Projeto sobre projeto em RTL com BO+BC

Problema 3 – máquina de vendas melhorada

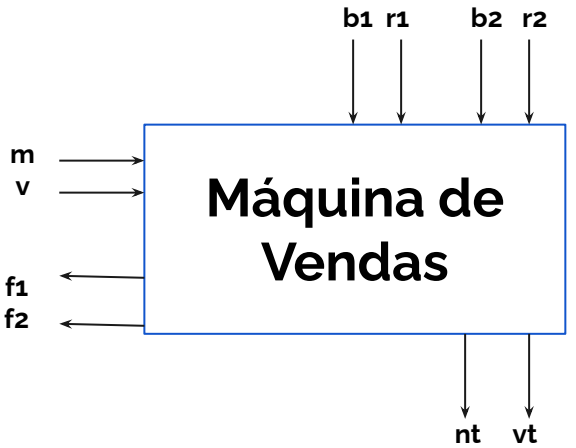
Equipe 0:

Frank B. Boa Morte (202006840007)
Fernando F. Dimas (201906840031)
Mercedes M. B. Diniz (201906840030)

Implementação do projeto

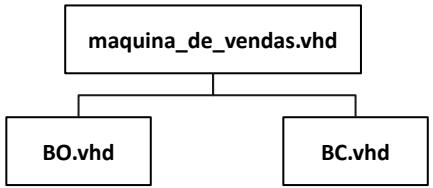
- **Entradas:**
  m - sinal do detector de moedas (1 bit)
  v - valor da moeda em centavos (8 bits)
  r1 - indica o custo do produto 1 (8 bits)
  r2 - indica o custo do produto 2 (8 bits)
  b1 - botão que seleciona o produto 1 (1 bit)
  b2 - botão que seleciona o produto 2 (1 bit)

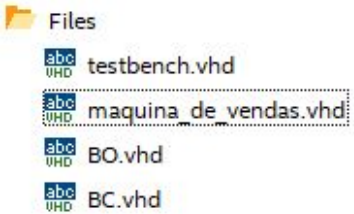
- **Saídas:**
  f1 - libera o produto 1 (1 bit)
  f2 - libera o produto 2 (1 bit)
  nt - indica a necessidade de troco (1 bit)
  vt - valor do troco (8 bits)

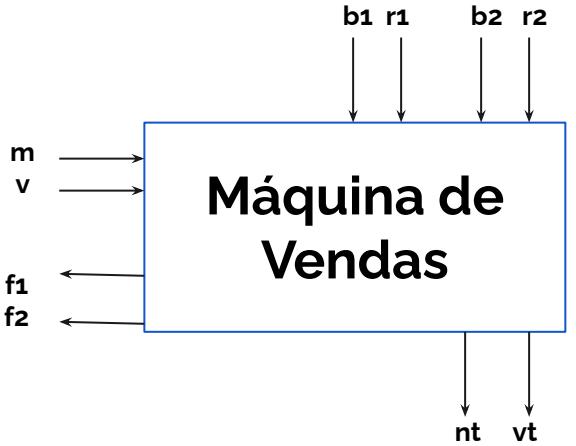


Implementação do projeto

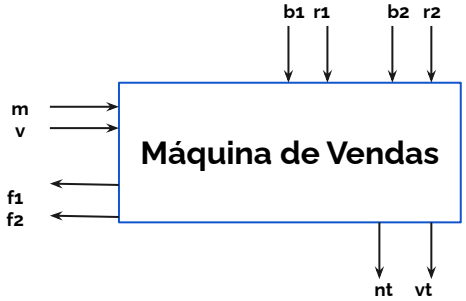
- **Hierarquia de Arquivos :**

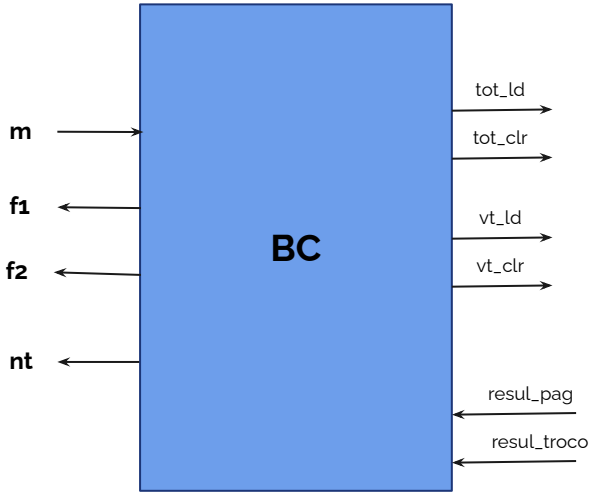


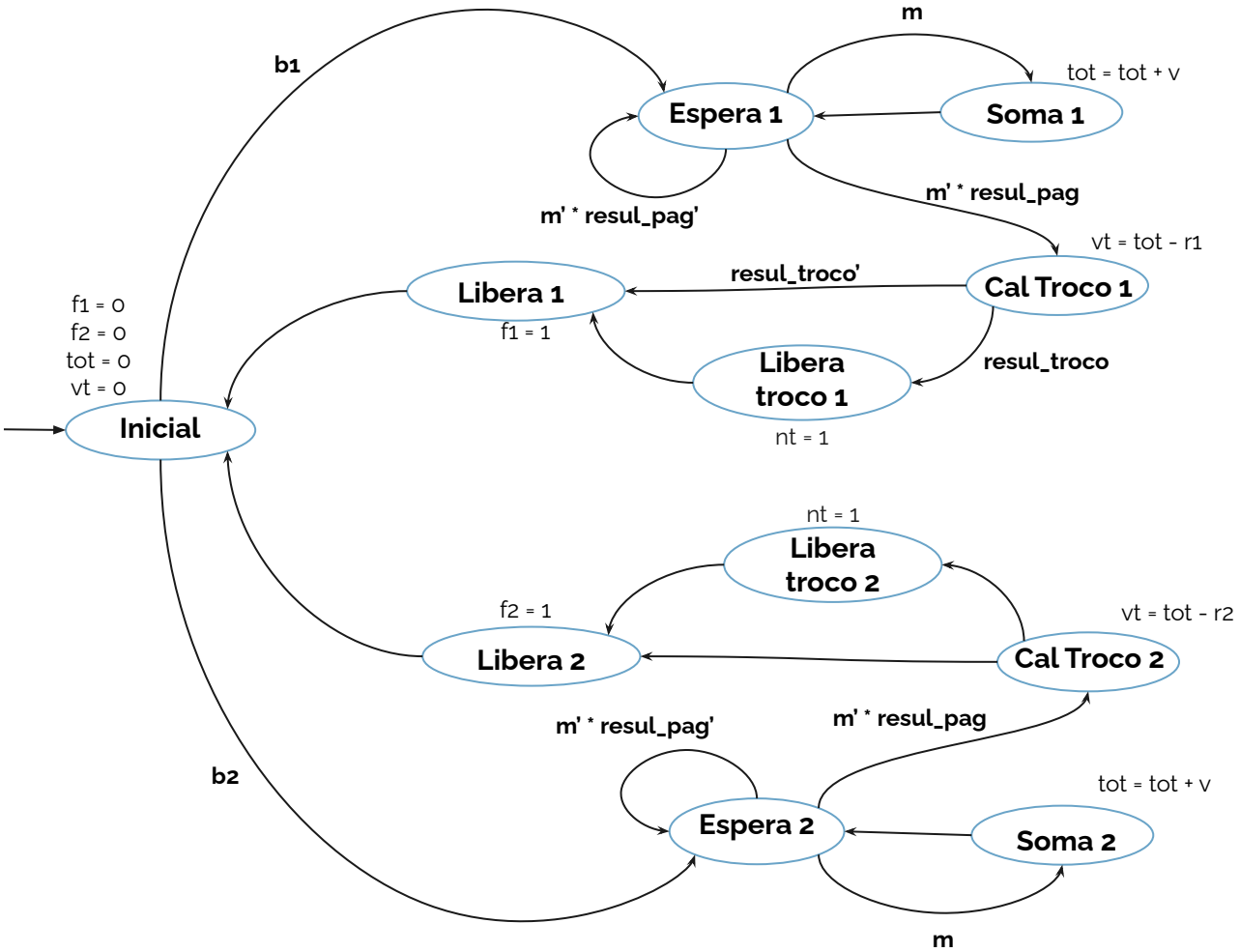




Bloco de Controle / Máquina de Estados






b1
m
tot = tot + v
Espera 1
Soma 1
m' * resul_pag'
m' * resul_pag
vt = tot - r1
Libera 1
resul_troco'
Cal Troco 1
f1 = 0
f2 = 0
tot = 0
vt = 0
f1 = 1
Libera
troco 1
resul_troco
Inicial
nt = 1
nt = 1
Libera
troco 2
f2 = 1
vt = tot - r2
Libera 2
Cal Troco 2
m' * resul_pag'
m' * resul_pag
b2
tot = tot + v
Espera 2
Soma 2
m

Bloco de Controle / Máquina de Estados

```
entity BC is
    -- Definndo as portas de entrada e saida
    port(
        clk, m, b1, b2, resul_pag, resul_troco : in std_logic;
        f1, f2, tot_ld, tot_clr, vt_ld, vt_clr, nt : out std_logic
    );
end BC;
```

- **Entradas:**
  m : detector de moeda
  b1 e b2 : seleciona o produto
  resul_pag : indica se o valor do produto já foi pago (tot<r?)
  resul_troco : indica se tem troco (vt>0?)

