centro universitário FEI

CURSO DE ENGENHARIA ELÉTRICA

NE7720 – LABORATÓRIO DE SISTEMAS DIGITAIS II

1° SEMESTRE DE 2021

# RELATÓRIO DO PROJETO 1: COFRE DIGITAL DE SEGURANÇA

| Nome do Aluno | Número (R.A.) | Turma |
|---|---|---|
| DANILO H. B. DOS SANTOS | 12.218.079-7 | 615 |

## INTRODUÇÃO

O objetivo geral deste projeto 1 do Laboratório de Sistemas Digitais II é desenvolver um sistema digital de controle para comandar a abertura de um cofre digital de segurança. Devemos desenvolver uma máquina de estado finita (FSM) e a lógica necessária para comando do sinal de abertura do cofre em um FPGA, utilizando a linguagem de descrição VHDL implementada no software Quartus Prime Lite Edition 16.1 e o método de projeto de um bit por estado (utilizando equações de estado). Bits armazenados significam que o circuito tem memória, o que é também conhecido como estado, resultando os chamados circuitos sequenciais. A senha para desbloqueio do cofre digital a ser desenvolvido é baseada no número de matrícula (R.A) descrito no cabeçalho deste documento.

Esse laboratório também tem como objetivos exercitar a metodologia de desenvolvimento de projetos de engenharia apoiada em computador (CAE) onde todo o desenvolvimento do software embarcado e modelo final será feito no software Quartus Prime Lite Edition 16.1 onde será feita a simulação funcional e interface final do sistema de controle digital totalmente implementada utilizando o FPGA da família MAX 10 (modelo: 10M50DAF484C7G) existente na placa de desenvolvimento Altera DE10-Lite. Esse processo envolve a compreensão do problema, seu planejamento, desenvolvimento da solução lógica, integração dos subsistemas, implementação no ambiente computacional, simulação, testes, depuração do projeto, implementação física e registro dos resultados.

A implementação do sistema de controle de abertura do cofre digital será feita através do código de segurança composto pelo acionamento correto de um conjunto de números associado aos dígitos do número de matrícula (R.A.). Esta relação é dada pela Tabela 1 abaixo, utilizando a parametrização de codificação descrita no roteiro do Projeto. O sistema a ser implementado deve possuir um teclado com dez chaves numéricas (acionadas em nível lógico alto), denominadas SW0, SW1, ..., SW9 que serão simuladas pelas chaves da placa DE10-Lite. O sistema também deve possuir um botão FECHA (botão KEY0) que, quando acionado gera o comando de fechamento do cofre O sistema possui um botão ABRE (botão KEY1) que, quando acionado, gera o comando que solicita a abertura do cofre. Entretanto, a abertura do cofre só ocorre caso a sequência de segurança tenha sido corretamente inserida.

Figura 1: Diagrama esquemático para o painel do cofre digital (Fonte: Roteiro do Projeto 1)

O Dígito de Controle no R.A. nos define o Número de Tentativas (NT) máximo no permitido para abertura do cofre pelo usuário, e para os demais dígitos (NA, NB, NC, ND, NE, NF) será definido a sequência de segurança (senha) para abertura do cofre digital, conforme Tabela 2.

Tabela 1: Parametrização da codificação do cofre digital de segurança para o R.A.

| RA | 1 | 2 | 2 | 1 | 8 | 0 | 7 | 9 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| SEQ | N/A | N/A | NF | NE | ND | NC | NB | NA | NT |
| | | | 2 | 1 | 8 | 0 | 7 | 9 | 3 |

Tabela 2: Sequência de Segurança do cofre digital para o R.A.

| SENHA | NA | NB | NC | ND | NE | NF |
|---|---|---|---|---|---|---|
| | 9 | 7 | 0 | 8 | 1 | 2 |

O sistema pode ser reiniciado quando as chaves SW8 e SW9 forem **simultaneamente acionados**. Esse acionamento deve conduzir o sistema ao seu estado inicial (**INICIO**), **de modo assíncrono**, independente do clock e de qual seja o estado atual do sistema.

Enquanto o sistema estiver no estado de **INICIO** o display de estado do cofre deve mostrar a letra "A", informando a condição de cofre ABERTO (um Led também deve indicar essa condição). Para travar o cofre o usuário deve acionar o botão FECHA. O display de estado do cofre deve mostrar a letra "F", informando a condição de cofre FECHADO (um Led também deve indicar essa condição).

Para abrir novamente o cofre o usuário deve inserir o código de segurança na sequência correta, seguido do acionamento do botão ABRE. Caso essa sequência ocorra de forma correta o display de estado do cofre deve mostrar a letra "A", informando a condição de cofre ABERTO. As chaves numéricas devem ser acionadas individualmente, partindo sempre da condição que a chave deve ser acionada e a seguir desativada (liberada). O número da chave selecionada deve ser apresentado em um display (Número Acionado). Se duas ou mais chaves forem acionadas simultaneamente o sistema não deve aceitar o valor de nenhuma das chaves acionadas.

O usuário deve ter um número máximo de tentativas permitido (NT). Caso esse número de tentativas seja atingido o sistema entra no estado de BLOQUEIO, mostrando a letra "b" no display de estado do cofre (um Led também deve indicar essa condição). O sistema só sai do estado de BLOQUEIO com o acionamento com o reinício do sistema, com o acionamento simultâneo das chaves SW8 e SW9 (essa funcionalidade não é informada ao usuário, sendo um comando reservado para o proprietário do cofre).

Qualquer erro na sequência de acionamento de segurança (números incorretos, ordem da sequência incorreta, sequência incompleta, sequência com inclusão de excesso de números) leva o sistema ao estado de **FALHA** (indicado por um Led, não visível pelo usuário). A partir do estado de **FALHA**, quando o usuário solicitar a abertura do cofre (acionando o botão ABRE) o display de estado do cofre deve mostrar a letra "E", informando a condição de ERRO da sequência.

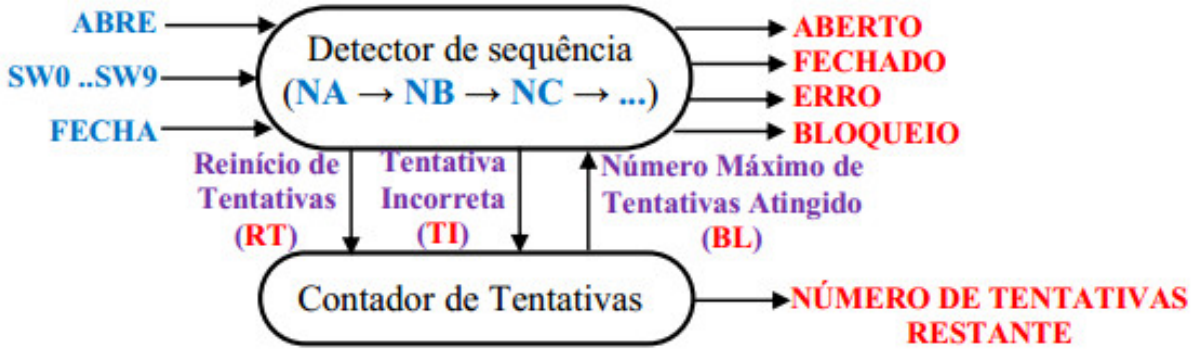
A sinalização de ERRO deve permanecer no display de estado do cofre apenas enquanto o botão ABRE estiver ativo. Quando o sinal ABRE for desativado, se o número máximo de tentativas de abertura (NT) não foi atingido o sistema deve aguardar nova sequência numérica, mostrando a letra "F" no display de estado do cofre. Se número de tentativas máximo foi atingido o sistema entra no estado de BLOQUEIO.

Podemos observar que enquanto o sistema estiver no estado de **FALHA** nenhuma sinalização é gerada ao usuário. Apenas quando é solicitada a abertura do cofre (acionamento do botão ABRE) é que a informação de ERRO é apresentada.

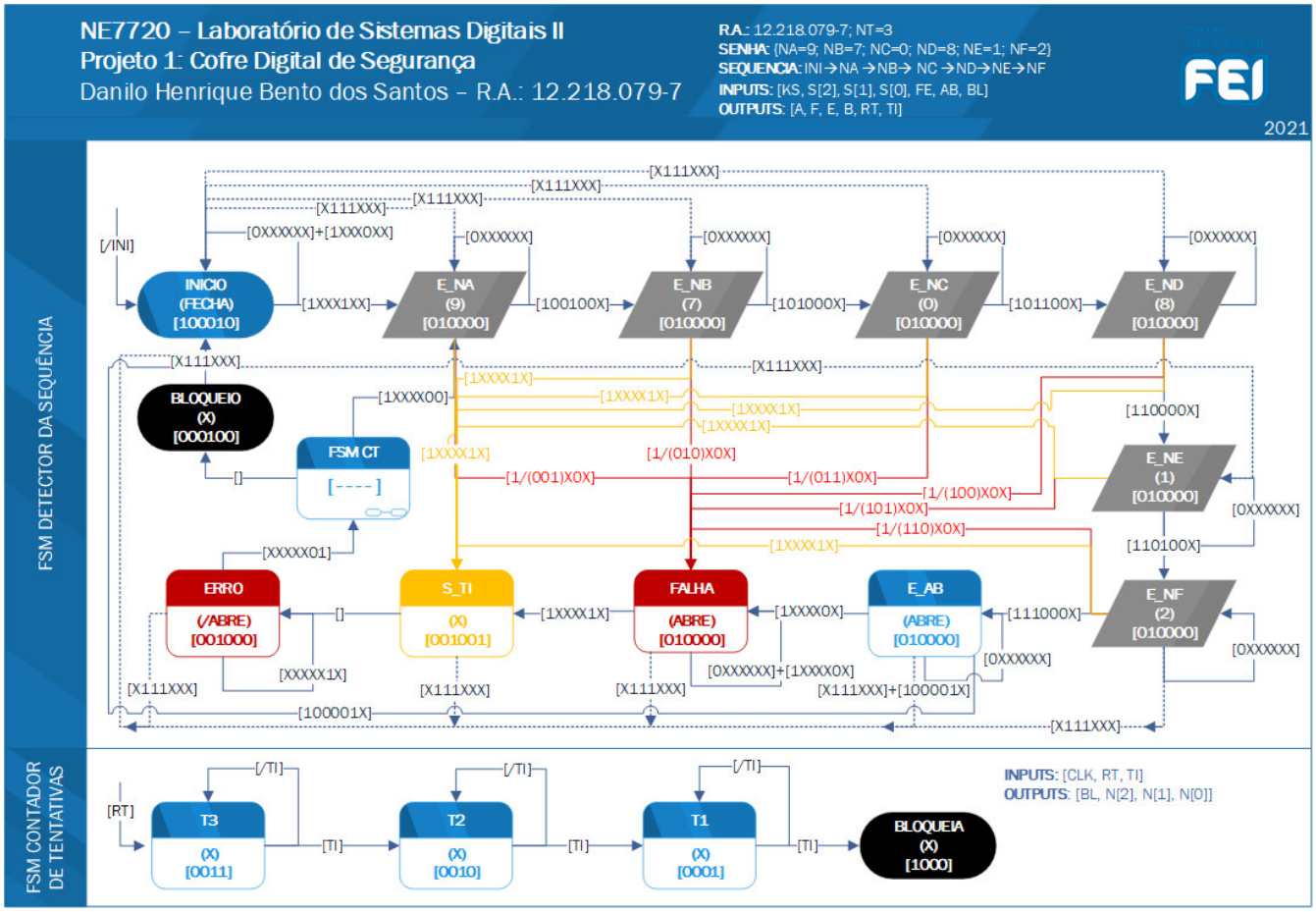
O sistema deve ter um display para mostrar o número de tentativas de abertura restante. A abertura do cofre com sucesso deve fazer o sistema retornar ao estado de **INICIO**, reiniciando a contagem do número de tentativas de abertura (o display de tentativas deve mostrar o número NT e o display de estado do cofre deve mostrar a letra "A").

