ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELETRÔNICA
LABORATÓRIO DE SISTEMAS DIGITAIS
PROF. HERMES AGUIAR MAGALHÃES

SISTEMA DE AUTOATENDIMENTO PARA BIBLIOTECAS

Guilherme de Souza Campos
Yan Victor Gomes Ferreira

Belo Horizonte
Junho de 2022

Introdução

No ambiente de uma biblioteca, é fundamental existir uma forma eficiente e segura de realizar o empréstimo e a devolução de livros. Na biblioteca da Escola de Engenharia da UFMG, a forma vigente como isso é feito se baseia em uma recepção na qual o usuário da biblioteca precisa passar para negociar o empréstimo ou devolução do livro com uma bibliotecária. Porém, por ser simples e repetitiva, a tarefa de realização do empréstimo ou devolução de um livro é passível de automatização, dispensando a presença de funcionários na recepção da biblioteca durante todo o seu período de funcionamento e, assim, representando uma possibilidade de redução de custos para a universidade. Diante disso, o presente trabalho foi desenvolvido com o intuito de projetar um sistema digital capaz de automatizar o empréstimo e a devolução de livros.

Como o sistema foi desenvolvido baseado na forma de funcionamento da biblioteca da Escola de Engenharia, um dos pilares do seu funcionamento é a chipagem de todos os livros do acervo da biblioteca, como ocorre no ambiente em questão. A nível de simplificação, o grupo definiu essa chipagem como sendo feita de forma com que cada livro emita um sinal de rádio único, com um **código binário de 16-bits**. Esse código pode ser lido por um sistema de portais antifurto, ilustrados na Fig. 1, equipados com um alarme que dispara se um aluno tentar sair da biblioteca com um livro cujo empréstimo ainda não foi realizado.

Figura 1 - Sistema de Portais Antifurto

Tomando como inspiração a tecnologia dos caixas eletrônicos que, no ambiente bancário, permitem ao cliente realizar as operações que deseja sem o intermédio de um funcionário do banco, o sistema desenvolvido pelo grupo funciona como um “caixa eletrônico para livros”, orientando o usuário da biblioteca na realização do empréstimo ou da devolução de uma obra e controlando o sistema antifurtos para evitar que um livro saia da biblioteca sem antes ser emprestado.

Para a utilização desse “caixa eletrônico”, ainda, cada aluno possui um **cartão que armazena um vetor de 32-bits**, sendo os primeiros 16-bits correspondentes ao seu número de identificação e os últimos 16-bits correspondentes a um código, que pode ser o código do livro que ele deve à biblioteca, no caso de ele estar em débito, ou um vetor de zeros, no caso de ele não estar em débito.

Componentes do sistema

Para realizar as funções desejadas, o sistema conta com um processador e componentes periféricos, cujas conexões estão representadas no diagrama de blocos da Fig. 2:

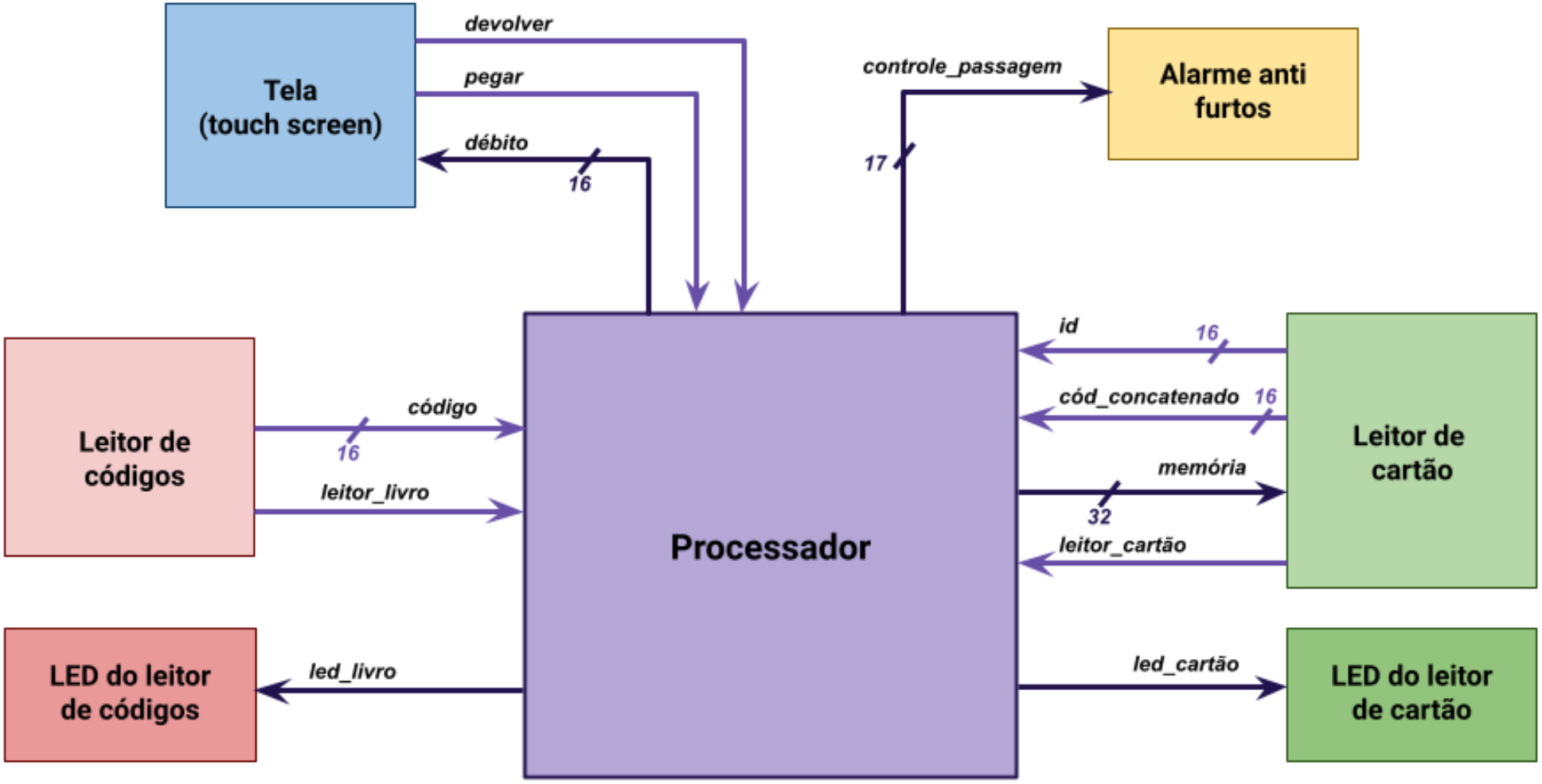


Figura 2 - Diagrama de Blocos do Sistema

Os componentes periféricos do sistema permitem a sua interação com o usuário e com o mundo externo, sendo eles:

- **Leitor de Cartão:**

    - Permite ao sistema ler as informações armazenadas no cartão do aluno - ou seja, o seu número de identificação e o código do livro (ou vetor de zeros) associado a ele - e sobrescrever essas informações quando necessário.
    - Entradas:
        - *memória* (32-bits): valor que é armazenado na memória do cartão após o empréstimo ou devolução de um livro. Se foi feito um empréstimo, é armazenado na memória o número de identificação do aluno (16-bits) concatenado com o código do livro emprestado (16-bits). Se foi feita uma devolução, é armazenado na memória o número de identificação do aluno concatenado com um vetor de zeros (16-bits).
    - Saídas:
        - *id* (16-bits)*:* número de identificação do aluno, correspondendo aos primeiros 16-bits armazenados na memória do cartão lido.
        - *cód_concatenado* (16-bits)*:* código concatenado ao número de identificação do aluno na memória do cartão. Se ele deve algum livro à biblioteca, esse código corresponde ao código do livro. Se não, o código é um vetor de zeros.

- **Leitor de Códigos:**

    - Permite ao sistema ler o código do livro que o aluno quer pegar ou devolver à biblioteca
    - Entradas: -
    - Saídas:
        - *código* (16-bits)*:* código associado exclusivamente ao livro que foi aproximado do leitor

- **Alarme Anti Furtos:**

    - Impede que o aluno saia da biblioteca com um livro cujo empréstimo ainda não foi realizado.
    - Entradas:

        - *controle_passagem* (17-bits)*:* código correspondente à concatenação de 1 bit de controle ao código de um livro. Se o bit concatenado for igual a '1', o alarme permite a passagem do livro. Se o bit for igual a 'o' a passagem é impedida.
    - Saídas: -

- **LEDs de Orientação:**

    - Apontam ao aluno quando inserir o cartão ou aproximar o livro dos leitores da máquina.
    - Entradas:
        - *led_cartão* (1-bit)*:* entrada que controla o estado do LED referente ao leitor de cartão. Se o bit for igual a '1', o LED acende, indicando que o aluno deve inserir o cartão no leitor. Se for igual a 'o', o LED apaga, indicando que o aluno pode retirar o cartão.
        - *led_livro* (1-bit)*:* entrada que controla o estado do LED referente ao leitor de códigos. Se o bit for igual a '1', o LED acende, indicando que o aluno deve aproximar o livro do leitor. Se for igual a 'o', o LED apaga, indicando que o aluno pode afastar o livro do leitor.
    - Saídas: -

- **Tela Touchscreen:**

    - Interface de interação entre o aluno e a máquina.
    - Entradas:
        - *débito* (16-bits)*:* saída correspondente ao código do livro que o aluno deve à biblioteca. O código é impresso na tela.
    - Saídas:
        - *pegar* (1-bit)*:* entrada que, se em nível lógico alto, indica que o aluno escolheu pegar um livro.
        - *devolver* (1-bit)*:* entrada que, se em nível lógico alto, indica que o aluno escolheu devolver um livro.

Implementação do sistema

1. FSM de alto nível

A primeira fase de implementação do sistema foi o desenvolvimento de uma máquina de estados de alto nível que modelasse o seu comportamento. A tabela 1

mostra as entradas, saídas e registradores da máquina. O diagrama da máquina de estados criada pelo grupo está mostrado na Fig. 3.

Entradas	Saídas	Registradores
pegar, devolver, id, cód_concatenado, leitor_cartão, leitor_livro, código	débito, led_cartão, led_livro, memória, controle_passagem	reg_id, reg_débito, reg_cód_concatenado, reg_código, reg_pegar, reg_devolver, reg_memória, reg_controle_passagem

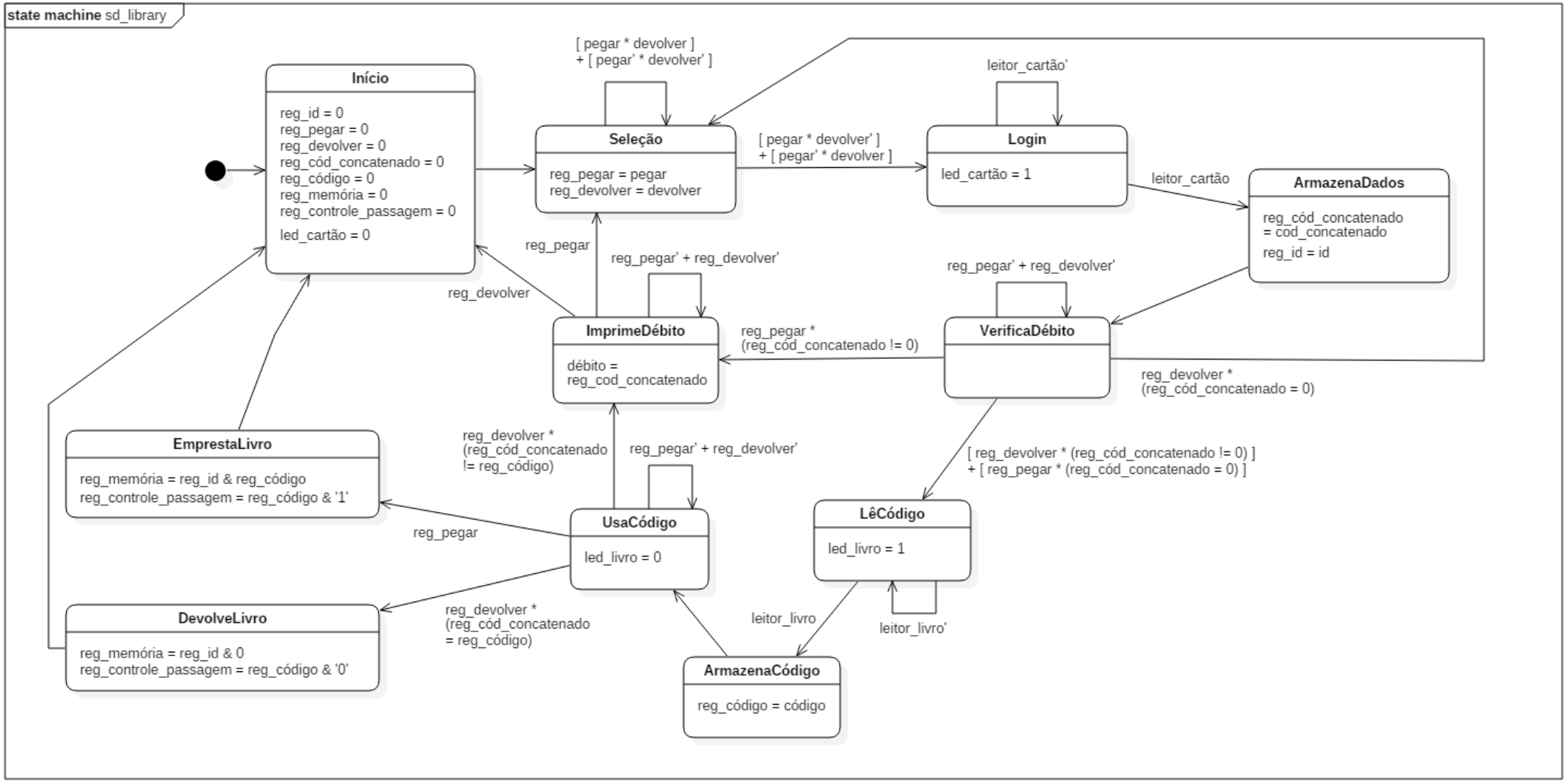


Figura 3 - FSM de Alto Nível

A operação da FSM começa no estado *Início,* que zera todos os registradores do sistema e garante que o LED próximo ao leitor de cartão esteja desligado, (funcionalidade que será explicada nos estados seguintes) e avança para o próximo estado após uma borda de subida do sinal de clock. No segundo estado, *Seleção*, o usuário pode escolher entre duas opções que são mostradas na tela do sistema: pegar um livro (correspondente à entrada de 1 bit *pegar*, que fica em nível lógico alto se o usuário selecionar essa opção na tela touchscreen*)* ou devolver um livro (correspondente à entrada de 1 bit *devolver*, que funciona de forma análoga à anterior*)*. O valor das entradas é armazenado nos registradores *reg_pegar* e *reg_devolver.* A FSM se mantém nesse estado até que o usuário selecione uma das opções e avança para o próximo estado quando uma das duas entradas estiver em nível lógico alto. No estado de *Login*, o LED próximo ao leitor de cartão do sistema acende, indicando que o usuário deve inserir o seu cartão. A máquina permanece nesse estado enquanto a entrada *leitor_cartão*, que indica se há um cartão inserido na máquina, estiver em nível lógico baixo. Quando o usuário insere o cartão, o sistema avança para o estado ArmazenaDados, em que o seu número de identificação, correspondente à entrada *id,* e o código concatenado a ele, correspondente à entrada *cód_concatenado,* são lidos e armazenados nos registradores *reg_id* e *reg_cód_concatenado*, respectivamente, e a FSM avança para o próximo estado.

