修 士 論 文

ウェーブパイプラインを適用した高性能DSPに関する研究

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

宇山 幸平

2006年3月

修 士 論 文

ウェーブパイプラインを適用した高性能DSP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 教授
審査委員　田中清史 助教授
審査委員　井口寧 助教授

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

410018 宇山 幸平

提出年月: 2006 年 2 月

概 要

近年、DSP の高性能化が進み高スループット化が要求されている。DSP は、積和演算を高速で処理するための乗算器を持っている。乗算器後段に累算加算器を置き、積和演算器として用いられている。

本研究の目的は、ウェーブパイプライン方式を用いた積和演算器の高スループット化である。ウェーブパイプライン化を簡単に行うためには、簡単な回路構成で積和演算器を設計する必要がある。本研究では、規則的な配列で積和演算器を構成し、ウェーブパイプライン化を行い、回路シミュレータ HSpice を使って配線遅延を含めたトランジスタレベルでの評価、検討を行う。

目 次

図 目 次

表 目 次

第1章　序論

1.1　研究の背景

近年、DSP の応用分野が増えている。次世代DSP は、ディジタル家電やモバイル端末といったマルチメディア装置の処理に対応できる高スループット化が要求されている。DSP は、積和演算を高速で処理するための乗算器を持っている。乗算器後段に累算加算器を置き、積和演算器として用いられている。現在、商品化されている高性能の DSP は、同期パイプライン方式を採用している。業界最高水準の DSP である Texas Instruments の C64x コアは 2 つの演算器を持ち、クロック周波数 1GHz で、8000MIPS の処理性能を実現し、DSL(digital subscriber line)、VOP(video object plane) などのディジタルコミニケーション、ワイヤレス通信基地局、画像処理といったアプリケーションに対して使用されている。

プロセッサは微細加工技術による MOS デバイスの比例縮小により速度が向上してきた。しかし、今日の微細加工技術では比例縮小則によりゲートの遅延時間が短くなる一方、配線の遅延時間が一定であるため、配線遅延の割合が増えている。これにより、最大遅延時間をクロックサイクルとする同期パイプライン方式でのスループット向上が望めなくなってきている。配線の遅延時間を減らすためにアルミニウムから銅に、二酸化シリコン膜から low-k 膜へと配線材料の変更をしているが、配線遅延の割合の増加を止めることができていない。

つまり、同期パイプライン方式よりも高クロック動作を可能にする新たな高スループット化手法が必要である。

1.2　研究の目的

本研究は、ディジタル信号処理で多用される積和演算を高スループットで処理する積和演算器設計を目的とする。高スループット化手法として、ウェーブパイプライン方式 [1] を用いる。ウェーブパイプライン方式は最大遅延時間と最小遅延時間との差をクロックサイクルとするため、従来の同期パイプライン方式よりも高クロック動作を可能にすることができる。しかし、回路設計が難しく、設計に時間がかかるという問題点がある。

ウェーブパイプライン化を簡単に行うためには、簡単な回路構成で積和演算器を設計する必要がある。本研究では、規則的な配列で積和演算器を構成し、ウェーブパイプライン方式を用いて、クロック周波数の限界を探る。

1.3　本論文の構成

本論文は、5 章で構成されている。

第 2 章では、 ウェーブパイプライン方式の概要とその問題点について述べる。

第 3 章では、 積和演算器の概要と演算器に生じるキャリー伝搬遅延とその解決法について述べ、シストリックアレイを用いた積和演算器の構成を示す。

第 4 章では、基本的な MOS トランジスタの性質を述べ、90 ナノプロセスルールを用いた MOS トランジスタの設計と論理ゲートの設計、全加算器と D-FF ラッチの設計とレイアウト、配線モデルの選定と HSpice による全加算器の遅延シミュレーション結果を示す。

第 5 章では、本研究の結論について述べる。

第2章　ウェーブパイプライン方式

2.1　同期パイプライン

パイプラインは、プロセッサ命令を複数のステージに分割することにより、スループットの向上をねらったものである。つまり、複数の命令が複数の資源を同一クロックサイクルで利用することを意味する。パイプラインでプロセッサを動作させるためには、各ステージ間でデータを保存しておく必要があるため、ステージ間にパイプラインラッチが必要である。クロックサイクルはすべてのステージの中で最大遅延時間を持つステージによって決まる。

図 1.1 では、EXE ステージが最大遅延時間を持つステージであり、このステージによりクロックサイクルが決定されている。

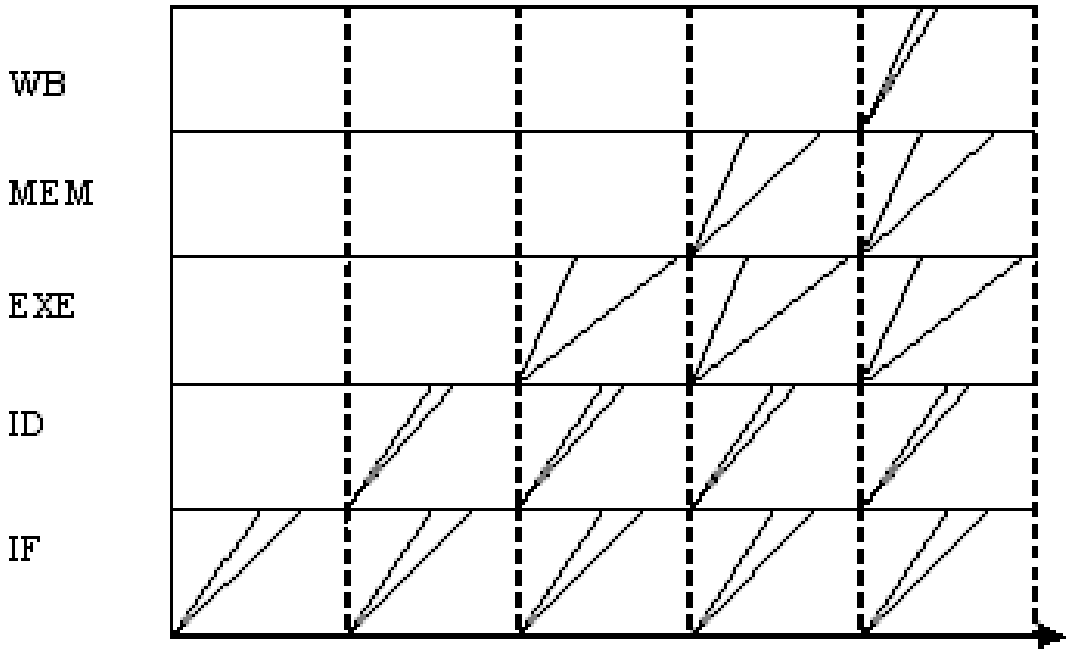


図 2.1: 同期パイプライン

2.2 ウェーブパイプライン

ウェーブパイプラインは、最大遅延時間と最小遅延時間の差をクロックサイクルとする。遅延差が小さくなれば、クロック周波数が向上することから、最小遅延時間のパスに遅延バッファを挿入する遅延均等化作業により、高クロック動作を実現することができる(図 2.2 参照)。図 2.3 には、ボトルネックとなるステージをウェーブパイプラインとした図を示す。ウェーブパイプラインにより、クロックサイクルは前のステージと同じであるが、ラッチのタイミングが違うことに注目してほしい。つまり、多相クロックを使用することを意味している。ウェーブパイプラインをボトルネックとなるステージに適用することにより、回路全体のクロックを向上させることができる。

しかし、ウェーブパイプラインには下記に示すような問題点がある。

1) 遅延バッファ挿入による面積増加
2) 厳密な遅延解析が必要
3) 多相クロックを使用するため、回路設計が難しく、遅延解析に不向き
4) 専用 CAD がないため、手作業による回路設計
5) 多相クロックを用いるのでフォワディングが困難

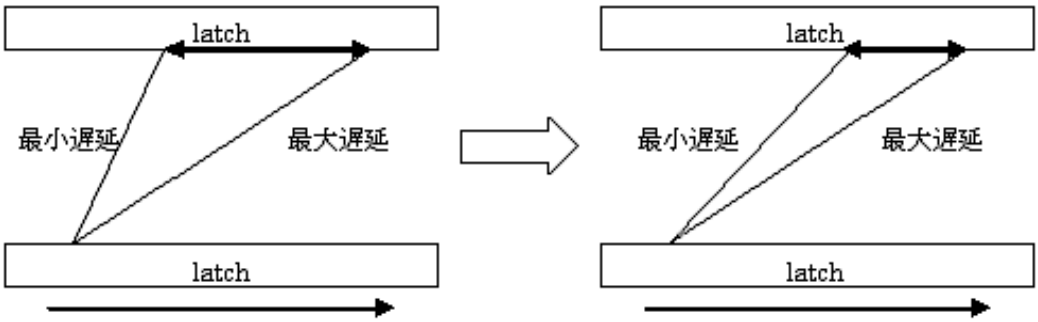


図 2.2: 遅延均等化

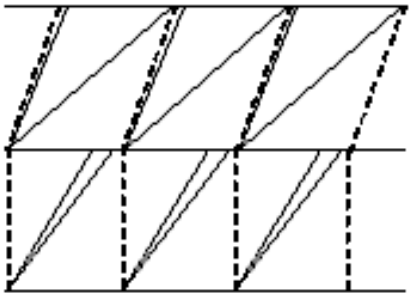


図 2.3: ウェーブパイプライン

第3章　積和演算器の構成

3.1　積和演算器

ディジタル信号処理では、積和演算を繰り返す畳み込み演算が多い。このため、DSP には積和演算を高速で処理するための乗算器が搭載されている。その乗算器の後段に累算加算器を置き、積和演算器として用いられている。

積和演算器は図 3.1 に示すように乗算器と加算器から構成される。積和演算を処理するために、まず乗算器で乗算を行う。次に乗算結果を乗算器後段の加算器に入れ加算を行う。加算結果は累算用レジスタに入り、加算器にフィードバックされることにより累算が行われる。このようにして積和演算を実現している。

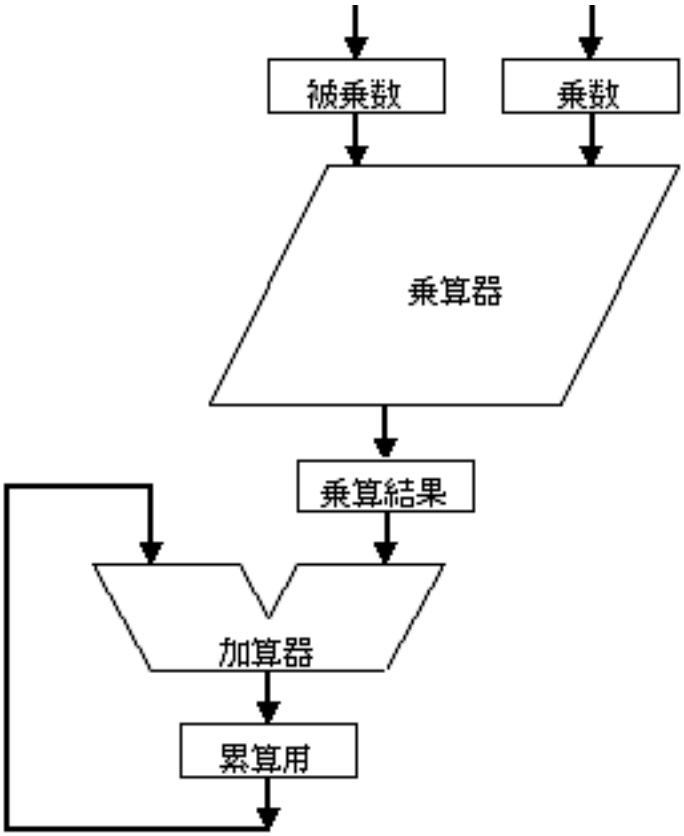


図 3.1: 積和演算器

3.2 乗算器

乗算器は、図 3.2 に示すように部分積生成回路と部分積加算回路から構成される。本論文では、並列加算回路を多段に接続した並列乗算器について述べる。

2 進数の部分積を生成するには、乗数と被乗数の論理積を取ればよい。よって、部分積生成回路には AND 素子を用いる。図 3.3 に示すように部分積の数だけ AND 素子が必要となる。

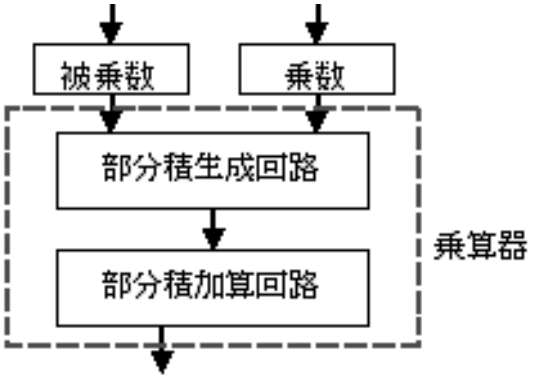


図 3.2: 乗算器

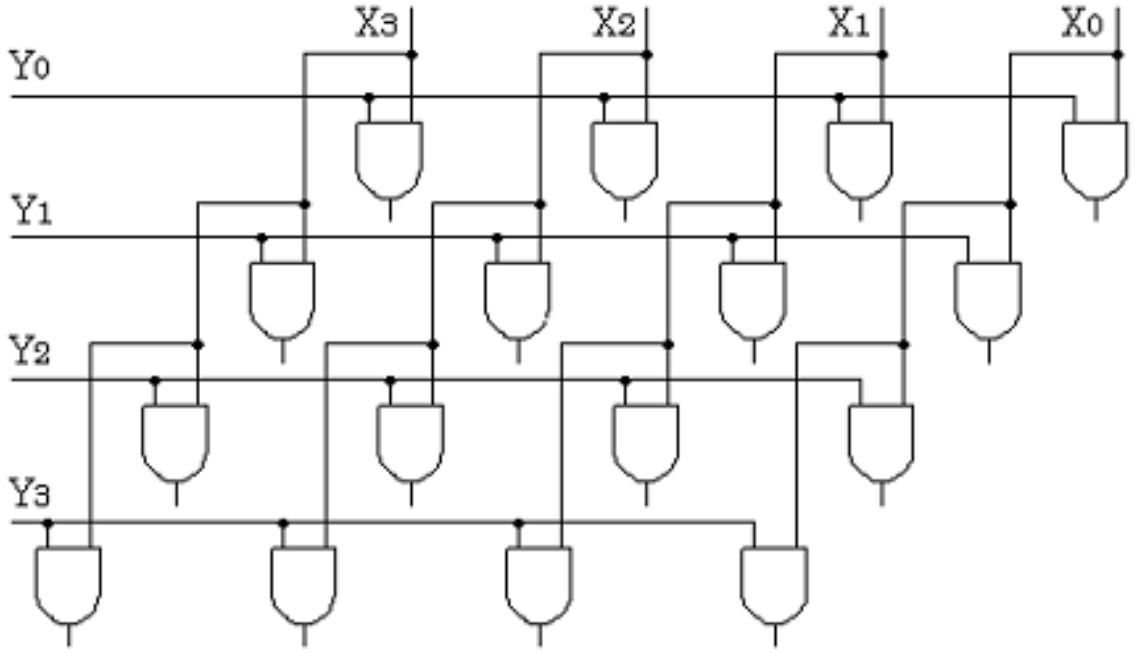


図 3.3: 4 × 4 ビット乗算の部分積生成回路

部分積加算回路は、部分積を加算するために全加算器 (FA) や半加算器 (HA) を用いる。図 3.4 に 4 × 4 ビット乗算の一般的な部分積加算回路を示す。図に示すように、キャリーセーブアダー接続の並列加算器が多段に接続されており、最終段には CLA（桁上げ先見）回路付きの加算器が用いられている。乗算の高速化のカギは、部分積加算回路の加算過程でキャリー伝搬遅延を生じないようにすることである。

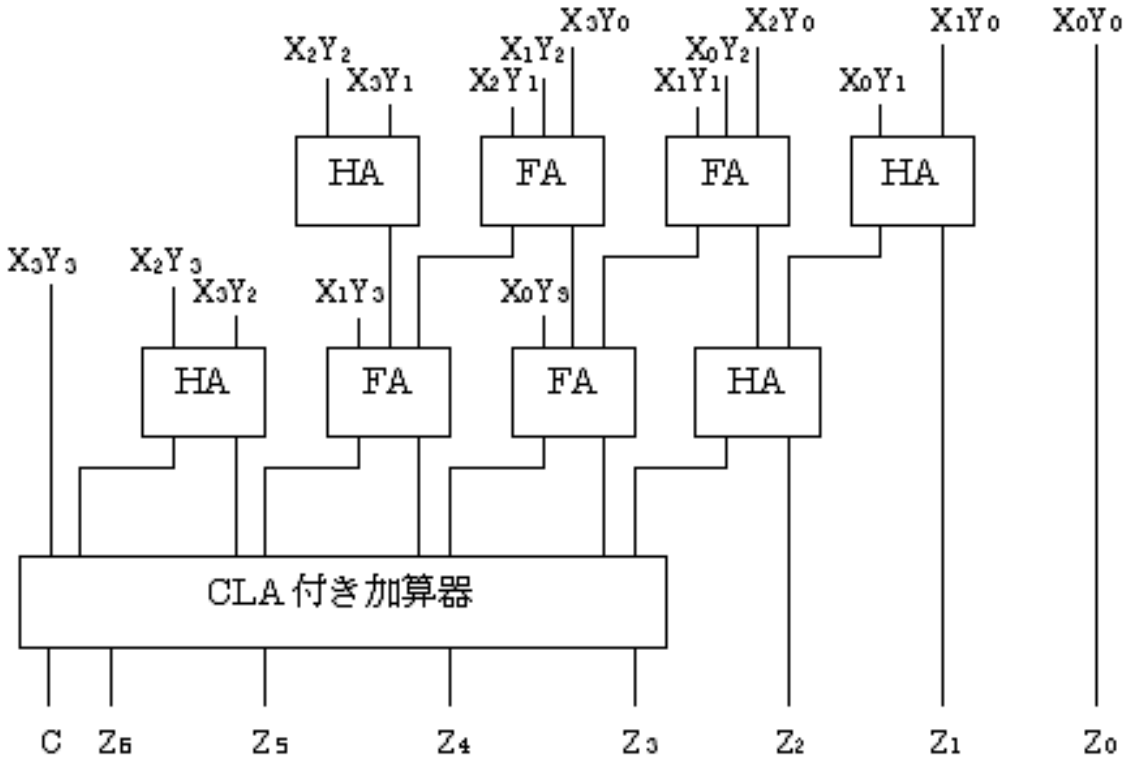


図 3.4: 4 × 4 ビット乗算の一般的な部分積加算回路

3.3 キャリー伝搬遅延

図 3.5 のようなリップルキャリー接続の並列加算器では、キャリー伝搬遅延が生じる。このため、上位ビットの加算は下からのキャリー伝搬を待たなくてはならず、接続されている全加算器の段数分遅延が生じる。このキャリー伝搬遅延が演算の高速化の妨げとなっている。

前節の図 3.4 で紹介したキャリーセーブアダー接続では、自段内でのキャリー伝搬をなくすため、キャリー信号を次の加算段に伝え、演算の高速化を行っている。しかし、最終段ではキャリー伝搬が生じる。キャリー伝搬遅延の影響をなくすために CLA(carry lookahead adder) 回路が用いられる。

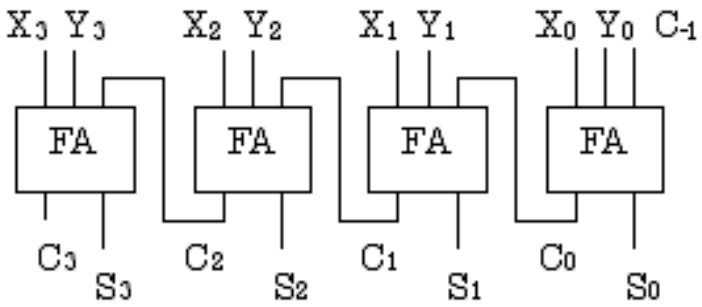


図 3.5: リップルキャリー接続の並列加算器

表 3.1: 全加算器の真理値表

X	Y	Ci	S	Co
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

全加算器の論理式からキャリー信号 C_i を表すと、

$$C_{out} = X \times Y + X \times C_{in} + Y \times C_{in} \tag{3.1}$$

$$G = X \times Y$$

$$P = X + Y$$

$$C_n = G_n + G_{n-1} \times P_n + G_{n-2} \times P_{n-1} \times P_n + \cdots + G_0 \times P_1 \times P_2 \times \cdots \times P_n + P_0 \times P_1 \times P_2 \times \cdots \times C_{-1} \tag{3.2}$$

となる。これは、すべての桁のキャリー信号が自桁の入力信号と最下位のキャリー信号だけで生成できることを表している。これを CLA と呼ぶ。図 3.6 に示す CLA 回路を用いれば、下からのキャリー信号を待たなくても自桁のキャリー信号を生成でき、演算を高速に行える。しかし、CLA 回路には下記に示すような問題点がある。

1) ビット数が増えると、ファンアウト数が増える

2) ビット数が増えると、ファンイン数が増える

3) ビット数が増えると、さらに配線が複雑になる
これらの問題点は、遅延時間に影響を与える。このため、ウェーブパイプライン化（遅延均等化作業）には適さないと考えられる。

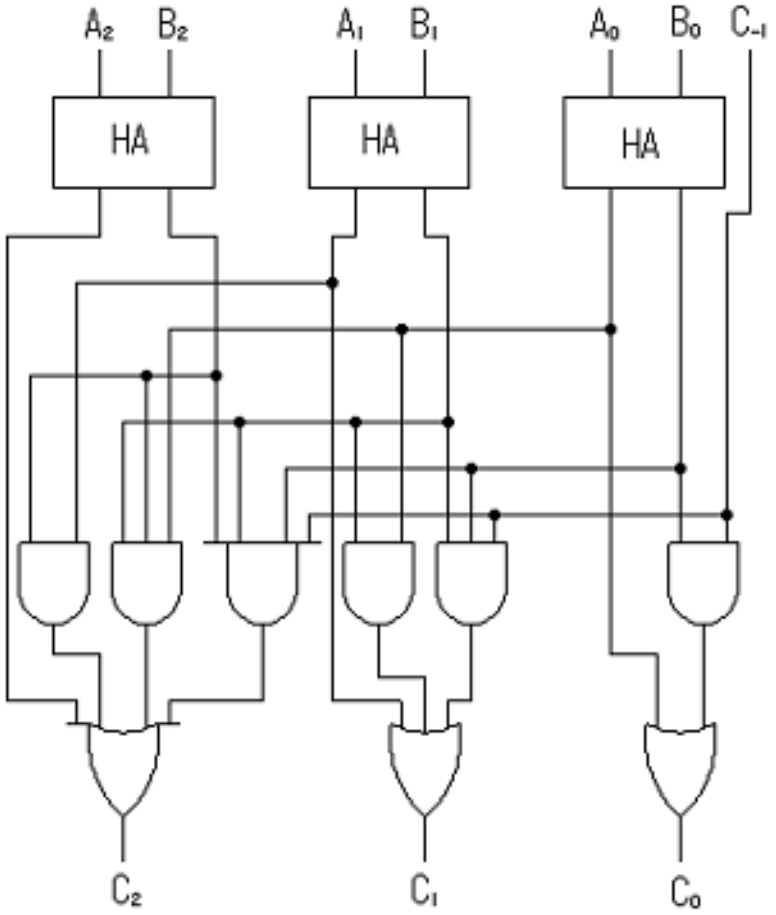


図 3.6: 3 ビット分の CLA 回路

3.4　シストリックアレイを用いた積和演算器

本研究では、ウェーブパイプライン化を簡単に行うために演算器の規則的な配列に着目し、簡単な回路構成で積和演算器を構成する。規則的な配列を用いたアーキテクチャとしてシストリックアレイがある。本節では、シストリックアレイを用いた積和演算器の構成について説明する。また、CLA 回路を使用せずにキャリー伝搬遅延をなくす方法としてシフトレジスタを用いた部分積生成について説明する。本研究で提案するシストリックアレイを用いた積和演算器ではキャリー伝搬遅延が生じない。

