MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOULOUD MAMMERI, TIZI-OUZOU

FACULTE DE GENIE ELECTRIQUE ET DE L'INFORMATIQUE
DEPARTEMENT D'ELECTRONIQUE

Mémoire de fin d'études

Présenté en vue de l'obtention du Diplôme d'Ingénieur d'Etat en Electronique

Options: communication et contrôle

Thème:

Année universitaire **2010/2011**.

Remerciements

Nous tenons à remercier dieu tout puissant de nous avoir donné la force et la patience pour mener ce travail à tèrme.

Nous tenons à remercier les promoteurs Mr BENNÂMEN.K et Mr KARA.R qui nous ont encadré et conseillé tout au long de notre travail.

Nous tenons a exprimé notre gratitude à toute personne ayant contribué de près ou de loin à la réalisation de notre travail.

Egalement nous tenons à remercier tous les enseignants qui ont contribué dans notre formation.

Bien entendu, nous remercions énormément nos parents, nos frères et sœurs qui nous ont soutenue et encouragé durant la réalisation de ce projet, leurs nombreux sacrifices nous ont bien aidé tout au long de nos études.

Enfin, un grand remerciement aux membres du jury et à M^{m} la présidente qui ont accepté de juger notre travail.

Dédicaces

Je dédie ce modeste travail à :

- *Mes chers parents et beaux parents.*
- *Mon mari qui ma beaucoup aidé.*
- *A Mes frères HILLAL et LYES*
- *A mes sœurs HAYET et ZAHIRA*
- *Ma sœur BAHIA et son mari RABAH et ces enfants LYES et ALLILO*
- *A ma sœur FADHILA et son mari MAHYEDDINE et ces enfants CELYAN et LOLA*
- *A toute ma famille sans exception*
- *Toutes mes amies.*
- *Toute la promotion 2011*

GHANIA ✍

Dédicaces

Je dédie ce modeste travail à :

- *Mes chers parents et beaux parents.*
- *Mon cher mari qui ma beaucoup aidé.*
- *Mon petit ange AYOUB.*
- *Mes sœurs NADIA, LYNDA et la petite THANINA.*
- *Mes sœurs SALIHA, FAZIA, FATIMA et leurs maris ainsi que leurs enfants*
- *Mes frères et leurs maries ainsi que leurs enfants.*
- *A toutes mes amies*

SABRINA ✍

NOMENCLATURE

Sigles utilisés :

FSM **:** Machine à états finis (*finite state machine)*

FSA : finite state automata

PLD : circuits logiques programmables (programmable logic device).

PAL : réseau logique programmable (Programmable Array Logic)

E.P.L.D : circuit logique programmable et effaçable (Erasable Programmable Logic Device)

UVPROM: sont des mémoires programmables par l'utilisateur. Elles sont effaçables en les mettant dans une chambre à ultraviolet (Ultra Violet Programmable Read Only Memory).

EEPROM: sont effaçables et programmables par l'utilisateur c'est-à-dire électriquement (Electrically Erasable Programmable Read Only Memory)

GAL : réseau logique générique ou encore (Generic Array Logic)

CPLD : circuit logique programmable complexe (Complex Programmable Logic Device)

LCA : réseau de cellules logiques (Logic Cell Array)

FPGA: Field Programmable Gate Array ou encore réseau de cellule logique programmable

JEDEC: Format de fichier de programmation des circuits logiques (image des fusibles à

Griller (Joint Electron Devices Engineering Council)

AFCET: Association Française pour le Cybernétique Economique et Technologie

RDP : Réseau De Petri

AMD : (Advanced Micro Devices), est une société technologique innovante

TEXAS Instruments (TI): est une entreprise d'électronique basée à Dalas, renommée dans le Domain des composants électronique passifs et des semi-conducteurs elle a notamment inventé de circuit integret, ouvrant ainsi la voie au développement de l'informatique tel qu'on le connait actuellement.

Introduction ……..…… 1

Chapitre. I. Les systèmes séquentiels ……..……. 2

I.1. Définition ….. …….....…….... 2
I.2. Systèmes séquentiels synchrones……... 2
I.2.1. Modèle des systèmes séquentiels synchrones……. 2
I.2.2. Synthèse de circuits séquentiels synchrones……..……..... 3
I.2.2.1. Description verbale du circuit séquentiel à réaliser…….. 3
I.2.2.2. Construction du diagramme des états……..... 4
I.2.2.3. Construction de la table des états……. 4
I.2.2.4. Réduction du nombre d'états……..... 5
I.2.2.5. Codage des variables d'état…….. 5
I.2.2.6. Réalisation du circuit……..... 6
I.3. Les systèmes séquentiels asynchrones……... 8
I.3.1. Introduction……. 8
I.3.2. Synthèse de circuits séquentiels asynchrones…….. 8
I.3.2.1. Méthode de synthèse……..... 8
I.3.2.1.1. Modélisation du cahier des charges……... 9
I.3.2.1.2. Graphe des états……..... 9
I.3.2.1.3. Table des états……... 9
I.3.2.1.4. Réalisation du circuit……..... 11

Chapitre. II. Notations sur les machines à états finis, réseaux de Pétri et Grafset……. 12

II.1. Les machines à états finis……..... 12
II.1.1. Introduction…… 12
II.1.2. Définition……... 12
II.1.3. Le graphe d'état……..... 13
II.1.3.1. Représentation graphique d'une machine à état fini…….. 13
II.1.3.2. Vérification de la représentation à l'aide de quelques règles simples..... 17

II.2. Réseaux de Pétri……. 20

II.2.1. Introduction……..... 20
II.2.2. Définition……...…… 20
II.2.3. Notation……. 21
II.2.4. Règle de validation et de franchissement d'une transition……. 22
II.2.5. Étude du marquage d'un réseau……...…….. 22
II.2.5.1. Graphe de marquage…… 23
II.2.6. Réseaux de Pétri particuliers……. 23
II.2.6.1. Graphe d'état……..... 23
II.2.6.2. Graphe d'événement…….. 24
II.2.6.3. Réseau de Pétri sans conflit……..... 24
II.2.6.4. Réseau de Pétri choix libre…….. 24
II.2.6.5. Réseau de Pétri simple……..... 25
II.2.6.6. Réseau de Pétri pur……. 25
II.2.6.7. Réseau de Pétri généralisé…….……. 26
II.2.6.8. Réseau de Pétri capacité……. 26
II.2.6.9. Réseau de Pétri a priorité……..…….…… 27
II.2.7. Propriétés……..... 27
II.2.7.1. Réseau de Pétri vivant.……..... ……. 27
II.2.7.2. Réseau de Pétri non vivant…….. 27
II.2.7.3. Réseau de Pétri sauf ……..……. 27
II.2.7.4. Réseau de Pétri - k- borné……..... 28
II.2.7.5. Réseau de Pétri non bornée……. 28
II.2.7.6. Réseau de Pétri marqué réinitialisable ……... 28
II.2.8. Présentation matricielle…… …… 28
II.2.9. Synthèse d'un automate avec les réseaux de Pétri……..... 30

II.3. Le Grafcet…….…… 31

II.3.1. Introduction…… 31
II.3.2. Structure graphique…… 31
II.3.2.1. Une liaison……..... 31
II.3.2.2. Une étape……..... 32

II.3.2.3. Une transition…….. 32
II.3.3. Règles d'évolutions……..... 33
II.3.4. Configurations courantes……..... 35
II.3.4.1. Convergence en OU…… 35
II.3.4.2. divergence en OU…… 35
II.3.4.3. divergence en ET……. 35
II.3.4.4. Convergence en ET……..... 35
II.3.5. Structures de base d'un grafcet……..... 35
II.3.5.1.séquence unique……..…… 36
II.3.5.2.Saut d'étapes……...…… 36
II.3.5.3.Reprise de séquence……. 36
II.3.6.Temporisation…….. 36
II.3.7. Mise en œuvre du grafcet……..... 36
II.3.7.1. Réalisation par câblage……...……. 36
II.3.7.1.1.Grafcet linéaire……..... 36
II.3.7.1.1.1.Règles d'implantation…… 36
II.3.7.1.2.Grafcet non linéaire……..... 38
II.3.7.1.2.1. Divergence simple en ET……..... 38
II.3.7.Synthèse d'un automate avec les réseaux de Pétri……..... 39
II.3.8. conclusion……..... 39

Chapitre. III. Gestion d'un carrefour par circuit logique programmable... 40

III.1. Introduction…….. 40
III.2. Structure de base d'un P.L.D……... 40
III.3.Liste de toutes les familles de P.L.D……..... 42
III.3.1. Les P.A.Ls……..... 42
III.3.1.1. Les différentes structures……... 43
III.3.1.1.1. Structure générale……... 43
III.3.1.1.2. Référence et nomenclature des P.A.Ls…… 44
III.3.1.2.1. PALs, AMD……...…… 44
III.3.1.2.2. PALs TEXAS instrument……..…….. 44
III.3.2. Les G.A.Ls……..... 44
III.3.2.1. Référence des G.A.Ls……..……..... 45

III.3.3. Les C.P.L.Ds……..……... 45
III.3.3.1. Structure générale d'un C.P.L.D……..……..... 45
III.3.4. Les L.C.As et F.P.G.As à anti-fusible……... 46
III.4. Programmation des P.L.Ds……..……..... 46
III..5. Application… ……..... 48
III.5.1.Position de problème……..... 48
III.5.2.Le fonctionnement……..... 49
III.5.3.Table et diagramme des états……..... 49
III.5.3.1.réalisation de la table des états ……..... 49
III.5.3.2.le diagramme des états……..... 50
III.5.4.programmation sous Win CUPL (machine à états)……...…… 51

Conclusion…….. 57

Introduction

L'évolution de l'industrie des circuits intégrés à connue un essor gigantesque durant cette dernière décennie. Elle à été tellement fulgurante qu'il est maintenant possible d'intégrer plusieurs systèmes complexes sur une seule puce. Cette évolution a été dictée et motivée par le besoin de réaliser des systèmes de plus en plus performants, compacts, légers et économes en énergie. Cependant, la gestion de la complexité de ces systèmes avec les outils d'aides a la conception traditionnelle devient une tache pénible et couteuse, voir impossible, alors, pour mieux gérer cette complexité toute en gardant les coûts de développement dans les limites du raisonnable, d'autres outils d'aides à la conception, dits outils de synthèses, ont vu le jour dès les débuts des années 1990.

Dans ce projet nous nous intéressons au passage de la machine à état aux réseaux de pétri et à l'intégration du projet dans un P.L.D en utilisant un logiciel dédié WinCUPL(ATMEL).

Pour munir à bien notre projet, on a scindé trois chapitres :

Le premier chapitre est divisé en deux parties : la première sera consacrée aux systèmes séquentiels synchrones et la deuxième partie aux systèmes séquentiels asynchrones, dans ces deux parties on s'intéressera à la synthèse.

Le deuxième portera trois méthodes de synthèse des circuits séquentiels, la première méthode est celle des machines à états finis et la deuxième c'est les réseaux de Pétri et en fin le grafcet.

Le troisième chapitre sera consacré à l'étude des circuits logique programmable, en particulier les PALs et les GALs. Ainsi l'implémentation d'un feu de carrefour sur un circuit programmable GAL16V8.

Chapitre. I. Les systèmes séquentiels.

I.1. Définition :

Les systèmes séquentiels peuvent être différenciés en fonction de leur mode de fonctionnement qui peut être synchrone ou asynchrone. Dans le mode synchrone, les éléments de mémorisation sont des bascules. Les modifications d'état du système ne peuvent donc intervenir qu'à des instants très précis déterminés par des signaux d'horloge. Par contre, dans le mode asynchrone, la fonction de mémorisation est réalisée par de simples boucles (rétroaction). L'évolution des états ne dépond donc que des modifications intervenant sur les entrées E_i de la machine. Comme illustré sur les figures ci dessous.

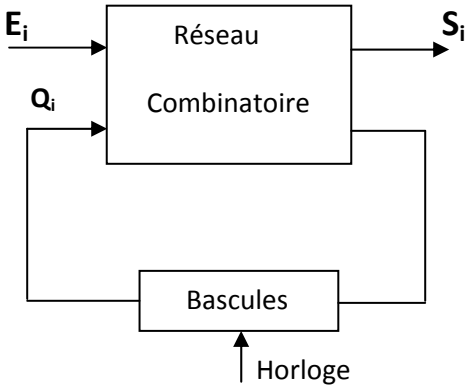


Fig. I.1. Système séquentiel synchrone.

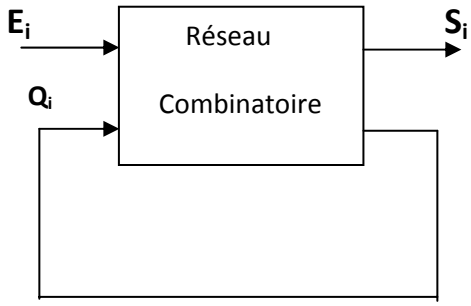


Fig. I.2. Système séquentiel asynchrone.

I.2. Systèmes séquentiels synchrones :

I.2.1. Modèles des systèmes séquentiels synchrones :

Dans un système séquentiel, la présence d'un même vecteur d'entrée n'entraîne pas nécessairement l'apparition d'une même combinaison sur les sorties. La combinaison de sortie dépend également des données internes caractérisant l'état du système. Un circuit séquentiel contient (r) éléments de mémoire élémentaire q_1, q_2,….q_r qui caractérise son état interne.

Les vecteurs E_i (entrées), S_i (sorties) et Q_i (états) évoluent à des instants biens déterminés, dans le système séquentiel synchrone on distingue deux modèles de machines:

-Modèle de Mealy

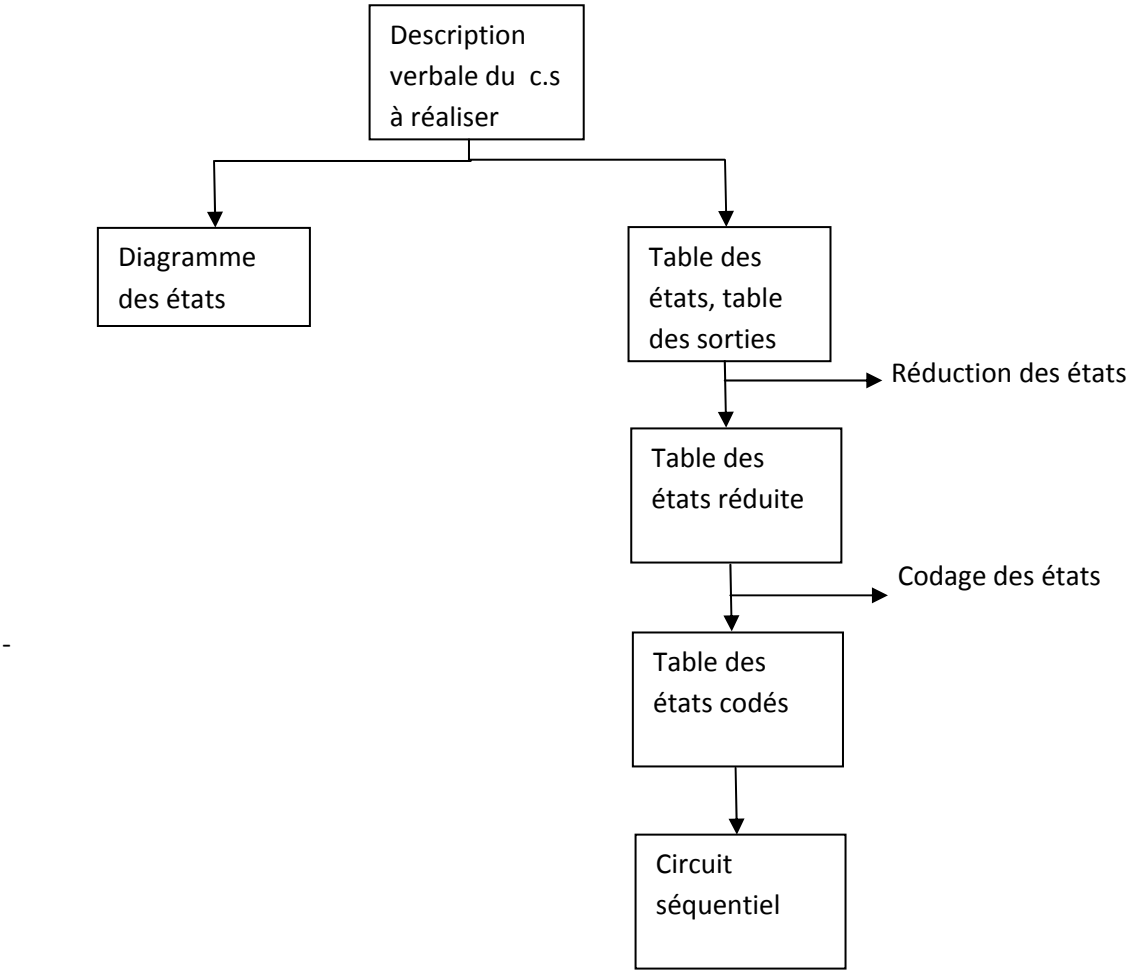
-Modèle de Moore

Mais quelque soit le type de machine (Moore ou Mealy) l'état suivant du système (Q_i^+) dépond de l'état actuel (Q_i) et des entrées (E_i). La différence entre ces deux machines (Mealy et Moore)

n'intervient que sur les sorties. Dans une machine de Mealy, les sorties (S_i) dépondent de l'état et des entrées (E_i), par contre dans une machine de Moore, les sorties (S_i) ne dépendent que de l'état (Q_i).

I.2.2. Synthèse de circuits séquentiels synchrones :

La procédure de synthèse des circuits séquentiels synchrones est illustrée par la figure suivante :



-

Fig. I.3. Les étapes de réalisation d'un circuit séquentiel synchrone

I.2.2.1. Description verbale du circuit séquentiel à réaliser :

Exemple : on veut réaliser un détecteur de la séquence (1010), le circuit a une seule entrée et une seule sortie, la sortie sera mise à 1 à chaque fois que la séquence (1010) est détectée. L'arrivée des séquences sur l'entrée (e) se fait au rythme des impulsions d'horloge.

I.2.2.2. Construction du diagramme des états :

Ce diagramme traduit le fonctionnement du cahier des charges, le circuit est supposé dans l'état initial (q_0), il peut recevoir sur son entrée (e) 0 ou 1 pour chacune de ces entrées le circuit évolu vers un autre état. L'évolution d'un état vers un autre état est représentée sur le diagramme des états par un arc reliant l'état de départ (état présent) à l'état d'arrivée (état suivant), Sur l'arc est spécifiée la valeur de l'entrée (e) et la sortie (s) de la manière suivante e/s.

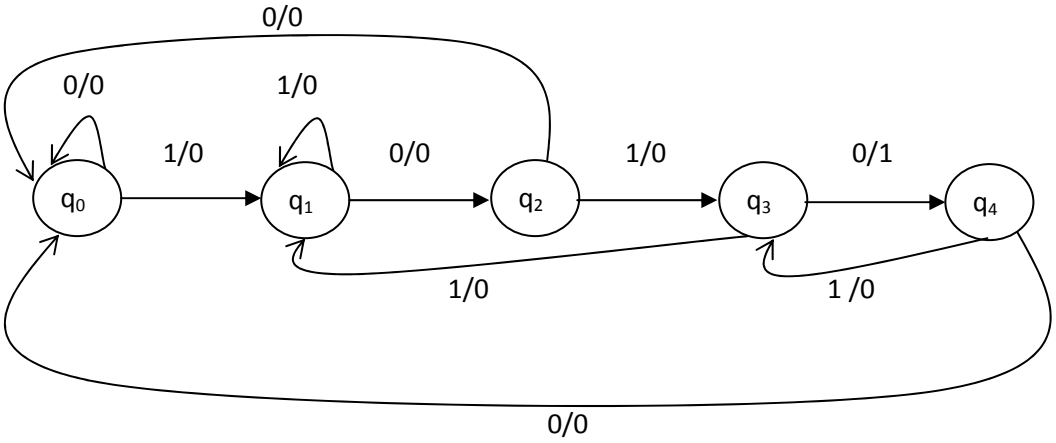


Fig. I.4. Diagramme des états du détecteur de séquences selon la machine de Mealy.

