Sistemas Embutidos (Avançados)

Síntese em *Hardware* Reconfigurável

Rodolfo Labiapari Mansur Guimarães

Última Atualização: 3 de outubro de 2016

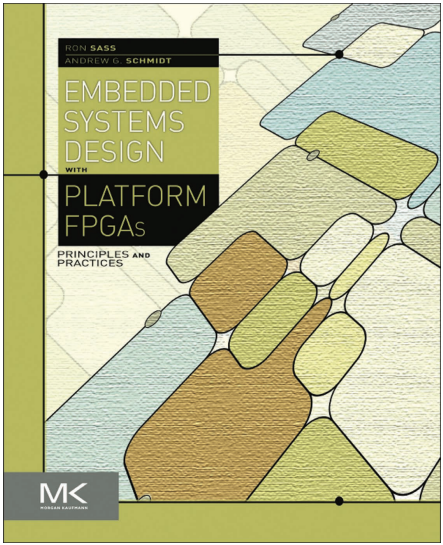
rodolfolabiapari@gmail.com
Lattes: `http://goo.gl/MZv4Dc`
Departamento de Computação – Universidade Federal de Ouro Preto
Ouro Preto - MG – Brasil

Livro Texto:

Embedded Systems Design with Platform FPGAs [1]

UFOP

Figura 1: Livro Texto: *Embedded Systems Design with Plataform FPGAs: Principles and Practices*.



O Que Será Abordado?

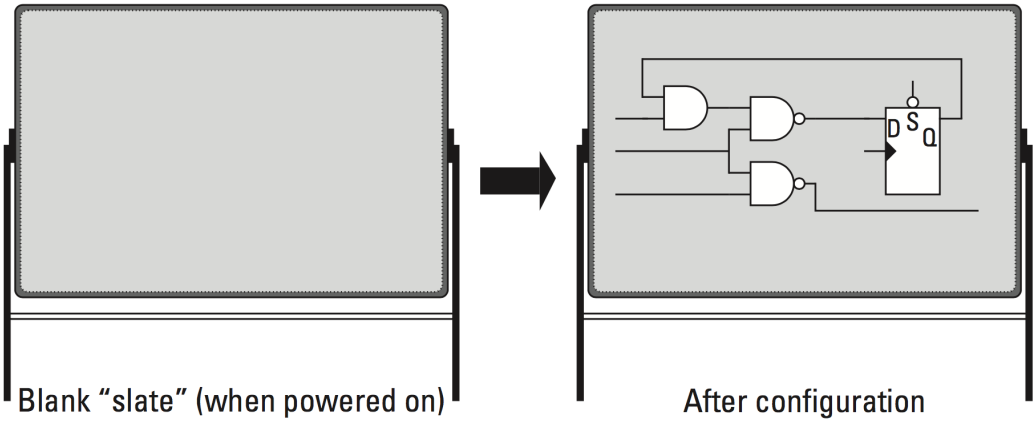
1. *Programmable Logic Devices* (PLD);
2. Propriedades do FPGA;
3. Escrita em VHDL para FPGAs.

UFOP

Introdução

UFOP

Figura 2: "FPGA é um **quadro negro** que pode ser **desenhado** qualquer circuito digital".



Embedded Systems

UFOP

Embedded Systems - Modelos de Execução

- Um dos maiores desafios do *design* de um sistema embarcado é o fato de ter *hardware* e *software* no *design* do projeto como um todo.
- FPGA é um *hardware* que é 'escrito como *software*'.
- Por exemplo: um *software* é executado de forma sequencial e com o resultado as operações são realizadas em ordem. Entretanto, circuitos digitais são naturalmente paralelo e engenheiros devem explicitar a sincronização disso no *design*.
- Para concretizar, suponha `ACC = R0+R1+R2` sendo todos os quatro, registradores.

UFOP

Figura 3: Execução **sequencial** de `ACC = R0+R1+R2`.

