Universidade de Brasília – UNB

Curso: Engenharia de Redes de Comunicação
Disciplina: Laboratório de Sistemas Digitais
Turma: 08

Relatório da Disciplina Laboratório de Sistemas Digitais
Tema: Experimento 07 – Máquinas de Estado de Moore

Aluno: Pedro Henrique Dias Avelar
Matrícula: 241037112
Professor: Eduardo Paiva

Referências

Figura 1:Diagrama de transição de estados para o contador BCD módulo 10 4
Figura 2:Resultado do testbench para o contador BCD módulo 10 10
Figura 3:Primeira parte do resultado do primeiro cenário de teste 10
Figura 4:Segunda parte do resultado do primeiro cenário de teste 10
Figura 5:Resultado dos cenários 2 e 3 11
Figura 6:Diagrama das conexões dos dois contadores BCD Módulo 10 em cascata para formar o Contador BCD Módulo 100 12
Figura 7:Resultado do testbench para o contador BCD módulo 100 15
Figura 8:Contagem de 0 a 10 15
Figura 9:Contagem de 10 a 20 16
Figura 10:Contagem de 20 a 30 16
Figura 11:Contagem de 30 a 40 16
Figura 12:Contagem de 40 a 50 16
Figura 13:Contagem de 50 a 60 16
Figura 14:Contagem de 60 a 70 17
Figura 15:Contagem de 70 a 80 17
Figura 16:Contagem de 80 a 90 17
Figura 17:Contagem de 90 a 99 17
Figura 18:Resultado dos cenários 2 e 3 de teste 18

Tabela 1:Exemplo de números nas representações binária convencional e BCD 3
Tabela 2:Tabela de transição de estados para o contador BCD Módulo 10 5

Código 1:Modelagem do contador BCD módulo 10 7
Código 2:Definição do tipo de dados customizado ESTADO_BCD 8
Código 3:Testbench para o contador BCD módulo 10 10
Código 4:Modelagem do contador BCD módulo 100 14
Código 5:Testbench para o contador BCD módulo 100 15

Introdução

O presente experimento tem os seguintes objetivos:

- Implementar um contador BCD módulo 10 como uma máquina de estados do tipo Moore
- Implantar um contador BCD módulo 100 usando contadores BCD módulo 10 em cascata

Atividade 1

Implemente em VHDL e simule no ModelSim um contador BCD módulo 10. Este contador deve ser implementado como uma máquina de Moore com 10 estados, cada um associado a um número de 0 a 9. Este contador deve ter duas saídas: Q, de 4 bits, que é a representação BCD do estado atual e RCO que é 0 no estado 9 e 1 nos outros estados. Este contador deve ter as seguintes entradas:

•CLOCK: é o clock do sistema. As transições de estado acontecem apenas na borda de subida do clock (inclusive transições causadas pelo RESET e LOAD).

•RESET: quando igual a 1, a máquina deve retornar ao estado inicial (associado ao número 0).

•ENABLE e RCI: ambas ativas em nível baixo. A contagem acontece apenas se ENABLE = RCI = 0.

•D: entrada de quatro bits.

•LOAD: se LOAD = 1, a máquina vai (de forma síncrona) para o estado associado ao número representado pelo valor da entrada D.

Se tanto a entrada LOAD quanto a entrada RESET estiverem no nível alto, RESET tem prioridade. Tanto LOAD quanto RESET têm prioridade sobre a contagem.

Um contador BCD é um circuito que, ao invés de contar usando a representação binária convencional, utiliza a representação BCD. A representação BCD (Binary-Coded Decimal) utiliza 4 bits para representar um número de zero a 9. Os 4 últimos bits representam a unidade, os 4 penúltimos bits representam a dezena, os 4 antepenúltimos bits representam as centenas e assim por diante. A tabela abaixo demonstra alguns exemplos:

Número Decimal	Binário Convencional	BCD
2	10	0010
10	1010	0001 0000
225	0111 1101	0010 0010 0101
7654	0001 1101 1110 0110	0111 0110 0101 0100
9999	0010 0111 0000 1111	1001 1001 1001 1001

Tabela 1:Exemplo de números nas representações binária convencional e BCD

De acordo com o comportamento descrito no enunciado da atividade 1, podemos modelar o diagrama de transição de estados para o contador BCD módulo 10 da seguinte maneira:

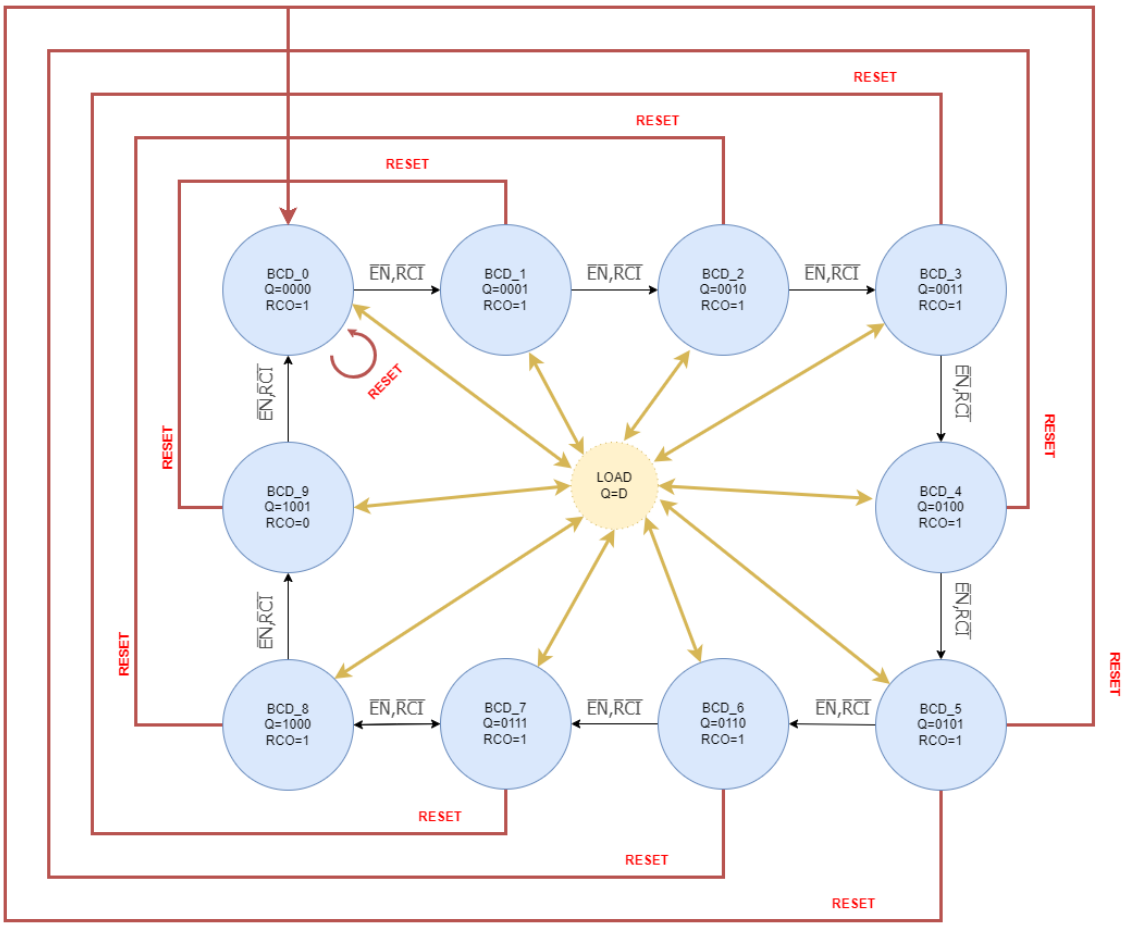

Figura 1:Diagrama de transição de estados para o contador BCD módulo 10

Os estados BCD_0 a BCD_9 representam, respectivamente, os dígitos de 0 a 9. Pelo diagrama, podemos ver que o contador irá percorrer em sequência os estados que representam os números de 0 a 9, em ordem crescente, quando as duas entradas ENABLE e RCI estiverem em nível baixo. As setas amarelas representam a operação de LOAD; no caso, LOAD não é um estado possível do contador, ele simplesmente representa a possibilidade do contador transitar de qualquer estado para qualquer outro estado sendo Q=D quando LOAD estiver ativo. E as setas vermelhas representam a operação de RESET; com a entrada RESET ativa, o contador irá sempre transitar para o estado BCD_0, independentemente das demais entradas.

