修 士 論 文

低消費電力プロセッサ設計に関する研究

指導教官　日比野靖 教授

審査委員主査　日比野靖

審査委員　篠田陽一

審査委員　堀口進

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

内藤 郁之

2001 年 2 月

要 旨

近年、プロセッサの性能の向上にともない、プロセッサの消費電力が増加している。一方、PDA や携帯電話の普及により低消費電力で高性能なプロセッサが必要となっている。本研究では、低消費電力を電圧の制御回路の付加やソフトウエア処理の導入により実行するのではなく、回路の設計方法をあらため、トランジスタのゲート幅 W の大きさを小さくすることで実現する。また、その結果、プロセッサの動作周波数が低下するが、ウエーブパイプラインの導入により周波数の向上をはかり、これらが低消費電力に対し有効であることを示す。

目 次

第 1 章

はじめに

近年、プロセッサの性能の向上にともない電力消費が増加している。現在、低消費電力で動作させるために、動作電圧を下げる、命令の実行に関係のない部分には電圧をかけないという方法が取られている [6]。これらの方法のうち前者では、処理速度が落ちるという問題がある。また、後者の方法では、制御回路が必要になり、回路が複雑になるということがあげられる。この問題を解決するために処理能力を下げずに消費電力を低下させる方法が必要となる。これを実現するために、回路中のトランジスタの大きさを小さくし、その結果、遅延時間の増加が起きるが、パイプラインに最大遅延と最小遅延の差で動作するウエーブパイプラインを導入することにより、低消費電力で、高速なプロセッサが実現可能なことを示す。

まず、第 2 章では、プロセッサの電力と CMOS トランジスタの動作原理から低消費電力化を実現する方法を述べる。第 3 章では、対象とするプロセッサのプロセスルールに沿ったトランジスタの特性を算出し、セルライブラリの特性を算出した。第 4 章では、電力測定のための方針とその結果を示す。第 5 章では、4 章の結果に基づき、考察を行なう。

第 2 章

低消費電力化を実現するための設計方法

プロセッサの電力の式から容量を減らすことで電力が減少することを示し、容量の減少にともなう影響について調べる。さらに、さらに、遅延モデルと電力モデルについても述べる。

2.1 プロセッサにおける電力

CMOS 技術におけるプロセッサの電力 Pは、以下の式で表される。

$$P = \sum CV^2 f \tag{2.1}$$

ここで、電力 Pは、容量 C、周波数 fおよび電圧の 2 乗 V^2に比例していることが分かる。したがって、これらの 3 つのパラメタを小さくすれば、電力は小さくなる。電圧および周波数を下げた場合は、パフォーマンスが低下する。また、容量を減らす方法では、動作をしていないブロックの電圧をかけないという方法があるが、この場合、制御回路が必要となる。容量を削減する方針として小さいトランジスタを用いてプロセッサを設計する。そのため周波数が低下するが、それに対し、パイプラインを最大遅延と最小遅延の差で動作するウエーブパイプラインにすることで、動作周波数の向上が可能になるため性能の維持を実現する。

2.1.1 Crusoe における低消費電力の実現方法

本節では、低消費電力プロセッサである Transmeta 社の crusoe[8] プロセッサの電力制御に関して簡単に述べる。

低消費電力プロセッサとして TRANSMETA の crusoe プロセッサがある。このプロセッサは、intel の x86 プロセッサの互換として動作するように設計してある。crusoe では、cpu のコアは、VLIW のプロセッサとして設計されいる。命令は、まず、コードモーフィングソフトウエアにより x86 の命令から、crusoe の VLIW 命令に書き換えられ、変換された命令がプロセッサで実行される。VLIW は 4 命令を順序通りに実行する方式になっており、アウトオブオーダー実行により実行後にリオーダーバッファにより整列される方式と比較すると、回路が簡単になるという性質を CRUSOE は持っている。ソフトウエアにより変換された命令がプロセッサで実行されるため、crusoe ではプロセッサのコアを簡単な構成にし、ダイサイズを小さくすることで低消費電力を実現している。そのため、通常のプロセッサに比べるとソフトウエア処理が行なわれるためパフォーマンスの低下がおこる。

また、crusoe では、電圧と周波数の組合せを複数用意してあり、プロセッサの負荷に応じてこの組合せを動的に変更することで、必要以上の電力を消費しないようになっている。この動的な変更はまず、周波数を下げ、そして、電圧を下げるという方針で行なわれている。また、この変更はアプリケーションの実行中かどうかに関わらず行なわれる。

また、通常行なわれる、アイドル状態の連続に対して、一定時間経つとクロックの供給を止めるという方法も採用されている。さらに、大きい命令キャッシュを用意することで、連続した命令に対してのパフォーマンスを維持するような設計になっている。同じ命令が入ってきた時は、code morphing software を介さずに、キャッシュから命令をロードして実行するようになっている。

crusoe では、ソフトウエア処理が導入されており、これがパフォーマンスを低下させる原因となっている。

2.2 MOSトランジスタの動作原理

MOS トランジスタの基本的な動作原理は、nMOS トランジスタの場合、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負電荷の制御である。トランジスタは、ゲートに電圧がかかりソースからドレインに電流がながれることにより動作する。また、トランジスタの動作速度は、キャリアの種類（電子、またはホール）により定まる移動度μとゲート長 L で決定される。図 2.1 のようにゲートの各パラメータを定義する。このとき、電子の平均速度 vは、電界 Eを用いて表すと次のようになる。

$$v = \mu E$$

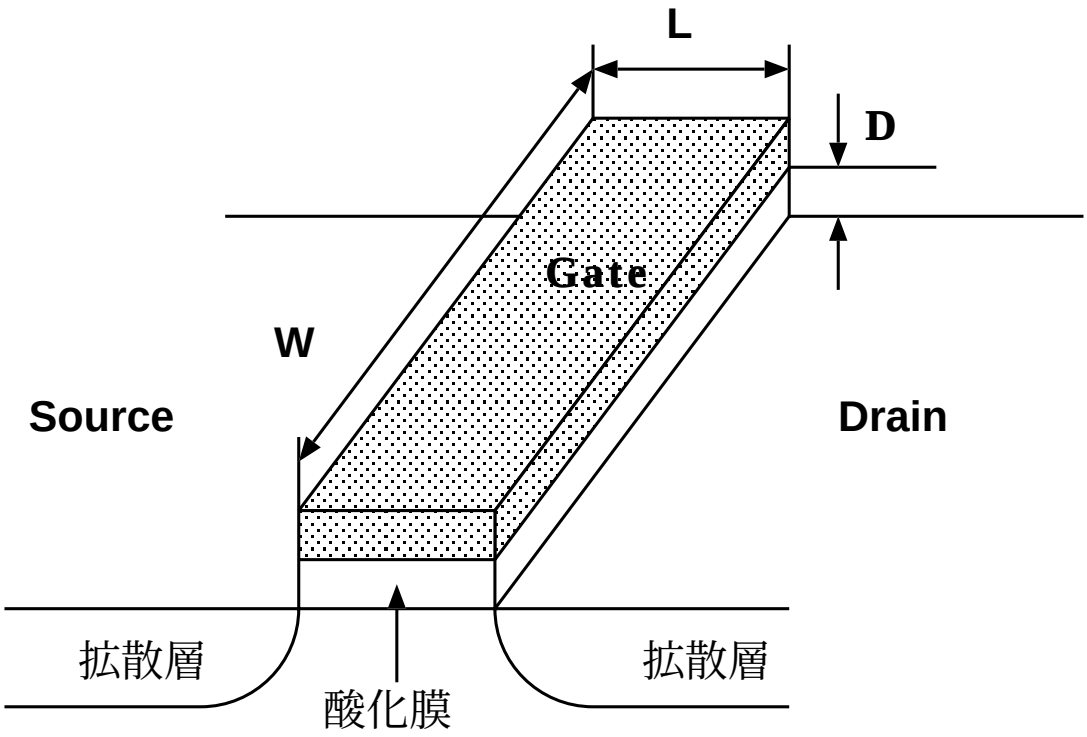

図 2.1: ゲートの諸寸法の定義

電界 Eは、ゲート長 L とドレインソース間電圧 V_{ds}を用いると次のようになる。

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

走行時間τは、電子の移動しなければならない距離 L と電荷の速度 vの比として次式で表される。

$$\tau = \frac{L}{v} = \frac{L}{\mu E} = \frac{L^2}{\mu V_{ds}} \tag{2.2}$$

ゲート容量 C_gは、ゲート長 L、ゲート幅 W、ゲート酸化膜の厚さ D、誘電率εとすると次式のように求まる。

$$C_g = \varepsilon \frac{WL}{D} \tag{2.3}$$

走行に関与する負電荷の総量 Q は、ゲートソース間電圧 V_{gs}としきい値電圧 V_{th}を用いて次式のようになる。

$$Q = -C_g(V_{gs} - V_{th}) = -\varepsilon \frac{WL}{D}(V_{gs} - V_{th})$$

ソース電極からドレイン電極に流入する電流の大きさ I_{ds}は、走行時間にチャネルに誘起した電荷量を電子が電極間を移動するのに必要な平均時間で割ったものに等しい。

$$I_{ds} = -I_{sd} = -\frac{dQ}{dt} = -\frac{Q}{\tau} = \frac{\mu\varepsilon W}{LD}(V_{gs} - V_{th})(V_{ds})$$

