Universidade Federal do Rio Grande do Norte

ELE1717 - SISTEMAS DIGITAIS

Fila (FIFO)

Projeto e Documentação de uma FILA

Discente:

Kaike Castro Carvalho

Docente:
Samaherni Morais Dias

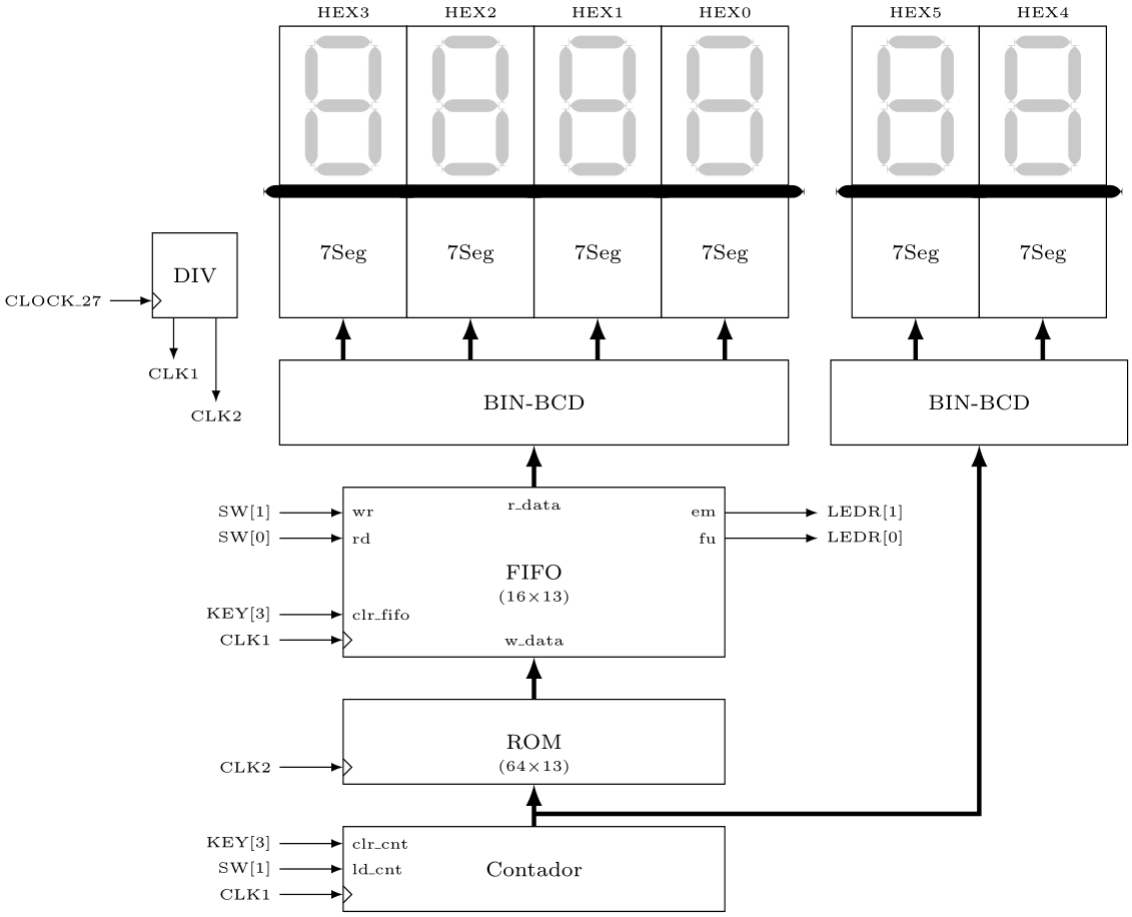
17 de março de 2020

1 Introdução

Esse documento descreve e documenta o projeto de uma fila para a disciplina de Sistemas Digitais (ELE1717). O trabalho foi implementado em VHDL, simulado com o auxílio do Quartus II e também executado no kit DE2 FPGA Cyclone II.

