AES暗号回路におけるSubBytesの低消費電力化

世古　忠

Low Power Method of SubBytes for AES Encryption Circuit

Tadashi Seko

This paper presents a new method to reduce the power consumption of SubBytes circuit for AES(Advanced Encryption Standard) circuit. I propose to represent the logic function of S-Box by BDD(Binary Decision Diagram) compactly and then assign the multiplexer to every node in BDD to translate the BDD to the circuit representation. VHDL description of the SubBytes circuit is generated from BDD automatically and it is synthesized to evaluate the circuit of SubBytes using Synopsys's Design Vis on. The experimental evaluations show that the proposed method attains much reduction ratio of power consumption, area and delay.

1．はじめに

近年，無線LANを利用したデータ通信が普及し，情報通信の安全を高めるための暗号化技術が重要となっている．2001年にRijndaelが提案しNIST（National Institute of Standards and Technology）によって次期米国標準の暗号として選定された，共通鍵暗号AES（Advanced Encryption Standard）[1,2]が，最近盛んに用いられるようになってきた．AESにおける暗号化・復号化の処理は演算量が多く，処理の高速化のために専用のLSIが作成されることが多くなっている．近年，携帯電話などの携帯端末が急速に普及し，AESを組込み用途で利用されることが多いが，そのバッテリーの寿命の点から回路の消費電力に対する制約が厳しくなっている．しかし，従来，AESの回路アーキテクチャや回路の速度に関する性能について報告がなされているものの低消費電力化についての研究は少ないが，文献[3]の研究はハザードが少なくなるようなS-BOXの2段論理の構成法を提案している．

本研究では，AES暗号回路の消費電力の大半を占めるS-Boxと呼ばれる組み合わせ回路部に着目する．S-Box部は，入力された平文のデータを置換表により暗号化されたデータに変換する部分であり，この回路の実現方法として，積和形の二段論理を用いる方法や，論理関数をBDD（Binary Decision Diagram）[4]を用いて効率よく表現し，回路を構成する方法がある．BDDを用いる方法は，比較的高速で，コンパクトな回路を構成することができる．本研究では効率のよいBDDパッケージを用いて論理関数を表現し，そのBDDの各節点にマルチプレクサを割当て回路を構成することでS-Boxを実現し，さらに消費電力を削減する方法を提案する．

2．共通鍵暗号AES

2.1　暗号化アルゴリズム

図1にAES暗号化アルゴリズムの処理手順[2]を示す．AESへの入力は128ビットの平文と暗号化鍵であり，出力は暗号化された128ビットのデータである.暗号化鍵は128, 192, 256ビットの3種類ある．入力データは，下位より8ビットごとに$a_{0,0}$, $a_{0,1}$, ・・・, $a_{3,3}$の16個に分割され表1のような4×4の行列として扱われる．

入力に対し，4つの演算SubBytes，ShiftRows，MixColumns，AddRoundKeyを順に並べた処理を1ラウンドとして，何回も繰り返し適用する．暗号化処理におけるラウンド数は鍵長によって変わり，128ビット鍵長に対しては11, 192ビット鍵長の場合は13, 256ビット鍵長の場合は15である．最初のラウンドではAddRoundKeyだけが行われ，最終ラウンドではMixColumnsは適用されない．また，復号化処理は暗号化処理と逆対称となっており，上記の4種類の処理を逆に行う．

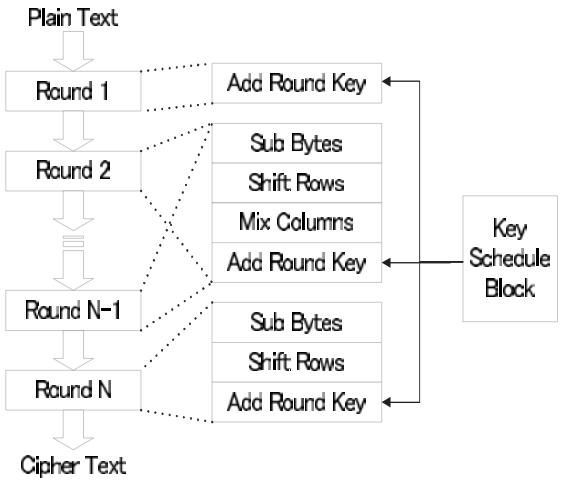

図 1: AES の暗号化アルゴリズム

表 1: データの表現

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

2.2 各処理部分

2.2.1 SubBytes

SubBytesは，行列の各要素に対してS-Boxと呼ばれる変換を適用する．S-Boxは8ビットデータを他の8ビットデータに変換するものであり，次の2つの処理が行われる．

