修士論文

自己同期システム向けダイナミック回路の低電圧動作における最適化

学籍番号 : 37-106449

神谷　歩未

指導教官 : 池田 誠准教授

東京大学大学院　工学系研究科　電気系工学専攻

2012 年 2 月 8 日提出

目次

図目次

表目次

第1章

序論

1.1 研究の背景

近年、快適さや利便性を求めるライフスタイルの多様化により、エネルギー消費は増加しつづけている。大量のエネルギーを消費しながら経済成長を遂げてきたが、エネルギーは有限であり、今年頭の東日本大震災以降は節電が毎日のように叫ばれている。従って、省エネルギー製品はますます需要が高まっている。また、消費電力を小さくすることでバッテリの長寿命化にもつながる。消費電力はトランジスタの静電容量×電圧の2乗×動作周波数に比例する [1] ため、省エネには電圧を下げることが最も有効だとされている。

しかし集積回路は、微細化や高速化が進むにつれ、ディジタル回路においても設計通りの性能を保証することが困難になり、特に低電圧動作において誤動作による信頼性の低下が問題となってきている。動作周波数や消費電力の増加および配線の微小化は IR ドロップや Ldi/dt ノイズによる電源電圧変動の増大をもたらし [2]、配線間の幅が小さくなることは信号間のクロストークや遅延変動や論理エラーの危険を増加させている [3]。

また、クロック同期回路においては、ばらつきによってタイミング制御や、クロック歪みおよび回路によるクロック遅延の問題がある [3]。これらの問題を解消する為に、ハードウェアを用いると、結局は消費電力が増大してしまう。一方、パイプライン技術を用いる解決法では、パイプラインの段数が増加するにつれ、スイッチング雑音による誤動作の確率が上がり、信頼性が低下することが問題視されている。

さらに、動作周波数や消費電力の増加及び配線の微小化は電源電圧変動の増大をもた

らし、今後もこの傾向は続くとされている。また、配線幅が小さくなることで信号間のクロストークによる遅延変動や論理エラーの危険が増している。信号遅延が大幅に遅くなると、クロック同期式回路においてクロックサイクル内に演算が終了せず, 誤動作に繋がる。その為、動作周波数の高い設計を要する LSI に対しては、遅延解析より歩留まりを向上させており、それらにおいて設計コストや TAT(Turn-around time) が問題となっている。

回路には、大きく分けて 2 種類の動作方法がある。クロックを用いて同期する同期式回路と、クロックに同期した動作を行わない自己期回路である。自己同期回路は、高速化や低電力化を目的として研究し使用されてきた [4, 5]。自己期回路は遅延変動によって誤動作を起こさないという特徴がある。また、回路の微小化は宇宙線に起因する中性子線や α 粒子によるソフトエラーに対する耐性を低下させている。これらの高エネルギー粒子は PN 接合以上に衝突すると大きな電荷を発生させるためである。この問題は、以前はメモリにおいてのみ考えられていたが論理回路においても問題となってきている [6]。この問題はクロストークによる論理エラーとともに過渡的なエラーとして信頼性の低下を招いている。この様な過渡的エラーの対策として、実行時エラー検出も重要視されてきている。そこで、終了検出とエラー検出が可能な自己同期回路は、上述した信頼性の低下問題に対する解決策として注目されている。

低電圧下において動作させる場合、この信頼性の確保がより重要となってくる。特に、近年注目をあびている医療分野やセンサネットワークの分野においても、低電圧かつ低消費電力の動作において、より信頼性の高い回路が求められている。本研究では、同期回路に比べて、製造ばらつきやなんらかのノイズに対してもクロックを使用しないため耐性があるとされている自己同期回路において、低電圧動作においても適応できる回路の最適化をいくつかの指標において行うことで、今後の自己同期回路設計の一助になることを目的としている。

1.1.1　自己同期回路

同期回路はクロックにあわせてタイミングを調節するのに対し、自己同期回路はデータパスの状態によってタイミングを決定する回路である。演算終了を確認後、信号を遷移させる為、動作は一定周期ではない。以下に同期回路と比較したときの自己同期回路

の長所を挙げる。

- 同期回路は何らかの原因で回路の遅延に変動が生じた際、その遅延が設計時にとっておいたマージンを含んだクロックサイクルに対し長くなった場合は誤作動を起こしてしまう。それに対し、自己同期回路では,演算の終了を確認してから次の演算を行うため、設計時に予想した速度以上の遅延の変動が起こった場合でも誤作動を起こすことがない、遅延変動耐性がある。

- 同期回路は常に一定のクロックを用いて動作させるため、そのクロック周期は論理回路の遅延の最悪の値にマージンを加えたものとしなければならない。それに対し、自己同期回路の遅延は最悪遅延の場合はそれに合わせるが、逆に早く演算が終了したものに大しては終了を検出して次の演算を行うため、全体として見たときに平均速度で動作しているといえる。

- 同期回路は一定の周波数で動作するため、回路自身が放出する電磁妨害波 (EMI:Electro magnetic interference) は回路のクロック周波数に集中してしまう。それに対し、自己同期回路は一定の周波数で動作するわけではないので、放出する電磁波のスペクトルは分散するため、同期回路に比べて EMI 特性は良いといえる。

自己同期回路には、主に上記に記したような長所を持っている。そのため、悪条件にさらされた場合であっても信頼性が高く、また同期回路よりも早く動作するシステムに応用できるという点で、大変注目をあびている。

1.1.2　本研究の目的

自己同期回路は信頼性において優れた特徴を持っているだけでなく、平均遅延での動作が可能となり同期回路に比べて高速なシステムに応用できる要素も持っている。そのため、高速化や低電力化などの研究が行われてきた [4, 5, 7]。しかしながら、低電圧動作によってより低電力動作を行う回路に関する研究や、またその低電圧下において自己同期回路の最適化に関しての研究はほとんど行われていない。本研究では、低電圧下での動作において自己同期システム向け回路単体の最適化を行い、最適化による回路単体における優位性を示すことにより、動作電圧を低くすることでより低電力動作を行う自己同期システムを実現するための一助となることを目的としている。

1.1.3　本論文の構成

本論文の構成は以下のようになっている。まず第 2 章で自己同期システムを構成する回路をダイナミックとスタティックの 2 種類において紹介し、その最適化手法について述べる。第 3 章では、最適化した回路をシミュレーションにより比較を行う。また、シミュレーションから動作限界などの決定方法についても述べる。第 4 章では、最適化した回路を実際に 0.18umCMOS プロセスにて試作しその測定結果を示す。また、特に動作電圧の低い点においての最適化の評価も行っている。第 5 章は、汎用性のある回路の一例として加算器について最適化を行い、その回路の測定結果を示し評価を行っている。最後に、第 6 章にて結論を述べる。

第2章

ダイナミック回路の最適化

2.1　自己同期システムを構成する回路

2.1.1　2線符号化

自己同期回路の終了検出方法には、二線式論理を用いたデータパスに冗長性を持たせる二線符号化 [8] を用いた方法があり、表. 2.1 に示したとおり ”0 ”を ”01 ”、”1 ”を ”10 ”と 1 つのデータを 2 値で表現する論理である。各論理ゲートの始めの出力は ”00 ”であり、信号が到達し出力論理が決定すると ”01 ”または ”10 ”の状態に遷移する。片方が立ち上がることで信号が決定し、その後リセットにより初期状態の ”00 ”に戻す。また, ”11 ”は正常な二線式回路では使用されず、後述のエラー検出に用いられている。

2.1.2　2線式論理を用いた終了検出およびエラー検出

2 線式符号化による 2 線式論理を用いることで、終了検出とエラー検出が可能になる。

まず、終了検出回路について紹介する。二線式論理では、信号の値が確定したときに

表 2.1 二線符号化.

