Arquitetura de Somador de Alto Desempenho Baseada no Recálculo Parcial com Carry Invertido

Guilherme Corrêa[1], Eduardo Mesquita[1], Helen Franck[1],
Luciano Agostini[1], José Luís Güntzel[2]

[1] Grupo de Arquiteturas e Circuitos Integrados – Departamento de Informática
Universidade Federal de Pelotas (UFPel) – Pelotas, RS – Brasil

[2] Laboratório de Automação do Projeto de Sistemas – Depto. de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brasil

{gcorrea_ifm, emesquita.ifm, hfranck.ifm, agostini}@ufpel.edu.br, guntzel@inf.ufsc.br

Abstract. Addition is an operation of primary importance for electronic systems because it can serve as basis for many other operations that may determine the whole system performance. Therefore, fast adder architectures are still a subject of interest for both Industry and Academia. However, in classical fast adders the modifications to speedup carry propagation at the architectural level result in significant resource overhead. This work presents a new fast adder architecture called RIC (Re-computing the Inverse Carry-in). The proposed adder and other classical fast adder architectures were synthesized and validated for Altera Stratix III FPGAs. The experimental results, concerning resources and critical delay, showed that the RIC adder has a much better ratio between hardware resources and number of additions per second than the other classical fast adder architectures analyzed.

Palavras-chave: somador rápido, arquitetura aritmética, propagação de *carry*.

1 Introdução

Circuitos aritméticos são encontrados em profusão na maioria dos sistemas eletrônicos atuais e, não raro, são os responsáveis pela limitação de desempenho destes sistemas. Os somadores são, sem dúvida, os circuitos aritméticos mais importantes, pois servem como base para realizar a maioria das demais operações aritméticas, tais como: subtração, multiplicação e divisão. Desta forma, o desempenho de qualquer processador aritmético é diretamente proporcional ao desempenho dos circuitos somadores que o compõe [1].

O somador mais intuitivo que se pode projetar é o *Ripple-Carry* (RCA) [2] [3] que se destaca pela grande simplicidade. Suas características servem como referencial para comparações com os demais tipos de somadores. Contudo, o RCA apresenta um desempenho pobre devido à propagação serializada do *carry*, conforme ilustra a Figura 1: o cálculo realizado por cada estágio de soma (somador completo ou *full adder*) depende do *carry* de saída do estágio anterior, resultando em significativo atraso de propagação. Como o atraso da cadeia de propagação do RCA é proporcional ao número de estágios de soma, esta arquitetura só se mostra útil para aplicações que

exijam desempenho quando operaram sobre números com poucos bits (tipicamente, até 8 bits). Por este motivo, arquiteturas de somadores rápidos têm sido desenvolvidas e seguem sendo foco de intenso interesse acadêmico e industrial.

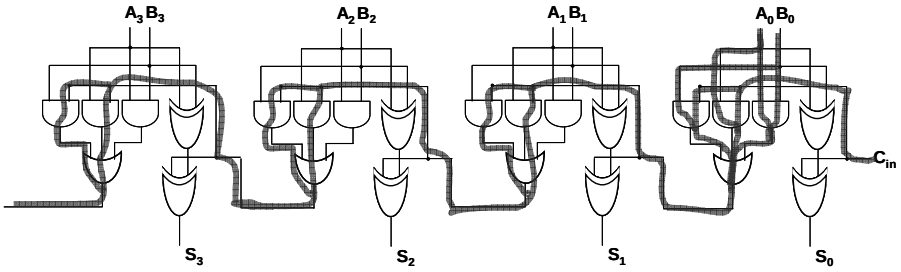

Figura 1. Prováveis caminhos críticos de um RCA de 4 bits

Com o objetivo de diminuir o atraso de propagação do *carry*, surgiram as diversas arquiteturas de somadores rápidos encontradas na literatura. Algumas destas exploram o estilo de implementação da lógica em tecnologia CMOS. Utilização de transistores de passagem e de transistores de pré-carga são técnicas de projeto adotadas nos somadores *Manchester Chain*, por exemplo [4]. Outros somadores se baseiam em modificações arquiteturais, como é o caso dos somadores *Carry Lookahead* [5], *Carry-Skip* [6] e *Carry-Select* [7]. Contudo, apesar das soluções baseadas em modificações arquiteturais se mostrarem bastante eficientes em termos de aceleração do cálculo, há um acréscimo significativo na quantidade de recursos utilizados, visto que essas arquiteturas, normalmente, se baseiam em utilização massiva de *hardware* paralelo.

