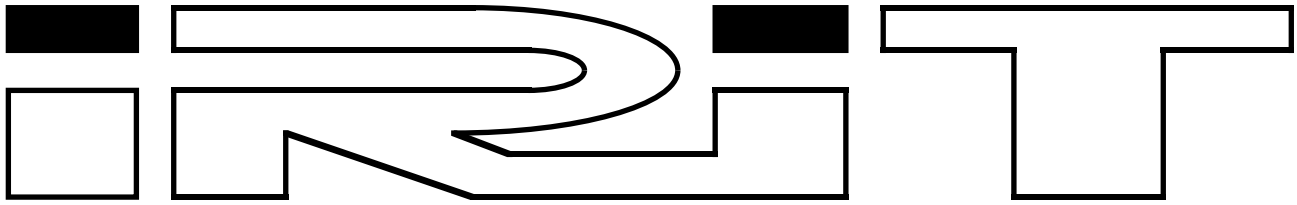
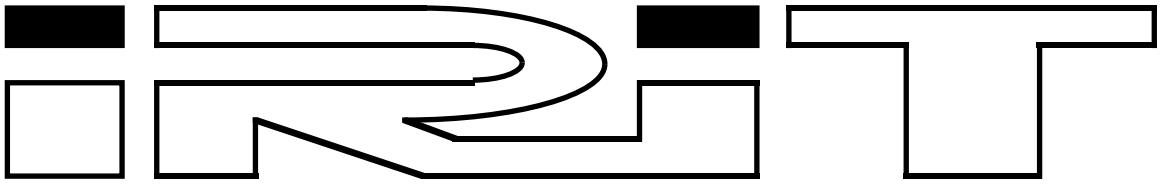


INSTITUT DE RECHERCHE EN INFORMATIQUE DE TOULOUSE

RAPPORT
INTERNE
N° 94_47_R

Configuration Exécutive des processeurs superscalaires à exécution non ordonnée

Stéphan Jourdan, Pascal Sainrat



INSTITUT DE RECHERCHE EN INFORMATIQUE DE TOULOUSE

RAPPORT
INTERNE
N° 94_47_R

Configuration Exécutive des processeurs superscalaires à exécution non ordonnée

Stéphan Jourdan, Pascal Sainrat

Novembre 1994 — 38 pages

Résumé : Cette étude a été menée afin de déterminer des configurations exécutives efficaces dans l'optique de processeurs superscalaires à initiation non ordonnée dont les degrés d'initiation varient entre 2 et 8. Des simulations dirigées par des traces ont été effectuées sur les 6 programmes test entiers et les 14 programmes test flottants du paquetage SPEC 92. Nous évaluons tout d'abord le nombre d'instructions qui peuvent être traitées simultanément par les différents pipelines. On applique alors certaines restrictions sur l'initiation des exécutions en rapport avec les classes d'instructions. Nous en concluons différentes configurations qui varient entre 6 et 9 unités fonctionnelles suivant le degré d'initiation du processeur. Un point important réside dans le fait que l'unité mémoire doit avoir plusieurs ports d'accès lorsque le degré d'initiation est au moins égal à quatre. Nous avons aussi montré que les instructions de multiplication entière et flottante peuvent être traitées par une même unité, offrant ainsi un bon compromis coût-performance. Ceci est également vrai pour les instructions de division. Finalement, nous rapportons les taux d'utilisation des différentes unités définies.

Abstract : This study has been carried out in order to determine cost-effective configurations of functional units for multiple-issue out-of-order superscalar processors. The trace-driven simulations were performed on the six integer and the fourteen floating-point programs from the SPEC 92 suite. We first evaluate the number of instructions allowed to be concurrently processed by the execution stages of the pipeline. We then apply some restrictions on the execution issue of different instruction classes in order to define these configurations. We conclude that six to nine functional units are necessary to exploit Instruction-Level Parallelism. An important point is that the Load/Store unit must have at least two data cache ports, especially in a 4 or more-issue processor. We also show that integer and floating-point multiply instructions can be cost-effectively handled by the same unit; this is also true for divide instructions. Finally, we report complementary results on the utilization rate of the functional units.

1. Introduction

De nos jours, l'approche superscalaire est unanimement reconnue, mais elle présente de nombreuses possibilités d'implémentation. Par exemple, l'implémentation d'une *Table d'Historique des Branchements*[1] et d'un *Tampon de Destination de Branchement*[2] peut être envisagée dans le cas d'une exécution spéculative [YePa92], ainsi que la présence de *stations de réservation* afin d'implémenter une initiation des exécutions non dépendante de l'ordre des instructions [Toma67] (politique d'initiation non ordonnée).

Quoiqu'il en soit, les gains en performance amenés par de telles implémentations se révèlent fort dépendants des capacités de calcul du modèle d'exécution.

A l'exception notable du livre de Mike Johnson [John91][3], peu d'études publiées n'ont été menées afin de déterminer la configuration exécutive la plus intéressante du point de vue coût-performance, et plus particulièrement pour des processeur superscalaires à initiation non ordonnée.

En pratique, un certain nombre de choix, que nous rappelons ci-dessous, ont été faits par les constructeurs de microprocesseurs.

1.1. PowerPC 604 [IbMo94]

- 2 UALs entières,
- 1 unité entière de multiplication et de division,
- 1 unité flottante,
- 1 unité mémoire avec un seul accès au cache de données.

1.2. MC 88110 [Moto92]

- 2 UALs entières,
- 1 unité entière de manipulation de champs de bits,
- 1 unité de multiplication entière et flottante,
- 1 unité de division entière et flottante,
- 1 unité arithmétique flottante,
- 1 unité mémoire avec un seul accès au cache de données,
- 2 unités graphiques.

1.3. DEC 21064 [McL93]

- 1 unité entière,
- 1 unité flottante,
- 1 unité mémoire avec un seul accès au cache de données.

Il faut de plus noter qu'à ce jour, seul les gammes Power d'IBM et PowerPC du consortium IBM-Motorola, comprennent des microprocesseurs superscalaires à initiation non ordonnée[4].

L'étude présente va donc porter sur le choix d'une configuration exécutive pour des processeurs superscalaires à initiation non ordonnée et pour un degré d'initiation variable[5]. Une définition de la configuration exécutive d'un tel processeur est l'énumération exhaustive de ces unités d'exécution ou unités fonctionnelles, ainsi que la topologie et le nombre des emplacements de mémorisation d'instructions en attente d'exécution.

[1] BHT ou Branch History Table en américain.

[2] BTB ou Branch Target Buffer en américain.

[3] Le travail présenté dans ce livre se révèle néanmoins un peu faussé par les latences utilisées qui sont celles des MIPS R2000/R3000.

[4] Ceci devrait changer d'ici peu.

[5] Le degré d'initiation renseigne sur le nombre maximum d'instructions qui peuvent être émise à chaque cycle.

Des trois processeurs énumérés précédemment, seul le PowerPC 604 est un processeur à initiation non ordonnée. La topologie est dans son cas, une distribution totale des emplacements de mémorisation sur les unités fonctionnelles avec 2 emplacements par unité.

Nous n'étudierons dans ce rapport seulement l'impact du nombre et du type d'unités fonctionnelles. Les hypothèses qui seront faites permettront de faire abstraction de la topologie.

2. Modèle d'exécution

L'architecture modélisée implémente une politique d'initiation des exécutions non ordonnée ainsi qu'une exécution spéculative afin d'exploiter au mieux le *Parallélisme au niveau des Instructions*[6].

Dans ce type d'architecture, après avoir été *recherchées* et *décodées*, un certain nombre d'instructions sont *émises* depuis le *Tampon d'Emission des Instructions* (TEI) dans un *Tampon d'Initiation des Instructions* (TII) à chaque cycle. La limite maximum de ce nombre d'instructions émises définit le degré d'initiation du processeur. Par conséquent, le processeur ne se retrouve plus bloqué lorsqu'une opérande n'est pas encore calculée : la valeur sera reportée dans le TII lors de la mise à jour du registre correspondant. Les entrées du TII sont similaires aux *Stations de Réservation* (SRs) de l'IBM 360/91 [Toma67], hormis le fait qu'elles peuvent être reliées à plus d'une *Unité Fonctionnelle* (UF). Une entrée du TII maintient l'opération ainsi que les valeurs des opérandes sources, ou à défaut des identificateurs permettant leur récupération. En fait, ces identificateurs renseignent sur les entrées du TII qui produiront les résultats requis.

Les entrées valides qui possèdent les valeurs de toutes leurs opérandes sources sont, chaque cycle, *initiées* dans les diverses unités fonctionnelles présentes, sous réserve qu'une unité ne peut commencer l'exécution que d'une instruction chaque cycle[7]. Lorsqu'une instruction a fini son exécution, le résultat est suivi si nécessaire vers les entrées du TII qui le requièrent.

Les *Techniques d'Initiation* sont les algorithmes qui arbitrent les entrées initiables qui requièrent une même unité fonctionnelle, et donc provoquent un conflit. Le TII peut être *unifié* (toutes les entrées sont reliées à toutes les UFs), *distribué* (toutes les entrées sont spécialisées pour exactement une UF), ou *mixte*. Un mécanisme similaire au TII est appelé la *node table* par Butler et Patt dans [BuPa92]. Dans cet article, ils montrent que hormis une[8], toutes les *techniques d'initiation* donnent des résultats similaires pour un processeur à haut degré d'initiation. Nous avons donc décidé d'adopter le plus naturel, l'algorithme du *plus ancien en premier*, où la priorité est celle de l'ordre d'émission des instructions.

Afin de gérer précisément les interruptions, une entrée associée à chaque instruction émise est enfilée dans un *Tampon de Réordonnancement* (TR) [SmPl85]. Le TR maintient ainsi l'ordre initial du programme, et sa taille définit le nombre maximum d'instructions qui peuvent être en cours de traitement après leur émission. Ce nombre maximum est la taille de la *Fenêtre d'Anticipation* (FA) du processeur. Lorsqu'une instruction est *exécutée*, le résultat est alors aussi enregistré dans l'entrée correspondante du TR, et le banc de registre n'est mis à jour que lorsque l'entrée correspondante dans le TR est à la tête du tampon. L'instruction est alors *terminée*, elle est donc enlevée du TR. Cette terminaison peut concerner plusieurs instructions chaque cycle, et elle est effectuée strictement dans l'ordre du programme (pas de dépassement).

En résumé, une instruction est dans l'ordre : recherchée, décodée, émise, exécutée puis terminée.

2.1. Hypothèses

Afin de définir des configurations exécutives les plus intéressantes, nous faisons abstraction de tout autre paramètre qui pourrait être source d'une dégradation de performance. Nous supposons donc :

- Un mécanisme de recherche d'instruction idéal : pas de défaut de cache d'instruction, pas de mauvaise prédiction, et l'émission n'est limitée que par le degré d'initiation et l'état de remplissage du TR,
- Une indépendance vis à vis de la hiérarchie mémoire, et donc pas de défaut de cache de données,

[6] ILP ou Instruction Level Parallelism en américain.

[7] A l'exception des unités non-pipelines qui n'acceptent aucune autre instruction tant qu'elles sont en cours de traitement.

[8] La *technique d'initiation de tête* où seul l'emplacement le plus anciennement alloué est initiable. Cette politique gère une grande partie des emplacements de mémorisation du PowerPC 604.

- Un TII unifié, afin de ne pas être dépendant de la topologie (il est considéré ainsi idéal). De plus, comme nous l'avons vu précédemment, le choix d'une autre technique d'initiation ne devrait pas modifier nos résultats.

L'énumération précédente des unités implémentées dans divers processeurs superscalaires actuels fait apparaître un certain découpage des instructions suivant leur fonctionnalité : nous les dénommons les *Classes d'Instructions* . Elles sont au nombre de 9 :

- ENTIERE : additions et soustractions entières, et opérations logiques,
- CHAMPS DE BITS : décalages et manipulations de champs de bits,
- MULTIPLICATION ENTIERE,
- DIVISION ENTIERE,
- ACCES AU CACHE DE DONNEES : chargement et enregistrement,
- ARITHMETIQUE FLOTTANTE : additions et soustractions flottantes,
- CONVERSION,
- MULTIPLICATION FLOTTANTE,
- DIVISION FLOTTANTE.

La table suivante présente l'allocation de ces *classes d'instructions* aux UFs des trois processeurs précédemment cités.

Classe d'instructions	PowerPC 604	MC 88110	Dec 21064
entière	2	2	1
champs de bits		1	
multiplication	1	1[9]	
division		1[10]	
arithmétique flottante	1	1	1
conversion			
multiplication flottante		1[1]	
division flottante		1[2]	
accès cache de données	1	1	1

Table 2-1 : Unités Fonctionnelles du PowerPC 604, MC 88110 et Dec 21064

2.2. Modélisation

Nous avons utilisé le logiciel SPA [Irla92] afin de produire les traces d'instructions provenant de l'exécution réelle des *programmes test* explicités à la section 3. Nous n'avons gardé de toutes celles produites, que les informations concernant le code opératoire de l'instruction exécutée ainsi que l'adresse dans le cas d'un accès mémoire. Les traces consistent donc en une liste de paires relatant l'exécution des programmes correspondants.

Ces traces vont être lues par un simulateur que nous avons écrit qui effectue une simulation cycle par cycle et donne comme résultat le *nombre moyen d'Instructions terminées Par Cycle* (IPC).

La FA a été modélisée par une file. L'état de ses entrées peut être : `vide`, `en_attente`, `en_cours` et `exécuté` suivant l'état de traitement de l'instruction correspondante. L'insertion dans la file se fait à partir de la liste des paires séquentiellement, et pour un nombre maximum correspondant au degré d'initiation. Le défilement ne s'effectue que sur des entrées dont l'état est `exécuté`, et pour un nombre maximum toujours égal au degré d'initiation.

[9] Les multiplicateurs entier et flottant forment une seule et même unité.
[10] Les diviseurs entier et flottant forment une seule et même unité.

Concernant les accès mémoire, un chargement peut dépasser (suivant l'ordre d'émission) un enregistrement, si et seulement si l'adresse de l'enregistrement est connue. De plus, un enregistrement ne peut être initié que lorsque toutes les instructions précédemment initiées ont été exécutées, et ceci afin de préserver la cohérence mémoire.

Les latences utilisées pour les unités fonctionnelles sont celles du PowerPC 604 qui sont rappelées dans la table 2-2. Toutes les UFs que nous définirons seront pipelines et mutuellement indépendante, à l'exception de l'unité de division que nous considérerons non-pipeline.

Classe d'Instructions	Latence d'Exécution	Classe d'Instructions	Latence d'Exécution
entière	1	arithmétique flottante	3
champs de bits	1	conversion	1
multiplication	3	multiplication flottante	3
division	20	division flottante	18 SP — 31 DP[11]
chargement/enregistrement	2/3		

Table 2-2 : Latences d'exécution

De plus, nous avons aussi modélisé un processeur scalaire à initiation ordonnée, où seul les dépendances de données sont recherchées[12]. L'hypothèse de base a donc été que toute instruction peut s'exécuter chaque cycle quelque soit l'état du processeur, sous réserve bien sûr que ses opérandes sources soient disponibles. Les résultats sur ce processeur correspondent donc à ce que l'on peut envisager de mieux sur un processeur RISC standard à initiation totalement ordonnée (sans prédiction de branchement). Les accélérations qui seront données par la suite seront calculées par rapport à ce processeur.

3. Les programmes test

3.1. Standard Performance Evaluation Corporation [SPEC92]

L'ensemble des résultats présentés dans ce rapport proviennent d'évaluations faîtes sur le dernier paquetage de programmes test fourni par SPEC. Ce paquetage est en fait décomposé en deux ensembles suivant les données que les programmes manipulent : CINT 92 (entier) et CFP 92 (flottant).