V_{ds}が小さい場合には、ドレイン電流 I_{ds}は、V_{ds}と実効ゲート電圧 $(V_{gs} - V_{th})$ に比例する。トランジスタのオン抵抗 Rは、V_{ds}が小さい場合、次のようになる。

$$R = \frac{V_{ds}}{I_{ds}} = \frac{L^2}{\mu C_g(V_{gs} - V_{th})} = \frac{DL}{\mu\varepsilon W(V_{gs} - V_{th})} \tag{2.4}$$

時定数 T_dはオン抵抗とゲート容量の積 RC_gであらわされる。

$$T_d = RC_g = \frac{L^2}{\mu(V_{gs} - V_{th})} \tag{2.5}$$

これが、配線遅延を考慮しない場合のゲート 1 段あたりの遅延時間となり、V_{th}が無視でき、V_{gs}と V_{ds}が等しい場合、走行時間τと等しくなる。

2.3 低消費電力のためのアプローチ

消費電力を下げるために、ゲート容量 C_gの減少を試み、ゲート幅 Wを小さくすることで実行する。この結果、動作周波数が落ちるが、これを補うためにウェーブパイプラインを導入する。

2.3.1 低消費電力化

チップ全体の電力 Pは、配線容量を C_lとすると、ゲートと配線での電力の総和で表される。

$$P = \sum_{k=1}^{n}(C_{g_k} + C_{l_k})V^2 f \tag{2.6}$$

(2.6) 式より、電圧、容量、周波数を下げれば電力が小さくなることがわかる。処理速度を落さないで電力を減少させるには、容量を小さくすることで可能になる。容量を小さくするためにゲート幅 Wを $1/\alpha$倍の大きさにする。ゲート幅を $1/\alpha$倍にしたときのトランジスタのオン抵抗 R_αとゲート容量 $C_{g\alpha}$は、(2.7),(2.8) 式のようになる。

$$R_\alpha = \alpha \times \frac{DL}{\mu\varepsilon W(V_{gs} - V_{th})} \tag{2.7}$$

$$C_{g\alpha} = \frac{1}{\alpha} \times \varepsilon\frac{WL}{D} \tag{2.8}$$

したがって、Wの大きさに比例して、ゲート容量は小さくなっているが、トランジスタのオン抵抗はWの大きさに反比例して大きくなることが分かる。このときの電力P_αおよび遅延T_αは、(2.9) 式および (2.10) 式のようになる。

$$P_\alpha = \sum_{k=1}^{n}(C_{l_k} + \frac{C_{g_k}}{\alpha})V^2 f \tag{2.9}$$

$$T_\alpha = \alpha R C_l + R C_g \tag{2.10}$$

(2.9) 式より、配線容量C_lの大きさは変わらないので、電力の低下が小さくなる。トランジスタのオン抵抗Rとゲート容量C_gの積は一定のままだが、配線容量は変わらず、トランジスタのオン抵抗が大きくなるため、配線容量とトランジスタのオン抵抗の積はゲート幅Wに応じてα倍になり、遅延が増大する。

2.3.2 動作周波数の補償

トランジスタのオン抵抗と配線容量の積が大きくなることにより、遅延が増大する。これを克服するために、ウェーブパイプラインを導入する。ウェーブパイプラインは、通常のパイプラインとは異なり、パイプラインステージが完全に処理を終える前に次の処理を開始する機構である。図 2.2(A) は、通常のパイプラインのステージ内でのデータの流れを示している。また、図 2.2(B) は、ウェーブパイプラインのデータの流れを示している。図 2.2で網のかかった部分にデータがながれているが、図 2.2(B) は、(A) とは異なり複数のデータがパイプライン中に流れているのが分かる。ウェーブパイプラインは、最大遅延により周波数が決まる通常のパイプラインと異なり、最大遅延$T_{d_{max}}$と最小遅延$T_{d_{min}}$の差の最大値で周波数fが決定される。

$$f = \frac{1}{max(T_{d_{max}} - T_{d_{min}})} \tag{2.11}$$

この式より、遅延差を小さくすることで高速に動作させることが可能であることが分かる。最大遅延を固定として、最小遅延を大きくしていくことで遅延差を小さくする。最大遅延$T_{d_{max}}$が大きいので、周波数fをもとの大きい値にするために、最小遅延$T_{d_{min}}$を大きくする。この遅延差の縮小にバッファを挿入するが、バッファ挿入による電力の増加分がゲート容量の減少分の消費電力よりも小さければ、電力の減少が可能である。

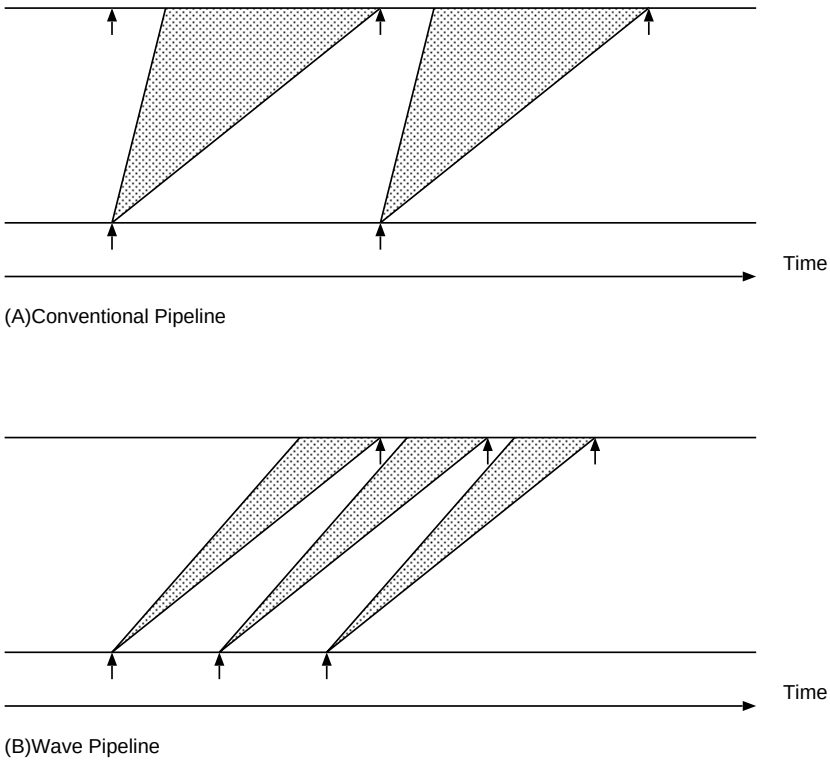

図 2.2: パイプラインとウェーブパイプライン

2.3.3 電力減少分と追加可能なバッファ容量

バッファ挿入後の電力 P'は、(以降では、容量は、全体の容量を表すこととして、$\Sigma C_i = C$ とする)

$$P' = (C_l + \frac{C_g}{\alpha})V^2 f + (C'_l + \frac{C'_g}{\alpha})V^2 f \tag{2.12}$$

