Avaliação de Circuitos Aritméticos em Tecnologias Nanométricas

Albano Borba, Douglas Borges, Cristina Meinhardt, Vagner Rosa

***Resumo*—Muitas técnicas utilizadas para realizar operações aritméticas estão presentes na literatura, visto que a computação é um pilar importante em sistemas computacionais. Dentre as funções aritméticas, as principais em sistemas computacionais são a soma e multiplicação. Verifica-se que o uso de diferentes arquiteturas na implementação destes sistemas podem variar o seu desempenho em função de propriedades como a tecnologia utilizada nos transistores e número de bits utilizados. Este trabalho apresenta um estudo comparativo sobre alguns modelos de somadores e multiplicadores de 4 bits, analisando características como tempo de atraso e consumo de energia. São comparadas quatro arquiteturas de somadores: *Ripple-Carry*, *Carry-Lookahead*, *Carry-Select* e o Somador *Kogge-Stone Tree* e três arquiteturas de multiplicadores: duas implementações do algoritmo Baugh-Wooley (BW1 e BW2) e uma do algoritmo de Booth. O somador *Kogge-Stone Tree* apresentou os melhores resultados de atrasos máximos, sendo 40% mais rápido que o *Ripple-Carry*. Entretanto, esta arquitetura apresentou mais que o dobro de consumo de energia médio. Considerando potência e desempenho, a arquitetura *Carry-Select* apresenta a melhor relação entre custo e benefício. Para todos os parâmetros avaliados nos multiplicadores, os resultados mostram que uma boa implementação do algoritmo BW é a melhor alternativa.**

***Index Terms*—Nanotecnologia, Circuitos Aritméticos, Somadores, Multiplicadores, Desempenho.**

I. Introdução

Os circuitos integrados (CIs) revolucionaram o mundo da eletrônica e estão presentes nos mais variados equipamentos eletrônicos utilizados hoje em dia. São compostos por vários dispositivos semicondutores sobre uma pastilha de silício. Devido à miniaturização dos componentes, podemos obter arquiteturas eletrônicas bastante complexas em uma pequena área. Com os componentes em escalas muito menores, na ordem nanométrica, os produtos eletrônicos atuais apresentam um ótimo desempenho e um aumento no número de funcionalidades [18].

Este trabalho é desenvolvido no grupo de pesquisa em sistemas digitais e embarcados - GSDE da Universidade Federal do Rio Grande - FURG, em parceria com a Universidade Federal de Santa Catarina - UFSC. O projeto é parcialmente financiado pelo PIBIC/PIBIT FURG, pelo CNPq, pela CAPES e pela FAPERGS

Albano Borba é atualmente aluno de graduação em Engenharia de Computação na Universidade Federal do Rio Grande - FURG, Brasil.

Douglas Borges é bacharel em Engenharia de Computação pela Universidade Federal de Rio Grande - FURG. Atualmente está cursando mestrado em Engenharia de Computação no PPGComp/FURG.

Vagner Rosa é doutor e mestre em Ciência da Computação pela Universidade Federal do Rio Grande do SUL - UFGRS. Atualmente, é professor do Centro de Ciências Computacionais da Universidade Federal do Rio Grande - FURG.

Cristina Meinhardt é doutora e mestre em Ciência da Computação pela Universidade Federal do Rio Grande do SUL - UFGRS. Atualmente, é professora do Instituto de Informática e Estatistíca, da Universidade Federal de Santa Catarina - UFSC.

O recente crescimento no uso de dispositivos de computação pessoal (computadores portáteis e aplicativos baseados em áudio e vídeo em tempo real) e sistemas de computação sem fio, fizeram o desempenho e a dissipação de potência tornarem-se parâmetros críticos no desenvolvimento de sistemas digitais. Neste contexto circuitos aritméticos, especialmente somadores e multiplicadores, fazem-se muito necessários, principalmente: 1) no processamento digital de sinais (DSP); 2) no processamento de imagens; e 3) em unidades aritméticas de microprocessadores. Somadores e multiplicadores rápidos e de baixo consumo energético são sempre necessários na construção de sistemas digitais. Nesse meio, é importante salientar que a multiplicação é significativamente afetada pelo consumo energético. Geralmente, uma maior dissipação de potência implica em uma operação em alta temperatura, que tende a gerar falhas no sistema [8].

Além disso, o avanço das redes sem fio traz a necessidade do uso de DSP SoCs de baixa potência e alta velocidade. Circuitos aritméticos são componentes de grande importância para estes circuitos e possuem uma influência bastante significativa nas características de desempenho e potência do sistema onde está inserido. Sendo assim, a escolha de uma determinada arquitetura de multiplicador pode trazer ganho ou perda de desempenho ao DSP [11]. Assim, o objetivo deste trabalho é explorar o projeto de circuitos aritméticos somadores e multiplicadores de 4 bits utilizando modelos disponíveis na literatura, com o uso de transistores MOSFET em tecnologias de litografia nanométrica para projeto de circuitos integrados CMOS. Diferentes arquiteturas de somadores e multiplicadores estão presentes na literatura. Este trabalho visa avaliar suas características quanto ao número de transistores, atrasos de propagação e consumo de energia e realizar uma comparação entre eles através dos resultados obtidos. O objetivo é identificar as melhores alternativas de arquiteturas para estes circuitos, de forma a obter o melhor desempenho ou menor consumo de energia por operação visando obter taxas de processamento compatíveis com as necessárias para imagens e vídeos.

Este artigo inicia apresentando as principais caracteristicas de circuitos somadores e multiplicadores no Capitulo II, introduzindo as arquiteturas de somadores e multiplicadores estudadas e suas principais características. Em sequência, é apresentada a metodologia adotada nesta primeira fase do projeto. Na Seção III, são apresentados os resultados encontrados e as considerações observadas na comparação destas arquiteturas. Finalmente, a Seção IV apresenta as considerações finais.

II. Circuitos Aritméticos

Circuitos aritméticos apresentam um papel fundamental no funcionamento de qualquer sistema eletrônico, dos mais simples controladores aos mais complexos microprocessadores. Estes circuitos são os principais responsáveis pela computação, ou seja, pela realização de operações matemáticas nos sistemas computacionais. Particularmente, somadores e multiplicadores são foco de diversas pesquisas, pois fazem parte do caminho crítico da maioria dos sistemas computacionais [1] [2].

Este trabalho aborda as principais caracteristicas de arquiteturas de somadores e multiplicadores, explorando o projeto destes circuitos em tecnologias CMOS nanométricas.

Um circuito lógico de suma importância na construção destes somadores e multiplicadores de n bits, é o somador completo *(Full Adder - FA)*. Um somador completo trata-se de um circuito que possui três sinais de entrada, A, B e *Carry In* (Cin), e gera como saída a Soma destes mais um *Carry Out* (Cout). A soma de dois bits pode ser obtida combinando os sinais A e B através de uma porta lógica XOR, para adicionar o *Carry In* insere-se uma nova XOR entre ele e o resultado obtido anteriormente, assim conclui-se que Soma = (AB) *Carry In*. O Carry Out é obtido através de uma porta OR que tem como entradas as saídas de duas portas AND. A primeira das AND's tem como entradas a saída da expressão (A XOR B) e o Carry In, e a segunda os bits A e B, logo Cout = (A B). Cin + (A.B).

