**UNIVERSIDADE ESTADUAL DE MARINGÁ**

**CIÊNCIA DA COMPUTAÇÃO**

**PROJETO: RELÓGIO DIGITAL**

**CIRCUITOS DIGITAIS II (6882)**

**PROFESSOR: NARDÊNIO ALMEIDA MARTINS**

|  |  |
| --- | --- |
| **Discentes** | **RA** |
| Gabriel Belini | 64374 |
| João do Nascimento | 77992 |
| Juliano Donini | 63284 |
| Ricardo Ohara | 59574 |
| Thiago Bucalão | 68962 |

**Maringá, 03 de Junho de 2013**

**1. Introdução**

Este projeto consiste na implementação do desenvolvimento de um relógio digital. A implementação será realizada através do Quartus II, versão 9.1.

Os arquivos criados no projeto foram: **BCD\_Horas.vhd, BCD\_Minutos.vhd, BCD\_Segundos.vhd, cont\_mod6\_minutos.vhd, cont\_mod6\_segundos.vhd, cont\_mod10\_minutos.vhd, cont\_mod10\_segundos.vhd, cont\_mod24\_horas.vhd, contadorFFT.vhd, Relogio.vhd** e **Relogio\_package.vhd.**

**2. Objetivo**

Estudar e projetar circuitos combinacionais e sequenciais usando VHDL.

**3. Fundamentação Teórica**

Foi projetado um relógio digital mostrando as unidades e as dezenas de horas, minutos e segundos, na qual corresponde mostrar os valores de 010 a 910 para as unidades e de 010 a 510 para as dezenas, permitindo a contagem de 0010 a 5910 minutos e segundos. Para a contagem de 0010 a 2310, esta corresponde a um contador de módulo 24. Os valores das unidades e dezenas de minutos e segundos e os valores das horas pode ser mostrado em um *display* por meio de decodificadores **BCD 8421 para 7 segmentos.**

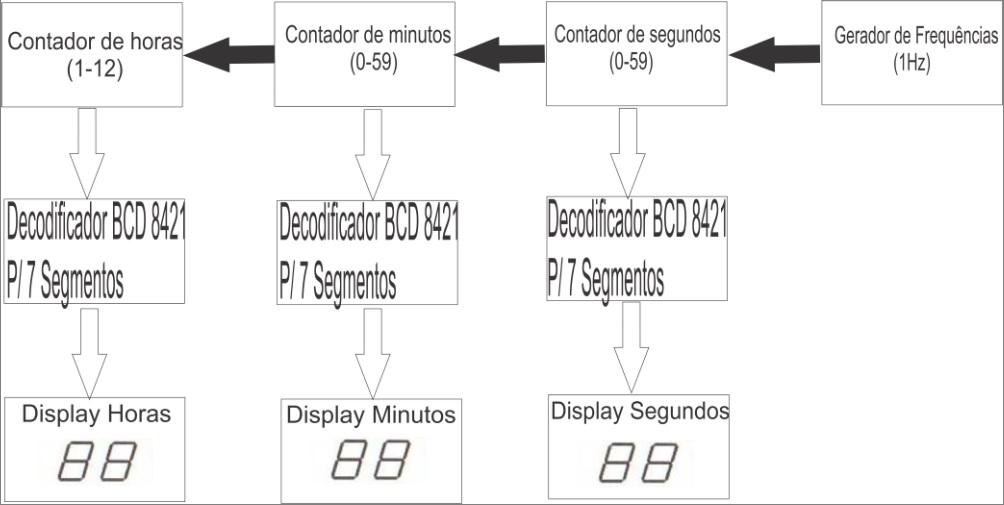
****

Figura 1 Diagrama de blocos do relógio digital.

De acordo com o diagrama de blocos, nota-se que a cada pulso do gerador de frequência, o contador de segundos apresenta sua contagem num display de 7 segmentos, gerando o pulso de clock para o contador de minutos, que apresenta no display de minutos. Este contador, gera o pulso de clock para o contador de horas. Assim sendo, podemos ver nos displays a contagem relativa às horas, minutos e segundos do relógio.

**3.1 BCD para 7 segmentos**

A sigla BCD representa as iniciais de *Binary Coded Decimal*, que significa **uma codificação do sistema decimal em binário.** Os termos seguintes (8421) significa os valores doa algarismos num dado número binário.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Decimal** | **BCD 8421** | | | |
|  | **A** | **B** | **C** | **D** |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |

Tabela 1 : Tabela Verdade.

O número de bits de um código é o número de dígitos binários que este possui. Percebe que, o código BCD 8421 é um código de 4 bits e que é válido de 010 a 910.

**3.2 Contadores**

Contadores são circuitos digitais que variam seus estados, sob controle de um clock, conforme com uma sequência predeterminada. São utilizados principalmente para contagens diversas, divisão de frequências, medição de frequência e tempo, geração de formas de onda e conversão de analógico para digital. São divididos em duas categorias: Contadores Assíncronos e Síncronos.

**3.2.1 Contadores Assíncronos**

Caracterizados por seus *flip-flops* funcionarem de maneira assíncrona (sem sicronismo), não tendo entradas clock em comum. A entrada clock se faz apenas no primeiro *flip-flop*, sendo as outras derivadas das saídas dos blocos anteriores. A estrutura básica de um contador deste tipo é um *flip-flop* do tipo JK.

**3.2.1.1 Contadores Assíncronos Decrescentes**

Sua principal aplicação é em situações onde se deve reconhecer quando um número desejado de pulsos de entrada ocorreu. O contador decrescente é inicializado com o número desejado e então habilitado a contar para baixo de acordo com os pulsos são aplicados. Quando o contador alcança o estado zero, isto é detectado por uma porta lógica cuja saída indica que o número de pulsos já ocorreu.

**3.2.1.2 Contadores Assíncronos de Módulo ≤ 2N**

O módulo de um contador é o número de estados únicos pelos quais o contador estabelece uma sequência. O número máximo de estados possíveis (módulo máximo) de um contador é , onde n é o número de flip-flops do contador. Os contadores podem ser projetados para ter um número de estados em sua sequência que é menor que o valor máximo de .

É necessário forçar o contador a reciclar antes que ele passe por todos os estados possíveis. Será citado um exemplo: um contador de década BCD tem que reciclar para o estado 0000 após o estado 1001. Um contador de década requer quatro flip-flops (três flip-flops são insuficientes pois = 8.

**3.2.2 Contadores Síncronos**

Possuem entradas clock curto-circuitadas, ou seja, o clock entra em todos os *flip-flops* simultaneamente, fazendo todos atuarem de forma sincronizada. Para que tenha mudanças de estado, deve-se utilizar as entradas J e K dos vários *flip-flops,* para que tenha nas saídas, as sequências desejadas.

Para estudar os contadores síncronos, deve ser feito a tabela verdade, analisando quais devem ser as entradas J e K dos vários *flip-flops*. Abaixo a tabela verdade do *flip-flop* JK.

|  |  |  |
| --- | --- | --- |
| **J** | **K** | **Qf** |
| 0 | 0 | Qa |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 |  |

Tabela 2: Tabela Verdade JK.

