المعهد الوطني للبريد والمواصلات
ⵜⴰⵙⴷⴰⵡⵜ ⵜⴰⵏⴰⵎⵓⵔⵜ ⵏ ⵜⵙⵉⵡⵍⵉⵏ ⴷ ⵜⵙⵙⵉⵡⵍⵉⵏ
Institut National des Postes et Télécommunications

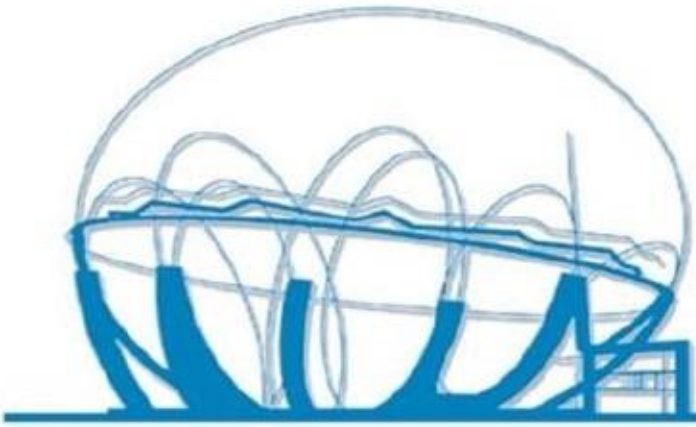
agence nationale de réglementation des télécommunications
الوكالة الوطنية لتقنين المواصلات
ⵜⴰⵖⵍⵉⵙⵜ ⵜⴰⵏⴰⵎⵓⵔⵜ ⵏ ⵜⵙⵙⵉⵡⵍⵉⵏ

Année universitaire: *2021/2022*

RAPPOT PROJET DE DEVELOPPMENT

Titre : Ascenseur commandé à base d'une carte FPGA

Filière : Systèmes Embraqués et Services Numériques



Réalise par :

MOHAMED OUAZZI

KHALID EL-KHOUR

Encadrants :

Oussama EL ISSATI

Assia EL HADBI

Institut National des Postes et Télécommunications

Soutenu le 09 juin 2022 à 12 : 40

Remerciement

Nous remercions d'abord notre coordinateur de filière monsieur Oussama EL ISSATI pour sa disponibilité. Nous le remercions de nous avoir encadré, orienté, conseillé et de nous avoir fait confiance.

Nous remercions également notre encadrant et notre professeur Mme Assia EL HADBI pour sa disponibilité, son encadrement, sa confiance et les conseils qu'elle nous a généreusement prodigués. Nous la remercions aussi de son suivi permanent de notre travail, ses remarques et suggestions sans lesquelles ce mémoire n'aurait pas lieu.

Nous remercions également l'ensemble des membres du jury pour avoir examiné notre travail.

Nous tenons enfin à remercier tous ceux qui ont contribué d'une façon ou d'une autre à la réalisation de ce travail.

Objectif

L'objectif initial de ce projet était de contrôler un ascenseur à quatre étages par une carte FPGA.

Table des matières

INTRODUCTION GENERALE

Avec la montée de l'urbanisation de notre société, l'ascenseur est devenu désormais un système indispensable pour répondre aux exigences modernes de notre vie en matière d'autonomie, de mobilité, d'accessibilité et de rapidité. Il est ainsi un élément essentiel des immeubles résidentiels, des bureaux, des aéroports, des centres de soins, des bâtiments publics, etc…. Il contribue ainsi à gagner du temps, faciliter les déplacements, le transport et les courses.

Les ascenseurs électriques, comme nous les avons aujourd'hui, ont commencé à être utilisé dans le XIXe siècle grâce à l'invention de l'Allemand von Siemens en 1880. Cet inventeur a ajouté le moteur électrique dans les ascenseurs. À son tour, la cabine comportant en dessous du moteur par l'intermédiaire des pignons des engrenages rotatifs. En 1887, un autre ascenseur avec moteur électrique a été inventé, mais cette fois travaillé par un tambour enroulé dans le

Câble de levage. Ascenseurs avec des boulons d'engrenages reliant le moteur à tambour utilisé dans les années suivantes.

Les ascenseurs sont de plus en plus modernisés. En 1949, la figure de l'ascenseur et les interrupteurs de commande automatiques installés. En outre, plus de mesures de sécurité sont installés dans les portes. En 1957, les portes ont cessé d'être manuel et est devenu automatique, comme vous pouvez le voir aujourd'hui.

Toutes ces innovations ont pu créer l'ascenseur qui sert aujourd'hui et se trouve dans la plupart des immeubles de grande hauteur dans le monde entier.

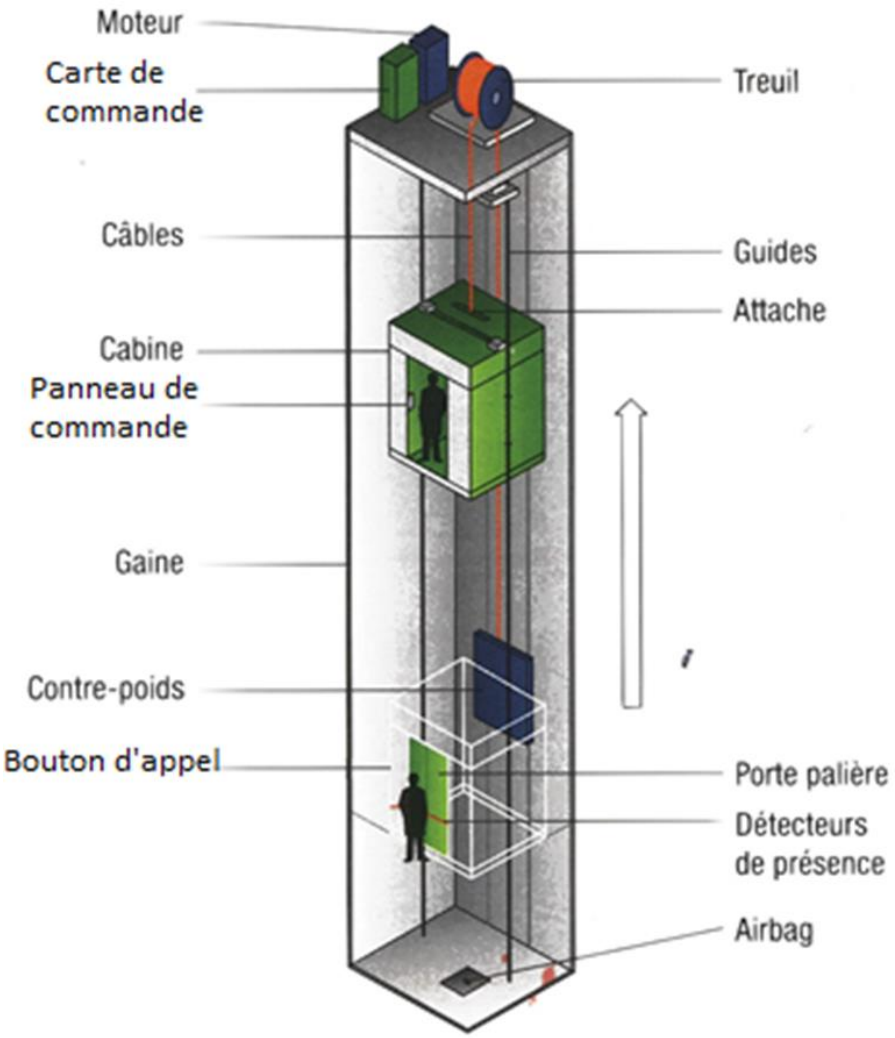
Chapitre 1

Chapitre 1 : Présentation générale sur l'ascenseur

Ce premier chapitre sera consacré à une présentation sur l'ascenseur. Les différents types et modes de fonctionnent, aussi nous avons montré les avantages et les inconvenants de chaque type, ensuite nous avons abordé les différentes parties de l'ascenseur de traction qui sera notre intérêt à ce projet, et à la fin, nous avons fait une présentation sur la maquette (prototype).

1. Définition

Un ascenseur est une installation de levage permanente desservant deux ou plusieurs niveaux, comprenant un espace clos, ou cabine, dont les dimensions et le mode de construction permettent l'accès aux personnes et qui se déplace entre des guides rigides verticaux.



2. Les types d'ascenseurs :

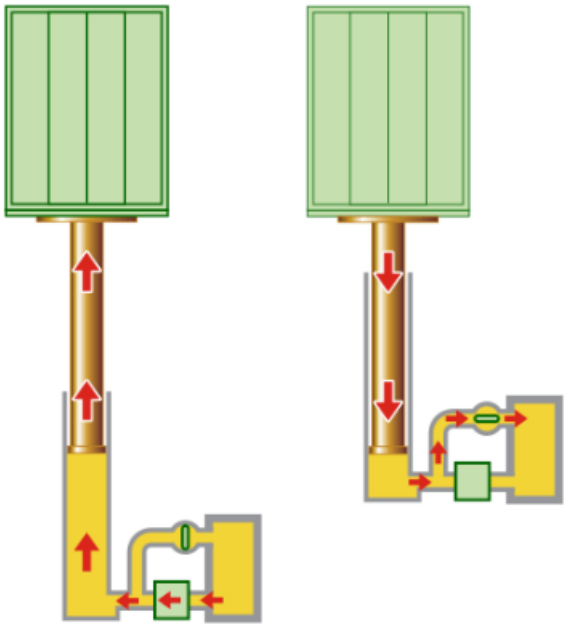
On distingue essentiellement deux types de familles d'ascenseur :

- les ascenseurs hydrauliques.
- les ascenseurs à traction à câble,

En règle générale, ces deux types utilisent l'énergie électrique pour déplacer les cabines verticalement (moteur électrique continu ou alternatif).

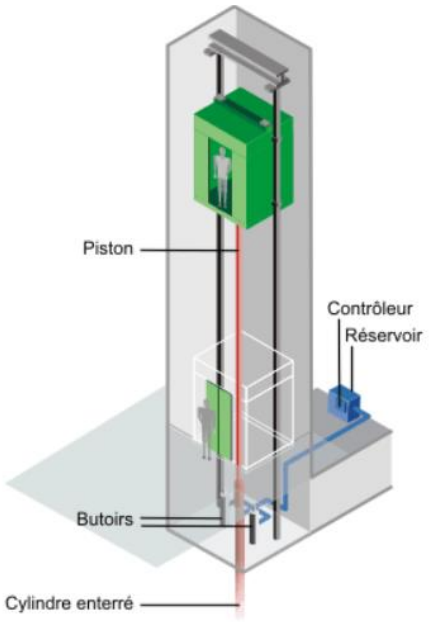
2.1 Les ascenseurs hydrauliques.

Comme toute machine hydraulique la pompe met sous pression l'huile qui pousse le piston hors du cylindre vers le haut. Lorsque la commande de descente est programmée, le by-pass (vanne) de la pompe permet de laisser sortie l'huile du cylindre vers le réservoir.