O próximo passo é montar a tabela de transição de estados para o contador BCD. Segue a seguir a tabela:

Nome	Estado Atual				Entradas								Saídas					
					RESET	ENABLE	RCI	LOAD	D				Nome	Próximo Estado (Q*)				RCO
	Q3	Q2	Q1	Q0					D3	D2	D1	D0		Q3	Q2	Q1	Q0	
BCD_0	0	0	0	0	1	X	X	X	X	X	X	X	BCD_0	0	0	0	0	1
BCD_1	0	0	0	1	1	X	X	X	X	X	X	X	BCD_0	0	0	0	0	1
BCD_2	0	0	1	0	1	X	X	X	X	X	X	X	BCD_0	0	0	0	0	1
BCD_3	0	0	1	1	1	X	X	X	X	X	X	X	BCD_0	0	0	0	0	1
BCD_4	0	1	0	0	1	X	X	X	X	X	X	X	BCD_0	0	0	0	0	1
BCD_5	0	1	0	1	1	X	X	X	X	X	X	X	BCD_0	0	0	0	0	1
BCD_6	0	1	1	0	1	X	X	X	X	X	X	X	BCD_0	0	0	0	0	1
BCD_7	0	1	1	1	1	X	X	X	X	X	X	X	BCD_0	0	0	0	0	1
BCD_8	1	0	0	0	1	X	X	X	X	X	X	X	BCD_0	0	0	0	0	1
BCD_9	1	0	0	1	1	X	X	X	X	X	X	X	BCD_0	0	0	0	0	1
Dont care	**X**	**X**	**X**	**X**	0	X	X	1	0	0	0	0	BCD_0	0	0	0	0	1
Dont care	**X**	**X**	**X**	**X**	0	X	X	1	0	0	0	1	BCD_1	0	0	0	1	1
Dont care	**X**	**X**	**X**	**X**	0	X	X	1	0	0	1	0	BCD_2	0	0	1	0	1
Dont care	**X**	**X**	**X**	**X**	0	X	X	1	0	0	1	1	BCD_3	0	0	1	1	1
Dont care	**X**	**X**	**X**	**X**	0	X	X	1	0	1	0	0	BCD_4	0	1	0	0	1
Dont care	**X**	**X**	**X**	**X**	0	X	X	1	0	1	0	1	BCD_5	0	1	0	1	1
Dont care	**X**	**X**	**X**	**X**	0	X	X	1	0	1	1	0	BCD_6	0	1	1	0	1
Dont care	**X**	**X**	**X**	**X**	0	X	X	1	0	1	1	1	BCD_7	0	1	1	1	1
Dont care	**X**	**X**	**X**	**X**	0	X	X	1	1	0	0	0	BCD_8	1	0	0	0	1
Dont care	**X**	**X**	**X**	**X**	0	X	X	1	1	0	0	1	BCD_9	1	0	0	1	0
BCD_0	0	0	0	0	0	0	0	0	X	X	X	X	BCD_1	0	0	0	1	1
BCD_1	0	0	0	1	0	0	0	0	X	X	X	X	BCD_2	0	0	1	0	1
BCD_2	0	0	1	0	0	0	0	0	X	X	X	X	BCD_3	0	0	1	1	1
BCD_3	0	0	1	1	0	0	0	0	X	X	X	X	BCD_4	0	1	0	0	1
BCD_4	0	1	0	0	0	0	0	0	X	X	X	X	BCD_5	0	1	0	1	1
BCD_5	0	1	0	1	0	0	0	0	X	X	X	X	BCD_6	0	1	1	0	1
BCD_6	0	1	1	0	0	0	0	0	X	X	X	X	BCD_7	0	1	1	1	1
BCD_7	0	1	1	1	0	0	0	0	X	X	X	X	BCD_8	1	0	0	0	1
BCD_8	1	0	0	0	0	0	0	0	X	X	X	X	BCD_9	1	0	0	1	0
BCD_9	1	0	0	1	0	0	0	0	X	X	X	X	BCD_0	0	0	0	0	1
BCD_0	0	0	0	0	0	X	1	0	X	X	X	X	BCD_0	0	0	0	0	1
BCD_1	0	0	0	1	0	X	1	0	X	X	X	X	BCD_1	0	0	0	1	1
BCD_2	0	0	1	0	0	X	1	0	X	X	X	X	BCD_2	0	0	1	0	1
BCD_3	0	0	1	1	0	X	1	0	X	X	X	X	BCD_3	0	0	1	1	1
BCD_4	0	1	0	0	0	X	1	0	X	X	X	X	BCD_4	0	1	0	0	1
BCD_5	0	1	0	1	0	X	1	0	X	X	X	X	BCD_5	0	1	0	1	1
BCD_6	0	1	1	0	0	X	1	0	X	X	X	X	BCD_6	0	1	1	0	1
BCD_7	0	1	1	1	0	X	1	0	X	X	X	X	BCD_7	0	1	1	1	1
BCD_8	1	0	0	0	0	X	1	0	X	X	X	X	BCD_8	1	0	0	0	1
BCD_9	1	0	0	1	0	X	1	0	X	X	X	X	BCD_9	1	0	0	1	0
BCD_0	0	0	0	0	0	1	X	0	X	X	X	X	BCD_0	0	0	0	0	1
BCD_1	0	0	0	1	0	1	X	0	X	X	X	X	BCD_1	0	0	0	1	1
BCD_2	0	0	1	0	0	1	X	0	X	X	X	X	BCD_2	0	0	1	0	1
BCD_3	0	0	1	1	0	1	X	0	X	X	X	X	BCD_3	0	0	1	1	1
BCD_4	0	1	0	0	0	1	X	0	X	X	X	X	BCD_4	0	1	0	0	1
BCD_5	0	1	0	1	0	1	X	0	X	X	X	X	BCD_5	0	1	0	1	1
BCD_6	0	1	1	0	0	1	X	0	X	X	X	X	BCD_6	0	1	1	0	1
BCD_7	0	1	1	1	0	1	X	0	X	X	X	X	BCD_7	0	1	1	1	1
BCD_8	1	0	0	0	0	1	X	0	X	X	X	X	BCD_8	1	0	0	0	1
BCD_9	1	0	0	1	0	1	X	0	X	X	X	X	BCD_9	1	0	0	1	0

Tabela 2:Tabela de transição de estados para o contador BCD Módulo 10

Com RESET ativo, todos os 10 estados do contador irão transitar para o estado BCD_0. Já com LOAD ativo, independente do estado atual, a máquina irá transitar para o estado definido pelos 4 bits da entrada D. Com RESET, ENABLE, RCI e LOAD inativos, a máquina irá atuar como contador, transitando sempre para o estado que representa o próximo digito ate chegar ao

estado BCD_9, o qual irá então transitar para o estado BCD_0. Por fim, estando todas as entradas inativas exceto a entrada RCI, a máquina irá transitar para o mesmo estado. Este comportamento será importante para a atividade 2 com o uso de dois contadores BCD módulo 10 em cascata. O mesmo comportamento ocorreria para o caso de ENABLE ativa e RCI inativa, porém esta situação já não é tão relevante para o experimento proposto.

