Laboratório de Sistemas Digitais – 2025.1

Relatório 7 – Máquina de Estados de Moore

João Pedro Fernandes Santos (222025342)

Turma 09

Introdução/Objetivos

Neste Experimento 7 de Sistemas Digitais, projetamos e implementamos em VHDL uma Máquina de Estados do tipo Moore para controlar uma máquina de vendas de moeda única, capaz de receber incrementos de 25 e 50 centavos, liberar o produto e devolver troco quando o montante atinge ou ultrapassa R$ 1,00. Inicialmente, elaboramos o diagrama de estados considerando os sete estados necessários (ZERO, VINTE_CINCO, CINQUENTA, SETENTA_CINCO, CEM, TROCO25 e TROCO50) e as transições associadas às entradas “01” (25 ¢), “10” (50 ¢), “11” (cancelamento) e “00” (ação neutra). Em seguida, implementamos a entidade e a arquitetura separando o processo síncrono de atualização de estado do processo combinacional de cálculo de próxima etapa e sinais de saída.

Para validar o funcionamento, construímos um testbench que exerce todas as operações da máquina: acumulação de moedas até a liberação do produto, devolução de troco em diferentes cenários e cancelamento em quaisquer estados intermediários. O testbench automatiza a aplicação de sequências de entradas variadas, compara saídas simuladas com o comportamento esperado e permite a inspeção detalhada das formas de onda. Com garantimos a correção funcional do projeto.

1. Atividade 1

1.1 Descrição da atividade

Nesta etapa, implementamos em VHDL a máquina de vendas conforme o roteiro e validamos seu funcionamento por simulação no ModelSim. A primeira tarefa foi traduzir o diagrama de estados em código, criando a entidade com relógio, vetor de entrada de duas linhas e três sinais de saída. Em seguida, separamos a lógica em dois processos: um sensível à borda de subida do clock para atualizar o estado atual, e outro puramente combinacional para determinar o próximo estado e acionar as saídas de liberação ou devolução de troco.

Com o componente pronto, desenvolvemos um testbench que percorre todos os cenários previstos:

- Acumulação de moedas até atingir ou exceder R$ 1,00, verificando a emissão de LIBERA_PRODUTO e o cálculo de troco adicional.

- Cancelamento em estados intermediários, conferindo a devolução correta de 25 ¢ ou 50 ¢.
- Sequência de duas moedas de 50 ¢, checando liberação e retorno de 50 ¢.

Cada condição foi acompanhada por assertions que comparam o comportamento simulado com o resultado esperado. A análise das formas de onda confirmou a conformidade de cada transição e garantiu a robustez do circuito antes de sua entrega.

