BDD分割を用いたパス・トランジスタ論理の合成

高田賢吾[†] 井上真一[†,☆] 沼昌宏[†]
瀧和男[†] 平野浩太郎[†]

BDD (Binary Decision Diagram) を用いてパス・トランジスタ論理回路を合成する場合，入力変数の数に比例するBDDの段数の増加によって，遅延時間が長くなる．また，"H" レベルを回復させるために挿入するバッファの数が増大する．そこで，BDDの任意のレベル区間を分離できるBDD分割を用いたパス・トランジスタ論理の合成手法を提案する．これによりBDDの段数を削減でき，遅延時間と挿入するバッファ数を削減することができる．

Synthesis of Pass Transistor Logic Circuits Based on Sliced BDD

KENGO TAKATA,[†] SHIN-ICHI INOUE,[†,☆] MASAHIRO NUMA,[†]
KAZUO TAKI[†] and KOTARO HIRANO[†]

In case of using BDD (Binary Decision Diagram) to synthesize pass transistor logic circuits, the circuit delay and the number of intermediate buffers increase according to the number of BDD stages, which is proportional to the number of primary inputs. We propose a synthesis technique for pass transistor logic based on sliced BDD, which is able to reduce the number of BDD stages, the circuit delay, and the number of transistors.

1. はじめに

近年の携帯機器の急速な普及や，LSIチップの発熱量の増加などから，低消費電力化を目的とするさまざまな研究が行われている[1),2)]．特に，パス・トランジスタ論理[3)~8)]が，CMOSスタティック回路に代わる有望な手法として注目されている．パス・トランジスタ論理は与えられた論理関数を，多くの場合，CMOSスタティック回路より少ないトランジスタ数で実現可能である．その結果，負荷容量が削減でき，低消費電力化を実現できる．

パス・トランジスタ論理の1つであるSPL (Single rail Pass-transistor Logic)[7)]は，動作速度よりも消費電力の削減を重視している．SPLのおもな特徴は，中間バッファ数を抑制したnMOS多段回路で構成されていること，トランジスタ数を削減するために，片方の論理をインバータで反転させることで正負，両論理を実現する，single-rail構造であること，などである．

† 神戸大学工学部
Faculty of Engineering, Kobe University
☆ 現在，川崎製鉄株式会社
Presently with Kawasaki Steel Corp.

BDD[9),10)]を用いてパス・トランジスタ論理を合成する手法がいくつか提案されている[3),6)~8)]．BDDの段数を削減することができれば，直列に接続されるトランジスタ数が減少し，遅延時間が削減される．さらに，挿入される中間バッファ数を削減できる．そこで，BDDの段数を削減することができるBDD分割を提案する．そして，BDD分割を用いたパス・トランジスタ論理の合成手法を提案する．

2. 定　義

ここでは，BDD[9),10)]，QRBDD (Quasi-Reduced BDD)[11)]，到達可能性行列[11)]を定義する．なおこれらの定義は，文献11)からの引用である．

2.1 BDD (Binary Decision Diagram)[9),10)]

$\mathcal{B}=\{0, 1\}$ 上の Ordered[10)] Shared[12)] BDDは，次のように定義される．

定義1 $\mathcal{B}$ 上のBDDは，七つ組 $\boldsymbol{B}=(X, N_V, N_C, I, e^0, e^1, lev)$ である．ただし，

$X=\{0,1,\cdots,n-1\}$ は変数のインデックスの集合．

N_V は変数ノードの集合．

$N_C=\{c_0, c_1\}$ は定数ノードの集合．

$I=\{i_1, i_2, \cdots, i_m\} \subset (N_V \cup N_C)$ は初期ノードの

集合.

$e^0, e^1 : N_V \to (N_V \cup N_C)$ はそれぞれノードから出る0エッジと1エッジを表す. ノード $v \in N_V \cup N_C$ は, $v \in I$, または $\exists u\,[e^0(u) = v \vee e^1(u) = v]$ を満たす.

$lev : (N_V \cup N_C) \to (X \cup \{n\})$ はノードのレベルを表し, $lev(v) = n$ iff $v \in N_C$, $lev(u) < lev(e^0(u))$, $lev(u) < lev(e^1(u))$ を満たす. □

変数順序については, レベルが大きいノードが定数ノードにより近くなるように定める. ノードの対 $(v, e^0(v))$, $(v, e^1(v))$ は, それぞれノード v の0エッジ, 1エッジと呼ばれる. BDDの各ノード v は,

$$f_{c_0} = 0\text{(恒偽関数)}, \quad f_{c_1} = 1\text{(恒真関数)},$$

$$f_v = \overline{x_{lev(v)}} \cdot f_{e^0(v)} + x_{lev(v)} \cdot f_{e^1(v)} \quad (v \in N_V)$$

によって定義される論理関数 $f_v : \mathcal{B}^n \to \mathcal{B}$ を表す. BDDの各初期ノードが表す関数の集合 $\{f_{i_1}, f_{i_2}, \cdots, f_{i_m}\}$ を, そのBDDが表現する関数という.

