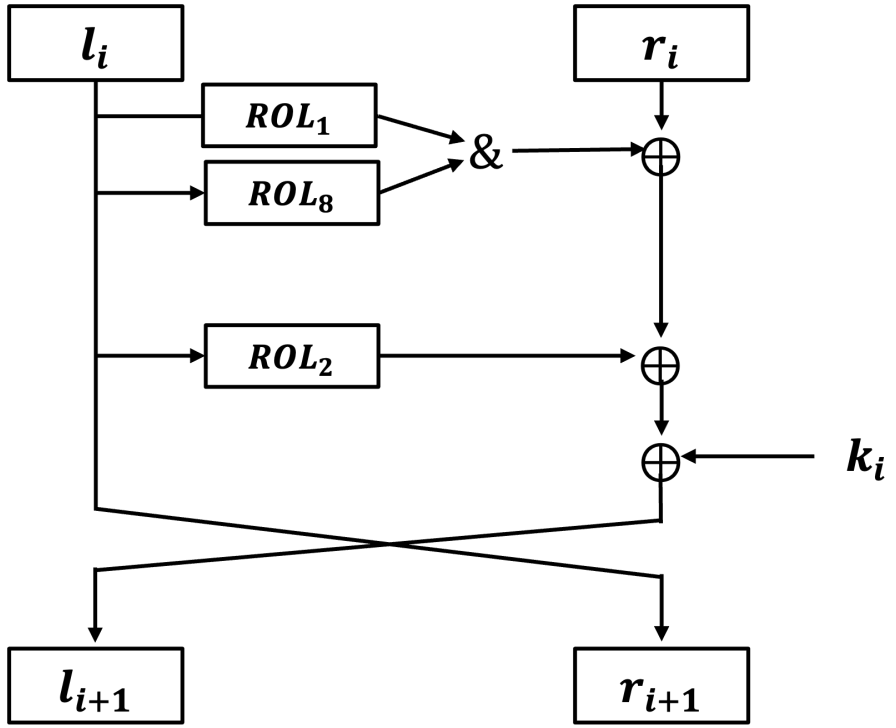
RISC-V 상에서 SIMON, SPECK-CTR 병렬 구현

https://youtu.be/afDsscioIlk

SIMON

- 2013년 미국 국가안보국에서 개발한 Feistel 구조의 경량블록 암호
- ARX(AND, Rotation, XOR) 구조

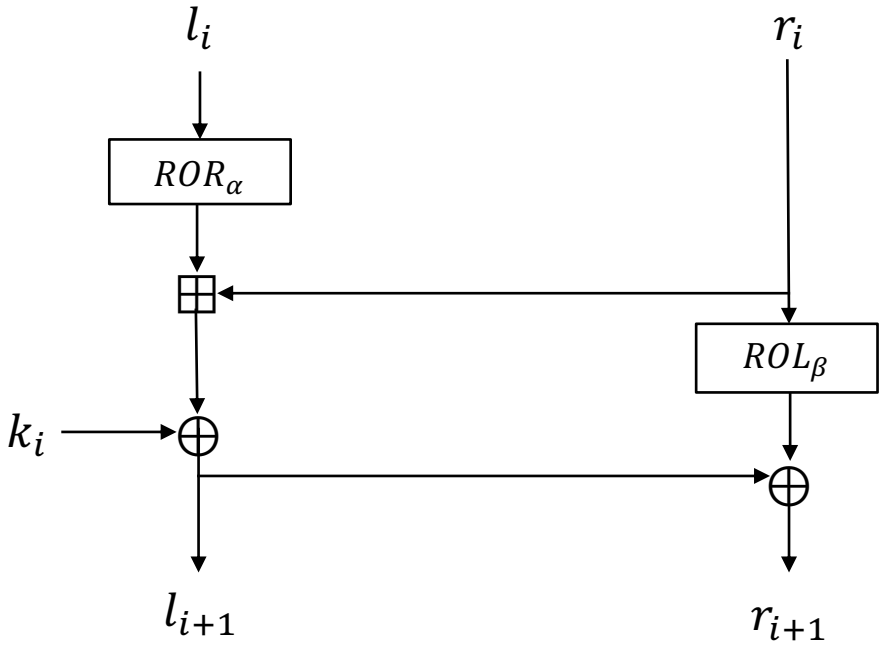
Block Size	Key Size	Rounds	Word Size
32	64	32	16
48	72	36	24
	96		
64	96	42	32
	128	44	
96	96	52	48
	144	54	
128	128	68	64
	192	69	
	256	72	



