RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

anr agence nationale de la recherche
AU SERVICE DE LA SCIENCE

Systematic
Paris Region Deep Tech Ecosystem

Could a CISC approach be interesting in securing a RISC(-V) architecture? The **SecV approach**

29/09/2023

Project leader: S. Pillement

SEC-V

IETR ims LS2N THALES

Fiche d'identité du projet

Liste des partenaires

- **Laboratoires** : IETR, IMS, LS2N
- **Groupe** : Thales (TRT et DIS)

AAP :

- Programme : PRCE
- Type : Recherche industrielle

Démarrage : octobre 2022
Durée : 42 mois
Coût total : 1885 k€

Projet suivi par l'ANSSI

Présentation

Contexte scientifique

Architectures embarquées hautes performances

- Hiérarchie mémoire avancée
- Spéculation

Sécurité des systèmes sur puce

Contexte industriel

- Cible
  - embarqué critique (SdF/SSI) pour l'embarqué et notamment les CPS/IoT
- Ouverture vers l'open-source matériel
  - RISC-V International, OpenHW group, etc.

Problématiques à résoudre

- Intégration de solutions de sécurité au niveau microarchitecture
- Limiter les surcoûts matériels et pertes de performances

SEC-V

Verrous et innovation

Intégration d'une unité de **décodage dynamique d'instructions** permettant le **monitoring** (identifier, détecter), l'**obfuscation** (protéger), et l'**adaptation dynamique** (réagir)

Principaux verrous

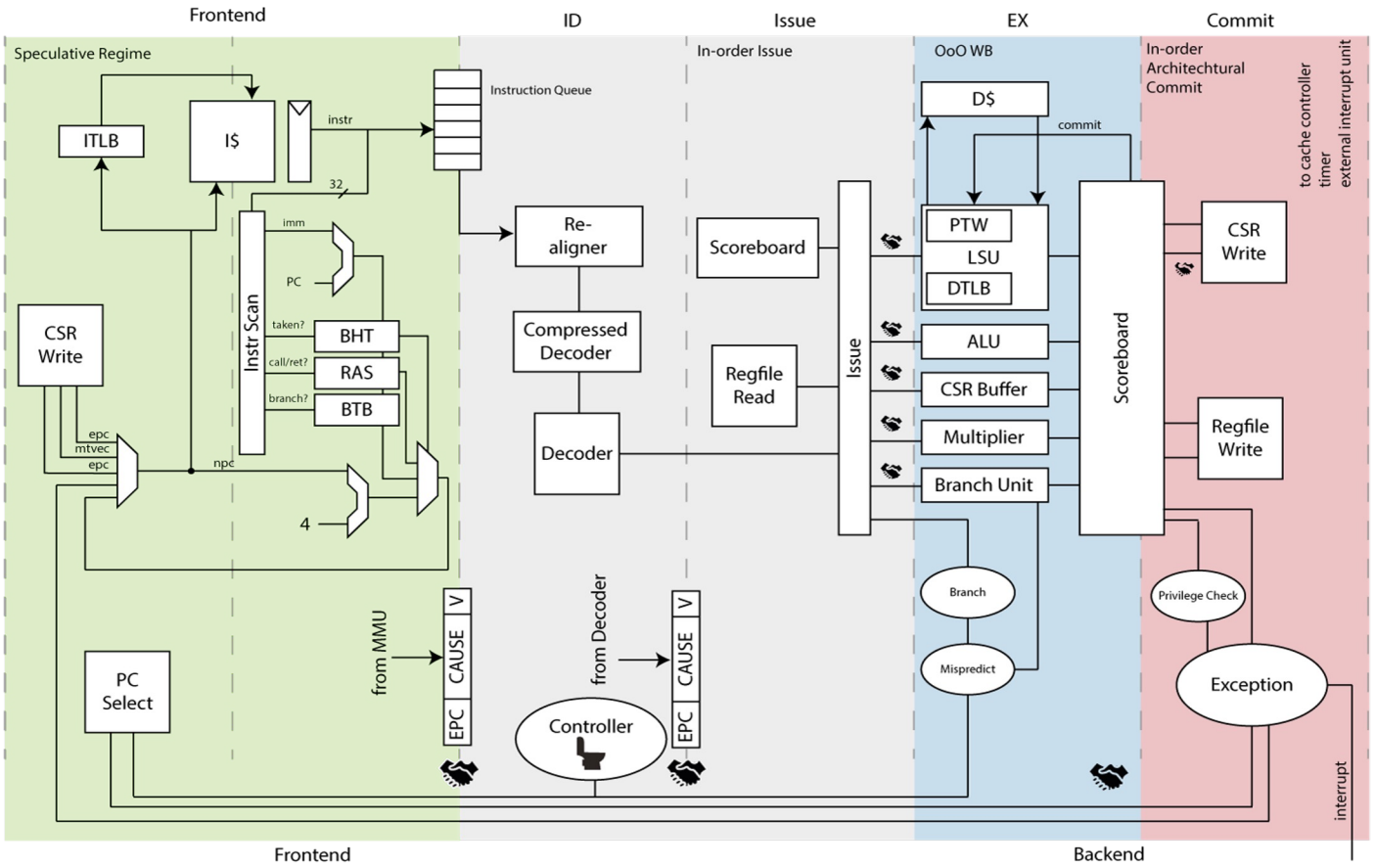
- Structure matérielle sécurisée et haute performance
- Gestion dynamique de la sécurité
- Approche innovante de sécurité par adaptation dynamique du fonctionnement de la micro-architecture

