Simpira 최적화

https://youtu.be/vbr62aAu2oo

Simpira

- **AES 라운드 함수**를 기반으로 하는 순열
  - 임의의 큰 입력 크기에 대해 매우 효율적인 순열을 제공
  - 최적화된 소프트웨어 구현을 위해 **Intel AES-NI 명령어 셋** 활용
  - 입력 길이 : b x 128-bit($b \in \mathbb{N}^+$)
- Simpira의 1라운드는 **AES의 라운드 함수 2번으**로 구성
  - 고정된 라운드키를 사용 → 고정된 출력 값
  - 고정된 라운드키를 0x00 설정하는 것은 안전하지 않음
    - 따라서, **라운드 함수($\mathbb{F}_{c,b}$)를 통해 연산 된 값(라운드 상수)을 사용**
  - **AES 라운드 함수 2번으로 전체 비트 확산 가능**

b : block 개수
c : 라운드 카운터

Algorithm 2 Simpira Algorithm

procedure Simpira($state, rk$)
1: $R \leftarrow 6$
2: **for** $c = 1$ **to** R **do**
3: $state \leftarrow \mathrm{F}_{c,b}(state)$
4: **end for**
5: $state \leftarrow InvMixColumns(state)$
6: **return** $state$
end procedure

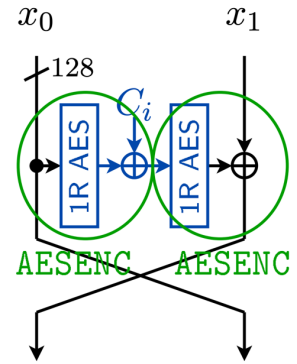
Algorithm 3 $\mathrm{F}_{c,b}$ Algorithm (b=1)

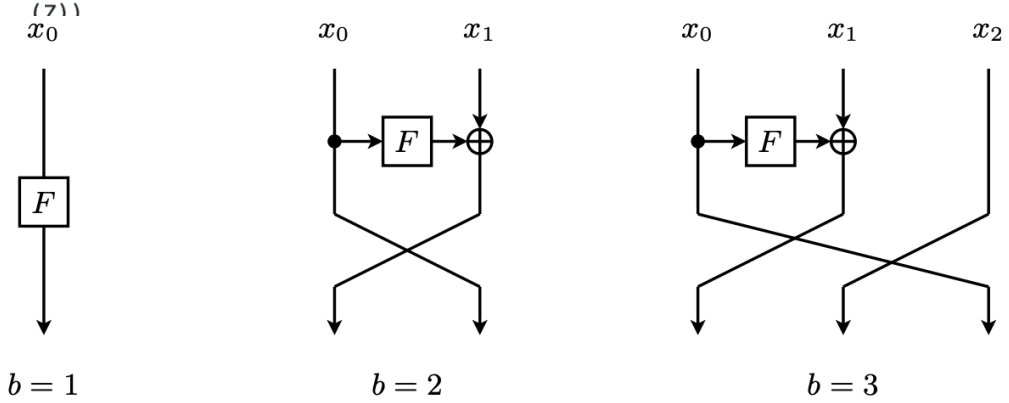
procedure $\mathrm{F}_{c,b}(state)$
1: $RK[0] = \mathtt{0x00} \oplus c \oplus b$
2: $RK[4] = \mathtt{0x10} \oplus c \oplus b$
3: $RK[8] = \mathtt{0x20} \oplus c \oplus b$
4: $RK[12] = \mathtt{0x30} \oplus c \oplus b$
5: **return** $AES(AES(state, RK), Z)$
end procedure