Assim, o contador foi modelado através do código a seguir:

```
01  -- Experimento 08 Atividade 1
02  -- Aluno: Pedro Henrique Dias Avelar 241037112
03  -- Turma 08
04  -- Data: 05/02/2025
05
06  -- Contador BCD Modulo 10
07
08  LIBRARY IEEE;
09  USE IEEE.STD_LOGIC_1164.ALL;
10  USE WORK.TYPE_ESTADO_BCD.ALL;
11
12  ENTITY CONTADOR_BCD_MOD10 IS
13      PORT (
14          CLOCK    : IN STD_LOGIC;
15          RESET    : IN STD_LOGIC;
16          ENABLE   : IN STD_LOGIC;
17          RCI      : IN STD_LOGIC;
18          D        : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
19          LOAD     : IN STD_LOGIC;
20          Q        : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
21          RCO      : OUT STD_LOGIC;
22          CURRENT_STATE: OUT ESTADO_BCD
23          );
24  END CONTADOR_BCD_MOD10;
25
26  ARCHITECTURE ARC_CONTADOR_BCD_MOD10 OF CONTADOR_BCD_MOD10 IS
27
28      SIGNAL ESTADO_ATUAL, PROXIMO_ESTADO: ESTADO_BCD;
29
30      BEGIN
31          CURRENT_STATE <= ESTADO_ATUAL; -- para poder observar o
estado no testbench
32
33          PROCESSO_SINCRONO: PROCESS (CLOCK)
34          BEGIN
35              IF RISING_EDGE(CLOCK) THEN
36                  ESTADO_ATUAL <= PROXIMO_ESTADO;
37              END IF;
38          END PROCESS PROCESSO_SINCRONO;
39
40          PROCESSO_COMBINACIONAL: PROCESS (ESTADO_ATUAL, RESET,
ENABLE, RCI, D, LOAD)
41          BEGIN
42              Q <= "0000";
43              RCO <= '1';
44              IF RESET = '1' THEN
45                  PROXIMO_ESTADO <= BCD_0;
46              ELSIF LOAD = '1' THEN
```

```
47                  CASE D IS
48                      WHEN "0000" => PROXIMO_ESTADO <= BCD_0;
49                      WHEN "0001" => PROXIMO_ESTADO <= BCD_1;
50                      WHEN "0010" => PROXIMO_ESTADO <= BCD_2;
51                      WHEN "0011" => PROXIMO_ESTADO <= BCD_3;
52                      WHEN "0100" => PROXIMO_ESTADO <= BCD_4;
53                      WHEN "0101" => PROXIMO_ESTADO <= BCD_5;
54                      WHEN "0110" => PROXIMO_ESTADO <= BCD_6;
55                      WHEN "0111" => PROXIMO_ESTADO <= BCD_7;
56                      WHEN "1000" => PROXIMO_ESTADO <= BCD_8;
57                      WHEN "1001" => PROXIMO_ESTADO <= BCD_9;
58                      WHEN OTHERS => PROXIMO_ESTADO <= BCD_0;
59                  END CASE;
60              ELSIF ENABLE = '0' AND RCI = '0' THEN
61                  CASE ESTADO_ATUAL IS
62                      WHEN BCD_0 => PROXIMO_ESTADO <= BCD_1;
63                      WHEN BCD_1 => PROXIMO_ESTADO <= BCD_2;
64                      WHEN BCD_2 => PROXIMO_ESTADO <= BCD_3;
65                      WHEN BCD_3 => PROXIMO_ESTADO <= BCD_4;
66                      WHEN BCD_4 => PROXIMO_ESTADO <= BCD_5;
67                      WHEN BCD_5 => PROXIMO_ESTADO <= BCD_6;
68                      WHEN BCD_6 => PROXIMO_ESTADO <= BCD_7;
69                      WHEN BCD_7 => PROXIMO_ESTADO <= BCD_8;
70                      WHEN BCD_8 => PROXIMO_ESTADO <= BCD_9;
71                      WHEN BCD_9 => PROXIMO_ESTADO <= BCD_0;
72                      WHEN OTHERS => PROXIMO_ESTADO <= BCD_0;
73                  END CASE;
74              ELSE
75                  PROXIMO_ESTADO <= ESTADO_ATUAL;
76              END IF;
77              CASE PROXIMO_ESTADO is
78                  WHEN BCD_0 => Q <= "0000";
79                  WHEN BCD_1 => Q <= "0001";
80                  WHEN BCD_2 => Q <= "0010";
81                  WHEN BCD_3 => Q <= "0011";
82                  WHEN BCD_4 => Q <= "0100";
83                  WHEN BCD_5 => Q <= "0101";
84                  WHEN BCD_6 => Q <= "0110";
85                  WHEN BCD_7 => Q <= "0111";
86                  WHEN BCD_8 => Q <= "1000";
87                  WHEN BCD_9 => Q <= "1001"; RCO <= '0';
88                  WHEN OTHERS => Q <= "0000";
89              END CASE;
90          END PROCESS PROCESSO_COMBINACIONAL;
91      END ARC_CONTADOR_BCD_MOD10;
```

Código 1:Modelagem do contador BCD módulo 10

Junto desse código, foi salvo em outro arquivo o código para armazenar o tipo de dados customizado ESTADO_BCD, cuja importação ocorre na linha 11 do código 1. Salvar essa declaração em um arquivo separado nos permite reutilizar o tipo customizado no código da atividade 2 e dos testbenches com facilidade.

```
01  library IEEE;
02  use IEEE.STD_LOGIC_1164.ALL;
03
04  PACKAGE TYPE_ESTADO_BCD IS
05  TYPE ESTADO_BCD IS (BCD_0,
06                      BCD_1,
07                      BCD_2,
08                      BCD_3,
09                      BCD_4,
10                      BCD_5,
11                      BCD_6,
12                      BCD_7,
13                      BCD_8,
14                      BCD_9);
15  END PACKAGE TYPE_ESTADO_BCD;
```

Código 2:Definição do tipo de dados customizado ESTADO_BCD

Na linha 22 do código 1 foi definida uma variável de saída do tipo ESTADO_BCD; similar ao que foi feito no experimento 7, essa variável não possui significado físico, servindo apenas para facilitar a observação do resultado da simulação realizada. O restante do código é também similar a modelagem realizada no experimento 7. Dividimos o funcionamento da máquina em dois processos. Na linha 33 do código 1 foi definido o processo síncrono, no qual a máquina sempre transita para o próximo estado definido pelo processo combinacional no evento de sincronia, o qual, conforme especificado no enunciado, é a borda de subida do clock. Na linha 40 do código 1 foi definido o processo combinacional. Definimos inicialmente as saídas Q e RCO como 0000 e 1, respectivamente. Isto pode ser visto como uma maneira de replicar uma máquina de risco mínimo – isto é, em caso de um comportamento inesperado, a máquina retorna para o estado BCD_0 – e também para facilitar a modelagem do código, visto que, como apenas no estado BCD_9 a saída RCO ficará desativada, atribuindo 1 a ela no início do processo combinacional faz com que precisemos nos atentar apenas aos casos em que ocorra transição para o estado BCD_9 para alterar seu valor.

