Instituto Mauá de Tecnologia
Praça Mauá 01
São Caetano Do Sul

Nota de aula I

Introdução a Lógica Programável

Rafael Corsi Ferrão

corsiferrao@gmail.com

4 de outubro de 2015

Sumário

1 Introdução

Com o avanco da tecnologia e da necessidade do mercado de eletrônica, recursos tido como diferencias em sistemas eletrônicos do passado (ex: USB, Ethernet, Vídeo ...) são considerados básicos em qualquer aparelho que venha a ser desenvolvido em dias atuais, além disso as demandas de processamento em eletrônica embarcada são cada vez maiores (ex: LeapMotion). Para suprir essas demandas o desenvolvimento de sistemas eletrônicos são cada vez mais dependentes da utilização de recursos computacionais e circuitos eletrônicos propriamente dito não são mais parte exclusiva do desenvolvimento mas sim partes de um sistema complexo.

Projetos dessa complexidade requerem um grande nível de abstração para reduzir o tempo de processamento e o aumento da eficiência do desenvolvedor. Porém quando o projeto é analisado em camadas mais elementares, como por exemplo, comunicação entre CIs é necessário o entendimento dos níveis de tensão e corrente envolvidos, tipo de trilhas a serem utilizados e diversos outros fatores.

Para passar de uma especificação realista para uma implementação funcional e eficiente, há a necessidade do conhecimento das tecnologias envolvidas além de boas ferramentas de desenvolvimento (EDA *)

Tradicionalmente, os aumentos nas funcionalidades estão atrelados a implementação de softwares mais complexos embarcados em sistemas microcontrolados (μC). Em muitos casos, a implementação em software possui desempenho a baixo do desejado, e muitas vezes sem uma solução no nível de optimização do programa. Para solucionar essa questão, opta-se por realizar tarefas críticas ou de grande processamento em hardware, essa abordagem porem eleva significativamente o custo do desenvolvimento.

Se analisarmos as arquiteturas dos microprocessadores, veremos que os mesmos já fornecem uma solução para esse problema, são os chamados de periféricos. Os periféricos são harwares pré definidos que executam tarefas concorrente ao programa em execução, como por exemplo: acesso a memória; comunicação serial .... Esses periféricos são configurados geralmente por registradores no μC, sendo a única maneira de alterar seu

Electronic Design Automation

comportamento. Soluções customizadas em hardware não são possíveis de serem implementadas nesse tipo de sistema, como por exemplo, a implementação de um protocolo de comunicação customizado.

Dispositivos lógicos programáveis (CLPD, FPGA) unem a flexibilidade e o poder da programação com a velocidade de uma implementação em hardware a um custo compatível com o mercado. Permitindo assim que programas descritos por linguagem especificas sejam sintetizado inteiramente em hardware, possibilitando que aplicações complexas sejam desenvolvidas.

1.1 Tecnologias

O processo básico de fabricação de um CI para construção de suas camadas é chamado de litografia, onde uma mascara define os padrões a serem gravados. Tipicamente um dispositivo pode possuir de 10 a 15 camada, sendo o número do processo de litografia o mesmo do de camadas a cada novo processo é utilizado uma mascara diferente. A área utilizado por um CI é normalmente padronizado para a área do menor transistor utilizado no circuito (em mícrons).

Dois tipos básicos de tecnologias são utilizadas, as chamadas CIs de aplicação especial ASIC* onde a cada novo projeto deve-se executar todo o processo de fabricacão de CIs, e os não-ASICs, ou de pratilheira, onde os hardwares já são pré definidos pelos fabricantes.

A seguir analisamos em poucos detalhes a tecnologias presentes no mercado de circuitos digitas.

1.1.1 Full ASIC

No Full ASIC o desenvolvedor deve levar em conta todos os especto de um circuito digital onde possui controle total do circuito inclusive de sua micro eletrônica, possibilitando desde a escolha do transistor até a área onde será implementado. Os designs em ASIC são o mais otimizados e possuem a melhor performance, porém a um alto custo de desenvolvimento. Não é uma técnica pratica para desenvolvimento de circuitos complexos onde milhares de transistores são utilizados. É utilizado geralmente para o desenvolvimento de estruturas básicas que serviram como base para outros desenvolvedores como por exemplo, elementos básicos de memória, elementos lógicos, entre outros. A execução do projeto deve passar por todos os processos de fabricação de componentes eletrônicos, desde a litografia até o encapsulamento.

