



UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE – CAMPUS NATAL

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ELE 0515 – CIRCUITOS LÓGICOS

MEMORIAL DESCRITIVO

PROJETO DE UM PARQUÍMETRO

JOSE RICARDO BEZERRA

LEVY GABRIEL DA SILVA GALVÃO

THIAGO MAIA SOUTO

NATAL – RN

JUNHO/2019

SUMÁRIO

MEMORIAL DESCRITIVO	3
1.1 Demanda	3
MÁQUINA DE ESTADOS DO PARQUÍMETRO	4
IMPLEMENTAÇÃO EM VHDL	7
3.1 Codificação	7
3.2 Entidade	7
SIMULAÇÃO	9
CONCLUSÃO	12
REFERÊNCIAS BIBLIOGRÁFICAS	13
CÓDIGO FONTE	14

1 MEMORIAL DESCRITIVO

O presente projeto de circuitos lógicos tem o objetivo de aprofundar os conhecimentos acerca do uso da linguagem de descrição de hardware VHDL no projeto de um parquímetro como requisito para obtenção parcial da nota da terceira unidade da disciplina de ELE0515 – Circuitos lógicos, ministrada pelo Prof. Dr. José Alfredo Ferreira Costa.

1.1 Demanda

O projeto visa construir um parquímetro. Ele deve ser projetado de tal maneira que permita que o usuário inicie uma sessão, insira as moedas, confirme o início da contagem e logo após acabe a sessão.

Cada sessão deve ser armazenada em um total diário, para que ao final do dia seja conhecido o número de sessões que ocorreram naquele parquímetro.

O parquímetro associa cada $\text{R\$}0,50$ a 15 minutos de estacionamento. Se $\text{R\$}0,25$ são inseridos é aguardado que múltiplos de $\text{R\$}0,50$ sejam inseridos para somar mais 15 minutos. O tempo máximo é 2 horas. Ou seja, no máximo $\text{R\$}4,00$ devem ser inseridos para que o tempo seja contado. Caso seja inserida uma quantia de dinheiro maior que esse valor, o excedente será destinado à caridade, sendo que apenas no máximo $\text{R\$}4,00$ irá para o governo.

Variáveis também armazenam o total diário que irá para o governo e caridade para que o operador saiba qual a quantia física que será destinada para cada um

A máquina de alto nível contou com 9 estados. Vale destacar que o botão reset (rst) pode ser acionado em qualquer estado e voltará para o estado de inicialização. O estado I representa a inicialização. Ele só será alcançado por meio do reset e ao iniciar o parquímetro. Ele será responsável por zerar as variáveis TGO, TCH e TD, que representam, respectivamente o total diário enviado para o governo, o total diário enviado para a caridade e o total de sessões por dia (TGO = TOTAL_GOVERNMENT, TCH = TOTAL CHARITY, TD = TOTAL DAY).

O estado W representa o estado de espera para uma nova sessão. Só sairá desse estado quando o botão de iniciar uma nova sessão for pressionado, ou seja, o botão b. Esse estado também irá zerar as variáveis que armazenam valores durante a sessão, que são: TS, TT, GO e CH, que representam, respectivamente, o valor de dinheiro armazenado naquela sessão, o valor de tempo que aquela quantia de dinheiro representa, o total de dinheiro para o governo e caridade, naquela sessão (TS = TOTAL_SECTION, TT = TOTAL_TIME, GO = GOVERNMENT e CH = CHARITY).

O estado S espera por uma moeda ser inserida ou por confirmar a inserção de moedas. O botão c representa se existe moeda ou não e o botão f representa se finalizou a inserção de moedas. Se ambos não forem pressionados, ele permanecerá naquele estado. Se c for pressionado e o f não, ele se encaminha para o estado de computar o TS e TT. Se o f for pressionado a máquina se encaminha para um estado que irá deduzir o tempo.

