CARACTERIZACIÓN DE SUMADORES EN TECNOLOGÍAS FUERTEMENTE SUBMICRÓNICAS[1]

Adrián Estrada, Carlos J. Jiménez, Manuel Valencia

Instituto de Microelectrónica de Sevilla – Dpto. de Tecnología Electrónica (Universidad de Sevilla)

estrada@imse.cnm.es, cjesus@imse.cnm.es, manolov@dte.us.es

ABSTRACT

En este trabajo se presentan los resultados de la caracterización de las implementaciones integradas con tecnología fuertemente submicrónica de una amplia gama de sumadores binarios. El objetivo básico es comparar sumadores paralelos clásicos con otros prefijos, determinando cuáles proporcionan mejores características al implementarlos en tecnologías de integración submicrónica. La caracterización se ha realizado siguiendo una metodología en la que hemos desarrollado los diseños de los sumadores buscando la independencia tecnológica y en la que se efectúan las medidas usando herramientas CAD comerciales. Los resultados, que se han obtenido para la tecnología de UMC 0'13 μm, muestran que las mejores prestaciones corresponden a sumadores paralelos prefijo. aunque no hay una estructura que pueda calificarse como globalmente la mejor.

1. INTRODUCCIÓN

Los circuitos sumadores paralelo de magnitudes binarias son, probablemente, los tipos de circuito más ampliamente utilizados en los sistemas digitales [1]. Están presentes como subsistema esencial en procesadores de propósito general, tanto en sus unidades aritmético-lógicas de punto fijo como en las de punto flotante, y en la mayoría de sistemas digitales de propósito específico. Salvo en algunas soluciones muy particulares para las operaciones complejas (multiplicación, división, etc.), son los sumadores paralelo, más que la implementación específica de los sumadores completos de 1 bit (*Full Adder*, FA), los que determinan la bondad del diseño en los circuitos que implementan dichas operaciones. De aquí que en este trabajo nos hayamos centrado en dichos sumadores paralelos.

En este trabajo estamos interesados en las implementaciones microelectrónicas. La evolución de las tecnologías de integración ha favorecido el diseño y la implementación de circuitos cada vez más pequeños y complejos. Cada nueva tecnología de fabricación reduce las escalas de integración precedentes al mismo tiempo que se aumenta la densidad de integración. Esta evolución tiene múltiples consecuencias entre las que nos interesa destacar:

1.- El criterio de diseño basado en reducir el área ya no es tan determinante, sino que ahora es muy importante el aumento de la velocidad de operación, parámetro que manifiesta un incesante y creciente interés. Ahora, pues, deben tenerse en cuenta estructuras de alto coste en puertas que antes fueron prohibitivas, aunque, por otra parte, siga teniendo vigencia el criterio de diseño de reducir el coste.

2.- El aumento de la densidad de integración y de la velocidad de operación han compensado el efecto de disminución del consumo de potencia debido a la disminución del tamaño y de la bajada del valor de alimentación. El efecto global es que el consumo de potencia es un parámetro que ha pasado de tenerse en cuenta en sólo algunos sistemas (como los portátiles o los bioelectrónicos) a, hoy en día, ser esencial para cualquier tipo de circuito, con el fin de evitar el exceso de temperatura, que puede llegar a provocar su mal funcionamiento durante la operación normal.

3.- Conforme decrecen las dimensiones mínimas de las tecnologías, aparecen más efectos que en las tecnologías previas podían ser ignorados. Así, en las tecnologías fuertemente submicrónicas, hay que plantear la revisión del comportamiento de los circuitos digitales con estas nuevas tecnologías, por lo que es necesario una caracterización de tanto de estructuras antiguas como nuevas, sobre todo en circuitos tan importantes como los sumadores.

1. Este trabajo ha sido parcialmente financiado por el proyecto DOC (TEC 2004-01509/MIC).

4/La tendencia incesante de cambios tecnológicos, junto a la cada vez mayor abstracción de los diseños, debe modificar los procedimientos de diseño hacia una menor dependencia tecnológica (en el sentido de que el resultado de diseño pueda ser aplicado fácilmente a varias tecnologías diferentes), así como al cada vez mayor apoyo en las herramientas CAD comerciales.

En este trabajo se presentan nuestros resultados de caracterización de implementaciones integradas con tecnología fuertemente submicrónica para una amplia gama de sumadores binarios. Dada la existencia de una gran cantidad de estructuras de sumadores en este trabajo, necesariamente, hemos hecho una selección. El objetivo básico es comparar sumadores paralelos clásicos con otros prefijos, determinando cuáles proporcionan mejores características al implementarlos en tecnologías de integración submicrónica. La caracterización se ha realizado siguiendo una metodología en la que hemos desarrollado los diseños de los sumadores buscando la independencia tecnológica y en la que se efectúan las medidas usando herramientas CAD comerciales, en particular, la medición de parámetros de área, retraso y consumo de potencia, utilizando siempre herramientas comerciales de diseño *semi-custom*.

Este artículo se estructura de la siguiente forma: en el siguiente apartado se presentan las estructuras de sumadores que se han caracterizado. A continuación se presenta la metodología de caracterización que se ha seguido y las pruebas y resultados de la caracterización realizada. Finalmente se extraen las conclusiones.

2. ESTRUCTURAS DE SUMADORES

Para la realización de la caracterización en tecnologías submicrónicas, se han escogido ocho estructuras de sumadores, cuatro de ellas más clásicas y otras cuatro que utilizan estructuras más novedosas.

A continuación se describen brevemente cada una de las estructuras que se van a caracterizar.