となる。電力が減少するための条件は、元の電力より小さくなることなので、(2.6) 式と (2.12) 式より次の条件が成り立つ。

$$\begin{aligned} P' &< P \\ (C_l + \frac{C_g}{\alpha}) + (C'_l + \frac{C'_g}{\alpha}) &< (C_l + C_g) \end{aligned} \tag{2.13}$$

(2.13) 式を整理すると次のようになる。

$$C'_l + \frac{C'_g}{\alpha} < C_g(1 - \frac{1}{\alpha})$$

ここで、配線容量 C_lがゲート容量の $1/\beta$倍とすると、C_gおよび C'_gの関係は、次のようになる。

$$\frac{C'_g}{C_g} < \frac{\alpha - 1}{\frac{\alpha}{\beta} + 1} \tag{2.14}$$

これが、追加できるバッファ容量の割合を示している。

ここで、電力を半分に削減することを目標にすると、(2.13) 式は、以下のようになる。

$$(C_l + \frac{C_g}{\alpha}) + (C'_l + \frac{C'_g}{\alpha}) = \frac{1}{2}(C_l + C_g)$$

よって、このときの条件は、次のようになる。

$$\frac{C'_g}{\alpha} + C'_l = (\frac{1}{2} - \frac{1}{\alpha})C_g - \frac{1}{2}C_l \tag{2.15}$$

また、今回は、トランジスタの大きさを 1/5 にしたので、条件式を$\alpha = 5$ として整理してみる。

$$C'_l + \frac{C'_g}{5} = \frac{3}{10}C_g - \frac{1}{2}C_l \tag{2.16}$$

したがって、もとの回路のゲート容量の 30%から配線容量の半分の差までが追加できる素子の容量となる。

2.4 遅延モデルと電力モデル

電力と動作周波数を求めるためのモデルをトランジスタのオン抵抗 R_{ON}、配線抵抗 R_l、ゲート容量 C_g、接合容量 C_dおよび配線容量 C_lを用いて求める。

2.4.1 抵抗および容量の算出

トランジスタのオン抵抗、ゲート容量および接合容量は Hspice シミュレーションにより求める。

配線抵抗および配線容量は以下のように求める。アルミニウムの抵抗率 $3 \times 10^{-2}[\mu\Omega \cdot m]$、ポリシリコンの抵抗率 $10[\mu\Omega \cdot m]$ とする。

配線抵抗は、抵抗率を$\rho[\Omega \cdot m]$、シート抵抗$\rho_s[\Omega]$ により求まる。シート抵抗$\rho_s[\Omega]$ は、配線の長さ $l[m]$、幅 $w[m]$、厚さ $t[m]$ とすると以下のように求まる。

$$\rho_s = \frac{\rho}{t}[\Omega]$$

このシート抵抗値ρ_sより、配線抵抗 R_lが次式で求まる。

$$R_l = \rho_s \frac{l}{w}[\Omega] \tag{2.17}$$

配線容量は、単位面積あたりの酸化膜容量から求める。単位面積あたりの酸化膜容量 C_o は酸化膜厚を $d[m]$ とすると以下の式で求められる。ただし、SiO_2 の比誘電率$\varepsilon_{ox} = 3.82$, 真空の誘電率$\varepsilon_0 = 8.85 \times 10^{-12}[F/m]$ とする。

$$C_o = \frac{\varepsilon_{ox}\varepsilon_0}{d}[F/m^2]$$

この単位面積あたりの酸化膜容量 C_oを用いて、配線容量 C_lは次式により求められる。

$$C_l = C_o \times l \times w[F] \tag{2.18}$$

2.4.2 遅延モデル

トランジスタの等価回路を抵抗と容量の形で表すと図 2.3(A) のようになる。図中、破線で囲まれた部分は、配線であることを示している。ある程度以上長い配線になると、集中定数回路での近似では、誤差が大きくなるので、図 2.3(A) の様に、配線の長さに応じて、π型の分布定数回路での抵抗とコンデンサの段数を決定した。

図 2.3(A) 回路では、遅延を求めるのが複雑であるので、図 2.3(B) のような簡略化した回路を用いる。図 2.3(B) の回路では、以下の式になる。

$$T_d = (R_{ON} + R_l) \times (C_d + C_l + C_g) \tag{2.19}$$

これらの 2 つの回路の誤差を評価し、その誤差の割合を実際のシミュレーションにより求まった周波数にかければ、ほぼ等しい値が求まる。このときの遅延の式は (2.20) 式で近似する。

$$T_d = \alpha \times (R_{ON} + R_l) \times (C_d + C_l + C_g) \tag{2.20}$$

2.4.3 電力モデル

前に述べたように、電力は、容量 C、電圧の 2 乗 V^2、周波数 fに比例する。また、入力の信号のパターンに電力が依存するため、常に同じ電力が消費されている訳ではない。そのため、評価には、時間的な影響を考慮する必要がある。また、一般的には、電力の測定は、あらかじめ想定された入力パターンに対して行なわれ、精密な評価というのは行なわれていない。

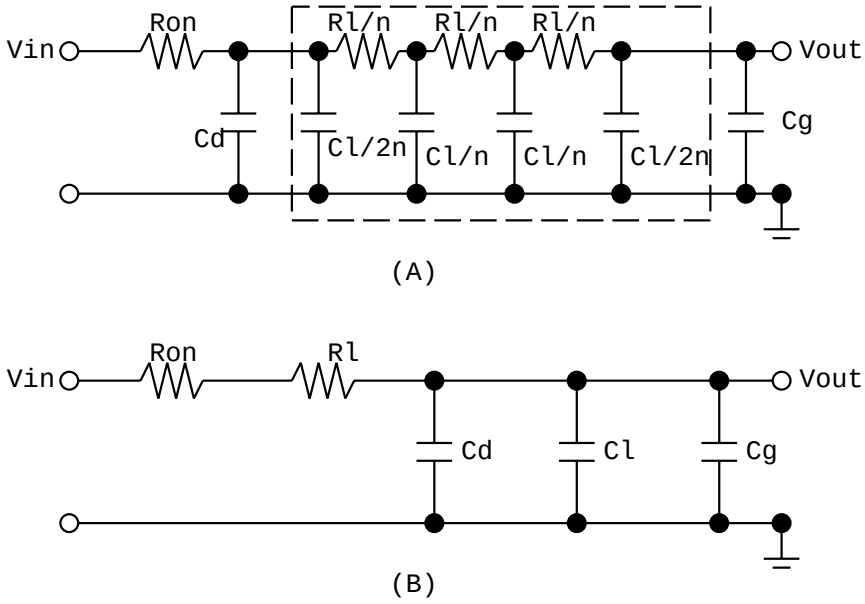

図 2.3: 遅延のための回路モデル

本研究では、電力の測定方法として以下のような方法をとることにする。ある時間でのゲートの値の変化が起こるかどうかに注目し、変化のあるノードの数を数え上げ、そして、次の時刻でも値の変化した数を数え集計する。これを繰り返して、シミュレーションをする時間にわたって和をとる。

第 3 章

設計

シミュレーションの対象になるプロセッサの設計とそのためのパラメタの算出を行なった。

3.1 設計の手順

以下のような手順で設計を行なった。それぞれの手順ごとに、以降、順を追って説明する。

1. 設計の対象となるプロセスルールを $0.1\mu m$ と決め、このプロセスルールでのパラメタの算出と、トランジスタと論理回路の spice シミュレーションと遅延の評価を行なった。
2. SPICE シミュレーションの結果に基づき CAD 用のセルライブラリを設計した。
3. シミュレーション用の回路を決定した。回路を CAD システム PARTHENON の記述言語 sfl により記述し、論理合成によりネットリストを抽出する。
4. 出力されたネットリストをもとに配置を行なう。また、同時に遅延均衡のためのバッファスロットも確保しておく。
5. ネットリストと配置情報より論理シミュレーションを行ない、電力を算出する。

3.2 HSPICE シミュレーション

CMOS トランジスタの動作速度と論理回路の動作速度を求めるために hspice によるシミュレーションを行なった。シミュレーションに必要なパラメタは、$4[\mu m]$ のものは文献

