デバイスの記事　ビギナーズ

共通かぎ暗号AES用 SubBytes変換回路設計仕様書

Design Wave 設計コンテスト2004

和田知久

今回はワイヤレスLANなどで使用される次期米国標準の128ビット共通かぎブロック暗号AES（Advanced Encription Standard）のSubBytes変換回路の設計を行います．AESの暗号化アルゴリズムは，128ビット（16バイト）の入力データに対し，四つの基本演算（ShiftRows，SubBytes，MixColumns，AddRoundKey）を複数回繰り返し適用します．簡単に言えば，16枚のトランプがあったとして，それに対してシャッフルと，数字の変更を繰り返し行うことで，初期の16枚のトランプの数字の並びを暗号化してしまうものです．（筆者）

今回の設計課題では，AES（Advanced Encription Standard）の暗号化処理のすべてではなく，特に回路設計で楽しむことができるSubBytes変換回路だけを設計します．機能は非常に単純であり，8ビットの情報をほかの8ビットの並びへと，1対1の変換を行うだけです．したがって，256ワード×8ビットのROMで変換テーブル（S-boxと呼ばれている）を構成するだけで実現することも可能です．しかし，そのSubBytes変換は2^8のガロア体の逆元計算と簡単な行列変換なので，演算によって実現することで，小さく高速で消費電力の小さい回路としても実現できます．要求されている設計内容は，HDL（VHDLもしくはVerilog HDL）による設計と論理合成です．論理合成ツールに制約はなく，FPGA向けに無償で提供されている論理合成ツールでも参加できます．HDL設計に興味のある方はどしどし参加してください．また，余裕のある方はFPGAなどに実装すれば，努力が認められて高い評価が得られると思います．トライしてみてください．

今回想定するシステムのブロック図を**図1**に示します．システムは大きく分けて，8ビット，すなわちバイト・データ出力する送信器（SENDER），そのデータをリアルタイムで暗号化するSubBytes変換回路，そしてその暗号化されたデータを解読する逆SubBytes変換回路によって構成されます．今回はモード切り替えによりSubBytes変換もしくは逆SubBytes変換の両方を実行できる変換回路を設計します．

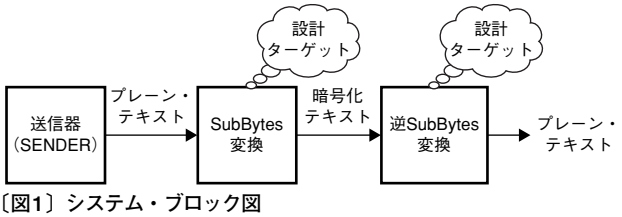
今回の課題ではガロア体の知識がなくても設計できるように，以下の仕様書でわかりやすく説明します．設計すべきことは比較的単純ですので，ディジタル設計の知識のある方は，自信を持って課題に取り組んでください．

SubBytes変換

SubBytes変換は，8ビットのバイナリ情報をほかの8ビットのビット列に変換するものです．例えば，00000000というすべて0の8ビットの情報は，01100011という8ビットのビット列に変換されます．

ここで重要なことは，異なるデータ（8ビットの数）は必ず異なるビット列に変換されるということです．もし，二つの異なるデータが同一のビット列に変換されてしまったら，変換後（暗号化後）の値からオリジナルの値に戻すことができなくなってしまいます．すなわち，

- 8ビットの数を変換しても8ビットの世界の中にいること（閉じている）
- 一義的な逆変換ができること（逆元が存在する）



〔図1〕システム・ブロック図

という特性があります．これは有限な数の世界の演算であるガロア体の性質を満たしています．

SubBytes変換は，二つの変換を直列に並べることで実現されます．

まず，2^8のガロア体$GF(2^8)$の逆数に変換します（値が異なる別の8ビットに変換される）．ただし，00000000（16進数表現で00h）は00000000（00h）に変換されます．

表1に逆数変換テーブルを示します．例えば，01010011という8ビットの数は，16進表示で53hになります．$X=5$，$Y=3$で表を参照すると，CAhという16進表示が得られます．これを2進数に変換すると，11001010になります．

ところで，ガロア体の性質は既約多項式で決定されます．ここで用いられている既約多項式は以下の式で与えられます．

$$m(x)=x^8+x^4+x^3+x+1$$

次に以下の行列変換を適用します．

$$\begin{bmatrix} b_7\\ b_6\\ b_5\\ b_4\\ b_3\\ b_2\\ b_1\\ b_0 \end{bmatrix} = \begin{bmatrix} 1&1&1&1&1&0&0&0\\ 0&1&1&1&1&1&0&0\\ 0&0&1&1&1&1&1&0\\ 0&0&0&1&1&1&1&1\\ 1&0&0&0&1&1&1&1\\ 1&1&0&0&0&1&1&1\\ 1&1&1&0&0&0&1&1\\ 1&1&1&1&0&0&0&1 \end{bmatrix} \times \begin{bmatrix} a_7\\ a_6\\ a_5\\ a_4\\ a_3\\ a_2\\ a_1\\ a_0 \end{bmatrix} \oplus \begin{bmatrix} 0\\ 1\\ 1\\ 0\\ 0\\ 0\\ 1\\ 1 \end{bmatrix}$$

