Síntesis Física del Módulo de División de N bits con Segmentación v1.10

Walter Calienes Bartra*†, Manuel Monge Osorio*, Carlos Silva Cárdenas*, Ricardo Reis†.
wecbartra@inf.ufrgs.br, mmonge@pucp.edu.pe, csilva@pucp.edu.pe, reis@inf.ufrgs.br
*Pontificia Universidad Católica del Perú
Grupo de Microelectrónica GuE, Lima-Perú.
http://www.pucp.edu.pe/grupo/microelectronica
†Universidade Federal do Rio Grande do Sul,
Programa de Pós-Graduação em Microeletrônica PGMicro, Porto Alegre-Brasil.
http://www.inf.ufrgs.br/pgmicro

***Abstract*—For the implementacion of an algorithm such as Recursive Least Square (RLS) is necessary to develop a fast Arithmetic Logic Unit (ALU) because the data must be processed in real time. A good approach for that is using Pipeline in the ALU operations. For the RLS algorithm, the division is the most critical component. The use of Non-restoring Division Algorithm with Pipeline is a good approach to solve this problem. A generic and modular VHDL code have been developed and implemented in a FPGA and designed using a TSMC 180nm technology.**

I. Introducción

Entre las mútliples funciones que realizan los micoprocesadores y microcontroladores se tiene la división aritmética que entre las operaciones básicas resulta la mas compleja por la cantidad de recursos lógicos que se necesitan. Existen varios métodos para realizarla de acuerdo con [1], siendo el de División sin Recuperación el más rápido de todos estos [2].

EL presente articulo muestra la implementacion de un módulo de division génerico de n bits en un FPGA y su síntesis para ser llevado a un ASIC para la implementación de un filtro RLS [3]. Además presenta los resultados y conclusiones obtenidos durante su desarrollo.

El artículo se ha estructurado de la siguiente manera: en la sección 2 se explican los conceptos básicos de la División. La secciones 3 y 4 describen el algoritmo usado para este módulo y su funcionamiento. En la sección 5 se describe la arquitectura del módulo. En las secciones 6, 7 y 8 se muestran la arquitectura usada para la descripción del circuito en un FPGA y sus resultados de simulación. El articulo concluye con la seccion 9, la cual muestra las conclusiones obtenidas.

II. Conceptos básicos de la División

Los numeros enteros son finitos, pueden ser multiplicados y es posible predecir la longitud del resultado, más no así en la división. La precisión de esta debe ser definida antes y la cantidad de ciclos de reloj que requiere para realizarse depende de esta precisión [1]. Una división de números enteros requiere de un Dividendo A y un Divisor B. Entonces se puede definir a una cantidad Q llamada Cociente y otra cantidad R denominada Residuo las cuales cumplen con la siguiente ecuación:

$$A = B \times Q + R \tag{1}$$

Esta ecuación esta definida en todo el campo de los números enteros, donde $-B \leq R < B$ y $sgn(R) = sgn(A)$.

III. Descripción del Algoritmo

Este módulo implementa una División Entera de n bits sin Recuperación [1] [4] (conocida también como División por Desplazamiento), usando el método de segmentación (*pipeline*). Este módulo requiere de dos números: A y B de n bits cada uno los cuales son el Dividendo y el Divisor respectivamente y una señal de reloj CLK que sirve para activar los registros del *pipeline*. Las salidas de este módulo son dos números enteros de n bits: el Cociente Q y el Residuo R.

La División sin Recuperación es una técnica de división de números enteros en la cual al dividendo B de (1) se le colocan $n-1$ bits a la derecha de este (lo que es equivalente a multiplicar el valor de B por 2^{n-1}) y se crea un contador de iteración $i = n-1$ para la operación. Se compara A con el nuevo B. Si el valor de A es mayor o igual al nuevo valor de B; entonces el bit i de Q será igual a 1 y A tomará el valor de $A-B$; caso contrario el bit i de Q será 0 y A permanecerá inalterado. Se decrementa a i en uno y se desplaza a B hacia la derecha un bit. Se revisa que el contador i no sea negativo; si no lo fuese se vuelve a hacer la comparación de A mayor igual a B, pero si i es negativo entonces se asigna a R el valor actual de A y se concluye la división. En la Figura 1 se muestra el diagrama de flujo de este algoritmo.