O estado seguinte da FSM, *VerificaDébito*, permite que os registradores do estado anterior sejam efetivamente atualizados e avança para diferentes estados de acordo com a situação do aluno para com a biblioteca:

A. Se o aluno escolheu pegar um livro (*reg_pegar = 1)* e o código concatenado ao seu número de identificação não é um vetor de zeros (*reg_cód_concatenado != 0)*, o sistema não permite que ele pegue um livro, já que ele apresenta um débito, avança para o estado *ImprimeDébito*, que mostra na tela o código do livro que ele deve devolver à biblioteca antes de pegar outro, e, na próxima borda de subida do clock, volta para o estado de *Seleção*.

B. Se o aluno escolheu pegar um livro (*reg_pegar = 1)* e o código concatenado ao seu número de identificação é um vetor de zeros (*reg_cód_concatenado = 0)*, indicando que ele não apresenta débito, o sistema permite que ele realize o empréstimo e avança para o estado *LêCódigo*. Nesse estado, o LED próximo ao leitor de códigos do sistema acende (*led_livro = 1)*, indicando que o aluno deve aproximar da máquina o livro que deseja pegar para que o seu código seja lido. A FSM se mantém nesse estado até a entrada *leitor_código* ficar em nível lógico alto, indicando que um livro foi aproximado do leitor. Quando *leitor_código = 1,* o sistema avança para o estado ArmazenaCódigo, em que o

código do livro lido é armazenado no registrador *reg_código,* e a máquina avança para o estado seguinte. No estado *UsaCódigo*, o LED próximo ao leitor de códigos é desligado, indicando que o usuário já pode afastar o livro da máquina, e, como *reg_pegar* = 1 nesse caso, a FSM avança para o estado *EmprestaLivro*. Nesse estado, armazena-se na memória do cartão do aluno o seu número de identificação concatenado ao código do livro que ele pegou e envia-se para o alarme antifurtos o código do livro emprestado concatenado a '1', indicando que, caso um livro com esse código passe por ele, o alarme não deve ser acionado. Por fim, volta-se para o estado inicial da FSM e o LED próximo ao leitor de cartão é desligado, indicando que o aluno pode retirar o cartão da máquina.

C. Se o aluno escolheu devolver um livro (*reg_devolver = 1)* e o código concatenado ao seu número de identificação é um vetor de zeros (*reg_cód_concatenado = 0)*, o sistema não permite que ele devolva um livro, já que ele não apresenta débito, e volta para o estado de *Seleção*.

D. Se o aluno escolheu devolver um livro (*reg_devolver = 1)* e o código concatenado ao seu número de identificação não é um vetor de zeros (*reg_cód_concatenado != 0)*, o sistema avança para o estado *LêCódigo* para verificar se o livro que ele quer devolver é o livro que ele deve à biblioteca. Nesse estado, como explicado anteriormente, o LED do leitor de códigos acende e, se um livro for detectado, o sistema avança para o estado ArmazenaCódigo, o seu código é armazenado no registrador *reg_código* e o sistema avança para o estado *UsaCódigo*:

    i. Se o código do livro lido for diferente do código concatenado ao número de identificação do aluno (*reg_código != reg_cód_concatenado)*, significa que o aluno está tentando devolver um livro que não é o que ele deve à biblioteca. Assim, o sistema avança para o estado *ImprimeDébito*, que coloca na tela o código do livro que o aluno precisa devolver, e volta para o estado inicial da FSM.

    ii. Se o código do livro lido for igual ao código concatenado ao número de identifcação do aluno (*reg_código = reg_cód_concatenado)*, significa que o aluno está tentando devolver o livro que ele realmente deve à biblioteca. Assim, o sistema avança para o estado *DevolveLivro*, que armazena na memória do cartão do aluno o seu número de identificação concatenado a um vetor de zeros - quitando seu débito com a

biblioteca - e envia para o alarme antifurtos o código do livro devolvido concatenado a ‘o’, indicando que, caso um livro com esse código passe por ele, o alarme deve ser acionado (evitando que o aluno saia da biblioteca com o livro). Por fim, a FSM volta para o estado inicial.

2. Caminho de dados

Após a modelagem com a FSM de alto nível, foi definido o caminho de dados do processador do sistema, como representado na Fig. 3:

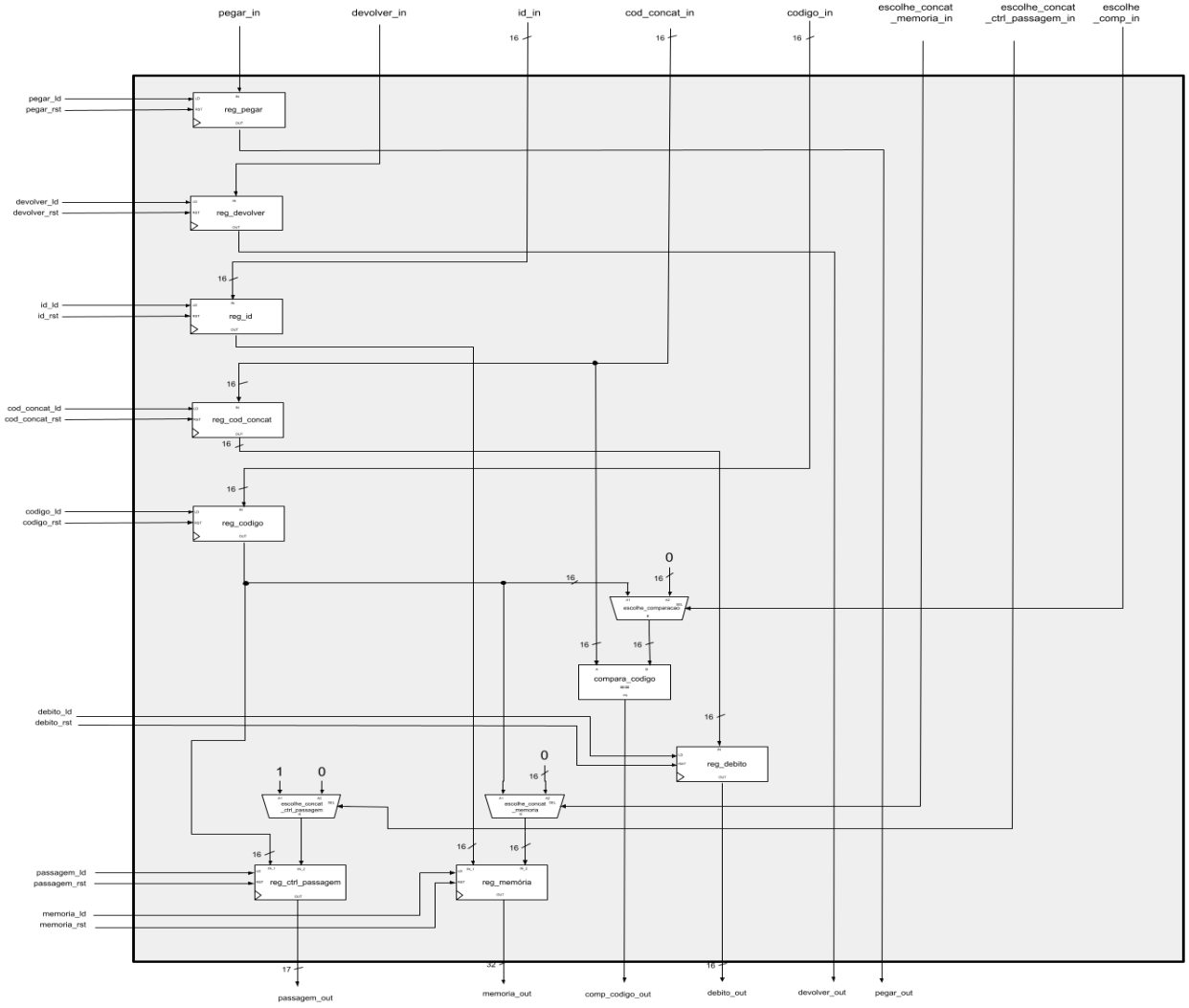


Fig. 3 - Caminho de Dados

Os componentes e conexões do caminho de dados foram definidos de forma a tornar possível que o processador realize todas as operações de alto nível presentes na FSM anterior, como comparações e armazenamento de dados em registradores. Uma breve explicação do funcionamento de cada componente, a sua descrição em VHDL, sua testbench e seu esquemático gerado pelo Quartus II estão apresentados a seguir

- Registradores (*reg_id, reg_pegar, reg_devolver, reg_cod_concat, reg_codigo, reg_memoria, reg_ctrl_passagem , reg_debito)*

  Os registradores utilizados no caminho de dados da máquina são ligados, todos, ao mesmo sinal de clock, e possuem uma entrada de dados, uma entrada de habilitação de escrita e uma entrada de habilitação da limpeza (clear) do registrador. Cada um deles desempenha uma função diferente para o funcionamento do sistema e o tamanho das suas palavras pode variar:

  - *reg_id* (16-bits): armazena o número de identifcação do aluno, que corresponde aos primeiros 16-bits armazenados no seu cartão
  - *reg_cod_concat* (16-bits): armazena o código concatenado ao número de identificação do aluno, correspondendo aos últimos 16-bits armazenados no seu cartão
  - *reg_pegar* (1-bit): armazena '1' se o aluno escolheu pegar um livro e '0' se ele não escolheu pegar um livro
  - *reg_devolver* (1-bit): armazena '1' se o aluno escolheu devolver um livro e '0' se ele não escolheu devolver um livro
  - *reg_codigo* (16-bits): armazena o código do livro lido pelo leitor de códigos do sistema
  - *reg_memoria* (32-bits): armazena o vetor de bits que será escrito na memória do cartão do aluno
  - *reg_ctrl_passagem* (17-bits): armazena vetor de bits que será enviado para o sistema de alarme anti furtos
  - reg_debito (16-bits): armazena código do livro que será mostrado na tela do sistema

  A seguir, temos a representação de um registrador de 16-bits, cujo código e testbench são análogos aos registradores de outros tamanhos.

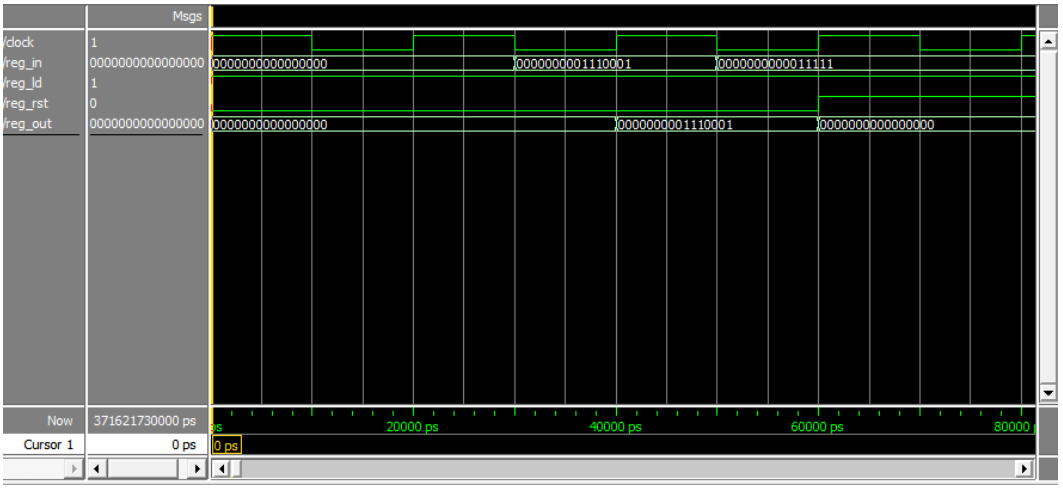
- *Descrição em VHDL*

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity reg_16bits is
  port (
          clock : in STD_LOGIC;

          reg_in  : in STD_LOGIC_VECTOR(15 downto 0);
          reg_ld  : in STD_LOGIC;
          reg_rst : in STD_LOGIC;
          reg_out : out STD_LOGIC_VECTOR(15 downto 0)
      );
end reg_16bits;

architecture rtl of reg_16bits is
begin
  process (reg_in, reg_ld, reg_rst, clock)
  begin
    if (reg_rst = '1') then
      reg_out <= "0000000000000000";
    elsif (rising_edge(CLOCK) and reg_ld = '1') then
      reg_out <= reg_in;
    end if;
   end process;
end rtl;
```

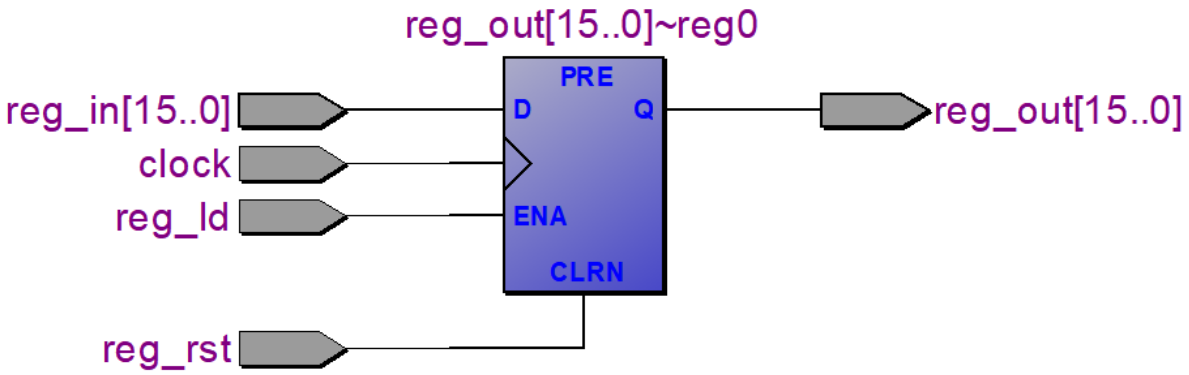
- *Testbench*

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity tb_reg_16bits is
end tb_reg_16bits;

architecture teste of tb_reg_16bits is
  component reg_16bits is
     port (
          clock : in STD_LOGIC;

          reg_in  : in STD_LOGIC_VECTOR (15 downto 0);
          reg_ld  : in STD_LOGIC;
          reg_rst : in STD_LOGIC;
          reg_out : out STD_LOGIC_VECTOR (15 downto 0)
          );
end component;

signal CLOCK : STD_LOGIC := '0';
signal REG_LD, REG_RST : STD_LOGIC;
signal REG_IN : STD_LOGIC_VECTOR(15 downto 0);

constant clk_period : time := 20ns;

begin

  reg_pegar : reg_16bits port map (clock=>CLOCK, reg_in=>REG_IN, reg_ld=>REG_LD, reg_rst=>REG_RST);
  clk_process : process
  begin
```

```
30      loop
31       CLOCK <= '1';
32       wait for clk_period/2;
33       CLOCK <= '0';
34       wait for clk_period/2;
35      end loop;
36    end process;
37
38    REG_IN <= "0000000000000000", "0000000001110001" after 30ns, "0000000000011111" after 50ns;
39    REG_LD <= '1';
40    REG_RST <= '0', '1' after 60ns;
41
42  end teste;
```



- *RTL Viewer*



- Comparador de igualdade de 16-bits (*compara_código)*

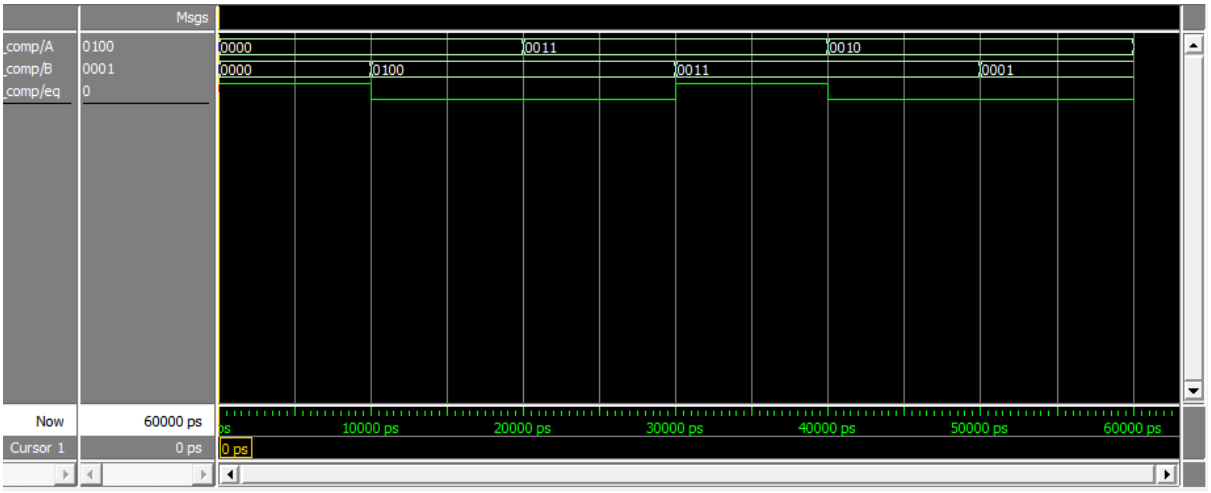
  Compara o código lido pelo leitor de códigos com o código concatenado ao número de identificação do aluno e com um vetor de zeros. A escolha de qual comparação fazer é realizada através de um MUX 2x1, que será descrito no próximo tópico. O comparador retorna '1' se os dados do comparador forem iguais e 'o', se forem diferentes.

