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.

分割 MTMDDs for CF マシンについて

中原　啓貴† 　笹尾　勤†† 　松浦　宗寛††

† 鹿児島大学 大学院 理工学研究科 電気電子工学専攻 〒 890–0065 鹿児島県鹿児島市郡元 1–21–40
†† 九州工業大学 大学院 情報工学府 情報創成工学専攻 〒 820–8502 福岡県飯塚市大字川津 680–4

あらまし　分割した回路を表現する多値決定グラフとして分割 MTMDDs for CF (Decomposed multi-terminal multi-valued decision diagrams for characteristic function) が提案されている．従来の研究より, 分割 MTMDDs for CF は複雑な関数をコンパクトに表現できることが知られている．本論文では, 分割 MTMDDs for CF を模擬するマシンについて述べる．まず, 分割 MTMDDs for CF について述べ, 分割 MTMDDs for CF を評価する命令セットについて述べる．次に, 分割 MTMDDs for CF を模擬するマシンについて述べる．MCNC ベンチマーク関数を用いて他のプロセッサとの比較を行った結果, 多出力論理関数の評価に関して FPGA 上に実現した分割 MTMDDs for CF マシンは Nios II より 13.12 倍高速であった．また, Atom 上のソフトウェアより 1.91 倍高速であった．消費電力遅延時間積に関して, 分割 MTMDDs for CF マシンは Nios II より 66.84 倍小さく, Atom より 18.66 倍小さかった．

On a Decomposed MTMDDs for CF Machine

Hiroki NAKAHARA†, Tsutomu SASAO††, and Munehiro MATSUURA††

† Faculty of Engineering, Kagoshima University, 1-21-40, Korimoto, Kagoshima 890–0065, Japan
†† Department of Creative Informatics, Kyushu Institute of Technology
680–4, Kawazu, Iizuka, Fukuoka, 820–8502 Japan

Abstract　A decomposed multi-terminal multi-valued decision diagrams for characteristic function (MTMDDs for CF) represents decomposed circuits. A previous work shows that the decomposed MTMDDs for CF is smaller than monolithic decision diagrams for complex functions. This paper shows the decomposed MTMDDs for CF machine. First, we introduce the decomposed MTMDDs for CF. Then, we consider the instruction sets to evaluate the decomposed MTMDDs for CF. Next, we show the architecture for the decomposed MTMDDs for CF machine. We compare the decomposed MTMDDs for CF machine with other MPUs using MCNC benchmark functions. The decomposed MTMDDs for CF machine is 13.12 times faster than Altera's Nios II processor, and is 1.91 times faster than Intel's Atom N455 processor. As for the power-delay product, it is 60.84 times smaller than Nios II processor, and is 18.66 times smaller than Atom N455 processor.

1. はじめに

ディープサブミクロン時代以前のプロセッサでは LSI のプロセスを微細化することで, 消費電力を抑えつつ性能向上を達成してきた．しかし, プロセス微細化が 90nm に突入するとリーク電流による消費電力の増加が問題となり, 消費電力性能に優れたプロセッサが要求されている [4].

決定グラフマシン [1], [8] とは決定グラフ (DD: Decision Diagram) を評価する専用プロセッサである．決定グラフマシンの応用として, 産業用シーケンサ [21], 論理シミュレーションアクセラレータ [5], パケット分類器が報告されている [14]．各節点の入力数を自由に設定できる決定グラフをヘテロジニアス MDD (Heterogeneous Multi-valued DD) という [10]. HMDD は各節点の入力数を適切に決めることで, 2 値の決定グラフである BDD (Binary Decision Diagram) と同じメモリ量で平均して約 2 倍高速に評価できる [15]. HMDD は各節点の入力数を増やすとメモリ量は増加するが, パス長を短縮でき, BDD と比較して遅延時間を短縮できる．つまり, HMDD マシンはメモリ量を増加することで遅延時間を短縮できる．HMDD マシンは汎用のプロセッサよりも高速であり, 同一性能では動作クロックを落とすことができるため, 動的消費電力が小さい．また, HMDD マシンは汎用のプロセッサよりも命令数が少なく, 制御回路が小さくなることから静的消費電力も小さい．従って, HMDD マシンは消費電力遅延時間積に優れたプロセッサであり, ベンチマークによる実測によると, HMDD マシンの消費電力遅延時間積が Intel 社の Core i5 よりも一桁優れている [12].

しかし, 従来の HMDD では複雑な関数を表現する場合, 節点数が指数関数的に爆発する問題があった．我々は回路の分割を表現する多値決定グラフ (分割 MTMDDs for CF: Decomposed multi-terminal multi-valued decision diagrams for characteristic function) を提案した [11]．分割 MTMDDs for CF は分割した回路の外部信号線と外部入力のファンアウトを許容するため, この決定グラフに特化した専用マシンが必要である．本論文では, FPGA と SRAM を用いた分割 MTMDDs for CF マシ

ンの実現法について述べる．次に, 従来の組込みプロセッサと比較を行い, 分割 MTMDDs for CF マシンが実行時間と消費電力, 及び消費電力遅延時間積に関して優れていることを示す．

第 2 章では決定グラフを定義し, 第 3 章では回路の分割を定義し, 第 4 章では分割 MTMDDs for CF について述べ, 第 5 章では分割 MTMDDs for CF マシンについて述べ, 第 6 章では実験結果を示し, 第 7 章で本論文のまとめを行う．

2. 決定グラフの定義

2.1 決定グラフ (DD: Decision Diagram)

論理関数 f に対して次に定義する**シャノン展開**を繰返し適用することで **2 分決定グラフ (BDD: Binary Decision Diagram)** を得る．

［定義 2.1］ 任意の論理関数 $f(x_1, x_2, \ldots, x_n)$ はシャノン展開を用いて

$$f(x_1, x_2, \ldots, x_i, \ldots, x_n) = \bar{x}_i f(x_1, x_2, \ldots, 0, \ldots, x_n) \vee x_i f(x_1, x_2, \ldots, 1, \ldots, x_n).$$

と展開でき, 変数の展開順序を**変数順序**という．

［定義 2.2］ BDD は節点と枝から構成される有向グラフである．節点を v, 節点集合を V と表記する．節点 v に対応する変数のインデックスを $index(v) \in \{0, 1, \ldots, n\}$ と表記する．$index(v) = 0$ のとき, 節点 v を**終端節点**といい, $index(v) \neq 0$ のとき, 節点 v を**非終端節点**という．終端接点は 0 または 1 の 2 値を取り, 論理関数 f の関数値に対応する．非終端節点 v は 2 つの子節点 $low(v), high(v) \in V$ に接続する枝を持つ．変数順序を固定した BDD を**順序付き BDD (OBDD: Ordered BDD)** という [2].

