Electronique numérique

TP n°3: Conception des FSMD

Objectif : Implémentation de circuit à partir de transferts de registres

I Introduction

L'approche utilisée dans les 2 LABs précédents est ce qu'on appelle une approche par les combinaisons. Cette approche consiste à définir les combinaisons de sortie en fonction des combinaisons d'entrées et des combinaisons des états internes. Cette approche est à réserver à des circuits peu complexes (circuits orientés timing). Pour des circuits plus complexes, généralement caractérisés par un comportement que l'on peut exprimer par un algorithme, l'approche par les activités (ou **transfert de registre**) est alors préconisée. Cette approche conduit à une solution architecturale conçue sur le modèle des FSMD (cf figure 1)

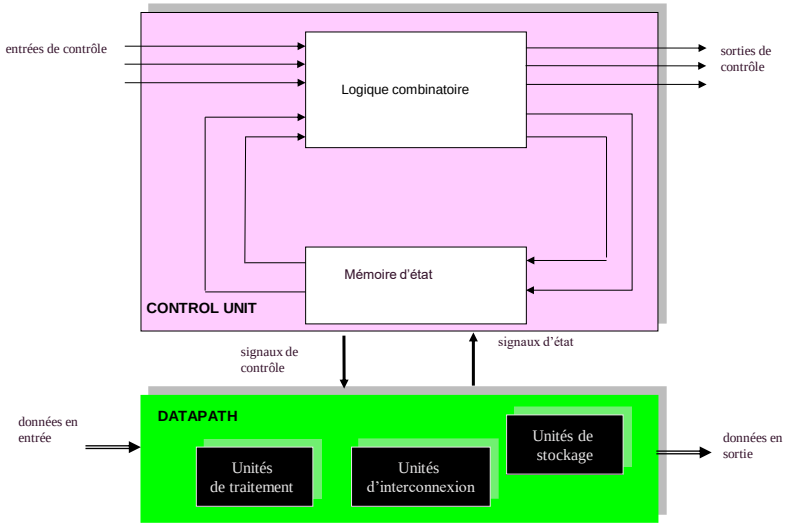

Figure 1: Modèle générique des FSMD

Dans cette approche, l'algorithme a pour particularité de décrire les opérations sur des variables, opérations placées sur des états (pas de contrôle) d'un séquenceur. Les opérations sont alors implémentées dans un chemin de donnes (datapath) dont le rôle est d'exécuter les opérations grâce à des éléments mémoires (registres, RAM, etc) , des opérateurs arithmétiques, logiques, de décalage (*,+,shift, etc) et des éléments d'interconnexion (multiplexeurs, portes 3 états). Les opérations sont séquencées par une unité de contrôle conçue sur le modèle des FSM et déterminant l'instant d'activation d'une ou plusieurs opérations (transferts) en fonction de l'état du chemin de données (indicateurs d'état) et des entrées du circuit. Dans ce LAB, on se propose d'illustrer cette technique de conception à partir d'un circuit transmetteur série asynchrone réalisant la transmission série d'une donnée parallèle fournit par un microprocesseur. Ce circuit, une fois simulé fonctionnellement, sera testé à l'analyseur logique interne au composant programmable.

II Le circuit transmetteur série

Dans les systèmes numériques, la fonction de transmission série est une fonction essentielle pour transmettre des données d'un site de production de ces données vers un site d'exploitation de ces données, sur des distances plus ou moins longues. Selon le débit de la transmission, 2 types de transmission sont exploitables :

- Transmission série **asynchrone** : la synchronisation entre l'émetteur et le récepteur est effectuée à chaque caractère émis grâce à un bit de *start* et un ou plusieurs bits de stop (cf circuit étudié en cours).
- Transmission série **synchrone** : la synchronisation est assurée en permanence (à chaque bit). Cette synchronisation peut se faire avec une horloge commune ou dans le signal des données.

