修 士 論 文

ウェーブパイプラインのための遅延均衡化回路構成と配置配線

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

宮前 義範

2003 年 3 月

修 士 論 文

ウェーブパイプラインのための遅延均衡化回路構成と配置配線

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 助教授
審査委員　堀口 進 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

110117 宮前 義範

提出年月: 2003 年 2 月

概 要

プロセッサの高速動作を目指して、デバイスや回路設計を含めた幅広い分野で研究が進められている。プロセッサの高速化は従来微細加工プロセスの進展による設計ルールの縮小によるところが大きかった。しかし極度に縮小されたデバイスの下では比例縮小則が成り立たず、最大遅延によって動作速度が決まる従来からの同期式パイプラインでは速度向上が望めなくなりつつある。一方で最大遅延と最小遅延の差でパイプラインを動作させる技術が研究されており、遅延差を効果的に縮める手法が提案されれば、従来型の同期式プロセッサを更に高速に動作させることが可能になる。

本研究は、遅延均衡を考慮した回路構成と配置配線で極限まで遅延を均衡化させる手法を提案する。

目 次

図 目 次

表 目 次

第1章　序論

1.1　本研究の背景

プロセッサを含むLSI回路は、依然としてムーアの法則に基づく高い性能向上率を保持しながら開発が進んでいる。2001年度版のSemiconductor Industry Association(SIA)の予測によれば[3]、2010年にはプロセッサのクロック周波数は11.5GHzにまで達するだろうと予測している。実際にはLSIに関する諸技術はSIAの予測よりやや先に進んでいるので、2010年にはSIAの予想よりも高い動作クロック周波数を達成するものと思われる。しかしSIAは、従来からの比例縮小則による性能向上が物理的な限界に差しかかりつつあることもまた明言しており、予測通りに性能向上が行われていくためには技術的なブレークスルーが必要だと言及している。Intelはこのことを「Transistor Challenges for the 21st Century」と表現し、半導体プロセス技術に関する様々な新技術に取り組んでおり、それにより今後当分の間ムーアの法則を維持していくことができると発表している[20]。

一方で比例縮小則によらないでスピードを上げる試みの必要性も指摘されている。CPUのスピードが上昇する一方で、記憶装置のスピードはCPUのそれに全く追い付いていないという状況にある。また単に動作スピードを上げるだけではなく、大量のトランジスタを使用してより効率的に命令を実行できる仕組みを構築しようとする方向もある。Flynnはその一例として、VLIWやマルチプロセッサを始めとするより多くの並列性を取り出すための様々な機構、設計を容易にするロバストなCADの開発の必要性などをあげている[10]。またHennesyとPattersonは著書の中で、CPUの発達の方向性として、1) トランジスタスピードの向上による利益を最大限享受できるようなアーキテクチャ、2) 大量のトランジスタリソースを最大限享受できるようなアーキテクチャの二つを示している[11]。回路が大規模になるにつれ、アーキテクチャ設計段階とプロセスに依存する回路設計の段階は分離される傾向にある。しかし真に優れた回路を設計するためには、設計段階の広い範囲を考慮した設計が必要となってくるだろう。

まとめると、プロセッサ内部における回路の高速化の要因として1) 比例縮小則による物理的な改善、2) プロセッサアーキテクチャの改善があげられる。しかしスケールダウンが進むにつれて量子的な効果が大きくなり、また配線遅延分に関してはプロセスの縮小によらず影響が一定であることから無視できなくなる。故にスイッチング時間そのものが思うように速くならない、また素子のスイッチング時間が高速になっても配線による遅延が縮まらないため回路の速度が上がらない等の現象が起き始めている。

そこでこうした物理的な要因を取り込んだCPUアーキテクチャとして、ウェーブパイ

プラインアーキテクチャの研究が進められている。従来からの同期式パイプラインでは最大遅延によって動作速度が決まるのに対し、ウェーブパイプラインでは最大遅延と最小遅延の差によって決定する。故に最大遅延がネックとなる従来からの同期式回路に対し、より高速に動作させる可能性を持つアーキテクチャと言える。既に乗算器やメモリなど部分的で規則的な回路に対して使われており、その有用性は実証されている。

1.2 本研究の目的

本研究は、ウェーブパイプライン技術のための遅延均衡化技法の一つを提案する。遅延均衡化を行う手法に関しては、現在のところ一つの素子レベルでの遅延均衡化手法や、規則的な回路に対してバッファを挿入する等が行われている程度で、回路全体に対しての遅延均衡化手法に関する詳細な手法はあまり検討されていない。バッファ挿入は配置配線や論理自体を変化させ、論理合成段階から配置配線段階にかけてのフェーズを繰り返し行う必要もでてくる。また $0.10\mu m$ 程度の小さなトランジスタを対象に、回路構成から配置配線までを考慮して遅延均衡化手法を検討したものは見受けられない。本研究では論理合成された回路データを入力とし、回路構成から配置配線段階にかけて遅延均衡化を行うことを提案する。それにより、加算器や論理演算など不均衡な 32 ビット ALU に対しても、規則的な回路である 16 ビット乗算器に対しても、論理合成段階に戻ることのないトップダウン設計でありながら、最大遅延動作の回路に対して約 2 倍の性能向上を達成した。

1.3 本研究の構成

本研究の構成は、まず本研究が対象としているウェーブパイプラインアーキテクチャの理論および研究動向に関して第 2 章で述べる。ウェーブパイプラインアーキテクチャでの遅延均衡回路の必要性と、本研究の位置づけをこの章で示す。

具体的な遅延均衡化手法に移る前に、まず第 3 章でディープサブミクロンプロセスの下での MOSFET および各層に関するパラメータ、遅延評価方法について触れる。それを元に第 4 章では CMOS 構成でのウェーブパイプラインアーキテクチャの可能性に関して考察する。以上二つの章を通して、CMOS 論理上でウェーブパイプラインを行うことの問題点を示し、本研究で提案する遅延均衡化手法の着目点を明らかにする。

第 5 章から第 6 章まで、本研究で提案する遅延均衡化手法を二つのフェーズに分けて述べる。第 5 章では、本研究が提案する遅延均衡化手法の理念を述べるとともに、まず素子の遅延差に着目した遅延均衡化手法を提案し、全加算器を対象に遅延均衡化を行うことで検討を行う。第 6 章では配置配線問題を考慮した遅延均衡化手法を提案し、4 ビット ALU を対象として遅延均衡化を行うことで検討をする。第 7 章で今回構築したシステムの流れを示し、比較的大規模な機能回路に対して遅延均衡化を試みる。第 8 章で総括を行う。

第2章　ウェーブパイプラインアーキテクチャ

本章ではまずウェーブパイプラインアーキテクチャに関する概略を述べ、本研究が対象とする遅延均衡化との関係を示す。後半でウェーブパイプラインアーキテクチャに関する先行研究を整理し、本研究の位置づけを明確にする。

2.1　ウェーブパイプラインアーキテクチャ概要

あるラッチからあるラッチにデータが伝播する状態を図 2.1 で表す。(a) は横軸を時間軸とした場合の図である。データの伝播を網掛けの三角形で示す。一般的に回路中を伝播するデータのスピードはその経路によって様々であり、あるステージを伝播する意味のあるデータの波の中で、次段のラッチに一番近い位置にあるデータ、および次段のラッチから一番遠いデータを三角形の二辺で表す。(b) はある時刻 $t = T$ に、ステージを上から見た図を模式的に表したものである。(b) では縦軸がラッチ間の信号伝播距離を示している。

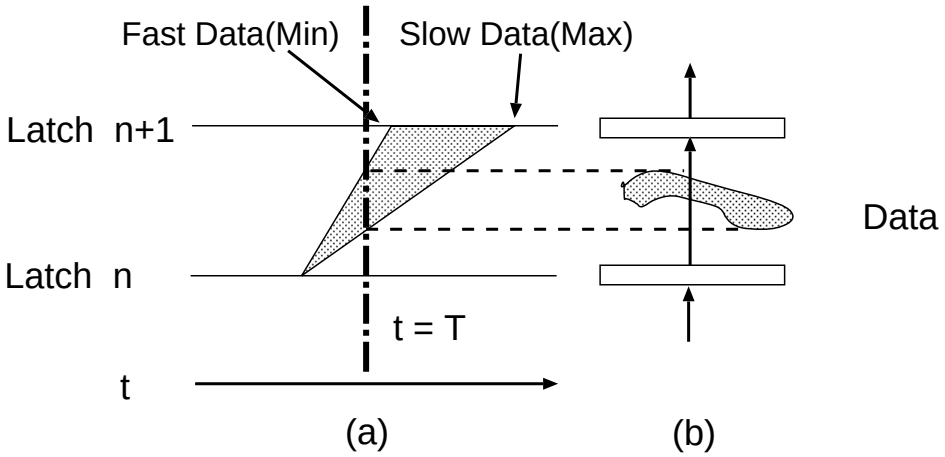

図 2.1: 回路中の模式的なデータ表現

同期式パイプラインのデータ伝播の様子を模式的に示したものを図 2.2 に示す。各ステージ間の全てのラッチには同相のクロックが印加される。全てのステージのサイクル時間は、最大伝播時間が一番遅くなるステージ (この場合はラッチ n-ラッチ n+1 間) に依存

する。ある時間 $t = T$ では、ステージ間には意味のあるデータの波は一つしか存在しない。全てのステージでデータの波がラッチに確実に取り込まれた段階で、全てのラッチに同時にクロックが印加される。この場合は最大遅延によってクロック周波数が制限されることになる。

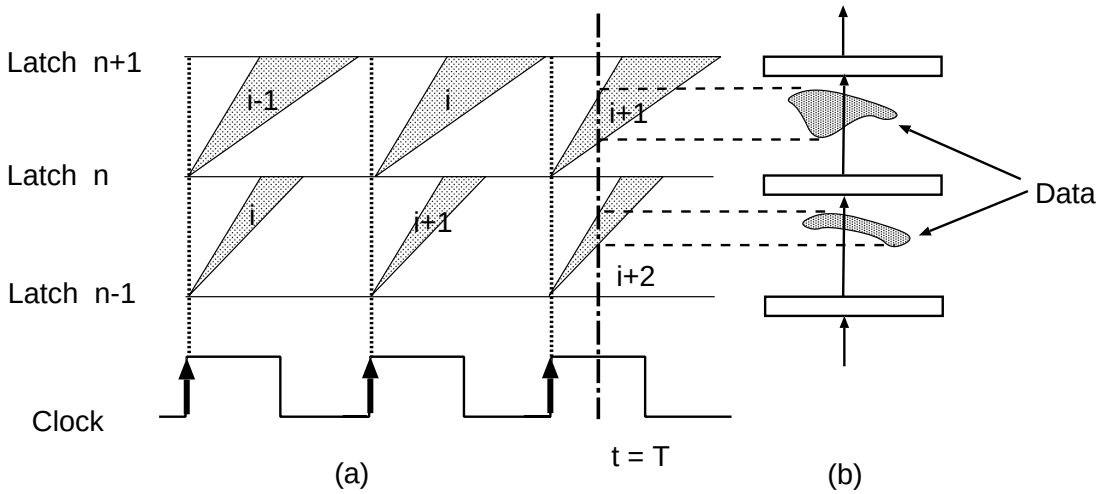

図 2.2: 同期式パイプラインの模式的なデータ表現

ウェーブパイプライン動作のデータ伝播の様子を模式的に示したものを図 2.3 に示す。各ステージ間のラッチにはステージ毎に違うタイミングでクロックが印加される。ある時間 $t = T$ では、ステージ間には意味のあるデータの波が一つ以上存在することがある。このとき、データの波が互いに重ならないようにデータを送り出せば、正しいデータを伝播させることができる。故に最大遅延が大きな回路であっても最小遅延との差が小さければ、データの波が重なることなく次々とデータをステージに投入できる。これがウェーブパイプラインの基本的な考え方である。

図 2.4 にウェーブパイプラインの概略を示す。図で横軸は時間軸を、縦軸はラッチ間の距離を示している。従来では最大遅延の波が到着してから続くデータを送り出していた。ウェーブパイプラインでは先行するデータがある程度まで進んだ段階で続くデータを送り出す。この時前後のデータが重ならないよう、最大遅延と最小遅延の差まで動作速度を詰めることができる。クロック周期は式 (2.1) によって表される。Δ_{DQ} はラッチのセットアップ時間、Δ_{QC} はホールド時間を、t_{skew} はクロックスキューをそれぞれ示す。

$$T_{CK} > (D_{max} - D_{min}) + \Delta_{DQ} + \Delta_{QC} + t_{skew} \tag{2.1}$$

回路規模が大きい場合、ラッチのセットアップ／ホールド時間およびクロックスキューに対して、回路内伝播遅延が大きくなる。この式は、最小遅延分 D_{min} を大きくすることでクロック周期を縮めることができることを示している。本研究が指す「遅延均衡化」とは、回路内の最大伝播遅延 D_{max} および最小伝播遅延 D_{min} との差を縮めることを指す。

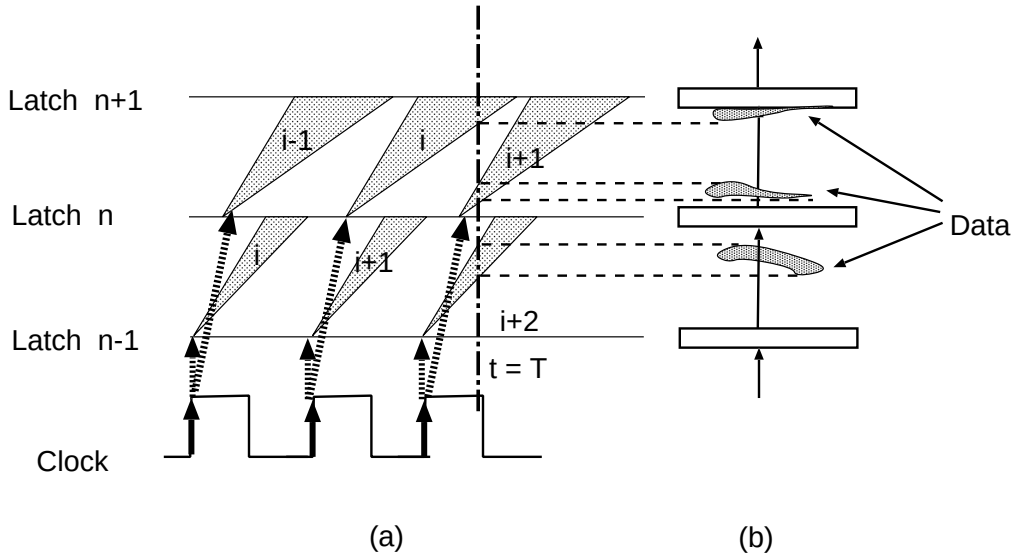

図 2.3: ウェーブパイプラインの模式的なデータ表現

ウェーブパイプラインアーキテクチャ上では、回路の遅延均衡化が直接性能にかかわってくる。

2.2 ウェーブパイプラインアーキテクチャに関する先行研究

以下本研究の位置付けを明確にするために、ウェーブパイプラインアーキテクチャに関する研究をまとめた。ウェーブパイプラインアーキテクチャ関する研究は、理論および実装とに渡って現在も行われている。その概要に関しては [7] が詳しい。

スタンフォード大学ではディープサブミクロンデバイス下での CPU アーキテクチャとして、プロセッサ **SNAP** を土台に様々な技術研究が行われている [9]。その内 16 ビット乗算器に対してウェーブパイプライン技術が実際に使用されている。この 16 ビット乗算器は大部分がパストランジスタ論理の $(4, 2)$ セレクタで構成されている。パストランジスタ論理を使用することで回路全体のレイテンシをさげ、遅延差を均衡化させる手法がとられている [12]。その他ウェーブパイプラインアーキテクチャの規則的な回路への適用例としては、SRAM をウェーブ動作させる例が参考になるだろう [16]。

CMOS 論理では、入力の遷移の違いにより多入力素子で生じる遅延差発生をどうしても避けることができない。そこで入力でのどの遷移に対しても反応速度が一様である Look Up Tabel(LUT) の特徴を生かして、乗算器を対象に FPGA 上でウェーブパイプライン化を適用する試みも行われている [4][5]。またある論理関数を完全二分木で構成し、二分木中の一つ一つのノードを (2,1) セレクタとして FPGA 上に実装することでパス間での遅延差を均衡させ、ウェーブ動作を実行する試みも行われている [22]。

ウェーブパイプラインのより応用的な試みとして、フィードバックを持つ回路に対して

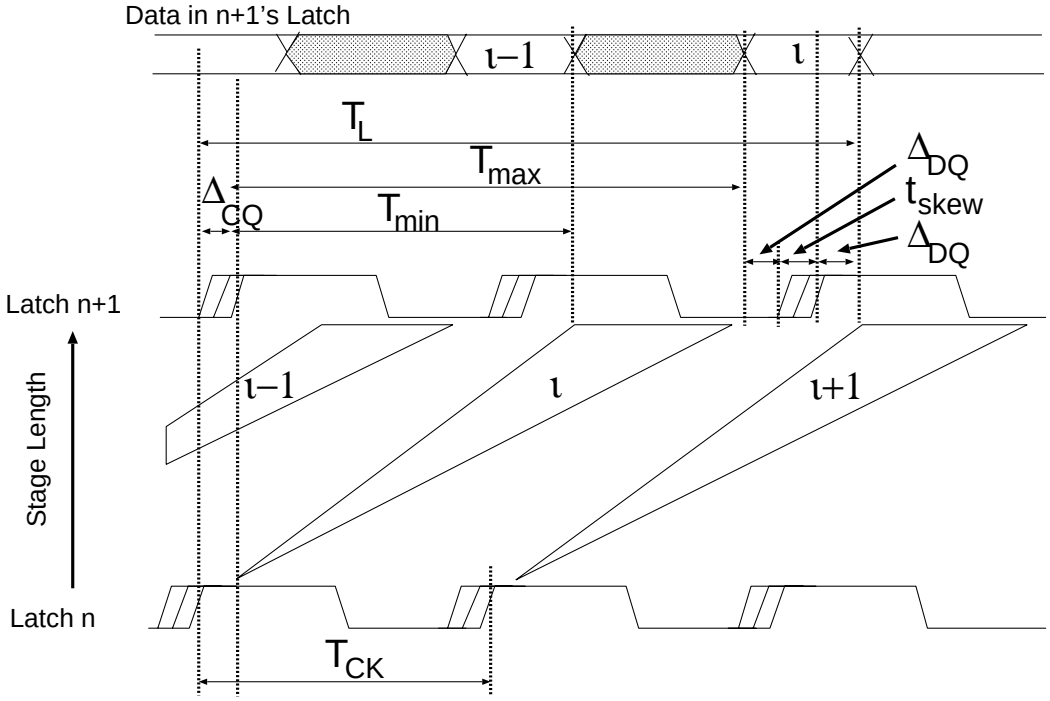

図 2.4: ウェーブパイプライン動作図 [7]

ウェーブ化を施すことや、非同期式回路に対してウェーブ化を施すことなどがある [19]。回路技術との組合せとしては他に、準同期回路技法でのクロックパスの操作によるタイミング調整と、従来からのバッファ挿入による遅延均衡化を組み合わせてウェーブパイプラインを行う方式が提案されている [29]。一方で回路の低消費電力動作のためにウェーブパイプラインを採り入れる試みなども提案されている [27][28]。またウェーブパイプラインアーキテクチャと CPU アーキテクチャを組み合わせた例として、マルチスレッドプロセッサに対してウェーブ化を施すことも提案されている [30]。マルチスレッドプロセッサでは投入された命令間に依存関係が生じない。故にフィードバック回路を持たない回路構成が可能になる。ウェーブパイプラインアーキテクチャはその性質上、フィードバックを持つ回路への導入は容易ではない。故にウェーブ化設計を考える際は、回路レベルだけではなくアーキテクチャレベルでもその特性を十分生かせるような設計が必要とされる。

以上のどの研究もウェーブパイプラインアーキテクチャ上での研究であり、遅延均衡化による恩恵を受けることができる。しかし遅延均衡化を行う手法そのものに関しては、一つの素子レベルでの遅延均衡化手法や規則的な回路に対してバッファを挿入する等が行われているものの、回路構成から配置配線までを考慮して詳細に検討した研究、あるいは回路全体に対して検討した研究はあまりない。また実際の回路の評価では FPGA 上での評価が多く、ディープサブミクロンデバイス下で評価が行われたものは少ない。本研究の目的は、フィードバックを持たない任意の回路全体に対し、ディープサブミクロンデバイス下での回路構成から配置配線までを考慮した遅延均衡化手法を提案することにある。

第3章　MOSFETモデルと遅延評価方法

具体的な遅延均衡化手法を示す前に、本章では設計ルール$0.10\mu m$を対象としてMOSFETおよび各配線層における各パラメータの算出モデルを示し、遅延均衡化手法の評価にあたり論理回路モデルを構築する。合わせて遅延評価方法に関しても考察を行う。

3.1　MOSFETの基本

デバイスパラメータに関する検討の前に、本節ではMOSFET、特にCMOS構成の回路に関する基本な性質を示すことで、続く各議論の土台を提示する。図3.1(a)に説明のために間略したMOSFETモデルを、(b)には回路でのシンボルを示す。

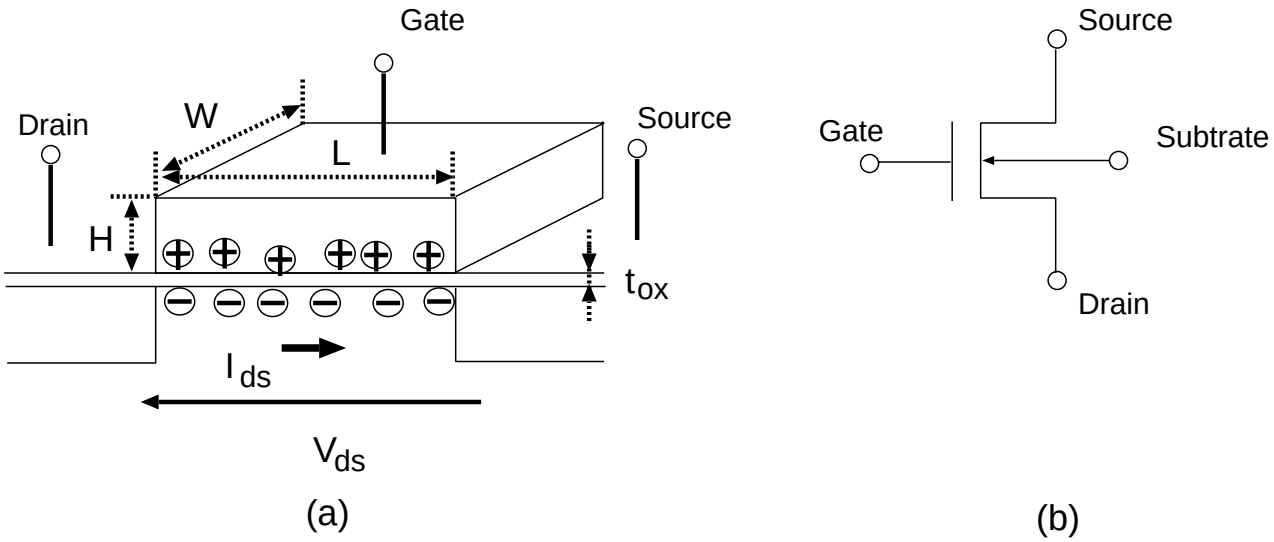

図 3.1: 簡略化したMOSFETにモデル

3.1.1　MOSFETの性質

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りソース·ドレイン間のチャネルを流れる電荷を制御することにある。ドレイン電極からソース電極へ流れる電流の大きさは、チャネルに誘起した電荷量を走行時間で割ったものに等しい。

また走行時間は、電子の移動すべき距離 (ゲート長) を電子の平均速度で割ったものに等しい。通常のトランジスタの場合、電子速度は加速するために加えた電界に比例する。この時、電子の平均速度 v_{ave} は、電界 E を用いて次式で表せる。

$$v_{ave} = \mu E \tag{3.1}$$

ここで比例定数 μ は移動度であり、温度に依存する。また電界 E は、ゲート長 L とドレイン · ソース間電圧 V_{ds} が小さいとき次式で与えられる。

$$E = \frac{V_{ds}}{L} \tag{3.2}$$

以上から走行時間 τ は、ドレイン · ソース間電圧 V_{ds} が小さい場合次式で与えられる。

$$\tau = \frac{L}{v} = \frac{L}{\mu E} = \frac{L^2}{\mu V_{ds}} \tag{3.3}$$

ゲート容量 C_g は、ゲート長 L、ゲート幅 W、ゲートの酸化膜の厚さ t_{ox} から次式で表せる。

$$C_g = \epsilon_0 \epsilon_{SiO_2} \frac{WL}{t_{ox}} \tag{3.4}$$

走行に寄与する電流の総量 Q はゲート容量 C_g にかかる電圧 $(V_{gs} - V_{th})$ で求めることができる。

$$Q = C_g(V_{gs} - V_{th}) \tag{3.5}$$

以上からドレイン · ソース電流 I_{ds} は以下の式で表せる。

$$I_{ds} = \frac{dQ}{dt} \approx \frac{Q}{\tau} = \frac{\mu \epsilon_0 \epsilon_{SiO_2} W}{L t_{ox}} (V_{gs} - V_{th}) V_{ds} \tag{3.6}$$

この式から、ドレイン · ソース間電流 I_{ds} とかかる電圧 V_{ds} に関する以下の関係式を導くことができる。

$$R_{on} = \frac{V_{ds}}{I_{ds}} = \frac{t_{ox} L}{\mu \epsilon_0 \epsilon_{SiO_2} W (V_{gs} - V_{th})} \tag{3.7}$$

この式は回路中の MOSFET がある負荷を駆動する際に、MOSFET を抵抗 R_{on} で置き換えたものとみなせることを示している。このようにして決まる R_{on} をオン抵抗と呼ぶ。実際には MOSFET のソース · ドレイン電流 I_{ds} は動的に変化する。故にオン抵抗 R_{on} もまた動的に変化する。しかし各 MOSFET を静的な抵抗とみなすことで、評価が単純化される。図 3.2 に MOSFET による回路 (a) をオン抵抗 R_{on} でモデル化した図を示す。

3.1.2 CMOS 論理の回路に関する性質

以上の議論は、キャリアが電子による NMOS(Negative-MOS) に関するものである。NMOS ではソースに対してドレインに正電圧がかかっている状態で、ゲートに正電圧がかか

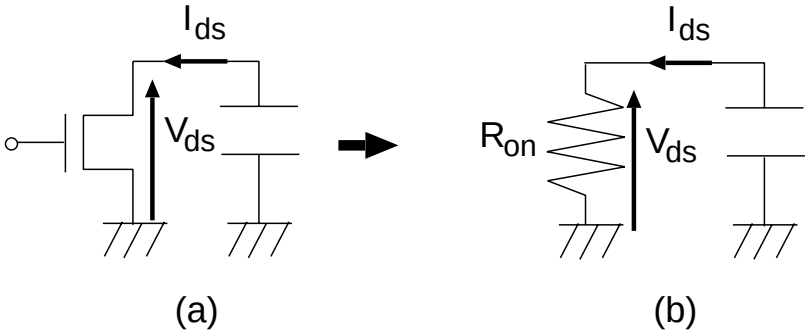

図 3.2: MOSFET のオン抵抗による置き換え

るとドレインからソースに電流が流れる。一方、キャリアがホールによる PMOS(Positive-MOS) は、ソースに対してドレインに負電圧がかかっている状態で、ゲートに負電圧がかかるとドレインからソースに電流が流れる。CMOS(Complementaly-MOS) はNMOS,PMOS の特徴を利用したものである。簡単なモデル図を図 3.3 に示す。入力として $V_{in} = 0$ すなわち“ 0 ”が入力されると PMOS が駆動し、出力側が $V_{out} = V_{dd}$ すなわち“ 1 ”が出力されたことになる。入力として $V_{in} = V_{dd}$ すなわち“ 1 ”が入力されると、NMOS が駆動し出力側が $V_{out} = 0$ すなわち“ 0 ”が出力される。つまり図 3.3 はインバータの回路を示している。

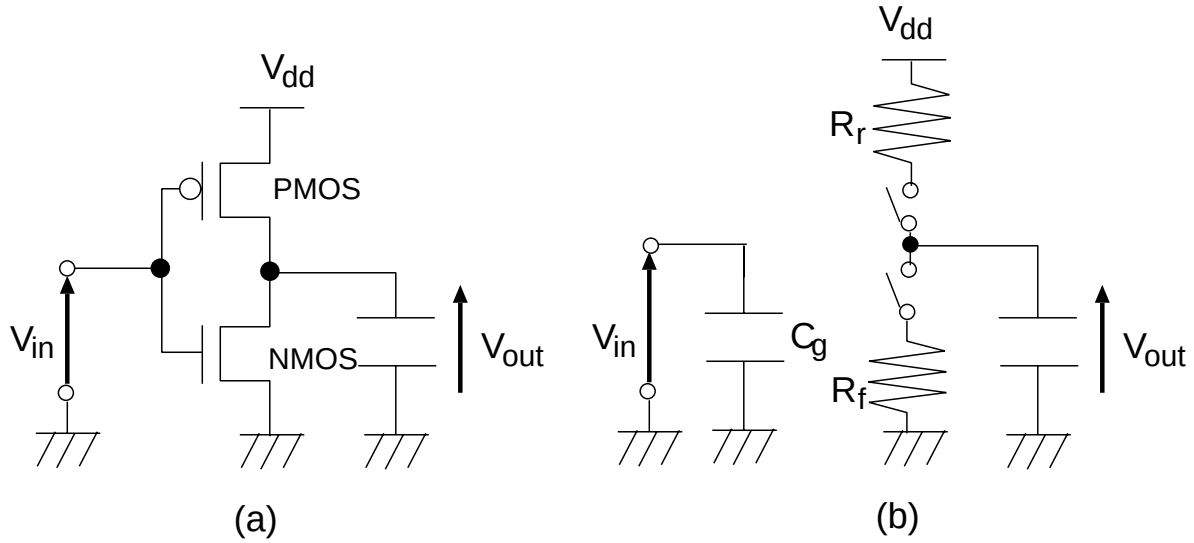

図 3.3: CMOS モデル

図 3.3 を遅延差という観点から見ると、オン抵抗として立ち上がりに使用される R_r および立ち下がりに使用される R_f と二つがあり、両者の値が異なる場合は遅延差が生じてしまうことになる。NMOS,PMOS では移動度 μ が異なるほか、動作時の温度や電圧などによりオン抵抗は変化するため、R_r, R_f を常に一致、あるいは一定の値にするのは不可