A partir desta tabela, será construída outra, relacionando os estados de saída e as entradas J e K.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Qa** | **Qf** | **J** | **K** |
| a) | 0 | 0 | 0 | X |
| b) | 0 | 1 | 1 | X |
| c) | 1 | 0 | X | 1 |
| d) | 1 | 1 | X | 0 |

Tabela 3: Tabela Verdade.

a) Se o flip-flop estiver em 0 (Qa = 0) e se for desejado que o estado a ser assumido seja 0 (Qf = 0), pode-se tanto manter o estado do flip-flop (J = 0, K = 0, → Qf = Qa), como fixar 0 (J = 0, K = 1 → Qf = 0), logo, se J = 0 e K = X, tem a passagem de Qa = 0 para Qf = 0.

b) Se o flip-flop estiver em 0 (Qa = 0) e se for desejado que o estado a ser assumido seja 1 (Qf = 1), pode-se tanto inverter o estado (J = 1, K = 1 → Qf = ), como fixar 1 (J = 1, K = 0, → Qf = 1), logo, se J = 1 e K = X, tem a passagem de Qa = 0 para Qf = 1.

c) Quando o flip-flop estiver em 1 (Qa = 1) e se for desejado que ele vá para 0 (Qf = 0), pode-se inverter o estado (J = 1, K = 1, → Qf = ) ou fixar 0 (J = 0, K = 1→Qf = 0), logo, se J = X e K = 1, tem a passagem de Qa = 1 para Qf = 0.

d) Quando o flip-flop estiver em 1 (Qa = 1) e se for desejado que ele permaneça em 1 (Qf = 1), pode-se manter o estado (J = 0, K = 0 → Qf = Qa) ou fixar 1 (J = 1, K = 0 → Qf = 1), logo, se J = X e K = 0, tem a passagem de Qa = 1 para Qf = 1.

**3.3 Registradores**

**3.3.1 Registradores de Deslocamento**

Trata-se de um certo número de flip-flops tipo JK mestre-escravo ligado de tal forma que as saídas de cada bloco sejam aplicadas nas entradas J e K respectivas do flip-flop seguinte, sendo o primeiro, com suas entradas ligadas na forma de um flip-flop tipo D.

**3.3.2 Conversor Série-Paralelo**

* **Informação Série:** Utiliza apenas 1 fio, sendo que os bits de informação vem sequencialmente, um após o outro.
* **Informação Paralela:** Todos os bits se apresentam simultaneamente. Necessita tantos fios quantos forem os bits contidos nela, além, do fio referencial do sistema.

**3.3.3 Conversor Série-Série**

Nessa aplicação, após a entrada da informação, se inibir a entrada de clock, a informação permanecerá no registrador até que haja uma nova entrada. É fácil observar que o registrador funciona como uma memória. A entrada de informação série se faz na entrada série do registrador e pode ser recolhida na saída Q0 do registrador.

**3.3.4 Conversor Paralelo-Paralelo**

A entrada paralela se faz através dos terminais preset e clear. Se inibir a entrada de clock, a informação contida no registrador pode ser acessada pelos terminais de saída Q3,Q2,Q1 e Q0.

**3.3.5 Conversor Paralelo-Série**

Para entrar com uma informação paralela, necessita de um registrador que apresente entradas Preset e Clear, que através desta faz-se que o Registrador armazene a informação paralela.

**3.4 Flip-flop JK Mestre-Escravo**

Desenvolvido para resolver um problema característico do Flip-Flop tipo JK, que é a alteração das entradas enquanto o sinal do clock for 1, alterando as saídas até que o clock seja 0. Para corrigir esse erro, foi desenvolvido um circuito que conforme é dado o pulso no clock suas entradas são bloqueadas e a saída só é fornecida quando o pulso deste clock é 0.

**3.5 Flip-flop tipo T**

É obtido a partir de um flip-flop JK Mestre-Escravo, na qual tem as entradas J e K curto-circuitadas, assim o circuito só pode assumir dois estados lógicos. Esse flip-flop é usado como célula principal dos contadores assíncronos, além de serem divisores de frequências.

**4. Metodologia**

O Relógio Digital possui um divisor de frequência a um contador de módulo 60 (contador de módulo 6 e contador de módulo 10), na qual divide a entrada de 60 pps a uma frequência de 1 pps. Ou seja, o contador de segundos divide em cont\_mod6\_segundos e cont\_mod10\_segundos, o contador de minutos divide em cont\_mod6\_minutos e cont\_mod10\_minutos e o contador de horas divide em um contador de módulo 24 (cont\_mod24\_horas).

No contador de segundos e minutos, o contador de módulo 6 possui 3 flip-flops do tipo T e o contador de módulo 10 possui 4 flip-flops do tipo T (lembrando que , onde n é o número de flip-flops do contador), então no contador de módulo 6 ( e no contador de módulo 10 (. No contador de horas, o contador de módulo 24 possui 5 flip-flops do tipo T (

A contagem em Segundos\_mod6 (vetor de 3 posições) e Min\_mod6 (vetor de 3 posições) será feita na tabela verdade de 0002 a 1002 e na contagem em Segundos\_mod10 (vetor de 4 posições) e Min\_mod10 (vetor de 4 posições) será feita na tabela verdade de 00002 a 10012 (de acordo com a tabela verdade BCD 8421 para 7 segmentos na seção 3.1).

Em BCD\_Segundos, BCD\_Minutos e BCD\_Horas, utilizou o comando WHEN ELSE.

No Relogio\_package contém a declaração dos componentes.

**5. Resultados (Simulações)**

**5.1 BCD\_Horas**

Library ieee;

Use ieee.std\_logic\_1164.all;

Library work;

Use work.all;

ENTITY BCD\_Horas IS

PORT(Horas : IN STD\_LOGIC\_VECTOR(4 DOWNTO 0);

unidade\_seg\_7: OUT STD\_LOGIC\_VECTOR(6 DOWNTO 0);

dezena\_seg\_7: OUT STD\_LOGIC\_VECTOR(6 DOWNTO 0));

END BCD\_Horas;

ARCHITECTURE Fluxo\_Dados OF BCD\_Horas IS

BEGIN

unidade\_seg\_7 <= "1111110" WHEN (Horas = "00000" or Horas = "01010" or Horas = "10100") ELSE --0,10,20

"0110000" WHEN (Horas = "00001" or Horas = "01011" or Horas = "10101") ELSE --1,11,21

"1101101" WHEN (Horas = "00010" or Horas = "01100" or Horas = "10110") ELSE --2,12,22

"1111001" WHEN (Horas = "00011" or Horas = "01101" or Horas = "10111") ELSE --3,13,23

"0110011" WHEN (Horas = "00100" or Horas = "01110") ELSE --4,14

"1011011" WHEN (Horas = "00101" or Horas = "01111") ELSE --5,15

"0011111" WHEN (Horas = "00110" or Horas = "10000") ELSE --6,16

"1110000" WHEN (Horas = "00111" or Horas = "10001") ELSE --7,17

"1111111" WHEN (Horas = "01000" or Horas = "10010") ELSE --8,18

"1110011" WHEN (Horas = "01001" or Horas = "10011") ELSE --9,19

"0000000";