* *Application-Specific Integrate Circuit*

1.1.2 Standard-cell ASIC

O projeto atual na camada chamada de "*gate-level*"onde utiliza de células lógicas previamente definidas e validadas para o desenvolvimento de circuitos lógicos. Essas bibliotecas podem possuir componentes complexos como memórias RAM e somadores, além dos elementos básicos como inversores, flip-flops tipo D. Como no Full ASIC execução do projeto deve passar por todos os processos de fabricação de componentes eletrônicos, desde a litografia até o encapsulamento.

1.1.3 Gate Array ASIC

No gate array ASIC um circuito lógico é construído a partir da interconexão de um único tipo de célula lógica. Fabricantes disponibilizam uma arquitetura onde um único modelo de célula (chamado de célula base) é distribuído ao longo do CI em posições fixas. A customização da CI lógica é feita pela interconexão dessas células. Os fabricantes disponibilizam ainda bibliotecas chamadas de macro células onde possuem interconexões pré definidas para a criação de blocos mais complexos.

Como a matriz de células básicas são padronizadas para todos os CIs, algumas camadas dos chips podem ser pré fabricados e durante a construção do chip padronizado deve-se somente alterar a camada de metal que especifica as interconexões.

1.1.4 Complex field-Programmable Device - CFPD

Nessa tecnologia, um dispositivo composto de uma matriz de células lógicas genéricas e uma malha de interconexões também genérica é fornecido ao usuário final. O dispositivo é configurável em ambas as partes o que possibilita o desenvolvimento da lógica desejada.

Existem duas categorias principais de dispositivos:

- *Complex Programmable Logic Device* (CPLD), possuem células lógicas complexas e normalmente formadas por flip-flops tipo D e um PAL * configurável. Porém apresentam menor possibilidade de interconexões por terem uma topologia mais centralizado (Fig. 1).
- *Field Programmable Gate Array* (FPGA), diferente dos CPLDs possuem células lógicas mais simples, formadas geralmente por um flip-flop tipo D e uma *look-up-table* ou alguns multiplexadores. As células lógicas são espalhadas por todo o dispositivo (Fig. 2). Devido a essa topologia, FPGAs são mais flexíveis e possibilitam o desenvolvimento de sistemas mais complexos.

*Programmable Array Logic: Consistem de uma pequena memória que pode ser configurada para executar operações lógicas e/ou aritméticas

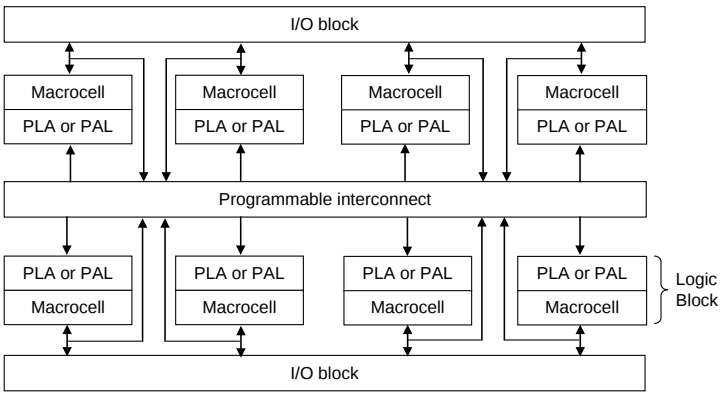

Figura 1: Arquitetura interna de um CPLD
Digital Systems Design with FPGAs and CPLDs - pg. 66

