Uma Arquitetura Rápida de Somador Binário de Alta Eficiência Energética

Abstract. *The design of battery-powered portable devices requires energy efficient fast adders. Although many adder architectures exist, only a few can be synthesized in a standard-cells flow. This paper presents the A1CSAH, a novel energy efficient fast adder architecture. The A1CSAH, along with other architectures (CRA, CSA and CLA), were synthesized for a 45nm standard-cells library. Synthesis results show that the A1CSAH is, on average, 10.8% faster and 3.4% more energy-efficient than the CLA, thus corresponding to an excellent fast adder option.*

Resumo. *O projeto de dispositivos portáteis alimentados por baterias requer o uso de somadores rápidos e energeticamente eficientes. Apesar de existirem muitas arquiteturas de somadores, poucas delas podem ser sintetizadas em um fluxo standard cells. Este artigo apresenta o A1CSAH, uma nova arquitetura de somador rápido energeticamente eficiente. O A1CSAH, juntamente com outras arquiteturas (tais como CRA, CSA e A1CSA), foram sintetizados para a biblioteca standard cells de 45nm da TSMC. Os resultados da síntese lógica mostram que o A1CSAH é, em média, 10,8% mais rápido e 3,4% mais eficiente do que o CLA, o que o torna uma excelente opção de somador rápido.*

1. Introdução

Nos últimos anos os dispositivos eletrônicos pessoais, tais como *smartphones*, *tablets* e consoles portáteis de jogos, têm sido responsáveis pela significativa expansão do mercado de eletrônica de consumo. Estes dispositivos devem operar com baterias e ao mesmo tempo, ser capazes de executar aplicações de domínios específicos, tais como codificação e decodificação de áudio e vídeo, as quais requerem um grande poder de processamento. A fim de satisfazer aos requisitos de baixo consumo de energia e alto desempenho, o projeto destes dispositivos exige o desenvolvimento de blocos de *hardware* capazes de realizar as aplicações críticas com o desempenho necessário e com alta eficiência energética, prolongando assim a carga da bateria.

A adição é a operação aritmética mais comumente utilizada em sistemas integrados, constituindo-se frequentemente em fator limitante do desempenho do sistema como um todo [Rabaey, Chandrakasan e Nikolic 2003]. Além de ser utilizada

como uma operação *per se*, a adição também serve de base para outras operações aritméticas também muito usadas, como a subtração e a multiplicação [Takagi et al. 2007]. Por outro lado, dependendo do domínio de aplicação alvo, os sistemas computacionais apresentam diferentes requisitos de velocidade, quantidade de recursos de *hardware* (área em silício) e consumo de energia, o que justifica o contínuo desenvolvimento de novas arquiteturas de somadores.

Um somador binário de *n* bits é um componente do bloco operativo (*datapath*) que adiciona dois operandos de entrada, os quais são números binários de *n* bits ($A_{0...n-1}$ e $B_{0...n-1}$), e eventualmente um *Carry-in* (CIN) de 1 bit. O resultado da operação, referido por "soma", é um vetor de *n* bits ($S_{0...n-1}$) e um *Carry-out* (COUT) de 1 bit. Como a grande maioria dos sistemas computacionais representa e manipula dados numéricos em base 2, os somadores binários geralmente são referenciados apenas por "somadores".

As diversas formas de se implementar o somador são geralmente referenciadas por "arquiteturas" de somadores. A arquitetura conhecida como *Carry-Ripple Adder* (CRA) [Hwang 1979][Oklobdzija 2001] inspira-se no método de adição manual, resultando em circuitos com área bastante reduzida, uma vez que faz uso de expressões Booleanas na forma fatorada. Contudo, nesta arquitetura o *carry* precisa ser propagado por todos os bits até atingir o bit mais significativo, resultando em grande atraso crítico e por conseguinte, baixa velocidade de operação.

A arquitetura *Carry-Select Adder* (CSA) [Bedrij 1962] foi proposta com o intuito de acelerar a propagação do *carry* em relação ao CRA. A filosofia deste tipo de somador consiste em tomar os operandos de entrada por segmentos, realizando a adição de cada segmento de forma paralela. Para cada segmento são computadas duas somas candidatas: uma com CIN=0 e a outra com CIN=1. Quando os resultados estiverem disponíveis, a soma candidata correta para cada segmento é selecionada. A área e o consumo de energia do CSA são mais do que o dobro da área e do consumo de energia do CRA. Porém, o atraso crítico do CSA é significativamente menor.

