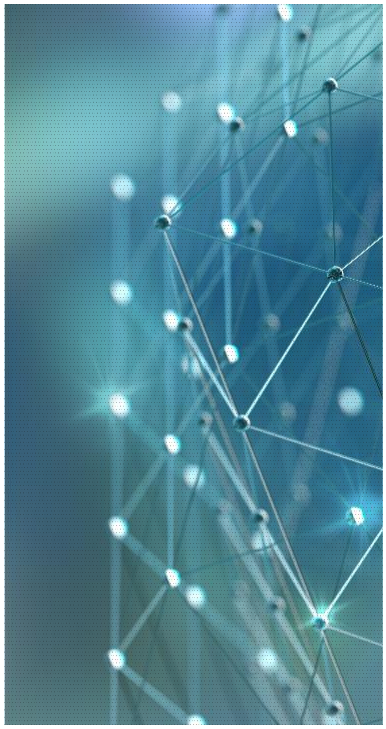


TE606 – Conception de systèmes numériques

Didier Meier

didier.meier@intervenants.efrei.net

2021-2022

1

Objectifs du module

- Concevoir et analyser un circuit logique (séquentiel et combinatoire) et un graphe d'état puis en réaliser les simulations et les synthèses sur un composant programmable
- Distinguer les cas d'utilisation d'un composant programmable par rapport aux CPU/GPU/ASIC/MCU
- Configurer un FPGA et connaitre les différentes étapes d'un outil d'EDA
- Décrire, simuler et synthétiser un système combinatoire et séquentiel à l'aide du langage de description VHDL
- Appliquer les règles essentielles de conception d'un circuit logique
- Maîtriser les techniques essentielles d'optimisation des composants programmables

2

Sommaire – conception numérique

- 1ère partie : des systèmes numériques aux composants à architectures programmables (CM : env. 5h)
  - Rappel des définitions systèmes combinatoires, séquentiels, synchrones, asynchrones… vers des structures génériques
  - De l'architecture générique d'un système séquentiel synchrone à l'architecture d'un composant programmable : PLD et CPLD
  - Evolution des architectures CPLD vers les architectures FPGA
- 2ème partie : l'utilisation d'un langage de description matériel pour la simulation et la synthèse de systèmes numériques : le VHDL (CM : env. 4h)
  - Première approche : synthèse VHDL d'un automate
  - Introduction au langage VHDL
  - Structure du langage VHDL
  - Signaux, variables et processus en VHDL
  - Typage des objets en VHDL

Sommaire – conception numérique

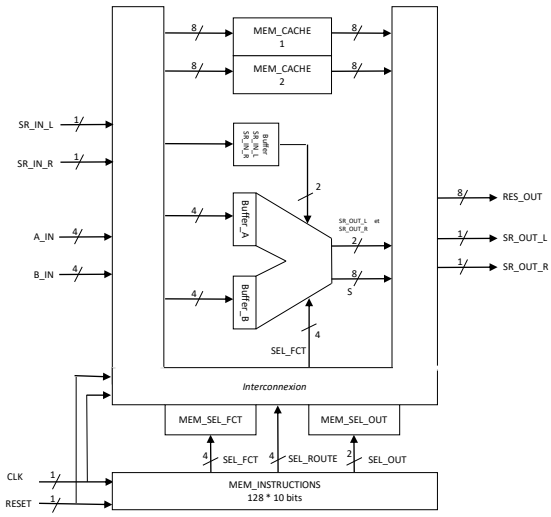
- 3ème partie : comment décrire, synthétiser et simuler en VHDL des systèmes numériques combinatoires et séquentiels (env. 2 → 3 CTP de 3h chacun)
  - Logique combinatoire
  - Logique séquentielle
- 4ème partie : optimisation des systèmes numériques sur composants programmable ; architectures et langage VHDL (CM : env. 1h)
  - Performance
- 5ème partie : conception, simulation et synthèse d'un microcontrôleur à l'aide du VHDL dans un composant à architecture programmable (PRJ : env. 9h)
  - Présentation du sujet et du fonctionnement cible
  - Conception théorique de l'architecture
  - Synthèse en VHDL
  - Réalisation des simulations (testbench) en VHDL

