Applications Opérateur de DCT

Nous faisons dans ce chapitre un récapitulatif des résultats en termes de surface et de chaîne longue que nous avons obtenus en appliquant nos différents algorithmes à plusieurs circuits arithmétiques. Ces résultats sont également comparés à ceux des architectures classiques de façon à montrer les améliorations obtenues.

Toutes les architectures présentées ont été obtenues avec l'utilisation de la bibliothèque de cellules précaractérisées de la chaîne Alliance [GP92] (en 0.35μm) et les outils de placement/routage/analyse de timing de Cadence : **Encounter**.

6.1 Filtre

Différentes architectures de filtre *FIR* (Finite Impulse Filter) existent [dBNNC03, dBNN04, NMDT04] ; nous nous intéressons particulièrement à celle présentée dans la Figure 6.1.

Comme on peut le voir, cette architecture ne contient ni soustraction ni opérateur avec sorties multiples. Les différents algorithmes s'utilisent donc avec uniquement la notation Carry-Save (et sans pré-traitement nécessaire) et sans une gestion spécifique en ce qui concerne les sorties multiples.

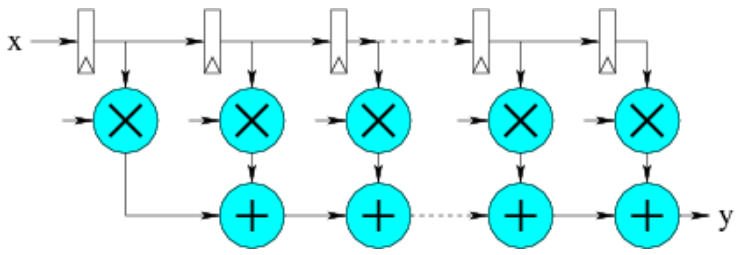
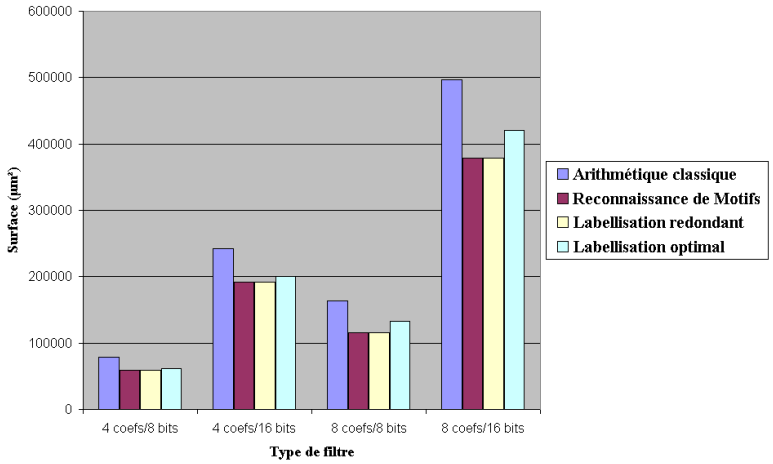


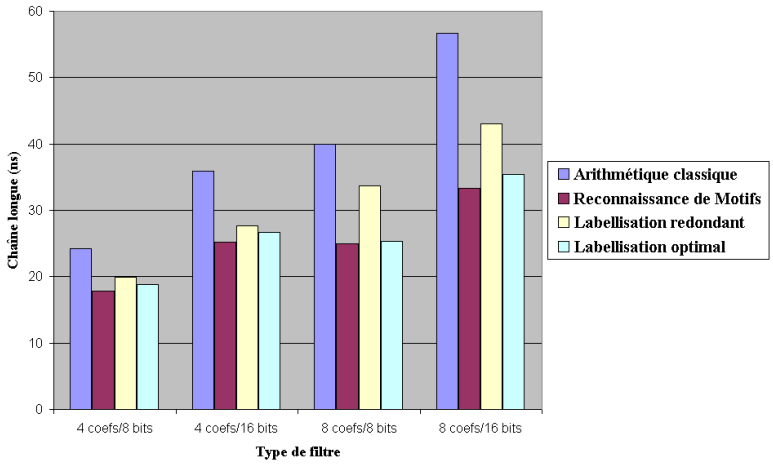
FIG. 6.1 – Architecture d'un filtre

Les résultats obtenus sont présentés dans les Figures 6.2 à 6.4. Ils résultent de l'optimisation de quatre versions différentes du filtre : 4 coefficients, valeurs sur 8 bits, 4 coefficients, valeurs sur 16 bits, 8 coefficients, valeurs sur 8 bits, 8 coefficients, valeurs sur 16 bits.



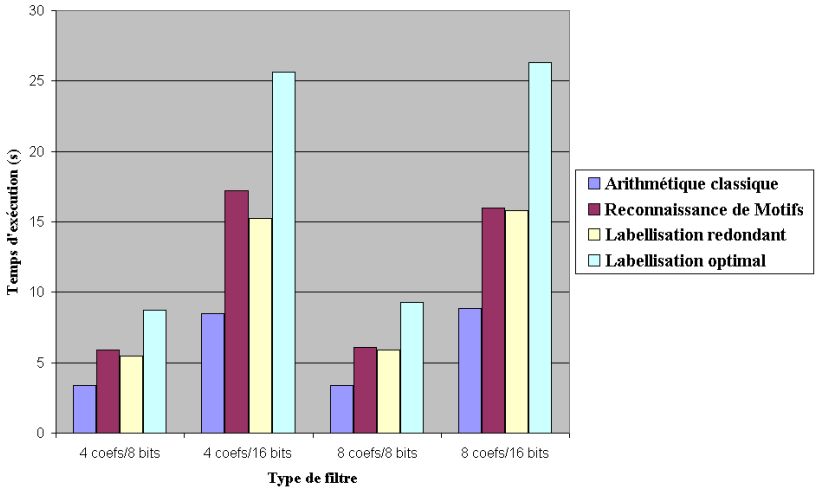
	Arithmétique Classique	Reconnaissance de Motifs		Labellisation Redondant		Labellisation Optimal	
	μm^2	μm^2	%	μm^2	%	μm^2	%
4coef/8bits	79 012	58 469	-26%	58 469	-26%	62 027	-21.5%
4coef/16bits	242 799	192 055	-20.9%	192 055	-20.9%	200 535	-17.4%
8coef/8bits	162 912	114 978	-29.4%	114 978	-29.4%	132 398	-18.7%
8coef/16bits	497 117	378 555	-23.8%	378 555	-23.8%	421 093	-15.3%

FIG. 6.2 – Résultats du filtre : Surface



	Arithmétique Classique	Reconnaissance de Motifs		Labellisation Redondant		Labellisation Optimal	
	ns	ns	%	ns	%	ns	%
4coef/8bits	24.27	17.88	-26.3%	19.93	-17.9%	18.83	-22.4%
4coef/16bits	35.89	25.23	-29.7%	27.71	-22.8%	26.62	-25.8%
8coef/8bits	40.02	24.99	-35.6%	33.71	-15.8%	25.35	-36.7%
8coef/16bits	56.68	33.34	-41.2%	43.07	-24%	35.41	-37.5%

FIG. 6.3 – Résultats du filtre : Chaîne longue



	Arithmétique Classique	Reconnaissance de Motifs		Labellisation Redondant		Labellisation Optimal	
	s	s	%	s	%	s	%
4coef/8bits	3.4	5.92	+74.1%	5.48	+61.2%	8.73	+156.7%
4coef/16bits	8.47	17.2	+103%	15.23	+79.8%	25.64	+202.7%
8coef/8bits	3.4	6.09	+79.1%	5.92	+74.1%	9.26	+172.3%
8coef/16bits	8.84	15.96	+80.5%	15.77	+78.4%	26.31	+197.6%

FIG. 6.4 – Résultats du filtre : Temps d'exécution

Ces résultats montrent tout d'abord que tous les algorithmes améliorent la surface et la chaîne longue, par rapport à une implantation en arithmétique classique. On remarque ensuite que chacun des algorithmes donne un résultat différent :

- d'un point de vue surface, ce sont les deux algorithmes *redondant dès que possible* qui fournissent le meilleur résultat,
- d'un point de vue chaîne longue, c'est la reconnaissance de motif qui donne le meilleur résultat, suivi de l'algorithme d'allocation optimals, puis de l'algorithme de labellisation redondant dès que possible.

