非対称な信号遷移を用いた高速論理回路に関する研究

森本，薫夫

(Degree)
博士（工学）

(Date of Degree)
2007-03-25

(Date of Publication)
2015-11-20

(Resource Type)
doctoral thesis

(Report Number)
甲4005

(URL)
https://hdl.handle.net/20.500.14094/D1004005

博　士　論　文

非対称な信号遷移を用いた高速論理回路に関する研究

平成19年2月

神戸大学大学院自然科学研究科

森本　薫夫

要旨

LSIの高集積化により，多種多様な機能を持った回路が実現できるようになり，回路内で処理しなければならないデータ量も増加する傾向にある．この膨大なデータを高速に処理するために，デジタルLSIの高速化技術は今後の設計開発において無くてはならないものとなっている．高速デジタルLSIの設計には，個々のシステムに応じたアーキテクチャの開発が重要である．アーキテクチャによる高速化アプローチとしては，並列処理やパイプライン処理，加算や乗算のための演算アーキテクチャなどがあり，これらを効率良く適用することで回路の高速化を実現できる．しかしながら，回路を構成する論理ゲートが最適に設計されていなければ，どんなに良いアーキテクチャを適用したとしても，設計した回路の性能を最大限に引き出すことはできない．そのため，論理ゲートレベルでの新しい回路設計手法の提案が必要不可欠である．

現在，デジタルLSI設計の多くで用いられているスタティックCMOS回路では，動作速度を高めるために，論理ゲートを構成するトランジスタのチャネル幅Wを大きく設計する．しかしながら，トランジスタの負荷容量が増加してしまうため，トランジスタのWに比例した動作速度を得ることができない．そこで，スタティックCMOSよりも高速動作可能な回路方式として，CMOS-DOMINOやDCVS-DOMINO回路方式などが考えられている．これらの回路方式は，遷移確率の高いプリチャージ制御のためのクロック信号を全ての論理ゲートに分配しなければならないため，消費電力が増加する．このプリチャージ制御のためのクロック信号を大幅に削減できる回路方式として，立ち上がり遷移を高速化したNORゲートと立ち下がり遷移を高速化したNANDゲートのみで回路を構成するSCSLが提案されている．SCSLはNANDとNORゲートのみで回路を構成するため，論理ゲートの段数が多くなってしまう．このため，論理ゲートの遅延時間に見合った回路性能を引き出すことができない．

本研究では，従来方式よりも高速，かつ低消費電力を実現する回路方式を用いて，大規模デジタルLSIの飛躍的な性能向上を実現することを目的とする．この目的を達成するため，論理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することで，スタティックCMOSに比べて格段に短い遅延時間を達成する

ASDDL(Asymmetric Slope Differential Dynamic Logic) と ASD-CMOS (Asymmetric Slope Differential CMOS) を提案する．ASDDL/ASD-CMOS は正論理と負論理の信号で論理値を表現する 2 線式論理回路であり，演算とプリチャージを交互に実行する．プリチャージは演算と同様に前段から後段の論理ゲートへ向けて順次伝搬させるため，SCSL と同様にプリチャージ制御のための信号線の分配を無くすことができ，従来の高速回路方式よりも低消費電力を実現する．また，ASDDL/ASD-CMOS 論理ゲートは BDD 表現を用いて設計を行なうことで，複雑な論理関数も 1 つの論理ゲートで設計することが可能であり，設計した回路の論理ゲート段数を大幅に減らすことができる．しかしながら，ASDDL/ASD-CMOS は演算とプリチャージを交互に実行する必要があるため，サイクルタイムが演算の遅延時間とプリチャージの遅延時間の総和となる．そこで，大幅にサイクルタイムを短縮することができるサイクルタイム短縮アーキテクチャを考案した．このアーキテクチャは入力側と出力側の 2 つの回路ブロックに分割し，それぞれの回路ブロックで演算とプリチャージを交互に行なう．これにより，全ての論理ゲートのプリチャージを演算の裏に隠すことができ，サイクルタイムは演算時間とほぼ等しくなる．

0.18-μm CMOS プロセス，電源電圧 1.8V にて，スタティック CMOS と DCVS-DOMINO との比較評価を行った．ASDDL と ASD-CMOS で設計した 16 ビット符号付き乗算器の遅延時間はそれぞれ 1.82ns，1.78ns であった．これは，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ライブラリを用いて，遅延時間最小の条件で設計したスタティック CMOS と比べて 32%，34%減少した．これらの遅延時間は DCVS-DOMINO の 96%，94%であり，ASDDL と ASD-CMOS は DCVS-DOMINO よりも高速に動作することを確認した．また，消費電力は DCVS-DOMINO に対してそれぞれ 20%，2%削減できた．さらに，0.13-μm CMOS プロセス，電源電圧 1.2V で試作したテストチップでは，ASD-CMOS 乗算器は 1.57ns で動作していることを確認した．

ASDDL/ASD-CMOS は，最も高速動作が可能とされる回路方式の 1 つである DCVS-DOMINO よりも高速・低消費電力を実現する．しかしながら，フルカスタム設計では大規模デジタル LSI への適用に膨大な設計コストが必要となる．そこで，ASDDL/ASD-CMOS の論理合成手法を提案し，スタティック CMOS の回路設計で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．しかしながら，スタティック CMOS 用の論理合成ツールでは，正負両論理の 2 線の入出力信号を持った論理ゲー

トをそのまま用いて合成することができない．そこで，独自に考案した中間ライブラリを用いて合成し，変換ツールを用いて合成結果に後処理を施すことで，ゲートレベルのネットリスト作成から配置配線まで全てのフローにおける自動設計を実現した．考案した中間ライブラリでは，2線の信号線を持ったASDDL/ASD-CMOS論理ゲートを入力は1線，出力は2線の中間セルで定義し，2線式論理回路の特徴を最大限に生かした論理合成を可能とした．実際に構築したASDDL用ライブラリは12種類の論理ゲートで構成しており，3入力以下の全ての論理関数を表現できるようした．変換ツールは中間セルで構成された合成結果をASDDL/ASD-CMOS論理ゲートに置き換え，信号線の再接続を行なう．さらに，サイクルタイム短縮アーキテクチャを自動で適用する機構を備えている．

構築した自動設計環境で，様々な制約条件を持ったASDDLの16ビット符号付き乗算器を0.18-μm CMOSプロセスで設計し，比較評価を行なった．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ASDDL乗算器の遅延時間は1.82nsであった．これはスタティックCMOS乗算器と比べて32%の改善であり，フルカスタム設計の乗算器と同等の高速動作を実現した．さらに，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，提案した論理合成手法を用いることにより1時間足らずで設計でき，ASDDL/ASD-CMOSを適用した大規模デジタルLSIの短期設計を可能とした．

ASDDL/ASD-CMOSは演算と同様にプリチャージを回路前段から順に伝搬させるため，プリチャージ制御のための信号線を大幅に削減でき，DCVS-DOMINOよりも低消費電力を実現する．しかしながら，プリチャージを必要としないスタティックCMOSと比べると消費電力は増大する．そこで，通常はASDDL/ASD-CMOSと同様に高速で動作するが，処理量が少ない場合には消費電力を引き下げることのできるASDMDL(Asymmetric Slope Dual Mode Differential Logic)を提案する．ASDMDLは，ASDDL/ASD-CMOSと同様に演算の前にプリチャージを行なう2相で動作する高速モードと，スタティックCMOSと同様にプリチャージなしの1相で動作する低消費電力モードという2つの動作モードを1つの回路で実現する2モード2線式論理回路である．この2つの動作モードを動作中の各タイミングによって切り替えることで，設計した回路の限界性能を引き出すことができる．

ASDDL/ASD-CMOSの論理合成手法を用いて，ASDMDLとスタティックCMOSを混載したデジタルコアの論理合成・自動配置配線を実現し，

クリティカルパス部分を中心にASDMDLを適用したASDMDL/CMOS混在プロセッサを0.18-μm CMOSプロセスにて試作し，性能検証を行なった．テストチップの性能比較では，高速モード時の最高動作周波数は232MHzであり，スタティックCMOSと比べて14%向上し，低消費電力モードの性能はCMOSと同等の性能を実現した．これにより，ASDMDLはASDDL/ASD-CMOSの高速動作とスタティックCMOSの低消費電力動作という2つの特性を1つの回路で実現できることを実証した．

以上より，スタティックCMOSでは到達不可能な高速動作が実現でき，従来提案されている高速回路方式の性能を凌駕する新たな回路方式を提案した．また，構築した自動設計環境を用いることによって，様々な制約条件，アーキテクチャを持った回路を短期間で設計することができるだけでなく，プロセッサの制御回路などのようなランダムロジックの設計を実現可能とした．これらの設計技術によって，大規模デジタルLSIの性能を飛躍的に向上が期待できる．

目次

1 序論 1
1.1 研究の主旨 1
1.2 アーキテクチャによる高速化アプローチ 2
1.3 スタティック CMOS 回路 4
1.3.1 スタティック CMOS の回路構成と動作速度 4
1.3.2 低しきい値電圧を用いた回路の高速化 8
1.4 従来研究 9
1.4.1 CMOS-DOMINO 回路方式 9
1.4.2 NORA 回路方式 11
1.4.3 DCVS-DOMINO 回路方式 13
1.4.4 SCSL 回路方式 14
1.5 研究の概要と本論文の構成 16
1.5.1 非対称な信号遷移を用いた高速論理回路方式 16
1.5.2 2 線 2 相式論理回路の自動設計システム 17
1.5.3 2 つの動作モードを有する高速論理回路方式 18

2 非対称な信号遷移を用いた高速論理回路方式 21
2.1 まえがき 21
2.2 高速論理回路方式 ASDDL/ASD-CMOS 22
2.2.1 ASDDL/ASD-CMOS の特徴 22
2.2.2 ASDDL と ASD-CMOS の回路構成と動作 24
2.3 サイクルタイム短縮アーキテクチャ 28
2.3.1 アーキテクチャの構成 28
2.3.2 Rcell と RREGcell の回路構成 32
2.4 ASDDL/ASD-CMOS の回路性能評価 34
2.4.1 シミュレーションによる性能評価 34
2.4.2 DCVS-DOMINO との比較 37
2.5 テストチップの動作検証 42
2.5.1 セルフクロックによる非同期回路 42
2.5.2 テストチップの回路構成 43

2.5.3 実測による動作検証と性能評価 45
2.6 むすび 47

3 2線2相式論理回路の自動設計システム 49
3.1 まえがき 49
3.2 ASDDL/ASD-CMOS論理合成手法 50
3.2.1 ASDDL/ASD-CMOS自動設計の流れ 51
3.2.2 中間ライブラリを用いた論理合成 53
3.2.3 ネットリスト変換 56
3.2.4 サイクルタイム短縮アーキテクチャの自動適用 . . 57
3.3 自動設計環境の構築と適用事例 58
3.3.1 ASDDL自動設計環境の構築 59
3.3.2 適用事例 61
3.3.3 シングルサイクル回路の性能比較 63
3.3.4 マルチサイクル化による回路性能への影響 65
3.4 むすび 66

4 2つの動作モードを有する高速論理回路方式 69
4.1 まえがき 69
4.2 2モード2線式論理回路ASDMDL 70
4.2.1 高速モードと低消費電力モード 70
4.2.2 ASDMDLの回路構成と動作原理 72
4.2.3 ASDMDLの回路性能評価 74
4.3 大規模回路への適用と回路検証 77
4.3.1 ASDMDLのパイプライン構成 77
4.3.2 スタティックCMOSとの混載設計のための自動合成 81
4.3.3 テストチップによる動作検証と性能評価 83
4.4 むすび 86

5 結論 89

謝辞 93

参考文献 95

発表論文一覧 103
本研究に関する発表論文 103

学術雑誌 103
学術講演 103
受賞 104

第1章

序論

1.1 研究の主旨

大規模半導体集積回路 (Large Scale Integrated Circuit; LSI) は高性能・高機能な製品を開発するために必要不可欠なものになっている．現在では，コンピュータ，オーディオ機器，携帯電話，テレビゲームといった従来からLSIを使用していた分野以外にも，自動車，家電，おもちゃといったあらゆる分野に幅広く使用されており，LSIに強く依存した社会が到来している．

近年，LSIの微細化技術は目覚しい進化を続けている．1970年代には数千トランジスタ規模のLSIが開発の中心であったが，現在では数億のトランジスタが1チップに集積され，微細化の勢いはとどまるところを知らない．このLSIの高集積化により，高度で多種多様な機能を持った回路が実現できるようになってきている．しかしながら，LSIに搭載された回路で処理しなければならないデータ量は日々増加する傾向にある．この増加し続ける膨大なデータを高速に処理するために，デジタルLSIの高速化技術は今後の設計開発において無くてはならないものとなっている．

膨大なデータを高速に処理する大規模デジタルLSI設計のためには，個々のシステムに応じた回路実現向けアーキテクチャの開発が重要である．アーキテクチャによる高速化アプローチの主要なものとして，回路の並列処理やパイプライン処理，加算や乗算を行なう演算器の演算アーキテクチャなどがある．設計するシステムに合ったアーキテクチャを選択し，効率良く適用することでシステムのスループットを高めることができる．このように回路の高スループット化が進むことによって，1クロックサイクル当たりのデータ処理量を減らすことができ，動作周波数を大幅に向上させることが可能となる．しかしながら，動作周波数はデータ処理を行う回路の遅延時間によって決定されるために，アーキテクチャを適用した回路を構成する論理ゲートが最適に設計されていなければ，どんなに良いアーキテクチャを選択，適用しても設計した回路の性能を最

大限に引き出すことはできない．そのため，論理ゲートレベルでの新しい回路設計手法の提案が必要不可欠である．

現在，デジタルLSI設計の多くで用いられている回路設計手法はPMOS (P-channel Metal Oxide Semiconductor) とNMOS (N-channel Metal Oxide Semiconductor) トランジスタで回路を構成するスタティックCMOS (Complementary Metal Oxide Semiconductor) である．スタティックCMOSは高速な動作を実現するために，論理ゲートを構成したトランジスタのチャネル幅Wを大きく設計する．しかしながら，トランジスタのWが大きくなることで，トランジスタの負荷容量が増加してしまうため，トランジスタのWに比例した動作速度を得ることができない．そこで，このスタティックCMOSでは到達不可能な高速動作を達成できる高速回路方式の研究が多く行なわれている[1–8]．しかしながら，これらの回路方式は，高速動作を実現することによる代償として消費電力が大きく増大するという問題がある．

本研究では，従来の高速回路方式を上回る高速動作と低消費電力を実現するASDDL (Asymmetric Slope Differential Dynamic Logic) とASD-CMOS (Asymmetric Slope Differential CMOS) を提案し，ASDDL/ASD-CMOSの特徴を生かした回路を設計するためのアーキテクチャについて説明する．また，大規模回路の短期設計を実現するために，ASDDL/ASD-CMOSの論理合成手法を提案する．さらに，通常はASDDL/ASD-CMOSと同様の高速動作を実現し，処理量の少ない演算を実行するときにはスタティックCMOSと同程度の消費電力を成し遂げるASDMDL (Asymmetric Slope Dual Mode Differential Logic) を提案する．

本章の構成を示す．1.2節では，アーキテクチャレベルでの回路の高速化技術について説明する．1.3節では，回路を設計するに当たって，一般的に用いられているスタティックCMOS回路について説明する．1.4節では，従来研究として提案されている高速動作可能な回路方式について説明し，従来研究における問題点を述べる．1.5節では，本研究の概要と本稿全体の構成について述べる．

1.2 アーキテクチャによる高速化アプローチ

集積回路技術の進歩により，LSIチップ上に実装できる回路規模が増大し，多彩な機能を持ったLSIが実現できるようになってきている．本節では，多機能化によって増加している膨大なデータを高速に処理するた

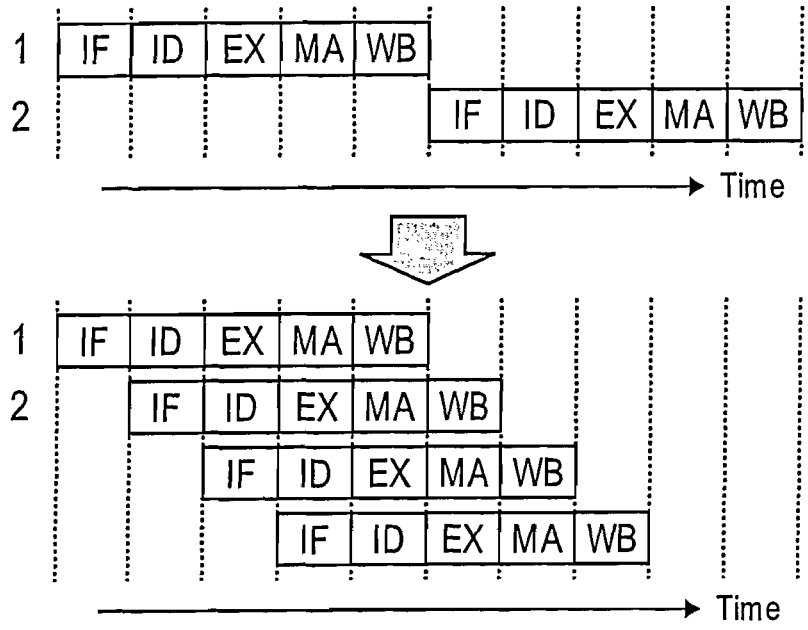

Figure 1.1: Basic five-stage pipeline structure.

めに欠かすことのできないアーキテクチャによる高速化技術について説明する.

アーキテクチャによる高速化アプローチとして用いられている技術には，複数個用意した回路を並列に動作させる並列処理や，演算を複数のステップに分割し，クロック毎に各ステップが独立して演算を行なえるようにするパイプライン処理などがある．並列処理は複数個の回路を用意し，それらの回路を同時に動作させるため，回路規模が用意した回路の個数に比例して大きくなる．一方，パイプライン処理は演算を分割することで並列動作を行なうため，回路規模は増加しない．そのため，パイプライン処理は多くのデジタルLSI設計に用いられている.

プロセッサ等で用いられている5段パイプラインの例をFig. 1.1に示す. 命令を実行するときには命令読み込み(IF)・命令解釈(ID)・実行(EX)・メモリアクセス(MA)・結果書き込み(WB)等を行う必要があり，プロセッサには個々の処理を行なう回路が搭載されている．1つの命令がこれら全てのステップを終えてから次の命令を実行する逐次実行では，実行しているステップ以外の処理回路はまったく動作していない．パイプライン化することで，これらの回路を有効に利用し，クロック毎に命令を次々に

投入・並列実行できるようになる．Fig. 1.1 に示した例では，1つの命令を5つのステップに分割する．各ステップの動作はプロセッサ内の異なる回路で実行されるため，命令ごとに各ステップの処理を時間的に1ステップずらして重ね合わせることができる．この結果，5つの異なるステップの処理を同時に実行できる．すなわち，各命令の完了には5つのクロック周期が必要であるのに対して，プロセッサ全体としてはクロック周期ごとに1個の命令が完了しているように見える．これにより，回路規模を増加させることなく，処理速度を向上させることができる．例えば，2001年に Intel 社によって製造・販売が開始された Pentium 4 は，20 段というきわめて細かい分割を行なったパイプライン構造を用いることで 2GHz 以上の動作周波数を実現している．

プロセッサは，データパスで加算や乗算などの論理演算も行なう．これらの演算を実行する加算器や乗算器などについても数多くの演算アーキテクチャが考えられている [9–15] ．パイプライン化，あるいは並列化されたシステムではそれらを構成する演算器によって動作周波数が決定されるため，高性能な演算アーキテクチャを組み込むことによって回路の限界に近い性能を引き出すことができる．

このようなアーキテクチャを設計するシステムに合わせて効率良く適用することで，回路の動作周波数を引き上げることができる．しかしながら，どんなに優れたアーキテクチャを回路に適用したとしても，それらを構成する論理ゲートが最適に設計されていなければ，設計した回路の性能は低下してしまう．例えば，論理ゲート自体の性能が本来の性能の半分しかないものを用いて回路を構成しても、回路性能は本来の性能の半分となってしまう．そのため，回路の高速化には論理ゲートレベルの回路設計技術は必要不可欠である．次節では，論理ゲートレベルの回路設計技術として，LSI 設計の多くで一般的に用いられているスタティック CMOS 回路について述べる．

1.3 スタティック CMOS 回路

1.3.1 スタティック CMOS の回路構成と動作速度

スタティック CMOS 回路は，PMOS トランジスタと NMOS トランジスタの2つの回路ネットワークで論理ゲートを構成する (Fig. 1.2(a)) ．論理ゲートを構成しているそれぞれのトランジスタネットワークで論理関

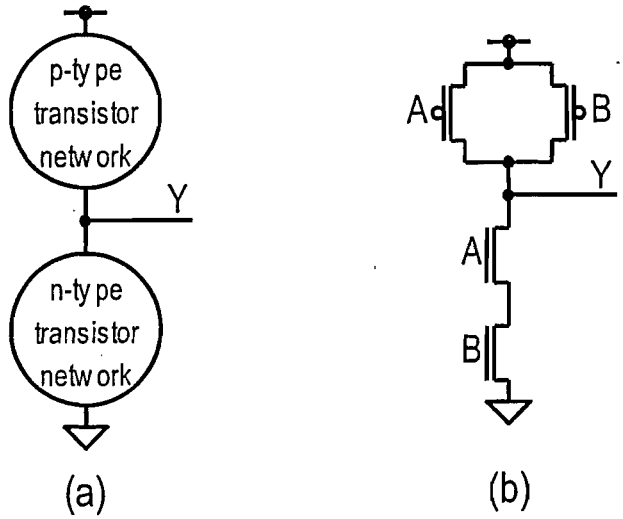

Figure 1.2: Circuit structure of static CMOS logic gate.

数を表現することで，スタティック CMOS は PMOS トランジスタネットワークで論理 1，NMOS トランジスタネットワークで論理 0 を生成する．

Fig. 1.2(b) にスタティック CMOS の 2 入力 NAND ゲートの例を示す．2 入力 NAND の論理式は以下のようにド・モルガンの定理を用いることで得られる．

$$Y = \overline{A \cdot B} \tag{1.1}$$

$$= \overline{A} + \overline{B} \tag{1.2}$$

PMOS トランジスタネットワークと NMOS トランジスタネットワークは，それぞれ式 (1.1)，式 (1.2) から生成され，双方のトランジスタネットワークを用いて 2 入力 NAND の論理関数を表現した回路が構成される．

論理ゲートの動作時には，立ち上がり遷移 (論理 0 から論理 1 への変化) と立ち下がり遷移 (論理 1 から論理 0 への変化) の 2 つの遷移が発生するため，遅延時間は両遷移のうちの遅い方の時間に依存する．スタティック CMOS では，論理 1 が PMOS トランジスタネットワークで生成され，論理 0 が NMOS トランジスタネットワークで生成されるため，立ち上がり遷移時間は PMOS トランジスタ，立ち下がり遷移時間は NMOS トランジスタのサイズによって決定される．これより，両遷移時間が対称となるようにトランジスタサイズを調節する (Fig. 1.3(a))．もしも，Fig. 1.3(b),

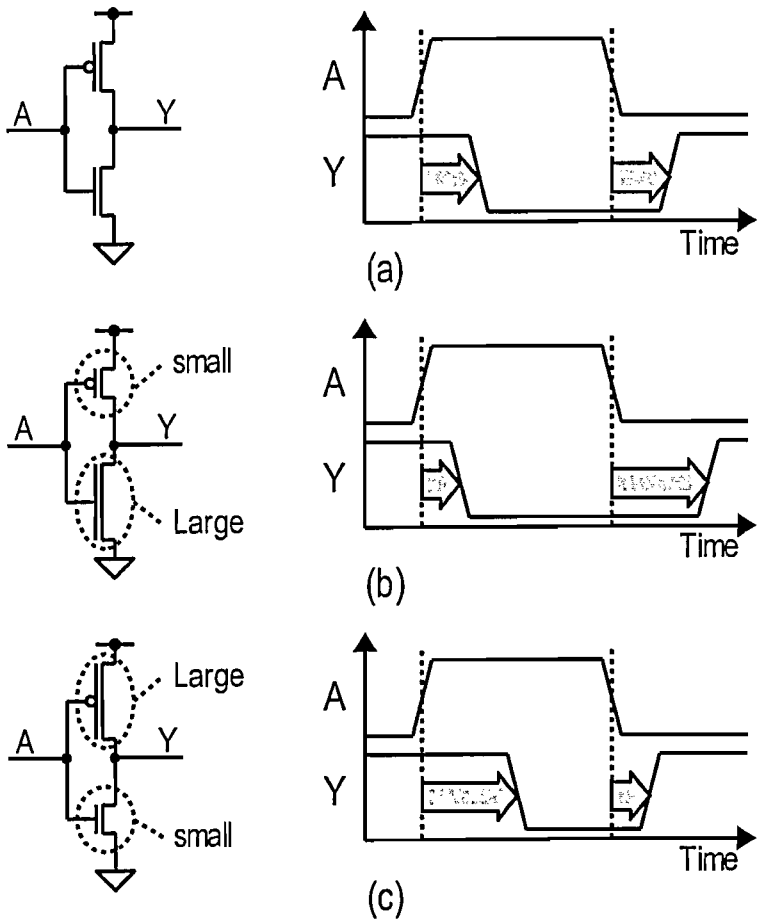

Figure 1.3: Relation between transistor size and delay time.

(c) のようにPMOS トランジスタとNMOS トランジスタのサイズ比を大きく設定し，一方の信号遷移だけが高速になるように設計したとしても，他方の信号遷移が遅くなってしまう．遅い方の遷移時間が論理ゲートの遅延時間となるため，立ち上がり遷移時間と立ち下がり遷移時間が等しくなるように設計した回路と比べて動作速度が遅くなる．そのため，スタティック CMOS は両遷移時間が等しくなるようにPMOS とNMOS トランジスタをサイジングする．

しかしながら，立ち上がり遷移時間と立ち下がり遷移時間が等しくなるようにサイズ比を設定すれば，どれだけでもトランジスタサイズを大

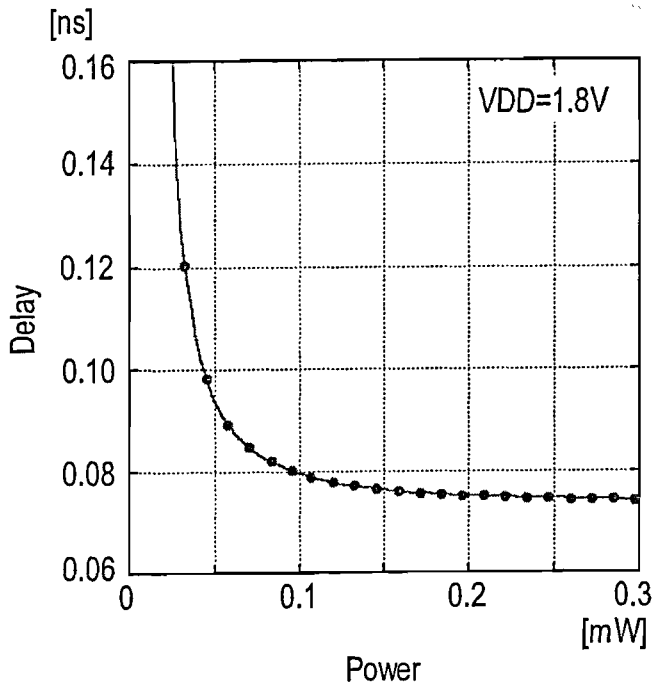

Figure 1.4: Delay time vs. power consumption simulated on 2-input NAND gate in static CMOS.

きくしても良いというわけではない．Fig. 1.4 にスタティック CMOS を用いて 0.18-μm CMOS プロセス，電源電圧 1.8V で設計した 2 入力 NAND ゲートの速度・電力グラフを示す．これは，PMOS と NMOS トランジスタを立ち上がりと立ち下がり遷移時間が等しくなるようなサイズ比と設定した上で，トランジスタの W を変化させて電力を増加させたときの遅延時間の変化を示している．Fig. 1.4 より，消費電力を 0.1mW 以下でトランジスタの W を少しずつ大きくしていくとドレイン電流が増加し，遅延時間が大きく改善されていることがわかる．しかしながら，0.1mW 以上でトランジスタの W を大きくしてもトランジスタの寄生容量が増加するため，トランジスタの W に比例した動作速度を得ることができず，遅延時間の改善は頭打ちとなる．すなわち，スタティック CMOS の動作速度には限界があり，設計した回路を高速化するためには論理ゲート間の駆動力と寄生容量の最適なバランスを取る必要がある．

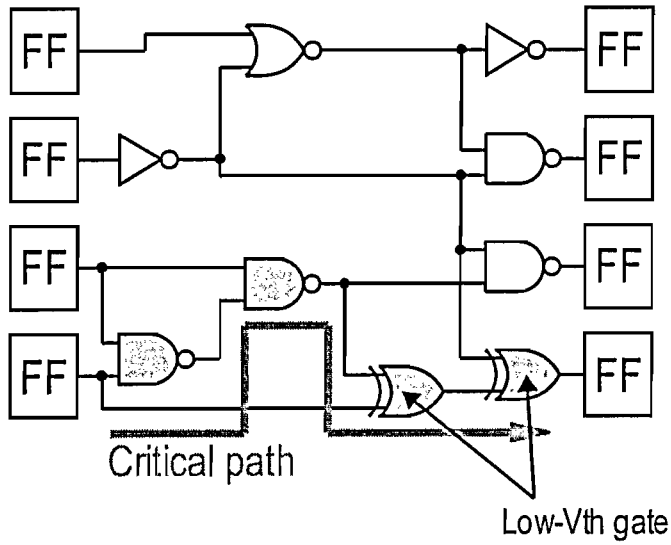

Figure 1.5: Selective Dual-Vth circuit.

