Watermarking strategies for IP protection of micro-processor cores

L. Parrilla[a], E. Castillo[a], U. Meyer-Baese[b], A. García[a], D. González[a], E. Todorovich[c], E. Boemo[c], A. Lloris[a]

[a]Dept. of Electronics and Computer Technology, Campus Universitario Fuentenueva, 18071, Univ. of Granada, Spain.

[b]Dept. of Electrical & Computer Engineering, FAMU-FSU College of Engineering, Florida, USA 32310-6046.

[c]School of Engineering, Universidad Autónoma de Madrid, 28049, Madrid, Spain.

ABSTRACT

Reuse-based design has emerged as one of the most important methodologies for integrated circuit design, with reusable Intellectual Property (IP) cores enabling the optimization of company resources due to reduced development time and costs. This is of special interest in the Field-Programmable Logic (FPL) domain, which mainly relies on automatic synthesis tools. However, this design methodology has brought to light the intellectual property protection (IPP) of those modules, with most forms of protection in the EDA industry being difficult to translate to this domain. However, IP core watermarking has emerged as a tool for IP core protection. Although watermarks may be inserted at different levels of the design flow, watermarking Hardware Description Language (HDL) descriptions has been proved to be a robust and secure option. In this paper, a new framework for the protection of µP cores is presented. The protection scheme is derived from the IPP@HDL procedure and it has been adapted to the singularities of µP cores, overcoming the problems for the digital signature extraction in such systems. Additionally, the feature of hardware activation has been introduced, allowing the distribution of µP cores in a "demo" mode and a later activation that can be easily performed by the customer executing a simple program. Application examples show that the additional hardware introduced for protection and/or activation has no effect over the performance, and showing an assumable area increase.

Keywords: Microprocessor, IP Cores, Intellectual Property Protection, FPGAs, Hardware Activation.

1. INTRODUCTION

The new design strategies based in the reuse of IP-cores[1] enables the optimization of company resources due to reduced development time and costs, especially when implementing highly complex systems. This is of special interest in the Field-Programmable Logic (FPL) domain. However, this design methodology introduces risks concerning the intellectual property protection (IPP) of those cores, with most forms of protection in the EDA industry being difficult to translate to this domain. Nowadays, IP core watermarking[2,3,4] has emerged as a tool for IP core protection. Although watermarks may be inserted at different levels of the design flow, watermarking Hardware Description Language (HDL) descriptions[5,6] has been proved to be a robust and secure option. The embedding of a watermark at this design level provides the most tampering resistant schemes since the signature is embedded in preliminary stages, and it is dragged through the whole design flow[6]. In this sense, IPP@HDL[6] procedure provides a protection framework for IP cores by spreading a digital signature at the HDL design level through memory structures or combinational logic included in the design.

In this design environment, microprocessor and microcontroller cores are widely used and are, in most cases, the basis to implement complex systems. The protection of these cores has become a priority for the developers. Although IPP@HDL can be applied directly to protect microprocessor (µP) based designs, some issues related to extracting the digital signature must be considered:

- In µP-based systems, the data bus is not always part of the I/O connections and it can be difficult to introduce the sequence to initiate the watermark extraction. The RESET pin can be an option to overcome this drawback.

Independent Component Analyses, Wavelets, Neural Networks, Biosystems, and Nanoengineering VIII, edited by Harold H. Szu, F. Jack Agee, Proc. of SPIE Vol. 7703, 77030L · © 2010 SPIE · CCC code: 0277-786X/10/$18 · doi: 10.1117/12.850526

- The data output of the microprocessor is usually connected to an I/O controller, making difficult the verification of the digital signature because the absence of direct access to output pins.

These considerations and the high number of IP cores including microprocessors and microcontrollers justify the development of an extension of IPP@HDL suitable for the protection of such systems. This paper presents a modification of IPP@HDL for µP based systems, maintaining the features of low area impact and non-relevant overhead for the protected system. The method, named µIPP@HDL, presents the same structure than IPP@HDL, providing robust watermarking protection, an easy and non-destructive procedure for signature extraction and adds the new feature of "Hardware Activation" that allows the distribution of "demo" versions of protected cores.