Para realizar o controle da sequência numérica de abertura do cofre e do contador do número de tentativas de abertura devem ser projetadas duas máquinas de estado inter-relacionadas, conforme representado no diagrama de blocos apresentado na Figura 2.

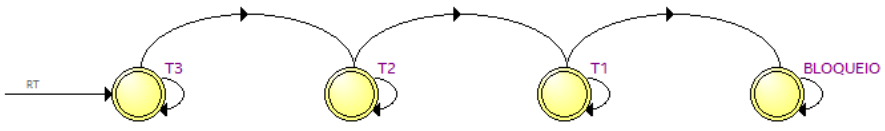
Figura 2: Diagrama de blocos geral das Máquinas de Estado (FSM) envolvidas no controle do cofre digital (Fonte: Roteiro do Projeto 1)



# 2. DIAGRAMA DE ESTADOS (FSM)

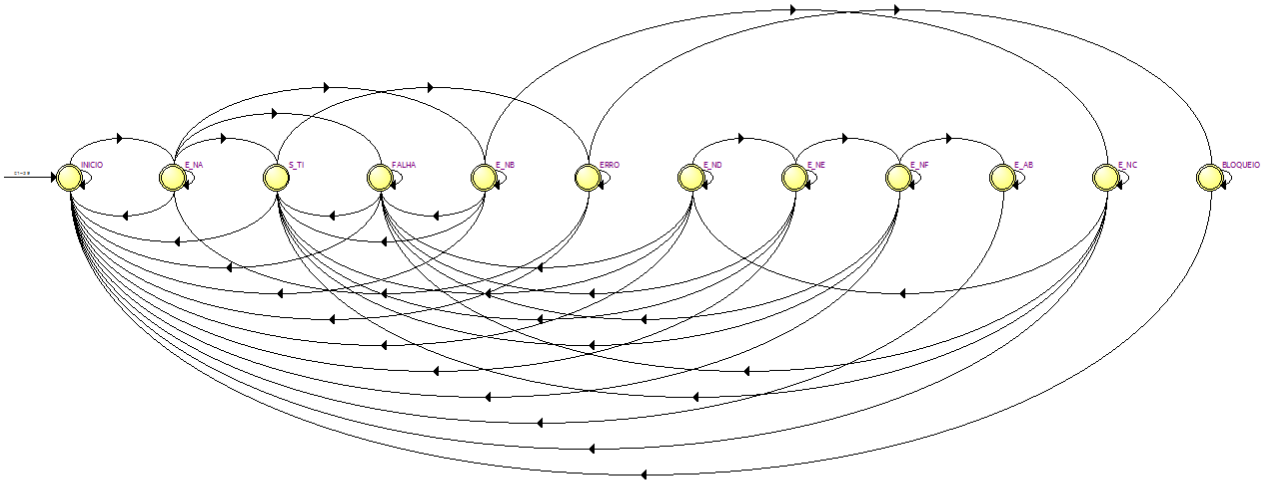



## CONTADOR DE TENTATIVAS (STATE MACHINE VIEWER)




| | Source State | Destination State | Condition |
|---|---|---|---|
| 1 | BLOQUEIO | BLOQUEIO | |
| 2 | T1 | T1 | (TI) |
| 3 | T1 | BLOQUEIO | (TI) |
| 4 | T2 | T1 | (TI) |
| 5 | T2 | T2 | (ITI) |
| 6 | T3 | T2 | (TI) |
| 7 | T3 | T3 | (ITI) |

**DETECTOR DA SEQUÊNCIA DE TENTATIVAS (STATE MACHINE VIEWER)**



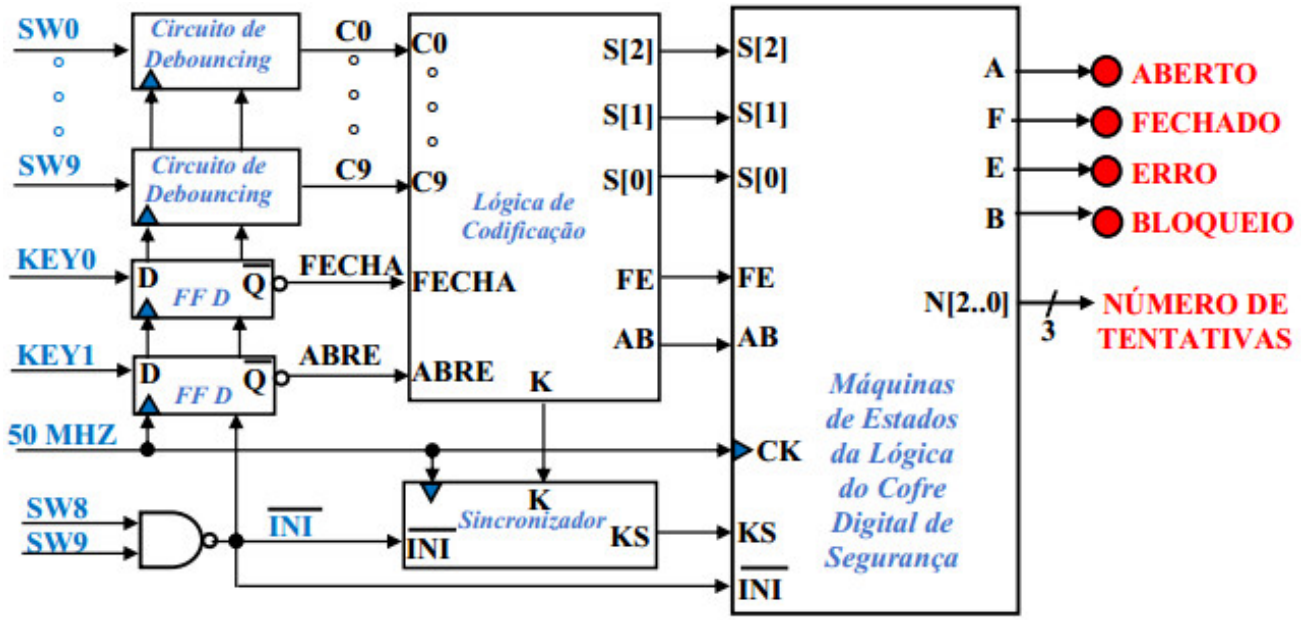

## 3. REQUISITOS DE DESENVOLVIMENTO DO PROJETO DIGITAL

A seguir, será apresentada a implementação das máquinas de estados (**FSM**) em **VHDL** para representação da lógica de sequenciamento dos códigos e de contagem do número de tentativas de abertura, conforme descrito no item 1. Cada **FSM** foi projetada utilizando-se a **metodologia de projeto de um bit por estado** (baseada em equações de estados), conforme visto no item anterior.

A descrição de arquitetura em **VHDL** será realizada utilizando-se o **formato de fluxo de dados** e o **formato comportamental**. A interligação dos elementos será realizada utilizando-se o **formato estrutural** (representando os **códigos VHDL em componentes**). O clock do sistema deve será o sinal de 50MHZ já disponível na placa DE10-Lite. O sistema deve será completamente síncrono, ou seja, todos os blocos devem ser sincronizados com o sinal de 50 MHz. A Figura 3 apresenta o diagrama de blocos principal do sistema de controle do cofre, contendo:

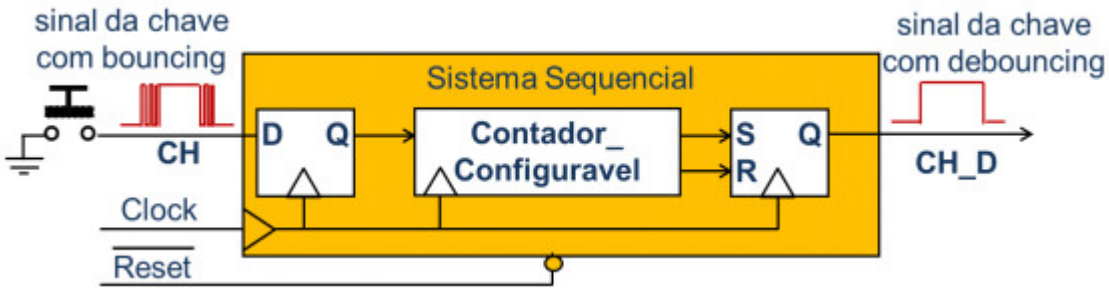
- Circuitos para eliminação de vibração (deboucing) das chaves;
- Circuitos para sincronização de botões (flip-flops tipo D);
- Circuito com a lógica de codificação do acionamento das chaves;
- Circuito com o sincronizador do acionamento das chaves;
- O bloco da FSM que contém a lógica de controle do cofre digital de segurança;

Figura 3: Diagrama de blocos global do sistema de controle do Cofre Digital de Segurança
(Fonte: Roteiro do Projeto 1)




As dez chaves utilizadas para entradas dos números secretos (SW0 a SW9) serão simuladas com o acionamento das chaves deslizantes da placa DE10-Lite. Estas chaves possuem trepidação de contatos (bouncing) e devem ser sincronizadas com o clock do sistema para evitar problemas de metaestabilidade (VAHID, 2008). Para eliminar essa vibração o sinal de cada chave passará por um circuito de "debouncing" e cujo diagrama de blocos é representado na Figura 4.

