Instructions SIMD flottantes 16 bits pour réduire la consommation dans les processeurs embarqués à jeux d'instructions spécialisables

Stéphane Piskorski*, Lionel Lacassagne** et Daniel Etiemble*

*LRI, **IEF, Université Paris Sud

Résumé
Nous évaluons les performances en termes de temps d'exécution et d'énergie consommée d'instructions SIMD flottantes 16 bits sur des benchmarks significatifs de traitement d'images utilisés notamment pour la stabilisation d'images. Les instructions sont ajoutées au processeur « soft core »Nios II pour deux kits d'Altera : Cyclone et StratixII. Pour les benchmarks pour lesquels l'utilisation d'entiers 16 bits est insuffisante, les instructions SIMD flottantes 16 bits monocycles fournissent un gain en vitesse et consommation sur la plupart des benchmarks ; cet avantage est plus significatif sur les versions les moins performantes des FPGA considérés.

Mots-clés : instructions flottantes SIMD 16 bits, processeurs embarqués, FPGA et traitement d'images.

1. Introduction

Pour les processeurs enfouis ou embarqués ou les processeurs de traitement du signal (DSP), l'utilisation des flottants (généralement limités au format simple précision) est souvent réservée aux processeurs « haut de gamme » du domaine. L'utilisation des entiers, soit avec le format classique en complément à deux, soit en virgule fixe est considérée comme la solution la plus économique. Cette opinion a également des défenseurs dans le cas des processeurs d'usage général. Par exemple, G Kolli justifie "l'utilisation de la virgule fixe au lieu du flottant pour de meilleures performances 3D" dans la bibliothèque GPP d'Intel (Graphics Performance Primitive) [1].
Le format flottant 16 bits[1] utilisé par Nvidia dans des processeurs graphiques (GPU) offre une solution intermédiaire entre entiers 32 bits et flottants 32 bits. En utilisant des instructions SIMD 16 bits entières existant dans les jeux d'instructions IA-32 et Power-PC, et en supposant que la latence et le débit de démarrage des instructions SIMD flottantes 16 bits seraient les mêmes que les instructions SIMD flottantes 32 bits du Pentium 4 et du Power PC, nous avons évalué par mesure de temps d'exécution en cycles d'horloge sur Pentium 4 et Power PC G4 et G5 les performances résultant de l'utilisation de flottants 16 bits sur un certain nombre de benchmarks représentatifs en traitement d'images et traitement multimédia [2,3].
Nous avons commencé l'évaluation de ces instructions flottantes 16 bits dans le contexte enfoui et embarqué en spécialisant[2] des instructions flottantes 16 bits sur le processeur Nios II qui est un « soft core »utilisé dans plusieurs familles de FPGA d'Altera. Dans [4], nous donnons les performances avec le kit Cyclone (composant FPGA EP1C20F400C7) avec des instructions ADDF16/SUBF16 et MULF16 s'exécutant en deux cycles alors que les autres instructions ajoutées au processeur s'exécutent en un cycle. Pour toutes les applications pour lesquels les flottants 16 bits ont une précision et une amplitude suffisantes, ils offrent l'avantage par rapport au calcul entier de permettre l'utilisation facile de l'approche SIMD, manuellement ou par compilation, avec l'avantage d'un format réduit à 16 bits, permettant d'effectuer de 2 opérations (registres de 32 bits du processeur Nios II) à 8 opérations en parallèle (registres de 128 bits des processeurs généralistes).

[1] Le format flottant F16 de NVidia est appelé F16 dans la suite de cet article. C'est la version 16 bits des formats flottants IEEE simple et double précision : 1 bit de signe, 5 bits d'exposant avec un excès de 15 et 10 bits de mantisse.
[2] « spécialiser »est utilisé ici comme traduction du verbe anglais « customize »

L'utilisation des instructions SIMD est généralement considérée comme un moyen de réduire les temps d'exécution. Cependant, elle peut également permettre de réduire la consommation, comme l'ont montré N. Drach et J. Sebot [5, 6] dans le cas de processeurs généralistes (superscalaires). Dans cet article, nous étendons l'étude au cas de processeurs embarqués sur FPGA ne disposant pas d'instructions SIMD natives. Sur un certain nombre de benchmarks de traitement d'images et multimédia, nous examinons les performances dynamiques et la consommation des versions utilisant des flottants 16 bits par rapport aux versions avec des formats entiers. Nous examinons notamment ces performances dans l'hypothèse où les opérations de base (addition/soustraction et multiplication) sont exécutables en un ou deux cycles d'horloge. Le processeur (soft core) Nios II d'Altera est utilisé avec le kit Cyclone déjà évalué avec des instructions ADDF16/SUBF16 et MULF16 en deux cycles et avec le kit StratixII qui est actuellement le plus performant disponible chez Altera. Les instructions supplémentaires sont réellement implantées sur le FPGA en même temps que le processeur et les résultats expérimentaux sont des mesures de temps d'exécution (en cycles d'horloge et ns) pour des programmes réellement exécutés. Nous présentons d'abord la méthodologie en détaillant les benchmarks utilisés, les environnements matériels et logiciels utilisés et les instructions ajoutées au jeu d'instructions Nios II. Nous présentons ensuite les performances obtenues pour les deux kits Altera en comparant les performances obtenues avec des instructions F16 en deux cycles et en un cycle avec les versions entières, qu'elles soient scalaires ou qu'elles utilisent également des instructions SIMD 16 bits « spécialisées ». L'impact sur la fréquence maximale du processeur et la performance globale du choix d'une implantation en un cycle ou deux cycles sont examinés.

