社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

EVBDDを用いた数値計算回路の構成

永山　忍† 笹尾　勤†† Jon T. Butler†††

† 広島市立大学 情報工学科　〒 731–3194 広島市 安佐南区 大塚東 3–4–1
†† 九州工業大学 電子情報工学科　〒 820–8502 福岡県 飯塚市 川津 680–4
††† 海軍大学院大学　アメリカ カリフォルニア州 モントレー

あらまし　本稿は, 三角関数, 対数関数, 平方根演算, 逆数演算などの初等関数およびその合成関数を計算する数値計算回路の構成法を提案する. 本数値計算回路は, 与えられた定義域を不等区間に分割し, 各区間毎に関数を多項式で近似する. 不等区間分割の実現に, EVBDD (Edge-Valued Binary Decision Diagram) を用いることで, 様々な関数を従来手法よりコンパクトかつ高速に実現できる. 実験により以下を示す: 1) EVBDD を用いた本手法は, MTBDD (Muti-Terminal BDD) を用いた従来手法の 7% のメモリ量かつ 40% の遅延時間で, 不等区間分割を実現できる. 2) それにより, 本数値計算回路は, 従来手法に基づく数値計算回路の 38% のメモリ量かつ 59% の遅延時間で関数を実現できる.
キーワード　EVBDD, 不等区間分割, 区分多項式近似, 数値計算回路, FPGA 実装.

Architecture for Numerical Function Generators Using EVBDDs

Shinobu NAGAYAMA†, Tsutomu SASAO††, and Jon T. BUTLER†††

† Department of Computer Engineering, Hiroshima City University, Ozuka-Higashi 3–4–1, Asa-Minami-Ku, Hiroshima, 731–3194 Japan
†† Department of Computer Science and Electronics, Kyushu Institute of Technology, Kawazu 680–4, Iizuka, Fukuoka, 820–8502 Japan
††† Department of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA 93943-5121 USA

Abstract　This paper presents an architecture and a synthesis method for fast and compact numerical function generators (NFGs) for trigonometric, logarithmic, square root, reciprocal, and combinations of these functions. Our NFG partitions a given domain of the function into non-uniform segments, and approximates the given function by a polynomial function for each segment. By using an edge-valued binary decision digram (EVBDD) to realize the non-uniform segmentation, we can implement fast and compact NFGs for a wide range of functions. Implementation results on an FPGA show that: 1) To realize a non-uniform segmentation, our method using EVBDD requires, on average, only 7% of the memory and 40% of the delay time needed by the existing method using multi-terminal BDD (MTBDD); and therefore, 2) our NFG requires, on average, only 38% of the memory and 59% of the delay time needed by the existing NFG using non-uniform segmentation and MTBDD. Our automatic synthesis system generates such fast and compact NFGs quickly.
Key words　Edge-valued binary decision diagram (EVBDD), non-uniform segmentation, piecewise polynomial approximation, numerical function generators (NFGs), FPGA implementation.

1. はじめに

三角関数, 対数関数, 平方根演算, 逆数演算などの初等関数およびその合成関数は, ディジタル信号処理, 通信, ロボット工学, 天体物理学などの様々な分野で広く利用され, 高速な計算を行うためにハードウェア実装が要求されている. また, 近年では, 携帯端末などの組み込みシステムでも高度なディジタル信号処理が行われようになり, 高速かつコンパクトなハードウェア実装がますます重要になっている.

初等関数をハードウェア実装する手法として, CORDIC (COordinate Rotation DIgital Computer) アルゴリズム [2], [31] に代表される繰り返しアルゴリズムがしばしば用いられる. 代表的な FPGA ベンダー [1], [32] も, 数値計算 IP (Intellectual Property) として CORDIC を用意している. CORDIC アルゴリズムは, 加減算, シフト演算, およびテーブル参照の単純な演算の繰り返しにより, 初等関数を計算できるため, コンパクトな数値計算回路で初等関数を実現できる. しかし, 一般に, 繰り返しアルゴリズムは計算時間が長く, 高速な計算が要求されるアプリケーションに不向きである. また, 複数の初等関数から成る合成関数を計算する場合, 個々の初等関数を CORDIC で順に計算していくと, さらに多くの計算時間を要するため, 合成関数を直接計算できる手法が必要である.

数値関数の高速な実装法として, 関数表を単一メモリで直接実装する方法がある. この手法は, 一回のメモリアクセスだけで関数を計算できるため, 非常に高速である. 近年のメモリの大容量化と低価格化のため, 低精度で関数を実装する場合 (入力のビット数が小さい場合), この手法は有効である. しかし, 高精度

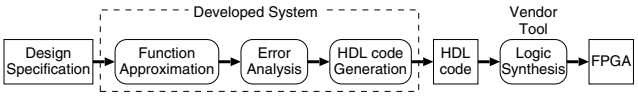


図 1　数値計算回路の合成フロー

で実装する場合には, 必要なメモリ量が大きくなりすぎ, 実装が困難になる.

実装に必要なメモリ量を削減するために, 与えられた関数を多項式 (区分多項式) で近似し, その多項式を実現する手法が提案されている [5], [7]ー[9], [14], [16], [22], [28]ー[30]. 既存手法の多くは, 定義域を等区間に分割し, 多項式近似を行う. $\sin(x)$ や e^x などの単純な初等関数を実現する場合, 等区間分割でも十分にコンパクトな数値計算回路を生成できるが, $\sqrt{-\ln(x)}$ などの複雑な関数では, 区間数が大きくなりすぎ実現できない場合がある. この問題を解決するために, 定義域を不等区間に分割する手法が提案された [4]. 不等区間分割法は, 同じ近似誤差の下で, 等区間分割法より少ない区間数で関数を多項式近似できる. しかし不等区間分割は, 複雑な区間指定回路が必要になり, 結果的に数値計算回路の面積や速度が劣化する場合がある. そこで [14]ー[16] は, 特殊な不等区間分割法を用いることで, 区間指定回路の複雑さを軽減した. 一方, [25], [26] は, LUT カスケードを用いて区間指定回路を実現することで, 区間指定回路自体の複雑さを軽減した. LUT カスケードは任意の不等区間分割を実現でき, その複雑さ (サイズ) は区間数に依存するため, [21] は, 二次多項式を用いて区間数を削減することによって区間指定回路のサイズを削減した.