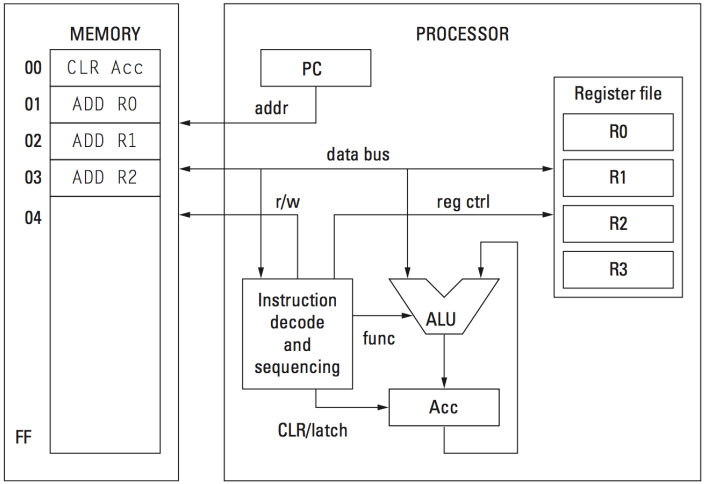


Figura 4: Execução da operação `ACC = R0+R1+R2` em **fluxo de dados**.

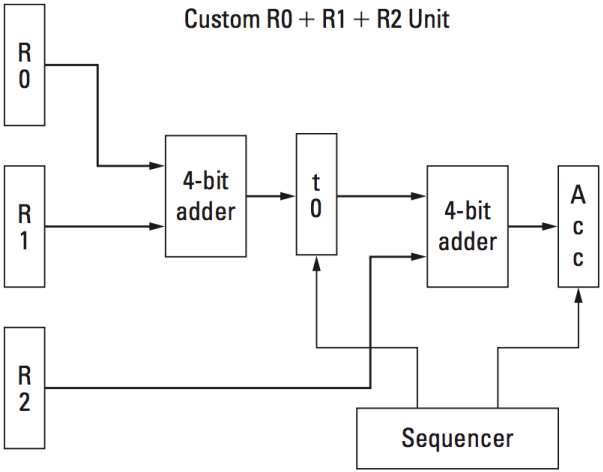


Figura 5: Comparação entre ambas as execuções sobre a operação `ACC = R0+R1+R2`.

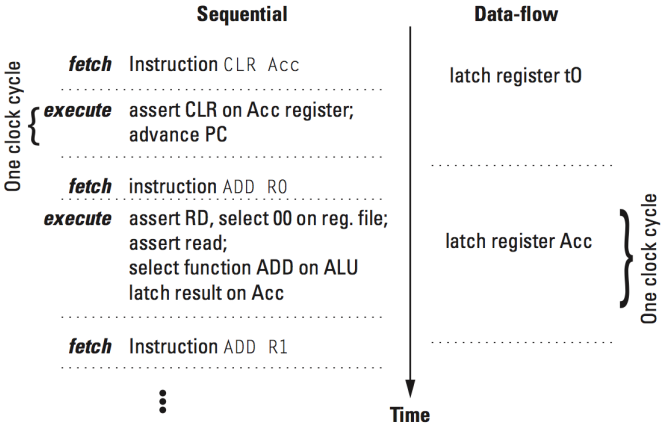
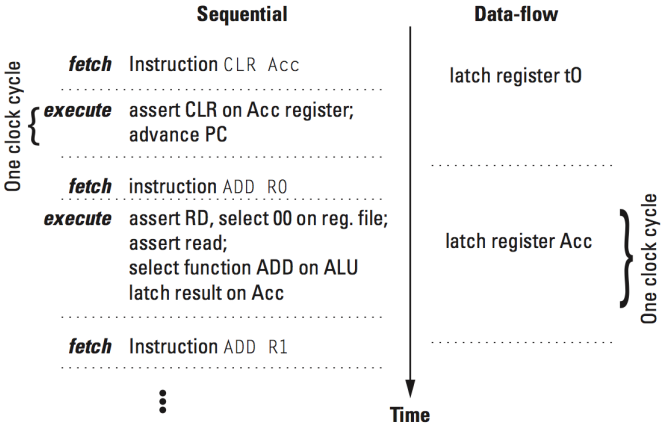


Figura 5: Comparação entre ambas as execuções sobre a operação `ACC = R0+R1+R2`.



Questão

Processador executando de modo sequencial e *software executando como um processador paralelo*?

Programmable Logic Devices (PLD)

Programmable Logic Devices - Introdução

- Transistores são **afixados** na placa no ato da fabricação.
  - Uma vez afixado na placa, sua **funcionalidade** também torna **fixa**.

Questão

Como pode-se construir um dispositivo que seja (re)configurado mesmo **após** sua fabricação?

UFOP

Programmable Logic Devices - Introdução

- Transistores são **afixados** na placa no ato da fabricação.
  - Uma vez afixado na placa, sua **funcionalidade** também torna **fixa**.

Questão

Como pode-se construir um dispositivo que seja (re)configurado mesmo **após** sua fabricação?