- **Saídas:**
  f1 e f2 : libera o produto
  tot_ld e vt_ld : habilita os registradores
  tot_clr e vt_clr : limpa os registradores

Bloco de Controle / Máquina de Estados

```
logica_proximo_estado : process(estado_atual, m, b1, b2, resul_pag, resul_troco)
begin
    case estado_atual is
        when Inicial =>
            if b1 = '1' and b2 = '0' then          -- selecional o produto 1
                prox_estado <= Espera1;

            elsif b1 = '0' and b2 = '1' then       -- selecional o produto 2
                prox_estado <= Espera2;

            else
                prox_estado <= Inicial;
            end if;
```

Bloco de Controle / Máquina de Estados

```
-- Estados do produto 1:
when Espera1 =>
    if m = '1' then                                  -- moeda inserida
        prox_estado <= Soma1;

    elsif m = '0' and resul_pag = '1' then           -- terminou de paga
        prox_estado <= Cal_troco1;

    else                                             -- não terminou de paga
        prox_estado <= Espera1;
    end if;

when Soma1 =>                                  -- o BO faz tot = tot + v p/ calcular o total de dinheiro inserido
        prox_estado <= Espera1;

when Cal_troco1 =>                                   -- o BO faz vt = tot-r p/ calcular o possivel troco
    if resul_troco = '1' then -- tem troco (vt>0)
        prox_estado <= Libera_troco1;
    else
        prox_estado <= Libera_prod1;
    end if;

when Libera_troco1 =>
    prox_estado <= Libera_prod1;

when Libera_prod1 =>
    prox_estado <= Inicial;
```

Bloco de Controle / Máquina de Estados

```
    -- Estados do produto 2:
    when Espera2 =>
        if m = '1' then                                  -- moeda inserida
            prox_estado <= Soma2;

        elsif m = '0' and resul_pag = '1' then           -- terminou de paga
            prox_estado <= Cal_troco2;

        else                                             -- não terminou de paga
            prox_estado <= Espera2;
        end if;

    when Soma2 =>                                        -- o BO faz tot = tot + v p/ calcular o total de dinheiro inserido
            prox_estado <= Espera2;

    when Cal_troco2 =>                                   -- o BO faz vt = tot-r p/ calcular o possivel troco
        if resul_troco = '1' then -- tem troco (vt>0)
            prox_estado <= Libera_troco2;
        else
            prox_estado <= Libera_prod2;
        end if;

    when Libera_troco2 =>
        prox_estado <= Libera_prod2;

    when Libera_prod2 =>
        prox_estado <= Inicial;
  end case;
end process;
```

Bloco de Controle / Máquina de Estados

```
logica_saida : process(estado_atual)
begin
    case estado_atual is
    when Inicial =>
        f1 <= '0';
        f2 <= '0';
        tot_ld <= '0';
        tot_clr <= '1';    -- limpa registrador tot
        vt_ld <= '0';
        vt_clr <= '1';     -- limpa registrador vt
        nt <= '0';
```

Bloco de Controle / Máquina de Estados

```
-- Estados dodo a seleção do produto 1:
when Espera1 =>
    f1 <= '0';
    f2 <= '0';
    tot_ld <= '0';      -- habilita registrador tot
    tot_clr <= '0';
    vt_ld <= '0';
    vt_clr <= '1';      -- limpa registrador vt
    nt <= '0';

when Soma1 =>
    f1 <= '0';
    f2 <= '0';
    tot_ld <= '1';      -- habilita registrador tot
    tot_clr <= '0';
    vt_ld <= '0';
    vt_clr <= '1';      -- limpa registrador vt
    nt <= '0';

when Cal_troco1 =>
    f1 <= '0';
    f2 <= '0';
    tot_ld <= '0';      -- habilita registrador tot
    tot_clr <= '0';
    vt_ld <= '1';       -- habilita registrador vt
    vt_clr <= '0';
    nt <= '0';
```

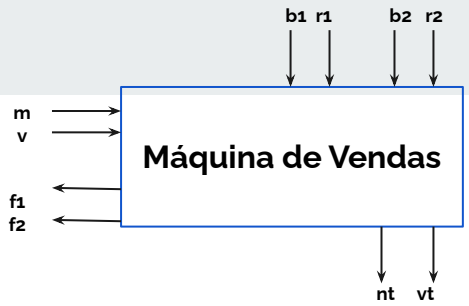
```
when Libera_troco1 =>
    f1 <= '0';
    f2 <= '0';
    tot_ld <= '0';
    tot_clr <= '1';     -- limpa registrador tot
    vt_ld <= '1';       -- habilita registrador vt
    vt_clr <= '0';
    nt <= '1';          -- indica que há troco (sendo o valor vt)

when Libera_prod1 =>
    f1 <= '1';          -- libera o produto 1
    f2 <= '0';
    tot_ld <= '0';
    tot_clr <= '1';     -- limpa registrador tot
    vt_ld <= '0';
    vt_clr <= '0';
    nt <= '0';
```

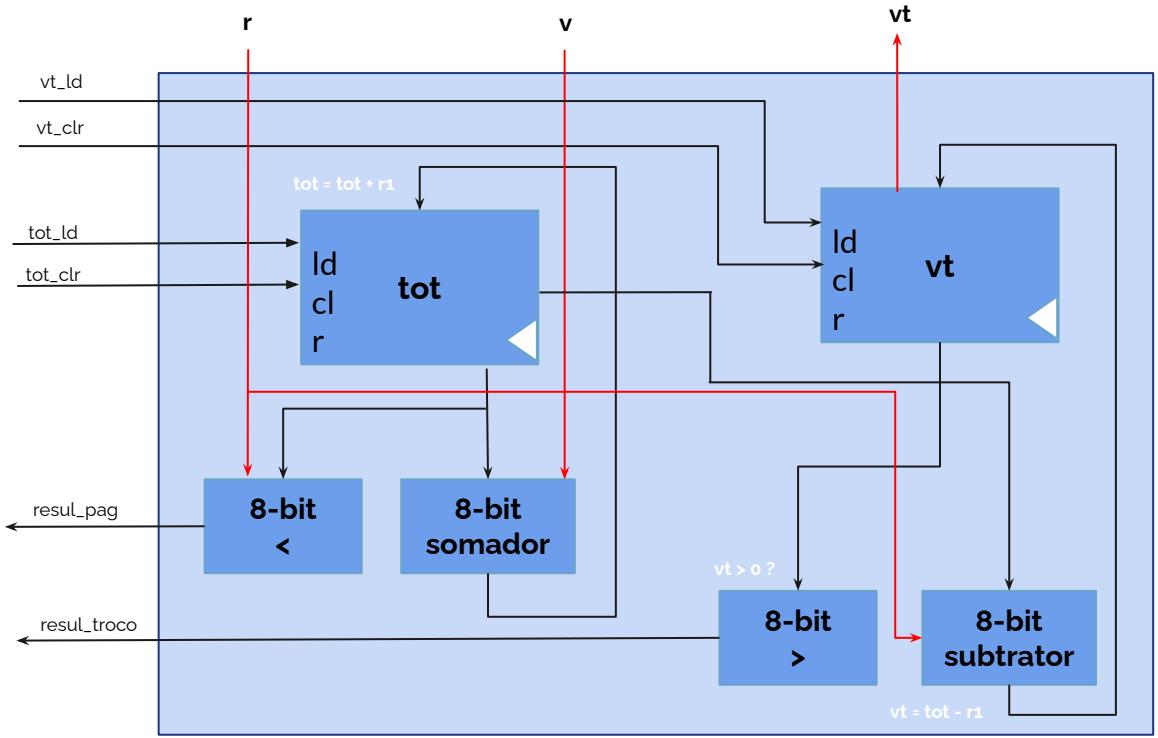
Bloco de Controle / Máquina de Estados

```
-- Estados dodo a seleção do produto 2:
when Espera2 =>
    f1 <= '0';
    f2 <= '0';
    tot_ld <= '0';      -- habilita registrador tot
    tot_clr <= '0';
    vt_ld <= '0';
    vt_clr <= '1';      -- limpa registrador vt
    nt <= '0';

when Soma2 =>
    f1 <= '0';
    f2 <= '0';
    tot_ld <= '1';      -- habilita registrador tot
    tot_clr <= '0';
    vt_ld <= '0';
    vt_clr <= '1';      -- limpa registrador vt
    nt <= '0';

when Cal_troco2 =>
    f1 <= '0';
    f2 <= '0';
    tot_ld <= '0';      -- habilita registrador tot
    tot_clr <= '0';
    vt_ld <= '1';       -- habilita registrador vt
    vt_clr <= '0';
    nt <= '0';
```

```
when Libera_troco2 =>
    f1 <= '0';
    f2 <= '0';
    tot_ld <= '0';
    tot_clr <= '1';     -- limpa registrador tot
    vt_ld <= '1';       -- habilita registrador vt
    vt_clr <= '0';
    nt <= '1';          -- indica que há troco (sendo o valor vt)

when Libera_prod2 =>
    f1 <= '0';
    f2 <= '1';          -- libera o produto 2
    tot_ld <= '0';
    tot_clr <= '1';     -- limpa registrador tot
    vt_ld <= '0';
    vt_clr <= '0';
    nt <= '0';
```

