Cahier des charges

Déroulement

- Le projet est réalisé par groupe de 2 au maximum.
- Le travail en dehors des heures de cours est **indispensable** à l'aboutissement du projet.
- Chaque personne doit tenir à jour un « cahier de laboratoire » où il note proprement ce qu'il a fait personnellement et quand il l'a fait (date et durée), les problèmes rencontrés, la raison de certains choix, etc. Le but est d'avoir une traçabilité individuelle du déroulement du projet. Ce cahier est à **envoyer par email après chaque leçon** au professeur.
- Chaque groupe doit rendre **par email** dans un document Nom1_Nom2.zip:
    - Un document complet (PDF) présentant la conception, à la fin du cours de la **semaine 12**
        - Schéma bloc du processeur (nanoProcesseur)
        - Schéma bloc du contrôleur (nanControleur)
        - Graph des états du séquenceur
        - Table de vérité des sorties du séquenceur
        - Plan mémoire
        - Cahier de laboratoire complet

      Remarques : Les documents faits proprement à la main (avec une règle) et scannés suffisent.
    - Un projet vivado avec testbench de votre nanoControleur qui effectue une multiplication en « assembleur » en utilisant le jeu d'instruction fourni au début du cours de la **semaine 16**. Des questions seront posées afin de vérifier la compréhension du fonctionnement du nanoControleur et de votre application « assembleur » en ROM.
        - Cahier de laboratoire complet est à rendre également
    - Un document complet (PDF) présentant les modifications de la conception pour ajouter la multiplication hardware, à la fin du cours de la **semaine 19**
        - Schéma bloc du processeur (nanoProcesseur)
        - Schéma bloc du contrôleur (nanControleur)
        - Graph des états du séquenceur (en principe identique au précédent)
        - Table de vérité des sorties du séquenceur
        - Plan mémoire
        - Jeu d'instructions
        - Cahier de laboratoire complet
    - Un projet complet de votre microcontrôleur à la fin du cours de la **semaine 24**
        - Projet Vivado avec les fichiers sources et le banc de test (testbench)
        - Cahier de laboratoire complet
    - Faire une présentation (oral sans support Powerpoint) ainsi qu'une démonstration sur le kit Xilinx durant le cours de la **semaine 24**.
        - Des questions seront posées par les professeurs afin d'évaluer votre degré de compréhension du projet.
- Il est autorisé de collaborer entre les groupes, par contre, la copie partielle ou totale et/ou la transmission de documents sont interdits et seront sanctionnés.
- Tous les documents non restitués dans les délais seront taxés de la note 1.

But

Intégrer dans le circuit logique programmable du Kit Xilinx 7 le nanoControleur permettant d'exécuter une multiplication logicielle écrite en assembleur « nanoProcesseur » et également modifier l'architecture du nanoControleur pour permettre de faire une multiplication matérielle via l'ALU. **Toutes les bascules doivent être synchronisées** (clockées) par le signal d'horloge de 100MHz et toutes doivent avoir un **reset asynchrone**.

Description du nanoControleur

Le nanoControleur est composé de :

- Un nanoProcesseur
- Une RAM de 32 bytes
- Une ROM contenant le programme exécuté par le nanoProcesseur
- Un décodeur d'adresses
- Un multiplexeur permettant de choisir les données lues par le nanoProcesseur
- Deux registres de sortie de 8 bits, deux ports d'entrée de 8 bits

Le nanoProcesseur est composé de :

- Un Program Counter de 8 bits
- Un registre IR (Instruction Register) qui mémorise l'instruction à exécuter
- Une ALU avec calcul des indicateurs (flag : Z, C, N et V)
- Un accumulateur 8 bits pour recevoir le résultat de l'ALU
- Un multiplexeur permettant de choisir les opérandes de l'ALU
- Un registre permettant de mémoriser les 2 opérandes de l'ALU
- Un registre de contrôle de 4 bits contenant Z C N V
- Un séquenceur sous la forme d'une machine d'états synchrones qui « cadence » le nanoProcesseur

Le nanoProcesseur est réalisé selon l'architecture de type Harvard qui sépare physiquement la mémoire de données et la mémoire programme. L'accès à chacune des deux mémoires s'effectue via deux bus distincts.

Rappel des indicateurs:

Flag	Nom	Description
Z	Zero flag	Indique que le résultat d'une opération arithmétique ou logique ou d'un chargement était zéro.
C	Carry flag	Retenue qui permet d'ajouter/soustraire des nombres supérieurs à un octet. Il est également utilisé pour étendre les décalages binaires.
N	Negative flag	Indique que le résultat d'une opération mathématique est négatif.
V	Overflow flag	Indique que le résultat signé d'une opération est trop grand (dépassement) pour tenir dans la largeur du registre en utilisant la représentation en complément à deux.