ただし，各係数の演算は2^1のガロア体$GF(2)$を想定するので，加算は排他的論理和XORで計算されます．00000000（00h）は結果的に01100011（63h）に変換されます．

11001010（CAh）であれば，

$$\begin{bmatrix} b_7\\ b_6\\ b_5\\ b_4\\ b_3\\ b_2\\ b_1\\ b_0 \end{bmatrix} = \begin{bmatrix} 1&1&1&1&1&0&0&0\\ 0&1&1&1&1&1&0&0\\ 0&0&1&1&1&1&1&0\\ 0&0&0&1&1&1&1&1\\ 1&0&0&0&1&1&1&1\\ 1&1&0&0&0&1&1&1\\ 1&1&1&0&0&0&1&1\\ 1&1&1&1&0&0&0&1 \end{bmatrix} \times \begin{bmatrix} 1\\ 1\\ 0\\ 0\\ 1\\ 0\\ 1\\ 0 \end{bmatrix} \oplus \begin{bmatrix} 0\\ 1\\ 1\\ 0\\ 0\\ 0\\ 1\\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1\cdot1\oplus1\cdot1\oplus1\cdot0\oplus1\cdot0\oplus1\cdot1\oplus0\cdot0\oplus0\cdot1\oplus0\cdot0\oplus0\\ \cdots\\ \cdots\\ \cdots\\ \cdots\\ \cdots\\ \cdots\\ \cdots \end{bmatrix} = \begin{bmatrix} 1\\ 1\\ 1\\ 0\\ 1\\ 1\\ 0\\ 1 \end{bmatrix}$$

となり，11101101（EDh）に変換されます．つまり，53hは逆元計算でCAhに変換され，行列計算によりEDhに変換されます．これがSubBytes変換となります．

〔表1〕$GF(2^8)$の逆数変換テーブル

X＼Y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	8D	F6	CB	52	7B	D1	E8	4F	29	C0	B0	E1	E5	C7
1	74	B4	AA	4B	99	2B	60	5F	58	3F	FD	CC	FF	40	EE	B2
2	3A	6E	5A	F1	55	4D	A8	C9	C1	0A	98	15	30	44	A2	C2
3	2C	45	92	6C	F3	39	66	42	F2	35	20	6F	77	BB	59	19
4	1D	FE	37	67	2D	31	F5	69	A7	64	AB	13	54	25	E9	09
5	ED	5C	05	CA	4C	24	87	BF	18	3E	22	F0	51	EC	61	17
6	16	5E	AF	D3	49	A6	36	43	F4	47	91	DF	33	93	21	3B
7	79	B7	97	85	10	B5	BA	3C	B6	70	D0	06	A1	FA	81	82
8	83	7E	7F	80	96	73	BE	56	9B	9E	95	D9	F7	02	B9	A4
9	DE	6A	32	6D	D8	8A	84	72	2A	14	9F	88	F9	DC	89	9A
A	FB	7C	2E	C3	8F	B8	65	48	26	C8	12	4A	CE	E7	D2	62
B	0C	E0	1F	EF	11	75	78	71	A5	8E	76	3D	BD	BC	86	57
C	0B	28	2F	A3	DA	D4	E4	0F	A9	27	53	04	1B	FC	AC	E6
D	7A	07	AE	63	C5	DB	E2	EA	94	8B	C4	D5	9D	F8	90	6B
E	B1	0D	D6	EB	C6	0E	CF	AD	08	4E	D7	E3	5D	50	1E	B3
F	5B	23	38	34	68	46	03	8C	DD	9C	7D	A0	CD	1A	41	1C

逆SubBytes変換について

逆SubBytes変換は，SubBytes変換の操作を逆に行う変換に対応するので，以下の二つの変換を直列に並べることで実現できます．

まず，行列の逆変換を行います．結果的に以下の行列変換式となります．

$$\begin{bmatrix} a_7\\ a_6\\ a_5\\ a_4\\ a_3\\ a_2\\ a_1\\ a_0 \end{bmatrix} = \begin{bmatrix} 0&1&0&1&0&0&1&0\\ 0&0&1&0&1&0&0&1\\ 1&0&0&1&0&1&0&0\\ 0&1&0&0&1&0&1&0\\ 0&0&1&0&0&1&0&1\\ 1&0&0&1&0&0&1&0\\ 0&1&0&0&1&0&0&1\\ 1&0&1&0&0&1&0&0 \end{bmatrix} \times \begin{bmatrix} b_7\\ b_6\\ b_5\\ b_4\\ b_3\\ b_2\\ b_1\\ b_0 \end{bmatrix} \oplus \begin{bmatrix} 0\\ 0\\ 0\\ 0\\ 0\\ 1\\ 0\\ 1 \end{bmatrix}$$

