



面試簡報

面試者：蘇柏丞



01

個人簡介

02

教育性晶片下線經驗

03

論文介紹

04

其他作品



01

個人簡介



► 個人簡介



姓 名：蘇柏丞

生 日：2001/07/13

信 箱：fossum2523@gmail.com

電 話：0981-933-963

學 歷：臺灣科技大學電子工程學系

指導教授：林銘波教授

實 驗 室：數位積體電路實驗室

經 歷：碩士班榮獲電子所碩士優秀獎學金

第十七屆數位訊號處理創思設計競賽 入圍

相關課程：NTU → 系統晶片設計實驗(A+)、高等數位訊號處理(A+)

NTUST → FPGA系統設計實務(A+)、超大型積體電路設計(A+)、
高等計算機演算法(A-)、超大型積體電路測試(修習中)

► 相關專題

➤ 碩士：

- 碩士論文 - 設計與實現一個相容於AXI-4介面的後量子密碼學ML-DSA之硬體加速器IP
- T18晶片下線 - 設計與實現同步AES-128加密IP
- SoC 期末專題
- 16-bit RISC CPU

➤ 大學：

- T18晶片下線 -設計與實現以CORDIC為基礎之FFT IP
- 利用ECG完成具身分辨識之血壓計

► 專業技能

程 式 語 言

- Verilog
- C
- Python
- Assembly Language

使 用 工 具

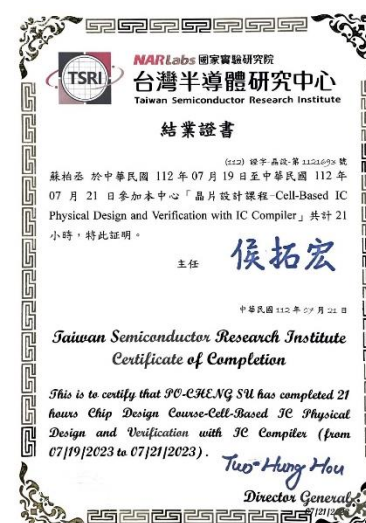
- Design Compiler
- IC Compiler
- Vivado
- VCS
- Verdi
- NC-Verilog

英 文 能 力

- TOEIC **675**



- TSRI Cell-Based IC Physical Design and Verification with IC Compiler (21 hours)
- TSRI Logic Synthesis with Design Compiler (21 hours)



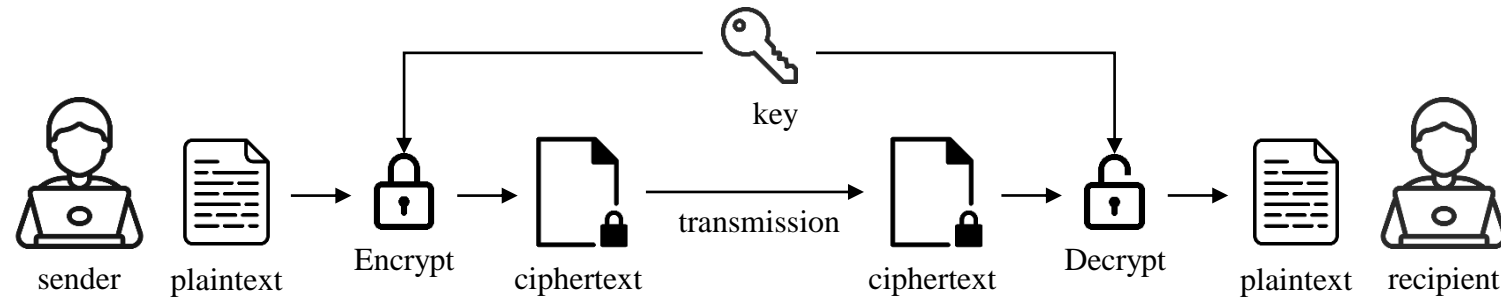


02

教育性晶片下線經驗

► 設計與實現同步AES-128加密IP

- ✓ Symmetric encryption
- ✓ Mathematical operations are relatively less complicated, lower hardware costs, and faster speed

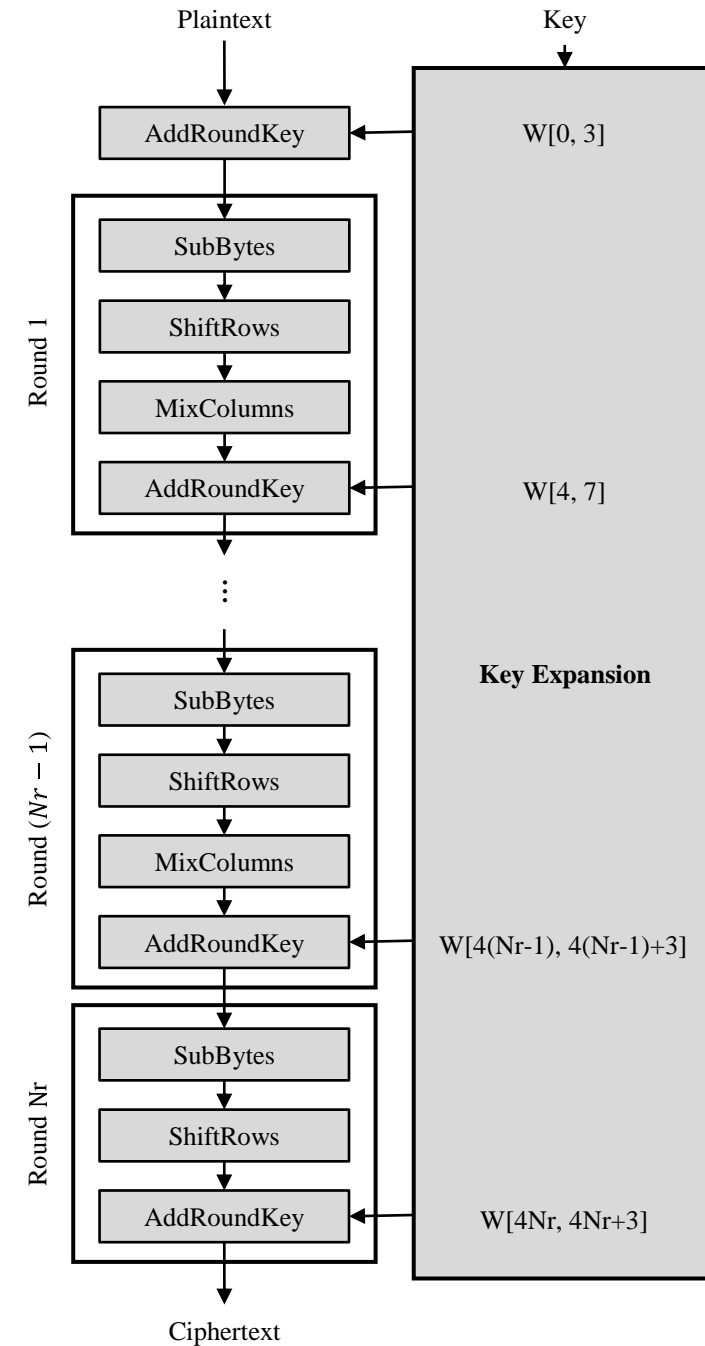


► AES-128 Algorithm

AES-128	
Key size	128 bits
Plaintext block size	128 bits
Number of rounds	10
Round key size	128 bits
Number of expanded keys	44

Two important parts :

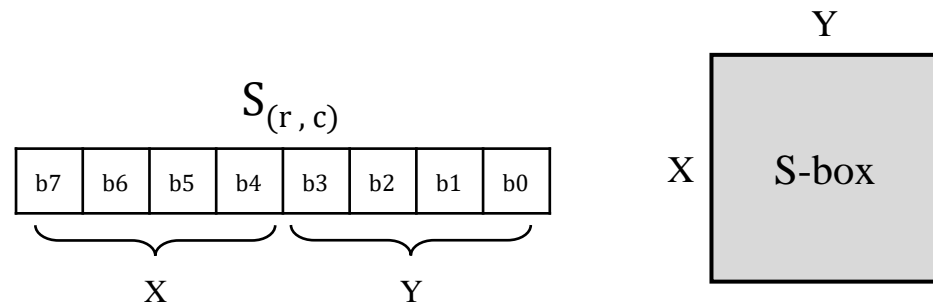
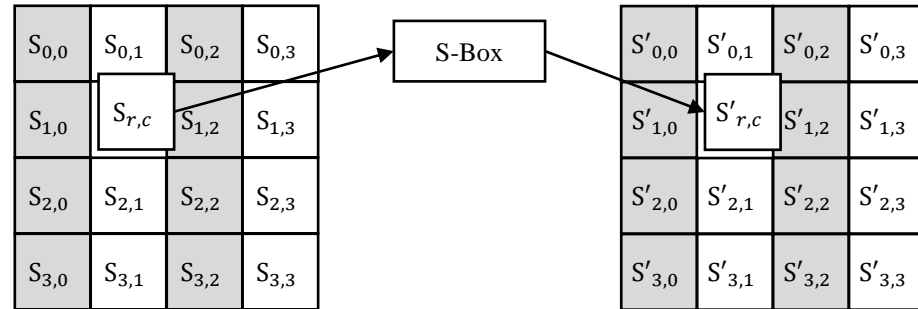
- ✓ Key expansion
- ✓ Message encryption



► AES-128 Algorithm – Substitute Bytes

Way-1 Look-up Functionality

Direct Logical Mapping

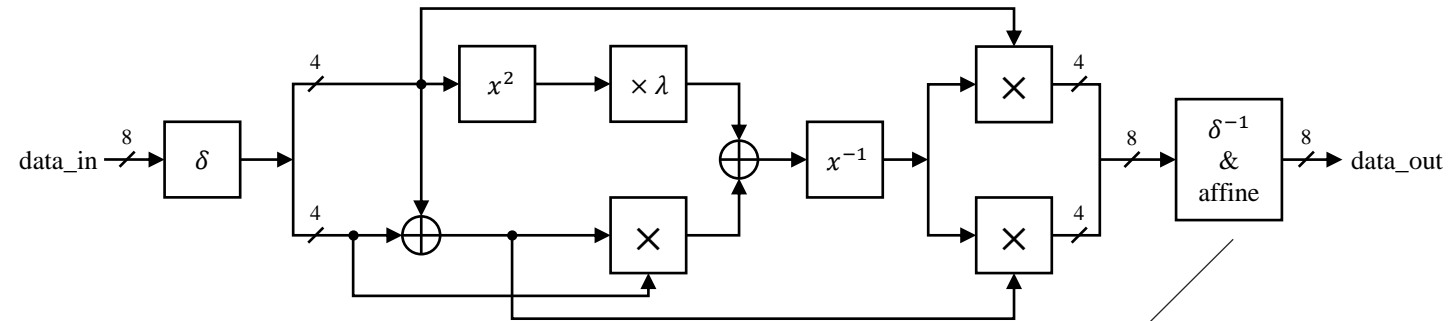


S-box		Column															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
Row	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e7	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

► AES-128 Algorithm – Substitute Bytes

Way-2 Composite Field Operation

Multiplicative Inverse + Affine Transformation



$$\delta^{-1} \times affine = \begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