1.3.2 低しきい値電圧を用いた回路の高速化

スタティック CMOS は高速な動作を実現するために，論理ゲートを構成したトランジスタのチャネル幅 W を大きく設計するが，トランジスタの負荷容量が増加してしまうため，トランジスタの W に比例した動作速度を得ることができない．

論理ゲートを構成するトランジスタはしきい値電圧を下げることによって，トランジスタのオン電流が増加し，動作速度を向上させることができる．そこで，しきい値電圧を低く設定することで回路全体の高速化を図る手法が考えられている [16–24] ．しかしながら，回路を構成する全ての論理ゲートに対してトランジスタのしきい値電圧を下げた場合には，トランジスタのソースとドレイン間を流れるサブスレッショルド・リーク電流が増え，消費電力が増大する．そこで，Fig. 1.5 に示されているように，設計した回路のクリティカルパスを構成する論理ゲートだけを低しきい値電圧を持った論理ゲート (Low-Vth gate) を用いることで，クリティカルパスの遅延時間を改善し，設計した回路の動作速度を向上させることができる．また，消費電力の増加を最小限に抑えることも可能である．

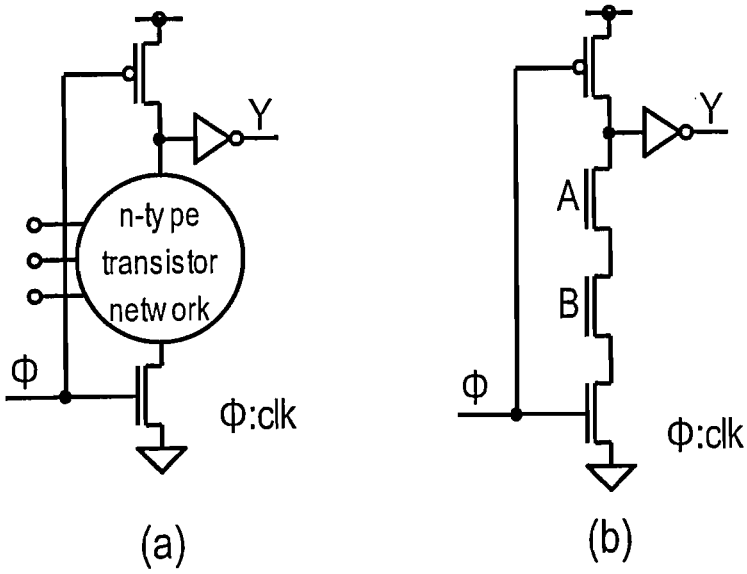

Figure 1.6: Circuit structure of CMOS-DOMINO logic gate.

1.4 従来研究

ここでは，スタティック CMOS よりも高速動作を実現することのできる従来研究について説明するとともに，従来研究における問題点や改善すべき点について述べる．ここで紹介するもの以外にも多くの回路方式が報告されている [5–8, 25–36] が，その中でも代表的な 4 つを取り上げて紹介する．

1.4.1 CMOS-DOMINO 回路方式

スタティック CMOS は PMOS トランジスタネットワークと NMOS トランジスタネットワークで回路を構成し，立ち上がりと立ち下がりの遷移時間が等しくなるようにそれぞれを構成するトランジスタサイズを決定する必要があった．それに対して，NMOS トランジスタネットワークだけで回路を構成することのできる CMOS-DOMINO 回路方式が考えられている [1, 37–43] ．CMOS-DOMINO は，Fig. 1.6(a) に示されているように，NMOS トランジスタネットワークとクロック信号を入力とする 2 つのトランジスタ，出力インバータによって構成される．NMOS トラ

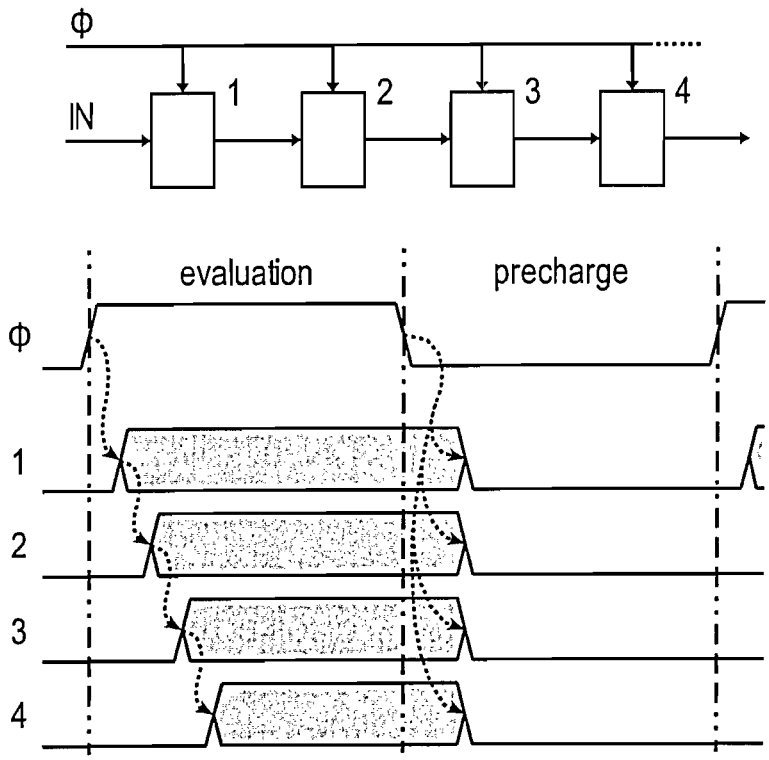

Figure 1.7: Operation diagram of CMOS-DOMINO logic circuit.

ンジスタネットワークはスタティック CMOS と同様に構成されるが，出力インバータによって出力信号は反転されるため，その出力結果はスタティック CMOS とは反転した論理値となる．CMOS-DOMINO の 2 入力 AND ゲートを Fig. 1.6(b) に示す．

CMOS-DOMINO は論理演算の前にプリチャージを行ない，回路をリセットする．このプリチャージは，2 つのトランジスタに入力したクロック信号によって制御される．クロック信号が “1” のときには演算を，“0” のときにはプリチャージを行い，Fig. 1.7 に示すように演算とプリチャージを交互に実行する．Fig. 1.6(b) に示した 2 入力 AND ゲートの動作について説明する．クロック信号が “0” のとき，2 入力 AND ゲートはプリチャージされ，出力インバータの入力に電荷が蓄えられる．クロック信号が “1” になると，入力信号 A と B が “1” のときには演算結果である論理 1 を出力する．どちらかの入力信号が “0” であったときには，出力インバータの入力は浮いた状態となるが，プリチャージによって蓄えられた

電荷によって論理0が維持される．

CMOS-DOMINOは演算結果を生成する回路がNMOSトランジスタネットワークであるため，動作速度を向上させるにはNMOSトランジスタだけを大きくすればよい．スタティックCMOSとは異なり，信号遷移の対称性を無視してトランジスタサイズを設定でき，PMOSトランジスタネットワークが無い分だけ論理ゲートの入力容量は小さくなるため，スタティックCMOS以上の高速動作を実現する．しかしながら，演算の前には必ずプリチャージを必要とし，これらの動作を制御するクロック信号を回路を構成した全ての論理ゲートに分配しなければならないため，消費電力が増大する．また，論理ゲート間にインバータを含めることができない．なぜならば，プリチャージの“0”が“1”となり，誤動作を引き起こすからである．これは，プリチャージ状態から演算状態に変化した瞬間に出力インバータの入力に蓄えられていた電荷が引き抜かれてしまうためであり，出力信号は論理0を出力することができなくなる．これらのことから，CMOS-DOMINOはインバータなしの回路ネットワークを生成する必要がある．

1.4.2 NORA回路方式

CMOS-DOMINOは信号遷移の対称性を無視して，NMOSトランジスタネットワークのみで演算を行なうことで高速動作を実現した．しかしながら，NMOSトランジスタネットワークだけで実現した論理関数はPMOSトランジスタネットワークでも実現することができる．そこで，NMOSトランジスタネットワークだけで演算を行なうN型論理ゲートとPMOSトランジスタネットワークだけで演算を行なうP型論理ゲートの2つを用いて回路を設計するNORA回路方式が報告されている[3]．Fig. 1.8にNORAのN型論理ゲートとP型論理ゲートの回路構成を示す．N型論理ゲートは，NMOSトランジスタネットワークとクロック信号を入力としたトランジスタのみで構成され，CMOS-DOMINOの論理ゲートから出力インバータを取り除いた構成である．一方，P型論理ゲートはPMOSトランジスタネットワークとクロック信号を入力としたトランジスタで構成される．NORAはCMOS-DOMINOと同様に演算とプリチャージを交互に行なうが，プリチャージによって出力される値が異なる．N型論理ゲートはプリチャージによって“1”を出力し，N型論理ゲートは“0”を出力する．

回路を設計する際は，Fig. 1.9に示されているようにこの2つの論理

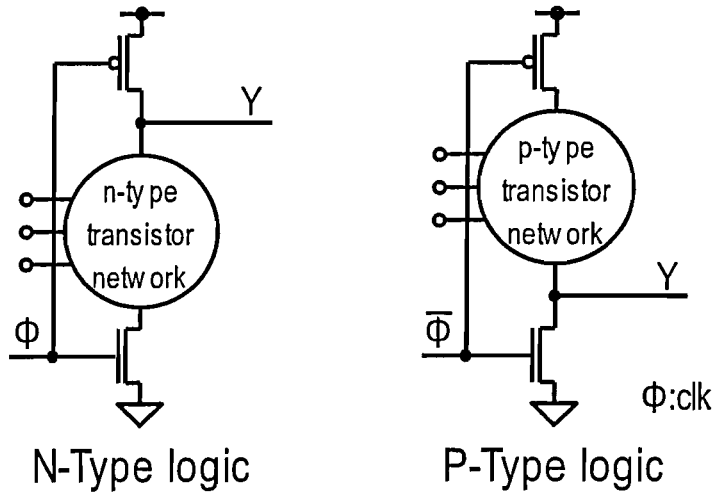

Figure 1.8: Circuit structure of NORA logic gate.

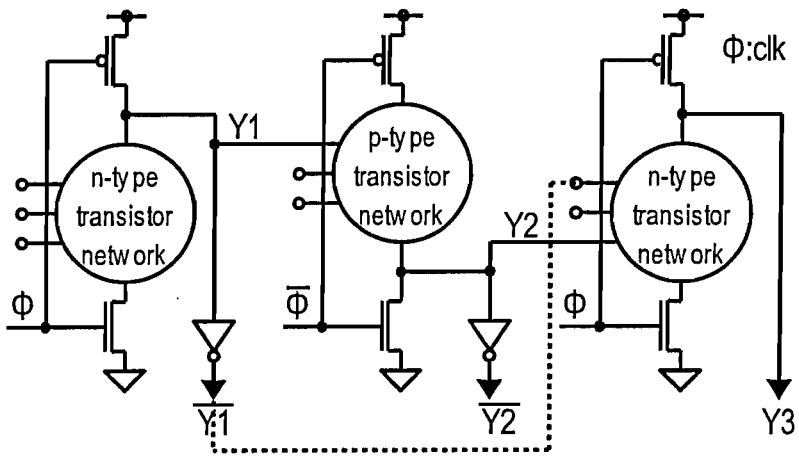

Figure 1.9: Circuit structure of NORA logic.

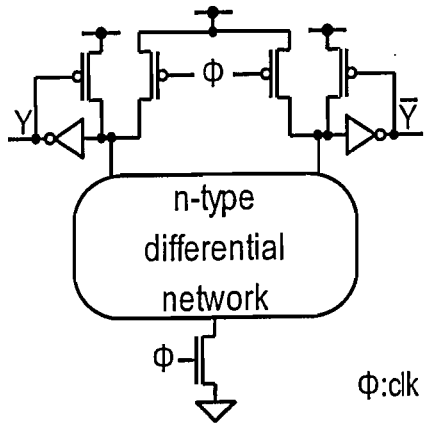

Figure 1.10: Circuit structure of DCVS-DOMINO logic gate.

ゲートを交互に接続して回路を構成する必要がある．そのため，設計する回路に規則性がないとN型とP型を交互に配置することができない．交互に配置できない場合には，インバータを経由して同型の回路と接続する必要がある．そのため，インバータの分だけ遅延時間が増加する．

NORAはCMOS-DOMINOとは異なり，論理ゲートに出力インバータが存在しない分だけ回路を高速化することができる．しかしながら，N型論理ゲートとP型論理ゲートを交互に配置する必要があるため，設計する回路構造の制約が大きく，設計時間が増加する．NORAもCMOS-DOMINOと同様に論理演算とプリチャージを交互に行なうため，消費電力はスタティックCMOSと比べて大きく増加する．

1.4.3 DCVS-DOMINO回路方式

DCVS(Differential Cascode Voltage Switch)-DOMINO [2, 44–49] は入力信号と出力信号を含んだ全ての信号線を正論理と負論理で表現する2線式論理回路であり，CMOS-DOMINOと同様に演算とプリチャージを交互に行なう．

DCVS-DOMINOの回路構成をFig. 1.10に示す．クロック信号が“0”のときは，クロック信号を入力に持つPMOSトランジスタがON状態となり，2線の出力信号は{0,0}を出力し，回路はリセットされる．クロック信号が“1”になると，一方の出力インバータの入力がGNDと接続し，“1”を出力する．他方の出力インバータの入力はどこにもつながってい

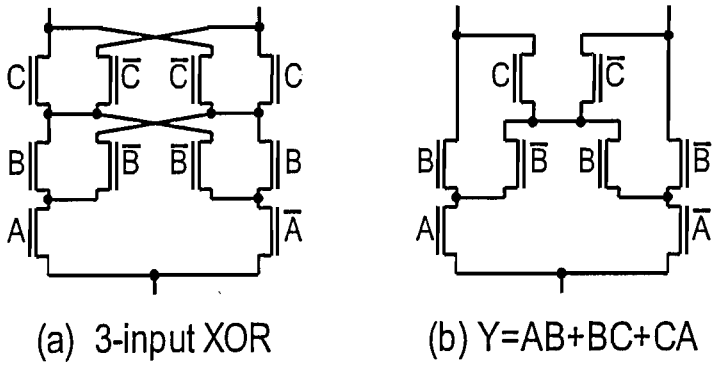

Figure 1.11: NMOS transistor network structure with differential signal.

ない浮いた状態となる．しかしながら，プリチャージによって蓄えられた電荷によって，帰還ループを形成しているPMOSトランジスタがON状態となり，出力インバータの電荷を保持され，反転論理である“0”を出力する．

論理を生成しているNMOSネットワークは，正論理しか持たない回路方式とは異なり，正論理と負論理の入力信号によって設計することができるため，より複雑な論理も1つのネットワークとすることが可能である．例えば，3入力XORや$Y = AB + BC + CA$のような1線信号の回路方式では多段の論理ゲートを必要とする論理関数も，Fig. 1.11に示すような1つのネットワークで構成することができる．これは，設計した回路の論理ゲート段数を大きく減らすことができ，これまでに紹介した2つの回路方式よりも高速動作を実現できる[46]．しかしながら，正論理と負論理の出力信号を生成する必要があるため，トランジスタ数が増加し，CMOS-DOMINOやNORAよりも消費電力が増加する．また，DCVS-DOMINOもCMOS-DOMINOと同様にインバータなしの回路ネットワークを生成する必要がある．

1.4.4 SCSL回路方式

これまで紹介した3つの回路方式はプリチャージを制御するために全ての論理ゲートに遷移確率の高いクロック信号を分配する必要があるため，

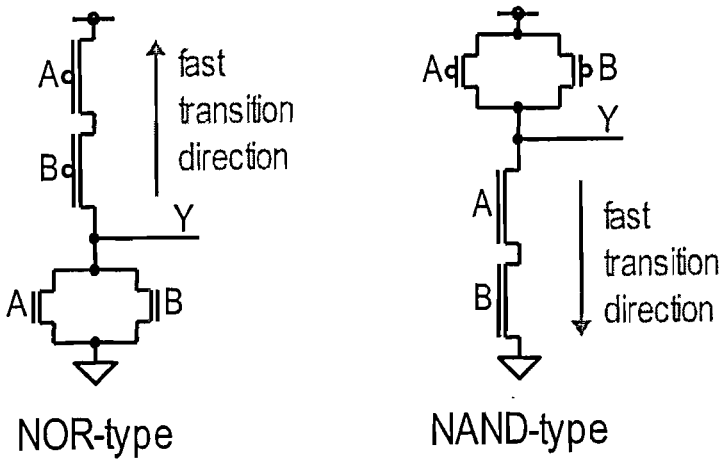

Figure 1.12: Circuit structure of SCSL logic gate.

スタティック CMOS に比べて消費電力が増大する．そこで，プリチャージを制御するための信号線をなくすことで，これまでに紹介してきた回路方式よりも低消費電力を実現し，スタティック CMOS よりも高速動作を成し遂げる SCSL(Selectively Clocked Skewed Logic) が提案されている [4, 50–56] ．この SCSL は，Fig. 1.12 に示すようなスタティック CMOS の NOR ゲートと NAND ゲートだけを用いて回路を設計する．Fig. 1.12 には 2 入力の NOR ゲートと NAND ゲートを示しているが 3，4 入力であっても問題はない．

スタティック CMOS は信号の遷移時間が対称となるようにトランジスタサイズを調節するが，この回路方式で用いる NOR ゲートと NAND ゲートは，それぞれ立ち上がり遷移，立ち下がり遷移の片側の遷移のみを高速化する．この 2 つの論理ゲートは NORA と同様に交互に配置され，論理演算とプリチャージを交互に行なう．プリチャージの制御には信号線を用いず，演算結果が回路の前段から後段に順次伝搬されていくのと同様に，プリチャージも順次伝搬させ，回路をリセットする．

SCSL の NOR ゲートは，プリチャージ信号として前段から “1” が到着し，後段に向けて “0” を出力する．また，NAND ゲートは前段から “0” が到着し，後段に向けて “1” を出力する．そのため，演算時における NOR ゲートと NAND ゲートの信号遷移はそれぞれ立ち上がり遷移，立ち下がり遷移となる．このため，それぞれ信号遷移を高速化することで回路全体の高速化を図ることができる．

SCSLはプリチャージを演算と同様に前段から後段の論理ゲートへ向けて順次伝搬させるため，プリチャージ制御のための信号線をなくすことができ，消費電力を削減することができる．しかしながら，NORAと同様に規則的な配置を必要とするため，設計コストが増加する．また，NORゲートとNANDゲートのみで回路を構成するため，論理ゲートの段数が多くなる．これにより，片側の遷移時間を高速化することで回路全体の高速化を図るという利点を損ねることがある．

1.5 研究の概要と本論文の構成

ここでは，研究の概要とともに，各章の構成について説明する．第2章では，立ち上がりと立ち下がりの信号遷移を意図的に非対称とすることで高速動作を実現したASDDL/ASD-CMOS回路方式を提案する．第3章では，第2章で提案したASDDL/ASD-CMOSの自動設計を実現するための論理合成手法を提案する．第4章では，回路の動作タイミングに合わせて2つの動作を切り替えることで，最適な回路性能を引き出すことのできるASDMDL回路方式を提案する．

1.5.1 非対称な信号遷移を用いた高速論理回路方式

第2章では，論理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することで，スタティックCMOSに比べて格段に短い遅延時間を実現するASDDLとASDCMOSについて説明する．

ASDDL/ASD-CMOSは正論理と負論理の信号で論理値を表現する2線式論理回路であり，演算とプリチャージを交互に行うことで動作する．高速動作を実現するDCVS-DOMINOはプリチャージを制御するために特別な信号線を用いるが，ASDDL/ASD-CMOSはSCSLと同様に論理ゲートの入力信号がその制御信号と同じ役割を担うため，遷移確率の高い制御信号の分配による消費電力を削減することができる．SCSLは立ち上がり遷移を高速化したNORゲートと立ち下がり遷移を高速化したNANDゲートのみで回路を構成する．しかしながら，NORゲートとNANDゲートしか用いることができないため，設計した回路によっては論理ゲートの段数が多くなってしまい，論理ゲートの動作速度に見合った回路性能を引き出すことができない．提案するASDDL/ASD-CMOSはNORゲー

トやNANDゲートだけでなく，複雑な論理関数も1つの論理ゲートで構成できる上に，プリチャージ制御のための信号をなくすことできるため，DCVS-DOMINOよりも高速，かつ低消費電力を実現できる回路方式である．

ASDDL/ASD-CMOSは立ち上がり遷移と立ち下がり遷移を非対称とすることで，演算の遅延時間を高速化できるが，1回の演算を行なってから次の演算を行なうまでのサイクルタイムが長くなるという欠点がある．そこで，演算の裏にプリチャージを隠すことができ，サイクルタイムは立ち上がり遷移となる演算時間とほぼ等しくすることができるサイクルタイム短縮アーキテクチャについて述べる．

ASDDLとASD-CMOSの性能検証のために16ビット符号付き乗算器を設計し，スタティックCMOSとの比較評価を示す．また，トランジスタレベルの回路構成，およびレイアウトにおけるDCVS-DOMINOとの回路比較について述べ，SPICEシミュレーションによる結果を示す．さらに，正常な動作を確認するために試作した実チップの回路構成について説明し，テストチップの実測評価を示す．

1.5.2 2線2相式論理回路の自動設計システム

第3章では，第2章で提案した2線2相式論理回路であるASDDL/ASD-CMOSの論理合成手法について説明する．これにより，制御回路などのランダムロジックを含んだ大規模LSIの自動設計を実現でき，短期設計を可能とする．

正負両論理の2線で全ての信号を表現する2線式論理回路は，多くの開発現場で使用されているツールで，論理ゲートをそのまま論理合成することができない．また，ASDDL/ASD-CMOS，および従来から研究されているDCVS-DOMINOのような演算の前にプリチャージを必要とする回路方式は，論理ゲート間にインバータが存在しない回路ネットワークを生成する必要がある．そこで，スタティックCMOSで回路を合成し，同じ論理関数を持った論理ゲートで置き換えるといった方法がとられている．その際，ド・モルガンの定理を用いてインバータのない回路ネットワークを生成することで自動設計を実現する事例が報告されている[57–62]．しかしながら，論理ゲートの置き換えやインバータの削除によって，論理ゲート間の駆動力と負荷容量のバランスが崩れてしまうため，設計した回路の最適な性能を得ることができない．

そこで，2線2相式論理回路の自動設計環境を構築し，ASDDL/ASD-

CMOS の自動設計を実現する．ASDDL/ASD-CMOS の論理合成には，スタティック CMOS で用いられている市販の論理合成ツールを利用し，ASDDL/ASD-CMOS の論理ゲートを疑似的に定義した中間ライブラリを用いて合成する．このライブラリにより合成した結果に後処理として変換を施すツールを用いて，ASDDL/ASD-CMOS の自動設計を行う．

ASDDL/ASD-CMOS の特徴を最大限に生かした論理合成を実現するために，論理ゲート間の駆動力と負荷容量のバランスを崩さずにインバータのない回路ネットワークを生成するための中間ライブラリを考案した．また，合成後の後処理を施すための変換ツールを独自に開発した．この変換ツールは，サイクルタイム短縮アーキテクチャを自動的に適用する機構も備えている．また，提案した論理合成手法を用いて実際に回路を設計するために構築したライブラリについても説明し，その回路の性能評価を示す．

1.5.3 2つの動作モードを有する高速論理回路方式

第4章では，高速モードと低消費電力モードの2つの動作モードを持った ASDMDL 回路方式について説明する．

ASDDL/ASD-CMOS はプリチャージのための制御信号をなくすことにより DCVS-DOMINO よりも高速・低消費電力を実現するが，スタティック CMOS と比べると消費電力が増加する．これは，設計した回路の全ての論理ゲートで立ち上がりと立ち下がりの2つの信号遷移がクロックサイクル毎に発生するためである．演算を行なう前には必ずプリチャージを行なうため，演算時には正論理あるいは負論理のどちらかの信号線で立ち上がり遷移が発生し，プリチャージ時には立ち下がり遷移が発生する．すなわち，2線の信号線の遷移確率は100%となるので，スタティック CMOS よりも大きい電力を消費する．

そこで，ASDMDL は ASDDL/ASD-CMOS と同様に演算の前にプリチャージを行なう2相で動作する高速モードとスタティック CMOS と同様にプリチャージなしの1相で動作する低消費電力モードという2つの動作モードを1つの回路で実現する．この2つの動作モードを動作中の回路の各タイミングによって切り替えることで，設計した回路の最適な回路性能を引き出すことができる．また，実際に大規模回路に適用するに当たって，ASDDL/ASD-CMOS で提案したサイクルタイム短縮アーキテクチャを利用したパイプライン構成について説明し，ASDMDL とスタティック CMOS が混在した大規模回路の論理合成システムについて述べる．

さらに，ASDMDLの性能評価として2入力NANDと2入力XOR，組み合わせ回路である16ビット符号付き乗算器を設計し，スタティックCMOSとの比較評価を行なった結果を示す．また，試作したSH3アーキテクチャに基づいた合成可能なIPコアの実測結果について述べる．

第2章

非対称な信号遷移を用いた高速論理回路方式

2.1 まえがき

本章では，非対称な信号遷移を用いて高速化を図った2線2相式論理回路ASDDL (Asymmetric Slope Differential Dynamic Logic) と ASD-CMOS (Asymmetric Slope Differential CMOS) を提案する．

近年のSoC開発において，高速低消費電力のデジタルプロセッサの要求はますます増加する傾向にある．高速な動作を実現するために，スタティックCMOSは立ち上がりと立ち下がりの遷移時間が対称となるように設計し，論理ゲートを構成したトランジスタのチャネル幅Wを大きく設計する．しかしながら，トランジスタのWが大きくなることで，トランジスタの負荷容量が増加してしまうため，トランジスタのWに比例した動作速度を得ることができない．

そのため，スタティックCMOSよりも高速に動作する回路方式が考えられている．これらの回路方式では設計した回路の全ての論理ゲートにプリチャージを制御する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．

本章では，従来の高速回路方式よりも高速動作を実現するASDDL /ASD-CMOSについて述べる．ASDDL/ASD-CMOSは演算の前に回路をプリチャージするために，サイクルタイムが長くなる．そこで，このサイクルタイムを大幅に短縮するアーキテクチャについて説明する．また，ASDDL/ASD-CMOSの性能評価としてスタティックCMOSとの比較結果，およびデジタルLSIの構成要素として用いられる回路方式の中で最も高速動作が可能とされる回路方式であるDCVS-DOMINOとの回路構成上の比較とシミュレーションによる性能比較を示す．さらに，正常な動作を確認するために試作したテストチップの回路構成と実測結果についても示す．

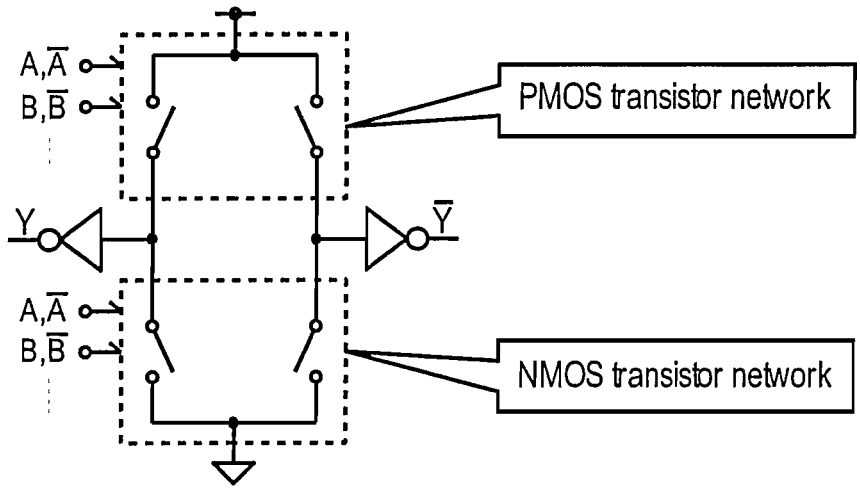

Figure 2.1: ASDDL/ASD-CMOS logic circuit.