I.2.2.3. Construction de la table des états :

Elle se fait à partir du diagramme des états, dans la 1ère colonne sont inscrits les états présents, dans les autres colonnes les états suivants, et la sortie (s) selon la valeur de l'entrée (e).

e / q	0	1
q_0	q_0,0	q_1,0
q_1	q_2,0	q_1,0
q_2	q_0,0	q_3,0
q_3	q_4,1	q_1,0
q_4	q_0,0	q_3,0

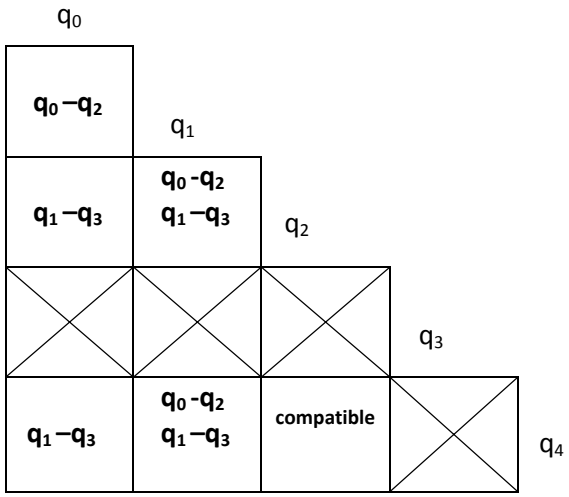
Tab. I.1. La table des états

I.2.2.4. Réduction du nombre d'états :

Deux états de la table sont compatibles si elles vérifient les deux conditions suivantes :

a. Elles sont compatibles en sortie.
b. Elles engendrent une paire d'états compatibles

On commence par mettre une croix dans toute case correspondant à deux états incompatibles en sorties, après il faut vérifier si les paires d'états restantes entrainent des paires d'états compatibles.



Etats	Codages binaires
q_0 ⟶ a	00
q_1 ⟶ b	01
q_2, q_4 ⟶ c	11
q_3 ⟶ d	10

Tab. I.2. La table des états réduite

I.2.2.5. Codage des variables d'état :

Coder une table des états revient à faire correspondre à chaque état de la table une combinaison binaire unique. Pour cela on applique les règles suivantes :

a. Les états présents qui ont des états suivants identiques dans chaque colonne de la table des états réduite, doivent prendre des codages adjacents.
b. Les états présents qui ont des états suivants identiques déjà adjacents dans déférentes colonnes de la table des états réduite, doivent prendre des codages adjacents.
c. Les états présents qui ont des états suivants identiques dans quelques colonnes de la table des états réduite, doivent prendre des codages adjacents mais pas toutes.
d. Les états suivants d'un état présent doivent prendre des codages adjacents.
e. Les codages doivent être faits de façon à simplifier les expressions de sortie

e / q	0	1
a	a,0	b,0
b	c,0	b,0
c	a,0	d,0
d	c,1	b,0

Tab. I.3. La table des états codés

	e / q_1q_0 0	1
a	00	00	01
b	01	11	01
c	11	00	10
d	10	11	01

tab. I.4. La table des états suivants

e / q_1q_0	0	1
00	0	0
01	0	0
11	1	0
10	0	0

Tab. I.5. La table d'état de la sortie (s)

I.2.2.6. Réalisation du circuit :

On peut réaliser ce circuit à base de bascules JK :

Rappelant la table de transition de la bascule JK :

	J	K
S_0	0	X
S_1	X	0
T_0	X	1
T_1	1	X

Tab. I.6. Table de transition de la bascule JK

e / q_1q_0	0	1
00	0	0
01	1	0
11	0	1
10	1	0

Tab. I.7. Table d'état future (Q_1^+)

e / q_1q_0	0	1
00	0	0
01	1	0
11	X	X
10	X	X

Tab. I.8. Table (J_1)

e / q_1q_0	0	1
00	X	x
01	X	X
11	1	0
10	0	1

Tab. I.9. Table de (K_1)

e / q_1q_0	0	1
00	0	1
01	1	1
11	0	0
10	1	1

Tab. I.10. Table de (Q_0^+)

e / q_1q_0	0	1
00	0	1
01	X	X
11	X	X
10	1	1

Tab. I.11. Table de (J_0)

Enfin, on obtiendra les équations finales suivantes :

e / q_1q_0	0	1
00	X	X
01	0	0
11	1	1
10	X	X

Tab. I.12. Table de (K_0)

$$\begin{cases} J_1 = \bar{e}\, q_0 \\ K_1 = e \oplus q_0 \\ J_0 = e + q_1 \\ K_0 = q_1 \\ S = \bar{e}\, q_0 q_1 \end{cases}$$

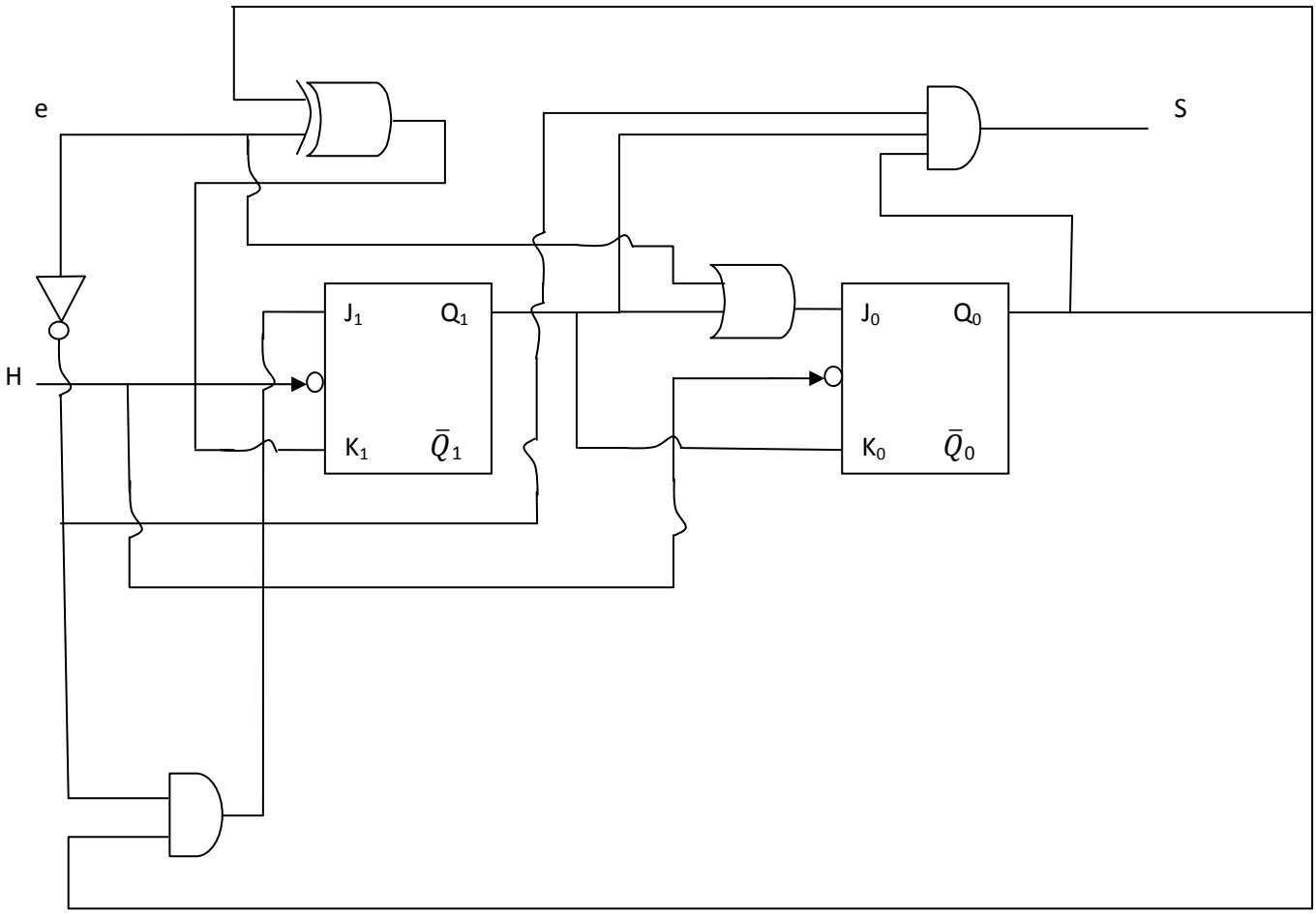


Fig. I.5. Circuit réalisé avec des bascules JK

I.3. Les systèmes séquentiels asynchrones :

I.3. 1. Introduction :

Dans quelques situations, les entrées du système à réaliser ne sont pas gérées par une horloge, sont sujettes à des modifications à des instants quelconques, dans ce cas on parle de systèmes séquentiels asynchrones et la synthèse de ces circuits est très similaire à celle des circuits synchrones. Il existe certaines différences dans la conception, notamment au niveau du codage de la table des états, dues essentiellement à l'absence d'un signal d'horloge.

I.3. 2. Synthèse des circuits séquentiels asynchrones :

I.3. 2.1. Méthode de synthèse :

La synthèse des circuits séquentiels asynchrones proposée a pour but de minimiser le nombre de fonctions du réseau combinatoire et par conséquent, le nombre de portes du circuit.

I.3. 2.1. 1. Modélisation du cahier des charges :

Tout comme les systèmes séquentiels synchrones nous appuierons ici, sur un exemple simple, pour développer la procédure de synthèse d'un circuit séquentiel asynchrone.

Exemple : on désire commander l'allumage ou l'extinction d'une lampe L à l'aide de deux boutons poussoirs (m) et (a). Initialement, la lampe (L) est éteinte et les deux boutons poussoirs relâchés. Son allumage est obtenu par action sur le bouton poussoir (m), le bouton poussoir (a) étant relâché. Son extinction est obtenue par action sur le bouton poussoir (a).

I.3. 2.1. 2. Graphe des états :

Les nœuds du graphe représentent les états du système et les arcs orientés, la possibilité de passer d'un état à un autre. Chaque état étant stable tant qu' il n'y a pas de variation d'entrée. Le graphe d'état représentant le cahier des charges précédent est représenté sur la figure (I.6), ce diagramme des états est un exemple de modèle de Moore.

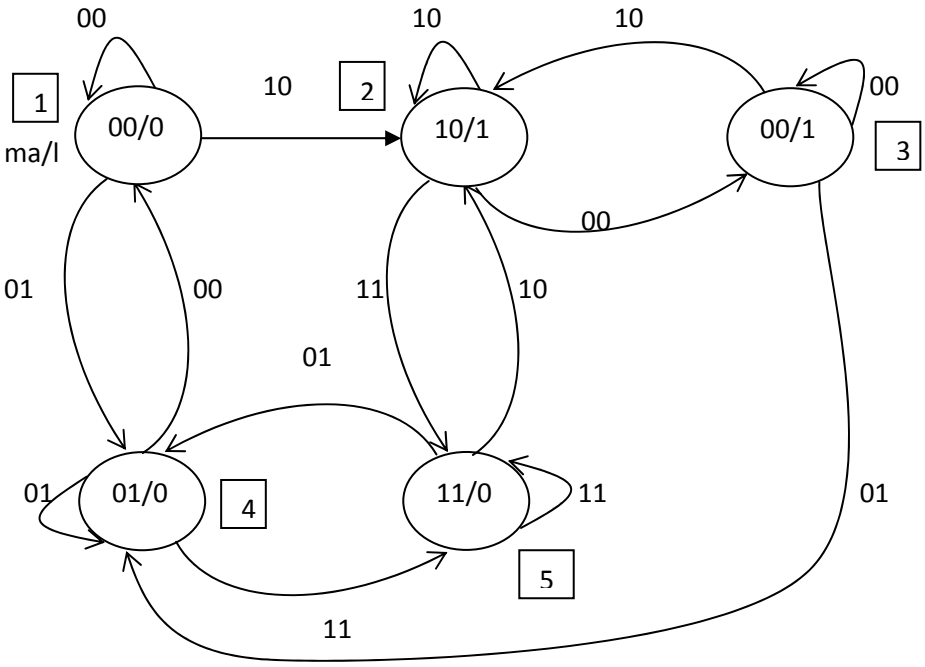


Fig. I.6. Diagramme d'état d'allumage et d'extinction d'une lampe selon la machine de Moore

I.3. 2.1. 3. Table des états :

Le cahier des charges d'un système peut également être modélisé sous forme tabulaire, cette table est directement déductible du graphe d'état. La principale caractéristique est qu'elle ne comporte qu'un seul état stable par ligne.

Puisque le diagramme représente 5 états, donc la table aussi va contenir 5 états.

La table des paires, la table des états réduite et la table des états codés sont données par les figures ci dessous, nous formons deux ensembles d'états compatibles. Il s'agit de l'ensemble (1, 4,5) représenté par q_0 dans la table des états réduite et de l'ensemble (2,3) représenté par q_1. On attribue 0 pour q_0 et 1 pour q_1, on obtient la table des états codée.

ma / q	00	01	11	10	L
1	1	4	X	2	0
2	3	X	5	2	1
3	3	4	X	2	1
4	1	4	3	X	0
5	X	4	5	2	0

Tab. I.13. La table des états

ma / q	00	*01*	*11*	*10*
q_0(1,4,5)	(1)	(4)	(5)	*2*
q_1(2,3)	(3)	*4*	*5*	(2)

Tab. I.14.La table des états réduite (q^+)

ma / q	00	01	11	10
0	(0)	(0)	(0)	1
1	(1)	0	0	(1)

Tab. I.15. La table des états codée

ma / q	00	01	11	10
0	0	0	0	1
1	1	0	0	1

Tab. I.16. Table d'excitation secondaire (y^+)

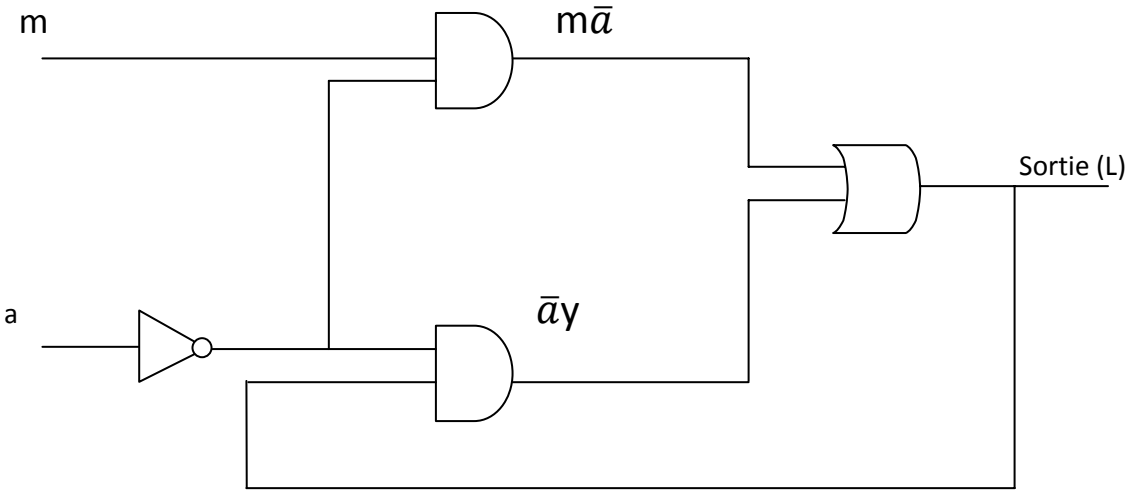
La table de sortie : pour les circuits séquentiels asynchrones, les soties ne sont définies que dans les états stables. Pour déterminer les expressions des sorties, nous commençons par remplir la table de sortie avec les valeurs des sorties correspondant aux états stables, puis nous déterminons les valeurs des sorties correspondant aux états transitoires.

ma / q	00	01	11	10
0	0	0	0	X
1	1	X	X	1

Tab. I.17. Table de sortie (L)

$q^+ = m\bar{a} + \bar{a}q = L$

I.3. 2.1. 4. Réalisation du circuit :



Chapitre. II. Notions sur les machines à états finis, les réseaux de Pétri et Grafcet.

II.1. Les machines à états finis :

II.1.1. Introduction :

Une machine à états finis **(**en anglais, FSM pour finite state machine**)** ou automate à états finis **(**FSA pour finite state automata**)** utilisée dans l'étude du calcul et des langues qui possède une quantité finie et constante de mémoire (les états). Une telle machine peut être conceptualiser comme un graphe orienté. Il y a un nombre fini d'états, un ou plusieurs états. Les états qui ne possèdent pas de transitions sont appelées états finaux. La transition suivie est déterminée par la valeur donnée en entrée. Les machines à états finis sont étudiées dans la théorie des automates, un sous-domaine de l'informatique théorique.

II.1.2. Définition :

Les machines à états sont des circuits de logique séquentielle servant exclusivement à générer des Signaux de commande. Il existe en effet 2 grands types de signaux en électronique :

Signaux à traiter : les données

Signaux pilotant le traitement : les commandes

Cette classification des signaux se retrouve au niveau des architectures des systèmes électroniques qu'on peut schématiser comme dans la figure (II.1.1) Où la partie contrôle, générant les commandes, est dissociée de la partie opérative, traitant les données. Les 2 parties sont toujours réaliser en logique séquentielle et dans une très grande majorité des cas en logique séquentielle synchrone.

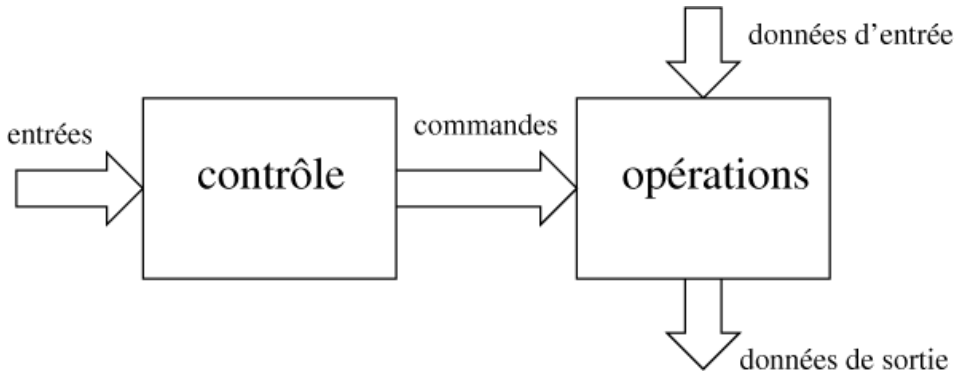


Fig. II.1.1. Architecture générique d'un circuit électronique

Pour la logique séquentielle synchrone, il existe 2 signaux de commandes importants :

a- L'horloge : pour le déroulement des séquences.

b- Le Reset : pour l'initialisation du système.

La machine à état représente la partie contrôle, c'est à dire le cerveau du système électronique et la partie opérative, les jambes.

Il existe beaucoup de déclinaisons de cette architecture, des plus compliquées comme les microprocesseurs qui ont plusieurs machines à états et plusieurs parties opératives, des plus simples mais tout aussi importantes comme les contrôleurs d'ascenseurs ou de machine à café.

Les états de la machine à états représentent toutes les valeurs que peuvent prendre les variables internes du circuit de logique séquentielle.

Le schéma de la machine à états générique est représenté en figure (II.1.2)

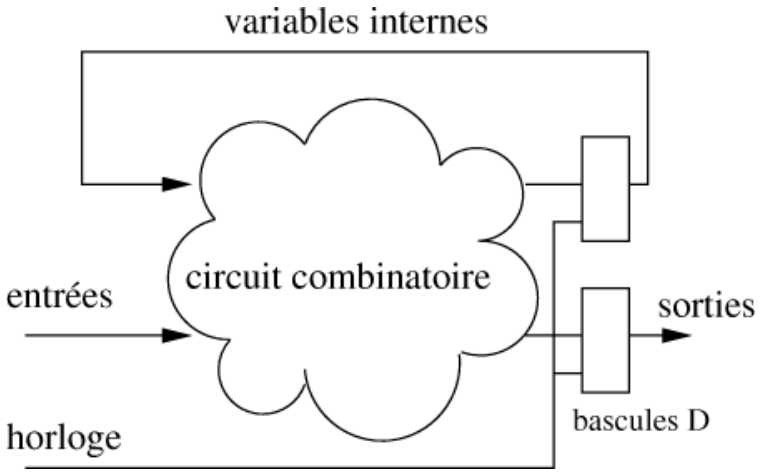


Fig. II.1.2. Schéma d'une machine à état

II.1.3. Le graphe d'état :

II.1.3.1. Représentation graphique d'une machine à état fini :

Dans une machine à état donnée, la loi d'évolution de l'état n'est évidemment pas aléatoire. Ces lois sont soigneusement choisies par le concepteur de la machine. Le graphe d'états est l'un des outils les plus utilisés pour la spécification de la machine à états (entrées, sorties, fonctionnement souhaité). Le graphe d'états, comme son nom l'indique, représente graphiquement les états d'une machine à états. Chaque état est dessiné sous la forme d'une bulle contenant son nom. On comprend immédiatement que cet outil ne sera pas d'un grand secours lorsque le nombre

d'états de la machine dépassera quelques dizaines. Prenons l'exemple d'une machine à laver où on considère 5 états comme illustré dans la figure (II.1.3)

On complète le graphe en figurant les transitions possibles par des flèches entre les états. On appelle état source l'état de départ d'une transition et état destination l'état d'arriver. La transition T0 a pri Prélavage pour état source et Lavage pour état destination. Certaines transitions ont le même état pour source et pour destination. Cela signifie que la machine peut rester dans le même état pendant un certain temps. La transition T1 est de cette sorte comme illustré dans la figure(II.1.4)

Muni de toutes les transitions possibles comme représenté dans la figure (II.1.5) Le graphe constitue une représentation assez dense de l'évolution possible de la machine au cours du temps.

