Relatório de Labolatório de Sistemas Digitais Avançados

Prática 5 – Timers and Real-Time Clock

Alunos:

José David Sousa de Araújo– Matricula: 1020678

Judah Holanda Correia Lima – Matricula: 1010500/5

José Kilson de Souza Oliveira – Matricula: 0810847/1

Sumário

[Material Utilizado 3](#_Toc388649935)

[Prática 5 – Timers and Real-Time Clock 4](#_Toc388649936)

[INTRODUÇÃO 4](#_Toc388649937)

[Códigos Auxiliares 4](#_Toc388649938)

[Módulo Removedor de Bounce 4](#_Toc388649939)

[Módulo Contador de Tamanho Variável 4](#_Toc388649940)

[Módulo Codificador para o Display de 7 Segmentos BCD (B2D) 5](#_Toc388649941)

[Módulo Codificador de dois digitos BCD (B2D) 5](#_Toc388649942)

[Parte 1 7](#_Toc388649943)

[Módulo Principal do Projeto 7](#_Toc388649944)

[Parte 2 8](#_Toc388649945)

[Módulo Principal do Projeto 8](#_Toc388649946)

[Parte 3 10](#_Toc388649947)

[Módulo Principal do Projeto 10](#_Toc388649948)

[Parte 4 12](#_Toc388649949)

[Módulo Principal do Projeto 12](#_Toc388649950)

[Conclusão 15](#_Toc388649951)

# Material Utilizado

* Kit de desenvolvimento DE0
* Software Quartus2 13.1 fornecido pelo fabricante Altera
* Sistema operacional: Windows 7 64 bits Serve Pack 1 e Windows 8.1 64 bits Serve Pack 1
* Estação de trabalho: Notebooks Core i7 Segunda Geração 4 Gb de Ram (Windows 7) e Core i7 Segunda Geração 8 Gb de Ram (Windows 8.1)

# Prática 5 – Timers and Real-Time Clock

## INTRODUÇÃO

Um temporizador é um dispositivo capaz de medir o [tempo](http://pt.wikipedia.org/wiki/Tempo), sendo um tipo de [relógio](http://pt.wikipedia.org/wiki/Rel%C3%B3gio) especializado. Ele pode ser usado para controlar a sequência de um [evento](http://pt.wikipedia.org/wiki/Evento) ou [processo](http://pt.wikipedia.org/wiki/Processo). Temporizadores podem ser [mecânicos](http://pt.wikipedia.org/wiki/Mec%C3%A2nico), [eletromecânicos](http://pt.wikipedia.org/wiki/Electromec%C3%A2nico), [digitais](http://pt.wikipedia.org/wiki/Digital), ou mesmo [programas de computador](http://pt.wikipedia.org/wiki/Programas_de_computador), uma vez que os [computadores](http://pt.wikipedia.org/wiki/Computador) contêm [relógios](http://pt.wikipedia.org/wiki/Rel%C3%B3gio_de_tempo_real).

Na arquitetura de circuitos [microprocessados](http://pt.wikipedia.org/wiki/Microprocessador" \o "Microprocessador), os temporizadores são utilizados para gerar bases de tempo que podem ser utilizadas para os mais diversos fins, como por exemplo gerar [sinais](http://pt.wikipedia.org/wiki/Sinal_(electr%C3%B3nica)) de [clock](http://pt.wikipedia.org/wiki/Clock" \o "Clock) para outros periféricos do chip, calcular intervalos de tempo ou medir período de sinais. Na arquitetura dos [microcontroladores](http://pt.wikipedia.org/wiki/Microcontrolador" \o "Microcontrolador) mais utilizadas atualmente no meio acadêmico "MCS51" e o "PIC16" estão presentes temporizadores de [8](http://pt.wikipedia.org/wiki/8_bits) e [16](http://pt.wikipedia.org/wiki/16_bits), esses podendo ser configurados pelo usuário para trabalhar de formas distintas.

Esta prática tem como finalidade o estudo do uso de clock em circuitos temporizados. Feitos na placa DE0 da Altera.

A descrição de hardware desta prática foi feita em Verilog no Kit de desenvolvimento da empresa DE0, fornecido pela Unifor.

## Códigos Auxiliares

### Módulo Removedor de Bounce

Esse módulo tem como finalidade a retirada do bounce de alguma chave do circuito.

module DeBounce (En, Clk, Clr, Q);

input En, Clk, Clr;

output reg Q;

always @ (posedge Clk)

begin

if (~Clr)

Q = 0;

else if (En)

Q = ~Q;

end

endmodule

### Módulo Contador de Tamanho Variável

Criando um contador de módulo-k, modificando o desenho de um contador de 8 bits para conter um parâmetro adicional. O contador irá contar de 0 a k -1, quando o contador atingir o valor k -1 o valor seguinte será 0.