⑴行列の8ビットデータをガロア体$GF(2^8)$（既約多項式は$m(x)=x^8+x^4+x^3+x+1$）上の元とみなし，その乗法逆元を求める．
⑵乗法逆元に対して$GF(2)$上の定数行列演算であるアフィン変換を行なって出力する．

SubBytes 部のアルゴリズムを実装する場合，上記の変換を施した表2のようなS-Box の変換表をあらかじめ用意しておき，これを真理値表とみなして論理回路を導く方法や，逆元演算回路とアフィン変換回路を別々に作り直列につなぐ方法などがある．

2.2.2 ShiftRows

ShiftRows 部では，表1に示すデータ行列の各行に対し，1行目は0ブロック，2行目は1ブロック，3行目は2ブロック，4行目は3ブロック左巡回シフトする処理を行う．表1のデータに対してShiftRowsの処理を行なった結果を表3に示す．

表 2: SubBytes の真理値表

	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

表 3: ShiftRows の処理結果

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,0}$
$a_{2,2}$	$a_{2,3}$	$a_{2,0}$	$a_{2,1}$
$a_{3,3}$	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$

2.2.3 MixColumns

MixColumns部ではデータ行列の各列に対して処理を行う．各列の$a_{0,j}$，$a_{1,j}$，$a_{2,j}$，$a_{3,j}$を$GF(2^8)$上の3次多項式の係数と考え，式⑴と式⑵に示すような多項式を定義する．

$$a(x)=a_{3,j}x^3+a_{2,j}x^2+a_{1,j}x+a_{0,j} \quad (1)$$
$$c(x)=03x^3+01x^2+01x+02 \quad (2)$$

MixColumnsによる処理結果は，$a(x)c(x) \bmod (x^4+1)$で定義され，求められた3次元多項式の係数4つを変換後の中間データ行列の$a_{i,j}$とする．

2.2.4 AddRoundKey

AddRoundKey部では，中間データ行列の各要素とラウンド鍵との排他的論理和を取ることで出力が得られる．

3. SubBytes部の消費電力削減法

3.1 AES暗号回路の消費電力

AES暗号回路全体の消費電力のうちSubBytes部で消費される電力が大半を占めており（典型的な回路構成では70 %以上［3］），SubBytes部の消費電力の削減が課題となっている．ここでは，SubBytes部の論理関数をBDDで表現し回路を実現する手法について述べる．

3.2 回路の実現法

3.2.1 S-Boxの2段論理による表現

S-Boxの論理関数は表2を真理表とみなして出力の各ビット毎にもとめることができる．例えば，表2より入力が$\{21\}_{16}$のとき出力は$\{FD\}_{16}$である．ここで，入力の$\{2\}_{16}$をa_7～a_4，$\{1\}_{16}$をa_3～a_0とし，出力の$\{F\}_{16}$をz_7～z_4，$\{D\}_{16}$をz_3～z_0とすると各ビット毎の論理式は以下の式(1)のような2段論理で表すことができる．