2 線の状態	信号の意味
00	初期状態
01	論理値 0
10	論理値 1
11	エラー状態

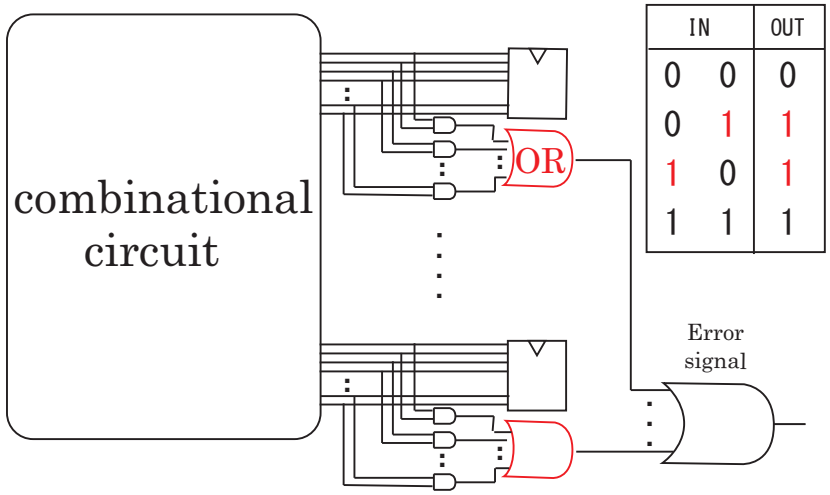
二線のどちらかは ”1 ”となっている。従って、図 2.1 の (a) のように各々の出力の OR をとり出力論理の決定を検出し、それら全ての AND をとることにより終了を検出できる。この検出回路を用いて、プリチャージ→演算→終了検出→レジスタ書き込み→プリチャージという流れで処理することによりクロックを用いた同期を必要としない演算回路を構成できる。

また、2 線式論理はその冗長性によりエラー検出が可能である。終了検出型プロセッサは演算が終わってからレジスタ書き込みが行われるため、タイミングによるエラーは起きず、エラーは信号がとるべき値と異なる論理エラーのみと考えられる。エラーが ”00 ”の場合はタイムアウト回路を設けることで検出できる。”11 ”でエラーの場合はエラーの次段以降も ”11 ”となるため、2 線の AND をとることで検出できる。プロセッサにおいては、図 2.1 の (b) のように、レジスタへの入力や外部出力信号の 2 線の AND をとり、それらの OR をとることで ”11 ”エラーの検出が可能になる。また、片方のエラーであれば ”00 ”あるいは ”11 ”となるが、2 線の両方が反転するエラーについてはどちらもプルダウンしていない為、原因が定常的なものであると考えられるので、テスト入力などで調べることができる。この様に, エラーを検出できる点も二線式論理のメリットである。

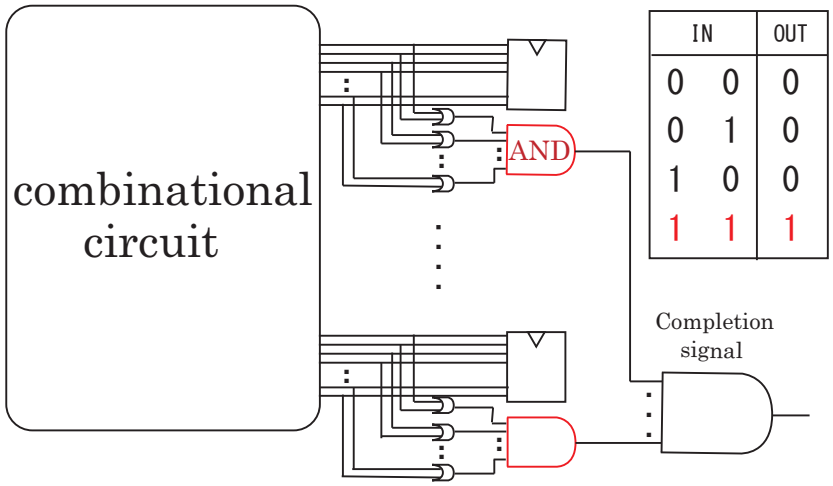
以上のように行うことで、終了検出とエラー検出が回路自身で行うことができ、図 2.2 に示すような自己同期的動作が可能になる。しかしながら、2 線式論理を用いるためには、必ず演算回路と相補的な回路が必要となるため、回路の面積は必ず大きくなってしまうという欠点がある。

2.1.3　2 線式ダイナミック回路

2 線式符号化を用いた回路の一つに DCVSL(Differential Cascode Voltage Switch Logic) 回路がある。一般的な DCVSL 回路の構造を図 2.3 に示した。DCVSL 回路は、2 線式論理の 2 線それぞれを出力する 2 つの相補的なダイナミック回路で構成されている。また、出力はドミノ回路を構成するためのインバータによって駆動されている。さらに、出力にインバータをつけることで、ダイナミック回路の駆動力が小さい点を補う役割を担っている。P と表記された入力はプリチャージ信号と呼び、P が low であるときがプリチャージ状態でリセットが行われ、逆に P が high であるときはエバリュエーショ



(a)Complete Detection Circuit



(b)Error Detection Circuit

図 2.1 終了検出およびエラー検出の方法

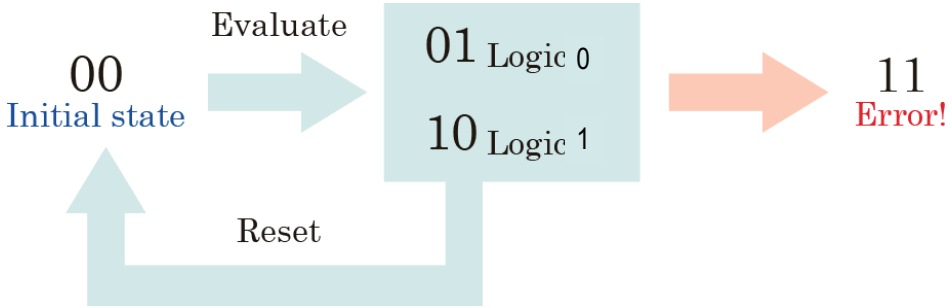


図 2.2 2 線式論理を用いた自己同期システムの概念図

ン状態となり演算が行われる。エバリュエーション状態においては、2 つのプルダウンネットワークのうちの一方のみがプルダウンされることにより、インバータを通した出力の片方のみが high となる回路になっている。DCVSL 回路はダイナミック回路であるため、エバリュエーション状態において入力が 0 から 1 に一度のみ遷移が許されるが、論理回路を全て DCVSL ドミノ回路で構成することで、要求は満たされる。

DCVSL 回路の一例として、NAND 回路を図 2.4 に示した。図の回路の左半分は通常の NAND 回路であり、右側にその相補的な回路を組み合わせる形になっている。例えば、この回路に A に 10(論理値 1) と B にも 10(論理値) を入力した場合を考える。まず最初にプリチャージをされ、左右の出力はともに 0 となっている。P が 1 になることで、pMOS 側は閉じ、逆に nMOS 側が open になって演算が開始される。A1 と B1 が共に 1 なので、左側の回路はプルダウンされて Y2 には 1 が出力される。右側の回路は、A2 と B2 が共に 0 であるため、プルダウンされることなく Y1 の出力は 0 のままである。つまり出力は”01”(論理値 0) となり、NAND の演算に合致していることがわかる。また、緑色の四角で囲んだキーパーは、pMOS 側に出力のフィードバックをかけることで、リークによる値の変化を止めて値を保持する役割を担っている。

DCVSL 回路の特長としては、自己同期システムにすることができるため平均遅延の動作が可能である点、スイッチングスピードが速い点、ダイナミックパワーが CMOS 回路に比べて小さくて済む点などが挙げられる。また、2 線式回路の欠点である面積については、ダイナミック回路にすることで、演算に必要なトランジスタ数は通常の CMOS 回路と同じであるため、プリチャージとキーパーのトランジスタ分のみ大きいといえる。

2.1.4　2 線式スタティック回路

プリチャージを必要とするダイナミック回路で構成された DCVSL 回路に対し、より低電圧で動作させるために考案されたのが 2 線式スタティック回路である。[12]2 線式スタティック回路は、プリチャージ用の pMOS 回路の代わりにプルダウンネットワークと相補的な回路を挿入することで DCVSL と同等の論理を実現するものである。2 線式スタティック CMOS 論理ゲートでは、相補的な pMOS、nMOS ネットワークの双方に入力信号が接続されるため、入力容量が大きくなるが、プリチャージ動作を必要としな