Pour toutes ces évaluations, nous avons utilisé la totalité des CINT 92 (6 programmes) et des CFP 92 (14 programmes). Une brève description de ceux-ci est donnée dans les tables 3-1 et 3-2. Les programmes ont été compilés sur une SPARCSTATION LX en utilisant les *makefiles* standards fournis avec le paquetage (toutes les optimisations sont activées).

3.1.1. CINT 92

`008.espresso`	Génère et optimise des PALs.
`022.li`	Utilise un interpréteur LISP pour résoudre le problème des 9 reines.
`023.eqntott`	Traduit une représentation logique d'une équation Booléenne en une table de vérité.
`026.compress`	Réduit la taille du fichier d'entrée par codage de Lempel-Ziv.
`072.sc`	Calcule des budgets, des mesures SPEC et des perspectives d'amortissement dans un tableur basé sur le paquetage UNIX orienté curseur *CURSES*.

[11] SP précise des calculs en Simple Précision et DP en Double Précision.

[12] Il est à noter que les dépendances de contrôle influent aussi puisque l'initiation est ordonnée : un branchement conditionnel dont la condition n'est pas calculée impliquera l'insertion de bulles dans le pipeline (il n'y a pas de prédiction).

085.gcc	Traduit des fichiers source C en des fichiers assembleur Sun-3 optimisés.

Table 3-1 : Programmes test utilisés provenant de l'ensemble CINT 92.

3.1.2. CFP 92

013.spice2g6	Simulation de circuits analogiques.
015.doduc	Effectue une simulation de Monte-Carlo de l'évolution du temps d'un modèle thermo-hydraulique pour un composant d'un réacteur nucléaire.
034.mdljdp2	Résolution d'équations de mouvement pour un modèle de 500 atomes en intéraction en utilisant le potentiel idéal de Lennard-Jones (double précision).
039.wave5	Résolution d'équation de Maxwell pour des particules sur une grille cartésienne.
047.tomcatv	Génère un système de coordonnées 2-dimensions maillé autour d'un domaine géométrique général.
048.ora	Lancer de rayon à travers une surface optique contenant des surfaces sphériques et planes.
052.alvinn	Exerce un réseau de neurones par rétropropagation.
056.ear	Simulation de l'oreille humaine par conversion d'un fichier de son en cochléogramme en utilisant des FFTs et d'autres fonctions mathématiques.
077.mdljsp2	Similaire à 034.mdljdp2, mais calcul en simple précision.
078.swm256	Résout les équations des écoulements superficiels par approximation des différences finies.
089.su2cor	Calcul des masses élémentaires de particules dans le cadre de la théorie de Quark Gluon.
090.hydro2d	Utilisation de l'équation hydrodynamique de Navier-Stockes afin de calculer les flux galactiques.
093.nasa7	Exécute 7 noyaux d'opérations effectuées fréquemment dans les applications de la NASA, comme la transformée de Fourrier et des manipulations de matrices.
094.fpppp	Calcul de dérivées intégrales de plusieurs électrons.

Table 3-2 : Programmes test utilisés provenant de l'ensemble CFP 92.

3.2. Traces

Comme mentionné précédemment, le logiciel SPA [Irla92] a été utilisé pour produire des traces d'instructions. Ces traces regroupent pour chaque programme test du CINT 92 les 10 premiers millions d'instructions. En raison d'initialisations ne traitant aucunement de données flottantes, nous n'avons pas tracé les 10 premiers millions d'instructions des programmes test du CFP 92 mais les 10 suivants. Enfin, les programmes test *alvinn* et *spice2g6* nécessitant presque 250 millions et 530 millions d'instructions respectivement pour initialiser leurs données, nous ne les avons tracés qu'après ces valeurs respectives.

3.3. Distributions

Les traces que nous avons utilisées possèdent les distributions suivantes, le premier chiffre concernant la moyenne sur les CINT 92 et le second sur les CFP 92.

- branchement : 18,66 % 6,86 %
- accès mémoire : 25,01 % 31,27 %
  - chargement : 18,36 % 22,87 %
  - enregistrement : 6,65 % 8,4 %
  - atomique : 0 % 0 %
- entier : 52,22 % 34,43 %
  - UAL : 48,27 % 30,71 %
    - arithmétique : 33,15 % 22,45 %
    - logique : 15,12 % 8,26 %
  - décalage : 3,86 % 3,28 %
  - multi-cycle : 0,09 % 0,44 %
- flottant : 0 % 24,43 %
  - conversion : 0 % 3,6 %
  - arithmétique : 0 % 12,22 %
  - multiplication : 0 % 8,08 %
  - division : 0 % 0,53 %
- autres : 4,11 % 3,01 %

3.4. Calcul de la moyenne

La manière de calculer la moyenne est de la plus grande importance : elle peut fausser les résultats. En effet, la moyenne des résultats doit être faite impérativement sur des temps d'exécution.

Un exemple est le temps d'exécution de programmes. Ainsi, supposons un programme P1 qui s'exécute en 3 secondes et un autre programme P2 qui requiert 5 secondes. La moyenne de l'exécution des deux programmes est 4 secondes, ce qui est juste car nous travaillons sur des temps (les secondes). A l'inverse, on peut dire que par seconde, on exécute $\frac{1}{3}$ de P1 et $\frac{1}{4}$ de P2. Si on effectue la moyenne sur ces valeurs, on obtient que par seconde, on exécute $\frac{7}{24}$ d'un programme, ce qui est égal si on passe à l'inverse à $\frac{24}{7}$ =3,4 secondes par programme. En fait, cette moyenne est fausse car s^{-1} n'est pas une unité de temps.

Dans le cas qui nous intéresse, l'unité de temps est le cycle puisqu'il est constant. Ainsi, puisque tous les programmes test ont le même nombre d'instructions (10.000.000 d'instructions chacun), l'unité CPI (nombre de cycles moyens nécessaire pour exécuter une instruction) est une unité de temps. L'IPC (inverse du CPI) ne l'est donc pas. Par contre, puisque tous les résultats seront donnés en IPC, il est nécessaire d'introduire une moyenne sur ces valeurs qui sera juste. On a pour N programmes test (CPI_n étant le CPI obtenu lors de l'exécution du programme test numéroté n, N étant le nombre programmes test) :

$$Moy_{CPI} = \frac{1}{N} \times \sum_{0 \le n < N} CPI_n$$

Or $CPI_n = \frac{1}{IPC_n}$

Donc $Moy_{IPC} = \frac{1}{Moy_{CPI}} = \frac{N}{\sum_{0 \le n < N} \frac{1}{IPC_n}}$

La moyenne que l'on vient de définir est appelée la *moyenne harmonique*, qui est à opposer à la moyenne arithmétique. Il est important de noter la relation qui existe entre ces deux moyennes :

$$Moy_{unité}^{harmonique} = \frac{1}{Moy_{unité^{-1}}^{arithmétique}}$$

De plus, ces moyennes ne doivent se calculer que sur des unités. En effet, une erreur serait de calculer des moyennes arithmétiques sur des accélérations (rapport entre 2 temps, donc sans unité). Ceci est illustré par l'exemple suivant :

Supposons les résultats suivants sur 2 configurations pour 2 programmes différents (chaque programme faisant 120 instructions) :

Programme	IPC (configuration 1)	IPC (configuration 2)	Accélération
1	2 soit 60 cycles	6 soit 20 cycles	3
2	0,4 soit 300 cycles	2 soit 60 cycles	5

La moyenne harmonique des accélérations est de 3,75. La moyenne arithmétique donne, elle, une accélération de 4.

Ces deux résultats sont faux. En effet, pour le programme 1 on sauve 40 cycles en passant de la configuration 1 à la configuration 2. Pour ce qui concerne le programme 2, on sauve 240 cycles : 6 fois plus ! L'accélération moyenne doit donc être beaucoup plus proche de 5 que de 3.

Pour calculer l'accélération moyenne, il est en fait nécessaire de faire la moyenne harmonique sur l'IPC (ou arithmétique sur le CPI) pour chaque configuration, puis de faire le rapport, et non de faire la moyenne sur les rapports. La vraie accélération moyenne est ici de 4,5 :

$$\text{Accélération}_{\text{moy}} = \frac{\frac{2}{\frac{1}{2}+\frac{1}{6}}}{\frac{2}{\frac{1}{0,4}+\frac{1}{2}}} = \frac{\frac{3}{2}}{\frac{2}{3}} = 4,5$$

4. Fenêtre d'Anticipation

4.1. Définition

La Fenêtre d'Anticipation (FA) est en fait une représentation abstraite de l'ensemble des instructions en cours de traitement. Une instruction entre dans cette FA à partir du moment où elle est émise. Elle en sort lorsqu'elle est terminée. Une implémentation matérielle possible est le couplage d'un tampon de réordonnancement et de stations de réservation. Dans ce cas, la FA est exactement l'abstraction du tampon.

La taille de cette FA joue donc un rôle primordial quant à la performance du microprocesseur. En effet, elle contiendra d'autant plus d'instructions exécutables en parallèle que sa

taille est grande. Bien entendu, ceci implique un coût matériel non négligeable, et donc il s'agit de la minimiser.

Cette section présente donc l'étude de l'impact de la taille de la FA sur l'IPC d'un microprocesseur superscalaire à initiation non ordonnée. Cette étude se fait avec les hypothèses de modélisation précédemment citées à la section 2. De plus, afin de ne pas falsifier les résultats par une configuration d'unités fonctionnelles arbitraire, on suppose une configuration infinie. Enfin, cette étude est faite sur des *programmes test* portant exclusivement sur des opérations entières (SPEC INT 92), puis sur d'autres portant principalement sur des opérations flottantes (SPEC FP 92) comme mentionné à la section 3.

Il est à noter que l'étude concerne des processeurs à degré d'initiation variable : 2, 4, 6, 8, 10 et infini. En ce qui concerne ce dernier cas, le processeur est modélisé comme ayant un degré d'initiation de 256, ce qui revient à dire que son degré est infini puisque la taille maximale de FA envisagée est de 256.

4.2. Simulation (SPEC Int 92)

4.2.1. Résultats

Les figures suivantes représentent les résultats de simulation obtenus pour chaque programme décrit chapitre 3.

Figure 4-1 : Compress

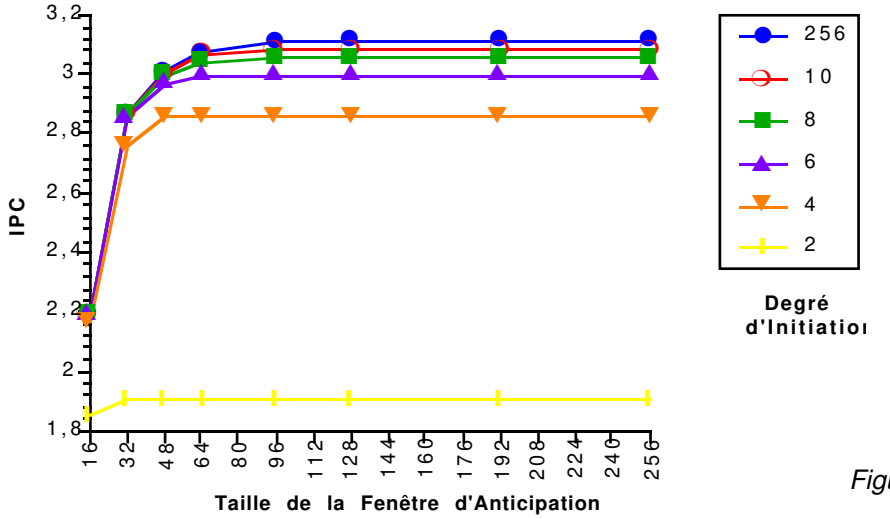
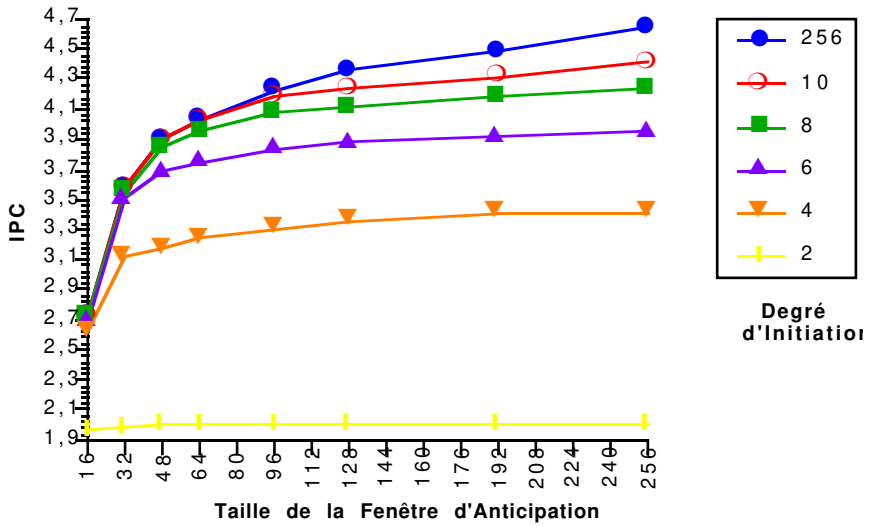


Figure 4-2 : Eqntott



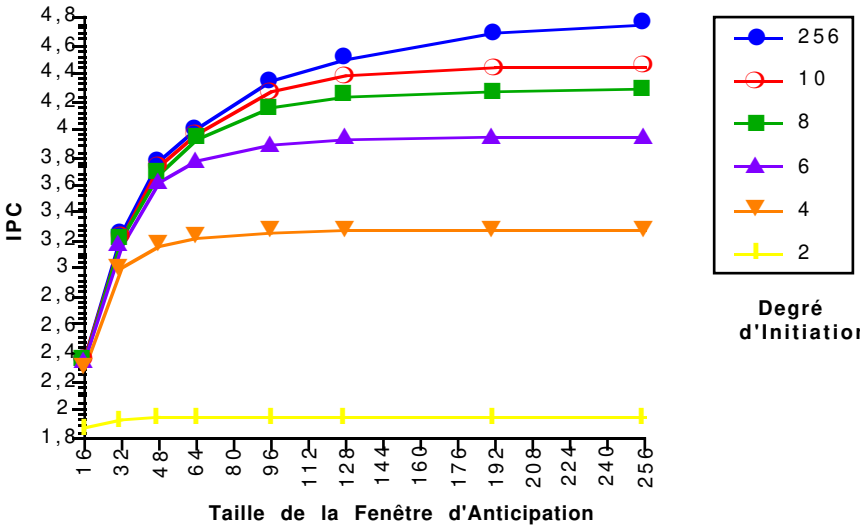
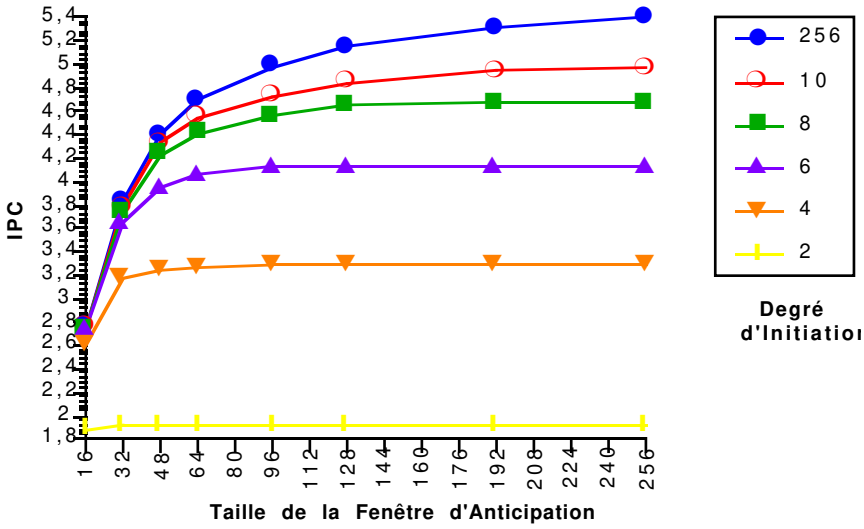


