UNIVERSIDADE FEDERAL DE MINAS GERAIS

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

INTRODUÇÃO AOS SISTEMAS LÓGICOS

Laboratório de Hardware 2

Contador de Módulos

Grupo 11
Christian Vieira
João Pedro
Marcus Oliveira

Professor: Antônio Otávio Fernandes
Monitor: Omar Vidal Pino

Belo Horizonte
31 de maio de 2016

Sumário

Lista de Figuras

Lista de implementações

Lista de Tabelas

4 Objetivo das atividades:

Projeto, implementação e verificação do funcionamento de circuitos *contadores síncrono* e *assíncrono* conectados à entrada de um decodificador *BCD/7-segmentos*, de forma a possibilitar o acompanhamento visual da contagem em um *display de LED de 7-segmentos*.

4.1 Decodificador *BCD/7-segmentos:*

Construir um circuito simples usando um *selector* de 4 linhas ligado às entradas *ABCD* de um *decodificador BCD/7-segmentos* e visualizar o valor *BCD* da entrada em um *display de LED de 7 segmentos* como se mostra na figura 1:

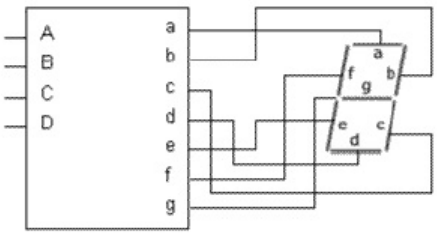

Figura 1 Display de 7 segmentos
Fonte: (FERNANDES; PINO, 2016)

O circuito exibido na figura 1 foi avaliado e simulado utilizando o *software Logisim*. Devido o *Logisim* não possuir em sua biblioteca de componentes o circuito integrado *(C.I.) decodificador BCD para 7 segmentos TTL: SN7447*, um decodificador alternativo foi construído usando uma implementação baseada em uma *ROM* a qual as entradas de endereços correspondem as entradas *BCD* e as saídas correspondem as saídas decodificadas para excitação do *display de 7 segmentos*. O circuito alternativo é exibido na figura 2.

A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	1	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	1	0	0
1	0	1	0	1	1	1	0	0	1	0
1	0	1	1	1	1	0	0	1	1	0
1	1	0	0	1	0	1	1	1	0	0
1	1	0	1	0	1	1	0	1	0	0
1	1	1	0	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1