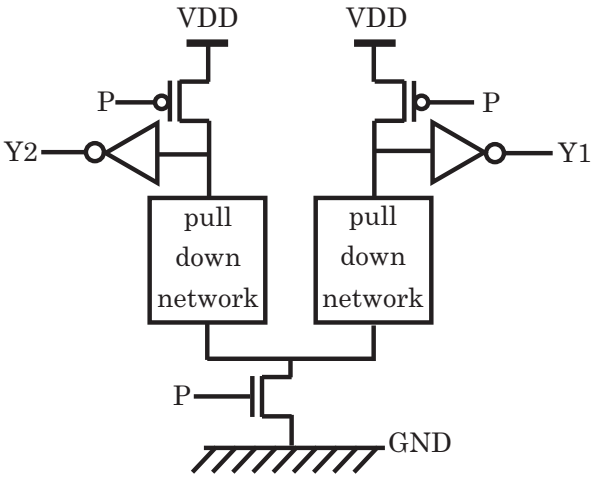


図 2.3 一般的な DCVSL 構造

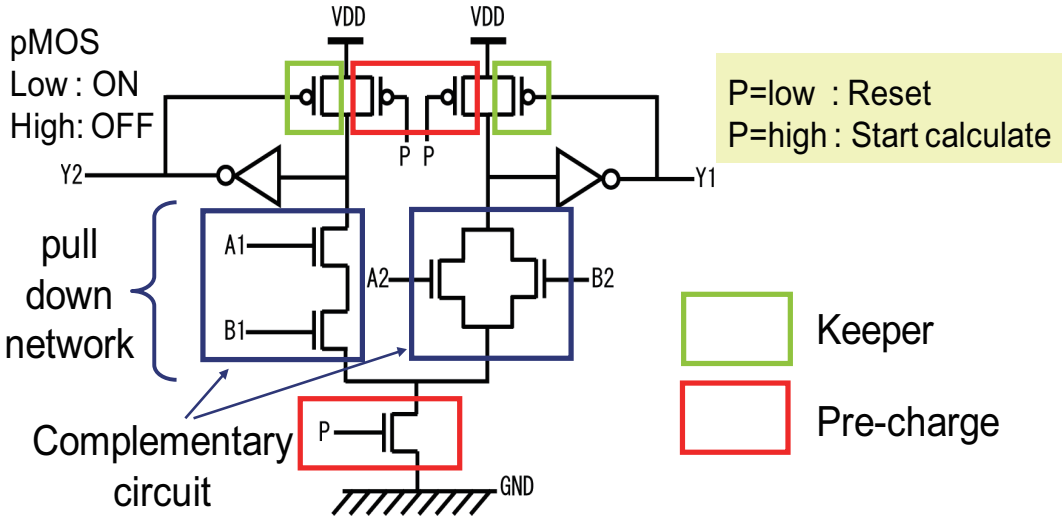


図 2.4 2 線 DCVSL 型 NAND 回路

いため動作電圧がしきい値以下でも動作すると考案された回路である。

2 線式 CMOS 論理を用いたスタティック NAND 回路を図 2.5 に示した。P に low を入れた時、出力も low となり初期状態となる。その後 P に high を入れることで 2 線論理を用いた演算の結果となる出力が得られる。CMOS 回路を 2 つ組み合わせたものに P を導入することで、演算終了後にリセットを行うことが可能になり、DCVSL ドミノと同様の機能を有する回路が構成できる。これにより、終了検出およびエラー検出が可能な自己同期的システムに用いることが可能となる。

2 線式スタティック回路は自己同期的に動作させることができるメリットに対し、入

力キャパシタが大きくなる点やスタティックにするためにダイナミック回路よりもトランジスタ数を多くしなければならずその結果回路面積が大きくなってしまうという問題を抱えている。

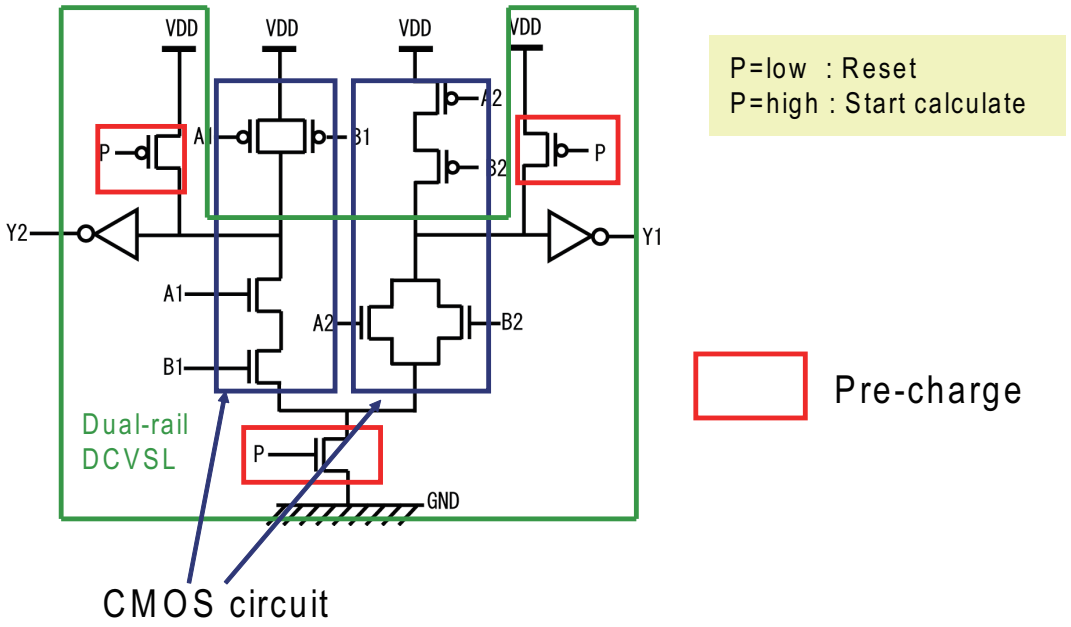


図 2.5 2 線 CMOS 型 NAND 回路

2.2 回路の最適化

2.2.1 最適化に用いる回路

同期回路の代表として通常の CMOS 回路 (以下 1 線 CMOS 回路) を、終了検出型自己同期回路の代表としては二線式論理を用いた自己同期回路で紹介した冗長性を持つ二線式回路の例として上述した 2 線式ダイナミック回路のひとつの DCVSL 回路と、2 線式スタティック回路の 2 つを取り上げた。1 線 CMOS は、pMOS 側と nMOS 側に相補的な回路を置くことでスタティック回路を実現した一般的な回路であり、クロックに同期して動く回路となっている。1 線式 CMOS 回路では、入力が遅れてくることによってパルス状のノイズが発生するグリッチの問題がある。グリッチは、余計なダイナミック電力を消費させたり内部雑音が増加したりと回路に悪影響を及ぼす。スタティック回路でグリッチの発生を防ぐにはカルノーマップ上で加えても変わらない式をたすことに

より防ぐ手法がある [9]。また、シミュレーションによりグリッチが起こる部分を見付けその部分に遅延を加えてグリッチを消去するツールもある。しかし、どちらの方法も回路生成前に膨大な量の検証が必要となったり、新たな回路をつけると回路が結局大きくなるという欠点がある。

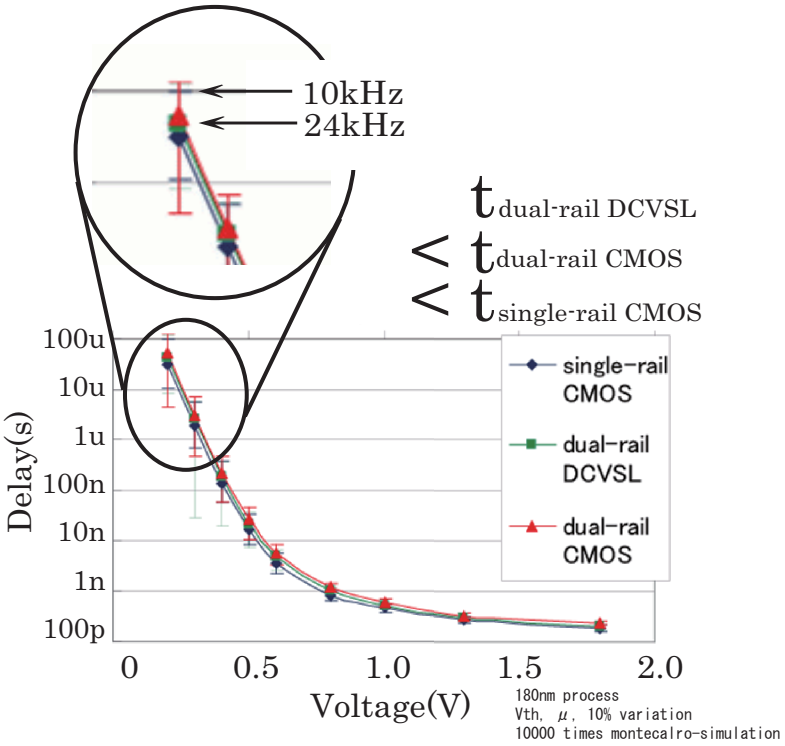


図 2.6 任意のトランジスタ幅における 3 回路の遅延のシミュレーション結果.

