【数学応用】

初等数式 $(A + B)^2$ を用いたデジタル乗算器の話

群馬大学理工学府　佐々木 秀、 小林 春夫

1　背景と目的

デジタル乗算器はデジタル計算機、DSP チップ（Digital Signal Processor : デジタル信号 処理に特化したマイクロプロセッサ）等に広く用いられている。2 進数を扱う乗算は、2 進数の加算を何回も繰り返すことで計算できるため、多くの計算量を必要とする。デジタル乗算器を直接的に実現すると全加算器の２次元配列（例えば後出の図 6 など）になり、回路規模・消費電力・演算時間が大きくなってしまう問題がある。そのため回路量・消費電力の減少および高速化のために様々なアルゴリズムが提案され、それに基づきデジタル乗算器が設計・実現されてきた。

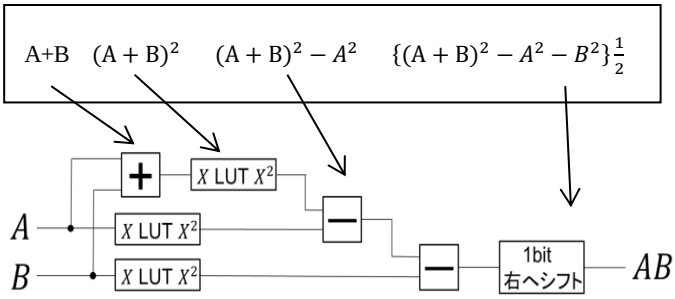
特にデジタル通信システムでは複雑・大量のデジタル演算をリアルタイムに行う必要があり、小規模なデジタル乗算器を実現できれば、一つのチップ内に多数実装・並列演算できるので、きわめて有効である。

ここでは 2 つのデジタル入力 A，B に対し、その和と 2 乗から積 AB を計算するのに、次の初等数学公式を用いてデジタル乗算器を実現する方式を検討する。

$$AB = \frac{1}{2}\{(A + B)^2 - A^2 - B^2\} \quad \cdots\cdots \quad (1)$$

この 2 乗演算のためにルックアップテーブル (Look-Up Table: LUT) を用いる。LUT とは、あらかじめ、ある入力 X に対する X^2 の数値をデジタルの表として用意しておくことである。また LUT と加算器、減算器のサイズを小さくするため A, B, A+B のそれぞれの上位ビット、下位ビットに分割して計算量を低減する方式 (Divide & Conquer 法) を用いる。

提案アルゴリズムを FPGA（Field-Programmable Gate Array：製造後に内部構成を自由に 設定できる集積回路）で実現・検証をすべく、検討したアルゴリズムで入力 8bit×8bit （出力 16bit）、および入力 16bit x 16bit (出力 32bit) の乗算器を設計し正しくデジタル乗算が行われていることを確認する。



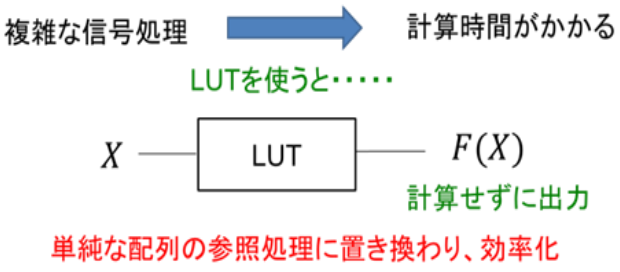
【図 1：式(1) を実現する回路構成】

2 検討する計算アルゴリズム

2.1 2乗則アルゴリズム

検討したアルゴリズムは式（1）の 2 乗則を用いる。1/2 は右シフト演算（実際には結線変更のみでよい）、二乗の計算部分はルックアップテーブル (Look-Up Table: LUT) を用いることで回路規模削減・低消費電力化・計算時間短縮を実現する。図 1 に式(1)を実現する回路構成を示す。LUT はメモリ (RAM または ROM)である。入力はメモリの”アドレス”であり、出力はメモリの”データ”である。メモリ内に計算データを記憶しておくことで、LUT で所望の計算結果を得ることができる。