2. Méthodologie

2.1. Les benchmarks utilisés

Nous avons montré dans [3] l'intérêt des instructions F16 SIMD pour des algorithmes de traitement d'images pour lesquels la dynamique des entiers 16 bits est insuffisante, sans toutefois nécessiter l'utilisation de flottants 32 bits. Sans flottants 16 bits, l'utilisation d'entiers 32 bits est inévitable, ce qui rend impossible l'utilisation d'instructions SIMD pour un processeur avec registres de 32 bits. Tout en conservant des benchmarks de ce type, nous examinons également des benchmarks pour lesquels l'utilisation des entiers s'impose habituellement. Les versions C de chaque benchmark utilisé sont disponibles en ligne [7].

2.1.1. Grayscale

Grayscale est un benchmark que nous avons dérivé de la description du benchmark du même nom de la suite EEMBC. La dynamique des entiers 16 bits y est suffisante. Il ne nécessite pas la dynamique des flottants 16 bits. C'est un filtre passe haut (figure 1) sur une image en niveaux de gris (les pixels sont codés avec des octets non signés). Il est représentatif des filtres spatiaux de traitement d'images bas niveau. En calcul entier, les multiplications et la division peuvent facilement être remplacés par des décalages ou des combinaisons de décalages et d'addition ou soustraction ($255 = 2^8 - 1\,; 28 = 2^5 - 2^2$).

$$\frac{1}{256}\begin{bmatrix} -28 & -28 & -28 \\ -28 & 255 & -28 \\ -28 & -28 & -28 \end{bmatrix}$$

FIG. 1 – Filtre « grayscale »d'après la description du benchmark de la suite EEMBC

2.1.2. Gradient et lisseur de Deriche

Le gradient de Deriche et la version vectorisable du lisseur de Deriche sont des opérateurs de convolution qui nécessitent une dynamique supérieure à celle des entiers 16 bits. Nous les avons utilisé pour

évaluer les versions SIMD F16 sur processeurs généralistes [2, 3]. Il sont aussi représentatifs des filtres spatiaux et ont des temps de calcul significatifs par rapport aux temps d'accès mémoire.
Le gradient a la particularité de ne pas comporter de multiplications, contrairement au lisseur. La comparaison des deux benchmarks permet de mettre en évidence le rôle des multiplications en fonction de la disponibilité ou non de blocs dédiés pour la multiplication entière dans les différents FPGA.

2.1.3. Détection de points d'intérêt : Harris

L'algorithme de Harris est présenté avec l'algorithme de Achard dans la figure 2. Ils ont en commun la première partie des calculs, avec successivement des gradient de Sobel et des filtres de Gauss et comprennent un nombre significatif de multiplications. Le calcul final distingue les deux algorithmes. Comme ils ont des temps d'exécution très semblables, nous ne donnons les résultats que pour Harris. Ces algorithmes sont représentatifs de traitements d'images de niveau moyen. Ils sont utilisés pour la détection de points d'intérêt dans une image, notamment pour la stabilisation d'images avec des caméras embarquées en robotique.

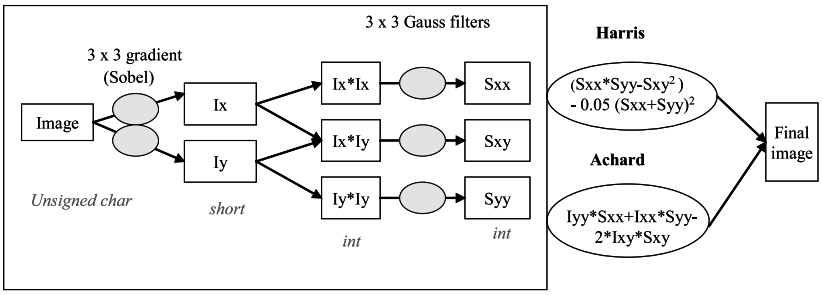

FIG. 2 – Les algorithmes de Achard et Harris

2.1.4. Flot optique

Les algorithmes de flot optique, comme ceux de détection des points d'intérêt, sont utilisés essentiellement pour la stabilisation d'images. Ils calculent les différences entre images successives résultant du mouvement. Nous utilisons l'algorithme de Horn et Shunk, dont la version C scalaire est donnée en [7] avec celle de l'ensemble des autres benchmarks. Son intérêt du point de vue évaluation de performances est l'utilisation inévitable d'une véritable division, à côté de divisions par des constantes qui peuvent être réalisées par des multiplications dans le cas flottant, ou des combinaisons de multiplications entières et de décalages dans le cas des formats entiers.

