ESTUDIO COMPARATIVO DE LOS DIVISORES EN TECNOLOGÍAS CMOS NANOMÉTRICAS

Gashaw Sassaw(1), Carlos J. Jiménez(2), José M. Mora(3) y Manuel Valencia(4)

1 Instituto de Microelectrónica de Sevilla, España, sassaw@imse.cnm.es,

2 Instituto de Microelectrónica de Sevilla / Universidad de Sevilla, España, cjesus@imse.cnm.es,

3 Instituto de Microelectrónica de Sevilla, España, jmiguel@imse.cnm.es,

4 Instituto de Microelectrónica de Sevilla / Universidad de Sevilla, España, manolov@imse.cnm.es,

RESUMEN / *ABSTRACT*

Son varios los algoritmos de divisores propuestos para su realización en hardware, sin que haya un 'mejor divisor'. La búsqueda de un diseño óptimo para cada aplicación específica hace que sea indispensable la investigación de los algoritmos existentes a medida que se produce el avance de la tecnología.
En este trabajo se presentan los resultados de la caracterización en área, tiempo y consumo de potencia de varias implementaciones de divisores en tecnologías CMOS nanométricas de 90 y 65 nm. Para la implementación se ha utilizado un flujo de diseño ASIC semicustom con elección entre tres voltajes umbrales.

Palabras claves: División binaria, divisores, VLSI nanométricos, caracterización en área, tiempo y consumo de potencia.

Several algorithms have been proposed for the hardware implementation of the division operation, without concluding "the best one". As the technology evolves, there is a never ending need to explore design tradeoffs and alternatives on existing division algorithms.
This paper presents the characterization results for the most common digit recurrence division algorithms in 90 and 65 nm CMOS nanotechnologies using ASIC semicustom design flows and triple different voltage (VT) device, measuring area and power consumption. This paper surveys different implementations of dividers in two CMOS nanotechnologies.

Key words: *Digital division, dividers, nanometer VLSI, area, timing, and power characterization.*

Titulo en Inglés : *COMPARATIVE STUDY OF DIVIDERS IN CMOS NANOTECHNOLOGIES*

INTRODUCCION

Una percepción común hasta hace unos diez años era que la división no necesita un hardware específico al no requerirse esa operación con mucha frecuencia. Sin embargo, se ha mostrado que ignorar su implementación puede tener como resultado una significativa degradación de las prestaciones en los procesadores y aplicaciones actuales [1]. Por esta razón, al requerirse altas prestaciones hay que introducir un circuito específico, el divisor, que acelere la operación de la división.

Hasta ahora, diferentes trabajos han propuesto varias clases de algoritmos e implementaciones hardware específicas de divisores [2] [3]. La forma más clásica de realizar divisores de alta velocidad se basa en obtener dígitos de forma recurrente. De esta idea han surgido distintas formas de realizar en hardware la división. Los algoritmos de dígitos recurrentes requieren que en cada iteración el número de bits del cociente que se obtenga como resultado sea fijo. La implementación de estos algoritmos típicamente no requiere gran complejidad y ocupan un área relativamente pequeña. Por contra, la latencia suele ser relativamente alta. La implementación más común de los divisores con algoritmo de dígitos recurrentes en hardware se conoce como algoritmo SRT en honor a las iniciales de los nombres de sus tres autores, Sweeney, Robertson y Tocher [4] [5].

La implementación microelectrónica de divisores depende fuertemente de la tecnología de integración. Por ejemplo, en

las actuales tecnologías CMOS nanométricas, el área de silicio es un parámetro cada vez menos restrictivo y los transistores conmutan a mayor velocidad, pero el consumo de potencia es una preocupación cada vez más importante que suele ir en detrimento de la velocidad o el coste en área. De aquí que sean necesarios nuevos estudios y revisiones de las diferentes estructuras de divisores.