2.2 Reduced BDDと Quasi-Reduced BDD[11)]

変数ノード v が $e^0(v) = e^1(v)$ を満たすとき, v は冗長であるという. また, 2つの変数ノード v_1 と v_2 が $e^0(v_1) = e^0(v_2)$ かつ $e^1(v_1) = e^1(v_2)$ を満たすとき, v_1 と v_2 は等価であるという. 冗長なノードと等価なノードは, BDDが表現する論理関数を変更することなく除去することができる. この操作を**既約化**(reduction)と呼ぶ. 与えられた変数の全順序の下で, ある論理関数 f を表すBDDは複数存在するが, これらに既約化を行って得られるものは, 唯一であることが知られている[10)]. このようなBDDを *Reduced* BDDと呼ぶ.

BDD $\boldsymbol{B}$ のうち, すべての変数ノード v が

$$lev(e^0(v)) = lev(e^1(v)) = lev(v) + 1$$

を満たすものを, 密なBDDと呼ぶ. 任意のBDDは, 冗長なノードを追加することにより, 密なBDDに変形することができる. 密なBDDから等価なノードの削除を行って得られるものを, QRBDD (Quasi-Reduced BDD)[11)]と定義する. QRBDDの例を**図1**(a)に示す. 0エッジと1エッジは, それぞれノードの左下と右下に描かれる.

2.3 レベル間の到達可能性[11)]

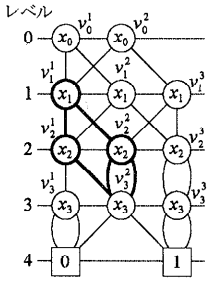
ベクトル $a = (a_0, a_1, \cdots, a_{n-1}) \in \mathcal{B}^n \quad (n = |X|)$ を, 変数ベクトル $x = (x_0, x_1, \cdots, x_{n-1})$ に対する**割当て**と呼ぶ. 割当ての下で, ノード間の**到達可能性**[11)]が定義される.

定義2 $u \xrightarrow{a} v$ は, 割当て a のもとでノード u から v に到達可能であることを示し, 以下のように定義される.

(1) $u \xrightarrow{a} u$.

(2) $e^b(u) = v \wedge b = a_{lev(u)}$ ならば $u \xrightarrow{a} v$.

(3) $u \xrightarrow{a} w$ かつ $w \xrightarrow{a} v$ ならば $u \xrightarrow{a} v$. □



(a) QRBDD

$$R_{1,3} = \begin{bmatrix} \overline{x_1}\,\overline{x_2} & \overline{x_1}x_2 + x_1 & 0 \\ \overline{x_1}\,\overline{x_2} & \overline{x_1}x_2 & x_1 \\ 0 & \overline{x_1} & x_1 \end{bmatrix}$$

(b) 到達可能性行列

図1 QRBDDと到達可能性行列
Fig. 1 QRBDD and reachability matrix.

2.4 到達可能性行列[11)]

定義3 $\boldsymbol{B}$ をQRBDDとする. $\boldsymbol{B}$ のレベル k, l のノード数をそれぞれ m_k, m_l とし, レベル k のノードを $\{v_k^1, v_k^2, \cdots, v_k^{m_k}\}$, レベル l のノードを $\{v_l^1, v_l^2, \cdots, v_l^{m_l}\}$ とする. **到達可能性行列** $R_{k,l}$ は, $m_k \times m_l$ 行列であり, その (i,j) 成分 $r^{i,j} : \mathcal{B}^n \to \mathcal{B}$ は,

$$r^{i,j}(a) = 1 \text{ iff } v_k^i \xrightarrow{a} v_l^j \tag{1}$$

と定義される. □

すなわち $R_{k,l}$ の成分 $r^{i,j}$ は, レベル k のノード v_k^i からレベル l のノード v_l^j へ到達できる割当てに対してのみ1となる論理関数である. たとえば, 図1(a)のBDDに対する到達可能性行列 $R_{1,3}$ は, 図1(b)のようになる. $R_{1,3}$ の成分 $r^{1,2} = \overline{x_1}x_2 + x_1$ は, ノード v_1^1 から v_3^2 への到達可能性を示す.

