社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2005－SLDM－122 (5)
2005／11／30

非対称な信号遷移を用いた高速ダイナミック回路の論理合成手法

森本 薫夫† 永田 真† 瀧 和男††

† 神戸大学 大学院自然科学研究科，工学部情報知能工学科
〒 657-8501 神戸市灘区六甲台町 1-1
†† エイ・アイ・エル株式会社
E-mail: †{morimoto,nagata}@cs26.scitec.kobe-u.ac.jp, ††taki@ailabo.co.jp

あらまし スタティック CMOS で用いられている高度な自動合成環境を利用した高速ダイナミック回路 ASDDL の論理合成手法を提案する．論理合成は独自に考案した中間ライブラリを用いて行い，合成結果を ASDDL 回路に変換することで，ゲートレベルのネットリスト作成から配置配線までの全てを自動設計する．0.18-μm プロセスで設計した 16 ビット乗算器の遅延時間は 1.82nsec であり，エネルギー遅延積が最適になるように作成された CMOS ライブラリを用いて合成した乗算器に対して 32%改善した．さらに提案した論理合成手法により大幅に設計期間が短縮され，CMOS と同程度の設計時間で高性能な回路の設計が可能となった．

キーワード 論理合成, ASDDL, 非対称な信号遷移, 高速動作

Logic Synthesis Technique for High Speed Dynamic Logic with Asymmetric Slope Transition

Masao MORIMOTO†, Makoto NAGATA†, and Kazuo TAKI††

† Graduate School of Science and Technology,
Department of Computer and Systems Engineering, Kobe University
1-1 Rokkodai-cho, Nada-ku, Kobe 657-8501, Japan
†† AIL Co.,Ltd.
E-mail: †{morimoto,nagata}@cs26.scitec.kobe-u.ac.jp, ††taki@ailabo.co.jp

Abstract This paper proposes a logic synthesis technique for asymmetric slope differential dynamic logic (ASDDL) circuits. The technique utilizes a commercially available logic synthesis tool that has been well established for static CMOS logic design, where an intermediate library is devised for logic synthesis likely as static CMOS, and then a resulting synthesized circuit is translated automatically into ASDDL implementation at the gate-level logic schematic level as well as at the physical-layout level. A design example of an ASDDL 16-bit multiplier synthesized in a 0.18-μm CMOS technology shows an operation delay time of 1.82 nsec, which is a 32% improvement over a static CMOS design with a static logic standard-cell library that is finely tuned for energy-delay products. Design time for an ASDDL based dynamic digital circuit is sharply shortened by the proposed logic synthesis technique, and comparable with a static CMOS design.

Key words logic synthesis, ASDDL, asymmetric slope, high-speed operation

1. はじめに

スタティック CMOS の回路設計で論理合成が多く利用されるようになり，高度な自動設計環境を用いて高性能な回路設計ができるようになっている．しかしながら，さらに高速低消費電力な回路の要求も増加している．高速動作を実現するために従来の CMOS はトランジスタのチャネル幅 W を大きく設計する必要があるが，トランジスタの容量が増加してしまうため，W に比例した動作速度を得ることができない．

そこで，CMOS よりも高速に動作するダイナミック回路 [1] [2] が提案されている．これらの回路方式は設計した回路の全ての論理セルにプリチャージ制御のためのクロック信号を分配しなければならない．それに伴って，大量のクロックバッファを挿入する必要があり，面積，消費電力が増加する．そこで，我々は全

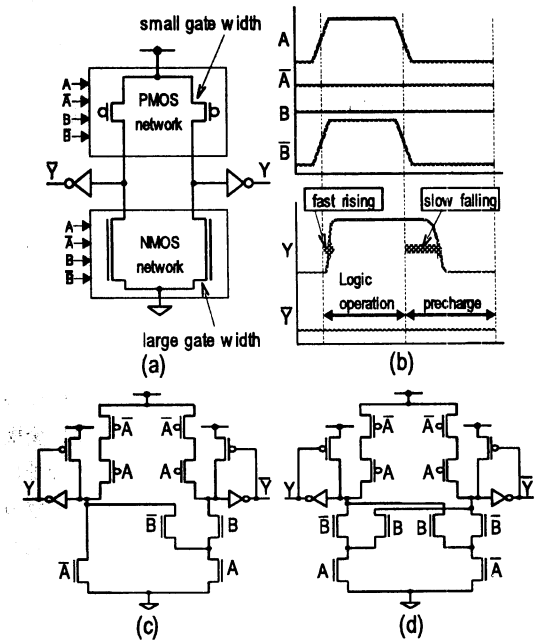

図 1 ASDDL circuit, (a) differential structure, (b) two-phase operation, circuit schematic of (c)NAND and (d)EXOR cell.