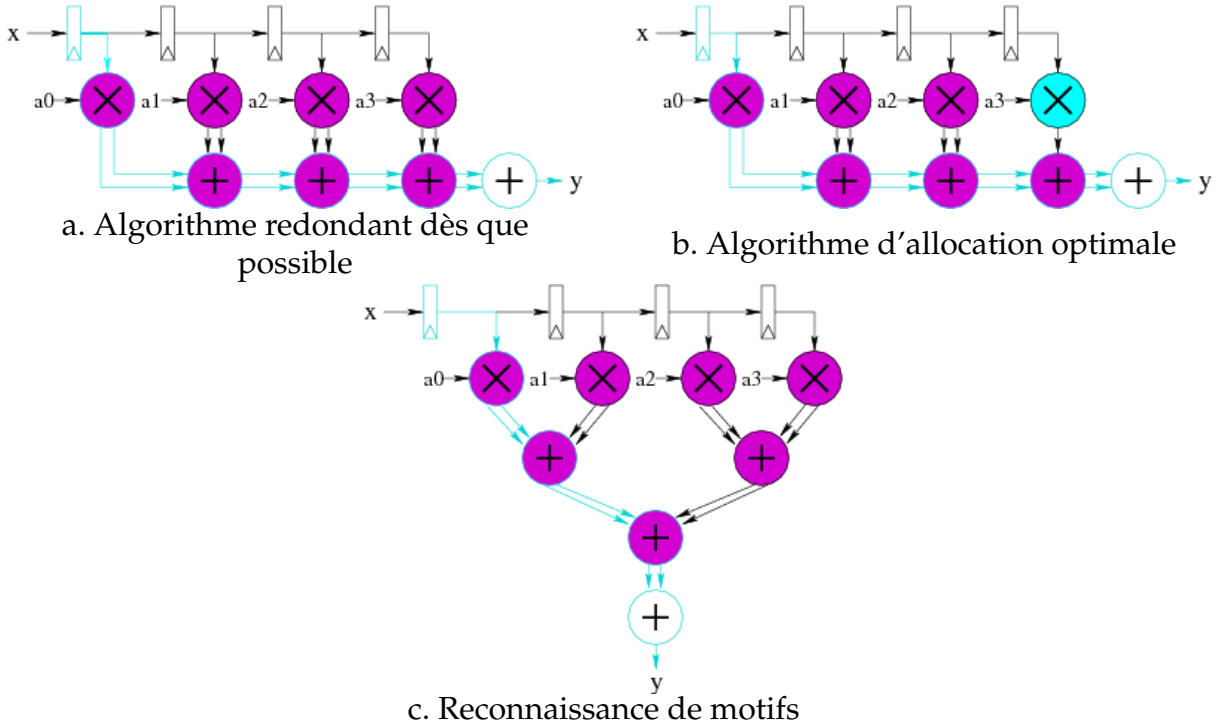
Cette différence est due au fait que les trois algorithmes ne génèrent pas la même architecture, comme montré dans la Figure 6.5 (exemple avec quatre coefficients).

Comparons tout d'abord les deux algorithmes de labellisation. L'algorithme d'allocation optimale améliore le délai au détriment de la surface par rapport à l'algorithme redondant dès que possible : en effet, le choix est fait de transformer uniquement trois multiplieurs dans leur version redondante (et donc laisser 1 multiplieur dans sa version classique, pour 4 coefficients, 5 pour 8 coefficients, ...), de façon à ce qu'au moins un additionneur de la chaîne longue (entre la sortie du premier registre et la sortie du circuit) soit un additionneur mixte et plus un additionneur redondant.

Cependant, le résultat obtenu par l'algorithme d'allocation optimale est moins performant que celui donné par la reconnaissance de motifs. En effet, alors que les algorithmes de labellisation "ne font que" modifier l'architecture choisie pour chaque

opérateur et pas la connectique, l'ensemble de motifs choisi exploite les propriétés de commutativité et d'associativité de l'addition pour effectuer de légères modifications à la connectique d'un circuit. Cette différence engendre qu'il y a au moins un additionneur de moins dans la chaîne longue de l'architecture générée par l'algorithme basé sur la reconnaissance de motifs.

Par ailleurs, notons que les améliorations sont plus importantes pour les circuits avec huit coefficients que pour ceux avec quatre coefficients. Autrement dit, plus la chaîne d'opérateurs arithmétiques est grande, meilleures sont les optimisations.



a. Algorithme redondant dès que possible

b. Algorithme d'allocation optimale

c. Reconnaissance de motifs

FIG. 6.5 – Architectures optimisées d'un filtre avec quatre coefficients

D'un point de vue temps d'exécution, nous pouvons voir que l'algorithme d'allocation optimale est plus lent que l'algorithme de labellisation redondant dès que possible, ce qui est en cohérence avec la théorie. Nous voyons également que la reconnaissance de motifs est elle à peine plus lente que l'algorithme redondant dès que possible.

6.2 Unité de calcul de distance

L'architecture d'une unité de calcul de distance (*DCU* : Distance Computation Unit) est composée de deux parties, comme montré dans la Figure 6.6 [DBM01]. La première effectue le calcul de la distance $(A_i - B_i)^2$, tandis que la seconde effectue la somme entre ces distances.

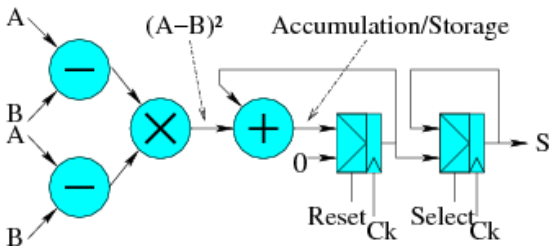


FIG. 6.6 – Architecture d'une DCU

Cet opérateur contenant des soustractions, nous présentons les résultats obtenus avec les différentes façons de prendre en compte cette opération. Sont donc comparées trois différentes gestions de la soustraction : l'utilisation des architectures Carry-Save uniquement, après transformation des soustractions en additions (*Carry-Save*), et l'utilisation des architectures Borrow Save, les additionneurs ayant une sortie Carry-Save (*Borrow-Save 1*) ou Borrow-Save (*Borrow-Save 2*).

Par ailleurs cette architecture ne contient pas d'opérateur avec sorties multiples. Nous utilisons donc la reconnaissance de motifs sans la gestion spécifique pour les sorties multiples.

Les résultats sont présentés dans les Figures 6.7 à 6.9. Ils résultent de l'optimisation d'une *DCU* avec les données en entrée sur 16 bits.

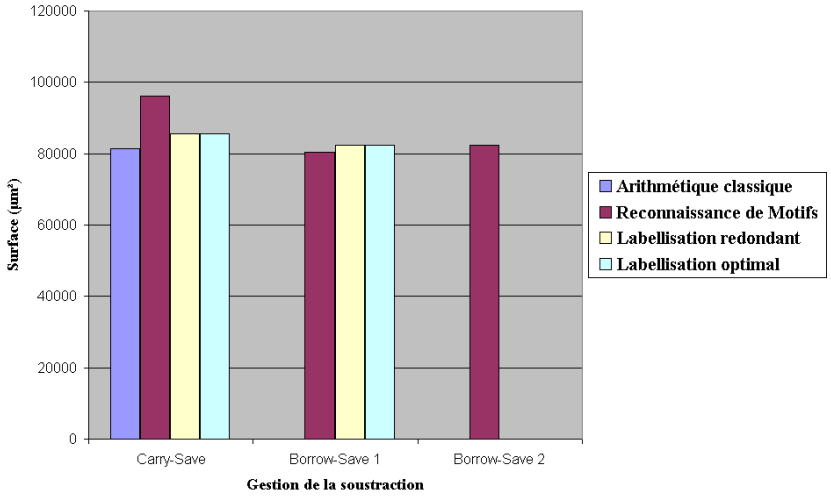
Ces résultats se résument à :

- dans le pire des cas, une légère détérioration de la surface, principalement pour l'optimisation Carry-Save,
- une amélioration de la chaîne longue pour tous les cas d'optimisation, avec de meilleures performances pour les optimisations Borrow-Save.