IV. Funcionamiento del Módulo

A continuación se enumeran las condiciones para que este módulo funcione apropiadamente. Se debe tener muy en cuenta la cantidad de bits que tendrá cada uno de los operadores de división debido a su forma modular y descripción genérica en VHDL.

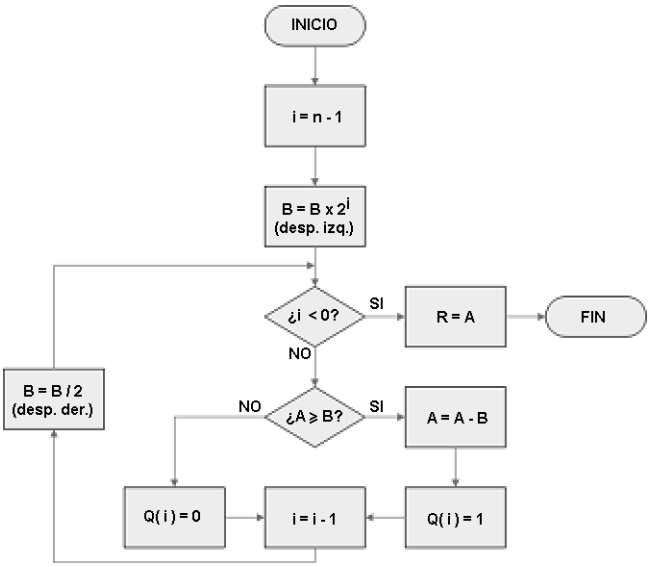

Figura 1. Diagrama de flujo del algoritmo de División.

- Se debe fijar el número de bits dentro del código del módulo (esto se hace en la línea `GENERIC(n:INTEGER:=8);` se cambia el ”8” por el número de bits que se desee). Por defecto es 8.
- Se requiere obligatoriamente de una señal de reloj ”CLK” para que funcionen los registros de *pipeline*. Por cada n bits se crean n niveles de registros de *pipeline*.
- Se debe cumplir que el dividendo A sea mayor que el divisor B para que se realice la división entera de forma correcta tal como se indica en (1), de no ser así Q tomará el valor ”0” y R tomará el valor de A.
- Cualquier división entre cero (lo que es lo mismo: $B = 0$) hará que Q tome el valor de $2^n - 1$ y R tome el valor de A.

V. Arquitectura del Circuito

La descripción del circuito se hizo siguiendo el diagrama de flujo mostrado en la Figura 1. Se puede observar un ciclo el cual se repite un número de veces igual al número de bits con que cuentan los operadores A y B de la división. Entonces esto hace suponer la modularidad de la arquitectura del circuito debido a la recurrencia en las mismas operaciones cada vez que se opera un bit del cociente Q. En la Figura 2 se muestra la arquitectura del circuito para un número de bits $n = 4$. Las operaciones de resta, comparación y desplazamiento a la derecha se repiten por cada bit de Q comenzando con el bit más significativo (en este caso de $n = 4$, Q tiene también 4 bits numerados del 3 al 0). El multiplexor de bus en cada etapa es clave para generar un nuevo valor temporal de A para poder continuar con la operación y obtener el residuo R al finalizar esta. A la salida de cada comparador, desplazador y multiplexor se colocan registros para generar el *pipeline*; haciendo esto se garantiza de que el resultado de la operación se obtendrá n ciclos de reloj después.

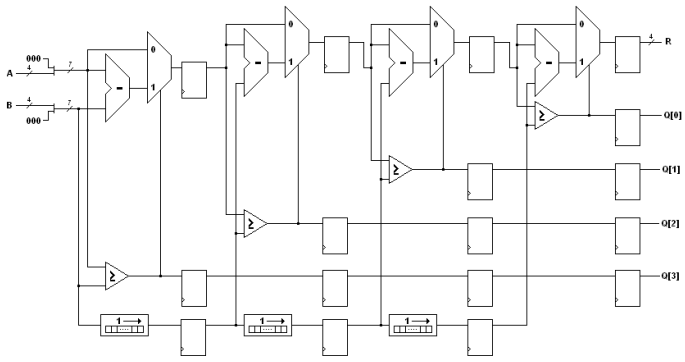

Figura 2. Arquitectura de la División de 4 bits.

