Unité de calcul modulaire
RALU "Register Arithmetic Logic Unit"

Mandat

Le but de ce projet est de concevoir, réaliser et tester une description VHDL synthétisable d'une unité modulaire pour des opérations arithmétique. Un tel circuit est communément nommé RALU pour "Register Arithmetic Logic Unit". Les nombres pourront être en représentation non-signée ou signée en C2. Ce circuit sera de type séquentiel, il devra répondre aux spécifications données ci-après.

Entrées

- Reset_i signal d'initialisation à action asynchrone, actif haut
- Horloge_i signal d'horloge du système d'une fréquence de 1 MHz avec un rapport cyclique de 50%.
- Code_i entrée indiquant quelle opération le circuit doit exécuter, ce signal est sur 3 bits. Il y a 8 opérations possibles.
- Enable_i signal synchrone avec Horloge, permettant l'exécution de l'opération.
- Operand_i opérande utilisée dans certaines opérations (N bits, non signé ou signé C2)
- Pour RALU modulaire de 4 bits: prévoir les entrées de cascade nécessaires (à définir).

Sorties

- Result_o résultat de l'opération effectuée (N bits, non signé ou signé C2)
- Err_Sgn_o Sortie indiquant le dépassement pour des nombres signés
- Err_NSgn_o Sortie indiquant le dépassement pour des nombres non-signés
- Pour RALU modulaire de 4 bits: prévoir les sorties de cascade nécessaires (à définir).

Spécification du fonctionnement souhaité

Au flanc actif du signal d'horloge, la sortie Result_o prend la valeur du résultat de l'opération spécifiée par l'entrée Code_i, si l'entrée Enable_i est active; sinon, la sortie Result_o garde sa valeur.

L'unité RALU est capable d'effectuer les opérations suivantes :

- Charger : affecter la valeur de l'entrée Operand à la sortie Result.
- Négation : affecter la valeur négative de l'entrée Operand à la sortie Result.
- Compter : ajouter 1 à Result.
- Décompter : soustraire 1 à Result
- Additionner : ajouter Operand à Result et affecter le résultat sur la sortie Result.
- Soustraire Op: soustraire Operand à Result et affecter le résultat sur la sortie Result.
- Décalage à gauche (SLL): décaler Result d'un bit à gauche avec insertion d'un zéro dans le poids faible (LSB).
- Rotation à gauche carry (RCL): décaler Result d'un bit à gauche avec insertion du Carry dans le poids faible (LSB) et sauvegarde du poids fort (MSB) dans le Carry.

Le circuit doit permettre d'effectuer des calculs pour des nombres entiers non signés et des nombres signés en "complément à 2".

La réalisation sera réalisée de façon modulaire. Celle-ci doit aussi permettre de pouvoir modifier la taille du composant de base RALU. Vous disposez d'un paquetage qui définit une constante Nbits_c indiquant la taille de la cellule de base modulaire. L'unité top réalisée sera prévue pour chainer 3 composants RALU.

Pour l'intégration dans un CPLD, vous choisirez un composant RALU (unité modulaire de base) avec une taille de 4 bits pour les nombres (Nbits_c = 4). Dès lors l'unité de calcul RALU_top permettra de réaliser des calculs pour des nombres de 12 bits. Cette unité comprendra 3 composants de calcul de 4 bits.

Marche à suivre

Voici les étapes :

1. Analyser le fonctionnement de l'unité RALU demandée. Comprendre la décomposition modulaire du système et des implications sur le chainage des modules 4 bits. Identifier et définir les spécifications d'un module 4 bits.
2. Concevoir une décomposition fonctionnelle de l'unité RALU en visant une quantité de matériel minimum;

   Etapes à réaliser en commun, puis chaque étudiant réalisera une version différente:

   1° Rechercher une décomposition modulaire et établir un schéma bloc;
   2° Choisir un code pour chaque opération à exécuter (optimiser décodage);
   3° Définir les entrées et sorties nécessaires pour le chainage des modules;
   3° Optimiser la décomposition, éventuellement modifier les codes des opérations.
   4° Etablir un schéma final de la décomposition de l'unité RALU
3. Concevoir un détecteur de flanc montant dans le composant Detect_Flanc. Celui-ci sert à détecter le flanc montant du signal Pulse afin de générer sur Enable une impulsion synchrone d'une période du signal Horloge.

Chaque étudiant réalisera une version différente de la décomposition, pour les points 3 à 12

4. Créer un composant nommé RALU, modulaire de N bits, avec les E/S nécessaires définies lors de votre conception (chainage des modules).
5. Faire une description en VHDL synthétisable de l'unité RALU modulaire de Nbits correspondant au schéma conçu au point 2.
6. Tester la description VHDL de l'unité RALU modulaire de Nbits avec N = 4 à l'aide d'une simulation interactive avec Top_Sim et la console REDS. Pour lancer la simulation, executer le script "run_ralu.tcl" depuis Questasim.
7. Faire la synthèse du composant RALU modulaire pour un nombre de bits de 4, 8 et 12. Vous devez analyser l'évolution de la quantité de logique
8. Compléter le schéma du composant RALU_12bits_top. Vous fixerez le nombre de bits Nbits_C = 4 pour cette étape. Celui-ci comporte les entrées code, Pulse, Horloge, nReset, Operand (sur 12 bits) et les sorties Err_nsgn, Err_sgn et Result (sur 12 bit). Le composant Detect_Flanc doit être mappé dans RALU_12bits_top entre le signal Pulse et génère le signal Enable pour les 3 composants RALU. Le composant RALU_12bits_top ne dispose d'aucunes entrées et sorties de chainage. Celui-ci fonctionne pour des nombres de 12 bits qui sont soit non-signés ou soit signés en "complément à 2".

9. Tester le composant RALU_12bits_top à l'aide du banc de test fourni, RALU_12bits_Top_tb. Vous devrez compléter la table de constantes spécifiant quel code vous avez utilisé pour chacune des opérations dans le paquetage du projet (RALU_Pkg).
10. Faire la synthèse et le placement-routage du composant RALU_12bits_top pour un circuit MAX EPM7128SLC84. Vous devez utiliser le fichier d'assignation des pins fournis dans le projet (RALU_12bits_top_assignation_pins.tcl).
11. Tester ce circuit sur une carte EPM 25p-25p, à l'aide de la console RALU_12bits_console.tcl.
12. Une synthèse de la quantité de logique obtenue par chaque étudiant sera réalisée. Les variations de quantité devront être commentées.

Remarque: Un fichier " A_LIRE_AVANT_TOUT_v0_2 .txt" vous donne toutes les informations utiles pour l'utilisation du projet et des outils EDA.