Ecole polytechnique de l'université de Nice-Sophia Antipolis

Elec4 – ASICs
TP n°2

Objectifs Etude des techniques d'implémentation des machines à états

I Introduction

En décrivant les FSM à partir d'un langage de description de matériel (HDL), le concepteur peut se concentrer sur le problème de la modélisation des séquences d'opérations sans avoir à se préoccuper de l'implémentation du circuit qui peut être laissée aux bons soins de l'outil de synthèse. Mais le concepteur doit connaître le modèle à respecter (template) de manière à ce que l'outil infère correctement la structure souhaitée pour implémenter la machine à état. La figure 1 ci-dessous rappelle le modèle général d'une machine d'état à structure de Moore. On distingue un registre d'état permettant de mémoriser l'état courant de la FSM (bloc M). Une FSM synchrone impose l'utilisation d'une source d'horloge permettant de synchroniser les changements d'états. Les changements d'états sont élaborés par un bloc *next state logic(F)*, bloc fonction de l'état courant et de l'ensemble éventuellement vide des entrées. Un bloc éventuel *output logic* s'occupe de la génération des sorties(G). Il est fonction de l'état courant (et d'une ou plusieurs entrées de la machine dans le cas d'une machine de Mealy qui ne seront pas étudiées ici).

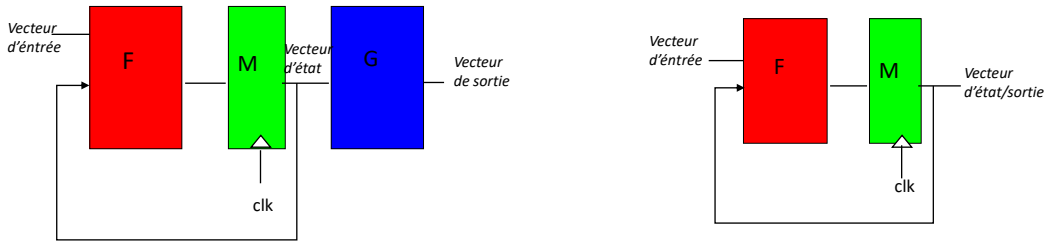


Figure 1: Modèles d'une machine d'état de Moore

Un des problèmes inhérents à l'implémentation des machines d'état porte sur le problème du codage. Lors de la description de la FSM, le concepteur utilise une notation symbolique pour référencer les états internes. Ceci permet de faire abstraction des affectations des codes aux différents états, affectations qui ne sont en effet pas essentielles pour vérifier la fonctionnalité de la machine. Precision synthesis, comme beaucoup d'outils de synthèse, affecte les codes durant le processus de synthèse, en respectant un style de codage défini par le concepteur. Parmi les nombreux styles de machine d'état, nous nous intéresserons aux résultats des stratégies les plus communément adoptés :

- le codage binaire (binary) : codage généralement retenu pour les CPLD, il génère une machine d'état minimisant le nombre d'éléments de logique séquentielle (FF). Par exemple, 4 états demandent 2 FF. Le codage binaire est généralement utile lorsque le critère d'optimisation souhaité porte sur la surface, les performances n'étant pas jugées critiques. Des règles heuristiques peuvent permettre de déduire un code permettant de minimiser la logique combinatoire

- le codage une bascule/état (OneHot) : codage généralement retenu pour les FPGA, il nécessite autant de FF que d'états. Par exemple, 4 états exigent 4 FF. Ce type de synthèse fournit généralement les meilleures performances et les délais d'apparition des sorties les plus courts, mais souvent au prix d'une surface plus importante.

- Le codage par les sorties (Moore) : codage qui affecte des codes aux états en fonction des combinaisons des sorties tout en s'assurant que les codes sont discriminants, ce qui peut nécessiter éventuellement l'ajout d'un ou plusieurs bits supplémentaires. Il permet en général de minimiser le temps de décodage des sorties.