本稿では, 区間指定回路のサイズと遅延時間をさらに削減するために, 新たな不等区間分割法とその実現法の両方を提案する. 新たな分割法で得られた不等区間分割を EVBDD を用いた区間指定回路で実現することにより, 従来手法よりコンパクトかつ高速な数値計算回路を自動生成できることを示す. 数値計算回路の自動合成フローを図 1 に示す. 本合成システムは, MATLAB 等の数値計算ソフト(注1) で記述された設計仕様から自動的に HDL コードを生成する. 設計仕様として, 数値関数 $f(X)$, X の定義域 $[A, B)$, 近似多項式の次数 d, および設計する数値計算回路の許容誤差を用いる. 本合成システムは, まず, 与えられた定義域を幾つかの区間へ分割し, 各区間を多項式近似する. 次に, 数値計算回路の誤差を解析し, 計算に用いる演算器の精度 (ビット数) を算出する. 最後に, 誤差解析で算出された精度をもとに, HDL コードを生成する.

2. 用語の定義

2.1 数値表現法と精度

［定義 1］ 二進固定小数点で表現された数値 X を

$$X = (x_{l-1}\, x_{l-2}\, \ldots\, x_1\, x_0.\, x_{-1}\, x_{-2}\, \ldots\, x_{-m})_2$$

と表記する. ただし, $x_i \in \{0, 1\}$, l は整数部のビット数, m は小数部のビット数である. 本稿では, 負数を 2 の補数で表現する.

［定義 2］ **誤差**とは, もとの値との差分絶対値を意味し, 特に, 本稿では, 関数近似で生じる誤差を**近似誤差**, 値を有限桁の二進固定小数点で表現した際に生じる誤差を**丸め誤差**という. **許容誤差**とは, 仕様で許容できる誤差の最大値を意味し, 特に, **許容近似誤差**は, 許容できる近似誤差の最大値を意味する.

［定義 3］ **精度 (precision)** は, 実数計算における有効桁数を意味する. 特に, n **ビット精度**とは, 実数計算において有効桁数が n ビット, すなわち, $n = l + m$ を意味する. 本稿で, n ビット精度の数値計算回路は, n ビット入力の回路を意味する.

［定義 4］ **確度 (accuracy)** は, 実数計算における小数点以下の有効桁数を意味する. m **ビット確度**とは, 実数計算において小数点以下の有効桁数が m ビットであることを意味する. 特に, 最大誤差が 2^{-m} のときの m ビット確度を **1 ulp** (unit in the last place) という. 本稿で, m ビット確度の数値計算回路は, 小数部 m ビット入力, 小数部 m ビット出力でなおかつ, 出力値が 1 ulp の回路を意味する. 1 ulp の出力値を得るためには, 計算途中では, それ以上の確度で計算する必要がある.

(注1)：現在の合成システムでは, フリーソフト Scilab [27] で仕様を記述できる.

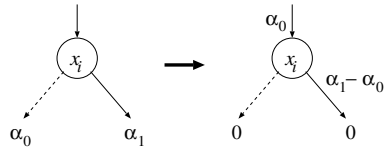
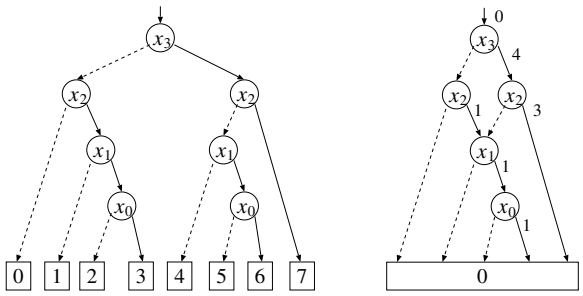


図 2　MTBDD から EVBDD への変換

x_3	x_2	x_1	x_0	f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	2
0	1	1	1	3

x_3	x_2	x_1	x_0	f
1	0	0	0	4
1	0	0	1	4
1	0	1	0	5
1	0	1	1	6
1	1	0	0	7
1	1	0	1	7
1	1	1	0	7
1	1	1	1	7

(a) 関数表



(b) MTBDD　　(c) EVBDD

図 3　MTBDD と EVBDD

2.2 Edge-Valued Binary Decision Diagram

［定義 5］ 二分決定グラフ (BDD: Binary Decision Diagram) [3] は, 論理関数: $\{0, 1\}^n \to \{0, 1\}$ をグラフで表現したもので, 論理関数にシャノン展開を繰り返し適用することで得られる. BDD は, 論理関数値 0 および 1 を表現する二つの終端節点と, 入力変数を表現する非終端節点から成る. 各非終端節点は, 変数値に対応する重みなしの二つの枝, 0-枝と 1-枝を持つ.

［定義 6］ MTBDD (Multi-Terminal BDD) [6] は, BDD を拡張した表現法であり, 整数関数: $\{0, 1\}^n \to Z$ を表現する. ただし, Z は整数集合を表す. MTBDD には, 整数値を表現する複数の終端節点がある.

