ELECINF102
Processeurs et Architectures Numériques

Contrôle de connaissances

15 juin 2016 à 8h30

Document autorisé : une feuille recto-verso

Durée: 1h30 minutes

Ce contrôle comporte 4 parties **indépendantes** :

1. Fonction "LUT"
2. Représentation en CA2 des nombres décimaux.
3. Filtre transposé
4. Decodeur Huffman

Consignes importantes : Si des **schémas** sont demandés dans les différents exercices, ils doivent être impérativement clairs, lisibles et sans ambiguïté. Les dimensions des bus doivent être indiquées. Si nécessaire le sens des signaux doit être précisé. Pour la logique synchrone, les signaux d'horloge et d'initialisation asynchrone (`reset_n`) ne seront pas représentés dans ces schémas.

N'oubliez pas d'inscrire nom, prénom, et numéro de casier sur votre copie.

Bon courage !

1 Fonction "LUT"

La figure 1 représente une "Look Up Table" (LUT) à 2 entrées A et B, elle est composée d'un arbre de 3 multiplexeurs. Une LUT à 2 entrées permet de réaliser une fonction programmable à 2 entrées. La programmation consiste à fixer une valeur constante sur les entrées w,x,y,z.

1. Si on fixe les entrées w,x,y,z à 4'b0001, quelle est la fonction logique S=f(A,B) implantée ?
2. Si on fixe les entrées w,x,y,z à 4'b1001, quelle est la fonction logique S=f(A,B) implantée ?
3. Combien de fonctions logiques peut on générer avec cette LUT à 2 entrées ?
4. Combien faut-il de LUT à 2 entrées pour réaliser une LUT à 4 entrées ? Faites un schéma.

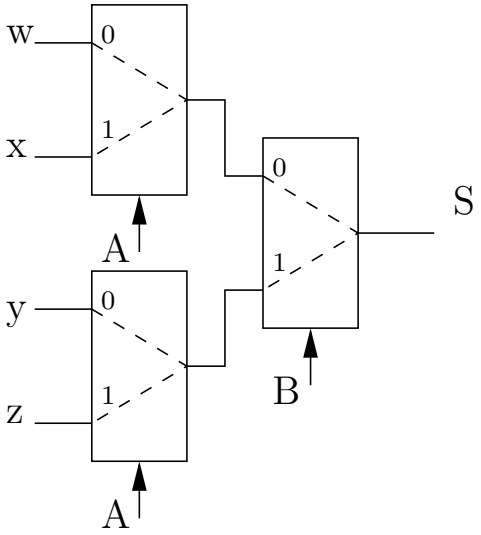

Figure 1 – *Fonction LUT*

2 Représentation en CA2 des nombres décimaux

Vous avez appris en cours comment coder en CA2 des entiers relatifs, et comment coder en binaire une approximation des nombre décimaux non signés. Voyons maintenant comment coder en CA2 une approximation des nombres décimaux signés.

On rappelle qu'un nombre décimal positif D peut être approximé en base 2 par un vecteur

$$(a_{n-1}, a_{n-2}, \ldots, a_1, a_0, a_{-m} \ldots a_{-m})$$

tel que :

$$D = a_{n-1} \cdot 2^{n-1} + \ldots + a_1 \cdot 2^1 + a_0 \cdot 2^0 + a_{-1} \cdot 2^{-1} + \ldots a_{-m} \cdot 2^{-m}$$

Où :

- $(a_{n-1}, \ldots a_0)$ est la partie entière de D (sur n bits).
- $(a_{-1}, \ldots a_{-m})$ est la partie fractionnaire de D (sur m bits).
- 2^{-m} représente la précision de cette approximation.
- Cette représentation est nommée "n.m".

Dans la suite de l'exercice, on chercher à représenter $-5,32$ sur 4.3 bits.

2.1 Première méthode.

Question 1 : en remarquant que $-5,32 = -6 + 0,68$, donnez une représentation de $-5,32$ sur 4.3 bits. Détaillez votre calcul.

2.2 Deuxième méthode.

Question 2 : Montrez que sur n bits, $X + \overline{X} + 1 = 0$.
Question 3 : Que devient cette équation si X est un nombre décimal relatif codé sur 4.3 bits ?
Question 4 : Utilisez le résultat précédent pour une donner une représentation de $-5,32$ sur 4.3 bits.