A nível de transistor existem diferentes topologias para o somador completo. Neste trabalho, adotou-se o circuito apresentado na Figura 1, o somador Mirror CMOS [14]. Este é composto por 28 transistores divididos em duas redes, pull up e pull down, sendo que estas redes se complementam logicamente. O somador CMOS tem como vantagem sua boa condutibilidade e robustez. Estas qualidades são muito importantes ao se trabalhar com tecnologias nanométricas de dimensões muito pequenas e que utilizam tensões baixas. A principal desvantagem fica sendo a sua alta capacitância de entrada, pelo fato de haverem muitos transistores conectados.

A. Somadores de n bits

Existem diferentes arquiteturas que implementam circuitos somadores de n bits. Cada uma das abordagens tem vantagens e desvantagens bem exploradas em relação à área, atraso e potência. O projeto principal no qual este trabalho está inserido explora algumas arquiteturas, visando o desenvolvimento de módulos de soma dedicados para o processamento de imagens. Este projeto está dividido em duas fases, uma inicial, exploratória para definir o desempenho de somadores binários, que são blocos construtivos mais fundamentais para o processamento de áudio e vídeo. Isso é fundamental para o desenvolvimento da segunda fase, que é o processamento de imagens propriamente dito

Esta arquitetura é chamada de CMOS, por explorar os dois planos logicamente e topologicamente complementares [2]. O somador completo admite as entradas "A", "B", que referem-se aos somandos da operação, e "Cin", que representa o sinal de *carry* proveniente do bit anterior da soma. O circuito gera os sinais "Sum", com o resultado da soma, e "Cout", para

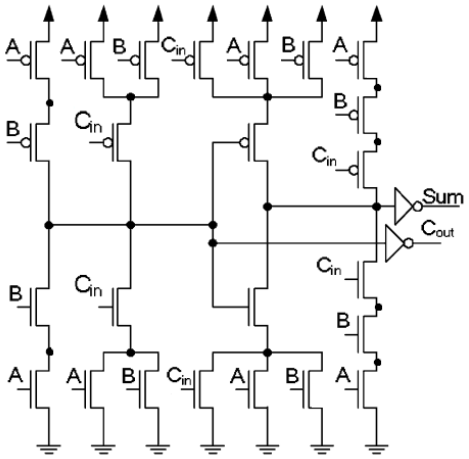

Figura 1. Somador CMOS

o *carry-out* gerado. Estes últimos são representados através equações:

$$Sum = A \oplus B \oplus Cin \tag{1}$$

$$Cout = A.B + A.Cin + B.Cin \tag{2}$$

A escolha deste circuito para o somador completo deve-se ao fato deste ser o mais frequentemente disponibilizado nas bibliotecas de células comerciais e por ser adotado nas unidades lógicas e aritméticas de processadores, tendo bom desempenho comparado a outras arquiteturas . Este módulo de 1 bit é utilizado para a construção de somadores de n bits.

A partir da tecnologia mencionada, como foco deste trabalho foram selecionados quatro diferentes arquiteturas para a implementação dos somadores de n bits: somador *Ripple-Carry*, somador *Carry-Lookahead*, somador *Carry-Select* e o somador *Kogge Stone Tree*. Os principais detalhes de cada uma destas arquiteturas são apresentados nas próximas sub-seções.

1) Somador Ripple-Carry: O somador *Ripple-Carry* é o mais simples que se pode construir. Como representado no exemplo de um somador *Ripple-Carry* de 4 bits da Figura 2, para um circuito de n bits esse somador demanda n módulos *FA* ligados em série. A principal desvantagem desta arquitetura é o caminho crítico gerado. Neste caso, o caminho mais demorado para um sinal ser gerado pelo circuito depende todos os módulos processarem seus valores de propagação de *carry* para então, o valor do bit mais significativo da soma poder ser gerado.

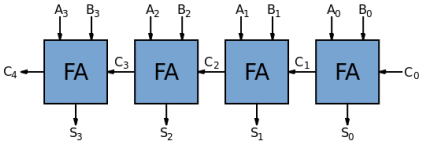

Figura 2. Somador Ripple-Carry

2) Sinais Auxiliares: Os próximos somadores a serem abordados, utilizam-se de dois sinais como auxílio no seu projeto. Estes sinais são utilizados para acelerar a propagação do *carry*

para os próximos bits, oferendo alternativas para redução do problema de caminho crítico do *Ripple-Carry*. Basicamente, são dois principais sinais: *generate e propagate*.

O primeiro deles é o *Generate* (G), que será "1"quando o *carry-out* for "1"independente do *carry-in*, como mostra a equação (3). Já o sinal *Propagate* (P) será "1"quando as entradas forem diferentes, de acordo com a equação (4), representando que o *carry-out* será igual ao *carry-in*. Esses sinais ainda podem ser relacionados com as equações (1) e (2) do *FA* para se obter os sinais de "Sum"e "Cout"em função de P e G, resultando em:

$$G = A.B \tag{3}$$

$$P = A \oplus B \tag{4}$$

$$Sum = A \oplus B \oplus Cin = P \oplus Cin \tag{5}$$

$$Cout = A.B + A.Cin + B.Cin = G + P.Cin \tag{6}$$

3) Somador Carry-Lookahead: Diferentemente do modelo *Ripple-Carry*, onde cada *FA* espera o *carry-out* do anterior para realizar a sua soma, no somador *Carry-Lookahead* cada módulo computa seu próprio bit de *carry-in* independentemente. A Figura 3 mostra a organização desse modelo para 4 bits, onde nenhum *FA* se comunica diretamente. De forma geral, um somador *Carry-Lookahead* de n bits necessita de n módulos somadores e a geração do sinal de propagação de *carry*.

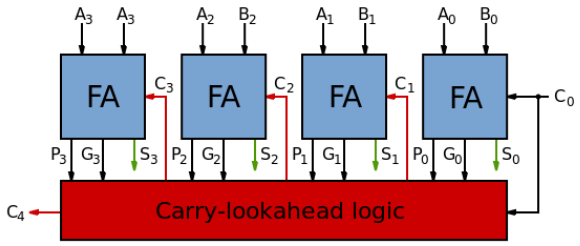

Figura 3. Somador *Carry-Lookahead*

Utilizando os sinais auxiliares o somador *Carry-Lookahead* consegue obter os sinais *carry-out* para cada bit através das equações (7), (8), (9) e (10), definidas como:

$$Cout_0 = G_0 + P_0.Cin_0 \tag{7}$$

$$Cout_1 = G_1 + P_1.G_0 + P_1.P_0.Cin0 \tag{8}$$

$$Cout_2 = G_2 + P_2.G_1 + P_2.P_1.G_0 + P_2.P_1.P_0.Cin0 \tag{9}$$

$$Cout_3 = G_3 + P_3.G_2 + P_3.P_2.G_1 + P_3.P_2.P_1.G_0 + P_3.P_2.P_1.P_0.Cin0 \tag{10}$$

4) Somador Carry-Select: O somador *Carry-Select* consiste basicamente em dois somadores *Ripple-Carry* e um multiplexador. Como mostrado na Figura 4, para a soma de 4 bits são necessários dois somadores em paralelo, onde o primeiro é alimentado com o *carry-in* "0"e segundo com *carry-in* "1". Então o multiplexador seleciona o sinal correto a partir do *carry-in* inicial da soma.

