コンピュータセキュリティ　14－17
（2001．7．25）

AES 暗号用 SubBytes,MixColumns 変換の方式設計

清家　秀律　　黒川　恭一

防衛大学校　情報工学科

〒239-8686　神奈川県横須賀市走水 1-10-20
Tel: 0468-41-3810

E-mail: {g39051,kuro}@cc.nda.ac.jp

あらまし

近年、ネットワークの発達、及び、コンピュータの高機能化により、DES の安全性が低下している。NIST は、これに伴い AES の標準化を行い、昨年 Rijndael が AES に決定した。

本論文では、AES のハードウェア化を行うため、SubBytes および MixColumns 変換部を様々な方式で設計した。さらに、これらの方式を再構成可能素子(FPGA)を用いて implementation し、速度及び回路規模等の面から比較・検討を行った。

キーワード　AES,Rijndael,FPGA,hardware,SubBytes,MixColumns

An architectural design of SubBytes and MixColumns for AES cryptography

Hidenori Seike　　Takakazu Kurokawa

Department of Computer Science,
National Defense Academy

1-10-20 Hashirimizu, Yokosuka-City, Kanagawa 239-8686, Japan
Tel: +81-468-41-3810

E-mail: {g39051,kuro}@cc.nda.ac.jp

Abstract

In recent years, security of DES is decreasing according to the progress of network technology as well as computer science. As a result, NIST carried out the standardization of AES and decided Rijndael for AES last year.

In this paper, we designed SubBytes and MixColumns transformation in various ways to realize an AES hardware system on a FPGA chip. Their performance is evaluated through their speed and circuit size.

key words　AES,Rijndael,FPGA,hardware,SubBytes,MixColumns

1 はじめに

DESの安全性低下に伴い、NISTはAESの標準化を行い、昨年10月RijndaelがAESに決定した。そして、90日間の一般からのコメント期間が5月29日に終了し、今年夏頃にNISTから正式に発表されることとなっている。

本研究では、AES 暗号のハードウェア化を行うため、SubBytes変換及びMixColumns変換部を様々な方式で設計した。さらに、設計したそれぞれの方式をXilinx社製FPGAであるVirtexにimplementationし、回路規模及び速度の面から比較検討を行った。

4章では、SubBytes変換の方式設計及びimplementation結果、5章ではMixColumns変換の方式設計及びimplementation結果について述べる。

2 AES(Rijndael)暗号の概要

AES暗号となったRijndaelは、ベルギーのJoan Daemen氏とVincent Rijmen氏により開発された暗号である。

アルゴリズムでは、128-bitの暗号化及び復号のdataと、128,192,256-bitの秘密鍵を使っている。

暗号化においては、SubBytes,ShiftRows,MixColumns,AddRoundKeyという4つの変換部により構成されたRoundを、入力された秘密鍵のbit数により10,12,14回実行する。ただし、Roundを実行する前に、入力dataにAddRoundKey変換が行われること、最終Roundは、MixColumns変換が実行されない点が例外となる。

復号においても、暗号化と同様であるが、Roundを構成している変換部がInvSubBytes,InvShiftRows,InvMixColumns変換に換わる。

拡大鍵の生成は、秘密鍵から行い,秘密鍵のbit数(128,192,256-bit)により生成される拡大鍵のbit数が決まる。

3 FPGAの概要

FPGAは、Xilinx社により開発された高速高集積度Field Plogrammable Gate Array(FPGA)で、開発システムであるXilinx社製Foundation等により作成した回路を専用のダウンロードケーブルを使用して、FPGAに書き込むことができる。また、外部ROM等を接続してそのROMにあらかじめ回路データをダウンロードしておけば、FPGA内部の回路をその時の状態に応じて書き換えることも可能である。