5ème partie : Projet

Conception, simulation et synthèse d'un microcontrôleur à l'aide du VHDL dans un composant à architecture programmable

Présentation générale de l'architecture cible

- L'objectif de ce projet consiste à réaliser un cœur de microcontrôleur intégrant :
  - Une unité de traitements arithmétiques et logiques
  - Des mémoires internes de calcul
  - Une mémoire d'instructions
- Ce cœur de microcontrôleur fonctionnera sur 4 bits codés en naturel signé pour les entrées et sur 8 bits codés en binaire naturel signé pour les sorties et les mémoires internes.
- A partir d'opérations de base, il est possible de réaliser l'ensemble des fonctions logiques et des opérations arithmétiques à l'aide d'automates (successions d'instructions).
  - Les fonctions plus « complexes » n'étant qu'une succession de fonctions simples avec parfois la mémorisation de résultats intermédiaires.
  - Il est donc nécessaire d'adjoindre à la structure de base
    - ✓ une mémoire interne permettant de stocker des résultats intermédiaires
    - ✓ Une mémoire interne permettant de stocker les instructions successives à réaliser
- Les derniers travaux consisteront à réaliser trois automates et de les implémenter dans la mémoire d'instructions.
  - Pour cette dernière partie, vous serez amenés à implémenter les trois automates permettant de piloter l'UAL afin de réaliser les fonctions choisies.



Fonctions réalisées par l'UAL

SEL_FCT[3]	SEL_FCT[2]	SEL_FCT[1]	SEL_FCT[0]	Significations
0	0	0	0	nop (no operation) S = 0 | SR_OUT_L = 0 et SR_OUT_R = 0
0	0	0	1	S = Déc. droite A sur 4 bits (avec SR_IN_L) | SR_IN_L pour le bit entrant et SR_OUT_R pour le bit sortant
0	0	1	0	S = Déc. gauche A sur 4 bits (avec SR_IN_R) | SR_IN_R pour le bit entrant et SR_OUT_L pour le bit sortant
0	0	1	1	S = Déc. droite B sur 4 bits (avec SR_IN_L) | SR_IN_L pour le bit entrant et SR_OUT_R pour le bit sortant
0	1	0	0	S = Déc. gauche B sur 4 bits (avec SR_IN_R) | SR_IN_R pour le bit entrant et SR_OUT_L pour le bit sortant
0	1	0	1	S = A * B multiplication binaire | SR_OUT_L = 0 et SR_OUT_R = 0
0	1	1	0	S = A + B addition binaire avec SR_IN_R comme retenue d'entrée | SR_OUT_L = 0 et SR_OUT_R = 0
0	1	1	1	S = A + B addition binaire sans retenue d'entrée | SR_OUT_L = 0 et SR_OUT_R = 0
1	0	0	0	S = A – B soustraction binaire | SR_OUT_L = 0 et SR_OUT_R = 0
1	0	0	1	S = A | SR_OUT_L = 0 et SR_OUT_R = 0
1	0	1	0	S = B | SR_OUT_L = 0 et SR_OUT_R = 0
1	0	1	1	S = not A | SR_OUT_L = 0 et SR_OUT_R = 0
1	1	0	0	S = not B | SR_OUT_L = 0 et SR_OUT_R = 0
1	1	0	1	S = A and B | SR_OUT_L = 0 et SR_OUT_R = 0
1	1	1	0	S = A or B | SR_OUT_L = 0 et SR_OUT_R = 0
1	1	1	1	S = A xor B | SR_OUT_L = 0 et SR_OUT_R = 0