3.4.1 シストリックアレイアーキテクチャ

シストリックアレイ [2] は 1980 年代に H.T.Kung により提案されたアーキテクチャである。複数の Processing Element を規則的に配置し、隣接する PE 間を配線接続し、一定のタイミングでデータを受け入れて処理する一般化されたパイプライン処理システムである。

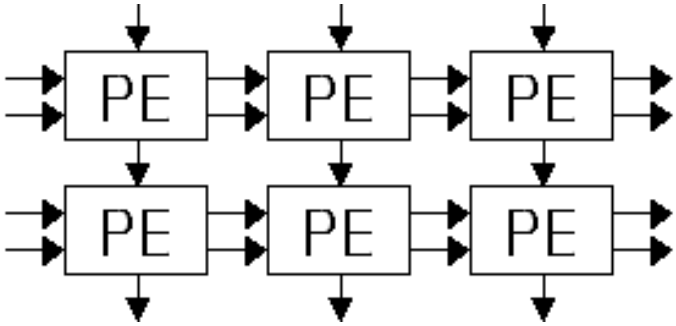


図 3.7: シストリックアレイ

3.4.2 シストリックアレイ乗算器

乗算器の部分積加算回路にシストリックアレイを適用する。部分積加算回路をすべてリップルキャリー接続の全加算器で構成する。全加算器の最大遅延時間をクロックサイクルとし、隣接する全加算器に信号を伝えていく。図 3.8 にシストリックアレイを適用した 4×4 ビット乗算の部分積加算回路を示す。実線の四角は全加算器、点線の四角はシストリックアレイ動作をサポートするためのダミーを表している。右側の図は部分積加算回路の点線で囲まれた部分の拡大図である。この部分積加算回路は、各全加算器間にパイプラインラッチがあり、10 段のパイプライン構成となっている。まず、最初のクロックで X_0Y_0 を全加算器に入れてやる。1 クロック遅らせて、X_1Y_0 と X_0Y_1 を全加算器に入れてやる。このとき、最初のクロックで全加算器に入れた X_0Y_0 の演算結果は隣接する次の全加算器に伝わっている。演算を終えたこの全加算器には、次の積和演算の次数の X_0Y_0 を入れてやる。さらに、1 クロック遅らせて X_2Y_0 と X_1Y_1 を全加算器に入れるという動作をする。このように 1 クロックずつタイミングを遅らせて部分積を部分積加算回路に入れてやることにより、キャリー伝搬遅延をなくすことができる。処理が終わった全加算器には、次の積和演算の次数のデータが流れてくるため、スループットは全加算器一段分の遅延時間となる (付録 A 参照)。

1 クロックずつタイミングを遅らせた部分積生成には、乗数と被乗数の各ビットに直列入力並列出力のシフトレジスタを用いる。図 3.9 に示すように 4×4 ビット乗算では、8 本のシフトレジスタが必要となる。小さな四角が 1 ビット分のレジスタを表している。こ

のシフトレジスタは左端から信号が入り、クロックごとに右に 1 ビットシフトしていく。シフトレジスタの並列出力を用い、必要とするタイミングで部分積生成が行えるように出力することにより、シストリックアレイ動作を実現することができる。

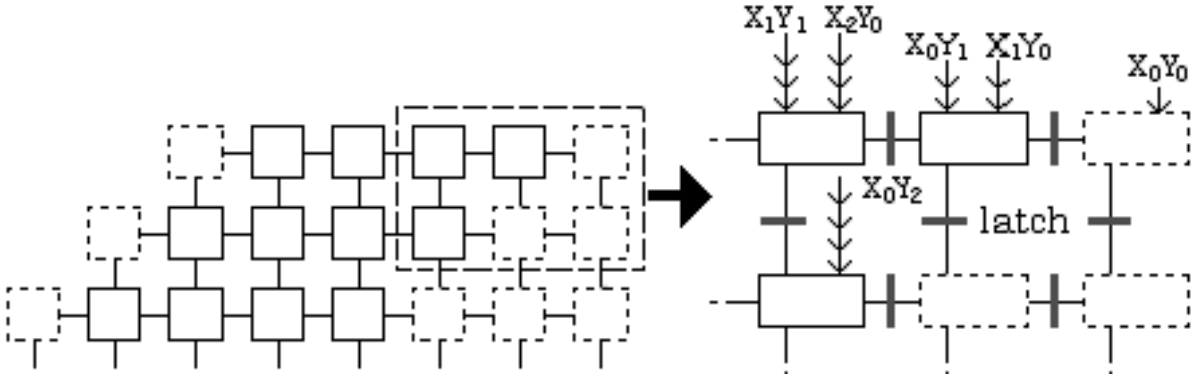


図 3.8: シストリックアレイを適用した 4 × 4 ビット乗算の部分積加算回路

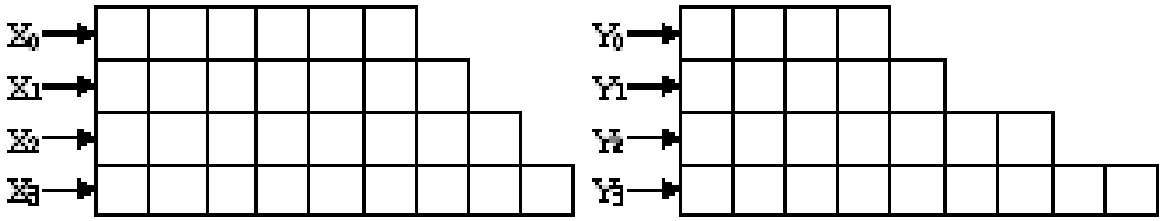


図 3.9: 4 × 4 ビット乗算の乗数、被乗数のシフトレジスタ

3.4.3 累算用加算器

積和演算器を構成する場合、乗算器後段に累算用加算器が必要となる。図 3.10 に示すように、シストリックアレイ乗算器の乗算結果の各ビットは 1 クロックずつ遅れて到着する階段状の遅延となる。これにより、累算用加算器もシストリックアレイ動作をするリップルキャリー接続の並列加算器にすることができる。3.3 節で説明したが、リップルキャリーの並列加算器は隣 (上位ビット) の全加算器にキャリーを伝えるためにキャリー伝搬遅延が生じる。しかし、あらかじめ入力データが下位ビットから順に 1 クロックずつ遅れて到着するため、キャリー伝搬遅延が生じない。スループットは全加算器一段分である。

累算用加算器の出力も階段状の遅延となるため、最後にシフトレジスタを用いて積和演算結果を一列に揃える桁合わせが必要となる。また、累算用加算器のビット数は積和演算のタップ数によって決まる。オーバーフローを起こさないように用意すればよい。

3.4.4　シストリックアレイ乗算器の面積改善

シストリックアレイ乗算器は従来の乗算器（図 3.4 参照) と比べると、乗数と被乗数の各ビットにシフトレジスタを用いるため面積が大きい。この小節では、設計に使用される MOS トランジスタの数を面積として議論する。面積改善方法として、乗算器の部分積加算回路のパイプライン段数を減らし、シフトレジスタのビット数を削減する。図 3.11 に全加算器を山形配置とした部分積加算回路を示す。山形配置は、点線で囲まれた部分の全加算器を下段に下ろしたものである。これにより、4 × 4 ビット乗算ではパイプライン段数を 10 → 7 段、シフトレジスタを 55 → 26 ビットに減らすことができる。乗算のビット数が増えても同様に約 2 分の 1 にシフトレジスタのビット数を減らすことができる。しかし、これでも従来の乗算器と比べると面積が大きい。表 3.2 に、これらの乗算器を用いて構成した積和演算器の面積比較を示す。累算用加算器には、乗数の 2 倍のビット分を用意してある。どちらもスループットは、全加算器一段分のパイプラインである。なお、これらの値は手計算により算出したものである。このため正確な値ではないが、CLA 回路の面積と、シストリックアレイに必要なシフトレジスタの面積では、後者の方が大きいと考えられる。よって、シストリックアレイを用いた積和演算器は従来の積和演算器 (CLA 回路使用) よりも面積が大きくなる。

表 3.2: 積和演算器の面積比較

	従来の乗算器を用いた積和演算器 [MOS 数]	シストリックアレイ乗算器を用いた積和演算器 [MOS 数]
4 × 4 ビット	2464	3072
8 × 8 ビット	9108	11328
16 × 16 ビット	34780	43392

3.4.5　ウェーブパイプラインの適用

今回の目的は、簡単にウェーブパイプライン化が行えるように簡単な回路構成で積和演算器を設計することである。シストリックアレイを適用した積和演算器はラッチと部分積生成回路を除けば、すべて全加算器で構成されていることに注目してほしい。つまり、遅延が均等な全加算器を設計することで積和演算器のウェーブパイプライン化が行える。全加算器は 3 入力であるため、入力の組み合わせが 64 通りある。つまり、64 通りの遅延解析が必要となる。

図 3.12 に本研究で提案する積和演算器の構成を示す。全加算器一段分の遅延時間で動作するようなパイプライン構成とする。

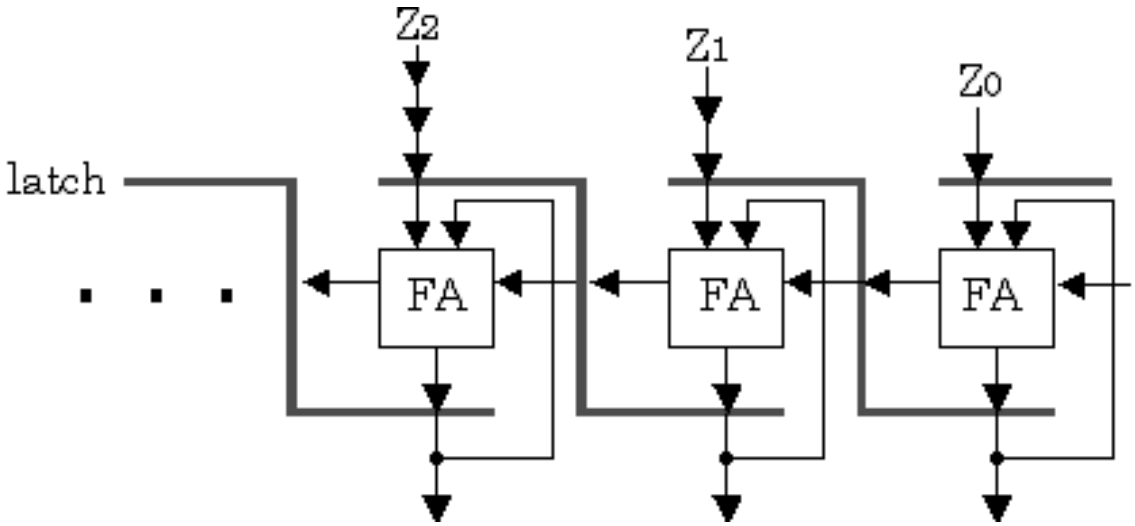


図 3.10: 累算用加算器

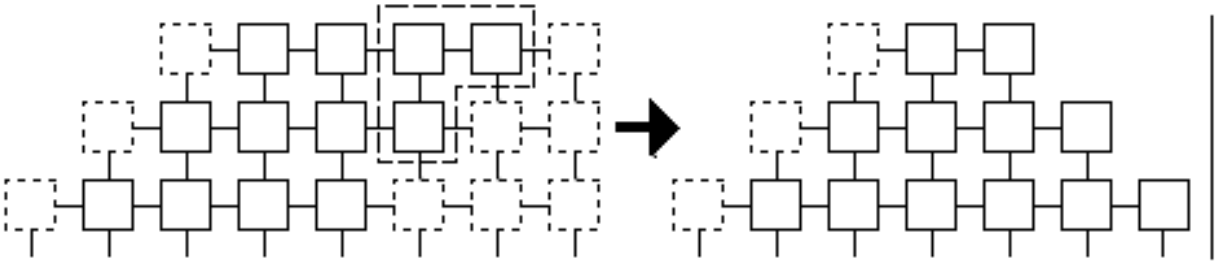


図 3.11: 山形配置の部分積加算回路

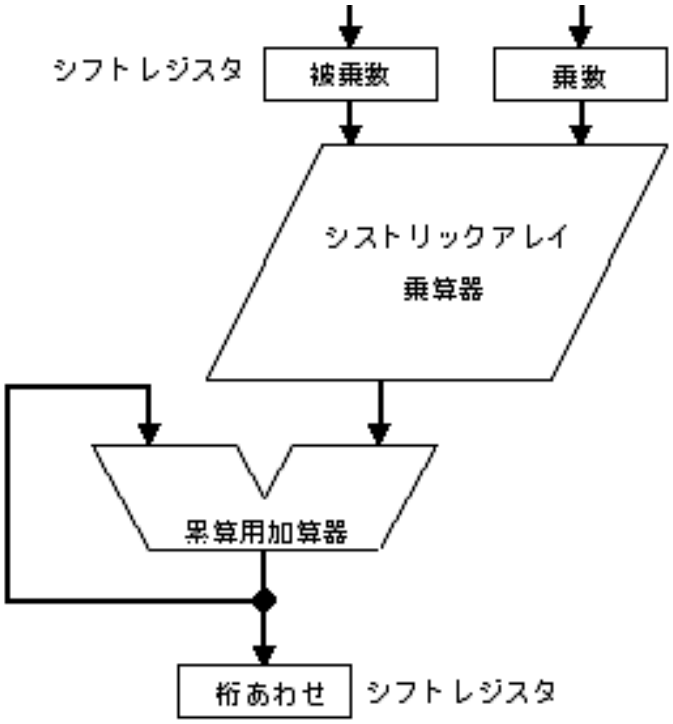


図 3.12: 本研究で提案する積和演算器構成

第4章　HSpiceによる遅延シミュレーション

4.1　MOSトランジスタの性質

4.1.1　MOSトランジスタの動作原理

MOSトランジスタの設計に入る前に簡単にMOSトランジスタの性質について述べる。図4.1はMOSトランジスタの断面図である。Lはゲート長、Wはトランジスタの幅、Dはゲート酸化膜厚である。MOSトランジスタは、ゲート電極の電圧を制御することによりソース－ドレイン間に流れる電流を制御している。

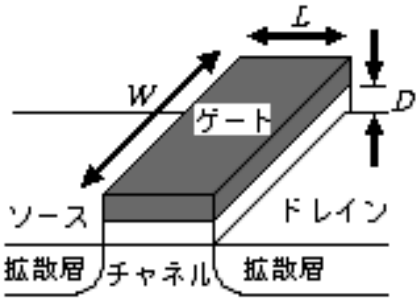


図 4.1: MOSトランジスタ

キャリアの走行時間をゲート長L、キャリア移動度μで表すと、

$$\tau = \frac{L}{\upsilon} = \frac{L}{\mu E_{ds}} = \frac{L^2}{\mu V_{ds}} \tag{4.1}$$

キャリアの走行速度　$\upsilon = \mu E$

電界　$E = \dfrac{V}{L}$

MOSトランジスタのゲート容量は次式で表される。

$$C_g = \varepsilon \frac{LW}{D} \tag{4.2}$$

誘電率　$\varepsilon = \varepsilon_r \varepsilon_0$

MOS トランジスタのソース－ドレイン間電流を式 4.1 と式 4.2 を用いて表すと、

$$I_{ds} = -\frac{dQ}{dt} \approx C_g(V_{gs} - V_{th})\frac{1}{\tau} = C_g(V_{gs} - V_{th})\frac{\mu V_{ds}}{L^2} = \frac{\mu\varepsilon W}{LD}(V_{gs} - V_{th})V_{ds} \tag{4.3}$$

MOS トランジスタのオン抵抗を式 4.3 を用いて表すと、

$$R_{on} = \frac{V_{ds}}{I_{ds}} = \frac{LD}{\mu\varepsilon W(V_{ds} - V_{th})} \tag{4.4}$$

MOS トランジスタを用いた素子の遅延時間は、オン抵抗とゲート容量の RC 積で表される。遅延時間を式 4.2 と式 4.4 を用いて表すと、

$$T_d = R_{on}C_g = \frac{L^2}{\mu(V_{gs} - V_{th})} \tag{4.5}$$

式 4.5 でわかるように、ゲート長 L を小さくすることで遅延時間を短くできる。プロセッサの速度向上は、微細加工技術における比例縮則によるものであることがわかる。

4.1.2 MOS トランジスタ論理回路の遅延特性

ここでは、例として CMOS インバータを用いて MOS の遅延特性を述べる。図 4.2(a) に駆動側インバータのファンアウト数 1 の回路、(b) にファンアウト数 n の回路を示す。図 4.3 には、それらの入出力波形を示す。(a) の回路では期待通りの出力波形が得られているのに対して、(b) の回路では出力波形が立ち上がりきるまでに立ち下がり始めており不定状態となる。この場合、クロック周波数を下げるか、または駆動側インバータの MOS の駆動能力を上げる必要がある。MOS の駆動能力を上げるには、MOS の幅 W を大きくすればよい。W を大きく取ることにより、抵抗値が下がり電流が流れやすくなる。

理論的に説明すると、インバータの遅延時間は自段の R_{on} と拡散容量 C_d、次段のゲート容量 C_g によって決まる。ファンアウト数が増加するとチャージする次段のゲート容量が増えるため、遅延時間が大きくなる。波形の立ち上がりが悪くなることを意味する。

4.2 回路設計

4.2.1 MOS トランジスタの設計

HSpice のシミュレーションで使用する MOS トランジスタを設計する。90 ナノプロセスルールを使用する。MOS トランジスタのパラメータは、BSIM ホームページ [3] を参考にした (付録 B 参照)。半導体メーカーが公表しているパラメータではないため、遠からず、近からずの値だと考えられる。設計した MOS トランジスタを図 4.4 に、パラメータを表 4.1 に示す。

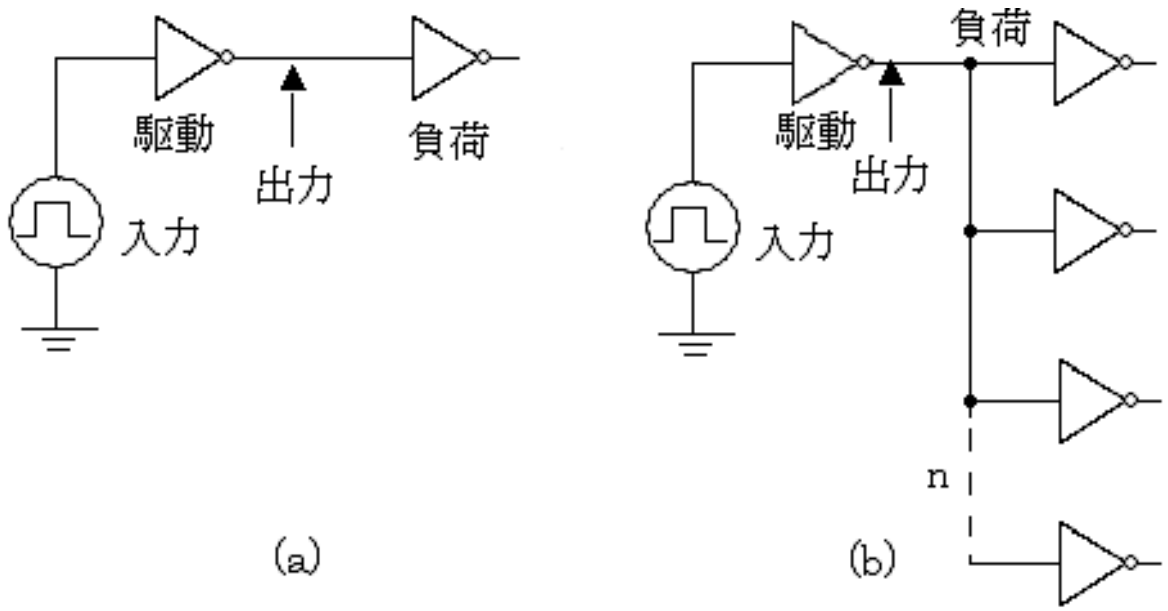


図 4.2: CMOS インバータ (a) ファンアウト数 1,(b) ファンアウト数 n

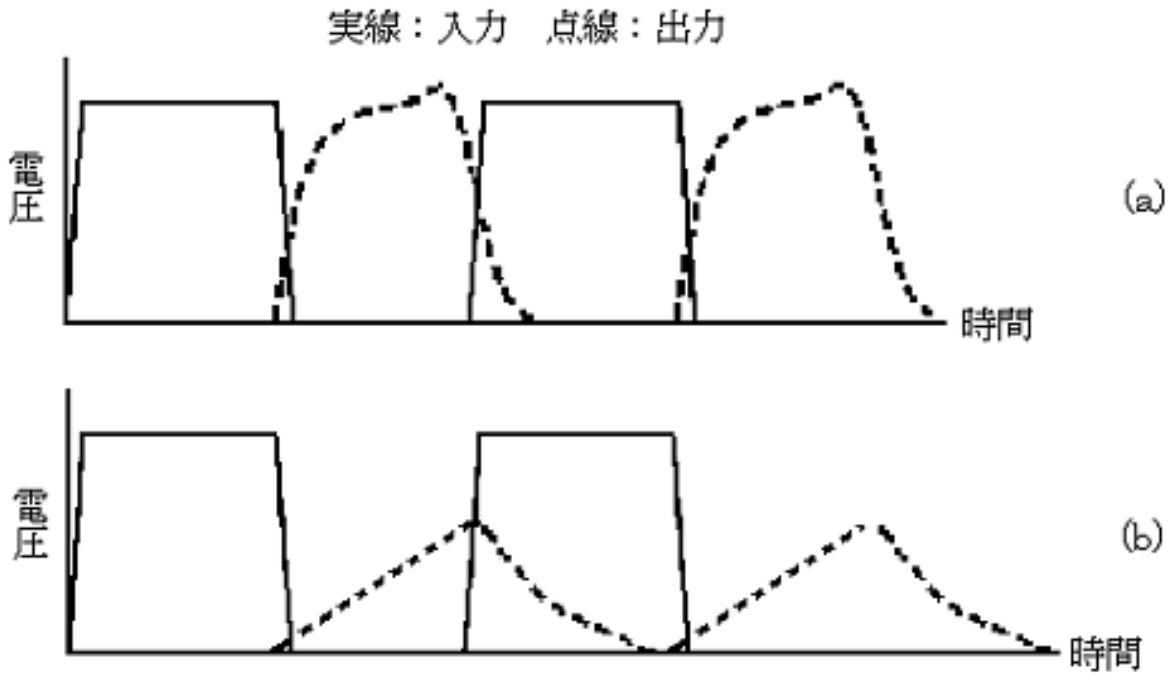


図 4.3: CMOS インバータ入出力波形 (a) ファンアウト数 1,(b) ファンアウト数 n

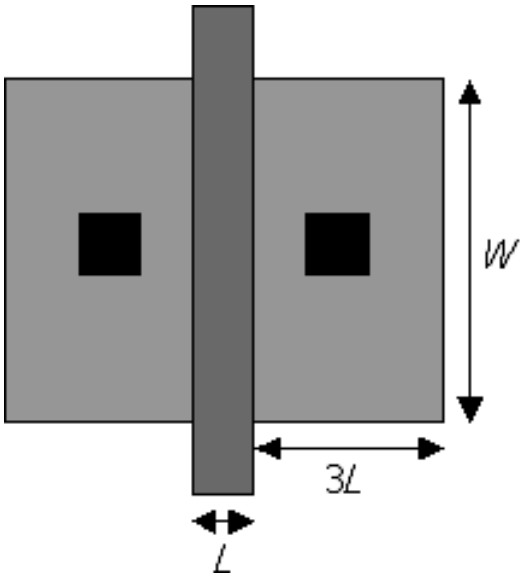


図 4.4: MOS トランジスタの設計

表 4.1: 設計した MOS トランジスタのパラメータ

MOS	n 形	p 形
界面近傍ピークドーピング濃度 $N_{ch}[cm^{-3}]$	9.7e17	1.04e18
ゲート長 L[nm]	90	90
ゲート酸化膜厚 t_{ox}[nm]	2.5	2.5
拡散の深さ X_j[nm]	40	40
移動度 $U0[cm^2/\mathrm{N/sec}]$	0.018	0.0055
スレショルド電圧 V_{th0}[V]	0.2607	-0.303