A arquitetura *Add-One Carry-Select Adder* (A1CSA) é derivada da CSA, tendo sido apresentada como uma alternativa de menor área e com menor atraso [Chang e Hsiao 1998]. A ideia básica é, em cada segmento utilizar um somador para realizar a adição com CIN=0 e ao resultado, somar 1 usando um circuito dedicado, menos complexo do que o somador. Desta forma, somente um bloco somador é usado por segmento de soma, reduzindo a quantidade de recursos em relação ao CSA.

Nas arquiteturas mencionadas nos parágrafos anteriores (CRA, CSA e A1CSA) o *carry* é propagado em tempo linear em relação ao número de bits dos operandos, pois ele deve ser propagado por todos os bits de soma. O *Carry-Lookahead Adder* (CLA) [Weinberger e Smith 1958] tem como principal característica o cálculo de todos os *carries* em paralelo, dentro de cada segmento de soma. Além disso, a propagação do *carry* entre os segmentos de soma faz uso de uma cadeia em formato de árvore binária, o que lhe confere tempo de propagação logarítmico [Oklobdzija 2001].

O presente artigo descreve uma nova arquitetura de somador rápido denominada de *Hierarchical Add-One Carry-Select Adder* (A1CSAH) [Omitido1], a qual foi desenvolvida pelo primeiro autor como trabalho de iniciação científica. Esta arquitetura

foi derivada do A1CSA mediante a adoção de uma cadeia de propagação de *carry* em forma de árvore binária, resultando em menor atraso em relação ao A1CSA original. Uma segunda contribuição do trabalho reside na otimização do bloco que realiza a seleção da soma correta dentro de cada segmento de soma. Tal otimização resultou em redução de recursos de *hardware* e em redução do atraso de propagação do *carry* do A1CSAH. O A1CSAH e os demais somadores mencionados nos parágrafos anteriores foram sintetizados para uma biblioteca *standard cells* com uso da ferramenta *Design Compiler*, da empresa Synopsys (Synopsys, 2009). Os resultados obtidos, em termos de área, atraso e eficiência energética, mostram que o A1CSAH é bastante apropriado para sistemas integrados que equipam dispositivos eletrônicos pessoais alimentados por bateria.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta detalhes das arquiteturas CRA, CSA, A1CSA e CLA. Na seção 3 é apresentado o A1CSAH, contribuição deste trabalho de iniciação científica. A Seção 4 descreve detalhes da síntese lógica dos somadores, ao passo que a Seção 5 apresenta e comenta os resultados da síntese. A Seção 6 apresenta as conclusões do trabalho.

2. Somadores Binários

A arquitetura CRA é construída a partir de um conjunto de circuitos *Full-Adders* (FAs) e eventualmente, um circuito *Half-Adder* (HA). Um HA é um circuito combinacional que adiciona um par de bits de pesos iguais (A_i, B_i), pertencentes aos operandos A e B respectivamente, e gera um bit de soma (S_i) e um bit de transporte de saída (ou *carry* de saída) (C_{i+1}). Um FA é circuito combinacional que adiciona três bits de pesos iguais, sendo dois deles (A_i, B_i) pertencentes aos operandos A e B e o terceiro, um bit de transporte de entrada (ou *carry* de entrada) (C_i) proveniente da adição de ordem imediatamente inferior, e gera um bit de soma (S_i) e um bit de transporte de saída (C_{i+1}) [Oklobdzija 2001].

A arquitetura CRA é construída conectando-se o *carry* de saída de um FA com o *carry* de entrada do FA seguinte, conforme ilustrado na Figura 1. Neste trabalho, o *carry* de entrada de menor ordem de um somador (C_0) é referenciado por CIN, ao passo que o *carry* de saída de maior ordem (C_{n-1}), é referenciado por COUT. Caso CIN seja sempre zero, a adição do par (A_0, B_0) pode ser feita com um HA. Como em qualquer outro somador, a velocidade de operação depende da propagação do *carry*. No CRA o *carry* é propagado em ***O(n)***, no qual *n* é o número de bits dos operandos.

