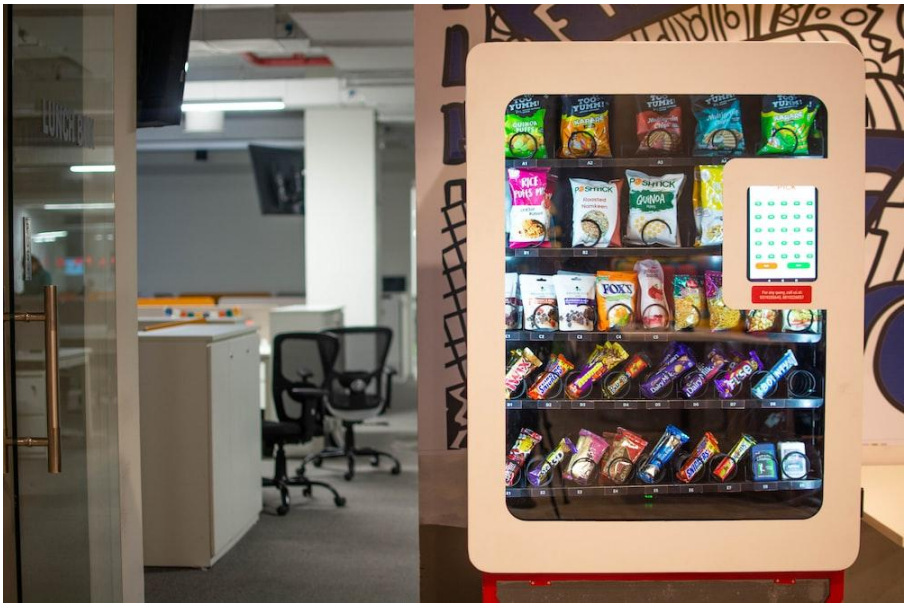
Trabalho Prático - Laboratório de Sistemas Digitais
Parte 2 do projeto RTL

Maria Eduarda Sampaio e Raíssa Gonçalves Diniz
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte - MG - Brasil

1. Introdução:

No ambiente comercial contemporâneo, a busca por soluções que aliem eficiência e praticidade tem impulsionado a adoção de sistemas automatizados em diversos setores. As vending machines, ou máquinas de venda automática, são uma expressão clara dessa tendência, representando um ponto de interseção entre conveniência e tecnologia. No contexto brasileiro, onde o cotidiano agitado demanda rapidez e simplicidade, essas máquinas podem oferecer um serviço valioso, diminuindo filas e proporcionando acesso imediato a produtos diversos, como bebidas.



Dessa maneira, ao longo deste relatório, propomos a criação de uma vending machine, um sistema automatizado de venda de bebidas que opera com eficiência e conveniência para o usuário. Utilizando um conjunto de entradas digitais, o cliente pode escolher e confirmar um pedido, cancelar a operação ou inserir dinheiro. Com base nas ações do usuário, a máquina responde com saídas correspondentes, que incluem a devolução de troco, a liberação do produto selecionado, e uma série de mensagens do sistema que guiam o usuário durante o processo de compra. Estas mensagens variam desde a seleção do produto até a confirmação do pedido, cancelamento, instruções para inserção de dinheiro, e para retirada do troco / produto. Este sistema automatizado busca

maximizar a eficiência da transação de venda, proporcionando uma experiência de compra fácil e intuitiva para o consumidor.

Ao tratar das etapas de projeção desse sistema, este relatório contribuiu imensamente para uma compreensão mais ampla de como os sistemas digitais podem ser otimizados para melhor servir aos seus usuários finais.

Caso a visualização de alguns dos diagramas não estiver clara, você pode checá-los em:
https://drive.google.com/file/d/15vPjWUDtAc2otTtf3cOVCS12G5ZS5pbB/view?usp=sharing

2. Funcionamento:

Nossa vending machine é projetada para fornecer um método automatizado de compra de bebidas que é gerenciado por uma Máquina de Estados Finitos (FSM). O processo inicia quando a máquina exibe uma mensagem, solicitando ao usuário que escolha um produto. Isso é feito ao receber uma entrada digital “Produto_selecionado” que permite ao usuário selecionar o item desejado.

Após o usuário ter escolhido seu produto, a máquina passa para o estado de validação do mesmo, isto é, verificar se o produto escolhido existe no sistema. Apesar de o número de produtos ser um valor de 3 bits, a máquina só possui 7 produtos (com os respectivos valores e preços de representados pelos valores de 1 a 7), pois o valor “000” é atribuído a um produto inválido. Caso seja inválido, a máquina aguarda a entrada de um valor válido, e só depois passa para o próximo estado.

Uma vez validado, a máquina passa para o estado de confirmação, onde espera uma entrada “Confirma” para confirmar o pedido. Se o usuário decidir cancelar a compra, um sinal “Confirma'” será enviado, que fará a máquina retornar ao estado inicial e emitir uma mensagem indicando que o pedido foi cancelado.

Se o pedido for confirmado, o próximo passo é a inserção do pagamento. A máquina exibe uma mensagem instruindo o usuário a inserir o dinheiro, processo que é monitorado pela entrada 'Dinheiro'. Se o valor da bebida for maior do que o valor inserido, a máquina fornece o troco e a bebida. Caso o valor fornecido seja menor do que o valor da bebida, a máquina fica em espera até que o valor seja, no mínimo, igual ao da bebida escolhida. Por fim, se o valor inserido for igual ao preço da bebida escolhida, ela avança para o próximo estágio.

Após a inserção e validação do dinheiro, a máquina ativa a saída 'Produto_liberado' para liberar o produto e 'Troco' para retornar o troco, se houver. Mensagens correspondentes são exibidas para orientar o usuário a retirar tanto o troco quanto o produto.

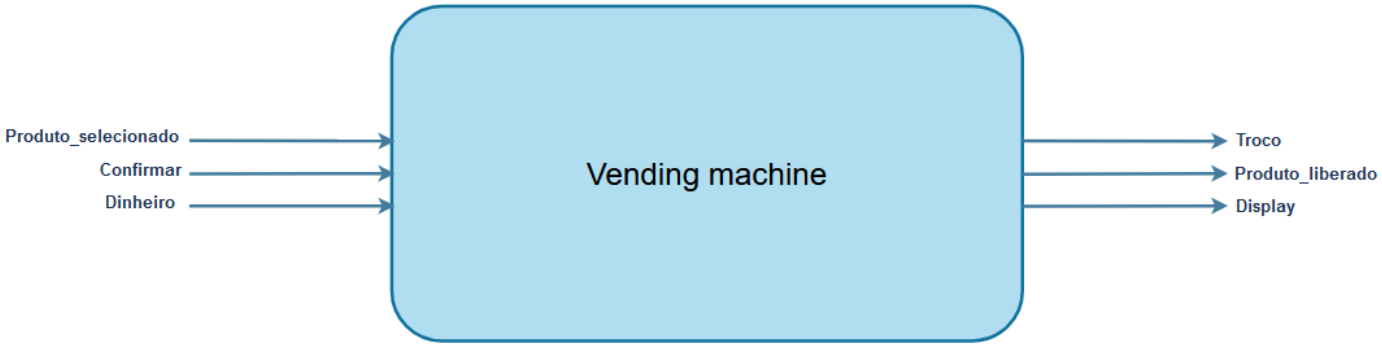
O dispositivo emprega um sistema de mensagens 'Display' para comunicar-se com o usuário durante cada fase da transação, exibindo mensagens pré-definidas como 'Escolha um produto' (M1), 'Produto inválido' (M2), 'Deseja confirmar?' (M3), 'Pedido cancelado' (M4), 'Insira o dinheiro' (M5), 'Retire seu troco' (M6), e 'Retire o produto' (M7).

Este ciclo de operações assegura que a vending machine opere de maneira eficiente e segura, mantendo o dinheiro e o produto seguros até que todas as condições para uma transação bem-sucedida sejam atendidas, retornando então ao estado inicial para começar uma nova venda.

3. Diagrama de blocos:

Entradas	Num de bits	Função
Produto_selecionado	3	Escolher pedido (7 opções de produto, 1 - 7)
Dinheiro	8	Receber os valores inseridos
Confirma	1	1 se o usuário confirma a compra; 0 se não

Saídas	Num de bits	Função
Troco	8	Retornar dinheiro (troco ou valor integral)
Display	8	Imprimir mensagem do sistema
Produt_liberado	1	1 para liberado; 0 se não



As mensagens do sistema serão exibidas de maneira mais simples. Segue o significado e correspondência de cada uma:

- M1: “Escolha um produto” será exibida como 1 (001);
- M2: “Produto inválido” será exibida como 2 (010);
- M3: “Deseja confirmar?” será exibida como 3 (011);
- M4: “Pedido cancelado” será exibida como 4 (100);
- M5: “Insira o dinheiro” será exibida como 5 (101);
- M6: “Retire o troco” será exibida como 6 (110);
- M7: “Retire o produto” será exibida como 7 (111).

4. Diagrama do caminho de dados e controladora:

O sistema deve operar da seguinte maneira:

Bloco de Controle

Este bloco é o cérebro da operação. Ele recebe inputs externos, como "Confirma" (quando um usuário confirma a seleção do produto), e processa esses sinais. A partir daqui, ele emite uma série de controles internos que direcionam as operações da máquina:

- **Produto Selecionado Id**: Ativa quando um produto é selecionado.
- **Produto_selecionado_clr**: Reseta a seleção do produto após a conclusão da transação.
- **Dinheiro _total_ Id**: Ativa quando o dinheiro é inserido.
- **Dinheiro_total_clr**: Reseta a contagem de dinheiro após a emissão de troco ou cancelamento da transação.
- **Troco_Id:** Ativa quando o troco é calculado.
- **Troco_clr:** Reseta o valor do troco após sua emissão e finalização da compra.
- **Mensagem_Id:** Carrega a mensagem específica para aquele estado da máquina.
- **Mensagem_clr:** Reseta a mensagem previamente salva.

O bloco de controle também recebe feedback do Bloco Operacional por meio de sinais que indicam o estado da transação, como:

- **DItP**: Indica se a quantidade de dinheiro inserida é menor que o preço do produto.

- **DeqP**: Indica se a quantidade de dinheiro inserida é igual ao preço do produto.
- **DgtP**: Indica se a quantidade de dinheiro inserida é maior que o preço do produto.
- **Preco_eq_0**: Indica se o preço do produto é zero, sinalizando um produto inválido.

Além disso, a controladora também recebe um sinal "**Confirmar**", que pode ser 1 (caso o usuário confirme a compra) ou 0 (caso ele deseje cancelá-la).

Com base nesses sinais, o Bloco de Controle toma decisões, como continuar esperando por mais dinheiro, proceder à entrega do produto, ou emitir troco.

