3K-8

ガロア体上の乗算器モジュールジェネレータの構築

岡本 広太郎　　本間 尚文　　青木 孝文
東北大学大学院情報科学研究科

1　まえがき

近年，携帯電話・RFID などの組込み機器には，高信頼・高安全通信のために誤り訂正回路や暗号化/復号回路が搭載されている．しかし，これらの回路を構成するガロア体上の算術演算回路の設計は困難な作業となっている．現在一般に用いられるハードウェア記述言語 (HDL) はガロア体上の演算を扱うための高水準なデータ構造や記法を持たないため，その設計時には AND や XOR などの低水準な論理式による記述を用いなければならない．また，一般に多入力多出力な当該回路では，その機能を計算機シミュレーションで検証するために膨大な時間が必要となり，場合によっては完全な検証が不可能となる．上記の問題に対して，本稿では，機能が完全に検証されたガロア体上の算術演算回路を短時間で生成可能なシステム (Galois-Field Arithmetic Module Generator: GF-AMG) を提案する．GF-AMG は，その内部にガロア体算術演算回路グラフ (Galois-Field Arithmetic Circuit Graph: GF-ACG) に基づく設計手法 [1],[2] を用いて，ガロア体上の乗算器の最適な構成の 1 つである Mastrovito 乗算器 [3] を自動生成する．

2　GF-ACG による Mastrovito 乗算器の記述

GF-ACG は，部分回路を表す演算ノードの集合 $\boldsymbol{N}$ と演算ノード間の接続を表す有向辺の集合 $\boldsymbol{E}$ により，$(\boldsymbol{N}, \boldsymbol{E})$ と表される．各演算ノード n ($\in \boldsymbol{N}$) は演算機能を表す機能表明の集合と内部構造を表す GF-ACG により表される．GF-ACG では，このように算術演算回路を階層的に表現することで設計の容易性を高めるとともに数式処理による効率的な検証を実現する．

図 1 にガロア体 $GF(2^2)$ 上の Mastrovito 乗算器（既約多項式 $IP : \beta^2+\beta+1$）の GF-ACG を示す．このように Mastrovito 乗算器は GF-ACG により 4 階層で記述することができる．図 1 (a)~(c) の各演算ノード（白色の部分）は，図 1 (b)~(d) の GF-ACG（灰色の部分）にそれぞれ対応する．また，表 1 に GF-ACG の各演算ノードの機能表明をそれぞれ示す．Mastrovito 乗算器の構造は，入力 a から行列 Z を生成する行列生成部と，Z と入力 b の行列の積を算出する行列演算部に分けられる．

このため，乗算部の内部構造は，行列生成部と行列演算部を実現する演算ノードにより構成される．また，行列生成部の内部構造は，Z の列ベクトル Z_i を生成する演算ノード MG により実現され，行列演算部の内部構造は，Z_i と b_i の積を算出する演算ノード MO と 2 入力の和を算出する演算ノード GFA により実現される．ここで，既約多項式 IP の次数（ガロア体の拡大次数）を m, 項数を t とすると，演算ノード MG，MO, GFA の数はそれぞれ $m-1$，m，$m-1$ となる．このとき，$m-1$ 個の GFA は木構造状に組み合わされる．また，MG，MO, GFA の内部構造はそれぞれ XOR，AND，XOR で実現され，その数はそれぞれ $t-2$，m，m となる．