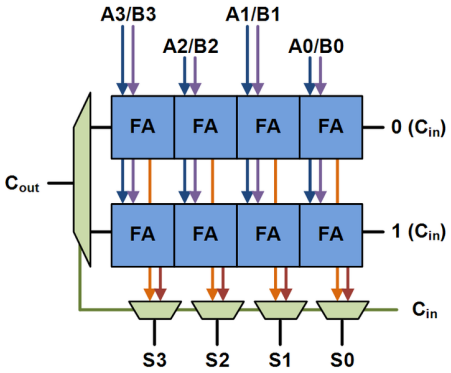

Figura 4. Somador *Carry-Select*

Apesar de apresentar uma boa vantagem na diminuição do caminho crítico em relação as outras abordagens, este somador necessita de mais hardware e consome mais energia. Para aplicações que buscam alto desempenho, somadores ligados em árvores são mais utilizados.

5) Somador Kogge-Stone Tree: Arquiteturas de somadores baseadas em árvores são propostas para somadores de vários bits, ou seja, módulos somadores maiores que 4 bits. Dentre as alternativas propostas na literatura, o somador *Kogge-Stone* destaca-se pelo seu desempenho [3]. A Figura 5 mostra um somador baseado na árvore de Kogge-Stone para 16 bits. Cada tipo de célula nessa arquitetura é responsável por uma equação diferente e dividem-se por camadas, sendo o número de camadas igual a $log_2 n$, como n representando o número de bits do somador.

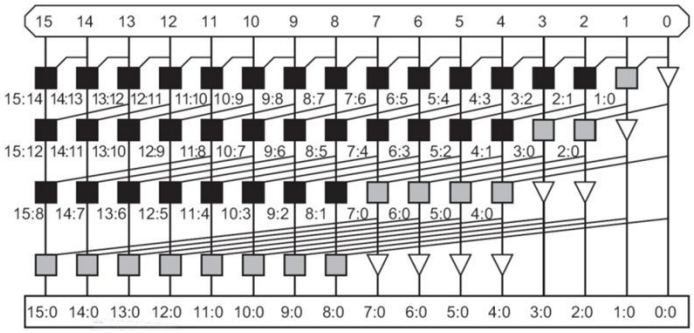

Figura 5. Somador *Kogge-Stone*

Inicialmente, calcula-se todos os sinais de *propagate* e *generate* para cada bit. As outras equações são representadas por:

$$Sum_i = P_i \oplus Cin_i \tag{11}$$

$$G_{i:j} = G_{i:k} + P_{i:k}.G_{k-1:j} \tag{12}$$

$$G_{i:j} = G_{i:k} + P_{i:k}.G_{k-1:j} e P_{i:j} = P_{i:k}.P_{k-1:j} \tag{13}$$

B. Multiplicadores

Há muitas décadas vem-se desenvolvendo circuitos com o objetivo de realizar a operação aritmética de multiplicação. Em um primeiro momento, essa operação era realizada utilizando

iterativamente um circuito somador. Porém, com o passar dos anos, surgiram algumas restrições em relação a taxa de clock, criando a necessidade de obter um hardware dedicado. Neste contexto, começaram a ser propostas implementações de multiplicadores matriciais [20].

O funcionamento de todas as arquiteturas de multiplicadores propostas na literatura pode ser dividido em 3 estágios: geração de produtos parciais, adição de produtos parciais e um estágio final de adição. A velocidade da multiplicação pode ser aumentada diminuindo-se o número de produtos parciais, pois quanto menor for o número de produtos parciais menor será a quantidade de somas necessárias para obter-se o produto final, por isso muitos algoritmos e modificações foram propostos para acelerar este procedimento [12].

Na literatura estão disponíveis multiplicadores com arquiteturas simples que realizam a operação de números binários sem sinal. As principais delas são os multiplicadores Array [8] e Wallace Tree [22]. A vantagem destas é o bom desempenho, porém, são limitadas pelo fato de trabalharem apenas com números binários positivos. Com o tempo foram sendo propostas implementações para a multiplicação de números positivos e negativos representados em complemento de 2.

A representação em complemento de 2 é a mais usada em sistemas digitais [5]. Esta representação adiciona complexidade aos algoritmos de multiplicação porque o sinal do número está inserido em sua própria representação, diferentemente da representação sinal magnitude, onde o sinal e o número podem ser separados, tornando a multiplicação muito mais simples.

Uma das implementações mais conhecidas é a do algoritmo de Booth Modificado [7]. Ela possui um esquema de recodificação junto à uma árvore de redução de profundidade logarítmica e um somador final. Essa implementação tem a vantagem de reduzir o número de produtos parciais gerados pela metade em comparação a geração de produtos parciais baseada em duas portas AND, diminuindo assim o circuito. Como as árvores de redução de profundidade logarítmica possuem estruturas variáveis, o uso de um circuito menor facilita a implementação e melhora o desempenho do multiplicador.

A motivação para o uso da arquitetura de Booth Modificado é o ganho em velocidade. Porém, o fato da dissipação de energia ter se tornado recentemente um fator crítico fez com que o uso de tal implementação deixasse de ser tão irrefutável. Apesar de seu alto consumo, o multiplicador de Booth ainda prevalece quando faz-se necessário um circuito de alta velocidade [20]

Dos trabalhos relacionados encontrados no levantamento bibliográfico destacam-se dois trabalhos recentes. O trabalho de [4] que apresenta uma comparação entre os multiplicadores de Booth Modificado, Vedic, Wallace e Dadda e o trabalho de [20] que compara os multiplicadores de Baugh-Wooley e Booth Modificado.

Das arquiteturas de multiplicadores disponíveis na literatura foram escolhidas duas que são consideradas as principais para multiplicação de números binários com sinal: a arquitetura Baugh-Wooley e a de Booth. Ambas são projetadas para a multiplicação de números inteiros positivos e negativos, representados em complemento de 2. Estas arquiteturas são alvo de muitos estudos e propostas de modificações para aumento de desempenho.

Em circuitos multiplicadores podemos considerar duas entradas, A e B. Este trabalho apresenta arquiteturas para a multiplicação de 4 bits onde as entradas são representadas pelos bits a3a2a1a0 e b3b2b1b0. A saída é o produto P que tem seus bits representados por $p7p6...p0$. Vale lembrar que o resultado de uma multiplicação de dois valores de n-bits é expresso por um valor de $2n$ bits [13]. As Seções a seguir descrevem as principais características dos multiplicadores estudados.

1) Baugh-Wooley: O algoritmo de *Baugh-Wooley* proposto em [5] é uma das melhores opções para multiplicação de números com sinal porque maximiza a regularidade do multiplicador e permite que todos os produtos parciais tenham bits de sinal positivo. A função booleana AND de cada bit do multiplicador por cada um do multiplicando produz os bits dos produtos parciais, estes são ajustados de forma que o sinal negativo seja movido para a última etapa da operação e somados até gerar o produto final [13]. A criação do array de produtos parciais da multiplicação de n-bit é composta por 3 etapas [20]: 1) O bit mais significativo das primeiras $n-1$ linhas de produtos parciais geradas e todos os bits da última linha exceto o mais significativo, são invertidos. 2) O valor lógico 1 é adicionado a coluna à esquerda do bit mais significativo do primeiro produto parcial gerado. 3) O bit mais significativo do resultado final é invertido. A Figura 6 ilustra o processo citado para uma multiplicação de 5 bits.