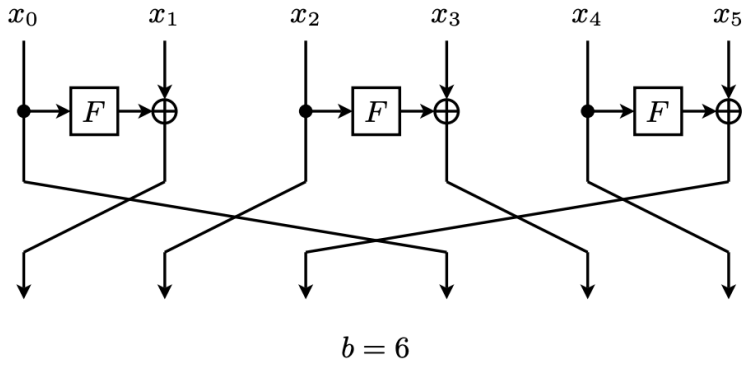
Simpira

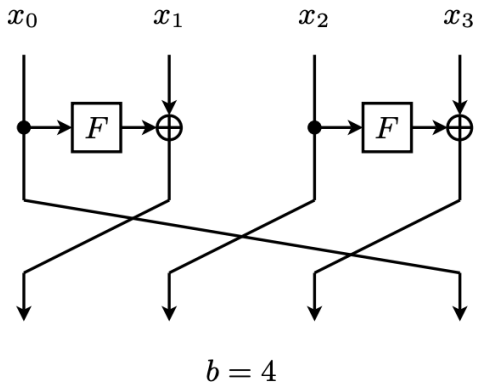
- F함수($\mathbb{F}_{c,b}$)
  - __mm_aesenc_sil128 (AES-NI)를 통해 매우 단순하게 구현

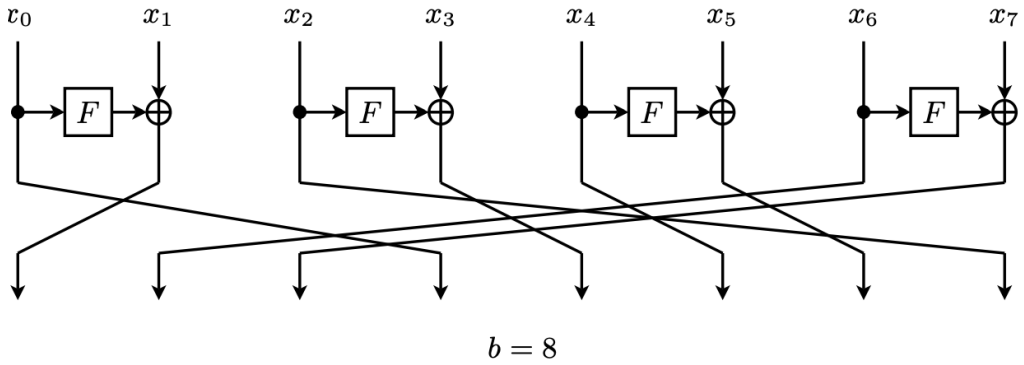
```
#define SIMPIRA_F(C, B, X, Z)                                          \
    _mm_aesenc_si128(                                                  \
        _mm_aesenc_si128((X),                                          \
                         _mm_set_epi32(0x00 ^ (C) ^ (B), 0x10 ^ (C) ^ (B),   \
                                       0x20 ^ (C) ^ (B), 0x30 ^ (C) ^ (B))), \
(Z))
```





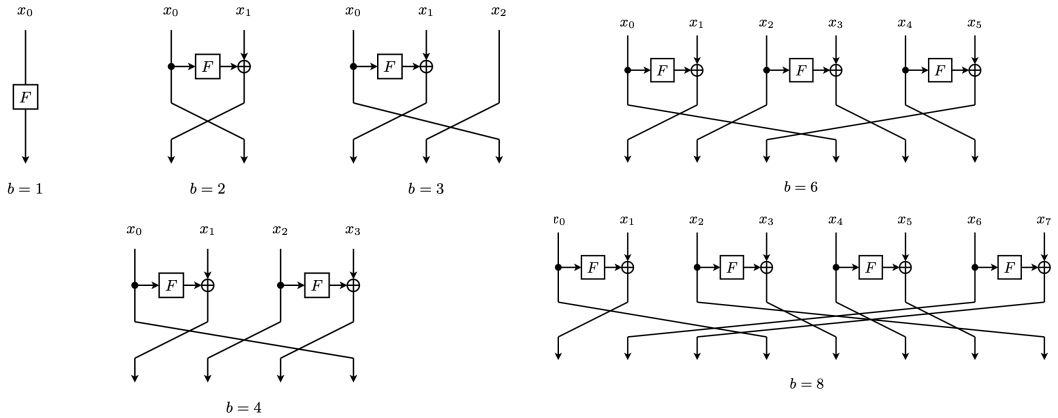






Simpira

- F함수($\mathbb{F}_{c,b}$)



