TB1 - Dezembro 2018

Implementação ASIC para um Compressor JPEG

Equipe de projeto

A equipe apresentada na Tabela I evidencia as pessoas diretamente envolvidas com o presente documento de especificação e projeto do ASIC.

Tabela I – Tabela de encargos

Nome	**Cargo**	**Contato**
Luis Felipe de Deus	Projetista Digital	felipe.deus@ecomp.ufsm.br
Nathanael Jorge Luchetta		nathanel.luchetta@ecomp.ufsm.br
Tiago Knorst	Projetista Digital	tiago.knorst@ecomp.ufsm.br

Fonte: Autores.

RESUMO

A proposição ASIC abordada nesse trabalho diz respeito a um compressor de imagens para o formato JPEG. O padrão é vastamente utilizado para o armazenamento e distribuição de conteúdos multimídia, havendo a codificação e decodificação usualmente realizada via software em processadores de propósito geral. Devido à ubiquidade do formato em dispositivos multimídia, o uso de unidades especializadas em SoCs é de grande valia, pois reduz o tempo de processamento e o consumo energético ao desempenhar a tarefa, além de disponibilizar o processador a desempenhar outras tarefas ao usuário.

Foram selecionadas para o projeto ASIC as unidades de maior custo computacional, a fim de viabilizar sua implementação ao longo do trabalho da disciplina. A principal unidade contemplada no projeto é o DCT, que desempenha a operação matemática da transformada discreta de cosseno sobre blocos da imagem, que resulta em dados no domínio da frequência e permitem a eliminação de redundâncias na imagem pouco perceptíveis à visão humana. O término do processamento da codificação fica a cargo do dispositivo ao qual o circuito ASIC estará acoplado, o qual terá apenas etapas de menor custo computacional para processar.

Como estratégia de verificação acoplou-se a descrição HDL do circuito proposto a uma implementação de código aberto disponível na internet. Foram selecionadas imagens comumente utilizadas para testes de compressores de imagem, usando-as para comparar de maneira quantitativa as perdas intrínsecas ao processo de compressão, além da comparação a imagem comprimida por software.

Sumário

1. INTRODUÇÃO

Este trabalho propõe o projeto de um circuito de propósito específico (ASIC) capaz de realizar a codificação de uma imagem em formato original *bitmap* para o formato JPEG.

JPEG (*Joint Picture Expert Group*) é um padrão internacional de compressão para imagens de tom contínuo, tanto para escala de cinza quanto coloridas. Seu vasto uso atualmente se deve à sua eficiência em reduzir drasticamente a quantidade de bits para se armazenar uma imagem, todavia mantendo níveis fidedignos de qualidade de imagem. Devido à sua capacidade de economia de armazenamento, esse formato é ubíquo em dispositivos multimídia, tendo aplicação em dispositivos de captura e exibição de imagens. (HUANG, 2008, p. 1).

A Figura 1 demonstra visualmente as etapas básicas entre a codificação e a decodificação de uma dada imagem.

Figura 1 - Fluxo básico de codificação e decodificação de imagens.

Fonte: (WEI, 2013).

O padrão segue quatro distintos modos de operação:

- **Modo sem perdas**: a imagem é codificada de modo a garantir a recuperação de cada pixel com exatidão em relação à imagem original, como ocorre na Figura 1;
- **Modo sequencial**: a imagem é comprimida de modo sequencial em uma única varredura, da esquerda para a direita e de cima para baixo;
- **Modo progressivo**: a imagem é comprimida com múltiplas varreduras, o que torna a imagem indistinguível previamente ao término da descompressão;
- **Modo hierárquico**: a imagem é comprimida em múltiplas resoluções de modo a se obter a de menor resolução enquanto a imagem original está em descompressão.

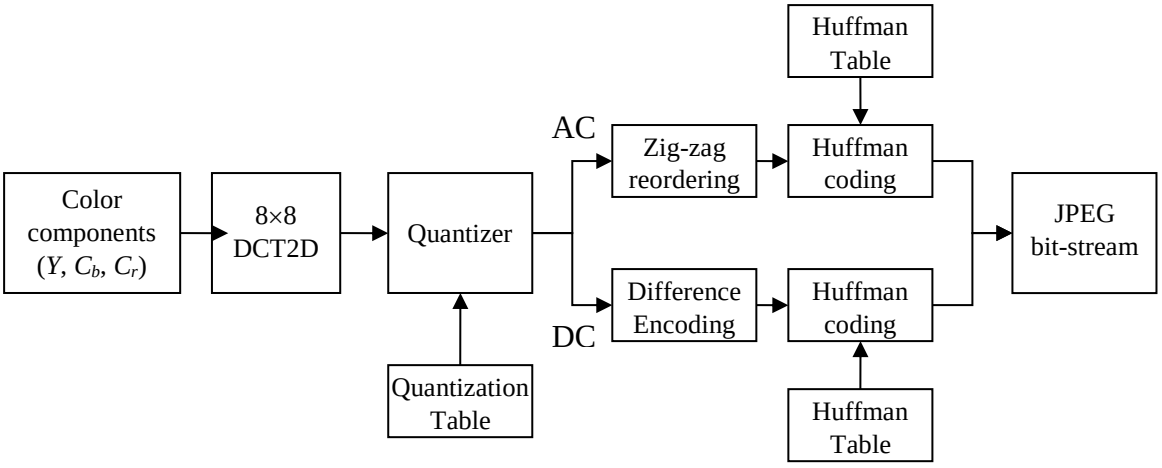
Os últimos três métodos utilizam-se da transformada discreta do cosseno (DCT) e posteriormente o processo de quantização, que reduzem a quantidade de informação necessária para representar a imagem, embora acarretem perdas intrínsecas a estes processos. Estas perdas variam de acordo com a precisão computacional do módulo DCT, como também à escolha da matriz de constantes utilizadas na tabela de quantização ilustrada na Figura 2. (HUANG, 2008, p. 1).

Será abordado nesse trabalho algumas das etapas do modo de operação sequencial, ilustradas pela Figura 2, que são:

- Conversão do espaço de cores de RGB para YCbCr e amostragem em blocos de 8x8 pixels;
- Transformada discreta do cosseno (DCT) de 2 dimensões;
- Reordenação em *zig-zag*;

As demais etapas do fluxo de compressão JPEG, tais como quantização, ordenação Huffman, entre outros, serão executas via *software* em um processador de proposito geral.

Figura 2 - Fluxo de codificação JPEG.



Fonte: (HUANG, 2008, p. 2).

1.1. Justificativa

Visto que grande parte dos dados que trafegam, seja na rede ou não, constituem conteúdo multimídia, aperfeiçoar esse processo é de grande valia, pois reduz a ocupação de banda e o consumo energético do sistema como um todo. Dentre o conteúdo multimídia boa parte são imagens, sejam em redes sociais, sites, aplicativos, ou até mesmo aplicações comerciais. (MIHIR, 2013, p. 1).

De maneira a reduzir a quantidade de bits que trafegam na rede, métodos de compressão de imagem são largamente utilizados, mas requerem computação de compressão e descompressão na origem e destino do sistema.

A execução de compressão e descompressão dos arquivos pode ser realizada por um circuito ASIC (*Application Specific Integrated Circuits*) de maneira mais eficiente em termos de desempenho e consumo energético, tendo em vista que o processo ocorre muitas vezes em circuitos embarcados, com restrições de energia e área de *hardware*, o emprego de um circuito ASIC com esse propósito é válido.

Pastuszak (2005) em sua implementação ASIC de um compressor JPEG, a uma frequência de 200 MHz, obteve em seus resultados aproximadamente 2 milhões de ciclos de *clock* para a codificação da imagem ilustrada pela Figura 6, o que resulta num tempo de codificação de 10 ms.

Para efeitos de comparação foi utilizado o simulador de arquiteturas de propósito geral gem5. A ferramenta extraiu o tempo de execução de um *software* referência JPEG *Compressor*, implementado por Kornel (2013), na compressão JPEG da Figura 6, tendo como base uma implementação em alto nível de arquitetura *out of order* do processador ARM Cortex A9, em frequência de 1 GHz, utilizado em *SoCs* voltados comumente a dispositivos móveis.

A Tabela II ilustra a comparação em termos de tempo de execução entre as plataformas citadas anteriormente, evidenciando a superioridade de desempenho de aproximadamente dez vezes em um circuito ASIC frente ao circuito de propósito geral executando o algoritmo.

Tabela II - Comparação de tempo de execução entre circuito ASIC e processador de propósito geral.

