決定グラフに基づく論理関数の評価システム

井口　幸洋† 笹尾　勤†† 松浦　宗寛††† 伊勢野　総†

An Evaluation System for Logic Functions Based on Decision Diagrams

Yukihiro IGUCHI†, Tsutomu SASAO††, Munehiro MATSUURA†††, and Atsumu ISENO†

あらまし　多出力論理関数の評価を決定グラフに基づくハードウェアで行うシステムを提案する．また，データ構造として，PMDD（Paged reduced ordered Multi-valued Decision Diagram）を提案する．論理関数をハードウェアで評価する方法としては，SBDD（Shared reduced ordered Binary Decision Diagram）に基づく方法が知られている．PMDD に基づく方法は，複数の計算ユニットを用いて多出力関数を評価するため，SBDD に基づく方法に比べ高速である．実現すべき多出力論理関数を PMDD で表現し，メモリに格納する．これに PMDD を探索する制御回路を付け加える．多数のベンチマーク関数を，SBDD と PMDD で実現し必要なメモリ量と計算時間を比較する．

キーワード　多出力論理関数，BDD，MDD，PMDD

1. まえがき

組込機器等で用いる多出力の論理関数を実現するには，種々の方法がある．専用の多段組合せ論理関数を実現する LSI を開発するには，設計費用や時間がかかりすぎる．ROM（Read Only Memory）や PLA（Programmable Logic Array）を用いて直接組合せ回路で実現する方法もある [17]．しかし，この方法は入力変数の個数 n が大きいとき，回路が大きくなりすぎる．出力値評価に時間がかかってもよい場合には，汎用マイクロプロセッサを利用できる．しかし，ソフトウェアでの実現は，専用の組合せ論理回路で実現する場合に比べ，2 けたから 3 けた遅くなる．

本論文では，専用の順序回路を用いて，多出力の論理関数を実現する方法について述べる．この場合，関数の評価をなるべく短時間に実行することを目標とする．まず，簡単のために n 変数の 2 値論理関数 $B^n \to B$ を考える．これを BDD（Binary Decision Diagram）[3], [9] で表現し，各非終端節点を If then else 文で置き換えると，f を実現する Branching Program を構成できる [1]．これを汎用マイクロプロセッサで実行することで論理関数が実現できる．この場合，$O(n)$ の計算時間が必要である．命令フェッチの時間を省略するため，BDD のデータ構造を直接探索するハードウェアを構成することもできる [6], [18]．この場合，ROM の大きさは，BDD の節点数に比例する．次に，2 値論理関数の各変数を k ビットずつグループ化し，多値入力 2 値出力関数に変換し，それに対する MDD（Multi-valued Decision Diagram）[2], [11], [15], [18] を構成する．そして，MDD のデータ構造を直接探索するハードウェアを構成することにより，BDD の場合に比べ評価速度を k 倍にできる．

多くの実用回路は多出力である．出力数を m とすると，m 個の出力関数を評価するには，SMDD（Shared Multi-valued Decision Diagram）を m 回探索する必要がある．本論文では，多出力関数の評価時間を短縮するため，SMDD を r 個に分割した PMDD（Paged reduced ordered Multi-valued Decision Diagram）を提案する．PMDD では，r 個のモジュールが同時に関数の評価を行うので，単一の SMDD を用いて多出力関数を評価する場合よりも r 倍高速にできる．

本論文の構成は次のとおり．まず，**2.** で BDD と

† 明治大学理工学部情報科学科，川崎市
Dept. of Computer Science, Meiji Univ., Kawasaki-shi, 214–8571 Japan
†† 九州工業大学情報工学部電子情報工学科・マイクロ化総合技術センタ，飯塚市
Dept. of Computer Science and Electronics, and Center for Microelectronic Systems, Kyushu Institute of Technology, Iizuka-shi, 820–8502 Japan
††† 九州工業大学情報工学部電子情報工学科，飯塚市
Dept. of Computer Science and Electronics, Kyushu Institute of Technology, Iizuka-shi, 820–8502 Japan

MDD について概説し，本評価システムでデータ構造として用いる PMDD を提案する．**3.** では，PMDD に基づきハードウェアで論理関数を評価する方法を提案する．最後に，**4.** で性能評価を行い，評価システムの試作に基づく実験結果を示し，考察を行う．

2. MDD と PMDD

ここより，簡単のために SBDD（Shared reduced ordered Binary Decision Diagram）[7], [12] を BDD [3] と表記する．BDD では，各非終端節点は変数に対応し，枝に付加された数字 0 は $low(v)$ を，1 は $high(v)$ を意味している．ここでは，根から終端節点までのすべての経路において入力変数の順番は同じである決定グラフを取り扱う．

多出力関数を表現するためには，SMDD（Shared reduced ordered Multi-valued Decision Diagram）を用いることもできる．今後，SMDD を簡単のために MDD と表記する．

［定義 1］ MDD(k) は，各非終端節点が 2^k 個の枝をもつ多値決定グラフ（Multi-valued Decision Diagram）である．MDD(1) と BDD とは同じである．

MDD(k) は，k 個の 2 値変数を同時に評価するので，MDD(k) に基づく方法は，BDD に基づく方法に比べ k 倍高速である．MDD(k) は，対応する BDD から簡単に求められる [10].