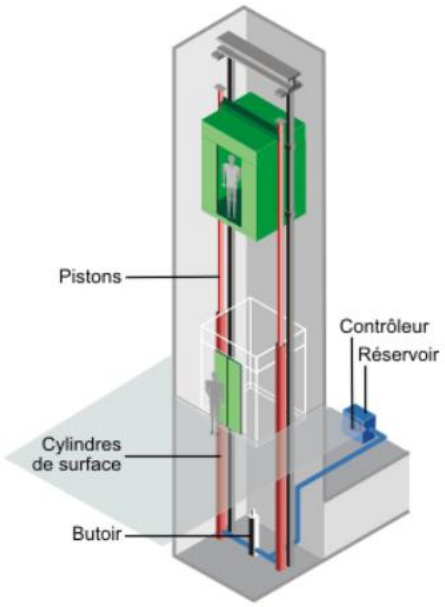


Plusieurs modèles existent sur le marché. On citera les ascenseurs hydrauliques :

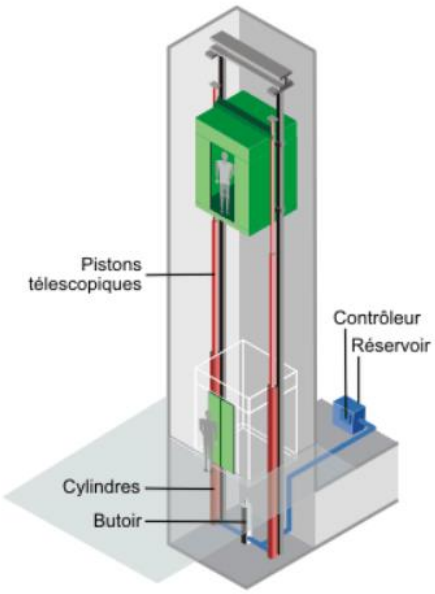
- à cylindre de surface,
- à cylindre enterré,
- Télescopiques à cylindre de surface.



À cylindre enterré.



À cylindre de surface.



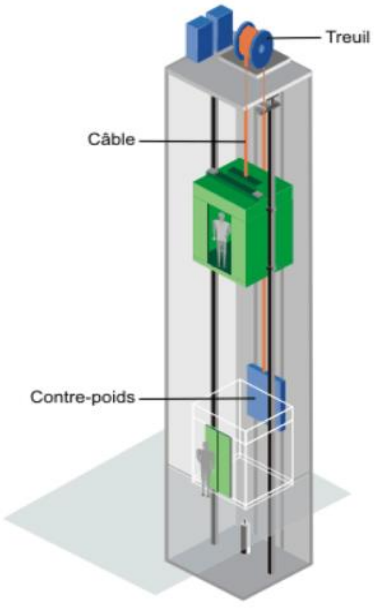
À cylindre de surface télescopique.

2.2 Les ascenseurs à traction à câble

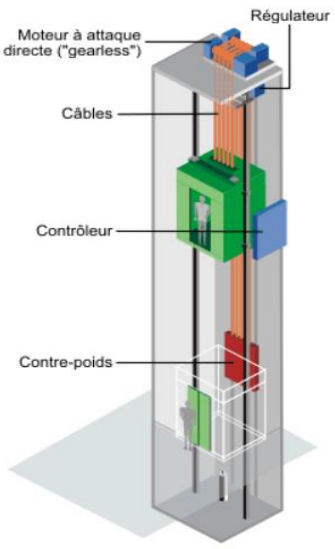
Les ascenseurs à traction à câbles sont les types d'ascenseurs que l'on rencontre le plus, notamment dans les bâtiments tertiaires.

Ils se différencient entre eux selon le type de motorisation :

- à moteur-treuil à vis sans fin,
- à moteur-treuil planétaire,
- à moteur à attaque directe (couramment appelé "Gearless" ou sans treuil),
- …



Ascenseur à moteur-treuil.



Ascenseur à moteur à attaque directe.

3. Principe de fonctionnement

Un ascenseur à contrepoids se compose d'une cabine qui se déplace dans un couloir vertical nommé gaine et qui est guidée par des rails afin d'éviter une collision avec le contre poids. Un frein situé dans la machinerie du moteur permet de stopper la cabine à l'étage demandé. Le déplacement en translation de la cabine est permis par un système de transmission de mouvement. Des câbles, actionnés par un treuil permettent de mettre en mouvement la cabine et le contrepoids. Le moteur du treuil permet la mise en mouvement. Le contrepoids est une charge lourde qui sert à équilibrer la charge de la cabine et à diminuer l'énergie à fournir par le moteur. Lorsque la cabine monte, le contrepoids descend. Le système comprend aussi des organes de commande pour enregistrer les appels des usagers et optimiser les déplacements de la cabine afin de répondre le plus rapidement possible aux différents appels. Enfin, l'ascenseur est équipé d'organes assurant la sécurité des passagers. Des freins d'urgence ou parachutes sont placés de chaque côté de la gaine et se déclenchent en cas de rupture du câble tracteur pour éviter la chute de la cabine. Ils sont déclenchés par un limiteur de vitesse lorsque la vitesse de la cabine est supérieure à la vitesse de déplacement normale (de 2 à 9 km/h selon les ascenseurs). Les parachutes bloquent alors de façon brutale la cabine sur les guides.

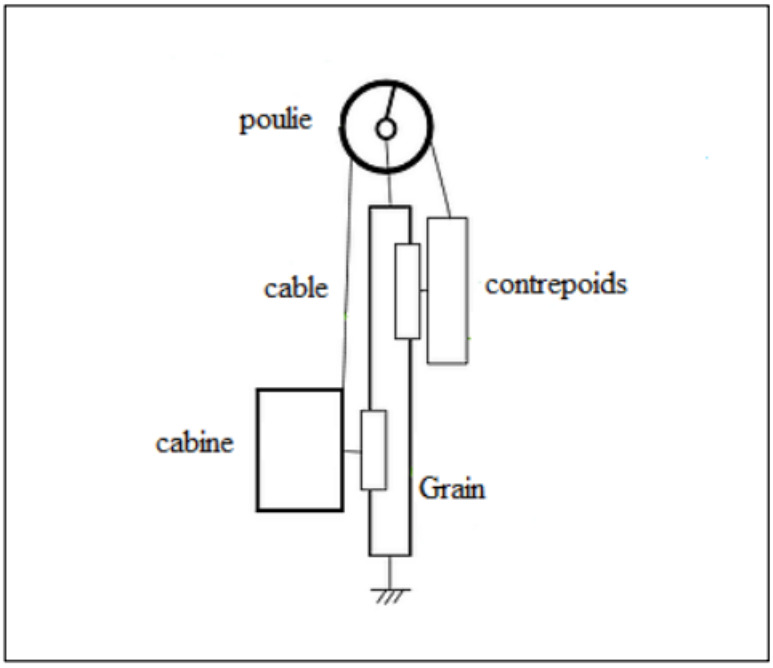


Figure 1-7:Principe de fonctionnement d'un ascenseur à traction

Le contrepoids est un peu plus lourd que la cabine : il sert à contrebalancer le poids de la cabine. Celle-ci est suspendue par des câbles grâce à des poulies, l'effort du moteur pour élever la cabine est réduit. Poids effectif = contrepoids moins poids de la cabine et de sa charge.

4. Schéma fonctionnelle :

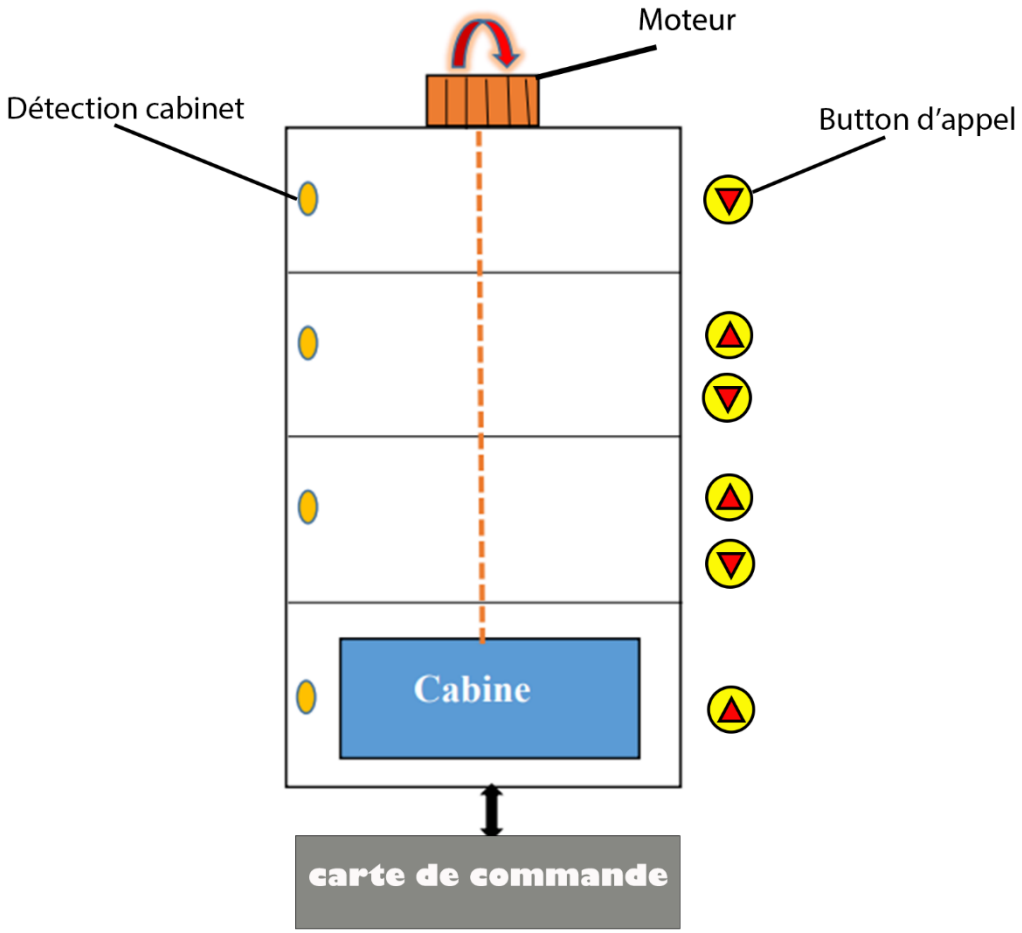


Fig. . Schéma fonctionnel de l'ascenseur

5. Conclusion

Dans ce chapitre nous avons tout d'abord fait une présentation générale des ascenseurs, Leurs différents types et modes de fonctionnent. Les ascenseurs hydrauliques sont plus lents et consomment plus que les ascenseurs électriques, à notre projet on s'intéresse au deuxième type. En effet il y a plusieurs solutions pour commander l'ascenseur parmi ces solutions on utilise la commande par une carte FPGA cyclone 2 qui sera l'objet du chapitre suivant.