Actualmente las tendencias de estudio principales se centran en la influencia de la aplicación y de la tecnología en el desarrollo de divisores. En esa línea, recientemente Sutter y otros [6 7], han comparado los resultados de varios algoritmos de divisores llevados a cabo en dos familias de dispositivos FPGAs. Más recientemente Pham y Swarzlander [8] han publicado resultados de síntesis en la tecnología CMOS de 65 nm, de un divisor con el algoritmo SRT base-4.

El objetivo del presente trabajo es caracterizar la implementación de varios algoritmos de divisores en dos tecnologías CMOS nanométricas, de 90 y 65 nm. Por una parte, además del propio diseño en sí en estas tecnologías, es interesante caracterizar esos circuitos para discernir la influencia tecnológica. Por otra parte, el estudio abarca las principales propuestas de divisores conocidas, lo que permitirá elegir cuál es la mejor para cada aplicación.

Este trabajo está organizado presentando, tras esta introducción, los algoritmos de divisores. En estos deben distinguirse las propuestas básicas y las que surgen para alta velocidad. Ambos tipos de propuestas deben ser tenidos en cuenta ya que el objetivo de bajo consumo suele ir en contra de la alta velocidad. Una vez presentados los algoritmos, mostramos nuestras implementaciones en tecnologías nanométricas antes mencionadas. Los resultados de la caracterización en área, tiempo y consumo de potencia de los circuitos resultantes son presentados a continuación, para terminar con las conclusiones más relevantes.

La forma más común de determinar el cociente y el resto de una operación de división es a través de la obtención de los dígitos de forma recurrente. Los algoritmos de dígitos recurrentes pueden ser implementados con circuitos secuenciales, donde un número fijo de bits del cociente se obtienen en cada ciclo del reloj. En este trabajo han sido clasificados en algoritmos básicos y de alta velocidad.

En este trabajo se asume que la nomenclatura tiene la siguiente correspondencia: A : dividendo, D: divisor, Q: cociente y R. resto. En todos los algoritmos de divisores, con el objetivo de prevenir el desbordamiento durante el proceso de la división, se tiene que cumplir la condición A < 2n-1D donde A es el dividendo, el D el divisor y n el numero de bits.

2. ALGORITMOS BÁSICOS

En los algoritmos básicos se han incluido tres algoritmos: Resta con restauración, nonperforming y resta sin restauración. La diferencia fundamental entre ellos es la forma de restaurar el resto parcial.

2.1 Algoritmo de resta con restauración

Con este algoritmo, cuyos pasos están detallados en la figura 1 el proceso de la operación de división se lleva a cabo de la siguiente forma: Antes de iniciar la operación, el valor del dividendo es asignado a un registro resto parcial. Luego se realiza una operación de resta del valor del divisor con el resto parcial desplazado un lugar a la izquierda. El resultado de la resta se toma como residuo temporal. De acuerdo al signo del residuo temporal se establecerá el valor correcto del cociente y del resto parcial. Si el signo del residuo temporal es positivo, se asigna un "1" al bit del cociente. De lo cantrario se asigna un "0". Además, si el residuo temporal es negativo, el resto parcial se debe restaurar a través de la suma del residuo temporal y el valor del divisor.

Inicializar R(0) = A
Operación recursiva
For i=0 … n-1
Rtemp(i+1) = 2R(i) – D
If Rtemp(i+1) ≥ 0 then
Q_{n-1-i} =1 y R(i +1)=Rtemp(i+1)
Else
Q_{n-1-i} =0 y R(i +1)=Rtemp(i+1) + D
End for

Fig. 1 Pasos de la resta con restauración

2.2 Algoritmo *Nonperforming*

Con este algoritmo se logra mejorar la velocidad de operación por medio de la reducción de la operación de suma que era necesario para restaurar el resto parcial en la variante anterior. Los pasos que hay que seguir para llevar a cabo la división son similares a los del algoritmo anterior salvo en el caso que el resto parcial sea negativo. Cuando el resto parcial es negativo la restauración del resto parcial se realiza recuperando el valor previo del resto parcial. Los pasos se ilustran en la figura 2.