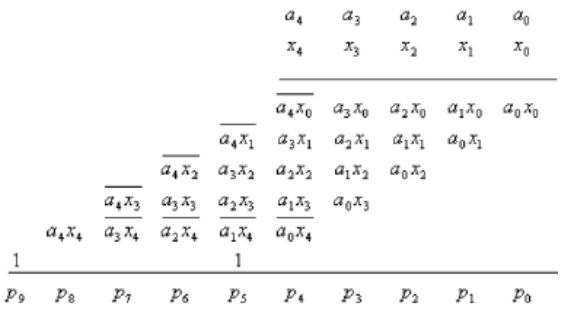

Figura 6. Multiplicação 5*5 *Baugh-Wooley*

Existem diferentes propostas de arquiteturas para o algoritmo de Baugh-Wooley, a arquitetura convencional (BW1) é apresentada na Figura 7. Nesta topologia, a multiplicação é realizada por um circuito composto por 15 somadores, 16 portas lógicas AND (NAND mais inversor) e 8 inversores. Neste diagrama de blocos, os menores representam as expressões lógicas A AND B (A.B), A' AND B (A'.B) e A AND B' (A.B'), enquanto os blocos maiores representam os somadores (FA).

Uma topologia alternativa também baseada no algoritmo de BW e mostrada em [19] é apresentada na Figura 8. Esta arquitetura apresenta uma organização baseada em células que podem ser de dois tipos diferentes, as cinzas que são usadas para operar bits em complemento de 2 e as brancas que são destinadas a multiplicação sem sinal. Células cinzas são compostas por um somador completo (*FA*) com uma porta

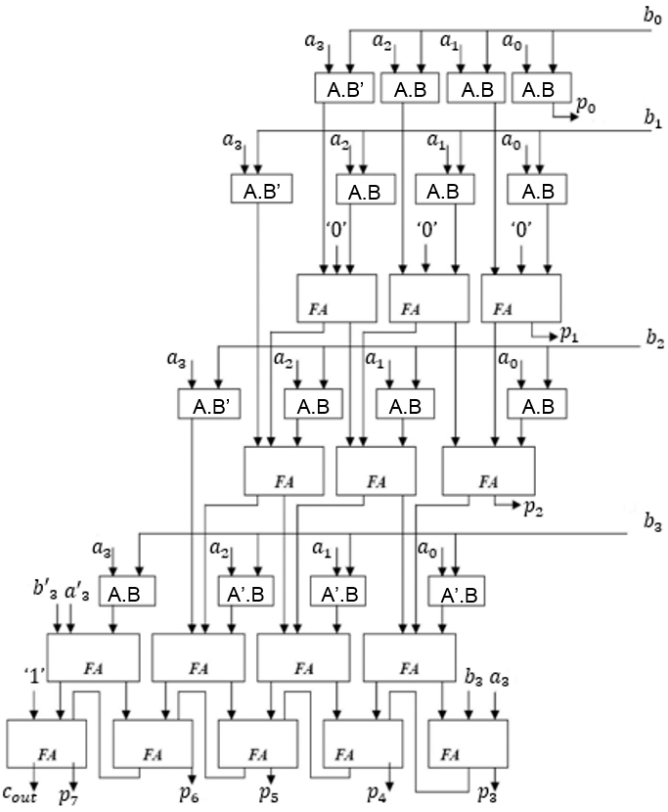

Figura 7. Multiplicador *Baugh-Wooley*

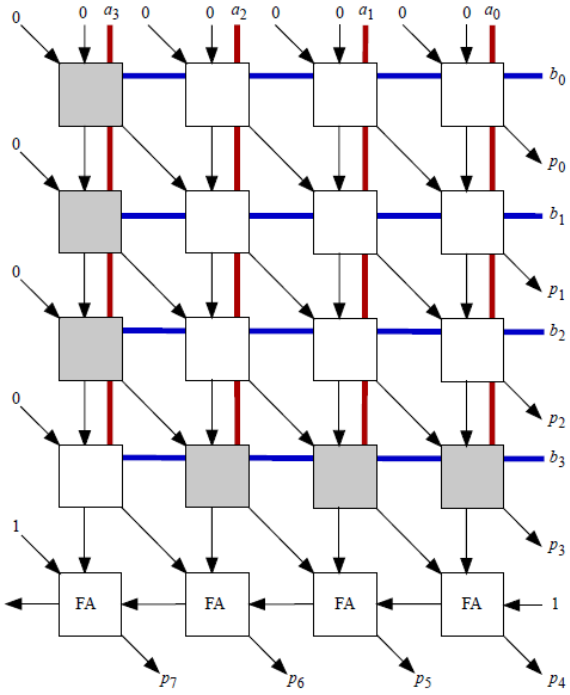

Figura 8. Outra implementação do Multiplicador *Baugh-Wooley* [19]

NAND conectada a uma de suas entradas, já as brancas, possuem uma porta lógica AND nesta função, como pode-se observar na Figura 9. Estas portas geram produtos parciais combinando um bit do multiplicador e um do multiplicando.

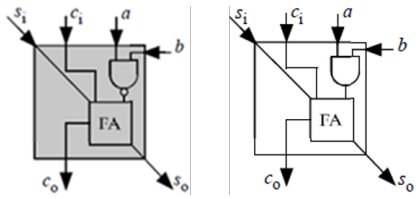

Figura 9. Células do Multiplicador *Baugh-Wooley*

Cada um dos blocos que formam a arquitetura recebe quatro entradas: a entrada B (linha horizontal), a entrada A (linha vertical), o *carry* da célula anterior (linha vertical) e a soma da célula anterior (linha diagonal) e produzem 2 saídas: a soma (linha diagonal) e o *carry* de saída (linha vertical) [19]. Para operações de 4x4 bits são necessárias 6 células cinzas 10 brancas e 4 somadores normais que forma o estágio final de soma.

2) Booth Radix-2: O algoritmo de *Booth* proposto em [6] forma a base dos algoritmos de multiplicação de números com sinal e possui uma implementação simples no nível de *hardware*. Este algoritmo é baseado em recodificar o valor do multiplicador x, para um valor z, deixando o multiplicando y em sua forma original. Na recodificação *Booth*, cada digito do termo multiplicador pode assumir os valores positivo, negativo e zero. Essa é uma notação especial para expressar dígitos com sinal, chamada codificação *Signed Digit* (SD) [10].

Na execução do algoritmo, a etapa mais importante é a recodificação, que tem suas regras mostrada na I utiliza um par de *bits* da entrada x como base. Se o par for "00" ou "11" é realizado apenas um *shift* para a esquerda sobre o produto. Se o par for "01" o valor da entrada y é somado ao produto parcial e se o par for "10" o valor de y é subtraído. Após cada recodificação os *bits* do novo produto devem ser deslizados para a esquerda [16].

Tabela I
TABELA DE RECODIFICAÇÃO BOOTH (RADIX-2)

Bit xi	Bit xi-1	Digito recodificado	Operação em y
0	0	0	Shift (0*y)
0	1	+1	Soma (+1*y)
1	0	-1	Sub (-1*y)
1	1	0	Shift (0*y)

Façamos um exemplo onde o multiplicador é $x = +4$ (0100) e o multiplicando é $y = +3$ (0011) e o produto parcial inicial é $P = 0$ (0000). Primeiramente coloca-se um *bit* "0" depois do bit menos significativo do multiplicador deixando-o no formato "01000".

