UNIVERSIDADE FEDERAL DE PERNAMBUCO
Departamento de Eletrônica e Sistemas

Interface de rádio digital

Projeto 3

Isaac Neves Farias

Matheus José Araújo Oliveira

Pedro Henrique de Albuquerque Gomes

Recife, 30 de agosto de 2021

Sumário

1 Introdução

O objetivo deste projeto é a aplicação prática dos conceitos vistos em aula, e nesse caso o desenvolvimento de sistemas em VHDL. Tendo isso em mente, o projeto consiste em implementar o simulador de interface de um rádio digital (AM/FM), com a funcionalidade de salvar as rádios preferidas.

O rádio digital deve funcionar na estação AM e FM e com frequências de 540 a 1600 kHz e 87.5 a 108 MHz, respectivamente. O rádio deve possuir um modo de operação normal e um modo de gravação (para gravar as estações favoritas).

Operando no modo normal deve ser possível selecionar qual o tipo de rádio, se a rádio pesquisa no modo crescente ou decrescente ou se a pesquisa deve ser pausada, ainda nesse modo é possível selecionar uma das estações favoritas as quais devem ter sido gravadas anteriormente.

No modo de gravação, os botões são responsáveis apenas por gravar a estação que está presente no LCD, se a memória de gravação estiver cheia então ela é subscrita.

A interface do rádio digital foi desenvolvida no software Quartus e executada na placa Cyclone IV.

Sobre a estrutura deste relatório, a seção 2, desenvolvimento, aprofunda nos passos tomados para a interface rádio digital funcionar nas especificações solicitadas. A seção é dividida em três etapas após uma visão geral do problema, em que abordam: sistema contador, controle e exibição. Uma divisão esquemática pode ser observada na figura 01.

Além disso, na seção 3, manual de operação, é explicado a interface do usuário do rádio digital implementado na placa Cyclone IV. Resultados & discussão, seção 4, traz a avaliação do produto final e discute sobre as dificuldades encontradas pela equipe durante o desenvolvimento do projeto. Por fim, na seção 5, conclusão, pensamentos finais são expressos sobre o projeto como um todo.

2 Desenvolvimento

A solução desenvolvida foi dividida em 3 partes:

1. Sistema contador

A partir do clock da placa, é utilizado um sistema redutor de frequência para se ajustar com a contagem do relógio. O clock tratado é enviado para contadores (período de 0.5 segundos) .

Após a aplicação do divisor de frequência é implementado o contador de fato, o qual possui condicionais de contagem. Contagem crescente ou decrescente, com limites superior e inferior e intervalo de passagem diferente.

2. Controle e debounce

Primeiramente, no módulo controle é implementado o debounce o qual é a parte responsável por tratar o sinal de entrada dos botões protegendo contra os efeitos de oscilação.

Após isso, é feita a interpretação dos sinais de entrada, relacionando quando os dois botões são pressionados ao mesmo tempo ou a função de um botão em determinado modo.

3. Exibição - LCD

Para a exibição no display é recebido as variáveis de interesse que determinam o que é mostrado, a seguir é implementado uma máquina de estados as quais são responsáveis pela configuração de cada caractere a ser escrito, por último é realizada a escrita de fato.

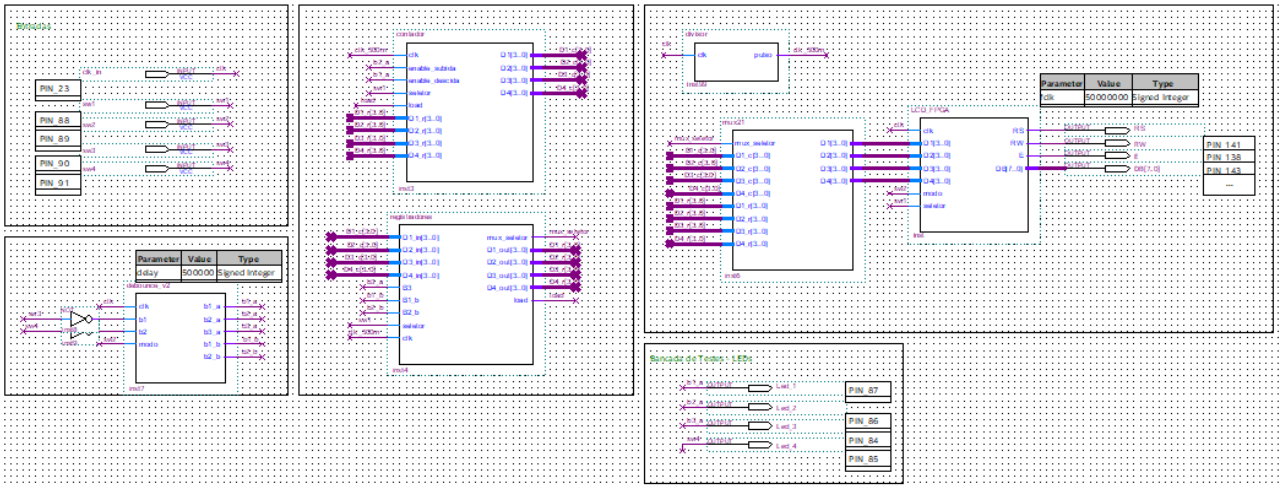

Figura 01: Projeto completo dividido nas partes do desenvolvimento.

