Introdução a Dispositivos Lógicos Programáveis

Kamila Rose da Silva
Curso Superior em Sistemas de Telecomunicações
IFSC - Instituto Federal de Santa Catarina, São José, SC

***Abstract*—**

***Index Terms*—FPGA, VHDL, DLP, Arquitetura**

I. Dispositivos Lógicos Programáveis

Segundo (PEDRONI, 2010) dispositivos lógicos programáveis são *c*hips que possuem *h*ardware programável, são reconfiguráveis pois sua programabilidade é em nível de *h*ardware, ou seja, este pode ser configurado de acordo com as necessidades do projeto. Elimina-se assim a necessidade da fabricação de um circuito integrado exclusivo para a aplicação. Esta é a principal característica que proporcionou o desenvolvimento tecnológico na área de implementação da eletrônica digital moderna, tornando seu uso crescente em projetos de *h*ardware.

Outra característica importante dos DLPs é a sua alta velocidade de operação, pois são integrados a eles, em alguns casos, um elevado número de portas lógicas e flip-flops. Outro fato que colabora para o processamento mais rápido está relacionado às linguagens de descrição em hardware em que o código descrito do circuito é realizado paralelamente. Isto justifica a possibilidade de implementar qualquer tipo de circuito digital atendendo às mais variadas especificações que se pode ter em um projeto, com velocidade superior à implementação por software (PEDRONI, 2010).

Os DLPs foram criados na década de 1970, com o intuito de possibilitar que circuitos lógicos combinacionais pudessem ser programáveis. Desde então, muitas pesquisas cercaram esta área, fato que permitiu um grande avanço tecnológico.

Os primeiros *c*hips integrados aos DLPs tinham apenas algumas portas lógicas, estes são conhecidos como SPLD (TOCCI; WIDMER; MOSS, 2007). Mas em meados da década de 80, foram desenvolvidos *c*hips com novas configurações, mais avançados, que possuem alta complexidade, como o CPLD e os FPGA.

A. SPLD - Simple Programmable Logic Device

Dentre os primeiros *c*hips criados estão o PAL e o GAL. A arquitetura presente no PAL, baseia-se em apenas algumas portas lógicas, fundamentadas na implementação de funções lógicas combinacionais. É um conjunto lógico composto por portas AND programáveis conectadas a portas OR fixas (TOCCI; WIDMER; MOSS, 2007).

Esses *c*hips implementam apenas circuitos combinacionais, pois sua arquitetura não contém nenhum registrador para permitir o armazenamento de informações, por esse motivo, mais tarde foi adicionado um flip-flop em cada saída de portas OR, o que possibilitou a construção de circuitos sequenciais. A programação da conexão entre os arranjos das portas de entradas podem ser feitas por meio de fusíveis ou antifusíveis. Estas conexões são realizadas apenas uma vez, geralmente através de um pulso de programação de tensão, que gera ou rompe um canal condutivo, proporcionando estas interconexões (BOCHETTI, 2004).

A principal característica que difere um dispositivo PAL e GAL é utilização de uma matriz EEPROM responsável por armazenar as interconexões realizadas a partir das entradas, assegurando sua reprogramação (TOCCI; WIDMER; MOSS, 2007).

Sua arquitetura, exibida na Figura 1, continua aplicando a implementação através de funções lógicas combinacionais encontradas em dispositivo PAL, porém houve uma substituição no que se refere as portas OR fixas, que foram substituídas por macrocélulas.

A inserção dessas OLMC nas saídas das portas AND programáveis, foi uma importante mudança em sua arquitetura em relação ao DLP anterior. Sua estrutura consiste de um flip-flop tipo D, uma porta OR fixa de oito entradas ligadas as portas AND programáveis, uma porta XOR programável e cinco multiplexadores usados para redirecionar o sinal, apresentados na Figura 2. O uso dessa macrocélula proporcionou o aumento dos modos de funcionamento do dispositivo tornando-o mais flexível (PEDRONI, 2010).