Le second type de transmission permet de meilleurs débits mais demande une circuiterie plus élaborée. La figure ci-dessous montre le symbole du circuit transmetteur série.

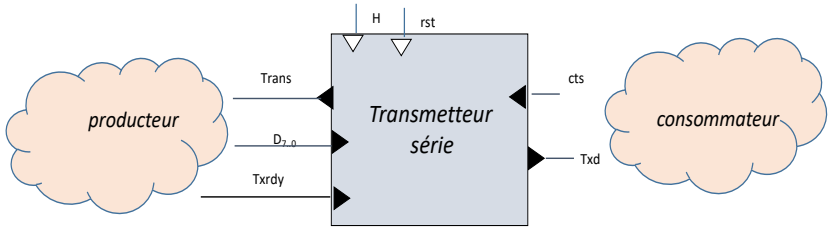

Figure 2: Symbole du circuit transmetteur série asynchrone

Les signaux du circuit sont les suivants :

- *D :* Donnée parallèle sur 8 bits à transmettre en série.
- *Txrdy :* permet d'indiquer qu'une nouvelle donnée *D* est disponible pour transmission
- *Trans :* signal indiquant le début de la transmission en série de la donnée D
- *Cts :* le consommateur indique qu'il est prêt (cts=0) ou non à recevoir une donnée.
- *Txd :* Donnée binaire véhiculant le signal en série

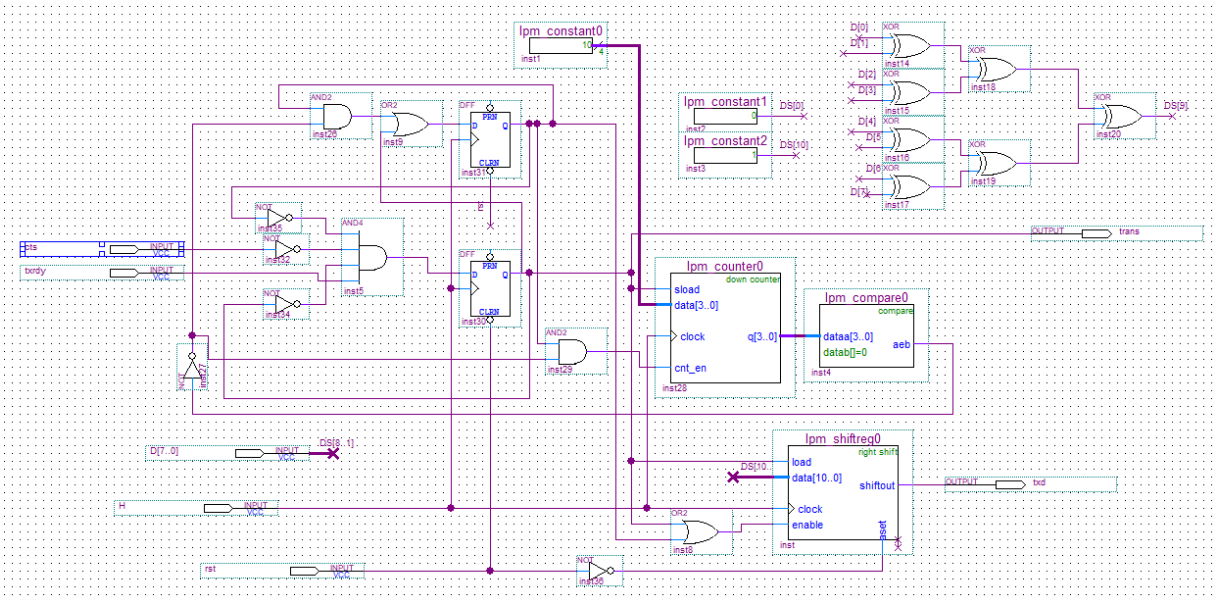

Figure 3: Solution RTL du transmetteur série

La figure ci-dessus montre une solution architecturale possible de ce circuit, solution exploitant les composants de la bibliothèque ALTERA. On peut observer que le registre à décalage ne dispose pas de commandes *load* et *shift* permettant de commander le décalage. Le chargement s'obtient par les deux signaux *sload* et *enable* à 1 tandis que le décalage s'obtient par un signal *sload* à 0 et l'entrée de contrôle *enable* à .

