HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Prof. Yann Thoma

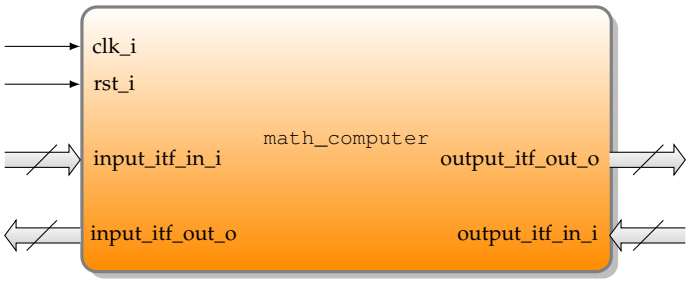
Laboratoire CSF
FIFO de taille générique
semestre printemps 2016 - 2017

Objectifs pédagogiques

Ce laboratoire a pour but de réaliser un calculateur séquentiel, voire pipeliné.

Cahier des charges

Il vous est demandé de réaliser un calculateur prenant 3 nombres en entrée et fournissant un résultat.
Un paramètre générique, `DATASIZE`, permet de spécifier la taille des opérandes et du résultat.



Entrées/sorties

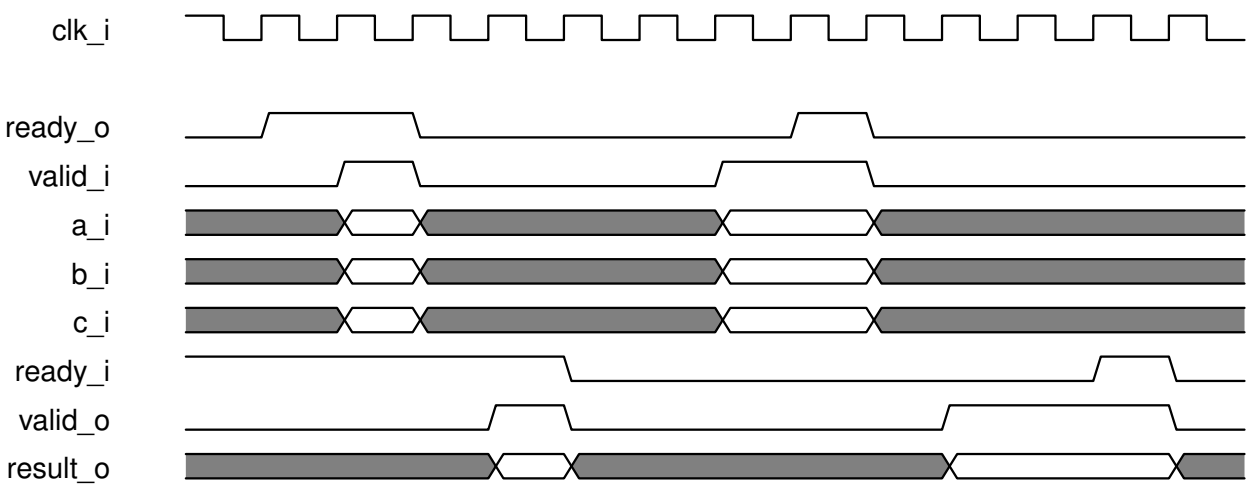
Les entrées/sorties sont celles de l'entité suivante :

```
entity math_computer is
generic (
  DATASIZE   : integer := 8
);
port (
  clk_i             : in  std_logic;
  rst_i             : in  std_logic;
  input_itf_in_i    : in  math_input_itf_in_t;
  input_itf_out_o   : out math_input_itf_out_t;
  output_itf_in_i   : in  math_output_itf_in_t;
  output_itf_out_o : out math_output_itf_out_t
);
end math_computer;
```

Les interfaces d'entrées/sortie sont déclarées dans le paquetage `math_computer_pkg.vhd`. Vous noterez l'utilisation de `record` pour les ports d'entrée/sortie. Ceci permet de simplifier un design en ayant moins de connexions à faire.
Le calculateur peut indiquer qu'il est prêt à accepter des données en activant la sortie `input_itf_out_o.ready`. Le calcul est lancé lorsque l'entrée `input_itf_in_i.valid` est placée à '1' et à ce moment-là les valeurs numériques sont pertinentes. Lorsque le calcul est terminé le composant doit activer `output_itf_out_o.valid` et placer le résultat sur le bus de sortie. Le signal d'entrée `output_itf_in_i.ready` permet au bloc instanciant le calculateur de le faire patienter en laissant `ready` à '0'.
Autant sur l'entrée que la sortie, les données sont échangées lorsque le `ready` et le `valid` sont actifs.

Vous noterez l'usage de vecteurs non contraints dans le design. Ceci nécessite de les contraindre à l'utilisation, ce qui est fait en l'occurence dans le testbench lors de l'instanciation du composant.

Le chronogramme suivant illustre le protocole d'entrée/sortie qui doit être respecté.



Le calculateur est décomposé en une partie contrôle et une partie traitement, qui vont devoir être modifiées ou non en fonction de l'exercice. Là encore des `record` permettent de communiquer entre les deux composants.

Travail à réaliser

Avant toute chose, dessinez le chronogramme du datapath proposé. Ceci vous facilitera la vie ensuite. (`math_computer_datapath.vhd`)
Gardez une version du projet pour chaque exercice. (Copie de tout le projet).

Exercice 1

Décrire un contrôleur capable d'effectuer l'opération suivante :

$$result = 2a + b - c$$

Pour ce faire, il est interdit de modifier le datapath. Votre intervention ne se fait donc que dans la partie contrôle.
Modifiez aussi le banc de test afin qu'il vérifie cette opération. Pour ce faire, modifiez la procédue `generate_reference()` dans le fichier `math_computer_tb.vhd`.

Exercice 2

Décrire un composant capable d'effectuer l'opération suivante :

```
if (a > b)
  return 2 a - b;
else
  return 2 b - a;
```

Pour ce faire vous avez le droit de modifier l'unité de traitement mais uniquement pour permettre à l'unité de contrôle de savoir si `a` est plus grand que `b` ou non.
Modifiez également le banc de test pour qu'il vérifie ce calcul.

Exercice 3

La version séquentielle de notre calculateur est sympathique, mais pas très optimisée en termes de débit. Pour une nouvelle application, modifiez `math_computer` pour pipeliner le calcul pour la fonction suivante :

$$result = 2a + b - c$$

Pour cette version pipelinée, vous vous êtes libres de modifier entièrement l'architecture de `math_computer`. Votre design doit pouvoir permettre de lancer un calcul à chaque cycle d'horloge. Attention toutefois à bloquer le fonctionnement si le bloc connecté à sa sortie n'est pas prêt à recevoir les données.
Vous avez comme condition de ne pas effectuer plus d'une opération arithmétique chainées par étage du pipeline. Il n'est, par exemple, pas permis d'effectuer $a + b + c$ dans un seul étage car il faudrait chainer deux additionneurs. Il est en revanche permis d'effectuer deux opérations indépendantes : $a + b$ et $a + c$ par exemple.

Travail à rendre

Une archive avec un répertoire par exercice (projet complet).