VI. Resultados de Síntesis en FPGA

Este módulo fue desarrollado en lenguaje de descripción de hardware VHDL. Los resultados obtenidos con esta división se probaron en un FPGA Altera Cyclone II EP2C35F672C6 y simulado en el Software Quartus II v8.0. A continuación se expondrán los resultados obtenidos durante la simulación de este circuito en dicho FPGA. Los resultados fueron iguales tanto en modo de optimización Balanceada, de Área y de Velocidad.

A. Frecuencia Máxima de Operación

Los resultados de desempeño para operadores de 8, 16, 24 y 32 bits se muestran en la Tabla I. Puede notarse que la Máxima Frecuencia de Operación casi es inversamente proporcional al Número de Bits del Módulo de División, esto debido a la existencia de los restadores y multiplexores que crecen debido al incremento de bits en los operadores. La Figura 3 grafica esta tendencia. Esta tendencia era algo que se esperaba debido al incremento de elementos lógicos al incrementarse la cantidad de bits de los operadores.

Tabla I
Resultados de Compilación de la División con distintos valores de bits para un FPGA Altera Cyclone II EP2C35F672C6.

Núm. Bits	Núm. Elem. Lógicos	Núm. Registros	Frec. Máxima
8 bits	221	205	245.40MHz
16 bits	1029	857	182.32MHz
24 bits	2066	1957	157.95MHz
32 bits	4189	3505	135.46MHz

B. Potencia Disipada

En la Figura 4 se muestran los resultados de Disipación de Potencia del FPGA sin sistemas de ventilación como para uno que tenga un disipador de calor de 23mm de lado y un flujo de ventilación forzada de 200 pies por minuto. Ambas fueron simuladas en un ambiente de 25°C. En este caso, la tendencia es prácticamente lineal.

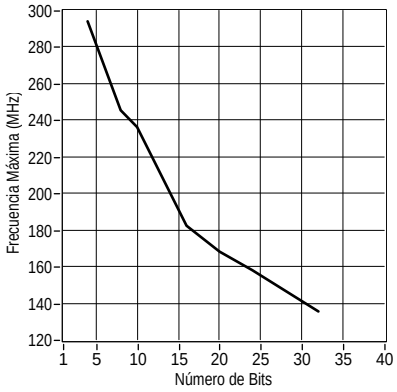

Figura 3. Frecuencia Máxima de Operación (f_{max}) del Módulo de División de N Bits implementado en un Altera Cyclone II EP2C35F672C6.

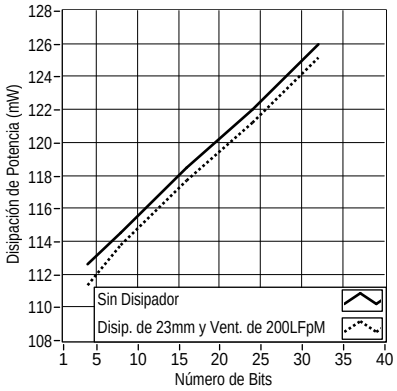

Figura 4. Disipación de Potencia (P_{dis}) del Módulo de División de N Bits implementado en un Altera Cyclone II EP2C35F672C6 ($T_{amb} = 25°$C).

C. Simulación

En la Figura 5 se muestra los resultados de la simulación usando el software Quartus II v8.0 de Altera y un número de bits de $n = 16$ para los operadores. En esta simulación se puede ver claramente como el resultado de la operación se obtiene luego de transcurridos n ciclos de reloj y el efecto de una división entre cero. Además se puede ver lo que ocurre en esta simulación cuando un dato es colocado durante la transición del reloj.

VII. Sintesis Lógica

Para realizar la síntesis lógica de este circuito se decidió, de acuerdo con [3], hacerlo de $n = 16$ bits; además se decidió que este tuviese un tope de frecuencia máxima de 200MHz con un reloj de 0.5ps tanto de tiempo de subida (t_r) como de bajada(t_f) para asegurar su confiabilidad y buen desempeño en frecuencias más bajas. La tecnología usada fue la TSMC de 180nm.