2.1.5. DCT directe (JPEG 6a)

La DCT directe (FDCT) de JPEG 6.a utilisée dans Mediabench[8]. Ce benchmark présente l'intérêt de posséder deux versions entières (normale et rapide) et une version flottante 32 bits avec laquelle il est possible de comparer la version F16. L'autre intérêt est qu'il n'existe pas de version SIMD triviale, ce qui nous conduira à comparer une version scalaire F16 avec les versions scalaires entières et flottantes.

2.2. FPGA avec processeurs « soft cores »

Nous utilisons deux kits différents d'Altera, le kit Cyclone et le kit StratixII. La version FPGA Cyclone utilisée contient jusqu'à 20k éléments logiques et 288 Kbits de mémoire RAM. La version FPGA StratixII utilisée contient jusqu'à 60k éléments logiques, plus de 2 Mbits de RAM, 36 blocs DSP permettant entre autres de disposer de 144 multiplieurs 18 bits x 18 bits. Pour notre utilisation, c'est essentiellement les blocs DSP qui introduisent une différence significative pour la réalisation des multiplieurs. Les deux FPGA peuvent utiliser le processeur « soft core »Nios II, dont les caractéristiques essentielles sont données dans la table 1. Il est possible de définir et ajouter de nouvelles instructions au processeur [9], comme le montre la figure 3. Les opérateurs matériels à ajouter sont définis en VHDL ou Verilog. Pour

Caractéristiques fixes	Caractéristiques paramétrables
Processeur RISC 32 bits (exécution scalaire)	Multiplication câblée
Prédiction de branchement	Division câblée
Prédicteur de branchement dynamique	Cache instruction (4Ko pour nos mesures)
Décaleur câblé (*barrel shifter)*	Cache données (2Ko pour nos mesures)
	Instructions supplémentaires (*customization)*

TAB. 1 – Caractéristiques du processeur Nios II

les opérateurs combinatoires, les interfaces définies sont les données d'entrée 32 bits (*dataa* et *datab*) et la sortie 32 bits (*result*). Quand le temps de propagation est supérieur au temps de cycle du processeur, des opérations multi-cycles sont utilisées. Elles ont les mêmes interfaces que les opérateurs combinatoires, plus les signaux d'horloge et de contrôle définis dans la figure 3. Les opérations multi-cycles suspendent le processeur jusqu'à ce que le signal *done* soit actif. Avec une interface processeur 32 bits, il est naturel de définir des opérations SIMD 2 x 16 bits ou 4 x 8 bits. L'utilisation d'opérandes F16 double le débit d'opérations par rapport à des calculs entiers ou flottants sur 32 bits. L'utilisation d'instructions ajoutées dans un programme C est triviale. Il y a deux types de « *define* » selon que les opérandes ont une ou deux entrées :

1. `#define INST1(A) __builtin_custom_ini(Opcode_INSTR1,(A))`
2. `#define INST2(A,B) __builtin_custom_inii(Opcode_INSTR2,(A),(B))`

Pour les deux kits, la fréquence de fonctionnement minimale est 50 MHz. La fréquence de fonctionnement maximale dépend des instructions spécialisées ajoutées, et est supérieure pour le kit StratixII. Le logiciel Quartus d'Altera a été utilisé pour générer le fichier de configuration des FPGA à partir des fichiers du Nios II et du code VHDL des instructions ajoutées. Tous les benchmarks ont été compilés avec l'environnement de développement d'Altera (IDE) qui utilise la chaîne de compilation GCC. L'option -O2 a été utilisée, en mode « *release* ». Les temps d'exécution ont été mesurés avec le timer haute résolution qui fournit le nombre de cycles d'horloge pour exécuter le benchmark. La plupart des résultats utilisent la métrique CPP (nombre de cycles d'horloge divisée par le nombre de pixels) dans le cas des benchmarks de traitement d'images. Les temps d'exécution ont été mesurés au moins 5 fois et la valeur moyenne a été prise.

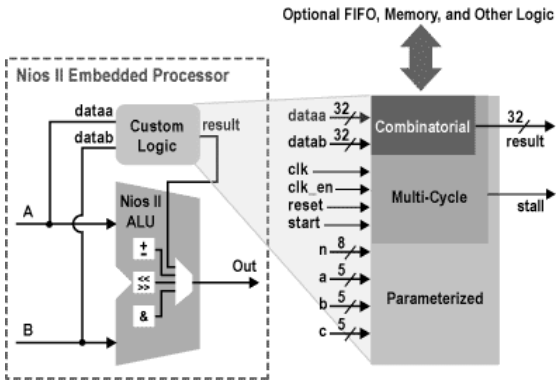

FIG. 3 – Ajout d'instructions au processeur Nios II

2.3. Instructions SIMD

La table 2 donne la liste des instructions SIMD F16 qui ont été définies et sont utilisées pour les deux FPGA. Les sept premières instructions correspondent aux manipulations de données pour permettre le