Uma fila é uma lista que se escreve no fim, mas é lida no início e que a leitura remove o item lido da do topo da lista. Também conhecida como sendo do tipo o primeiro que é o primeiro que sai - first-input fisrt-output (FIFO).

Figura 1: Visão geral



O projeto foi implementado usando o metódo de projeto em Nível de Transferência entre Registradores (RTL) o qual se divide em BLOCO DE CONTROLE e o BLOCO OPERACIONAL em que a combinação desses blocos é chamada de processaor.

Foi utilizado duas memória a ROM de (64 x 13) e uma RAM de (16 x 13) para que o processador faça as operações básicas com mais eficiência.

2 Objetivo

Desenvolver uma fila de 16 palavras de 13 bits cada e que a esrita dos valores na fila é por meio de uma mémoria ROM de 64 palavras de 13 bits cada.

3 Materiais e Métodos

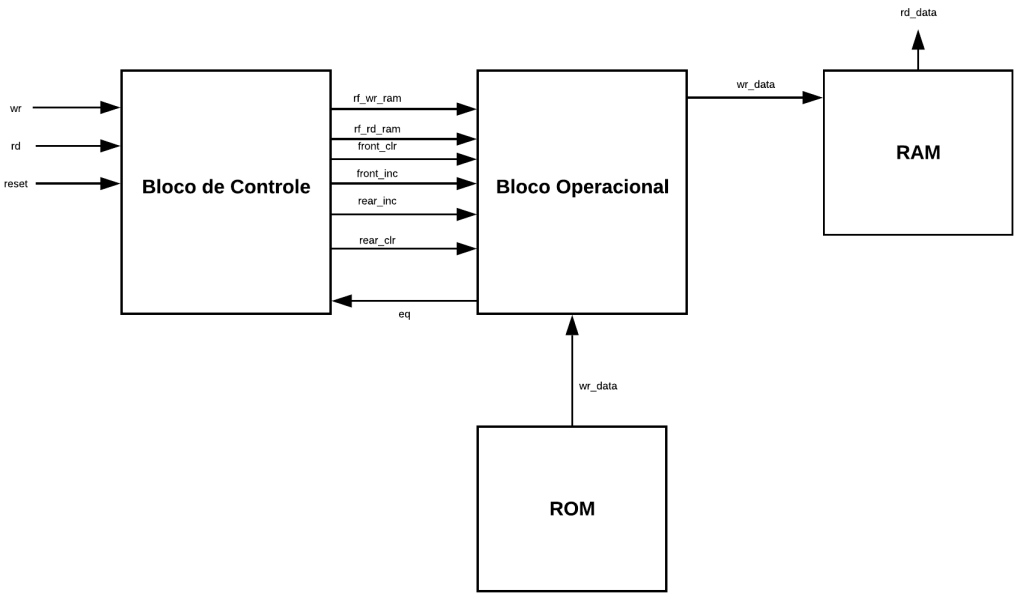
A construção do projeto foi baseado no modelo RTL e com isso seguiu cinco passos para a elaboração da solução final. A seguir estão listados os itens que descreve o projeto RTL o qual foi seguido passo a passo.

1. Obtenha uma máquina de estados de alto nível
2. Crie um bloco operacional
3. Conecte o bloco operacional a um bloco de controle
4. Obetenha a FSM do bloco de controle

O desenvolvimento da solução do problema foi baseado em blocos que facilita revisar os erros e concatenar as operações até formar uma extrutura maior.

Na Figura 2 exemplifica que pode se tratado a solução em blocos.