［定義 7］ EVBDD (Edge-Valued BDD) [12], [13] は, BDD を拡張した表現法であり, 整数関数を表現する. EVBDD は, 0 を表現する一つの終端節点と, 重み付きの 1-枝を持った非終端節点から成る. ただし, 枝の重みは整数値である. EVBDD は, MTBDD の各非終端節点に図 2 の変換を適用することで得られる. 図 2 で, 点線は 0-枝を表し, 実線は 1-枝を表す.

BDD, MTBDD, EVBDD の詳細については, [23] を参照されたい.

［例 1］ 図 3(b) の MTBDD および (c) の EVBDD は, 図 3(a) で与えられる関数 f を表現している. $f(1010)$ の関数値 5 を得るために, MTBDD では, 入力変数の値に従ってグラフを辿り, 到達した終端節点の値から関数値を得る. 一方, EVBDD では, 入力変数の値に従い, 終端節点までグラフを辿ったときの枝の重みの総和から関数値を得る. (例終り)

入力	数値関数 $f(X)$, 定義域 $[A, B)$, X の確度 m_{in}, 近似多項式の次数 d, 許容近似誤差 ε_a.
出力	不等区間 $[A, P_0), [P_0, P_1), \ldots, [P_{t-2}, B)$.
処理	
1.	$[A, B)$ における最大近似誤差 $\varepsilon_d(A, B)$ を計算する.
2.	$\varepsilon_d(A, B) < \varepsilon_a$ または $B - A \leqq 2^{-m_{in}}$ ならば終了.
3.	$P = (A + B)/2$ とし, $[A, B)$ を等区間 $[A, P), [P, B)$ に分割.
4.	各区間 $[A, P), [P, B)$ について上の処理を再帰的に繰り返し, 全区間で, 最大近似誤差が ε_a より小さくなったら終了.

図 4　定義域の再帰的分割アルゴリズム

3. 不等区間分割に基づく区分多項式近似

関数 $f(X)$ を区分多項式で近似するために, X の定義域 $[A, B)$ をいくつかの区間に分割し, その各区間で, $f(X)$ を多項式近似する. この場合, 近似多項式の係数を記憶するメモリ量を削減するために, できるだけ少ない区間数で関数を近似することが求められる. 従来法の多くは, 定義域を等区間へ分割し関数近似を行う [5], [7]–[9], [14], [16], [22], [28]–[30]. そのような等区間分割法は, $\sin(x)$ などの単純な初等関数を近似する場合には有効であるが, $\sqrt{-\ln(x)}$ などの複雑な関数では, 区間数が大きくなりすぎ実現できない場合がある. この問題を解決するために, 定義域を不等区間に分割する手法が提案された [4]. 不等区間分割は, 同じ近似誤差の下で, 等区間分割より少ない区間数で関数を近似できるが, 不等区間分割は, 複雑な区間指定回路が必要になり, 結果的に数値計算回路の面積や速度が劣化する場合がある. [15] は, 階層的分割とよばれる特殊な不等区間分割を用いることで, 区間指定回路の複雑さを軽減した. [15] で提案された階層的分割法は, 予め計算された 4 つの分割タイプから関数に適した分割を選択し, 不等区間分割を得る手法である. この手法では, ツールが関数に適した分割を選択するのではなく, 設計者が分割タイプを選択しツールに入力しなければならない. 本稿では, 関数に適した分割を自動生成するために, 新たな分割法, **再帰的分割法**を提案する.

3.1 再帰的分割法

本稿で提案する再帰的分割法を図 4 に示す. このアルゴリズムは, 入力として, 関数 $f(X)$, 定義域 $[A, B)$, 入力変数 X の確度 m_{in}, 近似多項式の次数 d, および許容近似誤差 ε_a を与えると, 各区間の最大近似誤差が許容近似誤差より小さくなるまで, 区間の二等分割を再帰的に繰り返し, t 個の不等区間 $[A, P_0), [P_0, P_1), \ldots, [P_{t-2}, B)$ を生成する. この再帰的分割の各区間の幅 w_i は, $w_i = 2^{h_i} \times 2^{-m_{in}}$ に制限され, 分割点 P_i は, 0 が最下位から h_i ビット連続する値 (つまり $P_i = (\ldots p_{-j+1}\, p_{-j}\, 00\ \ldots\ 0)_2$, ここで $j = m_{in} - h_i$) に制限されることに注意. 図 4 で示されているように, 区間数を削減するためは, 広い区間を小さい誤差で近似できる多項式が望ましい. 本稿では, Chebyshev 多項式を用いて関数を近似する. 関数 $f(X)$ の区間 $[S, E]$ における, d 次 Chebyshev 多項式近似の最大近似誤差 $\varepsilon_d(S, E)$ は, 以下の式で与えられる [17].

$$\varepsilon_d(S, E) = \frac{2(E - S)^{d+1}}{4^{d+1}(d+1)!} \max_{S \leqq X \leqq E} |f^{(d+1)}(X)|$$

ここで $f^{(d+1)}(X)$ は, 関数 f の $d+1$ 次導関数を表す. 本稿では, この式を用いて各区間の最大近似誤差を計算する.