Bloco Operacional





Bloco Operacional

```
entity BO is
    -- Definndo as portas de entrada e saida
    port(
        clk, tot_ld, tot_clr, vt_ld, vt_clr, b1, b2 : in std_logic;
        r1, r2, v : in std_logic_vector(7 downto 0);
        resul_pag, resul_troco : out std_logic;
        val_troco : out std_logic_vector(7 downto 0)
    );
end BO;
```

- **Entradas:**
  v : valor da moeda inserida
  b1 e b2 : seleciona o produto
  r1 e r2 : preço dos produtos
  tot_ld e vt_ld : habilita os registradores
  tot_clr e vt_clr : limpa os registradores

- **Saídas:**
  resul_pag : indica se o valor do produto já foi pago (tot<r?)
  resul_troco : indica se tem troco (vt>0?)
  val_troco : valor do troco (zero se não houver)

Bloco Operacional

```
sel_produto : process(b1, b2, r1, r2)
begin
    if b1 = '1' and b2 = '0' then -- selecionou o produto 1
        r <= r1;
    elsif b1 = '0' and b2 = '1' then -- selecionou o produto 2
        r <= r2;
    else
        r <= (others => '0');    -- limpa
    end if;
end process;
```

Bloco Operacional

```
registrador_tot : process(clk, tot_ld, tot_clr, resul_soma, val_tot)
begin
    if tot_clr = '1' then
        val_tot <= (others => '0');    -- limpa

    elsif rising_edge(clk) then
        if tot_ld = '1' then
            val_tot <= resul_soma;     -- salva
        else
            val_tot <= val_tot;        -- conserva o valor
        end if;

    end if;
end process;

registrador_vt : process(val_tot, r, vt_ld, vt_clr, resul_subt, val_vt)
begin
    if vt_clr = '1' then
        val_vt <= (others => '0');     -- limpa

    elsif val_tot >= r then
        if vt_ld = '1' then
            val_vt <= resul_subt;      -- salva
        else
            val_vt <= val_vt;        -- conserva o valor
        end if;

    end if;
end process;
```

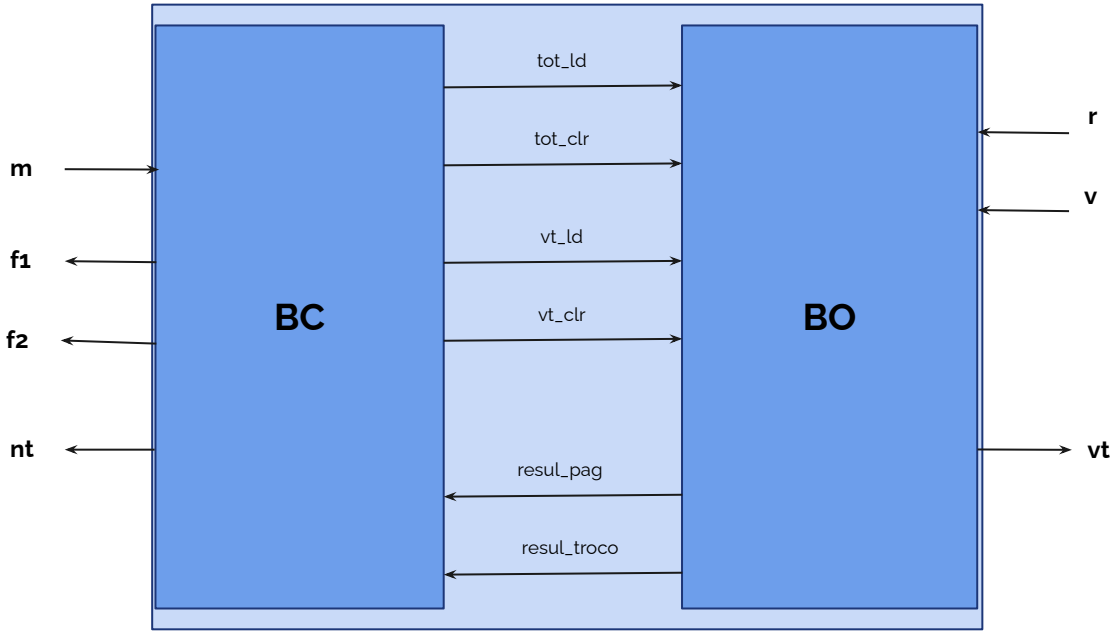
Bloco Operacional

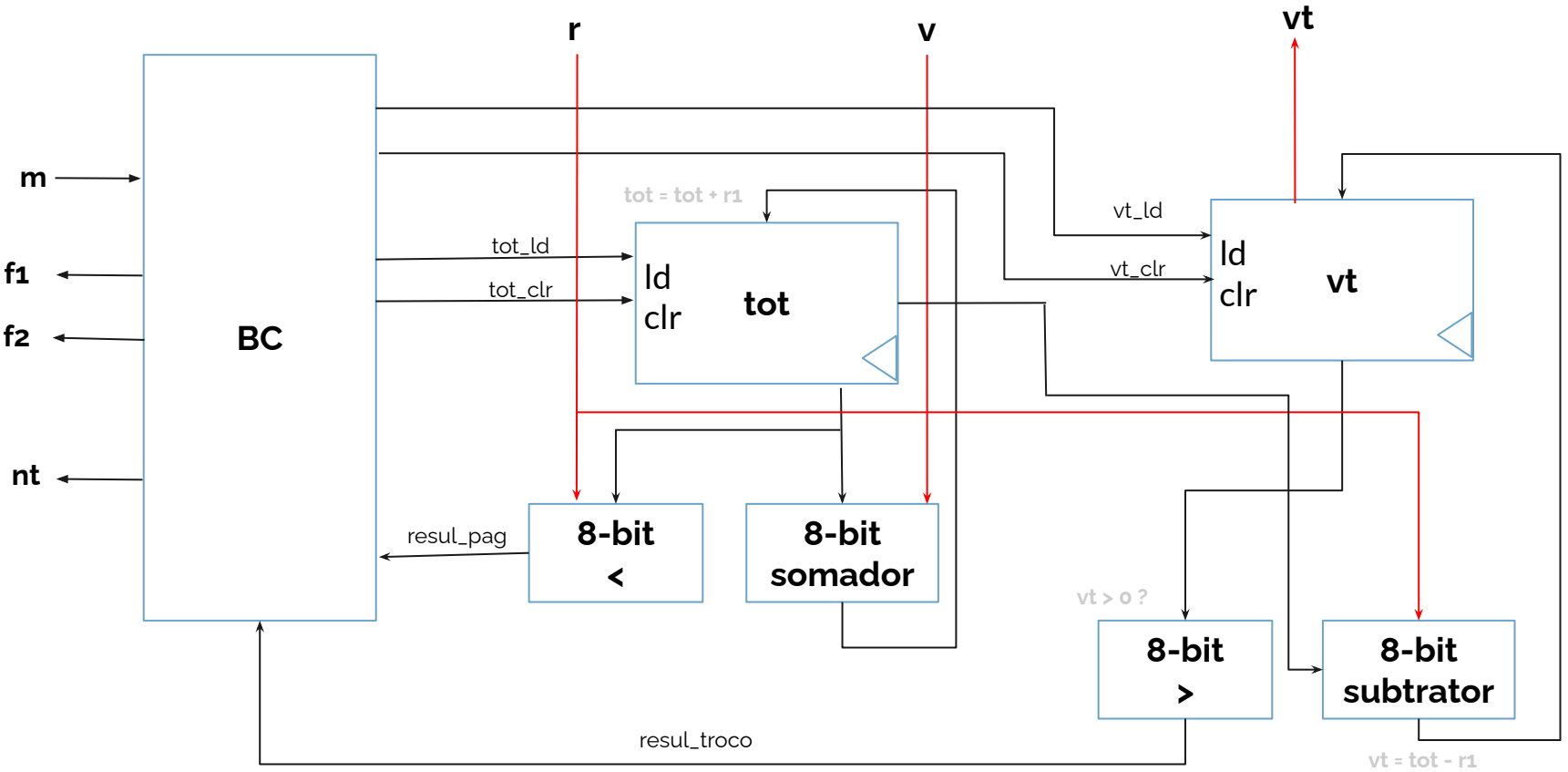
```
somador : process(val_tot, v)
begin
    resul_soma <= std_logic_vector(unsigned(val_tot) + unsigned(v));
    -- resul_soma
end process;

verifica_pagamento : process(val_tot, r)
begin
    if val_tot >= r then
        resul_pag <= '1';
    else
        resul_pag <= '0';
    end if;
    -- resul_pag
end process;
```

Bloco Operacional

```
subtrator : process(val_tot, r)
begin
    resul_subt <= std_logic_vector(unsigned(val_tot) - unsigned(r));
    -- resul_subt
end process;

verifica_troco : process(val_vt)
begin
    if val_vt > "00000000" then
        resul_troco <= '1';
        val_troco <= val_vt;
    else
        resul_troco <= '0';
        val_troco <= "00000000";
    end if;
    -- resul_troco
end process;
```

Máquina de Vendas (Interconexão entre o BC e o BO)




r
v
vt
m
f1
f2
nt
BC
tot = tot + r1
tot_ld
tot_clr
ld
clr
tot
vt_ld
vt_clr
ld
clr
vt
resul_pag
8-bit
<
8-bit
somador
vt > 0 ?
8-bit
>
8-bit
subtrator
resul_troco
vt = tot - r1