4.2.2 論理ゲート設計

全加算器に使用する論理ゲートを設計する。設計するのは、not ゲートと nand ゲート、遅延均等化に使用する buffer ゲートの 3 ゲートである。どんな入力の状態 (立ち下がり、立ち上がり) でも、遅延時間が等しくなるような論理ゲートを設計する。n 形と p 形の MOS はキャリアの移動度が違うため、同じ寸法で設計した場合では遅延時間が等しくならない。p 形の幅 W を n 形よりも大きめに設計することによって、入力の状態変化に対して遅延時間の等しい論理ゲートが設計できる。図 4.5 に設計した論理ゲートを示す。p 形 W+n 形 W=20L となるように設計している。表 4.2 に p 形と n 形の幅 W とゲート容量を示す。一例として、図 4.6 に not ゲートの入出力波形を示す。入力が立ち下がりの場合の遅延時間は、入力の立ち下がり 300mV から出力の立ち上がり 700mV までの時間である。

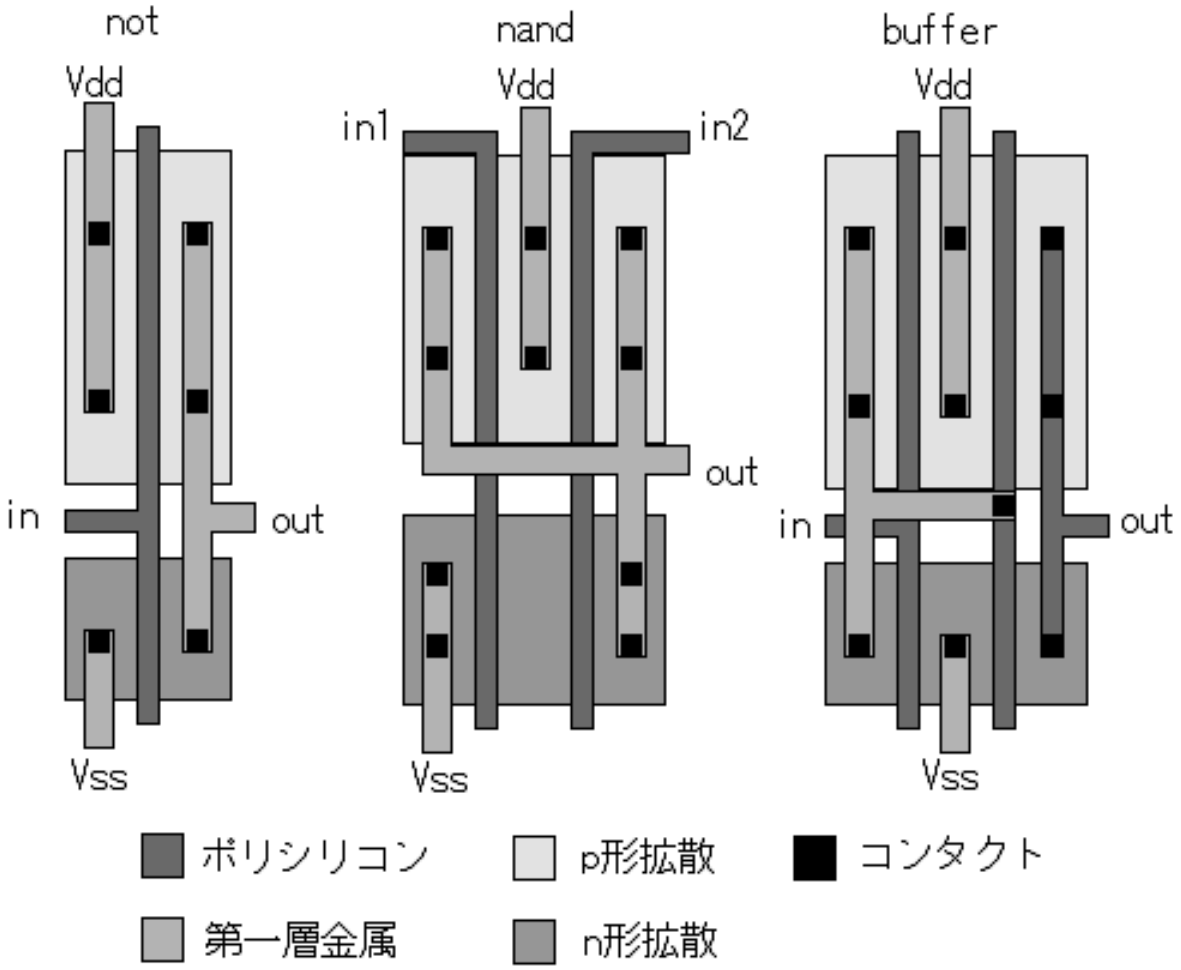


図 4.5: 論理ゲートの設計

表 4.2: 設計した論理ゲート

論理ゲート	not	nand	buffer
n 形 $W[\mu\mathrm{m}]$	0.54	0.72	0.54
p 形 $W[\mu\mathrm{m}]$	1.26	1.08	1.26
ゲート容量 C_g[fF]	2.24	2.24	2.24
共通電源 V_{dd}[V]	1.0	1.0	1.0
レイアウト面積 $S[\mu m^2]$	1.75	2.62	2.62
オン抵抗 R_{on}[kΩ]	3.01	3.47	2.97
拡散容量 C_d[fF]	1.67	2.79	1.67

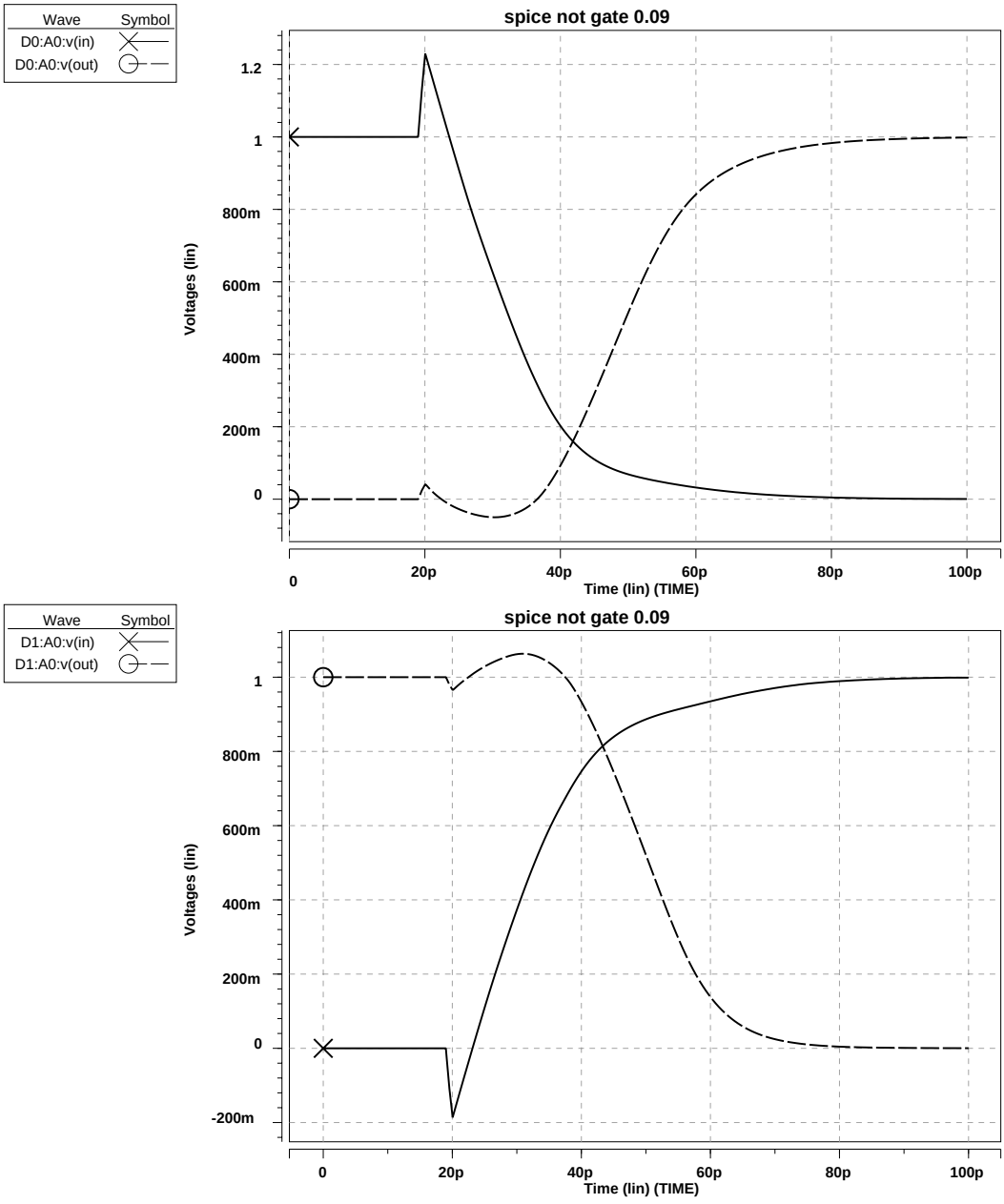


図 4.6: not ゲートの入出力波形 (入力立ち下がり; 入力立ち上がり)

4.2.3 全加算器の設計

全加算器の論理回路は XOR ゲートを用いた構成が一般的であるが、本研究では遅延均等化を考慮して not と nand ゲートのみで構成する。図 4.7 に not と nand ゲートにより構成した全加算器の論理回路を示す。論理段数は 2〜6 段である。

この論理回路図から最大遅延時間のパスと最小遅延時間のパスを推測することができる。最大遅延時間のパスは、論理ゲートを 6 段通るパスである。

入力 a または b → not → nand → nand → not → nand → not →出力 s

最小遅延時間のパスは、論理ゲートを 2 段通るパスである。

入力 a または b → nand → nand →出力 co

入力 ci → nand → nand →出力 s または co

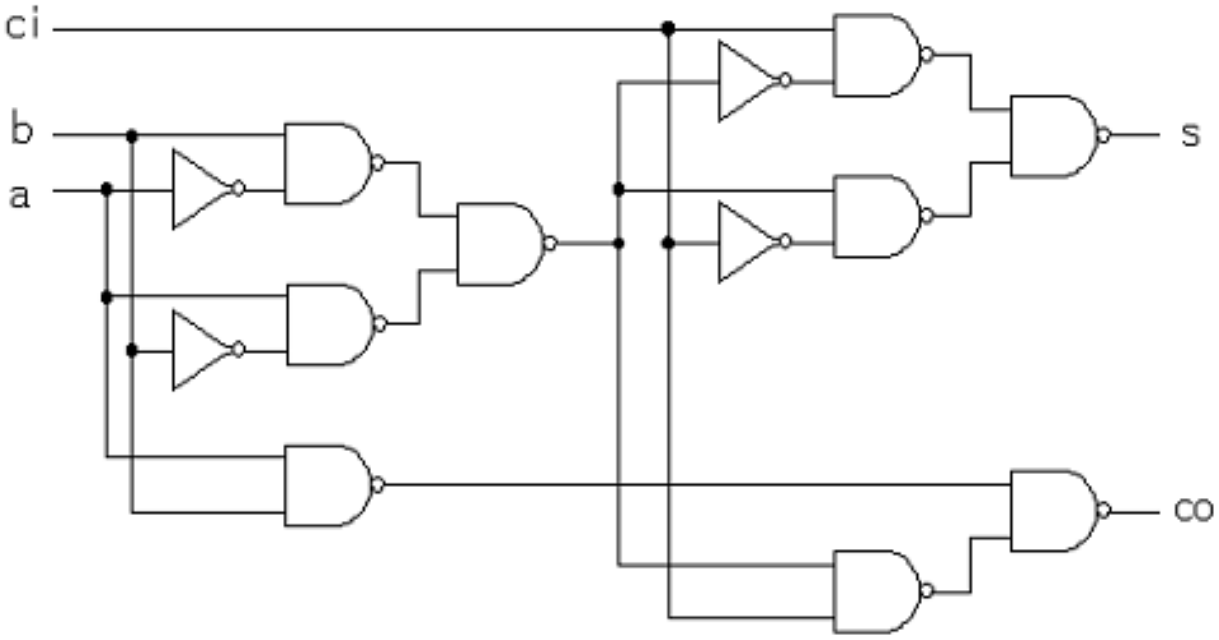


図 4.7: 全加算器の論理回路

4.2.4 全加算器の遅延均等化

ウェーブパイプライン化を行うため、最大遅延時間と最小遅延時間の差が小さくなるように設計する。遅延均等化には、最小遅延時間のパスに遅延バッファを挿入する。図 4.8 に遅延均等化後の全加算器を示す。この遅延均等化は、not ゲートを除く論理段数が等しくなるように遅延バッファを挿入したものである。これにより、論理段数が 4〜6 段となった。遅延差の短縮には効果的な方法である。しかし、MOS の個数は 36.3 パーセント (44 → 60 個) 増加した。面積増加を抑えるためには、うまくレイアウトする必要がある。

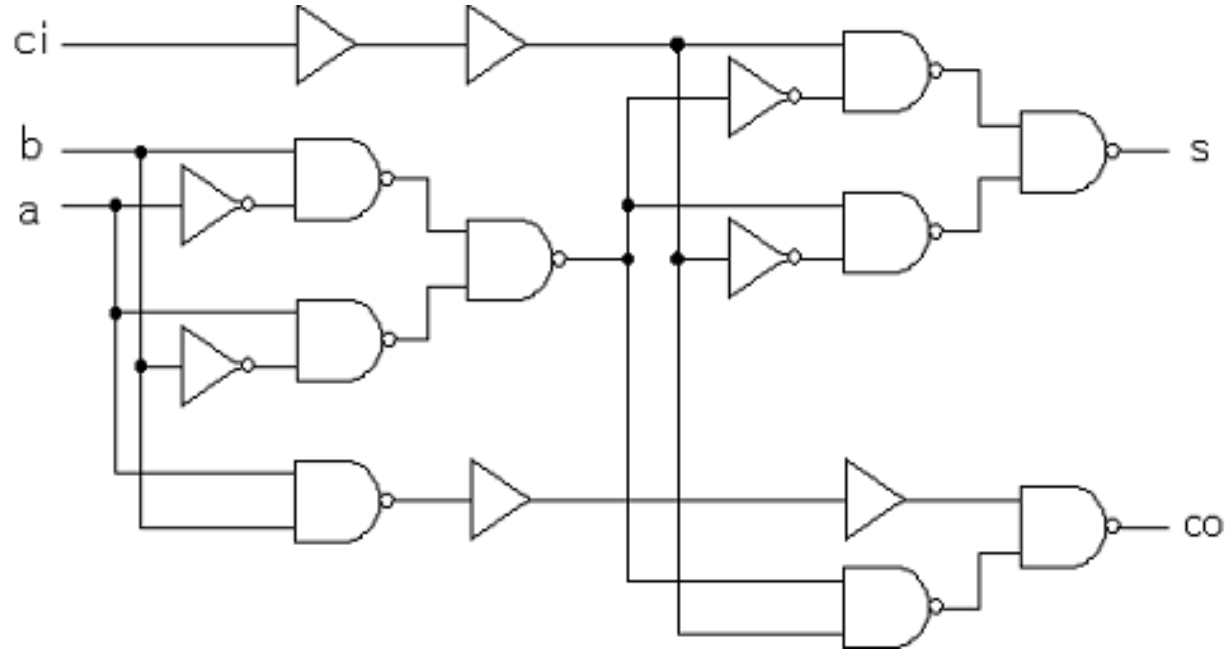


図 4.8: 遅延均等化後の全加算器

4.2.5 ラッチの設計

パイプラインラッチに使用するラッチを設計する。本研究では、図 4.9 に示すトランスファと not ゲートで構成した D-FF マスタースレーブを使用する。

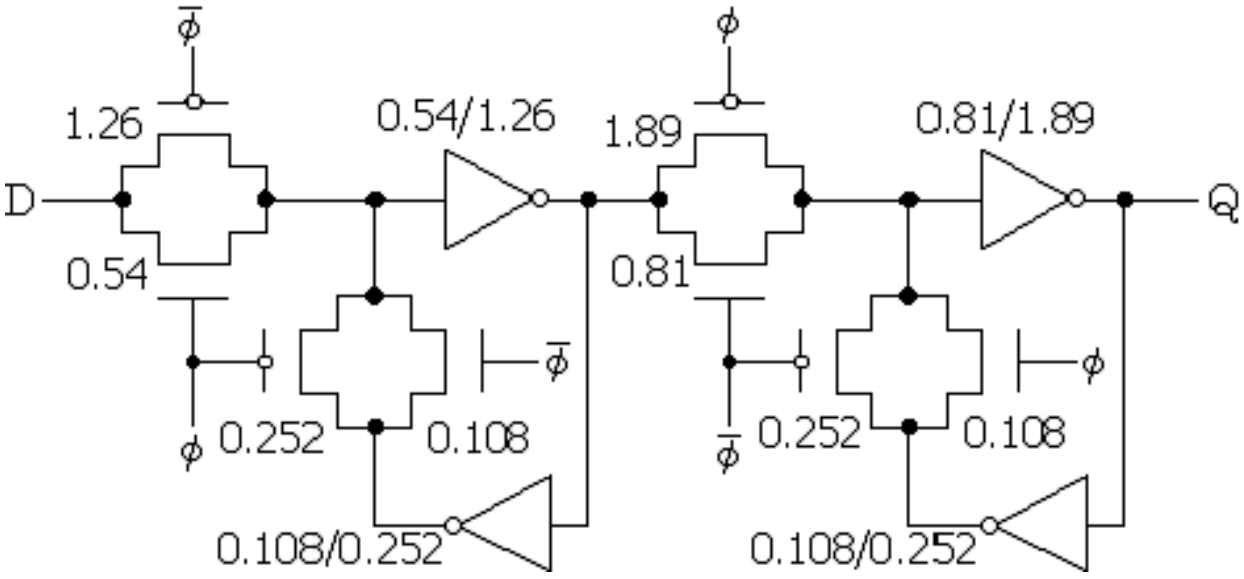


図 4.9: D-FF マスタースレーブ (n 形 W/p 形 W [μm])

ラッチの MOS の大きさは、入力側 (マスター) は全加算器の出力で駆動できる大きさ、出力側 (スレーブ) は全加算器の入力を駆動できる大きさとした。また、入力信号がフィードバック信号に打ち勝つようにするため、帰還側のインバータとトランスファゲートは小さく設計する必要がある。帰還側の MOS を小さくすることで、マスター側とスレーブ側の出力インバータの負荷の軽減にも効果がある。このように高クロック動作の回路を作るには、前後の MOS の大きさに注意する必要がある [4]。

4.3 全加算器の遅延シミュレーション

4.3.1 レイアウト

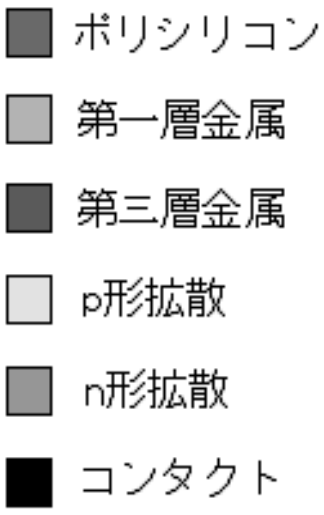
全加算器の遅延シミュレーションを行う前に、レイアウトを行い論理ゲート間の配線長を求める必要がある。レイアウトには固定格子を用いる [5]。1grid をゲート長 90nm とする。第一層目の金属配線には、120nm 幅の銅配線を用いた。電源配線やクロック配線は、第三層目 (銅配線) 以降の配線で行う。

図 4.10 に遅延均等化前の全加算器のレイアウトを示す。論理ゲートの MOS サイズは、すべて n 形 W+p 形 W=20L で設計している。

図 4.11 に遅延均等化後の全加算器のレイアウトを示す。論理ゲートの MOS サイズは、ラッチ出力の負荷を減らすために全加算器の入力側の MOS を小さく (n 形 W+p 形 W=10L, 5L) 、全加算器の出力の駆動能力を上げるために大きな MOS サイズ (n 形 W+p 形 W=30L) で設計している。高クロック動作を考慮した設計である。