2. PROTECTION USING WATERMARKING STRATEGIES

The protection of IP cores using watermarking involves two processes: watermark embedding and watermark extraction. The watermark may contain information about ownership or user identity. The watermark has to be nearly invisible to human and machine inspection and must not interfere with the design's functionality. In this field, several digital watermarking techniques have been proposed with different contributions, each categorized according to watermark application levels as follows:

- *Physical level watermarks*. A basic idea is to store the watermark in some of the unused configurable logic blocks (CLB) of the FPGA. This watermarking technique, in combination with "tiling" algorithms, is used in[7]. In this approach, the design is divided into a set of tiles with different logic block configurations. Signature embedding by means of time constrains[8] consists of specifying timing constraints in paths different from the "critical path". All of these techniques, as well as those proposed in[4,9,10], embed watermarks at the physical level.
- *High level watermarks*. The protection at high design levels[6,11,12,13] introduces great difficulties to remove the watermark, since the signatures is embedded in preliminary stages, being dragged through the whole design flow. In addition, the watermark could be embedded as a functional part of the design[6].

IPP@HDL[6] protects digital systems by spreading the bits of a digital signature through memory structures or combinational logic included in the high-level description of the design. Thus, the signature is propagated through the whole design flow down to the physical implementation, independently of the target technology (ASIC, FPGA, etc.). This signature spreading does not require additional system resources. In addition, the proposed watermarking technique includes an easy and secure procedure[6] for non-destructive signature extraction. This procedure requires some hardware to be included into the system, which will detect the petition for signature extraction and will perform this task, showing the signature bits as a data sequence at the output of the protected system. This watermarking technique makes the signature bits to be part of the original design, while the system itself extracts the signature bits when it is required to do so. This process is activated[6] by feeding the system with a predetermined Signature Extraction Sequence (SES), which may be either manually selected or generated with an LFSR. Thus, the protected system includes minimum modifications in order to detect this SES and consequently extract the signature. Concretely, once the SES has been detected, the signature extraction additional logic addresses or applies proper input patterns to every module of the system where signature blocks have been embedded and/or hosted, instead of those being applied during normal operation of the hardware. This additional hardware has also to conveniently route the output of these modules to the circuit output[6]. In this way, the circuit keeps working following its normal operation but, during a few clock cycles, the system output consists of the digital signature bit blocks. The digital signature is then obtained grouping these signature bit blocks properly and is ready for whatever validation it is required. Recently, this watermarking scheme has been extended with the development of an automated tool for signature extraction[14].

Therefore, IPP@HDL can be applied to any digital system, with the assumption that it read data at the input pins and presents the results at a set of output pins. The SES is entered at the inputs of the system and the extracted digital signature is observed at the output pins. This scenario is slightly different in µP-based systems: generally, the input data for a µP are stored in memory and the results are also stored in memory. Accordingly, it is difficult to introduce the SES using the input pins of the system, and to recover the digital signature from any output pins. Thus, in the next section some modifications for the IPP@HDL are proposed in order to enable the IP protection of µP-based systems.

3. PROTECION OF µP CORES

The underlying idea for adapting IPP@HDL to the protection of µP cores consists of introducing the SES by means of a byte sequence stored in memory, and recovering the extracted signature through the system's memory. There are two main approaches to accomplish these objectives,

- *Introduce a signature co-processor*: this implies the introduction of additional circuitry to scan the RAM memory in order to detect the SES, and then, proceed with the extraction of the signature, storing it in a reserved memory location.
- *Modify the µP,* extending the instruction set in order to perform the signature extraction.

These two alternatives are analyzed in the following.

3.1 Signature co-processor

In this approach, the SES is introduced into the memory using the standard peripherals provided by the system under protection, and the coprocessor scans the RAM memory in order to detect the SES. When SES is detected, the co-processor interrupts the µP, perform the signature extraction and stores digital signature in a prefixed memory location. The main advantages of this approach are:

- It can be applied to any µP, since the design is independent of the system under protection.
- The coprocessor works in parallel with the µP, and the impact on performance is negligible.

