**Projeto Relógio Digital**

**Processamento com Lógica Programável**

**Integrantes:**

Nome: João Gabriel dos Santos Silva TIA: 3164041-9  
Nome: Matheus Andrade Santana TIA: 3168716-4

**Objetivo**

Projetar um relógio digital (utilizando o FPGA , através de VHDL) com alarme ajustável, display com controle de brilho e com duas opções de toque para o alarme.

**Introdução**

Pôde-se aprender ao longo do semestre passado (na disciplina de dispositivo lógico programável) e nesse semestre a capacidade do FPGA de implementar um hardware descrito em HDL e com isso, produzir qualquer circuito combinacional ou sequencial. Neste caso será utilizado para simular, com lógica digital, um relógio digital.

Nesse relógio será possível ajustar o horário, controlar o brilho do display, ajustar alarmes e escolher estre dois tipos de toques.

**Arquitetura**

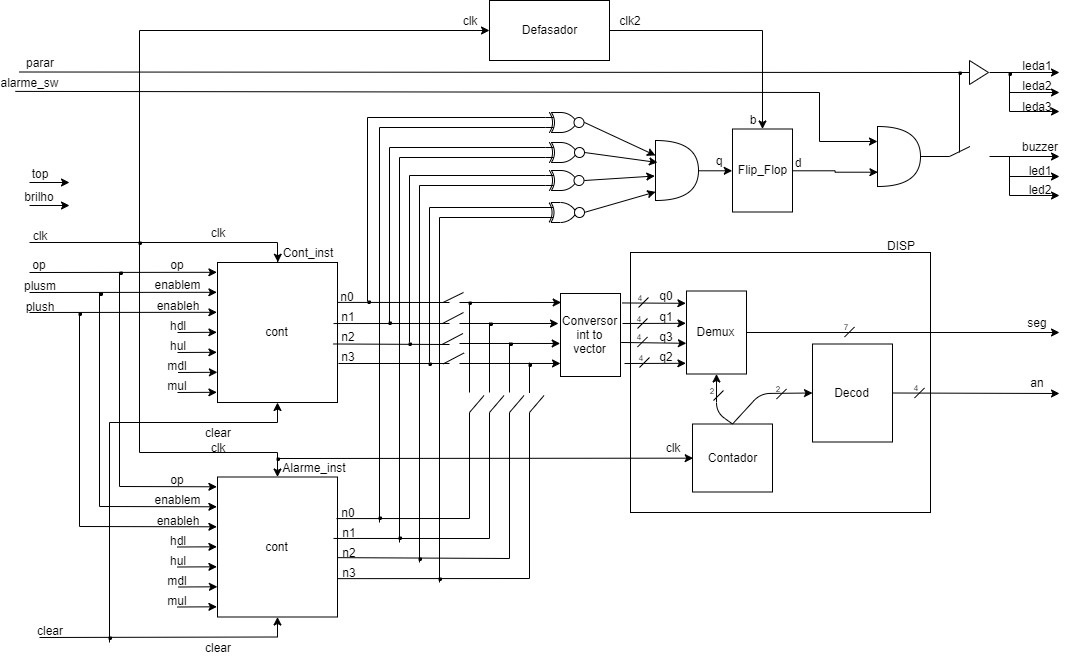
O bloco main é responsável pela comunicação entre os demais componentes do projeto. Além disso, nele é realizada a geração dos clocks utilizados no sistema.

A variação no brilho do display é realizada através da geração de um clock na main em que o display fica apagado durante o estado lógico ‘0’ dele, e aceso no estado ‘1’.

Na main são gerados mais dois clocks para ativação do buzzer, um na frequência de 264 Hz (do) e outro em 495 Hz (si).

No cont são realizados dois tipos de contagem dependendo da opção selecionada no switch. No modo de operação normal, o tempo é incrementado pelos minutos, e subsequentemente as horas. Na opções de ajuste, o incremento nos minutos e nas horas são feitos de forma separada, utilizando um clock de 16 Hz.