► AES-128 Algorithm – Substitute Bytes

Direct Logical Mapping :

✓ Area

```
Total cell area:      5604.984089
Total area:           54651.896534
1
```

✓ Timing

```
-----
data required time      2.05
data arrival time      -2.60
-----
slack (VIOLATED)       -0.55
```


Composite Field Operation :

✓ Area

```
Total cell area:      3961.742433
Total area:           25708.518007
1
```

✓ Timing

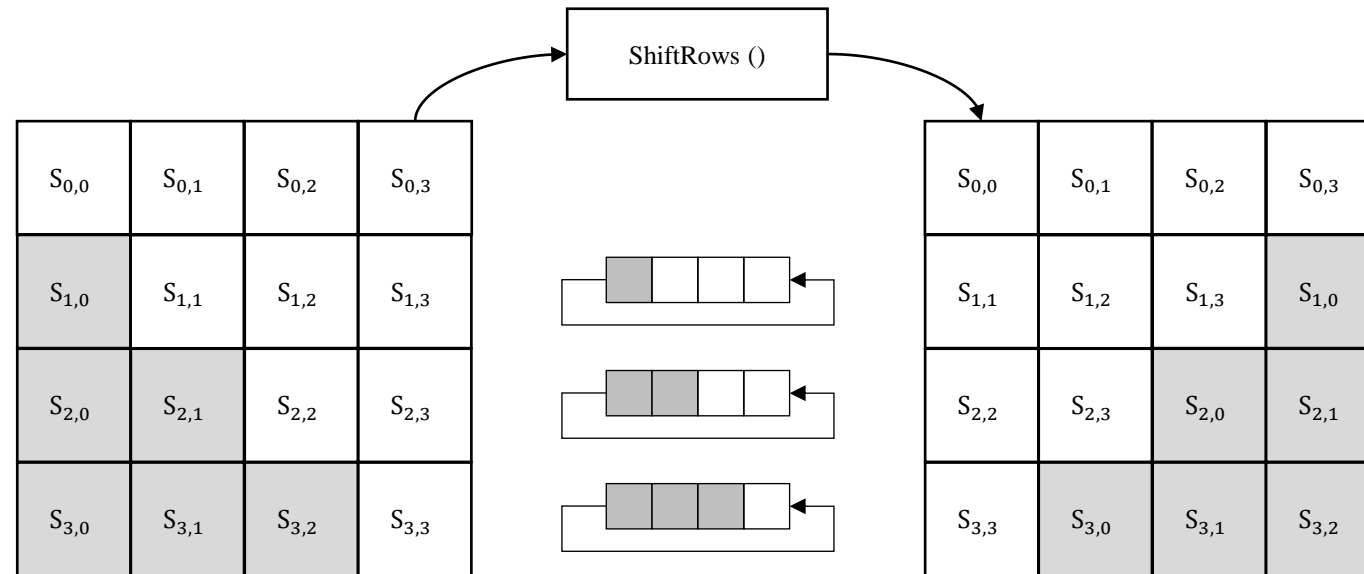
```
-----
data required time      2.06
data arrival time      -4.61
-----
slack (VIOLATED)       -2.55
```

TSMC 0.18 μm Cell-base Library		
Method	Direct Logical Mapping	Composite Field Operation
Gate Count	 562	397
Propagation Delay	2.60 ns	4.61 ns

► AES-128 Algorithm – Shift Rows

$$S'_{r,c} = S_{r,(c+shift(r,Nb)) \bmod Nb} \quad \text{for } 0 < r < 4 \text{ and } 0 \leq c < Nb \text{ and } Nb = 4$$

$$shift(1,4) = 1; \quad shift(2,4) = 2; \quad shift(3,4) = 3;$$

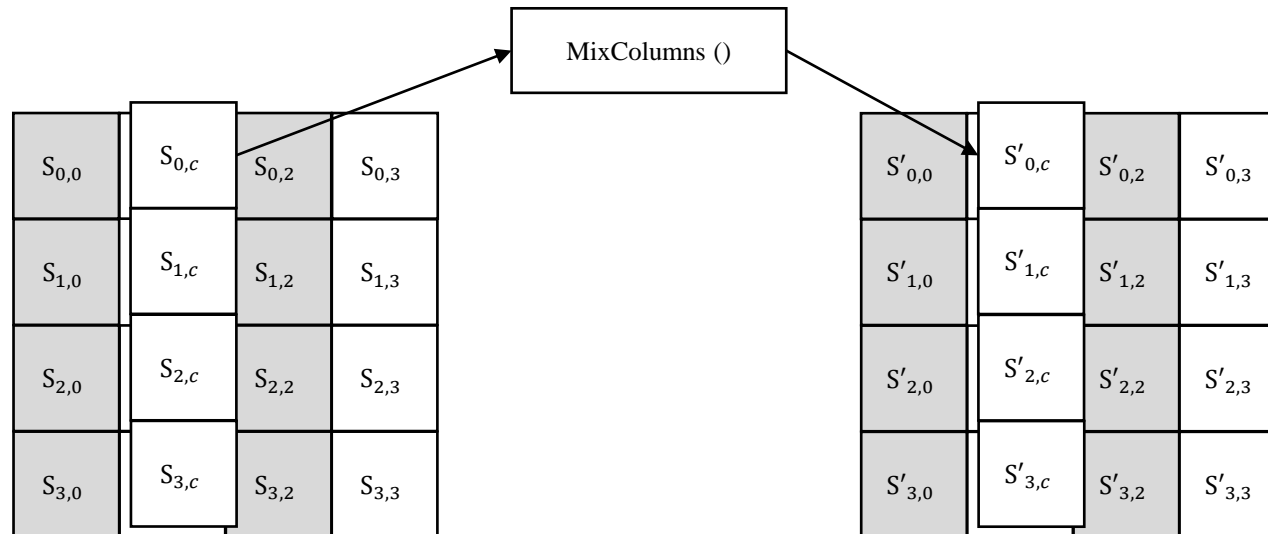


► AES Algorithm – Mix Columns

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

$$S'(x) = a(x) \otimes S(x)$$

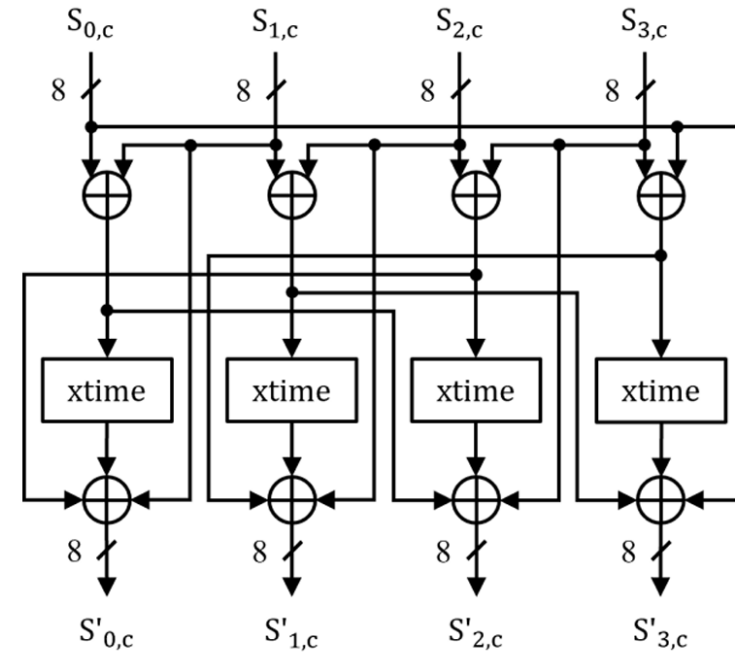
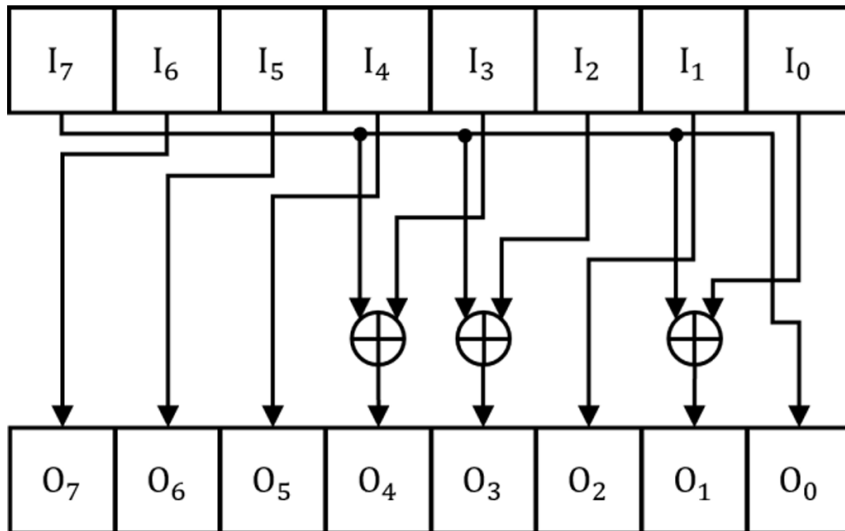
$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \quad \text{for } 0 \leq c < Nb$$



► AES-128 Algorithm – Mix Columns

$$\begin{cases} S'_{0,c} = (\{02\} \cdot S_{0,c}) \oplus (\{03\} \cdot S_{1,c}) \oplus S_{2,c} \oplus S_{3,c} \\ S'_{1,c} = S_{0,c} \oplus (\{02\} \cdot S_{1,c}) \oplus (\{03\} \cdot S_{2,c}) \oplus S_{3,c} \\ S'_{2,c} = S_{0,c} \oplus S_{1,c} \oplus (\{02\} \cdot S_{2,c}) \oplus (\{03\} \cdot S_{3,c}) \\ S'_{3,c} = (\{03\} \cdot S_{0,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus (\{02\} \cdot S_{3,c}) \end{cases}$$