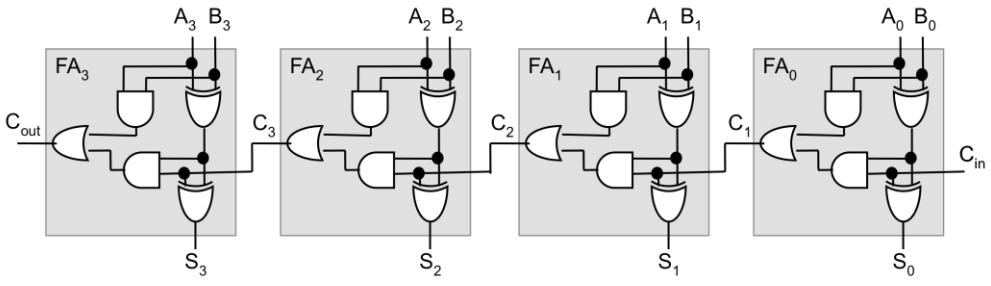

Figura 1. CRA de 4 bits

Uma maneira de reduzir o atraso crítico na adição de dois operandos de *n* bits é tomar os operandos em *m* segmentos de *k* bits (tipicamente, *k*=4), realizando em

paralelo a adição dos *m* segmentos. Tal abordagem deu origem à arquitetura CSA, proposta por Beidrij [Bedrij 1962]. No CSA, para cada segmento são computadas em paralelo duas somas candidatas: uma soma com *carry* de entrada igual a 0 (zero) e outra com *carry* de entrada igual a 1 (um). Para cada segmento, a soma correta é escolhida dentre as duas somas calculadas usando um sinal de seleção computado a partir dos *carries* gerados nos segmentos menos significativos. A Figura 2 mostra o diagrama de blocos de um CSA de 16 bits. Note que no segmento de menor ordem apenas uma soma precisa ser calculada. Nos demais segmentos são calculados duas somas. Cada soma é calculada usando um CRA como bloco somador básico.

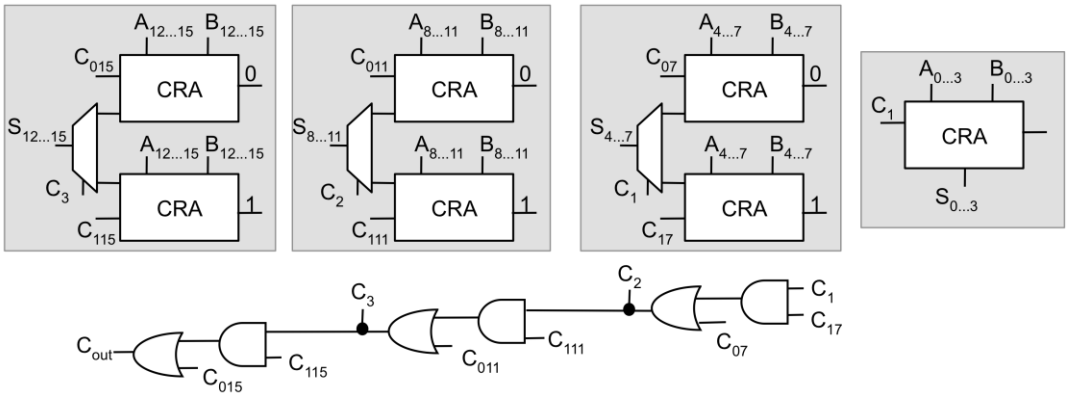

Figure 2. CSA de 16 bits

No CSA o *carry* é propagado como se ele "pulasse" cada segmento de soma. Se cada segmento possuir 4 bits, a propagação do *carry* no CSA seria, teoricamente, 4 vezes mais rápida do que a propagação do *carry* no CRA. Assim, o atraso crítico no CSA é caracterizado pela função assintótica ***O(n/4)***.

Chang e Hsiao propuseram uma arquitetura de somador derivada do CSA, batizada de *Add-One Carry-Select Adder* (A1CSA) [Chang e Hsiao, 1998]. Observando que em um CSA as duas somas calculadas em um dado segmento diferem sempre de uma unidade, eles substituíram o CRA usado para calcular a soma com *carry* de entrada igual a 1 por um circuito dedicado (e portanto com menos recursos de *hardware*), capaz de somar 1 ao resultado da soma com *carry* de entrada igual a 0. Tal circuito dedicado é denominado de A1 (*Add-One*). A Figura 3 mostra um A1CSA de 16 bits e 4 segmentos de soma com 4 bits.