2.2 高速論理回路方式 ASDDL/ASD-CMOS

ASDDL/ASD-CMOS は全ての論理を立ち上がり遷移で表現し，論理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 2 線式論理回路であり，従来の高速回路方式が持っていたプリチャージを制御するための専用の信号を削減することで従来のダイナミック回路に比べて小面積，低消費電力を実現する．本節では，この高速論理回路方式 ASDDL/ASD-CMOS の特徴と回路の構成方法について述べる．

2.2.1 ASDDL/ASD-CMOS の特徴

ASDDL/ASD-CMOS は，NMOS トランジスタと PMOS トランジスタから成る論理生成ネットワークと出力インバータによって構成される．その概略図を Fig. 2.1 に示す．ASDDL/ASD-CMOS は正論理と負論理の 2 本の信号で論理値を表現する 2 線式論理回路である．2 線式論理回路では，回路を構成した全ての信号線で正論理と負論理が存在するため，インバータを用いて負論理を生成する必要がなく，複雑な論理式もスタティック CMOS に比べて少ない論理ゲート段数で回路を構成することが可能となり，高速化に適している．

この正負両論理の 2 線の信号により表現される値は，有効値 (effective

Table 2.1: Operation state with two signal lines.

$\{x,\bar{x}\}$	logic	
{0,1}	0	effective value
{1,0}	1	effective value
{0,0}	-	reset value
{1,1}	-	no definition

value) としての論理0 ({0,1}) と論理1 ({1,0}), および休止値 (reset value) である {0,0} の3種類であり, {1,1} は使用されない (Fig. 2.1). ASDDL/ASD-CMOSは初期状態として休止値を与え, 休止値→有効値→休止値と交互に入力に与えることで, 1動作サイクルを形成する. 設計した回路の入力に与えられた有効値は前段から後段に向けて順次伝搬され, 回路の出力信号から出力された有効値が最終的な演算結果となる.

有効値を伝搬させる前には必ず休止値を伝搬させて回路をプリチャージさせることから, 回路の遅延時間は休止値から有効値への遷移である立ち上がりの伝搬時間となる ({0,0} → {0,1} あるいは {0,0} → {1,0}). これより, 信号の立ち上がり遷移と立ち下がり遷移を非対称とすることができ, 休止値から有効値への遷移である立ち上がりの伝搬時間をより高速となるように設計する事で, 回路全体の高速化を図る (Fig. 2.2). 有効値から休止値への遷移である立ち下がり伝搬時間については, 演算の遅延時間には影響がないため, 考慮する必要はない. この非対称な信号の遷移時間により, スタティック CMOS に比べて格段に短い遅延時間を実現することがを可能である.

この信号遷移の非対称性は回路を構成する論理ゲートのトランジスタサイジングにより実現する. Fig. 2.1 に示すNMOSネットワークのトランジスタはチャネル幅Wを大きく設計する. 逆に, 高速化の妨げとなるトランジスタ容量をできる限り小さくするために, PMOS トランジスタのWは小さくする. 出力インバータはNMOS トランジスタを小さく, PMOS トランジスタを大きくサイジングする. これにより, 基本論理ゲートの出力信号は立ち上がり遷移が高速, 立ち下がり遷移は遅いという非対称なものとなる.

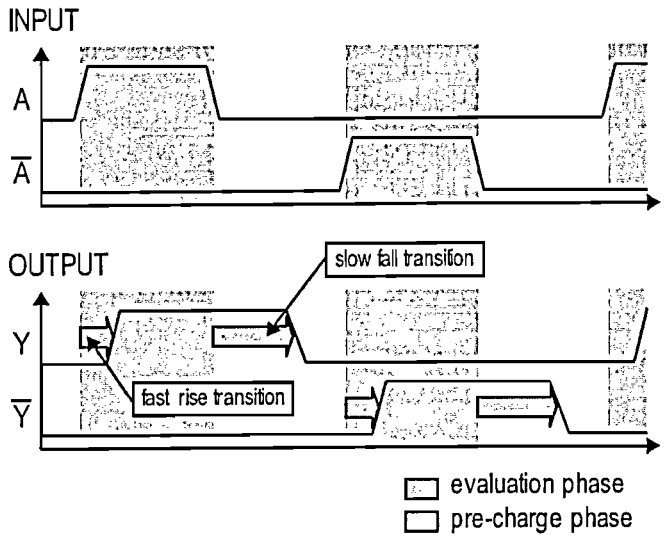

Figure 2.2: Operation diagram of ASDDL/ASD-CMOS logic circuit.

2.2.2 ASDDL と ASD-CMOS の回路構成と動作

回路の構成方法として，ダイナミック動作する ASDDL とスタティック動作する ASD-CMOS の 2 種類の回路構成を提案する．

Fig. 2.3(a) と (b) に ASDDL と ASD-CMOS の 2 入力 NAND の例をそれぞれ示す．NMOS ネットワークの構造は ASDDL と ASD-CMOS に共通であり，BDD 表現 (Binary Decision Diagram) [63, 64] を用いることで容易に設計することが可能である．Fig. 2.3(a) と (b) に示されている 2 入力 NAND の NMOS ネットワークを BDD 表現を用いて設計する方法を Fig. 2.4 に示す．まず，2 入力 NAND の論理関数より Fig. 2.4(a) のようなツリー構造を生成し，ノード数が最も少なくなるようにツリーを簡略化する (Fig. 2.4(b))．そして，簡略化されたツリー構造を上下反転させ，それぞれのノードを 2 つの NMOS トランジスタのセレクタ構造で置き換えることで NMOS ネットワークを生成することができる (Fig. 2.4(c))．このように BDD 表現を用いて NMOS ネットワークを生成することにより，スタティック CMOS では論理ゲートが 2，3 段となるような複雑な論理関数も 1 つの論理ゲートで設計することが可能であり，Fig. 2.3(c) と (d) に示すような 3 入力 XOR も 1 つの論理ゲートで実現できる．

ASD-CMOS の PMOS ネットワークは，NMOS ネットワークと双対の

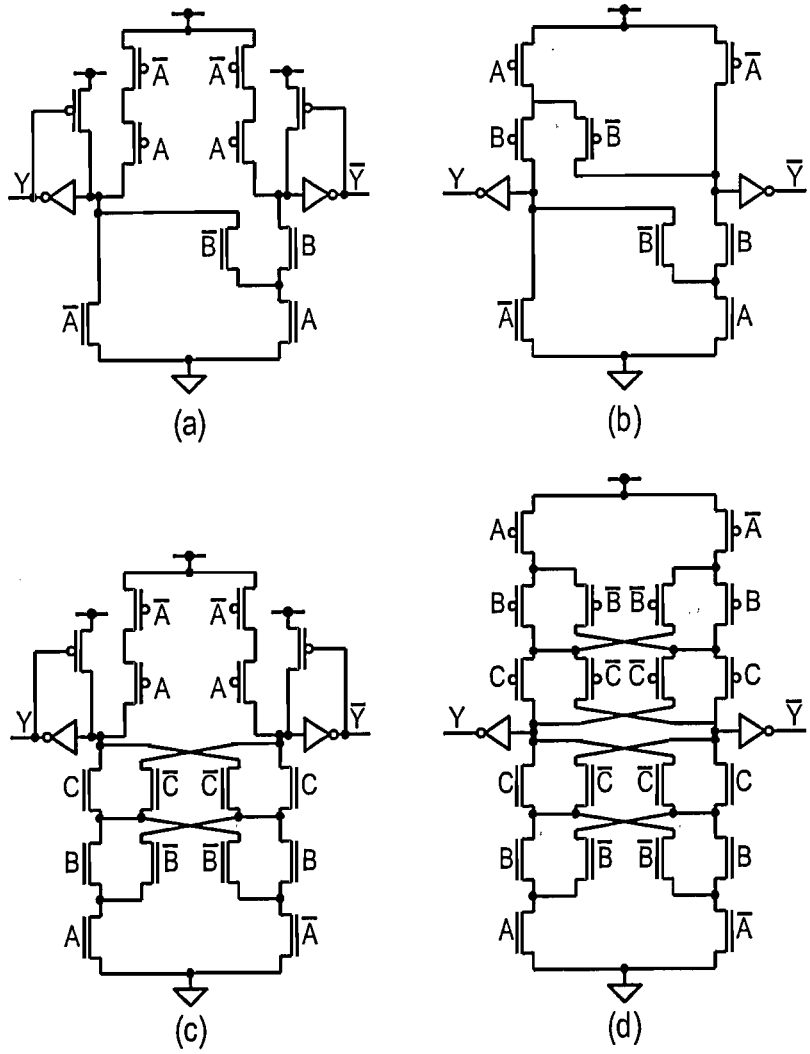

Figure 2.3: Cell Schematic of (a) ASDDL NAND cell, (b) ASD-CMOS NAND cell (c) ASDDL XOR cell, and (d) ASD-CMOS XOR cell.

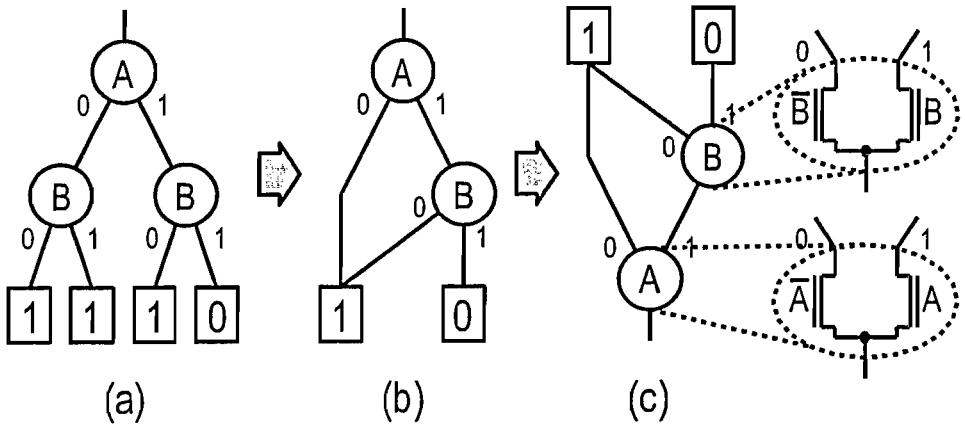

Figure 2.4: Binary Decision Diagram of 2-inputs NAND function.

構造を持ち，NMOS ネットワークのトランジスタを PMOS トランジスタに置き換えた構成となる．一方で，ASDDL の 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，論理ゲートの入力信号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，論理ゲートは休止値を出力する．さらに ASDDL はダイナミック動作時の出力信号を安定させるために，出力信号からの帰還プルアップ PMOS(pull-up PMOS) を配置する．

Table 2.2 と 2.3 に ASD-CMOS と ASDDL の動作の概要をそれぞれ示す．ASD-CMOS の論理ゲートは，全ての入力信号に休止値 {0,0} が入力されたとき，PMOS ネットワークは ON 状態，NMOS ネットワークは OFF 状態となり，出力インバータの入力は VDD と接続される．これにより，論理ゲートの出力信号は休止値を出力し，回路はプリチャージされる．ASD-CMOS の論理ゲートのすべての入力信号が有効値になると，一方の出力インバータの入力は PMOS ネットワークにより VDD と接続され，他方の出力インバータの入力は NMOS ネットワークにより GND と接続される．これにより，ASD-CMOS の論理ゲートは演算結果である有効値を出力する．ASDDL の論理ゲートでは，全ての入力信号に休止値 {0,0} が入力されると ASD-CMOS と同様の動作を行ない，出力信号から休止値が出力される．一方，すべての入力が有効値になると，PMOS ネットワークにより一方の出力インバータの入力が電源と接続される．しかしながら，他方の出力インバータの入力はどことも接続されずに浮いた状態と

Table 2.2: The ASD-CMOS operation of NMOS and PMOS networks.

Input	NMOS Network	PMOS Network
Every input has a reset value	OFF	ON
Every input has an effective value	one-side path is ON and opposite-side path is OFF	one-side path is ON and opposite-side path is OFF

Table 2.3: The ASDDL operation of NMOS and PMOS networks.

Input	NMOS Network	PMOS Network
Every input has a reset value	OFF	ON
Every input has an effective value	one-side path is ON and opposite-side path is OFF	OFF

なる．しかし，有効値が伝搬される前に出力していた休止値によって蓄えられた電荷が帰還プルアップPMOSをON状態にするので，出力インバータの入力はVDDを維持する．これにより，ASDDLは有効値を出力し，演算が行われる．

ここで，スタティックCMOSでは，論理ゲートのどれか1つの入力信号が変化しただけで出力信号も変化するために，グリッチが発生しやすい．一方，ASDDL/ASD-CMOSの論理ゲートはすべての入力信号に有効値が到着して始めて演算を行なう．そのため，グリッチの伝搬による誤動作は生じることはない．また，論理ゲートの配置によっては2線の信号の配線長が異なってしまう場合がある．しかしながら，ASDDL/ASD-CMOS論理ゲートにおける正負両論理の2線の信号線は完全差動で変化するわけではなく，プリチャージによってリセットされた{0,0}の状態からどち

らか一方の信号が変化するだけである．これより，2線信号の配線長が等しくなるように論理ゲートを配置する必要はない．

ASDDLのPMOSネットワークのトランジスタ数は，どんな論理関数においても直列接続された4つのPMOSトランジスタと2つのプルアップPMOSで構成でき，設計した回路全体のトランジスタ数を少なくすることができる．そのため，ASD-CMOSに比べて消費電力を小さくすることが可能である．一方，ダイナミック動作するASDDLに対してASD-CMOSはスタティック動作するために動作が非常に安定している．すなわち，ASDDLを設計する際には，チャージシェアリングの影響を考慮してトランジスタサイジングを設定する必要がある．

2.3　サイクルタイム短縮アーキテクチャ

ASDDL/ASD-CMOSは立ち上がり遷移である有効値の伝搬時間と立ち下がり遷移である休止値の伝搬時間を意図的に非対称とすることで，1回の演算時間は高速になる．しかしながら，1回演算を行なってから次の演算を行なうまでのサイクルタイムは長くなるという欠点がある．これは，ASDDL/ASD-CMOS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ためである．そのため，ASDDL/ASD-CMOSの論理ゲートのみで回路を構成しても，スタティックCMOSよりもサイクルタイムが遅くなってしまう．

そこで，ASDDL/ASD-CMOS特有のアーキテクチャを用いてこの問題を解決する．以下ではこのアーキテクチャをサイクルタイム短縮アーキテクチャと呼び，本節ではこのサイクルタイム短縮アーキテクチャの構成方法について述べる．

2.3.1　アーキテクチャの構成

サイクルタイム短縮アーキテクチャは，プリチャージである休止値の伝搬を演算となる有効値の伝搬の裏に隠し，あたかも有効値のみが伝搬しているように見せることのできるアーキテクチャである．以下に，この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際の設計方法について説明する．

設計した回路にサイクルタイム短縮アーキテクチャを適用するには，まず回路全体を入力側と出力側の2つの回路ブロックに分割する．分割し

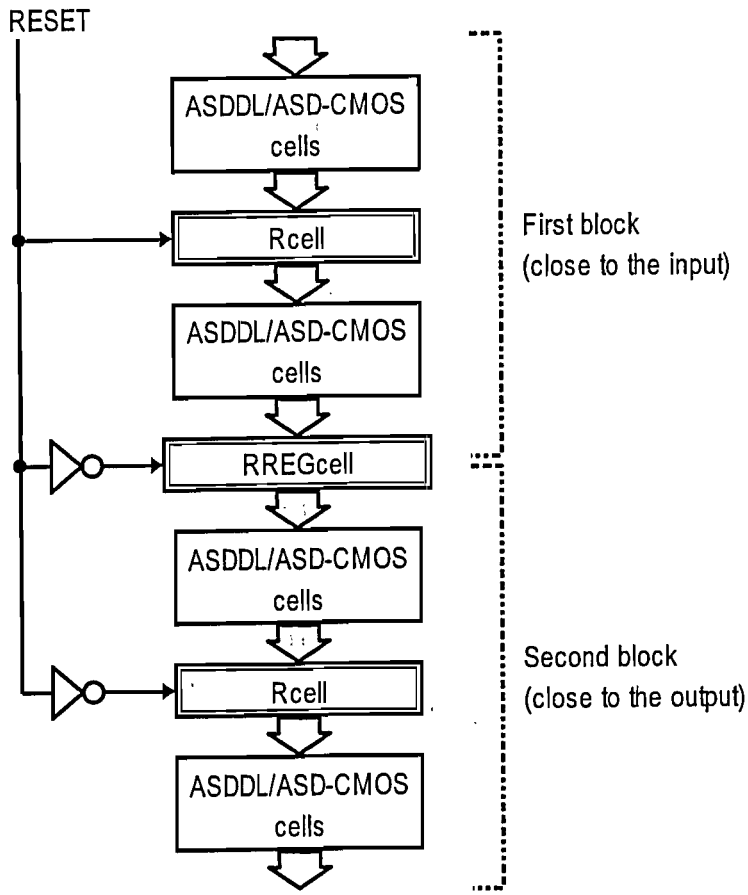

Figure 2.5: ASDDL/ASD-CMOS architecture with cycle time reduction.

たそれぞれの回路ブロックではASDDL/ASD-CMOSの2つの動作である有効値伝搬による演算と休止値伝搬による回路のプリチャージを交互に行なう．この2つの動作の切り替えを“RESET”信号で制御する(Fig. 2.5).

それぞれの回路ブロックの間には，入力に到着した有効値を保持する機能とプリチャージ状態のときに休止値を新たに出力する機能を持った“RREGcell”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，プリチャージ状態となったときに新たに休止値を伝搬する機能を持った“Rcell”を分割したそれぞれの回路ブロック内に配置する．Rcellを配置せずに回路を2つに分割するだけでは，立ち下がり遷移である休止値の伝搬時間は立ち上がり遷移である有効値の伝搬時間に比べて遅いため，伝搬時間の遅い立ち下がり遷移によってサイクルタイムが決定付けられてしまう．そのため，有効値の伝搬時間(立ち上がり伝搬)＞休止値の伝搬時間(立ち下がり伝搬)となるようにRcellを分割した回路に配置し，設計した回路のプリチャージに要する時間が，演算に要する時間よりも短くなるように調節する．

Fig. 2.6に，このサイクルタイム短縮アーキテクチャの動作波形の例を示す．動作開始時には，分割した前段回路ブロック(first block)のASDDL/ASD-CMOS論理ゲートには休止値が伝搬され，後段回路ブロック(second block)は不定値となっている．RESET信号が“0”になる(state 1)と，前段回路ブロックに有効値が伝搬され，演算が行なわれる．一方，後段回路ブロックではRREGcellとRcellのそれぞれから休止値が出力され，全ての論理ゲートがプリチャージされる．その後，RESET信号が“1”になる(state 2)と，前段回路ブロックの全ての論理ゲートがプリチャージされ，後段回路ブロックでは演算が行なわれる．このstate 1とstate 2の動作を繰り返すことで連続した演算を実現できる．

このように，半サイクル毎にそれぞれ分割した回路ブロックで有効値と休止値を交互に伝搬することにより，全ての論理ゲートのプリチャージを演算の裏に隠すことができ，あたかも高速な立ち上がり遷移のみで演算を行なっているように見える．すなわち，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，有効値の伝搬時間と休止値の伝搬時間の総和であったサイクルタイムが有効値の伝搬時間とほぼ等しくなり，大幅にサイクルタイムを短縮する事が可能となる．

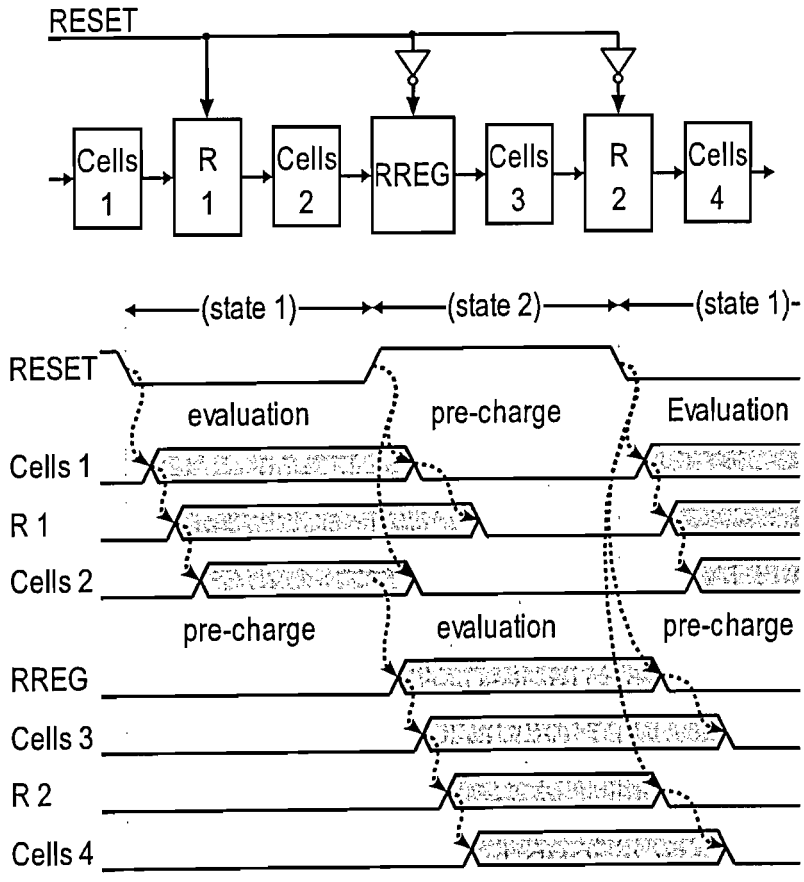

Figure 2.6: Operation diagram of ASDDL/ASD-CMOS logic circuit with cycle time reduction.

2.3.2　RcellとRREGcellの回路構成

ここでは，サイクルタイム短縮アーキテクチャを回路に適用した際に配置されるRcellとRREGcellの回路構成について説明する．

前節でも説明したように，Rcellは新たに休止値を出力する機能を持ち，RREGcellは有効値を保持する機能と新たに休止値を出力する機能を持つ．RcellとRREGcellは設計した回路にそれぞれの機能を付加するため，どうしても設計した回路の性能を低下させる．そこで，この性能低下を最小限にするために，それぞれの機能だけを持ったものを設計した回路に配置するのではなく，通常のASDDL/ASD-CMOS論理ゲートにそれぞれの機能を付加させる．これにより，サイクルタイム短縮アーキテクチャの適用による性能低下を最小限に抑えることができ，サイクルタイム短縮アーキテクチャを適用する前の回路とほぼ等しい性能が実現できる．

RcellとRREGcellの回路構成をFig. 2.7(a)と(b)にそれぞれ示す．RcellはASDDL/ASD-CMOS論理ゲートの出力インバータの部分を2入力のNORゲートに置換した構成になっており，置換したNORゲートの一方の入力に動作の切り替えを行なうRESET信号を接続する．RESET信号が“0”のときには，通常のASDDL/ASD-CMOS論理ゲートと同様にNMOSネットワークの論理関数に従った演算結果を出力する．一方で，RESET信号が“1”のときは，出力インバータ部分を置換したNORゲートのPMOSトランジスタがOFFとなり，出力信号とVDDとのパスが切断され，Rcellに有効値が入力されていたとしても後段への有効値の伝搬を遮断する．そのとき，RESET信号が入力されたNMOSトランジスタがONとなり，正負両論理の出力がGNDに接続され，休止値である{0,0}が出力される．

RREGcellはRcellのNORゲートの入力とPMOSネットワークの間にPMOSトランジスタ，GNDとの間にNMOSトランジスタを加えた構成となっており，この新たに加わった2つのトランジスタで出力からの帰還ループを形成し，値を保持する機能を果たす．このRREGcellはRESET信号が“1”のとき，Rcellと同様に休止値が出力される．このとき，RESET信号が“0”に変化する前に前段の論理ゲートから到着した有効値を受け取り，NMOSネットワークの論理関数に従った演算結果が生成される．しかしながら，その演算結果は置換されたNORゲートによってせき止められるため，2線の出力信号にはその演算結果が出力されることはなく，休止値が出力され続ける．その後，RESET信号が“0”に変化すると同時にその演算結果が出力され，後段の論理ゲートに伝搬される．RREGcellが

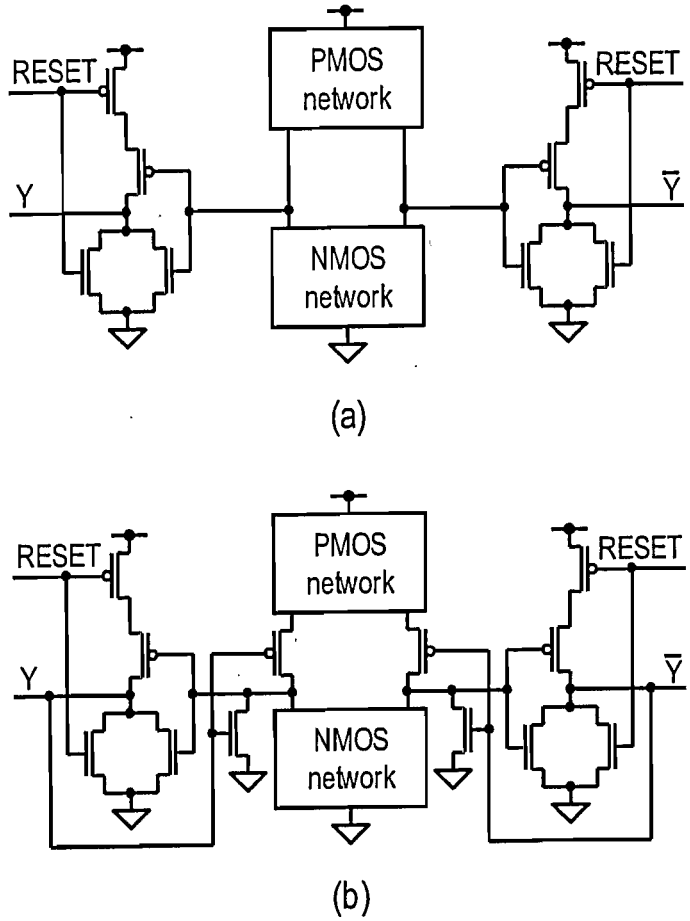

Figure 2.7: Circuit structures of (a) RREGcell, and (b) Rcell.

演算結果を出力すると，前段からは新しい値が到着する．もしも，到着した値が有効値であったならば，RREGcell が誤動作を引き起こす恐れがある．しかしながら，ASDDL/ASD-CMOS は有効値と休止値を交互に伝搬させることから，前段から到着する値は必ず休止値となる．この到着した休止値は帰還ループにより後段への伝搬がせき止められ，RREGcell の出力結果は保持される．

このように，Rcell と RREGcell は ASDDL/ASD-CMOS 論理ゲートに数個のトランジスタを加えるだけでそれぞれの機能を付加することができため，面積の増加は最小限に抑えることができる．また，演算とプリチャージを制御するための RESET 信号は Rcell と RREGcell だけに接

続され，その他の通常の論理ゲートには分配する必要がない．これにより，遷移確率の高いRESET信号を分配する論理ゲート数は少なくて済むため，消費電力の増加を低減できる．

2.4　ASDDL/ASD-CMOSの回路性能評価

本章では，非対称な信号遷移を用いた高速論理回路方式であるASDDLとASD-CMOSの性能評価について述べる．比較対象は，従来から幅広く用いられているスタティックCMOS，および現在報告されている回路方式の中で最も高速動作が可能とされる回路方式であるDCVS-DOMINOとした．

2.4.1　シミュレーションによる性能評価

非対称な信号遷移を用いた高速論理回路方式であるASDDLとASD-CMOSの性能評価のために，0.18-μm CMOSプロセスにて16ビット符号付き乗算器を設計し，従来から用いられているスタティックCMOSとの比較評価を行なった．

ASDDL，ASD-CMOS乗算器のアーキテクチャはASDDL，ASD-CMOS共に同じアーキテクチャであり，部分積生成，部分積加算そして桁上げ吸収回路で構成した．部分積生成部はバッファとAND/NANDゲートを用い，部分積加算回路にはBooth法を用いずにWallace treeのみで構成した．桁上げ吸収加算回路はBLCA (Binary Look-ahead Carry Adder) 回路を用いた．さらに，ASDDLとASD-CMOS乗算器はFig. 2.8に示すように回路を前後半のブロックに分割するためのRREGcell，各回路ブロックの中央部分にRcellを配置した．一方，スタティックCMOS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ライブラリを用い，遅延時間最小の条件で設計した回路を示している．その回路は基本ゲート間の付加容量と駆動力のバランスが最適になるように設計されており，スタティックCMOSで設計する上で最も高速動作可能な回路であると言える．

これらを自動配置配線によりレイアウトした後，RC抽出したものをSPICEシミュレーションにより比較評価した．その比較結果をTable 2.4に示す．Table 2.4は電源電圧は1.8Vの結果であり，消費電力は100MHz動作時の平均消費電力である．ここで，一般的なスタンダードライブラ

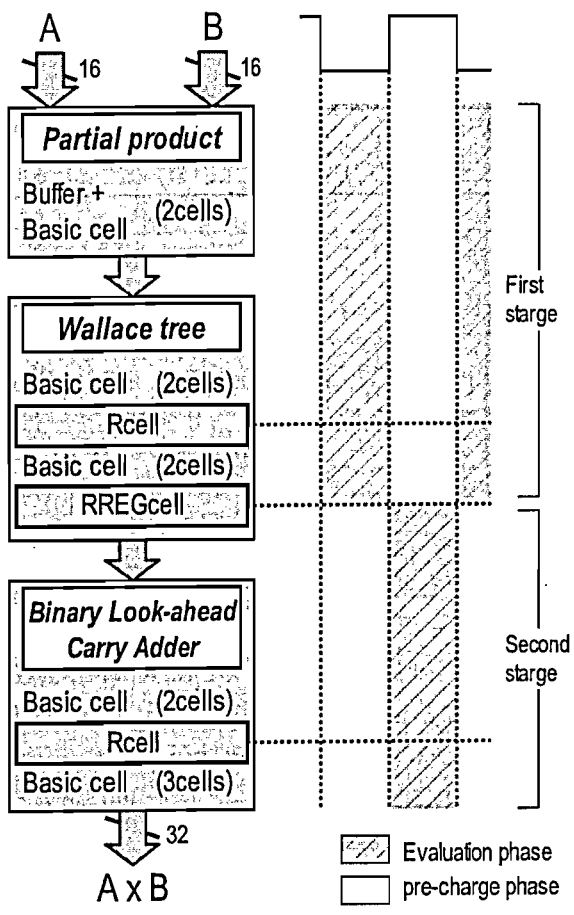

Figure 2.8: The architecture of ASDDL and ASD-CMOS 16-bits multiplier.