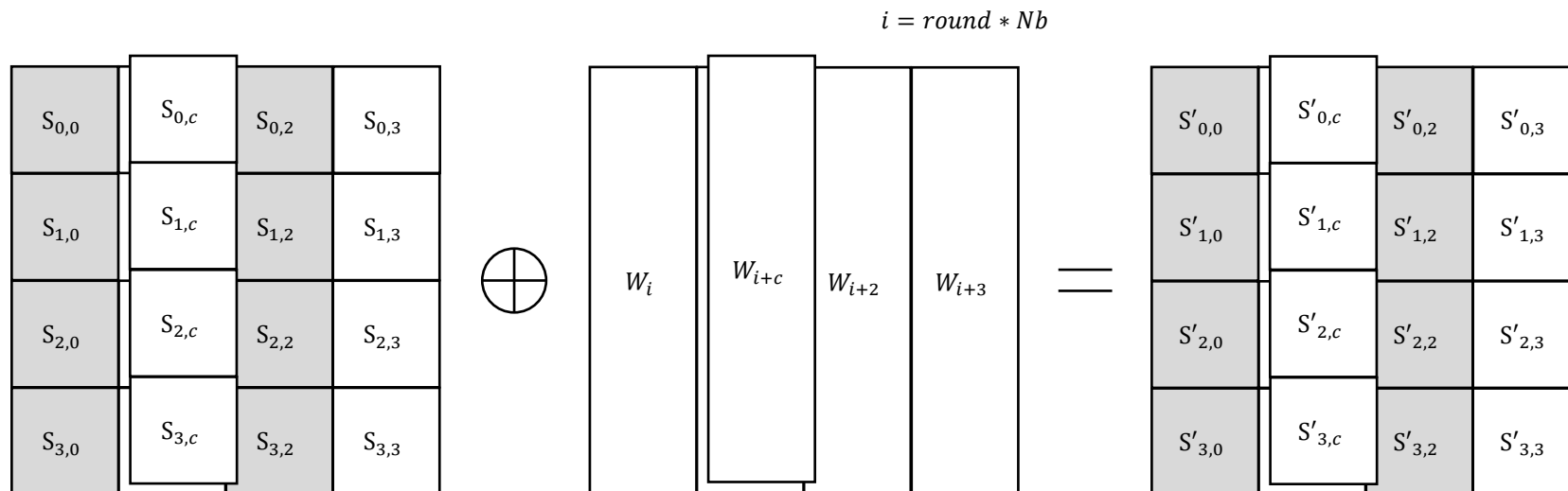
$$\begin{cases} S'_{0,c} = \{02\} \cdot (S_{0,c} \oplus S_{1,c}) \oplus \{01\} \cdot (S_{1,c} \oplus (S_{2,c} \oplus S_{3,c})) \\ S'_{1,c} = \{02\} \cdot (S_{1,c} \oplus S_{2,c}) \oplus \{01\} \cdot (S_{2,c} \oplus (S_{0,c} \oplus S_{3,c})) \\ S'_{2,c} = \{02\} \cdot (S_{2,c} \oplus S_{3,c}) \oplus \{01\} \cdot (S_{3,c} \oplus (S_{0,c} \oplus S_{1,c})) \\ S'_{3,c} = \{02\} \cdot (S_{0,c} \oplus S_{3,c}) \oplus \{01\} \cdot (S_{0,c} \oplus (S_{1,c} \oplus S_{2,c})) \end{cases}$$



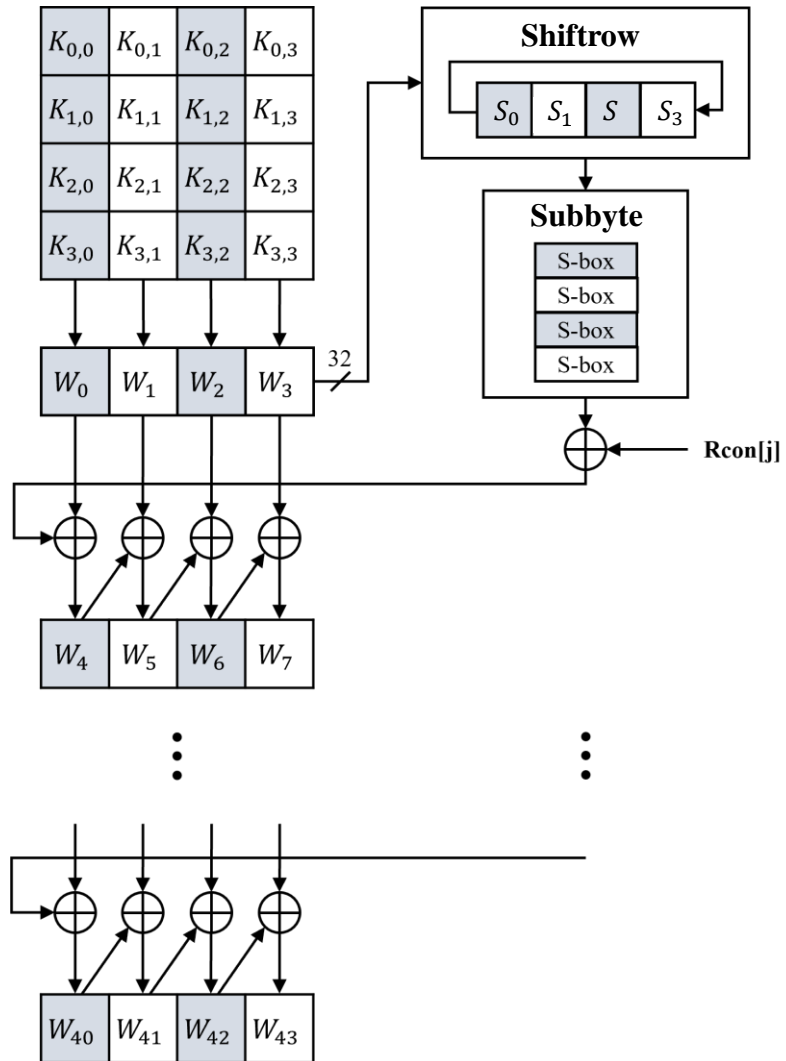
► AES-128 Algorithm – Add Round Keys

$$[S'_{0,c}, S'_{1,c}, S'_{2,c}, S'_{3,c}] = [S_{0,c}, S_{1,c}, S_{2,c}, S_{3,c}] \oplus [W_{round*Nb+c}]$$

for $0 \leq c < Nb$, $0 \leq round < Nr$

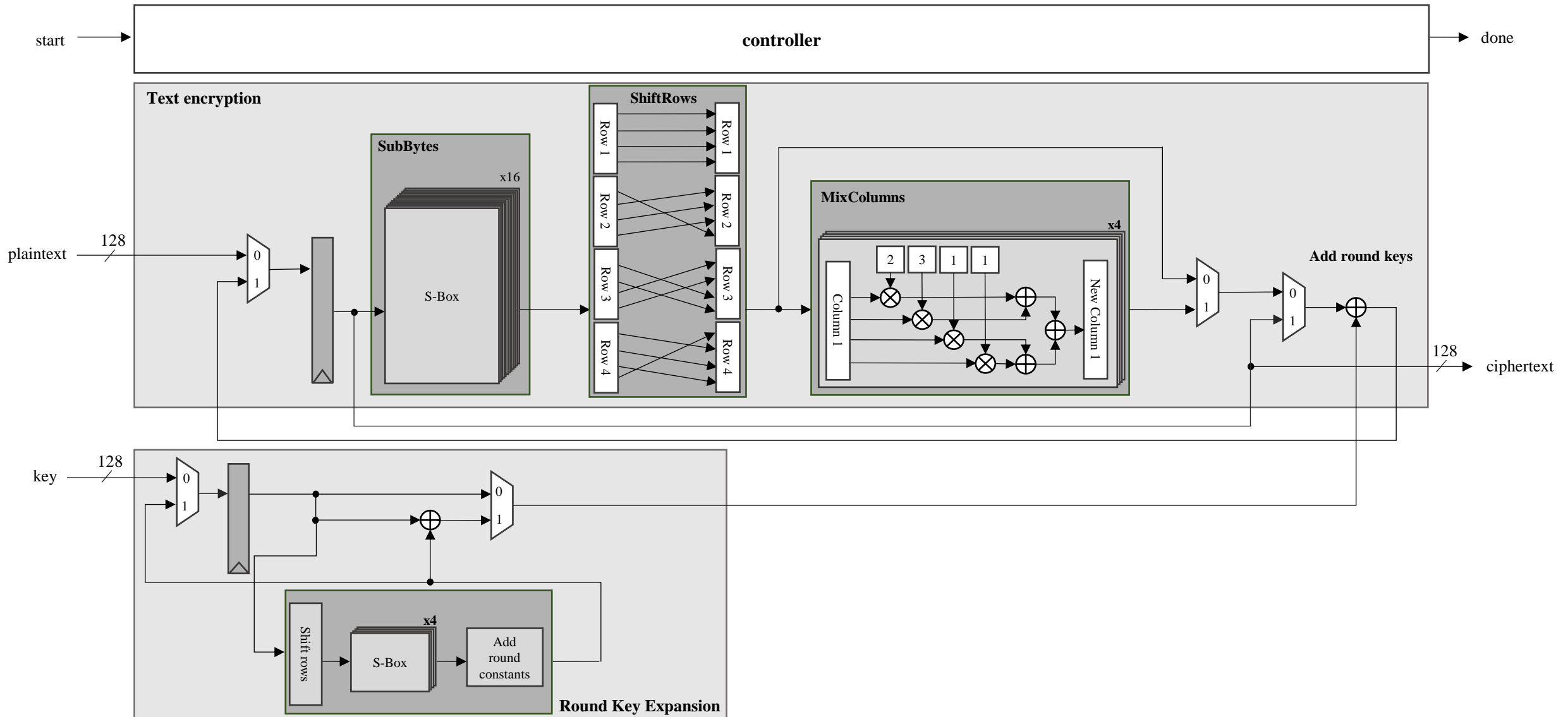


► AES-128 Algorithm – Key Expansion



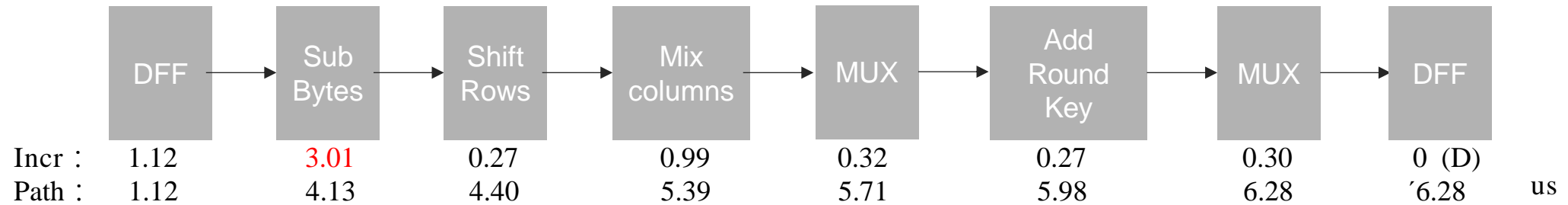
$$\begin{cases} Rcon[i] = \{temp[i], 0, 0, 0\} \\ temp[i] = 2 \cdot temp[i - 1] \\ temp[1] = 1 \end{cases} \quad \text{for } 1 \leq i \leq Nr$$