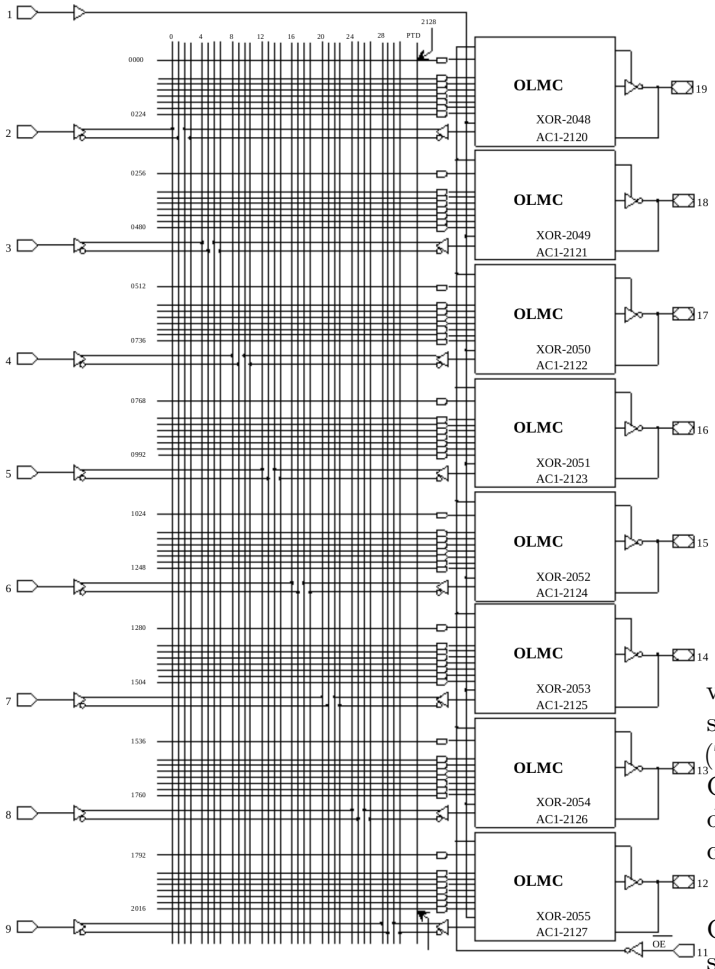
B. CPLD - Complex Programmable Logic Device

De acordo com (PEDRONI, 2010), CPLDs são *c*hips constituídos basicamente de vários blocos de SPLD, que estão conectados através de um arranjo de interconexões programável. Este arranjo de interconexões também é responsável pela conexão entre os *d*rivers de I/O e pos-

[0] <http://html.alldatasheet.com/html-pdf/27266/TI/PAL16L8/125/5/PAL16L8.html>

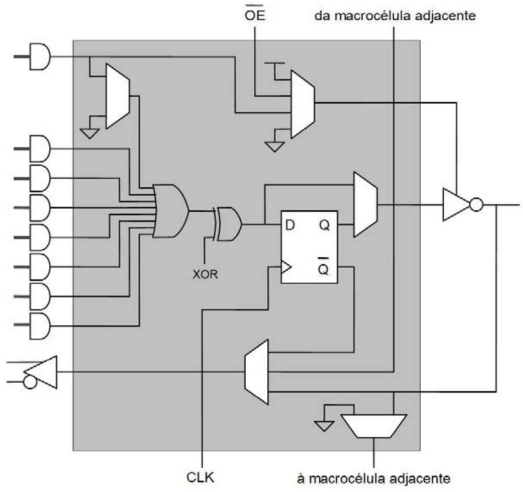
[1] <http://html.alldatasheet.com/html-pdf/70214/LATTICE/GAL16V8D/1255/5/GAL16V8D.html>

Fig. 1: Arquitetura GAL.



