UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

MATHEUS BERGER OLIVEIRA

PROJETO DE DIPLOMAÇÃO

DESENVOLVIMENTO DE UM CONTROLADOR DE VÍDEO EM VHDL

Porto Alegre
2010

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DESENVOLVIMENTO DE UM CONTROLADOR DE VÍDEO EM VHDL

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para Graduação em Engenharia Elétrica.

ORIENTADOR: Prof. Dr. Altamiro Amadeu Susin

Porto Alegre
2010

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

MATHEUS BERGER OLIVEIRA

DESENVOLVIMENTO DE UM CONTROLADOR DE VÍDEO EM VHDL

Este projeto foi julgado adequado para fazer jus aos créditos da Disciplina de "Projeto de Diplomação", do Departamento de Engenharia Elétrica e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Dr. Altamiro Amadeu Susin, UFRGS

Doutor pelo Institut National Polytechnique de Grenoble, França

Banca Examinadora:

Prof. Dr. Altamiro Amadeu Susin, UFRGS

Doutor pelo Institut National Polytechnique de Grenoble, França.

Prof. Dr. André Borin Soares, UFRGS

Doutor pela Universidade Federal do Rio Grande do Sul, Brasil.

Prof. Dr. Marcelo Götz, UFRGS

Doutor pela Universität Paderborn, Alemanha.

Porto Alegre, julho de 2010.

AGRADECIMENTOS

Agradeço ao LaPSI por possibilitar a realização do projeto oferecendo todos os recursos necessários. Em especial, aos professores: Prof. Dr. Altamiro Amadeu Susin e o Prof. Dr. André Borin Soares, pelos ensinamentos prestados.

Agradeço à Universidade Federal do Rio Grande do Sul e seus professores pela formação, apoio e ensinamentos concedidos.

Agradeço a minha família pelo auxilio e dedicação incondicionais. Finalmente, a todos meus amigos e colegas pela ajuda e ensinamentos prestados ao longo da jornada da minha formação.

RESUMO

O projeto tem como principal objetivo desenvolver em VHDL o estágio de saída de vídeo do decodificador H.264. Como principais funcionalidades do projeto estão à leitura de dados escritos na memória pelo decodificador, a adequação desses dados de maneira a estabelecer uma interface com o D/A de saída e a composição do vídeo através da sobreposição de duas imagens. A plataforma de desenvolvimento utilizada foi o kit XUP V5 (Xilinx).

Palavras-chaves: VHDL, FPGA, processamento de sinais, codec H.264.

ABSTRACT

The project's main objective is developing in VHDL the output stage of a H.264 video decoder. The main features of the project are the transfer of data written into memory by the decoder, the adequacy of these data in order to establish an interface with the D/A output and composition of the video through the superimposition of two images. The development platform used was the kit XUP V5 (Xilinx).

Keywords: VHDL, FPGA, signal processing, codec H.264.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

D/A: Conversor Digital Analógico

DCM: Digital Clock Manager

DCR: Control Register Bus

DDR: Double Data Rate

DFP: Digital Flat Panel

DVI: Digital Visual Interface

FPGA: Field Programmable Gate Array

H.264: Padrão de compressão de vídeo

I2C: Inter-Integrated Circuit

IP: Intellectual Property

JVT: Joint Video Team

LaPSI: Laboratório de Processamento de Sinais e Imagens

LUT: Look Up Table

MPEG: Moving Picture Experts Group

PLB: Processor local bus

PLL: Phase-locked loop

RAM: Random Access Memory

RGB: Red; Green; Blue

SBTVD: Sistema Brasileiro de Televisão Digital

UFRGS: Universidade Federal do Rio Grande do Sul

VGA: Video Graphics Array

VHDL: VHSIC Hardware Description Language

VHSIC: Very-high-speed integrated circuit

XUP: Xilinx University Program

1 INTRODUÇÃO

Mobilizado pelo evento da implantação do Sistema Brasileiro de Televisão Digital (SBTVD) o governo brasileiro realizou uma série de investimentos em pesquisa e desenvolvimento para a sua implantação. Contando com o apoio de algumas universidades brasileiras o SBTVD foi dividido e distribuído para os centros de pesquisa relacionados à área de processamento de sinais e microeletrônica.

À UFRGS coube o desenvolvimento do codificador/decodificador (*Codec*) de vídeo bem como a coordenação da rede H.264 – SBTVD. Essa rede tem como objetivo o desenvolvimento de produtos de interesse nacional na área de decodificação de sinais-fonte para o SBTVD. A rede H.264 – SBTVD reúne diferentes atividades listadas na tabela 1.

Tabela 1 Instituições e atividades realizadas na rede H.264 - SBTVD

Instituição	Atividade da equipe
UFRGS	Coordenação Geral
UFRGS	Decodificador H.264 em HDL
UFRGS	Codificador/decodificador escalável/alternativo
UFRGS	Codificador H.264 em HDL
UFRGS	Codificador H.264 em arquitetura computacional paralela
UFSC	Codificador H.264 em arquitetura computacional distribuída
UNICAMP	Codificador/Decodificador de áudio
USP	Terminal de acesso com decodificadores desenvolvidos
UFRN	Estimador de movimento em HDL
IME	Vídeo estéreo
UFRJ	Melhorias no padrão H.264
UNB	Codificador/decodificador de entropia
UNB	Transcodificação MPEG2 – H.264
CEITEC	Especificação de um ASIC – SoC para o receptor

O grupo de pesquisa da UFRGS encarregado de desenvolver o decodificador de vídeo no padrão H.264 em hardware (HDL) é o Laboratório de Processamento de Sinais e Imagens (LaPSI). Assim, toda a lógica feita é prototipada em kits de desenvolvimento fornecido pelos fabricantes de FPGA. Esses kits possuem uma ampla variedade de periféricos, proporcionando um ambiente completo para o desenvolvimento, por exemplo, de um decodificador. Atualmente os testes de hardware são feitos no kit XUP V2-P desenvolvido

pela empresa Digilent. Havendo a necessidade de mais espaço para a lógica, ou seja, de um FPGA maior, o grupo está portando o que já foi desenvolvido para o kit XUP V5 da Xilinx. O XUP V5 além de possuir um FPGA com maior capacidade tem uma interface de saída de vídeo diferente do kit XUP V2-P.

A necessidade de adequação do decodificador ao estágio de saída de vídeo do kit XUP V5 é o objetivo do projeto de diplomação. A partir dos dados gerados pelo decodificador e escritos na memória, o bloco de lógica deve realizar a leitura dos dados da memória e ajustá-los para o interfaceamento com o conversor D/A de saída. Logo o *Codec* H.264 é encarregado pela geração dos dados (*bitstream*) a serem lidos e exibidos. Para tanto, um estudo da interface entre o FPGA e o D/A do kit foi realizado, a fim de se verificar quais sinais o FPGA deve gerar para o atendimento das necessidades do D/A. A lógica desenvolvida foi feita em VHDL e sua simulação foi realizada com o auxílio do software ModelSim XE III 6.3. A ferramenta de síntese utilizada foi o ISE 10.1 (Xinlix).

2 O PADRÃO DE COMPRESSÃO DE VÍDEO H.264

O H.264 é um padrão de compressão de vídeo baseado no padrão MPEG-4. O padrão foi desenvolvido pela parceria da ITU-T *Video Coding Experts Group* com a ISO/IEC MPEG. Essa parceria veio a ser conhecida como *Joint Video Team* (JVT). A versão final do padrão foi mundialmente publicada no ano de 2003 [3].

O padrão abrange uma ampla gama de aplicações de comunicação de vídeo, como vídeo-telefonia e armazenamento. O projeto do H.264 se beneficia de avanços em técnicas de compressão bem conhecidas (como codificação por transformadas, predição e estimação de movimento), resultando em um sistema de alto desempenho [4].

2.1 Codificador/Decodificador H.264

O padrão H.264 não define um codificador/decodificador. Alternativamente, o padrão define a sintaxe do vídeo codificado (*bitstream*) e o método associado para decodificá-lo. Deste modo, o padrão permite a existência de diferentes implementações, o que garante alternativas de diferenciação entre os desenvolvedores de hardware e software [4].

O codificador H.264 pode ser divido nos seguintes blocos: predição, transformada, quantização e codificação de entropia. De maneira inversa, o decodificador se divide em: transformada inversa, quantização inversa, reconstrução e decodificação de entropia [4].

2.1.1 Codificador

O codificador recebe um quadro F_n que é particionado em unidades de macrobloco (blocos de 16x16 *pixels*). A predição *P* do macrobloco é feita com base em dados previamente codificados, dentro do quadro atual (*Intra*) ou a partir de outros quadros já codificados e transmitidos (*Inter*).

No modo *Intra*, *P* é formado a partir de amostras da vizinhança do macrobloco no quadro atual F_n que foram codificados, decodificados e reconstituídos (uF'_n). Existem até nove modos de predição Intra diferentes, e um modo é escolhido em função de alguma relação de custo (SAD, SADT, etc).

No caso de predição *Inter*, *P* é formado a partir de predições de estimação (EM) e compensação de movimento (CM) de um ou mais quadros de referência (quadros já decodificados).

O macrobloco predito *P* é subtraído do macrobloco atual, resultando no resíduo ou macrobloco de diferença D_n. Este último é transformado através de uma transformada de bloco (TD) e quantizado (Q), resultando em um conjunto de coeficientes quantizados Y^*. A seguir, os coeficientes são reordenados e codificados (codificação de entropia), e junto com informações necessárias para decodificar o macrobloco, formam o bitstream comprimido. Por fim, o bitstream é passado à camada de transporte para a transmissão ou armazenamento.

Os coeficientes quantizados Y^* são decodificados para reconstituição dos quadros (os quadros reconstituídos são utilizados no modo de predição *Intra* quadro). Assim, os coeficientes passam pela quantização inversa (QI) e pela transformada inversa (TI), resultando no macrobloco de resíduo D'_n. Como o processo de quantização introduz perdas, o resíduo D'_n é diferente de D_n.

O macrobloco predito *P* é adicionado à D'_n para criar um macrobloco reconstruído uF'_n. A seguir, um filtro é aplicado para reduzir os efeitos da quantização e um quadro de referência recuperado é criado a partir de uma série de macroblocos F'_n. A figura 1 mostra todas as etapas da codificação.

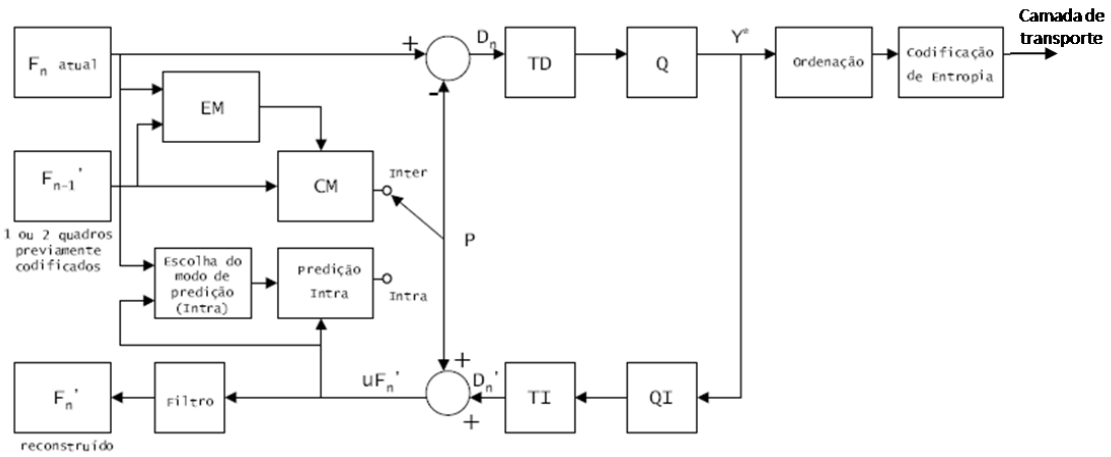

Figura 1 Etapas da codificação no padrão H.264 (Fonte: Teraoka, 2003)

2.1.2 Decodificador

O decodificador recebe um bitstream comprimido e demultiplexado a partir da camada de transporte. Os elementos passam por uma etapa de decodificação de entropia e são reordenados para recuperar os coeficientes quantizados *Y**, que passam pela quantização e transformada inversa, resultando em D'_n. O decodificador é capaz de criar o macrobloco *P* a partir de informações do cabeçalho. *P* é adicionado a D'_n para produzir uF'_n que é então filtrado para criar o macrobloco F'_n.

O propósito do caminho de reconstrução do codificador é garantir que tanto o codificador quanto o decodificador usem quadros de referência idênticos para criar o quadro e predição *P*. Caso as referências para recriar os macroblocos *P* não sejam idênticas, ocorrerá propagação de erro, que resulta no descasamento entre o codificador e o decodificador. A figura 2 ilustra as fases de decodificação.

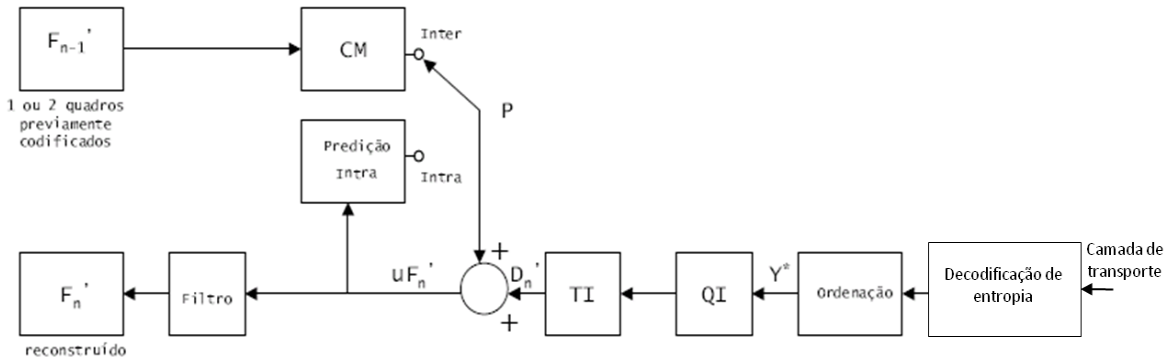

Figura 2 Etapas da decodificação no padrão H.264 (Fonte: Teraoka, 2003)

3 XILINX UNIVERSITY PROGRAM - XUP V5

O kit XUP V5 é uma plataforma de desenvolvimento rica em recursos e com uma ampla opção de interfaces de conexão [6]. É caracterizada principalmente pelo seu FPGA da família Virtex-5 (Xilinx). A figura 3 ilustra o kit.

