CONVERSOR DE ESPAÇO DE CORES PARALELO PARA A COMPRESSÃO DE IMAGENS JPEG

Luciano Volcan Agostini *agostini@inf.ufrgs.br*
Ivan Saraiva Silva* *ivan@dimap.ufrn.br*
Sergio Bampi** *bampi@inf.ufrgs.br*
Grupo de Arquiteturas e Circuitos Integrados
Universidade Federal de Pelotas – DMEC – Pelotas – Rio Grande do Sul – Brasil
Campus Universitário, s/nº - Caixa Postal 354 - CEP 96010-900
** Universidade Federal do Rio Grande do Norte – DIMAp – Natal – Brasil*
*** Universidade Federal do Rio Grande do Sul – II / GME – Porto Alegre – Brasil*

RESUMO

Este artigo propõe e desenvolve arquiteturas paralelas e com *pipeline*, para realizar a primeira etapa da compressão JPEG: a conversão do espaço de cores de RGB para YCbCr. O desenvolvimento das arquiteturas paralelas utilizou como base uma arquitetura não paralela desenvolvida em trabalhos anteriores. O paralelismo possibilitou um ganho significativo em termos de taxa de processamento de pixels, com um impacto muito pequeno na utilização de recursos. As arquiteturas propostas foram descritas em VHDL e sintetizadas para FPGAs da família Flex 10KE da Altera, atingindo freqüências próximas a 40MHz, o que permitiu uma taxa de processamento superior a 10 milhões de pixels por segundo. Estas arquiteturas são capazes de converter o espaço de cores de mais de 40 imagens coloridas de 640 x 480 pixels a cada segundo.

ABSTRACT

This paper proposes the design of parallel and pipelined architectures to the first step of the JPEG compression: the color space conversion from RGB to YCbCr. The design of these parallel architectures was made using a non parallel architecture designed in previous works. The parallelism makes possible a great gain in terms of pixels processing rate, with a very small impact in the use of resources. The proposed architectures were described in VHDL and synthesized for Altera Flex 10KE family FPGAs, reaching frequencies near to 40MHz. The parallel architectures reach a processing rate higher than 10 million pixels per second. These architectures are able to convert the color space of more than 40 color images of 640 x 480 pixels at each second.

CONVERSOR DE ESPAÇO DE CORES PARALELO PARA A COMPRESSÃO DE IMAGENS JPEG

Luciano Volcan Agostini *agostini@inf.ufrgs.br*
Ivan Saraiva Silva* *ivan@dimap.ufrn.br*
Sergio Bampi** *bampi@inf.ufrgs.br*
Grupo de Arquiteturas e Circuitos Integrados
Universidade Federal de Pelotas – DMEC – Pelotas – Rio Grande do Sul – Brasil
Campus Universitário, s/nº - Caixa Postal 354 - CEP 96010-900
** Universidade Federal do Rio Grande do Norte – DIMAp – Natal – Brasil*
*** Universidade Federal do Rio Grande do Sul – II / GME – Porto Alegre – Brasil*

RESUMO

Este artigo propõe e desenvolve arquiteturas paralelas e com *pipeline*, para realizar a primeira etapa da compressão JPEG: a conversão do espaço de cores de RGB para YCbCr. O desenvolvimento das arquiteturas paralelas utilizou como base uma arquitetura não paralela desenvolvida em trabalhos anteriores. O paralelismo possibilitou um ganho significativo em termos de taxa de processamento de pixels, com um impacto muito pequeno na utilização de recursos. As arquiteturas propostas foram descritas em VHDL e sintetizadas para FPGAs da família Flex 10KE da Altera, atingindo freqüências próximas a 40MHz, o que permitiu uma taxa de processamento superior a 10 milhões de pixels por segundo. Esta taxa de processamento possibilita, por exemplo, o processamento de mais de 40 imagens coloridas de 640 x 480 pixels a cada segundo.

1. INTRODUÇÃO

O padrão de compressão de imagens JPEG foi proposto originalmente em 1987 pelo *Join Photographic Expert Group* [1]. Rapidamente este padrão passou a ser o mais utilizado para imagens fotográficas, mantendo este status até a atualidade, porque possibilita excelentes taxas de compressão, com pequeno impacto na qualidade das imagens comprimidas.

A compressão JPEG possui cinco operações básicas, conforme a fig. 1. Estas operações são: conversão do espaço de cores, downsampling, DCT 2-D, quantização e codificação de entropia [2]. Este artigo irá focar no desenvolvimento de arquiteturas paralelas para a primeira etapa da compressão JPEG, ou seja, a conversão de espaço de cores.