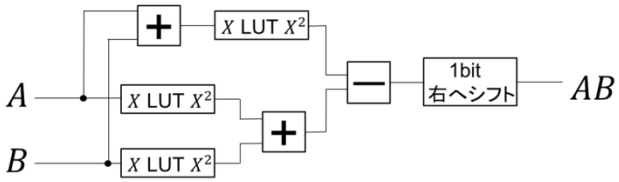
図 1 では 3 つの LUT を用いている。しかし演算時間のタイミングを考慮していないため、図 3 のような回路構成にすることで演算時間のバランスを考慮した回路も考えられる。



【図 2：ルックアップテーブル(LUT)回路】

2.2　対数を用いるアルゴリズム

デジタル乗算を容易に計算する方法の一つとして、対数を用いる演算する手法もある。２つのデー



【図 3：演算時間のバランスを考慮した回路】

タ A, B に対して AB の乗算を、加算器と LUT を用いて次のように計算する。

① A, B の対数 log A, log B を
対数データの LUT を用いて得る。

② log A + log B (= log AB) を加算器で得る。

③ log AB から AB を
指数データの LUT を用いて得る。

しかし高精度で対数、指数を得るためには LUT のビット数を大きくしなければならず、回路規模が大きくなってしまうので、今回は検討しない。

2.3 Divide & Conquer について

ここでは二乗計算を担う LUT の回路量低減について検討する。

適当な入力値 A のビット数 n を大きくすると出力 A^2 の 計算精度を保つためには LUT のサイズを（n^2 に比例して）大きくしなければならない。そこで n を上位ビットと下位ビットに分割して計算することを考える。

たとえば図 5(a) のように入力 A が 8 ビットで、その上位 4bit を A_H, 下位 4 ビットを A_L とする。ここで A^2 の計算は次のようになる。

$$A^2 = (A_H + A_L)^2 =$$
$$A_H^2 \text{ (8bit 左シフト)} + 2A_H A_L \text{ (4bit 左シフト)} + A_L^2 \quad \cdots(3)$$

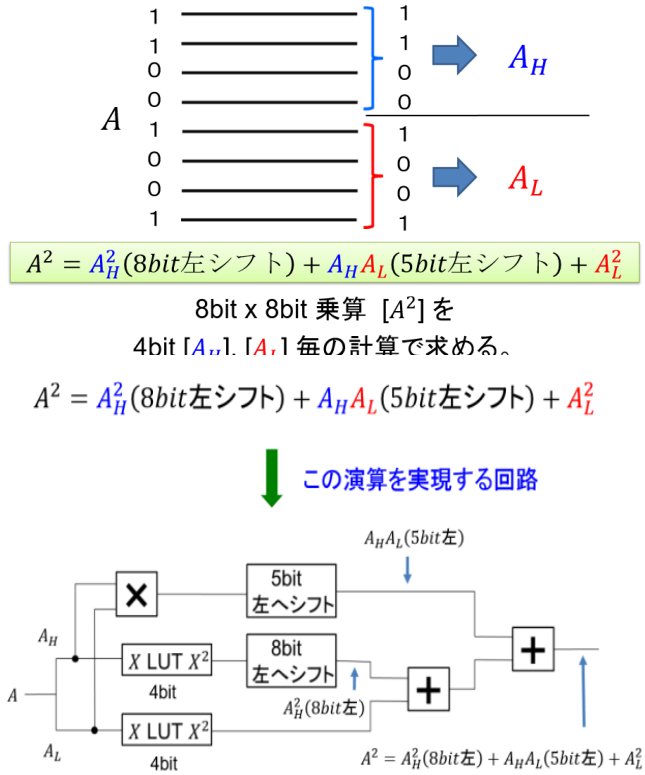
式(3) を用いた入力 A (8bit) の 2 乗 A^2 (16bit) 回路の計算回路を図 5(b) に示す。8 ビットの入力信号 A を上部 4 ビットと下部 4 ビットに分割してビット数を減らしてから計算し、シフト演算によって桁を揃える。ここでシフト演算は回路上では配線の配置を換えるだけで実現できるため余計な回路を必要しない。