簡単のため，2 値ベクトルを，対応する 10 進数で表記することもある．例えば，$(0,0)$ を 0, $(0,1)$ を 1, $(1,0)$ を 2, $(1,1)$ を 3 で表す．

［例 1］ 図 1 (a) に 8 入力 2 出力論理関数の BDD（MDD(1)）の例を示す．入力変数を (X_1, X_2, X_3, X_4) に分割し，$X_1 = (x_1,\ x_2)$, $X_2 = (x_3,\ x_4)$, $X_3 = (x_5,\ x_6)$, $X_4 = (x_7,\ x_8)$ とすると図 1 (b) に示す MDD(2) が得られる．MDD(2) を用いると，根から終端節点までの経路長が MDD(1) の半分になる（経路によっては必ずしも半分にならないことに注意）． □

次に，MDD(k) を r ページに区切った PMDD(k, r)（Paged reduced ordered Multi-valued Decision Diagram）を提案する．

［定義 2］ Paged reduced ordered MDD: PMDD(k, r)

入力変数を $X = (x_1, x_2, \ldots, x_n)$ とする，ただし，$n = t \cdot r$．根から終端節点までのどの経路においても，第 $t \cdot s + 1$ レベル $(s = 1, \ldots, r-1)$ には，必ず非終端節点が存在するような MDD を考える．第 1 ページは，レベル 1 からレベル t で構成され，第 2 ページはレベル $t+1$ からレベル $2t$ で構成される．残りのページについても同様．

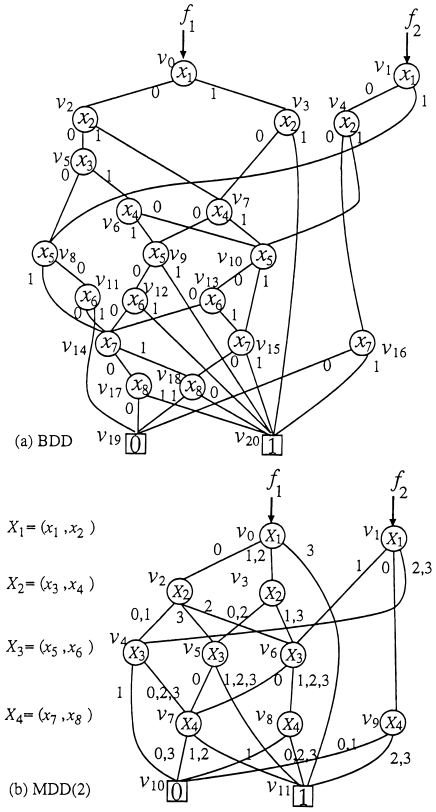

図 1 BDD と MDD(2)
Fig. 1 BDD and MDD(2).

［例 2］ 図 1 (b) の MDD(2) を 2 ページに分割すると，図 2 に示す PMDD(2, 2) を得る．この例では，x_1, x_2, x_3, x_4 を第 1 ページに，x_5, x_6, x_7, x_8 を第 2 ページに割り当てた．いくつかの非終端節点がページの境界上に付け加えられていることに注意． □

PMDD は，次の性質をもつ．

（1） 一つのページ内では，第 1 レベルにないどの節点も異なる関数を表現する（異なるページ内にある二つの節点は，同じ関数を表現してもよい）．

（2） ある節点から出ている枝は，そのページ内の節点に接続しているか，または，次ページの第 1 レベルの節点に接続している．

PMDD(k, r) は，MDD(k) を r ページに分割したものである．PMDD(1, 1) は ROBDD であ

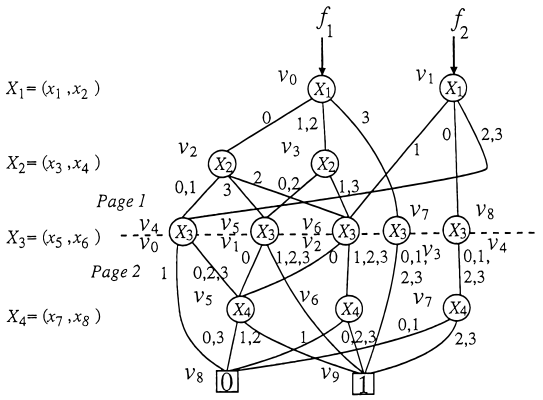

図 2　図 1 (b) の関数に対する PMDD(2, 2)
Fig. 2　PMDD(2, 2) for the function in Fig. 1 (b).

