AES S-box에 대한 양자 구현의 T-depth 관점에서의 최적화

오제노*, 차재영*

*고려대학교 (학부생)

Optimizating Quantum Implementation of AES S-box at a perspective of T-depth

Zeno Oh*, Jaeyoung Cha*

*Korea University (Undergraduate)

요 약

본 논문에서는 기존의 AES S-box의 양자 회로에서의 T-depth를 기존의 연구보다 최적화하는 방법을 제시한다. 이 과정에서 GF(2^4)에서의 역원을 구하는 새로운 방법을 제시하며, 이때 필요한 qubit 수와 T-depth를 계산한다.

I. 서론

AES를 Grover's algorithm[1]에 적용하기 위해서는 AES를 양자 회로로 구현하여야 한다. 2015년 Grassl은 AES를 양자 회로로 구현하여 주어진 평문-암호문 쌍에 대해 키를 찾는 문제를 Grover's algorithm을 적용하여 풀어낼 수 있음을 설명하였으나[2], 이 회로를 구현하기 위해서는 qubit와 gate 수가 지나치게 많이 필요하다는 문제가 있다. 본 논문에서는 AES, 그 가운데에서도 SubByte 과정에 대하여 qubit 수와 T-depth 관점에서 기존에 알려진 결과보다 효율적으로 수행하는 양자 회로를 제시한다.

II. 배경 지식

2.1 Grover's Algorithm

Grover's algorithm은 다음과 같은 function $f:\{0,1\}^n \mapsto \{0,1\}$에 대하여 quantum oracle $U_f:|x\rangle|y\rangle \mapsto |x\rangle|y\oplus f(x)\rangle$과 초기 상태인 $|\psi\rangle = H^{\otimes n}|0\rangle$가 주어졌을 때 $f(x_0)=1$을 만족하는 $x_0 \in \{0,1\}^n$을 다음과 같은 Grover iteration Q를 이용하여 찾는다[1].

$$Q = -H^{\otimes n}(I-2|0\rangle\langle 0|)H^{\otimes n}U_f$$

$f(x_0)=1$을 만족하는 x_0가 M개 존재한다고 할 때, 초기 상태 $|\psi\rangle$에 Grover iteration Q를 $\left\lfloor \frac{\pi}{4}\sqrt{\frac{N}{M}} \right\rfloor$ 번 적용하는 것으로 최소 $1-\frac{M}{N}$의 확률로 원하는 상태를 관측할 수 있다.

2.2 Multiplicative Inversion in GF(2^8)

2015년 Grassl은 AES의 모든 과정에 reversible quantum circuit을 구현하였고, 그 중 SubBytes 과정에서만 T gate를 필요로 한다는 것을 설명하였다[2]. 그러나 Grassl은 SubBytes 과정에서 GF(2^8) 상에서의 역원을 계

산하기 위해 GF(2^8) 상에서의 곱셈을 수행하는 회로를 삽입하였는데, 이 때문에 회로의 T-depth가 지나치게 크다는 문제가 있다. 이후 AES에서의 양자 구현을 다루는 많은 연구들은 GF(2^8) 상에서의 역원을 구하는데 소모되는 자원을 줄이는 것에 집중하고 있다[3, 4, 5].

2020년 Chung은 tower-field construction을 이용하여 GF(2^8) 상에서의 역원을 구하는 양자 회로를 비교적 적은 수의 qubit과 T-depth로 구현하였다[5, Table 1].

method	type	width	T-depth
Grassl[2]		44	217
Lagenberg[3]		32	120
Jaques[4]	balanced	41	35
	minimum depth	137	6
Chung[5]	minimum width	32	36
	balanced 1	34	31
	balanced 2	36	30
	minimum depth	54	20

Table 1: Resource comparison among researches

본 연구에서는 Chung의 회로에서 GF(2^4) 상에서의 역원을 구하는 부분에 대한 개선점을 제시한다. Chung의 연구에서는 이를 양자 회로로 구현할 때에 18개의 qubit과 T-depth 8을 요구한다. 본 연구에서는 회로 전체에 사용되는 qubit 수를 크게 늘리지 않으면서도 T-depth는 감소시키는 양자 회로를 구현하였다.