Figura 4: Diagrama de blocos do circuito eliminador de trepidação de chaves
(Fonte: Roteiro do Projeto 1)



Os botões **KEY0** e **KEY1** já possuem circuitos de debouncing associados no próprio hardware, mas precisam ser sincronizados com o clock através de Flip-Flops tipo D. Como esses botões **possuem lógica de ativação negada** devem ser utilizadas as saídas complementares (/Q) dos biestáveis para a ativação dos sinais **FECHA** e **ABRE** em nível lógico um;

A Lógica de Codificação será utilizada para simplificar o projeto da FSM. Essa lógica deve fornecer um código de três bits (S[2], S[1], S[0]) que representa qual chave foi acionada. Como interessam apenas as chaves associadas ao código secreto podem ser geradas códigos apenas para essas chaves (o acionamento das demais chaves é considerado erro na sequência de segurança). A Tabela 3 apresenta um exemplo de atribuição de códigos, mas cada aluno pode fazer a codificação que achar mais adequada.

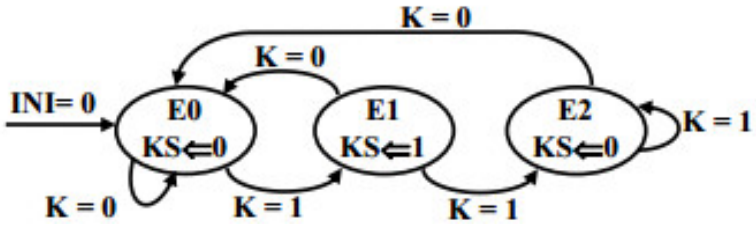
Tabela 3: Lógica de Codificação para detecção das chaves ativadas na sequência de segurança do cofre digital para o R.A.

| LÓGICA DE CODIFICAÇÃO CHAVE | | | | |
|---|---|---|---|---|
| CÓDIGOS | | | | |
| S[2] | S[1] | S[0] | EXPRESSÕES LÓGICAS CHAVES | NÚMERO |
| 0 | 0 | 0 | NENHUMA CHAVE ACIONADA ISOLADAMENTE | ERRADO |
| 0 | 0 | 1 | O9*Õ8*Õ7*Õ6*Õ5*Õ4*Õ3*Õ2*Õ1*Õ0 | 9 |
| 0 | 1 | 0 | Õ9*Õ8*O7*Õ6*Õ5*Õ4*Õ3*Õ2*Õ1*Õ0 | 7 |
| 0 | 1 | 1 | Õ9*Õ8*Õ7*Õ6*Õ5*Õ4*Õ3*Õ2*Õ1*O0 | 0 |
| 1 | 0 | 0 | Õ9*O8*Õ7*Õ6*Õ5*Õ4*Õ3*Õ2*Õ1*Õ0 | 8 |
| 1 | 0 | 1 | Õ9*Õ8*Õ7*Õ6*Õ5*Õ4*Õ3*Õ2*O1*Õ0 | 1 |
| 1 | 1 | 0 | Õ9*Õ8*Õ7*Õ6*Õ5*Õ4*Õ3*O2*Õ1*Õ0 | 2 |

A Lógica de Codificação deve gerar um sinal **K** que é ativado (em nível lógico um) enquanto **qualquer chave** estiver acionada (ou seja, **K='0'** significa que **nenhuma** chave está acionada).

A ativação do sinal **K** em nível lógico um deve gerar um sinal (**KS**) sincronizado com o sinal de clock do sistema (50 MHZ). O sinal **KS** deve ter a duração **exata de um período de clock**, independentemente do tempo em que a chave permanecer acionada. A **FSM** só deve mudar de estado enquanto o sinal **KS** estiver ativo. O diagrama de estados representado na Figura 5 representa a máquina de estados do circuito Sincronizador (exemplo 3.9 de Vahid, 2008).

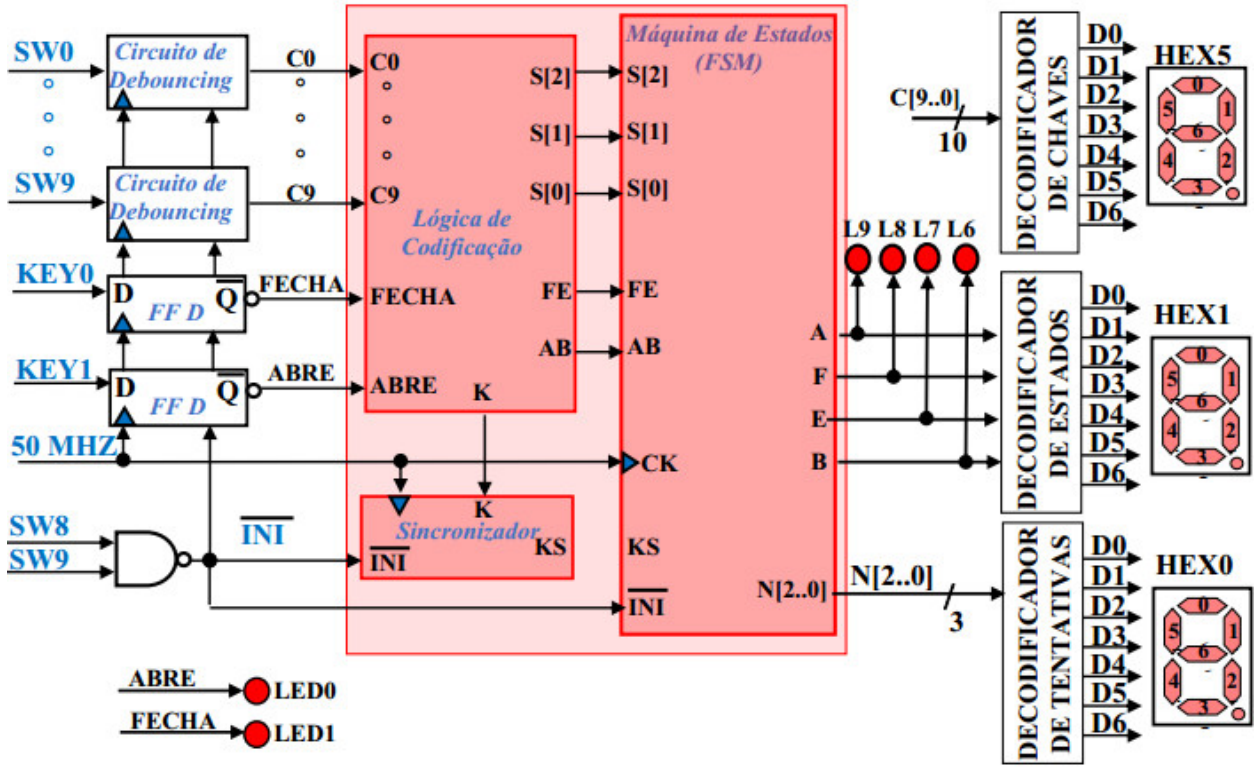
Figura 5: Diagrama de estados do circuito Sincronizador do sinal K (acionamento de uma chave).



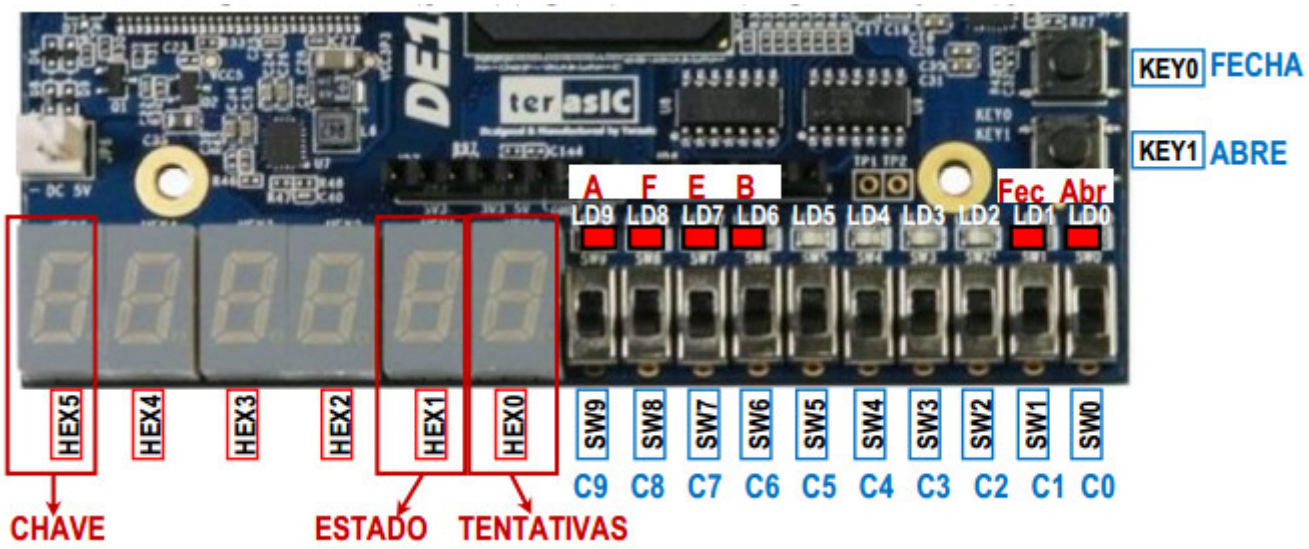
O sistema contará com um display de sete segmentos **HEX5** para apresentar o **número da chave acionada**. Quando **nenhuma chave** estiver acionada ou quando mais que uma chave estiver acionada o display **ficará apagado**. Os displays são de ânodo comum, logo possuem lógica de ativação negada, (acendem segmentos com nível lógico zero no sinal de saída do FPGA).