り，PMDD$(1, n)$ は QROBDD（Quasi-Reduced Ordered BDD）であることに注意．QROBDD では，根から定数ノードに至るすべての経路にそってすべての変数が出現する [16]．PMDD(k, r) は，r ページから構成された MDD(k) であり，各ページの第 1 レベルには必ず節点が存在する．

f を決定グラフ (DD) で表したとき，DD の節点数を $size(DD, f)$ とすると次の性質を得る．

［性質 1］

$size(PMDD(1,1), f) \leqq size(PMDD(1,r), f)$
$\leqq size(PMDD(1,n), f)$，ただし，$1 \leqq r \leqq n$.

3. 論理関数のハードウェアによる評価法

3.1 評価システムの動作

図 3 に 1 個の計算ユニットで論理関数の評価を行うシステムを示す．$x_1, \ldots, x_n$ は，入力ベクトル，$s_1, \ldots, s_{lw_1}$ は，何番目の出力を計算するかを指示する入力とする．提案するシステムは，メモリと制御回路とで構成される．評価対象の論理関数はあらかじめ MDD で表現し，システム内のメモリ内に格納する．論理関数を頻繁に書き換えたい場合は，メモリに RAM を用いればよい．また，それほど頻繁に変更する必要のない場合は，書換え可能な ROM（Flush ROM，EEPROM）を，また，固定の場合は ROM を用いればよい．これらは，アプリケーションに応じて決めればよい．

評価したい入力ベクトルをシステムに加える．システムは入力ベクトルに従って MDD をたどり，関数の評価値を返す．BDD を用いた場合，m 出力関数に対

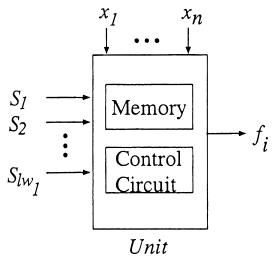

図 3　計算ユニット
Fig. 3　Single-unit evaluator.

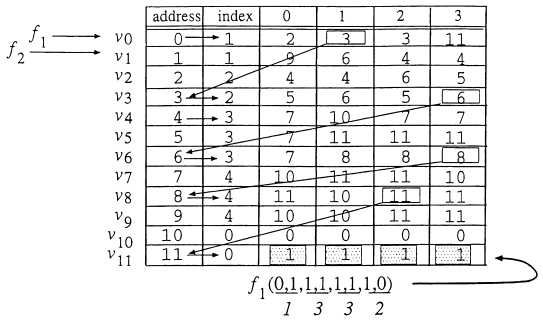

図 4　図 1 (b) に対する MDD(2) と計算の手順
Fig. 4　MDD(2) data for Fig. 1 (b) and evaluation procedure.

しては，BDD の枝を m 回たどらなければならない．したがって，図 3 に示す評価システムでは，$O(n \cdot m)$ 回のメモリアクセスが必要になる．

3.2 評価の高速化

本節では，関数評価の高速化のために二つの方法を示す．まず，MDD(k) を用いることで BDD よりも k 倍高速化する．次に，r 台の計算ユニットを用いて，r 倍高速化する．

3.2.1 MDD(k)

MDD(k) に対するデータは，BDD の場合と同様にメモリ内に格納する．MDD(k) 内のメモリアクセスは，$1/k$ に削減されるので，BDD に基づく場合より k 倍高速化できる．しかし，MDD(k) の各節点を表現するために必要なメモリは，対応する BDD の各節点を表現する場合より多くなる．

［例 3］ 図 4 は，図 1 (b) に示した MDD(2) のデータをメモリ上に表現した例である．f_1 の先頭の節点に対するデータは v_0 に格納されているとする，同様に f_2 の先頭の節点に対するデータは v_1 に格納されているとする．まず，f_1 の値を求めるために 0 番地を探索

開始アドレスとする．v_0 の index の値は 1 であるので，$X_1 = (x_1, x_2)$ の値を読み出す．$(x_1, x_2) = (0, 1)$ であるから X_1 の値は 1 であり，これは 1 枝を表しているので 1 枝の値を読み出す．値は 3 であるので，v_3 に進む．同様の操作を終端節点を表す $index = 0$ に到達するまで繰り返す．$index = 0$ に到達するとその 0 枝（1 枝，2 枝，3 枝のいずれでも可）の値 1 を読み出し f_1 $(0, 1, 1, 1, 1, 1, 1, 0) = 1$ が求まる．

