Rapport TP2 MCSED

Bénistant Raphaël Bourlot Xavier

3Imacs-ae-c, Binôme 1
30 mai 2020

Table des matières

1 Introduction

L'objectif de ce TP est de réaliser la commande d'une maquette d'ascenseur par un circuit logique programmable de type FPGA de la famille Xilinx. Les résultats attendus sont la gestion complète des appels des utilisateurs depuis et à l'extérieur de la cabine et la gestion complète du déplacement de la cabine en fonction de ces appels. Afin d'atteindre notre but, nous avons implémenté des machines à état sur le logiciel Xilinx ISE qui permet de faire la passerelle entre l'ordinateur, la carte FPGA de type Spartan 3 et la maquette d'ascenseur. Nous avons dans un premier temps géré les appels de la cabine, puis dans un seconde partie, nous avons inclus les déplacements de celle-ci.

2 Mémorisation des appels

Dans cette première partie, nous nous sommes attelés au codage de la gestion des appels à tous les étages que ce soit à l'intérieur ou à l'extérieur de la cabine. Chaque étage est modélisé par une machine à état qui comporte deux états possibles : APPEL ou PASAPPEL. Le Statechart correspond est le suivant :

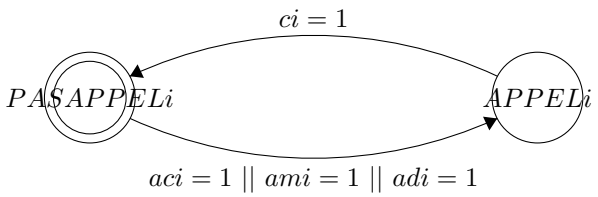

FIGURE 1 – Mémorisation des appels à l'étage i

avec

Variable	Entrée/Sortie
aci	Appel pour l'étage i
ami	Appel montée pour l'étage i
adi	Appel descente pour l'étage i
ci	Cabine à l'étage i

Naturellement, tous les étages présentent la même machine d'état. Pour obtenir le statechart réel, il suffit de remplacer toutes les occurences de i par le numéro de l'étage considéré :

Étage	Numéro correspondant
RDC	0
1	1
2	2
3	3

Á ce stade, aucun déplacement automatique de la cabine n'est encore possible. Cependant, lorsque nous effectuons des appels, ceux-ci sont bien pris en compte et lorsque nous faisons manuellement passer la cabine devant les étages correspondants, les appels sont démarqués.

3 Commande de la cabine

Dans cette deuxième partie, nous nous sommes occupés de gérer le déplacement de la cabine en fonction des différents appels. Pour réaliser cette gestion, nous avons mis en place deux machines à états qui présentent respectivement les états suivants : [M (montée), D (descente), N (neutre)] pour la première et [E0 (RDC), E1 (étage 1), E2 (étage 2), E3 (étage 3)] pour la seconde. La première à pour but de déterminer si l'ascenceur monte (M), s'il descend (D) ou s'il est en attente d'instruction (N). La deuxième machine à état quant à elle permet de repérer la position de l'ascenseur. La machine à état du système complet est la suivante :

Données :

— État initial = état doublement entouré.
— Priorités : transitions évaluées de haut en bas dans le graphe, toutes les transitions menant vers M avant celles menant vers D.