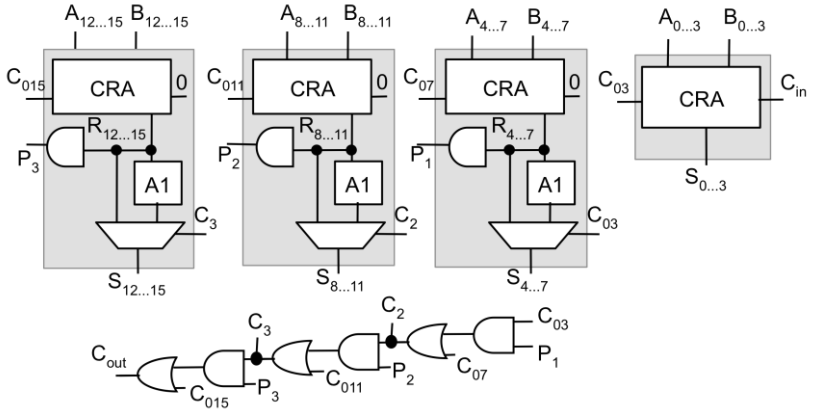

Figura 3. A1CSA de 16 bits

A concepção do A1 baseia-se nas Propriedades 1, 2 e 3 da adição binária, descritas a seguir [Mesquita et al. 2007].

Propriedade 1. *Dados dois números a serem somados, se ambos os números possuírem o bit menos significativo igual a 0, o resultado da soma destes números com CIN igual a 1 é idêntico ao resultado da soma com CIN igual a 0, exceto pela inversão do valor do bit menos significativo do resultado.*

Propriedade 2. *Se o resultado de uma adição entre dois números com CIN igual a 0 apresenta o bit menos significativo igual a 1, o resultado da adição destes mesmos dois números, mas com CIN igual a 1, é equivalente, sendo necessária a inversão dos m bits menos significativos do resultado até que seja encontrado o primeiro 0. Este último também deve ser invertido.*

Propriedade 3. *Dois somadores idênticos recebendo os mesmos operandos de entrada, porém com carries de entrada distintos, exibem o mesmo carry de saída, exceto quando o E lógico (AND) entre todos os sinais propagate dos somadores resulta em um valor igual a 1.*

O CLA é uma arquitetura somadora proposta por Weinberg e Smith [Weinberg e Smith, 1958] para propagar o *carry* em tempo logarítmico, ***O****(logn)*. No CLA, ao invés do *carry* "pular" um determinado número de bits, ele é calculado em paralelo para todos os bits dentro de um segmento de soma. Assim, cada soma de pares de bits dos operandos tem disponível o *carry* de entrada praticamente ao mesmo tempo, o que acelera o cálculo da soma. Entretanto, a quantidade de recursos necessários para o cálculo em paralelo dos *carries* cresce rapidamente com o aumento do número de bits do segmento de soma e desta forma, os segmentos de soma em um CLA geralmente são limitados a 4 bits.

Para cada par de bits (A_i, B_i) dos operandos, o CLA computa dois sinais: propaga (P_i) e gera (G_i). Esses sinais são utilizados para calcular em paralelo o *carry* de saída do respectivo par de bits (C_{i+1}). Cada bit da soma é calculado por meio de um OU exclusivo (XOR) entre C_i (C_{in}, no caso de S_0) e P_i, conforme mostra a Figura 4. Para acelerar ainda mais a propagação do *carry*, os blocos são conectados por um circuito em formato de árvore binária, conforme ilustrado na Figura 5. A função assintótica do atraso crítico para a arquitetura CLA nesse formato é ***O****(logn)*. Os sinais G_b e P_b, calculados em cada bloco CLA, são usados para computar o CIN para os bits mais significativos em relação ao bloco gerador desses sinais.