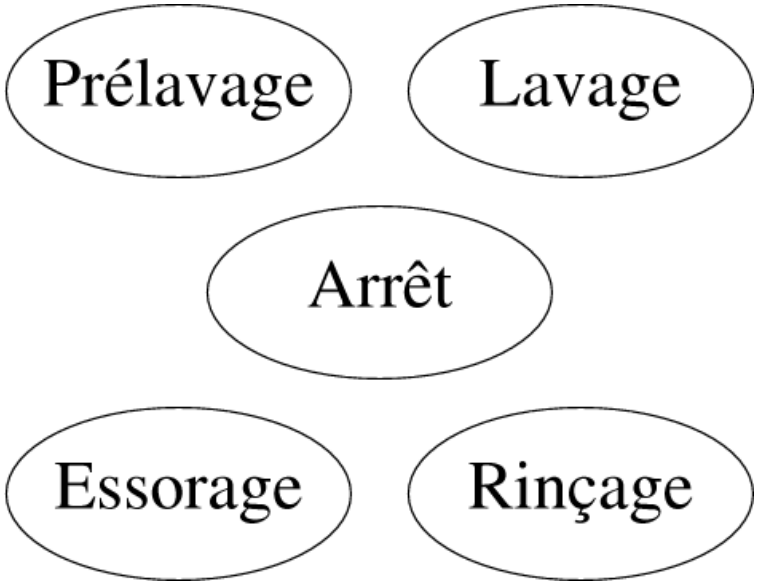


Fig. II.1.3. Graphe d'état au départ

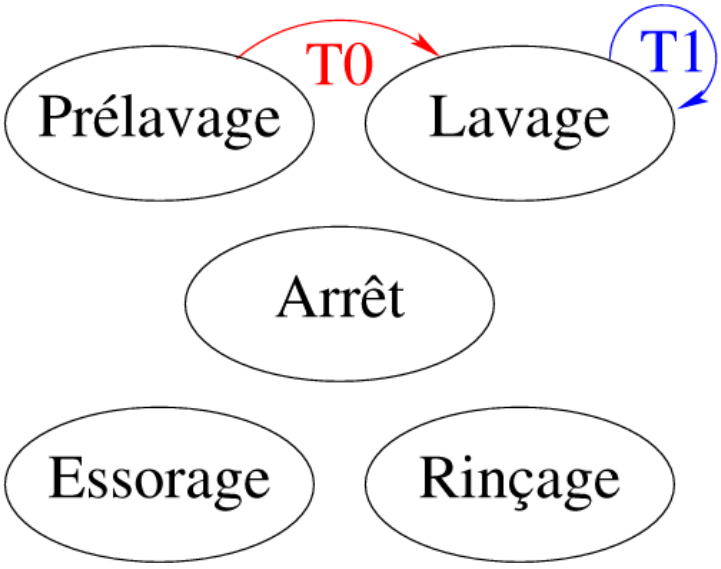


Fig. II.1.4. Graphe d'état avec quelques transitions.

A tout instant la machine est dans l'un des états représentés : c'est ce que nous appellerons l'état courant de la machine. A chaque front montant de l'horloge, la machine emprunte l'une des transitions possibles à partir de son état courant. Elle change alors d'état. Retenez bien cette conséquence du fait que notre machine est synchrone sur front montant de l'horloge : elle reste dans un état donné (une bulle du graphe) pendant le temps qui sépare deux fronts montants de l'horloge (voire plus si elle emprunte ensuite une transition vers le même état). Les transitions (les flèches du graphe), en revanche, sont quasi-instantanées puisqu'elles correspondent aux fronts montants de l'horloge.

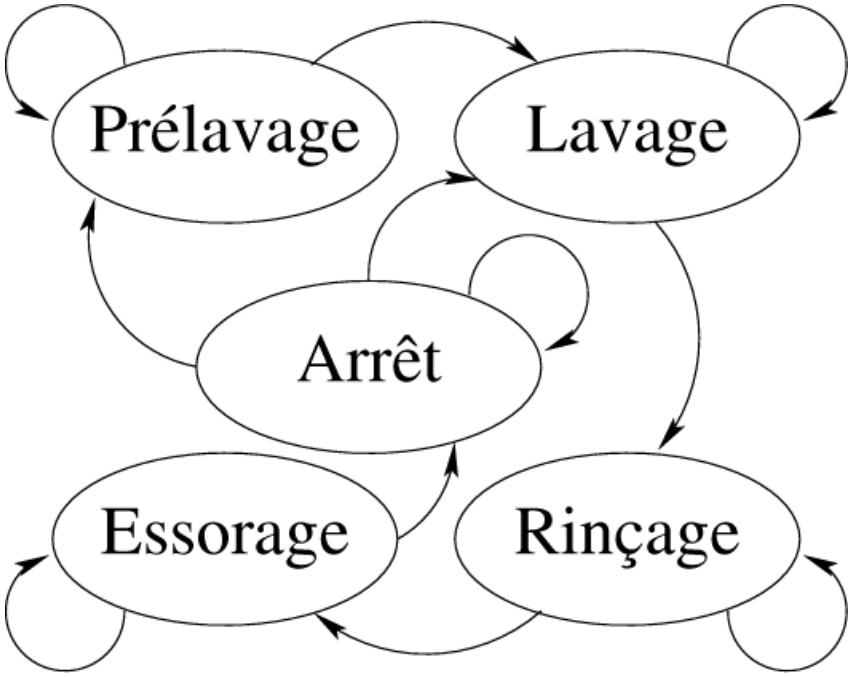


Fig. II.1.5. Graphe d'état avec les transitions

Pour enrichir encore notre graphe nous devons préciser les spécifications de la machine et plus particulièrement, la loi d'évolution des variables internes (l'état) en fonction des entrées. Supposant que les entrées de notre machine soient au nombre de trois:

M : : variable booléenne qui traduit la position du bouton Marche/Arrêt du lave-linge.
P : variable booléenne qui indique si le programme de lavage sélectionné par l'utilisateur comporte ou non une phase de prélavage.
C : valeur en minutes d'un chronomètre qui est remis à zéro automatiquement au début de chaque étape de lavage.

Les durées des différentes étapes de lavage sont fixées par le constructeur :

a- prélavage : 10 minutes
b- lavage : 30 minutes
c- rinçage : 10 minutes
d- essorage : 5 minutes

A partir de ces informations complémentaires nous pouvons faire figurer sur le graphe les conditions logiques associées à chaque transition. Avec un graphe ainsi complété comme il apparaît dans la figure (II.1.6), il devient très facile de comprendre ou de prévoir le comportement de la machine. On sait par exemple que lorsque la machine est dans l'état arrêt, elle y reste tant que M n'est pas vrai au moment d'un front montant de l'horloge. Dès que M est vrai au moment d'un front montant de l'horloge, la machine change d'état : elle passe dans l'état Prélavage si P est vrai et dans l'état Lavage si P est faux. Il est important de comprendre que la valeur des entrées de la machine n'a d'importance qu'au moment précis des fronts montants de l'horloge. C'est une conséquence du fait que notre machine est synchrone sur front montant de l'horloge.

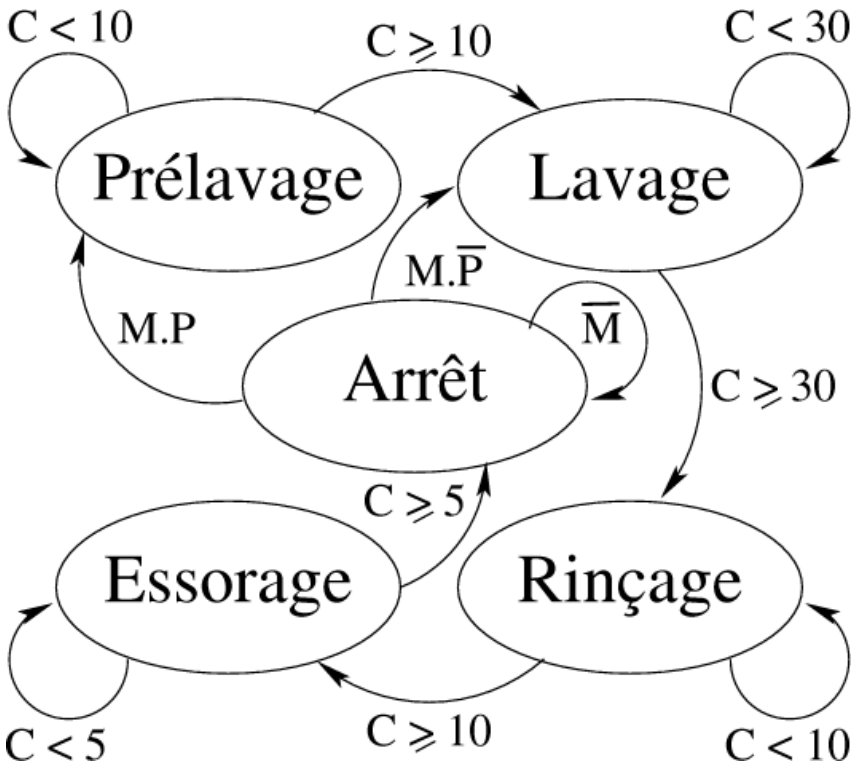


Fig. II.1.6. Graphe d'état avec les transitions spécifiées

Notre machine à états possède des entrées mais nous n'avons pas encore étudié les sorties. Or un circuit électronique sans sorties n'est que de peu d'utilité. Il existe deux sortes de machines à états : celles dont les sorties ne dépendent que de l'état courant (ce sont les machines dites de Moore) et

celles dont les sorties dépendent de l'état courant et des entrées (ce sont les machines dites de Mealy). Nous allons donc réduire encore la généralité de notre étude et nous concentrons sur les machines de Moore. Le programmateur de notre lave-linge est donc une machine de Moore dont les sorties ne dépendent que de l'état courant. Nous supposerons que ces sorties, chaque état que peut prendre la machine. Nous pouvons encore compléter le graphe d'états afin d'y faire figurer cette information. Le graphe est alors achevé comme illustré dans la figure (II.1.7)

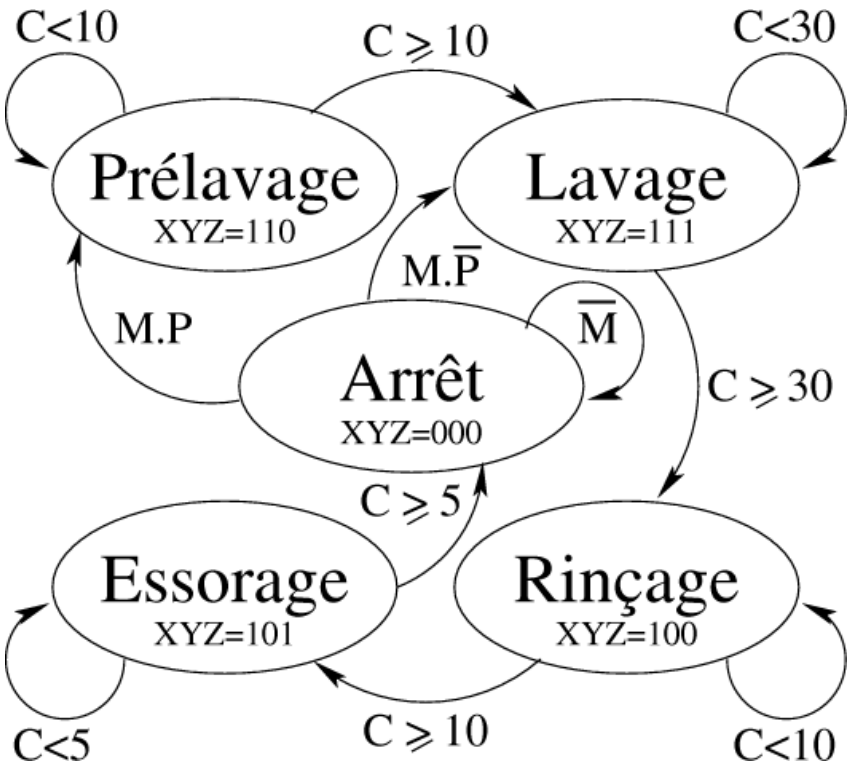


Fig.II.1.7.Graphe d'état final

II.1.3.2. vérification de la représentation à l'aide de quelques règles simples :

Les spécifications sont généralement écrites en langage naturel. La traduction des spécifications en graphe d'état est donc entièrement manuelle et les risques d'erreurs sont nombreux, comme c'est toujours le cas lorsqu'un humain intervient dans un processus. Si une erreur venait à se glisser dans le graphe elle se retrouverait probablement dans le circuit électronique final (il est peu probable, sauf intervention surnaturelle, qu'une deuxième erreur annule la première), ce qui est inacceptable : un lave-linge qui "oublie" de rincer n'est pas très satisfaisant,

sans parler des centrales nucléaires ou des avions de ligne. Il faut donc vérifier le graphe avant de poursuivre la réalisation de la machine. Comme toute bonne spécification il doit vérifier deux propriétés fondamentales :

a- il doit être complet ou non ambigu.

b- il doit être non contradictoire.

La première signifie que le comportement est toujours défini : à chaque front montant d'horloge, quel que soit l'état dans lequel se trouve la machine et quelles que soient les valeurs des entrées, on doit connaître l'état suivant. L'une des conditions associées aux transitions partant d'un état quelconque du graphe doit donc toujours être vraie. On peut traduire cette propriété sous forme d'équation booléenne en écrivant que le OU logique de toutes les conditions associées au transitions partant d'un état quelconque est toujours vrai :

Soient (C1, C2, ..., Ci, ..., Cn) ces conditions, alors :

$\sum_{i=1}^{i=n}$ i=1

Par exemple, pour le programmateur de notre lave-linge, les transitions partant de l'état Arrêt sont au nombre de trois comme indiqué en pointillé sur la figure (II.1.8)

Et les conditions associées sont :

$\bar{M}$, M*$\bar{P}$, M*P

Le OU logique de ces trois conditions vérifie donc :

$\bar{M}$ +M*$\bar{P}$ +M*P=$\bar{M}$+M ($\bar{P}$+P) = $\bar{M}$+M=1.

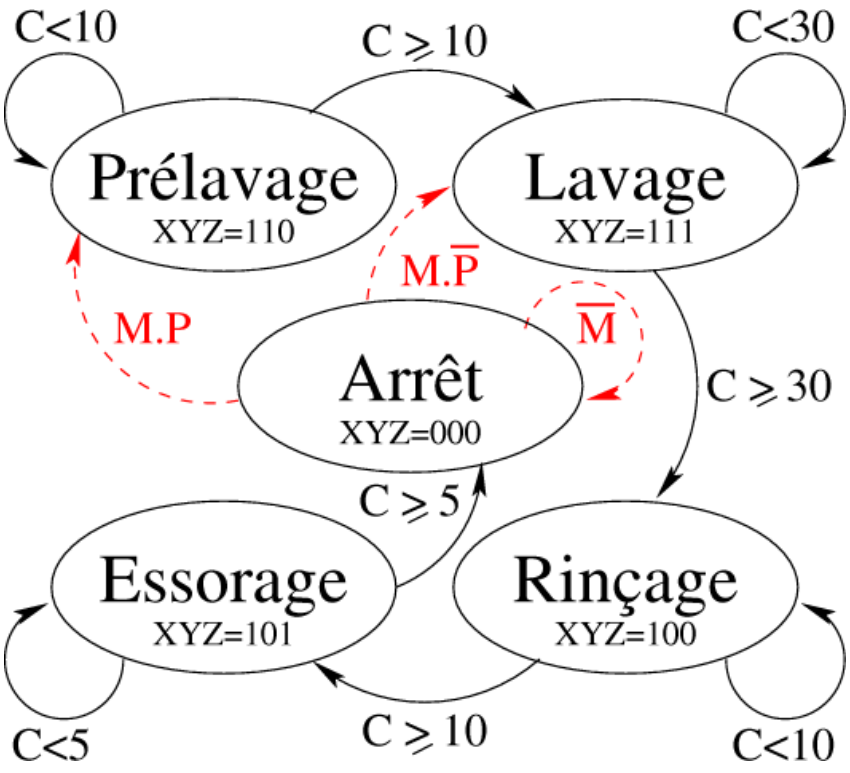


Fig . II.1.8. Graphe d'état final

L'état arrêt respecte donc la première règle. La deuxième règle signifie qu'à tout front montant d'horloge une seule transition est possible. Si plus d'une transition a sa condition associée vraie, le graphe est contradictoire (deux actions incompatibles sont simultanément possibles). Le respect de cette règle est plus difficile à vérifier : le OU logique de tout les ET logiques de deux conditions associées aux transitions partant d'un état quelconque est toujours faux :

$\sum_{i=1}^{i=n} \sum_{j=i+1}^{j=n} i\, c_j = 0.$

En reprenant l'état Arrêt du programmateur de lave-linge comme exemple :

$\bar{M}$ *M* $\bar{P}$+$\bar{M}$ *M*P+M$\bar{P}$*M*P=0+0+0=0

L'état arrêt respecte donc également la deuxième règle. Si elle est aussi vérifiée par les autres états, alors nous sommes en présence d'un véritable graphe de machine à états sans ambiguïté ni contradiction. Malheureusement cela ne prouve pas que le graphe soit conforme à la spécification. Il faut encore vérifier que la fonctionnalité est la même dans les deux descriptions. Il n'existe pas d'outils de vérification ou de formules logiques permettant de le faire. Vous pouvez par exemple parcourir le graphe état par état et, pour chacun d'eux, comparer la partie de spécification qui le concerne avec les conditions associées aux transitions sortantes. Toute méthode est bonne si elle permet d'éviter des erreurs à ce stade du travail de conception.

II.2. Réseaux de Pétri :

II.2.1. Introduction :

Les réseaux de Pétri constituent un outil très puissant. Créer par ¨ CARL ADAM PÉTRI¨ en 1962, pour les buts suivants :

Modélisation visualisation des comportements des systèmes dynamiques.

Description des relations existant entre les conditions et les évènements.

Visualisation de la synchronisation et le partage de ressources dans les systèmes automatisés.

II.2.2. Définition :

Un réseau de Pétri est un graphe orienté, défini par un quadruplet (T, P, A, M◦) ou :

T= $\{t_1, t_2,\ldots,t_l\}$ est un ensemble fini de transition représentées par des tirets figure (II.2.1.a)

P= $\{p_1, p_2, \ldots, p_m\}$ est un ensemble fini de places représentées par des cercles figure (II.2.1.b)

A= $\{a_1, a_2, \ldots, a_n\}$ est un ensemble fini d'arcs orientés qui assurent la liaison d'une place vers une transition ou d'une transition vers une place figure (II.2.1.c)

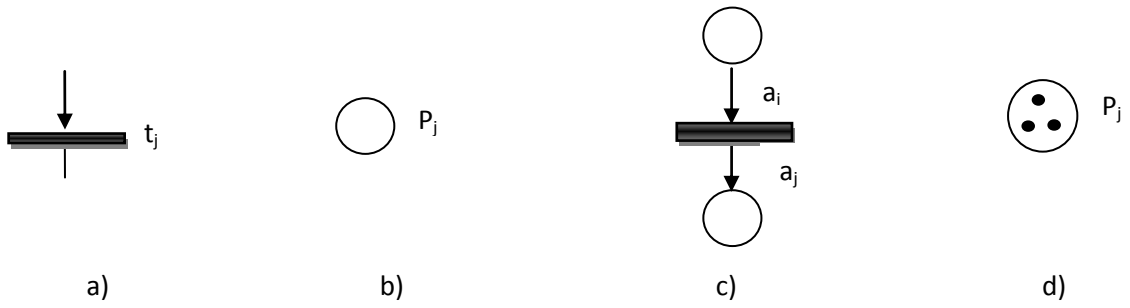


Fig. II.2.1. Le graphe orienté d'un réseau de Pétri

M◦:{P ⟶ N+} est le marquage initial du graphe précisé par la présence à l'intérieur des cercles représentant les places d'un nombre nul ou fini de marqueurs figure (II.2.1.d) Une place peut donc être vide ou marquée. A chaque transition est associée un ensemble de places d'entrée et un ensemble de place de sortie. Les places d'entrée sont les places d'où sont issus les arcs orientés vers la transition.les places de sortie sont les places ou aboutissent les arcs orientés issus de la transition.une place peut être à la fois place d'entrée et de sortie d'une transition.

II.2.3. Notation :

On note :

$^{\circ}T_j$ l'ensemble des places d'entrée de T_j.

Et T_j° l'ensemble des places de sortie de T_j.

$^{\circ}T_1 = \{P_1, P_2, P_3\}$

$T_1^{\circ} = \{P_4, P_5\}$

Transition source :

On appel transition source, une transition qui n'a aucune place d'entrée.