As drawbacks, the following issues stand out:

- Profuse additional circuitry is required in order to scan the memory without collision problems, thus important impact on area may occur.
- The coprocessor is an independent entity, so it is more vulnerable to attacks than if the protection is embedded into the µP.

3.2 Extension of the instruction set

The other approach considered consists of extending the instruction set of the µP taking advantage of unused opcodes. This method is in line with the protection scheme provided by IPP@HDL, because the protection is hosted inside the core. Thus, any attempt to modify or remove the signature will affect the correct functioning of the system, providing high invulnerability properties. The introduction of additional instructions must not have serious effects in performance, and the additional hardware needed for extraction is affordable. Regarding the drawbacks, this approach requires a specific design solution for every µP family under protection. However, it has not to be an obstacle if the changes to perform in the HDL code of the µP are not too complex.

This approach has been the preferred to be implemented into µIPP@HDL. The flow diagram to introduce a digital signature in a µP core by extending the instruction set is presented in Fig. 1, and it can be summarized as follows:

1) *Generation of the digital signature*. A hash cryptographic function, generally MD5 or SHA1, is applied to a public document containing the author of the core, the client, and the license agreement. This hash will be used as digital signature.
2) *Introduction of the digital signature*. The digital signature is embedded into the core under protection by introducing new instructions in the µP. Unused opcodes are selected to perform this operation and these new instructions will be used to extract the signature.
3) *Extraction of the signature*. When the signature needs to be extracted, the new instructions added to the instruction set of the µP have to be executed. Thus, a program needs to be stored into memory and run. This will be the SES in this protection scheme. The extracted signature is stored in memory locations specified in the extraction program.

With this scheme it is possible to claim the authorship of µP cores, in the same way of IPP@HDL. Additionally, taking advantage of the programming possibilities offered by µPs, new features can be enabled. Concretely, introducing the concept of hardware activation, it is feasible to distribute cores in “demo” mode for demonstration purposes and later “activate” the core to obtain full functionality. This interesting feature is also included in µIPP@HDL and provides a valuable tool for IP core developers to release their designs. The next section describes this issue.

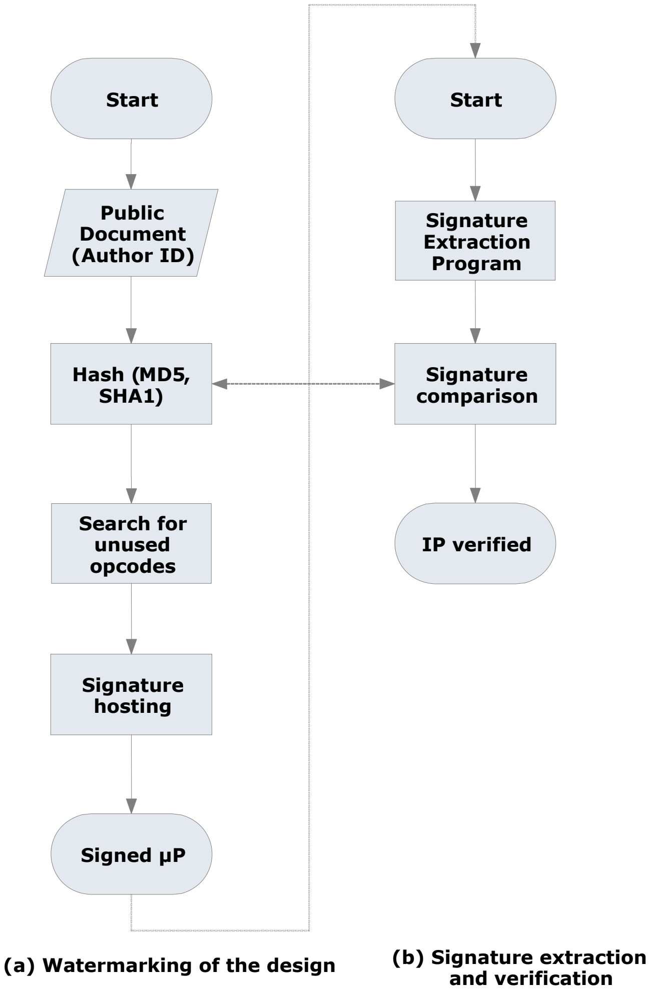
Start
Public Document (Author ID)
Hash (MD5, SHA1)
Search for unused opcodes
Signature hosting
Signed µP
Start
Signature Extraction Program
Signature comparison
IP verified
(a) Watermarking of the design
(b) Signature extraction and verification

Fig. 1 µP IP core Protection extending the instruction set.