2.1 Sistema contador

2.1.1 Redução da frequência do *clock*

Para o sistema redutor de frequência, o que é aplicado de fato é apenas um contador interno que quando alcança determinado número muda a variável de saída (funcionando também como um clock), isso ocorre de acordo com o necessário. Como o clock inicial é de 50 MHz, então ao dividir o clock por 12.500.000 reduzimos a frequência para 4 Hz, como a mudança de estado depende do clock então a frequência real obtida divide por 2 sendo 2 Hz, ou seja, com um intervalo de 0.5 segundos.
Todo o processo pode ser observado na Figura 02.

```
16  process(clk) -- Divisor do sinal de clock da placa
17  begin
18      if rising_edge(clk) then
19          count<=count+1;
20          if count=(12500000) then
21              count<=0;
22              auxiliar<= not auxiliar;
23          end if;
24      end if;
25  end process;
```

Figura 02: Divisor de clock.

2.1.2 Contador crescente e decrescente

O bloco contador (Figura 03) armazena os dados de contagem dos contadores AM e FM, tem como entrada o clock de 0.5s, os enables de subida e descida da contagem, o seletor AM/FM, os valores D1~D4 do registrador e o load que informa quando utilizar os valores recebidos do registrador, têm como saída os valores dos dígitos D1, D2, D3 e D4 de contagem, de acordo com o seletor.

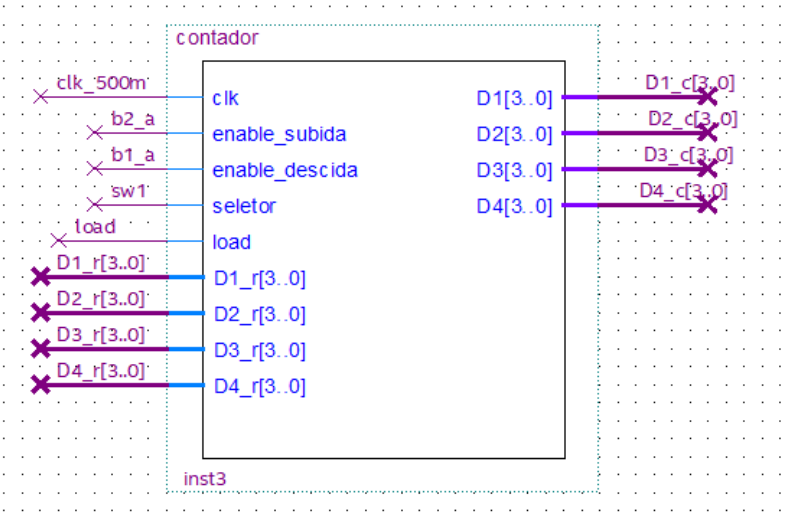

Figura 03: Bloco contador

A parte de contagem do projeto se comporta de modos diferentes dependendo de algumas variáveis visto que a contagem funciona de forma crescente ou decrescente e que no

modo AM a contagem é feita de 10 em 10 KHz, já no modo FM é feita de 5 em 5 MHz, como pode ser observado nas Figuras 04 e 05.

Foram utilizados dois contadores distintos para evitar possíveis problemas de overflow já que o range de trabalho AM/FM é diferente.

.

```
if enable_subida='1' and limite_subida=0 then
   if seletor='0' then --subida AM (de 10 em 10)
         if dezenas_am=9 then
            dezenas_am:=0;
            if centenas_am=9 then
               centenas_am:=0;
               milhares_am:=milhares_am+1;
            else
               centenas_am:=centenas_am+1;
            end if;
         else
            dezenas_am:=dezenas_am+1;
         end if;
   else --subida FM (de 5 em 5)
      if unidades_fm=5 then
         unidades_fm:=0;
         if dezenas_fm=9 then
            dezenas_fm:=0;
            if centenas_fm=9 then
               centenas_fm:=0;
               milhares_fm:=milhares_fm+1;
            else
               centenas_fm:=centenas_fm+1;
            end if;
         else
            dezenas_fm:=dezenas_fm+1;
         end if;
      else
      unidades_fm:=5;
      end if;
   end if;
```