O estado H é acessado quando há presença de moedas, ele vai enxergar o valor que a moeda representa por meio da variável A para o valor da moeda e vai somar ao TS. Se o valor de TS for abaixo de R\$4,00 ele se encaminha para o estado T que irá computar o valor de tempo que representa a moeda (lembrando $\text{R\$}50 = 15$ minutos) e somar ao TT. Depois do estado T, ele volta para o estado S. No estado H, quando o TS é maior que R\$4,00, ele não computa o tempo e volta para o estado S.

Quando no estado C, a cada ciclo de clock uma unidade de tempo será removida do tempo total equivalente às moedas inserida (no máximo 2h da variável TT). Se o tempo for diferente de zero e um botão d (fim de sessão) não for pressionado, ele permanecerá naquele estado contando o tempo. Só sairá desse estado quando o tempo TT for zero ou o botão d for pressionado.

Ao sair do estado C, a máquina se encaminha para dois estados diferentes. Irá para o estado CH quando o TS for maior que R\$4,00, assim, destinando R\$4,00 para o governo e o restante para a caridade e irá para o estado GO quando a quantia TS for menor que R\$4,00, destinando todo o dinheiro para o governo e nada para a caridade.

Saindo desses estados GO e CH, a máquina se encaminha automaticamente para um estado E que irá adicionar a TGO e TCH, os valores de GO e CH, respectivamente daquela sessão e somará mais uma sessão ao TD.

3 IMPLEMENTAÇÃO EM VHDL

3.1 Codificação

Antes de criar a entidade em si do código, foi definida algumas codificações binárias para os valores das moedas e para os tempo representados.

Foram usados três bits para representar as três moedas: 001 para ¢25, 010 para ¢50 e 100 para R\$1,00, ou seja, ¢25 representando o 1, e as outras moedas representando o dobro disso.

O tempo total foi representado por quatro bits, sendo que a cada ¢50 no TS seria adicionado uma unidade de tempo ao TT. Ou seja, cada unidade de tempo equivale a 15 minutos (0001 = 15min, 1000 = 2h). No máximo terá 8 unidade de tempo, ou seja 120 minutos = 2 horas. Três bits não seriam bastante, pois o zero teria que representar um valor, e perderia a representação para zero unidade de tempo, em si, ou outra qualquer.

3.2 Entidade

A entidade constitui-se das seguintes variáveis abaixo:

```
1. entity parkOmeter is  
2.     port(  
3.         b, c, f, rst, d: in std_logic;  
4.         A: in std_logic_vector (2 downto 0);  
5.         clk: in std_logic;  
6.         ts, go, ch, td : buffer std_logic_vector (5 downto 0);  
7.         tgo, tch : buffer std_logic_vector (7 downto 0);  
8.         tt : buffer std_logic_vector (3 downto 0)  
9.     );  
10.  
11. end parkOmeter;
```

Existem 6 entradas de um bit. O clk é uma delas e ele representa o clock da simulação. As entradas b, c, f, rst e d representam o seu equivalente na máquina de estados, respectivamente, botão de iniciar a sessão, botão de inserir moeda, botao de finalizar a inserção das moedas, botão de resetar e o botão de finalizar a contagem.

As saídas são todas tratadas como buffer, pois elas vão ser lidas ao longo da arquitetura. As saídas ts, go, ch, td, tgo, tch e tt, possuem a mesma equivalência de significados para aquelas representadas na máquina de estados. As saídas ts, go, ch e td possuem 6 bits apenas por uma conveniência de não serem tão grandes para uma sessão, pois não existirá tantas seções por dia e também cada sessão não haverá uma alta quantia de dinheiro (se quiser destinar dinheiro à

caridade, intencionalmente, a melhor escolha não será um parquímetro). Já as saídas tgo e tch possuem 8 bits, pois permitem uma quantidade de dinheiro maior, já que equivalem aos valores de dinheiro computados durante o dia inteiro.

O que tange a arquitetura, fora usado um processo que é sensível à borda de subida do clock e ao botão rst. Na ocorrência de algum deles, a máquina de estado é dada continuidade.

