

Calculadora Simples em VHDL

Versão 2015

RESUMO

Esta experiência consiste no projeto e implementação de um circuito digital simples com o uso de uma linguagem de descrição de hardware. São apresentados aspectos básicos da linguagem VHDL e exemplos de descrição de circuitos.

OBJETIVOS

Após a conclusão desta experiência, os seguintes tópicos devem ser conhecidos pelos alunos:

- *Linguagem de descrição de hardware;*
- *VHDL;*
- *Projeto com FPGA.*

1. PARTE TEÓRIA

1.1. Linguagens de Descrição de Hardware

Uma alternativa à entrada esquemática de um circuito digital em um sistema de projeto auxiliado por computador é utilizar uma ferramenta de projeto baseado em texto ou linguagem de descrição de hardware (HDL). Exemplos de HDLs são o AHDL (*Altera Hardware Description Language*) e os padrões VHDL e Verilog.

O projetista cria um arquivo de texto, seguindo certo conjunto de regras, conhecido como sintaxe da linguagem, e usa um compilador para criar dados de programação do dispositivo lógico programável (PLD). Esta descrição de hardware pode ser usada para gerar projetos hierárquicos, ou seja, um componente definido em uma descrição pode ser usado para gerar um hardware específico ou ser usado como parte de outro projeto.

As HDLs têm uma grande semelhança às linguagens de programação, mas são especificamente orientadas à descrição da estrutura e do comportamento do hardware. Uma grande vantagem das HDLs em relação à entrada esquemática é que elas podem representar diretamente equações booleanas, tabelas verdade e operações complexas (p.ex. operações aritméticas).

Uma descrição estrutural descreve a interconexão entre os componentes que fazem parte do circuito. Esta descrição é usada como entrada para uma simulação lógica da mesma forma que uma entrada esquemática.

Uma descrição comportamental descreve o funcionamento lógico de cada um dos componentes do circuito. Em relação ao tipo de descrição anterior, a descrição comportamental é realizada em um nível de abstração mais alto.

Uma HDL pode ser usada na descrição em vários níveis do circuito em desenvolvimento. Partindo de uma descrição de alto nível, pode ser usada para refinar e particionar esta descrição em outras de nível mais baixo durante o processo de desenvolvimento. A descrição final deve conter componentes primitivos e blocos funcionais.

Uma grande razão para o uso de HDLs é a síntese lógica. Uma descrição em HDL em conjunto com uma biblioteca de componentes é usada por uma ferramenta de síntese para a geração automática de um circuito digital. Além disto, estas ferramentas incluem uma etapa de otimização da lógica interna do circuito gerado, antes da geração das estruturas internas de armazenamento, da lógica combinatória e da estrutura de conexão dos componentes (*netlist*). A figura 1.1 abaixo mostra um diagrama mostrando as etapas principais de síntese lógica.

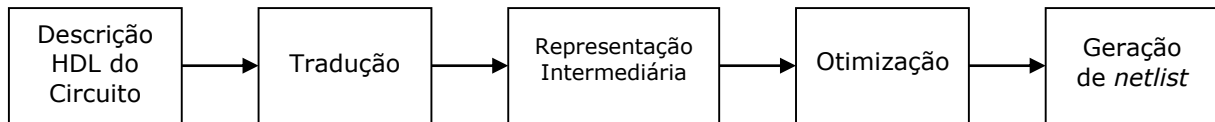


Figura 1.1 - Fluxo das etapas de alto nível da Síntese Lógica.

Atualmente, as HDLs mais utilizadas são o VHDL e o Verilog. Ambas as linguagens são hoje padrões aprovados e publicados pelo IEEE (Instituto dos Engenheiros Elétricos e Eletrônicos), tendo portanto várias ferramentas comerciais disponíveis. Esta padronização leva a uma grande vantagem no desenvolvimento de circuitos usando HDLs: a **portabilidade**. Visto que as ferramentas devem implementar as características padronizadas, fica muito fácil usar sistemas de desenvolvimento de fabricantes diferentes.

1.2. A Linguagem VHDL

O nome VHDL é um acrônimo de “*VHSIC Hardware Description Language*”. Já o termo VHSIC é o acrônimo de “*Very High Speed Integrated Circuit*”. Assim podemos traduzir, de forma literal, o nome VHDL como “linguagem de descrição de hardware para circuitos integrados de velocidade muito alta”.