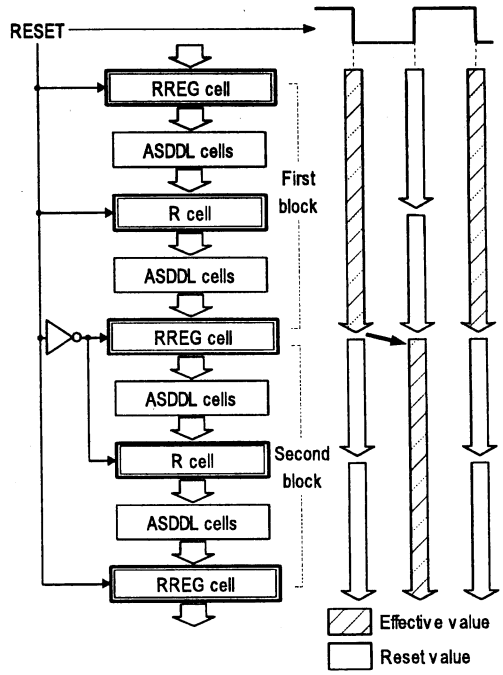

図 2 ASDDL architecture with cycle time reduction.

ての論理を立ち上がり遷移で表現し，回路のスイッチング動作における立ち上がり遷移時間を立ち下がり遷移時間よりも高速に設計した ASDDL (Asymmetric Slope Differential Dynamic Logic) を提案している [3]–[5]. ASDDL はプリチャージ制御のための信号線を必要としないため，従来報告されているダイナミック回路よりも高速かつ低消費電力となる [3].

本稿では ASDDL の論理合成手法を提案し，CMOS の回路設計で多く利用されている論理合成ツールを用いた自動設計環境を構築する．同時に CMOS 用の論理合成ツールを利用するために必要となる中間ライブラリを考案し，そのライブラリの合成結果を独自に開発した変換ツールで変換することにより ASDDL の自動設計を行なう．さらに，[6] で提案した手法よりもフルカスタム設計に非常に近い回路の合成を可能とし，ASDDL の特徴を最大限に生かした論理合成システムを実現する．

以下，第 2. 章では ASDDL の特徴，回路構成について説明し，サイクルタイムを短縮するためのアーキテクチャについて述べる．第 3. 章では ASDDL の論理合成手法について述べ，第 4. 章で設計したライブラリについて述べる．次に第 5. 章で設計事例およびその評価を示し，最後に第 6. 章でまとめる．

2. ASDDL 回路方式

2.1 回路構成

ASDDL は正論理と負論理の信号で論理値を表現する二線式論理回路である (図 1(a)) [3] [4]. 二線により表現される値は，有効値としての論理 0({0,1}) と論理 1({1,0}) および休止値 ({0;0}) の 3 種類であり，初期値として休止値を伝搬することで回路全体をリセットし，次に有効値を入力に与え順次後段に伝搬することで演算を行う (図 1(b)). 有効値を伝搬する前には必ず休止値を伝搬して回路をリセットすることから，回路の遅延時間は立ち上がりの伝搬 (有効値への遷移) 時間となり，立ち下がり遷移に対して立ち上がり遷移を非対称に最小化して設計する事で，回路全体の高速化を図る．この非対称な信号の遷移時間はトランジスタサイジングによって実現する．

図 1(c) と (d) に ASDDL の NAND セルと EXOR セルの例をそれぞれ示す．NMOS ネットワークの構成は，BDD 表現 (Binary Decision Diagram) を用いることで容易に設計でき，PMOS ネットワークはある 1 つの正負両論理 (例えば，{A,$\bar{A}$}) を入力に持つトランジスタを直列に接続した構成となる．これにより，その入力に休止値が到着すると PMOS ネットワークのトランジスタは" ON" 状態となり，回路は休止値を出力する．