Figure 4-3 : Espresso

Figure 4-4 : Gcc



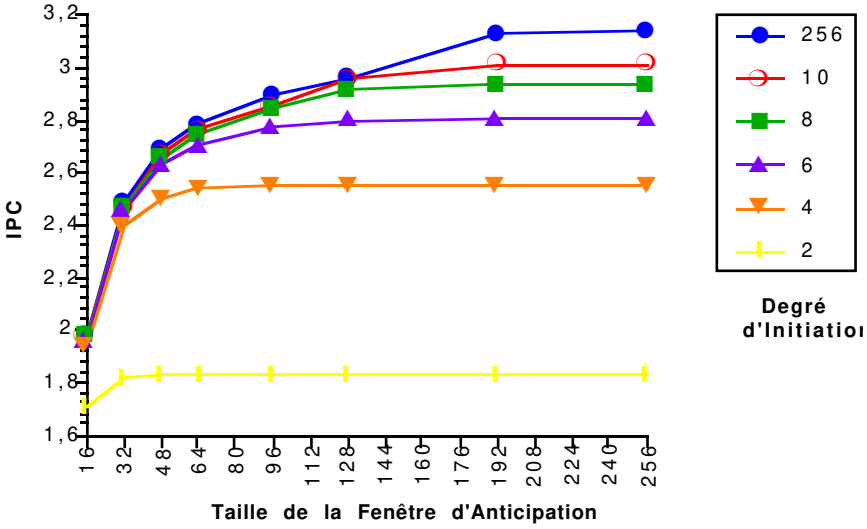


Figure 4-5 : Li

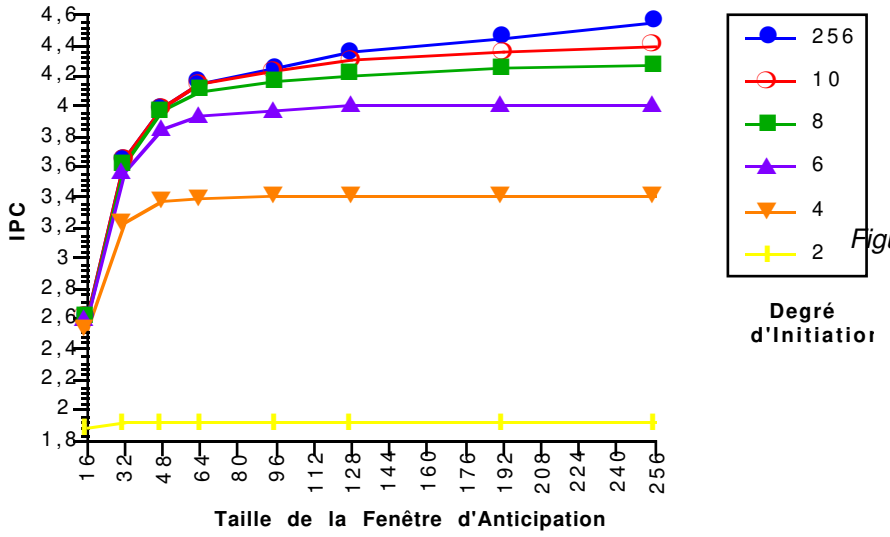


Figure 4-6 : Sc

4.2.2. Moyenne

La courbe moyenne de ces programmes-test est montrée sur la figure 4-7.

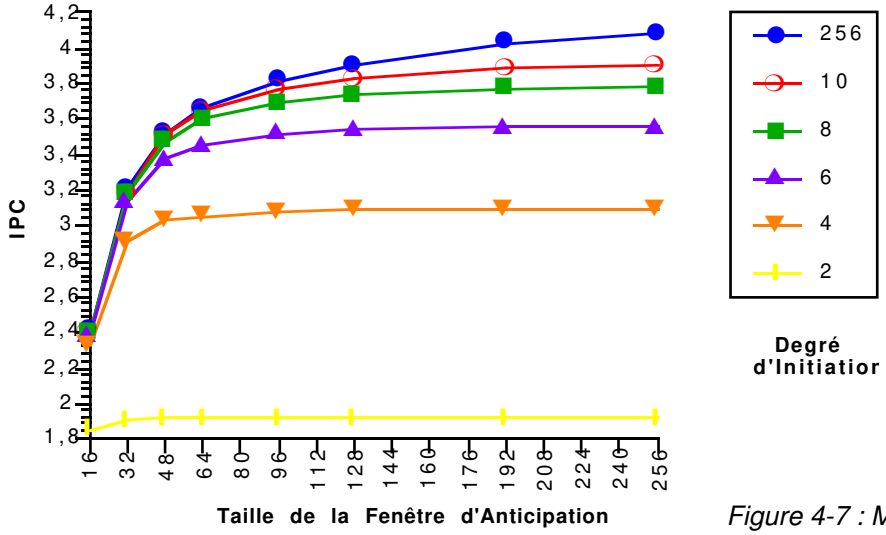


Figure 4-7 : Moyenne des SPECINT92

4.2.3. Accélération

Les accélérations par rapport au processeur scalaire spécifié section 2, sont données sur les figures 4-8 à 4-11 pour des degrés d'initiation 2, 4, 6 et 8 respectivement. En effet, on remarque bien qu'un processeur de degré 8 suffit à prendre en compte le parallélisme du programme du moins pour des proportions réalistes de la FA. On peut néanmoins s'interroger sur la faisabilité d'un processeur à plus haut degré d'initiation. Il serait alors nécessaire d'implémenter une FA très grande (dont la taille serait supérieure à 256), ce qui poserait la question de la validité des instructions en queue de FA, puisqu'elles seraient dépendantes d'un grand nombre de branchements prédits. On étudiera donc par la suite seulement des processeurs de degré 2, 4, 6 et 8.

4.2.4. Conclusions

L'ensemble de ces courbes met principalement deux points en évidence.

Premièrement, même si les densités d'intégration le permettaient, l'implémentation d'une FA de plus en plus importante n'engendrerait plus de gain de performance (les courbes sont asymptotiques). Bien sûr, ceci est du au fait qu'on limite le degré d'initiation. Par exemple, pour un degré de 2, on ne pourra que s'approcher de cette valeur en ce qui concerne l'IPC. Par contre, pour les autres degrés d'initiation, l'IPC limite est en dessous de la valeur escomptée. Puisque seules les dépendances de données sont modélisées, cette limitation est leur conséquence. Agrandir la FA n'amènerait donc aucun gain : les instructions supplémentaires seraient pour la plupart dépendantes au moins d'une autre précédemment présente dans la FA.

Enfin, toujours en ce qui concerne la FA, les courbes ont été données pour des valeurs allant jusqu'à 256. Actuellement, le PowerPC 604 par exemple implémente une FA dont la taille est 16. Des valeurs supérieures à 96, voire 64 sont par conséquent irréalistes de nos jours, mais sont données pour étude. On peut de toute façon remarquer que, comme mentionné précédemment, elles n'apportent pas de gain notable.

La table ci-dessous résume ainsi les configurations qui nous semblent offrir les meilleurs rapports coût-performance. La dégradation de performance par rapport à une taille de FA de 256, pour chaque degré, est inférieure à 5 %.

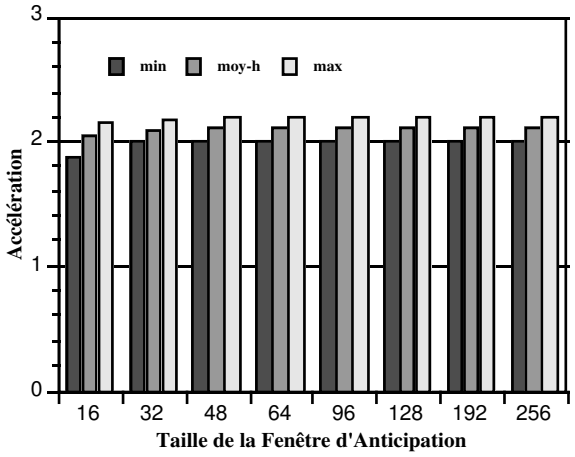


Figure 4-8 : degré 2

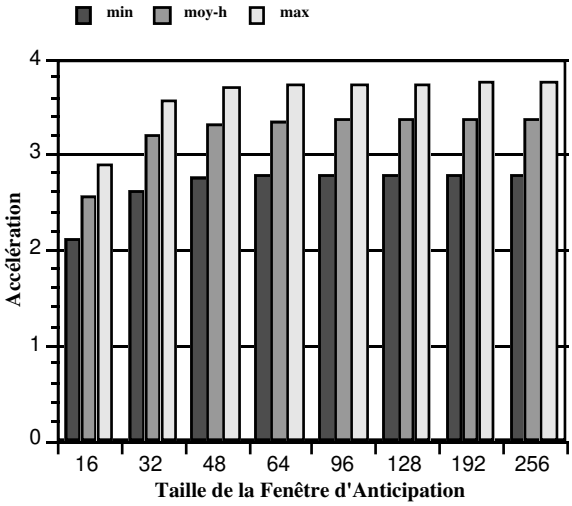


Figure 4-9 : degré 4

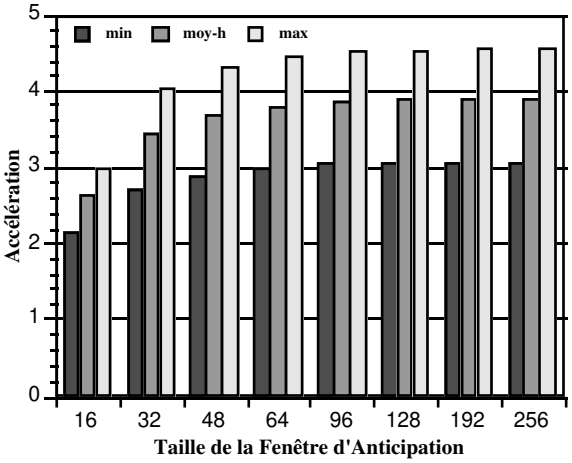


Figure 4-10 : degré 6

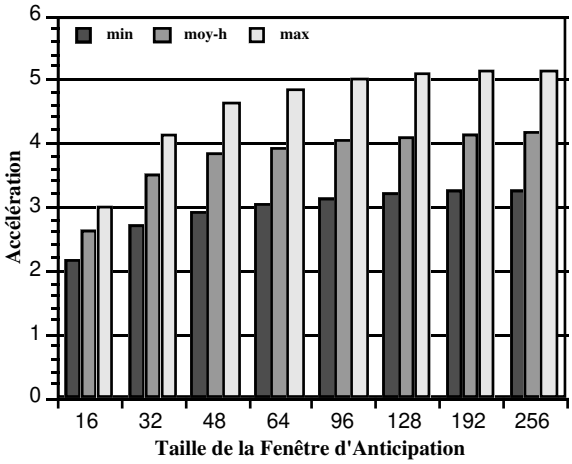


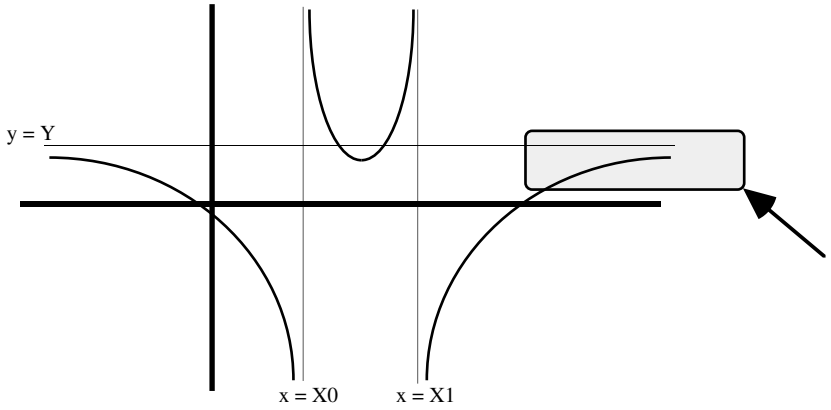
Figure 4-11 : degré 8

Degré	2	4	6	8
Taille de FA	16	32	48	64

Accélération	2,03	3,2	3,7	3,94

4.3. Modélisation mathématique

Une approximation des courbes moyennes par degré d'initiation est donnée ci-dessous. Une approximation linéaire donnant une trop grande erreur, l'approximation a été faite suivant l'inverse d'un polynôme de degré 2. Ceci introduit néanmoins deux zéro du polynôme (voir figure ci-dessous), impliquant un choix des coefficients tel que les fonctions soient définies pour une FA supérieure ou égale à 16.



Soit Y la valeur de l'asymptote horizontale, T la taille de la FA, la formule générale est :

$$\begin{cases} IPC(T) = Y\left(1 - \dfrac{1}{aT^2 + bT + 1}\right) \\ 16 \le T \end{cases}$$

Le tableau suivant donne les coefficients pour chacune des courbes, l'erreur commise par l'approximation étant inférieure à 0,6 %.

Degré d'initiation	2	4	6	8
a	0,3674	0,0191	0,0068	0,0034
b	-4,0213	-0,1219	0,0188	0,0528

4.4. Simulations (SPEC FP 92)

Les figures suivantes présentent les résultats de simulation obtenus pour chaque programme manipulant des données flottantes décrit en section 3.