Além disso, o sistema possui um display **HEX1** para apresentar uma letra correspondente ao estado atual da máquina de estados (**A, F, E** ou **b**). O display **HEX0** apresenta o número de tentativas de abertura restantes. Serão utilizados também quatro Leds vermelhos (**LED9** a **LED6**) para exibir o estado do sistema (Aberto, Fechado, Erro e Bloqueio), conforme representado na Figura 6. Os Leds vermelhos: **LED0** e **LED1** devem acender para indicar, respectivamente, a ativação dos sinais ABRE e FECHA.

Figura 6: Diagrama blocos completo do sistema de controle do cofre digital de segurança



## 3. INTERFACES DE SINALIZAÇÃO DE ESTADO




## 4. PROJETO LÓGICA DE CODIFICAÇÃO DOS PARÂMETROS (VHDL)

```
1   -- Quartus Prime VHDL FEI DIGITAL REPORT: LOGICA DE CODIFICAÇÃO
2
3   LIBRARY IEEE; -- declaracao de bibliotecas
4   USE IEEE.STD_LOGIC_1164.ALL;
5
6   ENTITY LOGICA IS -- declaracao da entidade
7      PORT(
8      C            : IN STD_LOGIC_VECTOR(9 DOWNTO 0); -- entrada das chaves
9      FECHA, ABRE  : IN STD_LOGIC; -- entrada dos botoes
10     S            : OUT STD_LOGIC_VECTOR(2 DOWNTO 0); -- saida de codigo
11     FE, AB, K    : OUT STD_LOGIC
12     );
13  END LOGICA;
14
15  ARCHITECTURE SELETOR OF LOGICA IS -- arquitetura da logica de codificacao
16  BEGIN
17     WITH C SELECT
18        S <= "001" WHEN "1000000000", -- chave 9
19             "010" WHEN "0010000000", -- chave 7
20             "011" WHEN "0000000001", -- chave 0
21             "100" WHEN "0100000000", -- chave 8
22             "101" WHEN "0000000010", -- chave 1
23             "110" WHEN "0000000100", -- chave 2
24             "111" WHEN "1100000000", -- chave 8 e 9 (Reset)
25             "000" WHEN OTHERS;       -- chaves invalidas
26        K <= (C(9) OR C(8) OR C(7) OR C(6) OR C(5) OR C(4) OR C(3) OR C(2) OR C(1) OR C(0) OR FECHA OR ABRE);
27        FE <= FECHA;
28        AB <= ABRE;
29  END SELETOR;
```

# 5. PROJETO LÓGICO DA MÁQUINA DE ESTADOS (VHDL)

```
1   -- Quartus Prime VHDL FEI DIGITAL REPORT: FSM DETECTOR DE SEQUÊNCIA
2
3   LIBRARY ieee;
4   USE ieee.std_logic_1164.all;
5
6   ENTITY SEQUENCIA IS
7       PORT (  INI, CK, KS         : IN STD_LOGIC;
8               AB, FE, BL          : IN STD_LOGIC;
9               S                   : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
10              RT, TI, A ,F, E, B  : OUT STD_LOGIC;
11              FL                  : OUT STD_LOGIC
12              );
13  END SEQUENCIA;
14
15  ARCHITECTURE BEHAVIOR OF SEQUENCIA IS
16      TYPE ME_1 IS (INICIO, E_NA, E_NB, E_NC, E_ND, E_NE, E_NF, E_AB, FALHA, S_TI, ERRO, BLOQUEIO);
17
18      SIGNAL E1         : ME_1;
19      SIGNAL ENTRADAS   : STD_LOGIC_VECTOR(6 DOWNTO 0);
20      SIGNAL SAIDAS     : STD_LOGIC_VECTOR(5 DOWNTO 0);
21
22
23
24  BEGIN
25      Entradas <= KS&S(2)&S(1)&S(0)&FE&AB&BL;      -- concatenação dos sinais de entrada RE e RD como um vetor
26
27      DETECTOR : PROCESS (CK, INI) -- processo para definição das transições dos estados
28      BEGIN
29          IF (INI='0') THEN E1 <= INICIO;              -- estado de reset do sistema
30          ELSIF (CK'event and CK='1') THEN -- detecção de borda de CLK
31              CASE E1 IS                                  -- sincronização com CLK_EN
32                  WHEN INICIO =>
33                      IF KS = '1' AND FE = '1' THEN E1 <= E_NA;
34                      ELSE E1 <= INICIO;
35                      END IF;
36                  WHEN E_NA =>
37                      IF ENTRADAS = "1001000" THEN E1 <= E_NB;
38                      ELSIF KS = '1' AND AB = '0' THEN E1 <= FALHA;
39                      ELSIF KS = '1' AND AB = '1' THEN E1 <= S_TI;
40                      ELSIF S(2) = '1' AND S(1) = '1' AND S(0) = '1' THEN E1 <= INICIO;
41                      ELSE E1 <= E_NA;
42                      END IF;
43                  WHEN E_NB =>
44                      IF ENTRADAS = "1010000" THEN E1 <= E_NC;
45                      ELSIF KS = '1' AND AB = '0' THEN E1 <= FALHA;
46                      ELSIF KS = '1' AND AB = '1' THEN E1 <= S_TI;
47                      ELSIF S(2) = '1' AND S(1) = '1' AND S(0) = '1' THEN E1 <= INICIO;
48                      ELSE E1 <= E_NB;
49                      END IF;
50                  WHEN E_NC =>
51                      IF ENTRADAS = "1011000" THEN E1 <= E_ND;
52                      ELSIF KS = '1' AND AB = '0' THEN E1 <= FALHA;
53                      ELSIF KS = '1' AND AB = '1' THEN E1 <= S_TI;
54                      ELSIF S(2) = '1' AND S(1) = '1' AND S(0) = '1' THEN E1 <= INICIO;
55                      ELSE E1 <= E_NC;
56                      END IF;
57                  WHEN E_ND =>
58                      IF ENTRADAS = "1100000" THEN E1 <= E_NE;
59                      ELSIF KS = '1' AND AB = '0' THEN E1 <= FALHA;
60                      ELSIF KS = '1' AND AB = '1' THEN E1 <= S_TI;
61                      ELSIF S(2) = '1' AND S(1) = '1' AND S(0) = '1' THEN E1 <= INICIO;
62                      ELSE E1 <= E_ND;
63                      END IF;
64                  WHEN E_NE =>
65                      IF ENTRADAS = "1101000" THEN E1 <= E_NF;
66                      ELSIF KS = '1' AND AB = '0' THEN E1 <= FALHA;
67                      ELSIF KS = '1' AND AB = '1' THEN E1 <= S_TI;
68                      ELSIF S(2) = '1' AND S(1) = '1' AND S(0) = '1' THEN E1 <= INICIO;
69                      ELSE E1 <= E_NE;
70                      END IF;
71                  WHEN E_NF =>
72                      IF ENTRADAS = "1110000" THEN E1 <= E_AB;
73                      ELSIF KS = '1' AND AB = '0' THEN E1 <= FALHA;
74                      ELSIF KS = '1' AND AB = '1' THEN E1 <= S_TI;
75                      ELSIF S(2) = '1' AND S(1) = '1' AND S(0) = '1' THEN E1 <= INICIO;
76                      ELSE E1 <= E_NF;
77                      END IF;
78                  WHEN E_AB =>
79                      IF ENTRADAS = "1000010" THEN E1 <= INICIO;
80                      ELSIF KS = '1' AND AB = '0' AND S(0) = '1' THEN E1 <= INICIO;
81                      ELSIF S(2) = '1' AND S(1) = '1' AND S(0) = '1' THEN E1 <= INICIO;
82                      ELSE E1 <= E_AB;
83                      END IF;
84                  WHEN FALHA =>
```

```
                    IF KS = '1' AND AB = '1' THEN E1 <= S_TI;
                    ELSIF S(2) = '1' AND S(1) = '1' AND S(0) = '1' THEN E1 <= INICIO;
                    ELSE E1 <= FALHA;
                    END IF;
                WHEN S_TI =>
                    IF S(2) = '1' AND S(1) = '1' AND S(0) = '1' THEN E1 <= INICIO;
                    ELSE E1 <= ERRO;
                    END IF;
                WHEN ERRO =>
                    IF AB = '0' AND BL = '0' THEN E1 <= E_NA;
                    ELSIF AB = '0' AND BL = '1' THEN E1 <= BLOQUEIO;
                    ELSIF S(2) = '1' AND S(1) = '1' AND S(0) = '1' THEN E1 <= INICIO;
                    ELSE E1 <= ERRO;
                    END IF;
                WHEN BLOQUEIO =>
                    IF S(2) = '1' AND S(1) = '1' AND S(0) = '1' THEN E1 <= INICIO;
                    ELSE E1 <= BLOQUEIO;
                    END IF;
                WHEN OTHERS => E1 <= BLOQUEIO;
                END CASE;
            END IF;
        END PROCESS;

    DETECTOR_SAIDAS : PROCESS (E1)
    BEGIN
      CASE E1 IS
          WHEN INICIO       => SAIDAS <= "100010"; FL <= '0';
          WHEN E_NA         => SAIDAS <= "010000";
          WHEN E_NB         => SAIDAS <= "010000";
          WHEN E_NC         => SAIDAS <= "010000";
          WHEN E_ND         => SAIDAS <= "010000";
          WHEN E_NE         => SAIDAS <= "010000";
          WHEN E_NF         => SAIDAS <= "010000";
          WHEN E_AB         => SAIDAS <= "010000";
          WHEN FALHA        => SAIDAS <= "010000"; FL <= '1';
          WHEN S_TI         => SAIDAS <= "001001";
          WHEN ERRO         => SAIDAS <= "001000";
          WHEN BLOQUEIO     => SAIDAS <= "000100";
      END CASE;
    END PROCESS;
    A<= SAIDAS(5);
    F<= SAIDAS(4);
    E<= SAIDAS(3);
    B<= SAIDAS(2);
    RT<= SAIDAS(1);
    TI<= SAIDAS(0);
END BEHAVIOR;
```

```
-- Quartus Prime VHDL FEI DIGITAL REPORT: FSM CONTADOR

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY CONTADOR IS
    PORT (  INI, CK, RT, TI       : IN STD_LOGIC;
            BL                    : OUT STD_LOGIC;
            N                     : OUT STD_LOGIC_VECTOR(2 DOWNTO 0)
            );
END CONTADOR;

ARCHITECTURE BEHAVIOR OF CONTADOR IS
    TYPE ME_2 IS (T3, T2, T1, BLOQUEIO);
    SIGNAL E2         : ME_2;
    SIGNAL ENTRADAS   : STD_LOGIC;
    SIGNAL SAIDAS     : STD_LOGIC_VECTOR(2 DOWNTO 0);

BEGIN
    DETECTOR : PROCESS (CK, TI, RT) -- processo para definição das transições dos estados
    BEGIN
        IF (RT='1') THEN E2 <= T3;              -- estado de reset do sistema
        ELSIF (CK'event and CK='1') THEN -- detecção de borda de CLK
            CASE E2 IS                                  -- sincronização com CLK_EN
                WHEN T3 =>
                    IF TI = '1' THEN E2 <= T2;
                    ELSE E2 <= T3;
                    END IF;
                WHEN T2 =>
                    IF TI = '1' THEN E2 <= T1;
                    ELSE E2 <= T2;
                    END IF;
                WHEN T1 =>
                    IF TI = '1' THEN E2 <= BLOQUEIO;
                    ELSE E2 <= T1;
                    END IF;
                WHEN BLOQUEIO => E2 <= BLOQUEIO;
                WHEN OTHERS => E2 <= BLOQUEIO;
                END CASE;
            END IF;
        END PROCESS;
```

```
42  ┣
43  ┃   DETECTOR_SAIDAS : PROCESS (E2)
44  ┃   BEGIN
45  ┃     CASE E2 IS
46  ┃         WHEN T3 => BL <= '0'; N <= "011";
47  ┃         WHEN T2 => BL <= '0'; N <= "010";
48  ┃         WHEN T1 => BL <= '0'; N <= "001";
49  ┃         WHEN BLOQUEIO => BL <= '1'; N <= "000";
50  ┣     END CASE;
51  ┗   END PROCESS;
```