Outro detalhe significativo foi a criação do tipo estado (state), como pode ser visto abaixo:

```
1.      type state is (wait_b, sel_type, add, time_count, end_park);
```

Esse tipo estado codifica os estados da máquina de estados, porém, de uma forma bem mais reduzida. O estado de inicializar ficou destinado apenas ao reset geral. O estado W ficou como wait_b; O estado S como sel_type; Os estados H e T se juntaram em um só, o estado add; O estado C é o estado time_count; e os estados CH, GO e E estão juntos no end_park. Isso tudo é possível devido à estrutura mais alto nível que o VHDL possui.

4 SIMULAÇÃO

Após compilar o código corretamente, foram feitas algumas simulações para tentar mostrar um pouco do funcionamento do parquímetro.

A primeira simulação vai mostrar um cliente que inseriu diversas moedas. A quantia de dinheiro ultrapassou R\$4,00 em ts que em binário equivale a 10000=16, mas a quantidade de tempo em tt permaneceu 2h, ou em binário 1000=8.

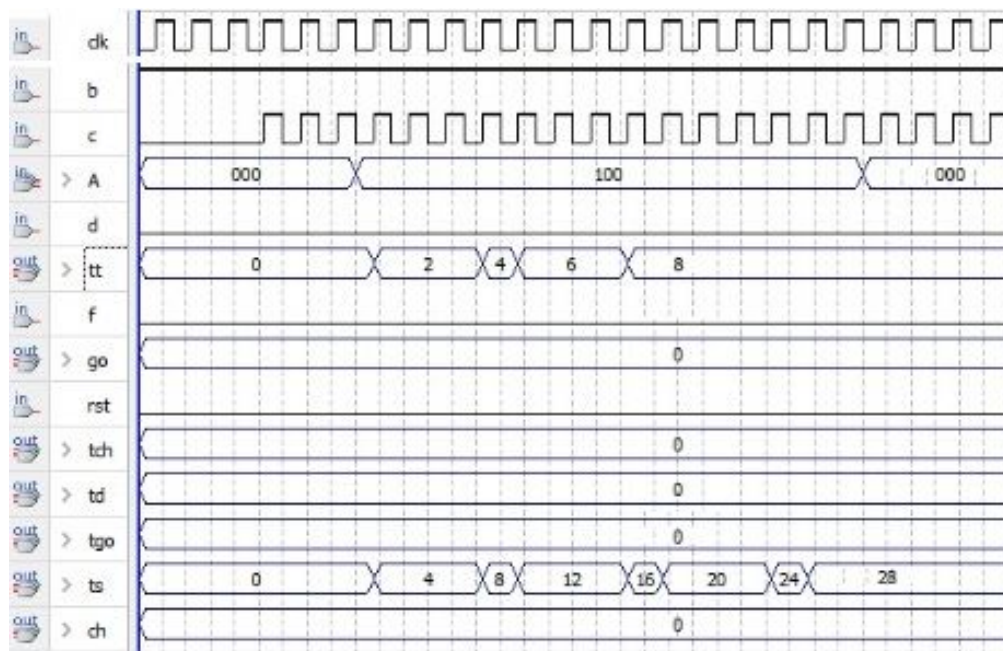


Figura 2 – Inserindo várias moedas de R\$1,00.

No caso desta simulação, o c assumiu o valor um em vários pulsos, permitindo que a moeda A de valor 100 = R\$1,00 fosse inserida algumas vezes. Como o botão f não foi acionado, o parquímetro não começou a decrescer o tempo. Junto a isso, os valores de GO, CH, TCH, TGO e TD não foram atualizados, porque o ciclo não fechou. Porém, o valor de TS e TT foram atualizados. TS foi adicionado de 4 em 4 unidades, que equivalem a múltiplos de ¢25, ou sejam, adicionando R\$1,00 de cada vez. No final foram inseridas 7 moedas de R\$1,00, contabilizando R\$7,00. Porém, observa-se que o tempo TT, que inicialmente foi adicionado de 2 em 2 unidades, cuja cada unidade equivale a 15min. Então foram adicionando de 30 em 30min. No final, o valor de TT estagnou em 2h que equivale a 8 unidades de TT ($8 \times 15\text{min} = 120\text{min} = 2\text{h}$). Assim, mostrou que não importa o quanto de dinheiro insira, o máximo de tempo será 2h.

