UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

Modulación por ancho de pulso de alta resolución en FPGA

Mario González Ermakov
Tutor: Alberto Sánchez González
Ponente: Ángel de Castro Martín

Junio 2019

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

AUTOR: Mario González Ermakov

TUTOR: Alberto Sánchez González

Trabajo realizado en el grupo

Hardware & Control Tecnology Laboratory

Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2019

Resumen

En este Trabajo Fin de Grado se estudiarán diferentes técnicas de modulación por ancho de pulso (PWM) de alta resolución utilizando para ello una FPGA de la familia 7 de Xilinx. En particular se explorarán diferentes métodos para crear DPWM (*Digital Pulse Width Modulation*) de alta resolución. En la actualidad este tema es objeto de numerosas investigaciones debido a la importancia que tiene en las ya asentadas y comúnmente utilizadas técnicas de conversión conmutada en la electrónica de potencia.

Dado que el control de convertidores de potencia conmutados suele realizarse mediante controladores digitales, su actuación también es digital, estando limitada en resolución por el bloque que genera la señal PWM (de la que depende la precisión de la tensión de salida). Sin embargo, los avances en los dispositivos semiconductores han permitido el desarrollo de nuevas tecnologías de semiconductores de banda ancha como el nitruro de galio, y el carburo de silicio, alcanzando frecuencias de conmutación muy altas (alrededor de decenas de megahercios). Dichas frecuencias requieren que la resolución en la actuación sea muy fina para poder aprovechar las ventajas de dichos dispositivos.

Todos los métodos analizados en este Trabajo Fin de Grado están diseñados en dos bloques bien diferenciados. Uno síncrono basado en un contador y un comparador, y otro asíncrono que será el encargado de obtener esa resolución adicional, y cuya arquitectura provee un mecanismo de retardo. Este mecanismo de retardo tendrá un elemento central en cada método analizado. En los dos primeros métodos este elemento será el Clocking Wizard, y en el tercer método será el IODELAY. Ambos componentes disponibles en las FPGAs de la familia 7 de Xilinx.

Posteriormente se realizarán medidas experimentales de cada método implementado sobre la FPGA, se compararán con las medidas que se obtendrían en un entorno ideal, y se llegará a unas conclusiones sobre las ventajas e inconvenientes que supondría integrar una arquitectura u otra en un sistema de control.

Abstract

This Bachelor Thesis will study different high resolution PWM (Pulse Width Modulation) techniques using an FPGA of the 7 Series of Xilinx. In particular, different methods will be explored to create high-resolution DPWM (*Digital Pulse Width Modulation*). Nowadays, this matter is under numerous investigations due to the relevance that it has in the already settled and commonly used switching conversion techniques in power electronics.

Since the switching power converters control is usually done through digital controllers, its actuation is also digital, being the resolution limited by the block that generates the PWM signal (on which depends the output voltage accuracy). However, Advances in semiconductor devices have allowed new high band semiconductor technologies development like Gallium nitride and Silicon carbide, reaching very high switching frequencies (around tens of megahertzs). These frequencies require that the actuation resolution be very fine to take advantage of these devices.

All the methods analyzed in this Bachelor Thesis are designed in two well-differentiated blocks. One synchronous block based on a counter and a comparator, and another asynchronous block that will be responsible of obtaining that additional resolution, and whose architecture provides a delay mechanism. This delay mechanism will have a central element in each analyzed method. In the first two methods this element will be the Clocking Wizard, and in the third method it well be the IODELAY. Both components are available in the 7 Series FPGA of Xilinx.

Later, experimental measurements of each method implemented on the FPGA will be made, they will be compared with the measures that would be obtained in an ideal environment, and some conclusions about advantages and disadvantages that would imply integrating one architecture or another in a control system will be reached.

Palabras clave

DPWM, FPGA, VHDL, alta resolución, Clocking Wizard, IODELAY, ciclo límite, control digital, convertidores conmutados.

Keywords

DPWM, FPGA, VHDL, high resolution, Clocking Wizard, IODELAY, limit cycle, digital control, switching converters.

Agradecimientos

Gracias a Alberto, mi tutor, por su inestimable ayuda, por todas esas horas en el laboratorio resolviendo todas mis dudas. Gracias también a Ángel por ofrecerme la posibilidad de realizar este TFG.

Gracias también a todos esos compañeros que en un momento u otro han aportado su granito de arena y han amenizado todas esas clases de teoría y prácticas de laboratorio.

Y gracias también a mis padres por apoyarme cada día y por guiarme en el camino para que no descarrile, dándome una educación y enseñándome unos valores que estoy orgulloso de haber recibido.

INDICE DE CONTENIDOS

1 Introducción 1
1.1 Motivación 1
1.2 Objetivos 1
1.3 Organización de la memoria 2
2 Estado del arte 3
2.1.1 PWM 4
2.1.2 Clocking Wizard 6
3 Métodos de alta resolución en PWM 7
3.1 Método 1 8
3.1.1 Mejora del método 1: “reducción del duty cycle del reloj” 13
3.2 Método 2 14
3.3 Método 3 20
4 Integración, pruebas y resultados 25
5 Conclusiones y trabajo futuro 33
5.1 Conclusiones 33
5.2 Trabajo futuro 34
Referencias 35

INDICE DE FIGURAS

FIGURA 2-1: CICLO LÍMITE [4]..... 4

FIGURA 2-2: SEÑAL PWM..... 5

FIGURA 3-1: SEÑAL DE SALIDA DEL BLOQUE SÍNCRONO..... 7

FIGURA 3-2: CONFIGURACIÓN CLOCKING WIZARD, MÉTODO 1..... 8

FIGURA 3-3: ARQUITECTURA DEL MÉTODO 1..... 9

FIGURA 3-4: SEÑAL QUARTERCYCLE..... 9

FIGURA 3-5: COMPORTAMIENTO IDEAL DEL MÉTODO 1 PARA *D[1,0] = "11"*..... 10

FIGURA 3-6: COMPORTAMIENTO NO IDEAL DEL MÉTODO 1 PARA *D[1,0] = "11"*..... 10

FIGURA 3-7: ARQUITECTURA SOLUCIÓN AL MÉTODO 1: FLIP-FLOPS Y REGISTRO INTERMEDIO..... 11

FIGURA 3-8: DIAGRAMA DE SOLUCIÓN AL MÉTODO 1: FLIP-FLOPS Y REGISTRO INTERMEDIO. *D[1,0] = "01"*..... 12

FIGURA 3-9: DIAGRAMA DE SOLUCIÓN AL MÉTODO 1: FLIP-FLOPS Y REGISTRO INTERMEDIO. *D[1,0] = "11"*..... 12

FIGURA 3-10: ARQUITECTURA SOLUCIÓN AL MÉTODO 1: REDUCCIÓN DUTY CYCLE DEL RELOJ..... 13

FIGURA 3-11: SEÑAL QUARTERCYCLE CON REDUCCIÓN DEL DUTY CYCLE DEL RELOJ..... 14

FIGURA 3-12: CONFIGURACIÓN CLOCKING WIZARD PARA EL MÉTODO 2..... 14

FIGURA 3-13: ARQUITECTURA DEL MÉTODO 2..... 15

FIGURA 3-14: ESQUEMA CLOCKING WIZARD, MÉTODO 2..... 16

FIGURA 3-15: DIAGRAMA PROTOCOLO DYNAMIC PHASE SHIFTING [15]..... 16

FIGURA 3-16: DIAGRAMA DE ESTADOS DEL PROTOCOLO DE DYNAMIC PHASE SHIFTING..... 17

FIGURA 3-17: DIAGRAMA DEL MÉTODO 2..... 17

FIGURA 3-18: DIAGRAMA DEL MÉTODO 2 CON PROBLEMA DE FASES CERCANAS A 180º..... 18

FIGURA 3-19: ARQUITECTURA SOLUCIÓN AL MÉTODO 2..... 19

FIGURA 3-20: DIAGRAMA DE SOLUCIÓN DEL MÉTODO 2..... 19

FIGURA 3-21: ESQUEMA SIMPLIFICADO IDELAYE2..... 20

FIGURA 3-22: ESQUEMA SIMPLIFICADO IDELAYCTRL..... 21

FIGURA 3-23: ARQUITECTURA DEL MÉTODO 3. 22

FIGURA 3-24: DIAGRAMA MÉTODO 3: RETRASO MENOR A MEDIO CICLO DE RELOJ (BIT D[5] = '0'). 23

FIGURA 3-25: DIAGRAMA MÉTODO 3: RETRASO MAYOR A MEDIO CICLO DE RELOJ (BIT D[5] = '1'). 23

FIGURA 4-1: OSCILOSCOPIO. 26

FIGURA 4-2: RESULTADOS DEL MÉTODO 1. 27

FIGURA 4-3: RESULTADOS SOLUCIÓN MÉTODO 1: REDUCCIÓN DUTY CYCLE DE RELOJ. 27

FIGURA 4-4: RESULTADOS DEL MÉTODO 2. 28

FIGURA 4-5: RESULTADOS DEL MÉTODO 3. 28

FIGURA 4-6: CONFIGURACIÓN FPGA, MÉTODO 1. 30

FIGURA 4-7: CAMINO DE LA SEÑAL DE SET ANTES DE LA REUBICACIÓN DEL *FLIP-FLOP*. 31

FIGURA 4-8: CAMINO DE LA SEÑAL DE SET TRAS REUBICACIÓN DEL *FLIP-FLOP*. 31

FIGURA 4-9: RESULTADOS DEL MÉTODO 1 ANTES Y DESPUÉS DE REUBICACIÓN COMPONENTES. .. 32

INDICE DE TABLAS

TABLA 4-1: RESOLUCIONES OBTENIDAS. 32

TABLA 4-2: RECURSOS UTILIZADOS. 32

1 Introducción

1.1 Motivación

El expandido uso de fuentes de alimentación y convertidores basados en técnicas de conmutación, frente a los antiguos e ineficientes convertidores basados en resistencias, requiere de un control, que ha evolucionado desde lo analógico hasta lo digital, que es el que está presente a día de hoy. El control digital tiene muchas ventajas tales como: la flexibilidad, pues el funcionamiento es implementado mediante software reprogramable, por lo que no hace falta realizar cambios en el hardware (que supondría un notable aumento de costes); fiabilidad, debido al uso de circuitos integrados aislados de variaciones de las condiciones ambientales y por tanto del envejecimiento de los chips; simple diseño y prototipado; entre otras. Sin embargo, estas ventajas del control digital se enfrentan a dos obstáculos que no estaban presentes en el control analógico: la pérdida de resolución "infinita", debido a la discretización en niveles de los valores de la señal analógica; la introducción de retrasos por muestreo y procesamiento en orden de conseguir lo anterior.

Una de las técnicas de modulación en el control de los interruptores de un convertidor conmutado es la conocida como DPWMs (*Digital Pulse Width Modulation*), que consiste en generar una señal digital de frecuencia fija, pero cuyo ancho de pulso (porcentaje de tiempo a '1' frente al periodo total) se varía. De esa forma, en función del ancho de pulso se logra entregar la potencia demandada con una eficiencia muy alta. Usualmente, esta técnica se desarrollaba mediante técnicas basadas en el empleo de contadores que trabajan a la frecuencia de reloj del sistema, y que, por tanto, era la que limitaba la resolución máxima de la señal PWM. Ante la actual llegada de nuevas tecnologías que exigen un aumento en la frecuencia de conmutación de los circuitos electrónicos, surge la necesidad de combatir este límite de resolución. Por consiguiente, en este TFG son objeto de estudio nuevas técnicas de implementación DPWM para cumplir este propósito. Para su desarrollo es interesante el uso de una FPGA (*Field-Programable Gate Array*), pues es un entorno reprogramable, con una capacidad de procesamiento muy rápida, un coste reducido, proceso de diseño más sencillo, circuitería más simple, etc.

1.2 Objetivos

El objetivo de este Trabajo de Fin de Grado es el de diseñar e implementar diferentes técnicas de PWM (*Pulse Width Modulation*, Modulación por ancho de pulso) mediante el uso de una FPGA. Se ha utilizado la FPGA de la familia Artix A7 de Xilinx.

- El objetivo principal es el de conseguir una resolución mayor a un ciclo de reloj de la FPGA, para cubrir la actual necesidad de aumento de resolución de los PWM en convertidores conmutados, pues son los que limitan su funcionamiento. En concreto se buscará lograr una resolución de al menos 4 veces mayor que el periodo del reloj.

- Buscar la mejor solución que logre un equilibrio entre:
    - Bajo uso de recursos.
    - Bajo precio.
    - Una buena linealidad.
    - Monotonía del sistema.
    - Rapidez en la obtención del ciclo de trabajo pedido.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Capítulo 1, Introducción**

  En este capítulo se expone la motivación de este TFG, situando el contexto en el que se desarrolla la materia. También se describen los objetivos que se esperan alcanzar.

- **Capítulo 2: Estado del Arte**

  Se expone el estado del arte, resaltando las limitaciones actuales del control digital y de las técnicas DPWM, estableciendo el punto de partida.

- **Capítulo 3: Métodos de alta resolución en PWM**

  En este capítulo se describen en detalle los distintos métodos estudiados indicando sus ventajas e inconvenientes.

- **Capítulo 4: Integración, Pruebas y Resultados**

  En este capítulo se muestran los resultados ideales y experimentales obtenidos mediante las distintas técnicas expuestas en el capítulo anterior, justificando las diferencias observadas entre unas y otras.

- **Capítulo 5: Conclusiones y Trabajo Futuro**:

  Aquí se expondrán las conclusiones sobre los resultados finales, realizando una pequeña comparativa entre los distintos métodos y recogiendo las ventajas de usar unos u otros en sistemas de control digital. Por último se establecerá los caminos hacia los que tenderían trabajos futuros relacionados con este tema.

2 Estado del arte

Las dos principales desventajas presentes en el control digital a día de hoy en el control digital son:

- El retardo de muestreo y de procesamiento
- Resolución limitada

La última, está generalmente producida por el ADC (*Analog to Digital Converter*) y por el PWM (*Pulse Width Modulation*). Sin embargo, el ADC supone un problema menor si aplicamos técnicas de enventanado. Entonces la mayor razón que limita la resolución es el PWM, pues la resolución del PWM debe ser mayor que la del ADC para evitar el ciclo límite.

Los ciclos límite (*limit cycle*) son oscilaciones periódicas de la señal de salida en régimen permanente que no se deben a la conmutación producida por el PWM, y que son indeseables tener (su amplitud y frecuencia son difíciles de predecir y es difícil analizar el ruido de la señal de salida y el EMI producido por el convertidor). En aplicaciones basadas en semiconductores de silicio, la frecuencia de funcionamiento de los controladores digitales es suficientemente alta para la correcta generación de señales de control. Sin embargo, con la irrupción de semiconductores basados en nitruro de galio y carburo de silicio, las frecuencias de conmutación se han podido elevar hasta decenas de megahercios, necesitando nuevas técnicas de control [1]. Por ejemplo, en [2] se puede ver un sistema que utiliza una frecuencia de conmutación de 40 MHz, y en [3] se consigue una frecuencia de 75 MHz.