SPECK

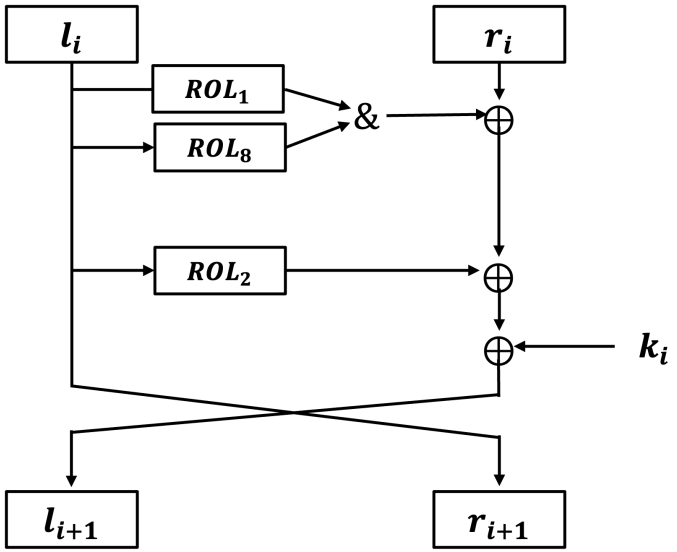
- 2013년 미국 국가안보국에서 개발한 Feistel 구조의 경량블록 암호
- ARX(Addition, Rotation, XOR) 구조

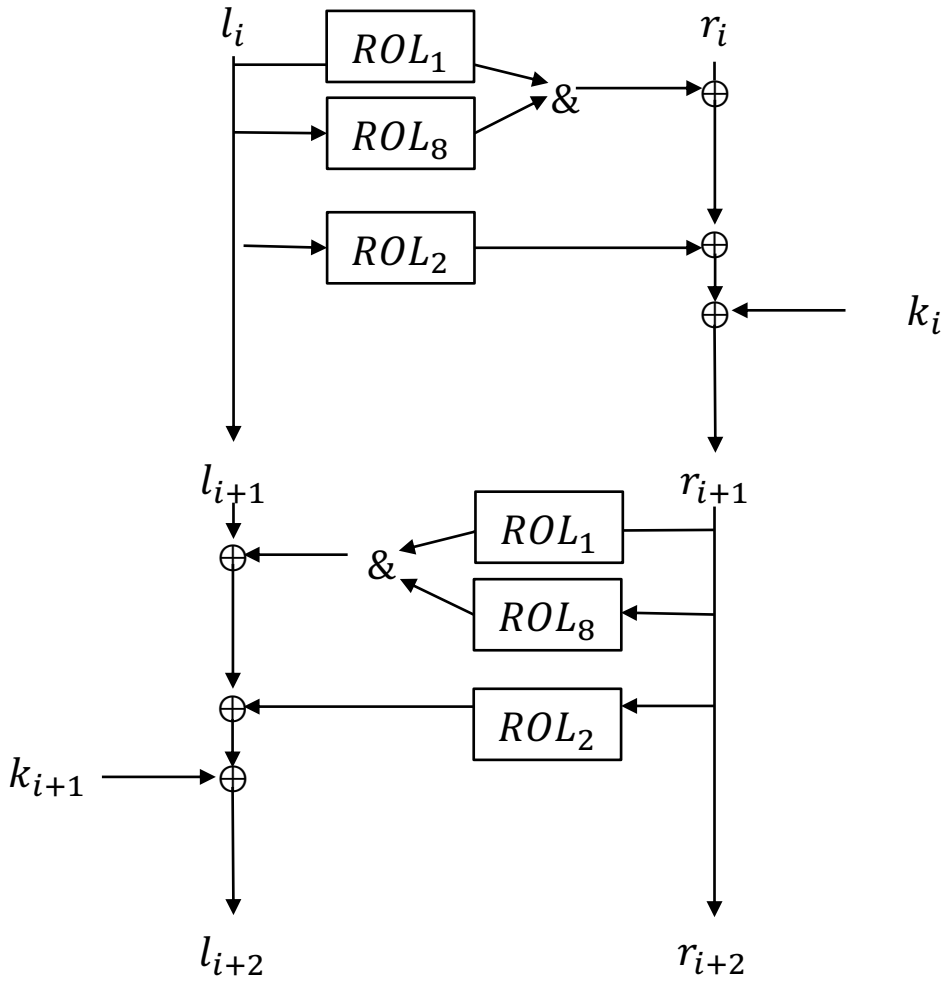
Block Size	Key Size	Rounds	Word Size	Rotation α	Rotation β
32	64	22	16	7	2
48	72	22	24	8	3
	96	23			
64	96	26	32		
	128	27			
96	96	28	48		
	144	29			
128	128	32	64		
	192	33			
	256	34			