Tabela 1 Tabela verdade para o circuito da figura 1

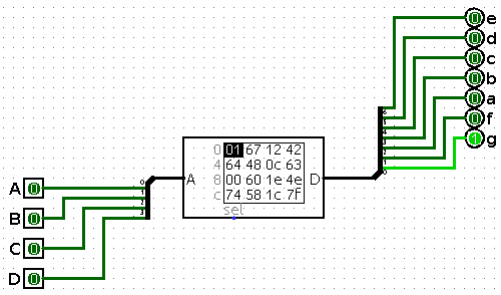

Figura 2 Circuito dec. 7seg. utilizando *ROM*

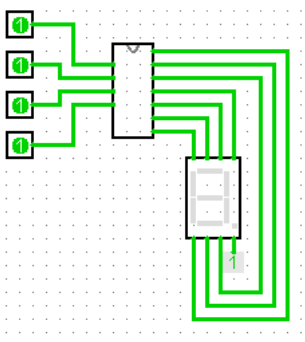

Figura 3 Circuito dec. 7seg. conectado ao *display*

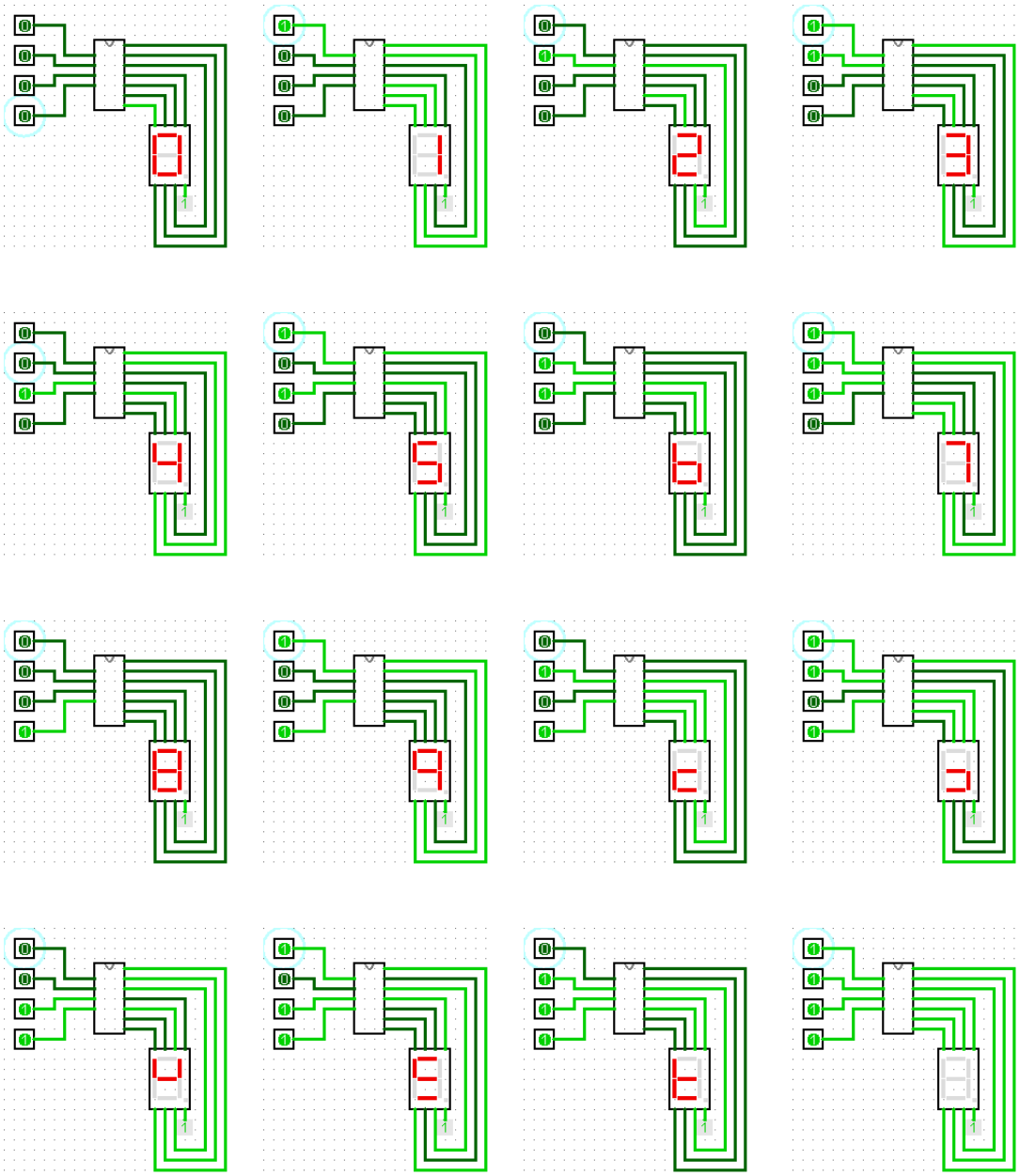

Figura 4 Apresentação da simulação do decodificador *BCD/7-segmentos*

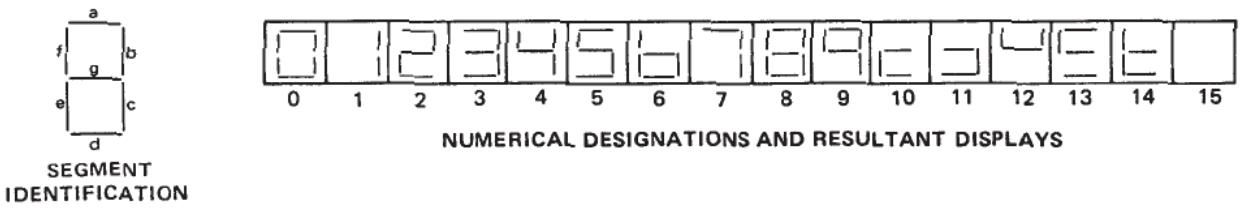

Figura 5 SN7447 Designação numérica e apresentação resultante no *display*
Fonte: (Texas Instruments, 2016)

Verificando as apresentações resultantes da decodificação utilizando o decodificador implementado em uma *ROM* (figura 4) com as apresentações exibidas pelo *C.I. SN7447* (figura 5), pode-se concluir que o *decodificador BCD/7-segmentos* implementado na *ROM* (figuras 2 e 3) possui o mesmo resultado. A implementação do *decodificador BCD/7-segmentos* possibilitou a verificação

do funcionamento do decodificador *SN7447* (maiores informações em: (Texas Instruments, 2016)) no *software Logisim*, fornecendo um dispositivo similar a ser utilizado nas subseções posteriores deste trabalho (subseções: 4.2 e 4.3).

4.2 Contador assíncrono de *3-bits:*

Construir um contador circular assíncrono de *3 bits* utilizando *flip-flops J-K*. Utilizar um *decodificador BCD/7-segmentos* para mostrar a contagem em decimal de "0" a "7" no *display* e possibilitar avaliar o perfeito funcionamento do circuito. Apresentar a forma de onda do circuito.

Um *contador assíncrono* é um tipo de contador em que a entrada de *clock* dos elementos de memória *(flip-flops)* não estão conectados em comum. A figura6 exibe o *grafo dirigido* com o esquema de contagem, onde as setas nas *arestas* representam uma transição de clock e os *nodos* representam um estado alcançado do contador. A figura 7 exibe o contador de *3 bits* conectado ao decodificador *BCD/7-segmentos*. Como não foi especificado nenhum início específico para a contagem, o valor de contagem pode iniciar entre "0" à "7".

