TP1 : Synthèse d'automates d'états finis

1. Avant-propos
2. 1 Introduction
3. 2 Automates de Moore et Mealy
4. 3 SYF et VHDL
5. 4 Exemple
6. 5 Réalisation du compteur
7. 6 Automate pour digicode
8. 7 Réalisation du digicode

Avant-propos

Le but de ces quatre séances de TP est de présenter quelques outils de la chaîne ALLIANCE dont :

> Les outils de synthèse logique SYF, BOOM, BOOG, LOON; L'outil de génération de data-path Stratus ; L'éditeur graphique de netlist XSCH; Les outils pour la preuve formelle FLATBEH, PROOF; Le simulateur ASIMUT;

Chaque outil possède ses propres options donnant des résultats plus ou moins adaptés suivant l'utilisation que l'on veut faire du circuit. fsm SYF vbe BOOM vbe BOOG vst LOON vst synthèse optimisation synthèse optimisation automate comportemental comportemental netlist netlist

FIG. 1 Ensemble de la synthèse

Les dépendances de données dans le flux sont matérialisées dans la réalité par une dépendance de fichier. Le fichier Makefile exécuté à l'aide de la commande make permet de gérer ces dépendances. Différents exemples de fichiers seront fournis durant le TP. Ce TP n'étant pas un cours sur le Makefile, nous nous limiterons à expliquer l'usage qui en est fait dans les exemples fournis. (cf. Annexe 1)

L'usage de Makefile est obligatoire pour ce TP ! ! !

1 Introduction

> Un circuit combinatoire pur ne dispose pas

de registres internes. De ce fait, ses sorties ne dépendent que de ses entrées primaires.

A l'inverse, un circuit séquentiel synchrone disposant de registres internes voit ses sorties changer en fonction de ses entrées mais aussi des valeurs mémorisées dans ses registres.

En conséquence, l'état du circuit à l'instant t+1 dépend aussi de son état à l'instant t. Ce type de circuit peut être modélisé par un automate d'états finis.

FIG. 2 Automate d'états

2 Automates de Moore et Mealy

L'automate de MOORE voit l'état de ses sorties changer uniquement sur front d'horloge. Les entrées peuvent donc bouger entre deux fronts sans modifier les sorties. Par contre dans le cas d'un automate de MEALY, la variation des entrées peut modifier à jour

3 SYF et VHDL

Afin de décrire de tels automates, on utilise un style particulier de description

VHDL qui définit l'architecture "fsm" (finite-state machine).

Le fichier correspondant possède également l'extension fsm. A partir de ce fichier,

l'outil SYF effectue la synthèse d'automate et transforme cet automate abstrait en un réseau booléen. SYF génère donc un fichier VHDL au format vbe. Comme la plupart des outils utilisés au laboratoire, il faut positionner certaines variables d'environnement avant d'utiliser SYF. Pour les connaître, reportez-vous au man de syf.

4 Exemple

Afin de se familiariser avec la syntaxe de description d'un fichier .fsm, un exemple de compteur de trois "1" successifs est présenté. Sa vocation est de détecter par exemple sur une liaison série une séquence de trois "1" successifs. Le graphe d'états que l'on cherche à décrire est représenté sur la figure . Le format fsm est également décrit dans une page man. Pensez à la consulter.