このように GF-ACG で Mastrovito 乗算器を記述することにより，その機能を数式処理により高速かつ形式的に検証することが可能となる．

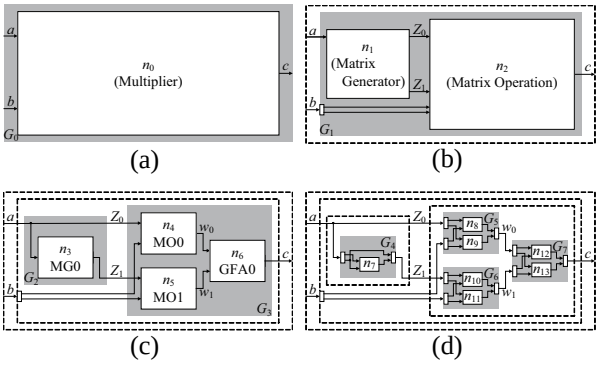

図 1: $GF(2^2)$ 上の Mastrovito 乗算器の GF-ACG

表 1: 図 1 の GF-ACG における各演算ノードの機能表明

[Multiplier] $n_0 = (\{c = a \times b\}, G_1)$
[Matrix Generator] $n_1 = (\{Z_0 = a \times \beta^0, Z_1 = a \times \beta^1, G_2)$
[MG0] $n_3 = (\{Z_1 = Z_0 \times \beta\}, G_4)$
$n_7 = (\{Z_{1,1}^{(p)} = Z_{0,0}^{(p)} + Z_{0,1}^{(p)}\}, nil)$
[Matrix Operation] $n_2 = (\{c \times \beta^1 = \sum_{i=0}^{1} Z_i \times b_i^{(e)} \times \beta^{1-i}, G_3)$
[MO0] $n_4 = (\{w_0 \times \beta^0 = Z_0 \times b_0^{(e)}\}, G_5)$
$n_8 = (\{w_{0,0}^{(p)} = Z_{0,0}^{(p)} \times b_{0,0}^{(p)}\}, nil)$
$n_9 = (\{w_{0,1}^{(p)} = Z_{0,1}^{(p)} \times b_{0,0}^{(p)}\}, nil)$
[MO1] $n_5 = (\{w_1 \times \beta^1 = Z_1 \times b_1^{(e)}\}, G_6)$
$n_{10} = (\{w_{1,0}^{(p)} = Z_{1,0}^{(p)} \times b_{1,1}^{(p)}\}, nil)$
$n_{11} = (\{w_{1,1}^{(p)} = Z_{1,1}^{(p)} \times b_{1,1}^{(p)}\}, nil)$
[GFA0] $n_6 = (\{c = w_0 + w_1\}, G_7)$
$n_{12} = (\{c_0^{(p)} = w_{0,0}^{(p)} + w_{1,0}^{(p)}\}, nil)$
$n_{13} = (\{c_1^{(p)} = w_{0,1}^{(p)} + w_{1,1}^{(p)}\}, nil)$

Development of Galois-field multiplier module generator
Kotaro OKAMOTO　Nafumi HOMMA　Takafumi AOKI
Graduate School of Information Sciences, Tohoku University

3 モジュールジェネレータの構成

GF-AMG では，GF-ACG とその各階層間の等価性判定に基づく形式的検証を用いた設計手法により機能が完全保証された乗算器のみを生成する．図 2 に GF-AMG の概要を示す．GF-AMG では，まず，設計仕様として任意の既約多項式（次数 2〜256）を入力すると，それに対応する GF-ACG を合成する．次に，合成された GF-ACG を形式的手法により検証する．検証手法については参考文献 [2] を参照されたい．その後，機能が保証された GF-ACG を HDL 記述に変換したコードを出力する．

図 3 に Mastrovito 乗算器の GF-ACG 合成アルゴリズムの概略を示す．図 3 のアルゴリズムは，既約多項式 IP を入力とし，GF-ACG G_0 を出力とする．まず，IP から次数 m，項数 t を求め，G_0 を合成する（1〜3 行目）．次に，G_0 を構成する演算ノード n_0 の内部構造 G_1 を合成する（4 行目）．この関数には，第 1 引数として n_0 を，第 2 引数として G_1 を構成する演算ノードの数を与える．その後，G_1 を構成する演算ノード n_1, n_2 に対しても同様の操作を行い，最下層までこれを繰り返す（5〜15 行目）．

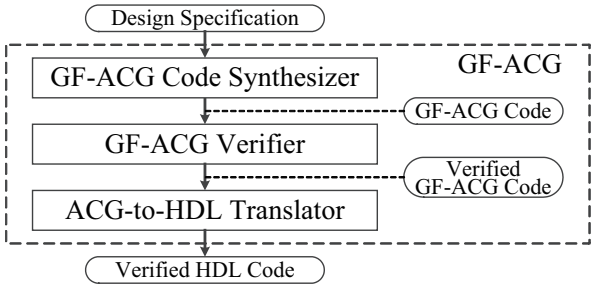

図 2: GF-AMG の概要

Input:	irreducible polynomial IP
Output:	GF-ACG $G_0 : (\boldsymbol{N}_0, \boldsymbol{E}_0)$