［定義 2.3］ [7] **多値決定グラフ (MDD(k): Multi-valued DD)** とは各非終端節点が 2^k 個の枝を持つ決定グラフである．

［定義 2.4］ **既約順序付き BDD (ROBDD: Reduced Ordered BDD)** とは OBDD に対して以下の 2 つの簡単化手法を繰返し適用して得られる決定グラフである．

1. 等価なサブグラフを共有する．
2. 節点 u の全ての出力枝が指す節点が, 節点 v の出力枝が指す節点と等しい場合, 節点 u を削除し, 節点 u の入力枝を節点 v に接続する．

ROBDD と同様に ROMDD(k) も定義できる．

［定義 2.5］ 単一出力論理関数を $f(X) : B^n \to B, B = \{0, 1\}$ とする． ここで, $X = (x_1, x_2, \ldots, x_n), x_i \in B$ は f の入力変数である． X の変数の集合を $\{X\}$ で表す． $\{X\} = \{X_1\} \cup \{X_2\} \cup \cdots \{X_u\}$ かつ $\{X_i\} \cap \{X_j\} = \phi (i \neq j)$ のとき, $(X_1, X_2, \ldots, X_u)$ を **X の分割**という．また, X_i を**超変数**という．$k_i = |X_i| (i = 1, 2, \ldots, u)$ とすると $k_1 + k_2 + \cdots + k_u = n$ である．k_i を**超変数 X_i のサイズ**という．

［定義 2.6］ 変数 X の分割を $(X_1, X_2, \ldots, X_u)$ とする．各節点 i の入力数を $k_i = |X_i|$ とするとき, $k = |X_1| = |X_2| = \cdots = |X_u|$ となる MDD を**ホモジニアス MDD (homogeneous MDD)** と呼び, $MDD(k)$ と表記する．一方, 各 k_i が必ずしも等しくない MDD を**ヘテロジニアス MDD (HMDD: Heterogeneous MDD)** [10] と呼ぶ．

ヘテロジニアス MDD はホモジニアス MDD の一般形である．以降, 本論文では多値決定グラフとしてヘテロジニアス MDD を扱う．

2.2 平均パス長 (Average Path Length)

各節点の評価時間が全ての等しいと仮定すると, 決定グラフの評価時間は**平均パス長** [3] に比例する．決定グラフマシンでは各節点を一定時間で評価する．従って, 評価時間は平均パス長に比例する．

［定義 2.7］ 決定グラフの根節点から終端節点までの経路を**パス (path)** と呼ぶ．パス上に現れる枝の個数を**パス長**という．

［定義 2.8］ 決定グラフの変数 X の分割を $(X_1, X_2, \ldots, X_u)$ とする．X_i を r 値の論理変数とし $c \in \{0, 1, \ldots, r-1\}$ とする．$P(X_i = c)$ は X_i が c となる確率を表す．r 値の変数によってパス p_i を通過する確率を**パス確率 (PP: Path Probability)** とし, $PP(p_i)$ と表記する．このとき $PP(p_i) = \sum_{\vec{c} \in C_i} P(X_1 = c_1) \times P(X_2 = c_2) \times \cdots \times P(X_u = c_u)$ である．ここで C_i はパス p_i を通過する変数 X の集合を現し, $\vec{c} = (c_1, c_2, \ldots, c_u)$ である．

［定義 2.9］ 決定グラフの**平均パス長 (APL: Average Path Length)** とは $APL = \sum_{i=1}^{N} PP(p_i) \times l_i$ である．ここで N はパス数を表し, l_i はパス p_i のパス長を表す．

2.3 多出力論理関数の表現

実用的な関数は多出力論理関数である．ここで, 多出力論理関数を表現する決定グラフの定義を行う．

［定義 2.10］ 入力変数を $X = (x_1, x_2, \ldots, x_n)$, 多出力関数を $\vec{f} = (f_1(X), f_2(X), \ldots, f_m(X))$ とする． **多出力関数の特性関数 (CF: Characteristic Function)** を $\chi(X, Y) = \bigwedge_{i=1}^{m} (y_i \equiv f_i(X))$ とする．

n 入力 m 出力関数の特性関数は, $(n + m)$ 変数の 2 値論理関数であり, 入力変数 $x_i (i = 1, 2, \ldots, n)$ の他に, 各出力 f_i に対して出力変数 y_i を用いる．$B = \{0, 1\}$ とする．$\vec{a} \in B^n$ かつ $F = (f_1(\vec{a}), f_2(\vec{a}), \ldots, f_m(\vec{a})) \in B^m$ とする．ここで, $\vec{b} \in B^m$ とすると,

$$\chi(\vec{a}, \vec{b}) = 1 \ (\vec{b} = F(\vec{a})) = 0 \ (otherwise)$$

となる．

［定義 2.11］ 多出力関数 $\vec{f} = (f_1, f_2, \ldots, f_m)$ の **MTBDD for CF (Multi-Terminal BDD for CF)** とは, 特性関数 χ を表現する MTBDD である．ただし, MTBDD の変数は, 根節点を最上位としたとき, 変数 y_i は f_i に依存する変数の下に置く．