Este trabalho apresenta uma nova arquitetura de somador rápido chamado RIC (*Re-computing the Inverse Carry-in* – Recálculo com *Carry-in* Invertido), a qual é inspirada na filosofia do somador *Carry-Select*. A arquitetura do somador RIC explora algumas propriedades básicas da adição binária que permitem recalcular a soma com o valor de *carry-in* invertido. Desta forma, é possível evitar a duplicação de estágios somadores, como ocorre no caso do somador *Carry-Select*. O somador RIC e os somadores *Ripple-Carry*, *Carry-Lookahead* e *Carry-Select* foram sintetizados para FPGAs da família Stratix III da Altera [8]. Os resultados de utilização de *hardware*, atraso crítico e dissipação de potência obtidos na síntese são comparados.

2 Arquiteturas de Somadores Rápidos

2.1 Somador *Carry-Lookahead*

Para acelerar a cadeia de propagação do *carry*, o somador *Carry-Lookahead* (CLA) realiza uma consulta a todos os estágios de entrada, gerando, simultaneamente, os *carries* referentes a cada um destes estágios [5]. Desta forma, todos os *carries* são calculados ao mesmo tempo, de forma paralela.

Cada *carry* gerado é aplicado ao estágio posterior ao que foi gerado, produzindo, assim, o resultado final da operação de soma referente àquele estágio. Dois sinais

auxiliares são utilizados para possibilitar a geração simultânea dos *carries*: *Carry Generate* e *Carry Propagate*, definidos pelas seguintes funções:

$$g_i = A_i \cdot B_i \quad (1)$$

$$p_i = A_i \oplus B_i \quad (2)$$

onde A_i e B_i são os operandos de entrada do i-ésimo estágio do somador *Carry Lookahead*.

Quando avaliado como verdadeiro, o sinal *Carry Generate* indica que um sinal de *carry* foi gerado no i-ésimo estágio, o que significa que o *carry* de saída do estágio anterior não precisará ser considerado. O sinal *Carry Propagate*, quando verdadeiro, indica que o i-ésimo estágio propaga o seu *carry* de entrada (C_{i-1}), independente do seu valor. O cálculo do *carry* no estágio i pode ser, portanto, definido pela seguinte função:

$$C_i = g_i + p_i \cdot C_{i-1} \quad (3)$$

Para que todos os *carries* sejam calculados em paralelo, as referências ao *carry* do estágio anterior da equação (3) precisam ser eliminadas. Assim, o conjunto de equações de um somador CLA de 3 bits, por exemplo, seria o seguinte:

$$C_0 = g_0 + p_0 \cdot C_{in} \quad (4)$$

$$C_1 = g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot C_{in} \quad (5)$$

$$C_2 = g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot C_{in} \quad (6)$$

A equação (7) define, de forma genérica, a geração do i-ésimo *carry* do CLA.

$$C_i = g_{i-1} + p_{i-1} \cdot g_{i-2} + ... + p_{i-1} \cdot p_{i-2} \cdot ... \cdot p_0 \cdot C_{in} \quad (7)$$

Como pode-se perceber pelas equações (4-7) a realização em *hardware* do CLA utiliza uma maior quantidade de recursos que a implementação de um RCA. Além disso, as equações (4-7) mostram que o grau de complexidade do cálculo do *carry* no CLA cresce muito rapidamente na medida que o número de bits dos operandos aumenta. Para contornar este problema, utilizam-se níveis hierárquicos, dividindo-se a adição em seções de 4 ou, no máximo, 8 bits, o que auxilia a manter sob controle a complexidade do circuito de geração dos *carries*.