そして，**表1**と同じ2^8のガロア体$GF(2^8)$の逆数に変換を行います．したがって，SubBytes変換も逆SubBytes変換

も同じ2^8のガロア体$GF(2^8)$の逆数計算を必要としています．

また，今回設計するのは切り替え信号により，SubBytes変換と逆SubBytes変換の機能を切り替えられる変換回路です．ガロア体の逆数回路を設計することで，**図2**のようなアーキテクチャによって今回の設計課題を実現できることになります．

INV信号が'0'のとき，入力はガロア体の逆数計算回路に入り，その出力は行列変換計算回路に入ります．逆にINV信号が'1'であれば，先に行列の逆変換が計算され，その出力はガロア体の逆数計算回路に入力され，逆数が計算されます．

ガロア体$GF(2^8)$の逆数計算と乗算

行列計算は式からもわかるようにANDゲートとXORゲートで実現できる単純な回路なので，$GF(2^8)$の逆数計算をいかにうまくやるかがこの課題の設計のポイントとなります．

結局のところ，逆数計算は8ビット入力に対して**表1**に示される値を出力すればよいだけです．したがって，表をそのままHDLで記述すれば，逆数演算回路を実現することができます．

しかし，それでは設計コンテストの課題としてはおもしろくありません．そこで，ここでは一つの実装例を示します．

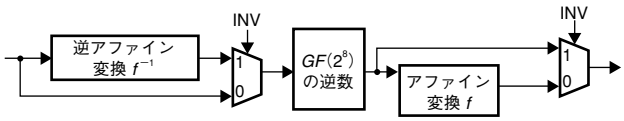
ある値yの逆数y^{-1}を求めるわけですが，$GF(2^8)$のガロア体の要素yには，

$$y^{2^8-1}=y^{255}=1$$

という性質があります．したがって，

$$y^{-1}=y^{-1}\cdot y^{255}=y^{254}$$

より，y^{254}を求めれば逆数が求まることになります．つまり，ガロア体の乗算器があればそれを繰り返し用いることでy^{254}を計算することができます．乗算器を多数用いた伊東・辻井のアルゴリズムによる逆元演算回路の例を**図3**に示します．



〔図2〕課題のアーキテクチャ例

ガロア体の乗算回路を繰り返し用いることで，逆元を計算できます．以下に，ガロア体の乗算回路の一つの実装例を示します．

ここで，二つの8ビットの数$A=(a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0)$，$B=(b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)$の乗算を考えます．二つの値は，以下のような多項式で表すことができます．

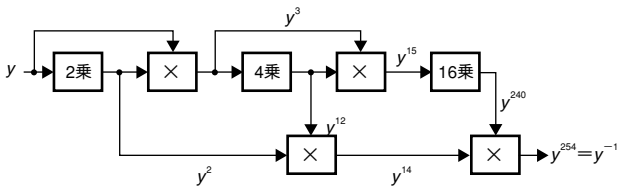
$$A(x)=a_7\cdot x^7+a_6\cdot x^6+a_5\cdot x^5+a_4\cdot x^4+a_3\cdot x^3+a_2\cdot x^2+a_1\cdot x^1+a_0$$
$$B(x)=b_7\cdot x^7+b_6\cdot x^6+b_5\cdot x^5+b_4\cdot x^4+b_3\cdot x^3+b_2\cdot x^2+b_1\cdot x^1+b_0$$

この多項式の乗算を行うと，

$$C(x)=c_{14}\cdot x^{14}+c_{13}\cdot x^{13}+c_{12}\cdot x^{12}+c_{11}\cdot x^{11}+c_{10}\cdot x^{10}+c_9\cdot x^9+c_8\cdot x^8+c_7\cdot x^7+c_6\cdot x^6+c_5\cdot x^5+c_4\cdot x^4+c_3\cdot x^3+c_2\cdot x^2+c_1\cdot x^1+c_0$$