Figure Graphe d'états d'un compteur de trois "1" successifs

```
entity circuit is port (
        ck, i, reset, vdd, vss : in bit ; o : out bit ) ;
end circuit ; architecture MOORE of circuit is
        type ETAT_TYPE is (E0, E1, E2, E3) ; signal EF, EP : ETAT_TYPE; -- pragma
        CURRENT_STATE EP -- pragma NEXT_STATE EF -- pragma CLOCK CK
begin process (EP, i, reset) begin
        if (reset=?1?) then
                EF<=E0;
        else
                case EP is
                        when E0 =>
                                if (i=?1?) then
                                        EF <= E1 ;
```

```
                else
                        EF <= E0 ;
                end if ;
        when E1 =>
                if (i=?1?) then
                        EF <= E2 ;
                else
                        EF <= E0 ;
                end if ;
        when E2 =>
                if (i=?1?) then
                        EF <= E3 ;
                else
                        EF <= E0 ;
                end if ;
        when E3 =>
                if (i=?1?) then
                        EF <= E3 ;
                else
                        EF <= E0 ;
                end if ;
        when others => assert (?1?)
                        report "etat illegal";
    end case ;
end if ;
    case EP is
        when E0 =>
```

```
                o <= ?0? ;
            when E1 =>
                o <= ?0? ;
            when E2 =>
                o <= ?0? ;
            when E3 =>
                o <= ?1? ;
            when others => assert (?1?)
                report "etat illegal" ;
        end case ;
end process ;
    process(ck) begin
        if (ck=?1? and not ck?stable) then
            EP <= EF ;
        end if ;
    end process ;
end MOORE ;
```

5 Réalisation du compteur

Ecrire la description d'un compteur de cinq "un" successifs sous la forme d'un

automate de Moore.

Positionner les variables d'environnement. Lancer SYF avec les options de codage -a, -j, -m, -o, -r et en utilisant les options

-CEV.

```
>syf -CEV -a <fsm_source> -
```

Un fichier Makefile vous est fourni pour vous faciliter la vie !

répertoire : ~trncomun/TP/2006/TP2/Fichiers/Fournis/Compteur5/. Utilisez ce Make file à compléter.

Visualiser les fichiers .enc.

Ecrire un fichier de vecteurs de test et simuler sous ASIMUT. Que se passe-t-il si le reset n'est pas positionné en début de pattern ? Pourquoi ?

6 Automate pour digicode

L'exemple qui suit servira dans toute la suite du TP. On veut réaliser une puce pour digicode . Les spécifications sont les suivantes :

Les chiffres de 0 à 9 sont codés en binaire naturel sur

la manière suivante :

A : 1010 B : 1011 Le digicode fonctionne en deux modes : Mode Jour : La porte s'ouvre en appuyant sur "O" Mode Nuit : La porte ne s'ouvre que si le code est

Pour distinguer les deux cas un "timer" externe calcule entre 8h00 et 20h00 et '0' sinon.

Le digicode commande une alarme dès qu'un des chiffres entrés n'est pas le bon L'automate du digicode revient dans son état d'attente si rien n'est entré au clavier

au bout de 5 secondes ou si l'alarme a sonné pendant 2mn- signal reset- Pour cela il reçoit un signal reset du timer externe.

La puce fonctionne à une fréquence de 10MHz. Toute pression d'une touche du clavier est accompagn Celui-ci signale à la puce que les données en sortie

signal est à 1 durant un cycle d'horloge. Le code est 53A17 L'interface de l'automate est le suivant :

in ck in reset in jour in i[3 :0] in O in press_kbd out porte out alarm

7 Réalisation du digicode

Dessiner le graphe d'états de l'automate. (Les corrections seront distribuées)

Le décrire au format .fsm .

Le synthétiser avec SYF en utilisant les options de codage -a, -j, -m, -o, -r et en

utilisant les options -CEV.

```
>syf -CEV -a <fsm_source> -
```

Ecrire le fichier .pat de vecteurs de test.

Simuler avec ASIMUT toutes les vues comportementales obtenues. Adaptez le Makefile (répertoire : ~trncomun/TP/2006/TP2/Fichiers/Fournis/digicode/)

pour qu'il couvre tous les encodages possibles.

Quelles sont vos remarques concernant la complexité des expressions (i.e temps)

et le nombre de registres (i.e surface) des descriptions comportementales suivant les encodages ? En déduire les deux groupes d'encodage.

Comparer aussi leurs nombres de littéraux.