Chapitre 1

Extensions SIMD pour l'accélération des décodeurs de CCE

Dans ce chapitre, l'amélioration des instructions scalaires opérant sur des données codées sur 8 bits vers des instructions de type SIMD afin d'exploiter l'architecture 32 bits des processeurs est détaillée. L'objectif est alors de permettre une montée en débit des décodeurs logiciels grâce aux stratégies de parallélisation utilisées dans la littérature. De manière analogue à l'intégration des instructions scalaires, les résultats issus du prototypage sur cible FPGA des processeurs RISC-V enrichis évaluent l'impact des propositions sur le débit et la complexité.

Il est à noter que dans ce chapitre, comme pour le précédent, nous étudierons des architectures à 2 registres d'entrée et un registre de sortie. Ce format correspond au format standard des jeux d'instructions des processeurs x86 (Intel, AMD), RISC et dérivés (ARM) et RISC-V. Une étude exploitant des architectures levant en partie cette restriction sera présentée dans le chapitre suivant.

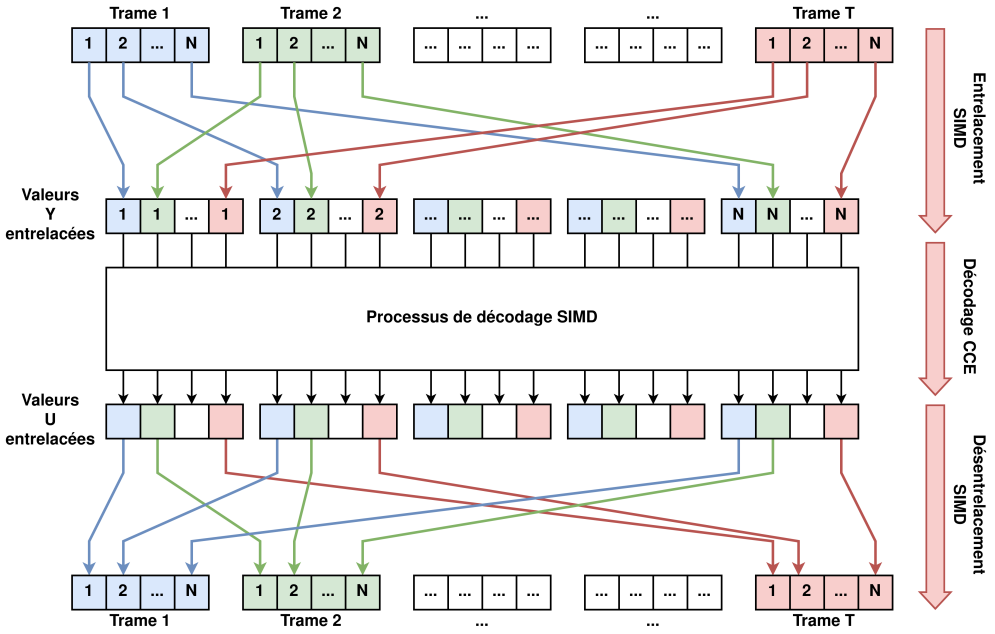

Figure 1.1 – Approche de la parallélisation inter-trames incluant les étages d'entrelacement dans l'ordre et inverse.

1.1 Introduction

L'étude et les expériences précédentes se sont focalisées sur des instructions de type SISD. Toutefois, dans la littérature, les décodeurs logiciels tirent parti des fonctionnalités SIMD des architectures multicœurs actuelles pour atteindre des niveaux de performances élevés. Ainsi, l'extension des instructions scalaires sur 8 bits vers des vecteurs de $Q \times 8$ bits est étudié et évalué dans cette section. Comme cela a été introduit dans le chapitre précédent, les décodeurs de CCE manipulent des données sur 8 bits. L'architecture interne des processeurs modernes, de leur chemin de données à leurs registres, sont sur 32 bits, voir 64 bits. Ainsi il est théoriquement possible sur des architectures 32 bits de traiter jusqu'à quatre données en parallèle en un même cycle d'horloge si les formulations des algorithmes le permettent. Le schéma de parallélisation inter-trames consiste à répliquer les mêmes opérations de décodage sur plusieurs trames indépendantes afin de mettre en exergue un parallélisme de calcul facilement accessible. Cela permet d'augmenter le débit des architectures au prix de faibles modifications dans les codes algorithmiques. Cette parallélisation inter-trames a été utilisée à l'origine dans des travaux relatifs à l'accélération des décodeurs CCE sur des architectures multicœurs [1-3]. Toutefois, afin de faciliter les accès mémoires et maximiser l'utilisation les unités de calcul SIMD, les données doivent

être arrangées correctement dans la mémoire. Un entrelacement des données contenues dans les différentes trames est réalisé en amont du processus de décodage, tel que cela est schématisé dans la figure 1.1. Pour maintenir le comportement du système, à la fin de chaque décodage, l'opération inverse est exécutée. Comme cela a été rapporté dans [3], le temps d'exécution des ces étapes d'entrelacement des données est négligeable par rapport au temps d'exécution des processus de décodage ces CCE.

À l'opposée de cette stratégie de parallélisation inter-trames opérant sur des jeux de données différents, la parallélisation intra-trame vise à exploiter le parallélisme de calcul interne à l'algorithme de décodage, tel que cela est mis en œuvre dans l'intégralité des décodeurs de CCE matériels. Cette approche permet de réduire fortement la complexité mémoire, car le processus de décodage ne mémorise et ne traite qu'une trame à la fois. Cela permet aussi de réduire la latence de traitement. Cependant, d'un point de vue logiciel, la formalisation de tels décodeurs est plus complexes à réaliser que pour l'approche inter-trames.

