Simulation Logico temporelle de circuits VLSI a l'aide d'un réseau de FPGA

Laurent VUILLEMIN, Pirouz BAZARGAN-SABET
LIP6 – ASIM
tour 55–65 2^{e} étage
4 place Jussieu
75252 Paris cedex 05
Email : {Pirouz.Bazargan-Sabet, Laurent.Vuillemin}@lip6.fr

27 février 1999

Introduction

La complexité sans cesse croissante des circuits VLSI digitaux augmente considérablement la durée des traitements et les ressources de calcul requises par le processus de vérification. Sachant qu'environ 70% du temps de développement d'un circuit est consacré à sa vérification, il est aisé de comprendre à quel point, la performance des outils de CAO peut être stratégique pour les concepteurs de circuits intégrés. La simulation logique est certainement l'un des outils qui occupe la plus grande partie du temps de vérification et réduire sa durée peut se traduire par un gain substantiel dans le temps de développement du circuit.

Afin d'augmenter la vitesse de simulation plusieurs approches ont été proposées comme la simulation distribuée sur un réseau de stations de travail ou les accélérateurs matériels de simulation. Mais la simulation la plus rapide est obtenu par l'usage d'émulateurs. En première approximation, de tels systèmes peuvent être vus comme de larges circuits reprogrammables. La description du circuit à vérifier est projetée sur ce réseau après une phase de synthèse. On obtient alors un prototype matériel dont la fréquence d'horloge est typiquement de 10 a 100 fois inférieure à celle du circuit réel. Toutefois, contrairement à l'ensemble des solutions précédentes, l'émulation fait disparaître toute information sur le fonctionnement temporel du circuit. L'objectif de notre travail est d'obtenir, à l'aide d'un émulateur, une simulation logico temporelle dont les performances en termes de vitesse sont comparables à celle d'un accélérateur de simulation. L'approche proposée dans cet article consiste à projeter sur l'émulateur un algorithme de simulation qui a déjà fait l'objet d'une réalisation logicielle dans notre laboratoire.

1 Architecture du simulateur

L'approche objet, utilisée pour le développement logiciel d'un simulateur logique nous a permis d'isoler les différents éléments de notre prototype matériel de simulation. Ainsi l'accélérateur de simulation se compose de 4 grandes parties :

- Un automate de contrôle;
- La représentation du circuit;
- La file des événements;
- l'échéancier.

L'automate de contrôle reproduit l'algorithme de simulation avec ses différentes phases. Nous avons opté pour l'algorithme de simulation à pilotage événementiel. Dans le simulateur logiciel, le circuit est représenté par des structures de données. Celles ci sont fondamentalement statiques. Il est donc possible de matérialiser ces structures par des mémoires. A chaque signal correspond une entrée dans ces mémoires, l'identificateur d'un signal est son adresse mémoire. Le caractère reconfigurable du réseau cible nous permet d'adapter la taille du simulateur au nombre de signaux du circuit.

A chaque signal est associée une expression booléenne qui permet de calculer la valeur du signal en fonction de la valeur des entrées. L'évaluation des expressions se fait en présentant aux entrées d'un multiplexeur la table de vérité de l'expression. Les commandes sont alors les valeurs des entrées de l'expression. Cette méthode d'évaluation a l'avantage de ne nécessiter que très peu de ressources matérielles tout en permettant une vitesse importante. L'expression est alors représentée par sa table de verité.

Le délai de propagation est représenté par un entier, le nombre de bits de cet entier est calculé en fonction du délai le plus grand du circuit. On peut ici tirer largement parti du caractère reprogrammable de l'architecture en adaptant cette représentation en fonction de l'ensemble des délais présents dans le circuit et de la précision souhaitée par l'utilisateur.

La liste des dépendances arrières se limite aux quatre adresses des signaux d'entrée de l'expression. Ces trois éléments sont regroupés dans une seule mémoire.

Le cas de la liste des dépendances avant est légèrement plus complexe. En effet un signal peut être utilisé par un nombre indéterminé d'expressions. Contrairement au cas des dépendances arrières, le nombre d'éléments de la liste n'étant pas fixe, il faut utiliser un mécanisme d'indirection. Une première mémoire contient pour chaque signal les adresses de début et de fin de la liste, une seconde mémoire contient les listes proprement dites.

L'algorithme event driven nécessite que l'on stocke la valeur courante et la valeur prévue, c'est à dire la valeur calculée au cycle courant, de chaque signal. Il faut donc une mémoire pour chacune de ces valeurs. Au cours de la phase d'évaluation de l'expression on doit lire au même cycle la valeur courante des quatre entrées. La mémoire de la valeur courante doit donc permettre quatre accès en lecture et un accès en écriture.

Deux mémoires sont utilisées pour la gestion de l'algorithme. Une contient les adresses des signaux à évaluer, l'autre les adresses des signaux à mettre à jour. L'échéancier est la structure qui subit le plus de modifications dans le passage du logiciel au matériel. Dans le cas du logiciel il s'agit d'une liste qui permet l'insertion d'une transaction à n'importe quel endroit en fonction de la date calculée. Ce mécanisme est trop complexe pour être directement transposé en matériel. Ici nous avons choisi de privilégier la rapidité et la simplicité aux dépends de l'encombrement. L'échéancier peut être vu comme une matrice à deux dimensions comprenant une ligne par signal et une colonne par date, à l'intersection d'une ligne et d'une colonne on stocke la valeur du signal à la date donnée.

2 Les aspects logiciels : la projection du circuit

Ce simulateur nécessite un pré-traitement logiciel du circuit à simuler. En effet il n'admet que des expressions à quatre entrées. Il faut donc découper les expressions originales de telle sorte qu'elles ne disposent plus que de quatre entrées. De plus on cherche dans cette phase à minimiser le nombre de signaux du circuit en regroupant les expressions d'arité inférieure à quatre. Cette optimisation ne peut être effectuée par les outils classiques de synthèse car on souhaite conserver les informations de délai associées à chaque signal.

3 La distribution

La projection d'un simulateur logiciel en matériel à elle seule ne suffit pas pour répondre à l'exigence de performance que nous souhaitons atteindre. Nous proposons donc de construire un simulateur distribué synchrone en assemblant plusieurs simulateurs élémentaires dont nous venons d'esquisser l'architecture. Le caractère reprogrammable peut être utilisé pour adapter l'architecture au partitionnement. Celui ci est obtenu par répartition équitable de la charge, calculée par une méthode probabiliste, sur chaque simulateur élémentaire.