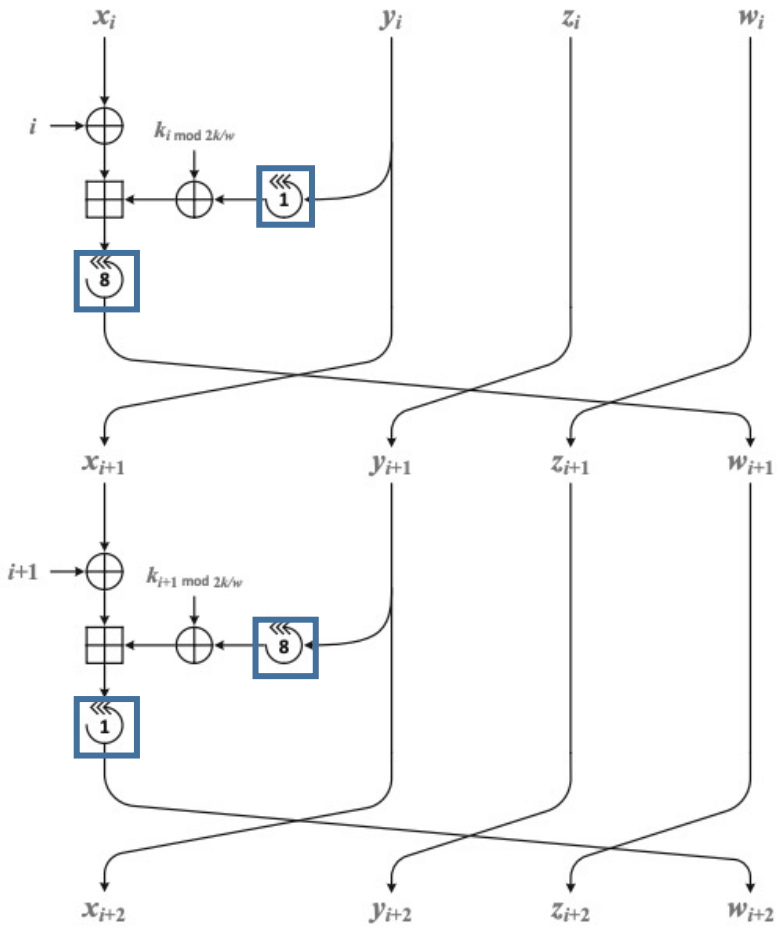
RISC-V 상에서 CHAM-64/128 병렬 구현

https://youtu.be/NtEuwSKBt2w

CHAM

- ICISC'17에서 발표된 국산 경량 블록암호
- ARX(Addition, Rotation, XOR) 연산
- Feistel 구조
  - 홀수 라운드에 ROL 연산(1, 8)
  - 짝수 라운드에 ROL 연산 (8, 1)

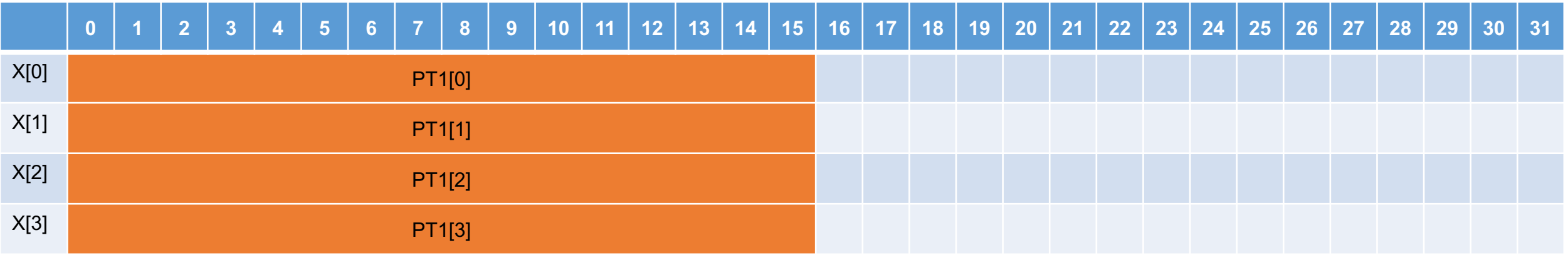


Cipher	n	k	r → r'(revised)
CHAM-64/128	64	128	80 → 88
CHAM-128/128	128	128	80 → 112
CHAM-128/256	128	256	96 → 120