Ces résultats concordent avec notre postula de base disant que les architectures Borrow-Save sont plus appropriées que les architectures Carry-Save pour l'optimisation de circuits contenant des soustractions. En effet, l'utilisation d'architectures Borrow-Save dans ce cas permet l'utilisation d'un multiplieur redondant avec entrées Borrow-Save, alors que l'utilisation d'architectures Carry-Save uniquement oblige à effectuer une inversion sur l'entrée à soustraire et une addition à '1' avec un additionneur.

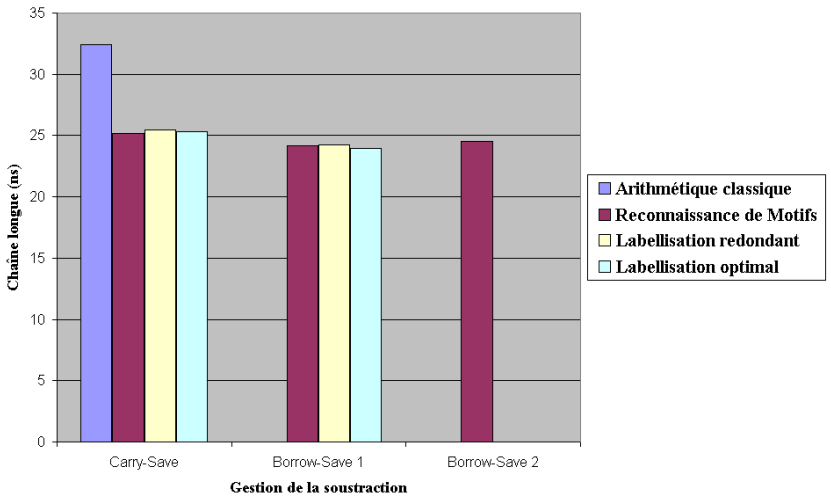
D'un point de vue temps d'exécution, on constate que l'algorithme de labellisation redondant dès que possible est le plus rapide des trois.

Dans l'article [DBM01], différentes implémentations d'une *DCU* ont été faites "à la main", en arithmétique classique et en arithmétique redondante. Les conclusions présentées se résument à une optimisation du délai allant de 10% à 15% pour un surcoût en surface limité à 17%. De notre côté, le délai est amélioré de 21.4% à 26.1% selon l'algorithme utilisé, avec une surface tantôt améliorée (-1.3%), tantôt dégradée (+5.2%). Nous pouvons en conclure que nos différents algorithmes produisent un meilleur délai couplé à un meilleur produit surface.délai.



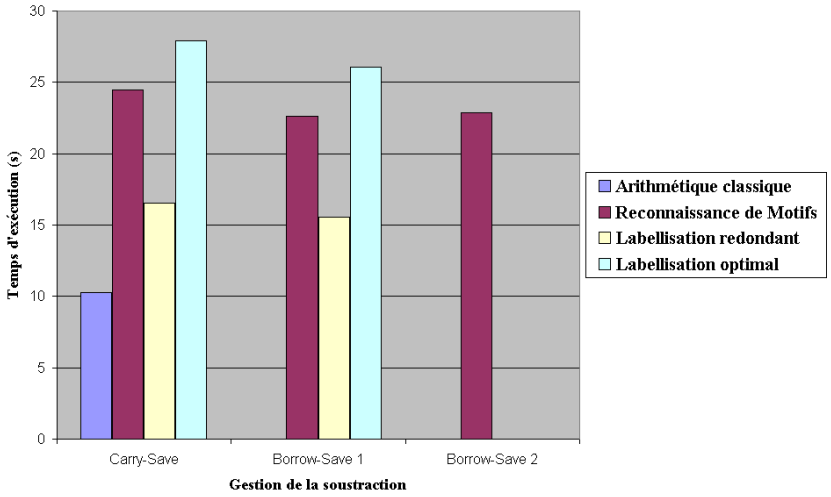
	Arithmétique classique	Reconnaissance de Motifs		Labellisation Redondant		Labellisation Optimal	
	μm^2	μm^2	%	μm^2	%	μm^2	%
Carry-Save	81 383	96 150	+18.1%	85 581	+5.1%	85 581	+5.1%
Borrow-Save 1	-	80 329	-1.3%	82 332	+1.2%	82 332	+1.2%
Borrow-Save 2	-	82 332	+1.2%	-	-	-	-

FIG. 6.7 – Résultats de la DCU : Surface



	Arithmétique classique	Reconnaissance de Motifs		Labelllisation Redondant		Labellisation Optimal	
	ns	ns	%	ns	%	ns	%
Carry-Save	32.43	25.18	-22.4%	25.48	-21.4%	25.32	-21.9%
Borrow-Save 1	-	24.14	-25.6%	24.24	-25.2%	23.97	-26.1%
Borrow-Save 2	-	24.55	-24.3%	-	-	-	-

FIG. 6.8 – Résultats de la DCU : Chaîne longue



	Arithmétique classique	Reconnaissance de Motifs		Labellisation Redondant		Labellisation Optimal	
	s	*s*	%	*s*	%	*s*	%
Carry-Save	9.62	23.12	+140.3%	15.46	+60.7%	25.79	+168.1%
Borrow-Save 1	-	23.04	+139.5%	14.12	+46.8%	24.71	+156.9%
Borrow-Save 2	-	24.15	+151%	-	-	-	-

FIG. 6.9 – Résultats de la DCU : Temps d'exécution

Différentes architectures de *DCT* (Discrete Cosinus Transform) existent. Nous nous sommes intéressés plus particulièrement au graphe de Loeffer [CLM89, DCM00]. Deux versions différentes de ce graphe ont été optimisées :

- une version avec 40 opérateurs arithmétiques (permettant d'obtenir tous les résultats en 1 cycle),
- une version avec 12 opérateurs arithmétiques (et un séquenceur) (permettant d'obtenir tous les résultats en 8 cycles).

Cet opérateur, quelque soit son implantation, possède de nombreux opérateurs avec sorties multiples, il est donc un bon exemple pour tester les trois comportements de la reconnaissance de motifs en cas de sorties multiples. Il contient également des soustractions, nous testons donc les trois façons de gérer cette opération.

6.3.1 Partition

L'architecture de la partition de la *DCT* est montrée dans la Figure 6.10.

Les résultats sont présentés dans les Figures 6.11 à 6.13. Ils proviennent de l'optimisation d'une *DCT* avec les données sur 8 bits, les coefficients sur 12 bits et les résultats sur 16 bits.