3.2 近似値の計算

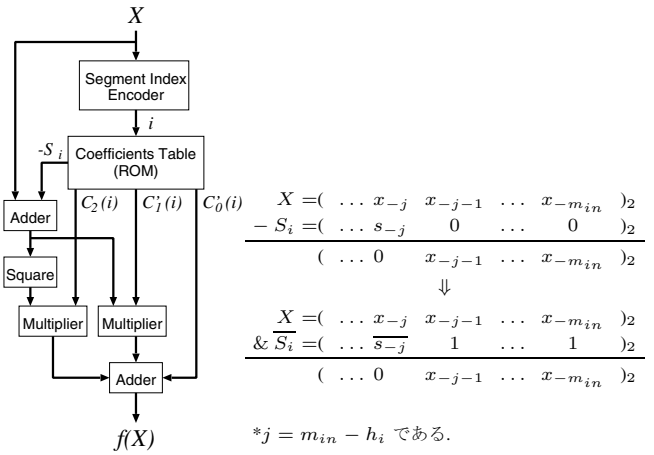
各区間において, 関数 $f(X)$ をそれぞれ異なる Chebyshev 多項式で近似するため, ある値 X における $f(X)$ の近似値は, X を含む区間の多項式 $g(X, i) = C_d(i)X^d + C_{d-1}(i)X^{d-1} + \ldots + C_0(i)$ で計算する. ここで i は, 区間に割り当てた区間番号であり, $C_d(i), C_{d-1}(i), \ldots, C_0(i)$ は, 区間 i における Chebyshev 多項式の係数である. 各係数の導出法は, [17] を参照されたい. 各区間 $[S_i, E_i)$ において, X に $X - S_i + S_i$ を代入することにより以下の式を得る.

$$g(X, i) = C_d(X - S_i)^d + C'_{d-1}(i)(X - S_i)^{d-1} + \ldots + C'_0(i) \quad (1)$$

ただし,

$$C'_j(i) = \sum_{k=0}^{d-j} \binom{j+k}{j} C_{j+k}(i) S_i^k \quad (j = 0, 1, \ldots, d-1).$$

(1) を用いることで, 乗算器のサイズを削減できるため, 本数値計算回路は, (1) を採用する.

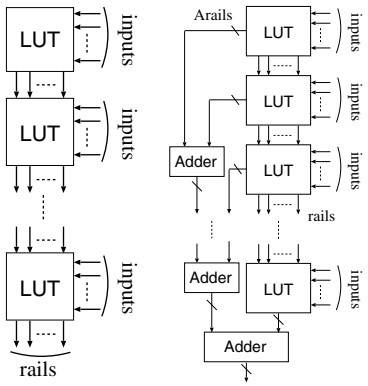


(a) 回路全体の構成　　(b) AND ゲートによる $X - S_i$ の計算

図 5　二次多項式に基づく数値計算回路の構成

区間	区間番号
$A \leqq X < P_0$	0
$P_0 \leqq X < P_1$	1
⋮	⋮
$P_{t-2} \leqq X < B$	$t-1$

(a) 区間指定関数

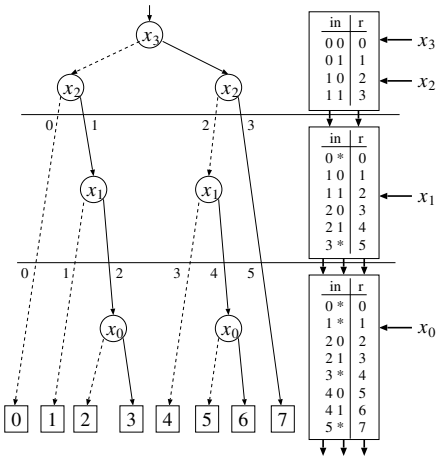


(b) LUT カスケード　(c) LUT カスケードと加算器による実現

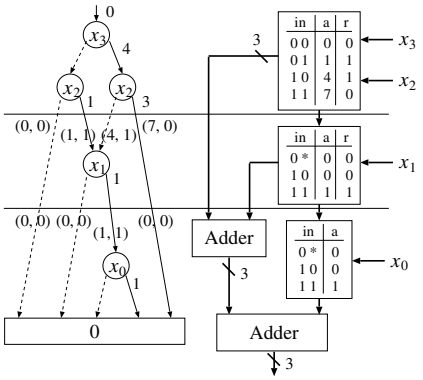
図 6　区間指定回路

4. 数値計算回路の構成

図 5 は, 二次多項式に基づく数値計算回路の構成を示している. 図 5(a) に示すように, 前節の (1) の多項式は, 区間指定回路, 係数表, べき乗回路, 乗算器, 加算器で実装できる. この回路構成は, 任意の不等区間分割を実現できるが, 3.1 節の再帰的分割を実現する場合, S_i は下位ビットが 0 の値に制限され, $X - S_i < E_i - S_i = 2^{h_i} \times 2^{-m_{in}}$ に制限されているので, 図 5(b) に示すように, $(X - S_i)$ の計算を $\overline{S_i}$ と AND ゲートによって実装できる. 区間指定回路は, X を入力として, X を含む区間の区間番号 i を出力する. X を n ビット精度とすると, 区間指定回路は, 図 6(a) に示されている区間指定関数 $seg\_func(X) : \{0,1\}^n \rightarrow \{0, 1, \ldots, t-1\}$ を実現する回路である. ここで, t は区間数を表す.



(a) MTBDD を用いた区間指定回路



(b) EVBDD を用いた区間指定回路

図 7　区間指定回路の実現例

4.1　区間指定回路の構成

区間指定関数 $seg\_func(X)$ を MTBDD で表現し, MTBDD を用いて関数分解することにより図 6(b) に示す LUT カスケード [11], [24], [25] が得られる. LUT カスケードは, 任意の $seg\_func(X)$ を区間数に依存するサイズで実現でき [25], 区間数を削減することで LUT カスケードのサイズを大幅に削減できることを [21], [26] で示した. 3. 節の再帰的分割によって得られた $seg\_func(X)$ も, コンパクトな LUT カスケードで実現できるが, 区間指定回路のメモリ量と遅延時間をさらに削減するために, 本節では, EVBDD を用いた区間指定回路の構成を示す.