$$
\begin{aligned}
z_0 = {} & \bar{a}_7\bar{a}_6\bar{a}_5\bar{a}_4\bar{a}_3\bar{a}_2\bar{a}_1\bar{a}_0 + \bar{a}_7\bar{a}_6\bar{a}_5\bar{a}_4\bar{a}_3\bar{a}_2 a_1\bar{a}_0 \\
& + \bar{a}_7\bar{a}_6\bar{a}_5\bar{a}_4\bar{a}_3\bar{a}_2 a_1 a_0 + \bar{a}_7\bar{a}_6\bar{a}_5\bar{a}_4\bar{a}_3 a_2\bar{a}_1 a_0 \\
& + \bar{a}_7\bar{a}_6\bar{a}_5\bar{a}_4\bar{a}_3 a_2 a_1\bar{a}_0 + \bar{a}_7\bar{a}_6\bar{a}_5\bar{a}_4\bar{a}_3 a_2 a_1 a_0 \\
& + \bar{a}_7\bar{a}_6\bar{a}_5\bar{a}_4 a_3\bar{a}_2\bar{a}_1 a_0 + \bar{a}_7\bar{a}_6\bar{a}_5\bar{a}_4 a_3\bar{a}_2 a_1\bar{a}_0 \\
& + \bar{a}_7\bar{a}_6\bar{a}_5\bar{a}_4 a_3\bar{a}_2 a_1 a_0 + \bar{a}_7\bar{a}_6\bar{a}_5\bar{a}_4 a_3 a_2\bar{a}_1 a_0 \\
& + \bar{a}_7\bar{a}_6\bar{a}_5\bar{a}_4 a_3 a_2 a_1\bar{a}_0 + \bar{a}_7\bar{a}_6\bar{a}_5 a_4\bar{a}_3\bar{a}_2\bar{a}_1\bar{a}_0 \\
& + \bar{a}_7\bar{a}_6\bar{a}_5 a_4\bar{a}_3\bar{a}_2 a_1 a_0 + \bar{a}_7\bar{a}_6\bar{a}_5 a_4\bar{a}_3 a_2\bar{a}_1 a_0 \\
& + \bar{a}_7\bar{a}_6\bar{a}_5 a_4\bar{a}_3 a_2 a_1\bar{a}_0 + \bar{a}_7\bar{a}_6\bar{a}_5 a_4 a_3\bar{a}_2\bar{a}_1\bar{a}_0 \\
& + \cdots \qquad (3)
\end{aligned}
$$

3.2.2 逆元計算回路

SubBytes部の回路を実現する方法として以上述べたS-Boxの真理値表より積和形やBDDを構成する方法の他に，逆元演算回路とアフィン変換回路を直列につなぐ方法がある．逆元演算回路を作成する方法としては，伊藤・辻井のアルゴリズム［5］がある．これはガロア体$GF(2^8)$の要素yの逆元がy^{-1}=254で求まることから，乗算器を多数利用して図2のように回路を構成する．

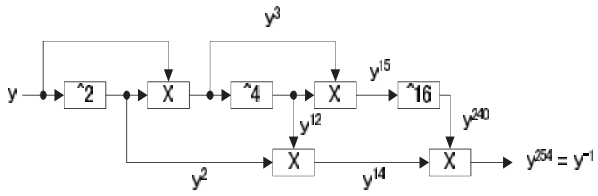

図 2: 伊藤・辻井のアルゴリズムによる逆元計算回路

3.3 BDDを用いた消費電力削減法の提案

本研究ではSubBytes部の論理関数を否定枝を用いたBDDを用いてコンパクトに表現し，BDDの各節点にマルチプレクサを割り当てVHDL記述の回路に変換し，消費電力優先の論理合成を行なうことによって消費電力を削減する方法を提案する．なお，BDD構築に当って消費電力が最小になるような変数順序を選択する．図3に提案する回路の生成手順を示す．はじめに，3.2.1でもとめたS-Boxの2段論理関数をBDD 表現に変換する．

その後，これをハードウェア記述言語VHDLで記述された回路表現に変換し，論理合成を行なう．

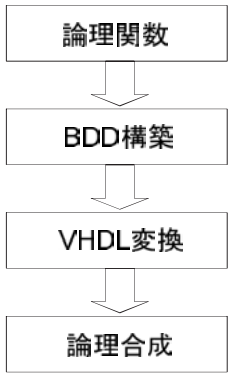

図 3: 回路の生成手順

3.3.1 BDD表現から回路記述への変換

BDD表現から論理回路へ変換する方法について述べる．BDDの各節点に対してマルチプレクサMUX(Multiplexer)を割り当て回路に変換する．例えば，論理関数f=A+BCのBDDの各変数節点にMUXを割り当て，定数節点に0,1の論理値を割り当て構成した回路を図4に示す．ここで回路は，変数の値をMUXの選択信号に入力することによって0と1の値がMUXによって出力に伝播し，最終的に論理関数の出力値がfに出力される．