2.2 サイクルタイム短縮アーキテクチャ

ASDDL は 1 回の演算時間 (遅延時間) は短いが，サイクルタイムが長くなる．これは休止値の伝搬時間は有効値の伝搬時間に比べて遅く，ASDDL のサイクルタイムは有効値の伝搬時間と休止値の伝搬時間の総和となることが原因である．これらの問題を解決するために，図 2 に示されているサイクルタイム短縮アーキテクチャを設計した回路に適用する [3] [5].

このサイクルタイム短縮アーキテクチャを適用するには，まず回路を入力側と出力側の 2 つのブロックに分割する．それぞれのブロックは有効値伝搬と休止値伝搬を交互に実行し，それらの動作の切り替えを"RESET" 信号で制御する．分割した回路ブロックの間にはリセット状態のときに休止値を新たに出力する機能と有効値を保持する機能を持った"RREG セル" が配置され，上記で説明した 2 つの動作を切り替えるインターフェースの役割を果たす．さらに，休止値伝播が有効値伝播よ

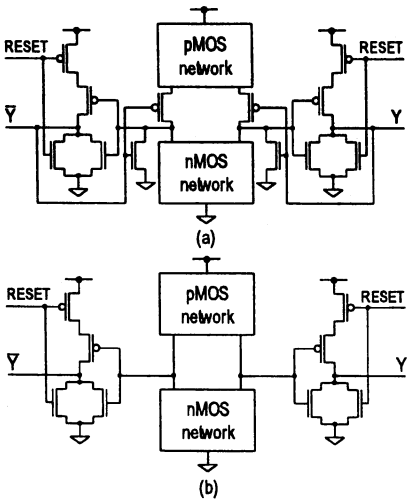

図 3 Circuit structures of (a) RREGcell, and (b) Rcell.

りも短くなるように，分割した回路ブロック内に新たに休止値を伝播する機能を持った"R セル" を配置する．RREG セルと R セルの回路構成を図 3(a) と (b) にそれぞれ示す．図に示されているように，RREG セルと R セルは ASDDL セルの出力インバータに数個のトランジスタを加えることにより，それぞれの機能を付加することができる．これにより，1 サイクルで回路全体に有効値と休止値の伝播を行うことができ，サイクルタイムは有効値の伝搬時間とほぼ等しくなる．

3. ASDDL 論理合成手法

3.1 合成フロー

ASDDL 独自の論理合成ツールを開発するには膨大なリソースを必要とするため，従来 CMOS で用いられている市販の論理合成ツールを利用する．しかしながら，CMOS 用の論理合成ツールでは正負両論理を持つ 2 線の入力をうまく合成できないため，中間ライブラリと呼ばれるライブラリを用いて合成する．これは CMOS 用の論理合成ツールを利用するために ASDDL セルを疑似的に定義したライブラリである．一方で，レイアウトを行うための配置配線用のライブラリを準備する．

以下に ASDDL 回路の自動合成設計フロー (図 4) を示す．

【ステップ 1】 中間ライブラリを用いて CMOS 用論理合成ツールで合成を行い，中間ネットリストを作成する．

【ステップ 2】 開発した変換ツールを用いて中間ネットリストを 2 線 2 相である ASDDL セルで構成された最終的なネットリストに書き換える．さらに，サイクルタイム短縮化手法を適用する．

【ステップ 3】 変換ツールによって作成された最終的なネットリストと配置配線用ライブラリを用いて，CMOS と同様に自動配置配線ツールで全体レイアウトを作成する．

3.2 中間ライブラリを用いた論理合成

市販の論理合成ツールでは 2 線の出力は定義できるが 2 線の入力は定義することができないため，中間ライブラリでは ASDDL セルを 1 線の入力と 2 線の出力を持った中間セルとして定義する．