本研究では、上述した 1 線 CMOS 回路と、前節にて紹介した自己同期システムに応用できる 2 線 DCVSL 回路と 2 線 CMOS 回路の 3 回路について比較を行った。まず最適化を行う前に任意のトランジスタサイズにおいて、3 回路のシミュレーション比較を行った。シミュレーションは実際の製造ばらつきを考慮して、しきい値と移動度を 10 %ランダムにばらつかせたモンテカルロシミュレーションを 10000 回行った。結果を図 2.6 に示す。図より、全ての回路において動作電圧が低くなるにつれて遅延は急激に大きくなっていることがわかる。各点における縦のバーは、10000 回のモンテカルロシミュレーションにおける最大遅延と最小遅延の値を示している。動作電圧が高い点においてはばらつきによる遅延の変動は小さいため、自己同期式回路の方のメリットはあまり発揮されない。しかし、動作電圧が低くなるにつれ、遅延の増大とともにその傾きが

急になっているため、遅延のばらつき幅が増大している様子がみてとれる。動作電圧が 0.2V の点において、2 線 DCVSL 回路は自己同期的に動作させることを考慮すると平均遅延で動くため約 24kHz で動作させることが可能である。それに対し 1 線 CMOS 回路は最悪遅延にマージンを加えたクロックで動作させると考えると、最悪遅延における動作速度の約 10kHz よりも遅くなってしまうということがわかる。この点からも、低電圧動作においては自己同期回路は同期回路に比べてメリットがあることがわかる。また、2 線 DCVSL 回路と 2 線 CMOS 回路については、動作電圧によらず 2 線 DCVSL 回路の方が遅延は小さくなっている。これは、ダイナミック回路は nMOS 側でディスチャージをするだけで演算結果が出るのに対し、スタティック回路は pMOS と nMOS のバランスによって決定されるため演算に要する時間がダイナミック回路より多くなることによると考えられる。

2.2.2　最適化における回路、遅延、電力の定義

回路単体を図 2.7 のように FO4 で構成して最適化を行い、それを組み合わせることで最適な回路を求めた。通常の場合、FO は 3 くらいであるため FO4 で駆動させることを想定しておくことで大抵の場合は動作すると考えた。最適化については 1 つの回路について行うため、図 2.7 に示すように真ん中の回路の入力と出力の差を遅延に、電力に関しても真ん中の回路のみ電源を分けてそこに流れる電流値に動作電圧をかけたものを回路単体の電力として定義した。

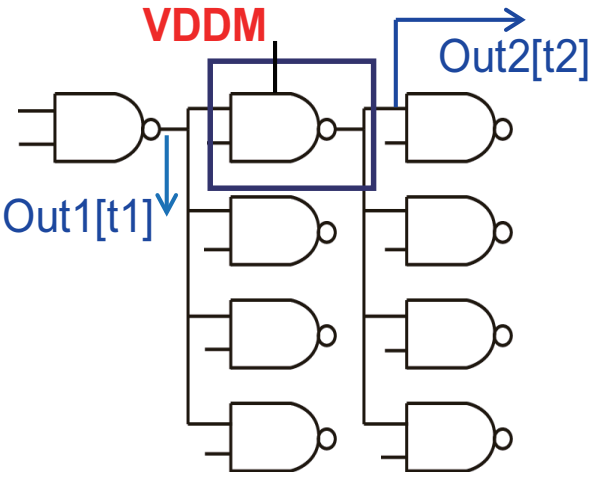


図 2.7 FO4(ファンアウト 4) 回路.

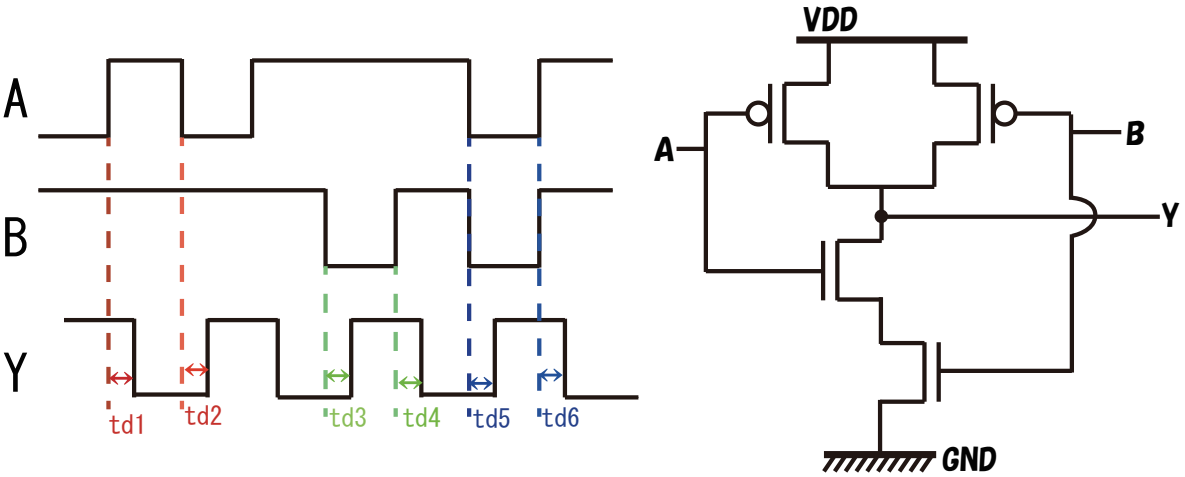


図 2.8 1 線 CMOSNAND 回路における遅延の分類.

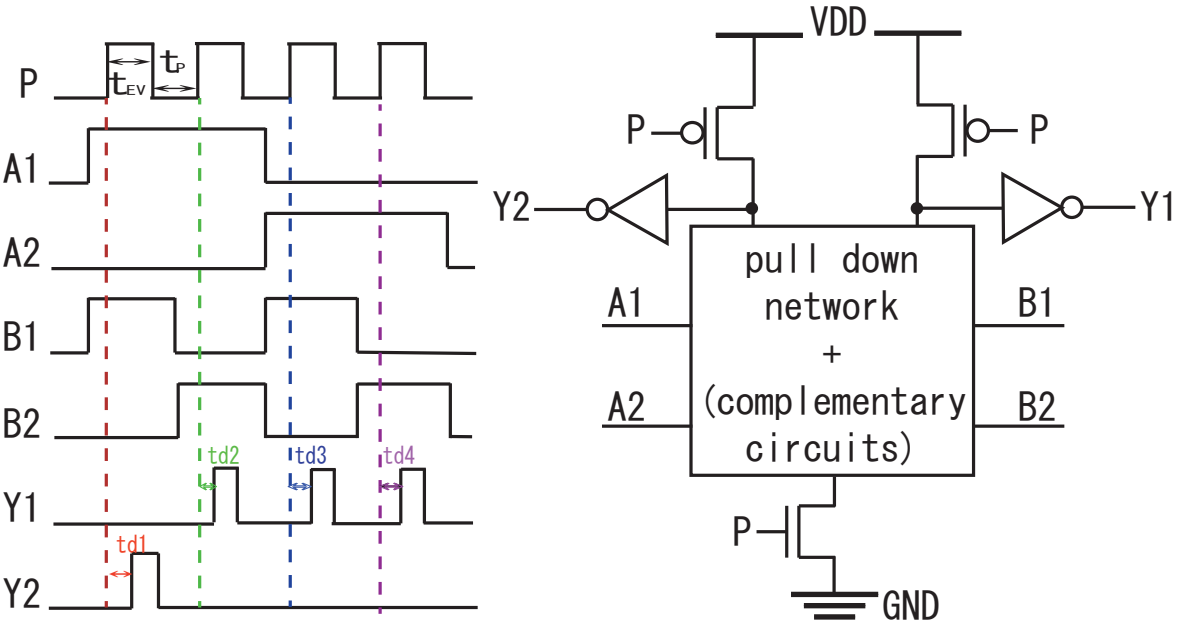


図 2.9 2 線式 NAND 回路における遅延の分類.

遅延については、考えうる 2 値の演算における遅延の平均の値を用いた。1 線 CMOS の場合は、図 2.8 に示すように各々の値が変化する際の出力の変化をみる td1～td6 の 6 通りの値の平均を用いた。それに対し、2 線式回路は毎回プリチャージが行われ値は出力の立ち上がりによってのみ変化するため、図 2.9 のように立ち上がりの遅延のみを見た td1～td4 の値を平均したものを 2 線式回路の平均遅延と定めた。

電力においては、最高動作周波数で動作させた場合に電源から流れる電流量を用いて

値を求めた。最高動作周波数は、最初に測定した遅延をもとに動作させ、プリチャージ時間 t_P と演算時間 t_{EV} が共に遅延時間と等しくなるようにした。これは、同じ演算回路を 2 つ用いて片方がプリチャージ期間中にもう一方が演算を行う方法で自己同期的システム [13] として動作させることを想定したシミュレーションとなっている。プリチャージに要する時間はトランジスタ数も少ないため、演算時間よりも短くなるはずである。演算時間 t_{EV} に比べ、プリチャージ時間 t_P が極端に短い場合はリーク電力が大きく見える可能性があるため、シミュレーションにより電力値の確認を行った。その結果を図 2.10 に示す。緑色の線 (delay=pre) が演算時間 t_{EV} とプリチャージ時間 t_P を共に遅延 td の平均と等しくした時の値で本研究で電流値測定に使用した波形であり、赤色の線 (pre=fast) はプリチャージ時間はプリチャージに要した時間に定めたときの波形である。動作電圧が高い点においては、2 線とも似た傾向を示していることがわかる。値が違うのは、プリチャージ時間を長くとればとる程リーク電力を多く含んだ場合の電力値になり相対的に小さくなってしまうからである。動作電圧が低い点においては、2 線の値がだんだん近づく傾向になっている。これは動作電圧が低くなるにつれてリークによる電力消費が大きくなってくるために、リークのみが流れる時間を多くとったとしても時間による平均の電力量は近い値になってくるためと考えられる。以上の結果より、プリチャージ時間 t_P と演算時間 t_EV を等しい時間として電力量を求めても、ダイナミック電力とリーク電力の両方をきちんと加味したグラフが作成できることが確かめられた。