2.1. Sumadores clásicos

2.1.1. Sumador rizado (Carry-ripple adder)

Conceptualmente, este es uno de los sumadores más simples. Se obtiene encadenando una secuencia de *n* sumadores completos (FA, *Full Adders*) mediante las salidas del acarreo de la etapa *i* a la entrada de acarreo de la etapa *i+1*. El problema que plantea este tipo de estructura reside en la propagación del acarreo. En este tipo de realizaciones el acarreo tiene que propagarse en serie desde la posición *0* hasta la posición *i* de manera que el sumador completo correspondiente a esa posición pueda generar el resultado y el acarreo de salida correcto. Hay que esperar que el acarreo se propague a través de todos los sumadores completos con el fin de obtener el resultado correcto de la suma.

Una forma de mejorar estos retrasos consiste en acelerar la computación de la red de acarreos. Existen varias modificaciones sobre esta estructura que mejoran este aspecto del sumador, como por ejemplo las estructuras *carry-skip adder*, *carry-completion adder* y *Manchester adder* [2], [4], [5].

2.1.2. Sumador Manchester (Switched carry-ripple adder)

Este sumador es una optimización del sumador rizado, consistente en incrementar la velocidad de cálculo de la red de acarreos mediante el empleo de una serie de funciones lógicas y conmutadores. Para cada pareja de bits x_i e y_i se generan una serie de señales que controlan el funcionamiento de estos conmutadores. Estas señales son: P_i, G_i y K_i, donde P_i se activa si se propaga el acarreo que proviene de la etapa anterior, G_i se activa si se genera acarreo en esta etapa i-ésima y K_i que se activa en el caso de que el acarreo sea cero. Estas señales se excluyen mutuamente de manera que sólo una de las señales está activa en cada momento. Todos los conmutadores se fijan al mismo tiempo y como resultado la propagación del acarreo sólo sufre un retraso introducido por el conmutador en cada etapa. La longitud de la red de acarreos está limitada por la tecnología utilizada para implementar los conmutadores.

2.1.3. Sumador con acarreo adelantado (Carry-lookahead adder)

El siguiente paso para acelerar la computación de la red de acarreo es el uso del "acarreo adelantado". Esta técnica consiste en calcular todos los acarreos intermedios en paralelo, evitando así tener que esperar hasta que el acarreo correcto se propague desde la etapa donde se ha generado hasta la siguiente etapa. Este cálculo puede realizarse gracias a que toda la información necesaria está contenida en los bits que componen los dos operadores. Mediante una serie de operaciones lógicas simples y utilizando las entradas del sumador es posible determinar qué etapa generará un nuevo acarreo o lo propagará hacia la siguiente.

Sin embargo, aunque esta técnica permite una mayor velocidad a la hora de calcular el acarreo, para operandos con un número de bits elevado, se necesita un gran número de puertas con un alto *fan-in* [2]. Para solucionar este problema, existen varias técnicas, como puede ser el particionado en varios bloques mas pequeños que

implementan el *acarreo adelantado* interconectados en serie a través de acarreos intermedios, como si de un *sumador rizado* se tratara (*ripple-block carry-lookahead*) [2].

Otra técnica que mejora la velocidad del sumador es calcular los acarreos intermedios mediante un segundo nivel de acarreo adelantado o generador de acarreo adelantado (*Block carry-lookahead adder*). Variantes de estas estructuras añaden más niveles intermedios y más unidades de cálculo de acarreo adelantado, como por ejemplo, el sumador de acarreo adelantado con super bloques (*Superblock carry-lookahead adder*), la versión mejorada de éste (*Improved superblock carry-lookahead carry*) o el sumador con acarreo adelantado con superbloques en serie (*Superblock-ripple carry-lookahead*) [2], [3].

2.1.4. Sumador con selección de acarreo (Carry-select adder)

Este tipo de sumadores está compuesto por tres sumadores de longitud *k*/2 que pueden ser de cualquier tipo, es decir, acarreo anticipado, rizado, Manchester, etc., unidos para formar un sumador de longitud *k*. Uno de los sumadores es usado para calcular la parte baja de los *k*/2 bits de suma. Los otros dos sumadores son utilizados para calcular los *k*/2 bits de la parte alta de la suma, con la característica de que uno de estos sumadores tiene como acarreo inicial C_0 fijado a uno y el otro sumador lo tiene fijado a cero. El resultado correcto de la parte alta de la suma es seleccionado mediante el acarreo de salida del sumador que calcula la parte baja de la suma. Esta selección se realiza a través de un multiplexor controlado, como se ha mencionado antes por el acarreo de salida del sumador encargado de computar la parte baja de la suma.

Con esta estructura, aunque se obtiene una mejora en los tiempos de cálculo, ésta es a costa de duplicar la circuitería necesaria para calcular una parte de la suma.

2.2. Sumadores paralelo prefijo (Parallel prefix adders)

Uno de los problemas que tienen los sumadores con acarreo adelantado es que al aumentar el número de bits considerablemente aumenta de forma considerable la complejidad de las ecuaciones para calcular el acarreo y el fan-in de las puertas implicadas en estas ecuaciones. Estos sumadores se basan en las mismas ecuaciones que usan los sumadores de acarreo adelantado, pero empleándolas de una forma más genérica, reduciendo esta complejidad y en consecuencia el fan-in de las puertas basándose en el calculo de los acarreos usando técnicas de cálculo paralelo prefijo [6]. Están orientados a aplicaciones de alto rendimiento [7],[8],[9].