分割したビット列をさらに分割する（Divide & Conquer 法）でビット数を分割することによって小さなビット数の LUT を用いて演算できる。

同じ手法を繰り返し行うことで、大きなビット数の計算を小さなビット数の LUT で計算できる。

図 5 を参照し、8bit で $A = 11001001 = (201)_{10}$ について考える。まず 10 進数を用いて理論計算を行う。

例：8bit（A=11001001）について考える。



【図 5 (b) ：データ A を上位、下位ビット分割し A^2 の計算を行う回路】

上部（A_H）4bit、下部（A_L）4bit に分割する。

$A_H = 1100\ (12)_{10}$、 $A_L = 1001\ (9)_{10}$

これを使って演算を行っていく。

まず分割した A_H、A_L を 2 乗する。

$A_H^2 = 12^2 = 144$

$A_L^2 = 81$

次に A_H^2 を 8bit 左にシフトして桁を合わせる。

A_H^2 (8bit 左) $= 12^2 \times 256 = 36864$

次に分割した A_H、A_L を足し、2 乗する

$A_H + A_L = 12 + 9 = 21$

$(A_H + A_L)^2 = 21^2 = 441$

そして求めた値を回路に基づいて減算していく。

$(A_H + A_L)^2 - A_H^2 = 441 - 144 = 297$

$(A_H + A_L)^2 - A_H^2 - A_L^2 = 297 - 81 = 216$

$\{(A_H + A_L)^2 - A_H^2 - A_L^2\}$ (4bit 左)

$= 216 \times 16 = 3456$

最後に A の 2 乗と、分割して求めた解が合っているかを確かめる。

$(A_H)^2$ (8bit 左) ＋ $(A_H + A_L)^2$ (4bit 左) ＋ $(A_L)^2$

$= 36864 + 3456 + 81$

$= 40401 = 201^2 = A^2$

このように、分割して求めた解とA^2の値が一致したので、この回路の正当性が示せたことになる。

次に、2 進数において上記と同様の手順で理論計算を行う。上部（A_H）4bit、下部（A_L）4bit に分割する。

$A_H = 1100\ (12)_{10}$、　$A_L = 1001\ (9)_{10}$

さらに演算を進める。

$A_H^2 = 10010000\ (144)_{10}$

A_H^2 (8bit 左) $= 1001000000000000\ (36864)_{10}$

$A_L^2 = 1010001\ (81)_{10}$

$A_H + A_L = 00010101\ (21)_{10}$

$(A_H + A_L)^2 = 110111001\ (441)_{10}$

$(A_H + A_L)^2 - A_H^2 = 100101001\ (297)_{10}$

$(A_H + A_L)^2 - A_H^2 - A_L^2 = 11011000\ (216)_{10}$

$\{(A_H + A_L)^2 - A_H^2 - A_L^2\}$ (4bit 左)

$= 110110000000\ \ (3456)_{10}$

$(A_H)^2$ (8bit 左) ＋ $(A_H + A_L)^2$ (4bit 左) ＋ $(A_L)^2$

$= 1001000000000000\ (36864)_{10}$

$+ 110110000000\ (7056)_{10}$

$+ 1010001\ (81)_{10}$

$= 1001110111010001\ (40401)_{10} = A^2$

以上より 10 進数の場合と同じ出力結果を得られることが確認できた。

2.3 問題の分割（Divide & Conquer 法）による演算速度の向上

大きなビット数を持つ LUT を分割することで回路面積を小さくすることが出来ると同時に、演算速度も大幅に縮小できる。

加算器はビット数が大きくなるほど消費電力や計算時間を大きくなってしまう[3]。そのためデータを分割することによって、演算処理をする加算器側のビット数が減り、演算速度が向上できる。また、より大きなビット数から分割することによって、その向上性能も大きくなることが分かる。