  - *Descrição em VHDL*

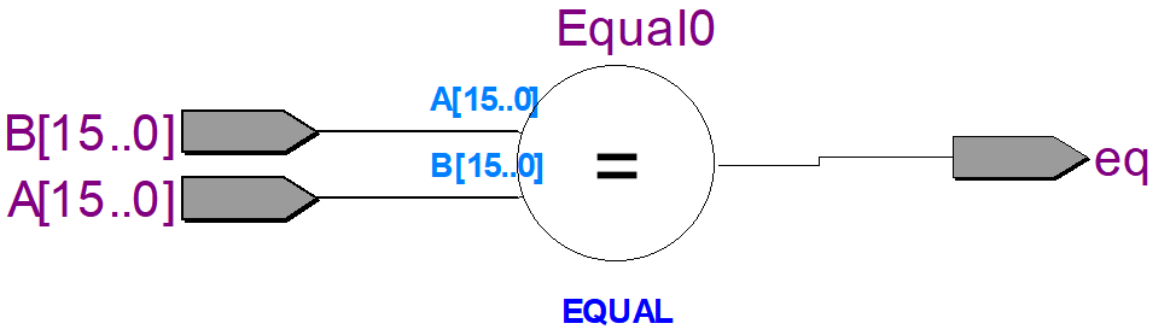
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity comp is
    generic (N:integer := 8);
    port (
            A  : IN STD_LOGIC_VECTOR(N-1 downto 0);
            B  : IN STD_LOGIC_VECTOR(N-1 downto 0);
            eq : OUT STD_LOGIC
          );

end comp;

architecture rtl of comp is
begin
     with A = B select
     eq <= '1' when true,
           '0' when others;
end rtl;
```

- *Testbench*

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity tb_comp is
end tb_comp;

architecture teste of tb_comp is

component comp is
    generic (N:INTEGER := 4);
    port (
            A  : IN STD_LOGIC_VECTOR(N-1 downto 0);
            B  : IN STD_LOGIC_VECTOR(N-1 downto 0);
            eq : OUT STD_LOGIC
         );
end component;

signal x, y : STD_LOGIC_VECTOR(3 downto 0);
signal s_eq : STD_LOGIC;

begin
   instancia_comp : comp  generic map (N => 4) port map(A=>x, B=>y, eq=>s_eq);

   x <= x"0", x"3" after 20 ns, x"2" after 40 ns, x"4" after 60 ns;
   y <= x"0", x"4" after 10 ns, x"3" after 30 ns, x"1" after 50 ns;
end teste;

```



- *RTL Viewer*



- Multiplexadores 2x1 (*escolhe_comparação, escolhe_concatenação_memória, escolhe_concatenação_controle_passagem)*

  - *escolhe_comparação:* coloca em uma das entradas do comparador *compara_código* um vetor de zeros ou o código concatenado ao número de identificação do aluno, de acordo com o nível lógico da sua entrada de seleção. Se a entrada de seleção está em nível lógico alto, o MUX coloca o código na entrada do comparador. Caso contrário, o MUX coloca o vetor de zeros na entrada do comparador.
  - *escolhe_concatenação_memória:* coloca em uma das entradas do registrador *reg_memória* um vetor de zeros ou o código do livro lido pelo leitor de códigos. Se a entrada de seleção está em nível lógico alto, o MUX coloca o código na entrada do registrador. Caso contrário, o MUX coloca o vetor de zeros na entrada do registrador.
  - *escolhe_concatenação_controle_passagem:* coloca em uma das entradas do registrador *reg_controle_passagem* '0' ou '1'. Se a entrada de seleção está em nível lógico alto, o MUX coloca '1' na entrada do

registrador. Caso contrário, o MUX coloca o ‘0’ na entrada do registrador.

- *Descrição em VHDL*

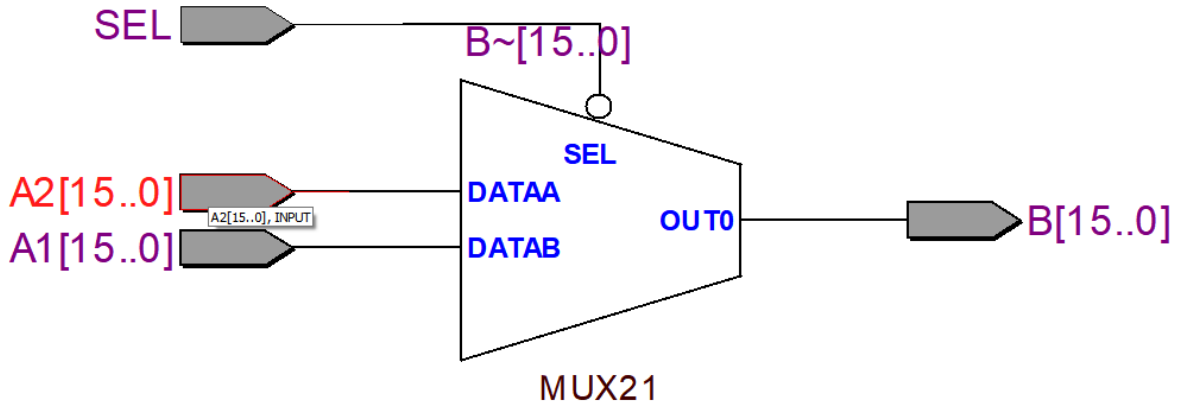
A seguir, temos a representação de um MUX para palavras de 16 bits, cujo código e testbench são análogos ao mux de 1 bit.

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all;
3
4   entity mux2 is
5       generic (N:integer := 16);
6       port(
7               A1      : in  STD_LOGIC_VECTOR(N-1 downto 0);
8               A2      : in  STD_LOGIC_VECTOR(N-1 downto 0);
9               SEL     : in  STD_LOGIC;
10              B       : out STD_LOGIC_VECTOR(N-1 downto 0)
11          );
12  end mux2;
13
14  architecture rtl of mux2 is
15  begin
16      p_mux : process(A1,A2, SEL)
17      begin
18        case SEL is
19          when '0' => B <= A1 ;
20          when '1' => B <= A2 ;
21           when others => B <= A2;
22        end case;
23      end process p_mux;
24  end rtl;
```

- *Testbench*

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all;
3
4   entity tb_mux2 is
5   end tb_mux2;
6
7   architecture teste of tb_mux2 is
8
9   component mux2 is
10     generic (N:integer :=15);
11     port(
12             A1      : in  STD_LOGIC_VECTOR(N-1 downto 0);
13             A2      : in  STD_LOGIC_VECTOR(N-1 downto 0);
14             SEL     : in  STD_LOGIC;
15             B       : out STD_LOGIC_VECTOR(N-1 downto 0)
16         );
17  end component;
18
19  signal s_a1, s_a2, s_b : STD_LOGIC_VECTOR(15 downto 0);
20  signal s_sel           : STD_LOGIC;
21
22  begin
23     instancia_mux2 : mux2 generic map (N=>15) port map(A1=>s_a1, A2=>s_a2, B=>s_b, SEL=>s_sel);
24
25     s_a1 <=  "0000000000011111", "1110000001100000" after 20 ns,
26                                  "0010101010100100" after 40 ns, "0111110000000011" after 60 ns;
27     s_a2 <=  "1001010000001111", "0100111000001111" after 20 ns,
28                                  "0000101100110011" after 40 ns, "1111000010000011" after 60 ns;
29     s_sel <= '1', '0' after 10 ns, '1' after 15 ns, '0' after 25 ns, '1' after 30 ns, '0' after 45 ns,
30                 '1' after 50 ns, '0' after 55 ns, '1' after 65 ns;
31  end teste;
```

- *RTL Viewer*



3. Controladora

Definido o caminho de dados, é necessário desenvolver uma controladora para finalizar o projeto do processador do sistema. A controladora é responsável por tratar da lógica combinacional que faz o sistema se comportar segundo a máquina de estados descrita anteriormente, passando de um estado para outro de acordo com as suas entradas e controlando para o caminho de dados de dados de acordo com o estado em que a máquina se encontra.

A primeira parte do desenvolvimento da controladora consiste em transformar a FSM de alto nível, abordada anteriormente, em uma FSM de baixo nível. A FSM de baixo nível desenvolvida pelo grupo está apresentada na Fig. 4.

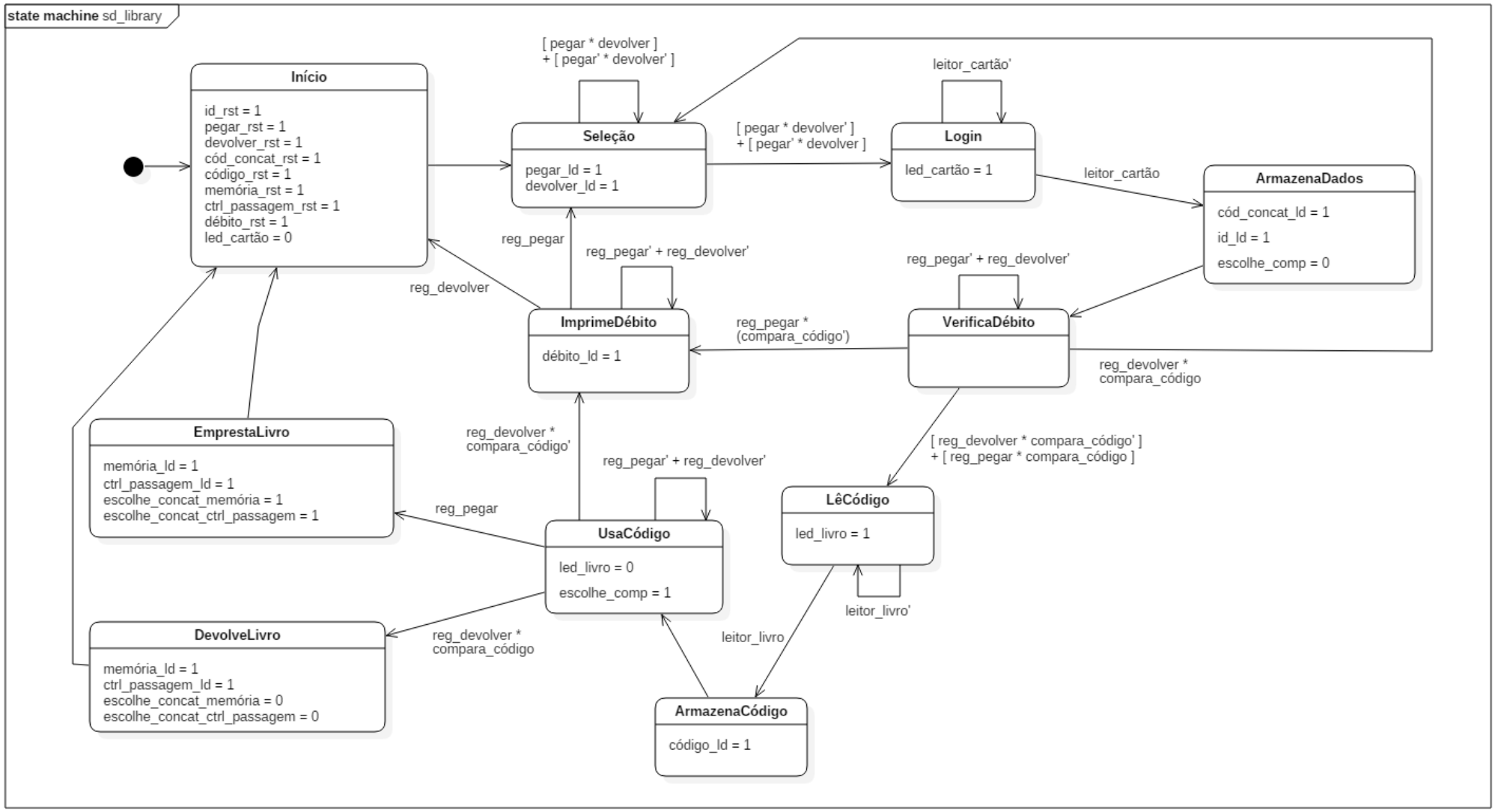


Figura 4 - FSM de Baixo Nível

O diagrama de blocos do processador, mostrando a interligação entre a controladora, o caminho de dados e o mundo externo, está apresentado na Fig. 5. Em seguida, na Fig. 6, é apresentada a arquitetura interna da controladora, com o seu bloco de controle e registrador de estados apresentados.