Inicializar R(0) = A
Operación recursiva
For i=0 … n-1
Rtemp(i+1) = 2R(i) – D
If 2R(i) – D ≥ 0 then
Q_{n-1-i} =1 y R(i +1)=2R(i) - D
Else
Q_{n-1-i} =0 y R(i +1)= 2R(i)
End for

Fig. 2: Pasos del non-performing

2.3 Resta sin restauración

Este algoritmo elimina la necesidad de restaurar el resto parcial. De acuerdo con lo ilustrado en la figura 3, después de cada operación de resta se verifica el signo del resto parcial desplazado. Si el resto parcial desplazado es positivo, se asigna un "1" al digito del cociente. El siguiente resto parcial se calcula a través de la resta del valor divisor del resto parcial desplazado. En el caso que el resto parcial desplazado sea negativo, el dígito del cociente es "0" y el siguiente resto

parcial se calcula con la suma del resto parcial desplazado y el valor del divisor.

Inicializar R(0) = A
R(1) = 2R(0) - D
Operación recursiva
For i=0 … n-1
If R(i) $\geq$ 0 then
Q_{n-1-i} =1 y R(i +1)=2R(i) - D
Else
Q_{n-1-i} =0 y R(i +1)= 2R(i) + D
End for
Corrección
if R(n) < 0 then
q_0 = 0 y R(n) = R(n) + D

Fig. 3: Pasos de la resta sin restauración

2. ALGORITMOS DE ALTAS PRESTACIONES

3.2. Algoritmo SRT

Este algoritmo fue propuesto para acelerar la velocidad de operación del algoritmo de resta sin restauración. En este algoritmo se asume que los operandos A y B son números fraccionarios normalizados. Según la figura 4a, el algoritmo de resta sin restauración considera números con signo como posibles valores del cociente. En vez de un "0" se considera un "-1" cuando el resto parcial desplazado es negativo, y un "1" cuando el resto parcial es positivo. La regla de selección del cociente se modifica según se muestra en la figura 4b añadiendo un "0" como otro de los posibles valores del cociente. El dígito del cociente tendrá un valor "0" cuando el resto parcial desplazado se encuentra entre "D" y "-D". El algoritmo RST es útil cuando el valor del divisor está restringido entre ½ y 1. La gran ventaja de esta modificación radica en que la comparación del resto parcial se limita a dos bits en vez del valor del divisor como en los casos anteriores. La selección del conciente y el cálculo del resto parcial se llevan a cabo a través de la siguiente ecuación:

$$q_i = \begin{cases} 1 & \text{si } 2R \geq 1/2 \\ 0 & \text{si } -1/2 \leq 2R(i) < 1/2 \\ -1 & \text{si } 2R < -1/2 \end{cases} \quad (1)$$

$$R(i+1) = 2R(i) - q_i D$$

Como los bits del cociente se generan en un formato de números con signo, es necesario realizar una conversión a un formato convencional. La conversión se puede llevar a cado de dos formas: una, al final de todas las iteraciones y otra, cada vez que se produzca un digito del cociente.

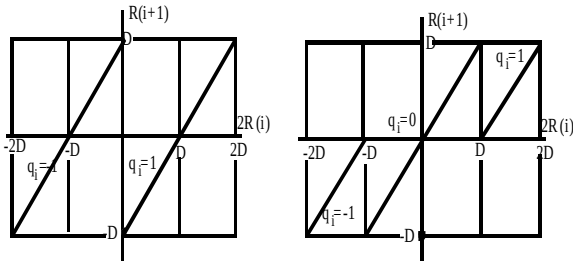

Fig. 4: División resta sin restauración, (a) con q_i\{-1,1\}, (b) con q_i\{-1,0,1\}.