Máquina de Vendas (Interconexão entre o BC e o BO)

```
entity maquina_de_vendas is
    port(
        Clk, M, B1, B2 : in std_logic;
        V, R1, R2 : in std_logic_vector(7 downto 0);
        F1, F2, NT : out std_logic;
        VT : out std_logic_vector(7 downto 0)
    );
end maquina_de_vendas;
```

Máquina de Vendas (Interconexão entre o BC e o BO)

```
-- Região de declaração:

-- Bloco de Controle
component BC is
    port(
        clk, m, b1, b2, resul_pag, resul_troco : in std_logic;
        f1, f2, tot_ld, tot_clr, vt_ld, vt_clr, nt : out std_logic
    );
end component;

-- Bloco Oeracional
component BO is
    port(
        clk, tot_ld, tot_clr, vt_ld, vt_clr, b1, b2 : in std_logic;
        r1, r2, v : in std_logic_vector(7 downto 0);
        resul_pag, resul_troco : out std_logic;
        val_troco : out std_logic_vector(7 downto 0)
    );
end component;

-- Declaração das constantes e variaveis:
signal resul_pag_aux, resul_troco_aux : std_logic;
signal tot_ld_aux, tot_clr_aux, vt_ld_aux, vt_clr_aux : std_logic;
```

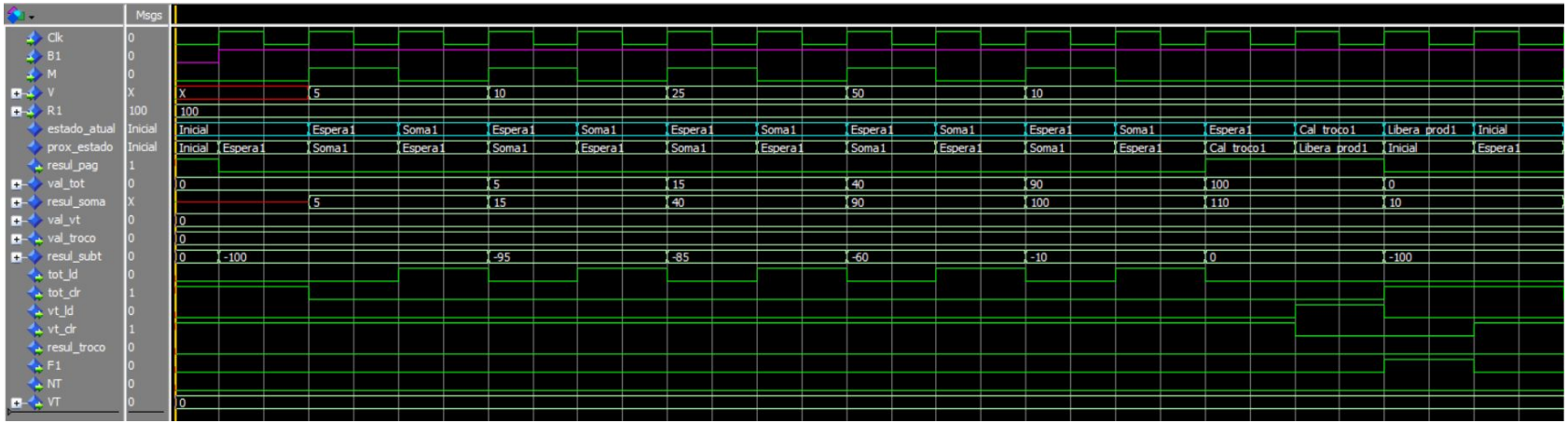
Máquina de Vendas (Interconexão entre o BC e o BO)

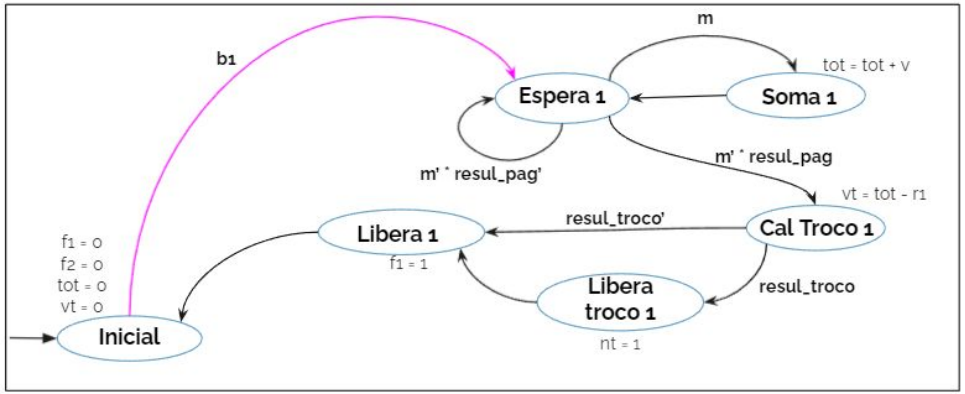
```
-- Instanciando o Bloco de Controle
inst_BC : BC
    port map(
        clk => Clk,
        m => M,
        b1 => B1,
        b2 => B2,
        resul_pag => resul_pag_aux,
        resul_troco => resul_troco_aux,
        f1 => F1,
        f2 => F2,
        tot_ld => tot_ld_aux,
        tot_clr => tot_clr_aux,
        vt_ld => vt_ld_aux,
        vt_clr => vt_clr_aux,
        nt => NT
    );
```

```
-- Instanciando o Bloco Operacional
inst_BO : BO
    port map(
        clk => Clk,
        tot_ld => tot_ld_aux,
        tot_clr => tot_clr_aux,
        vt_ld => vt_ld_aux,
        vt_clr => vt_clr_aux,
        b1 => B1,
        b2 => B2,
        r1 => R1,
        r2 => R2,
        v => V,
        resul_pag => resul_pag_aux,
        resul_troco => resul_troco_aux,
        val_troco => VT
    );
```

Testbench Produto 1

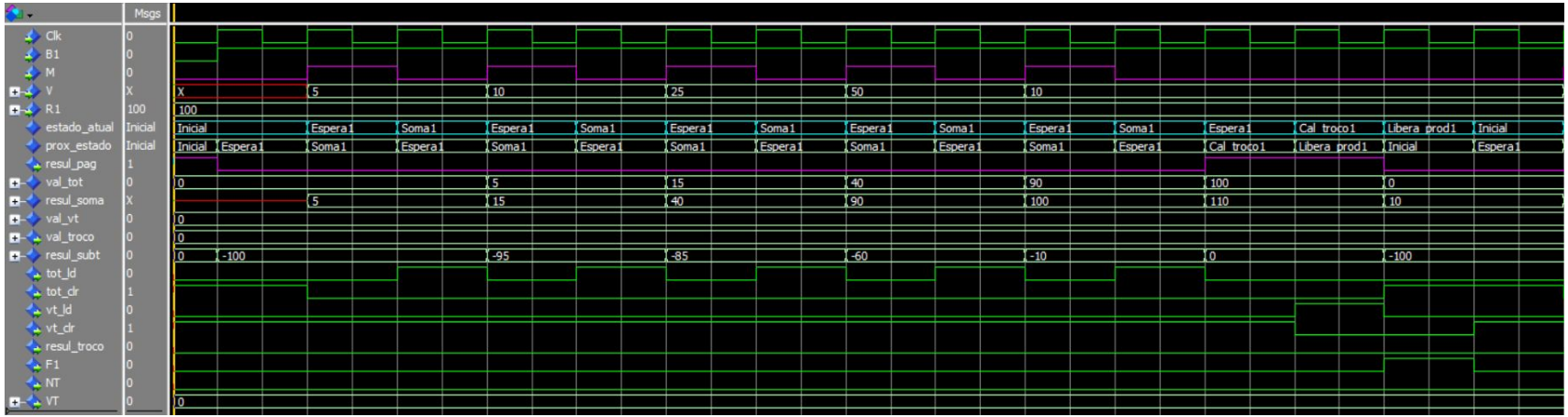
Estado Inicial:

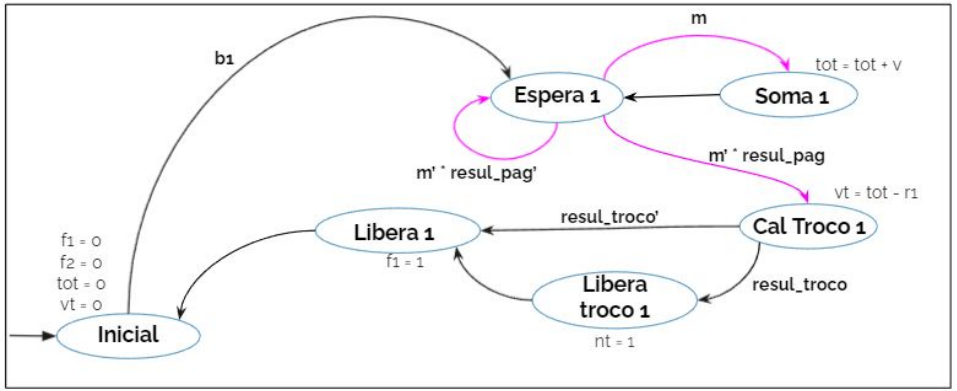




Testbench Produto 1

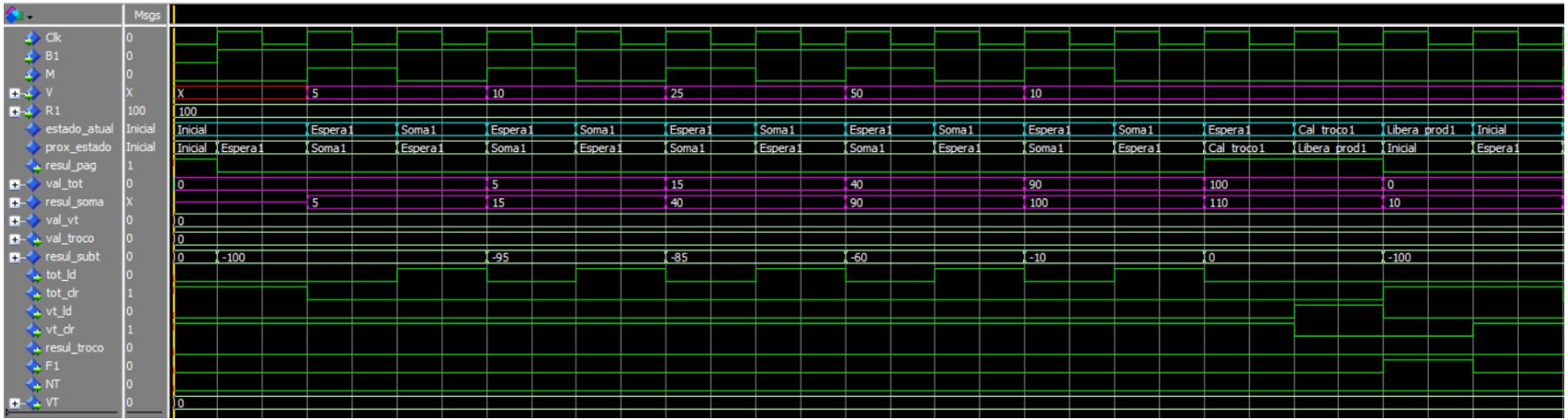
Estado Espera1:

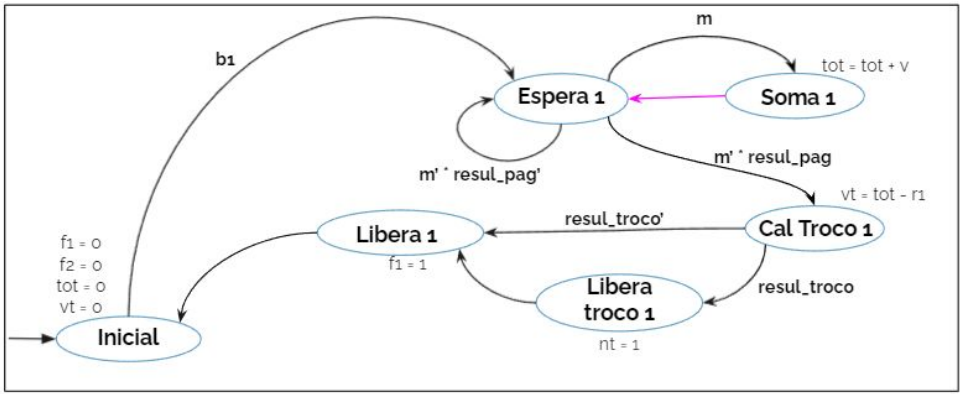




Testbench Produto 1

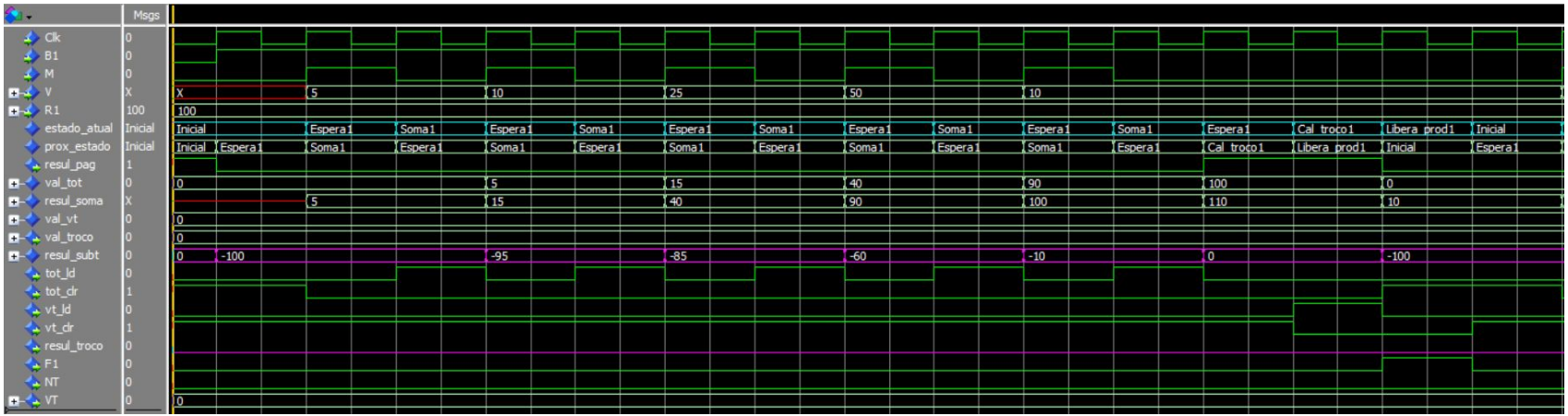
Estado Soma1:

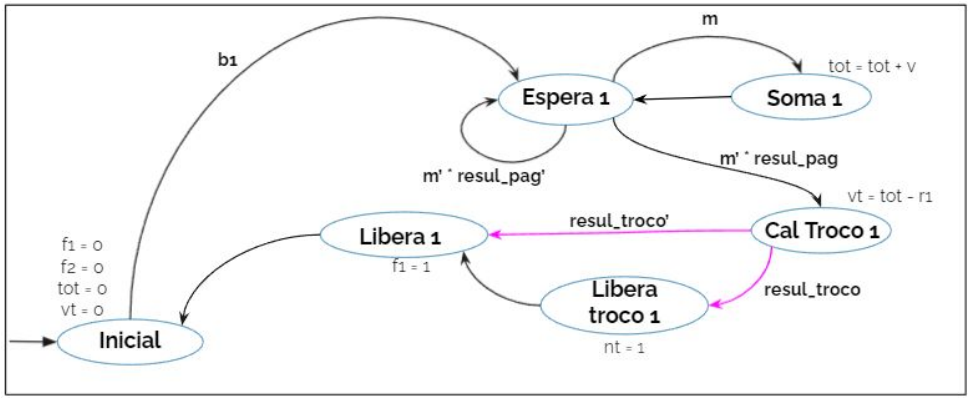




Testbench Produto 1

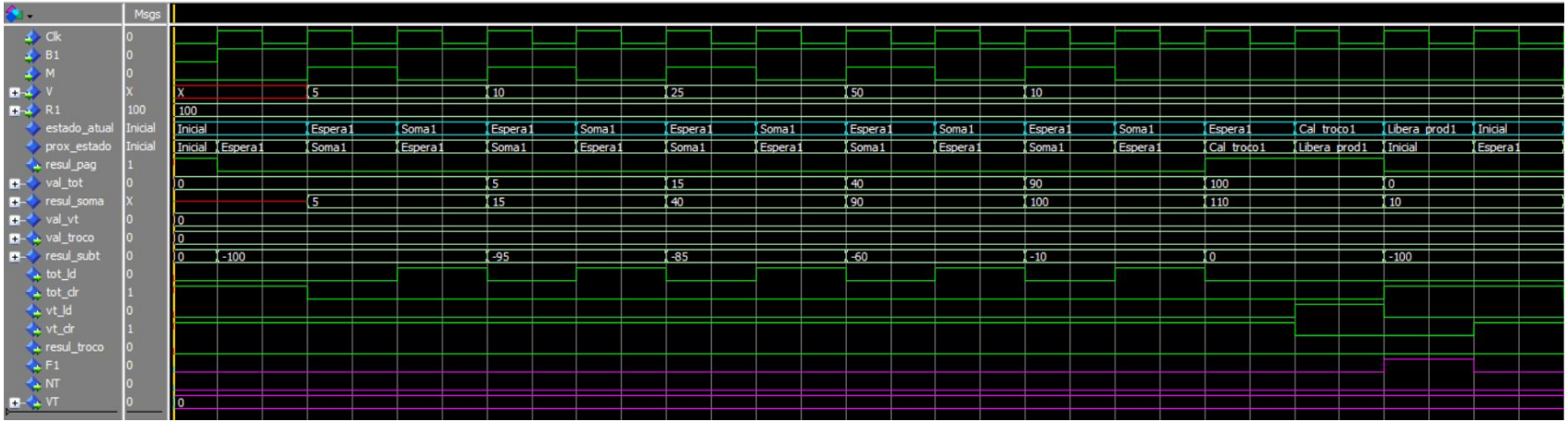
Estado Cal_Troco1:

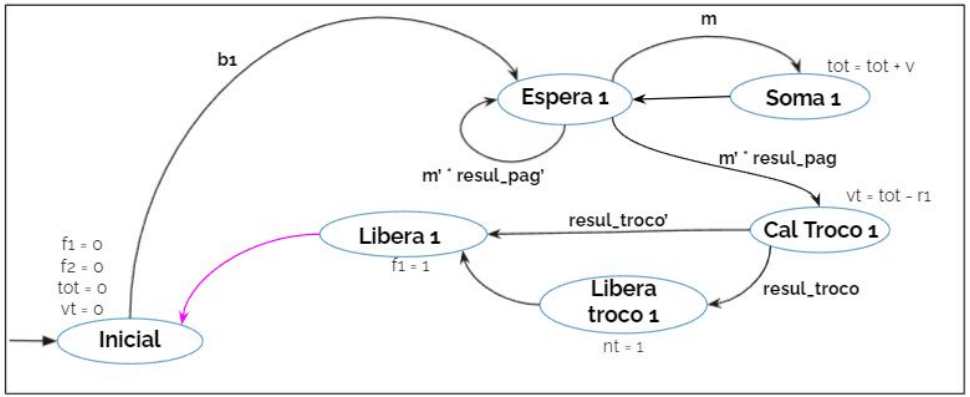




Testbench Produto 1

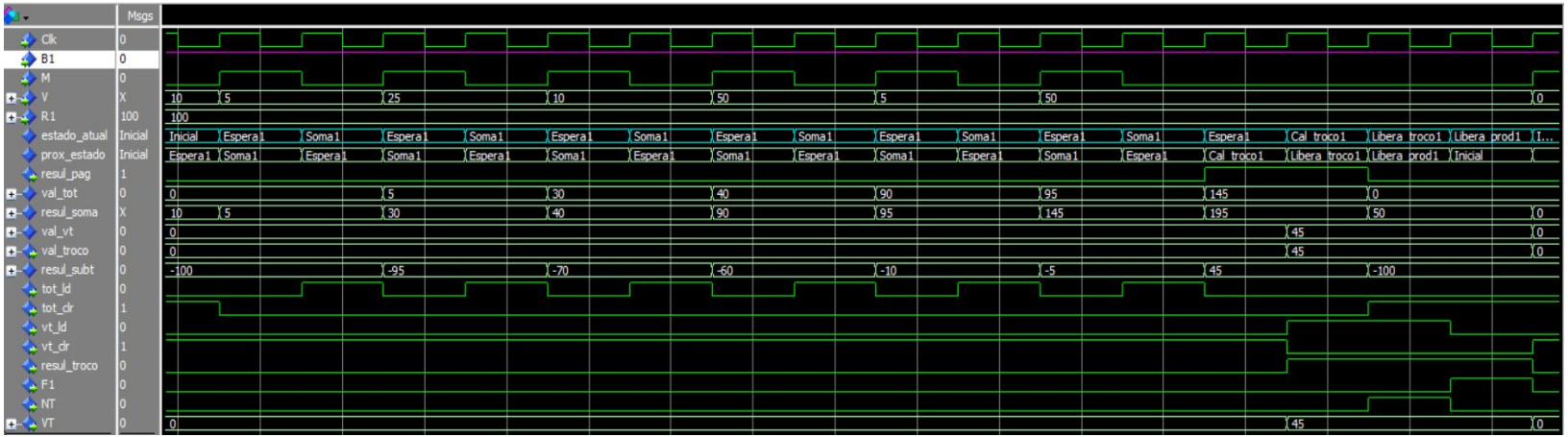
Estado Libera_Prod1:

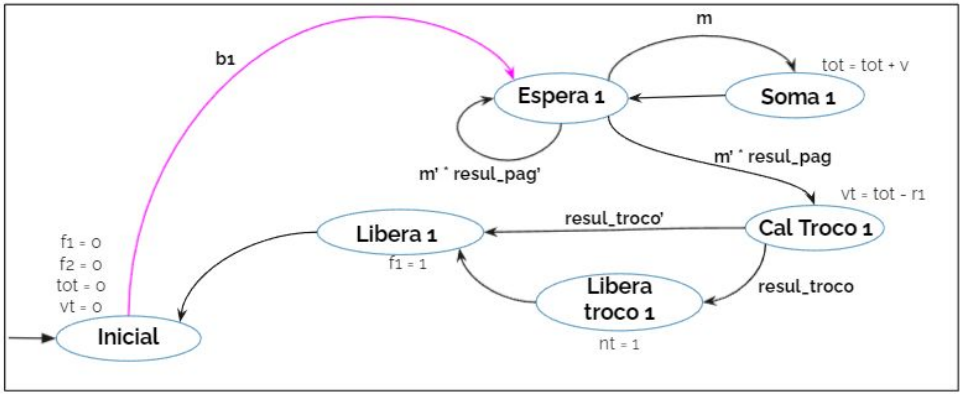




Testbench Produto 1

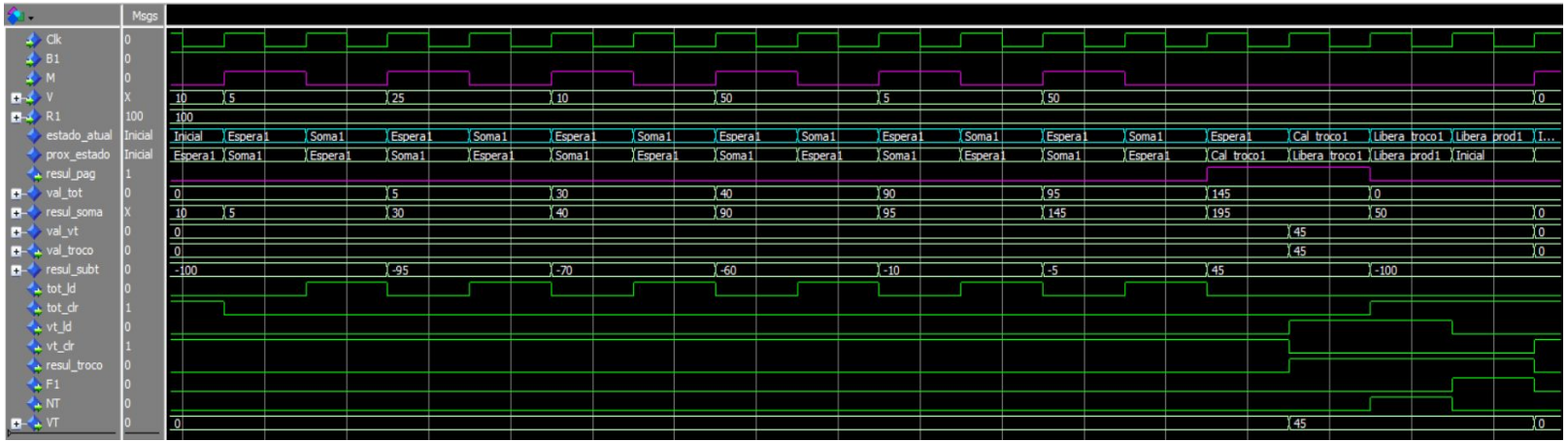
Estado Inicial:

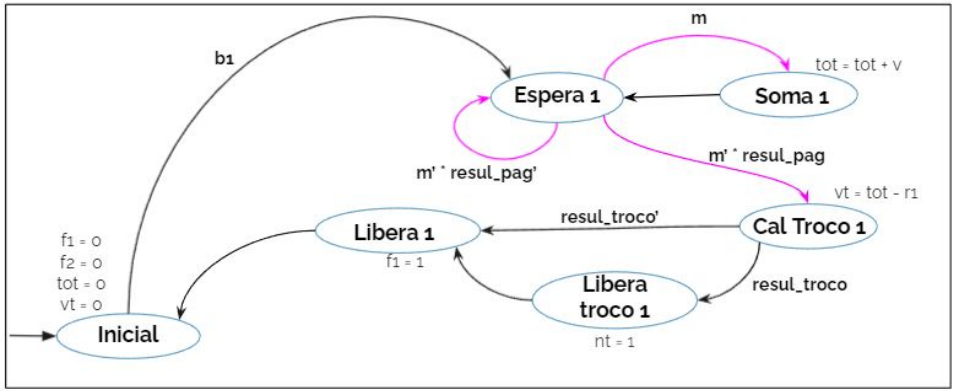




Testbench Produto 1

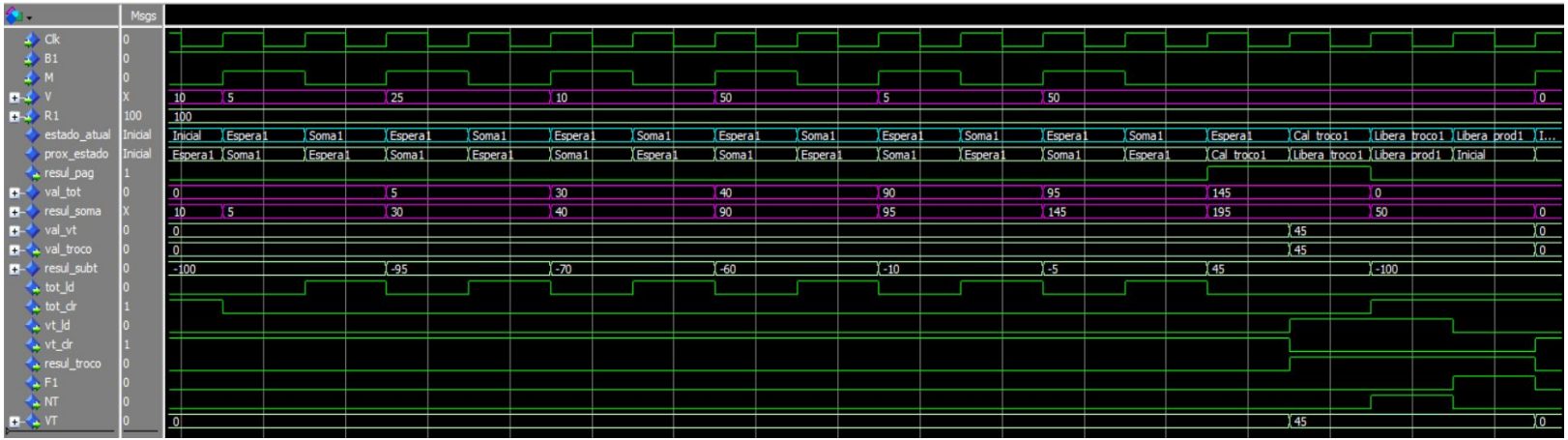
Estado Espera1:

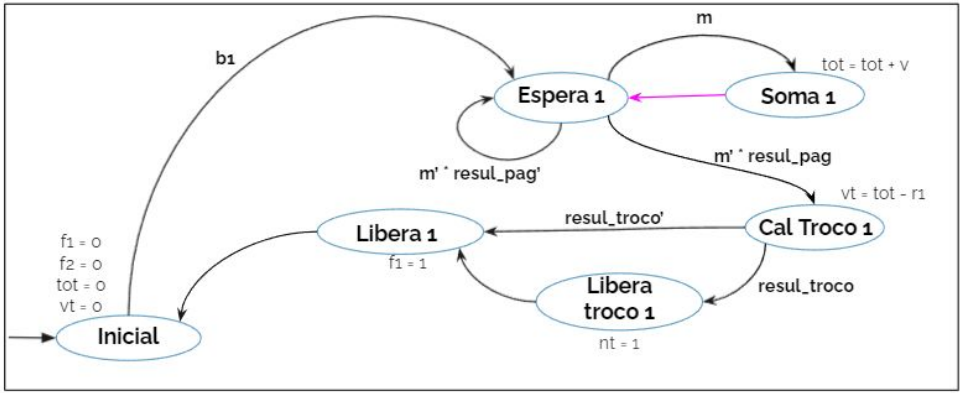




Testbench Produto 1

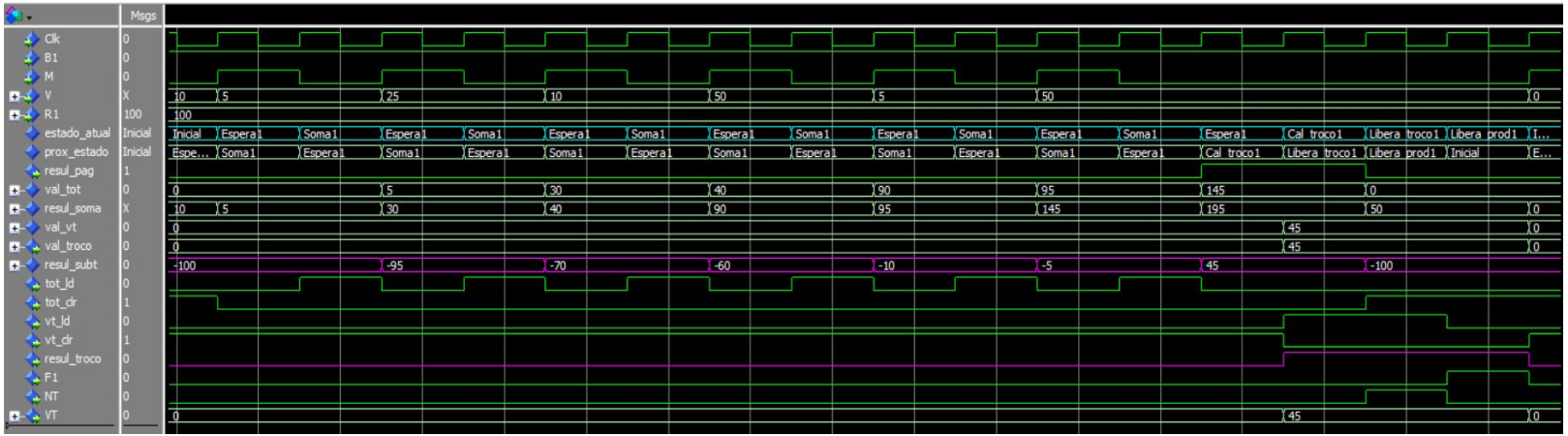
Estado Soma1:

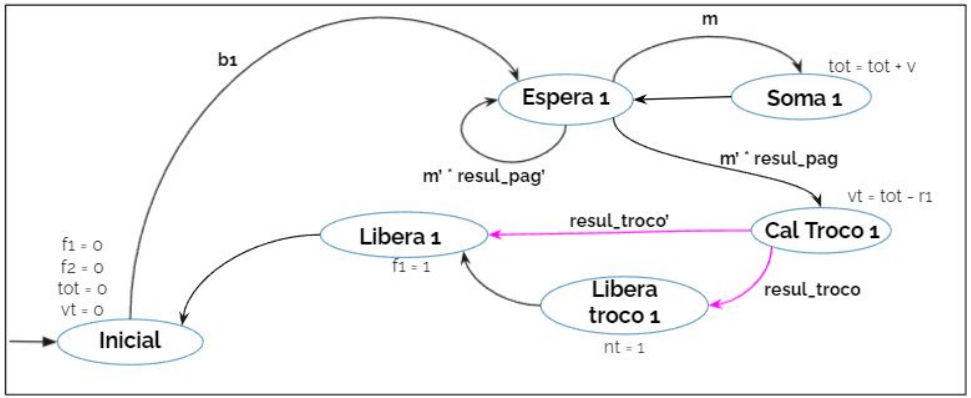




Testbench Produto 1

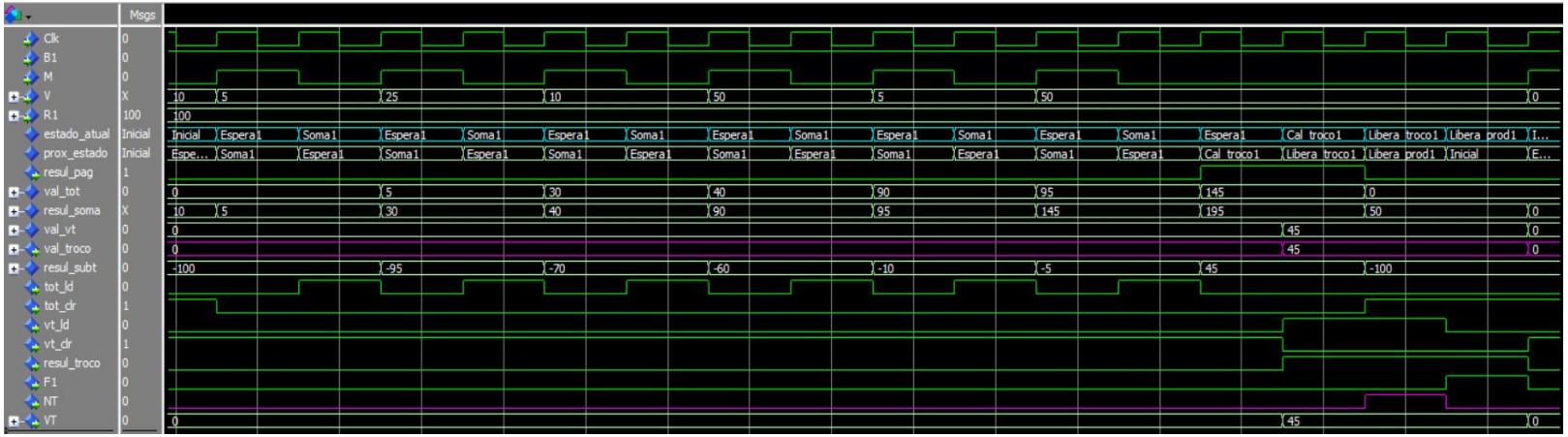
Estado Cal_Troco1:

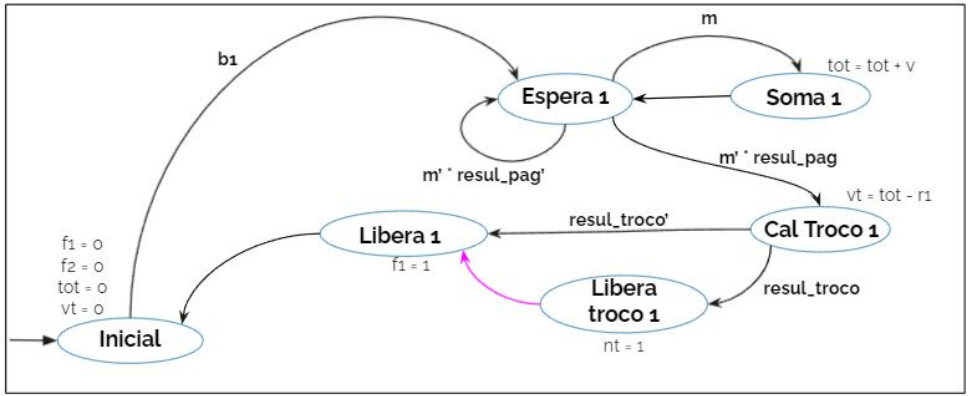




Testbench Produto 1

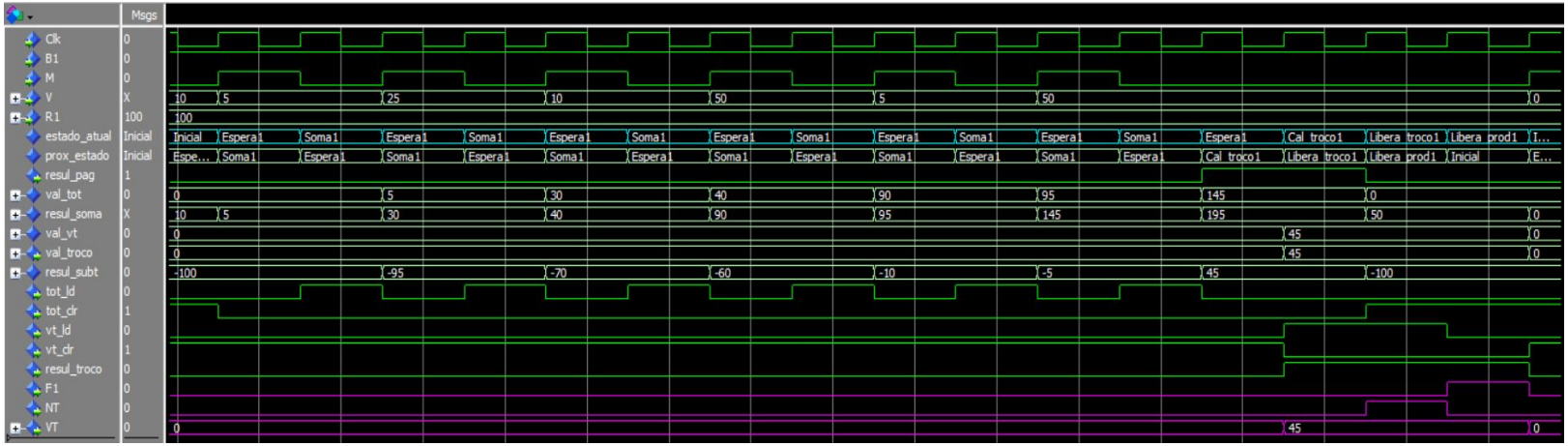
Estado Libera_troco1:

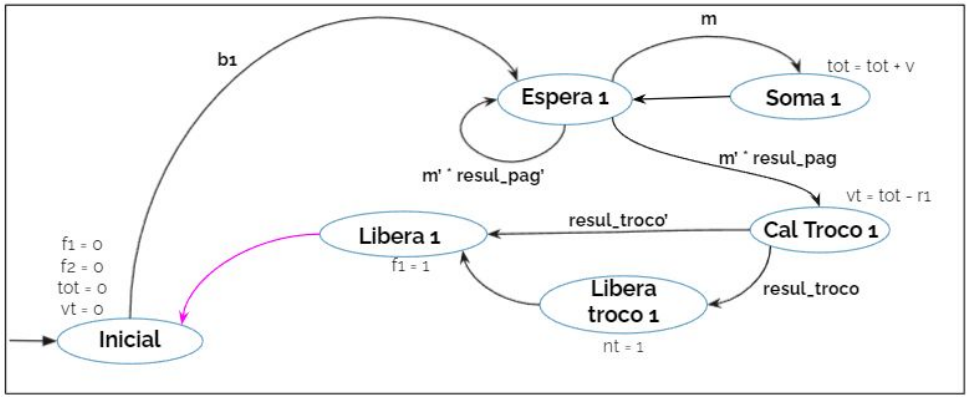




Testbench Produto 1

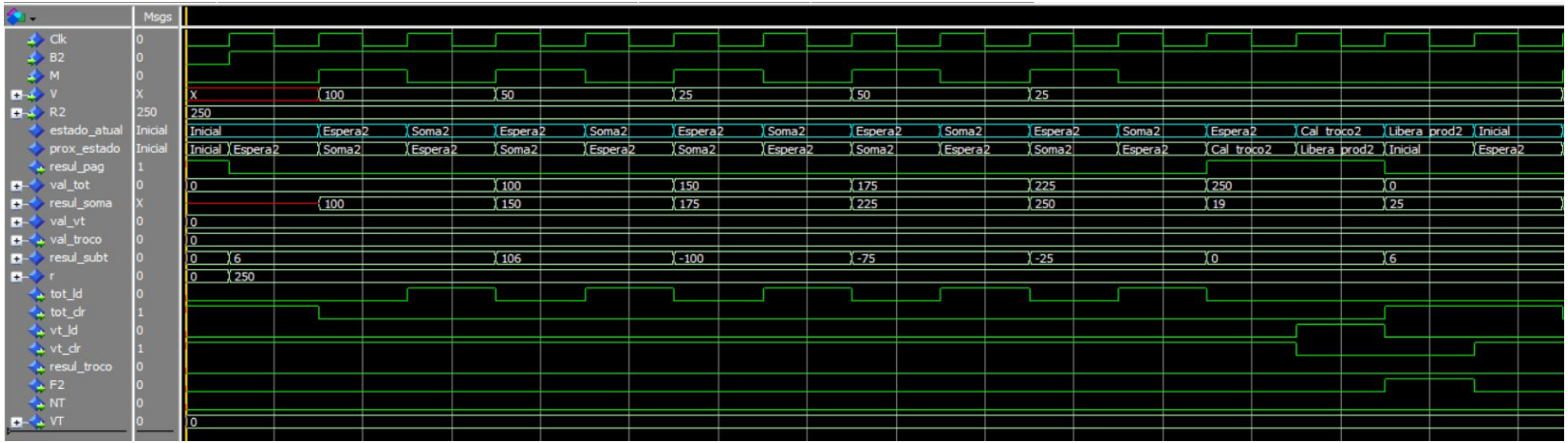
Estado Libera_Prod1:

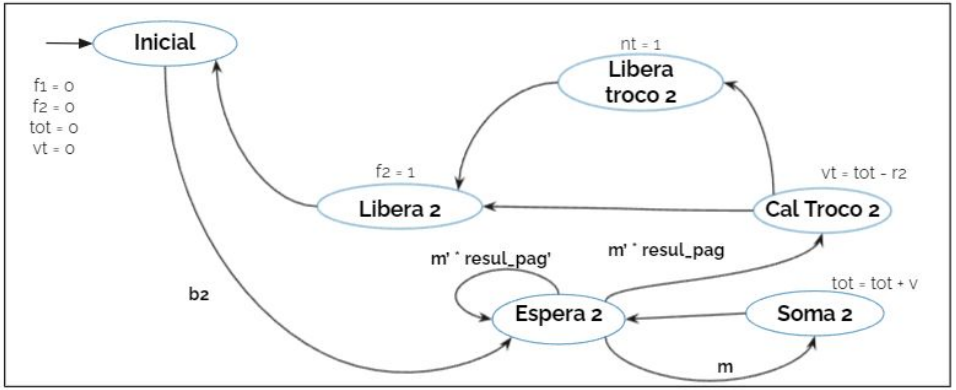




Testbench Produto 2

Estados:





Testbench

```
entity testbench is
end testbench;

architecture comportamento of testbench is
    -- Declarando o componente main
    component maquina_de_vendas is
    port(
        Clk, M, B1, B2 : in std_logic;
        V, R1, R2 : in std_logic_vector(7 downto 0);
        F1, F2, NT : out std_logic;
        VT : out std_logic_vector(7 downto 0)
    );
    end component;
```

```
-- Declarando as constantes e variaveis
signal Clk, M_aux, B1_aux, B2_aux, F1_aux, F2_aux, NT_aux : std_logic := '0';
signal VT_aux : std_logic_vector(7 downto 0);
constant preco_r1 :  std_logic_vector(0 to 7) := "01100100"; -- custo do produto 1 - R$ 1 = 100 centavos
constant preco_r2 :  std_logic_vector(0 to 7) := "11111010"; -- custo do produto 2 - R$ 2,50 = 250 centavos
constant m_valores : std_logic_vector(0 to 15) := "0101010101000001";
constant b1_valores : std_logic_vector(0 to 15) := "1111111111111111";
constant b2_valores : std_logic_vector(0 to 15) := "0000000000000000";

signal valor_moeda : std_logic_vector(7 downto 0);
```

Testbench

```
-- Instanciando a main
int_maquina_de_vendas : maquina_de_vendas
    port map(
    Clk => Clk,
    B1 => B1_aux,
    B2 => B2_aux,
    R1 => preco_r1,
    R2 => preco_r2,
    M => M_aux,
    V => valor_moeda,
    F1 => F1_aux,
    F2 => F2_aux,
    NT => NT_aux,
    VT => VT_aux
    );
```

Testbench

```
-- Processo 01: Gerar as entradas
gerador_entrada: process(Clk)
    variable i: integer := 0;
begin
    if rising_edge(Clk) then
        M_aux <= m_valores(i);
        B1_aux <= b1_valores(i);
        B2_aux <= b2_valores(i);
        i := i + 1;

        if i = m_valores'length then
            i := 0;
        end if;

    end if;
end process;

    -- Processo 02: ler os valores das entradas A e B do arquivo de texto e atribui para os respctivos inputs
lendo_entradas: process (M_aux)
    file F: TEXT open READ_MODE is "C:\Users\Mercedes\QuartusLite\Projetos\PHI_VHDL\projetos\proj3_phi_equipe0\entradas.txt";
    variable L: LINE;
    variable entrada : integer;
begin
    if M_aux = '1' then
        if not endfile(F) then
            READLINE(F, L);
            READ(L, entrada);
            valor_moeda <= std_logic_vector(to_unsigned(entrada, 8));
        else
            valor_moeda <= "00000000";
        end if;
    end if;
end process;

Clk <= not Clk after 5 ns;  -- variando o clock
```

Fim

https://github.com/mercedesDiniz/PHI_VHDL/tree/main/projetos/proj3_phi_equipe0