Fiche Systèmes a Evènement Discret (SED) Module 2

# Mise en œuvre de systèmes séquentiels logiques

Position du problème : Analyse Conception Mise En Œuvre (MOE) Prototypage Exploitation

Une MOE c’est : le choix d’une cible, le choix d’un codage[[1]](#footnote-2), la synthèse logique[[2]](#footnote-3) et la réalisation.

## Codage par minimisation du nombre de variables internes

Le nombre de bits du code des états est minimal minimisation du coût de mise en œuvre / complexification des équations

|  |  |
| --- | --- |
| Mode pulsé : Choisir un code de bits tel que puis affecter à chaque état un code binaire de bits ( pour chaque état) | |
| Mode fondamental :  (Attention aux courses critiques[[3]](#footnote-4))   * Choisir un code de bits tel que * Affecter un code bit à chaque état en respectant les contraintes d’adjacence | Une image contenant table  Description générée automatiquement |

## Codage 1 parmi

Le nombre de bits de est égal au nombre d’états du système simplification des équations / coût de mise en œuvre plus important. Par convention, chaque code ne comporte qu’un seul bit à 1 (le rang de la composante à 1 indique le numéro de l’état actif). La déduction des équations est plus simple.

|  |  |  |
| --- | --- | --- |
|  |  | Terme d’excitation + terme de maintien |

|  |  |
| --- | --- |
| Mode pulsé : Equations avec terme de  : |  |
| Mode fondamental : Le codage parmis n’est pas adjacent. Conventionnellement, on impose une course qui consiste à mettre à la correspondant a l’état suivant puis à éteindre la correspondant à l’état présent. Equations avec terme de  : |

# Mise en œuvre sur cible matérielle

Intérêt : Vitesse, coût, encombrement

Types de mis en œuvre :

* Par circuits combinatoires (ET, OU, XOR, NAND)
* Par Bascules (D, JK, RS)
* Par PLD (EPROM, PAL, CPLD, FPGA)

Une entrée supplémentaire indispensable : le signal d’initialisation

## Circuits combinatoires

|  |  |
| --- | --- |
| A base de porte combinatoire, c’est un type de mise en œuvre seulement adaptée au mode de fonctionnement fondamental.  Principe   * Chaque et chaque sortie est mise en œuvre par un bloc combinatoire * Les architectures électroniques (logigramme) des blocs sont déduites de la représentation algébrique  ( pour , pour ) | Méthode (minimisation des )   * Coder avec respect des adjacences * Construire les de chaque ET de chaque Sortie * Déduire une représentation algébrique * Déduire les logigrammes |

## Bascules

|  |  |  |
| --- | --- | --- |
| Choisir le type de bascule en fonction du mode fonctionnement.   * Mode pulsé D, JK, T * Mode fondamental RS   Une bascule = une  Utilisation des entrées asynchrones et des bascules pour l’initialisation | Une image contenant diagramme  Description générée automatiquement | Méthode   * Coder * Construire les tables d’excitation des bascules (en utilisant la table d’évolution de la bascule choisie) * Construire les * Déduire les équations d’excitation des bascules et celles des sorties * Déduire le logigramme |

Mode pulsé

Exemple : le malaxeur sans arrêt d’urgence (bascules ) : Minimisation du nombre de

|  |  |
| --- | --- |
| Une image contenant diagramme, table  Description générée automatiquement | . Ici, 1 seul évènement qu’on associe au des bascules |
| Une image contenant diagramme, table  Description générée automatiquement |

Exemple : le malaxeur avec arrêt d’urgence, réalisation 1 parmi avec bascules

|  |  |  |
| --- | --- | --- |
| Une image contenant diagramme, schématique  Description générée automatiquement | Une image contenant texte, lettre  Description générée automatiquement | Attention : Ne marche que si on peut garantir que les signaux associés aux évènements ici et ne sont pas à 1 au même instant    On « rate » un évènement. Si on était en le système va en au lieu de  Sinon : passer en mode fondamental |

Mode fondamental : Exemple de la perceuse

|  |  |  |
| --- | --- | --- |
| via bascule (1 parmi ) | Synchronisé () d’un mode fondamental | |
| Une image contenant diagramme  Description générée automatiquement    Attention aux problèmes de boucle ! | Pour éviter le problème de courses critiques   * On peut ajouter une entrée évènementielle périodique. Échantillonneur bloqueur * La fréquence de doit être suffisamment importante pour ne pas rater des évolutions significatives du vecteur d’entrée * La fréquence de afin d’échantillonner que lorsqu’il a atteint un état stable. * La vitesse de réaction du système est donc plus lente | Exemple de  par bascule  Une image contenant schématique  Description générée automatiquement |

## Utilisation de circuits logiques programmable

|  |  |
| --- | --- |
| Une image contenant diagramme  Description générée automatiquement | 1er Principe de    Aléas sur les entrées synchrone seulement  Taille mémoire :  Mot Adresses  Mot Mémoire :  La mémoire croît exponentiellement avec le nombre d’entrées  Méthode   * Coder * Faire le plan mémoire * Eventuellement, ajouter une combinatoire sur les sorties |

Exemple perceuse d’un mode fondamental

|  |  |  |
| --- | --- | --- |
| Une image contenant diagramme  Description générée automatiquement |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 2ème Principe de  :  Pour rendre la teille mémoire indépendante du nombre d’entrées   |  |  | | --- | --- | | Hypothèse |  |   Taille mémoire :  Adresses  Mot mémoire :  Méthode   * Modifier le graphe * Coder les états * Coder les évènements multiplexeurs * Faire le plan mémoire * Eventuellement ajouter une combinatoire sur les sorties | Architecture |