A. Compilación RTL

Para entrar a esta etapa, se requirió de la descripción en VHDL, las librerías de tecnología y de limitaciones que debe tener el el circuito en cuanto a la frecuencia máxima del reloj que se le pretende colocar. Para empezar a realizar la síntesis lógica se precisó que el circuito descrito en lenguaje VHDL pase a ser compilado y transformado en una descripción Verilog, la cual respete los reglas de Lógica de Transferencia de Registros o *RTL* [5], para que este pueda ser transformado inalmente en un módulo físico. Para esto se utilizó la herramienta *RTL Compiler* de *Cadence*. Esta herramienta también necesita tener acceso a los archivos de tecnología de Celdas Estandar para que pueda sintetizar el diseño respetando las reglas de diseño y las limitaciones definidas anteriormete. Esta, luego de realizar la síntesis lógica, entregó resultados estimativos de área, potencia y slacks, los cuales se presentan en las Tablas II, III y IV.

Tabla II
Área Estimada para el Módulo de División a $n = 16$ bits.

Área de Celdas	Área de la Red	Cantidad de Celdas
99732	42991	4491

Tabla III
Consumo de Potencia Estimada (nW) para el Módulo de División a $n = 16$ bits.

Pot. de Fuga	Pot. Interna	Pot. de la Red	Pot. Conmutación
2330.5	9595952.9	591697.7	10547650.6

Tabla IV
Slacks Estimados para el Módulo de División a $n = 16$ bits.

Grupos de Costo	Sint. Post Incremental	Sint. Post Mapeamiento
Entrada a Registro	963ps	1744ps
Registro a Registro	0ps	92ps
Registro a Salida	4171ps	4171ps

La Tabla II muestra el área referenciada a una celda estandar patrón de 5um de altura, mientras que en la Tabla IV se puede notar una cantidad de tiempo de *slacks* positivos cercanos a cero, lo que significa que se estima que el circuito tiene su límite de frecuencia a los 200MHz que se le definió en un principio.

B. Verificación Formal de la Lógica

Esta etapa es necesaria para asegurar que la descipción hecha en VHDL sea la misma que se sintetizó en la etapa de Compilación Lógica. Para esta tarea se requiere de tener la descripción VHDL original y la descripción Verilog del circuito basado en celdas estandar sintetizado previamente con el *RTL Compiler*. Para verificar todo esto se usó la herramienta *LEC Conformal* de *Cadence*. De acuerdo con los reportes de esta herramienta, existen diferencias entre la descripción VHDL y la de Verilog generada en la etapa anterior en cuanto a jerarquía, esto debido a que el *RTL Compiler* sintetizó 16

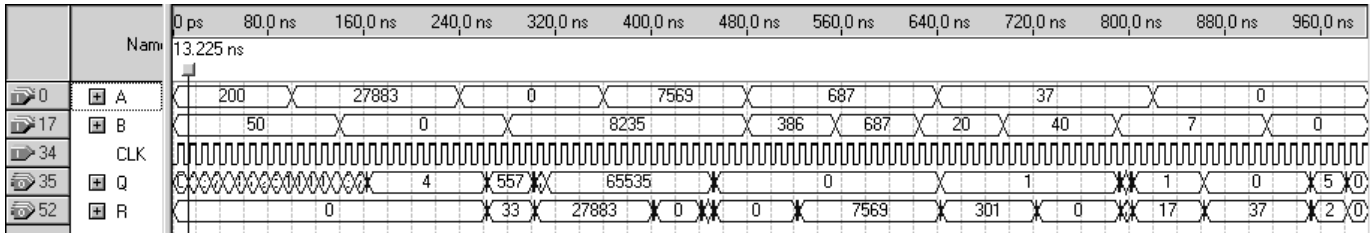

Figura 5. Simulación del Módulo de División para $n = 16$ bits.

módulos de substracción que no estaban descritos de forma literal en la descripción VHDL original, algo que era de esperarse de acuerdo a la arquitectura que se propuso en un principio.