SIMON 구현 기법

- 단일 평문 구현
  - 블록 이동 생략






```
.macro round
    lw      a4, 0(a1) //rk

    slli    t1, a3, 1  //x2 <<1
    srli    t2, a3, 31
    or      t0, t1, t2  //t0 = tmp

    slli    t1, a3, 8  //x2 <<8
    srli    t2, a3, 24
    or      t4, t1, t2

    and     t0, t0, t4

    slli    t1, a3, 2  //x2 <<2
    srli    t2, a3, 30
    or      t4, t1, t2

    xor     t0, t0, t4
    xor     t0, t0, a2
    xor     a2, t0, a4

    addi    a1, a1, 4
//*********************************//
    lw      a4, 0(a1)

    slli    t1, a2, 1  //x2 <<1
    srli    t2, a2, 31
    or      t0, t1, t2

    slli    t1, a2, 8  //x2 <<8
    srli    t2, a2, 24
    or      t4, t1, t2

    and     t0, t0, t4

    slli    t1, a2, 2  //x2 <<2
    srli    t2, a2, 30
    or     t4, t1, t2

    xor     t0, t0, t4
    xor     t0, t0, a3
    xor     a3, t0, a4

    addi    a1, a1, 4
.endm
```

SIMON 병렬 구현 기법

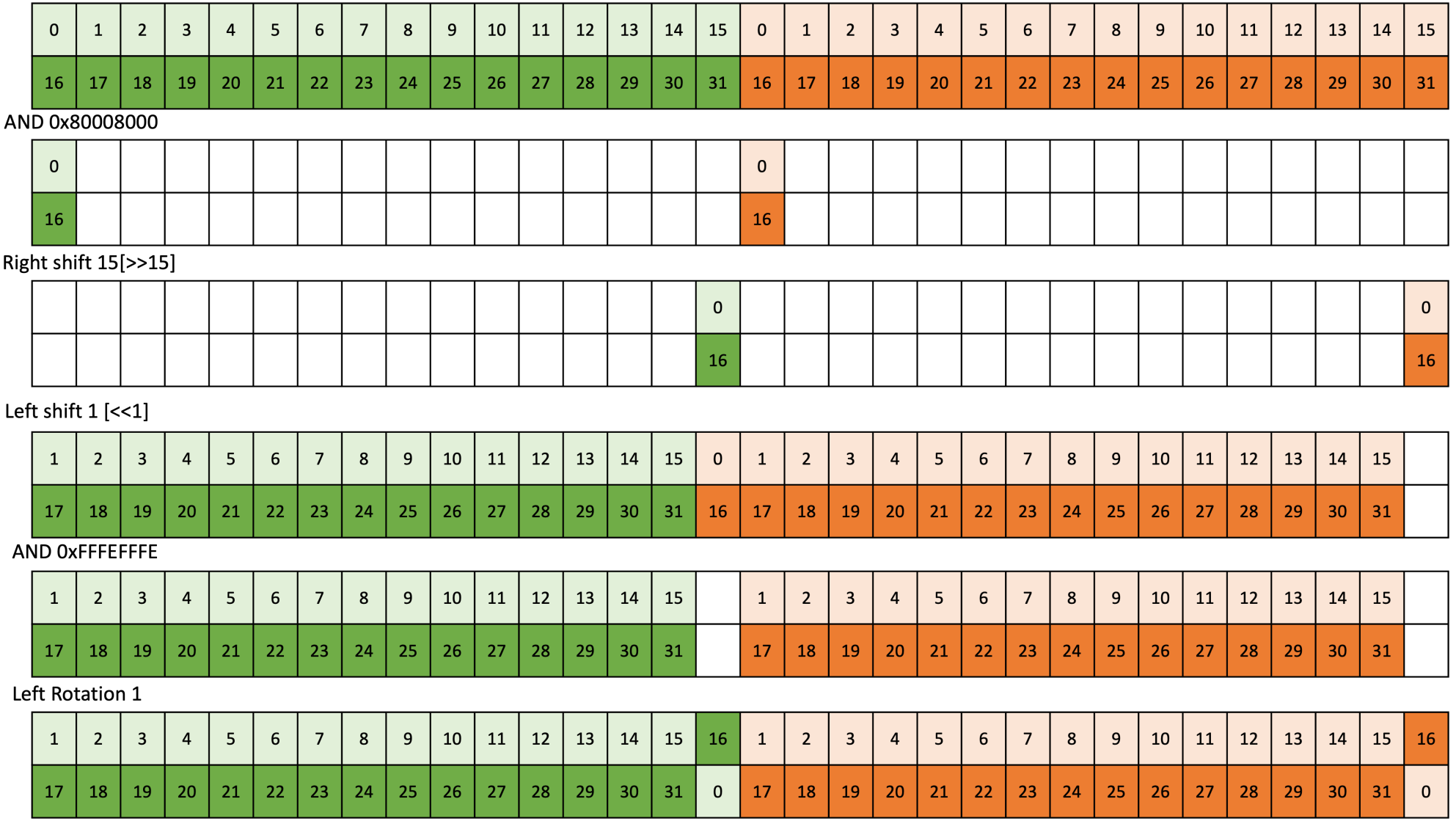
- 레지스터 내부 정렬

a2	PT1[0]
a3	PT1[1]
a4	PT2[0]
a5	PT2[1]

a2	Upper bit of PT2[0]	Upper bit of PT1[0]
a3	Lower bit of PT2[0]	Lower bit of PT1[0]
a4	Upper bit of PT2[1]	Upper bit of PT1[1]
a5	Lower bit of PT2[1]	Lower bit of PT1[1]