D-FF マスタースレーブを用いたラッチのレイアウトを図 4.12 に示す。

表 4.3 に、これらのレイアウト面積を示す。全加算器の遅延バッファ挿入によるレイアウト面積の増加は 13.3 パーセントである。



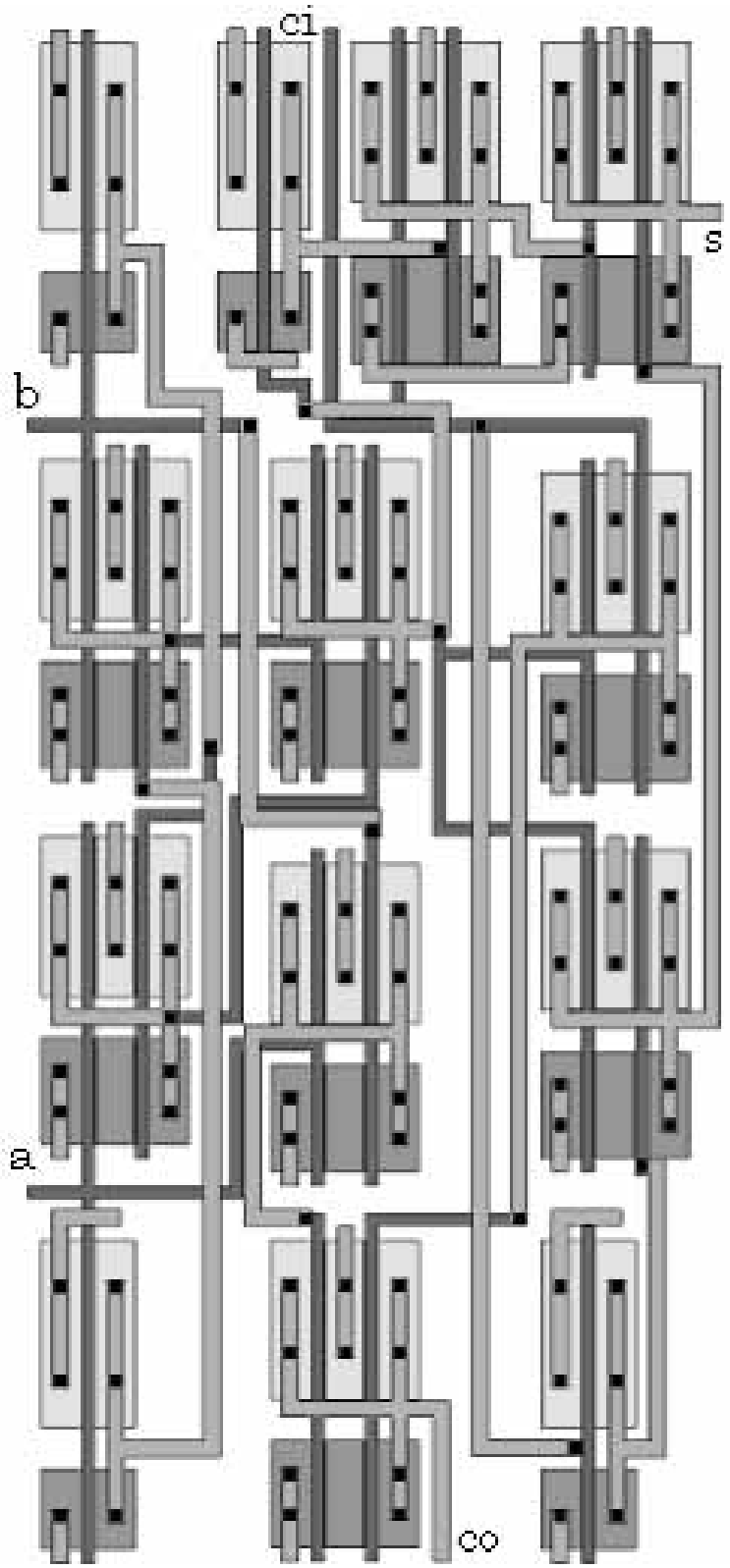


図 4.10: 全加算器のレイアウト

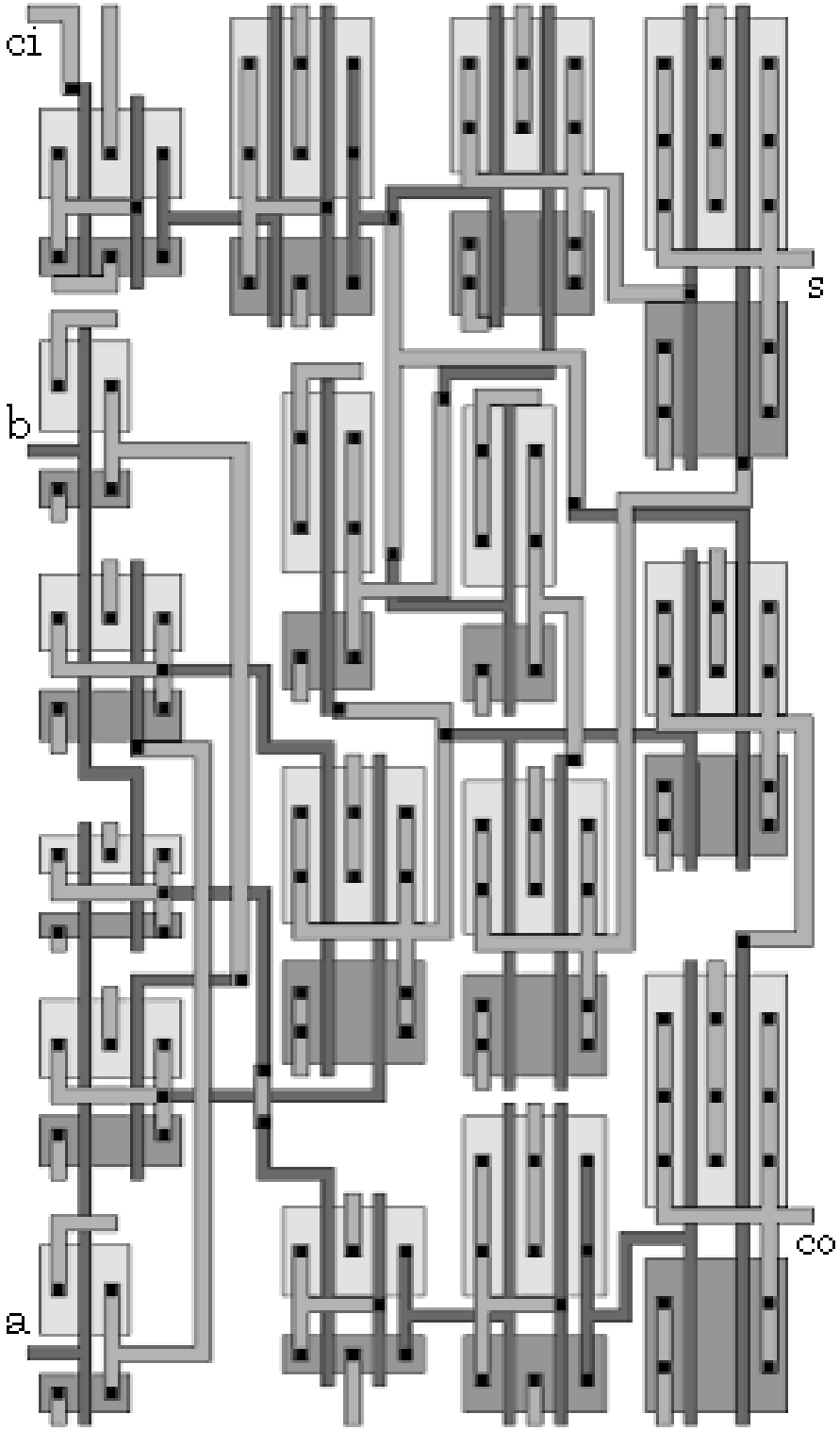


図 4.11: 遅延均等化全加算器のレイアウト

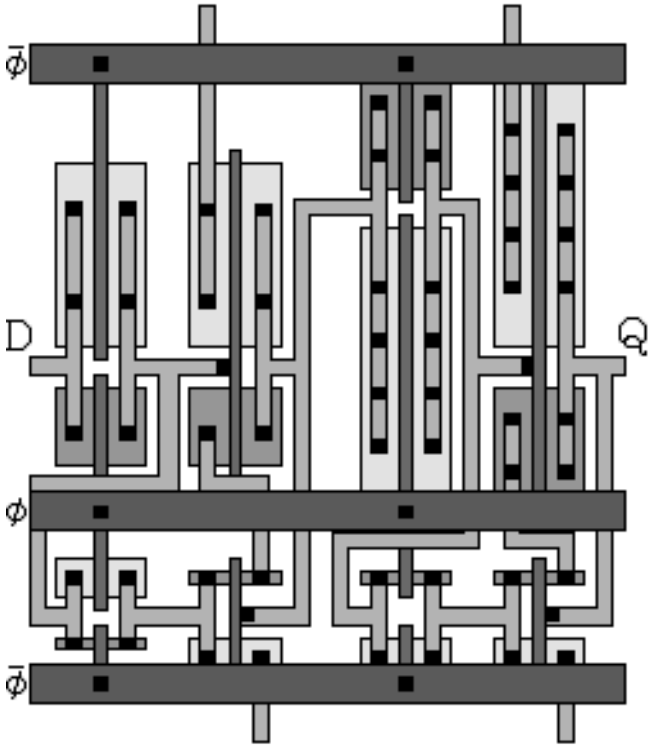


図 4.12: ラッチのレイアウト

表 4.3: レイアウト面積

	縦 [μm]	横 [μm]	面積 $S[\mu m^2]$
全加算器	10.44	4.68	48.86
遅延均等化全加算器	10.08	5.49	55.34
ラッチ	5.22	4.05	21.14

4.3.2 配線モデル

配線遅延を含めた遅延シミュレーションを行うために、配線モデルを決める必要がある[6]。LSI 中の配線は RC 分布定数線路となっている。本研究では、RC 一段の L 形集中定数回路として近似する。配線抵抗 R_l と配線容量 C_l は、下記の式を用いて算出する。
なお、抵抗率 ρ, 配線の厚さ t, 配線長 l, 配線幅 w, 酸化膜厚 d, 誘電率 ε_0, 二酸化シリコンの比誘電率 ε_{SiO_2} である。

$$R_l = (\frac{\rho}{t})(\frac{l}{w}) \tag{4.6}$$

$$C_l = (\frac{\varepsilon_0 \varepsilon_{SiO_2}}{d})wl \tag{4.7}$$

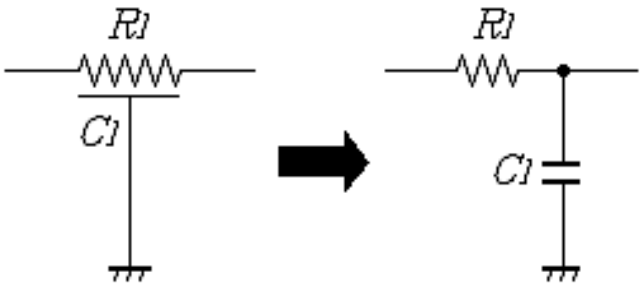


図 4.13: 配線モデル (RC 一段 L 形集中定数回路)

今回、使用した配線材料を表 4.4 にまとめる。配線間距離に関しては、図 4.14 に示す LSI の断面モデルで考えた。なお、L=90nm, t_{ox}=2.5nm, d=95nm, L'=120nm, d'=156nm である。HSpice には信号が伝搬する配線のみ RC 一段の L 形集中定数線路で記述した。

表 4.4: 配線材料

	抵抗率 $\rho[\Omega\mathrm{m}]$	比誘電率 ε_{SiO_2}
ポリシリコン	$9e^{-6}$	\
銅	$1.736e^{-8}$	\
二酸化シリコン	\	3.9

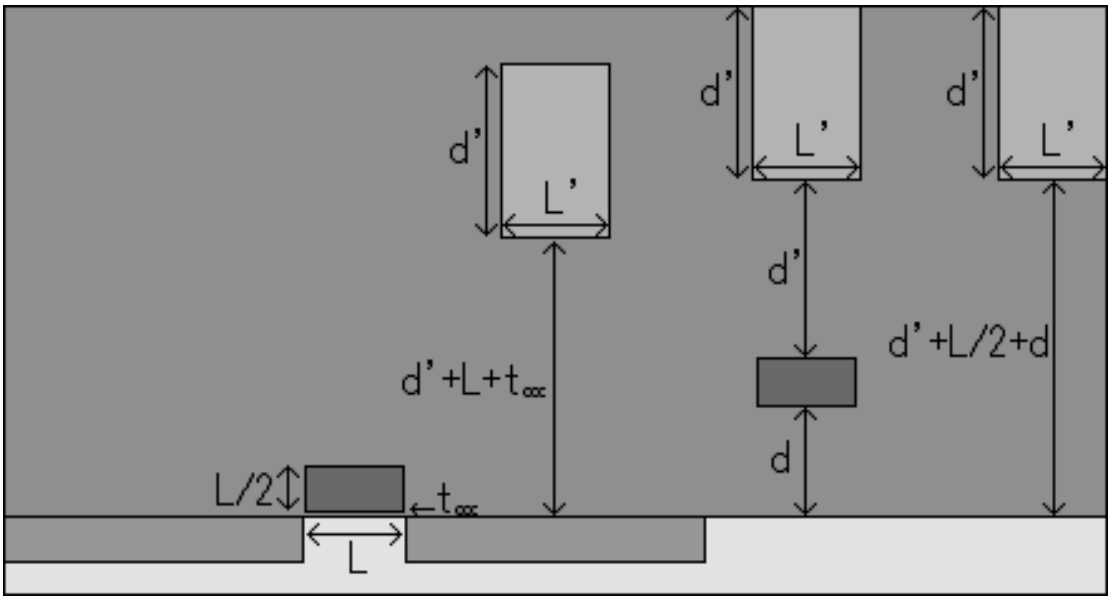


図 4.14: LSI の断面モデル

4.3.3 全加算器の遅延解析

HSpice(level49)[7] を用いて全加算器の入力変化 64 通りの遅延解析を行う。図 4.15 に示すように、全加算器の入力にはラッチの出力と同等の駆動能力を持つインバータを通して入力し、全加算器の出力にはラッチの入力と同等の負荷を持つインバータを接続する。

遅延解析の結果を表 4.5 と表 4.6 に示す。遅延時間は、0.25ps 刻みである。遅延均等化前の全加算器では、最大遅延と最小遅延の差が 242.5ps ある。全加算器の遅延均等化を行うことで、最大遅延と最小遅延の差が 62.25ps に縮まった。しかし、この遅延差=クロックサイクルにはならない。クロックサイクルは遅延差とラッチによるオーバーヘッド (遅延時間と後段のラッチのセットアップ、ホールドタイム) の和で決まる。

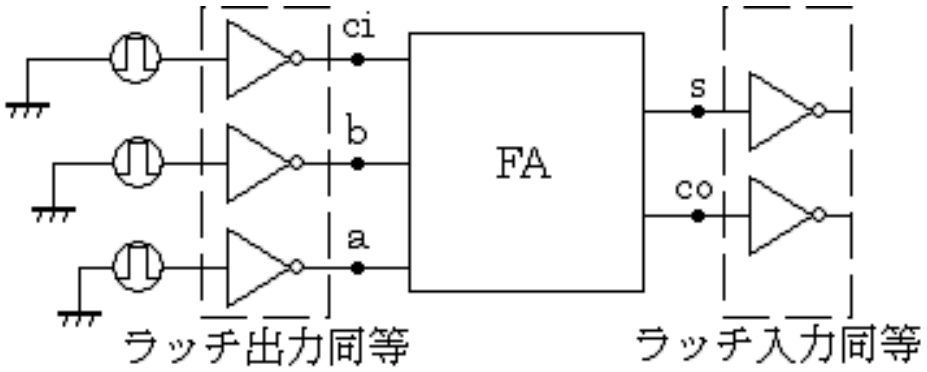


図 4.15: 全加算器の遅延解析

表 4.5: 全加算器の遅延解析

入力			出力		出力 s 側	出力 co 側
a	b	ci	s	co	遅延時間 [ps]	遅延時間 [ps]
0→0	0→0	0→0	0→0	0→0		
0→0	0→0	0→1	0→1	0→0	171.5	
0→0	0→1	0→0	0→1	0→0	276.5	
0→0	0→1	0→1	0→0	0→1		280
0→1	0→0	0→0	0→1	0→0	303.5	
0→1	0→0	0→1	0→0	0→1	171.25	300.75
0→1	0→1	0→0	0→0	0→1		154
0→1	0→1	0→1	0→1	0→1	171.5	163.5
0→0	0→0	1→0	1→0	0→0	179.5	
0→0	0→0	1→1	1→1	0→0		
0→0	0→1	1→0	1→1	0→0	275	
0→0	0→1	1→1	1→0	0→1	261.25	256.5
0→1	0→0	1→0	1→1	0→0	304.75	
0→1	0→0	1→1	1→0	0→1	289.5	284.25
0→1	0→1	1→0	1→0	0→1	179.5	162.75
0→1	0→1	1→1	1→1	0→1		154
0→0	1→0	0→0	1→0	0→0	267.75	
0→0	1→0	0→1	1→1	0→0	271.25	
0→0	1→1	0→0	1→1	0→0		
0→0	1→1	0→1	1→0	0→1	212	224.75
0→1	1→0	0→0	1→1	0→0		
0→1	1→0	0→1	1→0	0→1	212	229.25
0→1	1→1	0→0	1→0	0→1	305.75	287.25
0→1	1→1	0→1	1→1	0→1	318.25	151.5
0→0	1→0	1→0	0→0	1→0		215.75
0→0	1→0	1→1	0→1	1→0	271	261
0→0	1→1	1→0	0→1	1→0	219	226.75
0→0	1→1	1→1	0→0	1→1		
0→1	1→0	1→0	0→1	1→0	222.75	226.75
0→1	1→0	1→1	0→0	1→1		
0→1	1→1	1→0	0→0	1→1	298.75	
0→1	1→1	1→1	0→1	1→1	309	

a	b	ci	s	co	遅延時間 [ps]	遅延時間 [ps]
1→0	0→0	0→0	1→0	0→0	297	
1→0	0→0	0→1	1→1	0→0	305.75	
1→0	0→1	0→0	1→1	0→0		
1→0	0→1	0→1	1→0	0→1	212	225.25
1→1	0→0	0→0	1→1	0→0		
1→1	0→0	0→1	1→0	0→1	212	224.75
1→1	0→1	0→0	1→0	0→1	328.25	135.75
1→1	0→1	0→1	1→1	0→1	342.75	146.25
1→0	0→0	1→0	0→0	1→0	269.5	224.5
1→0	0→0	1→1	0→1	1→0	300.75	291
1→0	0→1	1→0	0→1	1→0	219.5	217.5
1→0	0→1	1→1	0→0	1→1		
1→1	0→0	1→0	0→1	1→0	219	226.75
1→1	0→0	1→1	0→0	1→1		
1→1	0→1	1→0	0→0	1→1	330.25	
1→1	0→1	1→1	0→1	1→1	332	
1→0	1→0	0→0	0→0	1→0		**120.25**
1→0	1→0	0→1	0→1	1→0	171	124.75
1→0	1→1	0→0	0→1	1→0	306.25	151.5
1→0	1→1	0→1	0→0	1→1	292.25	294.25
1→1	1→0	0→0	0→1	1→0	361	136.5
1→1	1→0	0→1	0→0	1→1	349.5	347.5
1→1	1→1	0→0	0→0	1→1		
1→1	1→1	0→1	0→1	1→1	171	
1→0	1→0	1→0	1→0	1→0	179.5	125.75
1→0	1→0	1→1	1→1	1→0		120.5
1→0	1→1	1→0	1→1	1→0	179.25	156.75
1→0	1→1	1→1	1→0	1→1	291.5	152
1→1	1→0	1→0	1→1	1→0	**362.75**	142
1→1	1→0	1→1	1→0	1→1	346.75	340.75
1→1	1→1	1→0	1→0	1→1	179.5	
1→1	1→1	1→1	1→1	1→1		

表 4.6: 遅延均等化全加算器の遅延解析

入力			出力		出力 s 側	出力 co 側
a	b	ci	s	co	遅延時間 [ps]	遅延時間 [ps]
0→0	0→0	0→0	0→0	0→0		
0→0	0→0	0→1	0→1	0→0	296.75	
0→0	0→1	0→0	0→1	0→0	293.25	
0→0	0→1	0→1	0→0	0→1		335.75
0→1	0→0	0→0	0→1	0→0	297.5	
0→1	0→0	0→1	0→0	0→1		336.5
0→1	0→1	0→0	0→0	0→1		335
0→1	0→1	0→1	0→1	0→1	296.75	335.5
0→0	0→0	1→0	1→0	0→0	279.25	
0→0	0→0	1→1	1→1	0→0		
0→0	0→1	1→0	1→1	0→0	303	
0→0	0→1	1→1	1→0	0→1	299.25	290.25
0→1	0→0	1→0	1→1	0→0	301.5	
0→1	0→0	1→1	1→0	0→1	304	294.5
0→1	0→1	1→0	1→0	0→1	279.25	336.5
0→1	0→1	1→1	1→1	0→1		334.75
0→0	1→0	0→0	1→0	0→0	**276.5**	
0→0	1→0	0→1	1→1	0→0	280.5	
0→0	1→1	0→0	1→1	0→0		
0→0	1→1	0→1	1→0	0→1	328	325.25
0→1	1→0	0→0	1→1	0→0		
0→1	1→0	0→1	1→0	0→1	328	325.5
0→1	1→1	0→0	1→0	0→1	287.25	325
0→1	1→1	0→1	1→1	0→1	291.5	325.5
0→0	1→0	1→0	0→0	1→0		277.5
0→0	1→0	1→1	0→1	1→0	307	301.25
0→0	1→1	1→0	0→1	1→0	338.5	313.25
0→0	1→1	1→1	0→0	1→1		
0→1	1→0	1→0	0→1	1→0	338.5	313.25
0→1	1→0	1→1	0→0	1→1		
0→1	1→1	1→0	0→0	1→1		
0→1	1→1	1→1	0→1	1→1	317.5	

a	b	ci	s	co	遅延時間 [ps]	遅延時間 [ps]
1→0	0→0	0→0	1→0	0→0	278.75	
1→0	0→0	0→1	1→1	0→0	283	
1→0	0→1	0→0	1→1	0→0		
1→0	0→1	0→1	1→0	0→1	328	325.75
1→1	0→0	0→0	1→1	0→0		
1→1	0→0	0→1	1→0	0→1	328	325.25
1→1	0→1	0→0	1→0	0→1	284.75	322.25
1→1	0→1	0→1	1→1	0→1	290.5	323.5
1→0	0→0	1→0	0→0	1→0		278.5
1→0	0→0	1→1	0→1	1→0	309.5	303.25
1→0	0→1	1→0	0→1	1→0	**338.75**	313
1→0	0→1	1→1	0→0	1→1		
1→1	0→0	1→0	0→1	1→0	338.5	313.25
1→1	0→0	1→1	0→0	1→1		
1→1	0→1	1→0	0→0	1→1		
1→1	0→1	1→1	0→1	1→1	315.5	
1→0	1→0	0→0	0→0	1→0		297.5
1→0	1→0	0→1	0→1	1→0	297	297.5
1→0	1→1	0→0	0→1	1→0	310	327.75
1→0	1→1	0→1	0→0	1→1		
1→1	1→0	0→0	0→1	1→0	312.75	320.25
1→1	1→0	0→1	0→0	1→1		
1→1	1→1	0→0	0→0	1→1		
1→1	1→1	0→1	0→1	1→1	296.75	
1→0	1→0	1→0	1→0	1→0	279.25	298.25
1→0	1→0	1→1	1→1	1→0		297.5
1→0	1→1	1→0	1→1	1→0	296.75	328.75
1→0	1→1	1→1	1→0	1→1	316	
1→1	1→0	1→0	1→1	1→0	295.75	321.75
1→1	1→0	1→1	1→0	1→1	318.75	
1→1	1→1	1→0	1→0	1→1	279.25	
1→1	1→1	1→1	1→1	1→1		