Plataforma	**Tempo de execução (ms)**
ARM Cortex A9	580
ASIC	10

Fonte: Autores.

1.2. Objetivo

Pretendeu-se implementar os principais circuitos citados nas etapas descritas anteriormente, de modo a propiciar a maior parte do fluxo de codificação JPEG. Essa implementação teve como objetivo obter desempenho acima de circuitos de propósito geral, como ilustrado pela Tabela II, como também menor dissipação de potência, a fim de torna-lo viável economicamente.

Todas as etapas deviam respeitar os requisitos funcionais do circuito, ou seja, não abrir mão de nenhum aspecto funcional do circuito em detrimento de outras restrições.

Por fim, ao final, a imagem comprimida havia de respeitar uma taxa máxima de perdas intrínsecas no processo, sendo possível o retorno à imagem original, ou seja, a realização do processo de descompressão.

1.3. Cronograma

O cronograma definido no Quadro 1 é constituído pelas seguintes etapas de implementação:

1. Especificação;
2. Codificação;
3. Verificação RTL;
4. Síntese Lógica;
5. Verificação pós-síntese;
6. Síntese física.

Quadro 1 - Cronograma das atividades desenvolvidas.

	semanas							
atividade	1	2	3	4	5	6	7	8
1	X							
2		X	X					
3			X	X				
4					X	X	X	
5					X	X	X	
6								X

Fonte: Autor.

2. CIRCUITO

A Figura 2 ilustra cada uma das etapas do fluxo de codificação JPEG, neste projeto serão desenvolvidos três blocos fundamentais no processo de compressão JPEG, portanto cada um será transformado em um bloco de *hardware*. Devido o enfoque do projeto, que é para utilização em sistemas embarcados, irá se priorizar a área em *chip*, que é diretamente proporcional à complexidade do circuito. Dessa maneira, será utilizado para o projeto a amostragem da imagem em blocos de 8x8 *pixels* RGB, utilizando 8 *bits* para cada cor (AL-ANI, 2013, p. 4). Seria possível a exploração de paralelismo, utilizando três blocos do DCT, o que não é contemplado neste projeto.

A etapa inicial diz respeito à conversão do espaço de cores, tendo em vista que a entrada é no padrão RGB (*Red Green Blue*) que descrevem o *pixel* da imagem de entrada, o qual será convertido para o espaço YCbCr, que é largamente utilizada em compressões digitais devido ao maior numero de correspondências na matriz de cores, onde a componente Y representa a informação de luminância, enquanto as componentes Cb e Cr possuem a informação de cor, em tons de azul e vermelho REF.

Seguindo o fluxo, o circuito irá processar a transformada discreta do cosseno bidimensional, o DCT tem sido usado como um núcleo para a compressão de padrões JPEG e MPEG, porém, de acordo com J. Chiang e H. Huang (1996), sua computação é tão intensiva e de grande necessidade que vários métodos de implementação para a função foram apresentados, sempre em busca de uma alta velocidade e qualidade de conversão.

O funcionamento do DCT está na escolha de uma base onde os primeiros elementos tenham entradas vizinhas com pouca variação de valores, ou seja, baixa frequência, enquanto os últimos tenham maior variação. Tendo em vista que os *pixels* em geral tem correlação com seus vizinhos, é interessante que os primeiros elementos tenham pouca variação de valores, enquanto os últimos elementos possuem valores com maior variação.

Por fim, o ultimo processamento é referente à operação de *scanner* zig zag, a matriz resultante do DCT é lida em "zig zag" da esquerda para direita e de cima para baixo, fazendo com que os primeiros elementos lidos sejam os de maior valor, ou seja, de maior energia.

2.1. Funcionalidades

Este projeto não implementa funções analógicas e possui apenas funcionalidades digitais, reunidas no Quadro 2.

Quadro 2 – Funcionalidades de topo do ASIC

Função	Descrição
1	Recebe uma entrada RGB de 24 bits através do pino *RGB_i*
2	Habilita a conversão de cores pelo pino *data_av_i*
3	Converte a entrada RGB para Y Cb Cr
4	Armazena separadamente as componentes de um bloco de 8x8 *pixels*
5	Habilita o envio de dados para o calculo da transformada discreta do cosseno pelo sinal *datai_dct_s*
6	Habilita o processamento da transformada discreta do cosseno pelo sinal *start_dct_s*
7	Envio de dados para a operação de zigzag através do sinal *dct_out_s*
8	Habilita o processamento da operação de zigzag através do sinal *ready_dct_s*
9	Informa o final do processamento do chip para uma componente através do pino *ready_o*
10	Gera saída de dados através do pino *data_o*

Fonte: Autores.

2.2. Diagrama de topo VHDL

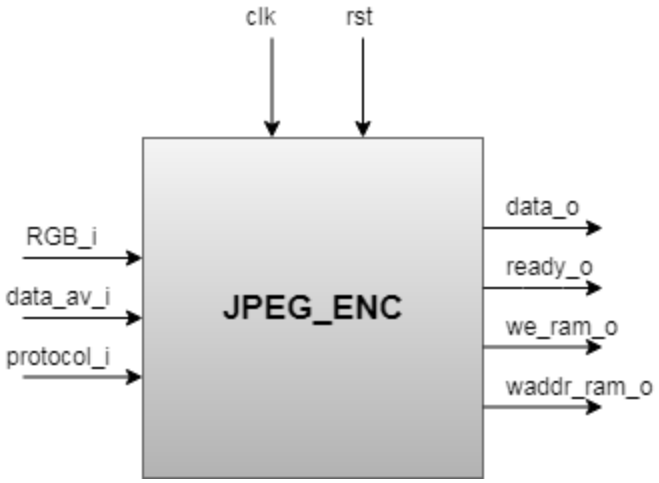
O diagrama de topo VHDL consiste, no modelo *black-box* (caixa-preta) com as entradas e saídas da implementação, descritas pelo Quadro 3 juntamente da representação binária necessária para cada caso.

Quadro 3 – Especificação das portas do circuito

nome	direção	tamanho (bits)	descrição
clk	*in*	1	clock global
rst	*in*	1	*reset* global
RGB_i	*in*	24	entrada de dados *pixel* RGB
data_av_i	*in*	1	habilita conversão de cores, dado valido.
protocol_i	*in*	1	habilita protocolo de buffer para entrada do dct
data_o	*out*	12	saída de dados
ready_o	*out*	1	sinal de pronto, nível alto indica dados pronto na saída
we_ram_o	*out*	1	habilita a escrita na RAM de saída
waddr_ram_o	*out*	7	endereço de gravação na RAM de saída

Fonte: Autores.

Figura 3 - Diagrama *black-box* da entidade topo



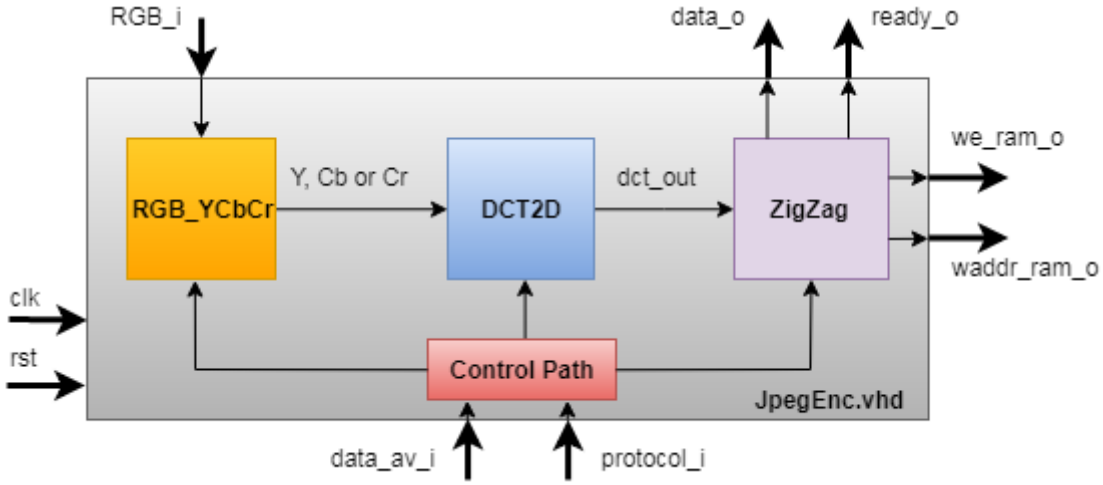
Fonte: Autores.