Table 2.4: Comparison of 16-bit multipliers in 0.18-μm technology at 1.8V.

	Delay [ns]	Power [mW]	Area [mm^2]
ASDDL	1.82	15.30	0.059
	(0.68)	(2.55)	(1.44)
ASD-CMOS	1.78	18.82	0.062
	(0.66)	(3.13)	(1.51)
static CMOS	2.68	5.99	0.041
	(1.00)	(1.00)	(1.00)

リを用いて16ビット乗算器を設計し，レイアウトから抽出したRCを含んだSPICEシミュレーションした結果では，遅延時間3.33ns，消費電力7.39mW，面積は0.058mm^{2}であった．それらの性能はTable 2.4に示したスタティックCMOSに比べて劣っており，比較対象として設計したスタティックCMOSの16ビット乗算器に用いたライブラリは最適な性能を発揮していることは明らかである．ASDDLとASD-CMOSは，Table 2.4にも示されているようにその最も高速動作を実現できるスタティックCMOSの乗算器よりも高速動作を実現している．ASDDLとASD-CMOSで設計した16ビット符号付き乗算器の遅延時間はそれぞれ1.82ns，1.78nsであり，これらはスタティックCMOSと比べて32%，34%減少した．

また，電源電圧を変化させたときの遅延時間の変化をFig. 2.9に示す．Fig. 2.9からも分かるように，電源電圧を低下させてもその遅延時間の改善率はほとんど変化ないことから，低電源電圧下においてもASDDLおよびASD-CMOSは高速動作を実現できることを確認した．

ASDDLとASD-CMOSは2線式論理回路であるため，論理ゲートのトランジスタ数はスタティックCMOSよりも多い．しかしながら，乗算器の設計にあたり，BDD表現による複雑な論理関数を多く適用できたために，ASDDLとASD-CMOSの面積はスタティックCMOSと比べてそれぞれ44%，51%の増加に抑えることができた．ここで，ASDDLとASD-CMOSは高速動作を実現することによる代償として，スタティックCMOSと比べて消費電力が増大する．これは，設計した回路を構成する全ての論理ゲートで立ち上がりと立ち下がりの両方の遷移が発生するためであり，最も重大な欠点であると言える．しかしながら，2.4.2節で詳しく述べるように，ASDDLとASD-CMOSの消費電力は従来の高速ダイナミック回路

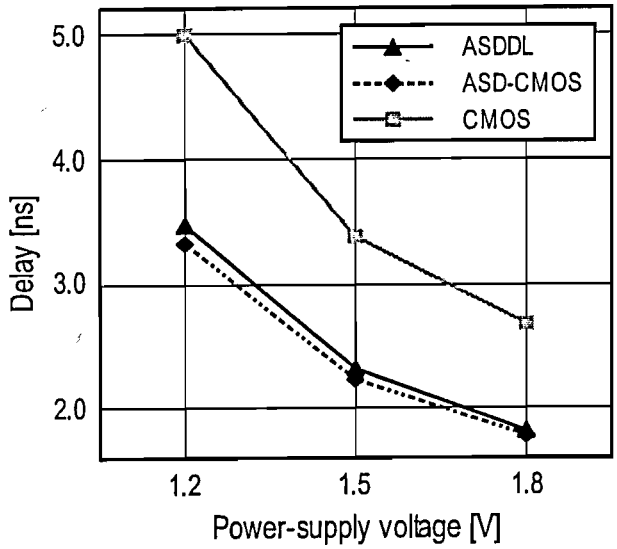

Figure 2.9: Change of delay time for decrease of power-supply voltage.

と比べると小さく抑えることができる.

2.4.2 DCVS-DOMINO との比較

DCVS-DOMINO はディジタル LSI の構成要素として用いられる論理回路の中で，最も高速動作が可能とされる回路方式の 1 つであり，プリチャージを制御するためにクロック信号を用いる.

ASDDL，ASD-DMOS，および DCVS-DOMINO の 2 入力 XOR の論理ゲート比較を Fig. 2.10 に示す. ASDDL と ASD-CMOS の基本論理ゲート自体にはプリチャージを制御するためのクロック信号のような特別な信号は必要ない. その代わりとして，論理ゲートの入力信号がその役割を担う. 一方，DCVS-DOMINO はプリチャージを制御するためのクロック信号を入力とした NMOS トランジスタが NMOS ネットワークと GND の間に存在する. それぞれ 3 つの回路方式において，演算結果を生成するのは NMOS ネットワークであるため，この NMOS ネットワークを含んだ出力インバータから GND までの NMOS トランジスタの段数が遅延時間に大きく左右する. すなわち，ASDDL/ASD-CMOS は NMOS トランジスタの直列段数を 1 段減らすことができるので，回路の立ち上がり遷移時間すなわち遅延時間をより高速にすることが可能である.

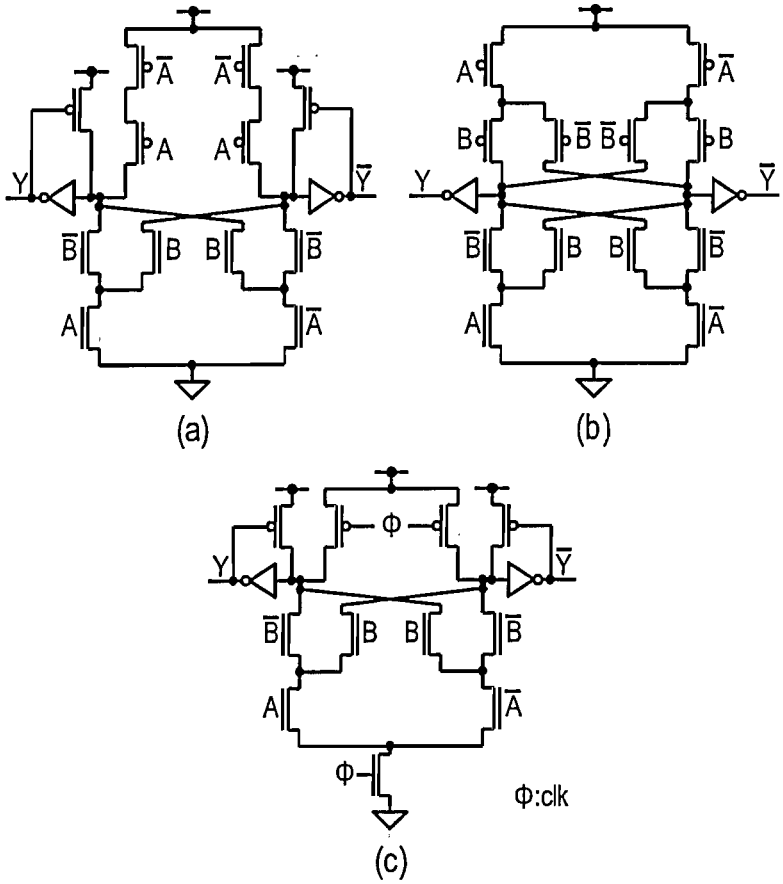

Figure 2.10: 2-input EXOR schematic of (a) ASDDL, (b) ASD-CMOS, and (c) DCVS-DOMINO.

また，DCVS-DOMINOでは，プリチャージ制御のための信号線は回路を構成した全ての論理ゲートに分配する必要があり，ASDDL/ASD-CMOSはスイッチング確率の高いそれらの配線の引き回しを大幅に削減することができる．その上，DCVS-DOMINOではそれらの制御信号の分配に大量のクロックバッファを挿入する必要があるのに対して，ASDDL/ASD-CMOSではRREGcellとRcellだけにRESET信号を分配するだけでよく，信号を分配すべき回路は全体の50%以下に抑えることができる．これより，ASDDL/ASD-CMOSはDCVS-DOMINOよりも回路面積を小さくでき，さらに低消費電力となる．

論理ゲートのレイアウトを自動配置配線で主に用いられるような長方

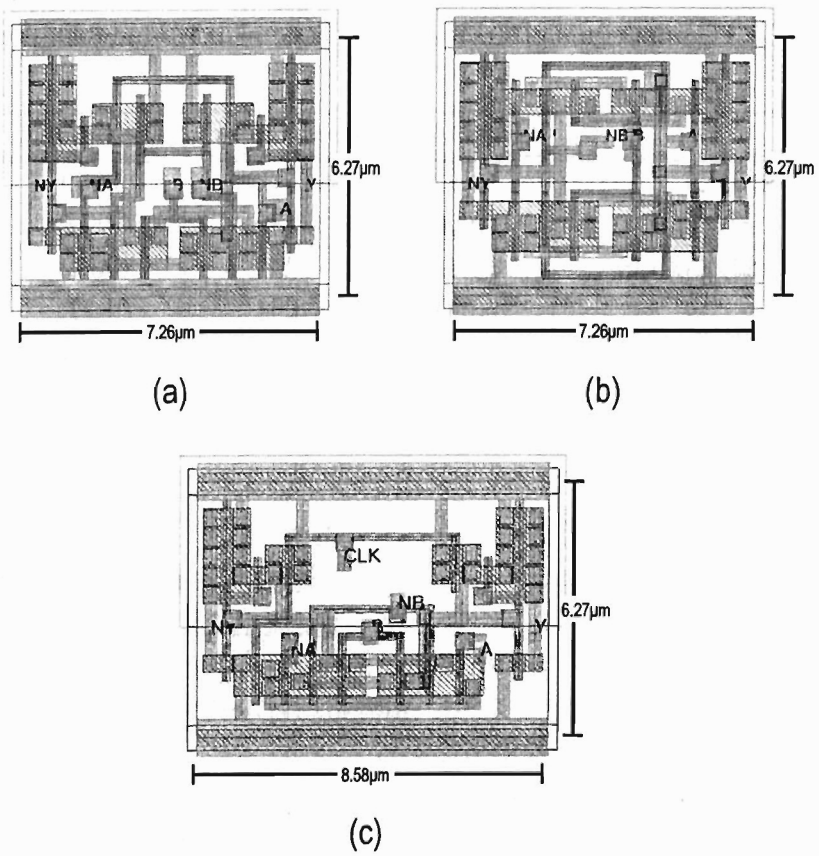

Figure 2.11: Circuit layout of (a) ASDDL EXOR cell, (b) ASD-CMOS EXOR cell, and (c) DCVS-DOMINO EXOR cell in a 0.18-μm CMOS technology.

形の構造として考えた場合，ASDDL と DCVS-DOMINO は NMOS トランジスタの数が多くなり，PMOS トランジスタを配置する面積は NMOS 側に比べて非常に小さなものとなる．ASD-CMOS では，NMOS ネットワークと PMOS ネットワークは双対構造を持ち，NMOS トランジスタと PMOS トランジスタの個数は等しく，PMOS トランジスタは NMOS トランジスタの個数を越えることはない．すなわち，Fig. 2.11 に示すようにそれら 3 つの論理ゲートの面積は主に NMOS トランジスタの個数によって決定される．ASDDL と ASD-CMOS はクロック信号を入力とするトランジスタが存在しないことから，DCVS-DOMINO に比べて NMOS トランジスタの個数を少なくすることができ，Fig. 2.11 からもわかるように

Table 2.5: Comparison of 16-bit multipliers in 0.18-μm technology at 1.8V.

	Delay [ns]	Power [mW]	Area [mm^2]
ASDDL	1.82	15.30	0.059
	(0.96)	(0.80)	(0.92)
ASD-CMOS	1.78	18.82	0.062
	(0.94)	(0.98)	(0.97)
DCVS-DOMINO	1.89	19.15	0.064
	(1.00)	(1.00)	(1.00)

論理ゲートの面積を小さくすることが可能となる．

Fig. 2.8 に示したアーキテクチャと同じ乗算器を DCVS-DOMINO 論理ゲートを用いて設計した．これらを自動配置配線によりレイアウトした後，RC 抽出したものを SPICE シミュレーションにより比較評価した結果を Table 2.5 に示す．ここで，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．

DCVS-DOMINO の論理ゲートを用いて設計した 16 ビット符号付き乗算器の遅延時間は 1.89ns であった．ASDDL および ASD-CMOS で設計した乗算器の遅延時間は，この DCVS-DOMINO の遅延時間に比べて 96%，94%であり，ASDDL と ASD-CMOS は DCVS-DOMINO よりも高速に動作することを確認した．しかしながら，PMOS ネットワークのトランジスタに論理ゲートの入力信号が接続されており，論理ゲート間の配線容量が DCVS-DOMINO よりも大きくなるため，遅延時間の改善率はそれぞれ 4%，6%程度にとどまっている．このことから，ASDDL/ASD-CMOS において，PMOS ネットワークのトランジスタの入力に接続する信号は，それぞれの信号の配線容量が均一になるように調節して回路を設計する必要がある．

ASDDL と ASD-CMOS の面積は DCVS-DOMINO よりも論理ゲート単体の面積が小さいことから，Fig. 2.12 に示すように回路全体としてもそれぞれ DCVS-DOMINO の 92%，97%となった．さらに回路のプリチャージを制御する信号線の大幅な削減によって総配線長が減少したことから，消費電力は DCVS-DOMINO に対してそれぞれ 20%，2%の削減を達成できた．

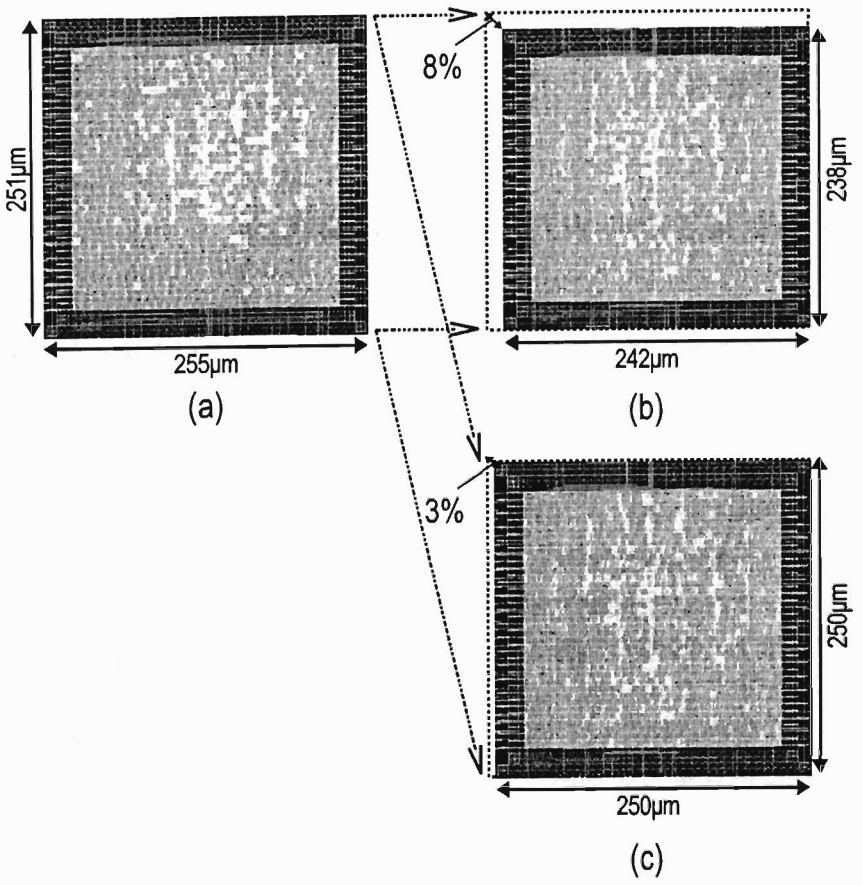

Figure 2.12: 16-bit multiplier layout of (a) DCVS-DOMINO, (b) ASDDL, and (c) ASD-CMOS in 0.18-μm CMOS technology.

2.5　テストチップの動作検証

これまでに，本提案回路であるASDDL/ASD-CMOSの特徴，および回路構成について説明し，サイクルタイムを高速な立ち上がり遷移である有効値の伝搬時間とほぼ等しくするサイクルタイム短縮アーキテクチャについて述べた．本節では，この提案回路方式の正常な動作を確認するために，0.13-μm CMOSプロセス，電源電圧1.2Vで試作したテストチップについて述べる．

2.5.1　セルフクロックによる非同期回路

動作検証のために設計したテストチップには，検証用回路としてFig. 2.8に示したアーキテクチャに基づいた16ビット符号付き乗算器をASD-CMOS論理ゲートを用いて設計し，搭載した．また，各回路ブロックの動作状況を確認し，その動作状況から回路全体を制御するためのクロック信号をチップ内部で生成するセルフクロックで動作するように回路全体が設計されている．

近年では，動作速度の高まりや回路規模が大きくなってきているため，同期クロックの発生回路や分配回路が複雑化しており，クロック設計がチップ設計上で困難な問題となってきている．そこで，各回路ブロック間の同期を取るための統一したクロック信号を使用しない非同期回路の研究が多く報告されている[65–72]．非同期回路はクロック信号の分配に要していた消費電力を削減でき，固有の周波数を持たないことから不要輻射雑音(EMI)を低くできる．また，回路ブロック間で処理の終了を確認してから次の演算ステップに進むため，トランジスタの性能バラつきに伴う問題が生じにくいなどの利点がある．

ASD-CMOS乗算器はTable 2.4にも示したように高速で動作するため，高い周波数のクロック信号を必要とする．しかしながら，I/Oの関係から高周波数クロックをチップ外部から直接入力することは不可能であり，精度のよい高周波数クロックを発生させる回路の設計は非常に難しい．そのため，グローバルクロックを使わずにシステムを制御する非同期で動作するように回路を設計した．また，これにより動作開始信号を切り替えるだけで，チップ内部のASD-CMOS乗算器が動作を開始するため、実測を容易に行うことができる．

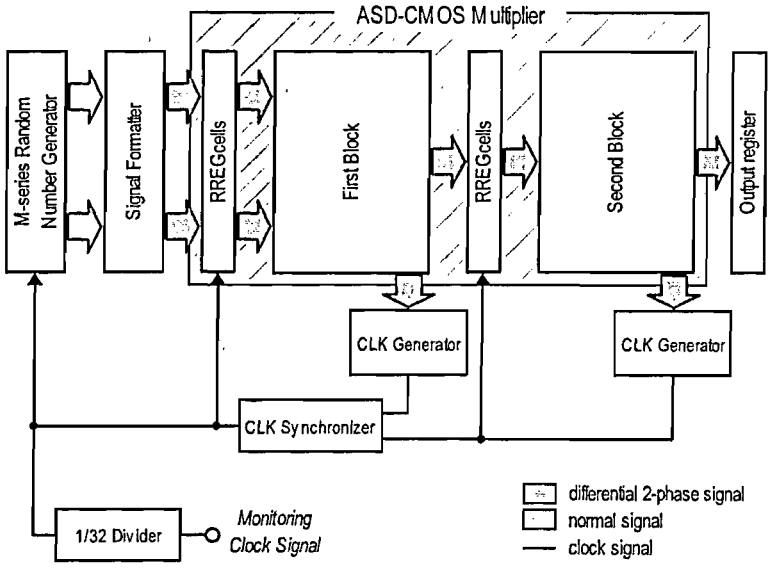

Figure 2.13: Block diagram of ASD-CMOS multiplier with CMOS controller for self-clocking.

2.5.2 テストチップの回路構成

テストチップのブロック図をFig. 2.13に示す．テストチップにはASD-CMOS論理ゲートで設計した16ビット符号付き乗算器の他に制御回路が搭載されている．制御回路はM系列乱数発生回路 (M-series Random Number Generator)，信号変換回路 (Signal Formatter)，クロック生成回路 (CLK generator)，クロック同期回路 (CLK synchronizer)，および32分周回路 (1/32 Divider) で構成されており，それらは全てスタティックCMOSで設計されている．さらに，ASD-CMOS論理ゲートを用いて設計した16ビット符号付き乗算器とスタティックCMOSで設計した制御回路のレイアウト配置はFig. 2.14のような構成となっている．

以下，制御回路を構成しているそれぞれの回路について説明する．

M系列乱数発生回路

M系列の疑似乱数を発生させる回路であり，ASD-CMOS乗算器に入力する2つの入力ベクタを生成する．生成された入力ベクタはスタティックCMOSで扱う1線1相の信号である．

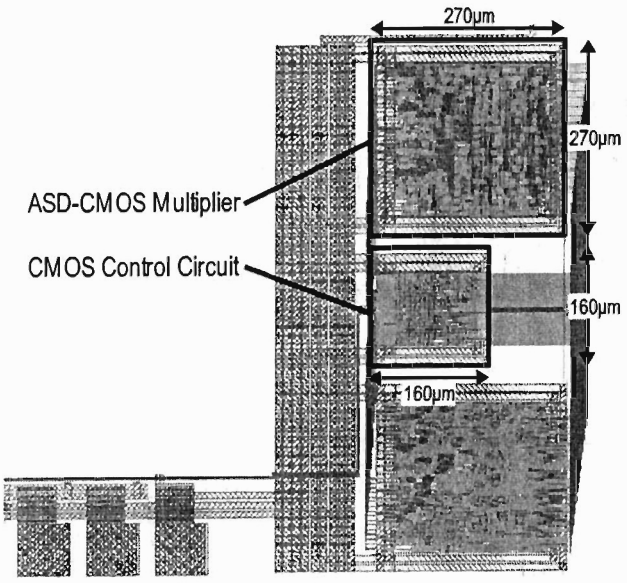

Figure 2.14: ASD-CMOS test chip layout.

また，回路に備え付けられたスキャンパスにより外部入力から固定の入力ベクタを入力することも可能となっている．

信号変換回路

M系列乱数発生回路で生成された入力ベクタを正負両論理の2線の信号に変換し，その変換された入力ベクタをASD-CMOS乗算器に入力する．

クロック生成回路

クロック発生回路は検証対象であるASD-CMOS乗算器の前段回路ブロックと後段回路ブロックが演算とプリチャージを全て完了しかたどうかを確認し，それらに基づいて回路を次の動作状態に移行するためのクロック信号を生成している．

それぞれの回路ブロックの出力信号には有効値である{0,1}と{1,0}と，休止値{0,0}が伝搬されてくる．それらの値より，Table 2.6に示すように現在の回路の動作状態が確認でき，これに従って次の状態へ遷移させる．

Table 2.6: Operation state of self-clocking circuit in test chip.

$\{x,\overline{x}\}$	state	→	next state
{0,1}	operation	→	pre-charge
{1,0}	operation	→	pre-charge
{0,0}	pre-charge	→	operation

クロック同期回路

ASD-CMOS は演算とプリチャージを交互に行なう．そこで，分割されたそれぞれの回路ブロックで演算状態とプリチャージ状態が正確に交互に実行されるようにするため，前後段回路ブロックの出力信号から生成された 2 つのクロック信号は同期を取ったのちに M 系列乱数発生回路と前段の RREGcell に分配される．それらのクロック信号が M 系列乱数発生回路に入力されることによって，各サイクル毎に新しい入力ベクタが乗算器に供給されるため，外部クロックなしで動作を行なうことができる．

32 分周回路

内部で生成されたクロック信号は，32 分周回路を通ってチップ外部に出力される．この信号から内部で動作している回路のサイクルタイムを得ることができ，チップ外部で測定されるサイクルタイムは 32 回の演算を行なった合計値となる．

ここで，外部で測定される測定用クロック信号は ASD-CMOS 乗算器の演算時間に加えてクロック生成および入力ベクター生成に要する遅延時間が加算されたものとなっているため，乗算器単体の遅延時間とは異なる．

2.5.3 実測による動作検証と性能評価

前節で説明したテストチップの動作検証と性能評価について述べる．

電源電圧 1.2V，セルフクロックで回路を動作させたときの測定用クロック信号の波形を Fig. 2.15 に示す．Fig. 2.15 から，チップ内部で生成されたクロック信号が 32 分周回路を通して正確に実測できている．このク

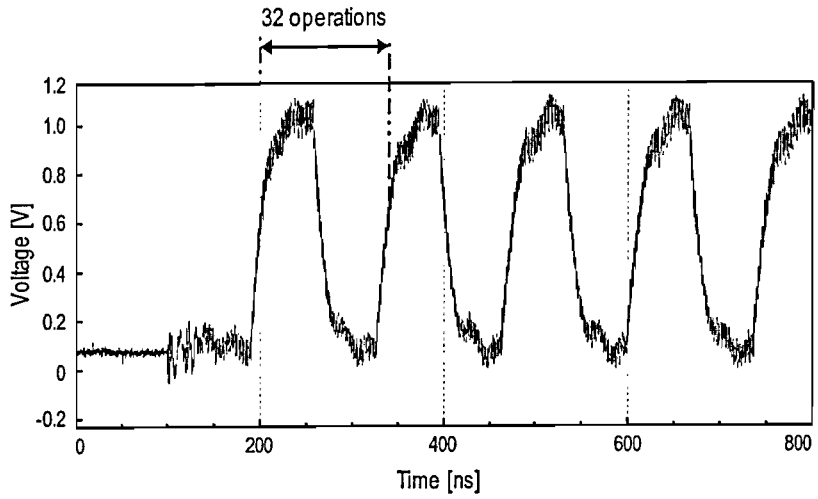

Figure 2.15: Measured waveform of self-clocking operation.

ロック信号は分割したそれぞれの回路ブロックの出力に有効値と休止値が交互に到着することによって生成されることから，それぞれの値が適切に回路内を伝搬し，動作していることが確認できる．

また，この信号より内部で動作している ASD-CMOS 乗算器の 1 演算に要するクロックサイクルは 4.25ns であることがわかる．前節でも説明したが，このクロックサイクルは検証対象の ASD-CMOS 乗算器単体のサイクルタイムではなく，クロック生成および入力ベクタ生成に要する遅延時間が加算されたものとなっている．

ここで，電源電圧 1.2V における実測と SPICE シミュレーションの比較結果を Table 2.7 に示す．Table 2.7 において，クロックサイクルはセルフクロックで動作させたときの 1 動作あたりのサイクルタイムであり，ASD-CMOS 乗算器の演算時間に制御回路の遅延時間を含んだ時間となっている．また，消費電力も同様のセルフクロック動作時のものである．クロックサイクルと消費電力における実測とシミュレーション間の誤差はそれぞれ 4%，11% であり，その結果はよく類似していることがわかる．実測したクロックサイクルは本来の ASD-CMOS 乗算器の演算時間と異なるため，実測と SPICE シミュレーションによるクロックサイクルの比率からチップ内の ASD-CMOS 乗算器の遅延時間を計算すると，Table 2.7 に示すように 1.57ns で動作していると考えられる．

Table 2.4 に示した 0.18-μm CMOS プロセス，電源電圧 1.8V のシュミ

Table 2.7: Comparison of measurement and simulation of ASD-CMOS 16-bit multiplier in 0.13μm technology at 1.2V.

	Measurement	Simulation
CLK cycle[ns]	4.25	4.43
Power[mW]	12.58	11.19
Delay[ns]	1.57	1.63

レーション結果と比較すると，ASD-CMOS 乗算器の遅延時間は 12%，消費電力は 33% 改善した．0.13-μm CMOS プロセスで試作した乗算器は 0.18-μm CMOS プロセスの乗算器に比べて回路全体の最適化があまりなされていないことから，0.18-μm CMOS プロセスで設計した乗算器のようにトランジスタサイジングなどを最適化することでさらなる性能向上が期待できる．

2.6 むすび

近年の SoC 開発において，高速低消費電力のデジタルプロセッサの要求はますます増加する傾向にある．そのため，高速動作可能な回路方式が報告されているが，遷移確率の高い制御信号を全ての論理ゲートに分配する必要があり，消費電力が増加する．本章では，従来の高速回路方式よりも高速動作と低消費電力を実現する ASDDL と ASD-CMOS を提案した．ASDDL/ASD-CMOS は正論理と負論理の信号で論理値を表現する 2 線式論理回路であり，論理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することで，スタティック CMOS に比べて格段に短い遅延時間を実現することができる．

ASDDL/ASD-CMOS は立ち上がり遷移となる演算の遅延時間を高速化することで，1 回の演算時間は高速になるが，サイクルタイムは長くなる．そこで，このサイクルタイムを大幅に短縮するサイクルタイム短縮アーキテクチャを考案した．このサイクルタイム短縮アーキテクチャは設計した回路を前後半の 2 つの回路ブロックに分割し，分割したそれぞれの回路ブロックで演算とプリチャージを交互に行なう．これにより，演算の裏にプリチャージを隠すことができ，演算時間とプリチャージ時間の総和であったサイクルタイムが演算時間とほぼ等しくなるため，サイクル

タイムの大幅な短縮を可能とした.