4.3.4 ラッチの動作確認

HSpice(level49) を用いてラッチの動作確認を行う。図 4.16 に示すように入力側には全加算器出力と同等の駆動能力を持つインバータ、出力側には全加算器入力と同等の負荷を持つインバータを接続してシミュレーションを行う。

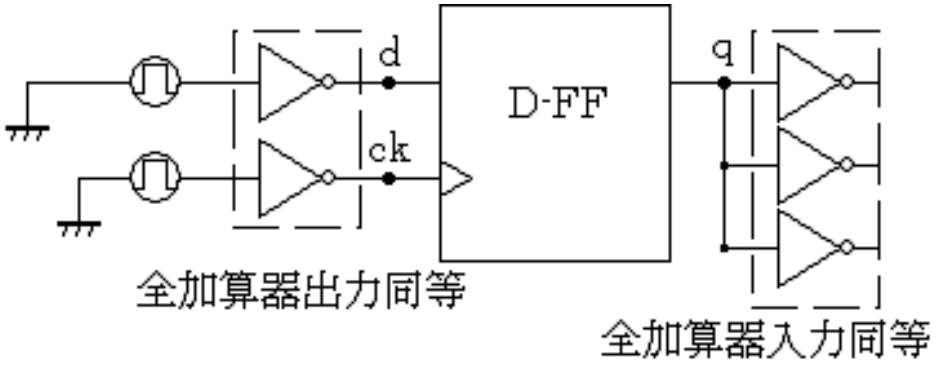


図 4.16: ラッチの動作確認

ラッチのトグルサイクル t_w の確認を図 4.17 に示す。デューティー比 50 パーセントのクロックサイクルを用いた。130ps サイクルでは、期待している出力波形が得られない。ラッチのトグルサイクルは 140ps が限界である。

ラッチのセットアップタイム t_s の確認を図 4.18 に示す。セットアップタイム 20ps では十分に値を取り込むことができないため、期待している出力波形が得られないセットアップタイムは 25ps 以上必要である。

ラッチのホールドタイム t_h の確認を図 4.19 に示す。ホールドタイム 0ps では、期待している出力波形が得られない。十分な波形を得るためには、ホールドタイムは 10ps 以上必要である。

ラッチの遅延時間を図 4.20 に示す。ラッチの遅延時間は、入力波形の状態による変動もあるが、マスター側は 57.5ps, スレーブ側 87.5ps である。

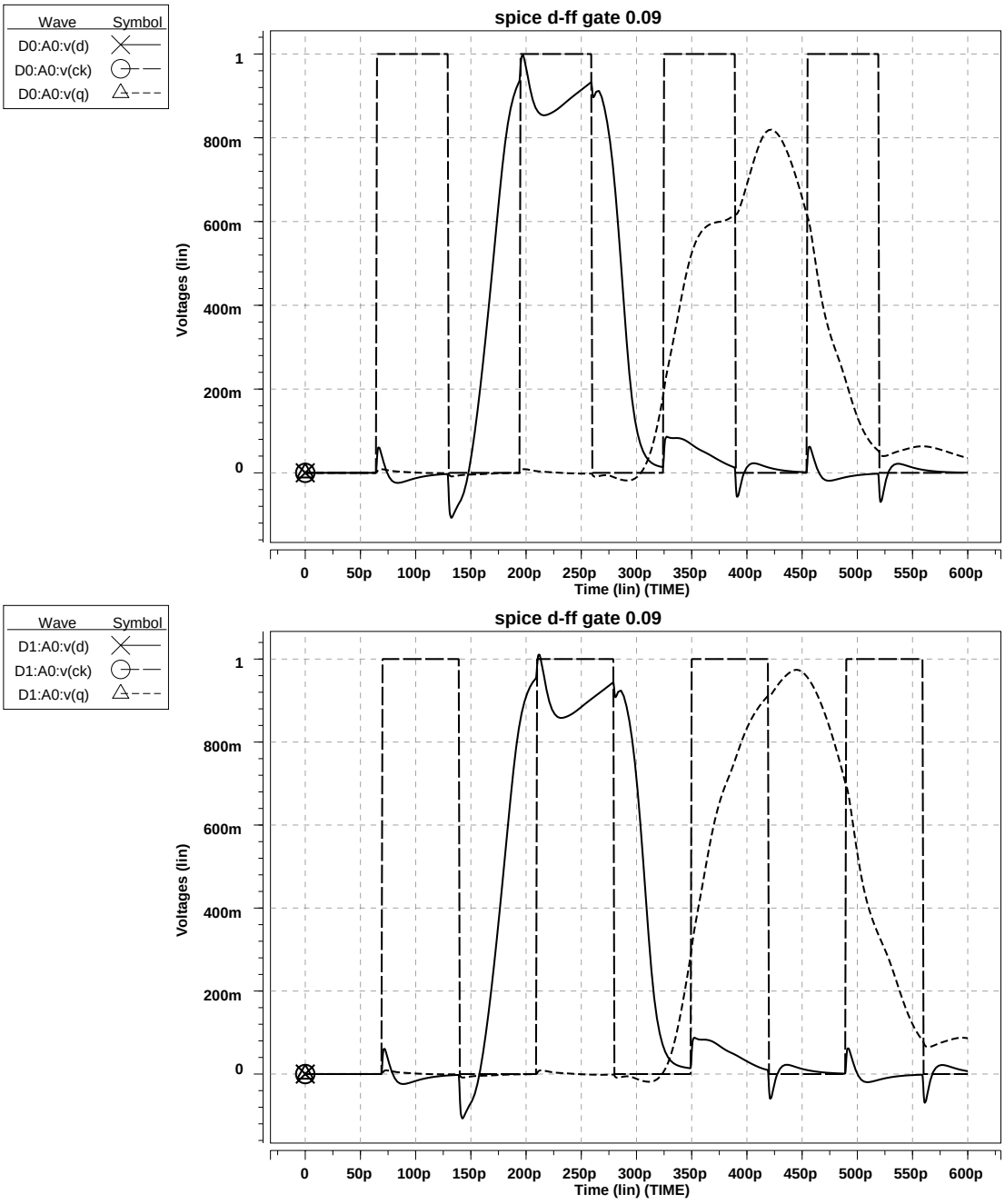


図 4.17: トグルサイクルの確認 (上;t_g=130ps, 下;t_g=140ps)

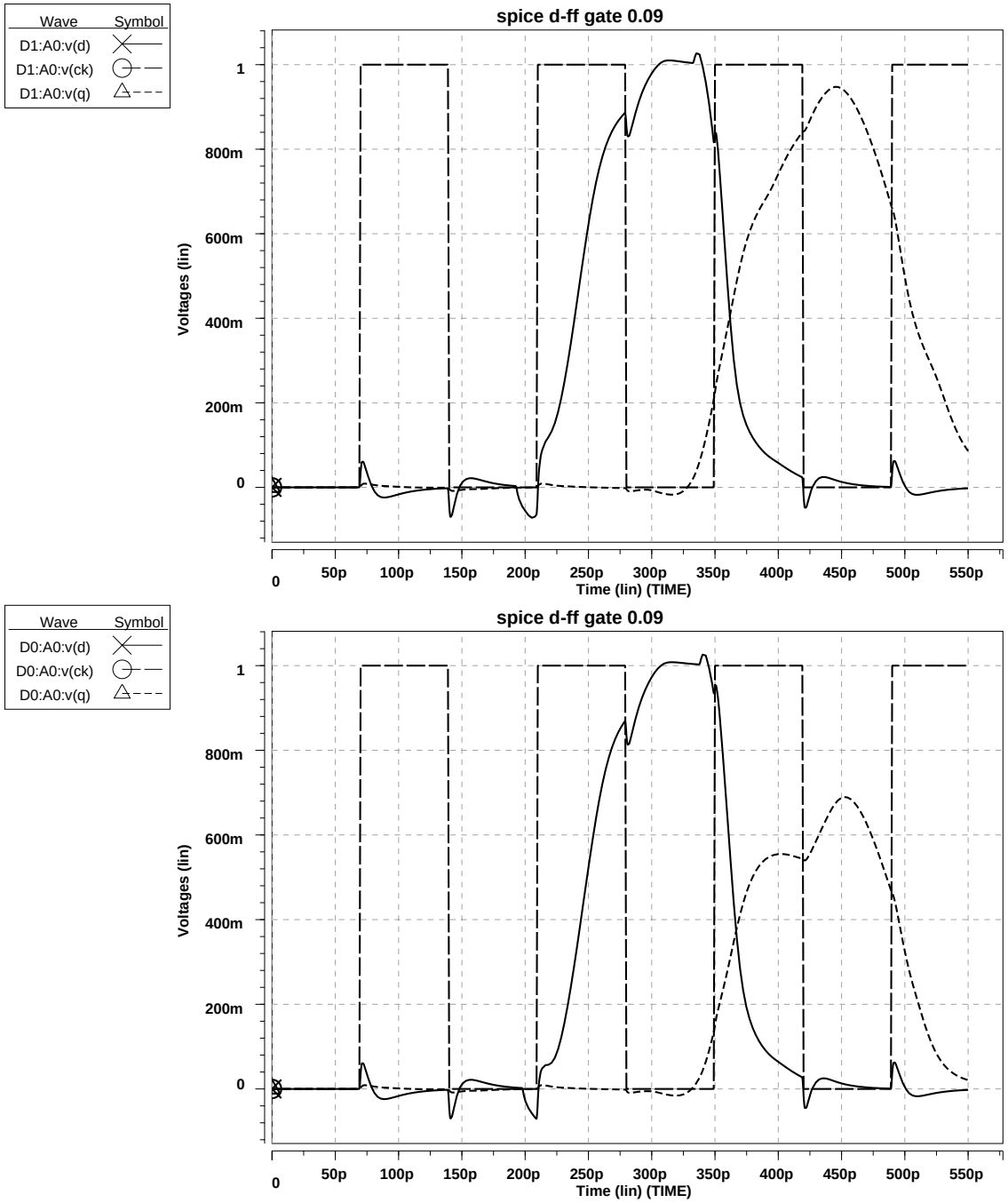


図 4.18: セットアップタイムの確認 (上;t_s=25ps, 下;t_s=20ps)

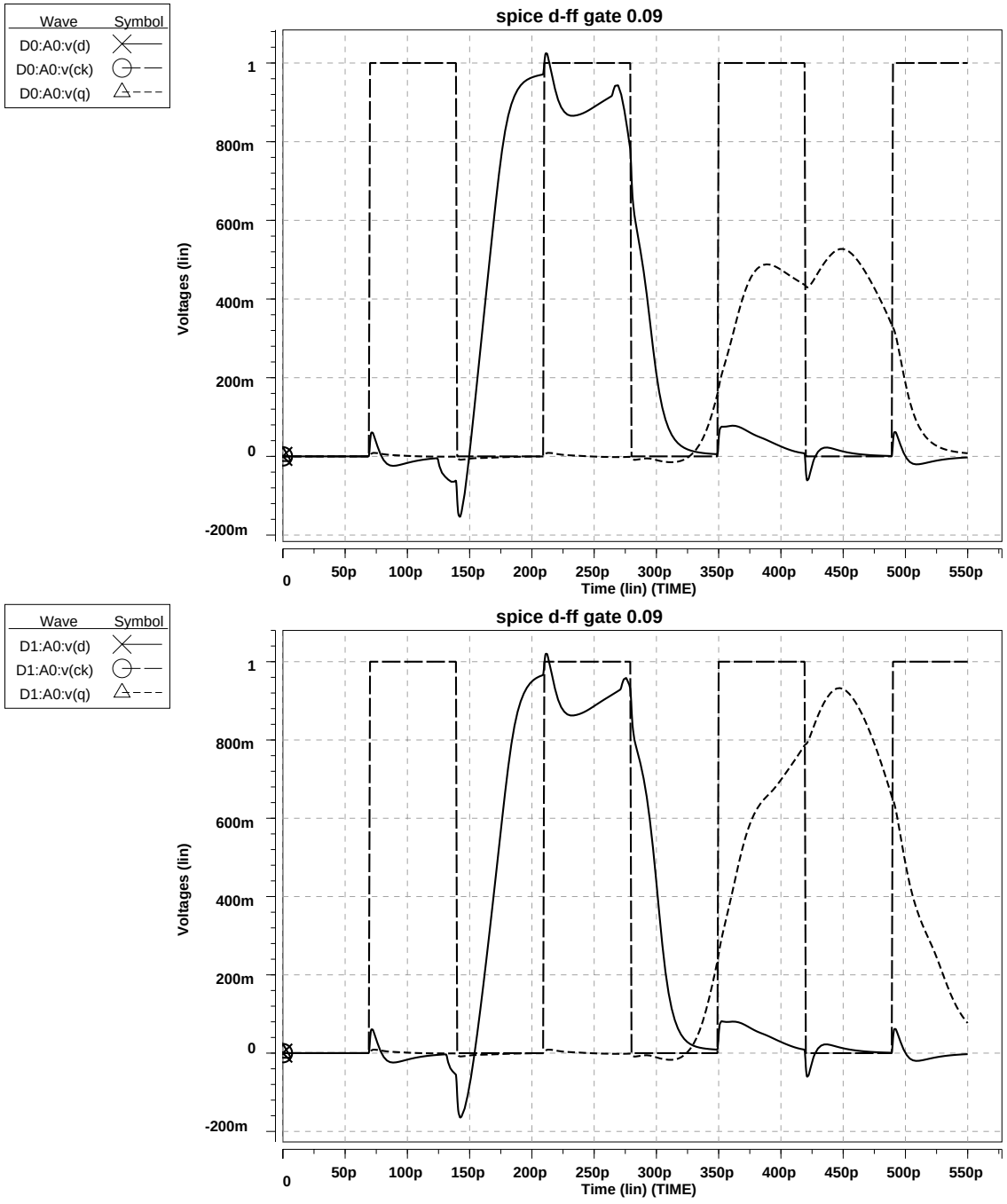


図 4.19: ホールドタイムの確認 (上;t_h=0ps, 下;t_h=10ps)

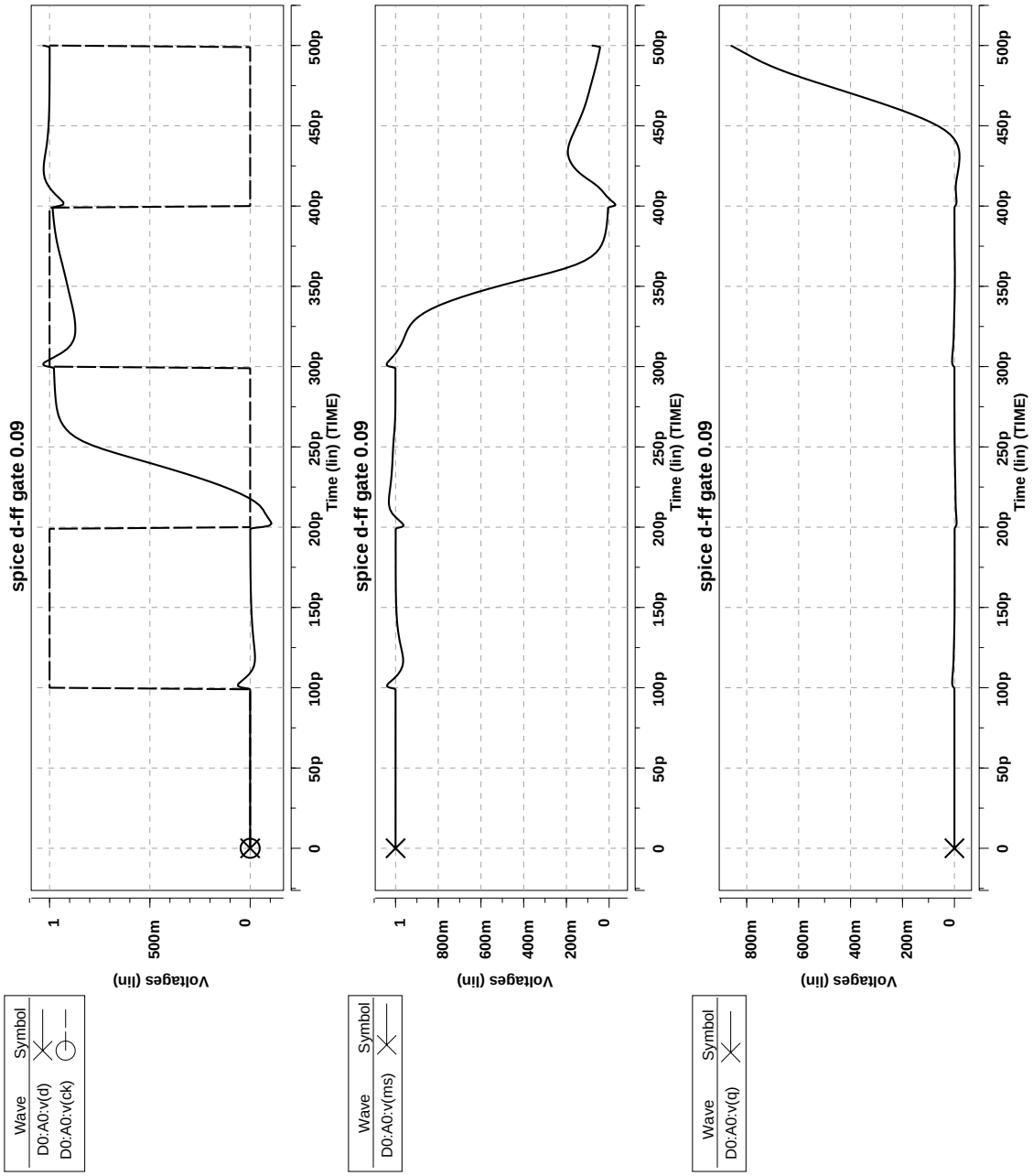


図 4.20: ラッチの遅延時間 (上から;d, ck; ms(マスタースレーブ間); q)

4.3.5　クロックサイクルの予想

全加算器の遅延解析とラッチの動作確認の結果を用いてパイプラインのクロックサイクルの予想をする。図 4.21 にラッチと組み合わせ回路のクロックサイクルを示す。マスタスレーブのラッチではマスター側はデータを取り込むための部屋であるため、セットアップとホールドタイムを満たす必要がある。スレーブ側はネガティブエッジでデータを吐き出す部屋であるため、その遅延時間 (ネガティブエッジからデータが出力されるまでの時間) を考慮してクロックサイクルを決める必要がある。

同期パイプライン方式のクロックサイクルは、全段のラッチのスレーブ側の遅延時間 t_{ds} と組み合わせ回路の最大遅延時間 t_{max}、後段のラッチのマスター側のセットアップタイム t_s とホールドタイム t_h の和で決まる。同期パイプライン方式での全加算器のクロックサイクルは、485.25ps(87.5+362.75+25+10) となる。

ウェーブパイプライン方式の全加算器のクロックサイクルは、組み合わせ回路による遅延時間の部分が最大遅延時間と最小遅延時間との差に変わるだけで、あとは同様にラッチによるオーバーヘッドを加算することによりもとめられる。クロックサイクルは、184.75ps(87.5+62.25+25+10) と予想される。

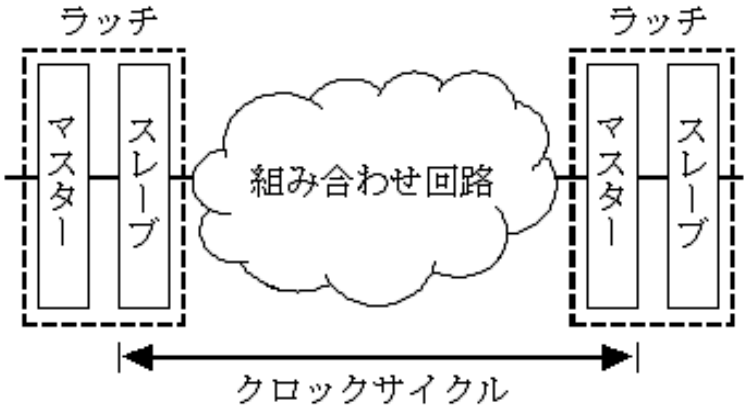


図 4.21: クロックサイクル

4.3.6　遅延均等化全加算器のラッチシミュレーション

HSpice(level49) を用いて、ラッチを含めた遅延均等化全加算器のシミュレーションを行う (図 4.22, 4.23 参照)。結果を図 4.24〜4.27 に示す。遅延均等化全加算器の前後のラッチが同相 (単相クロック) では、クロックサイクル 220ps(4.54GHz) が限界となる。多相クロックを用いると、クロックサイクル 200ps(5GHz) が限界となる。この場合、後ろのラッチのクロック ck は前のラッチのクロック cl より 50ps 遅くれたクロックを使用している。これ以上クロックサイクルを短くすると、全加算器の出力波形が立ち上がりまたは立ち下

がり切らず、ラッチすることができない。図 4.29(図 4.28 の信号の様子参照) に 190ps サイクルで動作させた全加算器の波形を示す。全加算器の出力 co の波形が立ち上がり切れていないため、ラッチすることができない。次の 3 つの原因が考えられる。