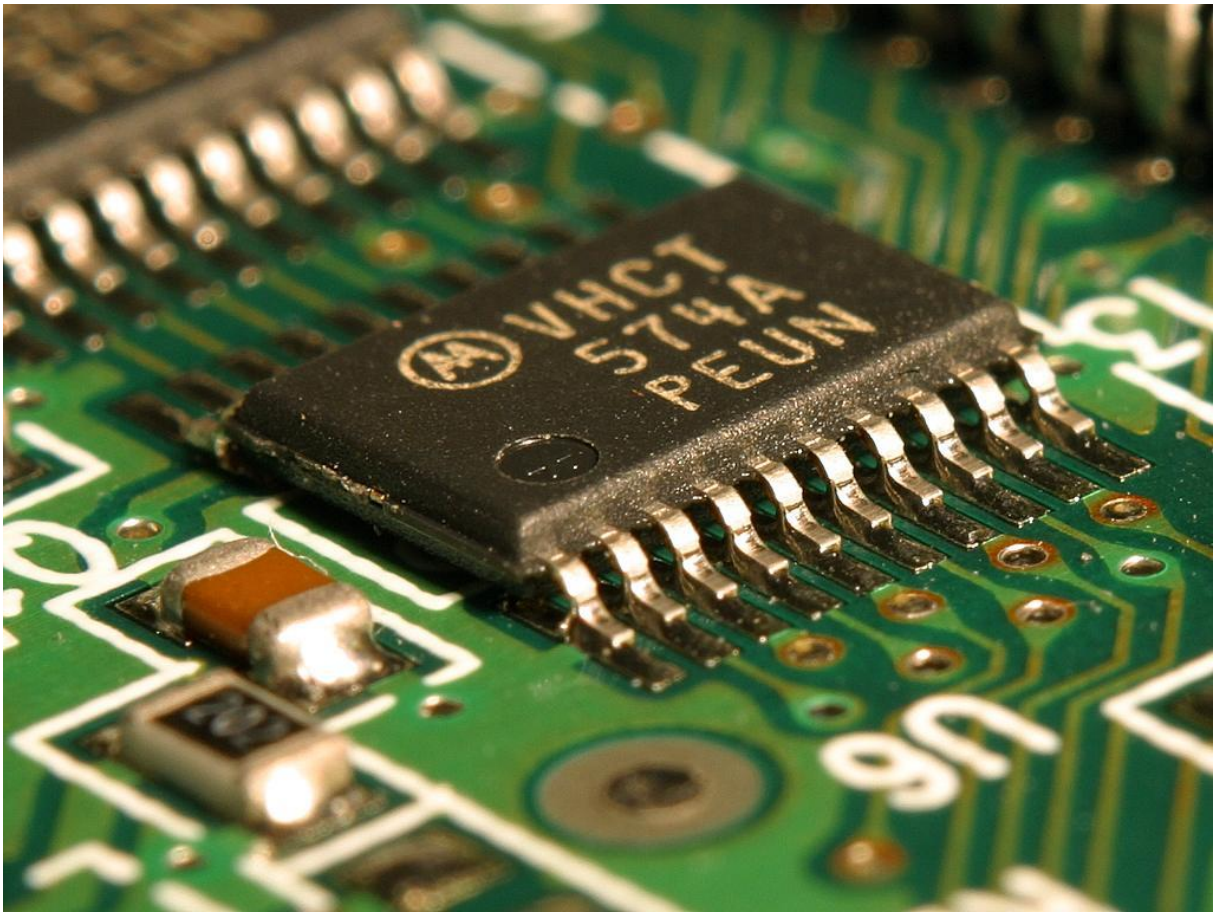

Chapitre 2

Chaptire2 : Etude de la carte programmable FPGA

1. Les circuits intégrés

1.1 Définition :

Le circuit intégré (CI), aussi appelé puce électronique, est un composant électronique, basé sur un semi-conducteur, reproduisant une ou plusieurs fonctions électroniques plus ou moins complexes, intégrant souvent plusieurs types de composants électroniques de base dans un volume réduit (sur une petite plaque), rendant le circuit facile à mettre en œuvre.



1.2 Les différents types de circuits intégrés

On peut classer les circuits intégrés en 3 grandes classes :

- **Les processeurs :**

Un processeur est un circuit intégré permettant d'implémenter n'importe quelle fonction en exécutant de manière séquentielle un code compilé. Le grand avantage du processeur est sa généricité, puisque n'importe quel programme peut y être exécuté. De plus, sa réalisation est très facile.

- **Le Circuit Intégré Spécifique à une Application (ASIC)**

Le Circuit Intégré Spécifique à une Application (ASIC) est une des manières de réaliser un système de calcul matériel. Pour chaque application, un circuit différent est créé, soit en le construisant entièrement, soit en configurant une grille d'éléments préconstruits.

- **Les circuits programmables**

Un circuit programmable, est un assemblage d'opérateurs logiques, combinatoires et de bascules, dans lequel la fonction réalisée n'est pas fixée lors de la fabrication. Il contient potentiellement la possibilité de réaliser toute une classe de fonctions, plus ou moins large suivant son architecture. La programmation du circuit consiste à définir une fonction parmi toutes celles qui sont potentiellement réalisables. Comme dans toute réalisation en logique câblée, une fonction logique est définie par les interconnexions entre des opérateurs combinatoires et des bascules, et par les équations des opérateurs combinatoires. Ce qui est programmable dans un circuit concerne donc les interconnexions et les opérateurs combinatoires. Les bascules sont le plus souvent de simples bascules D, ou des bascules configurables en bascules D ou T. La réalisation d'opérateurs combinatoires utilise des opérateurs génériques.

➔ Par la suite, on va s'intéresser spécialement à ce circuit programmable.

2. Les circuits FPGA

2.1 Définition

FPGA : Field-Programmable Gate Array, réseau de portes programmables

Un FPGA est un circuit intégré composé d'un grand nombre d'éléments logiques programmables reliés entre eux grâce à une matrice de routage elle aussi programmable. Cette structure permet au FPGA d'émuler n'importe quel circuit, à la seule condition que celui-ci ne soit pas trop gros pour ne pas épuiser les ressources logiques et de routage du FPGA.

2.2 Historique

En 1984 la société américaine Xilinx fut le premier inventeur du domaine en lançant le premier circuit FPGA commercial, le XC2000. Ce composant avait une capacité maximum

de 1500 portes logiques. La technologie utilisée était alors une technologie aluminium à 2µm avec 2 niveaux de métallisation. Xilinx sera suivi un peu plus tard, et jamais lâché, par son plus sérieux concurrent Altera qui lança en 1992 la famille de FPGA FLEX 8000 dont la capacité maximum atteignait 15000 portes logiques.

En 2000 et 2001, les deux concurrents Xilinx et Altera ont franchi une nouvelle étape au niveau de la densité d'intégration en proposant respectivement leurs circuits Virtex et Apex-II dont les capacités maximums avoisinaient les 4 millions de portes logiques équivalentes avec de plus l'introduction de larges bancs de mémoires embarquées. Aujourd'hui, les fréquences de fonctionnement de ces circuits sont de l'ordre de quelques centaines de Méga Hertz (ces dernières sont en réalité très dépendantes de l'application). Bien que ces valeurs soient relativement réduites par rapport aux ASICs, elles sont suffisantes pour une très large majorité d'applications actuelles.

À partir des années 2000, les capacités des FPGA ont permis d'offrir aux concepteurs une solution supplémentaire de réalisation pour une majorité d'applications. De plus, les outils de mise en œuvre des FPGA ont évolué, ils permettent la réalisation rapide d'applications complexes.

2.3 Les principaux fabricants

Constructeur	*Part du marché*
Xilinx	35,5%
Altera	32,7%
Lattice	16,1%
Actel	6,7%
Lucent Technologie	4,3%
Autres	4,7%

Tableau 1 : Répartition du marché des FPGA

2.4 Différents types de circuits FPGA

Pour franchir les inconvenients des mémoires, et dans le but de faire un ensemble de technologie complémentaire adaptable suivant l'environnement des cahiers des charges, il existe trois types d'FPGA reprogrammables suivant la technologie de mémorisation pour répondre aux différentes applications.

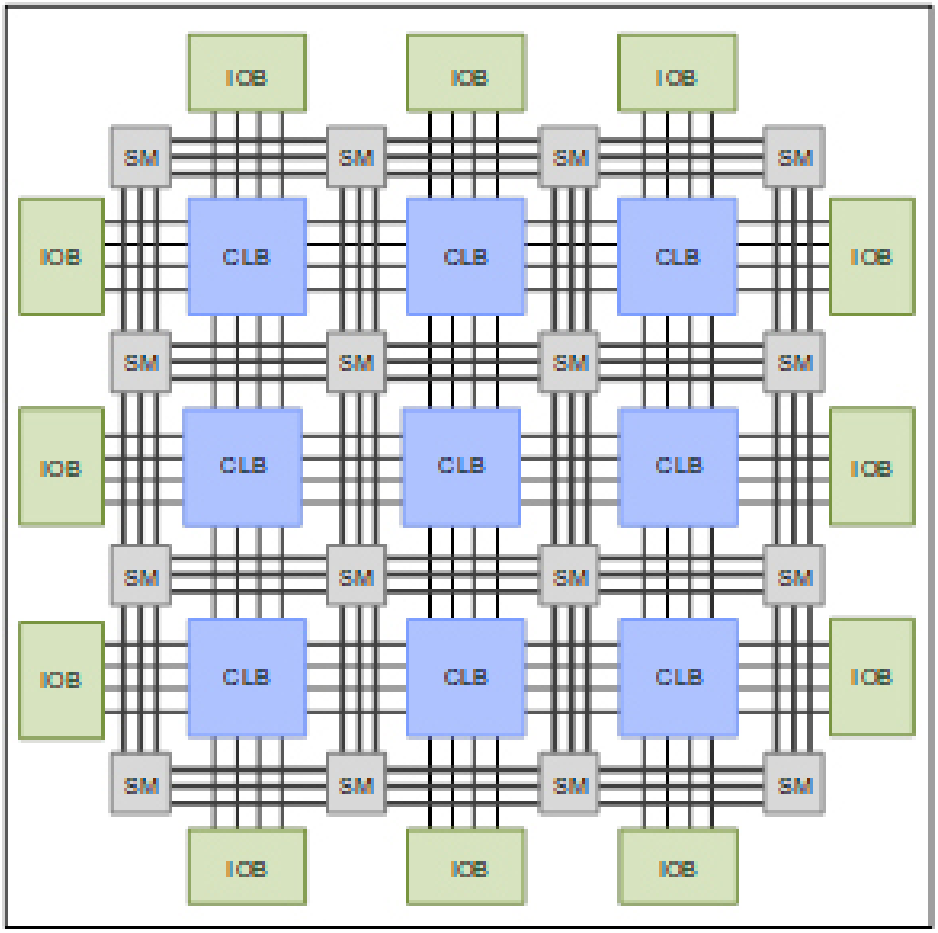
Ces trois principales technologies d'FPGA sont

- Technologie de programmation par EEPROM ou FLASH.
- Technologie de programmation par anti-fusible
- Technologie de programmation par RAM : Cette technologie permet d'avoir une reconfiguration rapide des FPGA. Les connexions sont des ensembles de transistors

commandés. L'inconvénient majeur de cette technologie c'est qu'elle nécessite beaucoup de place et il est nécessaire de sauvegarder le design du FPGA dans une autre mémoire flash.

2.5 Architecture de la carte FPGA

L'architecture interne des FPGA est différente d'un fondeur à un autre et même entre la différente gamme du même constructeur mais rien n'empêche que leurs ressemblances peuvent être rassemblées dans le schéma représentatif de la figure suivante :



En général, un FPGA est composé des blocs principaux électriquement configurables suivants :

- **Les blocs logiques configurables (CLB)**

Les blocs logiques configurables sont les principaux éléments d'un FPGA. Ils peuvent avoir un ou plusieurs générateurs de fonctions réalisées avec des tables de correspondance (LUT look-up tables) qui peuvent mettre en œuvre une logique arbitraire en fonction de leur configuration. , cette LUT peut être vue comme une mémoire (16 bits en général) qui permet de créer n'importe quelle fonction logique combinatoire de 4 variables d'entrées

- **Les ressources d'interconnexion**

Les ressources d'interconnexion au sein d'un FPGA permettent la connexion arbitraire des CLB et des IOB

- **Les blocs d'entrée-sortie (IOB)**

Les blocs d'entrée-sortie permettent l'interconnexion de la logique interne aux ports d'entrées et de sorties du FPGA. Les IOB ont leur propre mémoire de configuration, elle stocke les standards de tension et la direction des ports. Ces blocs sont présents sur toute la périphérie du circuit FPGA. Chaque bloc IOB contrôle une broche du composant et il peut être défini en entrée, en sortie, en signaux bidirectionnels ou être inutilisé.