能である。故に CMOS 論理では立ち上がり／立ち下がりで遅延差が生じてしまうことになる。

以上は CMOS 論理でのインバータの構造である。多入力素子ではその遅延差はより顕著になる。図 3.4(a) に 2 入力 NAND の回路図を、(b) に MOSFET をオン抵抗でモデル化した回路を示す。立ち上がりの際を考えると、出力が $0 \rightarrow 1$ と遷移する可能性としては、1) 二つの PMOS が同時に動作する場合、2) 片方の PMOS のみが動作する場合、3) 片方が動作中にもう片方が動作し始めるなどの状態が考えられる。各 PMOS,NMOS で互いに同じ特性を持つものとすると、1) の場合は 2 つのオン抵抗 R_f が並列に接続されていると考えることができる。この場合出力側を駆動するオン抵抗は二つのオン抵抗 R_f の合成抵抗となり、その大きさは $1/2R_f$ となる。2) の場合はインバータ同様、駆動するオン抵抗の大きさは R_f となる。故に立ち上がりでかつ温度／電圧などの動作条件が同一であったとしても、駆動するオン抵抗は状況によって最大オン抵抗と最小オン抵抗で 2 倍違う。今素子が負荷 C を駆動すると考えた場合、負荷 C が一定の電圧になる時間 t は R_{on} と C の時定数から以下の式で求めることができる。

$$t = R_{on}C \tag{3.8}$$

つまり立ち上がりのみに限って、かつ温度／電圧などの動作条件が一定であったとしても、最大遅延と最小遅延で 2 倍の差が出てしまうことになる。ピン数が n 本の NAND であれば、立ち上がりのみかつ動作条件一定でも最大遅延と最小遅延で n 倍の差がでてしまうことになる。CMOS 論理上でウェーブ動作回路を設計する場合、このように多入力素子により生じる遅延差が大きな問題となる。

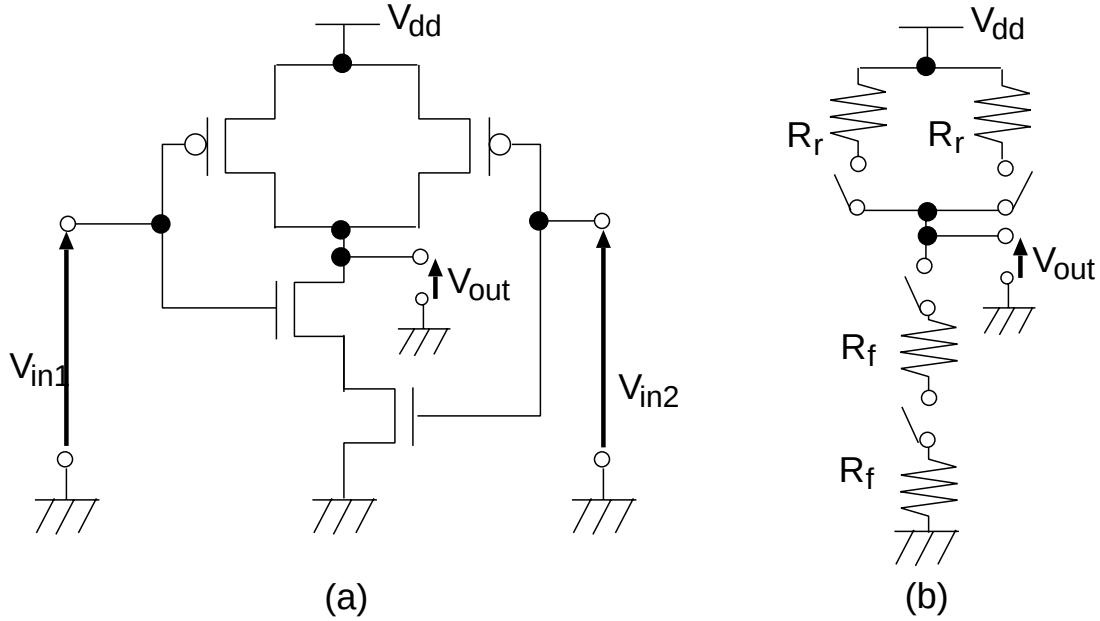

図 3.4: 2 入力 NAND モデル

3.2 MOSFET モデルとデバイスパラメータの決定

本節では MOSFET モデルおよびデバイスパラメータに関して検討する。まずパラメータ決定に際し基本方針を決め、一方で半導体デバイスに関する最近の動向を示す。その後 MOSFET および各配線層のモデルを構築する。MOSFET に関しては基本モデルを示し、各パラメータを物理的な式に従い算出した。各層でのパラメータに関しても同様に、まず基本モデルを提示し、物理的な式から各パラメータを算出した。

3.2.1 パラメータ決定に関する基本方針

デバイスパラメータに大きな影響を与えるプロセスに関する詳細な情報は、その重要性から多くの企業で非公開とされている。またデバイスが縮小するにつれてデバイスモデルが複雑化している。しかしデバイスモデルに必要なパラメータには、式の連続性を保証するためであったり経験則からくるものなど、本質的な物理的要因には直接影響しないものも多くある [8]。これらの理由から、詳細なパラメータを得ることは不可能である。しかしシミュレーションによる評価においては、むしろ物理的要因を重視してパラメータを算出し評価する方が分かり良い。そこで本研究では各デバイスパラメータ決定に関する基本方針を以下のように定めた。

1. **できるだけ最新のデータを「目安」に用いる。**
   デバイスに関する論文やテクニカルレポートに複数あたり、できるかぎり現実的な値を目安にする。但し掲載されているデータをそのまま使うことはしない。

2. **できるだけ物理的法則に従いパラメータを出す。**
   1. でのデータを目安にパラメータを自ら求める。求める過程は出来る限り物理的な裏付けがあるものとし、求めたデータが現実的かどうかを 1. でのデータと比較しチェックする。

3. **パラメータは必要最低限のものに抑える。**
   ある程度現実的なパラメータであれば十分である。

3.2.2 デバイスパラメータに関する事例

まずパラメータ導出の前に、各デバイスパラメータに関する現状を述べる。MOSFET に関しては、2001 年度版の半導体に関するロードマップによると [3]、製造ルール $0.10\mu m$ での酸化膜厚 t_{ox} は $1.1nm - 1.6nm$、有効ゲート長は $0.065\mu m$ にまで達すると予測している。一方で、半導体デバイスは物理的／材料的な限界に達しはじめたとの見解を示している。Intel は従来からの技法による限界を示しており [20]、酸化膜厚は $2.3nm$ 程度、ゲート長は $0.10\mu m$、有効ゲート長は $0.060\mu m$ 程度が限界であるとし、それ以上微細化を進める際にはより先端的な技術を導入しなければならないとしている。

次に各層でのパラメータに関して述べる。配線容量については、集積度が上がるにつれて配線間で生じるカップリングキャパシタンスの影響が支配的になりつつあるという状況にある。SEQENCE Design 社は、製造ルール $0.10\mu m$ では上下層とのエリア／フリンジングキャパシタンンスの容量の和に対して、カップリングキャパシタンスの容量が約 3 倍程度になると予測している [21]。このような配線で生じる容量の影響を軽減するために、内部誘電体 (ILD) について従来の SiO_2 に対して比誘電率の低い物質を使用することが提案されており、Intel では比誘電率 $k = 3.6$ 程度の $Fluorinated\,SiO_2$ を次期 ILD として発表している [23]。

3.2.3 MOSFET に関するパラメータ

以上を踏まえて、MOSFET に関する各パラメータを決定した。図 3.5 に簡略化した MOSFET に関する主要パラメータの概要を示す。

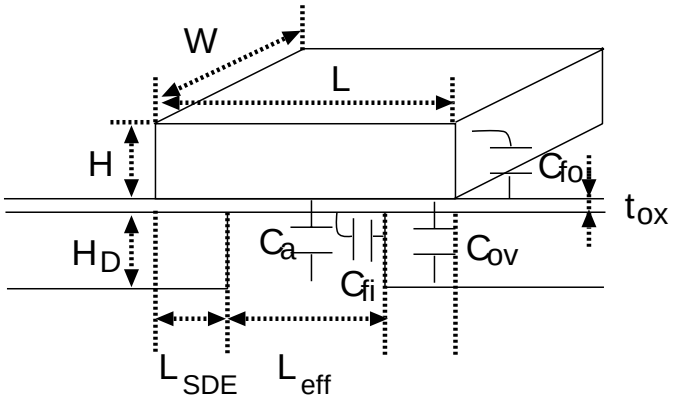

図 3.5: MOSFET モデル

また図中パラメータは以下のように設定した。

- t_{ox} : 酸化膜厚　$3.20 \times 10^{-9}[m]$
- L : ゲート長　$0.10 \times 10^{-6}[m]$
- W : (単位) ゲート幅　$0.20 \times 10^{-6}[m]$
- H : ゲート高さ　$5.00 \times 10^{-8}[m]$
- L_{SDE} : ソース／ドレインオーバラップ分　$0.00[m]$
- L_{eff} : 有効ゲート長　$0.10 \times 10^{-6}[m]$
- H_D : 拡散深さ　$5.00 \times 10^{-8}[m]$

また使用する MOSFET モデルは、BSIM3v3(Berkeley Short Channel IGFET Model 3 version 3) を使用することとした。

以下ゲート幅 $W = 0.10[\mu m]$ あたりの値である。酸化膜容量 C_{ox} に関しては次式が成り立つ。

$$C_{ox} = \epsilon_0 \epsilon_{SiO2} \frac{WL}{t_{ox}} = 0.106 \quad [fF] \tag{3.9}$$

外部フリンジングキャパシタンス C_{fo} および内部フリンジングキャパシタンス C_{fi} に関しては、BSIM3 で使用されている次式より求めた [8]。ここで α はゲートと基板面のなす角 [rad] であり、$\beta = (2\epsilon_{Si})/(\pi\epsilon_{SiO_2})$ である。

$$C_{fo} = \frac{\epsilon_0 \epsilon_{SiO2}}{\alpha} W \ln\left(1 + \frac{H}{t_{ox}}\right) = 0.006 \quad [fF] \tag{3.10}$$

$$C_{fi} = \frac{\epsilon_0 \epsilon_{SiO2}}{\beta} W \ln\left(1 + \frac{H_D \sin\beta}{t_{ox}}\right) = 0.014 \quad [fF] \tag{3.11}$$

以上から $W = 0.10[\mu m]$ 当たりのゲート容量 C_g を次式により求めた。

$$C_g = C_{ox} + C_{fo} + C_{fi} = 0.126 \quad [fF] \tag{3.12}$$

最小インバータでは $W_n = 0.20[\mu m], W_p = 0.50[\mu m]$ であるので、最小インバータでのゲート容量は $C_g = 0.882[fF]$ となる。なお動作条件は、動作電圧を 0.9V-1.0V の範囲で、動作温度を 25C-70C の範囲とした。

3.2.4 各層に関するパラメータ

配線容量として 3 つの容量を定める。エリアキャパシタンス C_a とは配線の上下面と各層間の間に生じる容量を指す。フリンジングキャパシタンス C_f は配線の側面と各層間との間に生じる容量を指す。また配線間に生じる容量をカップリングキャパシタンス C_x として定義する。図 3.6 に配線容量に関する関係図を示す。

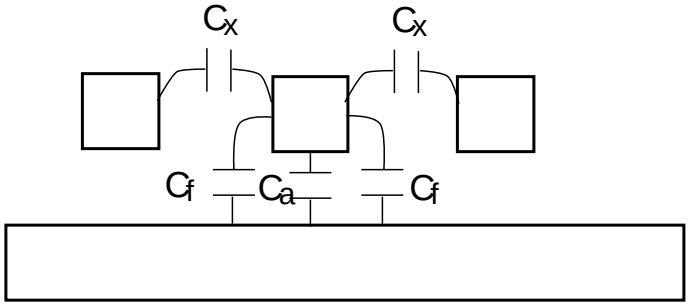

図 3.6: 配線容量概要

実際各容量を正確に評価するのは困難である。カップリングキャパシタンスについては、隣り合う配線の電位によって容量は変化する。故に配線／動作状況によって実質的な

配線容量は変化する。またエリア／フリンジングキャパシタンスについても同様に、各層間での配線／動作状況によって容量が変化する。最大遅延による動作を考えるならば、全ての配線容量が全く充電されていないとみなした場合が最悪遅延時間であるから、その場合を考慮すれば良い。一方ウェーブ設計という観点からすると遅延差をそろえることが重要であり、実質的な容量が変化することで遅延にばらつきが生じでしまうような配線は好ましくない。故に遅延差のばらつきを抑えるため、配線領域の両隣／上下はグランド線であることが望ましい。但し貴重な配線領域を無駄に使うことになる。

各配線層に関する基本方針を以下に示す。どんな論理的遷移に対しても出来る限り容量が一定となるように、各配線領域をグランドで遮蔽する。

- 配線は第一層、第三層で行い、どちらの層も同じ配線幅／配線間隔とする。
- 配線は銅配線とする。
- 第一層配線のため、基板面及び第二層はグラウンドに落とす。
- 第三層配線のため、第二層及び第四層はグラウンドに落とす。
- グローバルな配線は第五層以上で行う。
- 一本の配線を行った際、その両側にも配線を敷きグラウンドに落とす。

層に関するパラメータに関しては、Intel で公開されている製造ルール $0.090\mu m$ でのデバイス写真を参考に行った [6]。また、各層を埋める誘電体 (ILD) は Intel の Low-K 技術 (Fluorinated SiO_2) を仮定し、$K = 3.60$ とする。図 3.7 に各層の関係図を示す。配線間隔は、配線のしやすさから配線幅と同じにとった。

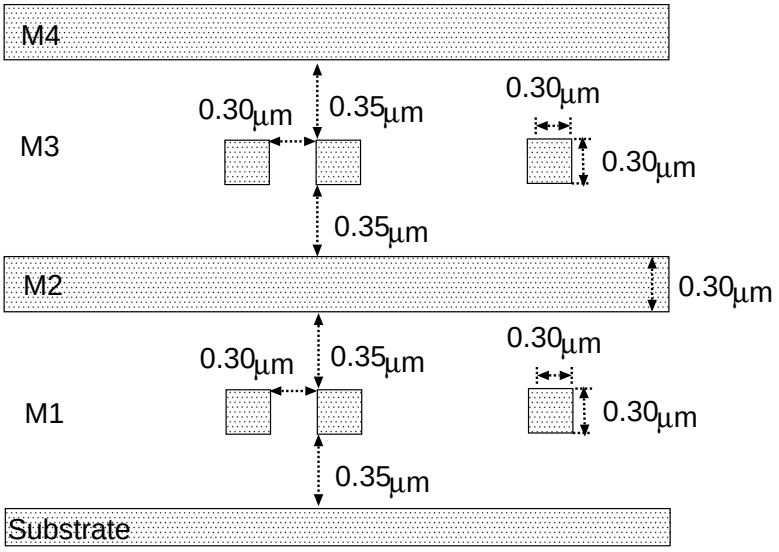

図 3.7: 各層に関するパラメータ概要

以下第一層 (第三層) に関する各種パラメータを求める。基板面 (第二層) とのエリア／フリンジングキャパシタンス $C_{s1}(C_{s3})$ に関しては、配線幅 W および配線高さ T, 上下層との距離 H として以下の式を用いた [24]。

$$C_{s1} = C_{23} = \epsilon_0\epsilon_k \left(1.15\left(\frac{W}{H}\right) + 2.80\left(\frac{T}{H}\right)^{0.222}\right) = 0.114 \quad [fF/\mu m] \tag{3.13}$$

カップリングキャパシタンス C_x については、平衡平面板コンデンサとして求めた。

$$C_x = 2 \cdot \epsilon_0\epsilon_k \frac{Wl}{D} = 0.053 \quad [fF/\mu m] \tag{3.14}$$

第二層 (第四層) とのエリア／フリンジングキャパシタンス $C_{12}(C_{34})$ も同様に式 (3.13) から求めた。

$$C_{12} = C_{34} = \epsilon_0\epsilon_k \left(1.15\left(\frac{W}{H}\right) + 2.80\left(\frac{T}{H}\right)^{0.222}\right) = 0.114 \quad [fF/\mu m] \tag{3.15}$$

以上から単位長さ当たりの配線容量 $C_{W1}(C_{W3})$ を求めた。

$$C_{W1} = C_{W3} = C_{s1} + C_{12} + C_x = 0.281 \quad [fF/\mu m] \tag{3.16}$$

一方、配線抵抗 $R_{W1}(R_{W3})$ は以下の様にして求めることができる。

$$R_{W1} = R_{W3} = \rho\frac{l}{S} = 0.227 \quad [\Omega/\mu m] \tag{3.17}$$

不純物濃度、移動度や飽和速度等他の物理的なパラメータに関しては、チップサービス提供組織 MOSIS[1] にて実際にチップ製作に使用されかつパラメータがオンラインで公開されている、TSMC 社の $0.18\mu m$ ルールでのデバイスパラメータと比較することで得た。付録 A の表 A.1 に移動度および飽和速度に関して得たパラメータを載せる。

3.2.5 MOSFET の特性および配線層に関する考察

以上で MOSFET に関して、遅延評価に必要なパラメータの計算を終える。最後に、パラメータを必要最小限に絞って構築したモデルを $0.18[\mu m]$ にサイズを大きくしたものと、参考にした TSMC 社のパラメータを使用したモデルとの比較を行なった。両方のモデルに共通するパラメータは同じ値とし、ゲートサイズを $W_n/W_p = 1.00[\mu m]/2.80[\mu m]$ とした際の $v_{ds} - i_{ds}$ 特性を評価した。図 3.8 に NMOS の特性を、図 3.9 に PMOS の特性を示す。NMOS の飽和領域において、TSMC 社のモデルに対して簡略化したモデルは電流増加量が大きくなる傾向にある。しかし少ないパラメータでありながら、構築したモデルでも TSMC 社のモデルに十分近い特性を得ることができた。また HSpice の静的なモデル[1]によるシミュレーションから得られるゲート容量と、本モデルから得られるゲート容

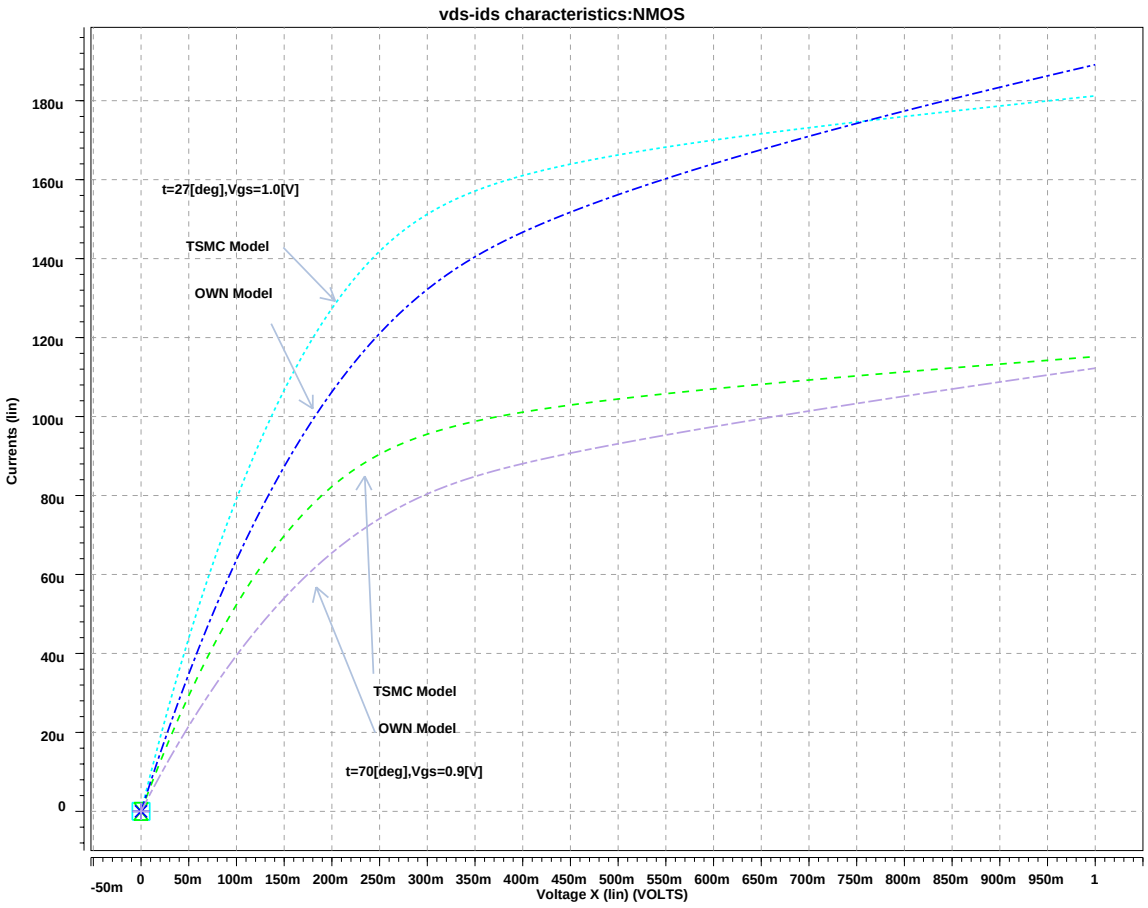

図 3.8: $W_n/W_p = 1.00\mu m/2.80\mu m$ での NMOS に関する $v_{ds} - i_{ds}$ グラフ

表 3.1: 静的モデルによるゲート容量比較

	NMOS[fF]	PMOS[fF]
TSMC 社	1.79	4.62
簡易モデル	1.71	4.78

量との比較を表 3.1 に示す。HSpice での静的なモデルと構築した容量モデルは、よく一致している。

以上評価した簡易モデルを使用して、製造ルール $0.10\mu m$ を想定した値で HSpice によるシミュレーションを行なうことで、MOSFET のゲートサイズを $W_n/W_p = 0.20[\mu m]/0.50[\mu m]$ とした際の $v_{ds} - i_{ds}$ 特性を評価した。グラフを図 3.10 に示す。このグラフから、簡易モ

[1]通常 BSIM3 ではキャパシタンスは電荷をベースに算出される。ここでは静的モデルを仮定した。

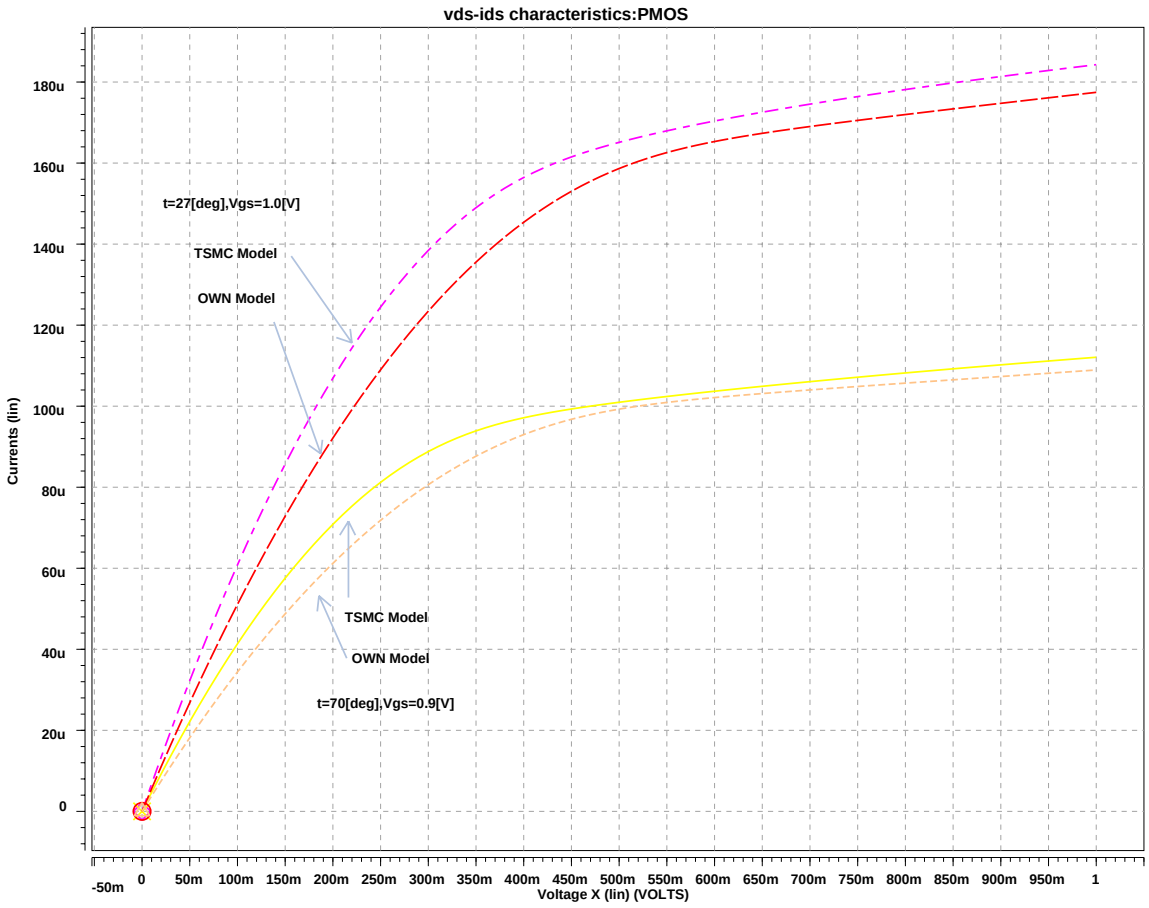

図 3.9: $W_n/W_p = 1.00\mu m/2.80\mu m$ での PMOS に関する $v_{ds} - i_{ds}$ グラフ

デルでの製造ルール $0.10[\mu m]$ における最小インバータに関して、最大オン抵抗 $R_{max} = 9.35[k\Omega]$、および最小オン抵抗 $R_{min} = 15.5[k\Omega]$ と算出した。

また本章で定義したモデルは配線容量による影響が大きく、30 グリッド $(=3.0\mu m)$ 程度配線すると配線容量の和がリファレンスインバータのゲート容量に匹敵する。このことからも、ディープサブミクロンデバイスの下では配線容量を考慮した評価が非常に重要である。

3.3 遅延評価方法に関する検討

続いて遅延評価方法に関する検討を行った。素子／回路のモデル化の方法および遅延評価モデルを示し、MOSFET を使用した場合／回路シュミレーションを行った場合の挙動

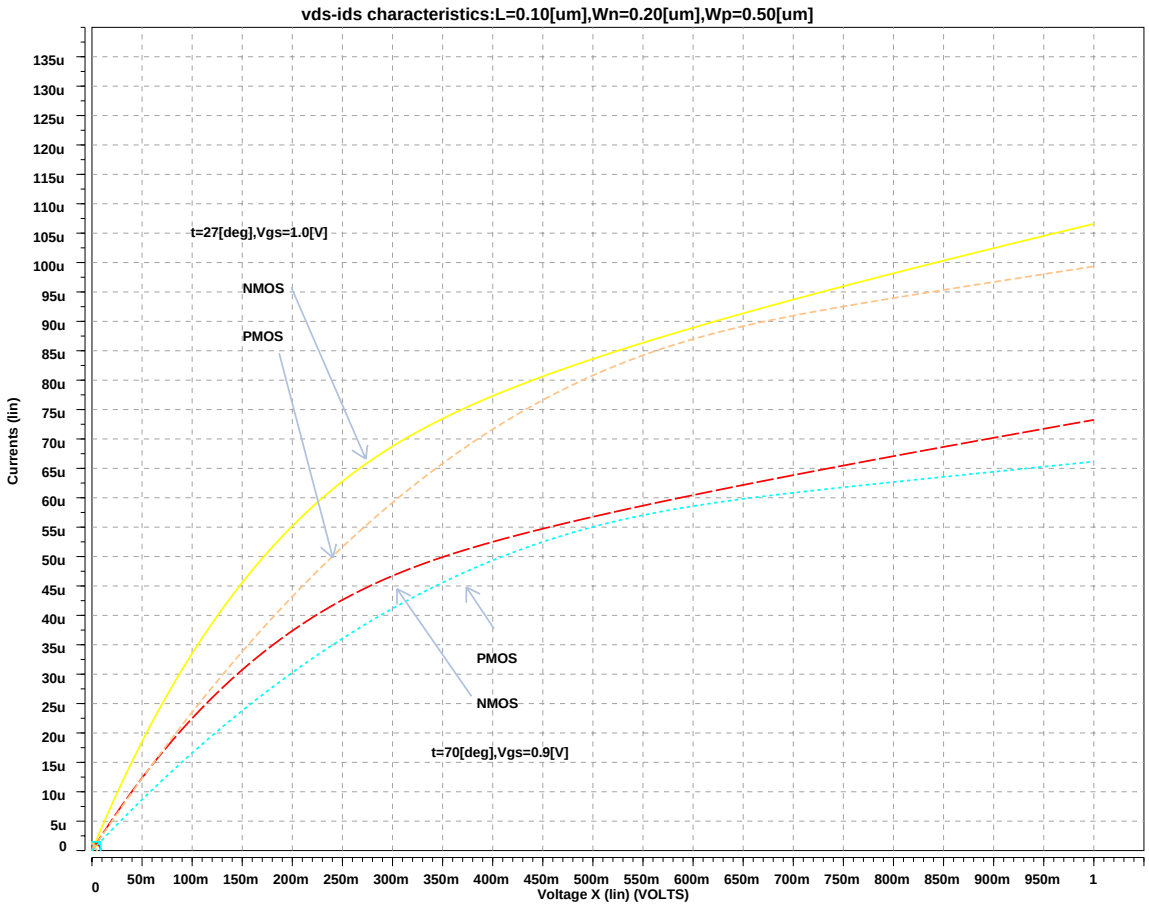

図 3.10: 簡易モデルによるゲート長 $L = 0.10[\mu m], W_n/W_p = 0.20\mu m/0.50\mu m$ での $V_{ds} - i_{ds}$ グラフ

と比較した。

3.3.1 素子のモデル化

遅延を簡単に評価するために、素子のモデルにはオン抵抗／拡散容量によってモデル化する方法をとる。オン抵抗は温度／電圧／サイズなど種々のパラメータから求まる値であるので、これら複数の要因をオン抵抗内に内包することができる。図 3.11 に各パラメータでモデル化した MOSFET モデルを示す [13]。パラメータは、入力容量 C_{in}、立上りオン抵抗 R_r、立ち下がりオン抵抗 R_f、および拡散容量 C_d から成る。

