システムＬＳＩ設計技術　93－8
(1999. 11. 27)

PQMDDを用いた多出力論理関数の実現

井口 幸洋† 笹尾 勤‡ 松浦 宗寛‡

†明治大学 理工学部 情報科学科
〒214-8571 川崎市
e-mail: iguchi@cs.meiji.ac.jp

‡九州工業大学 情報工学部 電子情報工学科
〒820-8502 飯塚市
e-mail: {sasao, matsuura}@cse.kyutech.ac.jp

あらまし　多出力の論理関数を順序回路を用いて実現する方法を提案する．また，データ構造として，PQMDD(Paged Quasi reduced ordered Multi-valued Decision Diagram) を提案する．論理関数を順序回路で実現する方法としては，SBDD(Shared reduced ordered Binary Decision Diagram) に基づく方法が知られている．PQMDD に基づく方法は，複数の順序回路を用いて多出力関数を並列に評価するため，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 PQMDD で表現し，メモリで実現する．これに PQMDD 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 PQMDD 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，PQMDD.**

Implementation of Multiple-Outputs Function using PQMDDs

Yukihiro IGUCHI† Tsutomu SASAO ‡ Munehiro MATSUURA‡

†Dept. of Computer Science
Meiji University
Kawasaki 214-8571, JAPAN
e-mail: iguchi@cs.meiji.ac.jp

‡Dept. of Electronics and Computer Science
Kyushu Institute of Technology
Iizuka 820-8502, JAPAN
e-mail: {sasao, matsuura}@cse.kyutech.ac.jp

Abstract　A sequential realization of multiple-output logic functions is presented. As the data structure, we propose PQMDD (Paged Quasi reduced ordered Multi-valued Decision Diagram). A conventional sequential realization is based on SBDDs (Shared reduced ordered Binary Decision Diagrams). A realization based on PQMDD is faster than one based on SBDD, since multiple-output functions can be evaluated in parallel. In the proposed method, multiple-output logic function is represented by a PQMDD, and stored in static memory. Dedicated control circuits traverse the PQMDD in parallel. We represent multiple-output function for benchmark functions by SBDDs and PQMDDs and compare the size of memory and computation time.

key words　**multiple-output logic function, BDD, MDD, PMDD, PQMDD.**

1 はじめに

組込み機器等で用いる, 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過ぎる. 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 [1]. しかし, この方法は入力変数の個数 n が大きい時, 回路が大きくなり過ぎる. 出力値評価に時間がかかっても良い場合には, 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桁から3桁遅くなる.

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. この方法は, マイクロプロセッサを用いた従来の方法に比べて高速である.

基本的な考え方を説明するために n 変数の二値論理関数 $B^n \to B$ を考える. 関数 f をBDD(Binary Decision Diagram)[2, 3] 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 [4]. 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 [5, 11]. この場合, ROMの大きさは, BDDの節点数に比例する.

次に, 二値論理関数の各変数を k ビットづつグループ化し, 多値入力二値出力関数に変換し, それに対するMDD(Multi-valued Decision Diagram) [5, 6, 7, 8] 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. 更にメモリアクセス回数を減らすためにPQMDD(Paged Quasi reduced ordered Multi-valued Decision Diagram) を提案する. この場合, インデックスを参照する必要がなく, メモリアクセス数を半減でき, 高速化が可能となる.

本稿の構成は次のとおり. まず, 第2節で多出力論理関数を実現する方法について考察し, PMDD, および, PQMDDを提案する. 第3節ではPQMDDに基づく多出力論理関数の実現法を提案する. 第4節で実験結果を示す.

2 PQMDDと多出力論理関数

ここより, 簡単のためにSBDD(Shared reduced ordered Binary Decision Diagram) [9] 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, 根から終端接点までの全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を簡単の為にMDDと表記する.

定義 2.1 *MDD(k)* とは, 各非終端節点が 2^k 個の枝を持つ多値決定グラフ *(Multi-valued Decision Diagrams)* 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) に基づく評価方法は, BDDに基づくものに比べ 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].