Instruction	Effet	Cycles
Notation	*Le mot X (32 bits) comprend deux F16 (XH et XB)*	
B2FH (A)	Convertit les deux octets hauts du mot de 32 bits A en deux F16	1
B2FL (A)	Convertit les deux octets bas du mot de 32 bits A en deux F16	1
B2FSRH (A)	Convertit les deux octets milieu de A en deux F16	1
B2FSRL (A,B)	Convertit l'octet de poids faible de A et l'octet de poids fort de B en deux F16 (combinaison décalage octet et conversion)	1
F2BH (A,B)	Convertit les deux F16 dans A en deux octets non signés dans la partie haute de B. La partie basse de A reste inchangée	1
F2BL (A)	Convertit les deux F16 dans A en deux octets non signés dans la partie basse d'un mot de 32 bits	1
FSR (A,B)	Met le F16 bas de A dans le F16 haut du résultat. Met le F16 haut de B dans le F16 bas du résultat	1
FADD (A,B)	$RL \leftarrow AL + BL$; $RH \leftarrow AH + BH$	2/1
FDP2 (A,B)	$RL \leftarrow AL / 2^{BL}$; $RH \leftarrow AH / 2^{HL}$	1
FMUL (A,B)	$RL \leftarrow AL * BL$; $RH \leftarrow AH * BH$	2/1
FSUB (A,B)	$RL \leftarrow AL - BL$; $RH \leftarrow AH - BH$	2/1
FDIV (A,B)	$RL \leftarrow AL / BL$; $RH \leftarrow AH / BH$	8

TAB. 2 – Instructions F16 SIMD utilisées

traitement SIMD et les suivantes correspondent aux calculs proprement dits. La division utilisée pour le flot optique est extraite de la bibliothèque VHDL de l'ENS-Lyon [10]. Sa latence est de 8 cycles. Pour toutes les opérations flottantes 16 bits, les nombres dénormalisés ne sont pas traités. La table 3 donne la liste des instructions SIMD I16 (entiers 16 bits) définies et utilisées pour le benchmark Grayscale, pour lesquels la dynamique des entiers 16 bits est suffisante. Ces instructions permettent de comparer les versions SIMD F16 et I16.

2.4. Instructions flottantes en un ou deux cycles ?

L'addition/soustraction et la multiplication F16 peuvent être implantées de manière pipelinée ou non. Compte tenu de la réduction de la taille de la partie fractionnaire et de la partie exposant avec les F16, les versions pipelinées ne nécessitent que deux cycles. C'est l'addition soustraction qui est critique à cause des différentes étapes :

– comparaison des parties exposant
– décalage de la partie fractionnaire pour le nombre ayant la plus petite partie exposant
– addition ou soustraction des mantisses (selon l'opération et les bits de signe)
– renormalisation.

Pour obtenir une version 1 cycle de l'additionneur, nous avons dupliqué un certain nombre d'opérateurs : A op B et B op A sont calculés simultanément et le bon résultat est récupéré lorsque le maximum de A et B est connu. De même, des opérateurs séparés traitent en parallèle des cas particuliers, comme le cas de la recherche du 0 significatif lors de la soustraction de nombres ayant le même exposant. La version la plus rapide de l'additionneur utilisée pour les implantations « combinatoires » est donc beaucoup plus coûteuse en matériel. Elle n'est pas optimale en terme de nombre d'éléments logiques utilisés. Nous donnons plus loin l'impact en termes de ressources matérielles et de puissance dissipée pour les différents cas, notamment les différentes combinaisons : addition et multiplication en 1 cycle, addition et multiplication en 2 cycles, et multiplication 1 cycle et addition 2 cycles.

2.5. Evaluation de la puissance dissipée

Alors que la mesure des temps d'exécution des benchmarks est simple, l'évaluation de la puissance dissipée est plus complexe. La seule technique précise serait la mesure physique du courant consommé par le FPGA sur les cartes Cyclone ou StratixII durant l'exécution du benchmark. Nous nous contentons d'une estimation fournie par le logiciel Quartus II (PowerPlay). Malheureusement, il est impossible d'extraire les informations sur l'activation des différents signaux sur un circuit aussi complexe

Instruction	Effet	Cycles
Notation	*Le mot X (32 bits) comprend deux entiers 16 bits(I16) (XH et XB)*	
B2HH (A)	Convertit les deux octets hauts du mot de 32 bits A en deux I16	1
B2HL (A)	Convertit les deux octets bas du mot de 32 bits A en deux I16	1
DECD (A,B)	Met l'octet bas de A dans l'octet haut du résultat ; met les trois octets hauts de A dans les trois octets bas du résultat	1
DECG (A,B)	Met les 3 octets bas de A dans les trois octets hauts du résultat ; met l'octet haut de B dans l'octet bas du résultat	1
H2BH (A,B)	Convertit les deux I16 dans A en deux octets non signés dans la partie haute de B. La partie basse de A reste inchangée	1
H2BL (A)	Convertit les deux I16 dans A en deux octets non signés dans la partie basse d'un mot de 32 bits	1
SHL2 (A)	RL $\leftarrow$ AL $\ll 2$; RH $\leftarrow$ AH $\ll 2$	1
SHL5 (A)	RL $\leftarrow$ AL $\ll 5$; RH $\leftarrow$ AH $\ll 5$	1
SHL8 (A)	RL $\leftarrow$ AL $\ll 8$; RH $\leftarrow$ AH $\ll 8$	1
SHR8 (A)	RL $\leftarrow$ AL $\gg 8$; RH $\leftarrow$ AH $\gg 8$	1
ADDH	RL $\leftarrow$AL + BL ; RH $\leftarrow$ AH + BH	1
SUBH	RL $\leftarrow$ AL - BL ; RH $\leftarrow$ AH - BH	1