Innovations

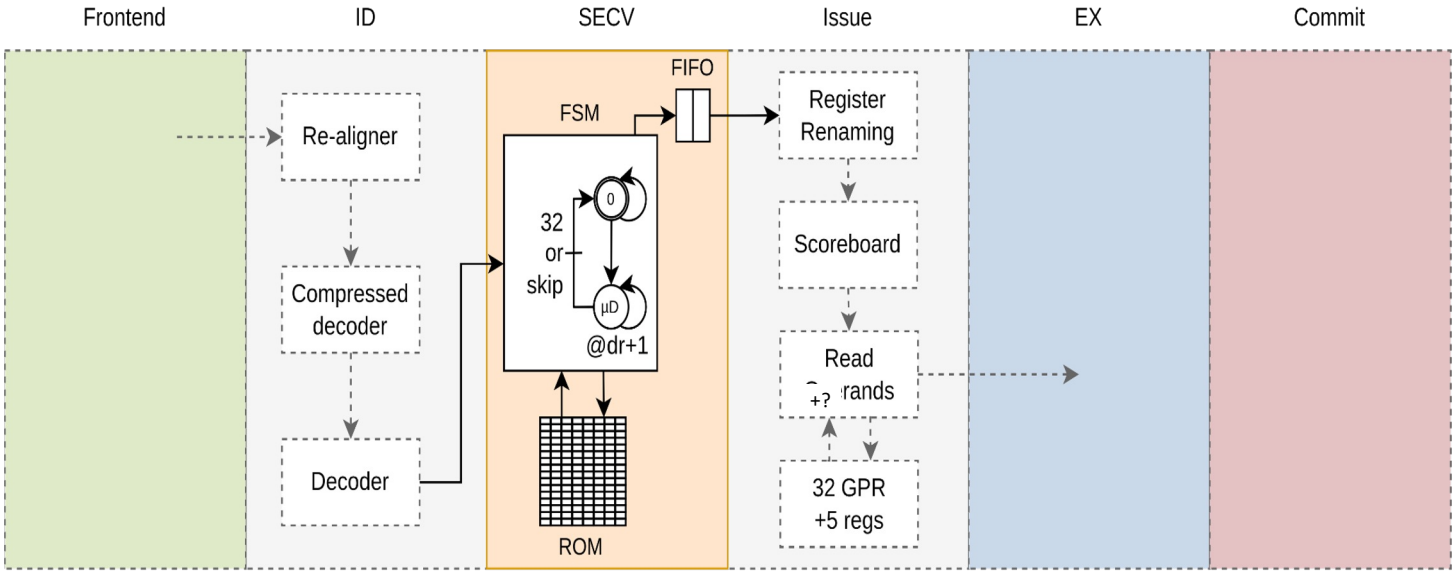
- Unité dynamique de décodage des instructions
- Politiques de gestion mémoire configurables
- Bloc de contrôle de flot et d'instructions