► AES-128 Block Diagram

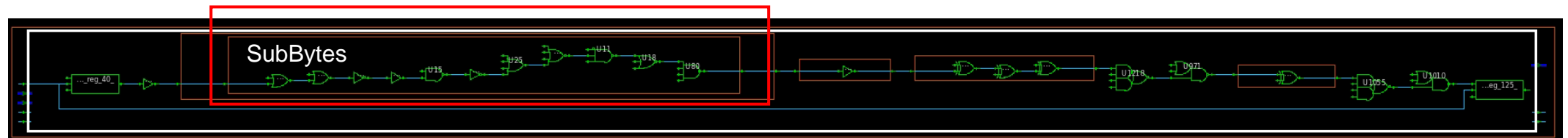


► AES-128 Critical Path

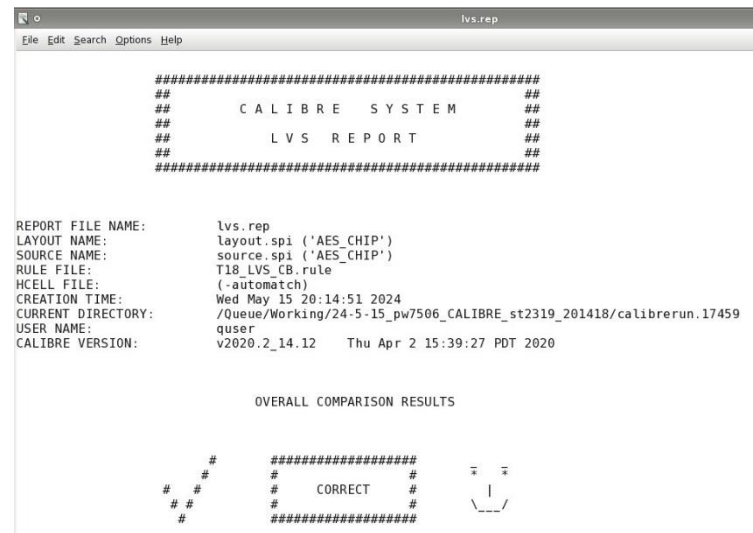
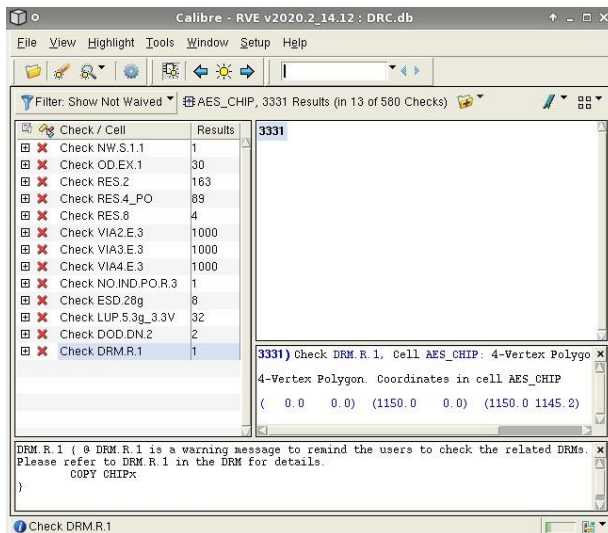
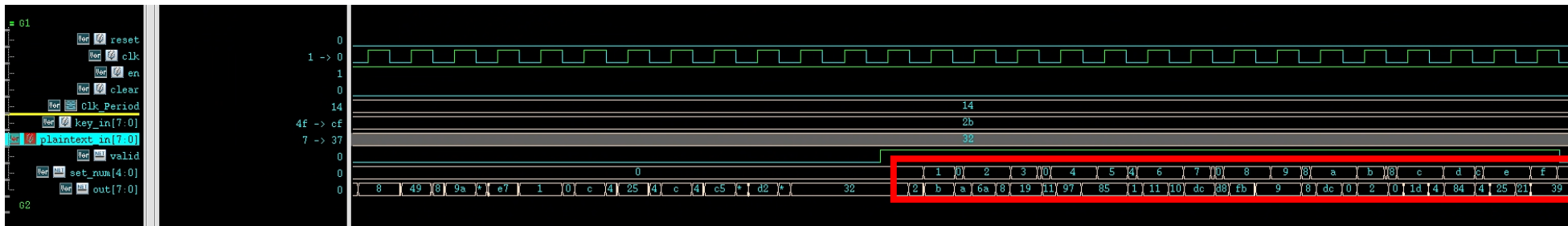
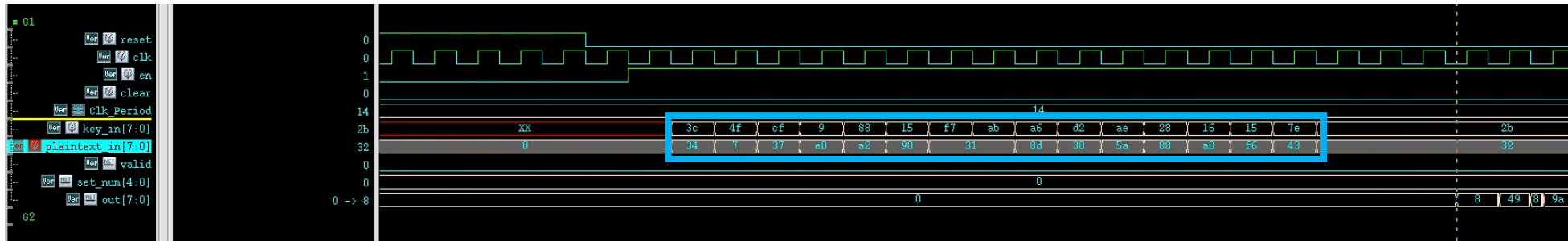
✓ Individual Function Delay :



✓ Synthesized Actual Path :

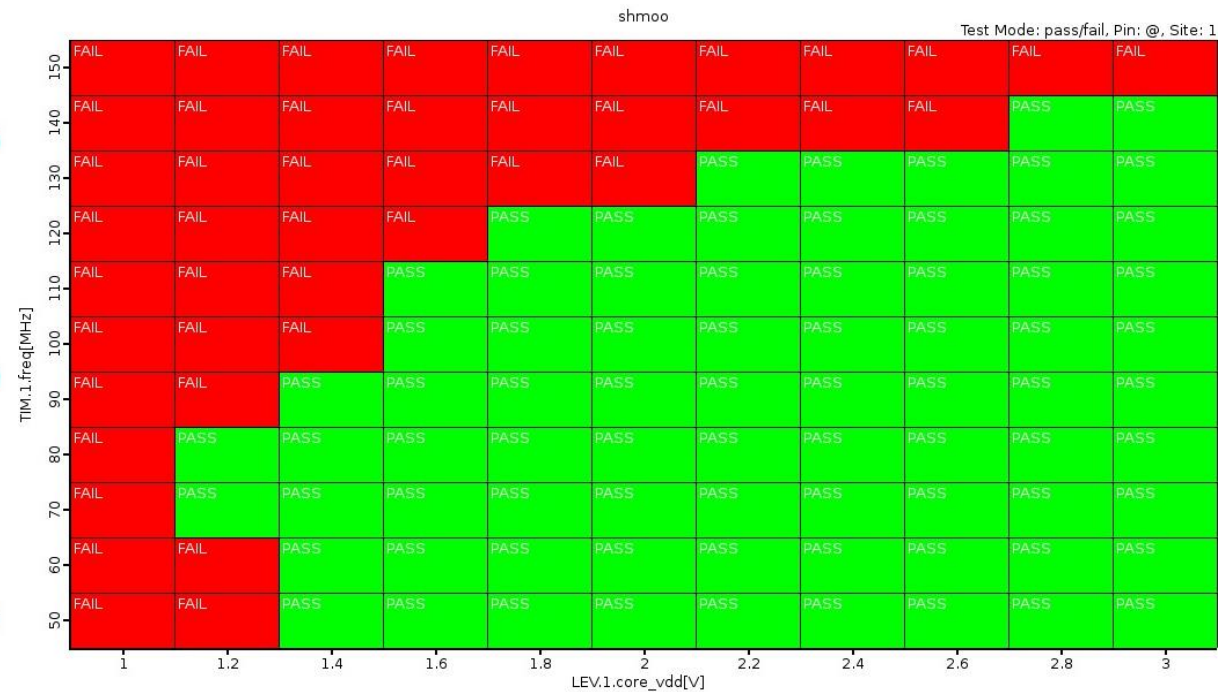
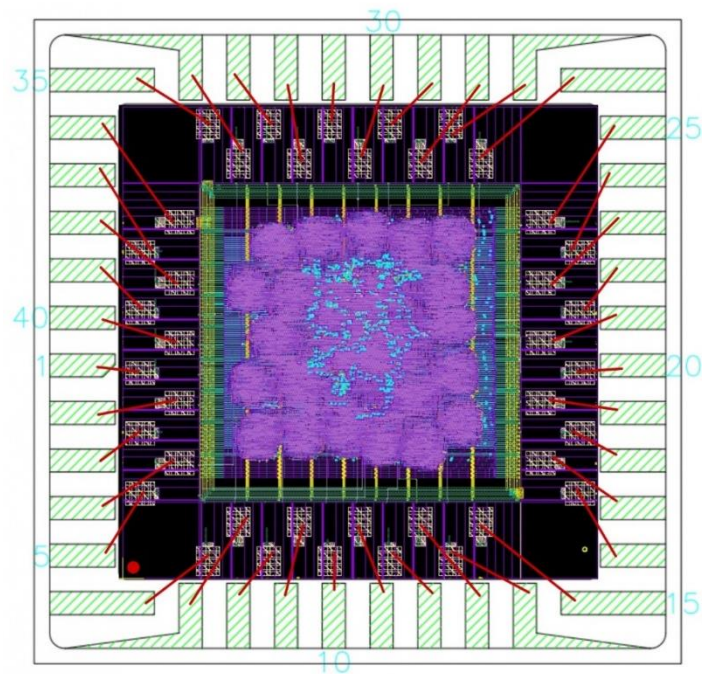


► AES-128 Verification Result



► AES-128 Result

Process	TSMC 0.18um
Package	SB40
Testing results	Pass
Frequency	120 MHz
Power	8.6103 mW
Area	1.145 x 1.150 mm ²





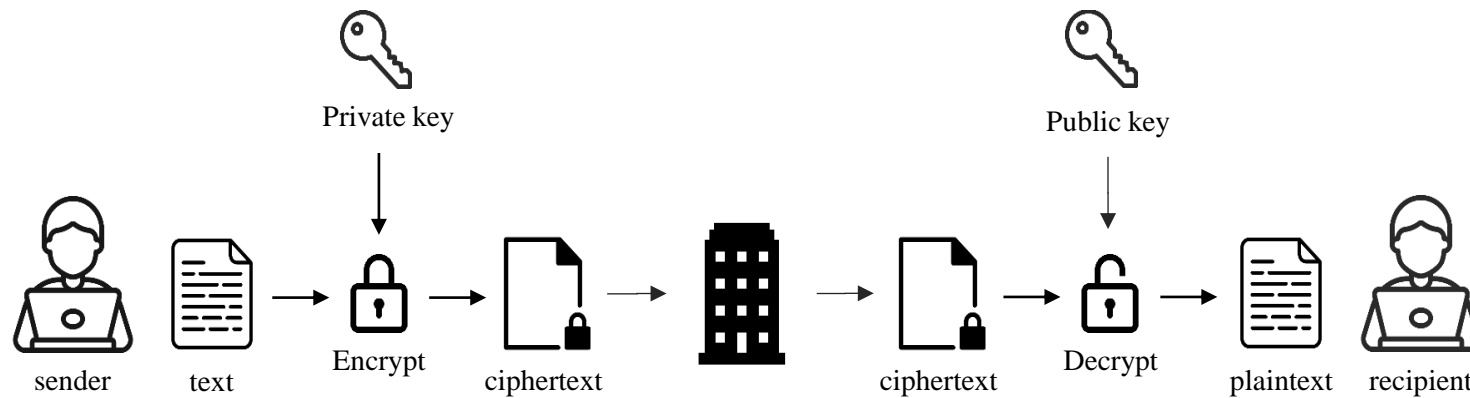
03

論文介紹



► 設計與實現基於AXI-4介面的後量子密碼學ML-DSA之硬體加速器

- ✓ **Shor's algorithm**, combined with a powerful **quantum computer**, will break RSA and ECC
- ✓ Defined a method for generating **Digital Signatures**
- ✓ The core security challenges of ML-DSA include **MLWE** and **MSIS**, it has potential resistance against both quantum and classical attacks
- ✓ Advantages include **fast arithmetic operations**, **efficient encryption**, and **compact signatures**