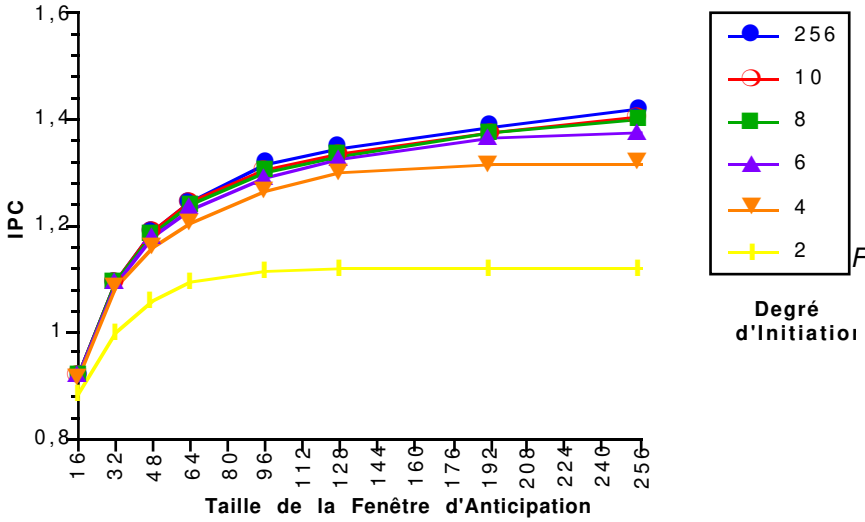


Figure 4-14 : Fpppp

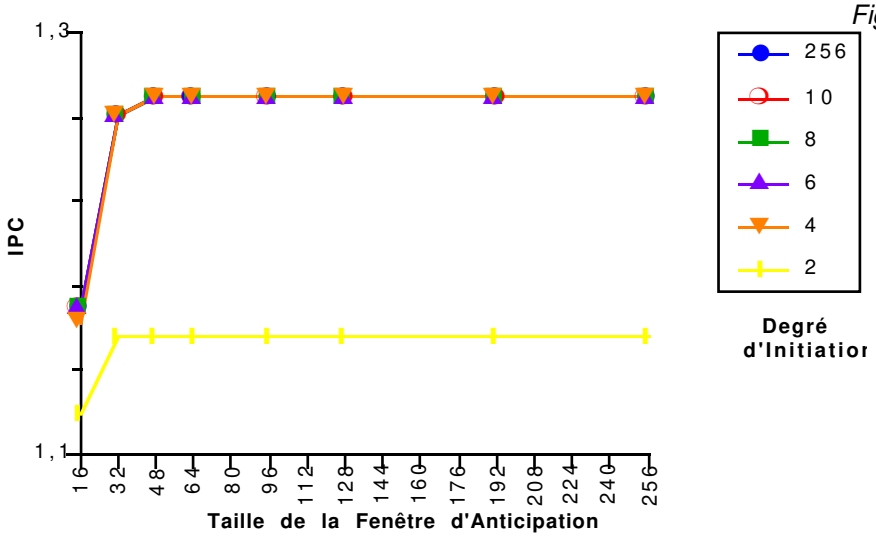


Figure 4-12 : Doduc

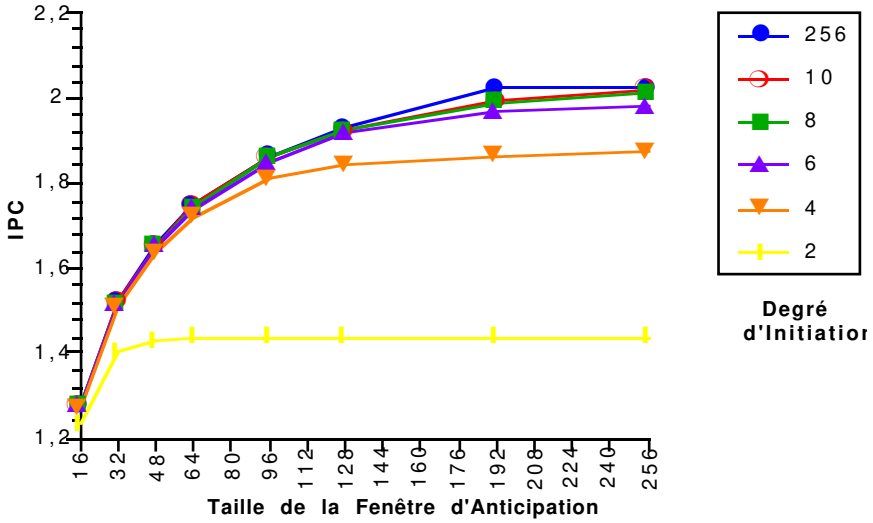


Figure 4-13 : Swm 256

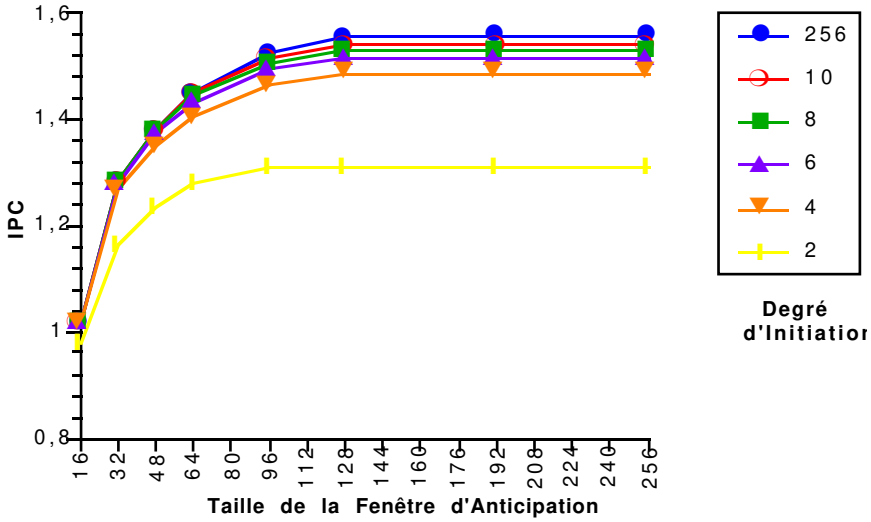


Figure 4-17 : Nasa 7

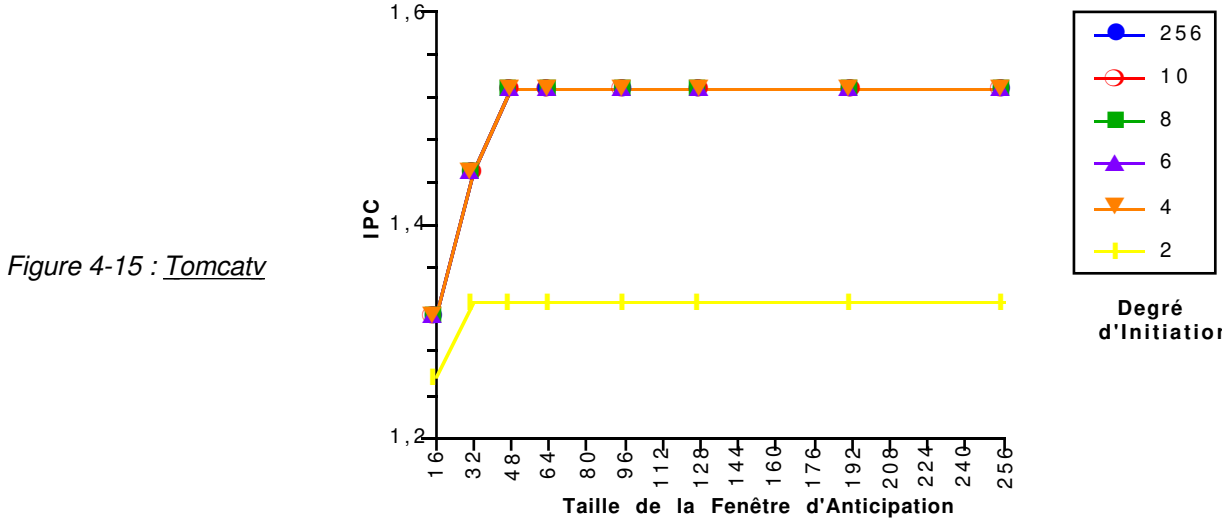


Figure 4-15 : Tomcatv

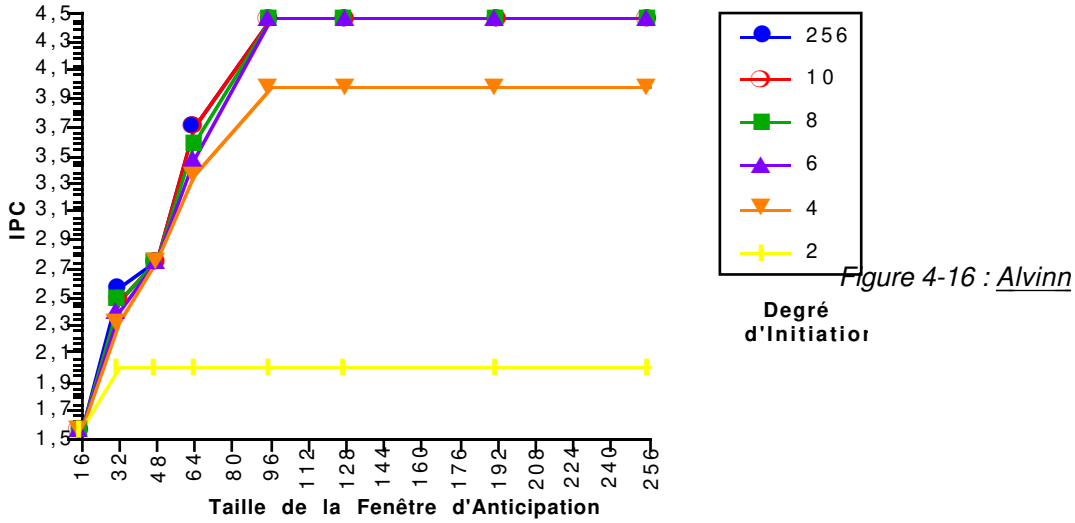


Figure 4-16 : Alvinn

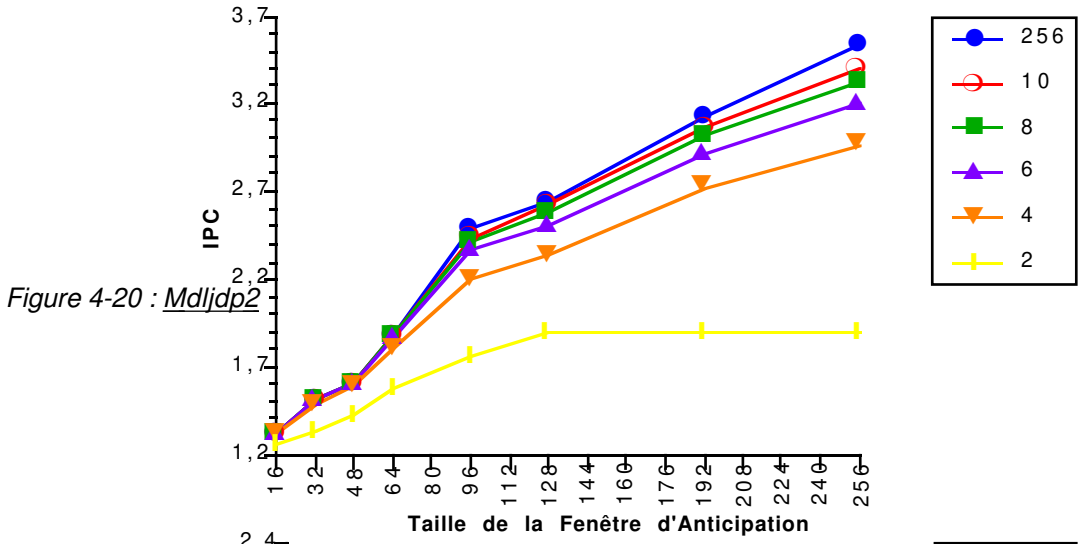


Figure 4-20 : Mdljdp2

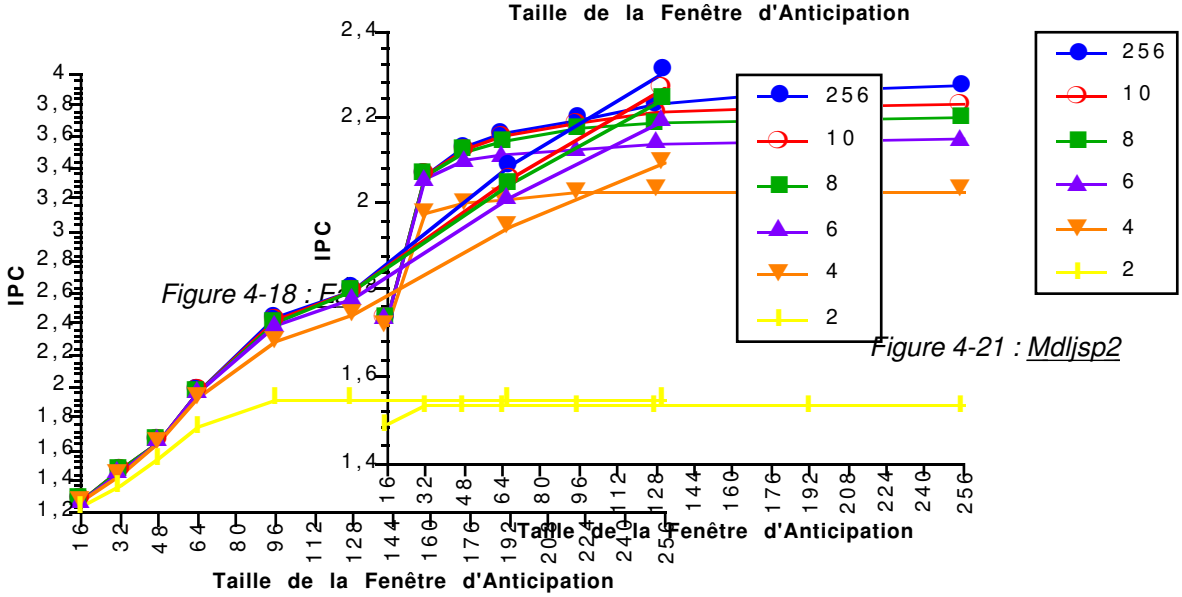


Figure 4-18 : Fp

Figure 4-21 : Mdljsp2

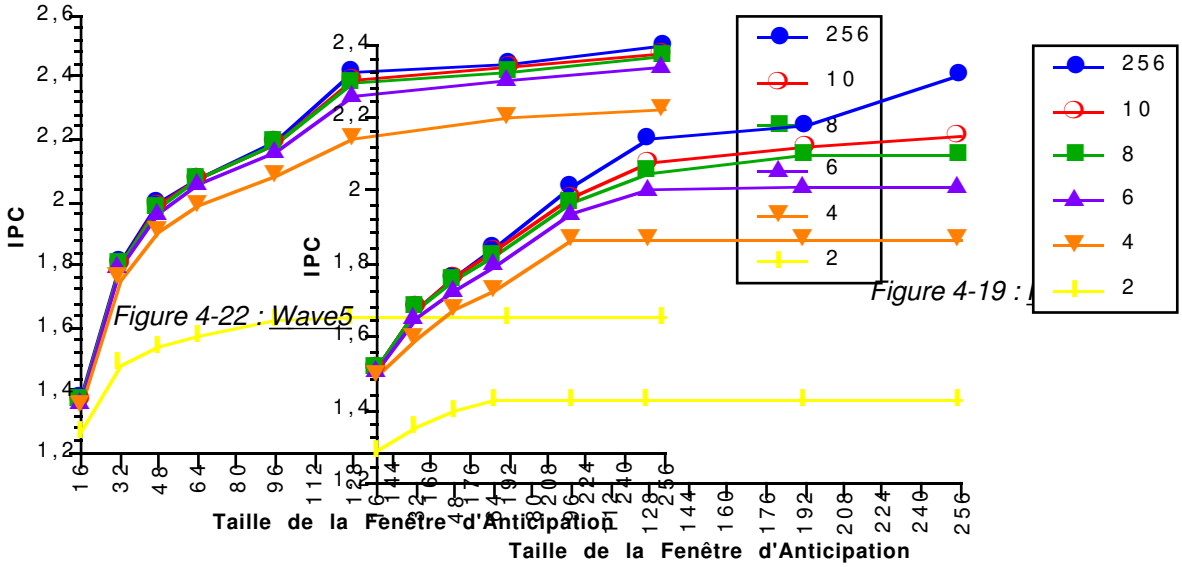


Figure 4-22 : Wave5

Figure 4-19 : I

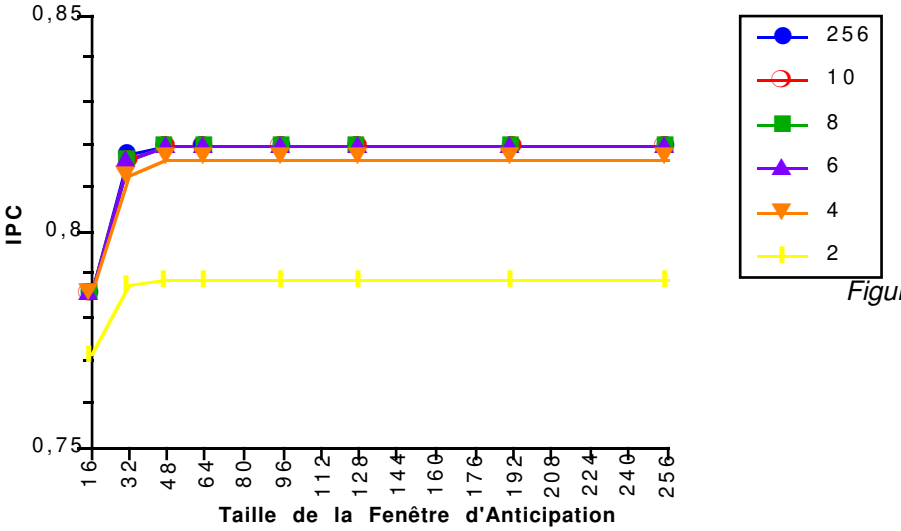


Figure 4-23 : Ora

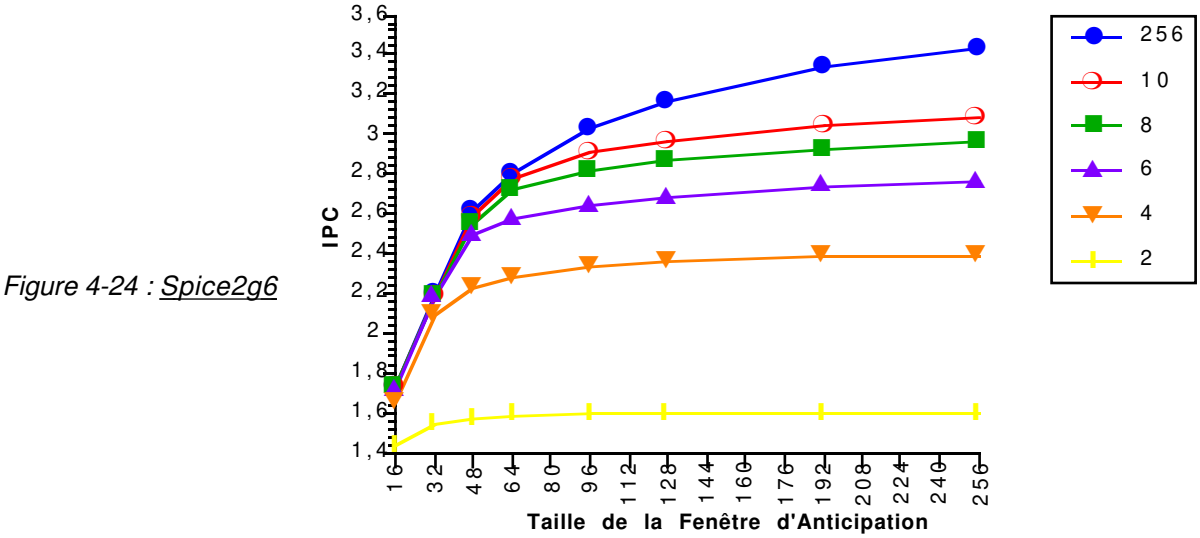


Figure 4-24 : Spice2g6

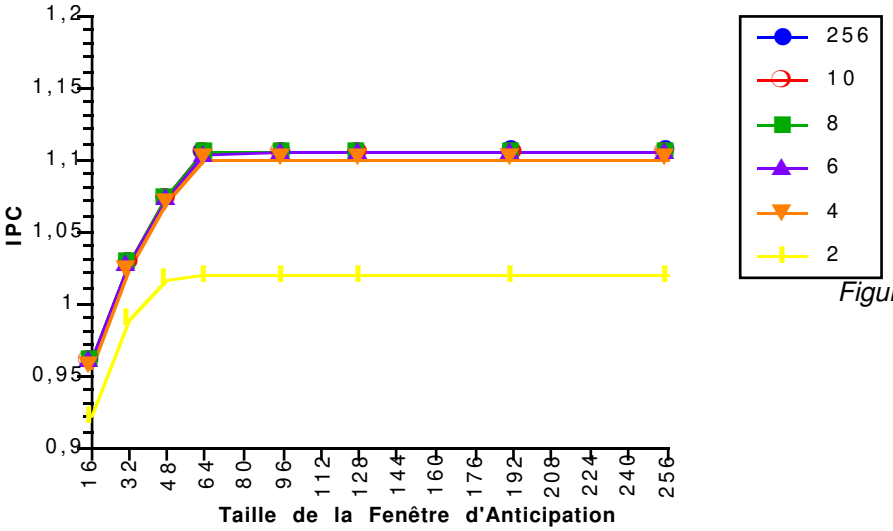


Figure 4-25 : Su2cor

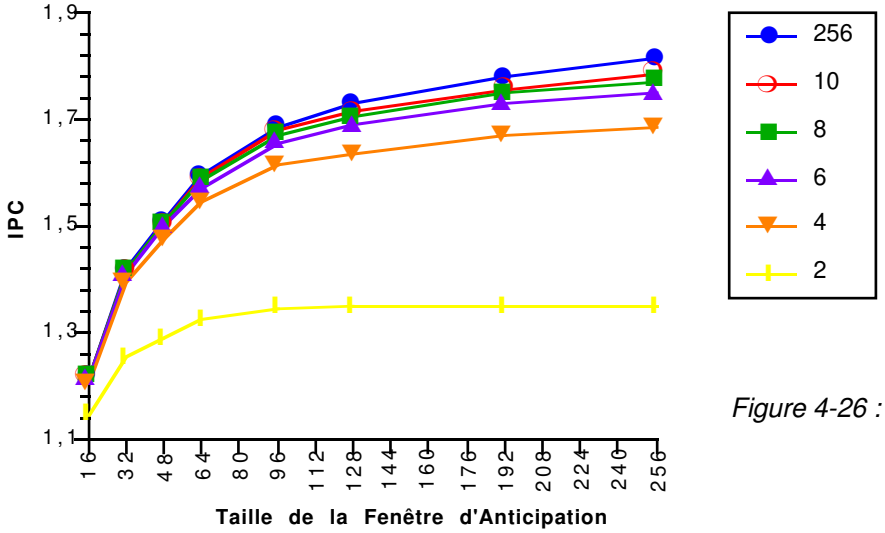


Figure 4-26 : Moyenne des SPECFP92

4.4.1. Moyenne et accélérations

La figure 4-26 présente la moyenne des résultats des 14 programmes test. Les figures 4-27 à 4-30 montrent les accélérations toujours par rapport au processeur scalaire à initiation ordonnée de la section 2, pour des degrés 2, 4, 6 et 8 respectivement.

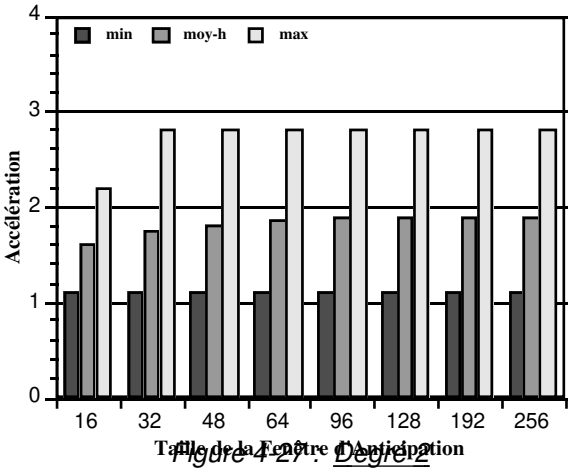


Figure 4-27 : Degré 2

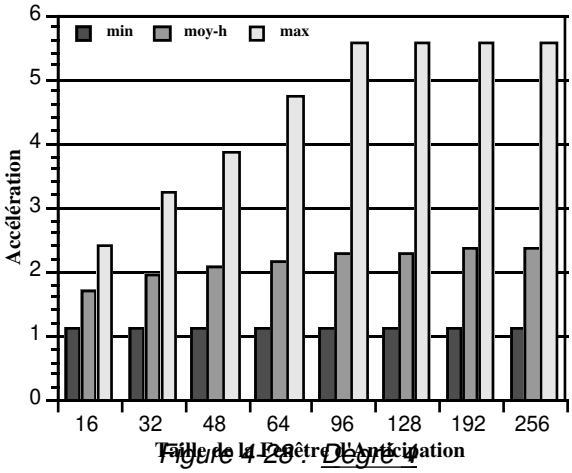


Figure 4-28 : Degré 4

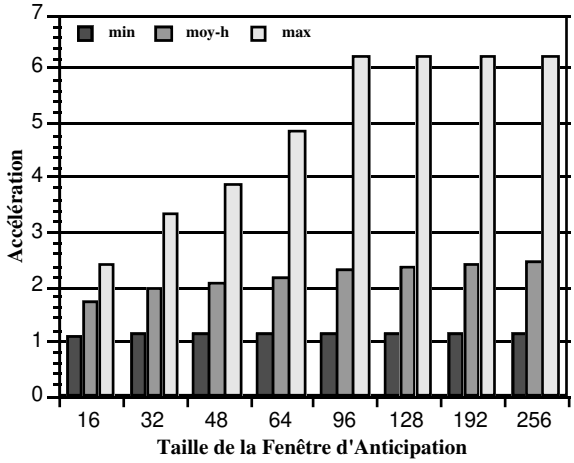


Figure 4-29 : Degré 6

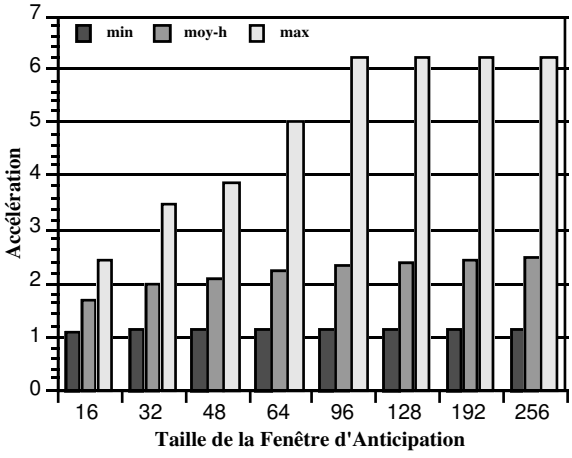


Figure 4-30 : Degré 8

4.4.2. Conclusion

Tout d'abord, on peut remarquer que les résultats sur les programmes test sont très disparates tant au niveau de l'IPC que du comportement vis à vis de la taille de la FA. Néanmoins, la moyenne montre un comportement clair qui n'est pas par contre asymptotique pour les valeurs de FA étudiées : l'IPC continue à augmenter sensiblement (hormis pour les degrés 2 et 4 où le maximum est atteint). De toute façon, il resort de ces courbes qu'une FA de grande taille est nécessaire pour extraire un niveau maximal de parallélisme (pour un degré 2, il est nécessaire d'avoir une FA de 64 instructions). Ceci vient du fait que les programmes flottants impliquent plus de dépendance que leurs homologues entiers (l'IPC obtenu est plus faible), et donc nécessite une plus grande anticipation. Cette anticipation cherche en fait à recouvrir plusieurs traitements parallélisables comme les itérations successives et indépendantes d'une boucle. Il est à remarquer que l'augmentation de l'IPC pour des FAs très grande serait plus importante dans le cas d'une compilation vraiment efficace.