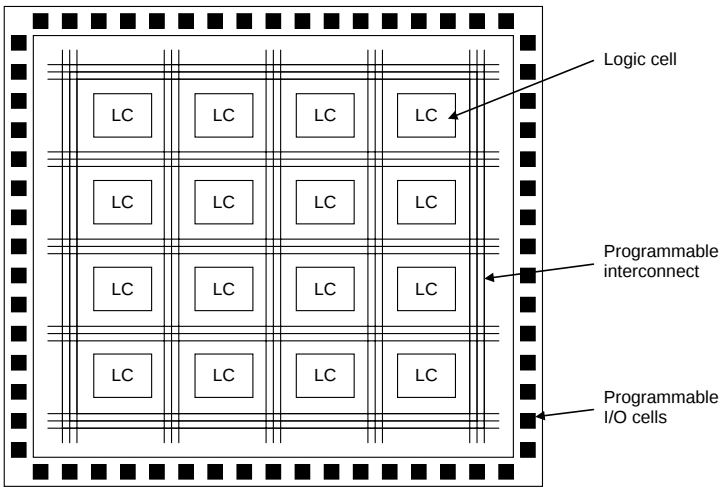

Figura 2: Arquitetura interna de uma FPGA
Digital Systems Design with FPGAs and CPLDs - pg. 68

1.2 Comparativo

A Fig. 3 ilustra o preço da produção de cada tecnologia por unidades produzidas. Como os CPFDs são padronizados, o custo de uma unidade é significativamente mais baixo do que o comparado para uma única unidade do ASIC porém em larga escala esse cenário é diferente, o custo de desenvolvimento do ASIC é diluído ao longo da produção, por exemplo, a fabricação de uma máscara custo em torno de um milhão de dólares, porem uma vez que desenvolvida pode ser utilizada para fabricar grandes escalas de dispositivos. Um exemplo do uso de ASICs são para fabricação de microcontroladores. Já CPFDs são utilizados para aplicações de alto desempenho porem com uma menor produção.

Além do custo, alguns CPFDs podem ser reprogramáveis o que possibilita a correção de bugs, atualizações de versões e uma maior facilidade de desenvolvimento. Diferente-

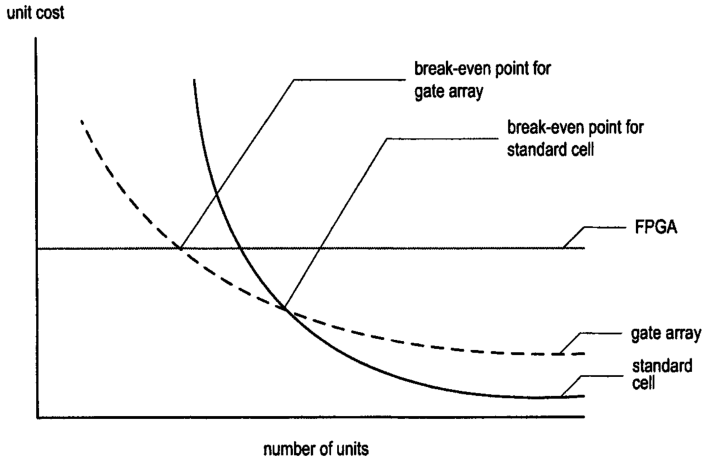

Figura 3: Ilustração da variação de preço x produção de cada tecnologia.
Design Using VHDL Coding for Efficiency, Portability, and Scalability - pg. 31

mente, ASICs não são reprogramáveis e devem seguir um procedimento cuidadoso de desenvolvimento e verificação pois um erro no design acarretara em milhares de unidades problemáticas.

A Fig. 4 é um mapa das tecnologias citadas, a escolha de uma tecnologia deve ser baseada em: Custo; Fornecedor; Tempo de desenvolvimento; Conhecimento prévio da tecnologia; Treinamento; Especificação; Limitações de volume/peso e consumo; Poder de processamento; Flexibilidade; Facilidade com manutenção entre outros.

2 FPGA

Apesar do curso focar no desenvolvimento de lógicas programáveis utilizando FPGA, a maioria das técnicas de projeto e validação tratadas ao longo do curso podem e são utilizadas nas outras tecnologias apresentadas anteriormente.