［定義 2.12］ [19] n 変数多出力論理関数 $\vec{f}$ を表現する MTBDD for CF の x_k と x_{k+1} の間を交差する枝の数を第 k 段目での **MTBDD for CF の幅**という．ここで, 同じ節点を指している枝は 1 と計数する．また, 幅を計数する際, 出力を表現する変数から, 定数 0 に向かう枝は無視する．したがって幅は入力変数の間でのみ定義できる．

3. 多段論理回路のクラスタ分割

3.1 組合せ論理回路のグラフ表現

本論文では組合せ論理回路を**有向非巡回的グラフ (DAG: Directed Acyclic Graph)** で表現する．DAG において, **外部入力節点**は外部入力端子を表現し, **外部出力節点**は外部出力端子を表現し, **中間節点**は 2 入力論理ゲート(注1)を表現する．論理ゲート i から j へ接続されている場合, DAG の節点 i から j への有向枝が存在する．

本論文では, 外部入力端子のファンアウトを許容せず, 別の節点として扱う．つまり, 外部入力節点数は必ずしもユニークな外部変数の個数とは一致しない．本論文では, ファンアウトを持たない外部入力端子を **(ユニークな入力変数名)_(通し番号)** で表記する．

［例 3.1］ 図 1 に 2 ビット乗算器を示す． ここで, $\{x_0, x_1, x_2, x_3\}$ は外部入力, $\{y_0, y_1, y_2, y_3\}$ は外部出力を表す． 図 2 に図 1 を表現する DAG を示す． ここで, $X = \{x_{0\_0}, x_{0\_1}, x_{1\_0}, x_{1\_1}, x_{2\_0}, x_{2\_1}, x_{3\_0}, x_{3\_1}\}$ は外部入力節点, $Y = \{y_0, y_1, y_2, y_3\}$ は外部出力節点, $W = \{w_0, w_1, w_2, w_3, w_4, w_5, w_6, w_7\}$ は中間節点を表す． ■

［定義 3.13］ X を外部入力節点の集合とする．ユニークな変数の集合を $U(X)$ で表す．また, ユニークな変数の個数を $|U(X)|$ で表す．

［例 3.2］ 図 2 の DAG において, $U(X) = \{x_0, x_1, x_2, x_3\}$, $|U(X)| = 4$ である． ■

本論文では, 分割された回路を決定グラフで表現する．ここでは, 回路の分割を定義する．

［定義 3.14］ **カット** (S, T) とは DAG の節点集合の分割であり, $s \in S$, $t \in T$ に対して, T のどの節点からも S のどの節点への有向枝が存在しないものである．

［定義 3.15］ 外部入力節点の集合を X とする．X に依存する節点集合を**依存節点集合**といい $D(X)$ で表す．依存節点集合に

(注1)：否定ゲート (NOT) は同一入力の NAND に置換する．

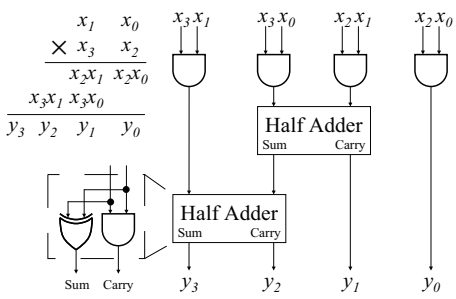

図 1　2 ビット乗算器.

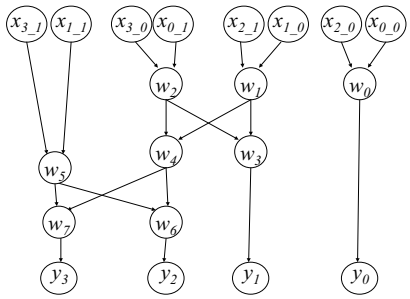

図 2　2 ビット乗算器を表現する DAG.

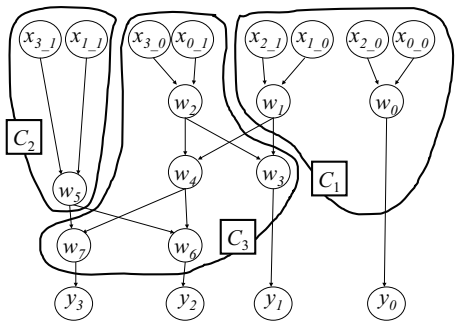

図 3　クラスタ分割の例.

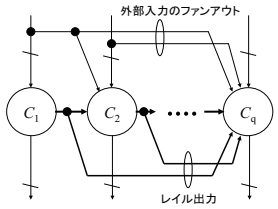

図 4　クラスタ分割を実現する回路.

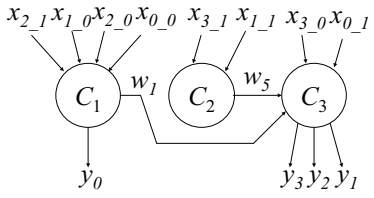

図 5　2 ビット乗算器のクラスタ分割を実現する回路.

は X 自身を含むが, 出力節点集合 Y を含まない.

［例 3.3］ $X = \{x_{1\_0}, x_{2\_1}, x_{0\_1}, x_{3\_0}\}$ とする. このとき, $D(X) = \{x_{1\_0}, x_{2\_1}, x_{0\_1}, x_{3\_0}, w_1, w_2, w_3, w_4\}$ である. ■

［定義 3.16］ 回路を表現する DAG の節点集合を V; 外部入力節点の集合 X の分割を $X = (X_1, X_2, X_3, ..., X_q)$; 外部出力節点の集合を Y とする. ただし, $i \neq j$ に関して, $X_i \cap X_j = \phi$. **トポロジカル順序付きカット集合** $\{(S_i, T_i)|i = 1, 2, \ldots, q\}$ とは以下の 3 つの条件を満たすカットの集合である.

