Projeto Final de Laboratório de Sistemas Digitais

Gabriel Matheus
Departamento de Engenharia Elétrica
Universidade de Brasília
Brasília, Brasil
17/0103498
shudugen2@gmail.com

Gabriel Moretto
Departamento de Engenharia Elétrica
Universidade de Brasília
Brasília, Brasil
15/0154917
gabrielmorettopvh@yahoo.com.br

Guilherme Braga Pinto
Departamento de Engenharia Elétrica
Universidade de Brasília
Brasília, Brasil
17/0162290
guib545@aluno.unb.br

***Resumo*—Este relatório tem como objetivo constatar categoricamente o que foi pedido, mostrando a elaboração em VHDL com o uso do ISE Webpack do controlador Myca-2, conjuntamente à sua aplicação na simulação de uma máquina de refrigerantes análoga à máquina feita em experimentos anteriores. Acomapnha-se representações visuais do projeto como um todo.**

I. Introdução

Vimos em anos recentes os grandes avanços de sistemas computadorizados, com a tendência cada vez maior à diminuição do tamanho dos componentes usados no dia-a-dia. Já fazem décadas que a resolução de problemas está cada vez mais relacionada à buscar-se uma solução em um nível lógico de grande complexidade, no lugar de se buscar uma solução de "força bruta". Este trabalho tem como foco um determido controlador, implementado em grupo.

Um controlador pode ter sua definição resumida por ser um software ou aparelho que processa diretamente o fluxo de dados entre duas entidades [1]. Podem ser microchips, placas ou outros hardwares que fornecem o controle de um dado periférico. O objetivo primário deste trabalho é desenvolver um controlador quer possa ser programado e reutilizado futuramente, simulando-o como uma entidade escrita em VHDL (VHSIC Hardware Description Language).

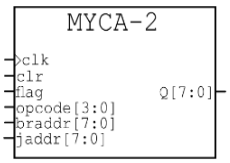

Figura 1. Esquemático de entradas e saídas do Myca-2

II. O que é um controlador qual é o uso do Myca-2?

Um dispositivo microprocessado como descrito anteriormente deve facilmente implementar um dado problema, baseando-se, por exemplo, em um diagrama de estados (onde cada estado pode ser adaptado para satisfazer uma condição para a progressão do projeto, tal condição podendo ser implementada pelo controlador). Em cima disso, cria-se uma lógica associando cada estado à condição que correlaciona a sua condição de avanço na lógica. Ou seja, descreve-se categoricamente o problema usando as ferramentas disponibilizadas pelo controlador.

O Myca-2 apresenta uma tabela de funcionamento como a mostrada abaixo. Com um clock, o controlador lê o Operation Code (Opcode, atrelado à um mnemônico específico), que definirá a operação a ser feita. Existem diferentes combinações, por exemplo, se contar um estado para o estado seguinte caso uma condição seja satisfeita, ou um branch ocorrerá. A entrada Branch Adress (braddr) informa o novo estado para o qual o controlador irá se dirigir caso este seja o caso, e a entrada de Flag mostra a qual condição se deve satisfazer para que dada operação ocorra. O diferencial deste controlador em comparação à sua versão mais simples é a possibilidade de se "pular para uma sub-rotina", o que explica os últimos estados de sua tabela, assim como a entrada Jam Adress (jaddr). A saída "Q"do controlador nos informa seu estado atual. A entrada de flag deve ser combinada com um Mux para que o uso do controlador seja abrangente [2].