同様に 1 番地からメモリをたどることで f_2 が求められる．図 1 (b) で示すように，MDD(2) は 10 個の非終端節点をもつ，一方，BDD = MDD(1) は，19 個の非終端節点となる．しかし，MDD(k) の各非終端節点は，2^k 個の次アドレスを必要とする． □

3.2.2 パイプライン化による高速化

決定グラフを格納するためのメモリシステムを 1 個のみとすると，一度に一つの入力ベクトルに対してのみしか評価ができない．そこで，PMDD(k, r) を評価システム内のデータ構造とし，1 ページに対し，1 個の評価ユニットを割り当てたパイプライン化した評価システムを提案する．提案システムの概念図を図 5 に示す．

これらのユニットは並列に動作するので，r 個の計算ユニットからなる評価システムは最大 r 倍のスループットとなる．また，PMDD の各ページ内の節点数は，1 個の MDD(k) に比べて少ないので，インデックスと次アドレスを記憶するためのメモリ量を削減できる．

［定義 3］ 評価システムの主要な用語とパラメータを以下のように定める．

- ページ：t 個の外部入力変数に対応するメモリ領域．
- ユニット：1 ページを処理するメモリと制御回路とで構成されるハードウェア．
- パイプライン：r 個のユニットを直列に接続したもの．

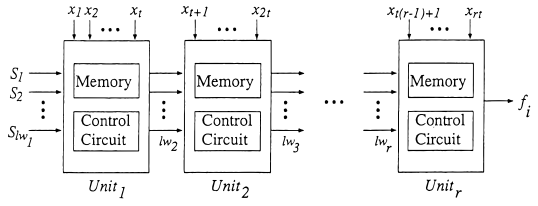

図 5 パイプライン化システムの概念図
Fig. 5 Concept of the pipelined system.

- n：入力変数の個数．
- m：出力の個数．
- r：ページ数 = ユニットの個数 = パイプラインの段数．
- t：1 ページ内の変数の個数．
- w_i $(i = 1, 2, \ldots, r)$：MDD(k) の幅．ただし，$w_1 = m$ とする．
- k：MDD の制御変数の個数（BDD は $k = 1$）．

［例 4］ 図 2 の PMDD(2, 2) をメモリ上に表現した例を図 6 に示す．このとき，$(X_1, X_2, X_3, X_4) = (1, 3, 3, 2)$ に対する論理関数の値を求めてみる．

- 時刻 1：$f_1(X_1, X_2, X_3, X_4) = f_1(1, 3, 3, 2)$，まず，1 ページ目で $f_1(X_1, X_2) = f_1(1, 3)$ を求める．v_0 の 1 枝の欄の 3 を読む．v_3 に進む．$index = 2$ であるので，$X_2 = 3$ であるから 3 枝の欄の 6 を読み出し，v_6 に進む．$index = 0$ であるので，0 枝の欄の 2 を読み出す．次ページの探索開始番地を v_2 と決定し，2 ページ目のユニットに渡す．
- 時刻 2：2 ページ目のユニットは，1 ページ目から受け取った v_2 からたどり始める．$f_1(X_3, X_4) = f_1(3, 2)$ に対応して v_2 の 3 枝の欄の 6 を読み出し，v_6 に進む．$index = 4$ であるので，$X_4 = 2$ であるから 2 枝の欄の 9 を読み出し，v_9 に進む．$index = 0$ であるので，0 枝の欄の 1 を読み出す．これは終端節点 1 を意味し

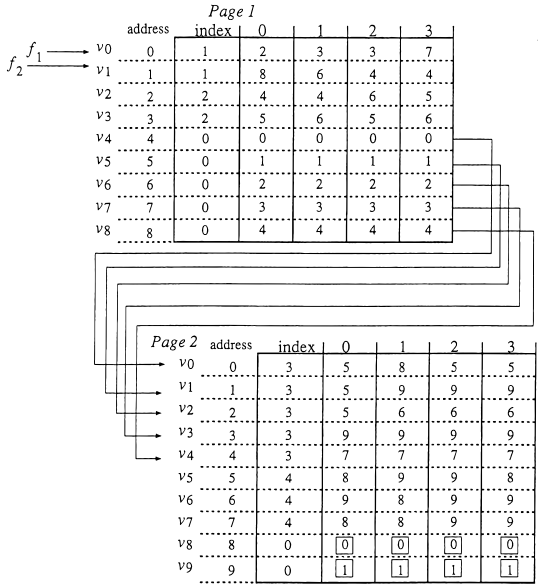

図 6 図 2 に示した PMDD(2, 2) のメモリ表現
Fig. 6 Memory representation of PMDD(2, 2) shown in Fig. 2.