[6]、0.5[μm] のものは文献 [10] よりもとめた。以下に元にしたパラメタと、0.1[μm] プロセス用に算出したパラメタを示す。

表 3.1: MOSFET モデル

	4μm	0.5μm	0.10μm
$Nsub[cm^{-3}]$		1E17	5E17
Cgso[F/m]		169p	169p
Cgdo[F/m]		169p	169p
$Cj[F/m^2]$	1E-4		1E-4
Cjsw(nMOS)[F/m]	9E-10		0.225E- 10
Cjsw(pMOS)[F/m]	8E-10		0.2E-10
Ld[m]		5E-8	1E-8
Pb[V]		0.75	0.75
Tox[m]		10n	2n
$Uo(nMOS)[cm^2/V \cdot s]$		1350	1350
$Uo(pMOS)[cm^2/V \cdot s]$		480	480
$Vmax[m/s]$			1E5
$Xj[m]$	9〜8E-6	1.125〜1E-6	0.225〜0.2E-6

$Nsub[cm^{-3}]$ 基板不純物濃度

$Cgso[F/m]$ 単位チャンネル幅あたりのゲート・ソースオーバーラップ容量

$Cgdo[F/m]$ 単位チャンネル幅あたりのゲート・ドレインオーバーラップ容量

$Cj[F/m^2]$ 単位接合面積あたりのゼロバイアスバルク接合底面の容量

$Cjsw[F/m]$ 単位接合周囲長あたりのゼロバイアスバルク接合側面の容量

$Ld[m]$ ゲートと拡散層とのオーバラップ長

$Pb[V]$ バルク接合電位

$Tox[m]$ ゲート酸化膜厚

$Uo[cm^2/V \cdot s]$ 表面移動度

$Vmax[m/s]$ キャリアの最大ドリフト速度

$Xj[m]$ ドレイン・ソース拡散深さ

3.2.1 MOSトランジスタの電流電圧特性

nMOS および pMOS トランジスタの電流電圧特性を 3.2節のパラメタを用いて HSPICE シミュレーションにより求めた。トランジスタのゲート長およびゲート幅はともに $0.1[\mu m]$ とした。図 3.1の V_{gs}は、上から 1.0[V],0.5[V] で、図 3.2の V_{ds}は、1.0[V] である。

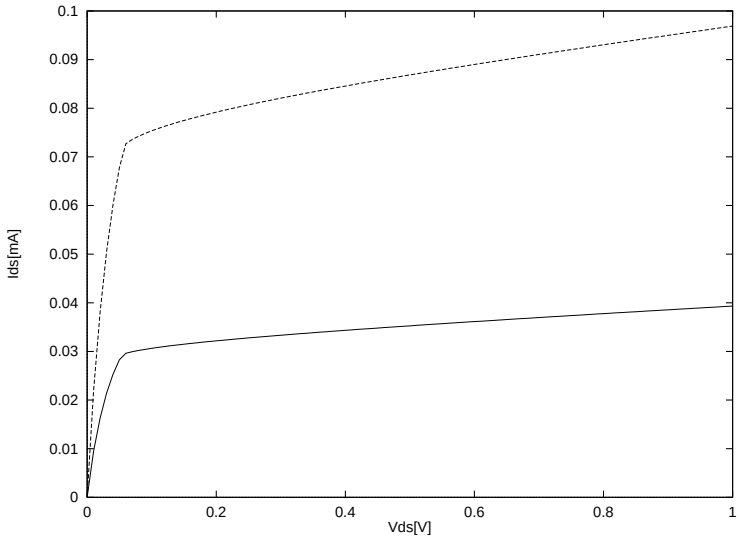

図 3.1: nMOS トランジスタの Ids–Vds 特性

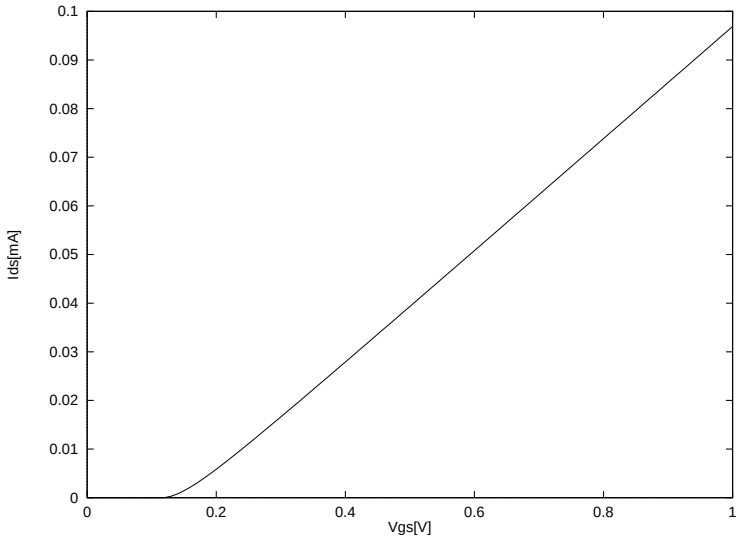

図 3.2: nMOS トランジスタの Ids–Vgs 特性

図 3.3の V_{gd}は、上から$-1.0[V]$,$-0.5[V]$、図 3.4の V_{ds}は、$-1.0[V]$ である。

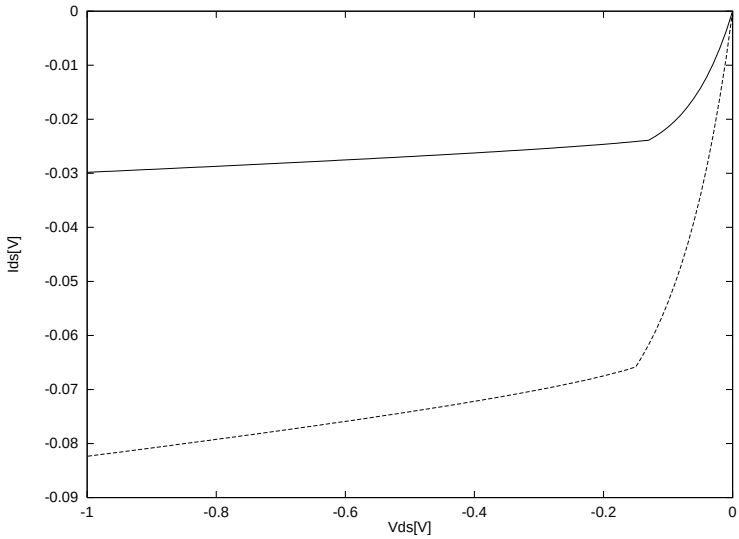

図 3.3: pMOS トランジスタの Ids–Vds 特性

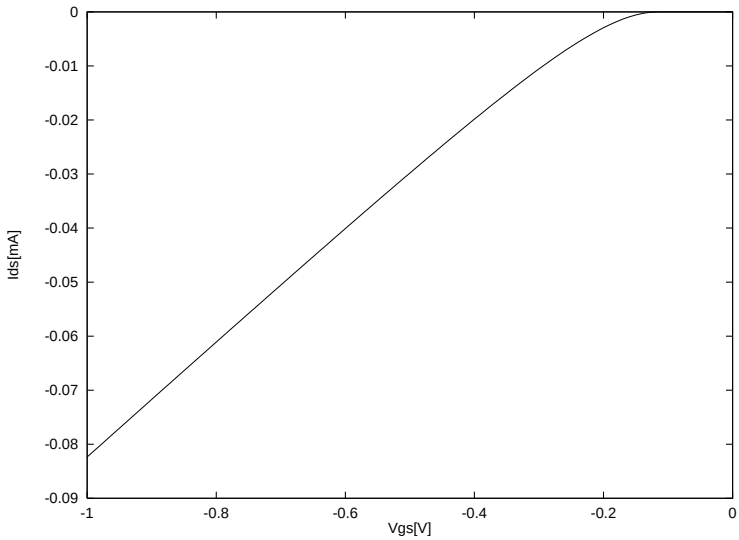

図 3.4: pMOS トランジスタの Ids–Vgs 特性

3.3 素子モデルおよび配線モデルのパラメータ

評価用のプロセッサのための素子モデルおよび配線モデルをを SPICE シミュレーションの結果にに基づき決定した。素子モデルは、セルライブラリとして parthenon に与える必要がある。