Tabela I
Instruções implementadas pelo Myca-2

Mnemônico	Operação caso Flag = 0	Operação caso Flag = 1
HIC	Mantém estado atual	Incrementa estado
HBC	Mantém estado atual	Executa branch
IBC	Incrementa estado	Executa branch
IUC	Incrementa estado	Incrementa estado
BUC	Executa branch	Executa branch
BSR	Incrementa estado	Executa branch para sub-rotina
RSR	Executa branch	Vai para o topo de pilha
LJA	Incrementa estado	Executa branch para jam Address

III. Como funciona o Myca-2 (myca2.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity myca2 is
    Port (
        clk: in STD_LOGIC;
```

```
        clr: in STD_LOGIC;
        flag : in STD_LOGIC;
        opcode : in
        STD_LOGIC_VECTOR (3 downto 0);
        braddr : in
        STD_LOGIC_VECTOR (7 downto 0);
        jaddr : in
        STD_LOGIC_VECTOR (7 downto 0);
        Q : out
        STD_LOGIC_VECTOR (7 downto 0));
end myca2;

architecture logic of myca2 is

component MUX4_8 is
port(

  D0: in STD_LOGIC_VECTOR(7 downto 0);
  D1: in STD_LOGIC_VECTOR(7 downto 0);
  D2: in STD_LOGIC_VECTOR(7 downto 0);
  D3: in STD_LOGIC_VECTOR(7 downto 0);
  S1: in STD_LOGIC_VECTOR(1 downto 0);
  Y1: out STD_LOGIC_VECTOR(7 downto 0));

end component;

component FullAdder_8bits is
  Port(
    A, B : in std_logic_vector(7 downto 0);
    cin : in std_logic;
    S : out std_logic_vector(7 downto 0);
    cout : out std_logic
    );
end component;

component opcodeDecoder is
    Port (
           deco_in : in
           STD_LOGIC_VECTOR (3 downto 0);
           deco_out : out
           STD_LOGIC_VECTOR (6 downto 0));
end component;

component stack is
  port
  (
  clk_2:  in  std_Logic;
  clr_2:  in  std_Logic;
  S_aux: in std_Logic_vector(1 downto 0);
  A: in std_Logic_vector(7 downto 0);
  Q_aux: out std_Logic_vector(7 downto 0)
  );

end component;

component unidade_updown is
    Port (
        updown  : in STD_LOGIC;
        clr : in STD_LOGIC;
        en  : in STD_LOGIC;
        clk : in STD_LOGIC;
        lkd : in STD_LOGIC;
        D   : in
        STD_LOGIC_VECTOR(7 downto 0);
        off : out STD_LOGIC;
        Q   : out
        STD_LOGIC_VECTOR(7 downto 0)
        );
end component;

signal Cout_Full_Adder, off_f: STD_LOGIC;
signal deco_in_f:
STD_LOGIC_VECTOR (3 downto 0);
signal deco_out_f:
STD_LOGIC_VECTOR (6 downto 0);
signal Out_Contador_up_down,
Out_FullAdder: STD_LOGIC_VECTOR (7 downto 0);
signal Out_stack,
Out_MUX: STD_LOGIC_VECTOR (7 downto 0);
signal entrada_mux: STD_LOGIC_VECTOR (1 downto 0);

begin

deco_in_f(0) <= opcode(0);
deco_in_f(1) <= opcode(1);
deco_in_f(2) <= flag;

entrada_mux(0) <= deco_out_f(3);
entrada_mux(1) <= deco_out_f(4);


U0: opcodeDecoder port map ( deco_in_f,
deco_out_f);

U1: FullAdder_8bits port map("00000000",
Out_Contador_up_down, deco_out_f(0),
Out_FullAdder, Cout_Full_Adder);

U2: stack port map(clk, clr,
deco_out_f(2 downto 1),
Out_FullAdder, Out_stack);

U3: MUX4_8 port map(jaddr, jaddr,
Out_stack, "00000000", entrada_mux,
Out_MUX);

U4: unidade_updown port map('0', clr,
deco_out_f(5), clk, deco_out_f(6),
Out_MUX, off_f, Out_Contador_up_down);