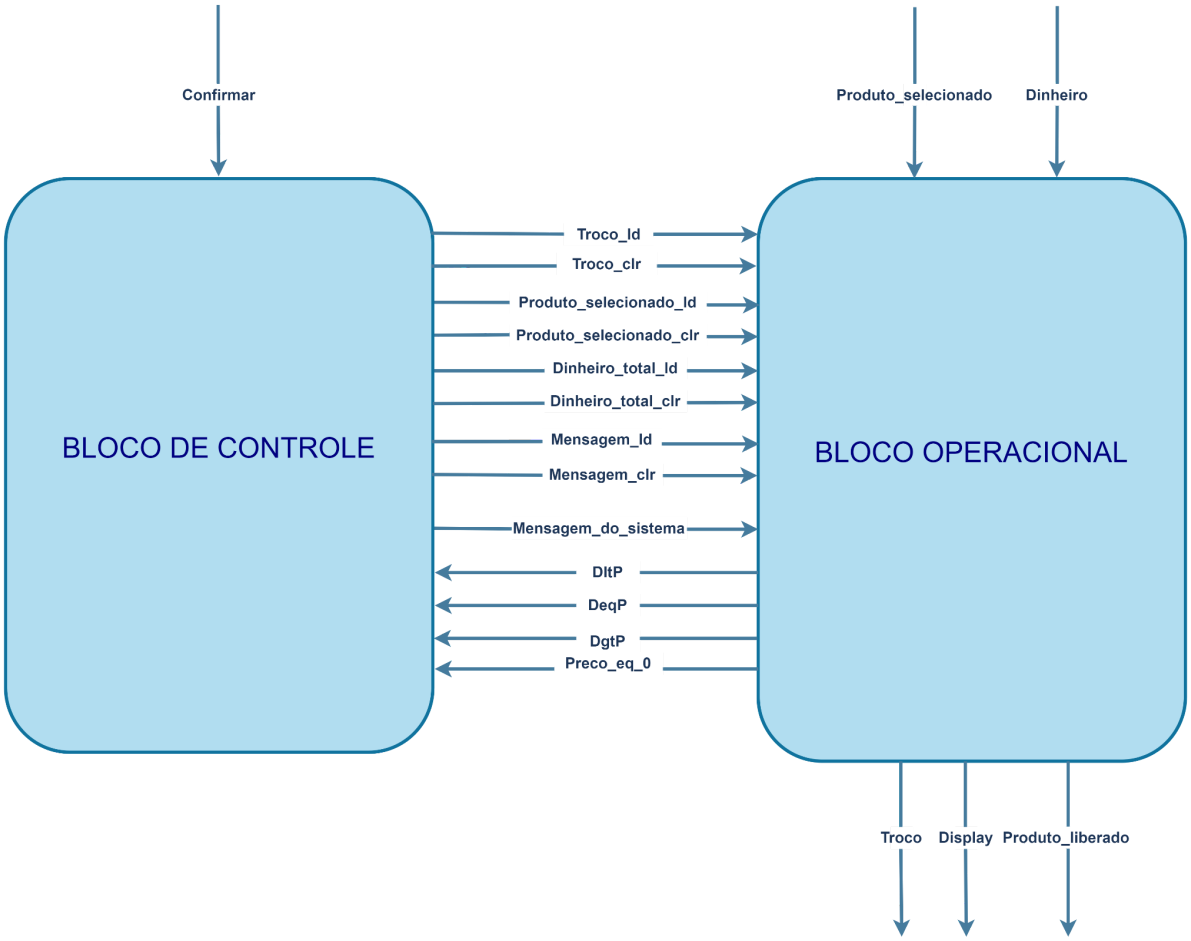
Bloco Operacional

Este bloco executa as ações físicas determinadas pelo Bloco de Controle. Ele contém mecanismos ou interfaces para aceitar dinheiro e selecionar produtos, além de dispositivos de saída para liberar o produto e o troco, e um display para mostrar mensagens ao usuário. As ações incluem:

- **Produto_selecionado:** Escolher o produto desejado (7 opções no total, 1 - 7).
- **Dinheiro:** Valor inserido para comprar o produto.
- **Troco**: É ativado quando o Bloco de Controle determina que mais dinheiro foi inserido do que o necessário, e o troco deve ser emitido.
- **Produto_liberado**: É ativado para liberar o produto selecionado após a confirmação e o pagamento adequado.
- **Display**: Mostra mensagens ao usuário, como instruções para inserir dinheiro ou confirmar a seleção do produto.

Integração dos Blocos

Quando um usuário interage com a máquina, selecionando um produto e inserindo dinheiro, essas informações são recebidas pelo Bloco Operacional e passadas ao Bloco de Controle. O Bloco de Controle, então, processa essas informações e emite comandos de volta ao Bloco Operacional para concluir a transação, seja dispensando o produto, emitindo troco, ou mostrando uma mensagem no display.



5. Máquinas de estado de alto e baixo nível:

No total, a Máquina de Estados Finitos (FSM) opera com 9 estados.

No estado "Início", a máquina reseta alguns valores, e mostra a mensagem de “Escolha um produto”. No estado "Valida produto", a máquina lê o produto selecionado e verifica sua validade. Se o produto for inválido (Preco_eq_0 = 1, pois foi definido que um produto inválido seria igual a 000), ela permanece neste estado (e a mensagem “Produto inválido” é exibida); caso contrário, avança para "Confirma".

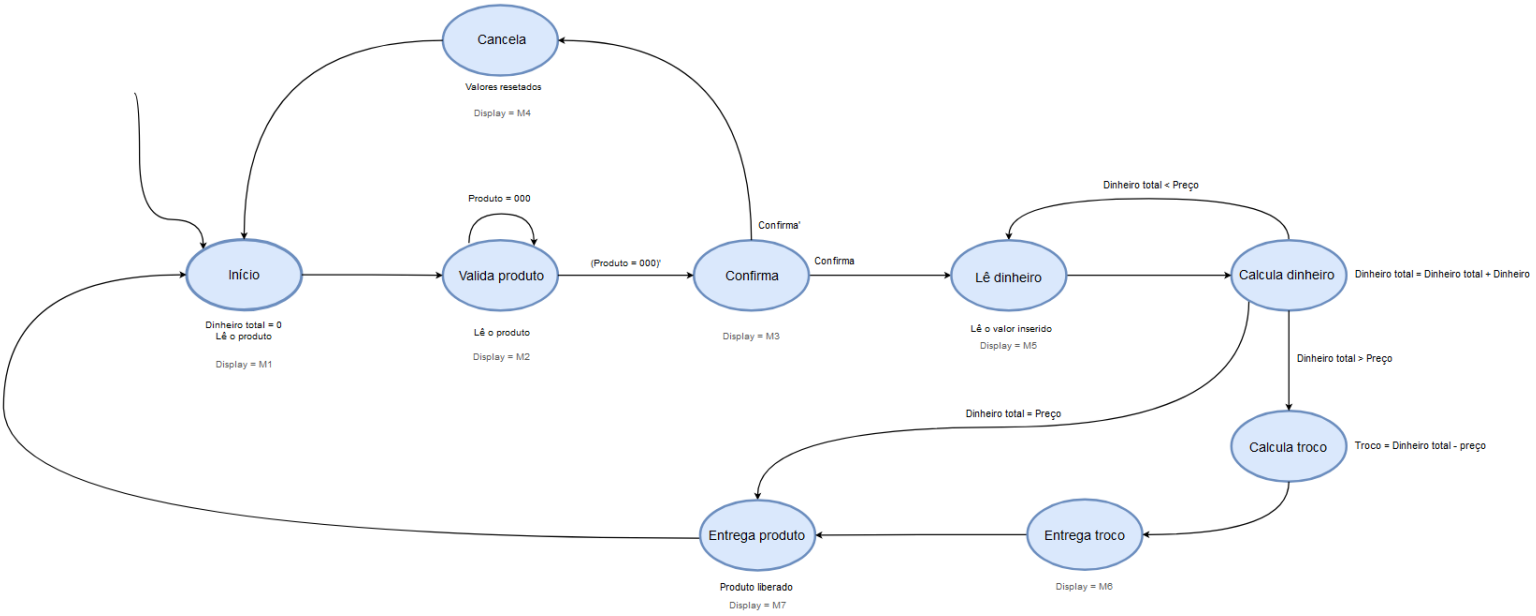
No estado "Confirma", a máquina aguarda a confirmação do usuário. Se o usuário desejar “não Confirmar” (cancelar), a máquina cancela a operação e retorna ao estado "Início". Se confirmado, ela transita para o estado "Lê dinheiro", exibindo uma mensagem para o usuário inserir o dinheiro e lendo os valores inseridos.

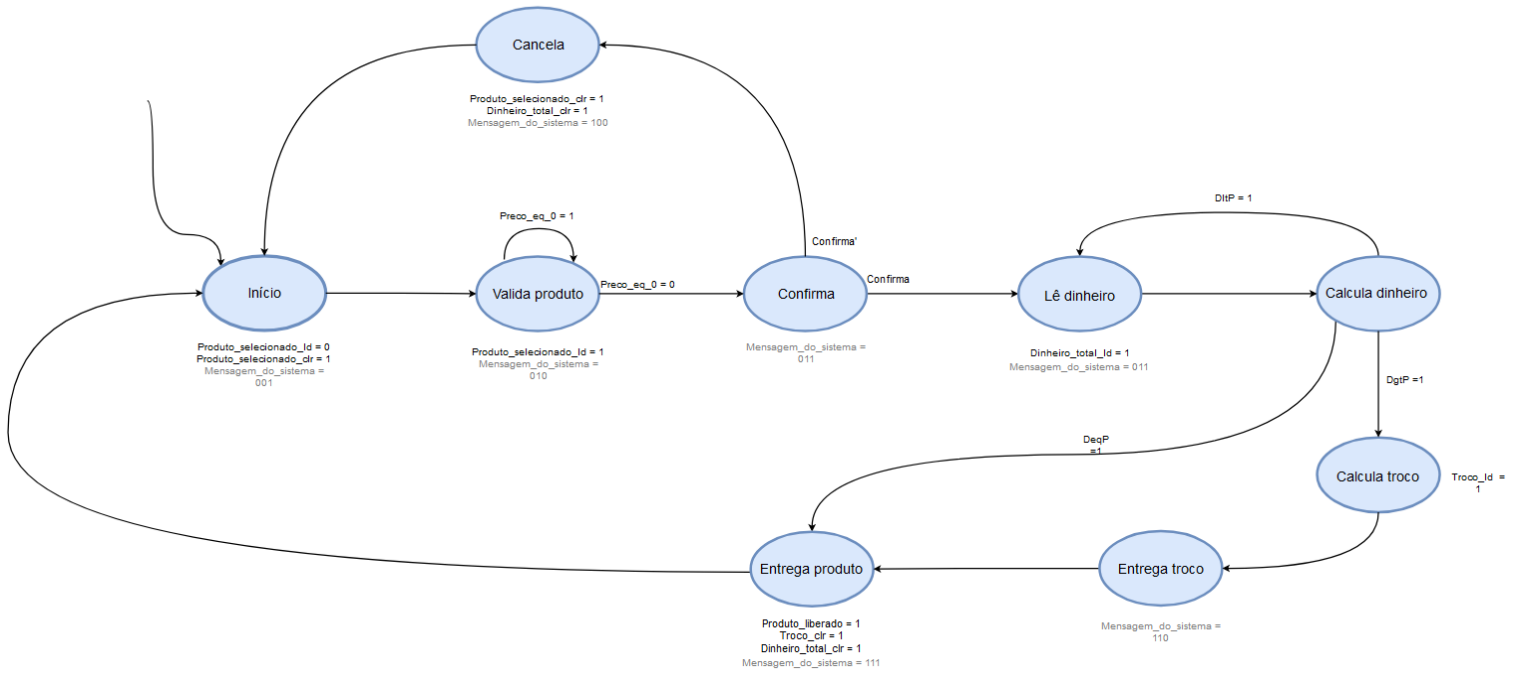
Após algum dinheiro ser inserido, a máquina avança para "Calcula dinheiro", onde realiza as operações necessárias. Enquanto o dinheiro inserido for menor que o preço do produto escolhido, ela volta para o estado de leitura, somando os valores iterativamente.