$$c_{14}=a_7\cdot b_7$$
$$c_{13}=a_7\cdot b_6\oplus a_6\cdot b_7$$
$$c_{12}=a_7\cdot b_5\oplus a_6\cdot b_6\oplus a_5\cdot b_7$$
$$c_{11}=a_7\cdot b_4\oplus a_6\cdot b_5\oplus a_5\cdot b_6\oplus a_4\cdot b_7$$
$$c_{10}=a_7\cdot b_3\oplus a_6\cdot b_4\oplus a_5\cdot b_5\oplus a_4\cdot b_6\oplus a_3\cdot b_7$$
$$c_9=a_7\cdot b_2\oplus a_6\cdot b_3\oplus a_5\cdot b_4\oplus a_4\cdot b_5\oplus a_3\cdot b_6\oplus a_2\cdot b_7$$
$$c_8=a_7\cdot b_1\oplus a_6\cdot b_2\oplus a_5\cdot b_3\oplus a_4\cdot b_4\oplus a_3\cdot b_5\oplus a_2\cdot b_6\oplus a_1\cdot b_7$$
$$c_7=a_7\cdot b_0\oplus a_6\cdot b_1\oplus a_5\cdot b_2\oplus a_4\cdot b_3\oplus a_3\cdot b_4\oplus a_2\cdot b_5\oplus a_1\cdot b_6\oplus a_0\cdot b_7$$
$$c_6=a_6\cdot b_0\oplus a_5\cdot b_1\oplus a_4\cdot b_2\oplus a_3\cdot b_3\oplus a_2\cdot b_4\oplus a_1\cdot b_5\oplus a_0\cdot b_6$$
$$c_5=a_5\cdot b_0\oplus a_4\cdot b_1\oplus a_3\cdot b_2\oplus a_2\cdot b_3\oplus a_1\cdot b_4\oplus a_0\cdot b_5$$
$$c_4=a_4\cdot b_0\oplus a_3\cdot b_1\oplus a_2\cdot b_2\oplus a_1\cdot b_3\oplus a_0\cdot b_4$$
$$c_3=a_3\cdot b_0\oplus a_2\cdot b_1\oplus a_1\cdot b_2\oplus a_0\cdot b_3$$
$$c_2=a_2\cdot b_0\oplus a_1\cdot b_1\oplus a_0\cdot b_2$$
$$c_1=a_1\cdot b_0\oplus a_0\cdot b_1$$
$$c_0=a_0\cdot b_0$$

で与えられる14次の多項式となります．各係数は$GF(2)$のガロア体の演算に従うので，乗算はAND演算，加算はXOR演算となります．



〔図3〕伊東・辻井のアルゴリズムによる逆元計算回路

ここで既約多項式＝0とし，$GF(2)$では加算と減算は同じであるので，

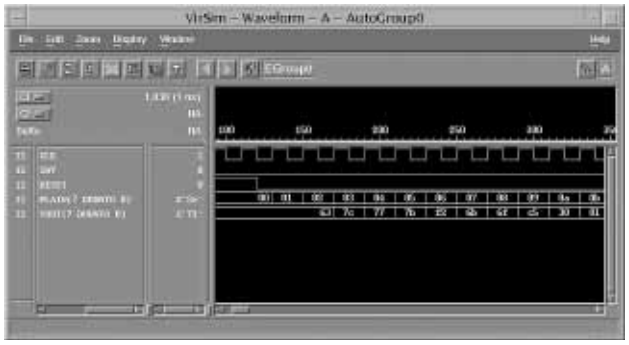
$$m(x)=x^8+x^4+x^3+x+1=0$$
$$x^8=x^4+x^3+x+1$$
$$x^9=x^5+x^4+x^2+x$$
$$x^{10}=x^6+x^5+x^3+x^2$$
$$x^{11}=x^7+x^6+x^4+x^3$$
$$x^{12}=x^8+x^7+x^5+x^4=(x^4+x^3+x+1)+x^7+x^5+x^4$$
$$=x^7+x^5+x^3+x+1$$
$$x^{13}=x^8+x^6+x^4+x^2+x$$
$$=(x^4+x^3+x+1)+x^6+x^4+x^2+x$$
$$=x^6+x^3+x^2+1$$
$$x^{14}=x^7+x^4+x^3+x$$

なる関係式を用いると，$C(x)$は7次以下の多項式$D(x)$に変換することができます．

$$D(x)=d_7\cdot x^7+d_6\cdot x^6+d_5\cdot x^5+d_4\cdot x^4+d_3\cdot x^3+d_2\cdot x^2+d_1\cdot x^1+d_0$$

$$d_7=c_7\oplus c_{11}\oplus c_{12}\oplus c_{14}$$
$$d_6=c_6\oplus c_{10}\oplus c_{11}\oplus c_{13}$$
$$d_5=c_5\oplus c_9\oplus c_{10}\oplus c_{12}$$
$$d_4=c_4\oplus c_8\oplus c_9\oplus c_{11}\oplus c_{14}$$
$$d_3=c_3\oplus c_8\oplus c_{10}\oplus c_{11}\oplus c_{12}\oplus c_{13}\oplus c_{14}$$
$$d_2=c_2\oplus c_9\oplus c_{10}\oplus c_{13}$$
$$d_1=c_1\oplus c_8\oplus c_9\oplus c_{12}\oplus c_{14}$$
$$d_0=c_0\oplus c_8\oplus c_{12}\oplus c_{13}$$

以上の計算から，二つの8ビットの数$A=(a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0)$，$B=(b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)$の乗算結果$D=(d_7, d_6, d_5, d_4, d_3, d_2, d_1, d_0)$が得られます．すなわち，実際の実装では多数のANDとXORを用いることにより乗算器を実現することができます．



〔図4〕シミュレーション結果

課　題

●レベル1：基本課題

基本課題で設計する回路は，**表2**の信号ビット幅とします．送信器SENDERのVHDLコード(sender.vhd)を**リスト1**に，ROMを用いて実装したSubBytes変換回路(SubBytesのみで逆SubBytesはできない)のVHDLコード(subbytes.vhd)を**リスト2**に，全体シミュレーションのテストベンチ(test_subbytes.vhd)を**リスト3**に示します(これらのファイルは本コンテストのホームページhttp://www.cqpub.co.jp/dwm/contest/からダウンロード可能)．必要に応じ，修正して使用してください．