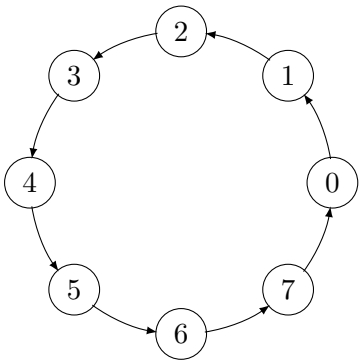

Figura 6 Grafo c/ esquema da contagem

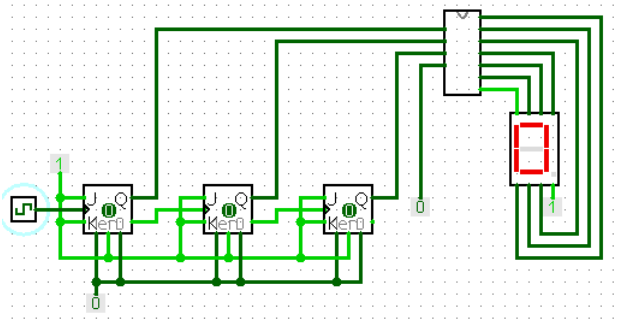

Figura 7 Contador assíncrono de *3 bits* conectado ao decodificador *BCD/7-segmentos*

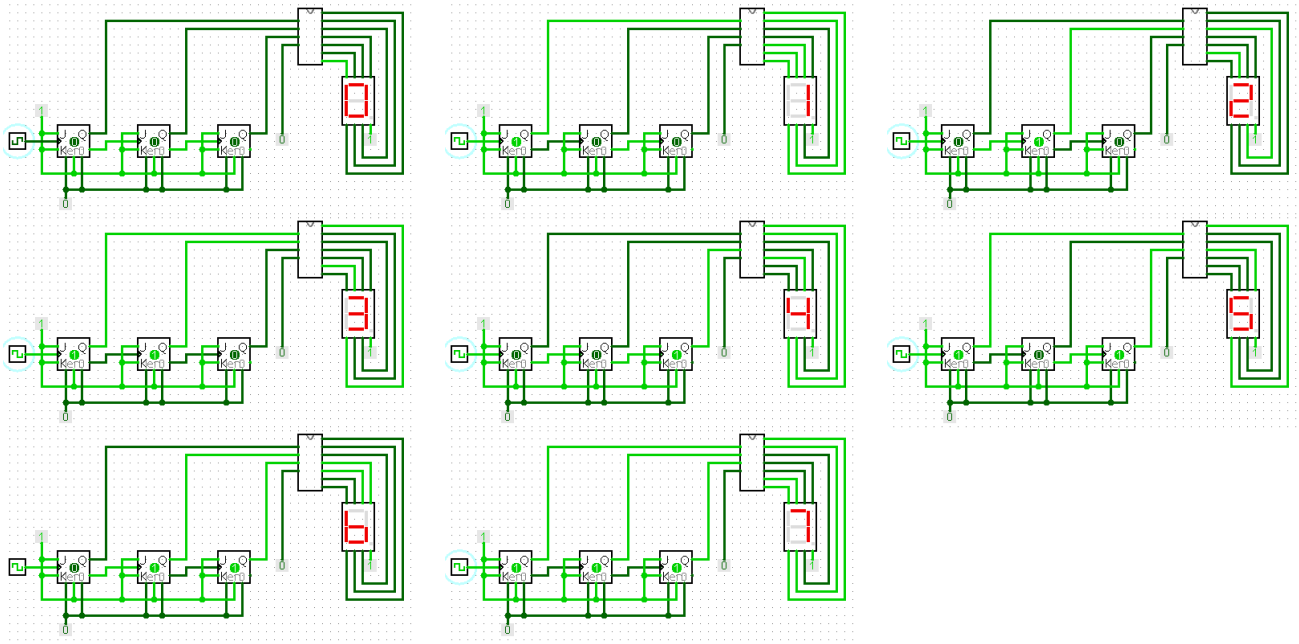

Figura 8 Apresentação da simulação do circuito contador assíncrono de *3 bits*

A simulação do circuito contador assíncrono é exibida na figura 8, onde pode-se observar a seqüência de contagem de "0" à "7", comprovando dessa maneira o resultado esperado de funcionamento.

As formas de ondas do circuito foram obtidas a partir da implementação do circuito em uma linguagem de descrição de *hardware*, uma vez que o laboratório da disciplina de *Introdução aos Sistemas Lógicos* não possui *osciloscópio* ou *analisador lógico* para obtenção e gravação das formas de onda. Assim, foram usados os *softwares GHDL* e *gtkwave*(BYBELL, 2016) para descrição e visualização das formas de onda (na figura 9; *clk* é o sinal de *clock*, *pren* e *clrn* são respectivamente os sinais de *preset* e *clear* assíncronos, *ff_q0, ff_q1* e *ff_q2* são as saídas *q* de cada *flip-flop J-K*, *disp_7seg_bcd* e *disp_7seg* são respectivamente a entrada e a saída do decodificador *BCD/7-segmentos*). A opção pela linguagem *VHDL* deu-se devido a curiosidade de descrever circuitos digitais utilizando uma linguagem de descrição fortemente tipada e com características das linguagens procedurais *ADA* e *Pascal*(GINGOLD, 2016). As descrições do decodificador *BCD/7-segmentos* e *flip-flops* podem ser visualizadas nas subseções do apêndice A: A.1 e A.2. As descrições do contador assíncrono e do *testbench* são dadas nas subseções do apêndice B: B.1 e B.2.