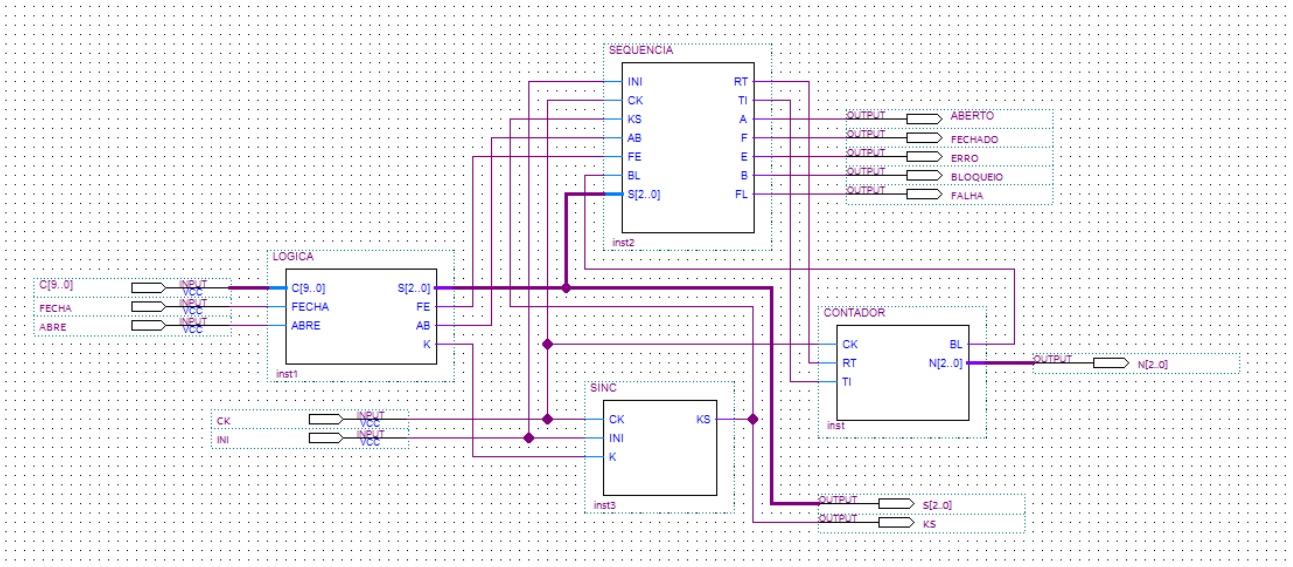
# 6. DECODIFICADORES PARA DISPLAY (VHDL)

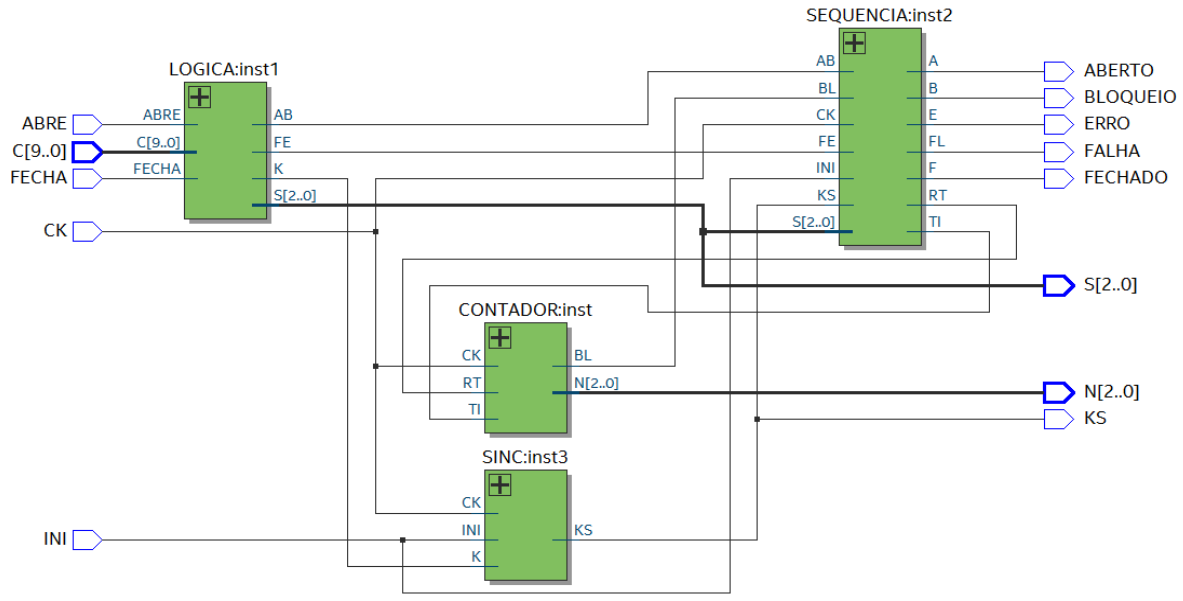
```
1   ⊞--*****************************************************************************************
13    LIBRARY IEEE;
14    USE IEEE.STD_LOGIC_1164.ALL;
15
16  ⊟ENTITY DEC_NT IS                  -- declaracao da entidade DEC_NT
17  ┃   PORT
18  ┃   (  N     : IN STD_LOGIC_VECTOR(3 downto 0);     -- vetor de entrada das chaves
19  ┣      HEX0  : OUT STD_LOGIC_VECTOR(6 downto 0));   -- vetor de saida D[6..0]
20  ┃ END DEC_NT;
21  ┗
22  ⊟ARCHITECTURE SELETOR OF DEC_NT IS
23  ⊟BEGIN
24  ┃   WITH N SELECT
25  ┃   HEX0     <= "0110000" WHEN "0011",  -- display 3
26  ┃               "0100100" WHEN "0010",  -- display 2
27  ┃               "1111001" WHEN "0001",  -- display 1
28  ┃               "1000000" WHEN OTHERS;  -- display 0
29  ┗END SELETOR;
30
```

```
1   ⊞--*****************************************************************************************
13    LIBRARY IEEE;
14    USE IEEE.STD_LOGIC_1164.ALL;
15
16  ⊟ENTITY DEC_STATE IS                  -- declaracao da entidade DEC_STATE
17  ┃   PORT
18  ┃   (  A     : IN STD_LOGIC;                          -- entrada A cofre aberto
19  ┃      F     : IN STD_LOGIC;                          -- entrada F cofre fechado
20  ┃      E     : IN STD_LOGIC;                          -- entrada E sequencia errada
21  ┃      B     : IN STD_LOGIC;                          -- entrada B cofre fechado
22  ┣      HEX1  : OUT STD_LOGIC_VECTOR(6 downto 0));    -- vetor de saida HEX1[6..0]
23  ┃ END DEC_STATE;
24  ┗
25  ⊟ARCHITECTURE SELETOR OF DEC_STATE IS
26  ┃   SIGNAL ENTRADAS: STD_LOGIC_VECTOR(3 downto 0);
27  ┗
28  ⊟BEGIN
29  ┃
30  ┃   ENTRADAS <= A & F & E & B;                        -- concatenação dos sinais de entrada
31  ┃
32  ┃   WITH ENTRADAS SELECT
33  ┃      HEX1  <= "0001000" WHEN "1000",  -- display A
34  ┃               "0001110" WHEN "0100",  -- display F
35  ┃               "0000110" WHEN "0010",  -- display E
36  ┃               "0000011" WHEN "0001",  -- display b
37  ┃               "1111111" WHEN OTHERS;  -- display Apagado
38  ┗END SELETOR;
39
```