► MLWE (module learning with errors)

Setup:

1. Modulus $q=7$.
2. Matrix A is of size 2×2 , with elements selected randomly.
3. Secret vectors s_1 and s_2 are both of size 2×1 .
4. Values for A , s_1 , s_2 :

$$A = \begin{bmatrix} 3 & 4 \\ 1 & 5 \end{bmatrix}, \quad s_1 = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \quad s_2 = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

Calculation Steps:

1. Calculate As_1 :

$$As_1 = \begin{bmatrix} 3 & 4 \\ 1 & 5 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 3 \cdot 2 + 4 \cdot 3 \\ 1 \cdot 2 + 5 \cdot 3 \end{bmatrix} = \begin{bmatrix} 6 + 12 \\ 2 + 15 \end{bmatrix} = \begin{bmatrix} 18 \\ 17 \end{bmatrix}$$

2. Add the secret vector s_2 to the result and take

modulus q :

$$t = As_1 + s_2 = \begin{bmatrix} 18 \\ 17 \end{bmatrix} + \begin{bmatrix} 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 19 \\ 21 \end{bmatrix}$$

$$t = \begin{bmatrix} 19 \bmod 7 \\ 21 \bmod 7 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$$

3. The public data is the matrix A and the result

vector t :

$$A = \begin{bmatrix} 3 & 4 \\ 1 & 5 \end{bmatrix}, \quad t = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$$

►MSIS (module shortest integer solution)

Setup:

1. Modulus $q = 7$
2. Matrix A is of size 2×2 , with elements selected randomly.
3. Values for A :

$$A = \begin{bmatrix} 5 & 7 \\ 3 & 2 \end{bmatrix}$$

Goal:

Find vectors z and u such that $Az + u = 0 \bmod q$.

Attempt to Solve:

1. Assume a vector z, u :

$$z = \begin{bmatrix} 2 \\ -1 \end{bmatrix}, \quad u = \begin{bmatrix} -3 \\ 5 \end{bmatrix}$$

2. Calculate $Az + u$ and take modulus q :

$$Az = \begin{bmatrix} 5 & 7 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \end{bmatrix} = \begin{bmatrix} 5 \cdot 2 + 7 \cdot (-1) \\ 3 \cdot 2 + 2 \cdot (-1) \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

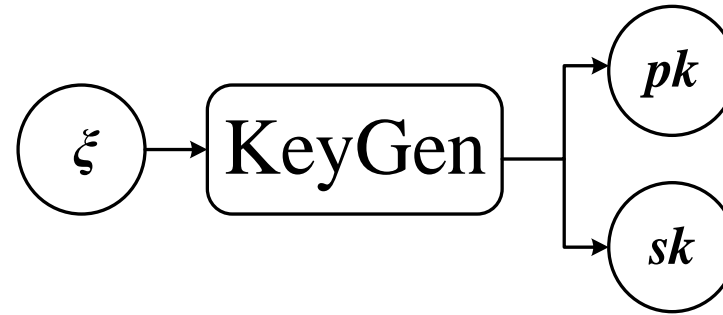
$$Az + u = \begin{bmatrix} 3 \\ 4 \end{bmatrix} + \begin{bmatrix} -3 \\ 5 \end{bmatrix} = \begin{bmatrix} 0 \\ 9 \end{bmatrix}$$

$$Az + u \bmod 7 = \begin{bmatrix} 0 \bmod 7 \\ 9 \bmod 7 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

► Algorithms

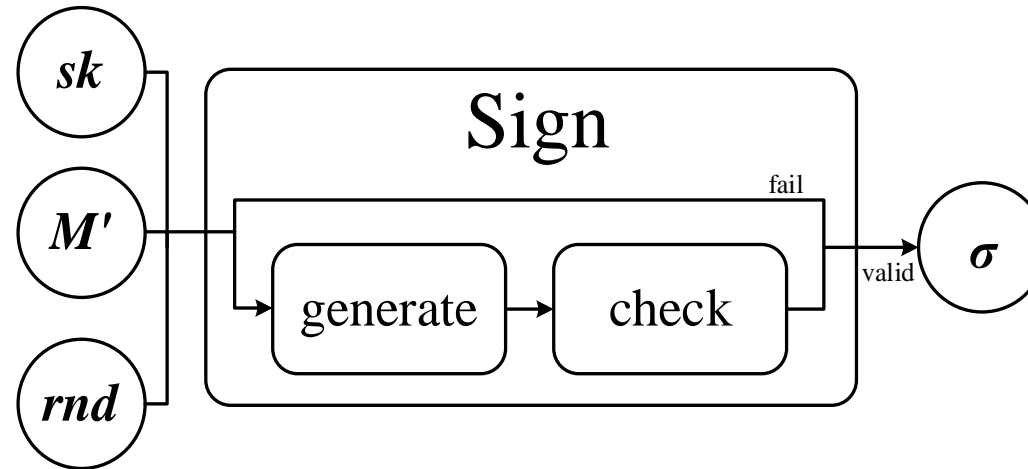
1. Key generation (KeyGen)

➤ MLWE

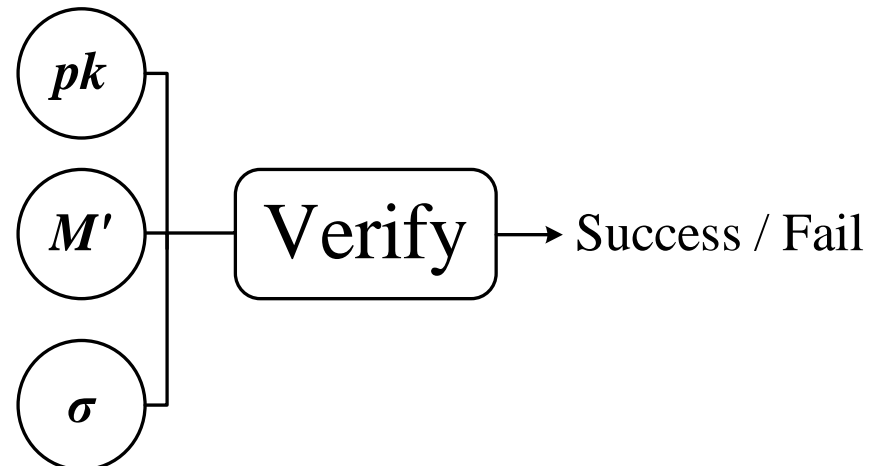


2. Signature generation (Sign)

➤ MSIS



3. Signature verification (Verify)



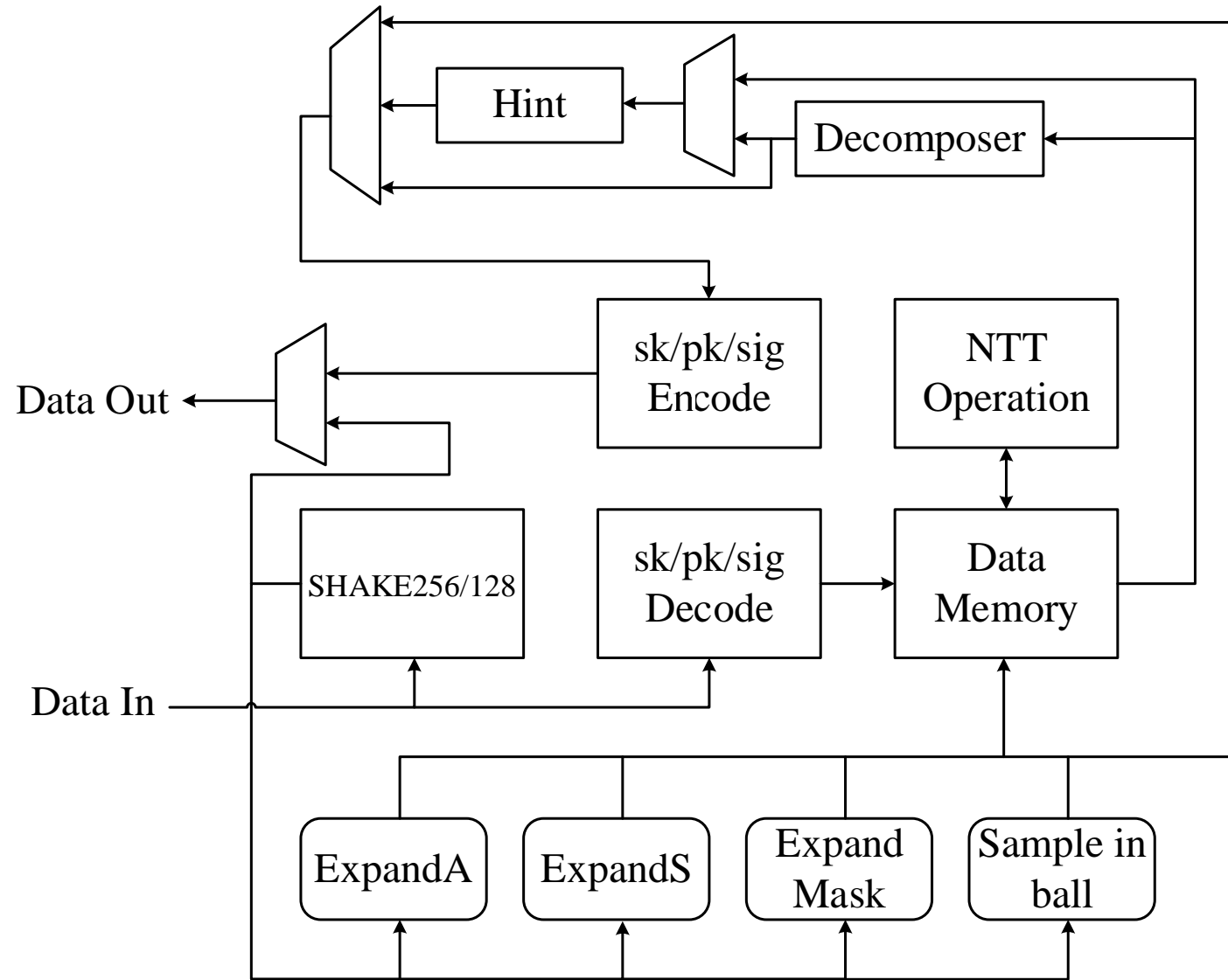
Symbols

- ξ : random seed
- pk : public key
- sk : secret key
- M' : hash message
- rnd : random number
- σ : signature

► Expected Results

- ✓ Implement the circuit on ASIC and FPGA
 - FPGA - XC7VX330T (Xilinx Virtex 7)
 - ASIC - TSMC 0.18 μm Standard Cell Library
- ✓ Implement ML-DSA-44 with area optimization as the top priority
 - Use memory to implement data access
- ✓ Improve the overall utilization of the circuit
 - Combine the KeyGen, Sign, and Verify modules into one
 - NTT implements using the Multi-Delay Path Commutator architecture
 - Shake256 and Shake128 simplify logic to share the same hardware.