ウェーブ動作を検証する際には、最大遅延および最小遅延が評価できるモデルがよい。ウェーブ動作評価用に構築した MOSFET モデルを図 3.12 に示す。各論理素子は入力容量

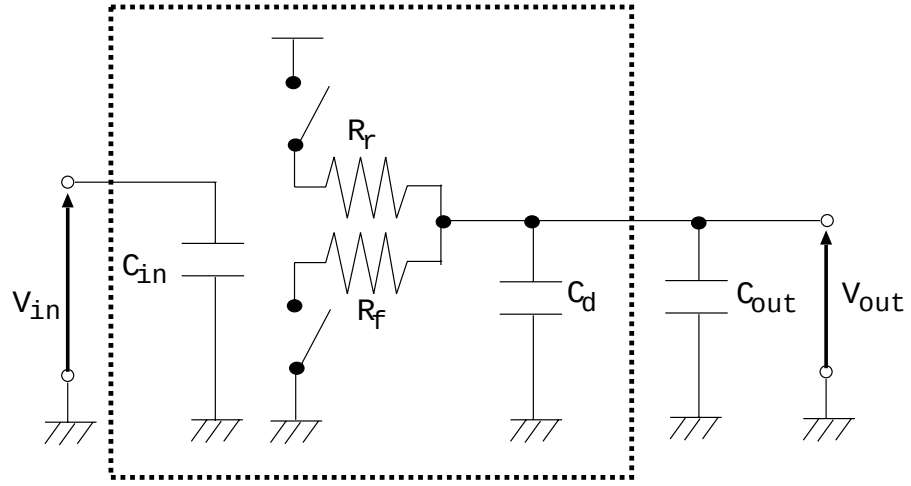

図 3.11: 論理素子モデル

C_{in}、最大オン抵抗 R_{max}、最小オン抵抗 R_{min}、最大遅延時の拡散容量 C_{dmax} および最小遅延時の拡散容量 C_{dmin} でモデル化される。

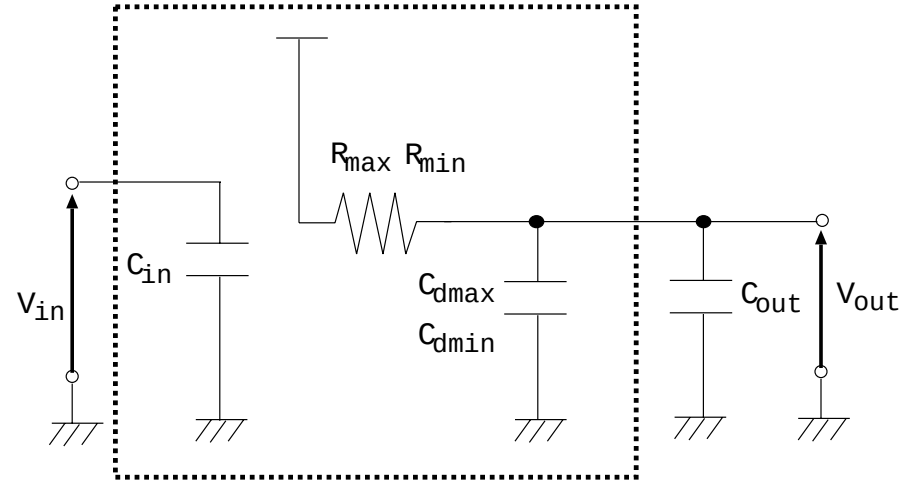

図 3.12: ウェーブ動作回路評価用論理素子モデル

3.3.2 回路のモデル化

配線は、その各部分で抵抗分 R_l、容量分 C_l およびインダクタンス分 L_l によってモデル化される。Elmore RC Delay[25] モデルなど、半導体チップ上での回路評価では従来から配線のインダクタンス分は無視され、R, C 成分のみからなる回路としてモデル化される。また配線を集中定数化することにより、配線経路によらない評価が行われる場合もある。しかしデバイスの縮小および回路の高速化が進むにつれて、インダクタンス分や回路中

に分散する配線抵抗分が無視できなくなってきている。より正確な遅延評価を行うため、配線経路を考慮した遅延評価や、Elmore のモデルをインダクタンス分も考慮したモデルに拡張した配線評価方法も提案されている [26]。

本研究では回路のモデルとして、配線経路を考慮に入れたモデルである Distributed RC Delay Model を使用する [15]。このモデルでは、抵抗分 R_l および容量分 C_l は配線経路に沿って分布しているとみなされ、そのようなモデルに対して上界／下界を与える式が提案されている。図 3.13 に Distributed RC Delay Model を示す。このモデルでは配線経路を考慮した測定結果を得ることができ、かつ各ピン毎での伝搬遅延の違いを考慮できる。さらに最大／最小伝搬時間を測定することができるため、ウェーブ設計を行う際に都合が良い。表 3.2 に導出式を示す。ここで R_{ki} はノード k,i のパスで共通部分の抵抗の和、C_k はノード k での容量分 (配線容量)、T_p, T_{Di}, T_{Ri} は時定数で、以下の定義からなる。また電圧は $0 \leq v_i(t) \leq 1$ に規格化された値を用いる。65%遷移での伝播時間を評価するのであれば、$v_i(t) = 0.65$ を代入する。

$$T_P = \sum_k R_{kk} C_k \quad \text{(回路毎に定まる)} \tag{3.18}$$

$$T_{Di} = \sum_k R_{ki} C_k \quad \text{(各ノード毎に定まる)} \tag{3.19}$$

$$T_{Ri} = \frac{1}{R_{ii}} \sum_k R_{ki}^2 C_k \quad \text{(各ノード毎に定まる)} \tag{3.20}$$

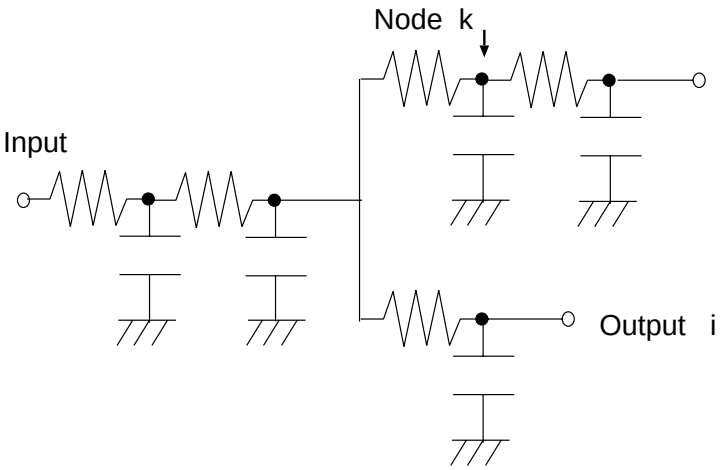

図 3.13: Distributed RC Delay Model

3.3.3 回路モデル評価

以上のように、本研究では各素子を入力容量／最大最小オン抵抗／拡散容量によりモデル化し、配線を Distributed RC Delay Model でモデル化する。このモデルを使用して、

表 3.2: Distributed RC Delay Model における各出力ピンでの遷移時間を求める式

下界

$v_i(t)$	t
$v_i(t) \leq 1 - \frac{T_{Ri}}{T_P}$	$T_{Di} - T_P\,(1 - v_i(t))$
$1 - \frac{T_{Ri}}{T_P} \leq v_i(t)$	$T_{Di} - T_{Ri} + T_{Ri} \ln \frac{T_{Ri}}{T_P(1 - v_i(t))}$

上界

$v_i(t)$	t
$v_i(t) \leq 1 - \frac{T_{Ri}}{T_P}$	$T_{Di} - T_P\,(1 - v_i(t))$
$1 - \frac{T_{Ri}}{T_P} \leq v_i(t)$	$T_{Di} - T_{Ri} + T_{Ri} \ln \frac{T_{Ri}}{T_P(1 - v_i(t))}$

素子および配線からなる回路を、*RC* 成分が至るところに分布した回路とみなして遅延評価を行う。しかし正確な波形を求めるのであれば、各素子での非線形な挙動をモデルにする必要があり、分布 *RC* 回路は回路方程式を立てて解く必要がある。ここではモデル化した回路と HSpice によるシミュレーションとを比較することで、構築した回路モデルの特性を評価した。図 3.14 に評価対象とした回路を示す。ここで使用している配線抵抗／配線容量／インバータモデルは、先に示した各パラメータに準じた値を使用している。またこの検証では動作電圧は 1.3V とし、オン抵抗は一定、遅延伝搬時間は各素子の出力側の電圧遷移が 65%に達した段階で評価した。回路中のノード 5 にて測定した波形を図 3.15 に示す。

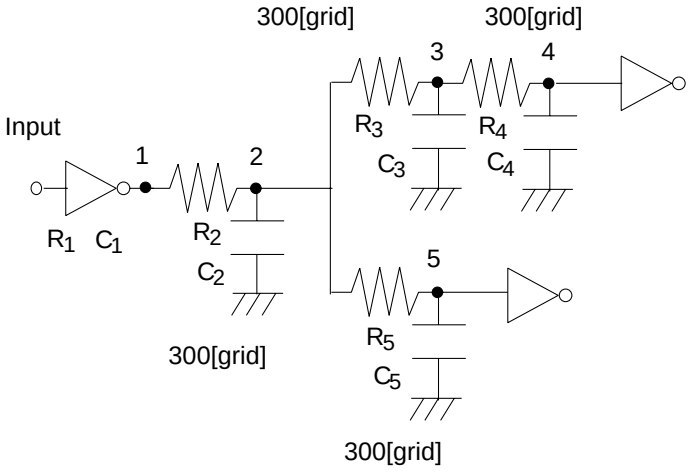

図 3.14: 評価対象とした回路

構築したモデルに関して、まず HSpice により数値解析を行った結果と、Distributed RC

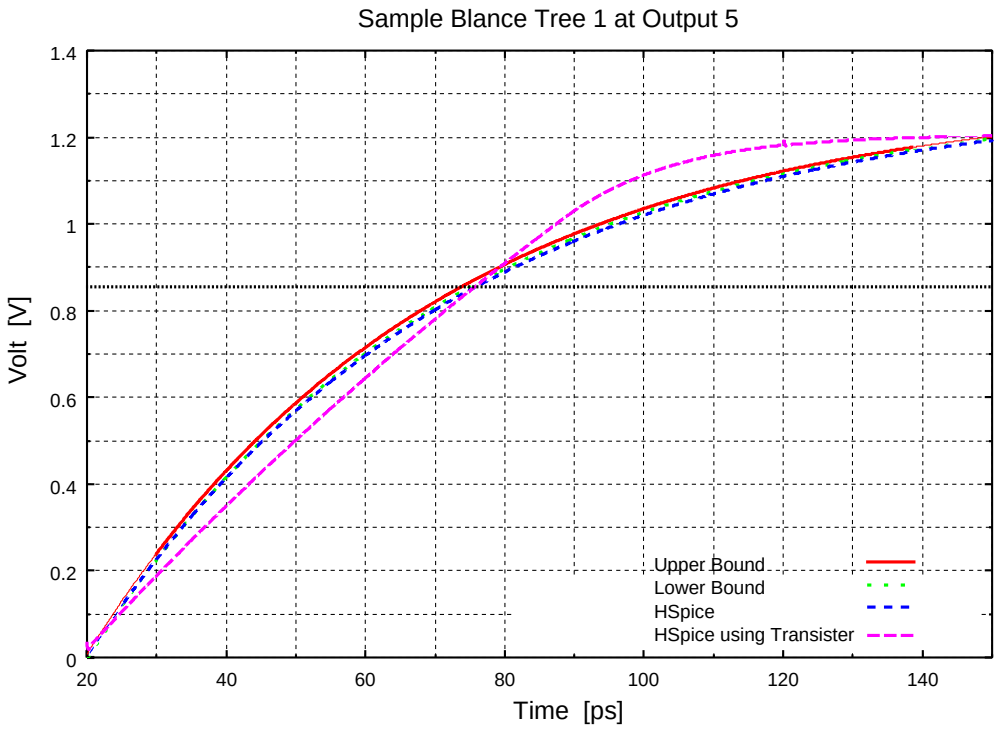

図 3.15: ノード 3-4 間の長さが 300grid の場合の、ノード 5 での電圧波形

Delay Model での上界／下界を評価した結果とを比較する。この比較では、HSpice での結果が Distributed RC Delay Model での上界／下界の範囲に収まっていることが確認できる。一方で素子のモデルに HSpice による動的なトランジスタ特性を使用した場合と比較すると、65%付近での遷移を境に双方の逆転現象が起こる。この現象は、Distributed RC Delay Model による解析では RC の単一指数関数によって挙動が記述され、動的なトランジスタ特性をそのまま反映することができないことによって起こる。また上界／下界の差は非常に小さく、1ps 程度である。これは配線の各部分での配線抵抗に対して素子のオン抵抗が非常に大きいため (配線抵抗の総和に対して、オン抵抗の大きさは約 20 倍)、回路の遅延に影響する抵抗分は実質的にはオン抵抗のみであることを示している。同じ理由で、出力側の各端子での伝搬遅延の違いもまた非常に小さい。実際ピン間での遅延の違いに関して、図 3.14 の回路ではノード 4 とノード 5 で伝搬遅延の差は 1ps 以内であった。さらに長い配線では、配線抵抗による影響が大きくなる。表 3.3 にノード 3-4 間の長さを変えた場合での、ピン毎の遅延時間を示す。配線長が長くなるにつれ上界および下界の差が大きくなり、また配線の“ かたより ”が大きくなるにつれ出力ピン同士の遅延差が大きくなることがわかる。

表 3.3: 図 3.14 の回路にて、ノード 3-4 間の長さを変えた場合の遅延時間

	ピン 4		ピン 5	
ノード 3-4 間の長さ [grid]	上界 [ps]	下界 [ps]	上界 [ps]	下界 [ps]
300	67	66	67	66
1000	104	101	103	99
3000	215	208	213	189

3.4 遅延要因による遅延差の分類

以上から回路中で発生する遅延差は、以下のように二つに大別することができる。

1. 一つのパス内で生じる遅延差
   (a) 多入力素子における入力の同時遷移数の違いにより生じる遅延差
   (b) 動作条件の変化によって生じる遅延差
2. 複数のパス間で生じる遅延差
   (a) パス毎の素子数／素子の種類／配線長の違いにより生じる遅延差

要因 1-(a) に関しては、多入力素子での入力における同時遷移数の違いによって、出力負荷を駆動する MOSFET の数が異なることにより遅延差が生じる。要因 1-(b) に関しては、動作温度／動作電圧の変動に対して MOSFET の特性が変化することにより遅延差が生じる。一方要因 2-(a) に関しては、パス毎に経路が異なることにより遅延差が生じる。このことは、パス毎の素子数／素子の種類／配線長によって信号伝播の“ 距離 ”が異なると考えることができる。パス毎の素子数／素子の種類／配線長によって仮想的に定まるパスの“ 距離 ”を、パスの**長さ** (Length) と言うことにする。

3.5 結論

本章では VLSI 上での回路をシミュレーションするにあたり、前半では MOSFET およびデバイスパラメータの設定および計算方法を示し、後半では遅延評価法の検討を行った。

MOSFET および各層に関するパラメータは、現在実際に使用／検討されているパラメータを参考にしながら、基本的には物理的な要因を元に計算により求めた。MOSFET に関しては、簡略化したモデルを構築し各パラメータを設定、それらの値から各 MOSFET モデルで使用されている式などを参考に各容量分を算出した。各層でのパラメータも同様に、現在実際に製造されかつデバイス写真が公開されているデータから簡略化したモデルを構築し、容量分／抵抗分を算出した。その他必要最低限のパラメータに関しては、実際にチップ製作に使用され、現在公開されているデバイスパラメータから得た。

遅延評価法の検討に関しては、まず素子および回路のモデルを提示し、先に構築／算出した各パラメータを元に実際の回路に近い値で遅延を計測、評価を行った。構築した遅延評価モデルは回路の伝播波形を正確にシミュレーションするものではなく、ある一定電圧に達するまでの時間を計測するモデルである。配線経路による差異を評価できるモデルであるので、配線抵抗がより支配的になるような領域でもこのモデルを使用して遅延評価を行うことができる。

最後に遅延差をその性質により分類した。この内本遅延均衡化手法がどの領域をターゲットとするかについては、次章以降で明らかにする。

第4章　CMOSウェーブパイプラインの可能性に関する考察

前章では実際に回路を評価するためのモデルを構築し、ディープサブミクロンデバイス下でのパラメータおよび評価方法を示しその検討を行った。それを元に本章ではディープサブミクロンデバイス下でCMOS論理を使って回路をウェーブ動作させることの可能性に関する議論を行う。CMOS上でのウェーブパイプラインの可能性に関する議論は、1995年にNowkaらによって発表されている[18]。しかしNowkaらによる議論はトランジスタのサイズが大きい領域におけるMOSFETモデルでの議論であり、ディープサブミクロンデバイスを前提とした議論ではない。本章ではまずNowkaらによるCMOS論理上でのウェーブパイプラインの可能性に関する議論を示す。次に可能性を議論するためのより簡単な評価方法を提案する。この章の最後に、実際にチップ製作に使用され現在公開されている各物理パラメータを使用して、ディープサブミクロンデバイス下でのウェーブパイプラインに関する可能性を議論する。

4.1　CMOS論理でのウェーブパイプラインの可能性に関する議論

まずCMOS論理でのウェーブパイプラインの可能性に関する議論を展開する。始めにNowkaらの長チャネルモデル上での議論を示す。次により複雑なモデルで話を進めるために、CMOS論理上での理想的なウェーブパイプライン回路を定義し、その際に新しいパラメータを導入する。

4.1.1　Nowkaらによる先行研究

NowkaらはMOSモデルとしてIDS Level 3モデルを使用して、CMOS上でのウェーブパイプラインの可能性に関するテクニカルレポートを発表している[18][17]。それによると、25C-125C,4.5V-5.5Vの範囲ではウェーブ度の限界は2であるという見解である。以下、Nowkaらの議論を追う。

回路中のあるパスiに関する伝播遅延時間T_iは、動作電圧V,動作温度τおよびデバイスによるパラメータPによって決まる。ここでパスの伝搬遅延は、ラッチでのセットアッ

プ時間およびホールドタイム時間、クロックスキューなどを考慮せず組合わせ回路の伝播遅延のみにより決定すると仮定する。この時回路の伝播遅延を以下の式で表す。

$$T_i = T_i(V, \tau, P) \tag{4.1}$$

回路の各パスでの伝播遅延時間には互いにばらつきがある。ある動作電圧 V ／動作温度 τ および回路に関する各パラメータ P の下で、回路中伝播遅延が最大となる場合の値 T_{max} および最小となる値 T_{min} を以下のように関連づける。

$$T_{max}(V, \tau, P) = \mathcal{A} T_{min}(V, \tau, P) \quad [\mathcal{A} \geq 1] \tag{4.2}$$

ここで定数 $\mathcal{A}$ は[1]、パスの長さの違いに依存する遅延差に関する遅延均衡度合いを示し、$\mathcal{A} = 1$ の場合は、ある動作条件では回路中全てのパスの遅延時間が等しいことを示す。

次に最小遅延パスに関して、動作電圧や動作温度が変化することにより伝播遅延が異なることを考える。動作電圧が最大でかつ動作温度が最低であるとき、伝播遅延は最小となる。一方、動作電圧が最小でかつ動作温度が最大であるとき、伝播遅延は最大となる。回路中最小遅延となるパスについて、動作温度の違いによる二つの伝播遅延時間に関してパラメータ $\mathcal{B}$ を導入し[2]、以下のように関係式を定義する。

$$\mathcal{B} = \frac{T_{min}(V_{min}, \tau_{max}, P)}{T_{min}(V_{max}, \tau_{min}, P)} \quad [\mathcal{B} \geq 1] \tag{4.3}$$

定数 $\mathcal{B}$ は動作温度／動作電圧の変化によって生じる一つのパス内での遅延均衡化度合いを示し、$\mathcal{B} = 1$ の場合は動作温度／動作電圧の変化による一つのパスでの遅延差は生じないことを意味する。以上,(4.2),(4.3) 式から、動作条件を含めて回路中最大となる伝播遅延 $T_{max}(V_{min}, \tau_{max}, P)$ および最小となる伝播遅延 $T_{min}(V_{max}, \tau_{min}, P)$ に関して以下の式が成り立つ。

$$T_{max}(V_{min}, \tau_{max}, P) = \mathcal{A}\mathcal{B} T_{min}(V_{max}, \tau_{min}, P) \tag{4.4}$$

遅延差 Δ は以下の式で定義される。

$$\begin{aligned} \Delta &= T_{max}(V_{min}, \tau_{max}, P) - T_{min}(V_{max}, \tau_{min}, P) \\ &= (\mathcal{A}\mathcal{B} - 1) T_{min}(V_{max}, \tau_{min}, P) \end{aligned} \tag{4.5}$$

従ってパイプライン動作する回路の一ステージに一度に入る波の数 (ウェーブ数) は、パラメータ $\mathcal{A}, \mathcal{B}$ を使用して以下のように表せる。

$$\begin{aligned} N &= \frac{T_{max}(V_{min}, \tau_{max}, P)}{\Delta} \\ &= \frac{\mathcal{A}\mathcal{B}}{\mathcal{A}\mathcal{B} - 1} \end{aligned} \tag{4.6}$$

[1]Nowka らの原論文では α が使用されているが、α バッファと区別するために本論文では $\mathcal{A}$ を用いる。
[2]Nowka らの原論文では β が使用されているが、β バッファと区別するために本論文では $\mathcal{B}$ を用いる。

パスの長さの違いにより生じる遅延差が 0 である場合は $\mathcal{A} = 1$ であり、

$$N = \frac{\mathcal{B}}{\mathcal{B} - 1} \tag{4.7}$$

で表せる。図 4.1 に各パラメータとウェーブ数との関係を示す [18]。$\mathcal{A}$ が 1 近い、すなわちパスの長さの違いにより生じる遅延差が小さい場合に、$\mathcal{B}$ を 1 に近づける、すなわち動作条件を安定させて遅延差を縮めることによる効果が非常に大きい事を示している。また式 (4.6) は $\mathcal{A}, \mathcal{B}$ に関して対称であるため、その逆もまた真である。つまり動作条件によって生じる遅延差が小さいとき、パスの長さの違いにより生じる遅延差を縮める効果が大きいことも意味している。また本論文では、ウェーブ数の逆数を取ったものを**圧縮比**として新しく定義する。圧縮比 $\mathcal{C}$ は最大遅延に対する遅延差の比を示し、最大遅延で駆動させた場合と比べて遅延差で駆動させた場合のクロックスピードの高速化率を示す。

$$\mathcal{C} = \frac{\mathcal{AB} - 1}{\mathcal{AB}} \tag{4.8}$$

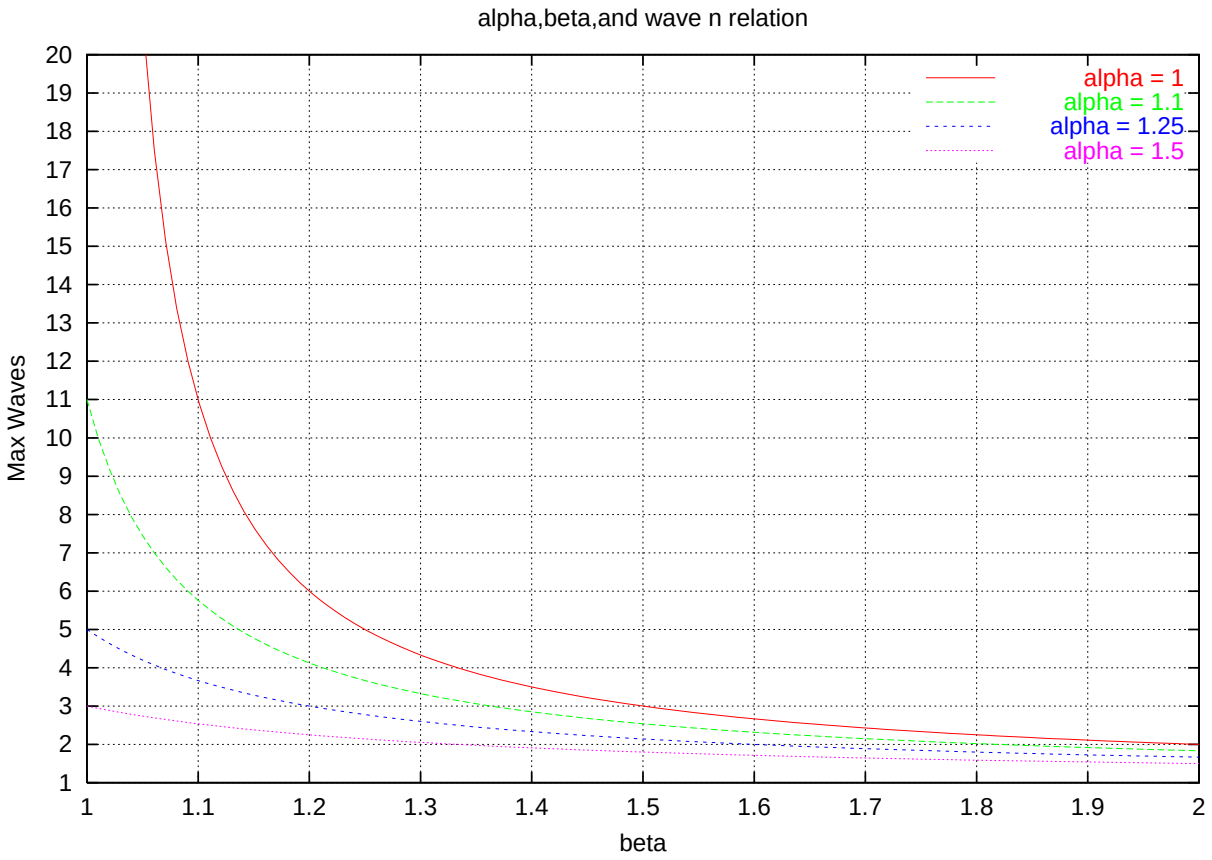

図 4.1: 各パラメータとウェーブ数の関係 [18]

Nowka らによる、MOS モデルとして IDS Level3 モデルを使用した場合に関する考察では、移動度に関する動作温度特性およびドレイン電流に関する動作温度／動作電圧依存性を考慮している。長チャネルモデルでは、線形領域でのドレイン電流式 I_{ds} は次式で表される。

$$I_{ds} = \mu_0 C_{ox} \frac{W_{eff}}{L_{eff}} \left(V_{gs} - V_{th} - \frac{1}{2} V_{ds} \right) V_{ds} \tag{4.9}$$

以下最大／最小動作温度を τ_{max}, τ_{min} および最大／最小動作電圧を V_{max}, V_{min} と定義する。移動度に関しては温度特性として以下の関係式を用いる。

$$\mu_0(\tau_{min}) = \mu_0(\tau_{max}) \left(\frac{\tau_{min}}{\tau_{max}} \right)^{-1.5} \tag{4.10}$$

一方式 (4.9) において V_{ds} を一定とすると、ドレイン電流 I_{ds} はゲート／ソース間電圧 V_{gs} に比例する。故に $\mathcal{B}$ は温度／電圧に関して以下の関係を持つ。

$$\mathcal{B} \approx \left(\frac{\tau_{min}}{\tau_{max}} \right)^{-1.5} \frac{V_{max}}{V_{min}} \tag{4.11}$$

Nowka らはさまざまな動作条件下で $\mathcal{B}$ の値を求め、評価を行った。表 4.1 に Nowka ら対象としたデバイスでの結果を示す。ここで最大ウェーブ数は、$\mathcal{A}$ を 1 とした場合の値である。パス間での遅延差が生じない場合、最大ウェーブ数 2 が限界であるとしている。また $\mathcal{B}$ の値から、最大遅延に対して 41.2% まで遅延差を縮めることができると見ることができる。

表 4.1: 条件別による $\mathcal{B}$ パラメータの値 [18]

温度条件	電圧条件	$\mathcal{A}$	$\mathcal{B}$	最大ウェーブ数	圧縮比
25C	5V	1	1	6	-
25C	4.5V-5.5V	1	1.2	4	0.167
25-125C	5V	1	1.4	2	0.286
25-125C	4.5V-5.5V	1	1.7	2	0.412

4.1.2 CMOS 論理上での理想的なウェーブパイプラインに関する議論

Nowka らの議論は、トランジスタのサイズが大きいデバイスでの議論であった。現行のデバイスは最早単純なデバイスモデルでは解析不可能であり、上記の議論は成り立たない。そこで次にディープサブミクロンデバイスの場合における検証を試みるため、より簡単な評価を提案する。

回路からある素子とその素子につながる配線／素子負荷を取出す。この部分回路が単純な Lumped model で記述される場合、発生する遅延 T は以下のように記述できる。ここ

で R_{on}, R_l はそれぞれ駆動素子のオン抵抗および配線抵抗を、C_d, C_l, C_g は駆動素子の拡散容量、配線容量および負荷側の素子のゲート容量の総計を示す。

$$T = RC = (R_{on} + R_l)(C_d + C_l + C_g) \tag{4.12}$$

R_{on} は素子のドレイン／ソース間電圧 v_{ds} およびドレイン／ソース電流 i_{ds} によって以下のように記述できる。

$$R_{on}(V, \tau, P) = \frac{dv_{ds}}{di_{ds}} \tag{4.13}$$

至近配線では配線抵抗は無視できるものとすると、ある素子が発生する伝播遅延は以下の式で記述できる。

$$T = R_{on}(V, \tau, P)(C_p + C_l + C_g) \tag{4.14}$$

インバータ 1 つが駆動する配線負荷および素子負荷で発生する遅延に着目し、式 (4.14) からインバータの出力側で発生する遅延差に関するパラメータ $\mathcal{B}'$ を以下のように定義する。

$$\begin{aligned} \mathcal{B}' &= \frac{T(V_{min}, \tau_{max}, P)}{T(V_{max}, \tau_{min}, P)} \\ &\approx \frac{R_{on}(V_{min}, \tau_{max}, P)}{R_{on}(V_{max}, \tau_{min}, P)} \end{aligned} \tag{4.15}$$

あるデバイスで得られるインバータに関するオン抵抗の最大／最小値を得ることができれば、インバータが生じる遅延差に関するパラメータ $\mathcal{B}'$ を得ることができる。