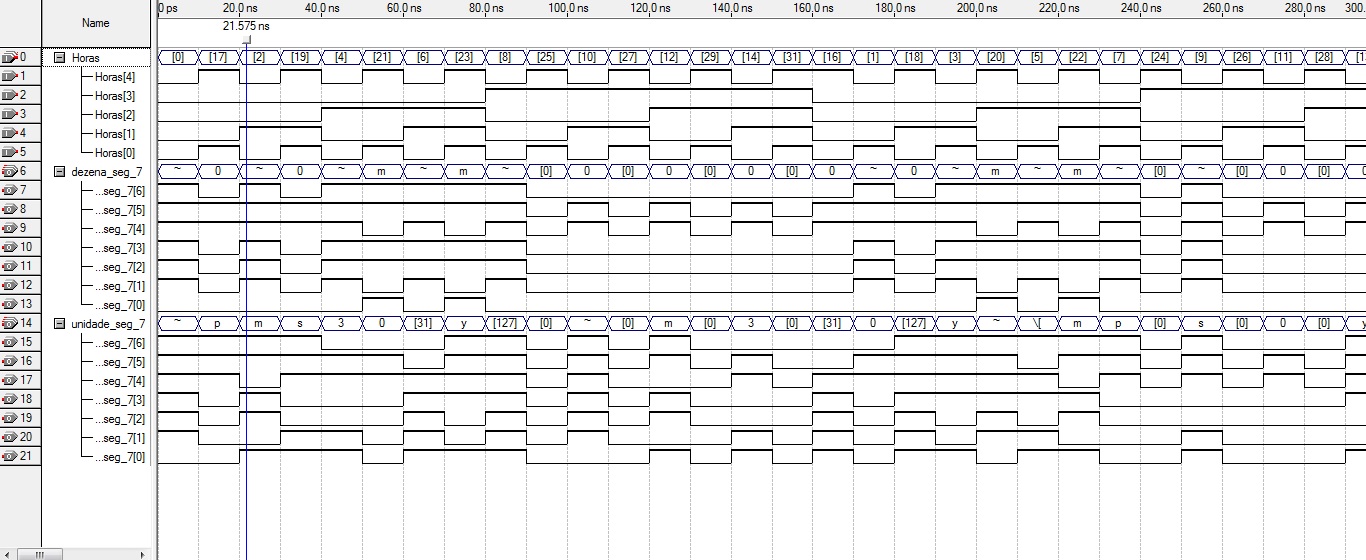
dezena\_seg\_7 <= "1111110" WHEN (Horas = "00000" or Horas = "00001" or Horas = "00010" or Horas = "00011" or Horas = "00100" or Horas = "00101" or Horas = "00110" or Horas = "00111" or Horas = "01000" or Horas = "01001") ELSE --00...09

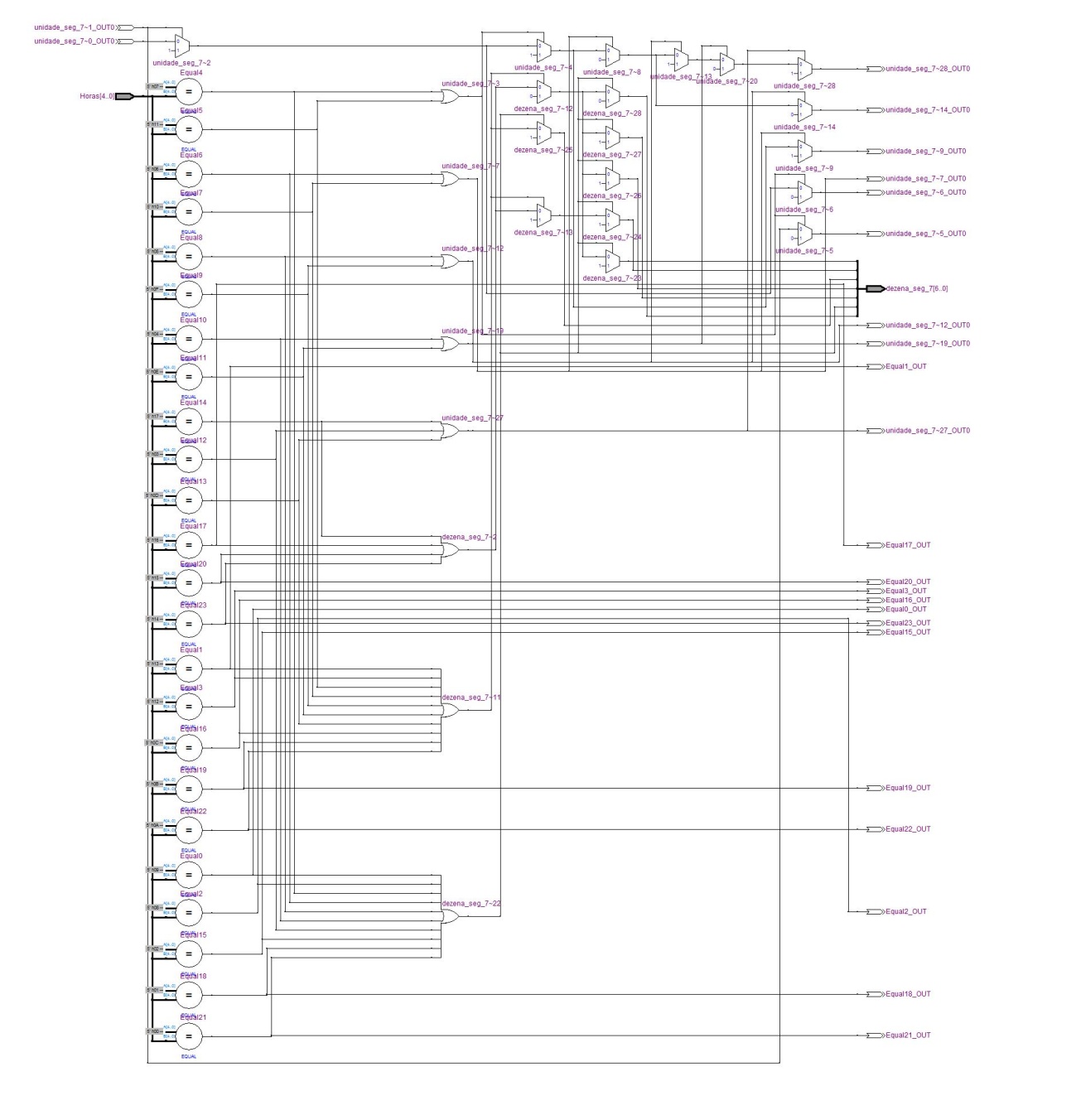
"0110000" WHEN (Horas = "01010" or Horas = "01011" or Horas = "01100" or Horas = "01101" or Horas = "01110" or Horas = "01111" or Horas = "10000" or Horas = "10001" or Horas = "10010" or Horas = "10011") ELSE --10...19

"1101101" WHEN (Horas = "10100" or Horas = "10101" or Horas = "10110" or Horas = "10111") ELSE --20...23

"0000000";

END Fluxo\_Dados;



****

**5.2 BCD\_minutos**

Library ieee;

Use ieee.std\_logic\_1164.all;

Library work;

Use work.all;

ENTITY BCD\_Minutos IS

PORT(Min\_mod10 : IN STD\_LOGIC\_VECTOR(3 DOWNTO 0);

Min\_mod6 : IN STD\_LOGIC\_VECTOR(2 DOWNTO 0);

unidade\_seg\_7: OUT STD\_LOGIC\_VECTOR(6 DOWNTO 0);

dezena\_seg\_7 : OUT STD\_LOGIC\_VECTOR(6 DOWNTO 0));

END BCD\_Minutos;