Figura 04: Contador crescente AM ou FM

```
elsif enable_descida='1' and limite_descida=0 then
   if seletor='0' then --descida AM (de 10 em 10)
         if dezenas_am=0 then
            dezenas_am:=9;
            if centenas_am=0 then
               centenas_am:=9;
               milhares_am:=milhares_am-1;
            else
               centenas_am:=centenas_am-1;
            end if;
         else
            dezenas_am:=dezenas_am-1;
         end if;
   else --descida FM (de 5 em 5)
      if unidades_fm=0 then
         unidades_fm:=5;
         if dezenas_fm=0 then
            dezenas_fm:=9;
            if centenas_fm=0 then
               centenas_fm:=9;
               milhares_fm:=milhares_fm-1;
            else
               centenas_fm:=centenas_fm-1;
            end if;
         else
            dezenas_fm:=dezenas_fm-1;
         end if;
      else
      unidades_fm:=0; --5 ou 0
      end if;
   end if;
end if;
```

Figura 05: Contador decrescente AM ou FM

Lembrando ainda, que para cada modo há um limite de contagem diferente, no modo AM o intervalo é de 540 a 1600 KHz, já no FM vai de 875 a 1800 MHz, o que pode também ser observado na Figura 06.

```
if seletor = '0' then --AM

   if (unidades=0 and dezenas=0 and centenas=6 and milhares=1) then

   limite_subida<=1; --1600

   elsif (unidades=0 and dezenas=4 and centenas=5 and milhares=0) then

   limite_descida<=1; --540
   else
   limite_subida<=0;
   limite_descida<=0;
   end if;

else  --FM

   if (unidades=0 and dezenas=8 and centenas=0 and milhares=1) then
      limite_subida<=1; --1080
   elsif (unidades=5 and dezenas=7 and centenas=8 and milhares=0) then
      limite_descida<=1; --875
   else
      limite_subida<=0;
      limite_descida<=0;
   end if;
end if;
```

Figura 06: Limites de contagem para AM e FM.

O load informa o momento que deve-se utilizar os valores recebidos do registrador, pode ser observado na Figura 07 a seguir.

```
-------------------------------------------------------------
----------------------seleção dos salvos--------------------
   if load='1' then
      if seletor='0' then --AM

         unidades_am:=conv_integer(unsigned(D1_r));
         dezenas_am:=conv_integer(unsigned(D2_r));
         centenas_am:=conv_integer(unsigned(D3_r));
         milhares_am:=conv_integer(unsigned(D4_r));

      else
         unidades_am:= conv_integer(unsigned(D1_r));
         dezenas_fm:=conv_integer(unsigned(D2_r));
         centenas_fm:=conv_integer(unsigned(D3_r));
         milhares_fm:=conv_integer(unsigned(D4_r));

      end if;
   end if;
```

Figura 07: Load do registrador.

Como foi trabalhado com inteiros dentro do bloco, precisa-se converter de vetor binário para inteiro na entrada do load e na saída dos valores precisa-se converter de inteiro para vetor binário, isso pode ser visualizado na Figura 08 a seguir.

```
-------------------------------------------------------------
----------------------seletor de saida ----------------------

      if seletor='0' then --AM

         D4 <= conv_std_logic_vector (unidades_am, 4);
         D3 <= conv_std_logic_vector (dezenas_am, 4);
         D2 <= conv_std_logic_vector (centenas_am, 4);
         D1 <= conv_std_logic_vector (milhares_am, 4);
      else
         D4 <= conv_std_logic_vector (unidades_fm, 4);
         D3 <= conv_std_logic_vector (dezenas_fm, 4);
         D2 <= conv_std_logic_vector (centenas_fm, 4);
         D1 <= conv_std_logic_vector (milhares_fm, 4);
      end if;
```

Figura 08: Conversão dos números inteiros para binário.

2.1.3 Registrador

Esse módulo, apesar de simples, é muito importante, pois ele é responsável por fazer o registro das 2 estações favoritas, tanto AM quanto FM. Desse modo os dígitos se tornam de fácil acesso para manipulação e apresentação no LCD. Figura 09.