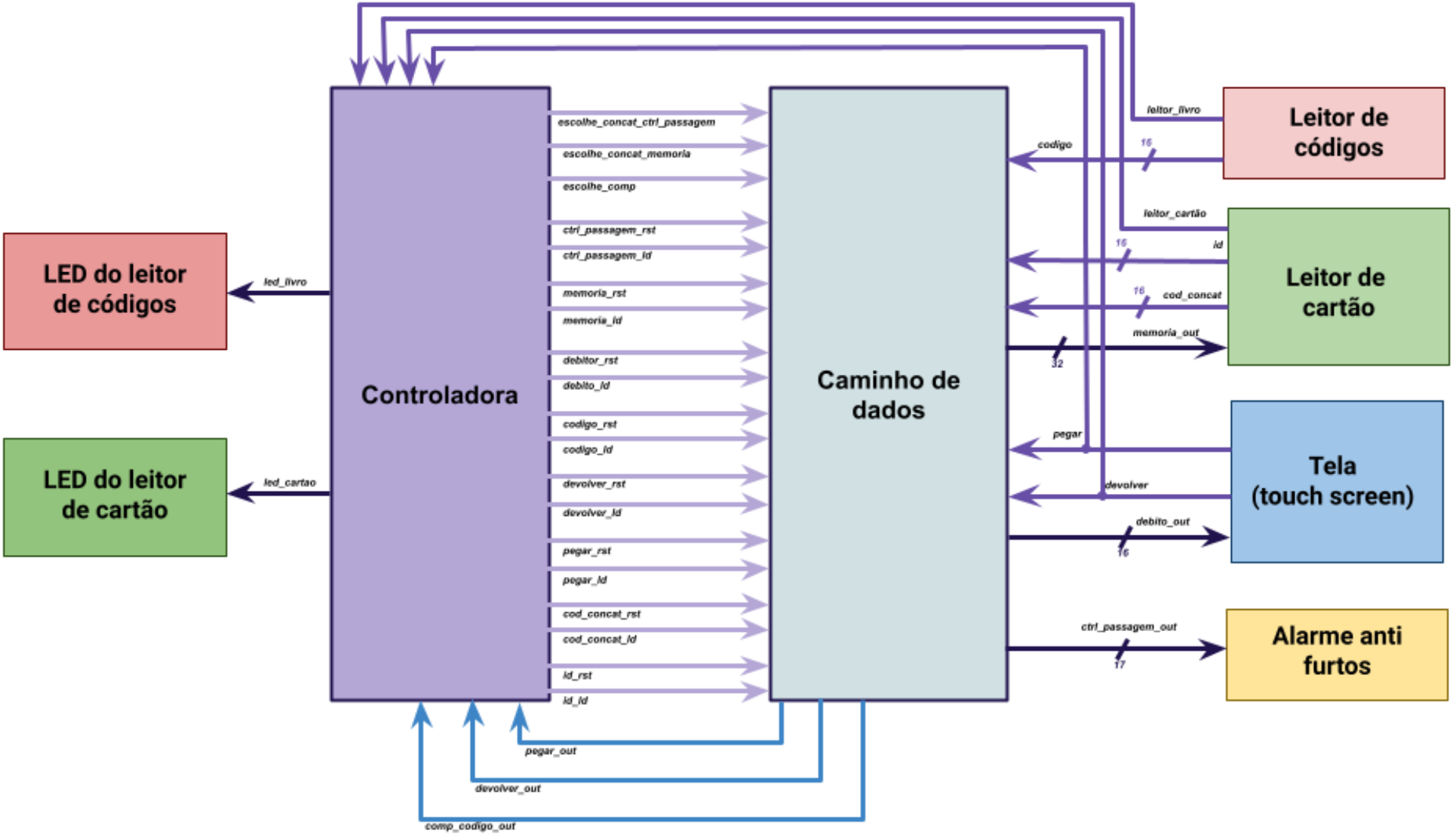


Figura 5 - Diagrama de Blocos do Processador

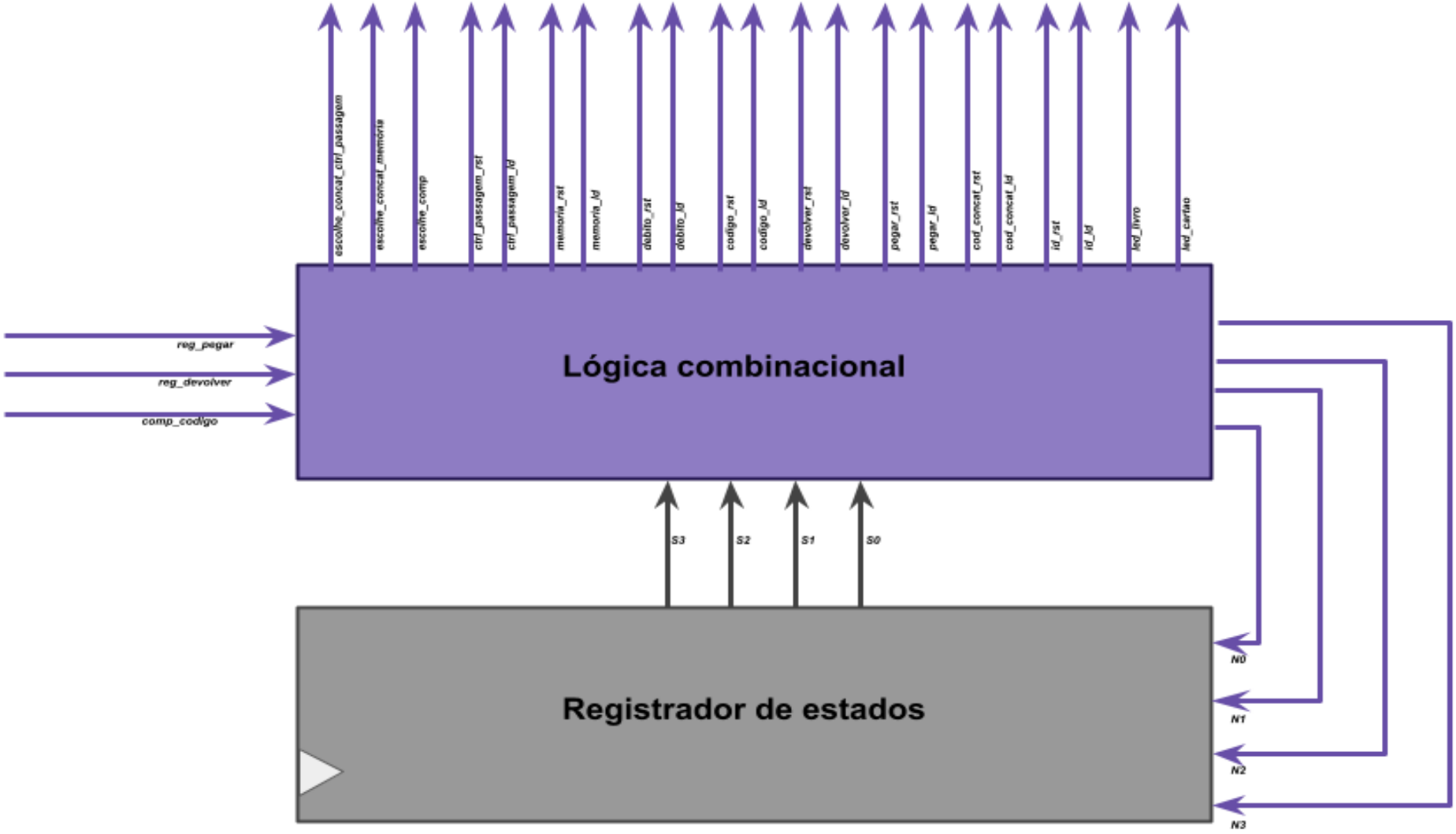


Figura 6 – Diagrama de Blocos da Controladora

4. Implementação do sistema em VHDL

Dando continuidade ao projeto do processador, o grupo realizou a sua descrição em linguagem VHDL de forma a efetivamente implementar a FSM que o modela.

- Descrição do caminho de dados

  Inicialmente, foi descrito o caminho de dados do processador a partir da instanciação e interligação dos componentes apresentados anteriormente neste documento. O resultado da descrição em VHDL do caminho de dados está apresentado a seguir:

  - Código em VHDL

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.all;
3
4  entity datapath is
5    port (
6            -- clock input
7            clock : in STD_LOGIC;
8
9            -- users inputs
10           pegar_in                      : in STD_LOGIC;
11           devolver_in                   : in STD_LOGIC;
12           id_in                         : in STD_LOGIC_VECTOR(15 downto 0);
13           cod_concat_in                 : in STD_LOGIC_VECTOR(15 downto 0);
14           codigo_in                     : in STD_LOGIC_VECTOR(15 downto 0);
15           escolhe_concat_memoria_in     : in STD_LOGIC;
16           escolhe_concat_ctrl_passagem_in : in STD_LOGIC;
17           escolhe_comp_in               : in STD_LOGIC;
18
19           -- registers
20           pegar_ld          : in STD_LOGIC;
21           pegar_rst         : in STD_LOGIC;
22           devolver_ld       : in STD_LOGIC;
23           devolver_rst      : in STD_LOGIC;
24           id_ld             : in STD_LOGIC;
25           id_rst            : in STD_LOGIC;
26           cod_concat_ld     : in STD_LOGIC;
27           cod_concat_rst    : in STD_LOGIC;
28           codigo_ld         : in STD_LOGIC;
29           codigo_rst        : in STD_LOGIC;
30           debito_ld         : in STD_LOGIC;
31           debito_rst        : in STD_LOGIC;
32           passagem_ld       : in STD_LOGIC;
33           passagem_rst      : in STD_LOGIC;
```

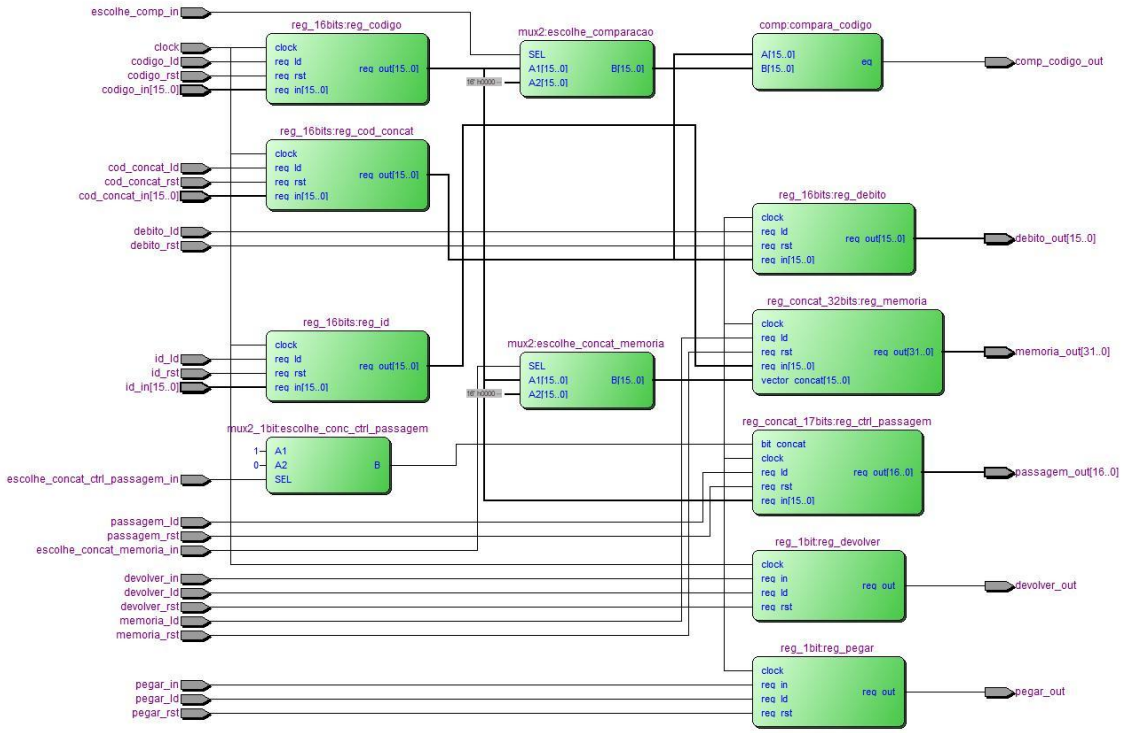
```
34
35              memoria_ld        : in STD_LOGIC;
36              memoria_rst       : in STD_LOGIC;
37
38              -- outputs
39              passagem_out      : out STD_LOGIC_VECTOR(16 downto 0);
40              memoria_out       : out STD_LOGIC_VECTOR(31 downto 0);
41              comp_codigo_out   : out STD_LOGIC;
42              debito_out        : out STD_LOGIC_VECTOR(15 downto 0);
43              devolver_out      : out STD_LOGIC;
44              pegar_out         : out STD_LOGIC
45          );
46  end entity;
47
48  architecture rtl of datapath is
49    -- registers
50    component reg_1bit is
51      port (
52            clock : in STD_LOGIC;
53
54            reg_in  : in STD_LOGIC;
55            reg_ld  : in STD_LOGIC;
56            reg_rst : in STD_LOGIC;
57            reg_out : out STD_LOGIC
58          );
59    end component;
60
61    component reg_16bits is
62      port (
63            clock : in STD_LOGIC;
64
65            reg_in  : in STD_LOGIC_VECTOR(15 downto 0);
66            reg_ld  : in STD_LOGIC;
```

```
67          reg_rst : in STD_LOGIC;
68          reg_out : out STD_LOGIC_VECTOR(15 downto 0)
69        );
70  end component;
71
72  component reg_concat_17bits is
73  port (
74          clock : in STD_LOGIC;
75
76          reg_in        : in STD_LOGIC_VECTOR(15 downto 0);
77          bit_concat    : in STD_LOGIC;
78          reg_ld        : in STD_LOGIC;
79          reg_rst       : in STD_LOGIC;
80          reg_out       : out STD_LOGIC_VECTOR(16 downto 0)
81      );
82  end component;
83
84  component reg_concat_32bits is
85  port (
86          clock : in STD_LOGIC;
87
88          reg_in        : in STD_LOGIC_VECTOR(15 downto 0);
89          vector_concat : in STD_LOGIC_VECTOR(15 downto 0);
90          reg_ld        : in STD_LOGIC;
91          reg_rst       : in STD_LOGIC;
92          reg_out       : out STD_LOGIC_VECTOR(31 downto 0)
93      );
94  end component;
95
96  -- muxes
97  component mux2_1bit is
98    port(
99          A1      : in  STD LOGIC;
```

```
100         A2      : in  STD_LOGIC;
101         SEL     : in  STD_LOGIC;
102         B       : out STD_LOGIC
103       );
104 end component;
105
106 component mux2 is
107   generic (N:integer := 16);
108   port(
109         A1      : in  STD_LOGIC_VECTOR(N-1 downto 0);
110         A2      : in  STD_LOGIC_VECTOR(N-1 downto 0);
111         SEL     : in  STD_LOGIC;
112         B       : out STD_LOGIC_VECTOR(N-1 downto 0)
113       );
114 end component;
115
116 component comp is
117  generic (N:integer := 8);
118  port (
119          A  : IN STD_LOGIC_VECTOR(N-1 downto 0);
120          B  : IN STD_LOGIC_VECTOR(N-1 downto 0);
121          eq : OUT STD_LOGIC
122        );
123
124 end component;
125
126 signal reg_id_out, reg_cod_concat_out, reg_codigo_out, mux_memoria_out, mux_comp_out : STD_LOGIC_VECTOR(15 downto 0);
127 signal mux_passagem_out : STD_LOGIC;
128
129 begin
130
131 reg_pegar : reg_1bit port map(clock, pegar_in, pegar_ld, pegar_rst, pegar_out);
132 reg_devolver : reg_1bit port map(clock, devolver_in, devolver_ld, devolver_rst, devolver_out);
```

