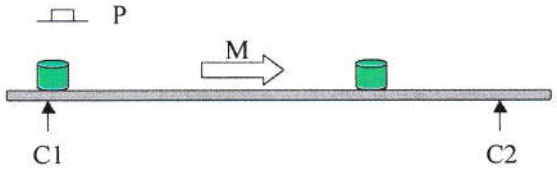
1. Modélisation d'un convoyeur

On considère le système représenté sur la figure suivante constitué d'un convoyeur linéaire, d'un bouton poussoir *P* et de deux capteurs de position tout-ou-rien *C1* et *C2*. Le convoyeur se déplace de gauche à droite grâce à l'actionneur *M* et est chargé de l'acheminement de pièce de *C1* vers *C2*.



En absence de pièce le convoyeur est supposé être en marche. Pour pouvoir charger une pièce à la position *C1*, l'utilisateur du système doit appuyer sur *P*. Cette action provoque l'arrêt du convoyeur à condition qu'aucune pièce ne soit déjà présente à cette position. Le convoyeur se remet en marche lorsque la pièce a été posée et que un nouvel appui sur *P* est détecté. Lorsque la pièce arrive en *C2*, le convoyeur stoppe à nouveau pour permettre le déchargement de la pièce. Il repart dès que la pièce a été déchargée à condition qu'aucun chargement sur *C1* ne soit simultanément en cours. Dans le cas contraire, l'arrêt du convoyeur est prolongé jusqu'à ce que le chargement soit aussi achevé.

a) Proposer une modélisation de cette commande lorsque seul le chargement est pris en compte.
b) Proposer un modèle pour le cas général
c) Proposer une mise en œuvre de cette commande par rebouclage direct.

2. Synthèse de la commande d'un portail à ouverture automatique

Il est demandé de modéliser le système de commande d'ouverture d'un portail automatique de garage. Ce système possède les caractéristiques suivantes :

- Deux signaux "**Ouv**" et "**Fer**" issus d'un boîtier de télécommande permettent la commande à distance de l'ouverture et de la fermeture. L'ouverture (resp. fermeture) est provoquée par la mise à 1 de la sortie "**O**" (resp. "**F**").
- Les positions ouverte et fermée sont repérées par les capteurs "**FinOuv**" et "**FinFer**".
- Un signal "**Eff**" est émis par les moteurs du mécanisme d'ouverture-fermeture si un des battants du portail heurte un obstacle pendant son mouvement. Un signal d'alarme "**Alarme**" est alors émis jusqu'à ce qu'un appui sur un poussoir "**Rearm**" relance le mouvement interrompu.
- Le portail en position fermée, l'appui sur un poussoir "**Verrou**" interdit sa commande depuis le boîtier de télécommande. L'appui sur le poussoir "**Rearm**" relance l'automatisme et provoque l'ouverture du portail.

La modélisation va être décomposée en trois étapes, comme indiqué ci-dessous.

a) Modéliser par un graphe d'états le cycle ouverture-fermeture du portail en l'absence d'obstacle et de demande de verrouillage.
b) **Après avoir retracé le graphe obtenu à la question précéden**te, y faire figurer la prise en compte de la détection d'un obstacle.
c) Reprendre **sur un troisième graphe** celui de la question 2 et y rajouter la prise en compte du verrouillage-déverrouillage.
d) Donner les expressions logiques correspondant à une réalisation par rebouclage direct.

3. Gestion d'un monte charge

Un monte-charge se déplace entre trois étages repérés par les capteurs « Et1 », « Et2 » et « Et3 ». Il possède 2 modes de fonctionnement sélectionnés par un interrupteur « AUT » : le mode manuel et le mode automatique. La cabine monte ou descend respectivement par mise à 1 des actionneurs « M » ou « D ».

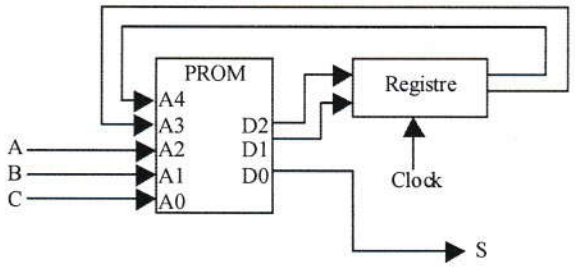
- le mode automatique (AUT=1) : lorsque la cabine est à un étage (capteur Eti enclenché), un appui même très bref sur un bouton poussoir m (resp. d) provoque sa montée (resp. sa descente) vers l'étage immédiatement supérieur (resp. inférieur), à condition que le mouvement soit possible.
- Le mode manuel (AUT=0) : la cabine monte (resp. descend) tant que le bouton poussoir m (resp. d) est actionné.