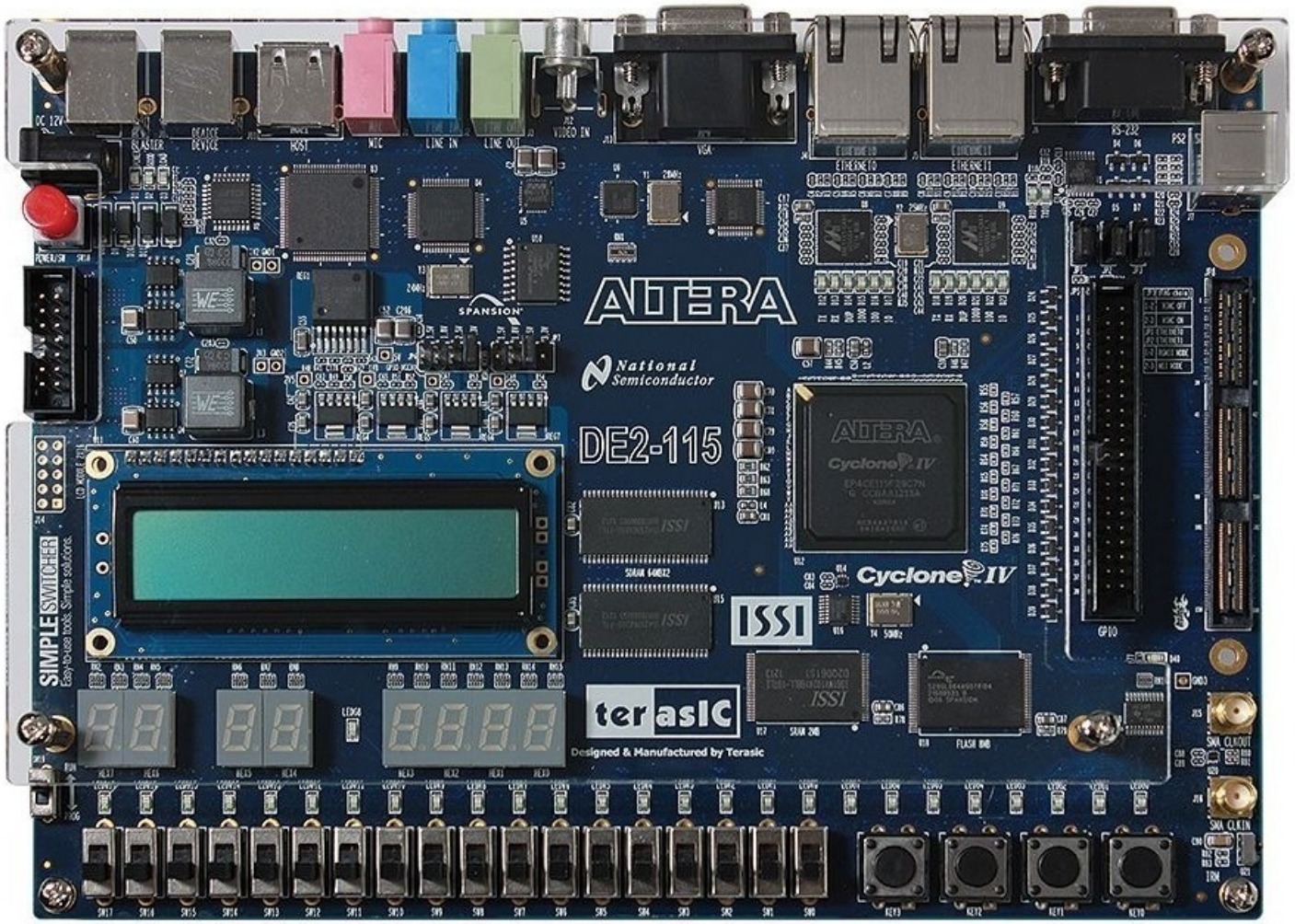
Figura 2: Diagrama de blocos



No passo número 1 foi criado o bloco de controle em que operará sinais boleanos e para isso a implementação da Máquina de Estados Finitos do inglês - Finite State Machine (FSM) que procura capcturar o comportamento desejado do sistema. No segundo passo, o bloco operacional visa trabalhar com dados onde são inseridos entradas com vários bits a serem operadas de acordo com o bloco de controle.