O bloco DISP é responsável por agregar os componentes demux, contador e decod para funcionamento do display de 7 segmentos. Ele recebe os valores gerados pelos contadores já convertidos para std\_logic\_vector, demultiplexa os bits para serem enviados para os diplays e varia a frequência de exibição para que o usuário possa enxergar os quatro valores diferentes.



**Implementação VHDL**

**Bloco principal-> main.vhd:**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

--op = 00 relogio normal

--op = 01 ajuste alarme

--op = 10 ajuste relogio

entity main is

port (

clk : in std\_logic;

plusm: in std\_logic;

plush: in std\_logic;

alarme\_sw: in std\_logic;

brilho: in std\_logic;

OP: in STD\_LOGIC\_VECTOR (1 downto 0);

parar: in std\_logic;

top: in std\_logic;

clear: in STD\_LOGIC;

buzzer: out std\_logic;

led1, led2: out STD\_LOGIC:='1';

leda1,leda2,leda3: out STD\_LOGIC;

seg: out STD\_LOGIC\_VECTOR (6 downto 0);

an: out STD\_LOGIC\_VECTOR (3 downto 0)

);

end main;

architecture Behavioral of main is

signal q0,q1,q2,q3,q0a,q1a,q2a,q3a : STD\_LOGIC\_VECTOR(3 downto 0); --vai para o display

signal count,count2,count3,count4,count5,count6 : integer :=1;

signal clkmin, clka, clks, clk2, a ,b, c, spulso, plus1, plus2, plus3, plus4, plus5, plus6, alarme, do, si, buzzer1,buzzer2, leds1, leds2, i1, i2, is1: std\_logic :='0';

signal mua, hua, mda, hda: integer range 0 to 9:=0;

signal muc, huc, mdc, hdc: integer range 0 to 9:=0; --vai para o display depois de ser contado

signal muca, huca, mdca, hdca: integer range 0 to 9:=0; --vai para o display depois de ser contado

signal n0a,n1a,n2a,n3a : integer range 0 to 9 :=0;

signal n0, n1, n2, n3 : integer range 0 to 9 :=0;

signal segl: STD\_LOGIC\_VECTOR (6 downto 0);

signal anl: STD\_LOGIC\_VECTOR (3 downto 0);

signal opl: STD\_LOGIC\_VECTOR (1 downto 0);

constant n: integer := 50000000;

begin

--n0,n1,n2,n3 vão para o contador, saem dele os sinais que vão para o display

Cont\_inst: entity work.Cont

port map (

clk => clkmin,

enablem => plus3,

enableh => plus4,

clear => clear,

op => op,

hdl => n0, --entradas

hul => n1,

mdl => n2,

mul => n3,

n0 => hdc, --saidas

n1 => huc,

n2 => mdc,

n3 => muc

);

Cont\_ajuste\_inst: entity work.Cont

port map (

clk => clka,

enablem => plus5,

enableh => plus6,

clear => clear,

op => op,

hdl => n0, --entradas

hul => n1,

mdl => n2,

mul => n3,

n0 => hdca, --saidas

n1 => huca,

n2 => mdca,

n3 => muca

);

Alarme\_inst: entity work.Ajuste\_alarme

port map (

clk => clka,

enablem => plus2,

enableh => plus1,

clear => clear,

hdl => hda, --entradas

hul => hua,

mdl => mda,

mul => mua,

n0 => n0a, --saidas

n1 => n1a,

n2 => n2a,

n3 => n3a

);

Def\_inst: entity work.Defasador

port map(

clk=>clk,

clk1=>a,

clk2=>b,

clk3=>c,

spulso=>spulso

);

FF1\_inst: entity work.Flip\_Flop

port map(

clk => b,

d => i1,---entrada

q => is1---saida

);

DISP\_inst: entity work.DISP

port map (

q0 => q0,

q1 => q1,

q2 => q2,

q3 => q3,

seg => segl,

an => anl,

clk => clk

);

process(clk)

begin

if(clk'event and clk='1') then

count <= count +1;

count2 <= count2 +1;

count3 <= count3 +1;

count4 <= count4 +1;

count5 <= count5 +1;

count6 <= count6 +1;

if(count = (n/2)\*60) then -- 1 minuto

clkmin <= not clkmin;

count <=1;

end if;

if(count2 = n/16) then

clka <= not clka;

count2 <= 1;

end if;

if(count6 = n/2) then -- 1 segundo

clks <= not clks;

count6 <= 1;

end if;

if(count3=(n/2)/264) then

count3<= 1;

do <= not(do);

end if;

if(count4=(n/2)/495) then

count4<= 1;

si <= not(si);

end if;

if(count5 = (n/10000)) then

clk2 <=not clk2;

count5 <=1;

end if;