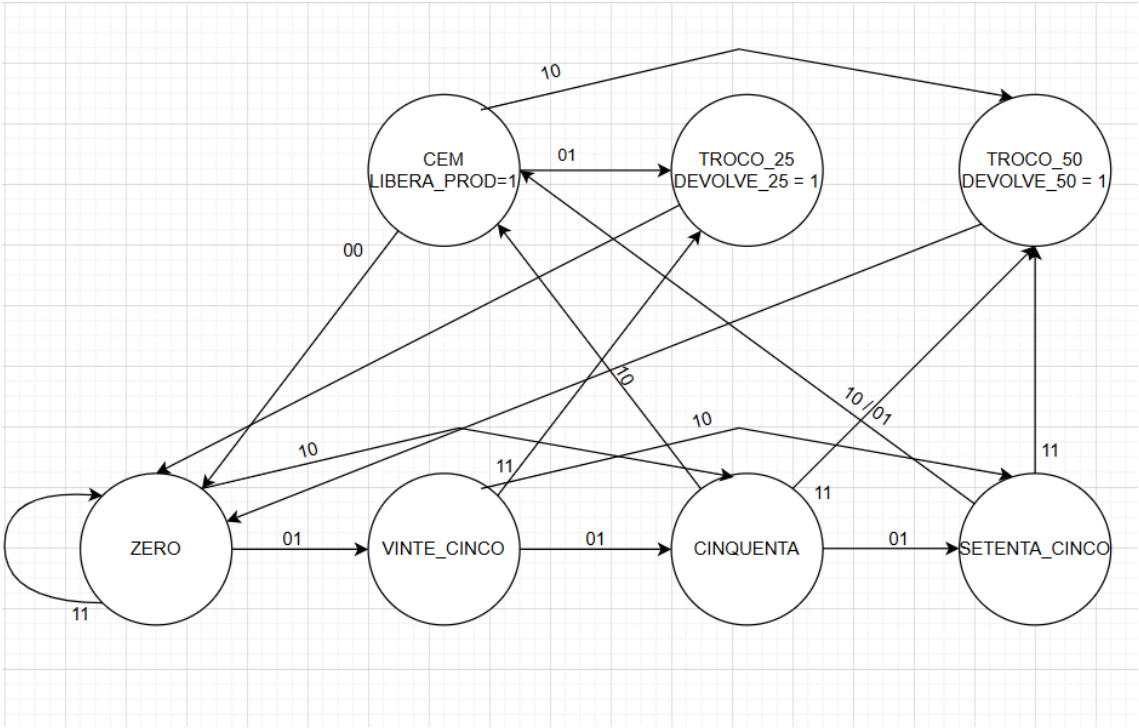

Figura 1: Diagrama de estados que descreve o funcionamento da maquina de vendas.

1.2 Implementação em VHDL

O código da máquina de vendas está estruturado de forma a separar claramente a lógica de controle de estados da geração de saídas. Logo na entidade, definem-se as portas de entrada e saída: um relógio (CLK), um vetor de 2 bits que indica a ação do usuário (inserir 25 ¢, 50 ¢, cancelar ou manter) e três sinais de saída que sinalizam liberação do produto ou devolução de troco.

Internamente, a arquitetura declara um tipo enumerado com sete estados correspondentes aos valores acumulados (ZERO, VINTE_CINCO, CINQUENTA, SETENTA_CINCO, CEM) e aos dois estados de troco (TROCO25, TROCO50). Dois sinais, ATUAL e PROXIMO, guardam o estado corrente e o próximo estado.

1. No processo sensível a CLK, a cada borda de subida copia-se PROXIMO para ATUAL, avançando a máquina para o próximo estado calculado.

2. No processo combinacional, sensível a mudanças de ATUAL ou de ENTRADA, todas as saídas são zeradas e assume-se inicialmente que PROXIMO permanece igual a ATUAL. Em seguida, um CASE principal escolhe o bloco de transições correspondente ao estado corrente, e dentro de cada bloco um segundo CASE interpreta o vetor de entrada:
    - Nos estados de soma (ZERO, VINTE_CINCO, CINQUENTA, SETENTA_CINCO), as entradas "01" e "10" fazem o avanço graduado dos valores, "11" dispara cancelamento para devolver o valor acumulado e "00" mantém o estado.
    - No estado CEM, além de ativar LIBERA_PRODUTO, o mesmo CASE combinacional decide se retorna a ZERO (sem nova moeda) ou entra em um dos estados de troco, conforme entrada extra.
    - Nos estados TROCO25 e TROCO50, o processo ativa o sinal de devolução correspondente e força PROXIMO a ZERO no ciclo seguinte.

Dessa maneira, toda a lógica de transição de estados e de geração de sinais de liberação ou devolução está contida em um único bloco combinacional, enquanto o bloco síncrono garante a atualização ordenada do estado.

O testbench cria um ambiente de simulação que gera automaticamente um clock periódico e aplica diversas sequências em ENTRADA para cobrir todos os casos: acumulação de moedas até liberar o produto, inserção extra para troco, cancelamentos em diferentes pontos e duplas de 50 ¢ para verificar devolução. Após cada cenário, asserts conferem se as saídas do DUT (unidade em teste) correspondem ao comportamento esperado; caso alguma discrepância ocorra, uma mensagem de erro é disparada. Ao final, um relatório indica conclusão dos testes. Essa abordagem garante verificação completa, sem intervenção manual, e fornece feedback imediato sobre a conformidade da implementação