2.2 Somador *Carry-Select*

Os somadores *Carry-Select* (CSA) [7] constituem-se em outra arquitetura de somadores projetados visando a aceleração da propagação do *carry*. Porém, o CSA se baseia em um acréscimo de recursos de mais de 100% em relação ao RCA, causando um incremento equivalente no consumo de energia. Desta forma, os CSAs são evitados sempre que houver restrições sérias em termos de custo da realização física e/ou consumo de energia.

No CSA, o cálculo da soma é dividido em *n* seções de *m* bits, de forma que cada seção possa ser calculada em paralelo, de forma independente. Quando todas as seções completam os seus cálculos, os resultados são reunidos para compor o resultado completo da soma. Para viabilizar o cálculo paralelo das seções, cada seção do CSA é composta por dois somadores RCA, sendo que um destes realiza a adição com *carry* de entrada igual a “0” e o outro realiza a adição com *carry* de entrada igual a “1”. Assim, ao contrário do que acontece no somador RCA, uma seção não precisa

esperar o *carry* de saída da seção anterior, pois o resultado é gerado em paralelo para ambas possibilidades de *carry* de entrada.

Além da duplicação existente em cada seção, cada seção utiliza um multiplexador para a seleção do resultado correto. A lógica de seleção do *carry* de entrada de cada bloco é baseada no *carry* de saída do bloco anterior. A seguir são apresentadas as equações utilizadas pelos circuitos lógicos que geram o sinal seletor do multiplexador de cada seção de um CSA de 16 bits.

$$Sel_1 = C_{in} \quad (8)$$

$$Sel_2 = C_{03} + (C_{13} \cdot C_{in}) \quad (9)$$

$$Sel_3 = C_{07} + (C_{17} \cdot (C_{03} + (C_{13}\, C_{in}))) \quad (10)$$

$$Sel_4 = C_{011} + (C_{111} \cdot (C_{07} + (C_{17} \cdot (C_{03} + (C_{13} \cdot C_{in}))))) \quad (11)$$

Com relação às equações (8-11), C_{0x} corresponde ao *carry* de saída do RCA de 4 bits que realiza a adição ($A_{x\text{-}3...x} + B_{x\text{-}3...x}$), a qual tem *carry* de entrada igual a "0", enquanto que o sinal C_{1x} corresponde ao *carry* de saída do RCA que realiza esta mesma adição, porém com *carry* de entrada igual a "1". Os demais sinais seguem este mesmo raciocínio.

É fácil notar que a complexidade desta lógica cresce rapidamente quando mais seções são adicionadas para se construir CLAs com maior número de bits. Desta forma, pode-se especular que, a partir de um certo número de bits dos operandos, o caminho crítico passa por um dos somadores RCA da primeira seção, segue pela cadeia de portas lógicas que realizam o sinal de seleção do multiplexador da última seção e termina na saída deste mesmo multiplexador. A Figura 2 ilustra o caminho crítico provável de um CSA de 32 bits.

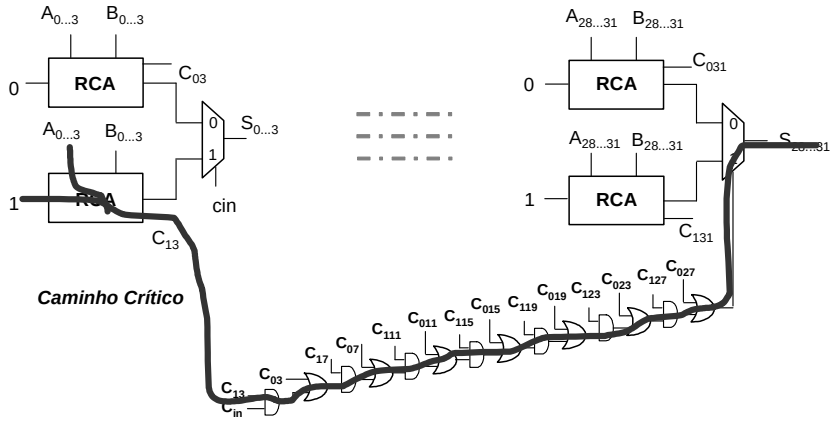

Figura 2. Caminho crítico de um CSA de 32 bits.