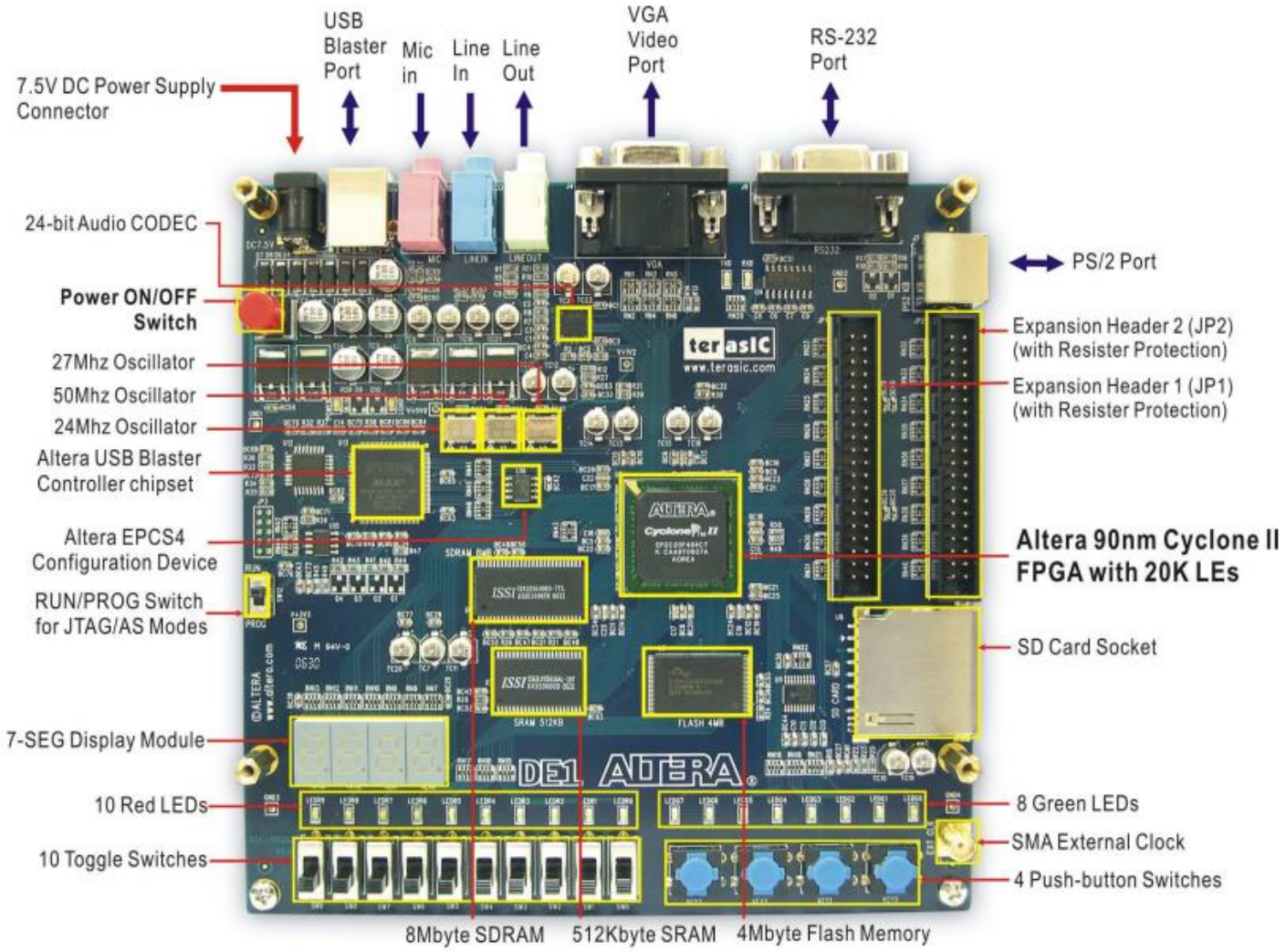
3. Choix de FPGA Altera Cyclone II (starter development kit)

3.1 Pourquoi Altera Cyclone II

Pour améliorer notre connaissances en langage de description matériels VHDL et dans les cartes programmables nous avons choisi de travailler notre projet avec une carte FPGA.

Au début, nous avons travaillé avec une carte Spartan-3 E xilinx et un logiciel d'implémentation ISE Design Suite, mais le kit de développement a un nombre très limité des ports entrée/sortie, ce qui nous amène à change cette carte par une autre de type Altera Cyclone II .

3.2 Disposition et composants



La carte DE1 possède de nombreuses fonctionnalités qui permettent à l'utilisateur de mettre en œuvre une large gamme de circuits conçus, des circuits simples à divers projets multimédias.

Le matériel suivant est fourni sur la carte DE1 :

• Altera Cyclone®

Dispositif FPGA II 2C20

• Dispositif de configuration série Altera - EPCS4

• USB Blaster (intégré) pour la programmation et le contrôle de l'API utilisateur ; JTAG et série active

Les modes de programmation (AS) sont pris en charge

• SRAM de 512 Ko

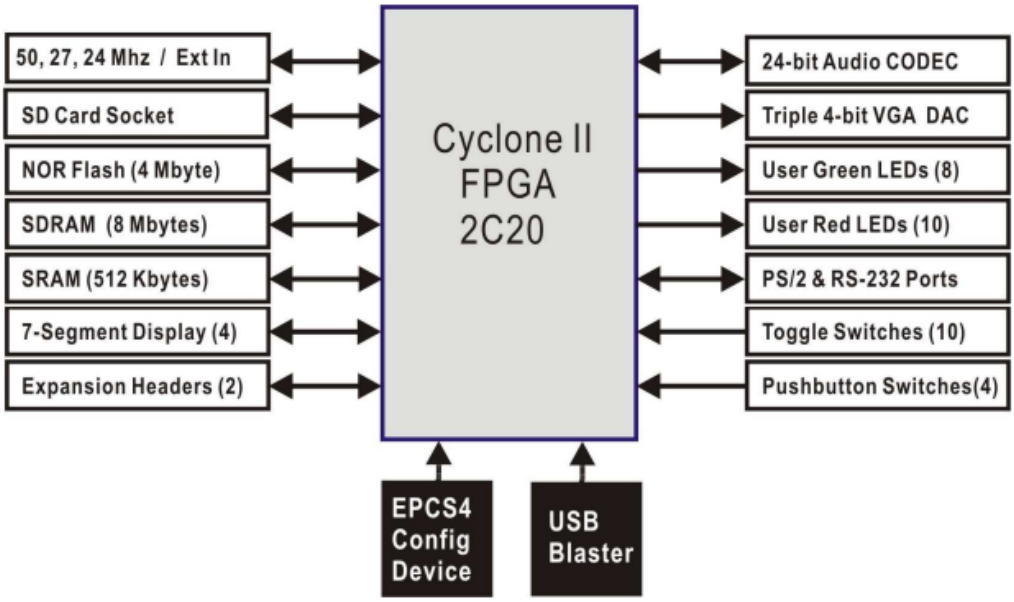
• SDRAM de 8 Mo

- Mémoire Flash de 4 Mo
- Prise pour carte SD
- 4 interrupteurs à bouton-poussoir
- 10 interrupteurs à bascule
- 10 LED utilisateur rouges
- 8 LED utilisateur vertes
- Oscillateur 50 MHz, oscillateur 27 MHz et oscillateur 24 MHz pour les sources d'horloge
- CODEC audio 24 bits de qualité CD avec prises d'entrée, de sortie et d'entrée microphone
- DAC VGA (réseau de résistances 4 bits) avec connecteur de sortie VGA
- Émetteur-récepteur RS-232 et connecteur à 9 broches
- Connecteur souris/clavier PS/2
- Deux embases d'extension à 40 broches avec protection de résistance
- Alimenté par un adaptateur CC 7,5 V ou un câble USB

3.3 Schéma fonctionnel de la carte DE1

Au-dessous on donne le schéma synoptique de la carte DE1. Pour offrir une flexibilité maximale à l'utilisateur,

Toutes les connexions sont effectuées via le dispositif Cyclone II FPGA. Ainsi, l'utilisateur peut configurer le FPGA pour mettre en œuvre n'importe quelle conception de système.



4 Conclusion

Au cours de ce chapitre, nous avons étudier le carte programmable FPGA , aussi les différentes composantes de kit de développement Altera cyclone II qui sera notre choix pour la carte de commande. Pour le chapitre suivant nous allons aborder la simulation et réalisation.