II Le bloc *controleurRS232C*

On rappelle que ce bloc a pour objectif de gérer les signaux de contrôle et d'état du modem selon une spécification proche de l'interface RS232C. Ce type d'interface permet la jonction d'un modem avec un service de transmission de données série type UART. La figure ci-dessous montre les signaux de composant l'interface du bloc:

- *Dtr* : signal positionné à bas (à '0') lorsqu'on souhaite mettre le modem en mode opérationnel
- *Dsr* : signal indiquant que le modem est opérationnel en réponse à *dtr*
- *Rts* ; signal positionné à bas pour commuter en mode émission.
- *cts :* signal indiquant à bas qu'on est en mode émission en réponse à *rts*
- *m:* mode de transmission (0 :halfduplex,1 :fullduplex)
- *mad,mam :* signal de contrôle des blocs démodulateur et modulateur

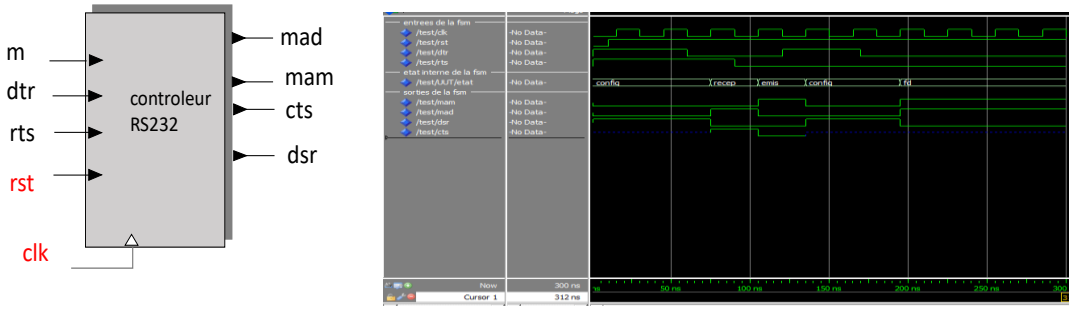


Figure 2: Symbole et chronogramme de spécification du bloc

La figure donne également un exemple de scénario de réponse attendu pour ce bloc. A la mise sous tension, le modem est en mode configuration (DSR vaut 1). Une fois en mode opérationnel (DTR=0), le modem se place en état de réception car le mode de transmission est half-duplex(m=0). Un passage en mode émission se caractérise par la mise à 0 de la sortie *cts*. On observe ensuite un retour en mode configuration et un passage en mode opérationnel full-duplex.

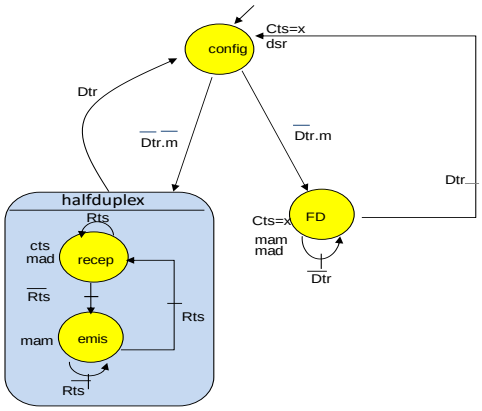


Figure 3: Symbole et diagramme d'état (machine de Moore) du circuit

La figure 2 donne une spécification du comportement du circuit sous forme d'un diagramme d'état. Seules les sorties à 1 ou *don't care* sont spécifiées. La logique de contrôle nécessite 4 états sachant que le mode half-duplex est décrit par un macro état (en bleu) formé de 2 états élémentaires. Le retour en mode de configuration s'effectue lorsque *dtr* vaut *1* et ce, quel que soit l'état interne courant du macro-état. Ce signal est par ailleurs prioritaire par rapport à *rts*. On notera que le signal *cts* ne doit passer en mode émission qu'au bout d'un temps de retournement du modem (environ 30 ms). Ceci peut être pris en compte au niveau de la période d'horloge qui définit les instants de transition des états.

III Préparation *(a faire viser par l'enseignant en début de séance)*

A. Donnez une description de l'entite *ctrlrs232c* en considérant tous les signaux de type *std_logic.*

B Proposez une première description d'architecture à partir du diagramme d'état dans un fichier *fsmsymb.vhd.* La définition des états s'effectuera sous forme symbolique grâce à un type énuméré.

type defetat **is** (config,recep,emis,fd**) ;**

L'état initial sera positionné sur *rst* vrai à l'état bas.