ており，出力 $f_1(X_1, X_2, X_3, X_4) = f_1(1, 3, 3, 2) = 1$ が求まる．

一方，このとき1ページ目では同時に $f_2(X_1, X_2) = f_2(1, 3)$ を評価する．f_1 の場合と同様に探索を行い，次ページの探索を開始する番地を v_2 と決定する．これを，2ページ目のユニットに渡す．

- 時刻3:2ページ目のユニットは1ページ目から受け取った v_2 より探索を開始し，$f_2(X_1, X_2, X_3, X_4) = f_2(1, 3, 3, 2) = 1$ が求まる． □

つまり，本提案システムにおける PMDD(k, r) の利用は，次の特長をもつ．

（1） パイプライン化に適している．

（2） ジャンプ先のアドレスがページ内に限定できるため，アドレスのビット長，インデックスのビット長を短くできる．

PMDD(k, r) を作るために次の戦略を用いる．

- ユニット内の変数の個数がほぼ同じになるように MDD(k) のページを分割する．これはパイプライン化システムのサイクルタイムを削減する．
- 各ページがユニット内のメモリに格納できる大きさに MDD(k) のページを分割する．例えば，ページの境界を幅が小さなところで区切る．もし，必要ならば，MDD(k) の変数順序を変える．

PMDD$(1, 1)$ に基づいた非パイプラインのシステムは，p 個の入力ベクトルを評価するのに $n \cdot p$ ステップ必要である．PMDD(k, r) に基づいたパイプライン化システムは，p 個の入力ベクトルを評価するのに $(n \cdot p/k)$ ステップ必要であり，PMDD$(k, 1)$ の r 倍のスループットである．各 $n/(k \cdot r)$ ステップごとに，評価結果が出力される．したがって，PMDD(k, r) のスピード向上比は，PMDD$(1, 1)$ の $k \cdot r$ となる．

3.3 順序回路の評価法

図7のような組合せ回路とフリップフロップからなる順序回路の組合せ回路部分を本方式で評価する場合を考える．ここで組合せ回路は，$(n+s)$ 入力，$(m+s)$ 出力で，s 個の D-FF で状態変数を表す．組合せ回路部の評価には，同じ入力ベクトルを多数回加えなければならない．

多数の異なる入力ベクトルに対する評価を考える．現在評価中のベクトルを $(m+s)$ 回加え，次の入力ベクトルの評価を開始するとき，状態変数の値がすべて更新されていないとパイプラインハザードを生じる．したがって，状態変数 s の評価を先に行った方が有利となる．

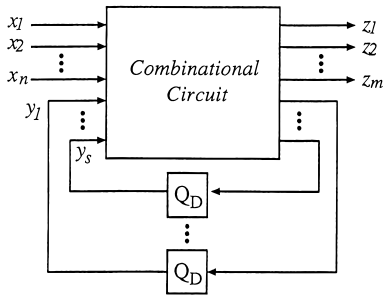

図7 順序回路のモデル
Fig. 7 A model of a sequential network.

［アルゴリズム 1］（順序回路の評価システム）

（1） $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ を入力ベクトルとする．

（2） $(m+s)$ 個の出力に対し，f_i の値を求める際，s 個のフリップフロップにつながる出力を最初に計算し，次に残りの出力を計算する．

（3） すべての出力の評価終了後，D-FF の値を更新する．もし，次の入力ベクトルがある場合はステップ1へ，ない場合は終了．

4. 性能評価

4.1 PMDD(k, r) の表現に要するメモリ容量

ISCAS ベンチマーク回路を PMDD$(k, 1)$，$(k = 1, 2, 3, 4, 5)$ で表現した場合の節点数を表1に示す．例えば，C3540 は 50 入力 22 出力であり，PMDD$(k, 1)$ $(k = 1, 2, 3, 4, 5)$ は，それぞれ 34693，24482，19126，16845，及び，12093 個の非終端節点をもつことがわかる．表1は，k を増加させたとき，節点数がどのように減少するかを示している．ただし，各々の節点は 2^k 個のアドレスを必要とすることに注意．PMDD$(k, 1)$ に対する入力の変数順序を決定するには発見的アルゴリズム [14] を使用した．ここで，BDD を最小にする変数の順序が必ずしも PMDD$(k, 1)$ $(k \geqq 2)$ のサイズを最小にするとは限らないことに注意．C6288 は，16 ビット乗算器であるが，PMDD を作ることができなかったので表には載せていない．他のベンチマーク関数については，すべて PMDD を構成でき，それらの幅（つまり，境界上の節点数）は，5420 以下であった．