1. 1 番目のカット (S_1, T_1) では, $S_1 = D(X_1)$, $T_1 = V - S_1$.
2. i $(1 < i < q)$ 番目のカット (S_i, T_i) では, $S_i = D(X_1 \cup X_2 \cup \ldots \cup X_i)$, $T_i = V - S_i$
3. q 番目のカット (S_q, T_q) では, $S_q = D(X)$, $T_q = Y$.

外部節点集合の分割が決まれば, トポロジカル順序付きカット集合は一意に決まる. この分割はトポロジカル順序を考慮しているので, 節点集合 S_i を順番に評価すれば回路を評価できる.

［定義 3.17］ トポロジカル順序付きカット集合 $\{(S_i, T_i)|i = 1, 2, \ldots, q\}$ において, 節点集合 $C_i = S_{i+1} - S_i$ を**クラスタ** i という. また, 与えられた多段論理回路の DAG の節点をクラスタ集合 $\{C_1, C_2, \ldots C_q\}$ に分割することを**クラスタ分割**という.

［例 3.4］ 図 2 に示した DAG の節点のクラスタ分割を $C_1 = \{x_{0\_0}, x_{2\_0}, x_{1\_0}, x_{2\_1}, x_{0\_1}, w_0, w_1, w_3\}$, $C_2 = \{x_{1\_1}, x_{2\_1}, w_5\}$, $C_3 = \{x_{3\_0}, w_2, w_4, w_6, w_7\}$ とした結果を図 3 に示す. ■

クラスタ集合 $\{C_1, C_2, \ldots, C_q\}$ は, トポロジカル順序付きカット集合の定義よりトポロジカルな順序で接続される. $i < j$ とすると, カットの定義からクラスタ C_i から C_j には有向枝が存在し得るが, クラスタ C_j から C_i への有向枝は存在しない.

［定義 3.18］ クラスタ集合 $\{C_1, C_2, \ldots C_q\}$ は図 4 に示す回路で実現できる. ここで, $i < j$ とする. クラスタ C_i から C_j へ接続している C_i の出力を**レイル出力**という. クラスタ C_i から C_j へ接続している C_j の入力を**レイル入力**という.

つまり, クラスタの入力は外部入力とレイル入力であり, クラスタの出力は外部出力とレイル出力である.

［例 3.5］ 図 5 に図 3 に示した 2 ビット乗算器のクラスタ分割を実現する回路を示す. ■

4. 分割 MTMDDs for CF

4.1 分割 MTBDDs for CF

クラスタ集合 $\{C_1, C_2, \ldots, C_q\}$ において, クラスタ C_i は論理回路の一部を表現する. ここで, 外部入力とレイル入力を入力とし, 外部出力とレイル出力を出力とする多出力論理関数を表現する MTBDD for CF を**クラスタ** C_i **を表現する MTBDD for CF** という. ここで, ある段のクラスタのレイル出力は MTBDD for CF の出力を表す変数となり, 次段以降のクラスタの入力となる.

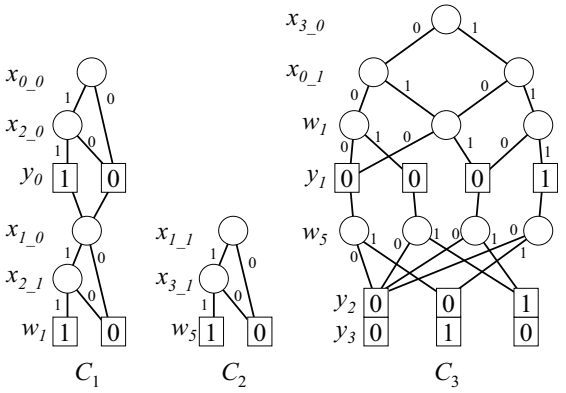

図 6　クラスタを表現する MTBDD for CF の例.

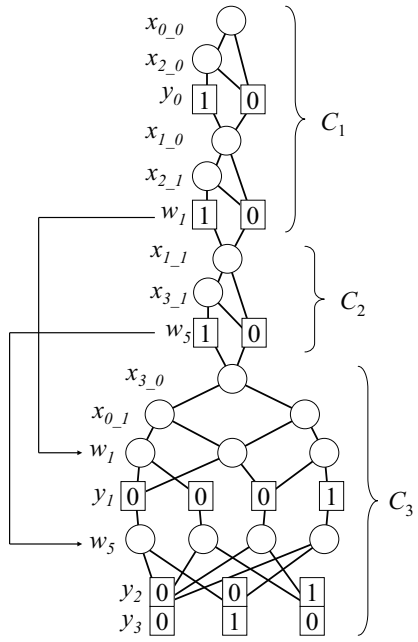

図 7　クラスタ分割を表現する MTBDD for CF (分割 MTBDD for CF) の例.

［例 4.6］ 図 6 に図 5 に示したクラスタ分割した 2 ビット乗算器の各クラスタを表現する MTBDD for CF を示す. ■

［定義 4.19］ 各クラスタを表現する複数の MTBDD for CF をトポロジカル順序で直列に接続した MTBDD for CF を**クラスタ分割を表現する MTBDD for CF** (略して分割 MTBDD for CF) という.

［例 4.7］ 図 7 に図 3 に示した 2 ビット乗算器の分割 MTBDD for CF を示す. ■

多段論理回路を決定グラフで表現する場合, 外部入力とレイル出力のファンアウトの制約に応じて, 様々な決定グラフに分

表 1　クラスタ分割を表現する決定グラフの分類.

決定グラフ	外部入力のファンアウト	レイル出力のファンアウト
単一 MTBDD for CF [9]	禁止	禁止
Indexed BDD [6]	許可	禁止
分割 MTBDD for CF	許可	許可

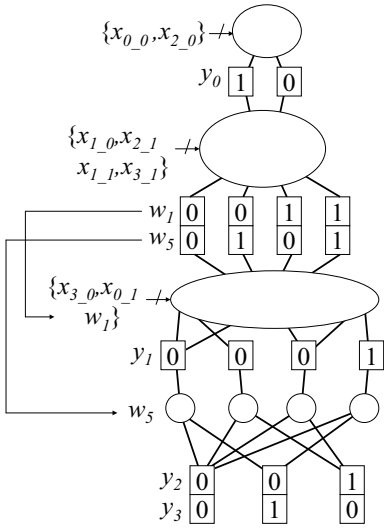

図 8　分割 MTMDD for CF の例.