今、CMOS 上でのウェーブパイプラインを行う場合における理想的な回路を考える。ここである回路において理想的にウェーブ化された回路とは、ある機能を持った回路を入力とした際に、同様の機能を実現しかつ圧縮比が最低、すなわちウェーブ数が最大となる回路と定義する。圧縮比が最低であっても、動作速度が最高速度であるとは必ずしも限らない。このとき理想的な回路とは、全ての素子がインバータで構成されている回路で全てのパスで段数が等しく、かつ各素子が持つ配線／素子負荷が全く等しいときである。その場合は回路全体の遅延差の圧縮比は、インバータで生じる遅延差の圧縮比になる。故に理想的な回路では以下の式が成り立つ。

$$\mathcal{B} = \mathcal{B}' \tag{4.16}$$

移動度やしきい値電圧その他各種のパラメータに関する温度依存性／電圧依存性をオン抵抗に内包できるため、デバイスに関する実際のパラメータが得られる場合は、インバータのオン抵抗を調べるという単純なシュミレーションにより、理想的な場合の $\mathcal{B}$ の値を得ることができる。

4.2 実際にチップ製作に使用されたパラメータでの検証

実際にチップ製作に使用されたパラメータを用いて検証を行った。使用した MOS パラメータは、第 4 章でパラメータの検証に用いたものと同様、MOSIS[1] にて公開されてい

る TSMC 社の 0.18μm プロセスでのパラメータ (Run:T29B) を使用した。インバータの NMOS 側のゲートサイズ W_n を 3 種類用意し、各サイズで遅延差がなるべく生じないように PMOS 側のゲートサイズ W_p を設定、評価した。また動作条件は 0.9V-1.0V,25C-70C を想定している。図 4.2 から図 4.4 にかけて、各サイズにおいて NMOS/PMOS とも最悪／最良条件で評価した vds-ids グラフを示す。また表 4.2 にそれらグラフをまとめた結果を示す。なお表中オン抵抗は、ドレイン · ソース間に動作条件中最大の動作電圧 V_{ds} がかかっている際の、ドレイン · ソース間電流 I_{ds} より求めた。

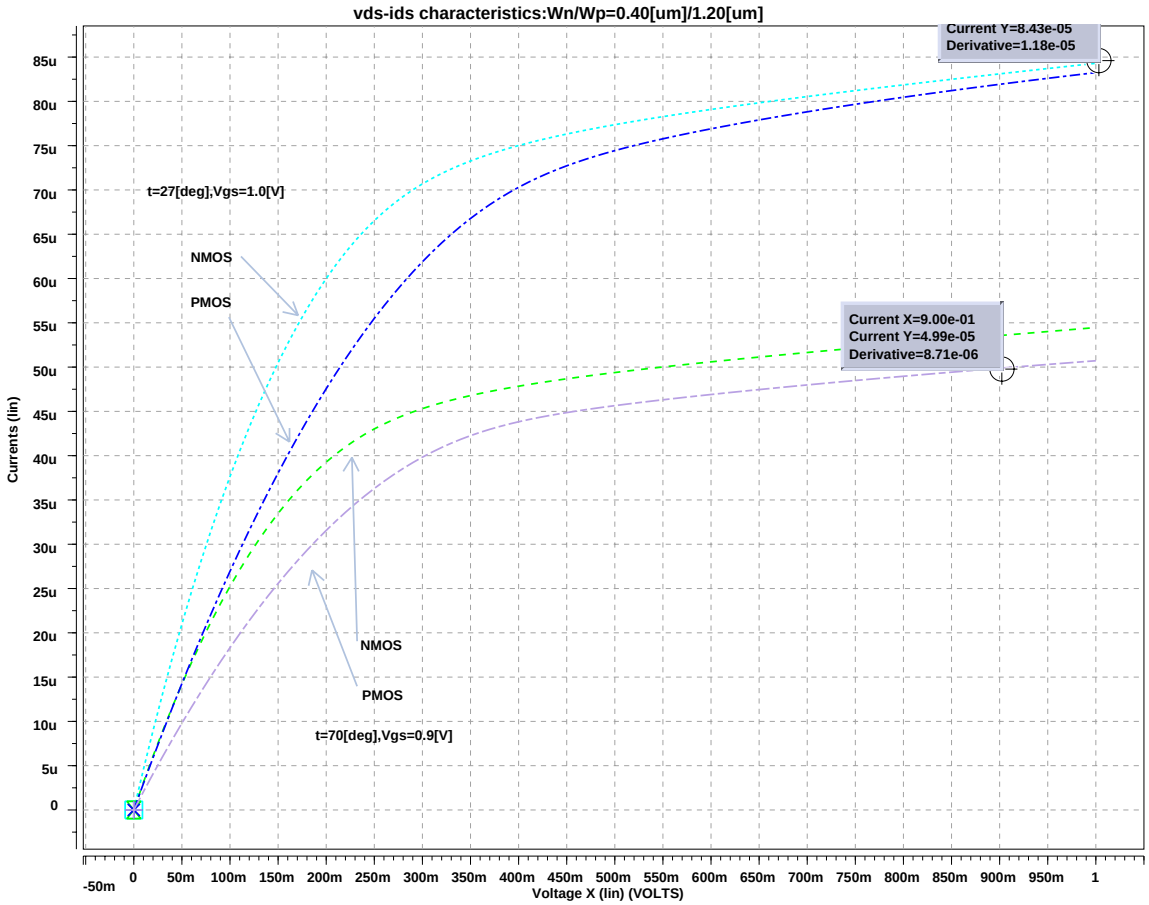

図 4.2: $W_n/W_p = 0.40\mu m/1.20\mu m$ でのインバータの $v_{ds} - i_{ds}$ グラフ

表 4.2: ゲートサイズによるインバータの各パラメータの値

W_n/W_p $[\mu m/\mu m]$	$R_{on-MAX}[k\Omega]$	$R_{on-MIN}[k\Omega]$	$\mathcal{B}'$	圧縮比
0.40/1.20	11.86	20.04	1.69	0.408
0.60/1.80	8.26	13.81	1.67	0.402
1.00/2.80	5.43	9.09	1.67	0.402

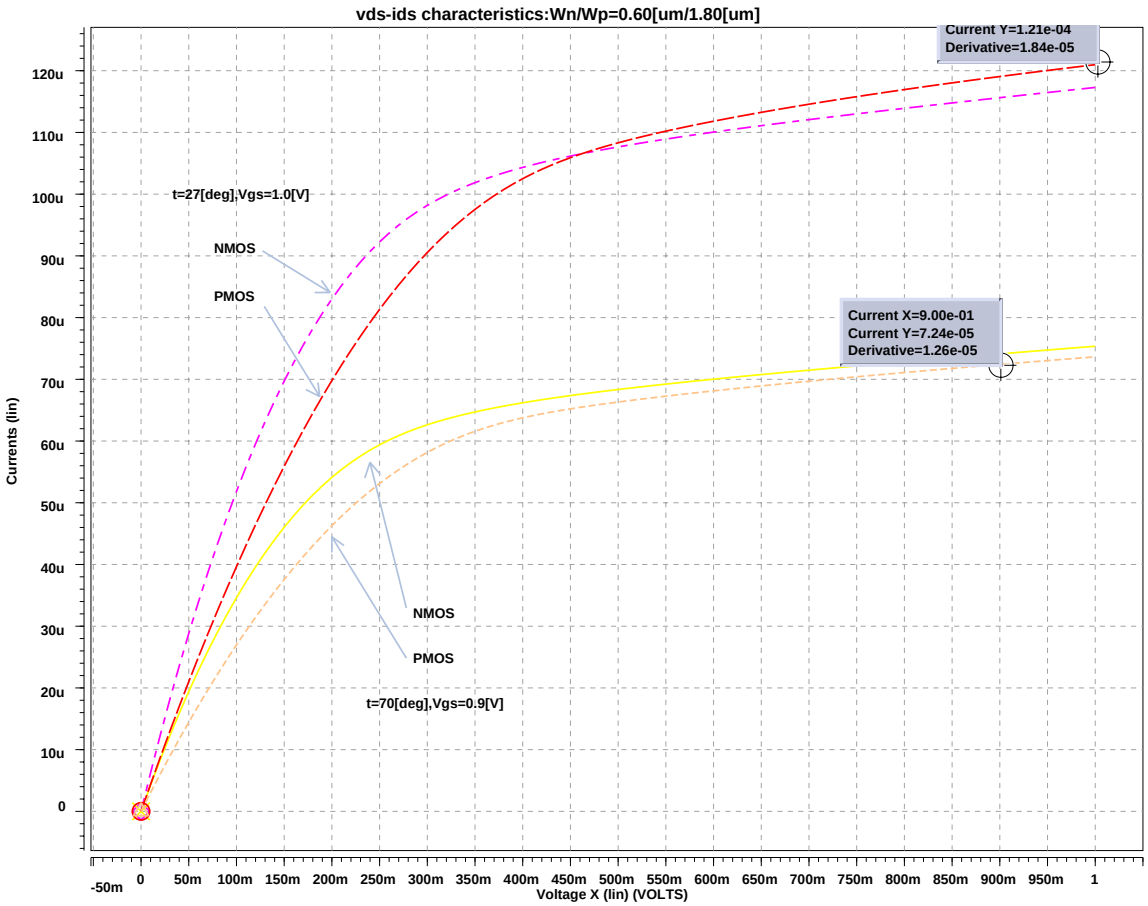

図 4.3: $W_n/W_p = 0.60\mu m/1.80\mu m$ でのインバータの $v_{ds} - i_{ds}$ グラフ

最大遅延を遅くすることなく理想的に回路の遅延を均衡化できた場合、最大遅延に対して約 40%のレイテンシで動作させる事が可能である。ゲートサイズ比が一定でないのは、素子が小さい領域、すなわちドレイン電流が小さい領域では、縮小化による様々な物理的影響を受けやすいためである。オン抵抗で評価することによって、こうした問題を全てオン抵抗に内包することができる。なお動作条件は Nowka らのに対して甘くした。

4.3 結論

以上実際に使われたパラメータを元に、CMOS 上でのウェーブパイプラインの可能性に関する検証を試みた。まず先行研究として長チャネル MOSFET モデルでの議論を示し、要因別の遅延均衡化度合いを示すパラメータとして、論理段数や配線長の違いにより生じる遅延差に関するパラメータ $\mathcal{A}$ および動作条件によって一つのパス内で生じる遅延差に関するパラメータ $\mathcal{B}$ を導入した。次にディープサブミクロンデバイス下で議論を行うために、より簡単な評価によるパラメータ $\mathcal{B}'$ を提示した。本提案による遅延均衡化手法に

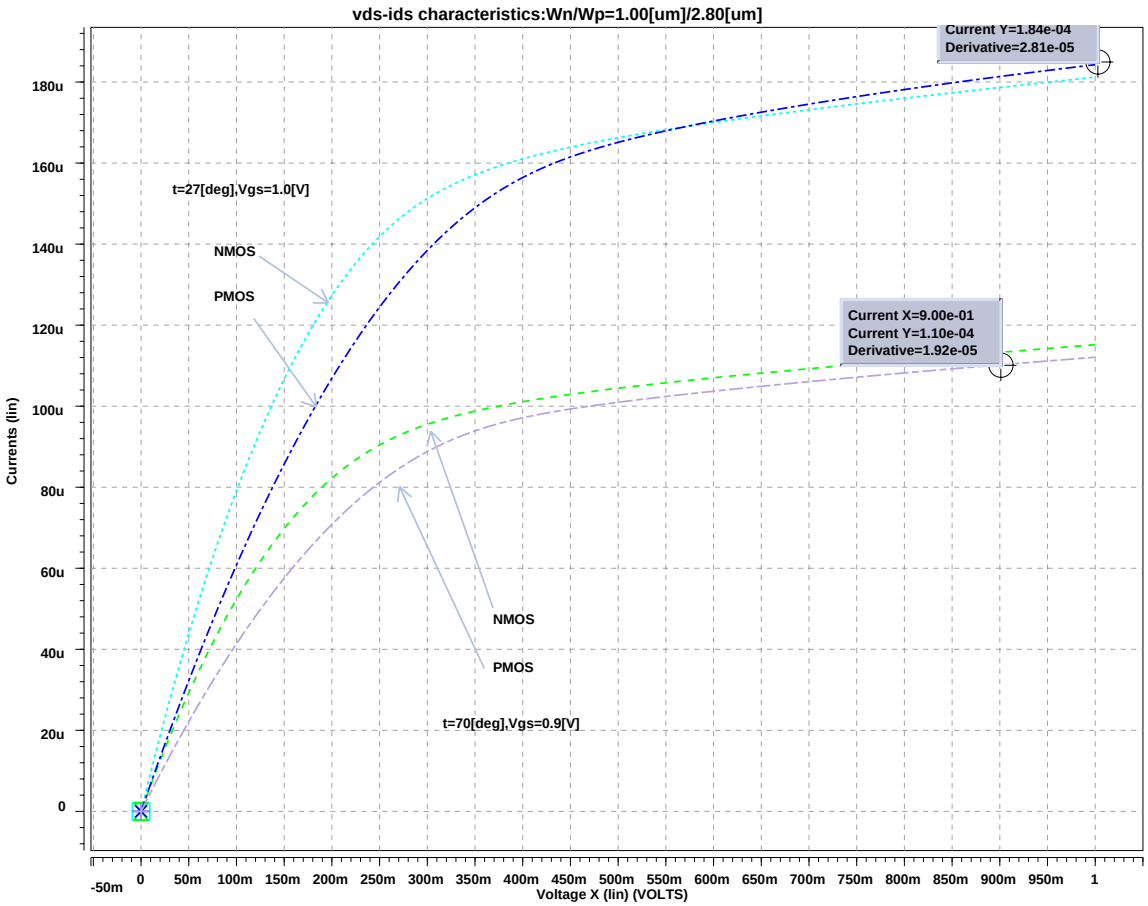

図 4.4: $W_n/W_p = 1.00\mu m/2.80\mu m$ でのインバータの $v_{ds} - i_{ds}$ グラフ

よる限界を述べ、理想的な回路では $\mathcal{B} = \mathcal{B}'$ であることを示した。最後に、実際にチップ製作に使用されているパラメータからシミュレーションにより β' を求め、現行のプロセスでも理想的には最大遅延の約 40%のレイテンシで動作させることが可能であることを示した。

本章での議論から、次の考察を得ることができる。動作環境をより安定することができれば、$\mathcal{B}$ の値を 1 に近づけることができる。故にパスの長さの違いにより生じる遅延差を取り除くことができれば、動作条件を安定させることでウェーブパイプラインアーキテクチャ上で高速な回路を設計することが可能である。本章での議論では、理想的な回路では $\mathcal{B} = \mathcal{B}'$ とした。しかし動作環境も含めた真の理想的な回路では

$$\mathcal{B} = 1 \tag{4.17}$$

である。次章以降、パスの長さの違いにより生じる遅延差を取り除く手法を提案する。

第5章　素子の遅延差に着目した遅延均衡化手法の提案

本章では、本研究で提案する遅延均衡化手法の概略について述べる。まず簡単化のため、動作環境は配置配線問題を考慮する必要が無く、素子のみを考慮することで遅延が評価できる環境であると仮定する。またこの段階では動作温度／動作電圧は一定であると仮定し、酸化膜厚など各物理的パラメータに関して深い考察は行わない。以上の環境の下で1ビット全加算器を対象に遅延均衡化を行うことで、提案する手法の基本的な効果を見る。

全加算器程度小さな機能ブロック単位で遅延差を軽減する試みについて、Kishigami らはパストランジスタ論理を使用して行っている [12]。本研究での提案は、CMOS 論理に対して適用する戦略である。

5.1　基本戦略

まず本節では、発生する遅延差を要因別に分け、遅延均衡化に関するいくつかの手法を定義する。続いて遅延評価に関する既存の簡単なモデルを示し、ウェーブ動作を評価するためにそのモデルを再構築する。その後構築した遅延モデルを使用して、各戦略がどのように遅延均衡化に作用するのかを示す。最後に定義した各戦略を元に遅延均衡化戦略の概要をまとめる。

5.1.1　遅延均衡化戦略概要

第3章で示した遅延差の性質による分類を以下に再掲する。

1. 一つのパス内で生じる遅延差
   (a) 多入力素子における入力の同時遷移数の違いにより生じる遅延差
   (b) 動作条件の変化によって生じる遅延差
2. 複数のパス間で生じる遅延差
   (a) パス毎の素子数／素子の種類／配線長の違いにより生じる遅延差

本提案では、まず要因 1-a) により生じる遅延差を抑えることを考える。多入力素子が大きな容量を駆動する箇所にバッファを挿入することで、多入力素子で生じる遅延差を最小限に抑える。一つのパス内で生じる遅延差を抑えた段階で、複数のパス間で生じる遅延差を抑えるようにインバータを挿入することで、パス間での遅延差を抑える。

5.1.2 論理合成段階での遅延均衡化戦略

挿入 (Inserting) 戦略

本研究で提案するアルゴリズムでは、まずバッファ挿入をその役目に応じて区別し、バッファ挿入戦略を定義する。挿入戦略を以下に、概略図を図 5.1 に示す。

α **挿入** :多入力素子の負荷を減らすように挿入する。

β **挿入** :素子の反応時間を抑えるように挿入する。

γ **挿入** :パス間で遅延均衡化させるように挿入する。

δ **挿入** :論理的に整合を取るように挿入する。

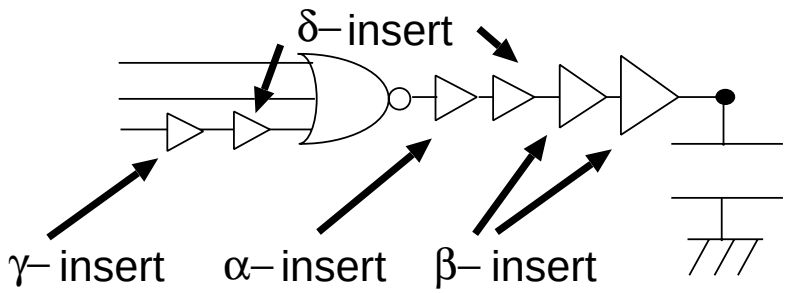

図 5.1: 各挿入戦略の概要

分解 (Decomposition)、再評価 (Resizing) 戦略

これらバッファは CMOS 回路ではインバータである。故にパス上に奇数個バッファを挿入することは通常許されない。また各論理素子の設計法によっては、NAND や NOR 素子により生じる遅延が異なることも起こりうる。このような場合に対し柔軟なバッファ挿入戦略を可能にするため、NAND → NOR,NOR → NAND の**分解** (Decomposition) 戦略を定義し必要に応じて用いる。分解戦略ではまず必要に応じて該当素子を分解する (図 5.2)。このとき素子の分解によって出力側に生じたインバータは、そのまま α バッファとして使用できる。

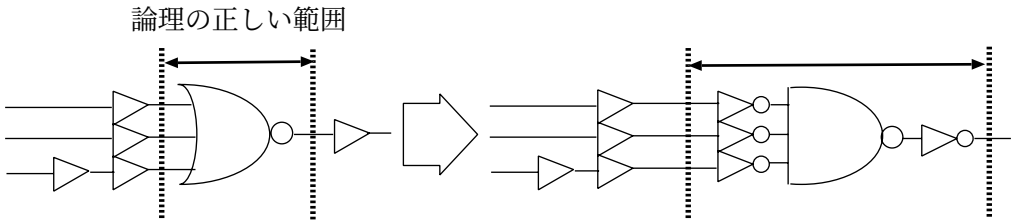

図 5.2: NOR → NAND にデコンポジションした状態

次に後ろのパスが奇数個のバッファからなるパスの場合、分解戦略によって生じたインバータをそのまま割り当てる。偶数個のバッファからなるパスの場合、論理的に整合を取るため δ バッファを挿入する。ここで δ バッファとして挿入する素子のゲートサイズは、負荷側の素子と同じ入力ゲート容量になるようにサイズを決定する。そうすることで、δ バッファの入力側につながる素子から見た負荷容量は変化せず、再度パスの遅延評価を行う必要がない。しかしこのようにして挿入した δ バッファは、論理から見て必要であるが遅延均衡から見た場合に無駄となる (図 5.3)。そこでインバータ列の遅延速度が早くなるように、必要に応じてパスの負荷側からゲートサイズを変更する**リサイジング** (Resizing) **戦略**を定義し、その影響を減らす努力を行う。

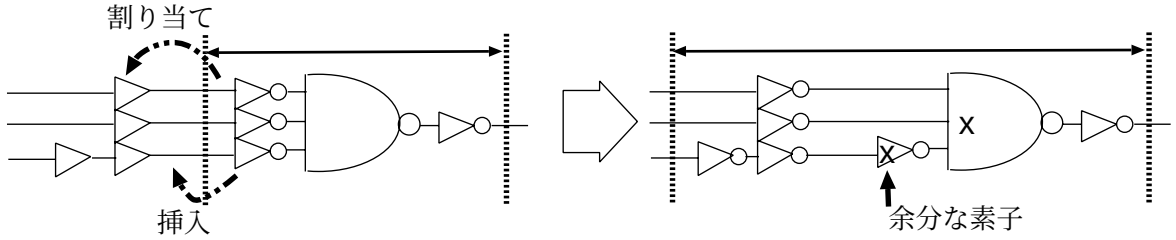

図 5.3: インバータの割当てが終了した状態

交換 (Exchanging)、再接続 (Reconnecting) 戦略

以上のバッファ挿入は、パスの入力側から各素子で発生する遅延差を最小にするように行っていく。そのため常にパスの負荷側で複数のパス間での遅延均衡化をとることになるので、遅延を稼ぐためにどうしてもサイズの大きなインバータ (γ, δ バッファ) を挿入する必要がある。そこで探索するパスにおいて挿入するインバータが決定した段階で、バッファをパスの前段に改めて挿入し直す**交換** (Exchanging) 戦略を定義する (図 5.4)。

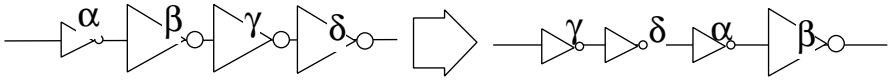

図 5.4: 交換 (Exchanging) 戦略

多入力素子では入力端子毎にオン抵抗および拡散容量が異なるため、入力端子毎に出力側での最大／最小遅延が異なる。この性質をモデル化するため、シュミレーションにより最大／最小遅延を生ずる際の拡散容量を入力端子毎に求めておく。そして入力までにおいて最大遅延が一番大きな信号線に対して、素子の出力側での最大遅延が一番小さくなる遷移を生ずる端子を割り当てる。こうすることで出力での遅延差を縮めることができる。この戦略を**再接続** (Reconnecting) **戦略**として定義する (図 5.5)。

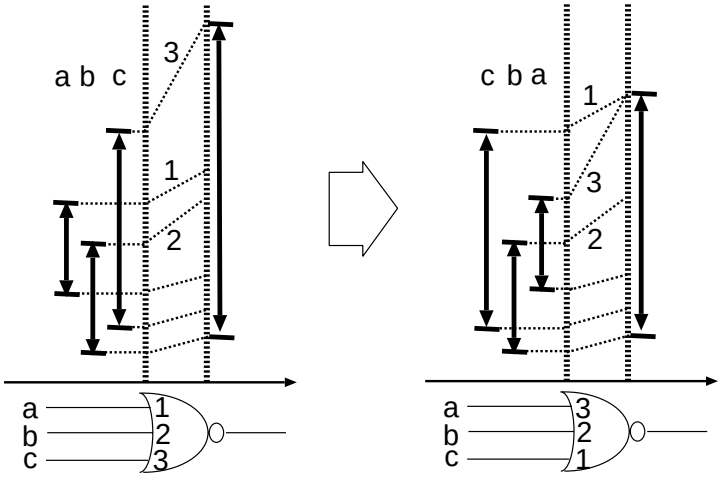

図 5.5: 再接続 (Reconnecting) 戦略

5.1.3 論理回路レベルでの回路評価手法

遅延評価は、HSpice で求めたパラメータにより Logical Effort の手法でモデル化し単純化を計った [13][14]。この手法では、論理素子を入力容量 C_{in}、立ち上がりの際のオン抵抗 R_r、立ち下がる際のオン抵抗 R_f および拡散容量 C_d によってモデル化する。しかし第 3 章でも述べた通り、ウェーブ化設計では発生する最大／最小遅延を評価できることが望ましい。そこで立ち上がりの際のオン抵抗 R_r および立ち下がる際のオン抵抗 R_f は、最大遅延時のオン抵抗 R_{max} および最小遅延時のオン抵抗 R_{min} に変更する。また拡散容量に関しても最大遅延時の拡散容量 C_{dmax} および最小遅延時の拡散容量 C_{dmin} に変更する。図 5.6 にモデルを再掲する。。以上のパラメータはチャネル幅 W から求めることができ、出力負荷を定めることで個々の素子ごとの最大遅延／最小遅延を評価する。

各素子での最大／最小遅延を求める式は次のようにモデル化する。g は各論理素子ごとの固有の値、h は入力負荷に対する出力負荷の比、p はインバータの入力容量に対する各論理素子ごと固有の寄生容量値の比である。また τ_{max}, τ_{min} はモデルインバータの最大／最小遅延の速度であり、これらのパラメータは HSpice から抽出する。実際の評価では τ の値は一つに統一し、他のパラメータを規格化して使用している。本章で使用した各パラメータに関しては付録 A.2.2 に示した。第 3 章で求めたオン抵抗に対して、高速で理想的

なデバイスを想定している[1]。

$$d_{max} = \tau_{max}(g_{max} \cdot h + p_{max}) \quad (5.1)$$
$$d_{min} = \tau_{min}(g_{min} \cdot h + p_{min}) \quad (5.2)$$
$$\tau_{max} = k_{max} R_{inv} C_{inv} \quad (5.3)$$
$$\tau_{min} = k_{min} R_{inv} C_{inv} \quad (5.4)$$

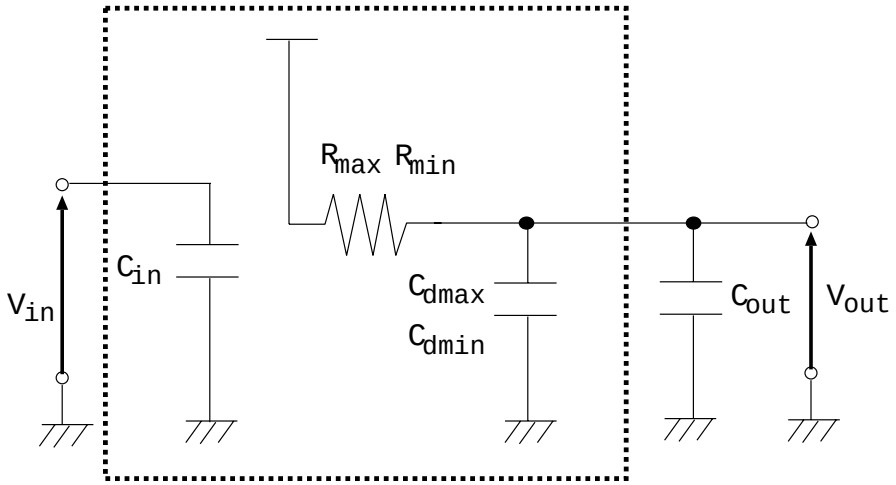

図 5.6: 論理素子のモデル化 (再掲)

このモデルでは式 (5.1),(5.2) が示すように、回路の遅延を単純に RC の積で求める。α バッファを挿入することによる効果は、入力容量に対する出力容量の比 h で説明できる。多入力素子が出力側に持つ負荷を減らすことで h を小さくでき、故に d_{max}, d_{min} の相対的な差を縮めることができる。同様に再接続戦略を行うことによる効果は、式 (5.1),(5.2) での寄生容量に関するパラメータ p によって説明できる。接続を変更することで p の値を変え、d_{max}, d_{min} の差を縮めることができる。但し拡散容量に対して出力負荷が大きい場合は、再接続戦略による効果は小さい。

5.1.4 遅延均衡化戦略

以上の各戦略を遅延均衡化戦略として次のように組み込むことで、遅延均衡化戦略を構築する。

ステップ 1 :遅延均衡の目標値、制約を決める。
一つの素子当たりの最大反応速度、多入力素子のゲートサイズの決定。

[1]最小インバータのオン抵抗 R_{on} が $1-2k\Omega$ 程度、動作環境が 27C、動作電圧が 1.3V で一定を想定している

ステップ 2 :入力側から素子を選択し、α, β バッファを挿入することで一つのパス内での遅延差を均衡化させる。
Decomposition を行うかどうかの決定。

ステップ 3 :γ, δ バッファを挿入して、複数のパス間での遅延均衡化を行う。
この段階でパスに挿入するバッファの数および位置が決まる。

ステップ 4 :Exchanging,Reconnecting,Resizing をこの順に行うことで、より遅延差を縮める努力をするとともにバッファサイズを小さくする。

ステップ 5 :負荷側の素子を選択して、ステップ 2 から繰り返す。

以上を入力側の素子から順に行っていくことで遅延均衡化を行う。ステップ 5 が終了すると論理素子による遅延均衡化が終了した段階となり、この後配置配線段階に移行する。

5.2 全加算器の遅延均衡化

前節で構築した遅延均衡化戦略に沿って、本節では全加算器に対して遅延均衡化を行うことでその効果を実証する。動作環境はこの章の最初で述べたように、動作温度および動作電圧が一定で、非常に高速に動作するトランジスタを持つ、ある種の理想的な状態を仮定したものである。使用したパラメータに関しては付録 A.2.2 の表 A.2 に載せた。

5.2.1 全加算器の評価

以上のプロセス下で全加算器の遅延均衡化を行った。図 5.7 に均衡化アルゴリズム適用前の回路を、図 5.8 にアルゴリズム適用後の回路を示す。表 5.1 には HSpice による性能評価を示す。図中の実数はインバータのゲートサイズを 3.40 としたときの各素子のゲートサイズ比を、各ギリシャ文字は基本戦略で述べた各戦略によって挿入されたバッファであることを示す。図 5.8 での 1 段目の 1-4 の各 NAND および 3 段目 3 の NAND 素子はそれぞれ元は NOR 素子だったものを Decomposition したものである。なお最終的な出力ピン S, C_OUT につながる負荷はインバータ 4 つ分を仮定し、入力信号 a, b, c に関する否定信号は回路の外から供給されていることを仮定する。

遅延均衡化前の最大遅延で動作する全加算器に比べて、2.94 倍の速度向上を達成したことを確認できた。ここで評価値の測定に関して、最大／最小の各遅延は出力電圧が 65%遷移した状態での最大／最小遅延を評価している。評価値では遅延差は 14.1ps となったが、実際に動作するのは 22ps であった。これは素子のトルグ周波数での速度よりも速くはウェーブ動作しないためである。実際にこの回路で最小となるトルグ速度を評価すると約 20-22ps であった。回路規模が大きくなり、最小トルグ速度よりも遅延差が十分大きくなる回路では、評価値に近い速度が得られる。また最大遅延の評価値は実測値に比べてタイ

