error (%)	Estimated	Error (%)
QDDFS-CORDIC(RTL)	1	4,608	6,651	55,2	4,388	-4,8
	10	6,174	9,717	65,5	6,031	-2,3
	20	7,002	10,772	61,0	6,660	-4,9
	30	7,362	11,059	57,0	6,750	-8,3
QDDFS-CORDIC(RTL-Area)	1	4,356	7,020	71,9	4,424	1,6
	10	5,868	10,313	86,7	6,427	9,5
	20	6,534	11,375	81.3	6,919	5,9
	30	6,876	12,140	76,6	6,982	1,5

Table 8.1: Total Power Estimation for QDDFS-CORDIC Designs

The power values in Table 8.1 are shown in mW/MHz as in all the test cases in this work. Nevertheless, it is important to note that the clock frequency was 100 MHz for all

the measurements and estimations in these QDDFS-CORDIC cases. The optimum value for the min. glitch duration, *Tg*, calculated for this device, is 650 ps.

8.1.2 Impact of the Input Patterns Definition

This set of test cases is particular from the point of view of the input patterns characterization. The input value for the FCW input port (frequency control word) must be calculated to obtain the specified output frequency, but it is fixed for the complete simulation and measurement process.

As the clock frequency is always 100 MHz, about the same power consumption could be expected for all the output frequencies. The number of operations per unit of time is the same (the number of points in Fig. 8.1). However note that, for the selected cases, the higher the output frequency the higher the power consumption. This is because, for the highest frequencies, there are fewer points per output cycle, and the discrete steps must be larger, generating more activity in the output MSBs. A difference about the double in the power consumption is observed in Table 8.1, and these cases are not the extreme ones considering the output frequency range of this QDDFS-CORDIC.

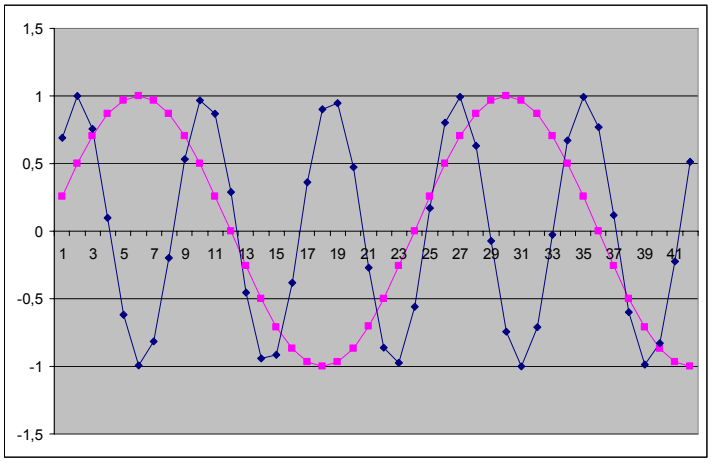


Fig. 8.1: Different output frequencies with the same clock period in QDDFS-CORDIC circuits

In despite of this circuit does not test the statistical technique, it is clear that the input characteristics will impact over the power figure.

8.1.3 Tool's Evaluation

In these special test cases, the transition analysis process is the most time consuming task as shown in Table 8.2 and Fig. 8.2. Nevertheless, it will be shown that this remains true for all the test cases. In this way, it is clear that an effort must be made to improve its efficiency.

	Task	**Exec. Time [sec]**
Activity Estimation	Input vector generation	0
	Simulation	18
	Saving	0
	Transition analysis	101
	Statistics computation	0
	Stopping criteria evaluation	0
Power Computation	VHDL parsing	0
	XDL parsing	0
	XML generation	65
	XPower execution	4
	PWA parsing	0
	Report writing	1

Table 8.2: A-DyP execution Time. QDDFS-CORDIC, 30 MHz, RTL

Task	**Exec. Time [sec]**
Total Execution	203
Initialization	4
Activity Estimation	119
Power Computation	77
Tcl/Tk script	3

Table 8.3: A-DyP total execution Time. QDDFS-CORDIC, 30 MHz, RTL

In order to have an idea about the absolute A-DyP execution times, Table 8.3 summarizes them by sub-system. The total execution time in the example is around 3 minutes.

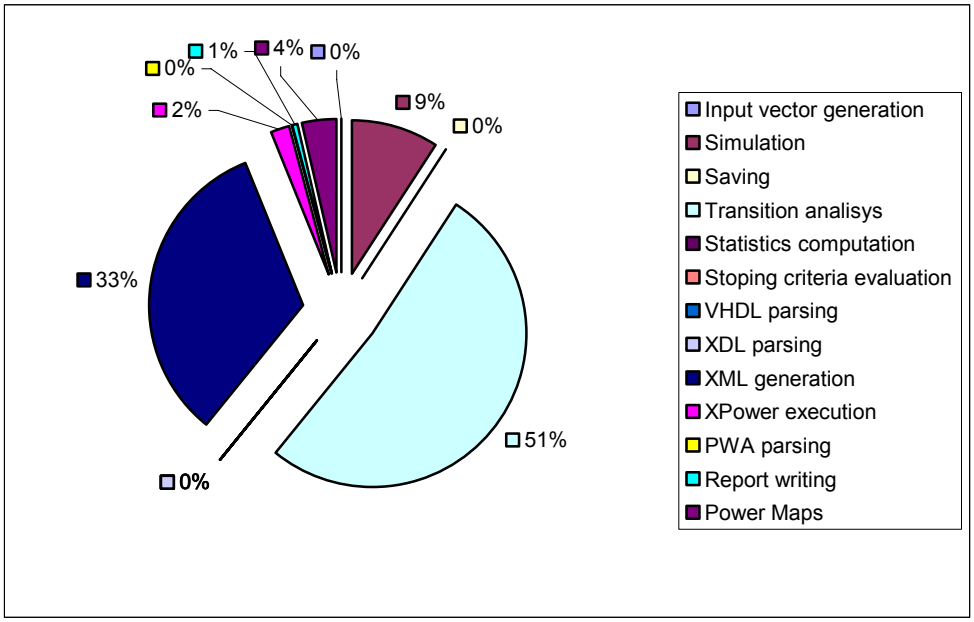


Fig. 8.2: A-DyP Execution Time for QDDFS-CORDIC

8.1.4 Power Maps

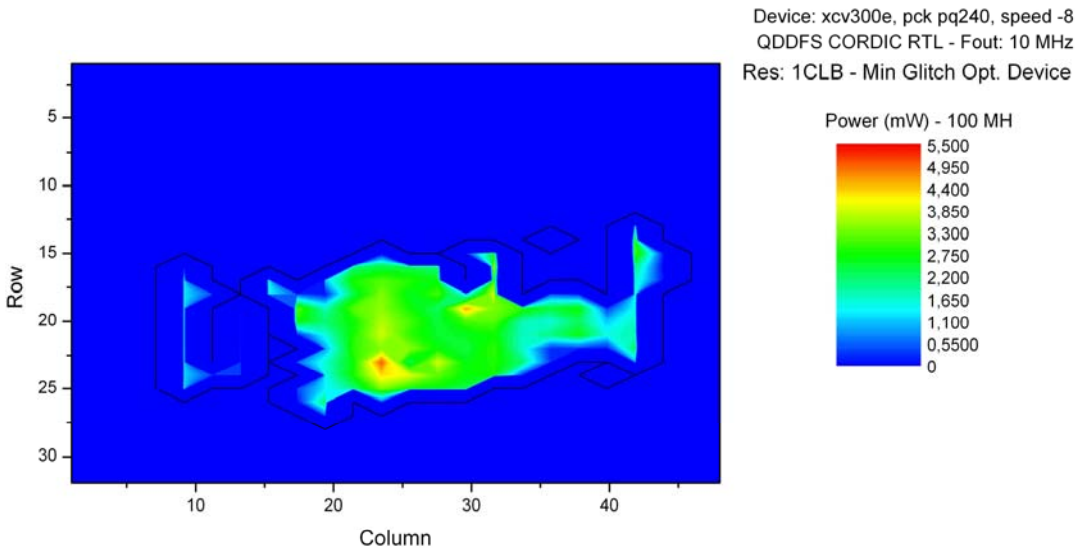


Fig. 8.3: Power Map. Resolution 1 CLB for QDDFS-CORDIC

As a first illustration, just one case is shown for the QDDFS-CORDIC RTL and Area-Restricted designs. It can be noted that neither the highest capacitance zones (Fig. 8.5 and 8.8) nor the highest activity points (Fig. 8.4 and 8.7) in the circuit have necessarily the highest power consumption (Fig. 8.3 and 8.6).

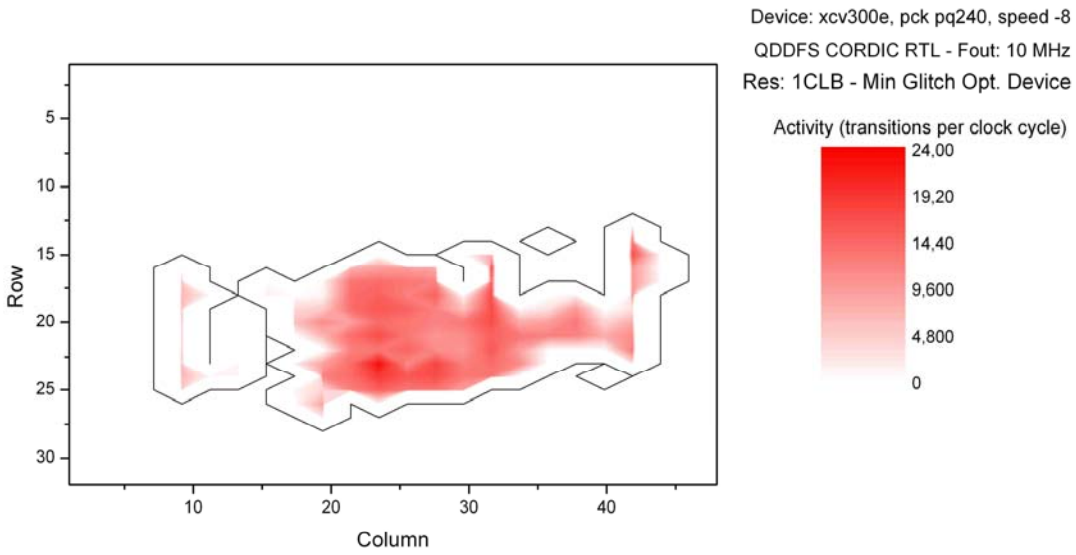


Fig. 8.4: Activity Map. Resolution 1 CLB for QDDFS-CORDIC

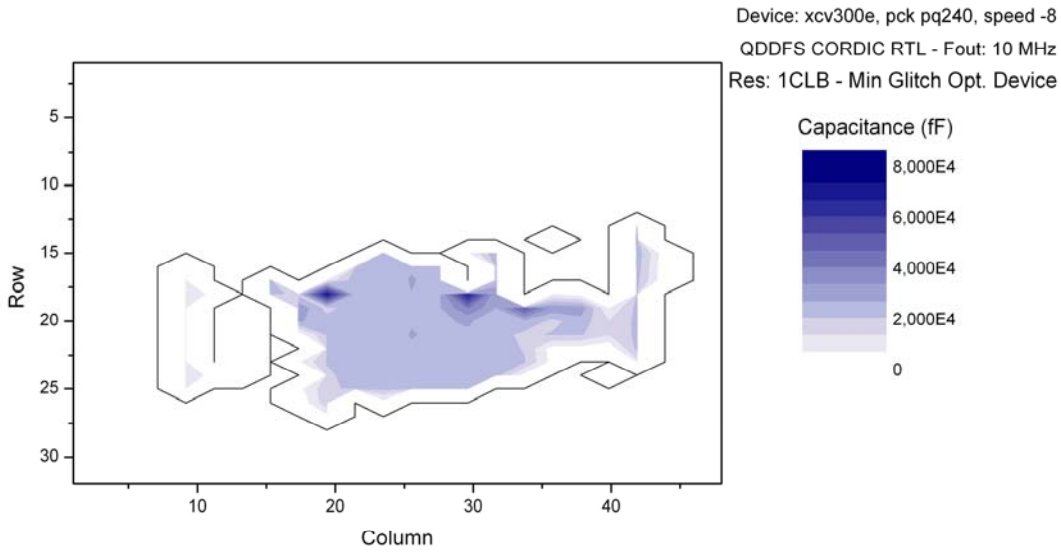


Fig. 8.5: Capacitance Map. Resolution 1 CLB for QDDFS-CORDIC

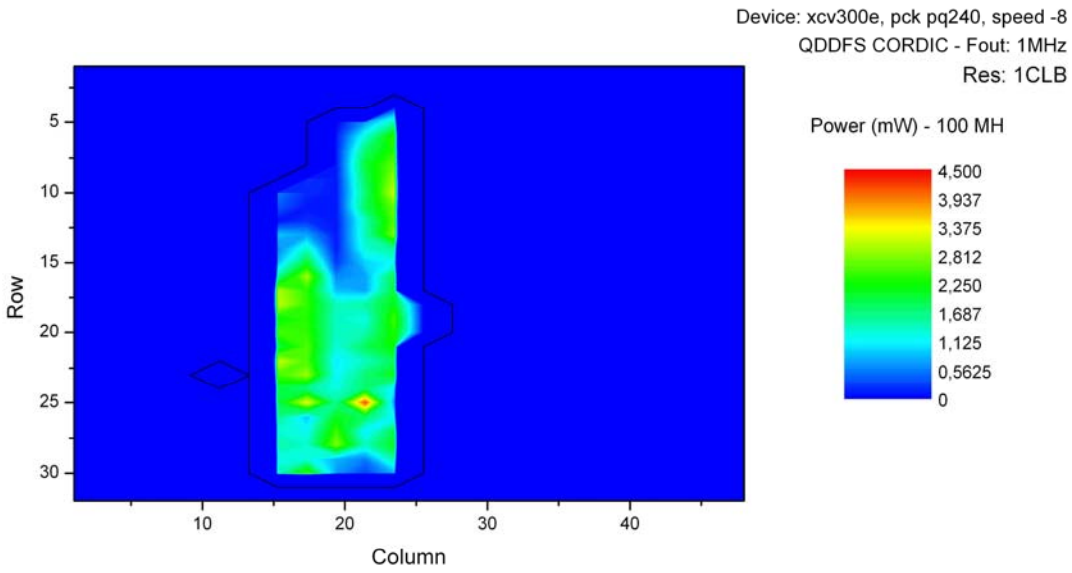


Fig 8.6: Power Map. Resolution 1 CLB for QDDFS-CORDIC, Area-Restricted

Figs. 8.6, 8.7 and 8.8 show the power, activity and capacitance maps for a QDDFS-CORDIC design with restricted area, whereas Figs. 8.3, 8.4 and 8.5 show the same maps but in this case, the design was synthesized without any area restriction.

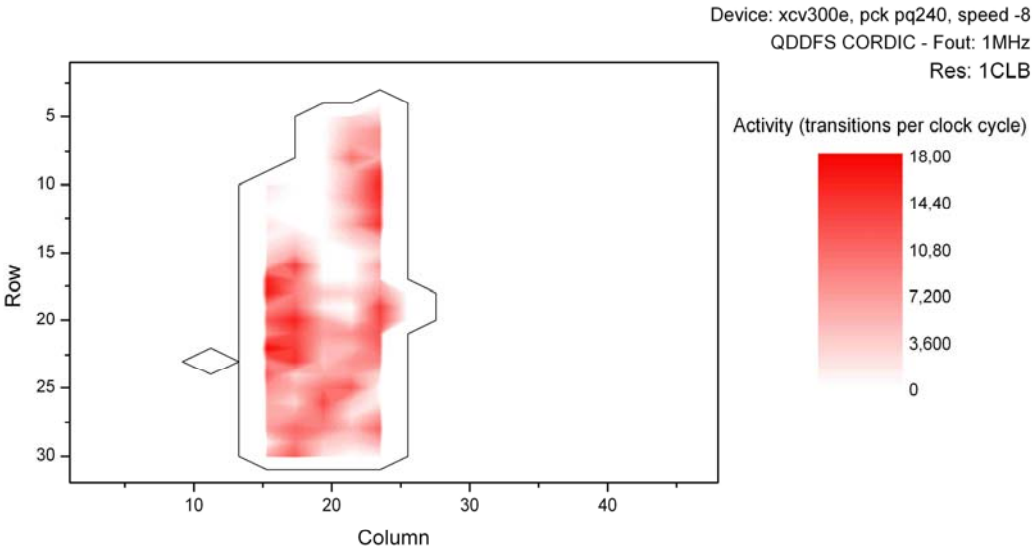


Fig. 8.7: Activity Map. Resolution 1 CLB for QDDFS-CORDIC, Area-Restricted

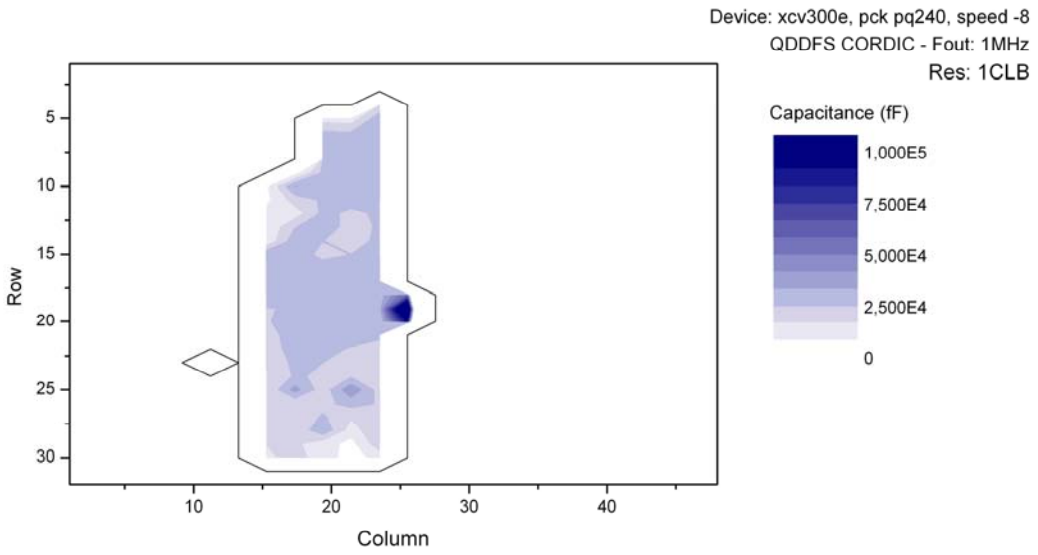


Fig. 8.8: Capacitance Map. Resolution 1 CLB for QDDFS-CORDIC, Area-Restricted

8.2 A First Complete Test Case: FIRDA Filters

For this set of test cases a complete tool evaluation was performed. The experiments were performed over different implementations of a FIR filter using distributed arithmetic [May01]. They use 64 6-bit coefficients, 8-bit input and output words, 12.5 MHz fixed sampling frequency, and a 2/3 cut-off frequency. The difference among these implementations is the internal digit size, from the serial version to the combinational one.

8.2.1 Total Dynamic Power Estimation

Both estimated and measured power values are shown in Table 8.4. Note that the clock frequency must be adjusted given the fixed sampling frequency of 12.5 MHz. For example, in the digit-4 case, the clock frequency must be 25 MHz because two 4-bit digits are processed for each sample.

The min. glitch duration, T_g, calculated as the optimum value for this device is 140 ps. Note that, filtering these short pulses, the error is less than 9% for all the cases. However, when no glitches are filtered, the error can reach more than 20%, always in excess.

Digit	Measured	Tg = 0		Opt. Tg	
		Estimated	Error (%)	Estimated	Error (%)
8	5.328	6.460	21.3	5.804	8.9
4	3.737	3.992	6.8	3.559	-4.8
3	2.625	2.902	10,6	2,626	0.0
2	1.804	1.942	7.7	1.736	-3.7
1	0.821	0.908	10.6	0.817	-0.4

Table 8.4: Total Power Estimation (mW/MHz) for FIRDA Designs

8.2.2 Estimating Power for Individual Nodes

The goal of this test is to evaluate if the results fit the user specified accuracy. The power consumption of the individual nodes must be bounded by a tolerated error within a specified level of confidence. The combinational FIRDA implementation is studied in these experiments.

Fig 8.9 shows relative error distributions for different levels of accuracy. This accuracy is specified in the upper right corner of each histogram. For example, 96/4 means 96% confidence and 4% error. With the statistical approach, the requirement is that the relative error for regular nodes must be bounded to ε with (1-α) x 100% confidence. Therefore, if the number of regular nodes is big enough, it is also expected that less than α x 100% of the regular nodes, have more than ε x 100% error. This is achieved in all the cases shown in Fig. 8.9.

As the power consumption for the individual nodes cannot be physically measured, the comparisons were made against the results obtained from a long simulation run (98% of confidence with less than 2% error). In this set of test cases the threshold for the activity mean, that divides regular and low activity nodes, is 0.25.

The statistical method is applied for the activity, but as the expectation is a linear operation, then (See Eq. 2.6):

$$E(P_i) = 0.5 \cdot C \cdot V_{dd}^2 \cdot E(A_i) \qquad \text{(Eq. 8.1)}$$

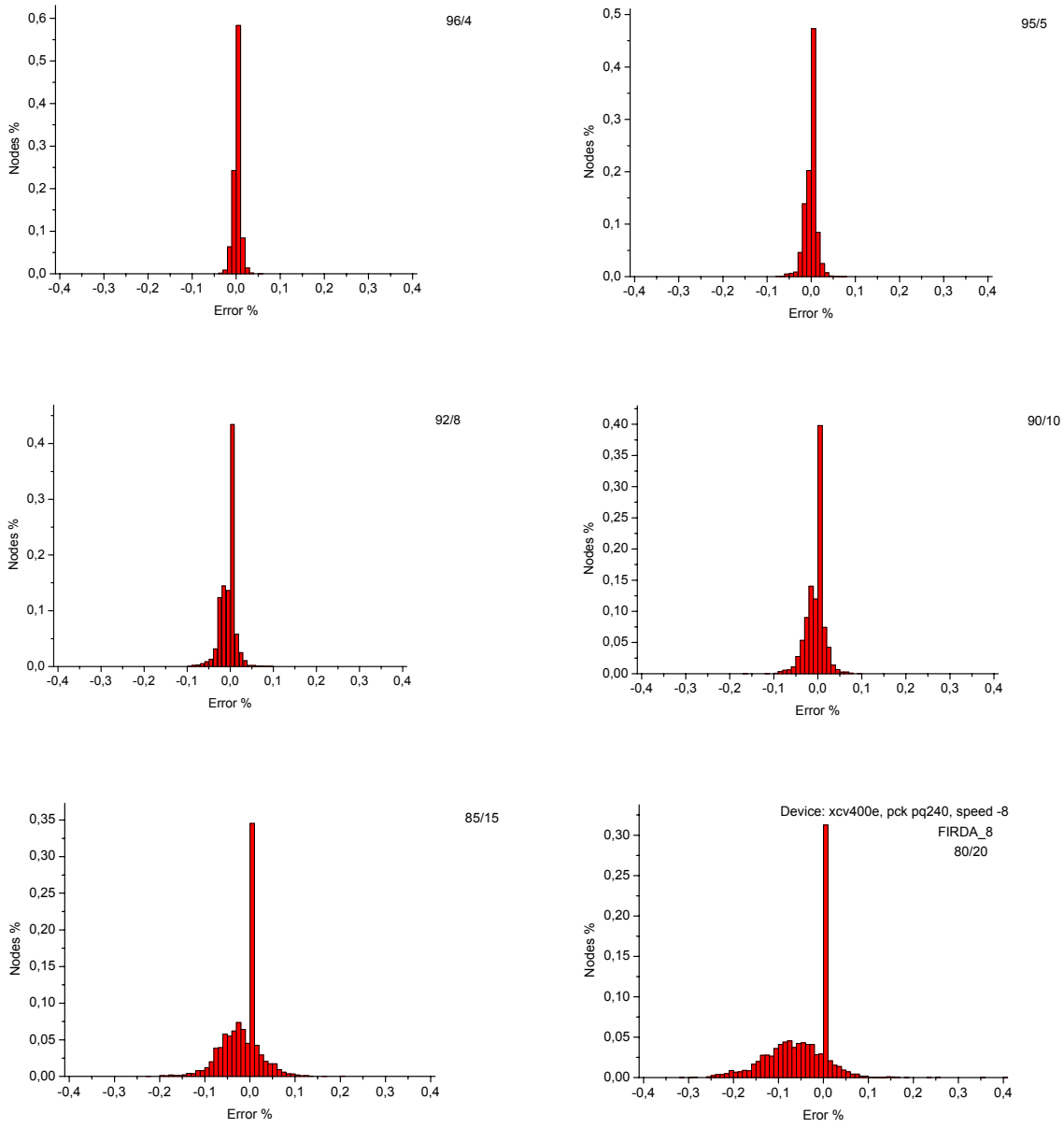


Fig. 8.9: Individual node power: relative error distributions for FIRDA_8

It is supposed that capacitance and supply voltage are constants. In this way, the relative error is the same for both the activity and power consumption for the individual nodes, except when the capacitance is zero. If we call the mean power, μ_P, and the mean activity, μ_A,

$$e_{ri} = \frac{\mu_{Pi,ref} - \mu_{Pi}}{\mu_{Pi,ref}} = \frac{0.5 \cdot C \cdot V_{dd}^2 (\mu_{Ai,ref} - \mu_{Ai})}{0.5 \cdot C \cdot V_{dd}^2 \mu_{Ai,ref}} = \frac{\mu_{Ai,ref} - \mu_{Ai}}{\mu_{Ai,ref}} \quad \text{(Eq. 8.2)}$$

However, when the capacitance reported by the vendor is zero, the power is zero whatever the activity value. Thus, a concentration of zeros in the distributions is observed.

