修 士 論 文

ウェーブパイプラインプロセッサに適した論理合成アルゴリズムの研究

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

森田 智祥

2008 年 3 月

修 士 論 文

ウェーブパイプラインプロセッサに適した論理合成アルゴリズムの研究

指導教官　日比野 靖 教授

審査委員主査　日比野 靖 教授
審査委員　田中 清史 准教授
審査委員　井口 寧 准教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

610087 森田 智祥

提出年月: 2008 年 2 月

概 要

近年、シングルプロセッサの性能向上が限界に近づきつつある。ウェーブパイプラインという動作原理を適用することによって、その性能限界を超えられる可能性がある。しかし、ウェーブパイプラインを想定したCADが存在しないために、非常に設計が困難であるという問題がある。

本稿では、ウェーブパイプラインの設計に適したCADの実現のための論理合成アルゴリズムを提案する。そして、提案手法を組み込んだシステムを作製し評価を行い、その有用性を示す。

目 次

図 目 次

表 目 次

第1章　はじめに

1.1　研究の背景

現在、プロセッサの構成が単一構成から複数構成へと移行しつつある。これは、最大遅延時間によって動作周波数が決定する現在のパイプラインプロセッサの動作原理では、パイプラインステージの分割によるクロックサイクルの短縮が限界に近づき、シングルプロセッサでの性能向上が望めなくなってきているためである。これに対して、最大遅延と最小遅延の差でクロックサイクル時間が決定するウェーブパイプラインという動作原理がある。この動作原理を適用することによって、シングルプロセッサの更なる性能向上が期待できる。

従来のパイプラインでは、最大遅延時間によって動作周波数が決定されるため、最大遅延時間の短縮が最も重要であった。これに対して、ウェーブパイプライン動作を実現する上で最も重要となるのは、最大遅延と最小遅延の差の短縮である。

一般的なCADは、従来のパイプラインの設計のための論理合成アルゴリズムが実装されている。したがって、アルゴリズムは、面積最小（素子数最小）又は最大遅延時間を最小とすることを目標とした多段論理合成アルゴリズムである。このため、各入力に対する論理段数にばらつきが生じることになる。ウェーブパイプラインでは遅延時間差を最小化することが最も重要である。よって、論理段数がばらつく多段論理合成では、遅延時間差を短縮するのは根本的に難しい。したがって最適設計のためには、ウェーブパイプラインの設計に適したCADがなければほぼ不可能である。

これまでの設計工程は、ウェーブパイプラインの設計に適したCADが存在しないため、一般的なCADを用いて論理回路を設計し、遅延バッファを挿入して遅延差を短縮させるというものであった[1]。この設計法では、必要以上に回路面積が増加することや十分に遅延差を短縮できないなどの問題が生じる。特に後者の問題は、ウェーブパイプラインを実現する上で最も重大な問題点となっている。

1.2 研究の目的

ウェーブパイプラインの最適な設計を実現するためには、遅延時間差を最小化する論理合成アルゴリズムを実装した専用 CAD が必要不可欠である。

本研究では、論理回路の各入力に対する論理段数を揃えることで、論理合成段階で遅延差を短縮する論理合成アルゴリズムを提案する。提案手法によって、必要以上の回路面積増加や遅延差短縮の限界という従来手法の問題を解決し、ウェーブパイプラインに適した論理回路を生成することが本研究の目的である。

1.3 本論文の構成

本論文は 7 章で構成される。第 2 章では、通常のパイプライン及びウェーブパイプラインの動作原理について述べる。第 3 章では、本研究で前提としている CMOS 素子の遅延特性について述べる。第 4 章では、従来のウェーブパイプラインの設計手法について説明し、その問題点を示す。第 5 章では、本研究で提案する論理段数を揃える論理合成法について述べる。まず、提案手法で用いるクワイン・マクラスキー法について説明する。次に論理段数を揃えるための戦略について説明する。第 6 章では、提案手法の評価について述べる。まず、提案手法の評価のために作製した論理合成システムの実装法について述べる。そして、そのシステムによって論理回路を作製し、評価して提案手法の有用性を示す。最後に第 7 章で本研究のまとめを述べる。

第2章　ウェーブパイプライン

2.1　パイプライン処理

パイプライン処理とは、複数の命令を小さな処理ステップに分解して、処理ステップ単位で時間的に重複させて並行的に実行する処理方式である。今日では、パイプライン処理はプロセッサにおける一般的な技術となっている。

ここで、以下の処理を実行する５つのユニットから成るプロセッサがあると仮定する。

- IF（命令フェッチ）
- ID（命令デコード）
- EXE（命令実行）
- MEM（メモリアクセス）
- WB（結果の書き込み）

このプロセッサの１命令は、IF から WB までの処理がすべて完了するまでとする。このプロセッサ上で、複数の命令を直列的に実行した例を図 2.1 に、並列的に実行したパイプライン処理の例を図 2.2 に示す。図 2.1 では、３命令実行するのに２０クロックサイクルかかっている。これに対して図 2.2 では、パイプライン処理により、３個の命令を並列に実行しているため、７クロックサイクルで処理することができている。よってこの例では、パイプライン処理によって直列的な処理のおよそ３倍の処理能力を得ていることになる。このように、パイプライン処理を行って命令を実行した場合の方がはるかに処理効率が良いことがわかる。

一般的に、パイプライン処理では IF から WB までの各処理の段階のことをステージと呼ぶ。以後本論文では、ステージという用語を用いて説明を行う。

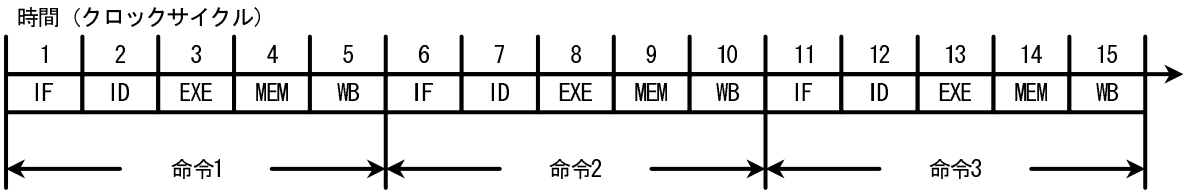


図 2.1: 直列的な処理

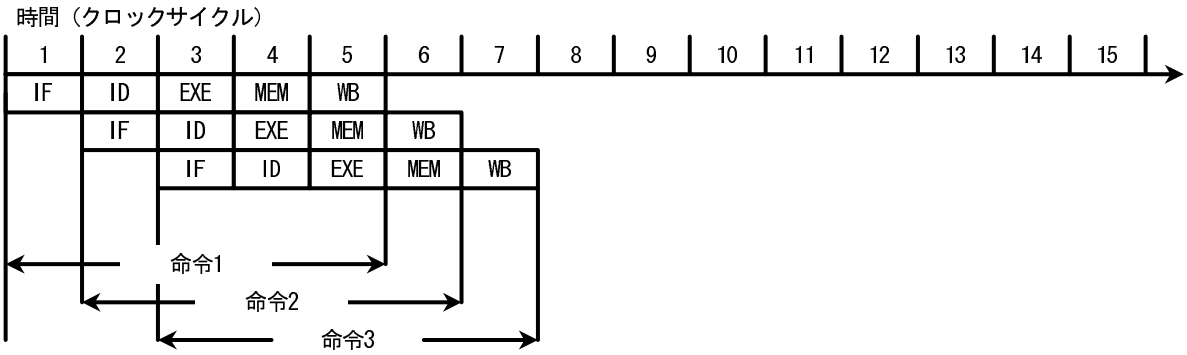


図 2.2: パイプライン処理

2.2　通常のパイプラインの動作原理

通常のパイプラインはクロックに同期して処理を行っている。クロックによって同期する同期式システムの場合、図 2.3 のように各々の命令を実現する回路の間にラッチなどの記憶素子を配置する。ラッチは、クロックが入力されてから次のクロックが入力されるまでの間、各々のステージの処理結果を保持し、クロックが入力されると次のステージに結果を受け渡すという役割を担っている。ラッチ間の各ステージの回路は、処理内容ごとに異なるので、処理時間も当然異なる。各々のステージが正確に処理を行うためには、１クロックの長さが各々のステージの処理時間よりも大きい必要がある。通常のパイプラインでは１クロックの長さは最も処理時間の長いステージに合わせて設計される。本論文では、各ステージの処理時間を遅延時間と呼ぶ。

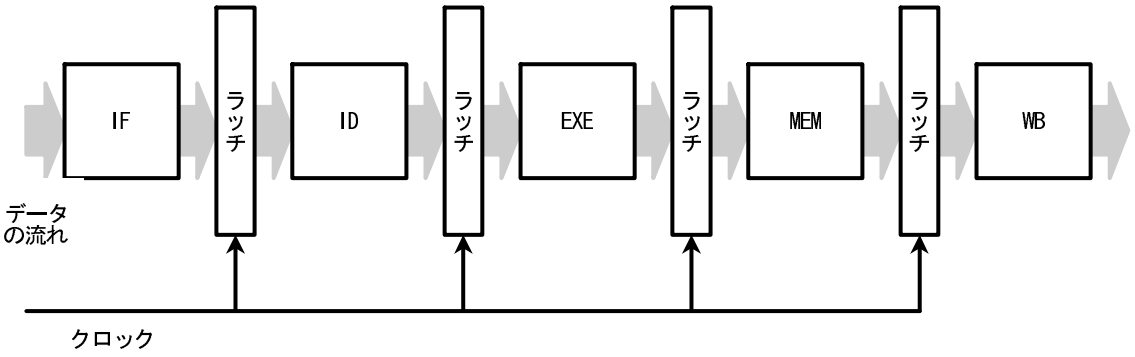


図 2.3: ラッチの配置

2.3　ウェーブパイプラインの動作原理

通常のパイプラインでは、2.2 節で述べたように最も遅延時間の長いステージによって動作クロック周波数が決定される。しかし、すべてのステージが最大の処理時間を必要としているわけではない。

図 2.4 は、あるステージの論理回路の例である。論理素子を通過する数が最も多いパス X と、最も少ないパス Y とでは、データがラッチに到着する時間が異なる。本論文では、各ステージの論理回路において図 2.4 のパス X のような最も長いパスの遅延時間を最大遅延、パス Y のような最も短いパスの遅延を最小遅延と定義し、最大遅延と最小遅延の差を遅延差と定義する。これらの遅延時間は、図 2.5 のように表わされる。

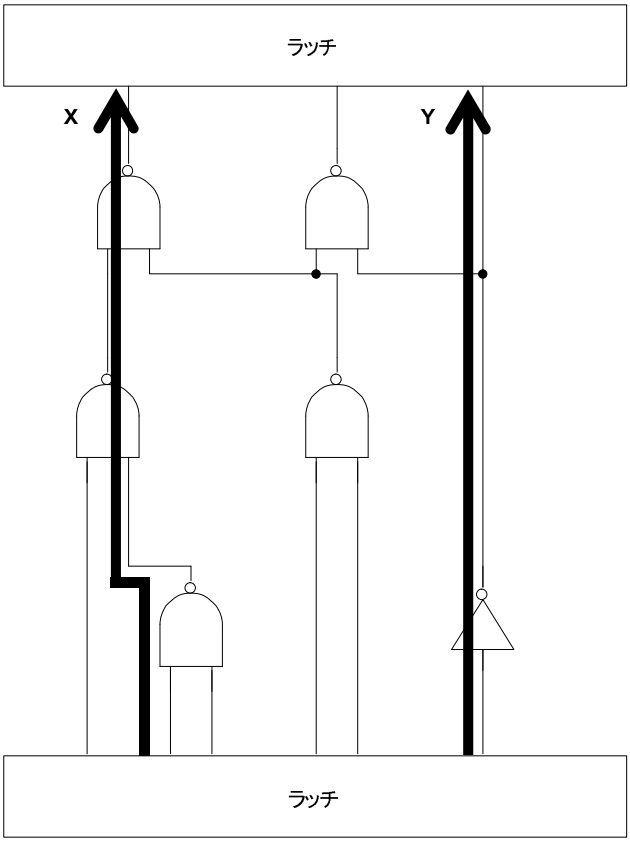


図 2.4: あるステージの論理回路の例

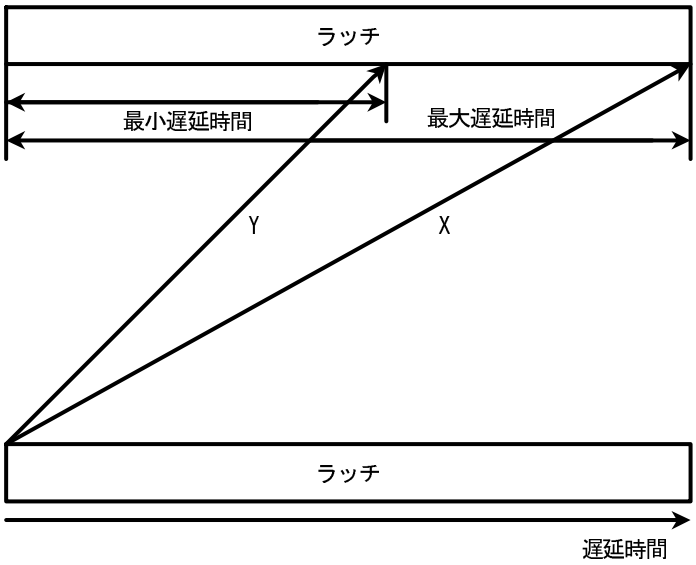


図 2.5: パス X 及び Y の遅延時間

パイプラインが正常に動作するためには、ラッチなどの記憶素子に正確に演算された値が記憶され、その値が次の処理に正確に伝わりさえすればよい。このように考えると、１クロックの長さは、最も処理時間の長いステージの処理に合わせる必要はなく、最も最大遅延と最小遅延の差が長いステージの処理時間に合わせればパイプライン処理は正確に動作することになる。このように、遅延差によって動作クロック周波数が決定されるパイプラインの動作原理をウェーブパイプラインと言う。

2.1 節のプロセッサにおいて、各ステージの最大遅延、最小遅延および遅延差が図 2.6 のようであるとする。遅延時間は図 2.5 のように表記されているものとする。このとき、最も遅延時間が長いのは EXE ステージであることから、通常のパイプラインでは、EXE ステージの最大遅延時間によって動作クロック周波数が決定される。これに対して、ウェーブパイプラインを用いてこのプロセッサモデルを動作させる場合、最も遅延差が長いのは EXE ステージであることから、EXE ステージの遅延差によって動作クロック周波数が決定される。ここで、図 2.6 の EXE ステージの最大遅延時間と遅延差を比較すると、遅延差の方が短い。よってこの例では、ウェーブパイプラインの方が通常のパイプラインよりも動作クロック周波数を高くできる。

