インタリーブ型剰余乗算回路の評価

葛 毅　　坂井 修一

1　はじめに

べき剰余演算は公開鍵暗号の基本演算であり，剰余乗算に分解できる．この中で有効なものに，乗/除算をインタリーブする SRT 除算に基づく方法 [1] と，モンゴメリ乗算に基づく方法 [2] がある．本稿では，高基数 SRT 除算に基づく構成法に関して，有効な二つの実装モデルを評価している．まず，従来構成を更に高基数化した場合について評価し [3]，次に，スケーラブルな回路構成法を示し，評価する [4].

2　剰余乗算のアルゴリズム

除算に基づく剰余乗算回路 [1, 5] の高速化手法を含んだアルゴリズムは次のようになる．基数を r，r は 2 のべき乗，演算ビット数を n，法を $N, 2^{n-1} \leq N < 2^n$，被乗数，乗数，積を各々$x, y, RR, -\rho_1 N \leq x, y, RR \leq \rho_1 N$，中間積を $R_j, -\rho_2 N \leq R_j \leq \rho_2 N$，$r\rho_1 = \rho_2$，$1/2 < \rho_1 \leq 1$，$y$ の上位数桁を変換した数を $\hat{y}_j$，$\hat{y}_j \in \{-r/2, ..., -1, 0, 1, ..., r/2\}$，法の倍数を q_j，$q_j \in \{-q_{max}, ..., -1, 0, 1, ..., q_{max}\}$ とする．剰余乗算 $RR = (xy) \bmod N$ は次式を $j = \lfloor n/\log_2 r \rfloor$ から $j = -1$ まで繰り返すことで行う．

$$R_j = rR_{j+1} + \hat{y}_j x - rq_j N \qquad (1)$$

q_j は $-\rho_2 N \leq R_j \leq \rho_2 N$ を満たすように決定する．R の初期値は 0 である．積は $RR = R_{-1}/r$ である．また，式 (1) が収束する条件と，隣り合う $q, q-1$ を選択できる領域が連続する条件から，条件 $\rho_2 \leq \frac{2rq_{max}}{2r-1}, \rho_2 > \frac{r}{2}$ が得られる．q_{max} の最小値は $q_{max} \geq \lfloor \frac{2r-1}{4} \rfloor + 1$ である．

3　回路構成

一般的な回路構成では，式 (1) を 1 サイクルで行う．そして，中間積を冗長表現にして効率化する．

一方，このような一般的な構成は，オペランドが 1024 ビット程度の場合，回路規模が 10 万ゲート以上とかなり大きくなる．そこで，n ビットのオペランドを，w ビット単位に分解する．$x = \sum_{i=0}^{n/w-1} x(i) \cdot 2^{wi}$, $y = \sum_{i=0}^{n/w-1} y(i) \cdot 2^{wi}$, $N = \sum_{i=0}^{n/w-1} N(i) \cdot 2^{wi}$, $R = \sum_{i=0}^{n/w-1} R(i) \cdot 2^{wi}$ である．n は w の整数倍とする．そして，1 サイクルで w ビットずづ処理するように式 (1) を次のように変形する．c は各回の桁上げである．

$$\begin{aligned} &c = 0; \\ &\textbf{for}(i = 0; i < n/w; i++) \\ &\quad c \cdot 2^w + R_j(i) = \\ &\qquad rR_{j+1}(i) + \hat{y}_j x(i) - rq_j N(i) + c; \end{aligned} \qquad (2)$$

上式をデータパスとすることで回路規模を小さくできる．式 (2) は，ある j に対して n/w 回実行される．ところで，式 (2) の中で法の倍数 q_j の選択には時間がかかる．また，中間積 R_{j+1} に依存する．そのため，従来の構成では，この処理がクリティカルパスに加わっていた．しかし，ワード単位の構成では，ひとつ先の法の倍数 q_{j-1} の選択を，式 (2) が n/w 回実行されている間に，並行して実行することができる．これにより，法の倍数選択の処理時間を隠蔽でき，効率化できる．また，この構成では，データパスはオペランドのビット幅 n に依存しない．

データパスは法の倍数を選択する **pre_select_block** と式 (1) を計算する **main_block** で構成される．**pre_select_block** は，法の倍数選択に必要な精度のビット数により決まるビット幅で計算を行えばよく，w に依存しない．この法の倍数選択回路の加算器のビット数は，$k = r(q' - 1/2)$，$q' \in \{-q_{max} + 1, ..., q_{max}\}$ を重複領域の中心線の傾きとすると，$f = \lfloor \log_2((2\rho_2 - r) \cdot 2^{n-1}) \rfloor$ とし，最下位ビットを 0 ビット目として，$f - 1$ ビット目以上で計算すればよい．また，加算器の上端のビット位置は，$\lfloor \log_2\{(2rq_{max} + \rho_2 - r/2) \cdot 2^{n+1}\} \rfloor$ である．