Chapitre 3

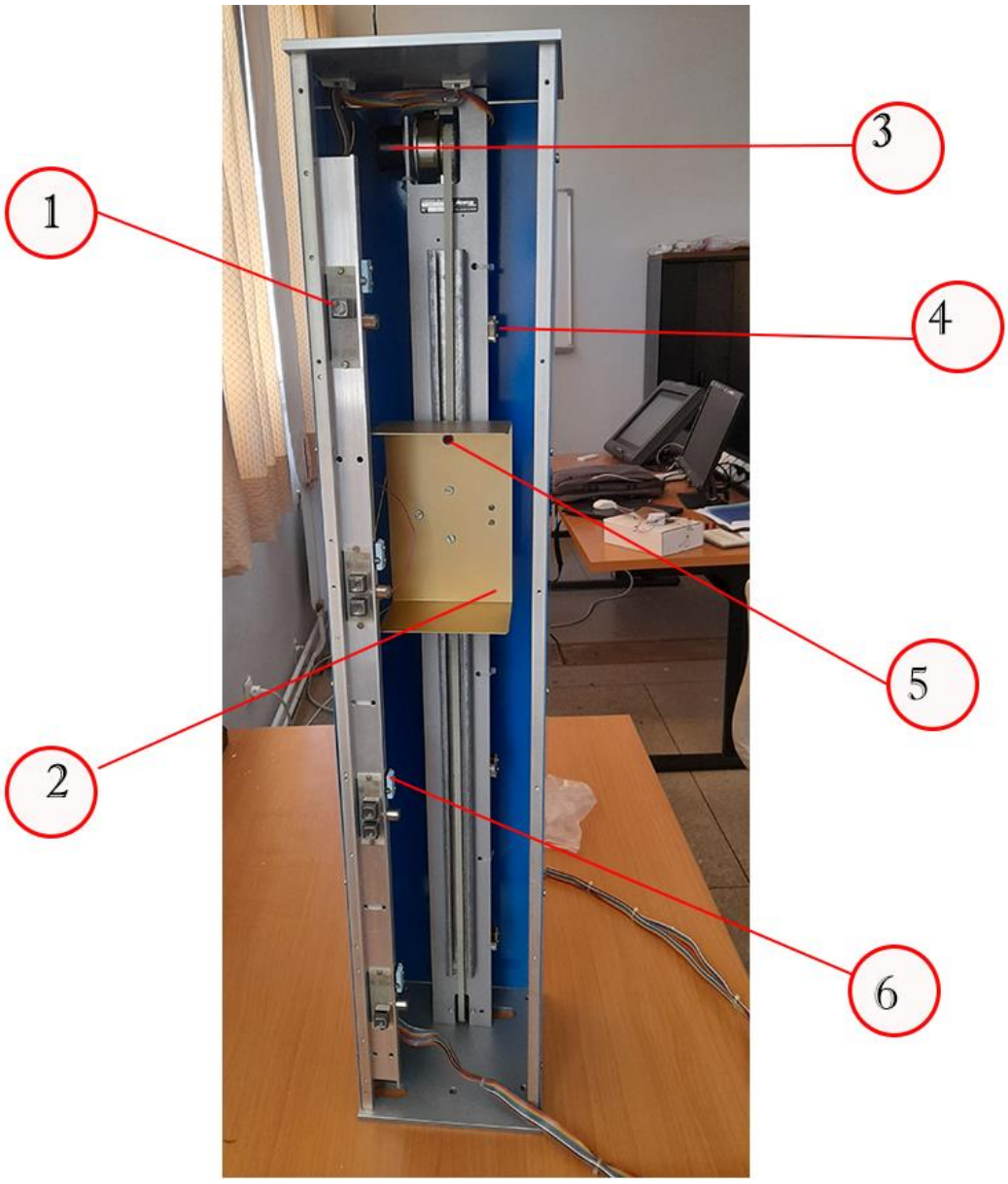
Chapitre 3 : Simulation et réalisation

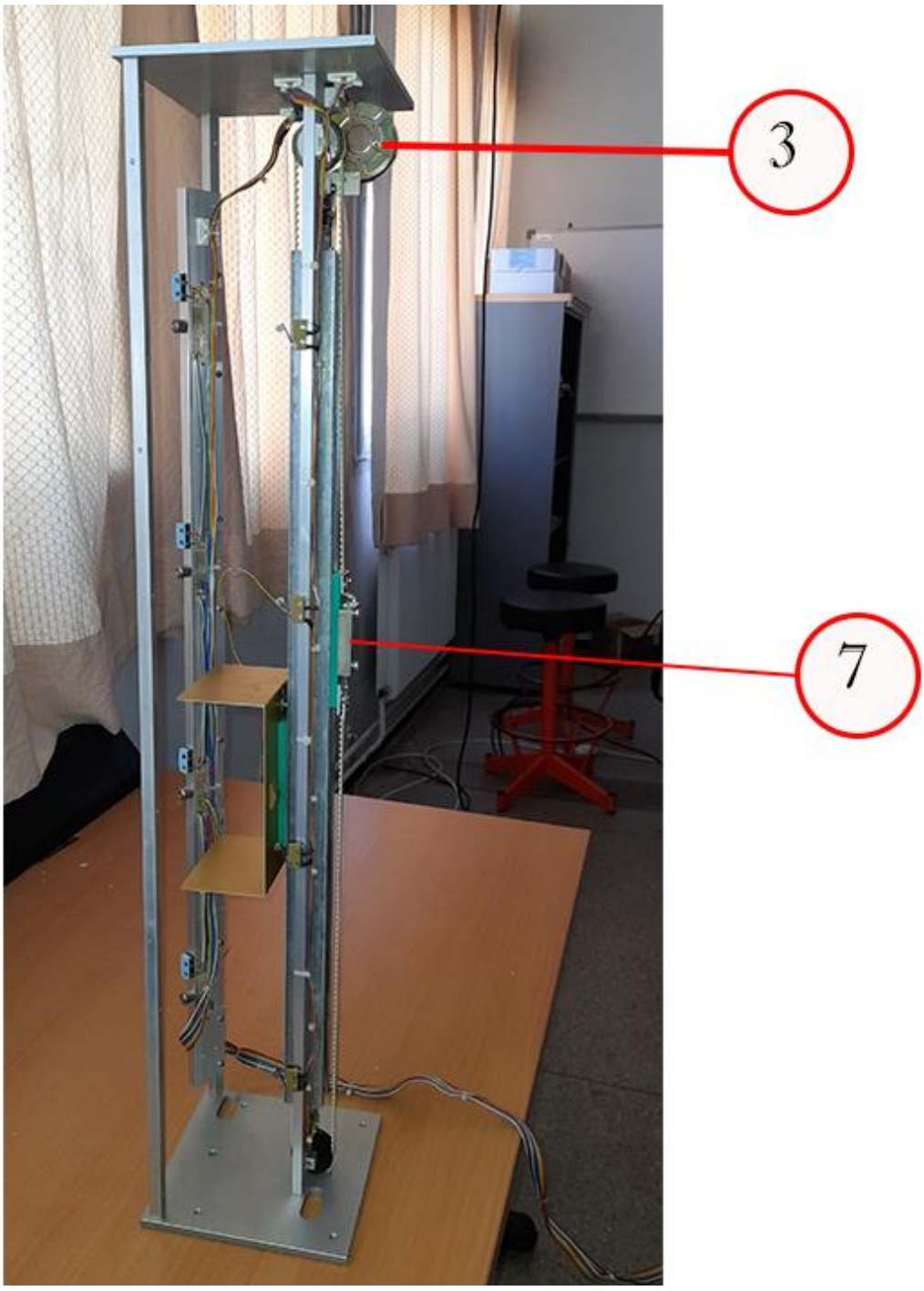
1. Modèle de l'ascenseur

1.1 Système commandé

Il s'agit d'une maquette d'ascenseur à quatre étages, entraînée par deux actionneurs Montée et Descente. La position de la cabine est repérée par quatre capteurs d'étage **det1, det2, det3 et det4** (avec **det**i = 1 cabine à l'étage i). Un dispositif de sécurité coupe le mouvement en cours si la cabine descend au-dessous de l'étage 1 ou si elle monte au-dessus de l'étage 4(**det_fin1, det_fin2).** Six poussoirs palier **bu1, bu2b, bu2h, bu3b, bu3h, bu4**(**buxh** et **buxb** pour palier Montée ou palier Descente) permettent à l'utilisateur d'appeler la cabine à partir du palier de l'étage x. Un contact Porte Ouverte passe à "1" dès qu'une des portes donnant accès à l'ascenseur est ouverte. Deux interrupteurs Stop et Reprise sont destinés, en particulier, à la gestion des urgences sur cette maquette

1.2 Les différentes composantes de l'ascenseur :

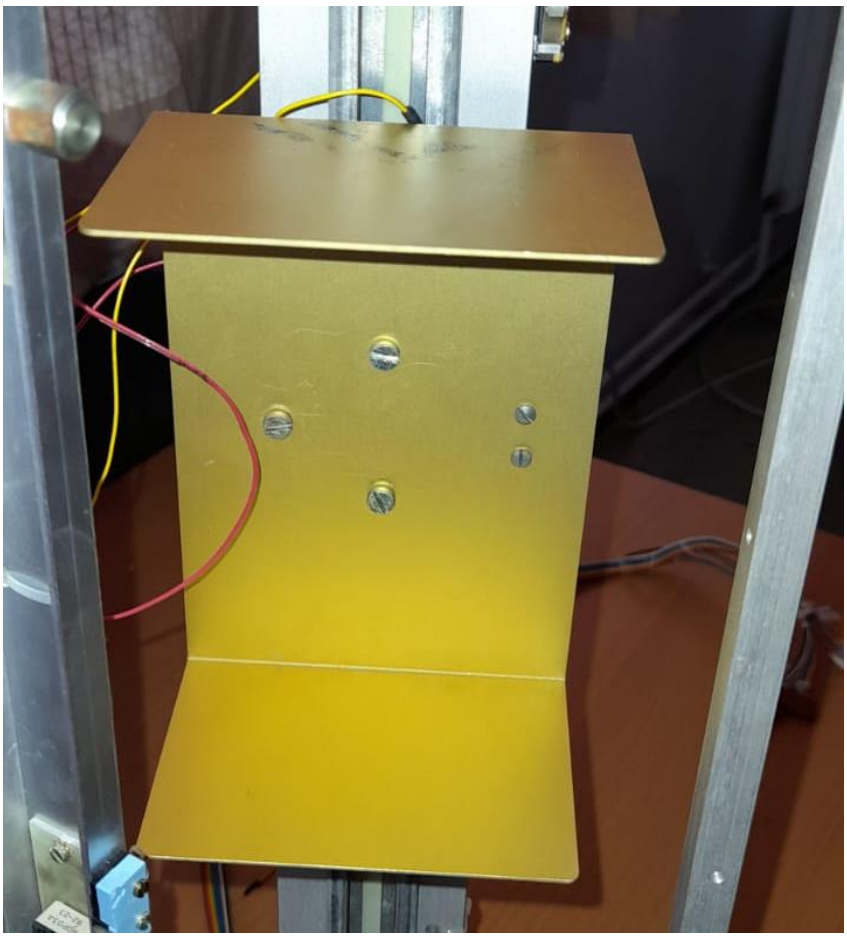




1 : bouton d'appel (HDP O3A 92-23)

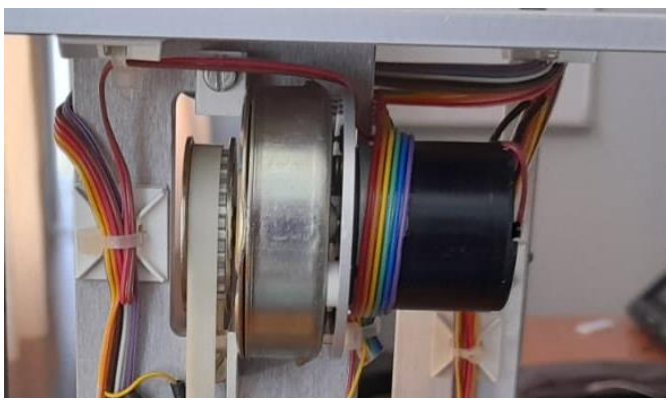
Permet à l'utilisateur d'appeler la cabine à un étage quelconque, chaque étage a son propre bouton.

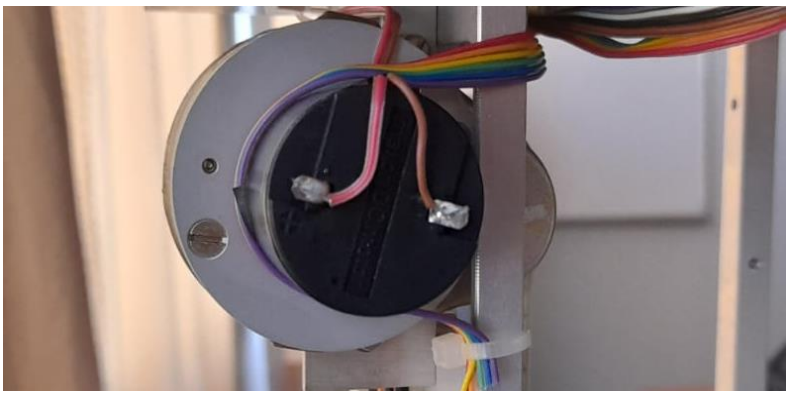
2: la cabine



3:Moteur (maxon DC motor)

Le moteur permet de monté ou descender la cabine par le mouvement de rotatin dans les deux sens selon la tension d'alimentation





4 _ 6 : interrupteur de fin course

C'est un capteur de contact (ouvert ou fermé). Il est souvent utilisé pour connaitre une position :

- la position d'un vérin (début et fin),
- la position d'une porte (ouverte ou fermée),
- La position d'un ascenseur (étage),
- La position d'une barrière automatique (ouverte ou fermée),
- La position d'une manette de jeu (bouton appuyé, direction souhaitée,…).