2.2.3　回路におけるトランジスタの分類

それぞれの NAND 回路において、最適化を行う個々のトランジスタサイズの分類を図 2.11 に示す。プリチャージ用の pMOS と入力の nMOS およびプリチャージの nMOS、出力インバータに分類されている。入力の nMOS トランジスタに関しては、各トランジスタの傾向が近いものであったため、同一のトランジスタサイズと定めて最適化を行った。

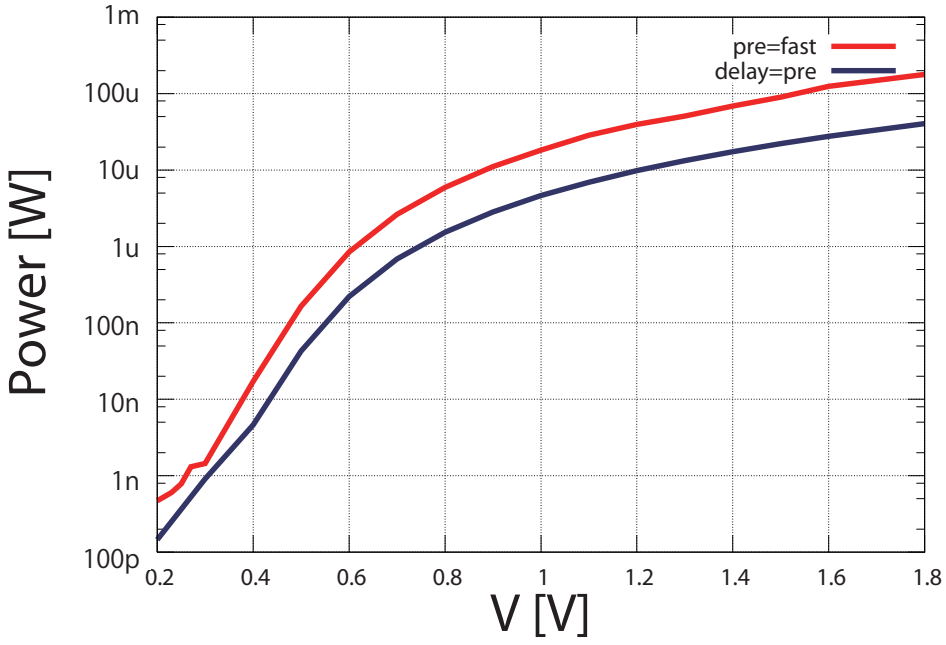


図 2.10 プリチャージ時間を変化させた時の電力値比較.

2.2.4　最適化手法

本研究では、遅延、電力、電力遅延積、動作限界電圧の 4 つの指標を定め、それぞれの指標において最適化を行った。最適化においては、前節で述べたトランジスタの分類に基づき、トランジスタサイズを変化させた際の値と最適化を行う指標のグラフを作成し、そのグラフの極小値を得るトランジスタのサイズを最適値と定めた。

動作限界においてはばらつき耐性の強い回路が低電圧動作に良いと考えた。そのため、最適化はまずしきい値付近である 0.5V においてトランジスタサイズと遅延のグラフを作成し、そのグラフにおいて傾きが小さい点を最適とした。傾きが小さい点とは、ばらつきによりトランジスタサイズが変化した際でも遅延トランジスタの遅延の変化が小さい点になるため、ばらつきに強い点を採用できていると考えた。そして、最適値においてしきい値と移動度ををばらつかせたモンテカルロシミュレーションを行い、その最適値を有するトランジスタでの動作限界を求めた。モンテカルロシミュレーションは 10000 回行い、出力が 0 のところが立ち上がってしまうなどのエラーが一つでも起こった場合、そこが限界の電圧とした。その後、求まった動作限界において再度同様に遅延の傾きを最適とする最適化を行い、それを繰り返すことで回路自身の動作限界を

求めた。

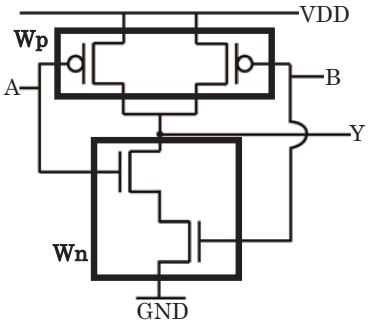
2.2.5　最適化の汎用性

本研究で用いた最適化は、大きな演算回路中に含まれることを想定した 1 つの回路について最適化を行っている。つまり、例えば最適化を行った回路を直列に並べた場合、それぞれの場所において遅延が小さくなる回路を用いているので何段直列に並べた場合であっても最適値に変化はない。同様に、電力で最適化した回路に関しても、ダイナミック電流に関しては最適値に変化はないと考えられる。直列に段数を増加させた場合における待機時間のリーク電流値に変化があると思われるが、動作電圧が高い点においてはリーク電流は小さく無視できるため最適値に影響しないと考えられる。動作電圧が低い場合においてはリーク電力が無視できない程大きくなるが、図 2.10 で示したようにリークを加味した値で最適化が行われていると見做した。

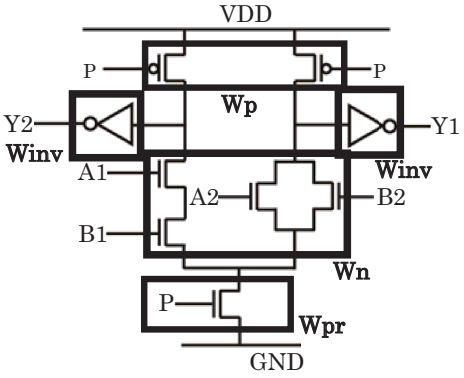
回路の縦積みの段数を増やした場合においても、一つの演算回路を 1 つのセルと考えることで同様に FO4 の回路を作成し 1 段分においての遅延および電力を測定する方法を最適化に適用できる。縦積み段数が増えるごとに、演算時の各トランジスタ特性が変わるため、並列のトランジスタサイズは同じとしたが、縦に並んでいるトランジスタに関しては分けて最適化を行った。遅延や電力における最適化は、2 入力 NAND 同様にトランジスタサイズを変化させてそこで最小の値をとるものを採用しているので回路として最小になる各々のサイズを見つけることができる。また、動作限界においても、縦積み段数を 2 段から 5 段までにおいて同様に最適化を行った結果、各電圧において最適化することでモンテカルロシミュレーションにおける動作限界を低くすることができた。しかしながら、入力を増加させるにつれて動作限界電圧は大きくなった。これは、例えば 5 入力 NAND には、縦積み 5 段のものと相補的に横に並列に 5 段並ぶものがある。ここに”10”の入力を 5 つ入れたとする。縦積み 5 段がディスチャージして”01”を出力しないといけない。しかし動作電圧を低くするにつれ遅延は大きくなり縦積み 5 段の反転に要する時間は増大する。それに対し、並列に 5 段並んでいる回路は動作電圧が低くなるにつれリークによる影響で値を維持できずディスチャージするまでの時間は短くなる。そのため 0 の値を維持しなければならない方が先に立ち上がってしまうというエラーが生じる。以上のことが原因により、縦積み段数が増加すると動作限界は低くなっ

たと考えられる。

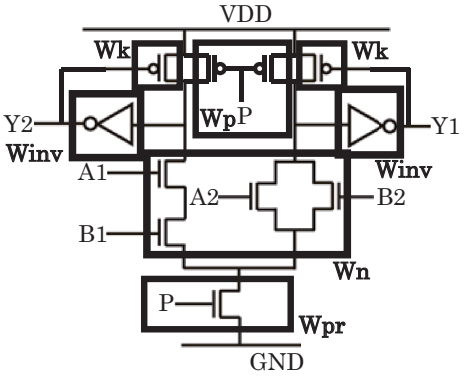
同期回路では、低電圧動作をさせるためにバックバイアスをかけ、しきい値を変動させる手法がとられる。同様に、2 線 DCVSL 回路に関してもバックバイアスをかけた場合を考えた。バックバイアスを印加することで、遅延における nMOS 回路の最適値に変化が起きた。しかしながら、電力の最適値および電力遅延積における最適値に変化はなかった。これより、遅延の小さい回路の場合はバックバイアスに関しても考慮に入れて最適化が必要であるが、電力および電力遅延積に関して最適化を行う場合は印加せずに最適化を行い、その後動作環境に応じてバックバイアスに印加する電圧とリーク電流について調節するだけで良いことが分かった。つまり、電力および電力遅延積に関してはバックバイアスにを考えずに本最適化を行えるということがわかった。



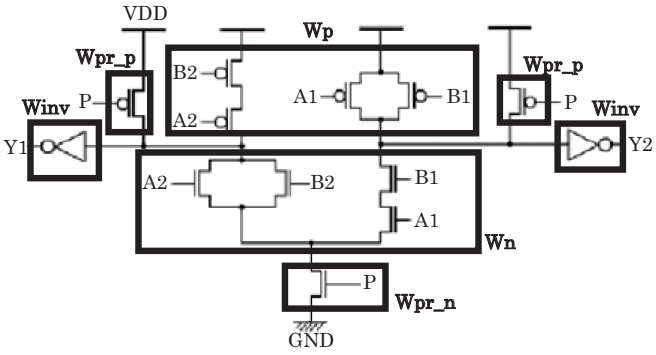
(a)1 線式 CMOS 型 NAND 回路



(b-1)2 線式 DCVSL 型 NAND 回路



(b-2)2 線式 DCVSL 型 NAND 回路
(keeper 有り)



(c)2 線式 CMOS 型 NAND 回路

図 2.11 3 種類の NAND 回路におけるトランジスタの分類.