類できる．多段論理回路のクラスタ分割による決定グラフを用いた表現法の分類を表 1 に示す．表 1 より, 分割 MTBDD for CF はレイル出力のファンアウトを許すという点で新規性がある.

4.2　分割 MTMDD for CF

分割 MTBDD for CF の各節点の入力数を多ビットに拡張することで, **分割 MTMDD for CF** を得る．多くの場合, MTMDD for CF は MTBDD for CF よりもメモリ量は増加するが APL を短縮でき, MTBDD for CF と比較して評価時間を短縮できる.

［例 4.8］ 図 8 に図 7 に示した 2 ビット乗算器の分割 MTBDD for CF を分割 MTMDD for CF に変換した例を示す．ここで, クラスタ C_0 の一部と C_1 を併合して変数順序を入れ替え, 新たな節点を作成していることに注意. ■

例 4.8 に示すように, 複数のクラスタを併合して多値の節点を作ることも可能である．また, 変数順序を入れ替えることも可能である.

5.　分割 MTMDDs for CF マシン

5.1　分割 MTMDDs for CF の模擬

分割 MTMDDs for CF を模擬するには

1. MTMDDs for CF の節点を模擬する命令セット
2. 命令セットを保持するメモリ
3. 命令セットを模擬する回路

が必要である．本章で各要素の実現について述べる.

5.2　分割 MTMDDs for CF を模擬する命令 [13]

分割 MTMDDs for CF の非終端節点は**間接多分岐方式 (indirect branch)** で模擬する．図 9 に間接分岐方式を示す．また, 図 10 に間接分岐命令を示す．間接分岐命令は, 現在の節点のインデックスに応じたアドレスを間接参照し, 分岐先のアドレスを読み出す．間接分岐命令はメモリ参照を 2 回行うが, メモリを効率よく利用できる利点がある．分割 MTMDDs for CF マシンで用いる命令数は 2 つだけなのでオペコードは 1 ビットあれば十分である．間接分岐命令はオペコードに 0 を割当てる．終端節点は**多出力・ジャンプ命令 (output and jump instruction)** で模擬する．図 11 に多出力・ジャンプ命令を示す．多出力命令は, 多ビットの出力値を出力し, 指定された番地にジャンプする．出力値とジャンプアドレスが別々の番地に保持されているので, 実行の際, メモリ参照が 2 回行われる．多出

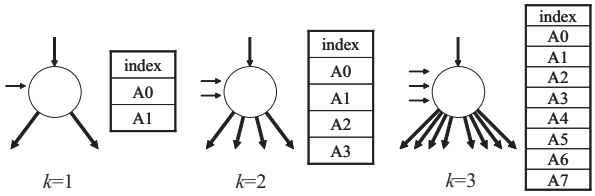

図 9　分割 MTMDDs for CF の非終端節点を評価する間接分岐方式.

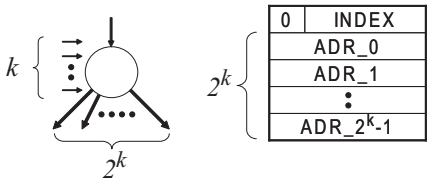

図 10　間接分岐命令.

図 11　多出力・ジャンプ命令.

力・ジャンプ命令にはオペコードに 1 を割当てる.

5.3　プログラムを保持するメモリ

分割 MTMDDs for CF のプログラムはオフチップメモリに格納する．FPGA はオンチップメモリを持つが, 容量が少ないため, 数 MBytes オーダのメモリ量が必要な MTMDDs for CF マシンではハイエンド FPGA が必要でありコストの面で実用的とは言えない．一方, SRAM は FPGA と比較して低価格であり, 特別なメモリインタフェースも不要であることから, 本論文ではオフチップメモリとして SRAM にプログラムを格納する．DDR-SDRAM も候補として挙げられるが, 専用のメモリコントローラが必要であり, リフレッシュ動作による電力消費が多いことから, 低消費電力向けのプロセッサには不向きである.

5.4　各命令を模擬する回路

5.4.1　外部入力とレイル出力のファンアウトの模擬

分割 MTMDDs for CF は外部入力とレイル出力のファンアウトを入力として扱う．従って, これらを接続するための回路が必要である．クラスタ分割により, 様々な分割 MTMDDs for CF を生成できるため, レイル出力と外部入力を任意に接続できる回路が必要である．本論文では, 分割 MTMDDs for CF の節点間の枝は FPGA で模擬する．これらはランダムロジックと同じであるが, 回路の大部分は分割 MTMDDs for CF の節点で模擬される．従って, 回路を直接 FPGA にマッピングする場合と比較して, 小規模 FPGA, あるいは CPLD でも実現可能であり, 低コストで回路を実現可能である.

分割 MTMDDs for CF では, メモリ量を増やすと節点の入力数を大きく取れるため, レイル出力や外部出力のファンアウトを削減でき, FPGA で模擬する部分をさらに削減できる．また, メモリ量を増やすことは分割 MTMDDs for CF の APL を削減でき, 高速に処理できる．従って,　メモリ量を増やすことは高速化と FPGA の面積削減に繋がり, 一定の性能を満たせば十分なリアルタイム処理では, 動作クロックを下げることができるため, 消費電力の削減に繋がる可能性がある.