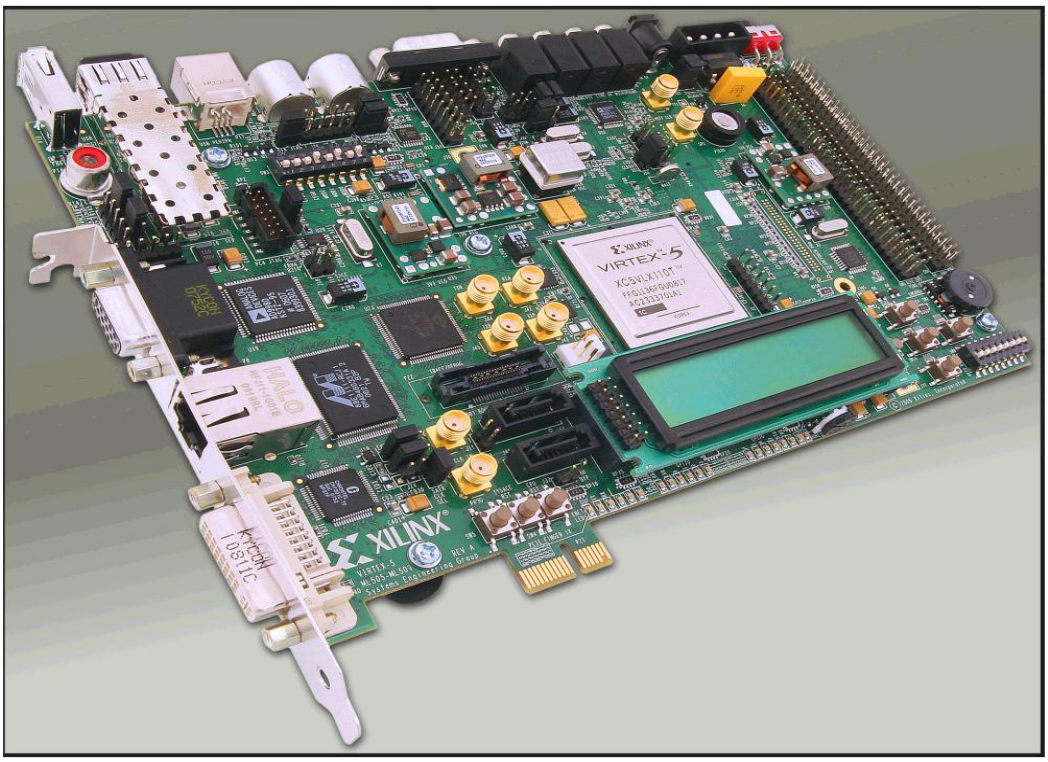

Figura 3 Ilustração do kit de desenvolvimento

Como já mencionado anteriormente, os principais motivos para o uso desse kit de desenvolvimento são a necessidade de um FPGA com maior capacidade e o fato do kit oferecer a opção de saída de vídeo digital. Além dessas funcionalidades cabe ressaltar:

- Duas PROMs (32 Mbytes cada) para armazenamento de configurações;
- 10/100/1000 tri-speed Ethernet PHY, suportando as interfaces MII, GMII, RGMII e SGMII;
- Gerador de relógio (clock) programável;
- Porta RS-232 para comunicação serial.

4 ESPECIFICAÇÕES DO PROJETO

Com a proposta de desenvolver o estágio de saída de vídeo do decodificador H.264, o projeto realiza o interfaceamento do FPGA com o D/A. Assim, a lógica desenvolvida deve ler os dados escritos pelo decodificador na memória e gerar os sinais para o atendimento das necessidades do conversor D/A da saída de vídeo. A figura 4 mostra o diagrama de blocos do projeto bem como sua integração ao decodificador H.264.

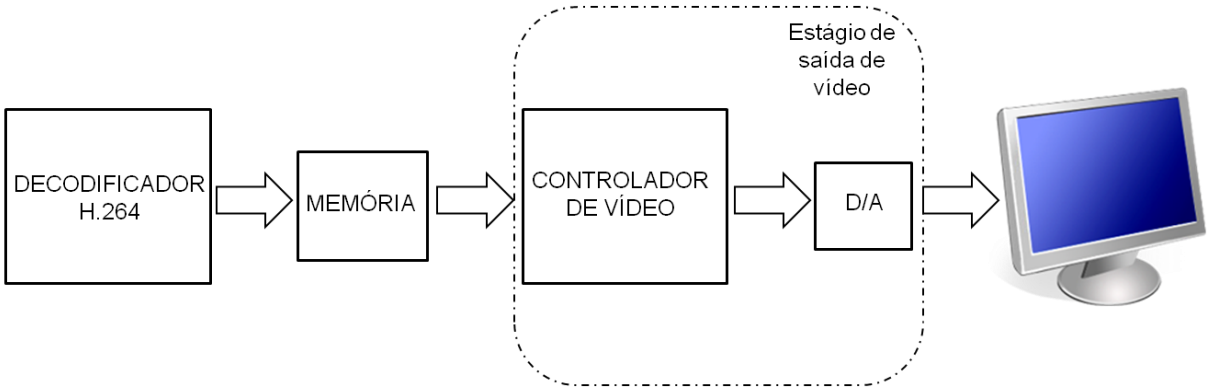

Figura 4 Diagrama de blocos do projeto

4.1 Análise da interface

Analisando o kit de desenvolvimento XUP V5 [6][7], a fim de identificar o D/A utilizado na placa, é possível encontrar o circuito *Chrontel* CH7301C. Esse D/A tem como suas principais características à capacidade de resolução de até 1600x1200 *pixels*, transmissão DVI (*Digital Visual Interface*) em 165M *pixels*/segundo e baixo *jitter* na geração do clock da alta freqüência [2]. O dispositivo possui uma serie de registradores de configuração acessados serialmente através de um protocolo I2C. O CH7301C recebe um sinal de entrada digital, codifica e transmite o dado através do padrão DVI ou DFP (*Digital Flat Panel*). O dispositivo recebe o dado digital com 12-bits de largura. A variabilidade de tensão na porta permite que o *chip* suporte diferentes formatos de dados, incluindo RGB e YCbCr.

Após o estudo da interface do dispositivo se definem quais sinais deverão ser gerados pelo FPGA. Logo, o bloco de lógica deve gerar dados na largura de 12-bits no sistema de cores RGB e transportar esses dados no padrão VGA (*Video Graphics Array*). Após leitura no seu datasheet se verifica a necessidade de transmitir a configuração de uma sequência de registradores através da interface I2C.

4.2 Escolha da interface

Definido os sinais necessários para o desenvolvimento do projeto e, após pesquisa no material do fabricante do kit se encontra um bloco de lógica (IP – *Intelectual Property*) que atende parcialmente as necessidades exigidas [8]. O diagrama de blocos do sistema encontrado está exibido na figura 5.

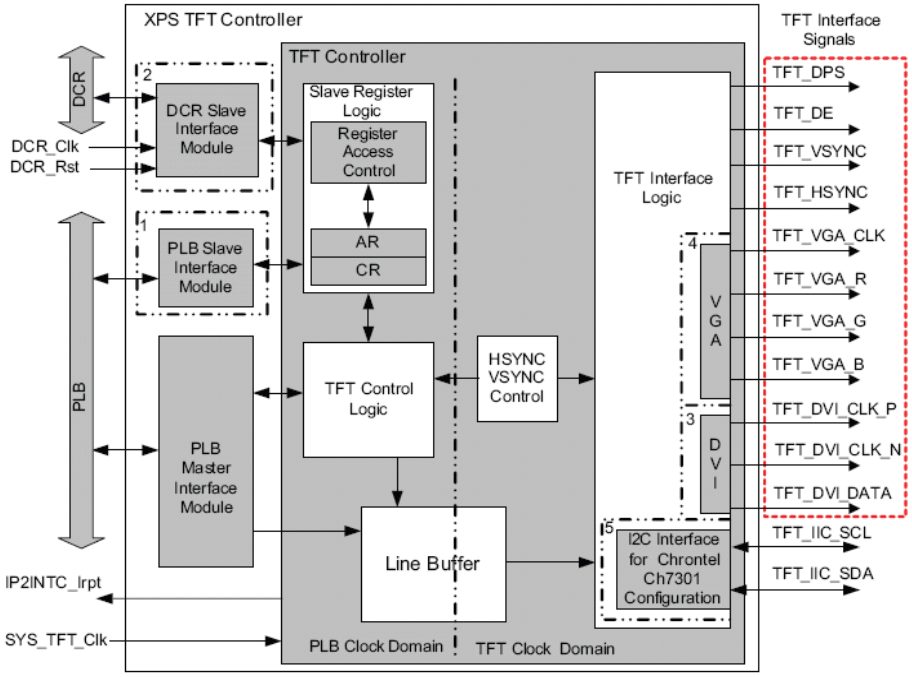

Figura 5 Diagrama de blocos do IP Xilinx (Fonte: TFT controller - Xilinx)

O hardware do bloco de lógica encontrado foi descrito em Verilog. Como a proposta do trabalho é desenvolver o estágio de saída de vídeo em VHDL, todos os códigos encontrados no material do fabricante foram reescritos em VHDL.

É possível verificar na figura 5 uma interface com o processador (barramento PLB) que tem como função o acesso a memória (escrita/leitura). Está presente também um barramento para a configuração do bloco (barramento DCR). O bloco implementa também uma interface com outro tipo de D/A (interface VGA), uma vez que esse IP pode ser utilizado em outros kits de desenvolvimento. Analisando o código de descrição do bloco, fica clara a necessidade de dois domínios de relógio. Visando aumentar a eficiência do sub-bloco de configuração I2C, este possui uma frequência de clock maior que a dos demais sub-blocos do circuito.

Com a proposta de simplificar o projeto, o bloco não terá qualquer forma de controle via processador e os dados estarão fixos nas memórias (não haverá escrita nas memórias). O motivo de os dados estarem fixos nas memórias é que auxilia na depuração do bloco desenvolvido, uma vez que isola o estágio de saída de vídeo do decodificador. Pelo fato de não haver controle via processador fica impossibilitada qualquer forma de acesso do usuário as configurações do D/A, logo as configurações básicas exigidas pelo componente devem ser realizadas pelo FPGA.

Assim, dado o material do IP e as necessidades descritas foi possível reduzir a quantidade de lógica restando apenas os seguintes sub-blocos: memória RAM (foram utilizadas memórias RAM disponíveis dentro do FPGA), gerador de sincronismo, configurador I2C e conversor de largura de dados. Esse último sub-bloco tem a finalidade de realizar a adaptação da largura do dado lido da memória para a largura de dados da entrada do D/A. Assim esse bloco tem a função de converter um dado com largura de 24 bits em dois dados de largura de 12 bits. Na figura 6 é possível observar as ligações do topo do design obtido através da análise do IP da Xilinx.

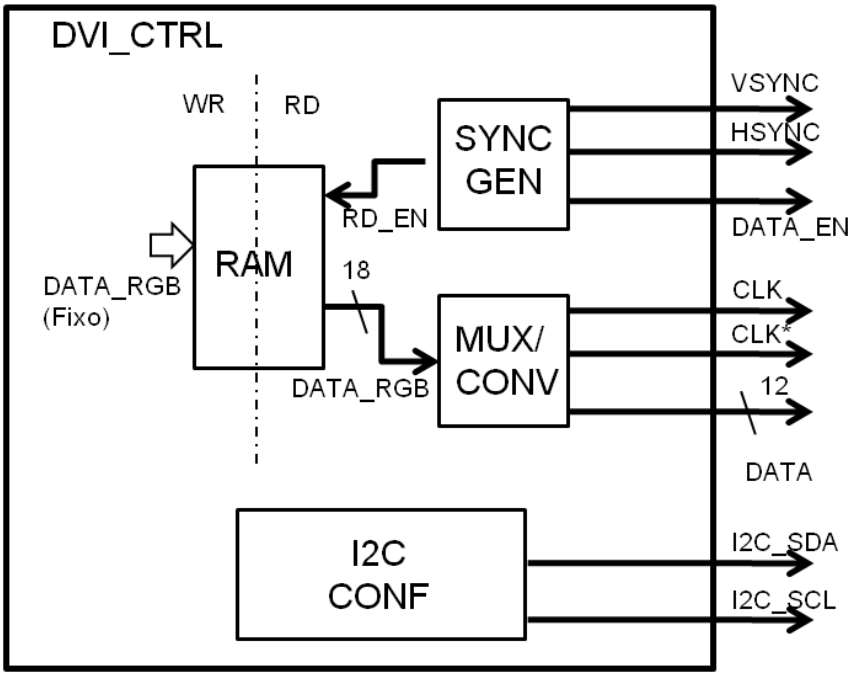

Figura 6 Diagrama de blocos após análise do IP

Baseado nas necessidades exigidas pelo bloco do IP somadas as necessidades do projeto se estabelece as seguintes especificações para o trabalho:

- Interfaceamento e geração dos sinais de sincronismo de vídeo necessários para o atendimento do D/A (sistema de cores RGB);
- Necessidade de acesso aos registradores de configuração do D/A através do protocolo I2C, ou seja, necessidade de um mestre I2C;
- Necessidade de geração de diferentes taxas de relógio, pois o IP da Xilinx necessita de uma freqüência de relógio diferente para o sub-bloco de configuração (bloco I2C);
- Troca dinâmica das resoluções de vídeo: sabendo que quando o projeto estiver integrado ao decodificador este gerará o bitstream a ser exibido junto com a resolução que se deseja exibir a imagem. Logo, o estágio de saída deve ser capaz de trocar sua resolução conforme a necessidade do decodificador;

- Será adicionada a funcionalidade de sobreposição de imagens de forma sincronizada na saída de vídeo, ou seja, a composição da imagem através da leitura de memórias distintas. Para tanto será adotada a metodologia de *Chroma key*, onde um dado padrão deverá indicar o momento da troca de leitura das memórias;
- Garantir uma ordem de inicialização dos sub-blocos do sistema, pois a configuração do D/A deve ser realizada antes do envio dos pulsos de sincronismo de vídeo para o D/A.