III. AES 양자 회로 개선점

3.1 Multiplicative Inverse in GF(2^4)

본 논문에서는 GF(2^4) 상에서의 역원 다항식의 계수 각각의 형태를 직접 구현하는 방법으로 이 문제에 접근하였다.

GF(2^4)의 원소를 표현하는 다항식 $a_3x^3+a_2x^2+a_1x^1+a_0$의 역원이 되는 다항식을 $b_3x^3+b_2x^2+b_1x^1+b_0$라 할 때 계수 b_i는 다음과 같이 나타났다.

$$b_i=\begin{cases}a_1+a_2+a_3+a_0a_3+a_1a_3+a_2a_3+a_1a_2a_3 & (i=0)\\ a_2+a_3+a_0a_1+a_0a_2+a_0a_3+a_0a_2a_3 & (i=1)\\ a_3+a_0a_1+a_0a_2+a_1a_2+a_1a_3+a_0a_1a_3 & (i=2)\\ a_0+a_1+a_2+a_3+a_0a_2+a_1a_2+a_0a_1a_2+a_1a_2a_3 & (i=3)\end{cases}$$

3.2 Proposed Quantum Circuit

각각의 b_i가 a_i들의 곱과 합으로 표현된다는 것은 a_0, a_1, a_2, a_3가 양자 회로의 input으로 주어졌을 때 XOR gate와 AND gate를 적절히 조합하여 b_0, b_1, b_2, b_3를 output으로 내놓는 것이 가능하다는 것을 의미한다. XOR gate는 CNOT gate를 이용하여 간단하게 구현할 수 있는 반면, AND gate는 T-depth가 1인 회로이므로 AND gate를 남용할 경우 양자 회로 전체의 T-depth가 크게 늘어나게 된다. 본 논문에서는 T-depth를 줄이기 위해서 다음 두 가지 *type1* scheme과 *type2* scheme을 제안한다.

3.2.1 Multiplicative Inverse in GF(2^4)

type1 scheme은 아래 제시된 수식의 관점에 따라 b_0, b_1, b_2, b_3를 양자 회로로 구현한다. 각각의 b_i들은 대략적으로 $a_i(a_ja_k+\dots)+\dots$ 의 형태로 표현이 가능하므로, 먼저 AND gate를 활용하여 a_ja_k 형태의 state를 모두 구현하고(step 1), XOR gate로 $a_ja_k+\dots$ 형태의 state를 구현한다(step 2). 이후, AND gate를 적용하여 $a_i(a_ja_k+\dots)$ 형태의 state을 구현하고(step 3), 최종적으로 XOR gate로 최종 목표인 b_i state에 도달한다(step 4).

$$\begin{aligned}b_0&=a_1+a_2+a_3+a_0a_3+a_1a_3+a_2a_3+a_1a_2a_3\\&=a_3(a_1a_2+a_0+a_1+a_2)+(a_1+a_2+a_3)\end{aligned}$$

$$\begin{aligned}b_1&=a_2+a_3+a_0a_1+a_0a_2+a_0a_3+a_0a_2a_3\\&=a_0(a_2a_3+a_1+a_2+a_3)+(a_2+a_3)\end{aligned}$$

$$\begin{aligned}b_2&=a_3+a_0a_1+a_0a_2+a_1a_2+a_1a_3+a_0a_1a_3\\&=(a_1a_3+a_2)(a_0+a_1)+(a_0a_1+a_3)\end{aligned}$$

$$\begin{aligned}b_3&=a_0+a_1+a_2+a_3+a_0a_2+a_1a_2+a_0a_1a_2+a_1a_2a_3\\&=a_2(a_1a_3+a_0a_1+a_0+a_1)+(a_0+a_1+a_2+a_3)\end{aligned}$$