サンプル・コードによるシミュレーション結果を**図4**に示します．SENDERが8ビットのPLAIN信号を出力し，その2サイクル遅れで，SUBBYTESがSubBytes変換値を出力しています．SENDERが毎サイクル8ビットのPLAIN信号を出力していますが，回路削減のためにガロア体の乗算を複数サイクルかけて実行することも可能であり，必要に応じてPLAIN信号を数サイクルに1回出力するなど，自由に変更を行ってください．

●レベル2：自由課題

レベル2では詳細な仕様は自由とします．SubBytes変換と逆SubBytes変換ができれば，どのようなアーキテクチャ，タイミング動作でもOKとします．

速度の測定単位

筆者の大学では，論理合成ツールとして米国Synopsys社のDesign Compilerを使用しますが，このツールはだれもが使えるわけではありません．そこで，使用する論理合

〔表2〕レベル1で使用する信号とビット幅

SubBytes			
信号名	入出力	ビット幅	説　明
CLK	IN	1	クロック入力
RESET	IN	1	'1' でリセット
XIN	IN	8	入力データ
INV	IN	1	'0' でSubBytes変換，'0' で逆SubBytes変換
YOUT	OUT	8	変換されたデータ出力

成ツールで50入力のXOR回路を合成していただき，その1段当たりの遅延時間を単位時間として速度の単位とします．

50入力XOR回路のVHDLソース・コードを**リスト4**に示します（このファイルも本コンテストのホームページからダウンロード可能）．Design Compilerでは6段のXOR回路が合成されます．クリティカル・パス遅延はreport_timingコマンドにより7.17であることがわかります．そこで，7.17/6＝1.195を単位（UNIT）とします．例えば，ある遅延が20ならば，20/1.195＝17.74UNIT遅延ということになります．

ちなみに面積はreport_areaコマンドのtotal cell areaになります．

提出レポートについて

レポートには以下の内容を含めてください．また，ページ数は少なめに，コンパクトにまとめてください．

＜表紙＞

1）代表者の氏名，チーム名，社会人/大学院修士/大学学部生/高専生の区別
2）共同設計者全員の名まえ（最高3名まで），会社/学校名（学籍番号，学年），住所，電話，E-mailなどの連絡先
3）取り組んだ課題（レベル1/レベル2）

＜内容＞

1）設計した回路ブロックの構成説明（ブロック図と説明）
2）設計した回路ブロックの動作説明
（動作波形図やパイプライン動作などの説明）
3）くふうした点，オリジナリティを出した点
（アピールが重要！）
4）クリティカル・パスのスピード，論理合成後の回路規模
5）VHDLもしくはVerilog HDLのコード
6）正常動作しているVHDL/Verilog HDLシミュレーションの波形
7）そのほか自由意見など

レポートはPDFファイルを推奨します．PDFファイルを作成できない場合はご相談ください．

社会人か学生かによって評価の方法が異なります．それにともない，レポートの送付先と締め切り日も異なります．

社会人：contest.dwm@cqpub.co.jp
締め切りは2004年1月31日（金）必着
学生：wada@ie.u-ryukyu.ac.jp
締め切りは2004年2月6日（金）必着

審査のポイント

速度，回路規模だけでなく，アーキテクチャのユニークさ，アイデア，おもしろさを十分考慮して審査します（ちゃんとアピールしてね！）．

社会人は，Design Wave設計コンテスト審査委員会が審査を行います．学生は，琉球大学で1次審査を行い，最終審査に残ったチームは，2004年3月5日に沖縄産業支援センターで開催予定の発表会に招待され，その会場で最終審査を行います．大学院修士，学部生，高専生のレベルに応じて審査します．

仕様書に従ってまじめに作るのもけっこうですが，おもしろいアイデアを歓迎します．他人と違ったことをしよう！ 仕様の部分変更など，柔軟に受け付けます．

ENJOY HDL！ 沖縄で会おう！

＊　＊　＊

本設計コンテスト学生部門は，主催：琉球大学工学部情報工学科，共催：沖縄産業振興センター，協賛：ソニーLSIデザインにより実施されています．

参考文献

1）http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
2）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年5月．
3）Chin-Pin Su et. al，“A Highly Efficient AES Cipher Chip”，*ASP-DAC* 2003，pp.561-562，Jan. 2003．
4）森岡澄夫，佐藤証，高野光司，宗藤誠治，「GF（((2^2) ^2) ^2）上の演算を用いたAESのS-Box構成法」，『第63回情報処理学会全国大会』，3G-04，2001年．
5）Satoh, A., Morioka, S., Takano, K. and Munetoh, S.，“A Compact Rijndael Hardware Architecture with S-Box Optimization”，*Advances in Cryptology - ASIACRYPT* 2001，LNCS Vol.2248，pp.239-254，2001．
6）森岡澄夫，「エラー訂正や暗号処理で使われる演算回路を極める」，『Design Wave Magazine』，pp.57-67，2003年7月号．
7）琉球大学学生LSIデザインコンテストホームページ，http://www.ie.u-ryukyu.ac.jp/~wada/design04/contest2004.html．