SEC-V

Architecture cible – CVA6



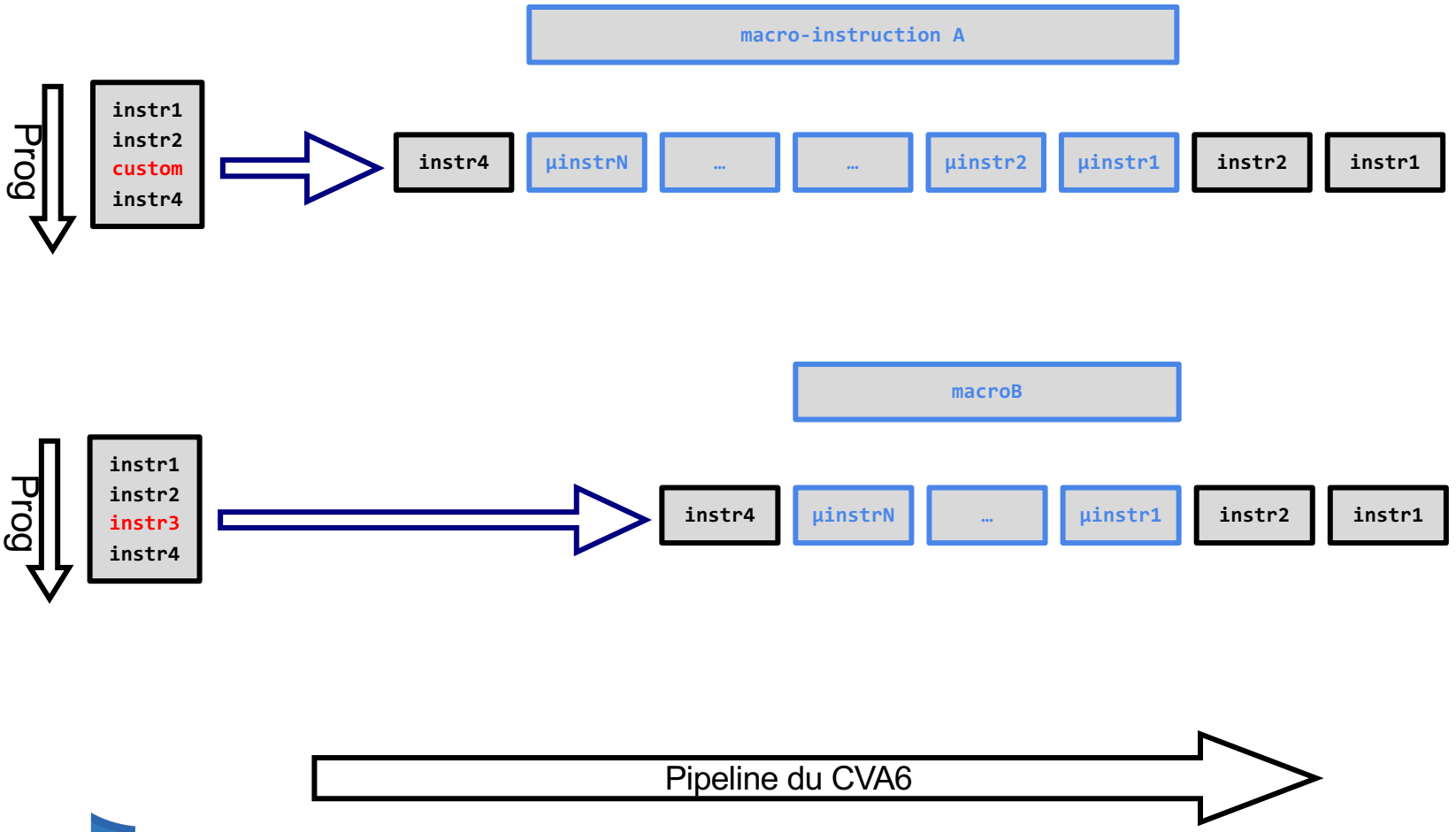
Proposition architecturale



➔ Ajout d'un étage (FSM, FIFO et ROM)

- Machine d'états
- File de type *First-In-First-Out*
- Mémoire morte

Le microdécodage d'instructions



Exemple : AES

$$s = b \oplus (b \circlearrowleft 1) \oplus (b \circlearrowleft 2) \oplus (b \circlearrowleft 3) \oplus (b \circlearrowleft 4) \oplus 0x63$$

Calcul d'un élément de la Forward Rijndael S-box [10]

Exemple : AES

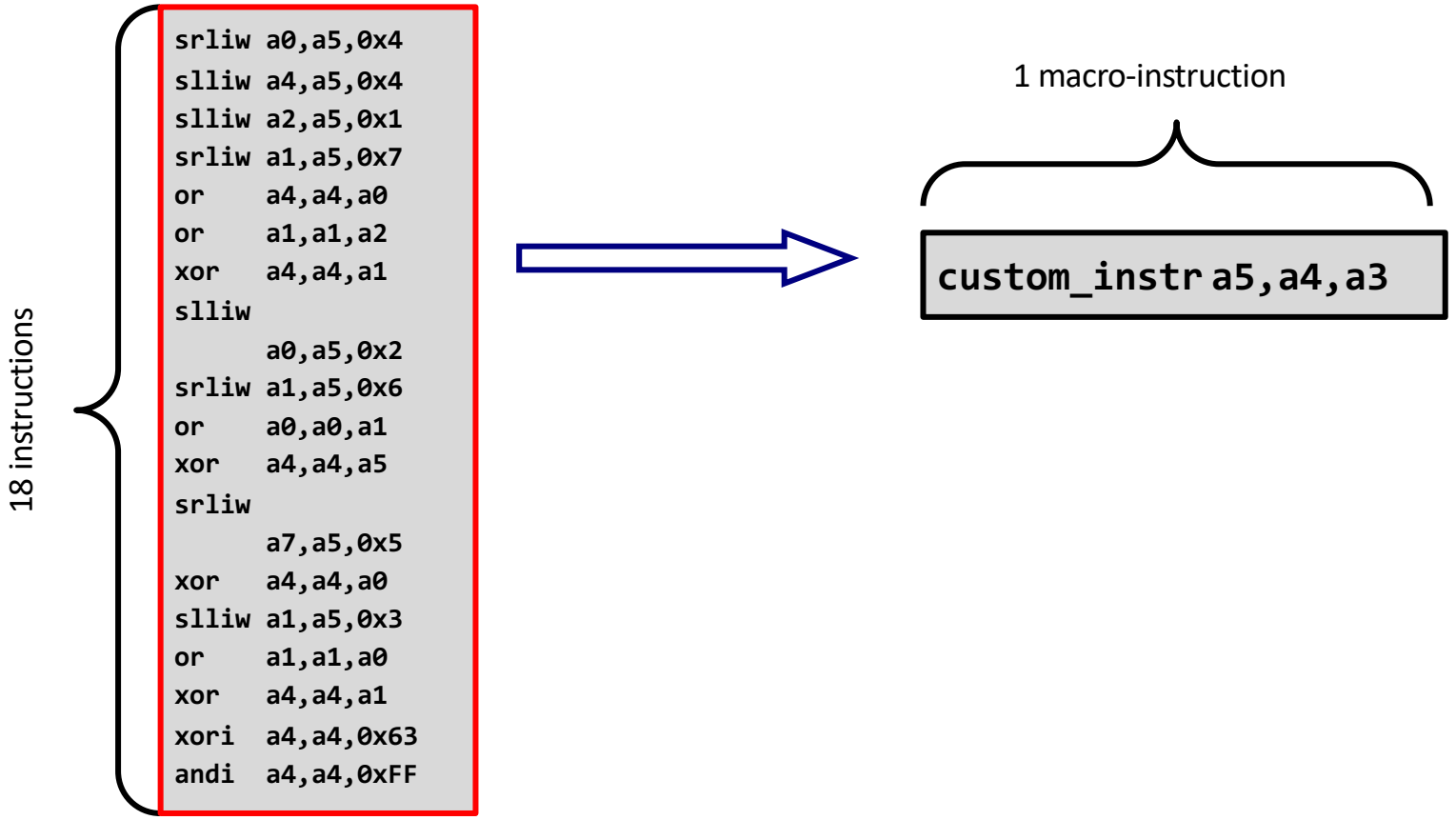
$$s = b \oplus (b \circlearrowleft 1) \oplus (b \circlearrowleft 2) \oplus (b \circlearrowleft 3) \oplus (b \circlearrowleft 4) \oplus 0x63$$

```
#define ROTL8(x,shift) ((uint8_t) ((x) << (shift)) | ((x) >> (8 - (shift))))

uint8_t temp = q ^ ROTL8(q, 1) ^ ROTL8(q, 2) ^ ROTL8(q, 3) ^ ROTL8(q, 4);

sbox[p] = temp ^ 0x63;
```