A Listagem 1 e 2 apresentam o código criado para a execução desta atividade e do teste por meio do arquivo testbench, respectivamente.

```
LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;

ENTITY MAQUINA_VENDAS IS

PORT(

  CLK  : IN  STD_LOGIC;

  ENTRADA      : IN  STD_LOGIC_VECTOR(1 DOWNTO 0);

  LIBERA_PRODUTO : OUT STD_LOGIC;

  DEVOLVE_25    : OUT STD_LOGIC;

  DEVOLVE_50    : OUT STD_LOGIC );

END MAQUINA_VENDAS;

ARCHITECTURE MAQUINA_VENDAS_ARCH OF
MAQUINA_VENDAS IS

TYPE ESTADO_TYPE IS (

  ZERO   , VINTE_CINCO,    CINQUENTA,
SETENTA_CINCO,  CEM,   TROCO25,   TROCO50 );

SIGNAL ATUAL, PROXIMO : ESTADO_TYPE;
```

```
BEGIN

 PROCESS(CLK)

  BEGIN

   IF rising_edge(CLK) THEN

    ATUAL <= PROXIMO;

   END IF;

  END PROCESS;

PROCESS(ATUAL, ENTRADA)

 BEGIN

  LIBERA_PRODUTO <= '0';

  DEVOLVE_25    <= '0';

  DEVOLVE_50    <= '0';

  PROXIMO      <= ATUAL;

CASE ATUAL IS
```

```
    WHEN ZERO =>
      CASE ENTRADA IS
       WHEN "01" => PROXIMO <= VINTE_CINCO;
       WHEN "10" => PROXIMO <= CINQUENTA;
       WHEN "11" => PROXIMO <= ZERO;
       WHEN OTHERS => PROXIMO <= ZERO;
      END CASE;
WHEN VINTE_CINCO =>
      CASE ENTRADA IS
       WHEN "01" => PROXIMO <= CINQUENTA;
       WHEN "10" => PROXIMO <= SETENTA_CINCO;
       WHEN "11" => PROXIMO <= TROCO25;
       WHEN OTHERS => PROXIMO <= VINTE_CINCO;
      END CASE;
WHEN CINQUENTA =>
      CASE ENTRADA IS
       WHEN "01" => PROXIMO <= SETENTA_CINCO;
       WHEN "10" => PROXIMO <= CEM;
       WHEN "11" => PROXIMO <= TROCO50;
       WHEN OTHERS => PROXIMO <= CINQUENTA;
      END CASE;
```

```
 WHEN SETENTA_CINCO =>
      CASE ENTRADA IS
       WHEN "01" => PROXIMO <= CEM;
       WHEN "10" => PROXIMO <= CEM;
       WHEN "11" => PROXIMO <= TROCO50;
      WHEN OTHERS => PROXIMO <=
SETENTA_CINCO;
      END CASE;
WHEN CEM =>
      LIBERA_PRODUTO <= '1';
      CASE ENTRADA IS
       WHEN "10" => PROXIMO <= TROCO50;
       WHEN "01" => PROXIMO <= TROCO25;
       WHEN OTHERS => PROXIMO <= ZERO;
      END CASE;
WHEN TROCO25 =>
      DEVOLVE_25 <= '1';
      PROXIMO    <= ZERO;
WHEN TROCO50 =>
      DEVOLVE_50 <= '1';
      PROXIMO    <= ZERO;
END CASE;
 END PROCESS;
END MAQUINA_VENDAS_ARCH;
```

Listagem 1: Implementação do circuito da Máquina de vendas em Vhdl.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;
ENTITY TB_MAQUINA_VENDAS IS
END ENTITY;
ARCHITECTURE BEHAVIOR OF
TB_MAQUINA_VENDAS IS
COMPONENT MAQUINA_VENDAS
   PORT(CLK  : IN  STD_LOGIC;
    ENTRADA      : IN  STD_LOGIC_VECTOR(1
DOWNTO 0);
    LIBERA_PRODUTO : OUT STD_LOGIC;
```

```
 DEVOLVE_25    : OUT STD_LOGIC;
    DEVOLVE_50    : OUT STD_LOGIC
 );
  END COMPONENT;
  SIGNAL CLK_TB         : STD_LOGIC := '0';
  SIGNAL ENTRADA_TB       : STD_LOGIC_VECTOR(1
DOWNTO 0) := (OTHERS => '0');
  SIGNAL LIBERA_PRODUTO_TB : STD_LOGIC;
  SIGNAL DEVOLVE_25_TB    : STD_LOGIC;
  SIGNAL DEVOLVE_50_TB    : STD_LOGIC;
 CONSTANT CLK_PERIOD : TIME := 20 NS;
```