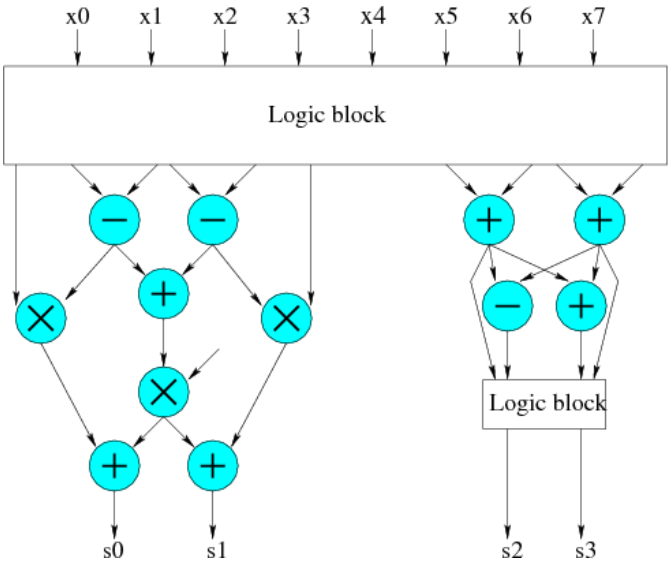


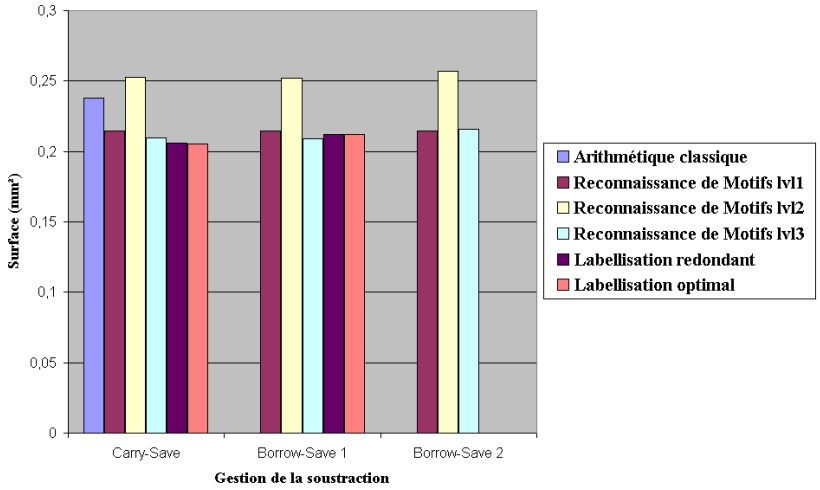
FIG. 6.10 – Partition de la DCT

En ce qui concerne la reconnaissance de motifs, ces résultats sont concordants avec la théorie : les optimisations sans et avec partage des ressources (*lvl2* et *lvl3*) donnent de meilleures chaînes longues que l'optimisation des arbres séparément (*lvl1*), tandis que l'optimisation des arbres séparément et l'optimisation avec partage des ressources donnent de meilleures surface que l'optimisation sans partage des ressources.

En ce qui concerne les algorithmes de labellisation, notons tout d'abord qu'ils donnent sensiblement les mêmes résultats. Le choix est donc fait par l'algorithme d'allocation optimale de transformer tous les opérateurs possibles en leur version redondante. Ces deux algorithmes donnent des résultats similaires aux meilleurs résultats fournis par la reconnaissance de motifs (proches de l'optimisation des arbres avec partage des ressources).

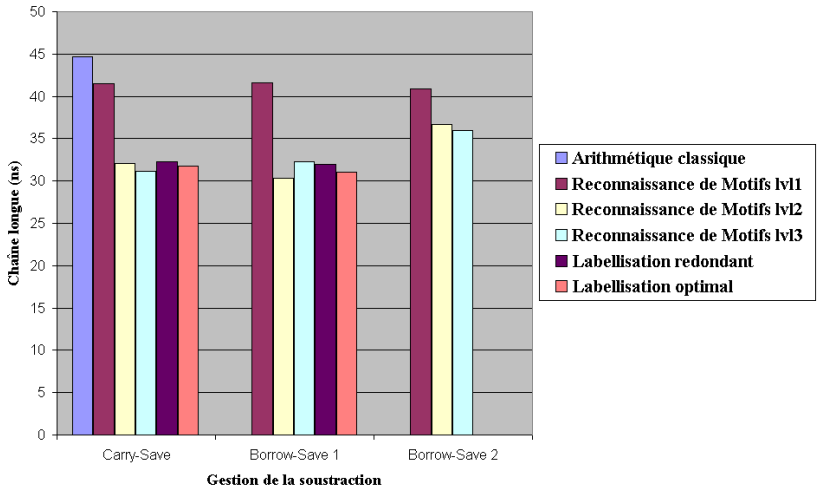
Par ailleurs, en ce qui concerne l'utilisation des architectures Borrow-Save, les résultats ne sont pas particulièrement plus performants qu'avec l'utilisation des architectures Carry-Save. Cela confirme le fait que le plus important est le contexte, et qu'il existe donc des cas pour lesquels les architectures Borrow-Save ne sont pas les plus optimales.

D'un point de vue temps d'exécution, on voit que l'algorithme de labellisation redondant dès que possible est, de loin, le plus rapide de tous. L'algorithme d'allocation optimale est lui le plus lent. Quant à la reconnaissance de motifs, elle est de plus en plus lente, plus l'option choisie en fonction des sorties multiples est complexe. En effet, là où seuls 3 motifs peuvent être remplacés quand on se limite à l'optimisation des arbres séparément (exemple de l'optimisation Carry-Save), l'optimisation sans partage de ressource permet d'en remplacer 12 (avec un surcoût de 9 instances dupliquées), ainsi que l'optimisation avec partage des ressources (qui contient également le post traitement permettant de fusionner 2 instances dupliquées inutilement).



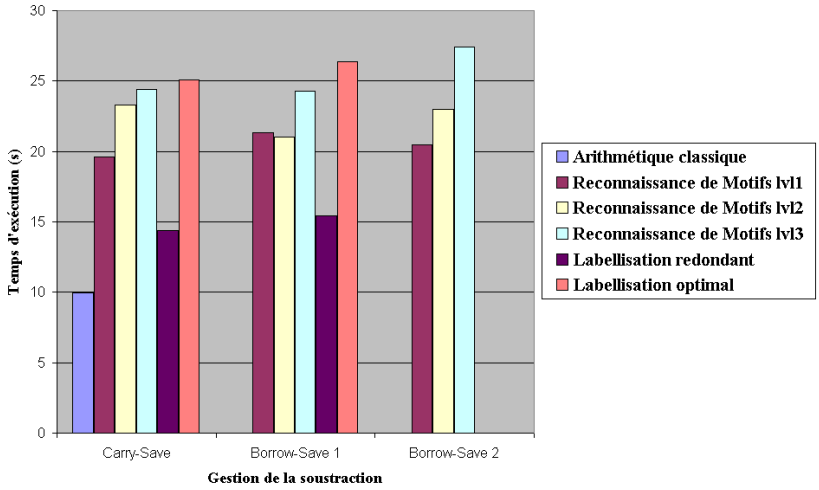
	Arithmétique classique	Reconnaissance de Motifs						Labellisation Redondant		Labellisation Optimal	
		Lvl1		Lvl2		Lvl3					
	mm^2	mm^2	%	mm^2	%	mm^2	%	mm^2	%	mm^2	%
CS	0.24	0.21	-12.5%	0.25	+4%	0.21	-12.5%	0.20	-16.7%	0.20	-16.7%
BS 1	-	0.21	-12.5%	0.25	+4%	0.21	-12.5%	0.21	-12.5%	0.21	-12.5%
BS 2	-	0.21	-12.5%	0.25	+4%	0.22	-8.3%	-	-	-	-

FIG. 6.11 – Résultats de la partition de la DCT : Surface



	Arithmétique classique	Reconnaissance de Motifs						Labellisation Redondant		Labellisation Optimal	
		Lvl1		Lvl2		Lvl3					
	ns	ns	%	ns	%	ns	%	ns	%	ns	%
CS	44.67	41.47	-7.2%	32.09	-28.2%	31.18	-30.2%	32.29	-27.7%	31.72	-29%
BS 1	-	41.6	-6.9%	30.3	-32.2%	32.24	-27.8%	31.92	-28.5%	31.04	-30.5%
BS 2	-	40.91	-8.4%	36.63	-18%	35.91	-19.6%	-	-	-	-

FIG. 6.12 – Résultats de la partition de la DCT : Chaîne longue



	Arithmétique classique	Reconnaissance de Motifs						Labellisation Redondant		Labellisation Optimal	
		Lvl1		Lvl2		Lvl3					
	s	*s*	%	*s*	%	*s*	%	*s*	%	*s*	%
CS	9.95	19.6	+97%	23.3	+134%	24.4	+145%	14.4	+45%	25.1	+152%
BS 1	-	21.35	+114%	21	+111%	24.3	+144%	15.4	+55%	26.4	+165%
BS 2	-	20.5	+106%	23	+131%	27.4	+175%	-	-	-	-

FIG. 6.13 – Résultats de la partition de la DCT : Temps d'exécution

Dans l'article [DCM00], différentes implémentations d'une *DCT* 2 dimensions (la combinaison de deux *DCT* 1 dimension avec adaptation des parties opératives) ont été faites "à la main", en arithmétique classique et en arithmétique redondante. Les conclusions présentées montrent une optimisation du délai de 20% et de la surface de 3%.