3.3.1 素子モデル

表 3.2: $0.1\mu m$ プロセスルールの素子モデル

ゲート	オン抵抗 $[K\Omega]$		入力端子容量 [fF]	拡散容量 [fF]	面積 $[\mu m^2]$
	nMOS	pMOS			
inv	21.9	26.8	0.156	0.019	0.84
buffer	43.8	53.6	0.156	0.038	1.68
nand	23.9	29.2	0.14	0.0206	0.99
nor	24.2	29.6	0.14	0.021	0.99
enor	24.2	29.6	0.24	0.0206	2.61
eor	24.2	29.6	0.24	0.0206	2.61

3.3.2 配線モデル

配線モデルは、(2.17),(2.18) 式より配線抵抗および配線容量を決定する。配線のパラメータを以下のように決定し、配線抵抗および配線容量を求める。

グリッド $1[grid] = 0.1[\mu m]$
配線幅 $0.1[\mu m]$
酸化膜容量 $169[\mu F/m^2]$

$$R_l = 0.0003[k\Omega/grid] \tag{3.1}$$
$$C_l = 0.00169[fF/grid] \tag{3.2}$$

3.3.3 配線遅延の評価

前節で決定した配線のパラメタを利用して、配線遅延の評価を行なう。このとき、配線のモデルは、第 2 章の 2 つのものを使い誤差の評価を行なう。図 3.5,3.7では、実線が入力波形、破線が多段のπ型回路のシミュレーション結果、そして、細かい破線が近似した回路でのシミュレーション結果である。

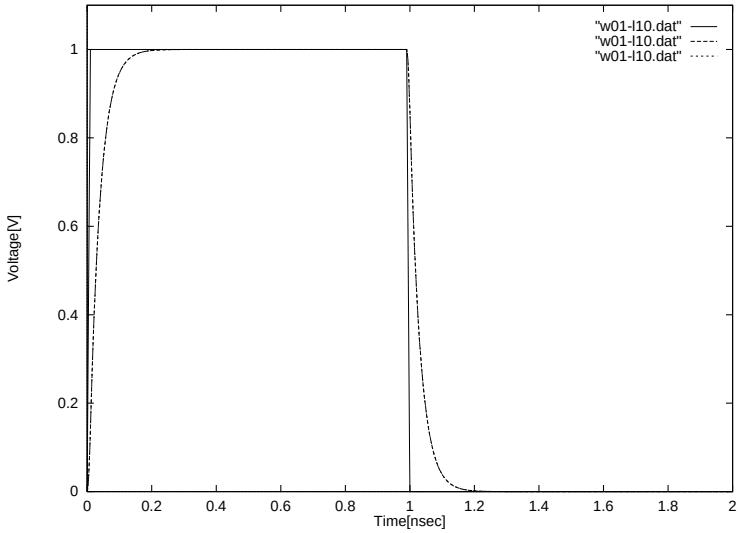

図 3.5: 配線長 $10[\mu m]$ のときの応答

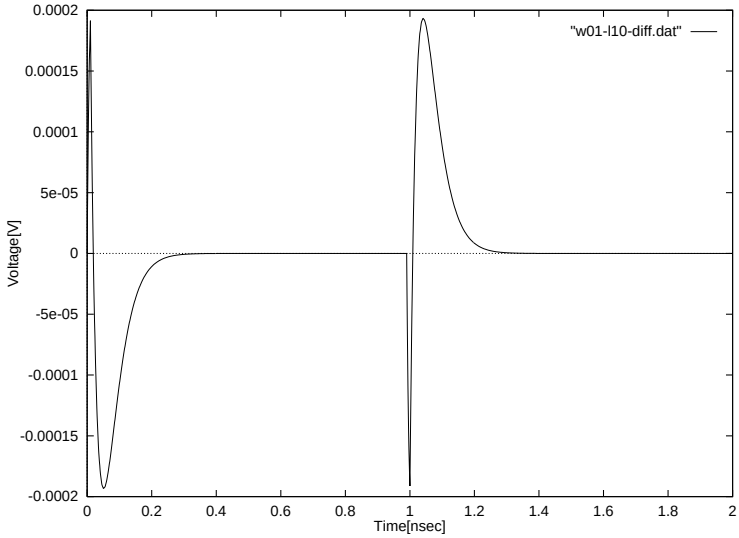

図 3.6: 配線長 $10[\mu m]$ のときの応答の誤差

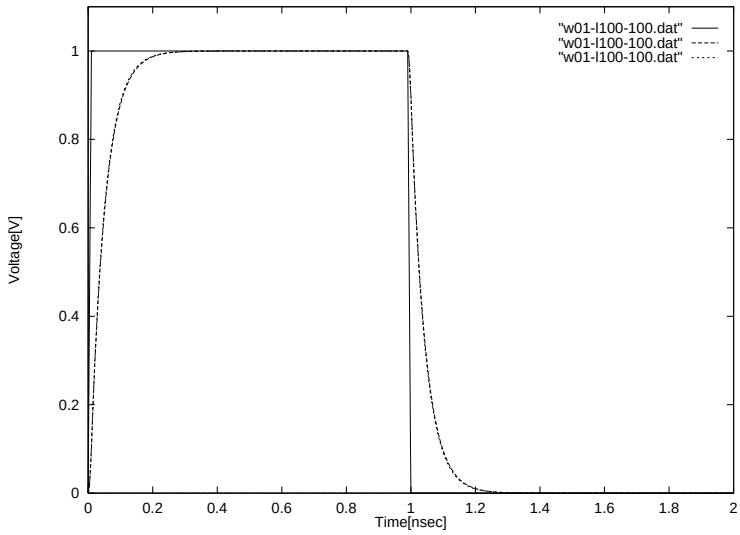

図 3.7: 配線長 $100[\mu m]$ のときの応答

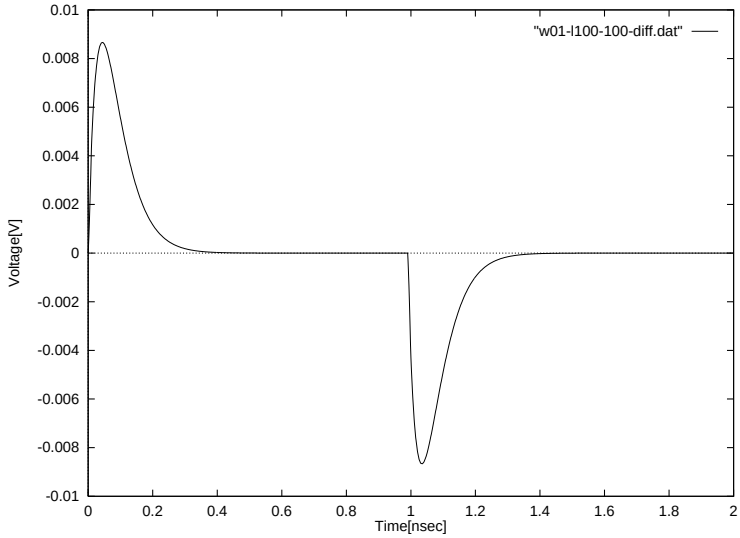

図 3.8: 配線長 $100[\mu m]$ のときの応答の誤差

これらの結果より、配線長が $100[\mu m]$ までの範囲では、最大誤差が 0.01 なので、配線遅延の影響はほとんどないとみなしてよい。

3.4　シミュレーション用のプロセッサの仕様

シミュレーションには、以下のような仕様のパイプラインプロセッサを想定している。

- 16 ビットのアドレスバスおよびデータバス
- メモリを命令とデータに分けない非ハーバードアーキテクチャ
- 5 段のパイプラインとする。構成は IF,ID,EXE,MEM,WB である。
- ロードストア形式の RISC プロセッサ
- ハザード検出機能はない
- 例外処理機能はない