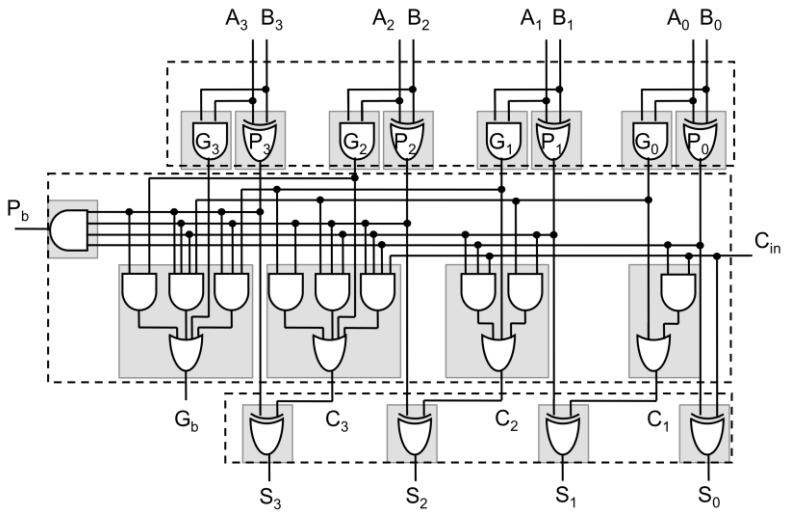

Figura 4. Bloco CLA de 4 bits

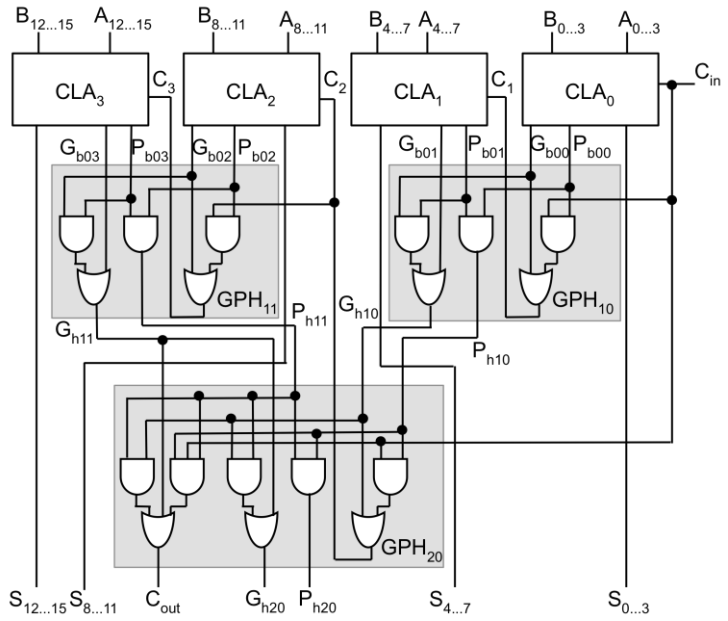

Figura 5. CLA de 16 bits com cadeia de propagação em formato de árvore binária

3. Somador *Add-One Carry-Select* Hierárquico (A1CSAH)

O somador proposto neste trabalho originou-se da ideia de otimizar o somador A1CSA em termos de atraso, sem aumentar demasiadamente a quantidade de recursos de *hardware* necessária para sua implementação em um fluxo de síntese *standard cells*.

O somador A1CSA proposto por Chang e Hsiao [Chang e Hsiao, 1998] e o modificado por Mesquita et al. para síntese em FPGA (e re-batizado de "RIC") [Mesquita et al., 2007] utilizam um multiplexador para selecionar entre a soma com *carry* de entrada igual a 0 e o resultado oriundo do circuito A1. A partir de uma análise detalhada das equações destas versões de A1CSA, concluiu-se ser possível eliminar este multiplexador através da absorção da operação de seleção por parte do bloco A1, o que originou as seguintes equações:

$$S_0 = CIN \oplus R_0 \quad (1)$$

$$S_1 = (CIN.R_0) \oplus R_1 \quad (2)$$

$$S_2 = (CIN.R_0.R_1) \oplus R_2 \quad (3)$$

$$S_3 = (CIN.R_0.R_1.R_2) \oplus R_3 \quad (4)$$

A Figura 6 mostra a implementação com portas lógicas das Equações anteriores, resultando no bloco chamado de *Add-One Select* (A1S). Tal bloco realiza a adição de 1 somente quando necessário, de acordo com as Propriedades 1, 2 e 3.

Com o intuito de acelerar a propagação do *carry* no A1CSA, a cadeia de propagação linear (ver Figura 3) foi substituída por uma cadeia em formato de árvore binária (similar a do CLA da Figura 5), fazendo uso do bloco A1S (ao invés do A1). Finalmente, mas não menos importante, como bloco somador utilizaram-se CLAs de 4 bits (no lugar de CRAs de 4 bits), também com o intuito de acelerar o cálculo da soma. As modificações realizadas no A1CSA deram origem a uma arquitetura de somador inédita, mostrada na Figura 7 e batizada de *Add-One Carry-Select Adder* Hierárquico (A1CSAH).