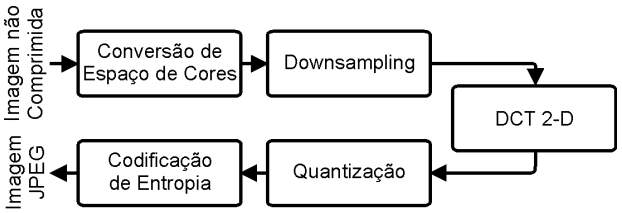

Figura 1 – Operações da compressão JPEG para imagens coloridas

Esse artigo propõe e desenvolve duas arquiteturas paralelas e com *pipeline* para realizar a conversão do espaço de cores de RGB para YCbCr, de acordo com o padrão JPEG. As duas arquiteturas desenvolvidas diferem apenas em algumas simplificações que foram realizadas, como ficará mais claro no decorrer deste artigo.

Estas arquiteturas foram desenvolvidas considerando como entradas valores de pixels no espaço de cores RGB, que é o mais utilizado em imagens digitais. Estes pixels possuem 8 bits para cada um de seus componentes de cor, em um total de 24bits. O conversor de espaço de cores transforma a informação de cor do espaço RGB para o espaço YCbCr e deve manter a representação de cada um dos componentes do novo espaço (Y, Cb e Cr) com a mesma quantidade de bits utilizados para representar cada um dos componentes do espaço de entrada (R, G e B).

Os valores de R, G e B devem estar disponíveis na entrada durante três ciclos de *clock* para que os cálculos referentes aos três elementos de saída (Y, Cb e Cr) sejam realizados.

Os componentes de saída Y, Cb e Cr são gerados de maneira intercalada e um novo componente Y, Cb ou Cr é gerado a cada ciclo de *clock* quando o *pipeline* está preenchido. Deste modo, as arquiteturas propostas possuem uma taxa de produção de um pixel (conjunto de três componentes de cor) no espaço YCbCr a cada três ciclos de *clock*.

O item dois deste artigo descreve a operação de conversão de espaço de cores de acordo com o padrão JPEG.

A seguir, no item três, é apresentada a arquitetura que embasou o desenvolvimento das arquiteturas propostas por este artigo.

O quarto item do artigo apresenta, em detalhes, as arquiteturas paralelas construídas para a conversão de espaço de cores.

O item cinco do artigo apresenta os resultados de síntese das arquiteturas desenvolvidas, comparando-os com os resultados obtidos para a arquitetura anteriormente desenvolvida.

Por fim, as conclusões são apresentadas no item seis do artigo.

2. A CONVERSÃO DE ESPAÇO DE CORES NA COMPRESSÃO JPEG

A conversão do espaço de cores é a primeira operação a ser realizada por um compressor JPEG, quando este recebe como entrada imagens no espaço de cores RGB. Os componentes R, G e B possuem um elevado grau de correlação, tornando difícil o processamento de cada uma das informações de cor de forma independente, por isso, a compressão JPEG tem uma eficiência muito maior para espaços de cores do tipo luminância e crominância do que para o espaço RGB [3]. Então, os pixels da imagem que estão no espaço de cores RGB, são convertidos para um espaço de cores do tipo luminância e crominância para a posterior compressão [4].

A conversão de espaço de cores só tem sentido quando a imagem que está sendo processada for uma imagem colorida, não se aplicando a imagens em tons de cinza.

Existem vários espaços de cores do tipo luminância e crominância. O espaço que será objeto das arquiteturas desenvolvidas neste artigo é chamado de YCbCr, onde o componente Y contém a informação de luminância da imagem, ou seja, contém as informações sobre os tons de cinza. Os componentes Cb e Cr contêm as informações de cores da imagem, sendo que o componente Cb contém a informação relativa à cor azul (*chrominance blue*) e o componente Cr contém a informação relativa à cor vermelha (*chrominance red*). Este espaço de cor é embasado no espaço YUV, que é utilizado no padrão europeu de televisão PAL, onde os componentes Cb e Cr são escalas deslocadas dos componentes U e V [5].

Os cálculos realizados na conversão do espaço de cores de RGB para YCbCr estão apresentados abaixo. São consideradas parcelas de cada um dos três componentes de cor dos *pixels* de entrada no cálculo dos componentes de cor dos *pixels* no espaço YCbCr [6].

$$Y_{i,j} = 0{,}299R_{i,j} + 0{,}587G_{i,j} + 0{,}114B_{i,j}$$
$$Cb_{i,j} = -0{,}169R_{i,j} - 0{,}331G_{i,j} + 0{,}5B_{i,j}$$
$$Cr_{i,j} = 0{,}5R_{i,j} - 0{,}419G_{i,j} - 0{,}081B_{i,j}$$

As arquiteturas desenvolvidas neste artigo são implementações que realizam, de maneira paralela e com técnicas de *pipeline*, os cálculos apresentados acima.