図 2.6: 各ステージにおける遅延時間

各ステージが図 2.6 の遅延時間で動作する場合の通常のパイプラインとウェーブパイプラインの動作の様子を図 2.7 及び 2.8 に示す。図 2.7 からわかるように、通常のパイプラインでは EXE ステージの最大遅延時間によってクロック周期が決まっているので、他のステージの最大遅延から次のクロックが入力されるまでの時間が大きい。これに対して、ウェーブパイプラインでは、図 2.8 で示されるように、各々のステージの最大遅延から次のクロックが入力されるまでの時間が、通常のパイプラインよりも小さい。このことからもウェーブパイプラインの処理効率の高さがわかる。

また図 2.8 を見るとわかるように、ウェーブパイプラインにおいては、クロックの周期は一定であるが各パイプラインステージに入るクロックのタイミングは同時ではない。これは、各ステージに存在しているデータが 1 命令分だけではなく複数存在している可能性があることを示している。すなわち、各ステージの論理回路で多くの時間何らかのデータが処理されていることを意味する。よってウェーブパイプラインでは、存在している資源をより無駄なく使うことができる。

ウェーブパイプラインの動作原理を適用することによって、プロセッサの性能を向上させられることは以前から知られている [3]。しかし、現在のマイクロプロセッサにはほとんど適用されていない。その理由の一つに、ウェーブパイプラインの適用を前提とした CAD が存在しないことが挙げられる。通常のパイプラインを前提とした CAD は、最大遅延のみを考慮しているため、遅延差を考慮しなければならないウェーブパイプラインの設計には不向きである。したがって、ウェーブパイプラインを適用するには専用の CAD の開発が必要不可欠である。

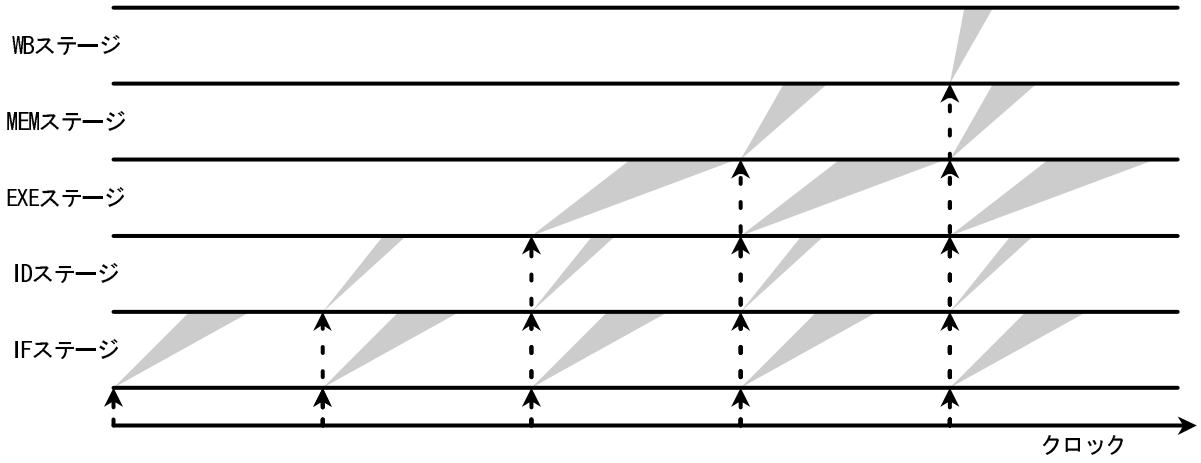


図 2.7: 通常のパイプラインの動作周期

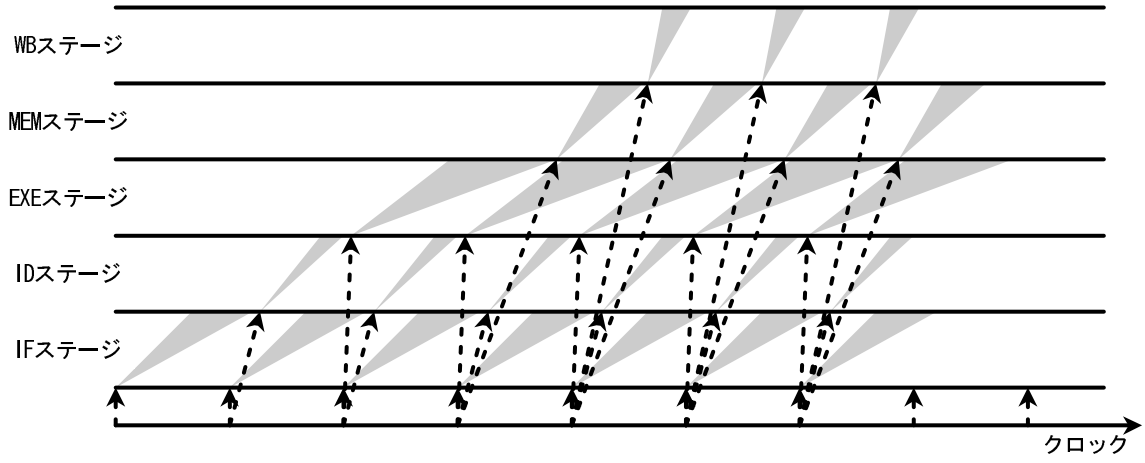


図 2.8: ウェーブパイプラインの動作周期

第3章　MOSFETの動作原理とCMOSの遅延特性

3.1　MOSトランジスタ

現在のマイクロプロセッサはCMOS技術によって成り立っている。CMOSとは、PMOSとNMOSによって構成される相補型のデバイスのことである。本研究においても、CMOSで構成された論理回路を前提として遅延などの評価を行う。そこで、本章ではCMOSで構成された素子の遅延特性について述べる。

CMOSの遅延特性を述べる前に、その構成要素となるMOSトランジスタの動作原理とその性質について述べる。MOSトランジスタは、電界効果トランジスタ (Field Effect Transistor:FET) の一種であり、電流通路の導電率を第３電極によって静電的に変化させ、電流を制御しようとする半導体素子である [6]。MOS技術を用いた集積システムは、３枚の導電性物質の層 (以下、レベル[1]と呼ぶ) を積層化し、層間には相互分離のための絶縁膜を配した構造を持っている。最上部には金属層レベルがある。中間部にはポリシリコン・レベルがあり、最下部には拡散層レベルがある。これら３つのレベル上では、導電経路の形状を定めるためのパターン (平面図形) の形成が必要となる。また、層間の絶縁膜においては、異種レベル上のいくつかの点を電気的に接続するために、接続用の開孔 (コンタクト孔) の位置決定が必要となる。これら形状決定用パターンは、写真の陰画に似たマスクにあらかじめ描いておき、回路製作の過程で各レベルに転写することにより作られる。

集積回路のチップ上では、図3.1に示すように、ポリシリコンの経路が拡散層の経路と立体交差する箇所にMOSトランジスタが形成される。ソースおよびドレイン端子は形状的には対称である。NチャンネルMOSトランジスタでは、ドレイン・ソース間電圧V_{ds}が通常正になるように端子名を決めている。ポリシリコンの経路と拡散層経路とが交差する長方形の領域をさらに拡大して図3.2に示す。この交差領域の中で、ポリシリコン部分をゲートと呼んでいる。素子作製の過程では、ポリシリコンの経路を形成した後に、不純物の拡散を行って拡散層経路を形成している。このため、ポリシリコン・ゲート及びゲート用酸化膜で被覆されたチャンネル領域では、拡散プロセスの影響が及ぶことがなく、拡散層による導電経路が形成されない。したがって、トランジスタのソース・ドレイン電極間には直接的な電気接続は存在しない。なお、金属、ポリシリコン、及び拡散層の各導電経路はいずれも十分な導電率を持っているので、特に注意しないかぎり、金属線に近いものと考えてよい。

ゲート電極上に何の電荷も存在しない場合には、ドレイン・ソース間の経路は遮断状

[1]各導電物質の層を含んだ幾何学的な平面を意味しており、導電物質の層そのものではない

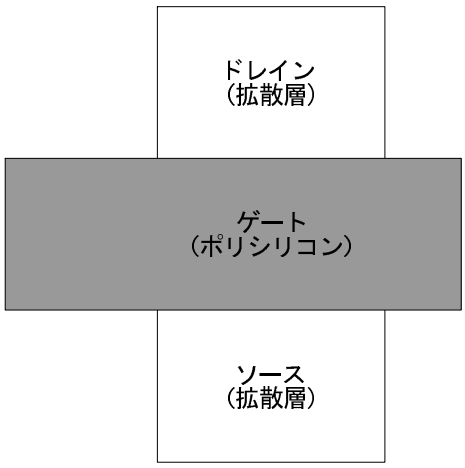


図 3.1: MOS トランジスタの上方から見た平面図

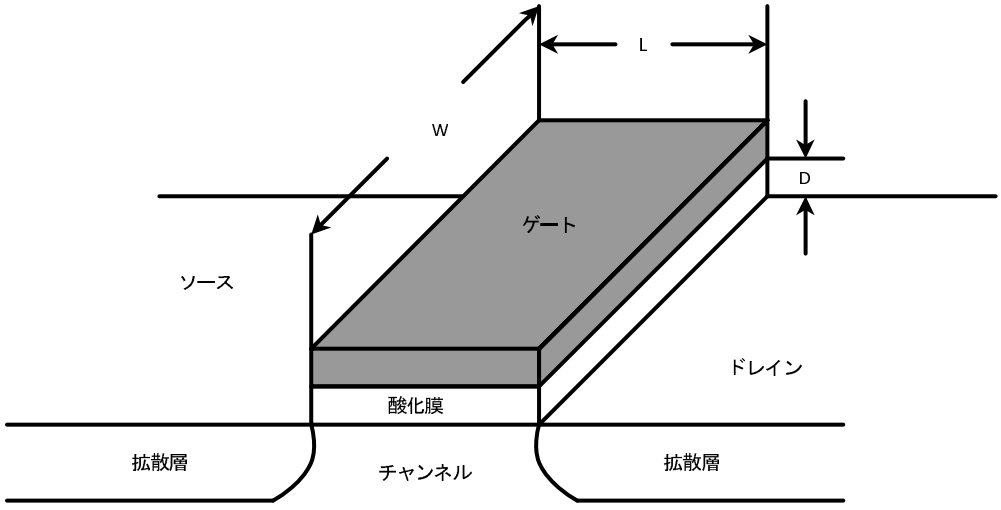


図 3.2: MOS トランジスタの断面図

態のスイッチと見なすことができる[2]。ゲートは基板から薄い酸化膜を介して分離されており、コンデンサとして作用する。ゲートに十分な量の正電荷が存在し、ゲート・ソース間電圧 V_{gs} が閾値電圧 V_{th} を超えると、多数の電子がゲート直下のチャンネル部分に惹き寄せられ、ドレイン・ソース間には導電性の経路が形成される。このようなトランジスタは、エンハンスメント形 MOSFET と呼ばれている。

MOS トランジスタの基本的な動作原理は、ゲート電極上の電荷の作用によってソース・ドレイン間のチャンネルを流れる負電荷の量とその流れを制御することにある。半導体内では、通常の場合、電子速度は加速するために加えた電界に比例する。よって、走行時間 τ は、ドレイン・ソース間電圧 V_{ds} が小さいとき[3]、ゲート長 L、キャリアの速度 v、キャリアの移動度 μ、ドレイン・ソース間電界 E_{ds} によって次式で与えられる。

$$\tau = \frac{L}{v} = \frac{L}{E_{ds}\mu} = \frac{L^2}{\mu V_{ds}} \tag{3.1}$$

また、ゲート容量 C_g は、誘電率 ϵ、ゲート長 L、ゲート幅 W、酸化膜の厚さ D によって、次式で表わされる。

$$C_g = \frac{\epsilon W L}{D} \tag{3.2}$$

これより、電荷 Q 及び、ドレイン・ソース間電流 I_{ds} はそれぞれ以下の式によって表わされる。

$$Q = -C_g V_{gs} = -\frac{\epsilon W L}{D} V_{gs} \tag{3.3}$$

$$I_{ds} = -I_{sd} = -\frac{Q}{\tau} = \frac{\mu \epsilon W}{LD} V_{gs} V_{ds} \tag{3.4}$$

式 3.4 より、ドレイン・ソース間電流 I_{ds} は、ドレイン・ソース間電圧 V_{ds} 及び V_{gs} の双方に比例することがわかる。

また、オームの法則より、ドレイン・ソース間の抵抗 R は、次式によって表わされる。

$$R = \frac{V_{ds}}{I_{ds}} = \frac{LD}{\mu \epsilon W V_{gs}} = \frac{L^2}{\mu C_g V_{gs}} \tag{3.5}$$

本論文では、式 3.5 の R を MOS トランジスタのオン抵抗 R_{on} と定義し、以降のオン抵抗とはこれを指すものとする。

MOS トランジスタによって構成される素子及びシステムの遅延時間はオン抵抗 R_{on} とゲート容量 C_g の積で表わされ、式 3.5 より、次式で表わされる。

$$R_{on} C_g = \frac{L^2}{\mu V_{gs}} \tag{3.6}$$

この式より、システムの遅延時間を抑えるには、ゲート長 L を短くすればよいことがわかる。近年の微細加工技術の進歩による汎用プロセッサの処理能力向上の要因を、式 3.6 は端的に表していると言えるであろう。

[2]チャンネル内に固定正電荷と電子を多数含んだディプリーション型素子では、ゲート電極上に電荷が存在しない場合に導通状態となり、負の電圧を加えると電流が減少していく。

[3]V_{ds} が大きくなると、チャンネル内での電子電荷の量が均一ではなくなり、加速電界 E も場所に依存し始める。

3.2 CMOSインバータの遅延特性

CMOS の遅延特性を示すために、例として CMOS インバータを用いる。CMOS インバータの遅延特性を拡張することによって、他の CMOS 素子の遅延特性を求めることができる。