1) セットアップやホールドタイムの見積もりのあまさ

2) ファンアウトが多い論理ゲートの出力波形は完全に立ち上がり切れていない。

3)2 入力の論理ゲートでは 2 つの信号の到着がずれると、波形の幅が小さくなる。

さらに、高クロックで動作を望むなら、1) 全体的に MOS のサイズを大きくする、2) 共通電源を上げる、3) 最先端の配線材料 (低誘電率の絶縁膜) の使用、4) 最適な MOS パラメータの選定など、あるが本研究では論じない。

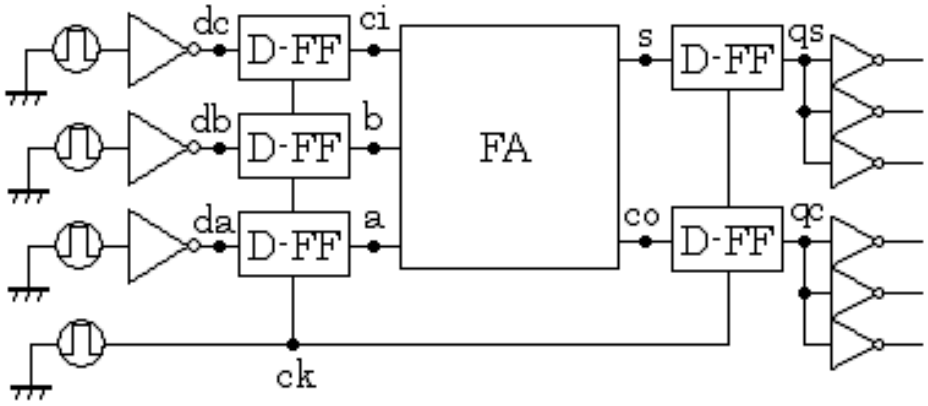


図 4.22: 全加算器の単相クロックラッチシミュレーション

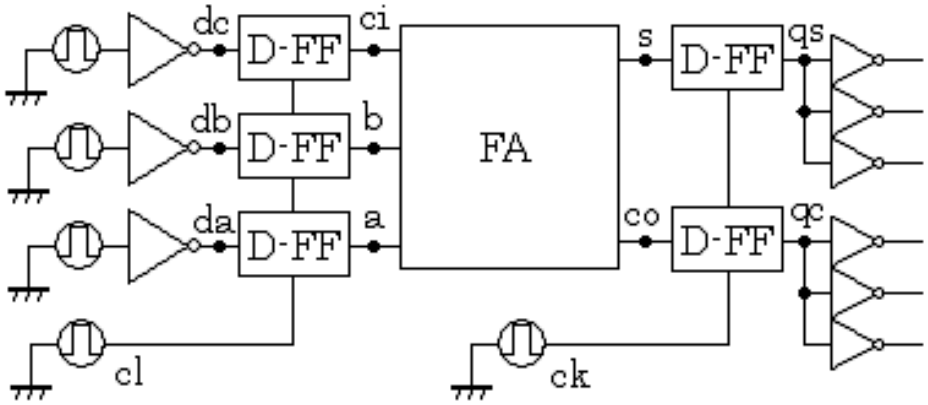


図 4.23: 全加算器の多相クロックラッチシミュレーション

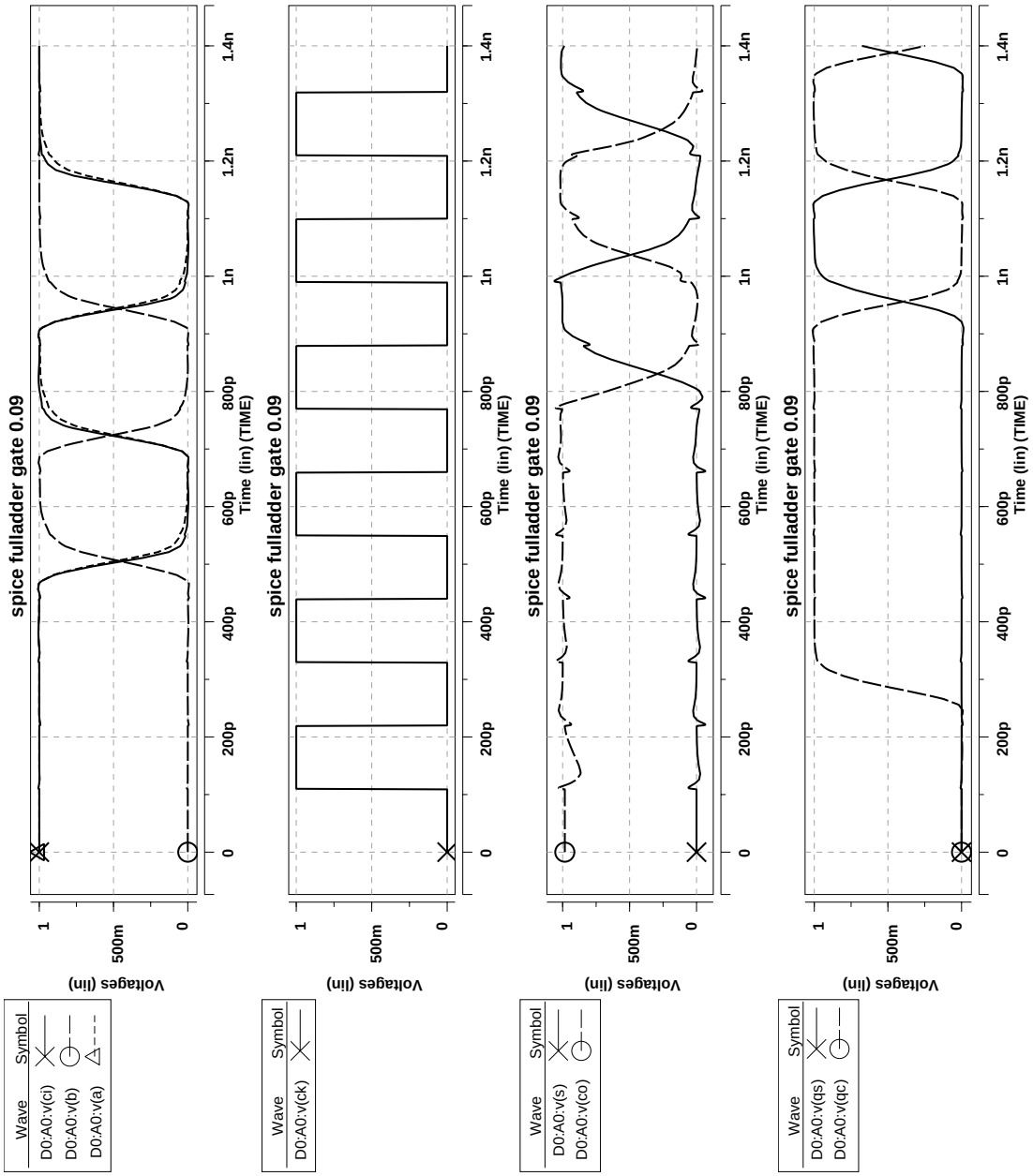


図 4.24: 全加算器の単相クロックラッチシミュレーション (上から; ci, b, a; ck; s, co; qs, qc) 最大遅延の入力変化 (a,b,ci,s,co)=(1,0,1,0,1)→(0,1,0,1,0)

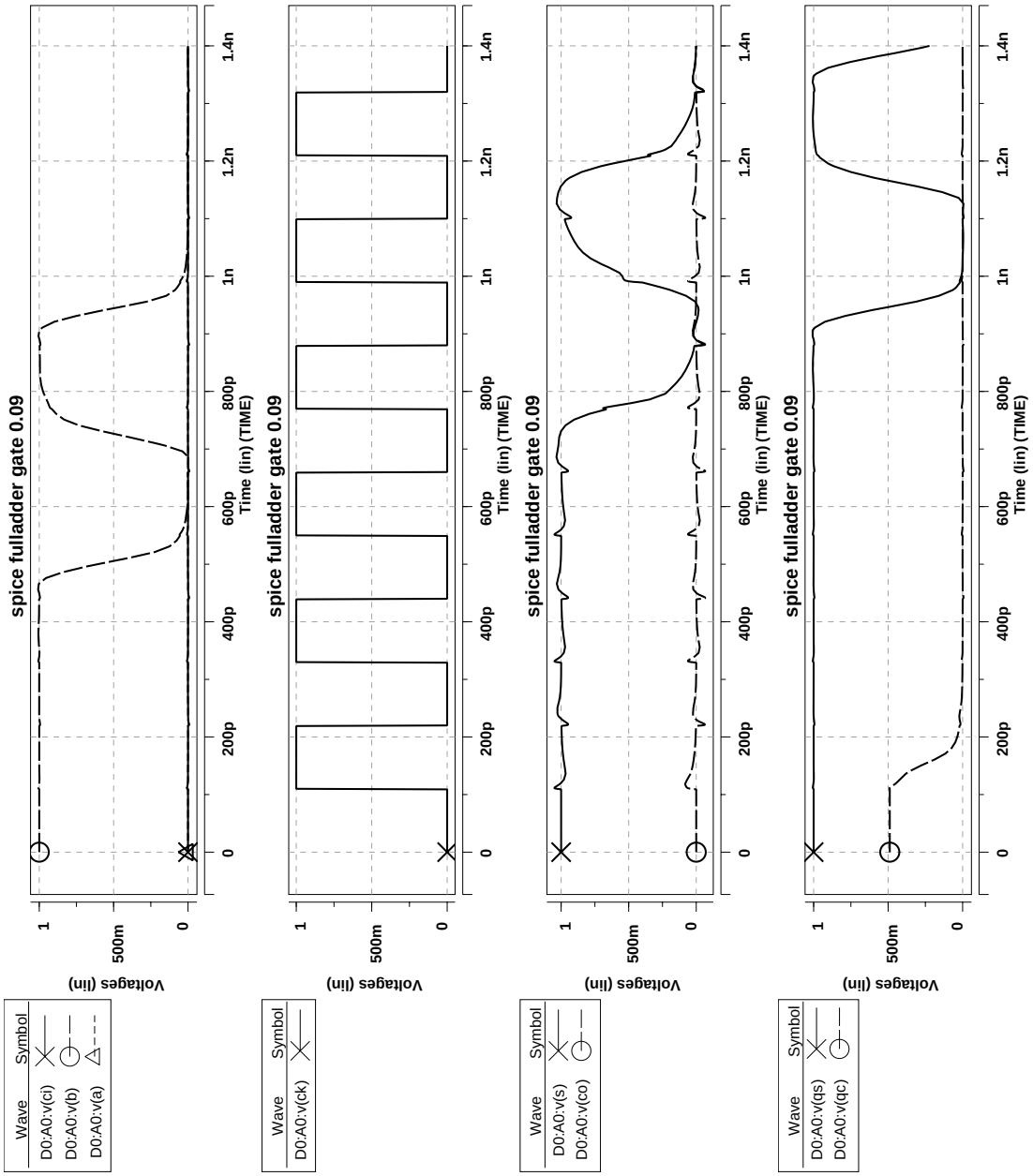


図 4.25: 全加算器の単相クロックラッチシミュレーション (上から; ci, b, a; ck; s, co; qs, qc) 最小遅延の入力変化 (a,b,ci,s,co)=(0,1,0,1,0)→(0,0,0,0,0)

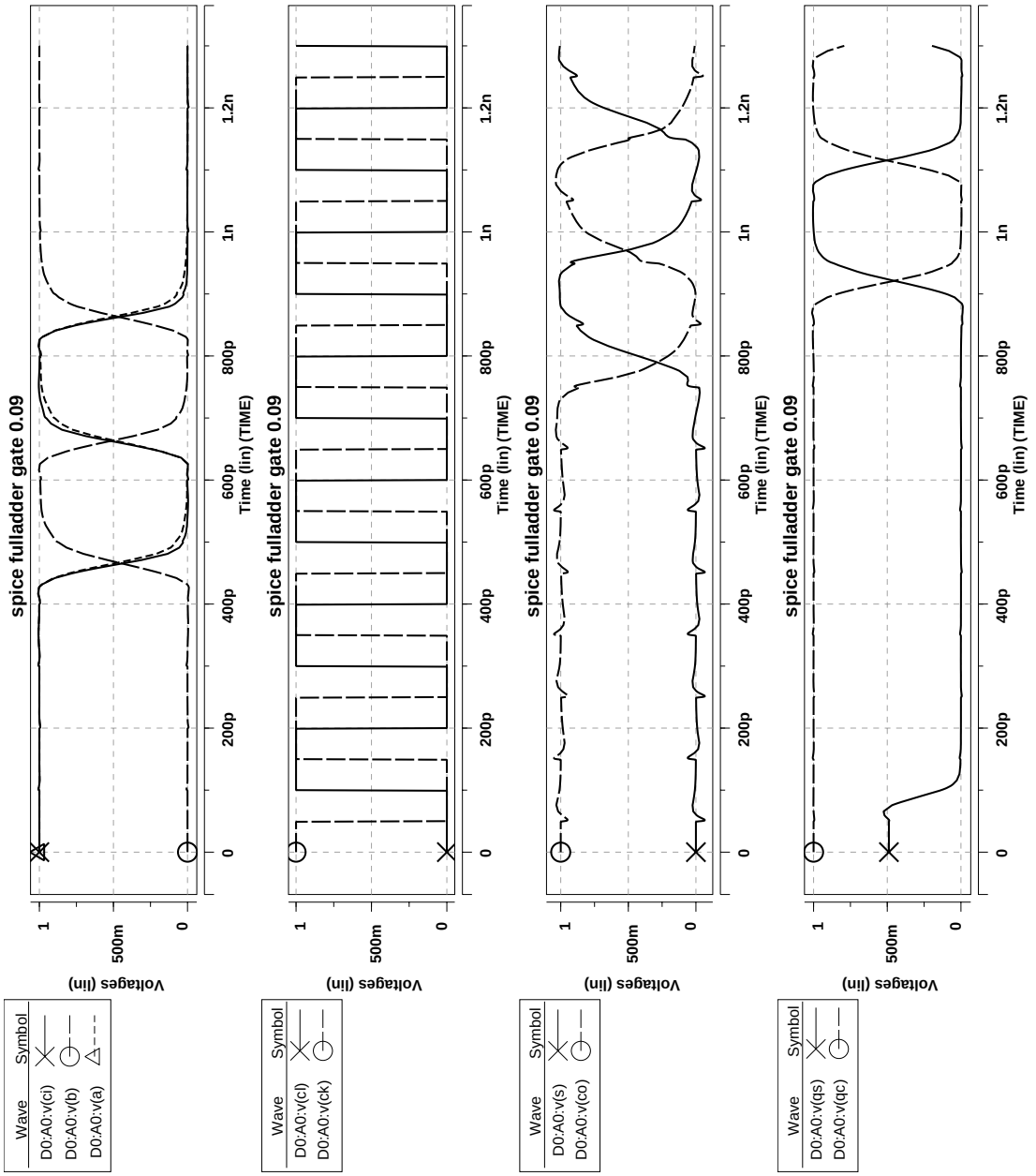


図 4.26: 全加算器の多相クロックラッチシミュレーション (上から; ci, b, a; cl, ck; s, co; qs, qc)　最大遅延の入力変化 (a,b,ci,s,co)=(1,0,1,0,1)→(0,1,0,1,0)

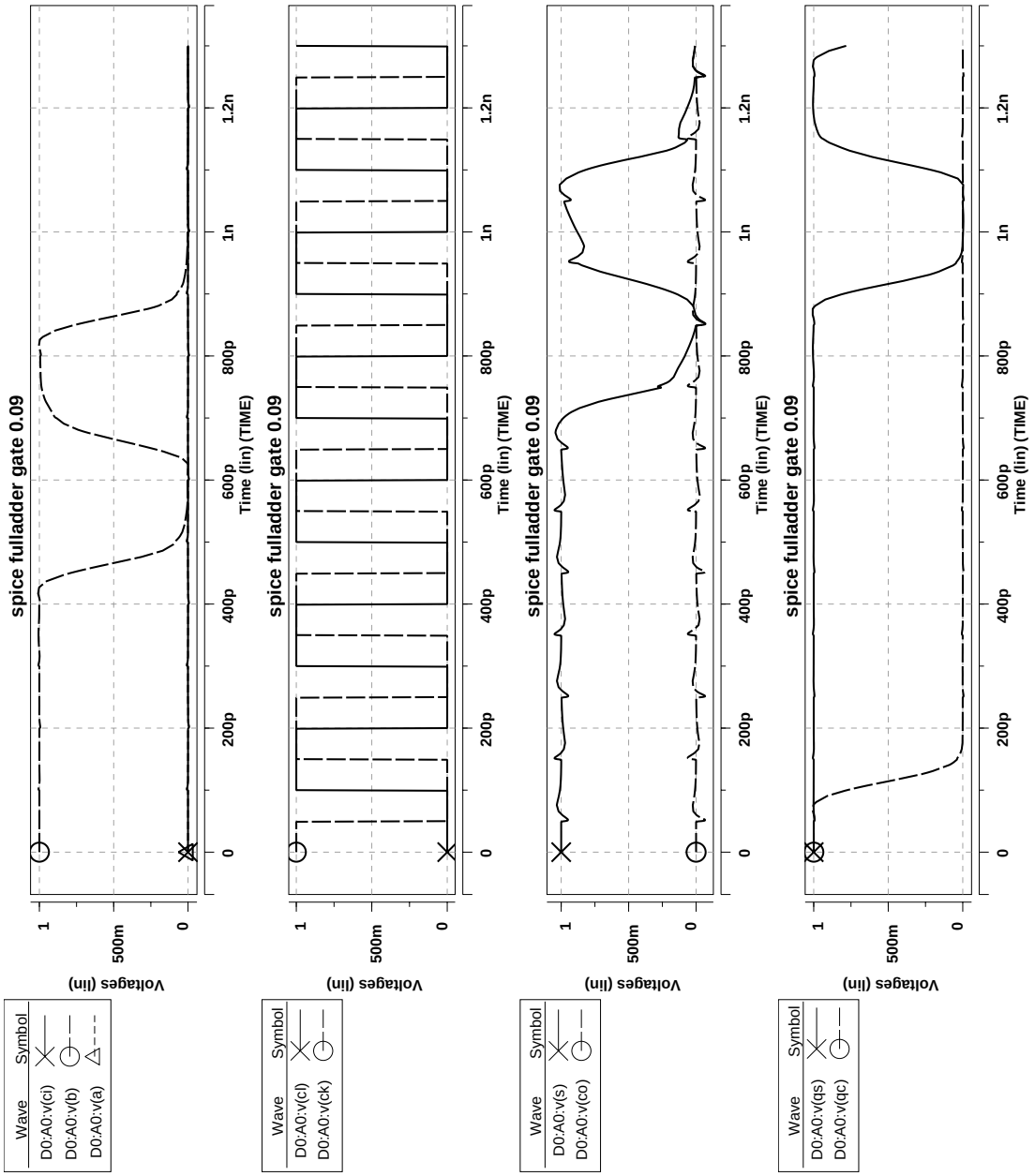


図 4.27: 全加算器の多相クロックラッチシミュレーション (上から; ci, b, a; cl, ck; s, co; qs, qc)　最小遅延の入力変化 (a,b,ci,s,co)=(0,1,0,1,0)→(0,0,0,0,0)

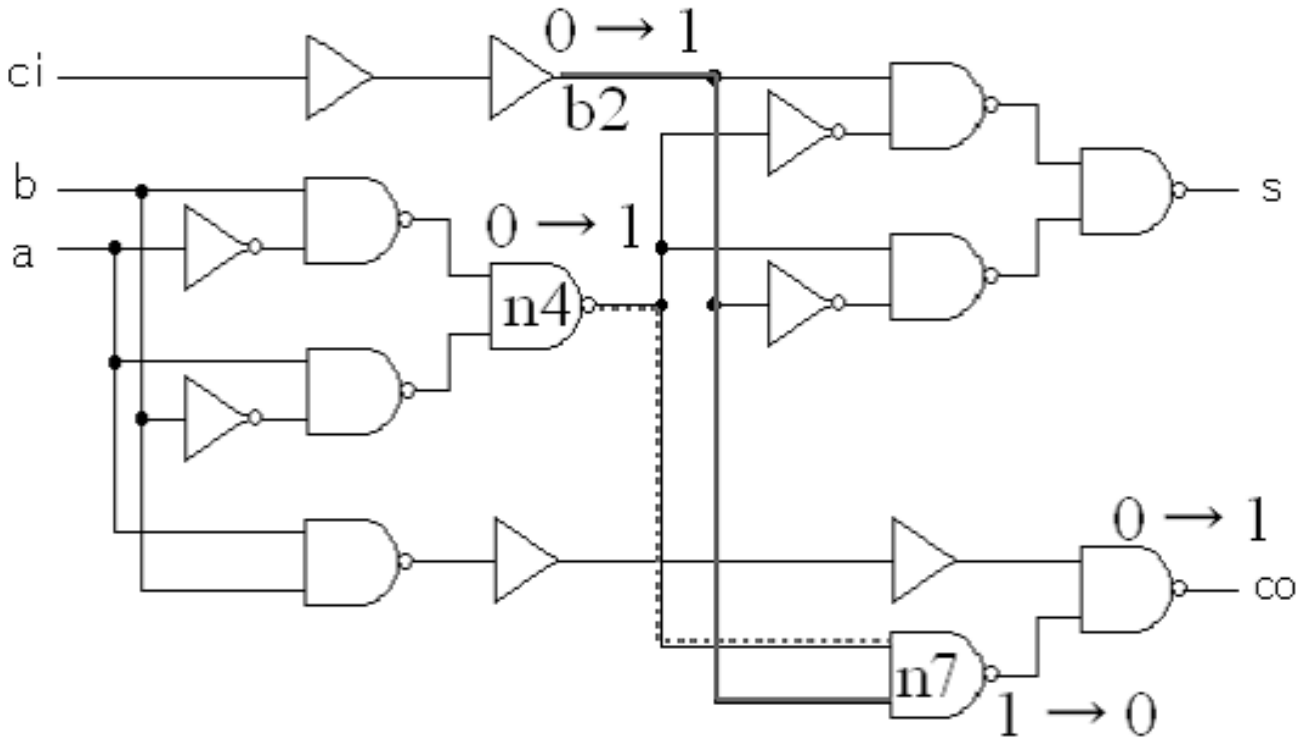


図 4.28: 190ps サイクル動作の全加算器内部の信号の様子

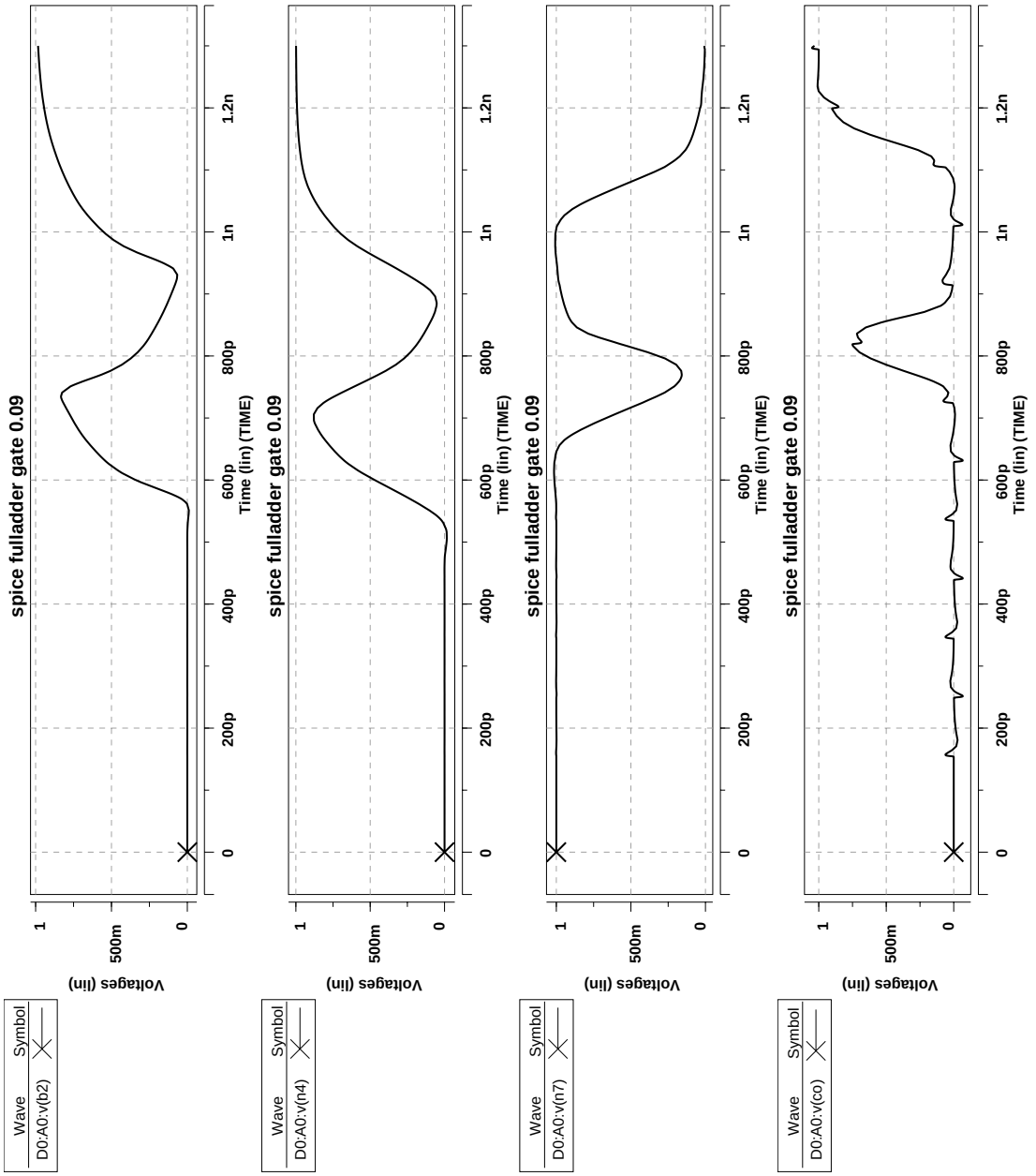


図 4.29: 全加算器 190ps サイクル動作 (上から; b2; n4; n7; co)