3. A ARQUITETURA ORIGINAL

A arquitetura do conversor do espaço de cores que serviu de base para as arquiteturas paralelas apresentadas neste artigo foi inicialmente proposta em [7] e aprimorada em [8]. Esta arquitetura será apresentada resumidamente neste item do artigo.

Os cálculos realizados pelo *datapath* do conversor de espaço de cores são, basicamente, multiplicações por constantes e somas, como já foi apresentado no item anterior.

A fig. 2 apresenta a arquitetura desenvolvida em [8]. Esta arquitetura foi projetada para operar em um *pipeline* de quatro estágios e possui uma latência de seis ciclos de *clock*. Um novo resultado válido é gerado a cada três ciclos de *clock* quando o *pipeline* está preenchido.

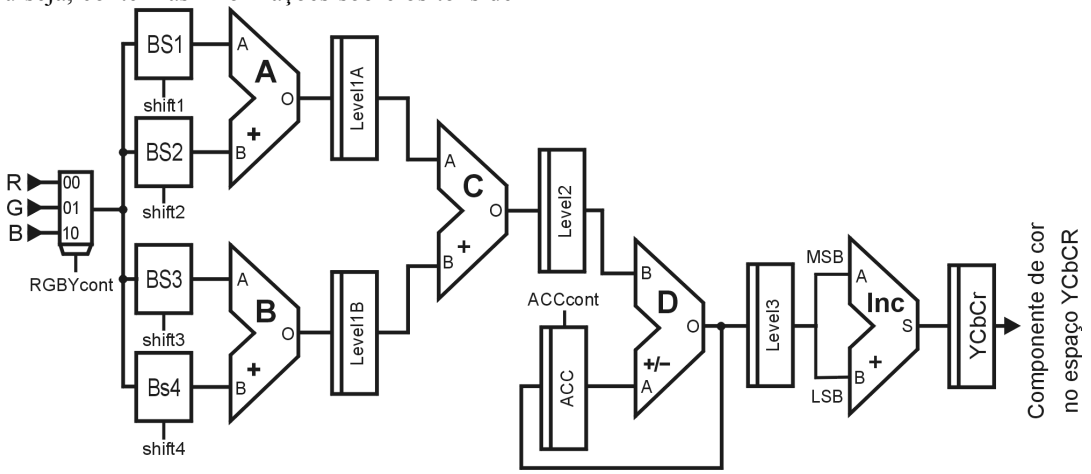

Figura 2 – Arquitetura original do conversor de espaço de cores

As somas foram desenvolvidas através do uso de somadores *ripple carry*, enquanto que as multiplicações foram desenvolvidas através do uso de somas de deslocamentos. Os deslocamentos foram obtidos através de *barrel shifters* otimizados e as somas, novamente com o uso de somadores *ripple carry*.

Cada multiplicação é obtida através da soma de quatro deslocamentos, que são realizadas pelos somadores **A**, **B** e **C** da fig. 2. O somador **D** faz a acumulação das multiplicações e pode realizar somas ou subtrações, de acordo com o componente de cor que está sendo calculado. O incrementador **Inc** arredonda o resultado da multiplicação, somando com a parte inteira do resultado, o bit mais significativo dentre os bits da parte fracionária.

Os *barrel shifters* foram simplificados, gerando o mínimo número possível de deslocamentos e eliminando os bits que são constantes para todos os deslocamentos gerados. Estas simplificações têm impacto também nos somadores e registradores utilizados na arquitetura, muito embora gerem números em diferentes escalas nas saídas dos *barrel shifters*.

Para somar os valores entregues por diferentes *barrel shifters* é necessário colocar os números na mesma escala e isto é feito através da concatenação com zeros.

As saídas dos *barrel shifters* foram truncadas na quarta casa fracionária como forma de minimizar os recursos utilizados pela arquitetura. O impacto deste truncamento é mínimo, uma vez que a arquitetura entrega números inteiros em sua saída e, então, mesmo as quatro casas fracionárias, consideradas nos cálculos internos, são descartadas na saída, que possuirá apenas oito bits.