SIMON 병렬 구현 기법

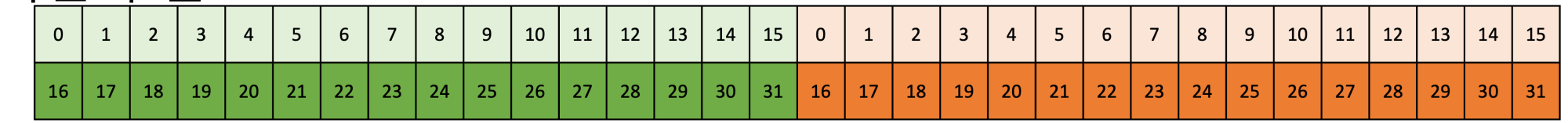
- 로테이션 구현



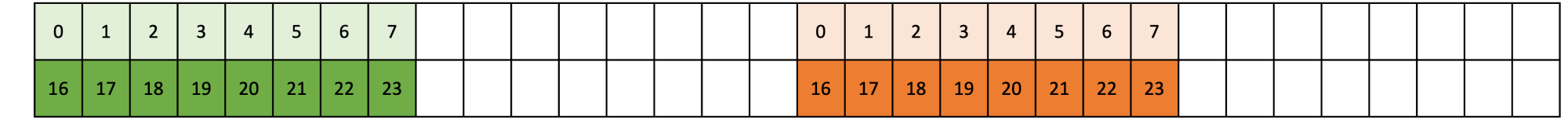
a2	Upper bit of PT2[0]	Upper bit of PT1[0]
a3	Lower bit of PT2[0]	Lower bit of PT1[0]
a4	Upper bit of PT2[1]	Upper bit of PT1[1]
a5	Lower bit of PT2[1]	Lower bit of PT1[1]

SIMON 병렬 구현 기법

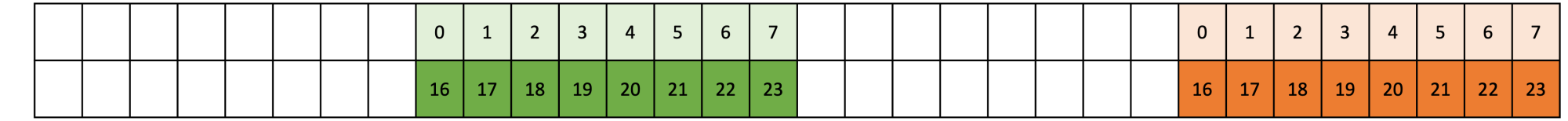
- 로테이션 구현



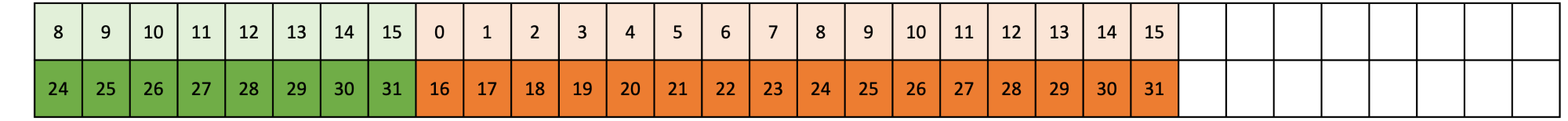
AND 0xFF00FF00



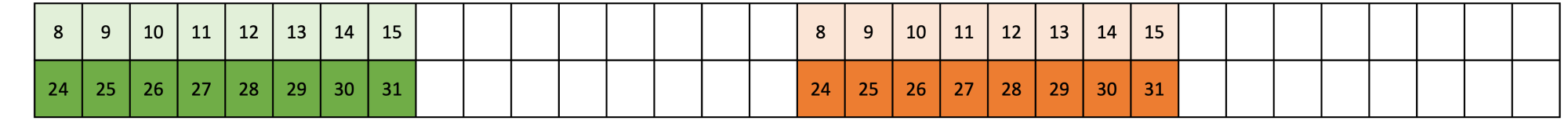
Right shift 8[>>8]



Left shift 8 [<<8]