Il apparaît donc que la taille de la FA ne peut être déterminée qu'en fonction des programmes flottants car ils impliquent une taille irréaliste. Il nous semble donc plus viable de garder les valeurs obtenues par les simulations sur les CINT 92, sachant que le processeur ne traitera qu'une partie du parallélisme qui lui est offert dans les programmes flottants. La perte en performance obtenue avec de telles valeurs vis à vis de celles obtenues avec une taille de FA de 256, ainsi que l'accélération, sont données dans la table ci-dessous. Ces résultats constituent de toute façon des résultats plus qu'honorables spécialement au vu de l'économie réalisée sur la taille de la FA.

Degré	2	4	6	8
Taille de la FA	16	32	48	64
Dégradation	15,3 %	17,4 %	14,3 %	10,5 %
Accélération	1,6	1,95	2,1	2,22

5. Opération Entière et Accès Mémoire

5.1. Opération Entière non multi-cycle et Accès mémoire

Dans la plupart des programmes, les instructions les plus usitées sont les instructions entières non multi-cycle et les accès à la mémoire (voir distribution à la section 3). Afin d'évaluer uniquement le besoin en unités traitant de telles instructions, une première étape consiste à ne pas limiter l'initiation des autres classes d'instructions. Par conséquent, pour les instructions autres que les instructions entières non multi-cycle et les instructions d'accès à la mémoire, toutes les instructions initiables présentes dans la fenêtre d'anticipation seront initiées.

Deux types d'unités sont donc définis pour l'instant :

- les unités entières qui traitent toute instruction entière non multi-cycle,
- l'unité mémoire qui est unique mais qui peut avoir des accès multiple à la hiérarchie mémoire.

Les accès multiples au cache de données peuvent être implémentés de diverses façons. La plus simple consiste à utiliser des mémoires dont la fréquence est au moins le double de celle du processeur. Un exemple de ce type d'implémentation est le Power 2 [SmWe94]. On peut également implémenter un cache à structure entrelacée permettant des accès simultanés à des bancs différents en supposant qu'il n'y a pas de conflit. Cette dernière solution est implémentée dans le Pentium d'Intel [Inte93] et le SGI TFP [Hsu94]. Comme les caches ne sont pas modélisés, nous ne nous attarderons pas à une structure de cache spécifique. Ainsi, la seule restriction d'initiation sur les accès mémoire est ce nombre de ports.

Les figures 5-1 à 5-6 montrent la performance obtenue en fonction du nombre d'unités entières de ports mémoire sur les programmes entiers, étant bien entendu que l'impact de ces restrictions est plus faible sur les programmes test flottants.