Dans tous les cas, comme cela a été démontré dans la littérature, les formulations algorithmiques permettant de tirer parti des spécificités architecturales tel que les unités SIMD sont plus complexes à décrire. De plus, les facteurs d'accélération n'évoluent pas toujours linéairement avec la taille des unités SIMD.

Pour pouvoir utiliser l'ensemble de ces stratégies de parallélisation, il est nécessaire d'étendre le jeu d'instructions afin pour pouvoir gérer par exemple les opérations conditionnelles. En effet, lors des traitements SIMD, les mêmes traitements sont appliqués à l'ensemble des données, rendant de fait impossible l'utilisation de structure de type if permettant d'inverser par exemple le signe d'une donnée en fonction d'un calcul de parité. Pour palier à cela, des instructions spécifiques ont été ajoutées à la liste des opérations scalaires dédiées déjà identifiées. La liste exhaustive des extensions nécessaires aux descriptions SIMD est fournie dans le tableau 1.1. Ce tableau indique pour chaque instruction son usage en fonction des stratégies de parallélisation. La description des traitements réalisés par ces instructions a été placé en annexe pour des raisons de lisibilité. L'ensemble des descriptions logicielles des décodeurs CCE ont été retravaillées afin d'exploiter ces instructions et offrir des stratégies de parallélisation inter et intra-trame. Un résumé du nombre d'instructions par famille de code et schéma de parallélisation est fourni dans le tableau 1.2. Dans la suite de ce chapitre, les résultats obtenus sont présentés.

CCE	Mnemonic	SISD (int8)	SIMD Inter (int8x4)	SIMD Intra (int8x4)
LDPC	i8_abs_pi8	✓	✓	✓
	i8_add_sat127_pi8	✓	✓	✓
	i8_invB_Aneq1_pi8	✓	✓	
	i8_xorA_signB_pi8	✓	✓	
	i8_cmpeq_pi8	✓	✓	✓
	i8_max_pi8	✓	✓	✓
	i8_min_pi8	✓	✓	
	i8_sub_sat127_pi8	✓	✓	✓
	i8_sign_pi8		✓	
	i8_cpysign_pi8			✓
	i8_signextract_pi8			✓
polaire	i8_add_sat127_pi8	✓	✓	✓
	i8_sub_sat127_pi8	✓	✓	✓
	i8_Fx_pi8	✓	✓	✓
	i8_Rx_pi8	✓	✓	✓
	i8_clrA_Bneq0_pi8		✓	✓
	i8_setMask_Aeq1		✓	✓
+ F-SC	i16_add_accA_loB_pi8		✓	
	i16_add_accA_hiB_pi8		✓	
	i8_sign_pi16		✓	
	i8_Hxor_pi8			✓
	i8_Hadd_pi8			✓
LDPC-NB	u8_minu_pu8	✓	✓	✓
	u8_addu_sat64_pu8	✓	✓	✓
	u8_subu_sat64_pu8	✓	✓	✓
	u8_cmpge_pu8		✓	
	u8_Aandb_lo8B_pu8		✓	
	u8_maxu_pu8		✓	
	u8_hminu_pu8			✓
turbo	i8_max_pi8	✓	✓	✓
	i8_scale_pi8	✓	✓	
	i8_add_pi8		✓	✓
	i8_sub_pi8		✓	✓
	i8_add_srl_pi8		✓	
	i8_srl1_pi8		✓	
	i8_sign_pi8		✓	
	i8_shuffle_pi8			✓
	i8_srl_pi8			✓
	i8_min_pi8			✓

Table 1.1 – Liste des instructions spécifiques avec Mnemonic et compabilité SISD/SIMD.

1.2 Expérimentations menées sur des décodeurs SIMD inter-trames

L'évaluation conduite s'est d'abord portée sur le schéma de parallélisation inter-trames. Ce choix est lié au fait que les modifications à apporter aux descriptions logicielles sont moins conséquentes que de basculer vers un schéma de parallélisation intra-trame. Les cœurs de processeur RISC-V étudiés étant des cœurs 32 bits, il était ainsi naturel de sélectionner un format SIMD de type *int8_t* $\times 4$ sachant que les décodeurs manipulent

CCE	SISD (int8)	SIMD Inter (int8x4)	SIMD Intra (int8x4)
LDPC	8	9	7
polaire	4	6	6
polaire F-SC	4	9	8
LDPC-NB	3	6	4
turbo	2	7	6

Table 1.2 – Synthèse du nombre d'instructions identifiées par famille de CCE.

naturellement des données sur 8 bits. Ce format SIMD permet le traitement de 4 données en parallèle par cycle d'horloge, et par voie de conséquence autorise 4 trames à être décodées de manière simultanée. Les opérations appliquées sur les 4 trames étant identiques, il est alors possible d'assurer une utilisation maximale des ressources de calcul.

Les résultats expérimentaux concernant le temps d'exécution des différents décodeurs sont reportés dans la table 1.3 et 1.4 et 1.5. Ces résultats sont comparés à ceux obtenus précédemment lorsque les instructions spécialisées manipulaient des instructions scalaires (SISD) (❷ *SISD*) et également avec les décodeurs de base sans l'aide d'instructions spécialisées ❶ *Baseline*. Les données présentées montrent qu'en moyenne, les instructions SIMD ajoutées permettent de réduire de 75% le temps de décodage des bits par rapport à la solution incorporant nos instructions SISD. Les débits sont également considérablement améliorés lorsqu'ils sont comparés avec la baseline et les instructions SISD. Ce constat est cohérent avec la stratégie de parallélisation employée, même s'il est difficile d'expliquer précisément les raisons des légères disparités entre les versions d'un même décodeur sur les différentes architectures.