```
133 reg_id : reg_16bits port map(clock, id_in, id_ld, id_rst, reg_id_out);
134 reg_cod_concat : reg_16bits port map(clock, cod_concat_in, cod_concat_ld, cod_concat_rst, reg_cod_concat_out);
135 reg_codigo : reg_16bits port map(clock, codigo_in, codigo_ld, codigo_rst, reg_codigo_out);
136 reg_ctrl_passagem : reg_concat_17bits port map(clock, reg_codigo_out, mux_passagem_out, passagem_ld, passagem_rst, passagem_out);
137 reg_memoria : reg_concat_32bits port map(clock, reg_id_out, mux_memoria_out, memoria_ld, memoria_rst, memoria_out);
138 reg_debito : reg_16bits port map(clock, reg_cod_concat_out, debito_ld, debito_rst, debito_out);
139
140 escolhe_conc_ctrl_passagem : mux2_1bit port map('1', '0', escolhe_concat_ctrl_passagem_in, mux_passagem_out);
141 escolhe_concat_memoria: mux2 generic map (N => 16) port map(reg_codigo_out, "0000000000000000", escolhe_concat_memoria_in, mux_memoria_out)
142 escolhe_comparacao : mux2 generic map (N => 16) port map(reg_codigo_out, "0000000000000000", escolhe_comp_in, mux_comp_out);
143
144 compara_codigo : comp generic map (N => 16) port map (reg_cod_concat_out, mux_comp_out, comp_codigo_out);
145
146
147 end rtl;
```

- RTL Viewer


escolhe_comp_in
reg_16bits:reg_codigo
clock
codigo_ld
codigo_rst
codigo_in[15..0]
req_out[15..0]
mux2:escolhe_comparacao
SEL
A1[15..0]
A2[15..0]
B[15..0]
comp:compara_codigo
A[15..0]
B[15..0]
eq
comp_codigo_out
reg_16bits:reg_cod_concat
cod_concat_ld
cod_concat_rst
cod_concat_in[15..0]
reg_16bits:reg_debito
debito_ld
debito_rst
debito_out[15..0]
reg_16bits:reg_id
id_ld
id_rst
id_in[15..0]
mux2:escolhe_concat_memoria
reg_concat_32bits:reg_memoria
vector_concat[15..0]
memoria_out[31..0]
mux2_1bit:escolhe_conc_ctrl_passagem
escolhe_concat_ctrl_passagem_in
reg_concat_17bits:reg_ctrl_passagem
bit_concat
passagem_out[16..0]
passagem_ld
passagem_rst
escolhe_concat_memoria_in
reg_1bit:reg_devolver
devolver_in
devolver_ld
devolver_rst
memoria_ld
memoria_rst
devolver_out
reg_1bit:reg_pegar
pegar_in
pegar_ld
pegar_rst
pegar_out

- Descrição da controladora

  Em seguida, a controladora foi descrita a partir da FSM de baixo nível apresentada anteriormente. O resultado da sua descrição em VHDL está apresentado a seguir:

  - Código em VHDL

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.all;
3
4  entity controladora is
5  port (
6         clk : in STD_LOGIC;
7
8         pegar           : in STD_LOGIC;
9         devolver        : in STD_LOGIC;
10        leitor_cartao   : in STD_LOGIC;
11        leitor_livro    : in STD_LOGIC;
12        reg_pegar       : in STD_LOGIC;
13        reg_devolver    : in STD_LOGIC;
14        compara_codigo  : in STD_LOGIC;
15
16        id_ld              : out STD_LOGIC;
17        id_rst             : out STD_LOGIC;
18        debito_ld          : out STD_LOGIC;
19        debito_rst         : out STD_LOGIC;
20        pegar_ld           : out STD_LOGIC;
21        pegar_rst          : out STD_LOGIC;
22        devolver_ld        : out STD_LOGIC;
23        devolver_rst       : out STD_LOGIC;
24        cod_concat_ld      : out STD_LOGIC;
25        cod_concat_rst     : out STD_LOGIC;
26        codigo_ld          : out STD_LOGIC;
27        codigo_rst         : out STD_LOGIC;
28        memoria_ld         : out STD_LOGIC;
29        memoria_rst        : out STD_LOGIC;
30        ctrl_passagem_ld   : out STD_LOGIC;
31        ctrl_passagem_rst  : out STD_LOGIC;
32
33        escolhe_comp                 : out STD_LOGIC;
34        escolhe_concat_memoria       : out STD_LOGIC;
35        escolhe_concat_ctrl_passagem : out STD_LOGIC;
```

```
36
37          led_cartao : out STD_LOGIC;
38          led_livro  : out STD_LOGIC
39
40      );
41  end controladora;
42
43  architecture rtl of controladora is
44
45      type estado is (INICIO, SELECAO, LOGIN, ARMAZENA_DADOS,
46                      VERIFICA_DEBITO, LE_CODIGO, ARMAZENA_CODIGO,
47                      USA_CODIGO, IMPRIME_DEBITO, DEVOLVE_LIVRO, EMPRESTA_LIVRO);
48
49      signal estado_atual    : estado := INICIO;
50      signal proximo_estado  : estado;
51
52
53  begin
54
55      process(clk) is
56      begin
57          if (rising_edge(clk)) then
58              estado_atual <= proximo_estado;
59          end if;
60      end process;
61
62      process(estado_atual, pegar, devolver, leitor_cartao,
63              leitor_livro, reg_pegar, reg_devolver, compara_codigo) is
64      begin
65
66          id_rst                      <= '0';
67          debito_rst                  <= '0';
68          pegar_rst                   <= '0';
69          devolver_rst                <= '0';
70          cod_concat_rst              <= '0';
```

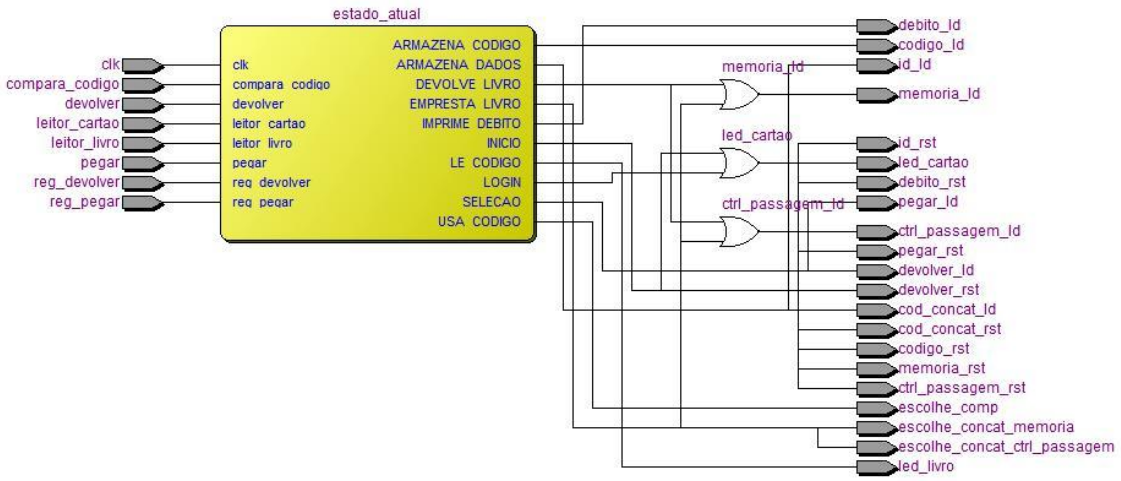
```
71          codigo_rst                  <= '0';
72          memoria_rst                 <= '0';
73          ctrl_passagem_rst           <= '0';
74          id_ld                       <= '0';
75          debito_ld                   <= '0';
76          pegar_ld                    <= '0';
77          devolver_ld                 <= '0';
78          cod_concat_ld               <= '0';
79          codigo_ld                   <= '0';
80          memoria_ld                  <= '0';
81          ctrl_passagem_ld            <= '0';
82          led_cartao                  <= '0';
83          led_livro                   <= '0';
84          escolhe_comp                <= '0';
85          escolhe_concat_memoria      <= '0';
86          escolhe_concat_ctrl_passagem <= '0';
87
88          case estado_atual is
89
90              when INICIO =>
91                  memoria_ld <= '0';
92                  ctrl_passagem_ld <= '0';
93                  id_rst <= '1';
94                  debito_rst <= '1';
95                  pegar_rst <= '1';
96                  devolver_rst <= '1';
97                  cod_concat_rst <= '1';
98                  codigo_rst <= '1';
99                  memoria_rst <= '1';
100                 ctrl_passagem_rst <= '1';
101                 led_cartao <= '1';
102
103                 proximo_estado <= SELECAO;
104
105             when SELECAO =>
```

```
106                 id_rst <= '0';
107                 debito_rst <= '0';
108                 pegar_rst <= '0';
109                 devolver_rst <= '0';
110                 cod_concat_rst <= '0';
111                 codigo_rst <= '0';
112                 memoria_rst <= '0';
113                 ctrl_passagem_rst <= '0';
114                 pegar_ld <= '1';
115                 devolver_ld <= '1';
116
117                 if (((pegar = '1') and (devolver = '0')) or
118                     ((pegar = '0') and (devolver = '1'))) then
119                     proximo_estado <= LOGIN;
120                 else
121                     proximo_estado <= SELECAO;
122                 end if;
123
124             when LOGIN =>
125                 pegar_ld <= '0';
126                 devolver_ld <= '0';
127                 led_cartao <= '1';
128
129                 if (leitor_cartao = '1') then
130                     proximo_estado <= ARMAZENA_DADOS;
131                 else
132                     proximo_estado <= LOGIN;
133                 end if;
134
135             when ARMAZENA_DADOS =>
136                 cod_concat_ld <= '1';
137                 id_ld <= '1';
138                 escolhe_comp <= '0';
139
140                 proximo_estado <= VERIFICA_DEBITO;
```

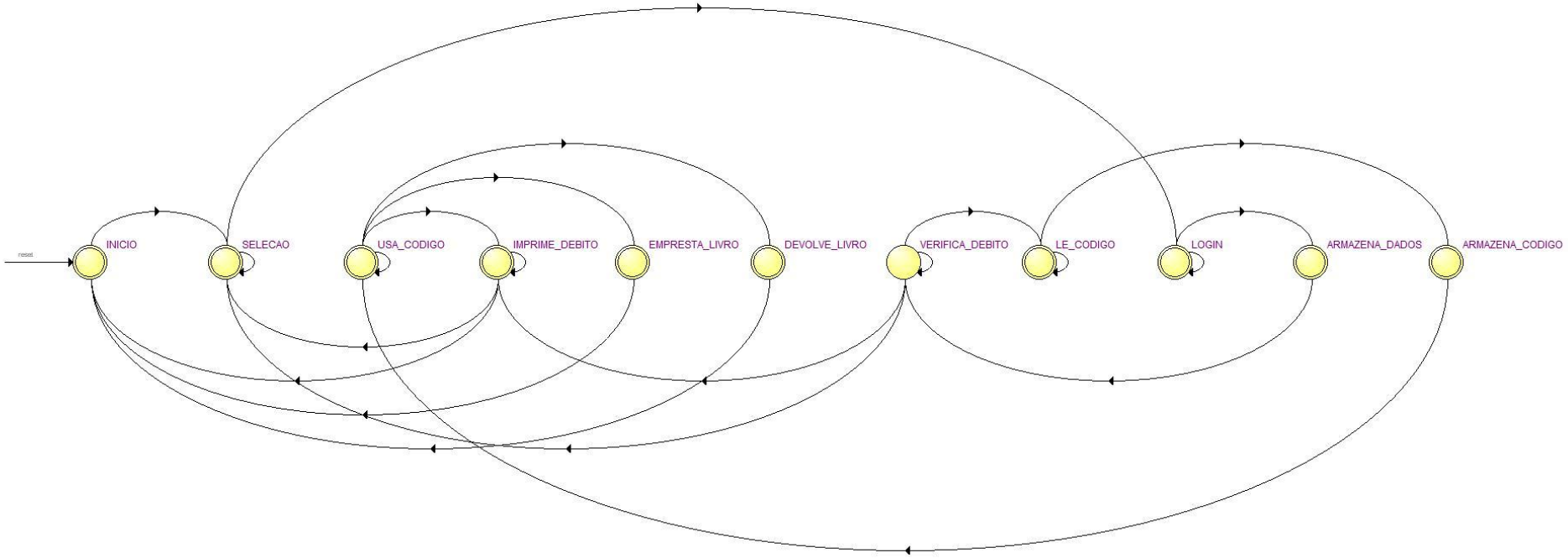
```
141
142         when VERIFICA_DEBITO =>
143             cod_concat_ld <= '0';
144             id_ld <= '0';
145
146             if ((reg_devolver = '1') and (compara_codigo = '1')) then
147                 proximo_estado <= SELECAO;
148             elsif ((reg_pegar = '1') and (compara_codigo = '0')) then
149                 proximo_estado <= IMPRIME_DEBITO;
150             elsif (((reg_devolver = '1') and (compara_codigo = '0')) or
151                    ((reg_pegar = '1') and (compara_codigo = '1'))) then
152                 proximo_estado <= LE_CODIGO;
153             else
154                 proximo_estado <= VERIFICA_DEBITO;
155             end if;
156
157         when LE_CODIGO =>
158             led_livro <= '1';
159
160             if (leitor_livro = '1') then
161                 proximo_estado <= ARMAZENA_CODIGO;
162             else
163                 proximo_estado <= LE_CODIGO;
164             end if;
165
166         when ARMAZENA_CODIGO =>
167             codigo_ld <= '1';
168
169             proximo_estado <= USA_CODIGO;
170
171         when USA_CODIGO =>
172             codigo_ld <= '0';
173             led_livro <= '0';
174             escolhe_comp <= '1';
175
```

```
176                 if (reg_pegar = '1') then
177                     proximo_estado <= EMPRESTA_LIVRO;
178                 elsif ((reg_devolver = '1') and (compara_codigo = '1')) then
179                     proximo_estado <= DEVOLVE_LIVRO;
180                 elsif ((reg_devolver = '1') and (compara_codigo = '0')) then
181                     proximo_estado <= IMPRIME_DEBITO;
182                 else
183                     proximo_estado <= USA_CODIGO;
184                 end if;
185
186             when IMPRIME_DEBITO =>
187                 debito_ld <= '1';
188
189                 if (reg_pegar = '1') then
190                     proximo_estado <= SELECAO;
191                 elsif (reg_devolver = '1') then
192                     proximo_estado <= INICIO;
193                 else
194                     proximo_estado <= IMPRIME_DEBITO;
195                 end if;
196
197             when EMPRESTA_LIVRO =>
198                 memoria_ld <= '1';
199                 ctrl_passagem_ld <= '1';
200                 escolhe_concat_memoria <= '1';
201                 escolhe_concat_ctrl_passagem <= '1';
202
203                 proximo_estado <= INICIO;
204
205             when others =>
206                 memoria_ld <= '1';
207                 ctrl_passagem_ld <= '1';
208                 escolhe_concat_memoria <= '0';
209                 escolhe_concat_ctrl_passagem <= '0';
210
211                 proximo_estado <= INICIO;
212         end case;
213     end process;
214 end rtl;
```

- RTL Viewer



- Diagrama da Máquina de Estados



- Tabela de transição de estados

	Name	EMPRESTA_LIVRO	DEVOLVE_LIVRO	IMPRIME_DEBITO	USA_CODIGO	ARMAZENA_CODIGO	LE_CODIGO	VERIFICA_DEBITO	ARMAZENA_DADOS	LOGIN	SELECAO	INICIO
1	INICIO	0	0	0	0	0	0	0	0	0	0	0
2	SELECAO	0	0	0	0	0	0	0	0	0	1	1
3	LOGIN	0	0	0	0	0	0	0	0	1	0	1
4	ARMAZENA_DADOS	0	0	0	0	0	0	0	1	0	0	1
5	VERIFICA_DEBITO	0	0	0	0	0	0	1	0	0	0	1
6	LE_CODIGO	0	0	0	0	0	1	0	0	0	0	1
7	ARMAZENA_CODIGO	0	0	0	0	1	0	0	0	0	0	1
8	USA_CODIGO	0	0	0	1	0	0	0	0	0	0	1
9	IMPRIME_DEBITO	0	0	1	0	0	0	0	0	0	0	1
10	DEVOLVE_LIVRO	0	1	0	0	0	0	0	0	0	0	1
11	EMPRESTA_LIVRO	1	0	0	0	0	0	0	0	0	0	1

- Descrição do sistema completo

  Por fim, foi feita a descrição do sistema como um todo, instanciando-se e interligando-se o datapath e a controladora. A descrição do sistema em VHDL está mostrada a seguir:

  - Código em VHDL

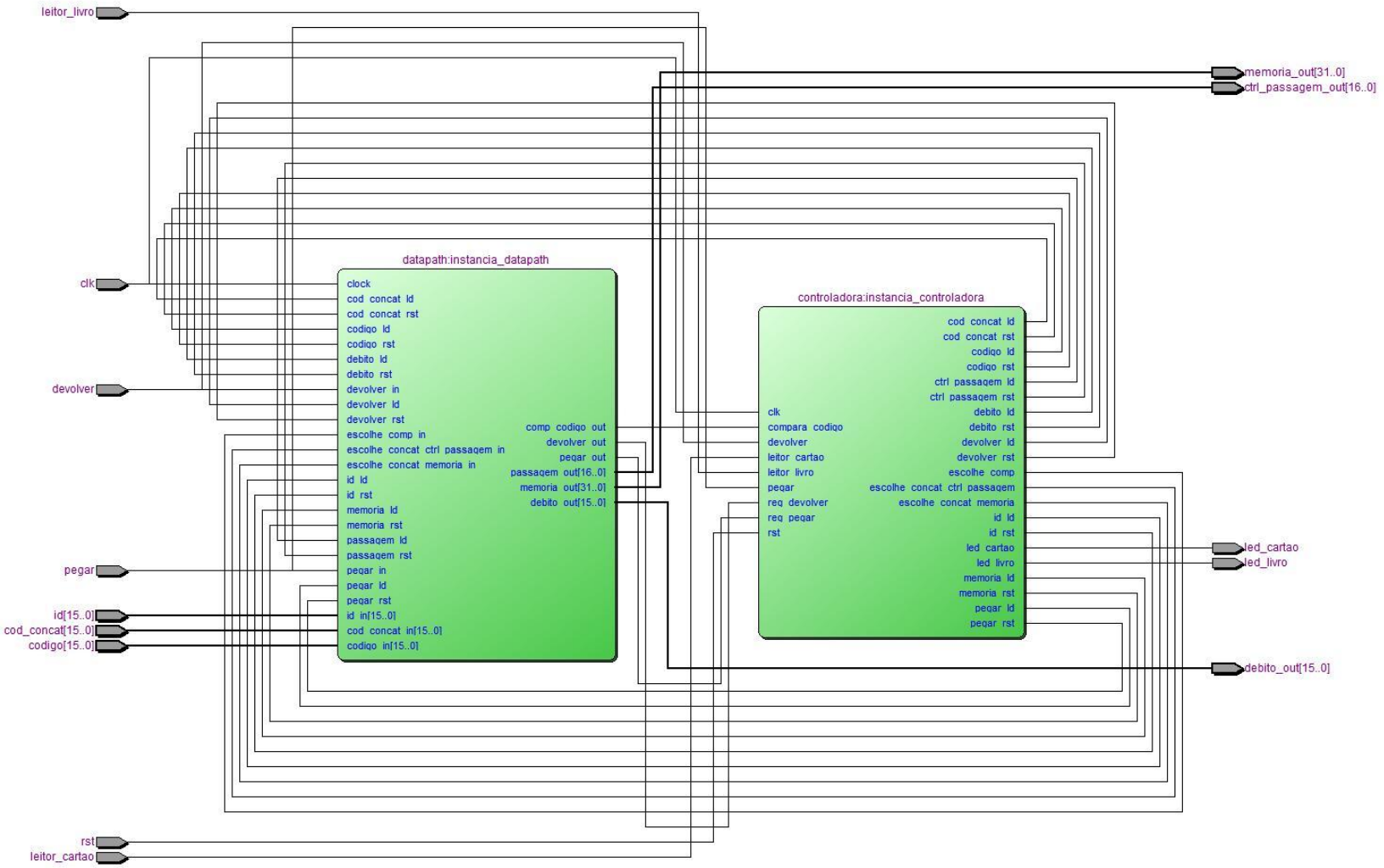
```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all;
3
4   entity sd_library is
5       port (
6       clk : in STD_LOGIC;
7
8       leitor_cartao : in STD_LOGIC;
9       leitor_livro  : in STD_LOGIC;
10      pegar         : in STD_LOGIC;
11      devolver      : in STD_LOGIC;
12      id            : in STD_LOGIC_VECTOR(15 downto 0);
13      cod_concat    : in STD_LOGIC_VECTOR(15 downto 0);
14      codigo        : in STD_LOGIC_VECTOR(15 downto 0);
15
16      led_livro         : out STD_LOGIC;
17      led_cartao        : out STD_LOGIC;
18      ctrl_passagem_out : out STD_LOGIC_VECTOR(16 downto 0);
19      memoria_out       : out STD_LOGIC_VECTOR(31 downto 0);
20      debito_out        : out STD_LOGIC_VECTOR(15 downto 0)
21      );
22  end entity;
23
24  architecture rtl of sd_library is
25
26      signal reg_pegar, reg_devolver, compara_codigo : STD_LOGIC;
27
28      signal pegar_ld, pegar_rst, devolver_ld, devolver_rst, cod_contat_ld, cod_concat_rst,
29             id_ld, id_rst, debito_ld, debito_rst, codigo_ld, codigo_rst,
30             memoria_ld, memoria_rst, ctrl_passagem_ld, ctrl_passagem_rst,
31             escolhe_comp, escolhe_concat_memoria, escolhe_concat_ctrl_passagem : STD_LOGIC;
32
33      component controladora is
34      port (
35         clk : in STD_LOGIC;
```

```
36
37          pegar          : in STD_LOGIC;
38          devolver       : in STD_LOGIC;
39          leitor_cartao  : in STD_LOGIC;
40          leitor_livro   : in STD_LOGIC;
41          reg_pegar      : in STD_LOGIC;
42          reg_devolver   : in STD_LOGIC;
43          compara_codigo : in STD_LOGIC;
44
45          id_ld             : out STD_LOGIC;
46          id_rst            : out STD_LOGIC;
47          debito_ld         : out STD_LOGIC;
48          debito_rst        : out STD_LOGIC;
49          pegar_ld          : out STD_LOGIC;
50          pegar_rst         : out STD_LOGIC;
51          devolver_ld       : out STD_LOGIC;
52          devolver_rst      : out STD_LOGIC;
53          cod_concat_ld     : out STD_LOGIC;
54          cod_concat_rst    : out STD_LOGIC;
55          codigo_ld         : out STD_LOGIC;
56          codigo_rst        : out STD_LOGIC;
57          memoria_ld        : out STD_LOGIC;
58          memoria_rst       : out STD_LOGIC;
59          ctrl_passagem_ld  : out STD_LOGIC;
60          ctrl_passagem_rst : out STD_LOGIC;
61
62          escolhe_comp                 : out STD_LOGIC;
63          escolhe_concat_memoria       : out STD_LOGIC;
64          escolhe_concat_ctrl_passagem : out STD_LOGIC;
65
66          led_cartao : out STD_LOGIC;
67          led_livro  : out STD_LOGIC
68        );
69      end component;
70
```

```
71      component datapath is
72        port (
73            clock                          : in STD_LOGIC;
74            pegar_in                       : in STD_LOGIC;
75            devolver_in                    : in STD_LOGIC;
76            id_in                          : in STD_LOGIC_VECTOR(15 downto 0);
77            cod_concat_in                  : in STD_LOGIC_VECTOR(15 downto 0);
78            codigo_in                      : in STD_LOGIC_VECTOR(15 downto 0);
79            escolhe_concat_memoria_in      : in STD_LOGIC;
80            escolhe_concat_ctrl_passagem_in : in STD_LOGIC;
81            escolhe_comp_in                : in STD_LOGIC;
82
83            -- registers
84            pegar_ld       : in STD_LOGIC;
85            pegar_rst      : in STD_LOGIC;
86            devolver_ld    : in STD_LOGIC;
87            devolver_rst   : in STD_LOGIC;
88            id_ld          : in STD_LOGIC;
89            id_rst         : in STD_LOGIC;
90            cod_concat_ld  : in STD_LOGIC;
91            cod_concat_rst : in STD_LOGIC;
92            codigo_ld      : in STD_LOGIC;
93            codigo_rst     : in STD_LOGIC;
94            debito_ld      : in STD_LOGIC;
95            debito_rst     : in STD_LOGIC;
96            passagem_ld    : in STD_LOGIC;
97            passagem_rst   : in STD_LOGIC;
98            memoria_ld     : in STD_LOGIC;
99            memoria_rst    : in STD_LOGIC;
100
101           -- outputs
102           passagem_out    : out STD_LOGIC_VECTOR(16 downto 0);
103           memoria_out     : out STD_LOGIC_VECTOR(31 downto 0);
104           comp_codigo_out : out STD_LOGIC;
105           debito_out      : out STD_LOGIC_VECTOR(15 downto 0);
```

```
106             devolver_out    : out STD_LOGIC;
107             pegar_out       : out STD_LOGIC
108         );
109     end component;
110 
111 begin
112 
113     instancia_controladora : controladora port map
114                              (clk,
115                               pegar, devolver, leitor_cartao, leitor_livro, reg_pegar, reg_devolver, compara_codigo,
116                               id_ld, id_rst, debito_ld, debito_rst, pegar_ld, pegar_rst,
117                               devolver_ld, devolver_rst, cod_contat_ld, cod_concat_rst,
118                               codigo_ld, codigo_rst, memoria_ld, memoria_rst, ctrl_passagem_ld, ctrl_passagem_rst,
119                               escolhe_comp, escolhe_concat_memoria,escolhe_concat_ctrl_passagem);
120 
121     instancia_datapath : datapath port map
122                          (clk,
123                           pegar, devolver, id, cod_concat, codigo, escolhe_concat_memoria, escolhe_concat_ctrl_passagem, es
124                           pegar_ld, pegar_rst, devolver_ld, devolver_rst, id_ld, id_rst,
125                           cod_contat_ld, cod_concat_rst, codigo_ld, codigo_rst, debito_ld, debito_rst,
126                           ctrl_passagem_ld, ctrl_passagem_rst, memoria_ld, memoria_rst,
127                           ctrl_passagem_out, memoria_out, compara_codigo, debito_out, reg_devolver, reg_pegar);
128 
129 end rtl;
```

- RTL Viewer


leitor_livro
clk
devolver
pegar
id[15..0]
cod_concat[15..0]
codigo[15..0]
rst
leitor_cartao
datapath:instancia_datapath
clock
cod_concat_ld
cod_concat_rst
codigo_ld
codigo_rst
debito_ld
debito_rst
devolver_in
devolver_ld
devolver_rst
escolhe_comp_in
escolhe_concat_ctrl_passagem_in
escolhe_concat_memoria_in
id_ld
id_rst
memoria_ld
memoria_rst
passagem_ld
passagem_rst
pegar_in
pegar_ld
pegar_rst
id_in[15..0]
cod_concat_in[15..0]
codigo_in[15..0]
comp_codigo_out
devolver_out
pegar_out
passagem_out[16..0]
memoria_out[31..0]
debito_out[15..0]
controladora:instancia_controladora
clk
compara_codigo
devolver
leitor_cartao
leitor_livro
pegar
req_devolver
req_pegar
rst
cod_concat_ld
cod_concat_rst
codigo_ld
codigo_rst
ctrl_passagem_ld
ctrl_passagem_rst
debito_ld
debito_rst
devolver_ld
devolver_rst
escolhe_comp
escolhe_concat_ctrl_passagem
escolhe_concat_memoria
id_ld
id_rst
led_cartao
led_livro
memoria_ld
memoria_rst
pegar_ld
pegar_rst
memoria_out[31..0]
ctrl_passagem_out[16..0]
led_cartao
led_livro
debito_out[15..0]

Simulação do sistema

De forma a validar a implementação do sistema feita pelo grupo e exemplificar o seu funcionamento, foi simulada uma situação possível de uso do sistema: a devolução de um livro por um usuário carregando o livro certo.

Neste caso, um usuário que está em débito com a biblioteca utiliza o sistema para devolver um livro e o livro que ele tenta devolver é o que ele deve à biblioteca, então o sistema permite a devolução, quita o débito do usuário e bloqueia a passagem do livro devolvido pelos portais anti furtos.

A simulação dessa situação no ModelSim está apresentada na Fig. 5 e será explicada em seguida:

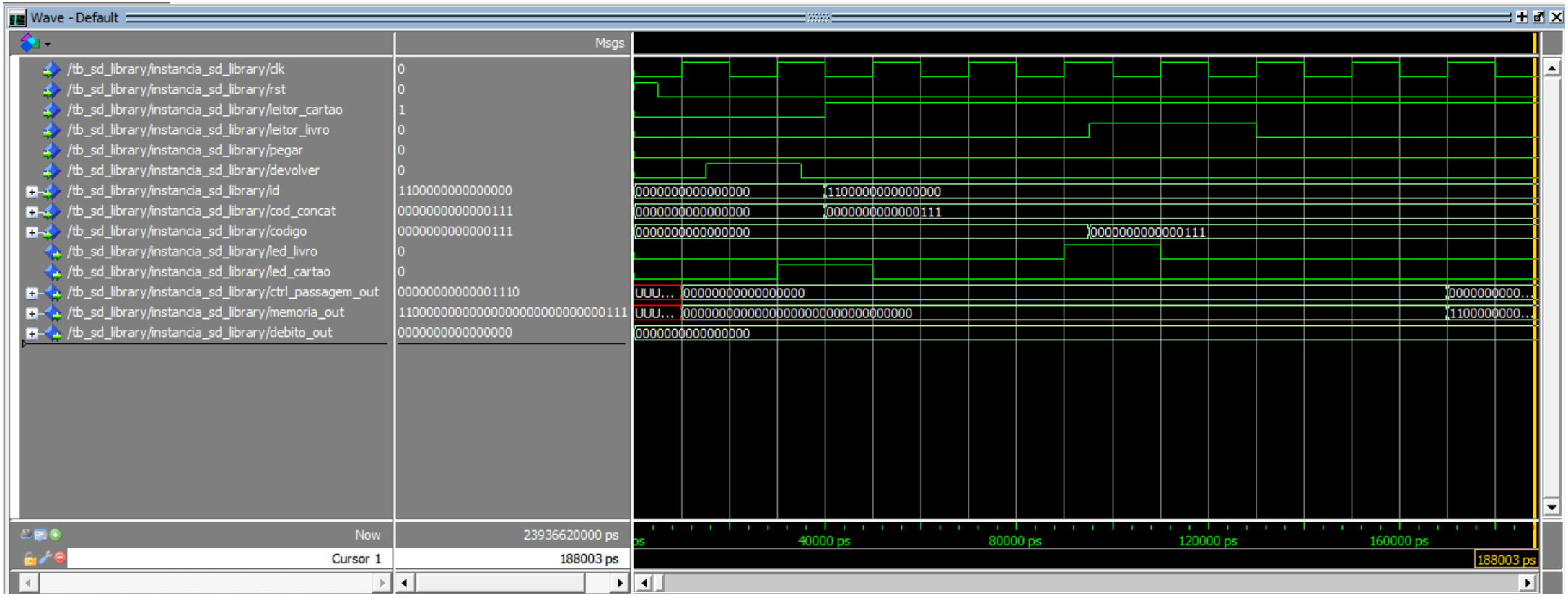
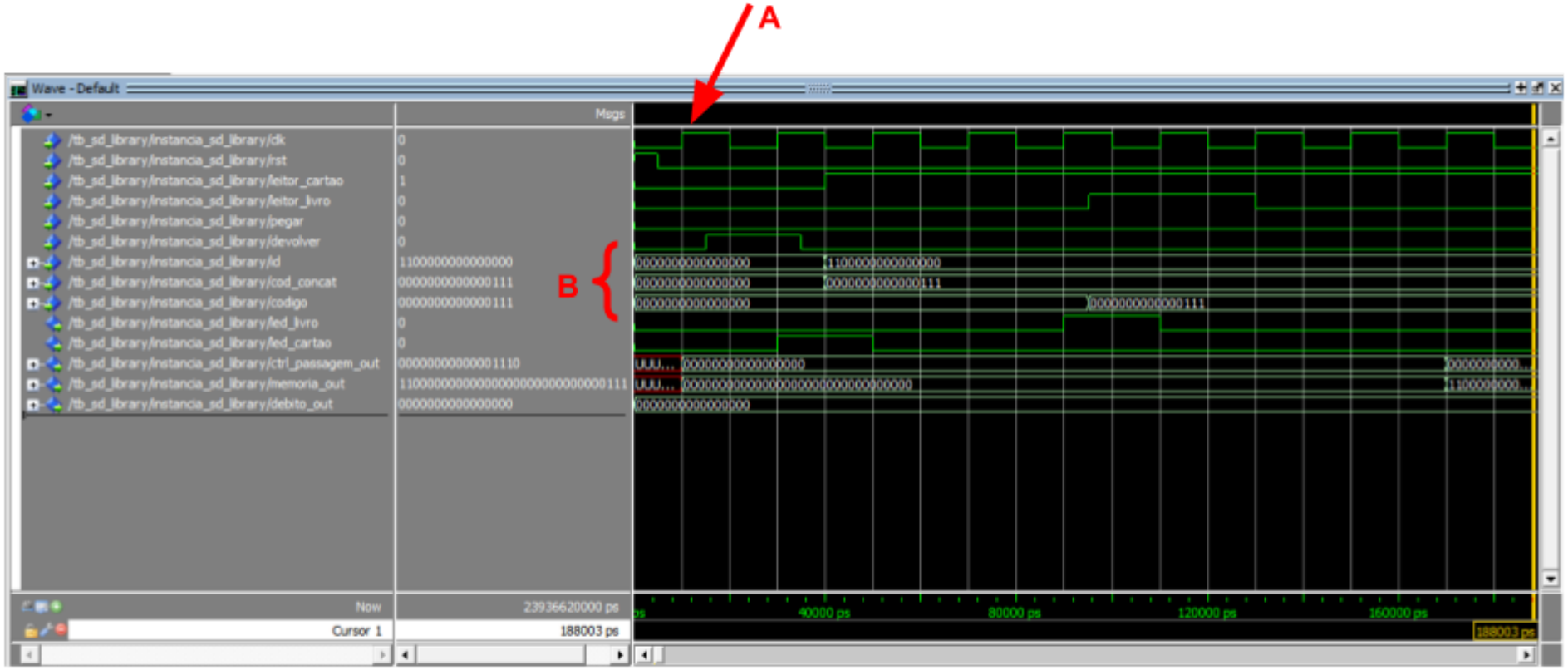