3.3. SRT modificado para sumadores con ahorro de acarreo

Esta modificación mejora la velocidad de operación mediante la introducción de sumadores de acarreo. La inclusión de estos sumadores introduce una modificación en la regla de selección del dígito del cociente. Los dígitos del cociente pueden tomar el mismo conjunto de valores del algoritmo SRT (caso 4.b), es decir q={-1, 0, 1}.

$$q_i = \begin{cases} 1 & \text{si } 0 \leq R_t < 1/2 \\ 0 & \text{si } R_t = -1/2 \\ -1 & \text{si } -5/2 \leq R_t \leq -1/2 \end{cases} \quad (2)$$

La redundancia del valor del cociente en el algoritmo SRT, se puede utilizar para evitar el cálculo con precisión del resto parcial en cada iteración. Debido a la introducción del sumador con ahorro de acarreo, el resto parcial estará en un formato con ahorro de acareo. La forma redundante del cociente es utilizada para realizar una comparación con pocos bits del resto parcial con las constantes para seleccionar el dígito del cociente. Cuando el dígito del cociente es redundante, la comparación entre las constantes y el resto parcial no necesita tener una precisión completa. La regla de selección en forma grafica del algoritmo SRT se ilustra en la figura 5. Sin embargo esta representación no única del conciente introduce una complicación adicional, ya que el formato redundante se debe convertir a un formato convencional. La regla de selección de la ecuación (2) se aplica en la asignación del dígito del cociente cuando se utiliza un sumador con ahorro de acarreo.

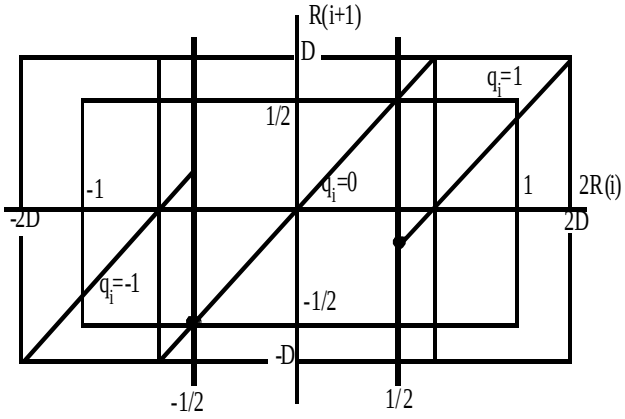

Fig. 5: División SRT

3.4. SRT modificado para base 4 con sumadores con ahorro de acarreo

En este algoritmo hay 4 posibles sustraendo: que los 4 múltiplos del divisor: 0xD,1xD, 2xD y 3xD. A diferencia de los demás, el múltiplo 3xD no se consigue con desplazamientos. El problema del cálculo de los múltiplos de divisores puede ser aliviado con el uso de números con signo. En este caso los dígitos para el cociente serán seleccionados dentro del conjunto {-2, -1, 0, 1, 2}. De esta forma, todos los múltiplos del divisor que son necesarios para llevar a cabo la operación de división se pueden determinar con desplazamientos, sin la necesidad de operaciones adicionales. La segunda consecuencia, probablemente la más importante, es la redundancia de la representación del cociente. Como se ha explicado anteriormente, la redundancia permite utilizar valores aproximados del divisor y del resto parcial durante la selección del digito del cociente.

4. CIRCUITOS REALIZADOS

Los divisores implementados en este trabajo tienen operandos de 64 bits (dividendo y el divisor) y los resultados, tanto el cociente como el resto, tienen una longitud de 64 bits.

4.1 Divisor con algoritmo resta con restauración

Como se puede apreciar en la figura 6 el circuito de división basado en el algoritmo de resta con restauración tiene un registro desplazador (R1) de 128 bits que, en primer lugar, almacena el resto parcial y, luego, los dígitos del cociente. Los 64 bits más significativos del registro desplazador se cargan al comienzo de la operación con el valor del dividiendo y luego se cargan con la salida de los sumadores. El sumador "S1" realiza la resta del valor del divisor del resto parcial en complemento a dos. El circuito también consta de otro sumador para realizar la restauración. El proceso de restauración es controlado por el bit más significativo de la salida del sumador principal.

