Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

---

Université de Carthage

Ecole Polytechnique de Tunisie

Projet Module Système Embarqués:

Sujet:
Commande d'une Serrure Numérique

Enseignant: Mr. Slim Ben Saoud

Réalisé par:
Abdessalem Achour | Galaaoui Salma

Contents

List of Figures

List of Listings

List of Tables

1 Introduction Générale

1.1 Cadre du projet

L'objectif de ce projet est de consolider les acquis du module Systèmes Embarqués à travers la conception d'une solution de commande d'une serrure numérique. Dans ce projet, on est demandé d'élabore la machines à état finis (Finite State Machine) de cette application, de coder cette machine en langage VHDL et de coder le Test Bench correspondant sur l'ISE de Xilinx (ancétre de Vivado Design Suite). On travaillera avec Vivado 2019.2.

Figure 1: Vivado Design Suite

1.2 Définition

La serrure est un système qui permet d'ouvrir ou de fermer une porte. Elle marche par l'actionnement d'une clé, d'une carte ou d'un code.

Figure 2: Serrure Mécanique

La serrure électrique est fabriquée en acier renforcé et ne peut pas être percée ou coupée. Il s'agit ici d'une grande amélioration par rapport aux serrures traditionnelles.

Les serrures électriques sont utiles car elles assurent une très bonne sécurité et elles sont plus faciles à utiliser par rapport aux serrures classiques. Elles offrent également des fonctionnalités nécessaires à une sécurité absolue.

Figure 3: Serrure Mécanique

L'ajour de la connectivité à la serrure permet de fonctionner doublement comme serrure et système d'alarme en ayant la possibilité de notifier le propriétaire du bien sécurisé par la serrure lors d'une intrusion, cette connectivité peut s'effectuer par le biais de plusieurs protocoles de communication (Bluetooth, WiFi, GSM..)

2 Cahier des Charges

Dans ce projet on se place le Use Case d'une serrure électrique dont la gâchette est commandée par un signal émis par un circuit FPGA qu'on se propose de concevoir. La serrure munie d'une clavier alphanumérique ne s'ouvre uniquement que si le code à 4 caractères composé correspond à la combinaison correcte.

Ci dessous le diagramme de la Machine à états Finis (FSM) de cette serrure:

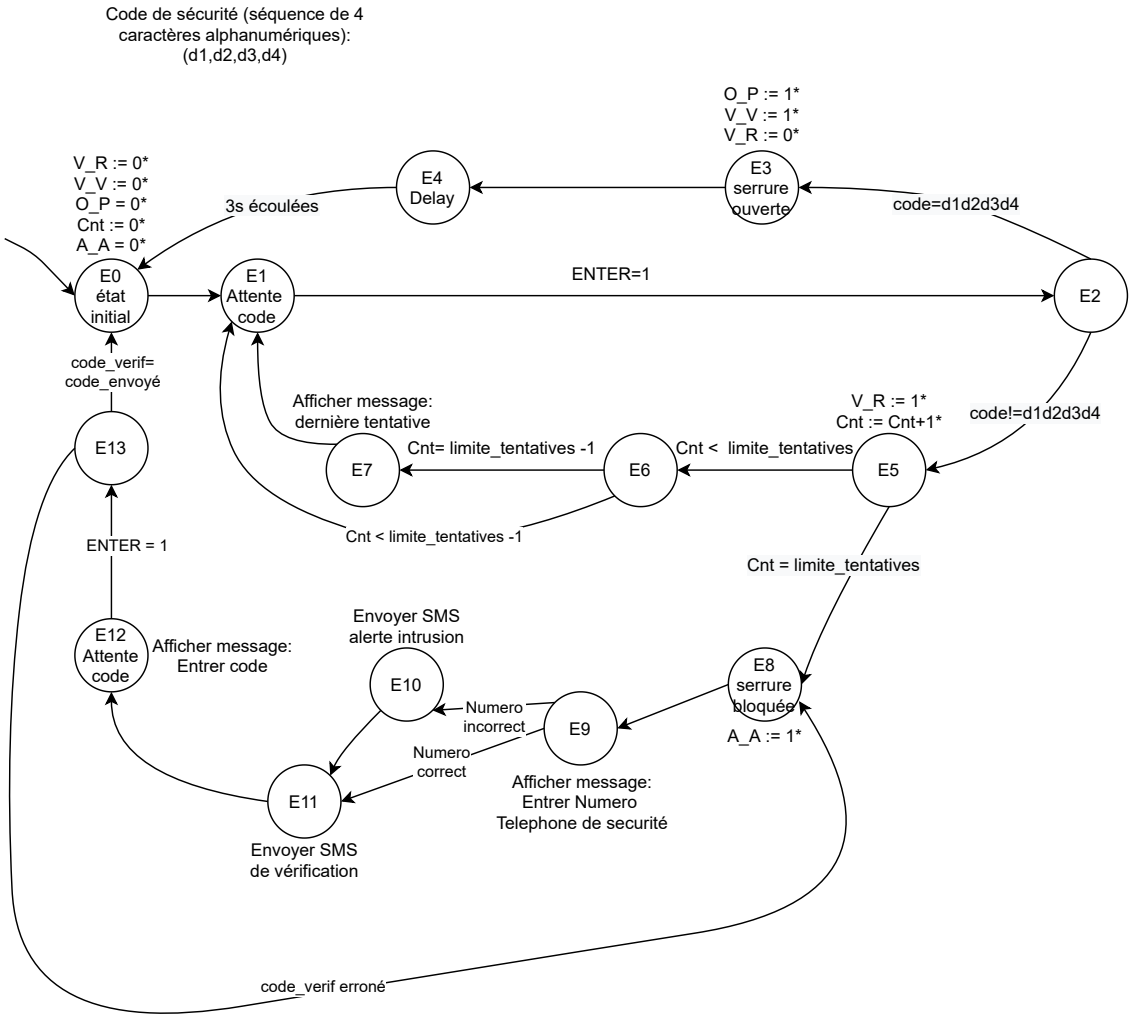

Figure 4: Finite State Machine

Table 1: Nomenclature de la FSM

Abbréviation	Type	Désignation
code	entrée	Variable qui contient le code saisi par l'utilisateur
code_vérif	entrée	Code saisi en cas de blocage de la serrure
O_P	signal de sortie	Signal ordre d'ouverture de la porte
Cnt	variable	Compteur qui indique le nombre de tentatives effectuées
V_R	signal de sortie	Signal d'activation du voyant rouge
V_V	signal de sortie	Signal d'activation du voyant vert
A_A	signal de sortie	Signal pour activer l'alarme
code_envoyé	sortie	Code envoyé par SMS au le numéro de téléphone en cas de blocage de la serrure
limites_tentatives	variable	Indique le nombre maximal de tentatives de saisie avant le blocage de la serrure

On remarque que pour le cas de cette application les signaux de sortie ne dépendent que de l'état actuel, il s'agit donc d'une **machine de Moore**.

La serrure est connectée au réseau cellulaire à travers un module GSM. Elle se trouve initialement dans un état de repos E0 où tout les voyants sont désactivés, lors de l'appui d'un bouton du clavier par l'utilisateur elle passe de l'état E0 à l'état E1 en attendant que l'utilisateur finisse la saisie du code et appuye sur le bouton "Enter" pour passer à l'état E2 ou se trouve un aiguillage selon le code saisi;

- Si le code saisi est conforme à la séquence correcte la serrure passe à l'état E3 ensuite à l'état E4 où elle reste ouverte pendant un intervalle de temps (maintient du signal d'ouverture de la porte).
- Si le code saisi est erroné, l'utilisateur à un nombre limité de tentatives pour reprendre le code (E5 -> E6 -> E7), avec l'affichage d'un message de rappel "Dernière Tentative!" dans le cas ou le nombre de tentatives restantes est à 1. Cependant si l'utilisateur épuise toutes ses tentatives possibles sans saisir la combinaison correcte la serrure se bloque (E8) et l'utilisateur doit à présent saisir le numéro de téléphone de sécurité (saisi par le propriétaire de la serrure lors de sa configuration initiale) pour pouvoir recevoir un code de déblocage par SMS. Si le numéro saisi ne correspond pas au numéro de sécurité correct, un SMS "Alerte Intrusion" est alors envoyé au propriétaire (E10) et la serrure reste bloquée jusqu'à ce que celui-ci vienne la débloquer en saisissant le code de déblocage reçu par SMS (E12 -> E13).