一般に, 隣接するレベル間の到達可能性を表す $R_{k,k+1}$ の要素 $r^{i,j}$ は, 変数 x_k のみに依存し,

$$r^{i,j} = \begin{cases} \overline{x_k} & (e^0(v_k^i) = v_{k+1}^j \wedge e^1(v_k^i) \neq v_{k+1}^j) \\ x_k & (e^0(v_k^i) \neq v_{k+1}^j \wedge e^1(v_k^i) = v_{k+1}^j) \\ 0 & (e^0(v_k^i) \neq v_{k+1}^j \wedge e^1(v_k^i) \neq v_{k+1}^j) \\ 1 & (e^0(v_k^i) = v_{k+1}^j \wedge e^1(v_k^i) = v_{k+1}^j) \end{cases} \tag{2}$$

のようになる. また $R_{k,h}, R_{h,l}$ が既知であれば, $R_{k,l}$ は

$$R_{k,l} = R_{k,h} \cdot R_{h,l} \tag{3}$$

により計算できる.

3. BDD分割

BDDの段数削減に効果的な, BDD分割を提案する. まずレベル区間を導入し, BDD分割の概念を述

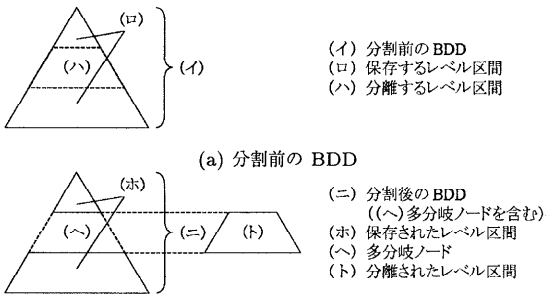


図 2 BDD 分割の概念
Fig. 2 Concept of sliced BDD.

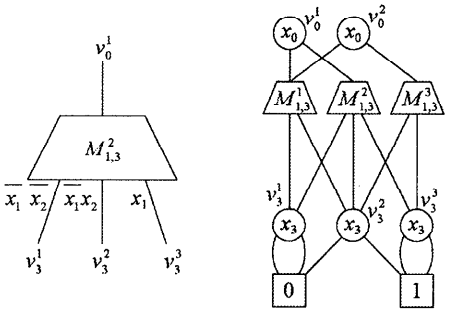


図 3 レベル区間の多分岐ノードによる置換
Fig. 3 Replacement of level band with multi-branch nodes.

べる．次に多分岐ノードを定義し，これによるレベル区間の置換方法を説明する．多分岐ノードによってレベル区間を置換することで，BDD を分割することができる．

3.1 BDD 分割の概念

定義 4 QRBDD $\boldsymbol{B}$ において，レベル区間ノード集合 $N_{V_{k,l}}$，レベル区間エッジ集合 $E_{k,l}$ を

$$N_{V_{k,l}} = \{v | v \in N_V, k \leq lev(v) < l\},$$
$$E_{k,l} = \{(v, e^0(v)), (v, e^1(v)) | v \in N_{V_{k,l}}\}$$

とする．レベル k とレベル l $(l < k)$ の間のレベル区間を，$L_{k,l} = (N_{V_{k,l}}, E_{k,l})$ の対と定義する． □

図 2 に BDD 分割の概念を示す．図 2 (a) において，(イ) 分割前の BDD より，(ハ) のレベル区間を分離する．なお，重複しない限り，複数のレベル区間を同時に分離できる．分割後の BDD を示す 図 2 (b) において，(ト) 分離した部分は，(ニ) 分割後の BDD とは独立した新たな BDD として再構成される（3.2 節 Step 1 参照）．(ホ) 保存された部分は (ヘ) 多分岐ノード（3.2 節参照）により接続される．この多分岐ノードの到達可能性は，再構成された (ト) により制御される．

3.2 多分岐ノードによるレベル区間の置換

定義 5 $V_{R_l}(v_k^i) = \{v_l^j\}$ を，ノード v_k^i から到達可能なレベル l のノードの集合とする．このとき，ノード v_k^i からそれぞれのノード $v_l^j \in V_{R_l}(v_k^i)$ への到達可能性を分岐数 $\left|V_{R_l}(v_k^i)\right|$ の多分岐ノード $M_{k,l}^i$ で表す．到達可能性行列 $R_{k,l} = \left[r^{i,j}\right]$ において，$M_{k,l}^i \xrightarrow{a} v_l^j$ iff $r^{i,j}(a) = 1$ である． □

図 3 (a) に多分岐ノードの例を示す．図の多分岐ノード $M_{1,3}^2$ は，図 1 (a) のノード v_1^2 からレベル 3 の各ノードへの到達可能性を表す．定義 5 より各分岐の到達可能性は，図 1 (b) の到達可能性行列 $R_{1,3}$ の，第 2 行の各成分によって決定される．