Q <= Out_Contador_up_down;
```

```
end logic;
```

IV. Máquina de refrigerante com o Myca-2

Desenvolvemos, após a elaboração geral do Myca-2, a máquina de refrigerantes que foi préviamente implementada em sala de aula. Esta é uma Máquina de Moore, ou seja, de estados finitos. As saídas dadas ao usuário são definidas unicamente pelo estado no qual a máquina se encontra, diferente de uma Máquina de Mealy [2].

A grande responsabilidade da máquina é contar uma determinada entrada, e quando dada condição for aceita, retornar uma flag. Esta operação é representada pela contagem de moedas do programa. Se a contagem não chega em seu objetivo (30 centavos), espera-se mais moedas. Se há exatamente 30 moedas, retornamos o refrigerante para o consumidor. Se o preço do refirgerante é ultrapassado, existe uma tratamento para converter este valor em um troco.

Tabela II
Representação dos Inputs e dos Outputs da Máquina de Refri

Varíavel	Responsabilidade
Moeda adicionada	Input
Moeda Retirada	Input
S=20 e S>30	Input
Refrigerante retirado	Input
Clear	Input
Clear Contador	Output
Troco	Output
Returna moeda	Output
Indicador de disponibilidade do Refrigerante	Output
Somar	Output
Load	Output

A. Diagrama de Estados

Em complemento ao diagrama, há a tabela de definição da transição dos estados, que define a condição para que cada operação seja implementada (flag para que seja feita a contagem e o endereço para qual se destina um branch.

Tabela III
Operações da Máquina de Refrigerantes

Estado	Instrução	Saída
A	HIC se CP = 1	Não há
B	HIC se CP' = 1	Não há
C	IBC se S<30 = 1	Não há
D	BSR se (S=30)' = 1	Não há
E	HIC se PDR = 1	DP = 1
F	BUC	CA = 1
G	HIC se CR = 1	RN = 1
H	IUC	DA = 1
I	HBC se CR' = 1	Não há
J	RSR se (S=30) = 1	Não há

A string a qual a ROM (read only memory) representa, em ordem: o código da operação a ser implementada, o endereço do Mux o qual repassará a flag, o branch address para operações onde há a necessidade e as saídas.

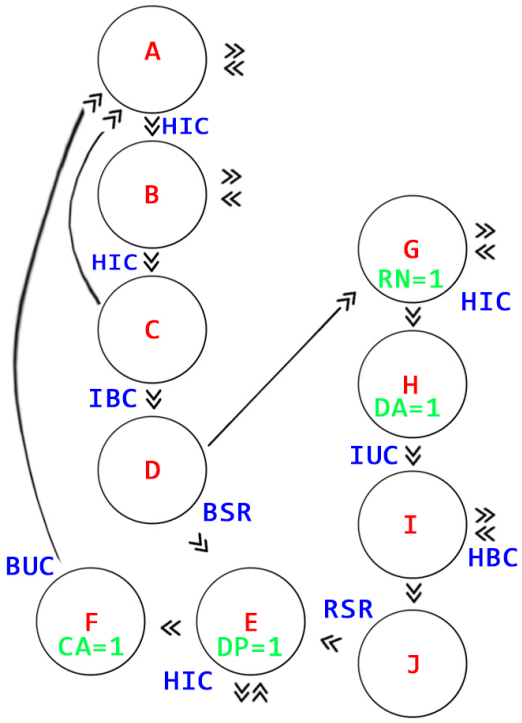

Figura 2. Diagrama de estados da Máquina de Refrigerantes, com foco em mostrar as instruções usadas do controlador

Tabela IV
O que está escrito na ROM para o funcionamento da Máquina de Refrigerantes

Estado atual	Saída respectiva da ROM
A	0000000000000001000
B	0000010000000000100
C	1000110000000000000
D	1010100000110000000
E	0001010000000000001
F	0110000000000100000
G	0001100000000000010
H	0100000000000010000
I	0001110000000000000
J	1101000000110000000

B. Conteúdo da ROM

V. Como funciona a máquina de refrigerante

A. Top Module ($Maquina_Refri.vhd$)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity Maquina_Refri is
    Port (
        COIN: in
        STD_LOGIC_VECTOR (1 downto 0);
        clk: in STD_LOGIC;
        clr: in STD_LOGIC;