Si l'on analyse dans un premier temps les codes CCE et que l'on regarde les décodeurs LDPC binaires et non binaires, le problème ne se pose pas. En effet, les gains avoisinent 75% et les débits sont améliorés d'un facteur supérieur à 6× sur toutes les architectures. Cela s'explique par le fait que ces algorithmes sont composés de nids de boucles et sont donc réguliers dans leur flot d'exécution.

Les codes polaires utilisent la récursivité durant le décodage, engendrant des appels de fonction non accélérés par les instructions SIMD introduites. En conséquence, les gains observés sont moindres, variant de ≈ 66% (**IBEX**, **RISCY**) à 70% (**SCR1**) en réduction de cycles/bit et de 4.4× à 5.3× en débit. Les codes polaires avec l'algorithme Fast-SC présentent des gains moindres, variant de 63% à ≈ 59% en cycles/bit également, et les

			LDPC OMS	SC	**polaire** F-SC	**LDPC-NB** MS	**turbo** MLM
PicoRV32	Cycles/Bit	SISD	484	91	40	11 234	202
		SIMD	-76.9%	-68.2%	-62.3%	-72.5%	-70.8%
IBEX	Cycles/Bit	SISD	199	35	16	4 245	63
		SIMD	-76.1%	-65.9%	-58.9%	-72.2%	-70%
SCR1	Cycles/Bit	SISD	172	34	15	4 226	61.3
		SIMD	-76.7%	-70.6%	-63.1%	-72.6%	-70.5%
RISCY	Cycles/Bit	SISD	176	28	13	948	54
		SIMD	-76.4%	-65.6%	-58.8%	-72.4%	-70%

Table 1.3 – Réduction du nombre de cycles d'horloge nécessaire au décodage d'un bit grâce à l'usage des instructions SIMD dédiées - Stratégie de parallélisation inter-trames.

débits varient de 4.1× à 4.9×. En effet, dans cet algorithme, la simplification de séquences d'instructions est plus ardue du fait de nœuds très spécifiques (*REP, SPC*). Ces nœuds comportent des opérations qui ne sont pas réductibles à des instructions spécialisées. Par exemple, les nœuds *SPC* nécessitent une recherche de minima avec leur position associée dans leurs versions inter-trames. Ces opérations conditionnelles sont plus simples à approcher avec une donnée unique, mais deviennent plus complexes à implanter avec 4 données distinctes.

Les turbo codes présentent une réduction des cycles nécessaires au décodage d'un bit de donnée en moyenne de 70%, conformément aux résultats attendus avec des débits variant de 3.8× à 4.5×.

En conclusion, d'un point de vue applicatif on peut noter qu'indépendamment de la famille de code testée, les versions inter-trames SIMD déployées sur les cœurs offrent systématiquement une réduction significative de $\approx 70\%$ en nombre de cycles nécessaires au décodage d'un bit et des débits augmentant d'un facteur de 5× en moyenne.

L'utilisation d'instructions de type SIMD nécessite l'utilisation d'opérateurs matériels plus complexes. En effet, 4 données différentes doivent subir le même traitement à chaque cycle d'horloge, impliquant une augmentation en théorie d'un facteur 4× de leur complexité matérielle. De plus, afin de décrire les algorithmes de décodage en utilisant une parallélisation inter-trames, de nouvelles instructions ont dû être introduites, augmentant la complexité globale des extensions proposées. La figure 1.6 fournie des informations concernant l'augmentation relative de la complexité matérielle des cœurs RISC-V, tandis que la figure 1.2 offre une représentation graphique des surcoûts absolus et la fréquence de fonctionnement maximale des différentes extensions. Ces chiffres ont été obtenus de

			LDPC OMS	SC	**polaire** F-SC	**LDPCNB** MS	**turbo** MLM
PicoRV32	❶ Baseline	cycles	217172	1333324	676218	3476435	2503511
	❷ SISD	cycles	131853	753144	332693	2875953	2133442
	❸ SIMD INTER	cycles	122462	916029	502106	3166507	2488956
	Gain (❶→❷)	cycles	85319	580180	343525	600482	370069
	Gain (❶→❷)	(%)	39.3%	43.5%	50.8%	17.3%	14.8%
	Gain (❶→❸)	cycles	94710	417295	174112	309928	14555
	Gain (❶→❸)	(%)	43.6%	31.3%	25.7%	8.9%	0.6%
	Débit ❶	Kbits/s	429	2106	4153	25	1696
	Débit ❷	Kbits/s	659	3357	7600	31	1948
	Débit ❸	Kbits/s	2657	11253	20477	102	6367
	Accélération (❶→❷)	8 bits	1.5×	1.6×	1.8×	1.2×	1.1×
	Accélération (❶→❸)	32 bits	6.2×	5.3×	4.9×	4.0×	3.8×
IBEX	❶ Baseline	cycles	84966	516441	261090	1390238	1018910
	❷ SISD	cycles	54340	288594	138544	1086873	782778
	❸ SIMD INTER	cycles	52025	394196	227587	1206494	938735
	Gain (❶→❷)	cycles	30626	227847	122546	303365	236132
	Gain (❶→❷)	(%)	36.0%	44.1%	46.9%	21.8%	23.2%
	Gain (❶→❸)	cycles	32941	122245	33503	183744	80175
	Gain (❶→❸)	(%)	38.8%	23.7%	12.8%	13.2%	7.9%
	Débit ❶	Kbits/s	317	1571	3107	18	1204
	Débit ❷	Kbits/s	515	2942	6129	24	1615
	Débit ❸	Kbits/s	2121	8333	14617	86	5402
	Accélération (❶→❷)	8 bits	1.6×	1.9×	2.0×	1.3×	1.3×
	Accélération (❶→❸)	32 bits	6.7×	5.3×	4.7×	4.7×	4.5×