III Préparation au LAB

On souhaite ajouter un diviseur de fréquence pour pouvoir moduler le débit du transmetteur série en fonction d'une consigne provenant du microprocesseur ou d'un opérateur. L'objectif est de multiplier la période d'un bit par 2, 4, 8 ou 16 lorsque l'entrée C vaut 0,1,2 ou 3 respectivement. Ceci s'obtient en agissant sur la période de l'horloge en sortie HD (cf figure

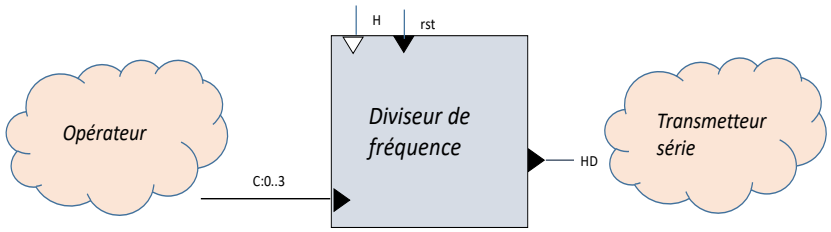

Figure 4: symbole du circuit diviseur de fréquence

La figure ci-dessous donne l'ASM décrivant le comportement attendu du circuit.

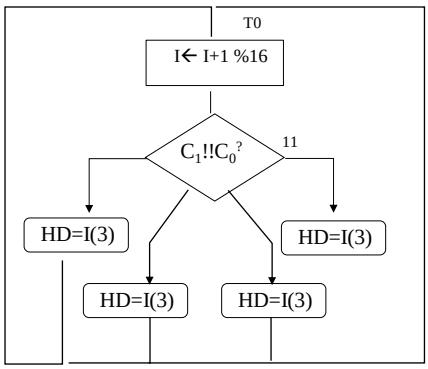

Figure 5: ASM du diviseur de fréquence

Comme on peut le noter, il s'agit d'une FSM à un seul état. Le circuit va se résumer à un chemin de données composé d'un registre compteur et d'une logique permettant de sélectionner la sortie du compteur servant d'horloge divisée. Il n'y a donc pas nécessité de définir une unité de contrôle séquentielle pour procéder à la génération des états suivants.

Questions : Donnez un bloc diagramme de ce circuit en procédant à l'allocation des composant *compteur* et *multiplexeur* nécessaires à l'implémentation des transferts de registre.

IV Travail en séance

1ère partie : Validation et test du circuit de transmission des données

***Questions* A.** Créez un projet *transmetteur* sous quartus 9.0 en sélectionnant le composant *cyclone II* EPC35F672C6 .

B. Procédez à l'implémentation du schéma RTL dans un fichier bloc diagram *transmetteur.bdf* en associant les composants de la bibliothèque ALTERA. Pour cela, on associera

- Une méga fonction *LPM_shiftreg* pour le registre à décalage *DI* sur 11 bits. Une entrée *aset* sera ajoutée pour forcer les bits du registre à 1 au départ.
- Une méga fonction *LPM_counter* pour la variable *I* sur 4 bits.
- Une méga fonction *LPM_compare* pour l'indicateur *z*.
- Une méga fonction *LPM_constant* pour la constante 10
- Des portes XOR pour le calcul de parité

Attention à ne pas copier-coller les composants de type *mega-fonction.* Ils doivent être créés et paramétrés à partir de la bibliothèque. L'unité de contrôle sera implémentée classiquement par des primitives.

C. Lancez l'analyse de timing (processing→classic timing analyser) et notez la période minimale qui doit être respectée pour un fonctionnement correct du circuit.