Si se tienen N_{ADC} bits de resolución en el ADC, en el convertidor se tendrán niveles de discretización separados:

$$\frac{\mathrm{V_{in}}}{2^{\mathrm{N_{ADC}}}}$$

Y si se tienen $\mathrm{N_{pwm}}$ bits de resolución en el PWM Mientras que en el PWM se tendrán:

$$\frac{\mathrm{Vin}}{2^{\mathrm{N_{pwm}}}}$$

Como se ha mencionado antes, si se tienen menos bits de resolución en el PWM que en el ADC se tendrá ciclo límite, como se muestra en la Figura 2-1a. Esto es debido a que no existe ningún valor en el PWM que pueda hacer un mapeo con el nivel de voltaje que se quiere en el ADC, porque falta resolución. De esta manera nos encontraremos en una continua oscilación entre los niveles de PWM más cercanos al voltaje requerido. En la Figura 2-1b, tenemos más bits de resolución en el PWM, y se puede representar de manera exacta cualquier valor que pida el ADC, luego la salida no se verá afectada por el ciclo límite.

Por lo tanto, nos vemos obligados a estudiar técnicas que permitan aumentar esta resolución del PWM. En la actualidad, este tema está siendo objeto de numerosas investigaciones que se están llevando a cabo.

Muchas de las soluciones que se proponen consisten en estructuras híbridas, entre el uso de una parte síncrona compuesta de un contador, y una parte asíncrona que implementa líneas de retardo. Gracias al uso del contador se obtiene una buena linealidad, y gracias al uso de las líneas de retardo se obtiene una mayor resolución.

En este trabajo, para implementar ambos bloques se ha usado una FPGA pues resulta sencillo implementar las líneas de retardo con componentes que en la actualidad tienen casi todas las FPGA, que son los Clocking Wizard.

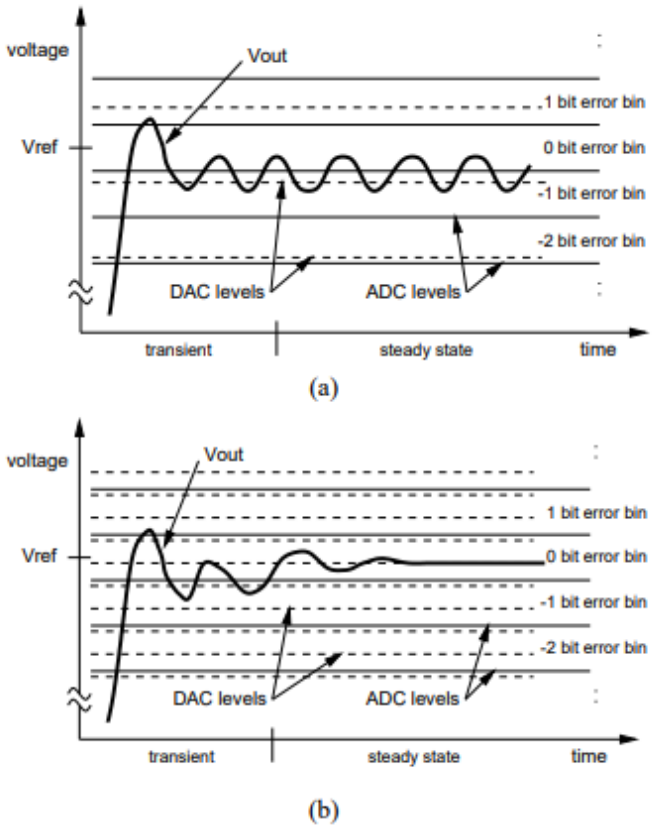

Figura 2-1: Ciclo límite [4].

2.1.1 PWM

La modulación por ancho de pulso es una técnica de modulación basada en la variación del ciclo de trabajo de una señal (ancho de pulso).

En la Figura 2-2, se pueden ver dos señales PWM con anchos de pulso distintos. La superior está activa un tiempo T_{ON} que dura la mitad de un periodo de duración T. Y la señal inferior está activa un tiempo T_{ON}, equivalente a un cuarto de T. En este contexto, se define el ciclo de trabajo como:

$$d = \frac{T_{ON}}{T}$$

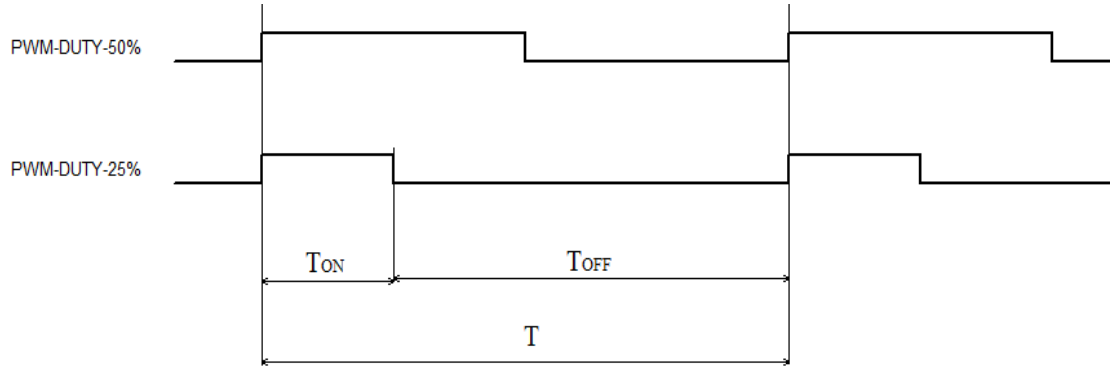

Figura 2-2: Señal PWM.

Donde t_{ON} es el tiempo en el que la señal está a un nivel lógico alto, y T es el periodo de la señal. En la señal de la figura de arriba este ciclo de trabajo es de 0,5 (50%) y en la de abajo es de 0,25 (25%).

Debido a que en este TFG se está estudiando la realización de DPWMs (PWM digitales) de alta resolución en FPGAs, la consigna de actuación al bloque DPWM es un número binario con un número limitado de bits. El formato usado en este TFG para la representación de esta variable será binario en coma fija, por lo que están compuestos por una parte entera y una parte decimal o fraccionaria. En este sentido, la parte entera representará el número de ciclos de reloj en los que la señal generada debe estar activa, mientras que los bits decimales indican las fracciones de ciclo de reloj en los que la señal de salida debe estar activa aparte del número de ciclos de reloj fijados en la parte entera. Si el ciclo de trabajo está representado por N_{duty} bits, los bits más significativos (MSB) indicarán la parte entera:

$$N_{ent} = N_{duty} - N_{dec}$$

Donde N_{ent} son los bits de parte entera y N_{ent} son los bits de parte decimal. El número de bits de la parte entera depende del periodo máximo que se necesite en la señal PWM generada, no habiendo limitaciones técnicas en esta parte. Sin embargo, el número de decimales, al estar representando fracciones de ciclo de reloj, están determinados por la resolución que se pueda conseguir utilizando las diferentes técnicas que se mostrarán en este TFG.

Normalmente los bloques PWM están implementados mediante un contador y comparador, lo que permite realizar la parte entera descrita anteriormente [5]. La parte entera, por tanto, es una parte síncrona que depende de la frecuencia máxima de reloj del dispositivo electrónico. Para la parte decimal es necesaria fracciones de ciclo de reloj, y por lo tanto puede considerarse como combinacional. En el caso de ASICs (*Application Specific Integrated Circuit*) esto se ha resultado en ocasiones con líneas de retardo [6]. Para FPGAs también se han propuesto diferentes soluciones [7]-[14]. Como se verá en este TFG, estas soluciones suelen estar basadas en un Clock Wizard, anteriormente llamado DCM (*Digital Clock Manager*).

2.1.2 Clocking Wizard

El Clocking Wizard es un componente que se encuentra en la mayoría de FPGA existentes en la actualidad. Con él es posible manipular la señal de reloj que conectamos a la entrada. Va a estar presente en el bloque asíncrono de todas las soluciones presentadas, y será fundamental en las dos primeras. En los diseños que se presentarán a continuación lo usaremos para:

- Multiplicar la señal de reloj del sistema, pues *el* Clocking Wizard te permite sintetizar un amplio rango de frecuencias.
- Crear una versión de la señal de reloj del sistema con una variación en su ciclo de trabajo.
- Crear una versión de la señal de reloj del sistema desfasada un cierto valor, pues implementa un DLL (*Delay Locked Loop*).

3 Métodos de alta resolución en PWM

Se van a describir tres métodos para obtener la señal PWM de alta resolución, analizando sus problemas y presentando algunas posibles soluciones a los mismos. En primer lugar, se verá un método que nos permitirá obtener una señal PWM basándose en la generación de cuatro señales desfasadas del reloj de entrada al Clocking Wizard y la posterior combinación de estas señales mediante puertas lógicas AND para activar la señal de reset de un *latch* de salida en el momento adecuado y obtener así el ancho de pulso deseado. El siguiente método permitirá obtener una resolución más alta mediante el uso de una característica que ofrece el Clocking Wizard llamada "*Dynamic Phase Shift*", que permitirá generar una señal con un desfase de reloj variable con la que se activará la señal de reset del *latch* antes mencionado. Finalmente, se presentará una última solución basada en un componente llamado IODELAY, el cual genera directamente una versión desfasada de su señal de entrada, sin necesidad de realizar el desarrollador un protocolo para este propósito como en el método anterior. El componente IODELAY es un bloque hardware que existe en las FPGAs de Xilinx de la familia 7. Los métodos 1 y 2 están basados en [12] y [7], respectivamente.

Todos los métodos que se van a presentar tienen una parte síncrona y otra asíncrona. En primer lugar, se va a describir la parte síncrona, pues es la común a todos los métodos, y la parte asíncrona de cada uno se irá describiendo en su correspondiente apartado.

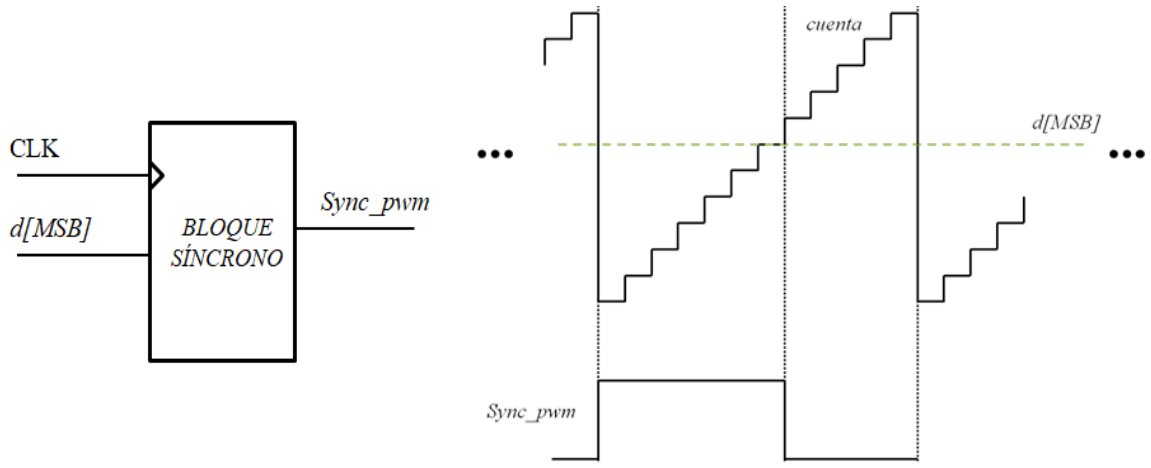

Figura 3-1: Señal de salida del bloque síncrono.

El bloque síncrono se fundamenta en el uso de un contador y un comparador. A la entrada del contador se conectarán los bits MSB de la señal que representa el ciclo de trabajo (*duty cycle*), a la que se llamará *d*. Mediante estos bits nos referiremos al número entero de ciclos de reloj que estará activa la señal de salida (*pwm*), mientras que con los bits LSB de *d*, nos referiremos a la fracción adicional de ciclo de reloj que *pwm* seguirá activa. El reloj que gobierna el contador será el del sistema. A la salida de este bloque, se tendrá una señal PWM cuadrada, que llamaremos *Sync-pwm*, cuyo ancho de pulso equivale al número de ciclos de reloj que indiquen los bits de parte entera de *d*. El proceso de generación de esta

señal es el siguiente: El reloj empieza la cuenta, la cual se incrementa en una unidad por ciclo, y mientras que este valor de cuenta no supere el valor que representen los bits de parte entera de *d*, se genera una señal a la salida que se mantiene en un nivel lógico alto. Cuando el comparador detecta que la cuenta ha alcanzado el valor antes mencionado, la señal a la salida cambia a un nivel lógico bajo. Este proceso se puede observar en la Figura 3-1. En este momento la señal *Sync_pwm* tiene una resolución de T_{clk}. Y a la salida del bloque asíncrono, la señal *pwm* tendrá una resolución de:

$$Resolución = \frac{T_{clk}}{\text{nº total de incrementos posibles del elemento retardante}}$$

Donde T_{clk} es el periodo de reloj del sistema. En este punto, se puede observar que para aumentar la resolución podríamos aumentar la frecuencia de reloj del sistema, pero esto no resulta siempre práctico, pues puede ser más caro, o simplemente porque la FPGA de la que se dispone no permite trabajar con un reloj de mayor frecuencia.

A partir de aquí se irá detallando cómo es el bloque asíncrono de cada diseño, que es precisamente el que nos va a permitir lograr una resolución mayor a un ciclo de reloj.

3.1 Método 1

Los bits decimales (LSB) de *d* se usarán para indicar qué fracción de ciclo queremos añadir a la señal *Sync-pwm*. Por ejemplo, suponiendo que la señal *d* es de 10 bits, de los cuales se usan 8 bits de parte entera y 2 bits de parte decimal. Si esta señal tiene un valor de "0000100110", se estará indicando que la señal de salida *pwm* tendrá un ancho equivalente a nueve ciclos de reloj más medio ciclo del mismo. Para conseguir esto, se usará el Clocking Wizard para generar 4 señales desfasadas del reloj del sistema, de valores 0º, 90º, 180º y 270º. En la Figura 3-2 se muestra la configuración de los relojes de salida de este componente en este método.

Output Clock	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)	
	Requested	Actual	Requested	Actual	Requested	Actual
clk_out1	100.000	100.000	0.000	0.000	50.000	50.0
clk_out2	100.000	100.000	90.000	90.000	50.000	50.0
clk_out3	100.000	100.000	180.000	180.000	50.000	50.0
clk_out4	100.000	100.000	270.000	270.000	50.000	50.0

Figura 3-2: Configuración Clocking Wizard, Método 1

Seguidamente estas señales se combinan mediante el uso de puertas AND de la manera en la que se muestra en la Figura 3-3, con el objetivo de obtener una señal que llamaremos *QuarterCycle*, cuyo ancho de pulso es de un cuarto de ciclo. Esta señal será la encargada de activar la señal de reset del *latch* de salida, y que se elegirá su desfase respecto al reloj de entrada en función del valor de *d[1,0]*, que será la señal de selección del multiplexor de la Figura 3-5.