3 Solution Conçue

3.1 Approche et raisonnement

Puisque le but de ce projet est d'apprendre à coder une FSM et le Test Bench correspondant, non pas de synthétiser le circuit et l'implémenter réellement sur un FPGA, on se permet dans la limite de cette application de simuler l'interaction entre l'utilisateur et la serrure à travers des fichiers texte, ceci est possible grâce à l'utilisation de la bibliothèque *textio* dans le Test Bench:

- La lecture du code saisi se fait depuis un fichier "in.txt".
- La lecture du numéro téléphone de sécurité se fait depuis un fichier "num.txt".
- La lecture du code de vérification envoyé par SMS se fait depuis un fichier "codeVin.txt".
- L'envoi du code de vérification se fait par l'écriture dans un ficher "codeVout.txt".
- L'affichage des messages d'alerte et de rappel se fait par l'écriture dans un fichier "log.txt".
- L'affichage du verdict (code correct ou incorrect) se fait par l'écriture dans un fichier "out.txt".

3.2 Description en VHDL de la FSM

3.2.1 Code Source

Lien du code source sur GitHub.

Les signaux de sorties à générer par la machine étant décrit clairement dans le schéma de la FSM on procède à la description de la machine en séparant dans deux process différents, l'évolution des états selon les entrée (transitions) et la génération de sorties.

- **Process: Évolution États**

```
process(etat,CLK) is


    variable code_correct : string(4 downto 1) := "AF10";
    variable numero_correct : string(8 downto 1) := "22222222";


    begin

        if(rising_edge(CLK)) then

            case etat is
                when E0 =>
                    etat <= E1;
                when E1 =>
                    if(enter = '1') then
                    etat <= E2;
                    end if;
                when E2 =>
                    if(code = code_correct) then etat <= E3;
                    else etat <= E5;
                    end if;
                when E3 =>
                    etat <= E4;
                when E4 =>
                    --hold lock open for 5 clk periods
                    if (delay < 4) then
                    delay <= delay + 1;
                    else
                    delay <= 0;
                    etat <= E0;
                    end if;
                when E5 =>
                    if(cnt < limite_tentatives) then etat <= E6;
                    else etat <= E8;
                    end if;
                when E6 =>
                    if(cnt = limite_tentatives-1) then etat <= E7;
                    else etat <= E1;
```

```
40                      end if;
41                  when E7 =>
42                      etat <= E1;
43                  when E8 =>
44                      etat <= E9;
45                  when E9 =>
46                      if(numero_tel = numero_correct) then etat <= E11;
47                      else etat <= E10;
48                      end if;
49                  when E10 =>
50                      etat <= E11;
51                  when E11 =>
52                      etat <= E12;
53                  when E12 =>
54                      if(enter = '1') then
55                      etat <= E13;
56                      end if;
57                  when E13 =>
58                      if(code_debloc = integer'image(code_debloc_verif))
59                      then etat <= E0;
60                      else etat <= E8;
61                      end if;
62
63              end case;
64          end if;
65      end process;
```

Listing 1: Process description de l'évolution des états

- **Process: Génération des Sorties**

```
process(etat)
    begin

        cur_state <= type_etat'pos(etat);
        --mapped state id
        Ntries <= limite_tentatives - cnt;
        --remaining tries to be mapped to TB

        case etat is

            when E0 =>
                VV <= '0';
                VR <= '0';
                cnt <= 0;
                OP <= '0';
                AA <= '0';

            when E3 =>
                VV <= '1';
                VR <= '0';
                OP <= '1';

            when E5 =>
                VR <= '1';
                cnt <= cnt+1;

            when E8 =>
                AA <= '1';

            when E11 =>
                code_debloc_verif <= code_verification;

            when others => null;

        end case;
    end process;
```