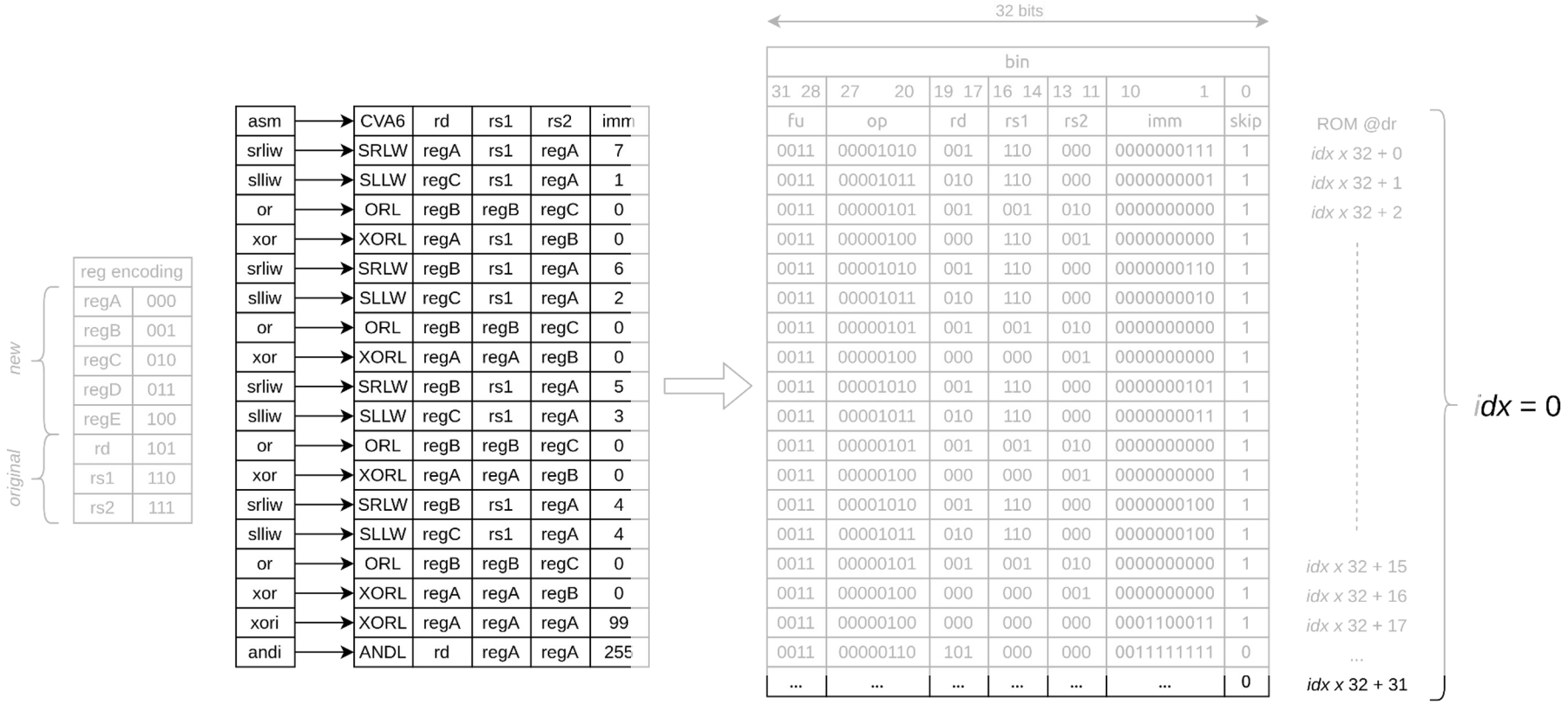
```
srliw a0,a5,0x4
slliw a4,a5,0x4
slliw a2,a5,0x1
srliw a1,a5,0x7
or    a4,a4,a0
or    a1,a1,a2
xor   a4,a4,a1
slliw
      a0,a5,0x2
srliw a1,a5,0x6
or    a0,a0,a1
xor   a4,a4,a5
srliw
      a7,a5,0x5
xor   a4,a4,a0
slliw a1,a5,0x3
or    a1,a1,a0
xor   a4,a4,a1
xori  a4,a4,0x63
andi  a4,a4,0xFF
```