5. DESCRIÇÃO DOS BLOCOS DE LÓGICA

A lógica desenvolvida pode ser dividida nos seguintes sub-blocos: memória RAM, gerador de sincronismo de vídeo, configurador I2C, conversor de largura de dados, bloco que realiza a troca dinâmica das resoluções e um gerador de resets.

5.1 Memória RAM (sobreposição de imagens)

Foram descritas em VHDL seis block RAMs com largura de dados de 8 bits e largura de endereço de 15 bits, ou seja, cada RAM possui o tamanho de 32k bytes. Cada memória RAM guarda uma componente de cores do sistema RGB (uma memória para armazenar os valores da componente R outra para os valores da componente G e outra para os valores da componente B) de cada imagem a ser exibida. Como a proposta do projeto é exibir duas imagens sobrepostas são necessárias seis RAMs para o armazenamento dos dados. Assim se pode dividir as memórias em dois bancos (um para cada imagem), sendo cada banco composto por três memórias. As imagens possuem a resolução de176x144 *pixels*. A figura 7 ilustra as ligações desse sub-bloco.

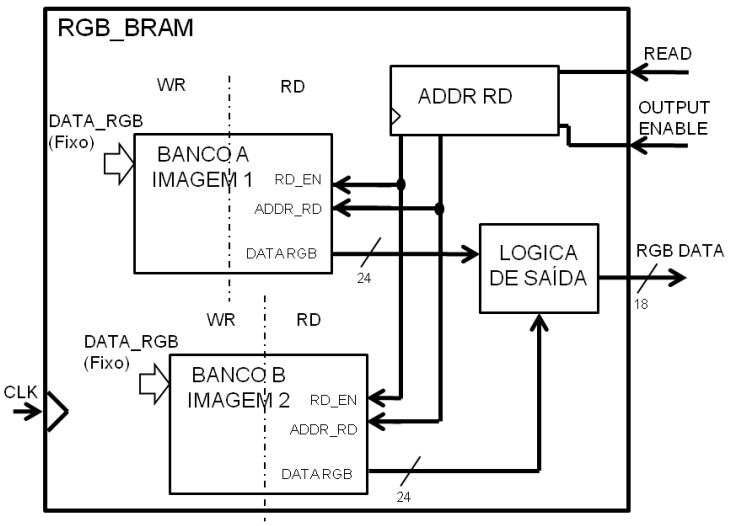

Figura 7 Sub-bloco RGB_BRAM

Como mencionado anteriormente, a função da memória no projeto é o armazenamento do vídeo digitalizado gerado pelo decodificador H.264. Entretanto, a fim de simplificar a depuração do projeto nada será escrito nas memórias, logo os dados lidos das memórias estarão fixos desde as suas inicializações. Assim o bloco de lógica encarregado de instanciar as RAMs deve somente prover o acesso de leitura gerando o endereçamento das memórias e compondo a imagem através da leitura dos dois bancos. Os pedidos de leitura das memórias são feitos conforme a exigência do sub-bloco gerador de sincronismo. Dessa forma a habilitação da leitura somente é permitida durante o período de tempo em que os dados devem ser exibidos no vídeo, logo o controle da habilitação de leitura das memórias é feito pelo gerador de sincronismo.

O fluxo dos dados (imagem) a serem exibidos no vídeo é resultado da ação de três sub-blocos: memória RAM, gerador de sincronismo de vídeo, e conversor de largura de dados (os dois últimos sub-blocos serão discutidos nos próximos itens). Assim a memória RAM tem a função de armazenar a imagem a ser exibida no vídeo, o gerador de sincronismo tem a função de habilitar da leitura dessa memória e o conversor recebe o dado vindo da memória e altera sua largura (quantidade de bits) a fim de realizar a interface entre o FPGA e o D/A.

A lógica de saída é capaz de mapear o quadro 640x480 indicando a posição dentro desse quadro em que a imagem contida na RAM deva ser exibida. Assim para compor a imagem através da leitura de memórias distintas, a lógica de saída deve ler e exibir o dado presente em um dos bancos de memória até que seja identificado um padrão que indique o momento de ler e exibir o dado presente no outro banco. Ou seja, dadas duas memórias: uma contendo um quadro vermelho com uma janela verde e outra somente com um quadro azul. Seja verde o padrão que indique o momento de troca de leitura de memória. Assim, a lógica de saída ao identificar o padrão verde deverá exibir no vídeo o dado presente na outra

memória, ou seja, ao invés de exibir um quadro vermelho com uma janela verde deverá exibir um quadro vermelho com uma janela azul. A figura 8 ilustra esse processo.

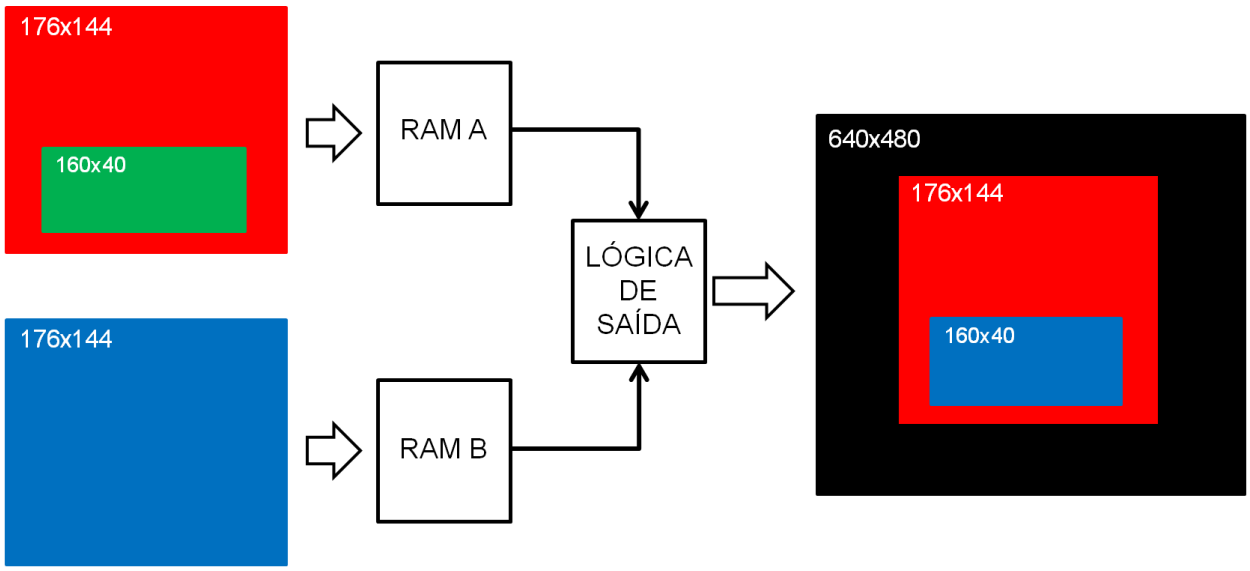

Figura 8 Ilustração do processo de composição da imagem

A forma adotada para sincronizar a leitura entre os dois bancos de memória foi utilizar a mesma geração de endereços e habilitação de leitura para ambos os bancos, sendo assim ambos os bancos são acessados no mesmo momento. Outra abordagem adotada no projeto foi desprezar os dois bits menos significativos de cada componente de cor RGB, com o propósito de evitar a saturação das cores exibidas no vídeo. Essa abordagem de se desprezar os dois bits menos significativos foi herdada dos blocos de lógica do IP da Xilinx [8]. A opção de se manter a largura de dados das memórias de oito bits se deve pela maior facilidade para geração dos valores das componentes RGB, ou seja, já existia uma ferramenta capaz de gerar os dados das componentes RGB na largura de oito bits. Tal abordagem não acarretará maior consumo de unidades elementares de memória, pois a ferramenta de síntese lógica irá interpretar as memórias descritas com largura de dados de seis bits (essa simplificação será detalhada no item 7.1).

5.2 Gerador de sincronismo de vídeo

Todas as demonstrações feitas nesse tópico se referem à temporização da resolução 640x480. A geração da temporização de vídeo é baseada em contadores, ou seja, os contadores definem os tempos em que cada sinal de vídeo deve permanecer em zero ou em um. Assim basta mudar os limites de contagem bem como a frequência do clock para obtermos os sinais de sincronia de vídeo em outra resolução [8].

Esse sub-bloco é responsável por gerar toda a sincronização de vídeo necessária, incluindo os tempos de back porch e front porch do Hsync e Vsync. Os tempos dos sinais de sincronização para mostrar um quadro de 640x480 pixels usando um clock de 25MHz são mostrados na tabela 2. O controlador tem 16,8ms para exibir cada quadro de 640x480 a 60 Hz (taxa de atualização). Por isso, para mostrar o quadro completo não se pode atualizar o dado presente na memória antes desse tempo.

Tabela 2 Tempos necessários para exibição de um quadro 640x480 (Fonte: TFT controller - Xilinx)

Symbol	Parameter	Vertical Sync			Horizontal Sync	
		Time	Clocks	Lines	Time	Clocks
T_{PULSE}	Sync pulse time	16.8 ms	420000	525	32 μs	800
T_{Disp}	Display time	15.4 ms	384000	480	25.6 μs	640
T_{PW}	Pulse width time	64 μs	1600	2	3.84 μs	96
T_{BP}	Back porch time	992 μs	24800	31	1.92 μs	48
T_{FP}	Front porch time	384 μs	9600	12	640 ns	16

5.2.1 Hsync (sincronismo horizontal)

O sinal de Hsync é ativo em nível baixo e seu período completo é de 800 ciclos de clock. Durante um período Hsync os pixels são transmitidos, ou seja, considerados válidos,

quando o sinal de DE (data enable) estiver no nível alto. O sinal DE deve ficar no nível alto durante 640 ciclos de clock. A duração do pulso de Hsync deve ser de 96 ciclos de clock. O intervalo de tempo entre o pulso de Hsync e o início da transmissão dos dados (subida do sinal DE) é chamado de back porch e tem duração de 48 ciclos de clock. Já o intervalo entre o fim da transmissão (descida do sinal DE) e o início de um novo pulso de Hsync é chamado de front porch e tem duração de 16 ciclos de clock. O diagrama de tempos do sincronismo horizontal é mostrado na figura 9.

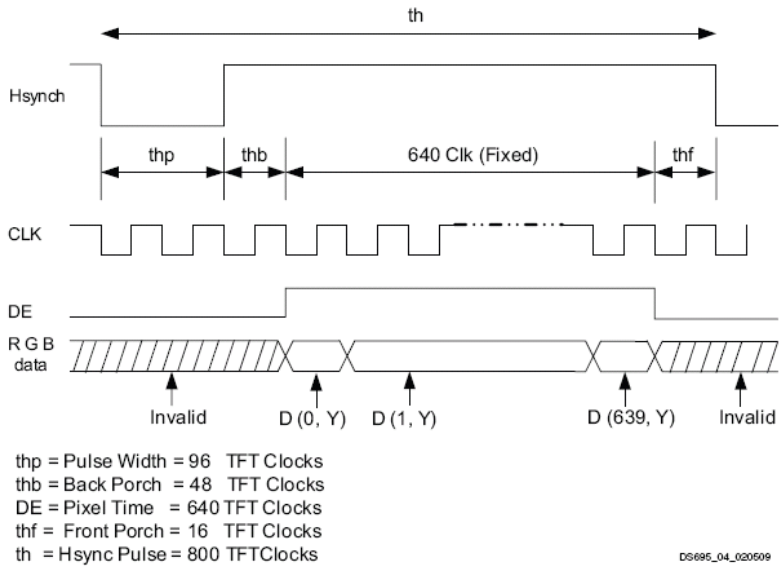

Figura 9 Sincronismo horizontal (Fonte: TFT controller - Xilinx)

5.2.2 Vsync (sincronismo vertical)

O sinal Vsync é ativo em nível baixo e seu período completo é de 525 ciclos de Hsync. O sinal Vsync deve ficar em nível baixo durante 2 ciclos de Hsync seguido de 480 ciclos de Hsync em nível alto (intervalo em que os dados estão validos). O intervalo de back porch é de 31 ciclos de Hsync e o intervalo de front porch é de 12 ciclos de Hsync. O diagrama de tempos do sincronismo vertical é ilustrado pela figura 10.