2.3. Diagrama de blocos e Interconexões

É possível visualizar o digrama de blocos do interior do projeto através da Figura 4, evidenciando as principais conexões bem como os pinos de entrada/saída. Destaca-se a entrada *RGB_i* descrita no Quadro 3, a qual interage com o bloco de conversão de cores, bem como as saídas do *chip* que tem sua origem do bloco "ZigZag". As saídas denominadas *we_ram_o* e *we_ram_addr* são utilizadas para gravação dos valores em uma memoria RAM externa ao *chip*, para, por exemplo, facilitar o uso de um processador de proposito geral, que eventualmente irá utilizar as saídas produzidas pelo *chip*.

As conexões de *clock*, *reset* foram omitidas do interior do diagrama para melhor visualização, entretanto, todos os blocos são síncronos, portanto dependem do *clock*. Outra constatação é de que os sinais do bloco *Control Path* foram retratos apenas como uma seta genérica que entra nos demais blocos.

Figura 4 - Diagrama de blocos e conexões simplificado da entidade topo



Fonte: Autores.

2.4. Especificação funcional (*datapath* e máquinas de estados)

Nesta seção são abordadas as características funcionais do circuito proposto, evidenciando o fluxo de dados por meio de máquinas de estado, digramas de blocos e descrições em alto nível, bem como as respectivas funcionalidades que cada bloco implementado desenvolve no projeto.

2.4.1. *Datapath*

2.4.1.1. Descrição

Para este trabalho os blocos descritos como: “RGB_YCb_Cr”, “DCT2D” e “ZigZag” não foram encapsulados dentro de uma entidade chamada *DataPath*. Nesta seção será abordado os blocos mencionados anteriormente como fazendo parte do *DataPath*.

Dentro do *DataPath* encontra-se o circuito de conversão de cores, que converte uma entrada de 24 *bits* RGB, para o domínio Y Cb Cr, a conversão acontece em um único ciclo, e gera as saída *Y_o*, *Cb_o* e *Cr_o*. Ao gerar uma saída da conversão de cores, o *ControlPath* gerência o armazenamento de cada componente, em 64 registradores para cada componente, que possuem largura de 8 *bits*, para suportar um bloco de 8x8 *pixels*.

Ao final do armazenamento os registradores das componentes *Y_array*, *Cb_array* e *Cr_array* possuem 64 *pixels* separados em componentes, é então iniciado o protocolo de envio para o *DCT2D*. Para o envio de dados ao bloco *DCT2D* é necessário particionar o envio, enviando 8 *bits* cada ciclo de *clock*, entretanto é necessário mandar uma componente completa por vez. Vale ressaltar a importância do método empregado, de armazenar em registradores de 8 *bits* separadamente todas as componentes pois facilita o envio ao DCT, minimizando o protocolo de envio.

Antes de mandar o primeiro *byte* para o DCT é colocado em nível alto o sinal *start_dct_s* por um ciclo de *clock* indicando o inicio do processamento, então, a cada ciclo é enviado um *byte*, sendo enviados todos os 64 *bytes* de cada componente por vez. Uma vez que o bloco *DCT2D* termina de calcular uma componente, é produzida uma nova matriz 8x8, esta agora de 12 *bits*, visto que ao aplicar a operação matemática são gerados novos valores de maior magnitude.

As saídas do DCT são descritas pelo sinal *dct_out_s* e o sinal *ready_dct_s* ativa o processamento no bloco *ZigZag*, o qual começa a ler a saída produzida pelo *DCT2D*. Por fim, após ter processado uma componente informa a CPU, externa ao *chip*, que o processamento desta componente foi finalizado, isto se dá através do pino de saída *ready_o*, e consequentemente o pino de saída de dados *data_o* produz as saídas do zigzag ao exterior do *chip*.

2.4.1.2. Função

Quadro 4 – Descrição das funções realizadas pelo bloco de dados

Função	Descrição
1	Converte a entrada RGB de 24 *bits* em componentes Y CB e Cr de 8 *bits* cada
2	Realiza a transformada discreta do cosseno a partir de 64 entradas de 8 *bits* de uma componente
3	Realiza o *scanner* zigzag procurando redundâncias

Fonte: Autores.

2.4.1.3. Quadro de conexões

Quadro 5 – Especificação das conexões do bloco de dados

Nome	Tamanho	Tipo	Conexão	Descrição
clk	1	entrada	pino externo	*Clock* global
rst	1	entrada	pino externo	*Reset* global
RGB_i	24	entrada	pino externo	Entrada de dados, um *pixel* RGB
data_av_i	1	entrada	pino externo	Habilita a conversão de cores
ready_o	1	saída	pino_externo	Indica o fim do processamento de uma componente
data_o	12	saída	pino_externo	Saída de dados vinda do ZigZag
we_ram_o	1	saída	pino externo	Habilita a gravação na RAM externa ao *chip*
waddr_ram_o	7	saída	pino externo	Endereço para escrita na RAM externa ao *chip*
datai_dct_s	8	entrada	Control Path	Entrada de dados do DCT2D
Y_s	8	saída	Control Path	Saída de dados da componente Y
Cb_s	8	saída	Control Path	Saída de dados da componente Cb
Cr_s	8	saída	Control Path	Saída de dados da componente Cr
start_dct_s	1	entrada	Control Path	Habilita o inicio do processamento do DCT2D

Fonte: Autores.

2.4.2. *Control path*

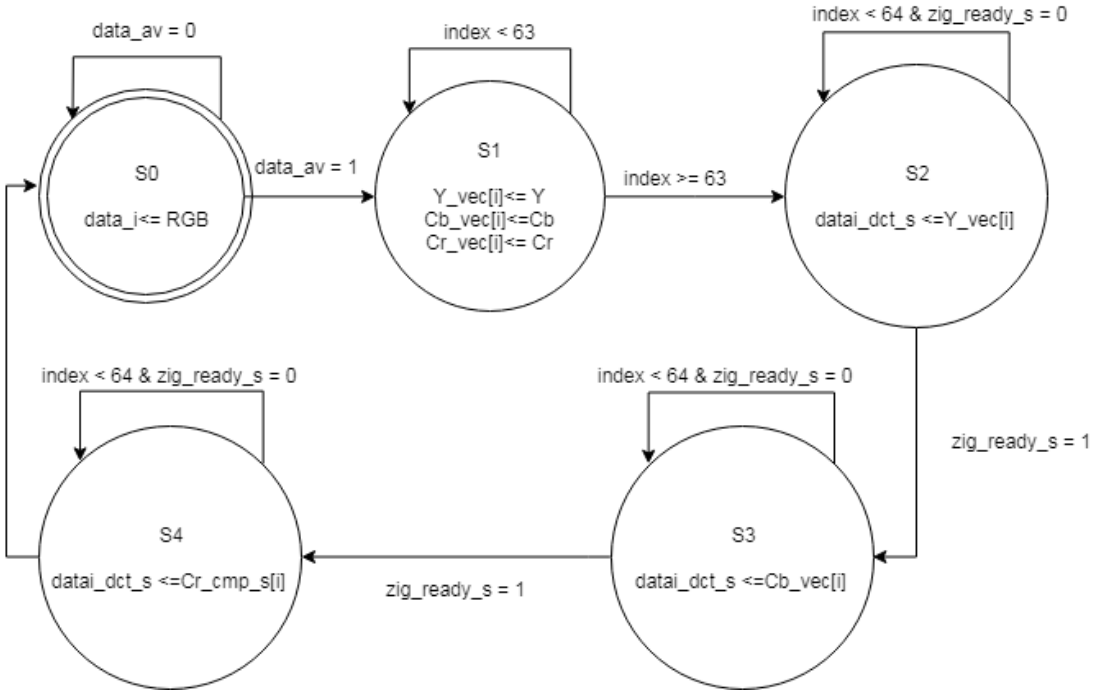
2.4.2.1. Descrição

O bloco denominado como *ControlPath* realiza o armazenamento de um bloco (64 *pixels*) das componentes separadamente em registradores, como é descrito na Figura 5 pelo estado S1. Realiza também a troca de estados que permite o envio ordenado de dados, através do sinal de índice *index* referente ao endereçamento de qual componente irá para o DCT.

Os dados são enviados para a realização da transformada discreta do cosseno, que opera no *DataPath*, e posteriormente os dados são enviados ao bloco *ZigZag*, que retorna o sinal de conclusão de processamento *zig_ready_s*.

Ao término do processamento de cada componente, o bloco de controle mantém o circuito no estado S0 a espera de nova requisição (acionada pelo pino de entrada *data_av_i*) e posteriormente dispara o protocolo de envio em uma nova requisição.