De notre côté, le délai est amélioré jusqu'à 32.2% selon l'algorithme utilisé, avec une surface tantôt améliorée (jusqu'à -16.7%), tantôt dégradée (+4%).

Nous pouvons en conclure que nos différents algorithmes permettent, selon le cas, d'optimiser de façon plus performante, l'un ou l'autre des deux critères.

6.3.2 Opérateur complet

L'architecture de l'opérateur complet de la *DCT* est montrée dans la Figure 6.14.

Les résultats sont présentés dans les Figures 6.15 à 6.17. Ils proviennent de l'optimisation d'une *DCT* avec les données sur 8 bits, les coefficients sur 14 bits et les résultats sur 16 bits.

En ce qui concerne la surface, tous les algorithmes sauf l'optimisation sans partage des ressources de la reconnaissance de motifs donnent une surface équivalente à celle d'origine. L'optimisation sans partage des ressources conduit elle, dans le pire des cas, à une surface preque doublée.

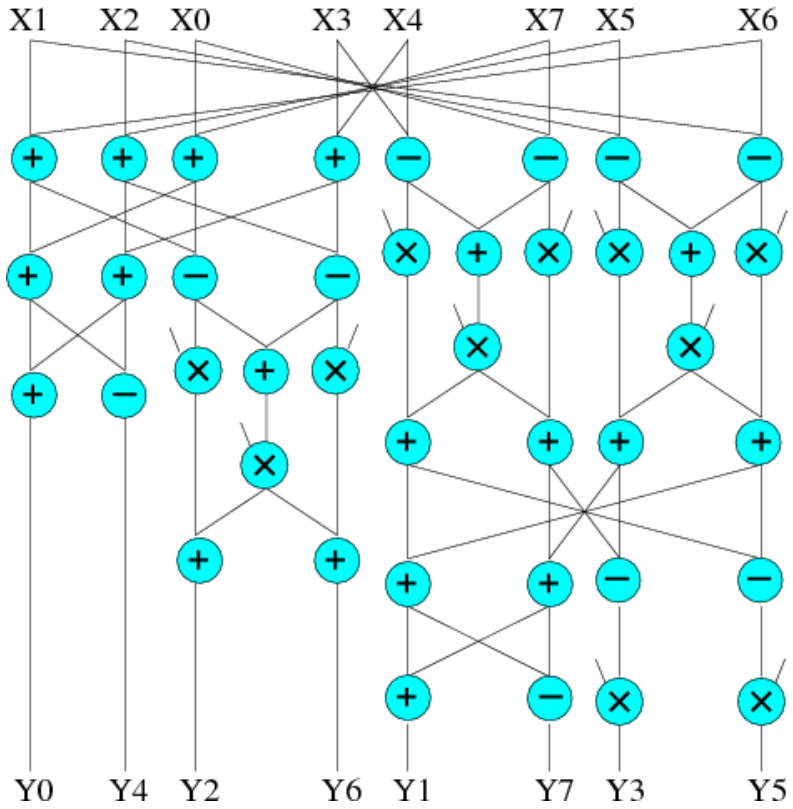


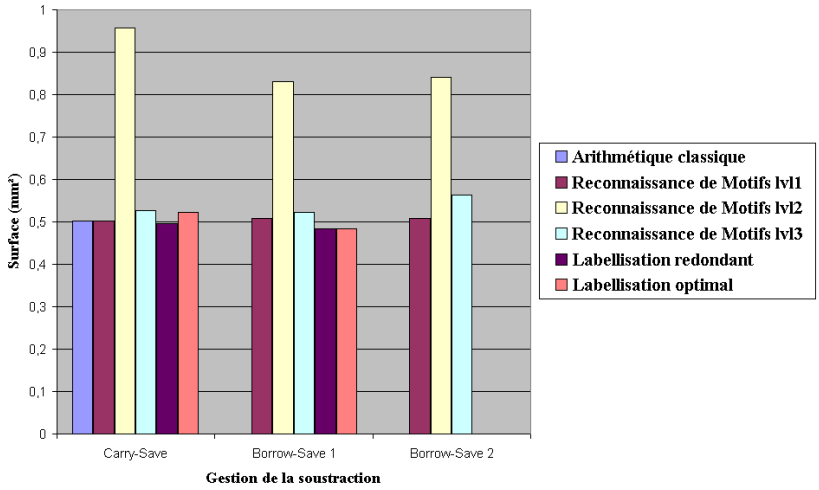
FIG. 6.14 – Opérateur de DCT

En ce qui concerne le délai, tous les algorithmes conduisent à une optimisation plus ou moins importante, les meilleurs résultats provenant des algorithmes de labellisation, et de l'optimisation avec partage des ressources.

Par aileurs, les résultats des algorithmes de labellisation sont similaires ; en effet, le choix est fait lors de l'algorithme d'allocation optimale de passer tous les opérateurs arithmétiques possibles en leur version redondante.

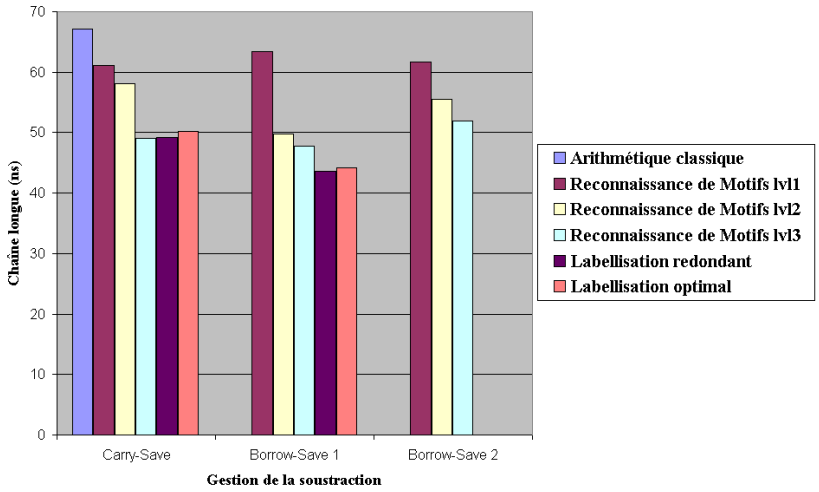
En ce qui concerne la gestion de la soustraction, on peut voir que dans ce cas, l'utilisation d'architectures Borrow-Save (avec additionneurs avec sorties Carry-Save) mène aux meilleures performances.

D'un point de vue temps d'exécution, c'est cette fois l'optimisation avec partage des ressources qui est la plus coûteuse. Cela est dû au post traitement visant à fusionner les instances inutilement dupliquées, qui devient de plus en plus coûteux plus le circuit sur lequel on travaille est grand (72 motifs remplacés au lieu de 11, 80 instances dupliquées et 33 instances fusionnées). Les quatre autres algorithmes ont un temps à peu près équivalent (avec cependant toujours l'algorithme d'allocation optimale un peu plus lent que l'algorithme redondant dès que possible, et l'optimisation sans partage des ressources un peu plus lente que l'optimisation des arbres séparément).