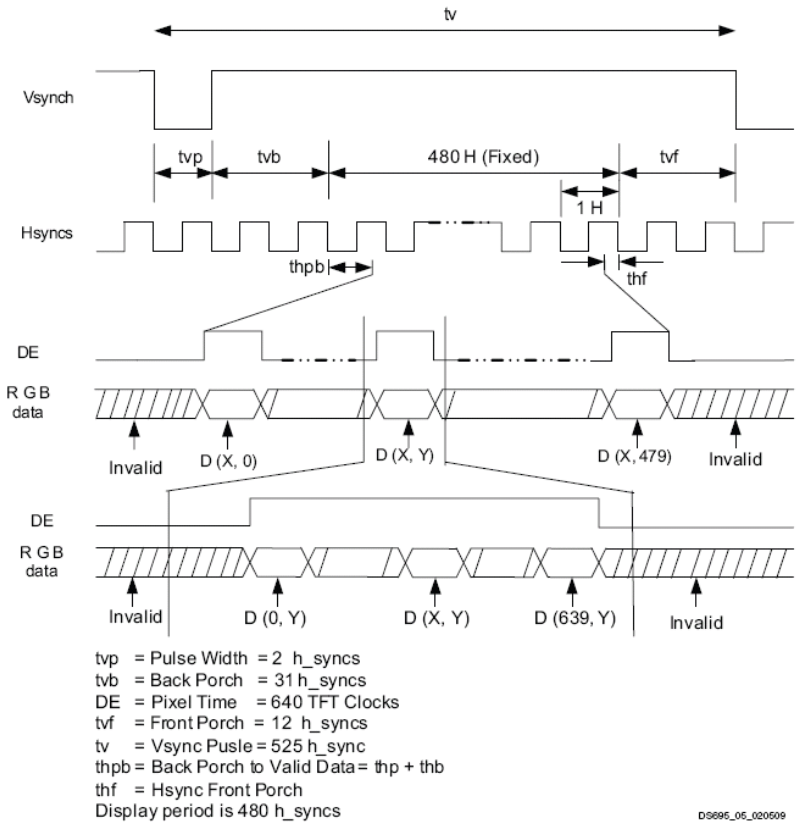

Figura 10 Sincronismo Vertical (Fonte: TFT controller - Xilinx)

5.3 Configurador I2C

Havendo a necessidade de acesso aos registradores de configuração do D/A e como sistema não contempla uma interface com processador, esse sub-bloco tem a função de realizar a sequência de configuração do D/A através de uma interface I2C [8]. Cabe ressaltar que o sub-bloco somente realiza a operação de escrita dos registradores. A figura 11 é uma ilustração das formas de onda para se estabelecer uma comunicação serial com o D/A.

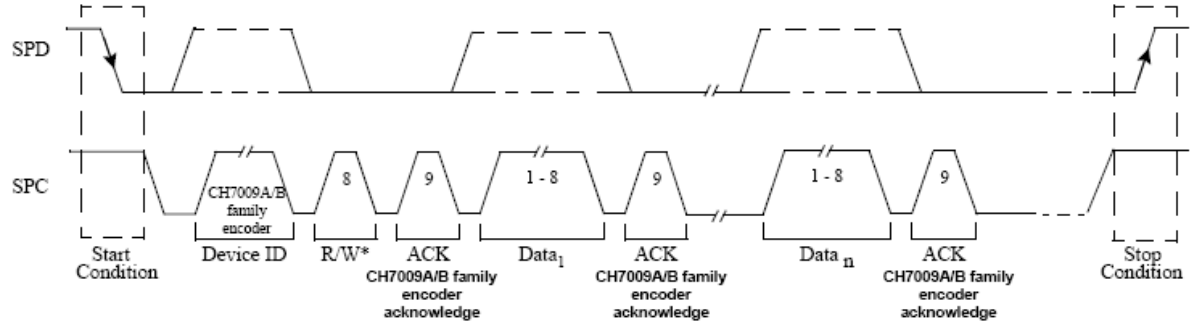

Figura 11 Formas de onda para acesso serial ao D/A (Fonte: AN-41 Chrontel)

O sub-bloco gera os sinais do protocolo I2C e configura o D/A de uma maneira fixa, sendo assim se o usuário desejar uma configuração diferente deverá resintetizar o código VHDL. Dessa forma, o sub-bloco é constituído, basicamente, por uma máquina de estados que simula um mestre I2C e envia a sequência de dados necessária para a configuração do D/A.

Com o propósito de acelerar a configuração e tendo em vista que o configurador I2C é independente dos outros sub-blocos do sistema, o clock desse sub-bloco é mais rápido que os dos demais sub-blocos. Sendo assim, o configurador I2C possui um clock de 200MHz enquanto o resto do sistema possui um relógio que pode variar entre 25MHz à 108MHz.

Analisando o datasheet do D/A é possível identificar a sequência básica de configuração exigida pelo componente [2]. Logo identificada à sequência de dados a ser enviada basta colocá-la no código VHDL através de constantes que o bloco fará a transmissão através do barramento I2C. A tabela 3 mostra a sequência de configuração exigida pela componente.

Tabela 3 Configuração básica do Chrontel CH7301 (Fonte: TFT controller - Xilinx)

Register Address (hex)	Register Name	Configuration Data (hex)	Access	Description
49	PM	C0	Write	Power Management Register
21	DC	09	Write	DAC Control Register
33	TPCP	08	Write	PLL Charge Pump Control Register
34	TPD	16	Write	PLL Divider Register
36	TPF	60	Write	PLL Filter Register

O primeiro registrador é encarregado de habilitar a alimentação do circuito [1][2]. Logo sem o acesso a esse registrador o D/A não funcionará. O segundo registrador tem a função de configurar o modo operação do D/A, ou seja, se o D/A opera no modo digital ou analógico. Os três últimos registradores são uma recomendação do datasheet e tem a função de configurar o *PLL* de saída. Esse *PLL* diminui o *jitter* nos sinais de saída do D/A.

5.4 Conversor de largura de dados

Esse sub-bloco tem a função de ajustar a largura dos dados para a saída [8]. Como é possível verificar, a saída dos dados lidos da memória possui uma largura de 18 bits. E analisando o datasheet do D/A é possível perceber que sua interface de dados tem largura de 12 bits. Assim, o dado lido da memória (no sistema RGB) com 18 bits de largura é convertido pelo sub-bloco para um dado de 24 bits de largura através da inserção de zeros entre os valores RGB (tabela 4). Após essa conversão de largura os dados são amostrados em um flip-flop DDR (Double Data Rate). Com isso os dados são amostrados em ambas as bordas de relógio. A figura 12 mostra o diagrama de tempos dessa conversão.

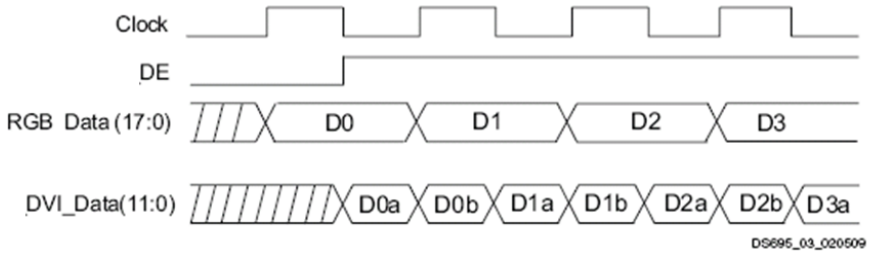

Figura 12 Diagrama de tempos da conversão de largura de dados (Fonte: TFT controller - Xilinx)

Tabela 4 Conversão da largura de dados (Fonte: TFT controller – Xilinx)

DVI Data	Data A	Data B
DVI_DATA[0]	G[2]	0
DVI_DATA[1]	G[3]	0
DVI_DATA[2]	G[4]	B[0]
DVI_DATA[3]	G[5]	B[1]
DVI_DATA[4]	0	B[2]
DVI_DATA[5]	0	B[3]
DVI_DATA[6]	R[0]	B[4]
DVI_DATA[7]	R[1]	B[5]
DVI_DATA[8]	R[2]	0
DVI_DATA[9]	R[3]	0
DVI_DATA[10]	R[4]	G[0]
DVI_DATA[11]	R[5]	G[1]

5.5 Troca dinâmica das resoluções (gerador de relógios)

Conforme já mencionado o sistema exige diferentes taxas de relógio para cada sub-bloco que o compõe. Além disso, o clock do sub-bloco responsável pela geração dos sinais de sincronismo de vídeo deve ter sua taxa atualizada conforme a resolução requerida.

Considerando as várias possibilidades de relógio que o kit de desenvolvimento oferece e as necessidades exigidas pelo sistema, se fez a escolha da utilização do clock de 100MHz (originado por um oscilador) [6][7] e derivar as demais frequências de clock utilizando o DCM. O DCM é um dispositivo que é capaz de gerar frequências de clock múltiplas e em

fase com o clock de entrada. O DCM é um componente oferecido pelo fabricante do FPGA, cabendo ao usuário somente configurá-lo conforme sua necessidade.

O sub-bloco é composto basicamente por DCMs e dois multiplexadores. Assim os DCMs têm a função de gerar as diferentes taxas de relógio e os multiplexadores têm a função de selecionar os parâmetros para os demais sub-blocos do sistema. A figura 13 mostra as ligações do sub-bloco.

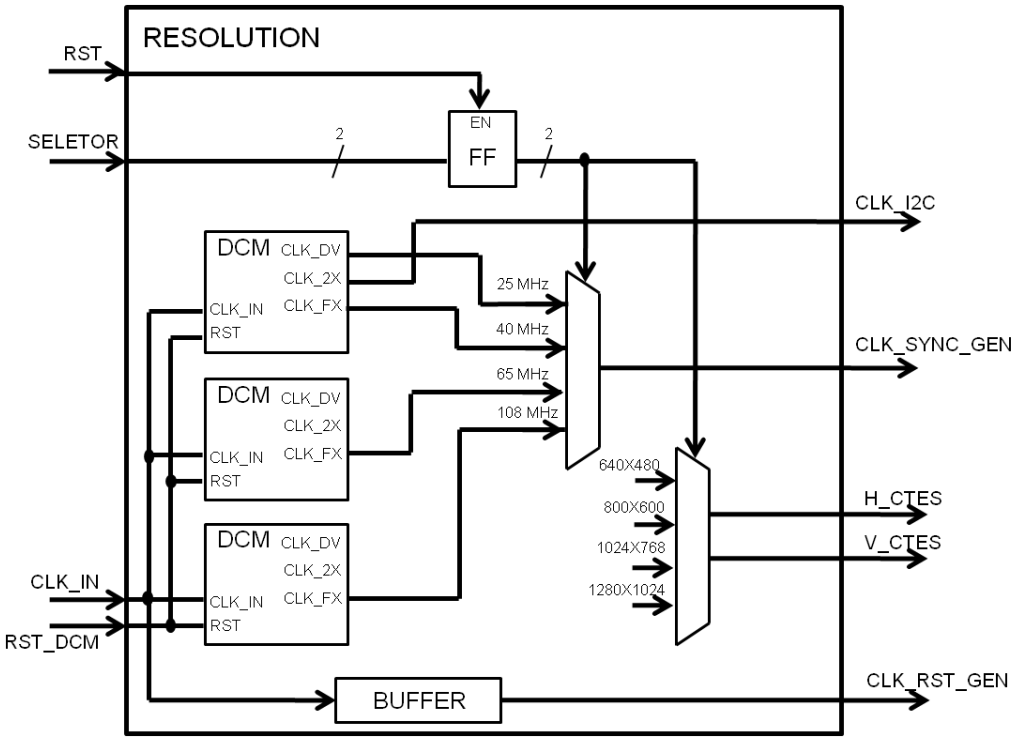

Figura 13 Sub-bloco RESOLUTION

As resoluções que o sub-bloco implementa são as seguintes: 640x480, 800x600, 1024x768, 1280x1024. A maneira encontrada para demonstrar sua funcionalidade foi através de chaves (dip switches) [6][7], tendo em vista que o projeto ainda não vai ser integrado ao decodificador. Uma vez que o usuário deseje trocar a resolução do vídeo exibido, esse deve atuar nas chaves e aplicar o reset do sistema logo após. Assim, após o reset um novo valor é repassado para os seletores dos multiplexadores fazendo com que a resolução mude.

Analisando a figura 13 se percebe que um dos multiplexadores é responsável por repassar ao sub-bloco gerador de sincronismo o clock conforme a resolução configurada. Já o segundo multiplexador tem a função repassar ao sub-bloco gerador de sincronismo os novos limites de contagem. Como já mencionado, a geração da temporização do vídeo é baseada em contadores que devem ter seus limites de contagem atualizado conforme a resolução. A tabelas 5 define os valores dos limites de contagem bem como a frequência de clock exigida para cada resolução implementada. Os valores presentes na tabela 5 são definidos pelo padrão VGA e foram encontrados após pesquisa bibliográfica [5].

Tabela 5 Limites de contagem e frequência de clock para cada resolução

RESOLUÇÃO	640x480		800x600	
PARÂMETRO	VERTICAL SYNC	HORIZONTAL SYNC	VERTICAL SYNC	HORIZONTAL SYNC
	LINHAS	CLOCKs	LINHAS	CLOCKs
PULSO	2	96	4	128
ESPERA ANTES	31	48	23	88
DATA VALID	480	640	600	800
ESPERA DEPOIS	11	16	1	40
FREQ DE CLOCK	25 MHz		40 MHz	

RESOLUÇÃO	1024x768		1280x1024	
PARÂMETRO	VERTICAL SYNC	HORIZONTAL SYNC	VERTICAL SYNC	HORIZONTAL SYNC
	LINHAS	CLOCKs	LINHAS	CLOCKs
PULSO	6	136	3	112
ESPERA ANTES	29	160	38	248
DATA VALID	768	1024	1024	1280
ESPERA DEPOIS	3	24	1	48
FREQ DE CLOCK	65 MHz		108 MHz	

As constantes “ESPERA ANTES” e “ESPERA DEPOIS” se referem aos tempos de espera antes e depois do dado ser validado, respectivamente. Esses termos são equivalentes aos termos “back porch” e “front porch” utilizados no item 5.2.

Esse sub-bloco se encarrega por fornecer também o relógio de 200MHz para o sub-bloco configurador I2C assim como fornecer o clock de 100MHz para o sub-bloco gerador de resets.