Figura 5 - *Finite State Machine* do *Control Path*



Fonte: Autores.

2.4.2.2. Função

Quadro 6 – Descrição das funções realizadas pelo bloco de dados

Função	Descrição
1	Efetua o armazenamento nos registradores de bloco separando as componentes
2	Efetua o envio ordenado (protocolo) de *bytes* para o *Data Path*
3	Gerar o sinal (*start_dct_s*) que habilita o inicio do processamento do DCT2D

Fonte: Autores.

2.4.2.3. Quadro de conexões

Quadro 7 – Especificação das conexões do bloco de dados

Nome	Tamanho	Tipo	Conexão	Descrição
clk	1	entrada	pino externo	*Clock* global
rst	1	entrada	pino externo	*Reset* global
data_av_i	1	entrada	pino externo	Habilita a o armazenamento dos registradores das componentes
protocol_i	1	entrada	pino_externo	Habilita o protocolo de envio do *byte* ao DCT2D
datai_dct_s	8	saída	Data Path	Entrada de dados do DCT2D
Y_s	8	entrada	Data Path	Saída de dados da componente Y
Cb_s	8	entrada	Data Path	Saída de dados da componente Cb
Cr_s	8	entrada	Data Path	Saída de dados da componente Cr
start_dct_s	1	saída	Data Path	Habilita o processamento do DCT2D

Fonte: Autores.

2.5. Estratégia de verificação

A verificação funcional do circuito se deu a partir de vetores de teste na entrada do circuito se baseando em imagens de referência da internet, exemplificado pela Figura 6 de tamanho 512x512 *pixels*, e pela Figura 7 de tamanho 64x64, foi utilizada a mesma imagem, porém redimensionada devido ao fato de que na análise de potência os arquivos de atividade de chaveamento ficavam muito extensos.

Como o *chip* desenvolvido não compreende todos os blocos de um compressor JPEG, será analisado os valores fazendo comparação através da simulação computacional com o circuito proposto por Krepa (2009), descrito em HDL, de no entanto não sintetizável.

Será efetuada a comparação da conversão de cores, pelo fato de ambos os projetos utilizarem a conversão para os espaços de cores YCbCr, e posteriormente a comparação com a transformada discreta do cosseno, mesmo que a implementação seja diferente, logo será possível analisar os percentuais de erro.

Por fim, será comparado o bloco de ZigZag apenas de maneira funcional, devido aos valores no bloco anterior (DCT) provavelmente serão diferentes.

Figura 6 - Imagem de teste: Lena 512x512

Fonte: (http://graphics.stanford.edu/~jowens/223b/lena/lena_color.jpg. Acessado em 24/09/2018).

Figura 7 - Imagem de teste: Lena 64x64

Fonte: (http://graphics.stanford.edu/~jowens/223b/lena/lena_color.jpg. Acessado em 24/09/2018).

Para esta análise foi efetuada a simulação pós-síntese física do circuito proposto, utilizando como vetor de teste a Figura 7 que possui tamanho 64x64.

Como descrito anteriormente, foram analisadas as saídas dos blocos deste projeto em comparação com o circuito de Krepa (2009), o mesmo possui um compressor JPEG completo,

descrito em uma HDL, entretanto não é fisicamente sintetizável, apenas utilizado para simulação, e neste projeto tem o objetivo de comparar o modelo teórico computacional, com o projeto físico elaborado neste trabalho.

Inicialmente, o primeiro aspecto de comparação é a verificação funcional da conversão de cores empregado do espaço de cores RGB para YCbCr, os resultados são apresentados na Figura 13 e

Figura 14 que encontram-se no Apêndice A. Respectivamente, a simulação proposta por Krepa (2009) elaborado no *software* ModelSim, e a simulação pós-síntese física elabora no NcLaunch, ferramenta da empresa Cadence. É possível notar que os resultados foram exatamente iguais para ambos os circuitos, sendo o percentual de erro igual à zero, comprovando a funcionalidade do circuito proposto.

O segundo bloco a ser verificado é a transformada discreta do cosseno, que como descrito na seção 2, tem a função de escolher uma base onde os primeiros elementos tenham entradas vizinhas com pouca variação de valores, ou seja, baixa frequência, enquanto os últimos tenham maior variação. A Figura 15 e

Figura 16 que se encontram no Apêndice B mostram os resultados obtidos, respectivamente a simulação de Krepa (2009) e a simulação pós-síntese física deste trabalho.

É possível verificar que em nível de valores, os resultados não coincidem, entretanto este fato é explicado devido as diferenças nas implementações, tais como constantes empregadas no cálculo ou a forma de implementação da transformada.

Por fim, o ultimo bloco diz respeito ao *scanner* zigzag, tal como o DCT, foi utilizado às mesmas métricas de comparação, entretanto como a entrada deste bloco está conectada ao bloco anterior, os valores não coincidirão, devido à circunstancias da implementação do DCT. Contudo o que deve ser observado é a funcionalidade do bloco em ambas as aplicações, a qual já foi descrita na seção 2, que é fazer com que os primeiros elementos lidos na nova matriz resultante sejam os de maior valor, ou seja, de maior energia, este processo facilitaria a codificação Huffman que emprega uma arvore binária para a matriz resultante, porém este bloco não foi desenvolvido neste projeto.

Os resultados podem ser visualizados na Figura 17 e Figura 18, que encontram-se no Apêndice C.

2.5.1. Métrica de desempenho

Com relação aos resultados apresentados na estratégia de verificação, foi elaborado uma métrica para amostrar de forma quantitativa, a diferença entre o bloco DCT2D proposto por Krepa (2009), e o que foi implementado neste projeto. É possível visualizar na Tabela III a relação do erro do DCT através da formula do erro médio quadrático entre as duas implementações, destaca-se que excluindo a primeira amostra, os outros percentuais foram relativamente baixos.

Tabela III - Métrica de desempenho e avaliação do bloco DCT2D

Métrica de relação do erro do DCT							
DADO	**1**	**2**	**3**	**4**	**5**	**6**	**MÉDIA**
Krepa	209	35	10	4086	4087	4058	**2081**
JpegEnc	0	3	9	4051	4061	4090	**2036**
Relação	10,200%	0,781%	0,024%	0,854%	0,635%	-0,781%	**1,952%**
Relação Quadrática	1,040%	0,006%	0,000%	0,007%	0,004%	0,006%	**4,211%**

Fonte: Autores.

3. PROJETO

Nesta seção será abordo o projeto HDL do *chip* JpegEnc, evidenciando as *constraints* empregadas, para uma melhor precisão de simulações, bem como justificando escolhas de projeto que foram tomadas ao longo do desenvolvimento.

No inicio do projeto, foi definido os blocos que este projeto iria contemplar, tendo em vista que o tempo de projeto era reduzido para uma aplicação tão grande, portanto foi escolhido blocos que ocupariam um alto nível de processamento, como as multiplicações que envolvem as conversões de cores, ou então a operação matemática da transformada discreta do cosseno.

Para o bloco de conversão de cores, foi implementado toda a conversão em um único ciclo de *clock* para não atrasar o processamento de dados, tendo em vista que depois da conversão é necessário o armazenamento separadamente das componentes, que deste modo, levam 64 ciclos de *clock*, isso porque a conversão demora apenas um ciclo, caso contrário, o numero de ciclos seria o produto dos ciclos de conversão por 64 *pixels*.

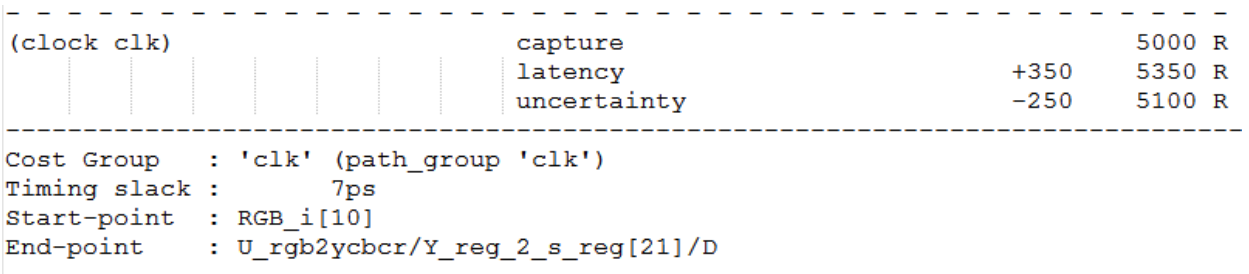
Nas primeiras versões do projeto, foi utilizado um controle *pipeline* para, na medida em que eram armazenadas as componentes já fosse enviado para a entrada do DCT, entretanto esta implementação se mostrou muito complexa, e iria aumentar muito a área do controle, portanto foi descartada. A modelo que foi feita é de que é necessário esperar todo o armazenamento das componentes para depois começar o envio de dados para o DCT (sequencial).