ARCHITECTURE Fluxo\_Dados OF BCD\_Minutos IS

BEGIN

unidade\_seg\_7 <= "1111110" WHEN Min\_mod10 = "0000" ELSE--0

"0110000" WHEN Min\_mod10 = "0001" ELSE--1

"1101101" WHEN Min\_mod10 = "0010" ELSE--2

"1111001" WHEN Min\_mod10 = "0011" ELSE--3

"0110011" WHEN Min\_mod10 = "0100" ELSE--4

"1011011" WHEN Min\_mod10 = "0101" ELSE--5

"0011111" WHEN Min\_mod10 = "0110" ELSE--6

"1110000" WHEN Min\_mod10 = "0111" ELSE--7

"1111111" WHEN Min\_mod10 = "1000" ELSE--8

"1110011" WHEN Min\_mod10 = "1001" ELSE--9

"0000000";

dezena\_seg\_7 <= "1111110" WHEN Min\_mod6 = "000" ELSE --0

"0110000" WHEN Min\_mod6 = "001" ELSE --1

"1101101" WHEN Min\_mod6 = "010" ELSE --2

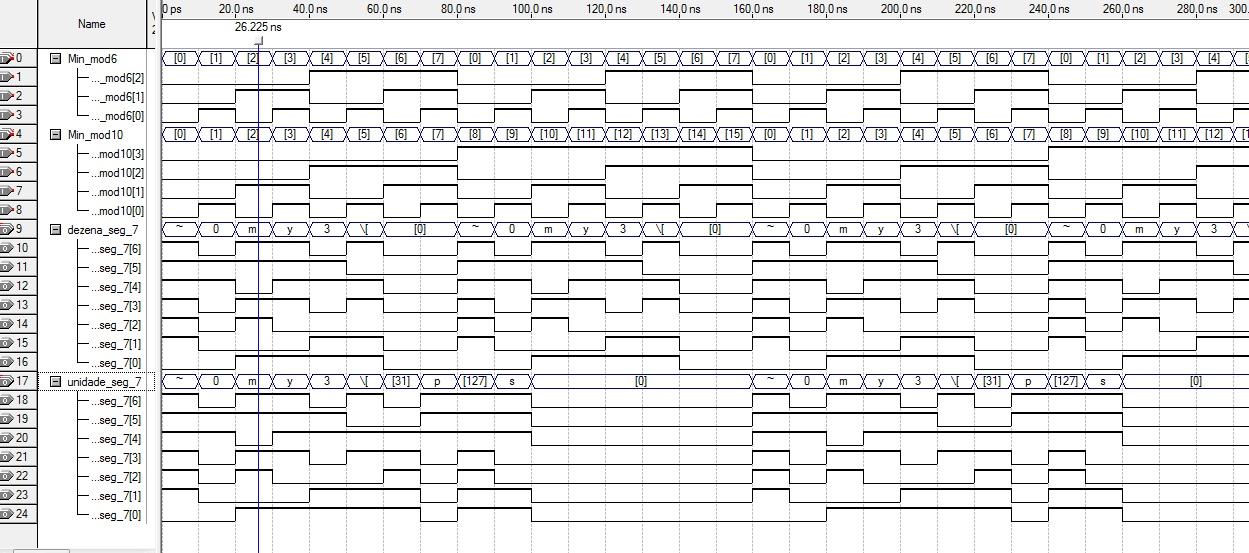
"1111001" WHEN Min\_mod6 = "011" ELSE --3

"0110011" WHEN Min\_mod6 = "100" ELSE --4

"1011011" WHEN Min\_mod6 = "101" ELSE --5

"0000000"; --conta de 0 a 59

END Fluxo\_Dados;



** 5.3 BCD\_Segundos**

Library ieee;

Use ieee.std\_logic\_1164.all;

Library work;

Use work.all;

ENTITY BCD\_Segundos IS

PORT(Segundos\_mod10 : IN STD\_LOGIC\_VECTOR(3 DOWNTO 0);

Segundos\_mod6 : IN STD\_LOGIC\_VECTOR(2 DOWNTO 0);

unidade\_seg\_7: OUT STD\_LOGIC\_VECTOR(6 DOWNTO 0);

dezena\_seg\_7: OUT STD\_LOGIC\_VECTOR(6 DOWNTO 0));

END BCD\_Segundos;

ARCHITECTURE Fluxo\_Dados OF BCD\_Segundos IS

BEGIN

unidade\_seg\_7 <= "1111110" WHEN Segundos\_mod10 = "0000" ELSE--0

"0110000" WHEN Segundos\_mod10 = "0001" ELSE--1

"1101101" WHEN Segundos\_mod10 = "0010" ELSE--2

"1111001" WHEN Segundos\_mod10 = "0011" ELSE--3

"0110011" WHEN Segundos\_mod10 = "0100" ELSE--4

"1011011" WHEN Segundos\_mod10 = "0101" ELSE--5

"0011111" WHEN Segundos\_mod10 = "0110" ELSE--6

"1110000" WHEN Segundos\_mod10 = "0111" ELSE--7

"1111111" WHEN Segundos\_mod10 = "1000" ELSE--8

"1110011" WHEN Segundos\_mod10 = "1001" ELSE--9

"0000000";

dezena\_seg\_7 <= "1111110" WHEN Segundos\_mod6 = "000" ELSE --0

"0110000" WHEN Segundos\_mod6 = "001" ELSE --1

"1101101" WHEN Segundos\_mod6 = "010" ELSE --2

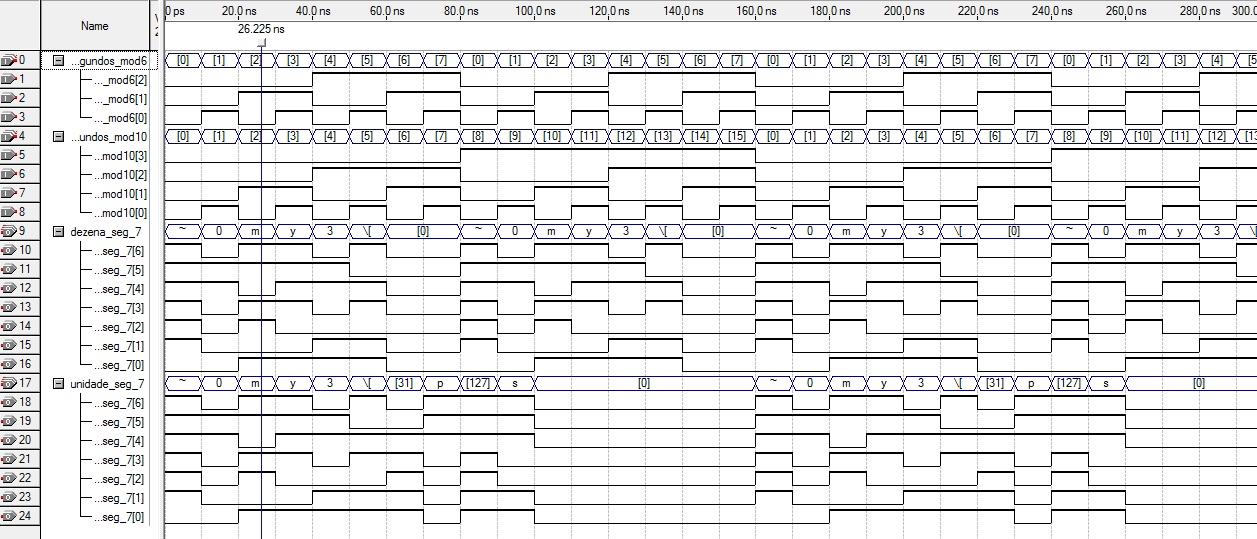
"1111001" WHEN Segundos\_mod6 = "011" ELSE --3