以下の手順により，多分岐ノードを用いてレベル区間 $L_{k,l}$ を置換できる．

Step 1：分離するレベル区間 $L_{k,l}$ に対応する到達可能性行列 $R_{k,l}$ の各成分を BDD で再構成する．

Step 2：分離するレベル区間 $L_{k,l}$ を多分岐ノードで置換する（図 3 (b) 参照）．

Step 1 で再構成された BDD は，多分岐ノードの分岐制御に用いられる．また，$L_{p,q}, L_{r,s}$ $(p < q \leq r < s)$ のような互いに重複しないレベル区間は，同時に多分岐ノードで置換することができる．

4. BDD 分割を用いたパス・トランジスタ論理の合成

ここでは，BDD 分割を用いてパス・トランジスタ論理を合成する手法について述べる．本論文では SPL[7] を合成対象とするが，BDD 分割の手法自体は，BDD に基づいて合成される他のパス・トランジスタ論理回路構成方式にも適用可能と考える．

まず，分離するレベル区間の選定方法について述べる．次に，パス・トランジスタ論理への適用方法を述べ，最後に本手法の効果を示す．

4.1 レベル区間の選定方法

分離するレベル区間の選定方法によって，トランジスタ数は大きく変化する．ここでは，自動選定と手動選定の 2 種類の選定方法を示す．

まずレベル区間の選定で用いる指標を定義する．

定義 6 分割前の BDD より合成された，パス・トランジスタ論理回路のトランジスタ数を T とする．レベル区間 $L_{k,l}$ を分離して合成したパス・トランジスタ論理回路のトランジスタ数を $S_{k,l}$ とする．**削減トランジスタ数 $D_{k,l}$** は，$D_{k,l} = T - S_{k,l}$ と定義される． □

ここで，レベル区間 $L_{k,l}$ を分離するとき，分離前のレベル区間（図 2 (a)(ハ)）の BDD ノード数を $N_{k,l}$，

分離されたレベル区間（図2(b)(ト)）から再構成されたBDDノード数を $C_{k,l}$，多分岐ノード（図2(b)(ヘ)）を実現するために必要となるトランジスタ数を $M_{k,l}$ とする．さらに，中間バッファを構成するトランジスタの総数に関して，レベル区間 $L_{k,l}$ の分離前に対する分離後の減少数を，中間バッファ・トランジスタ減少数 $I_{k,l}$ と呼ぶ．一般には，レベル区間 $L_{k,l}$ の分離による段数削減効果から，$I_{k,l} > 0$ となることが多い．

次節で述べるように，BDD 1ノードが2トランジスタに対応することから，削減トランジスタ数は

$$D_{k,l} = 2(N_{k,l} - C_{k,l}) - M_{k,l} + I_{k,l} \qquad (4)$$

と表される．ただし，レベル区間の選定時点では中間バッファ数は判断できないため，上記の $I_{k,l} = 0$ と仮定している．すなわち，$D_{k,l}$ には中間バッファ数の変化による影響は含まれない．

（1） **自動選定**

自動選定では，次のような手順で削減トランジスタ数 $D_{k,l}$ が大きいレベル区間から順に，すでに選定したレベル区間との重複がないレベル区間を選定する．

Step 1：レベル区間を $D_{k,l}$ $(0 \le k < l \le n)$ の降順にソートする．

Step 2：Step 1でソートした順に，同時に分離可能なレベル区間をすべて選定する．

レベル区間 $L_{k,l}$ を分離するとトランジスタ数が増加する場合，$D_{k,l}$ は負数となるが，中間バッファ・トランジスタ減少数 $I_{k,l}$ を考慮すると正数となる場合もある．よって，多少のトランジスタ数の増加は許容し，BDD段数の削減を優先して選定する．

（2） **手動選定**

自動選定ではBDD分割後に挿入される中間バッファを構成するトランジスタ数を考慮していないため，必ずしも適切な結果が得られるとは限らない．それに対して手動選定では，自動選定の結果や，各レベル区間を分離した場合の中間バッファ数などに関するデータをもとに，以下の2点を考慮して行われる．

(a) レベル区間の選定後に挿入される，中間バッファを構成するトランジスタの数

(b) クリティカル・パスの短縮

4.2 パス・トランジスタ論理への適用

BDDをもとにパス・トランジスタ論理を合成するには，まずBDDの0エッジ，1エッジのそれぞれを，nMOSトランジスタに置き換える．次に，ノード変数を x とすると，0エッジに対応するトランジスタのゲート入力に $\bar{x}$ を与え，1エッジに対応するトランジスタのゲート入力に x を与える．さらに，定数ノードの c_0 を V_{SS} で，c_1 を V_{DD} で置き換える．

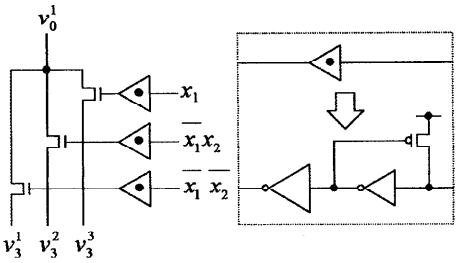
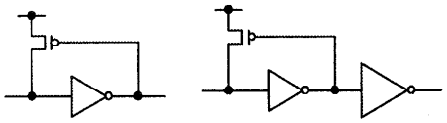


図4 多分岐ノードの実現方法
Fig. 4 Implementation of multi-branch node.