例 2.1 *MDD(k)* の例を図 *2.1*に示す. 図 *2.1(a)* の *MDD(1)* は, *8*入力 *2*出力の論理関数を表している. 入力変数 (X_1, X_2, X_3, X_4) を $X_1 = (x_1, x_2)$, $X_2 = (x_3, x_4)$, $X_3 = (x_5, x_6)$, $X_4 = (x_7, x_8)$ と分割すると, 図 *2.1(b)* に示す *MDD(2)* が得られる. *MDD(2)* では, 根から終端節点までの経路長が *MDD(1)* の半分になり, 評価時間は半分になる. 同様に, 図 *2.1(c)* に *MDD(3)* を示す. *MDD(3)* はダミー変数 x_9 を含むことに注意. ∎

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.

定義 2.2 *MDD(K)* が次の条件を満たすとき, それを *QMDD(Quasi reduced ordered Multi-valued Decision Diagrams)* という.

1. v_1 を根とした部分グラフと v_2 を根とした部分グラフとが等価であるような *2*つの相異なる節点 v_1, v_2 を持たない.

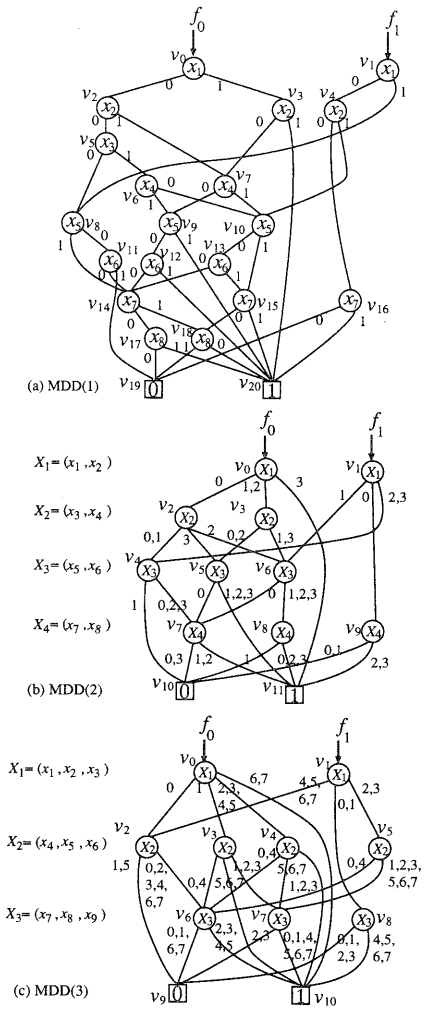

図 2.1: MDD(k) の例.

2. 根から終端節点までのどの経路も全ての変数を含む.

定義 2.3 *QMDD* の**幅**とは, *1* つの変数に関する節点数の最大値をいう.

QMDD(k) の節点数は, 対応する MDD(k) の節点数より大きく, 高々n 倍である. しかし, QMDD(k) では, 関数の評価の際, インデックスは 1 つづつ増えるのでインデックスを記憶する必要も参照する必要もない.

例 2.2 例 *2.1*の *MDD* に対応する *QMDD(k)* を図 *2.2* に示す. 図 *2.2*で * はすべての枝を表す.

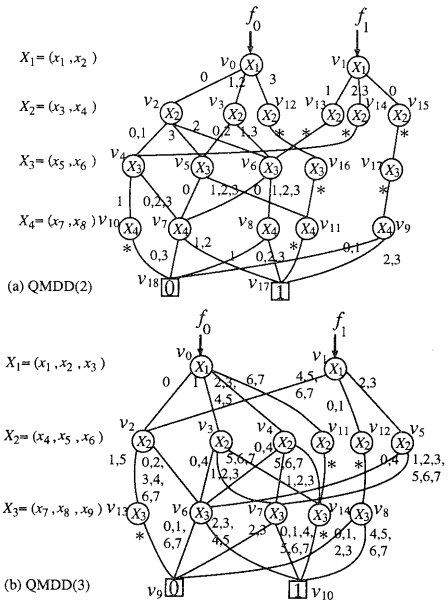

図 2.2: 例 2.1の MDD に対応する QMDD(k).