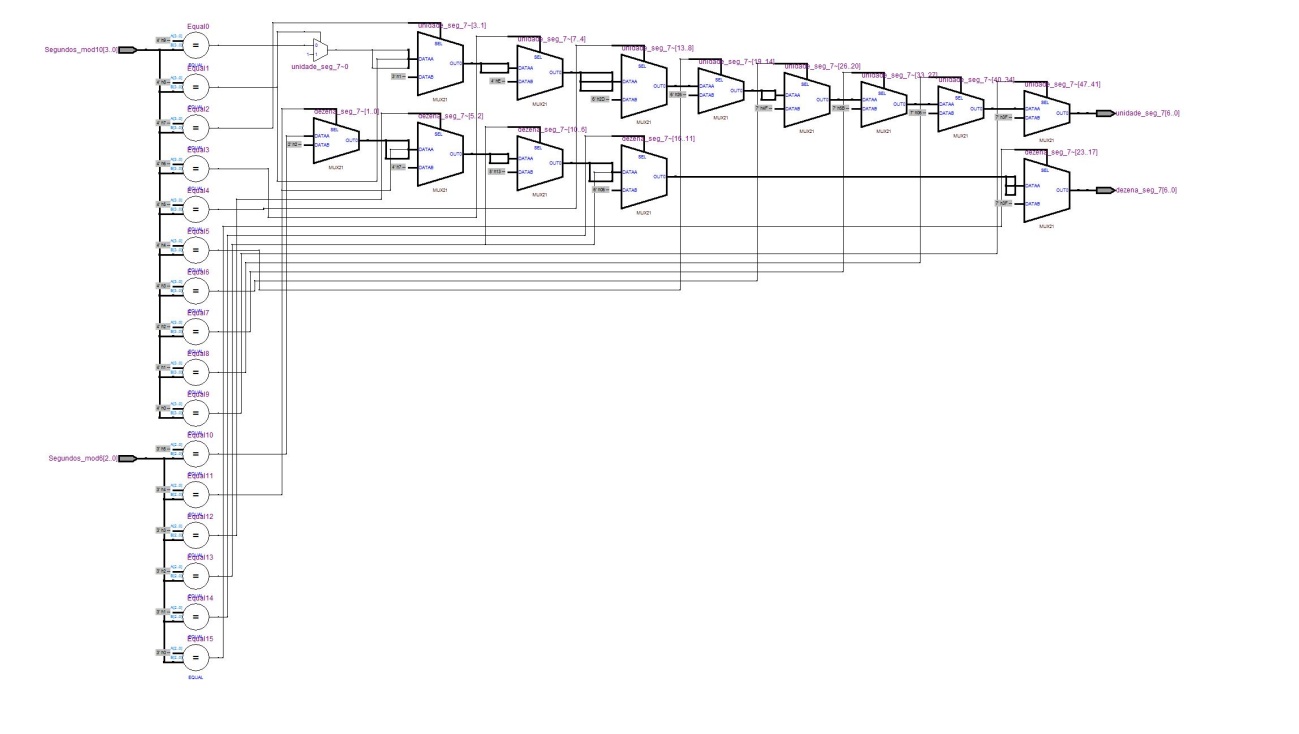
"0110011" WHEN Segundos\_mod6 = "100" ELSE --4

"1011011" WHEN Segundos\_mod6 = "101" ELSE --5

"0000000"; --conta de 0 a 59

END Fluxo\_Dados;



****

**5.4 cont\_mod6\_minutos**

Library ieee;

Use ieee.std\_logic\_1164.all;

Library work;

Use work.all;

ENTITY BCD\_Segundos IS

PORT(Segundos\_mod10 : IN STD\_LOGIC\_VECTOR(3 DOWNTO 0);

Segundos\_mod6 : IN STD\_LOGIC\_VECTOR(2 DOWNTO 0);

unidade\_seg\_7: OUT STD\_LOGIC\_VECTOR(6 DOWNTO 0);

dezena\_seg\_7: OUT STD\_LOGIC\_VECTOR(6 DOWNTO 0));

END BCD\_Segundos;

ARCHITECTURE Fluxo\_Dados OF BCD\_Segundos IS

BEGIN

unidade\_seg\_7 <= "1111110" WHEN Segundos\_mod10 = "0000" ELSE--0

"0110000" WHEN Segundos\_mod10 = "0001" ELSE--1

"1101101" WHEN Segundos\_mod10 = "0010" ELSE--2

"1111001" WHEN Segundos\_mod10 = "0011" ELSE--3

"0110011" WHEN Segundos\_mod10 = "0100" ELSE--4

"1011011" WHEN Segundos\_mod10 = "0101" ELSE--5

"0011111" WHEN Segundos\_mod10 = "0110" ELSE--6

"1110000" WHEN Segundos\_mod10 = "0111" ELSE--7

"1111111" WHEN Segundos\_mod10 = "1000" ELSE--8

"1110011" WHEN Segundos\_mod10 = "1001" ELSE--9

"0000000";

dezena\_seg\_7 <= "1111110" WHEN Segundos\_mod6 = "000" ELSE --0

"0110000" WHEN Segundos\_mod6 = "001" ELSE --1

"1101101" WHEN Segundos\_mod6 = "010" ELSE --2

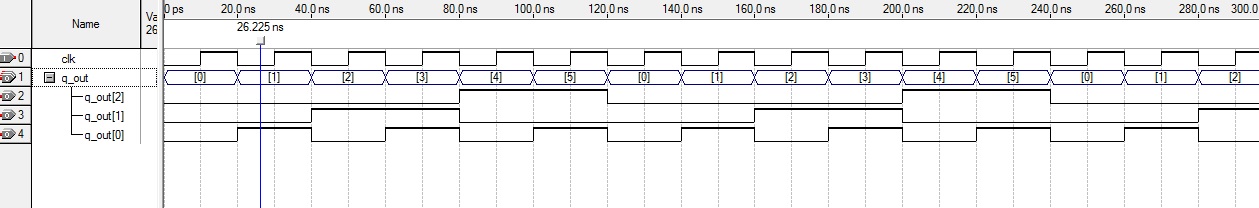
"1111001" WHEN Segundos\_mod6 = "011" ELSE --3

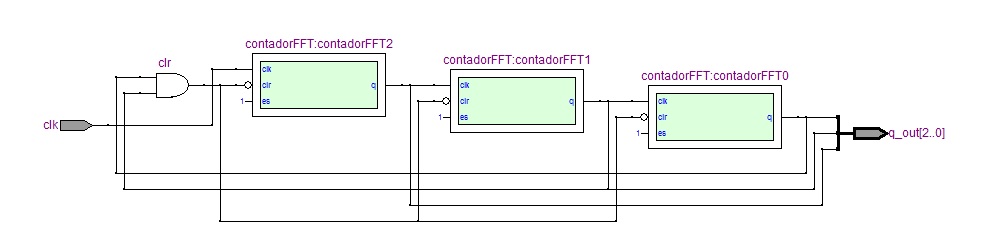
"0110011" WHEN Segundos\_mod6 = "100" ELSE --4

"1011011" WHEN Segundos\_mod6 = "101" ELSE --5

"0000000"; --conta de 0 a 59

END Fluxo\_Dados;





**5.5 cont\_mod6\_segundos**

Library ieee;

Use ieee.std\_logic\_1164.all;

Library work;

Use work.all;