Após a simulação o código em vhdl é passado para o software Quartus II em que há um ambiente de simulação e configuração para o circuito integrado Field Programmable Gate Array (FPGA) Ciclone II, a Figura 3 abaixo exibe o embarcado do FPGA.

Figura 3: Kit DE2



3.1 Bloco de Controle

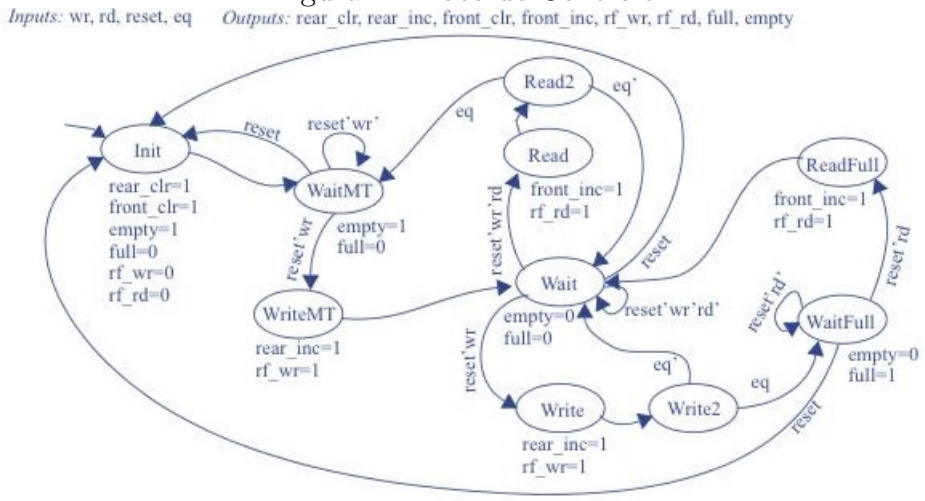
Esse processador tem predominância no trabalho de sinais, pois é no controlador que ocorre toda o gerenciamento dos estados.O bloco de controle possui dez estados e com isso garante que não seja lida uma fila vazia ou escrever uma fila cheia. Os estados são: init, waitMT, writeMT, wait, write, write2, waitFull, ReadFull, read e read2.

A transição dos estados para a escrita de um novo valor na fila é no init onde são zerados quaisquer valores nos contadores e memória da RAM e em seguida vai para o waitMT, pois faz com o usuário não leia uma fila vazia e também sinaliza que a fila está vazia com o bit empty igual a 1 e fica esperando o usuário adicionar um novo valor. Quando está realizando uma escrita o próximos estados são wait, write, write2 e retorna para wait, caso esteja cheia a fila o estado waitFULL fica travado até que aja uma leitura para o estado readFull.

A leitura inicia no estado wait e quando o bit rd é elevado para nível lógico 1 o modo seguinte é read e depois read2 assim retorna para wait.

A Figura 5, mostra o bloco completo de controle que as entradas estão com setas apontado para dentro do bloco e as saídas para fora.

Figura 4: Bloco de Controle



3.2 Bloco Operacional

O bloco operacional é composto por um multiplexador de 8x4, dois contadores de 4 bits, um flip flop tipo D e uma memória RAM de 16 palavras com 13 bits.

O multiplexador ele vai deixar passar o endereço de memória da escrita ou leitura de acordo com estado que a máquina estive e com a seleção por meio de um switch pelo usuário.