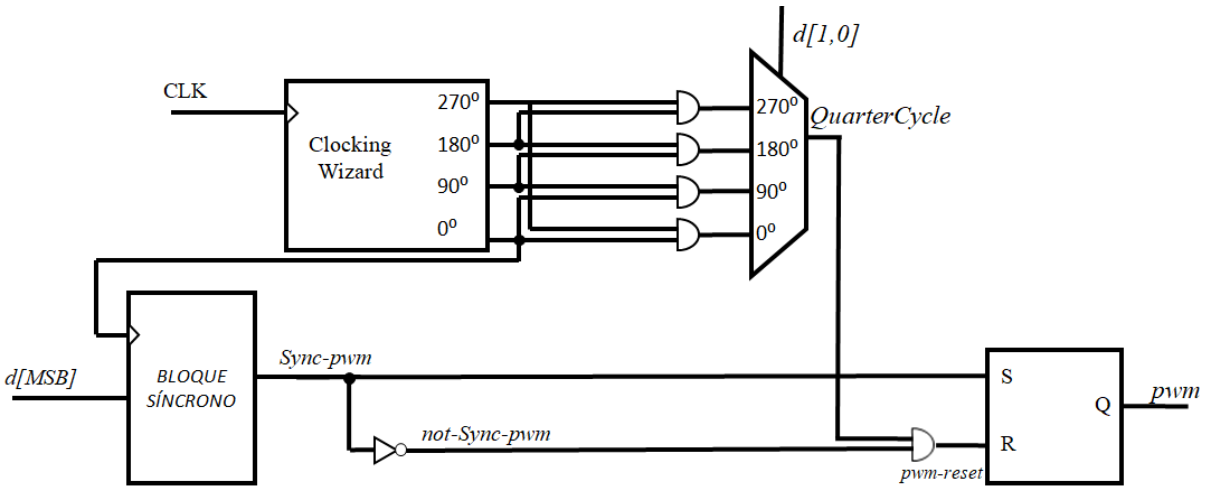

Figura 3-3: Arquitectura del Método 1.

Este diseño muestra un resultado correcto sobre una simulación de comportamiento teórico, sin ningún tipo de retrasos. El problema se presenta cuando se realiza una simulación no ideal, pues los retrasos introducidos por la lógica del diseño son significativos y provocan un incorrecto funcionamiento del sistema para el caso en que *d[1,0]* = "11".

Como se observa en la Figura 3-4**,** este pequeño retraso hace que la señal *QuarterCycle* esté a '1' durante una pequeña parte del ciclo siguiente, lo que produce el reseteo de la señal *pwm* antes de tiempo.

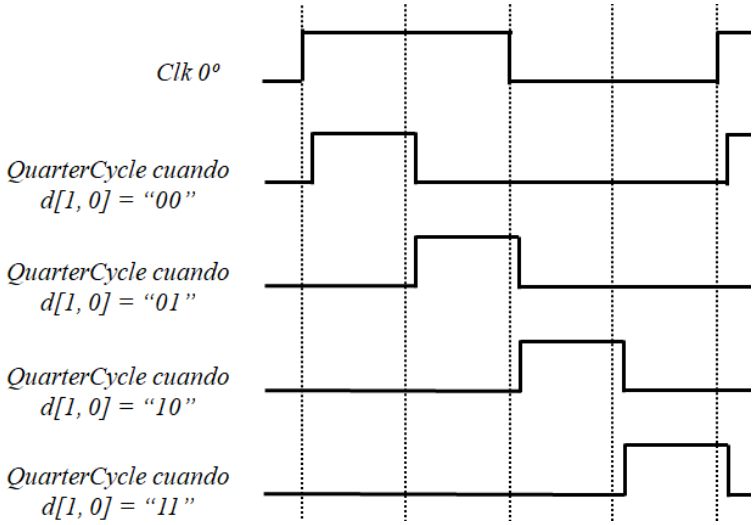

Figura 3-4: Señal QuarterCycle

En la Figura 3-5 se refleja el diagrama temporal de las principales señales este diseño para un comportamiento ideal, cuando *d[1,0]* = "11". Y en la Figura 3-6 se muestra el comportamiento que presenta este diseño en una simulación no ideal que incluye retrasos en las pistas del circuito. Como se observa en la Figura 3-6, cuando *not-Sync-pwm* vale '1', debido a los retrasos presentes en la lógica, *QuarterCycle* sigue valiendo '1' durante una pequeña parte del siguiente ciclo de reloj y consecuentemente la señal de Reset del *latch (HSM-reset)* se activa. De esta manera, la señal de salida en vez de seguir a '1' como indicaría la línea discontinua azul, se resetea prematuramente y el circuito termina comportándose como en el caso *d[1,0]* = "00", y consiguientemente el circuito no presenta un comportamiento monotónico.

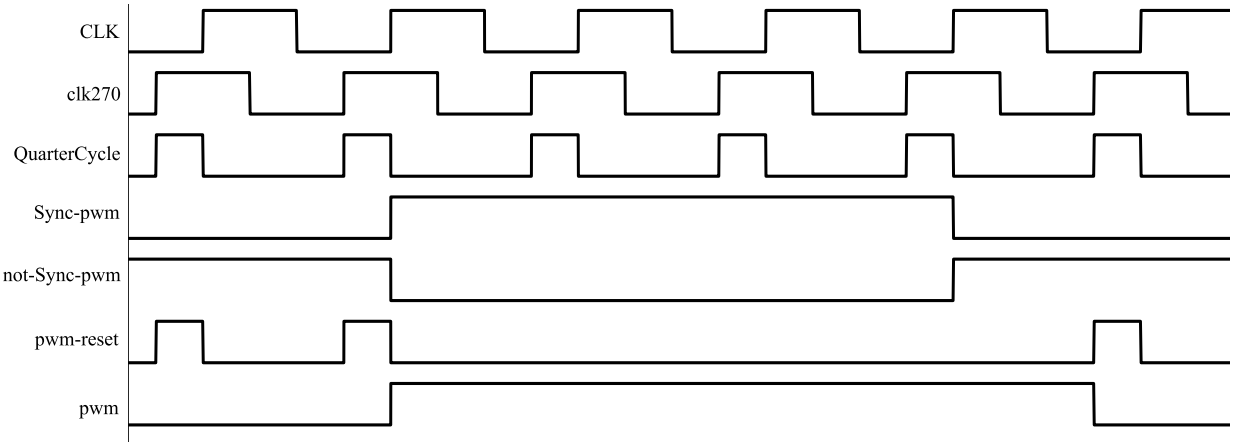

Figura 3-5: Comportamiento ideal del Método 1 para *d[1,0] = "11"*

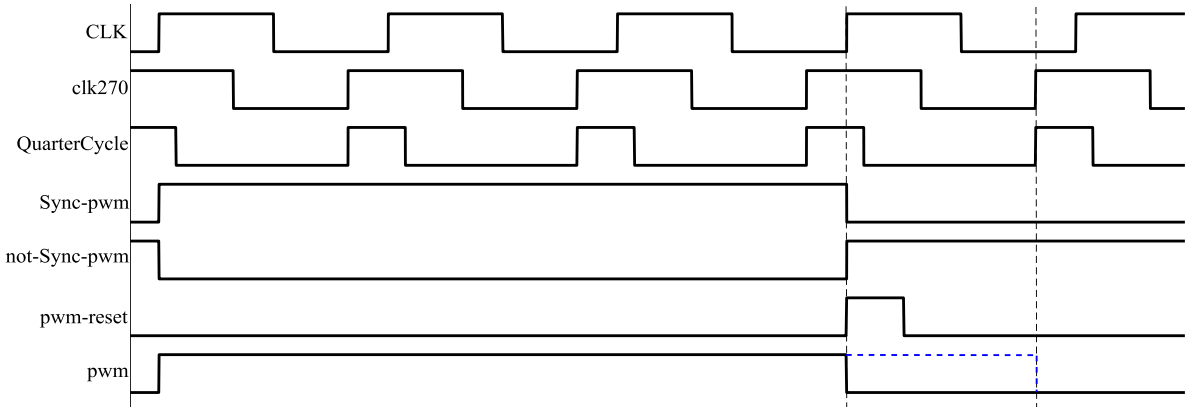

Figura 3-6: Comportamiento no ideal del Método 1 para *d[1,0] = "11".*

Como solución a este problema se propone el diseño que se muestra en la Figura 3-7, al que se le añade en el bloque asíncrono un *latch* y cuatro *flip-flops* que solventarán el efecto de estos retrasos. Cuando *d[1,0]* = "00", "01" o "10", el comportamiento del circuito es el mismo que antes, con la diferencia de que las señales de set y reset del *latch* de salida vienen de dos *flip-flops* controlados por *clk0* que garantizan no tener problemas de timing. Cuando *d[1,0]* = "11", la señal de reset no llega de un *flip-flop*, sino de un *latch* intermedio que a su salida tiene la señal inversa de *Sync-pwm*, pero desfasada 90º, lo que

evita que la señal de reset del *latch* de salida se active antes de tiempo, produciendo un funcionamiento esperado del sistema para este valor. Así, el circuito presenta un comportamiento monotónico.

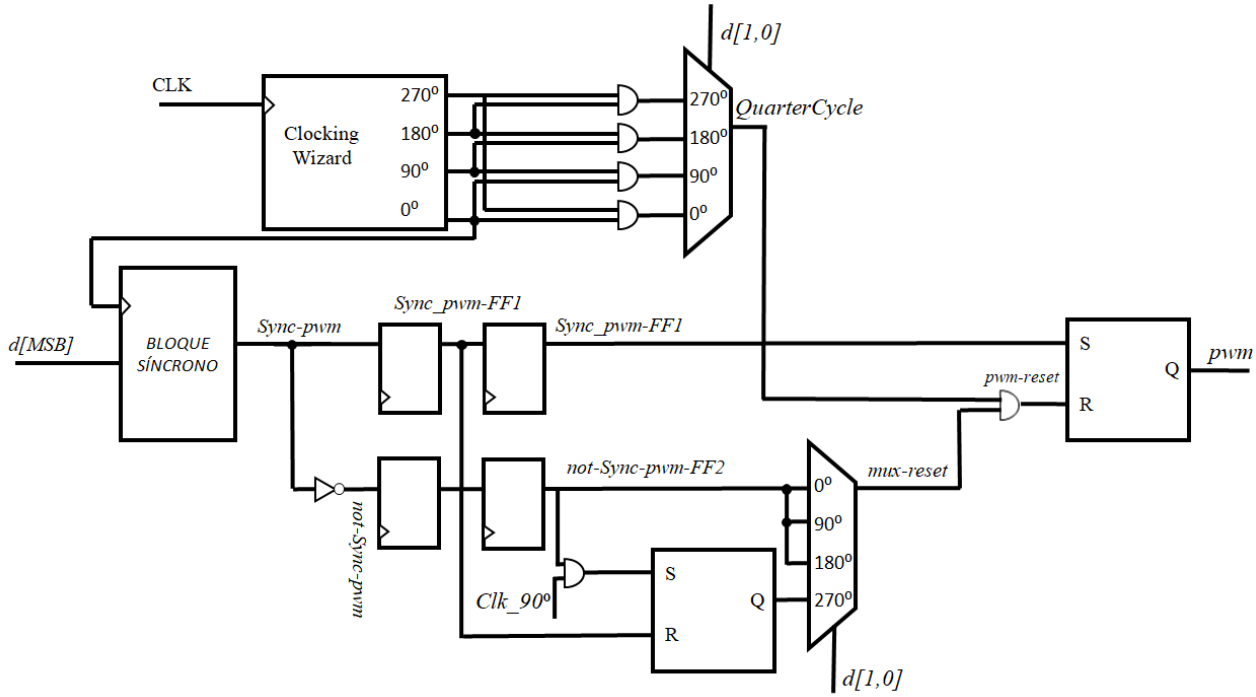

Figura 3-7: Arquitectura Solución al Método 1: Flip-Flops y registro intermedio.

Con este diseño, se obtiene una resolución que es cuatro veces mayor que la que se tiene en la señal *Sync-pwm.* Si se quisiese obtener una resolución mayor manteniendo esta arquitectura, bastaría con generar $2^{d[1,0]}$ señales desfasadas de reloj y combinarlas adecuadamente con puertas AND para que la señal *QuarterCycle* se mantuviese a un nivel alto la inversa parte de 1/*d[LSB]* ciclos de reloj. Por ejemplo, si se tuviesen 3 bits de resolución, habría que generar 8 relojes desfasados y la señal *QuarterCycle* tendría que mantenerse a '1' durante una octava parte de ciclo de reloj. Ha de notarse que esto tiene dos inconvenientes. En primer lugar, el número de compuertas necesarias aumenta en un factor de $2^{d[LSB]}$ por cada bit de resolución añadido, por lo que para un *d[LSB]* relativamente grande se nota un aumento en el uso de recursos considerable. El otro inconveniente es que conforme se aumenta *d[LSB]*, el tiempo en el que *QuarterCycle* está a nivel alto es cada vez menor, y los *latch* requieren un tiempo mínimo en el que la señal esté estable para un correcto funcionamiento.

Por tanto, a partir de cierto valor de *d[LSB]*, se necesitaría que *QuarterCycle* esté activa un tiempo mayor que 1/ *d[LSB]*, que resultaría de nuevo en un funcionamiento incorrecto del sistema para un número alto de bits de resolución**.** En la Figura 3-8, se muestra el comportamiento ideal de este diseño para *d[1,0]* = "01". Y en la Figura 3-9 se observa el papel del *latch* intermedio para garantizar el correcto funcionamiento del circuito para *d[1,0]* = "11". Ahora la franja de tiempo conflictiva indicada, ya no supone un problema.