提案したASDDLとASD-CMOSを用いて，0.18-μm CMOSプロセスにて16ビット符号付き乗算器を設計し，比較評価を行った．ASDDLとASD-CMOSで設計した乗算器の遅延時間はそれぞれ1.82ns，1.78nsであり，これら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ライブラリを用いて，遅延時間最小の条件で設計したスタティックCMOSの68%，66%であり，スタティックCMOSでは到達不可能な高速動作が実現できることを示した.

また，高速動作可能なDCVS-DOMINOとの比較では，ASDDL/ASD-CMOSはプリチャージのための制御信号を必要としないことから，回路構成において遅延時間，消費電力が改善されることを示した．これは，ASDDL/ASD-CMOSと同様のアーキテクチャを用いて設計したDCVS-DOMINO乗算器のシミュレーション結果で証明され，DCVS-DOMINOよりも高速動作を実現した．さらに，面積はDCVS-DOMINOの92%，97%であり，それに伴って消費電力はそれぞれ20%，2%削減した.

0.13-μm CMOSプロセスで試作したテストチップでは，各回路ブロックの動作状況に従って回路全体の制御に用いるクロック信号の生成をチップ内部で行なうセルフクロック方式で動作するように回路を設計し，ASD-CMOSの正常な動作を確認できた．また，0.18-μm CMOSプロセス，電源電圧1.8Vで設計したものに比べて0.13-μm CMOSプロセス，電源電圧1.2Vで設計した乗算器の遅延時間は12%改善した．この性能はトランジスタのサイジングの最適化を厳密に行うことでさらに向上することが期待できる.

第3章

2線2相式論理回路の自動設計システム

3.1 まえがき

本章では，2線2相式論理回路であるASDDL (Asymmetric Slope Differential Dynamic Logic) とASD-CMOS (Asymmetric Slope Differential CMOS) の自動設計を可能とする論理合成手法を提案する．

現在，LSIに関連した技術は急速な勢いで発展しており，LSIを使用することは高機能な製品を開発するためには必要不可欠なものになっている．同時に，LSI自体の高集積化も進み，数千ゲート規模の回路開発が中心だった頃から，数億ゲート規模の大規模集積回路へと移行が進んでいる．これらのLSIの高機能化と高集積化に伴い，回路の設計はますます複雑で困難なものになってきている．しかも，新製品の市場への投入を早め，先駆者利益を得るためには，可能な限り製品の開発期間を短くしなげればならない．このような設計の大規模化と設計期間の短縮を同時に実現するために自動設計環境が構築され，現在のLSI設計においては無くてはならないものとなっている．

2章で提案したASDDL/ASD-CMOSはスタティックCMOSでは到達不可能な高速動作を実現し，高速ダイナミック回路であるDCVS-DOMINOよりも低消費電力であることを示した．このASDDL/ASD-CMOSを用いることにより，高速動作を必要とするプロセッサ等の性能の飛躍的な向上が期待できる．しかしながら，現在，多くの開発現場で使用されている自動設計ツールでは，正論理と負論理という2つのペアで論理を表現する2線の信号線は個別の信号線として取り扱われてしまう．これより，2線2相式論理回路であるASDDL/ASD-CMOSは正負両論理を持った論理ゲートをそのまま用いた論理合成を行なうことができず，大規模回路を短期間での設計することが非常に困難となる．

また，従来のダイナミック回路の合成手法と同様に，ASDDL/ASD-

CMOSはインバータを含まない回路を設計する必要がある．設計した回路ネットワークの論理ゲート間にインバータが存在すると，有効値は反転した論理値として伝搬されるが，休止値は{1,1}となるために誤動作を引き起こすためである．

そこで，全ての信号線が正負両論理で表現される2線2相式論理回路の自動設計環境を構築し，ASDDL/ASD-CMOSの自動設計を実現する．ASDDL/ASD-CMOS論理ゲートをそのまま用いて論理合成できるツールを1から開発するには，膨大なリソースを必要とすることから，スタティックCMOSで用いられている市販の論理合成ツールを利用する．同時に，市販のスタティックCMOS用の論理合成ツールで2線信号を持った論理ゲートを合成するために特殊な論理合成用ライブラリを考案する．また，論理ゲート間にインバータを含まない回路ネットワークの論理合成を実現する．本章では，本提案手法を用いて，実際に回路の論理合成に用いたライブラリについても説明し，その論理合成結果の比較評価を示す．

3.2 ASDDL/ASD-CMOS論理合成手法

本節では，スタティックCMOS用に確立された高度な論理合成ツールを用いたASDDL/ASD-CMOSの論理合成手法について述べる．

現在，多くの開発現場で用いられている市販のスタティックCMOS用論理合成ツールは，論理ゲート間の駆動力と負荷容量のバランスを最適化し，多数用意された論理合成アルゴリズムを用いることで，設計者が求める様々な条件に合った回路を短期間で設計できる．ASDDL/ASD-CMOSの自動設計にスタティックCMOS用の論理合成ツールを用いることで，これらの高度に確立された設計環境を最大限に利用することができる．

提案するASDDL/ASD-CMOS論理合成手法では，スタティックCMOS用の論理合成ツールを利用するために，ASDDL/ASD-CMOSの論理ゲートを疑似的に定義した中間ライブラリ (Intermediate library) を用いて合成する．また，合成時にはインバータが論理ゲート間に存在しない中間ネットリスト (Intermediate netlist) を生成し，その中間ネットリストを変換ツール (Translation tool) によってASDDL/ASD-CMOSの論理ゲートに置き換えることで，論理ゲートとその間の配線情報を含んだ最終ネットリスト (Final netlist) を生成し，それを用いて全体レイアウトを作成する．これらの方法を用いることにより，フルカスタム設計に非

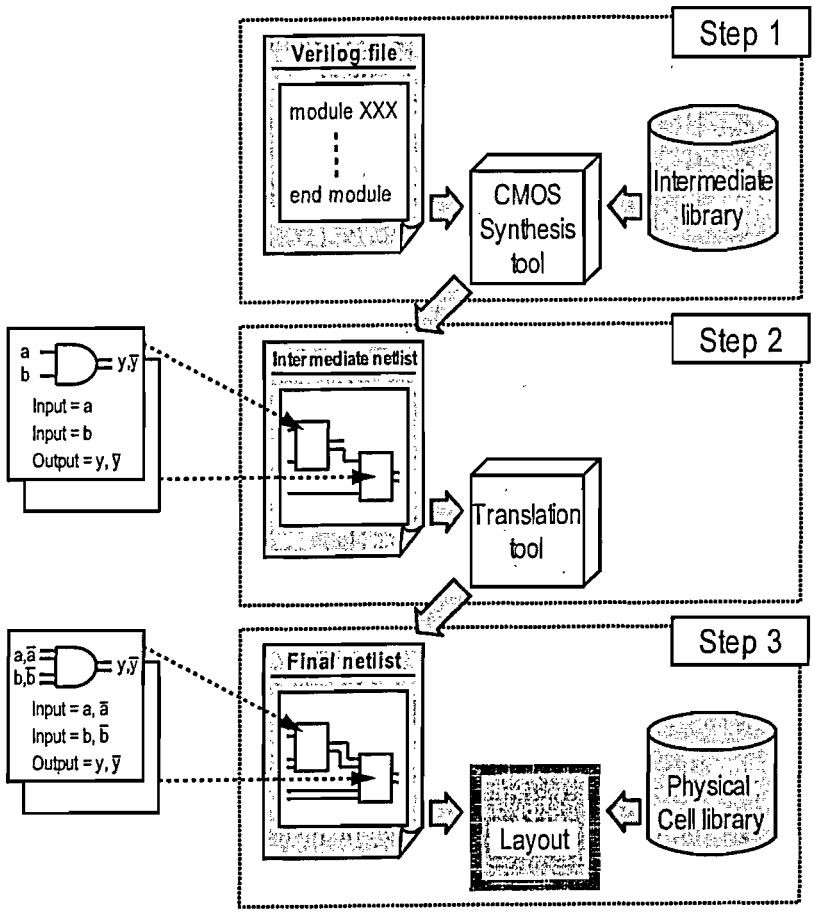

Figure 3.1: ASDDL/ASD-CMOS logic synthesis flow.

常に近い性能を持った回路の論理合成を可能とし，2線2相式論理回路ASDDL/ASD-CMOSの特徴を最大限に生かした論理合成を実現する．

3.2.1 ASDDL/ASD-CMOS自動設計の流れ

ASDDL/ASD-CMOSの自動設計フロー(Fig. 3.1)を以下に示す．

【ステップ1】

中間ライブラリを用いてCMOS用の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．

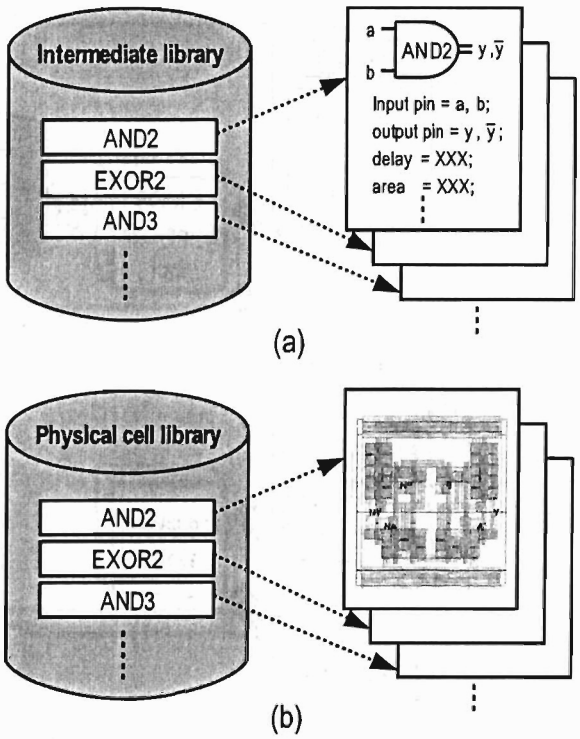

Figure 3.2: Summary of (a) intermediate library and (b) physical cell library for ASDDL/ASD-CMOS logic synthesis.

【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/ASD-CMOS論理ゲート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，サイクルタイムを短縮するためのアーキテクチャを回路に自動的に適用する．

【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，スタティック 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．

ステップ1で論理合成する Verilog ファイルは，スタティック CMOS を合成するときに用いるものとまったく同じ記述であり，ASDDL/ASD-CMOS を論理合成するために信号線を正負両論理の2線に書き換えたり，

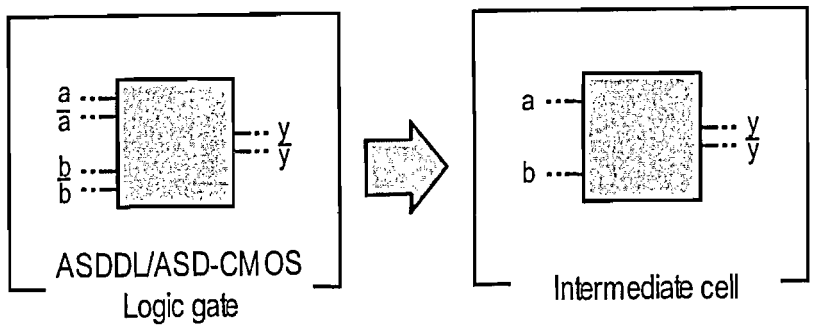

Figure 3.3: Intermediate cell defined in intermediate library.

特殊な記述や構文を用いる必要はない．そのため，中間ライブラリを用いるという点だけを注意すれば，設計する回路に合わせた制約条件を与えて，スタティック CMOS と同様に論理合成することで中間ネットリストを作成する．

中間ネットリストは，中間ライブラリ内で ASDDL/ASD-CMOS 論理ゲートを疑似的に定義した中間セルと呼ばれる論理ゲートで構成されている．そのため，その中間ネットリストを用いてレイアウトを行なうことはできない．レイアウトを行なうために用いる 2 線信号の ASDDL/ASD-CMOS 論理ゲートで構成された最終ネットリストは，ステップ 2 において開発した変換ツールによって中間ネットリストから生成される．この最終ネットリストはインバータがまったく存在しないネットリストとなっている．

最終ネットリストが生成された後は，ステップ 3 のように配置配線用のライブラリを用いて全体レイアウトを作成する．合成時に用いる中間ライブラリは Fig.3.2(a) に示されているように，論理合成のために必要な信号線やパラメータが定義されている．一方，配置配線用ライブラリはすべての ASDDL/ASD-CMOS 論理ゲートのレイアウトデータである (Fig. 3.2(b))．

3.2.2 中間ライブラリを用いた論理合成

本節では，市販のスタティック CMOS 用の論理合成ツールを利用して論理合成するために必要となる中間ライブラリについて説明し，インバータを含まないネットリストを合成するための方法について述べる．

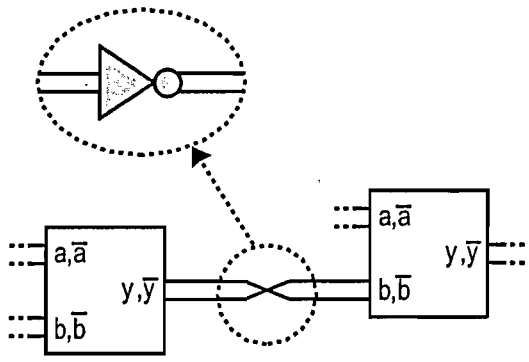

Figure 3.4: Inverter function on ASDDL/ASD-CMOS circuits.

市販のスタティック CMOS 用の論理合成ツールでは，正論理の入力信号 a に対して，負論理の入力信号 $\bar{a}$ は a の負論理として扱われず，a とはまったく別の信号と識別される．一方，負論理の出力信号 $\bar{y}$ は $\{y, \bar{y}\}$ というペアの信号であるとは定義できないが，y の反転論理の出力信号であるということは定義できる．そこで，中間ライブラリで定義されている中間セルは，Fig.3.3 に示すように ASDDL/ASD-CMOS 論理ゲートを 1 線の入力と 2 線の出力 ($\{y, \bar{y}\}$) で定義する．

1 線の入力と 2 線の出力の中間セルで定義された中間ライブラリで合成した後，変換ツールで ASDDL/ASD-CMOS の論理ゲートに変換する．この際，信号の再接続を必要とするため，設計した回路が合成時の性能よりも悪くなる可能性がある．そのため，中間ライブラリ内で定義されている中間セルの入出力ピンパラメータは，論理ゲートの正負両論理の信号の最悪値にする必要がある．すなわち，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性能よりも悪化することがないように，パラメータを設定する．

ここで，ASDDL/ASD-CMOS では，2 線の信号線がインバータを通ることによって休止値が {1,1} となり，誤動作を引き起こすため，インバータを取り除く必要がある．ASDDL/ASD-CMOS 論理ゲートの特徴として，Fig.3.4 に示すように 2 線の信号線を入れ換えることによって反転論理を生成できるため，回路の変換の際にインバータは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論理ゲート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．なぜならば，論理合成ツールは前段論理ゲートの駆動力と後段論理ゲートの

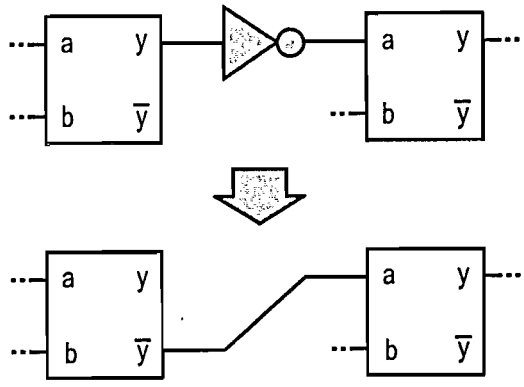

Figure 3.5: Inverting connection of signal line between intermediate cell.

入力容量から遅延タイミングを最適化するためである．すなわち，前段論理ゲートの駆動力とインバータの入力容量，インバータの駆動力と後段論理ゲートの入力容量で最適化されているが，インバータがなくなると前段論理ゲートの駆動力と後段論理ゲートの入力容量という関係となるためである．これらのことから，遅延タイミングの最適な回路を自動設計する際には，インバータは後処理で取り除くのではなく，論理合成の段階でなくす必要がある．

そこで，論理合成時における反転論理は 2 線の出力信号 ($\{y, \overline{y}\}$) で生成される．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，Fig.3.5 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ため，中間セル間のインバータはなくなり，駆動力と負荷容量のバランスが維持された状態でインバータを含まない回路を合成することができる．一方，設計した回路全体の入力側においては中間セルの 2 線の出力で反転論理を生成できない．しかしながら，全体回路の入力側に使用されたインバータは遅延タイミングを崩す原因にはならず，削除されたとしても遅延時間が改善するだけなので，この部分のインバータは後処理を施す変換ツールで配線情報だけに変換する．

スタティック CMOS 用の論理合成ツールで用いる合成ライブラリでは，インバータの定義はなくすことができない．そこで，中間ライブラリ内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ている．これにより，合成時にインバータの使用を制限している．しかしながら，インバータは反転論理を生成するためだけでなく，駆動力確保のために用いられることもある．そこで，駆動力の異なったバッファを

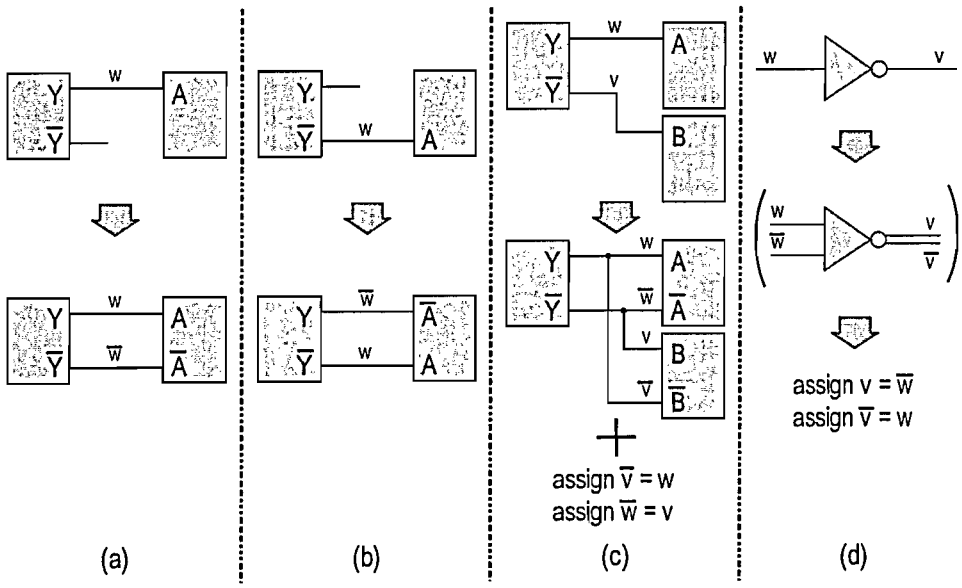

Figure 3.6: Translation for each pattern generated by logic synthesis using intermediate library.

中間ライブラリ内で数多く定義しておき，駆動力確保のためにインバータを利用させないようにする．

3.2.3 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/ASD-CMOS論理ゲートに置き換える．その際，中間セル間の信号線を適切に接続し直す．本節では，中間セルをASDDL/ASD-CMOS論理ゲートに置き換えるときに行なう信号線の再接続について説明する．

中間ライブラリを用いてスタティックCMOS用の論理合成ツールで合成すると，中間セル間の信号線はFig.3.6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．それぞれの接続パターンに対して，以下のように変換ツールが自動的に再接続を行なう．

Fig.3.6(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 (Fig.3.6(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 (Fig.3.6(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，Fig.3.6(d) のように "$assign\ v = \overline{w}$, $assign\ \overline{v} = w$" という記述を追加する．

3.2.4 サイクルタイム短縮アーキテクチャの自動適用

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場合，適切な箇所に Rcell，RREGcell 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．本節では，変換ツールにおける自動的な Rcell，RREGcell の挿入機構について説明する．

初めに，中間ライブラリを用いてスタティック CMOS 用の論理合成ツールで合成すると，Fig.3.7(a) に示すように，ASDDL/ASD-CMOS 論理ゲートのみで構成された回路が生成される．これの回路に対して，スタティック 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，Fig.3.7(b) に示したようにパイプラインレジスタを含んだ回路に合成し直す．このパイプラインレジスタが RREGcell を挿入する箇所を特定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．開発した変換ツールは，この挿入されたパイプラインレジスタの前段に接続されている論理ゲートが，その論理ゲートの論理機能を持ったまま RREGcell に変換され (Fig.3.7(c))，RESET 信号が適切に分配される．

さらに，休止値の伝搬時間は有効値の伝搬時間に比べて遅いため，以下のように分割したそれぞれの回路ブロックに Rcell が挿入される (Fig. 3.7(d))．まず，回路全体の入力側から順に有効値の伝搬時間が計算され，分割したそれぞれの回路ブロックの最大有効値伝搬時間が計算される．その後，各回路ブロックで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論理ゲートが Rcell に置き換えられる．休止値伝搬時間は置き換えられた Rcell から計算し直され，全ての論理ゲートの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．

これらの方法により設計した回路にサイクルタイム短縮アーキテクチャ

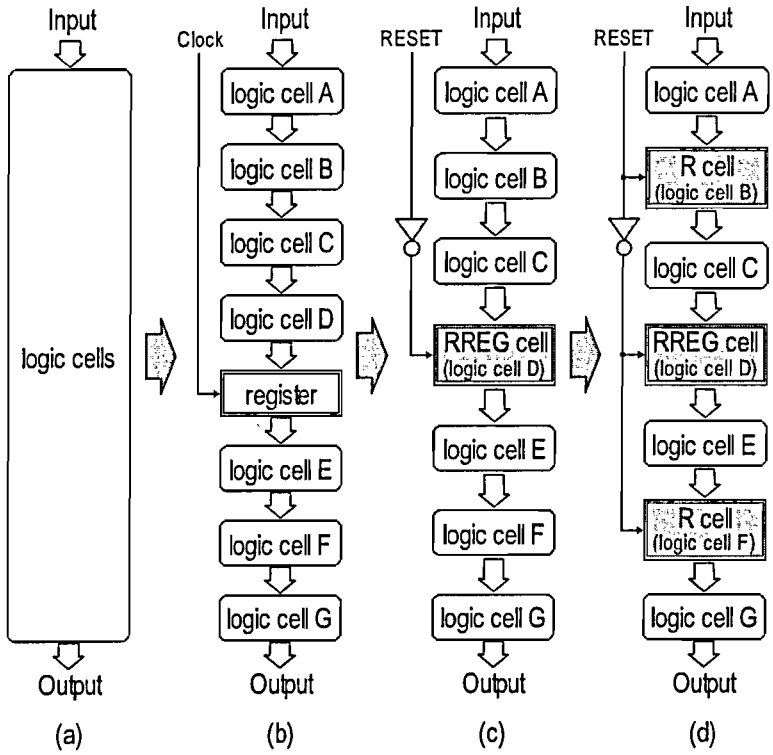

Figure 3.7: Applying cycle-time reduction technique.

が自動的に適用され，サイクルタイムの大幅な短縮化を実現する．

3.3 自動設計環境の構築と適用事例

本節では，提案した論理合成手法を評価するために，実際に構築したASDDL論理ゲートの自動設計環境について説明する．

提案した論理合成手法を用いる際には，スタティックCMOS用の論理合成ツール，変換ツール，中間ライブラリ，および配置配線用ライブラリを用意する必要がある．論理合成ツールと変換ツールは，どのような設計プロセスでも共通に使えるものであるのに対して，中間ライブラリと配置配線用ライブラリは設計するプロセスに応じて用意する必要がある．そこで，ASDDLの自動設計用に構築した中間ライブラリと配置配線用ライブラリについて述べる．

また，提案手法を用いて論理合成した適用事例について示し，それら

Table 3.1: 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

のSPICE シミュレーションによる回路評価，およびサイクルタイム短縮アーキテクチャの自動適用によってマルチサイクル化したときに生じる回路性能への影響について示す．

3.3.1 ASDDL 自動設計環境の構築

電源電圧 1.8V，0.18-μm CMOS プロセスにて，ASDDL 論理ゲートの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 ASDDL 論理ゲートの種類を Table 3.1 に示す．論理ゲートの駆動力は小さい順に 0X，1X$\cdots$8X で表している．

ここで，ASDDL 論理ゲートは入力や出力信号の 2 線を入れ換えることにより，1 つの論理ゲートで複数の論理関数を表現できる．例えば，Fig.3.8(a) に示すような AND の論理関数を持った論理ゲートがある．以下のように，この論理ゲートの入力と出力信号を入れ換えることで，Fig.3.8 のように (b)NAND, (c)OR, (d)NOR の論理関数を持った論理ゲートとして用いることができる．

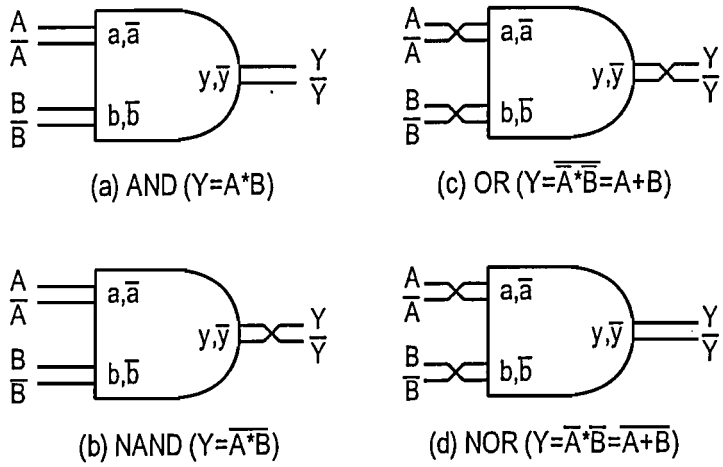

Figure 3.8: The variation of logic function in ASDDL logic cell.

NAND 出力の2線信号のみを入れ換える
$Y = \overline{A * B}$

OR 入力と出力の2線信号を入れ換える
$Y = \overline{\overline{A} * \overline{B}} = A + B$

NOR 入力の2線信号のみを入れ換える
$Y = \overline{A} * \overline{B} = \overline{A + B}$

このように，2入力のANDゲート1つで4つの論理関数を表現することができる．これらの性質により，ASDDLのライブラリは少ない論理ゲート数で構成することが可能である．そのため，構築したライブラリは12種類の論理ゲートで構成されており，これにより3入力以下の全ての論理関数を表現することができる．

また，サイクルタイム短縮アーキテクチャを適用した際には個々の論理関数を持ったRcellおよびRREGcellが必要となる．そこで，全ての論理ゲートに対してRcell，RREGcellを用意した．さらに，駆動力確保を目的に用いられる9種類の駆動力の異なるバッファも用意されている．配置配線用のライブラリでは，それら全ての論理ゲート，RcellとRREGcell，およびバッファのレイアウトで構成されている．

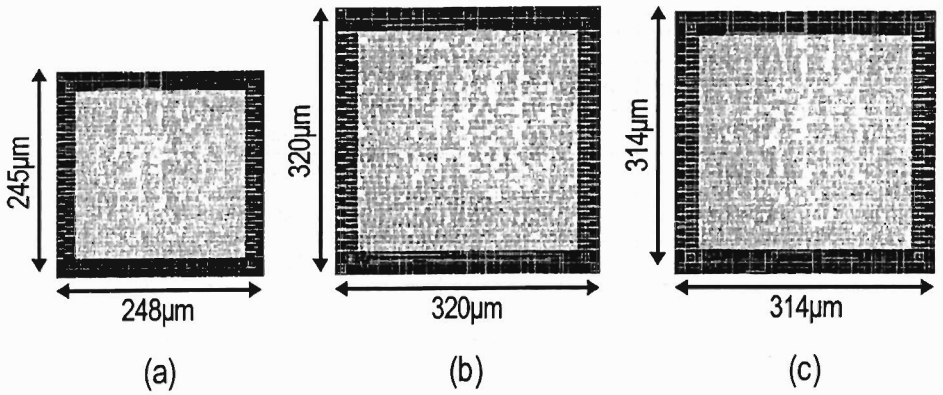

Figure 3.9: 16-bit ASDDL multiplier layout of (a) M16_Acla_A, (b) M16_Fcla_D, and (c) M16_D in 0.18-μm CMOS technology.