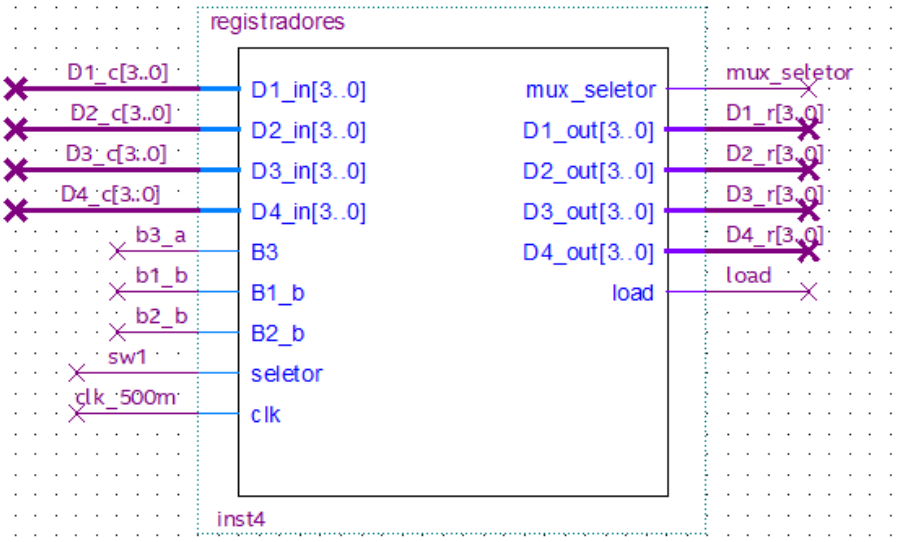

Figura 09: Registrador

O módulo tem como entrada os valores D1~D4 do bloco contador, o seletor AM/FM e 3 entradas importantes advindas do debounce_v2. As entradas b1_b e b2_b estão diretamente relacionadas aos *switchs* 1 e 2, que é utilizado para registrar as estações favoritas de acordo com o modo, consultar Figura 10. Enquanto isso, a entrada b3_a está relacionada ao aperto dos dois botões e será útil no momento de visualizar os registos, consultar Figura 11.

```
if seletor='0' then --AM

   if B1_b='1' then
      reg11_am:=D1_in;
      reg12_am:=D2_in;
      reg13_am:=D3_in;
      reg14_am:=D4_in;
   elsif B2_b='1' then
      reg21_am:=D1_in;
      reg22_am:=D2_in;
      reg23_am:=D3_in;
      reg24_am:=D4_in;
   end if;

else

   if B1_b='1' then
      reg11_fm:=D1_in;
      reg12_fm:=D2_in;
      reg13_fm:=D3_in;
      reg14_fm:=D4_in;
   elsif B2_b='1' then
      reg21_fm:=D1_in;
      reg22_fm:=D2_in;
      reg23_fm:=D3_in;
      reg24_fm:=D4_in;
   end if;

end if;
```

Figura 10: Rotina de registro de acordo com o seletor AM/FM

É importante ressaltar o uso do clock de 0.5s para cálculo de 2s após apertar os 2 botões para visualização dos registros.

```
if B3='1' then
      count:=count+1;
      if count>=4 then      --contador de 2s
         mux_seletor<='1';
         if seletor='0' then --AM
            if count>=6 then --1segundo aqui
               D1_out<=reg21_am;
               D2_out<=reg22_am;
               D3_out<=reg23_am;
               D4_out<=reg24_am;
            else  --1segundo aqui
               D1_out<=reg11_am;
               D2_out<=reg12_am;
               D3_out<=reg13_am;
               D4_out<=reg14_am;
            end if;
         else
            if count>=6 then --1segundo aqui
               D1_out<=reg21_fm;
               D2_out<=reg22_fm;
               D3_out<=reg23_fm;
               D4_out<=reg24_fm;
            else  --1segundo aqui
               D1_out<=reg11_fm;
               D2_out<=reg12_fm;
               D3_out<=reg13_fm;
               D4_out<=reg14_fm;
            end if;
         end if;
      else
         mux_seletor<='0';
      end if;
```

Figura 11: Lógica de visualização dos registros

2.2 Controle e debounce

As entradas de controle, Figura 12, são:

A switch 1 é a chave seletora do modo AM/FM, será utilizado a denominação ‘seletor’ em outros blocos quando menciona-la.

A switch 2 é a chave seletora do modo Rádio/Gravação, será utilizado a denominação ‘modo’ em outros blocos quando menciona-la.

A switch 3 e 4 serão utilizados como botões e serão responsáveis de forma indireta, já que passam por um tratamento no debounce (Figura 13), para subir e descer a frequência do relógio, serão mencionados em alguns blocos como sw3 e sw4.