Según las definiciones de P_i, G_i y K_i, dadas anteriormente, el cálculo del acarreco es:

$$c_{i+1} = G_i + P_i c_i$$

$$(P_{out}, G_{out}) = (P_i \cdot P_{i-1}, G_i + P_i \cdot G_{i-1}) \qquad (1)$$

La computación del acarreo usando estas expresiones puede verse como un problema recursivo que puede ser resuelto de forma más eficiente empleando la computación paralela prefija. Este tipo de computación se puede emplear con cualquier problema que use un operador asociativo, como es el caso del operador fundamental de carry [6], [7]. El esquema básico de este tipo de problema es, dadas *n* entradas y un operador asociativo cualquiera, calcular *n* salidas, siendo cada una de estas salidas dependiente de las entradas anteriores.

Los sumadores basados en este tipo de estructuras paralelas prefijas se denominan sumadores paralelos prefijos y son básicamente sumadores de acarreo adelantado con un esquema diferente para el cómputo de la red de acarreo. A continuación se presentan algunas de las estructuras paralelas prefijas más significativas.

2.2.1. Sumador paralelo prefijo de R.P. Brent y H.T. Kung (Brent-Kung Adder)

La Figura 1 muestra el árbol de computación de acarreos de un sumador tipo Brent-Kung de 16-bits. Los círculos blancos que aparecen constituyen la etapa denominada de precomputación, en la cual se calculan los bits de propagación y de generación. Los círculos negros representan el operador fundamental del carry. En cada circulo negro se realiza la operación descrita por la ecuación (1). Las líneas encadenan pares de grupos de propagación y generación de una etapa con otra de una fase siguiente, para construir un grupo mayor hasta llegar a obtener el acarreo de salida i-ésimo.

Por ejemplo, las ecuaciones para el cálculo del acarreo de salida C_{12}, según se muestra en la Figura 1, es la siguiente:

$$C_{12} = (G_{11:0}, P_{11:0}) = (G_{11:8}, P_{11:8}) \bullet (G_{7:0}, P_{7:0})$$

Donde el primero de los términos se calcula, primero en base a términos intermedios y posteriormente a términos primarios:

$$(G_{11:8},P_{11:8}) = (G_{11:10},P_{11:10}) \bullet (G_{9:8},P_{9:8}) =$$
$$= (g_{11}, p_{11}) \bullet (g_{10}, p_{10}) \bullet (g_9, p_9) \bullet (g_8, p_8)$$

Como se puede observar en la Figura 1, el número de etapas para esta estructura prefija son 7. En la primera mitad hasta la etapa 4 se calculan los acarreos C_1, C_2, C_4, C_8, C_{16}. El resto de las etapas hasta la séptima se utiliza para calcular el resto de acarreos. Esta estructura, como se puede comprobar, es bastante simple ya que los nodos tienen un *fan-out* bastante bajo [10]. Sin embargo, esto repercute de forma directa en una mayor profundidad de la estructura prefija y en consecuencia en un mayor retraso.

2.2.2. Sumador paralelo prefijo de P.M Kogge y H.S. Stone (Kogge-Stone prefix adder)

Este tipo de sumador plantea una estructura similar a la anterior, pero incrementando el número de líneas en cada etapa (Figura 2). El *fan-out*, al igual que en la estructura anterior, se sigue manteniendo en la unidad. Al aumentar el número de líneas por etapa, se consigue reducir el número de éstas, obteniendo un menor retraso a la hora de realizar la operación de suma.

Esta estructura resulta muy atractiva para aplicaciones de alto rendimiento. Sin embargo, debido a la complejidad que ofrece el árbol, la implementación de este circuito tiene un alto coste en área y en consumo. Por otra parte, esta estructura genera un layout regular y un *fan-out* controlado [9],[11].

2.2.3. Sumador paralelo prefijo de R.E. Ladner y M.J Fischer(Ladner-Fischer parallel prefix adder)

Ladner y Fischer introducen una estructura que tiende a reducir el número de etapas. Sin embargo y como se puede apreciar en la Figura 3, esta reducción es a costa de un *fan-out* mucho mayor que el resto de las estructuras prefijas presentadas.

Dado el escaso número de etapas que presenta, resulta bastante atractivo para aplicaciones que necesitan un alto rendimiento. Sin embargo, esta estructura plantea un *fan-out* elevado que redunda en una mayor lentitud de las puertas y en un aumento del consumo [9], [11].

2.2.4. Sumador paralelo prefijo de T. Han y D.A Carlson (Han-Carlson parallel prefix adder)

Esta estructura es un híbrido de las estructuras prefijas presentadas por Kogge-Stone y Ladner-Fischer. De nuevo se produce un incremento del número de etapas, aunque menor que en el propuesto por Brent-Kung, para reducir el *fan-out* del circuito (Figura 4).

El esquema planteado por Han-Carlson es similar al planteado por Brent-Kung haciendo las operaciones con los acarreos sólo con los bits pares. Las señales de propagación y generación de los bits impares se propagan a través del árbol prefijo. En la última etapa se combinan con las señales de acarreo impares para generara los acarreos pares. Al igual que el esquema planteado por Brent-Kung, este reduce la complejidad a costa de añadir etapas adicionales al árbol prefijo (cinco en el caso de la Figura 4).