A linguagem VHDL foi originalmente desenvolvida por empresas contratadas pelo governo americano e agora é um padrão requerido por todos os ASICs (*Application Specific Integrated Circuits*) projetados para o exército americano. Ele foi padronizado pelo IEEE em 1987 (Padrão 1076-1987 ou VHDL 87) e atualizado posteriormente em 1993, em 2003 e em 2008 (Padrão IEEE 1076-2008).

Todo arquivo VHDL requer ao menos duas estruturas: uma declaração de **entidade** e uma **arquitetura**. A declaração de entidade define os aspectos externos da função VHDL, isto é, os nomes e tipos das entradas e saídas e o nome da função. A arquitetura define os aspectos internos, isto é, como as entradas e saídas influem no funcionamento e como se relacionam com outros sinais internos.

Um exemplo de uma descrição VHDL é mostrado a seguir.

```

1  -- somador de 4 bits: descrição comportamental VHDL
2  library ieee;
3  use ieee.std_logic_1164.all;
4  use ieee.std_logic_unsigned.all;
5
6  entity somador_4bits is
7  port(B, A : in std_logic_vector(3 downto 0);
8        C0 : in std_logic;
9        S : out std_logic_vector(3 downto 0);
10       C4 : out std_logic);
11 end somador_4bits;
12
13 architecture comportamental of somador_4bits is
14 signal soma : std_logic_vector(4 downto 0);
15 begin
16   soma <= ('0' & A) + ('0' & B) + ("0000" & C0);
17   C4 <= soma(4);
18   S <= soma(3 downto 0);
19 end comportamental;
  
```

Os sinais de entrada e saída são do tipo `std_logic` e `std_logic_vector`. O tipo `std_logic` é definido no pacote `ieee.std_logic_1164` e pode assumir os valores 'U', 'X', '0', '1', 'Z', 'W', 'L', 'H' ou '-'. O tipo `std_logic_vector` representa um vetor de bits. O somador possui 3 sinais de entrada: A, B e C0, e 2 sinais de saída: S e C4.

São usados dois operadores: **+** representa uma adição e **&** representa uma concatenação de bits. Assim, `'0' & A` representa um vetor de 5 bits contendo '0' A(3) A(2) A(1) A(0).

¹ Se o objetivo for a síntese do circuito em uma FPGA, somente os valores 0, 1 e Z são relevantes. Os outros valores são interessantes nas etapas de simulação para verificação da correteza do projeto.

Após a realização de uma operação de adição com os sinais de entrada, a descrição atribui o bit mais significativo de soma a C4 (vai-um) e os outros bits em S (soma).

A descrição acima é um exemplo de uma **descrição comportamental**. O texto apresenta elementos lógicos de um somador binário e a ferramenta de projeto é responsável pela síntese do circuito digital (figura 1.2).

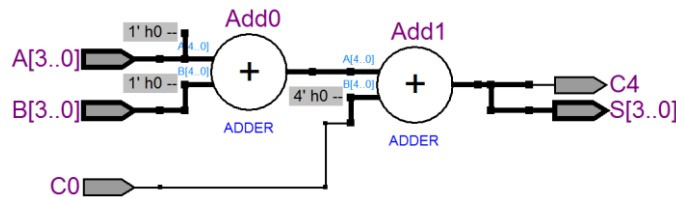


Figura 1.2 – Visão RTL do circuito sintetizado pelo Altera Quartus II.

Uma **descrição estrutural** do somador binário de 4 bits equivalente ao exemplo anterior deve levar em conta os elementos internos do circuito digital. Por exemplo, podemos considerar este somador de 4 bits contendo 4 somadores completos de 1 bit interligados entre si (figura 1.3).

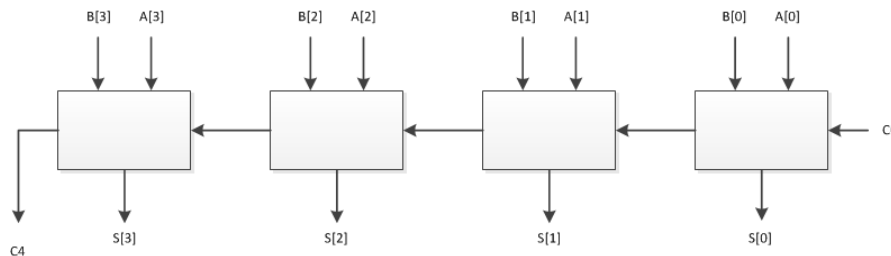


Figura 1.2 – Somador binário de 4 bits composto por 4 somadores completos de 1 bit.