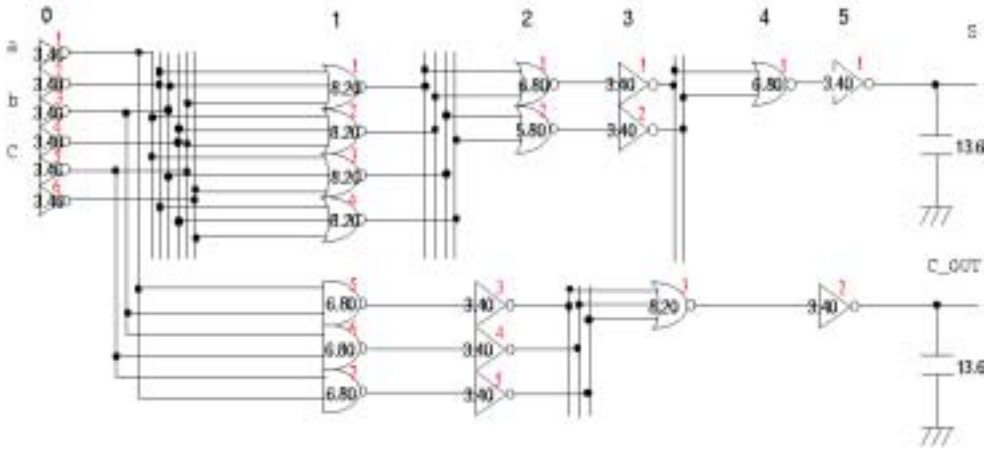

図 5.7: 遅延均衡化前の全加算器

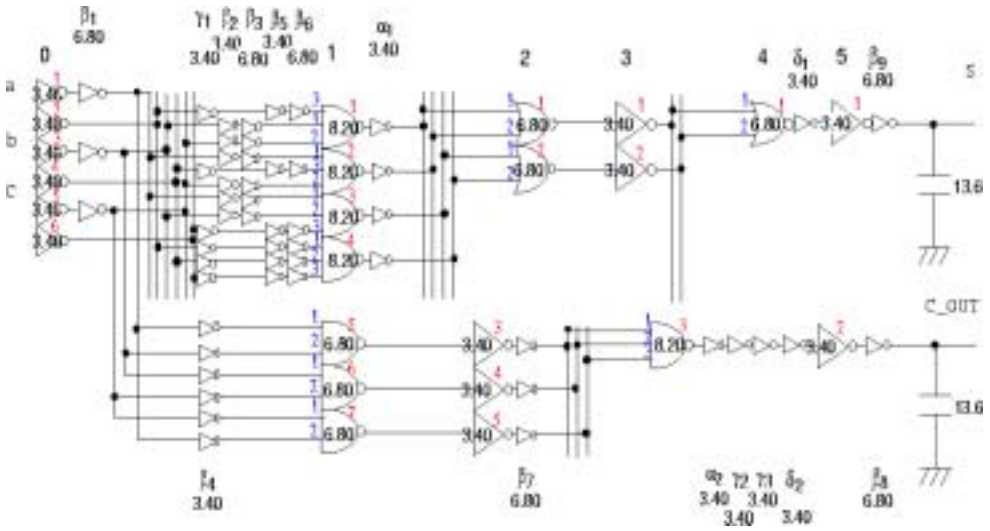

図 5.8: 遅延均衡化後の全加算器

表 5.1: BSIM3v2 モデルでの全加算器性能評価

	遅延均衡化前	ウェーブ版	
	通常動作時	評価値	HSpice
最大遅延時間 [ps]	64.7	67.4	67.1
最小遅延時間 [ps]	(32.2)	53.3	58.1
遅延差 ps]	(32.5)	14.1	11.0
クロック周期 [ps]	64.7	14.1	22

トな値になっているが、遅延差の評価としては HSpice での遅延差 11.0ps に対して 14.1ps と上回っており、安全な評価である。

5.3 結論

提案したアルゴリズムの最大のポイントは、CMOS 論理では避けられない多入力素子で生じる大きな遅延差を、インバータによって出来る限り抑えるところにある。さらに駆動素子としてもインバータを別に用意することで、回路で生じる遅延差をインバータが持つ遅延差に抑え込むという狙いがある。また本アルゴリズムは、多入力素子が大きな負荷を駆動するような回路において特に効果的である。今回対象とした全加算器では、多入力素子が駆動する負荷はせいぜい素子一つ程度である。また配線負荷を考慮していないため、多入力素子が特別大きな負荷を駆動することはない。配線を考慮しない段階での実験によって本研究で提案する遅延均衡化手法の概要を示し、ある程度の高速化達成および評価の正当性を示した。遅延差が発生する素子の直接負荷を減らし、大きな負荷を駆動する際には大きなインバータを用いるという本アルゴリズムの基本的な概念は、CMOS 論理に限らずパストランジスタ論理に対しても有効である。

しかし配線や配置の問題を含めた場合は、ここで提案した手法は効率的でない、実用的でないおそれがある。また配線負荷を考慮した場合における、本遅延均衡化戦略の効果を検証する必要がある。ここで指定した MOSFET に関するパラメータはあくまでも参考値であり、動作電圧／動作温度が一定という理想的な状態での測定である。故に各戦略別による効果や詳細な性能に関しては特別触れない。全加算器等の非常に小規模な回路で、配置配線問題が無視できる場合、この手法をそのまま適用することも可能である。

第6章　配置配線を含めた遅延均衡化手法の提案と評価

本章では、第5章で示した遅延均衡化手法を配置配線段階も考慮した手法として示す。最初に配置配線問題を考慮した場合の戦略に昇華させるための概要について示し、パス間での遅延均衡を取るための戦略と一つのパス内で生じる遅延差を抑えるための戦略とに分けて具体的な方法論を述べる。その後検討を行うため、4ビットALU(オリジナル回路の素子数111個、段数14段)に対して構築した戦略を適用し、評価を行う。

6.1　配置配線を考慮した評価方法と遅延均衡化手法

第5章で述べた論理素子のみを考慮した遅延均衡アルゴリズムでは、バッファ挿入の際に逐次遅延を計算していた。実際には配置配線段階で遅延を計測する場合、配線抵抗／容量を考慮する必要がある。しかし配置するまでは配線経路を決定することができす、正確な遅延を求めることができない。故に提案したアルゴリズムのままでは、配置配線段階でバッファ挿入を行う必要があるほか、その際挿入位置などにも気を配らねばならない。

そこで配置配線も考慮した遅延均衡アルゴリズムを構築しかつより実用的なものにするために、テクノロジマッピングを行う前のネットリストの段階で、位置情報を付加してしまうことを考える。通常ネットリストの段階では、回路データは素子の種別情報と結線情報を持つ程度である。ネットリストの段階で相対的な位置情報を付加することができれば、実配置前に遅延を見積もることができる。具体的には遅延評価の目安となる値を使って仮想的に配置してしまい、位置情報を含んだネットリストを生成することを考える。位置情報を含んだネットリストをそのまま配置配線すれば、自動的に遅延が均衡化された回路が生成される。そうすることでバッファ挿入のために論理合成段階へ戻ったり、あるいは配置配線段階で逐次遅延を計測したり、位置を気にしながらバッファを挿入することなく、論理合成段階から配置配線段階へとストレートに遅延を均衡化した回路を生成することが可能になる。

そのためにはネットリストの段階で、ある程度正確な遅延評価の目安となる値を定める必要がある。遅延は素子の位置情報や配線の経路など、実際の回路での物理的要因に基づくものである。故に少なくとも以下の事がネットリストの段階である程度予測がつけば良い。

1. **各素子の位置情報**

2. **配線の形状**

ネットリストの段階でもこれら二つの要素が十分容易に予測可能であり、かつ遅延均衡化に有利な配置配線手法を考える。

6.1.1 配線を考慮した遅延モデル

配線を考慮した遅延モデルとして、第 3 章で Distributed RC Delay Model を示した。しかし実際に Distributed RC Delay Model 上で評価するためには配線の経路情報が必要であり、ネットリストの段階でこれらを特定するのは困難である。図 6.1 に Distributed RC Delay Model に基づく遅延モデルを再度示す。回路は駆動側のオン抵抗／拡散容量および配線中の容量／抵抗、負荷側の容量分でモデル化される。モデル回路は一つの駆動ノードと複数の負荷ノード、および配線をモデル化した回路から成り、そのようにしてモデル化された回路を以後**配線木**と呼ぶ。

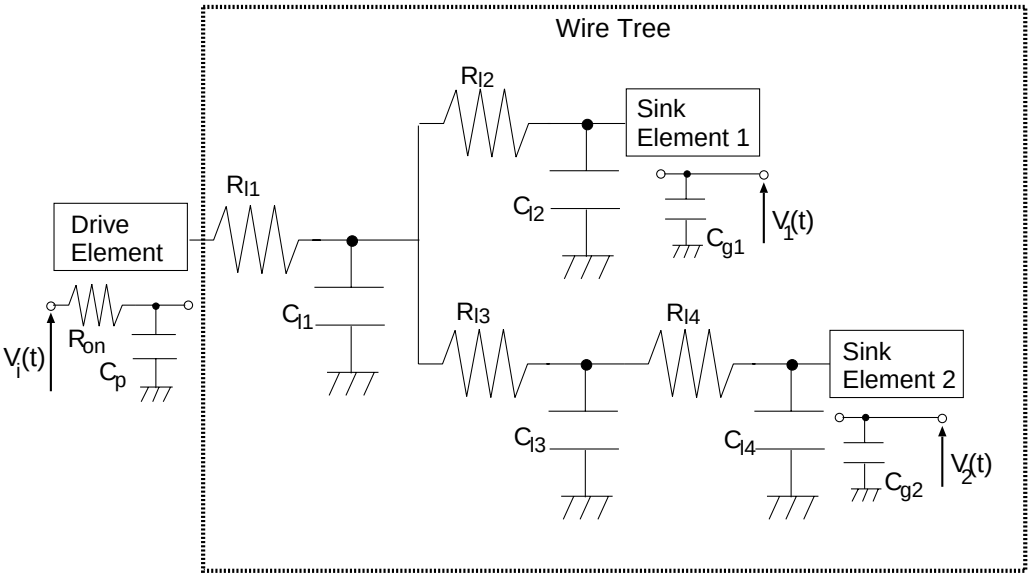

図 6.1: Distributed RC Delay Model 概要

単純に遅延を見積もるのであれば、回路中の抵抗分 R および容量分 C を集中定数として扱う方法がある (Lumped Model)。この方式であれば、配線の長さのみ解ればそこから長さに比例して抵抗分および容量分を求めることができる。図 6.2 にモデル図を示す。この場合の遅延は単に回路方程式を解けばよく、時定数として以下の式で評価することができる。

$$t = RC \tag{6.1}$$

この式では遅延は配線の経路によらず、配線部分は配線の長さのみ解れば良い。この式による評価は正確な遅延評価ではないが、遅延の程度を物理的な要因に基づいて評価する

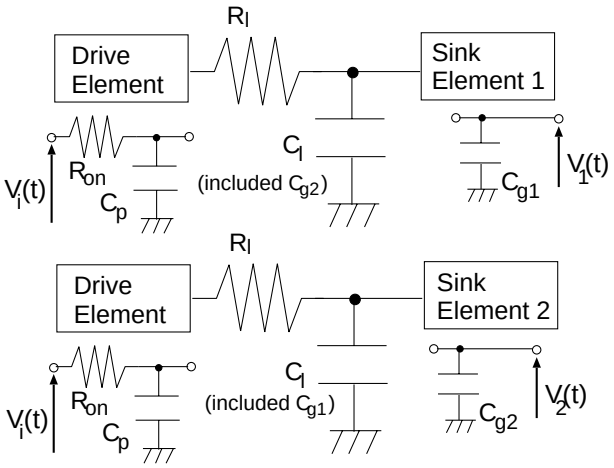

図 6.2: Lumped Model 概要

事ができ、遅延評価の一つの目安となり得る。そこでネットリスト段階で遅延を評価する目安値として**配線量**という指標を用いる。ここで配線量とは、配線の長さに負荷側の容量を配線長に換算したものを足し合わせたものである。 配線量と負荷側の容量から容量分 C を、オン抵抗から抵抗分 R を概算し、その積をとれば遅延の見積もり値となる。

6.1.2 評価方法

遅延評価は、配線木モデルとして Distributed RC Delay Model を使用し、MOSFET のオン抵抗／拡散容量などの値は HSpice から得た。配線の形状は、各素子の位置情報を元に仮想的な配線木を生成して評価した。

仮想配線木

回路評価の際には評価を簡略化するため、実際の配線木に対して仮想的な配線木を生成して評価する。以下駆動素子 i の出力点を駆動点 $D_i(x_{D_i}, y_{D_i})$、負荷素子側の入力点をシンク点 $S_k(x_{S_k}, y_{S_k})\{k = 0, 1, 2...n\}$ とする。概略図を図 6.3(a) に示す。

仮想配線木の生成は、以下の手順で行う。

1. シンク点の X 座標の平均値から固定分 Δ だけ負の方向にずらした場所 $x = C_{D_i}$ に、チャネルを一本生成する ($C_{D_i} = \sum x_k/n - \Delta$)。チャネルの長さ L_{D_i} は各シンク点の y 座標の最大値／最小値の差である ($L_{D_i} = \{max(y_j) - min(y_j)|j = 1, ...n\}$。

2. 駆動点 D_i から生成したチャネルに対して垂線を下ろし、チャネル上に接続点 $N_{D_i}(L_{D_i}, y_d)$ を生成する。

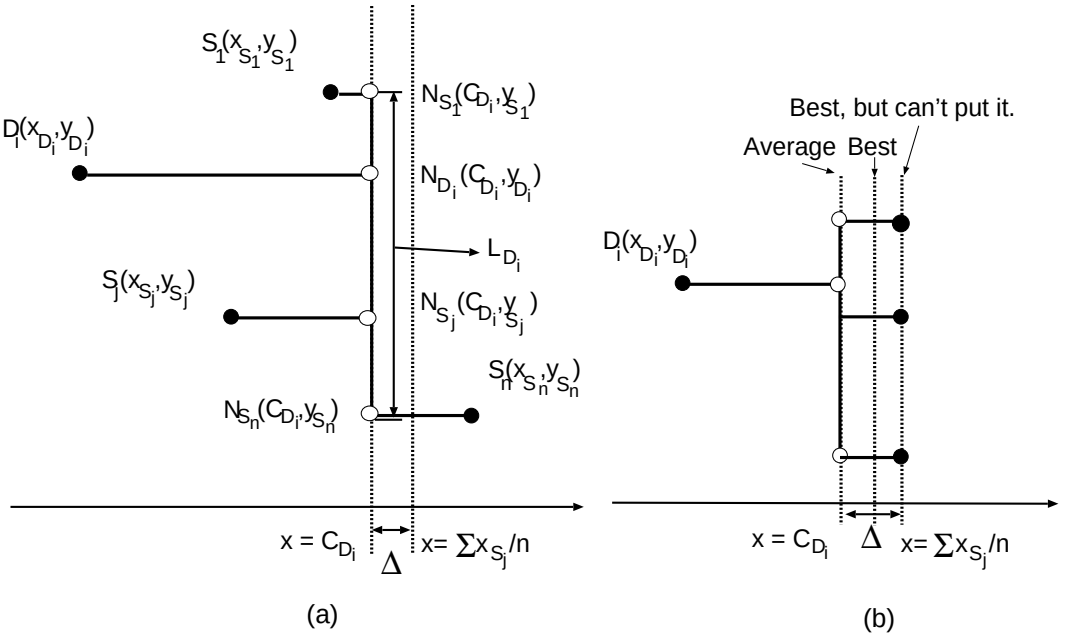

図 6.3: 仮想配線木の定義 (a) およびチャネルルーティングでの例 (b)

3. 各シンク点からチャネルに向かって垂線を下ろし、各シンク毎にチャネル上に接続点 $N_{S_j}\{j = 1...n\}$ を生成する。

4. 各点をつないで配線木を構成する。

このようにして生成された仮想配線木は、段数が均衡化された回路でかつ十分な配線領域が与えられている場合、実配線に非常に近い配線となる。チャネル生成位置を Δ 分ずらすのは、段数が均衡化された回路では各シンク点の X 座標が等しくなり、チャネルがシンク上を通っていくことになるためである。また配線木に分岐点が存在しない場合は、最短距離を通りさえすれば実配線量と仮想配線量は等しくなる。配線木が多くの分岐点を持つ、すなわちたくさんの負荷を持つ場合は、できる限りシンクに近い位置にチャネルを生成することで、配線木の量は小さくなる。しかし全てのチャネルをシンクに一番近い位置にとることはできないため、平均的なチャネル生成位置を Δ によって定めておく。図 6.3(b) にその様子を示す。

一方段数を均衡化しない場合でこの仮想配線木を使って評価した場合は、配線木の総量は実配線木と比べて短くなる傾向にある。よってオリジナルの回路は、この評価方式で評価した場合は実際より速い評価となることが予想される。

6.1.3 パス間遅延を解消するための戦略

γ, δ バッファ挿入に関する段数均衡化戦略

パス間遅延の解消するためには γ, δ バッファを入れる。この時バッファを挿入することによって稼げる遅延は、バッファのオン抵抗／拡散容量および負荷側のゲート容量だけでなく、配線負荷にも依存する。故に挿入する位置が大きな問題となる。図 6.4 に位置を考慮した挿入図を示す。配線木が分割を持たない場合 (図 6.4(a)) では、挿入する位置によって各素子および遅延素子が駆動する容量の大きさが異なる。配線木が分割を持つ場合 (図 6.4(b)) では、挿入する位置によって各負荷側の素子までの遅延時間が異なってくる。 またあるところに挿入しようとしても、すでにその位置が埋まっていたり、プロセスルール上挿入できない位置である可能性もある。ネットリストの段階でこれら配置配線段階での詳細を考慮しながら、厳密に遅延素子の挿入位置を決め配置を行うことは難しい。

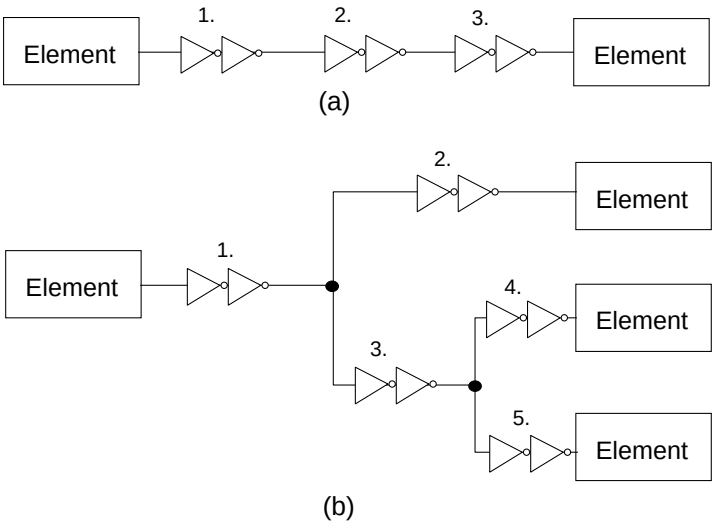

図 6.4: 遅延素子の挿入位置のバリエーション

直感的に考えると、回路中の全ての素子で等しい遅延が発生していた方がパス間での遅延均衡は行いやすい。従って回路中の全ての素子の配線木の総量ができるだけ均衡していた方が良い。全ての配線木の総量をできる限り均衡化させるようにするために、各素子間の配線の真ん中に遅延素子を挿入することを考える。位置情報を持たないネットリストの段階で以上のことを実行するために、入力から出力まで全てのパスで段数が等しくなるように遅延素子を挿入する。すなわち、段数をそろえるようにして γ, δ バッファを挿入することを考える。この作業はネットリストを一つの大きなグラフとして抽象化した上で、以下の 2 ステップで行われる。

1. 入力から全てのパスをたどり、各素子に段数をつける。
2. 再度入力から全てのパスをたどり、段数が均衡するように γ, δ バッファの組を挿入していく。

このとき挿入するバッファはγ, δバッファの組であり、**遅延バッファコンポジット** (Delay Buffer Composit:DBC) と呼ぶ。図 6.5 に回路の段数が均衡化されていく様子を示す。

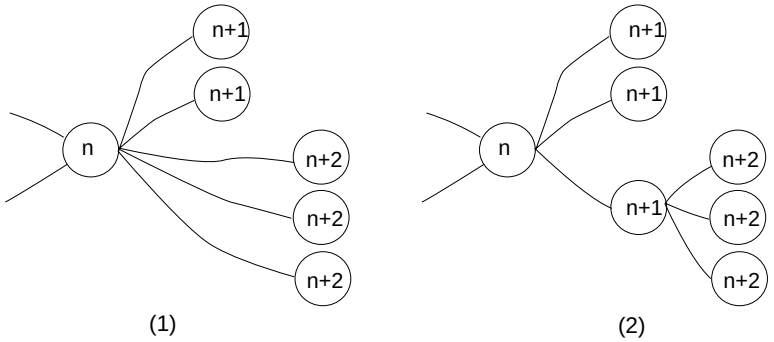

図 6.5: グラフを使った段数均衡化

配線量はまたその配線木が負荷に持つ素子数によっても異なる。負荷をたくさん持つ配線木であれば、配線木の総量は増えざるを得ない。そこで負荷数が多い配線木に DBC を挿入する際には、複数の DBC で負荷を分割する戦略を取る。本来ある素子が出力側に持つ負荷数は、素子のファンアウト能力を考慮して決定される。本提案では、ファンアウト制約を負荷に持つ素子の個数によって単純に制限する[1]。段数づけを行う前に、まずファンアウト制約を取り除くように遅延ブロックを挿入する。このとき負荷分割に DBC を使用することで、論理を正しく保ったまま負荷を分割することができる。但しこの段階では、回路の段数が論理合成直後の段数より増える可能性がある。その後段数づけを行い、段数をそろえるようにして DBC を挿入する。このとき段数を増やさないようにさらに負荷分割する。負荷分割には以下の 3 つの戦略を用意した。

1. 負荷数 n を、分割した各グループがファンアウト制約の個数を越えないように分割する (max-divide)。
2. 負荷数 n を、$\lceil\sqrt{n}\rceil$ 個のグループに分割する (sqrt-ceiling)。
3. 負荷数 n を、$\lfloor\sqrt{n}\rfloor$ 個のグループに分割する (sqrt-floor)。

戦略 1. は段数をつける前に先に適用し、戦略 2.3. は段数をつけたあとに適用する。ファンアウト制約からすれば戦略 1. のみで十分だが、戦略 2.3. によって、段数を増やさないように負荷をさらに分割することで配線木の均衡を図る。戦略 2. は挿入する DBC の数は多くなるものの、一つの DBC が持つ負荷は 3. に対してさらに小さくなる。

段数づけを行うと、各素子は段数を持つ。第 n 段目にある素子の分割戦略は以下の手順で行われる。

[1]本来ファンアウト制約は配線による抵抗／容量分も考慮すべきである

1. n+1 段より大きな段にある素子のうち、一番大きな段にある素子の集合を分割する。分割した各グループはそれぞれ一つの DBC で駆動され、挿入された DBC は負荷側の段に対して一つ手前の段に挿入される。

2. これを繰り返し、n+1 段目まで負荷分割を行う。

3. n+1 段目の素子は負荷分割せず、n 段目の素子自身で駆動する。

この様子を図 6.6 に示す。手順 3. において、n+1 段目の素子も n 段目で負荷分割し、元から n 段目にある駆動素子の段数を n-1 段目以降に下げることも考えられる。しかし回路の段数が元の段数よりも大きくなってしまう恐れがある。段数の増加は遅延差を増やす事につながる他、素子数も増大する。本提案では (ファンアウト制約を解消した後の) 元の回路の段数を増やすことなく、DBC 挿入／負荷分割を行うことにする。負荷均衡しきれない分に関しては駆動側の駆動素子サイズを変更することで対処する (可変駆動 β バッファ戦略:後述)。

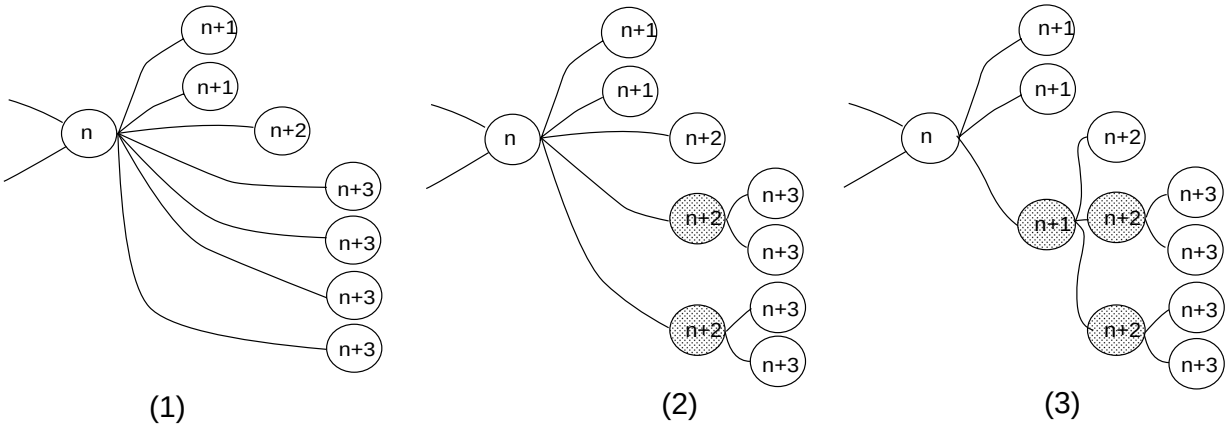

図 6.6: 負荷分割を伴う段数均衡化 (網かけは遅延ブロックを示す)

配線木均衡化のための配置戦略

段数を均衡化させた段階で、回路中の任意の素子において、その素子に至るまでの全てパスが通過する素子数は互いに等しくなる。この状態で仮想グリッドに配置をすることによって、ネットリストに位置情報を付加する。配線木を均衡化させるという観点と配置のしやすさという観点から、同じ段数の素子は全て同じ垂直座標上に置くこととする。このとき段数が均衡化されているので、ある n 段目の素子へのパスは全て n-1 段目からの入力を受け取ることになる。故に第 n 段の配置を行うときは、n-1 段目との位置関係さえ考えれば良い。段数を均衡化し、同一段の素子を同じ垂直座標上に配置するという手法は、遅延均衡の観点から見たときに以下の点で有利になる。

1. 通過素子数の均衡化
   回路中の任意の素子に関して、その素子までの全てのパスで通過する素子数が互いに等しくなる。パス間の遅延をとりやすい。

2. 配置の容易さ
   n 段目の配置の際は、n-1 段目との位置関係のみ把握すれば良い。配置問題を段数分に分割可能。

3. 配線形状の推定の容易さ
   段間に十分な配線領域があればチャネルルーティングを行うことができ、実際に配置をしなくとも仮想的な配置から、十分配線経路を推定することが可能である。配線経路の推定から得られる配線量を**見込み配線量**と定義する。配線形状の推定および見込み配線量の求め方については、仮想配線木の項で後述する。

配置をする際には、配線領域および素子領域が無駄に拡大することを防ぐ必要がある。ウェーブパイプライン設計では最大遅延がいくら大きくなっても遅延差が小さければ良い。このため配線木の総量を単に均衡化させるのであれば、素子を遠くの位置に置いて配線木の総量を増やすことも考えられる。しかしある制限がないと無駄に面積が広がってしまうという可能性もある。いくら配線木が均衡化しても、総量が大きくなり過ぎれば一つのパス内で発生する遅延差は大きくならざるを得ない。

以上を踏まえ、本提案では n 段目の配置を以下の手順で行う。

1. n-1 段目との強連結アルゴリズムによって素子を配置する位置を仮決めする。
   配置しようとする素子と前段でつながっている複数の素子との力関係を定義し、力学的手法により配置する強連結アルゴリズムによって、あらかじめ素子を配置する位置を固定する。まず n-1 段目の各素子を、負荷数の多い素子から順に選択し、その素子と負荷側でつながる n 段目の素子を選択、配置を行う。この段階では配線量および素子領域の総量に関する局所解が得られる。

2. n 段目内でのペア交換法によって配線木を均衡化させる。
   強連結アルゴリズムでは配線量極小の局所解が得られるのであって、極端に短い配線が生じる。この制限を緩和させるため、素子間でペア交換を行い、強連結アルゴリズムによる制限を和らげる (図 6.7(a))。この際、単にペア交換を行うだけでは入れ替えるパターンが限られるので、空いている領域にダミースロットを入れてペア交換を行うことで、見た目上は素子を単純に移動させることも行う。この様子を図 6.7(b) に示す。この操作は配線木量がある程度均衡化するまで繰り返し行われる。配線木量のちらばりに関しては標準偏差を用いて判定を行う。

今、段数が x 軸として定義される仮想グリッドに配置を試みる。ある素子 $E_i(x_{E_i}, y_{E_i})$ は入力側で素子 $e_j(x_{e_j}, y_{e_j})\{j = 1...n\}$ とつながっているものとする。また素子 E_i に割り振

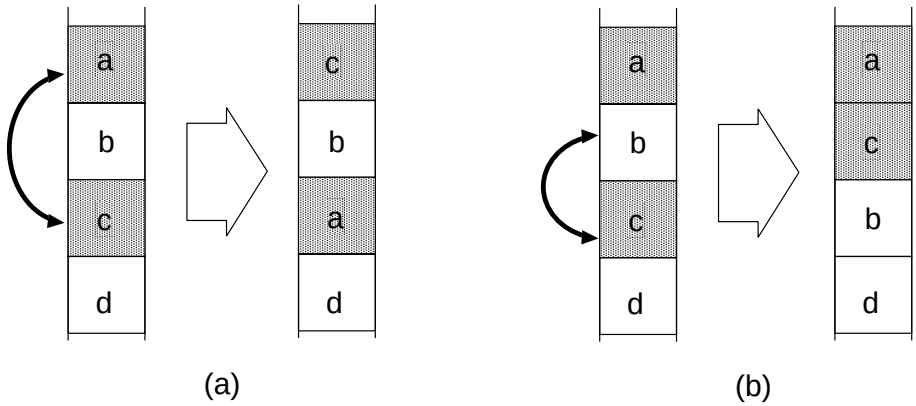

図 6.7: ペア交換法およびダミースロットを使ったペア交換による素子の移動 (網掛けは使用スロットを示す)