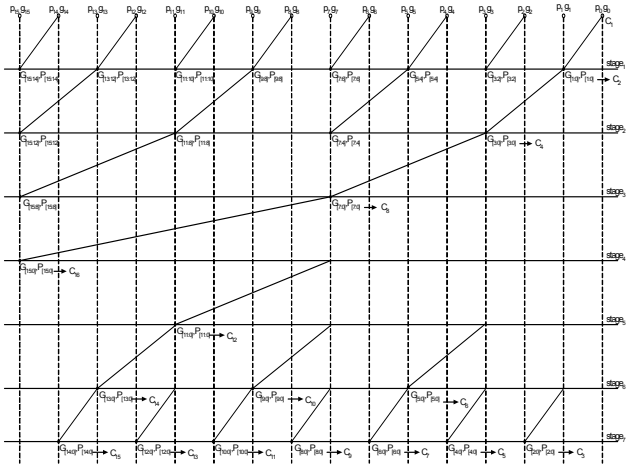

Figura 1: Sumador paralelo prefijo de 16-bits propuesto por R.P. Brent y H.T. Kung

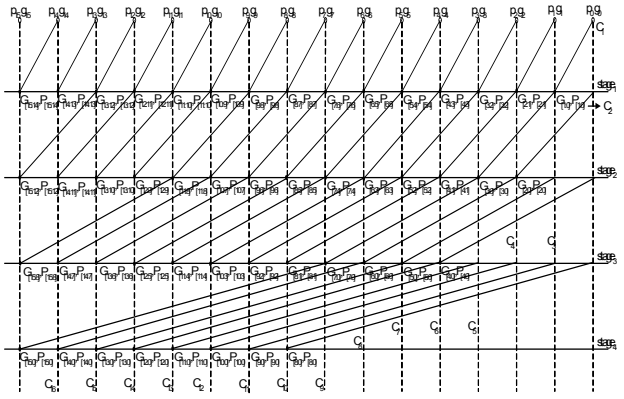

Figura 2: Sumador paralelo prefijo de 16-bits propuesto por P.M Kogge y H.S Stone.

3. METODOLOGÍA DE CARACTERIZACIÓN

El proceso de caracterización se va a centrar en la obtención el área ocupada por el diseño, el retraso máximo y el consumo de potencia. Este último parámetro, que tradicionalmente ha sido crítico únicamente en los sistemas portátiles, ahora, se ha convertido en un parámetro crítico para cualquier tipo de circuito independientemente de su grado de portabilidad.

Además, la metodología tiene las siguientes características:

- Es independiente de la tecnología en las primeras etapas del diseño.
- Utiliza flujo de diseño comerciales.

En los flujos de diseños comerciales es habitual comenzar las descripciones de los diseños a partir del nivel de transferencia de registros o *RT* mediante lenguajes de descripción de *hardware* (*HDL, Hardware Description Language)*. Esto proporciona una independencia de los diseños de la tecnología empleada, consiguiendo así la primera de las características de la metodología de caracterización. Estas descripciones, cumpliendo las restricciones de síntesis, pueden ser sintetizadas por las diferentes herramientas comerciales de síntesis existentes, con lo que se cumple la segunda de las características. Son estas herramientas las que seleccionarán las tecnologías objetivo, por lo que esta metodología es válida para todas las tecnologías de fabricación de circuitos integrados que cuenten con librerías de diseño para herramientas de síntesis.

3.1. Parámetros a caracterizar

3.1.1. Área ocupada por el circuito

El primer parámetro a obtener para la caracterización de los circuitos es el área ocupada por el diseño una vez sintetizado en una tecnología concreta. El área ocupada por un diseño basado en celdas determinada se puede expresar como la suma de las áreas de las celdas que lo componen, y del área utilizada para realizar las interconexiones entre éstas.

En tecnologías con sólo dos capas de metal las celdas se organizan en filas dejando un espacio entre ellas para poder realizar la interconexión entre las celdas. Sin embargo, en las tecnologías submicrónicas, debido a la disponibilidad de más de dos capas de metal, no es necesario dejar espacio entre las diferentes filas que componen el diseño. Dependiendo del número de metales empleados en el rutado, en algunos casos será necesario introducir ciertas celdas *vacías* por las que realizar rutado adicional.

El espacio introducido para rutado adicional depende fundamentalmente de la tecnología utilizada para implementar el diseño y en menor medida de la complejidad del diseño. En este caso no se va a considerar el área reservada para rutado adicional, ya que representa el mismo porcentaje para todos los diseños y en consecuencia no va a tener ningún efecto en la comparación.

El área ocupada por el diseño puede obtenerse en diferentes etapas del diseño, siendo la primera de ella después de la síntesis del diseño en HDL como suma del área ocupada por todas las celdas. Esto nos va a evitar tener que realizar el *layout* completamente rutado para obtener una estimación del área. La suma del área de todas las

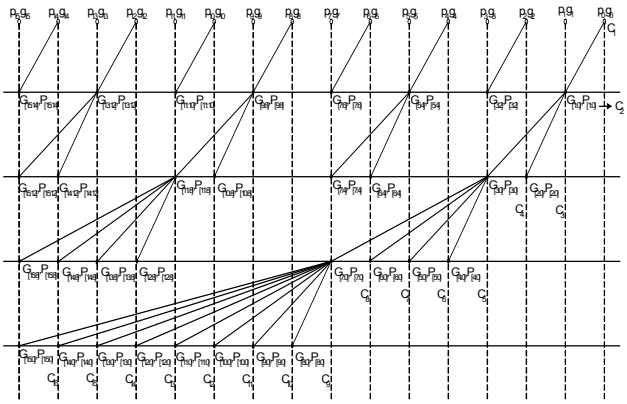

Figura 3: Sumador paralelo prefijo de 16-bits de R.E Ladner y M.J. Fischer.