3.1 FPGAについて

今回、使用したFPGAは、Xilinx社製のVirtex XCV1000である。表1にXCV1000の特徴を示す。

Virtex Deviceは、CLB(Configurable Logic Block)のArray、これらを取り囲むIOB(Programmable Input/Output Block)、そして、これらすべてを相互接続する配線リソースの階層構造からなる。CLBは、ロジックを構成する機能エレメントで、その基本構成ブロックはLC(Logic Cell)と呼ばれるものである。1つのLCには、4個の入力function generator,carry logic,記憶エレメントが内蔵されている。LC内部のfunction generatorは、4入力Lookup Tableとしてimplementされている。さらに、function generatorとしての動作のほかに、各LUTは16×1 bitのシンクロナスRAMを提供することができるほか、同一CLB Slice内の2個のLUTを組み合わせて、16×2bit,32×1bitのシンクロナスRAM又は、16×1bitのDual Port RAMを構成することができる。このLC2個をまとめて1個のCLB Sliceが構成され、さらに2個のCLB Sliceをまとめて1個のCLBが構成されている。

表1 XCV1000の特徴

System Gate	1,124,022
CLB Slice	12,288
I/O Pin	512
Block RAM	32×4,096 bit

3.2 Block RAMについて

Virtexは、Device内部にCLBとは別に専用のRAM領域を持っている。各Block RAMは、4,096-bitのRAM領域を持ち、Single Port RAM,Dual Port RAMとして使用することができる。

3.3 Distributed Memoryについて

CLB内部にあるLUTを用いて構成をしたMemoryである。このMemoryは、使用の目的により、ROM又はRAMとして構成することが可能で、RAMの場合、Single PortRAM及びDual Port RAMという2種類がある。

3.3 constraints editorについて

constraints editorとは、FPGAのDevice内部におけるピン配置及び論理デザインをCLBに配置,配線を行う際に制約を与えるeditorである。これを使うことにより、対象とするデザインに特化した配置、配線が可能となる。

3.4 floor plannerについて

floor plannerは、設計した論理デザインをターゲットFPGAに配置を行うためのGUIを使用した配置ツール

である。これを用いることにより、事前に論理デザインの配置を決定したり、implementation 後、配置を変更したりすることができる。

4 SubBytes 変換

4.1 SubBytes 変換の概要

SubBytes 変換は、非線形の byte 参照テーブルである s-box を用いて構成されている。s-box は、入力 1-byte に対し 1-byte の出力を持つ。そのため、これを 16-byte (128-bit)分組み合わせることにより SubBytes 変換となる。この参照テーブルを利用して、それぞれの byte の値に対して独立に処理がなされる。s-box の参照テーブルを表 2 に示す。入力された 1-byte data に対し、上位 4-bit の 16 進表現を X、下位 4-bit の 16 進表現を Y として表から変換後の値を参照する。入力 data が{c5}ならば、X=c,Y=5 の値{a6}となる。

この s-box は置換可能であり、また、次のような 2 つの変換により構成されている。

1 GF(2^8)で逆数の乗算をする。ただし、{00}は、自分自身とする。

2 以下に示す affine 変換 (GF(2)) を適用する。

$$b'(x) = (x^7 + x^6 + x^2 + x) + b(x)(x^7 + x^6 + x^5 + x^4 + 1) \bmod (x^8 + 1)$$

また、s-box の affine 変換は、matrix 表現も可能であり、以下のように書くことができる。

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

SubBytes 変換を図で示すと図 1 のようになる。

入力 128-bit を 8-bit づつ 16 分割(16-byte)し、それを 4 行 4 列の 2 次元ブロック状に配置する。また,出力についても同様にブロック状に配置する。SubBytes 変換に対する入力ブロックの r 行 c 列目のデータ Sr,c(1-byte) を s-box で変換することにより出力 S'r,c (1-byte)が得られる。

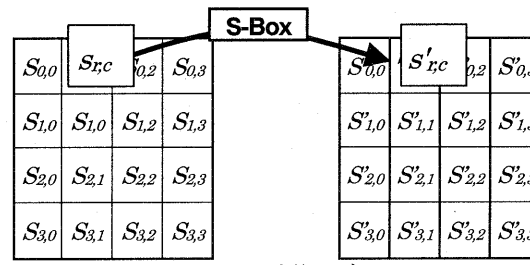

図 1 SubBytes 変換のブロック図

表 2 s-box の参照テーブル

X \ Y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
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	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

