

N° d'ordre: 3168

# THÈSE

Présentée devant

devant l'Université de Rennes 1

pour obtenir

le grade de : DOCTEUR DE L'UNIVERSITÉ DE RENNES 1  
Mention INFORMATIQUE

par

**Assia DJABELKHIR**

Équipe d'accueil : CAPS  
École Doctorale : MATISSE  
Composante universitaire : IFSIC/IRISA

Titre de la thèse :

***Etude de l'exécution dynamique et/ou spéculative et des  
processeurs enfouis : un cas d'étude de l'architecture  
découplée***

soutenue le 31 Mars 2005 devant la commission d'examen

MM. :	Nathalie DRACH	Rapporteurs
	Daniel LITAIZE	
MM. :	Thierry COLLETTE	Examineurs
	François BODIN	
	André SEZNEC	

# Table des matières

<b>Introduction générale</b>	<b>13</b>
<b>I État de l'art</b>	<b>17</b>
<b>1 Processeurs embarqués VS processeurs à usage général</b>	<b>19</b>
1.1 Les systèmes embarqués . . . . .	20
1.1.1 La technologie des processeurs embarqués . . . . .	21
1.1.2 De nouveaux challenges pour la recherche . . . . .	24
1.2 Les architectures à usage général . . . . .	24
1.2.1 Le processeur superscalaire . . . . .	25
1.2.2 Les processeurs superscalaires à exécution dans l'ordre ( <i>In-Order</i> ) . . . . .	25
1.2.3 Les processeurs superscalaires à exécution dans le désordre ( <i>Out-Of-Order</i> ) . . . . .	26
1.2.4 L'architecture découplée . . . . .	27
1.3 Tendances à l'intégration . . . . .	29
1.4 Problématique générale . . . . .	30
<b>2 Les architectures découplées</b>	<b>33</b>
2.1 Le principe des architectures découplées . . . . .	33
2.1.1 Découpler les accès mémoire ( <i>Decoupled Access/Execute architecture</i> ) . . . . .	34
2.1.2 Découpler le flot de contrôle (DCAE) . . . . .	36
2.2 Partitionnement et ordonnancement des flots d'instructions . . . . .	37
2.2.1 Partitionnement statique . . . . .	38
2.2.2 Partitionnement dynamique . . . . .	40
2.3 Exemples de processeurs découplés . . . . .	40
2.3.1 Le MIPS R8000 . . . . .	40
2.3.2 Le DSP-MCU ST100 . . . . .	41
2.4 <i>Loss Of Decoupling</i> (LOD) . . . . .	42
2.5 Conclusion . . . . .	43
<b>II EDA : une architecture découplée pour le domaine embarqué</b>	<b>45</b>
<b>3 Étude des caractéristiques des applications enfouies en vue d'une exécution sur une architecture découplée</b>	<b>47</b>
3.1 Description de l'architecture EDA . . . . .	48

3.2	Description de la méthodologie d'étude . . . . .	49
3.2.1	La régularité/prédictibilité du contrôle . . . . .	49
3.2.2	Les dépendances contrôle-mémoire (CMD) . . . . .	50
3.2.3	Les motifs de référence de données (DRP) . . . . .	51
3.3	Environnement de l'étude . . . . .	52
3.3.1	MiBench, une suite d'applications caractéristiques du domaine enfoui . . . . .	52
3.3.2	calvin2+DICE simulator . . . . .	55
3.4	Etude de la régularité des applications enfouies : évaluation et discussion . . . . .	56
3.4.1	Etude des boucles les plus internes . . . . .	56
3.4.2	Etude des dépendances contrôle-mémoire (CMD) . . . . .	60
3.4.3	Les motifs de référence de données (DRP) . . . . .	65
3.5	Conclusion . . . . .	68
<b>4</b>	<b>Etude d'une nouvelle organisation du cache suivant les motifs de référence de données</b> . . . . .	<b>71</b>
4.1	Configuration du cache de données suivant les DRP . . . . .	71
4.2	Le marquage statique/dynamique des DRP . . . . .	72
4.3	Etude de la politique de placement mémoire pour réduire la latence des accès indirects . . . . .	73
4.3.1	La méthodologie de simulation . . . . .	74
4.3.2	La fonction de coût . . . . .	75
4.3.3	Evaluation et discussion . . . . .	75
4.4	Travaux relatifs . . . . .	82
4.4.1	Les approches pour la performance . . . . .	82
4.4.2	Les approches pour réduire la consommation . . . . .	83
4.5	Conclusion . . . . .	85
<b>5</b>	<b>Une implémentation de l'architecture découplée EDA pour un jeu d'instructions classique</b> . . . . .	<b>87</b>
5.1	Description de l'architecture découplée EDA . . . . .	87
5.1.1	Sous-processeur d'exécution . . . . .	88
5.1.2	Sous-processeur d'accès mémoire . . . . .	89
5.1.3	Sous-processeur de contrôle . . . . .	90
5.2	Partitionner le flot d'instructions pour un jeu d'instructions à un fichier de registres unique . . . . .	90
5.2.1	Exemples de partitionnement dynamique . . . . .	92
5.3	Une mise en œuvre d'un processeur découplé à un jeu d'instructions ayant un fichier de registres unique . . . . .	94
5.3.1	Une copie du fichier de registres par sous-processeur . . . . .	95
5.3.2	Chaque sous-processeur reçoit une copie du code . . . . .	96
5.3.3	Scénario d'exécution sur le sous-processeur d'exécution . . . . .	97
5.3.4	Scénario d'exécution sur le sous-processeur de contrôle . . . . .	99
5.4	La validation des instructions : unité de validation . . . . .	100
5.4.1	La mise en œuvre de l'unité de validation . . . . .	101
5.5	Le mécanisme de découplage Vs le mécanisme d'exécution dans le désordre . . . . .	102
5.5.1	Le renommage des registres . . . . .	102
5.5.2	La logique d'émission . . . . .	103