ここでは簡単のため、4bit まで分割して理論計算と回路構成を行った。

3 シミュレーションによる検討

検討アルゴリズム・回路構成の正当性、つまり検討した内容が実装に反映させることが出来るかどうかを検証するため「Verilog HDL」（**C 言語に似た、チップのハードウェア開発言語**）を用いて回路シミュレーションを行った。

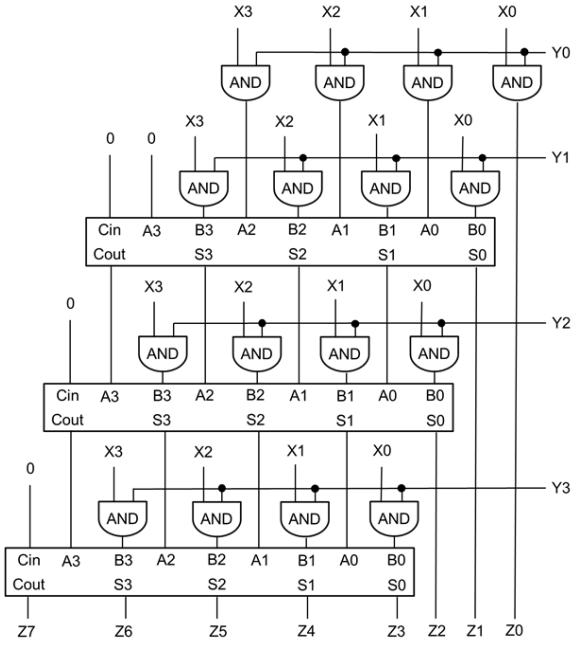
具体的には、シミュレーションソフト上に回路構成を実現し、二つの入力値を変えて演算させて結果を出力し、結果が正しいかどうかをチェックした。入力が 4bit×4bit で出力が 8bit の場合は、数字の組み合わせが 16×16 = 256 通り、入力が 8bit×8bit で出力が 16bit の場合、 256×256 = 65536 通り、入力が 16bit×16bit で出力が 32bit の場合は 65536×65536 = 4294967296 通りある。それらすべての数値で提案アルゴリズムが正しく乗算を行っていることを確認した。

シミュレーションの結果、このプログラムを使用すれば提案アルゴリズムの FPGA 実現ができることが分かった。

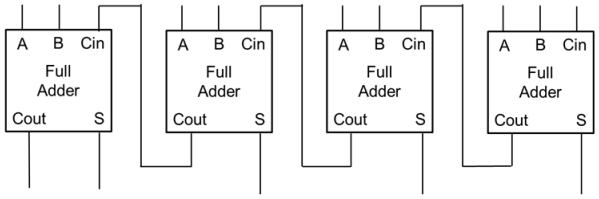
4 従来乗算器との比較

計算アルゴリズムを用いて乗算器の回路設計を行い、シミュレーション、FPGA の可変集積回路実装を経て正しく動作することを確認した。

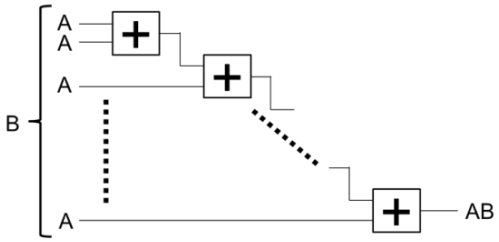
ここで、従来の乗算器との比較を行う。最初に述べたとおり、直接的なデジタル乗算器は加算器を連続的に組み合わせて構成されている（図 6 を参照）。