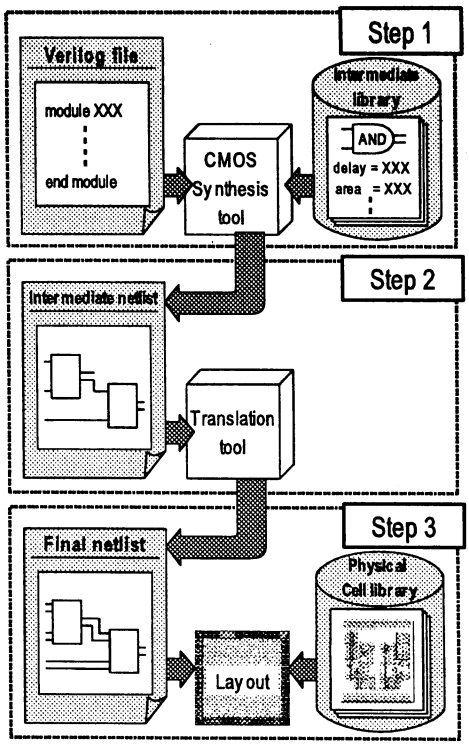

図 4 ASDDL logic synthesis flow.

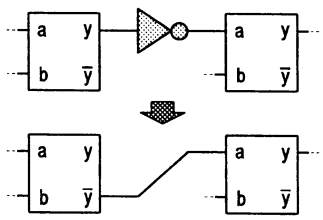

図 5 Inverter function on ASDDL circuits.

ここで，ASDDL セルは 2 線の信号線を入れ換えることによって正負両論理を表現できるため，インバータは必要なくなり，回路から簡単に削除することができる．しかしながら，合成後の回路からインバータが削除されるとセル間の駆動力と負荷容量の最適なバランスが崩れてしまう．また，インバータは論理合成ライブラリに必要不可欠であり，排除することができない．そこで，中間ライブラリではインバータの遅延時間や面積のパラメータを通常よりも大きく設定し，合成時にインバータの使用を制限する．このとき，もしも中間セル間の接続で反転論理を必要とする場合には，図 5 に示すように中間セルの負論理の出力信号が用いられる．そのため，合成した回路ではプライマリな入力にのみインバータが使用されるだけとなり，駆動力と負荷容量のバランスは維持された状態で回路を設計することができる．

3.3 ネットリスト変換

変換ツールは中間ネットリスト内の中間セルを ASDDL セルに置き換える．その時，セル間の信号線を適切に接続し直す．

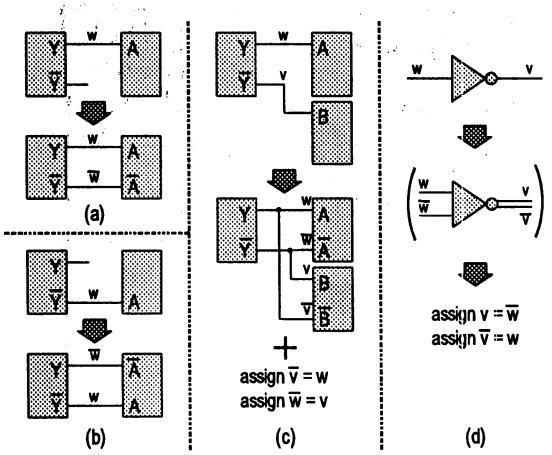

図 6 Translation for each pattern generated by logic synthesis using intermediate library.

中間ライブラリで合成すると中間セル間の信号線は図 6 に示すような接続パターンとなり，それらのパターンを基に接続し直すことで最終的なネットリストを作成する．

図 (a) のように正論理の出力 Y が後段セルの入力 A に配線 w で接続されている場合は，出力 $\overline{Y}$ と入力 $\overline{A}$ を配線 $\overline{w}$ で新たに接続する．負論理の出力 $\overline{Y}$ が後段セルの入力 A に配線 w で接続されている場合は，出力 Y と入力 $\overline{A}$ を配線 $\overline{w}$ で接続する (図 (b))．また，正論理の出力 Y と負論理の出力 $\overline{Y}$ の両方が後段セルの入力 A と B にそれぞれ配線 w と v で接続されている場合は，出力 $\overline{Y}$ と入力 $\overline{A}$ を配線 $\overline{w}$ で接続し，出力 Y と入力 $\overline{B}$ を配線 $\overline{v}$ で接続する (図 (c))．この時，配線 $\overline{v}$ と w，$\overline{w}$ と v はそれぞれ同じ配線であるため，"$assign\ \overline{v} = w$, $assign\ \overline{w} = v$" という記述を追加する．さらに，中間ネットリスト内で使用されたインバータは削除し，図 (d) のように "$assign\ v = \overline{w}$, $assign\ \overline{v} = w$" という記述を追加する．これらの変換は開発した変換ツールが自動的に変換を行う．