2.3 *Re-computing the Inverse Carry-in*

O somador *Re-computing the Inverse Carry-in* (RIC), foco deste trabalho, é uma nova arquitetura de somadores rápidos semelhante aos somadores CSA, mas que possui características únicas que lhe conferem o status de "somador inédito".

Ao contrário do CSA, o somador RIC não faz uso de dois blocos RCA em cada seção de adição. Ao invés disso, cada seção de adição do RIC é composta por um somador RCA, um bloco denominado *Re-computing Block* (RB) e um multiplexador. O RB substitui um RCA, porém a um custo bastante inferior ao do RCA. A Figura 3 apresenta a arquitetura de um somador RIC de 4 bits. O RB é ligado em série com a saída do RCA que recebe *carry* de entrada igual a "0". Esta estrutura implica em um pequeno acréscimo de atraso em cada seção do somador, em comparação ao atraso de uma seção de um CSA. Chamando de "R" o resultado gerado pelo CSA que possui *carry* de entrada igual a "0", pode-se afirmar que o objetivo do bloco RB é calcular "R+1" com um custo menor que o de um RCA. Com efeito, tal objetivo é atingido, pois o projeto do circuito RB faz uso das seguintes propriedades da adição binária [9].

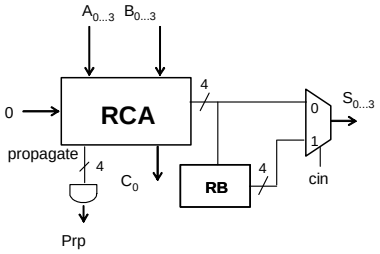

Figura 3. Somador RIC de 4 bits.

Propriedade 1: Dados dois números a serem somados, se ambos os números possuírem o bit menos significativo igual a "0", então o resultado da soma destes números com *carry* de entrada igual a "1" é idêntico ao resultado da soma com *carry* de entrada igual a "0", exceto pela troca do valor do bit menos significativo do resultado. A Figura 4 apresenta um exemplo da aplicação desta propriedade.

Cin	0		Cin	1	
A	1 1 0 0	(12)	A	1 1 0 0	(12)
B	0 1 0 0	(4)	B	0 1 0 0	(4)
Output	1 0 0 0 **0**	(16)	Output	1 0 0 0 **1**	(17)

Figura 4. Exemplo de soma utilizando a Propriedade 1 da soma binária.

Propriedade 2: Se o resultado de uma adição entre dois números com *carry* de entrada igual a "0" apresenta o bit menos significativo igual a "1", o resultado da adição destes mesmos dois números, mas com *carry* de entrada igual a "1", é gerado através da inversão dos *m* bits do resultado, da direita para a esquerda, até que seja encontrado o primeiro "0". Este último também deve ser invertido. A Figura 5 apresenta um exemplo da aplicação desta propriedade.

Cin	0		Cin	1	
A	0 0 0 1	(1)	A	0 0 0 1	(1)
B	0 1 1 0	(6)	B	0 1 1 0	(6)
Output	0 **1 1 1**	(7)	Output	**1 0 0 0**	(8)

Figura 5. Exemplo de soma utilizando a Propriedade 2 da soma binária.

Com base nas propriedades 1 e 2 é possível derivar um circuito mínimo para o RB, o qual ainda é submetido a portas XOR, que funcionam como inversores controlados.

Ao comparar-se a Figura 1 com a Figura 6, percebe-se que o bloco RB de 4 bits possui complexidade significativamente menor que um bloco RCA de 4 bits.

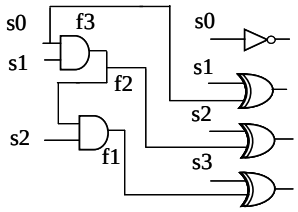

Figura 6. Bloco RB de um somador RIC de 4 bits.