Exemple :

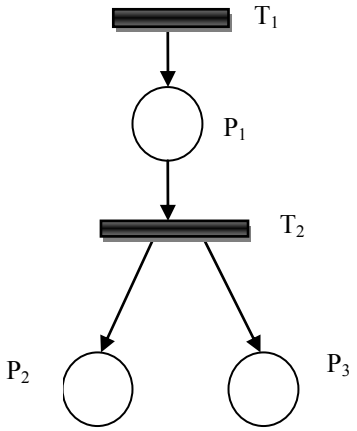


Fig. II.2.2. La transition source.

T_1 : transition source cette transition est toujours franchissable.

On appel une transition fini : une transition qui n'a aucune place de sortie.

T_1 est une transition puit.

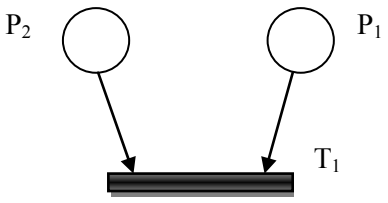


Fig. II.2.3. Transition puit.

Une place puit est une place qui n'a aucune transition de sortie.

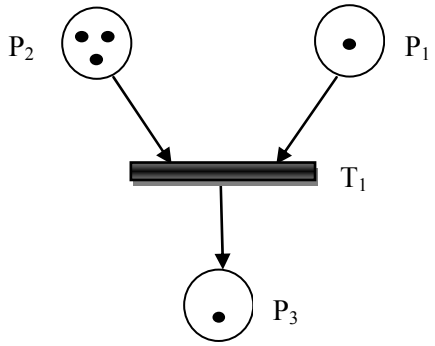


Fig. II.2.4. Place puit

P_3 est une place puit.

II.2.4. Règle de validation et de franchissement d'une transition :

L'application de la règle de validation et de franchissement d'une transition permet de faire évoluer séquentiellement le marquage d'un réseau. Par définition, une transition est validée si chaque place d'entrée de cette transition comporte au moins un marqueur. La transition T_1 du graphe élémentaire représentée par la figure (II.2.5.a) est validée, les places P_1et P_2 possédans chacune au moins un marqueur. Une transition validée peut être franchissable. L'opération de franchissement revient à enlever de chaque place d'entrée de la transition un marqueur et à ajouter à chaque place de sortie un marqueur. La figure (II.2.5.b) Représente le marquage des places P_1, P_2, P_3, P_4 résultantes du franchissement de la transition T_1.

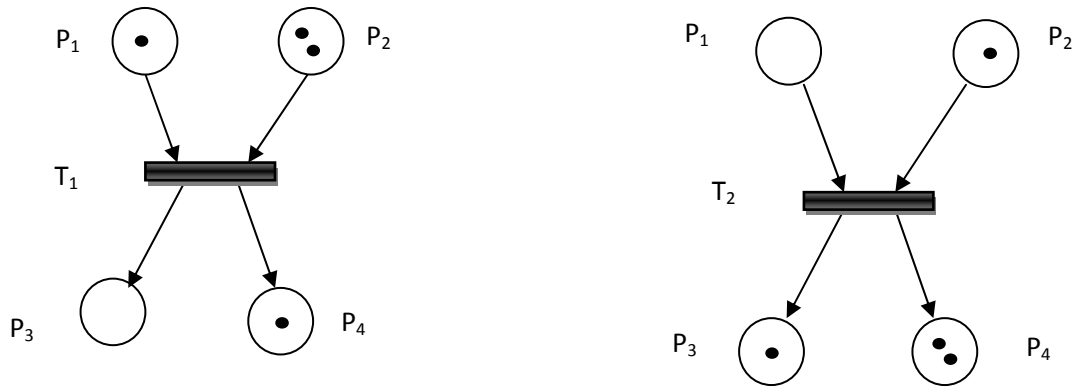


Fig. II.2.5. Franchissement des transitons.

II.2.5. Étude du marquage d'un réseau :

L'application séquentielle à un réseau de Pétri de la règle de validation et d'une ou d'un ensemble de transitions permet la détermination de l'ensemble des marquages qu'il est possible

d'atteindre à partir d'un marquage initial M_0.nous dénoterons [M]= {M_0, M1, M2,…} l'ensemble de ces marquages.

II.2.5.1. Graphe de marquage :

Le graphe de marquage indique les marquages successifs du marquage initial.

On utilise uniquement lorsque le nombre de successeur est fini.

Exemple :

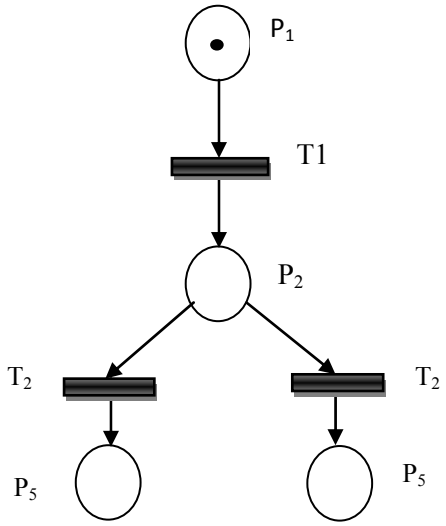


Fig. II.2.6. Graphe de marquage.

II.2.6. Réseaux de Pétri particulier :

II.2.6.1. Graphe d'état :

Un réseau de Pétri non marqué est un graphe d'état si et seulement si toute transition a exactement une seule place d'entrée et une seule place de sortie.

Exemple :

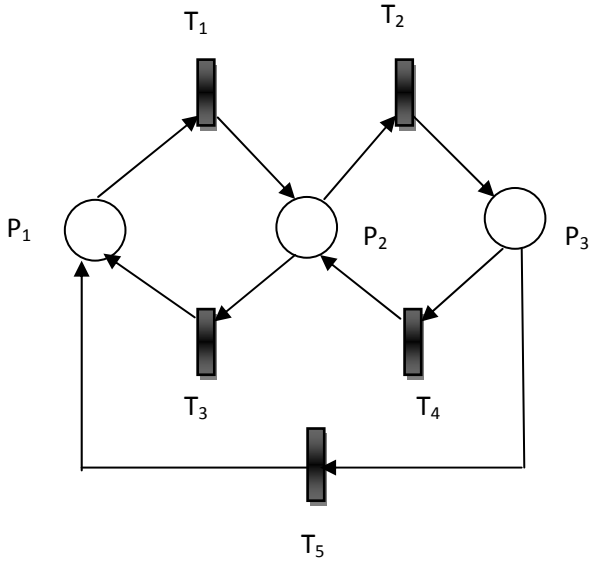


Fig. II.2.7. Graphe d'état.

Chacune des transitions T_1, T_2, T_3, T_4 et T_5 possède une seule place d'entrée et une seule place de Sortie.

II.2.6.2. Graphe d'évènement :

Un Réseau de Pétri est un graphe d'évènement si et seulement si chaque place possède exactement une seule transition d'entrée et une seule transition de sortie.

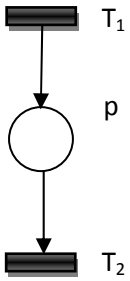


Fig. II.2.8.a. Graphe d'événement

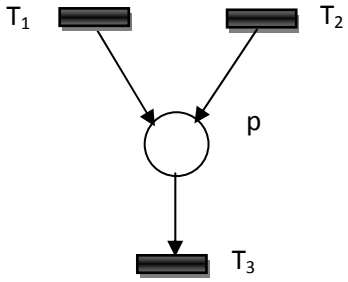


Fig. II.2.8.b. Non graphe d'événement

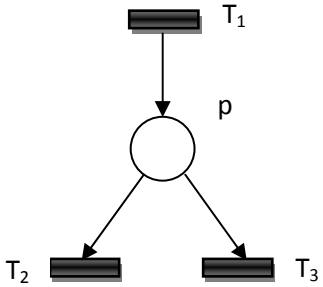


Fig. II.2.8.c. Non graphe d'événement

II.2.6.3. Réseaux de Pétri sans conflit :

Un réseau de Pétri sans conflit est un réseau dans lequel chaque place a au plus une transition de sortie.

Un réseau de Pétri avec conflit est un réseau qui possède donc une place avec au moins deux transitions de sorties. Un conflit est noté: [P_i , $\{T_1,T_2,T_n\}$] , avec (T_1,T_2,T_n) étant les transitions de sorties de la place P_i .

Exemple :

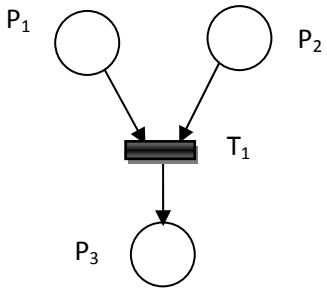


Fig. II.2.9.a. Un réseau de Pétri sans conflit

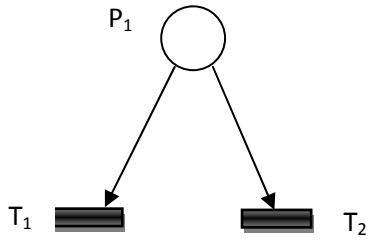


Fig. II.2.9.b. Un réseau de Pétri avec conflit

II.2.6.4. Réseaux de Pétri choix libre :

Un réseau de Pétri est choix libre est un réseau dans lequel pour tout conflit [P_i, $\{T_1$, T2,...,$T_n\}$] aucune des transitions ($T_1,T_2,\ldots,T_n$) ne possède aucune autre place d'entrée que P_i .

Exemple :

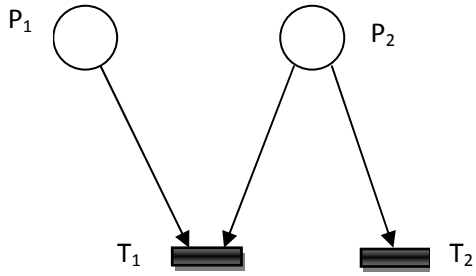


Fig. II.2.10.a. Réseau de Pétri avec conflit sans chois libre

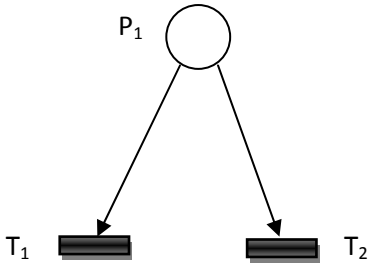


Fig. II.2.10.b. Réseau de Pétri avec conflit et chois libre

II.2.6.5. Réseau de Pétri simple :

Un réseau de Pétri simple est un réseau dans lequel chaque transition ne peut être concernée que par un conflit au plus.

Exemple :

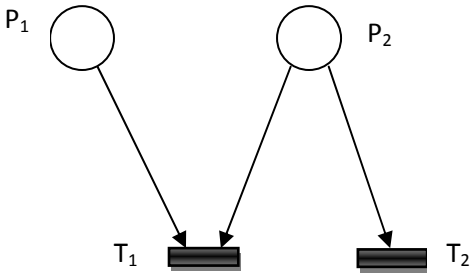


Fig. II.2.11.a. Un réseau de Pétri avec conflit sans Chois libre simple.

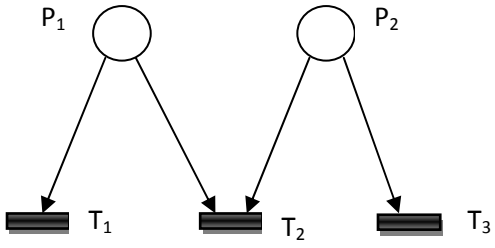


fig. II.2.11.b. Un réseau de Pétri avec conflit avec Chois libre non simple.

II.2.6.6. Réseau de Pétri pur :

Un réseau de Pétri pur est un réseau dans lequel il n'existe pas de transition ayant une place d'entrée qui soit à la fois place de sortie de cette transition. Figure (II.2.12.).

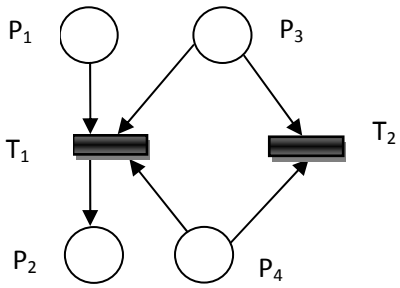


Fig. II.2.12.a. Un réseau de Pétri pur

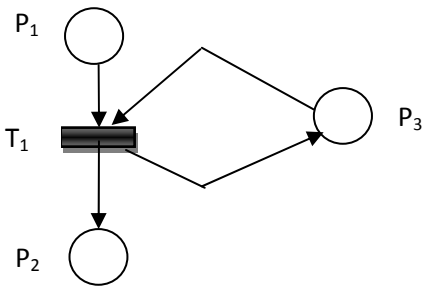


Fig. II.2.12.b. Un réseau de Pétri non

II.2.6.7. Réseaux de Pétri généralisés :

Un réseau de Pétri généralisé est un réseau dans lequel des poids (nombres entiers strictement positifs) sont associés aux arcs.

Si un arc (P_i, T_j) a un poids k : la transition T_j n'est franchie que si la place P_i possède au moins k jetons. Le franchissement consiste à retirer k jetons de la place P_i. Si un arc (T_j, P_i) a un poids k : le franchissement de la transition rajoute k jetons à la place P_i.

Lorsque le poids n'est pas signalé il est égal à un par défaut figure (II.2.13).

Exemple:

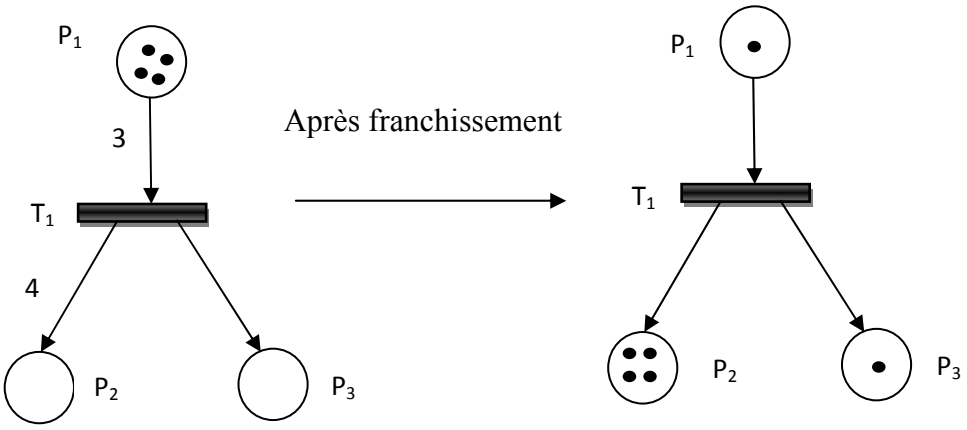


Fig. II.2.13. Réseau de Pétri généralisé

II.2.6.8. Réseaux de Pétri capacités :

Un réseau de Pétri capacités est un réseau dans lequel des capacités (nombres entiers strictement positifs) sont associées aux places. Le franchissement d'une transition d'entrée d'une place P_i dont la capacité est cap(Pi) n'est possible que si le franchissement ne conduit pas a un nombre de jetons dans P_i qui est plus grand que Cap(P_i).

Exemple :

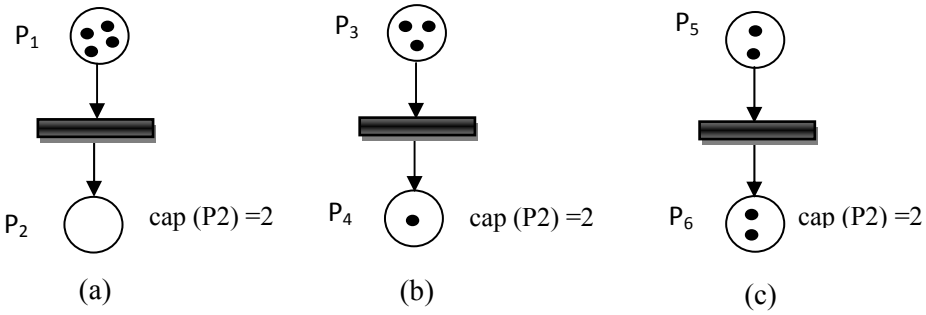


Fig. II.2.14. Un réseau de Pétri capacités

II.2.6.9. Réseau de Pétri a priorité :

Dans un tel réseau si on atteint un marquage tel que plusieurs transitions sont franchissables, on doit franchir la transition qui a la plus grande priorité.

Exemple :

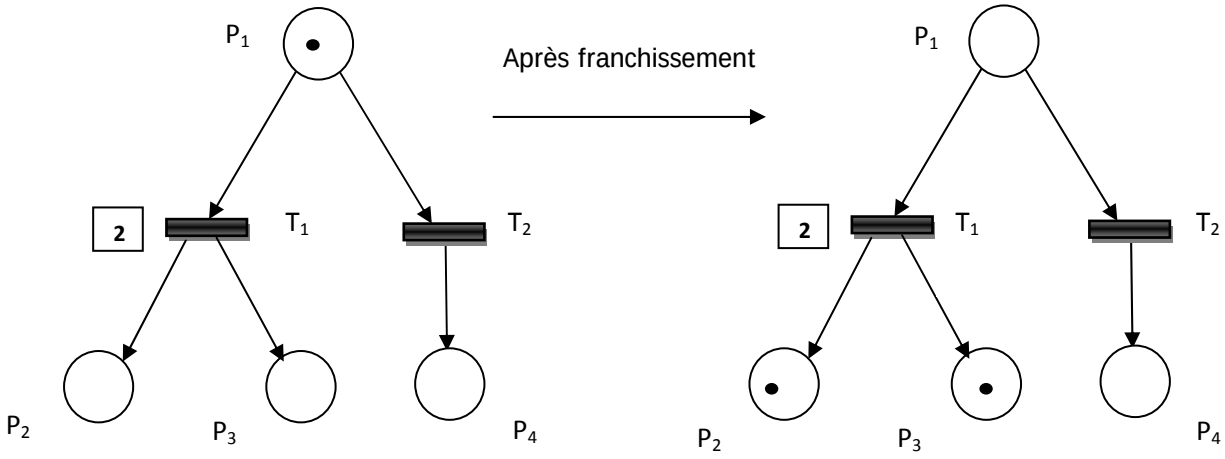


Fig. II.2.15. Un réseau de Pétri priorité après

II.2.7.Propriétés :

II.2.7.1. Réseaux de Pétri vivant :

Un réseau est dit vivant pour un marquage $M_\circ$ si toute transition du réseau peut être validée et franchie par une séquence finie de franchissement.

Un réseau de Pétri est vivant si toutes ses transitions sont vivantes.

Un réseau est quasi-vivant (non vivant) s'il contient au moins une transition quasi vivante

T_j est quasi vivante s'il a une seule chance d'être franchi.

II.2.7.2. Réseaux de Pétri non vivant :

Un réseau de Pétri non vivant si aucune transition n'est franchissable.

II.2.7.3. Réseaux de Pétri sauf :

Un réseau est dit sauf pour un marquage initial M_0 si quel que soit le marquage obtenu à partir de M_0 par une séquence finie de tirs aucune place ne possède plus d'un marqueur.

II.2.7.4. Réseaux de Pétri - k- bornés:

P_i est k-borné si et seulement si : □ M□ $^*M_\circ$ et $M(Pi) \leq k$ ou $k \geq 1$

Un réseau de Pétri est k-bornés, si et seulement si toutes ses places p_i sont k-bornées.

Remarque :

Lorsque k=1 le réseau de Pétri est binaire (sauf).

II.2.7.5. Réseaux de Pétri non borné :

Un réseau de Pétri est non bornée pour un marquage initial $M_\circ$ s'il existe au moins une place p_i pour la quelle le nombre de marquage peut augmenter indéfiniment.

II.2.7.6. Réseaux de Pétri marqués réinitialisable :

Pour un marquage initial $M_\circ$ si et seulement si :

M $^*M_\circ$, S une séquence si telle que : M $[S_i \longrightarrow M_\circ]$.

Remarque :

Un réseau peut être vivant mais pas réinitialisable.

II.2.8. Présentation matricielle :

Les propriétés et l'évolution du marquage d'un réseau de pétri {P, T, A, $M_\circ$} peuvent être étudiées en utilisant une représentation matricielle.

Matrice de description

Soit {P, T, A, $M_\circ$} un réseau de Pétri comportant n places

P= {P1, P2, …, Pn}

Et m transitions T= {t1, t2, …, tm}.On appel matrices de description du réseau les tableaux matriciels S (p, t) et E (p, t) suivants :

S (p, t)=[s_{ij}] ou s_{ij} =1 si la place p_i est une place de sortie de la transition t_j et $s_{ij=0}$ Dans le cas contraire.

E (p, t)=[e_{ij}] ou e_{ij}=1 si la place p_i est une place d'entrée de la transition t_j et e_{ij}=0 dans le cas contraire. Les matrices |S (p, t)| et |E (p, t)| comportent donc un nombre de lignes égal au nombre n

de places et un nombre de colonnes égale au nombre m de transition du réseau.