Esse módulo tem como finalidade criar um contador de 0 a k-1.

module counterModK(clk, cln, Q);

parameter n ;

parameter k ;

input clk, cln;

output [n-1:0] Q;

reg [n-1:0] Q;

always @(posedge clk or negedge cln)

begin

if (~cln)

Q <= 'd0;// Zera contador quando acinona o clear

else begin

Q <= Q + 1'b1; // incrementa

if (Q == k-1)

Q <= 'd0; // Zera contador quando estoura o tamanho

end

end

endmodule

### Módulo Codificador para o Display de 7 Segmentos BCD (B2D)

module b2d\_ssd (X, SSD);

input [4:0] X;

output reg [0:6] SSD;

always begin

case(X)

0:SSD=7'b0000001;

1:SSD=7'b1001111;

2:SSD=7'b0010010;

3:SSD=7'b0000110;

4:SSD=7'b1001100;

5:SSD=7'b0100100;

6:SSD=7'b0100000;

7:SSD=7'b0001111;

8:SSD=7'b0000000;

9:SSD=7'b0001100;

endcase

end

endmodule

### Módulo Codificador de dois digitos BCD (B2D)

module twodigit\_b2d\_ssd (X, SSD1, SSD0);

input [6:0] X;

output [0:6] SSD1, SSD0;

reg [3:0] ONES, TENS;

always begin// separa unidade e dezena

ONES = X % 10;

TENS = (X - ONES) / 10;

end

b2d\_ssd B1 (TENS, SSD1);

b2d\_ssd B0 (ONES, SSD0);

endmodule

## Parte 1

Este exercício tem como finalidade criar um contador de tamanhos diferentes sem usar módulos diferentes para cada tamanho:

### Módulo Principal do Projeto

Em “module **part1** (SW, LEDG);” é feita a declaração do módulo da lógica do circuito, como os sinais nele existentes.

Os conteúdos do contador vão ser exibidos nos LEDs verdes.

module part1(SW,LEDG);

input [9:0] SW;

output [9:0] LEDG;

wire Bt;

DeBounce D0(1, SW[4], SW[3], Bt);//Retira Bounce

counterModK counter0 (Bt, SW[0], LEDG[3:0]); // mostra o valor do contador nos LED’s 0-3

defparam counter0.n = 4;

defparam counter0.k = 9;

endmodule

Os testes da parte 1 foram realizados por sinais produzidos pelas chaves e as saídas verificadas através dos Leds, contidos no kit de desenvolvimento da altera. Quando aplicado o referido sinal a saída apresentou-se como esperado, então pode-se concluir que a implementação foi bem sucedida.

## Parte 2

Esta prática tem como finalidade implementar um circuito contador BCD – 3 dígitos. O conteúdo do contador será exibido nos displays de sete segmentos (HEX2 – 0). Será o sinal de controle será feito a partir do sinal de clock de 50 MHz proveniente da placa DE0 a intervalos de um segundo.

### Módulo Principal do Projeto

Em “module **part2** (CLOCK\_50, SW, LEDG, HEX0, HEX1, HEX2);” é feita a declaração do módulo da lógica do circuito, como os sinais nele existentes.

Os conteúdos do contador vão ser exibidos nos LEDs verdes e nos Displays.

module part2(CLOCK\_50,SW,LEDG, HEX0, HEX1, HEX2);

input CLOCK\_50;

input [9:0] SW;

output [9:0] LEDG;

output [0:6] HEX0;

output [0:6] HEX1;

output [0:6] HEX2;

wire [3:0] out;

wire [25:0] cycles;

wire [4:0] buffer;

wire [2:0]clear;

wire [25:0] CYCLES;

reg ones, tens, hundreds;

wire x2, y2, z2;

reg x1, y1, z1;

assign x2 = x1 && SW[0];

assign y2 = y1 && SW[0];

assign z2 = z1 && SW[0];

counterModK C0 (CLOCK\_50, SW[0], CYCLES);//conta 1s

defparam C0.n = 26;

defparam C0.k = 50000000; // o contador fará a contagem até 50M, pois como o clock é de 50 MHz, isso resultará em 1s. 50M/50MHz = 1s.