Inicialmente apresentamos a descrição do somador completo de 1 bit de forma comportamental. Esta descrição é mostrada abaixo (figura 1.3).

```

1  -- somador completo: descrição comportamental
2  library ieee;
3  use ieee.std_logic_1164.all;
4
5  entity somador_completo is
6  port (a, b: in std_logic;
7        cin : in std_logic;
8        s   : out std_logic;
9        cout: out std_logic);
10 end somador_completo;
11
12 architecture dataflow of somador_completo is
13 begin
14     s <= a xor b xor cin;
15     cout <= (a and b) or (a and cin) or (b and cin);
16 end dataflow;

```

Figura 1.3 – Descrição comportamental VHDL de um somador completo de 1 bit.

Uma vez disponível, o circuito do somador completo pode ser reutilizado para a descrição do somador binário de 4 bits. Para isto, deve ser definido como um componente e usado como um bloco básico: devem ser instanciados 4 elementos e suas entradas e saídas devem ser interligadas através de sinais internos. A seguir apresentamos o código VHDL do somador binário de 4 bits em descrição estrutural (figura 1.4).

```

1  -- somador de 4 bits: descrição estrutural
2  library ieee;
3  use ieee.std_logic_1164.all;
4
5  entity somador_4bits_estrutural is
6  port(a, b : in std_logic_vector(3 downto 0);
7        c0 : in std_logic;
8        s : out std_logic_vector(3 downto 0);
9        c4 : out std_logic);
10 end somador_4bits_estrutural;
11
12
13 architecture estrutural of somador_4bits_estrutural is
14     signal vai_um : std_logic_vector(0 to 2); -- vetor de 3 bits
15
16     component somador_completo is
17     port(a, b: in std_logic;
18         cin : in std_logic;
19         s : out std_logic;
20         cout: out std_logic);
21     end component;
22
23 begin
24     S1: somador_completo port map(a(0),b(0),c0,s(0),vai_um(0));
25     S2: somador_completo port map(a(1),b(1),vai_um(0),s(1),vai_um(1));
26     S3: somador_completo port map(a(2),b(2),vai_um(1),s(2),vai_um(2));
27     S4: somador_completo port map(a(3),b(3),vai_um(2),s(3),c4);
28 end estrutural;

```

Figura 1.4 – Descrição VHDL estrutural de um somador binário de 4 bits composto por 4 somadores completos de 1 bit.

Para o desenvolvimento de circuitos sequenciais, a linguagem VHDL possui recursos específicos. O trecho de código abaixo apresenta a descrição de um flip-flop tipo D sensível à borda de subida com entrada de reset assíncrono (figura 1.5).

```

1  -- flip-flop tipo D com clear assíncrono
2  library ieee;
3  use ieee.std_logic_1164.all;
4  --
5  entity ffD is
6  port ( CLK, CLR, D : in std_logic;
7        Q : out std_logic );
8  end ffD;
9  --
10 architecture ffD_arq of ffD is
11 begin
12     --
13     process (CLK, CLR)
14     begin
15         if CLR='1' then
16             Q <= '0';
17         elsif CLK'event and CLK='1' then
18             q <= D;
19         end if;
20     end process;
21     --
22 end ffD_arq;

```

Figura 1.5 – Descrição VHDL de um flip-flop D.

O trecho de código na especificação da arquitetura inclui um **processo**. Esta construção da linguagem representa um módulo de circuito sensível aos sinais CLK e CLR: quando qualquer um destes sinais for modificado, o código é “executado”. Inicialmente o sinal CLR é verificado e se assumir valor 1, a saída Q é resetada. Em caso contrário, se houve uma borda de subida de CLK, a entrada D é copiada para a saída².

² Ao se analisar este trecho de código, pode parecer que houve um erro de digitação com a presença dos sinais Q e q. Não é o caso, pois o VHDL não faz distinção entre maiúsculas e minúsculas. Então tanto Q como q representam o mesmo sinal no circuito.

Em uma descrição VHDL, todos os comandos são concorrentes, ou seja, o processamento deles é simultâneo. Dentro de um processo, os **comandos** são **sequenciais**, ou seja, seguem a mesma sequência de execução de comandos em uma linguagem de programação tradicional. Assim, na descrição do *flip-flop* D acima, a saída Q é modificada somente depois do teste do sinal CLR.