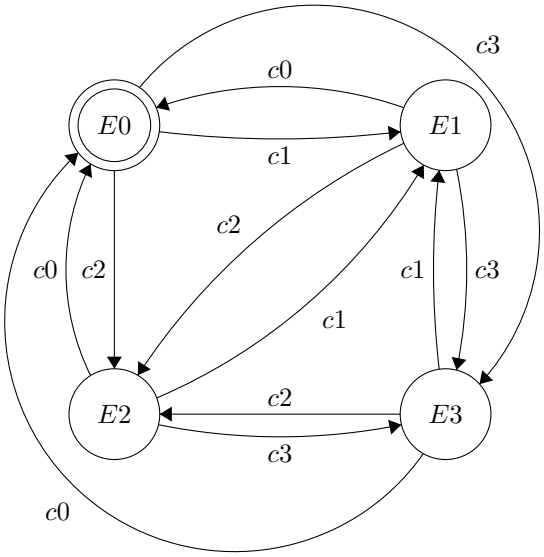

FIGURE 2 – Mémorisation de la position de la cabine d'ascenseur

avec

Variable	Entrée/Sortie
ci	Cabine à l'étage i

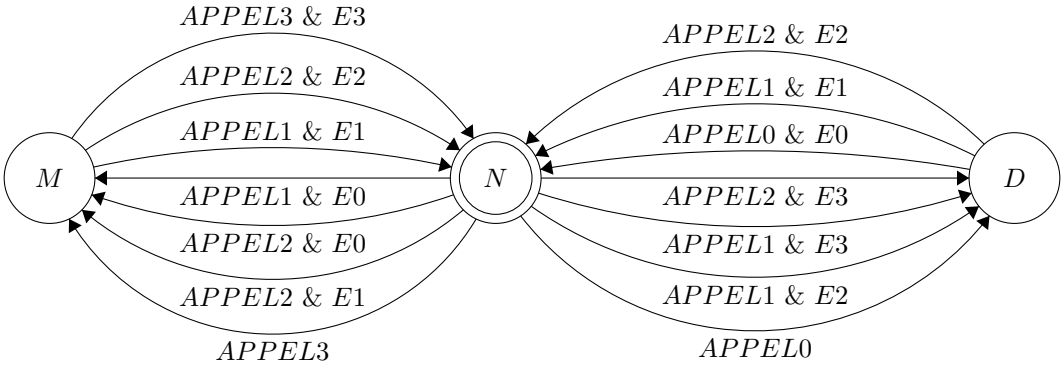

FIGURE 3 – Mémorisation de l'état de la cabine d'ascenseur avec priorités

All statecharts courtesy of "Finite State Machine Designer" by Evan Wallace, 2010 from http://www.madebyevan.com/fsm/

4 Conclusion

Bien que la prise en main du logiciel Xilinx ISE fut laborieuse, notamment à cause de la complexité de la compilation sur la carte FPGA de type Spartan 3, ce TP a été un excellent moyen d'utiliser directement nos connaissances théoriques sur les machines d'états acquéries au cours des cours magistraux de MCSED.
Nous avons dans un premier temps implémenté des machines à états très simples qui ne comprenaient que deux

états pour la mémorisation des appels avant de nous attaquer au vif du sujet qui était la gestion du déplacement de la cabine.
Au delà de la pénibilité de la déclarations des entrées et des sorties, la partie la plus délicate de notre projet a été d'ordonner la gestion des changements d'états pour le déplacement de la cabine pour ne pas voir la cabine faire volteface avant d'atteindre son étage-objectif premier. Pour ce faire, nous avons suivi le cahier des charges qui nous imposait de privilégier les étages encore accessibles sans modification du sens de déplacement de l'ascenseur.
Comme nous avions fini les deux premières étapes légèrement en avance, nous avons fait une tentative de gestion de l'ouverture des portes de l'ascenseur. Cependant, cette tentative n'a pas abouti par manque de temps.