► Block Diagram



► NTT Architecture

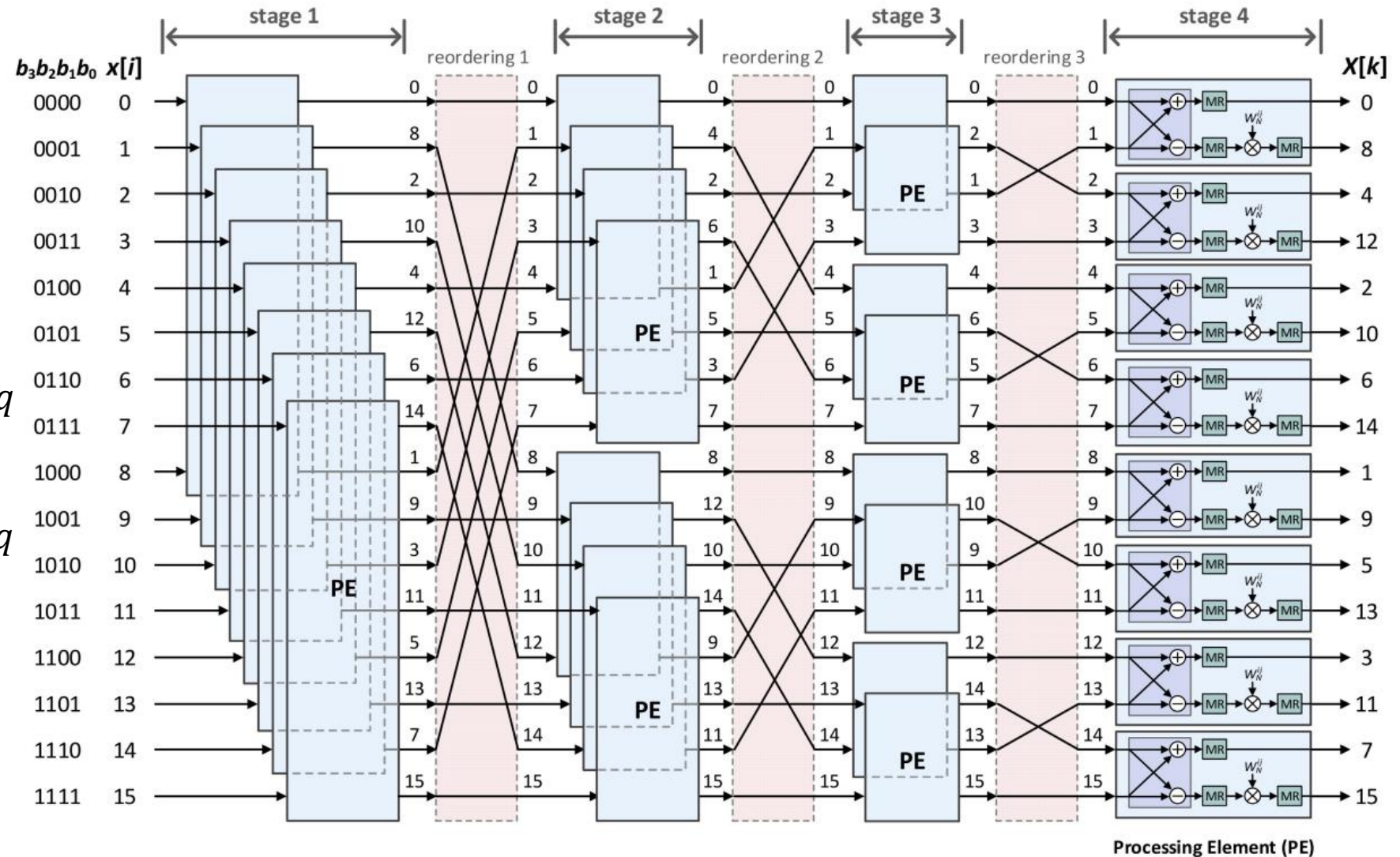
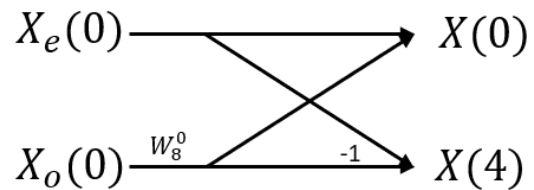
1. Hierarchical :

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \bmod q$$



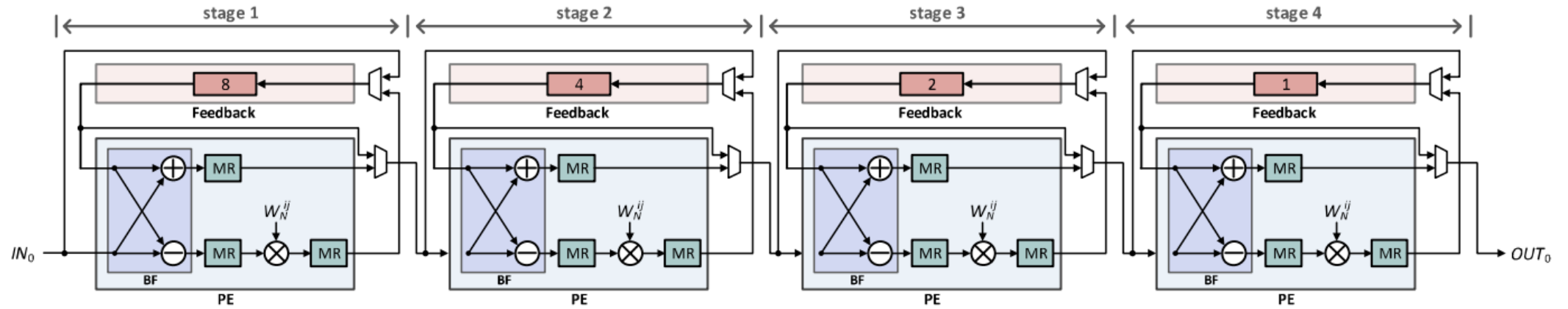
$$X_o(k) = \sum_{n=0}^{M-1} x(2n+1) W_M^{kn} \bmod q$$

$$X_e(k) = \sum_{n=0}^{M-1} x(2n) W_M^{kn} \bmod q$$

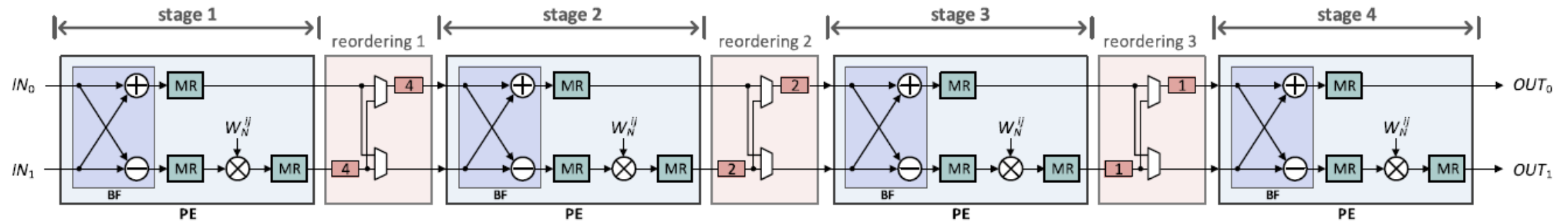


► NTT Architecture

2. SDF :



3. MDC :



► NTT Comparison on hardware complexity for $N = 256$

Architecture	Hierarchical	SDF	MDC
Butterflies	1024	8	8
Multipliers	1024	8	8
Modular Reduction	3072	24	24
Delays	-	255	127
Throughput	256	1	2
Utilization	100%	50%	100%





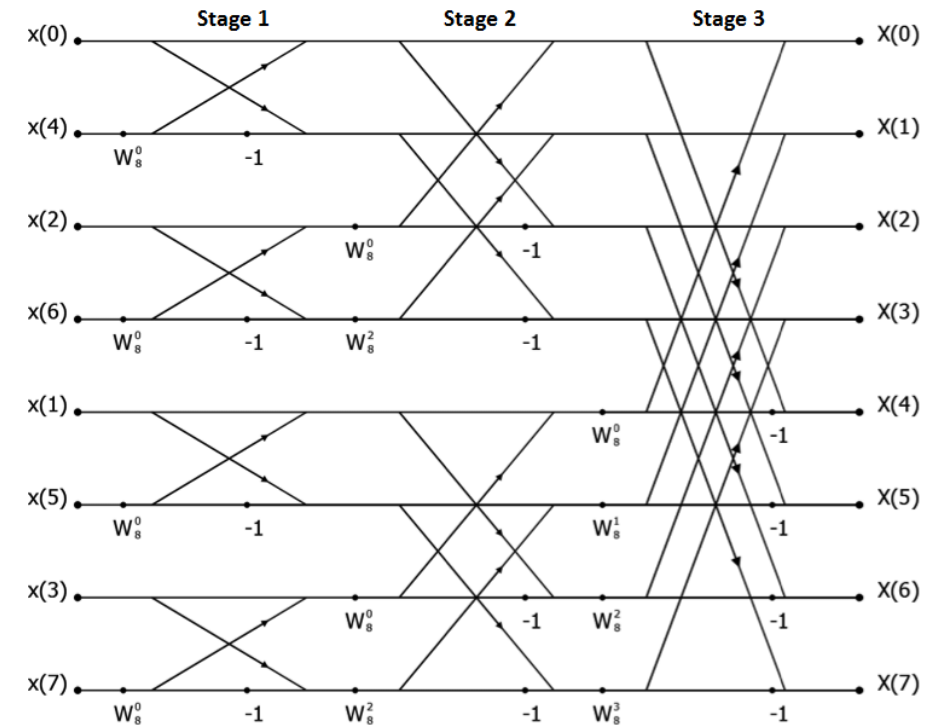
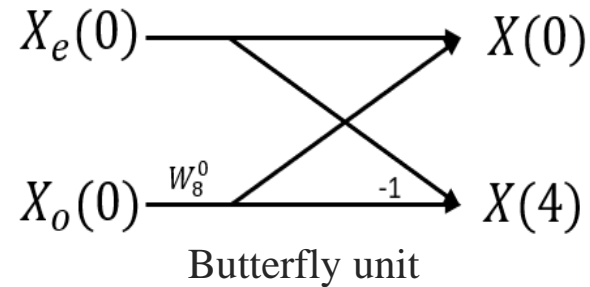
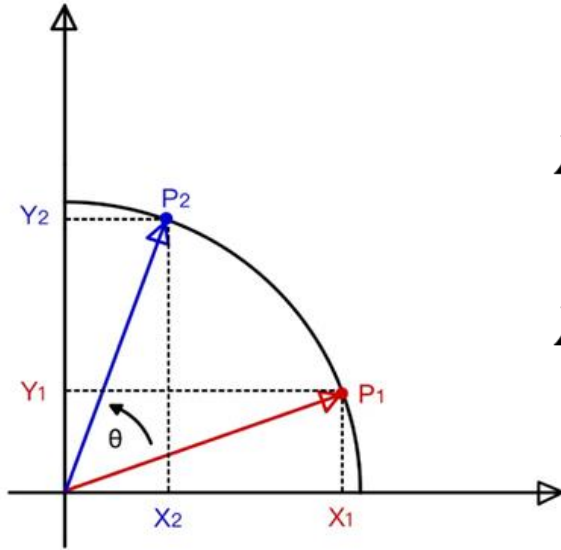
04