2.1 Fabricantes

Os principais fabricantes de FPGA* são:

- Xilinx `www.xilinx.com`, com 49% do mercado
- Altera `www.altera.com`, com 40% do mercado
- Lattice Semi `www.latticesemi.com`, com 6% do mercado
- Microsemi `www.microsemi.com`, com 4% do mercado

*Dados de 2013

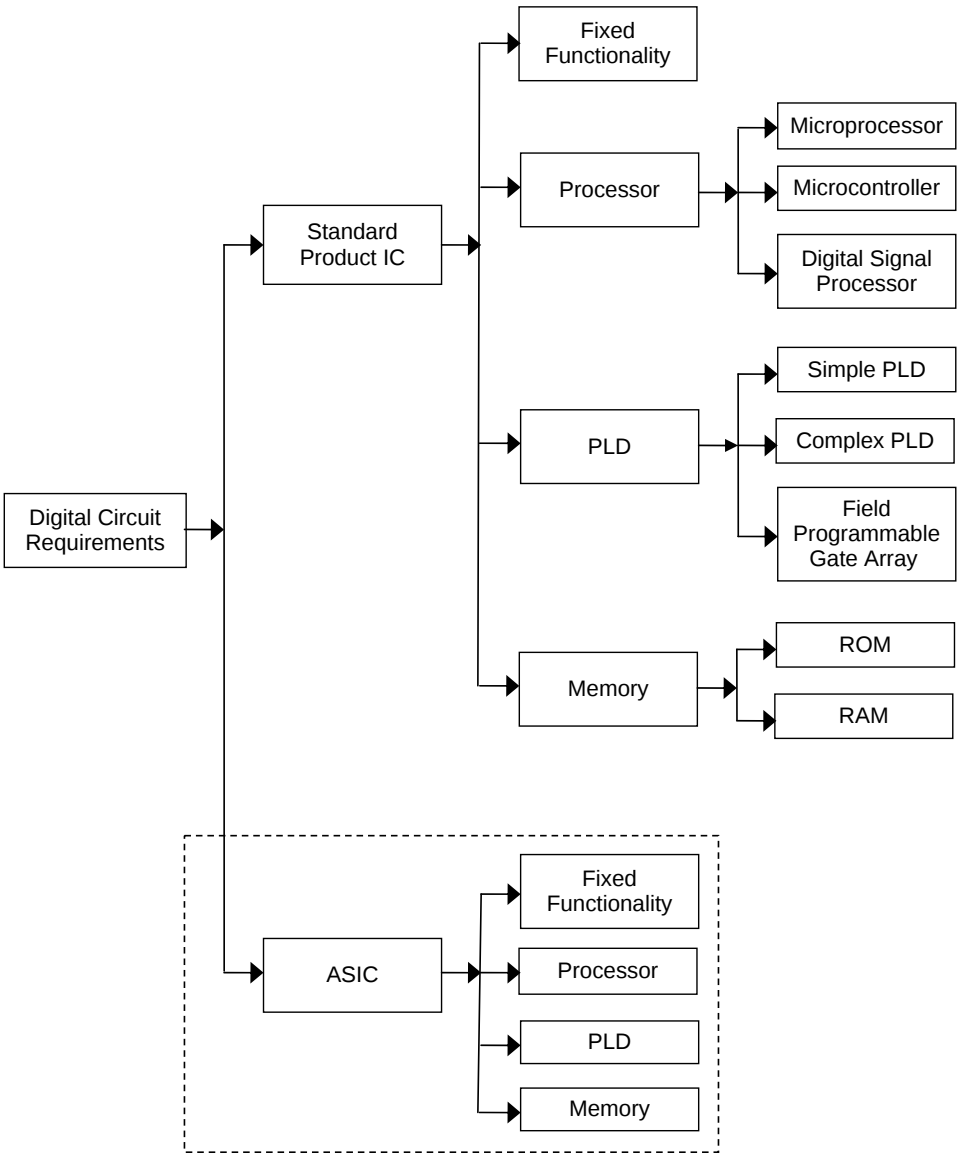

Figura 4: Mapa das tecnologias de projeto de sistemas eletrônicos .
(2008) Digital Systems Design with FPGAs and CPLDs - pg. 41

Os dispositivos fornecidas pelos diferentes fabricantes diferem basicamente na densidade de portas lógicas, velocidade de I/Os e principalmente nas ferramentas de desenvolvimento, sendo essa a maior dificuldade na migração para um novo fabricante.