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

定義 2.4 *Paged reduced ordered MDD: PMDD(k, r)* 入力変数を $X = (X_1, X_2, \ldots, X_n)$ とする, 但し, $n = t \cdot r$. 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 $(s = 1, \cdots r - 1)$ 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.

PMDD は, 次の性質を持つ:

1. 1 つのページ内では, 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.

定義 2.5 *Paged Quasi reduced ordered Multi-valued Decision Diagrams: PQMDD(k, r)* は, 全てのレベルに非終端節点を持つ *PMDD(k, r)* である.

これまで述べてきた決定グラフの関連を図 2.3に示す. BDD を k 変数づつグループ化し, 多値化したものが MDD(k) であり, これを r ページに分割したものが PMDD(k,r) である. ここで, PMDD(k,r) の各ページの第 1 レベルには必ず節点が存在する. QROBDD(Quesi Reduced Ordered Binary Decision Diagram) は, 根から終端節点に至る全ての経路にそって全ての変数が出現する BDD である [12]. MDD(k) と QMDD(k), PMDD(k,r) と PQMDD(k,r) との関係も同様.

PMDD(1, 1) は BDD であり, PMDD(1, n) は QROBDD であることに注意.

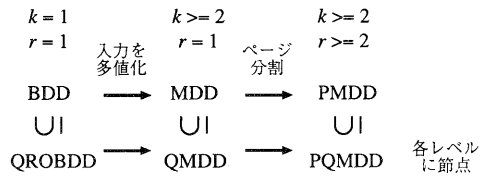

図 2.3: 決定グラフの関連.

定義 2.6 f を決定グラフ *(DD)* で表した時, *DD* の節点数を $size(DD, f)$ とする.

定理 2.1
$size(PMDD(1,1), f) \leq size(PMDD(1,r), f)$
$\leq size(PMDD(1,n), f)$,
$size(PMDD(k,r), f) \leq size(PQMDD(k,r), f)$.
ここで, $1 \leq r \leq n$.

定理 2.2 $size(PQMDD(k,r), f)$ は $r(1 \leq r \leq n)$ によらず一定.

PQMDD(k, r) は, 次の特長を持つ.

1. パイプライン化に適したデータ構造である.
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を短くできる.
3. インデックスを記憶する必要はない.

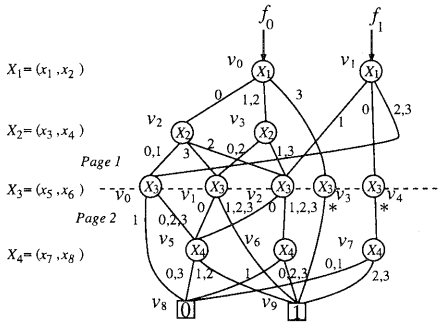

図 2.4: 例 2.1の関数に対する PMDD(2, 2).

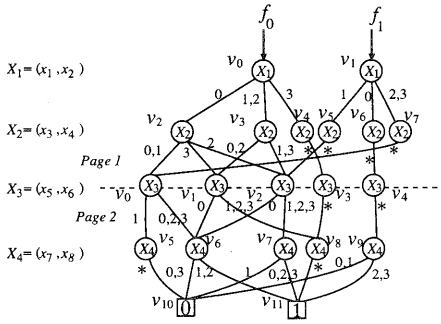

図 2.5: 例 2.1の関数に対する PQMDD(2, 2).

例 2.3 図 *2.1(b)* の *MDD(2)* を *2* ページに分割すると, 図 *2.4*に示す *PMDD(2,2)* を得る. 同様に図 *2.2(a)* の *QMDD(2)* を *2* ページに分割すると, 図 *2.5*に示す *PQMDD(2,2)* を得る. この例では, x_1 から x_4 を第 *1* ページに, x_5 から 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. ∎

3 PQMDD を実現する順序回路

順序回路を用いた多出力関数の実現方法としては, SBDD に基づく方法が知られている [5, 11, 13]. ここでは, 次の 3 つの方法を用いて多出力論理関数の計算を高速化する.