Fig. 1

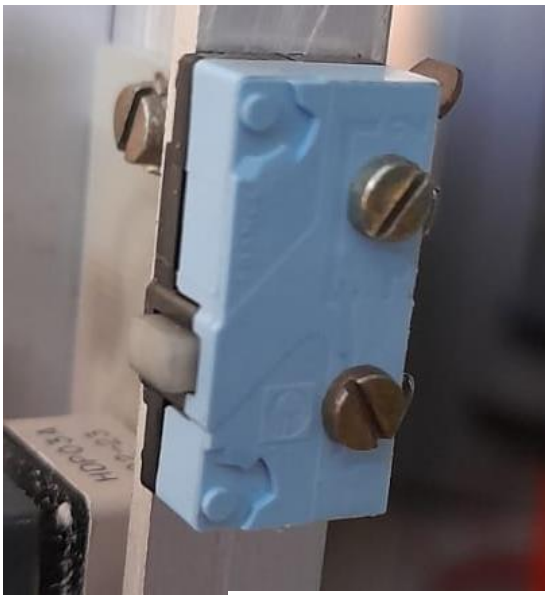


Fig. 2

Le capteur de la Fig.1 indique la position de la cabine

Le capteur de la Fig.2 détecté si la porte est fermé ou ouverte .

5 : Led d'urgence

Il ne s'allume qu'en cas d'urgence (problème)

7.Le contre poids

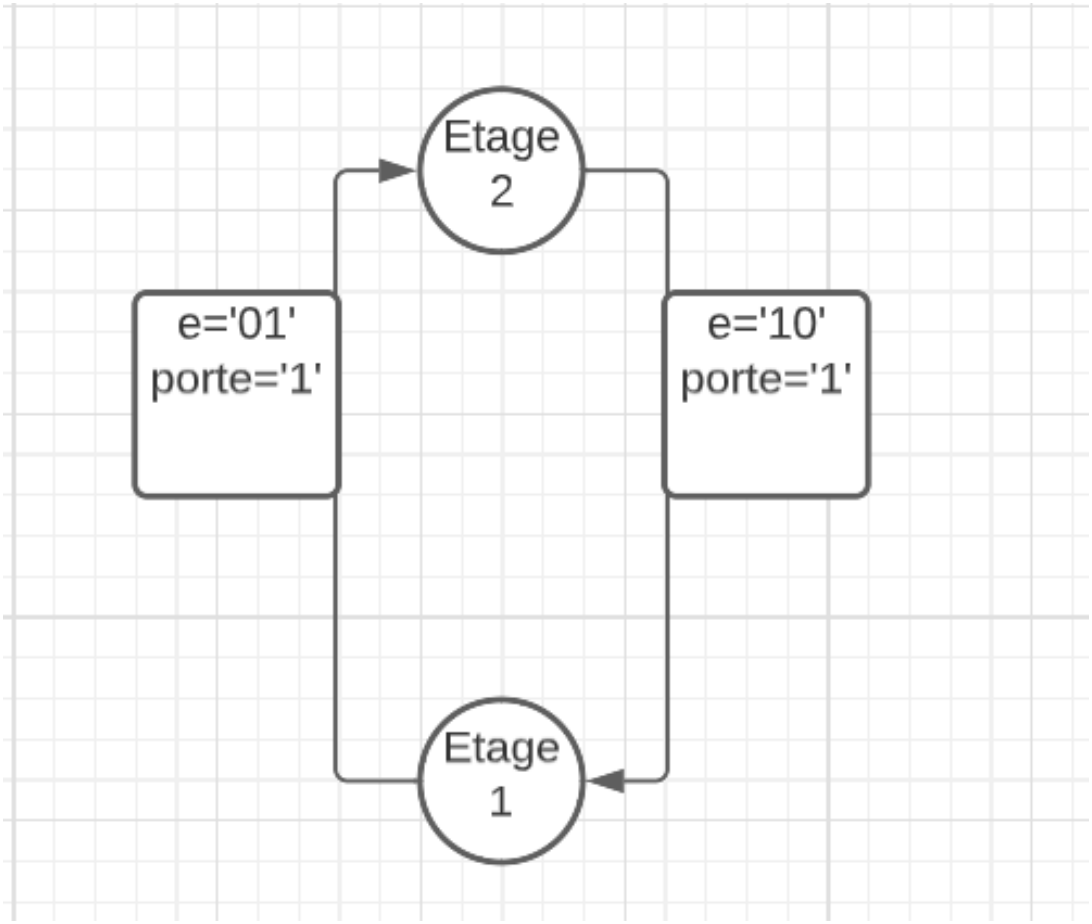
Le contrepoids est une charge lourde qui sert à équilibrer la charge de la cabine et à diminuer l'énergie à fournir par le moteur.

2.Principe de fonctionnement

- Lorsqu'un utilisateur appuie sur un bouton d'appel ,la led de ce bouton s'allume et le moteur tourne dans un sens convenable pour que la cabine atteindre l'étage demandé , dans ce moment la led s'éteindre .
- il y a une vérification de fermeture de tous les protes avant n'import mouvement de la cabine .
- les interrupteur de fin course indiquent la position où il situé la cabine , et permettent d'arrêter le mouvement lorsque on atteint l'étage demandé .

3. contrôler deux étages de l'ascenseur

3.1- machine à état :



e : c'est le vecteur d'appel le 0 signifier la pression de bouton

e= '01' c'est-à-dire e(1)='0' donc l'appel est vient de l'étage 2

e='10' c'est-à-dire e(0)='0' donc l'appel est vient de l'étage 1

3.2 le code VHDL

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


entity FSM2 is
Port (
   e: in std_logic_vector(1 downto 0) ;
   det1,det2: in std_logic ;
   reset : in std_logic ;
   motorP,motorM,led1,led2: out std_logic );
end FSM2 ;

architecture archi of FSM2 is
type etat is (e1,e2) ;
signal etata, etatf : etat ;
signal b :  std_logic_vector(1 downto 0) :="11" ;
begin

process(e,reset)
begin
if reset='1' then b <= "11" ;
else

   if e(0)='0' then b <="10" ;
   elsif e(1)='0' then b <="01" ;
   end if ;

end if ;
end  process ;
```

```
process(reset,b,etata,det1,det2)
begin
if reset ='1' then etata <=e1 ;etatf <=e1;motorP<='0' ;motorM<='0';
else
  case etata is
     when e1 => if b = "01" then
                  etatf <= e2 ;
                  motorP <='1';
                  motorM <='0';
                  if det2 ='0' then
                    etata <=e2;
                  end if;
                else etatf <= e1 ;
                  motorP <='0';
                  motorM <='0';
                end if;
     when e2 => if b="10" then
                etatf <= e1 ;
                motorP <='0';
                motorM <='1';
                if det1 ='0' then
                   etata <=e1;
                end if;
               else
                etatf <=e2 ;
                motorP <='0';
                motorM <='0';
               end if ;
     when others => null;
  end case;

end if ;
```

```
end process;
process (etata,etatf)
begin
case etata is
   when e1 => if etatf = e2 then led2 <='1' ;
              else led1 <='0' ;led2 <='0' ;
              end if;
   when e2 => if etatf = e1 then led1 <='1';
              else led2 <= '0' ;led1 <='0' ;
              end if;
   when others => null;
end case ;
end process;


end archi;
```

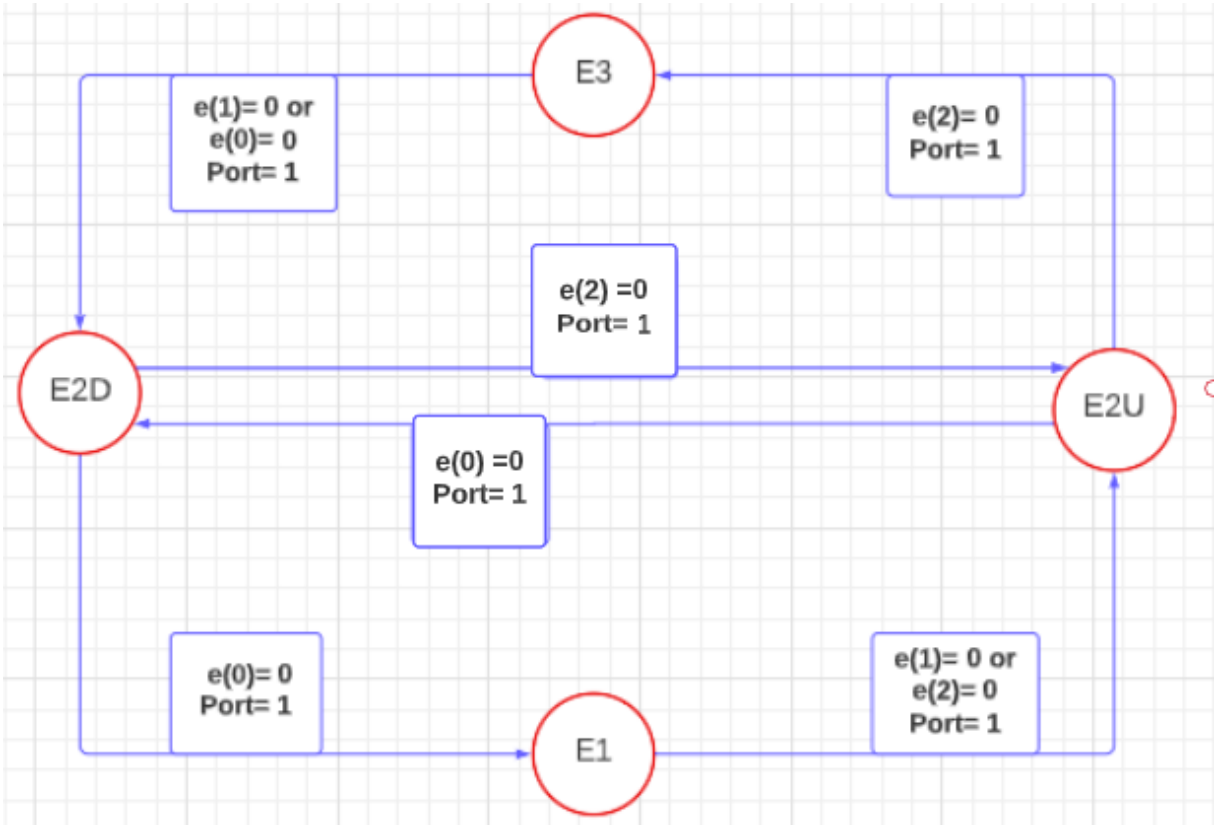
➢ **Configuration de Test Bench**