A segunda simulação, como pode ser vista na figura abaixo, vai tratar de inserir duas moedas de ¢50, finalizar a inserção de moedas com o botão f, decrescer o tempo e logo após

computar os devidos valores temporários e totais para o governo e caridade. Nesse caso, a intenção desta segunda simulação é inserir uma quantia que não extrapole o tempo de 2h, implicando que nenhum dinheiro irá para a caridade.

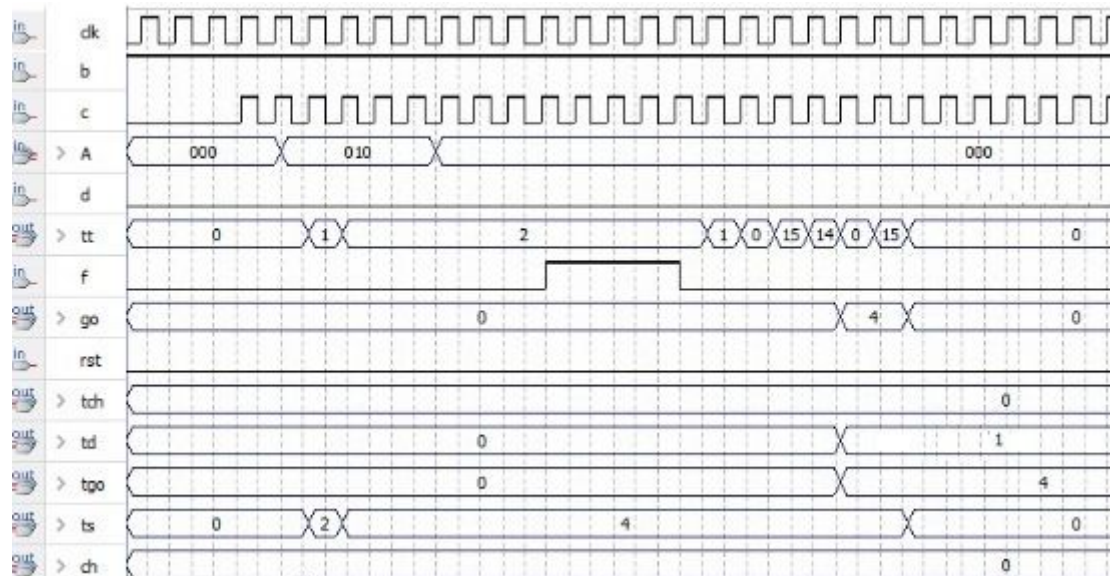


Figura 3 – Inserindo moedas de forma que não extrapole R\$4,00.

Inserire-se duas moedas de ¢50 com a codificação de 010 em A. No final da contagem, TT assume valor 2, ou seja 0010=30min. TS assume o valor de 4, que equivale a R\$1,00 no total. Logo após o botão f ser pressionado, o tempo TT decresce junto aos pulsos de clock. Após o tempo acabar, ele acaba sendo deduzido novamente, passando para 15, 15, ..., etc. Esse bug não tem efeito no funcionamento da máquina.

No final são computadas as 4 unidades de TS diretamente para o governo durante a sessão em GO e depois esse valor se adiciona a TGO, computando o total diário. Nada vai para a caridade, por isso TCH e CH são nulos.

A última simulação mostra a extrapolação dos R\$4,00.