평문 2개 병렬

- 평문 1개 암호화

Register	a3	a4	a5	a6
	PT1[0]	PT1[1]	PT1[2]	PT1[3]

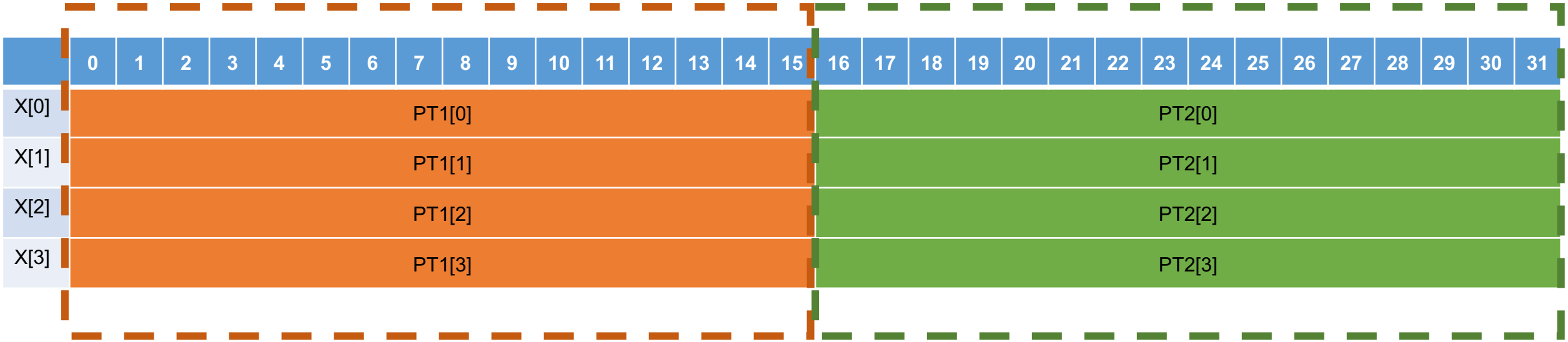


평문 2개 병렬

- 평문 2개 암호화

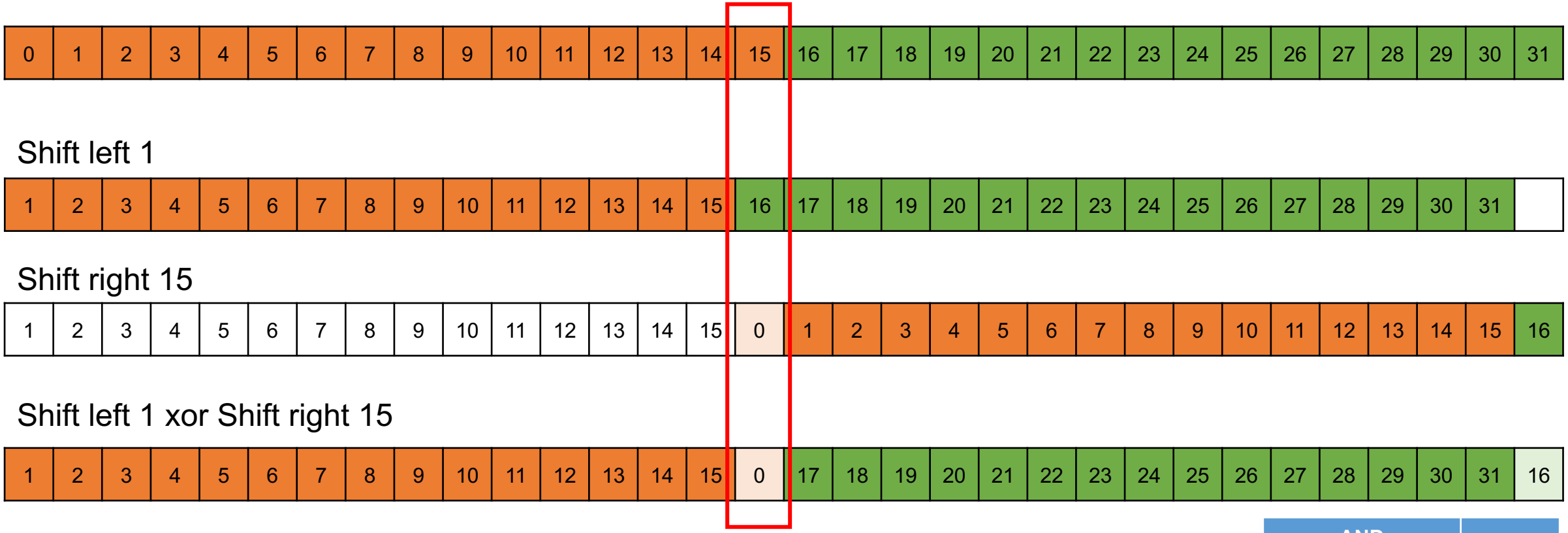
```
lh      a3, 0(a0)
lh      a4, 2(a0)
lh      a5, 4(a0)
lh      a6, 6(a0)

slli    a3, a3, 16
slli    a4, a4, 16
slli    a5, a5, 16
slli    a6, a6, 16

lh      s0, 8(a0)
lh      s1, 10(a0)
lh      s2, 12(a0)
lh      s3, 14(a0)

or      a3, a3, s0
or      a4, a4, s1
or      a5, a5, s2
or      a6, a6, s3
```