3.4 サイクルタイム短縮化手法の適用

サイクルタイム短縮化手法を適用する場合には適切な箇所に RREG セル，R セルを挿入する必要があり，これらも変換ツールが自動的に行う．

初めに，図 7(a) のように CMOS 用の論理合成ツールが備えているパイプライン化機能を用いてパイプラインレジスタを含んだ回路を合成する．このパイプラインレジスタが RREG セルを挿入するための目印となる．論理合成ツールのパイプライン化機構を用いることで，遅延時間が均等になるように回路を分割することができる．変換時には，パイプラインレジスタの前段に接続されたセルがその論理機能を持った RREG セルに置き換えられ (図 (b))，適切に RESET 信号が分配される．

さらに，休止値の伝搬は有効値の伝搬に比べて遅いため，以下のように分割したそれぞれの回路ブロックに R セルが挿入される (図 (c))．まず，回路の入力から順に休止値の伝搬時間が計算され，休止値伝搬時間がその回路ブロックの最大有効値伝搬時間よりも遅いセルが R セルに置き換えられる．休止値伝搬時間は置き換えられた R セルから計算し直され，全てのセルの

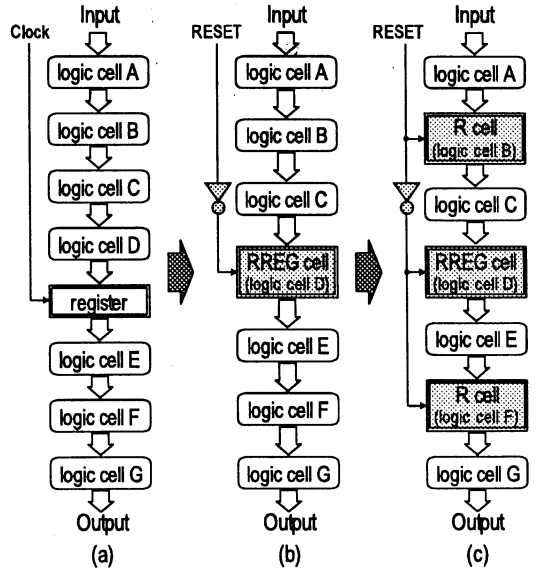

図 7 Applying cycle-time reduction technique.

表 1 The type of logic cells in ASDDL intermediate library.

Name	Logic	Rank
AD2A	$A+B$	2X 3X 4X
AD2B	$A \oplus B$	2X 3X 4X
AD3A	$A \cdot B \cdot C$	2X 3X 4X
AD3B	$A \oplus B \oplus C$	2X
AD3C	$(A \oplus B) \cdot C$	2X
AD3D	$A + B \cdot C$	2X 3X 4X
AD3E	$A \cdot B \cdot C + \overline{A} \cdot \overline{B} \cdot \overline{C}$	2X
AD3F	$A \cdot B + \overline{A} \cdot C$	2X 3X 4X
AD3G	$A \cdot B + B \cdot C + C \cdot A$	2X
AD3H	$A \cdot B \cdot C + \overline{A} \cdot \overline{B} + \overline{A} \cdot \overline{C}$	2X
AD3I	$A \cdot B + \overline{A} \cdot \overline{B} \cdot C$	2X
AD3J	$A \cdot B \cdot C + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot C$	2X
BUF		0X 1X 2X 3X 4X 5X 6X 7X 8X

休止値伝搬時間がその回路ブロックの最大有効値伝搬時間よりも短くなるまでこの工程が繰り返される．この方法は [6] で提案した論理合成システムを発展させたものであり，図 3 のような論理機能を持った RREG セル，R セルを挿入することで非常にフルカスタム設計に近い ASDDL 回路が合成可能となる．