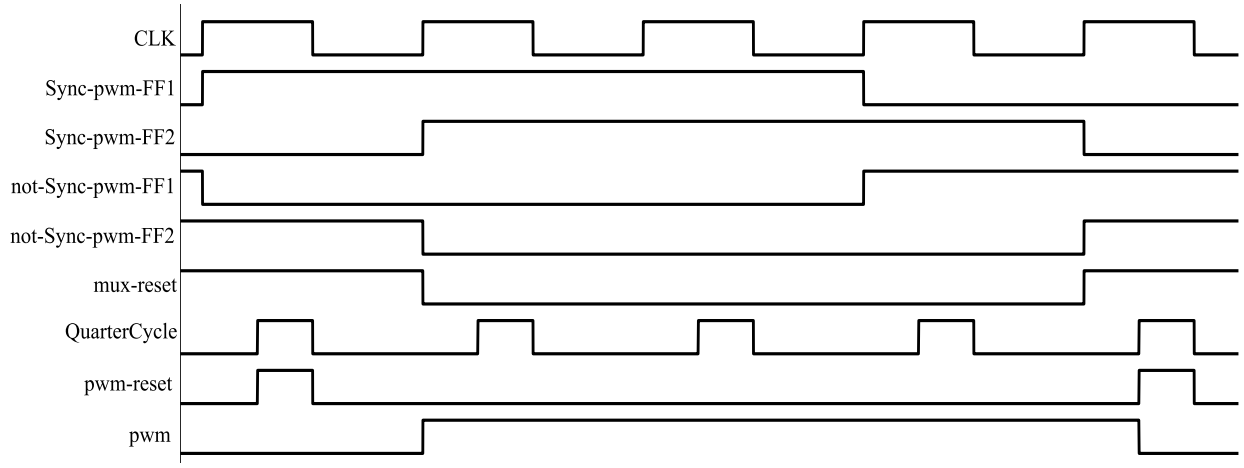

Figura 3-8: Diagrama de solución al Método 1: Flip-Flops y registro intermedio. *d[1,0] = "01".*

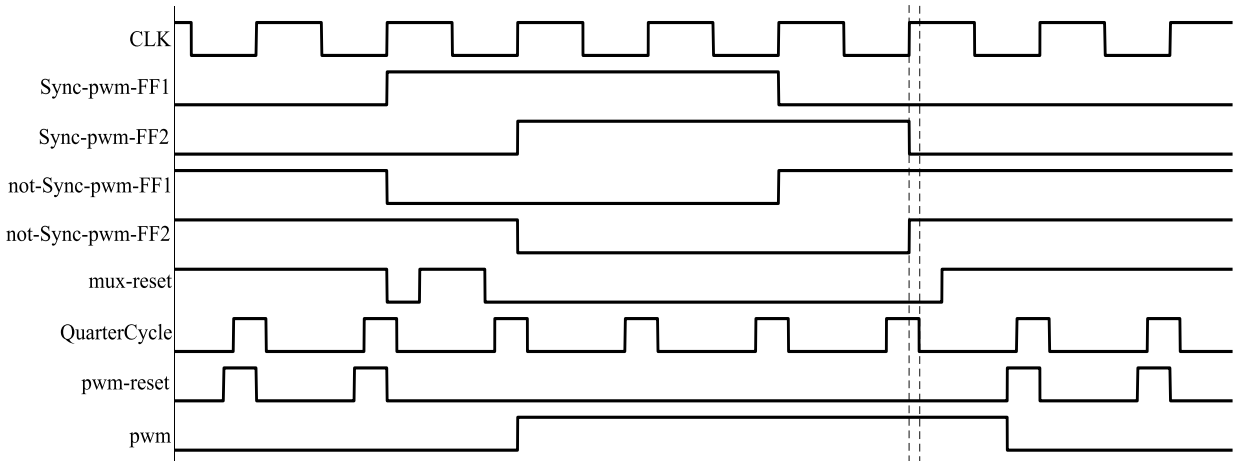

Figura 3-9: Diagrama de solución al Método 1: Flip-Flops y registro intermedio. *d[1,0] = "11".*

3.1.1 Mejora del método 1: “reducción del duty cycle del reloj”

En este apartado se propone una solución del problema asociado a los retrasos que hacen que la señal *QuarterCycle* se active un poco más tarde de lo debido, ocasionando que la señal de Reset del *latch* de salida se active antes de tiempo.

En vez de añadir *latch* intermedios en el diseño, se adoptará de nuevo la misma estructura que en la Figura 3-3**,** con la diferencia de que usaremos característica de la que disponen los Clocking Manager en la familia 7 de Xilinx. Y es que no solo se van a generar los relojes desfasados, sino que además se reducirán sus ciclos de trabajo con el fin de obtener una señal *QuarterCycle* con un ancho de pulso de duración menor a un cuarto de ciclo de reloj, lo suficiente para asegurar que los retrasos en la lógica no activen la señal de Reset del *latch* de salida indebidamente. Cabe destacar como aspecto positivo que mediante esta solución no es necesario el uso de las compuertas AND que se usaban anteriormente para este propósito, utilizando así menos recursos, pero como aspecto negativo, se seguiría teniendo un límite de resolución en cuanto a que la señal *QuarterCycle* debe mantenerse en un nivel estable durante un mínimo de tiempo a la entrada del *latch* para poder cambiar el valor de salida. La arquitectura resultante y la generación de la señal *QuarterCycle* se muestran en la Figura 3-10 y Figura 3-11, respectivamente.

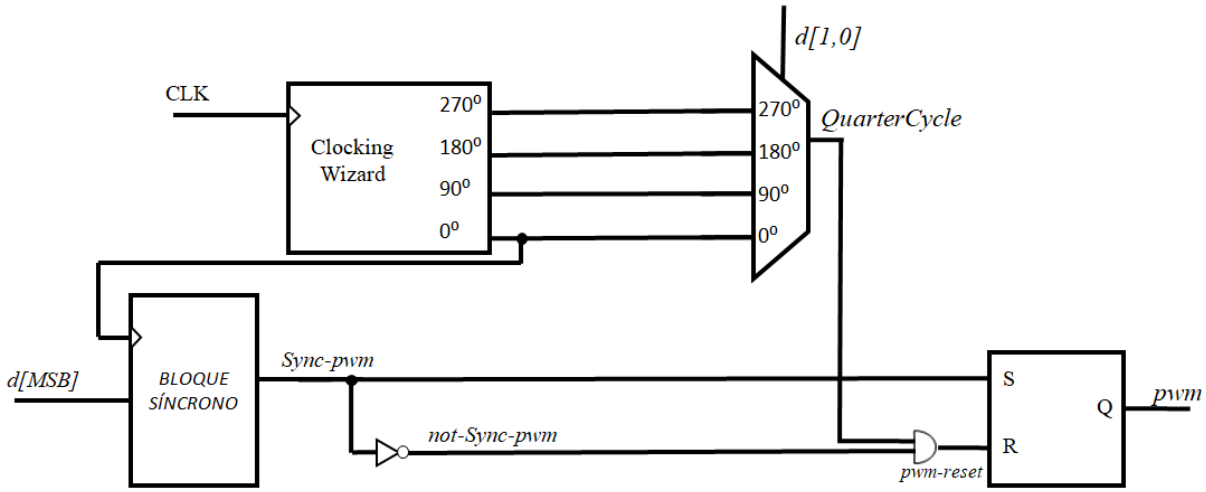

Figura 3-10: Arquitectura Solución al Método 1: reducción duty cycle del reloj.

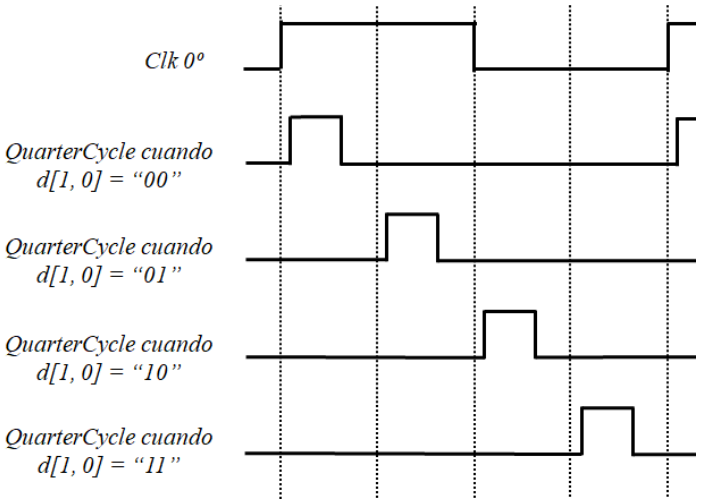

Figura 3-11: Señal QuarterCycle con reducción del duty cycle del reloj.

En conclusión, mediante este método, se consigue una resolución teórica de $\frac{T_{clk}}{4}$ en sus dos soluciones estudiadas. Como T_{clk} son 10 ns (f_{clk} = 100 MHz), la resolución sería de 2,5 ns, que equivale a una frecuencia de 400 MHz.

3.2 Método 2

Si bien es cierto que con el anterior diseño se logra obtener una resolución mayor a un ciclo de reloj del sistema, puede resultar insuficiente para aplicaciones que requieran una mayor precisión, y para aquellas que trabajen a una frecuencia relativamente elevada (en el orden de las decenas de MHz).

Esta nueva solución aprovecha una funcionalidad del Clocking Wizard que también tienen muchas FPGA actualmente, llamada "*Dynamic Phase Shift*". Aprovechando esta característica, no se está limitado a generar un número de relojes de salida que son versiones del reloj de entrada con un desfase fijo. En cambio, estará generándose un único reloj de salida con un desfase programable respecto al reloj de entrada en pequeños incrementos. Esto se consigue mediante un protocolo que involucra algunas señales de control. Por lo tanto, ahora la configuración de salida del reloj es la de la Figura 3-12:

Output Clock	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)		Drives	Use Fine PS
	Requested	Actual	Requested	Actual	Requested	Actual		
☑ clk_out1	100.000	100.000	0.000	0.000	50.000	50.0	BUFG	☑

Figura 3-12: Configuración Clocking Wizard para el Método 2.

Estos incrementos de tiempo varían en función de la FPGA que utilizada. En concreto para las FPGA de la familia 7 de Xilinx estos incrementos equivalen a:

$$inc = \frac{1}{56} * VCO$$

Donde *inc* es el incremento de tiempo entre dos valores de desfase consecutivos, y VCO es el periodo del oscilador controlado por voltaje. En la FPGA utilizada, este oscilador es de 1000 MHz, luego los incrementos que se consiguen son de 17,857 ps, que equivalen a una frecuencia de 56 GHz. Un valor con el que se puede conseguir una precisión mucho mayor respecto a los diseños anteriores. Como el reloj de nuestra FPGA es de 100 MHz, que equivale a un periodo de 10 ns, el número de incrementos para lograr un desfase de 360º será de:

$$\frac{10[ns]}{17{,}875[ps]} = 560\ incrementos$$

Para este método, se han usado 9 bits de parte fraccionaria, por lo que es necesario hacer un escalado.

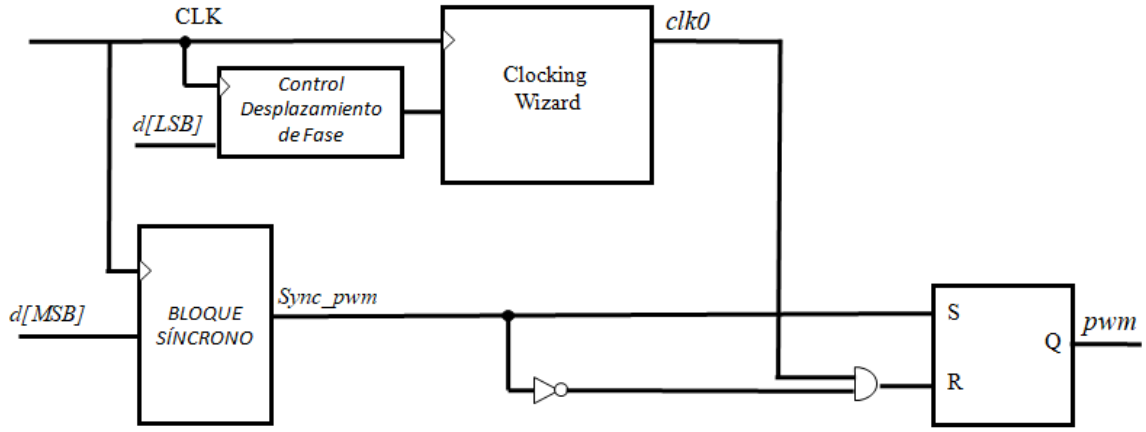

Figura 3-13: Arquitectura del Método 2.

Como se puede observar en la Figura 3-13**,** la estructura del bloque asíncrono es conceptualmente parecida al de los diseños anteriores: en el momento en que el reloj desfasado que generamos pasa a un nivel alto, se activa la señal de reset del *latch* de salida que cambia la *señal pwm* a un nivel bajo (si la señal de la parte síncrona es '0' en ese momento). La diferencia radica en cómo es generado el reloj que controla el la señal de reset. El desfase del reloj de entrada es controlado de forma dinámica (en lugar de en la etapa de síntesis del circuito), por el Clocking Wizard mediante un protocolo en el que están involucradas las siguientes señales:

- PSCLK (*Phase-Shift Clock*): el reloj de entrada, que sirve de referencia.

- PSEN (*Phase-Shift Enable*): es una señal de habilitación que permite que comience la operación de desplazamiento. Si es '1' durante un periodo de ciclo de PSCLK comienza una operación de desplazamiento.

- PSINCDEC (*Phase-Shift Increment/Decrement Control*): es una señal que, en función de su valor, '1' o '0', se indica que se va a realizar un incremento o un

decremento, respectivamente, de valor 17,875 ps como se ha indicado antes El tiempo que se tarda en hacer un incremento es siempre de 12 ciclos de PSCLK.

- PSDONE (*Phase Shift Done*): señal que indica que una operación de desplazamiento ha finalizado, que *clk0* ha alcanzado la misma fase (o la más cercana posible) a la de PSCLK. Su duración es de un ciclo de PSCLK.

- Los bits LSB de *d*, que indicarán la fracción de reloj que se desea añadir.

El Clocking Wizard quedaría entonces como en la Figura 3-14:
.

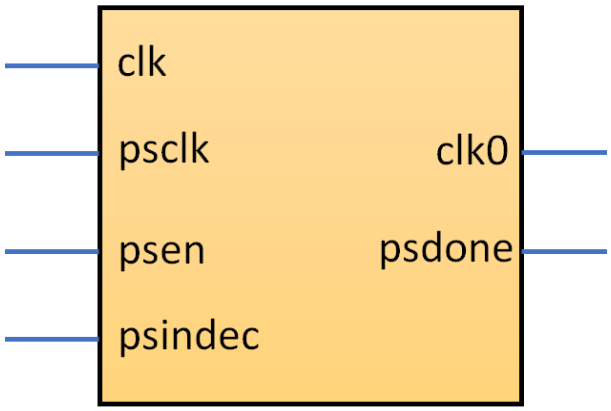

Figura 3-14: Esquema Clocking Wizard, Método 2.

Cabe destacar que mediante este protocolo se puede desfasar sin problema un reloj más de 360º, ya que no hay un máximo desplazamiento permitido. Pero esto será posible, siempre y cuando el periodo de reloj sea menor o igual que la suma de todos los incrementos posibles. El diagrama temporal de este protocolo se muestra en la Figura 3-15:

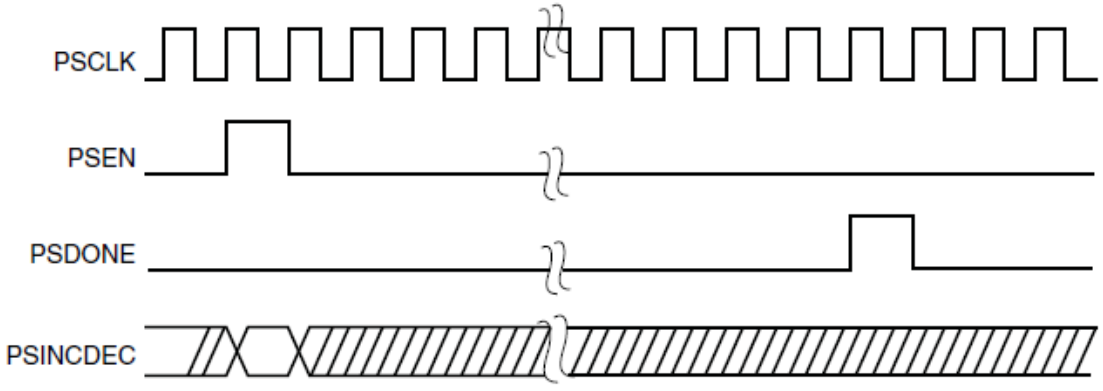

Figura 3-15: Diagrama protocolo Dynamic Phase Shifting [15].