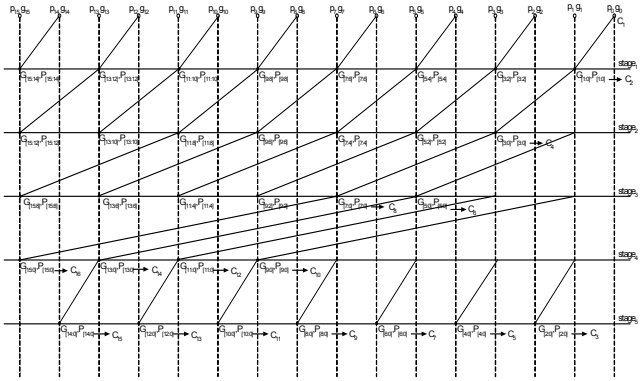

Figura 4: Sumador paralelo prefijo de 16-bits propuesto por T. Han y D.A. Carlson.

celdas puede obtenerse con la propia herramienta de síntesis.

3.1.2. Retraso máximo

El valor del retraso máximo puede obtenerse usando una herramienta de análisis estático o mediante el uso de simuladores lógicos temporales. Los simuladores son herramientas que ofrecen una gran precisión en la medición de retrasos. Sin embargo, la obtención del retraso máximo, necesita del conocimiento a *priori* del patrón de entradas concreto que activan el camino crítico. Dado que a *priori* no se conoce este patrón es necesario la realización de simulaciones exhaustivas (con todos los patrones), la medición de todos los retrasos para cada patrón y la obtención del mayor de ellos. Este análisis resulta bastante costoso en tiempo y en esfuerzo.

Por otra parte, las herramientas de análisis estático están específicamente diseñadas para realizar esta tarea, puesto que el cálculo del retraso lo hacen para cada posible ruta del diseño. Este tipo de análisis puede tener el problema de contabilizar rutas que no se lleguen a activar nunca. Sin embargo, estas situaciones son más críticas en circuitos secuenciales donde la existencia de retroalimentación puede hacer que la medición del retraso llegue a ser bastante compleja. En los circuitos combinacionales estas herramientas proporcionan buenos resultados con un esfuerzo mínimo. Además, la mayoría de las herramientas de síntesis que existen en el mercado incorporan este tipo de analizadores que además de proporcionar información temporal sobre el diseño, también son utilizadas para realizar las optimizaciones temporales.

3.1.3. Potencia consumida

El consumo de potencia, además de ser uno de los parámetros que más importancia ha ido adquiriendo conforme se reducía la escala de integración, es también uno de los más difíciles de medir con precisión.

La potencia consumida se puede dividir en potencia estática y potencia dinámica. La potencia estática suele ser calculada mediante la suma del consumo estático de cada una de las celdas que forman el diseño. Estos datos están incluidos en la librería de celdas. La potencia dinámica por su parte contribuye de mayor manera a la potencia global consumida. Su valor depende del número de transiciones en todos los nodos del circuito. Por ello un cálculo preciso de esta potencia pasa necesariamente por un cálculo preciso de las transiciones en todos los nodos del circuito, para lo que es necesario tener unos modelos de simulación bastante precisos de las celdas [11].

La actividad de conmutación de los nodos internos de circuitos realizados utilizando metodologías de diseño *semi-custom*, tiene que ser calculada utilizando simuladores lógicos, puesto que en general no se dispone de descripciones a nivel de transistores de las celdas de librería.

3.2. Procedimiento de caracterización empleado

La Figura 5 muestra el procedimiento de caracterización empleado en este trabajo. El flujo sigue los siguientes pasos:

El punto de partida es la realización de una descripción del diseño a nivel de transferencia de registros.

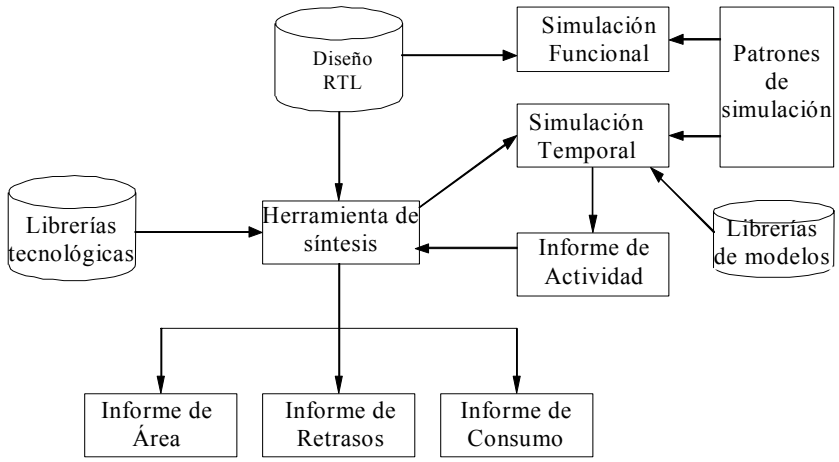

Figura 5: Diagrama de flujo del procedimiento de caracterización seguido.

Para realizar esta descripción se ha usado el lenguaje de descripción de *hardware* VHDL. Estas descripciones son independientes de la tecnología.

Antes de realizar la síntesis del diseño con una tecnología determinada, se ha verificado que las descripciones VHDL tienen un comportamiento funcional correcto, mediante la realización de simulaciones funcionales.

Una vez verificado el funcionamiento correcto de las descripciones VHDL, se procedió a su síntesis mediante una herramienta de síntesis. La herramienta de síntesis instancia el diseño en una tecnología concreta mediante el uso de celdas estándar. Una vez sintetizados los diseños se procede a la caracterización de los mismos.