［定義 4］ $MT(k, r)[Words]$ を PMDD(k, r) を表すのに必要な総メモリ容量とする．PMDD(k, r) のすべてのページで最も大きい節点数を#max_nodes とする．ここでシステムのメモリサイズは 16 の倍数と仮定する．つまり，1 ワードを 16 ビットとする．各ペー

ジはすべて同じ大きさのメモリをもつと仮定する．

$MT(k,r)$ は，以下のように求められる．

- *index* に対しては，$\lceil\lceil\log_2\lceil n/k\rceil\rceil/16\rceil$ ワード必要．
- 次アドレスを表示するためのポインタに $\lceil\lceil\log_2(\#max\_nodes)\rceil/16\rceil$ ワード必要．
- 各ノードに対して，2^k 個のポインタが必要．
- 1ページ内の節点の総数は $2^{\lceil\log_2(\#max\_nodes)\rceil}$．
- 総ページ数は，r．

よって，総メモリ容量は次式で表せる．

$$\begin{aligned} MT(k,r) = (&\lceil\lceil\log_2\lceil n/k\rceil\rceil/16\rceil + 2^k \\ &\times \lceil\lceil\log_2(\#max\_nodes)\rceil/16\rceil) \\ &\times 2^{\lceil\log_2(\#max\_nodes)\rceil} \\ &\times r[words] \end{aligned} \quad (1)$$

PMDD(k,r) を用いると，PMDD$(1,1)$ に比べ $r\cdot k$ 倍の高速化が達成できる．ゆえに，コスト/パフォーマンス $CP(k,r)$ を次式のように定義できる．

$$CP(k,r) = \frac{MT(k,r)}{MT(1,1)\cdot k\cdot r} \quad (2)$$

PMDD をシステムのデータ構造として用いるとき，重要なパラメータは1ページ内のメモリサイズと総メモリ容量である．1ページ内の節点数は関数，変数順序，ページの区切り方，ページ数に依存する．

システムのハードウェア量を見積もるために，ベンチマーク関数に対する PMDD$(1,r)$ $(r = 1,2,4,8,16,32)$ のメモリ量を求めた．表 2 は，ベンチマーク関数に対する必要なメモリ量 $MT(1,r)$ である．例えば，C2670 を表現する PMDD$(1,2)$ では，第1ページに 1.5 キロワード，第2ページには 12 キロワード必要であった．本表より，r が増えるにつれ $MT(1,r)$ も増えることがわかる．

表1　PMDD$(k,1)$ のノード数
Table 1　Number of nodes in PMDD$(k,1)$.

Func.	In	Out	PMDD$(k,1)$				
			$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
C432	36	7	1069	619	542	326	380
C499	41	32	27845	13543	9029	6319	5797
C880	60	26	4152	3038	2550	1930	1735
C1908	33	25	7432	4426	2869	2379	1589
C2670	233	140	2841	2473	2059	1905	1760
C3540	50	22	34693	24482	19126	16845	12093
C5315	178	123	2554	2074	1725	1565	1428
C7552	207	108	2934	2393	2003	1744	1551

表 3 は，C2670 に対するメモリ量 $MT(k,r)$ $(k = 1,2,3,4,5,\ r = 1,2,4,8,16,32)$ を示している．本表より次の興味深い特性がわかる：例えば，PMDD$(2,8)$ に基づくシステムは PMDD$(1,8)$ より速いにもかかわらず，前者は後者に比べ少ないメモリ量で実現できる．

表 4 に C2670 に対する $CP(k,r)$ $(k = 1,2,3,4,5,\ r = 1,2,4,8,16,32)$ を示す．太字の数値はコスト/パフォーマンスが 1.0 より小さいことを示す．コスト/パフォーマンスは，$k=2$, 3, 及び，4 がよいことがわかる．表 4 より，MDD(k) を多ページに分割するのが効果的であることもわかる．ただし，実際には，クロックスキュー，伝搬遅延等のために，r をむやみに大きくするとコスト/パフォーマンスは悪くなるかもしれない．これらについては，試作によって評価をしなければならない．

4.2　評価システムの試作

PMDD$(3,2)$ に基づいた評価システムの試作を行っ

表2　総メモリ容量 $MT(1,r)$
Table 2　Total memory size $MT(1,r)$.