Um exemplo da competição entre os fabricantes é a disputa entre Xilinx e Altera pelo I/O mais rápido, atualmente (2013) a velocidade de transmissão de um único I/O está em 12.5 Gbps.

As FPGAs utilizadas no curso serão **Xilinx**, devido a sua grande presença no mercado, forte representação no Brasil e boas ferramentas de desenvolvimento. Porem sempre que possível iremos mostrar um paralelo com outros fabricantes.

2.2 Aplicações

FPGAs são utilizadas em aplicações que necessitam de flexibilidade, poder de processamento, grandes quantidades de I/Os e com alto valor agregado. As aplicações vão desde a área médica até aerospaciais, instrumentação, podemos citar alguns exemplos de sua utilização:

- **Médica** : Diagnostico, monitoração e aplicações de terapias;
- **Aerospacial e Defesa** : Navegação, Vigilância, Drones, Radares, Experimentos científicos;

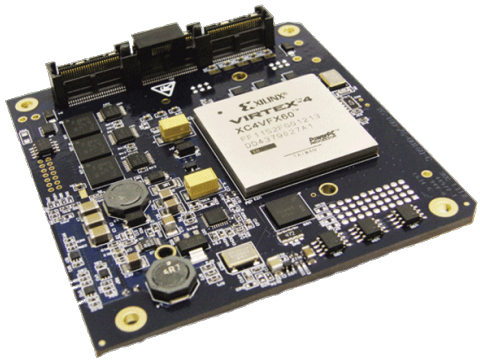

Figura 5: Placa de controle de um Cubsat com uma Xilinx Virtex 4
(2013) www.cubesatshop.com

- **Automotivo :** Módulos de assistência, Conforto e Conveniência;
- **Eletrônica de consumo :** Roteadores ethernet, Decodificadores TV digital;
- **Industria:** Controle de processo, Padrão de qualidade em tempo real;

3 Xilinx

É um empresa americana (`www.xilinx`) fundada em 1984 com faturamento anual de 2.17 bilhões de dólares (2013) foi a criadora da tecnologia FPGA em 1985. Atua atualmente no mercado de CPLDs, FPGAs, softwares de desenvolvimentos e propriedades intelectuais.

3.1 Famílias de FPGA

- **Spartan 6 :** Criada em 2009 é uma classe de baixo custo, baixo consumo energético e voltada para projetos de grandes volumes, produzida em 45nm.
- **Artix 7:** São em geral 50% mais baratas e possuem um consumo 35% menor comparado com a Spartan 6 e são baseadas na arquitetura da família Virtex

3.2 Softwares de desenvolvimento

Os softwares disponibilizados pela Xinix para desenvolvimento de aplicações com suas FPGA são o **ISE** e o **Vivado**. O Vivado é a versão mais atual da ferramenta de desenvolvimento, possuindo suporte somente para FPGAs mais modernas. Ambos possuem interfaces similar e um sistema de simulação nativo (importante para teste e validação). A ferramenta utilizada ao longo do curso será o Vivado.

Na Altera o software de desenvolvimento é chamado de **Quartus II** e possui todas as funcionalidades do corrente porém sem um sistema de simulação nativo, nesse caso opta-se pela utilização do software **ModelSim** que pode ser utilizado para simular projetos com FPGA Xilinx.

A Xilinx disponibiliza seus softwares com uma licença gratuita chamada de *web edition*, essa versão possui algumas limitações em termos de *IP cores* e otimização do projeto.

4 FPGA Arquitetura

4.1 Tipos

Existe basicamente dois tipos de FPGA: as reprogramáveis e as programáveis uma única vez (*One-Time programmable* - OTP). Existem quatro tipos de tecnologias que as FPGAs podem ser construídas:

- **SRAM :** Uma memória externar ou um microcontrolador programam a FPGA na inicialização, possibilita rápida reprogramação; A configuração é volátil (na falta de energia perde-se suas funções previamente carregadas)
- **Anti-Fuse-based :** É uma FPGA com configuração não volátil e não reconfigurável, fusíveis são "queimados" internamente para executar a lógica desejada;
- **EPROM-based :** Comportamento similar a dispositivos EPROM. Possui configuração não volátil e deve ser programada fora do circuito;
- **EEPROM-based :** Comportamento similar a dispositivos EEPROM. Possui configuração não volátil e deve ser programada e reprogramado fora do circuito;

FPGAs volátil levam em torno de milesegundos para carregar sua memória interna, esse tempo é dependente da família utilizada, capacidade da FPGA e velocidade da transmissão. Na configuração de FPGAs SRAM, os dados são normalmente armazenados em uma memória PROM e então carregados para a memória volátil da FPGA por um boot loader interno da FPGA ou por um microcontrolador/PC.