4. ライブラリの設計

電源電圧 1.8V，0.18-μm プロセスにて中間ライブラリおよび配置配線用ライブラリを作成した．作成したライブラリのセル種類を表 1 に示す．セルの駆動力は小さい順に 0X，1X$\cdots$8X で表している．また，全てのセルに対して RREG セル，R セルを用意した．中間ライブラリ内で定義した中間セルの入出力ピンパラメータは正負信号の最悪値とし，合成した回路の遅延時間は最悪値となるようにした．これにより変換ツールにより信号線の再接続が行われたとしても合成時の回路よりも遅延時

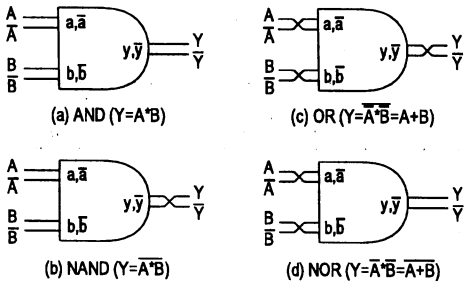

図 8 The variation of logic function in ASDDL logic cell.

間が悪化することはない.

ここで, ASDDL セルは入力や出力信号の 2 線を入れ換えることにより, 1 つのセルで複数の論理を表現できる. 図 8 は入力や出力信号を入れ換えることで (a)AND, (b)NAND, (c)OR, (d)NOR の論理を表現できることを示している. これらの特徴から ASDDL のライブラリは少ないセル数で構成することが可能であり, 作成したライブラリにおいても 3 入力以下の全ての論理を 12 種類のセルで表現できる [6]. 配置配線用ライブラリについても同様に 12 種類のセルで構成されており, 各セルのレイアウトを駆動力ごとに用意した.

5. 設計事例とその評価

5.1 設計事例

提案した論理合成手法を用いてアーキテクチャや合成時の制約条件が異なる 3 種類の 16 ビット乗算器を設計した. 設計した 16 ビット乗算器のアーキテクチャ, 制約条件を以下に示す. ここで, Acla と Fcla はそれぞれ area-optimized fast-carry-lookahead adder, fast-carry-lookahead adder を表す.

- M16_Acla_A
  - Wallace tree, Acla, 面積最小
- M16_Fcla_D
  - Wallace tree, Fcla, 遅延時間最小
- M16_D
  - アーキテクチャ指定なし, 遅延時間最小

0.18μm プロセスで作成した中間ライブラリを用いて論理合成した 16 ビット乗算器で使用されたセルの個数を表 2 に示す. 多く使用されているのは AD2A(2-input NAND) や AD3B(3-input EXOR) などのセルであるが, AD3F($A \cdot B + \overline{A} \cdot C$) や AD3J ($A \cdot B \cdot C + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot C$) のような複雑な論理を持ったセルも使用されている. これらのセルは CMOS では容易に実現できないが ASDDL では BDD を用いることで容易に実現でき, 複数の論理セルを 1 つのセルにできる上で高速, 低消費電力に向いている. しかしながら, フルカスタム設計では一見しただけでは用途が分からないためなかなか使用できないことから, 論理合成によってこのようなセルが使用されることは ASDDL を論理合成する上での大きなメリットであり, 2 線回路である ASDDL の特徴を最大限に引き出しているといえる.

5.2 シングルサイクル回路の評価

ASDDL 論理合成手法の有効性を検証するために, 以下の 5

表 2 Number of logic cells used in the 16-bit multipliers.

Cell name	M16_Acla_A	M16_Fcla_D	M16_D
AD2A	384	1062	874
AD2B	40	138	192
AD3A	28	89	30
AD3B	239	213	216
AD3C	0	0	3
AD3D	176	178	140
AD3E	0	0	0
AD3F	16	56	49
AD3G	174	101	192
AD3H	0	0	0
AD3I	0	0	0
AD3J	1	5	2
BUF	0	0	18
Total	1058	1842	1716

表 3 Comparison of 16-bit multipliers in 0.18-μm technology at 1.8V.