-----checando a opção selecionada nos switchs

if (op = "10") then --ajuste relogio

plus5 <= not(plusm);

plus6 <= not(plush);

plus3<='0';

plus4<='0';

q0 <= conv\_std\_logic\_vector(hdca,4);

q1 <= conv\_std\_logic\_vector(huca,4);

q2 <= conv\_std\_logic\_vector(mdca,4);

q3 <= conv\_std\_logic\_vector(muca,4);

n0 <= conv\_integer(unsigned(q0));

n1 <= conv\_integer(unsigned(q1));

n2 <= conv\_integer(unsigned(q2));

n3 <= conv\_integer(unsigned(q3));

elsif (op="01") then --ajuste alarme

plus2 <= not(plusm);

plus1 <= not(plush);

plus3 <= '1';

------salva o valor do display

q0 <= conv\_std\_logic\_vector(n0a,4);

q1 <= conv\_std\_logic\_vector(n1a,4);

q2 <= conv\_std\_logic\_vector(n2a,4);

q3 <= conv\_std\_logic\_vector(n3a,4);

hda <= conv\_integer(unsigned(q0));

hua <= conv\_integer(unsigned(q1));

mda <= conv\_integer(unsigned(q2));

mua <= conv\_integer(unsigned(q3));

------continua contando em background

n0 <= hdc;

n1 <= huc;

n2 <= mdc;

n3 <= muc;

else

plus3 <= '1';

plus4 <= '0';

q0 <= conv\_std\_logic\_vector(hdc,4);

q1 <= conv\_std\_logic\_vector(huc,4);

q2 <= conv\_std\_logic\_vector(mdc,4);

q3 <= conv\_std\_logic\_vector(muc,4);

n0 <= conv\_integer(unsigned(q0));

n1 <= conv\_integer(unsigned(q1));

n2 <= conv\_integer(unsigned(q2));

n3 <= conv\_integer(unsigned(q3));

if(alarme\_sw='1') then

if(n0a = hdc and n1a=huc and n2a=mdc and n3a=muc and is1/=i2) then

-- leds <= '0';

alarme <= '1';

end if;

end if;

end if;

if(alarme\_sw='1') then

leda1<='0';

leda2<='0';

leda3<='0';

else

-- leds <= '1';

alarme <= '0';

leda1<='1';

leda2<='1';

leda3<='1';

end if;

if(parar='0' and is1/=i2)then

i2<=is1;

alarme <= '0';

end if;

if(top='0')then

buzzer<=buzzer1;

else

buzzer<=buzzer2;

end if;

end if; ---if do clk

end process;

-------------------------------------------------------------------------

process(do)

begin

if(alarme='1')then

if(top='0')then

if (do='1') then

buzzer1 <= '0';

else

buzzer1 <= '1';

end if;

end if;

else

buzzer1 <= '1';

end if;

end process;

process(si)

begin

if(alarme='1')then

if(top='1')then

if (si='1') then

buzzer2 <= '0';

else

buzzer2 <= '1';

end if;

end if;

else

buzzer2 <= '1';

end if;

end process;

process(clk2)

begin

if(brilho='0')then

if(clk2='1')then

seg<=segl;

an<=anl;

else

seg<="1111111";

an<="1111";

end if;

else

seg<=segl;

an<=anl;

end if;

end process;

process(clka)

begin

if(alarme='1')then

if(clka='1')then

leds1<='1';

leds2<='0';

else

leds1<='0';

leds2<='1';

end if;

else

leds1<='1';

leds2<='1';

end if;

led1<=leds1;

led2<=leds2;

end process;

process(clkmin)

begin

if(clkmin'event and clkmin='1')then

if(op="00" or op="11")then

if(i1=i2)then

i1<=not(i1);

end if;

end if;

end if;

end process;

end Behavioral;