Register	a3		a4		a5		a6	
	PT1[0]	PT2[0]	PT1[1]	PT2[1]	PT2[2]	PT2[2]	PT1[3]	PT2[3]



평문 2개 병렬

- Rotation Left 1




```
slli    t0, t4, 1  //x2 <<1
and     t0, t0, t5
srli    t1, t4, 15
and     t1, t1, t6
xor     t4, t0, t1
```

```
li      t5, 0xfffefffe
li      t6, 0x00010001
```

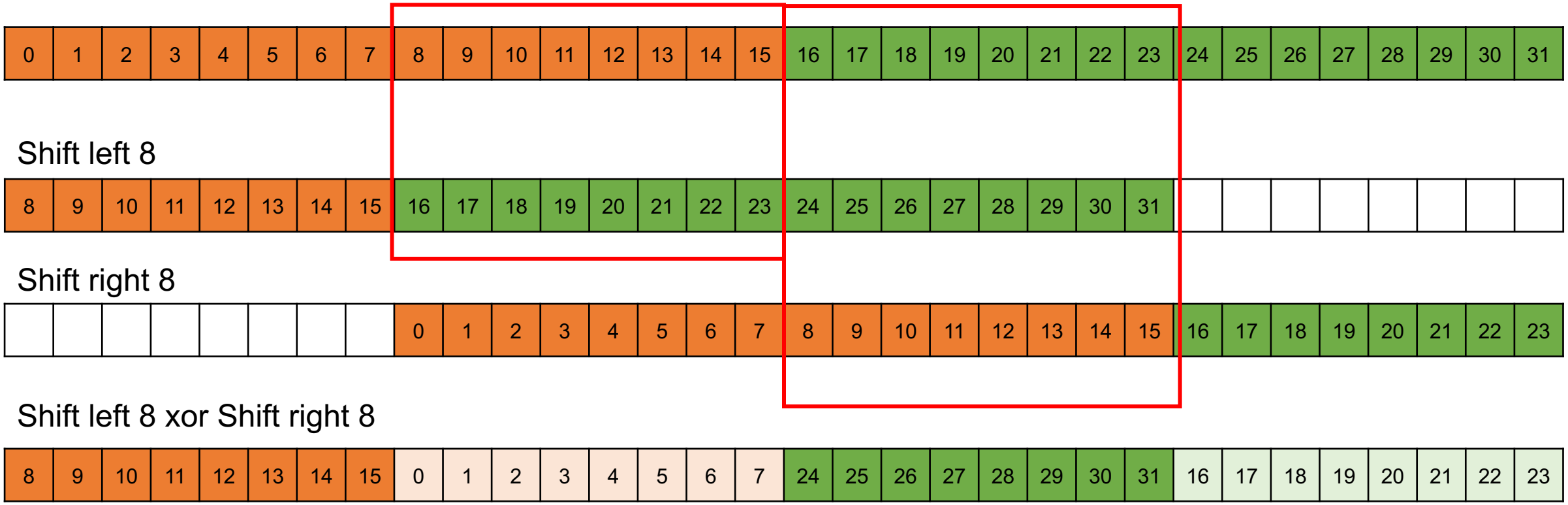
1111 1111 1111 1110

0000 0000 0000 0001

AND		
0	0	0
0	1	0
1	0	0
1	1	1

평문 2개 병렬

- Rotation Left 8




```
slli    t0, a3, 8    //x1<<8
and     t0, t0, s1
srli    t1, a3, 8
and     t1, t1, s2
xor     a3, t0, t1
```

```
li      s1, 0xff00ff00 //<<8
li      s2, 0x00ff00ff //>>8
```