図 3.3 に CMOS インバータにおける PMOS と NMOS の構成を示す。PMOS トランジスタは正孔がキャリアとして動作するトランジスタであり、NMOS は電子がキャリアとして動作するトランジスタである。PMOS トランジスタはゲート電圧が high のときに遮断状態となり、low のとき導通状態となる。NMOS トランジスタは電子がキャリアなので、正孔がキャリアである PMOS とは逆の働きをする。また PMOS と NMOS はキャリアが異なるのでキャリアの移動度も異なる。正孔の移動度は電子の移動度の約 1/3 であり [7]、PMOS と NMOS の信号変化の時間には違いが生じる。これは、式 3.5 において、PMOS と NMOS とでオン抵抗が異なるということになり、PMOS がオンになり信号変化が起こる時間と NMOS がオンになり信号変化が起こる時間が異なるということになる。本論文において、PMOS がオンになり信号が変化する場合の遅延時間を立ち上がり遅延時間、同様に NMOS の場合を立ち下がり遅延時間と定義する。式 3.5 から、ゲート長 L、ゲート幅 W、酸化膜の厚さ D を調節することにより、PMOS と NMOS のオン抵抗を合わせることができる。実際のプロセスの工程において、PMOS と NMOS のゲート長 L、酸化膜の厚さ D は揃えるのが一般的であるので、各々のゲート幅 W を調節することでオン抵抗を揃えて、立ち上がり時間と立ち下がり時間を揃えることになる。このように、ウェーブパイプライン動作を前提とする場合、パス間の遅延を揃えるだけでなく、素子自体の遅延差も揃えることが望ましい。

次に、CMOS インバータ同士の接続時における遅延時間を考える。図 3.4 に接続された CMOS インバータとその等価回路を示す。PMOS と NMOS の接続は図 3.3 と同様である。図 3.4(a) は接続された CMOS インバータを示し、図 3.4(b) は抵抗、スイッチ、コンデンサを用いて等価回路として表したものを示す。図 3.4(b) の値は、それぞれ、オン抵抗 R_{on}、MOS トランジスタのゲートの入力容量 C_g、ドレイン側における拡散領域で発生する拡散容量 C_d を表す。この等価回路の図において、CMOS インバータの前段は電圧が low とする。インバータの特性から、次段は電圧が high となる。

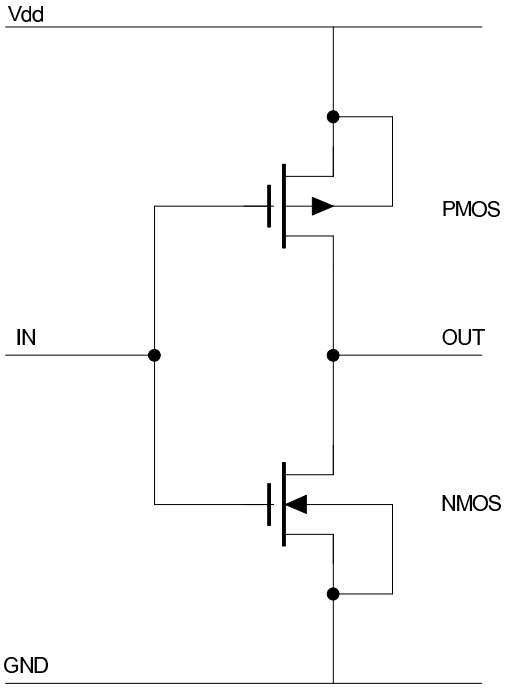
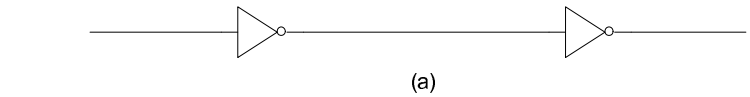


図 3.3: CMOS インバータの構成

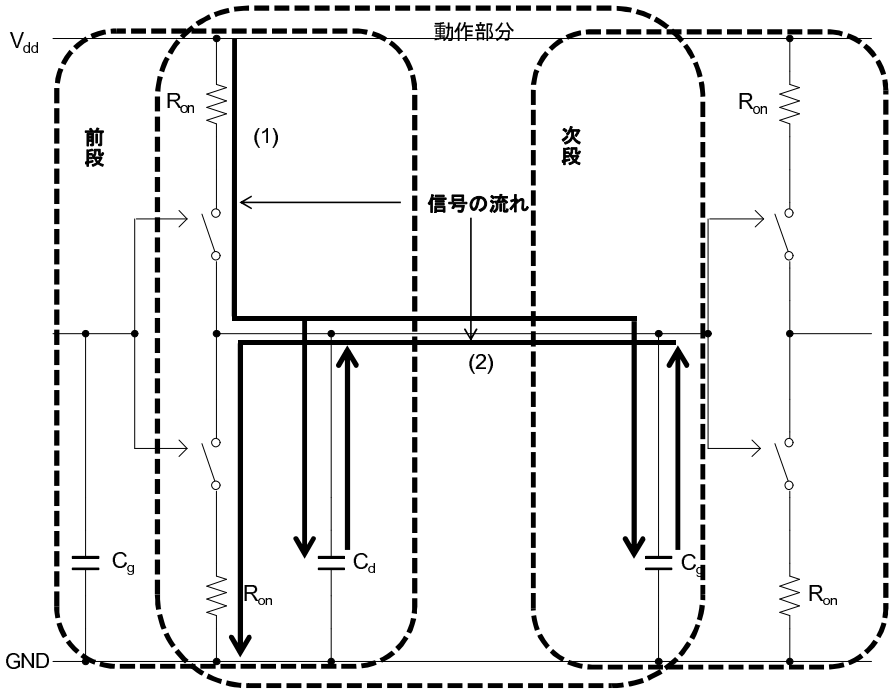
まず、前段のインバータの電圧が low である場合、PMOS 側がオンになり NMOS 側はオフとなる。このとき、電流は図 3.4 中の (1) の様に流れて拡散容量 C_d と入力容量 C_g を充電する。この充電が完了すると、次段のインバータの電圧が high となり、PMOS 側がオフ、NMOS 側がオンになる。次に、前段のインバータが high になった場合、PMOS 側がオフ、NMOS 側がオンになる。よって、電流 (1) によって充電された拡散容量 C_d と入力容量 C_g は、電流 (2) のように前段のインバータの NMOS に流れ込み放電される。したがって、次段のインバータの電圧が low になり、PMOS 側がオンになり NMOS 側はオフとなる。このように、接続されたインバータ間を流れる信号は、オン抵抗と拡散容量、入力容量の充放電によって伝わっていく。よって、この信号の遅延 T_d は、次式のように表わされる。

$$T_d = R_{on}(C_g + C_d) \tag{3.7}$$

式 3.7 において、R_{on} と C_d は前段のインバータのオン抵抗及び拡散容量であり、C_g は次段のインバータの入力容量である。このように、CMOS インバータにおける遅延は接続された各素子間で発生するものと考えることができる。本研究では、遅延の評価はすべて式 3.7 に基づいている。



(a)



(b)

図 3.4: 接続された CMOS インバータとその等価回路

このようなCMOSインバータの遅延特性は、他のCMOS論理素子にも適用することができる。例えば、図3.5に示す2入力NANDの場合、破線で示されているPMOSのネットワークとNMOSのネットワークを、それぞれ合成抵抗と考える。このとき、NMOS側のネットワークは、すべてONになった場合に導通状態になるので、すべてのNMOSの抵抗の直列接続と考えることができる。同様に、PMOSのネットワークの場合も、すべてがONになった時はすべてのPMOSの抵抗の並列接続と見なすことができる。図3.5の様な2入力NANDの場合、すべてのPMOS抵抗値がrであるとすると、すべてがONになった時、オン抵抗はr/2となり最も遅延時間が小さくなる。逆にどちらか一方がONになった時、オン抵抗はrとなり、最も遅延時間が大きくなる。このように考えると、2入力NANDの素子自体の遅延差は、rとr/2の差以下はあり得ない。同様に、n入力のNAND素子1個の遅延差は最小でrとr/nの差ということになる。よってCMOS論理素子の遅延時間差は多入力になるほど広がる。

なお、図3.4の様なモデルにおいて、実際には配線抵抗と配線容量というパラメータも存在する。現在の論理回路設計においては、配線抵抗と配線容量が遅延に及ぼす影響は非常に大きく、無視できるものではない。しかし、本研究では、論理回路の論理の設計までを考慮の範囲に設定しており、レイアウトには踏み込んでいないため、配線抵抗と配線容量は考慮しない。

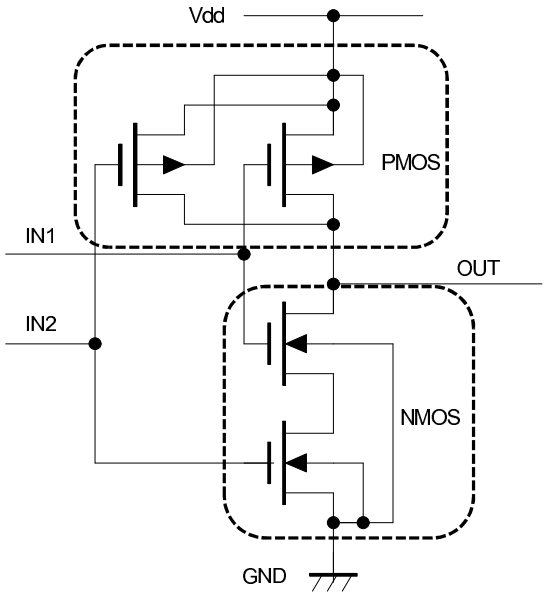


図 3.5: CMOS2入力NANDの構成

第4章　従来の遅延差短縮アルゴリズム

4.1　遅延バッファ挿入による遅延差短縮

第2章で述べたように、ウェーブパイプラインの性能を引き出すためには遅延差を短縮することが重要である。遅延差を短縮するためには、最大遅延を短縮する、あるいは最小遅延を最大遅延に近づけるという2通りの方法が考えられる。一般的なCADツールを使って設計する場合、付属している論理合成ツールは、最大遅延をできる限り短縮する論理合成を行う。そのため、出力された論理の最大遅延を最小遅延に近づけることは非常に難しい。そこで従来の遅延差短縮を実現する手法としては、最小遅延を最大遅延に近づけるという方法を採用している。具体的な戦略としては、池田によって考案された遅延バッファを挿入するアルゴリズム [1] が挙げられる。

従来のウェーブパイプラインの設計は、主にPARTHENON[4] のCADツールを用いて行われていた。図4.1に従来の設計のフローチャートを示す。

図4.1に示されるフローチャートでは、設計対象となる論理回路全体の遅延情報を元にレイアウトと遅延バッファ挿入を繰り返している。遅延の見積もり法と遅延バッファを挿入するアルゴリズムを次小節以降で説明する。

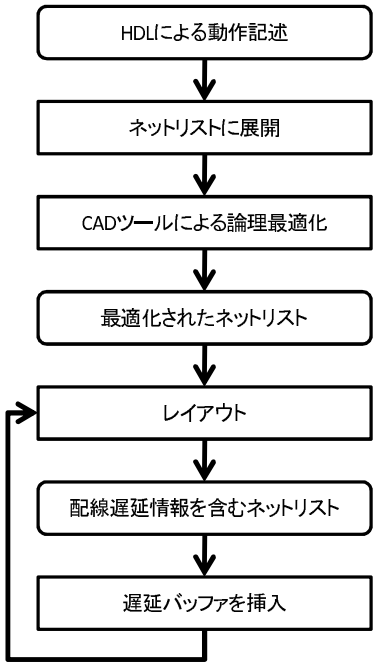


図 4.1: 従来の設計手法のフローチャート

4.1.1 遅延の見積もり

遅延差の短縮のためには、まず対象となる回路の遅延情報が必要である。ここでは、ある論理回路の入力から出力までの最小遅延および最大遅延の見積もり方法を説明する。

ある論理回路の入力から出力までのすべてのパスについて、そのパスが通過する素子の遅延の和を計算する。この遅延は素子間をつなぐ配線の遅延も含むものとする。第 3 章で述べたように、オン抵抗は PMOS、NMOS それぞれで異なっている。また、素子の入力数が多くなるほど、どの入力に信号が入ったかによって素子の遅延時間が異なってくる。ここでは、素子の遅延時間の最大値と最小値があらかじめ定義されているものとして考える。あらかじめ定義された値を用いて、パスに含まれるすべての素子遅延を、最小の遅延時間によって計算したものをそのパスの最小遅延とし、min{ 立ち上がり時間, 立ち下がり時間 } と表すものとする。同様に、パスのすべての素子の遅延を、最大の遅延時間によって計算したものをそのパスの最大遅延とし、max{ 立ち上がり時間, 立ち下がり時間 } と表す。このようにしてすべての素子における最小・最大遅延を見積もる。

具体的に、図 4.2 に示す回路の遅延の見積もりの例を示す。ここで素子 i の立ち上がり時間を Di、立ち下がり時間を di とする。入力端子 A から出 OUT までのパスの最大遅延を D_A とすると、$D_A = max\{D3, d3\}$、同様に入力 B、C からのパスの最大遅延は $D_B = D_C = max\{D1, d1\}+max\{D2, d2\}+max\{D3, d3\}$、D からパスの最大遅延は $D_D = max\{D2, d2\}+max\{D3, d3\}$ となり、出力 OUT における最大遅延は $max\{D_A, D_B, D_C, D_D\}$ と見積もる。最小遅延も同様に $min\{D_A, D_B, D_C, D_D\}$ となる。

論理の関係やデータ依存によって実際には活性化しないパスがあったり、パス上のすべての素子が min{ 立ち上がり時間, 立ち下がり時間 } あるいは max{ 立ち上がり時間, 立ち

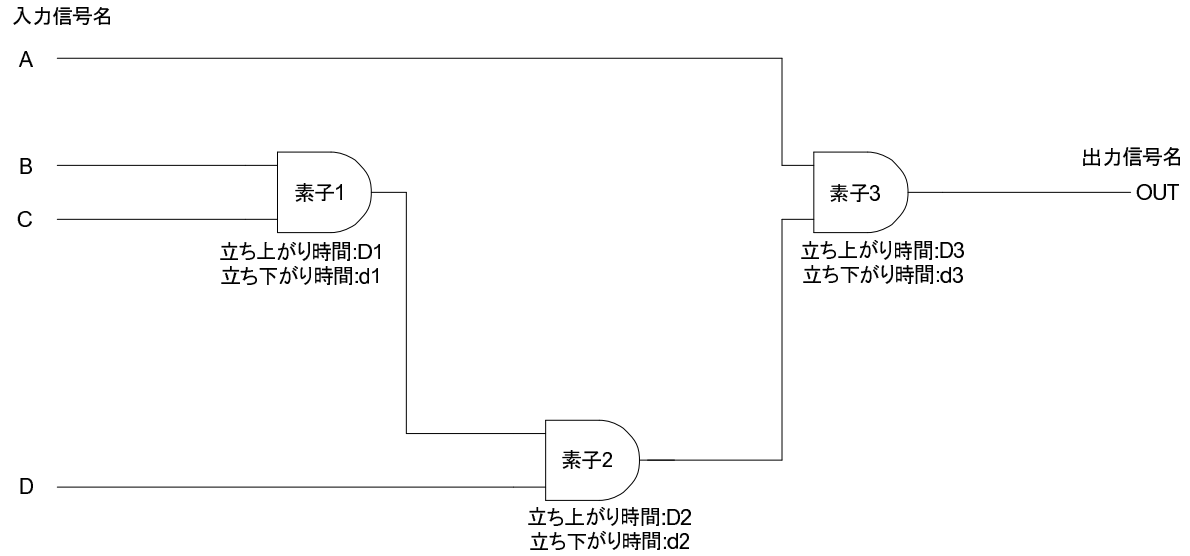


図 4.2: 遅延見積もりの例

下がり時間 } で動作することはないので、この見積もりは正確な値ではない。しかし、見積もられた最大遅延時間は十分に大きく、最小遅延時間は十分に小さい。よって遅延差の見積もりは十分に大きな値になり、これをもとに決めたサイクルタイムは正常な動作に十分な時間となる。