1: m := Degree(IP);
2: t := TermNumber(IP);
3: G_0 := $(\{n_0\}, \boldsymbol{E}_0)$;
4: G_1 := MultiplierStructure($n_0 \in G_0, 2$);
5: G_2 := MatrixGeneratorStructure($n_1 \in G_1, m-1$);
6: G_3 := MatrixOperationStructure($n_2 \in G_1, 2m-1$);
7: **for** $i = 0$ **to** $m-2$
8: G_{4+i} := MGStructure($n_{3+i} \in G_2, t-2$);
9: **end for**
10: **for** $i = 0$ **to** $m-1$
11: G_{m+3+i} := MOStructure($n_{m+2+i} \in G_3, m$);
12: **end for**
13: **for** $i = 0$ **to** $m-2$
14: G_{2m+3+i} := GFAStructure($n_{2m+2+i} \in G_3, m$);
15: **end for**
16: **return** G_0;

図 3: GF-ACG 合成アルゴリズム

4 生成実験

代表的な次数 (8, 16 ,32, 64, 128) の既約多項式を用いて，GF-AMG における生成時間を測定した．測定環境は，CPU が Intel Core2 Duo E4600 2.40GHz，メモリが 7GB である．GF-ACG による形式的検証には，汎用の数式処理システムである Risa/Asir を用いた．また，従来の設計手法との比較として，GF-ACG による形式的検証の代わりに Verilog-XL による論理シミュレーションを用いた場合の生成時間も測定した．表 2 にその結果を示す．表 2 の (A)，(B) はそれぞれ，形式的検証，論理シミュレーションを用いた場合の生成時間である．(B) では，次数（ビット長）に対して検証時間が指数関数的に増加することから，$GF(2^{16})$ を超える規模の乗算器の検証が終了せず，乗算器を生成することができなかった．一方，(A) では，次数に対する検証時間の増加を抑えられるため，機能が完全に検証された乗算器を短時間で生成できることが確認できる．なお，表 2 の生成時間の大半は検証時間が占めており，GF-ACG の合成時間，HDL への変換時間はそれぞれ，(A) の生成時間の 0.7%，15% 程度であった．

表 2: Mastrivito 乗算器の生成時間 (sec)

	$GF(2^8)$	$GF(2^{16})$	$GF(2^{32})$	$GF(2^{64})$	$GF(2^{128})$
(A)	3.374	5.188	9.487	19.55	52.61
(B)	0.279	9330	N/A	N/A	N/A

(A) 形式的検証，(B) 論理シミュレーション

5 まとめ

本稿では機能が完全に保証された Mastrovito 乗算器を短時間で生成可能なシステム GF-AMG を提案した．GF-AMG は Web サイト [4] から利用することができる．今後は，ガロア体所の他の算術演算回路も生成するシステムへの拡張を検討している．

参考文献

[1] K. Saito, N. Homma, and T Aoki. A graph-based approach to designing multiple-valued arithmetic algorithms. *Proc. 41th IEEE Int. Symp. Multiple-Valued Logic*, pp. 27–32, May 2011.

[2] N. Homma, K. Saito, and T. Aoki. A formal approach to designing cryptographic processors based on $GF(2^m)$ arithmetic circuits. *IEEE Trans. Information Forensics and Security*, Vol. 7, No. 1, pp. 3–13, February 2012.

[3] A. Halbutogullari and C.K. Koc. Mastrovito multiplier for general irreducible polynomials. *IEEE Trans. Computers*, Vol. 49, No. 5, pp. 503–518, May 2000.

[4] Arithmetic module generator for gf parallel multipliers. `http://www.aoki.ecei.tohoku.ac.jp/arith/gfamg/request_multiplier.py`.