Na linha 44 iniciamos uma sequência de IFs aninhados de modo a respeitar a ordem de prioridade das operações (RESET > LOAD > COUNT) do contador. Estando a entrada RESET ativa, o próximo estado sempre será BCD_0. Se não, se LOAD ativa (linha 46 do código 1), o próximo estado será definido pelo valor referente carregado nos bits da entrada D. Se não, estando RESET, LOAD inativas e ENABLE e RCI ativas (linha 60 do código 1) – reforçando que a lógica das entradas ENABLE e RCI são invertidas, isso é, elas estão ativas quando seu sinal está em baixa ou igual a zero – então o contador irá transitar para o próximo estado referente ao próximo número em sequência, exceto quando for o estado BCD_9, no qual então ele irá transitar de volta para o estado BCD_0. Por fim, caso nenhuma das condições anteriores seja atendida, então o contador irá transitar para o mesmo estado.

Finalizada a modelagem do contador, a próxima etapa é fazer o teste de funcionamento. Para isso, foi preparado o seguinte testbench:

```vhdl
-- Experimento 08 - TESTBENCH Q1
-- Aluno: Pedro Henrique Dias Avelar 241037112
-- Turma 08
-- Data: 05/02/2025

-- Testbench - Tempo de simulação: 310 NS

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE IEEE.NUMERIC_STD.ALL;
USE WORK.TYPE_ESTADO_BCD.ALL;

ENTITY TESTBENCH_E08_BCD_MOD10 IS
END TESTBENCH_E08_BCD_MOD10;

ARCHITECTURE ARC_TESTBENCH_E08_BCD_MOD10 OF
TESTBENCH_E08_BCD_MOD10 IS
COMPONENT CONTADOR_BCD_MOD10 IS
    PORT (
        CLOCK    : IN STD_LOGIC;
        RESET    : IN STD_LOGIC;
        ENABLE   : IN STD_LOGIC;
        RCI      : IN STD_LOGIC;
        D        : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        LOAD     : IN STD_LOGIC;
        Q        : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
        RCO      : OUT STD_LOGIC;
        CURRENT_STATE: OUT ESTADO_BCD
        );
END COMPONENT;

    SIGNAL CLOCK_TB: STD_LOGIC := '0';
    SIGNAL RESET_TB, ENABLE_TB, RCI_TB, LOAD_TB: STD_LOGIC;
    SIGNAL D_TB: STD_LOGIC_VECTOR(3 DOWNTO 0);
    SIGNAL Q_TB: STD_LOGIC_VECTOR(3 DOWNTO 0);
    SIGNAL RCO_TB: STD_LOGIC;
    SIGNAL CURRENT_STATE_TB: ESTADO_BCD;

    BEGIN
        DUT: CONTADOR_BCD_MOD10 PORT MAP (CLOCK_TB, RESET_TB,
ENABLE_TB, RCI_TB, D_TB, LOAD_TB, Q_TB, RCO_TB, CURRENT_STATE_TB);
        CLOCK_TB <= NOT CLOCK_TB AFTER 5 NS;
        PROCESS
        BEGIN
        REPORT "INICIANDO TESTE..." SEVERITY NOTE;
        -- CENARIO 1: CONTAR DE 0 A 9;
        RESET_TB <= '1'; WAIT FOR 10 NS;
        RESET_TB <= '0'; LOAD_TB <= '0'; ENABLE_TB <= '0';
RCI_TB <= '0';
        WAIT FOR 190 NS;
        -- CENARIO 2: LOAD DO NR 5; DEPOIS CONTAR ATE 9
        LOAD_TB <= '1'; D_TB <= "0101"; WAIT FOR 10 NS;
        LOAD_TB <= '0'; WAIT FOR 40 NS;
```

```
52          -- CENARIO 3: LOAD DO NR 7; DEPOIS ATIVAR RESET PARA
SOBRESCREVER O LOAD E DEPOIS CONTAR ATE 9
53          LOAD_TB <= '1'; D_TB <= "0111"; WAIT FOR 10 NS;
54          RESET_TB <= '1'; WAIT FOR 30 NS;
55          LOAD_TB <= '0'; RESET_TB <= '0'; WAIT FOR 20 NS;
56          REPORT "TESTE FINALIZADO!" SEVERITY NOTE;
57          WAIT;
58          END PROCESS;
```

Código 3:Testbench para o contador BCD módulo 10

O testbench teve uma duração total de 310 ns gerou o seguinte resultado:

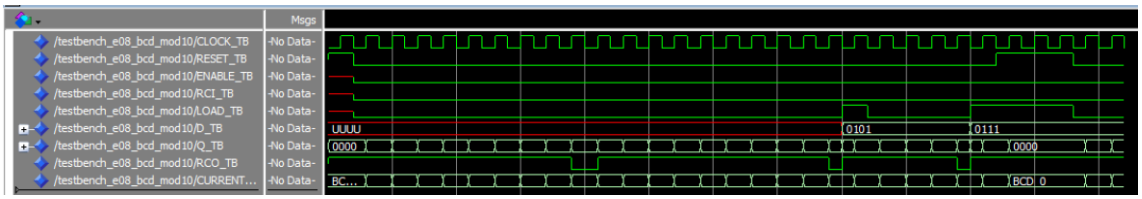

Figura 2:Resultado do testbench para o contador BCD módulo 10

O testbench consistiu em três cenários. Na linha 45 do código 3 está o primeiro cenário, uma simples contagem de 0 a 9, realizada duas vezes.

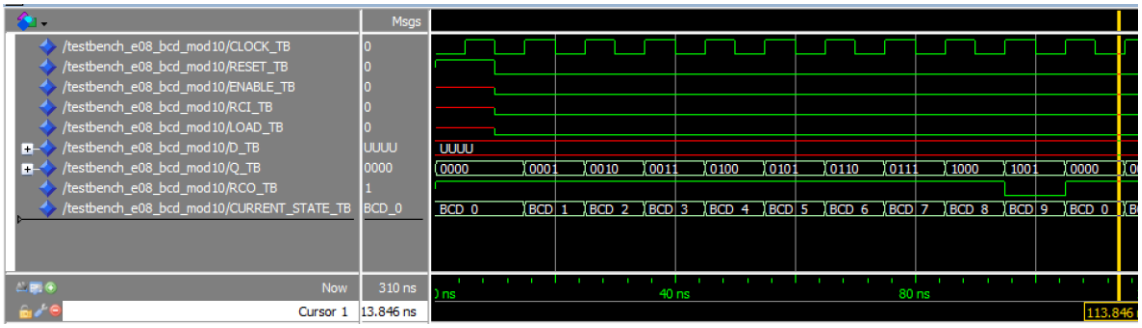

Figura 3:Primeira parte do resultado do primeiro cenário de teste

Inicialmente foi feito um RESET. Depois disso, o contador realizou sua função de contagem, percorrendo em sequência os estados BCD_0 a BCD_9, e então retornando ao estado BCD_0. Durante o estado BCD_9, a saída RCO ficou em baixa.