5.4.2　分割 MTMDDs for CF マシンのアーキテクチャ

図 12 に分割 MTMDDs for CF マシンを示す．図 12 において, **命令メモリ (Instruction memory)** は命令を保持し; **命令レジスタ (Instruction register)** は命令メモリから読み出した命令を保持し; **プログラムカウンタ (PC)** は現在評価を行っている節点のアドレスを保持する．間接分岐命令を実行するため, 入力を取り込む**フェッチモード (fetch mode)** と分岐を実行する**ジャンプモード (jump mode)** を実行する．多出力・ジャンプ命令は出力を実行する**出力モード (output mode)** を実行した後, ジャンプモードを実行する．制御回路を用いて 3 通りのモードを切り替える．分割 MTMDDs for CF は超変数のサイズが異なり, 外部入力, レイル出力, 及び順序回路の状態変数を**選択回路 (Selector)** で選択する．分割 MTMDDs for CF はトポロジカル順序で回路を分割しており, レイル出力は後段

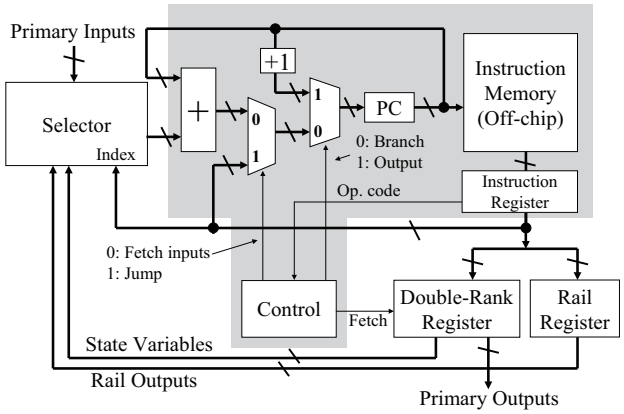

図 12　分割 MTMDDs for CF マシン.

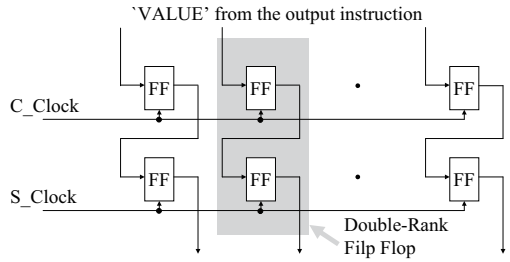

図 13　ダブルランク・レジスタ.

の節点のレイル入力となる. **レイルレジスタ (Rail register)** でレイル出力を保持する. 一方, 外部出力と状態変数は評価中の値と以前の値を保持しなければならないので, **ダブルランク・レジスタ (Double-Rank register)** [18] で保持する. 図 13 にダブルランク・レジスタを示す. ダブルランク・レジスタは評価中の値を保持するためのクロックと全ての評価が終わったときに値を転送するためのクロックを持ち, 各ビットは**ダブルランク・フリップフロップ**で構成されている.

以下に, 間接分岐命令と多出力・ジャンプ命令の実行を示す.

［アルゴリズム 5.1］ (間接分岐命令)

1. フェッチモードの実行

1.1 PC を参照し命令メモリを読み出す.

1.2 入力変数と PC のカウンタを加算するため, 制御回路は制御信号を出力する. 間接参照するアドレスを PC に取り込む.

2. ジャンプモードの実行

2.1 PC を参照しジャンプアドレスを読み出す.

2.2 ジャンプを行うため, 制御回路は制御信号を出力する. ジャンプアドレスを PC に取り込む.

［アルゴリズム 5.2］ (多出力・ジャンプ命令)

1. 多出力モードの実行

1.1 PC を参照し命令メモリを読み出す.

1.2 制御回路は制御信号を出力し, 現在のアドレスを 1 つ加算し PC に取り込む. 同時に出力をダブルランク・レジスタに出力する.

2. ジャンプモードの実行はアルゴリズム 5.1 で示したジャンプモードと同じ.

［例 5.9］ 図 14 に例 4.8 に示した分割 MTMDDs for CF を模擬する分割 MTMDDs for CF マシンの変数選択の例を示す.

1. 最上位の節点を評価するため, 外部入力 x_0, x_2 を選択する. 外部出力 y_0 をダブルランク・レジスタにセットする 14 (1).

2. 次の節点を評価するため, 外部入力 x_1, x_2, x_3 を選択する. レイル出力 w_1, w_5 をレイル・レジスタにセットする 14 (2).

3. 次の節点を評価するため, 外部入力 x_0, x_3 とレイル入力 w_1 を選択する. 外部出力 y_1 をダブルランク・レジスタにセットする 14 (3).

4. 最下位の節点を評価するため, レイル入力 w_5 を選択する. 外部出力 y_2, y_3 をダブルランク・レジスタにセットする. 全ての出力を評価したので, ダブルランク・レジスタの値を後段のレジスタに転送する 14 (4). ■

