高速乗算回路の一構成法

5H−10

佐藤証
日本アイ・ビー・エム株式会社東京基礎研究所

1. はじめに

インターネット上の電子商取引や，音楽・映画のデジタル・コンテンツ保護など，さまざまな用途に公開鍵暗号の利用が広がっている．筆者は組み込みハードウェアや認証サーバー用に、高速かつ小型のRSA暗号LSIを開発し[1]，また楕円暗号の回路構成法[2]についても提案を行ってきた．これらには加算器ベースの演算アクセラレータを用いていたが，同じアーキテクチャでは半導テクノロジーの進歩によって動作周波数が高くなりすぎ，消費電力の増加やロジック制御が難しくなるという問題を今後生じてくる．そこで加算器よりも1回あたりの処理データ量の多い乗算器を用い，それをゆっくり動作させたほうがこの点では有利となる．また一般に乗算器は加算器よりも構成が複雑で回路規模も大きいが，回路シミュレーション・論理合成ツールの進歩とLSIの微細加工技術の向上とともに，これらも解決されてきている．

しかしながら，半導体デバイスのパフォーマンスを最大限に引き出し，本当に小型で高速かつ消費電力の低いLSIを実現するためには，CADによる自動回路生成でなく人手によるカスタムレイアウト設計が欠かせない．とはいえ，これには多大な労力と設計時間が必要とされ，また人為的な設計ミスが入る可能性が大きいことも事実である．そこで本稿では，4-2加算ツリーと高速加算器による，カスタムレイアウト設計に適した高速乗算器の構成法について論ずる．

2. 高速加算器

図1はRSA暗号LSIに用いた加算器で，キャリー選択方式[3]とキャリー・スキップ方式[4]を組み合せることで高速化を図っている．各ブロック中の全加算器（以下FA:Full Adder）の数は，MSB側へ行くにつれ1ビットづつ増加している．これはブロック内部で発生してGIからGOへと伝播するキャリーと，ブロック上をスキップしていくキャリーCiの遅延を揃えて，両者に無駄な待ち時間が生じないようにするためである．PIはブロック内でその桁よりも下の各入力ビットXとYがXORされ，その結果が全てANDされたものである．さらにその桁のXとYのXOR結果とANDされてPO出力となる．下位ブロックのキャリーが確定する前は，加算出力ZはそのブロックへのX,Y入力だけに基づいて計算されている．そこへ下からキャリーが伝播し$C=1$となると，$PI=1$のビットについては桁上がりを受けてZがXORゲートにより反転される．このPIとZは事前に計算されているので，キャリー確定後，加算出力を得るのにANDとXORの2ゲート分の遅延時間しか生じない．この加算器の遅延ゲート数は，入力ビット数をnとするとき次式で与えられる．

$$\lceil \sqrt{2n} - 0.5 \rceil + 2 \qquad (\lceil\ \rceil \text{は小数点以下切り上げ})$$

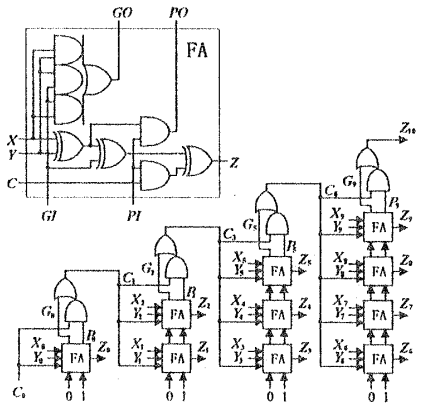

図1　キャリー選択・スキップを組み合せた加算器

図1では説明上FAをピラミッド型に配置したが，実際のレイアウトでは一列に並べられる．図2はRSA暗号LSIの演算コア部の顕微鏡写真で，4つに折り曲げられた1024ビット加算器が3つあり，その中に見える黒い筋は1ビットづつ増える各ブロックの切れ目である．

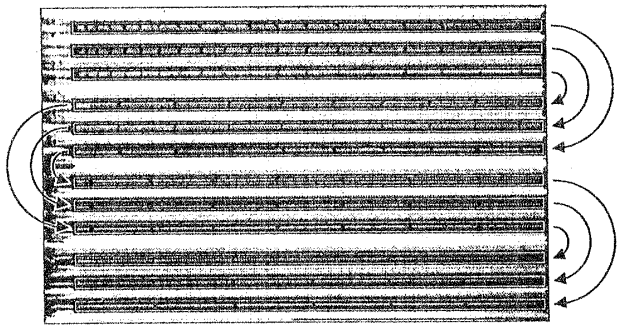

図2　RSA暗号LSIのコア部の顕微鏡写真

高速加算器として最も広く用いられているのはキャリー・ルック・アヘッド（以下CLA : Carry Look Ahead）方式である．表1はこのCLA回路を4桁づつ組み上げた加算器と，図1の方式のゲート遅延段数を，入力ビット数毎に比較したものである．128ビット入力までは

An Architecture of High-Speed Multiplier
Akashi SATOH
Tokyo Research Laboratory, IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi, Kanagawa 242-8502, Japan

図1の方式が，256ビット以降はCLAの方が速いが，乗算器への応用を考えると，64ビット×64ビットでも128ビット加算器しか必要としないので，前者を用いた方が有利である．配線等の回路エリアを考えた場合も図1の方式が優れており，また256ビット以上ではFAブロックをネストしてキャリーを多重に飛ばすことで，CLA以上の高速化も可能である[5]．