Fonte: Alldatasheet[1].

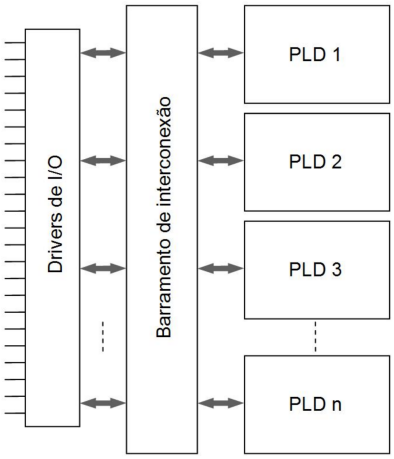
Fig. 2: Estrutura da macrocélula.



Fonte: (PEDRONI, 2010).

teriormente aos pinos I/O. Sua estrutura básica pode ser observada na Figura 3.

Fig. 3: Estrutura CPLD.



Fonte: (PEDRONI, 2010).

Dispositivos CPLDs costumam utilizar memórias não voláteis EEPROM ou Flash, para a interconexão interna, sendo este tipo de memória apagável e reprogramável (TOCCI; WIDMER; MOSS, 2007). São características dos CPLDs um menor consumo de potência, além da utilização de frequência máxima menores que de outros dispositivos como os FPGAs.

A Figura 4 mostra a arquitetura da série MAX3000 de CPLD da Altera. Esta é composta por diversos LAB, que são blocos de arranjos lógicos, onde cada LAB contém um conjunto de dezesseis macrocélulas, neste caso. Estas macrocélulas por suas vez, estão interligadas por PIA, que são arranjos de interconexões (PEDRONI, 2010). Essas mudanças viabilizaram aos CPLDs maior versatilidade para uso.

C. FPGA - Field Programmable Gate Array

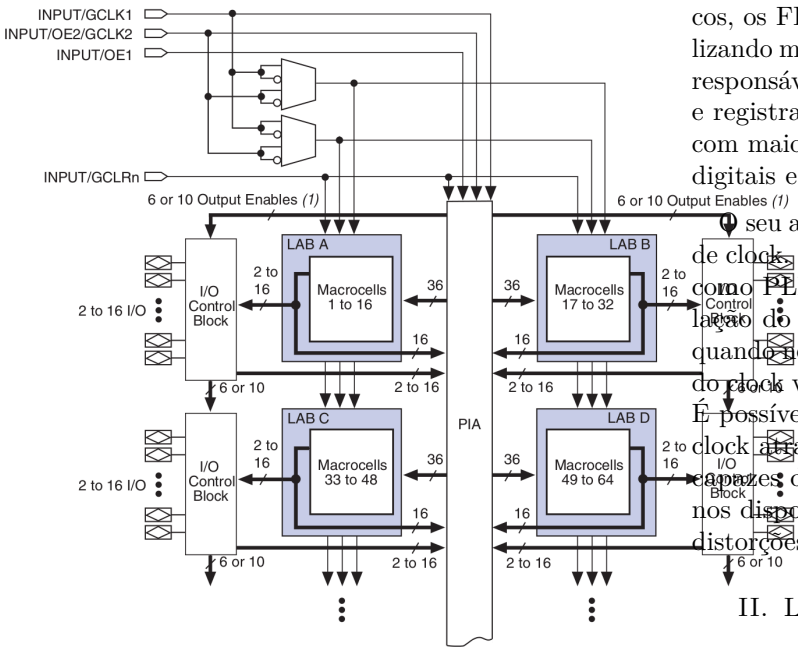
Os FPGA diferenciam-se dos demais tipos de dispositivos em vários aspectos que vão além da arquitetura, como por exemplo, a tecnologia empregada, o tamanho, além de possuir alta capacidade de processamento através do uso de blocos lógicos e melhor desempenho (PEDRONI, 2010).