5.6 Gerador de resets

A geração de resets para o sistema se torna um ponto crítico, uma vez que os sub-blocos necessitam de uma sequência correta de inicialização. Esse sub-bloco se encarrega por gerar todos os resets utilizados no sistema na sequência correta e de maneira sincronizada com o clock de cada sub-bloco. Partindo do pulso de reset recebido de um botão do kit o sub-bloco deve aplicar o reset da maneira mostrada na figura 14.

Figura 14 Seqüência de aplicação dos resets do sistema

A justificativa para a sequência da figura 14 ser seguida é que dado um pulso de reset o sub-bloco DCM demora algum tempo para que em suas saídas tenha um clock de qualidade, sendo assim até que o DCM se estabilize os sub-blocos que dependem de suas saídas devem permanecer com o reset ativo. Após a estabilização do DCM o próximo sub-bloco a ter seu reset desativado é o configurador I2C. Tendo em vista que todo o processo de configuração leva 6,5ms é conveniente manter o gerador de sincronismo e a memória com o reset ativo, uma vez que sem que o D/A esteja configurado a interface não está pronta para trafegar os

dados. Por fim se desativa o reset do gerador de sincronismo e da memória. A escolha do delay entre cada retirada de reset ser de 7ms se deve pelo fato de que a configuração do D/A leva 6,5ms, logo garante que o evento mais lento se estabeleça.

5.7 Topo do design

A figura 15 ilustra todas as ligações entre os sub-blocos do sistema.

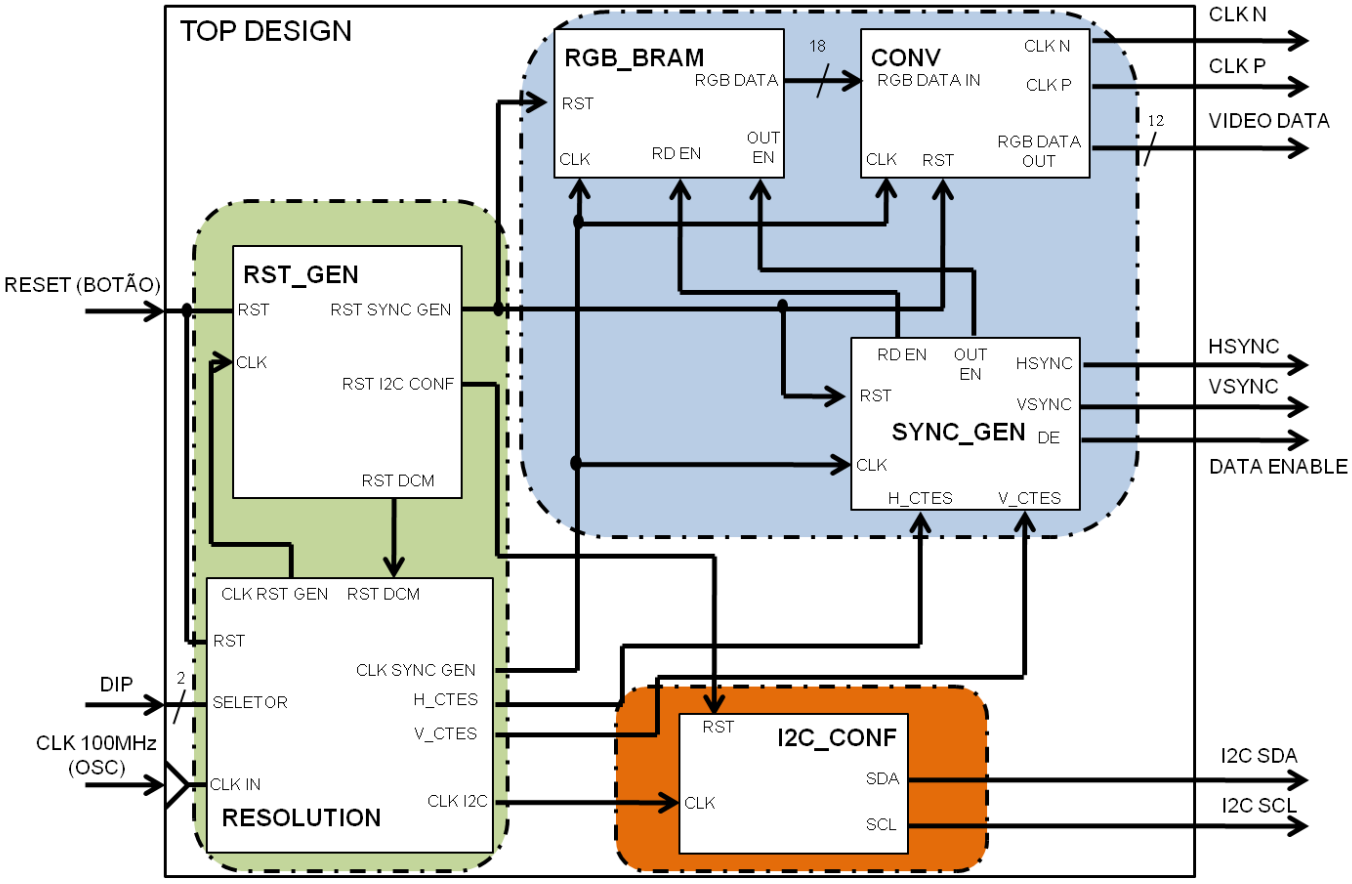

Figura 15 Ligações do topo do sistema. Em azul os sub-blocos responsáveis pelo fluxo de dados; Em verde a parte de controle do sistema; Em laranja sub-bloco de configuração

Analisando a figura 15 se pode perceber a interface do bloco de lógica desenvolvido com o mundo externo. O bloco necessita apenas de um clock mestre, um reset e as chaves para a escolha da resolução. Suas saídas são a interface de dados com o D/A bem como a interface de configuração I2C.

6. SIMULAÇÕES

As simulações realizadas no projeto foram obtidas com o auxílio do software ModelSim XE III 6.3 (versão para estudante). Este item se propõe a estabelecer a verificação das especificações do projeto, bem como cumprir com uma etapa fundamental no processo de desenvolvimento de um sistema digital: a simulação.

As simulações foram realizadas nos sub-blocos em que há a necessidade de visualização do atendimento de temporização ou a necessidade de visualização de alguma característica na forma de onda.

6.1 Gerador de sincronismo

As simulações desse sub-bloco têm por objetivo a verificação da temporização exigida para a sincronização do vídeo. Assim como no item 5.2, a temporização analisada se refere à resolução 640x480.

6.1.1 Hsync (sincronismo horizontal)

A figura 16 ilustra os timings de para a forma de onda de HSYNC obtidos pela simulação.

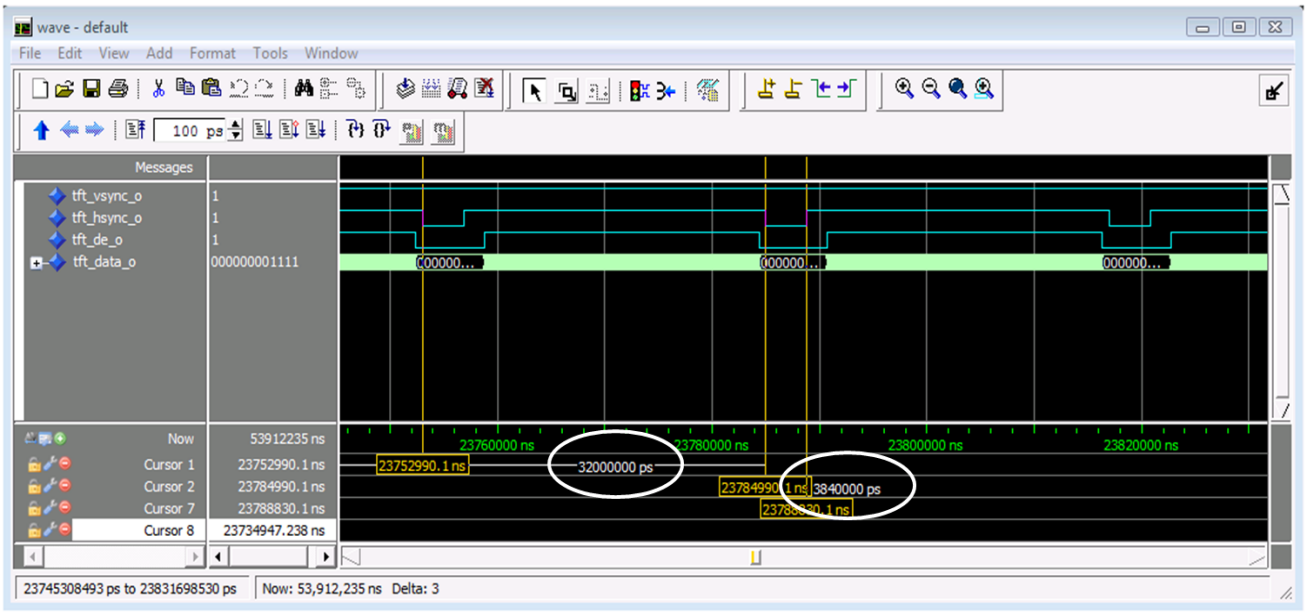

Figura 16 Verificação do período e largura de pulso de uma linha

Analisando a figura 16 é possível verificar o atendimento dos requisitos de timing exigidos pela tabela 2, onde o período de duração de uma linha horizontal deve ser 32 µs e a largura do pulso de sincronismo deve ser 3,84 µs.

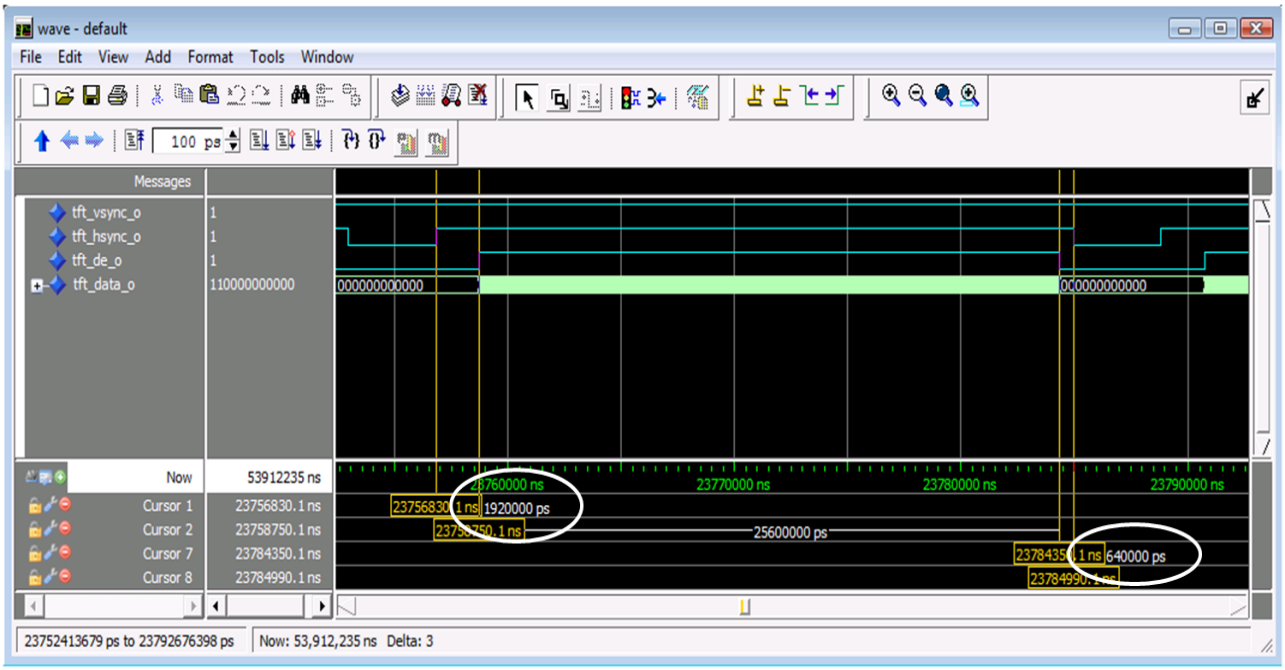

Figura 17 Verificação do timing de back porch e front porch para sincronismo horizontal

Na figura 17 fica claro o atendimento dos tempos de *back porch* e *front porch*, que pela tabela 2 devem ser respectivamente 1,92 µs e 640 ns.

6.1.2 Vsync (sincronismo vertical)

A figura 18 mostra a forma de onda para um período de sincronismo vertical, ou seja, um quadro completo.