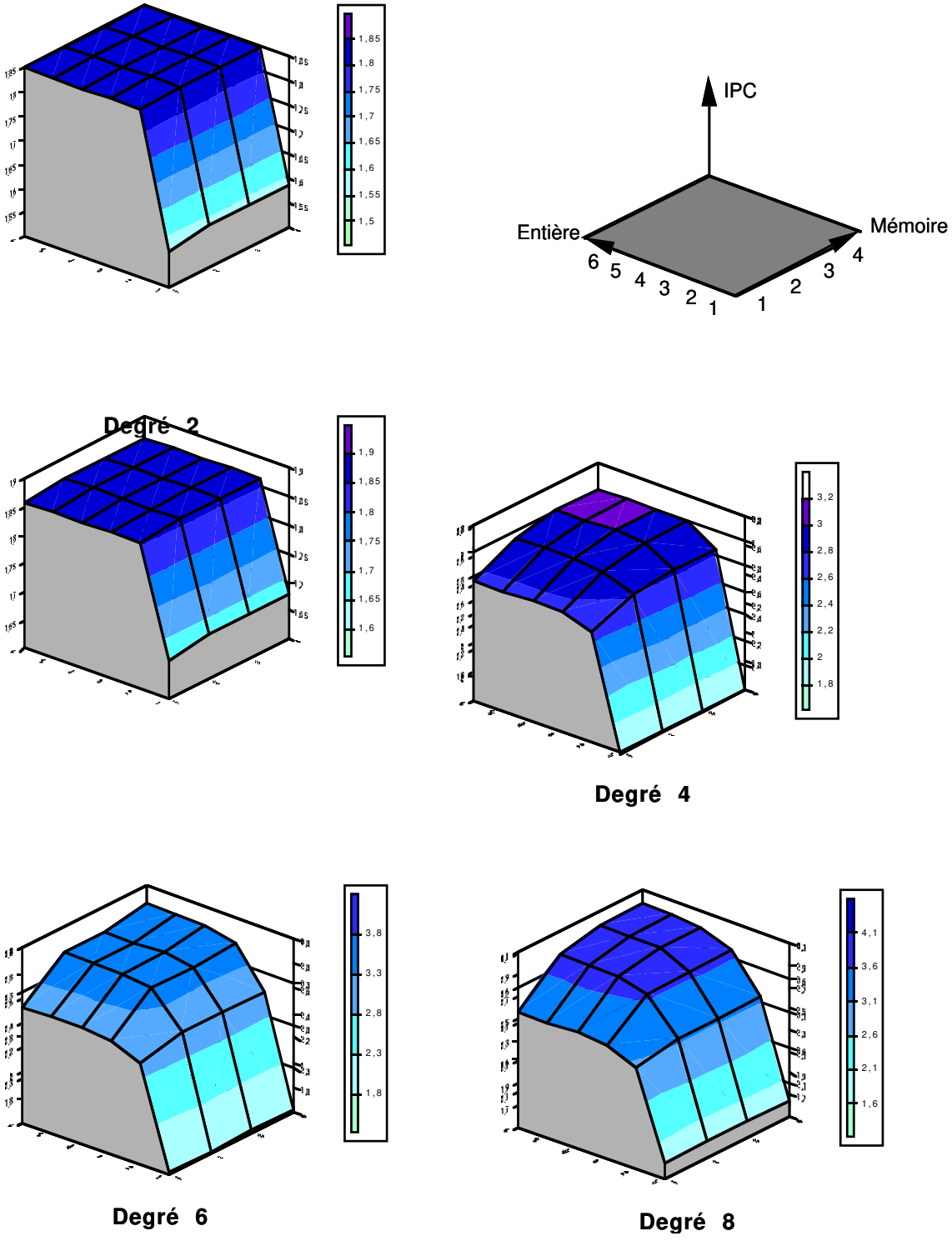


Figure 5-12: Compress

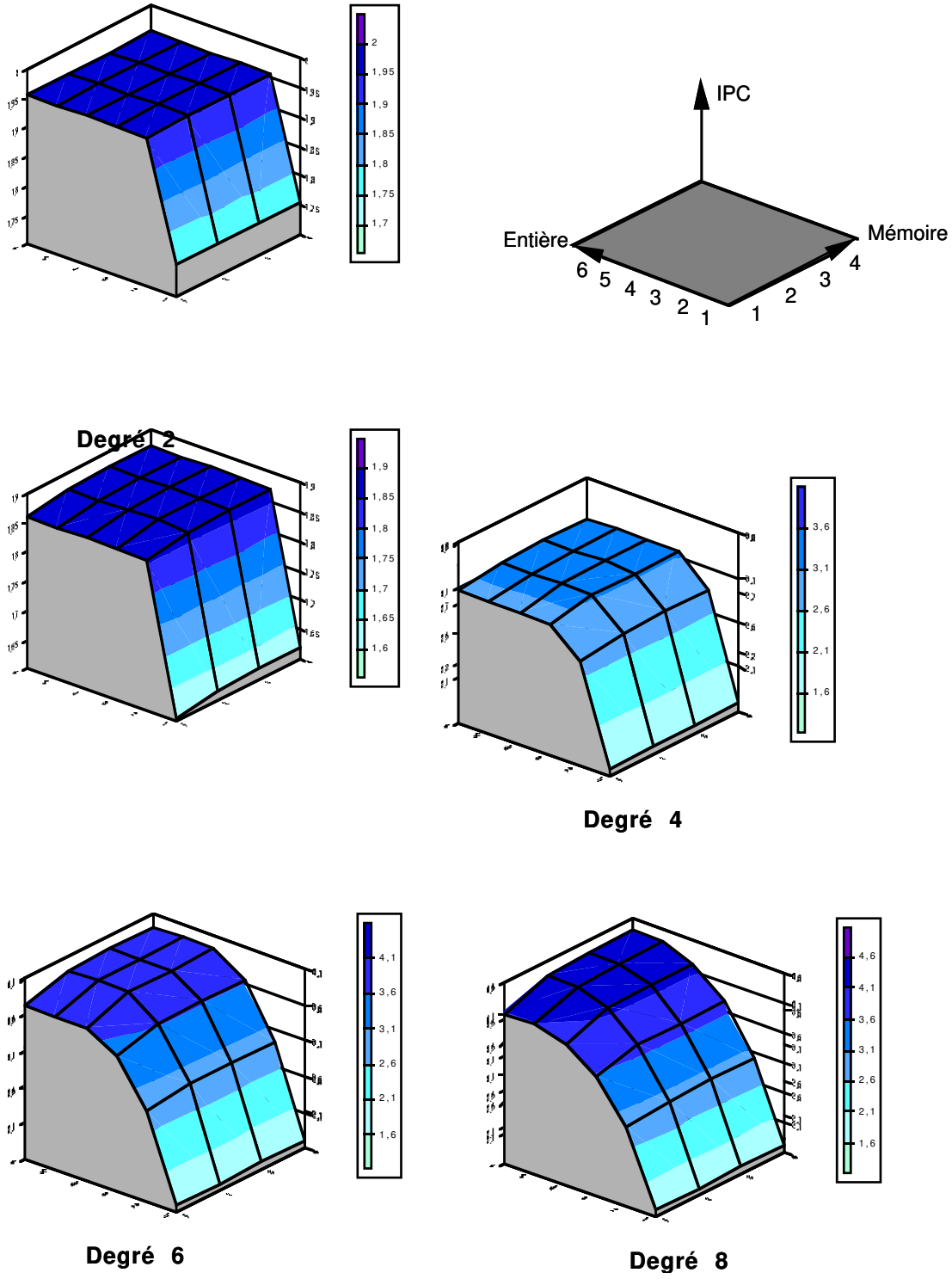
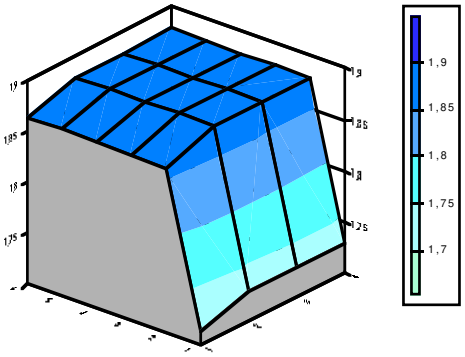
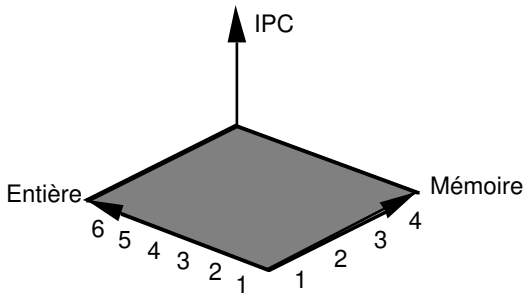
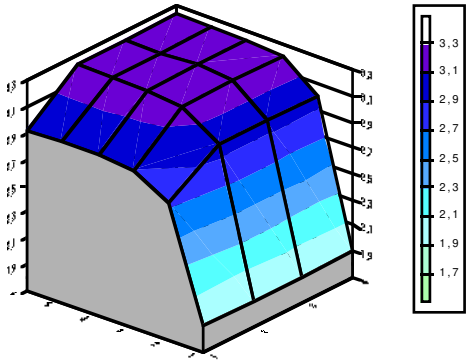


Figure 5-3: Egalité

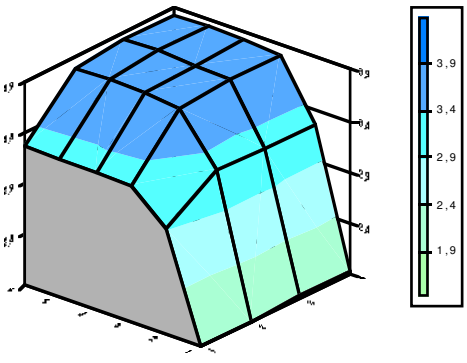




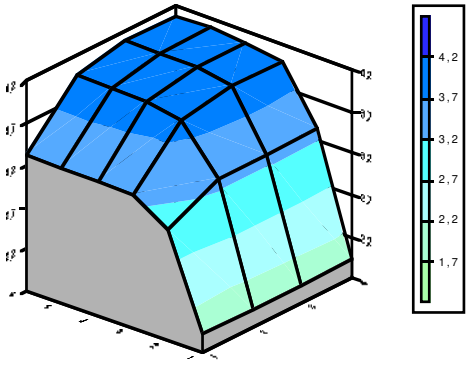
Degré 2



Degré 4

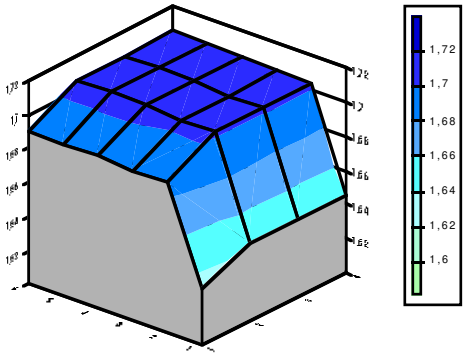


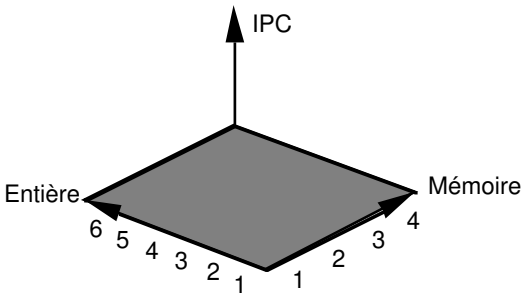
Degré 6



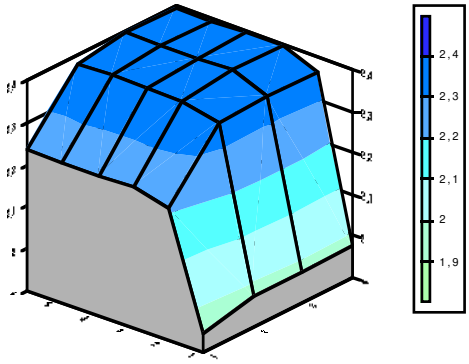
Degré 8

Figure 5-5: Sc

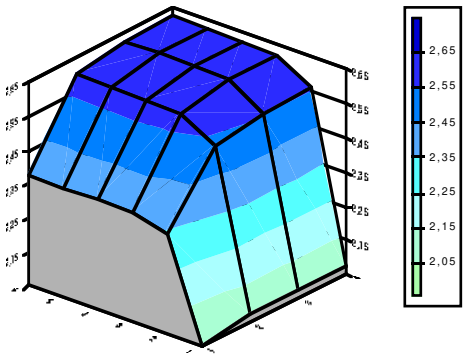




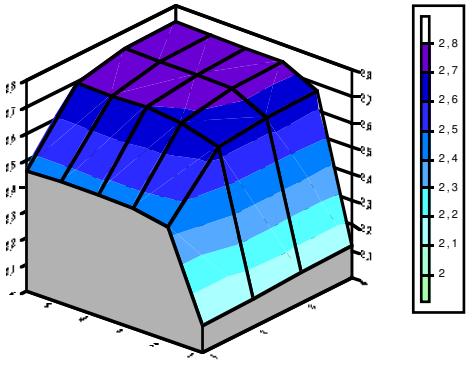
Degré 2



Degré 4



Degré 6



Degré 8

Figure 5-6: Li

5.1.1. Moyennes et conclusions

Les figures 5-7 à 5-10 donnent les moyennes pour chaque degré d'initiation.

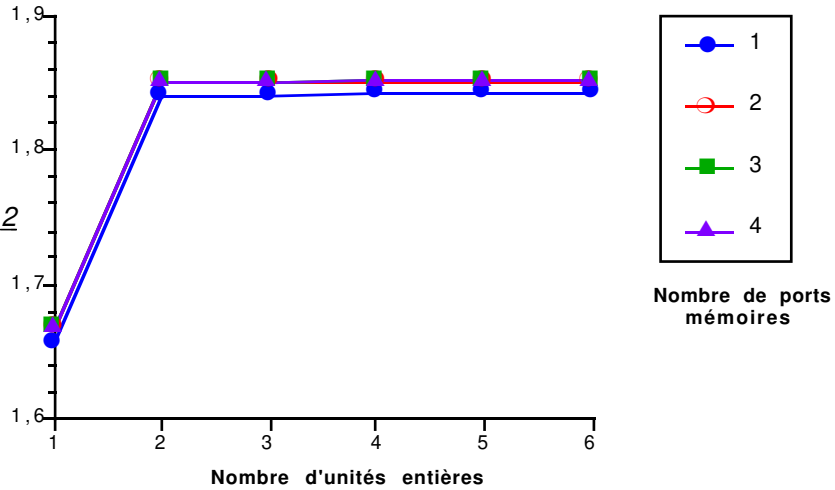


Figure 5-7: Degré 2

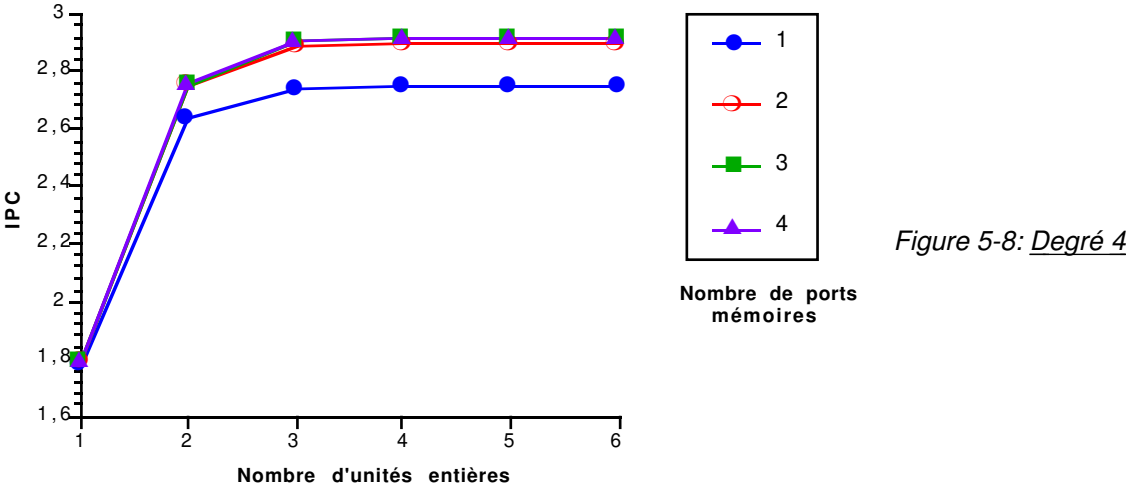


Figure 5-8: Degré 4

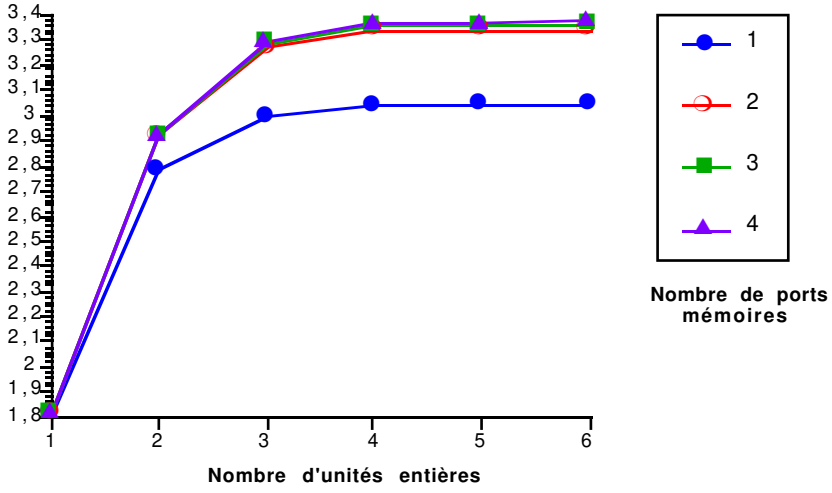


Figure 5-9: Degré 6

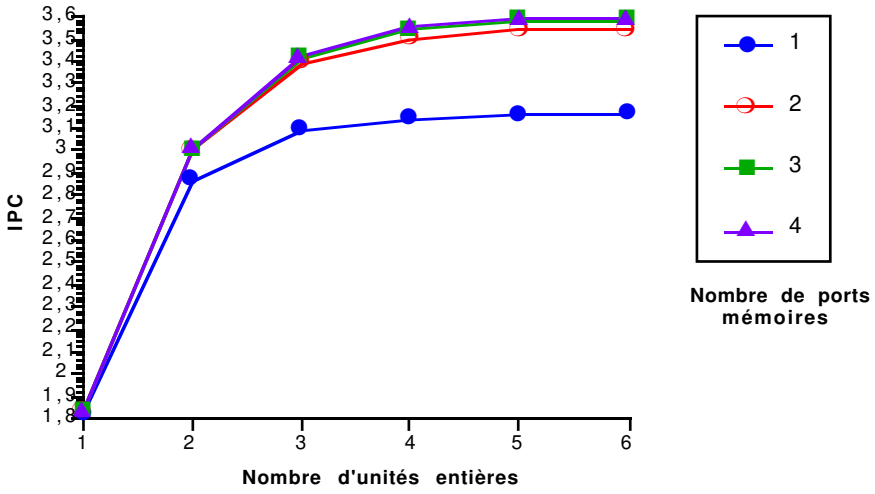


Figure 5-10: Degré 8

La figure 5-7 montre très nettement qu'un processeur de degré 2 proposant 2 unités entières et un seul port mémoire représente un bon compromis coût-performance. En effet, on peut voir que l'ajout de ports ou d'unités entières supplémentaires n'apporte presque aucun gain. Une telle configuration donne 99,5 % de l'IPC idéal, celui-ci étant l'IPC obtenu sans aucune limitation sur le nombre d'instructions initiées.