7

Les mémoires

- Les mémoires Buffer_A et Buffer_B
  - Les mémoires Buffer_A, Buffer_B permettent de stocker les données directement liées au cœur de l'UAL, c'est-à-dire à la sous-fonction arithmétique et logique.
  - Elles seront chargées (activées sur front montant de l'entrée clk) suivant les valeurs de l'entrée SEL_ROUTE.
- Les mémoires MEM_SR_IN_L et MEM_SR_IN_R permettent de stocker les valeurs des retenues d'entrées.
  - Elles sont systématiquement mémorisées à chaque front montant de l'horloge CLK.
- La mémoire MEM_SEL_FCT permet de mémoriser la fonction arithmétique ou logique à réaliser.
  - Elle est systématiquement chargée à chaque front montant d'horloge.
- Les mémoires MEM_CACHE_1 et MEM_CACHE_2
  - Les mémoires MEM_CACHE_1 et MEM_CACHE_2 permettent de stocker les données des entrées A_IN et B_IN ou les résultats intermédiaires de la sortie S, c'est-à-dire à la sortie de la sous-fonction arithmétique et logique.
    - ✓ Il est à noter que les sorties SR_OUT_L et SR_OUT_R ne peuvent pas être mémorisées.
    - ✓ Ces deux mémoires seront chargées (activées sur front montant de l'entrée CLK) suivant les valeurs de l'entrée SEL_ROUTE.
- La mémoire MEM_SEL_OUT
  - La mémoire MEM_SEL_OUT permet de stocker les données de l'entrée SEL_OUT.
  - Elle est systématiquement chargée à chaque front montant de l'horloge CLK.
- La mémoire MEM_INSTRUCTIONS
  - La mémoire MEM_INSTRUCTIONS permet de stocker les instructions successives à réaliser
  - Son pointeur est systématiquement indenté à chaque front montant de l'horloge CLK
- SEL_ROUTE permet de définir le transfert de données qui sera effectué lors du prochain cycle horloge (prochain front montant de l'horloge).
  - Elle ne doit donc pas être mémorisée comme MEM_SEL_FCT et MEM_SEL_OUT

8

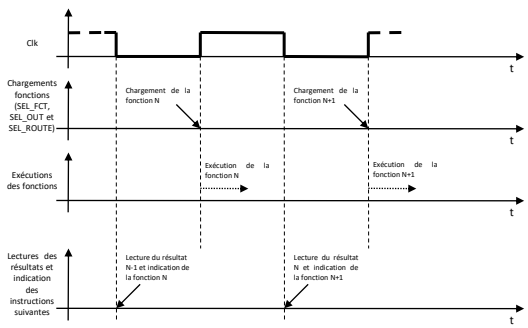
Les interconnexions et le routage des données

SEL_ROUTE3	SEL_ROUTE2	SEL_ROUTE1	SEL_ROUTE0	Significations
0	0	0	0	Stockage de l'entrée A_IN dans Buffer_A
0	0	0	1	Stockage de l'entrée B_IN dans Buffer_B
0	0	1	0	Stockage de S dans Buffer_A (4 bits de poids faibles)
0	0	1	1	Stockage de S dans Buffer_A (4 bits de poids forts)
0	1	0	0	Stockage de S dans Buffer_B (4 bits de poids faibles)
0	1	0	1	Stockage de S dans Buffer_B (4 bits de poids forts)
0	1	1	0	Stockage de S dans MEM_CACHE_1
0	1	1	1	Stockage de S dans MEM_CACHE_2
1	0	0	0	Stockage de MEM_CACHE_1 dans Buffer_A (4 bits de poids faibles)
1	0	0	1	Stockage de MEM_CACHE_1 dans Buffer_A (4 bits de poids forts)
1	0	1	0	Stockage de MEM_CACHE_1 dans Buffer_B (4 bits de poids faibles)
1	0	1	1	Stockage de MEM_CACHE_1 dans Buffer_B (4 bits de poids forts)
1	1	0	0	Stockage de MEM_CACHE_2 dans Buffer_A (4 bits de poids faibles)
1	1	0	1	Stockage de MEM_CACHE_2 dans Buffer_A (4 bits de poids forts)
1	1	1	0	Stockage de MEM_CACHE_2 dans Buffer_B (4 bits de poids faibles)
1	1	1	1	Stockage de MEM_CACHE_2 dans Buffer_B (4 bits de poids forts)