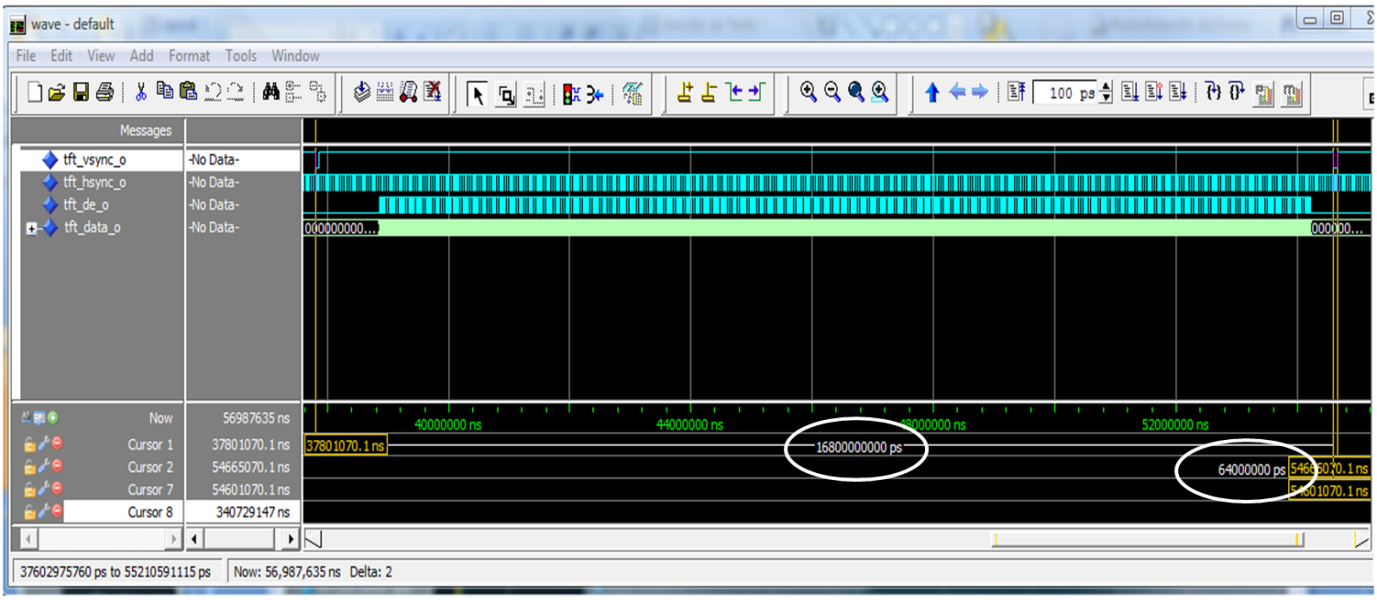

Figura 18 Verificação do timing de um período de sincronismo vertical

Observando a figura 18 se verifica o atendimento do timing. Pela tabela 2, o período total de duração do sincronismo vertical é de 16,8 ms bem como a largura do pulso de sincronismo é de 64 µs.

As figuras 19 e 20 mostram, respectivamente, em detalhe a temporização de back porch e front porch para o sincronismo vertical. A figura 20 também mostra em detalhe a largura de pulso do sincronismo vertical.

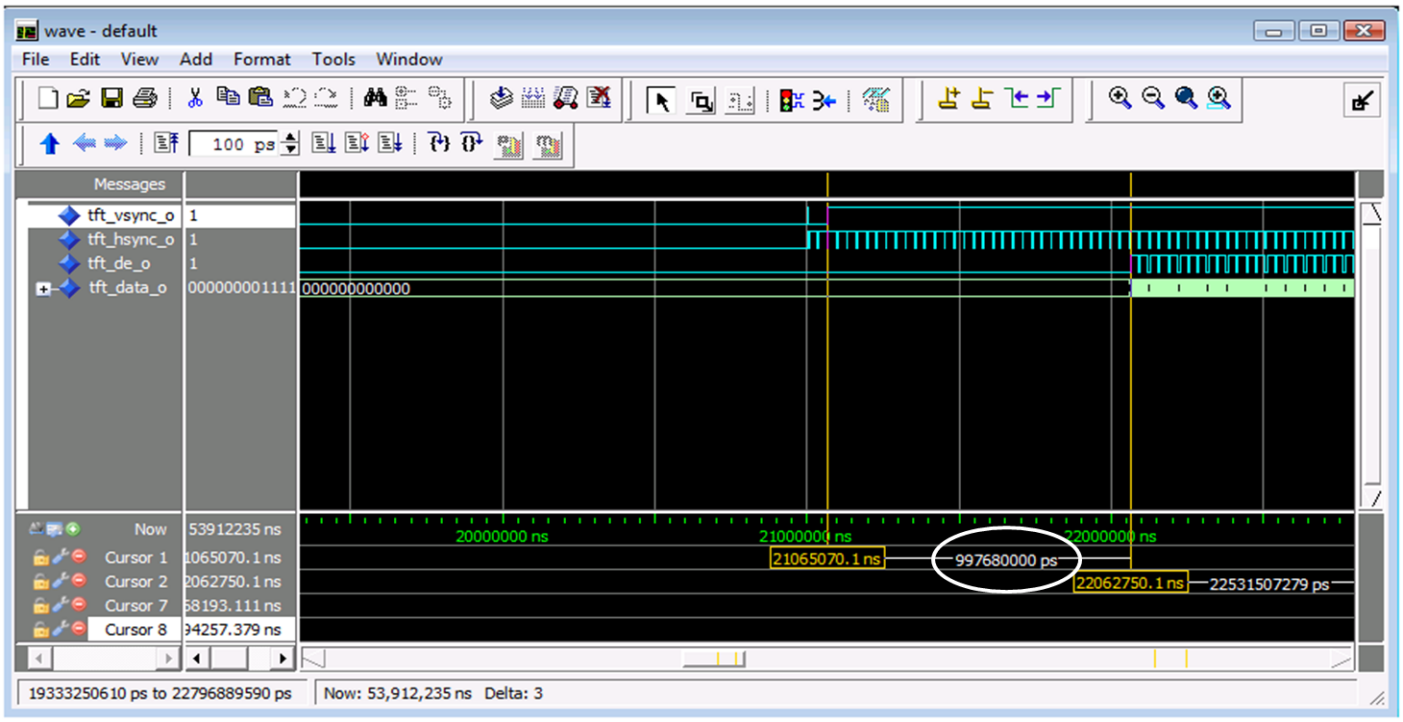

Figura 19 Verificação do timing de back porch para sincronismo vertical

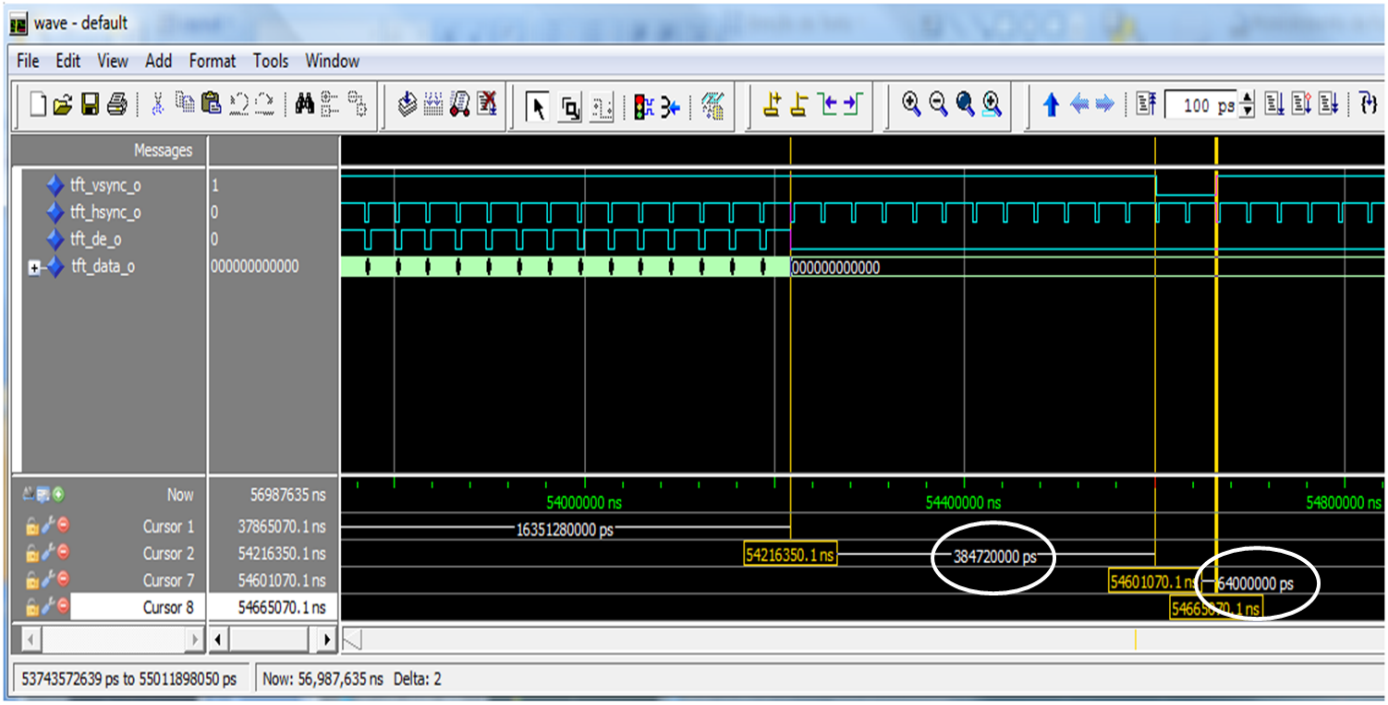

Figura 20 Verificação do timing de front porch para sincronismo vertical

Pela tabela 2 se observa os tempos de back porch e front porch sendo 992 µs e 384 µs, respectivamente. A pequena diferença encontrada entre o resultado da simulação e resultado dado pela tabela se deve a algum ajuste na resolução do simulador.

6.2 Conversor de largura de dados

Na figura 21 é possível observar a saída de dados para o D/A. Como mencionado anteriormente a saída deve ser amostrada em ambas as bordas de relógio. Na figura 21 fica clara essa funcionalidade.

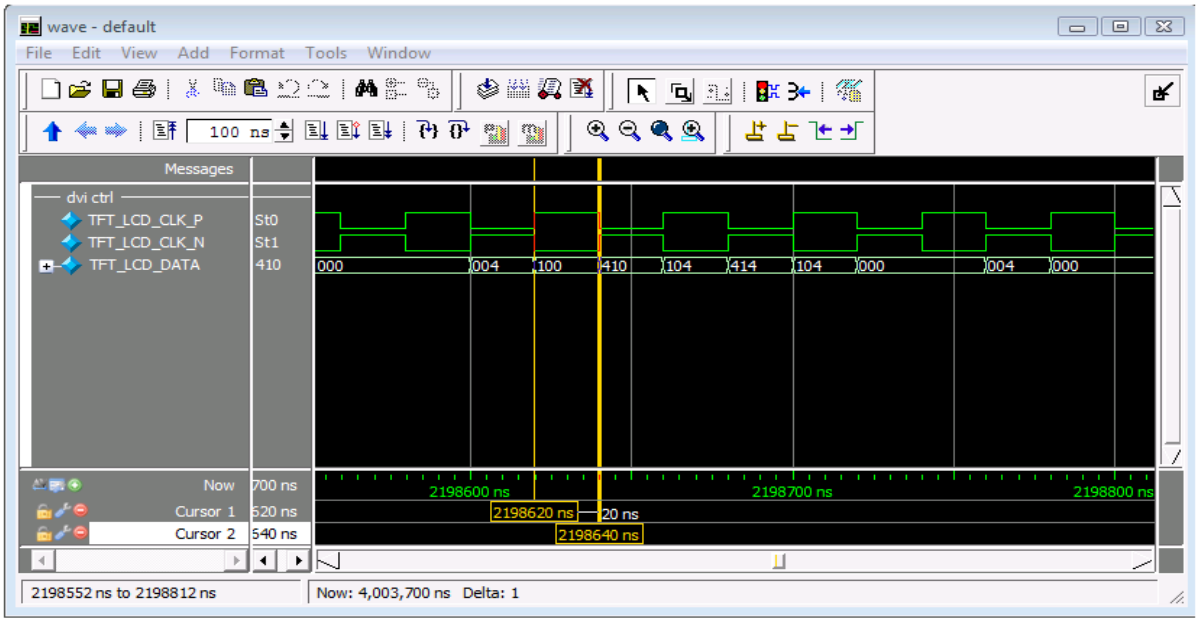

Figura 21 Forma de onda da saída de dados

6.3 Configurador I2C

Na figura 22 se pode verificar o atendimento do protocolo I2C e o envio dos dados para a configuração básica do D/A. Como já mencionado no item 5.6, se pode observar que o tempo de duração da configuração é de aproximadamente 6,5ms.

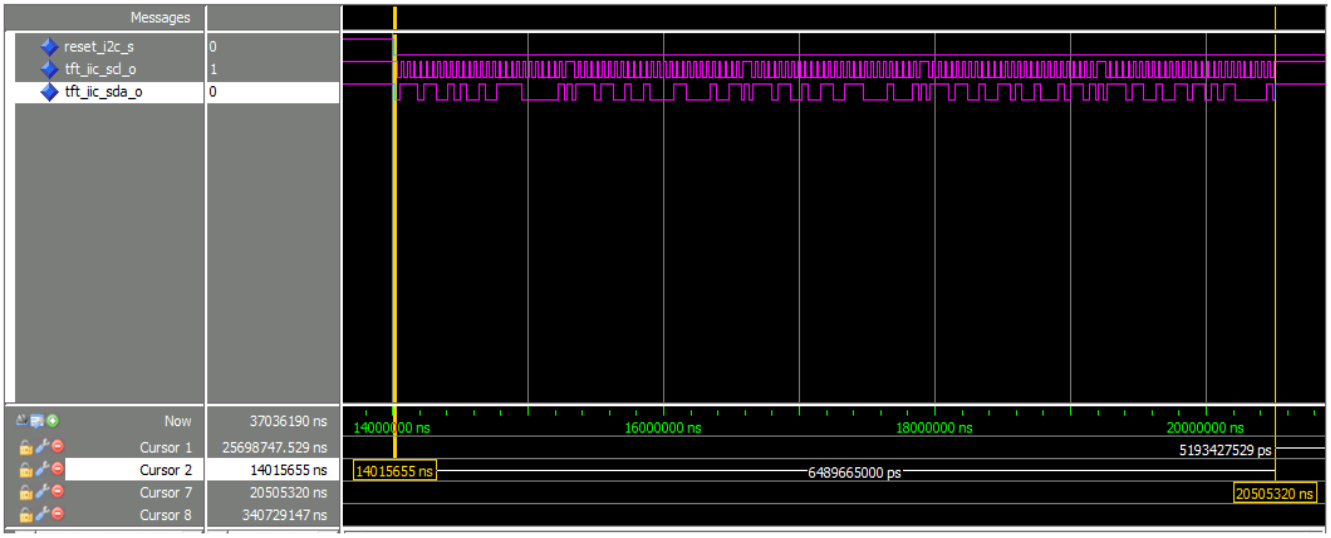

Figura 22 Forma de onda da interface de configuração I2C

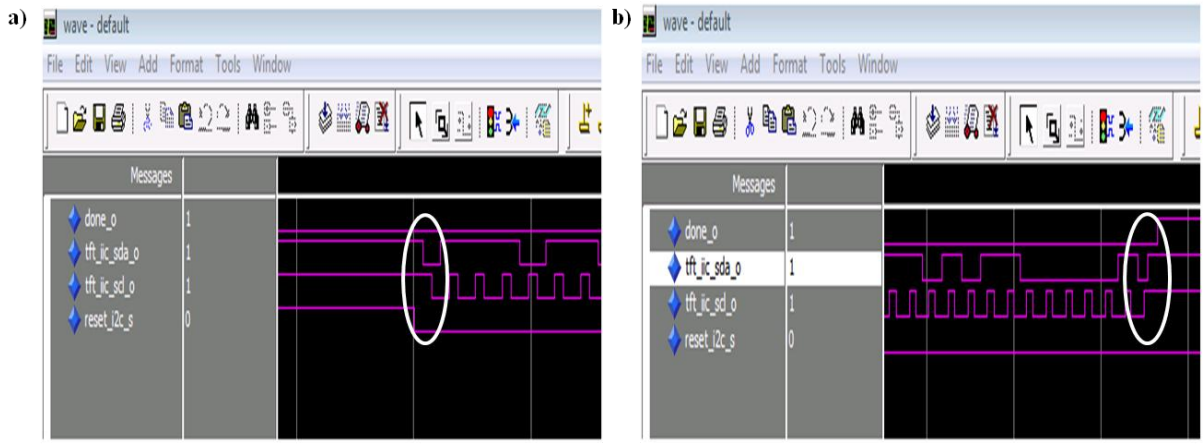

Figura 23 Análise da transmissão I2C a) Condição de início de transmissão; b) Condição de fim de transmissão