La caracterización del área ocupada y del retraso máximo de cada una de las celdas puede hacerse en base a los informes generados por la propia herramienta de síntesis.

Para conocer con precisión el consumo dinámico es necesario medir previamente la actividad de conmutación en los nodos internos, cosa que se hace empleando un simulador lógico. Para ello el primer paso consiste en exportar el diseño sintetizado a un formato susceptible de ser importado por el simulador. Para la simulación se emplea un conjunto de patrones que no pretenden realizar una comprobación funcional del circuito, sino obtener una estadistica o promedio, de manera que se usa un conjunto de patrones generados de forma aleatoria. Con esto, el simulador debe generar un fichero con la actividad de conmutación de cada nodo del circuito. Con este fichero, la herramienta de síntesis es capaz de generar un informe del consumo que tiene el circuito.

4. PRUEBAS REALIZADAS

El procedimiento de caracterización se ha aplicado a los ocho sumadores descritos anteriormente. Para ello, en primer lugar, se ha realizado una descripción VHDL de cada una de las estructuras de sumadores y se han verificado funcionalmente. Estas pruebas se han realizado para estructuras con 16, 32 y 64 bits.

Las herramientas empleadas para llevar a cabo las pruebas de la metodología han sido *Synopsys* y *Modelsim*. De *Synopsys* se ha empleado la aplicación *Design Analyzer* para sintetizar las descripciones VHDL en la tecnología UMC 0.13 µm. *Modelsim* ha sido utilizado para verificar la funcionalidad de las descripciones VHDL y generar los índices de actividad de los nodos de cada una de las estructuras seleccionadas.

Cada uno de estos sumadores ha sido optimizado en área y consumo, fijando las restricciones de diseño para área, consumo dinámico y consumo por pérdidas, a cero con el fin de indicar a la herramienta que intente minimizar estos valores lo máximo posible. Los valores de área y retraso máximo para estos circuitos, se han obtenido utilizando la herramienta *Design Analyzer Report*.

Al ser el consumo de potencia dependiente de las transiciones que se producen en todos los nodos del circuito y tener que realizar una simulación para obtener estos datos, ha sido necesario generar un conjunto de patrones de simulación. Para las pruebas realizadas se ha generado un conjunto aleatorio de 10^4 patrones. La frecuencia de operación ha sido fijada a 10 MHz.

4.1. Resultados obtenidos

Las pruebas han sido realizadas para la tecnología UMC 0.13 µm. La Tabla 1, muestra los resultados obtenidos para los parámetros de área, retraso, potencia y consumo.

La Figura 6, muestra el consumo de área para cada uno de los sumadores empleados. Como se puede observar, los sumadores paralelos prefijos son los que consumen mayor área, notandose aún más esta diferencia a medida que aumenta el número de bits que conforman los operandos de entrada del sumador. Esto es así, en este tipo de sumadores debido a la complejidad que presentan los árboles de computación que los forman.

En la Figura 7 se ilustra la evolución del consumo de área respecto al aumento de la longitud de los operandos, mostrandose un crecimiento más lineal en el caso de los sumadores clasicos, a excepción del sumador de *Selección de acarreo* (*carry select*), que tiene un compartimento más parecido a los sumadores paralelos prefijos. Esto es debido al elevado número de recursos que emplea para realizar la operación de suma, consecuencia de su naturaleza hibrida.

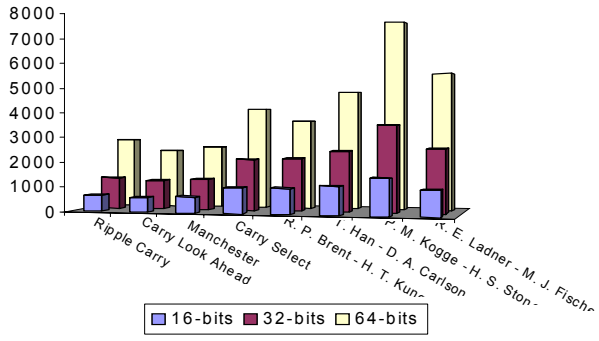

Figura 6: Consumo de área en función del número de bits.

	Área (μm^2)			Retraso (ns)			Potencia (μW)			Potencia*Retraso (fJ)*		
	16-bits	32-bits	64-bits	16-bits	32-bits	64-bits	16-bits	32-bits	64-bits	16-bits	32-bits	64-bits
Ripple Carry	665,28	1278	2823,5	4,89	7,67	13	14,19	20,91	49,92	69,4	160,4	649,6
Carry Look Ahead	573,69	1175	2358,7	2,95	5,87	17	7,3	11,54	30,18	21,7	67,8	513,5
Manchester	642,81	1257,9	2528	3,18	7,35	22,5	8,3	17,64	43,68	26,7	129,7	986,9
Carry Select	1047,1	2089,1	4143,7	2,11	3,93	11,6	13,1	28,57	73,76	27,7	112,3	858,6
Brent-Kung	1035	2151,3	3620,1	1,63	2,15	3,17	8,81	18,66	45,90	14,4	40,1	145,5
Han-Carlson	1149,1	2446,8	4843,5	1,36	1,77	3,29	9,33	19,33	44,25	12,7	34,2	145,6
Kogge-Stone	1468,8	3516,4	7756,9	1,42	1,63	2,21	10,8	25,32	54,91	15,4	41,3	121,4
Ladner-Fischer	1057,5	2583,3	5612,5	0,91	1,95	3,02	9,92	21,9	53,41	9,0	42,7	161,3

Tabla 1: Área, retraso y potencia para sumadores de 16, 32 y 64-bits. Tecnología UMC 0.13µm.