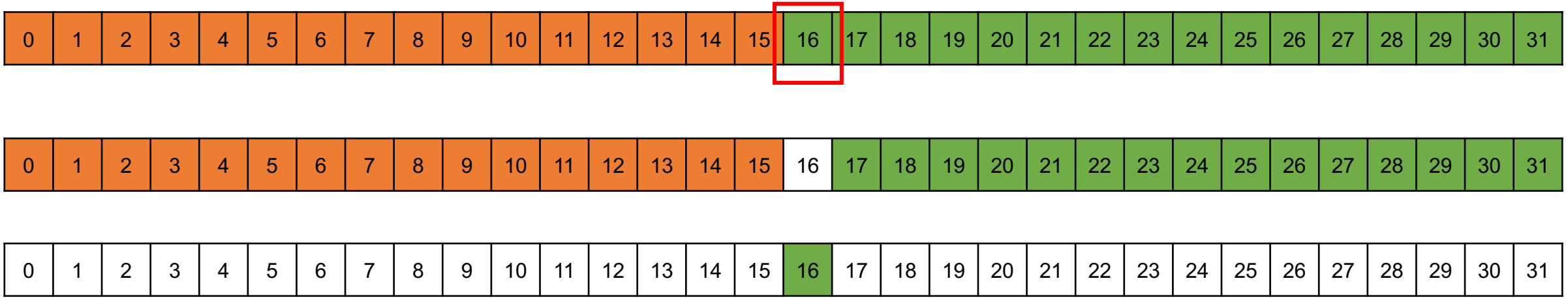
1111 1111 0000 0000

0000 0000 1111 1111

AND		
0	0	0
0	1	0
1	0	0
1	1	1

평문 2개 병렬

- 16-bit씩 병렬 덧셈 연산



ex)

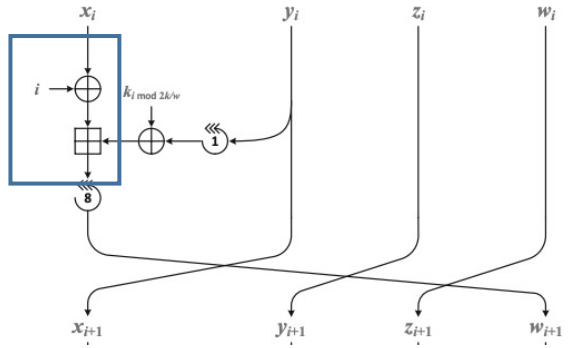
FFFF 1111 1111 1111 1111

7FFF 0111 1111 1111 1111

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

이진 연산에서 덧셈 연산 → xor 연산

```
mv      s5, t4
and     t4, t4, s3

xor     a3, a3, t3

mv      s6, a3
and     a3, a3, s3

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

add     a3, s5, s6
xor     a3, a3, t4

li      s3, 0x8000
li      s4, 0xffff7fff
```



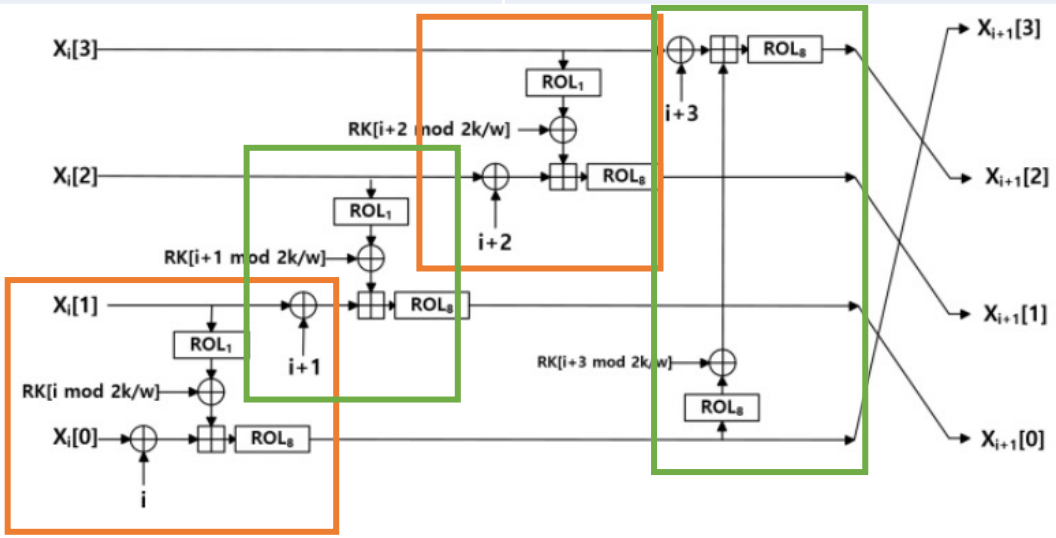
라운드 병렬

1 round , 3 round

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PT[0] → **PT[0]'**																PT[2] → **PT[2]'**															
PT[1]																PT[3]															

