Notes de Cours ELE8307

Olivier Sirois

2017-10-10

Contents

Chapter 1

Introduction

Conception de systèmes logiques

Objectif : Apprendre à concevoir des prototypes de circuits avec FPGA et de perfectionner les habiletés de travail en groupe.

Chapter 2

Cours 1 - Architecture des FPGAs

1946 : ENIAC
plusieurs caractéristique (2 moteurs), occupait beuacoup d'espace. 5000 additions par seconde (ou 40 division par seconde). 385 mul par second, 3 racine carrée par sec..

merveilleux a l'époque.

par contre, c'étais pas simple à programmer, il fallait faire les connections manuellement. C'étais effectivement dans les premiers ordinateurs. un 'bug' était réellement un insecte qui allait se placer dans l'ordinateur, il y en avait en moyenne 1 par jour, et sa pouvait frier toute le système..

C'est avec sa que Von Neumann à eu son idée géniale..

INSÉRÉ SLIDE 6

Par contre, c'étais tellement lourd de changé l'opération entre chaque instruction que Von Neumann à ajouté une nouvelle connection entre le contrôleur de la mémoire (chemin pointillé bleu). C'est de la ou est née l'idée de mettre les instructions en mémoire pour que le contrôleur puisse manuellement aller lire les instructions. C'est de la ou est née le paradigme classique, une section de la mémoire était réservé pour les instructions tandis que le reste était utilisé pour les données.

2.1 Charactéristiques

2.1.1 Niveau sur reconfiguration

Le programme

le niveau le plus élevé sur lequel nous alons travaillé.

jeu d'instruction

un instruction est une série d'opération qui va etre fait entre les registre. Nous allons définir nous même notre jeu d'instruction comme étant des séquences de micro-ops effectié sur les registre/ALUs. sa devient plus lourd à programmer/debugger

fonctions logique de haut-niveau

(ALU, coprocessor). À l'époque, beaucoup de recherche sur arch de niveau moyen, sa revient mais sous la forme de 'coarse-grained' sur FPGA (IP block). On était capable de venir configuré un chemin de donnée auto-synchroniser composé d'ALU.. etc.

on reste toujours au niveau de traitement de donnée.

fonctions logique de bas niveau

(LUT,PLA, porte, transistor). au niveau binaire. On peut changer le comportement des LUT sur les FPGA, (comprend les PLA). Sa peut même aller jusqu'au niveau du transistor. Évidemment, c'est beaucoup plus compliquer sauf que le gain de performance en vaut la chandelle.

Interconnexions

On peut même programmer les interconnexions entre les différentes composante logique de bas niveau. Sa donne une reconfigurabilité presque infini. Juste en interconnectant les transistor de manière astucieuse on a un degrée de configuration infini. Toute la complexité peut se résumer à la complexité de leur interconnexion. Évidemment, les interconnexion programmable prend beaucoup de place.. 80% de la configurabilité du FPGA est dans les interconnexions.

2.1.2 Taille

Le nombre d'élément configurable.. le nom le dit.

On essaie toujours de ramener sa a un équivalent en porte logique (NOR). sauf que c'est de la bouette.. souvent, les compagnies font leur benchmark sur des

circuits qui sont hyperspécifique et qui se fit exactement sur le FPGA. on peut souvent voir un facteur 8 entre ce qu'ils annoncent et ce qui fit vraiment sur le FPGA.

Maintenant on va surement parler d'élément reconfigurable (LUT à 4 entrées.. etc.). C'est beaucoup plus représentatif de la performance/taille. Il faut vraiment avoir une notion poussé du systèmes pour pouvoir le comparés.

2.1.3 Vitesse

le délais, fréquence d'opérations

2.1.4 Caractéristique physique

Les tension d'entrées, d'opérations

2.1.5 Méthode de configuration

protocole de configuration (série, parallèle, maître-esclave, cryptage..).

Aujourd'hui, presque toute les technologies propriétaire est dans le chip, alors souvent les compagnies offrent des options de cryptage pour pouvoir encrypter la mémoire sur le chips pour pouvoir protéger les designs des compagnies (et les votres).
Persistence. Si on coupe le courant est-ce que le circuit reste ?

2.1.6 reconfiguration partielle en temps réelle

Certain FPGA offrent des fonctionnalités ou il peuvent changer certaines zones en opration. C'est comme un circuit mutant qui peut se modifier pendant qu'il roule. Il peut aller chercher des sous-configurations en mémoire et la loader directement sur le fabric

2.1.7 Disponibilité d'IP

On peut avoir accès à des gros blocs de construction IP. C'est bloque sont super complexe en générale, souvent les compagnies peuvent vendre leur bloque pour que les concepteur puissent les utiliser. Souvent, en temps que designer, c'est mieux de pouvoir ce servir de modules déja concus pour accélérer le processus de développement + plus de performance

2.2 Architecture de système numérique

INSÉRÉ SLIDE 8

2.2.1 Mémoire

La mémoire est le circuit combinatoir configurable par excellence, en fait, c'est sa le but.

Avec de la mémoire, on peut théoriquement réalisé n'importe quel circuit combinatoire. En utilisant la mémoire de particulière, on peut créer des effets particulière. On peut utiliser certains bits de la mémoire pour faire des contrôles spéciales (bits D0 = 1 seulement lorsque addr=0x8), cela donne l'équivalent d'un ET logique, vue que l'addresse est un partout. On peut alors, en utilisant la valeur en mémoire, recrée une table de karnaugh dans la mémoire et on utilise la valeur de l'addresse pour faire une sorte de 'query' de la table de Karnaugh.

Avec cette puissante fonctionnalité, on peut recréer n'importe qu'elle fonction logique. n'importe quel circuit combinatoire peut s'implanter avec une mémoire.

Techniquement, aucun calcul n'est réalisé, la valeurdes fonctions est mémoirsé pour chaque combinaison des entrées. On peut alors implémenter:
$D * 2^A$
$A = D + K$

fonctions.

Décodeur texte

On peut voir que c'est magique pouvoir faire un circuit avec de la mémoire, sauf qu'avec l'exemple de décodage, sa prend pas grande chose pour faire un circuit de 659 milles annees lumiere de cotes..

C'est magique lorsque le nombre d'entrées est petits.