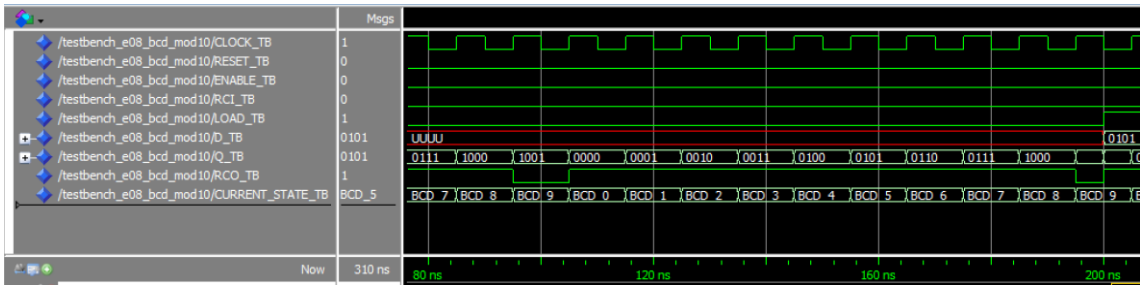

Figura 4:Segunda parte do resultado do primeiro cenário de teste

A segunda iteração do contador ocorreu conforme esperado, contando de 0 a 9 e com a saída RCO novamente ficando em baixa durante o estado BCD_9.

O segundo cenário de teste envolveu uma operação de LOAD do número 5 seguida da retomada da função de contagem.

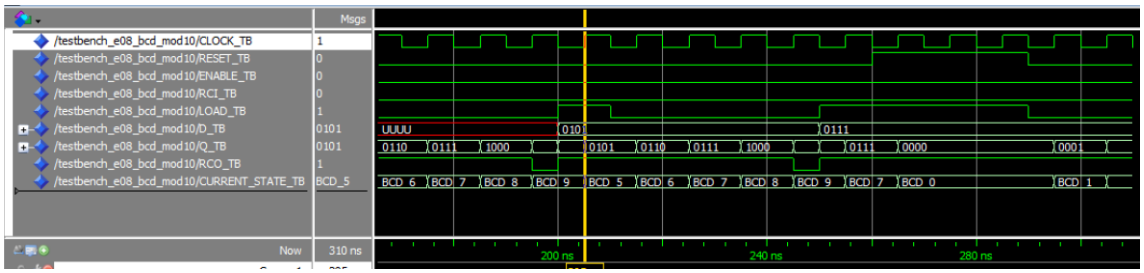

Figura 5:Resultado dos cenários 2 e 3

Na primeira borda de subida do CLOCK após a ativação da entrada LOAD e da entrada D com o número binário 5 (0101), o contador transitou para o estado BCD_5 e, em sequência, continuou a contagem até 9 após a desativação da entrada LOAD. E por fim, no terceiro cenário, foi feito novamente um LOAD do número 7, seguido pela ativação da entrada RESET. O LOAD fez com que o contador transitasse para o estado BCD_7 porém na próxima borda de subida imediata do CLOCK, a operação de RESET fez com que o contador transitasse e permanecesse no estado BCD_0, até a desativação das entradas RESET e LOAD, fazendo com que o contador retornasse a função de contagem, com a transição do estado BCD_0 para o estado BCD_1.

Atividade 2

Implemente em VHDL e simule no Modelsim um contador BCD módulo 100. Este contador deve ser implementado usando apenas os contadores módulo 10 desenvolvidos na atividade anterior como componentes. Este contador terá duas saídas de 4 bits: Q_UNIDADE e Q_DEZENA, que representam, respectivamente, a unidade e a dezena do número atual da contagem BCD. As entradas são:

- CLOCK: é o clock do sistema. As transições de estado acontecem apenas na borda de subida do clock (inclusive transições causadas pelo RESET e LOAD
- RESET: quando igual a 1, o contador deve retornar para 0
- ENABLE: ativa em nível baixo. A contagem acontece apenas se ENABLE = 0
- D_UNIDADE e D_DEZENA: entradas de 4 bits cada
- LOAD: se LOAD = 1, o contador deve fazer Q_UNIDADE = D_UNIDADE e Q_DEZENA = D_DEZENA (de forma síncrona)

Se tanto a entrada LOAD quanto a entrada RESET estiverem no nível alto, RESET tem prioridade. Tanto RESET quanto LOAD têm prioridade sobre a contagem.

A segunda atividade envolve o uso do contador BCD módulo 10 em cascata, similar ao que foi feito no experimento 5 com os somadores completos. O uso dos contadores em cascata nos permite representar números maiores, sendo cada contador responsável por armazenar um digito do número. O circuito proposto terá a seguinte configuração:

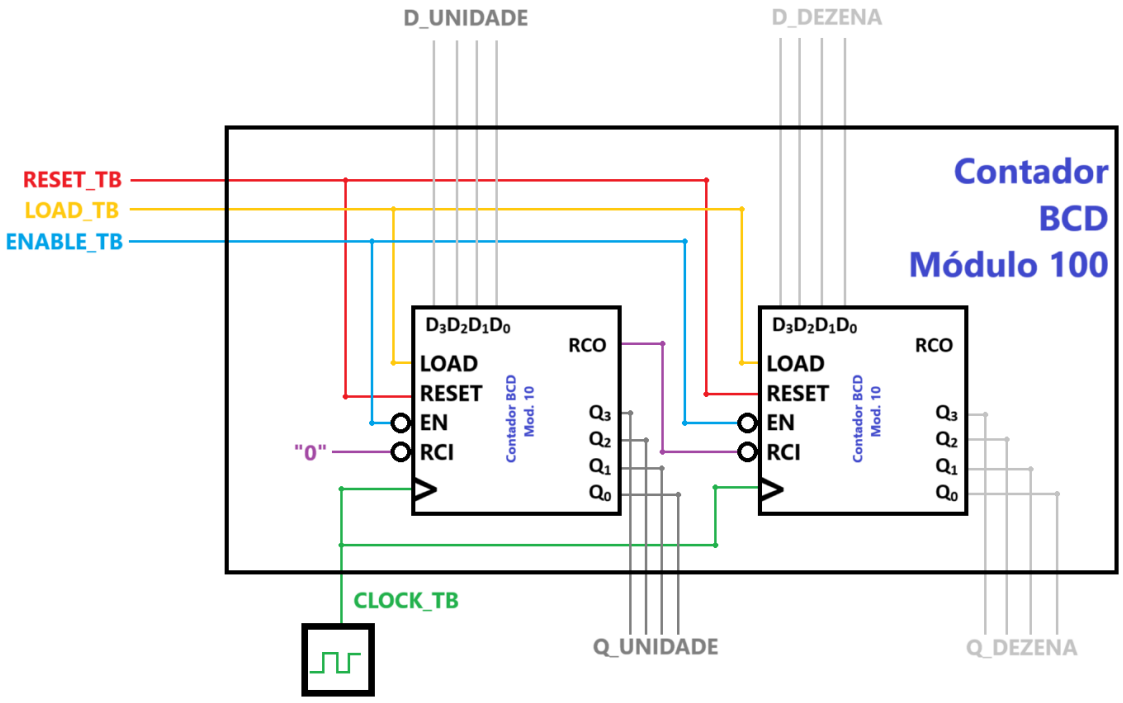

Figura 6:Diagrama das conexões dos dois contadores BCD Módulo 10 em cascata para formar o Contador BCD Módulo 100