Se o dinheiro total inserido for maior que o valor do produto, ela passa para o estado "Calcula troco", onde a diferença entre o valor do produto comprado e do valor inserido é calculada. Logo em seguida, a máquina passa para o estado “Entrega troco”, em que

mostra uma mensagem para que o cliente o retire. Por fim, ela passa para o estado de entrega e libera o produto.

Se não for necessário troco, ou seja, o valor inserido é exatamente o valor do produto, a máquina entrega o produto diretamente, no estado "Entrega", redefinindo os contadores e preparando para a próxima transação.





6. Caminho de dados

Registros e Operações de Entrada

- **Produto**: Quando um produto é selecionado, o sinal "Produto_selecionado_ld" é enviado para um registrador de 3 bits, que armazena o código do produto. Se o sinal "Produto_selecionado_clr" for recebido, o registro é limpo, indicando que não há seleção de produto atual.
- **Dinheiro**: O dinheiro inserido é representado por uma entrada de 8 bits. Cada inserção de dinheiro é adicionada ao total armazenado em um registrador de 8 bits. Se "Dinheiro_total_clr" for ativado, o registro de dinheiro é zerado.
- **Mensagem**: De acordo com o estado em que a FSM está, diferentes mensagens devem ser mostradas no display. O sinal "Mensagem_ld" é enviado para um registrador de 3 bits, que armazena a respectiva mensagem (determinada pela Controladora, de acordo com o estado em que se encontra a operação). Se o sinal "Mensagem_clr" for recebido, o registro é limpo, indicando que não há mensagem.

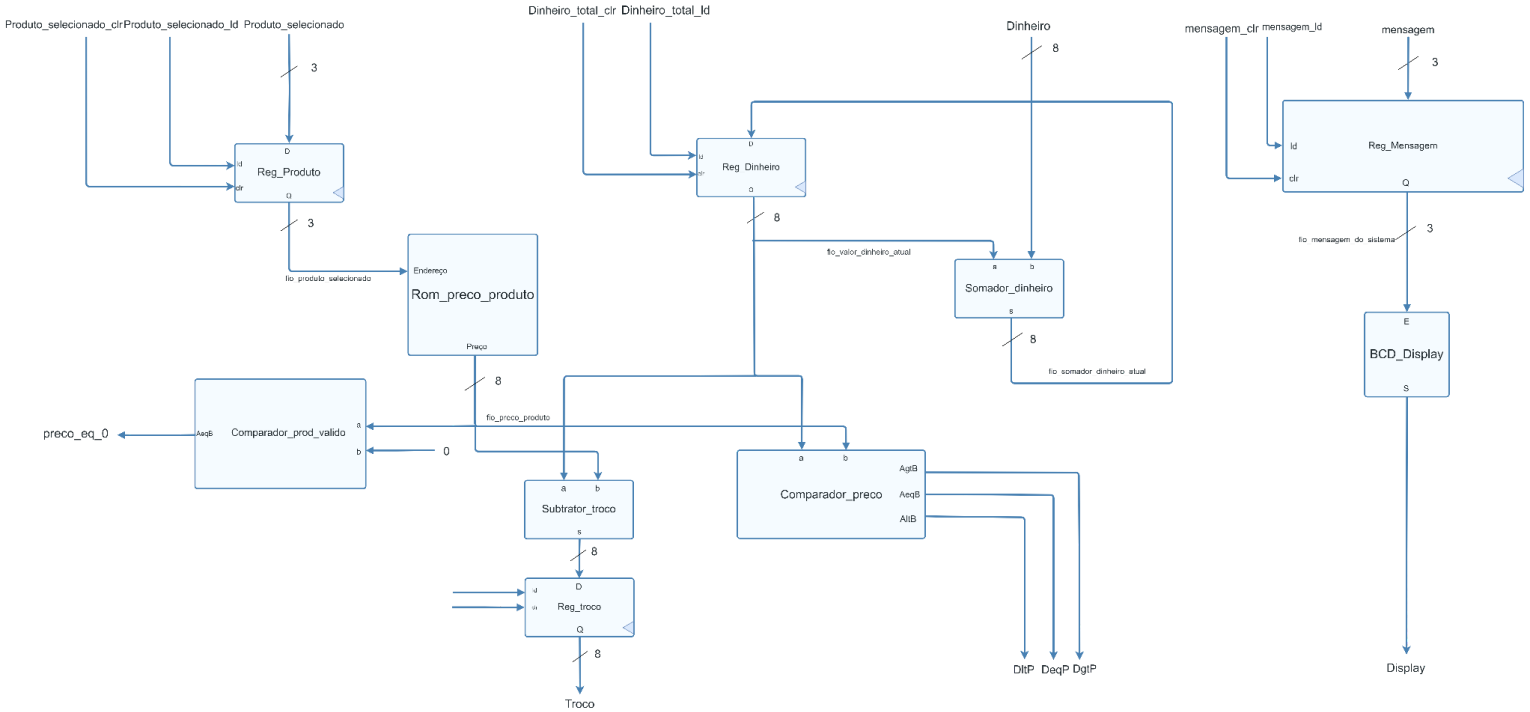
PS: Caso o projeto fosse testado, de fato, no FPGA, essa mensagem apareceria no display de sete segmentos (e, por isso, o componente).

Processamento e Comparação

- **ROM (Read-Only Memory)**: A seleção do produto é usada como um endereço para a ROM, que retorna o preço do produto correspondente (já que os valores são fixos). 000 sinaliza um produto inválido, e os outros 7 produtos (já que é um número composto por 3 bits), possuem seus respectivos preços com o mesmo valor que seu número (exemplo: produto 1 custa 1 real).
- **Comparadores**: O preço do produto recuperado da ROM é usado em várias comparações:
    - Com o total de dinheiro inserido, para determinar se o dinheiro é suficiente, menos ou mais do que o necessário (sinais DltP, DeqP, DgtP).
    - Com um valor fixo de zero para verificar se o produto selecionado é válido (sinal "produto_eq_0").
- **Somador**: O dinheiro inserido é somado ao dinheiro total, de forma que o cliente pode inserir valores quebrados, de poucos aos poucos, até atingir o valor do produto que deseja-se comprar.
- **Subtrator**: Usado para calcular a diferença entre o dinheiro total e o valor do produto, retornando o troco adequado.

Saídas:

- **Troco**: Se o dinheiro inserido for mais do que o necessário, o subtrator de 8 bits calcula o troco a ser devolvido.
- **Display**: Um registrador de 8 bits armazena mensagens do sistema que são exibidas para o usuário. O sinal "mensagem_ld" carrega uma nova mensagem, enquanto "mensagem_clr" limpa o display.
- **Preco_eq_0:** Verifica se o produto escolhido é válido. Caso o valor seja igual a 000, significa que ele é inválido.
- **DltP/ DeqP/ DgtP:** Com o total de dinheiro inserido, determinar se o dinheiro é, respectivamente, menos, suficiente ou mais do que o necessário.



7. Simulação individual de cada componente

Dado o contexto geral da problematização e o caminho de dados descrito acima, os componentes necessários para a execução do projeto são:

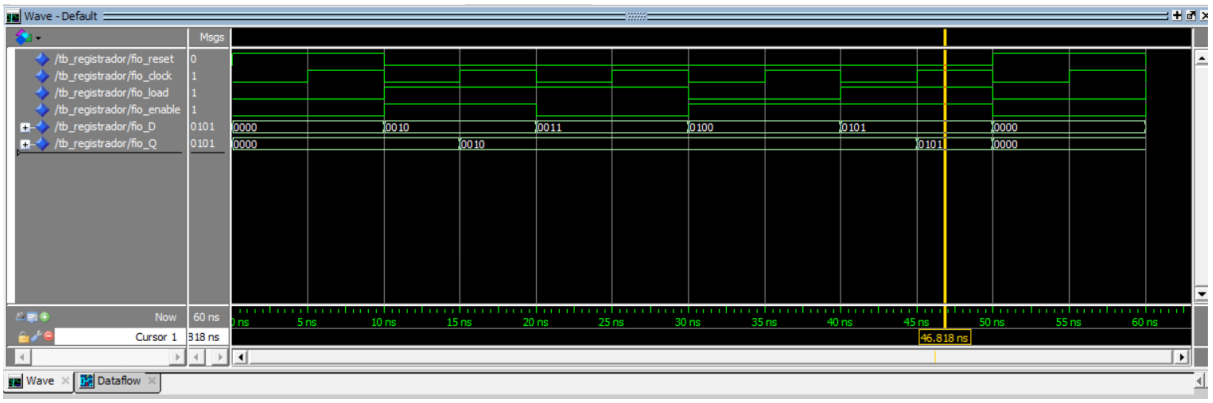
- Dois comparadores de 8 bits;
- Cinco registradores (genéricos; n bits);
- Um somador de 8 bits;
- Um subtrator de 8 bits.
- Uma ROM;
- Um display de sete segmentos.