(a) 出力インバータ (b) 中間バッファ

図5 各種バッファの構成
Fig. 5 Buffers.

図3(a)の多分岐ノード $M_{1,3}^2$ は，**図4**のようにトランジスタで実現される．各トランジスタのゲート入力は，再構成されたBDD（3.2節 Step 1参照）をもとに合成したパス・トランジスタ回路によって制御される．

各出力には，レベル回復用に **図5**(a) の出力インバータが付加される．トランジスタの直列段数が大きい部分には，図5(b) の中間バッファが挿入される．

4.3 本手法の効果

ここでは，本手法の効果について述べる．

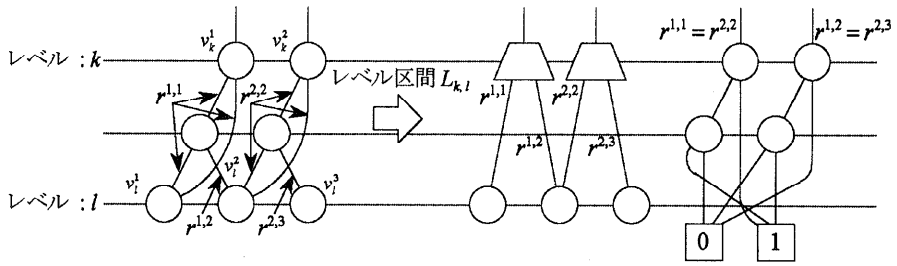
4.3.1 段数削減

レベル区間 $L_{k,l}$ を分離し，1段の多分岐ノードに置き換えることで，BDDの段数が $l-k+1$ だけ削減できる．これより，トランジスタの直列段数が減少し，遅延時間が削減できる．

4.3.2 同形パスに対応する制御論理の共有

図6(a) のレベル区間 $L_{k,l}$ に対応する到達可能性行列を $R_{k,l} = \left[r^{i,j}\right]$ とする．ただし図より，$r^{1,1} \equiv r^{2,2}$，$r^{1,2} \equiv r^{2,3}$ である．定義3より，$v_k^1 \xrightarrow{a} v_l^1$ iff $r^{1,1}(a) = 1$，$v_k^2 \xrightarrow{a} v_l^2$ iff $r^{2,2}(a) = 1$ である．ここで $r^{1,1} \equiv r^{2,2}$ であるため，$v_k^1 \xrightarrow{a} v_l^1$ iff $v_k^2 \xrightarrow{a} v_l^2$ である．これより，ノード v_k^1 からノード v_l^1 へのパスと，ノード v_k^2 からノード v_l^2 へのパスが同形であることが分かる．同様に，ノード v_k^1 からノード v_l^2 へのパスと，ノード v_k^2 からノード v_l^3 へのパスも同形である．

BDDでは，下位レベルの論理が異なれば，これら同形パスを共有することはできない．しかし，このようなレベル区間を分離し，独立したBDDとして再構



(a) 同形パス (b) 多分岐ノードに対する制御論理の共有

図 6 同形パスの共有
Fig. 6 Isomorphic logic paths and sliced result.

成することで，図 6 (b) のように多分岐ノードの制御論理を共有することができる．すなわち，到達可能性行列において論理的に等価な成分は，BDD 分割により共有することができる．

比較的大規模な BDD では，到達可能性行列に等価な成分が多く現れる場合がある．そのような場合に本手法を用いると，制御論理の共有によりトランジスタ数が削減され，より効果的である．

4.3.3 スニーク・パスの非存在性

BDD を用いてパス・トランジスタ論理回路を合成する場合，V_{DD} と V_{SS} の短絡であるスニーク・パスは生じない[3]．また BDD では，あるノード u からあるノード v に至るパスのうち，1 つの割当てに対応して到達可能となるパスは唯一である．さらに定義 3 より，$r^{i,j}(a)=1$ iff $v_k^i \xrightarrow{a} v_l^j$ である．これらより，到達可能性行列の同一行の成分は互いに排他的である．したがって，BDD 分割を用いたパス・トランジスタ論理回路の合成においても，スニークパスは存在しない．

5. 実験評価

文献 7) で述べられている SPL 用論理設計 CAD に，本手法を実装した．ここでは，同システムによる論理回路の合成実験，および回路シミュレータ HSPICE（MOS Level 3）による遅延時間と消費電力の評価結果について述べる．