```

```
        MOEDA_ACK :
        in STD_LOGIC;
        MOEDA_RETIRADA : in STD_LOGIC;
        REFRI_RETIRADO : in STD_LOGIC;
        RETURN_COIN : out STD_LOGIC;
        REFRI_DISPONIVEL: out STD_LOGIC;
                output_7seg :
                out STD_LOGIC_VECTOR (6 downto 0);
                output_an :
                out STD_LOGIC_VECTOR (3 downto 0)
                );

end Maquina_Refri;

architecture logic of Maquina_Refri is

component somador_reg is
    Generic( n : integer := 4
    );
    Port ( clk : in STD_LOGIC;
        clr : in STD_LOGIC;
        en : in STD_LOGIC;
        A : in
        STD_LOGIC_VECTOR (n-1 downto 0);
        B : in
        STD_LOGIC_VECTOR (n-1 downto 0);
        S : out
        STD_LOGIC_VECTOR (n-1 downto 0));
end component;

component decoMoeda is
    Port ( COIN_IN :
        in STD_LOGIC_VECTOR (1 downto 0);
        DECOIN_OUT :
        out STD_LOGIC_VECTOR (3 downto 0));
end component;

component contador_74LS169 is
    Port ( clk : in STD_LOGIC;
        clr : in STD_LOGIC;
        lkd : in STD_LOGIC;
        en : in STD_LOGIC;
        up_down : in STD_LOGIC;
        D : in
        STD_LOGIC_VECTOR (3 downto 0);
        o_f : out STD_LOGIC;
        Q : out
        STD_LOGIC_VECTOR (3 downto 0));
end component;

component deco7seg is
    Generic (clk_freq :
        integer := 100000000; -- 100MHz
        refresh_freq :
        integer := 30 -- 30Hz
    );
    Port ( clk_7seg : in STD_LOGIC;
        rst : in std_logic;
        input_1 : in
        STD_LOGIC_VECTOR (3 downto 0);
        input_2 : in
        STD_LOGIC_VECTOR (3 downto 0);
        input_3 : in
        STD_LOGIC_VECTOR (3 downto 0);
        input_4 : in
        STD_LOGIC_VECTOR (3 downto 0);
        output_7seg :
        out STD_LOGIC_VECTOR (6 downto 0);
        output_an :
        out STD_LOGIC_VECTOR (3 downto 0));
end component;

component comp4bit is port(
  a, b : in std_logic_vector(3 downto 0);
    agtbin, aeqbin,
    altbtin : in std_logic;
    agtbout, aeqbout,
    altbtout : out std_logic);
end component;

component myca2 is
    Port (
        clk: in STD_LOGIC;
        clr: in STD_LOGIC;
        flag : in STD_LOGIC;
        opcode : in
        STD_LOGIC_VECTOR (3 downto 0);
        braddr : in
        STD_LOGIC_VECTOR (7 downto 0);
        jaddr : in
        STD_LOGIC_VECTOR (7 downto 0);
        Q : out
        STD_LOGIC_VECTOR (7 downto 0));
    end component;

component ROM is

port(

addr : in std_logic_vector(7 downto 0);
d_out : out std_logic_vector(19 downto 0) );

end component;

component flag_myca is

    port(

    D0_F, D1_F,
    D2_F, D3_F,
    D4_F, D5_F,
    D6_F, D7_F: in STD_LOGIC;
```

```
    Sel_F: in
    STD_LOGIC_VECTOR(2 downto 0);