わだ・ともひさ
琉球大学 工学部 情報工学科

〔リスト1〕SENDERのVHDLソース・コード

```
-- Univ. of the Ryukyus LSI design contest 2004
-- SubBytes Transform Circuit for AES Cipher
-- file: sender.vhd
-- Sender generates 8 bit integer from 0 to 255
-- Tom Wada 2003/September/15

library IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;

entity SENDER is
    port ( CLK    : in  std_logic;
           RESET  : in  std_logic;
           PLAIN  : out unsigned (7 downto 0) );
end entity SENDER;

architecture RTL of SENDER is
-- signal define
signal  count : unsigned (7 downto 0); -- 8 bit counter

begin

-----------------------------------------
--  8 bit counter
-----------------------------------------
COUNTER: process(CLK)
begin
    if rising_edge(CLK) then
        if (RESET='1') then
            count <= "00000000";
        else
            count <= count + 1;
        end if;
    end if;
end process COUNTER;

-----------------------------------------
--  OUTPUT GEN
-----------------------------------------
PLAIN <= count;

end architecture RTL;
```

〔リスト3〕全体シミュレーションのテストベンチ

```
-- Univ. of the Ryukyus LSI design contest 2004
-- SubBytes Transform Circuit for AES Cipher
-- file: test_subbytes.vhd
-- TESTBENCH
-- Tom Wada 2003/September/15

library IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;

entity TEST_SUBBYTES is
end entity TEST_SUBBYTES;

architecture TESTBENCH of TEST_SUBBYTES is
  -- sender
  component SENDER
      port ( CLK    : in  std_logic;
             RESET  : in  std_logic;
             PLAIN  : out unsigned (7 downto 0) );
  end component SENDER;
  -- subbytes
  component SUBBYTES
      port ( CLK    : in  std_logic;
             RESET  : in  std_logic;
             XIN    : in  unsigned(7 downto 0);
             INV    : in  std_logic;
             YOUT   : out unsigned(7 downto 0) );
  end component SUBBYTES;

-- system clock
signal  CLK  : std_logic  :=  '0' ;
-- system reset
signal RESET : std_logic  :=  '1' ;
-- cycle count
signal cycle : integer :=0;
-- wires on the board
signal PLAIN  : unsigned (7 downto 0);
signal YOUT   : unsigned (7 downto 0);
signal INV    : std_logic  :=  '0' ;

begin

-- clock generator
    CLOCK_GEN: process
    begin
        if (cycle < 1000) then
            cycle <= cycle + 1;
            wait for 10 ns;
            CLK <= not CLK;
        else wait;
        end if;
    end process CLOCK_GEN;

-- reset sequence
    RESET_GEN: process
    begin
        LOOP1: for N in 0 to 5 loop
           wait until falling_edge(CLK);
        end loop LOOP1;
        RESET <  '0' ;
    end process RESET_GEN;

-- sender instance
    I_SENDER: SENDER port map(CLK,RESET,PLAIN);

-- subbytes instance
    I_SUBBYTES: SUBBYTES port map(CLK,RESET,PLAIN,INV,YOUT);

end architecture TESTBENCH;


configuration CFG_SUBBYTES of TEST_SUBBYTES is
    for TESTBENCH
    end for;
end configuration CFG_SUBBYTES;
```

