

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**](#_heading=h.5r5dhnmfrcch) **3**

[**1.1 Demanda**](#_heading=h.30j0zll) **3**

[**MÁQUINA DE ESTADOS DO PARQUÍMETRO**](#_heading=h.1fob9te) **4**

[**IMPLEMENTAÇÃO EM VHDL**](#_heading=h.3dy6vkm) **7**

[3.1 Codificação](#_heading=h.1t3h5sf) 7

[3.2 Entidade](#_heading=h.4d34og8) 7

[**SIMULAÇÃO**](#_heading=h.17dp8vu) **9**

[**CONCLUSÃO**](#_heading=h.35nkun2) **12**

[**REFERÊNCIAS BIBLIOGRÁFICAS**](#_heading=h.44sinio) **13**

[**CÓDIGO FONTE**](#_heading=h.2jxsxqh) **14**

# MEMORIAL DESCRITIVO

O presente projeto de circuitos lógicos tem o objetivo de aprofundar os conhecimentos acerca do uso d 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.

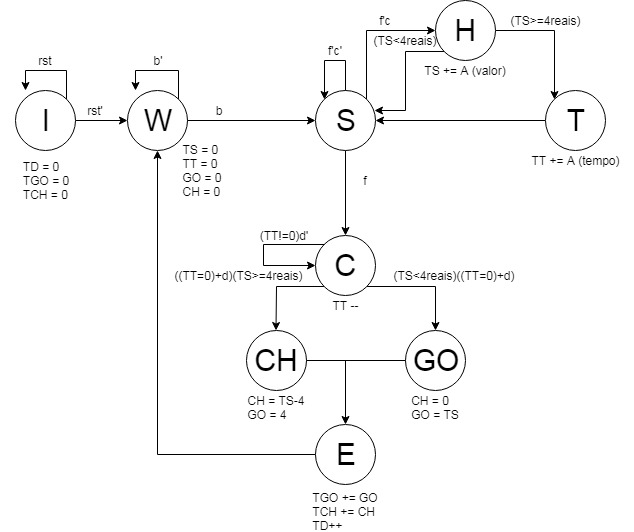
Cada sessão deve ser armazenada em um total diário, para que ao final do dia seja conhecido o número de sessões que ocorreram naquele parquímetro.

O parquímetro associa cada ¢50 a 15 minutos de estacionamento. Se ¢25 são inseridos é aguardado que múltiplos de ¢50 sejam inseridos para somar mais 15 minutos. O tempo máximo é 2 horas. Ou seja, no máximo 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 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

# MÁQUINA DE ESTADOS DO PARQUÍMETRO

O primeiro passo para o projeto é construir a máquina de estados para se ter uma noção das variáveis que vão existir e seus tipos.

**Figura 1 –** Máquina de estados de alto nível

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 eo 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 ¢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 contado 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.

# 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. );
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:

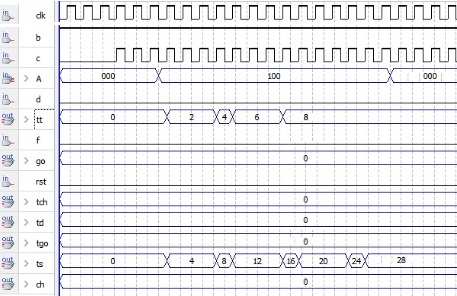
1. **type** state **is** (wait\_b, sel\_type, add, time\_count, end\_park);

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.

# SIMULAÇÃO

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

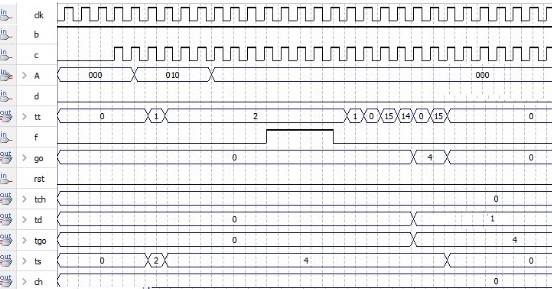
A primeira simulação vai mostrar um cliente que inseriu diversas moedas. A quantia de dinheiro ultrapassou R$4,00 em ts que em binário equivale a 10000=16, mas a quantidade de tempo em tt permaneceu 2h, ou em binário 1000=8.



**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\*15min = 120min = 2h). 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.

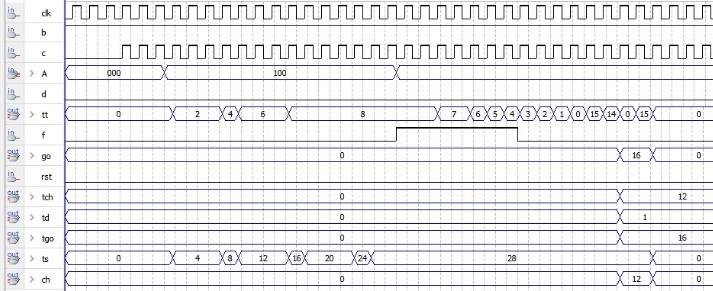


**Figura 3 –** Inserindo moedas de forma que não extrapole R$4,00.

Insere-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\*¢25=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.