4. HARDWARE ACTIVATION

The concept of hardware activation is based on the well known activation process of software applications[15]. However, some important differences in characteristics of software and hardware must bear in mind:

- Software applications have not restrictions in size due to low cost of massive memory devices. In hardware systems, the area is a critical design factor.
- Nowadays, all computers are connected to internet, so software applications can interact with the licenser to perform activation process. Hardware cores are not usually connected, thus activation has to be carried out by the client.

In this way, the proposed flow for the activation protection scheme is presented in Fig. 2 and explained below:

1) Generation of the activation number. The activation code is generated and embedded into the core.
2) Modification of the core. The µP core is modified in two ways:
    - New instructions are added to the µP core in order to detect the activation code. This task is carried out in a similar way than the digital signature introduction process above.
    - Some features of the µP core are disabled until the activation code is checked. The core remains in "demo" mode until the activation process is accomplished. The activation needs to be completed in every power-on of the system. Thus, the activation code must be included in the ROM or firmware of the system developed by the customer.
3) Activation. The activation process is performed by executing new instructions added to the µP. These instructions take the bytes of the activation code and compare them with the internal values embedded into the core. If the code is correct, the system is activated and all functionalities of the µP are enabled. Otherwise, the µP remains in "demo" mode.

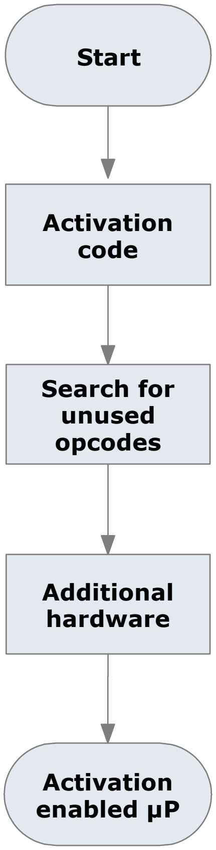

Fig. 2 µP hardware activation flow.

5. µIPP@HDL

The µIPP@HDL protection framework combines the IP protection provided by embedding a digital signature using watermarking techniques, and the distribution of limited versions of the core with the possibility of an easy activation for full functionality. These aspects have been detailed in previous sections and are based in the extension of the instruction set of the microprocessor to host the signature, facilitate the signature extraction and perform hardware activation. As in IPP@HDL, the protection is introduced at the high-levels of description, being dragged through the entire design flow. For the designer, the main work resides in the modification of the instruction set state machine in order to introduce the new instructions. When this task is completed, the protection of an individual core is reduced to changing some parameters in an HDL file. In the next section, a detailed example shows a practical application to protect a Z80[16] clone µP core, with Fig. 3 showing the general flow of µIPP@HDL.