Dans le cas particulier ou les places d'un réseau ne sont ni place d'entrée, ni place de sortie d'une même transition, il est possible de décrire complètement le réseau par la matrice |C (p, t)| tel que :

$$|C(p,t)| = |S(p,t)| - |E(p,t)|$$

Chaque coefficient c_{ij} de cette matrice est tel que :

C_{ij}=+1si la place i est une place de sortie de la transition,

C_{ij}=-1si la place i est une place d'entée de la transition,

C_{ij}=0 si la place i n'est ni une place d'entrée ni une place de sortie de la transition.

Exemple :

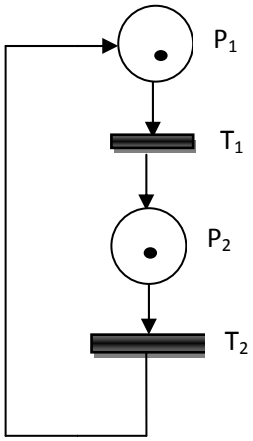


Fig. II.2.16. Présentation matérielle

$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ Matrice d'incidence avant |E (P, t)|.

$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ Matrice d'incidence arrière |S (P, t)|.

On appel matrice d'incidence la différance entre |S (P, t| et |E (P, t)|.

II.2.9. Synthèse d'un automate avec les réseaux de Pétri :

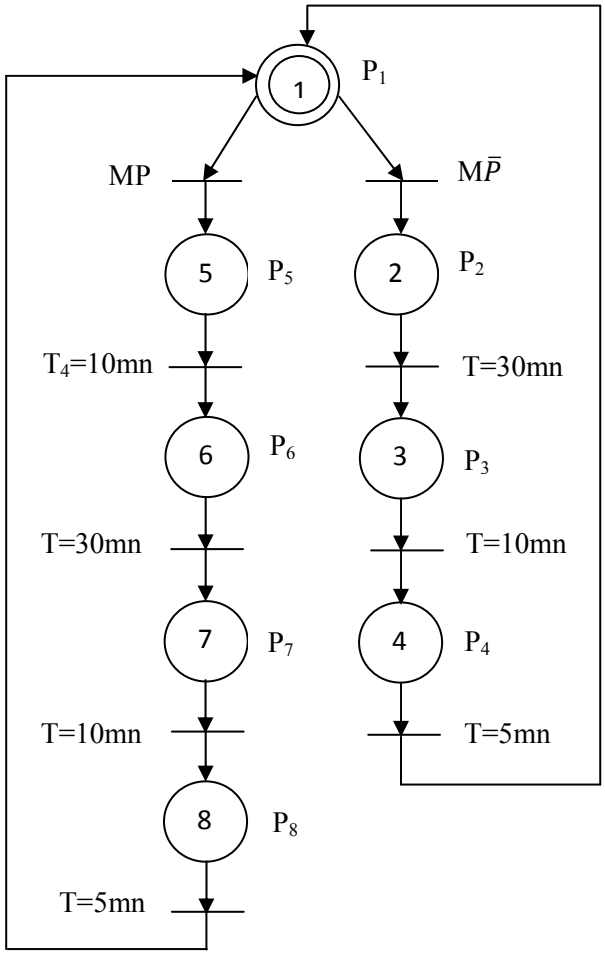


Fig. II.2.17. Machine à laver avec réseau de Pétri

Signification des étapes :

Etape (1) ⟶ Arrêt

Etape (5) ⟶ Prélavage

Etape (2) et (6) ⟶ Lavage

Etape (3) et (7) ⟶ Rinçage

Etape (4) et (8) ⟶ Essorage

II.3. Le grafcet

II. 3.1. Introduction :

Le langage grafcet (Graphe Fonction de Commande Etape Transition) a été introduit en 1977 par l'AFCET (Association Française pour le Cybernétique Economique et Technologie), le grafcet est un outil graphique qui permet de spécifier et concevoir un automatisme séquentiel directement exploitable par des automates programmables industriels.

Le grafcet repose sur l'utilisation d'instructions précises, l'emploi d'un vocabulaire bien défini, le respect d'une syntaxe rigoureuse et l'utilisation des règles de l'évolution. Il décrit le comportement des sorties **S**n a partir des entrées **E**n et de l'état interne du système comme le montre la figure ci dessous.

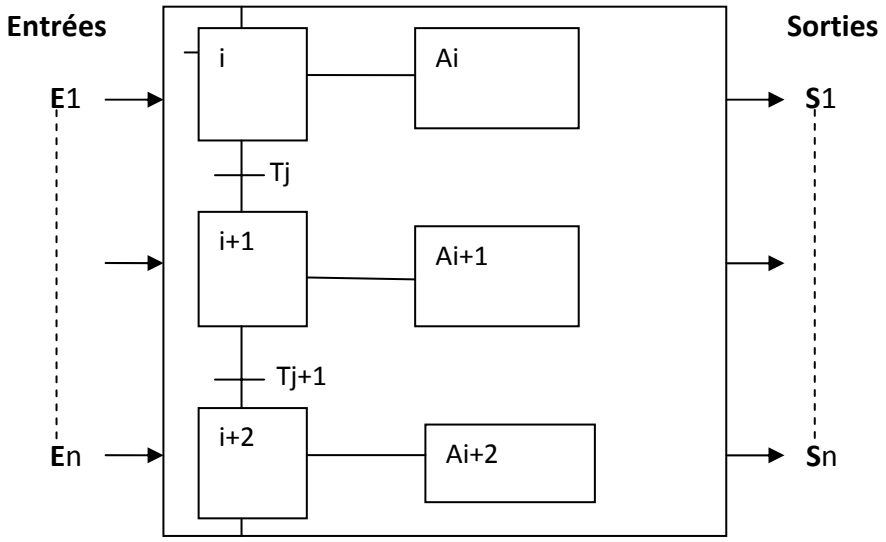


Fig. II.3.1. Schéma général d'un grafcet.

II. 3.2. Structure graphique :

La représentation de grafcet est faite à partir des éléments graphiques de base suivant:

II.3.2.1. Une liaison :

Est un arc orienté. A une des extrémités d'une liaison il ya une seule étape, à l'autre une transition**.** On la représente par un trait plein rectiligne vertical ou horizontal.

II.3.2.2. Une étape :

Correspond à une phase durant laquelle on effectue une action pendant une certaine durée. L'action doit être stable. On représente chaque étape par un carré, à l'intérieur est inscrit un numéro associé à cette étape, l'action est représenté à droite de l'étape par un rectangle comme le montre la figure suivante :

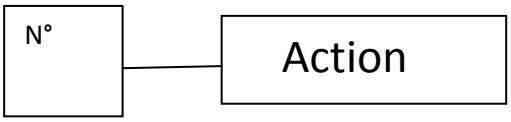


Fig. II.3.2. Illustration de l'action.

Une étape est dite active (étape initiale) lorsqu'elle est en fonctionnement, on la représente par un point à l'intérieur et un carré double

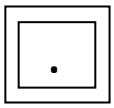


Fig. II.3.3. Une étape initiale active

II.3.2.3.Une transition :

Est une condition de passage d'une étape a une autre cette condition est définie par une réceptivité comme le montre la figure (II.3.4.a) .si plusieurs liaisons arrivant sur une transition on les fait converger sur une grande double barre horizontale comme le montre la figure (II.3.4.b)

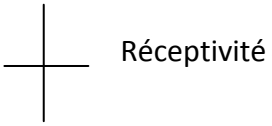


Fig. II.3.4.a. La réceptivité

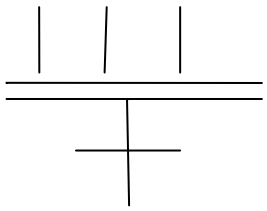


fig. II.3.4.b. La réceptivité de plusieurs liaisons

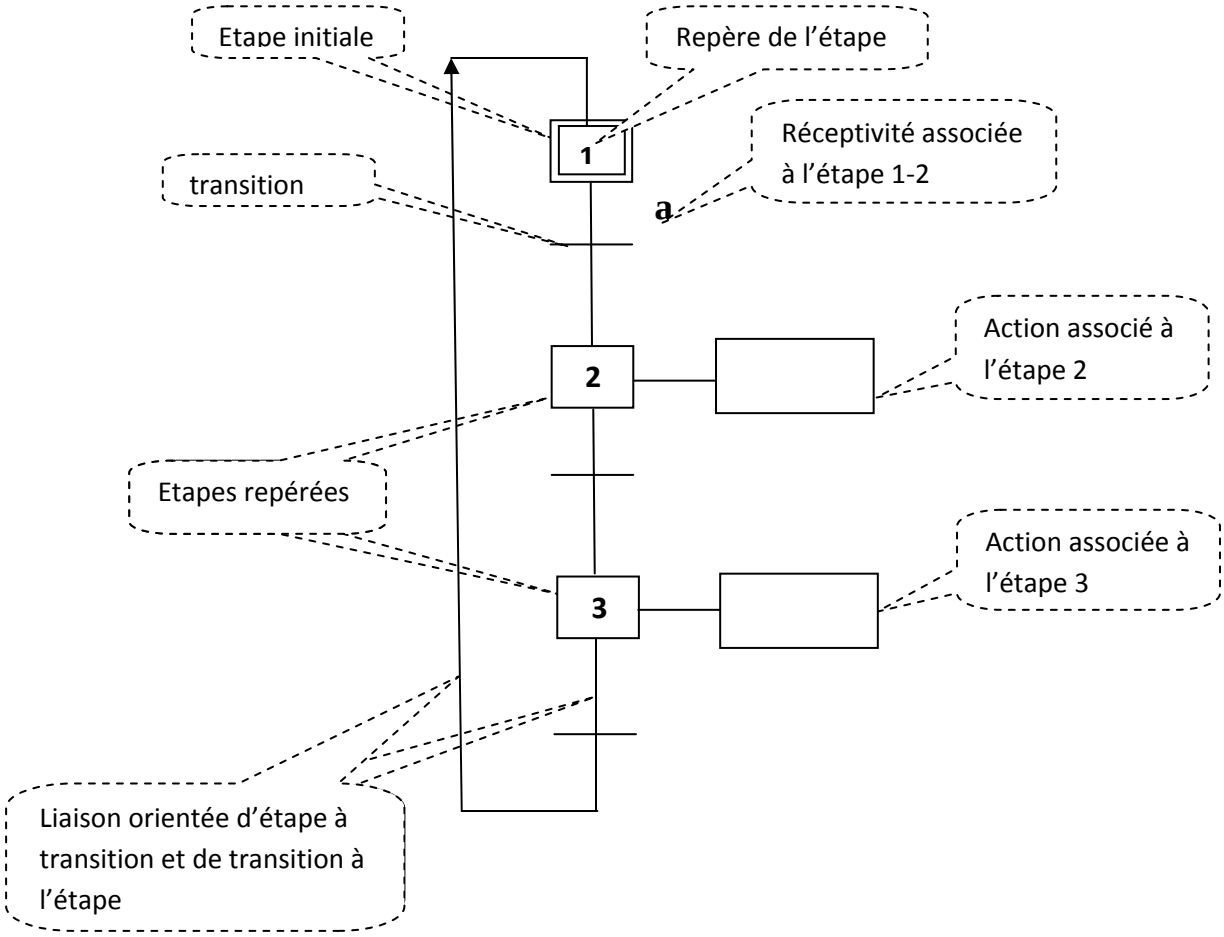


Fig. II.3.5. Schéma structurel d'un grafcet .

II.3.3.Règles d'évolution

La modification de l'état de l'automatisme est appelée évolution, et est régie par 5 règles :

R1 : Les étapes initiales sont celles qui sont actives au début du fonctionnement. On les représente en doublant les côtés des symboles. On appel début du fonctionnement le moment où le système n'a pas besoin de se souvenir de ce qui c'est passé auparavant (allumage du système, bouton "reset",...). Les étapes initiales sont souvent des étapes d'attente pour ne pas effectuer une action dangereuse par exemple à la fin d'une panne de secteur.

R2 : Une transition est soit validée, soit non validée (et pas à moitié validée). Elle est validée lorsque toutes les étapes immédiatement précédentes sont actives (toutes celles reliées directement à la double barre supérieure de la transition). Elle ne peut être franchie que lorsqu'elle est validée et que sa réceptivité est vraie. Elle est alors obligatoirement franchie.

R3 : Le franchissement d'une transition entraîne l'activation de toutes les étapes immédiatement suivante et la désactivation de toutes les étapes immédiatement précédentes (toute se limitant à 1 s'il n'y a pas de double barre).

R4 : Plusieurs transitions simultanément franchissables sont simultanément franchies (ou du moins toutes franchies dans un laps de temps négligeable pour le fonctionnement). La durée limite dépend du "temps de réponse" nécessaire à l'application.

R5 : Si une étape doit être à la fois activée et désactivée, elle reste active. Une temporisation ou un compteur actionné par cette étape ne seraient pas réinitialisés. Cette règle est prévue pour lever toute ambiguïté dans certains cas particuliers :

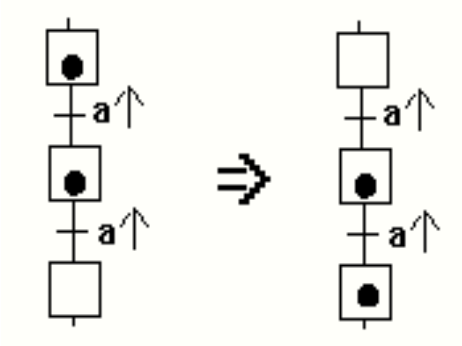
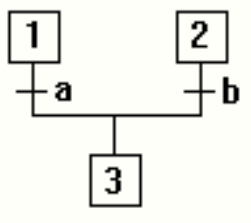


Fig. II.3.6. Evolution d'un grafcet

II.3.4. Configurations courantes :

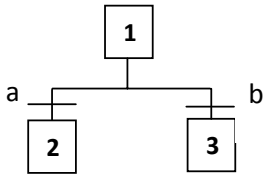
II.3.4.1.Convergence en OU :

Si 1 active et a est vrai, alors activation de 3 et désactivation de 1, 2 restes inchangés
Si 1 et 2activent et b vrai alors 3 seules active



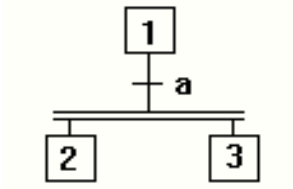
II.3.4.2. Divergence en OU :

si 1 active et si a vrai, alors désactivation de 1 et activation de 2, 3 inchangé.
si a et b vrai puis 1 active alors désactivation 1, activation 2 et 3 quel que soit leur état précédent. (règle 4)



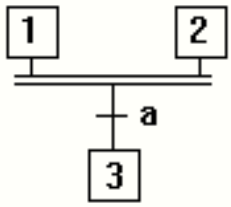
II.3.4.3.Divergence en ET :

si 1 active et si a vrai, alors désactivation de 1 et activation de 2 et 3.



II.3.4.4 Convergence en ET :

Si 1 active seule et avrai alors aucun changement. Si 1 et 2 activent et avrai, alors activation de 3 et désactivation de 1 et 2.



II.3.5. Structures de base d'un grafcet:

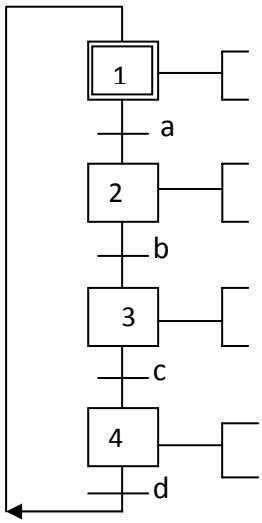


Fig. II.3.7.a. Séquence unique

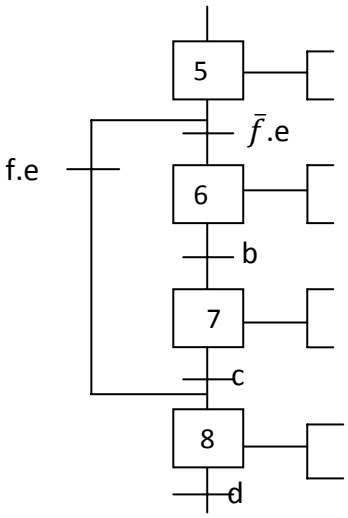


Fig. II.3.7.b. Saut d'étapes

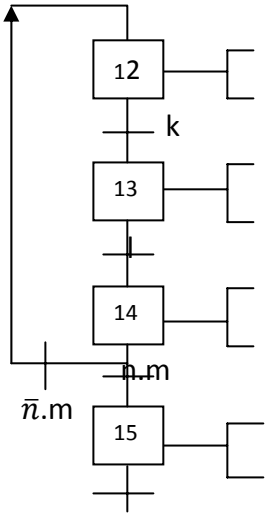


Fig. II.3.7.c. Reprise de séquence

II.3.5.1. Séquence unique :

Elle est composée d'une suite d'étapes pouvant être activé les unes après les autres.

II.3.5.2. Saut d'étapes :

Saut de l'étape 5 vers l'étape 8 par la réceptivité (f.e) comme le montre la figure (II.3.7.b).

II.3.5.3.reprise de séquence :

Reprise de la séquence 12, 13,14 tant que la réceptivité (n.m) n'est pas vraie comme le montre la figure (II.3.7.c).

II.3.6.Temporisation :

La prise en compte du temps peut aussi être réalisée dans la réceptivité. La temporisation est lancée dès l'activation de l'étape X3, elle n'est effective qu'au bout du temps T=3s. La réceptivité étant vraie, la transition est franchie.

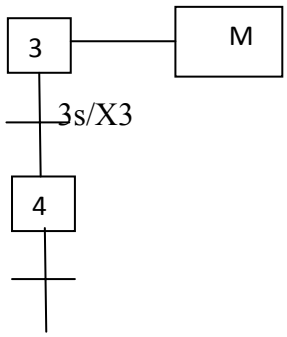


Fig. II.3.8. La temporisation

II.3.7. Mise en œuvre du grafcet :

II.3.7.1. Réalisation par câblage :

Le but de cette partie est de montrer comment mettre en œuvre un grafcet à l'aide de portes logiques et de bascules.

II.3.7.1.1. Grafcet linéaire :

Il suffit d'utiliser une bascule RS pour matérialiser une étape qui est (activée ou désactivée). Dans le cas d'un grafcet linéaire, on désactive une étape quand la suivante est active et la réceptivité est vraie.

II.3.7.1.1.1. Règles d'implantation :

a. chaque étape est associée à une bascule, la sortie $\mathbf{X_i}$ de la bascule est **1** si l'étape i est activée et $\mathbf{X_i=0}$ si l'étape **i** est désactivée.
b. la mise a **1** de $\mathbf{X_i}$ est assurée par l'équation logique : $\mathbf{X_i=X_{i-1}*R_{i-1}}$

c. La mise a zéro de $\mathbf{X_i}$ est assuré par : $\mathbf{R_i=X_{i+1}+I}$ **(**ou **I** prévu pour l'arrêt d'urgence).

d. si $\mathbf{X_i}$ est associée à l'étape initiale $\mathbf{X_1=X_N*R_N+I}$

e. une sortie $\mathbf{A_i}$ de l'automate sera réalisée à l'aide de la somme logique des sorties $\mathbf{X_i}$ correspondantes.

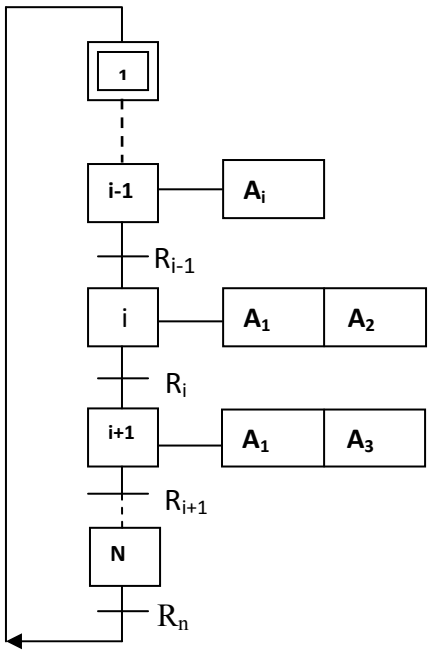


Fig. II.3.9. Grafcet de n étapes

Exemple : Etudions l'étape (2) de la figure(6)

Elle s'allume si l'étape 1 est active et d est vrai ($S2=Q_1.d$). Tout le temps qu'elle est active, la sortie X est allumée (X=Q2). Elle s'éteint normalement quand la réceptivité de sortie est vraie, mais (comme précisé plus haut) nous allons attendre pour éteindre l'étape (2) que l'étape 3 soit active (donc R2=Q3), et donc être sûr que l'étape 3 a eu le temps de prendre en compte l'information. Elle peut également être éteinte par init, puisqu'elle n'est pas initiale.