Assim como o CSA, o somador RIC também precisa escolher entre o valor correto da adição entre o resultado gerado pelo RCA e aquele gerado pelo RB. Contudo, ao contrário do CSA, o RIC não possui dois *carries* de saída em cada seção, pois este utiliza apenas um somador por seção. Desta forma, as equações lógicas utilizadas para escolha dos resultados em cada uma das seções devem levar em consideração o "E" lógico (AND) entre os sinais *propagate* do RCA, os quais não requerem hardware extra, uma vez que são utilizados na construção do próprio RCA.

As equações (12-15) geram os sinais que são aplicados aos multiplexadores em cada seção de um somador RIC. O sinal C_0 corresponde ao *carry* de saída de um RCA de 4 bits que realiza a adição ($A_{0...3} + B_{0...3}$) com *carry* de entrada igual a '0'. Já o sinal P_0 corresponde ao "E" lógico aplicado aos sinais de *propagate* deste mesmo RCA ($(A_0 \oplus B_0) \cdot (A_1 \oplus B_1) \cdot (A_2 \oplus B_2) \cdot (A_3 \oplus B_3)$). Os outros sinais das equações podem ser deduzidos seguindo este mesmo raciocínio.

$$Sel_0 = C_{in} \quad (12)$$

$$Sel_1 = C_0 + (P_0 \cdot C_{in}) \quad (13)$$

$$Sel_2 = C_1 + (P_1 \cdot (C_0 + (P_0 \cdot C_{in}))) \quad (14)$$

$$Sel_3 = C_2 + (P_2 \cdot (C_1 + (P_1 \cdot (C_0 + (P_0 \cdot C_{in}))))) \quad (15)$$

O atraso crítico do somador RIC pode ser determinado de maneira similar ao atraso crítico do CSA. As versões de 4 e 8 bits apresentam como atraso crítico o atraso de uma seção. Isto é, nestas versões, a soma dos atrasos do RCA, do RB e do multiplexador define o maior caminho de propagação. Já em versões a partir de 16 bits, o caminho crítico do RIC é determinado pela equação lógica de cada seção, também como no CSA. Neste caso, o atraso crítico é definido pela soma dos atrasos de um bloco RCA, da maior equação lógica de uma seção e do multiplexador. A Figura 7 destaca o caminho crítico de um RIC de 32 bits.

Ao se comparar os somadores CSA e RIC de 32 bits (Figuras 2 e 7, respectivamente), percebe-se que o caminho crítico do somador RIC possui uma porta lógica a menos que o caminho crítico do CSA. Assim, a partir de 16 bits, o desempenho do RIC passa a ser superior ao do CSA.

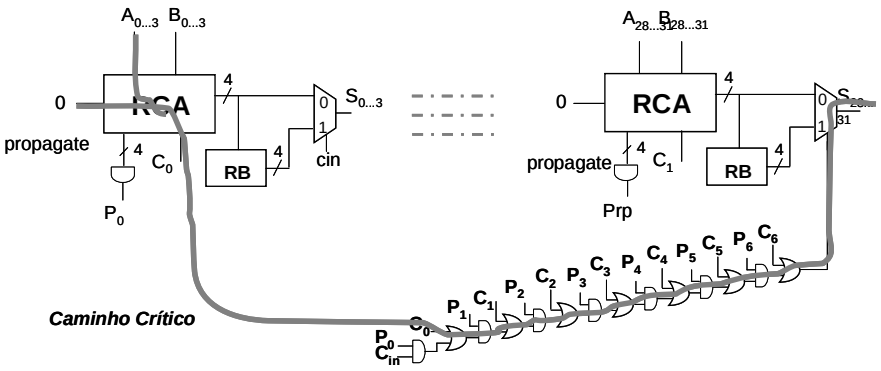

Figura 7. Caminho crítico de um somador RIC de 32 bits.

4 Experimentos e Resultados

A fim de comparar o somador RIC com as demais arquiteturas de somadores rápidos, foram descritos em VHDL, somadores RCA, CLA, CSA e o próprio RIC, com comprimentos de dados de 4, 8, 16, 32, 64 e 128 bits. Todos os somadores foram sintetizados para dispositivos FPGAs EP3SE50F484C2 da família Stratix III da Altera [8]. A síntese e a simulação foram realizadas através da ferramenta Quartus II, versão 7.2, também da Altera. Para avaliar a eficiência dos somadores RIC foram analisados três fatores, a saber: quantidade de recursos utilizados no FPGA, atraso crítico e potência dissipada pelo circuito.