1. 計算をメモリと簡単な制御回路からなる単純なハードウェアで実現する. ソフトウェアで実現する場

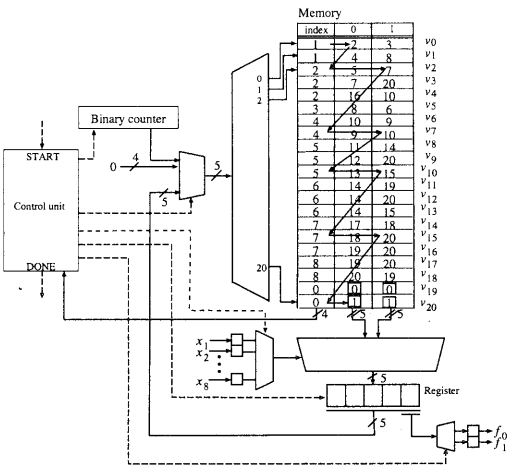

図 3.1: SBDD に基づく順序回路.

合の命令フェッチに要する時間を削減し高速化を行なう.

2. BDD の代わりに QMDD(k) を用いる. これにより計算を $2k$ 倍高速にする.

3. PQMDD(k, r) を用いる. r 個の計算ユニットをパイプライン化して用いる.

3.1 順序回路の動作

図 3.1に順序回路の概念図を示す. 提案する順序回路は, メモリと制御回路から構成される. 実現する論理関数を MDD(k) で表現し, これをメモリに書込んでおく. 順序回路は入力ベクトルに従って MDD(k) を辿り, 関数の評価値を返す.

例 3.1 図 *2.1(a)* の *SBDD(MDD(1))* のデータを回路に格納した例を図 *3.1* に示す. f_0 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_1 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. v_0 から v_{20} まで矢印に沿ってメモリを辿ることで, $f_0(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) = f_0\ (0, 1, 1, 1, 1, 1, 1, 0)$ の値を求める.

まず, f_0 の値を求めるために *2* 進カウンタの値を *0* にする. これがメモリの開始アドレスとなる. v_0 の *index* の値は *1* であるので, x_1 の値を読み出す. x_1 の値は *0* であり, これは *0* 枝をあらわしているので *0* 枝の値を読み出す. 値は *2* であるので, v_2 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*)* の値 *1* を読み出し $f_0\ (0, 1, 1, 1, 1, 1, 1, 0) = 1$ が求まる.

次に, *2* 進カウンタの値を *1* にし, これをメモリの開始アドレスとする. *1* 番地からメモリを辿ることで f_1 が求められる. ∎

1 つの枝を辿るのにメモリはインデックス部に 1 回, 0 枝, または, 1 枝のどちらか 1 回参照する. また, m 出力関数に対しては, m 回枝を辿らなければならない. 従って, 図 3.1に示す回路で n 入力 m 出力論理関数を評価するには, 最大 $2 \cdot n \cdot m$ 回のメモリアクセスが必要になる.

3.2 順序回路の高速化

本節では, 回路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 k 倍高速化する. また, QMDD を使いインデックス部を参照せずに計算することでメモリの読み出し回数を半分に減らすことで高速化する. 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) と QMDD(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を, $1/k$ に削減できるので, BDD に基づいた回路より k 倍高速にできる. この時も BDD と同様インデックス部を参照し, 次に加えるべき入力値を決定する必要が生じる. ここで, データ構造に QMDD を使用すれば入力値がどのような値をとっても全ての経路に同じ順番で入力変数が並んでいるので, インデックス部のメモリ内への格納も参照も不要になる. 従って, メモリの参照回数が半分になり, 高速化に役立つ. 但し, 非終端節点の個数は高々 n 倍に増える. 図 3.2に QMDD(2) に基づく順序回路を示す.

例 3.2 図 *2.2*に対する *QMDD(2)* に基づく順序回路を図 *3.2* に示す. v_0 から v_8 まで矢印に沿ってメモリを辿ることで, $f_0(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) = f_0(0, 1, 1, 1, 1, 1, 1, 0) = 1$ を得る. 各変数に対する値 $X_1 = (x_1, x_2) = (0, 1) = 1$, $X_2 = (x_3, x_4) = (1, 1) = 3$, $X_3 = (x_5, x_6) = (1, 1) = 3$, および, $X_4 = (x_7, x_8) = (1, 0) = 2$, に対応する枝を v_0 より辿る. *4* 回目のメモリ参照で到達した v_8 の *2* 枝の値は *1* であるので, f_0 の値は *1* であると評価できる.