2.3 Analyse.

Question 5 : Pourquoi ne trouve-t-on pas forcément les mêmes résultats avec les deux méthodes ? Expliquez ce que donne l'une et ce que donne l'autre .

3 Filtre transposé

Soit le code SystemVerilog suivant :

```
module transFIR (input  logic        clk,
                 input  logic [ 7:0] coef0, coefT, coefZ, // Coefficients
                 input  logic [15:0] sigIn,               // Signal d'entrée
                 output logic [15:0] sigOut);             // Signal de sortie

   // Signaux internes
   logic [PARAM:0] prod0, prodT, prodZ;
   logic [   25:0] int0, intT;

   always @(*) sigOut <= intT[25:10];

   always @(*) begin
      intT  <= prodT + int0;
      prodT <= coefT * sigIn;
      prod0 <= coef0 * sigIn;
   end

   always @(posedge clk) begin
      int0  <= prod0 + prodZ;
      prodZ <= coefZ * sigIn;
   end

endmodule
```

L'entrée `sigIn` correspond au signal sur lequel on applique le filtre. Les échantillons de ce signal arrivent à la cadence de l'horloge `clk`.

Les entrées `coef0`, `coefT` et `coefZ` correspondent aux coefficients du filtre. Ces coefficients restent constants durant l'utilisation du filtre.

`PARAM` est une valeur entière constante.

Question 3.1 Dessinez le schéma structurel équivalent à ce code en utilisant des registres et des opérateurs d'addition et de multiplication.

Question 3.2 Déterminez, en fonction du cycle d'horloge courant n et des cycles précédents, les équations des signaux internes `int0` et `prodZ` ($\texttt{int0}_n$ = ??, $\texttt{prodZ}_n$ = ??). En déduire l'équation du filtre complet, c'est-à-dire du signal de sortie `sigOut` en fonction du signal d'entrée `sigIn`.

Question 3.3 En justifiant votre réponse, proposez une valeur pour la constante `PARAM`.

Question 3.4 Justifiez le nombre de bits utilisés pour les signaux internes `int0` et `intT`. Est-il possible de modifier la taille de ces signaux sans perdre en dynamique ?

4 Décodeur Huffman

Soit le module `decode_huff` dont le squelette en SystemVerilog est le suivant :

```
module decode_huff(input logic clk,
                   input logic reset_n,
                   input logic data_in,
                   output logic [7:0] data_out);

   // ...
endmodule
```

Ce module reçoit en entrée sur `data_in` une série de bits codant des symboles pouvant prendre 8 valeurs (de 0 à 7) selon le codage suivant :

Code	Symbole
`0 0 0 0 0`	0
`0 0 0 0 1`	1
`0 0 0 1`	2
`0 1 0`	3
`0 1 1`	4
`0 0 1`	5
`1 0`	6
`1 1`	7

- Le signal `reset_n` est actif à l'état bas.
- Les bits du code reçu arrivent de façon synchrone avec l'horloge `clk`.
- Le premier bit du premier code est reçu au premier cycle suivant la fin du reset.
- Les bits se succèdent cycle après cycle en commençant par le plus à gauche dans la table ci-dessus.
- Lorsqu' un symbole est décodé, le bit de `data_out` d'indice correspondant au symbole doit passer à `1` pendant un cycle d'horloge.
- Les symboles sont reçus les uns après les autres sans interruption.

Question 1

Donnez le code (ou le schéma) du module `decode_huff` permettant d'effectuer le décodage.

Exemple : si le module reçoit sur `data_in` `0` puis au front d'horloge suivant `1` puis `1`, le bit numéro 4 de `data_out` doit passer à `1` pendant une période d'horloge avant de repasser à `0`.

Question 2

L'émetteur, auquel est connecté notre module, n'a pas tout le temps des données à nous transmettre. Un signal d'entrée `data_in_valid` est donc ajouté à notre module. La valeur `data_in` n'est à prendre en compte que lorsque `data_in_valid` vaut `1`. L'interruption de la transmission peut intervenir au milieu de la transmission d'un symbole (auquel cas elle ne doit être considérée que comme une simple pause et non pas l'arrêt de la transmission de ce symbole).

Donnez le code (ou le schéma) de la nouvelle version de votre module.