Blocs du nanoControleur

Le nanoControleur est décomposé hiérarchiquement afin d'obtenir des blocs simples, étudiés durant le cours de systèmes numériques. Vous trouverez ci-dessous une description des blocs qui figurent dans le nanoControleur :

ROM

Le bloc ROM contient le programme qui sera exécuté par le nanoControleur. Il est réalisé sous forme d'un multiplexeur (combinatoire avec assignation sélective concurente) qui fournit l'instruction (14 bits) à exécuter en fonction de la valeur du registre Program Counter PC. Pour les instructions sans opérande il faut ajouter 8 bits à '0', '1' ou '-' pour compléter le bus

RAM

Le bloc RAM permet de mémoriser les 32 bytes de données. Il est réalisé avec un bloc RAM (voir doc Xilinx). Un bus d'adresses de 5 bits permet d'accéder aux données 8 bits.

Le décodeur d'adresse

Ce bloc combinatoire génère les 3 signaux de sélection (chip select) pour les accès dans la RAM ou dans les 2 ports d'entrées ou sorties. Les signaux chip select sont générés en fonction du bus d'adresse. Le plan mémoire (memory map) définit les zones actives des chip select.

Le multiplexeur

Ce bloc combinatoire permet de sélectionner grâce aux chips select du décodeur d'adresse quelles données sont lues par le nanoProcesseur, à savoir soit la RAM, ou l'un des 2 ports d'entrée de 8 bits synchronisé.

Registres de sorties

Les 2 ports de sorties sont deux registres tampon (parallèles) de 8 bits qui mémorisent la valeur envoyée par le nanoProcesseur lors d'une instruction STORE.

Ports d'entrées

Il y a deux ports d'entrées de 8 bits.

nanoProcesseur

Le bloc nanoProcesseur est la CPU du nanoControleur. Il est également décomposé hiérarchiquement et il est décrit ci-dessous.

Blocs du nanoProcesseur

Le nanoProcesseur est décomposé hiérarchiquement afin d'obtenir des blocs simples, étudiés durant le cours de systèmes numériques. Vous trouverez ci-dessous une description des blocs qui figurent dans le nanoProcesseur:

Le séquenceur

Le séquenceur est le chef d'orchestre du nanoProcesseur. Il s'agit d'une machine à états synchrone de Mealy permettant de générer les différents signaux de contrôles des autres blocs en fonction de l'état actuel et des entrées du séquenceur.

Les entrées sont :
- L'opcode provenant du registre IR
- Les indicateurs (flags : Z, C, N et V) provenant du registre CCR

Les sorties sont (par ordre alphabétique) :
- Accu_load
  permet le chargement de la sortie de l'ALU dans le registre Accu

- CCR_load
  permet le chargement des indicateurs Z C N V dans le registre CCR

- data_wr
  permet l'écriture d'une donnée dans la RAM ou sur un port de sortie

- IR_load
  permet le chargement de l'instruction provenant de la ROM dans le registre d'instruction

- oper_load
  permet le chargement des opérateurs dans le registre du bloc operande_select

- oper_sel
  code de 3 bits permettant la sélection des opérandes utilisé par l'ALU en fonction de l'opcode

- PC_inc
  PC_inc vaut '1' quand le ProgramCounter (PC) doit être augmenté d'une unité.
  PC_inc vaut '0' quand le PC doit prendre une nouvelle valeur à cause d'un « branchement »

- PC_load
  permet le chargement du nouveau PC dans le registre PC

Le ProgramCounter

Le bloc ProgramCounter est un compteur qui contient l'adresse de la prochaine instruction à exécuter. La nouvelle adresse est soit l'adresse actuelle incrémentée de 1 ou soit une adresse provenant d'une instruction de branchement.

Le registre IR

Le registre d'instructions (IR) mémorise l'instruction à exécuter fourni par la ROM. Les instructions sont composées d'un opcode de 6 bits et d'un opérande de 8 bits.

Le registre Accu

Le registre Accu contient la valeur de l'accumulateur. Cette valeur provient de l'ALU. Elle est chargée à la fin du traitement l'ALU.

Le registre CCR

Le registre de contrôle CCR mémorise les indicateurs (flags : Z, C, N et V) qui sont calculés par l'ALU.

Le sélecteur d'opérandes « operande_select »

Il s'agit d'un multiplexeur qui permet de choisir les opérandes utilisés par l'ALU. La sélection se fait à l'aide d'un code de 3 bits fourni par le séquenceur.