De acordo com (TOCCI; WIDMER; MOSS, 2007) a arquitetura dos FPGAs, exibida na Figura 5, é composta por uma matriz de blocos lógicos programáveis, que são independentes entre si mas podem ser relacionados para realizar funções maiores.

Cada bloco lógico é composto por ALM. Um ALM possui um conjunto de células de armazenamento que compõem

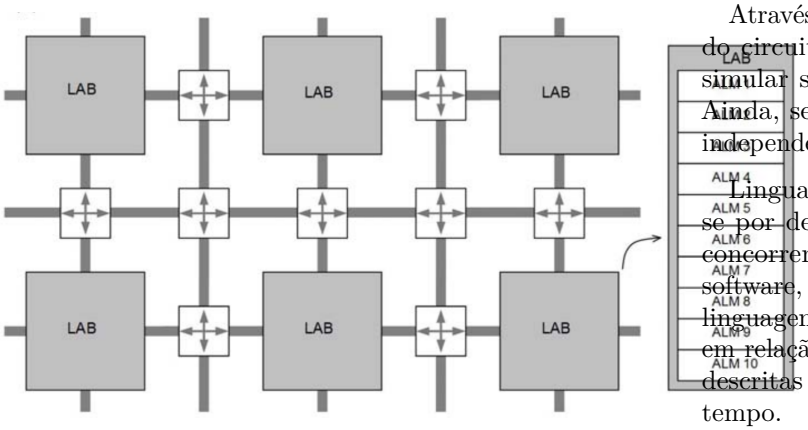
[2]<https://www.altera.com/en_US/pdfs/literature/ds/m3000a.pdf>

Fig. 4: Arquitetura da série MAX3000 da Altera.



Fonte: Altera[2].

Fig. 5: Arquitetura de um FPGA.



Fonte: (PEDRONI, 2010).

uma LUT, que funciona como uma tabela verdade programada de acordo com a descrição da função lógica necessária. Possuem até seis entradas, proporcionando combinações de até sessenta e quatro células de armazenamento com uma saída (COSTA, 2009).

A interconexão entre os blocos lógicos e os blocos de *I*nput/0utput são realizadas através de chaves de interconexão programáveis que são conectadas por canais de roteamento. Estas chaves de roteamento são geralmente baseadas em memória SRAM, que só podem ser programadas quando o dispositivo estiver energizado (COSTA, 2009).

De acordo com (PEDRONI, 2010) além dos blocos lógicos, os FPGAs são compostos por blocos RAM, disponibilizando memória volátil para o usuário e blocos DSP que são responsáveis pela execução de somadores, multiplicadores e registradores. Os blocos DSPs tem a função de efetuar com maior velocidade e eficácia o processamento de sinais digitais e aplicações de mídias como áudio e vídeo.

O seu alto desempenho justifica a necessidade de controle de clock. Para isto, são implementados circuitos conhecidos como PLL. Estes circuitos são responsáveis pela manipulação do clock que entra nos dispositivos. Eles realizam, quando necessário, deslocamentos, multiplicações e divisões do clock vindo de uma fonte externa para dentro do FPGA. É possível também, corrigir alguns efeitos indesejáveis do clock através de filtragens. Resumindo, estes circuitos são capazes de manipular e gerenciar a distribuição do clock nos dispositivos de maneira que não prejudique ou cause distorções no projeto implementado (PEDRONI, 2010).

II. Linguagens de Descrição em Hardware

De acordo com (TOCCI; WIDMER; MOSS, 2007), a HDL são utilizadas para descrição das configurações que se deseja obter em um determinado circuito digital, visando sua implementação em um *h*ardware.

Através dela é possível definir o tipo de comportamento do circuito, além da gerar a síntese da lógica descrita e simular sua saída para então implementá-la fisicamente. Ainda, segundo (PEDRONI, 2010), essas linguagens são independentes de fabricante ou tipo de tecnologia específica.