Dentro do contador BCD módulo 100 há dois contadores BCD módulo 10 que compartilham as entradas CLOCK, LOAD, RESET e ENABLE. A entrada RCI do primeiro contador, responsável pelo digito da unidade, deve ficar sempre ativa. Já a entrada RCI do segundo

contador, responsável pelo digito da dezena, está ligada a saída RCO do primeiro contador. Conforme a tabela 2, a saída RCO estará em baixo nível apenas quando o contador estiver no estado BCD_9 que representa o dígito 9. Por conta disso, o contador do dígito das dezenas irá realizar a operação de contagem apenas nas bordas de subida do clock quando o primeiro contador estiver no estado BCD_9 – isto é, quando o primeiro contador completar uma dezena. As entradas de carga paralela D_UNIDADE e D_DEZENA estão ligadas ao contador do dígito da unidade e ao contador do dígito da dezena, respectivamente, assim como as saídas Q_UNIDADE e Q_DEZENA. O contador BCD módulo 100 foi modelado através do seguinte código:

```
01  -- Experimento 08 Atividade 1
02  -- Aluno: Pedro Henrique Dias Avelar 241037112
03  -- Turma 08
04  -- Data: 05/02/2025
05
06  -- Contador BCD Modulo 100
07
08  LIBRARY IEEE;
09  USE IEEE.STD_LOGIC_1164.ALL;
10  USE WORK.TYPE_ESTADO_BCD.ALL;
11
12  ENTITY CONTADOR_BCD_MOD100 IS
13      PORT (
14          CLOCK: IN STD_LOGIC;
15          RESET: IN STD_LOGIC;
16          ENABLE: IN STD_LOGIC;
17          D_UNIDADE: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
18          D_DEZENA: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
19          LOAD: IN STD_LOGIC;
20          Q_UNIDADE: OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
21          Q_DEZENA: OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
22          CURRENT_STATE_UNIDADE: OUT ESTADO_BCD;
23          CURRENT_STATE_DEZENA: OUT ESTADO_BCD
24          );
25  END CONTADOR_BCD_MOD100;
26
27  ARCHITECTURE ARC_CONTADOR_BCD_MOD100 OF CONTADOR_BCD_MOD100 IS
28  --CONTADOR BCD10 DA QUESTAO 1
29  COMPONENT CONTADOR_BCD_MOD10 IS
30      PORT (
31          CLOCK    : IN STD_LOGIC;
32          RESET    : IN STD_LOGIC;
33          ENABLE   : IN STD_LOGIC;
34          RCI      : IN STD_LOGIC;
35          D        : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
36          LOAD     : IN STD_LOGIC;
37          Q        : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
38          RCO      : OUT STD_LOGIC;
39          CURRENT_STATE: OUT ESTADO_BCD
40          );
41  END COMPONENT;
42  SIGNAL RCO_UNIDADE: STD_LOGIC;
43  SIGNAL RCI_DEZENA: STD_LOGIC;
44  SIGNAL RCO_DEZENA: STD_LOGIC;
```

```
45  BEGIN
46  BCD_UNIDADE: CONTADOR_BCD_MOD10 PORT MAP (CLOCK, RESET,ENABLE,
'0', D_UNIDADE, LOAD, Q_UNIDADE,RCO_UNIDADE, CURRENT_STATE_UNIDADE);
47  BCD_DEZENA: CONTADOR_BCD_MOD10 PORT MAP (CLOCK, RESET,ENABLE,
RCI_DEZENA, D_DEZENA, LOAD, Q_DEZENA, RCO_DEZENA,
CURRENT_STATE_DEZENA);
48  RCI_DEZENA <= RCO_UNIDADE;
49  END ARC CONTADOR BCD MOD100;
```

Código 4:Modelagem do contador BCD módulo 100

Mais uma vez nas linhas 22 e 23 do código 4 foi implantada duas saídas do tipo ESTADO_BCD de modo a facilitar a visualização dos resultados da simulação. Na linha 42 foi definido o sinal RCO_UNIDADE que faz a ligação entre a saída RCO do contador do dígito de unidade com a entrada RCI do contador do dígito de dezena. O sinal RCO_DEZENA não faz nenhuma ligação – ele seria necessário apenas se quiséssemos adicionar mais um dígito para formar um contador BCD módulo 1000. Como representado na figura 6, os dois contadores módulo 10 compartilham as entradas CLOCK, RESET, ENABLE e LOAD.

```
01  -- Experimento 08 - TESTBENCH Q2
02  -- Aluno: Pedro Henrique Dias Avelar 241037112
03  -- Turma 08
04  -- Data: 05/02/2025
05
06  -- Testbench - Tempo de simulação: 1110 NS
07
08  LIBRARY IEEE;
09  USE IEEE.STD_LOGIC_1164.ALL;
10  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
11  USE IEEE.NUMERIC_STD.ALL;
12  USE WORK.TYPE_ESTADO_BCD.ALL;
13
14  ENTITY TESTBENCH_E08_BCD_MOD100 IS
15  END TESTBENCH_E08_BCD_MOD100;
16
17  ARCHITECTURE ARC_TESTBENCH_E08_BCD_MOD100 OF
TESTBENCH_E08_BCD_MOD100 IS
18
19  COMPONENT CONTADOR_BCD_MOD100 IS
20      PORT (
21          CLOCK: IN STD_LOGIC;
22          RESET: IN STD_LOGIC;
23          ENABLE: IN STD_LOGIC;
24          D_UNIDADE: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
25          D_DEZENA: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
26          LOAD: IN STD_LOGIC;
27          Q_UNIDADE: OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
28          Q_DEZENA: OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
29          CURRENT_STATE_UNIDADE: OUT ESTADO_BCD;
30          CURRENT_STATE_DEZENA: OUT ESTADO_BCD
31          );
32  END COMPONENT;
33
34  SIGNAL CLOCK_TB: STD_LOGIC := '0';
35  SIGNAL RESET_TB, ENABLE_TB, LOAD_TB: STD_LOGIC;
36  SIGNAL D_UNIDADE_TB, D_DEZENA_TB: STD_LOGIC_VECTOR(3 DOWNTO 0);
37  SIGNAL Q_UNIDADE_TB, Q_DEZENA_TB: STD_LOGIC_VECTOR(3 DOWNTO 0);
38  SIGNAL CURRENT_STATE_UNIDADE_TB, CURRENT_STATE_DEZENA_TB:
ESTADO BCD;
```

```
39
40  BEGIN
41      DUT: CONTADOR_BCD_MOD100 PORT MAP (
42          CLOCK_TB, RESET_TB, ENABLE_TB,D_UNIDADE_TB, D_DEZENA_TB,
43          LOAD_TB, Q_UNIDADE_TB, Q_DEZENA_TB,
CURRENT_STATE_UNIDADE_TB, CURRENT_STATE_DEZENA_TB
44      );
45      CLOCK_TB <= NOT CLOCK_TB AFTER 5 NS;
46      PROCESS
47      BEGIN
48      REPORT "INICIANDO TESTE..." SEVERITY NOTE;
49      -- CENARIO 1: CONTAR DE 0 A 99;
50      RESET_TB <= '0'; LOAD_TB <= '0'; ENABLE_TB <= '0';
51      WAIT FOR 990 NS;
52       -- CENARIO 2: LOAD DOS NUMEROS 19, 28, 37, 46, 55, 64, 73, 82,
91
53       RESET_TB <= '0'; LOAD_TB <= '1';
54       D_DEZENA_TB <= "0001"; D_UNIDADE_TB <= "1001"; WAIT FOR 10 NS;
55       D_DEZENA_TB <= "0010"; D_UNIDADE_TB <= "1000"; WAIT FOR 10 NS;
56       D_DEZENA_TB <= "0011"; D_UNIDADE_TB <= "0111"; WAIT FOR 10 NS;
57       D_DEZENA_TB <= "0100"; D_UNIDADE_TB <= "0110"; WAIT FOR 10 NS;
58       D_DEZENA_TB <= "0101"; D_UNIDADE_TB <= "0101"; WAIT FOR 10 NS;
59       D_DEZENA_TB <= "0110"; D_UNIDADE_TB <= "0100"; WAIT FOR 10 NS;
60       D_DEZENA_TB <= "0111"; D_UNIDADE_TB <= "0011"; WAIT FOR 10 NS;
61       D_DEZENA_TB <= "1000"; D_UNIDADE_TB <= "0010"; WAIT FOR 10 NS;
62       D_DEZENA_TB <= "1001"; D_UNIDADE_TB <= "0001"; WAIT FOR 10 NS;
63       -- CENARIO 3: COM A ENTRADA LOAD ATIVA, ATIVAR RESET
64       RESET_TB <= '1'; WAIT FOR 30 NS;
65      REPORT "TESTE FINALIZADO!" SEVERITY NOTE;
66      WAIT;
67      END PROCESS;
68  END ARC_TESTBENCH_E08_BCD_MOD100;
```