Le passage d'un mode de fonctionnement automatique à celui manuel doit être immédiat quels que soient le position et l'état de la cabine. Par contre, le retour en mode manuel ne sera pris en compte que si la cabine se trouve en face d'un étage (capteur Eti enclenché).

a) Effectuer la synthèse de ce système dans le cas du mode automatique.
b) Modifier le diagramme d'état obtenu pour y intégrer le fonctionnement du mode manuel.
c) Proposer une mise en œuvre par bascules RS (minimisation du nombre de bascules)
d) Proposer une mise en œuvre par bascules T (1 bascule par état)

4. Analyse

Le schéma de la figure suivante correspond à une mise en œuvre par mémoire d'une système séquentiel possédant 3 entrées [A, B, C] et une sortie [S]. Cinq bits d'adresse [A4..A0] et trois bits mémoire [D2..D0] sont utilisés. Le plan de la mémoire est indiqué dans le tableau placé à droite du schéma.



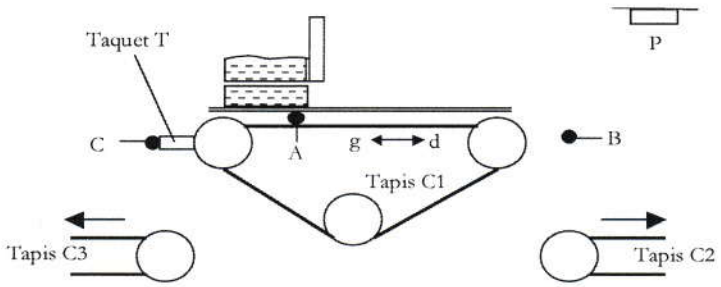
A4	A3	A2	A1	A0	D2	D1	D0
0	0	1	0	-	0	1	0
0	0	1	1	1	1	0	0
0	0	0	-	-	0	0	0
0	0	1	1	0	0	0	0
0	1	-	1	-	1	0	0
0	1	-	0	-	0	1	0
1	0	0	1	-	0	0	1
1	0	0	0	-	0	0	0
1	0	1	1	-	1	0	1
1	0	1	0	-	1	0	0
1	1	-	-	-	1	1	0

(- : 0 ou 1 indifféremment

a) Combien ce système possède t'il de variables internes ? Quelle est la taille mémoire utilisée ?
b) Déduire le graphe des états.
c) Proposer une mise en œuvre de même type dans laquelle la taille mémoire est minimisée et indépendante du nombre des entrées.

5. Convoyeur pour machine à scier

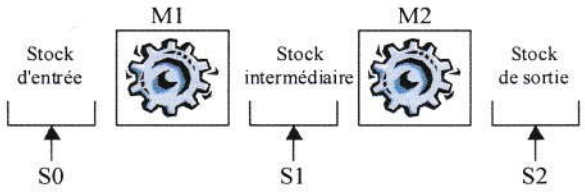
La figure 1 schématise en coupe un convoyeur qui reçoit des planches en provenance d'une machine à scier. Une planche est normalement formée d'une partie utile, qui sera amenée sur le tapis C2 et d'un rebut (ou croûte) qui sera dégagé sur le tapis C3. Pour cela, après avoir détecté l'arrivée d'une planche à l'aide du contact A, l'automatisme à concevoir doit mettre le tapis C1 en fonctionnement vers la droite (d). Il provoque le passage du rebut derrière le taquet T. Quand T touche le contact B, la partie utile de la planche est passée sur le tapis C2 et le tapis C1 repart vers la gauche (g) jusqu'à ce que T actionne le contact C. La croûte est ainsi transférée sur le tapis C3. Il peut arriver que la planche soit formée de deux parties utiles (sans croûte). Dans ce cas, l'opérateur appuie sur le bouton P (durée non spécifiée) pour déclencher un cycle différent : après avoir touché B, le taquet continue son mouvement jusqu'à déclencher une deuxième fois B de façon à transférer les deux parties utiles sur le tapis C2. Ensuite, il repart vers la gauche jusqu'au contact C. A partir de là, le système est prêt à recevoir une autre planche.



a) Décrire la commande du tapis C1 à l'aide d'un diagramme d'états réduit.
b) Proposer une mise en œuvre mémoire la plus économique possible.