MTBDD と同様に, EVBDD で $seg\_func(X)$ を表現し, EVBDD を分解することにより, 図 6(c) に示す LUT カスケードと加算器を組み合わせた回路で実現できる. LUT カスケードで, 隣接する LUT 間の信号線をレール (rail) といい, レールは, EVBDD を辿るときの部分グラフを表現する. 各 LUT からのレール以外の出力は辿った枝の重みの和を表現する. 本稿では, そのような出力を A レール (Arail) と呼ぶ.

［例 2］　図 3 の MTBDD と EVBDD を分解することにより, 図 7 の区間指定回路を得る. 図 7 は, MTBDD および EVBDD の分解と各 LUT との対応を図示している. この図で, 各 LUT 内の表中の 'r' は BDD の部分グラフを表現するレールを表し, 'a' は枝の重みの和を表現する A レールを表す. MTBDD で, 横線と交わる枝に割り当てた数字は, 部分グラフを表す. EVBDD で, 同様の枝に割り当てた "(a, r)" は, 枝の重みの和と部分グラフをそれぞれ表す. 図 7(a) の回路に必要なメモリ量は, $2^2\times2+2^3\times3+2^4\times3=80$ ビットであり, 回路段数は 3 段 (LUT 3 段) になる. 一方, (b) の回路のメモリ量は, $2^2\times4+2^2\times2+2^2\times1=28$ ビット, 回路段数は 4 段 (LUT 3 段 + 加算器 1 段) になる.　　(例終り)

本稿では, MTBDD および EVBDD を用いた区間指定回路 (図 6(b) と (c)) をそれぞれ **MT_SIE**, **EV_SIE** と表記する.

MT_SIE だけでなく, EV_SIE も, 次の定理が示すように, 任意の $seg\_func(X)$ を区間数に依存するサイズで実現できる.

［定理 1］　区間数を t としたとき, 任意の $seg\_func(X)$ は, レール数および A レール数が高々 $\lceil\log_2 t\rceil$ の EV_SIE で実現できる.(注2)

EV_SIE のメモリ量や段数は, EVBDD の分解法に依存する. 目的に応じた最適な分解を見つけるために, 本合成システムでは, [19], [20] で提案された分解法を応用する.

表 1　二次多項式近似に基づく様々な区間分割法における区間数の比較.

関数 $f(X)$	定義域 $[A,B)$	等区間数	不等区間数	再帰的分割 区間数 1	再帰的分割 区間数 2	再帰的分割 時間 [msec]
e^X	$[0,1)$	128	67	103	128*	10
$\sin(\pi X)$	$[0,0.5)$	128	74	112	128*	10
$\tan(\pi X)$	$[0,0.5)$	4,194,303	4,594	5,723	8,192	1,600
$\arcsin(X)$	$[0,1)$	8,388,607	256	363	512	70
$\sqrt{X}$	$(0,1)$	8,388,606	228	322	512	30
$\sqrt{-\ln(X)}$	$(0,1)$	8,388,606	698	967	1,024	190
$X\ln(X)$	$(0,1)$	2,097,152	172	250	256	10

*は等区間分割である.

実験環境　　許容近似誤差: 2^{-25}　　X は 23 ビット確度.

Sub Blade 2500 (Silver)　　CPU: UltraSPARC-IIIi 1.6GHz

メモリ: 6GB　　OS: Solaris 9　　C コンパイラ: gcc -O2

4.2　乗算器のサイズ削減

現在の FPGA は, 高速な乗算器を備えているが, 乗算器のサイズが大きくなるとそれに伴い遅延時間も大きくなるため, 高速な数値計算回路を得るためには, 乗算器のサイズ削減が重要である. 乗算器のサイズは, 係数および $(X-S_i)$ のビット数に依存するため, 本節では, それらのビット数を削減する手法を述べる.

係数のビット数を削減するために, 本合成システムは, スケーリング手法 [14] を用いる. 係数の値を右シフトによって小さくしてから係数表に格納し, 乗算器で計算した後, 左シフトでもとの値の大きさに戻す. スケーリング手法は, 乗算器のサイズ削減に有効であるが, 用いない手法に比べ, 丸め誤差が大きくなる. 本合成システムでは, 誤差解析によって, 許容誤差以内で最大のシフト量を算出する. 算出した結果, 全ての区間において, シフト量が 0 ならば, スケーリング手法を実装しない.

$(X-S_i)$ のビット数は, べき乗回路と乗算器のサイズに影響するため, $(X-S_i)$ の値の範囲の削減は, べき乗回路と乗算器のサイズを削減する. また, $(X-S_i)$ の値の範囲の削減は, 誤差も削減できる [10]. 各区間 $[S_i,E_i)$ において, $(X-S_i)<(E_i-S_i)$ であるため, 区間の幅 (E_i-S_i) の削減は, $(X-S_i)$ の値の範囲を削減することがわかる. しかし, 区間の幅の削減は, 区間数の増加につながり, 結果的にメモリ量の増加につながる. そこで, 本節でメモリ量を増加させずに区間の幅を削減する方法を示す.

図 5 の係数表は, 2^u ワードの ROM で実現される. ただし t を区間数としたとき, $u=\lceil\log_2 t\rceil$ である. これは, 区間数を 2^u まで増加させても実装される ROM のサイズは増加しないことを示している. また, EV_SIE においても, 定理 1 より, そのサイズは, u に依存するため, 区間数を 2^u に増加しても EV_SIE のサイズに大きな変化はない. 本合成システムでは, 区間数が 2^u になるまで, 区間幅の大きな順に区間を二等分割し続けることで, 区間幅の最大値を削減する. この区間幅の削減は, メモリ量を増やすことなく, ビット数と誤差の両方を削減できる.