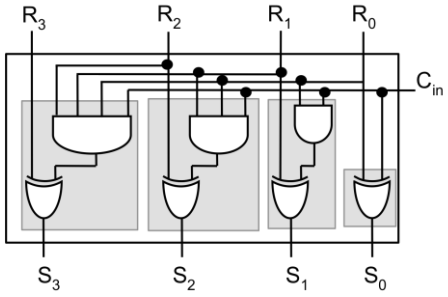

Figura 6. Bloco *Add-One Select* (A1S)

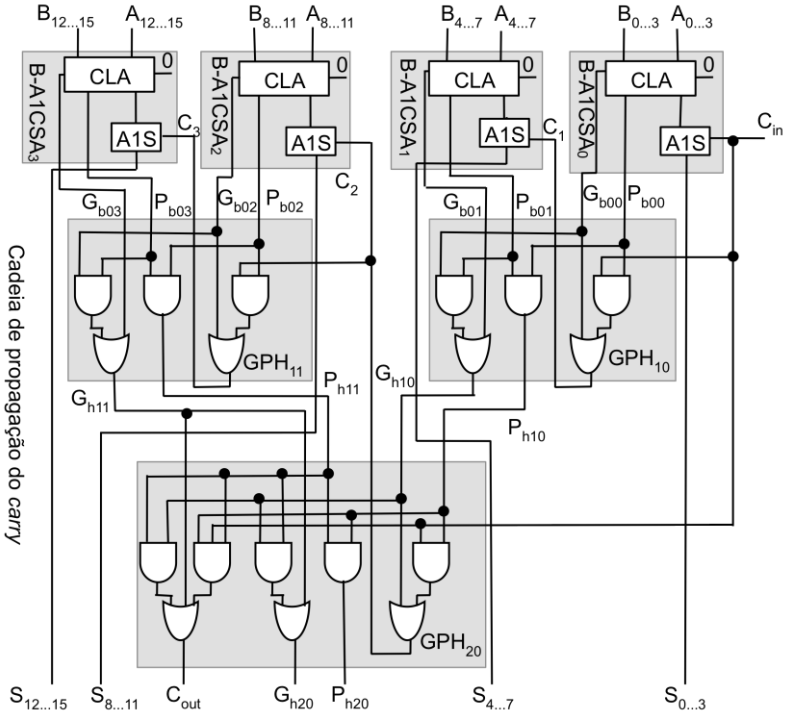

Figura 7. A1CSAH de 16 bits e sua cadeia hierárquica de propagação de *carry*

É interessante notar que o uso de CLAs para calcular as somas em cada segmento deve-se ao objetivo principal deste trabalho, qual seja, minimizar o atraso. Por outro lado, o A1CSA original [Chang e Hsiao, 1998] e sua versão para FPGA [Mesquita et al., 2007] adotaram CRAs porque visavam a minimização de recursos de *hardware*. No caso do A1CSAH, o uso do CLA traz como vantagem extra o fato dos sinais P_b e G_b (necessários para a seleção da soma correta em cadeias hierárquicas como as utilizadas no CLA e no A1CSAH) já estarem disponíveis. No caso de usar CRAs (com CIN=0) como unidade somadora básica, para cada CRA o P_b precisa ser calculado por meio de um "E" lógico entre todos os bits da soma (i.e., "E" lógico entre todos os bits da saída do CRA), ao passo que o G_b é o próprio COUT do CRA.

4. Síntese Lógica dos Somadores

Os somadores A1CSA, A1CSAH, CSA, CLA e CRA foram descritos em linguagem verilog para operandos de 8, 16, 32, 64, 128 e 256 bits, totalizando 30 versões de somadores. Os somadores CRA, CSA e A1CSA utilizaram CRAs de 4 bits como módulos básicos de adição, ao passo que os somadores CLA e A1CLAH utilizaram CLAs de 4 bits como módulos básicos de adição. A fim de facilitar a codificação em verilog, inicialmente foi criada uma descrição de CRA de 4 bits e uma descrição de CLA de 4 bits, para servirem como módulos básicos para a descrição dos somadores.

As 30 versões de somadores foram sintetizadas em nível lógico com o uso da ferramenta comercial *Design Compiler* da Synopsys [Synopsys, 2009] para a biblioteca *standard-cells* de 45nm da TSMC. O processo de síntese lógica foi controlado para evitar que as otimizações lógicas realizadas pelo *Design Compiler* descaracterizassem os somadores CSA, CLA, A1CSA e A1CSAH, uma vez que tais arquiteturas possuem redundância de *hardware*. Para tanto, foram desabilitadas a busca de funções equivalentes em módulos verilog diferentes e a eliminação de hierarquia entre os módulos verilog. Além disso, o código para as funções mais importantes do CLA de 4 bits foram separadas em módulos verilog distintos para evitar a remoção de lógica redundante, característica fundamental deste tipo de somador. (Na Figura 4, cada retângulo em cinza representa um módulo verilog diferente.)