第3章

低電圧動作向けダイナミック回路の実現と評価

本章では、最適化を行ったそれぞれの回路のシミュレーション結果を示す。自己同期回路は、中の回路さえきちんと動作すれば、クロックのタイミングを気にすることなく動かせる。そこで第1節では、動作限界の電圧を一つの指標として最適化を行った回路について記述する。前章で述べた最適化を行ったほか、その回路の動作限界をモンテカルロシミュレーションで求める他、発振回路を用いて求める手法も提案している。第2節では、遅延、電力、電力遅延積の3つの指標を用いて最適化を行った回路をシミュレーションより各々の指標で比較を行い、最適化を行う優位性を示した。

3.1 最低動作限界電圧において最適化した回路

3.1.1 最低動作限界電圧を指標とした最適化

最低動作限界電圧を指標として、1線CMOS式NAND回路、2線DCVSL式NAND回路、2線CMOS式NAND回路の3回路において最適化を行った結果を以下に示す。図2.11の(b-2)の2線式DCVSL回路のキーパーは、リーク電流により出力が1になるのを防ぐためにおいている。しきい値付近の0.5VでWを調節し、そのWで動作最低電圧を調べ,さらにその電圧でWを調整し、動作の限界まで動作するWを定めた.最適化の手法は,トランジスタの幅をリニアに動かし幅に対する遅延の大きさの変動が少ない部分の中で遅延が小さいトランジスタ幅を最適と決定した。決定する際、モンテカルロシミュレーションの下で動作することを第一に行い、回路の面積は考慮しなかった。

回路により動作限界があるが、その電圧で再度最適化をすることでより動作限界電圧の低い回路が導けている。1線式CMOS型回路の最低動作限界電圧の最適化の結果を図

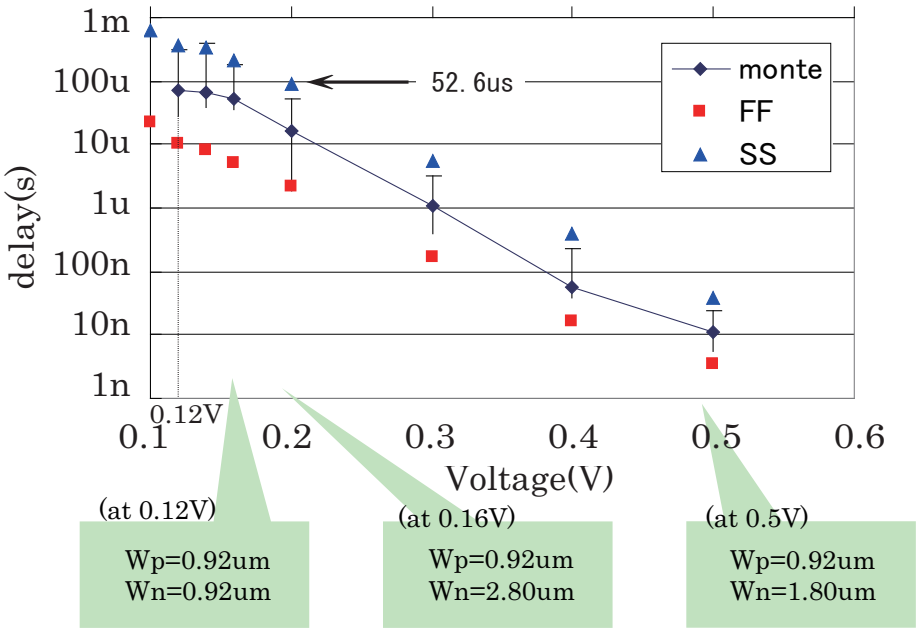


図 3.1 1 線 CMOS 回路の動作限界最適化

3.1 に示す。FF(fast-fast)/SS(slow-slow) シミュレーションでは 0.1V まで動作したが、ばらつきをもたせたモンテカルロシミュレーションでは 0.12V までしか動作しなかった。同様に, キーパー無しの 2 線 DCVSL 回路の最適化結果を図 3.2 に、2 線式 CMOS 回路の結果を図 3.3 に示した.

3 回路のシミュレーション結果より、ばらつきを持たせた際に最も低電圧で動く回路は 0.11V である 2 線式 CMOS 回路であった。しかし、2 線 DCVSL は 0.16V まで、1 線式 CMOS は 0.12V まで動作しており、特に 1 線式 CMOS においては 0.01V 差とほぼ同等の電圧までばらつきに対して動作していた。また、遅延時間では 1 線式 CMOS の最悪遅延は 2 線 CMOS の平均遅延よりも大きいため、遅延が最も小さいものは 2 線 DCVSL であり、最も大きいものは 1 線 CMOS であることがわかった。各々の回路のトランジスタのサイズだが、閾値付近ではトランジスタ幅を大きくする方が相対的にばらつきが小さくなり良しとなったが、限界電圧付近では大きさだけでなく、pMOS および nMOS の大きさのバランスが重要になってくることがわかった。

また、この測定を通して、低電圧動作をさせる場合、入力はツリー型をした単純な

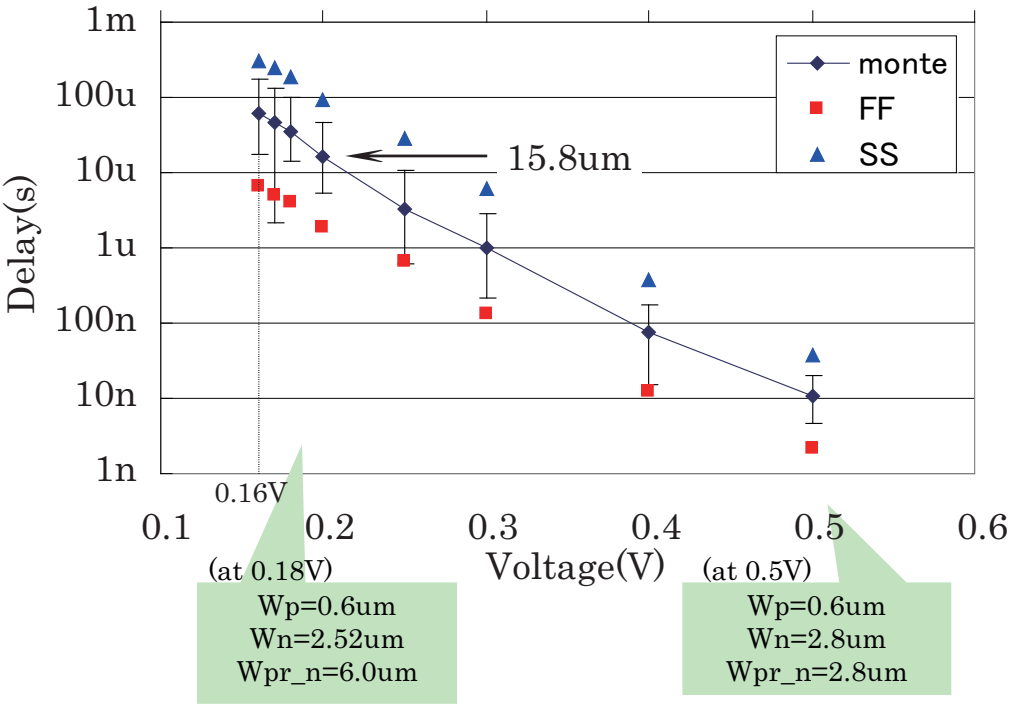


図 3.2 2 線 DCVSL 回路の動作限界最適化

バッファを用いた方が低電圧動作に向いていたが、出力バッファに関しては既存のバッファを用いても波形を確認できることがわかった。本チップ以降は、出力には既存のバッファを用いることにした。