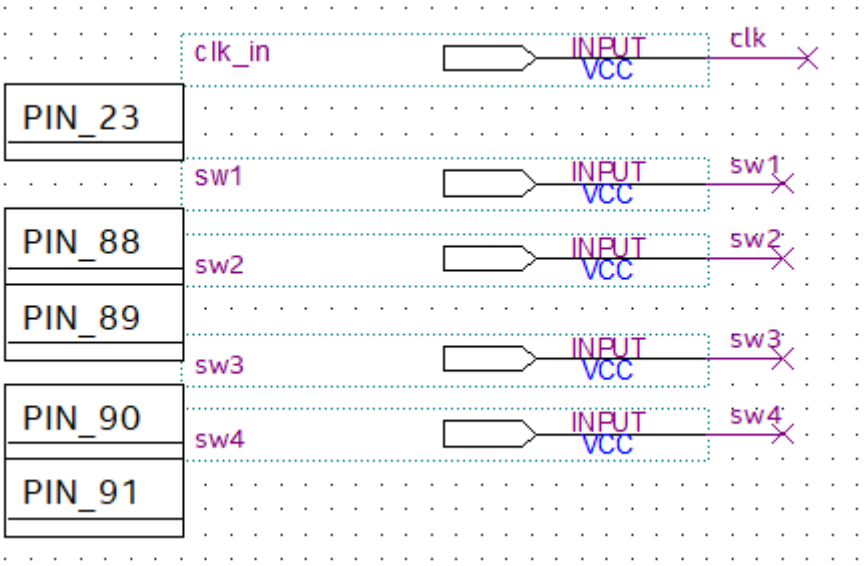

Figura 12: Entradas

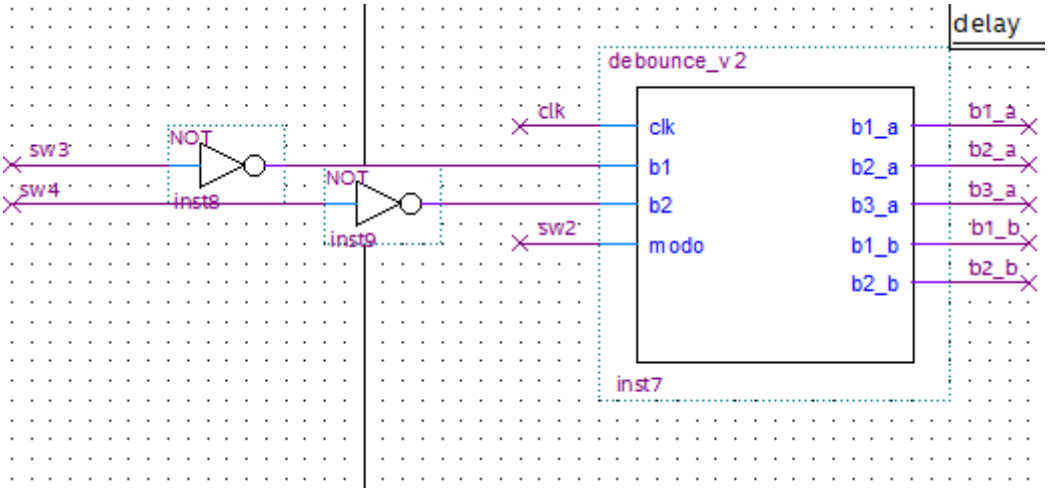

Figura 13: Debounce_v2

O debounce tem como função básica garantir o aperto limpo dos botões, retirando todas as oscilações que ocorrem imediatamente após o clique do botão, consultar Figura 14. Além disso, é realizado um tratamento para chegarmos às saídas de comportamento desejado por outros blocos e pelas especificações do projeto.

```
aux1 <= ffd1(0) xor ffd1(1); -- 1 se entre dois pulsos de clock a entra for diferente
aux2 <= ffd2(0) xor ffd2(1); -- detecta edges (completando o comentário de cima)

if rising_edge(clk) then --inicio1
   ffd1(0) <= b1;
   ffd1(1) <= ffd1(0); -- a entrada do botão de pulsos consecutivos

   ffd2(0) <= b2;
   ffd2(1) <= ffd2(0);

   if (aux1='1') then -- detectou borda(edge) antes do tempo
      cont1<=0;
      cont2<=0;
   elsif (cont1<delay) then -- sinal continuo por esse tempo
      cont1 <= cont1 + 1;
   else
      b1_f <= ffd1(1);

   end if; --fim2

   if (aux2='1') then -- detectou borda(edge) antes do tempo
      cont2<=0;
      cont1<=0;
   elsif (cont2<delay) then -- sinal continuo por esse tempo
      cont2 <= cont2 + 1;
   else
      b2_f <= ffd2(1);

   end if; --fim3
```

Figura 14: Rotina de debounce

As saídas b1_a e b2_a funcionam como uma chave que alterna sua saída para cada clique da sw3 e sw4 de entrada, consultar Figura 15.

A saída b3_a é equivalente a um terceiro botão que pode ser ativado ao apertar os dois botões de entrada ao mesmo tempo.

```
if (b1_f='1') and (b2_f='1') then --inicio4
    --enquato os dois botoes tiverem apertados
      b1_r<='0';
      b2_r<='0';
      b3_r<='1';

elsif (b1_f='1' and flag1=0) then

      b3_r<='0';
      b1_r<= not b1_r;
      flag1:=1;

elsif (b2_f='1' and flag2=0) then --inicio12

      b3_r<='0';
      b2_r<= not b2_r;
      flag2:=1;

else

   flag1:=0;
   flag2:=0;
   b3_r<='0';

end if;--fim4
```

Figura 15: Rotina de tratamento dos botões

Como os dois botões apresentam funções diferentes de acordo com o modo selecionado (radio/grav.) foi utilizado um case funcionando como um mux para definir o

botão de entrada na saída certa. Deste modo, a saída b1_b e b2_b só é definida de acordo com a entrada quando o rádio entra no modo gravação, consultar Figura 16.

```
--mux modo seleciona qual saída
      case modo is
         when '1' =>
            b1_a<=b1_r;
            b2_a<=b2_r;
            b3_a<=b3_r;
            b1_b<='0';
            b2_b<='0';
         when '0' =>
            b1_a<='0';
            b2_a<='0';
            b3_a<='0';
            b1_b<=b1_r;
            b2_b<=b2_r;
         when others =>
            b1_a<='0';
            b2_a<='0';
            b3_a<='0';
            b1_b<='0';
            b2_b<='0';
      end case;
```

Figura 16: Mux de saída

2.3 Exibição - LCD

O módulo de exibição, observada na Figura 17, consiste na aplicação do display LCD, alterando os caracteres dependendo da contagem e de acordo com o modo e seletor AM/FM.