El sumador que experimenta un mayor crecimiento de área al aumentar el número de bits es el propuesto por P. M. Kogge y H. S. Stone, debido a que tiene el árbol de computación más complejo de los de su clase. De los sumadores evaluados, el que presenta un crecimiento prácticamente lineal es el sumador paralelo prefijo propuesto por R. P. Brent y H. T. Kung, lo que lo convierte en un candidato más que aceptable para aplicaciones que tengan limitaciones de área.

En relación con el retraso, como se puede observar en la Figura 8, los sumadores paralelos prefijos tienen un comportamiento prácticamente lineal, siendo ideales para aplicaciones que precisen de velocidad. Los sumadores clásicos se ven afectados en mayor medida por los cambios en la longitud de los operandos, siendo el sumador Manchester el que peor se comporta. El sumador con *acarreo adelantado* (*carry look ahead*), tiene un comportamiento similar al sumador Manchester, debido al crecimiento de la complejidad de sus expresiones para el cálculo del acarreo. Dentro del conjunto de sumadores clásicos, los sumadores de Selección de acarreo (*carry select*) y acarreo adelantado (*carry look ahead*) son los que mejores prestaciones de velocidad ofrecen para el caso de 16-bits, así como para 32-bits. Sin embargo, para el caso de 64-bits, los sumadores de *acarreo adelantado* y los *sumadores Manchester* son los que ofrecen un peor rendimiento, siendo mejores los *sumadores rizados* (*ripple carry*) y los sumadores basados en la *selección de acarreo* (*carry select*). Respecto al conjunto de sumadores paralelos prefijos, todos ofrecen un comportamiento lineal, de manera que, en general, mantienen prácticamente las mismas diferencias de velocidad al variar la longitud de sus operandos. El más rápido de todos ellos es el propuesto por *R.E. Ladner y M.J. Fischer* para el caso de 16-bits, mientras que el propuesto por *P.M Kogge y H.S Stone* es el más rápido para el caso de 32 y 64-bits (Figura 9).

En relación al consumo, para el caso de 16, 32 y 64-bits el sumador que menos consumo presenta es el basado en la estructura de *acarreo adelantado*, mientras que el que más consume es el sumador basado en la estructura con

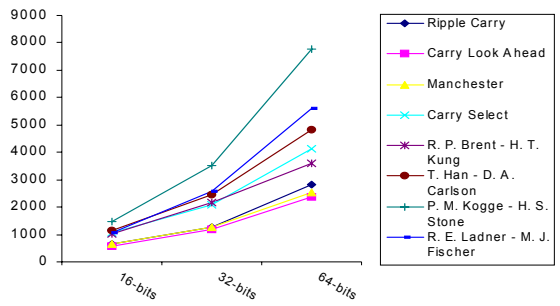

Figura 7: Evolución del consumo de área respecto al tamaño de los operandos

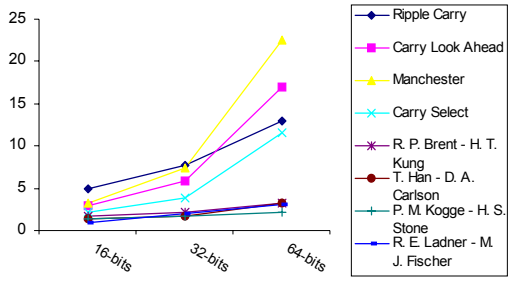

Figura 8: Evolución del retraso respecto al tamaño de los operandos

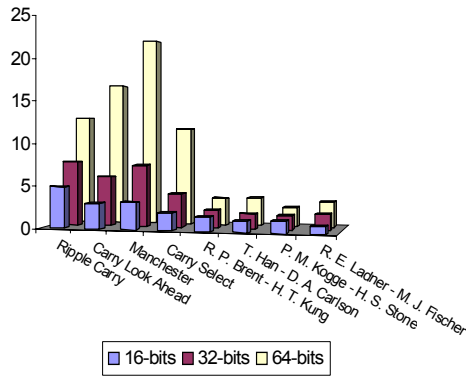

Figura 9: Retraso en función de la longitud de los operandos

selección de acarreo, debido al elevado número de recursos empleados para implementar esta estructura. Los sumadores paralelos prefijos, ofrecen una variación del consumo similar para el mismo número de bits. En término medio los sumadores paralelos prefijos ofrecen una buena opción para aplicaciones que requieran un bajo consumo de potencia (Figura 10).

Como se puede observar en la Figura 11, los sumadores paralelos prefijos presentan una relación prácticamente lineal frente al producto potencia retraso, de manera que son la opción a considerar en aplicaciones que requieran un cierto equilibrio entre la potencia consumida y la tiempo de respuesta. Los sumadores clásicos, sin embargo, ofrecen en general una peor relación entre ambos parámetros, de manera pueden ser apropiados para aplicaciones que requieran poco consumo o poco retraso, pero no en aquellas que requieran un compromiso entre ambos parametros.

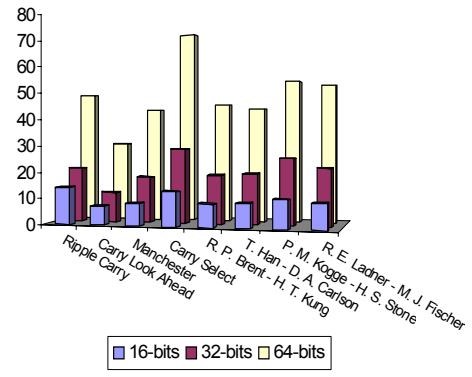

Figura 10: Consumo en función de la longitud de los operandos.