5.1 実験回路例

SPL 用論理設計 CAD により，多くの MCNC ベンチマーク回路例[13] について良好な結果が得られている[7]．本論文では，そのうち遅延時間に改良の余地があると見込まれる回路例 9 例に対して実験を行った．SPL 用論理設計 CAD によって動的変数順序付けを行った BDD に対して本手法を適用し，評価を行った．

回路シミュレーションは，プロセスルールを 0.8 μm，主トランジスタサイズを $W=8\,\mu$m，電源電圧を 5 V とし，HSPICE（MOS Level 3）によって行った．その際，ファンアウト先の各トランジスタ 1 個につき，単位配線と同等の容量（27.6 fF）が存在すると仮定した．動作周波数を 10 MHz，測定時間を 100 ns とした．消費電力については，ランダムな 100 通りの入力パターンを与え，その平均をとった．

5.2 トランジスタ数と段数の評価

表 1 に，トランジスタ数と最大段数の結果を示す．「BDD」は従来の BDD を用いた手法[7] による合成結果を示し，「BDD 分割」は提案手法による合成結果を示す．「BDD 分割」の「自動」と「手動」はそれぞれ，4.1 節で示したレベル区間の選定法，「自動選定」と「手動選定」に対応する．「比率」は，従来法に対する BDD 分割を用いた場合の比率である．なおこれ以降，文中の平均は「幾何平均」を表す．

中間バッファの挿入位置決定において用いる最大遅延制約については，**表 2** に示す CMOS スタティック回路の遅延時間に設定した．なおトランジスタ数には，中間バッファ，および出力バッファ，さらに BDD 分割適用時には多分岐ノード，および制御論理など，回路を構成するために必要となるすべてのトランジスタ数を含んでいる．

自動選定では，最大段数が平均で 57%となった．これにより中間バッファ数も，全体で 42%に削減できた．一方で，トランジスタ数は平均 111% (+11%) に増加した．

一方，手動選定では最大段数が平均で 43%となった．中間バッファ数は全体で 20%に，トランジスタ数は平均で 88%に削減された．手動選定では，自動選定で得られた合成結果や BDD の段数からの予想により，中間バッファに使用されているトランジスタ数を考慮してレベル区間を選定している．その結果，自動選定より良好な結果が得られたと考えられる．

5.3 遅延時間の評価

表 2 に，遅延時間に対する結果を示す．「CMOS」は CMOS スタティック回路の遅延時間を示す．「BDD」，

表 1 トランジスタ数と最大段数
Table 1 Transistor counts and max number of stages.

回路名	入力数	出力数	段数					中間バッファ数			トランジスタ数				
			BDD	BDD 分割				BDD	自動	手動	BDD	BDD 分割			
				自動	比率	手動	比率					自動	比率	手動	比率
apex6	135	99	19	13	0.68	13	0.68	9	0	0	1726	2118	1.23	1911	1.11
comp	32	3	31	15	0.48	8	0.26	42	0	0	559	571	1.02	440	0.79
example2	85	66	15	9	0.60	10	0.67	11	3	3	913	1018	1.12	949	1.04
i2	201	1	135	68	0.50	29	0.21	130	59	12	1605	1575	0.98	1015	0.63
i3	132	6	31	15	0.48	8	0.26	56	24	0	814	1128	1.39	730	0.90
i4	192	6	46	26	0.57	16	0.35	56	22	4	1054	1273	1.21	878	0.83
too_large	38	3	35	18	0.51	11	0.31	66	21	2	1137	1044	0.92	949	0.83
x1	51	35	22	18	0.82	19	0.86	60	18	28	1431	1444	1.01	1385	0.96
x4	94	71	14	8	0.57	12	0.86	48	55	47	1335	1593	1.19	1390	0.93
幾何平均	–		–	–	0.57	–	0.43		–		–	–	1.11	–	0.88

表 2 HSPICE による遅延時間の測定
Table 2 Delay estimation by HSPICE.

回路名	遅延時間 [ns]							
	CMOS	BDD	BDD 分割					
			自動	対 CMOS	対 BDD	手動	対 CMOS	対 BDD
apex6	13.83	11.11	11.95	0.86	1.08	11.95	0.86	1.08
comp	10.02	12.76	12.38	1.24	0.97	3.63	0.36	0.28
example2	12.48	13.18	8.53	0.68	0.65	8.73	0.70	0.66
i2	9.40	53.72	27.77	2.95	0.52	8.81	0.94	0.16
i3	5.45	14.17	7.27	1.33	0.51	2.26	0.41	0.16
i4	10.79	15.30	8.20	0.76	0.54	5.35	0.50	0.35
too_large	13.16	16.64	12.07	0.92	0.73	16.11	1.22	0.97
x1	9.87	14.06	12.39	1.26	0.88	11.62	1.18	0.83
x4	10.03	12.31	9.26	0.92	0.75	9.02	0.90	0.73
幾何平均	–		–	1.10	0.71	–	0.72	0.47