Posto isso, fizemos a primeira versão destes componentes e de seus testbenches. Abaixo segue um trecho do código e a simulação pelo ModelSim Altera de cada componente e uma breve explicação.

a. Registrador

O componente registrador possui tamanho genérico e, além das entradas reset e clock, possui as entradas load e enable, para mantermos mais controle sobre quando a saída deve ser alterada ou não. É possível ver sua implementação e a simulação pelo ModelSim Altera logo abaixo.

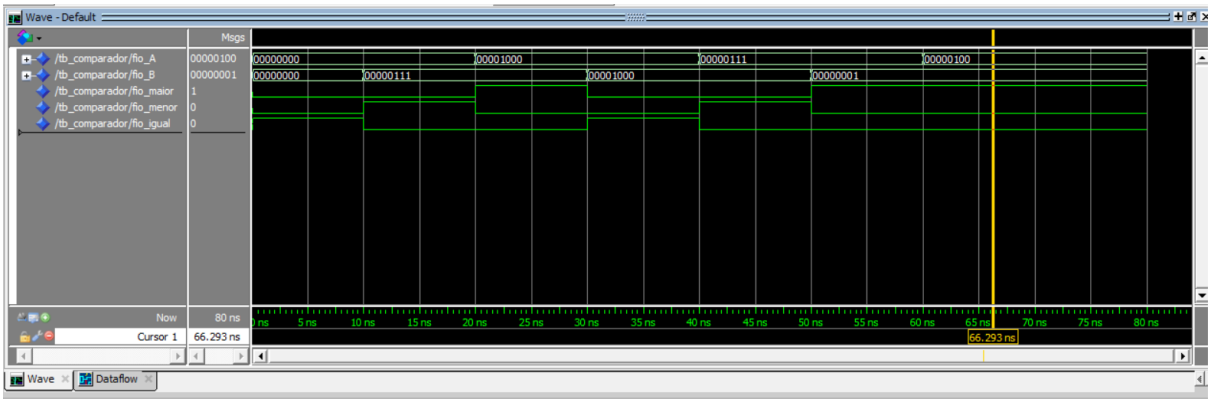
```
4  entity registrador is
5      generic (
6          W : integer := 8
7      );
8      port (
9          reset   : in std_logic;
10         clock   : in std_logic;
11         load    : in std_logic;
12         enable  : in std_logic;
13         D       : in std_logic_vector(W-1 downto 0);
14         Q       : out std_logic_vector(W-1 downto 0)
15     );
16 end registrador;
17
18 architecture RTL of registrador is
19     signal registro : std_logic_vector(W-1 downto 0) := (others => '0');
20 begin
21     process (clock, reset)
22     begin
23         if reset = '1' then
24             registro <= (others => '0');
25         elsif rising_edge(clock) then
26             if enable = '1' then
27                 if load = '1' then
28                     registro <= D;
29                 end if;
30             end if;
31         end if;
32     end process;
33
34     Q <= registro;
35 end RTL;
```



b. Comparador

O componente comparador possui três saídas de 1 bit, uma para cada condição de comparação (maior que, menor que e igual a) entre as entradas A e B, nesta ordem, como mostra a simulação e a descrição do comparador abaixo.

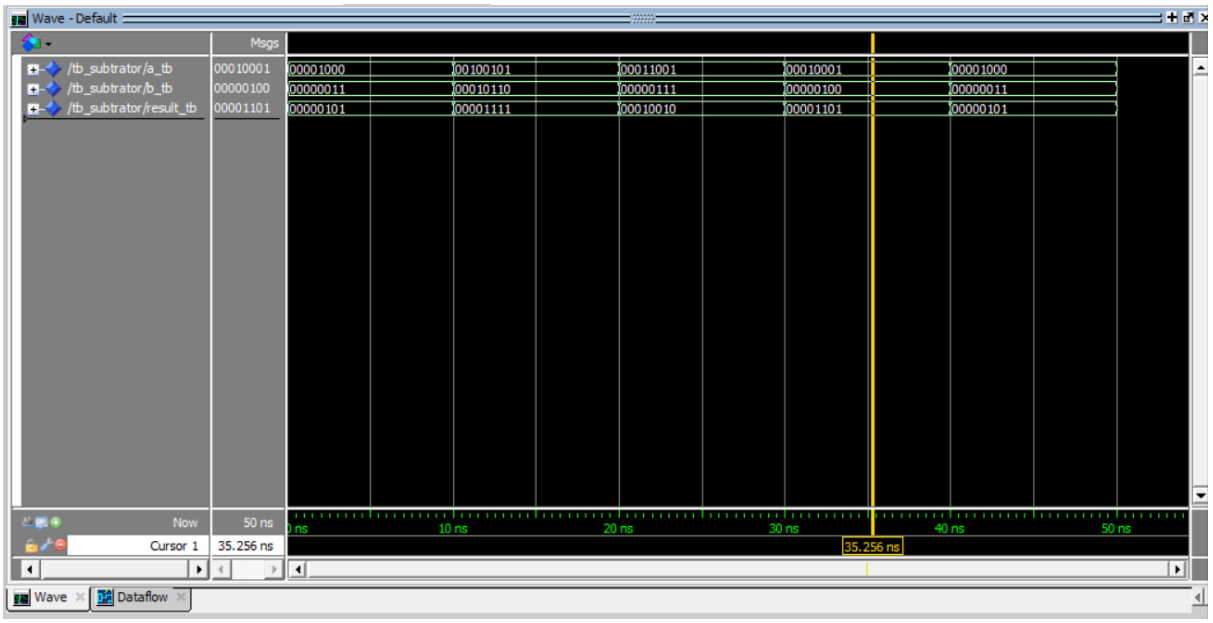
```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.all;
3  use ieee.numeric_std.all;
4
5  entity comparador is
6  generic (
7        W : integer := 8
8     );
9
10    port (
11       a  : in std_logic_vector   ((W-1) downto 0);
12       b  : in std_logic_vector   ((W-1) downto 0);
13       maior : out std_logic;
14       menor : out std_logic;
15       igual : out std_logic
16    );
17 end comparador;
18
19 architecture estrutural of comparador is
20 begin
21    maior <= '1' when unsigned(a) > unsigned(b) else '0';
22    menor <= '1' when unsigned(a) < unsigned(b) else '0';
23    igual <= '1' when unsigned(a) = unsigned(b) else '0';
24 end;
```



c. Subtrator

O componente subtrator possui tamanho genérico e realiza a subtração entre duas entradas do tipo unsigned “a” e “b”. É possível ver seu comportamento na simulação e sua implementação abaixo.

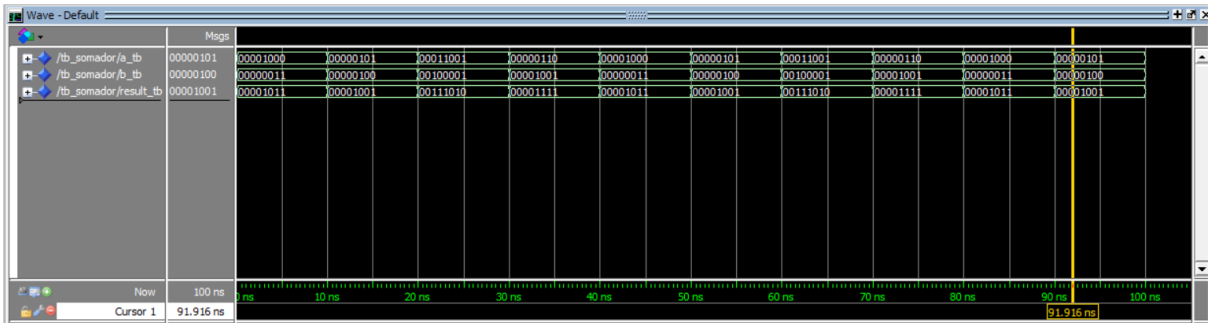
```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity subtrator is
   generic (
      W : integer := 8
   );
   port (
      a      : in std_logic_vector ((W-1) downto 0);
      b      : in std_logic_vector ((W-1) downto 0);
      result : out std_logic_vector ((W-1) downto 0)
   );
end entity;

architecture rtl of subtrator is
begin

   process(a,b)
   begin
         result <= std_logic_vector(unsigned(a) - unsigned(b));
   end process;
end rtl;
```



d. Somador

O componente somador funciona de forma similar ao subtrator. Os resultados da simulação podem ser vistos abaixo, junto com sua implementação.

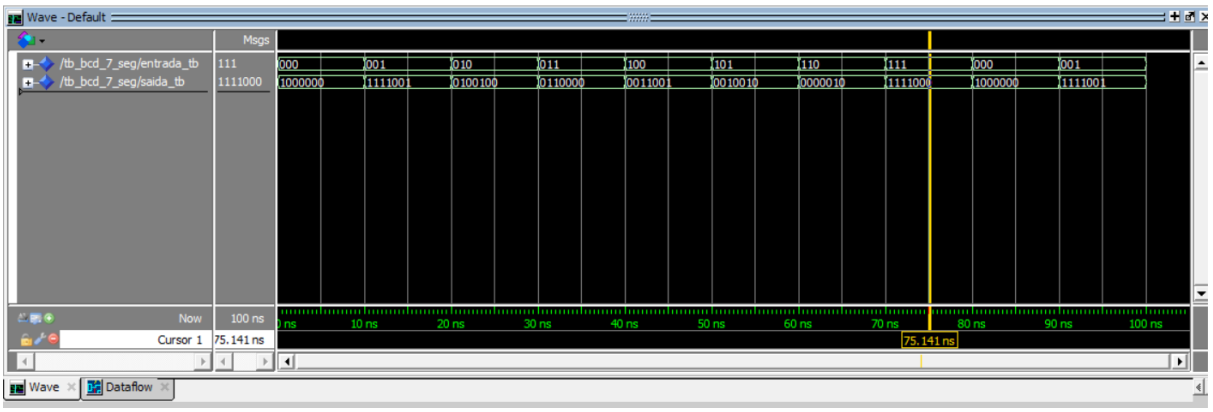
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity somador is
    generic (
        W : integer := 8
    );
    port (
        a      : in  std_logic_vector(W-1 downto 0);
        b      : in  std_logic_vector(W-1 downto 0);
        result : out std_logic_vector(W-1 downto 0)
    );
end entity;