表 1: 一般的な構成の合成結果

n	r	number of cycles	cycle time (ns)	total time (μs)	nubmer of gates (/2NAND)
1024	4	515	3.22	1.658	102059
	16	259	4.22	1.093	167771
	64	173	4.83	0.836	263464

表 2: 基数 4 のスケーラブルな構成の合成結果

logic block	w	cycle time (ns)	number of gates	time of modular multiplication	
				$n = 1024$	$n = 512$
pre_select_block	-	-	2043	-	-
main_block	32	2.69	3637	44μs	11μs
	64	2.90	7909	24μs	6.0μs
	128	3.07	16052	13μs	3.2μs

4 評価

基数 4 から基数 64 の一般的な構成と基数 4 のスケーラブルな構成を Verilog-HDL で設計し，論理合成ツール DesignCompiler で合成して得られる速度と面積の数値を用いて比較した．合成には CMOS 0.18μm セルライブラリを用いた．このライブラリはおよそ 300 種類のセルがある．

表 1 に一般的な構成の合成結果を示す．このうちのレジスタの面積は，1024 ビットで 37134 ゲートである．ゲート数は最小の 2 入力 NAND セルの面積 30.7 μm^2 より計算した．表 1 より基数 16, 64 では基数 4 に対し面積が各々約 1.6, 2.5 倍，速度が約 1.5, 2.0 倍である．表 2 に基数 4 のスケーラブルな構成の $w = 32, 64, 128$ におけるデータパスの合成結果を示す．クリティカルパスは **main_block** である．表より $w = 64$ のデータパスは 1 万ゲート以下 (2043+7909=9952) で構成できる．

SRT 除算に基づく剰余乗算回路とモンゴメリ乗算に基づく剰余乗算回路との違いは，式 (1) において後者の方が法の倍数の選択が容易なことである．前者の法の倍数選択回路の遅延の割合は，クリティカルパス全体のおよそ 2 から 3 割である．後者でも法の倍数選択回路の遅延があることを考慮すると，前者は，3 割程度遅延が大きいと考えられる．他文献 [6] によると，後者は 1024 ビットのべき剰余演算を 13730 ゲートのデータパスで，38.61ms で計算できる．

除算に基づく構成法は，モンゴメリ乗算に比べて，法の倍数の選択に時間がかかる．ワード単位に処理する構成では，この処理時間を本体の処理とオーバーラップさせることができ，このオーバーヘッドをなくしている．その処理に必要な回路規模も，基数 4 に基づく構成では，2000 ゲート程度と小さい．また，モンゴメリ乗算に必要な初期値の用意，前後処理を必要としない．w を大きくすれば，**pre_select_block** の割合は更に小さくなる．このことから，面積/速度の性能がモンゴメリ乗算に基づく構成法に匹敵すると考えられる．

5 おわりに

本稿では，高基数 SRT 除算に基づく剰余乗算回路における有効な二つの実装モデルを評価し，モンゴメリ乗算に基づく構成法と比較した．まず，従来の基数 4 の回路構成を更に高基数化し評価した．高速実装が要求される応用では更なる高基数化が有効であることが分かる．また，小型実装向きのスケーラブルな構成法を示し評価した．基数 4 におけるこの構成法では，1 万ゲート以下のデータパスで，剰余乗算を 24μs で実行できる．この性能はモンゴメリ乗算に基づく構成法に匹敵すると考えられる．

近年は，モンゴメリ乗算に基づく剰余乗算回路が有効であると考えられている．本稿では，除算に基づく構成法も有効な選択肢の一つであることを示した．

参考文献

[1] E. F. Brickell: “A fast modular multiplication algorithm with application to two key cryptography”, Advances in Cryptology - CRYPTO ’82, Chaum et al., Eds., New York, Plenum, pp. 51–60 (1983).

[2] P. L. Montgomery: “Modular Multiplication Without Trial Division”, Mathematics of computation, **44**, 170, pp. 519–521 (1985).

[3] 葛 毅, 櫻井隆雄, 阿部公輝, 坂井修一 : “RSA 暗号処理における高基数剰余乗算回路”, 信学技報, **ISEC2004-15**, pp. 15–20 (2004).

[4] 葛 毅, ルォン・ディン・フォン, 阿部公輝, 坂井修一 : “高基数 SRT 除算に基づくスケーラブル剰余乗算回路”, 暗号と情報セキュリティシンポジウム, pp. 307–312 (2005).

[5] N. Takagi: “A Radix-4 Modular Multiplication Hardware Algorithm for Modular Exponentiation”, IEEE Trans. Computers, **41**, 8, pp. 949–956 (1992).

[6] 佐藤証, 高野光司, 大庭信之 :“GF(p) 上の楕円曲線暗号回路のスケーラブルアーキテクチャ”, 信学論, **J85-A**, pp. 1264–1272 (2002).