Código 5:Testbench para o contador BCD módulo 100

O teste teve uma duração total de 1110 ns e gerou o seguinte resultado:

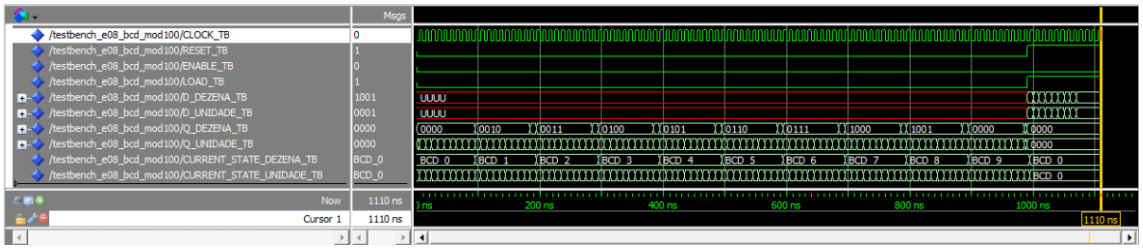

Figura 7:Resultado do testbench para o contador BCD módulo 100

O teste consistiu em 3 cenários. O primeiro cenário envolveu a contagem de zero a 99.

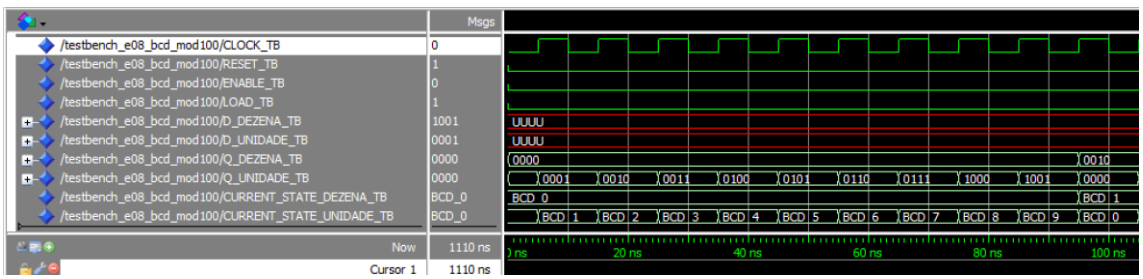

Figura 8:Contagem de 0 a 10

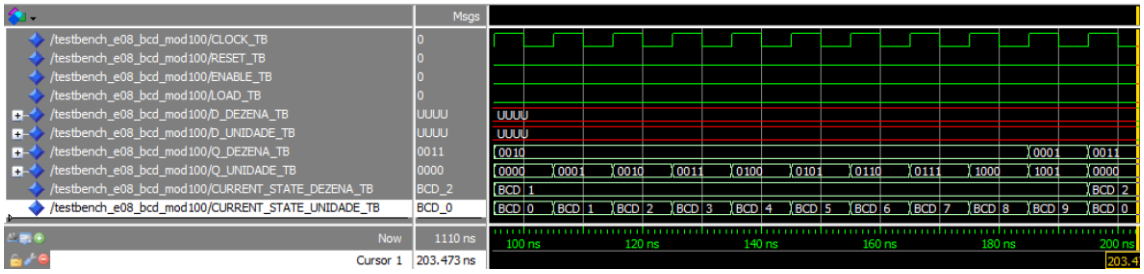

Figura 9:Contagem de 10 a 20

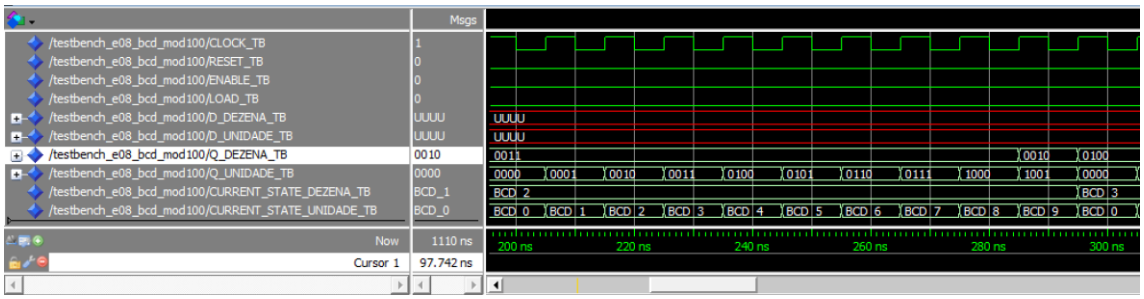

Figura 10:Contagem de 20 a 30

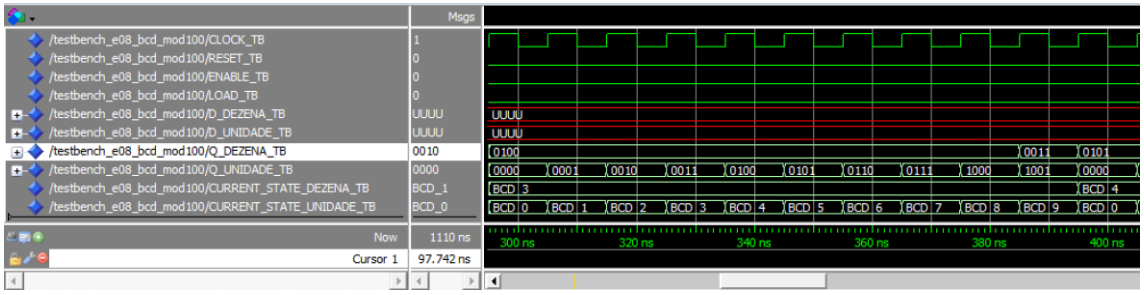

Figura 11:Contagem de 30 a 40

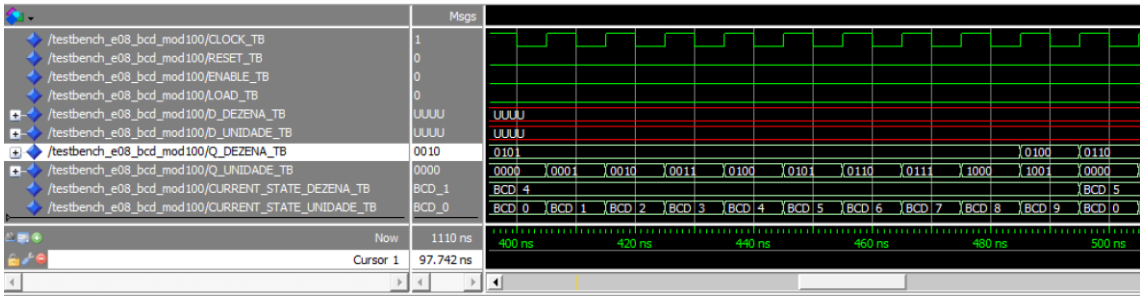

Figura 12:Contagem de 40 a 50

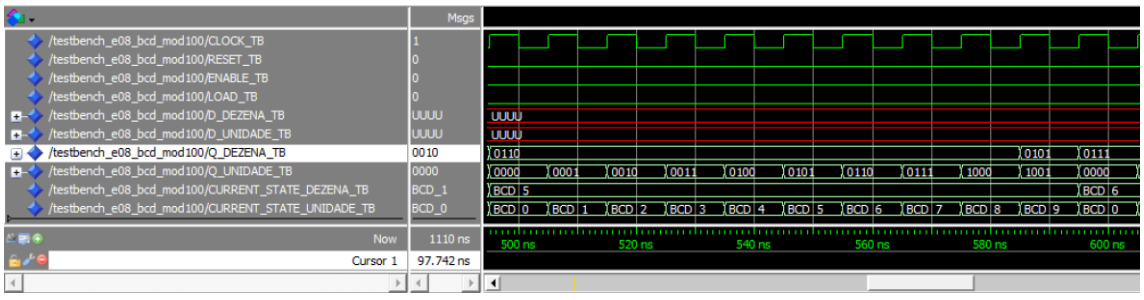