O CRA foi mapeado para o FA disponível na biblioteca *standard-cells* da TSMC, uma vez que tal célula já está otimizada tanto para área quanto para desempenho. Para os demais somadores o *Design Compiler* mapeou as equações lógicas para o melhor conjunto de células disponível na biblioteca.

5. Resultados Experimentais

A Tabela 1 mostra a área estimada pelo *Design Compiler* para as arquiteturas de somadores (em μm^2). O CRA é o somador com a menor área para todos os tamanhos de operandos, o que era de se esperar, uma vez que ele é implementado a partir das equações lógicas na forma fatorada. Assim, a área do CRA foi utilizada como base para normalizar a área dos somadores rápidos, como mostrado na Tabela 2. O A1CSA é o menor somador rápido e requer, em média, 50% mais área do que o CRA. Por outro lado, o A1CSAH é o maior somador, uma vez que sua cadeia de *carry* hierárquica exige uma quantidade significativamente maior de recursos. Em média, o A1CSAH é 119%

maior do que o CRA. O CLA e o CSA são os somadores com área intermediária, sendo, em média, 93% e 103% maiores do que o CRA, respectivamente.

Tabela 1 – Área (μm²)

	8 bits	16 bits	32 bits	64 bits	128 bits	256 bits	Média
CRA	35,3	70,6	141,1	282,2	564,5	1.129,0	370,4
CSA	57,0	135,7	293,0	607,7	1.237,1	2.495,9	804,4
A1CSA	45,9	102,3	215,2	441,0	892,6	1.795,8	582,1
CLA	66,5	135,3	272,9	548,1	1.098,4	2.199,2	720,1
A1CSAH	75,7	153,6	309,6	621,5	1.245,2	2.492,7	816,4

Tabela 2 – Área normalizada em relação ao CRA

	8 bits	16 bits	32 bits	64 bits	128 bits	256 bits	Média
CSA	1,62	1,92	2,08	2,15	2,19	2,21	2,03
A1CSA	1,30	1,45	1,53	1,56	1,58	1,59	1,50
CLA	1,89	1,92	1,93	1,94	1,95	1,95	1,93
A1CSAH	2,15	2,18	2,19	2,20	2,21	2,21	2,19

A Tabela 3 mostra o atraso crítico dos somadores (em nanossegundos), conforme estimado pelo *Design Compiler*. Como era de se esperar, o CRA é o somador que possui o maior atraso crítico para todos os tamanhos de operandos. Os atrasos críticos dos demais somadores, normalizados em relação ao CRA, são apresentados na Tabela 4. O A1CSA é, em média, 53% mais rápido que o CRA. O CSA e o CLA são, em média, 54% e 62% mais rápidos do que o CRA, respectivamente. O A1CSAH é o somador mais rápido dentre todos, sendo inclusive mais rápido do que o CLA, o qual é reconhecido como uma das arquiteturas somadoras com o menor atraso crítico [Rabaey, Chandrakasan e Nikolic, 2003]. O A1CSAH é, em média, 67% e 10,8% mais rápido do que o CRA e o CLA, respectivamente.

Tabela 3 – Atraso Crítico (ns)

	8 bits	16 bits	32 bits	64 bits	128 bits	256 bits	Média
CRA	0,41	0,74	1,42	2,68	5,19	10,32	3,46
CSA	0,29	0,39	0,62	1,04	1,86	3,48	1,28
A1CSA	0,37	0,44	0,61	0,88	1,55	2,86	1,12
CLA	0,36	0,43	0,52	0,60	0,69	0,79	0,56
A1CSAH	0,31	0,38	0,45	0,54	0,63	0,71	0,50

Tabela 4 – Atraso Crítico normalizado em relação ao CRA

	8 bits	16 bits	32 bits	64 bits	128 bits	256 bits	Média
CSA	0,72	0,52	0,44	0,39	0,36	0,34	0,46
A1CSA	0,90	0,60	0,43	0,33	0,30	0,28	0,47
CLA	0,88	0,58	0,36	0,22	0,13	0,08	0,38
A1CSAH	0,77	0,52	0,32	0,20	0,12	0,07	0,33

A Tabela 5 apresenta o produto entre o atraso crítico e a potência, *Power-Delay Product* - PDP (em picoJoules). O PDP pode ser interpretado como a potência necessária para um dado somador realizar uma operação de adição, quando operando na sua máxima frequência. Desta forma, quanto menor o PDP, maior é a eficiência energética do somador. A Tabela 6 mostra o PDP normalizado em relação ao CRA. O

CSA é o somador com a pior eficiência energética, sendo, em média, 18% menos eficiente do que o CRA. O A1CSA e o CLA são, em média, 41% e 46% mais eficientes do que o CRA, respectivamente. O A1CSAH é o somador com maior eficiência energética. Em média, ele é 48% mais eficiente do que o CRA.