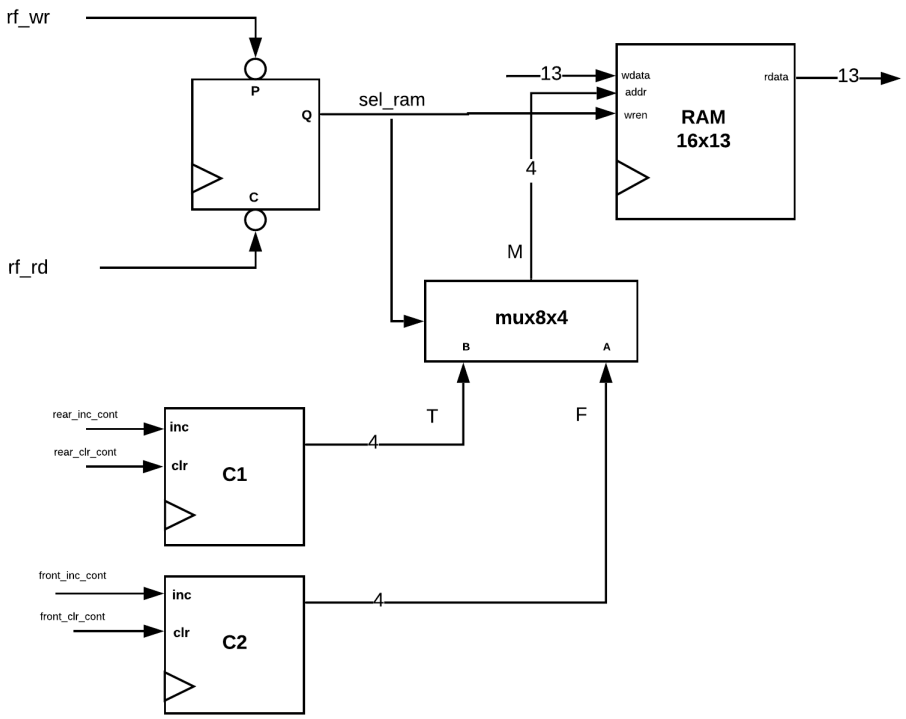
Existe dois contadores o primeiro conta o início da fila e é habilitado pelo bit (front wr inc) e o (rf wr) eleva o preset do flip flop D em 1 para que o multiplexador deixe o canal B passar o endereço de memória que esse contador de início de fila se ativa para escrita.

O segundo contador é o do fim da fila que é habilitado pelo bit (rear rd inc) e o (rf rd) eleva o preset para 1 e o canal A do multiplexador deixa passar o endereço de leitura.

A memória RAM foi escolhida para substituir o banco de registradores que poderia entrar na arquitetura, mas por motivo de facilitar o trabaho foi usando a memória RAM.

Por fim, o comparador de magnitude verifica se os contadores estão iguais, pois com isso se identifica que ou a fila está vazia ou cheia de acordo com o andar dos estados.

Figura 5: Bloco Operacional

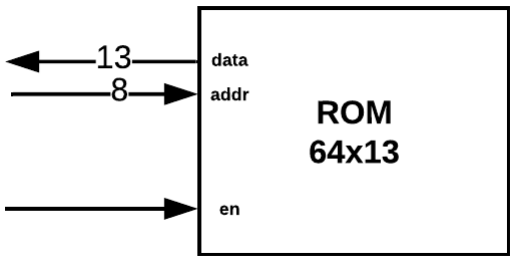


3.3 Mémoria ROM

A memória somente de leitura ou ROM (acrônimo em inglês de read-only memory) é um tipo de memória que permite apenas a leitura, ou seja, as suas informações são gravadas pelo fabricante uma única vez e após isso não podem ser alteradas ou apagadas, somente acessadas.

O uso da mémoria ROM é utilizada para salvar os valores da moedas que será requisitada pelo bloco operacional A (8 bits). Nela, há 64 posições de memória com 13 bits o tamanho de cada item e o en é o bit de habilitação de acesso a mémoria. O acesso do endereço de mémoria é feito por um contador de 6 bits, pois corresponde a quantidade de palavras da ROM.