```
reset <='1' ,'0' after 50 ns ;
e<="10","01" after 100 ns ,"10" after 200 ns ;
det1<='0' ,'1' after 100 ns ,'0' after 250 ns;
det2<='1' ,'0' after 150 ns ,'1' after 200 ns;
```

3.3 la simulation

4. contrôler trois étages de l'ascenseur

4.1- machine à état :



4.2 le code VHDL

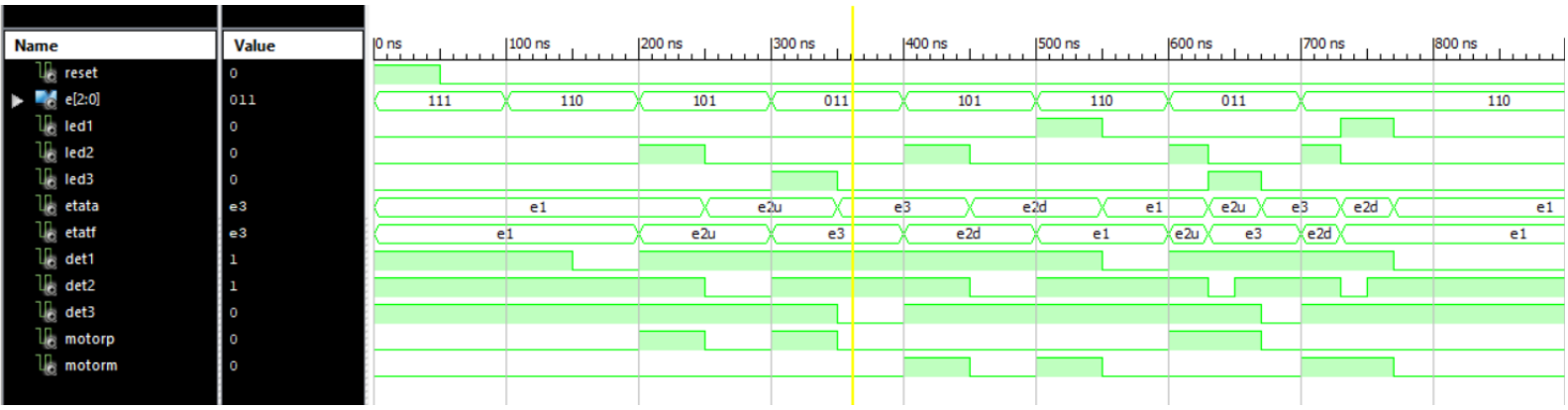
```
1
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4
5
6  entity FSM3 is
7  Port (
8     e: in std_logic_vector(2 downto 0) ;
9     det1,det2,det3 : in std_logic ;
10    reset : in std_logic ;
11    motorP,motorM,led1,led2,led3 : out std_logic );
12 end FSM3 ;
13
14 architecture archi of FSM3 is
15 type etat is (e1,e2d,e2u,e3) ;
16 signal etata, etatf : etat ;
17 signal b :  std_logic_vector(2 downto 0) :="111" ;
18 begin
19
20 process(e,reset)
21 begin
22 if reset='1' then b <= "111" ;
23 else
24
25    if e(0)='0' then b <="110" ;
26    elsif e(1)='0' then b <="101" ;
27    elsif e(2)='0' then b <="011" ;
28    end if ;
29
30 end if ;
31 end  process ;
```

```
35 process(reset,b,etata,det1,det2,det3)
36 begin
37 if reset ='1' then etata <=e1 ;etatf <=e1;motorP<='0' ;motorM<='0';
38 else
39 case etata is
40   when e1 => if b = "011" or b="101" then
41                etatf <= e2u ;
42                motorP <='1';
43                motorM <='0';
44                if det2 ='0' then
45                   etata <=e2u;
46                end if;
47              else etatf <= e1 ;
48                motorP <='0';
49                motorM <='0';
50              end if;
51   when e2u => if b="011" then
52                etatf <= e3 ;
53                motorP <='1';
54                motorM <='0';
55                if det3 ='0' then
56                   etata <=e3;
57                end if;
58               elsif b="110" then
59                etatf <=e2d;
60                motorP <='0';
61                motorM <='0';
62                if det2 ='0' then
63                   etata <=e2d;
64                end if;
65               else
66                etatf <=e2u ;
67                motorP <='0';
68                motorM <='0';
```

```
69                end if;
70    when e2d => if b="110" then
71                 etatf <= e1;
72                 motorP <='0';
73                 motorM <='1';
74                 if det1 ='0' then
75                    etata <=e1;
76                 end if;
77                elsif b="011" then
78                 etatf <= e2u ;
79                 motorP <='0';
80                 motorM <='0';
81                 if det2 ='0' then
82                    etata <=e2u;
83                 end if;
84                else
85                 etatf <= e2d ;
86                 motorP <='0';
87                 motorM <='0';
88                end if;
89    when e3 =>  if b="101" or b="110" then
90                 etatf <= e2d;
91                 motorP <='0';
92                 motorM <='1';
93                 if det2 ='0' then
94                    etata <=e2d;
95                 end if;
96                else
97                 etatf <= e3 ;
98                 motorP <='0';
99                 motorM <='0';
100               end if;
101
102   when others => null;
```

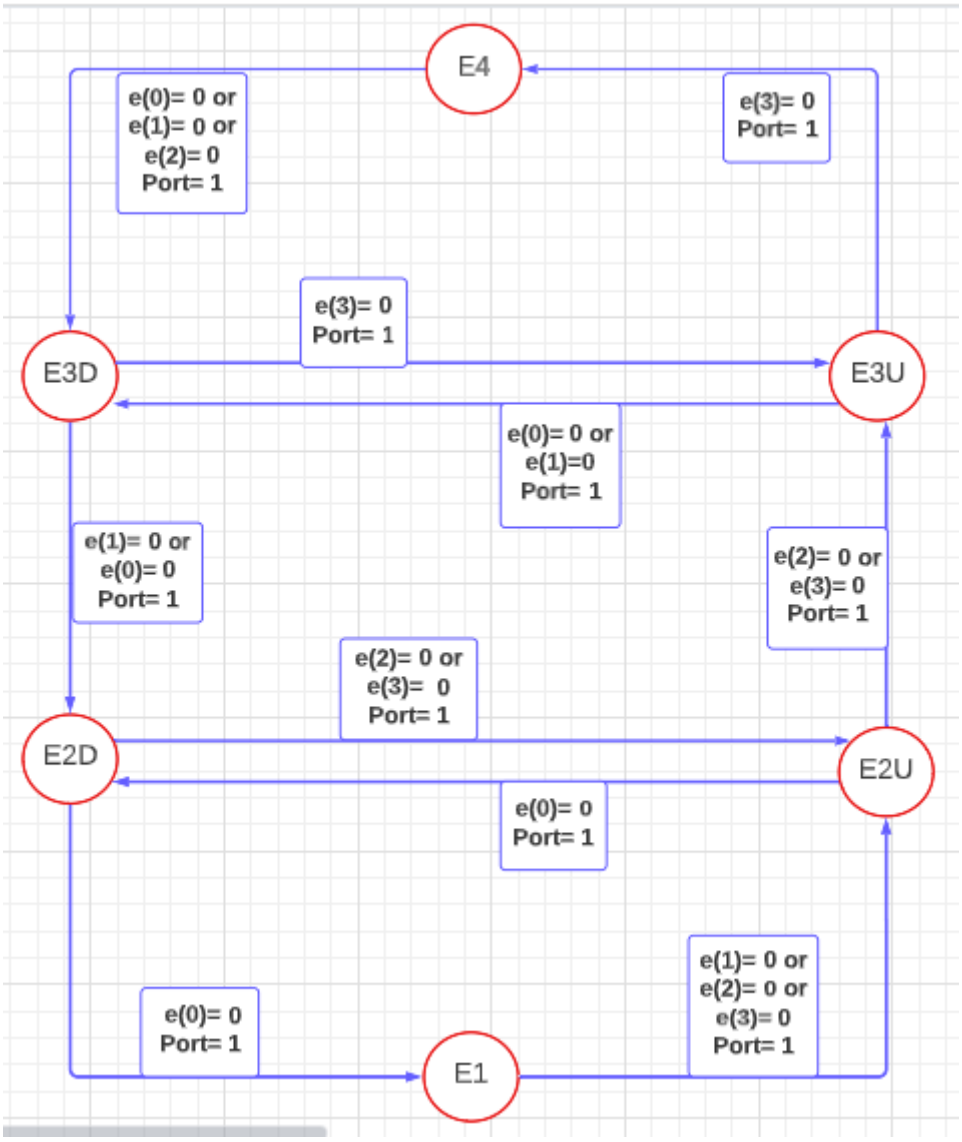
```
103 end case;
104 end if ;
105
106 end process;
107
108
109 process(etata,etatf)
110 begin
111 case etata is
112  when e1 => if etatf =e2u or etatf =e2d then led2 <='1' ;
113              elsif etatf=e3 then led3 <='1' ;
114              else led1 <='0';led2 <='0';led3 <='0' ;
115              end if ;
116  when e2u => if etatf =e1 then led1 <='1' ;
117              elsif etatf=e3 then led3 <='1' ;
118              else led1 <='0';led2 <='0';led3 <='0' ;
119              end if ;
120  when e2d => if etatf =e1 then led1 <='1' ;
121              elsif etatf=e3 then led3 <='1' ;
122              else led1 <='0';led2 <='0';led3 <='0' ;
123              end if ;
124  when e3 => if etatf =e1 then led1 <='1' ;
125              elsif etatf =e2u or etatf =e2d then led2 <='1' ;
126              else led1 <='0';led2 <='0';led3 <='0' ;
127              end if ;
128 end case;
129 end process ;
130
131
132 end archi;
133
134
```

4.3 la simulation



5. contrôler quatre étages de l'ascenseur

5.1- machine à état :



5.2 le code VHDL :

```
1
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4
5
6  entity cas4Etage is
7  Port (
8     e: in std_logic_vector(3 downto 0) ;
9     det1,det2,det3,det4 : in std_logic ;
10    reset ,por: in std_logic ;
11    motorP,motorM,led1,led2,led3,led4 : out std_logic );
12 end cas4Etage ;
13
14 architecture archi of cas4Etage is
15 type etat is (e1,e2d,e2u,e3d,e3u,e4) ;
16 signal etata, etatf ,etatm: etat ;
17 signal b :  std_logic_vector(3 downto 0) :="1111" ;
18 begin
19
20 process(e,reset)
21 begin
22 if reset='1' then b <= "1111" ;
23 else
24    if e(0)='0' then b <="1110" ;
25    elsif e(1)='0' then b <="1101" ;
26    elsif e(2)='0' then b <="1011" ;
27    elsif e(3)='0' then b <="0111" ;
28    end if ;
29 end if ;
30 end  process ;
31
34
35 process(reset,b,por,etata,det1,det2,det3,det4)
36
37 begin
38
39 if reset ='1' then etata <=e1 ;etatf <=e1;motorP<='0' ;motorM<='0';
40 else
41
42 case etata is
43   when e1 => if b = "0111" or b="1011" or b="1101" then
44                if por='1' then
45                   etatf <= e2u ;
46                   motorP <='1';
47                   motorM <='0';
48                else
49                   motorP <='0';
50                   motorM <='0';
51                end if;
52                if det2 ='0' then
53                   etata <=e2u;
54                end if;
55              else etatf <= e1 ;
56                motorP <='0';
57                motorM <='0';
58              end if;
59   when e2u => if b = "0111" or b="1011"  then
60
61                if por='1' then
62                   etatf <= e3u ;
63                   motorP <='1';
64                   motorM <='0';
65                else
66                   motorP <='0';
67                   motorM <='0';
```

```
68                end if;
69                if det3 ='0' then
70                   etata <=e3u;
71                end if;
72               elsif b="1110" then
73                etatf <=e2d;
74                motorP <='0';
75                motorM <='0';
76                if det2 ='0' then
77                   etata <=e2d;
78                end if;
79               else
80                etatf <=e2u ;
81                motorP <='0';
82                motorM <='0';
83               end if;
84    when e2d => if b="1110" then
85
86                if por='1' then
87                   etatf <= e1;
88                   motorP <='0';
89                   motorM <='1';
90                else
91                   motorP <='0';
92                   motorM <='0';
93                end if;
94                if det1 ='0' then
95                   etata <=e1;
96                end if;
97               elsif b="0111" or b="1011" then
98                etatf <= e2u ;
99                motorP <='0';
100               motorM <='0';
101               if det2 ='0' then
102                  etata <=e2u;
103               end if;
104              else
105               etatf <= e2d ;
106               motorP <='0';
107               motorM <='0';
108              end if;
109   when e3u => if b="1101" or b="1110" then
110               etatf <= e3d;
111               motorP <='0';
112               motorM <='0';
113               if det3 ='0' then
114                  etata <=e3d;
115               end if;
116              elsif  b="0111" then
117
118               if por='1' then
119                  etatf <= e4;
120                  motorP <='1';
121                  motorM <='0';
122               else
123                  motorP <='0';
124                  motorM <='0';
125               end if;
126               if det4 ='0' then
127                  etata <=e4;
128               end if;
129              else
130               etatf <= e3u ;
131               motorP <='0';
132               motorM <='0';
```