    S_M: out STD_LOGIC
  );

end component;

signal Resul_Detc_Moeda, ResSoma,
Preco_Refri: STD_LOGIC_VECTOR (3 downto 0);

signal Res_contador_74LS169:
STD_LOGIC_VECTOR (3 downto 0);

signal En_contador,
clr_contador, o_f : STD_LOGIC;

signal agtbout,
aeqbout, altbtout : STD_LOGIC;

signal CLR_CONT, DEC_TROCO,
SOMAR, LOAD_CNT: STD_LOGIC;


signal braddr_myca2,
Out_myca2: STD_LOGIC_VECTOR(7 downto 0);

signal opcode_myca2:
STD_LOGIC_VECTOR(3 downto 0);

signal Sel_Flag:
STD_LOGIC_VECTOR(2 downto 0);

signal Out_Flag: STD_LOGIC;

signal Out_ROM:
STD_LOGIC_VECTOR(19 downto 0);

signal display_braddr_myca2:
STD_LOGIC_VECTOR(3 downto 0);

signal display_opcode:
STD_LOGIC_VECTOR(3 downto 0);

signal not_moeda_ack,
not_moeda_retirada: STD_LOGIC;

begin

Preco_Refri <= "0110";
not_moeda_ack <= not(MOEDA_ACK);
not_moeda_retirada <=
not(MOEDA_RETIRADA);

U0 : decoMoeda port map (COIN,
Resul_Detc_Moeda);

U1: somador_reg port map(clk, clr,
SOMAR, Resul_Detc_Moeda ,
Res_contador_74LS169, ResSoma);

En_contador <= (LOAD_CNT or
DEC_TROCO);

clr_contador <= (clr or CLR_CONT);

U2: contador_74LS169 port map(clk,
clr_contador, LOAD_CNT, En_contador,
'0', ResSoma, o_f, Res_contador_74LS169);

U3: comp4bit port map(Res_contador_74LS169,
Preco_Refri, '0', '1', '0', agtbout,
aeqbout, altbtout);

U4: flag_myca port map(MOEDA_ACK,
not_moeda_ack, agtbout, altbtout,
aeqbout, REFRI_RETIRADO, MOEDA_RETIRADA,
not_moeda_retirada, Sel_Flag, Out_Flag);

U5: myca2 port map(clk, clr, Out_Flag,
opcode_myca2, braddr_myca2,
"00000000", Out_myca2);

U6: ROM port map(Out_myca2, Out_ROM);

REFRI_DISPONIVEL <= Out_ROM(0);
RETURN_COIN <= Out_ROM(1);
LOAD_CNT <= Out_ROM(2);
SOMAR <= Out_ROM(3);
DEC_TROCO <= Out_ROM(4);
CLR_CONT <= Out_ROM(5);

--braddr_myca2 <= Out_ROM(13 downto 6);

braddr_myca2(7) <= Out_ROM(13);
braddr_myca2(6) <= Out_ROM(12);
braddr_myca2(5) <= Out_ROM(11);
braddr_myca2(4) <= Out_ROM(10);
braddr_myca2(3) <= Out_ROM(9);
braddr_myca2(2) <= Out_ROM(8);
braddr_myca2(1) <= Out_ROM(7);
braddr_myca2(0) <= Out_ROM(6);


--Sel_Flag <= Out_ROM(16 downto 14);
Sel_Flag(2) <= Out_ROM(16);
Sel_Flag(1) <= Out_ROM(15);
Sel_Flag(0) <= Out_ROM(14);

--opcode_myca2 <= Out_ROM(19 downto 17);

opcode_myca2(2) <= Out_ROM(19);
opcode_myca2(1) <= Out_ROM(18);
```

```
opcode_myca2(0) <= Out_ROM(17);

display_braddr_myca2(3) <=
Out_myca2(3);
display_braddr_myca2(2) <=
Out_myca2(2);
display_braddr_myca2(1) <=
Out_myca2(1);
display_braddr_myca2(0) <=
Out_myca2(0);

display_opcode(3) <= '0';
display_opcode(2) <= opcode_myca2(2);
display_opcode(1) <= opcode_myca2(1);
display_opcode(0) <= opcode_myca2(0);

U7: deco7seg port map(clk, clr,
Preco_Refri, Res_contador_74LS169,
display_braddr_myca2, display_opcode,
output_7seg, output_an);

end logic;
```

