A-09

AES 暗号化回路の設計・実装と高速化の検討

橋爪 啓介†　　HoangAnhTuan†　　山崎勝弘†

†立命館大学大学院　理工学研究科

1　はじめに

AES は国立標準技術研究所(NIST)が政府標準暗号として 2001 年承認した共通鍵暗号方式である。Rinjndeal ではデータをブロックごとに分けて暗号化する。その際、ブロック長を 128 ビット、192 ビット、256 ビットから指定できる。さらに暗号化のときに使う鍵の長さも 128 ビット、192 ビット、256 ビットから指定できる。我々は、AES 暗号を対象として、ソフトマクロ CPU を用いて、ハードソフト最適分割の研究を行ってきた[1][2]。本稿では、AES 暗号化回路の高速化を目標とし、Verilog-HDL を用いて AES 暗号化回路をハードウェア設計し、Spartan3　FPGA ボード上へ実装した。回路規模と各モジュールの遅延時間を示すと共に、今後の高速化の検討をする。

2　AES 暗号アルゴリズム

図 1 は AES の暗号化の流れを示しており、128 ビットのデータに対応している。AES の入力データはバイトの行列になり、状態の行列に対応して処理する。ラウンドは 1 から 10 まであり ラウンド 1 からラウンド 9 までは同様に SubBytes 処理、ShiftRows 処理、MixColumns 処理、AddRoundKey 処理の 4 つの状態の行列で処理を行い、ラウンド 10 で SubBytes 処理、ShiftRows 処理、MixColumns 処理を行って暗号文を出力する。

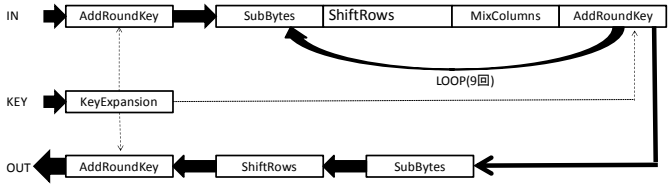

図 1：AES 暗号化フロー

2.1　KeyExpansion

鍵を拡張する KeyExpansion は処理の手順が一番煩雑である。本研究では鍵長を 128 ビットに限定しているので、それを中心に話を進める。鍵データの扱い方は 1 バイトを 1 ブロックとして列単位で見ている。ラウンドごとに　128 ビット拡張するので、128 ビットごとに鍵を RoundKey として分ける。図 2 に鍵拡張の例を示す。

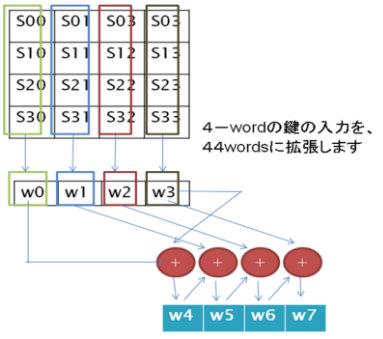

図 2：KeyExpansion 処理

2.2　AddroundKey

図 3 は AddRounKey 処理を示しており、鍵拡張部で拡張された鍵の中から 1 ラウンド分の鍵を持ってきて、入力データと排他的論理和(XOR)をとって出力する。

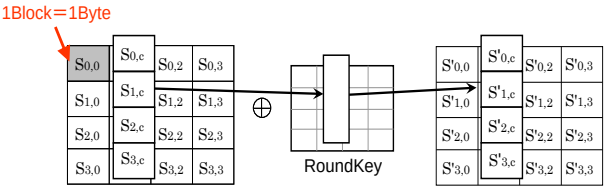

図 3：AddRounKey 処理

2.3　SubBytes

図4はSubBytes処理を示しており、入力された被変換データをS-boxというテーブルを参照して変換する。この処理は列単位で行う。Sboxテーブルは各ブロックにおいて、値を上位4ビットと下位4ビットに分解し、Sbox[上位4ビット ， 下位4ビット]としてテーブルとして参照する。