4.2 設計した方式

設計をした s-box は、SubBytes 変換中にある s-box を入力 1-byte に対し、出力 1-byte のテーブル参照方式で構成した。これらを 128-bit の入力分あわせることにより、SubBytes 変換となる。図2に s-box の構成図を示す。なお、図中にある s-box の部分が、4.2.1 以降に示す方式により構成されている。

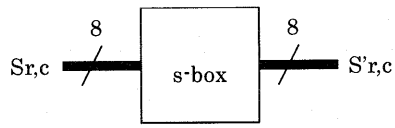

図2　s-box の構成図

4.2.1 Block RAM による方式

テーブル参照方式により、入力された 1-byte data を、BlockRAM の address とみなし、そこに SubBytes 変換後の値をあらかじめ入れておく。この BlockRAM を Single Port の場合は 16 個、Dual Port の場合は 8 個使用し、128-bit の SubBytes 変換を構成した。

4.2.2 VHDL による方式

入力された 1-byte data に対し、1 意に 1-byte の出力データが決まるため、256 通りの入出力を記述し、1-byte 入力 1-byte 出力という単位の組み合わせ回路を 16 個組み合わせて、128-bit の SubBytes 変換を構成した。この構成には clock を使用しないものと、clock を使用するものの 2 通りの方法を用いた。

4.2.3 Distributed ROM による方式

Block RAM と同様の方法により、Virtex 内部の LUT をベースとした ROM を使用して構成した。

また、ROM の内部構成は、data の出力に対し register を持たないもの、register を持ち 1 pipeline stage 及び、2 pipeline stage という 3 通りの方法を用いた。

4.2.4 Distributed RAM による方式

Distributed ROM と同様の方法を用いて構成した。ただし、RAM 内部のデータは一定で変わらないため、書き込み用バスは使用しない。

また、RAM の内部構成は、Single Port RAM の場合、register を持たないもの、register を持ち 1 pipeline stage 及び、2 pipeline stage という 3 通りの方法、Dual Port RAM の場合、register を持たないもの、register を持ち 1 pipeline stage、2 pipeline stage 及び、3 pipeline stage という 4 通りの方法を用いた。

4.3 implementation 結果

implementation 結果を表3に示す。この結果から、回路規模と速度について比較・検討を行った結果を以下に述べる。

4.3.1 回路規模

回路規模の面から比較を行った結果、特徴的な点を以下の2点にまとめることができる。

- Block RAM を使用した場合、CLB Slice,flip flop 及び 4 input LUT は 0 となるが、RAM 領域の gate 数がかさむため、total gate は大きくなってしまう。ただし、RAM 領域は CLB Slice を使用していないため、他の回路に使用可能な gate 数は大きくなる。結果から見れば、一番 total gate が少なかったものは、VHDL で記述し clock を使用しないものと、LUT を用いた ROM で register を持たないものである。しかし、先にも述べた通り、Block RAM を使用した場合、total gate は大きくなるが、CLB Slice を使用せず RAM 領域を使用しているため、もっともよい結果となった。
- clock の有無は、flip flop を持つか持たないか、すなわち、変換後の data を保持するための register が必要か否かということになる。ただし、Block RAM の場合、register を持っているが、RAM 領域の中に入っているため flip flop は不要となる。結局、AES 暗号全体を構成する際、SubBytes 変換後、data を保持するかということを考慮することにより、clock が必要かどうかが決まる。

4.3.2 速度

速度の面から比較を行った場合、clock を必要とする場合は Dual Port の Block RAM を、clock を必要としない場合は、遅延が最も少ない LUT で構成した ROM で register を持たないものを使用したものが最もよい結果が得られた。

AES 暗号の場合、多くの変換を round 数分実行しなければならないため、処理速度が速いほど変換速度も向上する。ただし、これは接続する機器の性能に依存する。また、速度が速くなる分、タイミングも厳しくなるため、より細かな設計が要求される。

4.3.3 総合

以上の検討の結果を総合的に判断すると、SubBytes 変換後 data を保持する場合は、Dual Port の Block RAM による方式が、また data を保持しない場合は Distributed RAM による方式で register のないものがよい。