A metade esquerda da Tabela 1 mostra o número de células lógicas (ALUTs) necessárias para a realização física de cada uma das versões dos somadores analisados. Com exceção da versão de 16 bits, o somador RIC mostrou os melhores resultados dentre todos os somadores rápidos avaliados. Como já era esperado, a arquitetura RCA utiliza menos recursos que as demais. A fim de possibilitar uma análise mais detalhada, a metade direita da Tabela 1 apresenta resultados comparativos que tomam o somador RCA como referência. Em todos os casos apresentados, os somadores CLA, CSA e RIC utilizaram mais recursos que o somador RCA. O somador RIC de 128 bits, por exemplo, utilizou cerca de 97% mais recursos que o somador RCA. Já o somador CSA da mesma configuração, apresentou um acréscimo de aproximadamente 138%.

Tabela 1. Número de ALUTs necessárias para a realização de somadores em FPGAs Stratix III da Altera e acréscimos aproximados em relação ao RCA.

Somador	Número de ALUTs utilizadas						Acréscimo em relação ao RCA (%)					
	4 bits	8 bits	16 bits	32 bits	64 bits	128 bits	4 bits	8 bits	16 bits	32 bits	64 bits	128 bits
RCA	33	62	120	236	468	932	-	-	-	-	-	-
CLA	63	120	230	468	924	1916	91	94	92	98	97	106
CSA	65	142	280	556	1110	2215	97	129	133	136	137	138
RIC	**55**	**118**	**232**	**460**	**918**	**1831**	**7**	**90**	**93**	**95**	**96**	**97**

O dado mais relevante da Tabela 1 é o fato do RIC exibir os melhores resultados médios de acréscimo de recursos em relação ao RCA. Somente em um caso (16 bits) o RIC é superado, mas a diferença não ultrapassa 1,6%. Em média, os acréscimos de recursos utilizados pelos somadores CLA, CSA e RIC em relação ao RCA foram, respectivamente, 96%, 128% e 90%. Deste modo, o RIC apresentou um resultado médio 6% melhor que o CLA e 38% melhor que o CSA neste quesito.

Em termos de atraso dos circuitos, como já era esperado, o RCA apresentou o pior desempenho. O CLA apresentou um resultado intermediário e os somadores CSA e RIC apresentaram desempenhos bastante semelhantes, sendo que o CSA foi mais rápido que o RIC apenas na versão de 16 bits. A metade esquerda da Tabela 2 mostra os dados obtidos através da ferramenta *TimeQuest Timing Analyzer*, da Altera. A metade direita desta tabela apresenta o aumento percentual aproximado do atraso dos somadores rápidos em relação ao RCA, para todos os tamanhos de operandos investigados. A partir da análise dos resultados pode-se perceber que todos os somadores rápidos mostraram-se mais lentos que o somador RCA em versões até 8 bits, conforme esperado. A partir de 16 bits, os somadores rápidos mostram um atraso crítico menor que o encontrado na arquitetura RCA e esta diferença aumenta à medida que aumenta o número de bits dos operandos.

Tabela 2. Atraso crítico (em ns) dos somadores sintetizados em FPGAs Stratix III da Altera e acréscimos de atraso aproximados em relação ao RCA.

Somador	**Atraso crítico (ns)**						**Acréscimo em relação ao RCA (%)**					
	4 bits	**8 bits**	**16 bits**	**32 bits**	**64 bits**	**128 bits**	**4 bits**	**8 bits**	**16 bits**	**32 bits**	**64 bits**	**128 bits**
RCA	1,61	2,68	5,12	9,66	18,63	36,97	-	-	-	-	-	-
CLA	2,75	3,96	4,14	4,94	5,84	7,05	71	48	-19	-48	-69	-81
CSA	2,56	2,94	3,49	4,57	4,86	7,36	59	10	-32	-53	-74	-80
RIC	**2,39**	**2,93**	**3,63**	**4,05**	**4,17**	**6,73**	**48**	**9**	**-29**	**-58**	**-78**	**-82**