La figure 5-8 montre par contre la nécessité d'implémenter 2 ports. Pour un même nombre d'unités entières, la différence entre 1 et 2 ports est supérieure à 5 %. Le bon compromis pour un processeur de degré 4 semble donc être une configuration de 3 unités entières et 2 ports mémoire. La performance mesurée est alors de 99,1 % de l'IPC idéal.

Les mêmes résultats peuvent être remarqués sur la figure 5-9 mais pour un processeur de degré 6. La performance mesurée pour une telle configuration est alors de 97 % de l'IPC idéal.

Un processeur de degré 8 nécessite par contre une unité entière supplémentaire comme le montre la figure 5-10. L'ajout d'un autre port n'est par contre pas nécessaire. La performance est alors supérieure à 97,5 % de l'IPC idéal. Enfin, on peut remarquer que l'implémentation d'un seul port va sévèrement limiter la performance. Bien que moindre pour des degrés inférieurs, cette limitation justifie les implémentations actuelles d'unité mémoire à plusieurs ports (*Power 2* par exemple).

La table suivante rappelle les configurations retenues. Il est à noter qu'un processeur de degré 4 n'implémentant qu'un port mémoire n'offre que 94,1 % de la performance idéale pour les programmes entiers. De plus, des simulations sur les programmes test flottants ont montré que de telles configurations offraient plus de 99 % de la performance idéale.

Degré	2	4	6	8
Entière / Mémoire	2/1	3/2	3/2	4/2
Accélération	2,02	3,17	3,59	3,84

5.2. Unité entière

L'unité entière définie jusqu'à maintenant dans cette section, comme dans le PowerPC 604, traitait toutes les opérations arithmétiques, logiques et de manipulation de champs de bits. Ceci énumère deux classes d'instructions qui sont ENTIERE et CHAMPS DE BITS. Comme dans le MC 88110, une séparation des unités suivant les classes d'instructions peut être envisagé. Mais, sans tomber dans un tel extrême, des solutions partielles peuvent exister comme un mixage d'unités traitant l'ensemble des deux classes d'instructions et d'autres ne traitant qu'une classe d'instructions. Il est à noter qu'une telle solution ne rajoute rien au contrôle (sous-entendu au temps de cycle) puisqu'il suffit d'avoir une liste des unités par classes d'instructions quant au choix d'allocation des unités. De plus, une priorité est nécessairement établie, elle devra alors privilégier les unités ne traitant que la classes d'instructions concernée par rapport à ceux traitant les deux classes d'instructions.

Les figures 5-11 à 5-14 présentent, pour chaque programme test et pour la moyenne de ceux-ci, les résultats des différentes configurations définies pour un degré d'initiation fixé. Elles sont classées par ordre de coût d'implémentation croissant. Dans ces figures, une *Unité Champs de bits*

(UC) désigne une unité ne traitant que des instructions appartenant à la classes d'instructions CHAMPS DE BITS, une *Unité Entière* (UE) une unité ne traitant que la classe d'instructions ENTIERE, et une *Unité Entière et Champs de bits* (UEC) une unité traitant les deux classes d'instructions.

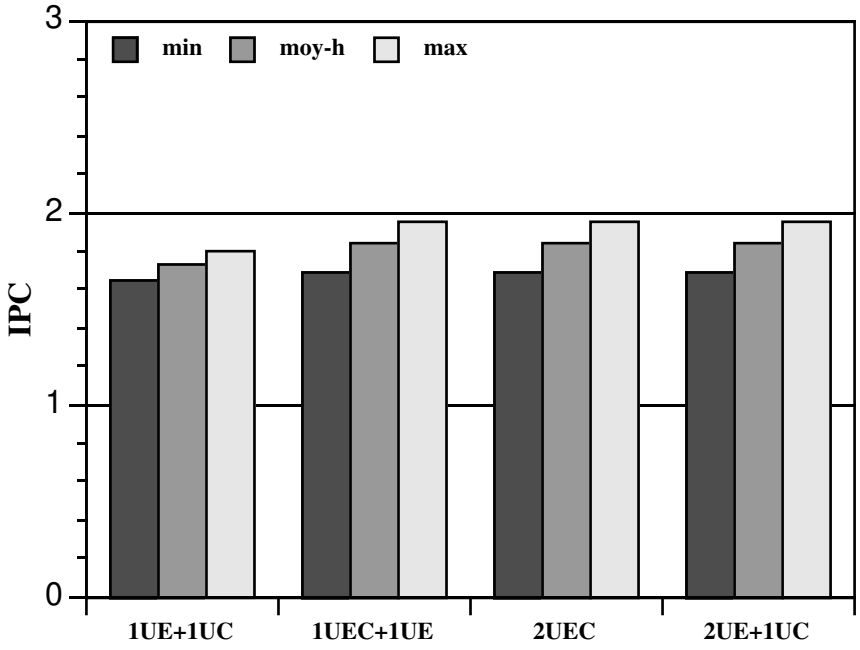


Figure 5-11 : Degré 2

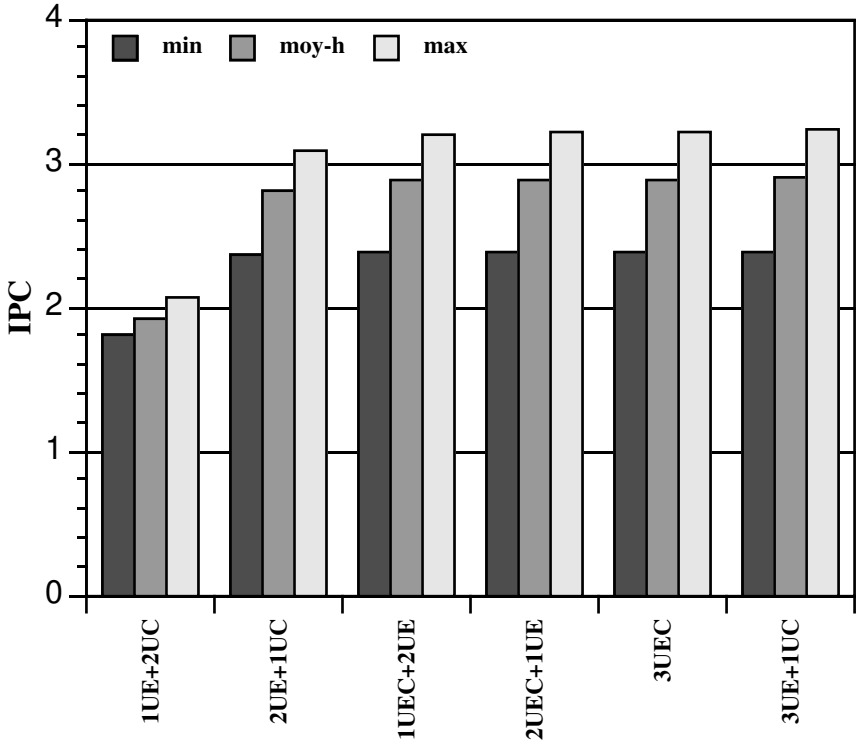


Figure 5-12 : Degré 4

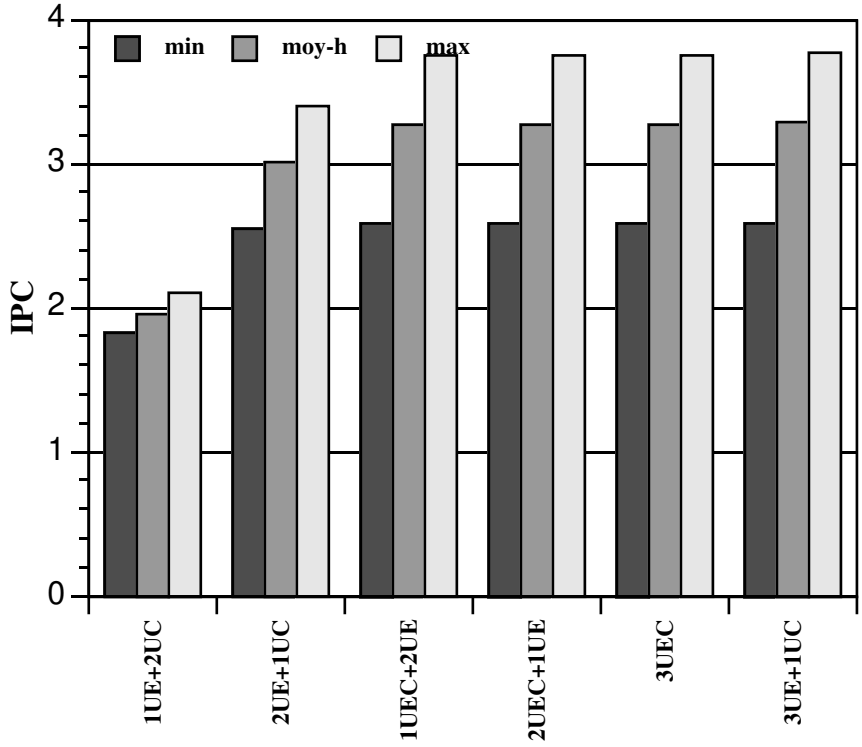


Figure 5-13 : Degré 6

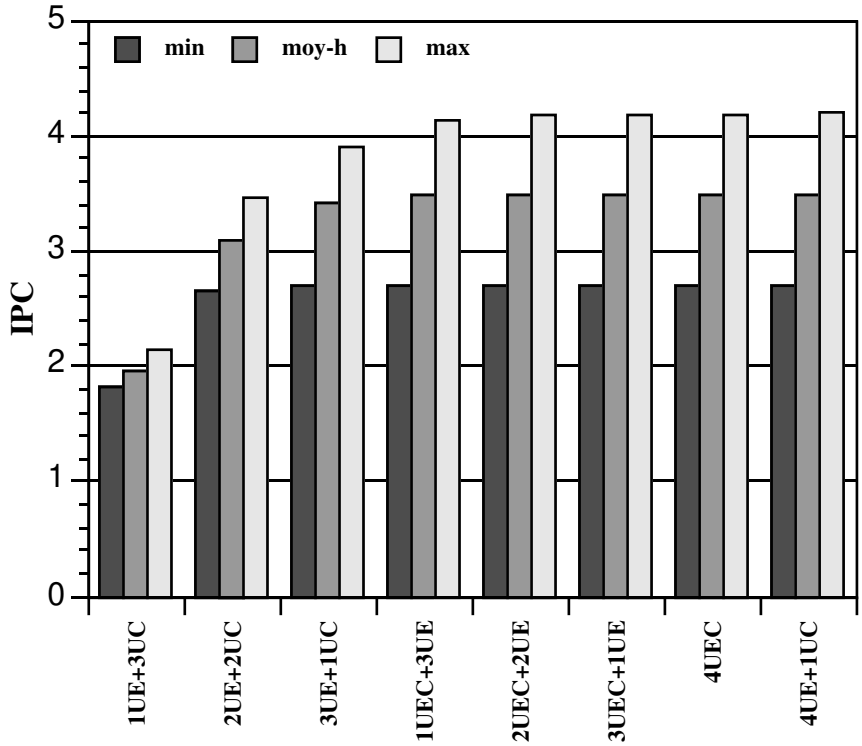


Figure 5-14 : Degré 8

Les figures 5-11 à 5-14 montrent parfaitement que les solutions extrêmes n'offre pas un bon rapport coût-performance. Pour ce qui concerne des processeurs de degré inférieur à 8, il semble que la solution la plus viable est d'implémenter une seule unité entière polyvalente, le reste ne s'occupant que de la classe d'instructions ENTIERE. Les performances obtenues sont alors supérieures à 99,8 % vis

à vis d'une configuration ayant des unités entières totalement polyvalentes. Enfin, pour un degré 8, il semble nécessaire d'envisager une configuration avec 2 unités polyvalentes, ce qui amène un petit gain en performance mais pour un coût relativement faible.

De plus, nous avons aussi évalué les effets de restrictions imposées sur les instructions entières multi-cycle. L'implémentation d'une seule unité les appréhendant a donné une performance semblable.

Finalement, nous avons abouti aux configurations ci-dessous dont nous conserverons les chiffres pour les sections suivantes :

	FA	Ports mémoire	UE	UEC	Multi-cycles	Accélération
Processeur de degré 2	16	1	1	1	1	2,02
Processeur de degré 4	32	2	2	1	1	3,16
Processeur de degré 6	48	2	2	1	1	3,59
Processeur de degré 8	64	2	2	2	1	3,84

6. Unité Flottante

Dans la plupart des microprocesseurs superscalaires actuels, l'ensemble des instructions flottantes sont traitées par une même unité flottante que nous dénommerons Unité Universelle Flottante (UUF). Ceci implique que les chemins de données entiers et flottants peuvent être séparés comme dans le PowerPC 604. Afin d'exploiter le parallélisme au niveau des instructions, il semble normal d'envisager l'implémentation de plusieurs de ces unités. Néanmoins, le coût en logique de plusieurs UUFs est très élevé.

Nous considérerons donc une seconde configuration où les instructions de division sont traitées par une unité séparée appelée *Unité de Division* (UD), les autres instructions flottantes étant traitées par une ou plusieurs unités que l'on dénomme *Unité Spécialisée Flottante* (USF). De plus, puisque les algorithmes de traitement des divisions entières et flottantes sont similaires, les divisions entières seront initiées dans l'UD. Il est à noter que cette unité est la seule non-pipeline, ce qui devrait simplifier le contrôle de certaines unités autrefois à la fois pipeline et non-pipeline.

Finalement, une troisième configuration peut être envisagée car l'unité multi-cycle entière ne traite plus que les multiplications et donc devient vraiment sous-utilisée. Nous considérerons donc dans cette configuration que les multiplications flottantes sont aussi traitées par cette unité que l'on appellera alors *Unité de Multiplication* (UM), tandis que les conversions et les opérations arithmétiques flottantes sont initiées dans des *Unités Flottantes* (UFl). Cette configuration correspond au découpage du MC 88110.