Algumas FPGAs mais modernas possibilitam ser reprogramadas em operação, nessa técnica, defini-se a área desejada para reprogramação enquanto outras zonas da FPGA continuam operando normalmente.

Fabricante	Tecnologia
Xilinx	SRAM
Altera	SRAM, Flash
Actel	Antifuse
Lattice	SRAM, Flash

Tabela 1: Principais famílias de alguns fabricantes
(2005) Rapid System Prototyping with FPGAs Accelerating the Design Process - pg. 23

A Tabela **??** lista as principais tecnologias de alguns fabricantes de FPGA. A Família tratada nesse curso será a SRAM porém sem muitas diferenças de desenvolvimento com as outras.

5 Base SRAM - Arquitetura

Uma FPGA é um circuito integrado (Fig. 6) que possui em seu núcleo diversos blocos lógicas distribuídos (CLB), que podem ser conectadas e combinadas através de uma matriz de roteamento. Nos arredores do chip existe um anel de blocos de I/O que possibilitam acesso aos sinais internos a FPGA (In, out, inout) e que sejam condicionados em diversos níveis lógicos (TTL, CMOS, LVDS, ...). Essa estrutura flexível possibilita que praticamente qualquer lógica síncrona e combinacionais seja implementada em uma FPGA desde que sejam respeitados as restrições de área, potência e temporização. A Fig. 2 ilustra a conexão dos blocos principais elementos de uma FPGA.

Além dos elementos citados anteriormente, as FPGAs possuem memórias internas de uso geral, PLLs, DSPs entre outros. As principais estruturas internas de uma FPGA são:

- Blocos lógicos
- Matriz de Roteamento e Sinais Globais
- Blocos de I/O
- Recursos de clock (PLL)
- Multiplicadores
- Memórias
- Recursos avançados

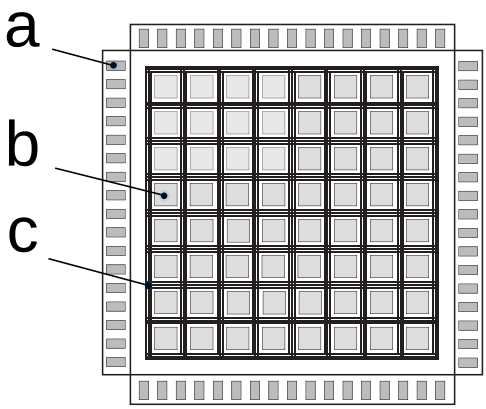

Figura 6: Arquitetura básica de uma FPGA: a: Bloco IO, b: Bloco lógico, c: Matriz de roteamento.
(2005) Rapid System Prototyping with FPGAs Accelerating the Design Process - pg. 41

5.1 Blocos Lógicos

São utilizados na literatura diferente nomes para blocos lógicos comuns (*commum logic block* - CLB) : *logic cell*, *slice*, *macrocell* e *logic element*. A estrutura dos blocos lógicos variam de fabricante e de família porem seguem uma estrutura básica onde contém um look-up-table (LUT) de N entradas, flip flops (elementos de memória), roteamento de sinais, sinais de controle e *carry*. A Fig. 7 ilustra de forma simplificada a arquitetura de um CLB.

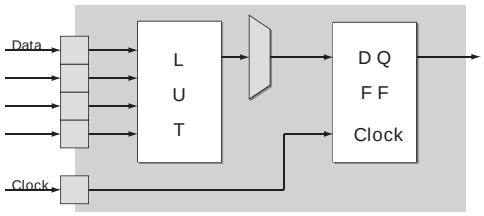

Figura 7: Arquitetura básica de um bloco lógico.
(2005) Rapid System Prototyping with FPGAs Accelerating the Design Process - pg. 42

Para implementações de lógicas mais complexas, utiliza-se um conjunto de CLBs interconectados pela matriz de roteamento. Nota-se que o clock é um fator importante no elemento.