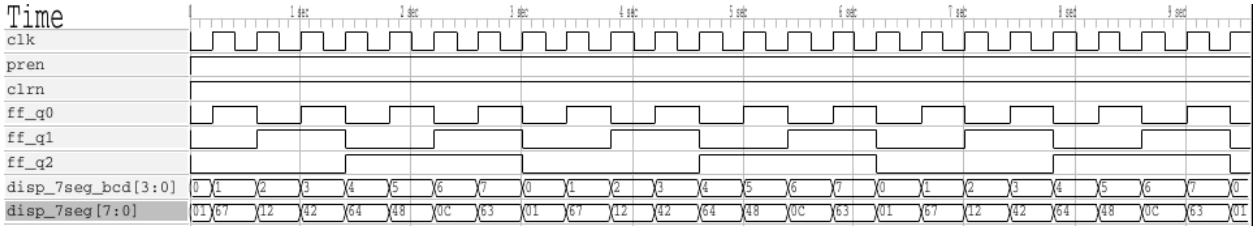

Figura 9 Forma de onda do contador assíncrono de *3 bits*

4.2.1 *Link* para visualização do contador assíncrono em funcionamento:

O funcionamento do contador assíncrono de *3 bits* pode ser visualizado em: <https://www.youtube.com/watch?v=BuaAKZl6dJA>. A figura 10 exibe a montagem final do circuito.

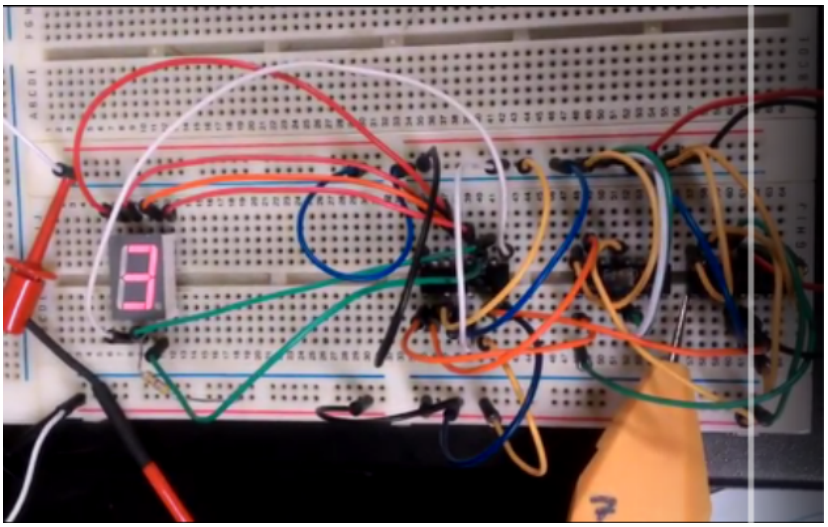

Figura 10 Exibição da montagem final contador assíncrono de *3 bits*

4.3 Contador circular síncrono módulo 6:

Modificar o projeto anterior de modo a implementar um *contador circular síncrono de módulo "6"* (com *flip-flops JK*). Utilizar um *decodificador BCD/7-segmentos* para mostrar a contagem em decimal no *display* e possibilitar avaliar o perfeito funcionamento do circuito. Apresentar e discutir a estratégia utilizada para reiniciar a contagem. Apresentar a forma de onda do circuito.

Um contador síncrono é um tipo de contador em que a excitação do pulso de *clock* é comum a todos os elementos de memória *(flip-flops)*. Especificamente no caso do contador de módulo 6, foi adicionado um circuito combinacional de forma a possibilitar o reinício da contagem sempre após o quinto estado estável do contador, ou seja, o contador sempre efetua a contagem de "0" à "5" ciclicamente. O grafo de comportamento do circuito é exibido na figura 11, uma das possíveis implementações pode ser visualizada na figura 12. Assim como no circuito assíncrono descrino na subseção 4.2, não foi especificado um início de contagem, portanto ao ser energizado o circuito poderá iniciar a contagem a partir de um dos valores de "0" à "5".