Figura 5 - Simulação do Sistema

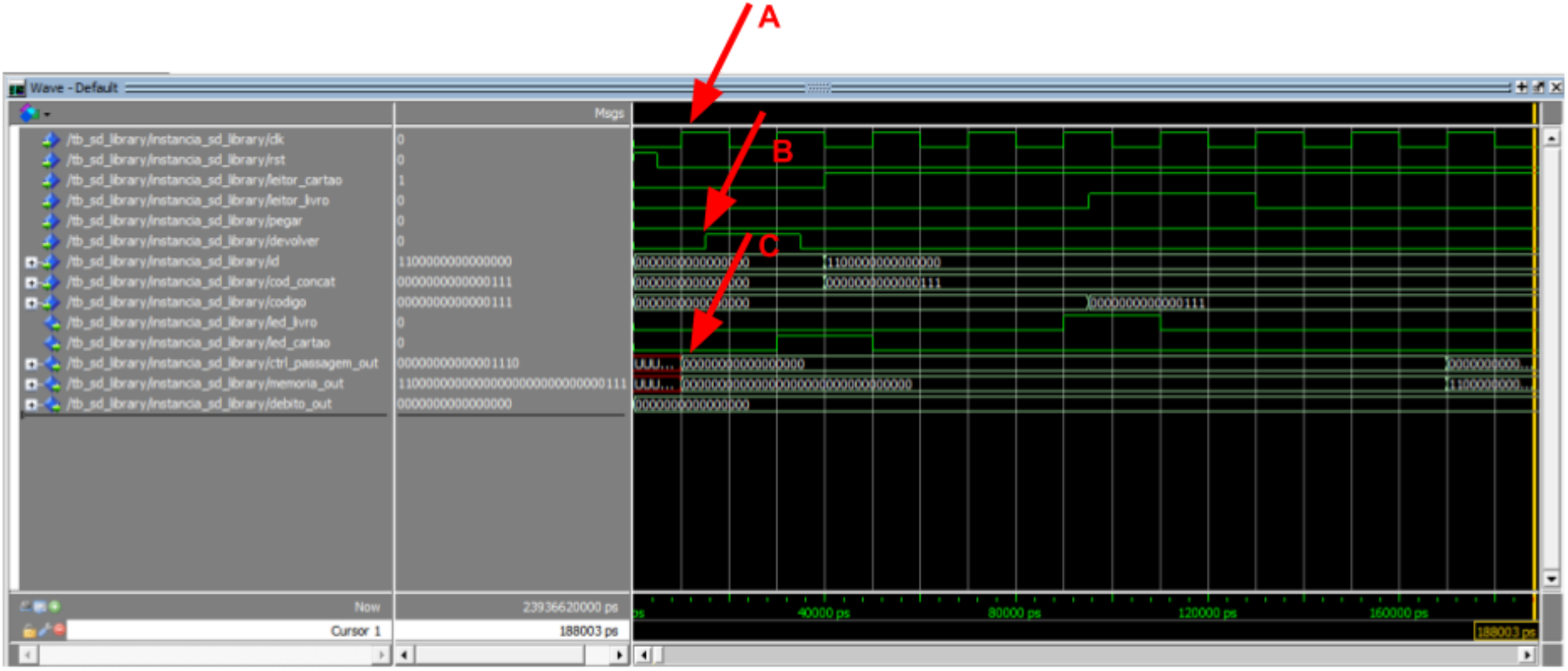
1. **Estado *Início***

Inicialmente, a entrada *rst* é colocada em nível lógico alto **(A)** para garantir que o sistema iniciará seu funcionamento com a máquina de estados no estado *Início.* Como o reset da FSM é independente de bordas de clock, a simulação já se inicia com a máquina no estado *Início* e os registradores têm sua entrada de reset ativada, fazendo o seu valor armazenado ser um vetor de zeros. Os únicos registradores que não são resetados nesse estado serão resetados no estado seguinte.



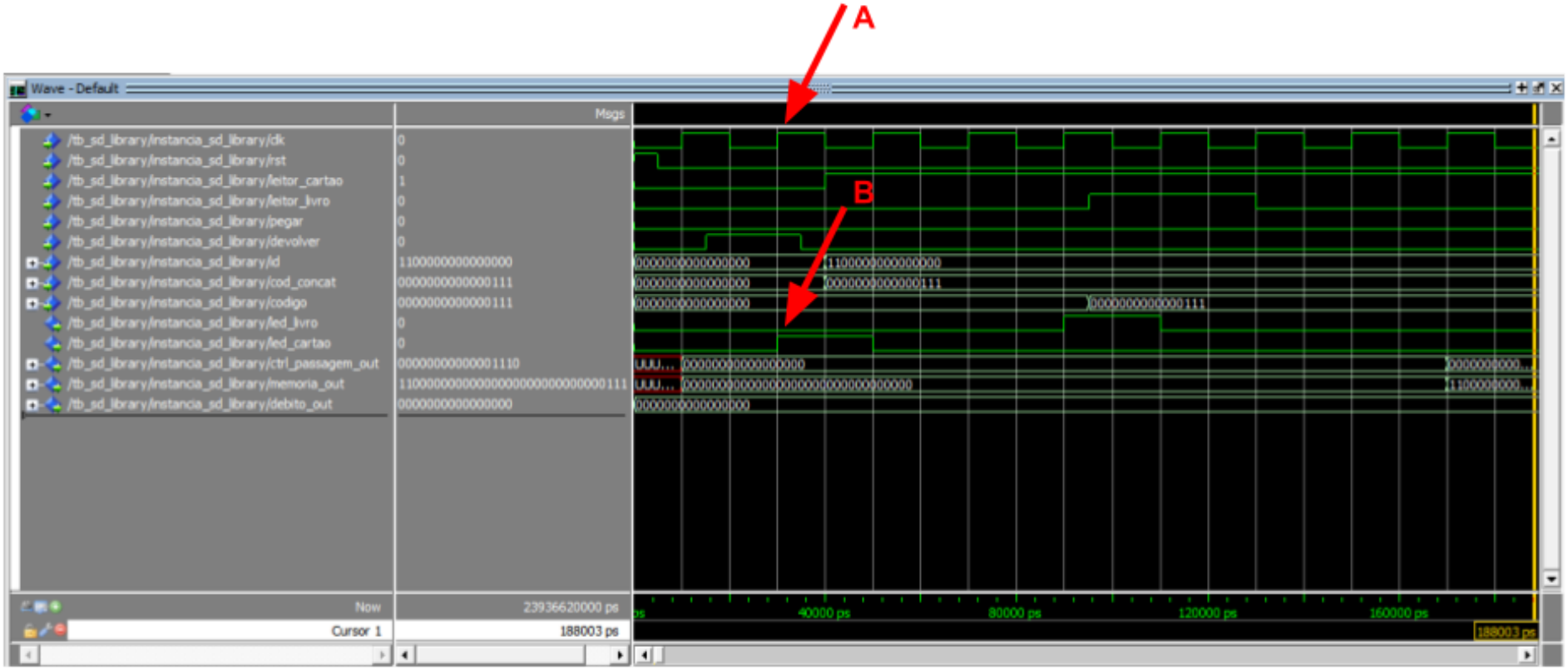
2. **Estado *Seleção***

Na primeira borda de subida do sinal de clock **(A)**, a FSM passa para o estado *Seleção*, resetando os registradores **(C)** que não haviam sido resetados no estado anterior e permitindo ao usuário escolher entre pegar e devolver um livro, Nesse caso, nota-se que a entrada *devolver* fica em nível lógico alto **(B)**, indicando que o usuário escolheu por devolver um livro.

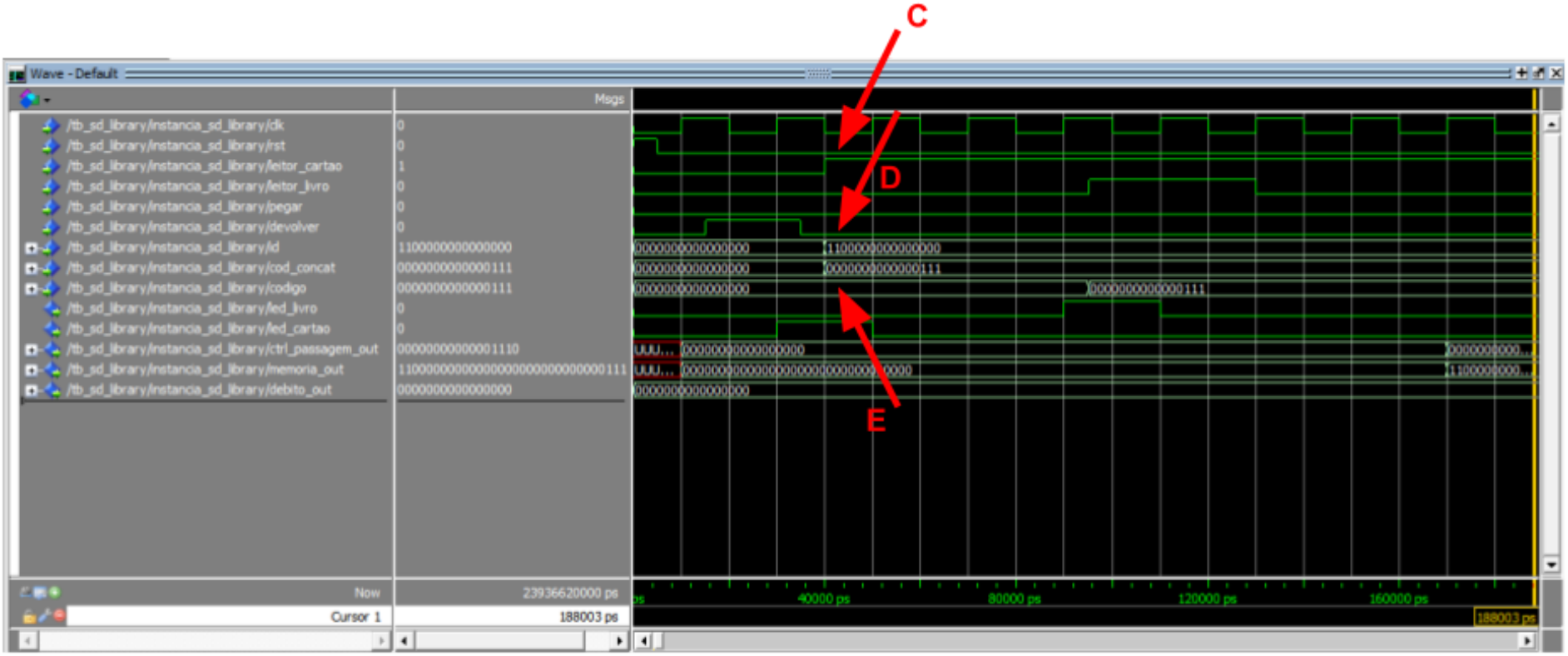


3. Estado *ArmazenaDados*

Na próxima borda de subida de clock **(A)**, a FSM passa para o estado *ArmazenaDados* e o LED do leitor de cartão (saída *led_cartão)* fica em nível lógico alto **(B)**, indicando ao usuário que ele deve inserir o seu cartão no leitor.

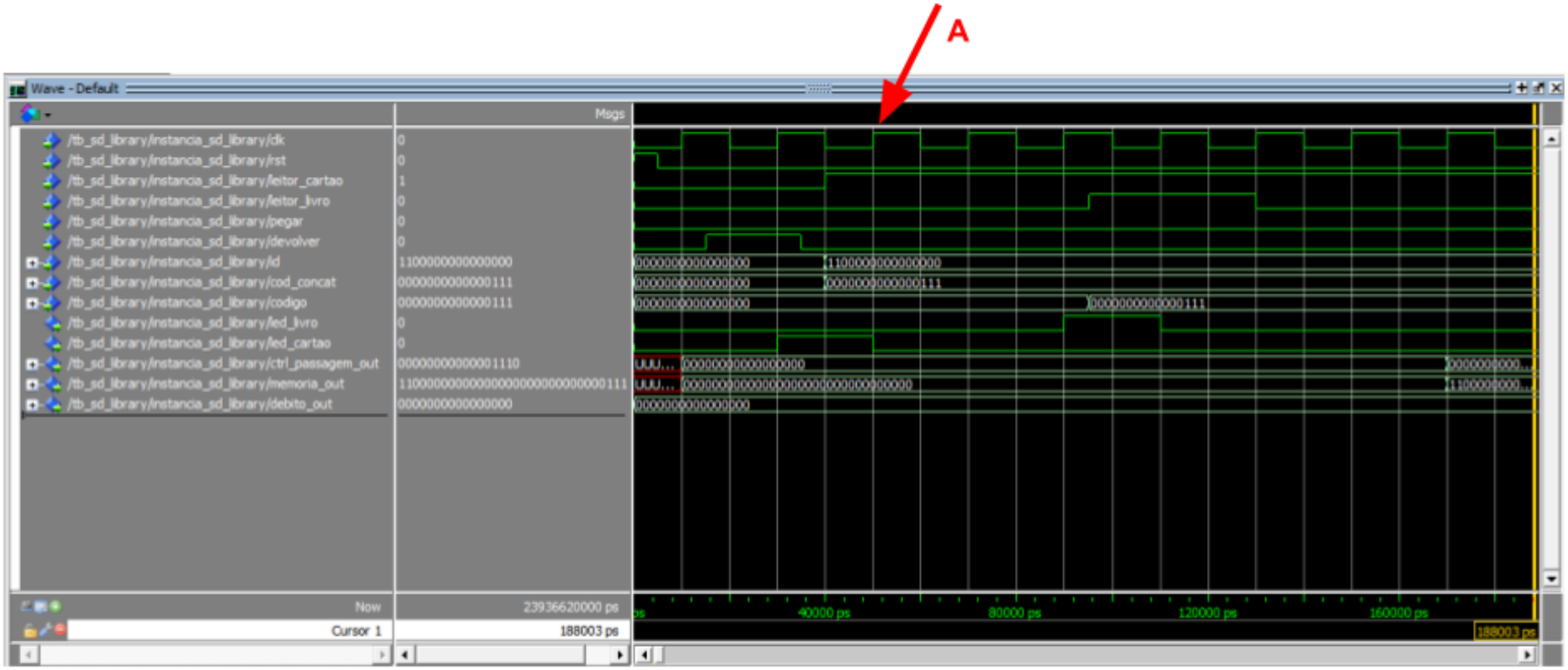


Alguns instantes depois, a entrada *leitor_cartão* - que indica se há ou não um cartão inserido no leitor - fica em nível lógico alto **(C)**, simulando a situação em que o usuário viu que o LED do leitor de cartão ficou aceso e inseriu o cartão na máquina. Com a inserção do cartão, os dados do usuários - *id* **(D)** e *cód_concat* **(E)** - são lidos pela máquina.