表1　CLAと図1の方式の遅延段数

ﾋﾞｯﾄ数	16	32	64	128	256	512	1024	2048
CLA	11	15	17	21	23	27	29	33
図1	8	10	13	18	25	34	47	66

3. 高速乗算器

一般にnビット乗算器は，n個の部分積をツリー状にキャリー保存方式で加え合わせていき，最集段の高速加算器によってキャリーをMSBまで伝播させている．途中の部分積加算に広く用いられるのが，3入力2出力のFAによる図3のWallaceツリー[6]である．ここでは8ビット×8ビット乗算器と小さいために規則正しく見えるものの，ビット数の増加とともに配線が非常に複雑となるので集積化にはあまり向かない．また図のツリーはできるだけ規則性を持たせて見易くしたために，最適な構成ではない．なおFAや半加算器HA (Half Adder)で，ツリーの遅延で後段に位置しているものほど濃いパターンで塗られている．

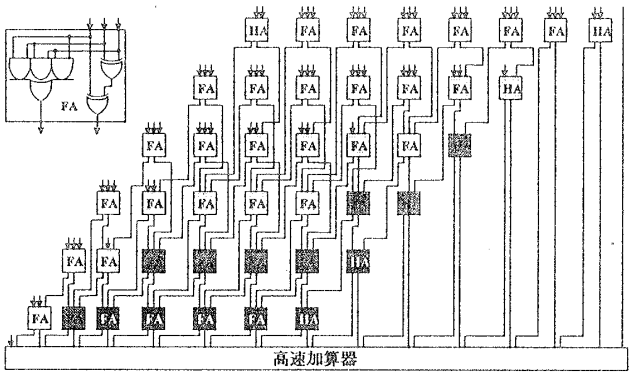

図3　Wallaceツリーによる乗算器

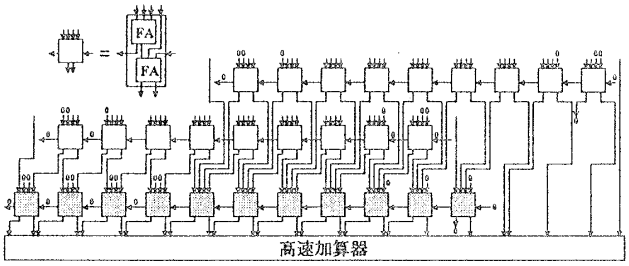

図4　4-2加算ツリーによる乗算器

これに対し図4の4-2加算器[7]は，4入力2出力の加算セルを横一列に接続し，それを縦に2分木状に接続していくもので，規則性に優れるため高集積化に向き，なおかつ高速という長所を兼ね備えている．実際，図3のWallaceツリーでは遅延ゲート数が最大8であるのに対し，図4では最大6ゲートと短い．さらに，各加算セルへ入力する部分積を作るには，乗数と被乗数で全ビットの組み合わせをANDする必要があるが，規則構造をもつ4-2加算ツリーでは，乗数と被乗数の配線を縦横にメッシュ状に張るだけでよいので非常に簡単である．

ANDはNAND+INVで2ゲート分の遅延となるので，4-2加算ツリーと図1の高速加算器を組み合せると，最終的な遅延ゲート数は2+6+8=16段となる．工夫をすればさらに1~2段減らすことも可能である．この構成によるnビット×nビット乗算器の遅延は，次の式で与えられ，n=16, 32, 64のとき，それぞれ18, 24, 34段となる．

$$3\lceil \log_2 n \rceil + \lceil \sqrt{4n} - 0.5 \rceil + 1$$

例えば0.25umCMOSテクノロジーを用いてカスタムレイアウトを行った場合，1ゲートの遅延は0.1ns程度となので，64ビット乗算器の遅延は3.4ns，およそ300MHzの高速動作が可能となる．

4. むすび

キャリー保存とキャリー・スキップの両方式を利用した加算器と4-2加算ツリーの組み合わせによる，高速かつ高集積化に向いた乗算器の構成を示した．この乗算器はカスタムレイアウトだけでなく，ゲート・アレイ実装にも適している．今後はこの乗算器を用いた公開鍵暗号回路を設計し，報告していきたい．

文献

1) Satoh et al.: "A High-Speed Small RSA Encryption LSI with Low Power Dissipation," LCNS-1369, pp.174-187 (1998).
2) 佐藤: "GF(p)上の小型・高速楕円暗号ハードウェア", 信学ソサイエティ大会, SA-55, (1998).
3) Lehman et al.: "Skip Techniques for High-Speed Carry Propagation in Binary Arithmetic Units," IEEE Trans. Elec. Comp., vol.EC-10, pp.691-689 (1961).
4) Bedrigi: "Carry-Select Adder," IRE Trans. Elec. Comp., vol.EC-11, pp.340-346 (1962).
5) 小林: "RSA アクセラレータの高速化限界," 情処Dicomo'97 ワークショップ, pp.587-592, (1997).
6) Wallace: "A Suggestion for a Fast Multiplier," IEEE Trans.Elec.Comp.,volEC-13,pp.14-17,(1964)
7) Vuillemin: "A Very Fast Multiplication Algorithm for VLSI Implementation," VLSI Journal vol.1, pp.39-52, (1983).