表3　SubBytes 変換構成比較表

			CLB Slice	flip flop	4 input LUT	IOB	Block RAM	total gate	Minimum Period (ns)	Maximum Frequency (MHz)	Maximum net delay (ns)
BlockRAM	Single Port		0	0	0	258	16	269144	12.195	82.001	8.837
	Dual Port		0	0	0	258	8	138072	11.678	85.631	8.381
VHDL	clock なし		1152	0	2304	256	0	18816			13.936
	clock あり		1152	128	2304	257	0	19840	21.940	45.579	16.437
LUT ROM	register なし		1152	0	2304	256	0	18816			9.248
	register あり 1 pipeline stage		1152	128	2304	257	0	26840	15.983	62.566	14.479
	register あり 2 pipeline stage		1168	672	2304	257	0	31192	12.623	79.220	10.217
LUT RAM	Single Port	register なし	1280	0	2560	256	0	276440	11.083	90.228	9.815
		register あり 1 pipeline stage	1280	128	2560	257	0	277464	15.316	65.291	13.812
		register あり 2 pipeline stage	1312	1200	2560	257	0	286040	12.266	81.526	9.900
	Dual Port	register なし	1600	0	3200	256	0	147288	14.178	70.532	13.379
		register あり 1 pipeline stage	1600	128	3200	257	0	148312	19.274	51.883	12.728
		register あり 2 pipeline stage	1680	2288	3200	257	0	165592	14.115	70.847	12.183
		register あり 3 pipeline stage	1680	2544	3216	257	0	169688	15.450	64.725	13.518

ただし、対象とする FPGA により Block RAM の数が異なるため、MixColumns 変換で Block RAM を使用した場合は、SubBytes 変換では Dual Port の Block RAM による方式のかわりに register を持ち 2 pipeline stage で Single Port の Distributed RAM による方式がよい。

5　MixColumns 変換

5.1　MixColumns 変換の概要

MixColumns 変換は、それぞれの列を 4 項の多項式とみなし、列ごとの値を処理する。この列は、GF(2^8)上の多項式とみなし x^4+1 で mod をとった固定した多項式 a(x) との乗算となる。なお固定した多項式 a(x)は、

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

で与えられる。

また、この多項式は行列の積として表現することも可能で、

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad 0 \le c < Nb$$

となる。

MixColumns 変換を図で表すと図 3 のようになる。MixColumns 変換に対して入力される c 列目のデータ $S_{0,c},S_{1,c},S_{2,c},S_{3,c}$ を MixColumns 変換で変換することにより $S'_{0,c},S'_{1,c},S'_{2,c},S'_{3,c}$ が得られる。

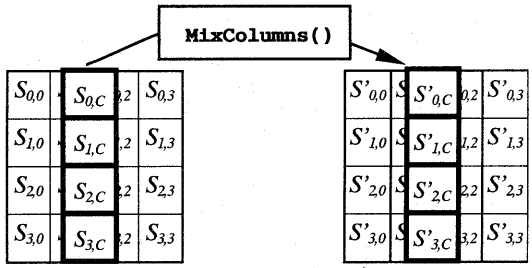

図3　MixColumns 変換のブロック図

5.2　設計した方式

設計をした MixColumns 変換は、入力 4-byte、出力 4-byte を 1 つのブロックとし、これらを 4 個組み合わせることにより 128-bit の MixColumns 変換となる。図4に MixColumns 変換の 1 ブロック分の構成図を示す。図中で 02 logic,03 logic と書いてある部分が、{02},{03}との乗算部分であり、内部は 5.2.1 以降に示す方式により構成されている。

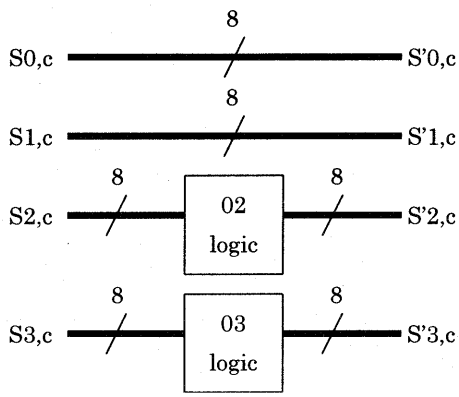

図4　MixColumns 変換の構成図(4-byte)