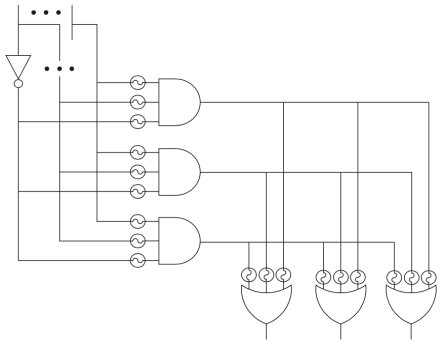
Resposta

Necessita-se de uma estrutura que suporte: 1) implementação de circuitos combinacionais arbitrários, 2) uma memória de armazenamento e 3) mecanismo de conexão entre eles.

Programmable Logic Devices - Programmable Logic Arrays

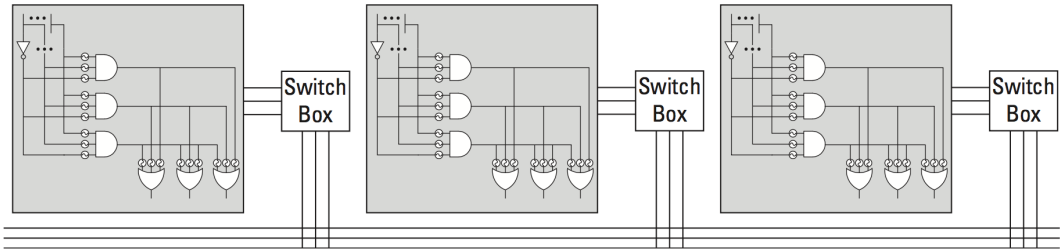
- Primeiro circuito desenvolvido.
- Utilizava a formulação de Soma de Produtos de uma função Booleana.
  - Possui um vetor de várias entradas AND e um vetor de OR.

Figura 6: Abstração do *Programmable Logic Arrays*.



- Logo em seguida, desenvolveu-se o *Complex Programmable Logic Devices* (CPLDs) que replicava os blocos PLA

Figura 7: Abstração do *Complex Programmable Logic Devices*.



Field-Programmable Gate Array (FPGA)

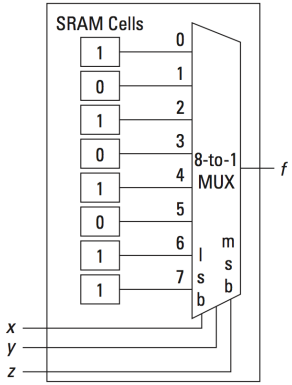
UFOP

- Supondo a implementação da função Booleana $f(x, y, z) = xy + z'$.

Figura 8: Gerador de Função. a) tabela verdade e b) Look-Up Table (LUT)

x	y	z	$xy + z'$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

(a)



(b)

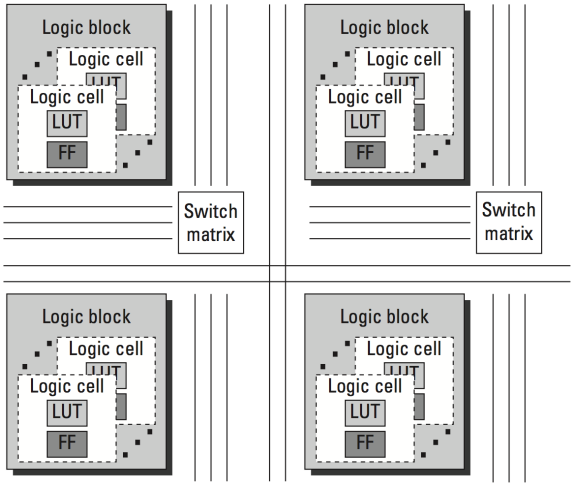
Field-Programmable Gate Array (FPGA) - Gerador de Funções

- Se cada saída é gravada numa memória, então basta fazer um multiplexador 8×1.
- É possível perceber que o *delay* de cada LUT é sempre constante (o que difere dos circuitos digitais).
  - Independente da complexidade dos circuitos, o atraso de propagação permanece sempre o mesmo.
- Para implementar uma função que utilize mais entradas que uma LUT:
  - Utiliza-se várias LUTs.
  - Cada LUTs torna resultado de uma composição.
    - Sub-funções possuem sub-conjuntos de entradas.
- Para isso, existe recursos no FPGA para realizar a conexão entre LUTs vizinhas com o mínimo de *delay*.
- Lembrando sempre que: **SRAM são voláteis**. Quando o FPGA é desligado, **tudo é perdido**. E essa capacidade de **re-escrita** é que permite sua 'reconfiguração'.