Listing 2: Process génération de sorties

3.2.2 Test Bench

```
process(current_state,clock) is

file buff_in : text open read_mode is "/home/salmag98/Vivado/Commande_Serrure/in.txt";
file buff_out : text open write_mode is "/home/salmag98/Vivado/Commande_Serrure/out.txt";
file buff_num : text open read_mode is "/home/salmag98/Vivado/Commande_Serrure/num.txt";
file buff_codeout : text open write_mode is "/home/salmag98/Vivado/Commande_Serrure/codeVout.txt";
file buff_codein : text open read_mode is "/home/salmag98/Vivado/Commande_Serrure/codeVin.txt";
file log : text open write_mode is "/home/salmag98/Vivado/Commande_Serrure/log.txt";
variable line_in, line_out, line_num, line_codeout, line_codein , line_log :   line;
--variable v_OLINE     :   line;
variable codein : string(4 downto 1);
variable debloc : string(4 downto 1);
variable num : string(8 downto 1);
variable v_TIME : time := 0ns;

begin

    Ent <= tern((current_state = 1) or (current_state = 12), '1', '0');
    --Ent is HIGH only when state is E1 or E11 and LOW otherwise

    if(rising_edge(clock)) then

        case current_state is

            when 1 =>
            readline(buff_in,line_in);
            read(line_in, codein);
            code_in <= codein;

            when 3 =>
            --entered code is correct
            write(line_out, time'image(now - v_TIME) & string'(" ") & codein & string'(" matching code"));
            writeline(buff_out, line_out);

            when 5 =>
            --entered code does not match
            write(line_out, time'image(now - v_TIME) & string'(" ") & codein & string'(" uncorrect code"));
            writeline(buff_out, line_out);

            when 7 =>
```

```
41              --Number of tries almost reached.
42              write(line_log, time'image(now - v_TIME) & string'(" Dernière
                ↪  tentative!!"));
43              writeline(log, line_log);
44
45              when 9 =>
46              write(line_log, time'image(now - v_TIME) & string'(" Entrer numéro
                ↪  de telephone de securité:"));
47              writeline(log, line_log);
48              readline(buff_num,line_num);
49              read(line_num, num);
50              num_tel <= num;
51
52              when 10 =>
53              write(line_log, time'image(now - v_TIME) & string'(" Alerte
                ↪  Intrusion!!"));
54              writeline(log, line_log);
55
56              when 11 =>
57              --send verification code
58              write(line_codeout, time'image(now - v_TIME) & string'(" ") &
                ↪  integer'image(code_debloc_v));
59              writeline(buff_codeout, line_codeout);
60
61              when 12 =>
62              --prompt user for verification code sent on cellphone
63              write(line_log, time'image(now - v_TIME) & string'(" Entrer
                ↪  code:"));
64              writeline(log, line_log);
65              -- "read" entered code
66              readline(buff_codein,line_codein);
67              read(line_codein, debloc);
68              code_d <= debloc;
69
70              when others =>
71              null;
72
73          end case;
74      end if;
75
76      --file_close(buff_in);
77      --file_close(buff_out);
78
79  end process;
```

Listing 3: Process Stimulus

Il faut bien entendu changer les chemins des répertoire des fichiers entrée et sortie (ligne 3 à 8) selon l'emplacement du projet.

En faisant la correspondance (mapping) des signaux d'entrée sortie et des variable identi-

fiants les états, le testbench devient interactif et cadencer par la même horloge que le process d'évolution des états dans le code source.

```
architecture Behavioral of Commande_Serrure_TB is
    constant ClockFrequency : integer := 100e6; -- 100 MHz
    constant ClockPeriod    : time    := 1000 ms / ClockFrequency;
    component SourceCode
        port ( code, numero_tel : in string;
            code_debloc : in string;
            code_debloc_verif : buffer integer;
            cur_state, Ntries : out integer;
            CLK, enter : in std_logic;
            OP, VV, VR, AA: out std_logic);
    end component;

--function switching outputs depending on whether the input condition is
 ↪  verified
--used for the value of Ent signal
function tern(cond : boolean; res_true, res_false : std_logic) return
 ↪  std_logic is
begin
  if cond then
    return res_true;
  else
    return res_false;
  end if;
end function;


    signal code_in : string(4 downto 1);
    signal code_d : string(4 downto 1);
    signal code_debloc_v : integer;
    signal num_tel : string(8 downto 1);
    signal current_state : integer;
    signal Tries_Remaining : integer;
    signal clock : std_logic := '0';
    signal Ent: std_logic := '0';
    signal Ouverture : std_logic;
    signal VoyantV : std_logic;
    signal VoyantR : std_logic;
    signal Alarme : std_logic;

begin
    UUT: SourceCode port map (code => code_in, numero_tel => num_tel,
     ↪  code_debloc => code_d, code_debloc_verif => code_debloc_v, cur_state
     ↪  => current_state, Ntries => Tries_Remaining, CLK => clock, enter =>
     ↪  Ent, OP => Ouverture, VV => VoyantV, VR => VoyantR, AA => Alarme);
```