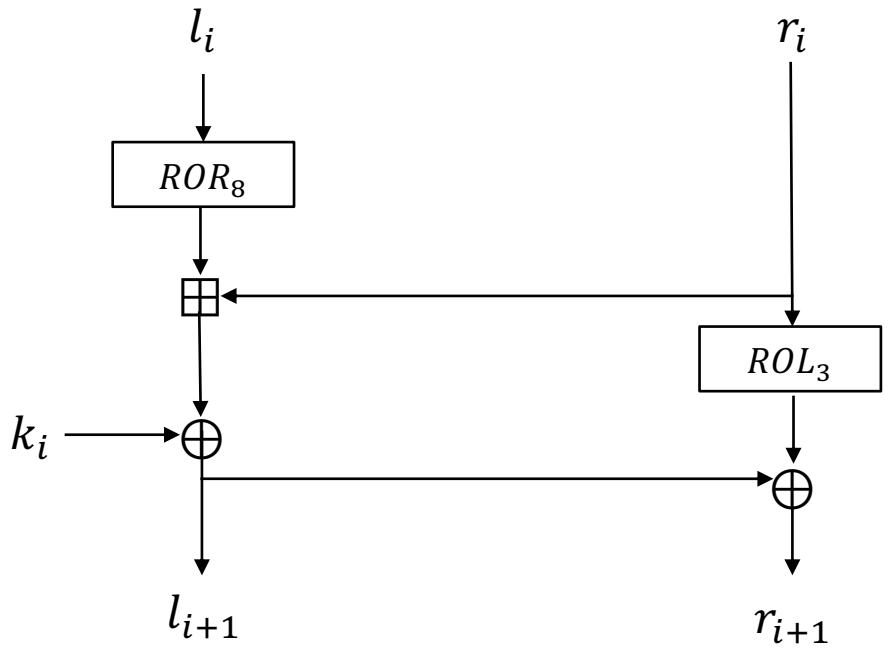
AND 0xFF00FF00



Left Rotation 8

SPECK 구현 기법

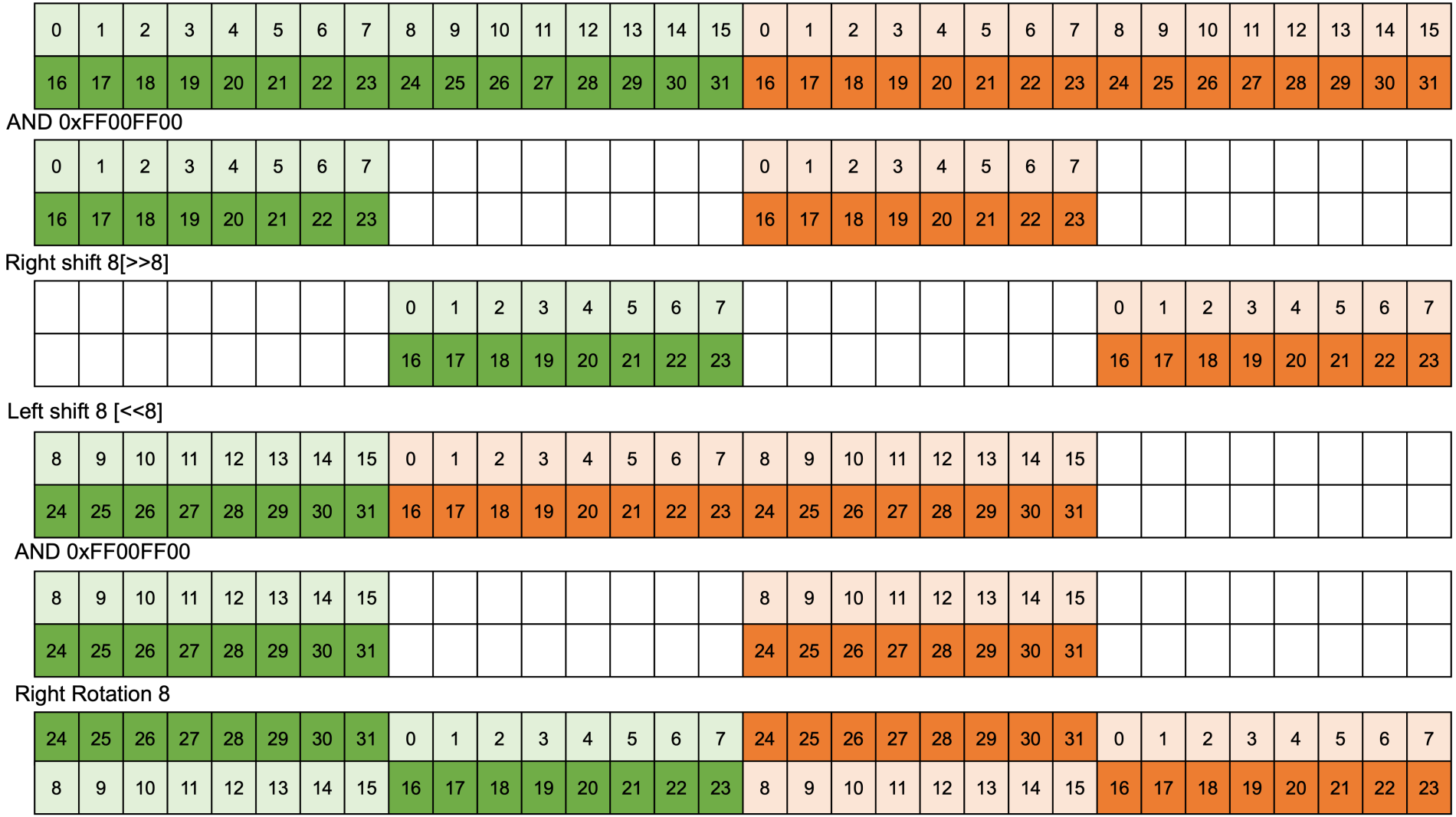
- 단일 평문 구현




```
.macro round
    lw      a4, 0(a1) //rk

    srli    t1, a3, 8
    slli    t2, a3, 24
    or      a3, t1, t2

    add     a3, a3, a2


    xor     a3, a3, a4


    slli    t1, a2, 3
    srli    t2, a2, 29
    or      a2, t1, t2

    xor     a2, a2, a3

    addi    a1, a1, 4

.endm
```