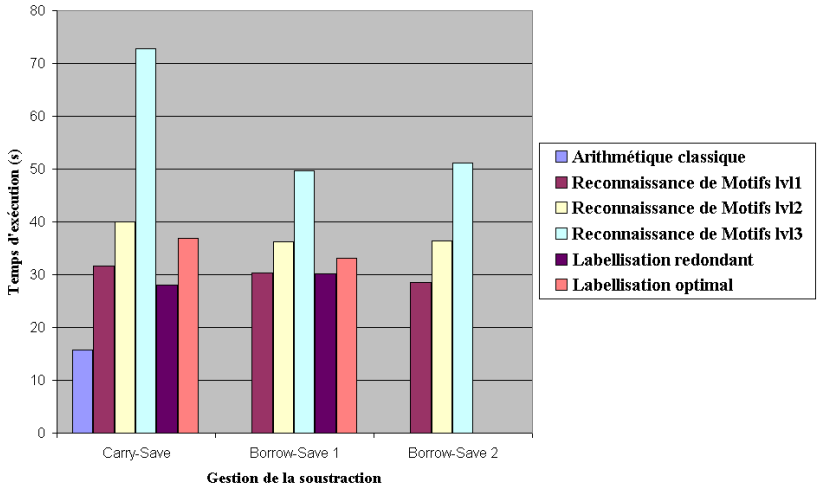
	Arithmétique classique	Reconnaissance de Motifs						Labellisation Redondant		Labellisation Optimal	
		Lvl1		Lvl2		Lvl3					
	mm^2	mm^2	%	mm^2	%	mm^2	%	mm^2	%	mm^2	%
CS	0.50	0.50	0%	0.96	+92%	0.53	+6%	0.50	0%	0.52	+4%
BS 1	-	0.51	+2%	0.83	+66%	0.52	+4%	0.48	-4%	0.48	-4%
BS 2	-	0.51	+2%	0.84	+68%	0.56	+12%	-	-	-	-

FIG. 6.15 – Résultats de la DCT complète : Surface



	Arithmétique classique	Reconnaissance de Motifs						Labellisation Redondant		Labellisation Optimal	
		Lvl1		Lvl2		Lvl3					
	ns	*ns*	%	*ns*	%	*ns*	%	*ns*	%	*ns*	%
CS	67.06	61.04	-9%	58.11	-13.3%	49.09	-26.8%	49.18	-26.7%	50.23	-25.1%
BS 1	-	63.39	-5.5%	49.71	-25.9%	47.83	-28.7%	43.57	-35%	44.21	-34.1%
BS 2	-	61.75	-7.9%	55.50	-17.2%	51.99	-22.5%	-	-	-	-

FIG. 6.16 – Résultats de la DCT complète : Chaîne longue



	Arithmétique classique	Reconnaissance de Motifs						Labellisation Redondant		Labellisation Optimal	
		Lvl1		Lvl2		Lvl3					
	s	*s*	%	*s*	%	*s*	%	*s*	%	*s*	%
CS	15.8	31.6	+100%	40.1	+153.8%	72.8	+360.7%	28	+77.2%	36.9	+133.5%
BS 1	-	30.3	+91.8%	36.2	+129.1%	49.6	+213.9%	30.1	+90.5%	33.1	+109.5%
BS 2	-	28.5	+80.4%	36.4	+130.4%	51.2	+224%	-	-	-	-

FIG. 6.17 – Résultats de la DCT complète : Temps d'exécution

6.4 Transformée de Fourier

6.4.1 le "papillon"

Le "papillon" est l'élément de base de l'opérateur de la transformée de Fourier (notée *FFT* pour *Fast Fourier Transform*). Son architecture est présentée dans la Figure 6.18.

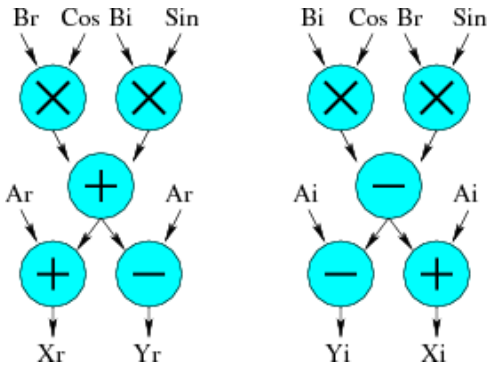


FIG. 6.18 – Architecture d'un papillon FFT

Cet opérateur contient des soustractions, nous présentons donc les résultats obtenus avec les différentes façon de gérer cet opérateur. Il possède également des opéra-

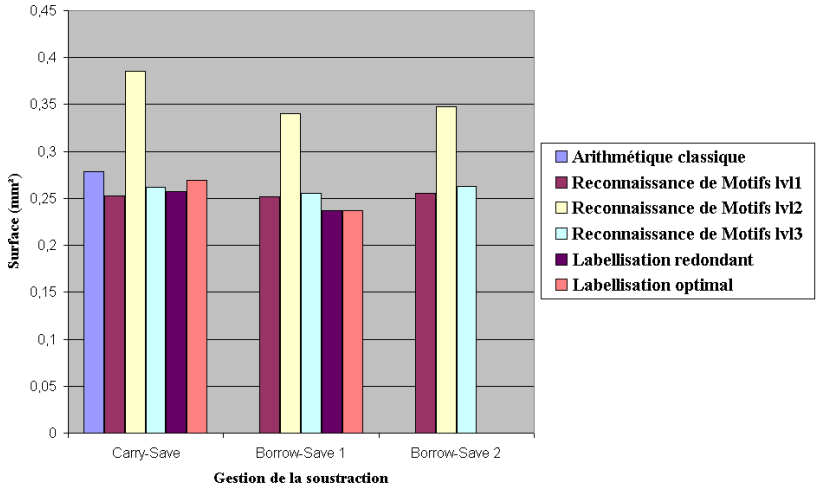
teurs possédant des sorties multiples. Nous présentons donc les différents comportements possibles avec la reconnaissance de motif.

Les résultats sont présentés dans les Figures 6.19 à 6.21. Il proviennent de l'optimisation d'un papillon avec les données sur 16 bits et les résultats sur 32 bits.

En ce qui concerne la surface, il y a une légère amélioration, sauf pour l'optimisation sans partage des ressources. Par ailleurs, les performances sont meilleures avec l'utilisation des architectures utilisant la notation Borrow-Save.

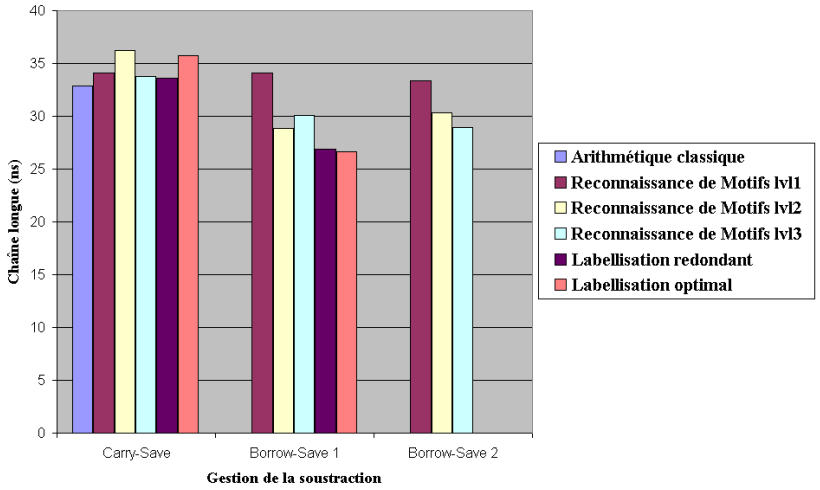
Cela est encore plus prononcé en ce qui concerne la chaîne longue : cet opérateur est un exemple typique montrant une dégradation de la chaîne longue avec l'utilisation des architectures Carry-Save, et une bonne optimisation dès que des architectures Borrow-Save sont utilisées.

D'un point de vue temps d'exécution, les conclusions sont les mêmes que pour tous les autres opérateurs du même ordre de grandeur : c'est l'algorithme d'allocation optimale qui est le plus lent, l'algorithme de labellisation redondant dès que possible est le plus rapide, et la reconnaissance de motifs entre les deux, étant de plus en plus lente en fonction de la gestion des sorties multiples.