En conclusión, en la Figura 3-16 se ven reflejados los estados por los que pasa este protocolo. El protocolo parte del estado LOCKING, y no cambia a IDLE hasta que la señal *locked* no se active. Este estado ha sido añadido para asegurar que todas las señales estén estables antes de comenzar y así evitar posibles errores que puedan pasar desapercibidos. En el estado IDLE se verifican las condiciones con las que se determina si se siguen

realizando o no incrementos o decrementos, es decir, si se ha llegado o no a la diferencia de fase solicitada. En el caso de que se inicie un incremento (*psen* = '1'), se pasa al estado WAITING, en el que se espera a que se complete la operación de incremento o decremento. Una vez finalizada esta operación, la señal *psdone* se activa y se vuelve al estado IDLE para comprobar de nuevo si se debe realizar otro incremento.

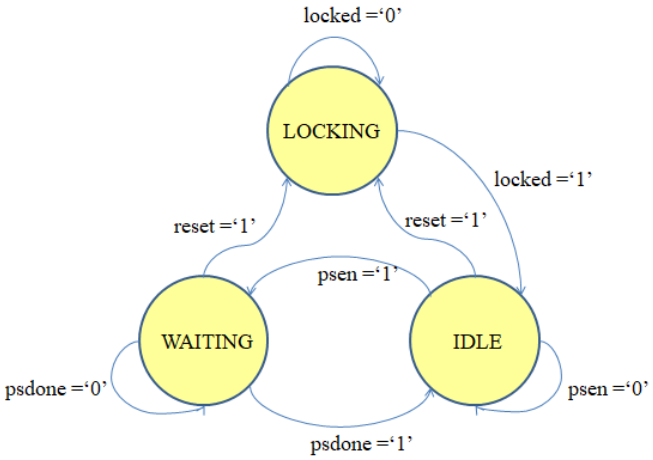

Figura 3-16: Diagrama de estados del protocolo de Dynamic Phase Shifting.

El funcionamiento de este método se muestra en la Figura 3-16. Sin embargo, con esta solución se vuelve a tener un problema parecido al que se tenía anteriormente como se observa en la Figura 3-17**.** Lo que pasa es que, si el *clk0* que se genera tiene un ciclo de trabajo del 50%, cuando se le indique que adopte un desfase de 180º o superior, la señal de reset del *latch* se activará cuando la señal de set ya es '0' y por tanto, tendremos un comportamiento del sistema erróneo.

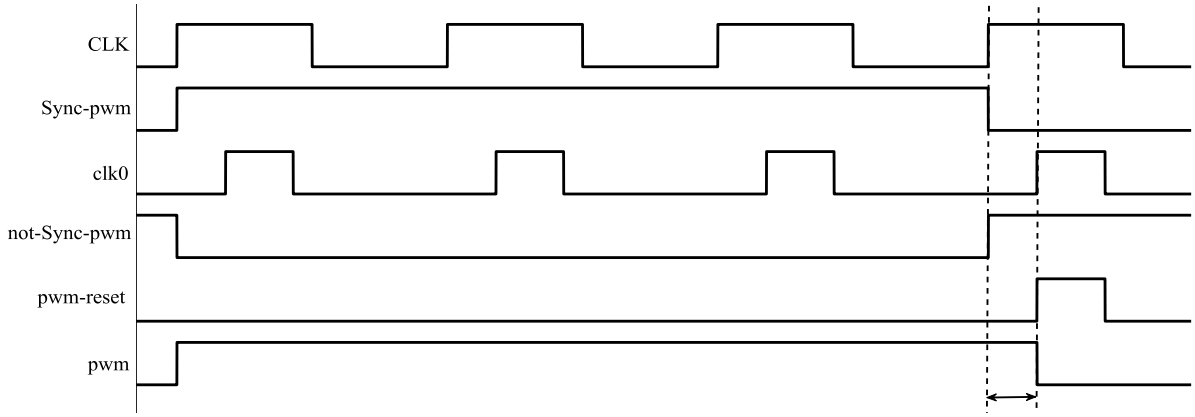

Figura 3-17: Diagrama del Método 2.

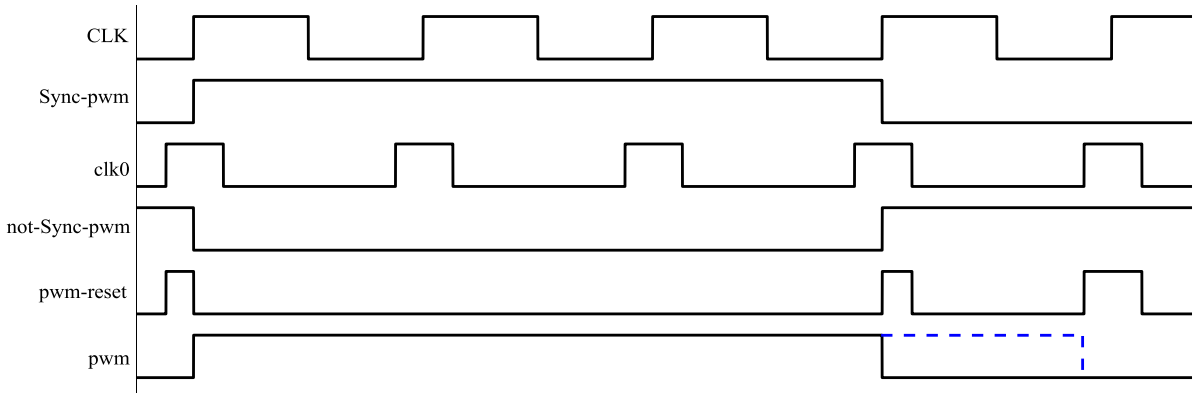

Figura 3-18: Diagrama del Método 2 con problema de fases cercanas a 180º.

Para resolver este problema, se podría proponer que *clk0* tuviese un ciclo de trabajo menor del 50%, por ejemplo, del 25%. Pero de esta manera lo único que se consigue es reducir el rango de valores de desfase que dan problemas, en vez de tenerse para desfases superiores a 180º, se tendrán para desfases superiores a 270º. En la Figura 3-18 puede verse que *clk0* tiene un ciclo de trabajo del 25% y que a partir de 270º hay un periodo de tiempo en el que esta señal se mantiene a '1' junto a *not-Sync-pwm*, y *pwm* se mantiene igual que *Sync-pwm.*

Para tener un comportamiento correcto para todos los valores se estudia la arquitectura de la Figura 3-19**,** en la se han añadido una serie de *flip-flops* intermedios, debido a dos razones. Una es para mitigar el efecto de los retrasos de las pistas de la FPGA, que a partir de cierta frecuencia de reloj, no son despreciables frente al tiempo de un incremento, ni frente a los retrasos de las pistas del circuito o de un componente lógico. En este TFG se ha trabajado con un reloj de 100 MHz, lo que equivale a un periodo de reloj de 10 ns. Y los retrasos en los caminos de la FPGA pueden ser de varias unidades de nanosegundo.

Además, estos *flip-flops* tienen otra función firmemente relacionada con la anterior. Con motivo de compensar los retrasos de los caminos principales de las señales de esta arquitectura para obtener buenos resultados experimentales, se ha realizado una colocación manual de algunos de estos *flip-flops*. Hacer lo mismo únicamente con componentes combinacionales resulta mucho más complejo.

Como se aprecia en la Figura 3-19, hay dos *flip-flops* gobernados por *clk0*, uno con flanco de subida, y el otro con flanco de bajada. El del flanco de bajada se usa cuando se pide un retraso menor a 180º, y el del flanco de bajada, se usa para cuando se pide un retraso mayor a este valor. El multiplexor elige en función de este valor una señal u otra. El comportamiento de las señales con esta arquitectura se muestra en la Figura 3-20. En concreto, se está pidiendo un valor de desfase superior a 180º.

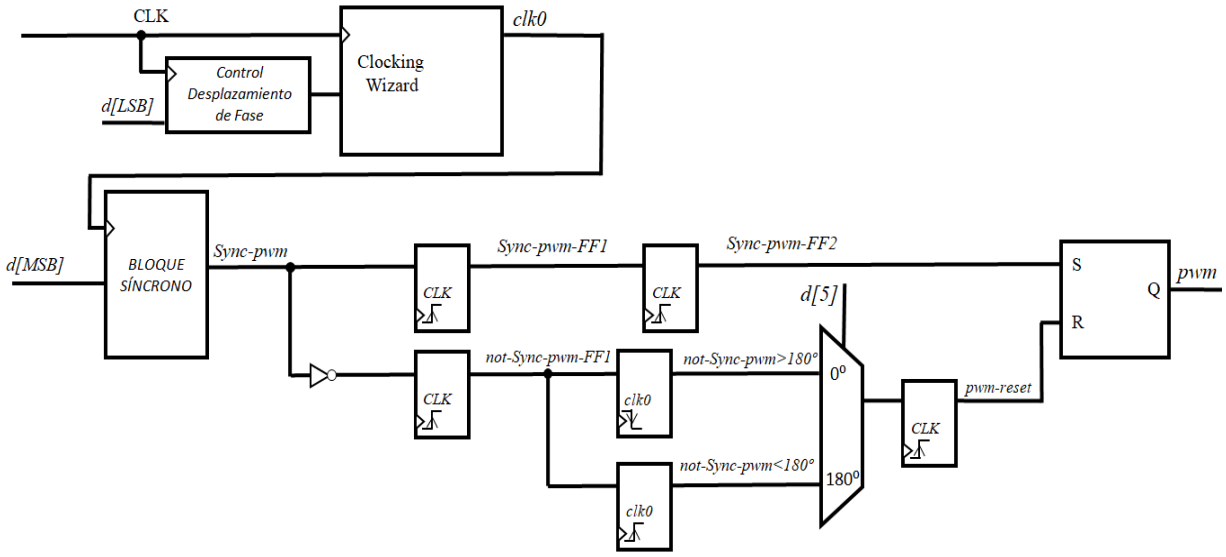

Figura 3-19: Arquitectura solución al Método 2.

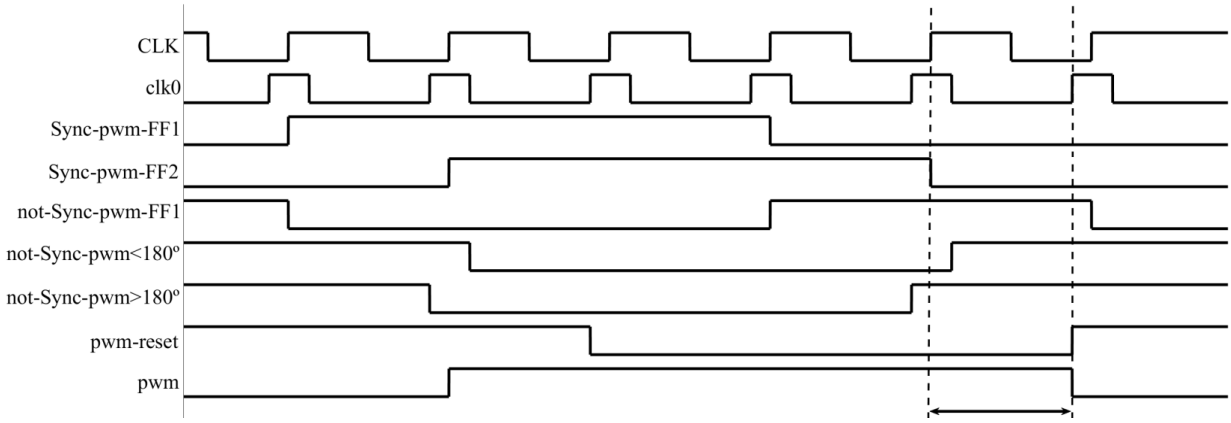

Figura 3-20: Diagrama de solución del Método 2.

Con la arquitectura de este método, se consigue una resolución mucho mayor (17,875 ps) que la del Método 1 (2,5 ns), de una manera sencilla. Como se mencionó anteriormente, se ha trabajado con 9 bits de resolución. Para conseguir esta resolución mediante el Método 1 se hubiesen necesitado generar 2^9 señales de reloj desfasadas, y combinarlas usando 2^9 puertas AND. En definitiva, resultaría algo muy poco práctico.

No obstante, este método tiene una limitación en lo que se refiere al tiempo que emplea en dotar a *clk0* de un determinado desfase respecto del reloj de entrada. Como se ha visto anteriormente, desde que se comienza un incremento (o decremento) hasta que termina se consumen 12 ciclos de reloj. Es por ello que esta limitación es más acentuada conforme se realizan desplazamientos de fase más altos. Teniendo en cuenta que la consigna del bloque DPWM vendrá dada por un regulador en lazo cerrado que la irá variando constantemente, el sistema puede resultar impráctico. Es muy probable que el regulador decida cambiar el ciclo de trabajo antes de que el protocolo de comunicación consiga el desfase esperado. Además, puede pensarse que este problema se da únicamente en régimen transitorio, pero incluso cuando el regulador trabaja en régimen permanente las pequeñas variaciones de la

consigna pueden ser lo suficientemente significativas como para tener un mal funcionamiento del sistema.

Es por ello que en la siguiente sección se describirá el tercer método, que tratará de alcanzar una buena resolución, y cuyo tiempo de operación sea menor que el del Método 2.

3.3 Método 3

Con objeto de subsanar el problema del tiempo que tarda el anterior método en generar la señal PWM de salida con el ciclo de trabajo deseado y manteniendo una buena resolución y linealidad, se estudia este método que hace uso de un componente hardware incluido en las FPGAs de la familia 7 de Xilinx llamado IODELAY.

El IODELAY trabaja con una frecuencia de reloj de referencia, f_{clk} de 200 MHz, y estos bloques trabajan con una resolución de:

$$Resolución = \frac{1}{2 * 32 * f_{clk}} \approx 78\ ps$$

Se encargará de generar una señal a su salida que sea una versión desfasada un cierto valor programable de la señal de entrada. Es decir, implementa un protocolo interno para realizar lo mismo que el Clocking Wizard en el método anterior. Solo que, en lugar de desfasar el reloj de entrada, se utilizará para desfasar directamente la señal de salida del bloque síncrono, *Sync-pwm.* En concreto, se va a usar el bloque de retraso IDELAYE2, para lo cual se necesitará utilizar de forma conjunta un bloque de control IDELAYCTRL. Este último bloque recibe un reloj de una frecuencia conocida y genera un reloj de salida que se conecta internamente al bloque IDELAYE2 para obtener desfases precisos.

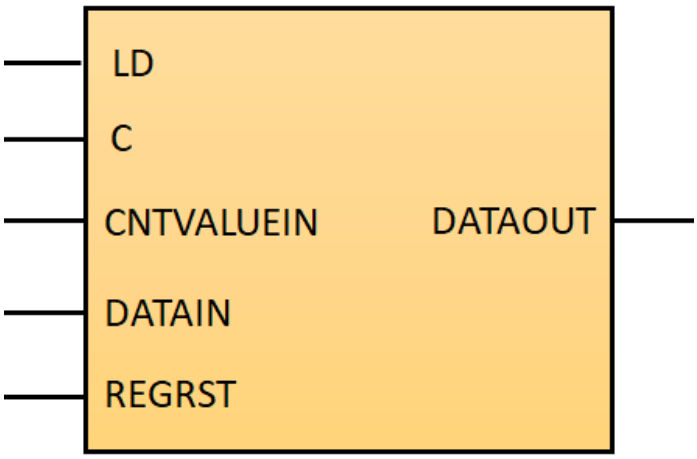

Figura 3-21: Esquema simplificado IDELAYE2.