	Delay [nsec]	Power [mW]	Area [mm^2]	Design Time
CMOS_M16	2.68	5.99	0.041	1hour
FC_M16	1.80	14.69	0.055	2 weeks
M16_Acla_A	2.00	13.32	0.061	1 hour
M16_Fcla_D	1.82	26.22	0.102	1 hour
M16_D	2.11	24.03	0.098	1 hour

つの回路を対象に比較評価を行った.

- CMOS_M16
  - スタティック CMOS で設計した乗算器
- FC_M16
  - フルカスタムで設計した ASDDL 乗算器
- Section 5.1 で論理合成した 3 種類の乗算器
  - M16_Acla_A, M16_Fcla_D and M16_D

CMOS_M16 はエネルギー遅延積および面積が最適になるように作成された CMOS ライブラリを用い, 遅延時間最小の条件で回路を合成した. FC_M16 は従来通りフルカスタム設計で設計した乗算器であり, Wallace tree と BLCA(Binary look-ahead carry adder) で構成した. また, FC_M16 および提案手法を用いて作成した 3 つの乗算器にサイクルタイム短縮化手法は適用されていない.

それぞれの回路を自動配置配線後に RC 抽出し, SPICE シミュレーションにより評価した結果を表 3 に示す. 電源電圧は 1.8V であり, 消費電力は 100MHz 動作時の平均消費電力である. また, フルカスタム乗算器 (FC_M16) を 1 としたときの性能比を図 9 に示す. M16_Fcla_D の遅延時間は 1.82nsec であり, これは最も高速な CMOS 設計と比べて 32%減少した. さらにフルカスタム設計の FC_M16 と比べても 1%の増加に押えることができ, フルカスタム設計に近い高速動作を実現した. 一方で, ASDDL は高速動作を実現するために全てのセルが毎クロックごとに立ち上がりと立ち下がりの 2 つの遷移を行うた

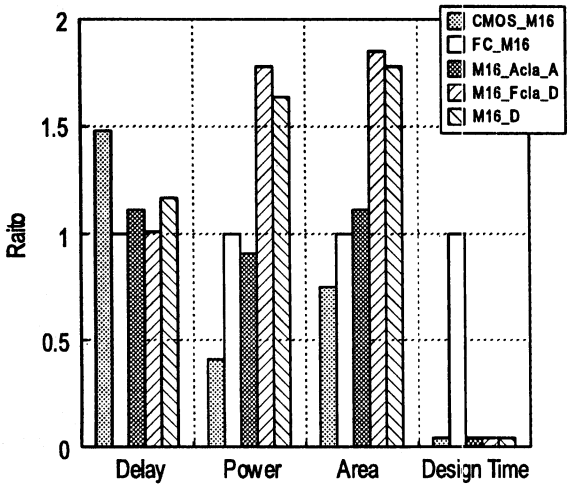

図 9 Performance ratio of 16-bit multipliers in 0.18-μm technology at 1.8V. The ratio is calculated as FC_M16 to be 1.00.

め，CMOS と比べて消費電力は増加する．しかしながら，論理合成によって複雑な論理を持ったセルが多く使用されたために，M16_Acla_A の消費電力は FC_M16 の 98%であった．また，一般的な論理合成ツールは僅かな遅延時間を削除するために面積を余計に大きくする傾向があり [7]，遅延最小条件で合成した M16_Fcla_D と M16_D の面積は FC_M16 に比べてそれぞれ 79%，76%増加した．

最も注目すべき点は設計期間の短縮である．フルカスタム設計では 2 週間程度の設計期間を要したのに対して，提案した ASDDL 論理合成手法を用いて設計した 16 ビット乗算器は 1 時間足らずで設計することができた．これより，短期設計を必要とする SoC などでも利用でき，高速動作を要求するデータパスなどのさらに性能向上が期待できる．

5.3 マルチサイクル回路の評価

以下に示した ASDDL16 ビット乗算器にサイクルタイム短縮化手法を適用した．

- S_M16_Acla_A, S_M16_Fcla_D and S_M16_D
  - M16_Acla_A，M16_Fcla_D，M16_D にサイクルタイム短縮化手法を適用した回路
- S_FC_M16
  - FC_M16 にサイクルタイム短縮化手法を適用した回路