$t_0=a_1\times a_2$	$t_1=t_0+a_0+a_1+a_2$	$t_2=a_3\times t_1$
$b_0=t_2+a_1+a_2+a_3$	$t_3=a_2\times a_3$	$t_4=t_3+a_1+a_2+a_3$
$t_5=a_0\times t_4$	$b_1=t_5+a_2+a_3$	$t_6=a_1\times a_3$
$t_7=t_6+a_2$	$t_8=a_0+a_1$	$t_9=t_7\times t_8$
$t_{10}=a_0\times a_1$	$b_2=t_9+t_{10}+a_3$	$t_{11}=t_6+t_{10}+a_0+a_1$
$t_{12}=a_2\times t_{11}$	$b_3=t_{12}+a_0+a_1+a_2+a_3$	

초기 input a_0, a_1, a_2, a_3에서부터 t_0, t_3, t_6, t_{10}

을 구하는 과정과, t_2, t_5, t_9, t_{12}을 구하는 과정은 16개의 qubit을 도입하여 병렬적으로 처리할 수 있고, 그 외의 과정은 모두 XOR gate를 통해 구현이 가능하다. 즉, b_0, b_1, b_2, b_3를 구하는 연산은 T-depth 2인 회로로 구현할 수 있다.

한편, b_0, b_1, b_2, b_3를 0으로 되돌리는 역연산 회로를 구현하는 방법으로 다음 두 가지 방법을 고려할 수 있다.

첫 번째 방법은 a_0, a_1, a_2, a_3로부터 b_0, b_1, b_2, b_3을 생성할 때 구해놨던 t_0, t_3, t_6, t_{10}을 재사용하는 것이다. 이 방법은 4개의 qubit을 추가로 요구하지만, T-depth를 늘리지 않는다. 이 inversion 회로와 inversion 역연산 회로를 더불어 *type1.D* scheme(D: Depth)이라 하겠다.

두 번째 방법은 역연산을 수행하는 시점에서 t_0, t_3, t_6, t_{10}을 다시 계산하는 방법이다. 이 방법은 t_0, t_3, t_6, t_{10}을 저장하고 있을 필요가 없으나, t_0, t_3, t_6, t_{10}을 재계산하기 위한 자원을 필요로 한다. 이 inversion 회로와 inversion 역연산 회로를 더불어 *type1.W* scheme(W: Width)이라 하겠다.

3.2.2 *type2* scheme

type2 scheme은 아래 제시된 수식의 관점에 따라 b_0, b_1, b_2, b_3를 양자 회로로 구현한다.

$$b_0 = a_1 + a_2 + a_3 + a_0a_3 + a_1a_3 + a_2a_3 + a_1a_2a_3$$
$$= a_3(a_1a_2 + a_0 + a_1 + a_2) + (a_1 + a_2 + a_3)$$

$$b_1 = a_2 + a_3 + a_0a_1 + a_0a_2 + a_0a_3 + a_0a_2a_3$$
$$= a_3(a_0a_2 + a_0) + (a_0a_1 + a_0a_2 + a_2 + a_3)$$

$$b_2 = a_3 + a_0a_1 + a_0a_2 + a_1a_2 + a_1a_3 + a_0a_1a_3$$
$$= a_3(a_0a_1 + a_1) + (a_0a_1 + a_0a_2 + a_1a_2 + a_3)$$

$$b_3 = a_0 + a_1 + a_2 + a_3 + a_0a_2 + a_1a_2 + a_0a_1a_2 + a_1a_2a_3$$
$$= a_1a_2(a_0 + a_3) + (a_1a_2 + a_0a_2 + a_0 + a_1 + a_2 + a_3)$$