```
BEGIN
 UUT: MAQUINA_VENDAS
  PORT MAP (
CLK         => CLK_TB,
    ENTRADA      => ENTRADA_TB,
    LIBERA_PRODUTO => LIBERA_PRODUTO_TB,
    DEVOLVE_25    => DEVOLVE_25_TB,
    DEVOLVE_50    => DEVOLVE_50_TB  );
CLK_PROCESS: PROCESS
 BEGIN
  CLK_TB <= '0';
  WAIT FOR CLK_PERIOD/2;
  CLK_TB <= '1';
  WAIT FOR CLK_PERIOD/2;
 END PROCESS;
STIM_PROC:
PROCESS
 BEGIN
  WAIT FOR 2*CLK_PERIOD;
ENTRADA_TB <= "01";
  WAIT FOR CLK_PERIOD;
  ENTRADA_TB <= "10";
  WAIT FOR CLK_PERIOD;
  ENTRADA_TB <= "01";
  WAIT FOR CLK_PERIOD;
  ENTRADA_TB <= "01";
  WAIT FOR CLK_PERIOD;
ASSERT LIBERA_PRODUTO_TB = '1'
    REPORT
"FALHA: PRODUTO NAO LIBERADO QUANDO TOTAL =
100" SEVERITY ERROR;
  ASSERT DEVOLVE_25_TB = '1'
    REPORT "FALHA: TROCO 25 NAO DEVOLVIDO"
SEVERITY ERROR;
  ENTRADA_TB <= "01";
  WAIT FOR CLK_PERIOD;
  ENTRADA_TB <= "11";
  WAIT FOR CLK_PERIOD;
ASSERT DEVOLVE_25_TB = '1'
    REPORT "FALHA: TROCO 25 NAO DEVOLVIDO
NO CANCELAMENTO" SEVERITY ERROR;
ENTRADA_TB <= "10";
  WAIT FOR CLK_PERIOD;
  ENTRADA_TB <= "10";
  WAIT FOR CLK_PERIOD;
  ENTRADA_TB <= "11";
  WAIT FOR CLK_PERIOD;
ASSERT DEVOLVE_50_TB = '1'
    REPORT "FALHA: TROCO 50 NAO DEVOLVIDO
NO CANCELAMENTO" SEVERITY ERROR;
ENTRADA_TB <= "10";
  WAIT FOR CLK_PERIOD;
  ENTRADA_TB <= "10";
  WAIT FOR CLK_PERIOD;
ASSERT LIBERA_PRODUTO_TB = '1'
    REPORT "FALHA: PRODUTO NAO LIBERADO
APOS 50+50" SEVERITY ERROR;
  ASSERT DEVOLVE_50_TB = '1'
    REPORT "FALHA: TROCO 50 NAO DEVOLVIDO
APOS LIBERACAO" SEVERITY ERROR;
ENTRADA_TB <= "00";
  WAIT FOR 2*CLK_PERIOD;
REPORT "FIM DO TESTBENCH: TODOS OS TESTES
COMPLETADOS." SEVERITY NOTE;
  WAIT;
END PROCESS;
END ARCHITECTURE;
```

Listagem 2: Implementação do circuito do Testbench da Máquina de vendas em Vhdl.

1.3 Simulação

Para verificar o funcionamento da Máquina de Vendas, construímos um testbench que gera um clock de 20 ns de período (10 ns em ‘0’, 10 ns em ‘1’) e aplica automaticamente as sequências de entrada para cobrir todos os cenários previstos. O procedimento foi:

1. Inicialização
   Mantivemos ENTRADA = "00" por duas etapas de clock (40 ns) para garantir que o estado interno começasse em ZERO.