Linguagens de descrição em *h*ardware caracterizam-se por descreverem funções que são realizadas de forma concorrente, diferentes das instruções de linguagens de software, que obedecem a ordem de descrição, ou seja, são linguagens sequenciais. O paralelismo significa um ganho em relação a execução sequencial, porque todas as funções descritas em nível de *h*ardware são executadas ao mesmo tempo.

Existem diversas linguagens para descrever *h*ardware, as mais utilizadas serão descritas a seguir. No desenvolvimento da proposta deste trabalho foi optado pela utilização da linguagem VHDL por ser largamente difundida para práticas de descrição e por ser uma linguagem que permite recursos para a modelagem de alto nível, além do uso de diagramas esquemáticos.

A. Verilog

O Verilog é uma linguagem de descrição em *h*ardware que se assemelha muito a linguagem de programação C. Foi criada na década de 1980, e possui sua última versão de padronização no ano de 2005. É caracterizada por ser uma linguagem que possui modelagem de nível de portas,

é sensível ao uso de letras maiúsculas ou minúsculas e permite a opção de temporizar o acontecimento de algum evento.

Fig. 6: Estrutura básica de um Flip flop tipo D em Verilog.

```
module d_ff(q,d,clk,reset);
     output q;
     input d,clk,reset;
     reg q;
     always @(posedge reset or negedge clk)
     if (reset)
         q<= 1'b0;
     else
         q<=d;
endmodule
```

Fonte: Elaborada pelo autor.

A estrutura básica de um código Verilog mostrada na Figura 6 contém um módulo e a descrição do circuito. No módulo são descritos as portas de entradas e saídas do circuito projetado. Na descrição são utilizados procedimentos como comando inicial, uma tarefa ou uma função, estes indicam o funcionamento do código. Como esta linguagem também é executada paralelamente, o Verilog proporciona um procedimento específico para execução do código em sequencial, atuando em paralelo com os demais comandos (MIDORIKAWA, 2007).

B. VHDL

O VHDL é umas das linguagens mais difundidas para descrição de *h*ardware de alta capacidade. Através da síntese deste código realizado pelo compilador, é gerado um circuito físico. O código VHDL descreve o tipo de comportamento ou a estrutura do circuito lógico projetado, sendo possível testá-lo antes da aplicação física em um determinado DLP (PEDRONI, 2010).

Sua principal característica é aplicação da execução concorrente, ou seja, o código deve ser interpretado como instruções que são executadas ao mesmo tempo. Diferente da execução realizada nos processadores e na CPU dos computadores, onde as instruções são executadas sequencialmente.

Para a implementação de circuitos e projetos sequenciais a linguagem VHDL disponibiliza o uso da atribuição comportamental que possibilita o uso de um processo. Dentro de sua estrutura o código é executado sequencialmente, porém, parte do código que não pertence a esse processo continua sendo executado paralelamente. Assim, o processo passa a ser concorrente com todas as demais instruções.

A estrutura principal do código VHDL é mostrada na Figura 7, e consiste em uma declaração de biblioteca e pacotes, a entidade do código e sua arquitetura.

Primeiramente, deve-se fazer a declaração das bibliotecas e pacotes que serão utilizados no projeto VHDL, a fim de que o compilador as reconheça e processe suas

Fig. 7: Estrutura básica de um Flip flop tipo D em VHDL.

```
library ieee;
use ieee.std_logic_1164.all;

entity d_ff is

   port
      d, clk, rst : in std_logic;
      q  : out std_logic);

end entity;

architecture ff of d_ff is

   begin

      FFD: process(clk, rst) is

      begin
         if (rst = '1') then
            q <= '0';
         elsif(clk'event and clk = '1') then
            q <= d;
         end if;
      end process;
end architecture;
```

Fonte: Elaborada pelo autor.

informações juntamente com o código. Duas bibliotecas estão disponíveis automaticamente, são elas a WORK e a STD. A biblioteca STD define os tipos dos dados básicos a qual um sinal pode pertencer. A biblioteca WORK define o local onde estão armazenados os arquivos do projeto em si (PEDRONI, 2010).