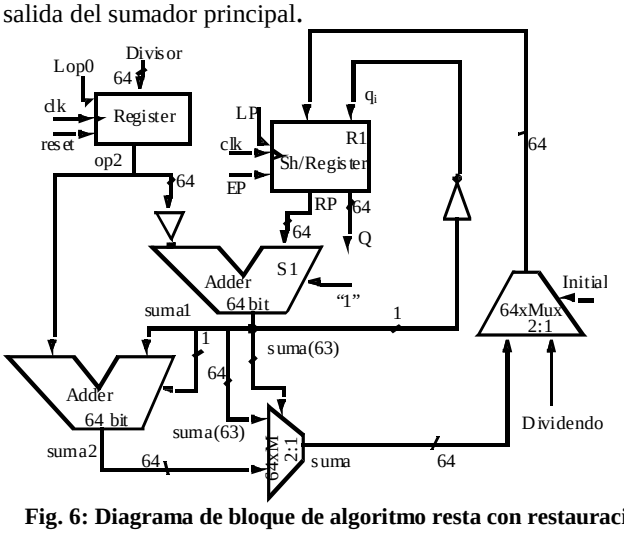

Fig. 6: Diagrama de bloque de algoritmo resta con restauración

4.2 Divisor nonpeforming

El circuito implementado para el algoritmo nonperforming tiene estructura similar al circuito anterior. La principal diferencia no tiene el sumador de restauración. En su lugar tiene un registro de 64 bits para almacenar el valor del resto parcial del ciclo de reloj previo. Con el objetivo de decidir el siguiente valor del resto parcial, el circuito dispone de un multiplexor de 64 bits para seleccionar entre la salida del sumador o el resto parcial previo.

4.3 Divisor resta sin restauración

Este circuito consta de un multiplexor de 64 bits para llevar a acabo la selección entre el complemento o el valor del divisor. El selector del multiplexor es controlado por el bit más significativo del resto parcial desplazado, que además es utilizado como acarreo de entrada del sumador. Cuando el bit más significativo del resto parcial desplazado es un "1", el multiplexor selecciona la salida del divisor y el acarreo a la entrada del sumador es un "0" De otra forma el multiplexor selecciona el complemento y el acarreo de entrada será un "1".

4.4 Divisor con algoritmo de SRT

Según la regla de selección del cociente la comparación se lleva a cabo entre dos constantes (-½ y ½) y el resto parcial. Para ello se ha incorporado un circuito combinaciónal que permite determinar el dígito del cociente de acuerdo a la comparación. El diagrama de flujo del divisor RST se muestra en la figura 7. Cuando la señal "Inicial" se pone a "1" el valor del dividendo se carga al registro desplazador (R1) a través del multiplexor M1. Cuando la señal "cond3" se pone a "1" el resto parcial previo se carga como el resto parcial siguiente en el registro desplazador R1, y el valor "00" es asignado como el valor del cociente. De otra forma, los multiplexores "m1" y "m2" permiten que la salida sea cargada en el registro desplazador R1. Cuando la señal "Cond1" es "0", la salida del sumador s1 se carga en R1, de lo contrario se carga la salida s2. El número con signo "01" es asignado al dígito del cociente cuando la señal "Cond1" está en "1" y el "11" es asignado cuando la señal "Cond3" está en "1". Como los dígitos del cociente se obtienen en forma redundante, en cada ciclo se van conviertiendo al formato convencional.

4.5 Divisor con algoritmo modificado de RST para sumadores con ahorro de acarreo.

Como los restos parciales se producen en forma de ahorro de acarreo, en la implementación de este algoritmo ha sido necesario utilizar dos registros, uno para la acumulación de la suma parcial y otro para la acumulación del acarreo parcial. Ésta y junto con el circuito combinacional que se utiliza para la implementación de la regla de selección forman parte la principal diferencia con el circuito anterior.

4.6 Divisor con algoritmo modificado de SRT para sumadores con ahorro de acarreo y de base para números enteros.

La diferencia principal con respecto a los circuitos anteriores es que la selección del cociente se ha realizado a través de la suma de los siete dígitos más significativos de la suma parcial y del acarreo parcial [10]. Luego, el resultado de la suma se utiliza para la selección del cociente en forma de números con signo.