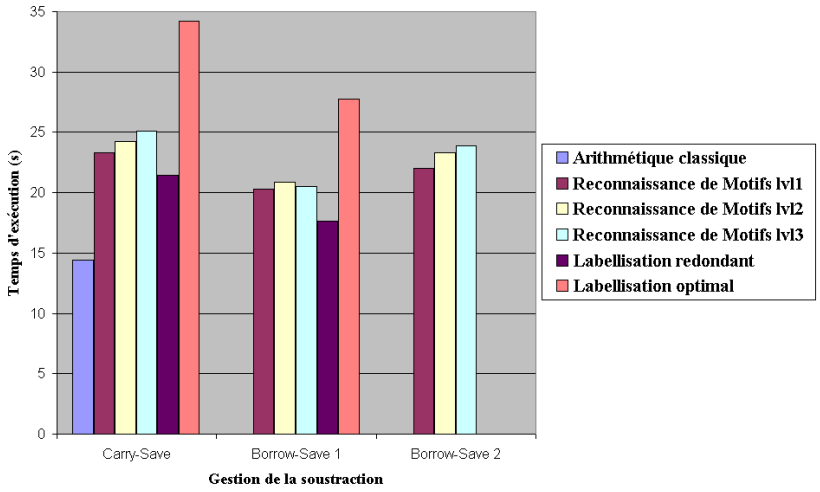
	Arithmétique classique	Reconnaissance de Motifs						Labellisation Redondant		Labellisation Optimal	
		Lvl1		Lvl2		Lvl3					
	mm^2	mm^2	%	mm^2	%	mm^2	%	mm^2	%	mm^2	%
CS	0.28	0.25	-10.7%	0.39	+39.3%	0.26	-7.1%	0.26	-7.1%	0.27	-3.6%
BS 1	-	0.25	-10.7%	0.34	+21.4%	0.25	-10.7%	0.24	-14.3%	0.24	-14.3%
BS 2	-	0.25	-10.7%	0.35	+25%	0.26	-7.1%	-	-	-	-

FIG. 6.19 – Résultats du papillon : Surface



	Arithmétique classique	Reconnaissance de Motifs						Labellisation Redondant		Labellisation Optimal	
		Lvl1		Lvl2		Lvl3					
	ns	*ns*	%	*ns*	%	*ns*	%	*ns*	%	*ns*	%
CS	32.85	34.06	+3.7%	36.27	+10.4%	33.79	+2.9%	33.59	+2.2%	35.7	+8.7%
BS 1	-	34.11	+3.8%	28.83	-12.2%	30.06	-8.5%	26.92	-18%	26.67	-18.8%
BS 2	-	33.39	+1.6%	30.34	-7.6%	28.94	-11.9%	-	-	-	-

FIG. 6.20 – Résultats du papillon : Chaîne longue



	Arithmétique classique	Reconnaissance de Motifs						labellisation Redondant		Labellisation Optimal	
		Lvl1		Lvl2		Lvl3					
	s	*s*	%	*s*	%	*s*	%	*s*	%	*s*	%
CS	14.44	23.28	+61.2%	24.27	+68.1%	25.12	+74%	21.44	+48.5%	34.21	+136.9%
BS 1	-	20.28	+40.4%	20.84	+44.3%	20.54	+42.2%	17.62	+22%	27.73	+92%
BS 2	-	22.02	+52.5%	23.34	+61.6%	23.87	+65.3%	-	-	-	-

FIG. 6.21 – Résultats du papillon : Temps d'exécution

6.4.2 FFT

A partir de l'élément de base qu'est le papillon, différentes architectures existent de façon à obtenir un opérateur *FFT*. Chacune de ces architectures correspond à un algorithme particulier. Parmi les algorithmes existants, seize ont été analysés dans [Meh91], répartis en deux catégories : les algorithmes à entrelacement temporel et les algorithmes à entrelacement fréquentiel.

Nous nous sommes intéressés plus particulièrement à un des algorithmes à entrelacement temporel, dont l'architecture est montrée dans la Figure 6.22 (Version avec 8 entrées/sorties). Cet algorithme est un algorithme à géométrie constante. Les accès nœuds sources et nœuds puits sont séquentiels. Les entrées sont désordonnées, les sorties et les coefficients ordonnés.

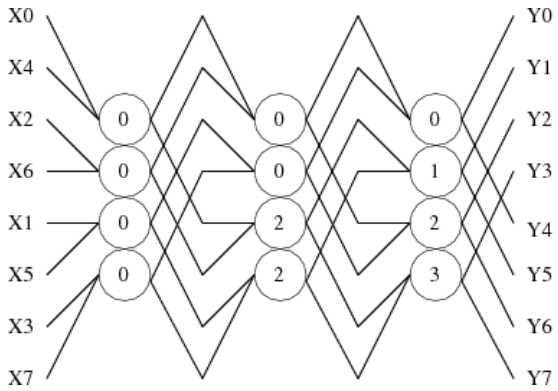


FIG. 6.22 – Architecture d'une FFT

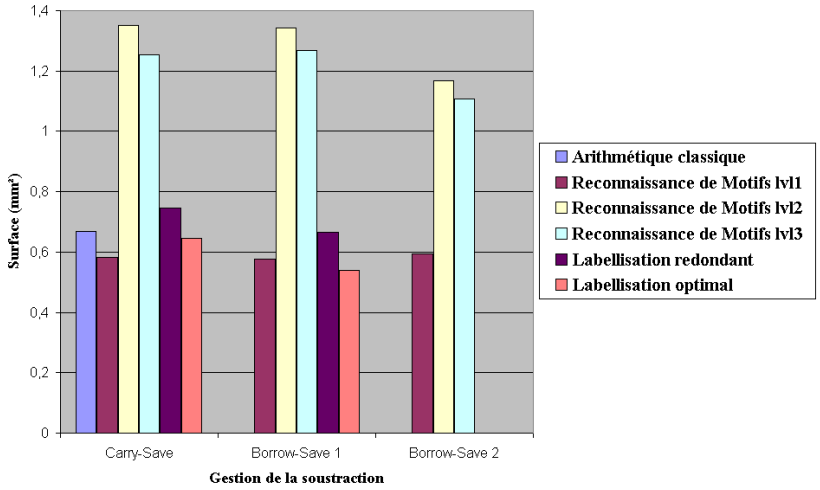
Comme précédemment, nous présentons les résultats obtenus avec tous les algorithmes à notre disposition, en ce qui concerne le traitement de la soustraction et celui des sorties multiples.

Les résultats sont présentés dans les Figures 6.23 à 6.25. Il proviennent de l'optimisation d'une *FFT* avec les données et coefficients sur 4 bits et les résultats sur 16 bits.

Ce circuit contient douze opérateurs de type papillon. Les ordres de grandeurs ne sont pas du tout les mêmes que précédemment.

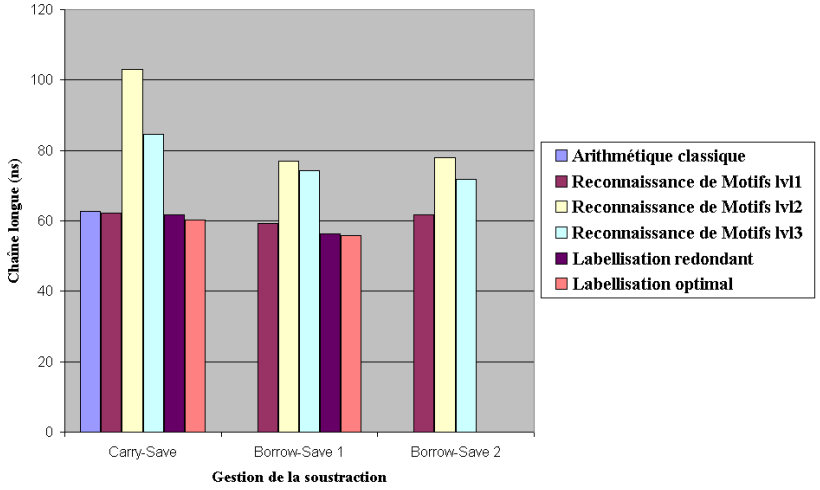
Commençons par la surface. L'optimisation sans partage des ressources dégrade la surface comme précédemment. Par contre ici, l'optimisation avec partage des ressources n'est guère meilleure. On voit avec cet exemple les limites d'une approche dédoublement des opérateurs / optimisation / fusion des opérateurs. Pour ce qui est de l'algorithme de labellisation redondant dès que possible, il dégrade la surface avec utilisation des architectures Carry-Save et l'améliore avec utilisation des architectures Borrow-Save. L'algorithme d'allocation optimale lui l'améliore dans tous les cas.