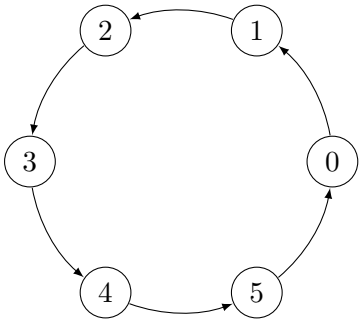

Figura 11 Grafo c/ esquema da contagem

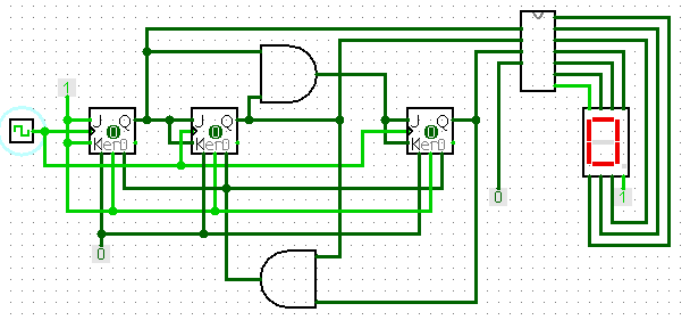

Figura 12 Contador síncrono de *3 bits* módulo 6

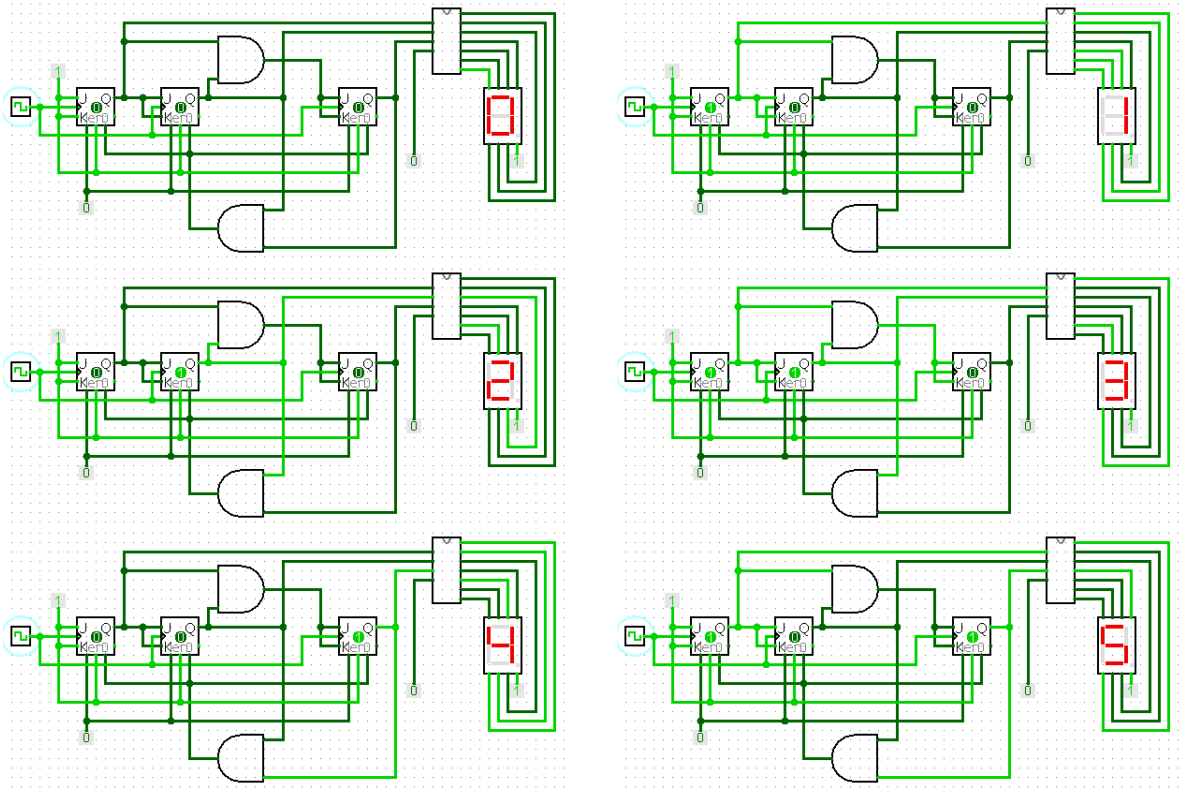

Figura 13 Apresentação da simulação do circuito contador sincrono módulo 6

A simulação do circuito contador síncrono é exibida na figura 13, onde pode-se observar a seqüência de contagem de “0” à “5”, comprovando dessa maneira o resultado esperado de funcionamento.

Assim como no caso do circuito assíncrono, as formas de ondas do circuito síncrono foram obtidas a partir da implementação do circuito em uma linguagem de descrição de *hardware* (de forma similar ao explanado na subseção 4.2). As descrições do decodificador *BCD/7-segmentos* e *flip-flops* podem ser visualizadas nas subseções do apêndice A: A.1 e A.2. As descrições do contador assíncrono e do *testbench* são dadas nas subseções do apêndice C: C.1 e C.2.

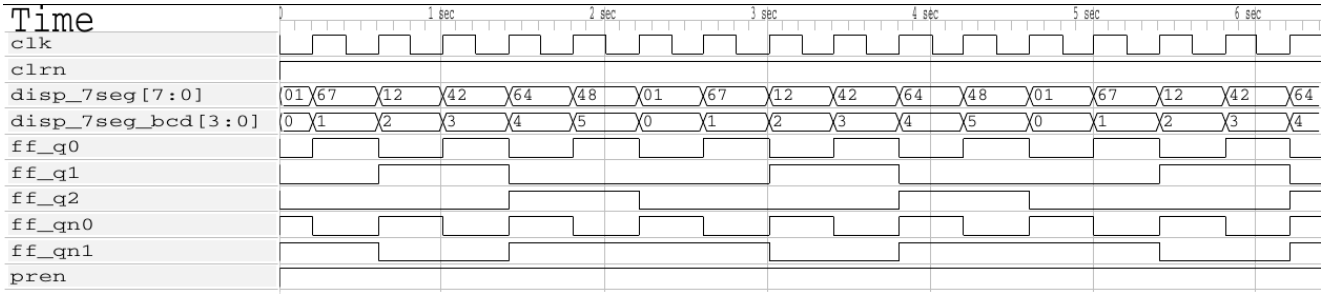

Figura 14 Forma de onda do contador síncrono de *3 bits* módulo 6

4.3.1 *Link* para visualização do contador síncrono em funcionamento:

O funcionamento do contador síncrono de *3 bits* módulo 6 pode ser visualizado em: <https://www.youtube.com/watch?v=RFVExji6z6c>. A figura 15 exibe a montagem final do circuito.