No bloco do DCT, como foi utilizado uma implementação bidimensional, foram estanciados dois *cores* do DCT, no entanto uma depende do outro, então, o segundo recebe a matriz de saída do primeiro, e é iniciado pelo sinal de *ready* do primeiro.

Do que se refere às *constraints* do projeto, foram usadas restrições para o *clock* como a incerteza de 0.25 ns e latência de 0.35 ns, e o *slew* que é o tempo de subida e descida da rampa do *clock*, foi utilizado 0.146 ns e 0.164 ns respectivamente. Existem também as restrições de tempo para a entrada no pino chegar ao interior do *chip* ou sair, respectivamente 1 ns e 2.98 ns.

O circuito após passar nos testes em simulação apenas de VHDL, foi estimado a maior frequência do *chip*, a qual foi de 200 MHz, como comprova a Figura 8 que evidencia o *report timing* da Encouter RTL. É interessante perceber o caminho crítico do *chip* que foi entre a entrada de dados e o bloco de conversão de cores, tendo em vista que este bloco é executado em um único ciclo de *clock.*

Figura 8 - *Report timing* JpegEnc @200MHz



Fonte: Encouter RTL – Cadence *Design Systems*

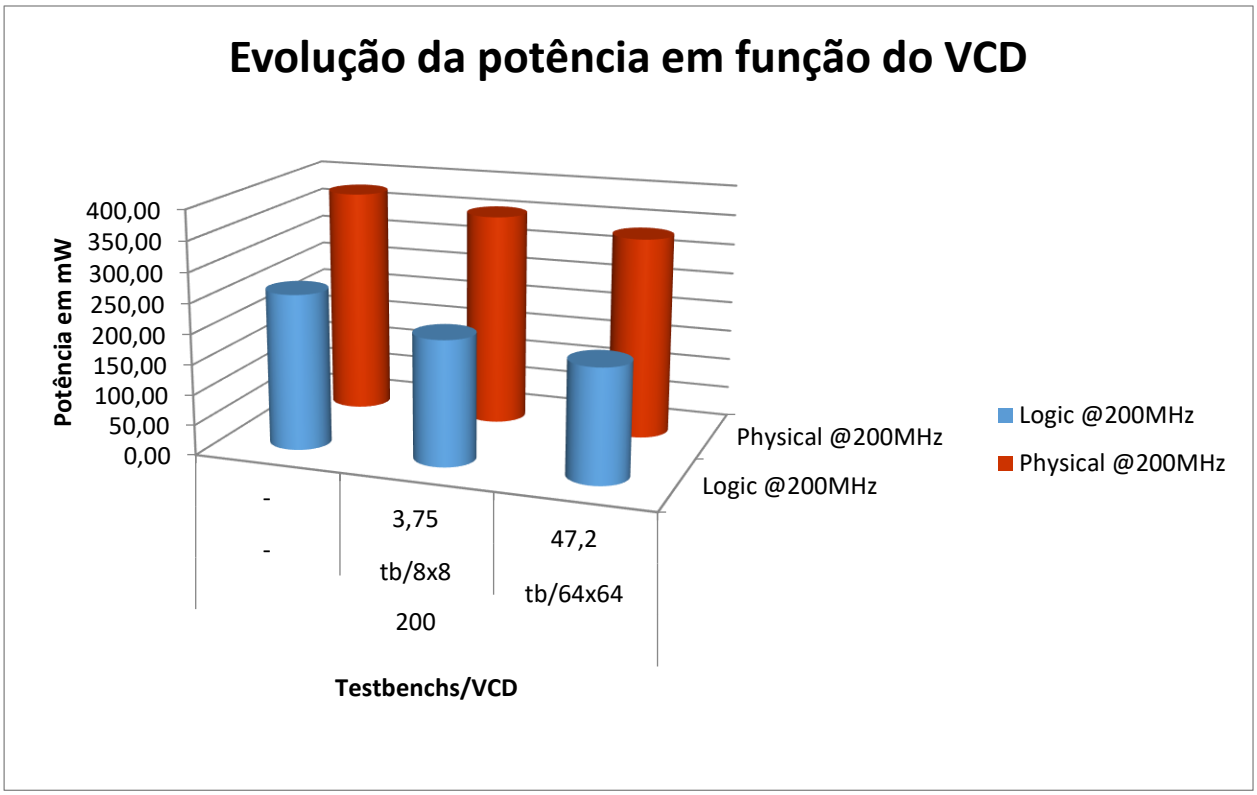
4. RESULTADOS

Nesta seção será abordado os resultados obtidos na síntese lógica e síntese física do *chip* JpegEnc, evidenciando os resultados de potencia, área, performance, e energia fazendo uso de diferentes técnicas para melhor visualizar os resultados obtidos.

A primeira análise é referente à potência consumida pelo *chip*, foi simulada a aplicação de imagem da Figura 7, de tamanho 64x64, e a mesma imagem redimensionada para 8x8, com base nisso foram obtidos os resultados descritos na Tabela V que se encontra no Apêndice E.

É possível verificar a potência do *chip* que ficou em torno de 200 mW na síntese lógica, e posteriormente na síntese física cerca de 350 mW, o aumento de potência é devido as otimizações em área que a ferramenta EDI da Cadence implementa, utilizando células maiores, que requerem maior potência. Para uma melhor visualização a Figura 9 mostra graficamente a evolução da potência em função da atividade de chaveamento com diferentes vetores de teste.

Figura 9 - Gráfico da evolução da potência em função dos vetores de teste



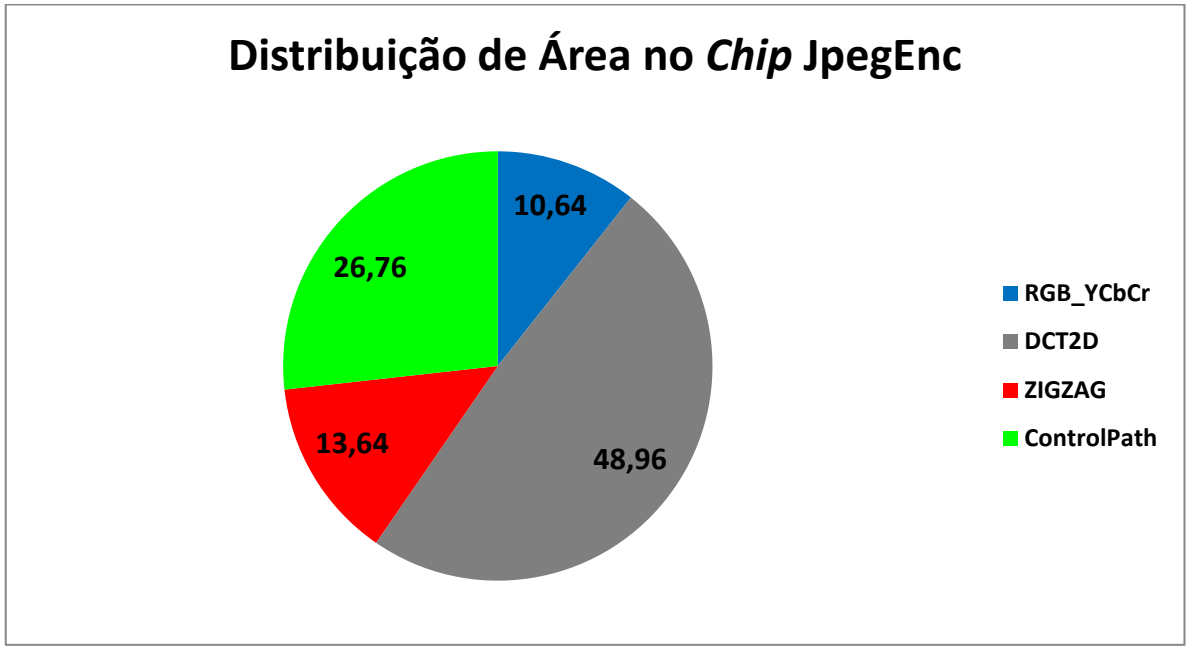
Fonte: Autores.

A segunda análise diz respeito a área do *chip*, como o enfoque deste projeto é para sistemas embarcados, então era desejado a menor área possível, no entanto foi requerido que o *chip* operasse na maior frequência então o aumento de área era inevitável.