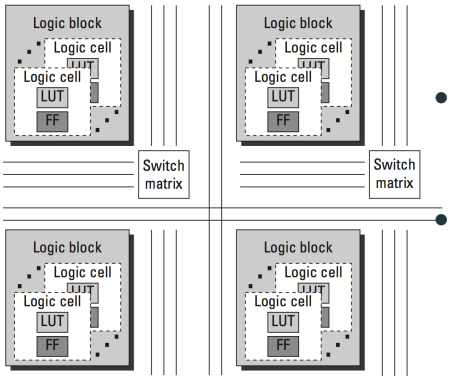
Field-Programmable Gate Array (FPGA) - Elementos de Armazenamento

- Flip-flops D são incorporados no FPGA.
- Tipicamente, funcionam como se fosse saídas do Gerador de Função mencionado anteriormente.
- Também, são utilizados de várias maneiras diferentes de acordo com o projeto.

Figura 9: Células Lógicas, Blocos Lógicos e Rede de roteamento.

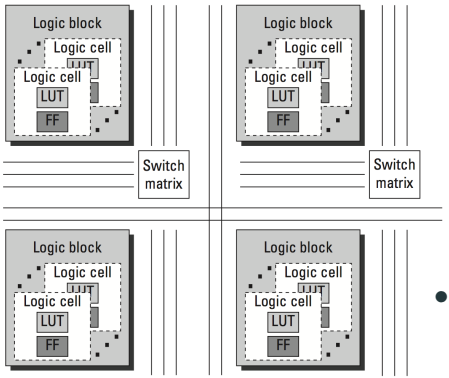


Field-Programmable Gate Array (FPGA) - Componentes

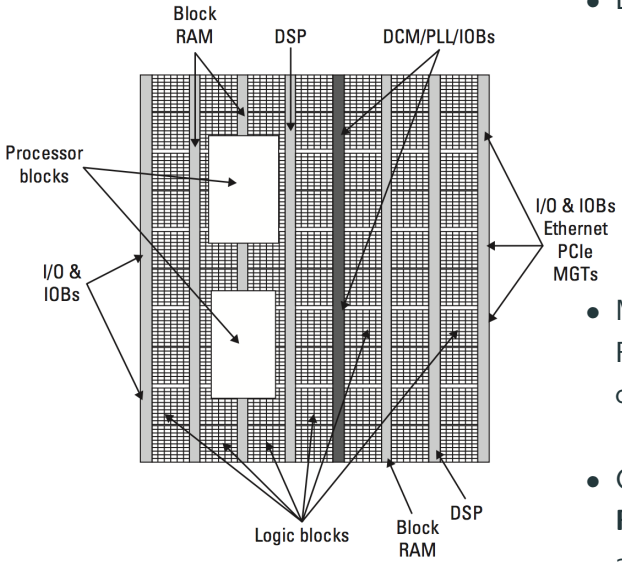


- Células e **Blocos Lógicos** constituem do nível mais baixo de *hardware* do FPGA.
- Blocos combinacionais/sequenciais constituem células lógicas.
- Blocos Lógicos podem ser utilizados para circuitos de propósito específicos como uma pequena ALU.
  - Permite que grupos lógicos de célula sejam geograficamente próximos permitindo caminhos de comunicação curtos e rápidos, e principalmente curto na implementação do *design*.

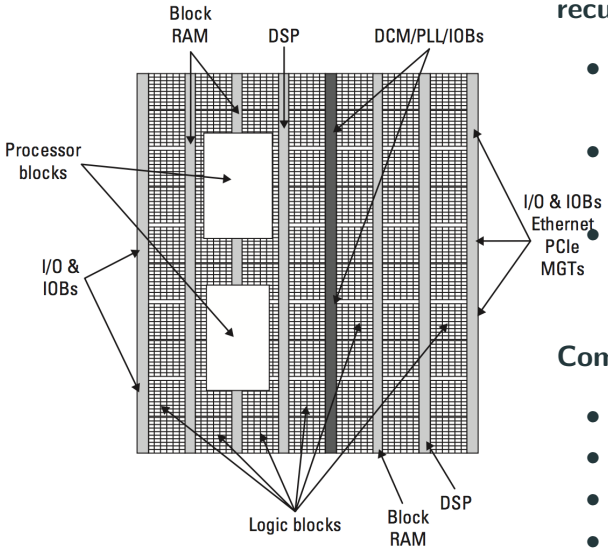
Field-Programmable Gate Array (FPGA) - Componentes