Listing 4: Port mapping - Test Bench

3.2.3 Graphique de Simulation

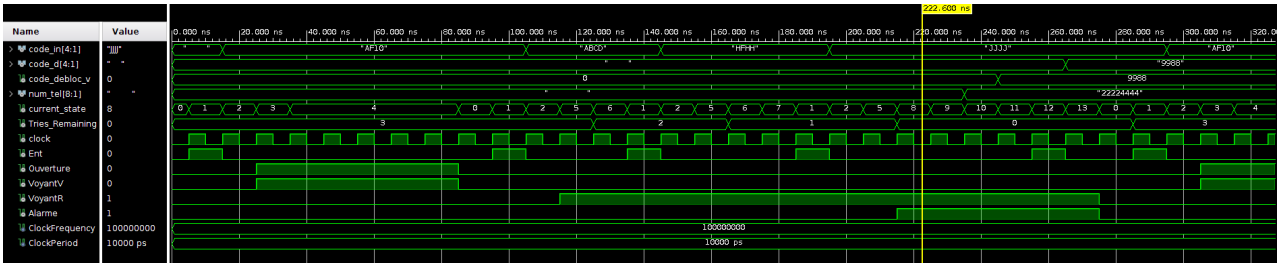

Figure 5: Simulation

- **Interprétations du Graphique:**

  On remarque bien que la FSM reste dans l'état E4 (i.e current_state = 4) pour une durée de 4 périodes d'horloge (maintien du signal d'ouverture).

  Le verdit étant émis au niveau de l'état E3 (resp. E5) si le code est conforme (resp. non conforme), on peut voir qu'a la sortie de l'état E3 dans le graphique de simulation le temps de simulation est à *35.000 ns*, on remarque effectivement que la première ligne du fichier *out.txt* générer par la simulation contient le même horodatage:

```
35000 ps AF10 matching code
125000 ps ABCD uncorrect code
165000 ps HFHH uncorrect code
215000 ps JJJJ uncorrect code
315000 ps AF10 matching code
405000 ps HHFH uncorrect code
445000 ps AF10 matching code
535000 ps AJD0 uncorrect code
575000 ps 1234 uncorrect code
625000 ps DKLF uncorrect code
845000 ps KDKK uncorrect code
885000 ps AF10 matching code
975000 ps DDJE uncorrect code
```

Listing 5: out.txt

De même pour la sortie de l'état E7 à *185.000 ns* la première ligne du fichier log.txt contient le même horodatage:

```
185000 ps Dernière tentative!!
235000 ps Entrer numéro de telephone de securité:
245000 ps Alerte Intrusion!!
265000 ps Entrer code:
595000 ps Dernière tentative!!
645000 ps Entrer numéro de telephone de securité:
655000 ps Alerte Intrusion!!
675000 ps Entrer code:
705000 ps Entrer numéro de telephone de securité:
715000 ps Alerte Intrusion!!
735000 ps Entrer code:
765000 ps Entrer numéro de telephone de securité:
775000 ps Alerte Intrusion!!
795000 ps Entrer code:
```

Listing 6: out.txt

4 Conclusion

Ce projet nous a permis de consolider nos acquis en description des Machines à États Finis mais aussi d'apprendre à manipuler un code VHDL et d'utiliser des bibliothèque selon le besoin pour concevoir une solution adéquate et élégante à un problème et de simuler son fonctionnement.

Comme perspectives pour ce projet, dans le cas ou on souhaite implémenter cette solution sur une carte Xilinx on peut envisager dans l'étape à venir de remplacer la lecture/écriture dans les fichiers texte par la connexion de la carte à un module GSM ainsi qu'un clavier alphanumérique muni d'un écran LCD pour tester la solution réellement.

Pour l'instant puisqu'il sagit d'un Test Bench le fonctionnement le plus proche d'une interaction Utilisateur-Serrure à travers des SMSs et des affichages serait de communiquer à travers les fichiers texte.