SPECK 병렬 구현 기법

- 로테이션 구현



SPECK 병렬 구현 기법

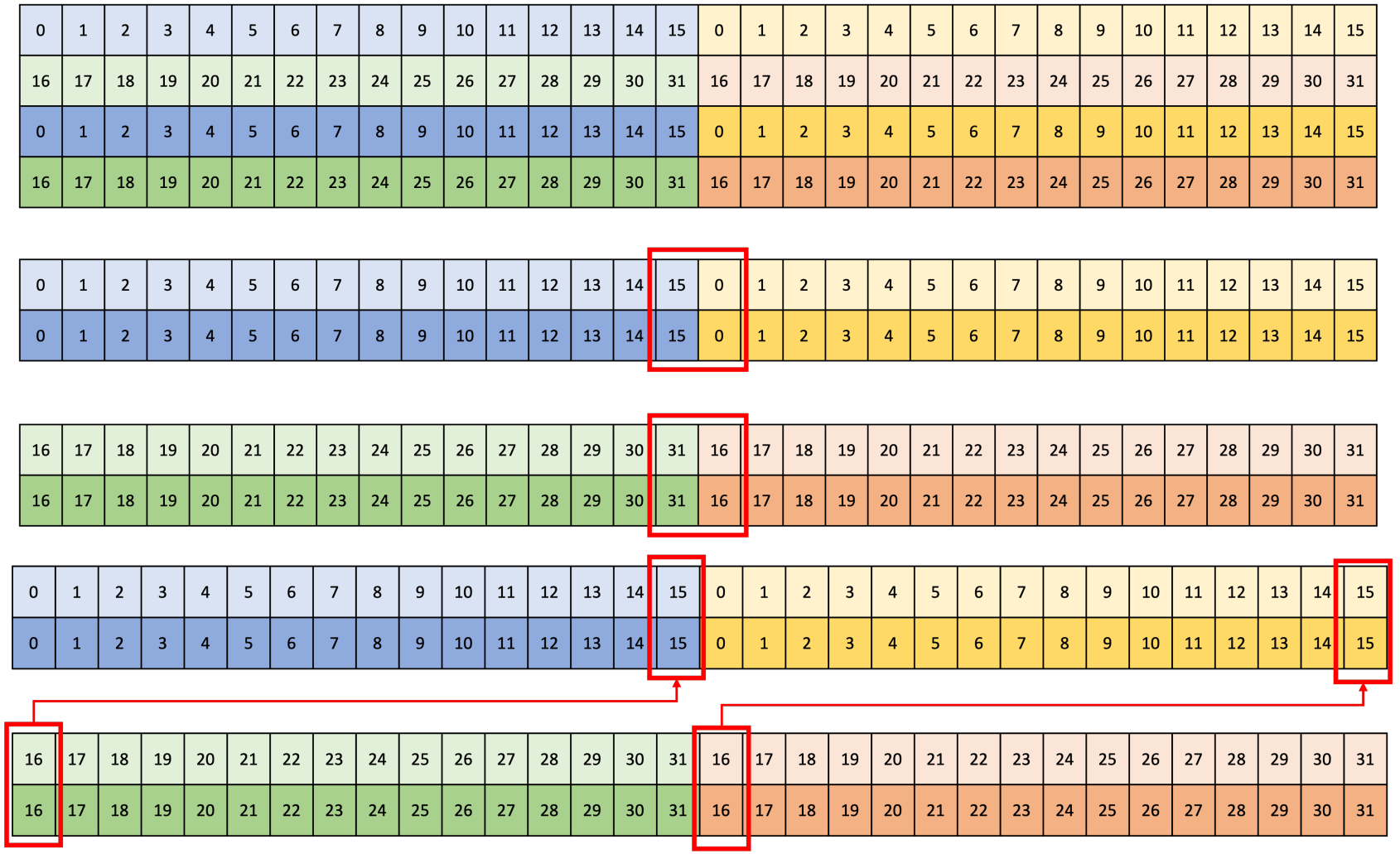
- 레지스터 내부 정렬

a2	PT1[0]
a3	PT1[1]
a4	PT2[0]
a5	PT2[1]

a2	Upper bit of PT2[0]	Upper bit of PT1[0]
a3	Lower bit of PT2[0]	Lower bit of PT1[0]
a4	Upper bit of PT2[1]	Upper bit of PT1[1]
a5	Lower bit of PT2[1]	Lower bit of PT1[1]