2. Venda até R$ 1,00 e liberação de produto
   - Em t = 40 ns, definimos ENTRADA = "01" (25 ¢).
   - Em t = 60 ns, passamos para ENTRADA = "10" (50 ¢).
   - Em t = 80 ns, voltamos a ENTRADA = "01" (25 ¢), totalizando R$ 1,00.
   - Logo após essa borda de clock, usamos WAIT UNTIL libera_produto = '1', aguardando até que a saída de liberação vá a ‘1’.
   - Um ASSERT confirma que o produto foi liberado corretamente.
3. Troco extra de 25 ¢
   - Com o estado em CEM, atribuímos ENTRADA = "01" novamente.
   - Em seguida, WAIT UNTIL devolve_25 = '1' faz o testbench pausar até que a máquina entre no estado TROCO25 e ative a saída de devolução.
   - Um ASSERT verifica a devolução de 25 ¢.
4. Cancelamento em 25 ¢
   - Retornamos a ZERO e, após inserir 25 ¢ (ENTRADA = "01"), mudamos para ENTRADA = "11" (cancelamento).
   - Novamente, usamos WAIT UNTIL devolve_25 = '1' para sincronizar com a ativação do troco, seguido de um ASSERT.
5. Cancelamento em 50 ¢
   - Inserimos 50 ¢ (ENTRADA = "10"), depois outro 50 ¢ para chegar a CINQUENTA, e em seguida ENTRADA = "11".
   - O testbench espera WAIT UNTIL devolve_50 = '1' para capturar o pulso de troco de 50 ¢ e usa ASSERT para confirmar.
6. Venda com duas moedas de 50 ¢
   - Inserimos “10” duas vezes, alcançando R$ 1,00.
   - Aguardamos WAIT UNTIL libera_produto = '1' para verificar a liberação, depois WAIT UNTIL devolve_50 = '1' para capturar o troco de 50 ¢.

- Dois ASSERT verificam ambos os sinais.

Na janela de formas de onda do ModelSim, vê-se claramente que cada WAIT UNTIL sincroniza exatamente com o momento em que a saída relevante salta para ‘1’, dispensando esperas arbitrárias e garantindo cobertura completa dos casos de venda, troco e cancelamento.

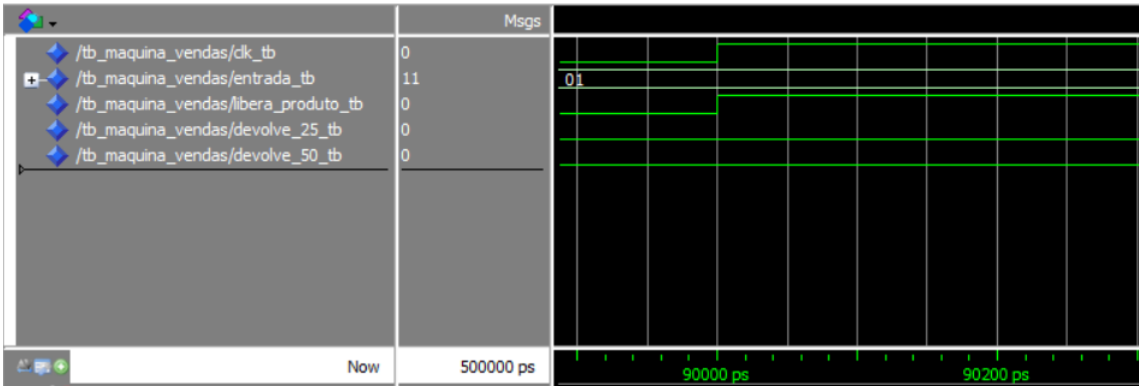

Figura 1: Simulação do Testbench da máquina de Vendas, ilustrando o item (2) da explicação da simulação, a liberação do produto na subida do clock.

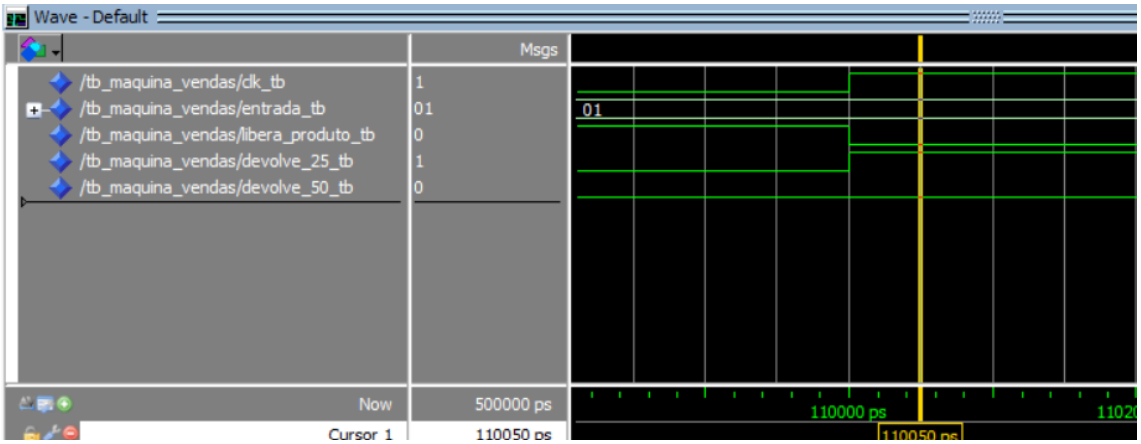

Figura 2: Simulação do Testbench da máquina de Vendas, ilustrando o item (3) da explicação da simulação, a devolução de 25 centavos na subida do clock.