Le registre Opérandes

Registre permettant de mémoriser les 2 opérandes de l'ALU

L'ALU

L'ALU effectue en parallèle (combinatoire) toutes les opérations arithmétiques et logiques sur les opérandes ainsi que les opérations de chargement. Un multiplexeur commandé par l'opcode permet de choisir le résultat de l'opération demandée.
L'ALU calcule également les indicateurs (flags : Z, C, N et V) (combinatoire). Un système de masque dépendant de l'opcode permet de ne pas modifier certains indicateurs selon l'opération effectuée.

L'Alu est donc séparé en deux parties distinctes :

1. Le bloc alu combinatoire qui calcule le résultat en fonction de l'opcode des opérandes et des flags, ce bloc gère également un signal local CCR_mask de 4 bit qui permet de savoir quel seront les flags mis à jours lors de l'instruction en cours. Créer un signal ALU_result sur 9 bits pour les opérations nécessitant la gestion du carry, ceci simplifiera grandement la gestion du flag C.

2. Un bloc combinatoire qui va calculer individuellement les flags Z, C, N et V en fonction de ALU_result, de CCR_mask, du registre CCR et selon le type de flags des opérandes et/ou de l'opcode.

Travail à réaliser :

1. Analyser le projet Vivado fourni et créer les documents suivants en fonction des fichiers *.vhdl :
   - Schéma bloc du processeur (nanoProcesseur)
   - Schéma bloc du contrôleur (nanControleur)
   - Graph des états du séquenceur
   - Table de vérité des sorties du séquenceur
   - Plan mémoire

2. Modifier le programme qui se trouve dans la ROM en ajoutant une multiplication (software) qui utilise uniquement le jeu d'instructions fourni. Un banc de test (testbench) doit également être créé afin de valider et vérifier le bon fonctionnement de votre programme.

3. Modifier l'architecture du nanoControleur pour permettre la multiplication de deux nombres de 8 bits non signée dans l'ALU. L'ALU étant prévu pour l'instant pour des résultats sur 8 bits, il faudra y apporter les modifications suivantes (liste non–exhaustive) :

   - Ajouter un registre accumulateur car le résultat de la multiplication est sur 16 bit.
   - Ajouter les instructions permettant d'effectuer la multiplication, ainsi que de copier le nouveau registre accumulateur dans l'ancien.
   - Modifier l'ALU pour ajouter la multiplication et gérer les flags
   - Modifier le sélecteur d'opérandes et le séquenceur.

   Autres modifications demandées qui ne sont pas liées à la multiplication :

   - Ajouter un port d'entrée de 4 bits permettant de lire l'état de 4 poussoirs dans le nanoControleur.
   - Ajouter un port de sorties de 8 bits permettant de piloter les 4 leds rouges et les 4 leds vertes du kit Xilinx 7.
   - Modifier les blocs décodage d'adresse (Address_Decode) et multiplexeur de données (Data_Multiplexer) en conséquence.

   Modifier les schémas blocs réalisé au point 1 pour correspondre à votre nouvelle architecture.

   Modifier le plan mémoire du nanoControleur réalisé au point 1 et y ajouter les ports d'entrées et sorties décrit ci-dessus.

   Modifier la table de vérité des sorties du séquenceur réalisée au point 1.

   Modifier le jeu d'instructions pour y ajouter les nouvelles opérations nécessaires pour la multiplication.

4. Ajouter et/ou modifier tous les fichiers VHDL structurels du nanoControleur en fonction des modifications réalisées sur vos schémas bloc.

   Ajouter et/ou modifier tous les fichiers VHDL comportementaux du nanoControleur en fonction des modifications réalisées sur vos schémas bloc.

Ecrire une application en assembleur tournant en boucle permettant de réaliser de manière software et hardware la multiplication de 2 nombres de 8 bits non signés en provenance des 2 ports d'entrée connectés sur 2 x 8 dilswitchs) et d'afficher le résultat sur 16 bits via les 2 port de sorties connectés sur 16 leds des deux bargraphes.

La position du Poussoir 3 du kit Xilinx 7 détermine si la multiplication est matérielle ou logicielle :

- Poussoir 3 pressé : multiplication matérielle à l'aide du bloc multiplication développé précédemment et implémenté dans le nanoControleur. Le micocontrolleur allume les leds vertes lors de la multiplication matérielle.

- Poussoir 3 non pressé : multiplication logicielle (assembleur) à écrire. Il faut savoir qu'une multiplication est une suite d'additions (voir cour p.25). Le micocontrolleur allume les leds rouges lors de la multiplication logicielle.

Créer dans le projet Vivado un banc de test (testbench) VHDL permettant de valider totalement les 2 cas de figures ci-dessus.

Analyser la simulation et comparer les temps d'exécution des 2 types de multiplications. Il faut être capable d'expliquer le déroulement d'instructions dans le nanoControleur.

Synthétiser, assigner les pattes, implémenter et programmer l'application sur le Kit Xilinx 7 et tester le système numérique réalisé sur le Kit Xilinx 7.