TAB. 3 – Instructions entières SIMD utilisées

	Grayscale	Gradient	Lisseur	Harris	Flot opt.	FDCT
B2FL, B2FH, B2SRL, B2SRH	XF	XF	XF	XF	XF	
F2BL, F2BH	XF	XF	XF	XF	XF	
FSR				XF	XF	
DP2			XF	XF	XF	
FADD2, FSUB2	XF2	XF2	XF2	XF2	XF2	XF2
FMUL2	XF2		XF2	XF2	XF2	XF2
FADD1, FSUB1	XF1	XF1	XF1	XF1	XF1	XF1
FMUL1	XF1		XF1	XF1	XF1	XF1
FDIV					XF	

TAB. 4 – Instructions spécialisées flottantes utilisées pour chaque benchmark

qu'un processeur avec les instructions ajoutées pendant l'exécution d'un programme. Sur l'ensemble processeur + instructions ajoutées, PowerPlay fournit une évaluation de la puissance consommée, en distinguant puissance statique et puissance dynamique en fonction d'un pourcentage d'activation de tous les signaux utilisés dans le circuit. Nous avons évalué la puissance totale dissipée (statique + dynamique) pour des pourcentages d'activation de 12,5%, 25% et 50% et nous montrerons que si les valeurs absolues diffèrent de manière significative, les rapports de consommation entre les différentes configurations de circuits restent très voisines, quel que soit le pourcentage d'activation utilisé pour déterminer la consommation.

3. Mesure de temps d'exécution et évaluation de la consommation

3.1. Les différentes configurations étudiées

La table 4 indique les instructions spécialisées utilisées par chaque benchmark. XF correspond aux programmes flottants, avec les variantes XF1 et XF2 selon que les instructions d'addition et de multiplication flottantes utilisent 1 ou 2 cycles. La version SIMD entière de Grayscale utilise l'ensemble des instructions de la table 3. On constate que tous les programmes flottants utilisent pratiquement les mêmes instructions, à l'exception notable du flot optique qui exige la division flottante.

Configuration	Contenu
Scalaire	Processeur avec multiplication et division câblées
F2	Processeur + instructions notées XF sauf FDIV + FADD2, FSUB2 et FMUL2
F21	Processeur + instructions notées XF sauf FDIV + FADD2, FSUB2 et FMUL1
F1	Processeur + instructions notées XF sauf FDIV + FADD1, FSUB1 et FMUL1
F2D	Configuration F2 + FDIV
F21D	Configuration F12 + FDIV
F1D	Configuration F1 + FDIV
SI16	Processeur avec toutes les instructions de la Table 3 (SIMD entiers)

TAB. 5 – Configurations de processeurs

Cyclone	N=256	N=256	N=256	N=256	N=260	N=256	N=260	N=64
	Grayscale	Gradient	Lisseur	Harris	Harris	Flot	Flot	FDCT
Scalaire	48.9	35.9	122.4	364.8	276.2	351	257.5	59.1
F2	48.8	29.3	36.6	240.2	208.7	208.65	194	44.1
F21		29.3	33.1	203.6	172.2	196.85	182.35	39.7
F1	31.4	15.6	22.5	154.8	123.3	131.9	117.2	19.8
SI16	31.9							

TAB. 6 – Performance en cycles par pixel (CPP) pour Cyclone à 50 MHz avec images N x N

On peut en déduire les différentes configurations, correspondant aux instructions spécialisées ajoutées au processeur, à tester d'un point de vue fréquence maximale d'exécution et puissance consommée. Les différentes configurations sont résumées dans la table 5.

3.2. Performances en cycles d'horloge.

Les performances, exprimées en cycles par pixel (CPP) pour tous les benchmarks, ont été mesurées avec F=50 MHz pour les kits Cyclone et StratixII. L'ensemble des résultats pour chaque benchmark pour des images (NxN) avec N=64, 128, 256 et 512 et pour les deux kits sont disponibles sur le site contenant le code des benchmarks [7]. Pour des raisons de place, nous fournissons dans les tables 6 et 7 les résultats pour N = 256 à l'exception de la FDCT pour laquelle N=64. Pour Harris et le flot optique, nous avons ajouté N=260. Cette valeur, différente d'une puissance de 2, permet de « contourner » le problème de défauts de cache de conflit liés à la correspondance directe utilisée dans les caches du Nios II.