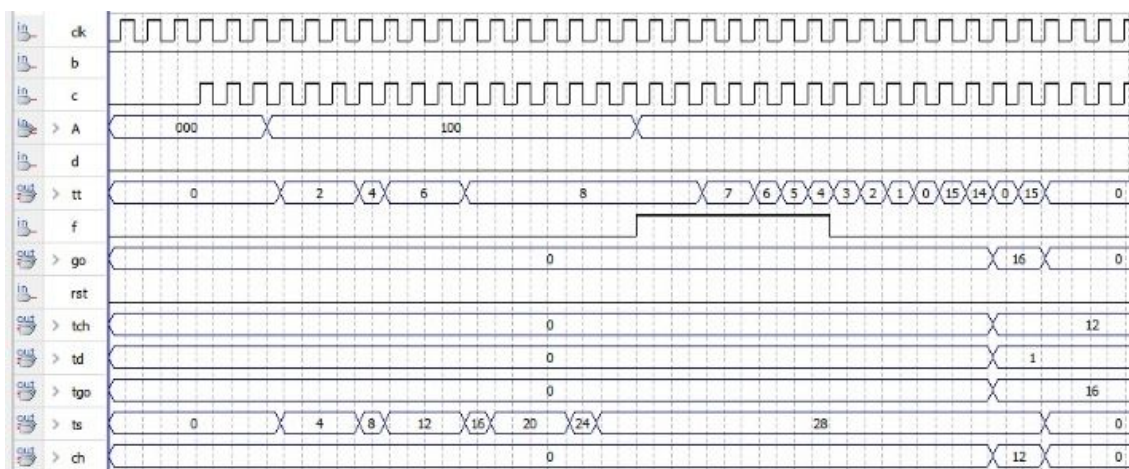


Figura 4 – Inserindo moedas que extrapole R\$4,00

Da mesma forma que a primeira simulação, foi-se inserindo moedas de R\$1,00. No final, TS ficou com 28 ($28 \times \text{R}\$0,25 = \text{R}\$7,00$), mas TT permaneceu com 8 unidades (2 horas). Ao botão f ser pressionado, TT foi sendo subtraído de 1 a cada ciclo de clock.

Ao final da sessão foram destinados 16 para GO e o restante de 12 para CH, ou seja, R\$4,00 temporários para o governo e R\$3,00 para a caridade. Ao final, TCH e TGO foram atualizados, ficando eles com os valores anteriores mais os valores de GO e CH da sessão. VAle destacar que, assim como na segunda simulação, uma sessão foi contabilizada, fazendo com que TD somasse mais 1.

5 CONCLUSÃO

É preciso desconfiar sete vezes do cálculo e cem vezes do matemático

Provérbio Indiano

O desenvolvimento do projeto foi de fundamental importância para aprofundar o conhecimento acerca do uso da linguagem VHDL para a descrição de hardware. Assim, os conhecimentos adquiridos contribuem para a execução de outros projetos, e a experiência só adiciona à complexidade, principalmente se necessitarem de componentes semelhantes, permitindo agilizar o projeto, pois o projetista já dispõe dos componentes.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ABNT, Associação Brasileira de Normas Técnicas. **NBR 10719 – Apresentação de relatórios técnico-científicos**. Rio de Janeiro: ABNT, Copyright © 1989.

MARCONI, Marina de A. & LAKATOS, Eva M. **Fundamentos de metodologia científica**. 5 ed. Editora Atlas. São Paulo, 2003.

VAHID, Frank. **Sistemas Digitais: Projetos, Otimização e HDLs**. 1 ed. Editora Bookman, 2008.

TOCCI, Ronald J. **Digital Systems: principles and applications**. 11 ed. Pearson Education India, 1991.