4.4　ウェーブパイプライン用累算用加算器

積和演算器は最終段にフィードバックを持つ累算加算器があり、多相クロックを用いた場合では前後のラッチのタイミングを合わせる必要がある。図 4.30 に多相クロックのウェーブパイプライン用の累算加算器を示す。全加算器後方に delay を入れ、前後のラッチのクロックのタイミングを合わせる (図 4.31 参照)。delay には遅延バッファや D-FF を用いればよい。これにより、全加算器前後のラッチを同相で動作させることができる。ウェーブパイプライン化により全加算器 (1 ステージ) の中に複数のデータが存在することになる。フィードバック機構が 1 つでは、累算が困難である。ウェーブパイプライン化されたデータを累算するために、2 つのフィードバック機構を設けた。これにより、多相クロックの積和演算器はクロック周波数 5GHz で動作することができる。

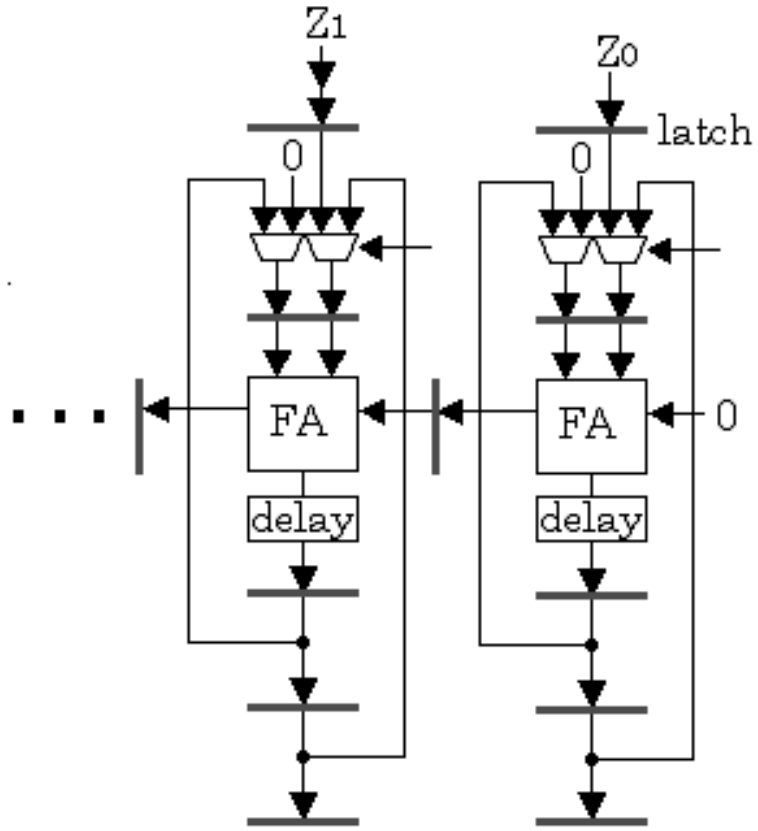


図 4.30: 多相クロックによるウェーブパイプライン用累算加算器

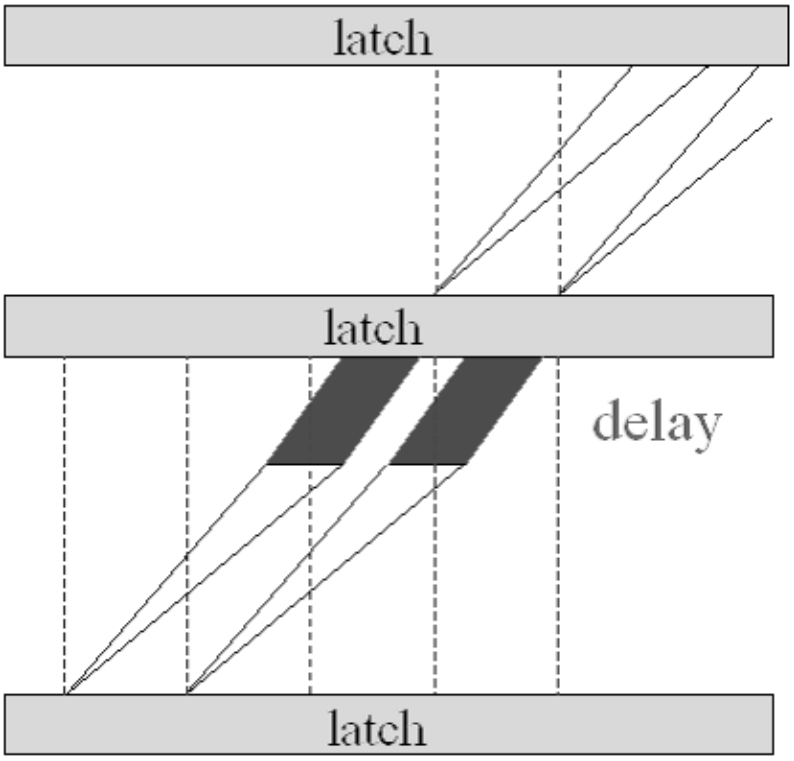


図 4.31: delay によるクロックのタイミング合わせ

第5章　結論

第3章では、簡単な回路構成の積和演算器として、シストリックアレイを用いた積和演算器の構成を示した。シフトレジスタを用いて部分積生成のタイミングを遅らせることにより乗算器の部分積加算部分では、キャリー伝搬遅延が生じない。しかし、多くのシフトレジスタを使用するため、従来のCLA回路を用いた積和演算器よりも、2割増しの面積となる。

第4章では、全加算器の遅延均等化とラッチの設計を行い、HSpiceを用いて全加算器の遅延解析とラッチの動作確認を行った。表5.1と表5.2に結果をまとめる。

表 5.1: 設計した全加算器の遅延時間と面積

	遅延均等化前	遅延均等化後
最大遅延時間 t_{max}[ps]	362.75	338.75
最小遅延時間 t_{min}[ps]	120.25	276.5
最大と最小の遅延差 t_{di}[ps]	242.5	62.25
使用MOS数[個]	44	60
レイアウト面積 $S[\mu m^2]$	48.86	55.34

表 5.2: 設計したマスタースレーブラッチの性能

トグルサイクル t_g[ps]	140
セットアップタイム t_s[ps]	25
ホールドタイム t_h[ps]	10
スレーブ側の遅延時間 t_{ds}[ps]	87.5
使用MOS数[個]	16
レイアウト面積 $S[\mu m^2]$	21.12

最後に遅延を均等化した全加算器の前後にラッチを付け、ウェーブパイプライン動作の確認をした。HSpiceのシミュレーション結果より、ウェーブパイプライン方式の全加算器はクロック周波数5GHz(200ps)まで動作可能である。理論上では、クロックサイクルは

約 185ps ($t_{ds}+t_{di}+t_s+t_h$) となり、15ps の差がある。この差の原因には、理論値 185ps 中のラッチのセットアップタイムやホールドタイムの見積もりのあまさなどが考えられる。ウェーブパイプライン化により、同期パイプライン方式を用いた全加算器 (2.06GHz) の 2.4 倍のクロック周波数となった。

本研究のシストリックアレイを用いた積和演算器は、すべて全加算器により構成されている (部分積生成部分を除く)。処理要素の単位が全加算器であることから、全加算器以上はクロック周波数を上げることができない。よって、多相クロックを用いた積和演算器のクロック周波数の限界は 5GHz となる。

シストリックアレイとウェーブパイプラインは、回路のウェーブパイプライン化のし易さでは相性がよい。しかし、面積では、シストリックアレイは冗長であり、さらにウェーブパイプライン化による面積の増加が伴ってしまう。面積の増加には、遅延バッファとシフトレジスタによるものがある。全加算器のウェーブ度が 2 の場合、キャリー伝搬遅延が生じないように部分積を部分積加算回路に入れるには、2 クロックずつタイミングを遅らせる必要がある。つまり、ウェーブ度を 2 にすると、シストリックアレイで使用するシフトレジスタは 2 倍のビット数必要となる。ウェーブ度 3 の場合は 3 倍のシフトレジスタが必要である。シフトレジスタが積和演算器の面積に占める割合が大きいほど、ウェーブ度増加による面積増加が大きいことになる。次式にシストリックアレイ積和演算器のシフトレジスタとその他 (部分積生成、部分積加算、累算加算、パイプラインラッチ) の使用 MOS 数の関係を示す。

$$n^2 \times p_r + [n^2 + 4n] \times f_a + [3n^2 + 8n] \times d_f > [(2n)^2/2 + (4n)^2/2] \times d_f \tag{5.1}$$

なお、n が乗数である。p_r が AND ゲートの MOS 数 (6 個)、f_a が全加算器の MOS 数 (60 個)、d_f がマスタースレーブラッチの MOS 数 (16 個) である。左辺の n^2 は乗算器部分の AND ゲートの数、n^2 は乗算器部分の全加算器の数、$4n$ は累算加算器の全加算器の数、$3n^2$ は乗算器のパイプラインラッチの数、$8n$ は累算加算器のパイプラインラッチの数を表している。この式は、累算加算器に乗数の 4 倍のビット分用意した積和演算器である。右辺の $(2n)^2/2$ は乗数と被乗数用のシフトレジスタに必要なラッチの数、$(4n)^2/2$ は積和演算器出力の桁合わせ用のシフトレジスタに必要なラッチの数を表している。この式は n が 7 まで成り立つ。つまり、8×8 ビット乗算からは、シフトレジスタが積和演算器の MOS 数の半数以上を占めることを意味している。次式は乗数の 5 倍のビット数の累算用加算器を持つ積和演算器である。

$$n^2 \times p_r + [n^2 + 5n] \times f_a + [3n^2 + 10n] \times d_f > [(2n)^2/2 + (5n)^2/2] \times d_f \tag{5.2}$$

式は n が 3 まで成り立つ。3×3 ビット乗算からは、シフトレジスタが積和演算器の MOS 数の半数以上を占めることを意味している。このように、累算加算器のビット数によってシフトレジスタの割合が変化する。表 5.3 に積和演算器の MOS 数の比較を示す。ウェーブ度 2 のシストリックアレイ積和演算器の MOS 数が従来の積和演算器の 2 倍の MOS 数より小さい値となれば有効である。

表 5.3: 積和演算器の MOS 数の比較

	従来の CLA 回路使用	シストリックアレイ使用	シストリックアレイウェーブ度 2
4 × 4 乗算累算加算器 2n ビット	2464	3072	4480
4 × 4 乗算累算加算器 4n ビット	3940	5088	8160
8 × 8 乗算累算加算器 2n ビット	9108	11328	16704
8 × 8 乗算累算加算器 4n ビット	13500	18432	30208

よって、この積和演算器の有効範囲は狭い。

本研究では積和演算器の高スループット化を目的とし、積和演算器のクロック周波数の限界を探った。

参考文献

[1] 日比野靖. マルチスレッド型超パイプラインプロセッサアーキテクチャ. 平成 10〜12 年度科研費成果報告 B210480058,2002.

[2] JOHN McCANNY.Systolic Array Processors.PRENTICE HALL,1989.

[3] BSIM3 Homepage,http://www-device.eecs.berkeley.edu/∼bsim3/.

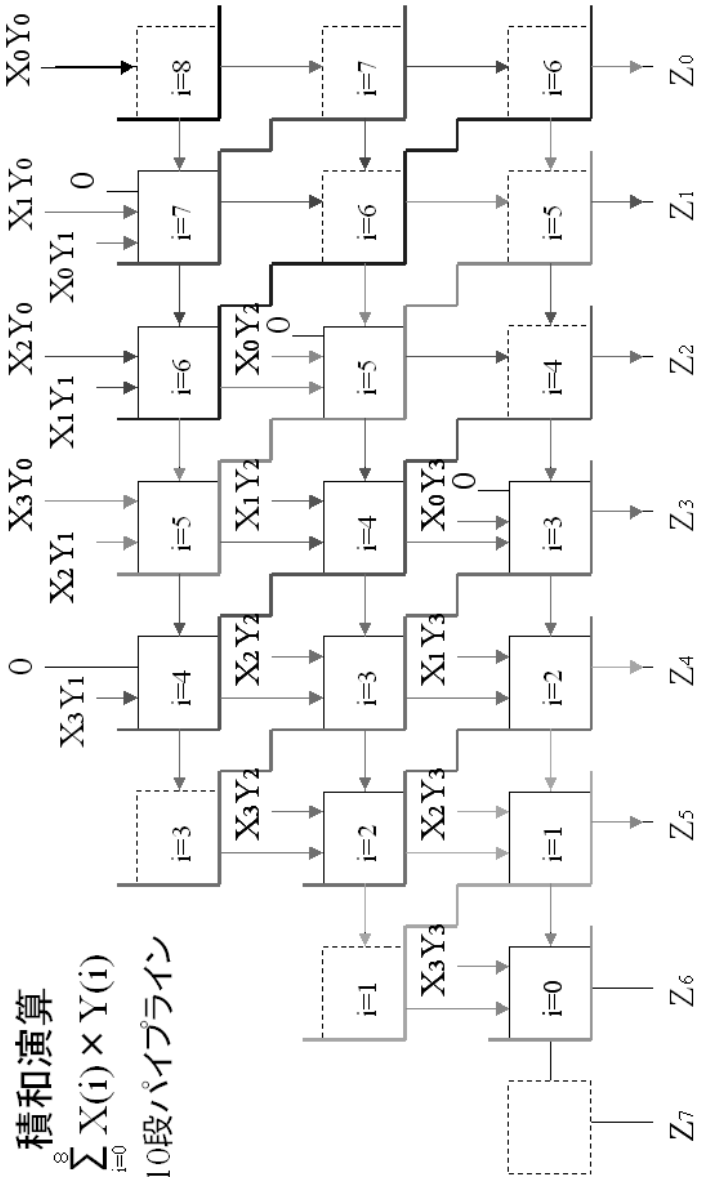
[4] 榎本忠儀.CMOS 集積回路. 培風館.pp.79-83,1996.

[5] Neil H.E.Weste and Kamran Eshraghian. CMOS VLSI 設計の原理. 丸善株式会社,1988.

[6] 飯塚哲哉.CMOS 超 LSI の設計. 培風館.pp.141-144,1989.

[7] 鳥谷部達.MOSFET のモデリングと BSIM3 ユーザーズガイド. 丸善株式会社,2002.

付録 A



4 × 4 シストリックアレイ部分積加算回路の積和演算データの流れ

付録B

90nmMOSトランジスタのパラメータ

* Predictive Technology Model Beta Version
90nm NMOS SPICE Parametersv (normal one)

.model NMOS NMOS +Level = 49
+Lint = 1.5e-08 Tox = 2.5e-09 +Vth0 = 0.2607 Rdsw = 180
+lmin=1.0e-7 lmax=1.0e-7 wmin=1.0e-7 wmax=1.0e-4 +Tref=27.0 version =3.1 +Xj= 4.0000000E-08 Nch= 9.7000000E+17 +lln= 1.0000000 lwn= 1.0000000 wln= 0.00 +wwn= 0.00 ll= 0.00 +lw= 0.00 lwl= 0.00 wint= 0.00 +wl= 0.00 ww= 0.00 wwl= 0.00 +Mobmod= 1 binunit= 2 xl= 0.00 +xw= 0.00 binflag= 0 +Dwg= 0.00 Dwb= 0.00
+ACM= 0 ldif=0.00 hdif=0.00 +rsh= 7 rd= 0 rs= 0 +rsc= 0 rdc= 0
+K1= 0.3950000 K2= 1.0000000E-02 K3= 0.00 +Dvt0= 1.0000000 Dvt1= 0.4000000 Dvt2= 0.1500000 +Dvt0w= 0.00 Dvt1w= 0.00 Dvt2w= 0.00 +Nlx= 4.8000000E-08 W0= 0.00 K3b= 0.00 +Ngate= 5.0000000E+20
+Vsat= 1.1000000E+05 Ua= -6.0000000E-10 Ub= 8.0000000E-19 +Uc= -2.9999999E-11 +Prwb= 0.00 Prwg= 0.00 Wr= 1.0000000 +U0= 1.7999999E-02 A0= 1.1000000 Keta= 4.0000000E-02 +A1= 0.00 A2= 1.0000000 Ags= -1.0000000E-02 +B0= 0.00 B1= 0.00
+Voff= -2.9999999E-02 NFactor= 1.5000000 Cit= 0.00 +Cdsc= 0.00 Cdscb= 0.00 Cdscd= 0.00 +Eta0= 0.1500000 Etab= 0.00 Dsub= 0.6000000
+Pclm= 0.1000000 Pdiblc1= 1.2000000E-02 Pdiblc2= 7.5000000E-03 +Pdiblcb= -1.3500000E-02 Drout= 2.0000000 Pscbe1= 8.6600000E+08 +Pscbe2= 1.0000000E-20 Pvag= -0.2800000 Delta= 1.0000000E-02 +Alpha0= 0.00 Beta0= 30.0000000
+kt1= -0.3700000 kt2= -4.0000000E-02 At= 5.5000000E+04 +Ute= -1.4800000 Ua1= 9.5829000E-10 Ub1= -3.3473000E-19 +Uc1= 0.00 Kt1l= 4.0000000E-09 Prt= 0.00
+Cj= 0.0015 Mj= 0.72 Pb= 1.25 +Cjsw= 2E-10 Mjsw= 0.37 Php= 0.773 +Cjgate= 2E-14 Cta= 0 Ctp= 0 +Pta= 0 Ptp= 0 JS=1.50E-08 +JSW=2.50E-13 N=1.0 Xti=3.0 +Cgdo=3.493E-10 Cgso=3.493E-10 Cgbo=0.0E+00 +Capmod= 2 NQSMOD= 0 Elm= 5 +Xpart= 1 cgsl= 0.582E-10 cgdl= 0.582E-10 +ckappa= 0.28 cf= 1.177e-10 clc= 1.0000000E-07 +cle= 0.6000000 Dlc= 2E-08 Dwc= 0

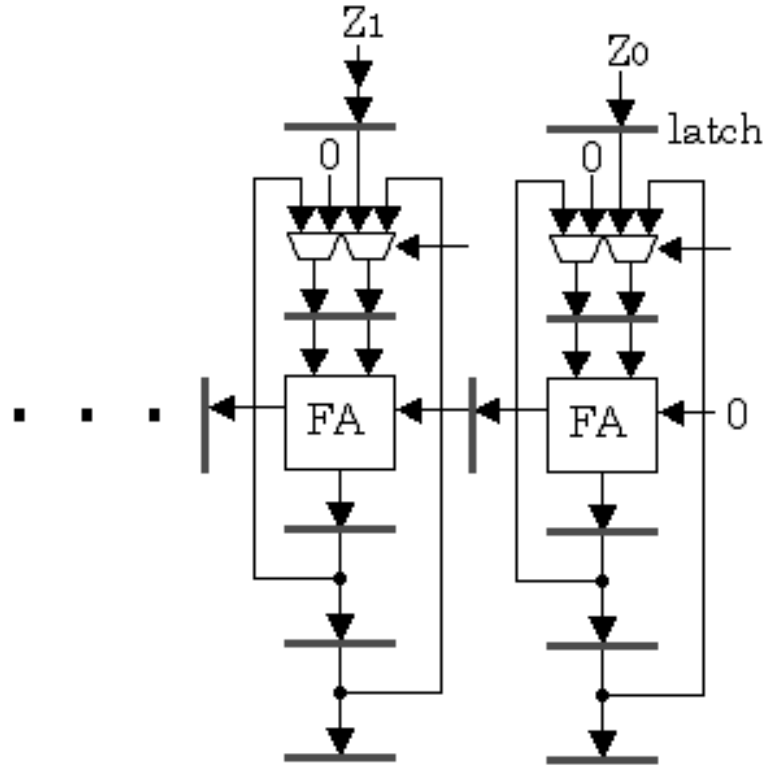
* * Predictive Technology Model Beta Version
90nm PMOS SPICE Parametersv (normal one)