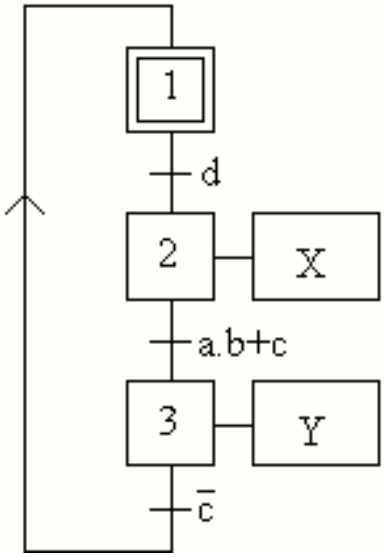


Fig. II.3.10. Exemple de grafcet

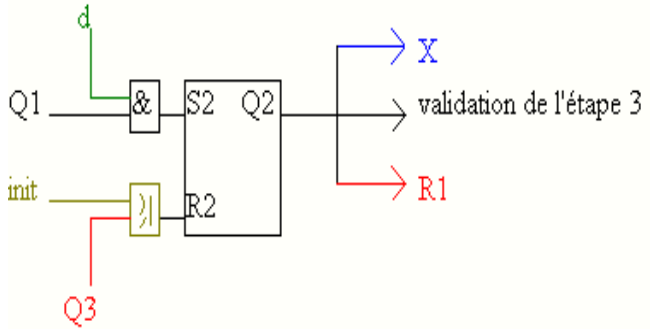


Fig. II.3.11. Etape(2) avec la bascule RS

Il suffit de répéter cela pour chaque étape et relier le tout.

II.3.7.1.2. Grafcet non linéaire :

II.3.7.1.2.1. Divergence simple en ET :

Quand la transition est franchissable, il suffit d'allumer deux étapes au lieu d'une. Le seul problème est la désactivation de l'étape précédente : il faut être sûr que les deux étapes suivantes ont eu le temps de prendre en compte l'information d'activation avant de désactiver la précédente (si l'on désactive dès qu'une des deux est active, la seconde ne s'activera plus).

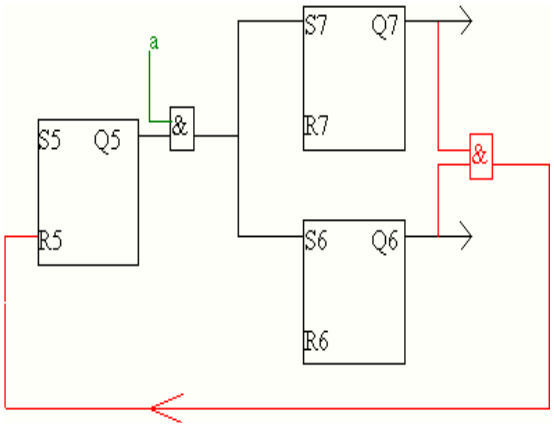


Fig. II.3.12. Divergence en ET avec des bascules RS

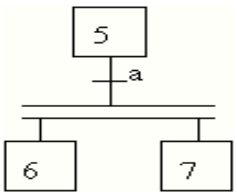


Fig. II.3.13. Divergence en ET

$R_5=Q_6*Q_7$, $S_7=Q_5*a$, $Q_6=Q_5*a$

II.3.7. Synthèse d'un automate avec grafcet :

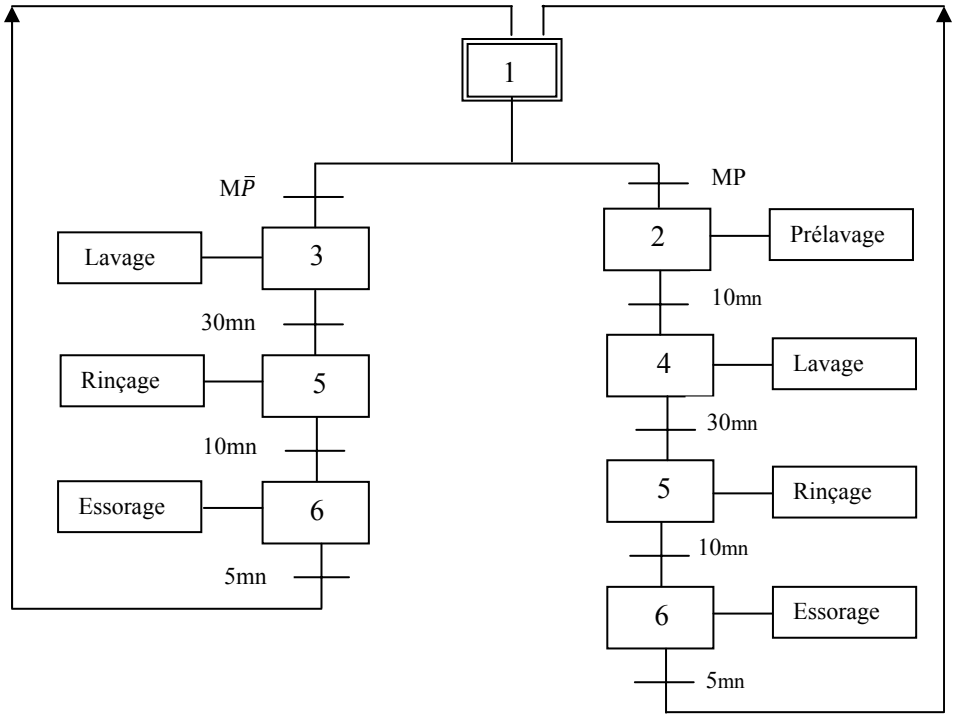


Fig. II.3.13. Synthèse d'un automate avec grafcet

III.8. Conclusion :

Après avoir étudié ces trois méthodes (machines à états finis, réseaux de Pétri et grafcet) on constate que la méthode avec machines à états finis a un inconvénient car elle ne sera pas d'un grand secours lorsque le nombre d'états de la machine dépassera quelques dizaines, c'est le cas contraire pour les deux autres méthodes et aussi elles sont faite avec clairette et sans ambigüité.

Chapitre. III. Gestion d'un carrefour par un circuit logique programmable

III.1. Introduction :

Il y a quelques années la réalisation d'un montage en électronique numérique impliquait l'utilisation d'un nombre important de circuits intégrés logiques. Ceci avait pour conséquences un prix de revient élevé, une mise en œuvre complexe et un circuit imprimé de taille importante.

Le développement des mémoires utilisées en informatique fut à l'origine des premiers circuits logiques programmables (PLD : programmable logic device). Ce type de produit peut intégrer dans un seul circuit plusieurs fonctions logiques programmables par l'utilisateur. Sa mise en œuvre se fait très facilement à l'aide d'un programmateur, d'un micro-ordinateur et d'un logiciel adapté.

III.2. Structure de base d'un P.L.D :

La plupart des PLDs suivent la structure suivante :

- Un ensemble d'opérateurs « ET » sur lesquels viennent se connecter les variables d'entrée et leurs compléments.
- Un ensemble d'opérateurs « OU » sur lesquels les sorties des opérateurs « ET » sont connectées.
- Une éventuelle structure de sortie (Portes inverseuses, logique 3 états, registres...).

Les deux premiers ensembles forment chacun ce qu'on appelle une **matrice**. Les interconnexions de ces matrices doivent être programmables. C'est la raison pour laquelle elles sont assurées par des **fusibles** qui sont « grillés » lors de la programmation. Lorsqu'un PLD est vierge toutes les connexions sont assurées.

Un exemple de ce type de structure est présenté par la figure(III.1). On remarquera que la norme adoptée est américaine.

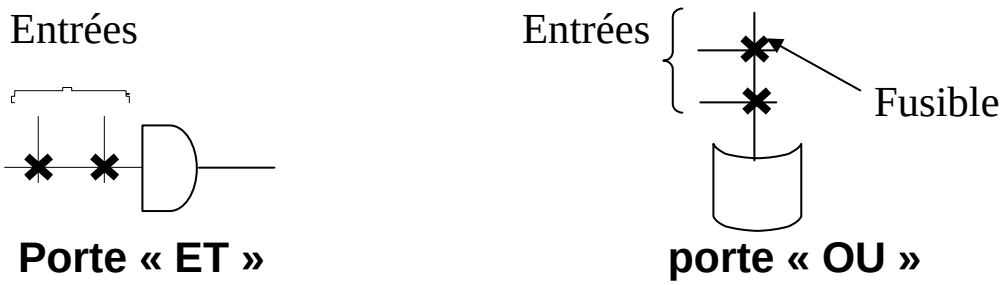


Fig. III.1. Symbolisation des portes logiques pour les PLDs

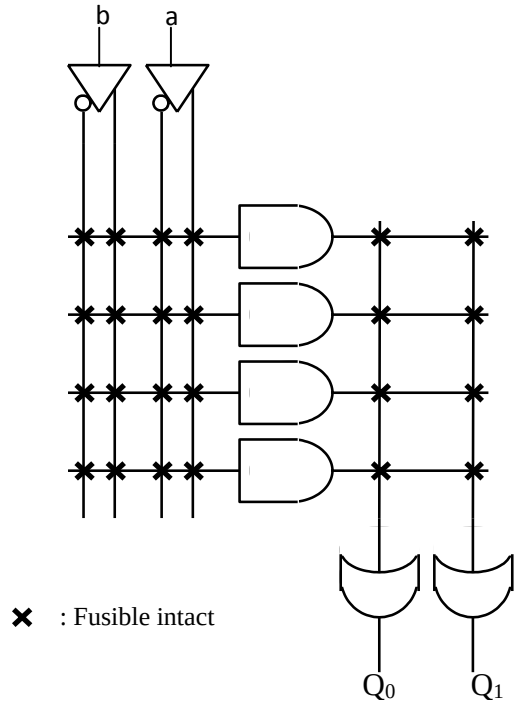


Fig.III.2. Structure de base d'un PLD

Si on veut obtenir les fonctions Q0 = a*b + $\bar{a}$*$\bar{b}$ et Q1 = $\bar{a}$*b + a*$\bar{b}$, on « grillera » des fusibles de façon à obtenir le schéma

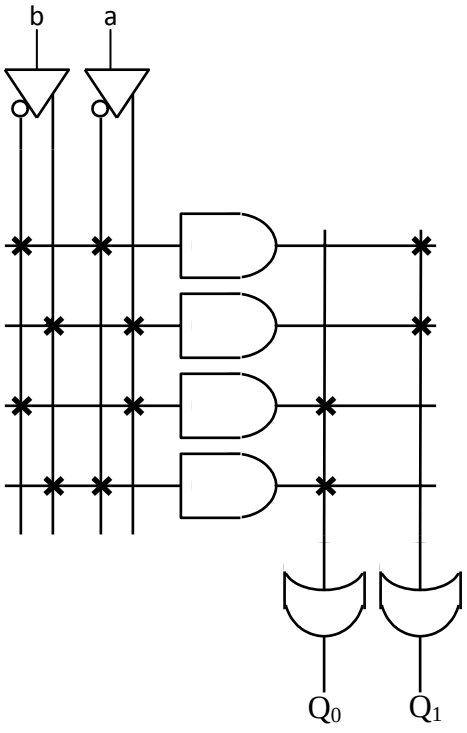


Fig. III.3. PLD programmé

III.3. Liste de toutes les familles de P.L.D :

III.3.1. Les P.A.L.s :

Signifie Programmable Array Logic, c'est à dire réseau logique programmable ou bien les E.P.L.D. Ce qui signifie Erasable Programmable Logic Device, c'est à dire circuit logique programmable et effaçable La programmation de ces circuits s'effectue par destruction de fusibles. On trouve U.V.P.R.O.M aux P.R.O.M. Les E.P.L.D peuvent être effacés par U.V. ou électriquement. Ils sont encore appelés P.A.L. CMOS.

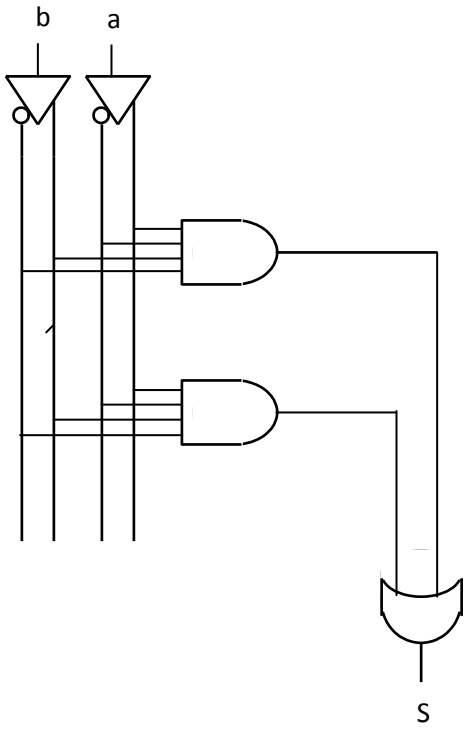


Fig. III.4. Structure simplifiée d'un PAL

La représentation schématique de la précédente structure demande beaucoup d'espace pour représenter un *P.A.L.* en entier. Les industriels ont adopté une autre représentation voir ci-dessous.

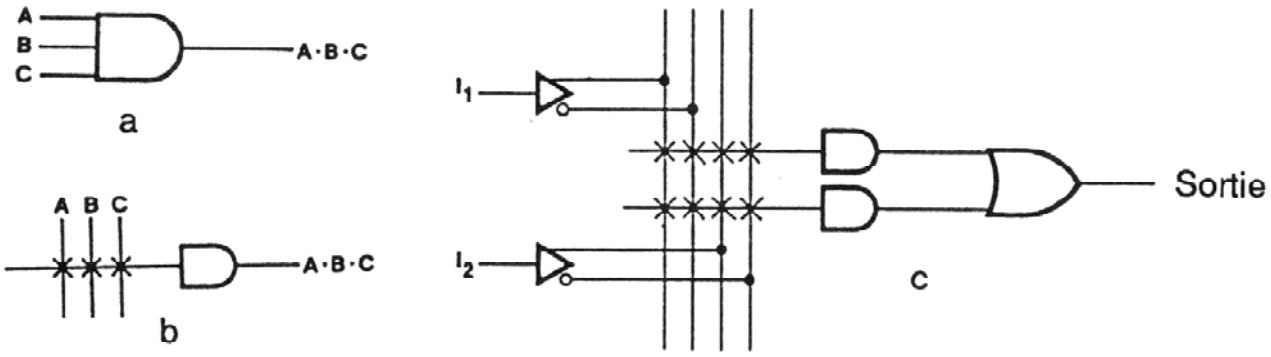


Fig. III.5. Autre structure d'un PAL

a: Porte ET à 3 entrées.
b: Porte ET à 3 entrées représentation P.A.L. les croix représentent les fusibles intacts.
c: Représentation de la structure interne d'un P.A.L.

Exemple:

Realisation d'un OU exclusive

I1 + I2 = I1. I2 + I1. I2

Cette équation se représente de la façon suivante :

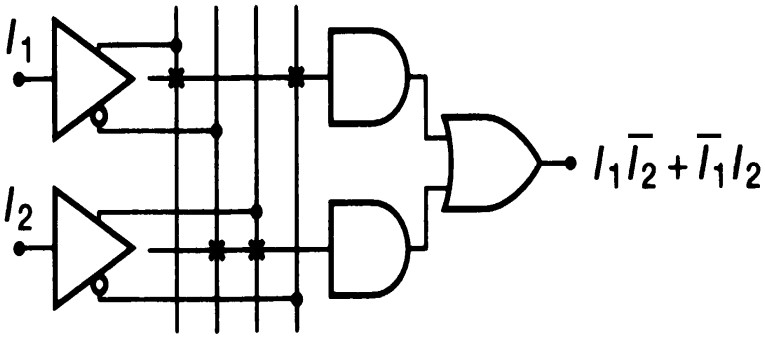


Fig. III.6. Réalisation de ou exclusive

III.3.1.1. Les différentes structures :

III.3.1.1.1. Structure générale :

D'un point de vue fonctionnel un P.A.L. est constitué d'une zone d'entrée de fusibles ou matrice de programmation et une structure de sortie non programmable déterminant le type de circuit voir schéma ci-dessous.

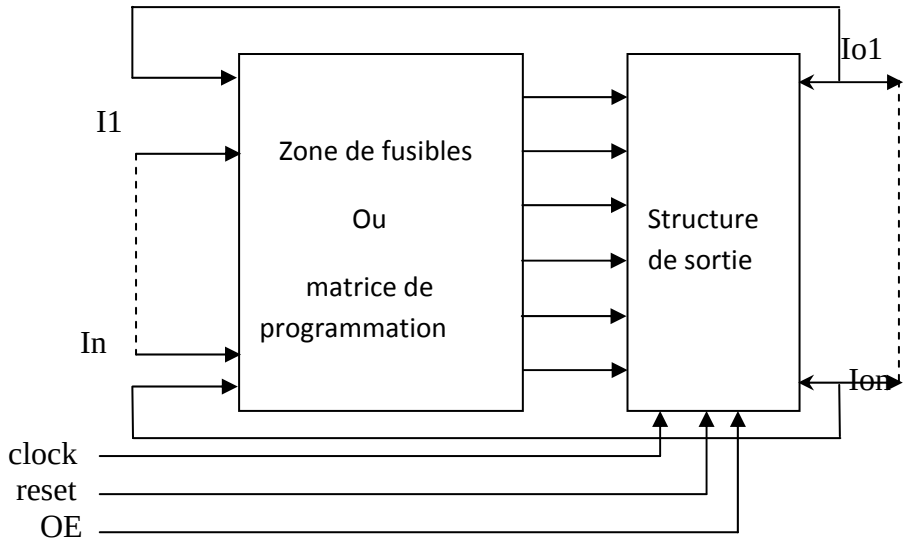


Fig. III.7. Structure générale d'un PAL

Dans l'exemple précédant, la programmation du OU exclusif était facilement réalisable, ce n'est pas toujours le cas. C'est pourquoi il existe un grand nombre de *P.A.L.* utilisant des structures de sorties différentes. On peut distinguer trois types de structures de base:

- Combinatoire.
- Séquentielle.

- Versatile.

III.3.1.1.2. Référence et nomenclature des P.A.Ls :

III.3.1.1.2.1.PALs, AMD :

PAL (CE) XX AB YY C ZZ DEF

- DEF : Type de boîtier
- ZZ : Vitesse
- C : Consommation
- YY : Nombre de sorties
- AB : Structure de sortie
- XX : Nombre d'entrées
- CE : CE pour version CMOS
- PAL

Lettre(s)code(s)	Structure de sortie
L	Combinatoire active bas
H	Combinatoire active haut
C	Combinatoire complémentaire
R	Registre synchrone (D)
RA	Registre asynchrone
X	Registre et OU exclusif
V	Versatile

Fig. III.8. Référence et nomenclature des P.A.Ls

III.3.1.1.2.2.PALs, TEXAS instrument :

TIBCE PAL XX AB YY ZZ CDEF

- CDEF : Boîtier, gamme, etc..
- ZZ : Vitesse
- YY : Nombre de sortie
- AB : Structure de sortie
- XX : Nombre d'entrées
- PAL : Nombre d'entrées
- TIBCE : Famille Bpolaire, CMOS, ECL

Exemple : PAL 16 L 8 H 15 PC

Type de boîtier : DIL plastique civile

Vitesse : 15 nS

Consommation : ½ puissance

Nombre de sorties : 8

Structure de sortie : Combinatoire active Bas

Nombre d'entrées : 16

III.3.2. Les G.A.L.s :

Ce qui signifie Generic Array Logic ou encore réseau logique générique ce qui veut dire les E.E.P.R.O.M aux P.R.O.M. Le nom de G.A.L. a été déposé par LATTICE SEMICONDUCTOR.

Leur fonctionnement est identique aux P.A.L.CMOS, ils sont programmables et effaçables électriquement.

Par soucis de remplacer les PALs, LATTICE a équipé la plupart de ses GALs de macro-cellules programmables permettant d'émuler n'importe quel PAL. Ces structures de sortie sont donc du type « Versatile » (V)

III.3.2.1. Référence des G.A.Ls :

Le nombre de types de GAL est de 8. Les deux 2 derniers-nés présentent une structure plus particulière que nous n'aborderons pas dans ce document. Les six plus anciens sont différenciés par leurs nombres d'entrées et de sorties. Ils possèdent une structure de sortie soit du type « Versatile » soit du type « Registre asynchrone ».

Référence :

GAL XX AB YY ZZ C DEF G

Nombre de sortie
Structure de sortie
Nombre d'entrées
Version : « », Commerce. - « I », Indus.
Boîtiers : « P », DIP - « J », PLCC
Consommation : « L », ½ P - « Q », ¼ P
Vitesse (nS)

III.3.3. Les C.P.L.Ds :

Ce qui signifie Complex Programmable Logic Device. Ces circuits sont composés de plusieurs P.A.L.s élémentaires reliés entre-eux par une zone d'interconnexion. Grâce à cette architecture, ils permettent d'atteindre des vitesses de fonctionnement élevées (plusieurs centaine de Mhz). Le nombre de portes peut varier entre 100 et 100 000 portes logiques et entre 16 et 1000 bascules.