3.1.2　最適化を行った回路の設計

低電圧におけるモデルファイルの正確性を確認するため、またモンテカルロシミュレーションの際に用いたばらつきが妥当であったか確かめる為に、最適化を行った回路のチップ作成を行った。

シミュレーションでは 1 回路分の遅延を測定したが、実際に測定する際は図 3.4 の様に FO4 の回路を 20 段並べ 1 段目 2 段目および 20 段目の出力を見れるようにし、1 段分の遅延と 20 段の平均遅延の両方を測定できるようにした。これにより、1 段分のシミュレーションの比較と共に、20 段の方ではトランジスタがばらつきを持った場合も動作するかを確かめることができる。また、既存の IO では 0.1V まで測定できるかが不明で

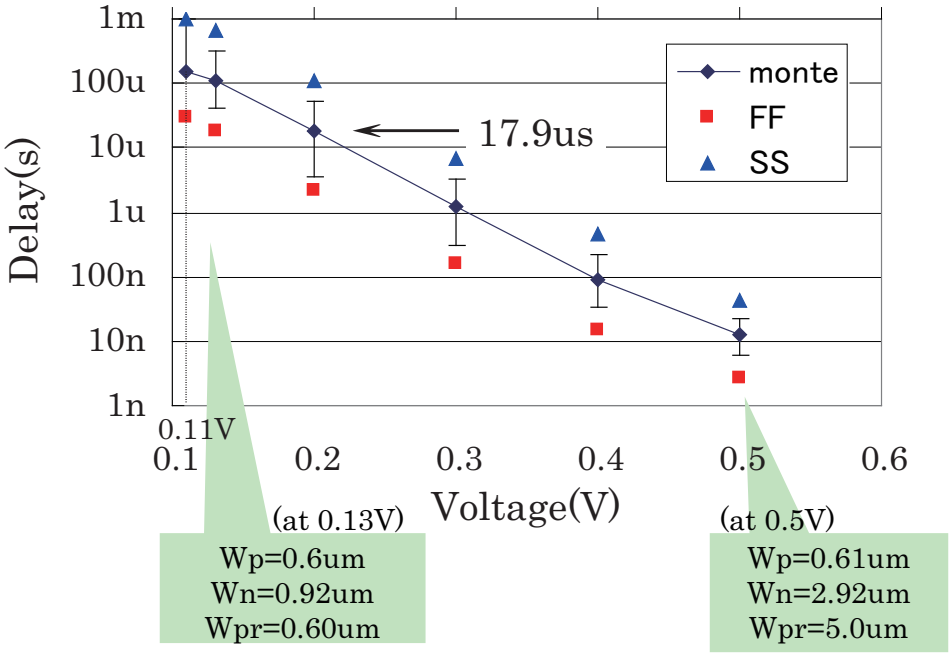


図 3.3 2 線 CMOS 回路の動作限界最適化

あった為、バッファ部分は自分で作成した。作成したバッファは、IO に ESD 対策を行えるような VDD 側と GND 側の両方にダイオードのみ付いたものを採用し、入力側にはインバータを tree 型にしたバッファを出力には 10~20pF 程度のコンデンサを挿入しても動く程度のバッファを用いた。

実際に作成したチップには、既存の IO バッファを使用した回路と、0.1V で動くような自作のバッファを入れた回路との 2 種類が入っている。これは、自作のバッファはあくまで 0.1V レベルの低電圧を測定する為であり、既存の IO は 1.8V からある程度の低電圧まで測定できる保証があるため、ある程度の低電圧までは既存の IO を用いたもので測定できるようにするためである。また、消費電力を測定するために 20 段繋がっている回路の供給電源は負荷として用いている他の回路の供給電源とわけて設計した。

3.1.3　発振回路を用いた動作限界の検討

動作の限界電圧を知る方法のひとつに、発振回路を作成しそれが発振し続けるかを確かめる方法がある。同期回路において発振回路とは、インバータチェインにより作成

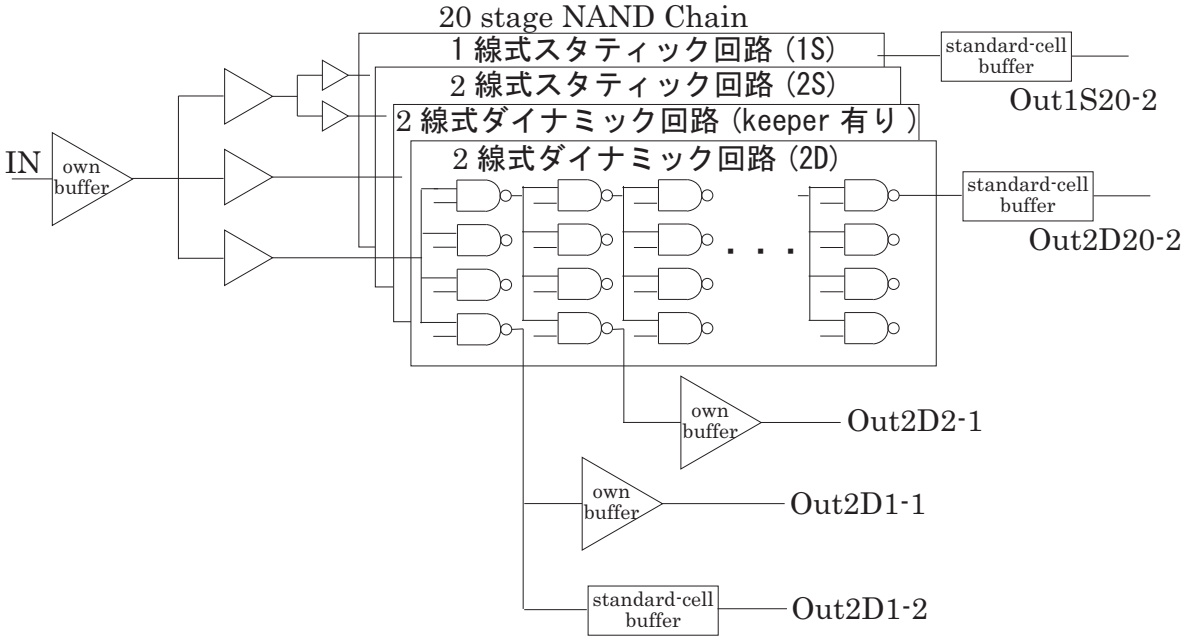


図 3.4 作成したチップの回路図

できる。しかし、自己同期回路ではインバータチェインの作成はできないため、終了検出を用いて発振させる回路を用いた。そこで、DCVSL による NAND 回路を用いたリング発振回路による評価を行った。1 つのセルを図 3.5 に示し、全体における発振回路の回路図を図 3.6 に示した。図 3.5 に示すように、同じ回路を 2 つ用意し、片方にプリチャージをもう片方にプリチャージと逆の信号を与えることで、2 つの演算回路のうちだちらかが必ず演算を行うような回路になっている [13]。また、演算回路のどちらかの出力が high になることで終了検出の出力 CD が反転するような構成になっているため、常にどちらかが演算を行っている動作率の高い回路が構成できる。図 3.5 の回路を図 3.6 のように 3 つ繋げ、最終段には終了検出のみして値を前の段に返す回路を構成することによって、発振回路を作成した。また、終了検出によりプリチャージ信号が変化するよりも前に、演算において両方の出力が high になるエラー状態を検出するために出力の両方が 1 になった時のみ出力が立ち上がるエラー検出回路もつけ、発振するかだけでなくエラーによる動作限界の確認もできるようにした。

まず、この発振回路を用いて周波数特性を調べた。その結果を図 3.7 に示す。青線が発振回路の周波数であり、緑線は NAND の遅延の逆数として得られた周波数である。この結果より、NAND 回路単体の逆数に比べて発振させた場合の周波数は 10 倍以上遅