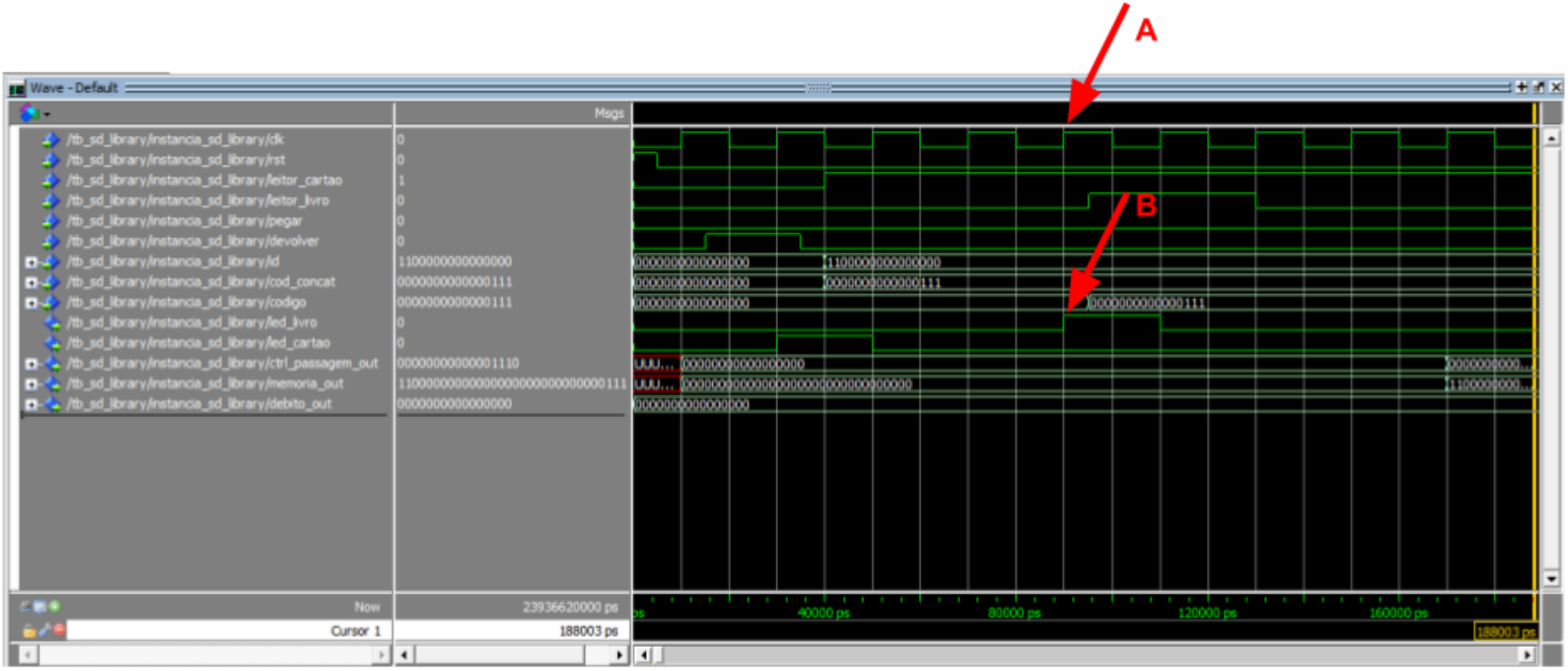
4. **Estado *VerificaDébito***

Na borda de subida de clock seguinte **(A)**, a FSM passa para o estado *VerificaDébito*, em que o código armazenado no cartão do aluno, correspondente à entrada *cód_concat*, é comparado a um vetor de zeros para verificar se ele está em débito (os dois são iguais) ou não (os dois são diferentes). Nesse caso, nota-se que o código concatenado é diferente de um vetor de zeros, indicando que o aluno apresenta débito na biblioteca.

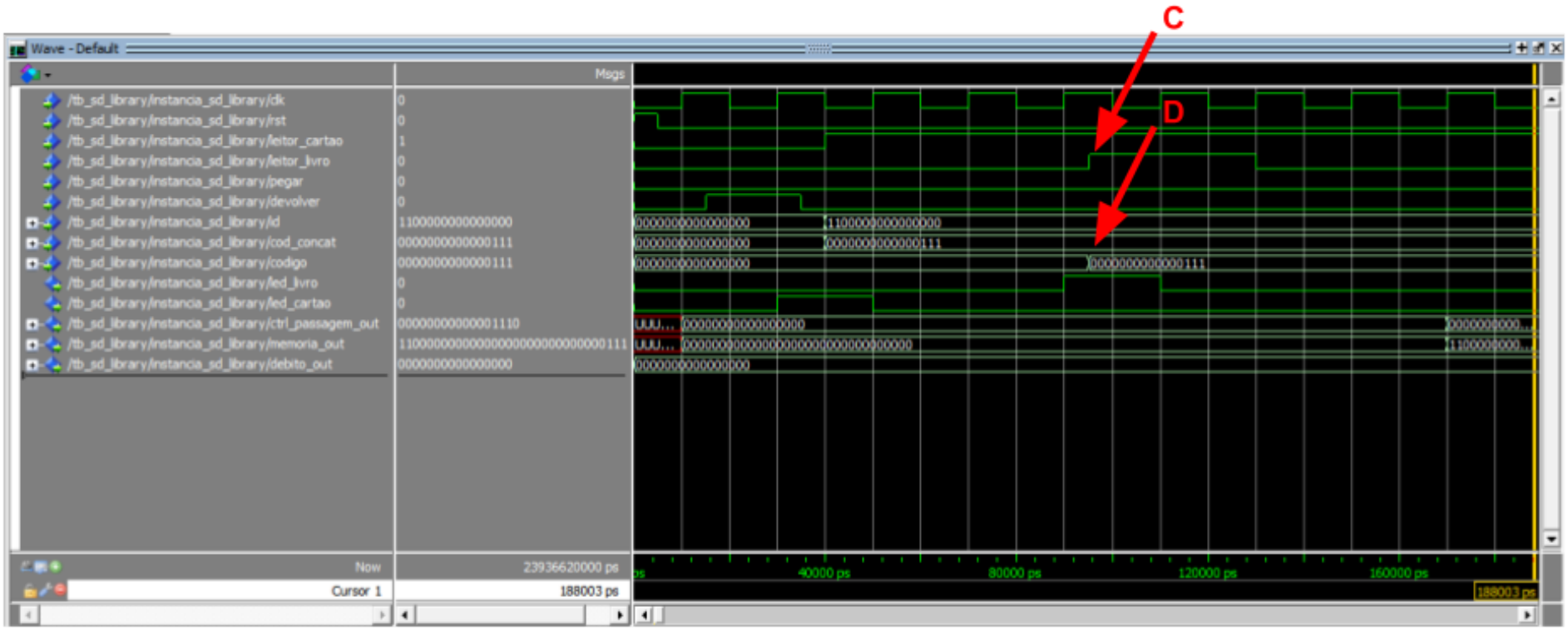


5. **Estado *LêCódigo***

Na borda seguinte de subida do sinal de clock **(A)**, como o aluno escolheu devolver um livro e apresenta débito, o sistema passa para o estado *LêCódigo*. Nesse estado, o LED do leitor de códigos é colocado em nível lógico alto **(B)**, indicando que o aluno deve aproximar dele o livro que deseja devolver.

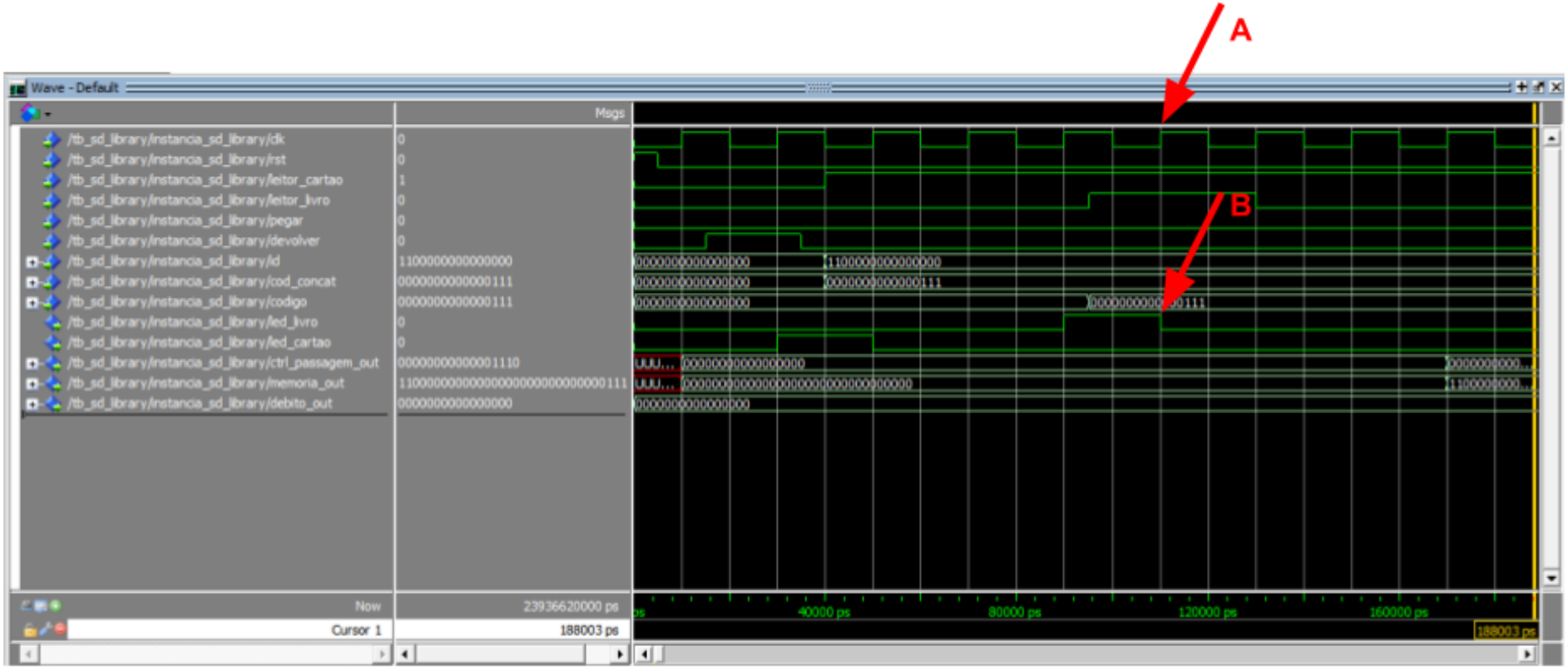


Ainda nesse estado, simulamos a situação em que o aluno viu o LED do leitor de códigos aceso e aproximou o livro dele, colocando a entrada *leitor_ livro* em nível lógico alto **(C)** e fazendo o código do livro que o aluno está tentando devolver aparecer na entrada *código* **(D)**.



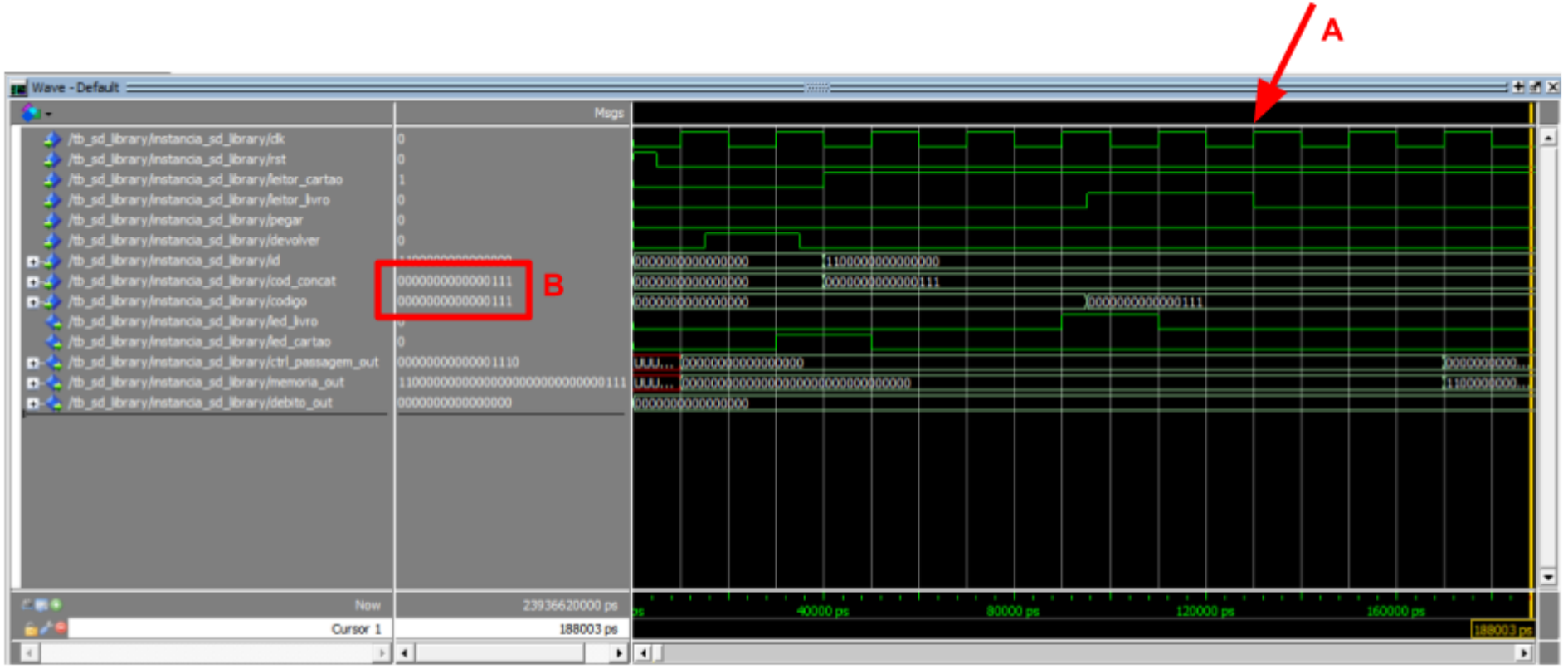
6. **Estado *ArmazenaCódigo***

Na borda seguinte de subida do clock **(A)**, o sistema passa para o estado *ArmazenaCódigo,* desligando o LED do leitor de livros **(B)** (o que indica que o aluno já pode afastar o livro do leitor).



7. Estado *UsaCódigo*

Na próxima borda de subida do clock **(A)**, o sistema passa para o estado *UsaCódigo,* em que o código do livro lido é comparado ao código concatenado lido do cartão do aluno para verificar se ele está tentando devolver o livro correto. Nota-se, por **(B)** que os códigos, nesse caso, são iguais, então o aluno poderá devolver o livro que carrega em mãos.



8. Estado *DevolveLivro*

Como o aluno escolheu devolver um livro e carrega o livro correto, a FSM passa para o estado *DevolveLivro* na borda de subida de clock seguinte **(A)**. Nesse estado, o débito do aluno é quitado, armazenando na memória do seu cartão o seu número de identificação (*id)* concatenado a um vetor de zeros, como pode-se ver na saída *memória* **(B)**. Além disso, o sistema envia para os portais antifurtos um sinal indicando que o livro devolvido não pode sair da biblioteca, concatenando ao código desse livro um 'o', como pode-se vee na saída *ctrl_ passagem* **(C)**. Com isso, o aluno consegue devolver com sucesso o livro que devia e conclui-se o funcionamento do sistema.