5.　実 験 結 果

5.1　区間数と分割アルゴリズムの計算時間

表 1 は, いくつかの数値関数に, 二次の Chebyshev 多項式近似に基づく様々な区間分割法を適用したときの区間数を比較している. 表 1 で, "不等区間数" は, [21] で提案された不等区間分割

(注2)：証明は, ページ数制限のために割愛する.

表 2　区間指定回路の FPGA 実装結果.

FPGA デバイス:	Altera Stratix EP1S10F484C5 (LE: 10,570 個, M4K: 60 個, M512: 90 個)											
論理合成ツール:	Altera QuartusII 5.0 (速度最適化, タイミング制約: 200MHz)											
関数	MT_SIE (不等区間分割)				MT_SIE (再帰的分割)				EV_SIE (再帰的分割)			
$f(X)$	メモリ量 [ビット]	LE	段数	遅延 [nsec]	メモリ量 [ビット]	LE	段数	遅延 [nsec]	メモリ量 [ビット]	LE	段数	遅延 [nsec]
e^X	26,368	73	8	27.5	0	0	0	0	0	0	0	0
$\sin(\pi X)$	26,880	71	8	27.5	0	0	0	0	0	0	0	0
$\tan(\pi X)$	1,802,240	–	5	–	1,687,552	–	5	–	15,108	142	7	24.1
$\arcsin(X)$	61,440	72	8	27.5	49,152	72	7	24.3	9,984	88	5	17.2
$\sqrt{X}$	61,440	72	8	27.5	44,544	71	7	24.1	9,216	91	5	17.2
$\sqrt{-\ln(X)}$	266,240	81	7	33.2	172,032	71	7	28.1	12,736	109	6	20.6
$X\ln(X)$	61,440	79	8	27.5	20,992	49	5	17.2	6,912	65	4	13.8

表中の “–” は組込み RAM が不足し実装できなかったことを示す.

表 3　24 ビット精度 (23 ビット確度) 数値計算回路の FPGA 実装結果.

FPGA デバイス:	Altera Stratix EP1S60F1020C5 (LE: 57,120 個, DSP: 144 個, M4K: 292 個, M512: 574 個)									
論理合成ツール:	Altera QuartusII 5.0 (速度最適化, タイミング制約: 200MHz)									
関数	不等区間分割に基づく MTNFG					再帰的分割に基づく EVNFG				
$f(X)$	メモリ量 [ビット]	LE	DSP	段数	遅延 [nsec]	メモリ量 [ビット]	LE	DSP	段数	遅延 [nsec]
e^X	39,040	689	10	13	99.6	8,064	432	10	3	25.1
$\sin(\pi X)$	36,864	635	10	13	99.1	7,936	395	10	3	28.3
$\tan(\pi X)$	2,867,200	–	16	11	–	973,572	1,059	16	12	92.3
$\arcsin(X)$	84,736	1,301	16	14	107.3	53,504	937	16	10	80.3
$\sqrt{X}$	83,712	1,041	16	14	116.5	52,224	962	16	10	85.5
$\sqrt{-\ln(X)}$	357,376	950	16	13	99.8	103,872	972	16	11	88.3
$X\ln(X)$	83,200	988	16	14	116.0	29,696	997	16	9	70.3

表中の “–” は組込み RAM が不足し実装できなかったことを示す.

法によって得られた区間数を表し, “再帰的分割” は, 3. 節で示した再帰的分割法を表す. “再帰的分割” の欄で, “区間数 1” は, 3. 節の再帰的分割法だけを適用して得られた区間数を表し, “区間数 2” は, 4. 節の乗算器のサイズ削減法を適用した後の区間数を表す. “時間 [msec]” は, 再帰的分割法と乗算器のサイズ削減法の両方を適用したときの CPU 時間をミリ秒単位で表している.

表 1 から, ほとんどの従来手法で用いられている等区間分割法は, $\tan(\pi X)$ などの関数には, 実用的でないことがわかる. 一方, 不等区間分割法と階層的分割法は, 様々な関数において実用的な区間数で関数を近似できる. [21] で提案した不等区間分割法が三つの手法のうちで最も少ない区間数で関数を近似できる. しかし, 本階層的分割法は分割点を制限しても, 表中のすべての関数において区間数が [21] の高々2.2 倍に増加しただけだった. すなわち, 本階層的分割法は分割点を制限しながらも, 関数に適した分割を高速に自動生成できる. 特に, 関数 e^X および $\sin(\pi X)$ において, 本階層的分割法は等区間分割を生成する. [21], [26] で示したように, これらの関数においては, 不等区間分割より等区間分割が適している. 以上の結果より, 本分割アルゴリズムは, 関数に適した分割を高速に生成でき, 関数によっては, 分割法を変更することなく等区間分割さえ生成できる.

5.2　区間指定回路の FPGA 実装結果

表 2 は, MT_SIE と EV_SIE の FPGA 実装結果を比較している. この実験では, MT_SIE および EV_SIE の各 LUT のサイズが FPGA 上のメモリブロック M4K のサイズになるように, MTBDD および EVBDD を分解した. 表中の “MT_SIE (不等区間分割)” は, 表 1 で得られた不等区間分割を MT_SIE で実現した結果, “MT_SIE (再帰的分割)” は再帰的分割を MT_SIE で実現した結果, そして “EV_SIE (再帰的分割)” は再帰的分割を EV_SIE で実現した結果を表す. 表 1 で示したように, e^X と $\sin(\pi X)$ では, 再帰的分割法が等区間分割を生成するため, 区間指定回路が不要になる. そのため, 区間指定回路のメモリ量や遅延時間は 0 になる.