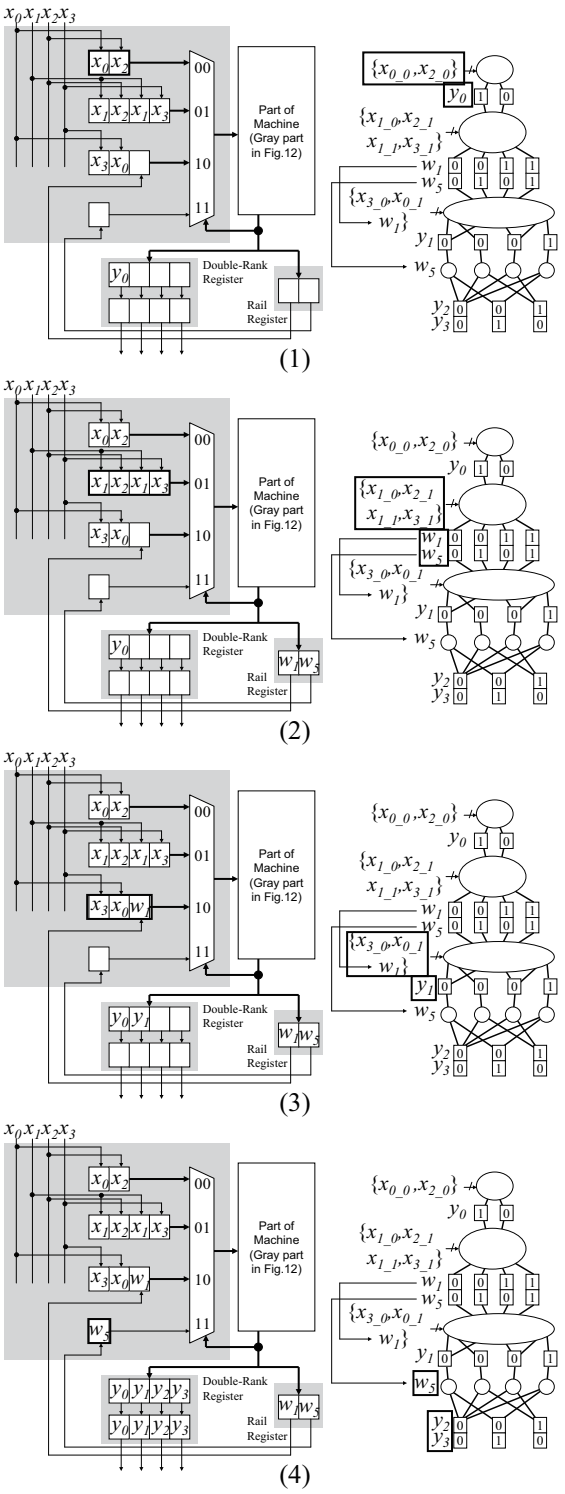

図 14　変数の選択例.

6. 実験結果

6.1 他のプロセッサとの比較

MCNC ベンチマーク関数 [20] を分割 MTMDDs for CF に変換し, 他のプロセッサと実行時間, 瞬間消費電力, 及び消費電力遅延時間積の比較を行った. 他のプロセッサは分割 MTMDDs for CF を模擬する C コードを実行コードに変換し, 実行した. 各プロセッサの実装法, 及び実行環境を示す.

a） 分割 MTMDDs for CF マシン

分割 MTMDDs for CF マシンを Altera 社 Cyclone III スタータキット (FPGA: Cyclone III, EP3C25) 上に実装し, LE 数と最大動作周波数を求めた. 合成ツールは Altera 社 Quartus II version.9.1 を用いた. プログラムを格納するため, スタータキットの外付け SRAM (1 MBytes) を用いた. 分割 MTMDDs for CF マシンは 1 命令を 2 クロックで模擬するため, 1 ベクトル

表 2　プロセッサの比較.

Name	In	Out	FF	APL	MTMDDs for CF@100MHz				Atom N455@1.67GHz		Nios II@100MHz	
					LE 数	最大動作周波数 [MHz]	Delay [nsec]	PowerDelay [Wnsec]	Delay [nsec]	PowerDelay [Wnsec]	Delay [nsec]	PowerDelay [Wnsec]
s641	37	23	14	30.0	421	323	600	562.5	1450	13268.5	10392	45148.2
s713	37	23	14	30.3	371	363	607	569.0	1370	12535.5	11050	48006.6
s820	20	19	5	49.9	425	332	998	935.2	1610	14731.5	9100	39536.0
s1196	16	14	18	99.1	636	264	1983	1858.6	2300	21045.0	24974	108497.0
s5378	35	49	164	343.8	1471	186	6877	6444.6	12330	112819.5	43825	190397.0
s9234	36	39	211	151.8	874	222	3037	2845.7	4740	43371.0	51223	222535.8
s38417	28	106	1636	2010.5	3577	129	40210	37677.2	103850	950227.5	553705	2405513.6
Ratio							1.00	1.00	1.91	18.66	13.12	60.84

を評価する時間は $\frac{1}{f_{MDD}} \times APL \times 2$ である．ここで, f_{MDD} は分割 MTMDDs for CF マシンの動作周波数であり, APL は分割 MTMDDs for CF の平均パス長である．分割 MTBDD for CF を得るため, まず, 回路の Verilog-HDL 記述に対して Quartus II で LUT 最小オプションを与えて論理回路を生成した．次に, 分割 MTBDD for CF の節点数の総和が最小となるクラスタ分割を貪欲な手法 [16] を用いて求めた．MTBDD for CF はシフティング [17] を用いて最適化した．そして, メモリ量制約を 1 [MBytes] とし, ダイナミック・プログラミングを用いて分割 MTMDDs for CF を生成した．

b） Altera 社 Nios II

Altera 社組込み向け Nios II/fast プロセッサ (100 MHz 動作, キャッシュ: 128KB) を分割 MTMDDs for CF マシンと同じ FPGA ボードで動作させた．C コードは gcc コンパイラを用いて実行コードに変換した．最大最適化オプション -O3 を用いた．実行コードは分割 MTMDDs for CF マシンと同じ SRAM (1 MB) に格納した．消費電力を測定するため, FPGA ボードと電源の間に抵抗を挿入し, 瞬間消費電流を測定した．瞬間消費電流は 0.48 [A], 電源の電圧は 9.01 [V] であったため, 瞬間消費電力は 4.34 [W] であった．

c） Intel 社 Atom N455

Intel 社 Atom N455 プロセッサ (1.67 GHz 動作, キャッシュ: 512KB) が搭載されている組込み向け CPU ボード (Advantech 社 PCM-3363) を用いた．C コードは gcc コンパイラを用いて実行コードに変換し, 最大最適化オプション -O3 を用いた．消費電力を測定するため, CPU ボードと電源の間に抵抗を挿入し, 瞬間消費電流を測定した．瞬間消費電流は 1.83 [A], 電源の電圧は 5.0 [V] であったため, 瞬間消費電力は 9.15 [W] であった．

表 2 に実行時間と消費電力遅延時間積の比較結果を示す．分割 MTMDDs for CF マシンは変数選択回路を FPGA 上に直接実現したため, ベンチマーク毎に必要 LE 数, 最大動作周波数が異なる．今回の実験では, 全ての最大動作周波数が 100 MHz を超えていたので, 100 MHz に設定し, 遅延時間と消費電力を求めた．