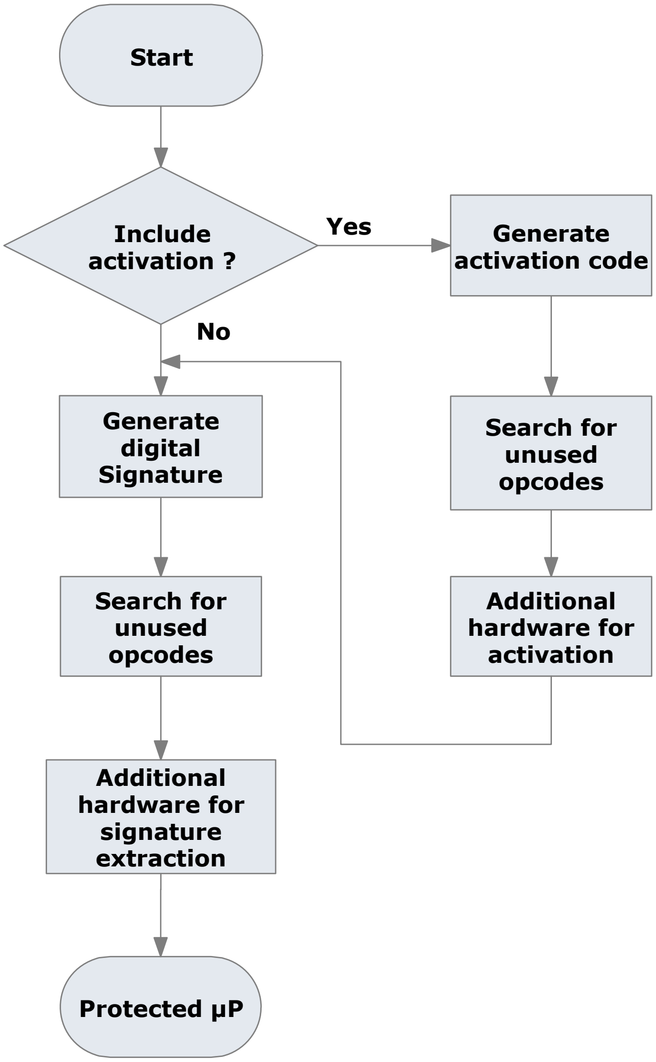
Start
Include activation ?
Yes
Generate activation code
No
Generate digital Signature
Search for unused opcodes
Search for unused opcodes
Additional hardware for activation
Additional hardware for signature extraction
Protected µP

Fig. 3 µIPP@HDL general flow diagram.

6. DESIGN EXAMPLE

To illustrate the protection scheme offered by µIPP@HDL and evaluate the impact on performance and area due to the additional hardware, a design example has been developed using a Z80 clone core named T80[17]. The µP has been implemented on a Virtex 5 device by Xilinx, using ISE 10.1 tools. For the IP protection, a MD-5 digital signature was hosted by extending the instruction set of the targeted µP. Four new instructions were created, with opcodes ED20, ED21, ED22 and ED23. Table 1 shows the name, opcode and operations performed by these instructions. SIG1 extracts four 8-bit blocks (BL01, BL02, BL03 and BL04) of the signature, and stores them in consecutive memory addresses beginning by the one pointed by the HL register pair. SIG2, SIG3 and SIG4 operate in a similar way. Thus, the signature extraction can be performed by executing the program listed in Fig. 4(a). Fig. 5 shows the extraction process for the signature C2DA0F795D03238FBD72AB3051000F81 in a post place&route simulation for a Xilinx Virtex 5 device[18] (xc5vlx30-1ff676). In the figure the 'C2'value in the data bus "d" can be observed when the content in address '0100' is read ("a" signal), 'DA' in address '0101' and so on.

The hardware activation feature has been achieved introducing also four new instructions (to handle a 64-bit activation code), which compare the activation blocks addressed by the HL register pair with the embedded ones. The activation is completed only if all the activation instructions are executed and all comparisons are successful. Fig. 4(b) shows the activation program, and Fig. 6 shows a simulation of the activation process. The signal "activacion" has value '1' when comparing activation blocks, and "activado" takes '1' value at the end of the simulation because all the comparisons were successful.

```
  Opcode     Mnemonic          Comments

  21  00  01:  LD  HL,$0100  ;      Memory  address  for
extraction
ED20:     SIG1          ; extraction of BL01 to BL04 blocks
ED21:     SIG2          ; extraction of BL05 to BL08 blocks
ED22:     SIG3          ; extraction of BL09 to BL12 blocks
ED23:     SIG4          ; extraction of BL13 to BL16 blocks
```

```
Opcode     Mnemonic          Comments

21 00 01: LD HL,$0100 ; address for activation code
36 A1:      LD(HL),$a1   ;  store the first activation block in memory
23:           INC HL        ;  next address
36B2        LD (HL),$b2   ;  store the second block
….                     ;  complete the store of the 64-bit activation code
ED30:     ACT1        ; check activation blocks BLA01 and BLA02
ED31:     ACT2        ; check activation blocks BLA03 and BLA04
ED32:     ACT3        ; check activation blocks BLA05 and BLA06
ED33:     ACT4        ; check activation blocks BLA06 and BLA07
```