Func.	In	Out	$MT(1,r)$					
			1	2	4	8	16	32
C432	36	7	6	6	6	6	12	12
C499	41	32	96	192	192	384	384	384
C880	60	26	24	24	48	48	96	96
C1908	33	25	24	48	48	48	96	96
C2670	233	140	12	24	24	48	48	96
C3540	50	22	192	192	384	384	768	768
C5315	178	123	12	12	24	24	48	48
C7552	207	108	12	12	12	24	24	48

[kilo words]

表3　C2670 に対する総メモリ容量 $MT(k,r)$
Table 3　Total memory size $MT(k,r)$ for C2670.

		r					
		1	2	4	8	16	32
k	1	12	24	24	48	48	96
	2	20	40	40	40	80	160
	3	18	36	72	72	144	144
	4	34	68	136	136	136	272
	5	66	132	264	264	528	528

[kilo words]

表4　C2670 に対するコスト/パフォーマンス $CP(k,r)$.
Table 4　Cost/Performance $CP(k,r)$ for C2670.

		r					
		1	2	4	8	16	32
k	1	1.00	1.00	**0.50**	**0.50**	**0.25**	**0.25**
	2	**0.83**	**0.83**	**0.42**	**0.21**	**0.21**	**0.21**
	3	**0.50**	**0.50**	**0.50**	**0.25**	**0.25**	**0.13**
	4	**0.71**	**0.71**	**0.71**	**0.35**	**0.18**	**0.18**
	5	1.10	1.10	1.10	**0.55**	**0.55**	**0.28**

表 5　PMDD(3, 2) に基づいた評価法とソフトウェアシミュレータの比較
Table 5　Comparison of PMDD(3, 2)-based evaluation with software simulation.

Function	In	Out	Prestissimo	PMDD
C432	36	7	3.2	1.7
C499	41	32	15.4	4.1
C880	60	26	19.2	10.9
C1908	33	25	7.6	5.9
C2670	233	140	55.6	227.5
C3540	50	22	29.0	3.3
C5315	178	123	112.8	151.9
C7552	207	108	46.5	156.1

[μs/vector]

た．各ユニットは 512 キロバイトの SRAM と XILINX 社の CPLD（XC95108-10PC84）で実現された制御回路とで構成した．ベンチマーク関数は，PMDD(3, 2) で表現した．入力ベクトルは LFSR（linear feedback shift register）によって生成したランダムパターンを用いた．試作したシステムは 24 MHz で動作した．ソフトウェアによる実現との比較参考のためソフトウェアシミュレータである高速な DD ベースのサイクルベースシミュレータ [10] と当方式で実現した場合との性能比較を表 5 に示す．例えば，C3540 は 50 入力 22 出力の関数である．1 個の入力ベクトルに対しすべての出力値を求めるのにソフトウェアシミュレータでは，29.0[μs]，PMDD(3, 2) に基づいた試作機では 3.3[μs] の計算時間を要している．MDD を生成する時間は，DD ベースのサイクルベースシミュレータでも同様に必要なので表には含んでいない．

ソフトウェアシミュレータに比べ低速なものもあるが，これはクロック周波数やユニット数を増やすことにより改善できる．例えば，$r = 32$，$k = 4$ の PMDD(k, r) を用いれば現在より約 20 倍高速にできる．この場合，表内のすべてのベンチマーク関数に対して本評価システムはソフトウェアシミュレータより高速になる．

4.3 考　　察

提案するシステムを用いるとき，我々は様々なパラメータを考えねばならない．MDD(k) 内の制御変数の個数；ユニットの個数 (r)；ユニット間の配線の個数；及び，一ユニット内の RAM のサイズ等である．現在の設計では，パラメータは以下のとおり．

（1） CPLD の中身を書き換えることで，システムは $k = 1, 2, 3$，及び 4 でも動作する．

（2） ユニット間の配線数は 16 である．つまり，$w_i = 2^{16}$．このことより，表 5 に示したすべてのベンチマークのシミュレーションができた．

（3） 一ユニット内の RAM のサイズは，512-kilo Byte である．もし必要ならユニットに RAM を追加することができる．

より多くのユニットを利用することでスループットは増加するが，r が非常に大きいと，クロックスキューなどの問題が起きるかもしれない．

ほとんどのシミュレーション対象に対してシステムの制御部を変更する必要はなく，各対象に対する PMDD を構成し，それを RAM に格納すればよい．この場合，変数の順序とユニットへの変数の割当てが重要である．CPLD は，試作に向いているので制御回路の実現に用いた．CPLD のリコンフィギュラビリティを用いて様々な k に対応することもできる．また，様々な決定グラフ，例えば，2 個の終端節点の代わりに多端子 $(0\cdots00)$，$(1\cdots11)$ をもった DD 等に対応するためにリコンフィギュラビリティを使用することもできる [15]．

5. む　す　び

本論文では，我々は PMDD に基づいた論理関数の評価システムを提案した．これは複数の制御回路で管理されたパイプライン化した MDD であり，汎用マイクロプロセッサで実現するよりも高速にできる．本方式は，クロックの周波数を上げずに処理能力を上げることができるので，低消費電力組込システム等にも応用が考えられる．