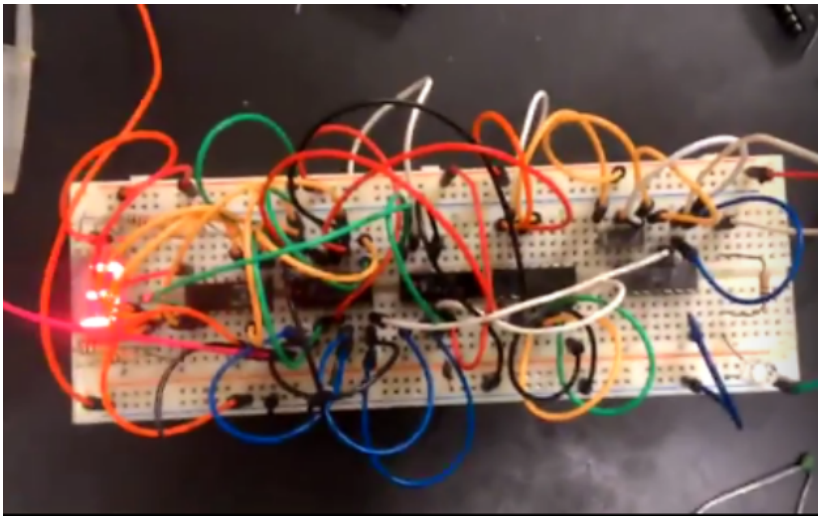

Figura 15 Exibição da montagem final contador síncrono de *3 bits* módulo 6

4.4 Conclusões

As execuções das atividades previstas proporcionaram contato direto com a tecnologia discreta de montagem de circuitos digitais seqüenciais assíncronos e síncronos em um ambiente de prototipação. O uso de uma ferramenta de simulação de circuitos digitais seqüenciais possibilitou a verificaçao do funcionamento dos circuitos projetados antes de serem montados, certificando dessa forma os resultados esperados da montagem prática.

Durante a execução das montagens práticas dos circuitos, dispendeu-se muito tempo na verificação das conexões elétricas nas matrizes de contatos utilizada *(protoboards)*. Por serem matrizes com muito tempo de uso, alguns pontos de conexão apresentavam desgastes (certamente devido à oxidação e/ou número de inserção/remoção de componentes), não sendo realizado o contato elétrico necessário para o funcionamento esperado do circuito, levando a erros intermitentes de decodificação e contagem. A seleção dos componentes utilizados para as montagens dos circuitos seqüenciais foi crítica uma vez que o laboratório não possui um testador de circuitos integrados digitais. Assim, houve casos em que o circuito não funcionou devido parte interna de um circuito integrado contendo dois *flip-flops J-K* internos apresentar um ou os dois blocos de *flip-flops* defeituosos. Outra dificuldade foi encontrar os componentes no laboratório, tais como o *display de 7-segmentos* do tipo anodo comum (ledtech, 2016).

Os circuitos avaliados no *software Logisim* apesar de terem o funcionamento idêntico ao esperado, não correspondem em sua totalidade aos circuitos experimentais montados no laboratório, uma vez que a biblioteca de componentes do *Logisim* não dispõem dos mesmos componentes da montagem tais como o decodificador *BCD/7-segmentos* e do *flip-flop JK*. Afim de contornar o problema e dispor de um circuito simulado que pudesse oferecer um comportamento tal qual esperado, no caso do decodificador *BCD/7-segmentos*, foi realizado um circuito com funcionamento idêntico ao *C.I. SN7447*(subseção 4.1) utilizando um decodificador *"mapeado"* em *ROM*. No caso do *flip-flop JK*, foi utilizado o que a biblioteca possui, com a ressalva de que os sinais de *PRESET* e *CLEAR* são ativos em alto (nível lógico alto - "H") e não em baixo (nível lógico baixo - "L"). A incompatibilidade no caso do *flip-flop JK* foi contornada na montagem experimental através da alteração da lógica adicional para *RESET* dos *flip-flop's* sempre que o limite da contagem fosse atingido (caso do contador síncrono de *3 bits* módulo 6). Ademais, um circuito *RC* foi montado de tal forma que o *spike* gerado pelo acoplamento capacitivo fosse capaz de gerar um pequeno atraso para o correto *RESET* dos *flip-flop's* (contador síncrono de *3 bits* módulo 6 – subseção 4.3). As descrições em *VHDL* (apêndices A, A.2, B e C) refletem com maior detalhe o comportamento do circuito, uma vez que a composição dos elementos lógicos utilizados refletem em funcionamento aos componentes físicos utilizados na montagem experimental.