É possível ver na Tabela V, que se encontra no Apêndice E, que a área de *chip* na síntese lógica foi em torno de 1,5 mm² e posteriormente na síntese física com as otimizações da ferramenta passou a ser menor que 1 mm². Os resultados empregados são satisfatórios visto que uma implementação JPEG em *hardware* não é algo trivial, a Figura 10 mostra um gráfico de pizza para uma melhor visualização de quanto cada bloco do circuito ocupa em área de *chip (die)*, estes valores são para a síntese lógica. Enquanto a Figura 11 é a imagem gerada pela

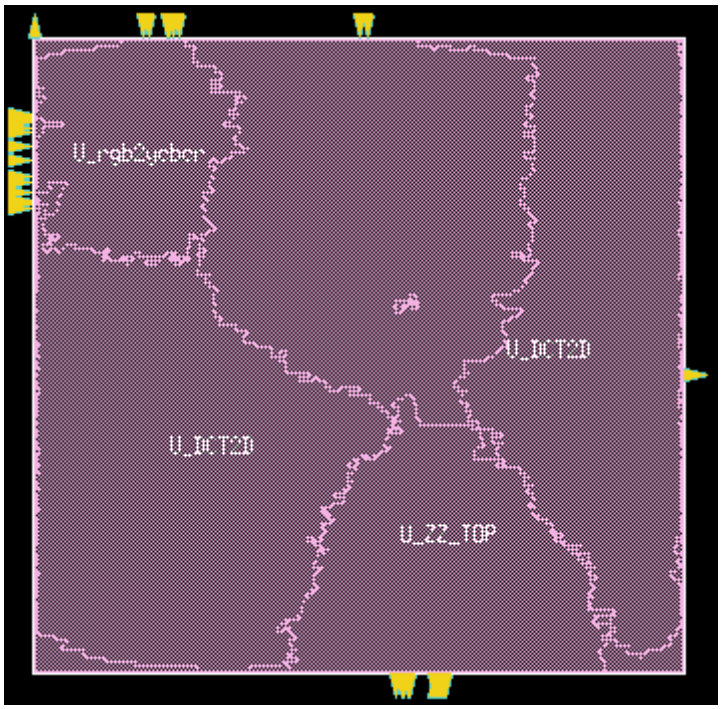
ferramenta EDI da Cadence, que mostra a distribuição física da área de cada bloco dentro do *chip*.

Figura 10 - Gráfico da distribuição de área no chip referente a síntese lógica



Fonte: Autores.

Figura 11 - Distribuição da área no *chip* referente a síntese física



Fonte: Encouter Digital Implementation – Cadence *Design Systems*

Por fim, a ultima análise diz respeito à performance do *chip*, e é neste resultado que é possível realmente afirmar e justificar o projeto do ASIC. De acordo com a Tabela V, descrita no Apêndice E, para uma imagem de tamanho 8x8 o *chip* JpegEnc leva 3,75 µs para efetuar o processamento, já em uma imagem de tamanho 64x64, foi requerido um tempo de processamento de 47,2 µs.

Para uma imagem de 512x512, que seria algo mais próximo a realidade, como descreve a simulação pós-síntese física da Figura 19 e Figura 20 que se encontra no Apêndice D, sendo a primeira retratando o final da simulação em 2,76 ms e a segunda mostrando uma visão macro do processamento de dados até ser concluída toda a imagem.

Fazendo uma comparação com o processador de proposito geral ARM Cortex A9, descrito na Tabela II, o *chip* JpegEnc leva aproximadamente 200 vezes menos tempo. Porem a comparação descrita anteriormente não é exatamente justo, pois o projeto não engloba todo o fluxo de compressão JPEG, levando em consideração isso, foi estimado que o *chip* levasse o dobro do tempo para fazer todo o processo, havendo ainda assim 100 vezes de ganho de desempenho.

Comparando-se aspectos não funcionais com o processador ARM, ilustrado pela Tabela IV, é possível observar que mesmo em tecnologia mais antiga em relação ao processador, o acréscimo de área e potência é baixo, dado o ganho de desempenho proporcionado pelo ASIC, tornando-o uma alternativa interessante como unidade especializada no SoC.

Tabela IV – Comparativo de área e potência entre o JpegEnc e o processador ARM Cortex A9.

Dispositivo	Área (mm²)	Potência (W)
ARM Cortex A9 (45 nm)	16,092	7,32316
JpegEnc (180 nm)	0,988	0,33404
Acréscimo ASIC sobre ARM	6,14%	4,56%

Fonte: Autores.

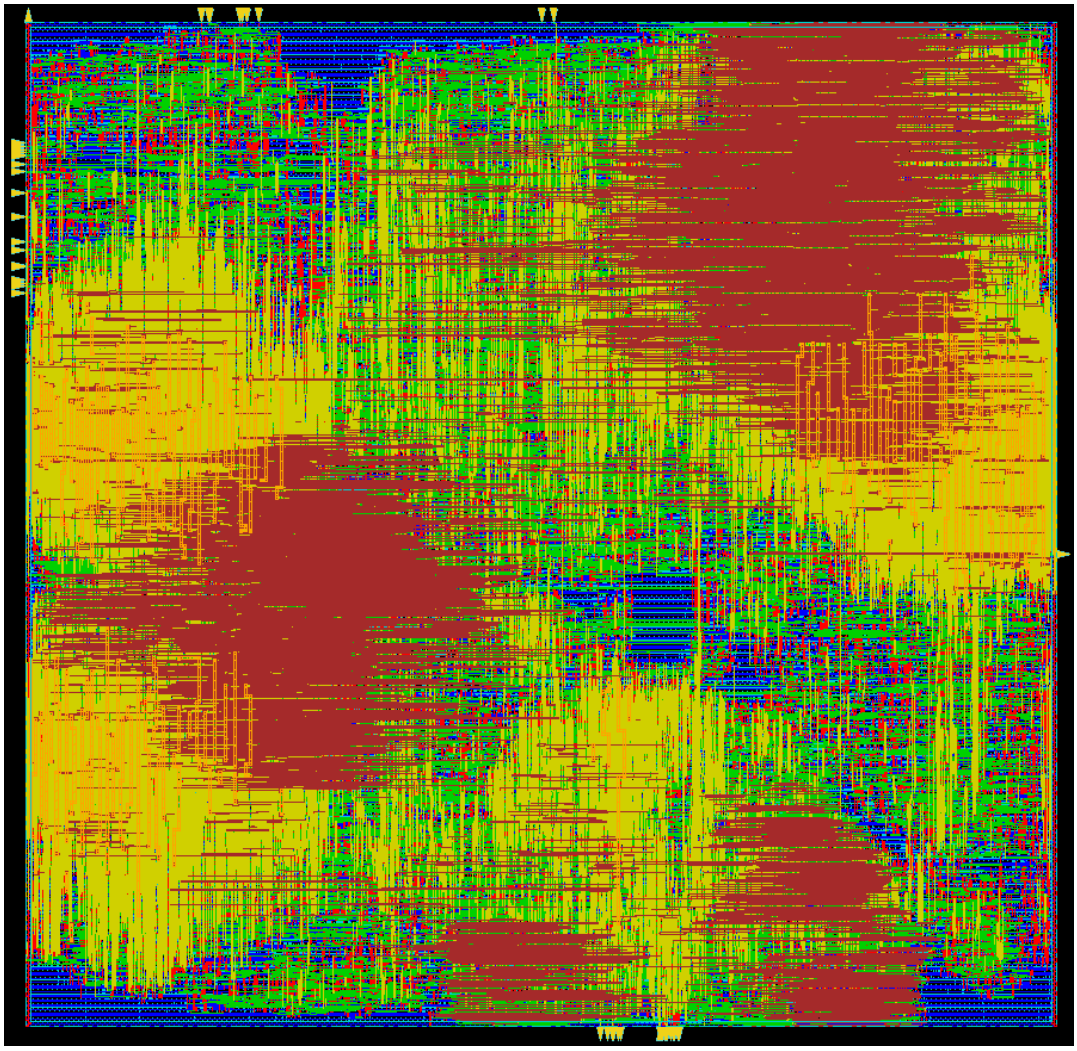
A segunda comparação é com uma referencia da mesma área de atuação, ou seja, um ASIC, descrito por Pastuszak (2005), e evidenciado na Tabela II, o circuito leva 30 ms para o processo inteiro de JPEG, portanto o *chip* JpegEnc levando em conta a estimativa anteriormente descrita, seria aproximadamente 11 vezes melhor em desempenho. No entanto ressalta-se que Pastuszak (2005) opera em 66 MHz, enquanto o *chip* JpegEnc em 200 MHz, não foi encontrado dados de área e potencia, entretanto potencialmente o circuito de Pastuszak (2005) seja melhor neste quesito.