〔リスト2〕SubBytes変換回路のVHDLソース・コード

```
-- Univ. of the Ryukyus LSI design contest 2004
-- SubBytes Transform Circuit for AES Cipher
-- file: SubBytes.vhd
-- combinational logic of SubBytes transform
-- Tom Wada 2003/September/15

library IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;

entity SUBBYTES is
    port (CLK    : in  std_logic;
          RESET  : in  std_logic;
          XIN    : in  unsigned( 7 downto 0 );
          INV    : in  std_logic;  -- NOT SUPPORTED FOR THIS
          YOUT   : out unsigned( 7 downto 0) );
end entity SUBBYTES;

architecture RTL of SUBBYTES is

-- type definition for 256 word x 8 bit Inverse ROM
type vectype is array (0 to 255) of unsigned (7 downto 0);
-- ROM invrom
constant invrom  : vectype := (
0 =>  "00000000" ,
1 =>  "00000001" ,
2 =>  "10001101" ,
3 =>  "11110110" ,
4 =>  "11001011" ,
5 =>  "01010010" ,
6 =>  "01111011" ,
7 =>  "11010001" ,
8 =>  "11101000" ,
9 =>  "01001111" ,
10 =>  "00101001" ,
11 =>  "11000000" ,
12 =>  "10110000" ,
13 =>  "11100001" ,
14 =>  "11100101" ,
15 =>  "11000111" ,
16 =>  "01110100" ,
17 =>  "10110100" ,
18 =>  "10101010" ,
19 =>  "01001011" ,
20 =>  "10011001" ,
21 =>  "00101011" ,
22 =>  "01100000" ,
23 =>  "01011111" ,
24 =>  "01011000" ,
25 =>  "00111111" ,
26 =>  "11111101" ,
27 =>  "11001100" ,
28 =>  "11111111" ,
29 =>  "01000000" ,
30 =>  "11101110" ,
31 =>  "10110010" ,
32 =>  "00111010" ,
33 =>  "01101110" ,
34 =>  "01011010" ,
35 =>  "11110001" ,
36 =>  "01010101" ,
37 =>  "01001101" ,
38 =>  "10101000" ,
39 =>  "11001001" ,
40 =>  "11000001" ,
41 =>  "00001010" ,
42 =>  "10011000" ,
43 =>  "00010101" ,
44 =>  "00110000" ,
45 =>  "01000100" ,
46 =>  "10100010" ,
47 =>  "11000010" ,
48 =>  "00101100" ,
49 =>  "01000101" ,
50 =>  "10010010" ,
51 =>  "01101100" ,
52 =>  "11110011" ,
53 =>  "00111001" ,
54 =>  "01100110" ,
55 =>  "01000010" ,
56 =>  "11110010" ,
57 =>  "00110101" ,
58 =>  "00100000" ,
59 =>  "01101111" ,
60 =>  "01110111" ,
61 =>  "10111011" ,
62 =>  "01011001" ,
63 =>  "00011001" ,
64 =>  "00011101" ,
65 =>  "11111110" ,
66 =>  "00110111" ,
67 =>  "01100111" ,
68 =>  "00101101" ,
69 =>  "00110001" ,
70 =>  "11110101" ,
71 =>  "01101001" ,
72 =>  "10100111" ,
73 =>  "01100100" ,
74 =>  "10101011" ,
75 =>  "00010011" ,
76 =>  "01010100" ,
77 =>  "00100101" ,
78 =>  "11101001" ,
79 =>  "00001001" ,
80 =>  "11101101" ,
81 =>  "01011100" ,
82 =>  "00000101" ,
83 =>  "11001010" ,
84 =>  "01001100" ,
85 =>  "00100100" ,
86 =>  "10000111" ,
87 =>  "10111111" ,
88 =>  "00011000" ,
89 =>  "00111110" ,
90 =>  "00100010" ,
91 =>  "11110000" ,
92 =>  "01010001" ,
93 =>  "11101100" ,
94 =>  "01100001" ,
95 =>  "00010111" ,
96 =>  "00010110" ,
97 =>  "01011110" ,
98 =>  "10101111" ,
99 =>  "11010011" ,
100 =>  "01001001" ,
101 =>  "10100110" ,
102 =>  "00110110" ,
103 =>  "01000011" ,
104 =>  "11110100" ,
105 =>  "01000111" ,
106 =>  "10010001" ,
107 =>  "11011111" ,
108 =>  "00110011" ,
109 =>  "10010011" ,
110 =>  "00100001" ,
111 =>  "00111011" ,
112 =>  "01111001" ,
113 =>  "10110111" ,
114 =>  "10010111" ,
115 =>  "10000101" ,
116 =>  "00010000" ,
117 =>  "10110101" ,
118 =>  "10111010" ,
119 =>  "00111100" ,
120 =>  "10110110" ,
121 =>  "01110000" ,
122 =>  "11010000" ,
123 =>  "00000110" ,
124 =>  "10100001" ,
125 =>  "11111010" ,
126 =>  "10000001" ,
127 =>  "10000010" ,
128 =>  "10000011" ,
129 =>  "01111110" ,
130 =>  "01111111" ,
131 =>  "10000000" ,
132 =>  "10010110" ,
133 =>  "01110011" ,
134 =>  "10111110" ,
135 =>  "01010110" ,
136 =>  "10011011" ,
137 =>  "10011110" ,
138 =>  "10010101" ,
139 =>  "11011001" ,
140 =>  "11110111" ,
141 =>  "00000010" ,
142 =>  "10111001" ,
143 =>  "10100100" ,
144 =>  "11011110" ,
145 =>  "01101010" ,
146 =>  "00110010" ,
147 =>  "01101101" ,
148 =>  "11011000" ,
149 =>  "10001010" ,
150 =>  "10000100" ,
151 =>  "01110010" ,
152 =>  "00101010" ,
153 =>  "00010100" ,
154 =>  "10011111" ,
155 =>  "10001000" ,
156 =>  "11111001" ,
157 =>  "11011100" ,
158 =>  "10001001" ,
159 =>  "10011010" ,
160 =>  "11111011" ,
161 =>  "01111100" ,
162 =>  "00101110" ,
163 =>  "11000011" ,
164 =>  "10001111" ,
165 =>  "10111000" ,
166 =>  "01100101" ,
167 =>  "01001000" ,
168 =>  "00100110" ,
169 =>  "11001000" ,
170 =>  "00010010" ,
171 =>  "01001010" ,
172 =>  "11001110" ,
173 =>  "11100111" ,
174 =>  "11010010" ,
175 =>  "01100010" ,
176 =>  "00001100" ,
177 =>  "11100000" ,
178 =>  "00011111" ,
179 =>  "11101111" ,
180 =>  "00010001" ,
181 =>  "01110101" ,
182 =>  "01111000" ,
183 =>  "01110001" ,
184 =>  "10100101" ,
185 =>  "10001110" ,
186 =>  "01110110" ,
187 =>  "00111101" ,
188 =>  "10111101" ,
189 =>  "10111100" ,
190 =>  "10000110" ,
191 =>  "01010111" ,
192 =>  "00001011" ,
193 =>  "00101000" ,
194 =>  "00101111" ,
195 =>  "10100011" ,
196 =>  "11011010" ,
197 =>  "11010100" ,
198 =>  "11100100" ,
199 =>  "00001111" ,
200 =>  "10101001" ,
201 =>  "00100111" ,
202 =>  "01010011" ,
203 =>  "00000100" ,
204 =>  "00011011" ,
```