D. Créez un fichier de simulation (*waveform*) en vous inspirant du scénario de test de la figure ci-dessous, scénario proposant la transmission du caractère 'A', suivie du caractère 'C'.

E. Validez le bloc avec le scénario proposé ci-dessous :

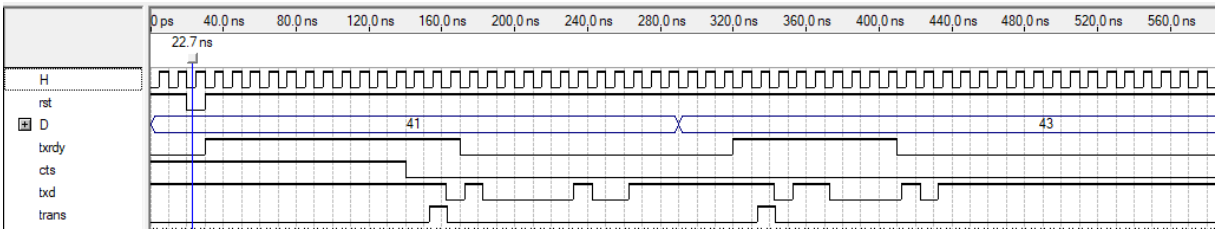

Figure 6: Exemple de résultat du test du circuit transmetteur

F. Procédez à la génération du symbole *transmetteur.bsf.*

G. Procédez à l'affectation des pattes du composant (menu *assigmments→pin_planner)* sur le modèle de la figue ci-dessus

	Node Name	Direction	Location	I/O Bank	VREF Group	Reserved	Group	PCB layer	I/O Standard
1	cts	Input	PIN_N25	5	B5_N1				3.3-V LVTTL (defaul
2	D[7]	Input	PIN_V2	1	B1_N0		D[7..0]		3.3-V LVTTL (defaul
3	D[6]	Input	PIN_V1	1	B1_N0		D[7..0]		3.3-V LVTTL (defaul
4	D[5]	Input	PIN_U4	1	B1_N0		D[7..0]		3.3-V LVTTL (defaul
5	D[4]	Input	PIN_U3	1	B1_N0		D[7..0]		3.3-V LVTTL (defaul
6	D[3]	Input	PIN_T7	1	B1_N0		D[7..0]		3.3-V LVTTL (defaul
7	D[2]	Input	PIN_P2	1	B1_N0		D[7..0]		3.3-V LVTTL (defaul
8	D[1]	Input	PIN_P1	1	B1_N0		D[7..0]		3.3-V LVTTL (defaul
9	D[0]	Input	PIN_N1	2	B2_N1		D[7..0]		3.3-V LVTTL (defaul
10	H	Input	PIN_D13	3	B3_N0				3.3-V LVTTL (defaul
11	rst	Input							3.3-V LVTTL (defaul
12	trans	Output							3.3-V LVTTL (defaul
13	txd	Output							3.3-V LVTTL (defaul
14	txrdy	Input	PIN_N26	5	B5_N1				3.3-V LVTTL (defaul

Figure 7 : Affectation des pattes du circuit

Compilez à nouveau votre projet. Cliquez sur le menu *tools*→*Signal Tap II analyzer*.