counterModK C1 (ones, x2, LEDG[3:0]);// conta unidades

defparam C1.n = 4;

defparam C1.k = 10;

counterModK C2 (tens, y2, LEDG[7:4]); // conta dezenas

defparam C2.n = 4;

defparam C2.k = 10;

counterModK C3 (hundreds, z2, out[3:0]); // conta centenas

defparam C3.n = 4;

defparam C3.k = 10;

always @ (negedge CLOCK\_50) begin

if (CYCLES == 49999999)

ones = 1; // Completou uma unidade

else

ones = 0; // Unidade não formada

if (LEDG[3:0] == 10) begin

tens = 1; // Completou uma dezena

x1 = 0;

end else begin

tens = 0; // Dezena não formada

x1 = 1;

end

if (LEDG[7:4] == 10) begin

hundreds = 1; // Completou uma centena

y1 = 0;

end else begin

hundreds = 0; // Centena não formada

y1 = 1;

end

z1 = 1;

end

b2d\_ssd H2 (out[3:0], HEX2);// mostra a centena

b2d\_ssd H1 (LEDG[7:4], HEX1);// mostra a dezena

b2d\_ssd H0 (LEDG[3:0], HEX0);// mostra a unidade

endmodule

Os testes da parte 2 foram realizados por sinais produzidos pelas chaves e as saídas verificadas através dos displays de 7 segmentos, contidos no kit de desenvolvimento da altera. Quando aplicado o referido sinal a saída apresentou-se como esperado, então pode-se concluir que a implementação foi bem sucedida.

## Parte 3

Esta prática tem como finalidade implementar um circuito que atua como um relógio de tempo real. Ele irá exibir os minutos (de 0 a 60) em HEX3 - 2 e os segundos (de 0 a 60) em Hex1 - 0.

### Módulo Principal do Projeto

Em “module **part3** (CLOCK\_50, SW, LEDG, HEX3, HEX2, HEX1, HEX0);” é feita a declaração do módulo da lógica do circuito, como os sinais nele existentes.

Os conteúdos do relógio vão ser exibidos nos LEDs verdes e nos Displays.

module part3 (CLOCK\_50, SW, LEDG, HEX3, HEX2, HEX1, HEX0);

input CLOCK\_50;

input [9:0] SW;

output [9:0] LEDG;

output [0:6] HEX0;

output [0:6] HEX1;

output [0:6] HEX2;

output [0:6] HEX3;

wire [25:0] PERSEC;

wire [31:0] PERMIN;

wire [37:0] PERHOUR;

wire [6:0] SECONDS, MINUTES;

wire [6:0] HOURS;

reg sec, min, hour;

counterModK C0 (CLOCK\_50, SW[0], PERSEC); //Conta 1 segundo

defparam C0.n = 26;

defparam C0.k = 50000000; // o contador fará a contagem até 50M, pois como o clock é de 50 MHz, isso resultará em 1s. 50M/50MHz = 1s.

counterModK C1 (CLOCK\_50, SW[0], PERMIN); //Conta 1 minuto

defparam C1.n = 32;

defparam C1.k = 3000000000; // o contador fará a contagem até 3G, pois como o clock é de 50 MHz, isso resultará em 1m. 3G/50MHz = 60s.

counterModK C2 (CLOCK\_50, SW[0], PERHOUR); //Conta 1 hora

defparam C2.n = 38;

defparam C2.k = 180000000000; // o contador fará a contagem até 180G, pois como o clock é de 50 MHz, isso resultará em 1h. 180G/50MHz = 3600s.

counterModK C3 (sec, SW[0], SECONDS);// Conta Segundos

defparam C3.n = 6;

defparam C3.k = 60; // Um minuto tem 60s, ou seja ele conta de 0 a 59

counterModK C4 (min, SW[0], MINUTES);// Conta Minutos

defparam C4.n = 6;

defparam C4.k = 60; // Uma hora tem 60m, ou seja ele conta de 0 a 59

counterModK C5 (hour, SW[0], HOURS);// Conta Horas

defparam C5.n = 5;

defparam C5.k = 24; // Um dia tem 24h, ou seja ele conta de 0 a 23

always @ (negedge CLOCK\_50) begin

if (PERSEC == 49999999)

sec = 1; // Completou 1 segundo

else

sec = 0; // Segundo incompleto

if (PERMIN == 2999999999)

min = 1; // Completou 1 minuto

else

min = 0; // Minuto incompleto

if (PERHOUR == 179999999999)

hour = 1; // Completou 1 hora

else

hour = 0; // Hora incompleta

end

twodigit\_b2d\_ssd D7 (HOURS, HEX3, HEX2); // exibe horas

twodigit\_b2d\_ssd D5 (MINUTES, HEX1, HEX0); // exibe minutos

endmodule

Os testes da parte 3 foram realizados por sinais produzidos pelas chaves e as saídas verificadas através dos displays de 7 segmentos, contidos no kit de desenvolvimento da altera. Quando aplicado o referido sinal a saída apresentou-se como esperado, então pode-se concluir que a implementação foi bem sucedida.