# 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.

# 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

1. **library** ieee;
2. **use** ieee.std\_logic\_1164.**all**;
3. **use** ieee.std\_logic\_signed.**all**;
4. **use** ieee.std\_logic\_unsigned.**all**;
5. **use** ieee.numeric\_std.**all**;
7. **entity** parkOmeter **is**
8. **port**(
9. b, c, f, rst, d: **in** std\_logic;
10. A: **in** std\_logic\_vector (2 **downto** 0);
11. *-- 001: 25 ¢ = 1*
12. *-- 010: 50 ¢ = 2*
13. *-- 100: 100 ¢ = 4*
14. clk: **in** std\_logic;
15. estado : **buffer** integer;
16. ts, go, ch, td : **buffer** std\_logic\_vector (5 **downto** 0);
17. tgo, tch : **buffer** std\_logic\_vector (7 **downto** 0);
18. tt : **buffer** std\_logic\_vector (3 **downto** 0)
19. );
21. **end** parkOmeter;
23. **architecture** behavior **of** parkOmeter **is**
25. **type** state **is** (wait\_b, sel\_type, add, time\_count, end\_park);
26. **signal** actual\_s, next\_s: state;
27. **signal** count : integer := 0;
29. **begin**
30. park: **process**(rst, clk)
31. **begin**
32. **if** (rst='1') **then**
33. td <= "000000";
34. tgo <= "00000000";
35. tch <= "00000000";
36. actual\_s <= wait\_b;
37. **elsif** rising\_edge(clk) **then**
38. **case** actual\_s **is**
39. **when** wait\_b =>
40. estado <= 0;
41. ts <= "000000";
42. go <= "000000";
43. ch <= "000000";
44. tt <= "0000";
45. **if** (b='1') **then**
46. next\_s <= sel\_type;
47. **else**
48. next\_s <= wait\_b;
49. **end** **if**;
50. **when** sel\_type =>
51. estado <= 1;
52. **if** (f = '1') **then**
53. next\_s <= time\_count;
54. **elsif** (f = '0' and c = '0') **then**
55. next\_s <= sel\_type;
56. **elsif** (f = '0' and c = '1') **then**
57. next\_s <= add;
58. **end** **if**;
59. **when** add =>
60. estado <= 2;
61. next\_s <= sel\_type;
62. ts <= std\_logic\_vector(to\_unsigned(to\_integer
64. (unsigned(ts))+to\_integer(unsigned(A)), 6));
65. **if** (to\_integer(unsigned(A)) = 1) **then**
66. count <= count + 1;
67. **end** **if**;
68. **if** (to\_integer(unsigned(ts)) < 16) **then**
69. **case** to\_integer(unsigned(A)) **is**
70. **when** 2 =>
71. tt <= std\_logic\_vector
73. (to\_unsigned(to\_integer(unsigned(tt))+1, 4));
74. **when** 4 =>
75. tt <= std\_logic\_vector
77. (to\_unsigned(to\_integer(unsigned(tt))+2, 4));
78. **when** **others** =>
79. tt <= tt;
80. **end** **case**;
81. **if** (count = 2) **then**
82. count <= 0;
83. tt <= std\_logic\_vector(to\_unsigned
85. (to\_integer(unsigned(tt))+1, 4));
86. **end** **if**;
87. **end** **if**;
88. **if** (count = 2) **then**
89. count <= 0;
90. **end** **if**;
91. **when** time\_count =>
92. estado <= 3;
93. tt <= std\_logic\_vector(to\_unsigned(to\_integer
95. (unsigned(tt))-1, 4));
96. next\_s <= time\_count;
97. **if** ((to\_integer(unsigned(tt))=0) or (d = '1')) **then**
98. next\_s <= end\_park;
99. *--elsif ((to\_integer(unsigned(tt)) > 0) and (d =*
101. '0')) **then**
102. *--next\_s <= time\_count;*
103. **end** **if**;
104. **when** end\_park =>
105. next\_s <= wait\_b;
106. estado <= 4;
107. tt <= "0000";
108. td <= std\_logic\_vector(to\_unsigned(to\_integer
110. (unsigned(td))+1, 6));
111. **if** (to\_integer(unsigned(ts)) < 16) **then**
112. go <= ts;
113. ch <= "000000";
114. tch <= tch;
115. tgo <= std\_logic\_vector(to\_unsigned
117. (to\_integer(unsigned(tgo))+to\_integer(unsigned(ts)), 8));
118. **elsif** (to\_integer(unsigned(ts)) >= 16) **then**
119. go <= "010000";
120. ch <= std\_logic\_vector(to\_unsigned
122. (to\_integer(unsigned(ts))-16, 6));
123. tch <= std\_logic\_vector(to\_unsigned
125. (to\_integer(unsigned(tch))+to\_integer(unsigned(ts))-16, 8));
126. tgo <= std\_logic\_vector(to\_unsigned
128. (to\_integer(unsigned(tgo))+16, 8));
129. **end** **if**;
130. **end** **case**;
131. actual\_s <= next\_s;
132. **end** **if**;
133. **end** **process** park;
134. **end** behavior;