4.1.2 遅延バッファ挿入アルゴリズム

遅延バッファを挿入する場合、以下を満たすことが望ましい。

- 余分な遅延バッファを入れない
- すべての素子について遅延が揃えられる
- 挿入した遅延バッファの数に対して対象となる論理回路の遅延差の最大値が単調減少となる (遅延バッファ挿入の効果や遅延差が短縮していく傾向を見ることによって、挿入終了の判断がしやすくなるため。)

これらを考慮した、池田によって考案された遅延バッファ挿入アルゴリズムを図 4.3 に示す。

4.2 遅延バッファ挿入手法の問題点

文献 [1] では、遅延バッファ挿入による遅延差短縮の効果は、バッファ挿入による回路面積の増加が約 2.8 倍であるのに対して、遅延差を元の論理回路の遅延差の 1 / 4 弱に縮められているということであった。よってウェーブパイプラインで動作させると、動作クロック周波数は 4 倍となり、かなりの性能向上が見込めるという結果であった。しかし、

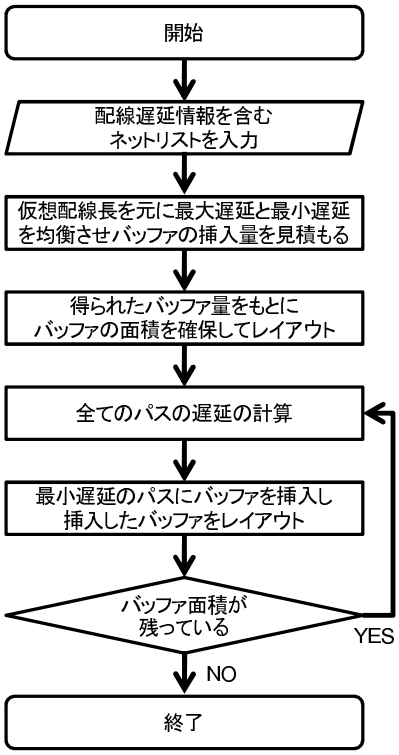
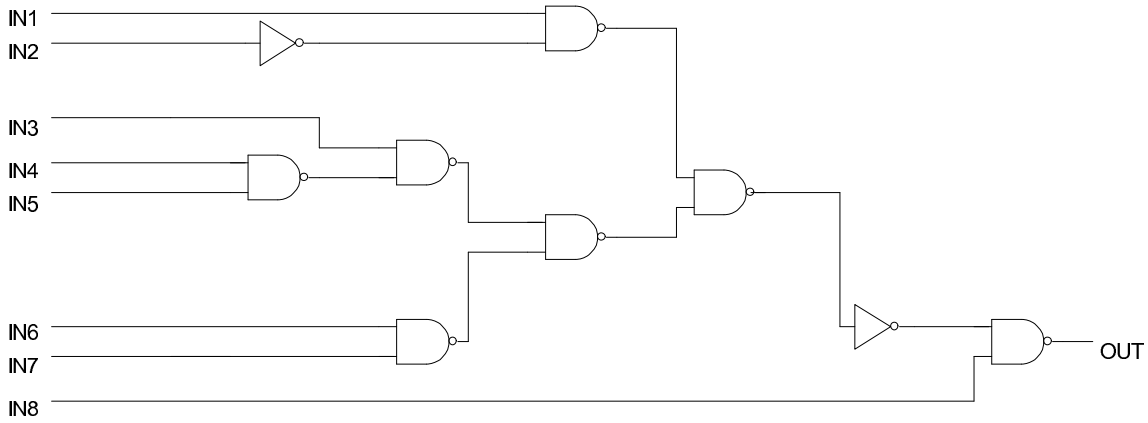


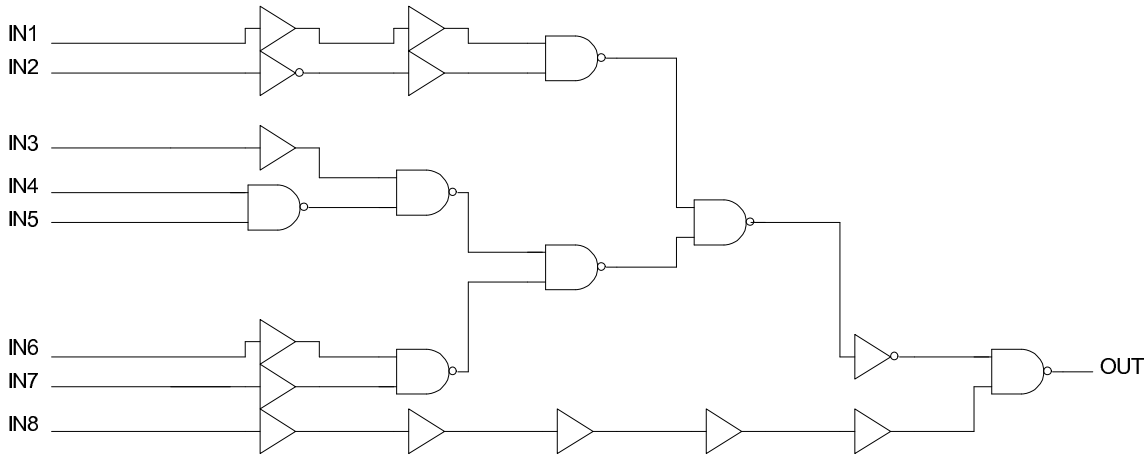
図 4.3: 遅延バッファ挿入アルゴリズムのフローチャート

この４倍という性能向上が面積増加のコストに見合っていないという結論に至っている。プロセッサなどの半導体によって生産される論理回路は、一般的に面積の指数乗で歩留まりが悪化する。よって、約３倍もの面積増加は、実際のプロセスにおける論理回路の歩留まりに非常に大きな悪影響を及ぼすことは明らかである。このように、遅延バッファ挿入による遅延差短縮では、面積増加に対する遅延差の短縮の割合が小さいため、実際にウェーブパイプラインを適用した論理回路を作製するメリットを十分に主張できない。

遅延バッファを挿入しても思う様に遅延差が縮まらない最大の原因は、論理合成法にあると考えられる。従来の設計法では、PARTHENON という通常のパイプラインの設計を前提とした CAD ツールを使用している。このため、論理合成は最大遅延時間を最小化、あるいは回路面積を最小化、または両方を考慮するという方針に基づいて行われる。PARTHENON を含めて、一般の CAD ツールには、このような方針に基づいて、論理を多段に構成し合成する多段論理合成が採用されている。図 4.4(a) に示すように、多段論理合成によって合成された論理回路では、論理が多段なので各パスの遅延はまちまちである。よって、図 4.4(b) に示すように、すべてのパスの遅延を遅延バッファ挿入によって揃えようとすると、膨大な量の遅延バッファが必要となってしまう。したがって、多段論理合成ではなく、面積を必要以上に増やすことなく理想的に遅延差を縮めることができる論理合成アルゴリズムが必要である。



(a) 遅延バッファ挿入前



(b) 遅延バッファ挿入後

図 4.4: 多段論理回路の例

第5章　遅延差短縮のための論理合成法

5.1　２段論理による論理合成

第 4 章で述べたように、従来の設計手法で用いられていた多段論理による論理合成では、遅延差短縮に限界が生じる。そこで本研究では論理の構成法として、２段論理を採用する。２段論理とは、図 5.1(a) のように、論理の１段目を AND ２段目を OR、または図 5.1(b) のように、１段目を OR ２段目を AND で構成する論理構成法である。前者を加法標準形と呼び、後者を乗法標準形と呼ぶ。

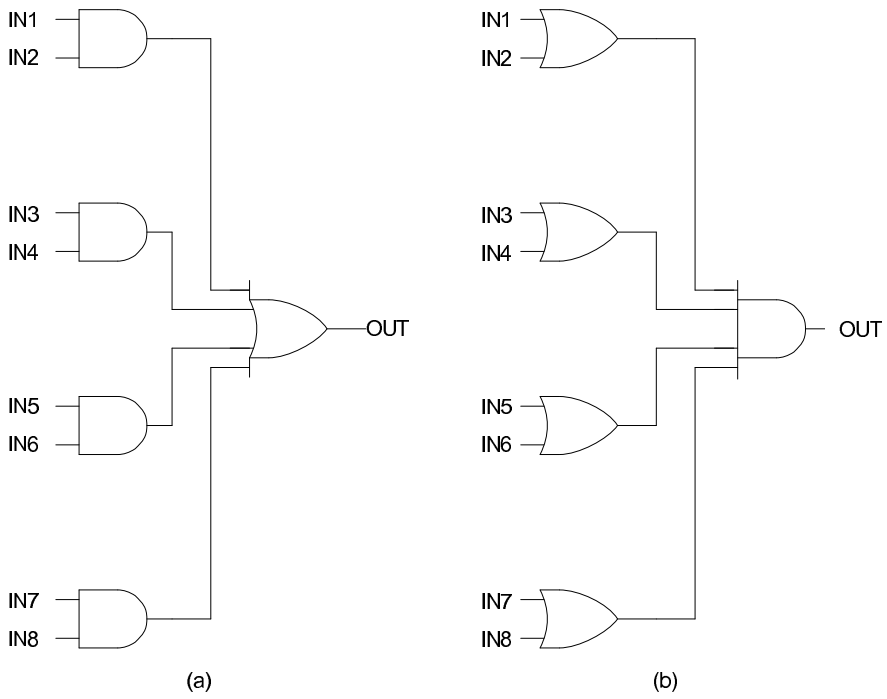


図 5.1: ２段論理の回路例

この図からわかるように、多段論理と違い、２段論理構成法では、各入力に対する論理段数が完全に揃っている。ただし、一般的に多段論理と比べて、２段論理では回路面積が大きくなってしまう。また、多入力の論理回路ほど２段目の論理素子が多入力になる。実際には、論理素子の入力数には上限があり、２段論理による論理回路の構成には限界がある。そのため、遅延差短縮のために２段論理を適用する場合、素子の入力数制限を回避するための工夫が必要となる。

本研究では、論理素子の入力制限に対して、最終段の論理素子の入力数が入力制限内に収まるまで論理を多段に拡張するという方針をとる。この対策によって、論理素子の入力制限に対応することが可能であるが、２段論理を多段に拡張するため、２段論理以上に回路面積が増加してしまう可能性が高い。このため、回路面積増加に対する何らかの対策が必要となるが、この問題に関しては、第 6 章で述べる。

5.2　クワイン・マクラスキー法

論理回路設計者が、ある機能を実現する論理を作製するとき、大規模な論理になると冗長な論理を取り除くことが難しくなってくる。一般的な CAD システムには論理を最適化する機能が実装されている。本研究では２段論理を論理合成の出発点とするため、一般的な CAD の多段論理による論理最適化ではなく、２段論理によって論理を最適化するアルゴリズムが必要となる。そこで、本研究では２段論理最適化のアルゴリズムとして、クワイン・マクラスキー法 [8] を採用する。このアルゴリズムを採用する理由としては、アルゴリズムが単純でわかりやすく、実装が容易であるという点が挙げられる。しかし、クワイン・マクラスキー法は、すべての組み合わせを調べて、最適解を見つけるアルゴリズムであるため、入力数が増えるに従って計算時間が指数乗で増加してしまうという欠点がある。図 5.2 にクワイン・マクラスキー法のフローチャートを示す。

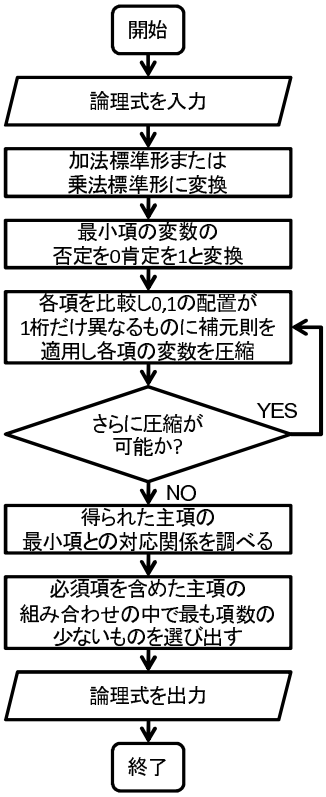


図 5.2: クワイン・マクラスキー法のフローチャート

図 5.2 に用いられている用語について解説する。最小項とは、加法標準形あるいは乗法標準形に変換された論理式の項のことであり、圧縮されていない元の項を表す用語である。補元則とは、ブール代数の基本法則の 1 つであり、以下の関係式のことである。

$$A \cdot \overline{A} = 0 \tag{5.1}$$

$$A + \overline{A} = 1 \tag{5.2}$$

式 5.1 は乗法標準形に変換した場合の変数圧縮に使用し、式 5.2 は加法標準形に変形した場合の変数圧縮に使用する。主項とは、圧縮された項のことである。主項の中で元の論理を構成するために必ず必要となる項を、必須項と言う。