このプロセッサを CAD である PARTHENON で使用される記述言語 SFL により記述し、論理合成を行ない素子の接続情報であるネットリストを得る。

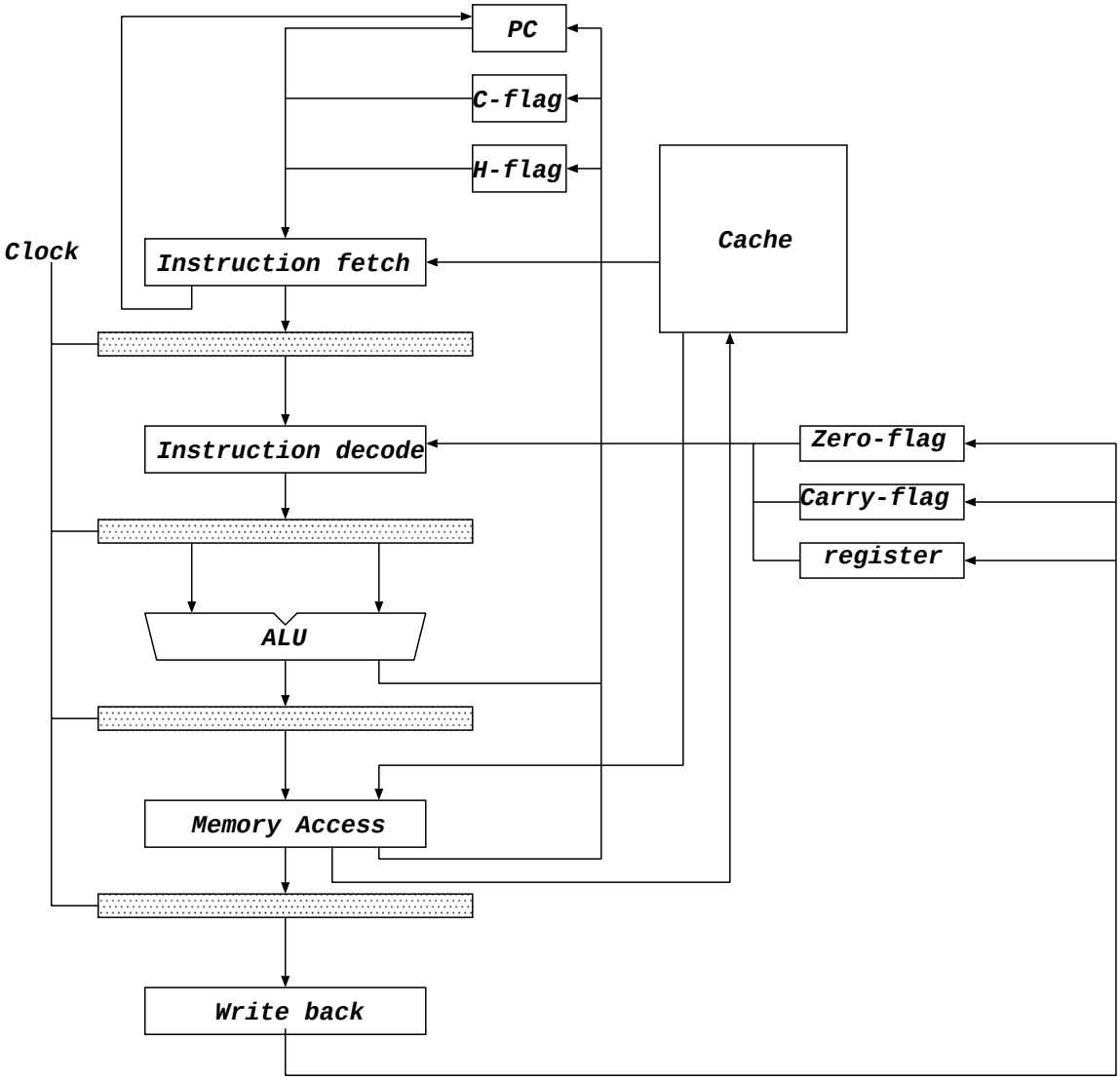

図 3.9: シミュレーションの対象にしたプロセッサの構成

3.5 配置

ネットリストを入力としてバッファを挿入するためのスロットをあらかじめ設定して、配置を行なう。配置は、ペア交換法を利用して最適化しながら行なう。ペア交換法は、ランダムに 2 つのセルを取り出して場所を交換し、配線が短くなるか判定し、短くなるなる場合のみ交換する方法である。本研究では、2 つのセルを取り出したあと、配線が短くならない場合は一方を捨てて、別のセルをランダムに取り出す。この結果、一つのセルに対し 10,000 回交換を繰り返しても配線が短くならない場合に終了するようにしてある。

3.6 遅延差短縮について

本節では遅延差短縮のアルゴリズムについて簡単に述べる。まず、出力端子からつながっているすべての入力端子への探索を行ない、最大遅延と最小遅延を求める。遅延差の情報より遅延素子を挿入して遅延差の短縮をはかる。このとき、素子を一つ入れると配置情報が変わるので、遅延素子を入れる度に最大遅延と最小遅延を算出する必要がある。

第 4 章

評価

本章では、3 章までで設計したプロセッサを用いて電力の測定及び評価を行なう。評価に関しては、厳密な電力の評価方法が一般には存在しないので、評価方法に関しても議論する。

4.1 電力の評価法の検討

既存の電力の評価法は、正確な電力を反映していない。したがって、電力の評価法の検討を行なう必要がある。Parthenon を用いてプロセッサを設計し、ネットリストを生成する。ネットリストから配置、配線のプログラムによりレイアウトと配線を行なう。このとき、遅延バランスを行なうことを考慮してあらかじめ、バッファ挿入のための空きスロットを確保しておく。遅延バランスをする場合は、その空きスロットに対して遅延素子を挿入することで実行する。これらの配置情報から、論理シミュレータにより、全体の遅延と電力を算出する。

4.2 シミュレーションに用いた回路

第 3 章で示したプロセッサの EXE ステージを対象とした。EXE ステージをシミュレーションの対象とした理由は、ウエーブパイプラインによる高速化が適しているためである。ウエーブパイプラインの性質から、高速化に適した条件がある。メモリへのアクセスがないことである。この条件に適合するステージは、EXE ステージである。したがって、EXE ステージを対象に選んだ。

parthenon の論理合成の結果、EXE ステージは、キャリールックアヘッド型の加算器と

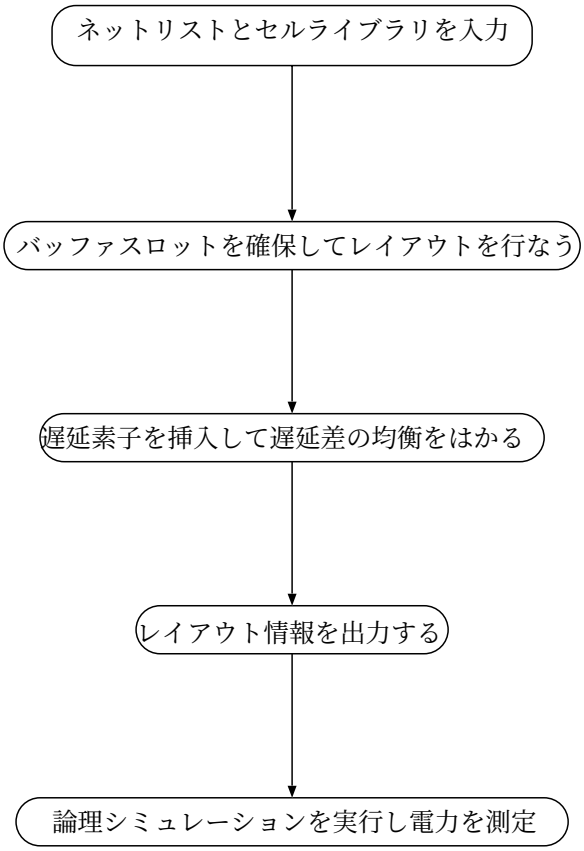
ネットリストとセルライブラリを入力
バッファスロットを確保してレイアウトを行なう
遅延素子を挿入して遅延差の均衡をはかる
レイアウト情報を出力する
論理シミュレーションを実行し電力を測定

図 4.1: 電力算出の手順

セレクタが 2 つの計 3 個のサブモジュールで構成されていた。このうち、加算器についてシミュレーションをすることとした。簡単のためキャリールックアヘッド型の 4 ビット加算器についてシミュレーションをした。

4.3 電力の算出

PARTHENON により論理合成により出力されるネットリストをもとに、配置を行ない、遅延素子を挿入して、遅延差の短縮を行なった。遅延差を短縮した後、論理シミュレーションにより電力を測定した。遅延差の短縮には、ネットリストを参照して出力側から見たときにどのくらいの遅延差があるか見積もる必要がある。ゲート容量、拡散容量、配線容量の和をそれぞれ求める。まず、すべての容量の和を求める。