Fig. 4. (a) Z80 program for MD5 signature extraction (b) Z80 program for hardware activation

Table 1. Instructions for signature extraction

Opcode	Name	Function
ED20	SIG1	(HL) ← BL01; HL←HL+1 (HL) ← BL02; HL←HL+1 (HL) ← BL03; HL←HL+1 (HL) ← BL04; HL←HL+1
ED21	SIG2	(HL) ← BL05; HL←HL+1 (HL) ← BL06; HL←HL+1 (HL) ← BL07; HL←HL+1 (HL) ← BL08; HL←HL+1
ED22	SIG3	(HL) ← BL09; HL←HL+1 (HL) ← BL10; HL←HL+1 (HL) ← BL11; HL←HL+1 (HL) ← BL12; HL←HL+1
ED23	SIG4	(HL) ← BL13; HL←HL+1 (HL) ← BL14; HL←HL+1 (HL) ← BL15; HL←HL+1 (HL) ← BL16; HL←HL+1

Table 2. Instructions for activation

Opcode	Name	Function
ED30	ACT1	BLA01 ↔ (HL) ; HL←HL+1 BLA02 ↔ (HL) ; HL←HL+1
ED31	ACT2	BLA03 ↔ (HL) ; HL←HL+1 BLA04 ↔ (HL) ; HL←HL+1
ED32	ACT3	BLA05 ↔ (HL) ; HL←HL+1 BLA06 ↔ (HL) ; HL←HL+1
ED33	ACT4	BLA07 ↔ (HL) ; HL←HL+1 BLA08 ↔ (HL) ; HL←HL+1

Table 3 presents the synthesis results for four versions of the T80 core. The first line corresponds to the unmodified T80, the second one to a T80 with a 128-bit digital signature for IP protection, the third one is a T80 with activation capability: in "demo" mode the CALL instruction is disabled; when activated, the full instruction set is available. The fourth one includes all the features of µIPP@HDL, IP protection and hardware activation.

Performance has been evaluated in terms of the maximum frequency, and the hardware introduced for signature extraction and hardware activation have no significative effects over this parameter. For area results, two parameters have been considered: the number of slice registers and the number of slice LUTS[18]. In both of them, assumable increments are required for the additional circuitry.

Table 3**.** Synthesis Results

µP	Slice Registers	Slice LUTs	Max. Fec.
T80	238	1412	104 Mhz
T80 with MD5 signature	249	1678	105 Mhz
T80 with 64-bit activation	248	1577	106 Mhz
T80 with MD5 and 64-bit activation	257	1628	107 Mhz

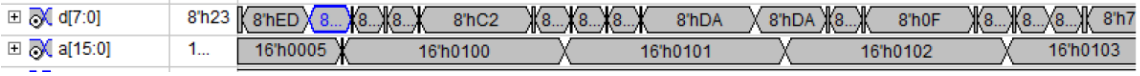

Fig. 5 Digital signature extraction in a T80 protected core.

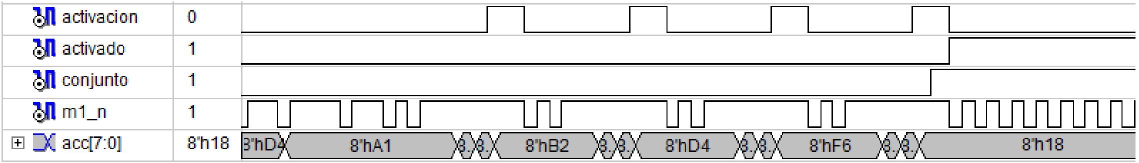

Fig. 6. Activation process in a T80 with hardware activation feature.

7. CONCLUSIONS

A new framework for the protection of µP cores has been presented. The protection scheme is derived from the IPP@HDL procedure and it has been adapted to the singularities of µP cores, overcoming the problems for digital signature extraction in such systems. Additionally, the feature of hardware activation has been introduced, allowing the distribution of µP cores in a "demo" mode and a later activation that can be easily performed by the customer by executing a simple program. A design example has been presented for a Z80 class core, showing that the additional hardware introduced for protection and/or activation has no effect over the performance, and showing an assumable area increase.