SEL_OUT1	SEL_OUT0	Significations
0	0	Aucune sortie : RES_OUT = 0
0	1	RES_OUT = MEM_CACHE_1
1	0	RES_OUT = MEM_CACHE_2
1	1	RES_OUT = S

Synchronisation temporelle

- Le diagramme ci-contre présente le fonctionnement temporel que devra respecter le montage
  - En cas de pilotage externe (sans mémoire d'instructions) pour tests
  - En interne avec la mémoire d'instructions
- Les composants externes utilisant cette UAL modifient les valeurs des entrées sur les fronts descendants de l'horloge CLK.
  - Au prochain font montant, l'UAL exécute les instructions.
  - Les résultats sont lus lors du front descendant suivant (tout en indiquant la nouvelle fonction à réaliser).
- L'entrée RESET permet d'initialiser de manière asynchrone toutes les mémoires à 0



Les instructions à coder pour tests et validations

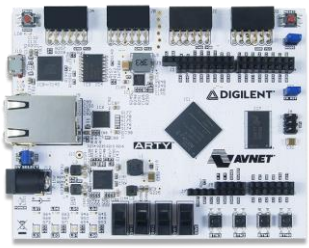
- Vous aurez à implémenter 3 fonctions spécifiques dans la mémoire à instructions :
  1) RES_OUT_1 = (A mult. B) *(RES_OUT_1 sur 8 bits)*
  2) RES_OUT_2 = (A add. B) xnor A *(xnor sur les 4 bits de poids faibles – RES_OUT_2 sur 4 bits)*
  3) RES_OUT_3 = (A_0 and B_1) or (A_1 and B_0) *(RES_OUT_3 sur le bit de poids faible)*

- On considère les entrées A, B, SR_IN_L et SR_IN_R stables toute la durée du calcul :
  - Ce qui, en réalité, ne sera pas toujours le cas…

- La sortie RES_OUT restera à 0 tant que le résultat final du calcul ne sera pas disponible (pas de résultats intermédiaires sur RES_OUT)
  - Une fois le résultat calculé, il sera maintenu sur la sortie RES_OUT jusqu'au prochain lancement du calcul
    - ✓ Pour les tests sur la carte FPGA, un signal spécifique sera également passé de 0 (calcul en cours) à 1 (calcul terminé) pour indiquer que le résultat RES_OUT est disponible

Définition de l'environnement de travail

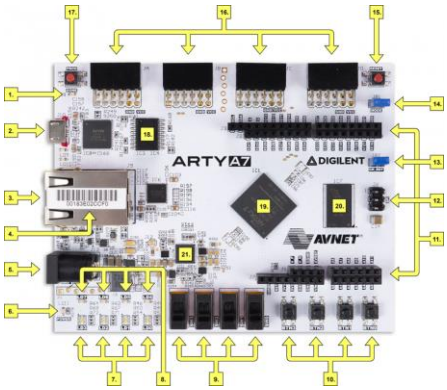
- Environnement de développement et de simulation :
  - EDA Playground
  - Xilinx Vivado 2018.2 (ou version ultérieure)



- Maquette de développement :
  - Carte ARTY de Digilent
    - ✓ Xilinx Artix-35T FPGA
  - La maquette de développement n'est pas nécessaire pour la réalisation du projet
    - ✓ Néanmoins, elle sera utilisée pour réaliser les implémentations lors de la dernière séance de projet

