Disciplina: PCS 3335 – Laboratório Digital A	
Prof.: *Glauber de Bona*	**Data:** *23/07/20*
Turma: *04*	**Bancada:** *B8*
Membros:	
9912532 - Rafael Lucchesi Piacente	
10772925 - Gustavo Donnini Chen	

Prova 2

Controle de Prioridade

1. Introdução

Esta experiência tem como principal objetivo desenvolver um circuito digital que aciona relógios com frequências diferentes conforme a prioridade dos eventos do ambiente.

2. Objetivo

Ao fim desta experiência, os seguintes tópicos devem ser conhecidos pelos alunos:

- Desenvolver interfaces de um circuito digital com o meio externo.

3. Planejamento

a. Projeto

i. Descrição funcional:

Nosso circuito recebe **n** entradas binárias e reconhece, dentre elas, a entrada de maior prioridade. Um bloco **Codificador** identifica a entrada mais importante e gera um código, que é recebido por um bloco **Seletor**. O bloco **Relógios** gera clocks nas frequências de 20, 200, 2000 e 20000Hz. A frequência é selecionada pelo bloco Seletor conforme o **Código** gerado pelo bloco **Codificador**.

Em nosso circuito, os dígitos mais significativos representam os alarmes de maior frequência e tem maior prioridade, sendo o de maior frequência e maior prioridade o bit a3.

ii. Diagrama de Blocos:

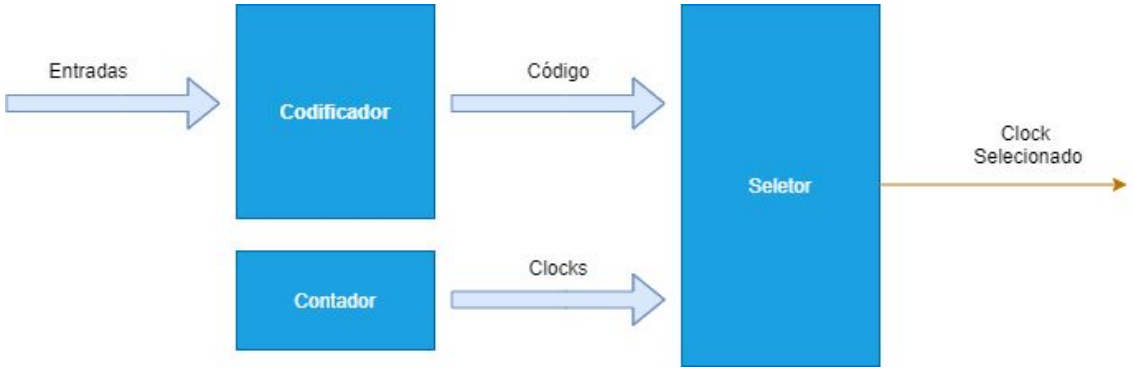


Figura 1: Diagrama de blocos do circuito.

iii. Diagrama Lógico

Tendo em mente a prioridade que desejamos atribuir para as entradas, desenvolvemos uma tabela verdade para projetar

o bloco codificador a partir de seu mapa de karnaugh.

Entradas				Saídas	
a3	a2	a1	a0	y1	y0
0	0	0	0	x	x
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	1	1
1	0	0	1	1	1
1	0	1	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

Karnaught Y1:				
a3a2/a1a0	00	01	11	10
00	x	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1
Exp.	Y1 = a3 + a2			

Karnaught Y0:				
a3a2/a1a0	00	01	11	10
00	x	0	1	1
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1
Exp.	Y1 = a3 + a3'*a2'*a1			

Figura 2: Mapa de Karnaugh do bloco Codificador.

O resultado do bloco codificador obtido através do mapa de Karnaugh foi o seguinte:

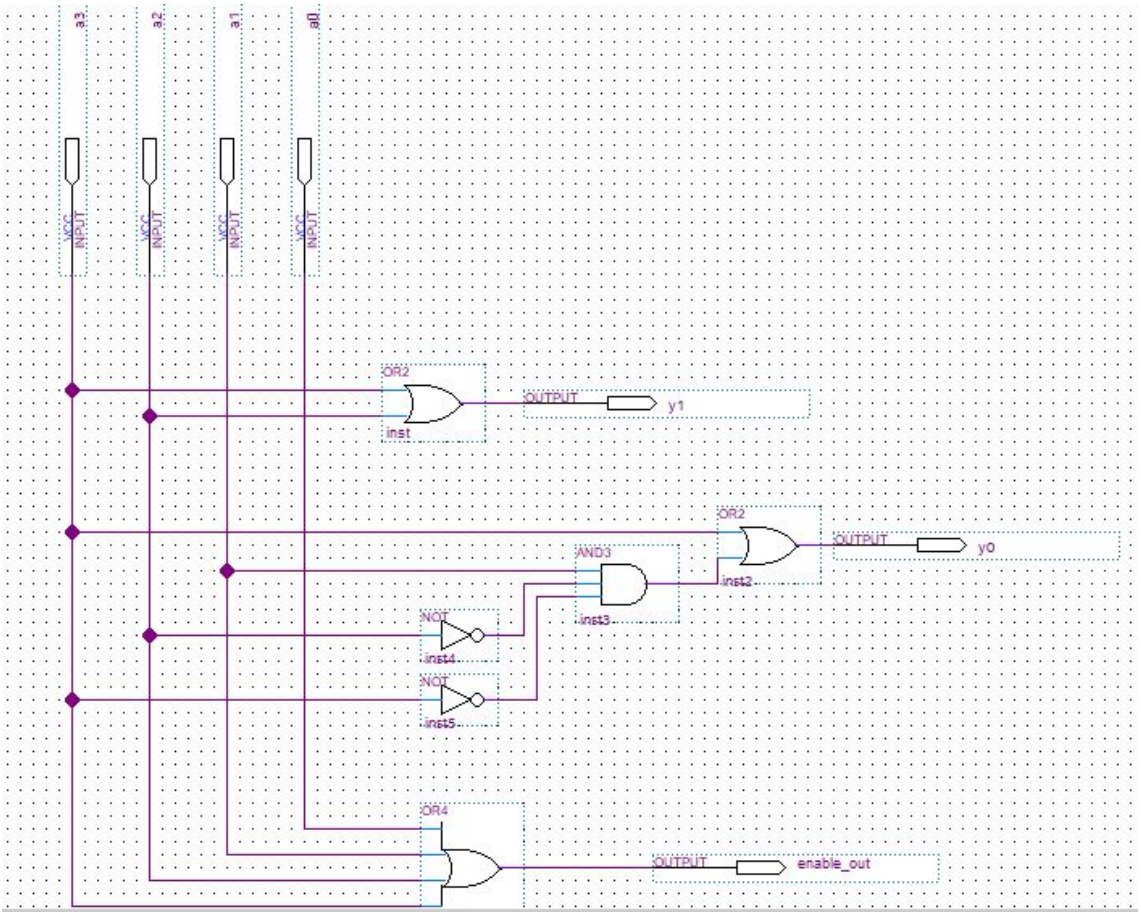


Figura 3: Montagem do bloco codificador.

Por causa do funcionamento que adotamos para o circuito seletor, tivemos que implementar a saída enable_out, que é '1' quando o circuito recebe qualquer entrada e '0' caso contrário.

Para nosso circuito temporizador, desenvolvemos o bloco através do seguinte código em VHDL:

contador.vhd

```
use IEEE.std_logic_1164.all;

entity contador is
    port (
        clock : in std_logic;
        out20, out200, out2k, out20k : out std_logic
    );
end contador;

architecture comportamental of contador is
--    signal cont20: integer range 1 to 2500000;
--    signal cont200: integer range 1 to 250000;
--    signal cont2k: integer range 1 to 25000;
--    signal cont20k: integer range 1 to 2500;       --valores originais

    signal cont20: integer range 1 to 1000;
    signal cont200: integer range 1 to 100;
    signal cont2k: integer range 1 to 10;
    signal cont20k: integer range 1 to 2;

    signal aux20: std_logic;
    signal aux200: std_logic;
    signal aux2k: std_logic;
    signal aux20k: std_logic;

begin

    process (clock)
    begin

        if clock'event and clock = '1' then
            if cont20 = 1000 then --original: 2500000
                cont20 <= 1;
            else cont20 <= cont20 + 1;
                end if;

                if cont200 = 100 then --original: 250000
                    cont200 <= 1;
            else cont200 <= cont200 + 1;
                end if;

                if cont2k = 10 then --original: 25000
                    cont2k <= 1;
            else cont2k <= cont2k + 1;
                end if;

                if cont20k = 2 then --original: 2500
                    cont20k <= 1;
            else cont20k <= cont20k + 1;
                end if;
          end if;

          if cont20 < 501 then aux20 <= '0'; else aux20 <= '1'; end if; --original: 1250001
          if cont200 < 51 then aux200 <= '0'; else aux200 <= '1'; end if; --original: 125001
          if cont2k < 6 then aux2k <= '0'; else aux2k <= '1'; end if; --original: 12501
          --if cont20k < 1 then aux20k <= '0'; else aux20k <= '1'; end if; --original
          if cont20k = 2 then aux20k <= '1'; else aux20k <= '0'; end if;
    end process;

    out20 <= aux20;
    out200 <= aux200;
    out2k <= aux2k;
    out20k <= aux20k;

end comportamental;
```

Figura 4: código VHDL do circuito contador.