**COMPONENTES:**

**Defasador.vhd:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Defasador is

port (

clk1, clk2, clk3, spulso, clkout: out std\_logic;

clk: in std\_logic

);

end Defasador;

architecture Behavioral of Defasador is

signal ea: std\_logic\_vector(1 downto 0) := "00";

signal counter: integer:=1;

begin

process(clk)

begin

if (clk'event and clk='1') then

if(ea="00") then

clk1 <= '1';

clk2 <= '0';

clk3 <= '0';

spulso <= '0';

ea <= "01";

end if;

if(ea="01") then

clk1 <= '0';

clk2 <= '1';

clk3 <= '0';

spulso <= '0';

ea <= "10";

end if;

if(ea="10") then

clk1 <= '0';

clk2 <= '0';

clk3 <= '1';

spulso <= '0';

ea <= "11";

end if;

if(ea="11") then

clk1 <= '0';

clk2 <= '0';

clk3 <= '0';

spulso <= '1';

ea <= "00";

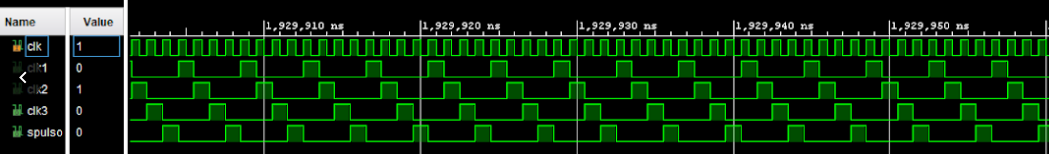
end if;

end if;

end process;

end Behavioral;

**Simulação:**



**Flip\_Flop.vhd:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Flip\_Flop is

port (

q: out std\_logic;

clk, d: in std\_logic

);

end Flip\_Flop;

architecture Behavioral of Flip\_Flop is

begin

process(clk)

begin

if(clk'event and clk='1') then

q <= d;

end if;

end process;

end Behavioral;

**Simulação:**



**Cont.vhd:**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity cont is

port (

clk : in std\_logic;

enablem: in std\_logic;

enableh: in std\_logic;

clear: in std\_logic;

op: in std\_logic\_vector(1 downto 0);

hdl,hul,mdl,mul: in integer range 0 to 9:=0;

n0,n1,n2,n3: out integer range 0 to 9:=0

);

end cont;

architecture Behavioral of cont is

begin

process(clk)

begin

if(clk'event and clk='1') then

if(op="10")then

if(enablem='1')then

if (mdl=5 and mul=9) then

n2 <= 0;

n3 <= 0;

else

if (mul<9) then

n3 <= mul+1;

else

n3 <= 0;

if (mdl<5) then

n2 <= mdl+1;

else

n2 <= 0;

end if;

end if;

end if;

end if;

if(enableh='1')then

if (hdl=2 and hul=3) then

n0 <= 0;

n1 <= 0;

else

if (hul<9 and hdl < 2) then

n1 <= hul+1;

elsif (hdl = 2 and hul<3) then

n1 <= hul+1;

else

n1 <= 0;

if (hdl<2) then

n0 <= hdl+1;

else

n0 <= 0;

end if;

end if;

end if;

end if;

else

if (enablem='1') then

if (hdl=2 and hul=3 and mdl=5 and mul=9) then

n0 <= 0;

n1 <= 0;

n2 <= 0;

n3 <= 0;

else

if (mul<9) then

n3 <= mul+1;

else

n3 <= 0;

if (mdl<5) then

n2 <= mdl+1;

else

n2 <= 0;

if (hul<9 and hdl < 2) then

n1 <= hul+1;

elsif (hdl = 2 and hul<3) then

n1 <= hul+1;

else

n1 <= 0;

if (hdl<2) then

n0 <= hdl+1;

else

n0 <= 0;

end if;

end if;

end if;

end if;

end if;

end if;

end if;

end if;

if(enablem='0' and enableh='0')then

n0 <= hdl;

n1 <= hul;

n2 <= mdl;

n3 <= mul;

end if;

if(clear='0')then

n0 <= 0;

n1 <= 0;

n2 <= 0;