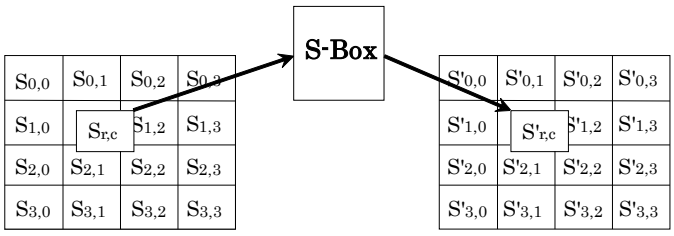

図4：SubBytes処理

2.4　ShiftRows

図 5 は ShiftRows 処理を示しており、16byte の入力データを 4byte 単位で区切った際の 4 つの行の各行毎に、0 から 3byte 単位での巡回シフト演算を行う。

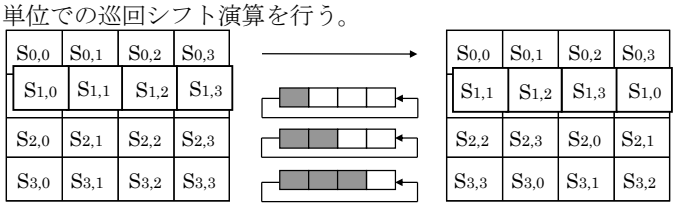

図 5：ShiftRows 処理

2.5　MixColumuns

図 6 は MixColumns 処理を示しており、GF 理論に基づいた行列演算を行う。まず、MixColumns 変換は次の行列をかけて計算することで結果を導ける。式内の右辺にある列ベクトルは被変換データで左辺にある列ベクトルは変換後データの列（4 ブロック）である。

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

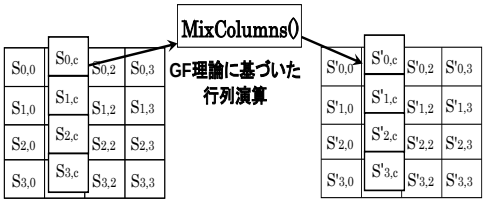

図 6：MixColumns 処理

2.5.1　バイト演算

上記の行列を用いて乗算を行えば一列分の変換データを導出できるが、そのための積和演算が有限体理論に準じているので注意する必要がある。有限体理論での積和演算は以下の項目についてのみ考慮すればよい。

(1)フィールドでの加算は排他的論理和である。

(2)$S_{2,0}$×01, $S_{3,0}$ ×01 の乗算は被乗数の値のままである。

(3)$S_{0,0}$×02 の乗算は、乗数 02 が 08 未満であるので $S_{0,0}$ を 1 ビット左シフトした値になる。

(4) $S_{1,0}$×03 の乗算は、(被乗数×01+被乗数×02)に分配できる。

次に MixColumns 処理で使われる GF 理論について説明する。GF(2)は 1 ビットの演算で、加算を排他論理和 (XOR) で行う。これによって GF(2)は 0～1 の集合になる。GF(2^8)は GF(2)を拡張したもので、ほぼ 8 ビット(1 バイト)のビット演算である。ただし、適切な既約多項式 p(x)を決めてあって、ビットがあふれてしまったときの処理を定めてある。GF(2^8)の規約多項式 p(x)は p（x）＝$x^8+x^4+x^3+x^2+1$ である。ガロア体の定義は p(x) = 0 なので、以下の関係が成り立つ。

p（x）＝$x^8+x^4+x^3+x^2+1=0$ より

$x^8=-x^4-x^3-x^2-1$　　$1+1=0$　より

$1=-1$　よって　$x^8=x^4+x^3+x^2+1$

つまり演算によってビットがあふれ、ビット 8 が 1 になってしまったとき、その値は下位の 8 ビット（$x^4 + x^3 + x^2 + 1$）に置き換えられることを意味する。そして下位 8 ビットとの加算によって 8 ビット幅の値を得られる。ここでの加算は排他的論理和で行うので、演算結果が 8 ビット幅を超えることはない。