Nevertheless, it is important to note in these experiments that beyond achieving the required accuracy, it is exceeded. This is due to the highest activity nodes, which converge earlier in the estimation process, and are over-analyzed. How can it be observed -and measured- that this accuracy is exceeded? Not less, but much less than α x 100% of the regular nodes, have more than ε x 100% error. To quantify it, it is interesting to study some "effective" accuracy value, a value that reflects the obtained accuracy with the statistical estimation tool, in opposition to the required accuracy.

8.2.2.1 The Effective Accuracy Notion

Accuracy is determined by the tolerated error and the confidence level. It is possible to specify a small error but with low confidence or a higher error with a high confidence level. However, in practice these scenarios are not the most significant ones. Usually, high accuracy is specified selecting a small error value with a high confidence level; and a low accuracy, selecting a high tolerated error with a low confidence level. That suggests that accuracy could be normalized tying together the error and the accuracy in just one value. For example the confidence-error pairs could be 99/1, 98/2, 97/3… 100-ε%/ε%. Given a power estimation run, it can be defined ε_c values so that the number of nodes with a relative error higher than ε_c is less than ε_c x 100% of the normal nodes.

How does this ε_c work? To clarify the idea, Table 8.5 shows a hypothetical example: a circuit with 100 normal nodes with the relative errors in power consumption shown in ascending order. These values are obtained from a run where the tolerated error is 15% with 85% of confidence.

-0,201	-0,077	-0,049	-0,022	0,009
-0,162	-0,075	-0,048	-0,021	0,011
-0,145	-0,074	-0,046	-0,019	0,013
-0,134	-0,072	-0,045	-0,018	0,016
-0,124	-0,070	-0,044	-0,016	0,019
-0,118	-0,068	-0,042	-0,015	0,020
-0,111	-0,067	-0,041	-0,014	0,023
-0,108	-0,066	-0,040	-0,012	0,025
-0,105	-0,065	-0,038	-0,011	0,027
-0,103	-0,064	-0,036	-0,009	0,029
-0,101	-0,062	-0,035	-0,008	0,031
-0,097	-0,060	-0,034	-0,006	0,035
-0,095	-0,059	-0,032	-0,004	0,039
-0,092	-0,057	-0,030	-0,003	0,043
-0,090	-0,056	-0,030	-0,001	0,048
-0,088	-0,054	-0,028	0,000	0,055
-0,086	-0,054	-0,027	0,001	0,061
-0,084	-0,053	-0,026	0,003	0,074
-0,082	-0,051	-0,025	0,006	0,103
-0,079	-0,050	-0,023	0,007	0,171

Table 8.5: Power values of a hypothetical run over a 100-nodes circuit

From Table 8.5, several ε_c values can be analyzed. In the first row of Table 8.6, there are 3% nodes with more than 15% error. In this case, it is clear that the required accuracy is exceeded. The accuracy would be satisfied even if 15% of the nodes have more than 15% error.

ε_c	Nodes	%Nodes
0.15	3	0.03
0.14	4	0.04
0.13	5	0.05
0.12	6	0.06
0.11	8	0.08
0.10	13	0.13
0.09	17	0.17

Table 8.6: Accuracy analysis of a 100-nodes circuit (required 15% error, 85% confidence)

It can also be observed that 8% of the nodes have more than 11% error. In this run, if 11% error and 89% confidence were specified, they would also be satisfied. Nevertheless, 13% of the nodes have more than 10% error. This means that a 10% error and 90% confidence is not reached. From this particular example, the effective accuracy notion can be perceived. The maximum accuracy that could be reached is 11% error with 89% confidence (in fact it is higher considering decimal fractions).

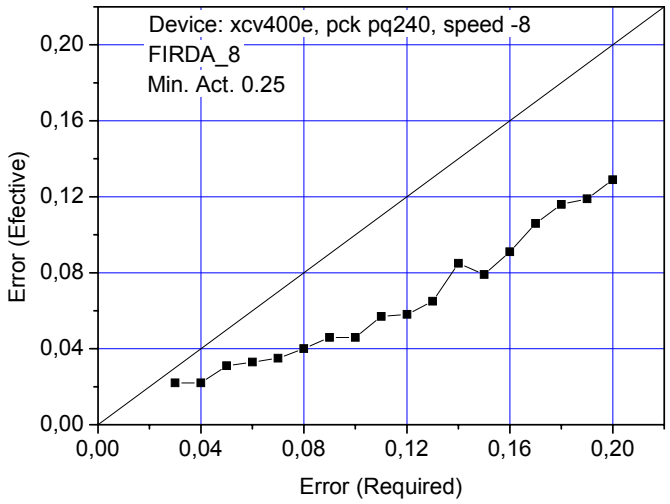


Fig. 8.10: Effective accuracy for FIRDA_8

More formally, the effective accuracy ε_{ef} is defined by the maximum ε_c that can be obtained from the estimation results. It means that the tightest accuracy this run could fit is with a confidence (1- ε_{ef}) x 100% that relative error is less than ε_{ef}.

Fig. 8.10 shows the results of this study for the combinational FIRDA implementation. The effective accuracy is around 1.8 times better than the user defined accuracy for this set of test cases. This value is computed according to Eq. :

$$\sum_{i=1}^{p} \frac{e_i}{e_{eff,i}} \Big/ p, \qquad \text{(Eq. 8.3)}$$

where e_i and $e_{eff,i}$ are the specified and the corresponding effective error respectively for the i-th of p estimation runs (p=18 in Fig. 8.10).

The observation described in this section gives the opportunity for an optimization: a smaller sample can satisfy the user specified error and confidence and it can be done without any loss of accuracy.

8.2.2.2 A-B Nodes Classification[6]

Besides the effective accuracy notion, the nodes do not converge linearly. For example, for the combinational FIRDA version (with 90% accuracy, error is less than 10%, and minimum activity 0.25) it is observed that 98% of the nodes, representing 99% of the power, have met the stopping criteria halfway through the estimation process. This behavior is observed in Fig. 8.11.

From an "economical" point of view, this last 1-2% of the nodes costs a fortune in execution time terms. Normally these extremely expensive 1-2% nodes could be accepted earlier in the estimation process with low accuracy loss. In this way, normal nodes can be classified in two groups: normal and with high cost.

According to this second observation, being ε the tolerated error and (1-ε) x 100% the level of confidence, we can consider the estimation process finished when more than (1-ε) x 100% of the normal nodes have converged. But in order to adjust the optimization strength, a new parameter is defined, so that when more than 1-

[6] The name comes from a technique applied in inventory control -and other areas in Operations Research- where the articles are classified in three groups, A, B, and C according to the total annual expenditure for each item. It helps a company identify the importantt items (5%-20% of type A, accounting for 55%-65 of sales). Then it is possible to concentrate effort on applying inventory control policies for these type A items, producing substancial savings. The ABC classification, sometimes referred to as the 80/20 rule and as Pareto analysis, was devised at General Electric during the 1950s.

*ε*OptStrength*% of the nodes have converged, the estimation process is considered finished. For example, with a specification that error is 10% with 90% of confidence, and 1000 normal nodes, if the optimization strength is 1.0, then the estimation is considered complete when more than 900 nodes have met the stopping criterion. If the parameter is set to 0.5, then the estimation is considered complete when more than 950 nodes have converged.

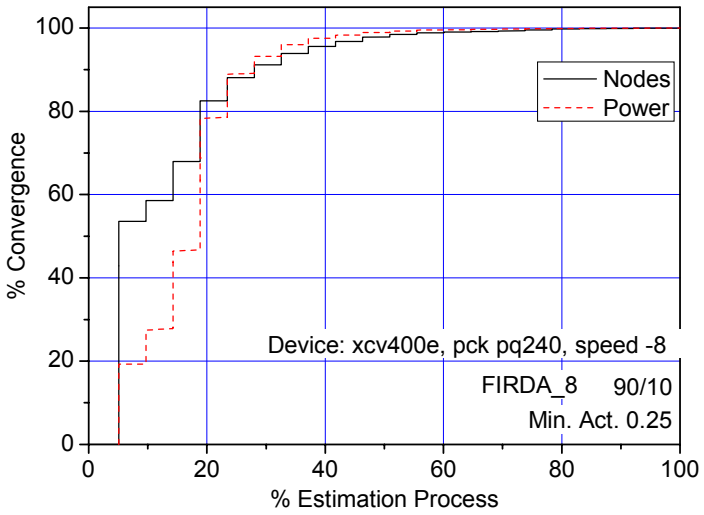


Fig. 8.11: Convergence for FIRDA_8

The condition to stop the estimation is then,

$$\frac{N_{no}}{N_{normal}} \leq e \cdot St \qquad \text{(Eq. 8.4)}$$

Where N_{normal} is the normal nodes count, N_{no}, is the number of normal nodes that have not converged, ε is the user specified error and St is the specified optimization strength.

Fig. 8.12 shows how, as the optimization strength is higher; the effective error approaches the specified one (10%) making every simulated clock cycle in the taken sample useful and efficient. Furthermore, a dramatic saving in execution time is observed. The savings are expressed in relative terms where 1.0 corresponds to the case without optimization strength. For example, Fig. 8.11 shows that when St = 1.0, the sample size is less than 40% of the one without optimization.

Fig. 8.13 shows relative error distributions for different user specified optimization strengths. In the experiments in this subsection it is specified with 90% confidence that error is less than 10%, and the minimum activity threshold is 0.25. Although the effective accuracy is a random variable, it is clear that it approaches to the one specified by the user as the optimization strength increases. The goal here is to connect the effective accuracy concept with the proposed nodes classification. It is verified that with this optimization technique the effective accuracy comes close to the specified one.

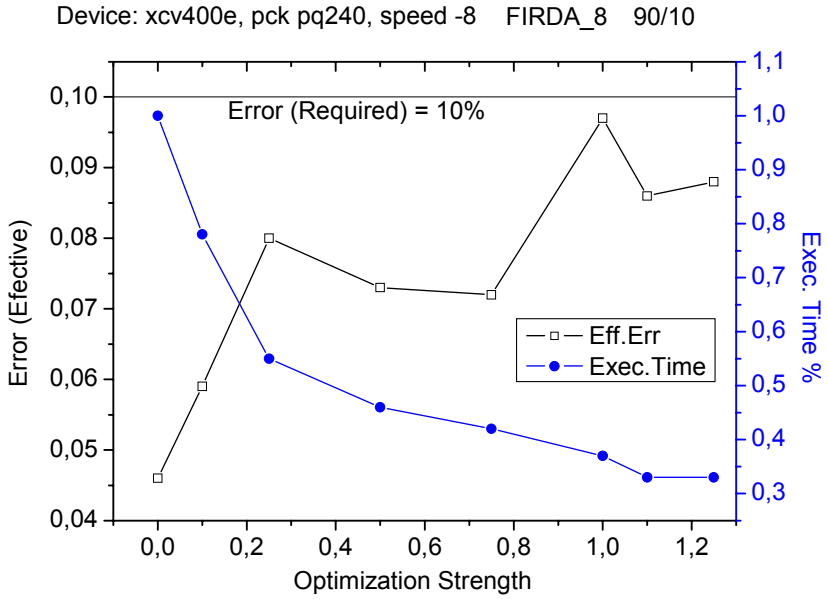


Fig. 8.12: Execution time and effective accuracy in function of the optimization strength for the FIRDA_8 test circuit

8.2.3 Accuracy vs. Execution Time Tradeoff

In is interesting to study how the execution time depends on the required accuracy. Besides the tolerated error and confidence, the accuracy is in principle also a function of the activity threshold that divides the nodes into normal and the low activity ones. The tunable accuracy-execution time properties are studied in this section. Fig. 8.14 on page 149 shows that for both normal and low activity nodes, as error decreases and the confidence level increases, the number of samples increases monotonically. For all the experiments in this set, the input signals were specified as time independent, with probability 0.5 and 0.5 transitions per clock cycle. The min. activity threshold is 0.25.

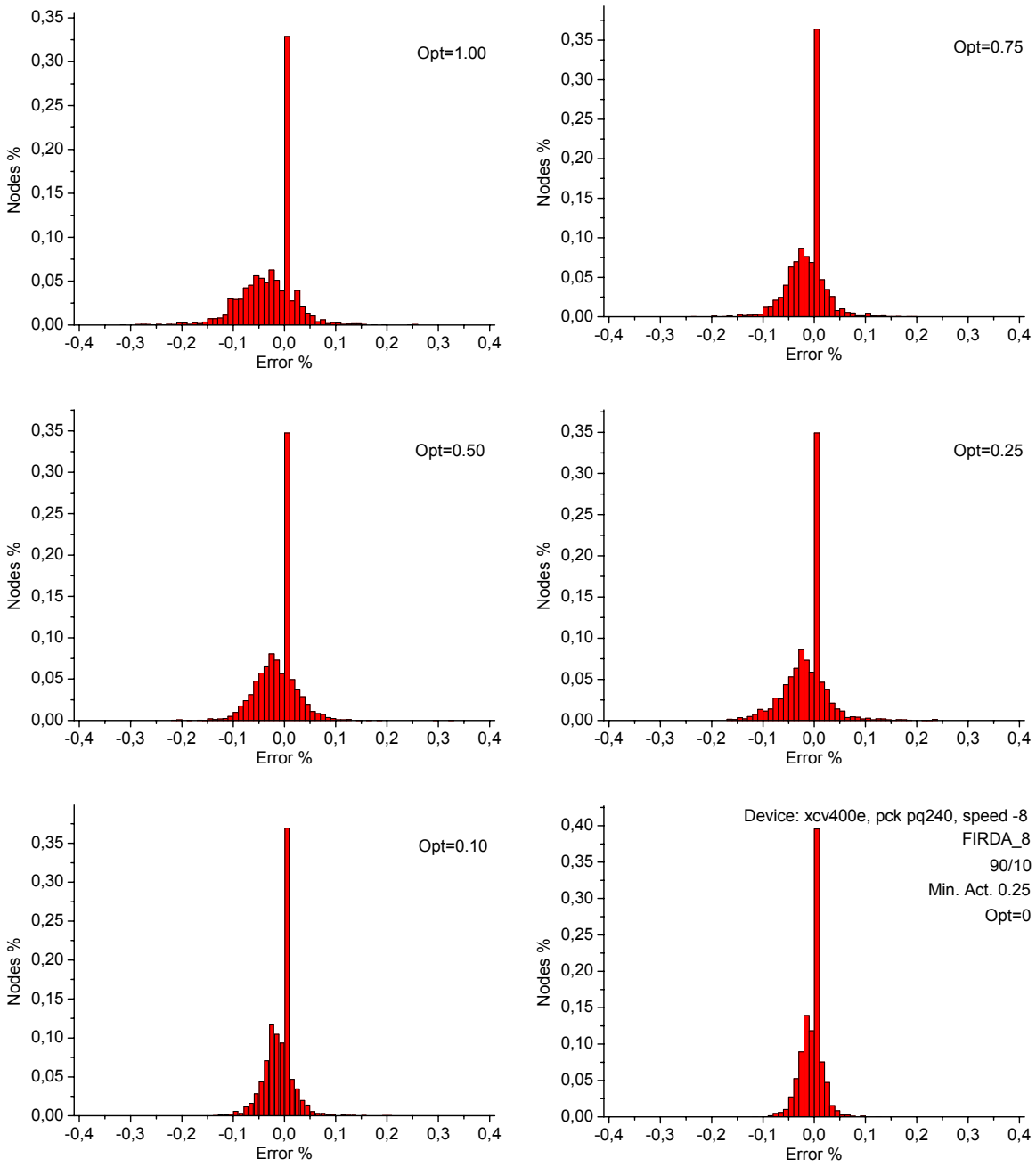


Fig. 8.13: Individual node power: relative error distributions for FIRDA_8

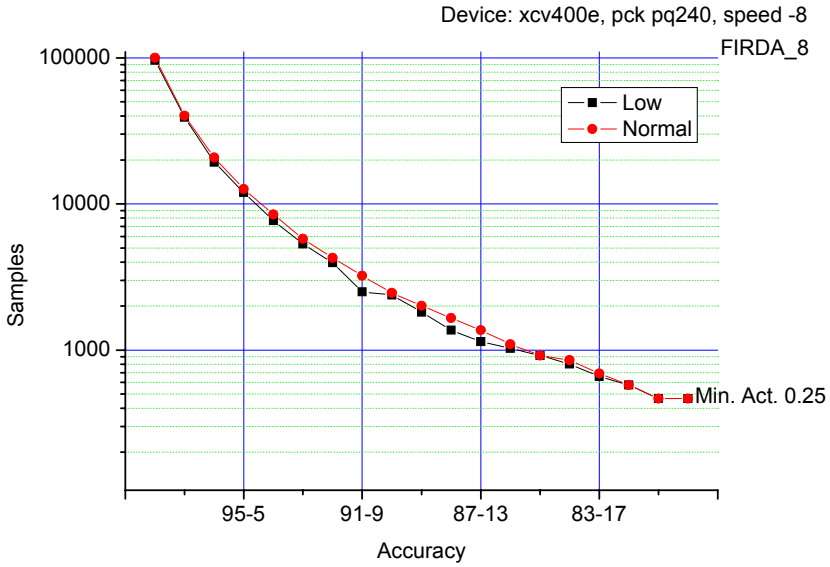


Fig. 8.14: Accuracy vs execution time tradeoff for FIRDA_8

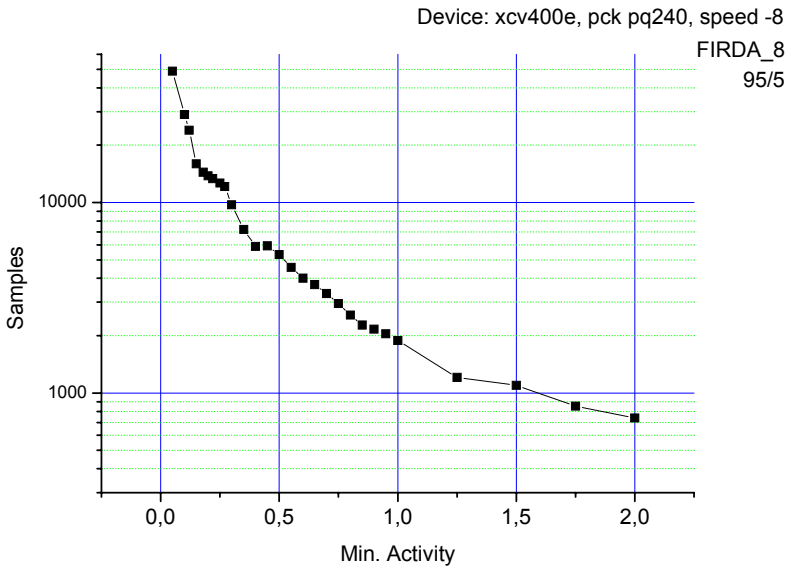


Fig. 8.15 Accuracy vs. execution time tradeoff for FIRDA_8

On the other hand, Fig. 8.15 shows the results of different runs with 95% confidence and 5% error, varying the min. activity threshold. As expected, the required number of samples increases monotonically as the minimum mean decreases.

Even with this strong functional relationship between the activity threshold and the execution time, a low impact on the total power estimation error is observed in this test

case where the activity threshold changes. In fact, the correlation between this parameter of accuracy and the total power relative error is very low, as shown in Fig. 8.16.

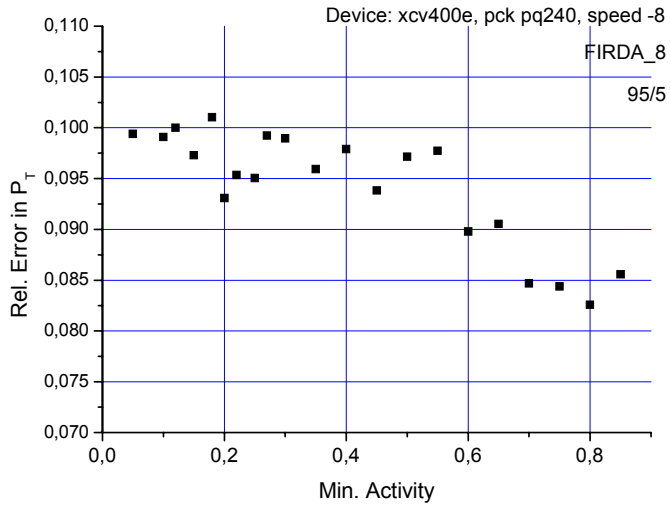


Fig. 8.16: Min. Activity threshold vs. total relative error in total power for FIRDA_8

A more meaningful picture representing the accuracy vs. execution time tradeoff is Fig. 8.17, where all the involved variables are present and the behavior of the estimation system is characterized. The *x*-axis represents the accuracy, where a x_i values correspond to an x_i% error with 100-x_i% confidence. This experiment confirms the robustness of the technique, allowing a tunable accuracy.

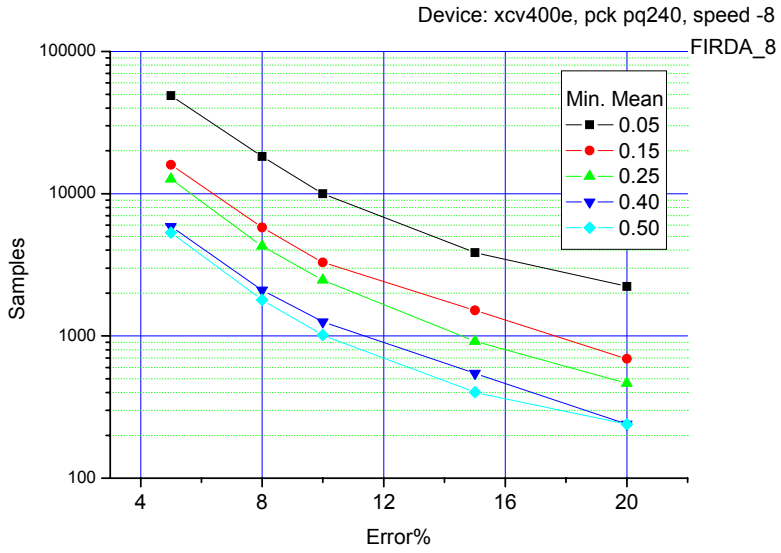


Fig. 8.17: Characterization of the accuracy/execution-time tradeoff for FIRDA_8

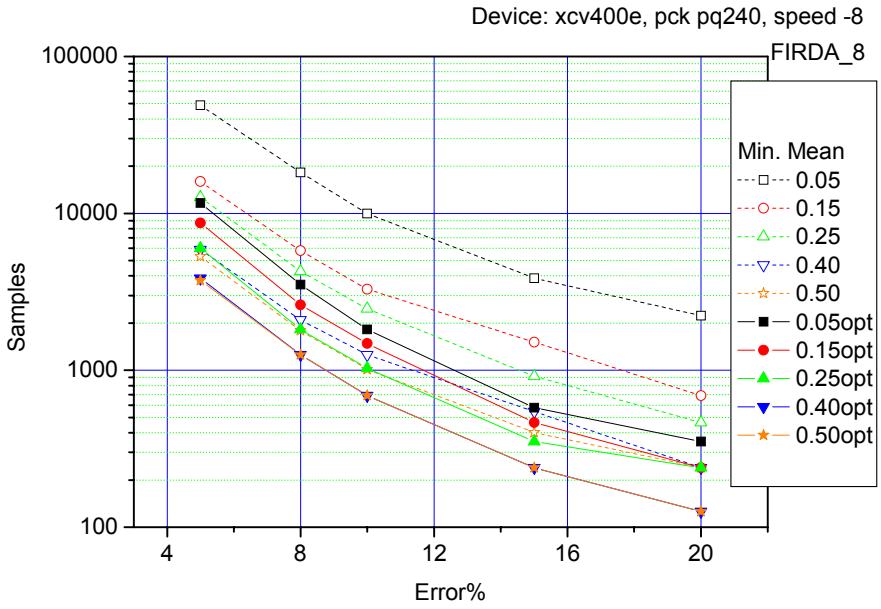


Fig. 8.18: Characterization of the accuracy/execution-time tradeoff for FIRDA_8. Optimization strength 0.75

Nevertheless, all the previous cases in this section were run with an optimization strength value equal to zero. Fig. 8.18 shows the same characterization for optimization strength 0.75 and the comparison with the non optimized case. The dashed lines represent the cases without optimization.