「BDD 分割」は，それぞれの手法を用いて合成したパス・トランジスタ論理回路の遅延時間を示す．「対 CMOS」および「対 BDD」は，それぞれ「CMOS」および「BDD」の遅延時間に対する本手法で合成した回路の遅延時間の比率を示す．

自動選定において，遅延時間は BDD に対して平均 71%となり，表 1 の最大段数の削減効果が現れている．CMOS スタティック回路に対しては，遅延時間は平均 10%増加しているが，従来法では遅延制約を満足できない回路例について，制約を満たすことが可能となっており，本手法の効果が示されている．

手動選定では，さらに良好な結果が得られている．遅延時間が，BDD に対して平均 47%に，CMOS に対して平均 72%に短縮されている．表 2 において，apex6 の「対 BDD」や too_large，x1 の「対 CMOS」などでは効果が得られていないが，これはレベル区間の選定において，トランジスタ数の増加を抑えたためである．トランジスタ数の増加を許せば，遅延時間をさらに短縮することも可能である．

5.4 消費電力の評価

表 3 に消費電力に対する結果を示す．

自動選定では，BDD に対して平均 113%に，CMOS に対して平均 112%に増加した．このように，9 回路例全体に対する平均では消費電力が増加しているが，そのうち 6 例に対する平均では，81%に削減されている．

手動選定では，BDD に対して平均 96%，CMOS に対して平均 95%に削減されている．手動選定による中間バッファ数の減少効果により，バッファで消費される貫通電流が削減されたと考えられる．

6. ま と め

BDD 分割を用いたパス・トランジスタ論理の合成手法を提案した．本手法により，BDD の段数削減や，同形パスに対応する制御論理の共有によるトランジスタ数削減の効果が得られる．

本手法を SPL 用論理設計 CAD に実装し，評価した．分離するレベル区間を手動で選定した場合，BDD の段数で 43%，中間バッファ数で 20%，トランジスタ数

表 3 HSPICE による消費電力の測定
Table 3 Power estimation by HSPICE.

回路名	消費電力 [mW]							
	CMOS	BDD	BDD 分割					
			自動	対 CMOS	対 BDD	手動	対 CMOS	対 BDD
apex6	5.34	3.02	3.58	0.67	1.19	3.33	0.62	1.10
comp	1.04	0.88	0.84	0.81	0.95	0.67	0.64	0.76
example2	1.87	1.20	1.35	0.72	1.13	1.25	0.67	1.04
i2	0.63	1.03	1.78	2.83	1.73	1.22	1.94	1.18
i3	0.54	1.18	1.53	2.83	1.30	1.19	2.20	1.01
i4	1.69	1.99	2.05	1.21	1.03	1.45	0.86	0.73
too_large	1.54	1.46	1.37	0.89	0.94	1.32	0.86	0.90
x1	2.24	1.90	1.95	0.87	1.03	1.98	0.88	1.04
x4	2.83	2.42	2.63	0.93	1.09	2.44	0.86	1.01
幾何平均	–		–	1.12	1.13	–	0.95	0.96

で 88%に削減する結果が得られた．さらに，HSPICE による回路シミュレーションの結果，遅延時間について従来の BDD を用いた場合に対して 47%，CMOS スタティック回路に対して 72%，消費電力について BDD に対して 96%，CMOS に対して 95%に削減する結果が得られた．

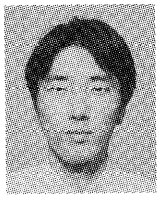
今後の課題としては，より効果的なレベル区間の自動選定手法の考案や，分割を前提とした BDD の変数順序の最適化などがあげられる．

謝辞 本研究において重要な位置を占める到達可能性行列についてご教授くださった，大阪大学大学院工学研究科情報システム工学専攻石浦菜岐佐助教授に深く感謝します．なお本研究の一部は，新エネルギー・産業技術総合開発機構平成 7 年度提案公募型・最先端分野研究開発事業の受託研究として行われた．