## Utilisation de circuits logique programmable : (Programmable Logic Array)

|  |  |
| --- | --- |
| Architecture   * Matrice ET programmable * Macrocellule * Technologie « Fuse » * Langage « PLDShell » * Simulation * Description des équations et du graphe |  |

Exemple perceuse (voir MAE page précédente)

|  |  |
| --- | --- |
| **CHIP** Perceuse PLD22V10  ; Définition des entrées  PIN 1 H  PIN 2 Start  PIN 3 A  PIN 4 B  PIN 5 C  PIN 6 D  PIN 7 INIT  ; Définition des sorties  PIN 22 Desc  PIN 21 Mont | **EQUATIONS**  ; Description des excitations des bascules D  Y1.D:=A\*Start\*/Y1\*/Y2+/B\*/Y1\*Y2+Y1\*Y2\*/(C\*P)+Y1\*Y2\*/D  Y1.CLKF=H  Y1.RSTF=INIT  Y2.D:=A\*Start\*/Y1\*/Y2+B\*/Y1\*Y2+Y1\*Y2\*(/C+P)  Une image contenant diagramme  Description générée automatiquementY2.CLKF=H  Y2.RSTF=INIT  ; Description des sorties  Desc=Y1\*/INIT  Mont=(/Y1\*(Y2+/A))\*/INIT  **SIMULATION**  ; Description d’un scénario d’évolution des entrées (A,B,C,D,Start,Init) |

## Utilisation de circuits logiques programmable : CPLD[[4]](#footnote-5) et FPGA[[5]](#footnote-6)

|  |  |
| --- | --- |
| Architecture d’un CPLD   * Difficulté de créer des PLD possédants beaucoup d’entrées * Idée : Intégrer plusieurs PLD |  |
| Architecture d’un FPGA   * Un constructeur origine : Xilinx * Même idée que CPLD mais le nombre de PLD intégré est beaucoup plus important[[6]](#footnote-7) * PLD de base (Logic Block) est + simple (~1-2macrocells) * Plusieurs « Switch Matrix » et « IO Block » Flexibilité * Technologie « Fuse » ou « SRAM » * Pb Routage ! |  |

## Un outil de synthèse : VHDL[[7]](#footnote-8)

Norme IEEE (1987) : Source VHDL Compilation Simulation Synthèse Placement [[8]](#footnote-12)+ Routage Fabrication

|  |  |
| --- | --- |
| Vue externe | Vue interne |
| Une image contenant diagramme  Description générée automatiquement | Une image contenant diagramme  Description générée automatiquement |

|  |  |  |
| --- | --- | --- |
| Vue externe | Vue interne | Vue interne |
| Une image contenant diagramme  Description générée automatiquement | Une image contenant diagramme  Description générée automatiquement |  |
|  | Instructions concurrentes : Réalisation combinatoire Les affectations se font en parallèle | Instructions séquentielles : Notion de PROCESS Réalisation synchrone Les affectations se font en séquence et ne sont répercutés en sortie qu’à la fin du process (horloge) |

Exemple

|  |  |
| --- | --- |
| asynchrone de MAE en VHDL | synchrone de MAE en VHDL |
|  | Une image contenant diagramme  Description générée automatiquementUne image contenant texte  Description générée automatiquement |
| synchrone abstraite de MAE en VHDL | |
|  | |

Principe et chronogramme d’exécution

|  |  |
| --- | --- |
| Cycle programme   * Lecture Entrée () * Calcul des états suivants () * Calcul des sorties ()   API + Microcontrôleur + PC (avec périphériques E/S) |  |

|  |  |
| --- | --- |
| Mise en œuvre synchrone | Mise en œuvre asynchrone |
|  |  |
| Les entrées doivent avoir la même valeur durant la totalité du cycle programme elles sont échantillonnées | Si la commutation de toutes les a lieu à la fin du bloc alors la mise en œuvre est synchrone |

Exemple : Malaxeur avec arrêt d’urgence synchrone, 1 parmi (Langage C)

Une image contenant texte

Description générée automatiquement

Une image contenant diagramme

Description générée automatiquementExemple : Malaxeur avec arrêt d’urgence

|  |  |  |
| --- | --- | --- |
| asynchrone, minimisation du nombre de (Langage C) | synchrone, minimisation du nombre de (Langage C) | synchrone, directe de la MEF (Langage C) |
|  | Une image contenant diagramme, schématique  Description générée automatiquement | Une image contenant texte, lettre  Description générée automatiquement |

1. Définition du nombre de d’un code binaire à chacun des états [↑](#footnote-ref-2)
2. Détermination d’une représentation « algébrique » du système [↑](#footnote-ref-3)
3. Si la course ne permet pas de se stabiliser sur l’état désiré pour les éviter 2 états voisins disposent de codes adjacents [↑](#footnote-ref-4)
4. Complex Logical Device [↑](#footnote-ref-5)
5. Field Programmable Gate Array [↑](#footnote-ref-6)
6. Plusieurs milliers dans un FPGA contre quelques dizaines dans un CPLD [↑](#footnote-ref-7)
7. VHSIC Hardware Description Language [↑](#footnote-ref-8)
8. Calcul des temps de propagations [↑](#footnote-ref-12)