Observando da direita para a esquerda o primeiro par formado é "00", isto faz com que o produto parcial inicial sofra um *shift*. O segundo par também é "00" então o produto parcial é deslocado novamente. O terceiro par é "10", logo deve-se subtrair o valor de y do produto parcial. Para realizar-se a subtração, o complemento de 2 de y, que tem valor "1101", é adicionado ao produto parcial. O último par formado é "01" então o valor de y, "0011", é adicionado ao produto parcial.

Depois de realizadas estas operações, o *bit* mais significativo de cada produto parcial deve ser replicado para a esquerda até

atingir o comprimento do produto final, isto faz com que o sinal do número seja preservado. Feito isto temos a soma total de produtos parciais que gera o produto final, representada na Figura 10.

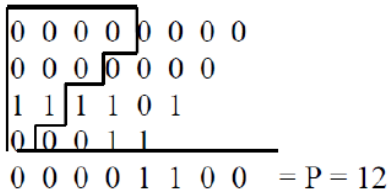

Figura 10. Soma Total

Uma implementação em hardware para esse algoritmo é apresentada na Figura 11. Esta arquitetura é composta basicamente por dois tipos de unidades: a unidade CT (*Control*) que recodifica o multiplicador e a unidade CAS (*Controlled add/subtract*) que realiza as operações definidas pela unidade de controle.

A unidade CT apresentada na figura12, tem como entradas dois bits do multiplicador, xi e $xi-1$, e gera dois sinais, enable e selection. Combinando-se as entradas através de uma porta lógica XOR obtêm-se o sinal enable, este define se o produto parcial será deslocado (se for 0) ou se será realizada uma operação aritmética entre o produto e a entrada y (se for 1). O sinal *selection* é simplesmente o valor da entrada xi e determina qual operação matemática será realizada caso o *enable* esteja ativado, se seu valor for "0" a entrada y será somada ao produto, se for "1", a entrada y será subtraída [9].

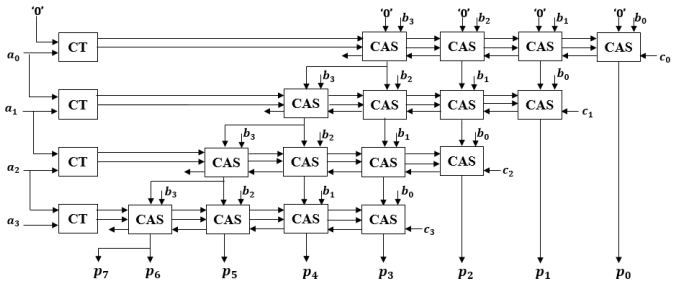

Figura 11. Arquitetura Booth Radix-2

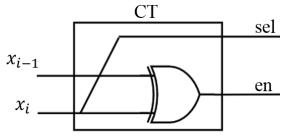

Figura 12. Unidade CT

A unidade CAS mostrada na Figura 13, possui cinco entradas: os sinais de controle *selection* e *enable*, a soma resultante do estágio anterior (Pin), um *bit* do multiplicando e um *carry* de entrada (Cin), e gera uma nova soma (Pout) e um *carry* de saída (Cout). O *bit* proveniente do multiplicando é combinado com o sinal *selection* através de uma porta lógica XOR em seguida a saída desta operação é combinada com o sinal *enable* através de uma porta AND, gerando por fim, o valor que será somada ao produto parcial anterior (Pin) através de um Somador Completo [9].

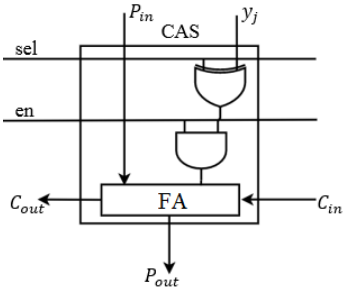

Figura 13. Unidade CAS

Além das duas unidades citadas anteriormente, cada linha do multiplicador possui uma porta lógica AND conectada a unidade CAS colocada mais à direita, esta porta combina os sinais de controle para gerar o *carry* de entrada inicial. Este *carry* serve para adicionar o valor "1" ao *bit* menos significativo quando é necessário converter a entrada y para complemento de 2.

III. Metodologia

Este trabalho apresenta as características de projeto de quatro circuitos somadores e três circuitos multiplicadores de 4 bits. Dois circuitos multiplicadores são baseados no algoritmo de *Baugh-Wooley* (BW1 e BW2) e um no algoritmo de *Booth*. O objetivo do estudo proposto é comparar as implementações em relação a seus tempos de atraso e ao consumo de energia. Para isso, foi utilizado o simulador elétrico Ngspice [15]. Esta ferramenta foi escolhida por atender a demanda do trabalho além de estar disponível gratuitamente.

Antes de realizar-se as simulações, percebeu-se que não faria sentido simular os circuitos aritméticos isoladamente pelo fato de que esses circuitos são usados conectados a outros componentes que possuem suas próprias capacitâncias e resistências. Para reproduzir isso, conectou-se dois inversores em série em cada uma das entradas. Note que esse procedimento não altera o valor lógico da entrada, apenas introduz um comportamento realístico para os sinais de entrada. Em cada uma das saídas, foi conectado um capacitor de 1fF (um femto Faraday).

As próximas Seções apresentam os detalhes adotados nestes experimentos, detalhando a tecnologia e dimensionamento adotados, o framework desenvolvido para obtenção dos arcos de atraso e as principais métricas de avaliação exploradas neste trabalho.

A. Tecnologia e Dimensioanamento

A tecnologia refere-se ao tamanho de transistor utilizado no projeto, mais precisamente ao comprimento de seu canal (L). Uma menor largura de canal oferece um menor caminho a ser atravessado pela corrente elétrica e isto proporciona uma menor necessidade de energia para que esta travessia seja realizada. Neste trabalho foi utilizada a tecnologia preditiva

de 16nm de alta perfomance (*high performance* – HP) disponibilizada pela PTM [17], ela possui uma tensão nominal de 0,7V e foi escolhida por ser uma das mais atuais tecnologias desenvolvidas.

Assim como a tecnologia definida, o dimensionamento do transistor também é um fator importante. Este parâmetro, refere-se a largura do canal (W), característica que interfere diretamente na quantidade de corrente que pode atravessar o transistor. Um canal de maior largura possibilita que mais corrente atravesse-o simultaneamente. Para definir a largura foram utilizadas algumas técnicas.

A primeira técnica adotada define que a largura mínima deve ser no mínimo o dobro do tamanho da tecnologia. Como utilizou-se uma tecnologia de $16nm$, a largura mínima utilizada foi $32nm$. No entanto, deve-se estar atento ao fato de que transistores NMOS são duas vezes mais rápidos que os PMOS, isto se deve a maior mobilidade dos elétrons, predominantes nos dispositivos NMOS, comparada a mobilidade das lacunas, predominantes nos transistores PMOS. Para compensar isso, foi definido que a largura dos transistores PMOS seria o dobro da dos NMOS, sendo assim foram usados transistores NMOS de $Wn = 32nm$ e transistores PMOS de $Wp = 64nm$.

Nos somadores também foi utilizada uma técnica conhecida como *Logical Effort* [21]. Esta abordagem dita que a largura (W) deve ser o tamanho mínimo multiplicado pelo número de transistores em série, além de englobar a técnica de mobilidade apresentada anteriormente. O circuito somador tem papel fundamental na construção de multiplicadores.