謝辞　貴重な御意見を頂いた富士通研究所の松永裕介博士，長崎大学の小栗清教授に深謝する．本研究の一部は文部省科研費による．

文　　献

[1] P. Ashar and S. Malik, “Fast functional simulation using branching programs,” ICCAD ’95, pp.408–412, Nov. 1995.
[2] B. Becker and R. Drechsler, “Efficient graph based representation of multivalued functions with an application to genetic algorithms,” Proc. International Symposium on Multiple Valued Logic, pp.40–45, May 1994.
[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] R.E. Bryant, “Boolean analysis of MOS circuits,” IEEE Trans. CAD of Integrated Circuits, vol.CAD-6, no.4, pp.634–649, July 1987.
[5] R.E. Bryant, “Symbolic simulation – Techniques and

applications," Proc. 27th Design Automation Conf., pp.517–521, June 1990.
[6] M. Davio, J.-P. Deschamps, and A. Thayse, DIGITAL SYSTEMS with algorithm implementation, John Wiley & Sons, 1983.
[7] N. Ishiura, H. Sawada, and S. Yajima, "Minimization of binary decision diagrams based on exchanges of variables," ICCAD-91, pp.472–475, 1991.
[8] S.D. Brown, R.J. Francis, J. Rose, and Z.G. Vranesic, Field Programmable Gate Arrays, Kluwer Academic Publishers, Boston, 1992.
[9] C. Lee, "Representation of switching circuits by binary decision programs," Bell System T. J., vol.38, pp.989–999, July 1959.
[10] P.C. McGeer, K.L. McMillan, A. Saldanha, A.L. Sangiovanni-Vincentelli, and P. Scaglia, "Fast discrete function evaluation using decision diagrams," ICCAD '95, pp.402–407, Nov. 1995.
[11] D.M. Miller, "Multiple-valued logic design tools," Proc. International Symposium on Multiple Valued Logic, pp.2–11, May 1993.
[12] S. Minato, N. Ishiura, and S. Yajima, "Shared binary decision diagram with attributed edges for efficient Boolean function manipulation," Proc. 27th ACM/IEEE Design Automation Conf., pp.52–57, June 1990.
[13] R. Murgai and M. Fujita, "Some recent advances in software and hardware logic simulation," 10th Int'l Conf. on VLSI Design, pp.232–238, Jan. 1997.
[14] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," ICCAD-93, pp.42–47, 1993.
[15] T. Sasao and J.T. Butler, "A method to represent multiple-output switching functions by using multi-valued decision diagrams," IEEE International Symposium on Multiple-Valued Logic, pp.248–254, Santiago de Compostela, Spain, May 1996.
[16] T. Sasao and M. Fujita, ed., Representations of Discrete Functions, Kluwer Academic Publishers, 1996.
[17] T. Sasao, Switching Theory for Logic Synthesis, Kluwer Academic Publishers, 1999.
[18] A. Thayse, M. Davio, and J.P. Deschamps, "Optimization of multiple-valued decision algorithms," ISMVL-79, pp.171–177, Rosemont, IL, May 1978.

（平成 12 年 9 月 4 日受付，12 月 21 日再受付）

井口　幸洋　（正員）

1982 明大・工・電子通信卒．1987 同大大学院博士課程了．同年同大・工・助手．1993 同大・理工・専任講師．2000 同大・理工・助教授，現在に至る．工博．この間，1996 年 4 月より 1 年間九州工業大学情報工学部にて私学研修員．論理設計，スイッチング理論，再構成可能なシステムに興味をもつ．情報処理学会会員．

笹尾　勤　（正員）

1972 阪大・工・電子卒．1977 同大大学院博士課程了．工博．同年同大・工・助手．1988 九州工大・情報・助教授，1993 同教授．2000 同マイクロ化総合技術センタ長併任，現在に至る．工博．この間，1982 年 2 月より 1 年間 IBM ワトソン研究所客員研究員，1990 年 1 月より 3 か月間米国海軍大学院教授．1979 丹羽記念賞，1987 ISMVL 論文賞各受賞．論理設計，スイッチング理論，多値論理などの研究に従事．著書『論理設計：スイッチング回路理論』（近代科学社），Logic Syntheis and Optimization（1993），Representations of Discrete Functions（1996），Switching Thoery for Logic Synthesis（1999）（Kluwer）等．情報処理学会会員，IEEE フェロー．

松浦　宗寛

1991 より九州工大技官．論理設計アルゴリズム，決定図技法，EXOR 論理回路，論理回路の分解に興味をもつ．

伊勢野　総

1996 明大・理工・物理卒．1998 同大大学院博士前期課程了．現在同大学院博士後期課程情報科学系在学中．論理設計，再構成可能システムに興味をもつ．