Os componentes de cor Y, Cb e Cr calculados pela arquitetura são armazenados no registrador de saída **YCbCr**. Este registrador permite o sincronismo da saída da arquitetura do conversor de espaço de cores, com os outros estágios do *pipeline* do compressor JPEG.

Cada *pixel* de entrada, com seus componentes R, G e B, deve ficar disponível durante nove ciclos de *clock* para o cálculo dos três componentes do espaço de cores YCbCr.

A taxa de produção desta arquitetura é três vezes inferior à taxa de consumo dos demais componentes já desenvolvidos do compressor JPEG [8], uma vez que os demais blocos constituintes do compressor consomem um elemento de cor no espaço de cores YCbCr a cada ciclo de *clock*. Esta significativa diferença entre a taxa de produção do conversor de espaço de cores e a taxa de consumo dos demais blocos do compressor JPEG faz com que, em caso de integração, os demais blocos do compressor JPEG passem dois terços do tempo sem executar operação alguma. Esta constatação, somada ao fato de que a complexidade dos cálculos executados pelos demais blocos do compressor é várias vezes superior à complexidade do cálculo executado pelo conversor de espaço de cores, fazem com que todo este desperdício em tempo de processamento seja completamente inaceitável. Deste modo, surgiu a necessidade, já apontada em trabalhos anteriores [8] de gerar uma nova arquitetura de conversor de espaço de cores capaz de produzir um elemento de cor a cada ciclo de *clock*. Foi esta a grande motivação para o desenvolvimento das arquiteturas apresentadas neste artigo.

4. AS ARQUITETURAS PARALELAS

A solução para o problema da pequena taxa de produção da arquitetura original de conversão de espaço de cores foi encontrada através do uso de uma arquitetura onde os cálculos relativos a cada componente de cor são todos realizados em paralelo. Desta forma, é possível gerar um elemento no espaço de cores YCbCr em cada ciclo de *clock* com o *pipeline* preenchido. Esta taxa de produção é a exata taxa de consumo dos demais blocos do compressor JPEG, o que permite uma perfeita integração entre estas arquiteturas.

As arquiteturas paralelas desenvolvidas neste artigo são similares, onde uma foi desenvolvida de maneira genérica e a outra foi desenvolvida a partir de simplificações realizadas sobre a primeira, como ficará claro nos próximos parágrafos. O *datapath* das arquiteturas paralelas está apresentado na fig. 3.

As arquiteturas paralelas foram desenvolvidas em um *pipeline* de cinco estágios, um a mais que na arquitetura original. Este estágio extra foi necessário para permitir a totalização, em paralelo, dos resultados parciais dos cálculos de cada componente de cor. Mesmo com um estágio a mais no *pipeline*, a latência das arquiteturas paralelas é de cinco ciclos de *clock*, um a menos que na arquitetura original, onde a latência é de seis ciclos.

Os cálculos realizados pelas arquiteturas paralelas são os mesmos realizados pela arquitetura original, ou seja, envolvem somas ou subtrações e multiplicações por constantes.

Da mesma forma que na arquitetura original, as multiplicações por constantes foram desenvolvidas através da soma de quatro deslocamentos. Os deslocamentos associados a cada constante em cada *barrel shifter* estão apresentados na tab. 1, onde **i** é o dado de entrada e **[x]** indica o número de deslocamentos à direita realizado sobre o dado de entrada.

Nas arquiteturas paralelas para o conversor de espaço de cores, como na arquitetura original, as saídas dos *barrel shifters* são truncadas na quarta casa fracionária para minimizar o uso de recursos.