Não foram retirados dados de potência do processamento da Figura 6 devido ao fato do tamanho do arquivo da atividade de chaveamento. Entretanto, tendo em vista dados de tempo e de potência é possível calcular a energia consumida pelo *chip* JpegEnc. A partir do produto de ambos se chega à estimativa de que para comprimir a imagem da Figura 7, de tamanho 64x64, seja necessário 15,76 µJ. É possível estimar a energia para a imagem da Figura 6 utilizando a potência do *worst case*, a qual a ferramenta proporciona sem vetores de teste, logo, se chegaria a 1,03 mJ.

A partir desta análise é possível concluir que o projeto ASIC do *chip* JpegEnc teve resultados expressivos, uma vez que o há grande ganho de performance em relação a um processador de propósito geral, com aumento relativamente baixo de área e potência ao chip.

Por fim, a Figura 12 mostra o *layout* depois da síntese física do *chip* JpegEnc pronto para ser fabricado, funcional a 200 MHz.

Figura 12 - *Layout* do *chip* JpegEnc



Fonte: Encouter Digital Implementation – Cadence *Design Systems*

5. CONCLUSÕES

Ao final do projeto, é possível destacar algumas conclusões da proposta abordada ao longo deste trabalho, como por exemplo, a viabilidade do projeto em questão, o *chip* JpegEnc teve resultados expressivos com relação a performance sobre as comparações executadas. Ainda mais notório, é fazer um paralelo com um sistema embarcado, onde o processamento de compressão de imagens seria executado em um circuito dedicado, aliviando o uso de CPU, para que a mesma se dedique a operações de gerencia, escalonamento de tarefas e SO (Sistema Operacional).

Com relação à área, a fabricação do *chip* é viável, tanto fisicamente quando economicamente, sendo 1 mm² nada extraordinário, a potência, no entanto excedeu a margem de expectativas, sendo em torno de 350 mW, porém ressalta-se que a tecnologia de células usada foi um tanto quanto obsoleta, justificando a potência encontrada. Por fim, mesmo que a potência foi além do que se esperava, o tempo de processamento excedeu positivamente as expectativas, impactando drasticamente na energia.

Do impacto das decisões de projeto nos resultados, pode-se destacar positivamente a utilização da conversão de cores em um ciclo, evitando um longo período de armazenamento de dados, e de certa forma, negativamente a utilização de controle sequencial em detrimento do controle *pipeline*, abordagem que certamente iria trazer resultados ainda mais expressivos em desempenho, entretanto menos expressivos em área e potência.

Para trabalhos futuros, destaca-se primeiramente o desenvolvimento dos outros blocos do fluxo de compressão JPEG, tais como a quantização e a codificação Huffman, posteriormente a implementação de um controle *pipeline* maximizando o desempenho, para a verificação da viabilidade, se comparado ao aumento de área e potência, será realmente interessante o aumento do desempenho em detrimento de outras características não funcionais.

Por fim, em um projeto mais avançado, poderia se compreender no mesmo *chip* a compressão e descompressão da imagem, mais uma vez, a análise de resultados se faria necessário para verificar a viabilidade, tendo em vista que o foco da proposta são os sistemas embarcados que possuem geralmente restrições severas de área.

REFERÊNCIAS

WEI, W. An Introduction to Image Compression, 2013.

HUANG, J. The JPEG Standard. 2008.

MIHIR, M.; VIPUL, P.; KAPIL, A. Efficient Progressive JPEG Decoder using JPEG Baseline Hardware.IEEE Second International Conference on Image Information Processing, 2013.

PASTUSZAK, G. A high-performance memory-efficient architecture of the bit-plane coder in JPEG 2000.IEEE International Conference on Multimedia and Expo, 2005.

KORNEL, L. JPEG Compressor, 2013. Disponível em: https://github.com/kornelski/jpeg-compressor. Acessado em 27/09/2018.

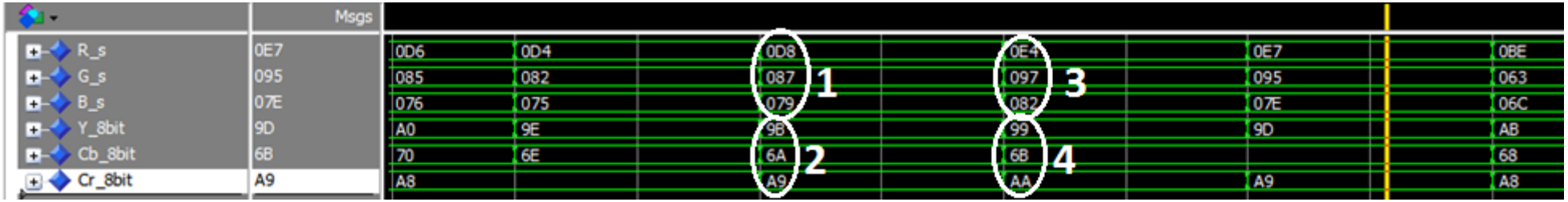
AL-ANI, M. S.; AWAD, F. H.The JPEG Image Compression Algorithm.International Journal of Advances in Engineering &Technology, May 2013.

CHIANG, J.; HUANG, H. New architecture for high throughput-rate real-time 2-D DCT and the VLSI design. In: *Proceedings Ninth Annual IEEE International ASIC Conference and Exhibit, Rochester, p. 219-222, 1996.*

KREPA, M. JPEG Encoder, 2009. Disponível em: https://opencores.org/projects/mkjpeg.

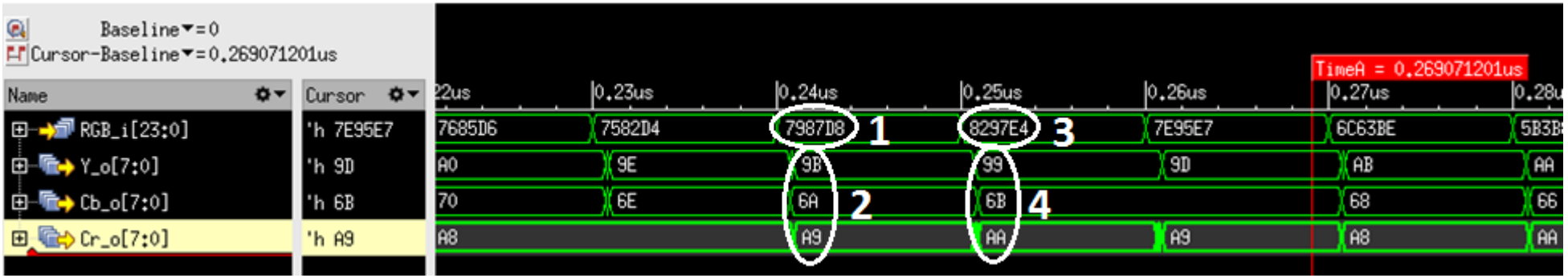
APÊNDICE A – SIMULAÇÃO ENTRE IMPLEMETAÇÕES DO BLOCO RGB_YCBCR

Figura 13 - Simulação da conversão de cores de Krepa (2009) @100 MHz



Fonte: Autores.

Figura 14 - Simulação pós-síntese física da conversão de cores do *chip* JpegEnc @200MHz

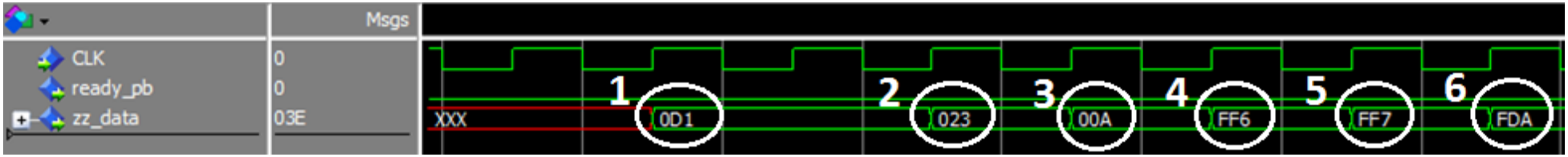


Fonte: Autores.

1 – Entrada dos dados RGB para o bloco de conversão RGB_YCBCR.
2 – Saída dos dados convertidos para o padrão YCbCr.
3 – Nova entrada de dados RGB.
4 – Nova saída de dados YCbCr.