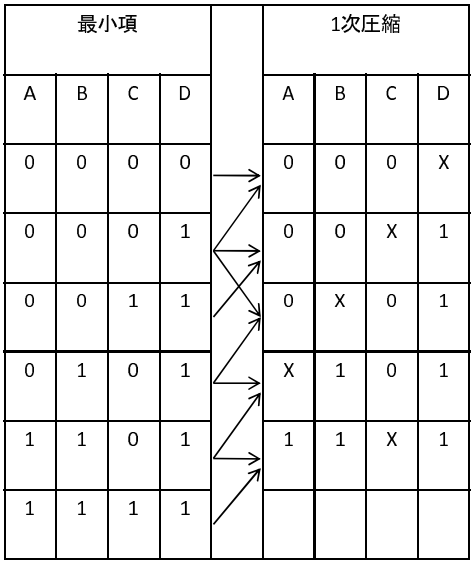


図 5.3: クワイン・マクラスキー法の変数圧縮の例

次に、図 5.2 に示したクワイン・マクラスキー法の手順について説明する。まず、入力した論理式をすべての最小項を含む 2 段論理表現に展開する。2 段論理の表現法は、加法標準形、乗法標準形どちらでもよい。次に、すべての最小項の変数に対して、否定を 0、肯定を 1 に変換する。その後、変換した各項を比較し、0、1 の配置が 1 桁だけ異なる項に補元則を適用し、変数を圧縮する。変数圧縮の例を図 5.3 に示す。この例では、1 次圧縮までで圧縮が終了しているが、1 次圧縮された項がさらに圧縮できるようであれば 2 次圧縮、3 次圧縮と同様に繰り返していく。圧縮の作業が終了すると、図 5.3 の矢印で示された主項と最小項との対応関係を全ての項について調べる。この中で必須項を含めた主項の組み合わせで最も項数の少ないものを選び出す。以上がクワイン・マクラスキー法の論理式最適化の流れである。

クワイン・マクラスキー法の具体的な実装方法については第 6 章で述べる。

5.3 論理段数を揃えるアルゴリズム

5.1 節で述べたように、大規模な論理回路を単純に 2 段論理により構成するのは、論理素子の入力制限から難しい。そこで、本研究では論理素子の入力制限に対して、最終段の論理素子の入力数が入力制限内に収まるまで論理を多段に拡張する。クワイン・マクラスキー法で最適化された論理をそのまま多段に拡張するのではなく、NAND 表現に変換した論理を入力として、素子の入力制限に応じた論理の多段化を行う。このような変換を行う理由は、実際の論理設計においては、すべて NAND 素子と NOT を表現するインバー

タによって回路を構成するためである。この方法によって、多入力１出力の論理の段数を揃えることが可能となる。

論理の多段化の例を図 5.4 に示す。図 5.4 は、ド・モルガンの定理によって、以下の様な式変形を行ったものと等価である。

$$\overline{\overline{ab}\cdot\overline{cd}\cdot\overline{ef}\cdot\overline{gh}\cdot\overline{ij}} = \overline{\overline{\overline{ab}\cdot\overline{cd}\cdot\overline{ef}\cdot\overline{gh}\cdot\overline{ij}}} = \overline{\overline{\overline{ab}\cdot\overline{cd}\cdot\overline{ef}}+\overline{\overline{gh}\cdot\overline{ij}}} = \overline{\overline{\overline{ab}\cdot\overline{cd}\cdot\overline{ef}}\cdot\overline{\overline{gh}\cdot\overline{ij}}} \tag{5.3}$$

ド・モルガンの定理とは、以下に示すような関係式のことである。

$$\overline{A\cdot B} = \overline{A} + \overline{B} \tag{5.4}$$

$$\overline{A+B} = \overline{A}\cdot\overline{B} \tag{5.5}$$

ド・モルガンの定理によって変形した式 5.3 を元に、図 5.4 の様に、図 5.4(a) の回路を図 5.4(b) の様に変形し、論理の多段化を行う。

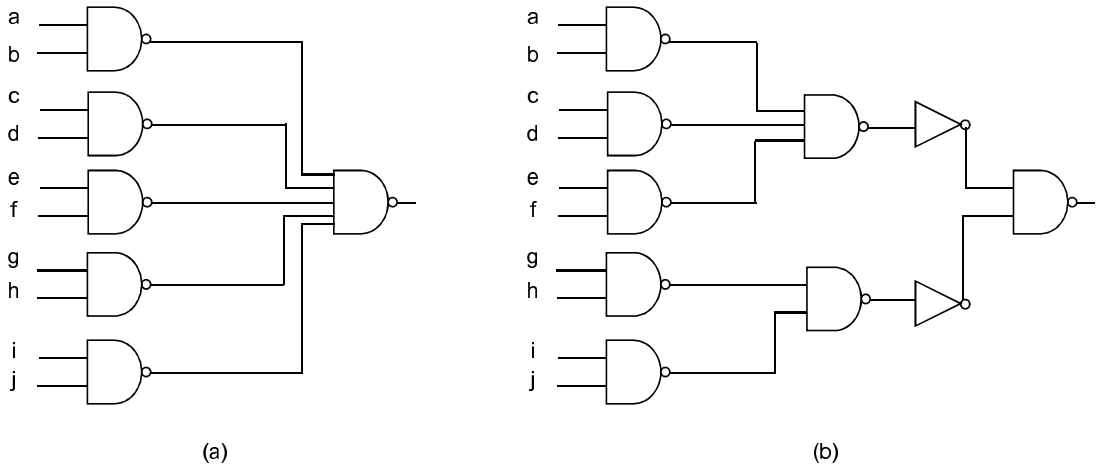


図 5.4: 論理の多段化の例

実際の論理回路においては、多入力多出力の論理回路が一般的である。本提案手法においては、多入力多出力の論理回路の段数を揃える場合、論理の出力ごとの回路に分けて考える。図 5.5 に 4 入力 2 出力の論理回路の段数調整の例を示す。もし、この回路を 2 入力 NAND で構成したいと考えたとき、図 5.5(a) の出力 X の 4 入力 NAND は入力数制限違反ということになる。論理段数を多段にしてすべて 2 入力 NAND 素子で構成すると図 5.5(b) の回路になる。このように構成すると、出力 Y の論理段数が出力 X よりも少なくなってしまうので、図 5.5(c) のように出力 Y にインバータを 2 個接続して論理段数を出力 X に合わせる。このような方法によって、多入力多出力の論理回路の論理段数を揃える。

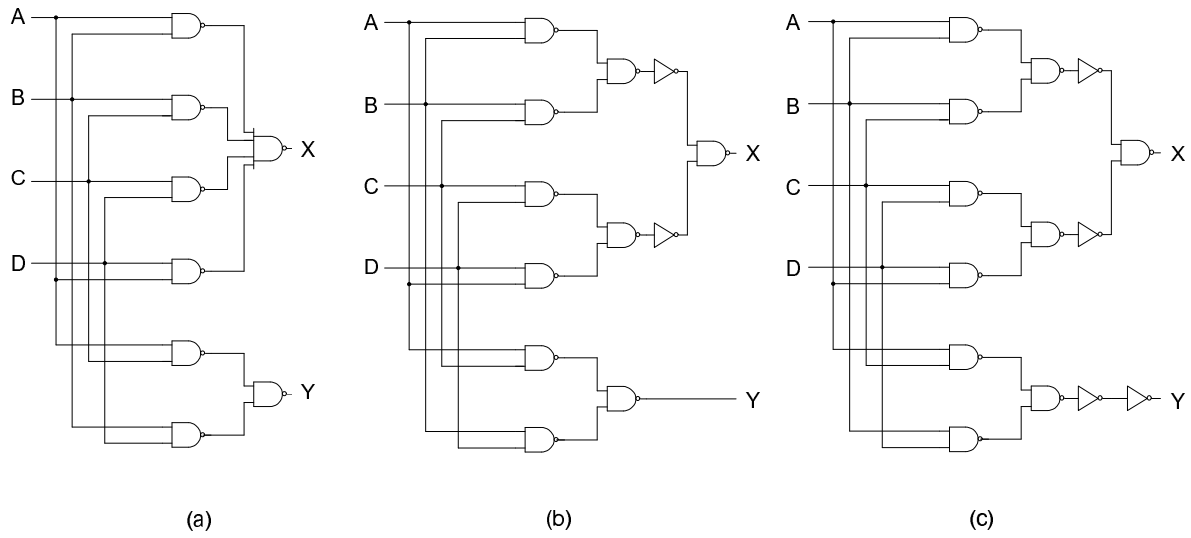


図 5.5: 論理回路の段数調整の例

図 5.6 に提案する論理合成の流れを示す。クワイン・マクラスキー法により最適化された論理を NAND 表現に変換し、入力数制限に応じて多段化することで、従来の設計手法と異なり、合成後の遅延バッファ挿入なしで遅延差が縮めることができると考えられる。

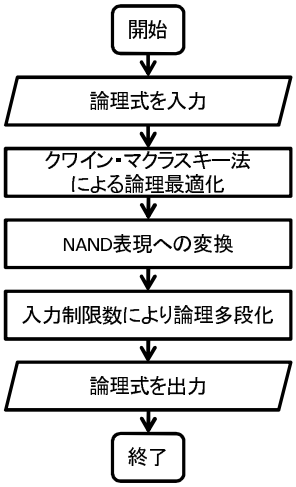


図 5.6: 提案手法のフローチャート

第6章　提案手法の実装とその評価

6.1　クワイン・マクラスキー法の実装

提案手法を評価するために、プログラムの実装を行った。本研究ではプログラミング言語に Common Lisp を採用した。LISP とは、LISt Processing の略であり、今日用いられている計算機言語の中で 2 番目に古い言語である [9]。Common Lisp は数多くの Lisp 方言の中の 1 つであり、その中で最も標準的な LISP 言語の 1 つである。本研究では、論理回路を合成した時の接続情報をリスト形式で表わすことが効率が良いと考え、リスト処理を得意とする Common Lisp によって実装を行った。ここでは、提案手法の実装法と、実装したシステムによる評価について述べる。

まず、第 5 章で説明したクワイン・マクラスキー法の実装について述べる。実装したクワイン・マクラスキー法は、図 5.2 のフローチャートよりも若干簡単化したものとなっている。

まず入力する論理式は真理値表によって与えるものとした。真理値表を入力とすることで、すべての最小項を含む 2 段論理表現に展開したことと等価になり、変換の手間がなくなる。また、最小項の変数の 0、1 への変換の手間もなくなる。よって、入力した値を用いてすぐに変数の圧縮作業を行うことができる。圧縮においては、圧縮対象となる項すべての組み合わせを調べる。圧縮できるかどうかの判断は、比較する項同士の排他的論理和を行うことによって判断する。排他的論理和とは、A、B の入力があった場合、どちらかが異なる値であった場合にのみ出力が 1 となる論理であり、この性質を利用して、各項を比較して 1 桁だけ値が 1 となる項を探し出し、1 が出力された桁を圧縮する。この操作を圧縮ができなくなるまで続ける。

次に、変数の圧縮が終了すると最小項と主項の対応関係を調べる。この処理を行うために、各最小項に番号を割り当てる。そして各主項と最小項との対応関係を調べて、各主項は対応している最小項の番号を記憶しておく。最後に、各主項同士のすべての組み合わせを調べて、各主項同士の番号を照らし合わせてすべての最小項の番号が揃う組み合わせを記憶する。記憶された結果の中で最も少ない項で構成されている論理を選びだして終了となる。

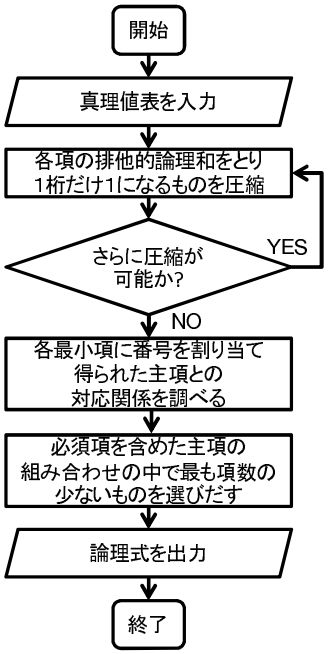


図 6.1: 実装におけるクワイン・マクラスキー法のフローチャート

なお、本研究で実装したクワイン・マクラスキー法の論理変数の入力数は９入力までとした。この程度の入力数であれば、現在の計算機ならば十分現実的な時間で処理を行うことが可能である。入力数の拡張は容易に行えるが、入力数が１０以上になると、計算に長時間を要するため、このような制限を設けた。

6.2 論理段数を揃えるアルゴリズムの実装

次に、論理段数を揃えるアルゴリズムの実装について説明する。図 6.2 に、論理段数を揃えるアルゴリズムの実装におけるフローチャートを示す。

まず、入力された論理式を NAND 表現に変換する。この処理については、式 5.4 及び式 5.5 に示したド・モルガンの定理を用いて単純に NAND の表現へ変形する。式 6.1 に、加法標準形で表わされた論理式を NAND 表現へ変換する例を示す。

$$ab + cd + ef = \overline{\overline{ab + cd + ef}} = \overline{\overline{ab} \cdot \overline{cd} \cdot \overline{ef}} \tag{6.1}$$

次に、NAND 表現に変換された論理式を、入力制限数によって多段化する。論理の多段化を行うにあたって、本研究では、論理回路の入力部分のインバータによる論理否定以外は、すべて同じ入力数の NAND 素子を使用して構成する。第 3 章で述べたように、CMOS の素子は入力数によって素子自体の遅延時間が異なっている。よって、１つの論

理回路で様々な素子を使用すると遅延がまちまちになり遅延差が増加してしまう可能性が高くなる。したがって、同じ入力数の NAND 素子を使用することで遅延のばらつきを抑えるという方針をとる。

同じ入力数の NAND 素子を使用して回路を構成する場合、より多入力の NAND 素子を用いることで入力制限が緩くなるので、多段化を行う確率が減ってくる。しかし、NAND 素子そのものの遅延は、多入力になるほど増加してしまう。つまり、遅延を最優先に考えると 2 入力 NAND によって回路を構成するのが最も良いが、最も入力制限が厳しくなるので、より回路を多段化して構成しなければならなくなる。本研究では、入力数の制限と遅延時間を考慮し、3 入力 NAND を使用して回路を構成する。本研究で作製したシステムでは、入力された論理式に 4 入力以上の NAND 素子があった場合、3 入力 NAND 素子によって論理を多段化して構成する。多入力多出力の論理の場合、この論理の多段化を第 5 章で述べたように、各出力の論理ごとに行う。

最後に各出力の論理段数を揃える。具体的には、多段化を行った各出力の論理の中で、最も論理段数の多いものに他の出力の論理段数を揃える。第 5 章の図 5.5 の例では、段数の調整のためにインバータを挿入していたが、本研究で作製したシステムでは、入力値を反転させるインバータ以外のすべての素子を 3 入力 NAND で構成する。これによって、各入力に対する論理段数のばらつきと、論理素子の遅延のばらつきの両方を抑えることができる。