In order to give more information about the results in Fig. 8.18, Table 8.7 shows the execution time savings with respect to the non optimized case.

	Min. Act.				
Error	**0,05**	**0,15**	**0,25**	**0,4**	**0,5**
5	0,76	0,45	0,53	0,35	0,30
8	0,81	0,55	0,57	0,40	0,30
10	0,82	0,55	0,58	0,45	0,32
15	0,85	0,69	0,62	0,56	0,40
20	0,84	0,65	0,49	0,47	0,47

Table 8.7: Execution time savings for FIRDA_8. Optimization strength 0.75

It is observed that the best savings are obtained in the most favorable cases, where the required accuracy and execution times are high.

8.2.4 Tool's Evaluation

As shown in section 8.1.3, beyond the results in relative terms, it is also interesting to consider the absolute execution times. Table 8.8 and Fig. 8.19 show the results for the combinational FIRDA with the optimization strength set to 0.75. The error is 5%, 95% confidence, and min. mean is 0.25.

	Task	Exec. Time [secs]
Activity Estimation	Input vector generation	13
	Simulation	742
	Saving	24
	Transition análisis	2669
	Statistics computation	16
	Stopping criteria evaluation	17
Power Computation	VHDL parking	1
	XDL parking	1
	XML generation	107
	XPower execution	12
	PWA parking	0
	Report writing	3
	Maps generation	10

Table 8.8: A-DyP execution Time for FIRDA_8

In order to have an idea about the absolute A-DyP execution times, Table 8.9 summarizes them by sub-system. This run, where the accuracy is relatively high requires 1 hour (Pentium 1.6 GHz, main memory 512 MB). In Table 8.10 the same design is tested with a lower accuracy requirement: 15% error with 85% confidence. Now the total execution time is about 6 minutes. Total time includes all the necessary procedures to run the test. For example, it includes the elaboration done by the

simulator for the VHDL design. Nevertheless, this time does not include the compilation time, which is necessary just once for the first run of each design.

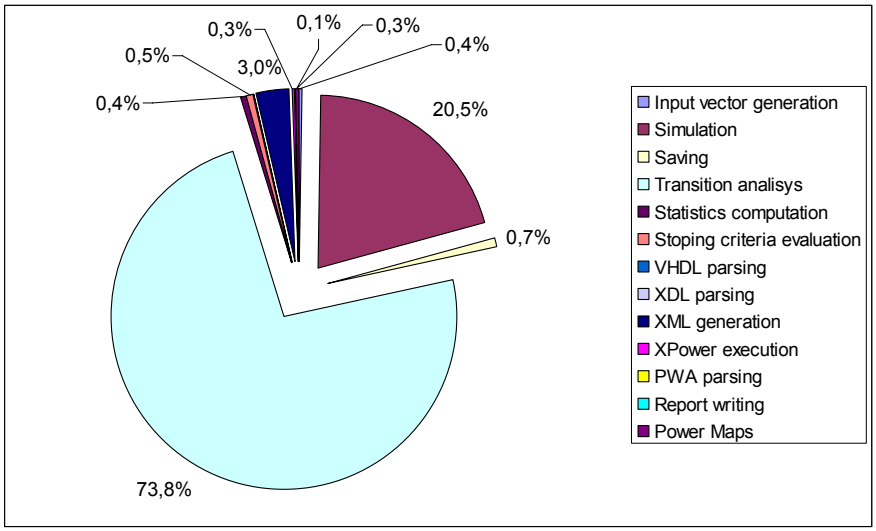


Fig. 8.19: A-DyP Execution Time for FIRDA_8

Task	**Exec. Time [sec]**
Total Execution	3678
Initialization	49
Activity Estimation	3481
Power Computation	134
Tcl/Tk script	14

Table 8.9: A-DyP total execution time. FIRDA_8, high accuracy

Task	**Exec. Time [sec]**
Total Execution	387
Initialization	51
Activity Estimation	191
Power Computation	141
Tcl/Tk script	4

Table 8.10: A-DyP total execution time. FIRDA_8, low accuracy

8.2.5 Power Maps

In this sub-section some power, capacitance and activity maps are shown. Fig. 8.20 and 8.21 show two power maps for the FIRDA_8 circuit with different resolutions. For the first one, the power consumption is added within a 1x1 CLB square. The second figure lumps together the power consumption within the 4x4 CLBs. Although it seems less accurate, the second alternative could give a better idea about the internal temperature in the die.

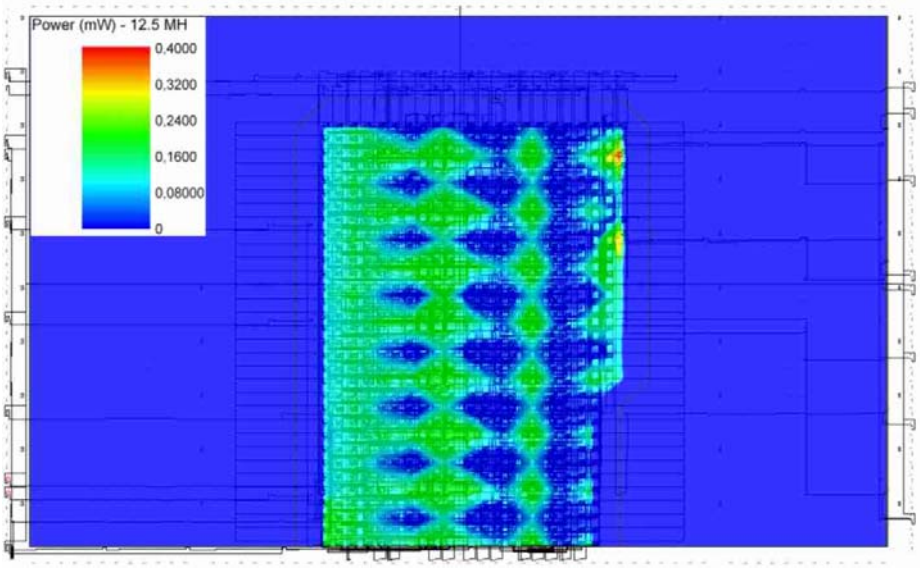


Fig. 8.20: Power Map. Resolution 1 CLB

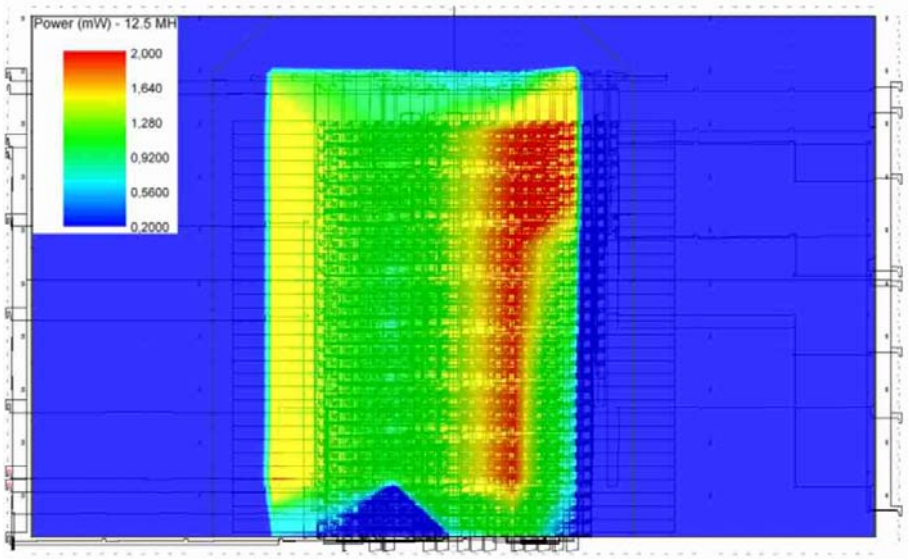


Fig. 8.21: Power Map. Resolution 4 CLBs

Fig. 8.22 and 8.23 show a capacitance and an activity map for the FIRDA_8 circuit respectively.

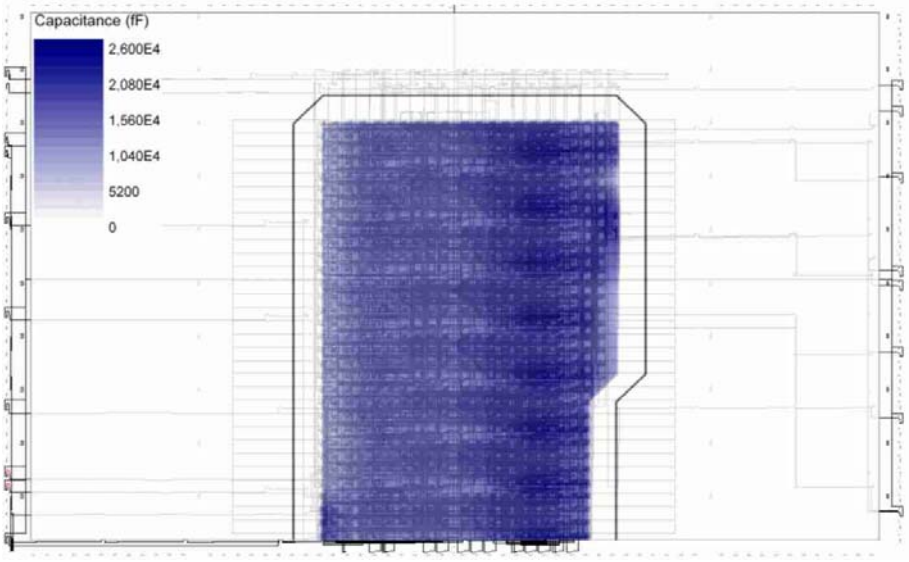


Fig. 8.22: Capacitance Map. Resolution 1 CLB

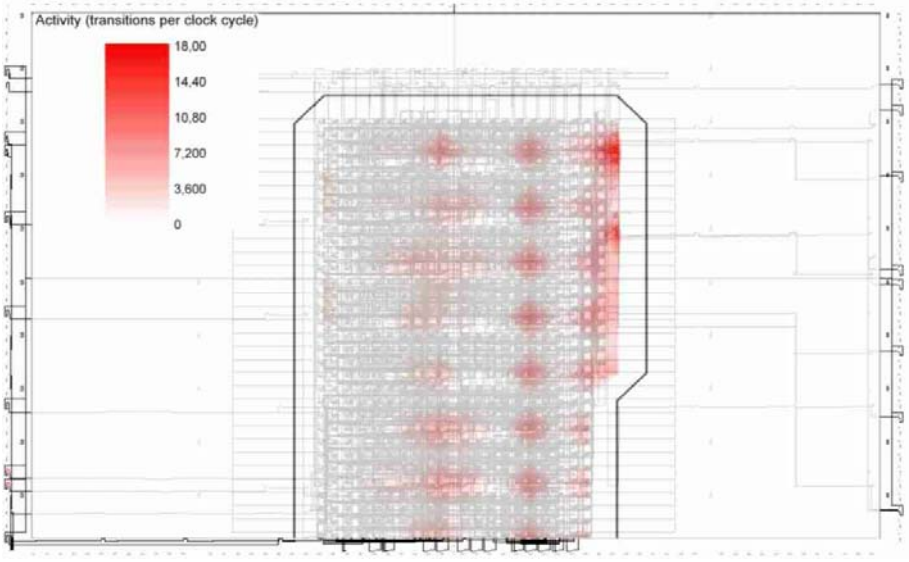


Fig. 8.23: Activity Map. Resolution 1 CLB

In Section 8.2.3 it is shown that correlation between the activity threshold and the *total* power relative error is very low. The total power is a macroscopic magnitude but it is also useful to study what happens at a higher resolution level, for example at the CLBs scale, in the programmable-element world.

Fig. 8.24 shows a map with the differences between 2 simulation runs with activity thresholds 0.18 and 0.8. Those are the estimations with the highest difference in total power for the activity threshold studied range. In this figure, the differences are around 10 µW, being the CLB power consumption in the 0-370 µW range. Consequently, these differences could be considered weak noise. Nevertheless, if the relative differences are computed, the results show high values. Relative differences are computed according to Eq. 8.5.

$$P_{rel,mM,x,y} = \frac{\left| P_{m,x,y} - P_{M,x,y} \right|}{P_{m,x,y}} \qquad \text{(Eq. 8.5)}$$

Where $P_{S,x,y}$ is the power consumed at CLB in column x, row y, and simulation run S. Average, standard deviation and maximum values for these differences are shown in Table 8.11 for square groups of n x n CLBs.

Resolution	Mean	Std. Deviation	Maximum
1x1	3,3%	7,5%	88,4%
2x2	2,9%	6,4%	82,2%
4x4	1,8%	1,4%	6,4%

Table 8.11: Relative differences between two simulation runs with different minimum activity thresholds

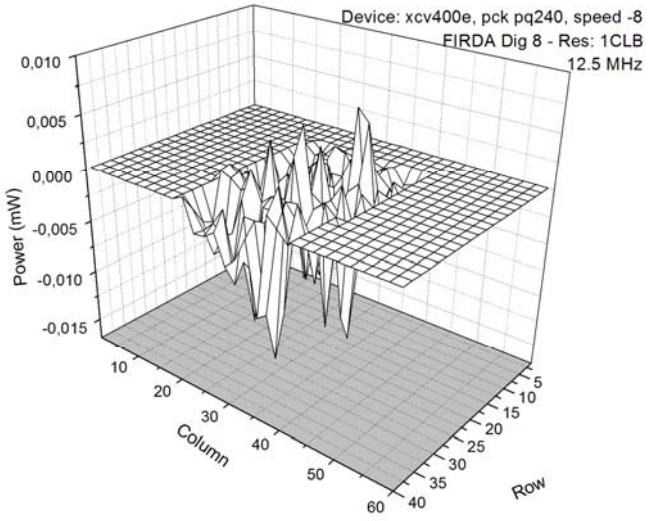


Fig. 8.24: Map that shows the difference between 2 estimations with activity thresholds 0.18 and 0.8. Resolution 1x1 CLB

Now, it is clear that the activity threshold *is* an accuracy parameter for individual node estimations and, in general, low resolution power estimations with the implemented technique.

Fig. 8.25 shows some projections from de 3D representation keeping specific rows constant.

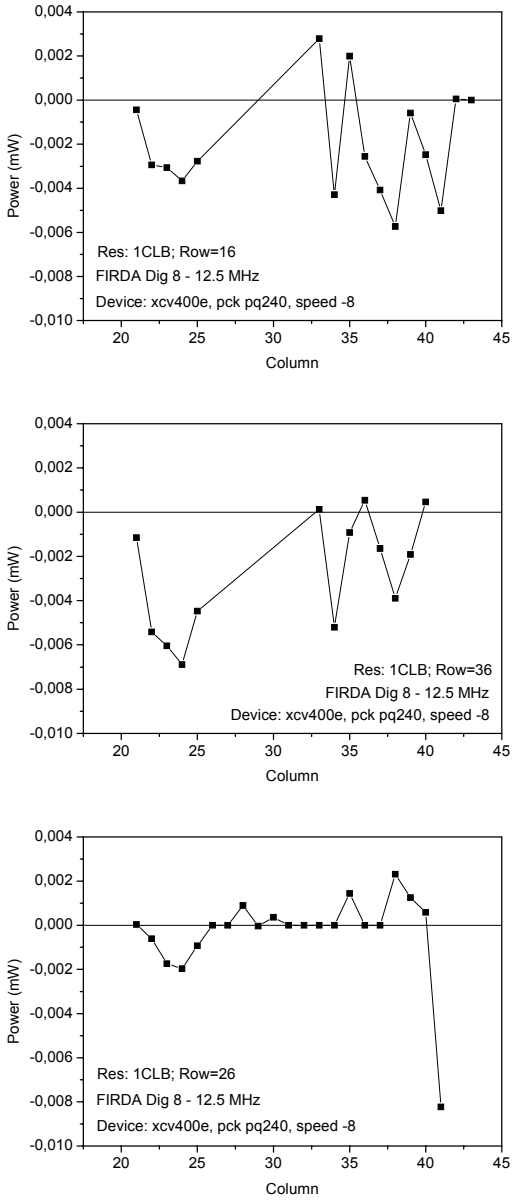


Fig. 8.25: 2D projections from the 3D map of Fig. 8.23

8.2.6 Energy Analysis or Energy of the Computation

All the FIRDA circuits are evaluated in this section, from the combinational version to the serial one, including digit sizes of 8, 4, 3, 2 and 1. The circuit behavior from the user point of view is the same for all the cases; the I/O data rate is also equal. Just the clock frequency must be adjusted to serve the incoming data taking into consideration the current digit size. The combinational version requires the slowest clock frequency but needs the highest area. The serial (digit-1) version needs the smallest area at the highest clock frequency. The question is now, which is the architecture with the lowest power figure?

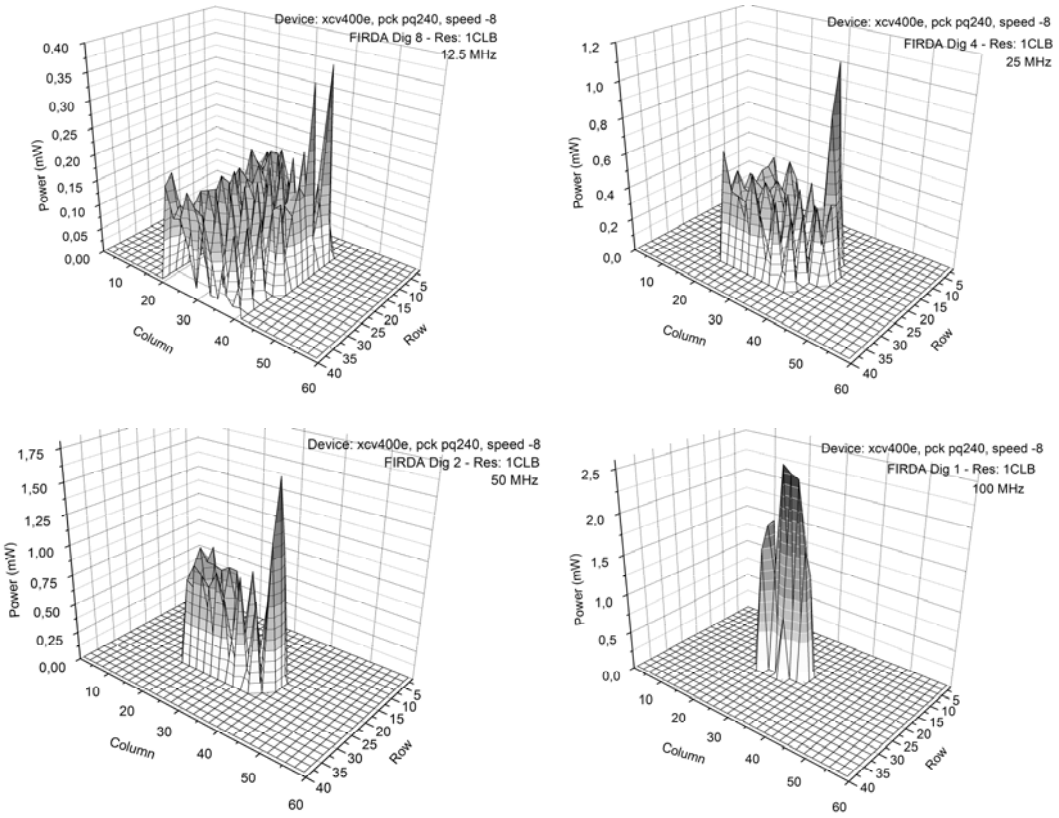


Fig. 8.26: Power Maps for the FIRDA circuits

Revisiting Eq. 2.7:

$$P = \frac{1}{2} V_{dd}^2 f_{clk} \sum_i \hat{C}_i \hat{E}(sw)_i \ ,$$

As the digit size decreases, f_{clk} must be increased and the total power will augment in the same way. At the same time, the area decreases and this means that capacitance is reduced. What will the prevailing effect be on power consumption? Will the power increase or decrease in serialized versions? The power maps for the different implementations are shown in Fig. 8.26. As the area increases, a lower power per unit of area is observed. Conversely, for the digit-1 case the area is minimal but the energy consumption is concentrated in this region of the die.

It is clear that the computation done over the data is the same, and at the same rate. Could it be thought of as some energy associated with a computation? In this case, the energy per operation should be the same for all the FIRDA architectures.

In [Fey96], a study about the energy and thermodynamics of computing is presented. In that work it is concluded that, for practical cases –practical in opposition to idealized machines that operates infinitesimally slow, for example-, there is an amount of energy proportional to the computational work. Therefore, this test case is an opportunity to verify these ideas (see Section 2.3.1).

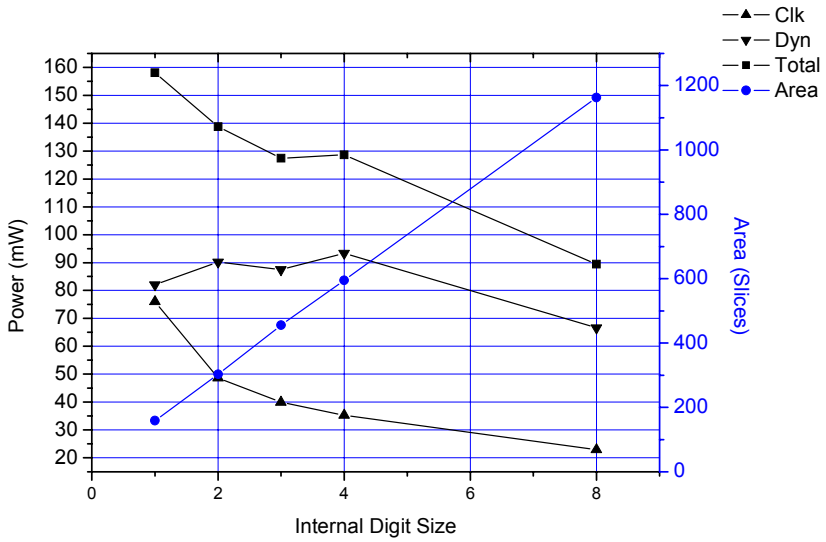


Fig. 8.27: Power and area for the different FIRDA versions

Depending on the application constraints, the best version of the filter can be selected according to its power consumption or area occupation. Fig. 8.26 shows that when the area increases, the total power consumption decreases. If the design is restricted by area, the serial version should be used, being 8 times smaller than the

parallel one. However, when the power budget is reduced, the parallel implementation is the preferred one, with half as much again as the serial version.

It is interesting to analyze the synchronization and logic power separately. Fig. 8.27 shows that the logic power varies slightly. It is higher in serial versions due to the additional control logic. In this way the idea of energy associated with a computation have an empirical counterpart. It is important to note that for this proposition to be true, the spurious activity must be reduced. This is the case in these circuits due to the use of pipelines.

If both power and area need to be optimized together, the serial version is the best choice, as shown in Fig. 8.28. In fact, the synchronization power for the serial version is a little more than 3 times higher than the parallel circuit: As in pipelining, additional power is required to manage the data. Consequently, total power, including synchronization and dynamic power for the serial version, is almost double that of the parallel implementation.