Figura 13:Contagem de 50 a 60

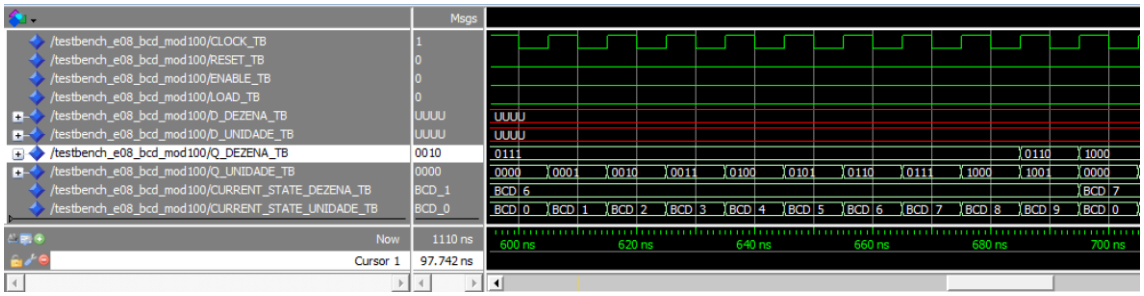

Figura 14:Contagem de 60 a 70

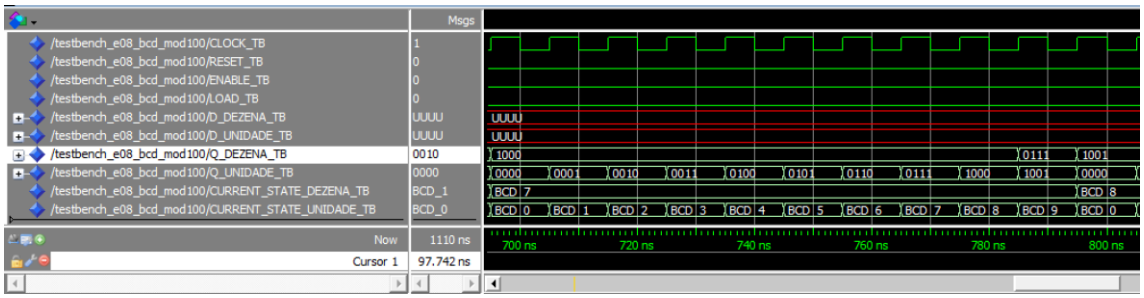

Figura 15:Contagem de 70 a 80

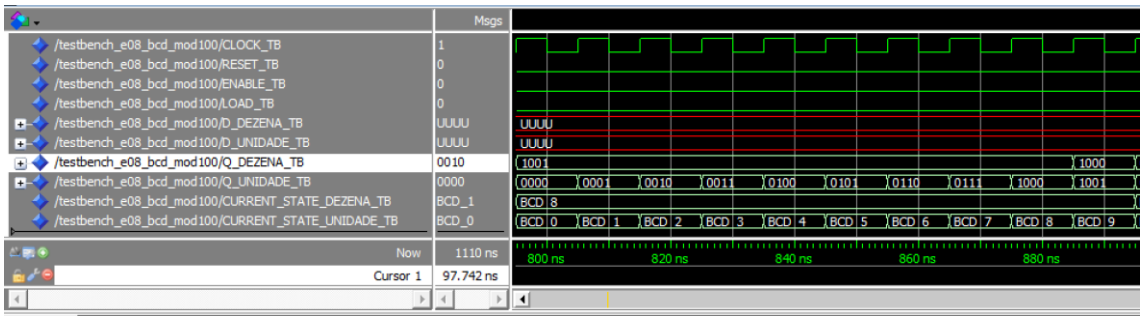

Figura 16:Contagem de 80 a 90

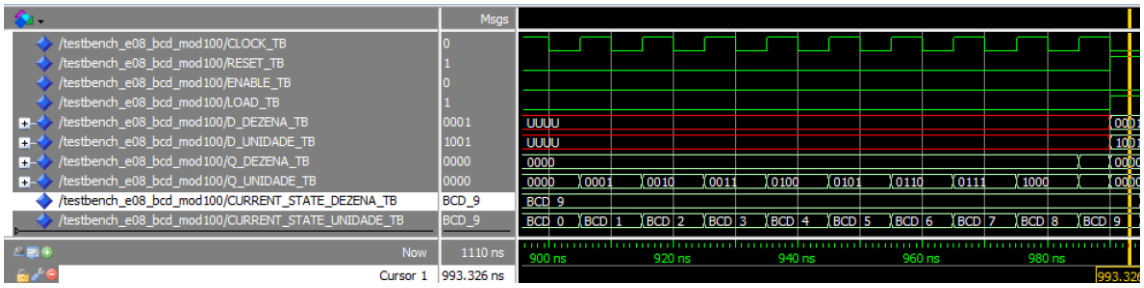

Figura 17:Contagem de 90 a 99

O segundo cenário envolveu a operação de LOAD para os seguintes números em sequência: 19, 28, 37, 46, 55, 64, 73, 82 e 91. Por fim após essa sequência, o cenário 3 envolveu acionar a entrada RESET com a entrada LOAD ainda ativa. Como RESET tem preferência, ambos os contadores voltaram e permaneceram no estado BCD_0.

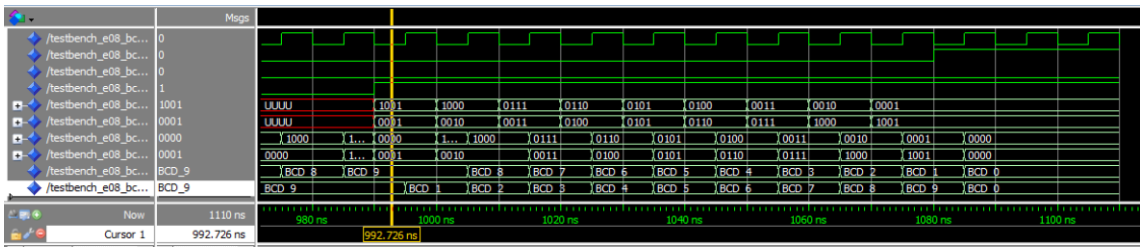

Figura 18:Resultado dos cenários 2 e 3 de teste