8. ACKNOWLEDGEMENTS

This work was partially funded by project TEC2007-68074-C02-01/MIC (Plan Nacional I+D+I, Spain). CAD tools and supporting material were provided by Altera Corp. trough University Program agreements. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsors.

REFERENCES

[1] Keating M. and Bricaud, P. [Reuse Methodology Manual for System-on-a-Chip Designs], Kluwer Academic Publishers (1998).

[2] Cox, I., Miller, M. and Bloom, J., [Digital Watermarking: Principles & Practice], Morgan Kaufmann (2001).

[3] Charbon E. and Torunoglu I., "Watermarking techniques for electronic circuit designs," Lecture Notes in Computer Science, vol. 2613, 147–169 (2003).

[4] Kahng, B., Lach, J., Mangione-Smith, W. H., Mantik, S., Markov, I. L., Potkonjak, M., Tucker, P., Wang, H. and Wolfe, G. "Watermarking techniques for intellectual property protection," in Proc. of the Design Automation Conference, 776–781 (1998).

[5] Houshanfar, F., Hong, I. and Potkonjak, M., "Behavioral synthesis techniques for intellectual property protection," ACM Transactions on Design Automation of Electronic Systems, vol. 10, no. 3, 523–545 (2005).

[6] Castillo, E., Meyer-Baese, U., García, A.. Parrilla, L. and Lloris, A., "IPP@HDL: Efficient intellectual property protection scheme for IP cores," IEEE Trans. VLSI Syst., vol. 15, no. 5, 578–591 (2007).

[7] Lach, J., Mangione-Smith, W.H., Potkonjak, M., "Fingerprinting Techniques for Field Pgroammable Gate Array Intellectual Property Protection," *IEEE Transactions on CAD*, vol. 20, no. 10 , 1253-1261. (2001).

[8] Jain, A.K., Yuan, L., Pari P.R. and Qu, G., "Zero Overhead Watermarking Technique for FPGA Designs," *Proc. Of Great Lakes Symposium on VLSI*, 147-152 (2003).

[9] Newbould, R.D., Irby, D.L., Carothers, J.D., Rodriguez, J.J., Holman, W., "Watermarking ICs for IP protecion," *Electronics Letteres*, vol. 38, no. 6, pp. 272-274 (2002).

[10] Kahng, A.B., Mantik, S., Markov, I.L., Potkonjack, M., Tucker, P., Wang, H. and Wolfe, G., "Robust IP Watermarking Methodologies for Physical Desing," *Proc. Of the 35th Design and Automation Conference,* 782-787 (1998).

[11] Hong, I. and Potkonjak, M., "Behavioral Synthesis Techniques for Intelecctual Property Protecion," *in Proc of the 36th Design Automation Conference,* 849-854 (1999).

[12] Fan, Y.C. and Tsa, H.W., "Watermarking for intellectual property protection," *Electronics Letters,* vol. 39, No. 18, 1316-1318, (2003).

[13] Charbon, E. and Torunoglu, I., "Watermarking techniques for electronica circuit design," *in Lecture Notes on Computer Science*, vol. 2613, 147-169 (2003).

[14] Castillo, E., Parrilla, L., García, A., Meyer-Baese, U., Lloris, A. and Botella, G., "Automated signature insertion in combinational logic patterns," in Proc. of 4th Southern Conference on Programmable Logic, 183–186 (2008)

[15] Richardson, III, [System for software registration], United States Patent No. 5,490,216. (2006).

[16] Gaonkar, Ramesh S. [The Z80 microprocessor: architecture, interfacing, programming, and design], 3rd ed., Ed. Upper Saddle River: Prentice-Hall (2001).

[17] Warner, D. T80 cpu [Online]. Available: http://www.opencores.org/?do=project&who=t80

[18] Xilinx, Inc., Virtex-5 User Guide. [Online]. Available: http://www.xilinx.com/support/documentation/userguides/ug190.pdf