ENTITY cont\_mod6\_segundos IS

PORT(clk : IN STD\_LOGIC;

q\_out: BUFFER STD\_LOGIC\_VECTOR(2 DOWNTO 0));

Constant es : STD\_LOGIC := '1';

END cont\_mod6\_segundos;

ARCHITECTURE estrutural OF cont\_mod6\_segundos IS

signal q0, q1, q2 : std\_logic;

signal clr : std\_logic;

BEGIN

clr <= '0' when (q0 = '1' and q1 ='1') else '1';

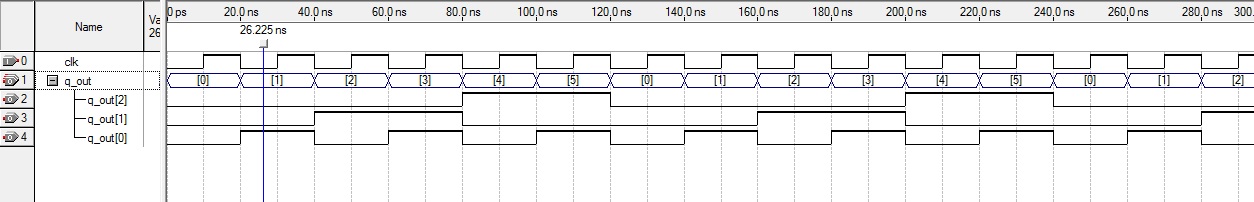
contadorFFT2 : contadorFFT PORT MAP(es, clr, clk, q2);

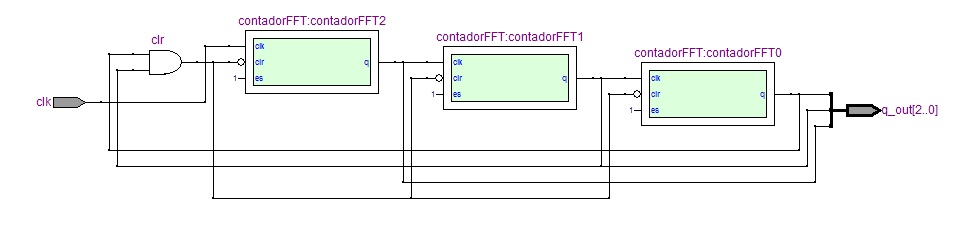
contadorFFT1 : contadorFFT PORT MAP(es, clr, q2, q1);

contadorFFT0 : contadorFFT PORT MAP(es, clr, q1, q0);

q\_out <= q0 & q1 & q2;

END estrutural;





**5.6 cont\_mod10\_minutos**

Library ieee;

Use ieee.std\_logic\_1164.all;

Library work;

Use work.all;

ENTITY cont\_mod10\_minutos IS

PORT(clk: IN STD\_LOGIC;

q\_out: BUFFER STD\_LOGIC\_VECTOR(3 DOWNTO 0));

Constant es : STD\_LOGIC := '1';

END cont\_mod10\_minutos;

ARCHITECTURE estrutural OF cont\_mod10\_minutos IS

signal q0, q1, q2, q3 : std\_logic;

signal clr : std\_logic;

BEGIN

clr <= '0' when (q2 = '1' and q0 = '1') else '1';

contadorFFT3 : contadorFFT PORT MAP(es, clr, clk, q3);

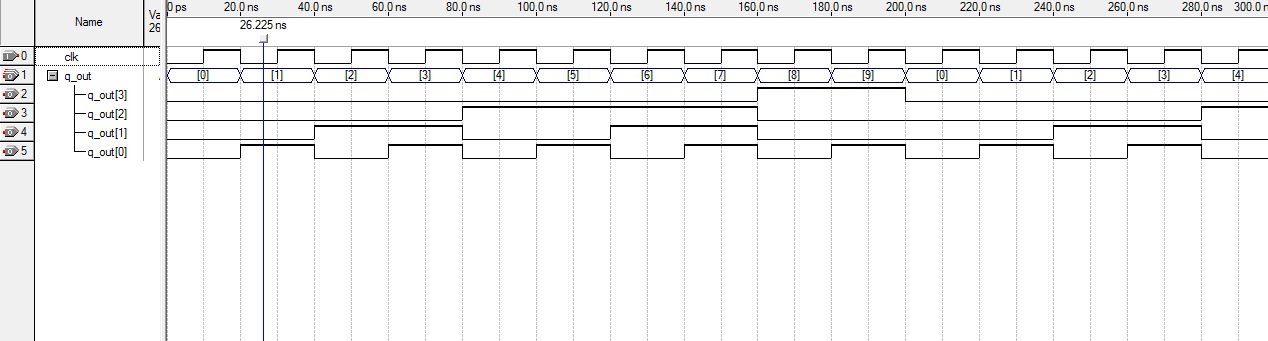
contadorFFT2 : contadorFFT PORT MAP(es, clr, q3, q2);

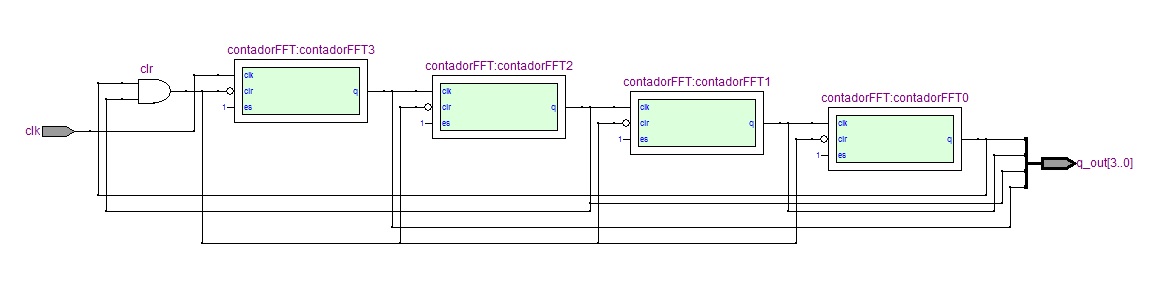
contadorFFT1 : contadorFFT PORT MAP(es, clr, q2, q1);

contadorFFT0 : contadorFFT PORT MAP(es, clr, q1, q0);

q\_out <= q0 & q1 & q2 & q3;

END estrutural;





**5.7 cont\_mod10\_segundos**

Library ieee;

Use ieee.std\_logic\_1164.all;

Library work;

Use work.all;

ENTITY cont\_mod10\_segundos IS

PORT(clk : IN STD\_LOGIC;

q\_out: BUFFER STD\_LOGIC\_VECTOR(3 DOWNTO 0));

Constant es : STD\_LOGIC := '1';

END cont\_mod10\_segundos;

ARCHITECTURE estrutural OF cont\_mod10\_segundos IS

signal q0, q1, q2, q3 : std\_logic;

signal clr : std\_logic;

BEGIN

clr <= '0' when (q2 = '1' and q0 = '1') else '1';

contadorFFT3 : contadorFFT PORT MAP(es, clr, clk, q3);

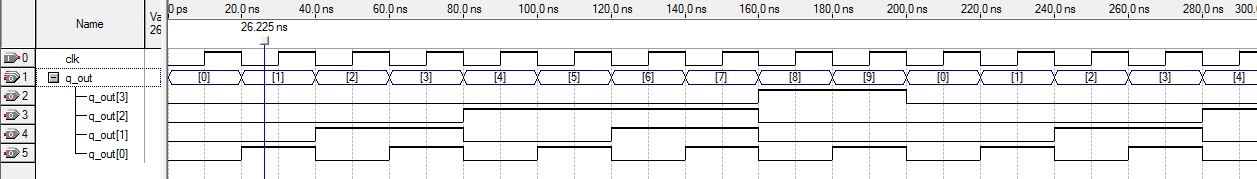
contadorFFT2 : contadorFFT PORT MAP(es, clr, q3, q2);