其他作品

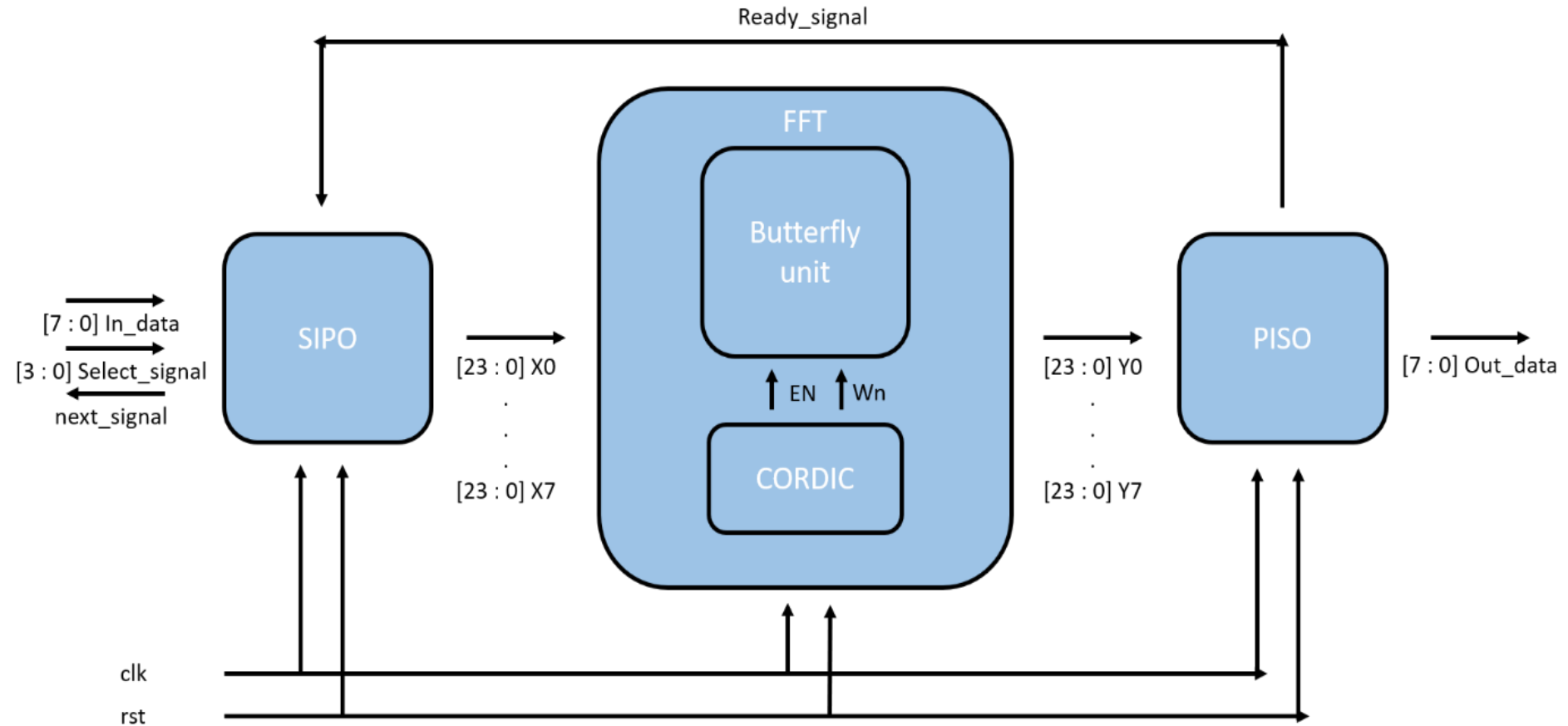


► 教育性晶片下線經驗-Implement FFT based on CORDIC

- ✓ Use Fast Fourier Transform to implement and accelerate Discrete Fourier Transform.
- ✓ Use the rotation factor of DFT into a butterfly architecture.
- ✓ Adopt CORDIC to perform trigonometric function operations and obtain rotation factors.
- ✓ Implement an 8-point FFT through the radix-2 butterfly architecture.

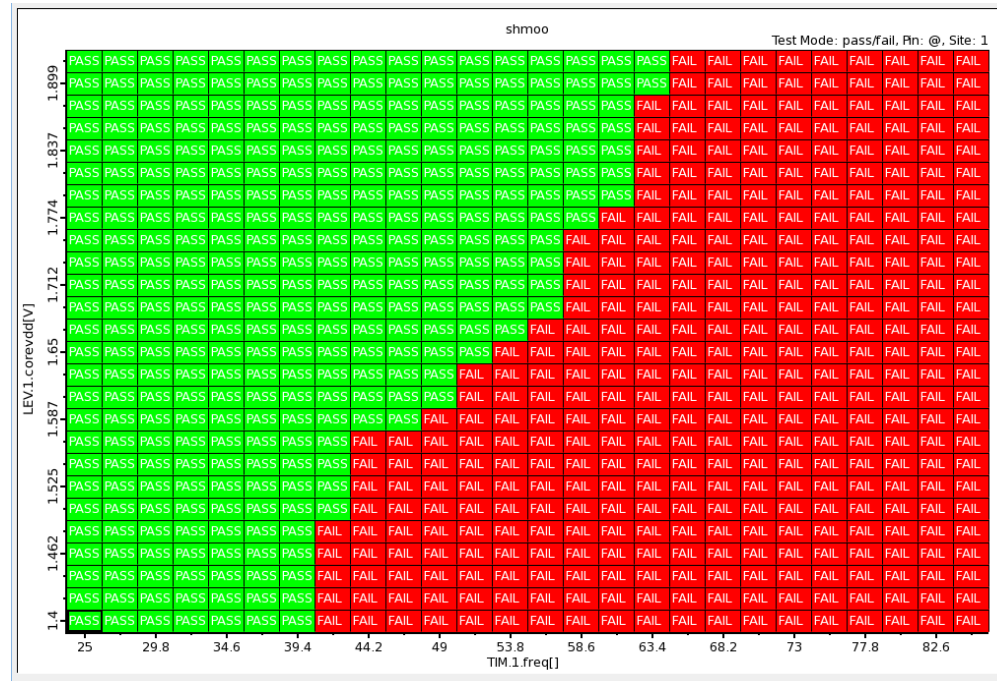
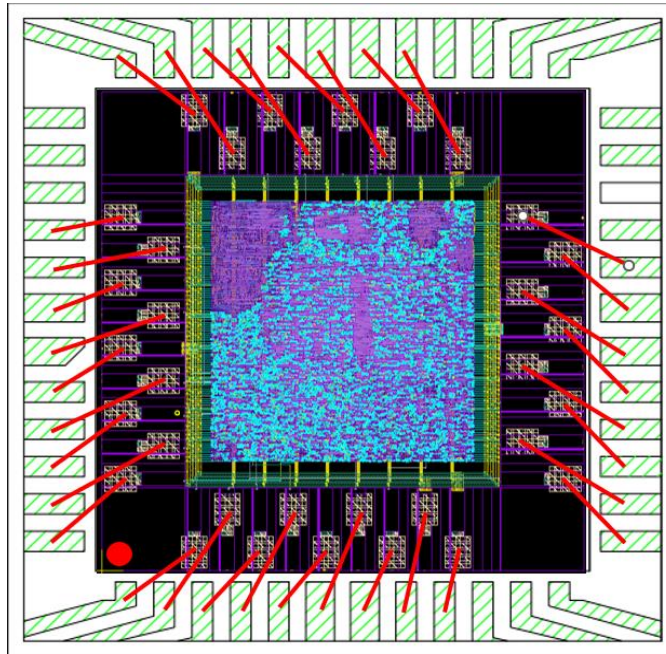


► Implement FFT based on CORDIC – Block Diagram



► Implement FFT based on CORDIC - Result

Process	TSMC 0.18um
Package	SB48
Testing results	Pass
Frequency	60 <i>MHz</i>
Power	11.8062 <i>mW</i>
Area	1.075 x 1.074 <i>mm</i> ²

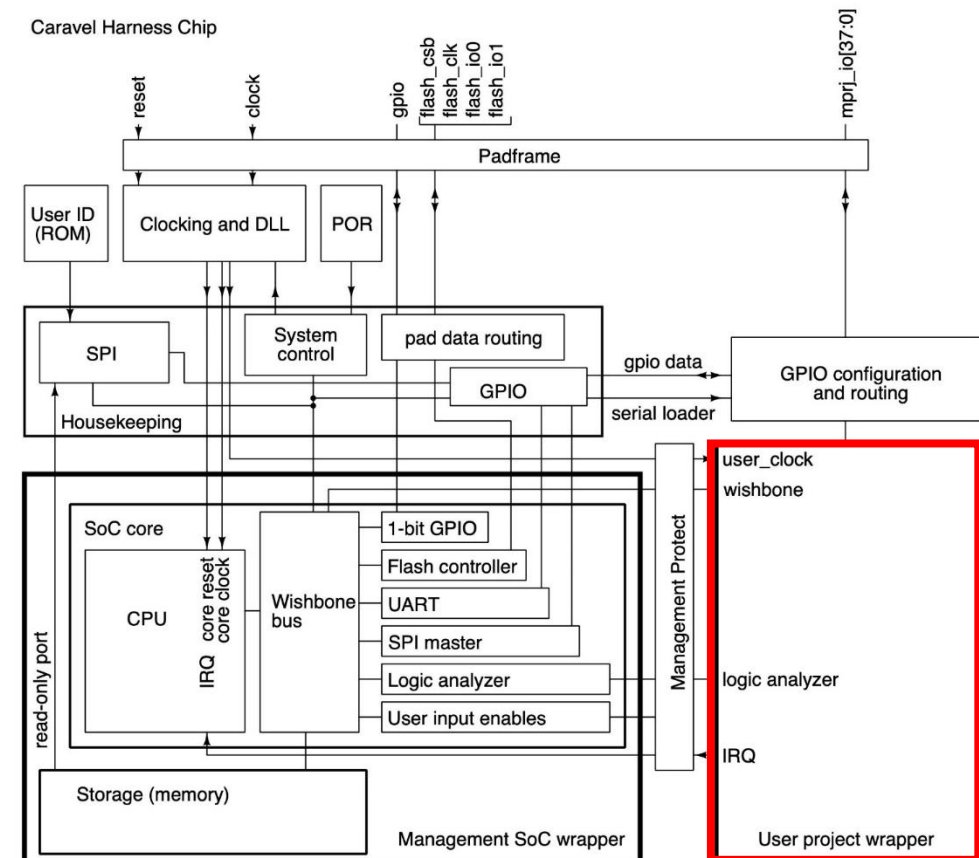


► SoC 期末專題

- ✓ Implementing Caravel SoC using Xilinx PYNQ-Z2 FPGA
- ✓ An accelerator for finite impulse response(64), matrix multiplication (4×4), and quicksort (11) has been designed and integrated into the SoC.
- ✓ The accelerator uses the AXI-4 interface for data transfer
- ✓ The baud rate for the UART has been changed from 9600 to 115200.
- ✓ The type of memory is BRAM with prefetch controller.