III.3.3.1. Structure générale d'un C.P.L.D :

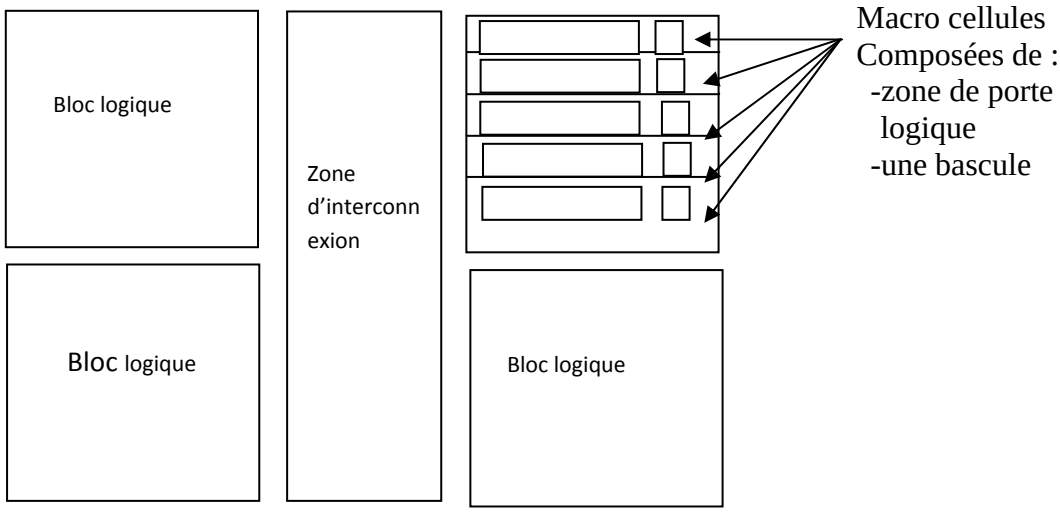


Fig. IV.9. Structure générale d'un C.P.L.D

III.3.4. Les L.C.A. & F.P.G.A. à anti-fusible :

- Les L.C.A. Ce qui signifie Logic Cell Array ou encore réseau de cellules logiques. Ces circuits sont composés de blocs logiques élémentaires de 2000 à 10000 portes que l'utilisateur peut interconnecter.

- Les F.P.G.A. à anti fusibles sont identiques aux L.C.A**.** Ces circuits peuvent être composés de plusieurs milliers voire de millions de portes logiques et de bascules. Les dernières générations de FPGA intègrent même de la mémoire vive (RAM). Les deux plus grands constructeurs de FPGA sont XILINX et ALTERA.

De plus en plus les capacités des C.P.L.Ds et des F.P.G.As se rapprochent. Le principal critère de choix entre les deux familles est la vitesse de fonctionnement. En effet les C.P.L.Ds acceptent des fréquences de fonctionnement beaucoup plus élevées que F.P.G.As.

III.4. Programmation des P.L.Ds :

La programmation des P.L.Ds nécessite un logiciel adapté pour le développement du programme et un programmateur permettant de « griller » les fusibles de circuit. Il est conseillé de suivre la démarche décrite par l'organigramme suivant

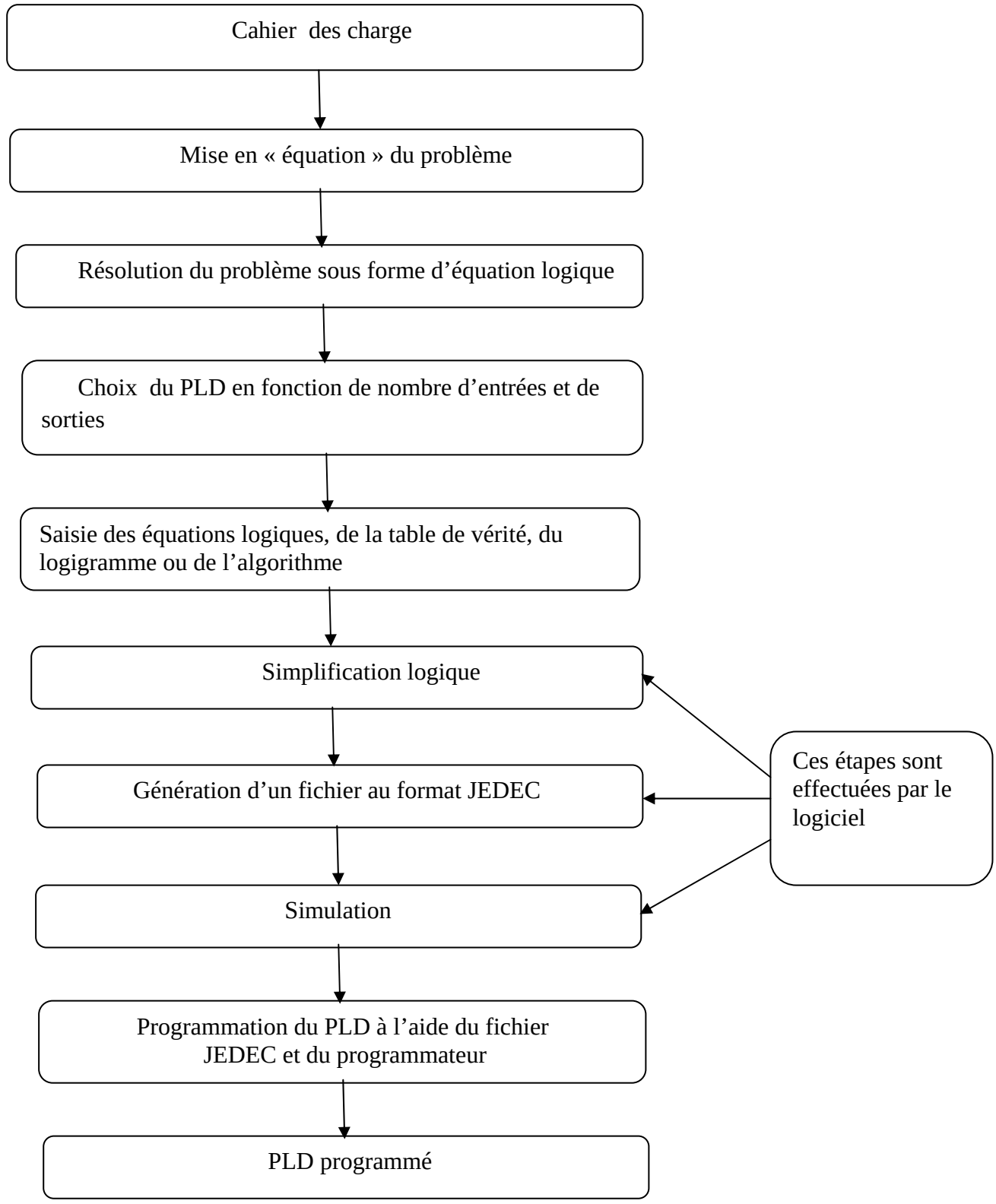


Fig. III.11. Schéma fonctionnel d'un outil de développement des P.L.Ds:

* Le logiciel de développement permet de simplifier les équations et de générer un fichier JEDEC à partir des données rentrées par l'opérateur. Il simule aussi le fonctionnement du PLD avec le

programme obtenu. Le logiciel « WinCUPL » permet d'effectuer ces Opérations pour la programmation des PLD

* Le fichier JEDEC est un ensemble de données binaires indiquant au programmateur les fusibles à « griller ».

* Le programmateur permet de « griller » les fusibles du PLD en fonctions des données du fichier JEDEC. Il est en général associé à un logiciel de pilotage.

III.5. application :

III.5.1. position du problème :

La régulation du trafic routier du croisement présente sur la figure ci-dessous est gérée par 4 feux. Ils fonctionnent par paires, Fh; Fv.

Chacun dispose de 3 lampes, vertes, oranges, rouges. 6 signaux logiques sont nécessaires a leur pilotage : Fhv ; Fho ; Fhr ; Fvv ; Fvo ; Fvr

4 capteurs de trafic sont positionnés de part et d'autre du carrefour. Ils fonctionnent également par paires.

Une première paire délivre un signal logique Vh qui vaut 1 si une voiture est détectée aux feux Ouest ou Est.

Une deuxième paire délivre un signal logique Vv qui vaut 1 si une voiture est détectée aux feux Nord ou Sud.

Lorsqu'une voiture est détectée sur l'axe horizontal quand le feu est au vert sur l'axe vertical, Fv passe à l'orange pendant 5 secondes puis passe au rouge alors que Fh passe au vert. Nous avons le comportement oppose lorsqu'une voiture est détectée sur l'axe vertical quand le feu est au vert sur l'axe horizontal

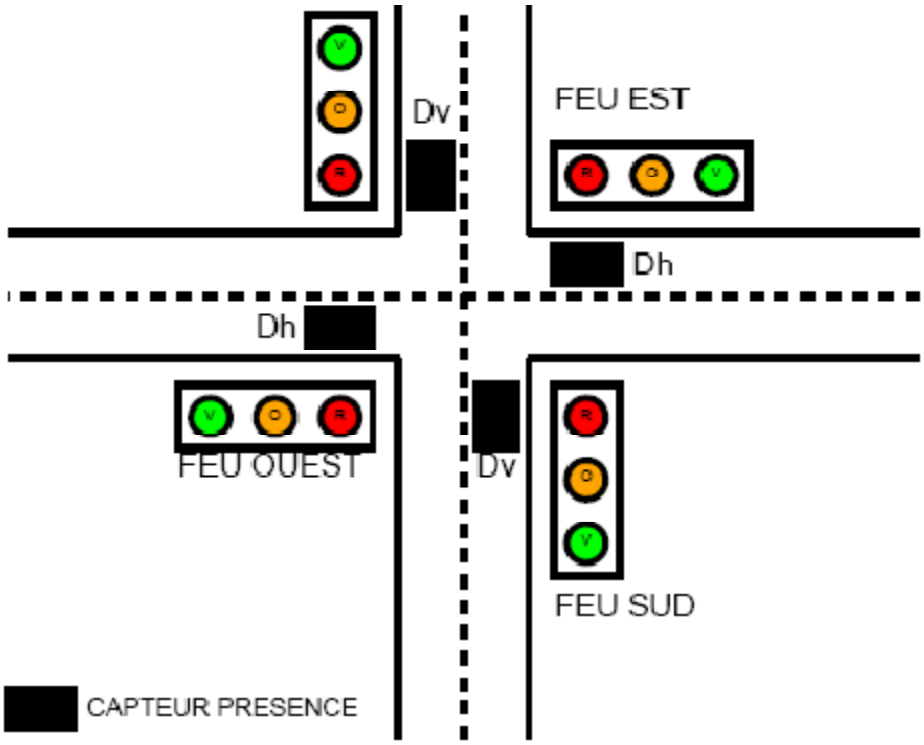


Fig.III .1. Feux de carrefour

III.5.2. Le fonctionnement :

Lorsqu'une voiture est détectée sur l'axe horizontal quand le feu est au vert sur l'axe vertical, Fv passe à l'orange pendant 5 secondes puis passe au rouge alors que Fh passe au vert. Nous avons le comportement opposé. Lorsqu'une voiture est détectée sur l'axe vertical quand le feu est au vert sur l'axe horizontal.

III.5.3. table et diagramme des états :

III.5.3.1. Réalisation de la table des états :

	Fhv	Fho	Fhr	Fvv	Fvo	Fvr
S_0	**1**	**0**	**0**	**0**	**0**	**1**
S_1	**0**	**1**	**0**	**0**	**0**	**1**
S_2	**0**	**0**	**1**	**1**	**0**	**0**
S_3	**0**	**0**	**1**	**0**	**1**	**0**

Tab.III.1.La table d'état de feux de carrefour

III.5.3.2. Le diagramme des états :

D'après la table des états précédente on aura 4 états à représenter (s_0,s_1,s_2,s_3),le premier états comportera le feu vert horizontal (vh) et rouge vertical (rh),le deuxième orange horizontal (oh) et rouge vertical (rv),le troisième états vert vertical (vv) et rouge horizontal (rh) et en fin le quatrième état y'aura l'allumage de feux orange vertical et rouge horésental.et le passage d'un état à un autre état est générer par les deux entrées correspondente aux deux capteurs (Dh) et (Dv) tel que (Dh) vaut 1 lors du passage de l'état initial (s_0) à l'état suivant (s_1),et (Dv) correspond au passage du l'état (s_2) à l'état (s_3) et le passage de l'état (s_1) à l'état (s_2) et de l'état (s_3) à (s_0) est réalisé a l'aide d'une remise à 1 (set).

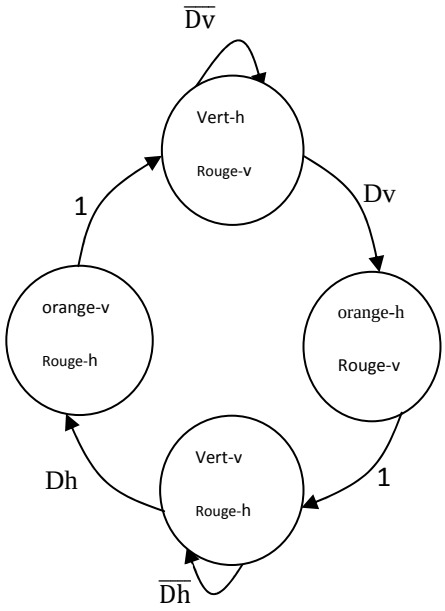
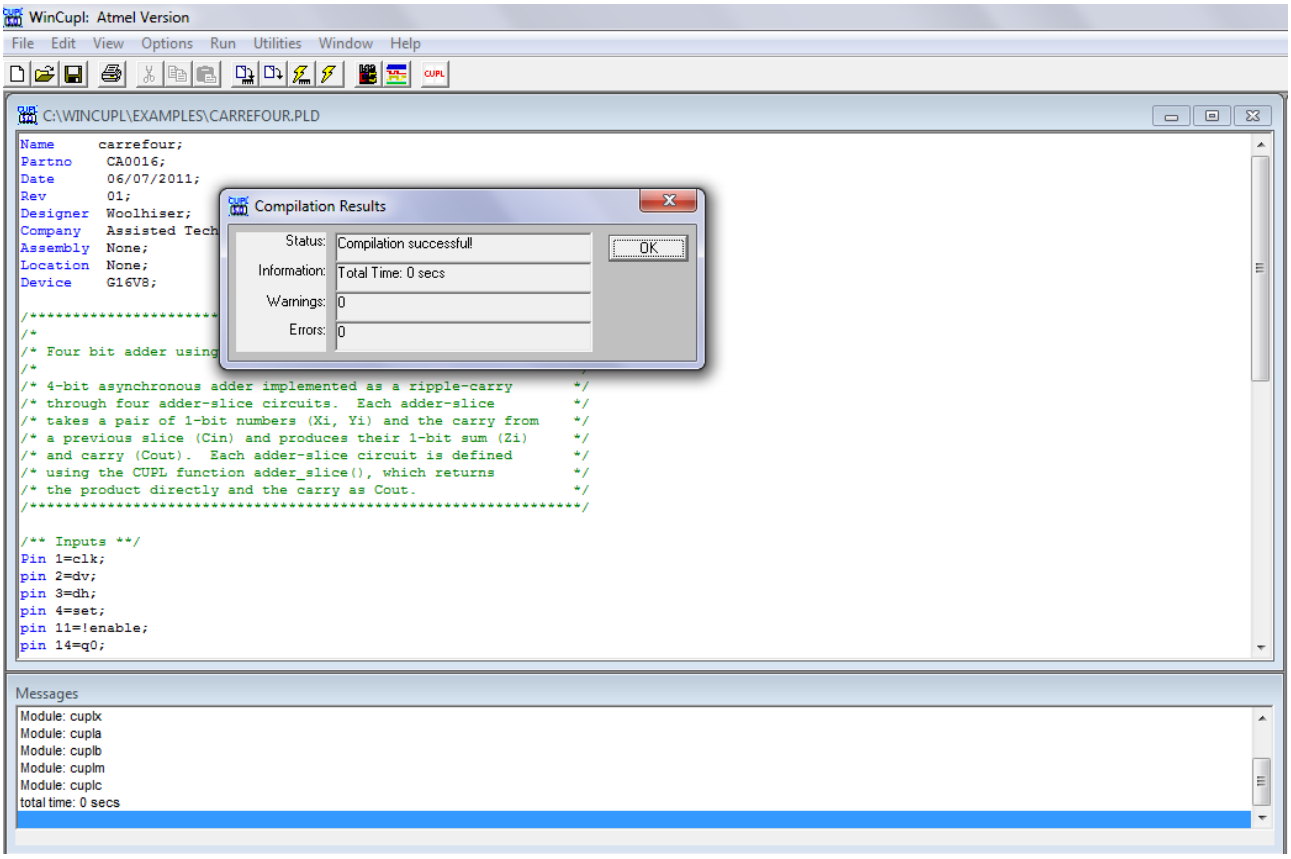


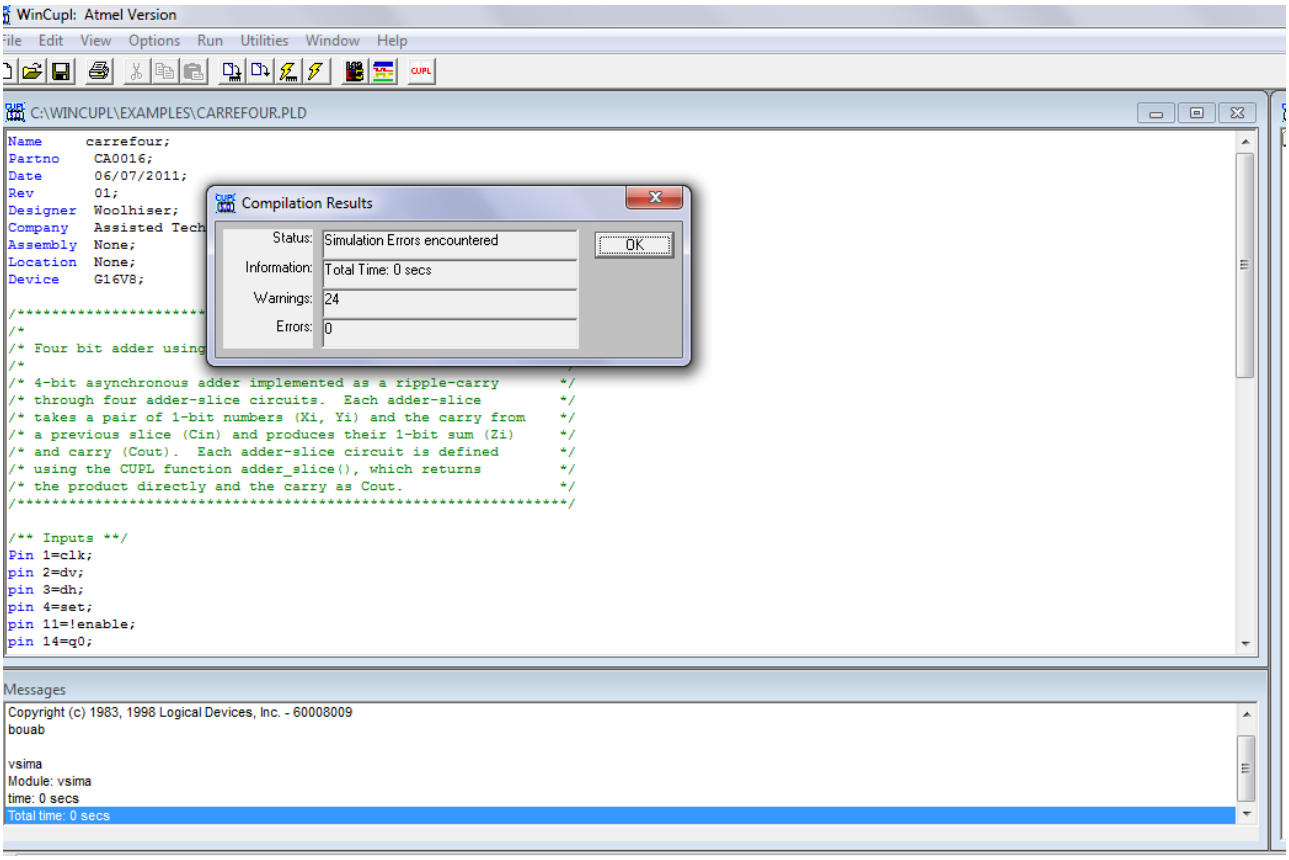
Fig.III.2. graphe d'état de feux de carrefour

III.5.4. programmation sous Win CUPL (machines à états):

Compilation :