architecture rtl of somador is
begin
    process(a, b)
    begin
      result <= std_logic_vector(unsigned(a) + unsigned(b));
    end process;
end rtl;
```



e. Display de 7 segmentos

O display de 7 segmentos possui 8 possibilidades de saída e foi testado com o arquivo testbench abaixo, que gerou a seguinte simulação:

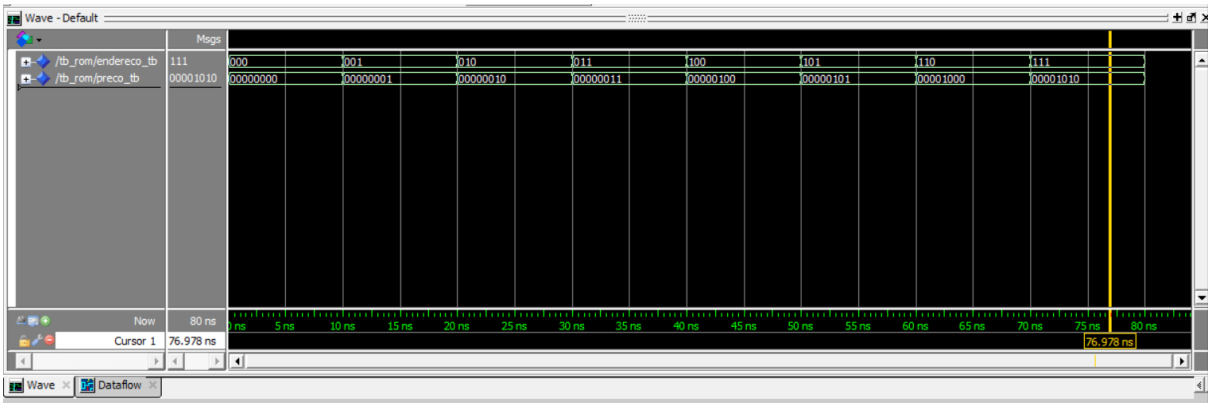
```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity Bcd_7seg is
port (
entrada: in std_logic_vector (2 downto 0);
saida: out std_logic_vector (6 downto 0)
);
end Bcd_7seg;

architecture with_select_bcd7seg of Bcd_7seg is
begin
with entrada select
saida <= "1000000" when "000", --0
"1111001" when "001", --1
"0100100" when "010", --2
"0110000" when "011", --3
"0011001" when "100", --4
"0010010" when "101", --5
"0000010" when "110", --6
"1111000" when "111", --7
"0000000" when others;

end with_select_bcd7seg;
```



f. ROM

A ROM (Read Only Memory) é usada para consulta do preço do produto. Dessa forma, como já explicado anteriormente e ilustrado pelo caminho de dados, a ROM do preço do produto foi desenvolvida e simulada da forma seguinte:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity ROM is
    port (
        endereco : in  std_logic_vector(2 downto 0);
        preco: out std_logic_vector(7 downto 0)
    );
end ROM;

architecture rtl of ROM is
    type ROM_type is array (0 to 7) of std_logic_vector(7 downto 0);
    constant ROM_data : ROM_type := (
        x"00", -- produto inválido
        x"01", -- Para produto 1
        x"02", -- Para produto 2
        x"03", -- Para produto 3
        x"04", -- Para produto 4
        x"05", -- Para produto 5
        x"08", -- Para produto 6
        x"0A"  -- Para produto 7
    );
begin
    preco <= ROM_data(to_integer(unsigned(endereco)));
end rtl;

```



8. Implementação do Caminho de Dados, da Controladora e da VendingMachine

Caminho de Dados:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity datapath is
    port (
        Produto_selecionado: in std_logic_vector(2 downto 0);
        Dinheiro: in std_logic_vector(7 downto 0);
        CLOCK: in std_logic;
        Produto_selecionado_ld: in std_logic;
        Produto_selecionado_clr: in std_logic;
        Dinheiro_total_ld: in std_logic;
        Dinheiro_total_clr: in std_logic;
        Mensagem_clr: in std_logic;
        Mensagem_ld: in std_logic;
        Troco_clr: in std_logic;
        Troco_ld: in std_logic;
        Mensagem_do_sistema: in std_logic_vector(2 downto 0);

        DgtP: out std_logic;
        DltP: out std_logic;
        DeqP: out std_logic;
        Preco_eq_0: out std_logic;
        Troco : out std_logic_vector(7 downto 0) := "00000000";
        Display : out std_logic_vector(6 downto 0)
    );
end datapath;

architecture arch of datapath is
    component registrador
       generic (
           W : integer := 8
       );
       port (
```

```
34        port (
35            reset   : in std_logic;
36            clock   : in std_logic;
37            load    : in std_logic;
38            enable  : in std_logic;
39            D       : in std_logic_vector(W-1 downto 0);
40            Q       : out std_logic_vector(W-1 downto 0)
41        );
42    end component;
43
44    component comparador
45        generic (
46          W : integer := 8
47        );
48        port (
49          a  : in std_logic_vector   ((W-1) downto 0);
50          b  : in std_logic_vector   ((W-1) downto 0);
51          maior : out std_logic;
52          menor : out std_logic;
53          igual : out std_logic
54        );
55    end component;
56
57    component somador
58        generic (
59          W : integer := 8
60        );
61        port (
62          a     : in std_logic_vector((W-1) downto 0);
63          b     : in std_logic_vector((W-1) downto 0);
64          result: out std_logic_vector((W-1) downto 0)
65        );
66    end component;
67
```

```
67
68      component subtrator
69          generic (
70              W : integer := 8
71          );
72          port(
73              a      : in std_logic_vector((W-1) downto 0);
74              b      : in std_logic_vector((W-1) downto 0);
75              result: out std_logic_vector((W-1) downto 0)
76          );
77      end component;
78
79      component BCD_7seg
80        port (
81            entrada: in std_logic_vector (2 downto 0);
82            saida: out std_logic_vector (6 downto 0)
83        );
84      end component;
85
86      component ROM
87        port (
88            endereco: in std_logic_vector(2 downto 0);
89            preco    : out std_logic_vector(7 downto 0)
90        );
91      end component;
92
93      -- sinais internos
94      signal fio_produto_selecionado: std_logic_vector(2 downto 0);
95      signal fio_valor_dinheiro_atual: std_logic_vector(7 downto 0);
96      signal fio_somador_dinheiro_atual: std_logic_vector(7 downto 0);
97      signal fio_mensagem_display: std_logic_vector(2 downto 0);
98      signal fio_preco_produto: std_logic_vector(7 downto 0);
99      signal fio_subtrator_reg_troco: std_logic_vector(7 downto 0);
100
```

```
100
101     begin
102     -- Instância dos componentes
103     Reg_Produto: registrador
104         generic map(
105             W => 3
106         )
107         port map(
108             reset => Produto_selecionado_clr,
109             clock => CLOCK,
110             load  => Produto_selecionado_ld,
111             enable => '1',
112             D      => Produto_selecionado,
113             Q      => fio_produto_selecionado
114         );
115
116     Reg_Dinheiro: registrador
117         generic map(
118             W => 8
119         )
120         port map(
121             reset => Dinheiro_total_clr,
122             clock => CLOCK,
123             load  => Dinheiro_total_ld,
124             enable => '1',
125             D      => fio_somador_dinheiro_atual,
126             Q      => fio_valor_dinheiro_atual
127         );
128
129     Reg_Mensagem: registrador
130         generic map(
131             W => 3
132         )
133         port map(
```

```
133         port map(
134            reset => Mensagem_clr,
135            clock => CLOCK,
136            load  => Mensagem_ld,
137            enable => '1',
138            D      => Mensagem_do_sistema,
139            Q      => fio_mensagem_display
140         );
141
142     Reg_Troco : registrador
143        generic map(
144              W => 8
145         )
146         port map(
147            reset => Troco_clr,
148            clock => CLOCK,
149            load  => Troco_ld,
150            enable => '1',
151            D      => fio_subtrator_reg_troco,
152            Q      => Troco
153         );
154
155   Comparador_prod_valido: comparador
156         generic map (
157              W => 8
158         )
159         port map (
160              a     => fio_preco_produto,
161              b     => (others => '0'),
162              igual => Preco_eq_0
163         );
164
165     Comparador_preco: comparador
166        generic map (
```

```
166       generic map (
167          W => 8
168       )
169       port map(
170          a => fio_valor_dinheiro_atual,
171          b => fio_preco_produto,
172          maior => DgtP,
173          menor => DltP,
174          igual => DeqP
175       );
176
177    Rom_preco_produto: ROM
178       port map(
179          endereco => fio_produto_selecionado,
180          preco    => fio_preco_produto
181       );
182
183    Somador_dinheiro: somador
184        generic map(
185            W => 8
186        )
187        port map(
188            a => fio_valor_dinheiro_atual,
189            b => Dinheiro,
190            result => fio_somador_dinheiro_atual
191        );
192
193    Subtrator_troco: subtrator
194        generic map(
195          W => 8
196        )
197        port map(
198          a => fio_valor_dinheiro_atual,
199          b => fio_preco_produto,
```