Devido ao laboratório não possuir um *osciloscópio/analisador lógico digital* que possibilitasse o registro das informações elétricas em função do tempo, as formas de ondas exibidas neste relatório foram obtidas a partir do *testbench* dos circuitos seqüenciais descritos na linguagem *VHDL* utilizando ferramentas de *software livre* tais quais *GHDL* (ferramenta para simulação e síntese usando a linguagem *VHDL*) (GINGOLD, 2016) e *gtkwave* (ferramenta para visualização das formas de ondas, uma espécie de analisador lógico digital) (BYBELL, 2016).

5 Referências Bibliográficas

BYBELL, T. *gtkwave*. 2016. [Online; acessado em 23/05/2016]. Disponível em: <https://sourceforge.net/projects/gtkwave/>.

FERNANDES, A. O.; PINO, O. V. *Laboratório de Hardware 2 - Contador de Módulos*. 2016. [Online; acessado em 23/05/2016]. Disponível em: <https://virtual.ufmg.br/20161/pluginfile.php/244608/mod_folder/content/0/Lab_2/lab2.pdf?forcedownload=1>.

GINGOLD, T. *GHDL*. 2016. [Online; acessado em 23/05/2016]. Disponível em: <https://sourceforge.net/projects/ghdl-updates/>.

ledtech. *LA(C)5621-11 BWRS 0.56"(14.22mm)SINGLE DIGIT DISPLAY*. 2016. [Online; acessado em 23/05/2016]. Disponível em: <http://www.ledtech.com.tw/newweb/newweb/SPEC/LA5621-11.pdf>.

Texas Instruments. *SN7447 BCD-To-Seven-Segment Decoders/Drivers*. 2016. [Online; acessado em 23/05/2016]. Disponível em: <http://www.ti.com/lit/ds/symlink/sn7447a.pdf>.

Apêndice A Descrição comum às implementações:

A.1 Decodificador *BCD/7-segmentos*

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4
5  entity dec_bcd_to_7seg is
6    port (
7      bcd:         in  std_logic_vector(3 downto 0);
8      seven_segs: out std_logic_vector(7 downto 0)
9    );
10 end entity dec_bcd_to_7seg;
11
12 architecture rtl of dec_bcd_to_7seg is
13   begin
14   bcd_to_7seg : process(bcd) begin
15     case bcd is
16       when "0000" => seven_segs <= x"01";
17       when "0001" => seven_segs <= x"67";
18       when "0010" => seven_segs <= x"12";
19       when "0011" => seven_segs <= x"42";
20
21       when "0100" => seven_segs <= x"64";
22       when "0101" => seven_segs <= x"48";
23       when "0110" => seven_segs <= x"0c";
24       when "0111" => seven_segs <= x"63";
25
26       when "1000" => seven_segs <= x"00";
27       when "1001" => seven_segs <= x"60";
28       when "1010" => seven_segs <= x"1e";
29       when "1011" => seven_segs <= x"4e";
30
31       when "1100" => seven_segs <= x"74";
32       when "1101" => seven_segs <= x"58";
33       when "1110" => seven_segs <= x"1c";
34       when others => seven_segs <= x"7f";
35     end case;
36   end process;
37 end architecture rtl;
```

Listagem 1 Decodificador *BCD/7-segmentos*

A.2 Flip Flop J-K:

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity flipflop_jk is
5    port (
6      clk, pren, clrn, j, k: in  std_logic;
7      q, qn: out std_logic
8    );
9  end entity flipflop_jk;
10
11 architecture rtl of flipflop_jk is
12   signal tmp_q, tmp_qn : std_logic;
13   signal jk : std_logic_vector(1 downto 0) := "00";
14   begin
15   jk <= j & k;
16   ff_jk : process(clk, pren, clrn)
17   begin
18     if(pren = '0' and clrn = '1') then
19       tmp_q  <= '1';
20       tmp_qn <= '0';
21     elsif(pren = '1' and clrn = '0') then
22       tmp_q  <= '0';
23       tmp_qn <= '1';
24     elsif(pren = '0' and clrn = '0') then
25       tmp_q  <= '1';
26       tmp_qn <= '1';
27     elsif(pren = '1' and clrn = '1') then
28       if(rising_edge(clk)) then
29         case jk is
30           when "00" =>
31             tmp_q  <= tmp_q;
32             tmp_qn <= tmp_qn;
33           when "01" =>
34             tmp_q  <= '0';
35             tmp_qn <= '1';
36           when "10" =>
37             tmp_q  <= '1';
38             tmp_qn <= '0';
39           when "11" =>
40             tmp_q  <= not tmp_q;
41             tmp_qn <= not tmp_qn;
42           when others =>
43             tmp_q  <= tmp_q;
44             tmp_qn <= tmp_qn;
45         end case;
46       end if;
47     end if;
48   end process;
49   q  <= tmp_q;
50   qn <= tmp_qn;
51 end architecture rtl;
```

Listagem 2 Flip-Flop JK

Apêndice B Contador assíncrono:

B.1 Contador assíncrono de 3 *bits*:

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity async_counter_top is
5    port (
6      clk, pren, clrn: in  std_logic
7    );
8  end entity async_counter_top;
9
10 architecture behavioral of async_counter_top is
11   component dec_bcd_to_7seg is
12     port (
13       bcd:         in  std_logic_vector(3 downto 0);
14       seven_segs: out std_logic_vector(7 downto 0)
15     );
16   end component;
17
18   component flipflop_jk is
19     port (
20       clk, pren, clrn, j, k: in  std_logic;
21       q, qn: out std_logic
22     );
23   end component;
24
25   signal ff_q0, ff_qn0, ff_q1, ff_qn1, ff_q2 : std_logic;
26   signal tmp:std_logic;
27   signal disp_7seg : std_logic_vector(7 downto 0);
28   signal disp_7seg_bcd : std_logic_vector(3 downto 0);
29
30   begin
31   ff_jk0 : flipflop_jk port map(clk, pren, clrn, '1', '1', ff_q0, ff_qn0);
32   ff_jk1 : flipflop_jk port map(ff_qn0, pren, clrn, '1', '1', ff_q1, ff_qn1);
33   ff_jk2 : flipflop_jk port map(ff_qn1, pren, clrn, '1', '1', ff_q2, tmp);
34   disp_7seg_bcd <= '0' & ff_q2 & ff_q1 & ff_q0;
35   bcd_7seg_dec0 : dec_bcd_to_7seg port map(disp_7seg_bcd, disp_7seg);
36 end architecture behavioral;
```

Listagem 3 Contador assíncrono de *3 bits*

B.2 Testbench do contador assíncrono de *3 bits*:

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity async_counter_tb is
5  end entity async_counter_tb;
6
7  architecture async_counter_behavioral of async_counter_tb is
8    constant  initial_time : time := 10 ms;
9    constant  clk_cycle : time  := 200 ms;
10
11   signal mclk, mpren, mclrn : std_logic;
12
13   component async_counter_top is
14     port (
15      clk, pren, clrn: in  std_logic
16     );
17   end component;
18
19   begin
20
21   initial_state: process
22   begin
23     mpren <= '1';
24     mclrn <= '0';
25     wait for initial_time/2;
26     mpren <= '1';
27     mclrn <= '1';
28     wait;
29   end process;
30
31   clk_gen: process
32   begin
33     mclk <= '1';
34     wait for initial_time;
35     loop
36       mclk <= not mclk;
37       wait for clk_cycle;
38     end loop;
39   end process;
40
41   async_counter_dut: async_counter_top port map(mclk, mpren, mclrn);
42
43 end architecture async_counter_behavioral;
```

Listagem 4 *Testbench* do contador assíncrono de *3 bits*

Apêndice C Contador síncrono:

C.1 Contador síncrono de *3 bits* módulo 6:

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity sync_counter_top is
5    port (
6      clk, pren, clrn: in  std_logic
7    );
8  end entity sync_counter_top;
9
10 architecture behavioral of sync_counter_top is
11   component dec_bcd_to_7seg is
12     port (
13       bcd:         in  std_logic_vector(3 downto 0);
14       seven_segs: out std_logic_vector(7 downto 0)
15     );
16   end component;
17
18   component flipflop_jk is
19     port (
20       clk, pren, clrn, j, k: in  std_logic;
21       q, qn: out std_logic
22     );
23   end component;
24
25   signal ff_q0, ff_qn0, ff_q1, ff_qn1, ff_q2 : std_logic;
26   signal clearff, and1 : std_logic;
27   signal disp_7seg : std_logic_vector(7 downto 0);
28   signal disp_7seg_bcd : std_logic_vector(3 downto 0);
29
30   begin
31   and1 <= ff_q0 and ff_q1;
32   clearff <= clrn and not(ff_q1 and ff_q2);
33   disp_7seg_bcd <= '0' & ff_q2 & ff_q1 & ff_q0;
34   ff_jk0 : flipflop_jk port map(clk, pren, clearff, '1', '1', ff_q0, ff_qn0);
35   ff_jk1 : flipflop_jk port map(clk, pren, clearff, ff_q0, ff_q0, ff_q1,
       ff_qn1);
36   ff_jk2 : flipflop_jk port map(clk, pren, clearff, and1, and1, ff_q2);
37   bcd_7seg_dec0 : dec_bcd_to_7seg port map(disp_7seg_bcd, disp_7seg);
38 end architecture behavioral;
```

Listagem 5 Contador síncrono de *3 bits* módulo 6

C.2 Testbench do contador síncrono de *3 bits* módulo 6:

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity sync_counter_tb is
5  end entity sync_counter_tb;
6
7  architecture sync_counter_behavioral of sync_counter_tb is
8    constant  initial_time : time := 10 ms;
9    constant  clk_cycle : time  := 200 ms;
10
11   signal mclk, mpren, mclrn : std_logic;
12
13   component sync_counter_top is
14     port (
15      clk, pren, clrn: in  std_logic
16     );
17   end component;
18
19   begin
20
21   initial_state: process
22   begin
23     mpren <= '1';
24     mclrn <= '0';
25     wait for initial_time/2;
26     mpren <= '1';
27     mclrn <= '1';
28     wait;
29   end process;
30
31   clk_gen: process
32   begin
33     mclk <= '1';
34     wait for initial_time;
35     loop
36       mclk <= not mclk;
37       wait for clk_cycle;
38     end loop;
39   end process;
40
41   sync_counter_dut: sync_counter_top port map(mclk, mpren, mclrn);
42
43 end architecture sync_counter_behavioral;
```

Listagem 6 *Testbench* do contador síncrono de *3 bits* módulo 6