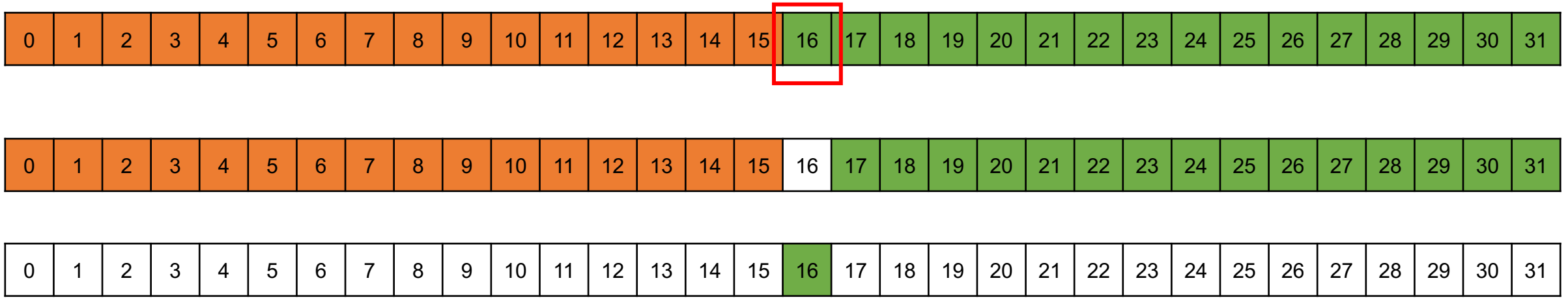
SPECK 병렬 구현 기법

- Addition 구현



SPECK 병렬 구현 기법

- Addition 구현



ex)

FFFF 1111 1111 1111 1111

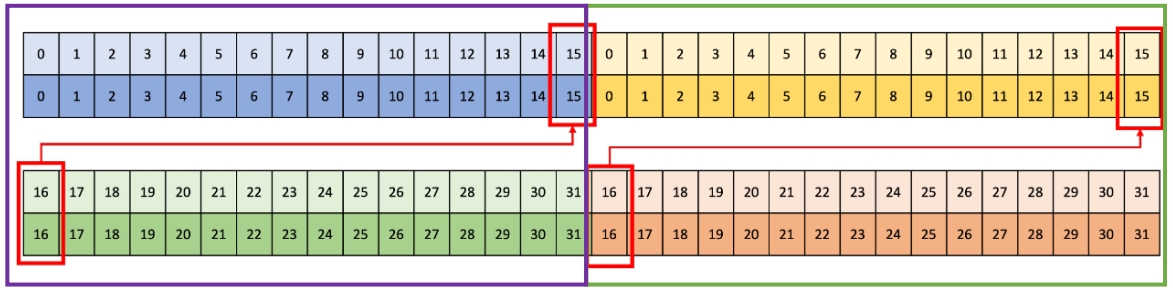
7FFF 0111 1111 1111 1111

```
  FFFF          7FFF      8000
+ FFFF        + 7FFF    + 8000
------        ------    ------
1 FFFE          FFFE    1 0000
```