Table 1.4 – Comparaison des performances des cœurs **IBEX** et **PicoRV32** usant d'instructions à 2 entrées, configuration SIMD inter-trame avec la fréquence de fonctionnement maximale par implantation.

manière similaire à ceux présentés dans la partie expérimentale relative aux instructions scalaires (SISD). De manière prévisible, l'augmentation de la complexité matérielle est plus marquée que lors des expérimentations pour les instructions scalaires. L'introduction de ces nouvelles instructions dans le cœur le moins complexe (**PicoRV32**) aboutie à une augmentation de la complexité matérielle de $\approx 50\%$ pour chacun des jeux d'extensions proposés. Ces valeurs sont liées à la faible empreinte "silicium" de sa version originale. Pour les cœurs RISC-V plus complexes, tels que les cœurs **IBEX** et **SCR1**, l'augmentation relative de la complexité est plus limitée, elle varie de +9.4% à +13.7% pour ce premier et de +7.6% à +10.1% pour le second. L'augmentation la moins marquée est ob-

			LDPC OMS	SC	**polaire** F-SC	**LDPCNB** MS	**turbo** MLM
SCR1	❶ Baseline	cycles	76775	397790	218120	1310272	927495
	❷ SISD	cycles	47995	274793	125037	1081939	758988
	❸ SIMD INTER	cycles	44000	343892	184528	1187911	897056
	Gain (❶→❷)	cycles	28780	122997	93083	228333	168507
	Gain (❶→❷)	(%)	37.5%	30.9%	42.7%	17.4%	18.2%
	Gain (❶→❸)	cycles	32775	53898	33592	122361	30439
	Gain (❶→❸)	(%)	42.7%	13.5%	15.4%	9.3%	3.3%
	Débit ❶	Kbits/s	362	2107	3842	20	1366
	Débit ❷	Kbits/s	586	3021	6639	24	1661
	Débit ❸	Kbits/s	2507	9735	18014	86	5487
	Accélération (❶→❷)	8 bits	1.6	1.4	1.7	1.2	1.2
	Accélération (❶→❸)	32 bits	6.9	4.6	4.7	4.3	4.0
RISCY	❶ Baseline	cycles	77678	348107	192935	1103396	890712
	❷ SISD	cycles	48096	228581	113843	878938	674609
	❸ SIMD INTER	cycles	45323	314875	187693	970983	810365
	Gain (❶→❷)	cycles	29582	119526	79092	224458	216103
	Gain (❶→❷)	(%)	38.1%	34.3%	41.0%	20.3%	24.3%
	Gain (❶→❸)	cycles	32355	33232	5242	132413	80347
	Gain (❶→❸)	(%)	41.7%	9.5%	2.7%	12.0%	9.0%
	Débit ❶	Kbits/s	105	706	1274	7	417
	Débit ❷	Kbits/s	170	1075	2159	9	551
	Débit ❸	Kbits/s	720	3122	5237	32	1834
	Accélération (❶→❷)	8 bits	1.6×	1.5×	1.7×	1.3×	1.3×
	Accélération (❶→❸)	32 bits	6.9×	4.4×	4.1×	4.5×	4.4×

Table 1.5 – Comparaison des performances des cœurs **SCR1** et **RISCY** usant d'instructions à 2 entrées, configuration SIMD inter-trame avec la fréquence de fonctionnement maximale par implantation.

servable pour le cœur le plus complexe, le **RISCY** pour lequel la variation est seulement de $\approx 4\%$ au maximum.

L'introduction des nouvelles instructions impacte aussi la fréquence de fonctionnement maximale des différents cœurs, tel que le démontrent les données rapportées dans le tableau 1.6 et la figure 1.2. Le cœur le plus impacté est le **PicoRV32** avec une baisse moyenne de $\approx 10\%$ de sa fréquence de fonctionnement, ce qui est $1.5\times$ plus important que précédemment. Pour le cœur **IBEX**, l'impact est quand à lui plus faible ou équivalent en fonction de l'extension considérée, tandis que pour le cœur **RISCY**, aucun impact n'a été observé. En effet, ce cœur est limité en fréquence par d'autres composants de son architecture. L'ensemble de ces résultats démontrent que malgré l'augmentation de

	BASE	LDPC	POLAIRE		LDPC-NB	TURBO
		OMS	SC	Fast-SC	MS	Max-Log MAP
Nb. Insn Cst.		9	6	9	6	7
PicoRV32	978 LUTs	+44.7%	+41.8%	+48.0%	+43.3%	+37.7%
	567 FFs	+3.4%	+2.3%	+3.0%	+2.3%	2.6%
	303 MHz	-12.8%	-8.2%	-8.5%	-7.8%	-6.7%
IBEX	2 446 LUTs	+12.2%	+12.4%	+13.1%	+13.7%	+9.4%
	883 FFs	0.0%	0.0%	0.0%	0.0%	0%
	100 MHz	+2.4%	+4.7%	+1.2%	+2.9%	+3.4%
SCR1	3 986 LUTS	+10.1%	+8.5%	+8.5%	+9.8%	+7.6%
	2 399 FFs	+0.1%	0.0%	0.0%	0.0%	0.0%
	102MHz	-0.9%	-0.1%	-0.1%	-2.8%	-2.9%
RISCY	10 302 LUTs	+3.1%	+3.8%	+4.1%	+4%	+3.2%
	3 033 FFs	0%	0%	0%	0%	0%
	30 MHz	0%	0%	0%	0%	0%

Table 1.6 – Impact des instructions SIMD dédiées sur la complexité matérielle et la fréquence de fonctionnement des cœurs RISC-V - Stratégie de parallélisation inter-trames