18 instructions

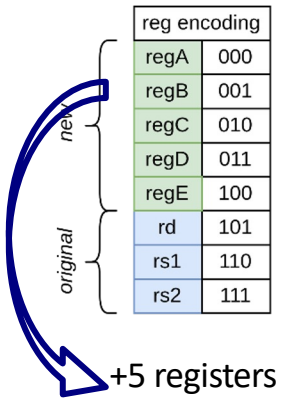
Exemple : AES


18 instructions
srliw a0,a5,0x4
slliw a4,a5,0x4
slliw a2,a5,0x1
srliw a1,a5,0x7
or a4,a4,a0
or a1,a1,a2
xor a4,a4,a1
slliw a0,a5,0x2
srliw a1,a5,0x6
or a0,a0,a1
xor a4,a4,a5
srliw a7,a5,0x5
xor a4,a4,a0
slliw a1,a5,0x3
or a1,a1,a0
xor a4,a4,a1
xori a4,a4,0x63
andi a4,a4,0xFF
1 macro-instruction
custom_instr a5,a4,a3

Exemple : AES


reg encoding
new
original
regA 000
regB 001
regC 010
regD 011
regE 100
rd 101
rs1 110
rs2 111
asm CVA6 rd rs1 rs2 imm
srliw SRLW regA rs1 regA 7
slliw SLLW regC rs1 regA 1
or ORL regB regB regC 0
xor XORL regA rs1 regB 0
srliw SRLW regB rs1 regA 6
slliw SLLW regC rs1 regA 2
or ORL regB regB regC 0
xor XORL regA regA regB 0
srliw SRLW regB rs1 regA 5
slliw SLLW regC rs1 regA 3
or ORL regB regB regC 0
xor XORL regA regA regB 0
srliw SRLW regB rs1 regA 4
slliw SLLW regC rs1 regA 4
or ORL regB regB regC 0
xor XORL regA regA regB 0
xori XORL regA regA regA 99
andi ANDL rd regA regA 255
32 bits
bin
31 28 | 27 20 | 19 17 | 16 14 | 13 11 | 10 1 | 0
fu | op | rd | rs1 | rs2 | imm | skip
0011 00001010 001 110 000 0000000111 1
0011 00001011 010 110 000 0000000001 1
0011 00000101 001 001 010 0000000000 1
0011 00000100 000 110 001 0000000000 1
0011 00001010 001 110 000 0000000110 1
0011 00001011 010 110 000 0000000010 1
0011 00000101 001 001 010 0000000000 1
0011 00000100 000 000 001 0000000000 1
0011 00001010 001 110 000 0000000101 1
0011 00001011 010 110 000 0000000011 1
0011 00000101 001 001 010 0000000000 1
0011 00000100 000 000 001 0000000000 1
0011 00001010 001 110 000 0000000100 1
0011 00001011 010 110 000 0000000100 1
0011 00000101 001 001 010 0000000000 1
0011 00000100 000 000 001 0000000000 1
0011 00000100 000 000 000 0001100011 1
0011 00000110 101 000 000 0011111111 0
... ... 0
ROM @dr
idx x 32 + 0
idx x 32 + 1
idx x 32 + 2
idx x 32 + 15
idx x 32 + 16
idx x 32 + 17
...
idx x 32 + 31
idx = 0

Exemple : AES



asm	CVA6	rd	rs1	rs2	imm
srliw	SRLW	regA	rs1	regA	7
slliw	SLLW	regC	rs1	regA	1
or	ORL	regB	regB	regC	0
xor	XORL	regA	rs1	regB	0
srliw	SRLW	regB	rs1	regA	6
slliw	SLLW	regC	rs1	regA	2
or	ORL	regB	regB	regC	0
xor	XORL	regA	regA	regB	0
srliw	SRLW	regB	rs1	regA	5
slliw	SLLW	regC	rs1	regA	3
or	ORL	regB	regB	regC	0
xor	XORL	regA	regA	regB	0
srliw	SRLW	regB	rs1	regA	4
slliw	SLLW	regC	rs1	regA	4
or	ORL	regB	regB	regC	0
xor	XORL	regA	regA	regB	0
xori	XORL	regA	regA	regA	99
andi	ANDL	rd	regA	regA	255