6. Commande d'une chaîne d'usinage

La figure suivante représente une portion d'une chaîne d'usinage composée de deux postes de tranformation et de trois stocks. Les pièces usinées passent toutes par le premier poste puis par le second en passant transitoirement par les stocks. Un stock ne peut contenir qu'une pièce au plus. La valeur des capteurs S_i informe sur la présence ou non d'une pièce dans un stock. L'arrivée d'une pièce dans le stock d'entrée et la suppression d'une pièce dans le stock de sortie se font de façon manuelle.

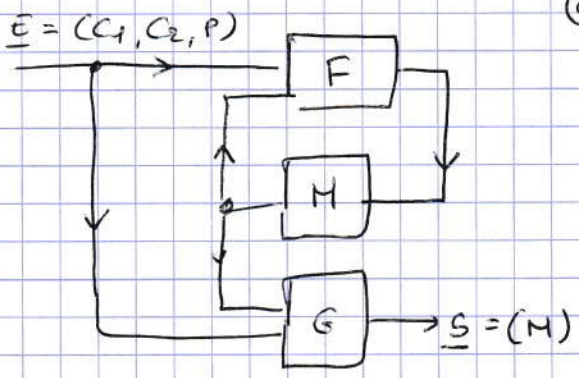


On désire concevoir un système de commande qui contrôle l'activité des postes grâce aux actionneurs M1 et M2 en fonction de l'état des stocks. Les contraintes à respecter sont les suivantes : *i*) une activité de transformation ne peut être amorcée que si une pièce est présente dans le stock d'entrée du poste considéré et si le stock de sortie de ce poste est vide ; *ii*) une activité s'achève sur un poste lorsque la pièce usinée arrive dans le stock de sortie de ce poste et *iii*) initialement, la chaîne de production est vide.

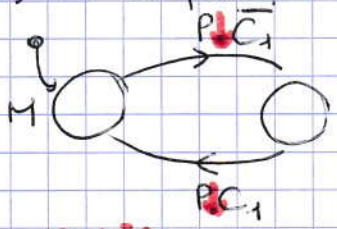
a) Modéliser le système de commande par un graphe d'états dans le cas où il n'y a qu'une seule pièce au maximum présente au même instant sur la chaîne.
b) Décrire le graphe d'états dans le cas où plusieurs pièces sont présentes.
c) Proposer une mise en œuvre en langage VHDL

TD SED

I Modélisation d'un convoyeur

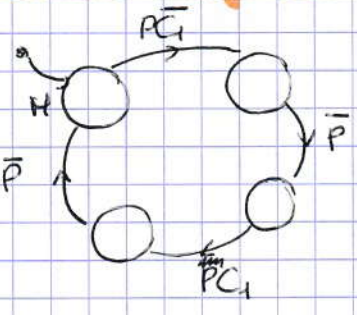


ⓐ Proposer une modélisation de cette commande lorsque seul le chargement est pris en compte ?