A entidade de um código VHDL define inicialmente a lista de portas de entradas e saídas do projeto, na subsessão PORT. A entidade também pode conter uma outra subsessão GENERIC, onde ocorre a declaração de parâmetros globais do código, neste caso apenas se for necessário (PEDRONI, 2010).

A arquitetura descreve o funcionamento do circuito. Esta descrição pode conter a declaração de sinais, tipos, constantes e funções de acordo com as necessidades de funcionamento do circuito. Além disto, a parte descritiva da arquitetura pode ser realizada de maneira que o código seja descrito concorrentemente ou sequencialmente, para isto existem três modelos principais de descrição, são eles: fluxo de dados, comportamental e estrutural, os quais são empregados de acordo com a necessidade de cada projeto.

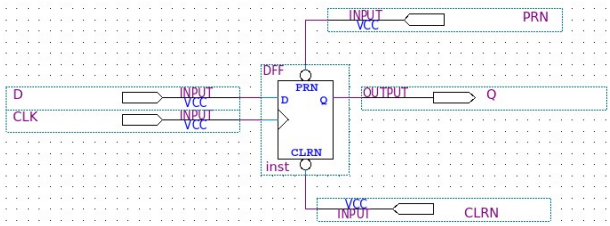
C. Diagrama Esquemático

A descrição em *h*ardware através de diagramas esquemáticos, segue o exemplo de implementação de simuladores de circuitos. Esta é uma linguagem de descrição dependente do fabricante ALTERA[3]. A grande vantagem da implementação virtual através de diagramas esquemáticos é a possibilidade da realização de testes de representação

[3]<www.altera.com>

das formas de ondas que indicam o funcionamento e comportamento do circuito. Esse fato simplifica e garante a execução correta do circuito projetado.

Fig. 8: Fluxo de desenvolvimento de um Flip flop tipo D usando o diagrama esquemático.



Fonte: Elaborada pelo autor.

Além da possibilidade de simulação para confirmar se o circuito está operando como esperado, os testes virtuais favorecem o usuário pelo fato de inibir erros ocasionados pela falha de equipamentos, como acontece com nas implementações reais em matrizes de contato. Também é possível efetuar análises de síntese antes de implementar o circuito em um DLP. Outro ponto positivo é que se pode obter a partir de diagramas esquemáticos seus respectivos códigos em diversas linguagens de descrição em hardware, como Verilog e VHDL.

A estrutura de um diagrama esquemático é apresentada na Figura 8. Inicialmente é definida a quantidade de pinos de entrada e saída, depois ocorre implementação do circuito desejado, através da biblioteca que proporciona inúmeras possibilidades para a inclusão de componentes. Além disso, pode-se também produzir seu próprio componente e salvar na biblioteca padrão do usuário. Em seguida, ocorre a ligação lógica entre os pinos de entrada, os componentes e os pinos de saída. A nomeação dos pinos também é necessária para identificação.

III. Plataformas de desenvolvimento

Para a implementação deste projeto, foi necessário o uso de duas plataformas de desenvolvimento onde os circuitos foram descritos e testados. Através delas foi realizado a coleta e análise dos resultados obtidos. A elaboração do circuitos foi feita no *s*oftware Quartus II enquanto os testes de funcionalidade realizados no *s*oftware Modelsim.

A. Quartus II - ALTERA

A plataforma de desenvolvimento utilizada para implementar este projeto é o Quartus II - *S*ubscription Program, da ALTERA[4]. Este ambiente de desenvolvimento possibilita a realização de projetos digitais com diversas formas de entrada. Como citado anteriormente, neste projeto foram utilizados diagramas esquemáticos e a linguagem de descrição em *h*ardware VHDL. As configurações utilizadas no Quartus II podem ser conferidas no Apêndice **??** e **??**.