Além de apresentar, a partir de 16 bits, atrasos significativamente menores que os verificados no RCA, o RIC também se mostra mais rápido que as outras duas arquiteturas avaliadas. O RIC de 32 bits, por exemplo, apresenta um atraso crítico cerca de 58% menor que o atraso crítico do RCA. Enquanto isso, o CLA e o CSA de mesmo comprimento de operando apresentam, respectivamente, atrasos 48,8% e 52,7% menores que o RCA.

A Tabela 3 apresenta o desempenho dos somadores expresso em milhões de cálculos por segundo. O elevado desempenho dos somadores rápidos fica evidenciado a partir de arquiteturas com mais de 8 bits. Considerando o extremo da tabela, ou seja, somadores com operandos de 128 bits, é possível perceber que o RIC é capaz de realizar quase 149 milhões de adições por segundo. Isso implica em cerca de 121 milhões de adições por segundo a mais que o RCA, o que representa uma taxa de processamento 449,7% superior à atingida pelo RCA para operandos com este número de bits.

Entre os somadores rápidos, o RIC novamente apresenta o melhor desempenho, atingindo 6,8 milhões de adições por segundo a mais que o CLA e 12,8 milhões de adições por segundo a mais que o CSA, resultando um ganho de 4,78% e 9,38%, respectivamente, nas taxas de processamento destes operadores.

Tabela 3. Desempenho das arquiteturas de somadores mapeadas em FPGAs Stratix III da Altera, expresso em milhões de adições por segundo.

Somador	Milhões de adições por segundo					
	4 bits	8 bits	16 bits	32 bits	64 bits	128 bits
RCA	620,347	372,995	195,465	103,563	53,674	27,052
CLA	363,769	252,143	241,663	202,347	171,174	141,904
CSA	390,625	339,789	286,779	218,914	205,973	135,943
RIC	**418,235**	**341,414**	**275,558**	**246,853**	**239,693**	**148,699**

Unindo as comparações apresentadas, é possível construir uma comparação cruzada, relacionando a quantidade de recursos de *hardware* investidos por cada arquitetura de somador rápido para cada milhão de adições por segundo atingido. O gráfico apresentado na Figura 8 demonstra esta relação. Nesta comparação cruzada fica ainda mais evidente a qualidade do somador RIC, pois ele apresenta sempre a melhor relação para todos os somadores rápidos investigados. Assim, é o RIC que gasta menos *hardware* para cada milhão de adições por segundo atingidos dentre os somadores rápidos investigados, demonstrando a eficiência desta nova arquitetura.

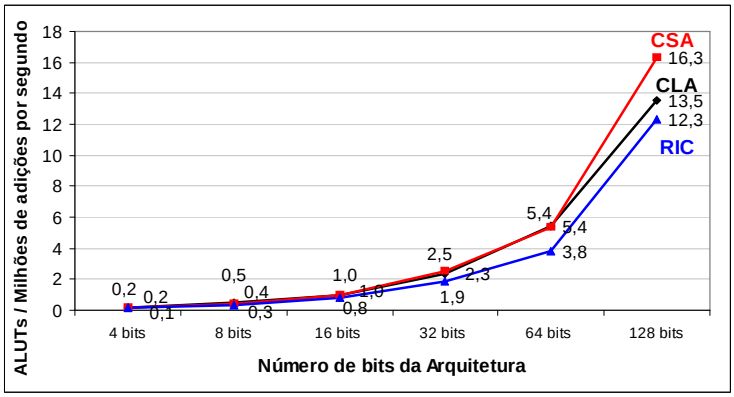

Figura 8. Relação entre *hardware* utilizado e número de adições por segundo para os somadores rápidos investigados.

Em média, os somadores RIC mostraram um desempenho 22% melhor que os CSAs. Com relação ao somador CLA, o RIC apresentou, em média, resultados 20,04% melhores. Os melhores casos de ganhos do RIC relativos às arquiteturas CSA e CLA foram de, respectivamente, 28,93% e 29,05% para somadores de 64 bits.