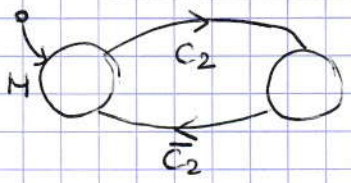


Modèle synchrone

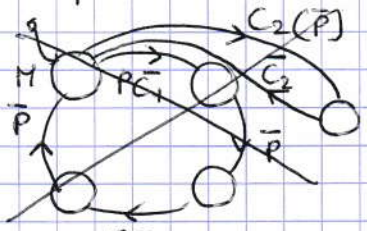
Modèle asynchrone

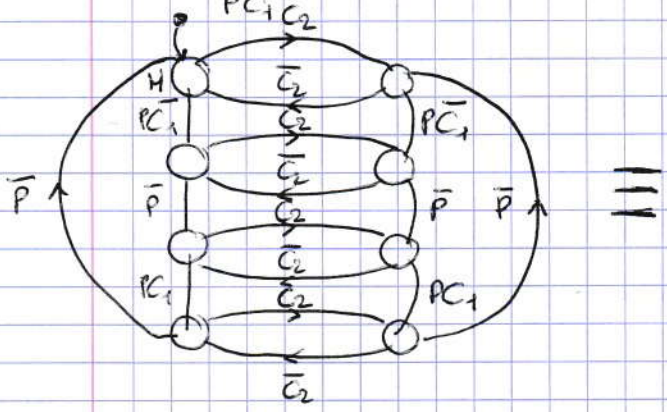


ⓑ Déchargement seul

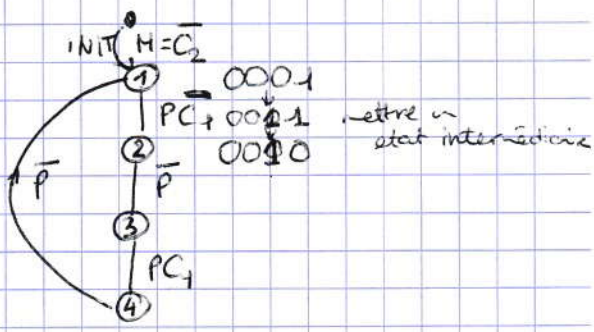


On peut "Mixer" nos 2 MEF





≡



ⓒ MEO par rebouclage direct

→ MEO 1 parmi N

On associe 1 VI à chaque état → l'état i est codé par la VI $y_i = SET_i + MAU_i = SET_i + y_i \overline{RESET_i}$

↑ mise à 1 ; ↑ maintient à 1

$$\begin{cases} y_0 = y_3 \cdot \bar{P} + y_0 \overline{\bar{y}_1} + INIT \\ y_1 = (y_0 P\bar{C}_1 + y_1 \bar{y}_2)\overline{INIT} \\ y_2 = (y_1 \bar{P} + y_2 y_3)\overline{INIT} \\ y_3 = (y_2 P\bar{C}_1 + y_3 y_0)\overline{INIT} \\ M = y_0 \bar{C}_2 \overline{INIT} \end{cases}$$

→ MEO par minimisation du nb de VI

4 états
Max constantes adjacentes = 2 } 2 variable introduites

$$\begin{cases} y_1 = (Y_1 Y_2 + Y_1 P)\overline{INIT} \\ y_2 = (\overline{Y_1} Y_2 + P\overline{C_1} Y_1 + Y_2)\overline{INIT} \\ M = Y_1 Y_2 C_2 \overline{INIT} \end{cases}$$

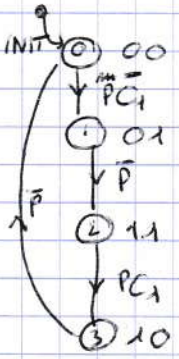
$Y_1 \backslash Y_2$	0	1
0	0	$\bar{P}$
1	P	1