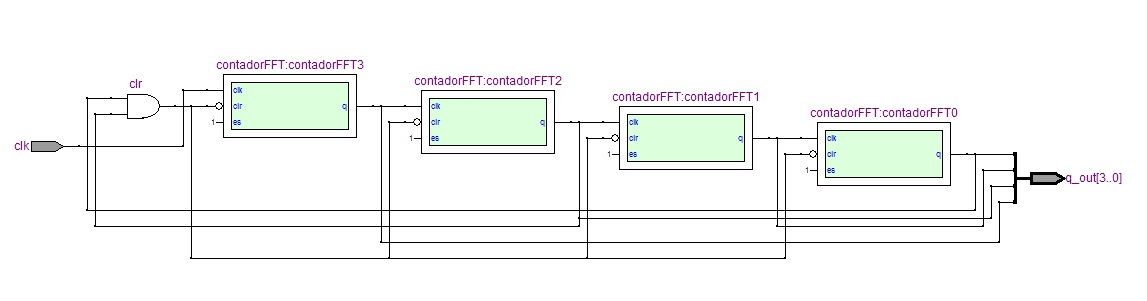
contadorFFT1 : contadorFFT PORT MAP(es, clr, q2, q1);

contadorFFT0 : contadorFFT PORT MAP(es, clr, q1, q0);

q\_out <= q0 & q1 & q2 & q3;

END estrutural;





**5.8 cont\_mod24\_horas**

Library ieee;

Use ieee.std\_logic\_1164.all;

Library work;

Use work.all;

ENTITY cont\_mod10\_segundos IS

PORT(clk : IN STD\_LOGIC;

q\_out: BUFFER STD\_LOGIC\_VECTOR(3 DOWNTO 0));

Constant es : STD\_LOGIC := '1';

END cont\_mod10\_segundos;

ARCHITECTURE estrutural OF cont\_mod10\_segundos IS

signal q0, q1, q2, q3 : std\_logic;

signal clr : std\_logic;

BEGIN

clr <= '0' when (q2 = '1' and q0 = '1') else '1';

contadorFFT3 : contadorFFT PORT MAP(es, clr, clk, q3);

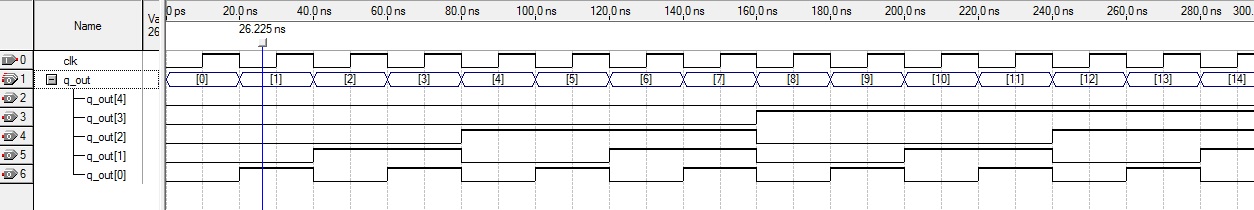
contadorFFT2 : contadorFFT PORT MAP(es, clr, q3, q2);

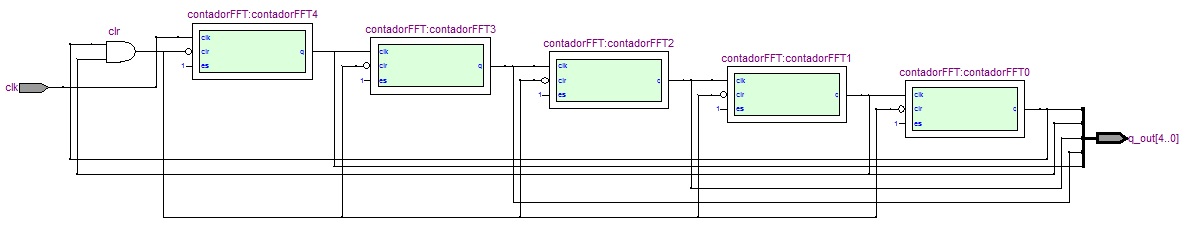
contadorFFT1 : contadorFFT PORT MAP(es, clr, q2, q1);

contadorFFT0 : contadorFFT PORT MAP(es, clr, q1, q0);

q\_out <= q0 & q1 & q2 & q3;

END estrutural;





**5.9 contadorFFT**

Library ieee;

Use ieee.std\_logic\_1164.all;

Library work;

Use work.all;

ENTITY contadorFFT IS

PORT(es, clr, clk : IN STD\_LOGIC;

q: BUFFER STD\_LOGIC);

END contadorFFT;

ARCHITECTURE logica OF contadorFFT IS

BEGIN

PROCESS(clr, clk, es)

VARIABLE qv : STD\_LOGIC;

BEGIN

IF (clr = '0') THEN

qv := '0';

ELSIF (falling\_edge(clk)) THEN

IF (es = '1') THEN

qv := NOT qv;

ELSE

qv := qv;

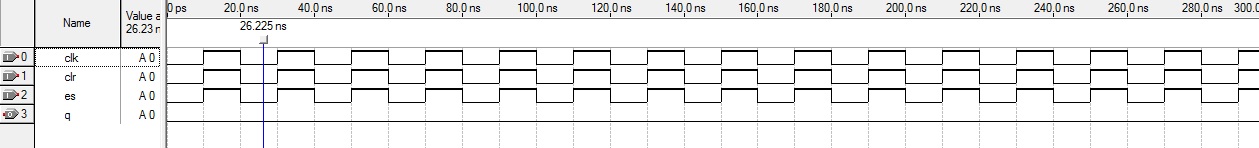
END IF;

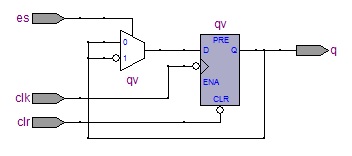
END IF;

q <= qv;

END PROCESS;

END logica;





**5.10 Relogio**

Library ieee;

Use ieee.std\_logic\_1164.all;

Library work;

USE WORK.Relogio\_package.all;

Entity Relogio IS

Port(clk: In Std\_logic;

-- SegundosUni, MinutosUni : OUT std\_logic\_vector (3 DOWNTO 0);

-- SegundosDez, MinutosDez : OUT std\_logic\_vector(2 DOWNTO 0);

-- Horas\_Relogio: OUT std\_logic\_vector (4 DOWNTO 0);

BCD\_SEG\_UNI, BCD\_SEG\_DEZ, BCD\_MIN\_UNI, BCD\_MIN\_DEZ, BCD\_HR\_UNI, BCD\_HR\_DEZ: OUT STD\_LOGIC\_VECTOR(6 DOWNTO 0));

constant es : STD\_LOGIC := '1';

END Relogio;

Architecture logic OF Relogio IS

SIGNAL SegundosUnidade, MinutosUnidade : std\_logic\_vector(3 downto 0);

SIGNAL SegundosDezena, MinutosDezena : std\_logic\_vector(2 downto 0);