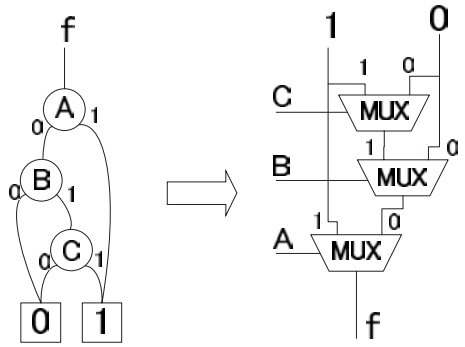

図 4: MUX による BDD の実現

3.3.2 否定枝付きBDDの回路変換法

論理関数を否定枝付きのBDD［6］で表現すると節点数が削減されることが多い．否定枝付きの節点をMUXに割り当てる場合，図5のように否定枝の丸印のところにはインバータを挿入する必要がある．この図においてxは選択信号であり，xが0のときはfに0が出力され，xが1のときは，fに1が出力される．

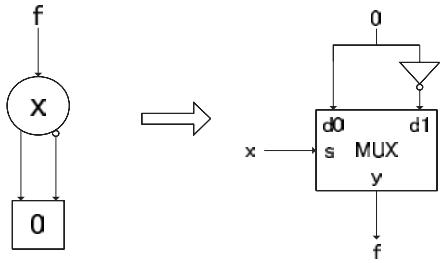

図 5: 否定枝を用いた BDD の MUX への割り当て方

4. 実験結果

提案手法を用いて自動生成した回路のVHDL記述を，東大VDECのDesign Visionを用いて論理合成し，回路の評価を行なった．表4に実験結果を示す．表4より提案法(BDD)は，積和形の方法および伊藤等の方法と比べて消費電力がそれぞれ7%及び69%削減された．また面積はそれぞれ23%，55%削減され，遅延時間もそれぞれ12%，79%削減され提案手法は消費電力だけではなく面積及び遅延時間の削減においても有効であることが確認できた．なお，表4の削減率1は，(A-C)/A×100［%］，削減率2は，(B-C)/B × 100［%］とする．

表 4: 実験結果

	消費電力 [μW]	面積 [ゲート]	遅延時間 [ns]
積和形 (A)	741.19	826.00	17.14
伊東・辻井 (B)	2213.40	1431.00	72.60
BDD(C)	688.51	639.00	15.00
削減率 1	7%	23%	12%
削減率 2	69%	55%	79%

5. まとめ

本論文ではAES暗号回路におけるSubBytes部の論理関数を否定枝を用いたBDDを用いて表現し，低消費電力化を行なう方法を提案した．提案法を評価するため実験を行なった結果，他の手法を用いた回路よりも消費電力，面積，遅延時間を削減することができ，本手法の有効性を確認した．今後は,本手法でSubBytesを構成した場合について，AES全体の回路の消費電力，面積，遅延時間の評価を行なう必要がある．

謝辞

本研究は東大VDEC提供のLSI設計CADツールを用いて行った事を記して感謝します．また，本研究の実験用プログラム作成に協力頂いた卒研生の百地康博君，専攻科特別研究生の木浦幹雄君に感謝します．

参考文献

［1］ http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, “ National Institute of Standard and Technology(NIST):Advanced Encryption Standard(AES)”, FIPS Publication 197(2001).

［2］ 神永正博，渡邊高志，“情報セキュリティの理論と技術”，森北出版株式会社，pp88-99，2005.

［3］ 森岡澄夫，佐藤証，“共通鍵暗号AES の低消費電力論理回路構成法”，情報処理学会論文誌，Vol.44， No.5，pp.1321-1328，2003.

［4］ Bryant,R.E.:Graph-Based Algorithms for Boolean Function Manipilation, IEEE Trans. Comput.,Vol.C-35,No.8,pp.677-691(1986).

［5］ 伊東利哉, 辻井重男,” 正規基底を用いた有限体における高速逆元算出アルゴリズム, ”, 信学論(A) ,vol.J70-A,no.11,pp.1637-1645,1987.

［6］ Shin-ichi Minato，*Binary Decision Diagrams and Applications For VLSI CAD*，KLUWER ACADEMIC PUBLISHERS，1996.