2 round, 4 round

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PT[1]																PT[3]															
PT[2]																**PT[0]'**															



라운드 병렬

```
.macro even_round
   //t4 = a4
      slli    s9, s9, 16
      and     a3, a3, a7
      srli    a3, a3, 16
      or      a3, a3, s9
      mv      t0, a3
      mv      t1, a4

      mv      t4, t0
      mv      a4, t0
      mv      a3, t1

      slli    t0, t4, 8  //x2 <<8
      and     t0, t0, s1
      srli    t1, t4, 8
      and     t1, t1, s2
      xor     t4, t0, t1

      lw      t2, 0(a1)   //rk[0]
      xor     t4, t4, t2//roundkey xor <<8
      addi    a1, a1, 4

      mv      s5, t4
      and     t4, t4, s3

      xor     a3, a3, a2  //rc

      mv      s6, a3
      and     a3, a3, s3

      xor     t4, t4, a3 //17번째 저장한 두개를 xor 해

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

      add     a3, s5, s6
      xor     a3, a3, t4

      slli    t0, a3, 1   //x1<<1
      and     t0, t0, t5
      srli    t1, a3, 15
      and     t1, t1, t6
      xor     a3, t0, t1

      mv      t0, a3
      mv      a3, s8
      mv      a4, t0

      add     a2, a2, s0   //t3 = rc
.endm
```

a3 (첫번째 블록)	PT[0]’	PT[2]’
a4 (두번째 블록)	PT[1]	PT[3]

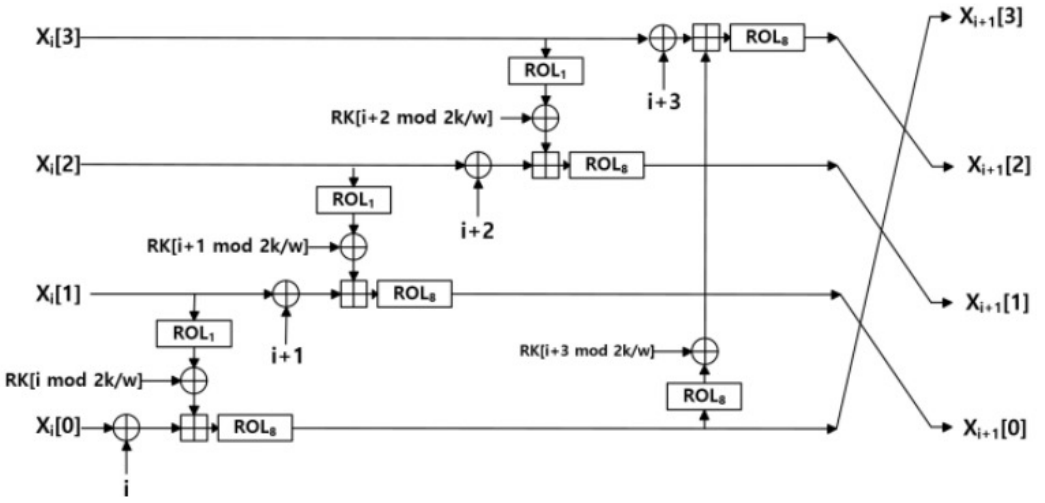
a3 (첫번째 블록)	PT[2]	PT[0]’
a4 (두번째 블록)	PT[1]	PT[3]

a3 (첫번째 블록)	PT[1]	PT[3]
a4 (두번째 블록)	PT[2]	PT[0]’

a3 (첫번째 블록)	PT[0]’	PT[2]’
a4 (두번째 블록)	PT[1]’	PT[3]’

성능평가

- 레퍼런스 C : 9215
- 레퍼런스 코드 → asm : 2252
- Asm에서 블록 이동 생략 : 1806
- 평문 2개 병렬 : 2841 / 2 = 1420
- 평문 2개 병렬 + 블록 이동 생략 : 2436 / 2 = 1218
- 라운드 병렬 (라운드 절반으로 감소 + 블록 이동 생략): 1552



Q & A