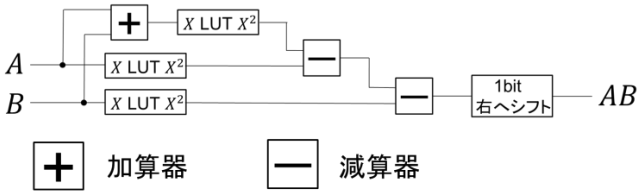
【図 6-1　従来の乗算器の回路図（4bit）】



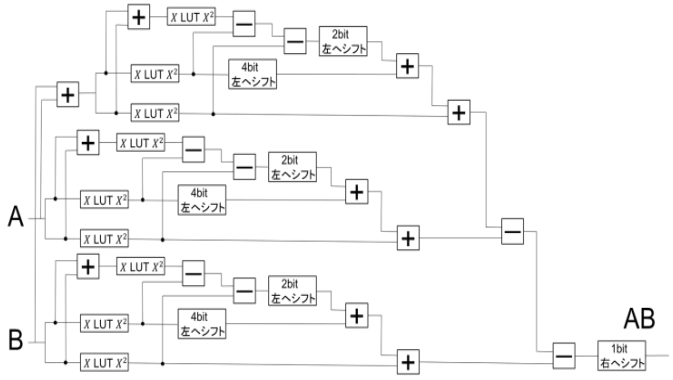
【図 6-2. 全加算器の回路図】



【図 7：従来の乗算器の回路図】



【図 8:構成した回路図(Divide &Conquer なし)】



【図 9:構成した回路図(Divide & Conquer 有り)】

5 まとめと今後の課題

二乗則の計算アルゴリズムを用いた回路構成と、そのハードウェア実現を検討し、FPGA 実装のためのシミュレーションで動作を確認した。今後は具体的には次を重点的に検証していく。

・従来法との回路量の定量的な比較
・計算精度と演算器・LUT のビット数の明確化
・実装 FPGA 動作クロック周波数）と
計算速度の明確化
・マイナスの数値の場合の計算。

デジタル乗算器はすべて２進数で表現されており、マイナスがつくと２の補数によってマイナスを表現する。ビット分割の際にはその配慮が必要であるが、今後はそれに対応していく。

また次の２乗則アルゴリズムも有力である。

$$AB = \frac{1}{4}\{(A + B)^2 - (A-B)^2\} \quad \cdots\cdots (2)$$

加減算回数は 3 回と同じであるが、LUT 参照処理は 2 回になる。ここでは簡単のため A, B が正の数の場合のみを扱っているので　式(1)を検討した。次のステップとして式(2)を検討する予定である。

謝辞： デジタル乗算器のアルゴリズムに関し有益なコメントをいただきました群馬大学　魏書剛先生、大阪工業大学　牧野博之先生に感謝いたします。FPGA 実装にアドバイスをいただきました弓仲康史先生、王俊善さん、李从兵さんに感謝いたします。論理演算式を導出してもらった孫逸菲さん、姚丹さん、原稿を見ていただいた田部井勝稲先生に感謝します。

【参考文献】

[1] A. V. Oppenheim, and R. W. Shafer, Digital Signal Processing, Printice-Hall (1975)
[2] L. Cohen, Time-Frequency Analysis, Prentice Hall
[3] A Technical Tutorial on Digital Signal Synthesis, Analog Devices, Inc. (1999).
[4] N. Weste, D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, Addison Wesley (2010)
[5] 佐々木秀、小林春夫「短時間スペクトラム解析の計算アーキテクチャの検討」第 5 回電気学会東京支部栃木・群馬支所合同研究発表会　ETT-15-69, ETG-15-69 宇都宮　(2015 年 3 月)
[6] 佐々木秀、小林春夫「2 乗則を用いたデジタル乗算器アルゴリズムの検討」第 38 回多値論理フォーラム、北海道大学　(2015 年 9 月)