$t_0 = a_1 \times a_2$	$t_1 = t_0 + a_0 + a_1 + a_2$	$t_2 = a_3 \times t_1$
$b_0 = t_2 + a_1 + a_2 + a_3$	$t_3 = a_0 \times a_2$	$t_4 = t_3 + a_0$
$t_5 = a_3 \times t_4$	$t_6 = a_0 \times a_1$	$b_1 = t_5 + t_6 + t_3 + a_2 + a_3$
$t_7 = t_6 + a_1$	$t_8 = a_3 \times t_7$	$b_2 = t_8 + t_6 + t_3 + t_0 + a_3$
$t_9 = a_0 + a_3$	$t_{10} = t_0 \times t_9$	$b_3 = t_{10} + t_0 + t_3 + a_0 + a_1 + a_2 + a_3$

type2 scheme은 *type1* scheme과 비교했을 때 두 가지 차이점이 있다. 첫 번째는 step 1 과정에서 계산하는 state의 개수가 3개로, *type1* scheme 비교해서 1개가 줄었다는 점이다. 두 번째는 step 3 과정에서 4개의 AND 연산을 수행할 때 4개 중에 3개의 연산이 a_3를 input으로 사용한다는 점이다.

앞서 설명한 차이점들에 의해 회로에서 달라지는 부분들이 존재한다. 먼저, step 3에서 필요로 하는 qubit 수가 증가한다. step 3은 AND 연산 4개를 병렬로 수행하여 t_2, t_5, t_8, t_{10} state을 구하는 단계이다. 여기서 t_2, t_5, t_8은 모두 a_3를 input으로 사용하기 때문에 a_3 state를 담고 있는 qubit 2개를 추가로 생성해 주어야만 AND gate 4개를 병렬로 배치할 수 있다. 이 점으로부터 *type2* scheme은 *type1* scheme과 비교해서 2개의 qubit을 추가로 필요로 한다.

다음으로, *type2* scheme의 역연산 회로에도 변화가 생긴다. *type2* scheme의 역연산 회로도 역연산을 수행하는 시점까지 t_0, t_3, t_6을 저장하는지의 여부에 따라 *type2.D* scheme, *type2.W* scheme을 고려할 수 있다. *type2.D* scheme의 경우 *type1.D* scheme에 비해 저장하고 있어야 할 state 개수가 4개에서 3개로 감소하며, *type2.W* scheme의 경우에도 *type1.W* scheme에 비해 AND 연산 횟수가 1회 감소한다.

IV. 평가

4.1 Multiplication Inversion in GF(2^4)

본 논문에서는 GF(2^4) 상에서의 곱셈 역원을 연산하는 회로로 *Type1.D* scheme, *Type1.W* scheme, *Type2.D* scheme, *Type2.W* scheme을 제시하였다. 각각의 회로에 필요한 qubit 수, T-depth, T gate 수를 계산하여 Chung의 연구와 비교한 결과를 Table 2에 정리하였다.

		# qubit	T-depth	# T
Chung[5]	$X_{(4),A_3}^{-1}$	18 (save : 2)	6	36
	$(X_{(4),A_3}^{-1})^{\dagger}$	16	2	12
	Total	18, 16	8	48
Oh, Cha	*Type1.D*	20 (save : 4)	2	32
	$(Type1.D)^{\dagger}$	12	0	0
	Total	20, 12	2	32
	Type1.W	16	2	32
	$(Type1.W)^{\dagger}$	20	1	16
	Total	16, 20	3	48
	Type2.D	21 (save : 3)	2	28
	$(Type2.D)^{\dagger}$	11	0	0
	Total	21, 11	2	28
	Type2.W	18	2	28
	$(Type2.W)^{\dagger}$	17	1	12
	Total	18, 17	3	40