Os componenetes estão na página no Github, podendo ser acessado em um link no final do relatório.

VI. Equemáticos em RTL

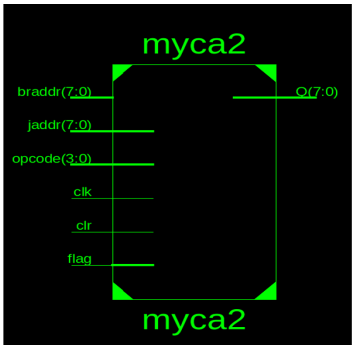

Figura 3. RTL do Myca-2, visão externa para seu uso facilitado

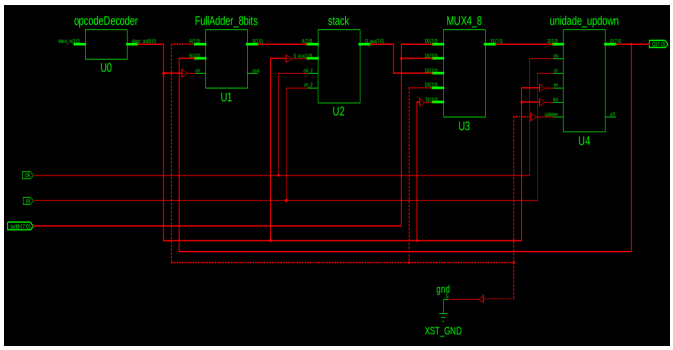

Figura 4. RTL do Myca-2, visão interna de seus componentes

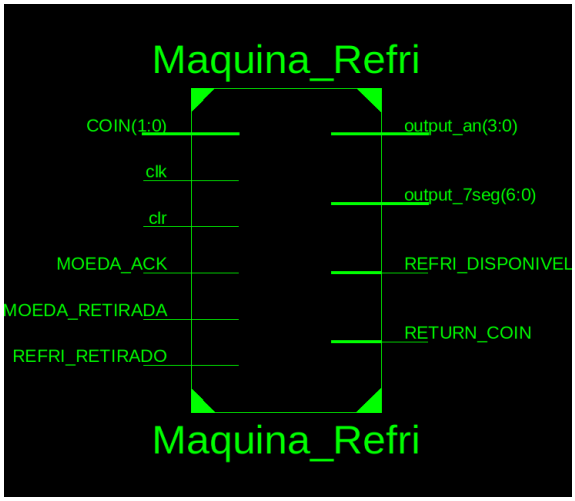

Figura 5. Exterior da Máquina de Refrigerantes

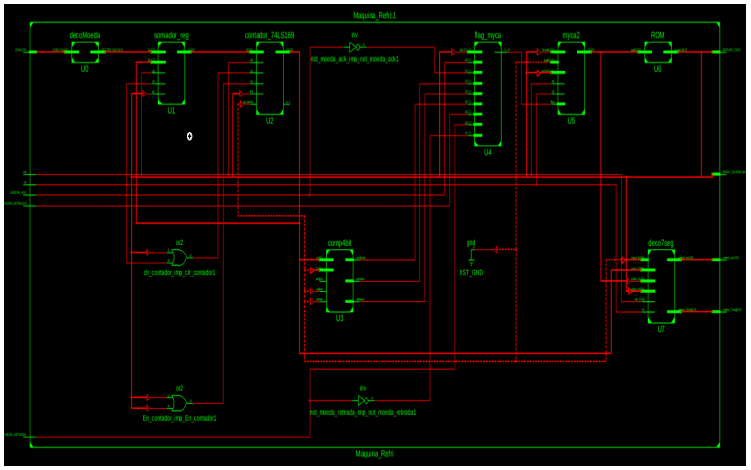

Figura 6. Interior da Máquina de Refrigerantes

VII. Link para o trabalho no Github (com os códigos e os esquemáticos)

Clique neste parágrafo para ser redirecionado para o link, onde os esquemáticos originais estão armazenados, junto ao código completo. Usuário: therealguib545, no repositório: Lab-SD-Projeto-Final-1-2019

Agradecimentos

Agradecemos ao professor da matéria, Sergio Andres Pertuz Mendez, aos professores da matéria de Sistemas Digitais da UnB, aos nossos colegas do curso de Engenharia da Computação e do curso de Engenharia Elétrica.

Referências

[1] Margaret Rouse, *What is a Controller?*
[2] John F. Wakerly, *Digital Design: Principles and Practices*