이진 연산에서 덧셈 연산 → xor 연산

SPECK 병렬 구현 기법

- Addition 구현



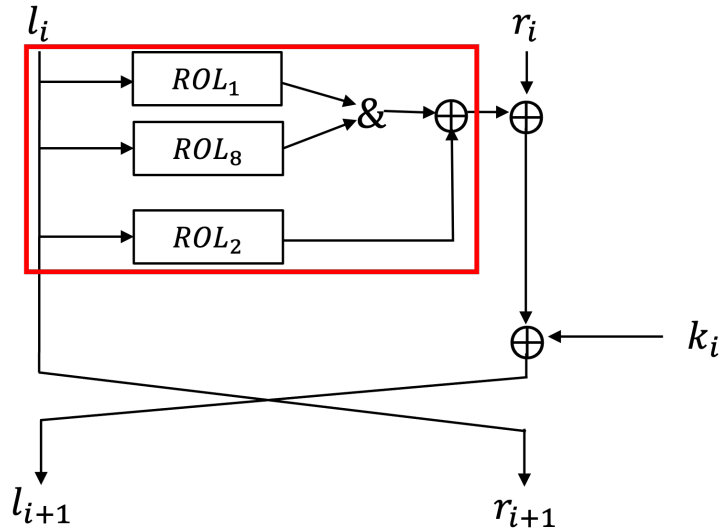

```
//하위비트
    mv      t2, a3

    mv      s5, a3
    and     a3, a3, t5

    mv      s6, a5
    and     a5, a5, t5

    add     s7, a3, a5
    and     s7, s7, t3
    srli    s7, s7, 16

    mv      a3, s5
    and     a3, a3, s3

    mv      a5, s6
    and     a5, a5, s3

    xor     a3, a3, a5

    and     s5, s5, s4
    and     s6, s6, s4

    add     a5, s5, s6

    sltu    ra, a5, s5
    slli    ra, ra, 16

    xor     a5, a5, a3

    mv      a3, t2
```

```
//상위비트
    mv      t2, a2

    mv      s5, a2
    and     a2, a2, s3

    mv      s6, a4
    and     a4, a4, s3

    xor     a2, a2, a4


    and     s5, s5, s4
    and     s6, s6, s4

    add     a4, s5, s6

    add     a4, a4, s7
    add     a4, a4, t3

    xor     a4, a4, a2

    mv      a2, t2
```

SIMON-CTR 구현 기법

- 단일 평문 구현의 사전 연산을 통한 생략
  - 2번의 XOR 연산 , 6번의 쉬프트 연산, 1번의 AND 연산, 1번의 OR 연산 생략

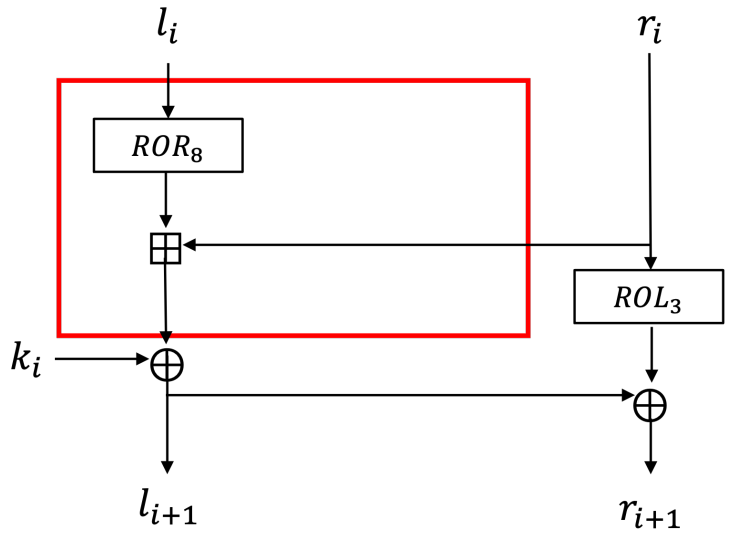
- 2개의 평문 병렬 구현의 사전 연산을 통한 생략
  - 10번의 XOR 연산, 10번의 쉬프트 연산, 12번의 AND 연산, 2번의 MV 연산 생략



SPECK-CTR 구현 기법

- 단일 평문 구현의 사전 연산을 통한 생략
  - 2번의 쉬프트 연산, 1번의 ADD 연산 생략

- 2개의 평문 병렬 구현의 사전 연산을 통한 생략
  - 6번의 XOR 연산, 6번의 쉬프트 연산, 15번의 AND 연산, 5번의 ADD 연산, 10번의 MV 연산, 1번의 STLU 연산 생략



성능 평가

SIMON			SPECK		
Ref-C	1-PT	2-PT	Ref-C	1-PT	2-PT
2,354.0004	**728.4186**	**2,055.1203**	1,873.4563	**327.436**	**1,767.6488**

SIMON-CTR			SPECK-CTR		
Ref-C	1-PT	2-PT	Ref-C	1-PT	2-PT
2,354.0004	**691.2163**	**2,021.7328**	1,873.4563	**325.4347**	**1,669.9448**

Q & A