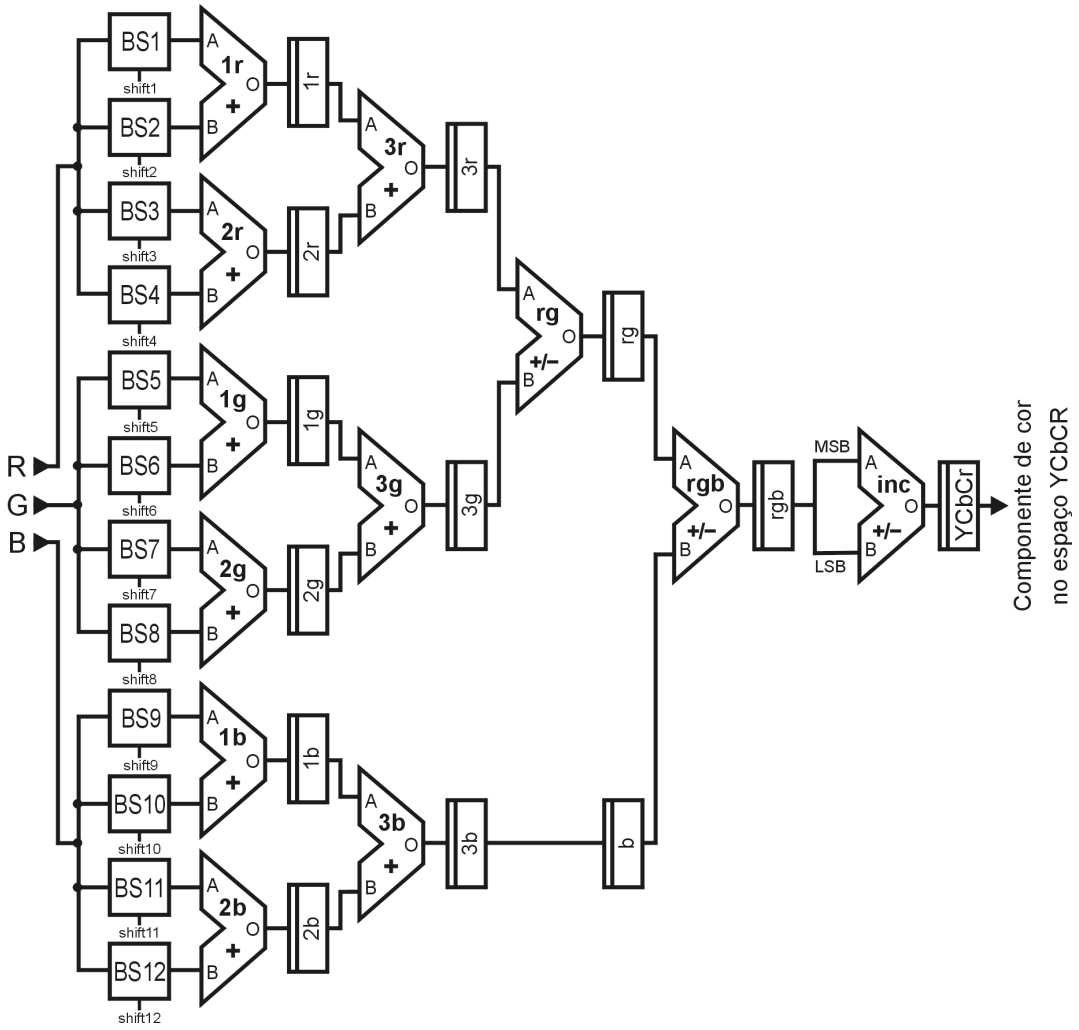

Figura 3 – *Datapath* do conversor de espaço de cores paralelo

Tabela 1 – Deslocamentos por *barrel shifter*

Constante	**BS1**	**BS2**	**BS3**	**BS4**
0,299	*i* [2]	*i* [5]	*i* [6]	*i* [8]
0,587	*i* [1]	*i* [4]	*i* [7]	*i* [6]
0,114	*i* [4]	*i* [5]	*i* [6]	*i* [8]
Constante	**BS5**	**BS6**	**BS7**	**BS8**
0,169	*i* [3]	*i* [5]	*i* [7]	*i* [8]
0,331	*i* [2]	*i* [4]	*i* [6]	*i* [9]
0,5	*i* [1]	-	-	-
Constante	**BS9**	**BS10**	**BS11**	**BS12**
0,5	*i* [1]	-	-	-
0,419	*i* [2]	*i* [5]	*i* [3]	*i* [6]
0,081	*i* [4]	*i* [6]	-	*i* [9]

A arquitetura paralela que convencionamos chamar de genérica tem este nome por conta dos *barrel shifters* que utilizou, que foram *barrel shifters* genéricos, capazes de realizar todos os deslocamentos necessários aos cálculos de qualquer das multiplicações por constantes. Esta arquitetura genérica possui a vantagem de ser facilmente instanciada e validada, por outro lado, tem a desvantagem de ocupar muitas células lógicas.

A arquitetura paralela simplificada minimiza o uso de recursos, por desenvolver cada um dos doze *barrel shifters* especificamente para as operações em que serão utilizados. Além disso, os *barrel shifters* não consideram, em suas saídas, aqueles bits que são constantes para todos os deslocamentos, o que permitiu o uso de somadores com menor número de bits, com nova minimização no uso de recursos.