Simulation :



```
Name     carrefour;
```

Le fichier PLD (carrefour.pld) :

```
Partno   CA0016;
Date     06/07/2011;
Rev      01;
Designer  BOUAB;
Company   Assisted Technology;
Assembly  None;
Location  None;
Device    GAL 16V8;

/** Inputs **/
Pin 1=clk;
pin 2=dv;
pin 3=dh;
pin 4=set;
pin 11=!enable;
pin 14=q0;
pin 15=q1;
/** Outputs **/
pin 16=z;
pin 5=vh;
pin 6=rv;
pin 7=oh;
pin 8=rh;
pin 9=vv;
pin 12=ov;
field state = [q1..0];
$define s0 'b' 00
$define s1 'b' 01
$define s2 'b' 10
$define s3 'b' 11
sequence state {
present s0  if dv next s1;
            default next s0;
present s1  if set next s2;
present s2  if dh next s3;
         default next s2;
present s3  if set next s0 out z;
}
```

Le fichier.doc (carrefour.doc) :

```
*******************************************************************************
                                        carrefour
*******************************************************************************

CUPL(WM)         5.0a Serial# 60008009
Device           gal 16v8ms  Library DLIB-h-40-11
Created          Mon Sep 19 11:38:53 2011
```

```
Name            carrefour
Partno          CA0016
Revision        01
Date            06/07/2011
Designer        BOUAB
Company         Assisted Technology
Assembly        None
Location        None

================================================================
=============
                              Expanded Product Terms
================================================================
=============

q0.d  =>
    dv & !q0 & !q1
  # dh & !q0 & q1

q1.d  =>
    q0 & !q1 & set
  # !q0 & q1

state =>
    q1 , q0

z.d  =>
    q0 & q1 & set

================================================================
=============
                                   Symbol Table
================================================================
=============

Pin Variable                                        Pterms   Max
Min
Pol   Name              Ext     Pin     Type     Used   Pterms
Level
--- --------            ---     ---     ----    ------  ------  --
---

    clk                         1        V        -       -
-
    dh                          3        V        -       -
-
    dv                          2        V        -       -
-
 !  enable                     11        V        -       -
-
    oh                          7        V        -       -
-
```

```
    ov                          12       V        -       -
-
    q0                          14       V        -       -
-
    q0                  d       14       X        2       8
4
    q1                          15       V        -       -
-
    q1                  d       15       X        2       8
4
    rh                          8        V        -       -
-
    rv                          6        V        -       -
-
    set                         4        V        -       -
-
    state                       0        F        -       -
-
    vh                          5        V        -       -
-
    vv                          9        V        -       -
-
    z                           16       V        -       -
-
    z                   d       16       X        1       8
4


LEGEND    D : default variable          F : field      G : group
          I : intermediate variable     N : node       M : extended
node
          U : undefined                 V : variable   X : extended
variable
          T : function


================================================================
=============
                                    Fuse Plot
================================================================
=============

Syn   02192 x Ac0   02193 -

Pin #19  02048  Pol x  02120  Ac1 -
 00000 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00032 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00064 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00096 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00128 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00160 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00192 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00224 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
Pin #18  02049  Pol x  02121  Ac1 -
 00256 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00288 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00320 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00352 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00384 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00416 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00448 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00480 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #17  02050  Pol x  02122  Ac1 -
 00512 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00544 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00576 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00608 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00640 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00672 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00704 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00736 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #16  02051  Pol -  02123  Ac1 x
 00768 --------x---------x---x---------
 00800 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00832 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00864 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00896 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00928 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00960 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00992 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #15  02052  Pol -  02124  Ac1 x
 01024 --------x----------x--x---------
 01056 ------------------x----x--------
 01088 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01120 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01152 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01184 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01216 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01248 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #14  02053  Pol -  02125  Ac1 x
 01280 x------------------x---x--------
 01312 ----x-------------x----x--------
 01344 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01376 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01408 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01440 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01472 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01504 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #13  02054  Pol x  02126  Ac1 -
 01536 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01568 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01600 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01632 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01664 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01696 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01728 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
 01760 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #12  02055  Pol x  02127  Ac1 -
 01792 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01824 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01856 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01888 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01920 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01952 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01984 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 02016 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx


LEGEND    X : fuse not blown
          - : fuse blown

=================================================================
=============
                                 Chip Diagram
=================================================================
=============

                          _______________
                         |  carrefour    |
                   clk x---|1           20|---x Vcc
                    dv x---|2           19|---x
                    dh x---|3           18|---x
                   set x---|4           17|---x
                    vh x---|5           16|---x z
                    rv x---|6           15|---x q1
                    oh x---|7           14|---x q0
                    rh x---|8           13|---x
                    vv x---|9           12|---x ov
                   GND x---|10          11|---x !enable
                         |_______________|
```

Annexe. Utilisation simplifié de WinCUPL

1.1. Introduction :

Ce logiciel permet à partir d'un ficher PLD de créer un fichier JEDEC Permettant de programmer un GAL ou un PLD, PAL grâce à un programmateur universel. Le fichier JEDEC contient la liste des fusibles à détruire et à conserver.

Le fichier PLD décrit :

Le circuit GAL utilisé,

Le brochage du GAL ou un PAL.

La logique.

1.2. WinCUPL :

WinCUPL nous permet de créer le fichier PLD. Une fois le fichier PLD crée et compilé, un fichier JEDEC (.JED) est crée.

1.3. Création du fichier PLD :

1.3.1. Structure :

La forme du fichier PLD est :

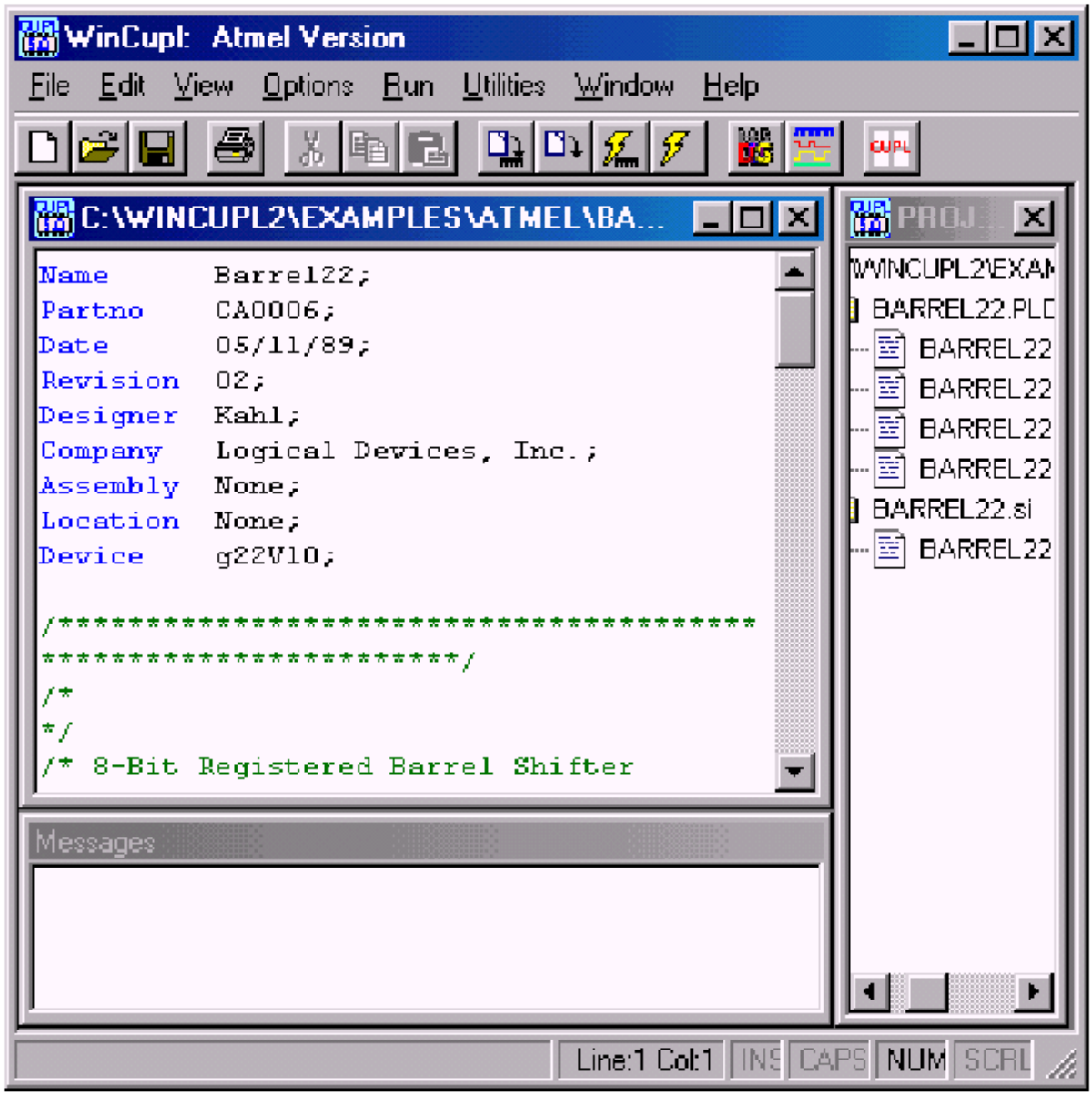
- Entête du document : description et commentaires /* *commentaires*/*

-Association des numéros de broche avec les noms des variables logiques des équations.

-Définitions des variables intermédiaires

-Equations logiques.

Description des fenêtres :



1.3.2. Exemple de fichier PLD : BCD 7 segments :

```
Name      BCD 7segment;
Partno    CA0016;
Date      11/05/2011;
Rev       01;
Designer  yakoubi;
Company   Assisted Technology;
Assembly  None;
Location  None;
Device    G16V8;

/** Inputs **/

Pin 2=x;
```

```
Pin 3=y;
pin 4=z;
pin 5=u;
/** Outputs **/
pin 12=a;
pin 13=b;
pin 14=c;
pin 15=d;
pin 16=e;
pin 17=f;
pin 18=g;

field input = [x,y,z,u];
field output = [a,b,c,d,e,f,g];

table input => output{
'b'0000 => 'b'1111110;
'b'0001 => 'b'0110000;
'b'0010 => 'b'1101101;
'b'0011 => 'b'1111001;
'b'0100 => 'b'0110011;
'b'0101 => 'b'1011011;
'b'0110 => 'b'1011111;
'b'0111 => 'b'1110000;
'b'1000 => 'b'1111111;
'b'1001 => 'b'1111011;
}
```

1.3.3. Syntaxe autorisée :

1.3.3.1. Les opérateurs logiques et arithmétiques :

opérateur	exemples	description	priorité
!	!A	NOT	1
&	A&B	AND	2
#	A#B	OR	3
$	A$B	XOR	4

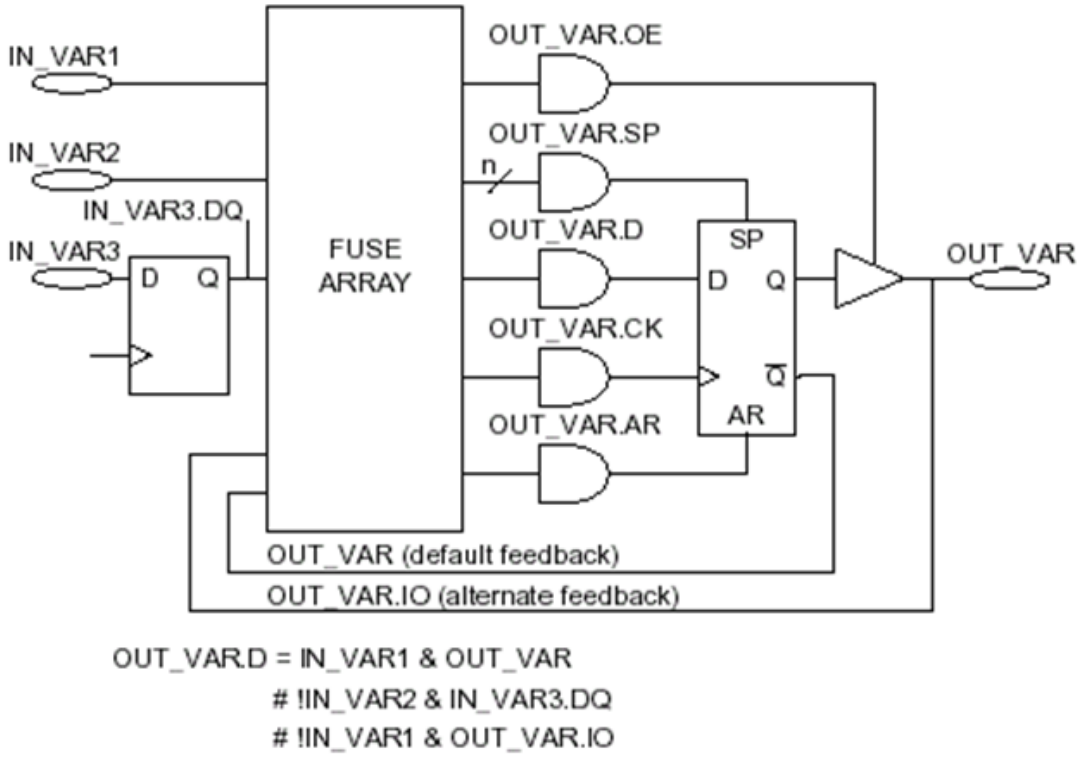
Table .V.1.1. Les opérateurs logiques

opérateur	exemples	description	priorité
**	2**3	Exponentiation	1
*	2*1	Multiplication	2
/	4/2	Division	2
%	9%8	Module	2
+	2+4	Addition	3
-	4-1	soustraction	3

Table. V.1.2. les opérateurs arithmétiques

1.3.3.2. Les extensions :

Les extensions définissent les fonctionnalités des entrées et sorties.




```
OUT_VAR.D = IN_VAR1 & OUT_VAR
          # !IN_VAR2 & IN_VAR3.DQ
          # !IN_VAR1 & OUT_VAR.IO
```

Figure.1circuit illustrant les extensions

prolongation	Coté utilisé	description
AR	L	asynchrone reset de flip-flop
CK	L	Programmable clock de bascule (flip-flop)
D	L	D input of D-type flip-flop
DQ	R	Q output of D type flip-flop
OE	L	Progrrammable output enable
SP	L	Synchrone preset enable
IO	R	Pin fedback path selexion

Table. V.1.3. PLD/CPLDS extensions de variable

1.4. Quelques exemples de fonctions logiques :

De nombreux exemples sont fournis dans le dossier : /Example/atmel/ la liste des exemples est contenue dans le fichier « Atmel Example List.pdf».

Exemples de portes logiques :

```
Name        Gates;
Partno      CA0001;
Revision    04;
Date        9/12/89;
Designer    belkadi;
Company     Logical Devices, Inc.;
```

```
Location        None;
Assembly        None;
Device          g16v8a;
/*************************************************************/
/*                                                           */
/*      This is a example to demonstrate how CUPL            */
/*      compiles simple gates.                               */
/*                                                           */
/*************************************************************/
/*
* Inputs:  define inputs to build simple gates from
*/
Pin 1 =  a;
Pin 2 =  b;
/*
* Outputs:  define outputs as active HI levels
*
*/
Pin 12 = inva;
Pin 13 = invb;
Pin 14 = and;
Pin 15 = nand;
Pin 16 = or;
Pin 17 = nor;
Pin 18 = xor;
Pin 19 = xnor;
/*
* Logic:  examples of simple gates expressed in CUPL
*/
inva = !a;            /* inverters */
invb = !b;
and  = a & b;          /* and gate */
nand = !(a & b);       /* nand gate */
or   = a # b;          /* or gate */
nor  = !(a # b);       /* nor gate */
xor  = a $ b;          /* exclusive or gate */
xnor = !(a $ b);       /* exclusive nor gate */
```

Exemple détecteur de séquence:

```
Name      sequence;
Partno    CA0016;a
Date      25/05/2011;
Rev       01;
Designer  bouab;
Company   Assisted Technology;
Assembly  None;
Location  None;
Device    G16V8;

/** Inputs **/
```

```
Pin 1=clk;
pin 2=x;
pin 11=!enable;
pin 14=q0;
pin 15=q1;
/** Outputs **/
pin 16=z;
field state = [q1..0];
$define s0 'b' 00
$define s1 'b' 01
$define s2 'b' 10
$define s3 'b' 11

sequence state {
present s0  if x next s1;
              default next s0;
present s1  if x next s1;
          default next s2;
present s2  if x next s3;
          default next s0;
present s3  if x next s1;
          default next s2 out z;}
```

Le programme pour projet:

```
Name     carrefour;
Partno   CA0016;
Date     06/07/2011;
Rev      01;
Designer   BOUAB;
Company   Assisted Technology;
Assembly  None;
Location  None;
Device    Gal 16V8;

/** Inputs **/
Pin 1=clk;
pin 2=dv;
pin 3=dh;
pin 4=set;
pin 11=!enable;
pin 14=q0;
pin 15=q1;
/** Outputs **/
pin 16=z;
pin 5=vh;
pin 6=rv;
pin 7=oh;
pin 8=rh;
pin 9=vv;
```

```
pin 12=ov;
field state = [q1..0];
$define s0 'b' 00
$define s1 'b' 01
$define s2 'b' 10
$define s3 'b' 11
sequence state {
present s0  if dv next s1;
              default next s0;
present s1  if set next s2;
present s2  if dh next s3;
          default next s2;
present s3  if set next s0 out z;
}
```

Conclusion

Notre travail a porté sur la synthèse d'un système séquentiel avec différentes méthodes (machines à états finis, réseaux de pétri et le grafcet), l'implémentation d'une application pour valider notre étude s'est fait sous l'environnement wincupl d'ATMEL

Un aperçu sur les systèmes séquentiels synchrones et asynchrones à été donné.

Dans le second chapitre on a traité les méthodes de synthèses, à savoir, les machines à états finis, les réseaux de Pétri et le grafcet. Ces méthodes sont indispensables dans le cas des systèmes complexes. Une étude comparative a été établie en se basant sur un cas pratique.

Au troisième chapitre on a vu les circuits logiques programmables, l'accent à été mis sur les PALs et les GALs. Le principe de fonctionnement de ce type de circuit est expliqué en détail. Les différentes étapes de la programmation du GAL16V8 ont été passées en revue. Dans ce chapitre nous avons posé le problème suivant : la gestion de feux de carrefour.

On a posé le problème puis on a étudié le fonctionnement ce qui nous a permis de tirer la table et le diagramme des états, ensuite nous sommes passé à la programmation en utilisant une machine à états finis décrivant le fonctionnement du carrefour. Une fois le programme sous winCupl compilé, le fichier Jedec permettant la programmation physique via le programmateur Data main du GAL16V8 est généré.

Nous devons reconnaitre que le sujet qui nous a été proposé a grandement contribué à l'enrichissement de nos connaissances, relevant aussi bien de l'aspect théorique que l'aspect pratique puisqu'il nous a permis d'employer un nouveau langage de description de haut niveau tel que WINCUPL. Il nous a permis aussi d'acquérir des connaissances sur les circuits programmables.

Nous espérons que nous avons apporté une petite contribution au domaine des circuits logiques programmables et nous souhaitons que les futurs ingénieurs contribueront à son enrichissement et son amélioration.

Bibliographie

Les livres :

NKESTA.A, « Informatique industrielle, circuits logique programmables, mémoires, PLD, CPLD, FPGA » Editions Ellipses, 1998

TAVENIER, CHRISTIAN « Circuits logique programmable » Editions DUNOD, 1996

Nicolas BERNARD, «Machine synchrone» Revue 3EI, n° 30, Septembre 2002, pp. 24-39

CIMELLI, R. BOURGERON «Guide du technicien en électronique » HACHETTE, Paris, 1995

P.GIRARD « Construction des machines séquentielles industrielles» Edition : Dunod 1973

THELLIEZ SYLVAIN « Pratique séquentielle et réseaux de pétri » Edition : Ayrolles 1978

GINDRE MARCEL « Logique séquentielle : cours et exercices » Editions : Ediscience international, 1994

GINDRE MARCEL « Logique combinatoire et technologie VOL.1, cours et exercices » Editions : Ediscience international, 1992

BOUTEILLE NOEL « Le grafcet : AFCET-ADEPA » Editions : CEPAD, 1995

MERENO.S « Le grafcet : conception –implantation dans les automates programmables » Editions : castilla 1996

DAVID RENE « Du grafcet aux réseaux de pétri » Editions : Hermes, 1978

MILSANT .M « Automatismes à séquences » Editions : Eyrolles, 1979

Les thèses :

Naak Houria, Hacid Hasinaet Moussouni Fadhila « Implémentation d'un variateur de vitesse à base d'un circuit FPGA » thèse pour l'obtention d'ingénieure d'états en électronique Mouloud Mammeri Tizi-Ouzou 2007

Chibah Arezki « conception d'un contrôleur d'étage de puissance par FPGA » pour l'obtention de diplôme de magistère en électronique réalisé par Mouloud Mammeri Tizi-Ouzou