- Blocos lógicos são conectados por meio de **Redes de Roteamento** que provê suporte até para circuitos complexos.
  - Roteia entradas e saídas dos blocos até o rede de roteamento geral.
  - Realiza a passagem de sinais de um cabo para outro de outro segmento (sendo o segmento curto ou longo).
- Como circuitos geralmente utilizam muitos blocos, eles são postos em cadeia proporcionando uma implementação potencialmente mais rápida e fácil de ser sintetizada.



- E o último é o **Bloco de I/O** (pinos).
  - Ficam localizados no aborda do circuito e conectam blocos lógicos e redes de roteamento com o mundo externo.
  - Cada pino possui sua característica especial.
- Muitos fabricantes comparam seus FPGAs por meio de quantidade de células disponíveis.
- Com o **Bloco Lógico**, **Rede de Roteamento** e **Pinos** temos uma abstração do FPGA.



FPGA combina lógica programável com recursos embarcados. Porquê?

- Recursos embarcados consomem menos recursos do FPGA.
- Recursos embarcados consomem menos energia.
- Recursos embarcados podem operar com altíssima frequência.

Componentes

- *Block RAM*.
- *Digital Signal Processing Blocks*.
- *Processors*.
- *Digital Clock Manager*.
- *Multi-Gigabit Transceivers*.

Hardware Description Languages (HDL)

UFOP

Hardware Description Languages

- Meio utilizado para a 'configuração' de um *hardware*.
- Linguagem de alto nível para descrição de circuitos digitais.
- Sua intenção inicial era para documentação e não para sintetização.
- Com o passar do tempo, foi possível ver que as linguagens poderiam ser utilizadas para simulação e consequentemente para síntese.
  - Entretanto, enquanto a simulação provê um grande conjunto de recursos pra auxiliar o projetista, o mesmo não acontece na síntese.
  - Enquanto muitas linguagens permitem a simulação, poucas fornecem recurso para síntese. Duas delas são Verilog e VHDL.
- Existem linguagens para aumentar a produtividade como SystemC, HandelC e Impulse.
  - Algumas são extensões de bibliotecas como C/C++, enquanto outras como C-like provê um ambiente mais amigável.

Hardware Description Languages - VHSIC Hardware Description Language

- **VHDL: V***HSIC* **H***ardware* **D***escription* **L***anguage*.
  - **VHSIC: V***ery* **h***igh-***s***peed* **i***ntegrated* **c***ircuit*.

- No VHDL, existe duas formas/estilos de escrever. Cada modelo tem seus impactos na síntese, simulação e em alguns caso até na produtividade. As formas são:
  - **Estrutural/Circuitos de Fluxo de Dados:** Descritos usando termos lógicos e sinais tal como AND, OR, Multiplicadores.
  - **Circuitos Comportamentais:** Descreve em uma linguagem imperativa procedural como cada saída é relacionada com cada entrada num processo. Ideal para sistemas complexos, entretanto é possível descrever comportamentos impossíveis.

- Exemplo de 1-bit somador completo:

Figura 10: Um 1-bit *full adder* em **Tabela Verdade**.

Row	Inputs			Outputs		Comment
	x	y	c_{in}	c_{out}	s	
0	0	0	0	0	0	$0+0+0=00_2$
1	0	0	1	0	1	$0+0+1=01_2$
2	0	1	0	0	1	$0+1+0=01_2$
3	0	1	1	1	0	$0+1+1=10_2$
4	1	0	0	0	1	$1+0+0=01_2$
5	1	0	1	1	0	$1+0+1=10_2$
6	1	1	0	1	0	$1+1+0=10_2$

Figura 11: Um 1-bit *full adder* em **Circuito**.