C. Proposez une nouvelle description de l'entité dans un fichier *fsmmoore*. Les états seront définis sous forme numérique dans cette description et *defetat* sera donc transformé en sous-type :

subtype defetat **is** std_logic_vector(2 **downto** 0);

Le code des états devra donc être défini de manière que les valeurs des sorties *mad* et *mam* puissent être directement déterminées par la valeur d'un bit du registre d'état, tout en satisfaisant l'impératif d'avoir des codes discriminants pour ces états. On parle encore de codage de moore ou par les sorties. *C*haque état élémentaire sera défini à partir d'une constante sur le modèle ci-dessous :

Constant idle : defetat:="…..”; -- combinaison des sorties à affecter

Le code VHDL sera décomposé en 3 parties : un process mémoire, un process de calcul de l'état suivant et des sorties *cts* et *dsr*. Des instructions d'affectations concurrentes simples mettant en relation les sorties *mad* et *mam* avec un des bits du registre d'état. Par exemple,

mad<=etat(0) ;

permettrait de relier la sortie *mad* directement au bit 0 de la machine d'états.

D. Définissez un testbench de validation commune à ces deux architectures dans *testfsm.vhd*. On prendra une période d'horloge de 10 ns.

E. Définissez un script de simulation RTL. Pour visualisez le '-', le radix de *cts* devra être positionné en *symbolic*.

F. Définissez un script de synthèse de votre circuit(sans placement routge). On définira les directives d'optimisation suivantes :

```
create_clock { clk } -name clk -period 1000.0
set_input_delay 0.300 -clock clk {m(*)}
set_output_delay 6.0 -clock clk {mad mam}
set_false_path -clock clk -to {cts dsr}
set_false_path -clock clk -from {rts dtr}
```

de manière à fixer une contrainte d'horloge et l'input delay des signaux depuis le bloc *display* et l'output delay des signaux de contrôle des blocs *modulateur* et *démodulateur*, ainsi qu'une indication d'asynchronisme pour le reste des signaux. Les signaux du blocs (hormis *mad* et *mam*) seront par ailleurs affectés aux pattes de sorties comme sur la figure ci-dessous.

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength
clk	Input	PIN_N2	2	B2_N1	PIN_N2	3.3-V LV...default)		24mA (default)
cts	Output	PIN_AE22	7	B7_N0	PIN_AE22	3.3-V LV...default)		24mA (default)
dsr	Output	PIN_AF22	7	B7_N0	PIN_AF22	3.3-V LV...default)		24mA (default)
dtr	Input	PIN_N25	5	B5_N1	PIN_N25	3.3-V LV...default)		24mA (default)
m	Input	PIN_V2	1	B1_N0	PIN_V2	3.3-V LV...default)		24mA (default)
mad	Output				PIN_T19	3.3-V LV...default)		24mA (default)
mam	Output				PIN_U20	3.3-V LV...default)		24mA (default)
rst	Input	PIN_W26	6	B6_N1	PIN_W26	3.3-V LV...default)		24mA (default)
rts	Input	PIN_N26	5	B5_N1	PIN_N26	3.3-V LV...default)		24mA (default)

Figure 4: Affectations des pattes aux signaux du bloc

G. Définissez un script de simulation post-routage de votre circuit.

IV Travail en séance

A. Validation fonctionnelle (Modelsim)

Validez les 2 descriptions sous l'outil de simulation à l'aide de votre testbench.

B. Compilation et synthèse des design (Precision)

Sous *precision*, définissez un projet *TP2* en renommez l'implémentation *en fsmsymb.*

B.1 Codage binaire

Cochez le codage binaire dans les options d'input de synthèse à partir du menu *tools→Options->input options*).

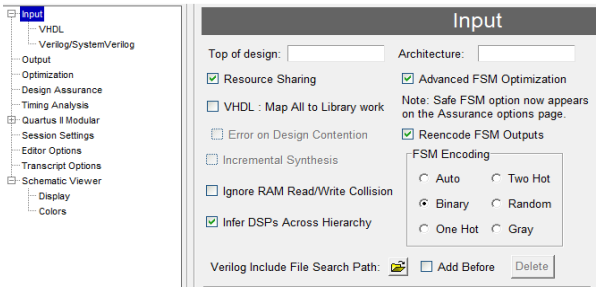


Figure 5: modification de l'option de codage de la fsm

Vous pouvez également integrer l'option dans la directive *setup* de votre script.
Lancez votre script de synthèse et sauvegardez les différents résultats des schémas

Question : Combien d'éléments mémoire trouve-t-on sur le schéma RTL pour le registre d'état? Justifiez le résultat

A l'aide de l'outil *trace backward→to registers* isolez le bloc de calcul des sorties (bloc G) ,*mam* et *mad* sur le schéma RTL depuis la sortie vers les registres. Sauvegardez. Procédez à la synthèse et Vérifiez la bonne prise en compte des directives d'optimisation.