3.3.2 適用事例

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ビット符号付き乗算器を設計した．設計した16ビット符号付き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

それぞれの16ビット符号付き乗算器は，0.18-μm CMOS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，変換ツールにより最終的なネットリストに変換した．また，Fig.3.9に示すような回路全体のレイアウトは配置配線用ライブラリを用いて行なった．

0.18-μm CMOSプロセスにおいて，考案した中間ライブラリを用いて論理合成した16ビット符号付き乗算器で使用された論理ゲートの個数を

Table 3.2: Number of logic gates 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

Table 3.2 に示す．様々な制約条件やアーキテクチャで乗算器を設計したが，全ての乗算器で多く使用されている論理ゲートは AD2A(2-input NAND : $Y = A + B$) や AD3B(3-input EXOR : $Y = A \oplus B \oplus C$) などであった．一方で，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関数を持った論理ゲートも使用されていることが確認できる．

これらの論理ゲートはスタティック CMOS では，論理関数が複雑になりすぎて 1 つの論理ゲートでは容易に実現できない．一方，ASDDL は論理関数を表現する NMOS ネットワーク部分を BDD で設計するため，こういった複雑な論理関数も 1 つの論理ゲートで構成することができ，高速化に向いている．しかしながら，回路図エディタを使って論理ゲートレベルで全て回路を設計するフルカスタム設計では，これらの複雑な論理関数を持った論理ゲートは一見しただけでは回路のどの部分に用いてよいのかが分からないため，ほとんど使用されない．論理合成によって，このような論理ゲートを多用できることは，2 線 2 相式論理回路である ASDDL を自動設計する上で，非常に大きなメリットであり，正負両論理

Table 3.3: Comparison of 16-bit multipliers in 0.18-μm CMOS technology at 1.8V.

	Delay [ns]	Power [mW]	Area [mm^2]	Design Time
CMOS_M16	2.68	5.99	0.041	1 hour
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

である2線の信号線を持ったASDDLの特徴を最大限に引き出していると言える.

3.3.3 シングルサイクル回路の性能比較

提案した論理合成手法の有効性を検証するために，ASDDLを用いて設計した以下の5つの回路について比較評価を行った.

- CMOS_M16
  - スタティックCMOSで設計した16ビット符号付き乗算器
- FC_M16
  - フルカスタムで設計したASDDLの16ビット符号付き乗算器
- 3.3.2節で論理合成した3種類の16ビット符号付き乗算器
  - 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構築されたスタティックCMOS用のライブラリを用い，遅延時間が最も小さくなるように制約条件を付けて論理合成した乗算器である．FC_M16は回路図エディタを用いて全てを手動で乗算器を設計したフルカスタム回路であり，Fig. 2.8に示したアーキテクチャと同じWallace treeとBLCAで構成されている．また，FC_M16および提案した論理合成手法を用いて設計した3つの乗算器にサイクルタイム短縮アーキテクチャは適用していないため，これらの回路はシングルサイクルで動作する．以上のような条件において回路の比較評価を行なった.

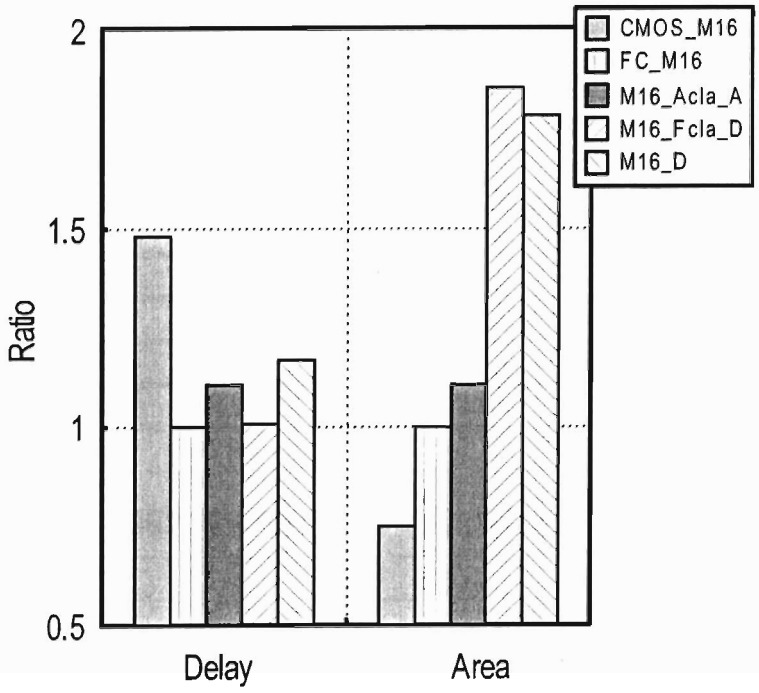

Figure 3.10: Performance ratio of 16-bit multipliers in 0.18-μm CMOS technology at 1.8V. The ratio is calculated as FC_M16 to be 1.00.

それぞれの回路を自動配置配線によりレイアウトを作成し，RC抽出した後，SPICEシミュレーションにより比較評価した結果をTable 3.3に示す．この比較結果は電源電圧は1.8Vで動作させた結果であり，消費電力は100MHz動作時の平均消費電力である．さらに，フルカスタムで設計した16ビット符号付き乗算器(FC_M16)を1としたときの性能比率をFig. 3.10に示す．

M16_Fcla_Dの遅延時間は1.82nsであり，これは最も高速に動作するように設計したスタティックCMOS乗算器と比べると32%の減少を成し遂げた．これは，フルカスタム設計したFC_M16と比較しても1%の増加に抑えることができ，フルカスタム設計の乗算器と同等の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論理ゲート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関数を持った論理ゲートが多く使用されたことから，M16_Acla_Aの消

費電力はフルカスタム設計のFC_M16の98%であった．また，一般的な論理合成ツールは僅かでも遅延時間を削減できるのであれば，面積を余計に大きくする傾向がある[73]．そのため，遅延最小の制約条件で論理合成した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．回路図エディタを用いて全ての回路を手作業で設計す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，提案した論理合成手法を用いて設計した16ビット符号付き乗算器は1時間足らずで設計することができた．フルカスタム設計では，回路を設計する前に最適だと考えられるアーキテクチャを選択する必要があり，そのアーキテクチャを用いて設計した回路が要求性能を満たさなかった場合には，さらに長い設計期間をかけて回路を作り直す必要がある．一方，論理合成では様々な制約条件，アーキテクチャを持った回路を短期間で設計することができるため，必要性能に合った最適なアーキテクチャを比較的簡単に見つけ出すことができ，高速動作可能な2線2相式論理回路ASDDLを適用した大規模LSIの短期設計が実現可能となる．また，この論理合成手法を用いることで，フルカスタムでは設計が困難となる規則性のないランダムロジックの設計も可能であり，高速動作を必要とするプロセッサなどの飛躍的な性能向上が期待できる．

3.3.4 マルチサイクル化による回路性能への影響

シングルサイクルで設計した16ビット符号付き乗算器にサイクルタイム短縮アーキテクチャを適用し，以下のような回路を設計した．

- 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アーキテクチャを自動適用した回路

- S_FC_M16
  - FC_M16にサイクルタイム短縮アーキテクチャを手作業で適用した回路

Tabel 3.4にサイクルタイム短縮化アーキテクチャを適用した16ビット符号付き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

Table 3.4: Comparison of 16-bit multipliers by applying cycle-time reduction technique in 0.18-μm CMOS technology at 1.8V.

	Delay [ns]	Minimum Cycle [ns]	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

力は100MHz 動作時の平均消費電力である．また，サイクルタイム短縮アーキテクチャを適用する前の回路性能を 1 としたときの増加量を Fig. 3.11 に示す．サイクルタイム短縮アーキテクチャが適用されることによって RREGcell と Rcell 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%未満に抑えることができた．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アーキテクチャ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．

3.4 むすび

現在，集積回路の微細プロセス技術の進化により，回路の設計はますます複雑で困難なものになってきている．設計の大規模化と設計期間の短縮を同時に実現するために，論理合成は大規模 LSI の設計に必要不可欠なものとなっている．本章では，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 ASDDL/ASD-CMOS 回路方式の論理合成手法を提案した．ASDDL/ASD-CMOS 論理ゲートをそ

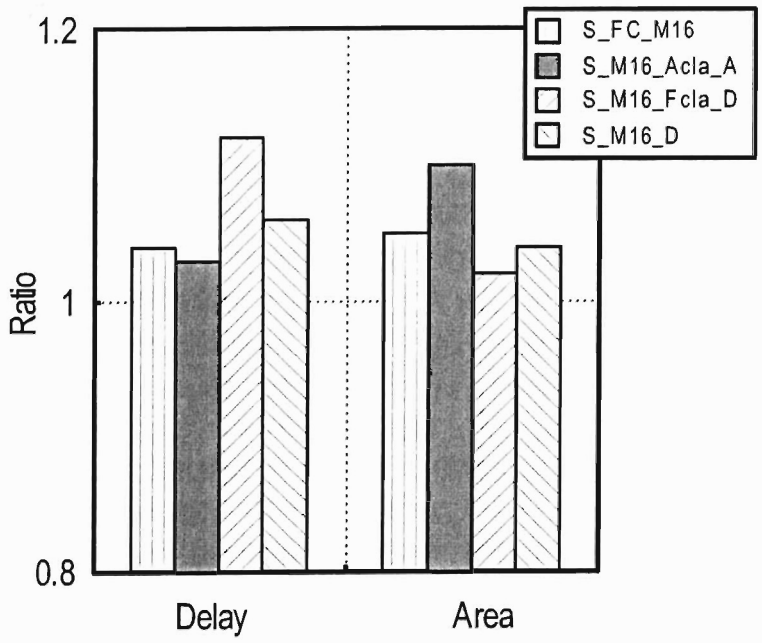

Figure 3.11: Performance ratio of multiple cycle multipliers by applying cycle-time reduction technique. The ratio is calculated as single cycle design to be 1.00.

のまま論理合成できるツールを作るには，膨大なリソースと手間を必要とすることから，スタティック CMOS に確立された市販の論理合成ツールを利用する．CMOS 用の論理合成ツールを用いるために，2 線の信号を持った論理ゲートを合成するための中間セル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/ASD-CMOS の自動設計を実現した．

中間ライブラリは，CMOS 用の論理合成ツールで 2 線の信号線を持った ASDDL/ASD-CMOS 論理ゲートの論理合成を実現するため，入力は 1 線，出力は 2 線の中間セルで定義した．正負両論理の 2 線で出力信号を定義することでインバータを用いずに反転論理を生成でき，論理ゲート間の駆動力と負荷容量のバランスが維持されたインバータなしの回路を合成することが可能となった．合成時にインバータの使用を制限するために，中間ライブラリ内で定義されているインバータのパラメータを通常よりも大きく設定した．

中間ライブラリで合成した結果に後処理を施すために変換ツールを開

発した．変換ツールは中間セルで構成された合成結果を ASDDL/ASD-CMOS 論理ゲートに置き換え，信号線の再接続を行なう．また，Rcell と RREGcell に配置し，サイクルタイム短縮アーキテクチャを自動で適用する機構を変換ツールに備えた．

提案した論理合成手法を用いて回路を設計するために，ASDDL 論理ゲートを論理合成するための中間ライブラリと配置配線用ライブラリを構築した．構築したそれらのライブラリはわずか 12 種類の論理ゲートで構成されているにも拘らず，3 入力以下の全ての論理関数を表現することができる．このライブラリを用いて構築した自動設計環境で，様々な制約条件を持った 16 ビット符号付き乗算器を 0.18-μm CMOS プロセスで設計し，比較評価を行なった．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 ASDDL16 ビット乗算器の遅延時間は 1.82ns 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を実現した．また，サイクルタイム短縮アーキテクチャの適用による回路性能の増加を 10% 未満に抑えることができた．

提案した論理合成手法により，様々な制約条件，アーキテクチャを持った回路を短期間で設計することができ，非対称な信号遷移を持った 2 線 2 相式論理回路 ASDDL/ASD-CMOS の大規模 LSI への適用が可能となり，飛躍的な性能向上が期待できる．

第4章

2つの動作モードを有する高速論理回路方式

4.1 まえがき

本章では，高速モードと低消費電力モードの2つの動作モードを持ったASDMDL (Asymmetric Slope Dual Mode Differential Logic) 回路方式を提案する．

製造プロセスの微細化に伴い，チップ上に集積されるトランジスタの数も増加し，全体的な消費電力と発熱も増えてきている．携帯電話やノートPCのようなバッテリ駆動の多機能モバイル機器に対する需要が高まる中，動作速度を向上させる技術に加えて，消費電力を抑えて長時間のバッテリ寿命を実現する技術も必要不可欠となっている．そこで，演算実行中にプロセッサの負荷の大きさに応じて電源電圧と動作周波数を動的に制御するプロセッサも存在する．これらのプロセッサは，処理量の多い演算を実行するときにはフルスピードで動作させ，処理量が少ないときには動作周波数と電源電圧を下げることで，常にフルスピードで動作させるプロセッサに比べて消費電力を抑えることができる．

ASDDL (Asymmetric Slope Differential Dynamic Logic) とASD-CMOS (Asymmetric Slope Differential CMOS) はプリチャージ制御のための信号線を大幅に削減することで，DCVS-DOMINOよりも低消費電力であることを2章で示した．しかしながら，スタティックCMOSと比べるとASDDL/ASD-CMOSの消費電力は増大する．なぜならば，設計した回路の全ての論理ゲートで立ち上がりと立ち下がりの2つの信号遷移が全てのクロックサイクル毎に発生するためである．これは，消費電力を低減させるために動作周波数を下げたとしても，立ち上がりと立ち下がりの2つの信号遷移は必ず存在するため，大きな消費電力の低下は望めない．

そこで，ASDDL/ASD-CMOSと同様の高速動作を実現し，処理量の少ない演算を実行するときにはスタティックCMOSと同程度の消費電力を

実現するASDMDLを提案する．また，3章で示した論理合成手法を用いて，ASDMDLとスタティックCMOSを混載したデジタルコアの論理合成・自動配置配線を実現し，クリティカルパス部分を中心にASDMDLを適用したASDMDL/CMOS混在プロセッサの性能検証について示す．

4.2　2モード2線式論理回路ASDMDL

ASDMDLは，ASDDL/ASD-CMOSと同様に演算の前にプリチャージを行ない，論理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することで高速動作を実現する2相動作モードと，プリチャージなしで動作する1相動作モードを切替えることのできる2モード2線式論理回路である．設計した回路の動作タイミングに応じてこの2つの動作モードを切替えることにより，最適な回路性能を引き出すことができる．本節では，ASDMDLが持つ2つの動作モードについて説明し，回路構成とその動作原理について述べる．また，ASDMDLの性能評価について示す．

4.2.1　高速モードと低消費電力モード

ASDMDLは，ASDDL/ASD-CMOSと同様に正負両論理の信号で論理値を表現する2線式論理回路であり，2線の信号線によって表現される値は，論理0({0,1})と論理1({1,0})，および回路のプリチャージとなる{0,0}の3種類である．また，ASDDL/ASD-CMOSと同程度の高速動作を実現し，さらに高速動作を必要としない場合には消費電力をスタティックCMOSと同等以下に引き下げることのできる回路方式であり，Fig. 4.1に示すように2相で動作する高速モード(ASDMDL-2ϕ)と1相で動作する低消費電力モード(ASDMDL-1ϕ)という2つの動作モードを持っている．

ASDMDL-2ϕはASDDL/ASD-CMOSとまったく同じ動作を行なう．すなわち，演算とプリチャージを交互に行ない，立ち上がり遷移を立ち下がり遷移に対して最小とすることで，高速動作を実現する．演算を行なう前に{0,0}が入力されることで回路をプリチャージするため，回路の遅延時間は立ち上がりの伝搬時間となり，立ち上がり遷移をより高速となるように回路を設計する．この立ち上がりと立ち下がりの非対称な信号遷移はトランジスタサイジングにより実現する．

一方，ASDMDL-1ϕはスタティックCMOSと同様にプリチャージなしの連続した演算を行なう．ASDMDL-2ϕの動作速度をより速くするために

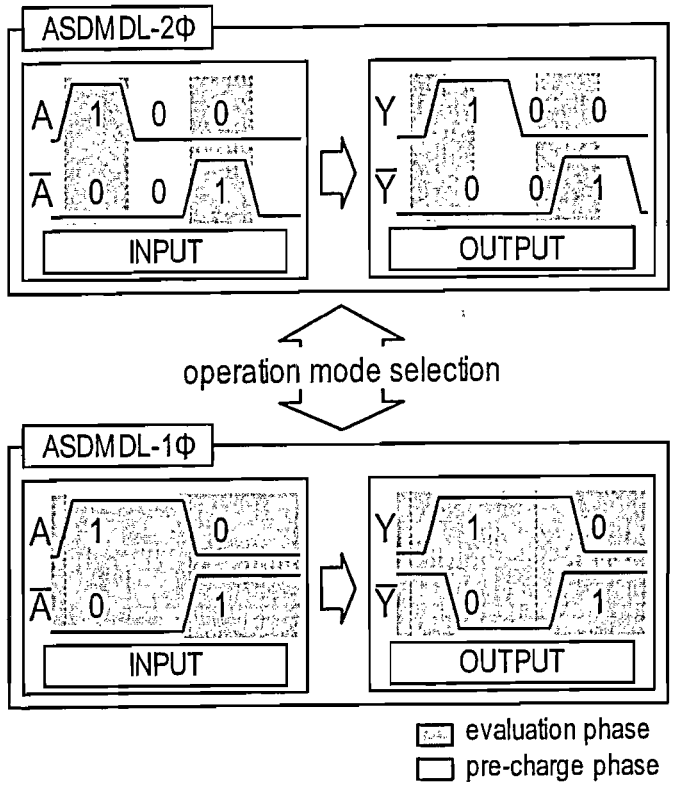

Figure 4.1: Signaling styles in ASDMDL-2ϕ and ASDMDL-1ϕ modes.

立ち上がりと立ち下がりの遷移時間が非対称に設計されていることから，ASDMDL-1ϕ 動作時の動作速度は低下する．しかしながら，プリチャージによる余分な信号遷移がなくなることで，消費電力は小さくなる．演算と演算の間にプリチャージを行なう ASDMDL-2ϕ 動作では，論理ゲートの出力結果が2回連続で論理0であったとしても，{0,1} → {0,0} → {0,1} と出力されるため，正論理，あるいは負論理の出力信号で必ず立ち上がりと立ち下がりの信号遷移が発生する．すなわち，正負両論理の2線信号を1つの信号線と考えると信号の遷移確率が 100%であると言える．それに対して，ASDMDL-1ϕ で動作させた場合には {0,1} → {0,1} となるために，信号線の余分な遷移が発生しない．このように，プリチャージがなくなることで余分な信号遷移がなくなり，ASDMDL-1ϕ 動作では消費電力を最小限に抑える事ができる．

ASDMDL-2ϕ 動作と ASDMDL-1ϕ 動作の2つの動作モードは論理ゲー

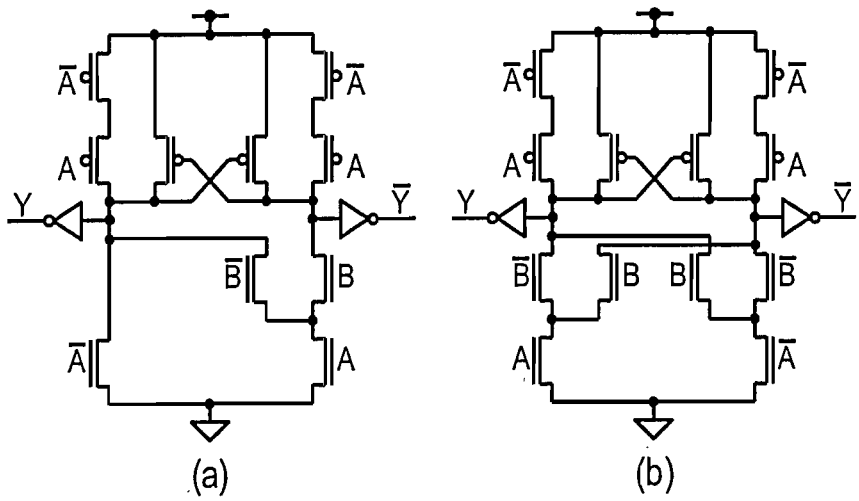

Figure 4.2: Circuit schematic of (a) 2-input NAND cell and (b) 2-input XOR cell.

トに入力される信号の状態によって切り替えられる．すなわち，演算の間にプリチャージを含んだ2相の信号が入力されてきた場合には高速モードで動作し，プリチャージ無しの1相で入力された場合は低消費電力モードで動作する．そのため，動作モードを切り替えるための専用の制御信号は必要せず，1つの回路で2つの動作を実現できる．

4.2.2 ASDMDLの回路構成と動作原理

本章では，ASDMDL論理ゲートの構成と2つの動作モードにおける動作原理について説明する．

Fig. 4.2(a) と (b) に ASDMDL の 2-input NAND と 2-input XOR の例をそれぞれ示す．ASDMDLの論理ゲートは，論理を生成するNMOSネットワーク，入力信号に {0,0} が到着したときにプリチャージを行なうための正負両論理 (例えば，{A,$\bar{A}$}) を入力に持つ直列接続のPMOSトランジスタ，クロスカップルトランジスタ，および出力インバータで構成されている．NMOSネットワークは，ASDDL/ASD-CMOSと同様にBDD表現を用いて設計する．

ASDMDL-2ϕ動作時，論理ゲートの入力信号に {0,0} の休止値が到着すると，NMOSネットワークはOFF状態，直列接続したPMOSトラン

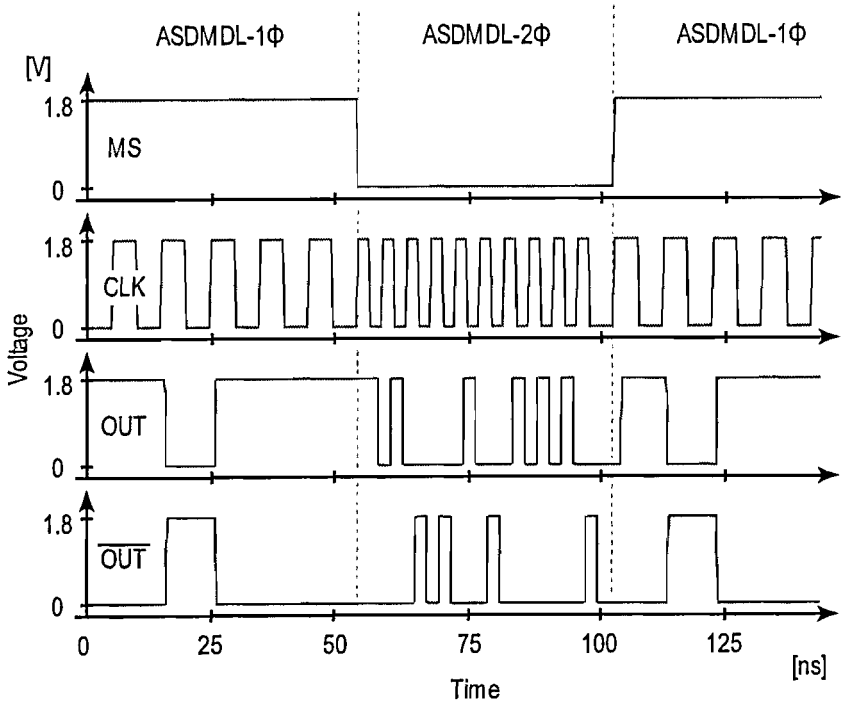

Figure 4.3: Operation waveform demonstrating change of signaling operation modes.

ジスタはON状態となるために休止値が出力され，回路はプリチャージされる．一方，有効値である{0,1}あるいは{1,0}が入力されると，NMOSネットワークにより一方の出力インバータの入力がGNDと接続され，演算結果が出力される．他方の出力インバータの入力では，プリチャージによって充電された電荷がクロスカップルトランジスタにより保持される．これより，論理ゲートから有効値が出力される．

ASDMDL-1ϕ動作はプリチャージなしの動作を行なう．有効値が入力されると，ASDMDL-2ϕ動作と同様にNMOSネットワークにより一方の出力インバータの出力信号から演算結果が出力される．他方の出力インバータはクロスカップルトランジスタにより演算結果の反転論理を出力する．

このように，動作モードを切り替えるための専用の制御信号は必要とせず，ASDMDL-2ϕとASDMDL-1ϕは論理ゲートに入力される信号の状態によって切り替えられる．そのため，Fig. 4.3に示したように動作中の

各タイミングに合わせてモードを切り替えることにより，入力パターンや処理量に応じた回路性能を引き出すことができる．

また，ASDMDLは複雑な論理関数をBDDで表現することによって，スタティックCMOSでは2，3個の論理ゲートを用いて表現する論理関数を1つの論理ゲートで構成することができる．これは，ASDMDL-1ϕ動作でも有効であり，スタティックCMOSに比べてASDMDLの消費電力の低減が見込める．

4.2.3 ASDMDLの回路性能評価

Fig. 4.2に示した2入力NANDと2入力XORを0.18-μm CMOSプロセスで設計し，スタティックCMOSとの比較評価を行なった．Fig. 4.4(a)と(b)に，ASDMDL-2ϕで動作させたときのASDMDL論理ゲートとスタティックCMOSの速度・電力グラフを示す．Fig. 4.4に示したグラフはトランジスタレベルのネットリストをSPICEにてシミュレーションした結果であり，電源電圧1.8Vにおいて遅延時間が最も高速になるようにトランジスタのWを大きくしたときの遅延時間と消費電力の関係を示している．

ASDMDL-2ϕ動作におけるASDMDL論理ゲートの遅延時間は，スタティックCMOSでは到達不可能な高速動作を実現し，同じ消費電力における2入力NANDと2入力XORの遅延時間改善率はスタティックCMOSと比較して18%，55%であった．ここで，2入力NAND論理ゲートの改善率は2入力XOR論理ゲートの改善率に比べて小さい．スタティックCMOSでは2入力NANDに比べて2入力XORの方が論理ゲートが複雑になる．一方，ASDMDLは正負両論理の2線の信号線で回路を構成する2線式論理回路であり，2入力NAND，2入力XOR共にBDD表現を用いてNMOSネットワークの構成だけを変更すればよい．Fig. 4.2にも示されているように2入力NANDと2入力XORはNMOSネットワーク内の論理を生成するトランジスタの段数は同じであるため，2入力XORの方がスタティックCMOSに比べてより改善率が高くなっている．

また，Fig. 4.4(b)の遅延・電力グラフにASDMDL-1ϕ動作時の性能とASDDLの性能も示している．ASDMDL-1ϕでは，ASDMDL-2ϕに比べて遅延時間が低下しており，スタティックCMOSとほぼ同等の性能であることが分かる．また，ASDMDL-2ϕとASDDLの性能もほとんど差はなく，ASDDLと同じ高速動作を実現している．これは，ASDMDLはASDMDL-2ϕ動作とASDMDL-1ϕ動作を切り替えることによって，高速

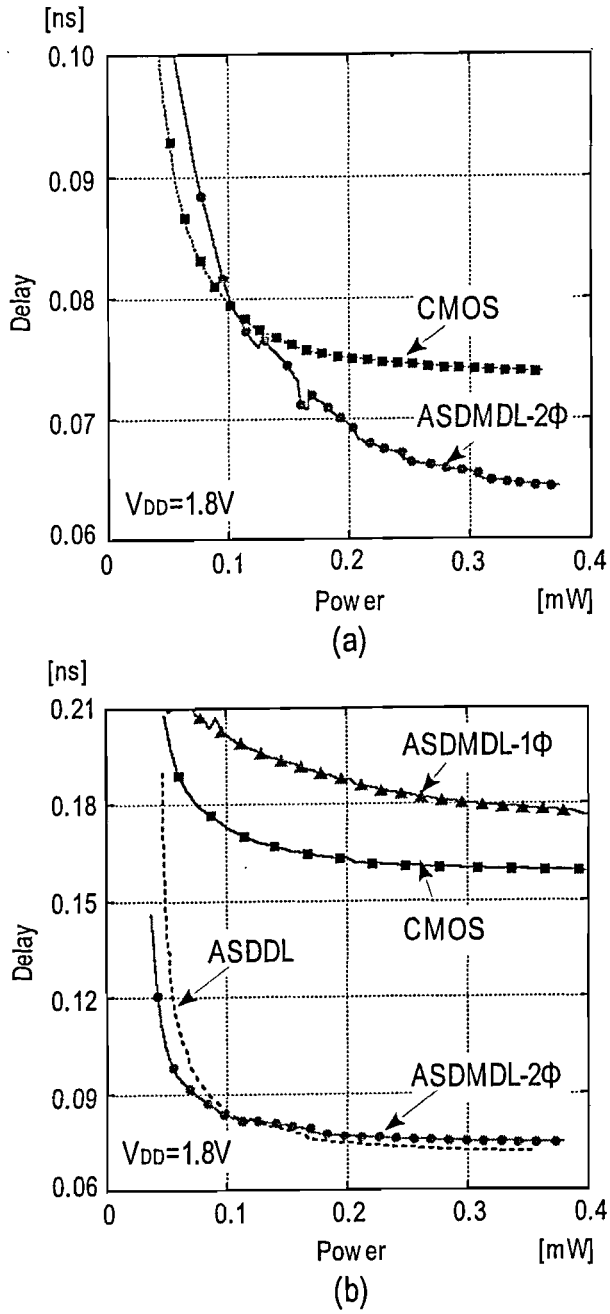

Figure 4.4: Delay time vs. power consumption simulated on (a) 2-input NAND gate and (b) 2-input EXOR gate.