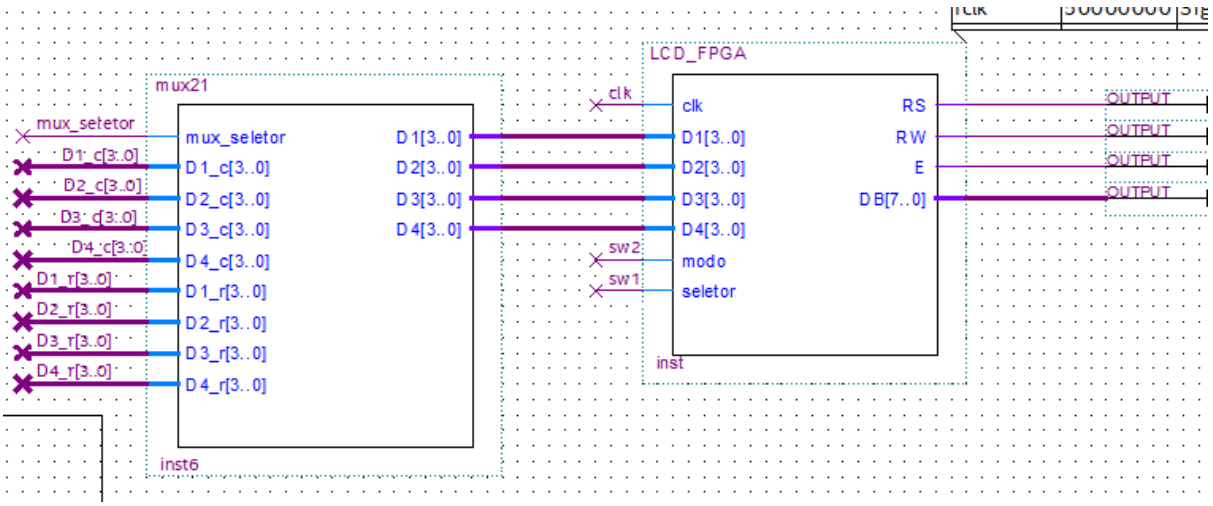

Figura 17 : Módulo LCD conectado ao mux

Os dados que são apresentados no LCD são pré-analisados no submódulo mux, identificando se os dígitos que devem aparecer são os da contagem ou os das estações que estão salvas no módulo registrador. O código do mux pode ser observado na Figura 18..

```
entity mux21 is
   port (
   mux_seletor : in std_logic;
   D1_c, D2_c, D3_c, D4_c, D1_r, D2_r, D3_r, D4_r: in std_logic_vector (3 downto 0);
   D1, D2, D3, D4 : out std_logic_vector (3 downto 0)
   );
end mux21;

architecture escolha of mux21 is
--signal aux1, aux2, aux3, aux4: std_logic;
begin

   process (mux_seletor,D1_c, D2_c, D3_c, D4_c, D1_r, D2_r, D3_r, D4_r)
   --variable count: integer range 0 to 5;
   begin
      if mux_seletor='1' then
         D1<=D1_r;
         D2<=D2_r;
         D3<=D3_r;
         D4<=D4_r;

      else

         D1<=D1_c;
         D2<=D2_c;
         D3<=D3_c;
         D4<=D4_c;

      end if;
```

Figura 18: Código do submódulo mux.

O mux seletor é controlado pelo bloco dos registrados que por sua vez recebe a informação do aperto dos dois botões e faz as contagens de tempo específicas para exibição dos registros.

A implementação do LCD é feita por meio de uma grande máquina de estados, onde os estados em que a entrada “RS” é igual a zero são utilizados para configuração do display e quando “RS” é igual a 1 é feita a escrita de fato no LCD, seguindo a tabela ASCII.

Importante lembrar também que o módulo possui um divisor de clock interno para 500 Hz, apenas por questão de tempo de comunicação entre os dispositivos.