Tabela 5 – Eficiência energética (pJ)

	8 bits	16 bits	32 bits	64 bits	128 bits	256 bits	Média
CRA	0,01	0,91	3,47	13,14	50,85	202,33	45,12
CSA	0,01	0,85	2,91	10,07	36,75	138,65	31,54
A1CSA	0,01	0,67	1,91	5,64	19,95	73,80	17,00
CLA	0,01	0,80	1,93	4,48	10,34	23,72	6,88
A1CSAH	0,01	0,76	1,83	4,34	10,19	22,99	6,69

Tabela 6 – Eficiência energética normalizada em relação ao CRA

	8 bits	16 bits	32 bits	64 bits	128 bits	256 bits	Média
CSA	0,95	0,93	0,84	0,77	0,72	0,69	0,82
A1CSA	1,04	0,74	0,55	0,43	0,39	0,36	0,59
CLA	1,13	0,88	0,56	0,34	0,20	0,12	0,54
A1CSAH	1,11	0,84	0,53	0,33	0,20	0,11	0,52

6. Conclusões

Este artigo apresentou uma nova arquitetura de somador rápido, denominada *Add-One Carry-Select Adder* Hierárquico (A1CSAH). Este somador, juntamente com o CRA, CLA, CSA e o A1CSA, foram otimizados e sintetizados com a ferramenta *Design Compiler* para uma biblioteca *standard-cells* comercial de 45nm.

Os resultados da síntese lógica mostraram que o A1CSAH é o somador mais rápido para operandos a partir de 16 bits, sendo, em média, 10,8% mais rápido do que o CLA. Além disso, o A1CSAH tem praticamente a mesma eficiência energética do CLA. Assim, o A1CSAH é uma arquitetura apropriada para projetos de circuitos integrados nos quais há requisitos de operações aritméticas intensas e alta eficiência energética.

Como trabalho futuro vislumbra-se utilizar o A1CSAH na síntese de blocos de *hardware* que sejam intensivos em cálculos e que sejam muito utilizados em aplicações portáteis operadas a baterias, tais como processamento de sinais digitais e codificação de vídeo.

7. Referências Bibliográficas

Bedrij, O. J. (1962) “Carry-Select Adder”, In: IRE Transactions on Electronic Computers, n. 11, p. 340–346.

Chang, T. Y. e Hsiao, M. J. (1998) “Carry-Select Adder Using Single Ripple-Carry Adder”, In: Electronics Letters, v. 34, n. 12, p. 2101–2103.

Hwang, K. (1979) “Computer Arithmetic: Principles, architecture, and design”, New York, John Wiley & Sons.

Mesquita, E. et al. (2007) “RIC Fast Adder and its SET-Tolerant Implementation in FPGAs”, In: IEEE International Conference on Field Programmable Logic and Applications, pp. 638–641.

Omitido1, Omitido2, Omitido3; Omitido para revisão às cegas.

Oklobdzija, V. G. (2001) “Speed VLSI Arithmetic Units: Adders and Multipliers”, In: Design of High-Performance Microprocessor Circuits, USA: IEEE Press.

Rabaey, J. M., Chandrakasan, A. e Nikolic, B. (2003) “Digital Integrated Circuits: A Design Perspective”, Second edition, USA: Prentice Hall.

Takagi, N. et al. (2007) "Adders" In: Chen, W.-K. (Ed.). The VLSI handbook. Second edition. USA: CRP Press - Taylor e Francis Group.

Vahid, Frank (2006). "Digital Design", Danvers, MA: John Wiley & Sons, Inc.

Weinberger, A. e Smith, J. L. (1958) “A Logic for High-Speed Addition”, In: National Bureau of Standards, v. 591, p. 47–56.

Synopsys, (2009) “Synopsys's Design Compiler User Guide”, Versão C-2009.06.