Para a estimação de potência dissipada foi utilizada a ferramenta *PowerPlay Power Analyzer*, da Altera. Embora os dados não permitam definir qual das arquiteturas mostrou os melhores resultados, percebe-se que o somador RIC não apresenta baixa eficiência em termos de dissipação de potência. Em arquiteturas de 8 bits, por exemplo, o RIC apresentou-se como o somador rápido que mais dissipa potência (1,5% a mais que o RCA). Por outro lado, considerando-se arquiteturas de 64 bits, o RIC apresenta os melhores resultados (2,3% a mais que o RCA, contra o acréscimo de 28,5% gerado pelo CSA). Nos outros quatro casos, o RIC mostrou ser uma alternativa intermediária em termos de dissipação de potência, o que é bastante aceitável quando se levam em consideração todos os outros resultados obtidos pelo projeto deste somador.

5 Conclusões e Trabalhos Futuros

Este artigo apresentou o somador RIC, uma nova arquitetura de somador rápido que, como o somador *Carry-Select*, realiza a adição em seções que operam de maneira paralela, a fim de acelerar a propagação do *carry*. Os dados de síntese, análise de *timing* e estimação de potência dissipada obtidos para FPGAs Stratix III mostraram que o somador RIC mostrou-se mais eficiente que os demais somadores rápidos na maioria dos casos.

Em uma comparação com o somador RCA, o somador RIC apresentou um acréscimo médio de 90% em uso de recursos, enquanto que o CSA e o CLA mostraram acréscimo médio de 128% e 96%, respectivamente. O CLA, contudo, apresentou os piores resultados em termos de atraso crítico, enquanto que o somador RIC figura novamente como a melhor arquitetura neste quesito.

Na relação entre utilização de recursos de *hardware* e número de adições por segundo atingido por cada arquitetura, o RIC ganha sempre de todos os somadores rápidos investigados, apresentando, no melhor caso, uma relação até 29,05% melhor que os demais somadores rápidos.

Em termos de dissipação de potência, os resultados, apesar de não apresentarem uma tendência favorável a nenhuma das arquiteturas analisadas, foram bastante aceitáveis para o somador RIC, que apresentou os resultados mais próximos àqueles apresentados pelo somador RCA. Assim, haja vista todas as melhorias proporcionadas pelo somador RIC nos demais critérios, pode-se concluir que os resultados de dissipação de potência deste somador são bons.

Em trabalhos futuros, pretende-se realizar análise de área (número de transistores), de atraso e de potência do somador RIC (e dos demais somadores investigados neste trabalho) a partir da geração de leiautes e extração dos mesmos para simulações no nível elétrico.

Referências

1. Oklobdzija, V: High-Speed VLSI Arithmetic Units: Adders and Multipliers. In: Design of High-Performance Microprocessor Circuits, IEEE Press, p. 181-204 (2001).
2. Hwang, K: Computer Arithmetic: Principles, Architecture, and Design. New York: Wiley, 423 p. (1979).
3. Ercegovac, M. D. e Lang, T.: Digital Arithmetic, Elsevier Science, San Francisco, USA, p. 59 (2004).
4. Kilburn, T et al.: Parallel Addition in Digital Computers: A New Fast ‘Carry’ Circuit, Proceedings of IEE, Vol. 106, pt. B, p. 464 (1959).
5. Weinberger, A. and Smith, J.L.: A Logic for High-Speed Addition, In: National Bureau of Standards, Circulation 591, p. 3-12 (1958).
6. Lehman, M. e Burla, N.: Skip techniques for high-speed carry-propagation in binary arithmetic units. IRE Transactions on Electronic Computers, vol. 10, pp. 691–698 (1961).
7. Bedrij, O. J.: Carry-Select Adder, In: IRE Transactions on Electronic Computers, p. 340 (1962).
8. Altera Corporation. http://www.altera.com
9. Kumar, K. e Lala, P.: On-line Detection of Faults in Carry-Select Adders. In: International Test Conference, pp. 912, (2003).