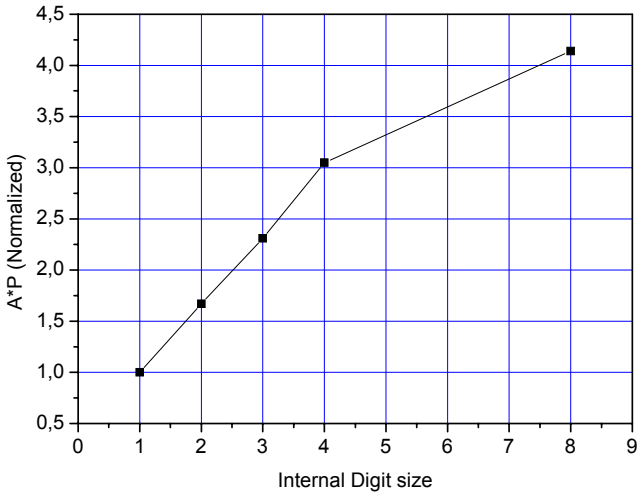


Fig. 8.28: Power*Area for the different FIRDA versions

8.3 Impact of the Input Patterns Definition

In this section some test circuits are evaluated with different input pattern definitions as shown in Table 8.12.

Test case number 1 can be considered as the standard one where the designer does not have information about the scenario where the circuit will be used, or he wants to obtain a first approach to the circuit power consumption values.

In cases 2 and 3 the activity is not the same for the different bits within the operands at the primary inputs but it is increased from one end to the other all along the operands. In the same way, cases 4 and 5 are close to the cases where the circuit inputs are 2's complement numbers and the sign bits switches less than the rest.

Case number 6 is a high activity and power consumption one. It can serve to know a power consumption value near to the maximum. On the other hand, case 7 is a low power one. A high signal probability means that almost all the time the signals are '1' and that the activity must be low.

Finally, cases 8 and 9 exercise the circuit within two particular situations were some circuit inputs are connected to a counter's output.

Test Case	Description
1	All inputs are independent random patterns
2	The activity for the MSB is set to 0.05 and is increased linearly to 0.95 for the LSB.
3	The activity for the MSB is set to 0.95 and is decreased linearly to 0.05 for the LSB.
4	The activity is set to 0.05 for the 4 MSBs and 0.5 for the rest.
5	The activity is set to 0.05 for the 8 MSBs and 0.5 for the rest.
6	The activity is set to 0.95 for all the bits.
7	The signal probability is set to 0.95 for all the bits.
8	The 4 LSBs are connected to a counter and the rest are independent random bits.
9	The 4 MSBs are connected to a counter and the rest are independent random bits.

Table 8.12: User Defined input Patterns

8.3.1 Total Dynamic Power Estimation

In this sub section the total measured and estimated power are reported for every input pattern definition specified in Table 8.12. The studied circuits are SUM32, MUL32, DIV16P and MUL16P and the results are presented in Tables 8.13 to 8.16 respectively. All of them are implemented in a Virtex XCV50PQ240-4 device.

Case	Measured	Tg = 0		Opt. Tg	
		Estimated	Error (%)	Estimated	Error (%)
1	1.352	1.40	3.5	1.36	0.6
2	1.258	1.42	12.9	1.35	7.2
3	1.273	1.39	9.6	1.36	6.8
4	1.217	1.26	3.2	1.21	-0.3
5	1.086	1.11	1.8	1.07	-1.6
6	1.800	2.61	45.2	2.47	37.3
7	0.159	0.15	-8.0	0.15	-6.5
8	1.319	1.38	4.5	1.31	-0.4
9	1.332	1.38	3.8	1.34	0.4

Table 8.13: Total Power Estimation for the SUM32 Design

In Table 8.14, for the combinational 32-bit multiplier, the estimation was not possible without any glitch filtering because the extremely high activity reported by the simulator produces overflows in the program that parses the VCD files. These estimations are made filtering glitches shorter than 50 ps. Even in this case, the error can reach more than 500%! always in excess. Filtering these short pulses, the error is less than 10% for all the cases except one with a 29% error.

Case	Measured	Tg = 50		Opt. Tg	
		Estimated	Error (%)	Estimated	Error (%)
1	18.859	92.261	389.2	17.256	-8.5
2	16.297	90.952	458.1	17.153	5.3
3	17.309	93.352	439.3	16.47	-4.8
4	16.634	89.223	436.4	16.483	-0.9
5	16.172	79.483	391.5	15.528	-4.0
6	22.616	120.561	433.1	22.756	0.6
7	4.898	31.709	547.5	6.3165	29.0
8	18.934	93.594	394.3	18.29	-3.4
9	19.034	92.692	387.0	18.141	-4.7

Table 8.14: Total Power Estimation for the MUL32 Design

Case	Measured	Tg = 0		Opt. Tg	
		Estimated	Error (%)	Estimated	Error (%)
1	4.710	7.25	53.9	4.60	-2.4
2	3.941	6.39	62.0	4.01	1.7
3	4.501	7.05	56.5	4.38	-2.6
4	3.672	5.86	59.7	3.76	2.3
5	4.238	6.95	63.9	4.30	1.5
6	4.978	11.53	131.7	6.13	23.2
7	1.366	2.13	56.0	1.51	10.7
8	4.703	7.17	52.4	4.57	-2.7
9	4.330	6.82	57.5	4.25	-2.5

Table 8.15: Total Power Estimation for the DIV16P Design

Case	Measured	Tg = 0		Opt. Tg	
		Estimated	Error (%)	Estimated	Error (%)
1	3,392	5,194	53,1	3,264	-3,8
2	2,939	4,813	63,8	2,938	-0,1
3	3,210	4,873	51,8	3,107	-3,2
4	2,763	4,349	57,4	2,769	0,2
5	2,814	4,462	58,5	3,155	12,1
6	3,966	7,764	95,8	4,216	6,3
7	0,732	1,247	70,3	0,917	25,2
8	3,359	5,126	52,6	3,222	-4,1
9	3,295	5,157	56,5	3,175	-3,6

Table 8.16: Total Power Estimation for the MUL16P Design

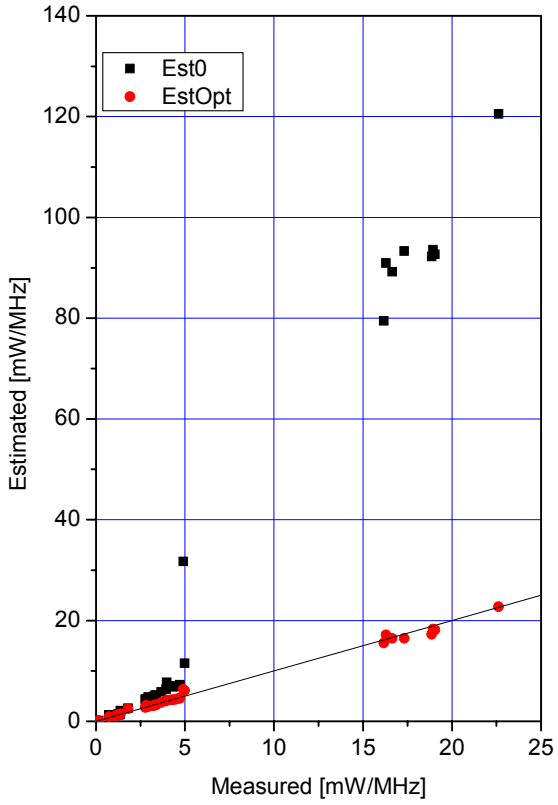


Fig. 8.29: Total Power Estimations in a Virtex 50 Device.

Fig. 8.29 graphically resumes the data shown in Tables 8.13 to 8.16. The black squares represent the estimations before an adequate glitch filtering while the red dots correspond to the values after a short pulse filtering with a value obtained for the device, in this case XCV50PQ240-4. As in all the experiments, it is noticeable or even absurd the overestimation in the number of transitions that a standard simulator report for a post PAR circuit. The line represents the situation without error.

The differences in total power consumption for every circuit and input pattern definition are shown in Fig. 8.30. Figs. 8.30.a, b, c and d show the result for SUM32, MUL32, DIV16P and MUL16P respectively.

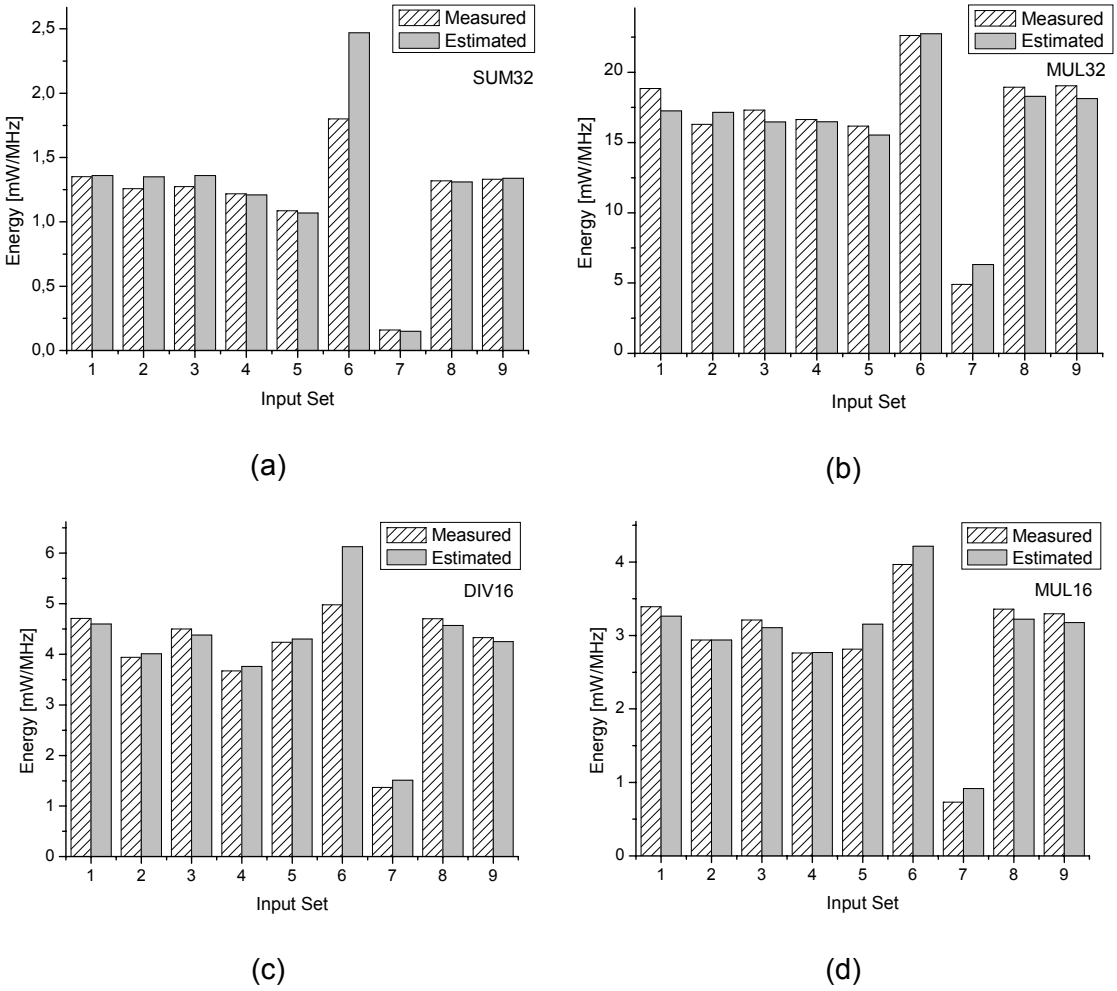


Fig. 8.30: Total Power for the different Input Patterns (See Table 8.11).

The main conclusion from Fig. 8.29 is that a user can define different input sets and obtain any power figure between zero and at least the value obtained for the input definition number 6. In other words, it is essential to specify an input set that is as real as possible in order to obtain meaningful results.

8.3.2 Dynamic Power Estimation for Individual Nodes

The power consumption for individual nodes is also completely different for each defined input pattern. Each set can produce hot spots or hot areas in different places in the FPGA and with different amplitudes. Fig. 8.31 shows a power map for the input pattern definition number 1 with the MUL16P design layout in the background. Cases 2-9 are shown in Fig. 8.32 for the same design. For all the figures the same colors represent the same power values as is shown in Fig. 8.31. Furthermore, the red color represents 2.5 mW or more.

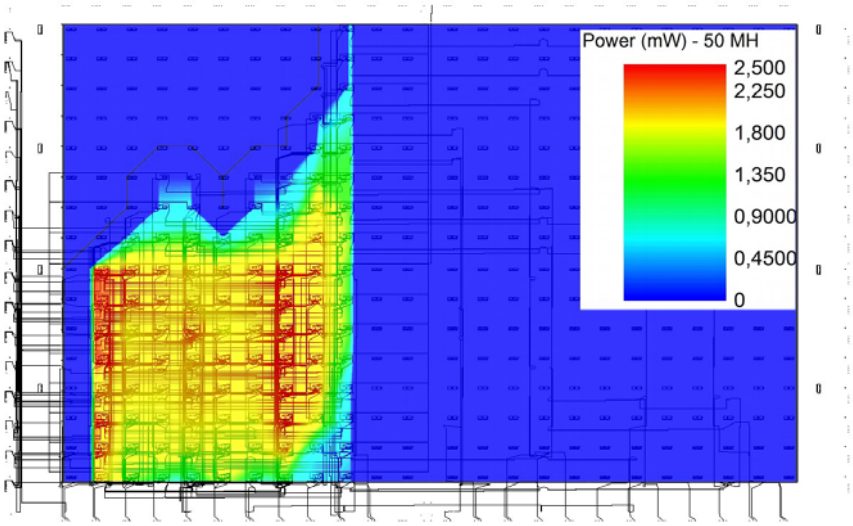
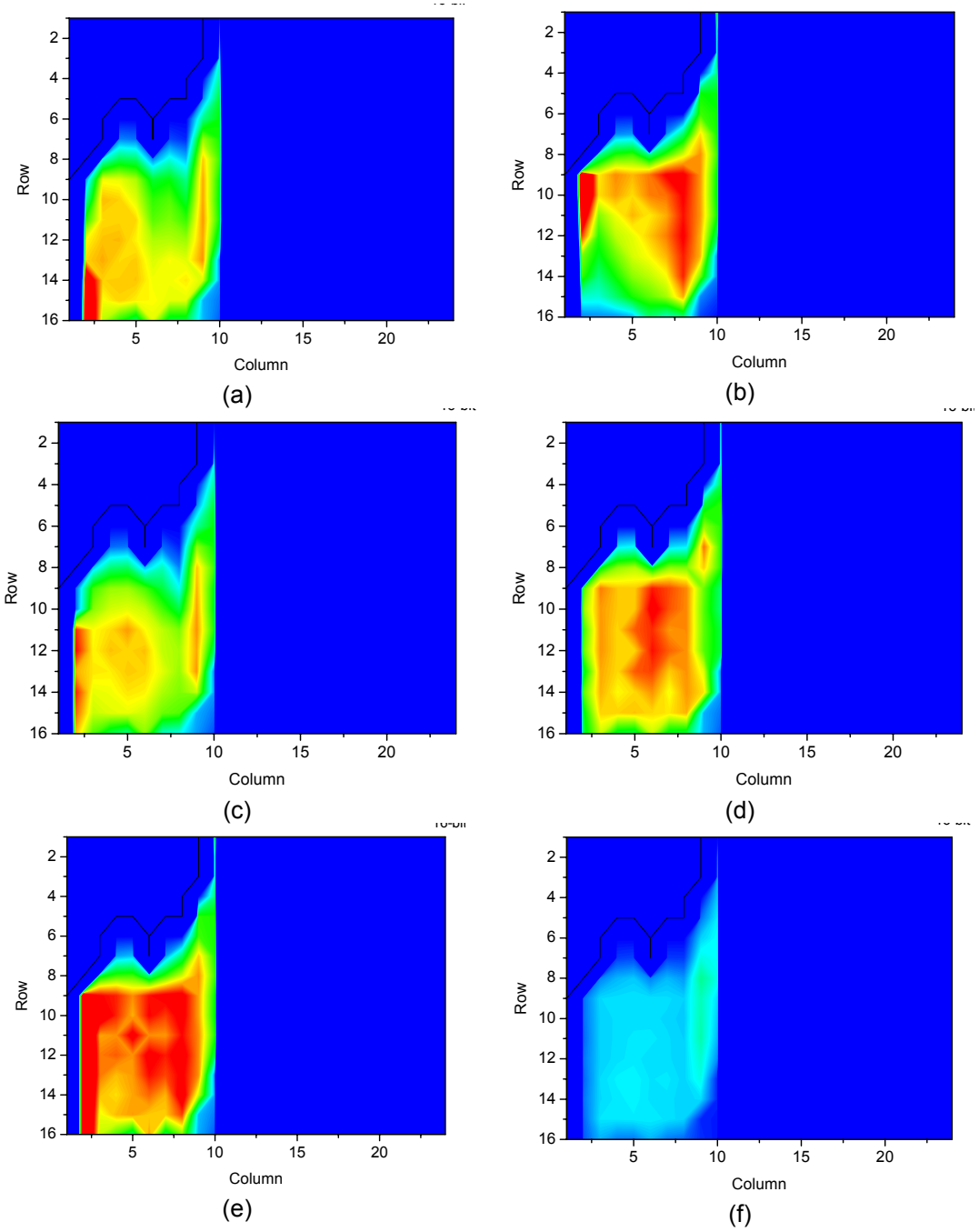
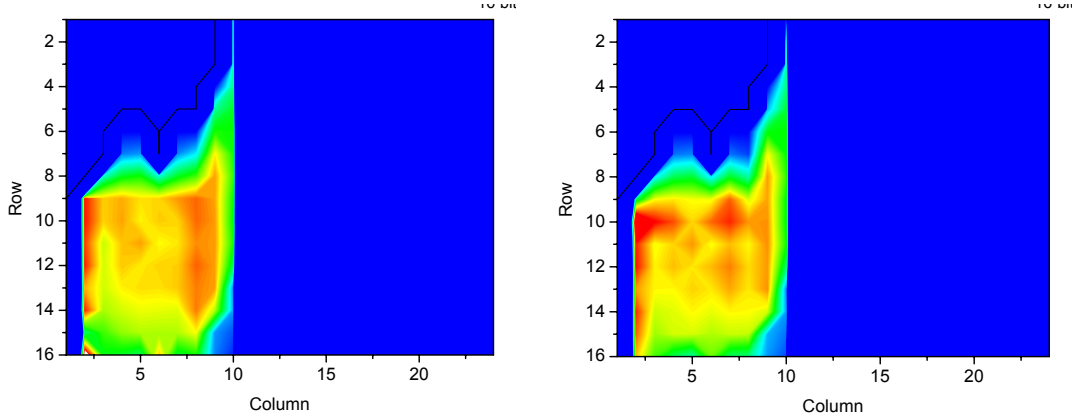


Fig. 8.31 Power Map for MUL16. Resolution 1 CLBs

In Fig. 8.32.a and 8.32.b it is clear that the inputs LSB (down) and MSB (up) are the most active respectively and gradually the power consumption decreases towards the MSB and LSB respectively.

Fig 8.32.e clearly represents the most power consumption case with almost all the design area in red. On the other hand, Fig. 8.32.f is the less active case where the 1 x 1 most active CLB squares consume about 0.7 mW.


Row
Column
(a)
(b)
(c)
(d)
(e)
(f)



(g) (h)

Fig. 8.32: Power Map. Resolution 1 CLBs

From these figures it can be observed that, given a power or activity map, it could be inferred which input patterns are applied to the primary inputs. Conversely, applying specific input pattern sets, it could be possible to generate specific power and activity maps, both static and dynamic. A consecutive sequence of maps could be considered or called a power movie. These movies naturally have several interesting physical properties like temperature and electromagnetic emission. It could also be possible to discover which inputs or programs are applied to a circuit watching these power movies.

8.3.3 Input Patterns from Real Scenarios

FFT_A, B, C and D are 64-point pipelined FFT implementations that fulfill the Hiperlan/2 and IEEE 802.11a-g standards. The target device for these designs is a Virtex XCV800HQ240-4. The tolerated error for the estimations is specified as 20% with 80% confidence. The activity threshold is 0.25 and the optimization strength is 0.75. Input data are modulated QAM and QPSK. The results are shown in Table 8.17.

	Circuit	Measured	Tg = 0		Opt. Tg	
			Estimated	Error (%)	Estimated	Error (%)
QAM	FFT_A	27.75	33.44	+21	27.41	-1
	FFT_B	27.75	32.54	+17	26.25	-5
	FFT_C	27.63	32.79	+19	26.71	-3
	FFT_D	24.88	33.67	+35	27.70	+11
QPSK	FFT_A	28.00	33.35	+19	27.56	-2
	FFT_B	27.13	32.49	+20	26.12	-4
	FFT_C	27.75	32.72	+18	26.59	-4
	FFT_D	24.88	33.59	+35	27.65	+11

Table 8.17: Total Power estimations for the FFT circuits

In Table 8.17, filtering the short pulses, the error is less than 11% for all the cases. However, when no glitches are filtered, the error can reach 35%, always in excess.

The results in Table 8.17 show a very small difference between these modulations at the average total power level. Fig. 8.33.a and 8.33.b show the power maps for these input data modulated QAM and QPSK respectively with a seemingly small difference for the average power consumption at the CLB level.

Nevertheless, ultimate conclusions must be derived studying the power consumption at the CLB level computing the individual relative differences according to Eq. 8.5.

Average, standard deviation and maximum values for these differences are shown in Table 8.18 for square areas of n x n CLBs.

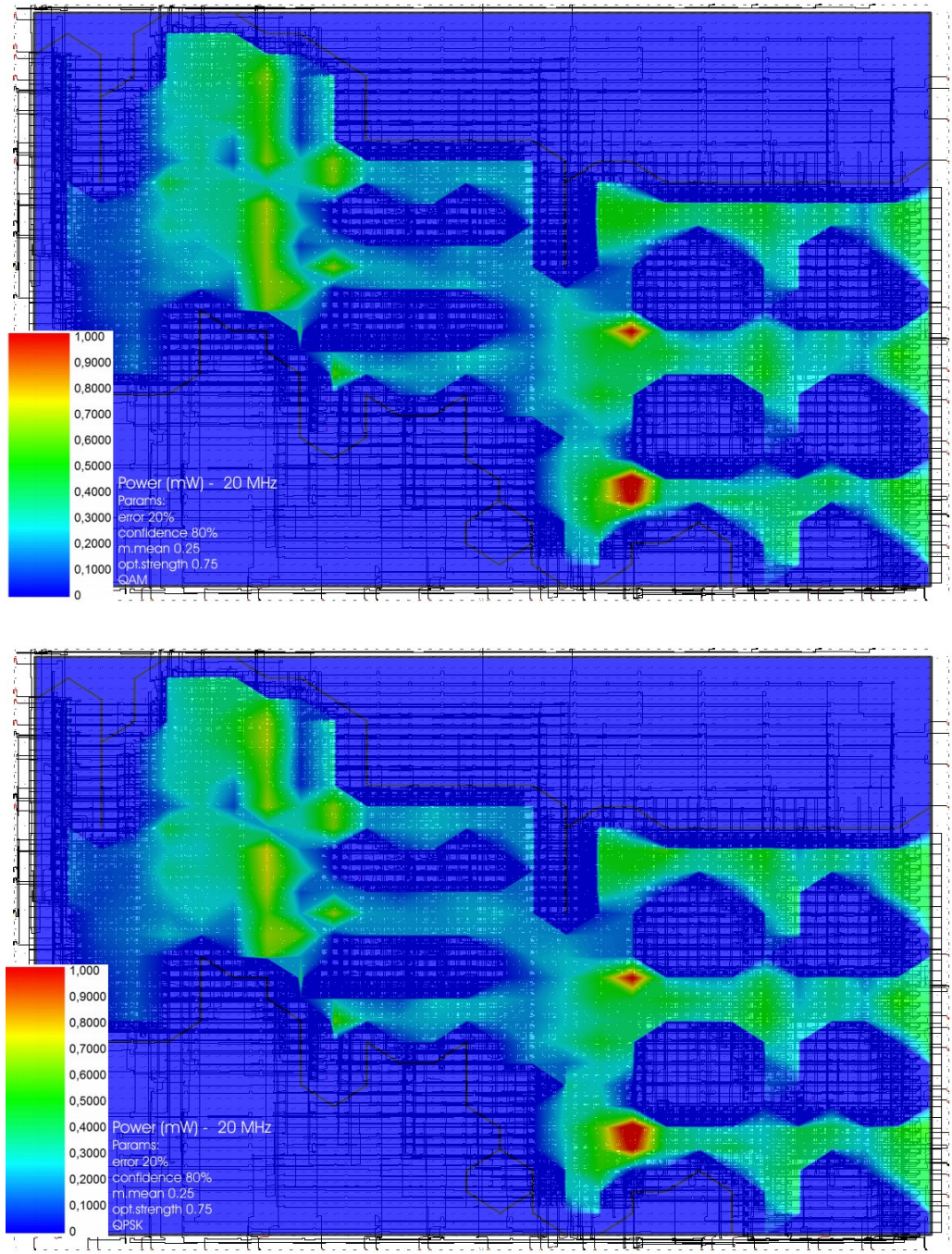


Fig. 8.33: Power Map. Resolution 1 CLBs

Resolution	Mean [%]	Std. Deviation [%]	Maximum [%]
1x1	10.9	13.2	197.3
2x2	7.9	8.7	60.0
4x4	5.4	6.8	37.3

Table 8.18: Relative differences between two simulation runs with different minimum activity thresholds

Table 8.18 shows that there is an important difference between the input patterns for individual node and, in general, low resolution power estimation in the implemented technique. This effect is attenuated as the resolutions decrease and, at the end, it is very low for total power. Fig. 8.34 shows the absolute values of the relative differences at the 1x1 CLB resolution.