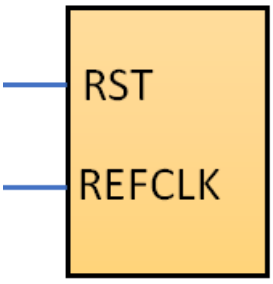

Figura 3-22: Esquema simplificado IDELAYCTRL.

En la Figura 3-21 se muestra un esquema simplificado del componente IDELAYE2 con las señales más importantes involucradas para el fin que se busca:

- LD: señal de habilitación de carga. Se activará esta señal para poder cargar el valor de desfase indicado por CNTVALUEIN.
- C: señal de reloj de entrada. En este caso, será el reloj del sistema de 100 MHz.
- CNTVALUEIN: señal de 5 bits de tamaño mediante la cual indicaremos el número de incrementos que deseamos desfasar la señal de entrada.
- DATAIN: señal de entrada que se quiere desfasar.
- REGRST: señal de reset activo-alto.
- DATAOUT: señal de salida, desfasada CNTVALUEIN respecto de la DATAIN.

En la Figura 3-22 se muestra el esquema del bloque de control IDELAYCTRL cuyas señales son:

- RST: señal de reset activo-alto.
- REFCLK: reloj de referencia de entrada. En este caso como ya se ha mencionado antes será de 200MHz, aunque también se puede conectar un reloj de 300 MHz.

El resto de las señales estarán desactivadas o se dejarán desconectadas según las especificaciones del fabricante. El reloj de salida que genera este módulo y que sirve de referencia para el bloque IDELAYE2 no se ve reflejado en las figuras.

La arquitectura que se adopta en este método es la reflejada en la Figura 3-23. Como se muestra la señal *Sync-pwm* se sigue generando de la misma manera, pues se sigue manteniendo el mismo bloque síncrono que en los dos anteriores métodos. Pero tiene una ligera diferencia, y es que si antes *Sync-pwm* estaba activa tantos ciclos como marca *d[MSB]*, ahora estará activa un único ciclo, ya que a la hora de realizar simulaciones después de implementación en la FPGA para valores pequeños de *d[LSB]*, la señal de reset se activaba antes de tiempo. La señal que indica el momento en el que la parte entera de *Sync-pwm* debe ponerse a '1' se llama *pwm-set*, y la que indicaría que se debe poner a '0' (cuando la cuenta del contador llegue al valor indicado por *d[MSB]*) se llama *pwm-reset.* Conceptualmente este bloque sigue haciendo lo mismo pero indicando únicamente los momentos de transición de lo que sería la señal *Sync-pwm.*

En lo que se refiere al bloque asíncrono, como se puede observar, el circuito trabaja con un reloj cuyo valor de frecuencia es de 200 MHz. Para generar este reloj se hace uso del Clocking Wizard, pues como se mencionó al principio de esta memoria, permite sintetizar un amplio rango de frecuencias. Cuando empieza la cuenta, *pwm-set* se pone a '1'. Esta señal es registrada por un *flip-flop* en flanco de subida, y su señal de salida, *Sync-rising*, pasa por un IODELAY al que se le indica que no aplique ningún retraso a la señal de entrada. Finalmente, la salida de este bloque, *Sync-delayed*, es la que indica la activación de *pwm*. Este bloque de IODELAY puede parecer que no es necesario, pues le indicamos que no retrase su señal de entrada, por lo que parecería lógico quitarlo. El problema es que experimentalmente se puede comprobar que este componente introduce un retraso de 600 ps incluso aunque se pida un retraso nulo. Este retraso fijo (*offset*) no es nada despreciable teniendo en cuenta que estamos trabajando con un reloj de 200 MHz, lo que equivale a un periodo de 5 ns. Entonces, si el resto de IODELAYs del circuito va a retrasar 600 ps más su señal de entrada de lo que se le indica retrasar en un principio, la misión de este IODELAY simplemente es la de compensar este retraso adicional y que la señal *pwm* se active en el momento correcto.

Por otro lado, hay una rama que registra en flanco de subida a la señal *pwm-reset*, y a su salida tendrá *Sync-off-rising*. Esta rama es la encargada de indicar el reseteo de *pwm*. Posteriormente, esta señal pasa a través de un IODELAY que le aplica un retraso con el valor que indica *d[4,0]*. El problema es que si cada incremento posible es de 78 ps, y con 5 bits de parte fraccionaria podemos realizar hasta 32 incrementos, lo máximo que sería posible desplazar la señal *Sync-off-rising* serían 78[ps] * 2^5 = 2,496 ns ≈ 2,5 ns. Entonces como el periodo de reloj es de 5 ns estaríamos limitados a tener *pwm* activa únicamente medio ciclo más. Por ello es por lo que hay un segundo *flip-flop* que registra *Sync-off-rising* en flanco de bajada que sirve para aplicar un retraso al reseteo de ese medio ciclo que no podríamos alcanzar con un solo IODELAY. Por esa razón la parte fraccionaria de *d* tiene un bit más, *d[6]*, que sirve como señal de selección del multiplexor, de tal manera que para retrasos menores a medio ciclo de reloj la señal de reset que se aplica en el *latch* de salida es *reset-delayed-rising*, y para retrasos mayores a medio ciclo de reloj se usa *reset-delayed-falling.*

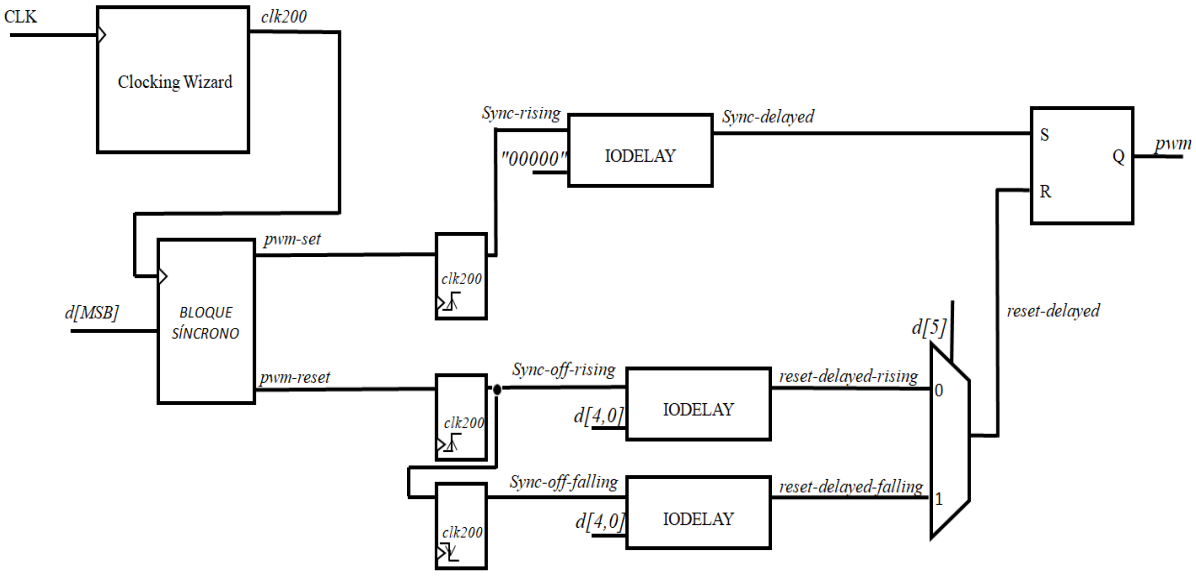

Figura 3-23: Arquitectura del Método 3.

En la Figura 3-24 se muestra el comportamiento de las señales de este método cuando se pide que *pwm* tenga una un ciclo de trabajo de parte fraccionaria menor que medio ciclo de reloj (bit d[5] = ‘0’), mientras que en la 3-25 se pide que sea mayor a medio ciclo de reloj (bit d[5] = ‘1’), pudiendo observar el papel de las señales de reset en ambos casos.

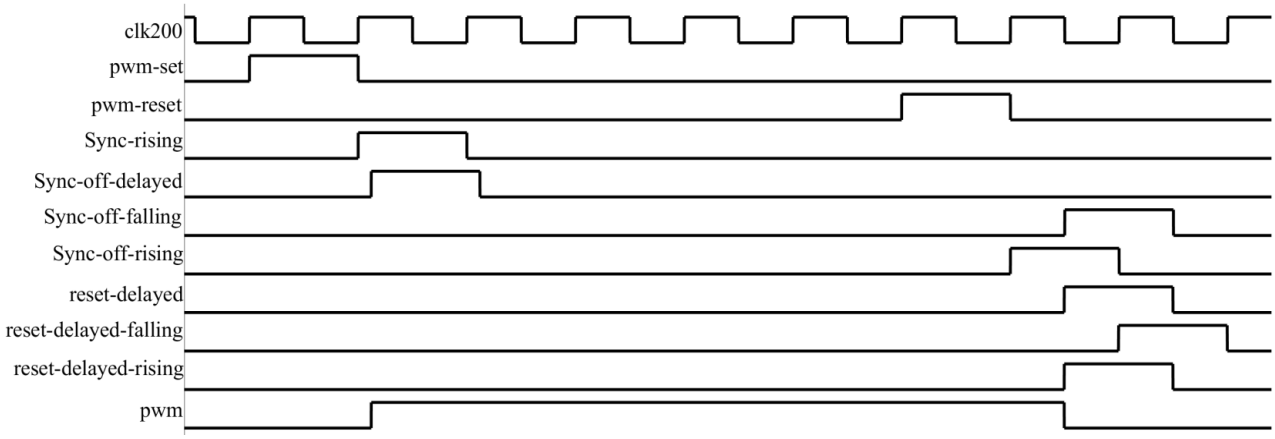

Figura 3-24: Diagrama método 3: retraso menor a medio ciclo de reloj (bit d[5] = ‘0’).

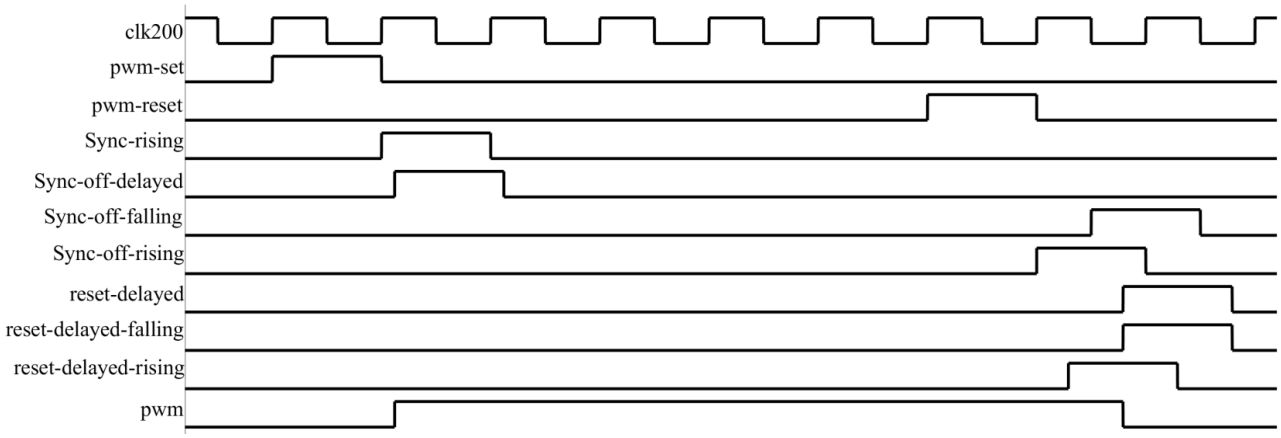

Figura 3-25: Diagrama método 3: retraso mayor a medio ciclo de reloj (bit d[5] = ‘1’).

Mediante esta manera se obtiene un sistema DPWM de alta resolución (6 bits) que posibilitan realizar pasos de 78 ps (12,82 GHz), con la FPGA utilizada. Aunque consigue una resolución más baja que el método anterior, es mucho más rápido, de tal manera que se elimina el problema de que el regulador actualice el valor de consigna antes de tener la señal *pwm* con el ciclo de trabajo final.

4 Integración, pruebas y resultados

En esta sección se recogerán los resultados ideales y experimentales de los métodos descritos anteriormente. Todos ellos han sido implementados en lenguaje VHDL utilizando la herramienta Vivado 2016.2, de Xilinx.

Para la parte de simulación del comportamiento de los sistemas se ha utilizado la herramienta ModelSim 10.1b. Se han realizado dos tipos de simulaciones:

- Simulaciones ideales: estas simulaciones no contemplan ningún tipo de retraso el circuito y se han realizado básicamente con el objetivo de comprobar que el diseño esté bien realizado y muestre el comportamiento esperado.

- Simulaciones posteriores a la implementación: estas simulaciones sí que tienen en cuenta los retrasos de la lógica de los diseños, tanto los retardos que introducen las pistas (que serán mayores conforme tienen mayor longitud), como los que introducen las propias puertas lógicas de la FPGA en lo que se refiere al procesamiento de las señales. Estos retardos también dependen de la propia FPGA utilizada. El objetivo de las simulaciones de este tipo tiene como objetivo asegurar que el sistema tiene una buena linealidad y monotonía.

Aunque la simulación ideal pueda arrojar buenos resultados, puede darse la posibilidad de que en la simulación posterior a la implementación estos resultados no sean correctos y el sistema presente falta de linealidad. Para solucionar esto, como se introdujo anteriormente en el Método 2, se ha hecho uso de una colocación manual de los componentes críticos que ayudan a la hora de ajustar los retrasos de determinados caminos que son origen de nuestro problema. Generalmente la mayoría de estos componentes son *flip-flops*, pues hacen más sencillo este proceso al permitir centrarse en la etapa del circuito que resulta problemática. Este proceso se ha realizado en Vivado de forma gráfica, de tal manera que se actualiza el fichero de restricciones con las condiciones impuestas. Para alcanzar el mismo fin también se podría haber realizado enrutado manual de las pistas, sin embargo, esto resulta complejo y poco práctico debido a la cantidad de pistas que pasan por diferentes *buffers* que genera el sintetizador.

Una vez verificados los resultados en el simulador y adoptada la configuración de colocación de componentes final, se procede a la comprobación de los resultados finales obtenidos en el osciloscopio. La FPGA que se ha usado en concreto es la XC7A35TICSG324-1L, cuya frecuencia de reloj principal es de 100 MHz. Y los resultados se han obtenido utilizando el osciloscopio de la Figura 4-1, en concreto, el modelo MSOX3014A de Agilent Technologies (100 MHz, 4 GSa/s).

Figura 4-1: Osciloscopio.