```
199           b => fio_preco_produto,
200           result => fio_subtrator_reg_troco
201         );
202
203    BCD_Display: BCD_7seg
204       port map (
205         entrada => fio_mensagem_display,
206         saida => Display
207       );
208
209  end arch;
```

Controladora:

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

entity Controladora is
    port (
        RESET, CLOCK, Confirmar: in std_logic;
        DltP, DeqP, DgtP, Preco_eq_0: in std_logic;
        Produto_liberado: out std_logic;
        Produto_selecionado_clr, Produto_selecionado_ld, Mensagem_ld, Mensagem_clr, Dinheiro_total_ld, Dinheiro_total_clr, Troco_clr, Troco_ld: out std_logic;
        Mensagem_do_sistema: out std_logic_vector(2 downto 0)
        );
end Controladora;

architecture arch of Controladora is
    type state_type is (inicio, validaProduto, confirma, cancela, leiaDinheiro, calculaDinheiro, calculaTroco, entregaTroco, entregaProduto);
    signal state, next_state: state_type;

begin
     process (state, Confirmar, DltP, DeqP, DgtP, Preco_eq_0)
     begin
        case state is
           when inicio =>
                Produto_selecionado_ld <= '0';
                Produto_selecionado_clr <= '1';
                Mensagem_do_sistema <= "001";
                Mensagem_ld <= '1';
                Mensagem_clr <= '0';
                Dinheiro_total_ld <= '0';
                Dinheiro_total_clr <= '1';
                Troco_clr <= '1';
                Troco_ld <= '0';
                Produto_liberado <= '0';
                next_state <= validaProduto;
```

```vhdl
                next_state <= validaProduto;

            when validaProduto =>
                Produto_selecionado_clr <= '0';
                Mensagem_clr <= '0';
                Dinheiro_total_ld <= '0';
                Dinheiro_total_clr <= '1';
                Produto_liberado <= '0';
                Troco_clr <= '1';
                Troco_ld <= '0';
                if Preco_eq_0 = '1' then
                   Produto_selecionado_ld <= '1';
                   Mensagem_do_sistema <= "010";
                   Mensagem_ld <= '1';
                   next_state <= validaProduto;
                elsif Preco_eq_0 = '0' then
                   Produto_selecionado_ld <= '0';
                   Mensagem_ld <= '0';
                   next_state <= confirma;
                else
                   Produto_selecionado_ld <= '0';
                   Mensagem_ld <= '1';
                   next_state <= cancela;
                end if;

            when confirma =>
                Produto_selecionado_ld <= '0';
                Produto_selecionado_clr <= '0';
                Mensagem_do_sistema <= "011";
                Mensagem_ld <= '1';
                Mensagem_clr <= '0';
                Dinheiro_total_ld <= '0';
                Dinheiro_total_clr <= '1';
                Produto_liberado <= '0';
```

```
                    Produto_liberado <= '0';
                    Troco_clr <= '1';
                    Troco_ld <= '0';
                    if Confirmar = '1' then
                        next_state <= leiaDinheiro;
                    else
                        next_state <= cancela;
                    end if;

                when leiaDinheiro =>
                    Produto_selecionado_ld <= '0';
                    Produto_selecionado_clr <= '0';
                    Mensagem_do_sistema <= "101";
                    Mensagem_ld <= '1';
                    Mensagem_clr <= '0';
                    Dinheiro_total_ld <= '1';
                    Dinheiro_total_clr <= '0';
                    Troco_ld <= '0';
                    Troco_clr <= '1';
                    Produto_liberado <= '0';
                    next_state <= calculaDinheiro;

                when calculaDinheiro =>
                    Produto_selecionado_ld <= '0';
                    Produto_selecionado_clr <= '0';
                    Mensagem_ld <= '0';
                    Mensagem_clr <= '0';
                    Dinheiro_total_ld <= '0';
                    Dinheiro_total_clr <= '0';
                    Troco_ld <= '0';
                    Troco_clr <= '1';
                    Produto_liberado <= '0';
                    if DgtP = '1' then
                        next_state <= calculaTroco;
```

```
101                 elsif DeqP = '1' then
102                    next_state <= entregaProduto;
103                 elsif DltP = '1' then
104                    next_state <= leiaDinheiro;
105                 else
106                    next_state <= cancela;
107                 end if;
108
109           when calculaTroco =>
110                 Produto_selecionado_ld <= '0';
111                 Produto_selecionado_clr <= '0';
112                 Mensagem_do_sistema <= "110";
113                 Mensagem_ld <= '1';
114                 Mensagem_clr <= '0';
115                 Dinheiro_total_ld <= '0';
116                 Dinheiro_total_clr <= '0';
117                 Produto_liberado <= '0';
118                 Troco_ld <= '1';
119                 Troco_clr <= '0';
120                 next_state <= entregaTroco;
121
122           when entregaTroco=>
123                 Produto_selecionado_ld <= '0';
124                 Produto_selecionado_clr <= '0';
125                 Mensagem_do_sistema <= "111";
126                 Mensagem_ld <= '1';
127                 Mensagem_clr <= '0';
128                 Dinheiro_total_ld <= '1';
129                 Dinheiro_total_clr <= '0';
130                 Produto_liberado <= '0';
131                 Troco_clr <= '0';
132                 Troco_ld <= '1';
133                 next_state <= entregaProduto;
134
```

```
134
135             when entregaProduto =>
136                   Produto_selecionado_ld <= '0';
137                   Produto_selecionado_clr <= '0';
138                   Mensagem_do_sistema <= "111";
139                   Mensagem_ld <= '1';
140                   Mensagem_clr <= '0';
141                   Dinheiro_total_ld <= '0';
142                   Dinheiro_total_clr <= '1';
143                   Produto_liberado <= '1';
144                   Troco_clr <= '1';
145                   Troco_ld <= '0';
146                   next_state <= inicio;
147
148             when cancela =>
149                   Produto_selecionado_ld <= '0';
150                   Produto_selecionado_clr <= '1';
151                   Dinheiro_total_ld <= '0';
152                   Dinheiro_total_clr <= '1';
153                   Mensagem_do_sistema <= "100";
154                   Mensagem_ld <= '1';
155                   Mensagem_clr <= '0';
156                   Troco_clr <= '1';
157                   Troco_ld <= '0';
158                   Produto_liberado <= '0';
159                   next_state <= inicio;
160
161             end case;
162     end process;
163
164     process (RESET, CLOCK)
165      begin
166         if RESET = '1' then
167             state <= inicio;
167          elsif rising_edge(CLOCK) then
168              state <= next_state;
169          end if;
170      end process;
171
172 end arch;
```

VendingMachine:

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity VendingMachine is
    port (
      CLOCK: in std_logic;
      Confirmar: in std_logic;
      Dinheiro: in std_logic_vector(7 downto 0);
      Produto_selecionado: in std_logic_vector(2 downto 0);
      Troco: out std_logic_vector(7 downto 0);
      Display: out std_logic_vector(6 downto 0);
      Produto_liberado: out std_logic
    );
end VendingMachine;

architecture RTL OF VendingMachine is
    signal fio_DltP, fio_DeqP, fio_DgtP: std_logic;
    signal fio_Preco_eq_0: std_logic;
    signal fio_Produto_selecionado_clr, fio_Produto_selecionado_ld: std_logic;
    signal fio_Dinheiro_total_ld, fio_Dinheiro_total_clr: std_logic;
    signal fio_Troco_ld, fio_Troco_clr: std_logic;
    signal fio_Preco_ld, fio_Preco_clr: std_logic;
    signal fio_Mensagem_clr, fio_Mensagem_ld: std_logic;
    signal fio_Mensagem_do_sistema: std_logic_vector(2 downto 0);

   component Controladora
    port (
       RESET, CLOCK, Confirmar: in std_logic;
       DltP, DeqP, DgtP, Preco_eq_0: in std_logic;
       Produto_liberado: out std_logic;
       Produto_selecionado_clr, Produto_selecionado_ld, Mensagem_ld, Mensagem_clr, Dinheiro_total_ld, Dinheiro_total_clr, Troco_clr, Troco_ld: out std_logic;
       Mensagem_do_sistema: out std_logic_vector(2 downto 0)
       );
```

```vhdl
       );
    end component;

    component DataPath
       port (
          Produto_selecionado: in std_logic_vector(2 downto 0);
          Dinheiro: in std_logic_vector(7 downto 0);
          CLOCK: in std_logic;
          Produto_selecionado_ld: in std_logic;
          Produto_selecionado_clr: in std_logic;
          Dinheiro_total_ld: in std_logic;
          Dinheiro_total_clr: in std_logic;
          Mensagem_clr: in std_logic;
          Mensagem_ld: in std_logic;
          Troco_clr: in std_logic;
          Troco_ld: in std_logic;
          Mensagem_do_sistema: in std_logic_vector(2 downto 0);

          DgtP: out std_logic;
          DltP: out std_logic;
          DeqP: out std_logic;
          Preco_eq_0: out std_logic;
          Troco : out std_logic_vector(7 downto 0);
          Display : out std_logic_vector(6 downto 0)
       );
    end component;

    begin

       DPath: DataPath
          port map(
             Produto_selecionado          => Produto_selecionado,
             Dinheiro                     => Dinheiro,
             CLOCK                        => CLOCK,
```

```
67              CLOCK                           => CLOCK,
68
69              Produto_selecionado_ld          => fio_Produto_selecionado_ld,
70              Produto_selecionado_clr         => fio_Produto_selecionado_clr,
71              Dinheiro_total_ld               => fio_Dinheiro_total_ld,
72              Dinheiro_total_clr              => fio_Dinheiro_total_clr,
73              Mensagem_clr                    => fio_Mensagem_clr,
74              Mensagem_ld                     => fio_Mensagem_ld,
75              Mensagem_do_sistema             => fio_Mensagem_do_sistema,
76              DgtP                            => fio_DgtP,
77              DltP                            => fio_DltP,
78              DeqP                            => fio_DeqP,
79              Preco_eq_0                      => fio_Preco_eq_0,
80              Troco_clr                       => fio_Troco_clr,
81              Troco_ld                        => fio_Troco_ld,
82              Troco                           => Troco,
83              Display                         => Display
84          );
85
86      Ctrl: Controladora
87          port map(
88              CLOCK                           => CLOCK,
89              Confirmar                       => Confirmar,
90              RESET                           => '0',
91
92              DltP                            => fio_DltP,
93              DeqP                            => fio_DeqP,
94              DgtP                            => fio_DgtP,
95              Preco_eq_0                      => fio_Preco_eq_0,
96              Produto_selecionado_clr         => fio_Produto_selecionado_clr,
97              Produto_selecionado_ld          => fio_Produto_selecionado_ld,
98              Dinheiro_total_ld               => fio_Dinheiro_total_ld,
99              Dinheiro_total_clr              => fio_Dinheiro_total_clr,
100             Troco_clr                       => fio_Troco_clr,
```

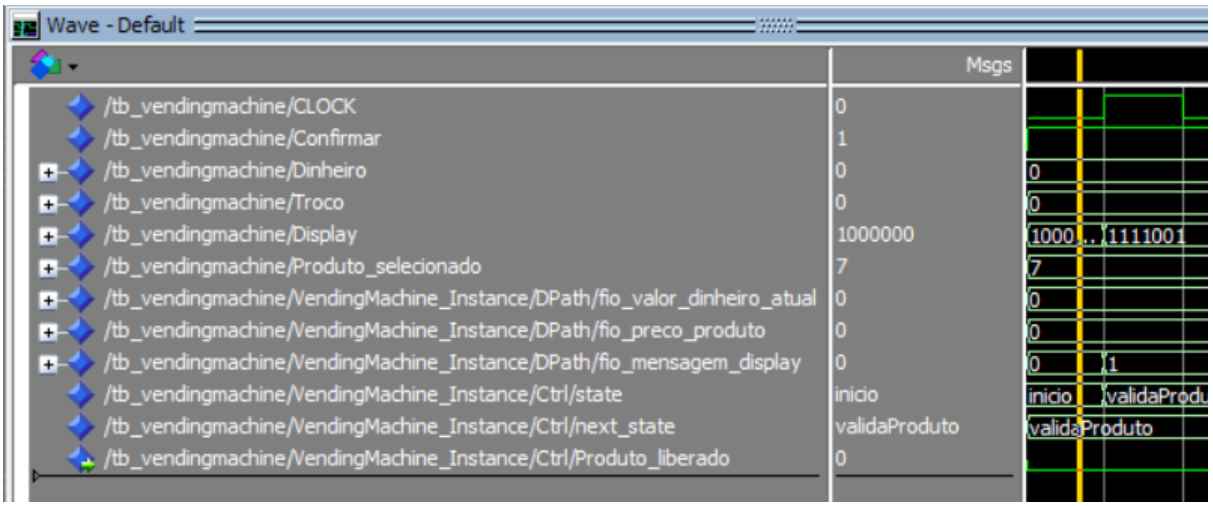
```
100             Troco_clr                       => fio_Troco_clr,
101             Troco_ld                        => fio_Troco_ld,
102             Mensagem_clr                    => fio_Mensagem_clr,
103             Mensagem_ld                     => fio_Mensagem_ld,
104             Mensagem_do_sistema             => fio_Mensagem_do_sistema,
105
106             Produto_liberado                => Produto_liberado
107         );
108 end RTL ;
```

9. Simulação de um cenário da vida real