Par rapport aux valeurs exprimées en CPP, les tables 8 et 9 donnent les accélérations par rapport à la version scalaire. Les résultats obtenus montrent l'impact de l'utilisation des instructions SIMD (deux opérations par instruction), de la latence des opérations arithmétiques flottantes (1 ou 2 cycles) et des défauts de cache liés à la correspondance directe. Pour Harris, qui accède à des octets, il y a quatre fois plus de défauts de conflit en scalaire qu'en SIMD (l'accès à quatre octets successifs se traduit par quatre accès au même mot de 32 bits qui provoquent chacun un défaut de cache à cause de la correspondance

StratixII	N=256	N=256	N=256	N=256	N=260	N=256	N=260	N=64
	Grayscale	Gradient	Lisseur	Harris	Harris	Flot	Flot	FDCT
Scalaire	44.6	35.8	60.2	315.6	227.4	288.6	195	21.9
F2	45.9	29	36	239.4	208	207.85	193.2	44.5
F21		29	32.5	202.8	171.4	196.25	181.5	39.7
F1	28.5	14.9	21.9	154.6	123	131.5	116.7	19.8
SI16	28.95							

TAB. 7 – Performance en cycles par pixel (CPP) pour StratixII à 50 MHz avec images N x N

Cyclone	N=256	N=256	N=256	N=256	N=260	N=256	N=260	N=64
	Grayscale	Gradient	Lisseur	Harris	Harris	Flot	Flot	FDCT
F2/S	1.00	1.23	3.34	1.52	1.32	1.68	1.33	1.34
F21/S		1.23	3.70	1.79	1.60	1.78	1.41	1.49
F1/S	1.56	2.30	5.44	2.36	2.24	2.66	2.20	2.98
SI16/S	1.53							

TAB. 8 – Accélération (à 50 MHz) par rapport à la version scalaire pour Cyclone

StratixII	N=256	N=256	N=256	N=256	N=260	N=256	N=260	N=64
	Grayscale	Gradient	Lisseur	Harris	Harris	Flot	Flot	FDCT
F2/S	0.97	1.23	1.67	1.32	1.09	1.39	1.01	0.49
F21/S		1.23	1.85	1.56	1.33	1.47	1.07	0.55
F1/S	1.56	2.29	2.75	2.04	1.85	2.19	1.67	1.10
SI16/S	1.54							

TAB. 9 – Accélération (à 50 MHz) par rapport à la version scalaire pour StratixII

directe, au lieu d'un seul accès en SIMD). Pour le flot optique, les accès concernent des mots de 16 bits, et il y a deux fois plus de défauts. Pour ces programmes, il faudrait utiliser un déroulage de boucle d'ordre 4 (resp. 2) pour obtenir les mêmes performances scalaires pour N=256 et N=260. La différence essentielle entre les versions Cyclone et StratixII est l'amélioration de la multiplication entière (blocs DSP) dans la version StratixII, qui conduit à une multiplication en 1 cycle pour StratixII. La DCT de JPEG, qui utilise les versions scalaires des flottants F16, le montre clairement. Alors que le gain est supérieur à 2 pour Cyclone avec des instructions monocycles, il reste proche de 1 pour StratixII. La version F21 présente un intérêt limité par rapport à la version F2, ce qui traduit le fait que les multiplications restent relativement peu nombreuses par rapport aux additions, y compris dans les benchmarks qui en utilisent le plus. Pour Grayscale, pour qui les versions entières peuvent éviter remplacer les multiplications et divisions en utilisant les décalages, les versions SIMD entières et flottantes sur 16 bits donnent des résultats équivalents avec des instructions F16 en 1 cycle. La version F1 utilise directement la multiplication en 1 cycle, alors que les multiplications par 255 ou 28 nécessitent plusieurs instructions.

3.3. Caractérisations des différentes configurations.

Pour les différentes configurations, nous avons recherché la fréquence maximale de fonctionnement du processeur pour les deux kits considérés. La table 10 résume les informations pour le kit Cyclone pour les différentes configurations : fréquence maximale de fonctionnement, ressources matérielles utilisées (exprimées en pourcentage du nombre de composants disponible) et la puissance consommé à la fréquence de fonctionnement maximale (avec l'hypothèse déjà indiquée d'un pourcentage d'excitation des signaux de 25%). La table 11 donne les mêmes informations pour le kit StratixII.

Par rapport à la configuration scalaire, la configuration Cyclone utilisant les instructions flottantes 1

Cyclone	Fmax (MHz)	Pd statique (mW)	Pd dyn. (mW)	Pd totale (mW)	% éléments logiques	% bits mémoire
Scalaire	105	129.5	572.0	701.5	15	23
SI16	105	129.5	638.9	768.4	19	23
F2	78	128.6	487.7	616.3	25	23
F21	55	127.7	343.3	471.0	23	23
F1	52	127.6	344.6	472.2	31	23

TAB. 10 – Caractérisation des configurations Cyclone

StratixII	Fmax (MHz)	Pd statique (mW)	Pd dyn (mW)	Pdd total (mW)	% ALUTS	% bits mémoire	% blocs DSP 9 bits
Scalaire	150	799	504	1303	5	2	8
SI16	140	798	484	1282	6	2	8
F2	130	800	507	1307	7	2	12
F21	92	793	353	1146	7	2	12
F1	80	792	330	1123	8	2	12
F2D	130	803	575	1378	9	2	12
F21D	92	795	386	1182	9	2	12
F1D	80	796	404	1378	11	2	12

TAB. 11 – Caractérisation des configurations StratixII

	Cyclone	StratixII
F2/S	0.74	0.87
F21/S	0.52	0.61
F1/S	0.50	0.53
SI16/S	1.00	0.93

TAB. 12 – Rapport des fréquences maximales d'exécution par rapport à la version scalaire

cycle double le nombre d'éléments logiques utilisés pour atteindre 31% de la quantité disponible. Pour StratixII, le pourcentage d'ALUTS passe de 5% à 8%, ce qui reste très faible.