A la hora de realizar estas pruebas experimentales, se necesitaría un osciloscopio capaz de medir, como se ha mencionado anteriormente, incrementos de 17,875 ps, equivalentes a una frecuencia de 56 GHz en el caso del Método 2, o de 78 ps equivalentes a una frecuencia de 12,82 GHz, en el caso del Método 3. Pero debido a que el osciloscopio con el que se han realizado estas medidas tiene una frecuencia de muestreo de 4GSamples/s no se pueden realizar medidas precisas debido a su falta de resolución. Por ello se ha decidido calcular el tiempo medio activo de la señal dependiendo del ciclo de trabajo. El osciloscopio no puede capturar de forma precisa los flancos de la señal *pwm*, sin embargo, tras tomar un elevado número de muestras se puede calcular el tiempo medio debido a que la frecuencia de muestreo del osciloscopio no estará acoplada con la frecuencia de la señal PWM. Por lo tanto, en las figuras 4-2, 4-3, 4-4, 4-5 y 4-9 se muestra el tiempo medio activo de la señal PWM generada.

En las Figuras 4-2 a 4-5 se muestran conjuntamente los resultados ideales y experimentales obtenidos para cada método por separado. Con puntos azules se representan los resultados experimentales, y con aspas negras los ideales. Las escalas de todos los ejes de todas las gráficas están en nanosegundos. El eje horizontal representa el valor esperado de ciclo de trabajo, y el eje vertical representa el valor que se ha medido experimentalmente. En los métodos 1 y 2 se aprecia una buena linealidad del sistema que se puede corroborar con la línea de tendencia de las muestras que se ha añadido.

Figura 4-2: Resultados del Método 1.

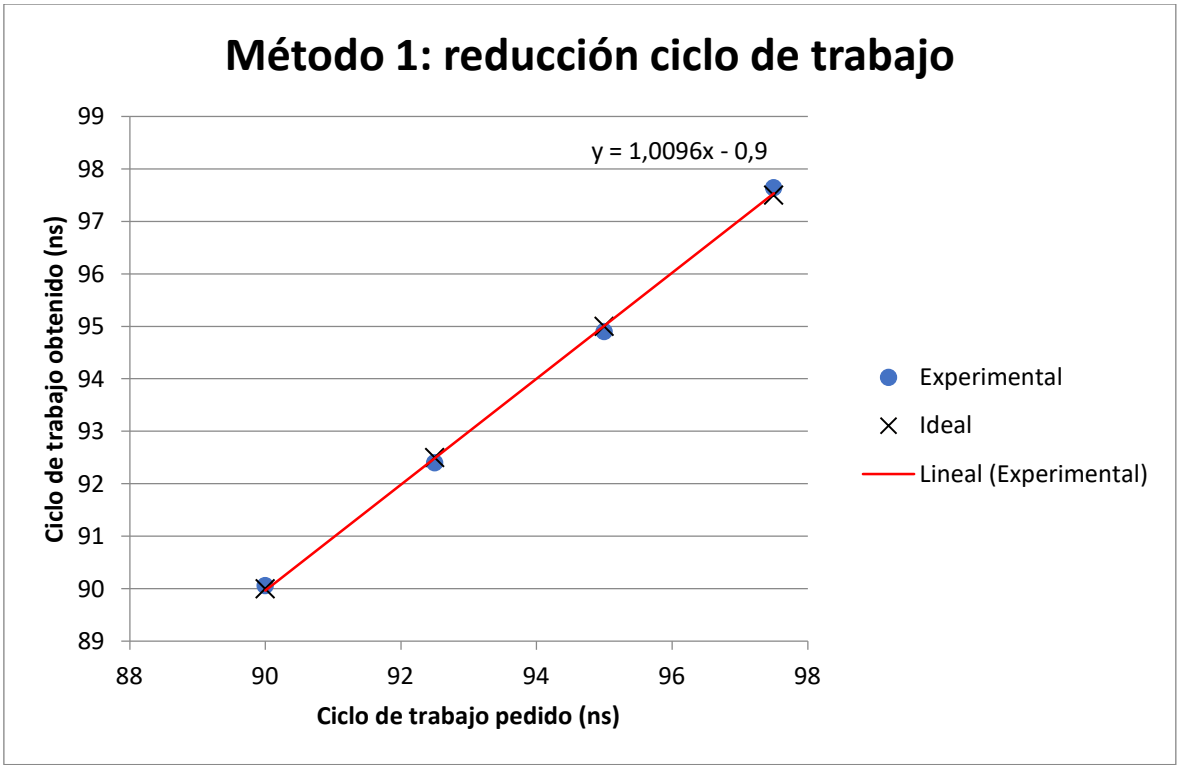

Figura 4-3: Resultados solución Método 1: reducción duty cycle de reloj.

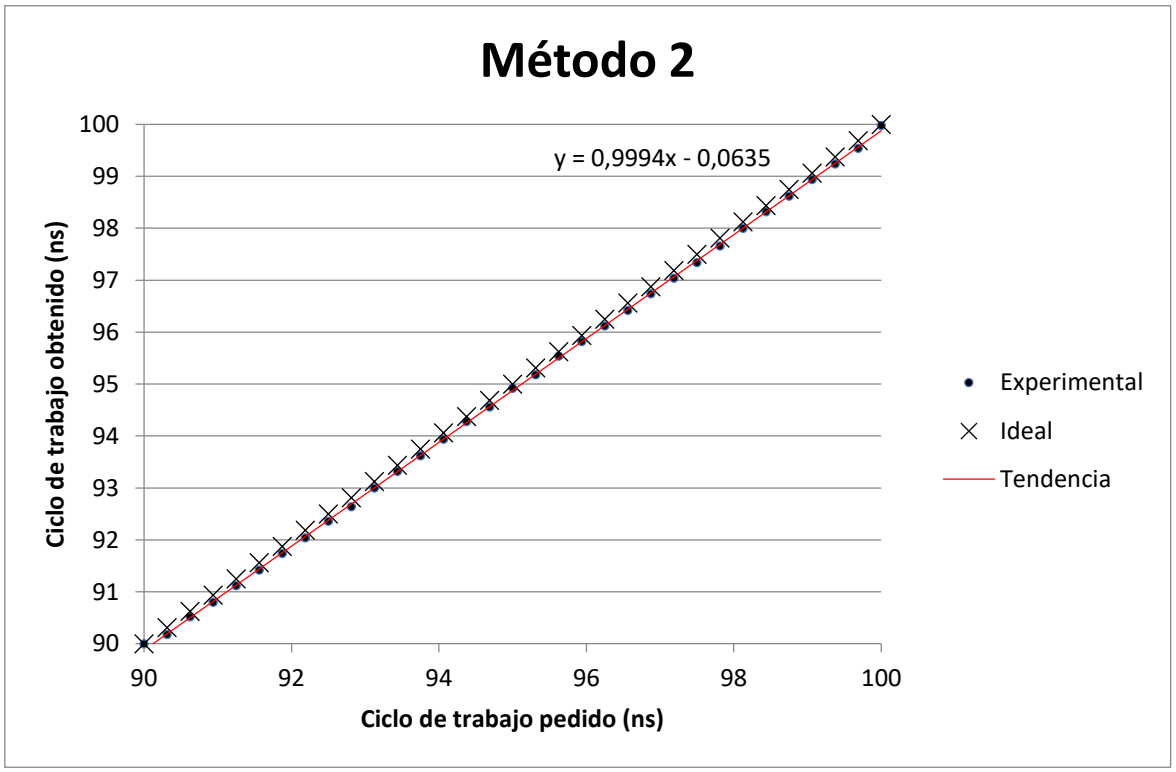

Figura 4-4: Resultados del Método 2.

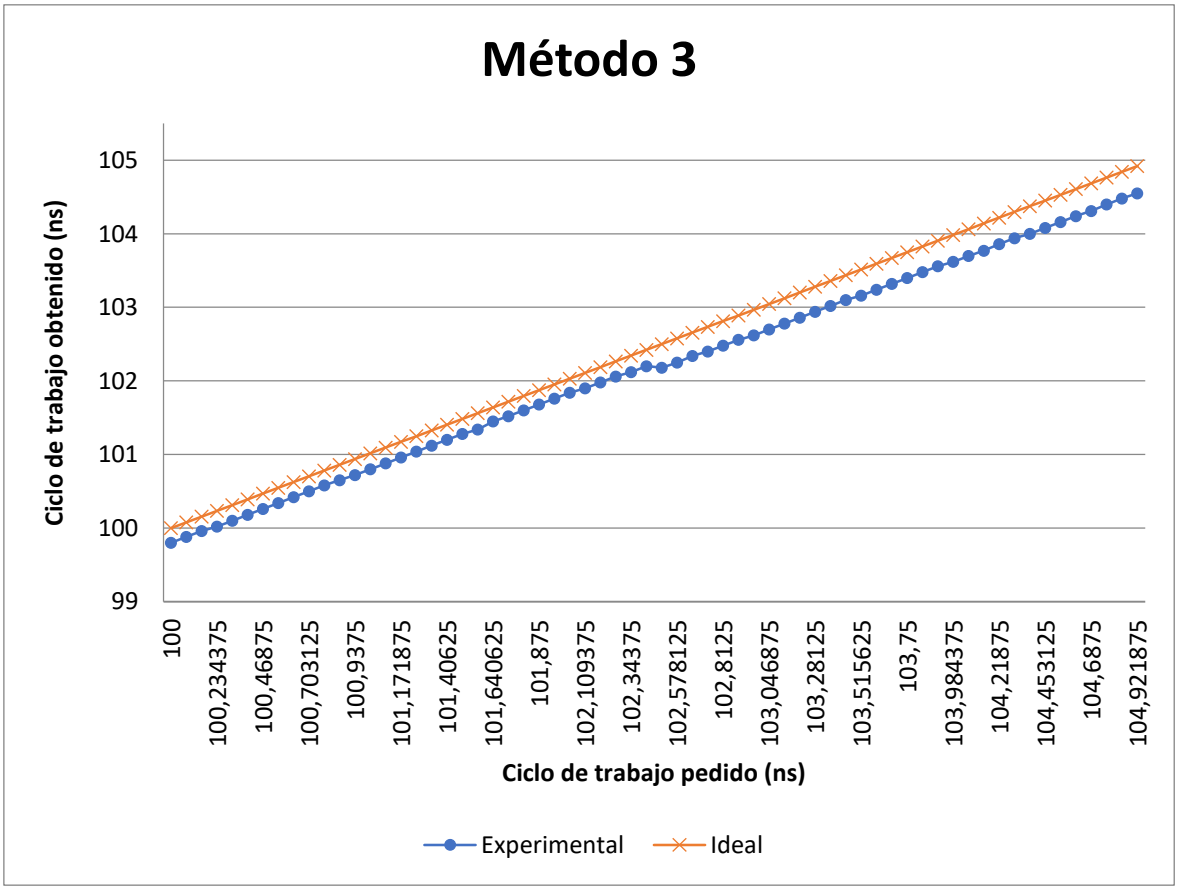

Figura 4-5: Resultados del Método 3.

Los resultados de la alternativa del Método 1 de la Figura 4-3, consistente en la reducción del ciclo de trabajo del reloj que permitía ahorrar todas las puertas AND de la Figura 4-2 así como eliminar la necesidad de utilizar *flip-flops*, son muy similares a los de la Figura 4-2, e incluso ligeramente mejores en cuanto a linealidad, como se observa en la línea de tendencia. Por lo tanto, la alternativa de la Figura 3-10, es preferible de utilizar ya que reduce la lógica empleada en el bloque asíncrono de manera significativa frente a la solución de la Figura 3-7.

Para tomar los resultados de todos los métodos se ha introducido la señal *d*, con la que indicamos el ciclo de trabajo deseado de la señal *pwm*, mediante la utilización del componente VIO (*Virtual Input Output*), a fin de simplificar el proceso debido a la gran cantidad de entradas utilizadas. La parte entera, *d[MSB]* ha sido en todos los casos de 8 bits. En el Método 1 la parte decimal, *d[LSB]* han sido 2 bits, con una frecuencia de reloj T_{clk} de 100 MHz, con lo que se esperaría una resolución teórica de 2,5 ns. En el Método 2 *d[LSB]* han sido 9 bits, T_{clk} de 100 MHz, y se esperaría una resolución de 17,875 ps. Y en el Método 3 *d[LSB]* han sido 6 bits, T_{clk} de 200 MHz y se esperaría una resolución teórica de 78 ps. En los métodos 1 y 2 se ha ido pidiendo un ciclo de trabajo desde 90 ns a 100 ns, mientras que en el método 3 se ha ido pidiendo que varíe de 100 ns a 105 ns. Es decir, en todos los casos se han ido pidiendo retrasos de hasta un ciclo de reloj completo.

El Método 3 también presenta un comportamiento muy lineal en dos tramos bien diferenciados, sin embargo, en conjunto presenta un pequeño pero apreciable "salto" cuando se pide que la fracción de tiempo a '1' de la señal PWM sea superior a medio ciclo de reloj. Este problema es debido a la presencia del multiplexor de la Figura 3-19 que selecciona una rama u otra en función de si se pide un retraso mayor o menor a 180º. En una simulación ideal este salto no existe, pero en una simulación real, hay que tener en cuenta que cada uno de los dos caminos de reset tiene un cierto retraso que es complicado de igualar mediante una colocación manual de los componentes, originando ese *offset*. Esto puede suponer un riesgo si es lo suficientemente grande para que el regulador no funcione de la manera esperada y actualice los valores de consigna de manera errónea. Por ejemplo, si a partir de un determinado valor del ciclo de trabajo de la señal PWM el regulador actualiza el valor de consigna a uno más grande, y debido a este salto obtiene un valor más bajo, el regulador puede terminar pidiendo un valor muy grande en la próxima ejecución.

A modo de reflejar la importancia de realizar una colocación manual de algunos componentes de la FPGA, en la Figura 4-6 se muestra la interfaz gráfica que ofrece la herramienta Vivado para facilitar este proceso, donde se puede observar la FPGA dividida en *slices* y los recursos utilizados resaltados en colores claros. En concreto se muestra la configuración del Método 1.

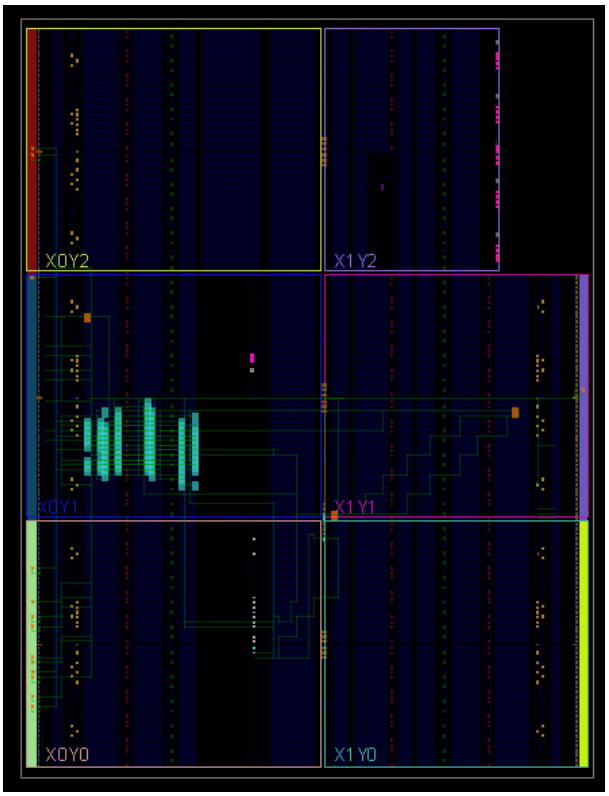

Figura 4-6: Configuración FPGA, Método 1.