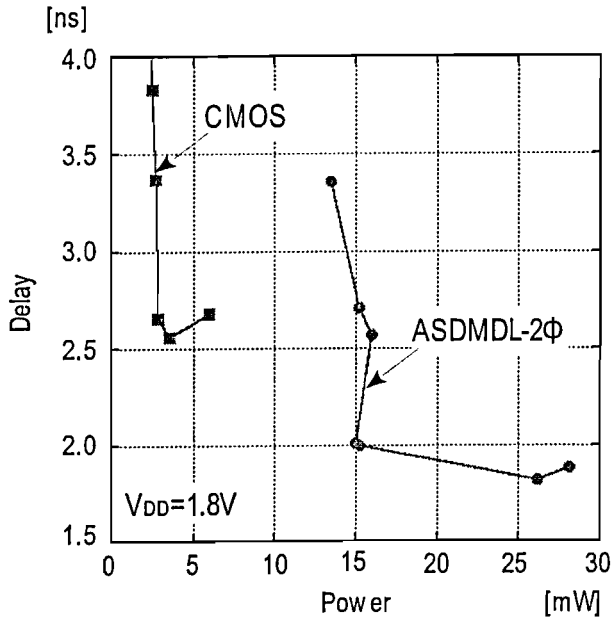

Figure 4.5: Delay time vs. power consumption simulated on 16-bit multiplier.

ダイナミック回路である ASDDL とスタティック CMOS 並の低消費電力性能を使い分けることができることを意味する．すなわち，ASDMDL は2つの異なった回路性能を1つの回路で実現できる回路方式であるといえる．

組み合あせ回路の比較評価として，上記で比較評価したのと同じ 0.18-μm CMOS プロセス，電源電圧 1.8V を用いて設計した 16 ビット符号付き乗算器の比較結果を Fig. 4.5 に示している．設計した乗算器は様々な異なった制約条件で論理合成を行ない，これらを自動配置配線によりレイアウトした後，RC 抽出したものを SPICE シミュレーションにより評価した．Fig. 4.5 より ASDMDL-2ϕ の遅延時間はスタティック CMOS に比べて 38% の改善が見られ，組合せ回路においてもその速度の優位性を証明できた．しかしながら，ASDMDL-2ϕ 動作は全てのクロックサイクルで立ち上がりと立ち下がりの2つの信号遷移が発生するため，ASDMDL-2ϕ の消費電力はスタティック CMOS と比べて大きく増加している．ここで，低消費電力のスタティック CMOS 動作と高速な ASDMDL 動作を要求する性能に応じて切り替えることができるならば，回路のエネルギー効率

を改善することができ，設計した回路の最適な性能を得ることができる．

4.3 大規模回路への適用と回路検証

本節では，高速モードと低消費電力モードの2つの動作モードを有するASDMDLの大規模回路への適用とその回路の性能評価について述べる．

近年の大規模回路では，高スループットを実現するためにほとんどの回路にパイプライン構造が採用されている．そのため，ASDMDLを大規模回路に適用するにあたっては，このパイプライン構造をASDMDLでどのように構成するかが重要となる．そこで，ASDMDLにおけるパイプライン構造について説明し，その構造におけるASDMDL-2ϕとASDMDL-1ϕの動作について述べる．

また，大規模回路をASDMDLで設計するにあたって，3章で提案した論理合成手法は2線式論理回路であるASDMDLにも転用が可能であることから，回路全てをASDMDLで構成することは可能である．しかしながら，ASDMDLはスタティックCMOSと比べてトランジスタ数も多く，ASDMDL-2ϕ動作時には消費電力も増加するため，設計した回路全てにASDMDLを適用すると面積や消費電力が著しく増大する．設計した回路性能をできる限り向上させるためには，高速動作を必要とする回路部分にのみASDMDLを適用し，その他は通常のスタティックCMOSで設計するのが適切である．そこで，ASDMDLとスタティックCMOSの混在回路設計に用いた自動設計環境について述べる．さらに，実際に設計したASDMDL/CMOS混在プロセッサの回路構成と性能検証について示す．

4.3.1 ASDMDLのパイプライン構成

ASDMDL-2ϕ動作は，ASDDL/ASD-CMOSと同様に演算の伝搬時間(立ち上がり遷移時間)に対してプリチャージの伝搬時間(立ち下がり遷移時間)が遅く，演算の前には必ず回路をプリチャージする必要があるため，サイクルタイムが長くなる．そこで，2.3節で説明したASDDL/ASD-CMOSのサイクルタイム短縮アーキテクチャを用いる．本節では，サイクルタイム短縮アーキテクチャを用いるために必要となるASDMDL用のRREGcellとRcellについて述べ，このアーキテクチャを利用したパイプライン構成について説明する．

ASDMDL用のRcellとRREGcellの回路構成をFig. 4.6(a)と(b)にそれぞれ示す．Fig. 4.6に示されているようにASDMDL-2ϕ動作とASDMDL-

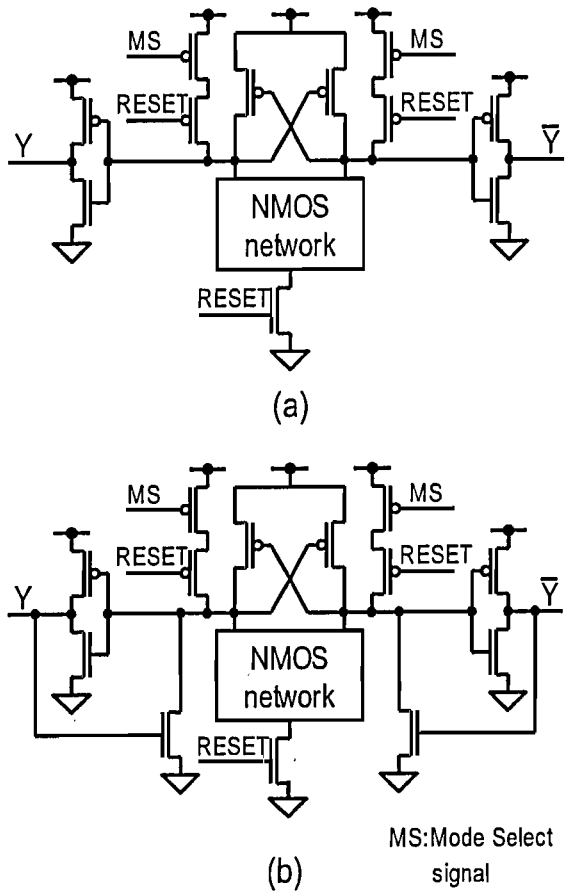

Figure 4.6: Circuit structures of (a) ASDMDL RREGcell, and (b) ASDMDL Rcell.

1ϕ動作を切り替えて動作できるように，Rcell と RREGcell には動作モードを切り替える信号 (MS:Mode Select signal) が分配され，プリチャージの有無を制御する．なお，動作モード切り替え信号は Rcell，RREGcell のみに分配され，通常の論理セルに分配する必要はない．

Rcell は，NMOS ネットワークと GND の間に RESET 信号を入力に持つ NMOS トランジスタを加え，直列接続の PMOS トランジスタの入力を動作モード切り替え信号と RESET 信号とした構成となっている．動作モード切り替え信号が “0” のとき，Rcell は ASDMDL-2ϕ で動作する．RESET 信号が “1” のときは，通常の論理ゲートと同様に NMOS ネットワークの論理関数に従った演算結果を出力する．RESET 信号が “0” にな

ると，NMOSネットワークとGNDの間に加えられたNMOSトランジスタがOFFし，出力インバータの入力がGNDから切り離される．しかしながら，動作モード切り替え信号とRESET信号が直列接続されたPMOSトランジスタによって，出力インバータの入力はVDDに固定されるため，Rcellの出力信号は{0,0}の休止値を出力する．一方，動作モード切り替え信号が“1”になるとASDMDL-1ϕで動作し，RESET信号が“1”ではASDMDL-2ϕと同様に演算結果を出力する．RESET信号が“0”に変化すると，出力インバータの入力がGNDから切り離され，動作モード切り替え信号を入力に持つPMOSトランジスタもOFF状態となっているため，出力インバータの入力は浮いた状態となる．しかしながら，RESET信号が“1”のときに蓄えられた電荷によって，演算結果はそのまま維持される．

RREGcellはRcellに出力信号からの帰還ループを付け加えた構成となっている．動作モード切り替え信号が“0”のときにはASDMDL-2ϕで動作し，Rcellと同様に動作する．しかしながら，演算結果である有効値を出力すると，後段から休止値が到着したとしてもその演算結果を保持し続ける．保持された演算結果は休止値の出力によってリセットされる．一方，動作モード切り替え信号が“1”になるとASDMDL-1ϕで動作し，RESET信号が“1”ではRREGcellが演算結果を出力する．この演算結果はRESET信号が“0”となっても，付加されたNMOSトランジスタによってNMOSネットワークとGNDが切断されるため，演算結果は変化することなく，保持される．

大規模回路に適用する際のパイプライン構造はサイクルタイム短縮アーキテクチャと同様に1パイプラインステージを2つに分割した構造にする．ASDMDLのパイプライン構造をFig. 4.7に示す．それぞれのパイプラインステージはRREGcellで入力側と出力側の2つに分割する．さらに，分割されたそれぞれの回路ブロックにはRcellが配置され，プリチャージよりも演算時間が短くなるようにする．また，配置されたRREGcellとRcellのRESET信号にはクロック信号を分配する．ASDMDL-2ϕ動作時にそれぞれのブロック(sub-block 1, sub-block 2)は演算と回路のプリチャージを半サイクル毎に交互に実行する．一方，ASDMDL-1ϕ動作時には各ブロックが半サイクル毎に演算を行なう．入力側と出力側にあるRREGcell(RREG AとRREG B)は，スタティックCMOSにおけるパイプライン構造のパイプラインレジスタの位置に相当する．また，RREGcellはクロック信号の“0”のときに演算結果を後段に伝搬するLatch機能を

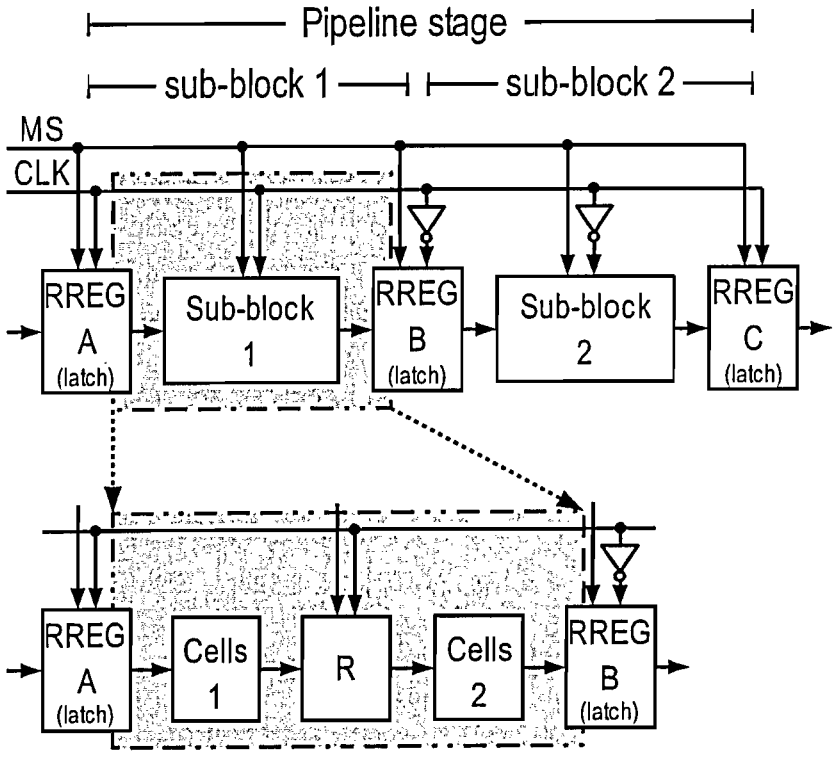

Figure 4.7: Pipeline structure of ASDMDL logic circuit.

もった回路であることから，ASDDL/ASD-CMOSにおけるパイプラインはクロック信号の両エッジで動作を切り替え，各回路ブロックが交互に演算を行なう構造となる．

Fig. 4.8に各回路ブロックの動作波形を示す．ASDMDL-2ϕ動作モードでは，初期状態としてsub-block 1はプリチャージされている．クロック信号が“1”になると順次演算が行なわれ，クロック信号が“0”になると，RREGcell とRcellから出力された{0,0}の伝搬によってsub-block 1はプリチャージされる(Fig. 4.8(a))．sub-block 2はクロック信号が反転して入力されるため，sub-block 1とは逆相で動作する．そのため，パイプラインステージは1サイクルで全ての演算とプリチャージを行ない，すべてのプリチャージを演算の裏側に隠すことができる．

ASDMDL-1ϕ動作時にはクロック信号が“1”になると，ASDMDL-2ϕ動作時と同様にsub-block 1は演算を行なう(Fig. 4.8(b))．しかしながら，クロック信号が“0”になってもsub-block 1はプリチャージを行なわ

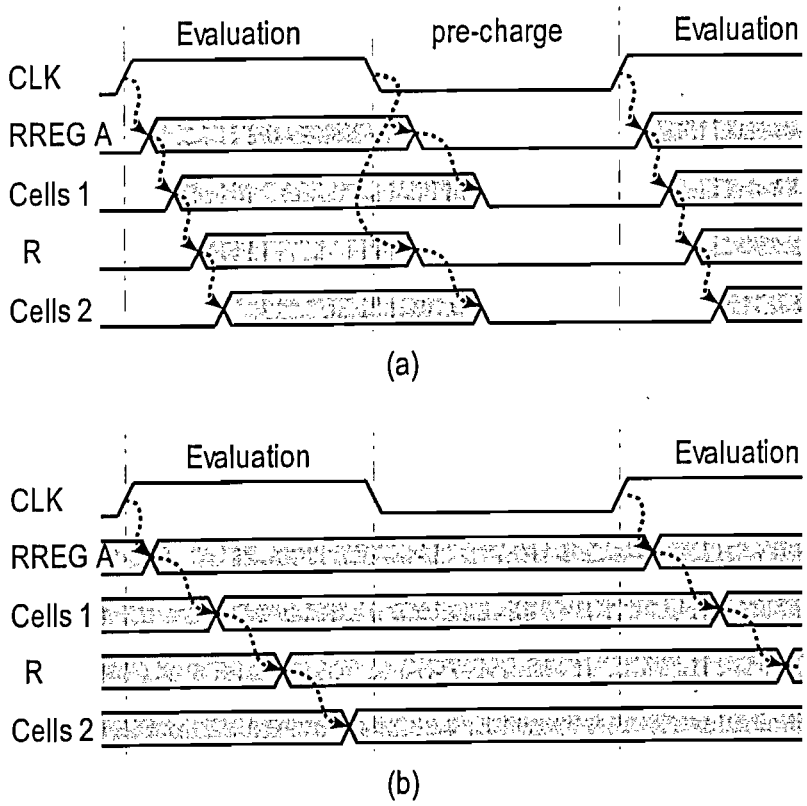

Figure 4.8: Operation diagrams of (a) ASDMDL-2ϕ operation and (b) ASDMDL-1ϕ operation.

ず，動作は停止している．一方，sub-block 2はRREG Bまで到着していた演算結果が入力され，演算を行なう．これにより，パイプライン構造の回路でもRREGcellとRcellに接続された信号1つでASDMDL-2ϕとASDMDL-1ϕを切り替えることができ，ASDMDLはパイプライン構造の回路にも適用可能である．

4.3.2 スタティックCMOSとの混載設計のための自動合成

ASDMDLはASDMDL-2ϕで回路を動作させることによりスタティックCMOSでは到達不可能な高速動作を実現するため，クリティカルパスに適用することで，設計した回路の動作周波数を向上させることができる．一方，クリティカルパスと比べて非常に短いパスにASDMDLを適用して

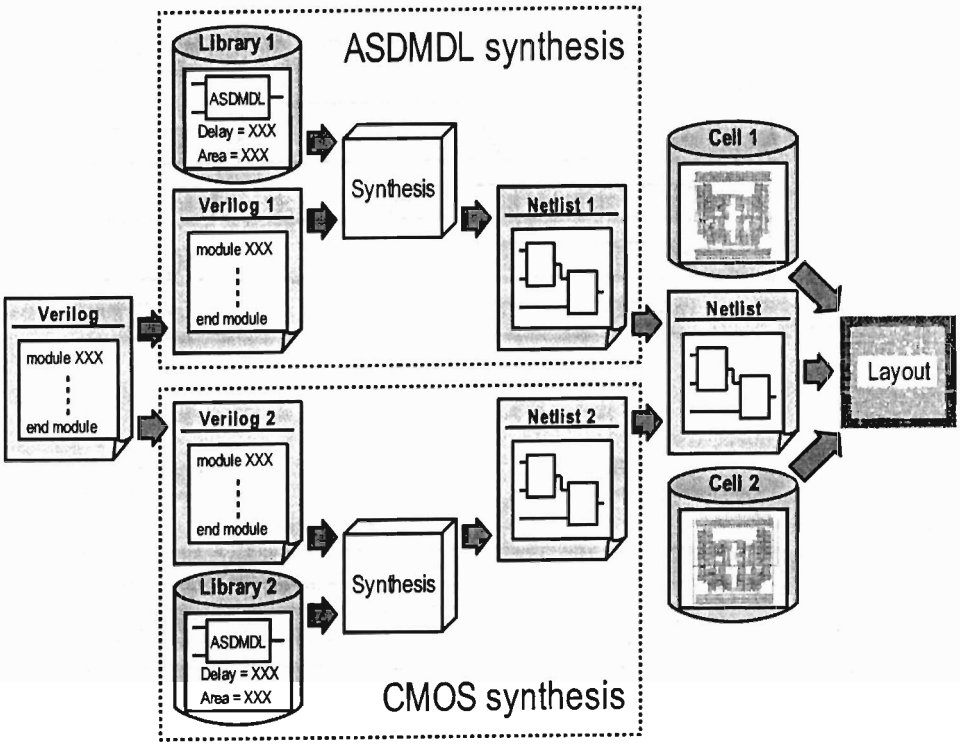

Figure 4.9: Synthesis flow of mixed ASDMDL/CMOS circuit.

も，動作周波数の改善効果は得られない．これは，設計した回路の動作周波数は最も遅延時間の長いクリティカルパスによって制限されるためである．それだけではなく，ASDMDL-2ϕ 動作時は ASDDL/ASD-CMOS と同様に消費電力はスタティック CMOS に比べて大きいため，設計した回路の消費電力が増加する．すなわち，動作周波数に関係のない短いパスに ASDMDL を適用したとしても，動作周波数は改善されず，消費電力が増加するだけとなる．そこで，ASDMDL は高速動作を必要とする回路部分だけに適用し，その他は通常のスタティック CMOS で設計することで，無駄な性能劣化を抑え，ASDMDL の利点を最大限に生かした回路を設計することができる．本節では，ASDMDL とスタティック CMOS を混在した大規模回路の自動設計について述べる．

ASDMDL/CMOS 混載回路の自動設計を行なう際は，どの回路モジュールに ASDMDL を適用するのかを設計者が判断し，論理記述された Verilog ファイルをスタティック CMOS と ASDMDL を適用する回路モジュール

に分割する．分割された回路モジュールはそれぞれを適用する回路方式で論理合成が行われる (Fig. 4.9)．ASDMDL の論理合成には 3 章で提案した論理合成手法を用いて行い，論理ゲート遅延平均化のパイプライン合成に基づいたサイクルタイム短縮アーキテクチャを適用した回路を合成する．合成された回路には独自に作成したスクリプトにより 1 線から 2 線，2 線から 1 線に変換するインターフェース回路が挿入され，スタティック CMOS で合成された回路に組み込むことで ASDMDL とスタティック CMOS が混載したネットリストを生成することができる．このネットリストとそれぞれの論理ゲートレイアウトを用いて，全体レイアウトを自動配置配線で作成する．ASDMDL 論理ゲートのレイアウトをスタティック CMOS のレイアウトと同じ高さにしておくことで，ASDMDL とスタティック CMOS を個別に扱うことなく，混載した自動配置配線を実現できる．

自動設計用の合成ライブラリは，ASDDL 用に構築した Table 3.1 に示した 3 入力以下の全ての論理を表現できる 12 種類の ASDMDL 論理ゲートで構成し，これらに対していくつかの駆動力のものを用意した．ASDMDL は ASDDL と同様に入力や出力信号の 2 線を入れ換えることで，複数の論理を表現できるため，少ない論理ゲート数でライブラリを構成できる．

4.3.3 テストチップによる動作検証と性能評価

本節では，0.18-μm CMOS プロセス，電源電圧 1.8V にて試作した SH3 アーキテクチャに基づいた合成可能な IP コアのテストチップの回路構成と実測評価について示す．

試作したテストチップには，ASDMDL/CMOS 混載構成と全スタティック CMOS 構成の 2 つのプロセッサ IP，及びテスト用回路で構成されている．Fig. 4.10 に試作したテストチップのレイアウトを示す．

全スタティック CMOS 構成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ライブラリを用い，動作周波数が最適になるように回路を設計した．一方，ASDMDL/CMOS 混載構成はクリティカルパスを含んだ回路モジュールのみに ASDMDL を適用し，その他はスタティック CMOS で構成している．ASDMDL の適用率は 4%である．Fig. 4.9 に示した自動設計を用いることにより，ASDMDL/CMOS 混載構成は全スタティック CMOS 構成と同様に論理合成から自動配置配線まで全てのフローを自動で行なっており，全スタティック CMOS 構成と同等のコストで設計できる．回路面積は ASDMDL/CMOS 混載構成と全スタティック CMOS

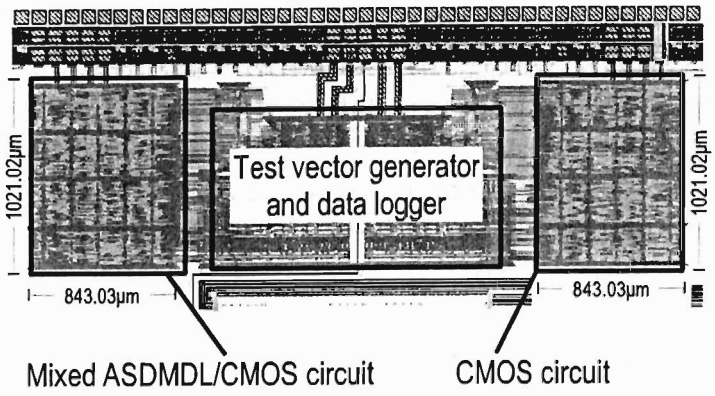

Figure 4.10: Test chip layout.

構成共に 843.03μm × 1021.02μm とした．回路規模は ASDMDL/CMOS 混載構成と全スタティック CMOS 構成共に約 25 万トランジスタであり，それぞれ回路のセル占有率は 89.22%，85.66%である．

テスト用回路は全スタティック CMOS 構成と ASDMDL/CMOS 混載構成のプロセッサを動作させるための命令とデータを格納するメモリに相当するレジスタアレイ，入力信号を制御するための制御回路，および出力信号の状態をレジスタに格納し，チップ外部に出力する回路で構成されている．

Fig. 4.11 にテストチップの実測による ASDMDL-2ϕ，ASDMDL-1ϕ，スタティック CMOS それぞれの Shmoo plot を示す．電源電圧 1.8V における ASDMDL-2ϕ の最高動作周波数は 232MHz であり，これは最も高速に設計された全スタティック CMOS 構成と比べて 14%向上した．なお，最も遅延時間の長いクリティカルパスを含んだ回路モジュール 1 つだけに ASDMDL を適用したことで，クリティカルパスが他のパスに変更されてしまったため，動作周波数の改善率は 14%にとどまったと考えられる．一方，ASDMDL-1ϕ の最高動作周波数は 208MHz であり，スタティック CMOS よりも 2%改善した．

しかしながら，ASDMDL-2ϕ は全てのクロックサイクルで立ち上がりと立ち下がりの 2 つの信号遷移が発生するため，ASDMDL-2ϕ の消費電力はスタティック CMOS と比べて増加する．Fig. 4.12 の動作周波数・電力グラフにも示したように，動作周波数 232MHz，電源電圧 1.8V における ASDMDL-2ϕ の消費電力は 81.18mW であった．

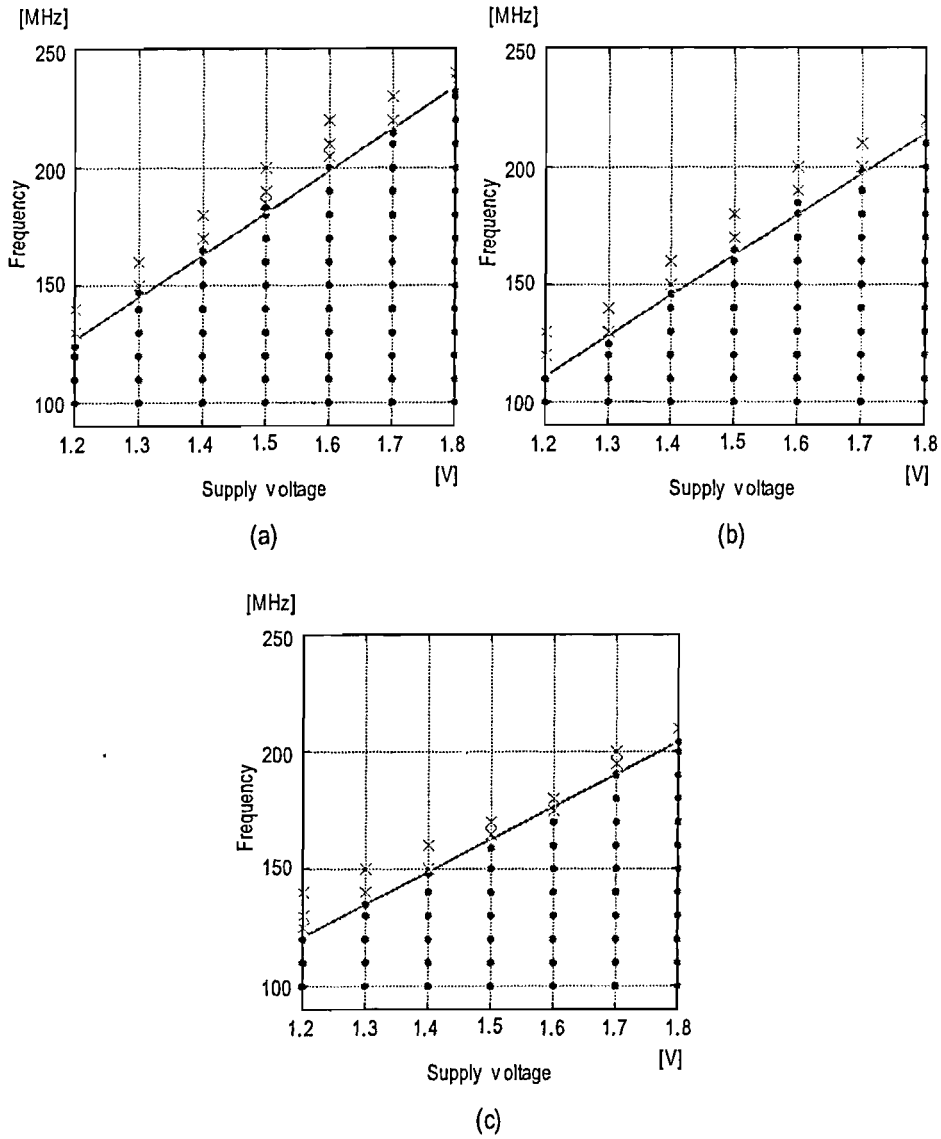

Figure 4.11: Measured Shmoo plots of (a) ASDMDL-2ϕ, (b) ASDMDL-1ϕ, and (c) CMOS.

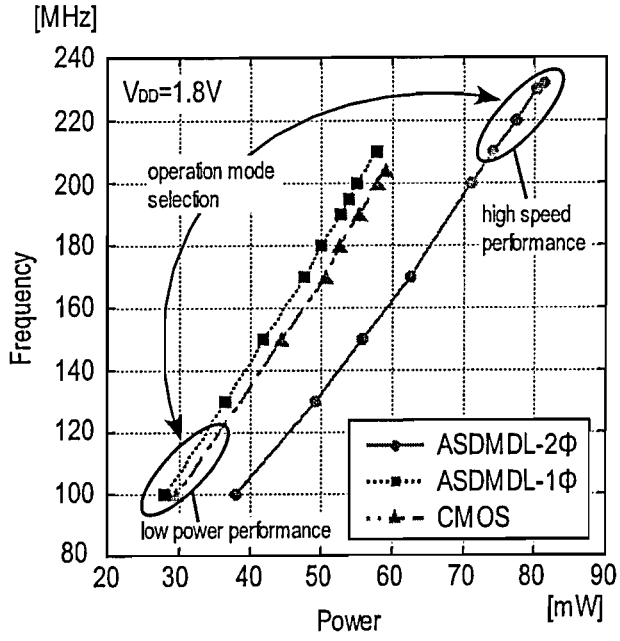

Figure 4.12: Measured operating frequency vs. power consumption.