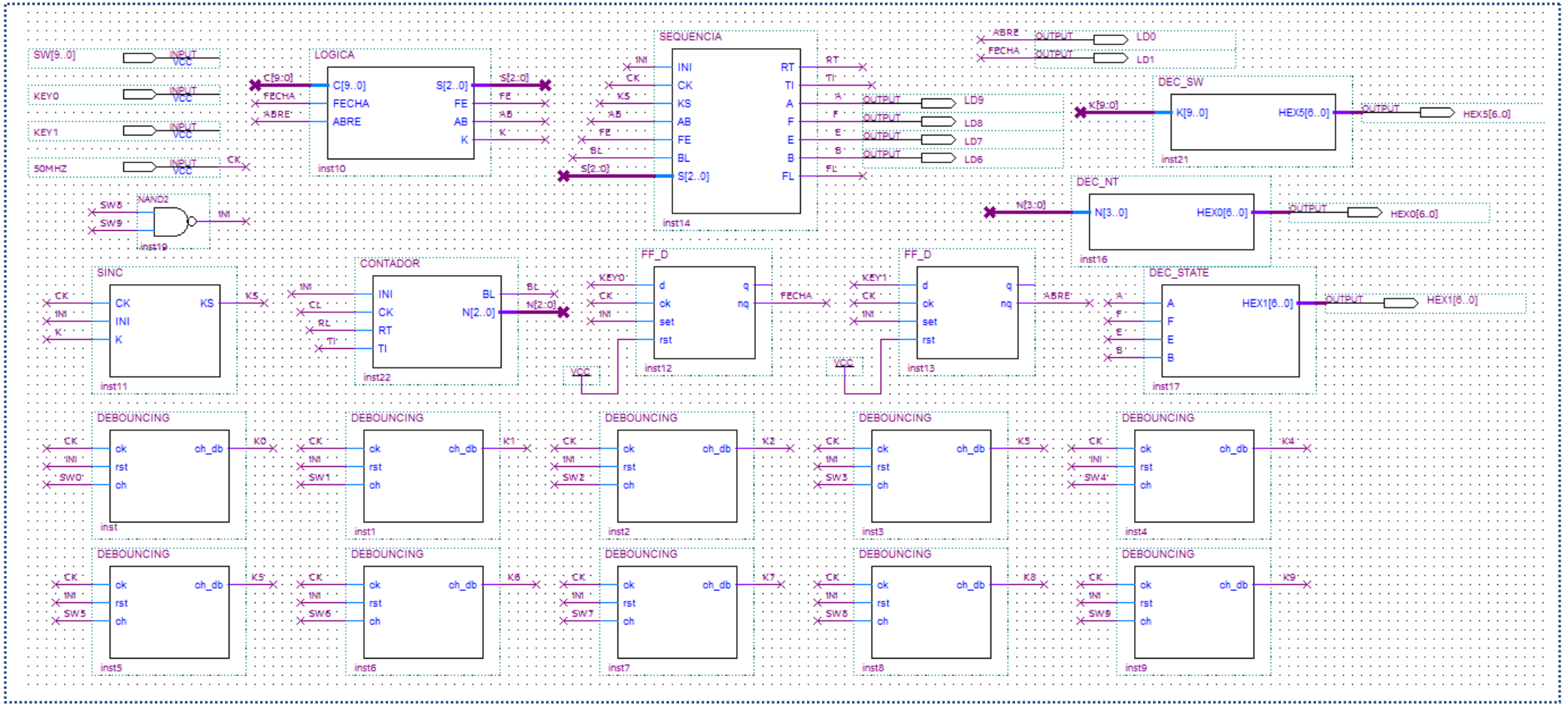
```
1   ⊞--*****************************************************************************************
13    LIBRARY IEEE;
14    USE IEEE.STD_LOGIC_1164.ALL;
15
16  ⊟ENTITY DEC_SW IS                  -- declaracao da entidade DEC_SW
17  ┃   PORT
18  ┃   (  K     : IN STD_LOGIC_VECTOR(9 downto 0);     -- vetor de entrada das chaves (K)
19  ┣      HEX5  : OUT STD_LOGIC_VECTOR(6 downto 0));   -- vetor de saida D[6..0]
20  ┃ END DEC_sw;
21  ┗
22  ⊟ARCHITECTURE SELETOR OF DEC_sw IS
23  ⊟BEGIN
24  ┃   WITH K SELECT
25  ┃      HEX5  <= "1000000" WHEN "0000000001",  -- display 0
26  ┃               "1111001" WHEN "0000000010",  -- display 1
27  ┃               "0100100" WHEN "0000000100",  -- display 2
28  ┃               "0110000" WHEN "0000001000",  -- display 3
29  ┃               "0011001" WHEN "0000010000",  -- display 4
30  ┃               "0010010" WHEN "0000100000",  -- display 5
31  ┃               "0000010" WHEN "0001000000",  -- display 6
32  ┃               "1111000" WHEN "0010000000",  -- display 7
33  ┃               "0000000" WHEN "0100000000",  -- display 8
34  ┃               "0010000" WHEN "1000000000",  -- display 9
35  ┃               "1111111" WHEN OTHERS;  -- display Apagado
36  ┗END SELETOR;
37
```

# 7. DIAGRAMA ESQUEMÁTICO E VISÃO RTL DOS COMPONENTES UTILIZADOS NA SIMULAÇÃO FUNCIONAL DO COFRE DIGITAL

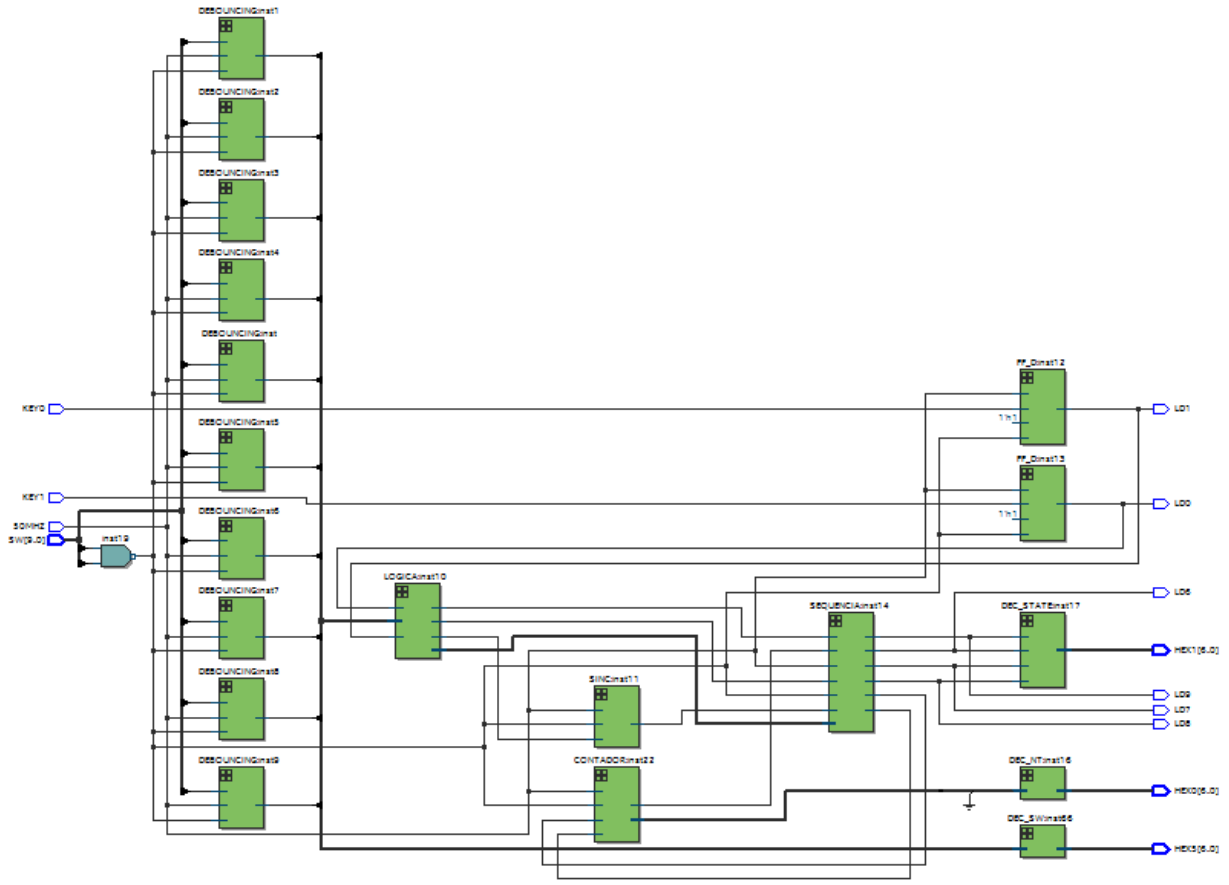


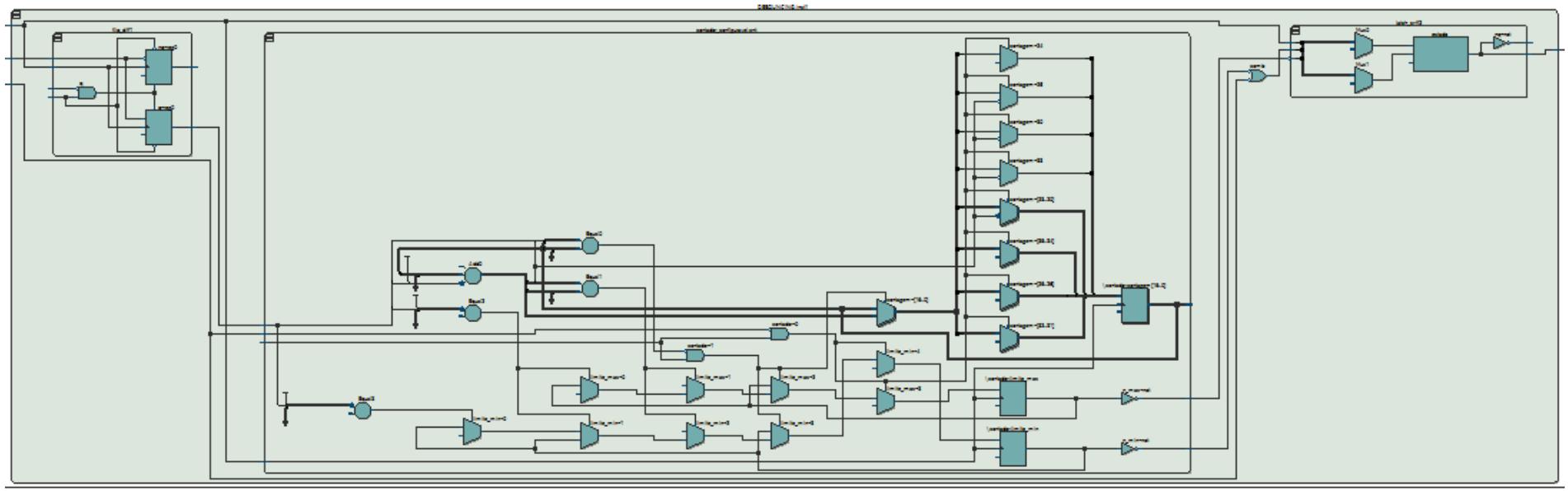
# 9. DIAGRAMA DE BLOCOS DO PROJETO COMPLETO