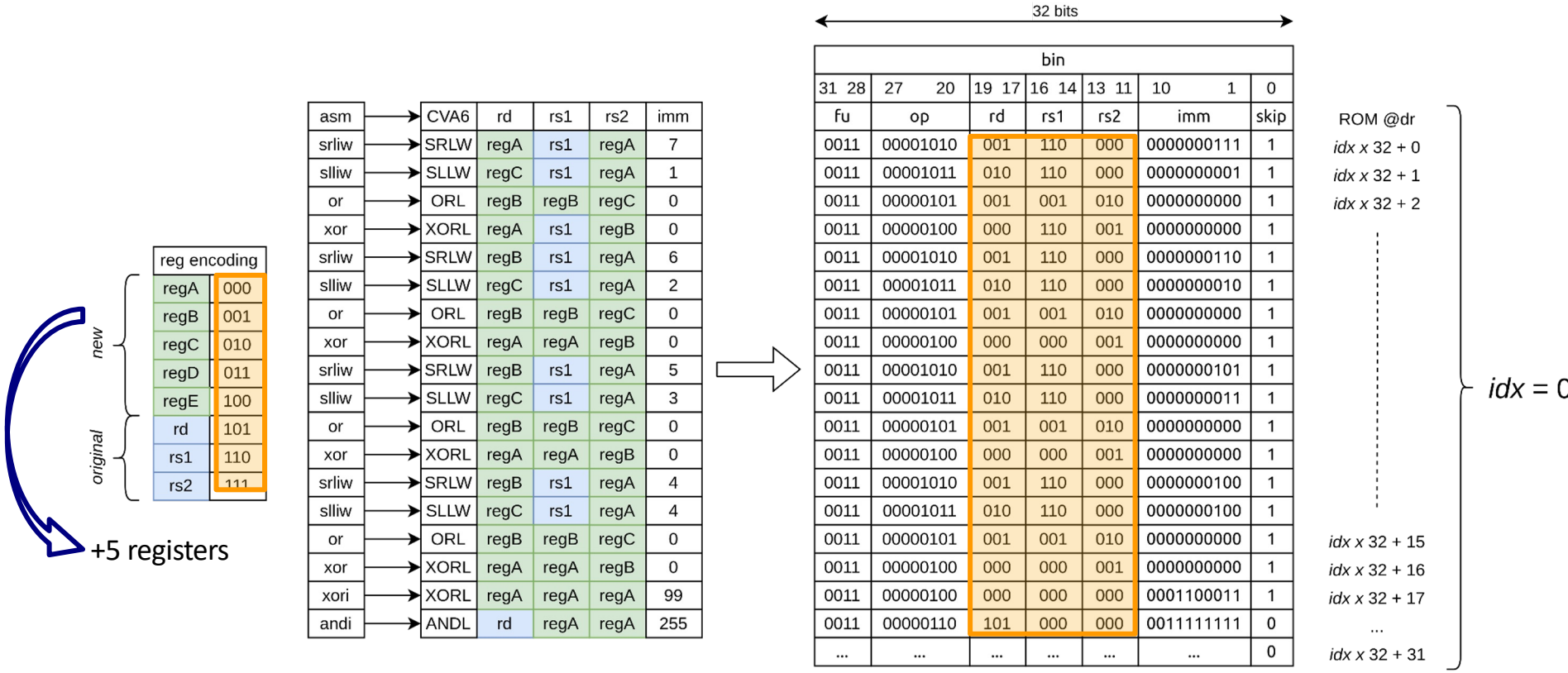
32 bits

bin						
31 28	27 20	19 17	16 14	13 11	10 1	0
fu	op	rd	rs1	rs2	imm	skip
0011	00001010	001	110	000	0000000111	1
0011	00001011	010	110	000	0000000001	1
0011	00000101	001	001	010	0000000000	1
0011	00000100	000	110	001	0000000000	1
0011	00001010	001	110	000	0000000110	1
0011	00001011	010	110	000	0000000010	1
0011	00000101	001	001	010	0000000000	1
0011	00000100	000	000	001	0000000000	1
0011	00001010	001	110	000	0000000101	1
0011	00001011	010	110	000	0000000011	1
0011	00000101	001	001	010	0000000000	1
0011	00000100	000	000	001	0000000000	1
0011	00001010	001	110	000	0000000100	1
0011	00001011	010	110	000	0000000100	1
0011	00000101	001	001	010	0000000000	1
0011	00000100	000	000	001	0000000000	1
0011	00000100	000	000	000	0001100011	1
0011	00000110	101	000	000	0011111111	0
...	...	...	...	...	...	0