Um exemplo mais complexo é um registrador deslocador bidirecional com funções de reset assíncrono e carga paralela (figura 1.6).

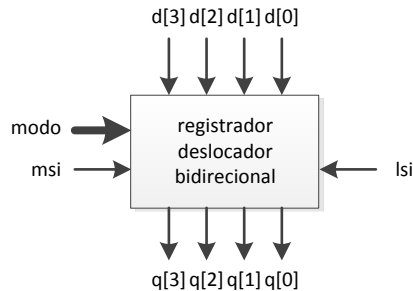


Figura 1.6 – Registrador deslocador bidirecional.

O código VHDL é composto de um processo apenas e utiliza um sinal interno chamado **estado** para armazenar o conteúdo intermediário das operações do deslocador. Para as operações de deslocamento de bits, usa-se o operador de concatenação de bits **&**. A execução dos deslocamentos para a direita e para a esquerda concatenam os bits relevantes do deslocador e realizam a atribuição ao sinal estado. A seguir temos a descrição completa do deslocador bidirecional (figura 1.7).

Mais detalhes sobre a linguagem VHDL podem ser obtidos na referência [D'Amore, 2012].

1.3. Aritmética Binária

Quando se deseja implementar um circuito digital que realize operações aritméticas, é necessário fazer algumas considerações em como executar estas operações sobre dados binários. Normalmente, quando se considera valores com sinal, adotamos a notação de complemento de dois. Sabe-se que numa operação aritmética em complemento de dois, não se corrige o resultado como no caso da notação em complemento de um. É necessário, porém, somar-se 1 ao complemento bit a bit do número:

$$(\text{Complemento de } 2) = (\text{Complemento de } 1) + 1.$$

Numa subtração, portanto, costuma-se “forçar” um “vem-um” na coluna de bits menos significativos dos operandos:

Exemplos:

$$7 - 3 = 4$$

				1	“vem-um” forçado
0	1	1	1		7
1	1	0	0		-3 (complemento de 1)
0	1	0	0		4

$$4 - 6 = -2$$

				1	“vem-um” forçado
0	1	0	0		4
1	0	0	1		-6 (complemento de 1)
1	1	1	0		-2 (complemento de 2)

Numa soma de dois números de mesmo sinal (positivos ou negativos), a complementação não é necessária e, portanto, não há “vem-um” forçado.

Exemplo:

$$-2 + (-3) = -5$$

1	1	1	0	-2 (complemento de 2)
1	1	0	1	-3 (complemento de 2)
1	0	1	1	-5 (complemento de 2)

```

1  -- registrador deslocador bidirecional com carga paralela
2  -- entrada de modo:
3  --    00: hold
4  --    01: load
5  --    10: shift left
6  --    11: shift right
7  --
8  -- msi: entrada serial mais significativa
9  -- lsi: entrada serial menos significativa
10
11 library ieee;
12 use ieee.std_logic_1164.all;
13
14 entity shift_reg is
15
16     port (
17         d           : in  std_logic_vector (3 downto 0);
18         modo        : in  std_logic_vector (1 downto 0);
19         clk, reset_n : in  std_logic;
20         msi, lsi    : in  std_logic;
21         q           : out std_logic_vector (3 downto 0));
22
23 end shift_reg;
24
25 architecture comportamental of shift_reg is
26
27     signal estado : std_logic_vector (3 downto 0);
28
29 begin -- comportamental
30
31     q <= estado; -- muda saída.
32
33     process (clk, reset_n)
34     begin
35         if reset_n = '0' then -- reset assíncrono (ativo baixo)
36             estado <= "0000";
37         elsif clk'event and clk = '1' then -- na borda de subida do clock
38             if modo = "00" then -- Hold.
39                 estado <= estado;
40             elsif modo = "01" then -- Load.
41                 estado <= d;
42             elsif modo = "10" then -- Shift left.
43                 estado <= estado (2 downto 0) & lsi;
44             else -- Shift right.
45                 estado <= msi & estado (3 downto 1);
46             end if;
47         end if;
48     end process;
49
50 end comportamental;

```

Figura 1.7 – Descrição VHDL de um registrador deslocador bidirecional.

A figura 1.8 mostra um circuito de soma/subtração em complemento de 2.