表 2 より, 分割 MTMDDs for CF マシンは Nios II より 13.12 倍高速であり, Atom より 1.91 倍高速であった．また, 消費電力遅延時間積に関して, 分割 MTMDDs for CF マシンは Nios II より 66.84 倍小さく, Atom より 18.66 倍小さかった．従って, 分割 MTMDDs for CF マシンは電力効率に優れたマシンと言える．

7. ま と め

本論文では, 分割 MTMDDs for CF を模擬するマシンについて述べた．分割 MTMDDs for CF を間接分岐命令と出力命令で模擬する．節点間をジャンプする入力変数とレイル変数, 及び制御回路は FPGA で実現し, 命令セットは外付けメモリに格納する．他のプロセッサとの比較を行った結果, 分割 MTMDDs for CF マシンは Nios II より 13.12 倍高速であり, Atom より 1.91 倍高速であった．また, 消費電力遅延時間積に関して, 分割 MTMDDs for CF マシンは Nios II より 66.84 倍小さく, Atom より 18.66 倍小さかった．

8. 謝　　辞

本研究は, 一部, 日本学術振興会・科学研究費補助金, および, 文部科学省・知的クラスター創成事業 (第二期) の補助金による．

文　　献

[1] R. T. Boute, “The binary-decision machine as programmable controller,” *Euromicro Newsletter,* Vol. 1, No. 2, pp. 16-22, 1976.

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

[3] J. T. Butler, T. Sasao, and M. Matsuura, “Average path length of binary decision diagrams,” *IEEE Trans. on Compt.*, Vol. 54, No. 9, Sep. 2005, pp. 1041-1053.

[4] P. P. Gelsinger, “Microprocessors for the new millennium: Challenges, opportunities, and new frontiers,” *ISSCC '01*, pp. 22-25.

[5] Y. Iguchi, T. Sasao, M. Matsuura, and A. Iseno, “A hardware simulation engine based on decision diagrams,” *Asia and South Pacific Design Automation Conference (ASPDAC2000)*, Jan., 26-28, Yokohama, Japan, pp.73-76.

[6] J. Jain, J. Bitner, M. S. Abadir, J. A. Abraham, and D. S. Fussell, “Indexed BDDs: algorithmic advances in techniques to represent and verify Boolean functions,” *IEEE Trans. on Compt.*, Vol. 46, No. 11, Nov. 1997, pp. 1230-1245.

[7] T. Kam, T. Villa, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, “Multi-valued decision diagrams: Theory and applications,” *Multiple-Valued Logic*, Vol.4, no.1-2, 1998, pp.9-62.

[8] D. Mange, “A high-level-language programmable controller: Part I-II,” *IEEE Micro,* Vol. 6, No. 1, pp. 25-41 (Part I), Vol. 6, No. 2, pp. 47-63 (Part II), Feb/Mar, 1986.

[9] M. Matsuura, T. Sasao, J. T. Butler, and Y. Iguchi, “Bi-partition of shared binary decision diagrams,” *IEICE Trans. on Fund. of Electronics*, Vol.E85-A, No.12, Dec. 2002, pp.2693-2700.

[10] S. Nagayama and T. Sasao, “Compact representations of logic functions using heterogeneous MDDs,” *33rd IEEE Int'l Symp. on Multiple-Valued Logic (ISMVL2003)*, May, 2003, pp.247-255.

[11] H. Nakahara, T. Sasao, and M. Matsuura, “Multi-terminal multi-valued decision diagrams for characteristic function representing cluster decomposition,” (Draft).

[12] H. Nakahara, T. Sasao, and M. Matsuura, “A low power-delay product processor using multi-valued decision diagram machine,” (Draft).

[13] H. Nakahara, T. Sasao, and M. Matsuura, “A comparison of multi-valued and heterogeneous decision diagram machines,” *Journal of Multiple-Valued Logic*, (To be published).

[14] H. Nakahara, T. Sasao, and M. Matsuura, “Packet classifier using a parallel branching program machine,” *13th EUROMICRO Conf. on Digital System Design (DSD-2010)*, Lille, France, Sept., 2010, pp.745-752.

[15] H. Nakahara, T. Sasao and M. Matsuura, “A comparison of architectures for various decision diagram machines,” *ISMVL2010*, Barcelona, Spain, May, 26-28, 2010, pp.229-234.

[16] H. Nakahara, T. Sasao, and M. Matsuura, “A PC-based logic simulator using a look-up table cascade emulator,” *IEICE Trans. on Fund. of Electronics, Communications and Computer Sciences*, Vol. E89-A, No. 12, Dec., 2006, pp. 3471-3481.

[17] R. Rudell, “Dynamic variable ordering for ordered binary decision diagrams,” *IEEE/ACM Int'l Conf. on Computer-Aided Design (ICCAD1993)*, Nov., 1993, pp. 42-47.

[18] T. Sasao, H. Nakahara, M. Matsuura and Y. Iguchi, “Realization of sequential circuits by look-up table ring,” *The 2004 IEEE Int'l Midwest Symp. on Circuits and Systems (MWSCAS 2004)*, Hiroshima, July 25-28, 2004, pp.I:517-I:520.

[19] T. Sasao and M. Matsuura, “A method to decompose multiple-output logic functions,” *41st Design Automation Conference (DAC2004)*, June, 2004, pp. 428-433.

[20] S. Yang, “Logic synthesis and optimization benchmark user guide version 3.0,” *MCNC*, Jan. 1991.

[21] P.J.A.Zsombor-Murray, L.J. Vroomen, R.D. Hudson, Le-Ngoc Tho, and P.H. Holck, “Binary-decision-based programmable controllers, Part I-III,” *IEEE Micro* Vol. 3, No. 4, pp. 67-83 (Part I), No. 5, pp. 16-26 (Part II), No. 6, pp. 24-39 (Part III), 1983.