As Figuras 19 e 20 ilustram o que foi descrito acima.

```
when   FunctionSet28   =>
RS<=  '0'; RW<=  '0';
DB   <=   "00111000";
nx_state <= ClearDisplay;

when ClearDisplay =>
RS<= '0'; RW<= '0';
DB <= "00000001";
nx_state   <=   DisplayControl;

when   DisplayControl   =>
RS<= '0';   RW<=  '0';
DB   <=  "00001100";
nx_state <= EntryMode;

when EntryMode =>
RS<= '0'; RW<= '0';
DB <= "00000110";
nx_state   <=  SetAddress;


when SetAddress =>
RS<=   '0';   RW<=   '0';
DB   <=  "10000000";   --COMANDO PARA POSICIONAR O CURSOR NA LINHA 0 COLUNA 5
nx_state  <= WriteData1;
```

Figura 19: Máquinas de estado no estágio de configuração.

```
when  WriteData1 =>
RS<=   '1';   RW <='0';
DB   <=   X"4d";                --'M
nx_state <= WriteData2;

when WriteData2 =>
RS<= '1'; RW<= '0';
DB <= X"4f";                    --'O'
nx_state <= WriteData3;

when WriteData3 =>
RS<= '1'; RW<= '0';
DB <= X"44";                    --'D'
nx_state  <= WriteData4;

when  WriteData4   =>
RS<=   '1';   RW<=   '0';
DB   <=  X"4f";                 --'O'
nx_state  <= WriteData5;

when  WriteData5   =>
RS<=   '1';   RW<=   '0';
DB   <=  X"20";                 --'_'
nx_state  <= WriteData6;
```

Figura 20: Máquinas de estado no estágio de escrita.

No estágio de escrita as informações enviadas para as portas de comunicação variam de acordo com as chaves seletoras, que pode ser visualizado na Figura 21 abaixo.

```
when  WriteData12   =>
RS<=   '1';   RW<=   '0';
if seletor='1' then
DB   <=  X"46";                 --'F'
else
DB   <=  X"41";                 --'A'
end if;
nx_state  <= WriteData13;

when  WriteData13   =>
RS<=   '1';   RW<=   '0';

DB   <=  X"4d";                 --'M'

nx_state  <= SetAddress1;
```

Figura 21: Modificações nas saídas de acordo com as entradas

3 Manual de operação

Nesta seção consta o manual de operação do rádio digital, a Figura 22 abaixo apresenta uma legenda que determina o local de cada botão, chave e display citados nos próximos parágrafos.

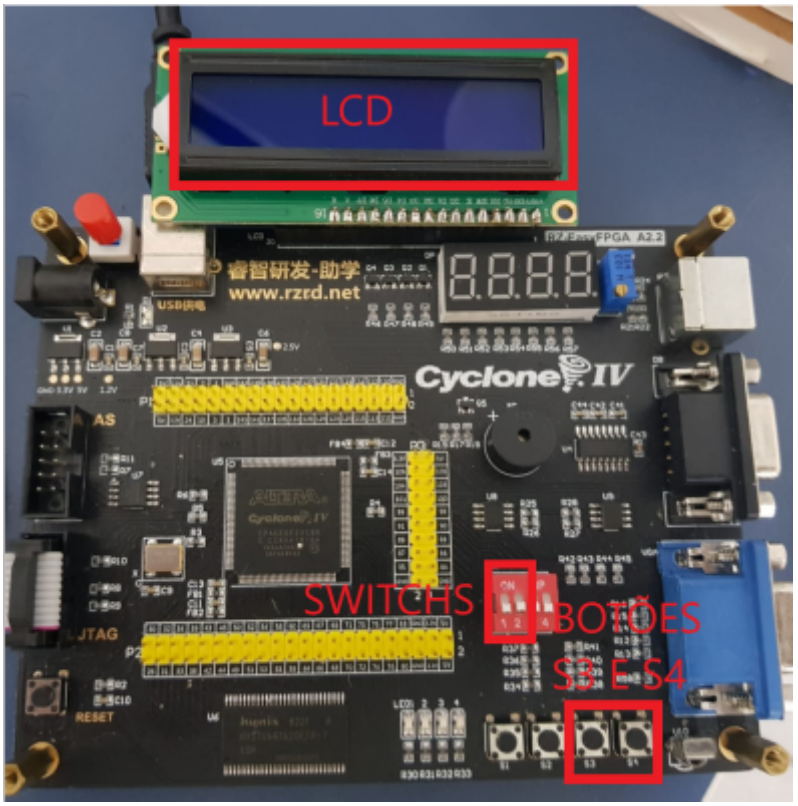

Figura 22: Esquemático Cyclone IV

.O rádio digital possui dois modos de funcionamento, o modo normal da rádio e o modo de gravação, a escolha do modo de operação é determinada pela switch 2, nível alto para modo normal e nível baixo para modo de gravação.

A partir disso, é possível explicar a rotina de funcionamento do dispositivo em dois modos.

Modo normal:

- O rádio pode funcionar em AM (540 a 1600 kHz) e FM (87.5 a 108 MHz). A alternância entre os modelos é feita pelo switch 1, nível alto para FM e baixo para AM .
- Ao clique do botão S4 a estação começa a subir ao paço de 10 KHz (AM) ou 0.5 MHz (FM), isso acontece no intervalo real de 0.5s em 0.5s, se o botão é pressionado pela segunda vez a contagem é pausada.
- Ao pressionar o botão S3 o rádio começa a passar as estações no modo decrescente, sendo apertado pela segunda vez em seguida a rádio também é pausada.