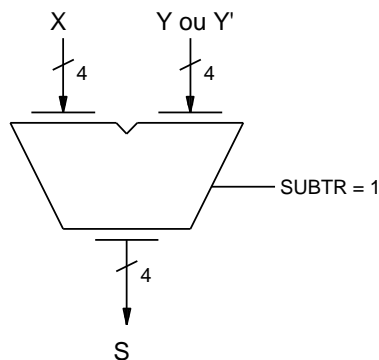




Figura 1.8 - Circuito de Soma / Subtração em Complemento de 2.

As operações de multiplicação e divisão podem ser implementadas em um circuito digital através de um processo iterativo que envolve diversos passos computacionais [Midorikawa, 2004] [Tocci & Widmer, 2011]. Por exemplo, a figura 1.9 abaixo ilustra um exemplo de multiplicação.

Figura 1.9 – Exemplo de Multiplicação Binária calculada de forma iterativa.

13	1101	multiplicando
<u>11</u>	<u>1011</u>	multiplicador
	0000	valor inicial do produto parcial
+	<u>1101</u>	soma multiplicando, bit do multiplicador é 1
	1101	
	0110 1	desloca para a direita
+	<u>1101</u>	soma multiplicando, bit do multiplicador é 1
1	0011 1	
	1001 11	desloca para a direita
	0100 111	só desloca para a direita, bit do multiplicador é 0
+	<u>1101</u>	soma multiplicando, bit do multiplicador é 1
		
	1 0001 111	
		
	1000 1111	desloca para a direita (fim)
143	10001111	produto final

Embora estas operações sejam complexas, certas situações podem levar a simplificações no circuito digital que as implementa. Uma destas situações é a da **divisão por dois**, onde, no caso de números positivos sem sinal, a operação pode ser implementada apenas por um deslocamento de bits de uma posição à direita.

Exemplos:

$$6 / 2 = 3$$

0	1	1	0	6
<hr/>				desloca à direita de uma posição
0	0	1	1	3

$$5 / 2 = 2$$

0	1	0	1	5
<hr/>				desloca à direita de uma posição
0	0	1	0	2

No caso de números com sinal, na divisão, é necessária a realização de um deslocamento aritmético.

O mesmo pode ser aplicado para a multiplicação de números inteiros positivos por dois, onde esta operação pode facilmente ser implementada com um deslocamento de bits de uma posição à esquerda.

2. PARTE EXPERIMENTAL

A parte experimental desta experiência diz respeito à implementação do circuito de uma calculadora **realizada em VHDL** pela composição de componentes básicos elementares como um somador binário e um registrador deslocador. Nesta experiência sobre Dispositivos Programáveis será desenvolvido o projeto utilizando-se o dispositivo Altera Cyclone II EP2C35F672C6.

2.1. Atividades Pré-Laboratório

- a) **Especificação do Projeto:** Projetar o circuito da calculadora usando uma **descrição VHDL estrutural** que segue o diagrama de blocos da figura 2.1 abaixo. Ele consiste de um registrador deslocador A de 4 bits que armazena o valor do último resultado e um somador binário. Uma das entradas do somador vem do registrador deslocador A e a outra de chaves de entrada ENT. Todos os dados tem 4 bits.

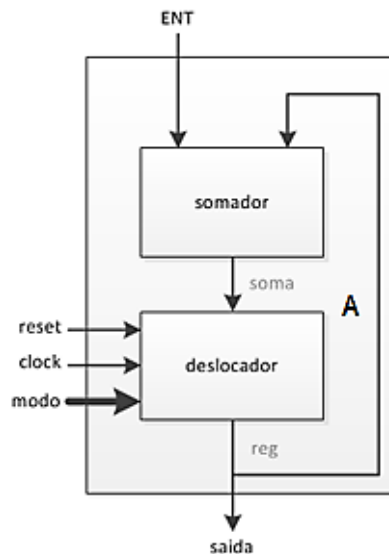


Figura 1.3 – Diagrama de blocos do circuito da calculadora.

As operações deste circuito são controladas por sinais de entrada:

- *reset*: zera o valor do registrador deslocador A (ativo em baixo);
- *clock*: sinal de clock do registrador;
- *modo*: sinal de controle do registrador.

O modo de funcionamento do registrador deslocador A segue a tabela abaixo:

Modo	Descrição
0 0	Carrega valor da entrada
0 1	Desloca 1 posição para a esquerda
1 0	Desloca 1 posição para a direita
1 1	Mantém o valor

- b) Elaborar um plano de testes, identificando sinais de depuração necessários para verificar o funcionamento do circuito projetado.
- c) Acrescentar no planejamento simulações verificando o correto projeto da calculadora simples. Use os casos de teste elaborados no item anterior. Anexe as formas de onda no planejamento.