y_1

$Y_1 \backslash Y_2$	0	1
0	$P\bar{C}_1$	1
1	0	$P+\bar{C}_1$

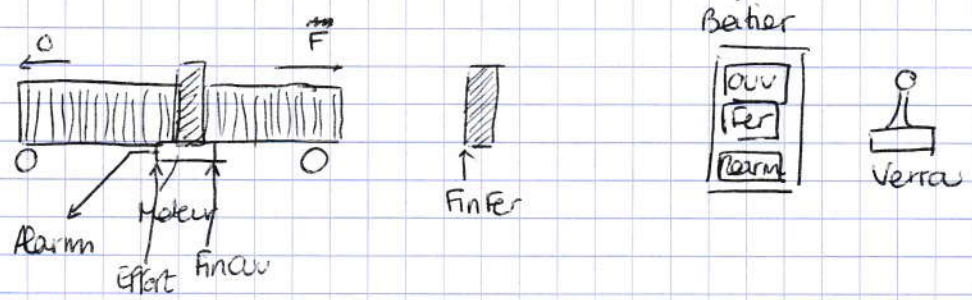
y_2

$(P+\bar{C}_1)\overline{INIT}$

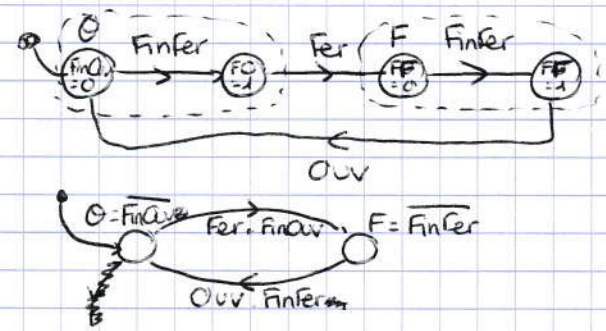


(2) Synthèse de la commande d'un portail à ouverture automatique

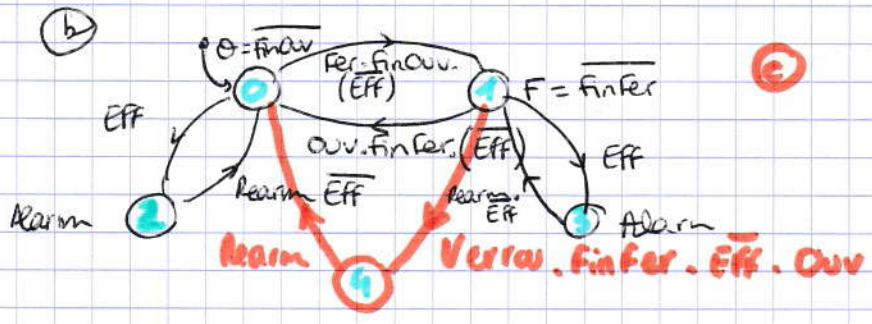
(a)



(a)



(b)

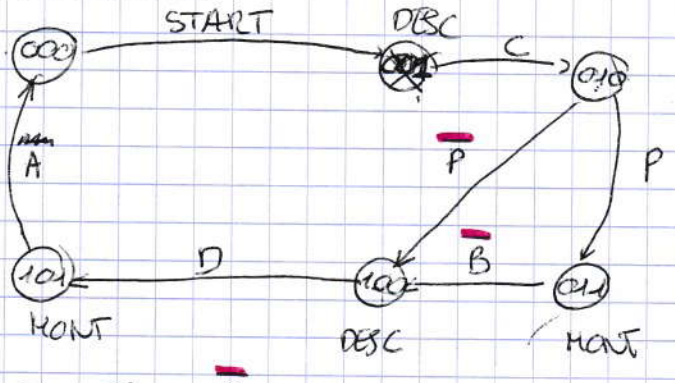


→ MEO 1/N

$$\begin{cases} y_0 = Y_1.Ouv.FinFer.\overline{Eff} + Y_2.Rear\_.\overline{Eff} + Y_4.Rear\_ + Y_0(Y_2+Y_1) \\ y_1 = \\ y_2 = \\ y_3 = \\ y_4 = \end{cases}$$

Cour SED

Modifier le graphe



Coder les états

Y_1 Y_2 Y_3	Y_{11} Y_{12} Y_{13}	Y_{21} Y_{22} Y_{23}	(Action sur état) Mont	Desc
0 0 0	0 0 0	0 0 1	0	0
0 0 1	0 0 1	0 1 0	0	1
0 1 0	1 0 0	0 1 1	0	0
0 1 1	1 0 0	0 1 1	1	0
1 0 0	1 0 0	1 0 1	0	1
1 0 1	1 0 1	0 0 0	1	0
	condition barré	condition non-barré		

VHDL

~~Archit~~
Entity CommandePerseuse is
~~A : std_logic~~
Port(
Start, A, B, C, D, P : in std_logic
Mont, Desc ; out std_logic
)
end entity;
Architecture CommandePerseuse is
begin(

desc when X="001" or X="100" else
mont when X="101" else
~~state change <= "000"~~

FX = "000"

Architecture --
--Déclaration des signaux internes
signal ~~Y1, Y2, Y3 : std_logic~~ ; Y1 present, Y2 present, Y1 suiv, Y2 suiv

Y1 <= ...
Y2 <= ...
Y3 <= ...