Figura 6: Extrutura da mémoria ROM

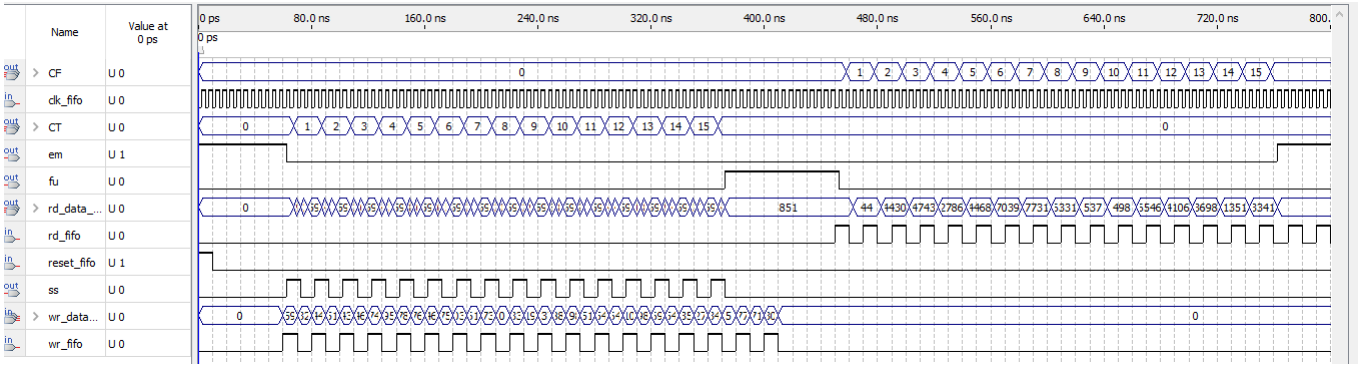


4 Resultados e Discurssões

O projeto foi criado no Quartus II que permitiu gerar as formas de ondas com mais detalhes e a Figura 7 mostra um exemplo em que foi chaveado dezoito vezes a da memória ROM na fila, portanto estava no modo de leitura. Após isso, a fila ficou cheia e foi sinalizado como pedido no exercício..

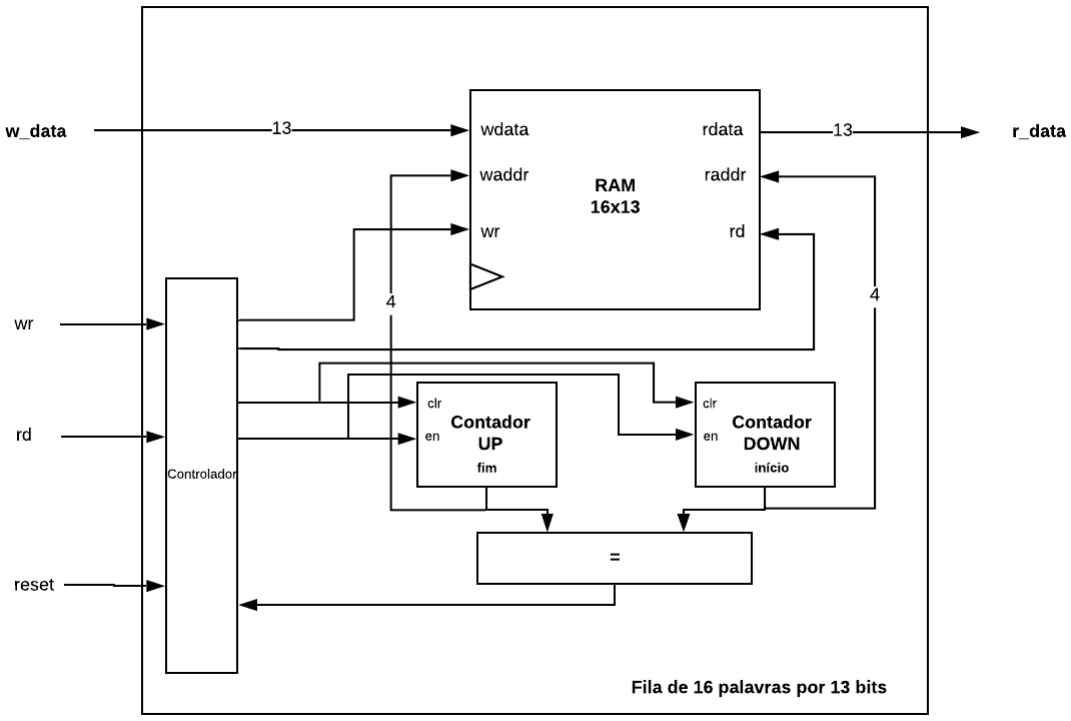
Em seguida foi realizada a leitura de dezoito vezes e a sinalização de fila vazia foi ativada como mostra a Figura 7.