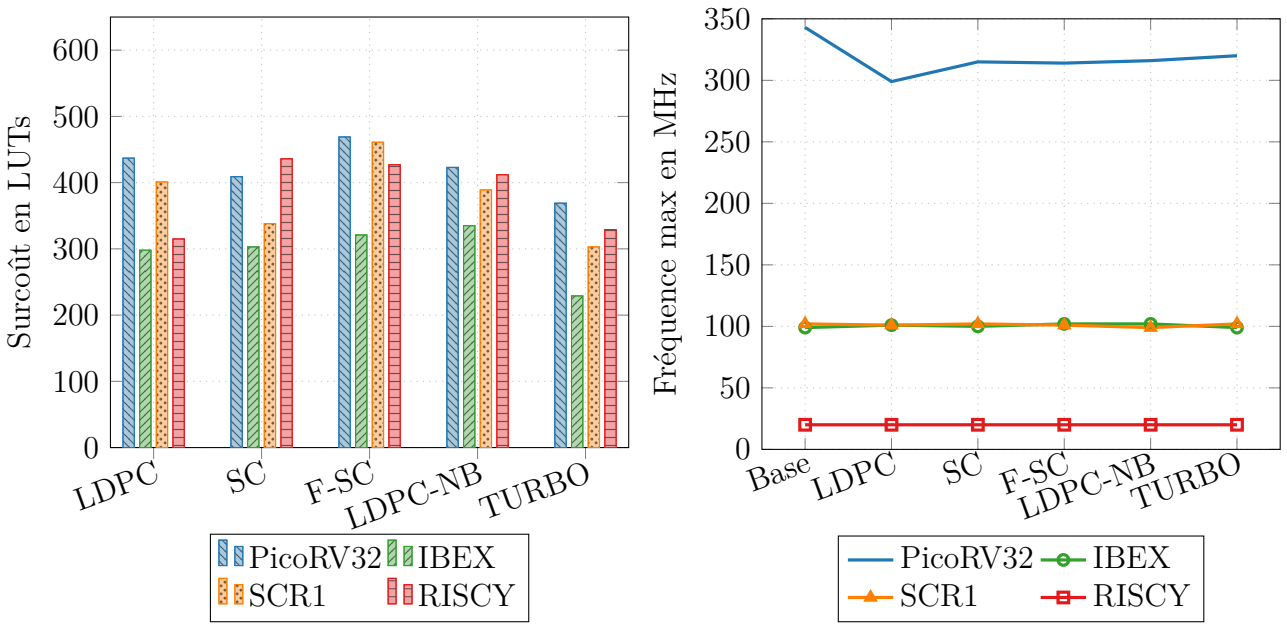

Figure 1.2 – Impact de l'ajout des instructions spécialisées : surcoût en LUTs et fréquence de fonctionnement.

la complexité matérielle des architectures et la baisse de la fréquence de fonctionnement, les instructions SIMD proposées permettent d'améliorer notablement les performances globales des décodeurs logiciels déployés sur les cœurs RISC-V.

1.3 Expérimentations menées sur des décodeurs SIMD intra-trame

Cette dernière section présente les gains obtenus lorsque les extensions SIMD sont utilisées pour accélérer des décodeurs utilisant une stratégie de parallélisation intra-trames. Par opposition à la parallélisation inter-trames, l'approche intra-trame utilise le parallélisme interne à l'algorithme pour minimiser la latence de décodage. La mise en œuvre de cette stratégie de parallélisation nécessite toutefois l'usage d'instructions SIMD différentes de celles employées pour réaliser de l'inter-trames comme cela est détaillé dans le tableau 1.1. À partir des instructions identifiées et intégrées dans les cœurs de processeur RISC-V, nous avons développé des versions spécifiques des codes sources des décodeurs logiciels dans des formes proches de celles décrites dans [4-6]. Ces décodeurs ont été évalués avec les cœurs RISC-V déployés sur circuit FPGA. Les temps d'exécution et les débits maximum ainsi obtenus ont ensuite été comparés. Les données issues de ces expérimentations sont résumées dans le tableau 1.7 et 1.8.

Les résultats expérimentaux mettent tout d'abord en évidence une accélération des décodeurs spécialisés dans le décodage des codes polaires et des turbo codes. Les gains en termes de temps d'exécution sont pour ces familles de codes compris dans l'intervalle de $\approx$ 58% à 68%. Ces gains sont calculés par rapport aux versions scalaires ❷ *SISD* développées précédemment. Les gains relatifs aux versions de base ❶*Baseline* sont également présentés. Concernant les décodeurs LDPC, les limitations architecturales des cœurs RISC-V employés réduisent drastiquement l'intérêt de ce type d'implantation.

En effet, le processus de décodage LDPC réalise des accès 32 bits non alignés à la mémoire. Ce type d'accès, relativement atypique, n'est pas supporté par la majorité des cœurs RISCV testés, à l'exception du cœur **IBEX**.

Pour les décodeurs LDPC non binaires, l'accélération est faible en comparaison de son implantation inter-trames, avec une réduction du temps d'exécution de $\approx$ 5% à 6.8%. Ces gains limités sont liés à la nature de l'algorithme avec un schéma intra-trames, car il est difficile d'extraire des instructions spécialisées opérant sur 32 bits. À titre d'exemple, les fonctions de calcul utilisées pour le décodage des nœuds de parités sont dépendants d'accès en mémoire ordonnés pour chaque donnée de 8b. Les fonctions de conversion entre le domaine naturel et le domaine des $\mathbb{GF}$ sont également problématiques. Ainsi, sauf pour certaines sections du décodeur, extraire des instructions compatibles avec un alignement des données en mémoire bénéfique à l'exécution intra-trame est impossible. Si l'on s'in-