2.5.2　ビット演算

MixColumns 処理をバイト演算として設計すると回路規模と実行時間が増大する。これらを小さくするためにビット演算を使用すると fx×02 と fx×03 は

fx×02 = $\{a_7\cdots\cdots a_0\}\times\{00000010\} = \{d_7\cdots\cdots d_0\}$

$d_7 = a_6$　$d_6 = a_5$　$d_5 = a_4$　$d_4 = a_3+a_7$　$d_3 = a_2+a_7$　$d_2 = a_1$　$d_1 = a_0 + a_7$　$d_0 = a_7$

fx×03 も同様に

$d_7 = a_6+a_7$　$d_6 = a_5+a_6$　$d_5 = a_4+a_5$　$d_4 = a_3+a_4+a_7$　$d_3 = a_2+a_3+a_7$　$d_2 = a_1+a_2$

$d_1 = a_0+a_1+a_7$　$d_0 = a_0+a_7$ となる。つまりこれらの考えを用いれば MixColumns 処理をハードウェア設計するとき排他的論理和のみで設計できるので回路規模が縮小できると考えられる。

3　AES 暗号化回路の設計・実装と評価

まず Xilinx 社の ISE を使用して AES 暗号化回路を Verilog-HDL で設計した。次にそれぞれの処理を論理合成し動作を確認した。さらに top_module を設計して論理合成し、完了すればSPARTAN-3　スターターボード　XC3S200 に実装した。

表 1：各モジュールの回路規模と遅延時間

モジュール名	回路規模(Slices)	最大遅延時間(ns)
AddRoundKey	74	7.76
KeyExpansions	757	53.18
SubBytes	1024	10.33
ShiftRows	14	4.10
MixColumns(ビット演算)	133	9.27
MixColumns(バイト演算)	133	9.27

表 1 に各モジュールの回路規模と最大遅延時間を示す。MixColumns 処理では、ビット演算が回路規模、実行速度ともにバイト演算より小さいと考えていたが、実際、バイト演算と回路規模、実行速度ともに変わらなかった。また全てのモジュールのなかで、ShiftRows 処理の動作周波数が 1 番大きかったので実行速度も 1 番速いことが確認できた。SubBytes 処理の回路規模が 1 番大きい理由としては図 4 のように 1byte ずつ処理をしていくからであるのと S-BOX が回路規模のなかに含まれているからだと思われる。

4　高速化の検討

今回の研究では各モジュールを 1 クロックで実行させる回路の設計を行ったが、今後は AddRounKey 処理を 1 クロック、MixColumns 処理を 3 クロックで実行する回路の設計を行ってシステムの評価をしたいと考えている。次に SubBytes 処理の回路規模が各モジュールのなかで一番大きいと確認できたので、Sbox のデータを BlockRAM に入れて処理をする回路を設計して、回路規模、実行速度を小さくしていきたいと考えている。さらに AES 暗号化フローの 1 ラウンドを複数クロックで完了する場合と、1 ラウンドを 1 クロックで完了する場合とで回路規模と遅延速度を比較する。また AES 暗号化フローをパイプライン化して AES 暗号化回路の実行速度が速くなるかを検討していきたいと考えている。

5　おわりに

本稿では、インターネット上での通信に幅広く用いられる AES 暗号化についてハードウェア設計を行い、FPGA ボード上へ実装した。回路規模と最大遅延時間を測定して今後の高速化の方法について考察した。今後は AES 暗号化回路のパイプライン化による高速化を目指した回路を設計する予定である。

[参考文献]

[1] 梅原直人，古川達久，的場督永，山崎勝弘，小柳滋，ソフト・マクロCPU を用いた回路設計とハードウェア/ソフトウェア最適分割法の検討，情処関西支部大会，C-06，2005

[2]梅原直人，古川達久，的場督永，山崎勝弘，小柳滋，ハード/ソフト最適分割を考慮したAES暗号システムとJPEG エンコーダの設計と検証，第 4 回情報技術フォーラム（FIT 2005），C-034，2005