Dado o clock de 50MHz da FPGA, fazer os clocks de 20,

200, 2000 e 20000Hz foi relativamente simples. Ao contar até 50.000.000 obtemos 1Hz. Para aumentar a frequência apenas dividimos essa contagem pelo clock desejado, aí basta contar até este número.

Tendo esse bloco montado, partimos para a elaboração do bloco seletor:

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all;
3
4   entity seletor is
5    port(
6
7        A,B,C,D : in STD_LOGIC;
8        S0,S1,ENABLE : in STD_LOGIC;
9        Z: out STD_LOGIC
10   );
11  end seletor;
12
13  architecture bhv of seletor is
14  begin
15
16  process (A,B,C,D,S0,S1) is
17  begin
18
19    if ENABLE = '0' then Z <= '0'; end if;
20
21    if (S0 ='0' and S1 = '0' and ENABLE = '1') then
22        Z <= A;
23    elsif (S0 ='1' and S1 = '0' and ENABLE = '1') then
24        Z <= B;
25    elsif (S0 ='0' and S1 = '1' and ENABLE = '1') then
26        Z <= C;
27    elsif (ENABLE = '1') then
28        Z <= D;
29    end if;
30
31  end process;
32  end bhv;
```

Figura 5: código VHDL do bloco Seletor.

Nosso bloco seletor nada mais é do que um MUX 4x2 com uma entrada enable. Inicialmente, tentamos usar o MUX padrão oferecido pelo Quartus, mas este não possuía a entrada enable. Por esse motivo, acabamos elaborando nosso próprio MUX.

Com isso apenas precisamos integrar o circuito da seguinte forma:

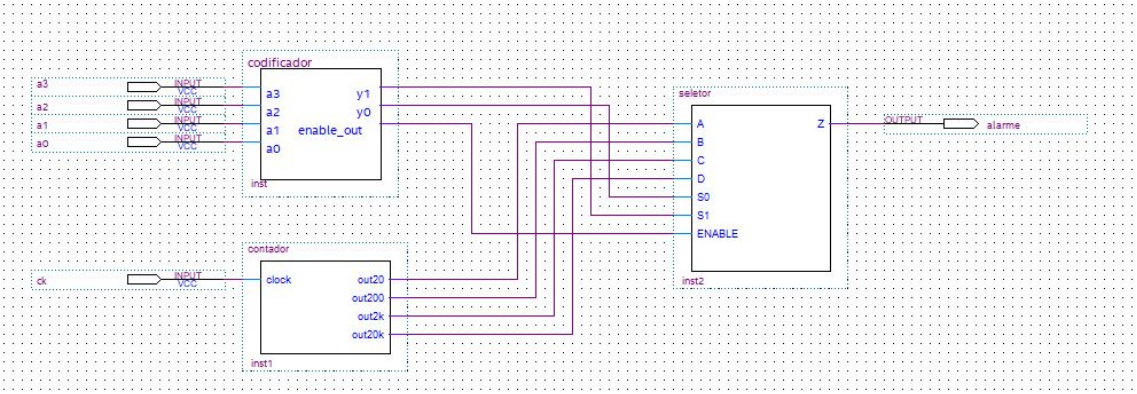


Figura 6: circuito completo após a montagem.

Nesta montagem, o circuito codificador estabelece a prioridade das entradas e gera uma palavra de 2 bits em sua saída, que representa a freqüência do sinal que deve ser utilizado pelo alarme. Esta palavra entra nas portas seletoras do MUX, que efetivamente escolhe a freqüência do sinal.

- Vale notar que a palavra '00' no bloco seletor significa que o alarme de 20Hz está ativo, e que o bloco codificador gera em sua saída o código '00' tanto quando a entrada é '0001' quanto '0000'. Por esse motivo, utilizamos a porta enable_out do bloco codificador para informar o bloco seletor se algum alarme deve ser acionado ou não.

iv. Simulações:

Inicialmente, simulamos os blocos separadamente:

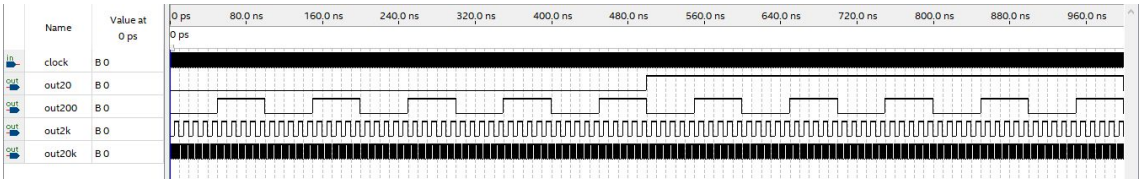


Figura 7: simulação do bloco Contador com o tick escalonado para possibilitar a simulação.

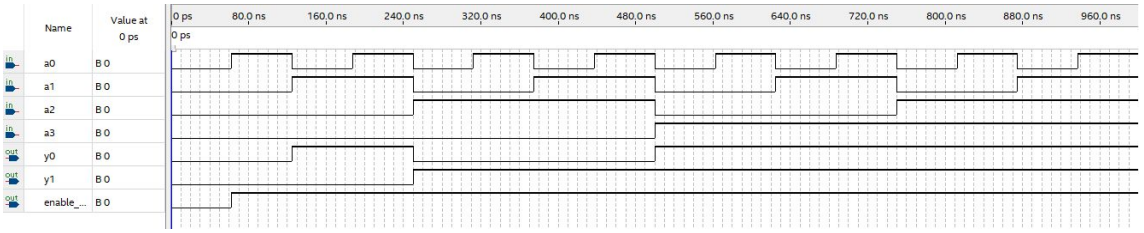


Figura 8: simulação do bloco Codificador.

Para o circuito completo, as entradas dos bits mais significativos (a3) selecionam os alarmes de maior frequência e

possuem maior prioridade:

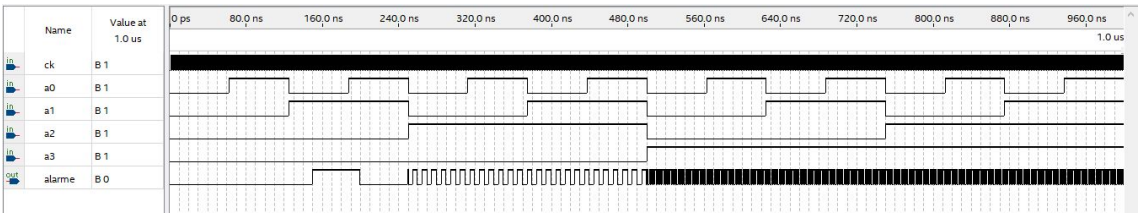


Figura 9: simulação do circuito completo.