〔リスト2〕SubBytes変換回路のVHDLソース・コード（つづき）

```
205 =>  "11111100",
206 =>  "10101100",
207 =>  "11100110",
208 =>  "01111010",
209 =>  "00000111",
210 =>  "10101110",
211 =>  "01100011",
212 =>  "11000101",
213 =>  "11011011",
214 =>  "11100010",
215 =>  "11101010",
216 =>  "10010100",
217 =>  "10001011",
218 =>  "11000100",
219 =>  "11010101",
220 =>  "10011101",
221 =>  "11111000",
222 =>  "10010000",
223 =>  "01101011",
224 =>  "10110001",
225 =>  "00001101",
226 =>  "11010110",
227 =>  "11101011",
228 =>  "11000110",
229 =>  "00001110",
230 =>  "11001111",
231 =>  "10101101",
232 =>  "00001000",
233 =>  "01001110",
234 =>  "11010111",
235 =>  "11100011",
236 =>  "01011101",
237 =>  "01010000",
238 =>  "00011110",
239 =>  "10110011",
240 =>  "01011011",
241 =>  "00100011",
242 =>  "00111000",
243 =>  "00110100",
244 =>  "01101000",
245 =>  "01000110",
246 =>  "00000011",
247 =>  "10001100",
248 =>  "11011101",
249 =>  "10011100",
250 =>  "01111101",
251 =>  "10100000",
252 =>  "11001101",
253 =>  "00011010",
254 =>  "01000001",
255 =>  "00011100"
);

-- signal defininition
signal invout  : unsigned (7 downto 0);  -- Inverse output
signal affine  : unsigned (7 downto 0);
                                -- affin transform output

begin
----------------------------------
-- INVERSE OUTPUT
----------------------------------
INVERSE: process(CLK)
begin
        if rising_edge(CLK) then
            if (RESET='1') then
                invout <= "00000000";
            else
                invout <= invrom(TO_INTEGER(XIN));
            end if;
        end if;
    end process INVERSE;
    ----------------------------------
    -- AFFINE TRANSFORM
    ----------------------------------
    AFFINE_TRAN: process(CLK)
    begin
      if rising_edge(CLK) then
         affine(7) <= invout(7) xor invout(6) xor invout(5) xor invout(4) xor invout(3) xor '0';
         affine(6) <= invout(6) xor invout(5) xor invout(4) xor invout(3) xor invout(2) xor '1';
         affine(5) <= invout(5) xor invout(4) xor invout(3) xor invout(2) xor invout(1) xor '1';
         affine(4) <= invout(4) xor invout(3) xor invout(2) xor invout(1) xor invout(0) xor '0';
         affine(3) <= invout(7) xor invout(3) xor invout(2) xor invout(1) xor invout(0) xor '0';
         affine(2) <= invout(7) xor invout(6) xor invout(2) xor invout(1) xor invout(0) xor '0';
         affine(1) <= invout(7) xor invout(6) xor invout(5) xor invout(1) xor invout(0) xor '1';
         affine(0) <= invout(7) xor invout(6) xor invout(5) xor invout(4) xor invout(0) xor '1';
      end if;
    end process AFFINE_TRAN;
    ----------------------------------
    -- OUTPUT GENERATION
    ----------------------------------
    YOUT <= affine;

    end architecture RTL;
```

〔リスト4〕50入力XOR回路のVHDLソース・コード

```
library  IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;

entity PARITY is
    port ( A : in  unsigned(49 downto 0);
           Y : out std_logic );
end PARITY;

architecture RTL of PARITY is
begin

    process(A)
    variable TMP : std_logic;
    begin
        TMP := '0';
        for i in 0 to 49 loop
            TMP := TMP xor A(i);
        end loop;

        Y <= TMP;
    end process;
end RTL;
```