表 4.1: 容量の和

W の大きさ $[\mu m]$	ゲート容量 $[fF]$	拡散容量 $[fF]$	配線容量 $[fF]$
0.1	9.82	2.71	13.5
0.5	49.1	11.8	14.9

これより、配線容量はほとんど変化していないことが分かる。

表 4.2: 面積、最大遅延および素子数

W の大きさ $[\mu m]$	ゲート数	面積 $[\mu m^2]$	遅延時間 [psec]
0.5	62	129.85	8
0.1	62	72.1	32.5

また、$W = 0.1[\mu m]$ のときの遅延素子数と遅延差を求めた。

表 4.3: $W = 0.1[\mu m]$ のときの遅延差短縮後の遅延素子数および遅延差

遅延素子数	遅延差 $[psec]$	面積 $[\mu m^2]$
20	8	92.3

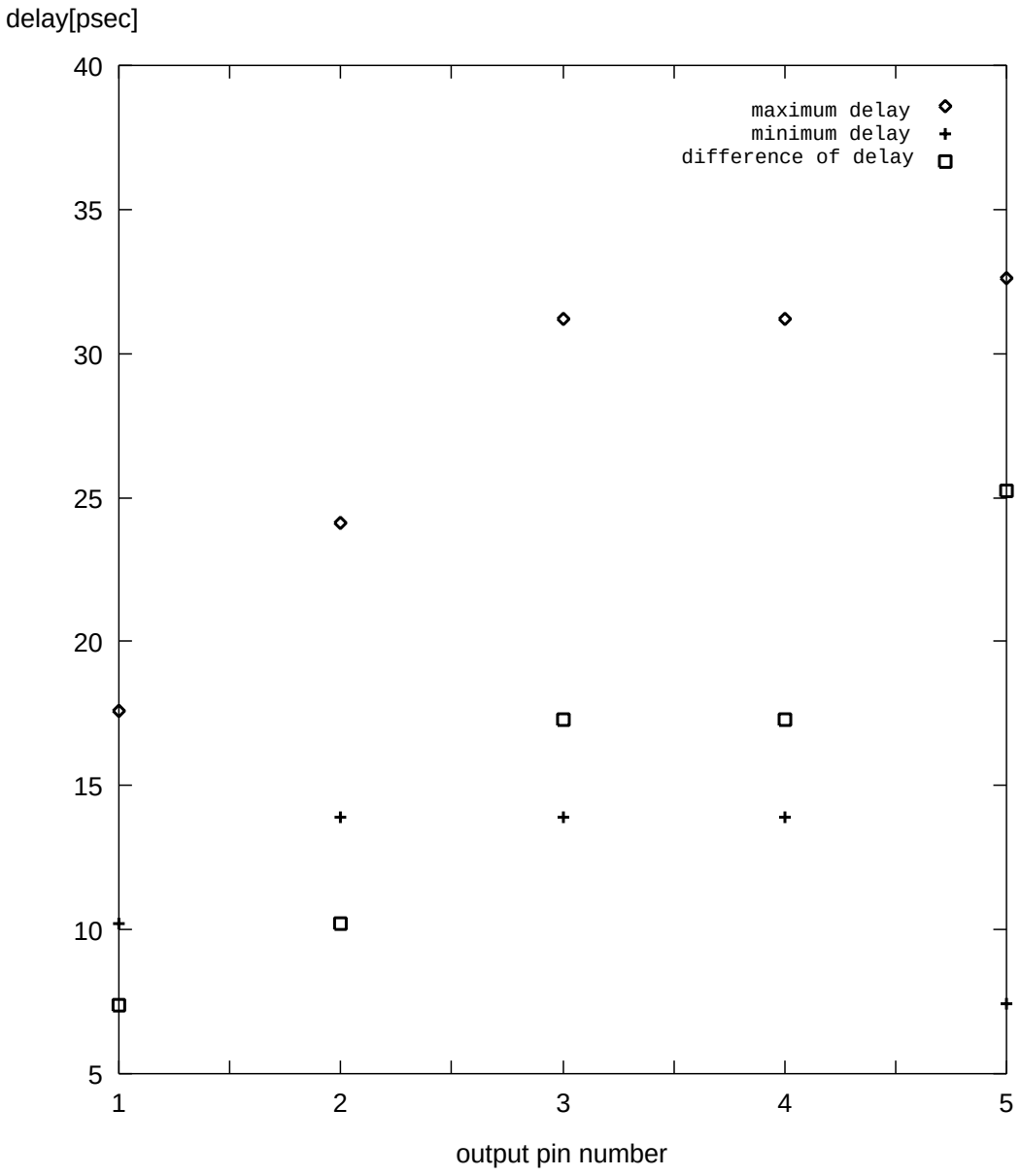

図 4.2: 4 ビットキャリールックアヘッド加算器の遅延差

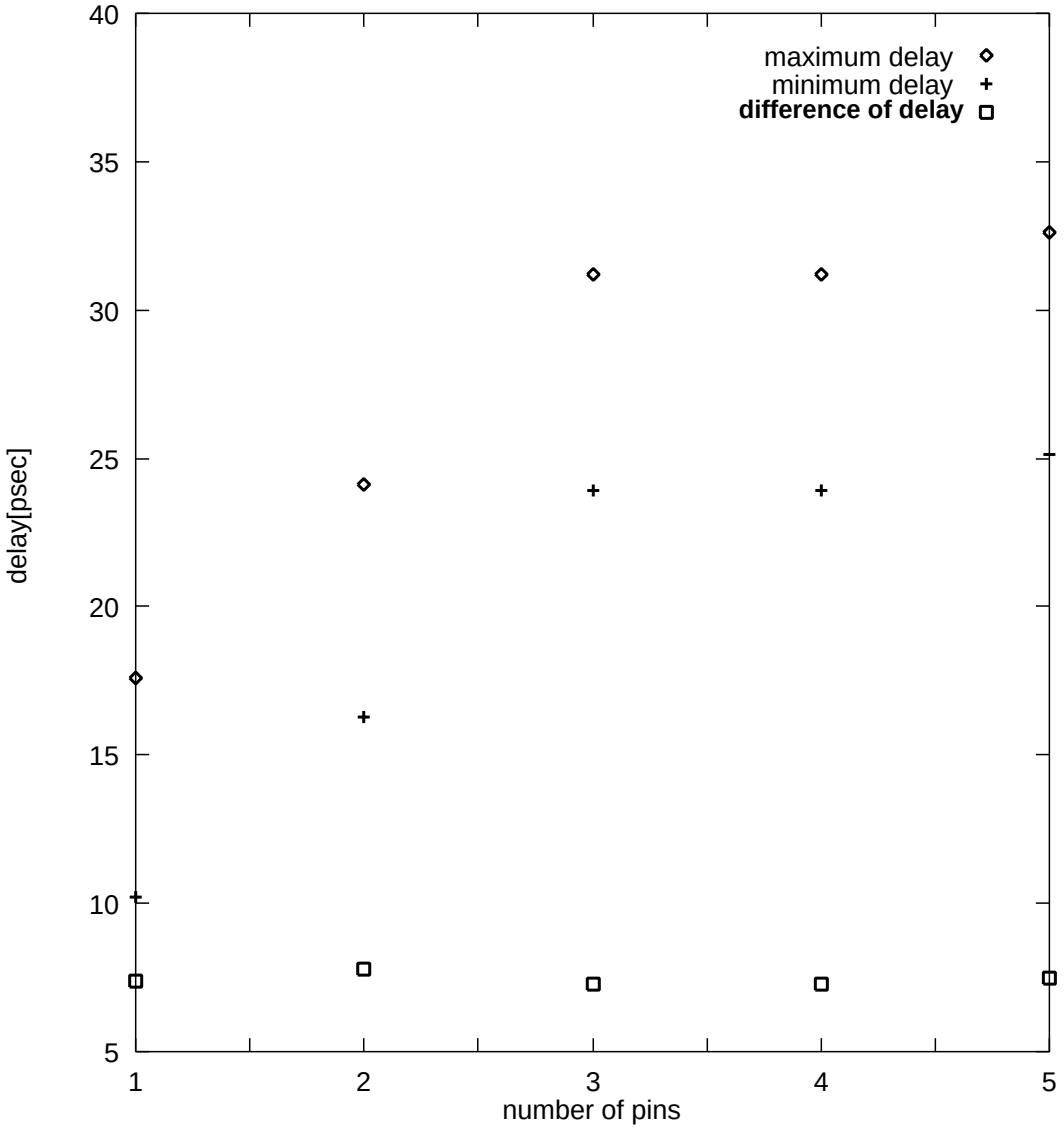

図 4.3: 4 ビットキャリールックアヘッド加算器遅延差短縮後の遅延差

最大遅延が大きくなっているが、W を 1/5 にしているのに 5 倍になっていない。これは、配線が最大で $20\mu m$ 程度であったため、配線遅延があまり大きくならずにすんだためである。