Intégration sur la carte de développement ARTY

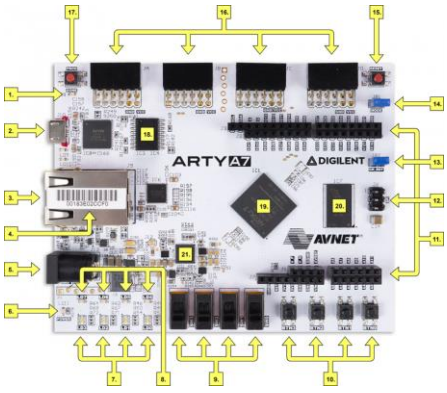
- La carte de développement



Callout	Description	Callout	Description	Callout	Description
1	FPGA programming DONE LED	8	User RGB LEDs	15	chipKIT processor reset
2	Shared USB JTAG / UART port	9	User slide switches	16	Pmod connectors
3	Ethernet connector	10	User push buttons	17	FPGA programming reset button
4	MAC address sticker	11	Arduino/chipKIT shield connectors	18	SPI flash memory
5	Power jack for optional external supply	12	Arduino/chipKIT shield SPI connector	19	Artix FPGA
6	Power good LED	13	chipKIT processor reset jumper	20	Micron DDR3 memory
7	User LEDs	14	FPGA programming mode	21	Dialog Semiconductor DA9062 power supply

Intégration sur la carte de développement ARTY

- Réalisation de tests sur la carte de développement :
  - Fonctionnement à 100 MHZ (clk)
  - A = B :
    - ✓ SW(3-0) : User slide switches
  - SR_IN_L et SR_IN_R = 0 (non utilisés pour le test)
  - Reset global :
    - ✓ btn(0) : User push buttons
  - Lancement des calculs :
    - ✓ btn(1) : RES_OUT_1
    - ✓ btn(2) : RES_OUT_2
    - ✓ btn(3) : RES_OUT_3
  - Résultats de calculs :
    - ✓ 8 leds (LD_7 à LD_0) avec du vert
    - ✓ Résultat disponible : $8^{ème}$ led (LD_0) avec du rouge (LSB)
    - ✓ SR_OUT_L : $5^{ème}$ led (LD_3) avec du bleu
    - ✓ SR_OUT_R : $6^{ème}$ led (LD_2) avec du bleu



Description de l'entité globale et fichier de contraintes

- Pour l'implémentation sur FPGA, vous devrez respecter la description de l'entité globale afin que les ports d'entrées et de sorties puissent correspondre avec la carte de développement :
  - La description de l'entité est disponible en téléchargement sur votre espace Moodle

- Afin de permettre de réaliser les tests sur carte FPGA, vous devrez utiliser le fichier de contraintes spécifique à la carte de développement :
  - Le fichier de contraintes est disponible en téléchargement sur votre espace Moodle

Livrables attendus et évaluation

- Définition des livrables (rapports et archives) et résultats attendus (validations)
  - A la fin de la période dédiée à la réalisation du projet, vos résultats seront évalués en combinant :
    - ✓ Une validation technique (intégration du montage dans un composant programmable et réalisation d'une séquence automatique de tests).
    - ✓ Une validation théorique dans l'environnement de développement (à l'aide de vos archives de projets et de votre rapport technique) via différentes simulations.
    - ✓ L'évaluation de votre rapport technique du projet.
  - Il est donc primordial de fournir les différents éléments demandés.
    - ✓ Aucun retard ne sera toléré.
    - ✓ Tout élément manquant lors de la remise des archives pourra nuire à l'évaluation de votre projet.
      - Soyez donc vigilent à respecter scrupuleusement les éléments demandés.

- Livraisons de documents et d'archives
  - A la fin de la période impartie à la réalisation du projet, il vous sera demandé de fournir :
    - ✓ 1 archive ZIP au format « noms_des_élèves_du_binôme ».zip contenant
      - L'ensemble de votre(vos) dossier(s) du projet
      - 1 fichier au format « noms_des_élèves ».txt contenant un résumé de chacune de vos entités en VHDL
      - 1 rapport détaillé du mini-projet au format électronique (.doc ou .pdf).

- L'archive devra être déposée sur Moodle avant la fin du délai de réalisation du projet
  - Aucun retard ne sera toléré
  - Le transfert de ces fichiers par courriel n'est pas accepté

Fin de la partie 5

Fin du module de conception des systèmes numériques

17