3.4. Performances en temps d'exécution.

La table 12, dérivée des tables 10 et 11, donne le rapport des fréquences maximales d'exécution entre les différentes configurations par rapport à la fréquence maximale de la version scalaire.

La combinaison des performances en CPP (tables 8 et 9) et des rapports de fréquence (table 12) permet d'établir les accélérations (en temps réel d'exécution) pour les différentes configurations à fréquence maximale par rapport à la version scalaire à fréquence maximale (tables 13 et 14).

On constate que les versions SIMD F16 ont des performances inférieures pour Grayscale par rapport au SIMD entiers 16 bits pour des raisons évidentes (la fréquence de fonctionnement de la version F1 est très inférieure à celle de SI16). Pour la FDCT de JPEG, pour laquelle on utilise une version scalaire des F16, il n'y a gain qu'avec Cyclone, à cause de la latence importante de la multiplication entière. Comme déjà indiqué, la version F21 est moins bonne que les versions F2 et F1. Pour tous les autres benchmarks, les versions F1 sont toujours meilleures que la version scalaire avec Cyclone. Avec StratixII, les versions scalaires sont meilleures lorsque N n'est pas une puissance de 2 (ou lorsqu'un déroulage de boucle des versions scalaires conduit à un seul accès par mot de 32 bits, comme pour les versions SIMD). Avec Harris et le flot optique, le rapport reste assez proche de 1 (0,99 et 0,89).

Cyclone	N=256	N=256	N=256	N=256	N=260	N=256	N=260	N=64
	Grayscale	Gradient	Lisseur	Harris	Harris	Flot	Flot	FDCT
F2/S	0.74	0.91	2.48	1.13	0.98	1.25	0.99	1.00
F21/S		0.64	1.94	0.94	0.84	0.93	0.74	0.78
F1/S	0.77	1.14	2.69	1.17	1.11	1.32	1.09	1.48
SI16/S	1.53							

TAB. 13 – Accélération des temps d'exécution pour Cyclone

StratixII	N=256	N=256	N=256	N=256	N=260	N=256	N=260	N=64
	Grayscale	Gradient	Lisseur	Harris	Harris	Flot	Flot	FDCT
F2/S	0.84	1.07	1.45	1.14	0.95	1.20	0.87	0.43
F21/S		0.76	1.14	0.95	0.81	0.90	0.66	0.34
F1/S	0.83	1.22	1.47	1.09	0.99	1.17	0.89	0.59
SI16/S	1.44							

TAB. 14 – Accélération des temps d'exécution pour StratixII

	% d'activation	F2/S	F21/S	F1/S	SI16/S	F2D/S	F21D/S	F1D/S
Cyclone	12.5%	0.87	0.68	0.67	1.07	0.93	0.72	0.71
Cyclone	25%	0.88	0.67	0.67	1.10	0.95	0.72	0.72
Cyclone	50%	0.89	0.66	0.67	1.12	0.97	0.72	0.72
StratixII	12.5%	1.00	0.91	0.90	0.99	1.04	0.93	1.02
StratixII	25%	1.00	0.88	0.86	0.98	1.06	0.91	0.92
StratixII	50%	1.01	0.84	0.82	0.98	1.09	0.88	0.90

TAB. 15 – Rapport des puissances consommées par les différentes configurations en fonction du pourcentage d'activation des signaux dans l'ensemble processeur + instructions ajoutées

3.5. Energie consommée.

L'énergie consommée est le produit de la puissance dissipée par le temps d'exécution. Nous rappelons ici que la puissance dissipée par les différentes configurations est une estimation (avec un taux d'excitation des signaux de 25%) alors que tous nos autres résultats correspondent à des mesures.

Les rapports de puissance dissipée, déduits des tables 10 et 11 sont donnés dans la table 15. On constate que les rapports de consommation restent très voisins lorsque le pourcentage d'activation des signaux varie de 12,5% à 50%. Dans la suite, nous utiliserons un pourcentage d'activation de 25% pour comparer les rapports entre les différentes configurations pour l'énergie consommée et le produit délai x énergie.

Des rapports de temps d'exécution (tables 13 et 14) et des rapports de puissance, on peut déduire le rapport des énergies consommées entre les différentes configurations. Ils sont donnés dans les tables 16 et 17.

A nouveau, la version SIMD entière 16 bits s'impose pour Grayscale. Les versions F16 n'ont d'intérêt avec la FDCT qu'avec Cyclone, pour compenser la lenteur des multiplications entières. A nouveau, la version F1 est la plus intéressante. Elle permet des gains d'énergie très significatifs avec tous les benchmarks pour Cyclone. Pour StratixII, les gains sont moins significatifs, notamment lorsqu'on évite le problème des défauts de cache de conflit liés à la correspondance directe avec $N = 2^k$. La table 18 résume pour Cyclone et StratixII les rapports des produits Energie * Délai pour F1 par rapport à la version scalaire. Les résultats obtenus sont cohérents avec ceux de la consommation d'énergie, ce qui montre que la réduction de consommation ne se traduit pas par une dégradation des temps d'exécution.