- Se os dois botões são pressionados ao mesmo tempo durante 2 segundos, são exibidas sequencialmente as estações gravadas (AM ou FM), quando o botão para de ser pressionado a estação que estava no LCD é selecionada.

Modo de gravação:

- Existem dois espaços de memória de gravação para AM e dois para FM , o botão S3 grava na memória 1 e o S4 na memória 2.
- Se a memória já estiver com outra gravação então ela é subscrita.

Rotina simples de operação: Ligar o rádio, escolher as estações preferidas utilizando os botões para subir e descer as estações, entrar no modo gravação, gravar a estação na memória que desejar, sair do modo de gravação e utilizar os dois botões posteriormente para voltar a estação salva.

4 Resultados & Discussão

4.1 Resultados

De modo geral os requisitos para o funcionamento do projeto foram correspondidos, entretanto há erros sistemáticos no processo de debounce, o que muitas vezes acaba danificando o funcionamento como o todo visto que o sistema depende dos comandos do usuário para funcionar de acordo.

Para melhor compreensão do sistema a visualização RLT de todo o projeto pode ser observada na Figura 23.

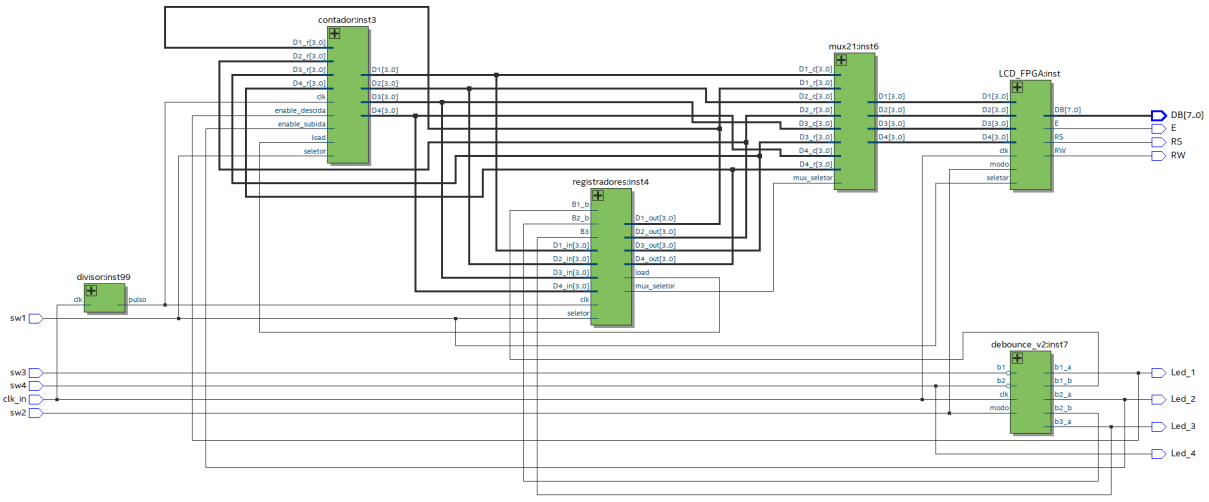

Figura 23: Visualização RLT do projeto

4.2 Principais problemas encontrados

1. Escrever os digitos no LCD: utilizou-se o ‘&’ para concatenar strings de bits e conseguir escrever os valores de D1~D4 no LCD;

2. Incompatibilidade das entradas de 4bits do registrador no contador com os inteiros usados para contar: foi feita uma transformação de inteiro para std_logic_vector utilizando a função cov_integer;

3. Implementação do debounce: o debounce foi implementado da mesma forma do projeto anterior, utilizando a lógica de dois flipflops com atraso de um clock para garantir estabilidade e um contador de delay da saída, porém o funcionamento ficou fora do previsto e a saída antes de soltar o botão varia entre 0 e 1 rapidamente, que foi constatado posteriormente pela utilização do LED, mas algumas funcionalidades do bloco ficaram dentro do esperado garantindo um funcionamento precário do projeto, infelizmente;
4. Load dos registradores no contador: o Load é realizado de forma esperada, porém após o load os contadores têm seu funcionamento prejudicado e o projeto tem um comportamento um pouco fora do previsto.

5 Conclusão

O resultado final apresentado foi satisfatório, visto o que é pedido pelo experimento. Por meio dessa prática foi possível desenvolver e otimizar o conceito de gerenciamento de projetos, assim como uma experiência em equipe.

Além disso, vale destacar que, devido à crescente complexidade dos projetos dos circuitos de eletrônica digital, fica evidente a importância de desenvolver habilidades em *HDLs*, sendo *VHDL* uma das linguagens mais utilizadas na área. A prática do presente projeto permitiu aos alunos a exploração e desenvolvimento de tais habilidades.

Por fim, ressaltando a importância do rádio digital é possível pensar que por grande parte da história como o maior meio de comunicação da humanidade, revolucionando o conceito de mídia e conectando todo o mundo.