Já na figura 23 se pode analisar o atendimento das condições de início e fim de transmissão conforme a protocolo I2C. Onde a condição de início se caracteriza pela linha de clock permanecer no nível ‘1’ e na linha de dado ocorrer uma transição do nível ‘1’ para ‘0’. De maneira análoga, a condição de fim se caracteriza pela linha de clock permanecer no nível ‘1’ enquanto na linha de dado ocorre a transição do nível ‘0’ para ‘1’.

6.4 Gerador de Resets

A figura 24 mostra o funcionamento do gerador de resets. Podem-se notar os delays de 7ms entre os diferentes resets gerados.

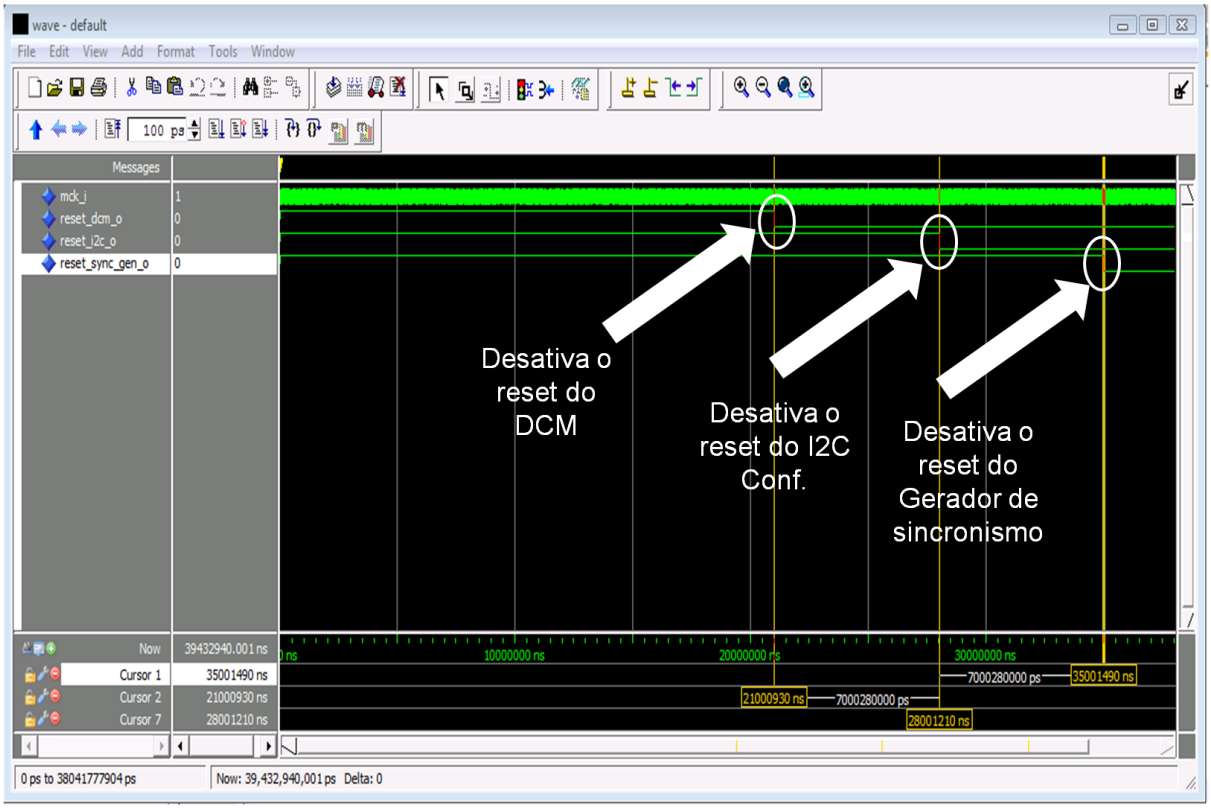

Figura 24 Forma de onda do gerador de resets

6.5 Topo do design

Assim após a simulação de cada sub-bloco isolado foi realizada uma simulação do topo do sistema. Em detalhe na figura 25 está à ação do sub-bloco gerador de reset. Essa figura ilustra o início de operação do sistema assim como sua ordem de inicialização.

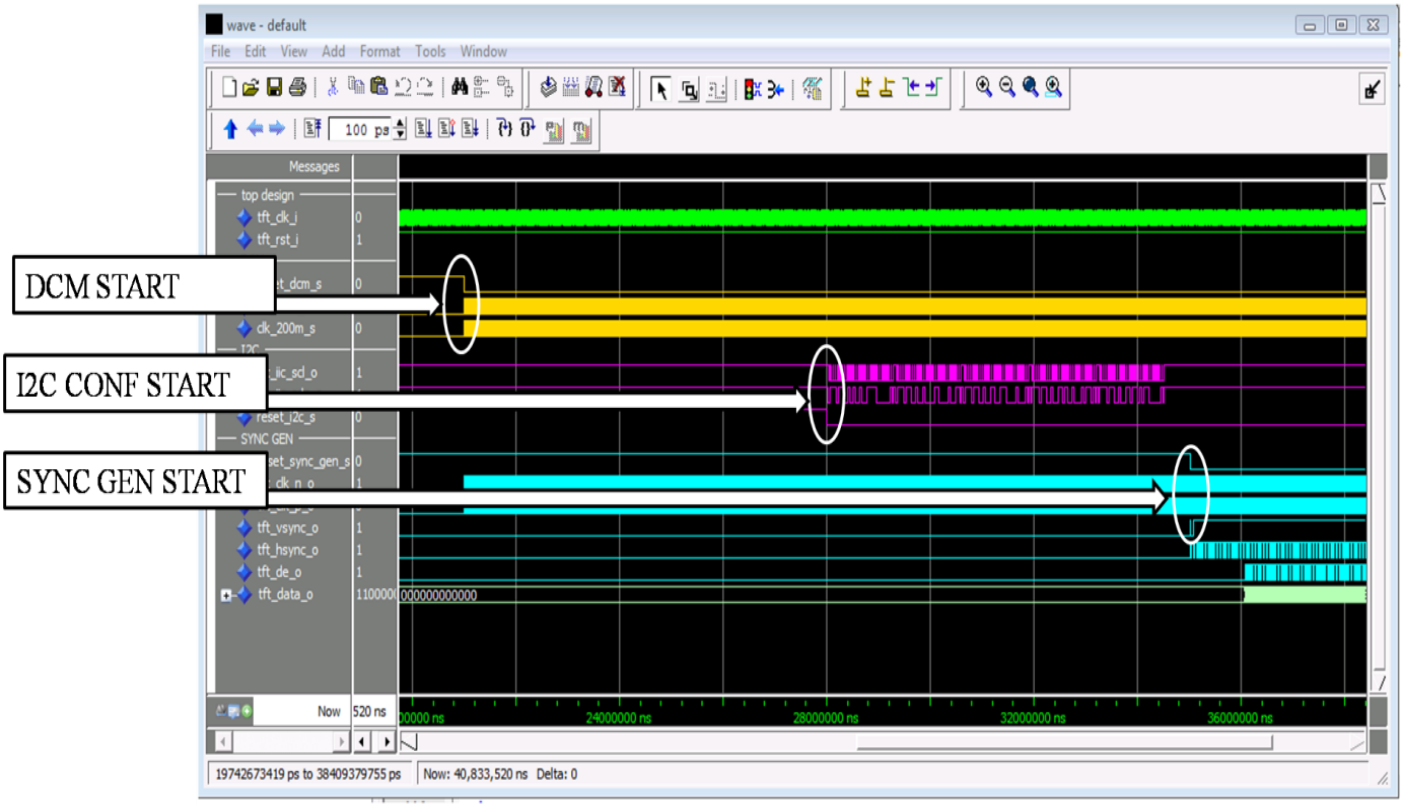

Figura 25 Simulação do topo do sistema

7. IMPLEMENTAÇÃO EM HARDWARE

Após o período de simulação do projeto e a verificação do atendimento das especificações foi realizada a síntese da lógica desenvolvida. Com o propósito de analisar o tamanho ocupado dentro do FPGA (quantidade de *LUTs* e de *Flip-Flops*), cada sub-bloco foi sintetizado separadamente.

Somente o topo do design foi sintetizado e implementado em hardware, uma vez que os sub-blocos separadamente não realizam a operação desejada. A ferramenta de síntese utilizada foi o software *ISE10.1* (Xilinx).

7.1 Memória RAM

Como descrito no item 5.1 esse sub-bloco é composto por seis block RAMs de 32k bytes cada. Sua lógica de controle para sobreposição de imagens e leitura da memória é realizada por contadores e comparadores. A tabela 6 mostra o resultado da síntese do sub-bloco (obtido pela ferramenta).

Tabela 6 Report de utilização do componente – RGB_BRAM

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	54	69120	0%
Number of Slice LUTs	90	69120	0%
Number of fully used LUT-FF pairs	54	90	60%
Number of bonded IOBs	23	640	3%
Number of Block RAM/FIFO	38	148	25%
Number of BUFG/BUFGCTRLs	1	32	3%

Analisando a tabela 6 se percebe que a ferramenta sintetizou a lógica esperada, estando presentes no report de síntese as memórias RAMs assim como as unidades lógicas necessárias para o atendimento dos requisitos do sub-bloco.

Pelo fato dos dois últimos bits de dados das memórias serem desprezados, a ferramenta interpreta as memórias descritas contendo a largura de dados de seis bits, porém a memória utilizada para indicar o momento de sobreposição das imagens à ferramenta mantém com largura de dados de oito bits. Sendo assim a ferramenta interpreta cinco memórias com largura de dados de seis bits e uma memória com largura de dados de oito bits. Esse fato se percebe no número total de memórias elementares utilizadas. Sabendo que cada unidade de memória elementar contém 36k bits [9] se obtém o cálculo realizado pela ferramenta:

- Dado que a largura de endereços das memórias descritas é de 15 bits (32k endereços);
- Então para memórias interpretadas com largura de dados de 6 bits o consumo de memórias elementares é: (6x32k)/36k = 6 memórias elementares;
- Para memórias interpretadas com largura de dados de oito bits o consumo de memórias elementares é: (8x32k)/36k = 8 memórias elementares;
- Logo o total de memórias elementares é: 5x6 + 8 = 38 unidades de memória elementares.

O tamanho de ocupação do sub-bloco no FPGA é pequeno, uma vez que a utilização de LUTs e Flip-Flops foi baixa.

7.2 Gerador de sincronismo de vídeo

O processo de geração do sincronismo de vídeo é baseada em contadores, comparadores (parte operativa) e uma máquina de estados para controle dos contadores (parte de controle). A tabela 7 ilustra o resultado da síntese lógica realizada pela ferramenta.

Tabela 7 Report de utilização do componente – SYNC_GEN

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	46	69120	0%
Number of Slice LUTs	131	69120	0%
Number of fully used LUT-FF pairs	46	131	35%
Number of bonded IOBs	46	640	7%
Number of BUFG/BUFGCTRLs	1	32	3%

Devido sua baixa complexidade se observa que sub-bloco ocupa pequena quantidade de elementos lógicos.

7.3 Configurador I2C

Com a função de executar um mestre I2C (conforme o protocolo), o sub-bloco é composto por uma máquina de controle para implementar o protocolo, e por sua parte operativa. O resultado de sua síntese esta na tabela 8.

Tabela 8 Report de utilização do componente – I2C_CONF

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	80	69120	0%
Number of Slice LUTs	130	69120	0%
Number of fully used LUT-FF pairs	79	131	60%
Number of bonded IOBs	6	640	0%
Number of BUFG/BUFGCTRLs	1	32	3%

7.4 Conversor de largura de dados

Esse sub-bloco é responsável pela lógica de cola entre as interfaces do FPGA e o D/A. É basicamente composto por *Flip-Flops DDR* para amostrar os dados em ambas as bordas do relógio. Cabe ressaltar o baixo número de LUTs utilizado por esse sub-bloco. Esse fato se

deve pelo motivo da baixa complexidade do sub-bloco, tendo em vista que ele realiza poucas operações lógicas. A tabela 9 ilustra seu resultado de síntese.