			SC	polaire FSC	LDPCNB MS	Turbo MLM
PicoRV32	❶ Baseline	cycles	1333324	676218	3476435	2503511
	❷ SISD	cycles	753144	332693	2875953	2133442
	❸ SIMD INTRA	cycles	327502	133244	2713146	788086
	Gain❶→❷	cycles	580180	343525	600482	370069
	Gain❶→❷	%	43.5%	50.8%	17.3%	14.8%
	Gain❶→❸	cycles	1005822	542974	763289	1715425
	Gain❶→❸	%	75.4%	80.3%	22.0%	68.5%
	Débit❶	Kbits/s	2106	4153	25.2	1695
	Débit❶	Kbits/s	3357	7599	30.5	1947
	Débit❶	Kbits/s	7797	18704	25	5049
	Accel❶→❷	8bits	1.6	1.8	1.2	1.1
	Accel❶→❸	32bits	3.7	4.5	1.0	3.0
IBEX	❶ Baseline	cycles	516441	261090	1390238	1018910
	❷ SISD	cycles	288594	138544	1086873	782778
	❸ SIMD INTRA	cycles	46379	126879	1025576	286696
	Gain❶→❷	cycles	227847	122546	303365	236132
	Gain❶→❷	%	44.1%	46.9%	21.8%	23.2%
	Gain❶→❸	cycles	470062	134211	364662	732214
	Gain❶→❸	%	91.0%	51.4%	26.2%	71.9%
	Débit❶	Kbits/s	1571	3107	18	1204
	Débit❶	Kbits/s	2942	6129	24	1615
	Débit❶	Kbits/s	6711	13560	25	4571
	Accel❶→❷	8bits	1.9	2.0	1.3	1.3
	Accel❶→❸	32bits	4.3	4.4	1.4	3.8

Table 1.7 – Comparaison des performances des cœurs usant d'instructions à 2 entrées, configuration SIMD intra-trame avec la fréquence de fonctionnement maximale par implantation.

téresse maintenant à l'évolution de la complexité matérielle des cœurs enrichis, dont les valeurs sont fournies dans le tableau 1.9, on peut remarquer que les extensions nécessaires pour assurer une parallélisation intra-trame ont un coût équivalent à celles utilisées pour une parallélisation inter-trame. Les variations en termes de complexité matérielle se situent dans l'intervalle $[-3\% ; +8\%]$ en fonction des cœurs et des CCE. Les disparités dans ces gains et dans ces pertes sont difficilement explicables et semblent être uniquement liées aux choix réalisés par les outils de synthèse logique. Par exemple, pour l'extension dédiée au décodage *Fast-SC*, elle est $\approx 5\%$ moins onéreuse sur le cœur **PicoRV32** tandis que dans le même temps, elle est $\approx 7\%$ plus complexe sur le cœur **IBEX**. Ces observations liées à l'augmentation de la complexité matérielle entre ces deux types d'extension se retrouvent lorsque l'on analyse l'impact que peuvent avoir les extensions sur la fréquence maximale de fonctionnement des cœurs. Les ordres de grandeur sont conservés malgré quelques disparités.

			SC	Polaire FSC	LDPCNB MS	Turbo MLM
SCR1	❶ Baseline	cycles	397790	218120	1310272	927495
	❷ SISD	cycles	274793	125037	1081939	758988
	❸ SIMD INTRA	cycles	115954	49161	1018811	283452
	Gain❶→❷	cycles	122997	93083	228333	168507
	Gain❶→❷	%	30.9%	42.7%	17.4%	18.2%
	Gain❶→❸	cycles	281836	168959	291461	644043
	Gain❶→❸	%	70.9%	77.5%	22.2%	69.4%
	Débit❶	Kbits/s	2107	3842	20	1366
	Débit❶	Kbits/s	3020	6638	24	1660
	Débit❶	Kbits/s	7218	17391	26	4334
	Accel❶→❷	8bits	1.4	1.7	1.2	1.2
	Accel❶→❸	32bits	3.4	4.5	1.3	3.2
RISCY	❶ Baseline	cycles	348107	192935	1103396	890712
	❷ SISD	cycles	228581	113843	878938	674609
	❸ SIMD INTRA	cycles	95375	44752	832577	216187
	Gain❶→❷	cycles	119526	79092	224458	216103
	Gain❶→❷	%	34.3%	41.0%	20.3%	24.3%
	Gain❶→❸	cycles	252732	148183	270819	674525
	Gain❶→❸	%	72.6%	76.8%	24.5%	75.7%
	Débit❶	Kbits/s	706	1274	7	417
	Débit❶	Kbits/s	1075	2159	9	551
	Débit❶	Kbits/s	2577	5492	9	1719
	Accel ❶→❷	8bits	1.5	1.7	1.3	1.3
	Accel ❶→❸	32bits	3.6	4.3	1.3	4.1

Table 1.8 – Comparaison des performances des cœurs **SCR1** et **RISCY** usant d'instructions à 2 entrées, configuration SIMD intra-trame avec la fréquence de fonctionnement maximale par implantation.

L'ensemble de ces résultats expérimentaux mettent en évidence que les extensions dédiées à la parallélisation intra-trame offrent, pour les turbo codes et les codes polaires, des facteurs d'accélération substantiels ($> 50\%$ du temps d'exécution) avec un impact limité sur les caractéristiques des cœurs si l'on omet le cœur **PicoRV32**.