2.2. Implementação do Projeto Exemplo

- d) Usando o software Quartus II, implemente o projeto da calculadora na placa de desenvolvimento DE2 da Altera com a seguinte designação de sinais do projeto:

- ENT[0..3] : chaves SW0 a SW3
- reset : botão KEY2
- clock : botão KEY3
- modo : chaves SW16 e SW17
- saída[0..3] : leds verdes LEDG0 a LEDG3

DICA: lembrem-se que os botões na placa DE2 são ativos em baixo. O projeto deve levar isto em consideração. Use a tabela de designação de pinos da placa DE2.

- e) Execute os testes planejados e documente os resultados obtidos no relatório.

2.3. Operação do Circuito

Agora o grupo deve desenvolver “programas” (sequência de sinais) para realizar algumas computações descritas abaixo.

- f) Considere a seguinte sequência de instruções:

Reset	-- zera registrador deslocador A
Carrega 5 em A	-- soma 5
Soma 0001 ₂	-- soma 1
Multiplica por 2	-- desloca para a esquerda

- g) Qual deve ser o resultado final em A (registrador deslocador) após a execução desta sequência?
 h) Execute os comandos no circuito projetado. Anote os resultados obtidos no relatório.
 i) Desenvolver um plano de execução para os seguintes cálculos:

$F1 = A + B + A$
$F2 = ((A * 4) + B) / 8$

- j) Elaborar uma tabela contendo todos os sinais que devem ser ativados para a execução de cada plano.
 k) Executar cada plano para pelo menos um conjunto de valores positivos.

2.4. Modificação do Projeto Base

- l) Uma pequena **modificação** ou adaptação do projeto base será solicitada aos alunos. Esta modificação deverá ser implementada no circuito projetado pelo grupo. A descrição e a documentação desta modificação devem ser incluídas no relatório.

2.5. Atividades Pós-Laboratório

- m) Após a conclusão das atividades programadas, responda as perguntas abaixo:

1. Mostre uma aplicação do circuito estudado nesta experiência.
2. O circuito da calculadora simples executa operações válidas com valores em complemento de dois? Explique.
3. Comente sobre a facilidade no uso de uma linguagem de descrição de hardware para modificar o projeto da calculadora para usar dados de entrada de 32 bits em comparação com a estratégia de projeto com captura esquemática.
4. Como é possível substituir o somador por um subtrator neste circuito?
5. Como a saída do circuito poderia ser ligada em um display de sete segmentos? Descreva sua implementação.

3. BIBLIOGRAFIA

1. ALTERA. **Quartus II Introduction Using VHDL Designs**. University Program. 2010. Disponível em: ftp://ftp.altera.com/up/pub/Altera_Material/9.1/Tutorials/VHDL/Quartus_II_Introduction.pdf
2. D'AMORE, R. **VHDL - Descrição e síntese de circuitos digitais**. 2ª edição, LTC, 2012.
3. MIDORIKAWA, E. T. **Multiplicador binário**. Apostila de Laboratório Digital, versão 2004.
4. MIDORIKAWA, E.T. **Introdução às Linguagens de Descrição de Hardware**. Apostila de PCS2304, 2007.
5. PCS-EPUSP. **Calculadora Simples**. Apostila de Laboratório Digital, Escola Politécnica da USP, 2008.
6. RANZINI, E.; HORTA, E. L.; MIDORIKAWA, E. T. **Projeto de circuitos com MAX+PLUS II**. Apostila de Laboratório Digital. Departamento de Engenharia de Computação e Sistemas Digitais, Escola Politécnica da USP. 2002.
7. RANZINI, E.; HORTA, E. L. **Introdução aos Dispositivos Lógicos Programáveis**. Apostila de Laboratório Digital. Escola Politécnica da USP, 2000 (revisão em 2011).
8. TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L. **Digital Systems: principles and applications**. 11th ed., Prentice-Hall, 2011.

4. EQUIPAMENTOS NECESSÁRIOS

- 1 placa de desenvolvimento FPGA DE2 da Altera com o dispositivo Altera Cyclone II EP2C35F672C6.
- 1 computador PC com programa Altera Quartus II e interface USB.

Histórico de Revisões

E.T.M./2012 – versão inicial.
 E.T.M./2013 – revisão.
 E.T.M./2014 – revisão.
 E.T.M./2015 – revisão.