O Quartus II permite que na compilação e síntese dos projetos, o projetista tenha acesso a informações sobre a capacidade do *h*ardware, a quantidade de elementos lógicos utilizado pelo circuito e a velocidade de execução. Mais detalhes podem ser obtidos através do tutorial[5] disponibilizado pela Altera.

Outro ponto relevante é o uso da ferramenta *T*imeQuest Timing Analyser, que possibilita a verificação dos atrasos de propagação e caminho crítico obtidos em cada circuito testado. Esta funcionalidade do Quartus II, disponibiliza um resumo do máximo atraso de propagação obtido de um determinado caminho percorrido pelo sinal dentro das trilhas do FPGA utilizado.

Estas análises de tempo geradas pela plataforma, baseiam-se em modelos que levam em consideração diversos quesitos rigoroso que interferem na verificação dos atrasos, desde temperatura, tensão, modos de configuração, além do tipo de FPGA utilizado. Para cada FPGA existe uma tabela com listas de atrasos para cada elemento físico do dispositivo. O modelo definido na utilização deste projeto foi o *s*low 85°C. Como característica principal, este possui baixa tensão e temperatura de junção de 85°C. O uso deste se deve a característica de captura do pior caso e os atrasos mais lentos (ALTERA, 2010). Este modelo também é responsável pela captura da frequência atingida pelo circuito.

B. Modelsim

Para as simulações dos circuitos implementados foi utilizado o simulador Modelsim, da Mentor Graphics[6]. Nele é possível definir as formas de ondas de cada entrada de um projeto e simular suas respectivas saídas. Ele permite também a geração de *T*estBenches, que são códigos genéricos que podem ser aplicados em um determinado projeto diversas vezes, facilitando a compreensão do funcionamento de um circuito específico além de economizar tempo, uma vez que o código teste já esteja pronto, basta executá-lo para visualizar os resultados.

Para a realização dos testes de verificação de funcionamento dos circuitos aritméticos, foram desenvolvidos códigos em VHDL para a geração de estímulos de entrada. Todos os circuitos foram testados para o modo funcional e temporal, este último além de avaliar a funcionalidade, leva em conta os tempos de propagação obtidos em cada circuito.

Os *s*cript tcl gravados em arquivos com extensão .do permite uma padronização no teste dos somadores, subtratores

[4] <www.altera.com>

[5] <ftp://ftp.altera.com/up/pub/Altera_Material/14.0/Tutorials/VHDL/Quartus_II_Introduction.pdf>

[6] <http://www.mentor.com>

e multiplicadores.

References

ALTERA. Guaranteeing silicon performance with fpga timing models. Altera Corporation, San Jose, CA, 2010. Disponível em: <https://www.altera.com/en_US/pdfs/literature/wp/wp-01139-timing-model.pdf>. page.55

BOCHETTI, M. R. Mecanismo de reconfiguração dinâmica aplicados ao projeto de um processador de imagens reconfiguráveis. 2004. page.11

COSTA, C. d. *Projetos de Circuitos Digitais com FPGA*. [S.l.]: Editora Érica, 2009. ISBN 978-85-365-0239-7. page.33

MIDORIKAWA, E. T. Uma introdução às linguagens de descrição de hardware. 2007. page.44

PEDRONI, V. A. *Eletrônica Digital Moderna e VHDL: Princípios Digitais, Eletrônica Digital, Projeto Digital, Microeletrônica e VHDL*. [S.l.: s.n.], 2010. ISBN 9788535234657. page.11, page.22, page.33, page.44

TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L. *Sistemas digitais: princípios e aplicações*. [S.l.]: Pearson Prentice Hall, 2007. page.11, page.22, page.33