.model PMOS PMOS +Level = 49
+Lint = 1.5e-08 Tox = 2.5e-09 +Vth0 = -0.303 Rdsw = 300
+lmin=1.0e-7 lmax=1.0e-7 wmin=1.0e-7 wmax=1.0e-4 +Tref=27.0 version =3.1 +Xj= 4.0000000E-08 Nch= 1.0400000E+18 +lln= 1.0000000 lwn= 0.00 wln= 0.00 +wwn= 1.0000000 ll= 0.00 lw= 0.00 +lwl= 0.00 wint= 0.00 wl= 0.00 +ww= 0.00 wwl= 0.00 Mobmod= 1 +binunit= 2 xl= 0.00 xw= 0.00 +binflag= 0 Dwg= 0.00 Dwb= 0.00
+ACM= 0 ldif=0.00 hdif=0.00 +rsh= 7 rd= 0 rs= 0 +rsc= 0 rdc= 0
+K1= 0.3910000 K2= 1.0000000E-02 K3= 0.00 +Dvt0= 2.6700001 Dvt1= 0.5300000 Dvt2= 5.0000000E-02 +Dvt0w= 0.00 Dvt1w= 0.00 Dvt2w= 0.00 +Nlx= 7.5000000E-08 W0= 0.00 K3b= 0.00 +Ngate= 5.0000000E+20
+Vsat= 1.0500000E+05 Ua= -5.0000000E-10 Ub= 1.5000000E-18 +Uc= -2.9999999E-11 +Prwb= 0.00 Prwg= 0.00 Wr= 1.0000000 +U0= 5.5000000E-03 A0= 2.0000000 Keta= 4.0000000E-02 +A1= 0.00 A2= 0.9900000 Ags= -0.1000000 +B0= 0.00 B1= 0.00
+Voff= -7.0000000E-02 NFactor= 1.5000000 Cit= 0.00 +Cdsc= 0.00 Cdscb= 0.00 Cdscd= 0.00 +Eta0= 0.2500000 Etab= 0.00 Dsub= 0.8000000
+Pclm= 0.1000000 Pdiblc1= 1.2000000E-02 Pdiblc2= 7.5000000E-03 +Pdiblcb= -1.3500000E-02 Drout= 0.9000000 Pscbe1= 8.6600000E+08 +Pscbe2= 1.0000000E-20 Pvag= -0.2800000 Delta= 1.0100000E-02 +Alpha0= 0.00 Beta0= 30.0000000
+kt1= -0.3400000 kt2= -5.2700000E-02 At= 0.00 +Ute= -1.2300000 Ua1= -8.6300000E-10 Ub1= 2.0000001E-18 +Uc1= 0.00 Kt1l= 4.0000000E-09 Prt= 0.00
+Cj= 0.0015 Mj= 0.7175511 Pb= 1.24859 +Cjsw= 2E-10 Mjsw= 0.3706993 Php= 0.7731149 +Cjgate= 2E-14 Cta= 9.290391E-04 Ctp= 7.456211E-04 +Pta= 1.527748E-03 Ptp= 1.56325E-03 JS=2.50E-08 +JSW=4.00E-13 N=1.0 Xti=3.0 +Cgdo=3.49E-10 Cgso=3.49E-10 Cgbo=0.0E+00 +Capmod= 2 NQSMOD= 0 Elm= 5 +Xpart= 1 cgsl= 0.582E-10 cgdl= 0.582E-10 +ckappa= 0.28 cf= 1.177e-10 clc= 5.4750000E-08 +cle= 6.4600000 Dlc= 2E-08 Dwc= 0

パラメータの詳しい説明は省略する。HSpice のマニュアルを参照してほしい。

付録C

単相クロックを用いたウェーブパイプライン用累算加算器を示す。単相クロックでは、クロック周波数 4.54GHz が限界である。ウェーブパイプライン化により、全加算器の中には複数のデータが存在するため、2 つのフィードバック機構を用いて累算を行う。



単相クロックによるウェーブパイプライン用累算加算器

付録D

追加実験

1. 目的

すべての入力に対し、遅延が均等な全加算器を設計する。完全な遅延均等化設計により、シストリックアレイ乗算器の部分積生成回路のパイプラインラッチの段数を減らすことができる。本論文に示した全加算器１段にラッチが入っている場合では、全加算器の遅延差よりもラッチによるオーバーヘッドが大きく、全加算器の遅延均等化が少し損をした感じとなっている。完全な遅延均等化全加算器を設計し、ラッチの挿入段数を減らす。ラッチの段数の削減により、消費電力やレイアウト面積、レイテンシを小さくすることができる。

2. 遅延均等化方法

完全な遅延均等化全加算器を設計するために、遅延素子として 1 入力のナンドゲートとトランスファゲートを使用する。トランスファゲートは MOSFET の幅 W を変化させることにより、オン抵抗を変えることができ、遅延時間の調節を容易にできる。これら遅延素子を使い、全加算器の論理ゲートの段数を完全に揃える。

3. 遅延均等化全加算器の設計

図 1 に遅延均等化全加算器を示す。1 入力ナンドゲートとトランスファゲートを用いて論理段数を揃えた全加算器回路である。また、ファンアウト数が多い場所や最終段には駆動バッファを挿入した。図中の数値は NMOS と PMOS の幅 W の比率を示している（例；1.5 / 3.5 は、NMOS の幅 1.5L / PMOS の幅 3.5L）。L は MOSFET のゲート長 90nm である。なお、ノットゲートとトランスファゲートの遅延時間が等しくなるようにトランスファゲートの W を決めた。この全加算器は、論理段数を揃え、同じ論理段数の場所に同じ素子を使用した究極の遅延均等化回路設計となっている。

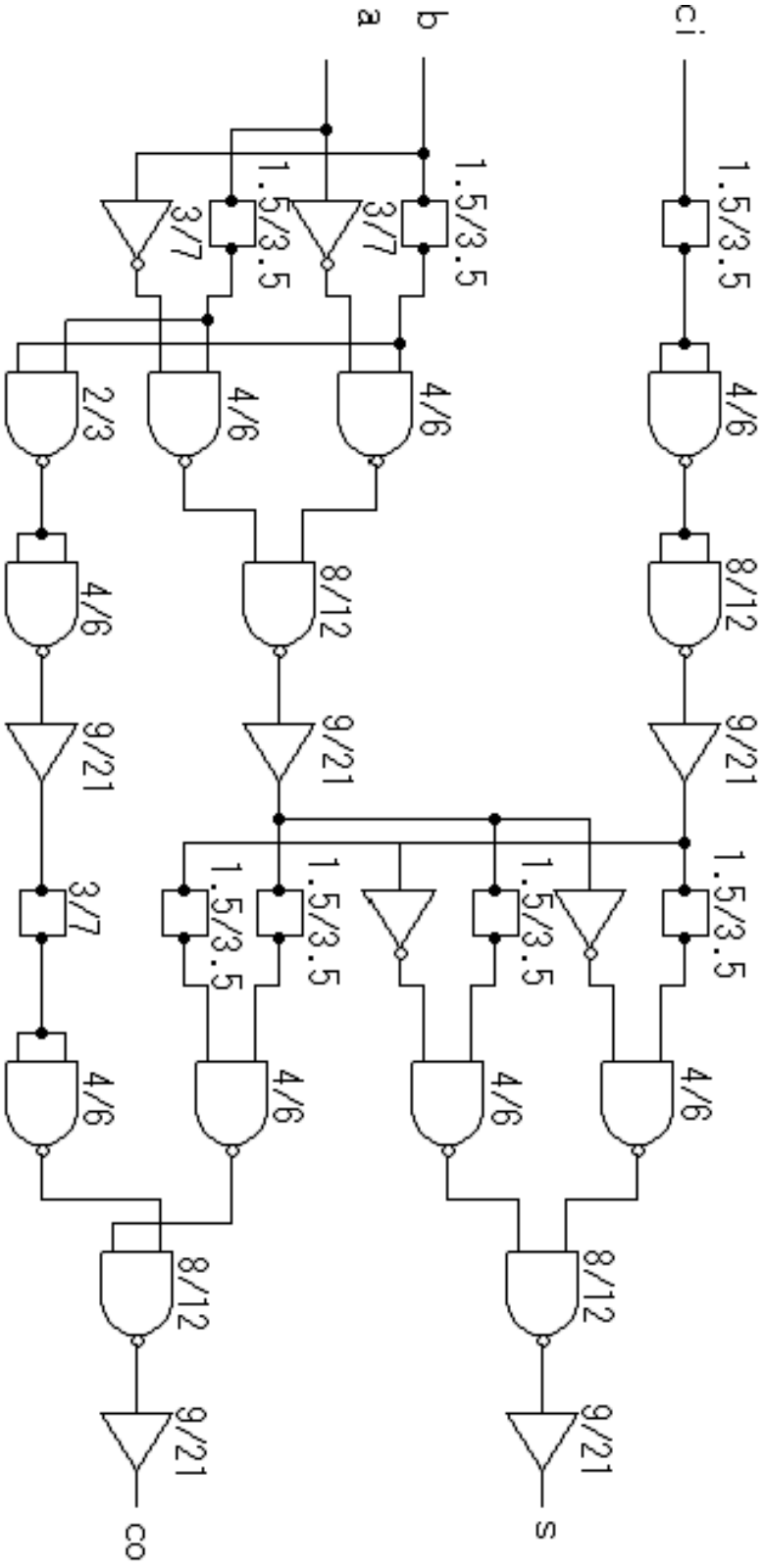


図 1. 完全な遅延均等化設計の全加算器

3.HSpice による全加算器の遅延解析

遅延均等化全加算器の入力変化 64 通りの遅延解析を行う (図 2 参照)。今回は、配線遅延を含まないゲート遅延のみのシミュレーションである。遅延素子として 1 入力ナンドゲートとトランスファゲートを使用した全加算器の遅延均等化効果を検証する。表 1 に遅延解析の結果を示す。図 3 に各入力変化の遅延の分布を示す。論理段数を揃えることにより、遅延の分布が対称になっている。

最大遅延時間は 271.5ps、最小遅延時間は 232.5ps となった。よって、最大と最小の遅延差は 39ps となった。遅延差を平均遅延時間（257.5ps）の 1 割弱に抑えたかったが、うまくいかなかった。その原因はナンドゲートにある。ナンドゲートには入力変化による遅延時間にバラツキがあるため、どうしても遅延差を縮めることができない。

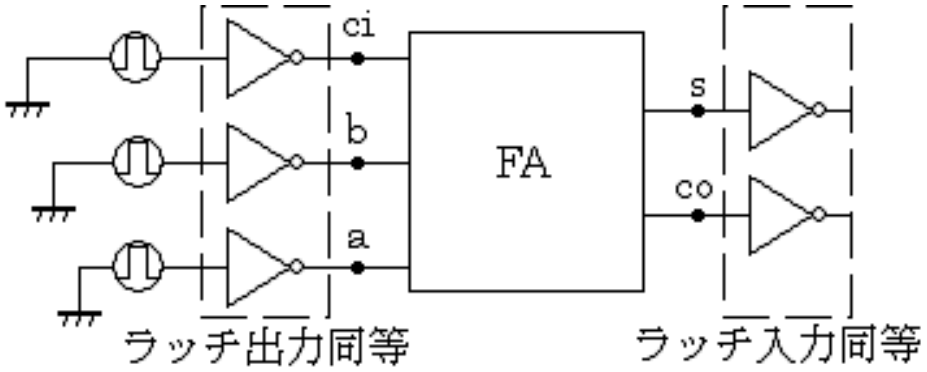


図 2. 全加算器の遅延解析

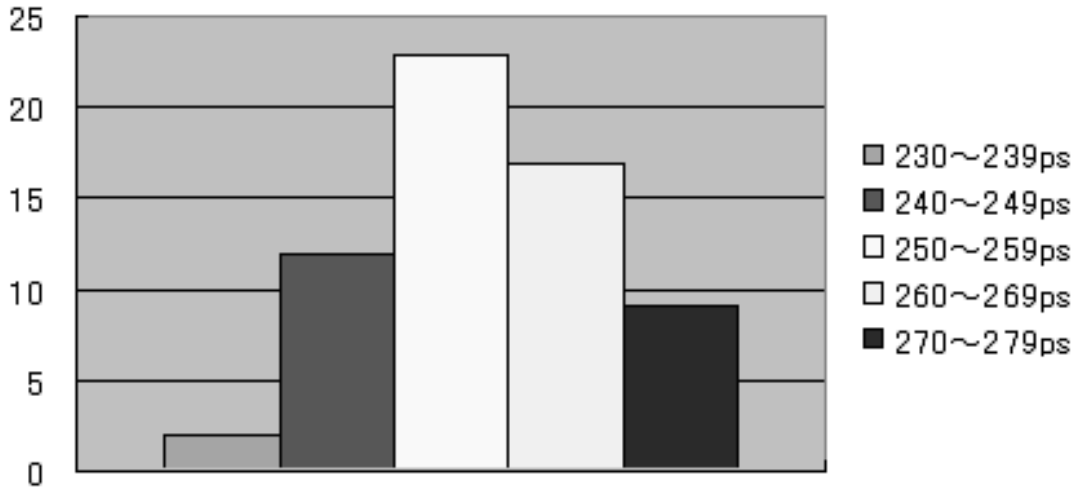


図 3. 遅延の分布

表 1. 全加算器の遅延解析結果

時刻 t					時刻 t+1						
A	B	Ci	S	Co	A	B	Ci	S	delay[ps]	Co	delay[ps]
0	0	0	0	0	0	0	0	0		0	
0	0	0	0	0	0	0	1	1	270	0	
0	0	0	0	0	0	1	0	1	251	0	
0	0	0	0	0	0	1	1	0		1	271
0	0	0	0	0	1	0	0	1	249.5	0	
0	0	0	0	0	1	0	1	0		1	271
0	0	0	0	0	1	1	0	0		1	270.5
0	0	0	0	0	1	1	1	1	264.5	1	270
0	0	1	1	0	0	0	0	0	254	0	
0	0	1	1	0	0	0	1	1		0	
0	0	1	1	0	0	1	0	1		0	
0	0	1	1	0	0	1	1	0	254	1	251.5
0	0	1	1	0	1	0	0	1		0	
0	0	1	1	0	1	0	1	0	252	1	249.5
0	0	1	1	0	1	1	0	0	251	1	270.5
0	0	1	1	0	1	1	1	1		1	270.5
0	1	0	1	0	0	0	0	0	249.5	0	
0	1	0	1	0	0	0	1	1		0	
0	1	0	1	0	0	1	0	1		0	
0	1	0	1	0	0	1	1	0	269	1	267.5
0	1	0	1	0	1	0	0	1		0	
0	1	0	1	0	1	0	1	0	269	1	267.5
0	1	0	1	0	1	1	0	0	246.5	1	260.5
0	1	0	1	0	1	1	1	1		1	263.5
0	1	1	0	1	0	0	0	0		0	**232.5**
0	1	1	0	1	0	0	1	1	257	0	249.5
0	1	1	0	1	0	1	0	1	258	0	251
0	1	1	0	1	0	1	1	0		1	
0	1	1	0	1	1	0	0	1	258.5	0	251.5
0	1	1	0	1	1	0	1	0		1	
0	1	1	0	1	1	1	0	0		1	
0	1	1	0	1	1	1	1	1	253.5	1	
1	0	0	1	0	0	0	0	0	250	0	
1	0	0	1	0	0	0	1	1		0	
1	0	0	1	0	0	1	0	1		0	
1	0	0	1	0	0	1	1	0	269.5	1	267.5
1	0	0	1	0	1	0	0	1		0	
1	0	0	1	0	1	0	1	0	269	1	267.5
1	0	0	1	0	1	1	0	0	245.5	1	262
1	0	0	1	0	1	1	1	1		1	264.5
1	0	1	0	1	0	0	0	0		0	233
1	0	1	0	1	0	0	1	1	257	0	250
1	0	1	0	1	0	1	0	1	254	0	247.5
1	0	1	0	1	0	1	1	0		1	
1	0	1	0	1	1	0	0	1	258	0	251
1	0	1	0	1	1	0	1	0		1	
1	0	1	0	1	1	1	0	0		1	
1	0	1	0	1	1	1	1	1	252.5	1	
1	1	0	0	1	0	0	0	0		0	244.5
1	1	0	0	1	0	0	1	1	269.5	0	247
1	1	0	0	1	0	1	0	1	249.5	0	268.5
1	1	0	0	1	0	1	1	0		1	
1	1	0	0	1	1	0	0	1	247	0	266
1	1	0	0	1	1	0	1	0		1	
1	1	0	0	1	1	1	0	0		1	
1	1	0	0	1	1	1	1	1	270	1	
1	1	1	1	1	0	0	0	0	253.5	0	247.5
1	1	1	1	1	0	0	1	1		0	245.5
1	1	1	1	1	0	1	0	1		0	**271.5**
1	1	1	1	1	0	1	1	0	253	1	
1	1	1	1	1	1	0	0	1		0	269
1	1	1	1	1	1	0	1	0	250.5	1	
1	1	1	1	1	1	1	0	0	254	1	
1	1	1	1	1	1	1	1	1		1	

4. 今後の課題

今回は時間がなかったため、ゲート遅延のみの評価である。本研究で使用した 90nm プロセスルールでは、全体の遅延時間における配線遅延時間の占める割合が大きい。そのため、全加算器のレイアウトを行い、配線遅延を含めた遅延解析が必要である。正確な配線遅延を知るためには、配線モデルが必要である。今日では、縦方向だけでなく横方向の寄生容量を考慮しなければならない。

最後に配線の問題とウェーブパイプラインについて述べる。

4.1 クロストーク

微細加工技術の進歩により、配線間隔が狭くなりクロストークが問題となってきている。図 4 に示すような並行した長い配線があるとする。配線間には、カップリング容量 Cc がある。クロストークとは、aggressor の信号変化がカップリング容量を介して隣接する配線に影響を与える現象である。このクロストークによるノイズが論理しきい値を超えれば、誤作動を引き起こすことになる。

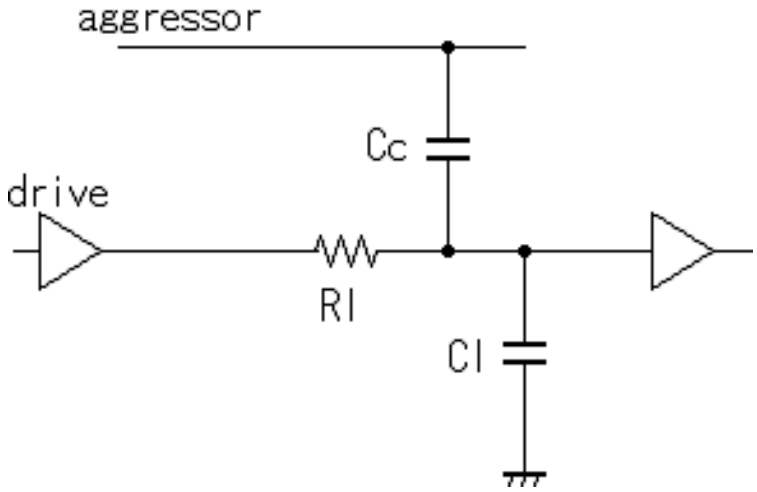


図 4. クロストーク

4.2 RLC 伝送線路モデル

LSI の配線は、RC 集中定数線路のモデルで計算されてきた。しかし、回路の高周波化や銅による配線抵抗の低下によりインピーダンス比 $\omega L/R$ が増加し、インダクタンス L が無視できなくなるだろう。

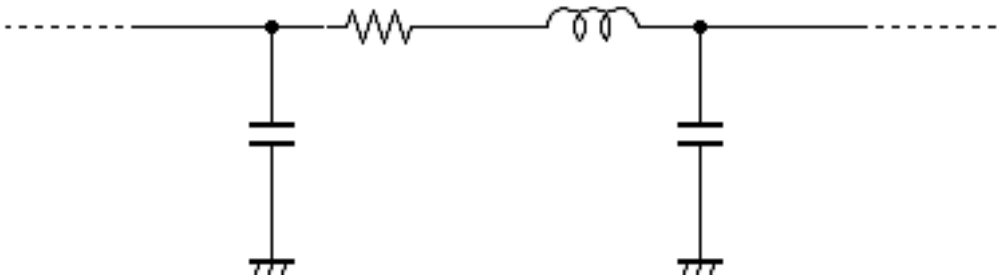


図 5.RLC 伝送線路モデル

4.3　低消費電力技術とウェーブパイプライン

近年、LSI の低消費電力設計が必要とされている。携帯のモバイル端末向けの LSI では、バッテリー駆動であるため低消費電力であることが要求される。パソコンに搭載される CPU では LSI チップの発熱が問題になり、放熱対策が必要となっている。このように最近では、低消費電力技術が必要である。下記に低消費電力技術を 2 つ述べる。また、これらの技術とウェーブパイプライン方式の相性について議論する。

4.3.1　ゲート・サイジング

組み合わせ回路内でクリティカルパス上でない論理ゲートに対して、最大遅延時間を越えない範囲で MOSFET の W を小さくし、ゲート容量を削減する方法をゲート・サイジングと呼ぶ。

4.3.2　Dual-Vt

微細加工技術の進歩により、リーク電流が問題となってきている。スレショルド電圧が小さくなると、サブスレショルド・リーク電流は指数関数的に増大する。クリティカルパス以外の論理ゲートに高いスレショルド電圧の MOSFET を使用し、リーク電流を削減する方法を Dual-Vt と呼ぶ。

4.3.3　ウェーブパイプライン方式

ウェーブパイプライン方式は、最大遅延時間と最小遅延時間との差をクロックサイクルとするパイプライン方式である。つまり、遅延が均等な回路を設計することにより高クロック動作を実現することができる。

低消費電力技術のゲート・サイジングや Dual-Vt は、遅延に対して考えるとクリティカルパス以外のパスを遅らせることができる技術である。これらの技術をウェーブパイプラインに利用すると、回路の遅延均等化に必要な遅延素子の数を減らすことができるだろう。

謝辞

最後に本研究を御指導くださった日比野靖教授に深く感謝いたします。教科書には載っていない多くのことを学べて幸せに思います。

また、貴重な御意見をくださった本学の田中清史助教授、井口寧助教授に深く感謝いたします。

その他、貴重な御意見、御討論を頂ました計算機アーキテクチャ講座の皆さんや私を支えてくれた人達に深く感謝するとともに御礼申し上げます。