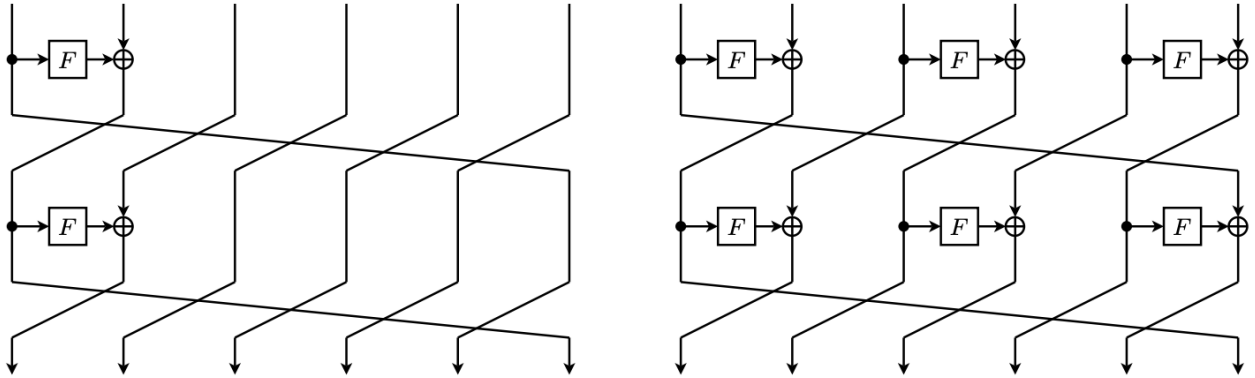


Fig. 1. Two common classes of Generalized Feistel Structures (GFSs) are the Type-1 GFS (left) and the Type-2 GFS (right). For each example, two rounds are shown of a GFS that operates on $b = 6$ subblocks. We will initially consider these GFSs in this

b	structure	round	AESENC
1	AES permutation	6	12
2	Feistel	15	30
3	Type-1 GFS	21	42
≥4(6, 8 제외)	Type-2 GFS	(Feistel round) 15	60
6	Suzaki-Minematsu Improved Type-2 GFS	(Feistel round) 15	90
8	Suzaki-Minematsu Improved GFS	(Feistel round) 18	144

Simpira 최적화

- 라운드 상수를 사용한 고정된 라운드키를 사용하기 때문에 **사전 연산 가능**

Algorithm 3 $\mathrm{F}_{c,b}$ Algorithm (b=1)

procedure $\mathrm{F}_{c,b}(state)$

1: $RK[0] = \texttt{0x00} \oplus c \oplus b$
2: $RK[4] = \texttt{0x10} \oplus c \oplus b$
3: $RK[8] = \texttt{0x20} \oplus c \oplus b$
4: $RK[12] = \texttt{0x30} \oplus c \oplus b$
5: **return** $AES(AES(state, RK), Z)$

end procedure

Simpira 최적화

- 하나의 라운드당 2개의 라운드키 사용 총 12번의 라운드키 사용
  - 상수 라운드 키와 라운드키 전체가 0x00인 경우
  - 6라운드에서 0x00을 라운드 키로 사용

- AES-NI 대신 직접 AES 연산을 개별 구현하기 때문에 생략 가능

RK[0]	0x00 ⊕ c ⊕ b	RK[4]	0x10 ⊕ c ⊕ b	RK[8]	0x20 ⊕ c ⊕ b	RK[12]	0x30 ⊕ c ⊕ b
RK[1]	0x00	RK[5]	0x00	RK[9]	0x00	RK[13]	0x00
RK[2]	0x00	RK[6]	0x00	RK[10]	0x00	RK[14]	0x00
RK[3]	0x00	RK[7]	0x00	RK[11]	0x00	RK[15]	0x00

Figure 2. Values of each roundkey; RK = Roundkey, c is a counter that is initialized by one, and incremented after every use of $F_{c,b}$, Every $F_{c,b}$ consists of two AES round, where the round constants that are determined from (c, b), b is number of blocks.