1.4 Synthèse de la partie expérimentale

Cette dernière sous-section se propose de synthétiser les différents résultats présentés précédemment afin de conclure sur l'intérêt des extensions proposées. Le tableau 1.10 récapitule les données issues des différentes expérimentations. Les gains minimums, maximums et moyens en termes de réduction de cycles d'horloges nécessaires pour décoder 1 bit d'information sont fournis pour les 4 architectures testées. Il est important de rappeler que (a) les gains pour les versions SISD sont relatifs aux codes logiciels sans instruction

	BASE	LDPC OMS	POLAIRE SC	POLAIRE Fast-SC	LDPC-NB MS	TURBO Max.Log-Map
Nb. Insn Cst.		7	6	8	4	6
PicoRV32	978 LUTs	+44.4%	+41.9%	+43.3%	+37.2%	+54.3%
	567 FFs	+3.7 %	+2.3%	+2.6%	+1.4%	+2.6%
	303 MHz	-9.2 %	-9.0%	-7.8%	-21.6%	-6.3%
IBEX	2 446 LUTs	+11.0%	+16.3%	+20.5%	8.9%	+22.4%
	883 FFs	0.0%	0.0%	0.0%	0.0%	+0.8%
	100 MHz	+3.4%	+5.0%	-2.2%	+3.0%	+6.9%
SCR1	3 986 LUTs	+10.1%	+8.5%	+11.6%	+5.2%	+11.0%
	2 399 FFs	0.0%	0.0%	0.0%	0.0%	0.0%
	102 MHz	-2.6%	-0.1%	-0.8%	-0.7%	-3.0%
RISCY	10 302 LUTs	+4.6%	+3.8%	+4.5%	+3.4%	+5%
	3 033 FFs	0.0%	0.0%	0.0%	0.0%	0.0%
	30 MHz	0.0%	0.0%	0.0%	0.0%	0.0%

Table 1.9 – Impact des instructions SIMD dédiées sur la complexité matérielle et la fréquence de fonctionnement des cœurs RISC-V - Stratégie de parallélisation intra-trame

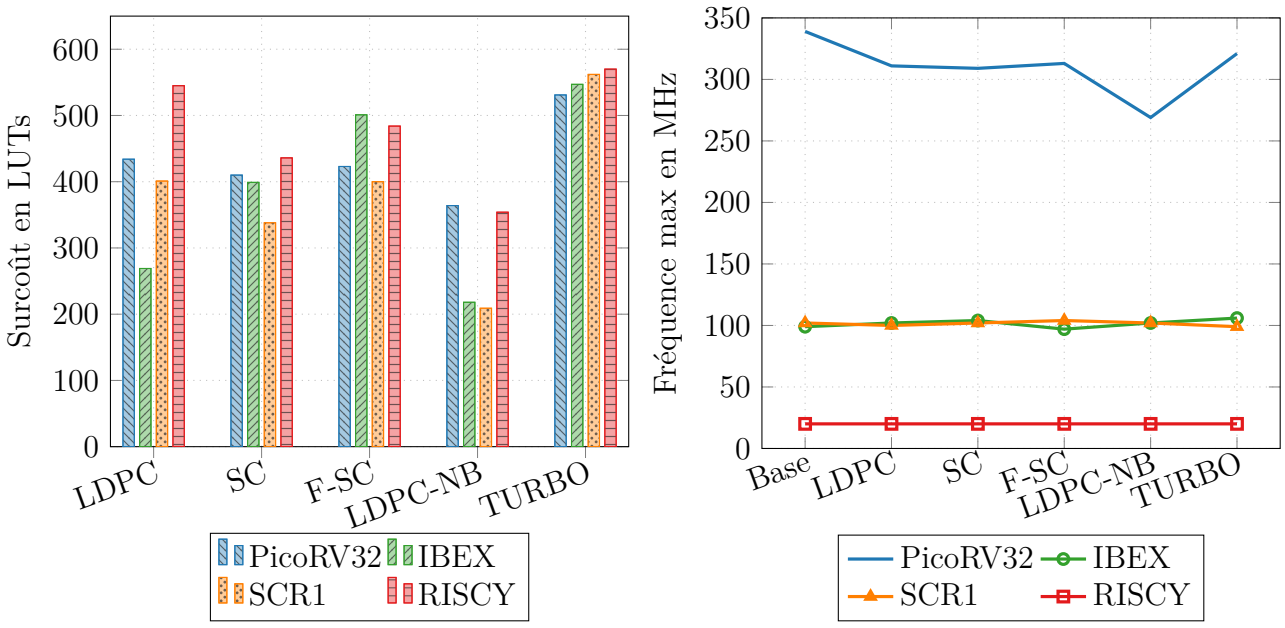

Figure 1.3 – Impact de l'ajout des instructions SIMD intra-trame : surcoût en LUTs et fréquence de fonctionnement

spécialisée, tandis que les gains inter/intra-trame ont été calculés par rapport aux versions SISD ; (b) pour les les versions inter-trames, le gain en terme de temps d'exécution permet une augmentation du débit des décodeurs, mais ne réduit pas la latence de décodage par rapport aux implantations SISD, tandis qu'avec la parallélisation intra-trame,

CCE	Implantation	**Réduction** (cycles/bit en %) min	moy	max	**Surcoût matériel** (LUTs en %) min	moy	max
LDPC	SISD	36.0%	37.7%	39.3%	1.8%	12.5%	34.0%
OMS	INTER	76.1%	76.6%	77.1%	3.1%	17.5%	44.7%
	INTRA	14.7%	14.7%	14.7%	4.6%	17.5%	44.4%
polaire	SISD	30.9%	38.2%	44.1%	2.1%	5.1%	12.3%
SC	INTER	65.6%	67.4%	69.6%	3.8%	16.6%	41.8%
	INTRA	56.0%	57.2%	58.3%	3.8%	17.6%	41.9%
polaire	SISD	41.0%	45.4%	50.8%	2.1%	5.1%	12.3%
F-SC	INTER	58.8%	60.8%	63.1%	4.1%	19.2%	48.0%
	INTRA	57.7%	59.8%	60.7%	4.5%	24.6%	63.2%
LDPC NB	SISD	17.3%	19.2%	21.8%	0.8%	5.1%	12.5%
MS	INTER	72.2%	72.4%	72.6%	4.0%	17.7%	43.3%
	INTRA	5.3%	5.6%	5.8%	3.4%	13.7%	37.2%
turbo	SISD	14.8%	20.1%	24.3%	0.6%	4.7%	12.6%
Max-Log-Map	INTER	70.0%	70.3%	70.8%	3.2%	14.5%	37.7%
	INTRA	62.7%	64.3%	68.0%	5.0%	23.2%	54.3%