A Annexes

A.1 Vidéo du montage en fonctionnement

Vidéo du montage en fonctionnement

A.2 Code VHDL

```
1  ----------------------------------------------------------------------
2  -- Create Date :     21/11/13
3  -- Authors :EC GLC
4  -- Project Name :     Ascenseur Deplacement Simple
5  -- Additional Comments :
6  ----------------------------------------------------------------------
7  library IEEE;
8  use IEEE.STD_LOGIC_1164.ALL;
9  use IEEE.STD_LOGIC_ARITH.ALL;
10 use IEEE.STD_LOGIC_UNSIGNED.ALL;
11
12 entity ascenceur is
13     Port (      -- Moteur de montee
14           MONTER : out STD_LOGIC;
15          -- Moteur de descente
16          DESCENDRE : out STD_LOGIC;
17          -- Voyant cabine
18          A1 : out STD_LOGIC;
19          A2 : out STD_LOGIC;
20          A3 : out STD_LOGIC;
21          A0 : out STD_LOGIC;
22           -- Reset
23          RESET : in STD_LOGIC;
24          -- Capteurs de presence de la cabine a l'etage 0 et 1
25          --c0 : in STD_LOGIC;
26          c1 : in STD_LOGIC;
27          c2 : in STD_LOGIC;
28          c3 : in STD_LOGIC;
29          c0 : in STD_LOGIC;
30          -- Appels depuis la cabine enregistres aux etages 1
31          ac1 : in STD_LOGIC;
32          ac2 : in STD_LOGIC;
33          ac3 : in STD_LOGIC;
34          ac0 : in STD_LOGIC;
35
36          am1 : in STD_LOGIC;
37          am2 : in STD_LOGIC;
38          am3 : in STD_LOGIC;
39          am0 : in STD_LOGIC;
40
41          ad1 : in STD_LOGIC;
42          ad2 : in STD_LOGIC;
43          ad3 : in STD_LOGIC;
44          ad0 : in STD_LOGIC;
45
46
47          -- Signal d'horloge
48          CLK: in STD_LOGIC
49            );
50 end ascenceur;
51
```

```
52  architecture MachineAEtats of ascenceur is
53
54   type etat4 is (E0,E1,E2,E3);
55   signal position: etat4;
56
57   type etat5 is (M,D,N);
58   signal montdes: etat5;
59
60   type etat1 is (APPEL1, PASAPPEL1);
61   signal mode1: etat1;
62
63    type etat2 is (APPEL2, PASAPPEL2);
64   signal mode2: etat2;
65
66    type etat3 is (APPEL3, PASAPPEL3);
67   signal mode3: etat3;
68
69    type etat0 is (APPEL0, PASAPPEL0);
70   signal mode0: etat0;
71  begin
72
73      GestEtage : process
74      begin
75
76      --sur le front montant de l'horloge
77      wait until CLK='1' and CLK'event;
78      --description de la machine a etat
79      --initialisation
80      if (RESET='1') then
81                  position<=E0;
82                  montdes<=N;
83      elsif ((c0='1')) then
84                  position<=E0;
85      elsif ((c1='1')) then
86                  position<=E1;
87      elsif (c2='1') then
88                  position<=E2;
89      elsif (c3='1') then
90                  position<=E3;
91      end if;
92
93
94      if (mode1=APPEL1 and (position=E0) and (montdes=M or montdes=N)) then
95          MONTER<='1';
96          montdes<=M;
97          DESCENDRE<='0';
98      elsif (mode2=APPEL2 and (position=E0) and (montdes=M or montdes=N)) then
99          MONTER<='1';
100         montdes<=M;
101         DESCENDRE<='0';
102     elsif (mode2=APPEL2 and (position=E1) and (montdes=M or montdes=N)) then
103         MONTER<='1';
104         montdes<=M;
105         DESCENDRE<='0';
106     elsif (mode3=APPEL3 and (montdes=M or montdes=N)) then
107         MONTER<='1';
108         montdes<=M;
109         DESCENDRE<='0';
```

```
110       elsif (mode2=APPEL2 and (position=E3) and (montdes=D or montdes=N)) then
111           MONTER<='0';
112           DESCENDRE<='1';
113           montdes<=D;
114       elsif (mode1=APPEL1 and (position=E3) and (montdes=D or montdes=N)) then
115           MONTER<='0';
116           DESCENDRE<='1';
117           montdes<=D;
118       elsif (mode1=APPEL1 and (position=E2) and (montdes=D or montdes=N)) then
119           MONTER<='0';
120           DESCENDRE<='1';
121           montdes<=D;
122       elsif (mode0=APPEL0 and (montdes=D or montdes=N)) then
123           MONTER<='0';
124           DESCENDRE<='1';
125           montdes<=D;
126
127       else
128           MONTER<='0';
129           DESCENDRE<='0';
130           montdes<=N;
131       end if;
132
133       end process GestEtage ;
134   --
135
136           MAppel1 : process
137       begin
138
139       --sur le front montant de l'horloge
140       wait until CLK='1' and CLK'event;
141       --description de la machine a etat
142       --initialisation
143       if (RESET='1') then
144                   mode1<=PASAPPEL1;
145       elsif (mode1=PASAPPEL1 and (ac1='1' or am1='1' or ad1='1')) then
146                   mode1<=APPEL1;
147       elsif (mode1=APPEL1 and c1='1') then
148                   mode1<=PASAPPEL1;
149
150       end if;
151
152       case mode1 is
153       when APPEL1 => A1<='1';
154       when PASAPPEL1 => A1<='0';
155       end case;
156
157       end process MAppel1 ;
158
159               MAppel2 : process
160       begin
161
162       --sur le front montant de l'horloge
163       wait until CLK='1' and CLK'event;
164       --description de la machine a etat
165       --initialisation
166       if (RESET='1') then
167                   mode2<=PASAPPEL2;
```

```
168         elsif (mode2=PASAPPEL2 and (ac2='1' or am2='1' or ad2='1')) then
169                     mode2<=APPEL2;
170         elsif (mode2=APPEL2 and c2='1') then
171                     mode2<=PASAPPEL2;
172
173         end if;
174
175         case mode2 is
176         when APPEL2 => A2<='1';
177         when PASAPPEL2 => A2<='0';
178         end case;
179
180         end process MAppel2 ;
181                     MAppel3 : process
182         begin
183
184         --sur le front montant de l'horloge
185         wait until CLK='1' and CLK'event;
186         --description de la machine a etat
187         --initialisation
188         if (RESET='1') then
189                     mode3<=PASAPPEL3;
190         elsif (mode3=PASAPPEL3 and (ac3='1' or am3='1' or ad3='1')) then
191                     mode3<=APPEL3;
192         elsif (mode3=APPEL3 and c3='1') then
193                     mode3<=PASAPPEL3;
194
195         end if;
196
197         case mode3 is
198         when APPEL3 => A3<='1';
199         when PASAPPEL3 => A3<='0';
200         end case;
201
202         end process MAppel3 ;
203
204                     MAppel0 : process
205         begin
206
207         --sur le front montant de l'horloge
208         wait until CLK='1' and CLK'event;
209         --description de la machine a etat
210         --initialisation
211         if (RESET='1') then
212                     mode0<=PASAPPEL0;
213         elsif (mode0=PASAPPEL0 and (ac0='1' or am0='1' or ad0='1')) then
214                     mode0<=APPEL0;
215         elsif (mode0=APPEL0 and c0='1') then
216                     mode0<=PASAPPEL0;
217
218         end if;
219
220         case mode0 is
221         when APPEL0 => A0<='1';
222         when PASAPPEL0 => A0<='0';
223         end case;
224
225         end process MAppel0 ;
```

```
226  -- fin
227
228
229  end MachineAEtats;
```