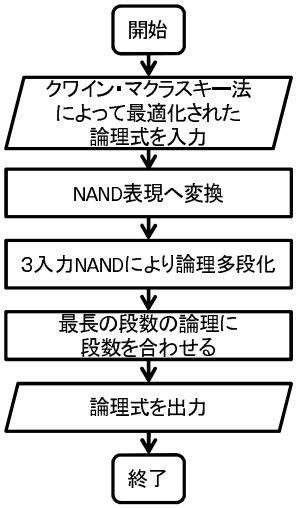


図 6.2: 実装における論理段数を揃えるアルゴリズムのフローチャート

6.3　作製した論理合成システムによる評価

作製した論理合成システムによって、回路を作製し提案手法の評価を行った。評価用の回路の選定においては、提案手法の効果がわかりやすく現れる様に、各入力から各出力に対する論理段数にばらつきがある論理回路が良いと考える。この条件を満たす回路として、本研究では７セグメントデコーダを採用した。

７セグメントとは、電卓や時計などの表示部に用いる表示部品で、７つのセグメントのON/OFF で数字や一部のアルファベットを表示する。それぞれのセグメントを慣例としてa〜g と呼ぶ。７セグメントデコーダは、入力された BCD コード[1]や１６進数の値を７セグメントで表示できるようにデコードする。本研究で用いる７セグメントデコーダは、入力を BCD コード、出力をセグメントの a〜g として０〜９をデコードし、それ以外の入

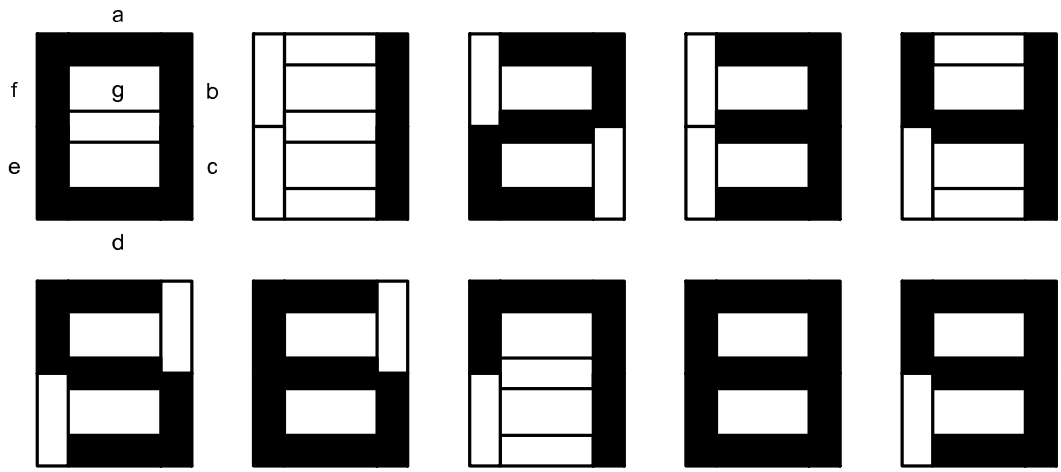


図 6.3: ７セグメントの表示例

[1]BCD コードは２進化１０進符号とも呼び、Binary Coded Decimal の略であり、ディジタル回路で１０進数を取り扱うとき頻繁に用いられる。

表 6.1: 7セグメントデコーダの真理値表

IN3	IN2	IN1	IN0	SEG-a	SEG-b	SEG-c	SEG-d	SEG-e	SEG-f	SEG-g
0	0	0	0	0	1	1	1	1	1	1
0	0	0	1	0	0	0	0	1	1	0
0	0	1	0	1	0	1	1	0	1	1
0	0	1	1	1	0	0	1	1	1	1
0	1	0	0	1	1	0	0	1	1	0
0	1	0	1	1	1	0	1	1	0	1
0	1	1	0	1	1	1	1	1	0	1
0	1	1	1	0	1	0	0	1	1	1
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1
1	0	1	0	1	1	1	1	0	0	1
1	0	1	1	1	1	1	1	0	0	1
1	1	0	0	1	1	1	1	0	0	1
1	1	0	1	1	1	1	1	0	0	1
1	1	1	0	1	1	1	1	0	0	1
1	1	1	1	1	1	1	1	0	0	1

“10” 以上は “E” を表示

6.3.1 論理合成による回路作製

本研究では、論理合成の段階で論理段数の等しい論理回路を出力することで、遅延を揃えることを目的とする。よって多段論理によって合成された論理回路よりも遅延差が縮まっていることが予想される。具体的に示すために、提案手法の比較対照として、SIS という多段論理合成ツールを採用した。SIS(Sequential Interactive System) はカルフォルニア大学バークレー校で開発された多段論理合成ツールである [10]。Synopsys 社の Design Compiler に代表される商用論理合成ツールの基本的な仕組みは SIS と同様であると考えてよい。

SIS によって論理合成した 7 セグメントデコーダを図 6.4 に示す。また、提案手法によって論理合成した 7 セグメントデコーダを図 6.5 に示す。2 つの図を見比べると図 6.4 では、論理が多段に構成されており、図 6.5 では、論理段数がきちんと揃えられていることがわかる。単純に論理素子の数を見比べると多段論理の方が論理素子数が少ないことがわかる。図 6.5 の回路を見ると、冗長な入力数の 3 入力 NAND が非常に多いことがわかる。これは、論理回路中に無駄なトランジスタが存在していることを意味している。よって、必要以上に論理回路面積が増加していることになる。この冗長な素子を最適な入力数に置き換えることによって、必要以上の面積増加を防ぐことを考える。冗長な素子を最適な入力数に置き換えた、提案手法の 7 セグメントデコーダを図 6.6 に示す。このように提案手法では、冗長な素子の置き換えによって面積増加を防ぐという方針をとる。

なお、この提案手法のシステムと SIS は、ともに回路の接続情報を出力するシステムなのでこの論理回路図は、出力された論理回路接続情報を元に手動で作製した。また、7 セグメントデコーダの入力側には、各入力の遅延がインバータ一段分の遅延に揃うように、トランジスタサイズを調節したインバータを余分に配置している。

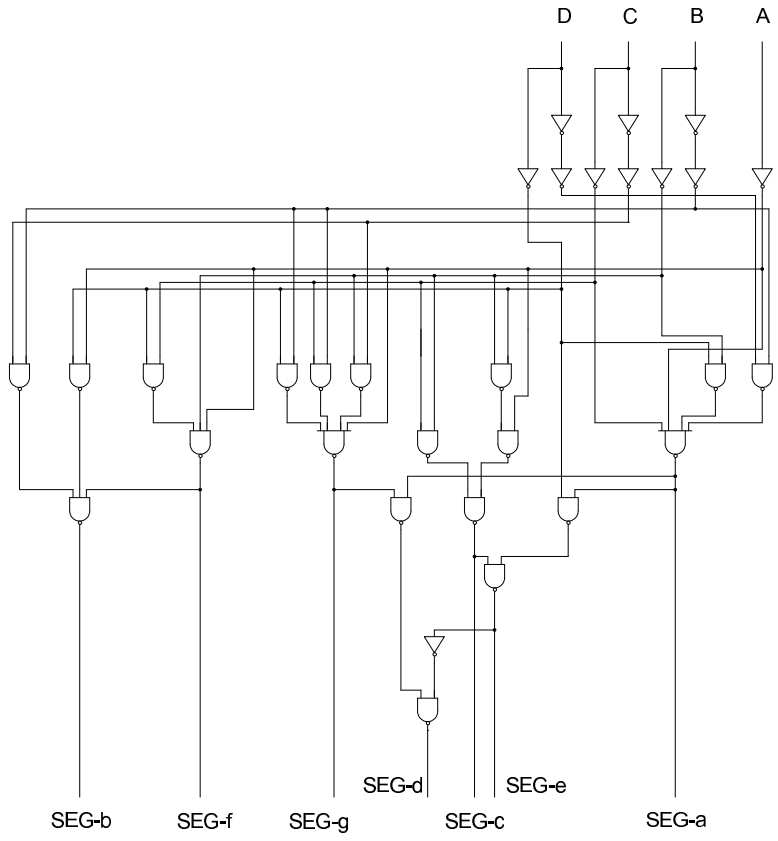


図 6.4: SIS によって論理合成した 7 セグメントデコーダ

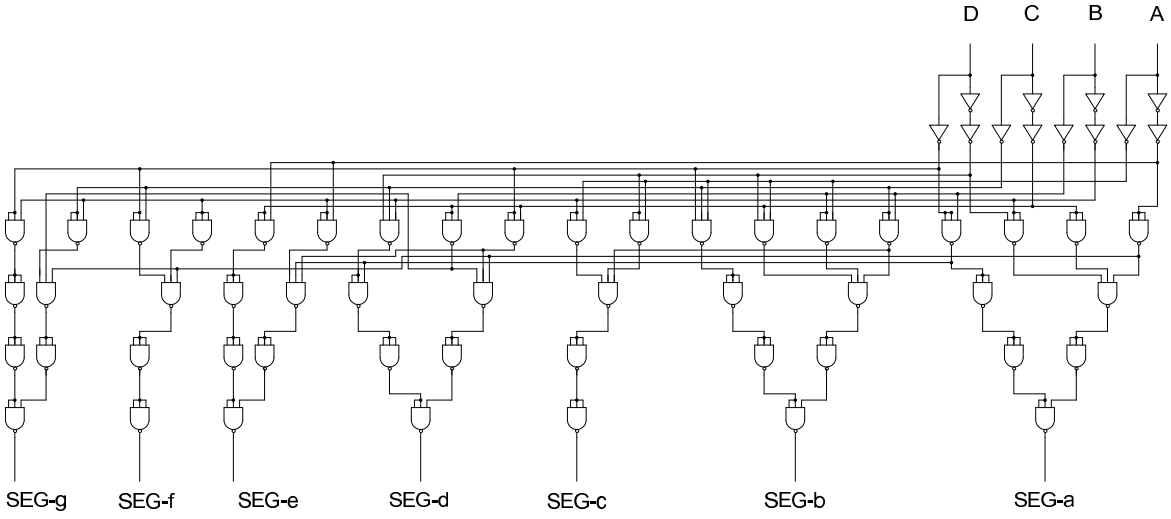


図 6.5: 提案手法によって論理合成した 7 セグメントデコーダ

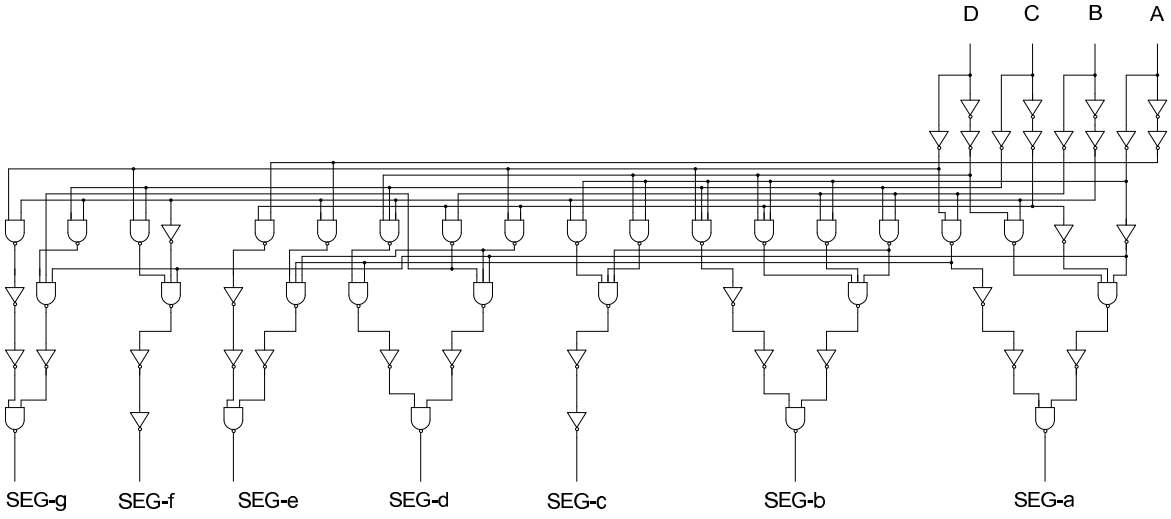


図 6.6: 冗長な素子を置き換えた提案手法の 7 セグメントデコーダ

6.3.2 評価用セルモデルの作製

提案手法と SIS で作製した論理回路を評価するために、評価用のセルモデルを作製した。セルモデルの作製においては、文献 [11] のインバータのセルモデルのパラメータを用いた。設計に用いたインバータのモデルのパラメータを、表 6.2 に示す。このパラメータは、$W = 0.2\mu m$、$L = 0.1\mu m$ の $0.1\mu m$ プロセスルールに基づいて設計されている。

表 6.2: 設計に用いたインバータのモデルのパラメータ

ゲート	オン抵抗(kΩ)		入力容量 Cg(fF)	拡散容量 Cd(fF)	面積 (μm²)
	pull-up	pull-down			
inv	27.4	19.8	0.585	0.462	0.84

本研究では、２つの評価用セルモデルを設計した。１つは提案手法の効果がわかりやすく現れる様に、論理素子の最大遅延と最小遅延をほぼ等しい値としている理想的なモデルである。この素子モデルのパラメータを表 6.3 に示す。もう一方は、現実的な論理素子遅延を盛り込んで、できる限り論理素子の最大遅延と最小遅延を縮める設計を行った現実的なモデルである。この素子モデルのパラメータを表 6.4 に示す。両方とも $0.1\mu m$ プロセスルールを想定している。

各素子の名称は、NAND ２は２入力 NAND を表し、以下 NAND ３、NAND ４はそれぞれ３入力、４入力を表している。$INV\ 3_{=NAND\ 3}$ は、３入力 NAND にオン抵抗を近づけた論理素子モデルである。このように下の添え字は、どの素子モデルのオン抵抗に近づけたかを表すものとする。また、$INVinCg\ 3_{=INVinCg\ 7}$ の様に、INVin と表記されている論理素子は、７セグメントデコーダの入力のインバータを表すものとする。Cg3 の部分は、その素子の次段の Cg を３個駆動する素子であることを表し、この素子のオン抵抗を INVinCg ７に近づけた論理素子であることを表す。以下 Cg の次の数字はその素子の次段の Cg の駆動個数を表すものとする。

表 6.4 に示した現実的なモデルの設計は、第 3 章で説明した MOSFET モデルに基づいている。各論理素子は、素子自身の遅延差を最小にするように PMOS と NMOS のゲート幅 W の比率を決定している。また、ゲートの入力容量は第 3 章の式 3.2 を見るとわかるように、$W \times L$ の値で決まる。今回作製したセルモデルでは、$W \times L$ の値を常に一定にして、すべての論理素子のゲートの入力容量を合わせている。よって各論理素子によって、ゲート長 L の値も異なっている。このように、ゲート長 L とゲート幅 W を調節することで、各論理素子のオン抵抗を調整している。