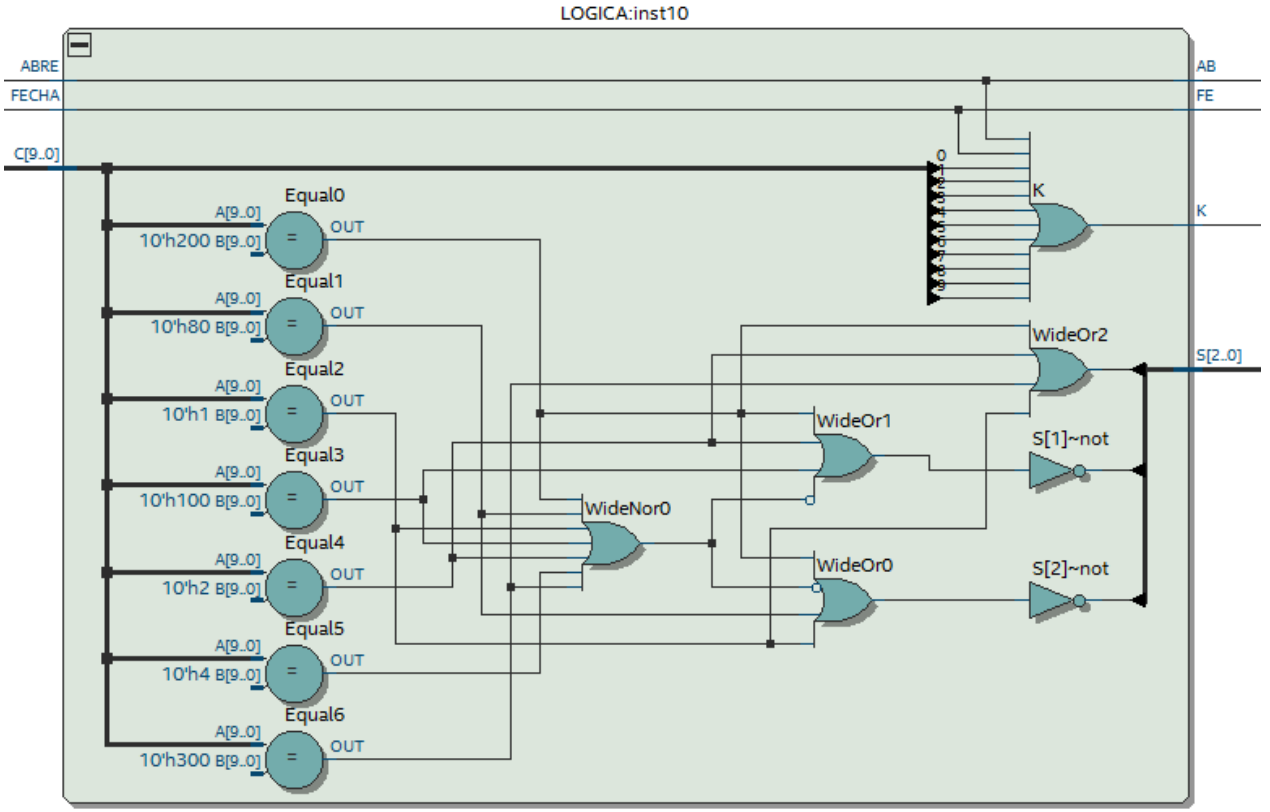
# 10.DIAGRAMA RTL DO PROJETO COMPLETO



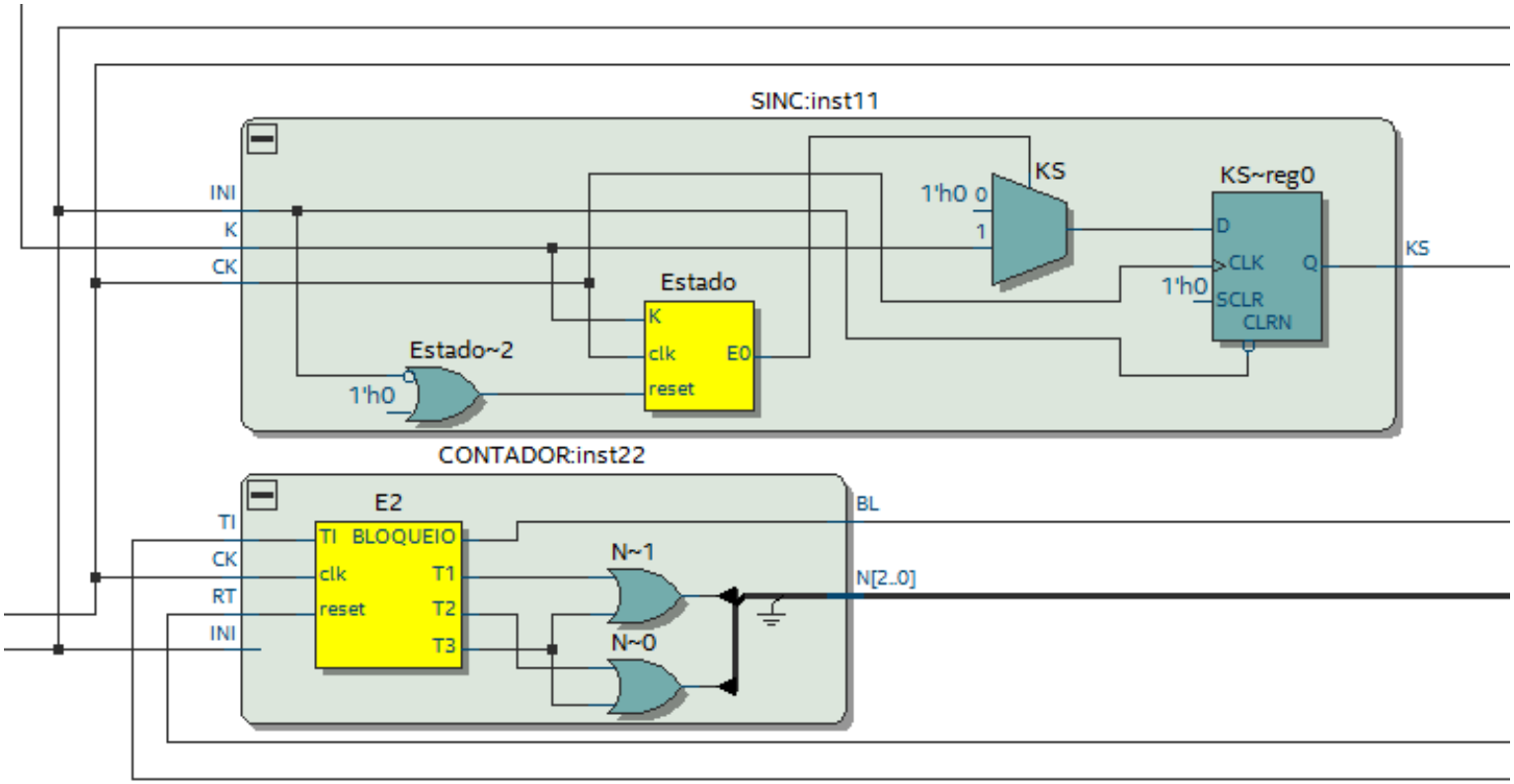
# 11.DIAGRAMA RTL DO CIRCUITO DEBOUNCING



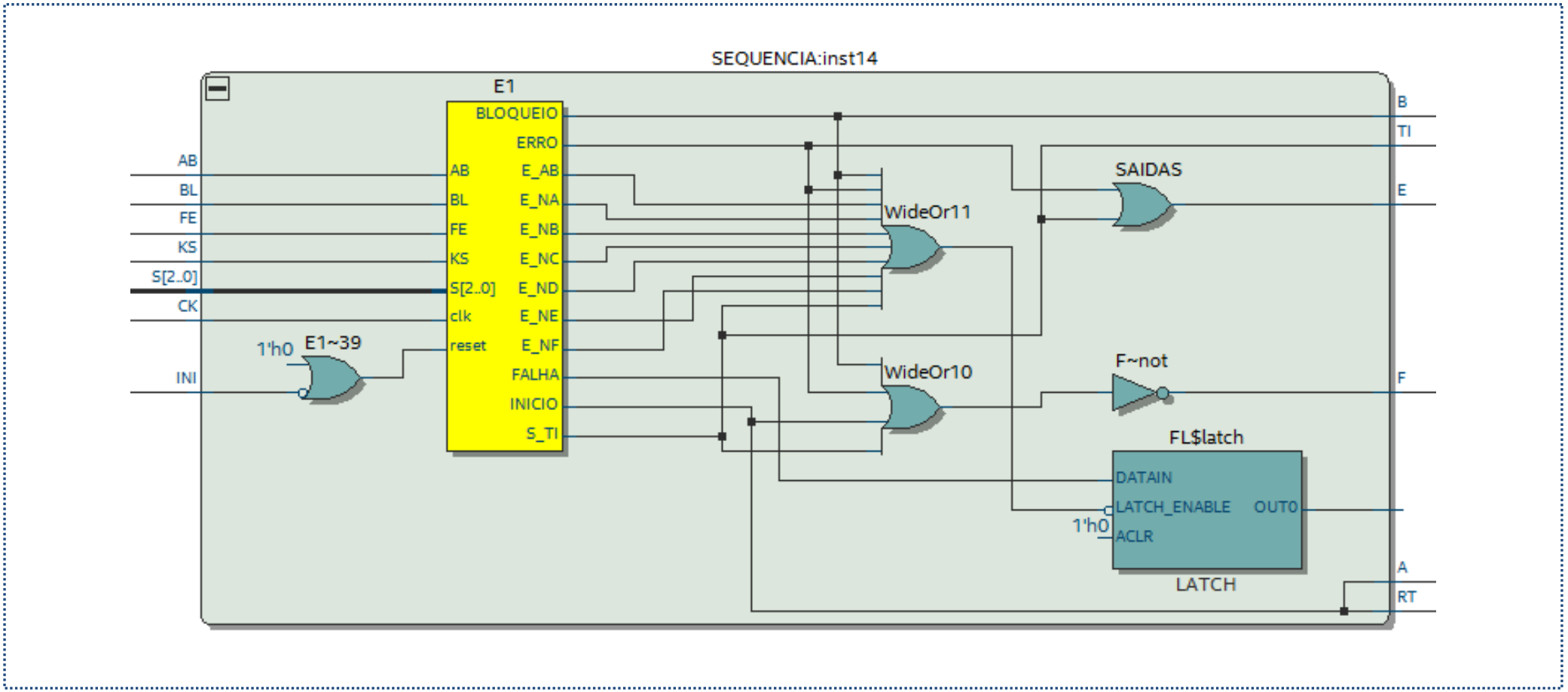
# 12.DIAGRAMA RTL DA LÓGICA DE CODIFICAÇÃO