5.1.1 LUT

Cada LUT pode implementar qualquer qualquer função booleana com N ou menos entradas, tradicionamente as LUTs possuem 4 entradas (podendo variar entre chips).

As LUTs são implementadas em elementos de memoria e podem ser utilizadas também como FIFOs (First In First Out).

A Fig. 8 ilustra a implementação da equacão: $(A.C + \bar{B}.\bar{C}).D$ em uma LUT, sendo as entradas A,B,C,D tratadas como endereço de uma memória que possui a solução da lógica armazenada.

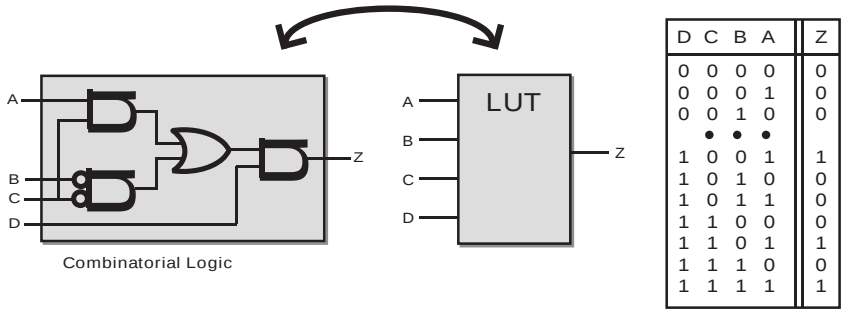

Figura 8: Exemplo de implementação de uma lógica booleana em uma LUT.
(2005) Rapid System Prototyping with FPGAs Accelerating the Design Process - pg. 42

5.2 Blocos de memória

São memórias dedicadas de uma FPGA e podem ser tanto do tipo RAM quanto ROM, e devem receber os mesmos cuidados de uma memória convencional, tais como:

- tempo de acesso
- endereçamento
- atraso na propagação do dado
- síncronas

As memórias Xilinx são classificadas em :

-

5.3 Matrizes de roteamento - *Switch Box*

Os blocos lógicos são conectados através de matrizes de roteamento, chamado normalmente de *Switch Box* (Fig. 9). Possibilitando que logicas mais complexas ou com maior componentes de entrada/saída seja desenvolvida com os CLB. O roteamento é normalmente gerido pelo compilador com pouca interferência do programador, salve nas configurações de copilação.

Devido ao atraso na propagação dos sinais entre blocos, o roteamento influencia diretamente nas constantes de tempo do projeto. Além da implementação da lógica

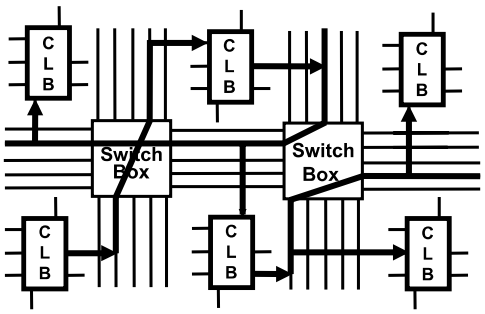

Figura 9: Matrizes de roteamento.
(2010) Handbook of FPGA Design Security- pg. 10

o roteamento deve garantir os atrasos mínimos para que a implementação do projeto garantida um alto valor de *Mean Time Between Failures* (MTBF) .

5.4 Sinais Globais - *Global Signals*

Os sinais globais são "trilhas"disponíveis internamente no chip com um baixo atraso, de uso para propagação de clocks, sinais de enable e reset. O uso dessas “trilhas“ são normalmente interpretadas pelo copilador e a atribuição ocorre de forma transparente ao projetista.

Os sinais globais, principalmente o de clock devido a sua alta frequência, são projetados como uma linha de transmissão onde os fatores de impedância da linha, terminação, são criticamente levados em consideração.