```vhdl
library ieee;
use ieee.STD_LOGIC_1164.ALL;
use ieee.STD_LOGIC_ARITH.ALL;
use ieee.STD_LOGIC_UNSIGNED.ALL;

entity tb_VendingMachine is
end tb_VendingMachine;

architecture testbench of tb_VendingMachine is
    signal CLOCK, Confirmar: std_logic := '0';
    signal Dinheiro, Troco: std_logic_vector(7 downto 0) := "00000000";
    signal Display: std_logic_vector(6 downto 0):= "0000000";
    signal Produto_selecionado: std_logic_vector(2 downto 0) := "000";

    component VendingMachine
        port (
            CLOCK, Confirmar: in std_logic;
            Dinheiro: in std_logic_vector(7 downto 0);
            Produto_selecionado: in std_logic_vector(2 downto 0);
            Troco: out std_logic_vector(7 downto 0);
            Display: out std_logic_vector(6 downto 0)
        );
    end component;

begin
    VendingMachine_Instance: VendingMachine
        port map(
            CLOCK => CLOCK,
            Confirmar => Confirmar,
            Dinheiro => Dinheiro,
            Produto_selecionado => Produto_selecionado,
            Troco => Troco,
            Display => Display
        );
```

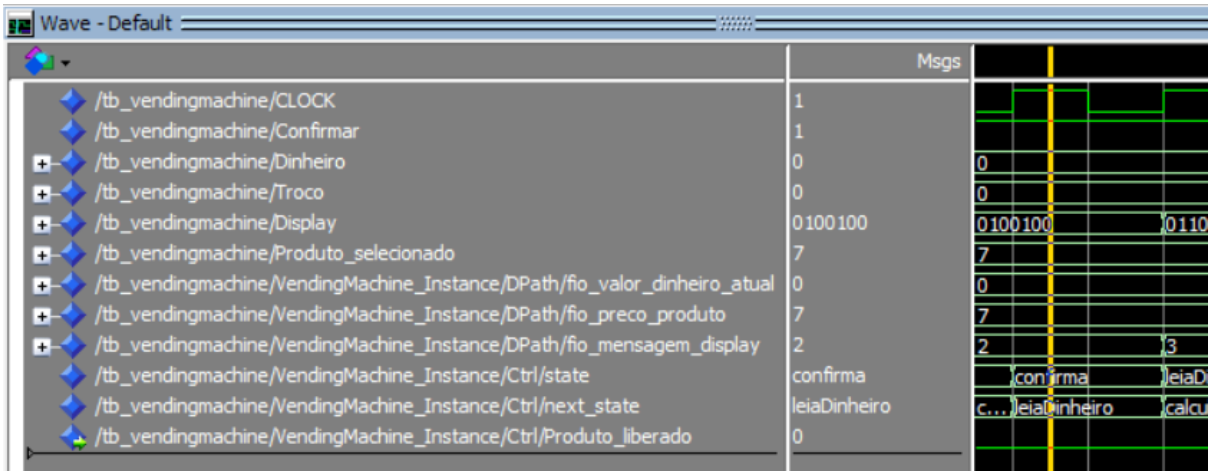
```
34          );
35
36      clk_process: process
37      begin
38          CLOCK <= '0';
39          wait for 10 ns;
40          CLOCK <= '1';
41          wait for 10 ns;
42      end process;
43
44      process
45         begin
46         --Produto_selecionado <= "011"; - 130 ns
47         --Confirmar <= '1';
48         --Dinheiro <= "00000110";
49
50         --Produto_selecionado <= "100"; -- 100 ns
51         --Confirmar <= '0';
52         --Dinheiro <= "00000000";
53
54         Produto_selecionado <= "111"; -- 220 ns
55         Confirmar <= '1';
56         Dinheiro <= "00000000";
57         wait for 80 ns;
58         Dinheiro <= "00000101";
59         wait for 20 ns;
60         Dinheiro <= "00000110";
61
62         wait;
63      end process;
64  end testbench;
```

- **Simulação - ModelSim Altera**

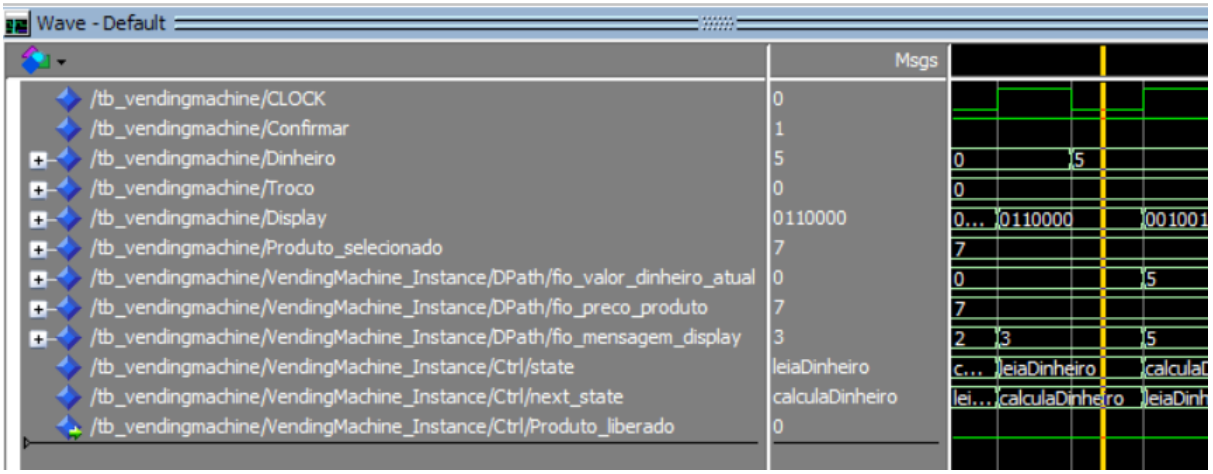


No estado inicial é solicitado ao usuário que o mesmo selecione um produto para seguir com a compra, a partir do código da mensagem impressa pelo display de 7 segmentos. Em seguida, a máquina segue para o estado de validação do produto.

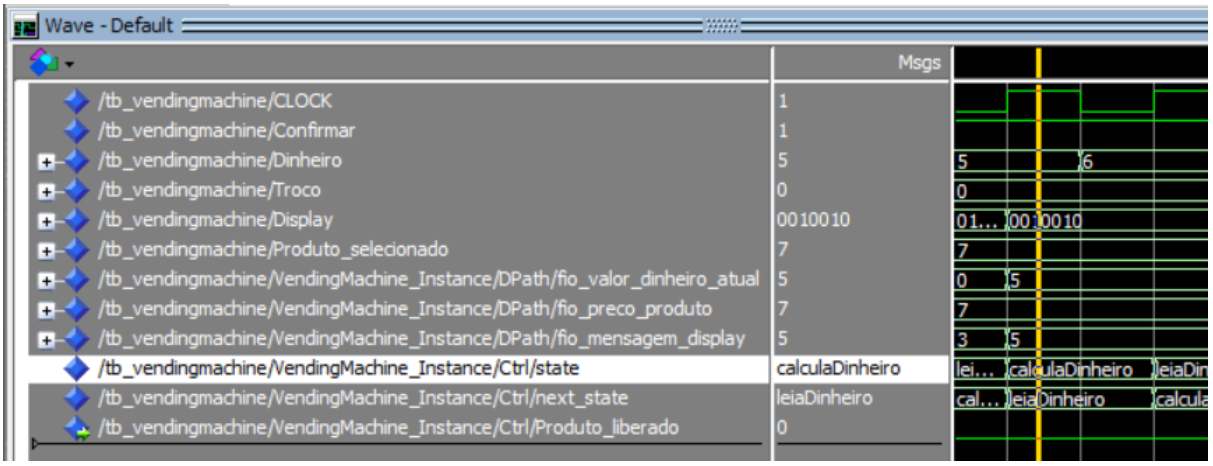
No estado de validação, o caminho de dados seleciona o preço do produto usando o componente ROM, que recebe como entrada o produto selecionado. Caso o produto selecionado seja ‘000’, ou seja, nenhum produto tiver sido selecionado de fato, a máquina continua no estado de validar produto. Como o produto foi selecionado (produto ‘111’), o preço do produto também foi selecionado, neste caso sendo igual a ‘111’ (7 em decimal).



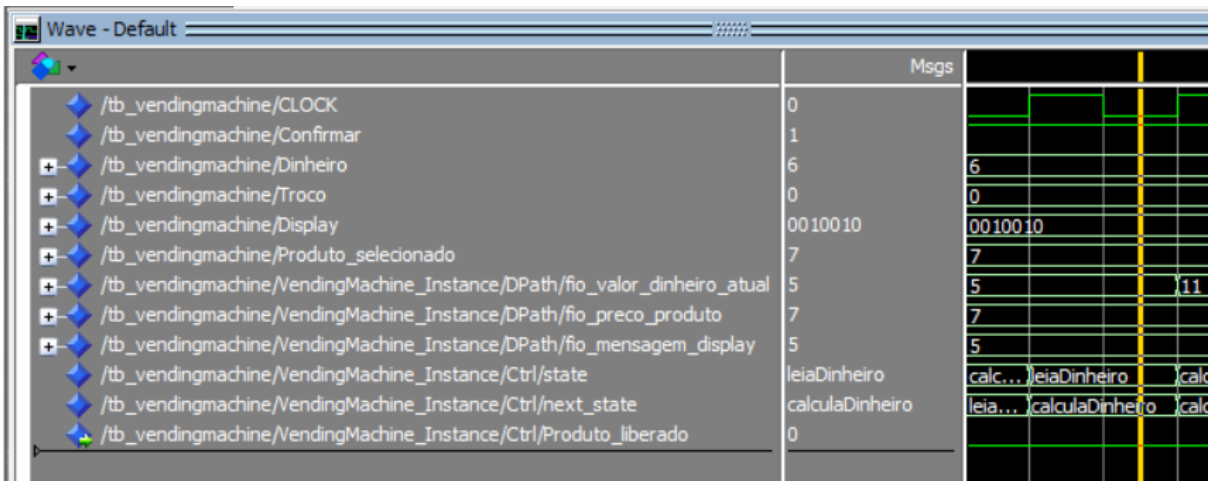
Em seguida, a máquina pede que o usuário confirme a compra e imprime a mensagem correspondente a este estado, a mensagem M3. Caso o usuário não confirmasse, a máquina seguiria para o estado “Cancela”, que limparia as entradas dos registradores e retornaria para o estado inicial. Neste caso, o usuário confirma a compra, portanto segue para o estado em que a máquina lê o dinheiro inserido pelo usuário.