B. Dados a serem avaliados

Após uma pesquisa bibliográfica, chegou-se à conclusão que seria pertinente avaliar quatro fatores neste trabalho: número de transistores, atrasos de propagação, potência e o produto entre a potência e o atraso (*Power Delay Product* - PDP). Cada um deles será explicado nas subseções seguintes.

1) Número de Transistores: Neste quesito o trabalho deteve-se a fazer uma estimativa do número total de transistores presentes em cada arquitetura. No geral, este número pode ser usado para estimar a área ocupada pelo circuito porém isto não é tão preciso porque cada arquitetura possui um leiaute e pode ser construída adotando diferentes modos de dimensionamento de cada dispositivo.

2) Atrasos: Os tempos de atraso mostram o quanto um circuito é eficiente. Estes valores indicam o tempo de resposta das portas lógicas e fazem-se uteis para avaliar o desempenho. Neste trabalho foi estudado o tempo de propagação. Este trata-se do intervalo de tempo que a saída do circuito demora para transicionar de valor (de 0 para 1 ou de 1 para 0), devido a uma transição na entrada. Estes tempos são medidos a partir de 50% da transição da onda de entrada até 50% da transição da onda da saída. Existem dois tipos de tempo de propagação: o *high-to-low* (tpHL) onde a saída vai de 1 para 0 e o *low-to-high* (tpLH) onde a saída vai de 0 para 1.

Nos circuitos estudados, foi feita uma avaliação exaustiva de todos os tempos de propagação. Para isso, desenvolveu-se aplicações para identificar todos os arcos de atraso dos circuitos somadores e dos circuitos multiplicadores. Arcos de atraso são pares de possíveis entradas que possuem apenas um bit de diferença e geram saídas diferentes. A Tabela II mostra um exemplo de arco, onde o *bit* a0 transiciona de 0 para 1, provocando uma alteração no *bit* menos significativo da saída, $p0$. Esses arcos consistem em pares de entradas (sendo cada entrada representada por dois números binários com n bits) que diferem em apenas 1 *bit* mas que gerem saídas (a soma dos dois números) distintas, sendo estas os n bits de soma e o último *carry-out*. Tendo em vista a automatização desse processo, desenvolveu-se um $framework$ que encontra o conjunto de arcos válidos de acordo com o número n de bits solicitados.

Tabela II
EXEMPLO DE ARCO DE ATRASO

TpLH [1,17]			
ID entrada	Entrada (AB)	Operação (A x B)	Saída (Produto)
1	00000001	0000 x 0001	00000000
17	00010001	0001 x 0001	00000001

C. Framework para identificação dos arcos de atraso

A principal função do framework é a identificação dos arcos. Para isso, foram utilizados dois laços encadeados, percorrendo assim todas as combinações de uma entrada com as demais entradas para identificar as que diferem em apenas 1 *bit*. Ao identificar pares de entradas com apenas um bit de diferença, é verificado se este par provoca alguma alteração em alguma das saídas dos circuitos somadores ou multiplicadores. Se houver transição devido a este par, o arco é adicionado a uma lista de arcos de cada uma das saídas sensibilizadas com a transição.

Após montar a lista de arcos esta é percorrida por um laço de repetição. A cada iteração as fontes de entrada do circuito são alteradas de acordo com o par que forma o arco e é feita uma chamada ao Ngspice para simular o circuito com esta configuração e gerar os valores de atraso e consumo de energia.

Gasto de energia e atraso são parâmetros de extrema importância para avaliar circuitos elétricos e desenvolver melhorias. Para verificar a eficiência utilizando essas variáveis, a partir do conjunto de arcos de transição, um outro $script$ monta arquivos $spice$ para a simulação dos tempos de propagação e o consumo de energia para a avaliação do desempenho dos arcos de transição. O arquivo $spice$ é divido em três partes, uma primeira onde são declaradas as fontes que compõem o circuito. Cada bit de entrada é descrito como uma fonte DC, com exceção do *bit* variável que é simulado em estado de subida, *low-high*, e de descida, *high-low*. Na segunda parte é descrito o somador, de acordo com o modelo a ser utilizado na avaliação, gerando os sinais de saída. Por fim são calculados os tempos de atrasos dos sinais e o seu consumo de energia para serem comparados.

A mesma aplicação configurou os arquivos de entrada do simulador e fez as devidas chamadas de simulação para todos os casos de estudo.

É importante destacar que as simulações foram feitas representando os bits de entrada com fontes de tensão onde, em cada par, a fonte que representava o bit que transiciona

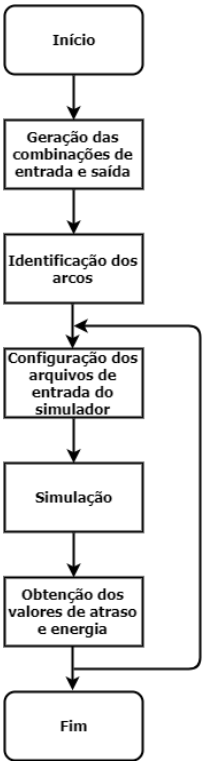

Figura 14. Fluxograma Aplicação

no arco, mudava de estado depois de um período de tempo, enquanto as demais permaneciam constantes. O período de tempo adotado para o sinal de entrada ficar estável em cada um dos níveis lógicos foi de $2ns$. A aplicação identificou 5088 arcos de atraso para a multiplicação. A Figura 14 apresenta um fluxograma dos passos desenvolvidos pelo programa para a multiplicação. A primeira etapa consiste em preencher duas matrizes de dimensão 8x256. A matriz Entrada possui como linhas todas as combinações de entradas possíveis para o multiplicador, já a matriz Saída apresenta o produto dos quatro primeiros bits pelos quatro últimos da linha i correspondente na Entrada.

1) Potência: Observar a potência dissipada dos circuitos faz-se importante para escolher uma arquitetura que consuma menos energia. Isto se faz cada vez mais importante devido ao grande desenvolvimento dos dispositivos eletrônicos portáteis que fazem uso de baterias. Também é importante ressaltar que alto consumo pode gerar sobreaquecimento do dispositivo gerando falhas operacionais.

O simulador elétrico utilizado neste trabalho não possuí uma função especifica para medir a potência, mas determina a energia consumida em um determinado intervalo de tempo. Através da Equação 1 obtêm-se a energia, calculando a integral da corrente consumida da fonte durante o tempo de cada simulação (4ns). Nesta equação, i corresponde à corrente, Δt à variação de tempo e Vdd à tensão na fonte.

$$Energia = \int_0^{\Delta t} (Vdd)dt \tag{14}$$

Depois de obtido o valor da energia, calculou-se a potência a partir da Equação 2, que determina que a potência média é a razão entre a energia consumida ao longo do intervalo de tempo, multiplicados pela tensão de alimentação. Estes valores foram observados para cada simulação de arco de atraso.

$$P(Vdd) = \frac{Energia}{\Delta t} * Vdd \tag{15}$$

2) Cálculo do PDP: Para verificar a eficiência utilizando potência e atraso, foi utilizado um fator denominado *power-delay-product* (PDP) que consiste no produto entre os mesmos. Este trabalho avalia os valores de PDP para os cinco arcos de que apresentaram o maior valor de atraso, considerando a respectiva potência dissipada nestes casos.

$$PDP = Atraso * Potencia \tag{16}$$

IV. Resultados

Os dados obtidos, com exceção do número de transistores, foram tratados utilizando as funções matemáticas: média, máxima e desvio padrão. Os resultados serão mostrados divididos em quatro seções: a) Estimativa do número de transistores; b) Medição dos atrasos de propagação; c) Medição de Potência; d) *Power-delay-product* (PDP).