Table 1.10 – Impacts des instructions sélectionnées en termes de réduction de cycles d'horloge par bit décodé et le surcoût en LUT

l'augmentation du débit s'accompagne d'une réduction de la latence du même ordre de grandeur. Des colonnes similaires fournissent les métriques relatives aux surcoûts en LUT, cependant toutes ces valeurs doivent se lire relativement aux cœurs RISC-V originaux.

Ce tableau met en relief les gains obtenus pour chacune des familles de codes correcteurs d'erreurs considérés. Il est à noter que les gains sont substantiels en SISD lorsque l'on observe le temps d'exécution. Cependant, tous les algorithmes de décodage n'en bénéficient pas de manière équivalente. Par exemple, les turbo codes, avec seulement 2 instructions spécialisées possibles, ne gagnent que $\approx$ 15 20%. Le constat est similaire pour le décodage des codes LDPC-NB, qui avec uniquement 3 instructions spécialisées ne voit son temps d'exécution réduit que de $\approx$ 20%. En fonction des cœurs RISC-V considérés, le surcoût matériel induit par l'introduction de ces nouvelles instructions est bien inférieur aux gains apportés : de $\approx 2\times$ à $\approx 8\times$.

Les versions inter-trames (INTER) qui traitent en interne 4 données de 8 bits en parallèle, réduisent le temps de décodage des bits d'information de 58% et 77% par rapport aux décodeurs SISD utilisant des instructions spécifiques. Les gains ainsi obtenus permettent d'augmenter le débit binaire des décodeurs. Avec cette stratégie de parallélisation et aux instructions spécifiques ajoutées, l'ordre de grandeur des gains observés est

similaire pour les 4 cœurs RISC-V et les différents algorithmes de décodage. L'augmentation de la complexité matérielle des cœurs de processeur RISC-V est plus conséquente que lors des expérimentations SISD, cependant les jeux d'instructions étant différents, l'augmentation moyenne du nombre de LUT post-implantation varie de 1.5× à 4×. À l'exception du cœur **PicoRV32** pour lequel l'augmentation de la complexité matérielle atteint environ 50%, pour les autres cœurs cette augmentation est beaucoup plus faible (ex. $\leq 4\%$ pour le cœur **RISCY**).

Les expérimentations ciblant un schéma de parallélisation intra-trame pour les algorithmes de décodage offrent des résultats plus mitigés. L'intérêt de ce schéma de parallélisation est lié à la réduction de la latence de décodage. Il s'inspire des stratégies mises en œuvre dans les architectures matérielles. D'un point de vue théorique, les gains mesurés devraient être similaires à ceux obtenus pour les expérimentations INTER car 4 données 8 bits sont ici aussi traitées en parallèle. Cependant, comme le démontrent les résultats rapportés dans le tableau 1.10, les facteurs d'accélération obtenus pour les codes LDPC et LDPC-NB sont bien inférieurs à ceux obtenus par leurs homologues INTER. Ce constat est lié aux formulations des algorithmes de décodage qui limitent l'exploitation du parallélisme et impactent négativement sur la conception des instructions spécialisées. Les résultats sont meilleurs pour les autres familles de CCE utilisées, pour lesquelles des réductions de 57% à 64% du temps d'exécution sont mesurées. Le coût matériel des instructions spécialisées rajoutées dans ce contexte s'avère, en outre plus onéreux que pour la parallélisation inter-trames (jusqu'à 10%).

Pour conclure, l'ensemble de ces résultats expérimentaux démontrent l'intérêt des extensions proposées pour l'accélération du décodage des CCE actuels. Afin d'améliorer les performances ainsi obtenues, différentes pistes d'amélioration existent, tel que cela sera détaillé dans le prochain chapitre.

1.5 Conclusion

Ce chapitre permet de conclure sur l'efficacité de l'intégration d'instructions de type SIMD, pour l'accélération des CCE lorsque appliqués à leurs algorithmes de décodage. En effet, avec les résultats obtenus par les expérimentations menées, les débits augmentent en concordance avec la réduction de la latence des décodeurs étudiés.

On retrouve là une philosophie qui a abouti par exemple à l'intégration d'extensions SSEx, AVX sur processeur X86_64. Toutefois, cette amélioration des performances en-

gendre un surcoût matériel dont l'impact varie en fonction de la complexité initiale du cœur tel que cela est mis en évidence dans le tableau 1.10. Afin de minimiser cette augmentation de la complexité matérielle, il serait possible de pousser encore plus loin l'intégration de nos extensions dans les architectures cibles. Cela pourrait se faire en réutilisant par exemple tout ou partie des ressources de l'ALU, mais on perdrait alors le principe de *généricité*, sur lequel repose une grande partie de l'intérêt de la solution étudiée.

Cette première étude a été volontairement limitée à l'utilisation de jeux d'instructions à deux registres sources. Cette caractéristique a été initialement maintenue pour permettre une plus forte adaptabilité des instructions identifiées pour des ISA de cœur des calculs généralistes. Cependant, il est intéressant d'étendre cette première étude, avec des architectures proposant l'utilisation d'instruction à trois registres sources. Cette nouvelle approche est présentée dans le chapitre suivant ; nous y analyserons les avantages et les inconvénients des extensions que nous avons proposées.