5.5.3	Mécanisme de réparation des défauts de prédiction de branchements	104
5.6	Compiler pour une architecture découplée EDA . . . . .	105
5.6.1	Partitionnement statique du code . . . . .	105
5.6.2	Optimisation/ordonnancement statique du code . . . . .	107
5.7	DS-sim : Un simulateur <i>cycle-accurate</i> dirigé par la trace pour une architecture découplée EDA . . . . .	107
5.7.1	La mise en œuvre de DS-sim . . . . .	108
5.7.2	Etude de performance de l'architecture découplée EDA . . . . .	109
5.8	Conclusion . . . . .	120
	<b>Conclusion générale</b>	<b>123</b>

## Résumé

Dans cette thèse, nous proposons d'utiliser l'architecture découplée pour la conception des processeurs embarqués de haute performance. Cette proposition est motivée par la complexité des nouvelles applications enfouies grand public. L'exécution découplée constitue une solution intermédiaire entre l'exécution dans l'ordre, simple mais dont les performances restent limitées, et l'exécution dans le désordre, qui permet de bonnes performances, mais qui s'avère complexe et coûteuse.

L'architecture découplée atteint de hautes performances si ses sous-processeurs sont complètement découplés en exécutant les flots d'instructions du code. L'apparition de certaines dépendances entre les sous-processeurs cause une perte de découplage, qu'on appelle un événement LOD (*Loss Of Decoupling*). Les événements LOD constituent la cause principale de perte de performance sur une architecture découplée. Par conséquent, il est important que le code s'exécutant sur ce type d'architecture soit "découplable", tel est le cas des applications régulières.

Dans cette thèse, nous avons étudié les caractéristiques d'un ensemble de benchmarks enfouis afin de comprendre et prédire leur comportement sur une architecture découplée, et de quantifier le risque d'occurrence des événements LOD. Cette étude a permis de classer les applications enfouies en catégories suivant leur régularité et a montré que la quasi-totalité de ces applications peuvent être amenées à de bonnes performances sur une architecture découplée. Pour d'autres applications, nous proposons une politique de placement des données mémoire "critiques" qui a pour but de réduire l'impact, parfois dramatique, des irrégularités causées par l'utilisation de ces données. Nous avons présenté un cas d'étude de la politique de placement des données utilisées comme adresses pour les accès indirects, afin de réduire la latence des dépendances mémoire-mémoire.

Nous avons montré qu'une architecture découplée EDA, constituée de trois sous-processeurs découplés, peut être implémentée pour un jeu d'instructions classique à un fichier de registres unique. Nous avons présenté un mécanisme de partitionnement dynamique du code sur les trois sous-processeurs, qui se base sur le partitionnement dynamique du fichier de registres en sous-ensembles de registres spécifiques pour chaque type de calcul : accès, contrôle, ou traitement. Nous avons présenté une mise en œuvre de cette architecture et décrit les scénarios de communication entre les sous-processeurs découplés ainsi que la validation dans l'ordre des instructions. Ces mécanismes ont été mis en œuvre par DS-sim, un simulateur de l'architecture découplée EDA.

*Mots-clés : processeurs embarqués, architecture découplée, benchmarks enfouis, régularité du code, caractéristiques des applications, loss of decoupling (événements LOD), caches mémoire, simulation de la microrarchitecture, simulateur DS-sim.*