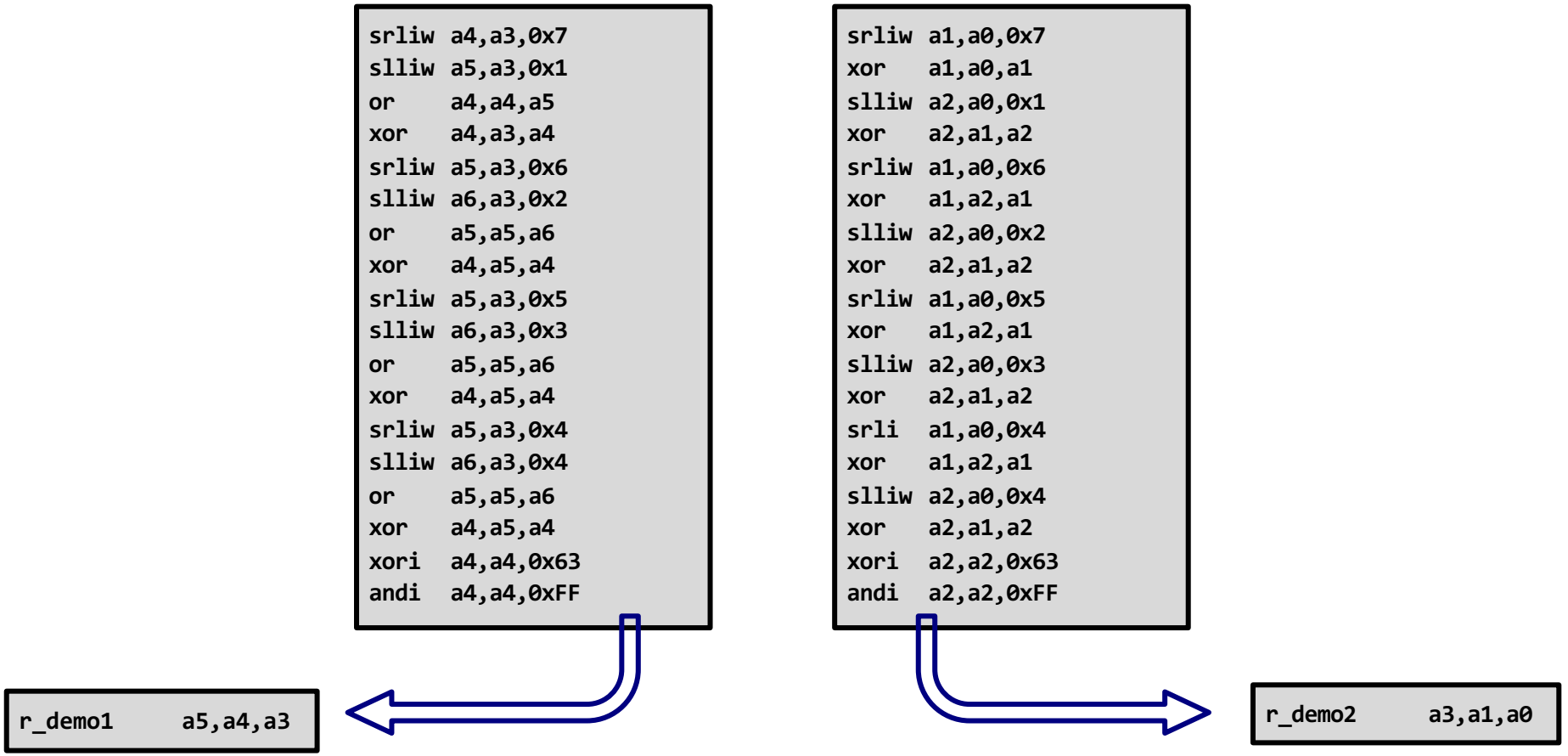
ROM @dr
idx x 32 + 0
idx x 32 + 1
idx x 32 + 2
...
idx x 32 + 15
idx x 32 + 16
idx x 32 + 17
...
idx x 32 + 31

idx = 0

Exemple : AES


reg encoding
regA 000
regB 001
regC 010
regD 011
regE 100
rd 101
rs1 110
rs2 111
new
original
+5 registers
asm CVA6 rd rs1 rs2 imm
srliw SRLW regA rs1 regA 7
slliw SLLW regC rs1 regA 1
or ORL regB regB regC 0
xor XORL regA rs1 regB 0
srliw SRLW regB rs1 regA 6
slliw SLLW regC rs1 regA 2
or ORL regB regB regC 0
xor XORL regA regA regB 0
srliw SRLW regB rs1 regA 5
slliw SLLW regC rs1 regA 3
or ORL regB regB regC 0
xor XORL regA regA regB 0
srliw SRLW regB rs1 regA 4
slliw SLLW regC rs1 regA 4
or ORL regB regB regC 0
xor XORL regA regA regB 0
xori XORL regA regA regA 99
andi ANDL rd regA regA 255
32 bits
bin
31 28 | 27 20 | 19 17 | 16 14 | 13 11 | 10 1 | 0
fu | op | rd | rs1 | rs2 | imm | skip
0011 00001010 001 110 000 0000000111 1
0011 00001011 010 110 000 0000000001 1
0011 00000101 001 001 010 0000000000 1
0011 00000100 000 110 001 0000000000 1
0011 00001010 001 110 000 0000000110 1
0011 00001011 010 110 000 0000000010 1
0011 00000101 001 001 010 0000000000 1
0011 00000100 000 000 001 0000000000 1
0011 00001010 001 110 000 0000000101 1
0011 00001011 010 110 000 0000000011 1
0011 00000101 001 001 010 0000000000 1
0011 00000100 000 000 001 0000000000 1
0011 00001010 001 110 000 0000000100 1
0011 00001011 010 110 000 0000000100 1
0011 00000101 001 001 010 0000000000 1
0011 00000100 000 000 001 0000000000 1
0011 00000100 000 000 000 0001100011 1
0011 00000110 101 000 000 0011111111 0
... ... 0
ROM @dr
idx x 32 + 0
idx x 32 + 1
idx x 32 + 2
idx x 32 + 15
idx x 32 + 16
idx x 32 + 17
...
idx x 32 + 31
idx = 0

Exemple : AES


srliw a4,a3,0x7
slliw a5,a3,0x1
or a4,a4,a5
xor a4,a3,a4
srliw a5,a3,0x6
slliw a6,a3,0x2
or a5,a5,a6
xor a4,a5,a4
srliw a5,a3,0x5
slliw a6,a3,0x3
or a5,a5,a6
xor a4,a5,a4
srliw a5,a3,0x4
slliw a6,a3,0x4
or a5,a5,a6
xor a4,a5,a4
xori a4,a4,0x63
andi a4,a4,0xFF
srliw a1,a0,0x7
xor a1,a0,a1
slliw a2,a0,0x1
xor a2,a1,a2
srliw a1,a0,0x6
xor a1,a2,a1
slliw a2,a0,0x2
xor a2,a1,a2
srliw a1,a0,0x5
xor a1,a2,a1
slliw a2,a0,0x3
xor a2,a1,a2
srli a1,a0,0x4
xor a1,a2,a1
slliw a2,a0,0x4
xor a2,a1,a2
xori a2,a2,0x63
andi a2,a2,0xFF
r_demo1 a5,a4,a3
r_demo2 a3,a1,a0