Para somar os valores gerados pelos *barrel shifters* simplificados é preciso concatenar suas saídas com zeros,

para colocá-las todas na mesma escala. A concatenação com zeros pode ocorrer à esquerda ou à direita, de acordo com o *barrel shifter* considerado. Os números são concatenados com zeros porque a entrada possui apenas números positivos, por isso, não é necessário controlar a extensão do sinal, que é conhecido e tem o valor zero.

Os *barrel shifters* **BS1**, **BS2**, **BS3** e **BS3** e os somadores **1r**, **2r** e **3r** na fig. 3, são responsáveis pelo cálculo relativo ao componente de cor R, para os cálculos de Y, Cb e Cr, ou seja, realizam os cálculos:

- 3r = 0,299 x R para Y,
- 3r = 0,169 x R para Cb e
- 3r = 0,5 x R para Cr.

Os *barrel shifters* **BS5**, **BS6**, **BS7** e **BS8** e os somadores **1g**, **2g** e **3g** são responsáveis pelo cálculo relativo ao componente de cor G, realizando os cálculos:

- 3g = 0,587 x G para Y,
- 3g = 0,331 x G para Cb e
- 3g = 0,419 x G para Cr.

Da mesma forma, os *barrel shifters* **BS9**, **BS10**, **BS11** e **BS12** e os somadores **1b**, **2b** e **3b** são responsáveis pelo cálculo relativo ao componente de cor B, realizando os cálculos:

- 3b = 0,114 x B para Y,
- 3b = 0,5 x B para Cb e
- 3b = 0,081 x B para Cr.

O somador **rg** na fig. 3 realiza adições e subtrações e é responsável por gerar a totalização dos cálculos relativos aos componentes de cor R e G, para os cálculos de Y, Cb e Cr, realizando os cálculos:

- rg = 3r + 3g para Y,
- rg = 3r + 3g para Cb e
- rg = 3r – 3g para Cr.

O registrador **b** na fig. 3 é utilizado apenas para o correto funcionamento do *pipeline*. Este registrador recebe, com um ciclo de atraso, o resultado do somador **3b**.

O somador **rgb** realiza adições e subtrações, podendo considerar como negativa qualquer de suas entradas, de acordo com o cálculo a ser realizado. O somador **rgb** é responsável pela totalização do cálculo relativo aos componentes de cor R, G e B, para os cálculos de Y, Cb e Cr, através dos cálculos:

- rgb = rg + b para Y,
- rgb = – rg + b para Cb e
- rgb = rg – b para Cr.

Por fim, o somador **inc** funciona como incrementador que é usado para arredondar o resultado dos cálculos anteriores. O cálculo realizado por **inc** é a soma dos 8 bits da parte inteira com o bit da parte fracionária do resultado armazenado em **rgb**.

A tab. 2 apresenta uma comparação entre os *barrel shifters* utilizados na arquitetura genérica e os *barrel shifters* utilizados na arquitetura simplificada, onde são comparados os números de bits nas saídas e o número de deslocamentos realizados. A partir do número de deslocamentos é possível calcular o número de bits utilizados para o controle de cada um dos deslocadores, que passou de quatro bits na solução genérica para dois ou um bit na solução simplificada. Desta forma, com os dados da tab. 2 é possível perceber o impacto das simplificações na utilização de recursos pelos *barrel shifters*.

Tabela 2 – Número de bits na saída e número de deslocamentos de cada *barrel shifter* utilizado nas soluções paralela genérica e simplificada

Barrel	**Solução Genérica**		**Solução Simplificada**	
Shifter	**N° de bits**	**N° de desloc.**	**N° de bits**	**N° de desloc.**
BS1	12	11	8	3
BS2	12	11	6	2
BS3	12	11	6	3
BS4	12	11	4	2
BS5	12	11	7	2
BS6	12	11	7	2
BS7	12	11	7	3
BS8	12	11	6	2
BS9	12	11	9	2
BS10	12	11	7	3
BS11	12	11	6	2
BS12	12	11	4	3

A tab. 3 apresenta outra comparação entre as arquiteturas simplificada e genérica, mas agora levando em consideração os somadores utilizados em cada uma das arquiteturas e os respectivos números de bits em suas saídas. Novamente é possível perceber a minimização no uso de recursos gerada pelas simplificações realizadas.

A tab. 3 também apresenta as operações realizadas pelos somadores, de onde pode-se perceber que os somadores do primeiro e segundo estágio do *pipeline* (**1r**, **2r**, **1g**, **2g**, **1b**, **2b**, **3r**, **3g** e **3b**), responsáveis pelos resultados parciais relativos aos componentes R, G e B, realizam apenas somas. O somador **rg**, no terceiro estágio do *pipeline*, totaliza os resultados parciais relativos a R e

G e realiza somas e subtrações. O somador **rgb**, no quarto estágio do *pipeline*, é responsável pela totalização do resultado e realiza somas e subtrações, sendo especialmente desenvolvido para permitir que qualquer das entradas seja considerada negativa. Por fim temos o somador **inc**, no quinto estágio do *pipeline*, que é utilizado para arredondar o resultado.