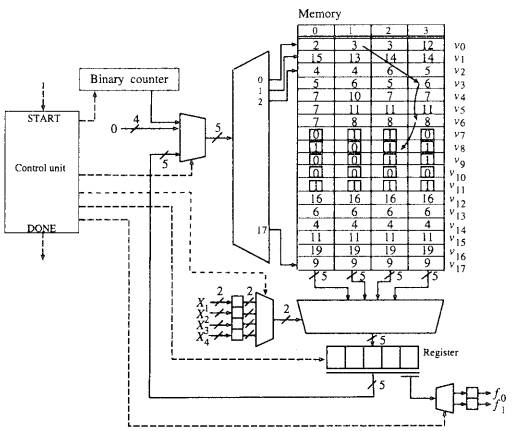

図 3.2: QMDD(2) に基づく順序回路.

同様に, f_1 は v_1 からメモリを辿ることによって評価できる. どのような入力ベクトルに対しても 4 回のメモリ参照で値が求まり, インデックスの参照は不要であることに注意.

図 *2.2(b)* に示したように, *QMDD(3)* の非終端節点の個数は *15* 個である. しかし, *QMDD(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 PQMDD(k, r) に基づいた順序回路

決定グラフを格納するためのメモリシステムが 1 個の場合, 一度に 1 つの入力ベクトルに対してのみしか評価ができない. ここでは, PQMDD(k, r) に基づいた順序回路を提案する. PQMDD(k, r) に対応する r 個のメモリとそれを辿る r 個の順序回路を用意し, これをパイプライン化を行なうことでスループットの向上を図る.

例 3.3 図 *2.5*に対する *PQMDD(2,2)* の順序回路を図 *3.3* に示す.

- 時刻 *1:* $f_1(X_1, X_2, X_3, X_4) = f_1(1, 3, 3, 2)$, $f_2(X_1, X_2, X_3, X_4) = f_2(1, 3, 3, 2)$ を求める. 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 に進み, $X2 = 3$ であるから *3* 枝の欄の *2* を読み出す. 次ページの探索を開始する番地を 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 から辿り始める. $f_1(X_3, X_4) = f_1(3, 2)$ に対応して v_2 の *3* 枝の欄の *7* を読み出す. 次に, v_7 の *2* 枝の欄の *1* を読み出す. これは終端節点 *1* を意味しており, 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, この時 *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. v_1 の *1* 枝の欄, v_5 の *3* 枝と辿り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* ページ目から受けとった v_2 より, $f_2(X_3, X_4) = f_2(3, 2)$ に対応して v_2 の *3* 枝の欄の *7* を読み出し, 次に v_7 の *2* 枝の欄の *1* を読み出す. これは, 終端節点 *1* を意味しており, $f_2(X_1, X_2, X_3, X_4) = f_2(1, 3, 3, 2) = 1$ が求まる.

∎

上例では 1 個の入力ベクトルに対してのみの評価を示した. 上例では 2 出力の例であったが m 出力で $m >> r$ の場合, オーバーヘッドを無視すれば, QMDD で実現した順序回路の r 倍のスループット (上例では 2 倍) が得られる.

4 実験結果

PQMDD(k,r) を生成する C プログラムを開発し, 多くの論理関数を PQMDD(k,r) で表現した. ベンチマーク回路に対する PQMDD(k, r), $(k = 1, 2, 3)$ の総節点数の比較を表 4.1に示す. 例えば, accpla は 50 入力, 69 出力の関数であり, SBDD で表現した場合の節点数は 5564 個である. これを PQMDD(1, 1) で実現すると節点数は 10508 個. 同じく PQMDD(1, 2) で実現すると 1 ページ目の節点数は 6137 個, 2 ページ目は 4371 個となり, 結局総節点数は 10508 個となる. ここで, k が変わらなければページ数によらず総節点数は変化しないことに注意 (定理 2.2参照). k を 1 から 3 へと変化させると, 節点数は 10508 個から 3416 個に減る. ただし, PQMDD(k,r) は, 2^k 個の次アドレスを必要とする. 従って, これを格納するためのメモリ容量は, それぞれ, 21016, 21416, 27328 となる.