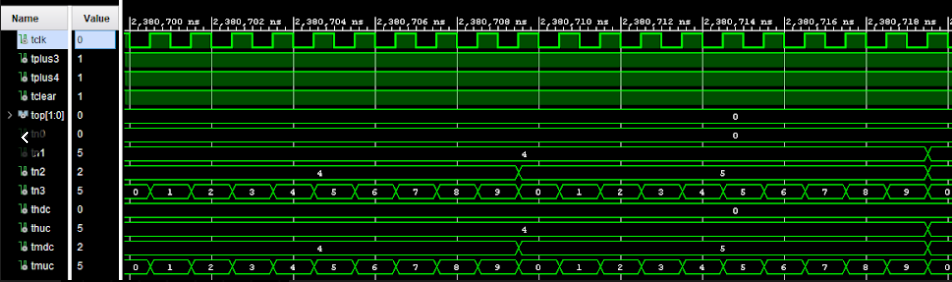
n3 <= 0;

end if;

end process;

end Behavioral;

**Simulação:**



**DISP.vhd:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity DISP is

Port ( q0: in STD\_LOGIC\_VECTOR (3 downto 0);

q1: in STD\_LOGIC\_VECTOR (3 downto 0);

q2: in STD\_LOGIC\_VECTOR (3 downto 0);

q3: in STD\_LOGIC\_VECTOR (3 downto 0);

seg: out STD\_LOGIC\_VECTOR (6 downto 0);

an: out STD\_LOGIC\_VECTOR (3 downto 0);

clk: in STD\_LOGIC);

end DISP;

architecture Behavioral of DISP is

signal e: STD\_LOGIC\_VECTOR (1 downto 0);

begin

Demux\_inst: entity work.Demux

port map (

e => e,

q0 => q0,

q1 => q1,

q2 => q2,

q3 => q3,

seg => seg

);

Decod\_inst: entity work.Decod

port map (

an => an,

e => e

);

Cont\_inst: entity work.Contador

port map (

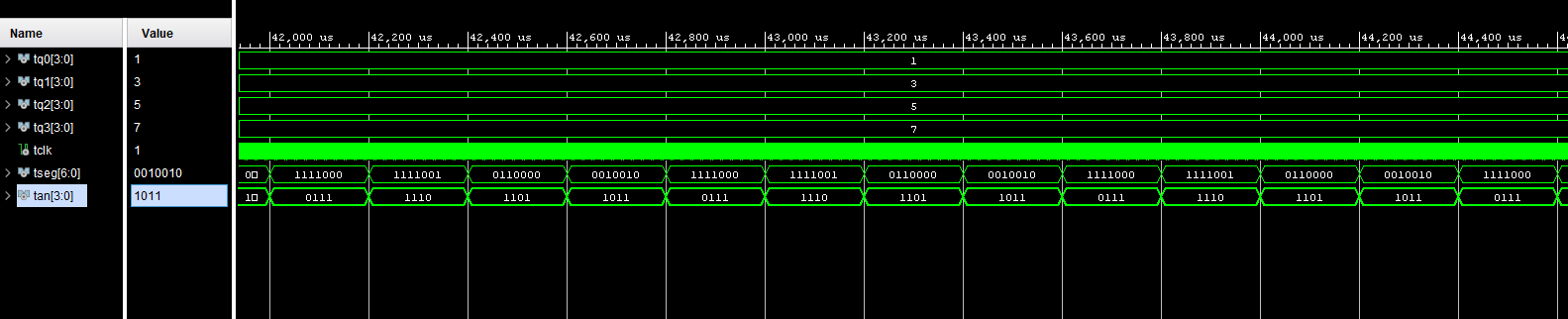
clk => clk,

e => e

);

end Behavioral;

**Simulação:**

****

**Contador.vhd:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Contador is

Port (clk: in std\_logic;

e: out std\_logic\_vector(1 downto 0):="00"