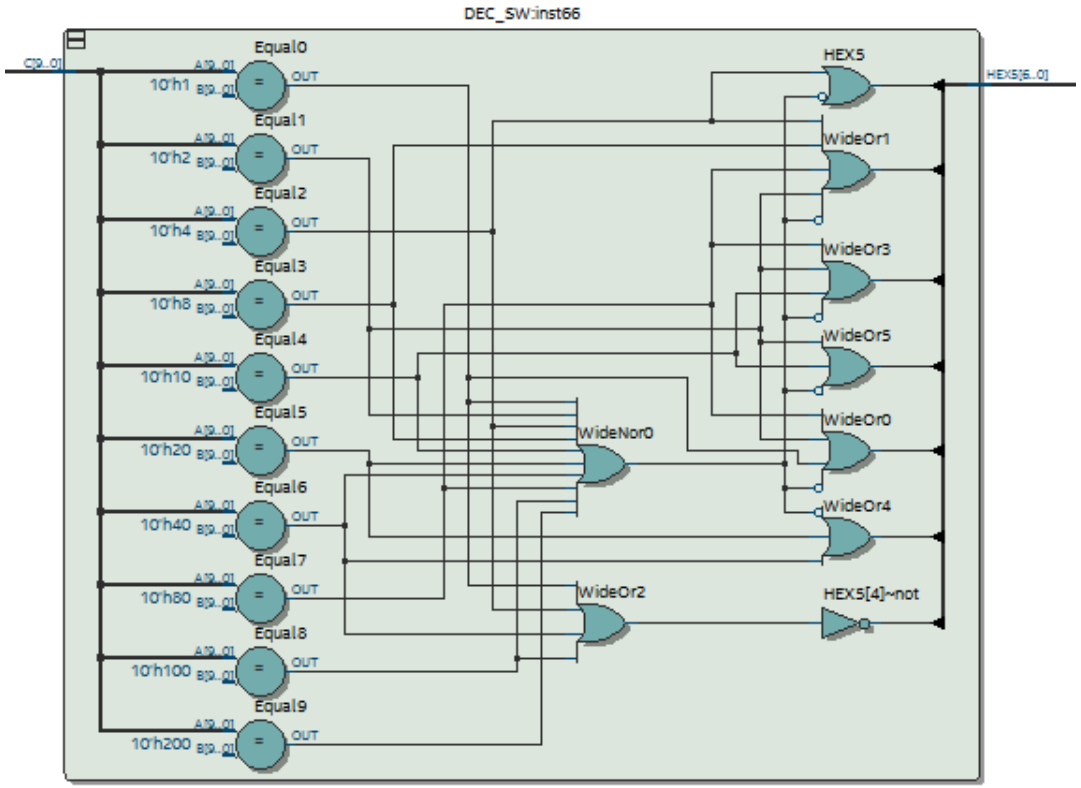
## 13.DIAGRAMA RTL DA LÓGICA DO CIRCUITO SICRONIZADOR E CONTADOR DE TENTATIVAS



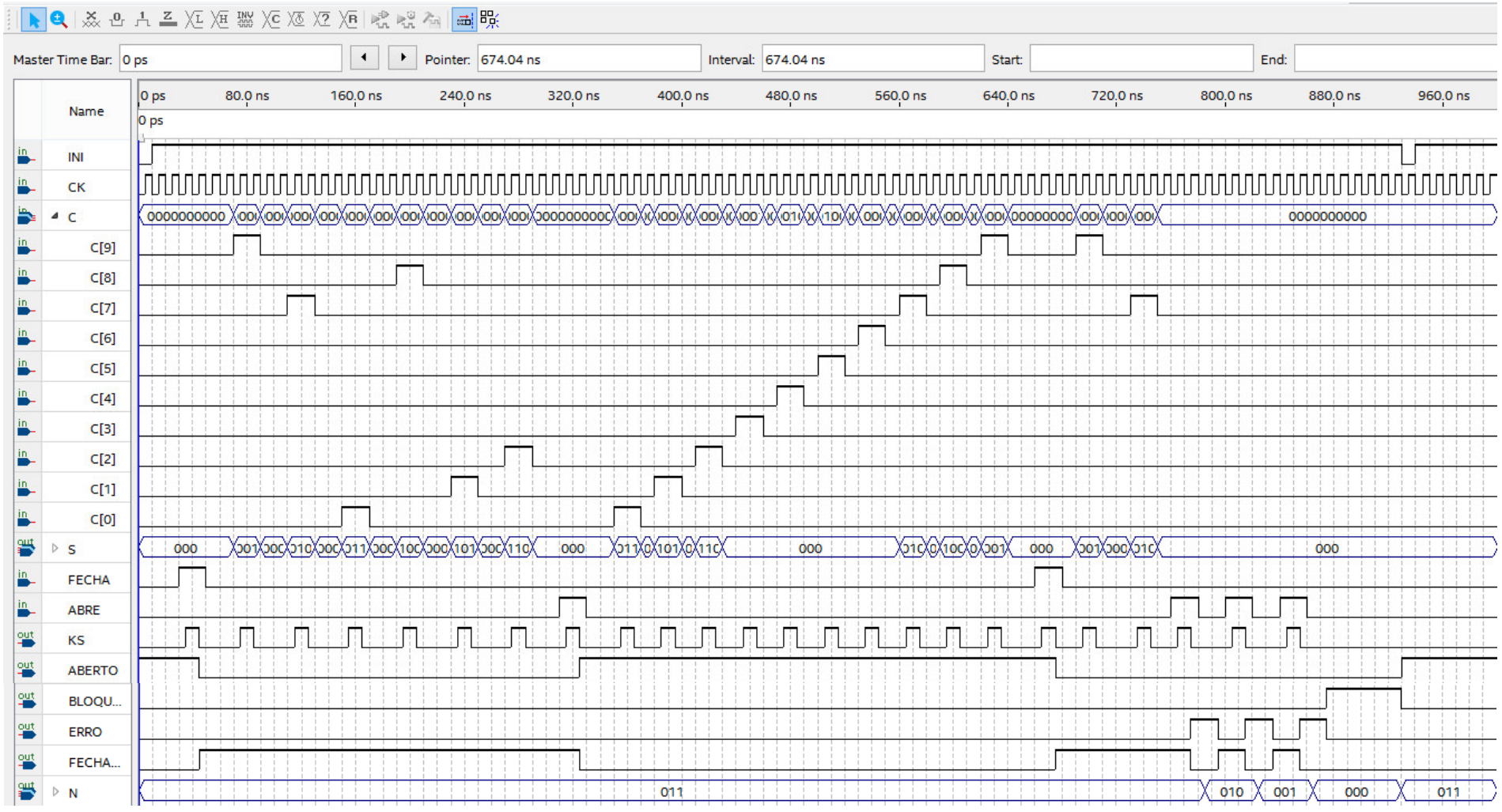
# 14.DIAGRAMA RTL DA LÓGICA DO CIRCUITO DA SEQUÊNCIA DE SEGURANÇA (SENHA)



# 15.DIAGRAMA RTL DA LÓGICA DO CIRCUITO DECODIFICADOR PARA AS CHAVES



# 16.SIMULAÇÃO FUNCIONAL DA MÁQUINA DE ESTADOS



# 17.CONCLUSÃO

O objetivo geral deste projeto 1 do Laboratório de Sistemas Digitais II foi alcançado, no qual desenvolvemos um sistema digital de controle para comandar a abertura de um cofre digital de segurança. Para tanto, foi iniciado o desenvolvimento pela uma máquina de estado finita (FSM) e a lógica necessária para comando do sinal de abertura do cofre em um FPGA, utilizando a linguagem de descrição VHDL implementada no software Quartus Prime Lite Edition 16.1 e o método de projeto de um bit por estado (utilizando equações de estado). Bits armazenados significam que o circuito tem memória, o que é também conhecido como estado, resultando os chamados circuitos sequenciais. A senha para desbloqueio do cofre digital a ser desenvolvido é baseada no número de matrícula (R.A) descrito no cabeçalho deste documento.

Esse projeto de implementação também alcançou objetivos exercitar a metodologia de desenvolvimento de projetos de engenharia apoiada em computador (CAE) onde todo o desenvolvimento do software embarcado e modelo final será feito no software Quartus Prime Lite Edition 16.1 onde será feita a simulação funcional e interface final do sistema de controle digital totalmente implementada utilizando o FPGA da família MAX 10 (modelo: 10M50DAF484C7G) existente na placa de desenvolvimento Altera DE10-Lite. Esse processo envolve a compreensão do problema, seu planejamento, desenvolvimento da solução lógica, integração dos subsistemas, implementação no ambiente computacional, simulação, testes, depuração do projeto, implementação física e registro dos resultados.

Através dos itens que correspondem à visão RTL dos principais circuitos, foi comprovada a complexidade da implementação e como a metodologia de um bit por estado é facilitadora para a visão de implementação do projeto. A conformidade do Sistema Digital para o Cofre de Segurança foi garantida através da simulação via University UWF (Wave Form) realizada no software.