PQMDD(2,r) に基づいた関数の評価は, PQMDD(1,r) に基づいた関数評価に比べ, 2 倍高速である. しかし, ほとんどのベンチマーク関数において必要なメモリ容量はほぼ同程度である. 従って, $k = 2$ を選択するとよいことがわかる. また, いくつかの関数に

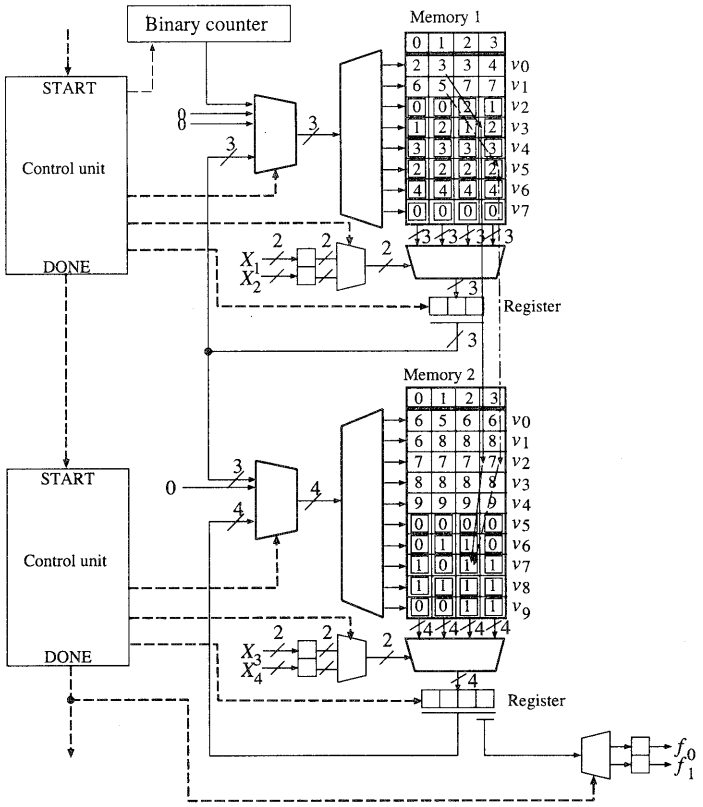

図 3.3: PQMDD(2, 2) に基づく順序回路.

対しては, $k = 3$ の場合でも, メモリ容量の増加が穏やかな関数もある.

5 あとがき

本稿では, 多出力関数を順序回路で実現する方法について述べた. 順序回路による多出力論理関数の実現方法としては, SBDD に基づくものが知られている. PQMDD(k,r) に基づく方法は, SBDD に基づくものに比べ, 最大 $k \times r$ 倍高速化できる. ただし, 必要なメモリは, 増加する.

提案する方法を用いることで, パイプライン化, つまり, 多くの RAM ブロックを同時に動作させ, クロック周波数を増加することなしに評価の高速化を達成できる.

謝辞

本研究の一部は文部省科研費による.

参考文献

[1] T. Sasao, *Switching Theory for Logic Synthesis*, Kluwer Academic Publishers, 1999.

[2] C. Lee, "Representation of switching circuits by binary decision programs," *Bell System T. J.*, 38, pp. 989–999, July 1959.

[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] P. Ashar and S. Malik, "Fast functional simulation using branching programs," *ICCAD'95*, pp. 408–412, Nov. 1995.

[5] A. Thayse, M. Davio, and J. P. Deschamps, "Optimization of multiple-valued decision algorithms," *ISMVL-79*, Rosemont, IL. pp. 171–177, May 1978.

[6] D. M Miller, "Multiple-valued logic design tools," *Proc. of International Symposium on Multiple Valued Logic*, pp. 2–11, May 1993.