表 6.3: 評価用セルの理想的モデル

	Ron−max[kΩ]	Ron−min[kΩ]	Cg[fF]	Cd[fF]	面積[μm^2]
INV	4.720	4.720	2.920	2.310	2.52
NAND2	5.341	5.326	2.920	2.901	3.90
NAND3	6.855	6.841	2.920	2.926	5.28
NAND4	8.616	8.604	2.920	2.862	6.66
$INV_{=NAND3}$	8.612	8.588	2.920	1.711	2.60
$NAND2_{=NAND3}$	7.440	7.415	2.920	2.460	4.03
$INVinCg3_{=INVinCg7}$	10.491	10.462	2.920	1.550	2.64
$INVinCg4_{=INVinCg7}$	7.986	7.964	2.920	1.777	2.59
$INVinCg5_{=INVinCg7}$	6.469	6.451	2.920	1.974	2.56
$NAND3Cg3_{=NAND3Cg4}$	8.907	8.902	2.920	2.567	5.38
$NAND3Cg2_{=NAND3Cg4}$	12.491	12.484	2.920	2.167	5.53
$NAND3_{=NAND3Cg4}$	21.960	21.948	2.920	1.635	5.85
$INV_{=INVCg4}$	15.825	15.781	2.920	1.262	2.72
$NAND2Cg3_{=INVCg4}$	5.668	5.650	2.920	2.818	3.91
$NAND2Cg2_{=INVCg4}$	8.083	8.057	2.920	2.360	4.01
$NAND3_{=INVCg4}$	13.052	13.045	2.920	2.120	5.55
$NAND2_{=INVCg4}$	14.022	13.976	2.920	1.792	4.20
$INVCg1_{=INVCg6}$	23.711	23.645	2.920	1.031	2.82
$INVCg2_{=INVCg6}$	13.022	12.985	2.920	1.392	2.68
$INVCg4_{=INVCg6}$	6.919	6.899	2.920	1.909	2.57
$INVCg5_{=INVCg6}$	5.614	5.599	2.920	2.119	2.54

表 6.4: 評価用セルの現実的モデル

	Ron−max[kΩ]	Ron−min[kΩ]	Cg[fF]	Cd[fF]	面積[μm^2]
INV	4.720	4.720	2.920	2.310	2.52
NAND2	6.700	3.350	2.920	3.255	3.90
NAND3	8.684	2.895	2.920	3.768	5.28
NAND4	10.682	2.671	2.920	3.495	6.66
$INV_{=NAND3}$	13.497	13.459	2.920	1.367	2.69
$NAND2_{=NAND3}$	10.468	5.234	2.920	2.604	4.02
$INVinCg3_{=INVinCg7}$	10.491	10.462	2.920	1.550	2.64
$INVinCg4_{=INVinCg7}$	7.986	7.964	2.920	1.777	2.59
$INVinCg5_{=INVinCg7}$	6.469	6.451	2.920	1.974	2.56
$NAND3Cg3_{=NAND3Cg4}$	11.088	3.696	2.920	3.334	5.37
$NAND3Cg2_{=NAND3Cg4}$	15.592	5.197	2.920	2.812	5.53
$NAND3_{=NAND3Cg4}$	26.290	8.763	2.920	2.165	5.81
$INVCg4_{=NAND2Cg3}$	5.822	5.806	2.920	2.081	2.55
$NAND2Cg2_{=NAND2Cg3}$	9.328	4.664	2.920	2.758	3.99
$NAND3_{=NAND2Cg3}$	13.568	4.523	2.920	3.014	5.46
$NAND2_{=NAND2Cg3}$	16.096	8.048	2.920	1.792	2.10
$INVCg1_{=INVCg6}$	23.711	23.645	2.920	1.031	2.82
$INVCg2_{=INVCg6}$	13.022	12.985	2.920	1.392	2.68
$INVCg4_{=INVCg6}$	6.919	6.899	2.920	1.909	2.57
$INVCg5_{=INVCg6}$	5.614	5.599	2.920	2.119	2.54

6.3.3 評価用セルモデルを用いた評価

設計したセルモデルを用いて評価を行った。評価対象となる値は、遅延差と面積である。また、多段論理合成で作製した論理回路の最大遅延と、提案手法を用いて作製した論理回路の遅延差を比較することで、動作クロック周波数の増減を見積もる。

まず、提案手法の効果がはっきり現れる理想的モデルによる評価の結果を示す。表 6.5 に理想的セルモデルによる遅延の評価結果を示し、表 6.6 に理想的セルモデルによる面積の評価結果を示す。表 6.5 で太い枠で示されている部分が、各論理回路における最大遅延と最小遅延である。

表 6.5: 理想的セルモデルによる遅延の評価結果

	提案手法 (最大遅延)[ps]	提案手法 (最小遅延)[ps]	面積最適化した提案手法 (最大遅延)[ps]	面積最適化した提案手法 (最小遅延)[ps]	多段論理合成 (最大遅延)[ps]	多段論理合成 (最小遅延)[ps]
seg_a	333.149	307.885	279.129	252.516	224.774	168.482
seg_b	332.373	307.012	277.997	252.573	185.244	144.617
seg_c	332.373	307.012	277.714	252.199	187.130	155.615
seg_d	333.149	307.012	279.129	252.516	327.134	189.811
seg_e	333.067	307.012	279.168	252.516	286.954	155.603
seg_f	333.149	307.025	278.370	252.199	165.187	133.602
seg_g	333.149	307.025	279.129	252.681	199.768	143.358
遅延差	26.137		26.969		193.532	

表 6.6: 理想的セルモデルによる面積の評価結果

	提案手法 (面積)[μm^2]	面積最適化した提案手法(面積)[μm^2]	多段論理合成 (面積)[μm^2]
seg_a	48.76	30.17	14.46
seg_b	49.48	36.14	13.08
seg_c	27.54	18.80	15.60
seg_d	43.31	31.82	10.32
seg_e	38.10	25.22	7.80
seg_f	27.54	17.32	9.18
seg_g	38.10	25.22	18.36
INV(input)	30.67	30.67	25.80
total	303.49	215.34	114.60

表 6.5 より、提案手法は各セグメントの遅延がほぼ揃っていることがわかる。面積を最適化した提案手法の論理回路においても、同様である。これに対して、SIS によって多段論理合成された論理回路は、遅延がまちまちになっている。また、最大遅延は、面積を最適化していない提案手法の回路と、多段論理合成の回路がほぼ同じ値になっている。本来、論理段数だけで比較すると多段論理の方が論理段数が多いので回路の最大遅延も大きくなるはずである。そうならない理由は、提案手法の回路に遅延の大きい論理素子を用いているためである。提案手法の回路では、遅延の大きい論理素子を意図的に入れることによって、遅延差の増加を防いでいる。面積を最適化した提案手法の回路においても、同様に遅延差の増加を防いでいるが、こちらの方が遅延時間そのものが短くなっている。これは、3 入力の NAND よりも遅延の短い 2 入力 NAND とインバータが、論理回路の構成に含まれるため全体の遅延時間も短くなる。

この論理回路では、提案手法と多段論理合成ともに、次段の入力容量 Cg が複数に接続されている多出力の論理素子がいくつか存在する。次段の入力容量が増加すると、第 3 章の式 3.7 から、遅延時間が増加してしまう。よって、提案手法の論理回路では、このような多出力の素子の遅延時間に、他の素子の遅延時間を合わせるように論理素子を設計し配置した。表 6.3 の INV の値を用いて、式 3.7 により遅延時間を求めると、約 24.686[ps] となるので、表 6.5 より、提案手法の論理回路の遅延差は、ほぼインバータ 1 段分の遅延差に抑えることができている。

表 6.5 に示される様に、素子のオン抵抗の最大値と最小値が揃っている、理想的な論理素子によって動作させることができれば、提案手法では理想的に遅延差を縮めることが可能になる。具体的な数字を見ると、理想的な論理素子による評価では、提案手法と多段論理合成の遅延差を比較すると、面積最適化前の提案手法が 26.137[ps] に対して、多段論理合成では、193.532[ps] であり、$193.532/26.137 \simeq 7.405$ より、遅延差は 7 分の 1 以下に短縮されている。また、多段論理の最大遅延が 327.134[ps] であるので、$327.134/26.137 \simeq 12.516$ より、ウェーブ化によって通常のパイプライン動作よりも約 12.5 の動作周波数の向上が見込まれる。面積を最適化した提案手法と、多段論理合成との遅延差の比較においても、面積を最適化した提案手法が 26.969[ps] であるので、$193.532/26.969 \simeq 7.176$ より、遅延差は 7 分の 1 以下に短縮されている。ウェーブ化による動作周波数は、$327.134/26.137 \simeq 12.130$ より、約 12.1 倍通常のパイプライン動作より向上することになる。したがって、面積を最適化しても性能はほとんど低下しないという結果が得られている。

提案手法を用いることによる回路面積の増加分は、表 6.6 より、面積最適化をしなければ $303.49/114.60 \simeq 2.65$ より約 2.65 倍に増加するが、面積を最適化することによって、$215.34/114.60 \simeq 1.88$ より約 1.9 倍の増加にとどまり、面積増加以上の性能増加が期待できる結果となっている。

次に、現実的なセルモデルを用いた評価の結果を示す。表 6.7 に現実的セルモデルによる遅延の評価結果を示し、表 6.8 に現実的セルモデルによる面積の評価結果を示す。表 6.7 で太い枠で示されている部分が、各論理回路における最大遅延と最小遅延である。

表 6.7: 現実的セルモデルによる遅延の評価結果

	提案手法 (最大遅延)[ps]	提案手法 (最小遅延)[ps]	面積最適化した提案手法 (最大遅延)[ps]	面積最適化した提案手法 (最小遅延)[ps]	多段論理合成 (最大遅延)[ps]	多段論理合成 (最小遅延)[ps]
seg_a	415.877	201.250	312.353	202.121	259.797	118.543
seg_b	414.716	201.111	317.021	188.710	226.195	123.880
seg_c	414.716	201.111	309.212	199.792	217.978	134.982
seg_d	415.877	201.111	316.985	198.261	389.036	142.335
seg_e	415.683	201.250	316.678	201.972	342.542	134.970
seg_f	415.877	201.111	300.095	199.792	193.473	112.971
seg_g	415.877	201.111	316.678	200.045	228.758	110.744
遅延差	214.766		128.311		278.292	

表 6.8: 現実的セルモデルによる面積の評価結果

	提案手法 (面積)[μm^2]	面積最適化した提案手法(面積)[μm^2]	多段論理合成 (面積)[μm^2]
seg_a	48.68	27.97	14.46
seg_b	49.36	33.91	13.08
seg_c	27.47	14.52	15.60
seg_d	43.26	31.67	10.32
seg_e	38.03	21.11	7.80
seg_f	27.47	14.97	9.18
seg_g	38.03	21.11	18.36
INV(input)	30.67	30.67	25.80
total	302.96	195.92	114.60

表 6.7 より、理想的なセルモデルの場合と同様に、提案手法は各セグメントの遅延がほぼ揃っていることがわかる。面積を最適化した提案手法の論理回路においては、理想的なセルモデルの場合よりも若干ばらつきが生じているが、遅延はほぼ揃っている。これに対して、SIS によって多段論理合成された論理回路は、理想的なセルモデルの場合と同様に、遅延がばらついている。

現実的なセルモデルでは、論理素子 1 個当たりの最大遅延が増加しているため、提案手法と多段論理合成ともに最大遅延が増加している。また、論理素子 1 個当たりの最小遅延は減少しているため、最小遅延は減少している。このため、表 6.7 より、提案手法と多段論理合成の遅延差の値がかなり近づいている。現実的なセルモデルでの評価では、素子自体の遅延差が 1 段ごとに加算される。提案手法では、3 入力 NAND によって回路を構成しているため、3 入力 NAND 自体の遅延差が 1 段ごとに加算されている。よって、現実的なセルモデルでは 3 入力 NAND 自体の遅延差がそのまま結果に表れている。面積を最適化した提案手法の論理回路では、面積最適化前の論理回路よりも、遅延差が縮まっている。これは、理想的なセルモデルを用いた場合と同様に、冗長な 3 入力 NAND の置き換えによるものである。

表 6.7 に示されるように、現実的な論理素子による評価では、提案手法と多段論理合成の遅延差を比較すると、面積最適化前の提案手法が 214.766[ps] に対して、多段論理合成では、278.292[ps] であり、$278.292/214.766 \simeq 1.296$ より、遅延差は 2 割ほど短縮されている。また、多段論理の最大遅延が 389.036[ps] であるので、$389.036/214.766 \simeq 1.811$ より、ウェーブ化によって通常のパイプライン動作よりも約 1.8 の動作周波数の向上が見込まれる。面積を最適化した提案手法と、多段論理合成との遅延差の比較においては、面積を最適化した提案手法が 128.311[ps] であるので、$278.292/128.311 \simeq 2.169$ より、遅延差は 2 分の 1 以下に短縮されている。また、ウェーブ化による動作周波数は、$389.036/128.311 \simeq 3.032$ より、約 3 倍通常のパイプライン動作より向上することになる。現実的なセルモデルを用いた評価では、冗長な 3 入力 NAND を取り除くことによって性能向上が見込まれるという結果となった。

表 6.8 より、提案手法を用いることによる回路面積の増加分は、面積最適化をしなければ $302.96/114.60 \simeq 2.64$ より約 2.64 倍に増加する。理想的なセルモデルの場合と、提案手法の論理回路面積が若干異なるのは、回路を構成する時のセルモデルの選び方が、若干異なっているからである。面積を最適化することによって、$195.92/114.60 \simeq 1.71$ より約 1.7 倍の増加にとどまり、現実的なセルモデルにおいても、面積増加以上の性能増加が期待できる結果となっている。

第7章　おわりに

本論文では、ウェーブパイプラインに適した論理合成アルゴリズムについて議論してきた。

シングルプロセッサのさらなる性能向上の可能性を持つ動作原理としてウェーブパイプラインがある。最大遅延で動作クロック周波数が決定する。通常のパイプラインとは異なり、ウェーブパイプラインでは、遅延差によって動作クロック周波数が決定される。そのためウェーブパイプラインの処理能力向上のためには、遅延差短縮が非常に重要である。