);

end Contador;

architecture Behavioral of Contador is

signal ee: std\_logic\_vector(1 downto 0):="00";

signal num: integer range 0 to 99999:=0;

signal clk2: std\_logic:='0';

begin

process(clk)

begin

if (clk'event and clk='1') then

num <= num-1;

if (num <= 0) then

num <= 99999;

clk2 <= not(clk2);

end if;

end if;

end process;

process(clk2)

begin

if (clk2'event and clk2='1') then

case ee is

when "00" => ee <= "01";

when "01" => ee <= "10";

when "10" => ee <= "11";

when "11" => ee <= "00";

when others => ee <= "00";

end case;

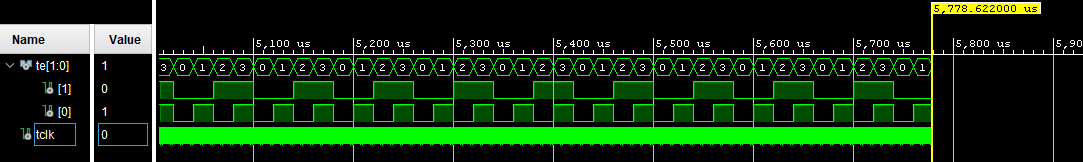
end if;

end process;

e <= ee;

end Behavioral;

**Simulação:**

****

**Demux.vhd:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Demux is

Port (q0: in STD\_LOGIC\_VECTOR (3 downto 0);

q1: in STD\_LOGIC\_VECTOR (3 downto 0);

q2: in STD\_LOGIC\_VECTOR (3 downto 0);

q3: in STD\_LOGIC\_VECTOR (3 downto 0);

e: in STD\_LOGIC\_VECTOR (1 downto 0);

seg: out STD\_LOGIC\_VECTOR (6 downto 0)

);

end Demux;

architecture Behavioral of Demux is

signal ent: STD\_LOGIC\_VECTOR (3 downto 0);

begin

with e select

ent <= q0 when "00",

q1 when "01",

q2 when "10",

q3 when "11",

"0000" when others;

with ent select

seg <= "1000000" when "0000",

"1111001" when "0001",

"0100100" when "0010",

"0110000" when "0011",

"0011001" when "0100",

"0010010" when "0101",

"0000010" when "0110",

"1111000" when "0111",

"0000000" when "1000",

"0010000" when "1001",

"0001000" when "1010",

"0000011" when "1011",

"1000110" when "1100",

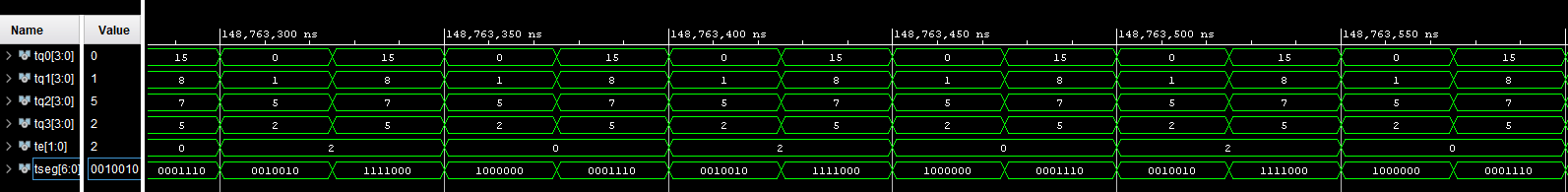
"0100001" when "1101",

"0000110" when "1110",

"0001110" when others;

end Behavioral;

**Simulação:**

****

**Decod.vhd:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Decod is

Port (e: in STD\_LOGIC\_VECTOR (1 downto 0);

an: out STD\_LOGIC\_VECTOR (3 downto 0) );

end Decod;

architecture Behavioral of Decod is

begin

with e select

an <= "1110" when "00",

"1101" when "01",

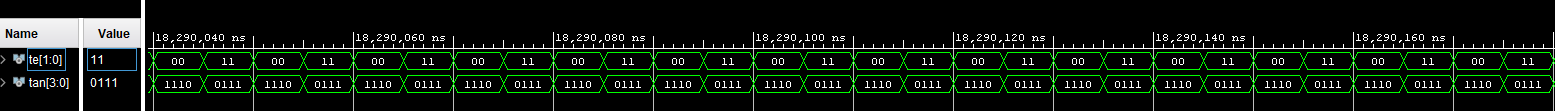
"1011" when "10",

"0111" when "11",

"1111" when others;

end Behavioral;

**Simulação:**

****

**Lista de materiais utilizados:**

* Display;
* Placa ALTERA Cyclone IV FPGA Board A-C4E6.