## Parte 4

Esta prática tem como finalidade criar um circuito que tem como entrada uma das primeiras oito letras do alfabeto exibindo-as em código Morse em um LED verde. Quando um usuário pressiona o push button, o circuito deve exibir o código Morse para uma letra especificada pelos switches (000 para A, 001 para B, etc), usando pulsos de 0,5 segundo para representar pontos, e pulsos de 1,5 segundos para representar traços.

### Módulo Principal do Projeto

Em “module **part4** (CLOCK\_50, KEY, SW, LEDG, HEX3, HEX2, HEX1, HEX0);” é feita a declaração do módulo da lógica do circuito, como os sinais nele existentes.

Os conteúdos do código Morse vão ser exibidos no LED verde.

module part4 (CLOCK\_50, KEY, SW, LEDG, HEX3, HEX2, HEX1, HEX0);

input CLOCK\_50;

input [3:0] KEY;

input [2:0] SW;

output reg [9:0] LEDG;

output [0:6] HEX3, HEX2, HEX1, HEX0;

wire [25:0] HALFSEC;

wire HALFPULSE;

wire [3:0] INDEX;

wire reset;

reg half;

reg [13:0] SIGNAL;

// generate half second clock

counterModK C0 (CLOCK\_50, KEY[0], HALFSEC); // Conta ½ segundo (ponto)

defparam C0.n = 26;

defparam C0.k = 25000000; // o contador fará a contagem até 25M, pois como o clock é de 50 MHz, isso resultará em ½s. 25M/50MHz = ½s.

// generate half second pulse (full second signal with 50% duty cycle)

counterModK C1 (half, KEY[0], HALFPULSE); // Conta ½ segundo de pulso

defparam C1.n = 1;

defparam C1.k = 2;

always @ (negedge CLOCK\_50) begin

if (HALFSEC == 24999999)

half = 1;// Completou ½ segundo

else

half = 0;// ½ Segundo incompleto

end

//Sinais de led ligado em ½ Segundo de “passo”

// A o- -\_-- 101100000000

// B -ooo --\_-\_-\_- 110101010000

// C -o-o --\_-\_--\_- 110101101000

// D -oo --\_-\_- 110101000000

// E o - 100000000000

// F oo-o -\_-\_--\_- 101011010000

// G --o --\_--\_- 110110100000

// H oooo -\_-\_-\_- 101010100000

always @ (negedge KEY[1]) begin

case (SW[2:0])

0: SIGNAL = 14'b00101100000000; // A

1: SIGNAL = 14'b00110101010000; // B

2: SIGNAL = 14'b00110101101000; // C

3: SIGNAL = 14'b00110101000000; // D

4: SIGNAL = 14'b00100000000000; // E

5: SIGNAL = 14'b00101011010000; // F

6: SIGNAL = 14'b00110110100000; // G

7: SIGNAL = 14'b00101010100000; // H

endcase

end

assign reset = KEY[1] && KEY[0];

counterModK C2 (HALFPULSE, reset, INDEX);// Contador de pulso

defparam C2.n = 4;

defparam C2.k = 14;

always begin // “Imprime” os pulsos

case (INDEX)

0:LEDG[0] = SIGNAL[13];

1:LEDG[0] = SIGNAL[12];

2:LEDG[0] = SIGNAL[11];

3:LEDG[0] = SIGNAL[10];

4:LEDG[0] = SIGNAL[9];

5:LEDG[0] = SIGNAL[8];

6:LEDG[0] = SIGNAL[7];

7:LEDG[0] = SIGNAL[6];

8:LEDG[0] = SIGNAL[5];

9:LEDG[0] = SIGNAL[4];

10:LEDG[0] = SIGNAL[3];

11:LEDG[0] = SIGNAL[2];

12:LEDG[0] = SIGNAL[1];

13:LEDG[0] = SIGNAL[0];

endcase

end

//always LEDR[1] = HALFPULSE;

Endmodule

Os testes da parte 4 foram realizados por sinais produzidos pelas chaves, e as saídas verificadas através dos Leds de acordo com a tabela mostrada logo abaixo. Quando aplicado o referido sinal a saída apresentou-se como esperado, então pode-se concluir que a implementação foi bem sucedida.



Tabela do Código Morse

# Conclusão

De acordo com a prática realizada, pôde – se aprender a utilização básica desse tipo de circuitos estudados logo acima e conclui-se que o uso de clocks em circuitos temporizados, mostrou-se extremamente útil em uma vasta gama de aplicações. A maior parte da implementação é modularizado, apresentando os resultados esperados da prática, sendo todas as tarefas executadas perfeitamente.