5.5 Blocos de I/O - *I/O Blocks*

Similar aos microcontroladores, na FPGA você pode escolher entre uma ampla gama de codificações de sinais, como por exemplo :

- Low-Voltage CMOS (LVCMOS);
- Low-Voltage TTL (LVTTL);
- IOS específicos para memória;
- IOS específicos para comunicações seriais de alta velocidade (Gbps).

Maiores detalhes podem ser encontrados em : 2005 - *High-Speed Serial I/O Made Simple A Designers' Guide, with FPGA Applications*, Xilinx.

5.6 Conversores AD/DA

Poucas FPGAs possuem conversores digital/analógico (DA) ou analógico/digital (AD) dado que os requisitos desses conversores variam entre projetos como por exemplo sua resolução ou taxa de amostragem. Existem diversas formas de conexão de conversores na FPGA, pode-se utilizar chips com comunicação serial (SPI, I^2C) ou paralelo.

6 Tamanho de uma FPGA

7 Linguagem de descrição de hardware (programação)

A linguagem de programação de dispositivos lógicos programáveis é conceitualmente diferentes daquelas de programação de processadores/microcontroladores uma vez que não se programa instruções previamente definidas, mas sim interconexões entre células lógicas, blocos de memória, níveis de sinais e operações lógicas em **hardware**. Esse tipo de programação de hardware é chamado de *hardware description language* (HDL).

Existem diversos tipos de HDL, mas duas linguagens são predominantes no mercado, o *Verilog* bastante utilizado na América do Norte e o *VHDL* utilizado com mais frequência na Europa e Brasil. O *Verilog* possui uma aproximação maior com a linguagem C, e permite um desenvolvimento mais rápido porém possibilita que erros sejam cometidos com mais facilidade.

O VHDL por sua vez é uma linguagem criada para o governo Americano, primeiramente ela permite a descrição estrutural de um design, ou seja, como esse design é decomposto em sub-designs e como esses sub-designs são interconectados. Consequentemente ela permite a implementação de funções e simulações do design.

8 VHDL

A linguagem de descrição de hardware VHDL (Very High Speed Integrated Circuit (VHSIC) Hardware Description Language) é uma linguagem modular de alto nível para descrição de sistemas digitais. *

Um sistema digital pode ser representado em diferentes níveis de abstração. O nível mais alto de abstração é o nível comportamental, que descreve o sistema em termos do que ele realiza (ou como se comporta) em vez do mesmo ser descrito em termos de seus componentes e interconexões entre os mesmos. Uma descrição comportamental especifica a relação entre os sinais de entrada e saída. Como exemplos de descrição comportamental

*Texto extraído da apostila do Prof. Sergio Ribeiro - IMT: DESENVOLVIMENTO DE PROJETOS USANDO LÓGICA PROGRAMÁVEL - INTRODUÇÃO AO QUARTUS II - ALTERA

podemos citar: expressões “booleanas”, descrições algorítmicas, e aquelas usando lógica de transferência de registradores.

O nível de abstração estrutural descreve um sistema usando uma coleção de portas (“gates”) e componentes que são interconectados para realizar um dada função. Uma descrição estrutural pode ser comparada a um diagrama esquemático. A descrição estrutural normalmente é muito próxima da realização física do sistema.

A figura a seguir ilustra a descrição de um circuito lógico usando o nível de abstração comportamental e o nível estrutural.

Em VHDL, o nível comportamental de um sistema digital pode ser descrito através de declarações seqüenciais e/ou concorrentes. A base para a modelagem seqüencial é a declaração PROCESS, que permite descrever o comportamento do sistema ao longo do tempo usando comandos seqüenciais. Descrição de um sistema através de declarações concorrentes é comumente denominada modelagem por fluxo de dados (“dataflow modelling”). A modelagem por fluxo de dados descreve um circuito em termos de sua função e fluxo de dados através do mesmo. Isto é diferente da modelagem estrutural que descreve um circuito em termos das interconexões de seus componentes. Declarações de atribuição concorrente de sinais são eventos disparados e executados tão logo um evento (mudança de estado) em um sinal ocorra.

No restante deste capítulo serão descritos exemplos usando tanto o nível de abstração comportamental (declarações concorrentes e usando PROCESS) quanto o estrutural. Serão descritos exemplos de circuitos combinatórios, seqüenciais e máquinas de estado