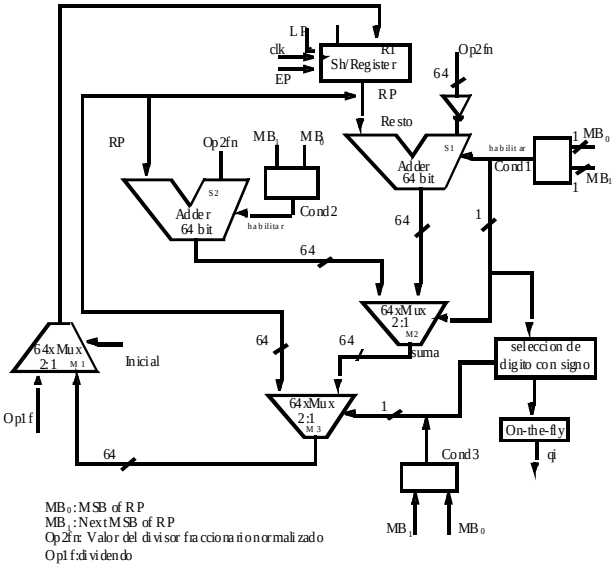

Fig. 7: Diagrama de bloque división con SRT

5. RESULTADOS

Los diseños han sido implementados en dos tecnologías CMOS, de 90 y 65 nm. La herramienta utilizada para la síntesis de los circuitos es Design Compiler (Synopsys). Queremos enfatizar que han sido utilizadas tres diferentes librerías de componentes con tres voltajes umbral: baja (LVT), estándar (SVT) y alto (HVT). Los dispositivos (HVT) se utilizan para la reducción de la corriente de fuga, y los dispositivos (LVT) se utilizan para mejorar la velocidad de operación a costa del aumento de la fuga de potencia. Los dispositivos con voltaje de umbral estándar ofrecen un equilibro entre el consumo de potencia y la velocidad de operación.

El análisis de los resultados se ha conducido de dos formas: desde el punto de vista de la tecnología y de la estructura de los divisores.

La tecnología de 90 nm ofrece mejor tiempo de retraso que la de 65 nm, porque las librerías están optimizadas para altas prestaciones, sin embargo la tecnología de 65 nm está optimizada para el consumo de potencia. Los siete divisores han sido descritos con VHDL. La síntesis y la caracterización de potencia, área y retraso se ha llevado a cabo a través de las herramientas de Synopsys y Cadence.

Los resultados del área, el retraso y el consumo de potencia en la tabla 1. En la tabla 1 la potencia consumida se da en tres columnas. Las columnas que se denominan como interna y red forman parte la potencia dinámica. La potencia interna: es aquella potencia que se disipa dentro del entorno de una celda. La disipación de potencia interna también incluye la perdida de potencia debido al cortocircuito momentáneo de las puertas de los transistores N y P. La de red se refiere a la potencia de conmutación que se origina debido a los estímulos de la entrada. En este trabajo la actividad de conmutación se ha realizado con una frecuencia de reloj de 50 Mhz.

Los divisores realizados en la tecnología de LVT necesitan más espacio, independientemente de la tecnología utilizada. El consumo de potencia debido a la fuga de potencia se ha visto incrementado cuando los circuitos se implementan con librerías LVT.

De forma general, se ha notado un incremento medio de 700% en la fuga de potencia de los diseños que utilizan librerías LVT con respecto a los que utilizan HVT. Hay un equilibro entre el consumo potencia y la velocidad de operación cuando el circuito se implementa con dispositivos de STV. También se ha visto que el tiempo de retraso se reduce en los circuitos implementados con LVT. La ganancia media de velocidad es cerca de 33% cuando la implementación se lleva a cabo con LVT comparado con HVT. La fuga de potencia de los divisores implementados en la tecnología 90 nm es mayor con que los implementados con 65 nm.

De acuerdo a la estructura de los divisores se observa que el circuito del divisor básico ocupa mejor área en el chip, un tiempo de retraso medio y consumo de potencia menor, especialmente en la implementación de 65 nm.