Environnement d'expérimentations

➔ **Comparaison CVA6 vierge / enrichi**

➔ **1 – Evaluation du surcoût matériel :**

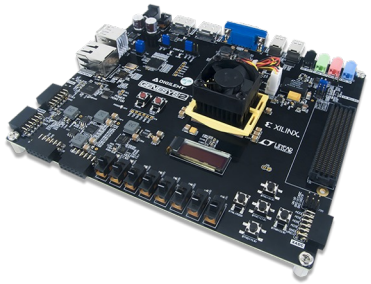
- Vivado 2020.2
- FPGA Kintex-7 (Genesys II de Digilent)
- 2 versions (P=2 puis P=64)

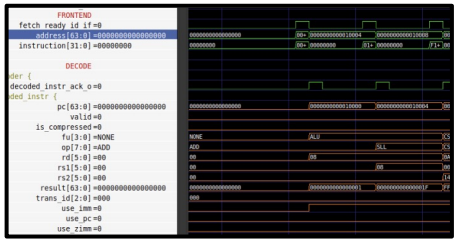
➔ **2 – Mesure de l'impact sur les performances temporelles :**

- Verilator v4.110
- Riscv64-unknown-elf-gcc v12.1.0

➔ **Protocole de mesures :**

- 10 mesures par application
- Mesure via l'instruction *rdcycle*
- Application simulées en version *baremetal* puis exécutées sur Linux embarqué





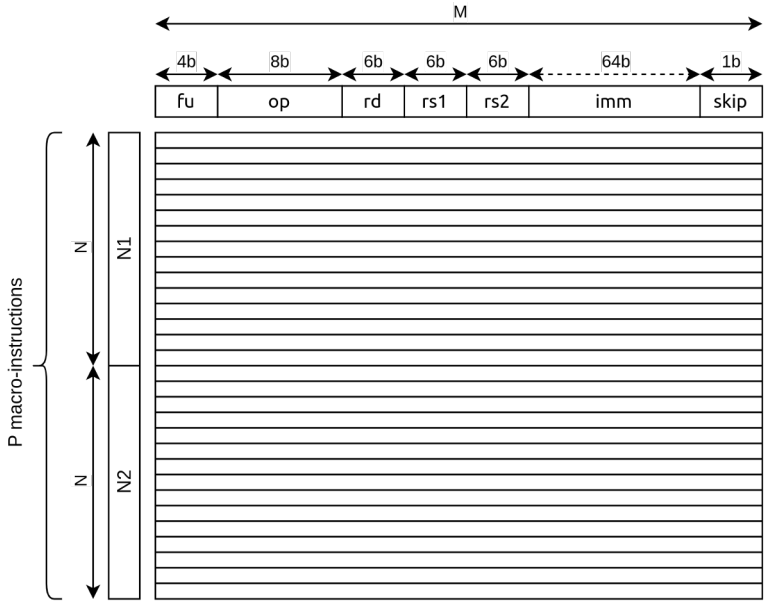
Evaluation du surcoût matériel

➔ Complexité matérielle (M=32, **P=2,** N=32):

	LUT	SRL	FF	BRAM36	BRAM18	DSP
Coeur vierge	44 148	0	24 076	36	0	27
Coeur enrichi (2 instr.)	45 880 +3,9%	0 -	25 078 +4,2%	36 -	0 -	27 -

➔ Complexité matérielle (M=32, **P=64**, N=32):

	LUT	SRL	FF	BRAM36	BRAM18	DSP
Coeur vierge	44 148	0	24 076	36	0	27
Coeur enrichi (64 instr.)	45 937 +4,1%	0 -	25 068 +4,1%	38 +5,6%	0 -	27 -



Evaluation temporelle

		Temps d'exécution (cycles d'horloge)	
		CVA6 d'origine	**CVA6 + SECV**
Simulation Verilator	**Sans microdécodage**	55564	55562
	Avec microdécodage		47145 (-15%)
FPGA+Linux	**Sans microdécodage**	68859	68048 (-1.2%)
	Avec microdécodage		47554 (-15%)

SEC-V

Résultats

Architecture open source sécurisée et hautes performances

Réalisations

- Description RTL de l'architecture
- Intégration dans le cœur CVA6 de l'OpenHW Group
- Flot de conception
- Evaluations de la sécurité des mécanismes proposés

SEC-V

Secure-V Project

Secure RISC-V processor

Merci pour votre attention !
Questions ?

https://secv.univ-nantes.fr

Sébastien Pillement, Maria Méndez Real, Juliette Pottier, Bertrand Le Gal, Thomas Nieddu, Sébastien Faucou, Jean-Luc Bechennec, Mickael Briday, André Sintzoff, Jimmy Le Rhun, Olivier Gilles, Guillaume Bouffard, Philippe Trebuchet

SEC-V