SYL_Labo2

Questions

Question 1

Identifiez et documentez les différents opcodes entraînant une erreur.

Les opcodes qui génèrent des erreurs sont ceux qui ne sont pas pris en charge par les blocs. C'est-à-dire:

- Tous ceux du format `010xxx`
- `011000`
- `011111`

Ceci peut être contrôlé en calculant le nombre total d'opcode possible et enlever chacun de ceux qui sont définis. Un opcode est défini sur 6 bits et à donc 64 valleurs possibles allant de `000000` à `111111`.

Pour le bloc `add/sub`, il y a 4 opcodes qui ont chacun deux valeurs indéterminées. Ceci veut dire qu'on a `4 * 2^2` donc 16 opcodes possible.

Pour le bloc `comparateur`, il y a 6 opcodes définis.

Pour le bloc `logique`, il y a, comme pour le bloc add/sub, 4 opcodes qui ont chacun deux valeurs indéterminées. Ceci veut dire qu'on a `4 * 2^2` donc 16 opcodes possible.

Pour le bloc `custom`, il y a 1 opcode avec 4 valeurs indéterminées. Ceci représente `1 * 2^4` donc 16 opcodes possible.

Si on reprend notre total de 64 opcodes possibles et qu'on enlève tout ceux qui sont définis (16 pour add/sub, 7 pour comparateur, 16 pour logique et 16 pour custom) on trouve qu'il devrait y avoir 10 opcodes non définis. Ceci est bien le cas si on considère les opcodes cités plus haut. Il y a un opcode avec 3 bits indéterminés et 2 opcodes définis. Ceci fait un total de 10 opcodes entrainant une erreur ce qui correspond à la valeur attendue.

Question 2

Reprenez le schéma disponible en Figure `1` et donnez, pour chaque bloc, les bits d'opcode utilisés (l'idée est de répondre aux oints d'interrogation dans le schéma). Vos réponses doivent être de la forme "les bits d'opcode du bloc add/sub sont les bits [X:Y]".

add/sub:

Les bits d'opcode du bloc add/sub sont les bits [3:2]

comparateur:

Les bits d'opcode du bloc comparateur sont les bits [2:0]

unité logique:

Les bits d'opcode du bloc unité logique sont les bits [1:0]

custom:

Le bloc custom n'a pas besoin de faire de choix d'opération

Question 3

Lorsqu'un overflow se produit, le signe n'est plus géré correctement et la sortie `negative_o` n'a donc pas la bonne valeur. Corrigez ce problème et expliquez votre raisonnement dans le fichier pdf.

Lorsqu'on a un overflow, le signe n'est plus géré correctement car le bit de signe est décalé d'un bit vers la gauche. Ceci correspond à la carry.

Avec un multiplexeur, on peut donc sélectionner si on utilise le MSB ou la carry pour le bit de signe. Ceci permet de gérer le signe correctement.

Question 4

Mettez en place une table de vérité, puis justifiez le développement de votre circuit.

Les nombres binaires qui sont des puissances de 2 ne possèdent qu'un seul 1 et tous les autres bits sont à 0. Ceci est dû au fait qu'un bit à 1 représente une puissance de 2.

Notre fonction logique devra donc contrôler qu'au maximum 1 bit est à 1 dans le nombre en entrée.

La table de vérité que nous voulons obtenir avec notre circuit est donc:

A	B	C	D	S
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Nous pouvons donc déduire la fonction `A/B/C/D + /AB/C/D + /A/BC/D + /A/B/CD`. Celle ci peut facilement être simplifiée en `(A/B + /AB)/C/D + (C/D + /CD)/A/B` qui en tour devient `(A^B)/C/D + (C^D)/A/B`.

Cette fonction devient facile à schématiser.

Question 5

En plus des opcodes invalides, certains cas d'utilisation doivent générer une erreur. Déterminez ces différents cas et documentez les avant d'implémenter les modifications nécessaires.

Dans le cas où nous faisons des addition ou soustraction et qu'il y a un overflow, celui-ci n'était pas pris en compte. nous avons donc ajouter des cas d'erreurs dans le schéma `ALU_Top` pour signaler une erreur lors d'un overflow lorsqu'on emploi le bloc `Add/Sub`.

Question 6

Avec `a_i[3:0] = 0001`, `b_i[3:0] = 1001` et `op_i[6:0] = 011011` quelle est la valeur observée pour l'overflow ? Et pour la sortie `erreur_o` ? Cela correspond-t-il à ce que vous attendiez à la suite de la Question 5?

La valeur de `Overflow` est à 1 et la valeur de `erreur` est à 0. Ceci est exactement ce que nous attendions car l'overflow représente une erreur seulement si l'opcode désigne une utilisation de l'alu pour faire une addition ou une soustraction.