5.2.1　Block RAM による方式

table 参照方式により、入力された 1-byte data を RAM の address とみなし、MixColumns 変換の固定した多項式中の{02},{03}に対する入力 data との積をあらかじめ RAM 内部に格納しておく。ただし、入力 data と{01}との積は入力 data 自身であるため、{01}との積には RAM を使用しない。

この Block RAM を Single Port の場合は 32 個、Dual Port の場合は 16 個使用して MixColumns 変換を構成した。

5.2.2　schematic による方式

入力 data と固定した多項式の積は、部分部分で見てみると入力 data と{01},{02},{03}とのそれぞれの積となる。この点を利用し、入力 data と{02},{03}それぞれを求める乗算回路を構成した。ただし、入力 data と{01}との積は入力 data 自身であるため、乗算回路は使用しない。そして、これらを組み合わせて MixColumns 変換を構成した。

また、内部構成には、clock を使用しないものと、clock(2-clock)を使用するという 2 通りの方法がある。clock を用いる方法では、入力 data 1-byte を shift していき、2-clock で乗算結果が出力される。

5.2.3　Distributed ROM による方式

Block RAM と同様の方法を用いて、Virtex 内部の LUT をベースとした ROM を使用して、MixColumns 変換を構成した。

また、ROM の内部構成は、変換後の data 出力に対し、register を持たないもの、register を持ち 1 pipeline stage、及び、2 pipeline stage のものという 3 通りの方法を用いた。

5.2.4　Distributed RAM による方式

Distributed ROM と同様の方法を用いて、構成した。ただし、RAM 内の data は変化しないため、書き込み用のバスは使用しない。

また、RAM の内部構成は、変換後の data 出力に対し、Single Port RAM の場合、register を持たないもの、register を持ち 1 pipeline stage、及び、2 pipeline stage のものという 3 通りの方法、Dual Port RAM の場合、register を持たないもの、register を持ち 1 pipeline stage、2 pipeline stage および、3 pipeline stage のものという 4 通りの方法を用いた。

5.3　implementation 結果

implementation 結果を表 4 に示す。この結果から、回路規模と速度について比較・検討を行った結果を以下に述べる。

5.3.1　回路規模

回路規模の面から比較を行った結果、特徴的な点を以下の 2 点にまとめることができる。

- Block RAM を使用した場合、SubBytes 変換の implementation 結果と同様に、total gate は大きくなってしまうが、RAM 領域は CLB Slice を使用していないため、他の回路に使用可能な gate 数は大きくなる。結果から見れば、一番 total gate が少なかったものは、schematic で回路設計したもので clock を使用しないもの、すなわち、flip flop を持たない

ものであった。しかし、先にも述べた通り、Block RAM を使用した場合、total gate は大きくなるが、CLB Slice を使用せず RAM 領域を使用しているため、もっともよい結果となった。

- clock の有無は、SubBytes 変換の implementation 結果と同様に、AES 暗号全体を構成する際、MixColumns 変換後 data を保持するかということを考慮することにより、clock が必要かどうかが決まる。

5.3.2 速度

速度の面から比較を行った場合、clock を必要とする場合は Single Port の Block RAM を、clock を必要としない場合は遅延が最も少ない schematic で回路構成して register を持たないものが最もよい結果が得られた。

また、4.3.2 でも述べた通り、接続する機器やタイミング等についても考慮しなければならない。

5.3.3 総合

以上の検討の結果を総合的に判断すると、MixColumns 変換後 data を保持する場合は、Single Port の Block RAM による方式が、また data を保持しない場合は LUT を使用した ROM で register のない方式がよい。ただし、対象とする FPGA により Block RAM の数が異なるため、SubBytes 変換で Block RAM を使用した場合は、Single Port の Block RAM による方式のかわりに clock を使用した schematic による方式、もしくは register を持ち 1 pipeline stage の Distributed ROM による方式がよい。