```
133                end if;
134    when e4 => if b = "1110" or b="1011" or b="1101" then
135
136                 if por='1' then
137                    etatf <= e3d ;
138                    motorP <='0';
139                    motorM <='1';
140                 else
141                    motorP <='0';
142                    motorM <='0';
143                 end if;
144                 if det3 ='0' then
145                    etata <=e3d;
146                 end if;
147               else etatf <= e4 ;
148                 motorP <='0';
149                 motorM <='0';
150               end if;
151    when e3d => if b="0111" then
152                 etatf <= e3u;
153                 motorP <='0';
154                 motorM <='0';
155                 if det3 ='0' then
156                    etata <=e3u;
157                 end if;
158                elsif b="1110" or b="1101" then
159
160                 if por='1' then
161                    etatf <= e2d ;
162                    motorP <='0';
163                    motorM <='1';
164                 else
165                    motorP <='0';
166                    motorM <='0';
```

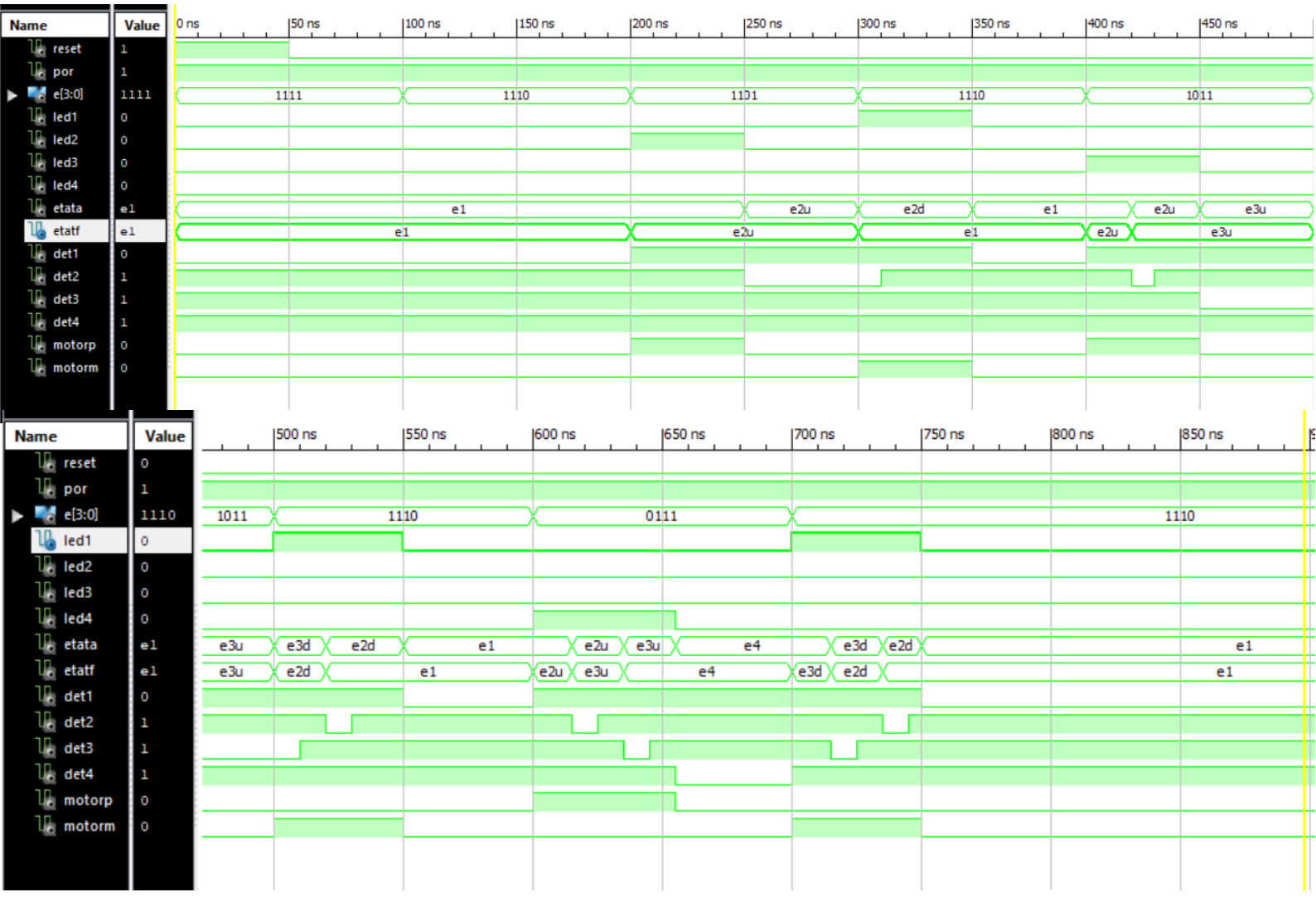
```
167                   end if;
168                   if det2 ='0' then
169                      etata <=e2d;
170                   end if;
171                  else
172                   etatf <= e3d ;
173                   motorP <='0';
174                   motorM <='0';
175                  end if;
176
177     when others => null;
178  end case;
179  end if ;
180
181  end process;
182  process (b,etata,etatf)
183  begin
184  case etata is
185   when e1 => if b="1101" then led2<='1' ;
186               elsif b="1011" then led3<='1' ;
187               elsif b="0111" then led4<='1' ;
188               else led1 <='0';led2<='0';led3 <='0';led4 <='0' ;
189               end if;
190   when e2u|e2d => if b="1110" then led1<='1' ;
191                   elsif b="1011" then led3<='1' ;
192                   elsif b="0111" then led4<='1' ;
193                   else led1 <='0';led2<='0';led3 <='0';led4 <='0' ;
194                   end if;
195   when e3u|e3d => if  b="1110" then led1<='1' ;
196                   elsif  b="1101" then led2<='1' ;
197                   elsif  b="0111" then led4<='1' ;
198                   else led1 <='0';led2<='0';led3 <='0';led4 <='0' ;
199                   end if;
200   when e4 => if  b="1101" then led2<='1' ;
```

```
200   when e4 => if  b="1101" then led2<='1' ;
201               elsif  b="1011" then led3<='1' ;
202               elsif  b="1110" then led1<='1' ;
203               else led1 <='0';led2<='0';led3 <='0';led4 <='0' ;
204               end if;
205   end case ;
206   end process;
207
208
209   end archi;
210
```

- **Test Bench**

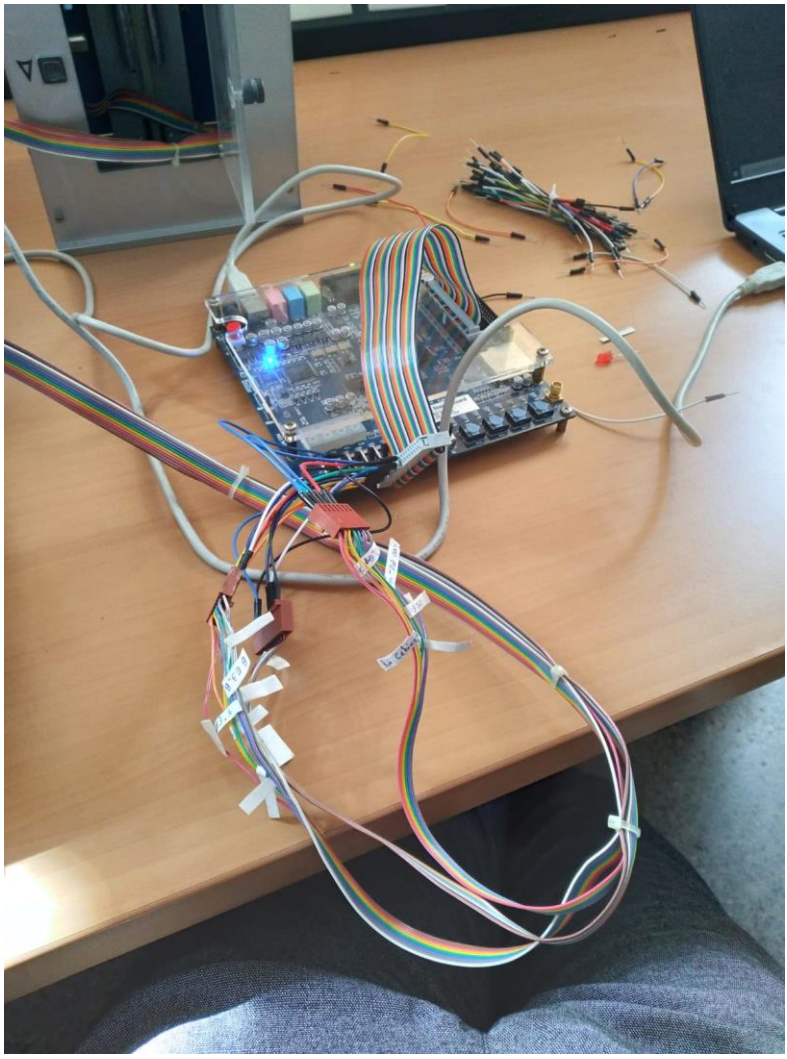
```
reset <='1' ;'0' after 50 ns;
e <='1111','1110' after 100 ns,'1101' after 200ns ,'1110' after 300 ns ,'1011' after 400 ns, '1110' after 500 ns ,
    '0111' after 600 ns,'1110' after 700 ns;
det1 ='0','1' after 200 ns,'0' after 350 ns,'1' after 400 ns,'0' after 550 ns,'1' after 600 ns,'0' after 750 ns;
det2 ='1','0' after 250 ns,'1' after 300 ns,'0' after 420 ns,'1' after 430 ns,'0' after 520 ns,'1' after 530 ns,
      '0' after 615 ns,'1' after 625 ns,'0' after 735 ns,'1' after 745 ns;
det3 ='1','0' after 450 ns,'1' after 500 ns,'0' after 635 ns,'1' after 645 ns,'0' after 715 ns,'1' after 725 ns;
det4 ='1','0' after 655 ns ,'1' after 700 ns;
```

5.3 la simulation



6. Réalisation :

En raison d'un ensemble de problèmes et d'obstacles auxquels nous avons été confrontés, notamment le non fonctionnement de plusieurs composants de l'ascenseur, les problèmes des courts circuits, l'éteindre de la carte FPGA d'une manière incompréhensible …. Nous n'avons pas fait de progrès significatifs à ce côté pratique.



Conclusion générale

Finalement, malgré quelques problèmes que nous avons rencontrés, notamment, le problème du temps de l'accès au salle des travaux pratiques qui n'est pas assez suffisant pour bien avancer dans notre projet, malgré que nous y allions constamment. Nous avons réussi à faire un code VHDL qui commande l'ascenseur de quatre étages et le simuler.

Ce projet était intéressant pour plusieurs raisons. D'abord, l'ascenseur est un système répandu dans la vie courante qui nous est familier. Mais surtout cela nous a permis de travailler en groupe, en collectivité. Cette expérience nous a donné, d'ores et déjà, une idée du monde du travail. Ainsi que nous avons dû faire face à plusieurs problèmes plus ou moins faciles à résoudre.