Se puede concluir que los sumadores paralelos prefijos, junto con las estructuras de acarreo adelantado, son la mejor opción para aplicaciones que requieran velocidad y bajo consumo. Sin embargo, las estructuras clásicas, son ideales para aplicaciones que tengan restricciones de área.

5. CONCLUSIONES

En este articulo se ha presentado una caracterización de diversas estructuras de sumadores en tecnologías fuertemente submicrónicas. La caracterización de dichos circuitos se ha realizado en base a los tres parámetros más importantes: área ocupada por el circuito, máximo retraso entre las entradas y las salidas y la potencia consumida.

El procedimiento de caracterización seguido tiene las siguientes características:

- Utilización de herramientas de CAD comerciales.
- Aplicable a diferentes tecnologías.
- Empleo de flujos de diseño *semi-custom*.

Con todo esto, el procedimiento parte de la descripción de los circuitos usando lenguajes de descripción de *hardware* manteniendo además las restricciones impuestas por las principales herramientas de síntesis. De esta forma se consigue mantener el diseño independiente de la tecnología en esta primera etapa del diseño.

El empleo de herramientas de síntesis comerciales transforma esta descripción en un *netlist* de celdas de una tecnología concreta. La propia herramienta de síntesis permite la obtención de los datos de área ocupada y de retraso máximo entre entradas y salidas.

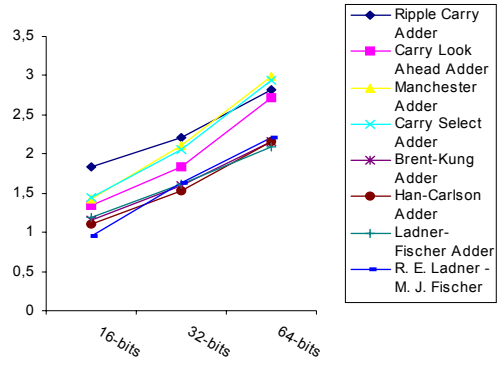

Figura 11: Evolución del la relación consumo-potencia.

Para la obtención del consumo de potencia es necesario el paso previo de calcular los índices de actividad de los nodos intermedios del circuito. Para ello el diseño sintetizado es exportado y simulado en un simulador lógico capaz de generar dicha información. Con este fichero, la herramienta de síntesis genera el dato de consumo dinámico del circuito.

La caracterización se ha llevado a cabo sobre diferentes estructuras de sumadores propuestas en la literatura, tanto clásicas como más novedosas.

Se ha realizado una descripción VHDL sintetizable de todos los sumadores escogidos, algunas de las cuales son parametrizables en cuanto al número de bits del sumador. Para aquellos que no han podido ser parametrizados se han realizado descripciones con 16 y 32 y 64 bits en los operandos. Tras comprobar su comportamiento funcional se ha procedido a su síntesis y a la obtención de los parámetros que los caracterizan.

Finalmente se han analizado los datos con objeto de obtener las recomendaciones de las estructuras analizadas en función de los requerimientos de la aplicación en la que vayan a ser utilizados. Como conclusiones generales puede extraerse que las estructuras de sumadores clásicas ofrecen, en general, un menor consumo de área pero presentan también un retraso mucho mayor. Por otro lado, aunque las estructuras más novedosas tienen un mayor consumo de potencia, los valores obtenidos para la figura de mérito retraso*potencia son mejores para éstas estructuras que para las clásicas.

6. REFERENCIAS

[1] Ch. Nagendra, M. J. Irwin, R. M. Owens, "Area-Time-Power Tradeoffs in Parallel Adders", *IEEE Trans. on Circuits and Systems - II: Analog and Digital Signal Processing*, Vol. 43, Nº 10, octubre 1996, pp. 689-701.

[2] Amos R. Omondi, *Computer Artihmetic Systems*, Prentice Hall International, 1994

[3] Milos D. Ercegovac, Thomas Lang, Ed. Morgan and Kaufmann, *Digital Aritmetic*, 2004

[4] Behrooz Parhami, *Computer Arithmetic Algorithms and Hardware Design*, Ed. Oxford University Press, 2000

[5] Haru Yamamoto, Shane Erikson, *Design and Comparison of Standard Adder Schemes*, Haru Yamamoto, Shane Erikson, Winter 2004, U.C.L.A.

[6] P. M. Kogge. *Parallel Solutions of Recurrence problems*, I.B.M. Journal Research and Development, March 1994.

[7] P. M. Kogge and H. S. Stone, "A *parallel Algorithm for the efficient solution of general class of recurrence equations*", IEEE Transaction Computers, vol V-22, Nº8, 1973, pp 786-793.

[8] H. Ling, **"***High speed binary adders*", I. B. M. Journal Research and Develoment, vol 25, p.155-66, 1981.

[9] T. D. Han, D. A. Carlson. "*Fast area-efficient VLSI adder*", 8th Symposium on computer arithmetic, May 1987.

[10] R. P. Brent and H. T. Kung, "*A regular layout for parallel adders*", IEEE Transaction on Computer, C-31, Marzo 1982, pp 260-264.

[11] C. Baena, J. Juan Chico, M. J. Bellido, P. Ruiz de Clavijo, C. J. Jimenez and M. Valencia, "*Measurement of Switching Activity of CMOS Digital Circuits at Gate Level*", PATMOS 2002, pp. 353-362.