Tras realizar la implementación del diseño y analizar en simulación los caminos que deberían ser modificados para compensar sus retrasos y obtener los resultados deseados, se ha procedido al cambio de ubicación del último *flip-flop* de esos caminos, pues así es más fácil controlar esos retrasos. Por ejemplo, en la Figura 4-7 y en la 4-8 se muestra el tramo final del camino de set, es decir, desde el último *flip-flop* hasta el registro de salida, antes y después de ser modificados. En la parte inferior de cada figura se observa el retardo de ese tramo. Por último, en la Figura 4-9 se han representado las medidas experimentales antes y después de realizar este proceso, así como las que deberían resultar en un entorno ideal. Se observa cómo los resultados son claramente peores si no se realiza dicho proceso.

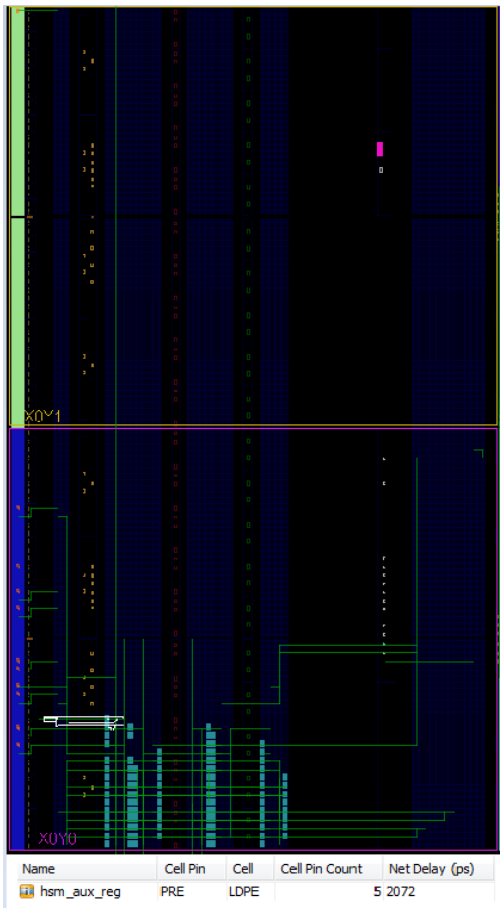

Figura 4-7: Camino de la señal de set antes de la reubicación del *flip-flop*.

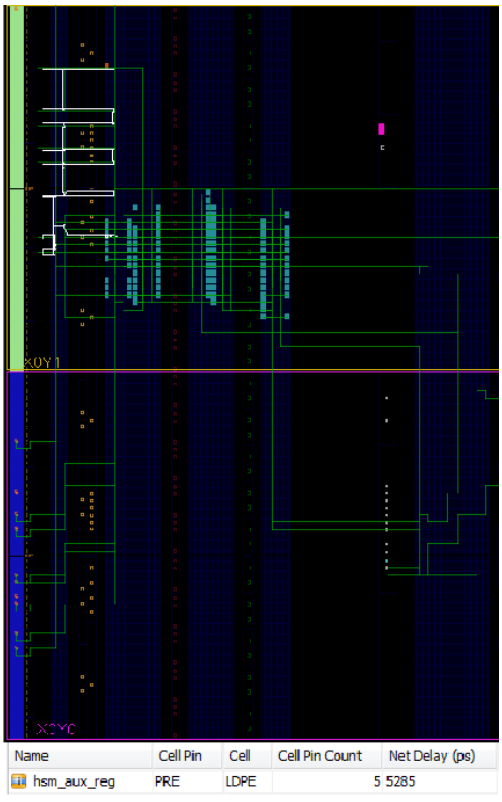

Figura 4-8: Camino de la señal de set tras reubicación del *flip-flop*.

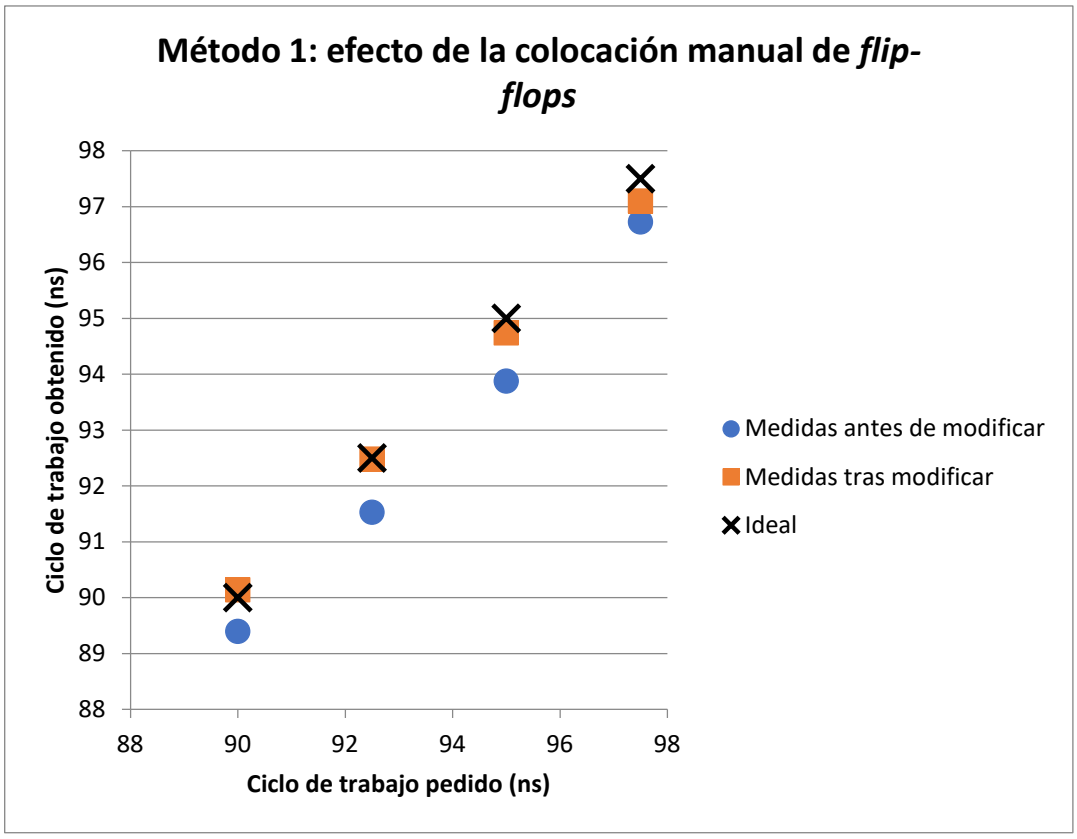

Figura 4-9: Resultados del Método 1 antes y después de reubicación componentes.

Finalmente, en la Tabla 4-1 se reflejan las resoluciones obtenidas experimentalmente (teniendo en cuenta la limitación comentada en el párrafo anterior), y en la Tabla 4-2 se recogen los recursos utilizados por cada método, incluyendo los recursos utilizados únicamente por el bloque síncrono.

Método	Resolución obtenida
Método 1	2,5 ns
Método 2	20 ps
Método 3	80 ps

Tabla 4-1: Resoluciones obtenidas.

Método	LUTs	Registros	DSP	Clocking Wizard	IDELAYE2
Método 1	352	647	0	1	0
Método1, reducción duty	350	642	0	1	0
Método 2	413	725	1	1	0
Método 3	355	662	0	1	3
Bloque Síncrono	348	639	0	0	0

Tabla 4-2: Recursos utilizados.

5 Conclusiones y trabajo futuro

5.1 Conclusiones

Tras la realización de este TFG, se han estudiado y puesto a prueba distintas técnicas de modulación PWM de alta resolución sobre una misma FPGA, destinando un conjunto de bits para indicar el número de ciclos de reloj en los que la señal PWM debe estar activa, más un conjunto de bits para añadir una fracción de ciclo de reloj adicional. Se ha cumplido el objetivo de obtener una resolución mínima de 2 bits de parte fraccional, consiguiendo hasta un máximo de resolución de 9 bits en el Método 2. Se ha puesto de manifiesto la importancia del estudio de estas técnicas ante el actual aumento de la frecuencia de trabajo que se está experimentando en el control digital mediante convertidores conmutados.

Todos los métodos se han implementado en una FPGA de la familia 7 de Xilinx con la que se aprovecha la flexibilidad y reprogramabilidad que nos brindan estas plataformas. Las técnicas que se han explorado han combinado las antiguas técnicas de modulación PWM basadas únicamente en contadores con otras más modernas basadas en el uso de líneas de retardo, que son las que han posibilitado añadir esa resolución extra a la señal PWM. En este trabajo estas líneas de retardo han tenido como elemento central en su implementación los componentes Clocking Wizard e IODELAY que ofrece Xilinx.

Todo el trabajo se ha implementado en VHDL y se ha simulado con ModelSim para la verificación de todos los sistemas. Una vez realizadas las simulaciones ideales se han realizado simulaciones posteriores a la implementación con objeto de asegurar la precisión y linealidad del sistema, recurriendo a una colocación manual de componentes críticos en los casos necesarios. Por último, se han realizado medidas experimentales utilizando el osciloscopio y se ha ofrecido una comparativa con los resultados ideales.

Tras la puesta a prueba de estas técnicas y su comparación se han conseguido 2,5 ns de resolución generando mediante el Clocking Wizard, 4 relojes con un desfase fijo y posteriormente combinados mediante la lógica adecuada. Por otra parte, el método 2 ha conseguido 20 ps de resolución mediante la utilización del Clocking Wizard para la generación de un reloj con un desfase variable mediante un protocolo efectivo pero lento y probablemente poco adecuado para su integración, debido precisamente a la lentitud del protocolo para modificar el desfase del reloj. Por otra parte, el tercer método ha obtenido 80 ps de resolución mediante el uso de los bloques IODELAY que ofrecen buenos resultados lineales y que se alcanzan con la velocidad necesaria. Cabe destacar que todos los sistemas se han probado son adecuados de implementar en FPGA de bajo coste como la utilizada, obteniéndose mejores resultados y más posibilidades en otras de gama más alta. No obstante, el método más útil es el tercer método debido a que obtiene una resolución de 80 ps (equivalente a 12,5 GHz), a la vez que la modificación del desfase es inmediata, no necesitándose un protocolo de tipo incrementa/decrementa para cada pequeño incremento de desfase.

5.2 Trabajo futuro

En la actualidad se están realizando numerosas investigaciones relacionadas con este TFG. En vista a los resultados obtenidos se podrían realizar trabajos futuros cuyo objetivo sea:

- Aumentar la resolución utilizando una FPGA de la familia Kintex-Ultrascale de la Serie 7 de Xilinx, adoptando una arquitectura similar a la del Método 3, con la diferencia de que los bloques de retardo, llamados IDELAYE3, son una evolución de los utilizados en este TFG (IDELAYE2). Estos nuevos bloques, que no se han utilizado debido a que el precio del dispositivo es muy elevado, permiten una resolución de 5 ps frente a los 78 ps que se han conseguido en este TFG.

- Además, haciendo uso de los bloques IDELAYE3, se resolvería el problema de linealidad del Método 3 mostrado en la Figura 5-4 ocasionado por el multiplexor, ya que permiten realizar el retraso de forma asíncrona de un ciclo de reloj completo. De esta manera bastaría con tener una única rama de reset y no sería necesario añadir un multiplexor.

- Probar todos los métodos propuestos integrándolos en un convertidor de potencia con elementos de conmutación de alta frecuencia para poder comprobar en lazo cerrado su viabilidad de uso.

Referencias

[1] E. Todorovich, A. Sanchez & A. de Castro, "An Application of the Hardened Floating-Point Cores on HIL Simulations", *X Southern Conference on Programmable Logic (SPL)*, pp. 113-119, abr 2019.

[2] M. Rodríguez, Y. Zhang, and D. Maksimovic. High-frequency PWM ´ buck converters using GaN-on-SiC HEMTs. IEEE Transactions on Power Electronics, 29(5):2462–2473, May 2014.

[3] J. G. Kassakian and T. M. Jahns. Evolving and emerging applications of power electronics in systems. IEEE Journal of Emerging and Selected Topics in Power Electronics, 1(2):47–58, June 2013.

[4] A. V. Peterchev and S. R. Sanders, "Quantization resolution and limit cycling in digitally controlled PWM converters," IEEE Trans. Power Electron., vol. 18, no. 1, pp. 301–308, Jan. 2003.

[5] D. Navarro, Ó. Lucía, L. A. Barragán, J. I. Artigas, I. Urriza, and Ó. Jiménez, "Synchronous FPGA-based high-resolution implementations of digital pulse-width modulators," *IEEE Trans. Power Electron.*, vol. 27, no. 5, pp. 2515–2525, 2012.

[6] A. Syed, E. Ahmed, D. Maksimović, and E. Alarcón, "Digital pulse width modulator architectures," *PESC Rec. - IEEE Annu. Power Electron. Spec. Conf.*, vol. 6, pp. 4689–4695, 2004.

[7] A. de Castro and E. Todorovich, "High Resolution FPGA DPWM Based on Variable Clock Phase Shifting," in *IIOAB Journal*, 2010, vol. 25, pp. 1115–1119.

[8] D. Navarro, Ó. Lucía, L. A. Barragán, J. I. Artigas, I. Urriza, and Ó. Jiménez, "Synchronous FPGA-based high-resolution implementations of digital pulse-width modulators," *IEEE Trans. Power Electron.*, vol. 27, no. 5, pp. 2515–2525, 2012.

[9] A. Syed, E. Ahmed, D. Maksimović, and E. Alarcón, "Digital pulse width modulator architectures," *PESC Rec. - IEEE Annu. Power Electron. Spec. Conf.*, vol. 6, pp. 4689–4695, 2004.

[10] D. Costinett, M. Rodriguez, and D. Maksimovi, "Simple Digital Pulse Width Modulator with 60 Picoseconds Resolution Using a Low-cost FPGA," pp. 1–7, 2012.

[11] V. Yousefzadeh, T. Takayama, and D. Maksimovi, "Hybrid DPWM with digital delay-locked loop," *Proc. IEEE Work. Comput. Power Electron. COMPEL*, pp. 142–148, 2006.

[12] S. C. Huerta, A. de Castro, O. García, and J. A. Cobos, "FPGA-based digital pulsewidth modulator with time resolution under 2 ns," *IEEE Trans. Power Electron.*, vol. 23, no. 6, pp. 3135–3141, 2008.

[13] M. Scharrer, M. Halton, and T. Scanlan, "FPGA-based digital pulse width modulator with optimized linearity," *Conf. Proc. - IEEE Appl. Power Electron. Conf. Expo. - APEC*, pp. 1220–1225, 2009.

[14] J. Quintero, M. Sanz, A. Barrado, and A. Lázaro, "FPGA based Digital Control with High-Resolution Synchronous DPWM and High-Speed Embedded A/D Converter," *Conf. Proc. - IEEE Appl. Power Electron. Conf. Expo. - APEC*, pp. 1360–1366, 2009.

[15] "7 Series FPGAs Clocking Resources", UG472 (v1.14) July 30, 2018, https://www.xilinx.com/support/documentation/user_guides/