られた段数を L_{E_i} と表す。この時素子 e_j が負荷側の素子 E_i に対して及ぼす力 $\mathbf{F}_{e_j}(e_j, E_i)$ は $x = L_{E_i}$ 軸上一次元であり、

$$F(e_j, E_i) = k(y_{e_j} - y_{E_i}) \tag{6.2}$$

と定義する。ここで k は比例定数である。力の定義を図 6.8 に示す。連結している素子 e_j が E_i に対して及ぼす力 $\mathbf{F}_{e_j}$ の和 $\sum \mathbf{F}_{e_j}$ が 0 になる地点を選択し、素子 E_i を置こうとする。これを候補点とする。式 (6.2) および $\sum \mathbf{F}_{e_j} = 0$ より素子 E_i の候補点 (x_{E_i}, y_{E_i}) は次式で定義される。

$$(x_{E_i} \quad , \quad y_{E_i}) = \left(L_{E_i} \quad , \quad \frac{1}{n} \sum_{i=1}^{n} y_i \right) \tag{6.3}$$

求めた候補点にすでに素子がおいてある場合は、$x = L_{E_i}$ 上を正負の方向に領域を探索し、候補点から一番近い空きスロットを検索しそこに置く。強連結アルゴリズムによってあらかじめ素子を配置する位置を固定するので、配置領域が広がっていくことはない。以上のアルゴリズムは、強連結アルゴリズムでできるだけ配線領域／素子領域を抑えながら、ダミースロットを使ったペア交換法で配線木量を均衡化させる事を狙ったものである。しかしながら強連結アルゴリズムによる位置の固定は、空きスロットを生じさせる事になる。先に定義した力関係に束縛されるため、対象とする段に対して前段の素子数が非常に多い場合は、対象とする段に多くの空きスロットができるという欠点がある。

以上の配置戦略を各段に対して行うことで仮想グリッドに配置をしていく。配線木量は見込み配線量を指標に用いる。このとき負荷側の容量も配線長に変換することで、負荷側の容量も考慮した見込み配線量に換算する。以上の戦略が終わった段階で、各素子の (負荷側の容量も含めた) 配線木が均衡化した状態となる。

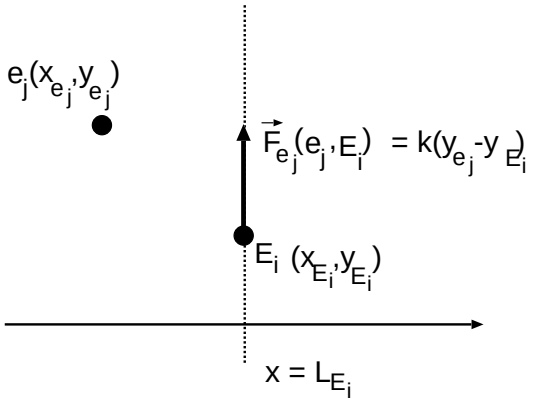

図 6.8: 力の定義

6.1.4 一つのパス内での遅延差を均衡化させるための戦略

一つのパス間で遅延差を均衡化させるために、論理素子のみを考慮したアルゴリズムでは α, β バッファを定義した。配置配線を考慮したアルゴリズムでは、テクノロジマッピングの際に α, β バッファ挿入を行う。

ブロッキング

α, β バッファおよび α, β バッファが対象とする素子は、配線の影響が小さくなるように互いに近い場所に置くのが良い。そこである素子をテクノロジマッピングする際に、α, β バッファを挿入したセルを動的に生成することで、一つのパス内での遅延差を抑える。同一セル内に α, β バッファおよび対象とする素子を置くことで、配線により多入力素子で生じる遅延差の影響を最小限に抑える。 この手法を**ブロッキング**と定義する。概念図を図 6.9 に示す。同一セル内に置いてしまうことで、α, β バッファの配置問題を簡単にする狙いもある。

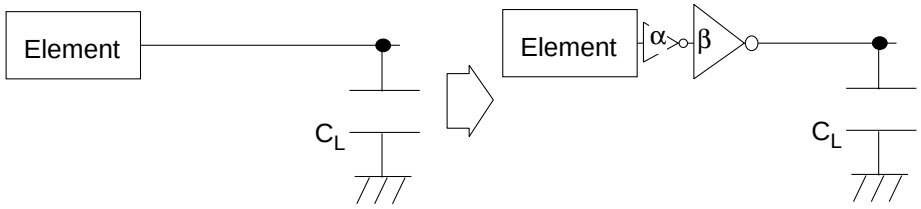

図 6.9: α, β バッファによるブロッキング

ネットリストの段階では各素子は単一の素子として扱われていたが、実際の配置では α, β バッファからなる**ブロック**と呼ばれるセルに置き換えられる。各ブロックで α バッファにはリファレンスとなる一番小さなサイズのインバータを挿入し、β バッファには大

きなインバータを使用する。

βバッファはαバッファの負荷を減らす役割の他に、配線木を駆動する駆動素子としての役割がある。単に多入力素子を大きくするのではなく別に駆動インバータを用意することで、配線木で発生する最大遅延を縮めながら最小遅延の低下を最小限に抑えることができる。これはインバータの最大／最小オン抵抗の差が小さいことを利用している。注意しなければならないことは、βバッファのサイズによってαバッファの反応速度が異なるため、ブロック内部で発生する遅延が異なることである。故に用意するβバッファのサイズに応じて、各素子ごとにそれぞれブロック内で発生する遅延を計測しておく必要がある。例えばinv,nand2,nand3,nor2,nor3と5つ素子が使用可能で、βバッファとして4つのサイズを使用可能とするならば、合計で20個のブロックのライブラリを用意しておく必要がある。セル設計に関する詳細を付録Aに載せた。また付録Aの表A.5に、今回使用した各ブロックのパラメータを示す。ブロック内部で発生する遅延は配線木で発生する遅延と比較して小さく、ピン毎の拡散容量の違いも配線部分の容量と比較すると小さい。故に前章で定義した再接続戦略およびリサイジング戦略は、配置配線を考慮した戦略では適用しないこととした。

固定／可変駆動βバッファ

単一パス内の遅延差、特に配線木で発生する遅延差を軽減するというのであれば、大きなβバッファを一つ用意すれば良い。しかしいくら配線木を均衡化させるような配置戦略をとっているとはいえ、どうしても配線木の総量には差が出てくる。そこで駆動側のβバッファを配線量や負荷側の容量に合わせて可変にすることで、配線木で発生する遅延をさらにそろえることを目指す。配線量や負荷側の容量に合わせてβバッファを選択するという事は、すなわち式(6.1)の値を均衡化することに他ならない。配線量や負荷側の容量に関係なく固定サイズのβバッファを使用する戦略を**固定駆動βバッファ戦略**と呼ぶ。一方、配線量や負荷側の容量に合わせて可変のβバッファを用いる戦略を**可変駆動βバッファ戦略**と呼ぶ。今回はβバッファはサイズ別に4つ用意したが、式(6.1)を一定に揃えることを考えればβバッファの選択肢はできる限り細かくかつ小さいサイズから大きいサイズまで広く選択できる方が良い。

6.1.5 遅延均衡化戦略の手順

以上の戦略を組み合わせることで、位置情報を含んだネットリストを生成し、テクノロジマッピングをして実際の配置配線を行う。以下手順と計算量を示す。ここでn:入力素子数、n':DBC挿入後のブロック(素子)数、m:回路の段数、k_m:DBC挿入後の第m段でのブロック(素子数)を示す$(m, k, n < n')$。実装したプログラムでは各素子に関する位置情報などは、段数ごとのリストとして格納されていることを考慮している。

順番	説明	計算量
1.	オリジナル回路の段数づけ	$O(n)$
2.	DBC 挿入、負荷分割	$O(n)$
3.	強連結アルゴリズム	$O(k_{m-1}k_m)$
4.	ペア交換法	$O(k_{m-1}k_m)$
5.	3-4 を段数分繰り返し	
6.	ライブラリ生成	$O(n')$

故に全体の計算量 $T(n, n', m)$ は

$$T(n, n', m) = O(n) + \sum_{i=1}^{m} \{O(k_i k_{i-1}) + O(k_i k_{i-1})\} + O(n') < O(n'^2) \tag{6.4}$$

であり、$O(n'^2)$ より小さくなる。これは配置を段数ごとに分割した効果による。しかし実際にはある一定のばらつきに収まるまで繰り返しペア交換を行うため、ペア交換法による改善がアルゴリズムの時間の多くを占める。

6.2 シュミレーション実験による各戦略の検証

ここで対象とする回路は、 111 個の論理素子からなる 4 ビット ALU である。回路中、排他的論理和素子は 2 入力 NAND を 4 つ用いて構成している。このとき、入力から出力までの各パスの中で論理段数が最大となる段数は 14 段である。信号線など、パスによっては回路中を大きくスルーしていくものもあり、各パスは不均一な段数から成る。 これらのことから、小規模な回路ではあるがパイプラインの一ステージとして、十分目安になるものと考えている。付録 B. に論理回路図を載せる。また各素子のパラメータを付録 A.2.3 の表 A.3 に、各ブロックのパラメータは付録 A.2.3 の表 A.4 に載せた。

MOSFET および各配線層に関する各パラメータは、前章までで検証した値に準じたものを用いる。故に数十グリッドの配線で、インバータのゲート容量が配線容量に等しくなるという環境でシュミレーションを行った。また、同じく前節で検証した通り、ウェーブ版の 4 ビット ALU は配線の両側をグラウンド線で電気的に遮蔽することとする。

6.2.1 シュミレーション前検証

実際に検証を進める前に、仮想配線木を使用したことによる影響および配線負荷と素子負荷の影響の度合いに関して考察を行った。

仮想配線木による影響

回路段数の均衡化を行った際、実際に手作業で配線を行った回路を用意し、仮想配線木を自動的に生成した場合の配線木とを比較した。図 6.10 は実配線木長に対する、実配線木

長と仮想配線木長の差の割合を示しており、割合が負である素子では実配線木よりも仮想配線木の方が長いことを示している。図中横軸は回路中の各素子を番号づけしたもので、特に意味はない。負荷が一つで配線木の途中で分割がない場合、実配線木も仮想配線木も総量が等しくなっている。また配線木が途中で枝分かれする場合は、チャネルをどこにとるかによって実配線と仮想配線で差が出る。仮想配線木で生成したチャネルよりも負荷側に近いところのチャネルを取ることができた場合、実際の配線量は仮想配線量より小さくなる。仮想チャネルよりも駆動側に近いところでしかチャネルを取ることができなかった場合、実際の配線量は仮想配線量より多くなる。

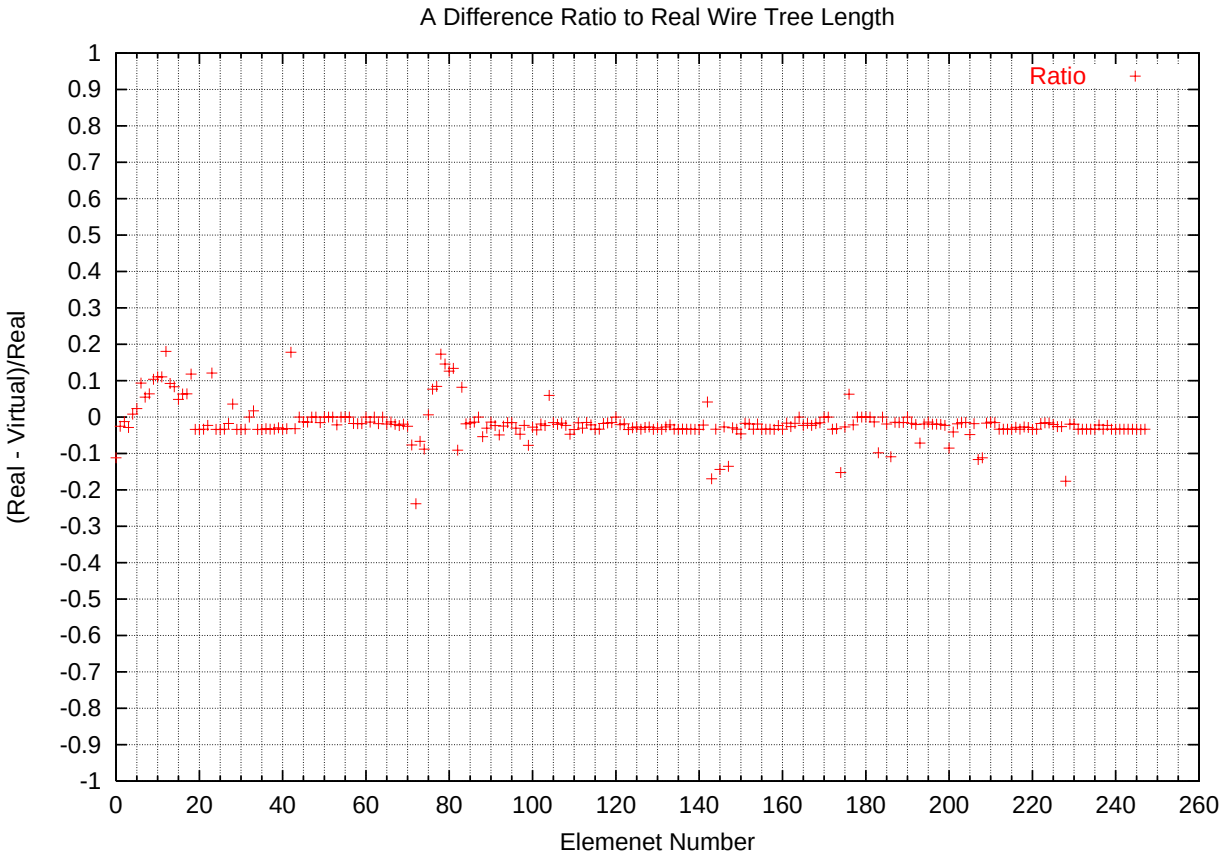

図 6.10: 仮想配線木長に対する実配線木長と仮想配線木の差の割合

配線負荷と素子負荷による影響

配線負荷と素子負荷が回路遅延に与える影響を知るため、ある回路均衡化作業で生成された回路から、各素子ごとに仮想配線木と負荷側のゲート容量を算出した。各配線木の総量に対するゲート素子の総量の比を図 6.11 に示す。全ての素子で配線容量の方がゲート

容量を 2-10 倍程度上回っている。今回負荷側に使用する素子は可能な限り最小の寸法を使用しているため、配線容量の依存度が非常に大きくなっていることが確認できる。

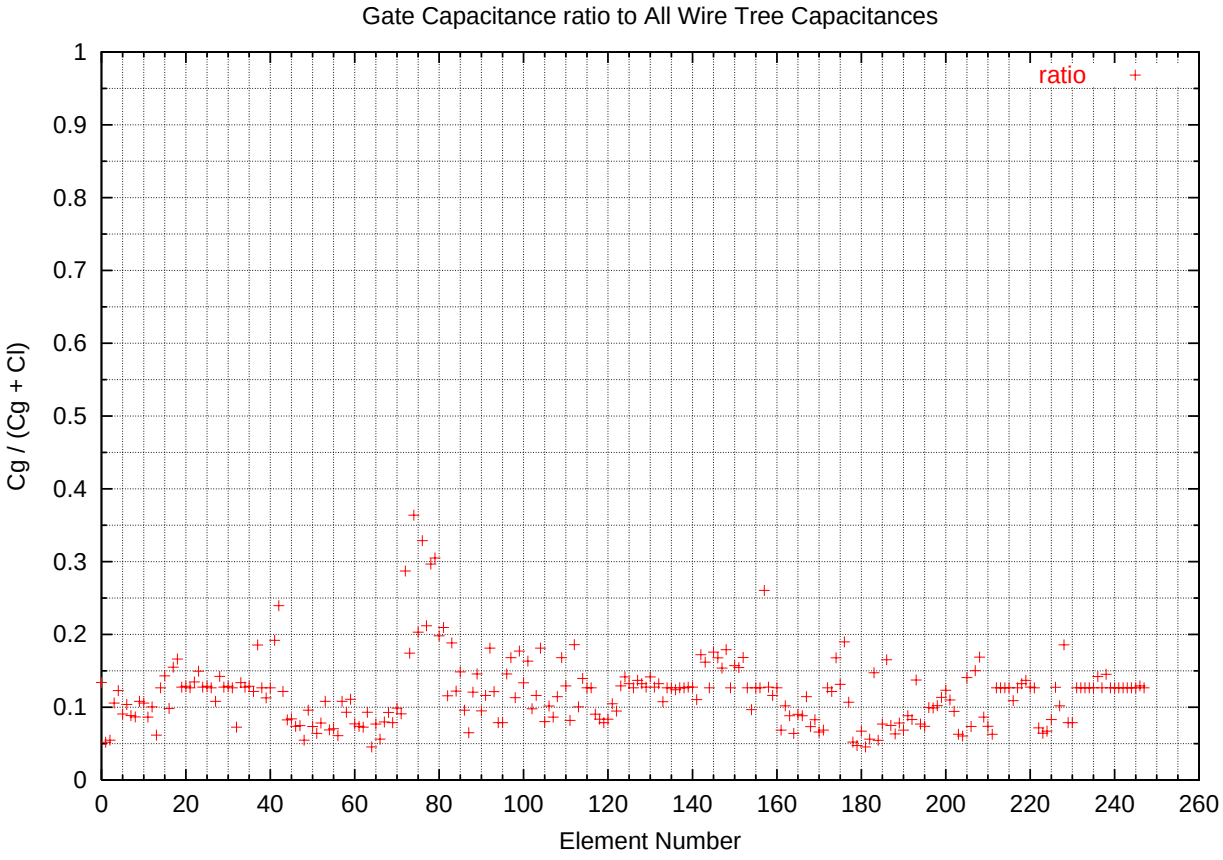

図 6.11: 配線木の全容量に対する負荷側のゲート容量の総計の比

遅延要因分析

回路中の各素子において遅延差が発生する要因は、以下の 3 つに分けることができる。

要因 1. パス間での遅延均衡を取り切れなかったことによって生じるパス間遅延差

要因 2. ブロック内部で発生するブロック内遅延 (差)

要因 3. 負荷側の配線木／負荷素子で発生する配線木遅延 (差)

このうち、パス間での遅延差の原因となるのは 1. であり、一つのパス内で発生する遅延差の要因となるのは 2.3. である。また遅延差がインバータの最大／最小オン抵抗の差に依存するものは 3. である。今回回路中の全ての素子ごとに発生している遅延差を各要因別

に分け、回路全体で要因別の総計を出すことで発生した遅延の要因についても考察する。要因 1. に関しては発生遅延差分を、要因 2.3. に関しては発生遅延差／最大遅延に関する数値を抽出した。

6.2.2 各種戦略定義と結果

以上の環境の上で、各戦略を設定し評価を行った。以下各戦略においてそれぞれ観点別に結果を示す。

段数均衡化による遅延均衡化

まず入力回路に対して負荷制限を取り除いただけの回路 (戦略 0.) と、DBC によって段数を均衡化した回路 (戦略 1-1) およびさらに段数を増やさないで負荷分割をした場合の回路 (戦略 1-2) に関する結果を表 6.1 にまとめた。表中圧縮比とは、最大遅延に対する遅延差の割合を示す。また DBC 数とはファンアウト制約を取り除くために挿入した分も含める。故に戦略 0. でも多少 DBC が挿入されることになる。戦略 1-1,1-2 の配置アルゴリズムはダミースロットを使ったペア交換法を用いている。なお、特別な設定がない限り、ファンアウト制約を通常 5 に定めた。また表中総配線量とは仮想配線木の総量を指し、グランド線は含めていない。

単に段数均衡化を行うことによって最小遅延を 235ps 増やすことができた。これは段数が少ないパスでの高速な反応を、インバータによって抑えることができているためである。戦略 1-1. および 1-2. から、さらに段数を増やさないように負荷分割することによって、 最大遅延を 125ps 抑えることに成功した。戦略 1-2. での最大遅延は、戦略 0. での最大遅延にほぼ等しい。単に段数を均衡化させるだけでは、挿入したインバータが (ファンアウト制約を越えな い範囲で) 多くの負荷を持つ場合に、最大遅延に関して新しくクリティカルパスとなる可能性がある。戦略 1-1.,1-2. から、負荷分割を適度に行うことで、他のパスにはなるべく影響を与えないようにしながら、高速なパスにバッファを挿入し最小遅延を大きくすることが可能であることを示している。但し配線量はその分増加する。また挿入したブロックは γ バッファを δ バッファでブロックしたもので、ブロック内で発生する遅延差は全体で発生する遅延差に対して小さい。

ブロッキングによる遅延均衡化

戦略 1-2. により単に段数を均衡化した回路に対し、固定駆動 β バッファ戦略で全ての素子をブロックした回路を比較した。α, β バッファによるブロッキングの効果、および β バッファで配線木を駆動することによる遅延差の短縮の効果を見る。表 6.2 に結果を示す。

単に段数を均衡化したときに比べて、遅延差を 54.9%と半分以上にまで減らすことができた。圧縮比で比べると最大遅延に対して遅延差の割合が 13.1%に低下していることか

表 6.1: 段数均衡化戦略と最終結果

戦略概要

戦略区分	戦略概要
0.	入力からファンアウト制約を除去したのみ
1-1.	段数均衡化を行う。但しファンアウト制約が取れている場合はそれ以上負荷分割を行わない。
1-2.	段数均衡化を行う。段数を増やさないようにさらに細かく分割する (sqrt-ceiling)

最終結果

戦略	最大遅延 [ps]	最小遅延 [ps]	遅延差 [ps]	圧縮比	挿入 DBC 数	挿入 インバータ数	総配線長 [grid]
0.	1071	167	904	0.844	10	20	68581
1-1.	1186	402	784	0.661	125	250	91622
1-2.	1061	405	656	0.618	152	304	99177

要因別

戦略区分	要因 1.	要因 2.			要因 3.		
戦略	遅延差 [ps]	遅延差 [ps]	最大 [ps]	圧縮比	遅延差 [ps]	最大 [ps]	圧縮比
0.	2276	-	-	-	5532	10622	0.521
1-1.	1293	75	656	0.114	6178	14184	0.435
1-2.	1052	91	798	0.114	6500	15282	0.425

ら、駆動 β バッファによって最小遅延の低下をできるだけ抑えながら最大遅延を大きく低下することができていると判断できる。実際要因別の遅延差から判断すると、各素子の出力側で生じている配線木の遅延圧縮比を 31.3%に減らすことができているのが解る。これはほぼインバータの最大オン抵抗に対する、最大／最小オン抵抗の差の割合に等しい。一方で素子数は 2.73 倍になる。しかし通常の回路設計の場合でも論理素子をそのまま置くことはなく、駆動用に大きな素子を使ったり、インバータによって負荷を分散させることは行われていることを考えれば、直接この素子数変化を大きいと見なすことはできないであろう。

可変駆動 β バッファおよび配置戦略による遅延均衡化

次に、固定駆動 β バッファ戦略 (戦略 2.) に対して、可変駆動 β バッファ戦略を行い比較する。配置法による効果も見るため、複数の戦略を試行した (戦略 3-1.3-2.3-3.)。表 6.3 に戦略概要および各結果を示す。

表 6.2: 固定駆動 β バッファ戦略と最終結果

戦略概要

戦略区分	戦略概要
2.	固定駆動 β バッファで DBC を含む全ての素子をブロックする。

最終結果

戦略	最大遅延 [ps]	最小遅延 [ps]	遅延差 [ps]	圧縮比	挿入 DBC 数	挿入 インバータ数	総配線長 [grid]
1-2.	1061	405	656	0.618	152	304	99177
2.	738	378	360	0.487	152	830	105620

要因別

戦略区分	要因 1.	要因 2.			要因 3.		
戦略	遅延差 [ps]	遅延差 [ps]	最大 [ps]	圧縮比	遅延差 [ps]	最大 [ps]	圧縮比
1-2.	1052	91	798	0.114	6500	15282	0.425
2.	441	1567	5121	0.306	1771	5654	0.313

戦略 2. と 3-1. から、遅延差をさらに 23ps 縮めることができ、ダミースロットを使った戦略 3-2. と比較すると 126ps 縮めることに成功している。DBC を挿入することにより回路中の多くの素子は負荷が 1-3 個程度となり、段数をそろえるようにして挿入した DBC では負荷が 1 つというのが大半である。故に配線木をできるだけ均衡化させる効果は大きい。さらに空きスロットを使ったダミースロット使用のペア交換法によって、圧縮比換算で 40%程度にまで遅延を圧縮することができかつ配線量を抑えることに成功し、大きな効果を得られた。また今回はファンアウト制約を小さくすることによる効果は見られなかった。ファンアウト制約をやみくもに小さく取ると、段数が増大する可能性もある。いくら配線木の容量をそろえようとしてファンアウト制約を小さくしたとしても、段数が増えると一つのパス内で発生する遅延差の累計が増える。

デコンポジション／および負荷分割戦略変更による遅延均衡化

最後にその他の戦略として、あらかじめ 3 入力 NOR を 2 入力 NOR に分解した回路 (戦略 4.)、バッファを節約するように分割戦略を取ったもの (戦略 5., 戦略 6.) を行った。表 6.4 に概要および各結果を示す。

戦略 3-2. および 5. を比較して、負荷分割戦略は今回は挿入数をやや減らす方向に戦略を取る戦略 5. の方がわずかに良く、配線量も少なくて済む。しかしどちらがよいという評価までには至らなかった。またデコンポジションを行う戦略 4. は、NOR 素子によるブロック内遅延差を抑えることで、圧縮比は良く保っているものの段数が増えたことが大き

表 6.3: 可変駆動 β バッファ各戦略と最終結果

戦略概要

戦略区分	戦略概要
3-1.	可変駆動 β バッファで DBC を含む全ての素子をブロックする。配置ではペア交換法にてダミースロットを使用しない。
3-2.	戦略 3-1. に対して、ダミースロットを使用したペア交換法を用いる。
3-3.	戦略 3-2. に対して、ファンアウト制約を 4 にとる。

最終結果

戦略	最大遅延 [ps]	最小遅延 [ps]	遅延差 [ps]	圧縮比	挿入 DBC 数	挿入 インバータ数	総配線長 [grid]
2.	738	378	360	0.487	152	830	105620
3-1.	713	376	337	0.472	152	830	105382
3-2.	603	359	244	0.405	152	830	88193
3-3.	608	360	248	0.408	158	838	89848

要因別

戦略区分	要因 1.	要因 2.			要因 3.		
戦略	遅延差 [ps]	遅延差 [ps]	最大 [ps]	圧縮比	遅延差 [ps]	最大 [ps]	圧縮比
2.	441	1567	5121	0.306	1771	5654	0.313
3-1.	348	1596	5230	0.306	1691	5394	0.313
3-2.	313	1595	5220	0.306	1457	4665	0.312
3-3.	304	1622	5326	0.304	1491	4776	0.312

く、性能向上にまでは至らなかった。また負荷分割を行わず単に段数をそろえる戦略である戦略 6. は、性能の面で戦略 3-2 に対し明らかに悪い。要因を見ると戦略 6. は素子数が少ないにも関わらず、回路全体で発生しているパス間の遅延差の累計が 95ps ほど多い。戦略 1-1,1-2 で検討した、最大遅延を増やさず速いパスを遅くする負荷分割戦略の効果に加えて、パス間で遅延均衡をとる際にも負荷分割戦略が有効であることを示している。

6.3 シュミレーション結果

以上 4 ビット ALU での評価を通じて、配置配線問題を含めた遅延均衡化手法に関しての種々の手法を提案し検討した。本節はその考察である。

表 6.4: その他各種戦略と最終結果

戦略概要

戦略区分	戦略概要
4.	戦略 3-2 に対して、NOR3 → NAND3 のデコンポジションを回路全体に施したもの。入力段数が 16 段になる。
5.	戦略 3-2. に対して、DBC 挿入数が小さくなるように負荷分割戦略を取ったもの (sqrt-floor)。
6.	戦略 3-2. に対して、DBC を挿入する際にファンアウト制約以下には負荷分割を行わないもの (戦略 1-1. に対して可変駆動戦略を取る)。

最終結果

戦略	最大遅延 [ps]	最小遅延 [ps]	遅延差 [ps]	圧縮比	挿入 DBC 数	挿入 インバータ数	総配線長 [grid]
3-2.	603	359	244	0.405	152	830	88193
4.	687	409	278	0.405	190	936	111465
5.	596	355	241	0.404	131	746	82559
6.	636	360	276	0.434	125	722	81728

要因別

戦略区分	要因 1.	要因 2.			要因 3.		
戦略	遅延差 [ps]	遅延差 [ps]	最大 [ps]	圧縮比	遅延差 [ps]	最大 [ps]	圧縮比
3-2.	313	1595	5220	0.306	1457	4665	0.312
4.	436	1776	6292	0.282	1824	5838	0.312
5.	305	1494	4826	0.310	1352	4327	0.312
6.	409	1465	4712	0.311	1331	4258	0.312

6.3.1 手法に関するまとめ

構築した手法を従来から (世間一般で行われている) の回路設計の立場から比較すると、配線／素子負荷を考慮して駆動側の素子の大きさを求めることで、回路を高速にする手法を取り入れているという点では従来手法と同一である。一方で大きくする素子を論理回路での素子ではなく、新たに駆動用のインバータを用いることで、最小遅延の低下をできるだけ抑えながら回路を高速に動作させるという点で異なる。また α バッファを導入することで駆動用のインバータによる属性の反転を防ぎながら、論理素子で発生する遅延差をできる限り抑えようとする。配置配線では、配線量が最小になるように配置するのではなく、負荷側の素子のゲート容量も含めて、各素子での出力側の配線量ができる限り均衡化するように仮想配置することを提案した。

6.3.2 パスの長さの違いにより生じる遅延差

前章でCMOS上でのウェーブパイプラインのパラメータとして、論理段数および配線長の違いによる遅延均衡の程度に関するパラメータ$\mathcal{A}$、および動作条件の違いにより生じる遅延均衡の程度に関するパラメータ$\mathcal{B}$を定義した。二つのパラメータから、まず論理段数および配線長の違いによる遅延均衡の改善度合いを考察する。パラメータ$\mathcal{B}$は、本提案での理想的な状態ではインバータの最大オン抵抗に対する最小オン抵抗の比$\mathcal{B}'$であるとした。付録A.2.3の表A.3より、仮定しているパラメータの下では$R_{max} = 4.28[k\Omega], R_{min} = 2.95[k\Omega]$としているので、$\mathcal{B}$の値を以下のようにして求めることができる。

$$\mathcal{B} \approx \mathcal{B}' = \frac{4.28}{2.95} = 1.45 \tag{6.5}$$

圧縮比$\mathcal{C}$の定義より、パスの長さの違いにより生じる遅延差の程度を示すパラメータ$\mathcal{A}$は以下の式で求めることができる。

$$\mathcal{A} = \frac{1}{\mathcal{B}(1-\mathcal{C})} \tag{6.6}$$

表6.5に各戦略での結果のまとめおよび$\mathcal{A}$の値を示す。最終的には$\mathcal{A} = 1.16$まで改善できたことがわかる。単に段数を均衡化するだけでも、パラメータ$\mathcal{A}$上で2.03にまで改善可能であるが、各種パス遅延均衡化をとることによって、パラメータ$\mathcal{A}$上で比較してさらに倍程度パスの長さの違いにより生じる遅延差を改善することができたことを示している。