VIII. Síntesis Física

Luego de culminar la etapa de Síntesis Lógica de manera satisfactoria, se procedió a sintetizar el Módulo de División de $n = 16$ bits en un módulo físico, listo para ser colocado en cualquier proyecto en que se requiera. Para realizar esta tarea se usara la herramienta *SoC Enconter v6.2* de *Cadence*. Para esto se requirió de cargar el diseño previamente sintetizado por el *RTL Compiler*, al hacerlo se presentará un Entorno Límite (conocida como *Bouding Box*) donde se iniciará la labor de desarrolo físico del chip.

A. Planeamiento de Base

En esta etapa (conocida comunmente como *Floorplanning*) se definieron las medidas de los márgenes para colocar los anillos y mallas (*Stripes*) de alimentación y cuales son los metales con que se van a hacer estos. Se eligió un margen de 10um de ancho para colocar los anillos de poder y un porcentaje de utilización de espacio del 70% con el origen de coordenadas en la esquina inferior izquierda de la Entorno Límite que contendrá al circuito.

B. Planeamiento de Potencia

Aquí se deben colocar los anillos y mallas para alimentar a las celdas lógicas dentro del módulo. Para esto, se definió el ancho de los metales de los anillos de VDD y VSS a 2um, con una separación de 2um y centrados en el canal del margen definido anteriormente en el Planeamiento de Base. Los metales usados para hecer estos anillos fueros los metales 5 y 6 (que son, en la tecnología TSMC, los de mayor rango). Luego de esto, se pasó a diseñar los *stripes* de la malla de manera de tener 1 par de ellos hechos con el metal 2 (el cual es uno de los metales de más bajo rango en la tecnología TSMC), cada línea posee un ancho de 2um y una separación entre líneas de cada conjunto de 2um. Debido a que las celdas tienen una altura normada de 5um, se hicieron los *stripes* horizontales usando un ruteamiento especial debido a que estas tienen las alimentaciones en los lados horizontales de estas. Para ello se elije al metal 1 y se dejó que la herramienta conecte todas las vias de alimentación tanto a los anillos como a los *stripes* anteriormente creados. Con esto ya se tuvo completada la base de potencia del módulo.

C. Emplazamiento de Celdas Estándar

Se ordenó a la herramienta que coloque las celdas estándar teniendo en cuenta el portentaje de utilización descrito en el Planeamiento de Base. Se le indicó además que haga un emplazamiento completo, el cual contempla un emplazamiento previo y luego uno incremental; este último mejora la calidad tanto del circuito en si como de su temporización. El resultado de esta etapa se puede ver en la Figura 6. Al final de este emplazamiento se pudo notar que la mayoría de las celdas ocupan la zona diagonal del Entorno Límite definido por el *RTL Compiler*.

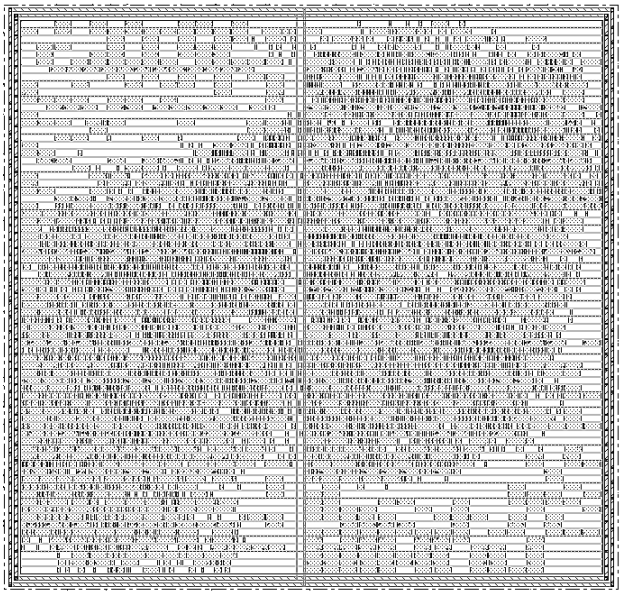

Figura 6. Emplazamiento Completo de Celdas del Módulo de División a $n = 16$ bits.