CÓDIGO FONTE

```
2. library ieee;
3. use ieee.std_logic_1164.all;
4. use ieee.std_logic_signed.all;
5. use ieee.std_logic_unsigned.all;
6. use ieee.numeric_std.all;
7.
8. entity parkOmeter is
9.     port(
10.         b, c, f, rst, d: in std_logic;
11.         A: in std_logic_vector (2 downto 0);
12.         -- 001: 25 ¢ = 1
13.         -- 010: 50 ¢ = 2
14.         -- 100: 100 ¢ = 4
15.         clk: in std_logic;
16.         estado : buffer integer;
17.         ts, go, ch, td : buffer std_logic_vector (5 downto 0);
18.         tgo, tch : buffer std_logic_vector (7 downto 0);
19.         tt : buffer std_logic_vector (3 downto 0)
20.     );
21.
22. end parkOmeter;
23.
24. architecture behavior of parkOmeter is
25.
26.     type state is (wait_b, sel_type, add, time_count, end_park);
27.     signal actual_s, next_s: state;
28.     signal count : integer := 0;
29.
30. begin
31.     park: process(rst, clk)
32.     begin
33.         if (rst='1') then
34.             td <= "000000";
35.             tgo <= "00000000";
36.             tch <= "00000000";
37.             actual_s <= wait_b;
38.         elsif rising_edge(clk) then
39.             case actual_s is
40.                 when wait_b =>
41.                     estado <= 0;
42.                     ts <= "000000";
43.                     go <= "000000";
44.                     ch <= "000000";
45.                     tt <= "0000";
46.                     if (b='1') then
47.                         next_s <= sel_type;
48.                     else
49.                         next_s <= wait_b;
50.                     end if;
51.                 when sel_type =>
52.                     estado <= 1;
53.                     if (f = '1') then
54.                         next_s <= time_count;
55.                     elsif (f = '0' and c = '0') then
56.                         next_s <= sel_type;
57.                     elsif (f = '0' and c = '1') then
58.                         next_s <= add;
59.                     end if;
60.                 when add =>
61.                     estado <= 2;
62.                     next_s <= sel_type;
```

```

63.                                     ts <=
std_logic_vector(to_unsigned(to_integer
64.
65. (unsigned(ts))+to_integer(unsigned(A)), 6));
66.                                     if (to_integer(unsigned(A)) = 1) then
67.                                         count <= count + 1;
68.                                     end if;
69.                                     if (to_integer(unsigned(ts)) < 16) then
70.                                         case to_integer(unsigned(A)) is
71.                                             when 2 =>
72.                                                 tt <=
std_logic_vector
73.
74. (to_unsigned(to_integer(unsigned(tt))+1, 4));
75.                                     when 4 =>
76.                                         tt <=
std_logic_vector
77.
78. (to_unsigned(to_integer(unsigned(tt))+2, 4));
79.                                     when others =>
80.                                         tt <= tt;
81.                                     end case;
82.                                     if (count = 2) then
83.                                         count <= 0;
84.                                         tt <=
std_logic_vector(to_unsigned
85.
86. (to_integer(unsigned(tt))+1, 4));
87.                                     end if;
88.                                     end if;
89.                                     if (count = 2) then
90.                                         count <= 0;
91.                                     end if;
92.                                     when time_count =>
93.                                         estado <= 3;
94.                                         tt <=
std_logic_vector(to_unsigned(to_integer
95.
96. (unsigned(tt))-1, 4));
97.                                     next_s <= time_count;
98.                                     if ((to_integer(unsigned(tt))=0) or (d =
'1')) then
99.                                         next_s <= end_park;
100.                                     --elsif ((to_integer(unsigned(tt)) > 0)
and (d =
101.
102. '0')) then
103.                                     --next_s <= time_count;
104.                                     end if;
105.                                     when end_park =>
106.                                         next_s <= wait_b;
107.                                         estado <= 4;
108.                                         tt <= "0000";
109.                                         td <=
std_logic_vector(to_unsigned(to_integer
110.
111. (unsigned(td))+1, 6));
112.                                     if (to_integer(unsigned(ts)) < 16) then
113.                                         go <= ts;
114.                                         ch <= "000000";
115.                                         tch <= tch;
116.                                         tgo <= std_logic_vector(to_unsigned
117.
118. (to_integer(unsigned(tgo))+to_integer(unsigned(ts)), 8));
119.                                     elsif (to_integer(unsigned(ts)) >= 16)
then

```

```

120.                                     go <= "010000";
121.                                     ch <= std_logic_vector(to_unsigned
122.
123. (to_integer(unsigned(ts))-16, 6));
124.                                     tch <= std_logic_vector(to_unsigned
125.
126. (to_integer(unsigned(tch))+to_integer(unsigned(ts))-16, 8));
127.                                     tgo <= std_logic_vector(to_unsigned
128.
129. (to_integer(unsigned(tgo))+16, 8));
130.                                     end if;
131.                                     end case;
132.                                     actual_s <= next_s;
133.                                     end if;
134.     end process park;
135. end behavior;

```