表3 MixColumns 変換構成比較表

			CLB Slice	flip flop	4 input LUT	IOB	Block RAM	total gate	Minimum Period (ns)	Maximum Frequency (MHz)	Maximum net delay (ns)
BlockRAM	Single Port		64	0	128	258	32	532056	9.430	106.045	7.545
	Dual Port		64	0	128	258	16	269912	12.786	78.211	9.611
VHDL	clock なし		144	0	280	256	0	1680			7.510
	clock あり		273	257	280	258	0	11760	13.164	75.965	16.893
LUT ROM	register なし		2368	0	2944	256	0	27648			12.943
	register あり 1 pipeline stage		2368	256	2944	257	0	36696	17.733	56.392	14.457
	register あり 2 pipeline stage		2400	1344	2944	257	0	45400	18.011	55.522	15.853
LUT RAM	Single Port	register なし	2624	0	5248	256	0	546648	17.964	55.667	16.696
		register あり 1 pipeline stage	2624	256	5248	257	0	548696	21.542	46.421	17.276
		register あり 2 pipeline stage	2688	2400	5248	257	0	565848	18.112	55.212	15.744
	Dual Port	register なし	3264	0	6528	256	0	288344	19.417	51.501	18.831
		register あり 1 pipeline stage	3264	256	6528	257	0	290392	26.069	38.360	20.059
		register あり 2 pipeline stage	3424	4576	6528	257	0	324952	24.308	41.139	22.376
		register あり 3 pipeline stage	3424	5088	6560	257	0	333144	20.099	49.754	18.167

6 結論

AES暗号のSubBytes及びMixColumns変換についてFPGA化を念頭に入れて各種の方式設計を行い、4.3及び5.3のような結果が得られた。

SubBytes変換回路としては、その変換後dataを保持する場合は、Dual PortのBlock RAMによる方式が、またdataを保持しない場合はLUTを使用したROMでregisterのない方式がよい。

一方、MixColumns変換回路としては、その変換後dataを保持する場合は、Single PortのBlock RAMによる方式が、またdataを保持しない場合はLUTを使用したROMでregisterのない方式がよい。

ただし、SubBytes及びMixColumns変換両方でBlock RAMを使用した場合、対象とするFPGAによっては、Block RAMの数が足りなくなってしまう可能性がある。

その場合、SubBytes変換ではDual PortのBlock RAMによる方式のかわりにregisterを持ち2pipeline stageのDistributed ROMによる方式がよい。また、MixColumns変換では、Single PortのBlock RAMによる方式のかわりにclockを使用したschematicによる方式、もしくはregisterを持ち1pipeline stageのDistributed ROMによる方式がよい。

今回のimplementationは、それぞれの変換部単体で行ったが、AES暗号全体を構成する際、以下の点を考慮した上で、各変換部分の方式設計を選択する必要がある。

- 対象とするDeviceのtotal gate,Block RAM数
- 接続する機器の性能
- AES暗号の仕様

7 今後の課題

今後の課題としては、本論文による結論を元に、復号時のInvSubBytes及びInvMixColumns変換の設計を行い、暗号化及び復号の回路設計を行うことが残されている。さらに、それぞれの回路において動作確認をした後、暗号化及び復号の同一チップへの設計・実装を行うこともあげられる。

参考文献

[1] Xilinx,
http://www.jp.xilinx.com/
[2] 東京エレクトロンデバイス,
http://xpg.teldevice.co.jp/
[3] NIST,
http://csrc.nist.gov/encryption/aes/rijndael/
[4] Rijndael,
http://www.esat.kuleuven.ac.be/~rijmen/rijndael/
[5] Joan Daemen, Vincent Rijmen, "AES Proposal : Rijndael," Mar.9.1999
[6] NIST, "a Draft Federal Information Processing Standard (FIPS) for the AES," February 28, 2001
[7] NIST, "Public Comments on the Draft Federal Inf- ormation Processing Standard (FIPS)for the Advance- d Encryption Standard (AES),"
[8]吉田　たけお, 尾知　博, "特集 入門!VHDLによる回路設計の基礎," Design Wave magazine, pp.19-93, CQ出版, DEC 2000
[9]長谷川　裕恭, VHDLによるハードウェア設計入門, CQ出版, 1995