D. Sintesis de Reloj

Para esta etapa la herramienta procededió a sintetizar un Árbol de Reloj (conocido como *Clock Tree Syntesis* o CTS) con las limitaciones dadas en un principio ($t_f = t_r = 0.5ns$ y 200MHz). El resultado de esta síntesis puede verse en la Figura 7. Esta CTS no sólo sintetizó el cableado del reloj,

también agregó celdas para corregir retrasos inadmisibles a las limitaciones dadas y re distribuyó las celdas secuenciales y combinacionales para mejorar el paso de dicha red, la cual tiene prioridad en el emplazamiento del cableado.

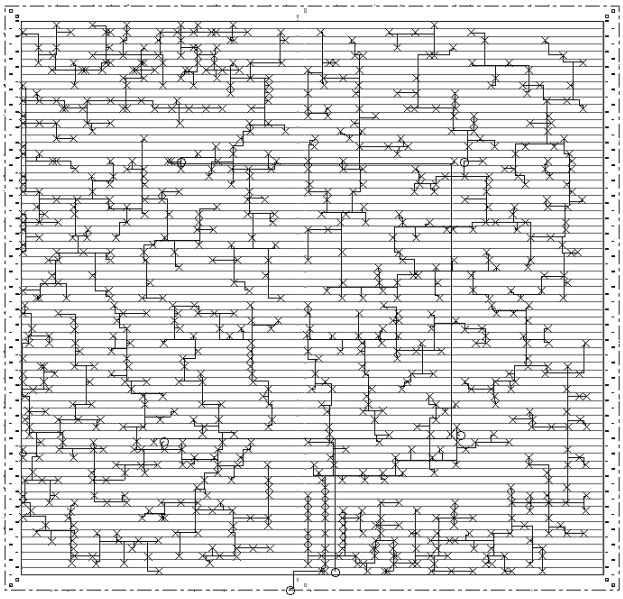

Figura 7. Síntesis del Árbol de Reloj del Módulo de División a $n = 16$ bits.

E. Enrutamiento de las Redes del Circuito y Análisis de Congestión

Luego de haber hecho el Planeamiento de Base, diseñar el Plan de Potencia, emplazar las Celdas Estandar de la librería de tecnología y terminar de hacer la CTS se pudo empezar a hacer el Enrutamiento de las Redes de señales entre las celdas (esta etapa del proceso es conocida como *Routing*). Para esto se eligió hacer un *Nano Routing*, debido a que este contempla hacer un enrutamiento trivial, luego hace un enrutamiento global y finalmente realizar un enrutamiento detallado para corregir el ancho de los metales y recolocarlos para respetar las reglas de diseño y limitaciones para la tecnología usada y las planteadas inicialmente, además de la corrección del enrutamiento para evitar el Efecto Antena. Antes de concluir esta etapa se hizo un Análisis de Congestión el cual indica el porcentaje de que líneas de metal estan pasando juntas por una misma unidad cuadrada de área llamada Celda Global (*Gcell*), estos resultados pueden apreciar en la Tabla V. Se puede notar que la sobre congestión máxima no llega al 1% y es mayormente causada por líneas de Metal 2.

F. Análisis y Optimización de Temporización

Luego de hacer el enrutamiento completo del circuito se realizó un análisis de Temporización para corroborar que el reloj actua dentro de las limitaciones establecidas de 200MHz. Al hacer el análisis se obtuvieron los resultados mostrados en la Tabla VI. Este análisis fue realizado sólo para el tiempo de *Setup*. El tiempo de *Hold* pasó las pruebas de *slacks* negativos sin problemas.

Tabla V
ANÁLISIS DE CONGESTIÓN.