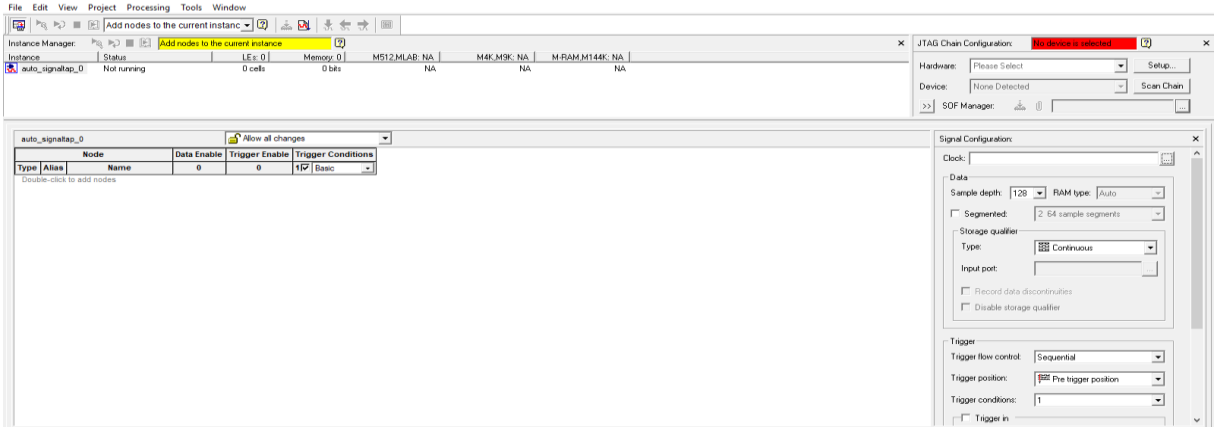

Figure 8 : Fenêtre de définition des signaux de test à l'analyseur logique

Dans la fenêtre qui apparait, cliquez sur … de *clock* (à droite) et sélectionner *H* après avoir cliquer sur *list*.

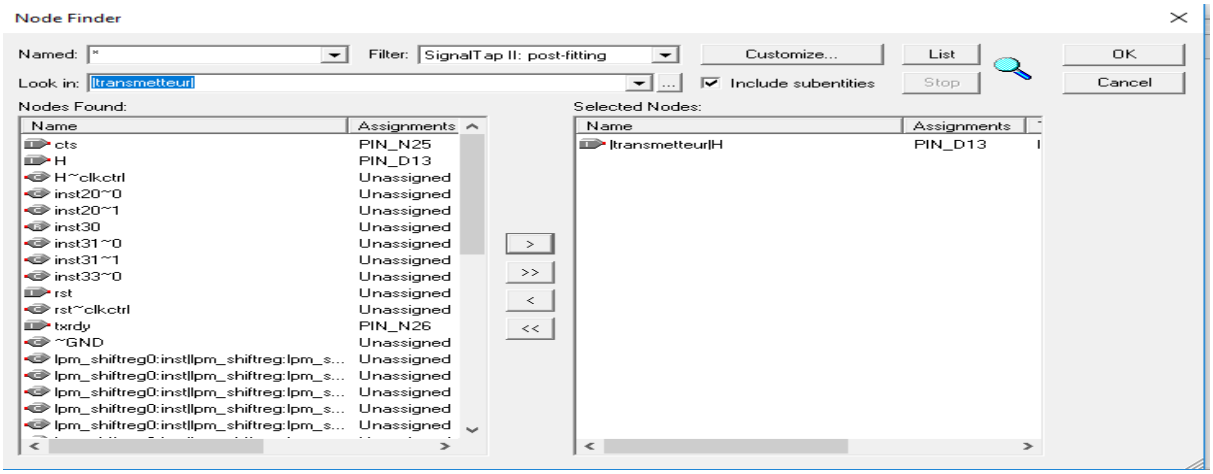

Figure 9: Spécification de l'horloge de référence pour l'analyseur

Positionnez *sample depth* à 32K pour spécifier la taille de la mémoire d'enregistrement. Compilez à nouveau votre projet. Programmez ensuite le circuit. Sélectionnez finalement votre matériel dans la fenêtre de l'analyseur logique comme indiqué ci-dessous.

Figure 10: Fenêtre après compilation, programmation et sélection du hard du hardware

Lancez l'analyseur logique en *run analysis* avec Cts=0 et Txrdy=1 puis avec CTS=0.