A l'aide des outils de rapport de surface et timing, relevez le nombre de LC, LUT, registres, la fréquence maximale et le slack de la solution générée.

C.2 Codage one hot

Modifiez le codage des états en cochant *one-hot* cette fois ci et décochez l'option *Reencode FSM outputs.* Compilez et synthétisez.

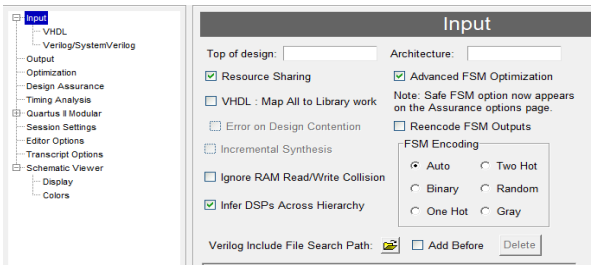
Question Combien d'éléments mémoire observe-t-on à présent sur le schéma RTL pour le registre d'état? Justifiez

Isolez le bloc de sortie en se plaçant sur les registres et cliquant sur *trace backward → to outputs*.

A l'aide des outils de rapport de surface et timing, relevez le nombre de LC, LUT, registres, la fréquence maximale et le slack de la solution générée.

C.3 Codage de moore

Créez une nouvelle implémentation *fsm_moore* et chargez l'entité ainsi que le fichier *fsmmoore.vhd*.




```
> setup_design -reencode_fsm_outputs=false
```

Figure 6: Modification de l'option de réencodage

Décochez l'option *Reencode FSM outputs*idans le menu *options.* Lancez votre script de synthèse.

Question Combien d'éléments mémoire observe-t-on à présent sur le schéma RTL pour le registre d'état ? Justifiez

Isolez à nouveau le bloc de calcul de la sortie sur le schéma et vérifiez notamment l'absence de bloc de sortie pour *cts* et *dtr* (le modèle de FSM en partie droite de la figure 1).

A l'aide des outils de rapport de surface et timing, relevez le nombre de LC, LUT, registres, la fréquence maximale et le slack de la solution générée.

D. Placement routage (Precision)

Lancez le placement routage et vérifiez les résultats de surface et de timing (*fitter* et *timequest*) retournés par *quartus* à l'issue de la synthèse physique.

E. Simulation post routage (ModelSim)

Une fois *modelsim* lancé, exécutez votre script de simulation post routage. Attention éventuellement à la configuration (for) dans le fichier de test que vous pourrez mettre en commentaire ou modifier avec *structure* comme nom d'architecture à simuler. Comparez (éventuellement avec waveform compare) avec le résultat de la simulation RTL pour cette simulation effectuée après implémentation.

F. Test du bloc (Quartus)

Branchez votre carte et programmez votre circuit. En vous reportant à la figure 7 validez le circuit sur la plaque d'expérimentation en testant différentes combinaisons d'entrée de *dtr*,*rts* et *m* et en observant les effets sur les sorties matérialisées par des diodes.

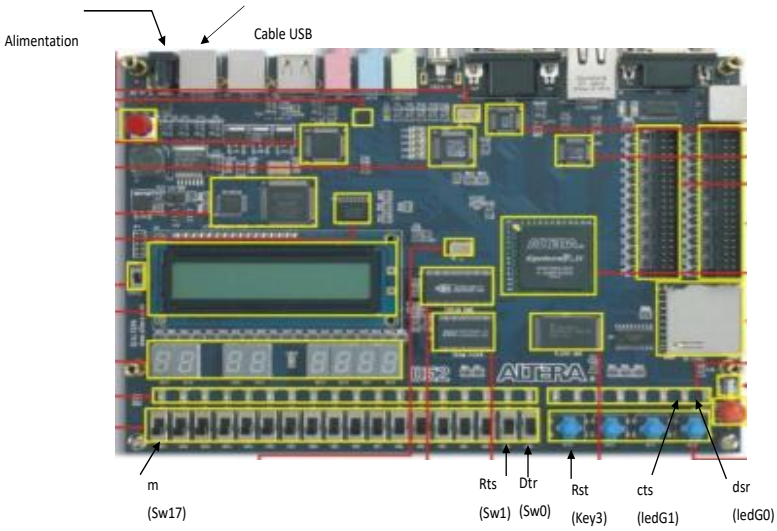


***Figure 7**: Plaque d'expérimentation pour le test du circuit*

Capturer quelques résultats des tests effectués à l'aide de l'appareil photo d'un portable.