Em seguida fizemos uma simulação em que a entrada a0 foi acionada individualmente, pois a frequência do alarme correspondente a esta entrada é muito baixa, impossibilitando vê-la numa simulação com outros alarmes de frequências mais altas.

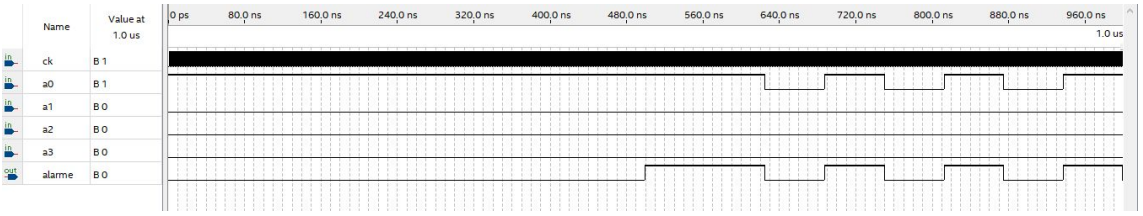


Figura 10: Simulação individual da entrada a0 no circuito completo.

É possível ver que o clock gerado pelo circuito contador continua gerado clocks independentemente das entradas estarem ativas ou não. Como se trata de um alarme, isto não interfere no funcionamento do circuito pois não precisamos que estes clocks estejam sincronizados com nenhum outro circuito.

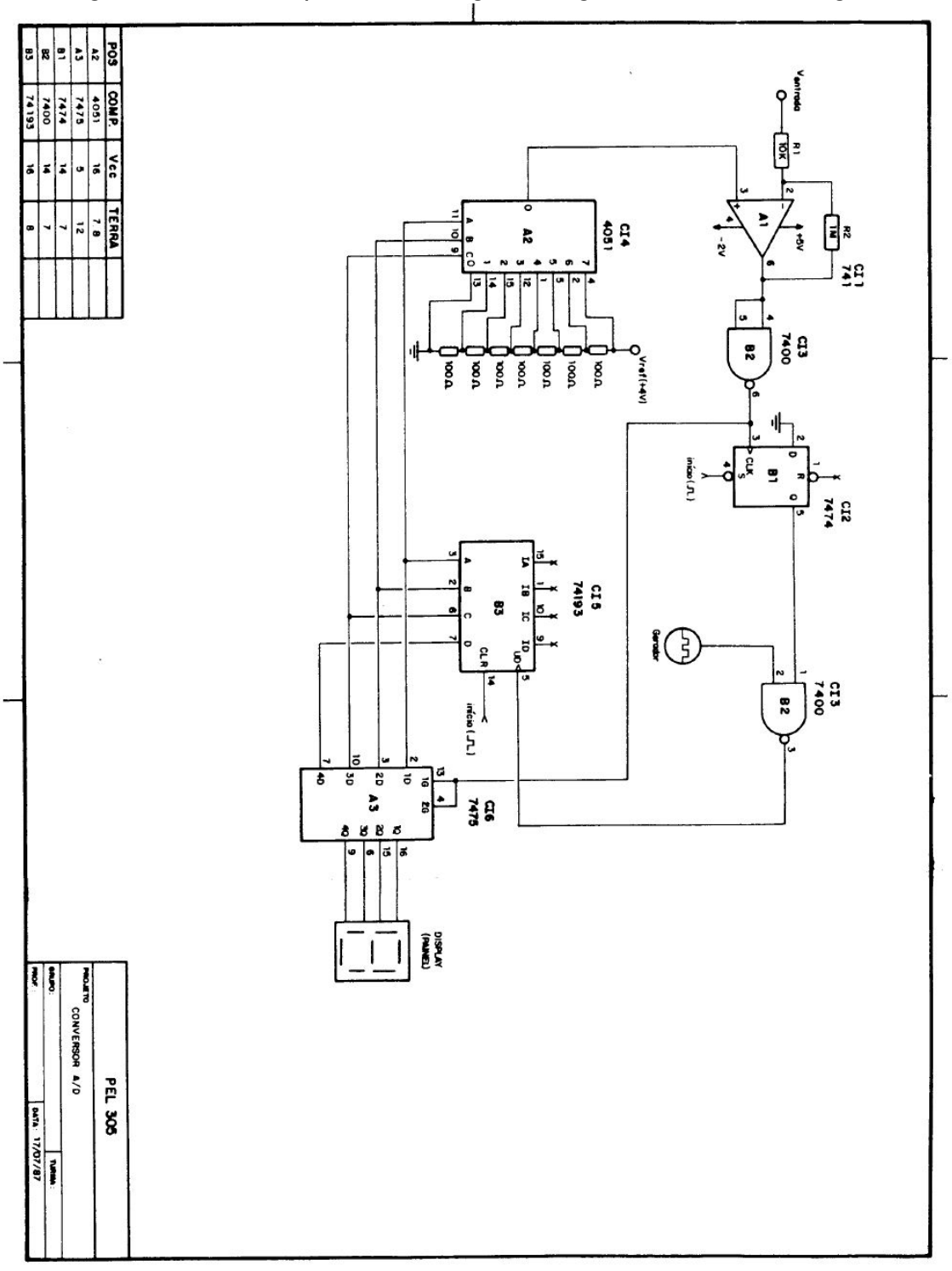
a. Estratégia de montagem, testes e depuração

Para realizar a montagem deste circuito, testaremos cada parte individualmente e checamos se os resultados saem como os da simulação. Se todos funcionarem corretamente, partiremos para integrar o circuito e mais uma vez comparando com os resultados esperados.

Apêndices

A – Formato padrão de diagrama lógico de um circuito digital

Figura 8 – Formato padrão de diagrama lógico de um circuito digital.



Referências

1. Apostilas e documentos de apoio do site ~/labdig do PCS.
2. Apostilas disponíveis na plataforma e-Disciplinas.
3. Apostilas do Laboratório de Sistemas Digitais.
4. Texas Instruments. TTL Logic Data Book, 1994.

5. WAKERLY, John F. Digital Design Principles & Practices. 4th edition, Prentice Hall, 2006.