A. Estimativa do número de transistores

O número de transistores total em cada arquitetura permite uma estimativa de área. Os circuitos dos somadores apresentam 102 transistores no *Ripple-Carry* (RC), 240 no *Carry-Lookahead* (CL), no 280 *Carry-Select* (CS) e 192 no *Kogge-Stone Tree* (KST). O número de transistores nos multiplicadores é superior, sendo 532, 644 e 888 para as arquiteturas BW1, Bw2 e Booth respectivamente. A Figura 15 apresenta um gráfico de comparação dos valores obtidos. Avaliando os valores apresentados pode-se notar que entre os somadores o *Ripple-Carry* apresenta o menor número de transistores, e o BW1 é o menor dos multiplicadores. Em contrapartida, *Booth* é o circuito composto por mais transistores. Nos somadores, arquiteturas com propagação de carry, *Carry-Select* e *Carry-Lookahead*, apresentaram as maiores estimativas de área.

exe

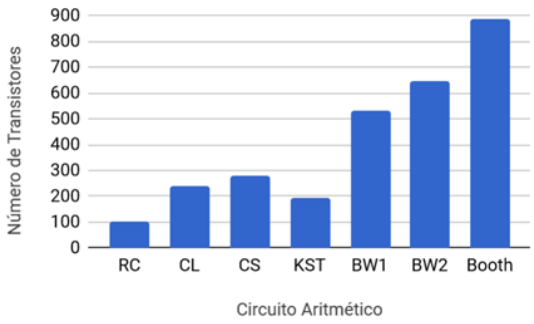

Figura 15. Comparação número de transistores

B. Análise dos tempos de propagação

A análise dos tempos de propagação permitem estimar o pior atraso de cada circuito, ou seja, os tempos críticos. Para os circuitos somadores os resultados são apresentados na Tabela III. Observa-se que para 4-bits a arquitetura *Carry-Select* apresenta os melhores resultados com tempos máximos de atraso significantemente inferiores aos demais somadores. O somador *Carry-Lookahead* apresentou o pior desempenho, com tempos superiores a todos os outros somadores avaliados, e com grande desvio padrão entre os atrasos.

Tabela III
COMPARAÇÃO DOS TEMPOS DE PROPAGAÇÃO DO SINAL

Modelo	Maiores atraso (ns)			Média (ns)	DP (ns)
Ripple-Carry	0,81	0,81	0,79	0,25	1,65
Carry-Lookahead	3,56	3,56	3,56	0,64	5,65
Carry-Select	0,40	0,40	0,40	0,34	0,97
Kogge-Stone Tree	1,39	1,40	1,40	0,50	3,41

O multiplicador BW1 possui os menores tempos médios para todas as saídas, com exceção da p7 onde o Booth mostra-se mais eficiente. O multiplicador Booth possui atrasos médios menores que BW2 para as quatro saídas correspondentes aos bits mais significativos do produto. Entretanto, o mais importante é analisar os valores máximos de atraso porque estes mostram com o circuito se comporta no pior caso. O multiplicador BW1 possui os menores máximos enquanto Booth apresenta os maiores. As Figuras 16 e 17 apresentam gráficos de comparação entre os tempos de atrasos médios, máximos e desvio padrão para ambos multiplicadores.

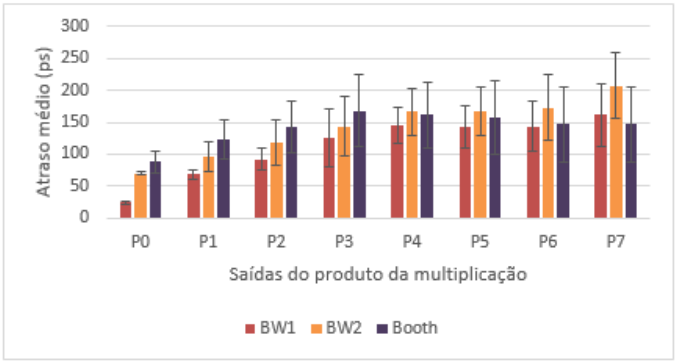

Figura 16. Comparação atrasos médios (ps) e os desvios padrão para cada uma das saídas

A arquitetura BW1 apresenta tanto o menor atraso médio como também o menor valor máximo. Os multiplicadores BW2 e Booth apresentaram atrasos médios semelhantes, porém o primeiro leva vantagem em questão de eficiência por possuir um atraso máximo menor.

C. Potência

Na Tabela IV estão dispostos os valores médios e máximos relacionados a potência dos circuitos aritméticos. Dentre os somadores, o *Carry-Select* foi o que apresentou o menor consumo de energia. Por outro lado, o *Kogge-Stone Tree* apresentou grande consumo de energia, sendo praticamente 5 vezes superiores ao encontrados no *Ripple-Carry*.

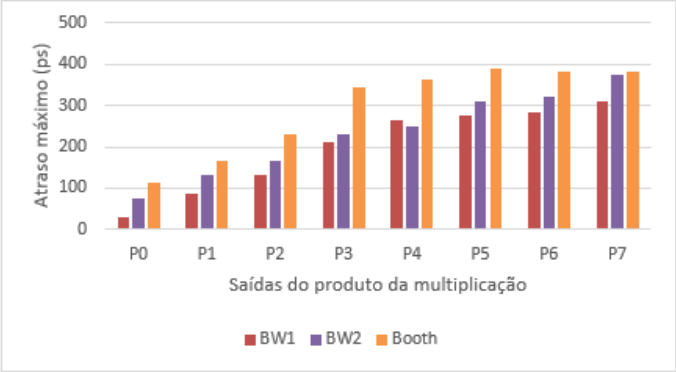

Figura 17. Comparação atrasos máximos (ps)

Tendo em vista os resultados para os multiplicadores, percebe-se que BW1 e BW2 dispõem de valores semelhantes, além de apresentar um consumo energético 29% e 24% menor que o do Booth considerando a potência média. Considerando a potência máxima a economia ainda é maior, aproximadamente 33% .

Tabela IV
POTÊNCIA MÉDIA E MÁXIMA

Arquitetura	Médio (μW)	Máximo (μW)
Ripple-Carry	1,81	3,59
Carry-Lookahead	0,76	1,73
Carry-Select	0,15	0,35
Kogge-Stone Tree	5,19	9,11
BW1	3,70	7,09
BW2	3,92	7,00
Booth	5,19	10,57

D. Ánalise do PDP

A Figura 18 ilustra uma comparação entre os valores de PDP calculados para cada arquitetura de circuito aritmético avaliado. O somador *Carry-Select*, devido ao seu baixo consumo energético, apresentou um PDP significativamente melhor comparado aos demais somadores. Conforme os resultados obtidos, pode-se constatar que BW1 apresentou um PDP máximo 16 menor que o do BW2 e 46 menor que o do Booth.