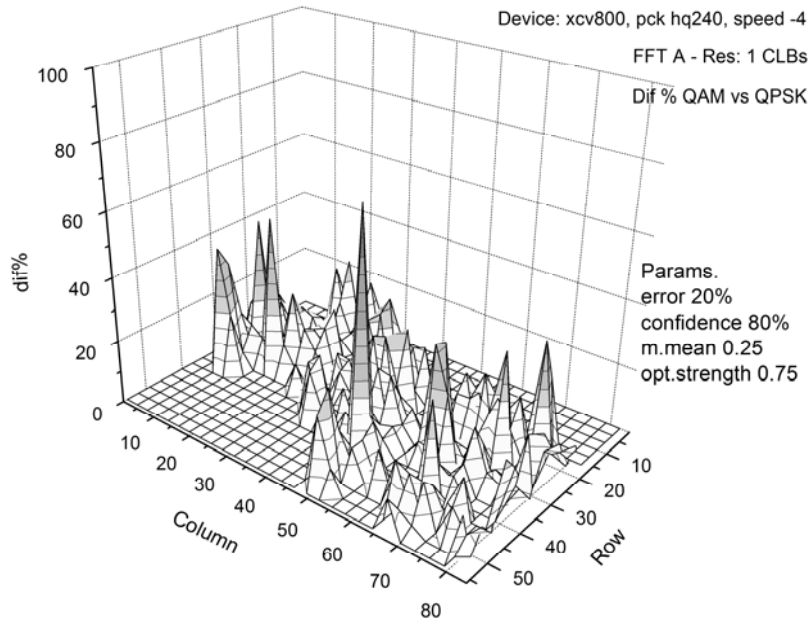


Fig. 8.34: Relative power differences between QAM and QPSK modulations at 1x1 CLB resolution

8.4 Additional Experiments on Virtex-II

Up to now, all the experiments were performed on Virtex and Virtex-E devices. However, it is interesting to check if the tool and methodology is applicable to another Xilinx family such as Virtex-II. The new experiments are performed on a design with ten

16-bit combinational multipliers with registered inputs and outputs. For this test case a complete tool evaluation was run.

8.4.1 Impact of the Input Pattern Definition on Total Power

Both estimated and measured power values, in mW/MHz, are shown in Table 8.19. The design is evaluated with the different input pattern defined in Table 8.12.

The min. glitch duration, T_g, calculated as the optimum value for this device and test case is 900 ps. Note that, filtering these short pulses, the error is less than 6% for all the cases except one with 18.8%. On the other hand filtering glitches shorter than 50 ps, the error can reach more than 350%, always in excess. As in the combinational 32-bit multiplier, it was not possible the estimation without any glitch filtering because the extremely high activity reported by the simulator produces overflows in the program that parses the VCD files.

Case	Measured	Tg = 50		Opt. Tg	
		Estimated	Error (%)	Estimated	Error (%)
1	10.97	43.92	300.7	11.13	1.6
2	10.14	42,30	317.3	10.14	0.0
3	9.81	39.90	306.7	10.14	3.3
4	9.31	36.57	292.7	9.17	-1.5
5	11.56	41.22	256.5	10.92	-5.6
6	15.01	62.09	313.6	14.83	-1.2
7	3.25	14.77	354.2	3.86	18.8
8	10.96	45.26	312.8	10.95	-0.1
9	10.81	43.88	305.9	10.80	-0.1

Table 8.19: Total Power Estimation for the 10MULT16-C Design

The differences in total power consumption for every input pattern definition are shown in Fig. 8.35.

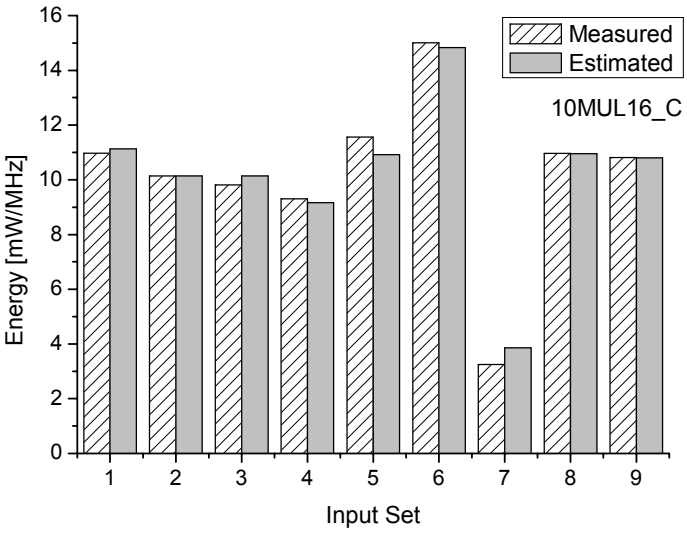


Fig. 8.35: Total Power for the different Input Patterns (See Table 8.11) for 10MULT16-C.

In the same way as shown in Fig. 8.30, Fig. 8.35 indicates that an as real as possible input set specification is essential in order to obtain meaningful results.

8.4.2 Dynamic Power Estimation for Individual Nodes

In this section it is shown if the results, at individual node level, fit the user specified accuracy as in section 8.2.2. Fig 8.36 shows relative error distributions for different levels of accuracy. This accuracy is specified in the upper right hand corner of each histogram and it is achieved in all the cases shown in this figure.

As the power consumption for the individual nodes can not be physically measured, the comparisons were made against the results obtained from a long simulation run with a sample size of 65158 clock cycles with the following parameters: 95% of confidence that error is less than 5%, the threshold for the min. activity is 0.05, and the optimization strength equal to zero. In this set of test cases the threshold for the activity mean, that divides regular and low activity nodes, is 0.25, the optimization strength is 0.50, and the input patterns are generated independently.

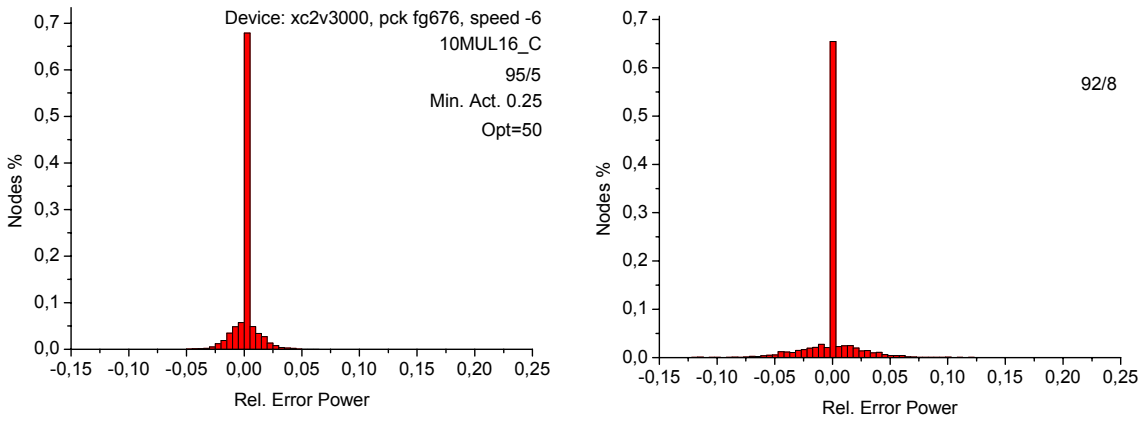


Fig. 8.36: Individual node power: relative error distributions for 10MUL16_C

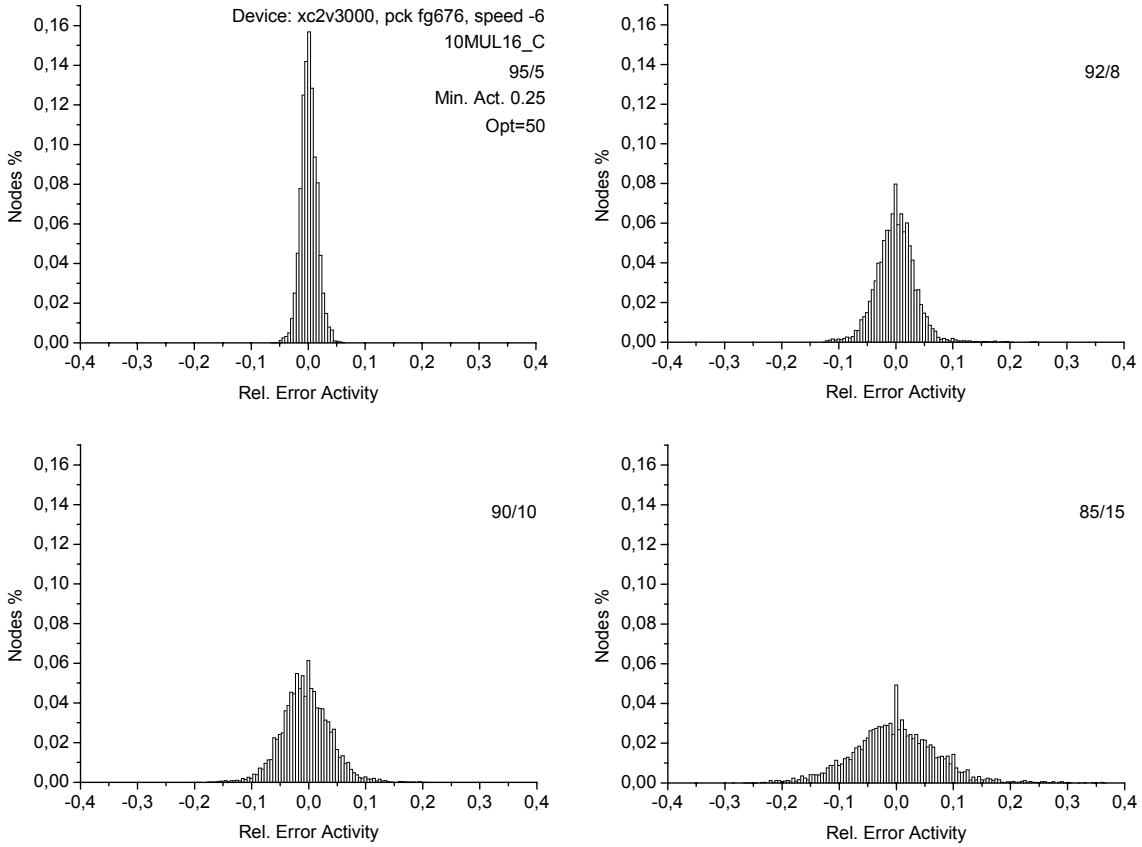


Fig. 8.37: Individual node activity: relative error distributions for 10MUL16_C

As there are a number of nodes where the reported capacitance is zero, a noticeable high column in Fig. 36 is observed. In this case, it is more interesting to see

what happens to the activities. With the same parameters, Fig. 8.37 shows activity relative error distributions for different levels of accuracy.

8.4.3 Accuracy vs. Execution Time Tradeoff

As in section 8.2.3, how the execution time depends on the required accuracy is studied. Fig. 8.38 shows a graphical representation of the accuracy vs. execution time tradeoff where all the variables are present and the estimation system is characterized. The *x*-axis represents the accuracy, where a x_i values correspond to an x_i% error with 100-x_i% confidence. This experiment also confirms the robustness of the technique, allowing a tunable accuracy. Dashed lines represent the cases without optimization and the solid lines correspond to simulation runs with optimization strength 0.5.

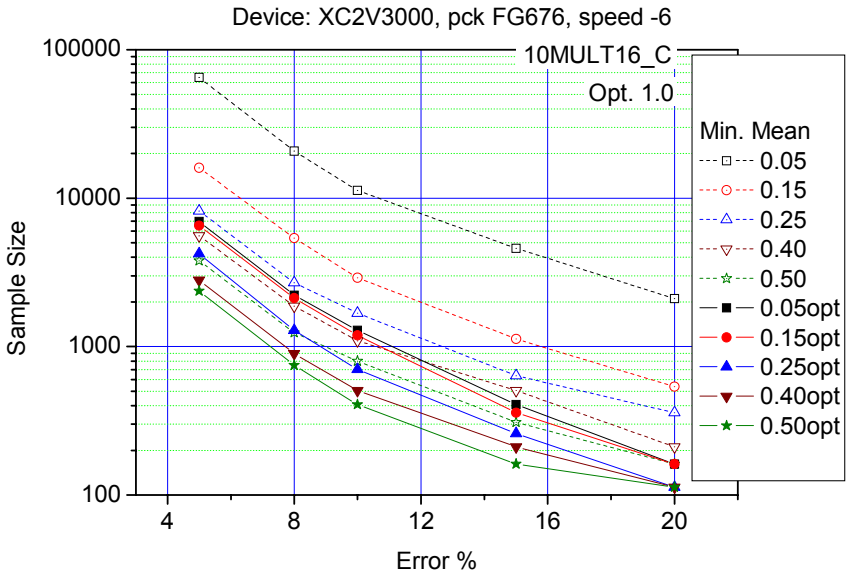


Fig. 8.38: Accuracy/execution-time tradeoff for 10MULT16_C

In order to give more information about the results in Fig. 8.38, Table 8.20 shows the execution time savings with respect to the non optimized case.

As pointed out previously, this optimization without any loss of accuracy is possible because the effective accuracy is higher than the specified one and the nodes do not converge linearly. Fig. 8.39 shows how 98% of the nodes representing 99% of the power met the stopping criterion halfway through the simulation run. The estimation

time and the nodes percent are presented in relative terms to make easier further comparisons.

	Min. Act.				
Error	**0,05**	**0,15**	**0,25**	**0,4**	**0,5**
5	0,84	0,56	0,31	0,46	0,27
8	0,88	0,58	0,43	0,47	0,32
10	0,87	0,52	0,50	0,45	0,37
15	0,88	0,60	0,51	0,58	0,48
20	0,88	0,61	0,55	0,46	0,30

Table 8.20: Execution time savings for 10MULT_C. Optimization strength 0.50

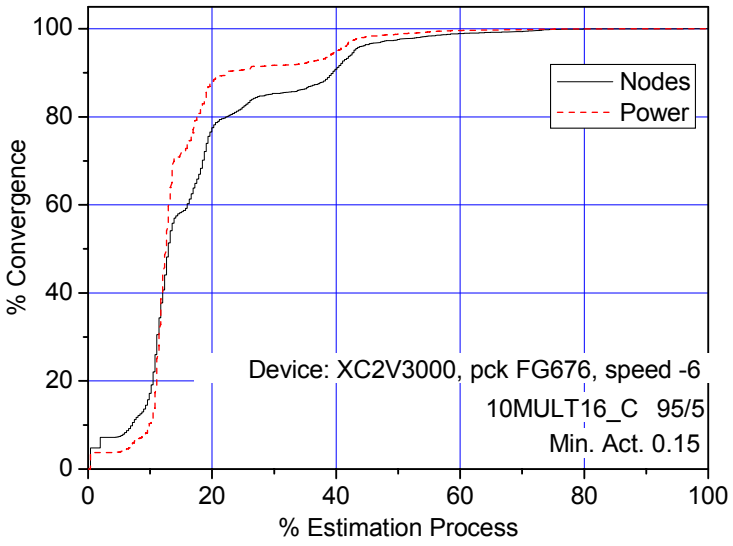


Fig. 8.39: Convergence time for 10MULT16_C

Table 8.21 is similar to Table 8.20 but the optimization strength is 1.00.

To complete the results shown in Fig. 8.38, Fig. 8.40 adds a third axis to show the behavior of the optimization strength parameter.

	Min. Act.				
Error	**0,05**	**0,15**	**0,25**	**0,4**	**0,5**
5	0,89	0,59	0,48	0,50	0,38
8	0,89	0,61	0,52	0,52	0,40
10	0,89	0,59	0,58	0,54	0,49
15	0,91	0,68	0,59	0,58	0,48
20	0,92	0,70	0,68	0,46	0,30

Table 8.21: Execution time savings for 10MULT_C. Optimization strength 1.00

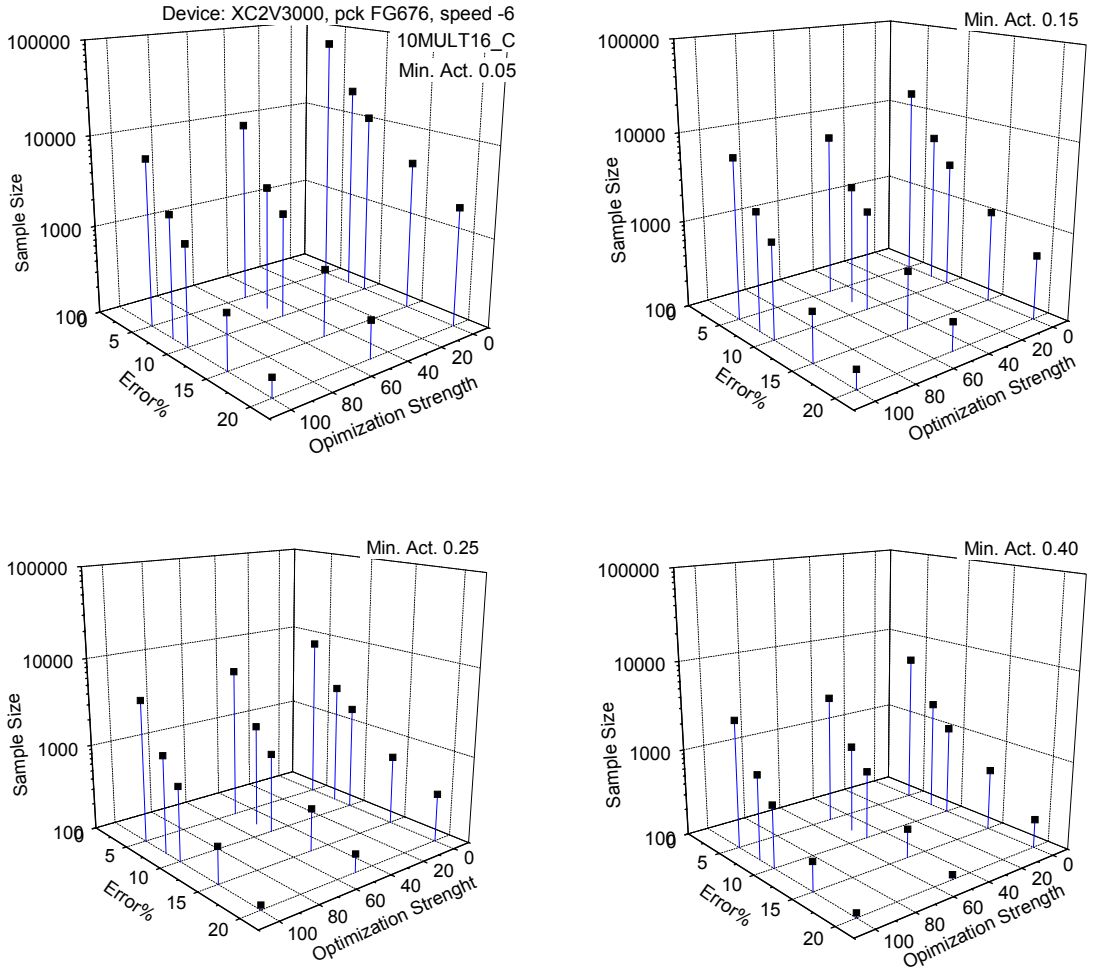


Fig. 8.40: Accuracy/execution time tradeoff for 10MUL16_C for different optimization strength values

8.4.4 Tool's Evaluation

Table 8.23 and Fig. 8.41 show the run times estimating the power consumption for the 10MULT16_C test circuits with optimization strength 1.00. The error is 10%, 90% confidence, and the threshold for the low activity nodes is 0.25 transitions per clock cycle. Table 8.22 summarizes the execution times by sub-system

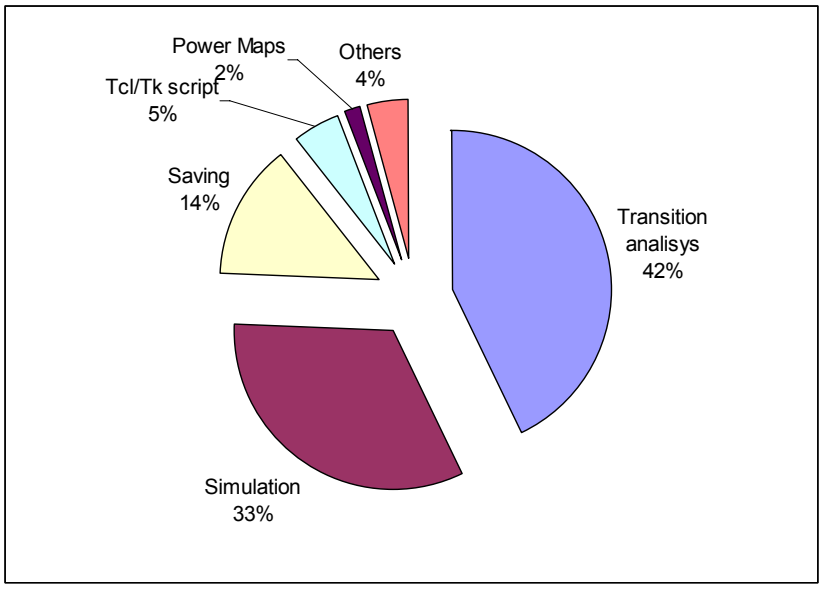


Fig. 8.41: A-DyP relative execution times for 10MULT16_C

Task	**Exec. Time [min]**
Total Execution	42
Initialization	0
Activity Estimation	38
Power Computation	2
Tcl/Tk script	2

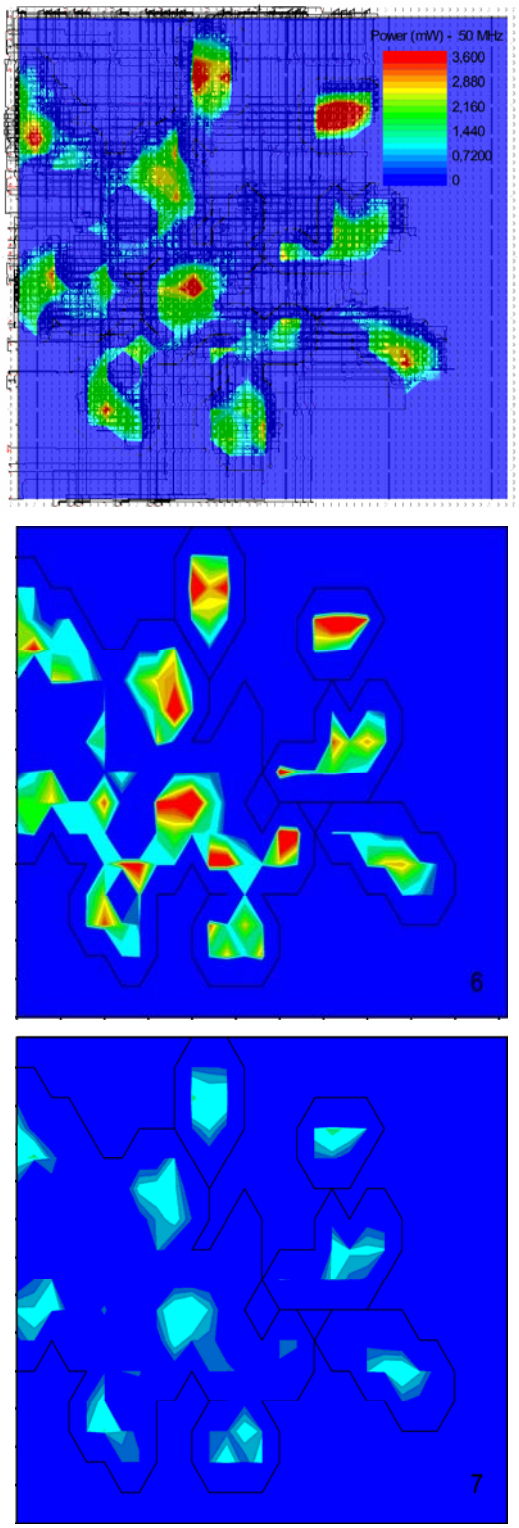
Table 8.22: A-DyP total execution time for 10MULT16_C

	Task	Exec. Time [secs]
Activity Estimation	Input vector generation	8
	Simulation	817
	Saving	347
	Transition analysis	1063
	Statistics computation	13
	Stopping criteria evaluation	12
Power Computation	VHDL parsing	11
	XDL parsing	1
	XML generation	0
	XPower execution	16
	PWA parsing	1
	Report writing	25
	Maps generation	40

Table 8.23: A-DyP absolute execution time for 10MULT16_C

8.4.5 Power Maps

Each input pattern definition can produce hot spots or hot areas in different places in the FPGA and with different amplitudes. Fig. 8.42.a, b and c show power maps for the input pattern definition number 1, 6, and 7 respectively (See Table 8.12) for the 10MUL16_C design. The layout is in the Fig.8.42.a's background. For all the figures the same colors represent the same power values. Furthermore, red represents 3.6 mW or more. The parameters for the simulation runs were 85% confidence that error is less than 15%, T_g is 900 ps, the threshold for the low activity nodes is 0.25 transitions per clock cycle, and the optimization strength is 0.50.