En ce qui concerne la chaîne longue, les algorithmes agrandissant la surface engendrent également des chaînes longues augmentées. Cette conclusion n'est pas justifiée d'un point de vue arithmétique, c'est l'optimisation des arbres séparément qui devrait



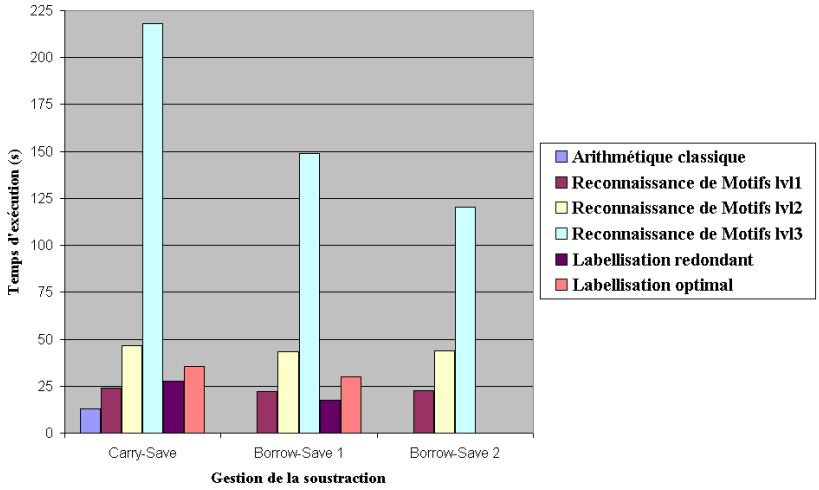
	Arithmétique classique	Reconnaissance de Motifs						Labellisation Redondant		Labellisation Optimal	
		Lvl1		Lvl2		Lvl3					
	mm^2	mm^2	%	mm^2	%	mm^2	%	mm^2	%	mm^2	%
CS	0.67	0.58	-13.4%	1.35	+101.5%	1.25	+86.6%	0.75	+11.9%	0.65	-3%
BS 1	-	0.57	-14.9%	1.34	+100%	1.27	+89.5%	0.66	-1.5%	0.54	-19.4%
BS 2	-	0.59	-11.9%	1.17	+74.6%	1.11	+54.7%	-	-	-	-

FIG. 6.23 – Résultats de la FFT : Surface



	Arithmétique classique	Reconnaissance de Motifs						Labellisation Redondant		Labellisation Optimal	
		Lvl1		Lvl2		Lvl3					
	ns	ns	%	ns	%	ns	%	ns	%	ns	%
CS	62.64	62.25	-0.6%	102.91	+64.3%	84.54	+35%	61.82	-1.3%	60.16	-4%
BS 1	-	59.29	-5.3%	76.91	+22.8%	74.34	+18.7%	56.35	-10%	55.71	-11%
BS 2	-	61.77	-1.4%	77.87	+24.3%	71.9	+14.8%	-	-	-	-

FIG. 6.24 – Résultats de la FFT : Chaîne longue



	Arithmétique classique	Reconnaissance de Motifs						Labellisation Redondant		Labellisation Optimal	
		Lvl1		Lvl2		Lvl3					
	s	*s*	%	*s*	%	*s*	%	*s*	%	*s*	%
CS	13.11	24.14	+84.1%	46.74	+256.5%	218.22	+1564%	27.81	+112.1%	35.45	+170.4%
BS 1	-	22.28	+69.9%	43.27	+230%	148.76	+1034%	17.68	+34.8%	30	+128.8%
BS 2	-	22.73	+73.4%	43.88	+234.7%	120.23	+817%	-	-	-	-

FIG. 6.25 – Résultats de la FFT : Temps d'exécution

donner les moins bons résultats (ce sont d'ailleurs les moins bons résultats parmi les autres algorithmes). Cependant, d'un point de vu pratique, on travaille ici sur un circuit suffisamment gros pour que les phases de placement / routage influencent fortement les performances temporelles ; ces deux optimisations génèrent des architectures "pénalisées" par leur surface (presque double) dans lesquelles le temps passé dans les fils est très important comparé à des architectures plus petites. Pour ce qui est des trois autres algorithmes, c'est l'algorithme d'allocation optimale qui donne les meilleures résultats, et l'utilisation des architectures Borrow-Save.

Notons par ailleurs que, que ce soit du point de vue de la surface ou de la chaîne longue, l'algorithme d'allocation optimale donne ici des performances différentes de celles obtenues avec l'algorithme de labellisation redondant dès que possible. Cela est dû au fait que plusieurs instances restent dans leur version classique : 32 sur 60 par exemple pour l'optimisation Borrow-Save.

L'analyse des temps d'exécution confirme la conclusion qui avait été émise lors de l'étude de la *DCT* complète : pour les gros circuits, l'algorithme d'allocation optimale est beaucoup plus rapide que l'optimisation avec partage des ressource de la reconnaissance de motifs. L'algorithme de labellisation redondant dès que possible est comme toujours le plus rapide.

6.5 Conclusion

Nous avons présenté l'utilisation de nos différents algorithmes sur plusieurs opérateurs arithmétiques.

Les différentes architectures ainsi obtenues ont été comparées, entre elles, et par rapport à des architectures n'utilisant pas l'arithmétique redondante.

Nous avons ainsi pu montrer que de façon générale, nous obtenons des performances intéressantes, tant du point de vue de la surface que de celui de la chaîne longue.

En ce qui concerne l'utilisation des architectures Borrow-Save lors de l'optimisation de circuits contenant des soustractions, on peut conclure qu'elle mène, dans le pire des cas, à des performances identiques à celles obtenues sans utiliser ces architectures. Dans la plupart des cas, les performances obtenues sont meilleures, et ce, quelque soit l'algorithme utilisé.

En ce qui concerne la gestion des opérateurs avec sorties multiples, notre conclusion est que l'optimisation avec partage des ressources est celle qui produit les meilleures performances. C'est en total accord avec notre étude théorique.

Enfin, en ce qui concerne plus particulièrement les algorithmes que nous avons mis en place, les deux algorithmes de labellisation apparaissent comme une meilleure approche que la reconnaissance de motifs.

En effet, pour pouvoir comparer les différents algorithmes entre eux, il faut comparer l'optimisation avec partage des ressources. Cette version de la reconnaissance de motifs est beaucoup trop coûteuse, et ne conduit pas à des performances optimales.

La reconnaissance de motifs est donc une approche intéressante, mais est plutôt destinées aux circuits ne possédant pas d'opérateurs avec sorties multiples. En ce qui concerne le filtre par exemple, on a pu voir que cette approche permet des modifications d'architectures plus importantes que celles des deux autres algorithmes, et qu'elle peut donc amener à des performances meilleures.

Parmi les deux algorithmes de labellisation, l'approche *tout en redondant* allie de bons résultats et un temps de calcul très peu important. L'approche cherchant la solution optimale quant à elle amène également à de bonnes performances, avec un temps de calcul généralement doublé. Dans la plupart des cas, la chaîne longue est légèrement meilleure et la surface un peu dégradée par rapport à l'autre approche.

Enfin nous avons pu constater que nos algorithmes génèrent des architectures ayant des surfaces et chaînes longues meilleures que celles faites à la main (pour les circuits pour lesquels cette comparaison a pu être faite). Si l'on prend également en compte le temps nécessaire à la conception de ces circuits à la main, et les connaissances en arithmétique nécessaires, on en conclut aisément l'intérêt de tels outils automatiques.