Capa	Sobr. Cong. # Gcell (1-2)	Sobr. Cong. # Gcell (3-4)	Sobr. Cong. # Gcell (5-6)	% Gcell Sobr. Cong.
Metal 1	0(0.00%)	0(0.00%)	0(0.00%)	(0.00%)
Metal 2	91(4.92%)	14(0.76%)	5(0.27%)	(5.95%)
Metal 3	0(0.00%)	0(0.00%)	0(0.00%)	(0.00%)
Metal 4	0(0.00%)	0(0.00%)	0(0.00%)	(0.00%)
Metal 5	0(0.00%)	0(0.00%)	0(0.00%)	(0.00%)
Metal 6	0(0.00%)	0(0.00%)	0(0.00%)	(0.00%)
Total	91(0.82%)	14(0.13%)	5(0.05%)	(0.99%)

Tabla VI
ANÁLISIS DE TEMPORIZACIÓN ANTES DE LA OPTIMIZACIÓN PARA EL TIEMPO DE *Setup*.

Modo *Setup*	Total	Reg. a Reg.	Ent. a Reg.	Reg. a Sal.
WNS(ns)	-0.088	-0.088	1.318	3.392
TNS(ns)	-0.350	-0.350	0.000	0.000
Rut. con Viol.	6	6	0	0
Rutas Totales	1593	1496	65	32

En la Tabla VI puede notarse la existencia de un Peor *Slack* Negativo (WNS) de -0.088ns y un total de *Slacks* Negativos de -0.35ns causados por 6 rutas del CTS que deben ser optimizados para poder llegar a la meta de *Slacks* positivos cercanos a cero. Luego de usar la opción de optimización de la herramienta se obtuvieron los resultados de la Tabla VII, los cuales son optimos para que el circuito pueda tener un reloj de 200MHz como máximo.

Tabla VII
ANÁLISIS DE TEMPORIZACIÓN LUEGO DE LA OPTIMIZACIÓN PARA EL TIEMPO DE *Setup*.

Modo *Setup*	Total	Reg. a Reg.	Ent. a Reg.	Reg. a Sal.
WNS(ns)	0.010	0.010	0.487	3.289
TNS(ns)	0.000	0.000	0.000	0.000
Rut. con Viol.	0	0	0	0
Rutas Totales	1593	1496	65	32

Adicionalmente, luego de la optimización, se obtuvieron los datos físicos de las redes del circuito los cuales son mostrados en las Tablas VIII y IX.

G. Análisis de Potencia Estimada y por Simulación

Luego de realizar el análisis de Temporización la herramienta proporciona una forma de ver el consumo de potencia

Tabla VIII
LONGITUD (μm) DE METALES DEL MÓDULO DE DIVISIÓN PARA $n = 16$ BITS.

Metal 1	Metal 2	Metal 3	Metal 4	Metal 5	Metal 6	Total
4285	41549	51549	25306	3621	574	119341

Tabla IX
NÚMERO DE VIAS DEL MÓDULO DE DIVISIÓN PARA $n = 16$ BITS.

Metal 1-2	Metal 2-3	Metal 3-4	Metal 4-5	Metal 5-6	Total
15866	15124	4638	449	45	36122

estimada del circuito , de forma pesimista usando la mitad de la máxima frecuencia de funcionamiento. En la Tabla X se pueden ver estos resultados, donde se puede apreciar que el máximo consumo en el peor caso es de 23.88mW. Para realizar el análisis de potencia por simulación se requiere de extraer todas las capacitancias y resistencias de cada red del módulo, además de todos los retardos de cada celda. Además de esto es preciso hacer un archivo de banco de prueba (*testbench*) en el cual se datallarán los valores de las entradas A y B en determinados tiempos y un cálculo de retardos de las etapas del circuito para poder generar un archivo de forma de onda VCD. Este archivo se coloca como entrada al Analizador de Potencia de la herramienta para que nos informe acerca de la potencia promedio. Los resultados finales de este análisis se detallan en la Tabla XI y se puede ver que el máximo consumo luego de la simulación es de 2.1565mW, casi el 10% de la Potencia Estimada Total. Cabe mencionar que las señales que se colocan al archivo de *testbench* son críticas para realizar un adecuado Análisis de Potencia. Estos dos análisis se hicieron considerando una tensión $V_{DD} = 1.62V$, temperatura $T_{amb} = 125^oC$ y una frecuencia de reloj de $200MHz$.

Tabla X
POTENCIA ESTIMADA (mW) DEL MÓDULO DE DIVISIÓN PARA $n = 16$ BITS PARA $V_{DD} = 1.62V$, $T_{amb} = 125^oC$ Y $200MHz$.