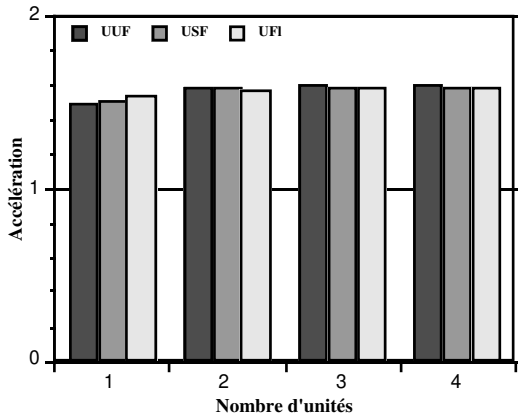


Figure 6-1 : degré 2

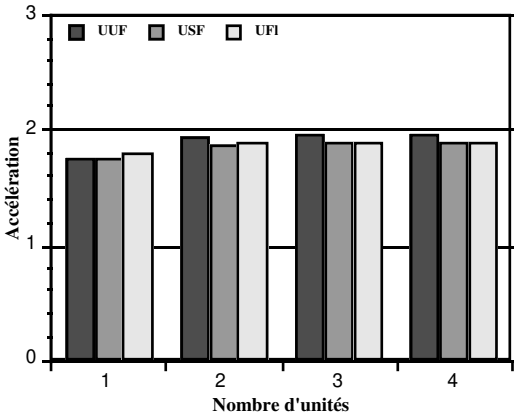
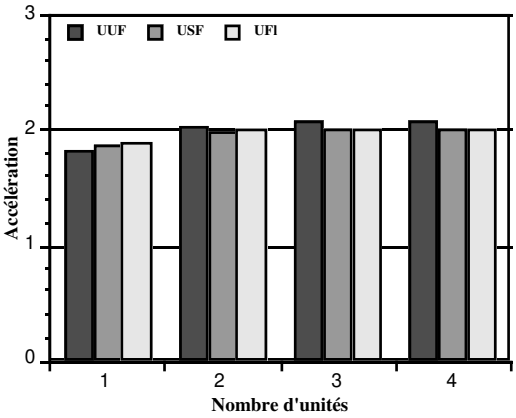


Figure 6-2 : degré 4

Figure 6-3 : degré 6



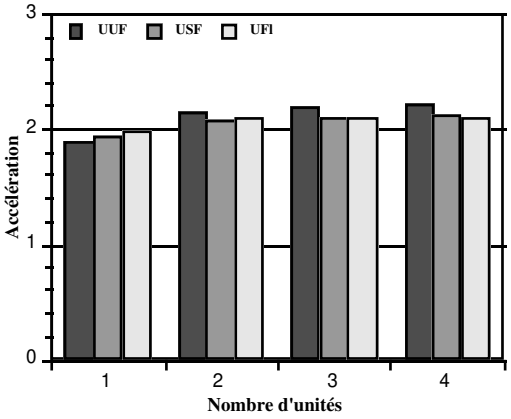


Figure 6-4 : degré 8

Les figures 6-1 à 6-4 tracent les accélérations pour des processeurs superscalaires à initiation non ordonnée de degré 2 à 8, en fonction du nombre d'unités flottantes principales (UUF, USF ou UFl). Il est à noter que d'autres simulations dont les résultats ne sont pas décrits ici, ont montré que la mise en œuvre de plus d'une UD ou d'une UM n'améliorent pas la performance.

La première conclusion de ces figures est que l'implémentation séparée d'une UD et d'une UM est une solution efficace quant à un compromis coût-performance.

Hormis pour un processeur de degré 2, 2 UFls sont suffisantes pour exploiter le *Parallélisme au niveau des instructions* : on obtient alors 95 % ou plus de la performance maximale. Pour un processeur de degré 2, 1 seule UFL suffit, obtenant aussi 95 % du maximum.

En conclusion, les configurations finales retenues sont les suivantes :

	FA	Ports mémoire	UE	UEC	UM	UD	UFl	Accélération (flottant)
Processeur de degré 2	16	1	1	1	1	1	1	1,53
Processeur de degré 4	32	2	2	1	1	1	2	1,88
Processeur de degré 6	48	2	2	1	1	1	2	1,99
Processeur de degré 8	64	2	2	2	1	1	2	2,09

7. Occupation des UFs

CINT92 - Degré	2	4	6	8
Entier[13]	49,6 %	52,1 %	59,5 %	48,3 %
Arithmétique	69,9 %	62,8 %	65,7 %	60,2 %
Champs de bits	29,2 %	31,3 %	47,7 %	36,6 %
Mémoire	45,9 %	36,1 %	41,1 %	44 %
Multiplication	0,2 %	0,3 %	0,4 %	0,4 %
Division	0,1 %	0,1 %	0,1 %	0,1 %
Flottant	0 %	0 %	0 %	0 %

CFP92 - Degré	2	4	6	8
Entier	21,1 %	18,1 %	19,6 %	16,5 %
Arithmétique	31,5 %	22,2 %	23,3 %	22,7 %
Champs de bits	14,1 %	10,2 %	12,7 %	10,1 %
Mémoire	34 %	21,5 %	23,1 %	24,8 %
Multiplication	9,5 %	12 %	13 %	14,3 %
Division	12 %	13 %	14 %	16 %
Flottant	17,3 %	11,1 %	11,9 %	12,9 %

[13] ENTIER concerne toutes les unités ARITHMETIQUES entières ou CHAMP DE BITS.

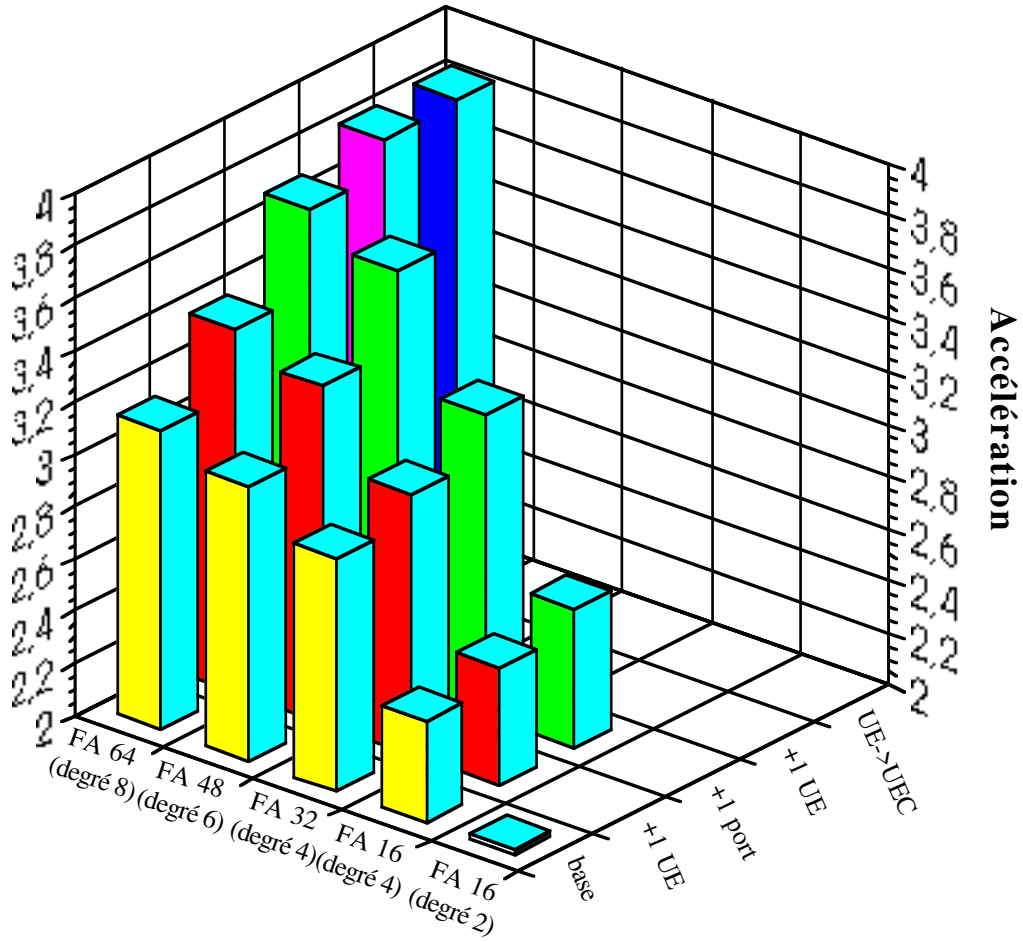
8. Conclusion

Nous avons dans ce rapport, montré les résultats de simulations dont le but était de déterminer des configurations d'unités fonctionnelles offrant le meilleur compromis coût-performance, pour des processeurs superscalaires à initiation non ordonnée. Tout le long du document, nous avons considéré des processeurs de degré 2, 4, 6 et 8 tout en ne prenant en compte que les dépendances de données : mémoire et registres. Nous avons donc abouti à des configurations qui sont supposées ne pas fausser toute étude future. Ces configurations sont rappelées ci-dessous avec les accélérations obtenues par rapport au processeur scalaire à initiation ordonnée décrit dans la section 2. Nous obtenons 6 unités pour un processeur de degré 2, et jusqu'à 9 unités pour un processeur de degré 8 [14].

	FA	Ports mémoire	UE	UEC	UM	UD	UFl	Accélération (entier)	Accélération (flottant)
Degré 2	16	1	1	1	1	1	1	2,02	1,53
Degré 4	32	2	2	1	1	1	2	3,16	1,88
Degré 6	48	2	2	1	1	1	2	3,59	1,99
Degré 8	64	2	2	2	1	1	2	3,84	2,09

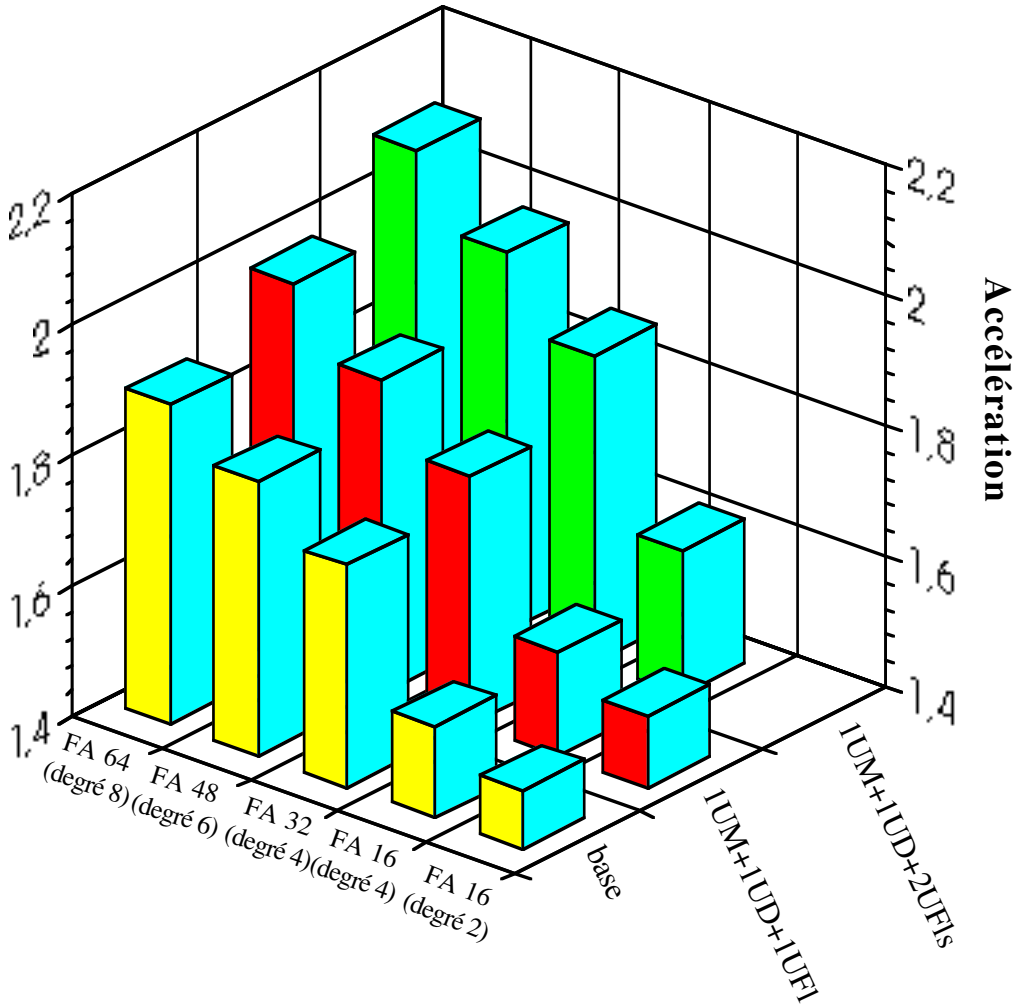
Les graphes 3D ci-dessous présentent les accélérations par rapport au processeur scalaire avec les hypothèses de simulation. Nous considérons l'augmentation de la taille de la FA et du degré d'initiation sur l'axe des abscisses, et l'addition d'unités fonctionnelles et de ports du cache sur l'axe des ordonnées. Nous nous intéresserons surtout à des comparaisons vis à vis de la configuration exécutive du PowerPC 604 (FA 16 - DEGRE 4 - BASE) pour comparaison.

[14] l'unité mémoire, quelque soit son nombre de ports, est considérée comme une seule unité.



Le premier graphe, qui concerne les programmes sur les entiers, montre que pour une FA de la taille du PowerPC 604, l'adjonction d'autres unités n'offre pas un bon compromis coût-performance. Cependant, accroître simultanément la taille de la FA et le nombre d'unités est une façon efficace d'améliorer les performances. On peut aussi remarquer l'impact de ports mémoire additionnels sur la performance. La meilleure configuration qui est un processeur de degré 8 avec 2 ports et 2 unités entières supplémentaires, apporte une accélération relative vis à vis de la configuration du PowerPC 604 de 1,61. A l'inverse, réduire le degré d'initiation à deux instructions par cycle diminue la performance d'environ 15,5 %.

Dans toute cette étude, nous avons supposé que tous les ajouts ont été faits sans dégrader le temps de cycle.



Le second graphe présente les mêmes résultats mais sur les programmes flottants. Pour chaque taille de FA et degré d'initiation, on a associé au processeur les unités entières définies pour ces valeurs : seul donc varie les unités flottantes. De même que pour les entiers, avec une taille de FA égale à 16, on remarque que l'ajout d'unité apporte peu relativement (5 %). Il est donc préférable d'augmenter la FA dans ce cas. La configuration maximale quant à elle offre une accélération relative vis à vis de la configuration du PowerPC 604 de 1,36.

9. Bibliographie

[BuPa92] M. Butler & Y. Patt, “An Investigation of the Performance of Various Dynamic Scheduling techniques”, *Proceedings of the 25th Annual International Symposium on Microarchitecture* (december 1992), pp. 1-9

[Hsu94] P. Yan-Tek Hsu, "Designing the TFP Microprocessor", *IEEE Micro,* April 94, pp. 23-33

[IbMo94] "PowerPC 604 RISC Microprocessor Technical Summary", *IBM and Motorola Advance Information*, MPR604TSU-1

[Inte93] "Pentium Processor User's Manual", *Intel*, 1993

[Irla92] G. Irlam, The Spa Package, 1992, available from gordoni@cs.adelaide.edu.au

[John91] M. Johnson, "Superscalar Microprocessor Design, *Prentice-Hall*, 1991

[McL93] E. McLellan, "The Alpha AXP Architecture and the 21064 Processor", *IEEE Micro,* vol. 13, N° 3, 1993

[Moto91] "MC88110: Second Generation RISC Microprocessor User's Manual", *Motorola*, 1991.

[SmPl85] J.E. Smith, A.R. Pleszkun, "Implementation of Precise Interrupts in Pipelined Processors", *Proceedings of the 12th Annual International Symposium on Computer Architecture* (june 85), pp. 36-44

[SmWe94] J.E. Smith, S. Weiss, "Power and PowerPC, Principles, Architecture, Implementation", *Morgan Kaufmann Publishers*, Inc., 1994

[SPEC92] SPEC 92 — Technical Manual — Rev. 1.1

[Toma67] R.M. Tomasulo, "An efficient Algorithm for Exploiting Multiple Arithmetic Units", *IBM Journal,* vol. 11, January 1967.

[YePa92] T.Y. Yeh, Y.N. Patt, “A Comprehensive Instruction Fetch Mechanism for a Processor Supporting Speculative Execution”, *Proceedings of the 25th Annual Symposium on Microarchitecture*, Portland, Oregon, December 1992.

INSTITUT DE RECHERCHE EN INFORMATIQUE DE TOULOUSE

UPS, 118 route de Narbonne. 31062 Toulouse Cedex, France
Tél. (33) 61 55 67 65 / fax. (33) 61 55 62 58

ENSEEIHT, 2 rue Camichel. 31071 Toulouse Cedex, France
Tél. (33) 61 62 78 62 / fax. (33) 61 58 82 09

w^3 : http://www.irit.fr/