従来は一般的なCADにより論理合成を行ったあと遅延バッファを挿入するという手法がとられていたが、遅延バッファ挿入による回路面積の大幅な増加が問題となっていた。

そこで、本研究では、論理合成段階でウェーブパイプライン処理に適した論理回路を合成する手法を提案した。提案手法では、論理段数を揃えることで遅延差を短縮させる。

従来手法で用いられている多段論理合成との比較の結果、多段論理に対して回路面積1.7倍で3倍の動作クロック周波数の向上が見込めるという結果であった。論理段数を揃えることによって、従来手法よりも回路面積の増加を抑えられたと考えることができる。この結果から、論理段数を揃える論理合成アルゴリズムは、遅延差を揃え面積の増加も抑えることができると言える。よって、提案手法は有用であると言える。

本研究では、論理回路を構成する論理素子として、3入力NANDを採用した。3入力NANDによって論理回路を構成した後に、冗長な素子を最適な入力数の論理素子に置き換えることによって、遅延差の短縮と面積の増加の抑制を図った。結果としては、冗長な素子の最適化によって多段論理合成よりも遅延差を短縮することが可能となったが、3入力NANDのみでの合成結果では、ほとんど遅延差を縮めることができなかった。この結果を考えると、論理回路を構成する素子に3入力NANDを採用したことが最適ではなかったのではないかとも考えられる。

今後の課題としては、論理回路を構成する上での最適な論理素子の選定が挙げられる。また、本研究の結果から、論理段数を揃えても論理素子自体の遅延によって遅延差の短縮に限界が生じることが示された。このことから、論理素子自体の遅延差を最小に抑えるセルモデルの設計も非常に重要であると考えられる。以上の様なことをさらに考慮することによって、本提案手法を用いてさらに遅延差を短縮することが可能であると考えられる。

謝辞

本研究を進めるにあたって、非常に熱心にご指導くださいました日比野靖教授に深く感謝いたします。研究以外にも社会人時代の経験や研究者としての心構え、困難に立ち向かっていく上での姿勢、社会や経済などの話など、今後生きていく上でとても貴重で重要な考え方を学ばせていただきました。本当にありがとうございました。田中清史准教授には、研究での御助言以外にも私生活の上で非常にお世話になりました。深く感謝いたします。ありがとうございました。菅原英子助教には、日頃の研究への御助言や研究に関する原稿の書き方など、非常にお世話になりました。本当にありがとうございました。井口寧准教授には研究に関する貴重な御助言を頂き、心から感謝いたします。ありがとうございました。

そして、研究や日常生活など、様々な楽しみを与えて頂いた計算機アーキテクチャ講座の皆様に深く感謝いたします。本当にありがとうございました。

参考文献

[1] 池田 吉朗, “ ウェーブパイプラインを用いたマルチスレッド型プロセッサアーキテクチャに関する研究 ”, 北陸先端科学技術大学院大学修士論文, 1999.

[2] David A.Patterson, John L.Hennessy, 成田 光明 訳 “ コンピュータの構成と設計 - ハードウェアとソフトウェアのインターフェース- 第 3 版 [下] ”, 日経 BP 社, 2006.

[3] C. Thomas Gray, Wentai Liu, Ralph K. Cavin, “ Wave Pipelining: Theory And CMOS Implementation ”, Kluwer Academic Publishers, 1994.

[4] PARTHENON HOME PAGE,

http://www-lab09.kuee.kyoto-u.ac.jp/parthenon/NTT/index_j.htm

[5] Carver Mead, Lynn Conway 著, 菅野 卓雄, 榊 裕之訳, “ 超 LSI システム入門 ”, 培風館, 1981.

[6] 高橋 清, “ 半導体工学 (第二版) -半導体物性の基礎- ”, 森北出版, 1993.

[7] 菅野 卓雄監修, 飯塚 哲哉編, “ CMOS 超 LSI の設計 ”, 培風館, 1993.

[8] 並木 秀明, 前田 智美, 宮尾 正大著 “ 実用入門 ディジタル回路と Verilog-HDL ”, 技術評論社, 1996.

[9] Rodney A. Brooks 著, 井田 昌之訳, “ ワークステーションシリーズ Common Lisp プログラミング ”, 丸善, 1986.

[10] VDEC 監修, 浅田 邦博, 藤田 昌宏共編 “ システム LSI 設計自動化技術の基礎 パブリックドメインツールの利用法 ”, 培風館, 2005.

[11] 芝山 達哉, “ 遅延素子にパストランジスタを用いたウェーブパイプラインプロセッサの低消費電力化 ”, 北陸先端科学技術大学院大学修士論文, 2002.

Appendix A

２入力NANDによる論理回路の構成とその評価

追加実験として、２入力 NAND のみによって論理回路を構成し、３入力 NAND と比較して評価を行った。第 6 章で行った評価と同様に、７セグメントデコーダを作製し回路の遅延差と面積を算出した。

図 7.1 に２入力 NAND による７セグメントデコーダの回路図を示す。この回路図はすでに冗長な素子を置き換えた後の回路図となっている。

図 7.1 の２入力 NAND による論理回路を評価するために、新たに２入力 NAND 評価用のセルモデルを作製した。作製したセルモデルのパラメータを表 7.1 に示す。このセルモデルは、現実的な素子の遅延を考慮したモデルとなっており、表の素子名は第 6 章で用いた表現と同様である。

表 7.1: 新たに作製した評価用セルモデルのパラメータ

	Ron-max[kΩ]	Ron-min[kΩ]	Cg[fF]	Cd[fF]	面積[μm^2]
INVinCg3=INVinCg6	8.999	8.974	2.920	1.674	2.61
INVinCg4=INVinCg6	6.919	6.899	2.920	1.909	2.57
INVinCg5=INVinCg6	5.614	5.599	2.920	2.119	2.54
NAND2=NAND2Cg3	16.098	8.048	2.920	2.100	4.16
NAND2Cg2=NAND2Cg3	9.330	4.664	2.920	2.758	3.99
INV=NAND2Cg3	19.859	19.804	2.920	1.127	2.77
INVCg2=NAND2Cg3	10.918	10.888	2.920	1.520	2.65
INVCg4=NAND2Cg3	5.822	5.806	2.920	2.081	2.55
INV=NAND2	8.999	8.974	2.920	1.674	2.61

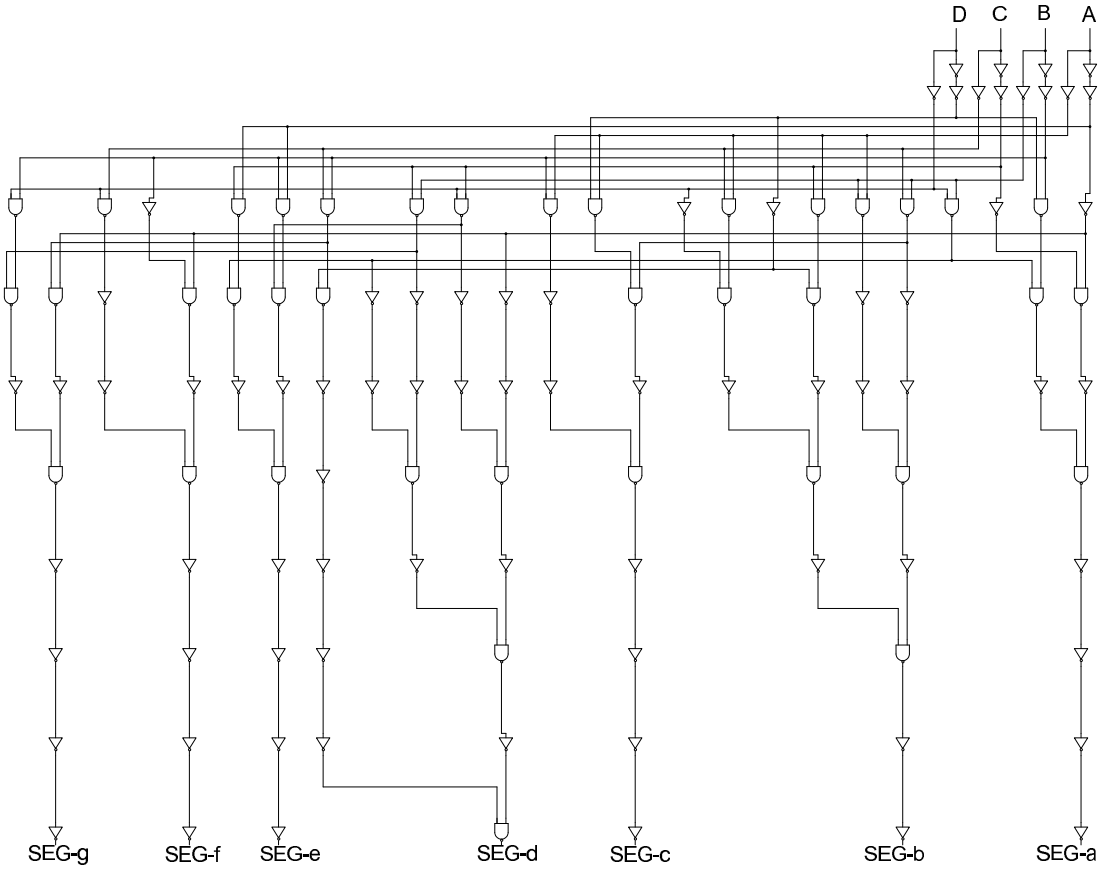


図 7.1: 冗長な素子を置き換えた 2 入力 NAND による 7 セグメントデコーダ

図 7.1 の回路の評価の結果を示す。表 7.2 に 2 入力 NAND により構成された回路の遅延の評価結果を示し、表 7.3 に 2 入力 NAND により構成された回路の評価結果を示す。表 7.2 及び表 7.3 において、比較のために、第 6 章で示した面積最適化後の 3 入力 NAND により構成された回路の評価結果と多段論理合成による回路の評価結果を示す。表 7.2 で太い枠で示されている部分が、各論理回路における最大遅延と最小遅延である。

表 7.2 に示されるように、2 入力 NAND と 3 入力 NAND の遅延差を比較すると、2 入力 NAND が 133.568[ps] に対して、3 入力 NAND では、128.311[ps] であり、2 入力 NAND の方が若干遅延差が増加している。表 7.3 より、2 入力 NAND を用いる場合と、3 入力 NAND を用いる場合の回路面積は、$340.80/195.92 \simeq 1.74$ より、2 入力 NAND を用いる方が約 1.74 倍増加している。このことから、今回採用した 3 入力 NAND は、2 入力 NAND よりも回路の遅延差と面積を共に抑えることができると言える。

表 7.2: 2 入力 NAND により構成された回路の遅延の評価結果

	2入力NAND (最大遅延)[ps]	2入力NAND (最小遅延)[ps]	3入力NAND (最大遅延)[ps]	3入力NAND (最小遅延)[ps]
seg_a	412.601	305.538	312.353	202.121
seg_b	412.597	285.149	317.021	188.710
seg_c	412.272	305.530	309.212	199.792
seg_d	418.717	301.271	316.985	198.261
seg_e	412.601	305.389	316.678	201.972
seg_f	411.905	326.229	300.095	199.792
seg_g	412.303	305.530	316.678	200.045
遅延差	133.568		128.311	

表 7.3: 2 入力 NAND により構成された回路の面積の評価結果

	2入力NAND (面積)[μm^2]	3入力NAND (面積)[μm^2]
seg_a	40.41	27.97
seg_b	66.87	33.91
seg_c	34.04	14.52
seg_d	72.32	31.67
seg_e	35.33	21.11
seg_f	32.65	14.97
seg_g	31.17	21.11
INV(input)	28.02	30.67
total	340.80	195.92

Appendix B

評価用セルモデルの設計に用いたパラメータ

表 7.4 に評価用セルモデルの設計に用いたパラメータを示す。各パラメータは、ゲート長 L、ゲート幅 W、pMOS のゲート幅 Wp、nMOS のゲート幅 Wn を表し、l はソース及びドレインの幅を表す。

表 7.4: 評価用セルモデルの設計に用いたパラメータ

	L[μm]	W[μm]	Wp[μm]	Wn[μm]	WpとWnの間隔[μm]	l[μm]
INV	0.100	2.000	1.160	0.840	4.000	0.475
NAND2	0.100	2.000	0.818	1.182	4.000	0.475
NAND3	0.100	2.000	0.631	1.369	4.000	0.475
NAND4	0.100	2.000	0.513	1.487	4.000	0.475
$INV_{=NAND3}$	0.169	1.183	0.686	0.497	4.817	0.475
$NAND2_{=NAND3}$	0.125	1.600	0.654	0.946	4.400	0.475
$INVinCg3_{=INVinCg7}$	0.149	1.342	0.778	0.564	4.658	0.475
$INVinCg4_{=INVinCg7}$	0.130	1.538	0.892	0.646	4.462	0.475
$INVinCg5_{=INVinCg7}$	0.117	1.709	0.991	0.718	4.291	0.475
$NAND3Cg3_{=NAND3Cg4}$	0.114	1.754	0.234	1.520	4.246	0.475
$NAND3Cg2_{=NAND3Cg4}$	0.135	1.482	0.198	1.284	4.518	0.475
$NAND3_{=NAND3Cg4}$	0.179	1.117	0.149	0.968	4.883	0.475
$INVCg4_{=NAND2Cg3}$	0.111	1.802	1.045	0.757	4.198	0.475
$NAND2Cg2_{=NAND2Cg3}$	0.118	1.695	0.693	1.002	4.305	0.475
$NAND3_{=NAND2Cg3}$	0.125	1.600	0.505	1.095	4.400	0.475
$NAND2_{=NAND2Cg3}$	0.155	1.291	0.528	0.763	4.709	0.475
$INVCg1_{=INVCg6}$	0.224	0.893	0.518	0.375	5.107	0.475
$INVCg2_{=INVCg6}$	0.166	1.205	0.699	0.506	4.795	0.475
$INVCg4_{=INVCg6}$	0.121	1.652	0.958	0.694	4.348	0.475
$INVCg5_{=INVCg6}$	0.109	1.835	1.064	0.771	4.165	0.475
$INVinCg3_{=INVinCg6}$	0.138	1.449	0.840	0.609	4.551	0.475
$INVinCg4_{=INVinCg6}$	0.121	1.652	0.958	0.694	4.348	0.475
$INVinCg5_{=INVinCg6}$	0.109	1.835	1.064	0.771	4.165	0.475
$INV_{=NAND2Cg3}$	0.205	0.976	0.566	0.410	5.024	0.475
$INVCg2_{=NAND2Cg3}$	0.152	1.316	0.763	0.553	4.684	0.475
$INVCg4_{=NAND2Cg3}$	0.111	1.802	1.045	0.757	4.198	0.475
$INV_{=NAND2}$	0.138	1.449	0.840	0.609	4.551	0.475