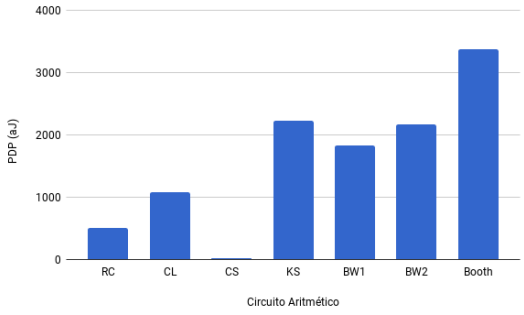

Figura 18. Comparação dos PDP's (aJ)

V. Conclusão

Neste trabalho foram avaliadas diferentes arquiteturas de circuitos aritméticos de 4 bits para números positivos e negativos, com entrada representada em complemento de 2. Estes circuitos foram simulados adotando a tecnologia CMOS de 16nm. Estudou-se a quantidade de transistores utilizada e as características elétricas de atraso e potência. Nos somadores, observou-se que a arquitetura *Carry-Select* apresentou os melhores resultados para o pior atraso e atraso médio. Na avaliação de potência a arquitetura *Kogge-Stone Tree* apresentou um consumo muito elevado, se mostrando pouco viável para a análise em somadores de 4 bits. Já o modelo *Carry-Select* gerou um baixo consumo e consequentemente um PDP significativamente melhor quando comparado aos demais somadores. A estimativa do número de transistores se mostrou mais elevada para as arquiteturas com propagação de sinal, mas espera-se que esse número cresca exponencialmente de acordo com o número de bits utilizados para somadores em árvores, como na arquitetura *Kogge-Stone Tree*.

Em relação aos multiplicadores, quanto aos tempos de atraso conclui-se que BW1 é a arquitetura com melhor desempenho por apresentar os menores valores médios e máximos. Referente à potência , chegou-se à conclusão de que o uso das arquiteturas BW1 e BW2 impacta respectivamente em economia de 29% e 24% no consumo médio, em relação ao Booth. Este resultado está relacionado ao fato de que o multiplicador Booth apresenta um maior número de transistores, como já foi dito anteriormente. No que diz respeito ao PDP, é possível observar que BW1 apresentou um valor máximo 16% menor que o do BW2 e 46% menor que o do Booth. Este resultado indica que BW1 possui o melhor desempenho avaliando-se atraso e consumo simultaneamente.

Este trabalho demonstra a relevância da escolha da arquitetura mais adequada para cada tipo de projeto de circuito aritmético e o impacto nos atrasos e potência em circuitos de 4 bits. Para circuitos aritméticos de maior número de bits, como 6, 16, 32 ou 64 bits, determinadas arquiteturas podem ter suas características melhor explorada, sendo diferente a conclusão de melhor arquitetura. Como continuidade deste trabalho, estas arquiteturas serão exploradas com maior número de bits, assim como serão explorados outros circuitos somadores de 1 bit como célula componente dos demais circuitos aritméticos. Estes circuitos tem aplicação direta em circuitos integrados para processamento de video e tratamento de sinais.

Agradecimentos

Os autores gostariam de agradecer ao suporte fornecido pelo CNPq, CAPES, FAPERGS e PIBIC/PIBIT FURG.

Referências

[1] M. Alioto and G. Palumbo, “Analysis and comparison on full adderblock in submicron technology,”IEEE Transactions on Very Large ScaleIntegration (VLSI) Systems, vol. 10, no. 6, pp. 806–823, 2002.

[2] C.-H. Chang, J. Gu, and M. Zhang, “A review of 0.18um full adderperformances for tree structured arithmetic circuits,”IEEE Transactionson very large scale integration (VLSI) systems, vol. 13, no. 6, pp. 686–695, 2005.

[3] P. M. Kogge and H. S. Stone, “A parallel algorithm for the efficientsolution of a general class of recurrence equations,”IEEE transactionson computers, vol. 100, no. 8, pp. 786–793, 1973.

[4] S. Abraham, S. Kaur, S. Singh Study of Various High Speed Multipliers. Int. Conf. on Computer Communication and Informatics (ICCCI-2015), Jan. 08-10, 2015, Coimbatore, INDIA.

[5] C. Baugh, B. Wooley. A Two’s Complement Parallel Array Multiplication Algorithm. IEEE Trans. on Computers, Vol. C-22, No. 12, Dec. 1973.

[6] A. Booth. A Signed Binary Multiplication Technique. The Quarterly Journal of Mechanics and Applied Mathematics, vol. 4, no.2, Jan, 1951, pp. 236–240.

[7] J. Fadavi-Ardekani. MxN Booth Encoded Multiplier Generator Using Optimized Wallace trees. IEEE Trans. on Very Large Scale Integration (VLSI) Systems, vol. 1, no. 2,pp. 120–125, 1993.

[8] N. Goel, L. Garg. Comparative Analysis of 4-bit CMOS Multipliers. Int. Conf. on VLSI, Communication; Instrumentation (ICVCI) 2011.

[9] W. Huang, Z. Cai, S. Shen. Comparison of Different Architectural Design Methods for Multiplier. Disponível em: //venividiwiki.ee.virginia.edu/mediawiki/index.php/ClassECE6332Fall15GroupMultipler.

[10] S. Kaur, S. Manna. Implementation of Modified Booth Algorithm (Radix-4) and its Comparison with Booth Algorithm (Radix-2). Advance in Eletronic and Eletric Engineering. ISSN 2231-1297, Vol 3, no 6 (2013), pp. 683-690.

[11] A. Khatibzadeh, K. Raahemifar, M. Ahamadi, M. A Novel Multiplier For High-Speed Applications. IEEE International SOC Conference, 2005. 25-28 Sept.

[12] V. MANJUNATH et al. Design and Implementation of 16x16 Modified Booth Multiplier. Online Int. Conf. on Green Engineering and Technologies (IG-GET 2015).

[13] P. Mohanty. An Efficient Baugh-Wooley Architecture for Signed and Unsigned Fast Multiplication. NIET Journal of Engineering; Technology, Vol.1, Issue, 2013.

[14] K. Navi et al. A novel low-power full-adder cell for low voltage. The VLSI Journal, v. 42, n. 4, p.457-467, set. 2009.

[15] NGSPICE. NGSPICE Circuit Simulator. Disponível em: http://ngspice.sourceforge.net.

[16] D. Patidar, J. Chitode. Efficient Architecture for Radix-2 Booth Multiplication Using 4:2 Compressors. International Journal Of Scientific Research And Education. Vol 3, no 6, pp:3529-3538. Junho-2015.

[17] PTM. Predictive Technology Model. Disponível em: http://ptm.asu.edu.

[18] J.Rabaey, A. Chandrakasan, B. Nikolic. Digital integrated circuits: a design perspective. 2nd ed. Upper Saddle River: Prentice Hall, 2003.

[19] V. Rajmohan, O. Maheswri. Design of Compact Baugh-Wooley Multiplier Using Reversible Logic. Circuits and Systems, 7, 1522-1529. 2016.

[20] M. Sjalander, P. Larsson-Edefors. High-speed and low-power multipliers using the Baugh-Wooley algorithm and HPM reduction tree. Electronics Circuits and Systems 2008. ICECS 2008, pp. 33-36, 2008.

[21] I. Sutherland, R. Sproull, D. Harris. Logical Effort: Designing Fast CMOS Circuits. San Francisco, USA: Morgan Kaufmann Publishers, 1998.

[22] C. Wallace. A Suggestion for a Fast Multiplier. IEEE Trans. on Electronic Computers, EC-13, pp. 14-17, 196.