Pot. de Fuga	Pot. de Conmutación	Pot. Interna	Potencia Total
0.0024134	7.12762	16.750	23.880

Tabla XI
POTENCIA PROMEDIO POR SIMULACIÓN(mW) DEL MÓDULO DE DIVISIÓN PARA $n = 16$ BITS PARA $V_{DD} = 1.62V$, $T_{amb} = 125^oC$ Y $200MHz$.

Pot. de Fuga	Pot. de Conmutación	Pot. Interna	Potencia Total
0.0024131	0.42027	1.7339	2.1565

IX. CONCLUSIONES

Finalmente este módulo tiene 65 terminales, un área de $158019\mu m^2$, 4890 redes cableadas, se usaron 4511 celdas estandar y se sintetizaron 16 sub módulos usando la tecnología TSMC de 180nm.

Es muy útil para un diseñador contar con un código de división genérica y modular que pueda ser modificado dependiendo de las necesidades del proyecto.

Si se hace una comparación con la Figura 4 a $n = 16$ bits, la tabla III, la tabla X y la tabla XI se puede notar la diferencia entre una implementación en FPGA, una estimación por tecnología y una síntesis física, haciendo a la síntesis física mejor en consumo de potencia, mas si se observan las tablas IV y la tabla VII se debe considerar el gran esfuerzo que se debe hacer para llegar a la meta de *slacks* positivos que tiendan a cero y con esto asegurar que el circuito llegue a funcionar correctamente a un máximo de 200MHz.

El uso de la segmentación o *pipeline* no solo aumenta la complejidad del diseño, sino que también le da una mejoría en cuanto al proceso de los datos que entran al módulo ya que las respuestas tardan una cantidad de ciclos de reloj igual al número de bits de los operadores de entrada. Si bien es cierto la lógica combinacional y el incremento del número de bits en los operadores limita la máxima velocidad del módulo, este es útil para realizar aplicaciones como Unidades Aritméticas para tratamiento de señales de voz o en procesos de mediana frecuencia. Se puede mejorar este módulo usando una tecnología más pequeña, mas si se hace eso entonces ocurren fenómenos como electromigración, ruido por acoplamiento, problemas de integridad de señal y caidas de tensión inadmisibles los cuales son casi imperceptibles en tecnologías de mayor tamaño.

Finalmente, este circuito se seguira mejorando para cubrir algunos faltos como la introducción de pines de reinicio, mejoras en la congestión de pistas, análisis de *IR Drop* y electromigración, así como generar verificaciones finales del diseño. También se proyecta hacer una descripción para este en Verilog, hacer parametrizables y genéricas las etapas de *pipeline* y la colocación de registros a las entradas A y B para aumentar su estabilidad.

AGRADECIMIENTOS

Los autores agradecen al Instituto de Informática de la Universidad Federal de Rio Grande del Sur por el uso de sus laboratorios para el desarrollo del presente artículo. También se agradece al Ing. Jorge L. Tonfat Seclén, al Ing. Jose F. Quenta Cuno, a Glauco Borges Valim dos Santos, a Leandro Max, al prof. Sergio Bampi y a todos aquellos que laboran en el Laboratorio 67-219 del Instituto de Informática de la UFRGS por su incondicional apoyo.

REFERENCIAS

[1] J. P. Deschamps, G. J. A. Bioul, G. D. Sutter, *Synthesis of Arithmetic Circuits*, 3rd ed. Wiley-Interscience, 2006.
[2] Nikolay Sorokin, *Implementation of high-speed fixed-point dividers on FPGA*. Pacific National University, Tikhookeanskaya str., 136, Khabarovsk - Russia, 2006.
[3] J. Benavides, W. Calienes, C. Silva, *Diseño de una Arquitectura para la Implementación de un Filtro Adaptativo RLS sobre un FPGA*. XV Workshop Iberchip, Buenos Aires - Argentina, 2009.
[4] V. A. Pedroni, *Circuit Design with VHDL*. MIT Press, 2004.
[5] P. P. Chu, *RTL Hardware Design Using VHDL: Coding For Efficiency, Portability, And Scalability*. Wiley-IEEE Press, 2006.