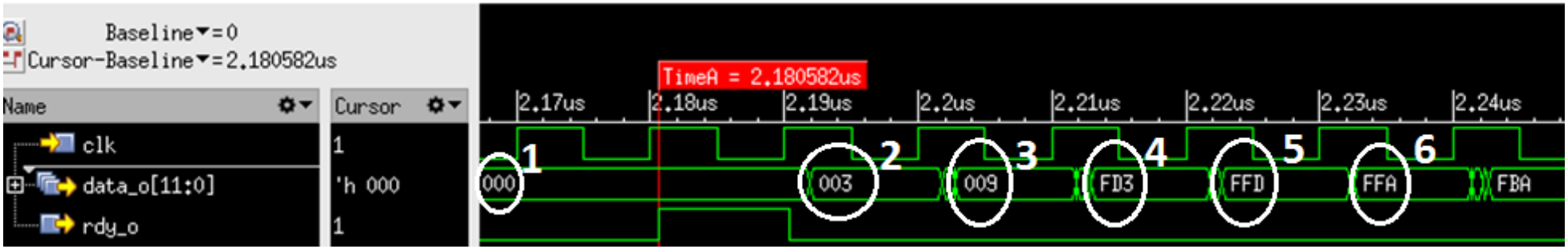
APÊNDICE B – SIMULAÇÃO ENTRE IMPLEMETAÇÕES DO BLOCO DCT2D

Figura 15 - Simulação do DCT de Krepa (2009) @100 MHz



Fonte: Autores.

Figura 16 - Simulação pós-síntese física do DCT do *chip* JpegEnc @200MHz

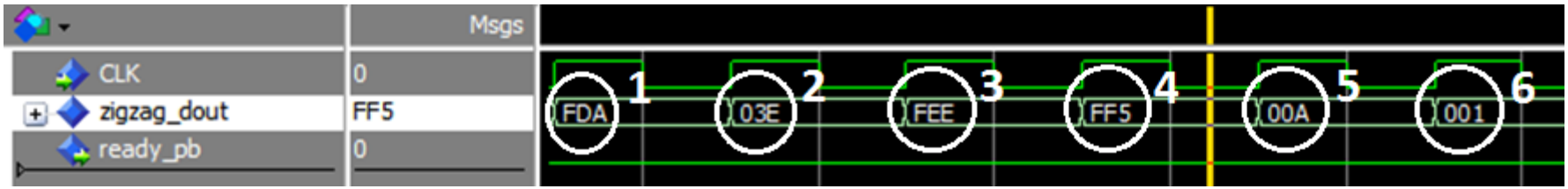


Fonte: Autores.

1 ao 6 – Comparação pós cálculo DCT entre dados da simulação lógica do circuito de referência e os dados da simulação pós-síntese física do JpegEnc.

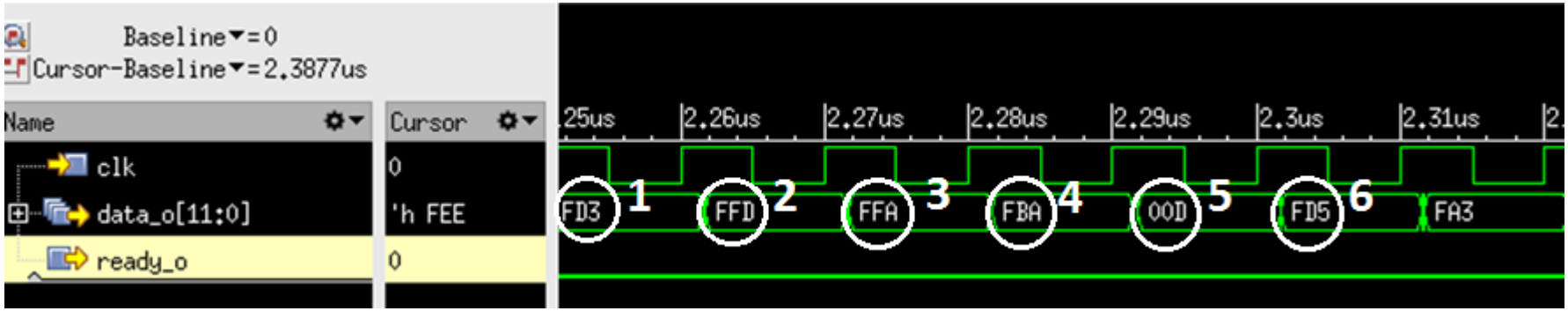
APÊNDICE C – SIMULAÇÃO ENTRE IMPLEMETAÇÕES DO BLOCO ZIGZAG

Figura 17 - Simulação do ZigZag de Krepa (2009) @100 MHz



Fonte: Autores.

Figura 18 - Simulação pós-síntese física do ZigZag do *chip* JpegEnc @200MHz

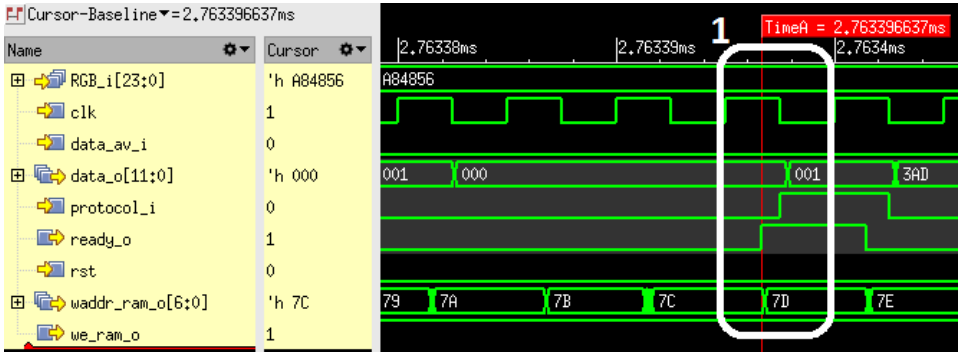


Fonte: Autores.

1 ao 6 – Comparação pós ordenação ZigZag entre dados da simulação lógica do circuito de referência e os dados da simulação pós-síntese física do JpegEnc.

APÊNDICE D – SIMULAÇÃO COMPLETA DO VETOR DE TESTE LENA 512x512

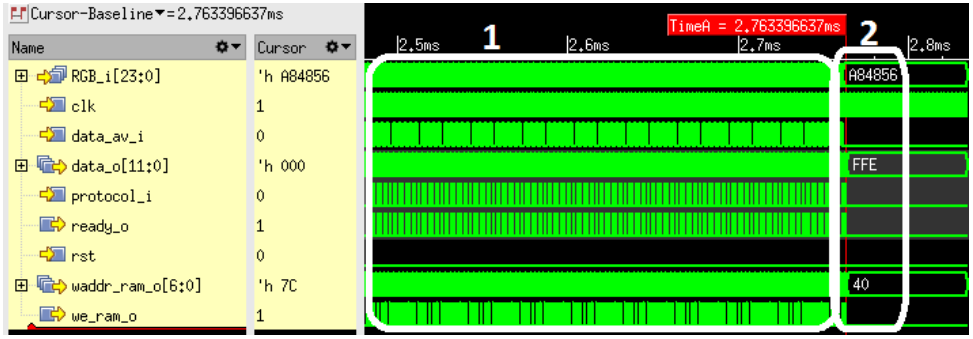
Figura 19 - Simulação pós-síntese física do vetor de teste Lena 512x512 do *chip* JpegEnc @200MHz



Fonte: Autores.

1 – Término do processamento de toda a imagem no instante 2,765 ms da simulação.

Figura 20 - Simulação pós-síntese física completa do vetor de teste Lena 512x512 do *chip* JpegEnc @200MHz



Fonte: Autores.

1 – Período de intensivo processamento de compressão da imagem.

2 – Estabilização dos últimos valores de saída do circuito.

APÊNDICE E – TABELA DE SÍNTESE LÓGICA E SÍNTESE FÍSICA

Tabela V - Resultados de síntese lógica e física para duas aplicações

			Logic Synthesis						*Physical Synthesis*					
freq MHz (clk)	testbench / # vectors	period of sim. (us)	*power* mW	*# cells*		*total area* mm²	# equivalent gates	*timing slack* ps	*power* mW	*# cells*		*total area* mm²	# equivalent gates	*timing slack* ps
			total	seq	comb				*total*	seq	comb			
200	--	--	258,02750	5791	11728	1,4895	98952	7	376,62000	5593	12051	0,988	65636	0,111
	8x8	3,75	207,26000						353,86000					
	64x64	47,2	188,13000						334,04000					

Fonte: Autores.