(a)

(b)

(c)

Fig. 8.42: Power Maps for 10MUL16_C. Resolution 4x4 Slices

Fig 8.42.b clearly represents the highest power consumption case with most hot spots. On the other hand, Fig. 8.42.c is the less active case where the 4 x 4 most active slices square squares consume about 1.5 mW. As in Fig. 8.32, it can be observed that given a power or activity map, it could be inferred which input patterns are applied to the primary inputs. Conversely, applying specific input pattern sets, it could be possible to generate specific power and activity maps.

8.5 Conclusions

A number of experiments were performed on three Xilinx families (Virtex, Virtex-E and Virtex-II) in order to show that the proposed statistical technique works with these technologies. The FPGAs have the same basic structure, i.e. the programmable block array and the network to interconnect them. However they correspond to different VLSI technological generations.

Several sources of error were detected in the power estimations within the current Sw/Hw framework: pulses shorter than the physically possible ones, simulator inaccuracies to propagate glitches, lack of information about the error in the reported capacitances, and probably others. As the observed error is always in excess, a glitch filtering policy was proposed as a calibration resource that leads to accurate results.

An improvement to the classical Monte Carlo power estimation method for individual nodes has been presented. Although the method is implemented and evaluated within the particular Xilinx ISE design flow and devices, there are no restrictions to apply the technique within other FPGA design environments or even general CMOS design flows. The problem with the classical statistical estimation method is the execution time. Current big designs could require unacceptable run times when the user specifies medium to high accuracy requirements. The proposed A-B technique takes up shorter execution times enabling its practical use within existing design flows. Moreover, the proposed technique is simple and very easy to implement. It has been shown that the optimization is done without loss of accuracy at the individual nodes level. This is because it makes use of the extra accuracy generated running the classical approach that is effectively higher than the one specified by the user. To quantify and measure this extra precision, a definition of effective accuracy was proposed.

The experiments confirm the robustness of the technique, allowing a tunable accuracy. According to the precision required at each moment in the design process, appropriate values can be set for both the minimum mean activity and the error-confidence pair. The execution time of the tool monotonically increases with the required precision. It has also been verified that the actual relative error for individual nodes is bounded by the one specified by the user.

An as real as possible input set specification was verified as being essential in order to obtain meaningful results. Although it is a known issue in the power estimation world, here it was confirmed over the FPGA technology with for the QDDFS-CORDIC case, where the inputs are fixed during each specific evaluation.

Finally, an experiment was performed to verify the energy of computation principles resumed in section 2.3.1. The different implementations of the FIRDA circuit are an appropriate test case because they perform the same computation. It is observed that total power is higher for the serial version. However, this is due to the increase in clock power. Dynamic power excluding the clock is almost the same for all the implementations, from the combinational case to the serial one.

References

[And04] Anderson, J.H.; Najm, F.N., “Power estimation techniques for FPGAs”, IEEE Trans. on VLSI Systems, Volume 12, Issue 10, pp. 1015-1027, Oct. 2004

[May01] Uwe Meyer-Baese, “Digital signal processing with field programmable gate arrays”, Springer, 2001.

[Bae02] C. Baena, J. Juan-Chico, M.J. Bellido, P. Ruiz, C.J. Jimenez, and M. Valencia Barrero, “Measurement of the Switching Activity of CMOS Digital Circuits At the Gate Level”, Lecture Notes in Computer Science. Vol. 2451. 2002, pp. 353-362.

[Fey96] Richard P. Feynman, “Feynman Lectures on Computation”, Perseus Books, 1996.

Chapter 9.

"Lo último que uno sabe es por donde empezar" from Pensées (1660) by Blaise Pascal (1623-1662)

9 Conclusions and Future Works

In this chapter the main conclusions and contributions of this thesis are summed up. Also the publications generated from this work and the ideas that can be developed in the future are listed below. Finally, some recommendations about how to build a power estimation system and how to estimate power along the design process are proposed.

Power estimation within a real scenario was found to be a very challenging task. As the developed software is included within FPGA design flows, reverse engineering is the only tool available to solve some practical problems. This is due to the lack of support to third-party integration in the commercial FPGA world, particularly to the issues related to power estimation.

9.1 Main Contributions of this Thesis

A lot of techniques for power estimation have been published but almost all do not go beyond simulation. On the other hand there are a relatively small number of papers about power estimation techniques specifically developed, adapted or tested on FPGAs. In this way, the main contribution of this thesis is the development of a power estimation methodology for FPGA and a power estimation tool which was calibrated against physical measurements.

A platform for power estimation on FPGA was developed. Over this platform a statistical power estimation tool was built. A number of measurements were performed

over three Xilinx families (Virtex, Virtex-E and Virtex-II) in order to show that the statistical technique works with these technologies covering more than 10 years of PLD evolution. There are also experiments on the Xilinx 4000 family but as the capacitances were not provided for them; those results are not presented in this thesis. All these FPGAs have the same basic structure, i.e. the programmable block array and the network to interconnect them. However they correspond to different VLSI technology generations and there are additional resources within the programmable blocks in the newer families. The conclusions presented in this chapter are based on thousands of hours of experimentation on the circuits and development boards described in Chapter 7.

9.1.1 The Power Platform Framework and A-DyP

The current version of A-DyP, the statistical power estimation tool, is not an end in itself, but a foundation upon which other power-aware tools can be built. It is hoped this tool is the first iteration necessary for the development of a more general power estimation framework called Power Platform in this thesis. For this reason, the programs, data structures, formats and software technologies were designed or selected according to this goal. On the other hand, all the programs and scripts in A-DyP access data in the Power Database and the configuration files which include application and project parameters. The appropriate way to work with this information is through a software layer independent of the database engine and file formats.

A power estimation tool must be integrated within a design flow. In this way, standard formats must be selected to enable interoperability though different software vendors and versions.

Upon the Power Platform, a statistical-based power estimation tool for FPGA devices has been developed with the following features:

- A Tcl/Tk script implements the high-level instructions of the estimation algorithm and integrates the programs that deal with specific functionality within the system. One such script enables the reuse of the programs and is a fundamental piece in the Power Platform framework.
- Any standard simulator that deals with VCD activity files can be used in the inner loop of the Monte-Carlo program making the technique easy to

implement. In order to manage post PAR designs the simulator must also support VITAL and SDF files.

- If the accuracy selected is not too high, the execution time is reasonable, even for current big designs.
- A simple input specification can be defined.
- Temporal, and spatial (at the internal nodes level) correlations are considered.
- The most accurate model available can be used, i.e. glitches can be taken into account. All the presented experiments are performed over post PAR designs, but there is no restriction to apply A-DyP to post synthesis, map, or post place designs.

The experiments confirm the robustness of the technique, allowing a tunable accuracy. According to the precision required at each moment in the design process, appropriate values can be set for both the minimum mean activity threshold and the error-confidence pair. The execution time of the tool monotonically grows with the required precision. It has also been verified that the actual relative error for individual nodes is bounded by the one specified by the user.

Finally, the importance of properly defined input pattern characteristics is pointed out. The use of this tool with a default or arbitrary input pattern can result in an activity figure with unpredictable error. It was verified that it is essential to specify an as real as possible input set in order to obtain meaningful results. Although it is a known issue in the power estimation world, here it was confirmed with the FPGA technology even for the QDDFS-CORDIC case, where the inputs are fixed during each specific evaluation.

9.1.2 Short-pulse Filtering as a Calibration Resource

There are several sources of error in the power estimations within the current Sw/Hw environment: the simulator reports pulses shorter than the physically possible ones and have inaccuracies propagating glitches; there is a lack of information about the error in the reported capacitances; and probably others. As the observed error is always in excess, a glitch filtering policy is proposed to calibrate the tool. It is shown that this strategy led to accurate results.

Although it is a solution within the current environment, these sources of error should be eliminated improving the simulators with better models.

9.1.3 A-B Nodes Classification

A new improvement for the classical Monte Carlo power estimation method for individual nodes has been presented. Although the method is implemented and evaluated within the particular Xilinx ISE design flow and devices, there are no restrictions to apply the technique within other FPGA design environments or even general CMOS design flows.

The problem with the classical statistical estimation method is the execution time. Current big designs could require unacceptable run times when the user specifies medium to high accuracies. The proposed A-B technique has reasonable execution times enabling its practical use within existing design flows. Moreover, the proposed technique is simple and very easy to implement.

It has been shown that the optimization is done without loss of accuracy at the individual nodes level. This is because the A-B method makes use of the extra accuracy generated running the classical approach that is effectively higher than that specified by the user. To quantify and measure this extra precision, a definition of effective accuracy is proposed.

9.1.4 Energy of the Computation

An experiment was performed to measure the energy of computing according to the thermodynamic principles resumed in section 2.3.1. The different implementations of the FIRDA circuit are an appropriate test case because they perform the same computing. It is observed that total power is higher for the serial version. However, this is due to the increase in clock power. Dynamic power excluding the clock is almost the same for all the implementations, from the combinational to the serial case.

9.2 Reverse Engineering

As mentioned in section 9.3.1, the lack of integration of current Xilinx tools with third-party programs in the power estimation area leads to reverse engineering tasks. These tasks were particularly time consuming in this thesis and the hardest one was that

related to capacitances retrieval. In section 4.4 it is explained that the different names found in the different programs of the design flow to identify the same circuit nodes prevent knowing to which node correspond some values in the capacitances report. Even parsing the VHDL simulation model, where the also-known-as option writes within comments the alternative names, the problem is not completely solved due to inconsistencies and remaining bugs. On the other hand, there is no documented procedure to retrieve the capacitances with the appropriate precision in fF since ISE 6.

Other reverse engineering tasks are explained in Chapter 6. E.g. complete documentation in order to parse XDL files is not found.

9.3 Publications

The most important papers related to the topics covered in this thesis are, in chronological order:

- E. Todorovich and E. Boemo, “A-B Nodes Classification for Power Estimation”, 16th International Conference on Field Programmable Logic and Applications (FPL 2006), Madrid, Spain, August 28-30, 2006.
- E. Todorovich, F. Angarita, E. Boemo, “Statistical Power Estimation for Fpga’s”, 15th International Conference on Field Programmable Logic and Applications (FPL 2005), pp. 515-518, ISBN: 0-7803-9362-7, August 24-26, Tampere, Finland.
- E. Todorovich, E. Boemo, F. Cardells, J. Valls, “Power Analysis and Estimation Tool integrated with XPower”, Twelfth ACM International Symposium on Field-Programmable Gate Arrays, FPGA 2004, Monterey, California, USA, February 22-24, 2004. ISBN 1-58113-829-6.
- E. Todorovich, M. Gilabert, G. Sutter, S. Lopez-Buedo, and E. Boemo, “A Tool for Activity Estimation in FPGAs”, Lecture Notes in Computer Science, Vol. 2438, pp. 340-349. Springer-Verlag, Berlin Heidelberg 2002.
- G. Sutter, E. Todorovich, S. Lopez-Buedo, E. Boemo, “Low-Power FSMs in FPGA: Encoding Alternatives”, Lecture Notes in Computer Science, Vol. 2451, pp. 363-370. Springer-Verlag, Berlin Heidelberg 2002.

- G. Sutter, E. Todorovich, S. Lopez-Buedo, and E. Boemo, "FSM Decomposition for Low Power in FPGA", Lecture Notes in Computer Science, Vol. 2438, pp. 350-359. Springer-Verlag, Berlin Heidelberg 2002.

Other partial results were also published in the following conferences about FPGA technology and power consumption:

- F. Angarita, J. Marin-Roig, E. Todorovich, E. Boemo, “Relación área-potencia en la implementación con aritmética distribuida de un Filtro FIR en FPGA”, JCRA 2005, pp. 59-63, ISBN: 84-9732-439-0, Granada, 13-16 Septiembre 2005.
- Todorovich E., A. Holderbeke, N. Acosta, E. Boemo, “Estimación de Consumo de Potencia en FPGA a través de un Servicio Web”, Jornadas de Computación Reconfigurable y Aplicaciones, JCRA 2004, Barcelona, España, 13-15 de Septiembre de 2004.
- Sutter G., Todorovich E. and E. Boemo, “Design of Power Aware FPGA-based Systems”, Jornadas de Computación Reconfigurable y Aplicaciones, JCRA 2004, Barcelona, España, 13-15 de Septiembre de 2004.
- Todorovich E., “Tcl/Tk para Herramientas EDA”, JCRA 2003, Madrid, España, 10-12 de octubre de 2003. Pág 529-538.
- Todorovich E., Sutter G., Boemo E., “Estimación de Actividad para FPGA Basada en una Técnica Estadística”, JCRA 2003, Madrid, España, 10-12 de octubre de 2003. Pág 217-224.
- Sutter G., López-Buedo S., Todorovich E., Boemo E., “Logic Depth, Power, and Pipeline Granularity: Some Examples on FPGAs.”, JCRA 2003, Madrid, España, 10-12 de octubre de 2003. Pág 201-208
- E. Todorovich y N. Acosta, "Estimación de Capacidad en FPGAs Comerciales", VIII Congreso Argentino de Ciencias de la Computación, CACIC 2002, Universidad de Buenos Aires, 15 al 18 de octubre de 2002.
- G. Sutter, E. Todorovich y E. Boemo, “Metodología para la reducción de consumo en circuitos integrados reprogramables”, III Workshop de

Investigadores en Ciencias de la Computación, San Luis, Argentina, 22-24 de Mayo 2001, pp. 14-17.

- E. Todorovich, G. Sutter, N. Acosta, S. López-Buedo y E. Boemo "Relación entre Velocidad y Consumo en FPGAs", VII WorkShop de IBERCHIP, Montevideo, Uruguay, 21-23 de Marzo, 2001.
- G. Sutter, E. Todorovich, S. López-Buedo y E. Boemo "Propiedad Conmutativa y Diseño de Bajo Consumo: Algunos Ejemplos en FPGAs", VII WorkShop de IBERCHIP, Montevideo, Uruguay, 21-23 de Marzo, 2001.
- E. Todorovich, G. Sutter, N. Acosta and E. Boemo, “End-user low-power alternatives at topological and physical levels. Some examples on FPGAs”, XV Conference on Design of Circuits and Integrated Systems (DCIS2000), Le Corum, Montpellier, France, November 21-24, 2000.
- E. Boemo, S. López-Buedo, E. Todorovich and N. Acosta, “Profundidad de Lógica y Determinismo de las Herramientas de Emplazamiento y Rutado. Algunos Experimentos en FPGAs”, VI Workshop IBERCHIP, Sao Paulo, Brasil, 16-18 de marzo del 2000, pp. 280-285.
- N. Acosta, E. Todorovich, C. Collado y K. Larsen, “Multiplicadores paralelos: estado del arte y análisis de su materialización en FPGA”, VI Workshop IBERCHIP, Sao Paulo, Brasil, 16-18 de marzo del 2000, pp 158-168.
- E. Boemo, S. López-Buedo, N. Acosta and E. Todorovich, “Local vs. global interconnections in pipelined arrays: an example of interaction between architecture and technology”, DCIS’99, in Palma de Mallorca (Baleares), Spain, November 16-19, 1999, pp. 158-168.

9.4 Future Tasks

Once the power estimation problems, together with the techniques proposed in the specialized literature at the different levels of abstraction are carefully studied; and the current software and hardware technologies in the area are exercised by means of the practice and experimentation for a long time, then it is possible to think about a number of related future tasks:

Testing A-DyP on other Xilinx FPGA technologies. At the present time these technologies are, at least, Virtex-II pro, Virtex-4, and Spartan-2 and 3. Each technology and software version has their own features that must be considered in order to gauge power. Several programs within the Power Computation Sub-system should be extended, the whole system should be revised and the Power Platform framework could be improved with every newly analyzed FPGA family.

Porting A-DyP to Altera FPGAs. This is another interesting and challenging task. The Activity Estimation Sub-system can be mostly reused but the Power Computation Sub-system must be rewritten. It is a good opportunity to improve the Power Platform framework.

Other statistics-based power estimation methods. There are several techniques based on statistics developed for specific power estimation problems that can be studied and implemented. These problems include total power estimation for big sequential circuits, advanced sampling techniques for hierarchical designs, etc.

Peak or maximum power estimation. This is more than a single problem but a family of problems related with different definitions of maximum power in CMOS circuits: e.g. the peak single-cycle power is the maximum total power consumed during one clock cycle; the peak n-cycle power is the maximum average power of a contiguous sequence of n clock cycles; the peak sustainable power is the maximum average power that can be sustained indefinitely. These tasks could reuse and improve the Power Platform framework.

Static Power optimization. Due to technology scaling, static power is an increasingly (40% of total power at the 70 nm technology node) dominant component in current ICs. In this way, static power optimization is an interesting subject for research in FPGAs [And06].

Early capacitance estimation. It is interesting to estimate the design capacitances before the place and route process [And06]. Before layout, capacitances depend on parameters like the node fanout. In general, the relationship can be mathematically stated by an n-degree polynomial function. The coefficients of this function calibrate the function for different FPGA families or devices. These values can be obtained by an evolutionary algorithm.

EDA tools for Low Power Design. Beyond power estimation, a complete power aware design flow includes synthesis, technology mapping, placement, and routing tools for low power design. A clear opportunity exists for research on all these EDA tools.

Temperature Estimation. It could be useful to develop a tool that exploits the relationship between power dissipation and temperature in order to extend the Power Platform framework to support temperature estimation for FPGAs.

9.4.1 High-level Power Estimation

The macromodelling approach is considered the solution for the RTL power estimation problem [Bru05]. Nevertheless, there are several interesting situations and related problems not considered yet, that can be studied, i.e.:

Hard core power estimation. This thesis analyzes the dynamic power consumption in the programmable fabric of FPGAs, i.e. the configurable logic blocks and routing resources that connects them. This is the most important part in Low Power Design because of its power inefficiency. On the other hand, hard cores in FPGAs are expected to perform as well as their equivalent in ASIC. Nevertheless, designers are interested in total power. Consequently, the hard-core power consumption on the FPGA must also be considered.

High-level power estimation based on neural networks. Gate-level power estimation techniques are both useful and accurate. However, the required run time can prevent calling them within optimization loops. Therefore, a power aware design flow needs complementary high-level and very fast power estimation tools. Given the number of slices, minimum period, and other parameters it could be interesting to use neural networks to estimate total average power. However, as the power consumption also depends on the activity, this additional input could be obtained from a very fast run with a low accuracy specification and a zero delay simulation model.

IP power characterization. Over the last few years, functional macro verification has emerged as an essential skill in the hardware engineer's education and practice. IP power and activity characterization can be included within these standard practices. The power can be reported for several well defined scenarios when a specific technology and implementation tools are selected. Even when the core is available for

a number of current and future technologic targets, the activity characteristics should be specified with the core documentation. Furthermore, assertions about average and maximum activity should be specified in the future hardware verification plans.

Power Estimation for embedded processors. Nowadays a number of embedded soft and hard cores for microprocessors are available for designers. This increases the research interest in areas such as low power microprocessor soft cores, low power buses, high-level power estimation for this type of circuits, power-aware compilers and operating systems, and in general new techniques to reduce the computational complexity, i.e. activity of the implementations.

Impact of RTL synthesis. As pointed out in [Bru05], the RTL power estimation of the fine-grain primitives is difficult due to the impact of the synthesis. In this way it is interesting to measure the noise in power consumption of these primitives applying different synthesis tools and options.

9.5 How to Estimate Power Consumption

Power consumption can be estimated at any point in the design cycle. You can use spreadsheets provided by the FPGA vendors to estimate the power consumption if you have not begun your design, or if your design is not complete. To use these spreadsheets, you need to enter device resources, operating frequency, toggle rates etc. As you do not have an existing design, then you will need to estimate the number of device resources your design could use, the specified operating frequency, and a safe average toggle rate (i.e. a high enough value).

After your have the first versions of your design, you can use A-DyP to estimate the power consumption. In this case, very short runs with low accuracy, high activity threshold, and high optimization strength are recommended. At this point in the design cycle it is interesting to know the power consumption range. You can identify hot spots and which parts of the design consumes the most power leading the optimization efforts.

When you have stable and verified versions within the specified power budget, you can run highly accurate simulations with as real as possible input patterns in order to

obtain precise power values, say within 10% error in relation to potential physical measurements of the circuit on the PCB.

9.6 How to Build a Power Estimator

One of the most important decisions software (and currently, hardware) designers must face is whether they will integrate their contributions with open source tools or within commercial IDEs. Power estimation is a typical, but not the only case where this decision has fundamental and practical consequences.

In this thesis the second option is the selected one. This enables us to work with real devices. I.e. measure their current consumption with an ammeter and compare these physical amounts with software estimations. Experiments with actual devices give a fundamental advantage over approaches working with theoretical models that are compared with other models supposedly more accurate.

However, working with proprietary software has also significant drawbacks. This is particularly hard when the integration with third-party tools is not considered or foreseen in the design of that proprietary software. In this case the lack of information and support could force the application of challenging reverse engineering techniques. To avoid this time consuming task, FPGA vendors could provide open, even more, standard interfaces to retrieve and provide the necessary information accurately in the different parts of the design flow.

A general objective when writing a power-aware tool is software quality. In order to reuse all or parts of the developed software, eventually port it, and interoperate with other tools in an EDA design flow, standard formats must be used as far as possible.

Another goal is to produce reusable software components. In this way, a power platform has been proposed with a basic middleware that connects the database with the application programs. Several programs can be used as-is or adapted for other power estimation tools e.g. the VCD parser. There are other tools for which the reuse is limited within the Xilinx scope e.g. the XDL parser.

References

[And04] Anderson, J.H.; Najm, F.N., “Power estimation techniques for FPGAs”, IEEE

Trans. on VLSI Systems, Volume 12, Issue 10, pp. 1015-1027, Oct. 2004

[And06] Anderson, J.H.; Najm, F.N., “Active Leakage Power Optimization for FPGAs”, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 25, No. 3, March 2006.

[Bru05] M. Bruno, A. Macii, and M. Poncino, “RTL power estimation in a HDL-based design flow”, IEE Proc.-Comput. Digit. Tech., Vol. 152, No. 6, November 2005.

Complete References List

Each chapter has their own list of references; however, it is useful to show the complete list in order to make easier further bibliography searches.

[Aho86] Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, Compilers: Principles, Techniques, and Tools, Addison-Wesley, 1986.

[Alt04] Altera Corp. “Power Calculator User Guide”, March 2004, available at http://www.altera.com/support/devices/estimator/pow-powerplay.html

[Alt05] Altera Corp., “Stratix-II Device Handbook, 2005”, available at http://www.altera.com

[And04] Anderson, J.H.; Najm, F.N., “Power estimation techniques for FPGAs”, IEEE Trans. on VLSI Systems, Volume 12, Issue 10, pp. 1015-1027, Oct. 2004

[And06] Anderson, J.H.; Najm, F.N., “Active Leakage Power Optimization for FPGAs”, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 25, No. 3, March 2006.

[Ang01] J. Angel, “Emerging Technology: Energy Consumption and the New Economy”, Network Magazine, January 5, 2001.

[Ang03] Angarita F.E., Canet M.J., Valls J., _ardéi F: Implementación de un Core-IP “Filtro FIR Basado en Aritmética Distribuida”, III Jornadas de Computación Reconfigurable y sus Aplicaciones – JCRA 2003. Madrid, 2003.