表 2 から, 再帰的分割を用いることで, MT_SIE のメモリ量と遅延時間の両方を削減できることがわかる. 特に, 関数 $X\ln(X)$ において, 再帰的分割を用いた場合, MT_SIE のメモリ量は, 不等区間分割を用いた場合の 34%, そして遅延時間は, 63% に削減できる. そして, EV_SIE を用いることで, メモリ量と遅延時間をさらに削減できる. 表 2 のすべての関数において, EV_SIE のメモリ量と遅延時間は, MT_SIE より遙かに小さい. 例えば, 関数 $\tan(\pi X)$ において, EV_SIE のメモリ量は, 不等区間分割を実現する MT_SIE のメモリ量のわずか 0.8% である. $\tan(\pi X)$ は, 不等区間数が大きいので, MT_SIE のメモリ量も大きくなる. $\tan(\pi X)$ において, MT_SIE を用いた数値計算回路のメモリ量は, 等区間分割に基づく数値計算回路のメモリ量の約 1.5% であるが, FPGA で実装するにはまだ大きすぎる. しかし, 再帰的分割と EV_SIE を用いることにより FPGA で十分に実装できるほど, メモリ量を大幅に削減できる. 論理素子では, EV_SIE は, MT_SIE の高々1.5 倍に増加するだけなので, EV_SIE は, MT_SIE よりコンパクトかつ高速であるといえる.

5.3　数値計算回路の FPGA 実装結果

表 3 は, MT_SIE を用いた数値計算回路 [21] (MTNFG) と EV_SIE を用いた数値計算回路 (EVNFG) の FPGA 実装結果を比較している. ただし, MTNFG は不等区間分割に基づき, EVNFG は再帰的分割に基づいている. どちらの数値計算回路も 24 ビット精度 (23 ビット確度) である.

表 2 と表 3 から, MT_SIE のメモリ量は, 数値計算回路全体のメモリ量の 3 分の 2 以上を占めていることがわかる. 一方, 再帰的分割と EV_SIE を用いることにより, 区間指定回路に必要なメモリ量を数値計算回路全体の 4 分の 1 以下に削減できる. そのため, EVNFG のメモリ量は, MTNFG の 21% ～ 63% に削減できる. 特に, 関数 $\arcsin(X)$ と $\sqrt{X}$ においては, 表 1 に示されているように, 再帰的分割は, 不等区間分割に比べ約 2 倍の係数表が必要になるにも関わらず, EV_SIE を用いることにより, 数値計算回路全体のメモリ量を MTNFG の 63% に削減できる.

また, 表 3 で示されているように, EVNFG は, LE 数や遅延時間についても MTNFG より小さい. 実験で使用した関数において, EVNFG の遅延時間は, MTNFG の約 25% ～ 89% だった.

本数値計算回路 EVNFG を [15] の階層的分割に基づく数値計算回路と比較するために, 関数 $X\ln(X)$ の 24 ビット精度の数値計算回路 EVNFG を Xilinx Virtex-II FPGA (XC2V4000-6) に実装した. 論理合成ツール Synplify Premier 8.5 を用いて実装結果を得た. [15] で提案された数値計算回路のメモリ量は 40, 446 ビット, 遅延時間は 103.7 nsec であるのに対し, 本数値計算回路のメモリ量と遅延時間は, 30, 976 ビット (77%) および 63.3 nsec (61%) だった.

以上の結果から, 再帰的分割と EV_SIE を用いた本手法は, 様々な数値関数において, 従来手法よりコンパクトかつ高速な数値計算回路を生成できる.

6. 結論とコメント

本稿は, 三角関数, 対数関数, 平方根演算, 逆数演算などの初等関数を計算する数値計算回路の構成法を提案した. 与えられた定義域を再帰的分割法で不等区間に分割し, その不等区間分割の実現に EVBDD を用いることにより, 様々な関数をコンパクトかつ高速に実現できる. 実験により以下を示した: 1) MTBDD を用いた区間指定回路は, 再帰的分割法を用いることでメモリ量を 49% まで, そしてなおかつ, 遅延時間を 53% まで削減できた. 2) 区間指定回路に EVBDD を用いることでメモリ量と遅延時間をさらに削減でき, MTBDD を用いた従来手法に比べ, メモリ量を 7% に, そして遅延時間を 40% まで削減できた. 3) 区間指定回路で得られたメモリ量の削減と高速化の結果, 数値計算回路のメモリ量と遅延時間は, 従来手法の 38% と 59% になった.

謝　　辞

本研究の一部は, 平成 18 年度科学研究費補助金 (若手研究 (B)) 課題番号 18700048 および平成 18 年度広島市立大学教員特定研究費 (一般研究) 課題番号 6101 による.

文　　献

[1] ALTERA, http://www.altera.com/

[2] R. Andrata, "A survey of CORDIC algorithms for FPGA based computers," *Proc. of the 1998 ACM/SIGDA Sixth Inter. Symp. on Field Programmable Gate Array (FPGA'98)*, pp. 191–200, Monterey, CA, Feb. 1998.

[3] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. Comput.*, Vol. C-35, No. 8, pp. 677–691, Aug. 1986.

[4] A. Cantoni, "Optimal curve fitting with piecewise linear functions," *IEEE Trans. on Comp.*, Vol. 20, No. 1, pp. 59–67, Jan. 1971.

[5] J. Cao, B. W. Y. Wei, and J. Cheng, "High-performance architectures for elementary function generation," *Proc. of the 15th IEEE Symp. on Computer Arithmetic (ARITH'01)*, Vail, Co, pp. 136–144, June 2001.

[6] E. M. Clarke, K. L. McMillan, X. Zhao, M. Fujita, and J. Yang, "Spectral transforms for large Boolean functions with applications to technology mapping," *Proc. of 30th ACM/IEEE Design Automation Conference*, pp. 54–60, June 1993.