一方，4.2.2節でも述べたようにスタティック CMOS では実現が困難となる複雑な論理を持った論理ゲートが多く使用されたため，ASDMDL-1ϕの消費電力はスタティック CMOS よりも3%改善しており，スタティック CMOS と類似した回路性能を実現している．

このように，ASDMDL は2相動作である ASDMDL-2ϕ と単相動作である ASDMDL-1ϕ を切り替えることで，ダイナミック回路の高速動作とスタティック CMOS の低消費電力動作という特性を1つの回路で実現でき，動作タイミングに合った動作を選択することで，最適な回路性能を得ることができる．

4.4 むすび

近年，モバイル機器に対する需要が高まる中，動作速度の向上させる技術に加えて，消費電力を抑えて長時間のバッテリ寿命を実現することが不可欠となっている．そこで，演算実行中に負荷の大きさに応じて電源電圧と動作周波数を動的に制御するプロセッサが多く存在する．本章

では，通常は高速動作を実現するASDMDL-2ϕで動作し，処理量が少ない場合には消費電力を引き下げることのできるASDMDL-1ϕで動作するASDMDL回路方式を提案した．ASDMDL-2ϕは演算の前にプリチャージを行なう2相で動作し，論理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することで高速動作を実現する．一方，ASDMDL-1ϕはプリチャージなしの1相で動作し，プリチャージによる余分な信号遷移をなくすことで消費電力を下げる．この2つの動作モードは論理ゲートに入力される信号の状態によって切り替えられ，通常の論理ゲートには特別な制御信号を必要としない．

0.18-μm CMOSプロセスで設計した2入力NANDと2入力XORの性能評価を行った．ASDMDL-2ϕ動作における2入力NANDと2入力XORの遅延時間はスタティックCMOSと比較して18%，55%であり，ASDDL/ASD-CMOSと同様にスタティックCMOSでは到達不可能な高速動作を実現した．一方，ASDMDL-1ϕ動作はスタティックCMOSとほぼ同等の性能であった．これにより，ASDMDLは高速ダイナミック回路とスタティックCMOSの性能を1つの回路で実現できる回路方式であることを示した．さらに，組合せ回路の性能評価として，0.18-μm CMOSプロセス，電源電圧1.8Vで設計した16ビット符号付き乗算器のASDMDL-2ϕにおける遅延時間は，スタティックCMOSに比べて38%改善した．

ASDMDLを実際に大規模回路に適用するに当たって，ASDDL/ASD-CMOSで提案したサイクルタイム短縮アーキテクチャを利用したパイプライン構成を考案した．また，2線2相式論理回路ASDDL/ASD-CMOSの論理合成手法を用いることにより，ASDMDLとスタティックCMOSを混在した大規模回路の論理合成を実現し，自動設計用ライブラリを構築した．

2つの動作モードを有するASDMDL回路を用いて，SH3アーキテクチャに基づいた合成可能なIPコアのテストチップを0.18-μm CMOSプロセスにて試作した．ASDMDL-2ϕの最高動作周波数は232MHzであり，スタティックCMOSと比べて14%向上した．また，ASDMDL-1ϕの性能はCMOSと同等の性能を実現し，ダイナミック回路の高速動作とスタティックCMOSの低消費電力動作という特性を1つの回路で実現できることを実証した．これより，動作条件に合った動作を選択することで最適な回路性能を得ることができ，大規模回路の性能を飛躍的に向上させることが可能となる．

第5章

結論

LSIの高集積化により，高度で多彩な機能を持った回路が実現できるようになってきている．しかしながら，回路内で処理しなければならないデータ量は日々増加する傾向にあり，高速回路設計技術は必要不可欠なものとなっている．本研究では，全ての論理を立ち上がり遷移で表現し，論理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することで，従来から提案されている高速回路方式を上回る動作速度を実現する2線式論理回路の設計手法を提案した．

第1章では，研究の概要として，アーキテクチャレベル，および回路設計レベルの高速化技術について説明し，その問題点について述べた．第2章では，従来提案されている高速回路方式よりも高速低消費電力動作を実現するために，非対称な信号遷移を用いたASDDL (Asymmetric Slope Differential Dynamic Logic) とASD-CMOS (Asymmetric Slope Differential CMOS) を提案した．第3章では，大規模回路設計に対してASDDL/ASD-CMOSを適用することを目的に，2線2相式論理回路の自動設計システムについて説明した．第4章では，高速モードと低消費電力モードを有するASDMDL (Asymmetric Slope Dual Mode Differential Logic) について述べた．各章の研究内容について，以下に結論を述べる．

第2章では，従来の高速回路方式よりも高速動作と低消費電力を実現するASDDLとASD-CMOSを提案した．ASDDL/ASD-CMOSは正論理と負論理の信号で論理値を表現する2線式論理回路であり，論理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することで，スタティックCMOSに比べて格段に短い遅延時間を実現する．しかしながら，ASDDL/ASD-CMOSは演算とプリチャージを交互に実行する必要があるために，サイクルタイムが長くなる．そこで，演算の裏にプリチャージを隠し，サイクルタイムを演算時間とほぼ等しくするサイクルタイム短縮アーキテクチャを考案した．0.18-μm CMOSプロセス，電源電圧1.8Vにて，スタティックCMOSとDCVS-DOMINOとの比較評価を行った．ASDDLとASD-CMOSで設計した乗算器の遅延時間は，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ラ

イブラリを用いて，遅延時間最小の条件で設計したスタティック CMOS の 68%，66%であった．DCVS-DOMINO との比較では，ASDDL/ASD-CMOS はプリチャージのための制御信号を必要としないことから，回路構成において遅延時間，消費電力が改善されることを示した．SPICE シミュレーションでは，ASDDL と ASD-CMOS は DCVS-DOMINO よりも高速動作を実現できることを証明し，消費電力はそれぞれ 20%，2% 削減した．また，0.13-μm CMOS プロセス，電源電圧 1.2V で試作したテストチップでは，ASD-CMOS の遅延時間は 1.57ns であることを確認した．

第 3 章では，全ての信号線が正負両論理で表現した 2 線式論理回路である ASDDL/ASD-CMOS の論理合成手法を提案し，スタティック 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した．スタティック 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/ASD-CMOS の自動設計を行なった．また，変換ツールによりサイクルタイム短縮アーキテクチャの自動適用を実現した．提案した論理合成手法を用いて回路を設計するために，ASDDL の中間ライブラリと配置配線用ライブラリを構築した．構築したそれらのライブラリは 3 入力以下の全ての論理関数を表現できる 12 種類の論理ゲートで構成した．構築した自動設計環境を用いて，様々な制約条件を持った 16 ビット符号付き乗算器を 0.18-μm CMOS プロセスで設計した．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 ASDDL16 ビット乗算器の遅延時間は 1.82ns であり，フルカスタム設計に近い回路性能を実現した．また，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を実現した，また．サイクルタイム短縮アーキテクチャの自動適用による回路性能の増加を 10%未満に抑えた．

第 4 章では，通常は ASDDL/ASD-CMOS と同様に高速で動作し，処理量が少ない場合には消費電力を引き下げることのできる ASDMDL (Asymmetric Slope Dual Mode Differential Logic) を提案した．ASDMDL は，演算の前にプリチャージを行なう 2 相で動作する高速モードとプリチャージなしの 1 相で動作する低消費電力モードの 2 つの動作を制御信号なしで切り替える．大規模回路設計のために，ASDMDL 用のパイプライン構成を考案した．また，ASDMDL とスタティック CMOS を混在した大規模回路の論理合成を実現し，自動設計用ライブラリを構築した．0.18-μm CMOS プロセスで設計した 2 入力 NAND と 2 入力 XOR の性能評価を行っ

た．高速モード動作における2入力NANDと2入力XORの遅延時間はスタティックCMOSと比較して18%，55%であり，ASDDL/ASD-CMOSと同様にスタティックCMOSでは到達不可能な高速動作を実現した．組合せ回路の性能評価として設計した16ビット符号付き乗算器の遅延時間は，スタティックCMOSに比べて38%改善した．また，SH3アーキテクチャに基づいた合成可能なIPコアのテストチップを0.18-μm CMOSプロセスにて試作した．高速モード時の最高動作周波数は232MHzであり，スタティックCMOSと比べて14%向上し，低消費電力モードの性能はCMOSと同等の性能を実現した．ダイナミック回路の高速動作とスタティックCMOSの低消費電力動作という特性を1つの回路で実現できることを実証した．

以上の研究成果より，ASDDL/ASD-CMOSはスタティックCMOSでは到達不可能な高速動作が実現でき，従来提案されている高速回路方式よりも高速動作・低消費電力を達成した．また，提案した論理合成手法により，様々な制約条件，アーキテクチャを持った回路を短期間で設計することができ，プロセッサの制御回路などのようなランダムロジックの設計が可能となった．さらに，ASDMDLの2つの動作モードを切り替えて動作させることで，回路の動作タイミングに合った最適な回路性能を得ることができ，デジタルLSIの性能を飛躍的に向上させることが期待できる．

謝辞

本研究の機会を与えていただき，研究のご指導を賜りました神戸大学工学部情報知能工学科・永田 真 助教授に深く感謝致します．本研究の遂行にあたり，終始にわたって懇切なる御指導，御鞭撻を賜りました．心より感謝いたします．

また，研究室配属当初の指導教員であられました瀧 和男 教授（現 エイ・アイ・エル株式会社代表取締役社長）には，広い見識と鋭い発想で熱心に御指導ならびに御鞭撻，本研究に関して惜しみない御協力，御尽力を頂きました．また，本学退職後も助言や励ましのお言葉を掛けてくださいました．心から感謝の意を表します．

本論文をまとめるにあたり貴重な御助言，御指導をいただきました情報知能工学科・吉本 雅彦 教授，電気電子工学科・沼 昌宏 教授，ならびに情報知能工学科・羅 志偉 教授に深く感謝いたします．

研究生活に関して何かと御世話になり，また研究に限らず様々な視野から日々御議論戴きました同学科・鎌田 十三郎 助手に深く感謝の意を表します．

同じ研究グループとして様々な御協力をして頂きましたCS26講座の卒業生である田中 義則 氏に御礼申し上げます．本研究の自動設計環境を構築する際には，様々なアイデアを提供して頂き，環境構築のための作業を手伝ってくださいました．心より感謝いたします．

また，同じグループのメンバーとして，あるいは後輩として様々な御協力，及び無理難題を聞いて戴きましたCS26 LSIグループの諸氏に心から感謝致します．特に，同じ博士後期課程の学生として，日々の議論を通じて研究に多大なる刺激を与えて頂きました福水 洋平 氏，野口 宏一朗 氏，小坂 大輔 氏，深澤光弥 氏，松野 哲郎 氏に感謝いたします．

研究室での日常生活においてお世話になりましたCS26の皆様に感謝申し上げます．特に，無償奉仕であるにも拘らず計算機環境の構築ならびに維持に日々奮闘戴いたCS26計算機管理者集団の諸氏にはあらためて感謝の意を表します．

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，提供によるCadenceツールおよびSynopsysツールを用いて設計が行われたものである．また，本回路方式の性能評価のためにSH3-DSPコアの設

計データを提供して下さいました株式会社ルネサステクノロジに感謝致します.

最後に私をここまで育てて下さいました両親，ならびに温かく見守ってくれた兄と姉に心より感謝します.

参考文献

[1] R.H. Krambeck, C.M. Lee, and H.S. Law, "High-speed compact circuits with CMOS," *IEEE Journal of Solid-State Circuits*, Vol. SC–17, No. 3, pp. 614–619, 1982.

[2] L.G. Heller, W.R. Griffin, J.W. Davis, and N.G. Thoma, "Cascode voltage switch logic: A differential CMOS logic family," *in IEEE International Solid-State Circuits Conference*, pp. 16–17, 1984.

[3] N.F. Goncalves and H.J.D. Man, "NORA: A racefree dynamic CMOS techuique for pipelined logic structure," *IEEE Journal of Solid-State Circuits*, Vol. SC–18, No. 3, pp. 261–266, 1983.

[4] A. Solomatnikov, D. Somasekhar, K. Roy, and C.K. Koh, "Skewed CMOS: Noise-immune high-performance low-power static circuit family," *Proceedings of IEEE International Conference on Computer Design*, pp. 424–427, 2000.

[5] L.C.M.G. Pfennings, W.G.J. Mol, J.J.J. Bastiaens, and J.M.F.V. Dijk, "Differential split-level CMOS logic for subnanosecond speeds," *in IEEE International Solid-State Circuits Conference*, pp. 212–213, 1985.

[6] L.C.M.G. Pfennings, W.G.J. Mol, J.J.J. Bastiaens, and J.M.F.V. Dijk, "Differential split-level CMOS logic for subnanosecond speeds," *IEEE Journal of Solid-State Circuits*, Vol. SC–20, No. 5, pp. 1050–1055, 1985.

[7] T.A. Grotjohn and B. Hoefflinger, "Sample-set differential logic (SSDL) for complex high-speed VLSI," *IEEE Journal of Solid-State Circuits*, Vol. SC–21, No. 2, pp. 367–369, 1986.

[8] J.A. Pretorius, A.S. Shubat, and C.A.T. Salama, "Latched domino CMOS logic," *IEEE Journal of Solid-State Circuits*, Vol. SC–21, No. 4, pp. 514–522, 1986.

[9] A. Naini, D. Bearden, and W. Anderson, "A 4.5 ns 96b CMOS adder design," *Proceedings of IEEE Custom Integrated Circuits Conference*, pp. 25.5.1–25.5.4, 1992.

[10] S. Naffziger, "A sub-nanosecond 0.5μm 64b adder design," *in IEEE International Solid-State Circuits Conference*, pp. 362–363, 1996.

[11] M.R. Santoro and M.A. Horowitz, "SPIM: A pipelined 64×64-bit iterative multiplier," *IEEE Journal of Solid-State Circuits*, Vol. 24, No. 2, pp. 487–493, 1989.

[12] G. Goto, T. Sato, M. Nakajima, and T. Sukemura, "A 54 × 54-b regularly structured tree multiplier," *IEEE Journal of Solid-State Circuits*, Vol. 27, No. 9, pp. 1229–1236, 1992.

[13] G. Goto, A. Inoue, R. Ohe, S. Kashiwakura, S. Mitarai, T. Tsuru, and T. Izawa, "A 4.1-ns compact 54 × 54-b multiplier utilizing sign-select booth encoders," *IEEE Journal of Solid-State Circuits*, Vol. 32, No. 11, pp. 1676–1682, 1997.

[14] S. Kao, R. Zlatanovici, and B. Nikolic, "A 240ps 64b carry-lookahead adder in 90nm CMOS," *in IEEE International Solid-State Circuits Conference*, pp. 438–439, 2006.

[15] S. Kao, R. Zlatanovici, and B. Nikolic, "A low-leakage 2.5GHz skewed CMOS 32b adder for nanometer CMOS technologies," *in IEEE International Solid-State Circuits Conference*, pp. 380–381, 2005.

[16] T. Kobayashi and T. Sakurai, "Self-adjusting threshold-voltage scheme (SATS) for low-voltage high-speed operation," *Proceedings of IEEE Custom Integrated Circuits Conference*, pp. 271–274, 1994.

[17] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada, "1-V power supply high-speed digital circuit technology with multithreshold-voltage CMOS," *IEEE Journal of Solid-State Circuits*, Vol. 30, No. 8, pp. 847–854, 1995.

[18] S. Mutoh, S. Shigematsu, Y. Matsuya, H. Fukuda, and J. Yamada, "1V multi-threshold CMOS DSP with an efficient power management technique for mobile phone application," *in IEEE International Solid-State Circuits Conference*, pp. 168–169, 1996.

[19] T. Kuroda, T. Fujita, S. Mita, T. Nagamatsu, S. Yoshioka, K. Suzuki, F. Sano, M. Norishima, M. Murota, M. Kako, M. Kinugawa, M. Kakumu, and T. Sakurai, "A 0.9-V, 150-MHz, 10-mW, 4mm2, 2-d discrete cosine transform core processor with variable threshold-voltage scheme," *IEEE Journal of Solid-State Circuits*, Vol. 31, No. 11, pp. 1770–1779, 1996.

[20] J. Kao, A. Chandrakasan, and D. Antoniadis, "Transistor sizing issues and tool for multi-threshold CMOS technology," *Proceedings of ACM/IEEE Design Automation Conference*, pp. 409–414, 1997.

[21] L. Wei, Z. Chen, M. Johnson, and K. Roy, "Design and optimization of low voltage high performance dual threshold CMOS circuits," *Proceedings of ACM/IEEE Design Automation Conference*, pp. 489–494, 1998.

[22] V. Sundararajan and K.K. Parhi, "Low power synthesis of dual threshold voltage CMOS VLSI circuits," *Proceedings of IEEE International Symposium on Low Power Electronics and Design*, pp. 139–144, 1999.

[23] M. Hirabayashi, K. Nose, and T. Sakurai, "Design methodology and optimization strategy for dual-vth scheme using commercially available tools," *Proceedings of IEEE International Symposium on Low Power Electronics and Design*, pp. 283–286, 2001.

[24] K. Usami, N. Kawabe, M. Koizumi, K. Seta, and T. Furusawa, "Automated selective multi-threshold design for ultra-low standby applications," *Proceedings of IEEE International Symposium on Low Power Electronics and Design*, pp. 202–206, 2002.

[25] Y. Ji-Ren, I. Karlsson, and C. Svensson, "A true signal-phase-clock dynamic CMOS circuit technique," *IEEE Journal of Solid-State Circuits*, Vol. SC–22, No. 5, pp. 899–901, 1987.

[26] S.L. Lu, "Implementation of iterative networks with CMOS differential logic," *IEEE Journal of Solid-State Circuits*, Vol. SC-23, No. 4, pp. 1013–1017, 1988.

[27] C.Y. Wu and K.H. Cheng, "Latched CMOS differential logic (LCDL) for complex high-speed VLSI," *IEEE Journal of Solid-State Circuits*, Vol. 26, No. 9, pp. 1324–1328, 1991.

[28] S.L. Lu and M.D. Ercegovac, "Evaluation of two-summand adders implemented in ECDL CMOS differential logic," *IEEE Journal of Solid-State Circuits*, Vol. 26, No. 8, pp. 1152–1160, 1991.

[29] B.S. Kong, J.S. Choi, S.J. Lee, and K. Lee, "Charge recycling differential logic for low-power application," *in IEEE International Solid-State Circuits Conference*, pp. 302–303, 1996.

[30] D. Somasekhar and K. Roy, "Differential current switch logic: A low power DCVS logic family," *IEEE Journal of Solid-State Circuits*, Vol. 31, No. 7, pp. 981–991, 1996.

[31] S.Y. Choe, G.A. Rigby, and G.R. Hellestrand, "Half-rail differential logic," *in IEEE International Solid-State Circuits Conference*, pp. 420–421, 1997.

[32] D. Somasekhar and K. Roy, "LVDCSL: Low voltage differential current switch logic, a robust low power DCSL family," *Proceedings of IEEE International Symposium on Low Power Electronics and Design*, pp. 18–23, 1997.

[33] D. Harris and M.A. Horowitz, "Skew-tolerant domino circuits," *IEEE Journal of Solid-State Circuits*, Vol. 32, No. 11, pp. 1702–1711, 1997.

[34] J. Park, J. Lee, and W. Kim, "Current sensing differential logic: A CMOS logic for high reliability and flexibility," *IEEE Journal of Solid-State Circuits*, Vol. 34, No. 6, pp. 904–908, 1999.

[35] B.S. Kong, J.D. Im, Y.C. Kim, S.J. Jang, and Y.H. Jun, "Asynchronous sense differential logic," *in IEEE International Solid-State Circuits Conference*, pp. 284–285, 1999.

[36] C. Kim, J. Lee, K.H. Baek, E. Martina, and S.M. Kang, "High-performance low-power skewed static logic in very deep-submicron (VDSM) technolog," *Proceedings of IEEE International Conference on Computer Design*, pp. 59–64, 2000.

[37] V. Friedman and S. Liu, "Dynamic logic CMOS circuits," *IEEE Journal of Solid-State Circuits*, Vol. SC–19, No. 2, pp. 263–266, 1984.

[38] J.A. Pretorius, A.S. Shubat, and C.A.T. Salama, "Analysis and design optimization of domino CMOS logic with application to standard cells," *IEEE Journal of Solid-State Circuits*, Vol. SC–20, No. 2, pp. 523–530, 1985.

[39] C.K. Erdelyi, "Random logic design utilizing single-ended cascode voltage switch circuits in NMOS," *IEEE Journal of Solid-State Circuits*, Vol. SC–20, No. 2, pp. 591–594, 1985.

[40] J.T.Y. Chang and E.J. McCluskey, "Detecting resistive shorts for CMOS domino circuits," *Proceedings of the International Test Conference*, pp. 890–899, 1998.

[41] M. Shoji, "FET scaling in domino CMOS gate," *IEEE Journal of Solid-State Circuits*, Vol. SC–20, No. 5, pp. 1067–1071, 1985.

[42] V.G. Oklobdzija and R.K. Montoye, "Design-performance trade-offs in CMOS-Domino logic," *IEEE Journal of Solid-State Circuits*, Vol. SC–21, No. 2, pp. 304–306, 1986.

[43] J. Yuan and C. Svensson, "High-speed CMOS circuit technique," *IEEE Journal of Solid-State Circuits*, Vol. 24, No. 1, pp. 62–70, 1989.

[44] E.J. Yoffa and P.S. Hauge, "ACORN: A local customization approach to DCVS physical design," *Proceedings of 22nd Design Automation Conference*, pp. 32–38, 1985.

[45] K.M. Chu and D.L. Pulfrey, "Design procedures for differential cascode voltage switch circuits," *IEEE Journal of Solid-State Circuits*, Vol. SC–21, No. 6, pp. 1082–1087, 1986.

[46] K.M. Chu and D.L. Pulfrey, "A comparison of CMOS circuit techniques: Differential cascode voltage switch logic versus conventional logic," *IEEE Journal of Solid-State Circuits*, Vol. SC–22, No. 4, pp. 528–532, 1987.

[47] N. Kanopoulos and N. Vasanthavada, "Testing of differential cascode voltage switch (DCVS) circuits," *IEEE Journal of Solid-State Circuits*, Vol. 25, No. 3, pp. 806–813, 1990.

[48] N. Kanopoulos, D. Pantzartzis, and F.R. Bartram, "Design of self-checking circuits using DCVS logic: A case study," *IEEE Transactions on Computers*, Vol. 41, No. 7, pp. 891–896, 1992.

[49] P. Ng, P.T. Balsara, and D. Steiss, "Performance of CMOS differential circuits," *IEEE Journal of Solid-State Circuits*, Vol. 31, No. 6, pp. 841–846, 1996.

[50] N. Sirisantana, A. Cao, S. Davidson, C.K. Koh, and K. Roy, "Selectively clocked skewed logic (SCSL): A robust low-power logic style for high- performance applications," *Proceedings of IEEE International Symposium on Low Power Electronics and Design*, pp. 267–270, 2001.

[51] W. Jeong, K. Roy, and C.K. Koh, "High-performance low-power carry select adder using dual transition skewed logic," *Proceedings of European Solid-State Circuits Conference*, pp. 145–148, 2001.

[52] N. Sirisantana and K. Roy, "A time borrowing selectively clocked skewed logic for high-performance circuits in scaled technologies," *Proceedings of European Solid-State Circuits Conference*, pp. 181–184, 2003.

[53] N. Sirisantana and K. Roy, "Selectively clocked cmos logic style for low-power noise-immune. operations in scaled technologies," *Proceedings of Design and Test in Europe Conference and Exhibition*, pp. 1160–1161, 2003.

[54] A. Cao, N. Sirisantana, C.K. Koh, and K. Roy, "Synthesis of selectively clocked skewed logic circuits," *Proceedings of International Symposium on Quality Electronic Design*, pp. 229–234, 2002.

[55] A. Cao, N. Sirisantana, C.K. Koh, and K. Roy, "Integer linear programming-based synthesis of skewed logic circuits," *Proceedings of Asia and South Pacific Design Automation Conference*, pp. 820–823, 2003.

[56] A. Cao, N. Sirisantana, C.K. Koh, and K. Roy, "Synthesis of skewed logic circuits," *ACM Trans. on Design Automation of Electronic Systems*, Vol. 10, No. 2, pp. 205–228, 2005.

[57] R. Puri, A. Bjorksten, and T.E. Rosser, "Logic optimization by output phase assignment in dynamic logic synthesis," *Proceedings of International Conference on Computer-Aided Design*, pp. 2–7, 1996.

[58] G. Yee and C. Sechen, "Dynamic logic synthesis," *Proceedings of IEEE Custom Integrated Circuits Conference*, pp. 345–348, 1997.

[59] T. Thorp, G. Yee, and C. Sechen, "Domino logic synthesis using complex static gates," *Proceedings of International Conference on Computer-Aided Design*, pp. 242–247, 1998.

[60] P. Patra and U. Narayanan, "Automated phase assignment for the synthesis of low power domino circuits," *Proceedings of ACM/IEEE Design Automation Conference*, pp. 379–384, 1999.

[61] K.W. Kim, C.L. Liu, and S.M. Kang, "Implication graph based domino logic synthesis," *Proceedings of International Conference on Computer-Aided Design*, pp. 111–114, 1999.

[62] D. Samanta, N. Sinha, and A. Pal, "Synthesis of high performance low power dynamic CMOS circuits," *Proceedings of Asia and South Pacific Design Automation Conference*, pp. 99–104, 2002.

[63] S.B. Akers, "Binary decision diagrams," *IEEE Transactions on Computers*, Vol. 27, No. 6, pp. 509–516, 1978.

[64] T. Karoubalis, G.P. Alexiou, and N. Kanopoulos, "Optimal synthesis of differential cascode voltage switch (DCVS) logic circuits using ordered binary decision diagrams (OBDDs)," *Proceedings of European Design Automation Conference*, pp. 282–287, 1995.

[65] G.M. Jacobs and R.W. Brodersen, "A fully asynchronous digital signal processor using self-timed circuits," *IEEE Journal of Solid-State Circuits*, Vol. 25, No. 6, pp. 1526–1537, 1990.

[66] A.J. McAuley, "Dynamic asynchronous logic for high-speed CMOS systems," *IEEE Journal of Solid-State Circuits*, Vol. 27, No. 3, pp. 382–388, 1992.

[67] A.J. McAuley, "Four state asynchronous architectures," *IEEE Transactions on Computers*, Vol. 41, No. 2, pp. 129–142, 1992.

[68] J.D. Garside, "A CMOS VLSI implementation of an asynchronous ALU," *Proceedings of IFIP Conf. Asynchronous Design Methodologies*, pp. 181–192, 1993.

[69] M.A. Franklin and T. Pan, "Performance comparison of asynchronous adders," *Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 117–125, 1994.

[70] S. Hauck, "Asynchronous design methodologies: An overview," *Proceedings of IEEE*, Vol. 83, No. 1, pp. 69–93, 1995.

[71] J.V. Woods, P. Day, S.B. Further, J.D. Garside, N.C. Paver, and S. Temple, "AMULET1: An asynchronous ARM microprocessor," *IEEE Transactions on Computers*, Vol. 46, No. 4, pp. 385–398, 1997.

[72] G.A. Ruiz, "Evaluation of three 32-bit CMOS adders in DCVS logic for self-timed circuits," *IEEE Journal of Solid-State Circuits*, Vol. 33, No. 4, pp. 604–613, 1998.

[73] 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," 2003.

発表論文一覧

本研究に関する発表論文

学術雑誌

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

[2] Masao Morimoto, Yoshinori Tanaka, Makoto Nagata, and Kazuo Taki, "Logic Synthesis Technique for High Speed Differential Dynamic Logic with Asymmetric Slope Transition", *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E88-A, no. 12, pp. 3324–3331, Dec 2005.

[3] Masao Morimoto, Makoto Nagata, and Kazuo Taki, "Asymmetric Slope Dual Mode Differential Logic Circuit for Compatibility of Low-power and High-speed Operations", *IEICE Transactions on Electronics*, to appear.

学術講演

[4] 森本薫夫, 永田真, 瀧和男, "非対称な信号遷移を用いた高速論理回路方式", 電子情報通信学会 デザインガイア 2004, ICD2004-139, pp. 25–30, Nov 2004.

[5] 森本薫夫, 永田真, 瀧和男, "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", 電子情報通信学会 デザインガイア 2005, ICD2005-153, pp. 25–30, Nov 2005.

[6] 森本薫夫, 永田真, 瀧和男, "高速モードと低消費電力モードを有する2線式論理回路の設計手法", 電子情報通信学会 デザインガイア 2006, VLD2006-60, pp. 53–58, Nov 2006.

[7] 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”, 電子情報通信学会 デザインガイア 2001, Nov 2001.

[8] 田中義則, 森本薫夫, 永田真, 瀧和男, “2 線 2 相式論理回路方式 ASDDL/ASD-CMOS の論理合成手法”, 電子情報通信学会 デザインガイア 2003, ICD2003–149, pp. 55–60, Nov 2003.

[9] 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 ICD/VLD 共催研究会, ICD2003–235, pp. 37–42, Mar 2004.

[10] 瀧和男, 八木幹雄, 森本薫夫, 尾形俊郎, 池見憲一, 北村清志, “高速消費電力論理回路方式 ASDDL/ASD-CMOS とその評価”, 電子情報通信学会 DA シンポジウム 2001 論文集, pp. 113–118, Jul 2001.

受賞

[11] 森本薫夫, 永田真, 瀧和男, “高速モードと低消費電力モードを有する 2 線式論理回路の設計手法”, 電子情報通信学会 デザインガイア 2006 ポスタ賞.