SIGNAL Horas : std\_logic\_vector (4 downto 0);

SIGNAL BcdSegUni, BcdSegDez : std\_logic\_vector(6 downto 0);

SIGNAL BcdMinUni, BcdMinDez : std\_logic\_vector(6 downto 0);

SIGNAL BcdHrsUni, BcdHrsDez : std\_logic\_vector(6 downto 0);

BEGIN

Segundos0 : cont\_mod10\_segundos PORT MAP (clk, SegundosUnidade);

Segundos1 : cont\_mod6\_segundos PORT MAP (SegundosUnidade(3), SegundosDezena);

Minutos0 : cont\_mod10\_minutos PORT MAP (SegundosDezena(2), MinutosUnidade);

Minutos1 : cont\_mod6\_minutos PORT MAP (MinutosUnidade(3), MinutosDezena);

Horas0: cont\_mod24\_horas PORT MAP(MinutosDezena(2), Horas);

BcdSegundos : BCD\_Segundos PORT MAP (SegundosUnidade, SegundosDezena, BcdSegUni, BcdSegDez);

BcdMinutos : BCD\_Minutos PORT MAP (MinutosUnidade, MinutosDezena, BcdMinUni, BcdMinDez);

BcdHoras : BCD\_Horas PORT MAP (Horas, BcdHrsUni, BcdHrsDez);

-- SegundosUni <= SegundosUnidade;

-- SegundosDez <= SegundosDezena;

-- MinutosUni <= minutosUnidade;

-- MinutosDez <= MinutosDezena;

-- Horas\_Relogio <= Horas;

BCD\_SEG\_UNI <= BcdSegUni;

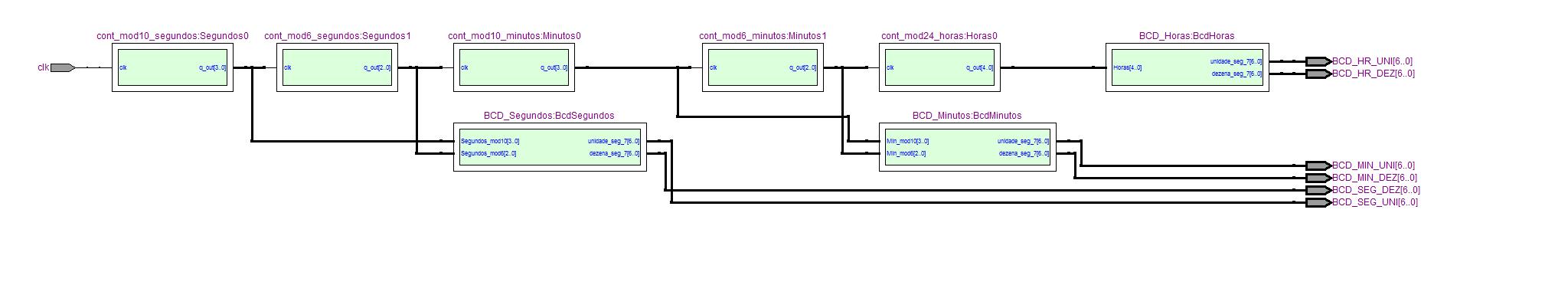
BCD\_SEG\_DEZ <= BcdSegDez;

BCD\_MIN\_UNI <= BcdMinUni;

BCD\_MIN\_DEZ <= BcdMinDez;

BCD\_HR\_UNI <= BcdHrsUni;

BCD\_HR\_DEZ <= BcdHrsDez;

END logic;

**5.11 Relogio\_package**

Library ieee;

Use ieee.std\_logic\_1164.all;

Library work;

Use work.all;

Package Relogio\_package IS

COMPONENT cont\_mod6\_segundos IS

PORT(clk : IN STD\_LOGIC;

q\_out: BUFFER STD\_LOGIC\_VECTOR(2 DOWNTO 0));

--Constant es : STD\_LOGIC := '1';

END COMPONENT;

COMPONENT cont\_mod10\_segundos IS --ES = entrada serial

PORT(clk : IN STD\_LOGIC;

q\_out: BUFFER STD\_LOGIC\_VECTOR(3 DOWNTO 0));

-- Constant es : STD\_LOGIC := '1';

END COMPONENT;

COMPONENT cont\_mod6\_minutos IS

PORT(clk : IN STD\_LOGIC;

q\_out: BUFFER STD\_LOGIC\_VECTOR(2 DOWNTO 0));

-- Constant es : STD\_LOGIC := '1';

END COMPONENT;

COMPONENT cont\_mod10\_minutos IS --ES = entrada serial

PORT(clk: IN STD\_LOGIC;

q\_out: BUFFER STD\_LOGIC\_VECTOR(3 DOWNTO 0));

-- Constant es : STD\_LOGIC := '1';

END COMPONENT;

COMPONENT cont\_mod24\_horas IS

PORT(clk : IN STD\_LOGIC;

q\_out: BUFFER STD\_LOGIC\_VECTOR(4 DOWNTO 0));

-- Constant es : STD\_LOGIC := '1';

END COMPONENT;

COMPONENT contadorFFT IS

PORT(es, clr, clk : IN STD\_LOGIC;

q: BUFFER STD\_LOGIC);

END COMPONENT;

COMPONENT BCD\_Segundos IS

PORT(Segundos\_mod10 : IN STD\_LOGIC\_VECTOR(3 DOWNTO 0);

Segundos\_mod6 : IN STD\_LOGIC\_VECTOR(2 DOWNTO 0);

unidade\_seg\_7: OUT STD\_LOGIC\_VECTOR(6 DOWNTO 0);

dezena\_seg\_7: OUT STD\_LOGIC\_VECTOR(6 DOWNTO 0));

END COMPONENT;

COMPONENT BCD\_Minutos IS

PORT(Min\_mod10 : IN STD\_LOGIC\_VECTOR(3 DOWNTO 0);

Min\_mod6 : IN STD\_LOGIC\_VECTOR(2 DOWNTO 0);

unidade\_seg\_7: OUT STD\_LOGIC\_VECTOR(6 DOWNTO 0);

dezena\_seg\_7 : OUT STD\_LOGIC\_VECTOR(6 DOWNTO 0));

END COMPONENT;

COMPONENT BCD\_Horas IS

PORT(Horas : IN STD\_LOGIC\_VECTOR(4 DOWNTO 0);

unidade\_seg\_7: OUT STD\_LOGIC\_VECTOR(0 TO 6);

dezena\_seg\_7: OUT STD\_LOGIC\_VECTOR(0 TO 6));

END COMPONENT;

END Relogio\_package;

**Bibliografia**

CAPUANO, Francisco Gabriel; IDOETA, IVAN V, **ELEMENTOS DE ELETRÔNICA DIGITAL**. Editora Erica, 2001.

TOCCI, Ronald J,; WIDMER, Neal S. **SISTEMAS DIGITAIS: PRINCÍPIOS E APLICAÇÕES**. 8. ed. Prentice-hall, 2003.

FLOYD, Thomas. **SISTEMAS DIGITAIS: FUNDAMENTOS E APLICAÇÕES**. 9.ed.Bookman, 2007.

COSTA, CESAR DA; MESQUITA, LEONARDO; PINHEIRO, EDUARDO. **ELEMENTOS DE LÓGICA PROGRAMÁVEL COM VHDL E DSP - TEORIA E PRÁTICA. EDITORA**. ÉRICA, 2009.