Cyclone	N=256	N=256	N=256	N=256	N=260	N=256	N=260	N=64
	Grayscale	Gradient	Lisseur	Harris	Harris	Flot	Flot	FDCT
F2/S	1.18	0.96	0.35	0.78	0.89	0.79	1.09	0.88
F21/S		1.05	0.35	0.72	0.80	0.79	1.09	0.86
F1/S	0.87	0.59	0.25	0.58	0.61	0.61	0.80	0.46
SI16/S	0.71							

TAB. 16 – Rapport des énergies consommées pour Cyclone (<1 est meilleur pour les versions F)

StratixII	N=256	N=256	N=256	N=256	N=260	N=256	N=260	N=64
	Grayscale	Gradient	Lisseur	Harris	Harris	Flot	Flot	FDCT
F2/S	1.19	0.94	0.69	0.88	1.06	0.88	1.21	2.34
F21/S		1.16	0.77	0.92	1.08	1.01	1.38	2.60
F1/S	1.03	0.70	0.59	0.79	0.87	0.79	1.03	1.46
SI16/S	0.68							

TAB. 17 – Rapport des énergies consommées pour StratixII (<1 est meilleur)

FPGA		N=256	N=256	N=256	N=256	N=260	N=256	N=260	N=64
		Grayscale	Gradient	Lisseur	Harris	Harris	Flot	Flot	FDCT
Cyclone	F1/S	1.13	0.52	0.09	0.49	0.55	0.46	0.74	0.31
StratixII	F1/S	1.24	0.58	0.40	0.73	0.89	0.67	1.16	2.48

TAB. 18 – Rapport des produits « Energie x Délai » entre la version F1 et la version scalaire (<1 est meilleur pour F1)

4. Conclusion

Nous avons montré l'intérêt de l'utilisation d'instructions SIMD 16 bits à la fois pour diminuer les temps d'exécution et la puissance dissipée sur un certain nombre de benchmarks représentatifs du traitement d'images, notamment ceux utilisés en stabilisation d'images comme les algorithmes de recherche des points d'intérêt et de flot optique. Ces instructions sont facilement utilisables dans les processeurs « soft core » permettant la « spécialisation » des instructions. Les instructions flottantes 16 bits les plus efficaces sont celles qui s'exécutent en un cycle. Elles sont intéressantes à la fois pour les versions « moyenne » (Cyclone) ou « haut de gamme » (StratixII) des FPGA d'Altera. L'impact est plus grand sur les versions moins performantes (et moins chères) des FPGA. L'intérêt des instructions flottantes 16 bits augmenterait considérablement avec une interface 64 bits pour les instructions spécialisables. 4 opérations par instruction SIMD serait alors possible, sans une augmentation très importante des ressources utilisées et de la consommation. Ceci nécessiterait une modification significative du jeu d'instructions du Nios II, avec des instructions d'accès mémoire sur 64 bits. Des travaux similaires à ceux décrits dans cet article sont en cours avec le processeur Xtensa de Tensilica pour lequel des registres 128 bits sont disponibles, permettant de définir des instructions SIMD aussi larges que celles qui existent dans les extensions SSE et Altivec des processeurs généralistes.

Bibliographie

1. G. Kolli. « Using Fixed-Point Instead of Floating-Point for Better 3D Performance », Document technique, Intel Optimizing Center
2. Disponible à l'adresse : `http://www.devx.com/Intel/article/16478`, juillet 2003.
3. D. Etiemble, L. Lacassagne, "16-bit FP sub-word parallelism to facilitate compiler vectorization and improve performance of image and media processing", in Proceedings ICPP 2004, Montreal, Canada
4. D. Etiemble et L. Lacassagne, "Des flottants 16 bits sur microprocesseurs d'usage général pour images et multimédia", Technique et science informatiques, Vol 24, n°6, pp 645-665, 2005
5. D. Etiemble, S. Bouaziz and L. Lacassagne, "Customizing 16-bit floating-point instructions on a Nios II processor for FPGA image and media processing", in IEEE Workshop on Embedded Systems for Real Time Media Processing (Estimedia), Jersey City, September 2005.
6. N. Drach et J. Sebot, SIMD ISA Extensions : Tradeoff between Power Consumption and Performance on a Superscalar Processor, Kool Chips Workshop, Micro 33 (Monterey CA), December 2000
7. Julien Sebot et Nathalie Drach, SIMD extensions : Reducing Power Consumption on a Superscalar Processor for Multimedia Applications, Cool Chips IV (Tokyo Japan), April 2001
8. Code et ensemble des résultats, `http://www.lri.fr/~de/F16/SympAA'06.pdf`
9. C. Lee, M. Potkonjak, W.H. Mongione-Smith, "Mediabench : A Tool for Evaluating and Synthesizing Multi-

media and Communication Systems", Proceeding Micro-30 conference, Research Triangle Park, NC, December 1995.
10. Altera, "Nios Custom Instructions, Tutorial", June 2002, http://www.altera.com/literature/tt/tt_nios_ci.pdf
11. J. Detrey and F. De Dinechin, "A VHDL Library of Parametrisable Floating-Point and LSN Operators for FPGA", http://perso.ens-lyon.fr/jeremie.detrey/FPLibrary/