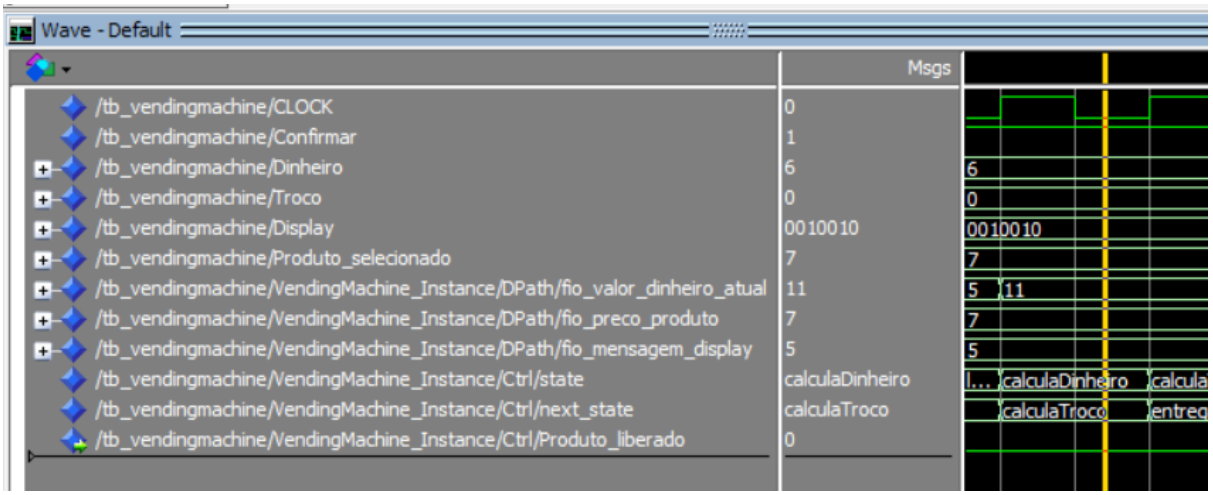
Neste estado, a máquina lê o dinheiro inserido pelo usuário e segue para o estado “Calcula Dinheiro”, que faz as devidas comparações entre o dinheiro inserido e o preço do produto.



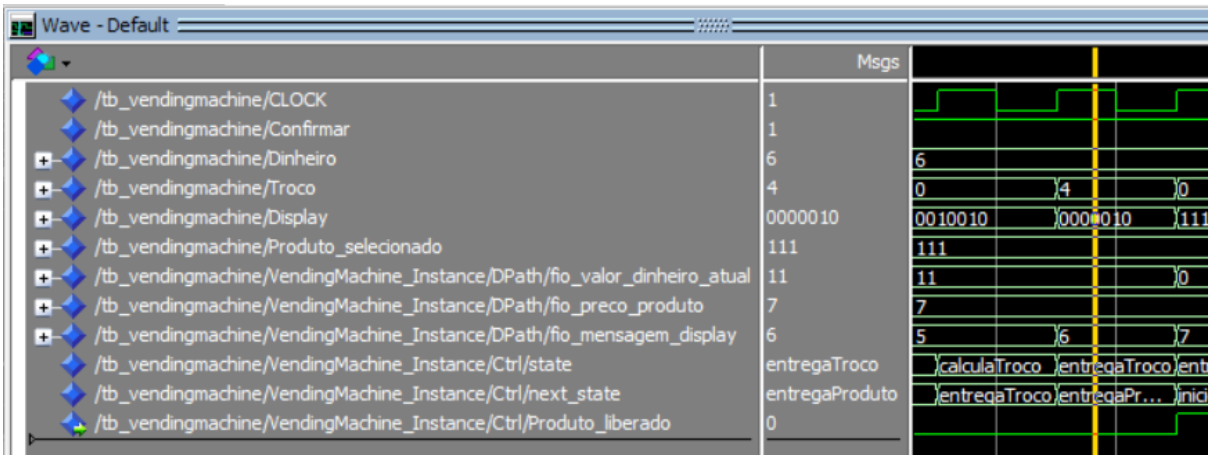
O estado “Calcula Dinheiro” verifica as seguintes possibilidades: se o dinheiro inserido é maior, menor ou igual ao preço do produto. Caso dinheiro inserido fosse igual, o próximo estado da máquina seria o estado de “Entrega Produto”, pulando o estado de calcular e entregar troco, que nesta situação são desnecessários. Neste caso, como o dinheiro inserido é de R$5,00 e o preço do produto é R$7,00, a máquina segue para o estado de “Ler Dinheiro” novamente, pois o valor inserido é insuficiente para a compra.



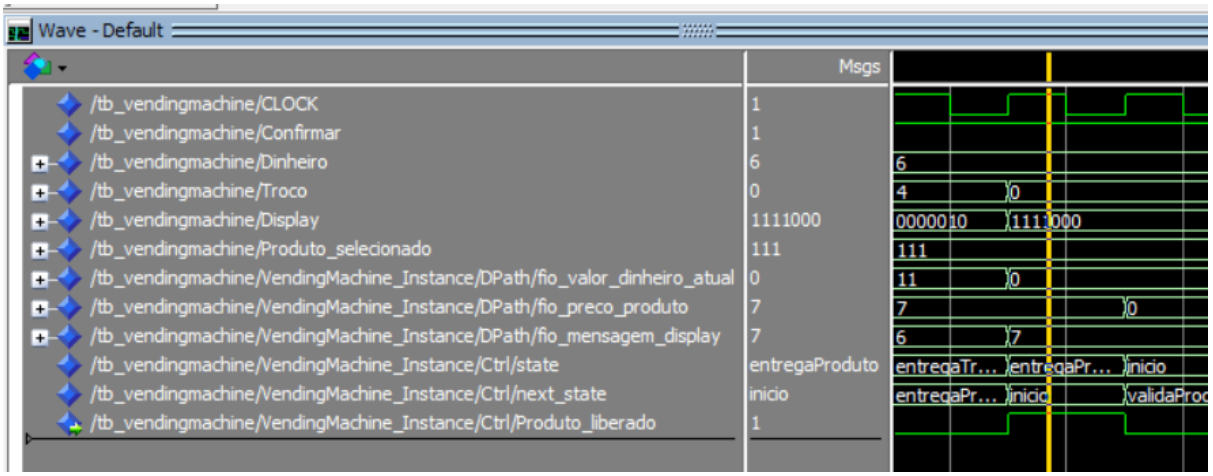
O usuário insere mais R$6,00, e de novo a máquina segue para o estado de “Calcula Dinheiro”, para avaliar a relação entre o total de dinheiro inserido e o preço do produto. A mensagem continua a mesma, solicitando ao usuário dinheiro (M5).



O estado “Calcula Troco”, então, faz a comparação entre o valor total do dinheiro inserido, que é R$11,00 (R$5,00 + R$6,00), com o preço do produto, que é R$7,00. Como dinheiro total inserido é maior que o preço do produto, a máquina segue para o estado de “Calcula Troco”.

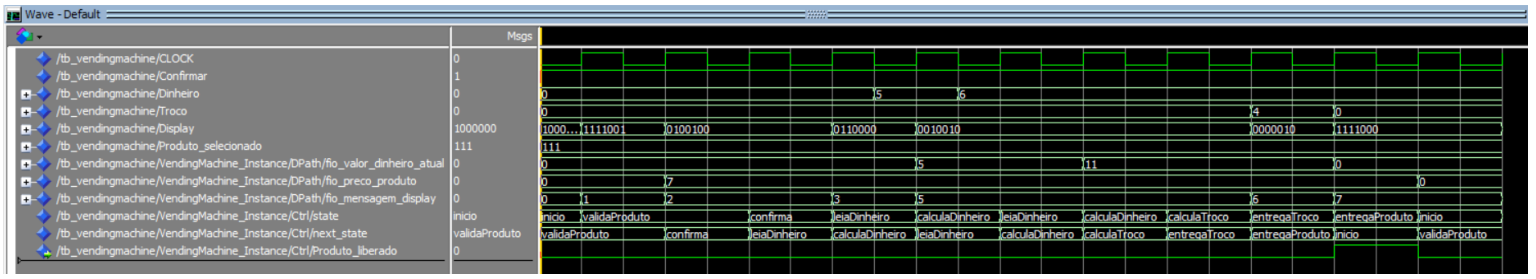


O estado “Calcula Troco” calcula o troco a partir do subtrator entre o dinheiro total inserido e o preço do produto, e em seguida passa para o estado que libera o troco, chamado “Entrega Troco”. Nele, a saída de troco vai de R$0,00 para R$4,00, que é o resultado da diferença (R$11,00 - R$7,00) entre o dinheiro inserido e o preço do produto, como dito anteriormente, e imprime a mensagem M6, que solicita ao usuário que retire o troco. A máquina então segue para o estado de “Entrega Produto”.



No estado de “Entrega Produto” os registradores são limpos, bem como a saída do Troco, e a saída “Produto_liberado” é acionada (muda para ‘1’), representando ao cliente que o produto está liberado para retirada. A mensagem M7 também é apresentada ao usuário, que solicita que o mesmo retire o produto. Em seguida, a máquina retorna ao estado inicial.

O testbench pode ser visto de forma geral logo abaixo.


Wave - Default
Msgs
/tb_vendingmachine/CLOCK
/tb_vendingmachine/Confirmar
/tb_vendingmachine/Dinheiro
/tb_vendingmachine/Troco
/tb_vendingmachine/Display
/tb_vendingmachine/Produto_selecionado
/tb_vendingmachine/VendingMachine_Instance/DPath/fio_valor_dinheiro_atual
/tb_vendingmachine/VendingMachine_Instance/DPath/fio_preco_produto
/tb_vendingmachine/VendingMachine_Instance/DPath/fio_mensagem_display
/tb_vendingmachine/VendingMachine_Instance/Ctrl/state
/tb_vendingmachine/VendingMachine_Instance/Ctrl/next_state
/tb_vendingmachine/VendingMachine_Instance/Ctrl/Produto_liberado
0
1
0
0
1000000
111
0
0
0
inicio
validaProduto
0
1000...
1111001
0100100
0110000
0010010
0000010
1111000
5
6
4
0
11
7
3
2
validaProduto
confirma
leiaDinheiro
calculaDinheiro
calculaTroco
entregaTroco
entregaProduto
inicio