表 6.5: 各戦略毎の結果の再掲およびパラメータ$\mathcal{A}$

戦略区分	最大遅延 [ps]	最小遅延 [ps]	遅延差 [ps]	圧縮比	$\mathcal{B} = 1.45$ の時の$\mathcal{A}$
0.	1071	167	904	0.844	4.42
1-1.	1186	402	784	0.661	2.03
1-2.	1052	405	656	0.618	1.81
2.	738	378	360	0.487	1.34
3-1.	713	376	337	0.472	1.31
3-2.	603	359	244	0.405	1.16
3-3.	608	360	248	0.408	1.16
4.	687	409	278	0.405	1.16
5.	596	355	241	0.404	1.16
6.	636	360	276	0.434	1.22

6.3.3 最大遅延動作回路との比較

最後に最大遅延で動作する回路との比較を検討する。先に述べた戦略 0. は段数均衡化を行わない回路であり、最大遅延で動作する。しかし各素子の駆動能力を上げることでより高速に動作することができ、ウェーブ動作の回路と単純に性能比較することはできない。そこで戦略 0. で生成された回路の各素子をブロック化することで駆動能力を上げた回路を、ウェーブ動作の回路との比較に用いた。ウェーブ動作の回路として、戦略 3-2. をベースに駆動用 β バッファのサイズ割当てなどを変更した回路を用いた。表 6.6 に最終的な結果を、表 6.7 にパラメータ $\mathcal{A}$ の値を示す。ここでの比較において、表中挿入 DBC 数とはファンアウト制約を取るために挿入した分を含まない数を指す。故に最大遅延動作版では 0 である。

表 6.6: 4 ビット ALU での遅延均衡化結果

戦略	最大遅延 [ps]	最小遅延 [ps]	遅延差 [ps]	圧縮比	挿入 DBC 数	挿入インバータ数	総配線長 [grid]
最大遅延動作版	524	157	367	0.700	0	262	63187
ウェーブ動作版	624	395	229	0.367	152	830	117626

表 6.7: 4 ビット ALU での遅延均衡化に関する各値

戦略区分	最大遅延 [ps]	最小遅延 [ps]	遅延差 [ps]	圧縮比	$\mathcal{B} = 1.45$ の時の $\mathcal{A}$
最大遅延動作版	524	157	367	0.700	2.30
ウェーブ動作版	624	395	229	0.367	1.09

性能向上比で見ると、動作クロック比で 2.29 倍高速に動作させることが可能になった。一方総素子数は最大遅延動作版で 373 個であるのに対し、ウェーブ動作版では 941 個であり、2.55 倍増加した。また配線領域に関して、実配線領域のみを比較すると 1.86 倍に増加した。実際にはウェーブ版では配線の両隣はグランド線であるので、実配線のさらに 2 倍の領域を必要とする。故に配線領域は実質 3.72 倍になったと言える。

6.4 結論

本章では配置配線問題を考慮した遅延均衡化戦略を定義し、4 ビット ALU に対して適用することでその効果を検証した。まず配置配線問題を考慮した場合に、ネットリストの

段階で遅延を評価できるような配置配線を提案した。それを元にパス間での遅延差を解消する手法と、一つのパス間で生じる遅延差を抑える手法を複数提示した。実際に遅延評価を行うにあたり、評価を簡単化するため仮想配線を定義し、その影響を考察した。また素子負荷に対して配線負荷の程度を調べ、その影響を考察した。さまざまな戦略を用意して 4 ビット ALU に対して適用することで、その効果を検討した。

最大遅延動作版との比較において、第 5 章で導入した $\mathcal{A}, \mathcal{B}$ を使用しての考察から、実際には $\mathcal{A}$ は 1.00-1.09 の間にあると推測できる。図 4.1 に示したグラフにより、これ以上 $\mathcal{A}$ を改善しても性能向上は望めない。このことから本章で構築した遅延均衡化戦略によって、遅延段数および配線長の違いにより生じる遅延差は極限まで取り除くことができたと考える。この状態からはむしろ $\mathcal{B}$ を小さくする、すなわち動作条件の違いにより生じる遅延差を抑えることが重要となる。

以上から、再論理合成／再配置配線をすることなくかつ配置配線も考慮した遅延均衡化を達成できたと考える。本手法により一つのパス内での遅延均衡、および複数のパス間での遅延均衡を同時に改善する事が可能である。また配線による影響が大きいディープサブミクロンデバイスの下でも、十分に遅延均衡化を達成することが可能であることも実証した。一方で理想的には、圧縮比はオン抵抗の最大分に対する、最大オン抵抗と最小オン抵抗の差の比まで圧縮することができるとしたが、結果的には 36%が限界であった。これは本手法を用いてもパスの長さによる遅延差を取りきることができないことを示しており、本手法の限界も示している。また素子数および配線面積が増大することも本手法の欠点である。この件に関しては次章でも検討する。

第7章　任意回路に対する遅延均衡化手法

前章では配置配線を考慮した任意回路を遅延均衡化するための手法を提案し、小規模な回路に適用することで検証を行った。本章ではより大きな回路規模の機能回路に対して遅延均衡化手法を適用することで、遅延均衡化手法の評価を行う。

提案している遅延均衡化手法は、設計フローの中にネットリスト→ネットリスト変換というフェーズを新たにもうけ、位置情報を含んだネットリストを出力するものである。また従来ウェーブ動作の回路では規則的な回路を対象にしているのに対し、フィードバックのない任意の回路に対して適用可能である。前章で構築した遅延均衡化手法をさらに実用的なものにするために、論理合成器との組合わせが必要不可欠であると考える。

そこで実際に既存の論理合成ツールを上位として、汎用回路のウェーブ化設計に関して検討を行った。上位ツールとしてハードウェア設計システム **PARTHENON** を使用した [2]。また対象回路として、現在 Web 上でソースが公開されている、**PARTHENON** システムで設計された 32 ビットプロセッサである **FDDP(Four-Day-Designed Processor)** を用い [2]、その中に含まれる部分回路に対して遅延均衡化を試みた。

7.1　遅延均衡化回路設計の手順

まず LSI 設計フロー全体における、本研究による遅延均衡化手法の位置づけを示し、今回実際に構築した LSI 設計フローを示す。本研究で提案する遅延均衡化手法は、論理合成段階が終わった時点でのネットリストを入力とする。以後ネットリスト→ネットリスト変換を行なった後テクノロジマッピングを行ない実配置配線をする流れの中で、一連の遅延均衡化戦略を適用する。故にアーキテクチャ設計および論理合成は別に行なう必要がある。

7.1.1　アーキテクチャ設計および論理合成段階

LSI 設計の上位にあたるアーキテクチャ設計から論理合成段階では、NTT で開発されたハードウェア設計支援システム **PARTHENON**(Parallel Architecture Refiner THEorized by Ntt Original coNcept) を使用した。動作記述言語 SFL をベースに、クロックに同期した回路設計に特化したシステムが特徴である。本研究では任意回路の遅延均衡化手法を研究の対象としており、今回は **PARTHENON** を単に論理合成ツールとして使用して

いる。そのため本研究ではその本質的な能力を生かしていないことをまず始めに断っておく。

PARTHENON システムの合成系では、回路合成は 2 段階に分けて行われる。まずテクノロジに依存しない部分で論理的な回路合成が行われ、続いてテクノロジに依存した部分で回路合成が行われる。本研究で提案した遅延均衡化も同様に、まずテクノロジに依存しない部分で段数を均衡化し、素子の仮想配置を行う。続いてテクノロジマッピングを行いながら実配置し、その際配線木の均衡化や素子のブロック化を行う。そこで現時点では、**PARTHENON** ではテクノロジに依存しない部分までの回路合成を行ない、そのデータを遅延均衡化ツールの入力とする。

7.1.2 ネットリスト変換から配置配線段階

論理合成された回路を入力とし、遅延均衡化作業に移る。論理合成終了段階で得られるネットリストには、テクノロジに依存する部分や位置情報などは含まれず、テクノロジに対して独立である。

まずネットリスト→ネットリスト変換を施す。ネットリストは独自形式のデータ構造に変換された後、段数づけが行なわれ、段数を均衡化するように素子が挿入される。この際必要に応じて負荷を分割しながら素子が挿入される。この段階で、素子に段数情報が付加される。次に入力段から順に仮想配置が行なわれる。配置は仮想グリッド上に配置され、素子はそれぞれ個々に仮想的な位置情報と見込み配線量を持つ。こうして変換されたネットリストは、挿入された DBC を含む結線情報を持ち、各素子毎に仮想的な位置情報と見込み配線量を持った状態で出力される。これが位置情報を持つネットリストである。この段階でも、テクノロジに対しては独立である。

配置配線段階では、位置情報を含んだネットリストを入力とする。まず全ての素子は、見込み配線量を参考にブロッキングされ、内部に α, β バッファを含むセルライブラリが動的に割り当てられる。次に回路の入力側から順に素子が配置される。この際配線領域やブロックのサイズなどの条件から、相対位置から絶対位置に変換される。また付加側の素子の位置が確定した段階で、実配線が行なわれる。今回は仮想配線木が生成されるが、実際には経路を探索しながら配線が行なわれる。配線が確定した段階で即座に遅延が計測され、駆動側の素子情報が更新される。最終的に実配置配線が終了し、個々のテクノロジに依存した回路が生成される。

7.1.3 構築したシステム

今回実際に構築したシステムのフローを以下に示す。フロー中、手順3. までは **PARTHENON** システムが提供するツール群を使用し、残りは独自の遅延均衡化ツールを使用する。

1. (フィードバックを含まない) 組合せ論理回路を SFL 言語で記述する。

2. 論理合成プログラム **SFLEXP** を適用し、nld 形式のネットリストを得る。

3. 論理合成プログラム **OPT_MAP** を適用し、テクノロジに依存しない範囲で論理最適化を行う。論理最適化された nld 形式のファイルを得る。

4. 得られた nld 形式のネットリストから、独自形式のネットリスト変換する (**NLDTONETLIST**)。

5. 独自形式のネットリストを対象に、テクノロジとは無関係な部分の遅延均衡化プログラムを適用し、位置情報を含んだネットリストを得る (**NETTONET**)。

6. 実配置／ (仮想) 配線を行い、評価する (**PLACEROUTE**)。

その他遅延均衡化／評価段階のサブモジュールとして、物理パラメータ計算プログラム **PARAM**、配線木評価プログラム **DELAYEVAL**、セルライブラリ生成プログラム **ELEMENTBASE(NORMAL,BLOCKED)** を独自に用意した。図 7.1 に構築したシステムの概要を示す。

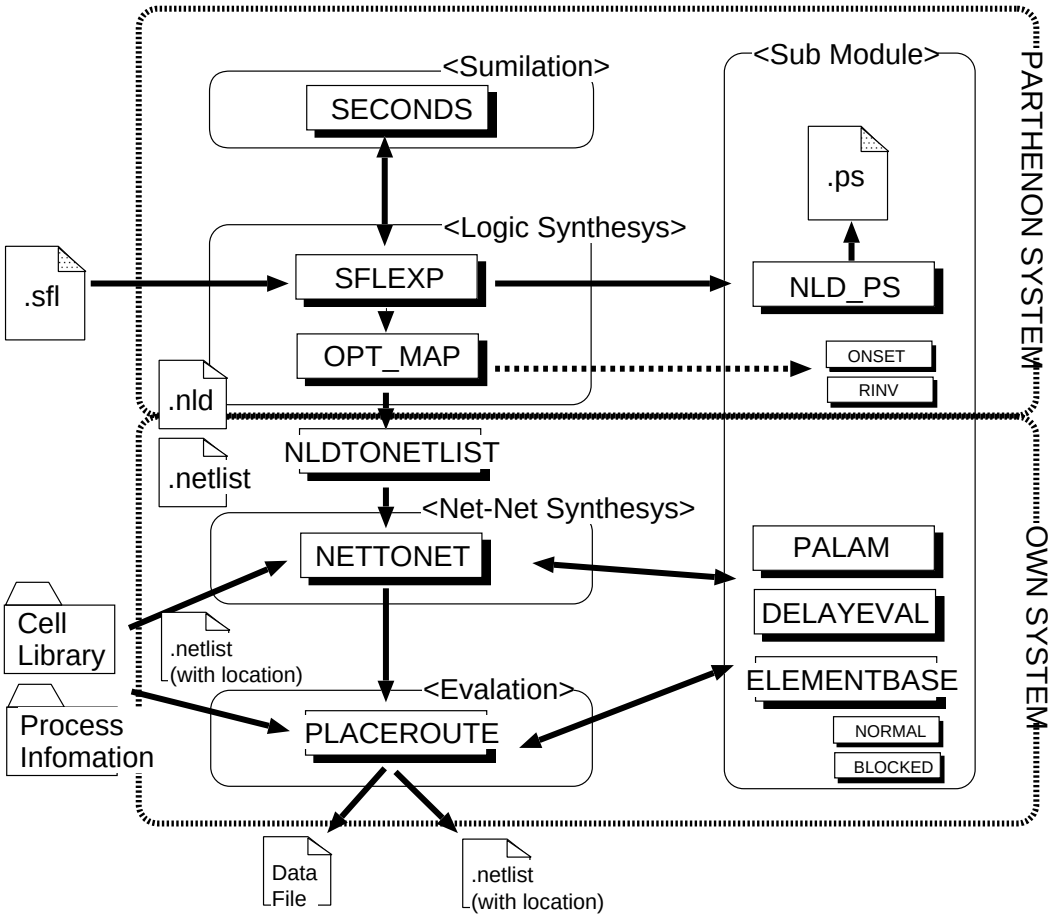

図 7.1: 汎用回路ウェーブ化システム関係図

7.1.4 対象とする回路

遅延均衡化の対象として、NTT 情報通信研究所にて実際に製作され、現在 Web 上で SFL 記述が公開されている 32 ビットプロセッサ **FDDP**(For-Day-Designed Processor) を使用した。**FDDP** は DLX のサブセットを命令に持ち、5 段パイプラインで各命令を処理する。CPU 内部は 32 ビット ALU や 16 ビット乗算器など個々の部分モジュールから構成されている。今回は **FDDP** の部分モジュールである 32 ビット ALU および 16 ビット乗算器に対して遅延均衡化手法を適用した。

7.2 32ビット ALU での検証

FDDP 内の 32 ビット ALU に対して遅延均衡化を試みた。ALU 内部は 32 ビット桁上げ先見加算器や論理演算回路からなり、外部信号線によって制御される。論理演算回路を通るパスや制御信号の多くが回路を大きくスルーしていくため、全体では不均衡な回路である。なお今回合成に際し、以下の制約を科している。

- 使用する素子はインバータ、2 入力 NOR/NAND および 3 入力 NOR/NAND の 5 種類。
- インバータは駆動用に 5 種類のサイズを用意。
- ファンアウト制約は通常 8 に取る。

PARTHENON での合成を行いファンアウト制約を取り除いた段階で、素子数は 1771 個、段数は 50 段になった。

比較実験として、二つの回路を用意した。最大遅延で動作する回路として、段数均衡化を行わず、1771 個の素子全てをブロック化することで各素子の駆動能力を上げた回路を用意した。一方ウェーブ化した回路として、前章での戦略区分 3-2. にあたる戦略により遅延均衡化を行った回路を用意した。

7.2.1 戦略別の評価および検討

遅延均衡化戦略での配線木の均衡化度合を見るため、各段毎の配線木の総量をプロットした図を 7.2 に示す。配線木均衡化戦略によって各段毎に確実に配線木の総量が均衡化されていることがわかる。

それでも配線木の容量はある程度ばらつく。次に固定／可変 β バッファによる発生遅延の違いを見る。図 7.3 に各素子の配線木の容量に対して、配線木で発生している最大遅延をプロットした図を示す。固定 β バッファ戦略では、一貫して一つのサイズの β バッファを使用している。可変 β バッファ戦略では、負荷の少ない方ではより小さなサイズの β バッファを使用し、大きな負荷を持つところでは、より大きなサイズの β バッファを使用

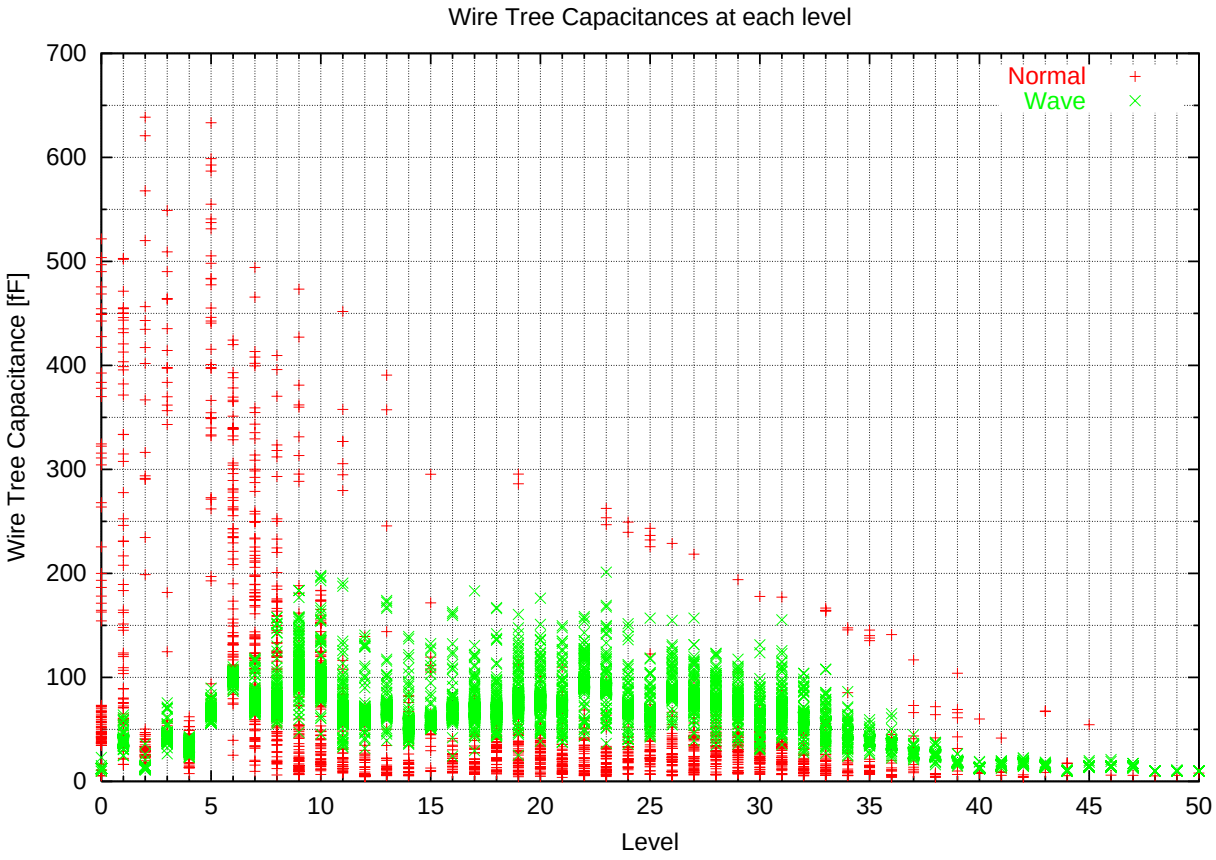

図 7.2: 32 ビット ALU における各段毎の配線木の容量

している。表 7.1 に、ある固定駆動 β バッファ戦略および可変駆動 β バッファ戦略による 32 ビット ALU の最終結果を、表 7.2 にパラメータ $\mathcal{A}$ の値を示す。可変戦略の方が若干 $\mathcal{A}$ の値は良いものの、性能差はほとんどない。

本遅延均衡化手法を大きな回路に対して適用する際には、β バッファの割り当て戦略が重要となる。可変 β バッファ戦略を考えると、β バッファのサイズは広い範囲にかつ細かい選択肢があることが望ましい。しかしそうでない場合は、大きな β バッファを固定して使用することで、発生する遅延差の縮小を狙った方がよいことも考えられる。使用可能な β バッファのサイズが豊富でない場合の戦略の一つとして、配線木の負荷平均が小さい段に対しては可変 β バッファ戦略を適用することでパス間の遅延をとり、配線木の負荷平均が大きい段に対しては大きなサイズのインバータによる固定 β バッファ戦略を適用することで、各パス毎で発生する遅延差を減らすなどの割り切った戦略が考えられる。

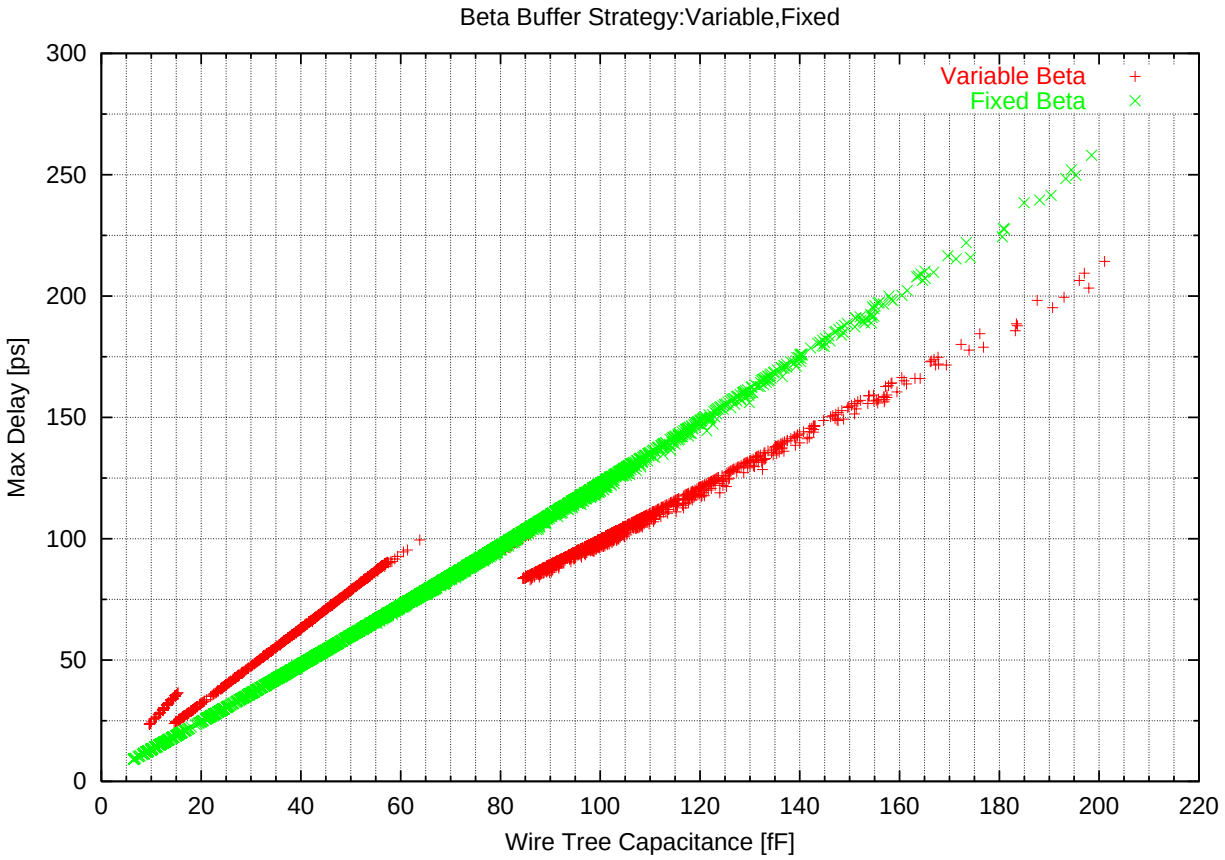

図 7.3: 配線木の容量に対する最大遅延

表 7.1: 戦略別による 32 ビット ALU での遅延均衡化結果

戦略	最大遅延 [ps]	最小遅延 [ps]	遅延差 [ps]	圧縮比	遅延 DBC 数	挿入 インバータ数	総配線長 [10^3grid]
固定戦略版	4905	2716	2189	0.446	9571	41826	2603
可変戦略版	4931	2829	2102	0.426	9571	41826	2717

表 7.2: 戦略別による 32 ビット ALU での遅延均衡化に関するパラメータ

戦略区分	最大遅延 [ps]	最小遅延 [ps]	遅延差 [ps]	圧縮比	$\mathcal{B} = 1.45$ の時の $\mathcal{A}$
固定戦略版	4905	2716	2189	0.446	1.24
可変戦略版	4931	2829	2102	0.426	1.20

7.2.2 最大遅延動作版とウェーブ動作版での比較

表 7.5 に最終的な結果を、表 7.6 には遅延均衡化度合を示すパラメータ $\mathcal{A}$ の値を示す。表中挿入 DBC 数とあるのは、ファンアウト制約を取るために挿入した分を含まない数を

指す。故に最大遅延動作版では 0 である。

表 7.3: 32 ビット ALU での遅延均衡化結果

戦略	最大遅延 [ps]	最小遅延 [ps]	遅延差 [ps]	圧縮比	遅延 DBC 数	挿入 インバータ数	総配線長 [10^3grid]
最大遅延動作版	4066	411	3655	0.899	0	3542	534
ウェーブ動作版	4931	2829	2102	0.426	9571	41826	2717

表 7.4: 32 ビット ALU での遅延均衡化に関するパラメータ

戦略区分	最大遅延 [ps]	最小遅延 [ps]	遅延差 [ps]	圧縮比	$\mathcal{B} = 1.45$ の時の $\mathcal{A}$
最大遅延動作版	4066	411	3655	0.899	6.83
ウェーブ動作版	4931	2829	2102	0.426	1.20

最終的な性能向上比で比較すると、ウェーブ動作版は最大遅延動作版に対して 1.93 倍高速に動作させることが可能になった。一方挿入したインバータを含め総素子数で比較すると、最大遅延動作版では 5313 個であるのに対しウェーブ動作版では 43597 個となり、8.21 倍に増加した。一方配線面積に関しては、実配線量で比較すると 5.09 倍増加した。グランド線も含めると 10.2 倍程度増加したことになる。また、最大遅延動作版では素子面積の 3 倍確保した。故に使用総面積概算では $(8.21 + 3 * 10.2)/(1 + 3) = 9.70$ 倍程度増加したことになる。

遅延均衡化の度合いで見ると、ウェーブ動作版では圧縮比を 42.6% まで抑えることに成功した。また表 7.2 から、パスの長さの違いによる遅延差の均衡化度合を示すパラメータ $\mathcal{A}$ は $1.00 - 1.20$ である。前章での 4 ビット ALU 同様、パスの長さの違いにより発生する遅延差を縮めることによる性能改善はほぼ極限に達しており、これ以上の性能向上を目指すならば動作環境の安定化を目指すべきである。

7.3 16ビット乗算器での検証

規則的な回路での例として、**FDDP** 内の 16 ビット乗算器に対して遅延均衡化を試みた。乗算器はキャリーセーブアダー (CSA) 方式を採用しており、部分和生成部分と桁上げ先見加算器からなる。なお、合成条件は 32 ビット ALU の場合と同様とした。**PARTHENON** での合成を行いファンアウト制約を取り除いた段階で、素子数は 3890 個、段数は 58 段であった。

比較実験として、32 ビット ALU の際と同様な二つの回路を用意した。図 7.4 に各段毎の配線木の容量をプロットしたものを、表 7.5 に最終的な結果を、表 7.6 には遅延均衡化度合を示すパラメータ $\mathcal{A}$ の値を示す。

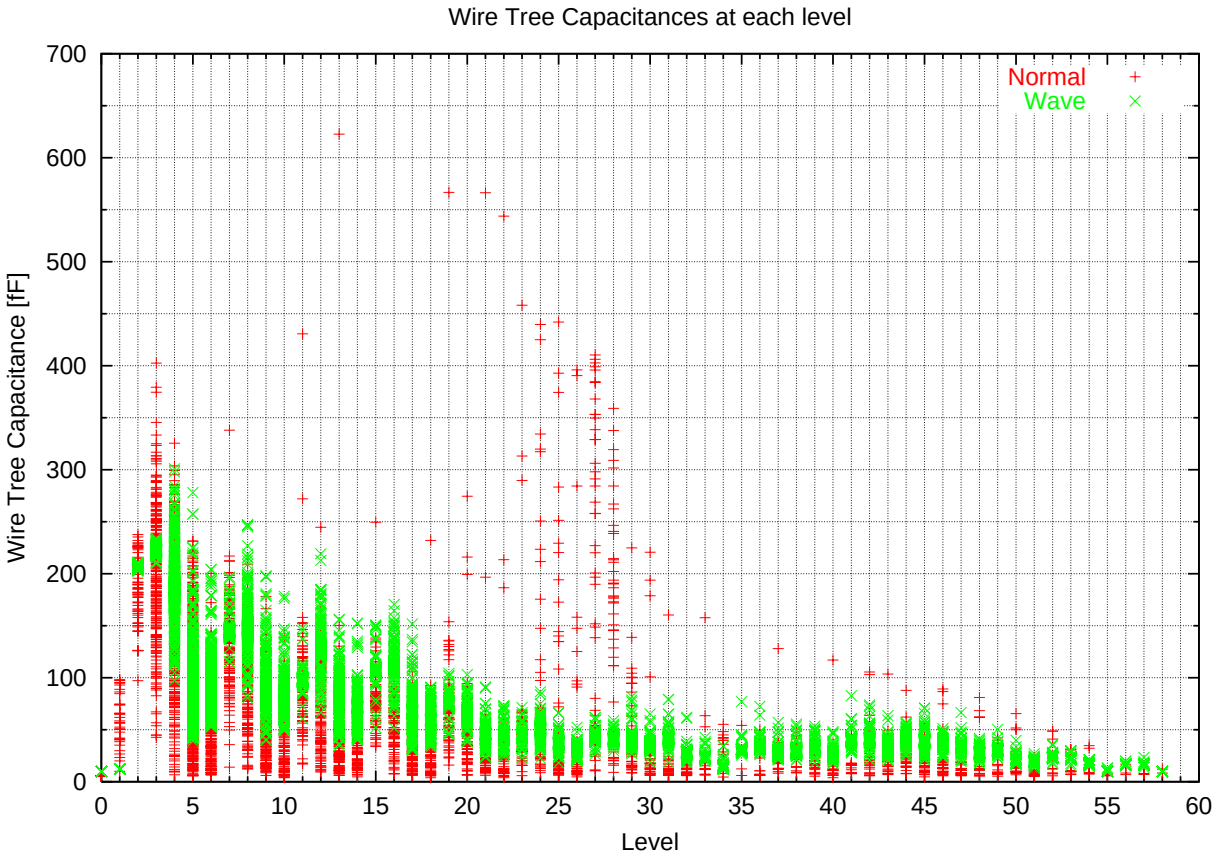

図 7.4: 16 ビット乗算器における各段毎の配線木の容量

表 7.5: 16 ビット乗算器での遅延均衡化結果

戦略	最大遅延 [ps]	最小遅延 [ps]	遅延差 [ps]	圧縮比	遅延 DBC 数	挿入 インバータ数	総配線長 [10^3grid]
最大遅延動作版	5219	499	4720	0.904	0	11670	1130
ウェーブ動作版	5885	3213	2672	0.454	6227	32688	2780