[7] D. Defour, F. de Dinechin, and J.-M. Muller, "A new scheme for table-based evaluation of functions," *36th Asilomar Conference on Signals, Systems, and Computers,*, Pacific Grove, California, pp. 1608–1613, Nov. 2002.

[8] J. Detrey and F. de Dinechin, "Second order function approximation using a single multiplication on FPGAs," *Proc. Inter. Conf. on Field Programmable Logic and Applications (FPL'04)*, pp. 221–230, 2004.

[9] J. Detrey and F. de Dinechin, "Table-based polynomials for fast hardware function evaluation," *16th IEEE Inter. Conf. on Application-Specific Systems, Architectures, and Processors (ASAP'05)*, pp. 328–333, 2005.

[10] N. Doi, T. Horiyama, M. Nakanishi, and S. Kimura, "Minimization of fractional wordlength on fixed-point conversion for high-level synthesis," *Proc. of Asia and South Pacific Design Automation Conference (ASPDAC'04)*, pp. 80–85, 2004.

[11] Y. Iguchi, T. Sasao, and M. Matsuura, "Realization of multiple-output functions by reconfigurable cascades," *International Conference on Computer Design: VLSI in Computers and Processors (ICCD'01)*, Austin, TX, pp. 388–393, Sept. 23–26, 2001.

[12] Y-T. Lai and S. Sastry, "Edge-valued binary decision diagrams for multi-level hierarchical verification," *Proc. of 29th ACM/IEEE Design Automation Conference*, pp. 608–613, 1992.

[13] Y-T. Lai, M. Pedram, and S. B. Vrudhula, "EVBDD-based algorithms for linear integer programming, spectral transformation and functional decomposition," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, Vol. 13, No. 8, pp. 959–975, Aug. 1994.

[14] D.-U. Lee, W. Luk, J. Villasenor, and P. Y.K. Cheung, "Non-uniform segmentation for hardware function evaluation," *Proc. Inter. Conf. on Field Programmable Logic and Applications*, pp. 796–807, Lisbon, Portugal, Sept. 2003.

[15] D.-U. Lee, W. Luk, J. Villasenor, and P. Y. K. Cheung, "Hierarchical segmentation schemes for function evaluation," *Proc. of the IEEE Conf. on Field-Programmable Technology*, Tokyo, Japan, pp. 92–99, Dec. 2003.

[16] D.-U. Lee, W. Luk, J. Villasenor, and P. Y. K. Cheung, "A Gaussian noise generator for hardware-based simulations," *IEEE Trans. on Comp.*, Vol. 53, No. 12, pp. 1523–1534, Dec. 2004.

[17] J. H. Mathews, *Numerical Methods for Computer Science, Engineering and Methematics*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1987.

[18] J.-M. Muller, *Elementary Function: Algorithms and Implementation*, Birkhauser Boston, Inc., Secaucus, NJ, 1997.

[19] S. Nagayama and T. Sasao, "Compact representations of logic functions using heterogeneous MDDs," *IEICE Trans. on fundamentals*, Vol. E86-A, No. 12, pp. 3168–3175, Dec. 2003.

[20] S. Nagayama and T. Sasao, "On the optimization of heterogeneous MDDs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, Vol. 24, No. 11, pp. 1645–1659, Nov. 2005.

[21] S. Nagayama, T. Sasao, and J. T. Butler, "Programmable numerical function generators based on quadratic approximation: architecture and synthesis method," *Proc. of Asia and South Pacific Design Automation Conference (ASPDAC'06)*, Yokohama, Japan, pp. 378–383, 2006.

[22] J.-A. Piñeiro, S. F. Oberman, J.-M. Muller, and J. D. Bruguera, "High-speed function approximation using a minimax quadratic interpolator," *IEEE Trans. on Comp.*, Vol. 54, No. 3, pp. 304–318, Mar. 2005.

[23] T. Sasao and M. Fujita (eds.), *Representations of Discrete Functions*, Kluwer Academic Publishers 1996.

[24] T. Sasao, M. Matsuura, and Y. Iguchi, "A cascade realization of multiple-output function for reconfigurable hardware," *Inter. Workshop on Logic Synthesis (IWLS'01)*, Lake Tahoe, CA, pp. 225–230, June 12–15, 2001.

[25] T. Sasao, J. T. Butler, and M. D. Riedel, "Application of LUT cascades to numerical function generators," *Proc. the 12th workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI'04)*, Kanazawa, Japan, pp. 422–429, Oct. 2004.

[26] T. Sasao, S. Nagayama, and J. T. Butler, "Programmable numerical function generators: architectures and synthesis method," *Proc. Inter. Conf. on Field Programmable Logic and Applications (FPL'05)*, Tampere, Finland, pp. 118–123, Aug. 2005.

[27] Scilab 3.0, INRIA-ENPC, France, http://scilabsoft.inria.fr/

[28] M. J. Schulte and J. E. Stine, "Symmetric bipartite tables for accurate function approximation," *13th IEEE Symp. on Comput. Arith.*, Asilomar, CA, Vol. 48, No. 9, pp. 175–183, 1997.

[29] M. J. Schulte and J. E. Stine, "Approximating elementary functions with symmetric bipartite tables," *IEEE Trans. on Comp.*, Vol. 48, No. 8, pp. 842–847, Aug. 1999.

[30] J. E. Stine and M. J. Schulte, "The symmetric table addition method for accurate function approximation," *Jour. of VLSI Signal Processing*, Vol. 21, No. 2, pp. 167–177, June 1999.

[31] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electronic Comput.*, Vol. EC-820, No. 3, pp. 330–334, Sept. 1959.

[32] Xilinx, http://www.xilinx.com/