Main Contributions:

- ✓ Design of FIR, MM, and QS accelerators
- ✓ Testbench design and verification

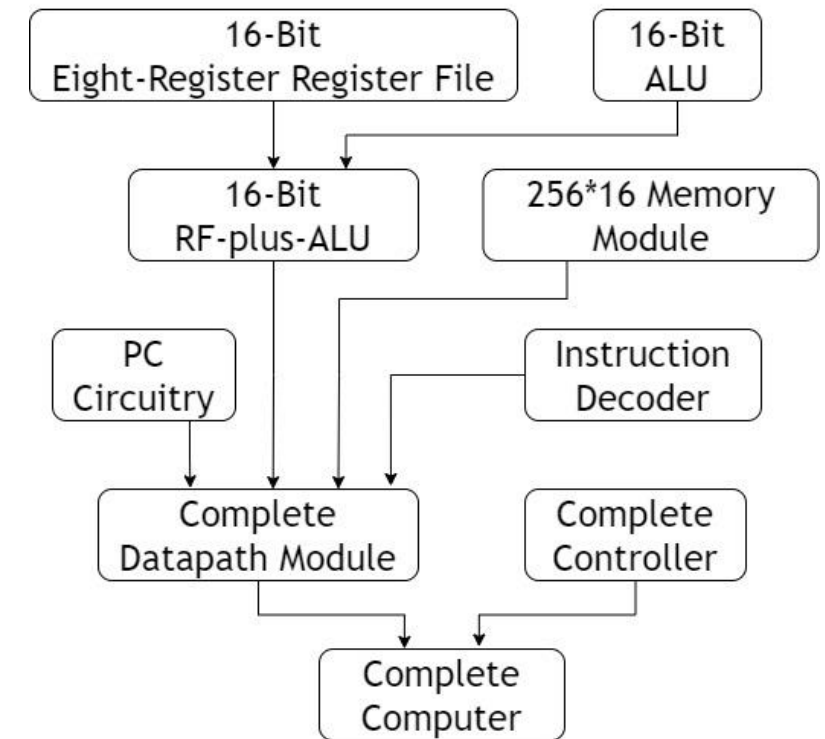


► 16-bit RISC Implementation

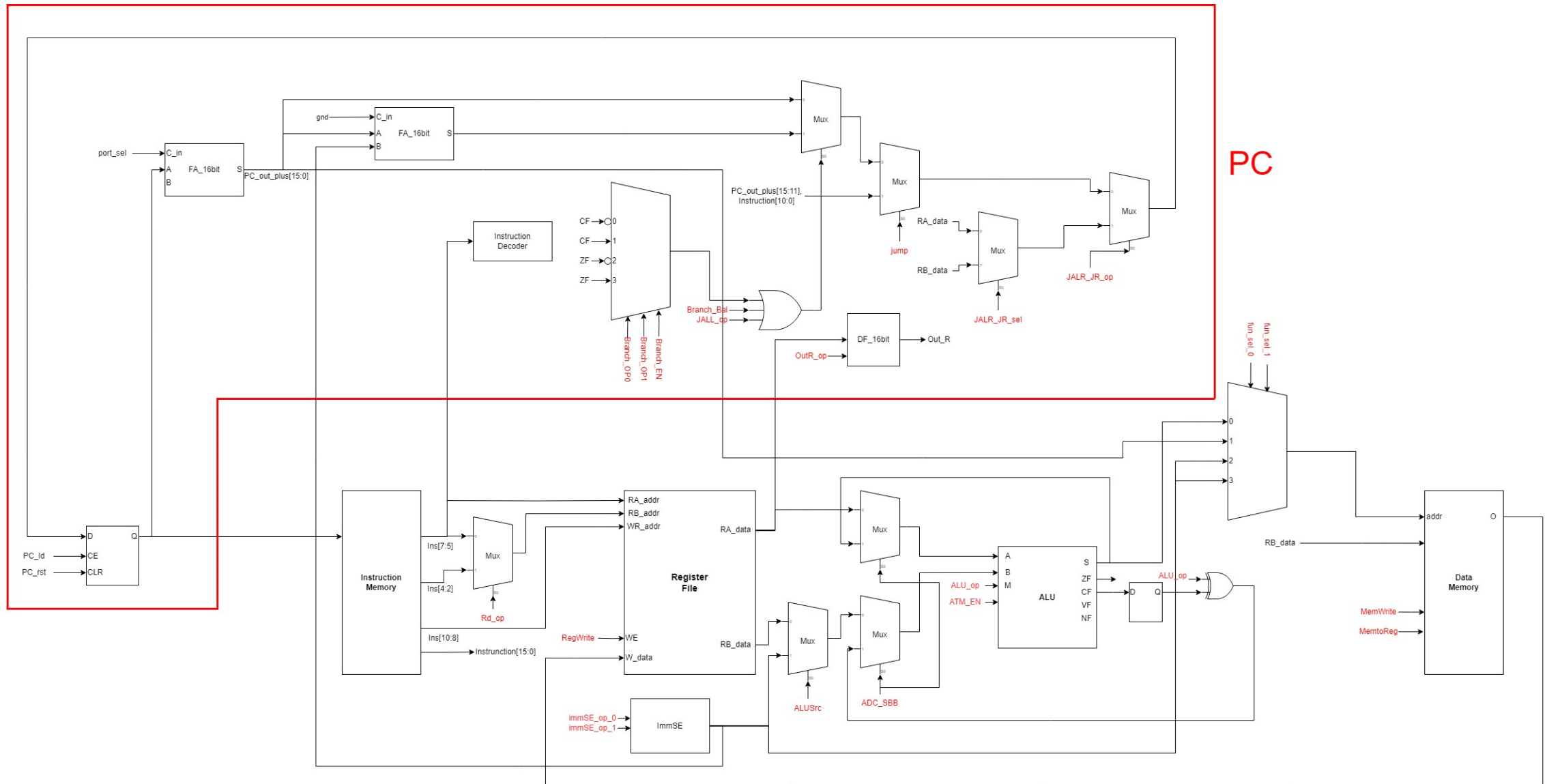
Table 1: The basic instruction set of the 16-bit RISC processor.

Mnemonic	Operation	N	Z	V	C	Instruction format
LHI Rd, #imm8	$Rd \leftarrow \{\text{imm8}, Rd[7:0]\}; (\text{imm8} = 0 \text{ to } 255)$	-	-	-	-	00001_ddd_iiiiiii
LLI Rd, #imm8	$Rd \leftarrow \{8'h0, \text{imm8}\}; (\text{imm8} = 0 \text{ to } 255)$	-	-	-	-	00010_ddd_iiiiiii
LDR Rd, [Rm, #imm5]	$Rd \leftarrow \text{Mem}[Rm + \text{imm5}];$	-	-	-	-	00011_dddmmm_iiii
LDR Rd, [Rm, Rn]	$Rd \leftarrow \text{Mem}[Rm + Rn];$	-	-	-	-	00100_dddmmmnnn_00
STR Rd, [Rm, #imm5]	$\text{Mem}[Rm + \text{imm5}] \leftarrow Rd;$	-	-	-	-	00101_dddmmm_iiii
STR Rd, [Rm, Rn]	$\text{Mem}[Rm + Rn] \leftarrow Rd;$	-	-	-	-	00110_dddmmmnnn_00
ADD Rd, Rm, Rn	$Rd \leftarrow Rm + Rn;$	*	*	*	*	00000_dddmmmnnn_00
ADC Rd, Rm, Rn	$Rd \leftarrow Rm + Rn + C;$	*	*	*	*	00000_dddmmmnnn_01
SUB Rd, Rm, Rn	$Rd \leftarrow Rm - Rn;$	*	*	*	*	00000_dddmmmnnn_10
SBB Rd, Rm, Rn	$Rd \leftarrow Rm - Rn - \overline{C};$	*	*	*	*	00000_dddmmmnnn_11
CMP Rm, Rn	$Rm - Rn;$	*	*	*	*	00110_xxxmmmnnn_01
ADDI Rd, Rm, #imm5	$Rd \leftarrow Rm + \#imm5; (\text{imm5} = 0 \text{ to } 31)$	*	*	*	*	00111_dddmmm_iiii
SUBI Rd, Rm, #imm5	$Rd \leftarrow Rm - \#imm5; (\text{imm5} = 0 \text{ to } 31)$	*	*	*	*	01000_dddmmm_iiii
MOV Rd, Rm	$Rd \leftarrow Rm;$	-	-	-	-	01011_dddmmm_xxx_xx
BCC label	$\overline{C}: PC \leftarrow PC + \text{SignExtend}(\text{label});$	-	-	-	-	1100_0011_disp8
BCS label	$C: PC \leftarrow PC + \text{SignExtend}(\text{label});$	-	-	-	-	1100_0010_disp8
BNE label	$\overline{Z}: PC \leftarrow PC + \text{SignExtend}(\text{label});$	-	-	-	-	1100_0001_disp8
BEQ label	$Z: PC \leftarrow PC + \text{SignExtend}(\text{label});$	-	-	-	-	1100_0000_disp8
B[AL] label	$PC \leftarrow PC + \text{SignExtend}(\text{label});$	-	-	-	-	1100_1110_disp8
JMP label	$PC[10:0] \leftarrow \text{label};$	-	-	-	-	10000_label11
JAL Rd,label	$Rd \leftarrow PC; PC \leftarrow PC + \text{SignExtend}(\text{label});$	-	-	-	-	10001_ddd_disp8
JAL Rd,Rm	$Rd \leftarrow PC; PC \leftarrow Rm;$	-	-	-	-	10010_dddmmm_xxx_yy
JR Rd	$PC \leftarrow Rd;$	-	-	-	-	10011_ddd_xxxxxxxx
OutR Rm	$\text{OutR} \leftarrow Rm;$	-	-	-	-	11100_xxx_mmm_xxx_00
HLT	Set done flag to 1 and halt CPU;	-	-	-	-	11100_xxxx_xxxxx_01

Note that the memory address in each of LDR and STR instructions corresponds to a word.



► 16-bit RISC Implementation



A large, thin black circle is centered on the page. Two solid black dots are positioned on the circle's circumference, one on the left and one on the right. Short, thin black line segments extend from each dot, pointing towards the center of the circle.

感謝聆聽

Thank you for your attention.