Dentro de los divisores de altas prestaciones, el circuito SRT es el de menor área, y un tiempo de retraso similar tiempo al del divisor SRT modificado con sumador con ahorro de acarreo. El divisor SRT de base 4 con sumador con ahorro de acarreo es el que ocupa mayor área. Hay una reducción de número de ciclos en este divisor, sin embargo la latencia sigue siendo alta.

El divisor resta sin restauración ocupa menor área y el divisor SRT ofrece menor tiempo de retraso en ambas tecnologías. El divisor SRT realizado con sumador rizado y el SRT con sumador con ahorro de acarreo tienen semejante tiempo de retraso y consumo de potencia. Sin embargo SRT con ahorro de acarreo tiene mayor área con respecto al SRT.

6. CONCLUSIONES

En este trabajo se han explorado diferentes implementaciones de divisores en dos tecnologías CMOS nanométricas. Se ha hecho el estudio de las prestaciones de 6 estructuras de divisores, realizando la descripción en VHDL.

Una de las principales conclusiones es que la reducción del tamaño de los componentes debido al avance de la tecnología no siempre implica una reducción en el tiempo de retraso. En las tecnologías utilizadas, las librerías de 90 nm están optimizadas para rendimiento, mientras que las de la tecnología de 65 nm lo están para bajo consumo de potencia. Esto hace que los diseños realizados en 90 nm sean más rápidos, mientras que los realizados en 60 nm tengan el más bajo consumo de potencia.

Con respecto a los algoritmos de división, los resultados obtenidos permiten apreciar que la complejidad adicional que introducen en el circuito los divisores de mejores algoritmos, no siempre garantiza una mejora de las prestaciones. De acuerdo con los resultados, la implementación con menor área es el divisor resta sin restauración, que también tiene el menor consumo de potencia. La implementación del divisor SRT tiene el mejor tiempo de retraso.

7. BIBLIOGRAFÍA

1. S. F. Oberman and M.J. Flynn, "Design Issues in Division de Other Floating-Point Operation ", *IEEE Trans. Computers*, vol. 46, no. 2, pp. 154-161, feb. 1997.
2. S. F. Oberman and M.J. Flynn, "Division Algorithms and implementations", *IEEE Trans. Computers*, vol. 48, no. 8, pp. 833-854, feb. 1997.
3. Israel Coren, "*Computer arithmetic algorithms* " , ED AK Peters Ltd. 2nd edition, 2001.
4. J.E Robertson, "A new Class of digital Division methods ", *IRE Trans. Electronic Computers*, Quarterly J. Mech. Appl. Math., vol. 11, pt. 3, pp 364-284, 1958.
5. G. Sutter, J-P. Deschamps, G. Bioul and E. Boemo, "Power Aware Dividers in FPGA", *PATMOS 2004*, LNCS 3254, pp. 574-584, 2004.
6. G. Sutter, G. Bioul J-P. Deschamps, and "Comparative study of SRT-dividers in FPGA", *FPL 2004*, LNCS 3203, pp. 209-220, 2004.
7. T.N. Pham, E.E. Swartzlander, Jr., "*Design of Radix-4 SRT Dividers in 65 Nanometer CMOS Technology*", Proc. Of Application-specific Systems, Architectures and Processors (ASAP 2006)
8. T.N. Pham, E.E. Swartzlander, Jr., "Design of Radix 4 SRT Dividers for Single Precision DSP in Deep Submicron CMOS Technology", *IEEE Int. Symposium on Signal Processing and Information Technology*, 2006, pp. 236-241.
9. M.D. Ercegovac and T. Lang, "*Division and Square Root*", Kluwer Academic Publishers, 1994.