Simpira 최적화

- InvMixColumn 연산 생략

Algorithm 3 $F_{c,b}$ Algorithm (b=1)

procedure $F_{c,b}(state)$

1: $RK[0] = \texttt{0x00} \oplus c \oplus b$
2: $RK[4] = \texttt{0x10} \oplus c \oplus b$
3: $RK[8] = \texttt{0x20} \oplus c \oplus b$
4: $RK[12] = \texttt{0x30} \oplus c \oplus b$
5: **return** $AES(AES(state, RK), Z)$

end procedure

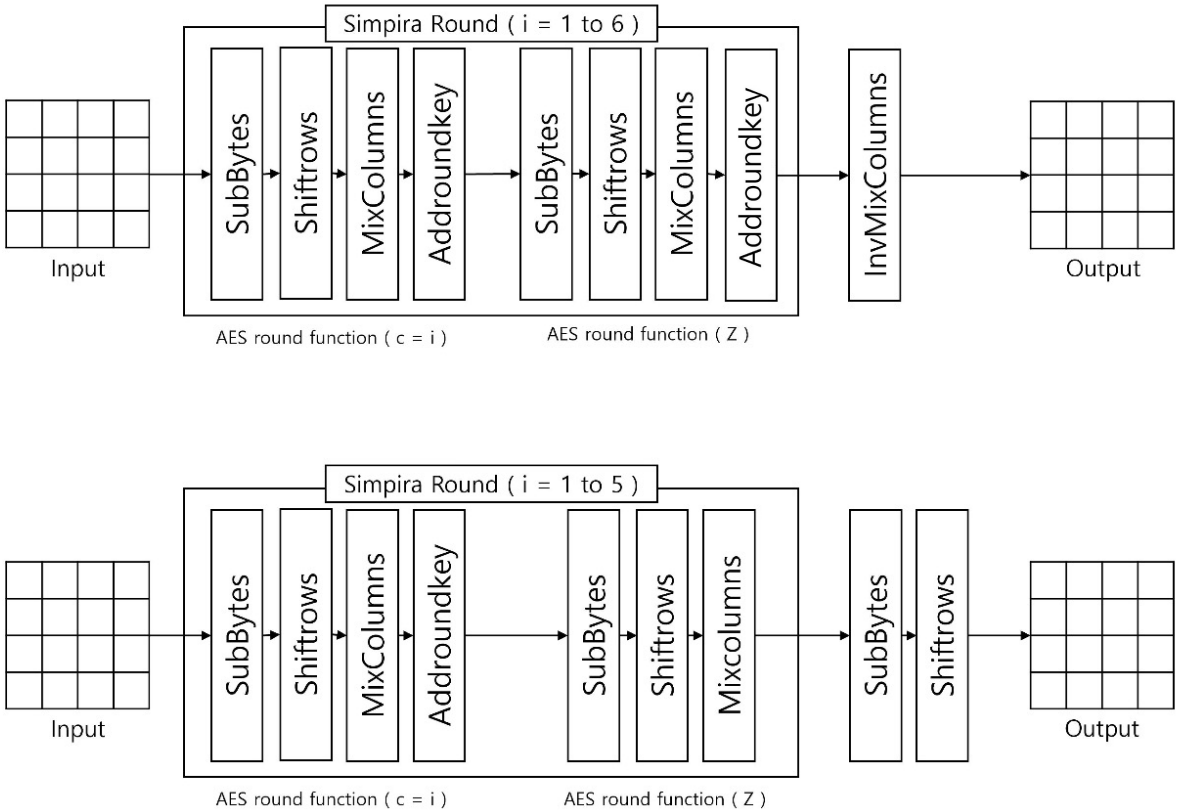


Figure 3. (Top) original Simpira structure / (Bottom) optimized Simpira structure.

성능 평가

- AVR : 레퍼런스 대비 5.76% 성능 향상
- RISC-V : 레퍼런스 대비 37.01% 성능 향상

Implementation	Processor	Code size
Our work	AVR	2,122
	RISC-V	1,806
Our work*	AVR	1,978
	RISC-V	1,281

Implementation	Processor	Clock cycles
Ref-C	AVR	14,334
	RISC-V	38,942
Our work	AVR	2,862
	RISC-V	1,106
Our work*	AVR	2,485
	RISC-V	1,052

Q & A