Tabela 3 – Número de bits de saída e tipos de operações nos somadores utilizados nas soluções paralela genérica e simplificada

Somador	Solução Genérica N° de bits	Solução Simplificada N° de bits	Operações
1r	12	11	A+B
2r	12	7	A+B
3r	12	12	A+B
1g	12	10	A+B
2g	12	8	A+B
3g	12	12	A+B
1b	12	12	A+B
2b	12	7	A+B
3b	12	12	A+B
rg	12	12	A+B, A-B
rgb	9	9	A+B, A-B, B-A
inc	8	8	A(8:1)+A(0)

5. RESULTADOS DE SÍNTESE

As arquiteturas desenvolvidas foram descritas em VHDL e sintetizadas para FPGAs da família FLEX 10KE da Altera, com o auxílio da ferramenta Maxplus2 [9]. Os resultados obtidos na síntese das arquiteturas estão apresentados na tab. 4. A síntese foi direcionada para dispositivos EPF10K30ETC144-1 [10].

Na tab. 4, é possível observar algumas das características mais importantes extraídas do processo de síntese. A arquitetura original é a que ocupa o menor número de células lógicas, mas também é a que possui a menor taxa de processamento.

Em todos os aspectos relacionados na tab. 4, a arquitetura paralela simplificada obteve vantagens em relação à arquitetura genérica. A economia de recursos com as simplificações ficou em 71%, gerando um acréscimo de 34% na freqüência máxima de operação e, por conseqüência, na taxa de processamento, que atingiu 13 milhões de pixels por segundo na arquitetura simplificada. Por conseqüência, como esperado, não se justifica o uso da arquitetura paralela não simplificada.

Tabela 4 – Resultados de síntese comparativos

	Arquiteturas		
	Original	**Paralela Genérica**	**Paralela Simplificada**
Células Lógicas	281	1.494	433
Período (ns)	30,8	34,4	25,6
Freqüência (MHz)	32,47	29,06	39,06
Processamento (Mpixel/s)	3,61	9,69	13,02

O único inconveniente da solução simplificada está na dificuldade de compreender, passo a passo, as operações realizadas. As manipulações binárias envolvem, além dos deslocamentos, concatenações com zeros, diminuindo a transparência dos cálculos que são realizados.

Em estimativa anterior [8], era previsto um acréscimo no uso de recursos na ordem de 175% com a construção de uma arquitetura paralela, em relação à arquitetura original do conversor de espaço de cores. Ainda assim, este trabalho, apontava claramente a necessidade do desenvolvimento de uma arquitetura paralela de conversor de espaço de cores, uma vez que a maximização do desempenho do compressor JPEG justificaria o investimento no uso de recursos [8]. A construção da arquitetura paralela apresentada neste artigo demonstrou que a estimativa original estava equivocada, sendo que os dados de síntese apontaram um incremento de 54% no uso de recursos, muito abaixo do esperado. Este pequeno impacto no uso de recursos justifica ainda mais fortemente a sua utilização no compressor JPEG.

A arquitetura paralela simplificada é capaz de processar 13 milhões de pixels por segundo, que é uma taxa 260% superior àquela atingida pela arquitetura original, que é capaz de processar, no máximo, 3,6 milhões de pixels por segundo. Esta elevada taxa de produção, se justifica não só pelo pequeno impacto no uso de recursos, mas também pela possibilidade da perfeita integração desta arquitetura com as demais arquiteturas do compressor JPEG já foram desenvolvidas [8].

O uso da arquitetura paralela do conversor de espaço de cores contribuirá significativamente para o acréscimo no desempenho do compressor JPEG no qual esta arquitetura será inserida, uma vez que a sua taxa de produção é idêntica a taxa de consumo dos blocos já desenvolvidos do compressor JPEG. A taxa de produção

da arquitetura do conversor de espaço de cores original é de um elemento de cor a cada três ciclos de *clock*, enquanto que a máxima taxa de consumo dos demais blocos do compressor JPEG é de um elemento de cor a cada ciclo de *clock*. Deste modo, os demais blocos do compressor JPEG são sub-utilizados quando a arquitetura original de conversão de espaço de cores é utilizada, pois não é permitido a estes blocos operarem em sua taxa máxima de consumo. A arquitetura paralela do conversor de espaço de cores produz um elemento de cor a cada ciclo de *clock*, permitindo que os blocos do compressor JPEG operem na sua máxima taxa de consumo e, por conseqüência, em sua máxima taxa de produção, gerando um acréscimo estimado de cerca de 200% no desempenho do compressor JPEG.