[APX03] Xilinx Inc. “Virtex II Prototype Platforms User Guide”, UG015 (v1.1) Jan 2003, available at http://www.xilinx.com

[APX99] Xilinx Inc. “Xilinx Prototype Platforms User Guide for Virtex and Virtex-E Series FPGAs”, Data Sheet DS020 (v1.1) Dec 1999, available at http://www.xilinx.com

[Bae02] C. Baena, J. Juan-Chico, M.J. Bellido, P. Ruiz, C.J. Jimenez, and M. Valencia Barrero, “Measurement of the Switching Activity of CMOS Digital Circuits At the Gate Level”, Lecture Notes in Computer Science. Vol. 2451. 2002, pp. 353-362.

[Bar05] Luiz André Barroso, “The Price of performance”, ACM Queue, pp. 49-53, September 2005.

[Ben03] Charles H. Bennett, “Notes on Landauer’s Principle, Reversible Computation and Maxwell’s Demon”, Studies in History and Philosophy of Modern Physics, v. 34, pp. 501-510, 2003.

[Ben82] C.H. Bennett, “The Thermodynamics of Computation – a Review” Internat. J. Theoret. Phys. 21, pp. 905-940 (1982).

[Ben94] L. Benini, M. Favalli, and B. Ricco, “Analysis of hazard contribution to power dissipation in CMOS IC’s”. In Proceedings of the 1994 International Workshop

on Low Power Design, pp 27-32, April 1994.

[Bet99] Vaughn Betz and Jonathan Rose, “Circuit Design, Transistor Sizing and Wire Layout of FPGA Interconnect”, IEEE Custom Integrated Circuits Conference, 1999.

[Boe95] Boemo, E., Gonzalez de Rivera, G., Lopez-Buedo, S., Meneses, J., “Some Notes on Power Management on FPGAs”, LNCS, No. 975, Springer-Verlag, Berlin (1995) 149-157.

[Bru05] M. Bruno, A. Macii, and M. Poncino, “RTL power estimation in a HDL-based design flow”, IEE Proc.-Comput. Digit. Tech., Vol. 152, No. 6, November 2005.

[Büh00] M. Bühler, M. Papesch and U.G. Baitinger, “Accurate and Approximate Methods for Speeding up Signal Activity Estimation on Gate Level”, PATMOS 2000, pp. 179-188.

[Bur93] R. Burch, F. N. Najm, P. Yang, and T. Trick. “A Monte Carlo approach for power estimation” IEEE Transactions on VLSI Systems, 1(1):63–71, March 1993.

[Buy05] K.M. Büyükşahin and F.N. Najm, “Early Power Estimation for VLSI Circuits”, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, 1(7):1076–1088, July 2005.

[Car03] F. Cardéis, J. Valls, “Area-Optimized Implementation of Quadrature Direct Digital Frequency Synthesizers on LUT-based FPGAs”, IEEE Trans. On Circuits and Systems II, vol. 50, no. 3, march 2003.

[Cho96] T. Chou y K. Roy, “Accurate Power Estimation of CMOS Sequential Circuits”, IEEE Trans. On VLSI Systems, Vol.4, nº3, pp.369-380. September 1996.

[Cir87] M. A. Cirit, “Estimating Dynamic Power Consumption of CMOS Circuits”, Proc. ICCAD, pp. 534-537, November 1987.

[Den94] C. Deng. “Power analysis for CMOS/BiCMOS circuits. “ In Proceedings of the 1994 International Workshop on Low Power Design, pages 3–8, April 1994.

[Din00] Ding, C-S., C-T. Hsieh and M. Pedram, “Improving efficiency of the Monte Carlo power estimation”, IEEE Trans. on VLSI Systems, Vol. 8, No. 5, (2000) pp. 584-593.

[Faw97] Fawcett, B.: FPGAs, Power and Packages. XCELL (1997)

[Fey96] Richard P. Feynman, “Feynman Lectures on Computation”, Ed. A.J.G. Hey and R.W. Allen. Addison-Wesley, 1996.

[Gar00] Andrés David García García, “Etude sur l’estimation et l’optimisation de la consommation de puissance des circuits logiques programmables du type FPGA”, Ph. D. Thesis, Ecole Nationale Supérieure des Télécommunications, paris, 2000.

[Geo94] B. George et al, “Power Analysis for Semi-Custom Design”, IEEE 1994 Custom Integrated Circuits Conf., pp.249-252, New York: IEEE Press 1994.

[Gho92] Abhijit Ghosh, Srinivas Devadas, Kurt Keutzer, Jacob White, “Estimation of Average Switching Activity in Combinational and Sequential Circuits”, In Procs. of the 29th ACM/IEEE Conference on Design Automation, pp. 253-259, 1992

[Gun01] S.H. Gunther, F. Binns, D.M. Carmean and J.C. Hall, “Managing the Impact of Increasing Microprocessor Power Consumption”, Intel Technology Journal Q1, 2001.

[Guy98] Alain Guyot and Sélim Abou-Samra, “Low Power CMOS Digital Design”, In proc. Of International Conference on Microelectronics 1998 (ICM’98), Monastir, Tunisia, December 1998.

[Hay01] Brian Hayes, “The Computer and the Dynamo”, American Scientist, Vol 89, No 5, September-October, 2001. pp. 390-394.

[ITRS04] ITRS Technology Working Group, “Overall Roadmap Technology Characteristics (ORTC)”, from the International Technology Roadmap for Semiconductors (ITRS). 2004 Upgrade. Available at http://public.itrs.net

[ITRS05] International Technology Roadmap for Semiconductors, 2005 Edition, available at http://public.itrs.net

[Iye86] R. Iye, D. Rossetti and M. Hsueh, “Measurement and Modelling of computer reliability as affected by system activity”, ACM Trans. On Computer Systems, 4(3):214-237, Aug. 1986.

[Kao02] James Kao, Siva Narendra, Anantha Chandrakasan, “Subthreshold leakage modeling and reduction techniques”, In proc. of the 2002 IEEE/ACM international conference on Computer-Aided Design, pp. 141-148, 2002

[Kaw00] K. Kawamoto, J.G. Koomey, B. Nordman, R.E. Brown, M.A. Piette and A.K. Meier, “Electricity Used by Office Equipment and Network Equipment in the U.S.”, Proceedings of the 2000 ACEEE Summer Study on Energy Efficiency in Buildings. Asilomar, CA. August 2000.

[Kaw01] K. Kawamoto, J.G. Koomey, B. Nordman, R.E. Brown, M.A. Piette, M. Ting and A.K. Meier, “Electricity Used by Office Equipment and Network Equipment in the U.S.: Detailed Report and Appendices”, Lawrence Berkeley National Laboratory Internal Report LBNL-45917, University of California. February 2001.

[Kle05] M. Klein, “The Virtex-4 Power Play”, Xcell Journal, Spring 05

[Kol01] J. Koliski, R. Chary, A. Henroid and B. Press, “Building the Power-Efficient PC”, Intel Press, 2001.

[Koz01] J. Kozhaya and F. N. Najm, “Power estimation for large sequential circuits”, IEEE Transactions on VLSI, vol. 9, no. 2, pp. 400-407, April 2001.

[Kwa98] B. Kwak, and E.S. Park, “An Optimization-Based Error Calculation for Statistical Power Estimation of CMOS Logic Circuits,” in Procs. of the Design Automation Conference, San Francisco, California, USA, pp. 690-693, 1998.

[Lai01] John A. Laitner, Jonathan Koomey, Ernst Worrell, Etan Gumerman. “Re-estimating the Annual Energy Outlook 2000 Forecast Using Updated Assumptions about the Information Economy”. Presented at the American Economic Association Conference. New Orleans, LA. January 7 2001. (Also LBNL-46418).

[Lal97] Pradeep Lall, “Influence of Temperature on Microelectronics and System Reliability”, CRC Press, 1997

[Lan61] R. Landauer, “Irreversibility and Heat Generation in the Computing Process”, IBM Journal of Research and Development, Vol 5, N 3, pp. 261-269, 1961.

[Lan94] P. Landman, Low-Power Architectural Design Methodologies, Ph. D. Thesis, Electronic Research Laboratory, University of California, Berkeley, August 1994.

[Li03] Fei Li, Deming Chen, Lei He, Jason Cong: “Architecture evaluation for power-efficient FPGAs”, Proc. Of Int. Symp on Field Programmable Gate Arrays, 2003, pp. 175–184

[Lop03] Sergio López Buedo, “Técnicas de Verificación Térmica para Arquitecturas Dinámicamente Reconfigurables”, Ph. D. Thesis, Departamento de Ingeniería Informática, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Julio 2003.

[Lop97] S. López-Buedo, “Técnicas de Diseño de Alta Velocidad y Bajo Consumo” Memoria del Proyecto de Fin de Carrera, ETSI Telecomunicación, Universidad Politécnica de Madrid, Septiembre 1997.

[May01] Uwe Meyer-Baese, “Digital signal processing with field programmable gate arrays”, Springer, 2001.

[Mod03] Model Technology, ModelSim SE User’s Manual, 2003.

[Naj90] F. Najm, R. Burch, P. Yang, and I. Hajj, “Probabilistic simulation for reliability Analysis of CMOS VLSI circuits,” IEEE Transactions on Computer-Aided Design, vol. 9, no. 4, pp. 439-450, April 1990 (Errata in July 1990).

[Naj91] F. Najm, “Transition density, a stochastic measure of activity in digital circuits,” 28th ACM/IEEE Design Automation Conference, San Francisco, CA, pp. 644-649, June 17-21, 1991.

[Naj93] F. Najm, “Transition density: a new measure of activity in digital circuits,” IEEE Transactions on Computer-Aided Design, vol. 12, no. 2, pp. 310-323, February 1993.

[Naj94] F. Najm, "A survey of power estimation techniques in VLSI circuits," IEEE Transactions on VLSI Systems, vol. 2, no. 4, pp. 446-455, Dec. 1994.

[Naj95] F. N. Najm, S. Goel, and I. N. Hajj, "Power estimation in sequential circuits" ACM/IEEE Design Automation Conference, pp. 635-640, 1995.

[Naj98] F. N. Najm and M. G. Xakellis, “Statistical estimation of the switching activity in VLSI circuits”, VLSI Design, vol. 7, no. 3, pp. 243-254, 1998.

[Osm98] Timothy A. Osmulski, Implementation and Evaluation of a Power Prediction Model for a Field Programmable Gate Array, Master's Thesis, Department of Computer Science, Texas Tech University, Lubbock, TX, May 1998.

[Ped94] M. Pedram, "Power estimation and optimization at the logic level," Int'l Journal of High Speed Electronics and Systems, Vol. 5, No. 2 (1994), pp. 179-202.

[Ped97] M. Pedram, “Design technologies for Low Power VLSI”, In Encyclopaedia of Computer Science and Technology, Vo. 36, Marcel Dekker, Inc., 1997, pp. 73-96.

[Poo02] Kara K.W. Poon, Andy Yan, Steven J.E. Wilton, “A Flexible Power Model for FPGAs”, LNCS, Volume 2438, Jan 2002, pp. 312-321.

[Poo05] Kara K.W. Poon, Steven J.E. Wilton, and A. Yan, “A Detailed Power Model for Field-Programmable Gate Arrays,” ACM Transactions on Design Automation of Electronic Systems (TODAES), vol. 10, issue 2, pp. 279-302, April 2005.

[Rab96] Rabaey, Jan M. “Digital integrated circuits: a design perspective”. Upper Saddle River: Prentice-Hall International, 1996.

[Rab96b] Jan M. Rabaey and Massoud Pedram. “Low power design methodologies”. Boston, Kluwer Academic, 1996.

[San03] T. Sansaloni, A. Pérez-Pascual, J. Valls, “Area-efficient FPGA-based FFT processor”, Electronic Letters, Vol.39, N.41, September 2003

[Sax02] V. Saxena, F. N. Najm, and I. N. Hajj, "Estimation of state line statistics in sequential circuits," *ACM Transactions on the Design Automation of Electronic Systems*, Vol. 7, No. 3, pp. 455-473, July 2002.

[Sch95] P.H. Schneider, “PAPSAS: A Fast Switching Activity Simulator”, PATMOS’95, 1995, pp. 351-360.

[Sch96a] P. Schneider and S. Krishnamoorthy. “Effects of correlations on accuracy of power analysis - an experimental study”, International Symposium on Low Power Electronics and Design, Monterey, California, United States, 1996, pp. 113-116.

[SDF01] IEEE Std 1497-1999, IEEE Standard for Standard Delay Format (SDF) for the Electronic Design Process. The Institute of Electrical and Electronics Engineers, Inc. 3 Park Avenue, New York, NY 10016-5997, USA, 2001.

[Sha02] L. Shang, A. S. Kaviani, K. Bathala, “Dynamic Power Consumption in Virtex-II FPGA Family”, FPGA 2002 Monterey, California, USA, February 24-26, 2002, pp. 157-164.

[Smi01] Van Smith, “Pentium 4 Thermal Throttling”, available at http://www.vanshardware.com.

[Sut01] G. Sutter, E. Todorovich, S. López-Buedo y E. Boemo “Propiedad Conmutativa y Diseño de Bajo Consumo: Algunos Ejemplos en FPGAs”, VII WorkShop de IBERCHIP, Montevideo, Uruguay, 21-23 de Marzo, 2001.

[Sut05] Gustavo Sutter, “Aportes a la Reducción de Consumo en FPFAs”, Ph. D. Thesis, Departamento de Ingeniería Informática, Escuela Politécnica Superior, Universidad Autónoma de Madrid, April 2005.

[Tan99] Tan, J: Virtex Power Estimator User Guide. XAPP 152 (1999)

[Tek01] Tektronix Inc., ”TLA 704 Logic Analizar User Guide” available at http://www.tektronix.com

[Tek02] Tektronix Inc. “TLA7PG2 Pattern Generator Module” available at http://www.tektronix.com

[Tod00] E. Todorovich, G. Sutter, N. Acosta and E. Boemo, “End-user low-power alternatives at topological and physical levels. Some examples on FPGAs”, XV Conference on Design of Circuits and Integrated Systems (DCIS2000), Le Corum, Montpellier, France, November 21-24, 2000.

[Tod01] E. Todorovich, G. Sutter, N. Acosta, S. López-Buedo y E. Boemo “Relación entre Velocidad y Consumo en FPGAs”, VII Workshop IBERCHIP, Montevideo, Uruguay, 21-23 de Marzo, 2001.

[VCD01] IEEE Std 1364-2001 (Revision of IEEE Std 1364-1995), IEEE Standard Verilog Hardware Description Language. The Institute of Electrical and Electronics Engineers, Inc. 3 Park Avenue, New York, NY 10016-5997, USA, 2001.

[VIT01] IEEE Std 1076.4-2000, IEEE Standard for VITAL ASIC (Application Specific Integrated Circuit) Modeling Specification. The Institute of Electrical and Electronics Engineers, Inc. 3 Park Avenue, New York, NY 10016-5997, USA, 2001.

[Will99] Craig Willert, “The Evolution of Programmable Logic Design Technology”, XCell

Journal, Issue 32, 2nd quarter 1999, pp. 5-8.

[Xac94] M. Xakellis and F. Najm, “Statistical Estimation of the Switching Activity in Digital Circuits”, 31st ACM/IEEE Design Automation Conference, San Diego, CA, pp. 728-733, 1994.

[XCG03] Xilinx Inc. “CORE Generator Guide”, an Xilinx ISE 6 Software Manual, 2003, available at http://www.xilinx.com

[Xde03] Xilinx Inc. “Development System Reference Guide”, an Xilinx ISE 6 Software Manual, 2003, available at http://www.xilinx.com

[XDOM] Dieter Köhler, ”Extended Document Object Model“, available at http://www.philo.de/xml/dom/

[XDS01] Xilinx Inc. “Virtex Data Sheets: Virtex 2.5 V FPGAs”, 2001, available at http://www.xilinx.com

[XDS02] Xilinx Inc. “Virtex-E Data Sheets: Virtex-E 1.8 V FPGAs”, 2002, available at http://www.xilinx.com

[XDS05] Xilinx Inc. “Virtex-II Platform FPGAs: Complete Data Sheet”, 2005, available at http://www.xilinx.com

[XDS06] Xilinx Inc. “Virtex-4 Data Sheet: DC and Switching Characteristics”, 2006, available at http://www.xilinx.com

[Xil] Xilinx Inc. at http://www.xilinx.com

[Xil00] Xilinx Inc.: XC4000XL Power Calculation. XCELL, Nº27 (2000) pp 29

[Xil01] Xilinx Inc.: XPower Tutorial: FPGA Design, XPower (v1.1). (2001) Available at http://www.xilinx.com.

[Xil05] Xilinx Inc., “Development System Reference Guide”, 2005.

[Xil97h] Xilinx Press, “A Simple Method of Estimating Power in XC4000XL/EX/E FPGAs“, Application Brief, XBRF 014 June 30, 1997.

[XilPow] Xilinx Inc. “Power Central”, available at http://www.xilinx.com/products/design_resources/power_central/

[XilSaV] Xilinx Inc., “Synthesis and Verification Design Guide”, available at http://www.xilinx.com

[XilUser] Xilinx Inc.: “ISE 7 User Guide”. Available at http://www.xilinx.com

[XST03] Xilinx Inc. “XST User Guide”, an Xilinx ISE 6 Software Manual, 2003, available at http://www.xilinx.com

[Yua97] L. Yuan, C. Teng, S. Kang, “ Statistical estimation of average power dissipation in CMOS VLSI circuits using nonparametric techniques”, Procs. of the 1996 international symposium on Low power electronics and design, Monterey, California, United States, pp. 73 – 78, August 1996.

[Yua98] L. Yuan, C. Teng, S. Kang, “Statistical estimation of average power dissipation using nonparametric techniques”, IEEE Trans. on VLSI Systems, Vol 6, No 1, pp. 65-73, Mar 1998.