最終的な性能向上比で比較すると、ウェーブ動作版は最大遅延動作版に対して 1.95 倍高速に動作させることが可能になった。一方挿入したインバータを含め総素子数で比較すると、最大遅延動作版では 11734 個であるのに対しウェーブ動作版では 36386 個となり、3.10 倍となった。乗算器では制御信号や回路を大きくスルーするパスが少なく規則的であ

表 7.6: 16 ビット乗算器での遅延均衡化に関するパラメータ

戦略区分	最大遅延 [ps]	最小遅延 [ps]	遅延差 [ps]	圧縮比 [%]	$\mathcal{B} = 1.45$ の時の $\mathcal{A}$
最大遅延動作版	5219	499	4721	90.4%	7.18
ウェーブ動作版	5885	3213	2672	45.4%	1.26

るため、性能向上率に対して挿入するバッファ数は 32 ビット ALU に対して少なくてよい。一方配線面積に関しては、実配線量で比較すると 2.46 倍増加した。グランド線も含めると 4.92 倍程度増加したことになる。また、32 ビット ALU 同様最大遅延動作版において、配線面積は素子面積の 3 倍確保した。故に使用総面積概算では $(3.10+3*4.92)/(1+3) = 4.47$ 倍程度増加したことになる。

遅延均衡化の度合いで見ると、ウェーブ動作版では圧縮比を 45.4% まで抑えることに成功した。この値は 32 ビット ALU での圧縮比 42.6% と比較すると、規則的な回路であるにもかかわらず 16 ビット乗算器での圧縮比はやや劣る結果となった。また表 7.6 から、パスの長さの違いによる遅延差の均衡化度合を示すパラメータ $\mathcal{A}$ は $1.00 - 1.26$ である。前二つの回路と比較すると圧縮比はやや劣るとはいえ、パスの長さの違いにより発生する遅延差短縮による性能改善と、動作条件の違いによる遅延差短縮による性能改善とを比較して、前二つの回路同様これ以上の性能向上を目指すならば動作環境の安定化を目指すべきであろう。

7.4 結論

本章では既存の論理合成ツールとの組合せによる任意回路のウェーブ化システムを構築し、CPU の部分回路に対して遅延均衡化手法を適用し検討を行った。まず設計過程の概要を述べ、**PARTHENON** システムと本研究で提案する遅延均衡化手法との組合せを検討した。その後 32 ビットプロセッサ **FDDP** 内の機能回路に対して遅延均衡化手法を適用し、最大遅延動作の回路との比較検討を行った。遅延均衡および性能向上という点で、論理合成／再 (実) 配置配線することなく、大規模で不規則な回路に対しても本手法が十分効果を発揮することを実証した。一方で遅延均衡化に伴う面積の増加率に対して、本手法での遅延均衡化による性能向上率はよくないことも明らかになった。これは回路を大きくスルーして行くパスが多い場合、どうしても挿入バッファ数が多くなってしまうことは避けられないことと、配線木均衡化に伴う配線量の増加およびグランド線による面積増加が要因である。また、32 ビット ALU および 16 ビット乗算器ともにウェーブ動作で約 2 倍の性能向上を達成した。このことは、本手法が不規則／規則的な回路のどちら対しても同程度の性能改善が見込めることを意味する。一方で規則的な回路の方が挿入素子数や配線領域の増加が少ないことも明らかになった。

今回は **PARTHENON** システムを論理合成器として用いる設計手法を構築した。しかしながら、**PARTHENON** システムの真髄はアーキテクチャ設計と回路設計以降を完全に分離し、ユーザにテクノロジを意識させない設計を可能にする点にある。**PARTHENON** システムにウェーブ動作も考慮したタイミング設計を行える機能を持たせることができれば、**PARTHENON** システムを使った高位段階からの汎用回路のウェーブ化にも道が拓けるだろうと考えている。例えば同期パイプラインの一部をウェーブ動作させたり、あるいは全体をウェーブ動作できるようにクロックパスを生成できるようなシステムを組み込むことなどが考えられる。この件に関しては、今後の目標として挙げておくにとどめる。

第8章　総括

本研究では、ウェーブパイプラインアーキテクチャにおける遅延均衡化手法を提案した。以下、本研究の内容を簡単にまとめる。

まず第2章では本研究が対象としているウェーブパイプラインアーキテクチャの理論および研究動向に関して述べた。ウェーブパイプラインアーキテクチャでの遅延均衡回路の必要性と、本研究の位置づけを示した。

第3章で小さなトランジスタサイズの下でのMOSFETおよび各層に関するパラメータ、遅延評価方法について示した。それを元に第4章ではCMOS構成でのウェーブパイプラインアーキテクチャの可能性に関して考察を行なった。二つの章を通して、CMOS論理上でウェーブパイプラインを行うことの問題点として、パスの長さにより生じる遅延差、多入力素子で生じる遅延差および動作環境の違いにより生じる遅延差を取り上げた。また動作環境を含めた場合の理想的な回路に関して触れた。

第5章から第6章まで、本研究で提案する遅延均衡化手法を二つのフェーズに分けて述べた。第5章では、本研究が提案する遅延均衡化手法の理念を述べるとともに、まず素子の遅延差に着目した遅延均衡化手法を提案し、全加算器を対象に遅延均衡化を行うことで検討を行った。第6章で配置配線問題を考慮した遅延均衡化手法を提案し、4ビットALUを対象として遅延均衡化を行うことで検討を行なった。第7章で今回構築したシステムの流れを示し、比較的大規模な機能回路に対して遅延均衡化を試みた。

ここで本研究が対象とした問題領域を整理する。第3章第3節で遅延差を性質毎に分類した。本研究が遅延均衡化の対象としたのは、その中で多入力素子で発生する遅延差とパス間での遅延差である。またディープサブミクロンデバイスでは配線の影響を強く受けるため、配置配線を考慮した遅延均衡化手法が必要となることを挙げた。従来からのインバータ挿入では、挿入した後論理合成段階に戻って再配置配線を行なう必要があり、スパイラル設計を必要としていた。以上を踏まえた上で、任意の回路全体に対して適用でき、極限まで遅延均衡化を達成する方式を提案することが本研究の最終的な目標であった。

CMOS論理上の多入力素子で発生する遅延差に関しては、小さなインバータを多入力素子の近くに置くことで直接の負荷を減らす戦略を提案した。また負荷側を駆動する際には新しく大きなインバータを用意することを提案した。配置配線を考慮する段階では、テクノロジマッピングの段階でこれら二つのインバータと多入力素子を同一セル内に置くことで、多入力素子にかかる配線負荷を最低限に抑えかつ各インバータの配置問題を簡単化

することを提案した。パス間での遅延差に関しては、段数の均衡化、各素子の配線木の均衡化および駆動素子の可変化などの各戦略を提案した。そのためにネットリストの段階で仮想配置することで遅延情報を得て、ネットリスト-ネットリスト変換によってパス間の遅延均衡を行なうことを提案した。これにより、論理合成段階から配置配線段階までトップダウンで遅延均衡化を行なうことを可能にした。

以上の各戦略により、不均衡／均衡な回路にかかわらず性能向上を達成することができた。パスの長さの違いによる遅延差および一つのパスに関する遅延差両方に関して、ウェーブ動作によるこれ以上の性能向上を目指すならば、残る遅延差発生要因である動作環境の変化により生じる遅延差を押え込むべきだと十分判断できるところまで均衡することができた。故に、パスの長さの違いによる遅延差を極限まで解消することに成功し、同時に一つのパス内で発生する遅延差に関しても、インバータでの最大オン抵抗に対する最小オン抵抗の比に近い値まで押え込むことに成功したと言える。一方で本戦略では挿入素子数および配線量の増加に関してはほぼ野放しで、不均衡な回路であれば面積が 10 倍にもなることもまた明らかとなった。最大遅延動作の回路に対して性能向上比が約 2 倍であることを考えると、コストパフォーマンスはよくない。しかしトランジスタ数は現在も豊富でこれからも増え続けること、また動作環境を安定化させることによりさらに性能を向上させることが可能であり、本提案によるウェーブパイプラインアーキテクチャは一つの選択肢として十分なり得ると考えている。

本研究の今後の課題として、実際の設計ツールへの本格的な組み込みや、実 LSI 上での検証が挙げられる。今回論理合成ツールとして使用した **PARTHENON** との組合せは一つの案である。同期回路／ウェーブ回路混合設計のためには、クロックタイミング生成／グローバルルーティング／機能回路単位での配置問題などを検討する必要がある。また本手法と安定な動作環境との組合せによる、より高速な動作の可能性を検証することも挙げられる。さらに実用的なものに仕上げるためには、ウェーブパイプラインアーキテクチャを生かした機能回路を考える必要もある。CPU アーキテクチャであれば、マルチスレッドプロセッサや TM32 のような古典的な VLIW アーキテクチャとの相性がよいだろう。

以上が本研究のまとめである。今後ますます LSI 回路は大規模化し、それに伴い設計の各段階はますます分離されていくであろう。しかし真に優れた回路、真に優れたシステムを設計するためには、物理的な要因が統べるプロセス段階から、回路設計、アーキテクチャ設計、そしてソフトウェア設計に渡る段階まで、幅広い段階を考慮する必要があると考えている。特にアーキテクチャ設計段階と回路設計、プロセス設計の各段階に関しては、ハードウェア記述言語の普及によりますます分離傾向にあり、各段階を結ぶ強力な技術が必要とされている。本研究ではウェーブパイプライン設計においてそれら段階を一つに統合することを提案したが、同時に本論文が各設計段階を結ぶ技術の必要性を訴える論文の一つになればと考える。

付 録A　セル設計とレイアウトに関する付録

本章では、本研究で設計したセルおよび関連するデータを載せておく。ここに載せたデータは本文中で取り決めた仮定に基づくシュミレーション値であり、あくまでも参考程度に過ぎないことをあらかじめ断っておく。

A.1　基本レイアウト

以下インバータに絞って話を進める。図 A.1 に最小サイズのインバータのレイアウト図を示す。ここで図中 g はゲートサイズの大きさを示すパラメータであり、PMOS および NMOS の幅の和 $W = W_p + W_n$ が $0.20\mu m$ の時、$g = 1.0$ とおく。最小サイズのインバータで $g = 3.5$ である。

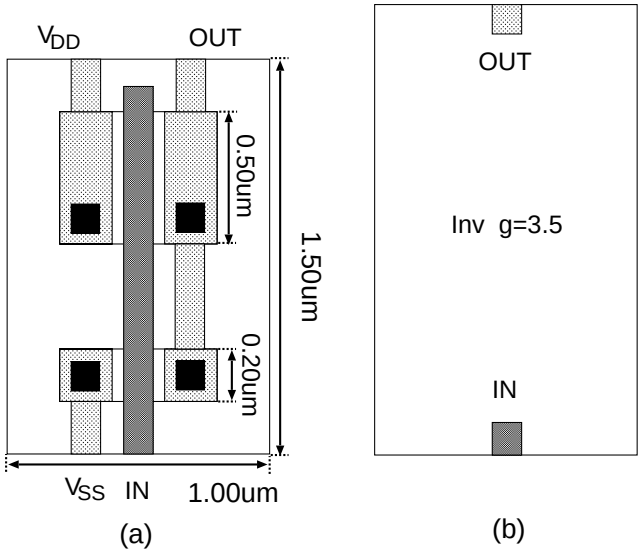

図 A.1: リファレンスインバータのレイアウト

本研究では配置配線のしやすさから通常のセル設計とは異なるスタンスを取っている。まず入力ピンおよび出力ピンは、ゲートの高さ方向に出す。これは回路の成長方向と配

線方向を一定の向きにそろえることを狙ったものである。実際には最小寸法のサイズ (図 A.1(a)) に対して、出力ピンと入力ピンを同一軸上に置く設計とした (図 A.1(b))。また MOSFET の幅を大きくする際には、その成長方向はセルの高さ方向とし、セルの幅方向は常に一定の大きさを保つ。これは通常の方式とは逆になるが、素子 (およびブロック) の成長方向と回路全体の成長方向、配線方向をそろえることを狙ったものである。本研究では段毎の配置を行って行くので、セルの幅が一定である方が都合がよい。但し各段でセルの高さが不揃いになるという欠点があるほか、一方向に均一で長い拡散層を生成する必要があるなど製作上の困難がある。概観図を図 A.2 に示す。

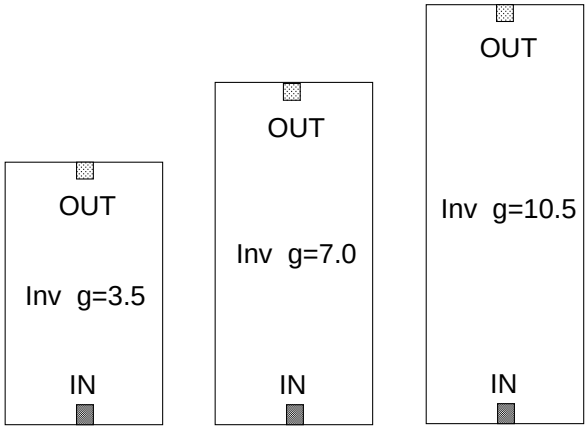

図 A.2: サイズの異なるインバータ

次に α, β バッファでブロック化されたインバータの概要図を図 A.3 に示す。ブロック化されたセルは 3 つの領域からなる。素子エリア (Element Area) には通常の論理素子が置かれる。各種の論理素子で一番最小となるサイズの素子がここには置かれるが、それぞれは高さが一定になるように設計されている。一方で幅は素子毎に異なるため、本研究では一番大きなサイズの素子に合わせた幅が常に確保される。面積をより有効に使うならば、幅を可変にすることも考えられる。より入力数の多い素子を使用する場合は、幅を可変にすることが必要となるだろう。内部電源ライン領域 (Inner Power Supply Line Area) には内部電源供給用の線が配線される。ブロッキングエリア (Blocking Area) には α, β バッファの各素子が置かれる。α バッファは一番小さなサイズのインバータが置かれ、β バッファは負荷側の配線木や負荷数に応じて高さ可変のインバータが置かれる。また各素子間はセル内部で結線されている。なお今回は、β バッファとして 4 つのサイズのインバータを用意した。

セルの幅を固定にし高さを可変にすることおよび素子をブロックすることは、α, β バッファの配置問題を楽にしかつできるだけ該当素子の近くに α, β バッファを置くことを狙ったものである。一方面積や配置形状という点からすれば、無駄な部分も多い。図 A.4 は通常行われる高さ一定のレイアウトと、本研究で採用した幅一定のレイアウトの概観図である。本研究ではどのようなブロックに対しても一定の領域を確保するため、実際には図中 (b) のように高さが不揃いのブロックが並ぶことになる。しかもブロック内部にも使用さ

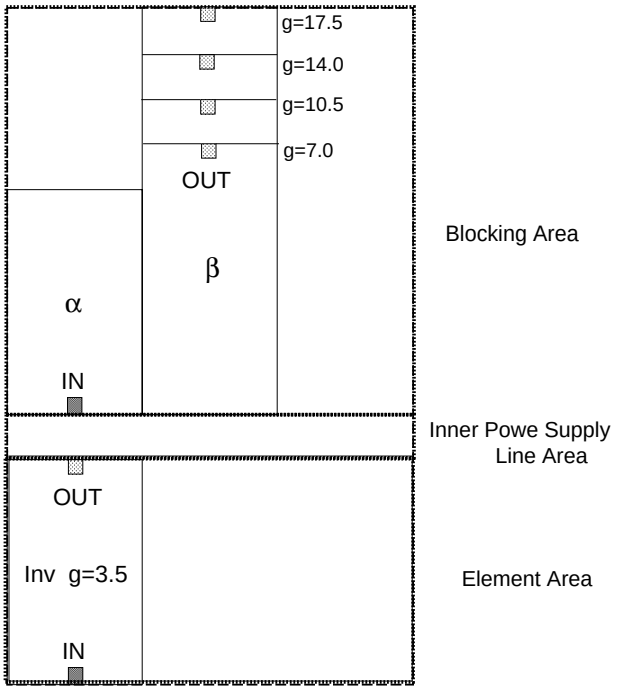

図 A.3: ブロック化されたインバータのレイアウト

れていない領域が存在する。本研究では遅延均衡化という観点からこのような設計としたが、ここに示した方針は一例であり、より優れた設計が可能であると考える。

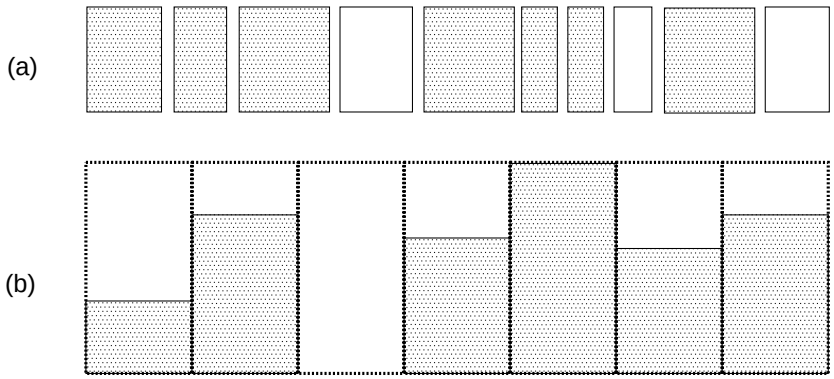

図 A.4: (a) 高さ一定のレイアウトと (b) 幅一定のレイアウト

A.2 各種データ

A.2.1 不純物濃度、移動度および飽和速度に関するパラメータ

第 3 章で得た不純物濃度、移動度および飽和速度に関する HSpice パラメータは以下の通りである。

表 A.1: 不純物濃度、移動度および飽和速度に関するパラメータ

パラメータ名	名前	単位	数値 ($NMOS$)	数値 ($PMOS$)
NCH	チャネル不純物濃度	$1/cm^3$	$2.35 * 10^{17}$	$2.12 * 10^{17}$
NSUB	基板不純物濃度	$1/cm^3$	$2.12 * 10^{17}$	$1.40 * 10^{17}$
U0	移動度	$cm^2/V \cdot sec$	263.33	117.96
UA	移動度劣化の一次係数	m/V	$-1.36 * 10^{-9}$	$1.65 * 10^{-9}$
UB	移動度劣化の二次係数	$(m/V)^2$	$2.25 * 10^{-18}$	$1.17 * 10^{-21}$
UC	移動度劣化係数の基板効果	m/V^2	$5.20 * 10^{-11}$	$-1.00 * 10^{-10}$
VSAT	飽和速度	m/sec	$1.08 * 10^5$	$2.08 * 10^5$

A.2.2 第 5 章で使用したパラメータ

第 5 章で仮定した MOSFET モデルは、酸化膜厚 $t_{ox} = 2.0[nm]$、ゲート長 $L = 0.10[\mu n]$ および動作環境として動作温度 27C ／動作電圧 1.3V の一定の環境を仮定した、高速に動作する MOSFET を想定している。表 A.2 に HSpice のシミュレーションにより求めた各素子のパラメータを、表 A.3 に Logical Effort モデルにより評価する際に使用したパラメータを示す。実際の評価では、最小パラメータは入力素子数で割ったものを使用する (同時にオン／オフになる抵抗の分速度が速くなる)。なお式 (5.2) において、$\tau_{min} = 2.17$ であった。

表 A.2: 第 5 章で使用した各種類における最小サイズの素子のパラメータ

ライブラリ名	ピン番号	$R_{max}[k\Omega]$	$R_{min}[k\Omega]$	$C_{pmax}[fF]$	$C_{pmin}[fF]$
inv		2.12	1.89	0.616	0.460
nand2	1	1.21	0.548	1.44	2.58
	2	1.27	0.548	2.16	2.58
nor2	1	1.41	0.817	1.68	2.62
	2	1.43	0.817	3.34	2.61
nand3	1	1.19	0.350	1.81	4.99
	2	1.25	0.350	3.05	4.99
	3	1.37	0.350	3.85	4.99
nor3	1	1.29	0.596	2.27	4.49
	2	1.32	0.596	4.76	4.49
	3	1.35	0.596	6.61	4.49

表 A.3: 各多入力素子のパラメータ ($\tau_{min} = 2.17$)

素子名	端子番号	最大		最小	
		g_{max}	p_{max}	g_{min}	p_{min}
INV		1.12	0.60	1.00	0.54
NAND2	1	1.28	0.80	1.16	1.30
	2	1.34	1.26	1.16	1.30
NOR2	1	1.49	1.09	1.73	1.97
	2	1.51	2.19	1.73	1.97
NAND3	1	1.52	0.99	1.34	2.41
	2	1.60	1.76	1.34	2.41
	3	1.75	2.43	1.34	2.41
NOR3	1	1.65	1.35	2.28	3.69
	2	1.69	2.90	2.28	3.69
	3	1.72	4.10	2.28	3.69

A.2.3 第 6 章で使用したパラメータ

表 A.4 に各種類における最小サイズの素子の各パラメータを示す。各素子での最大遅延時のオン抵抗が、インバータの最大遅延時のオン抵抗に揃うように設計した。故に各素子におけるオン抵抗は、インバータでのオン抵抗を基準に定めている。また各素子での拡散容量に関しては、ピン毎に HSpice にてシミュレーションを行って得た。第 3 章で比較対象とした TSMC 社のオン抵抗値で比較すると、高速に動作するデバイスである。

表 A.4: 各種類における最小サイズの素子の各パラメータ

ライブラリ名	ピン番号	$R_{max}[k\Omega]$	$R_{min}[k\Omega]$	$C_{pmax}[fF]$	$C_{pmin}[fF]$
inv		4.28	2.95	0.408	0.415
nand2	1	4.28	1.48	0.554	0.769
	2	4.28	1.48	0.902	0.769
nor2	1	4.28	1.48	0.714	1.29
	2	4.28	1.48	1.67	1.29
nand3	1	4.28	0.983	0.709	2.03
	2	4.28	0.983	1.29	2.03
	3	4.28	0.983	1.76	2.03
nor3	1	4.28	0.983	1.07	3.06
	2	4.28	0.983	2.55	3.06
	3	4.28	0.983	3.86	3.06

各素子での最大遅延時のオン抵抗がインバータの最大遅延時のオン抵抗に揃うように設計したため、移動度が小さいPMOSFETを直列に繋ぐ必要があるNOR素子では、必然的にゲートサイズが大きくなる。そのためこのような方針に基づいたセル設計では、NOR素子の面積／拡散容量はNAND素子に比べて大きくなってしまうことは避けられない。

表A.5にβバッファサイズの違いによる各ブロック素子の内部遅延を示す。ここでdbとはブロック化されたDBCを指し、インバータ2個からなる素子をブロックしたものである。

表 A.5: βバッファサイズの違いによる各ブロック素子の内部遅延

素子名	最大／最小遅延 [ps]	βバッファのサイズ			
		$g = 7.0$	$g = 10.5$	$g = 14.0$	$g = 17.5$
inv	最大	11.1	14.3	17.5	20.7
	最小	9.6	12.0	14.4	16.9
db	最大	16.1	19.5	22.6	25.8
	最小	12.3	14.5	16.7	19.0
nand2	最大	12.6	15.9	19.8	23.2
	最小	8.1	10.3	12.7	15.1
nor2	最大	16.7	20.2	23.4	26.6
	最小	8.8	11.5	14.4	17.1
nand3	最大	15.7	18.6	22.2	25.8
	最小	7.2	9.5	11.8	14.1
nor3	最大	26.4	30.1	33.7	37.1
	最小	7.2	9.9	12.5	15.2

参考文献

[1] The MOSIS service, http://www.mosis.org/.

[2] PARTHENON web site, http://www.kecl.ntt.co.jp/parthenon/.

[3] Semiconductor Industry Assciation. Nathinal technology roadmap for semiconductors, 2001.

[4] Eduardo I. Boemo, Sergio Lopez-Buedo, and Juan M. Meneses. The wave pipeline effect on LUT-based FPGA architectures. In *FPGA*, pp. 45–50, 1996.

[5] Eduardo I. Boemo, Sergio L.pez-Buedo, and Juan M. Meneses. Some experiments about wave pipelining on FPGA's. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 6, No. 2, pp. 232–237, 1998.

[6] Mark Bohr. Industry-leading transistor peformance demonstrated on inter's 90-nanometer logic process. *Interl Reserch and Developemnt*, 2002.

[7] W.P. Burleson, M. Ciesielski, F. Klass, and W. Liu. Wave-pipelining: A tutorial and research survey. *IEEE Transactions on VLSI Systems, Vol.6, No.3, Sep.'98, pp. 464-470.*, 1998.

[8] Yohua Cheng and Chenming Hu. MOSFET MODELING and BSIM3 USER'S GUIDE 日本語版. 丸善株式会社, 2002.

[9] M. J. Flynn et.al. The SNAP Project: Towards Sub-Nanosecond Arithmetic. In S. Knowles and W. H. McAllister, editors, *Proc. 12th IEEE Symposium on Computer Arithmetic*, 1995.

[10] M.J. Flynn. What's ahead in computer design? In *23rd EUROMICRO Conference '97 New Frontiers of Information Technology September 01 - 04*, 1997.

[11] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach 3rd Edition.* Morgan Kaufmann Publishers, 2002.

[12] Kevin J. Nowka Hidechika Kishigami and Michael J. Flynn. Delay balancing of wave pipelined multiplier counter trees using pass transistor multiplexers. Technical Report CSL-TR-96-692, 1996.

[13] B.Sproull I.Sutherland. Logical effort:desiging for speed on the back of an envelope. In *Advanced Research in VLSI,Proceedings of the 1991 University of California, Santa Cruz,Comference*, 1999.

[14] B.Sproull I.Sutherland and D.Harris. *Logical Effort:Desiging Fast CMOS Circuits*. Morgan Kaufmann Publishers, 1991.

[15] J.Rubinstein, P.Penfield Jr, and M.A.Horowitz. Signal delay in RC tree networks. *IEEE Trans. Computer-Aided Design Integrated Circuits Syst*, Vol. CAD-2, pp. 202–211, July 1983.

[16] K. Nowka and M. Flynn. Wave pipelining of high performance CMOS static RAM, 1994.

[17] Kevin J. Nowka. High-performance CMOS system design using wave pipelining. Technical Report CSL-TR-96-693, 1996.

[18] Kevin J. Nowka and Michael J. Flynn. Environmental limits on the performance of CMOS wave-pipelined circuits. Technical Report CSL-TR-94-600, 1994.

[19] M. Garg O. Hauck and S. A. Huss. Efficient and safe asynchronous wave-pipeline architectures for datapath and control unit applications. In *9th Great Lakes Symposium on VLSI*, 1999.

[20] Paul Packan Scott Thompson and Mark Bohr. Mos scaling:transistor challenges for the 21st century. *Intel Technology Journal 3rd quarter '98*, 1998.

[21] Inc. Sequence Design. Impact of coupling on interconnect delays:evil is in the modeling, http://www.sequencedesign.com/, 2002.

[22] Amit Singh, Arindam Mukherjee, and Malgorzata Marek-Sadowska. Interconnect pipelining in a throughput-intensive fpga architecture. In *Ninth international symposium on Field programmable gate arrays*, pp. 153–160. ACM Press, 2001.

[23] Scott Thompson.et.al. 130nm logic technology featuring 60nm transistors,low-k dielectrics,and cu interconnects. *Intel Technology Journal*, 2002.

[24] T.Sakurai and K.Tamaru. Simple formulas for two- and three dimensional capacitances. *IEEE Transaction on Electron Device*, Vol. ED-30, No. 2, p. 2, Feb 1983.

[25] W.C.Elmore. The transient response of damped liner networks with particular regard to wire-band ampilifers. *Journal of Applied Physics*, Vol. 19, pp. pp.55–63, July 1948.

[26] Y.I.Ismail, E.G.Friedman, and J.L.Neves. Equivalent elmore delay for RLC trees. *IEEE Transactions on Computer-Aided Design, Vol. 19, No. 1*, Vol. 19, No. 1, pp. 83–97, January 2000.

[27] 芝山達也. 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. Master's thesis, 北陸先端科学技術大学院大学, 2001.

[28] 松居明宏. ウェーブパイプラインによる低消費電力プロセッサ設計と試作による性能実証. Master's thesis, 北陸先端科学技術大学院大学, 2001.

[29] 大石亮介. フィードバックのあるパイプライン回路のウェーブ化に関する研究. Master's thesis, 北陸先端科学技術大学院大学, 2001.

[30] 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. Master's thesis, 北陸先端科学技術大学院大学, 1999.

謝辞

本研究を進めるにあたり、研究にかかわることだけでなく人生の先輩としても常に熱心にかつ冷静な御指導を頂きました、日比野靖教授にまず深く感謝致します。また同じ計算機アーキテクチャ講座の田中清史助教授には、時に厳しい御指摘を賜わりましたことを感謝致します。堀口進教授からも貴重な御意見を頂きました。また同講座の宮崎純助手にはさまざまな場面で御意見を頂くなど、普段からお世話になりましたことを深く感謝致します。

日比野·田中研究室の皆様には、同じ分野を学び研究する者の立場からさまざまな御指摘を頂きました。講座外の本学の仲間達からも、他分野からの多くの刺激を頂きました。この大学で皆様と同じ時代に同じ場所で学業に励むことができたことを誇りに思います。本学外の友人からも、たくさんの励ましを頂きました。

最後に、経済的余裕もなく常に病が絶えない家庭状況の中、ご支援くださった親戚の方々に、常に力強く励ましてくださった兄に、そしてこれまで暖かい愛情で育ててくださった両親に、心からの感謝を申し上げます。