Este significativo impacto no desempenho não se refletirá proporcionalmente na utilização de recursos, uma vez que estima-se que o uso da arquitetura paralela irá trazer um acréscimo máximo de 3% no uso de recursos no compressor JPEG. Portanto, é plenamente justificável o uso da arquitetura paralela do conversor de espaço de cores no compressor JPEG já desenvolvido.

6. CONCLUSÕES

Este artigo apresentou o desenvolvimento de arquiteturas paralelas para a conversão de espaço de cores de RGB para YCbCr, sendo apresentados detalhes arquiteturais, resultados do processo de síntese e resultados comparativos entre a arquitetura original e as arquiteturas paralelas.

Foram desenvolvidas duas arquiteturas, sendo uma genérica e outra simplificada, onde foram apresentados os resultados comparativos de síntese, que indicaram o impacto na minimização no uso de recursos e na maximização no desempenho, obtidos com as simplificações realizadas. A partir dos resultados de síntese definiu-se que a arquitetura genérica não seria útil para a aplicação alvo, optando-se, portanto, pelo uso exclusivo da arquitetura simplificada.

Considerando uma imagem colorida de 640 x 480 pixels, a arquitetura original do conversor de espaço de cores atinge uma taxa de processamento máxima de 11,7 imagens por segundo. Com o uso da arquitetura paralela simplificada, atinge-se a marca de 42,4 imagens por segundo, um incremento superior a 260% na taxa de processamento.

Os resultados de síntese também indicaram um acréscimo de 54% no uso de recursos na arquitetura paralela simplificada em relação à arquitetura original, um impacto muito pequeno, quando comparado aos ganhos de desempenho obtidos.

Através dos resultados de síntese também foi possível estimar que, em caso de uso da arquitetura paralela do conversor de espaço de cores no compressor JPEG, o desempenho do compressor crescerá cerca de 200% e sofrerá um acréscimo mínimo de cerca de 3% na utilização de recursos.

Com base nos dados apresentados neste artigo é possível concluir que é plenamente justificável a utilização de uma arquitetura paralela para o conversor de espaço de cores no contexto da compressão JPEG em hardware.

Como trabalho futuro está prevista a integração da descrição VHDL do conversor de espaço de cores paralelo às demais descrições dos blocos do compressor JPEG que já foram desenvolvidas e integradas.

7. REFERÊNCIAS

[1] "Home site of the JPEG and JBIG committees", 2002. <http://www.jpeg.org>

[2] The International Telegraph and Telephone Consultative Committee (CCITT). "Digital Compression and Coding of Continuous-Tone Still Images – Requirements and Guidelines". Rec. T.81, 1992.

[3] J. Miano. *Compressed Image File Formats – JPEG, PNG, GIF, XBM, BMP*, Addison Wesley, USA, 1999.

[4] W. Pennebaker e J. Mitchell. *JPEG Still Image Data Compression Standard*, Van Reinhold, USA, 1992.

[5] Z. Li. "Color in Image and Video". Simon Fraser University, 2002. <http://www.cs.sfu.ca/CourseCentral/365/li/material/notes/Chap3/Chap3.3/Chap3.3.html>.

[6] V. Bhaskaran e K. Konstantinides. *Image and Video Compression Standards Algorithms and Architectures*, Kluwer Academic Publishers, USA, 1999.

[7] L. Agostini e S. Bampi. "Arquitetura Integrada para Conversor de Espaço de Cores e Downsampler para a Compressão de Imagens JPEG". VII WORKSHOP IBERCHIP, 2001, Montevideo - Uruguai. 1 CD.

[8] L. Agostini. *Projeto de Arquiteturas Integradas para a Compressão de Imagens JPEG*. Dissertação de Mestrado –Universidade Federal do Rio Grande do Sul. II. PPGC, Porto Alegre, Brasil-RS, 2002.

[9] "Altera: The Programmable Solutions Company". San Jose, Altera Corporation, 2002. <http://www.altera.com>.

[10] Altera Corporation. "FLEX 10KE – Embedded Programmable Logic Devices Data Sheet – version 2.3". San Jose, Altera Corporation, 2001. 1 CD.