AUTORES

1. Gashaw Sassaw Teshome, Ingeniero en Control Automatico, ISPJAE, Habana, Cuba, 1989, Estudiante de Doctorado en la Universidad de Sevilla, Sevilla, España desde 2005. Interés de investigación: Circuitos Aritméticos y Criptografía. sassaw@imse.cnm.es

2. Carlos J. Jiménez, Doctor en físicas por la Universidad de Sevilla es en la actualidad profesor de dicha universidad e investigador adscrito al Instituto de Microelectrónica de Sevilla. Sus tareas de investigación se centran el diseño de circuitos digitales de altas prestaciones, con aplicaciones en circuitos aritméticos como en circuitos criptográficos. cjesus@imse.cnm.es

3. Manuel Valencia, Licenciado y Doctor (1986) en Ciencias Físicas, Catedrático de Universidad (2000) en la ETS Ingeniería Informática de la Universidad de Sevilla, manolov@us.es, también pertenece al Instituto de Microelectrónica de Sevilla (CNM-CSIC). Su actividad investigadora se inserta en el Código UNESCO 3307-03 (Diseño de circuitos) y -93 (Microelectrónica. Diseño), concretamente en el Diseño digital VLSI y con FPGAs, temporización, bajo consumo y bajo ruido, circuitos aritméticos y criptográficos, arquitecturas de procesadores de propósito específico, y modelado y simulación lógico-temporal.

4. Jose M. Mora, Ingeniero de Telecomunicaciones por la Universidad Politecnica de Madrid, en la actualidad Titulado Superior Especializado del CSIC. Su trabajo se centra en tareas de diseño y test de circuitos integrados y FPGAs en el Instituto de Microelectrónica de Sevilla perteneciente al Centro Nacional de Microelectrónica. jmiguel@imse-cnm.es

Tabla I: Resultado del área y retraso

Nombre del Diseño	Tecnología.	Voltaje de umbral	Retraso (ns)	Área (um^2)	Consumo de potencia(uw)		
					Interna	fuga	red
Resto con restauración	90 nm	HVT	4.71	18320	3.24	6.1e-3	2.59
		LVT	2.63	19724	4.08	0.55	2.79
		SVT	3.45	18528	3.66	0.06	2.76
	65 nm	HVT	7.72	11597	2.8	9.9e-7	1.89
		LVT	3.13	12291	2.8	1.7e-4	1.96
		SVT	4.23	12121	2.83	1.3e-3	1.93
Non-performing	90 nm	HVT	0.74	19574	3.4	8.4e-3	2.74
		LVT	0.48	19787	4.48	0.62	2.96
		SVT	0.57	18851	4.01	7.4e-2	2.88
	65 nm	HVT	4.23	12121	2.81	8.7e-7	1.90
		LVT	0.60	10234	2.81	1.2e-4	1.98
		SVT	0.81	10455	2.84	0.9e-3	1.97
Resto sin restauración	90 nm	HVT	0.53	17058	4.29	6.1e-3	3.02
		LVT	0.32	17462	4.7	0.49	2.9
		SVT	0.37	16833	7.51	5.5e-2	4.23
	65 nm	HVT	0.86	9117	2.9	7.7e-5	1.9
		LVT	0.38	8960	2.86	9.9e-3	1.97
		SVT	0.48	8633	2.74	0.7e-3	1.9
RST	90 nm	HVT	0.43	45126	5.51	1.3e-2	3.6
		LVT	0.28	44605	8.48	0.9e-3	4.3
		SVT	0.34	43796	8	0.11	4.44
	65 nm	HVT	0.83	25987	3.23	0.2e-3	1.89
		LVT	0.37	25948	2.1	2.4e-2	1.4
		SVT	0.46	25305	2.12	1.8e-3	1.4
RST con CSA	90 nm	HVT	0.44	52947	5.58	1.6e-3	3.61
		LVT	0.30	53240	8.18	1.3	4.27
		SVT	0.36	52776	7.64	0.15	4.29
	65 nm	HVT	0.84	30089	2.44	0.2e-3	1.65
		LVT	0.37	31517	3.15	3.7e-2	1.99
		SVT	0.49	29610	3.16	2.1e-3	1.97
SRT de base 4	90 nm	HVT	8.16	82662	5.83	3.4e-2	0.96
		LVT	5.04	85700	6.53	2.83	0.96
		SVT	5.88	86172	6.16	0.4	0.95
	65 nm	HVT	15	46932	3.25	3.6e-4	1.47
		LVT	6.03	46047	3.07	4.8e-2	1.49
		SVT	8.47	45118	2.25	3.5e-3	0.94