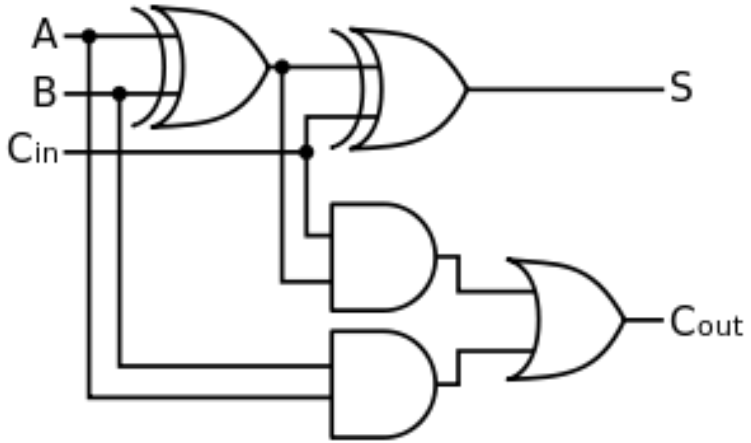


Figura 12: Um 1-bit *full adder* em **VHDL**.

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

entity fadder is port (
  a, b : in std_logic;
  cin  : in std_logic;
  sum  : out std_logic;
  cout : out std_logic);
end fadder;

architecture beh of fadder is
begin
  sum  <= a xor b xor cin;
  cout <= (a and b) or (b and cin) or (a and cin);
end beh;
```

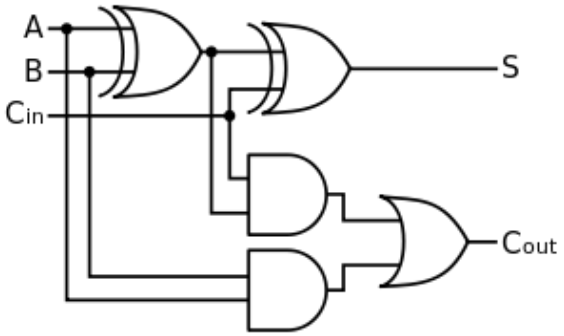
Figura 13: Um 1-bit full adder em **VHDL**.

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

entity fadder is port (
  a, b : in std_logic;
  cin  : in std_logic;
  sum  : out std_logic;
  cout : out std_logic);
end fadder;

architecture beh of fadder is
begin
  sum  <= a xor b xor cin;
  cout <= (a and b) or (b and cin) or (a and cin);
end beh;
```

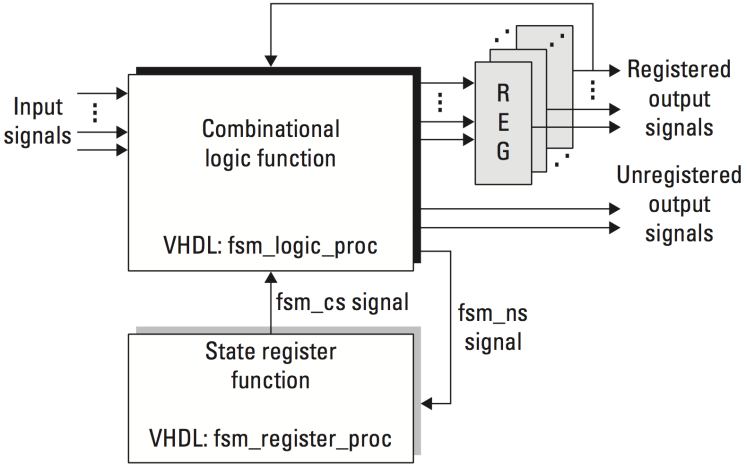
Figura 14: Um 1-bit full adder em **Circuito**.



- Útil para vários propósitos.
- Diferentes formas de escrita resultam em diferentes sínteses.
- Recomenda-se seguir o guia de síntese, isto é, *Xilinx Synthesis Technology (XST) User Guide*.
- Exemplo de somador de 8-bit:

Livro, página 64.

Figura 15: Exemplo abstrato de uma máquina de estado.



Sistemas Embutidos (Avançados)

Síntese em *Hardware* Reconfigurável

Rodolfo Labiapari Mansur Guimarães

Última Atualização: 3 de outubro de 2016

rodolfolabiapari@gmail.com
Lattes: `http://goo.gl/MZv4Dc`
Departamento de Computação – Universidade Federal de Ouro Preto
Ouro Preto - MG – Brasil

Bibliografia

UFOP


R. Sass and A. Schmidt.
Embedded Systems Design with Platform FPGAs Principles and Practices.
Morgan Kaufmann, 1 edition, 2010.

Sistemas Embutidos (Avançados)

Síntese em *Hardware* Reconfigurável

Rodolfo Labiapari Mansur Guimarães

Última Atualização: 3 de outubro de 2016

rodolfolabiapari@gmail.com
Lattes: `http://goo.gl/MZv4Dc`
Departamento de Computação – Universidade Federal de Ouro Preto
Ouro Preto - MG – Brasil