Tabela 9 Report de utilização do componente – CONV

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	14	69120	0%
Number of Slice LUTs	2	69120	0%
Number of fully used LUT-FF pairs	0	16	0%
Number of bonded IOBs	40	640	6%
Number of BUFG/BUFGCTRLs	1	32	3%

7.5 Troca dinâmica das resoluções

Sub-bloco responsável pela geração e distribuição de todos os relógios do sistema. Composto por DCMs, multiplexadores e buffers de relógio o sub-bloco é capaz de trocar a frequência de clock fornecida para o sistema conforme a resolução exigida pela imagem.

Conforme esperado, analisando a tabela 10 se percebe a presença de três DCMs verificando a descrição feita no item 5.5.

Tabela 10 Report de utilização do componente – RESOLUTION

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	9	69120	0%
Number of Slice LUTs	17	69120	0%
Number of fully used LUT-FF pairs	4	22	18%
Number of bonded IOBs	76	640	11%
Number of BUFG/BUFGCTRLs	9	32	28%
Number of DCM_ADVs	3	12	25%

7.6 Gerador de resets

Responsável pela garantia da ordem de inicialização dos sub-blocos do sistema, o gerador de resets possui uma máquina de controle e contadores para definir uma nova base de tempo (para a geração dos delays). A tabela 11mostra o resulado de sua síntese.

Tabela 11 Report de utilização do componente – RESET_GEN

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	32	69120	0%
Number of Slice LUTs	38	69120	0%
Number of fully used LUT-FF pairs	31	39	79%
Number of bonded IOBs	8	640	1%
Number of BUFG/BUFGCTRLs	1	32	3%

7.7 Topo do design

Após a sintetizar com êxito cada sub-bloco separadamente, o topo do sistema foi montado e sintetizado. Devido às simplificações realizadas pela ferramenta de síntese, a ocupação do topo do sistema no FPGA não é a soma das ocupações de cada sub-bloco. A tabela 12 apresenta o resultado final do espaço ocupado pelo projeto no FPGA. Cabe ressaltar que o resultado presente na tabela 12 não é um report de síntese, mas sim um report final da ocupação do FPGA (após o processo de implementação executado pela ferramenta).

Tabela 12 Report de utilização do componente – TOP_DESIGN

Device Utilization Summary				
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	263	69,120	1%	
Number used as Flip Flops	263			
Number of Slice LUTs	400	69,120	1%	
Number used as logic	397	69,120	1%	
Number using O6 output only	280			
Number using O5 output only	32			
Number using O5 and O6	85			
Number used as exclusive route-thru	3			
Number of route-thrus	35	138,240	1%	
Number using O6 output only	35			
Slice Logic Distribution				
Number of occupied Slices	167	17,280	1%	
Number of LUT Flip Flop pairs used	416			
Number with an unused Flip Flop	153	416	36%	
Number with an unused LUT	16	416	3%	
Number of fully used LUT-FF pairs	247	416	59%	
Number of unique control sets	27			
IO Utilization				
Number of bonded IOBs	27	640	4%	
IOB Flip Flops	14			
Specific Feature Utilization				
Number of BlockRAM/FIFO	38	148	25%	
Number using BlockRAM only	38			
Total primitives used				
Number of 36k BlockRAM used	38			
Total Memory used (KB)	1,368	5,328	25%	
Number of BUFG/BUFGCTRLs	10	32	31%	
Number used as BUFGs	10			
Number of DCM_ADVs	3	12	25%	

Pode-se observar na tabela 12 que o sistema digital desenvolvido ocupa pouco espaço dentro do FPGA. Analisando o item *Slice Logic Utilization* se obtém a quantidade de elementos lógicos necessários para a implementação do projeto. Já o item *Slice Logic Distribution* demonstra a forma com que a ferramenta distribuiu esses elementos lógicos. O espaço ocupado no FPGA não foi um fator controlador nesse projeto. Porém essa característica se torna interessante uma vez que o FPGA deve comportar blocos de maior complexidade para o desenvolvimento do decodificador H.264.

8. RESULTADOS

Para verificar o processo de sobreposição as imagens da figura 26 foram armazenadas no FPGA. Ambas as imagens possuem o tamanho 176x144. O padrão de cor utilizado como indicativo para a área de sobreposição foi a cor preta da imagem presente na figura 26 b.

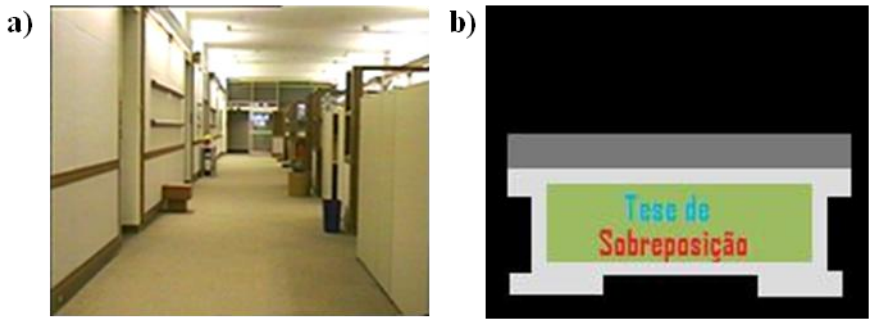

Figura 26 Imagens de teste a) Hall; b) Imagem teste

O resultado presente na figura 27 é uma foto do monitor de LCD. Se pode perceber que sobreposição das duas imagens foi realizada com sucesso.

Figura 27 Verificação do funcionamento da sobreposição de imagens

Os resultados para o processo de troca das resoluções estão presentes na figura 28. São fotos do mesmo monitor LCD usado para testar a sobreposição de imagens. Para a verificação das resoluções foi acessado o menu do monitor para que fosse exibido na tela a resolução do vídeo.

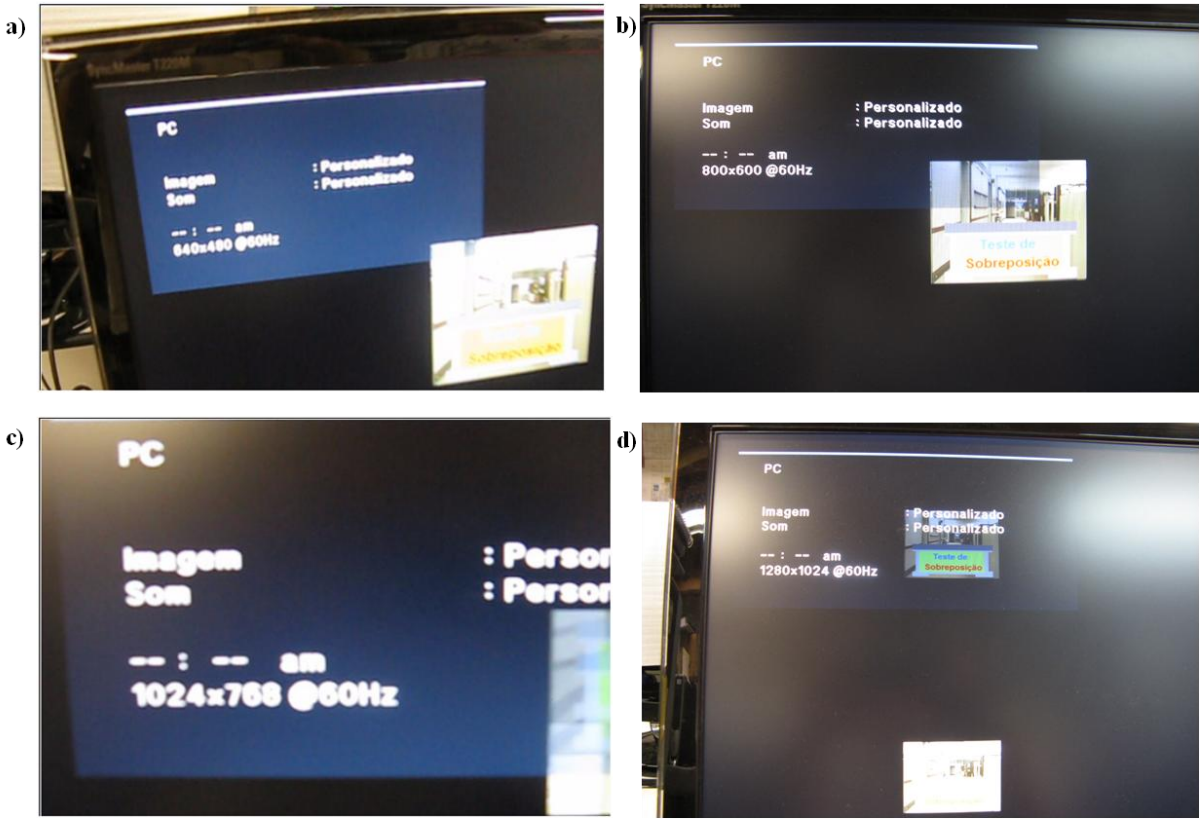

Figura 28 Verificação do funcionamento da troca de resoluções a) Resolução 640x480; b) Resolução 800x600; c) Resolução 1024x768; d) Resolução 1280x1024

9. MELHORIAS

Visando a integração do projeto com o decodificador H.264 é possível propor algumas melhorias: introdução de um barramento de leitura de memória externa (barramento DDR), realizar a composição da imagem adotando a metodologia de transparências e utilizar o padrão de cores YCbCr.

A introdução do barramento de leitura de memória externa se deve pela necessidade de espaço para o armazenamento dos dados de vídeo. Sabendo que o decodificador poderá gerar imagens em resolução *full HD* não é viável armazenar dentro do FPGA imagens com esse tamanho. Sua implementação deve ser realizada conforme as definições do fabricante da memória utilizada. Um fato a ser considerado deve ser as taxas de relógio de escrita e leitura da memória, uma vez que as operações de escrita e leitura serão realizadas simultaneamente.

A adoção da metodologia de transparências para a composição da imagem exibida tem a função de mostrar no vídeo duas imagens sem que uma imagem apague a outra. Para implementar essa metodologia basta que não haja a troca entre as duas imagens em determinada região (metodologia utilizada nesse trabalho), mas que elas sejam somadas ponderadamente.

Visando a integração do projeto como decodificador H.264 se propõe a troca do padrão RGB para o padrão de cores YCbCr, uma vez que a saída de dados do decodificador é no padrão YCbCr. Sua implementação é simples, havendo somente a necessidade de realizar algumas operações aritméticas sobre os dados no padrão RGB.

10. CONCLUSÃO

Todas as etapas do projeto obtiveram êxito. Todas as especificações apresentadas foram implementadas e seus resultados foram satisfatórios. Destacam-se no período inicial do projeto a intensa pesquisa sobre assuntos relacionados ao trabalho, bem como a manipulação de ferramentas de simulação e de síntese lógica.

A lógica desenvolvida foi prototipada no kit de desenvolvimento com sucesso. Todas as funcionalidades propostas tiveram seus resultados verificados através da exibição de imagens em um monitor LCD.

Sendo o desenvolvimento desse projeto o primeiro contato com processamento de imagem, algumas dificuldades foram encontradas. Tais dificuldades foram sanadas com intensa pesquisa na bibliografia, bem como nas reuniões com os professores do LaPSI. Outro fator relevante para a obtenção do sucesso nesse projeto foi o conhecimento prévio em linguagem de descrição de hardware assim como na manipulação das ferramentas de simulação e síntese. Esse conhecimento foi adquirido ao longo da formação acadêmica junto com a atividade de estágio desenvolvida ao longo do curso.

Por fim, a realização desse projeto foi uma forma de aprendizado enriquecedora e estimulante. Pois trata de um assunto que não tem ampla abordagem durante a formação dos alunos e pelo motivo que o bloco de lógica desenvolvido fará parte do projeto do decodificador de vídeo H.264, ou seja, vai se integrar ao projeto do SBTVD.

11. REFERÊNCIAS BIBLIOGRÁFICAS

[1] CHRONTEL, **Application Notes – Registers Read/Write Operation AN41.** 13p. Disponível em <http://www.chrontel.com/pdf/an41.pdf >. Acesso em março de 2010.

[2] CHRONTEL, **Data Sheet do componente CH7301C.** 29p. Disponível em <http://www.chrontel.com/pdf/7301ds.pdf >. Acesso em março de 2010.

[3] RICHARDSON, I. E. G. **H.264 and MPEG-4 Video Compression.** Chichester: Wiley, 2003. p.1-98. ISBN 0-470-84837-5.

[4] TERAOKA, K. **Implementação do codificador de vídeo H.264 com transformada flict.** 2003. p. 6-9. Dissertação de mestrado – Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, 2003.

[5] VGA signal timing, **Definição dos limites de contagem pelo padrão VGA.** Disponível em <http://tinyvga.com/vga-timing>. Acesso em abril de 2010.

[6] Xilinx, **ML505/ML506/ML507 Evaluation Platform (User Guide).** 60p. Disponível em <http://www.xilinx.com/support/documentation/boards_and_kits/ug347.pdf>. Acesso em março de 2010.

[7] Xilinx, **ML50x Schematics .** p. 3-6. Disponível em <http://www.xilinx.com/support/documentation/boards_and_kits/ml50x_schematics.pdf>. Acesso em março de 2010.

[8] Xilinx, **TFT Controller (DS695 product Specification).** 27p. Disponível em <http://www.xilinx.com/support/documentation/ip_documentation/xps_tft.pdf>. Acesso em março de 2010.

[9] Xilinx, **Virtex-5 FPGA DataSheet (Product Specification).** Disponível em <http://www.xilinx.com/support/documentation/data_sheets/ds202.pdf>. Acesso em março de 2010.