Appendix A

Tcl/Tk script for the A-DyP Power Estimation Tool

Code A.1 is the Tcl/Tk script used in this work to integrate the several programs needed for the A-DyP Power Estimation Tool. The estimation process is triggered when the `powerEstim` procedure is called (without arguments) within the Modelsim command line. It was tested with Modelsin 5.7g SE.

```
1  ##########################################################
2  #
3  #
4  # ESTIMATE.TCL
5  #
6  # Activity Estimation using Statistics with Modelsim
7  #
8  # Elías TODOROVICH
9  #       etodorov@uam.es
10 #
11 #      Script TCL/TK
12 #
13 ##########################################################

14
15
16 # Global Scope Variables
17 # Related with text specified indirectly in Tk labels
18
19 set naestate  Start
20 set sampleNum 0
21
22
23 ###################################
24 #
25 # Main Procedure: Power Estimation
26 #
```

```
27 proc powerEstim {} {
28
29   set actEstimFinished true
30   set suspendActEstim false
31
32   # Statistics on Execution Time
33
34   set ts [clock seconds]
35   set te 0
36   set tact 0
37   set tcap 0
38   set tini 0
39
40   # Sample Block Number
41
42   set blkNr 0
43   set fName transitions$blkNr.vcd
44   set existe [ file exists $fName ]
45   while { $existe == 1 } {
46     incr blkNr
47     set fName transitions$blkNr.vcd
48     set existe [ file exists $fName ]
49   }
50   puts ""
51   puts "Proccesing Block Sample number $blkNr..."
52   puts ""
53
54   # Power Estimation
55
56   if { [iniPowerEstim $blkNr] } {
57     puts ""
58     puts "Starting ACTIVITY Estimation..."
59     puts ""
60
61     # If you are not using SE comment the next line
62     # Not all Modelsim simulators support Tk
63     drawFeedbackDialog
64
65     activityEstim
66
67     capacitanceEstim
68
69     set te [clock seconds]
70
71     if { $actEstimFinished=="true" } {
72       writeTotalExecTimeStat
73       saveResults
74     }
75
76   } else {
77     puts ""
78     puts "Error: File user.ini not found."
79     puts ""
80   }
81 }
```

```
82
83
84 ###################################
85 #
86 # Initialize Power Estimation
87 #   i: Current Bloch Number
88 #
89 proc iniPowerEstim { i } {
90
91   upvar 1 ts ts
92   upvar 1 tini tini
93
94   set existTables [ file exists "SamplesTb.db"]
95   if { $existTables } {
96     set genera " ------------ Continuing...--------------"
97     set tauxini [clock seconds]
98     do ini_msim.do
99     set tauxend [clock seconds]
100     set tini [expr $tauxend - $tauxini]
101   } else {
102     # Show the user interface to set the parameters
103     exec User.exe
104
105     set file [ open transitions.log { RDWR CREAT TRUNC } ]
106     set genera " ------------ File transitions.log ------"
107     close $file
108     set tauxini [clock seconds]
109     do first_ini_msim.do
110     set tauxend [clock seconds]
111     set tini [expr $tauxend - $tauxini]
112   }
113
114   # Adds all VHDL signals to a VCD file
115   vcd add -file transitions$i.vcd *
116
117
118   set existINIFile [ file exists "User.ini" ]
119   if { $existINIFile } {
120     set file [ open transitions.log { RDWR APPEND } ]
121     puts $file $genera
122     puts $file " ----------------------------------------"
123     set fecha [clock format $ts -format "  %d %b %Y
   %H:%M:%S"]
124     puts -nonewline $file " Generated at: "
125     puts $file $fecha
126     puts $file ""
127     close $file
128   }
129   return $existINIFile
130 }
131
132
133 ###############################
134 #
135 # Tk: Build the top level widget
```

```
136 #
137 proc drawFeedbackDialog {} {
138
139   upvar 1 ts ts
140   upvar 1 te te
141   upvar 1 actEstimFinished actEstimFinished
142   upvar 1 suspendActEstim suspendActEstim
143   global sampleNum
144   global naestate
145   set maxWidth 35
146
147   # (If exists) Destroy the feedback window
148
149   catch {destroy .main}
150
151   # Creates the Estimation Process Drawback Window
152
153   toplevel .main
154   wm deiconify .
155   wm title .main "State of the Estimation Proccess"
156
157   #
158   # Create a frame for the information labels
159   #
160
161   frame .main.top -bd 10 -bg black
162   pack  .main.top -side top -fill x
163
164   # Create the start date and time label
165
166   set fechaStart [clock format $ts -format "%d %b %Y
   %H:%M:%S"]
167   append init "Start at: " $fechaStart
168   label .main.top.start -text $init -bg black -fg white
169   pack  .main.top.start
170
171   # Create the status label
172
173   set naestate "Initializing..."
174   label .main.top.status -textvariable naestate -bg black -
   fg white -width $maxWidth -anchor center -font {bold 12}
175   pack  .main.top.status
176
177   # Create feedback information labels
178
179   label .main.top.samplestitle -text "Samples Processed: "
   -bg black -fg white -font {bold 12}
180   label .main.top.samples -textvariable sampleNum -bg black
   -fg white -font {bold 14}
181   pack  .main.top.samplestitle
182   pack  .main.top.samples
183
184   #
185   # Create a frame for buttons
186   #
```

```
187
188   frame .main.bottom -bd 10 -bg black
189   pack  .main.bottom -side bottom -fill both -expand true
190
191   button .main.bottom.cerrar -text CLOSE -command { destroy
   .main }
192   pack   .main.bottom.cerrar -side right -padx 2m -pady 2m
193
194   button .main.bottom.cancelar -text SUSPEND -command { \
195        set suspendActEstim true
196        set actEstimFinished false
197        set naestate "WAIT..."
198        update
199       }
200   pack   .main.bottom.cancelar -side right -padx 2m -pady
   2m
201 }
202
203
204 ######################################################
205 #
206 # Estimate Average Transition Number for all Nodes
207 #
208 proc activityEstim {} {
209   upvar 1 blkNr blkNr
210   upvar 1 te te
211   set END_SIM false
212   upvar 1 suspendActEstim suspendActEstim
213   global sampleNum
214   global naestate
215
216   # Statistics on execution time
217   set tgen 0
218   set tsim 0
219   set tsav 0
220   set ttrn 0
221   set tupd 0
222   set tcut 0
223   upvar 1 tact tact
224
225   while { $END_SIM == "false" } {
226
227     set naestate "Generating..."
228     puts $naestate
229     update
230     set tauxini [clock seconds]
231     exec generator.exe -pg tla -d [pwd]
232     set tauxend [clock seconds]
233     set tgen [expr $tgen + $tauxend - $tauxini]
234
235     set naestate "Simulating..."
236     puts $naestate
237     update
238     set tauxini [clock seconds]
239     do simulate.do
```

```
240     set tauxend [clock seconds]
241     set tsim [expr $tsim + $tauxend - $tauxini]
242
243     set naestate "Saving..."
244     puts $naestate
245     update
246     set tauxini [clock seconds]
247     saveVec
248     set tauxend [clock seconds]
249     set tsav [expr $tsav + $tauxend - $tauxini]
250
251     set naestate "Processing..."
252     update
253     puts "Analizing transitions[expr $blkNr - 1].vcd..."
254     set tauxini [clock seconds]
255     catch "exec Transitions.exe -d [pwd] transitions.vcd"
   sampleNum
256     set tauxend [clock seconds]
257     set ttrn [expr $ttrn + $tauxend - $tauxini]
258     puts "$sampleNum clock cycles analized..."
259
260     set naestate "Updating..."
261     puts $naestate
262     puts ""
263     update
264     set tauxini [clock seconds]
265     exec Update.exe -d [pwd]
266     set tauxend [clock seconds]
267     set tupd [expr $tupd + $tauxend - $tauxini]
268
269     set tauxini [clock seconds]
270     catch { exec Cuter.exe -d [pwd]} END_SIM
271     set tauxend [clock seconds]
272     set tcut [expr $tcut + $tauxend - $tauxini]
273   }
274
275   set tact [expr $tgen + $tsim + $tsav + $ttrn + $tupd +
   $tcut]
276
277   quit -sim
278   set naestate "END OF ACTIVITY ESTIMATION!"
279   puts ""
280   puts $naestate
281   puts ""
282   puts "See transitions.log for details in pwd."
283   puts ""
284   update
285
286   # Tk Command
287   # Destroy the suspend button
288
289   destroy .main.bottom.cancelar
290
291   # transitions.log: le agrega estadisticas de tiempo
292   # de ejecucion y la hora de finalizacion
```

```
293   set file [open transitions.log {RDWR APPEND}]
294
295   puts $file "Activity Estim. Time Statistics (seconds)"
296   puts $file [append z1 "Input vector generation\t" $tgen]
297   puts $file [append z2 "Simulation\t" $tsim]
298   puts $file [append z3 "Saving\t" $tsav]
299   puts $file [append z4 "Transition analisys\t" $ttrn]
300   puts $file [append z5 "Statistics computation\t" $tupd]
301   puts $file [append z6 "Stoping criteria evaluation\t"
   $tcut]
302   puts $file ""
303
304   close $file
305 }
306
307
308 ################################################
309 #
310 # Capacitance Estimation
311 #
312 proc capacitanceEstim {} {
313   upvar 1 te te
314   upvar 1 ts ts
315
316   global naestate
317
318   # This procedure colects Statistics on execution time
319   #   tvhd txdl txml txpw tpwr tmap
320   upvar 1 tcap tcap
321
322   puts ""
323   puts "Starting POWER Estimation..."
324   puts ""
325
326   # VHDL Parsing
327   set naestate "Analizing vhdl file..."
328   puts $naestate
329   update
330   set tauxini [clock seconds]
331   exec parserVHD.exe -d [pwd]
332   set tauxend [clock seconds]
333   set tvhd [expr $tauxend - $tauxini]
334
335   # XDL Parsing
336   set naestate "Analizing phisical info from xdl file..."
337   puts $naestate
338   update
339   set tauxini [clock seconds]
340   exec xdlParser.exe -d [pwd]
341   set tauxend [clock seconds]
342   set txdl [expr $tauxend - $tauxini]
343
344   # XML Activity Report
345   set naestate "Generating Activity Rep. in XML format..."
346   puts $naestate
```

```
347   update
348   set tauxini [clock seconds]
349   #exec XMLRep.exe -d [pwd]
350   set tauxend [clock seconds]
351   set txml [expr $tauxend - $tauxini]
352
353   # Connection with XPower
354   set naestate "Generating Power Rep. with XPower..."
355   puts $naestate
356   update
357   set tauxini [clock seconds]
358   do Connect2XPower.do
359   set tauxend [clock seconds]
360   set txpw [expr $tauxend - $tauxini]
361
362   # XPower Report Parsing: Not neccesary now
363   #set naestate "Analizing Power Rep (pwr: text format)..."
364   #puts $naestate
365   #update
366   #set tauxini [clock seconds]
367   #exec pwr.exe -d [pwd]
368   #set tauxend [clock seconds]
369   #set tpwr [expr $tauxend - $tauxini]
370
371   # PWA Parsing
372   set naestate "Analizing Capacitance Report..."
373   puts $naestate
374   update
375   set tauxini [clock seconds]
376   exec PWAparser.exe -d [pwd]
377   set tauxend [clock seconds]
378   set tpwa [expr $tauxend - $tauxini]
379
380   # Calculates Power and write a Report
381   set naestate "Writing Power Report..."
382   puts $naestate
383   update
384   set tauxini [clock seconds]
385   exec report.exe -d [pwd] -v
386   set tauxend [clock seconds]
387   set trep [expr $tauxend - $tauxini]
388
389   # Power Mapping
390   set naestate "Maping to Physical Positions..."
391   puts $naestate
392   update
393   set tauxini [clock seconds]
394   exec powerMap.exe -d [pwd] -r 1
395   exec powerMap.exe -d [pwd] -r 2
396   exec powerMap.exe -d [pwd] -r 4
397
398   exec activityMap.exe -d [pwd] -r 1
399   exec activityMap.exe -d [pwd] -r 2
400   exec activityMap.exe -d [pwd] -r 4
401
```

```
402   exec capacitanceMap.exe -d [pwd] -r 1
403   exec capacitanceMap.exe -d [pwd] -r 2
404   exec capacitanceMap.exe -d [pwd] -r 4
405   set tauxend [clock seconds]
406   set tmap [expr $tauxend - $tauxini]
407
408   set tcap [expr $tvhd + $txdl + $txml + $txpw + $tpwa +
   $tmap + $trep]
409
410   set naestate "END OF POWER ESTIMATION!"
411   puts ""
412   puts $naestate
413   puts ""
414   puts ""
415   update
416
417   # Write Execution Time Statistics on the Log File
418   # File: transitions.log
419
420   set file [open transitions.log {RDWR APPEND}]
421
422   puts $file ""
423   puts $file "Capacitance Estimation Time Statistics
   (seconds)"
424   puts $file [append z1 "VHDL parsing\t" $tvhd]
425   puts $file [append z2 "XDL parsing\t" $txdl]
426   puts $file [append z3 "XML generation\t" $txml]
427   puts $file [append z4 "XPower execution\t" $txpw]
428   puts $file [append z5 "PWA parsing\t" $tpwa]
429   puts $file [append z6 "Report writing\t" $trep]
430   puts $file [append z7 "Power Maps\t" $tmap]
431   puts $file ""
432
433   close $file
434
435   # Ring the terminal bell
436   bell
437 }
438
439
440 ##########################################
441 #
442 # Save a VCD file for further analysis
443 #
444 proc saveVec {} {
445   upvar 2 blkNr blkNr
446
447   # Flushes the contents of the VCD file buffer to the last
   VCD file
448   vcd flush transitions$blkNr.vcd
449
450   # Turns off VCD dumping to the last file and records all
   VCD variable values as x
451   vcd off transitions$blkNr.vcd
452   vcd flush transitions$blkNr.vcd
```

```
453
454   # Es importante para abrir el archivo VCD que esta siendo
   usado por el simulador:
455   # hasta que termina la simulación.
456   file copy -force transitions$blkNr.vcd transitions.vcd
457
458   # Increments the number of simulation blocks
459   incr blkNr
460
461   # Adds all VHDL signals to the next VCD file
462   vcd add -file transitions$blkNr.vcd *
463 }
464
465
466 ################################################
467 #
468 # Finalization Tasks for Power Estimation
469 #
470 proc saveResults {} {
471
472   # Use different names for back-up files
473   set x 1
474   set nom "estim"
475   append bkDir $nom $x
476
477   while { [ file isdirectory $bkDir ] } {
478     set bkDir $nom
479     incr x
480     append bkDir $x
481   }
482
483   set bkDir [append nom $x]
484   file mkdir $bkDir
485
486   foreach f [glob -nocomplain TekPatGen*.txt] {
487     file copy $f $bkDir/$f
488     file delete $f
489   }
490   foreach f [glob -nocomplain NodesTb.*] {
491     file copy $f $bkDir/$f
492     file delete $f
493   }
494   foreach f [glob -nocomplain SamplesTb.*] {
495     file copy $f $bkDir/$f
496     file delete $f
497   }
498   if { [file exists "User.ini"] } {
499     file copy "User.ini" $bkDir/User.ini
500   }
501   foreach f [glob -nocomplain *.vcd] {file delete $f}
502   if { [file exists "transitions.log"] } {
503     file copy "transitions.log" $bkDir/transitions.log
504     file delete "transitions.log"
505   }
506   if { [file exists "test_xpwr.xml"] } {
```

```
507     file copy "test_xpwr.xml" $bkDir/test_xpwr.xml
508     file delete test_xpwr.xml
509   }
510   if { [file exists "test.pwr"] } {
511     file copy "test.pwr" $bkDir/test.pwr
512     file delete test.pwr
513   }
514   if { [file exists "report.txt"] } {
515     file copy "report.txt" $bkDir/report.txt
516     file delete report.txt
517   }
518   foreach f [glob -nocomplain *.dat] {
519     file copy $f $bkDir/$f
520     file delete $f
521   }
522   foreach f [glob -nocomplain CapMapTb.*] {
523     file delete $f
524   }
525   foreach f [glob -nocomplain ActMapTb.*] {
526     file delete $f
527   }
528   foreach f [glob -nocomplain PowerMapTb.*] {
529     file delete $f
530   }
531 #  foreach f [glob -nocomplain SlicesTb.*] {
532 #    file delete $f
533 #  }
534 }
535
536
537 ################################################
538 #
539 # Write on Log file Total Execution Time Statistics
540 #
541 proc writeTotalExecTimeStat {} {
542   upvar 1 te te
543   upvar 1 ts ts
544   upvar 1 tact tact
545   upvar 1 tcap tcap
546   upvar 1 tini tini
547   global naestate
548
549   set difHora -1
550
551   # Total Execution TIME
552   set TotalExecTime [expr $te - $ts]
553
554   #
555   # Tk Commands
556   # Create the end date and time label
557   #
558
559   set fechaEnd [clock format $te -format "%d %b %Y
   %H:%M:%S"]
560   append endinit "End at: " $fechaEnd
```

```
561   #label .main.top.end -text $endinit -bg black -fg white
562   #pack  .main.top.end
563
564   # Elapsed Time
565   set elapsedTime [expr $te - $ts]
566   set naestate "END - TIME: "
567   set min [clock format $elapsedTime -format "%M:%S"]
568   set hora [clock format $elapsedTime -format "%H"]
569   set hora [expr $hora+$difHora]
570   #if { $hora > 23 } {
571   # set hora [expr $hora-24]
572   #}
573   append et "Elapsed Time: " $hora
574   append et ":" $min
575   set naestate $et
576   update
577
578   # Tcl/Tk Script Execution Time
579   set tclExecTime [expr $TotalExecTime - $tact - $tcap -
   $tini]
580
581   # Write on Log File
582   set file [open transitions.log {RDWR APPEND}]
583
584   puts $file ""
585   puts $file "Power Estimation Execution Time Statistics
   (seconds)"
586   puts $file [append z1 "Total Execution\t" $TotalExecTime]
587   puts $file [append z2 "Initialization\t" $tini]
588   puts $file [append z3 "Activity Estimation\t" $tact]
589   puts $file [append z4 "Capacitance Estimation\t" $tcap]
590   puts $file [append z5 "Tcl/Tk script\t" $tclExecTime]
591   puts $file ""
592
593   #puts $file $fechaEnd
594   close $file
595
596 }
```

Code A.1: Power Estimation main Tcl/Tk script

Appendix B

Input Patterns File (.do)

Code B.1 shows the complete contents of an input pattern .do file. This example corresponds to a simulation run for the FIRDA_8 test circuit. These do files are generated by the `generate.exe` program according to the user specifications. See Chapter 4 for more details about the `generate.exe` program. These scripts are run with Modelsin 5.7g SE.

```
1  #
2  # SIMULATE.DO
3  #
4  # Activity Estimation using Statistics with Modelsim
5  #
6  # Elías TODOROVICH
7  #       etodorov@uam.es
8  #
9  #
10 #
11 force R 0
12 force C 1 8ns , 0 48 ns -repeat 80 ns
13 force ND 1
14 force I0 00000000 0 ns , 01101111 80 ns , 01001001 160 ns ,
   01100110 240 ns , 00111101 320 ns , 10001111 400 ns ,
   10011000 480 ns , 00101011 560 ns , 11001011 640 ns ,
   01010011 720 ns , 01111101 800 ns , 10100101 880 ns ,
   00011100 960 ns , 11110100 1040 ns , 11011101 1120 ns ,
   10101011 1200 ns , 01111000 1280 ns
15 run 1360 ns
16 force I0   11011100 80 ns , 00110101 160 ns , 10101010 240
   ns , 01001001 320 ns , 11010100 400 ns , 11000011 480 ns ,
   00000110 560 ns , 10100101 640 ns , 11100000 720 ns ,
   10000000 800 ns , 11011100 880 ns , 01101101 960 ns ,
   01010001 1040 ns , 11011011 1120 ns , 01110001 1200 ns ,
   10011101 1280 ns
17 run 1280 ns
18 force I0   01101001 80 ns , 00111000 160 ns , 10110100 240
   ns , 01101111 320 ns , 01111111 400 ns , 01000000 480 ns ,
   00111010 560 ns , 01101100 640 ns , 11111101 720 ns ,
```

```
   00011011 800 ns , 00010101 880 ns , 00110100 960 ns ,
   10101001 1040 ns , 10011111 1120 ns , 11010110 1200 ns ,
   10100000 1280 ns
19 run 1280 ns
20 force I0   11000001 80 ns , 11110001 160 ns , 10100011 240
   ns , 10010000 320 ns , 11010101 400 ns , 10011101 480 ns ,
   01111000 560 ns , 01011111 640 ns , 11010001 720 ns ,
   11100101 800 ns , 10011100 880 ns , 00010100 960 ns ,
   11101110 1040 ns , 00001000 1120 ns , 00101000 1200 ns ,
   01100100 1280 ns
21 run 1280 ns
22 force I0   00011100 80 ns , 00110100 160 ns , 11000000 240
   ns , 10110110 320 ns , 00100000 400 ns , 00000101 480 ns ,
   10101010 560 ns , 00001111 640 ns , 00110001 720 ns ,
   00101011 800 ns , 01110001 880 ns , 11000101 960 ns ,
   11111001 1040 ns , 00011101 1120 ns , 01110010 1200 ns ,
   00100111 1280 ns
23 run 1280 ns
24 force I0   11010100 80 ns , 11110101 160 ns , 01100101 240
   ns , 11101000 320 ns , 00001111 400 ns , 00011110 480 ns ,
   00000011 560 ns , 10010110 640 ns , 01101000 720 ns ,
   11011010 800 ns , 10111101 880 ns , 01001010 960 ns ,
   01100100 1040 ns , 11010011 1120 ns , 00101100 1200 ns ,
   11101000 1280 ns
25 run 1280 ns
26 force I0   10011101 80 ns , 10001010 160 ns , 10011000 240
   ns , 00101110 320 ns , 10000010 400 ns , 00110001 480 ns ,
   01010000 560 ns , 10001011 640 ns , 01111010 720 ns ,
   00010000 800 ns , 01010010 880 ns , 10010111 960 ns ,
   00100011 1040 ns , 10011110 1120 ns , 00101011 1200 ns ,
   11110001 1280 ns
27 run 1280 ns
```

Code B.1: Input Pattern simulate.do example file

Appendix C

The Configuration .ini File

Code C.1 shows the complete contents of an A-DyP configuration .ini file. These .ini files are text files but they can be generated by the User Interface (see Figs. 4.2-5) according to the user specifications. See Chapter 4 for more details about this program. This example corresponds to a simulation run for the 10MULT16_C circuit.

In section FILES, from lines 1 to 5, the names of the required files in A-DyP are specified. They must be in the working directory. In section CIRCUIT FEATURES, from lines 7 to 11, the nodes number is annotated. Also, the name of the top level entity, the device name, and device family are specified. From lines 13 to 16, in the CIRCUIT CONFIGURATION section, parameters related with circuit characteristics are detailed. Next, in the STAT PARAMETERS section, from lines 18 to 21, the required parameters for the statistical technique are selected. In the OPTIMIZATION section, from lines 23 to 27, parameters related with A-DyP performance are specified. Finally, in the CLOCK, PORTS, and PORTx sections, the input pattern characteristics for every input port in the DUT including the clock are detailed.

```
1  [FILES]
2  Ncd=mult_rep.ncd
3  Vhdl=mult_rep_timesim.vhd
4  Sdf=mult_rep_timesim.sdf
5  Pcf=mult_rep.pcf
6
7  [CIRCUIT FEATURES]
8  VHDL Nodes=27471
9  Top Entity=mult_rep
10 Device=xc2v3000fg676-6
11 Device Family=VIRTEX_II
12
13 [CIRCUIT CONFIGURATION]
14 Min Glitch=900
```

```
15 Latency=2
16 SetupCyclesR=0
17
18 [STAT PARAMS]
19 Confidence=95,00
20 Error=5
21 Bound=0,05
22
23 [OPTIMIZATION]
24 OptStrength=100
25 Save Samples=NOSAVE
26 All Samples=MIN
27 VCDStart=27484
28
29 [CLOCK]
30 Name=CLK
31 Period=20
32 Period Scale=ns
33 Setup Time=8
34
35 [PORTS]
36 Port1=CLK C 20 ns
37 Port2=B R 16
38 Port3=A R 16
39 Count=3
40
41 [PORT2]
42 Pos0=R 0,5 0,5
43 Pos1=R 0,5 0,5
44 Pos2=R 0,5 0,5
45 Pos3=R 0,5 0,5
46 Pos4=R 0,5 0,5
47 Pos5=R 0,5 0,5
48 Pos6=R 0,5 0,5
49 Pos7=R 0,5 0,5
50 Pos8=R 0,5 0,5
51 Pos9=R 0,5 0,5
52 Pos10=R 0,5 0,5
53 Pos11=R 0,5 0,5
54 Pos12=R 0,5 0,5
55 Pos13=R 0,5 0,5
56 Pos14=R 0,5 0,5
57 Pos15=R 0,5 0,5
58 Last Value=1101000110001110
59
60 [PORT3]
61 Pos0=R 0,5 0,5
62 Pos1=R 0,5 0,5
63 Pos2=R 0,5 0,5
64 Pos3=R 0,5 0,5
65 Pos4=R 0,5 0,5
66 Pos5=R 0,5 0,5
67 Pos6=R 0,5 0,5
68 Pos7=R 0,5 0,5
69 Pos8=R 0,5 0,5
```

```
70 Pos9=R 0,5 0,5
71 Pos10=R 0,5 0,5
72 Pos11=R 0,5 0,5
73 Pos12=R 0,5 0,5
74 Pos13=R 0,5 0,5
75 Pos14=R 0,5 0,5
76 Pos15=R 0,5 0,5
77 Last Value=1111000100101110
```

Code C.1: A-DyP typical configuration .ini file

Appendix D

Power Report File

Code D.1 shows the contents of an A-DyP power report file. This example corresponds to a simulation run for the 10MULT16_C circuit with the parameter specified in Code C.1.

```
1  ------------------------------------------------------------
2  |
3  | TOTAL AND INDIVIDUAL NODES POWER ESTIMATION V1.0
4  |                      Universidad Autonoma de Madrid
5  | - e-mail: etodorov@uam.es
6  | - REPORT -
7  | - This file was generated on 17/02/2006 15:20:44
8  |
9  ------------------------------------------------------------
10
11
12  ------------------------------------------------------------
13 |                Summary                                     |
14  ------------------------------------------------------------
15 - Statisticals Parameters:
16             Confidence   : 95,00 %
17             Error        : 5 %
18             Minimal Mean : 0,05  transitions per clock cycle
19
20 - Taken Samples:          6973
21 - Low Density Nodes:      5,99 %
22 - Regular Density Nodes:  94,01 %
23
24 - Total Transition Count per Clock Period: 12846,8
25 - Nodes Count                            : 27471
26 - Transition average density per Node    : 0,468
27
28  ------------------------------------------------------------
29 |
30 |                Power Summary                               |
31  ------------------------------------------------------------
32
33 - Total Estimations:
```

```
34        - Avg. Power [mW]:               881,51
35        - Capacitance [pF]:              25910
36        - Switched Cap. [fF per clock]:  289,36
37
38 - Estimations for the Logic Resources:
39
40  ------------------------------------------------------------
41
42 |  FPGA REs.   | Power [mW] | Capacitance [pF] | Switched
   Cap [fF per Clock] |
43  ------------------------------------------------------------
44                 0,00          0                   0,00
45   X_AND2        87,78         3991                39,01
46   X_BUF_PP      163,97        7595                72,88
47   X_BUFGMUX     32,63         290                 14,50
48   X_FF          137,11        5050                60,94
49   X_INV         0,00          0                   0,00
50   X_LUT4        87,25         2782                38,78
51   X_MUX2        82,32         4665                36,59
52   X_ONE         0,00          0                   0,00
53   X_ROC         0,00          0                   0,00
54   X_TOC         0,00          0                   0,00
55   X_TRI_PP      290,45        1536                26,67
56   X_XOR2        0,00          0                   0,00
57   X_ZERO        0,00          0                   0,00
58
59 - Estimations for the General PFGA Resources:
60  ------------------------------------------------------------
61 |  FPGA REs.   | Power [mW] | Capacitance [pF] | Switched
   Cap [fF per Clock] |
62  ------------------------------------------------------------
63
64   Signals       180,43        14908               80,19
65   Clocks        32,63         290                 14,50
66   I/O           290,45        1536                26,67
67   Logic         378,00        9175                168,00
68
69 -The 50 Highest Power Consumption INTERNAL* Nodes are:
70  ------------------------------------------------------------
71 | Nodename        | Avg. Activity | FPGA REs. | Avg. Power |
72  ------------------------------------------------------------
73
74   mult_i0_p[3]          0,4902       X_FF       0,7636
75   mult_i0_p[0]          0,3559       X_FF       0,7007
76   b_7_ibuf              0,4913       X_BUF_PP   0,6242
77   b_15_ibuf             0,4839       X_BUF_PP   0,6139
78   b_6_ibuf              0,4799       X_BUF_PP   0,6092
79   mult_i4_a_int[15]     0,4932       X_FF       0,5584
80   mult_i4_a_int[6]      0,4879       X_FF       0,5524
81   mult_i4_a_int[5]      0,4811       X_FF       0,5381
82   b_14_ibuf             0,4816       X_BUF_PP   0,5296
83   b_10_ibuf             0,4859       X_BUF_PP   0,5252
84   mult_i1_a_int[8]      0,4961       X_FF       0,5196
85   b_13_ibuf             0,4961       X_BUF_PP   0,5193
86   a_8_ibuf              0,4972       X_BUF_PP   0,5145
```

```
87   mult_i2_a_int[7]      0,4810       X_FF       0,5134
88   mult_i9_a_int[13]     0,4919       X_FF       0,5119
89   b_9_ibuf              0,4856       X_BUF_PP   0,5078
90   mult_i0_a_int[1]      0,5028       X_FF       0,4917
91   b_4_ibuf              0,4831       X_BUF_PP   0,4911
92   a_4_ibuf              0,5012       X_BUF_PP   0,491
93   mult_i2_a_int[15]     0,4932       X_FF       0,488
94   mult_i9_a_int[15]     0,4932       X_FF       0,4869
95   mult_i1_a_int[4]      0,4998       X_FF       0,4867
96   b_3_ibuf              0,4885       X_BUF_PP   0,4848
97   a_10_ibuf             0,4876       X_BUF_PP   0,4817
98
99    ...
100
101  ------------------------------------------------------------
102
103 * Clk buffers and I/O Blocks are not included.
104
105 -Complete List of nodes in alphabetical order:
106  ------------------------------------------------------------
107 |
108 | Nodename       | Avg. Activity | FPGA REs. | Avg. Power |
109  ------------------------------------------------------------
110
111   a[0]         0,4863              0
112   a[10]        0,4864              0
113   a[11]        0,4813              0
114   a[12]        0,4886              0
115   a[13]        0,4915              0
116   a[14]        0,4816              0
117   a[15]        0,4952              0
118   a[1]        0,5028              0
119   a[2]        0,4912              0
120   a[3]        0,4896              0
121
122    ...
123
```

Code D.1: A-DyP typical power report file