4.4 電力の評価

前節の結果より、プロセッサの電力の評価が可能であるが、さらに、一般的なライブラリを利用したときの電力との比較を行ない、最終的な評価を行なう。$W = 0.5\mu m$ のトランジスタで設計した回路および $W = 0.1\mu m$ のトランジスタで設計した回路の 2 つを用意した。$W = 0.1um$ の回路では、遅延差短縮を行ない、遅延素子を挿入した後の回路に対してシミュレーションを行なった。

表 4.4: 電力の比較 (1 サイクル当たり)

W の大きさ $[\mu m]$	電力 P[pW]	全体が動作している時の電力 P_0[pW]	P/P_0
0.1	9.68	13.8	0.7
0.5	18.3	40.9	0.45

また、配線容量による電力は配線の断面の形状は、トランジスタの大きさに関係なく一定としたため、ほとんど変化していない。結果として、W を $0.1\mu m$ にした場合、電力は、$W = 0.5\mu m$ のときのほぼ半分になっている。

第 5 章

考察

トランジスタの大きさ W を $0.5\mu m$ と $0.1\mu m$ にしたときの 4 ビットキャリールックアヘッド型の加算器で電力の測定を行なった。$0.5\mu m$ で設計した回路は、通常のパイプライン動作に対応した回路とし、$0.1\mu m$ で設計した回路では、ウーブパイプラインでの動作を考慮し、最大遅延と最小遅延の遅延差を小さくするように最適化を行なった。その結果、電力は $0.5\mu m$ で設計した回路のほぼ半分となった。これは、遅延素子であるバッファ挿入によるものと考えられる。

また、全容量が一度にオンオフをするという仮定のもとで、電力がどの程度になるかを調べた。その結果、非常に大きな電力となることが分かった。

5.1 遅延の評価について

本論文において、容量の減少による電力の削減と、同時に、高速化を実現するための方法を見い出すことを目的とした。第 2 章において、電力が容量に依存して減少させることが可能なことを示した。その結果、遅延の増大という問題を引き起こすことになった。しかし、設計の段階において、配線の形状と配線抵抗の評価を行なったところ、配線の形状を工夫して配線抵抗を小さくし、配線容量を大きくすることで、遅延を抑えることができた。配線容量と配線抵抗を最長の配線に対して求めた。

$$R = 0.0003 \times 230 = 0.069[\Omega] \tag{5.1}$$

$$C = 0.00169 \times 230 = 0.387[fF] \tag{5.2}$$

ゲートのオン抵抗が $20[\mathrm{k}\Omega]$ 以上であることを考えると、100 倍以上開きがあるので、遅延には、ほとんど影響を与えていないことが分かる。このことより、配線抵抗は遅延の算出

には無視しても差し支えないことがわかる。

5.2 容量の分布

コンデンサ容量がどの程度であるかということが、重要である。実際、理想的に全容量を求めてみたが、非常に大きな値であった。また、ゲート容量に対して、拡散容量はその15%であるが、無視できない量であった。一方、配線容量は、配線が長い場合は、ほぼゲート容量と同じ程度になるので、こちらも無視できる値ではなかった。

5.3 電力と容量の関係

電力は、定義に従うと、容量に比例している。したがって、容量がどこに充電されているかを調べる必要がある。そのため、本研究では、論理シミュレーションと同時に電力を算出する方法をとった。電力が回路の入力に依存する。本来、電力の評価は、設計時には、途中で算出するのは困難にちかい。また、論理レベル設計から、電力を知る方法も通常はない。そのため、論理設計と同時に電力を評価できるのは有効な方法である。

電力が半分になったのは、容量が全体で大きく減らないからである。特に配線容量は、ほとんど減らなかった。そのため、ゲート容量および拡散容量の減少が電力の減少に影響している。

電力が半分になる条件式より追加可能な素子の容量が全ゲート容量の 30%から、配線容量の半分の値の差で求まる。したがって、今回の 4 ビットキャリールックアヘッド型加算器では、全素子数 66 に対して、挿入したバッファ数が 20 個であったため、この条件をほぼ満たしているということがいえる。

本手法では、全体が動作するときの電力 P_0とシミュレーションにより求められた電力 P の比を求めた。その結果、$W = 0.5[\mu m]$ のときは、0.45 であったが、$W = 0.1[\mu m]$ と W を小さくして、ウエーブパイプラインを導入して動作させたときは 0.7 になった。この設計方法では、電力は小さくなっているが、電力の比が高いため、動作している時間が大きいことが分かる。したがって、今回の回路を用いて実際にプロセッサの実装を行なうときには、熱の問題が出てくる。

第 6 章

まとめ

第 1 章では、本論文の構成を説明した。続く、第 2 章では、電力の式から容量を減らすことで低消費電力が可能であることを示し、理論的なことがらを背景として、容量を小さくした場合の影響を調べた。第 3 章では、シミュレーションに必要なパラメータの算出、素子、配線モデルを決定し、シミュレーション対象となるプロセッサを設計した。第 4 章では提案手法による電力の削減が可能かどうかのシミュレーションを行ない、評価を行なった。第 5 章では、考察を行ない、本手法が有効であることを示した。

謝辞

本研究を進めるにあたり、終始熱心な御指導を頂きました日比野靖教授に心から感謝いたします。

また、適切な御助言をして頂きました本学の堀口進教授、篠田陽一助教授、宮崎純助手に深く感謝致します。

さらに、プロセッサに関する基本的な助言を頂いた本研究室 OB の永田真也君、永谷充孝君に感謝いたします。その他、貴重な御意見を頂きました日比野研究室の皆様に厚く御礼申し上げます。

参考文献

[1] Wayne P.Burleson, “Wave-Pipelining: A Tutorial and Research Servey”, IEEE Transaction on VLSI System,1998

[2] Anantha P.Chandrakasan,Samuel Sheng,Robert W. Brodersen, “Low-Power CMOS Digital Design”, IEEE journal of Solid State Circuit

[3] 池田吉朗, “ウエ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,Feb.1999

[4] 鵜飼和歳, “パイプライン化によるキャッシュの高周波数動作の可能性に関する研究” 北陸先端科学技術大学院大学修士論文,Feb.1999

[5] 永谷充孝, “ウエーブパイプラインを用いたプロセッサの設計に関する研究” 北陸先端科学技術大学院大学修士論文,Feb.2000

[6] 榎本忠儀著, “CMOS 集積回路-入門から実用まで-” 培風館,1996

[7] NEC, http://www.nec.co.jp/japanese/today/newsrel/0002/0702.html,2000

[8] TRANSMETA http://www.transmeta.com

[9] Intel http://www.intel.com/developer/

[10] Neil H.E. Weste & Kamran Eshraghain 著, 富沢孝, 松山泰男監訳 “CMOS VLSI 設計の原理 システムの視点から” 丸善,1988

[11] C. ミード,L. コンウェイ著, 菅野卓雄, 榊裕之監訳, “超 LSI システム入門” 培風館,1981

[12] 菅野卓雄監修, 飯塚哲哉編, “CMOS 超 LSI の設計” 培風館,1989

[13] M. アナラトーネ著, 平野浩太郎, 冨田昌宏訳 “ディジタル CMOS の回路設計” コロナ社,1992

[14] 菅野卓雄監修, 電子情報通信学会編, “ULSI 設計技術” 電子情報通信学会,1993

[15] 松山泰男, 富沢孝著, “VLSI 設計入門” 共立出版,1983

[16] 西久保靖彦著, “ASIC 回路設計シミュレータ SPICE 入門” 日本工業技術センター,1988

[17] Avant! corporation, “Star-Hspice User's Manual - Volume 1,2,3”, 1996