(Voir corrigé modèle)

Mont <=
Desc <=

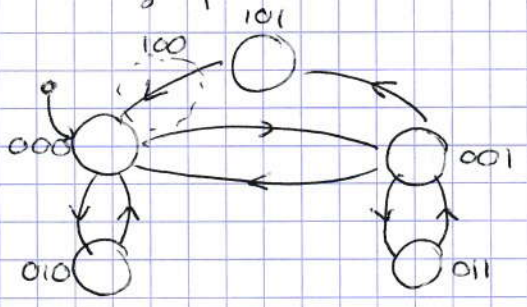
~~int ETAT, ETAT_S~~
int F, AU

TD SEQ (suite)

② Synthèse commande portail automatique (suite)

→ Codage par minimisation de variable introduite (VI)

$Y_1 Y_2 Y_3$



ⓓ Faire ~~ex~~ 3TKVI, 1 par chaque variables internes y_i

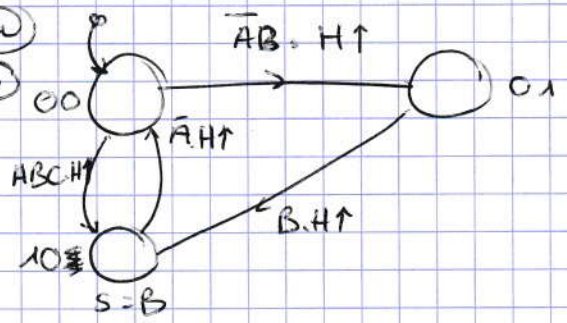
$$\begin{cases} y_1 = \\ y_2 = \\ y_3 = \end{cases}$$

Exercice ③ Gestion de porte charge

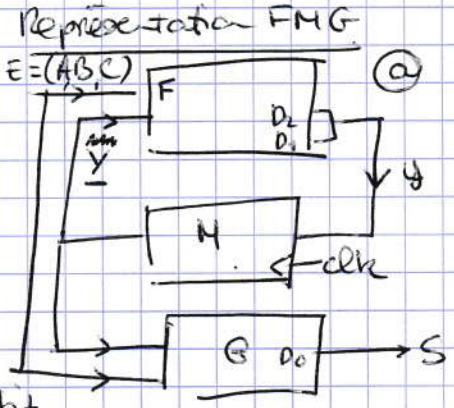
(voir photos)

Exercice ⓐ

ⓑ



Représentation FMG



ⓐ Taille mémoire utile $2^5 \times 3 = 96$ bit

ⓐ

Y_1	Y_2	A	B	C	y_1	y_2	S
A_4	A_3	A_2	A_1	A_0	D_2	D_1	D_0
0	0	1	0	–	0	1	0
0	0	1	1	1	1	0	0
0	0	0	0	–	0	0	0
0	0	1	1	0	0	0	0
0	1	–	1	–	1	0	0
0	1	–	0	–	0	1	0
1	0	0	1	–	0	0	1
1	0	0	0	–	0	0	0
1	0	1	1	–	1	0	1
1	0	1	0	–	1	0	0
1	1	–	–	–	1	1	0

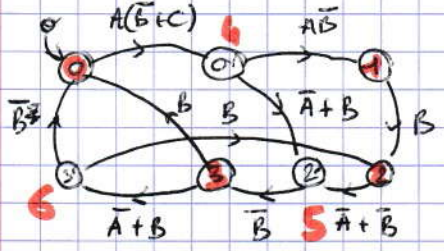
changement d'état ↑ ↓ maintient ↑

```
int Etat_p, Etat_s;
int P, Start, A, B, C, D;
int Mont, Desc
int main(void){
;
  Etat_p = 1;
  while(1){

    switch(Etat_p){
      case 1;
        Mont = A;
        if(A==1 && Start==1)
          Etat_p = 2;
      case 2:
        Desc = 1;
        if(C==1 && P==1)
          Etat_p = 3;
        if(C==1 && P==0)
```

TD SEQ (suite)

Exercice 4 (suite)

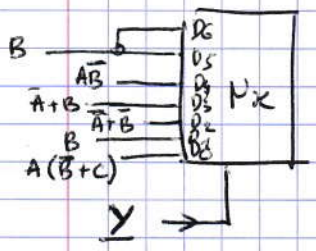


$A\bar{B} + ABC = A(\bar{B} + BC)$
$= A(\bar{B} + C)$

● numéro de l'état

On a 7 états ⇒ il faut au moins 3 entrées d'adresse sur la PROM

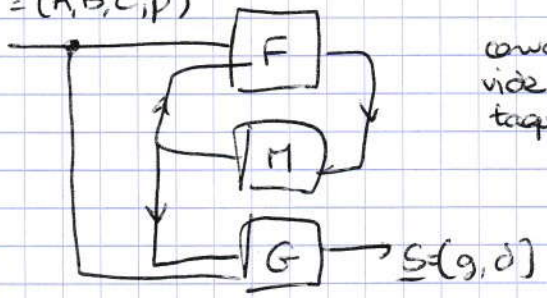
Il doit y avoir $3+3+1 = 7$ bits de données
Taille de la mémoire : $2^3 \times 7 = 56$ bits < 96