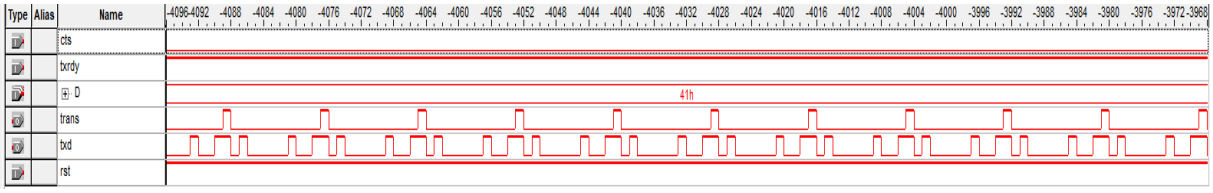

Figure 11: Trace à l'analyseur logique pour la transmission du caractère 'A'

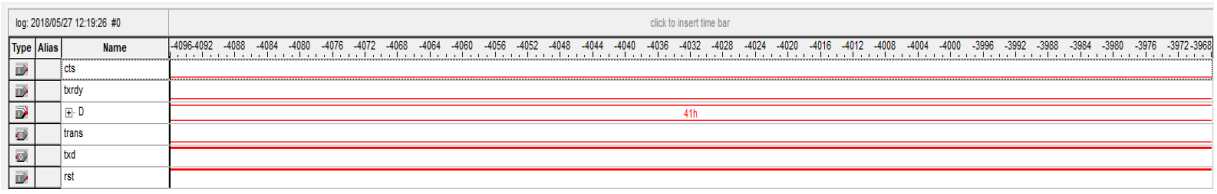

Figure 12: Trace montrant l'absence de transmission (CTS=1)

2ème partie : Validation du diviseur de fréquence

On souhaite finalement ajouter un diviseur de fréquence afin de pouvoir faire varier le débit de la transmission.

A. Dans un nouveau fichier bloc diagram, procédez à la saisie de votre solution du diviseur de fréquence. Compilez après avoir défini le fichier en top level.

B. Validez fonctionnellement votre circuit grâce à un chronogramme après avoir modifier l'entrée de la simulation.

C. Générez le symbole du diviseur *divfreq.bsf.*

D. Dans un 3ème bloc diagram, procédez à la connexion des 2 composants et validez fonctionnellement votre solution par une combinaison des 2 fichiers de simulations précédents

3ème partie : Validation du circuit transmetteur avec débit variable

A. Dans un ultime fichier bloc diagram *tserie.bdf* , procédez à la connexion de 2 instances des locs transmetteur et diviseur .Attention au reset Compilez après avoir défini le fichier en top level.

.

B. Validez votre circuit à l'aide d'un chronogramme montrant la modification du signal tx lorsqu'on modifie la consigne de débat *C.*

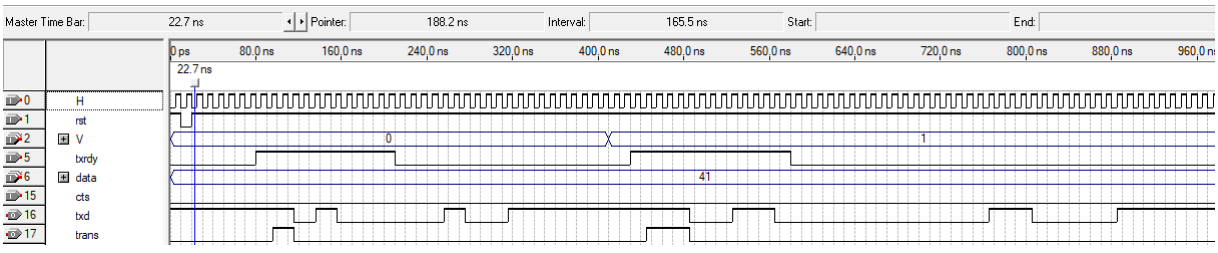

Figure 13: Exemple de réponse du circuit avec modification de consigne