Table 2: Multiplication Inversion in GF(2^4) 구현에 필요한 자원량 비교

Type1.D scheme과 *Type2.D* scheme는 역연산을 수행할 때까지 일부 state들을 저장하고 있어야 하므로 보다 많은 수의 qubit을 필요로 하지만, 그 대신 T-depth와 T gate 수는 크게 절약할 수 있었다. *Type2.W* scheme은 앞선 두 scheme에 비해 상대적으로 적은 수의 qubit을 필요로 하는 반면 T-depth와 T gate 수는 조금 늘었지만, 여전히 Chung의 연구보다는 효율적인 모습을 보였다. *Type1.W* scheme의 경우, 역연산 회로에서 지나치게 많은 자원을 요구하다 보니 앞선 scheme들과 비교했을 때 우위에 서는 점이 없었다.

4.2 Multiplicative Inversion in GF(2^8)

앞서 제시한 scheme들은 GF(2^4) 상에서의 역원을 계산하는 회로이며, 이것은 GF(2^8) 상에서의 역원을 계산하는 회로의 일부이다.

Chung의 연구에서는 GF(2^8) 상에서의 역원을 계산하는 회로로써 min width, balanced 1, balanced 2, min depth를 제시하였기에, 각각의 회로에 본 논문에서 제시한 scheme들을 적용하여 얻은 결과를 qubit 수와 T-depth 관점에서 분석하여 Table 3에 정리하였다.

			min width	balanced 1	balanced 2	min depth
Chung [5]	$X^{-1}_{(4),A_3}$ scheme	# Qubit	32	34	36	54
		T-depth	36	31	30	20
Oh, Cha	*Type1.D* scheme	# Qubit	34	36	38	54
		T-depth	30	25	24	14
	Type1.W scheme	# Qubit	36	36	36	54
		T-depth	31	26	25	15
	Type2.D scheme	# Qubit	33	35	37	54
		T-depth	30	25	24	14
	Type2.W scheme	# Qubit	33	34	36	54
		T-depth	31	26	25	15

Table 3: Multiplication Inversion in GF(2^8) 구현에 필요한 자원량 비교

V. 결론 및 제언

Chung은 GF(2^8) 상에서의 역원을 구하기 위하여 order가 낮은 galois field GF(2^4)를 활용하였고, GF(2^4)에서의 곱셈 및 곱셈에 대한 역원을 계산하는 회로를 설계하여 적용하였다. 본 논문에서는 GF(2^4) 상에서 곱셈에 대한 역원을 구하는 새로운 방법을 제시하였고, 해당 방법을 양자 회로로 구현할 때에 필요한 qubit 수와 T-depth가 기존의 방법보다 효율적임을 확인하였다. 이제, GF(2^4)에서의 곱셈 회로가 사용되는 횟수를 줄이거나, GF(2^4)에서의 곱셈 회로 자체를 보다 최적화하는 방향으로 AES의 SubBytes 양자 회로 구현에 필요한 자원량을 줄일 수 있을 것으로 기대한다.

[참고문헌]

[1] Grover, Lov K. "A fast quantum mechanical algorithm for database search." Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. 1996.

[2] Grassl, M., Brandon Langenberg, M. Rötteler and R. Steinwandt. “Applying Grover's Algorithm to AES: Quantum Resource Estimates.” PQCrypto. 2016.

[3] Langenberg, Brandon, Hai-Son Pham and R. Steinwandt. “Reducing the Cost of Implementing AES as a Quantum Circuit.” IEEE Transactions on Quantum Engineering 1 (2020): 1-12.

[4] Jaques, Samuel, M. Naehrig, M. Roetteler and Fernando Virdia. “Implementing Grover Oracles for Quantum Key Search on AES and LowMC.” Advances in Cryptology - EUROCRYPT 2020 12106 (2019): 280 - 310.

[5] Chung, Doyoung, Jooyoung Lee, Seungkwang Lee and Dooho Choi. “Towards Optimizing Quantum Implementation of AES S-box.” IACR Cryptol. ePrint Arch. 2020 (2020): 941.