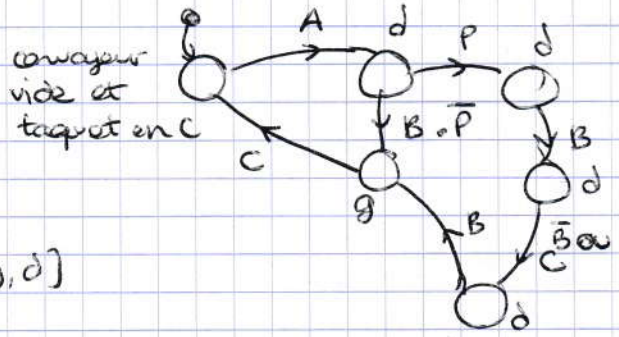


y			y_{D_1}			y_{D_0}			S
A_2	A_1	A_0	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	1	0	0	0	0	0	0
0	0	1	0	1	0	0	0	1	0
0	1	0	1	0	1	0	1	0	1
0	1	1	1	1	0	0	1	1	0
1	0	0	0	0	1	0	1	0	0
1	0	1	0	0	0	0	1	1	0
1	1	0	0	1	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0

Exercice 8

$\underline{E} = (A, B, C, P)$





```
library ieee;
use ieee.std_logic_1164.all;

Entity ChaineUsinage is
   Port
(S2, S1, S0, INIT : In std_logic;
  M2,M1    : Out std_logic);
End ChaineUsinage;

Architecture RealisationDirecte of ChaineUsinage is
-- Déclaration des états
Type Etat is (Etat0, Etat1, Etat2, Etat3, Etat4, Etat5,
Etat6, Etat7);
Signal EtatPresent, EtatSuivant : Etat;
Begin
-- Description du Bloc F
Process (S2,S1,S0)
Begin
Case EtatPresent is
   When Etat0 =>
    If S0='1' then EtatSuivant <= Etat1;
    else EtatSuivant <= EtatPresent;
    End If;
   When Etat1 =>
    If S1='1' then EtatSuivant <= Etat6;
    else EtatSuivant <= EtatPresent;
    End If;
   When Etat2 =>
    If S2='0' then EtatSuivant <= Etat1;
    Elsif S1='1' then EtatSuivant <= Etat3;
    else EtatSuivant <= EtatPresent;
    End If;
   When Etat3 =>
    If S2='0' then EtatSuivant <= Etat6;
    else EtatSuivant <= EtatPresent;
    End If;
   When Etat4 =>
    if S2='1' then EtatSuivnat <= Etat3;
    else EtatSuivant <= EtatPresent;
    End If;
  When Etat5 =>
   If S1='1'  then EtatSuivant <= Etat4;
   Elsif S2='1'  then EtatSuivant <= Etat2;
   else EtatSuivant <= EtatPresent;
   End If;
  When Etat6 =>
   If S2='1' then EtatSuivant <= Etat7;
   ElsIf S0='1' and S2='0' then EtatSuivant <= Etat5;
   else EtatSuivant <= EtatPresent;
   End If;
  When Etat7 =>
   If S2='0' then EtatSuivant <= Etat0;
   ElsIf S0='1' and S2='1' then EtatSuivant <= Etat2;
   else EtatSuivant <= EtatPresent;
   End If;
  When others => EtatSuivant <= EtatPresent;
End Case;
End Process;

-- Description du bloc M
Process (H, Init)
Begin
If (Init='1') then
  EtatPresent <= Etat0; -- état initial
ElsIf (H'event and H='1') then -- front montant
  EtatPresent <= EtatSuivant; -- changement d'état
End If;
End Process;

-- Description du Bloc G
With EtatPresent select
  M1 <= (not init) when Etat1|Etat2|Etat5,
        0 when others;
With EtatPresent select
  M2 <= (not init) when Etat4|Etat5|Etat6,
        0 when others;
End RealisationDirecte;
```