Figura 7: Simulação no Quartus II



O projeto de uma fila basea-se em dois contadores que são ativados por um controlador e a gravação dos dados são colocados na memória RAM. A figura 8 mostra a estrutura de blocos.

Figura 8: Arquitetura de uma Fila



5 Conclusão

O desenvolvimento do projeto teve uma base de simulação antes de levar a placa do kit DE2, com isso o projeto da fila foi construída para ser apresentada nos displays de 7 segmentos e as entradas utilizando os botões push-bottom e chaves tipo switches.

6 Anexo

- Código da main

```
library ieee;
use ieee.std_logic_1164.all;

entity main is

  port(SW0, SW1: in std_logic;
                  clk_main, clk_rom: in std_logic;
                  KEY3: in std_logic;
                  H0, H1, H2, H3: out std_logic_vector(6 downto 0);
                  HX0, HX1: out std_logic_vector(6 downto 0);
                  LEDR0, LEDR1: out std_logic);

end;

architecture ckt of main is


component fifo is

port(clk_fifo, wr_fifo, rd_fifo, reset_fifo: in std_logic;
                em, fu, ss: out std_logic;
                CF, CT: out std_logic_vector(3 downto 0);
                wr_data_fifo: in std_logic_vector(12 downto 0);
                rd_data_fifo: out std_logic_vector(12 downto 0));

end component;

component mainrom IS
        PORT
        (
                address                 : IN STD_LOGIC_VECTOR (5 DOWNTO 0);
                clock                   : IN STD_LOGIC  := '1';
                q                       : OUT STD_LOGIC_VECTOR (12 DOWNTO 0)
        );
END component;

component bin_bcd is
```

```
port(SW : in  std_logic_vector(7 downto 0);
     HEX0, HEX1, HEX2 : out std_logic_vector(6 downto 0));

end component;

component contador8bit is
  port(
  clk_contador8bit: in std_logic;
  clr_contador8bit: in std_logic;
  SEL_contador8bit: in std_logic;
  Qs_contador8bit: out std_logic_vector(7 downto 0));

end component;


signal WR_DATA, RD_DATA: std_logic_vector(12 downto 0);
signal endereco: std_logic_vector(5 downto 0);

 begin

FILA: fifo port map(clk_fifo => clk_main, wr_fifo => SW1, rd_fifo => SW0,
reset_fifo => not KEY3, em => LEDR1, fu => LEDR0, wr_data_fifo => WR_DATA, rd_data_fifo => R

CONTADOR: contador8bit port map(clk_main, KEY3, not SW1, Qs_contador8bit(5 downto 0) => ende

ROM: mainrom port map(endereco, clk_rom, WR_DATA);

H0123: bin_bcd port map(RD_DATA(7 downto 0), H0, H1, H2);

H4: bin_bcd port map(SW(4 downto 0) => RD_DATA(12 downto 8),
                                       SW(5) => '0',
                                       SW(6) => '0',
                                       SW(7) => '0',
                                       HEX0 => H3);

SG: bin_bcd port map(SW(5 downto 0) => endereco,
                                       SW(6) => '0',
                                       SW(7) => '0',
                                  HEX0 => HX0, HEX1 => HX1);

end ckt;
```