表 4.1: Number of nodes in PQMDDs.

Name	In	Out	BDD	PQMDD					
				$k=1$		$k=2$		$k=3$	
				Nodes	Mem	Nodes	Mem	Nodes	Mem
accpla	50	69	5564	10508	21016	5354	21416	3416	27328
adr10	20	11	193	329	658	138	552	116	928
amd	14	24	458	767	1534	384	1536	252	2016
apex1	45	45	4978	7657	15314	3773	15092	2365	18920
apex3	54	50	935	5541	11082	2775	11100	1851	14808
apex5	117	88	1390	10806	21612	5434	21736	3687	29496
apex6	135	99	3235	25590	51180	12810	51240	8635	69080
apex7	49	37	307	2117	4234	1063	4252	739	5912
b10	15	11	451	722	1444	363	1452	246	1968
b2	16	17	4454	5330	10660	2787	11148	1803	14424
b3	32	20	486	1982	3964	990	3960	672	5376
bca	26	46	1424	2005	4010	998	3992	677	5416
bcb	26	39	1292	1809	3618	904	3616	620	4960
bcc	26	45	1133	1539	3078	779	3116	527	4216
bcd	26	38	846	1155	2310	582	2328	400	3200
cps	24	109	2318	3103	6206	1592	6368	1048	8384
des	256	245	3726	62417	124834	31194	124776	20911	167288
duke2	22	29	976	1590	3180	774	3096	563	4504
ex4	128	28	522	4458	8916	2234	8936	1504	12032
exep	30	63	611	1582	3164	805	3220	535	4280
frg2	143	139	6520	30493	60986	15349	61396	9971	79768
i5	133	66	311	5507	11014	2776	11104	1871	14968
i6	138	67	518	15159	30318	7596	30384	5052	40416
i7	199	67	610	26309	52618	13126	52504	8816	70528
i8	133	81	4366	39595	79190	19788	79152	13087	104696
i9	88	63	2277	6854	13708	3415	13660	2325	18600
ibm	48	17	408	1535	3070	776	3104	537	4296
jbp	36	57	421	1751	3502	897	3588	606	4848
k2	45	45	1275	4290	8580	2139	8556	1426	11408
mainpla	27	54	3308	4365	8730	2177	8708	1423	11384
misex3	14	14	1301	1378	2756	690	2760	458	3664
mlp10	20	20	79066	81130	162260	41342	165368	25365	202920
opa	17	69	542	621	1242	335	1340	238	1904
pdc	16	40	656	1001	2002	500	2000	338	2704
seq	41	35	1248	3614	7228	1786	7144	1261	10088
soar	83	94	551	5307	10614	2682	10728	1789	14312
spla	16	46	625	846	1692	415	1660	300	2400
ti	47	72	704	3196	6392	1608	6432	1123	8984
ts10	22	16	4391	8513	17026	4034	16136	3247	25976
xparc	41	73	1875	5059	10118	2532	10128	1737	13896

k: The number of variables in a group.
Nodes: The number of nodes.
Mem: Total memory size.

[7] B. Becker and R. Drechsler, "Efficient graph based representation of multivalued functions with an application to genetic algorithms," *Proc. of International Symposium on Multiple Valued Logic*, pp. 40-45, May 1994.

[8] 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 29-31, 1996.

[9] 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.

[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] M. Davio, J. -P. Deschamps, and A. Thayse, *DIGITAL SYSTEMS with algorithm implementation*, JOHN WILEY & SONS, 1983.

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

[13] C. C. Clare, *Designing Logic Systems Using State Machines*, McGraw-Hill, New York, 1973.

[14] T. Kam, T. Villa, R. K. Brayton, and A. L. Sagiovanni-Vincentelli, "Multi-valued decision diagrams: Theory and Applications," *Multiple-Valued Logic*, 1988, Vol. 4, No. 1-2, pp. 9–62, 1998.

[15] Y. Iguchi, T. Sasao, and M. Matsuura, "A hardware simulation engine based on decision diagrams," *Asia and South Pacific Design Automation Conference, Proc. ASP-DAC'2000*, Jan. 2000.