参 考 文 献

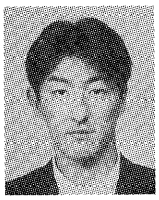
1) Devadas, S. and Malik, S.: A survey of optimization techniques targeting low power VLSI circuits, *32nd Design Automation Conf.*, pp.242–247 (1995).
2) 日経マイクロデバイス（編）：低電力 LSI の技術白書—1 ミリ・ワットへの挑戦，日経 BP 社 (1994).
3) Sakurai, T., Lin, B. and Newton, A.R.: Multiple-output shared transistor logic (MOSTL) family synthesized using binary decision diagram, *Dept. EECS, Univ. of Calif., Berkeley, ERL Memo M90/21* (1990).
4) Yano, K., Yamanaka, T., Nishida, T., Shimohigashi, K. and Shimizu, A.: A 3.8-ns CMOS 16×16-b multiplier using complementary pass-transistor logic, *IEEE Journal of Solid-State Circuits*, Vol.25, No.2, pp.388–395 (1990).
5) Sasaki, Y., Yano, K., Yamashita, S., Chikata, H., Rikino, K., Uchiyama, K. and Seki, K.: Multi-level pass-transistor logic for low-power ULSIs, *Proc. International Symposium on Low Power Electronics* (1995).
6) Yano, K., Sasaki, Y., Rikino, K. and Seki, K.: Top-down pass-transistor logic design, *IEEE Journal of Solid-State Circuits*, Vol.31, No.6 (1996).
7) Konishi, K., Kishimoto, S., Lee, B.-Y., Tanaka, H. and Taki, K.: A logic synthesis system for the pass-transistor logic SPL, *SASIMI '96*, pp.32–39 (1996).
8) Buch, P., Narayan, A., Newton, A.R. and Sangiovanni-Vincentelli, A.: On synthesizing pass transistor networks, *IWLS '97* (1997).
9) Akers, S.B.: Binary decision diagrams, *IEEE Trans. Comput.*, Vol.C-27, No.6, pp.509–516 (1978).
10) Bryant, R.E.: Graph-based algorithms for Boolean function manipulation, *IEEE Trans. Comput.*, Vol.C-35, No.8, pp.677–691 (1986).
11) Ishiura, N.: Synthesis of multilevel logic circuits from binary decision diagrams, *IEICE Trans. Inf. & Syst.*, Vol.E76-D, No.9, pp.1085–1092 (1993).
12) Minato, S., Ishiura, N. and Yajima, S.: Shared binary decision diagram with attributed edges for efficient Boolean function manipulation, *Proc. 27th ACM/IEEE Design Automation Conf.*, pp.52–57 (1990).
13) Yang, S.: Logic synthesis and optimization benchmarks user guide version 3.0. MCNC (1991).

(平成 10 年 9 月 21 日受付)
(平成 10 年 12 月 7 日採録)



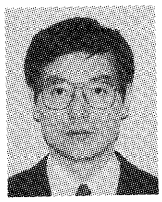
高田 賢吾（学生会員）

1974年生．1997年神戸大学工学部電気電子工学科卒業．1999年同大学大学院修士課程修了．同年，同大学大学院博士課程進学，日本学術振興会特別研究員．低消費電力回路設計技術に興味を持つ．



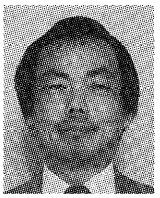
井上 真一（正会員）

1971年生．1995年神戸大学工学部電子工学科卒業．1997年同大学大学院修士課程修了．同年川崎製鉄（株）入社．在学中はLSI設計，低消費電力回路の論理合成に関する研究に従事．現在はLSI事業部に在籍．



沼 昌宏（正会員）

1960年生．1983年東京大学工学部精密機械工学科卒業．1985年同大学大学院修士課程修了．同大学助手を経て1989年同大学講師．工学博士．1990年5月より神戸大学大学院自然科学研究科講師．1995年同大学工学部電気電子工学科助教授．1996年文部省在外研究員として米国カリフォルニア大学サンタバーバラ校に派遣．主にLSI CAD，アクセラレータ，画像処理に関する研究に従事．IEEE，ACM，電子情報通信学会，エレクトロニクス実装学会各会員．



瀧 和男（正会員）

1952年生．1976年神戸大学工学部電子工学科卒業．1979年同大学大学院修士課程システム工学修了．工学博士．同年（株）日立製作所入社．1982年（財）新世代コンピュータ技術開発機構に出向．遂次型および並列型推論マシンと並列応用プログラムの研究開発に従事．1990年同機構第1研究室室長．1992年9月神戸大学工学部情報知能工学科助教授．1995年4月同学科教授．LSI設計技術とCAD，並列処理とマシンアーキテクチャ，脳型コンピュータ等に興味を持つ．電子情報通信学会，IEEE，ソフトウェア科学会，ACM，日本神経回路学会各会員．

平野浩太郎

1935年生．1960年大阪大学工学部通信工学科卒業．1962年同大学大学院修士課程修了．1965年同博士課程修了．工学博士．同年神戸大学工学部電気工学科助教授．現在は，同大学大学院自然科学研究科教授，同大学工学部電気電子工学科教授を兼任．ディジタル信号処理，ディジタル通信，LSI設計の研究に携わる．IEEE Fellow，Eta Kappa Nu，電子情報通信学会，システム制御情報学会各会員．