表 4 にサイクルタイム短縮化手法を適用した乗算器の性能を示す．これは自動配置配線後に RC 抽出し，電源電圧 1.8V，SPICE シミュレーションにより評価した結果であり，消費電力は 100MHz 動作時の平均消費電力である．サイクルタイム短縮化手法が適用されることによって RREG セルと R セルが追加されたが，それによる遅延時間および面積の増加は 10%未満に押えることができた．また，サイクルタイムは有効値の伝搬時間と休止値の伝搬時間の総和であったが，サイクルタイム短縮化手法を適用することで回路の休止値伝搬は有効値伝搬の裏に隠れるため，最小のサイクルタイムは遅延時間にほぼ等しくなる．これより，設計した乗算器の最小サイクルタイムはシングルサイクルの乗算器の 50%以下にでき，パイプライン動作を見越した大規模回路の設計にも有用である．

表 4 Comparison of 16-bit multipliers by applying cycle-time reduction technique in 0.18-μm technology at 1.8V.

	Delay [nsec]	Minimum Cycle [nsec]	Power [mW]	Area [mm^2]
FC_M16	1.80	5.71	14.69	0.055
S_FC_M16	1.88	1.88	15.70	0.058
M16_Acla_A	2.00	5.16	13.32	0.061
S_M16_Acla_A	2.06	2.06	15.39	0.067
M16_Fcla_D	1.82	5.87	26.22	0.102
S_M16_Fcla_D	2.04	2.04	27.17	0.104
M16_D	2.11	6.29	24.03	0.098
S_M16_D	2.23	2.23	25.41	0.102

6. おわりに

信号の立ち上がり遷移と立ち下がり遷移を意図的に非対称とすることで高速化を図った高速ダイナミック回路 ASDDL の論理合成手法を提案した．独自に考案した中間ライブラリと変換ツールを用いることにより，CMOS 用の論理合成ツールを利用した ASDDL の自動設計を可能とした．

0.18μm プロセス，電源電圧 1.8V のシミュレーション結果では，提案した論理合成手法を用いて設計した 16 ビット乗算器の遅延時間は 1.82nsec であった．これはエネルギー遅延積が最適になるように作成された CMOS ライブラリを用いて論理合成した乗算器と比較して 32%改善した．さらにフルカスタム設計では 2 週間かかる設計を提案した論理合成手法により 1 時間足らずでの設計することができ，短期間で様々な制約条件と備えた高性能な回路の設計が可能となった．

謝辞 本研究は東京大学大規模集積システム設計教育研究センターを通し，シノプシス株式会社の協力で行われたものである．

文　献

[1] P.Ng, P.T.Balsara, and D.Steiss, "Performance of CMOS Differential Circuits", IEEE J. Solid-State Circuits, vol.31, no.6, pp.841–846, Jun 1996.

[2] K.M.Chu and D.L.Pulfrey, "A Comparison of CMOS Circuit Techniques: Differential Cascode Voltage Switch Logic Versus Conventional Logic", IEEE J. Solid-State Circuits, vol.22, no.4, pp.528–532, Aug 1987.

[3] Masao Morimoto, Makoto Nagata, Kazuo Taki, "High-Speed Digital Circuit Design using Differential Logic with Asymmetric Signal Transition", IEICE Trans. on Electronics, vol.E88-C, no.10, pp.2001–2008, Oct 2005.

[4] 瀧和男, 八木幹雄, 森本薫夫, 尾形俊郎, 池見憲一, 北村清志, "高速低消費電力回路方式 ASDDL/ASD-CMOS とその評価", DA シンポジウム 2001 論文集, pp.113–118, Jul 2001.

[5] 八木幹雄, 森本薫夫, 瀧和男, 北村清志, "高速低消費電力論理回路方式 ASDL のパイプライン化手法とその評価", デザインガイア 2001.

[6] 田中義則, 森本薫夫, 永田真, 瀧和男, "高速論理回路方式 ASDDL/ASD-CMOS の論理合成手法", 電子情報通信学会技術報告 ICD2003-235, pp.37–42, 2004

[7] 株式会社半導体理工学研究センター, "RTL 設計スタイルガイド Verilog-HDL 偏", 株式会社半導体理工学研究センター, 2003.