Dados da placa:

* 6272 elementos lógicos (LE);
* 392 blocos de matriz lógica (LAB);
* 200MHz de frequência operacional máxima;
* 270Kbits de memória;
* 2 PLLs;
* 91 I/O.

Recursos disponíveis da placa:

* Entrada para teclado PS2;
* Comunicação serial RS232:
* Saída de vídeo VGA;
* Display de 7 segmentos com 4 dígitos multiplexados;
* Buzzer piezo;
* 4 dip-switch;
* 4 botões;
* 4 LEDs;
* Memória SDRAM de 64Mbits;
* Cristal oscilador de 50MHz;
* JTAG;
* Sensor de temperatura LM75A (I2C/2-Wire);
* EEPROM de 1024x8 - AT24C08 (I2C/2-Wire);
* Receptor infravermelho (acompanha controle remoto);
* Conexão para display LCD do tipo 16x2 ou 20x4 com ajuste de contraste incorporado;

**Listagem de softwares:**

* Quartus Prime 18.0
* Vivado

**Funcionamento:**

Quando ligado, a contagem do relógio inicia em 00:00 e serão exibidos em quatro displays de 7 segmentos os valores das horas e dos minutos.

Ele possui 3 modos de operação definidos pela posição dos switchs 1 e 2 da placa, um modo de operação normal (contagem do tempo e exibição), ajuste do relógio e ajuste do alarme.

Durante o ajuste do relógio, o fpga para a contagem atual e espera o usuário finalizar a alteração para poder continuar contando. Durante o ajuste do alarme a contagem continua normalmente em segundo plano.

O switch 8 é responsável por habilitar ou não o alarme. O switch 7 pelos toques do alarme e o switch 6 pelo brilho do display.

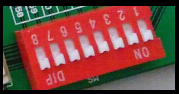
O botão K2 é utilizado para incrementar os minutos e o K3 as horas. O botão K4 desliga o alarme e o botão K5 reseta o relógio e o alarme.

**Manual de operação:**

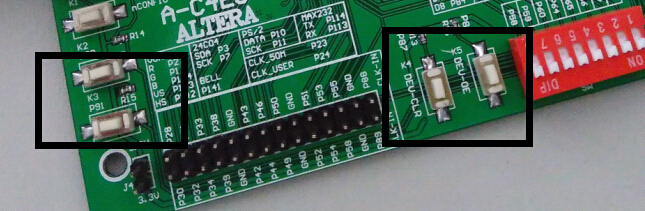
No swtich 1 e 2 são selecionados os modos de operação:

* “11” e “00” para operação normal;
* “10” para ajuste do tempo;
* “01” para ajuste do alarme.

Obs: chave pra cima representa o estado ‘1’, e para baixo ‘0’.



Os ajustes são realizados utilizando os botões K2, para os minutos, e K3, para as horas.



Para habilitar o alarme é necessário deixar o switch 8 em estado ‘0’, e em ‘1’ para desabilitar. Quando habilitado, os leds D12, D13 e D14 ficarão acessos.

Quando o alarme for ativado, os leds D5 e D9 acenderão alternadamente e o buzzer tocará. Para desativar é necessário pressionar o botão K4.

Existem dois toques disponíveis selecionados pelo switch 7. Em ‘1’ o som escolhido é dó, e em ‘0’ é si.

O switch 6 ajusta o brilho. Em ‘1’ ele estará em modo noturno, e em ‘0’ modo diurno.

O botão K5 reseta os contadores.

**Conclusão**

Um ponto que pode ser destacado no desenvolvimento deste projeto foi a componentização e testes individuais de cada componente, em VHDL, já que é possível isolar erros que em num programa completo não seriam facilmente detectados. Além de que a componentização também reduz o tamanho do programa.