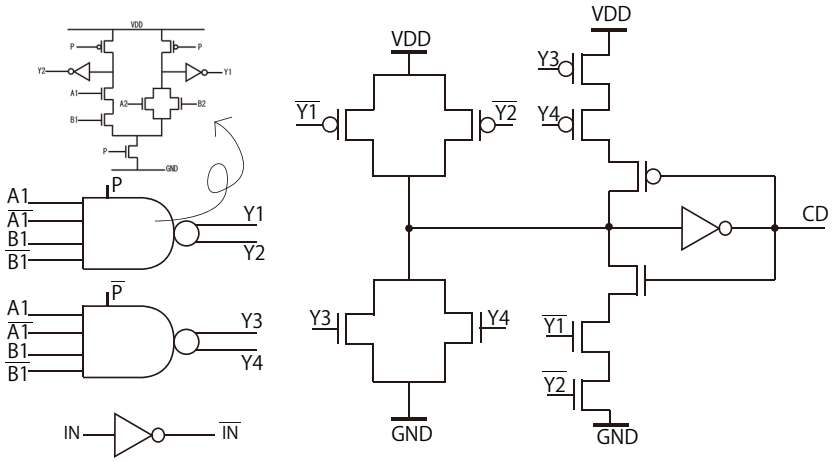


図 3.5 DCVSL による NAND 回路の終了検出回路図

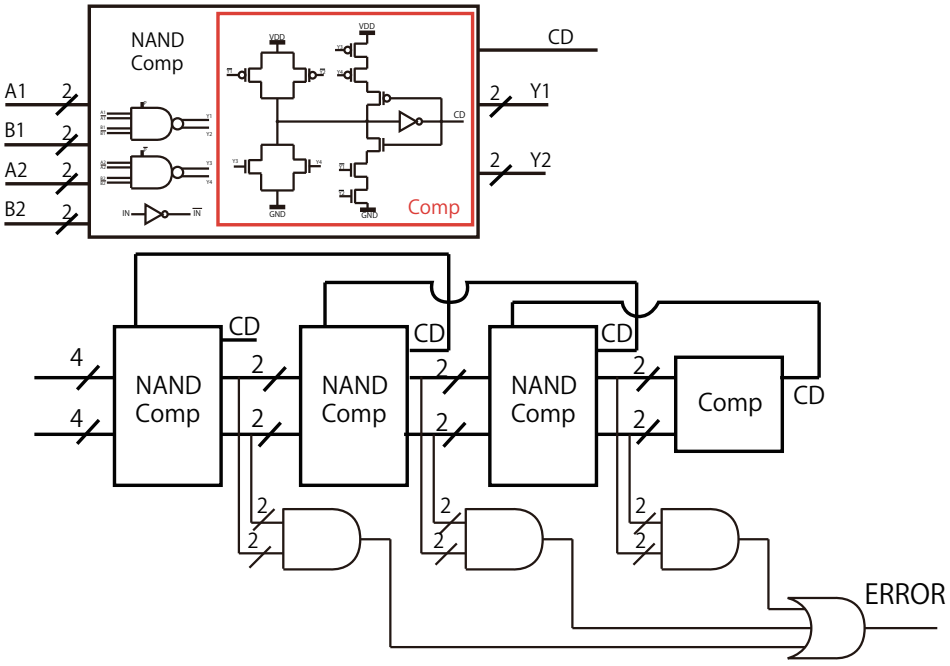


図 3.6 2 線 DCVSL 回路によるリング発振回路のブロック図

い結果となっていた。これは、NAND チェインは完全に立ち上がる前に次の段に出力が伝達しており 1 段目が立ち上がってから 100 段目が立ち上がるまでの時間を 99 段分で割った値を使用しているのに対し、発振回路では次の段の終了検出の値が反転して戻ってくるまでの時間が遅延となっているため、2~3 倍の時間がかかってしまうことによる。そのため、終了検出部分に関しても最適化を行う必要がある。

この発振回路を用いて、最低動作限界電圧を指標に最適化を行った回路に関しての動

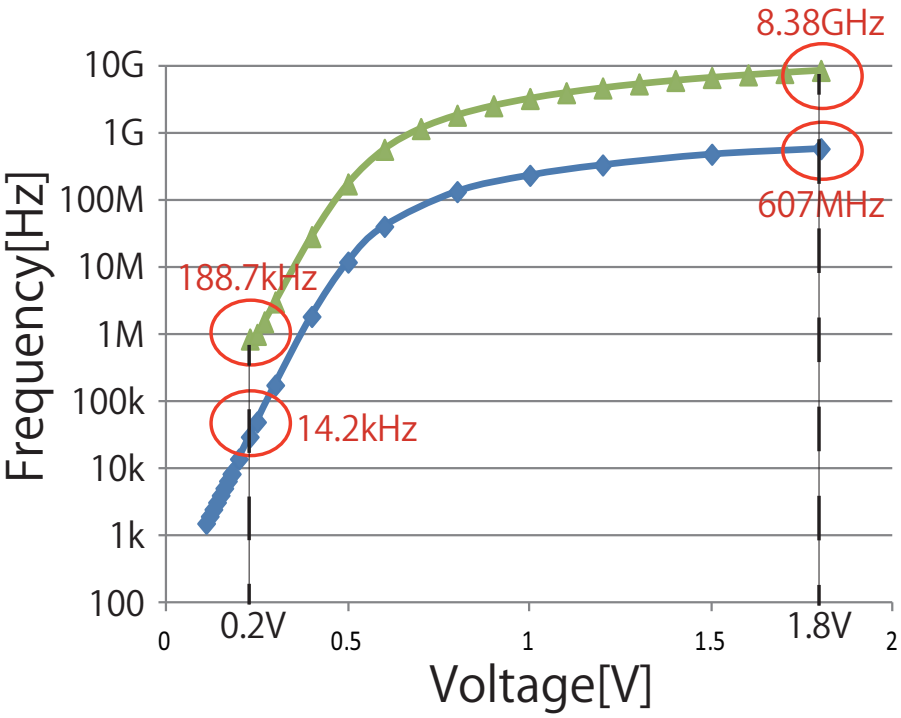


図 3.7 発振回路の周波数特性のシミュレーション結果

作限界について調べた。最適化を行った回路の発振回路を作成し、モンテカルロシミュレーションを行った。先に述べたように終了検出にかかる時間にも動作が依存するため、終了検出部分に関しては先に最適化を行っている。その結果、動作限界は 0.18V であり、前節で行ったシミュレーションの動作限界電圧 0.16V に比べて高い電圧値になった。これは、前節で行ったモンテカルロシミュレーションにおいては、1 をとる出力の立ち上がりが、リークにより 0 である出力の立ち上がりに比べて速ければ動作しているとした。しかしながら、発振回路では終了検出をしている期間は値を保持しつづけなければいけない。また、発振回路においては終了検出の部分のばらつきによって保持しなければならない時間が長くなる。これらの原因で制約が厳しくなるため発振回路においてのモンテカルロシミュレーションでの限界電圧は大きくなったと考えられる。実際に自己同期的に動作させる場合は、このような終了検出をつける必要があるため、モンテカルロによる動作限界の値は発振回路を用いて得られた値の方が実際に近いと思われる。

3.1.4　限界電圧の理論計算

動作限界について、理論的な限界を調べるために、流れる電流の値を pMOS と nMOS で比較して算出した。まず最初に、単純なインバータに関して考えた。通常、2 線式回路においてインバータは線を逆につなぐことで実現できるため使用することはないが、プリチャージを必要とするダイナミック回路の簡単な例として今回は使用した。インバータの回路図を図 3.10 に示す。出力に駆動力を上げるためにインバータをとりつけた為、実際はバッファ回路になっている。

まずはキーパーのない (a) の回路図について考えた。図に示すように 3 つのトランジスタに流れる電流をそれぞれ I1、I2、I3 とした。それぞれの電流において、$I = \frac{W}{L} uC_0 exp(\frac{q(V_{gs} - V_{th})}{nk_BT})$ の式を用いて電流値を算出し、P=0,IN=1(プリチャージ)、P=1,IN=1/0(演算) においてそれぞれ動作限界に関する式を求めた。

まず、トランジスタサイズを全て等しくした場合において、exp の中の n の値を変化させた際の図を図 3.8 に示した。n の値が大きくなると動作限界電圧の計算による理論値は小さくなるという結果が得られた。理想的なトランジスタ構造の場合 n=1 だが、実際はそれよりも大きな値をとるため、以降の式では n=1.4 という値を用いて計算した。

理論値計算の結果、室温 (T=27 度) 環境下において IN=0 のときにリークで反転しない条件 $W_1 \geqq 0.282W_2$ の下で、演算可能な動作限界電圧は $V > 0.0458 + \frac{log(\frac{W_p}{W_{IN}})}{log(9.98 \times 10^{11})}$ より求まることがわかった。ここで、$\frac{W_p}{W_{IN}}$ の値を変化させた際の限界電圧値の変化を図 3.9 のに示す。図からもわかるように、W_{IN} が大きくなるにつれ動作限界電圧は低くなっている。また、W_{IN} を大きくすることで演算時に流れる電流も大きくなり結果的に演算時間は小さくなる。つまり演算時においては $\frac{W_p}{W_{IN}}$ が小さい方が良いが、リークで反転しない制約条件があるため、限界値は $V > 0$ となり、$W_p = 0.282W_{IN}$ とした時は電圧が少しでもかかると動作するということがわかった。トランジスタ幅の条件式の符号がイコールの時は pMOS 側と nMOS 側の電流が釣り合う時であり、かかる電圧に依存せず動作しない点である。そのため、トランジスタ幅を限界条件で決定すると動作しない可能性があるため、マージンを持った設計が必要であると思われる。しかしなが

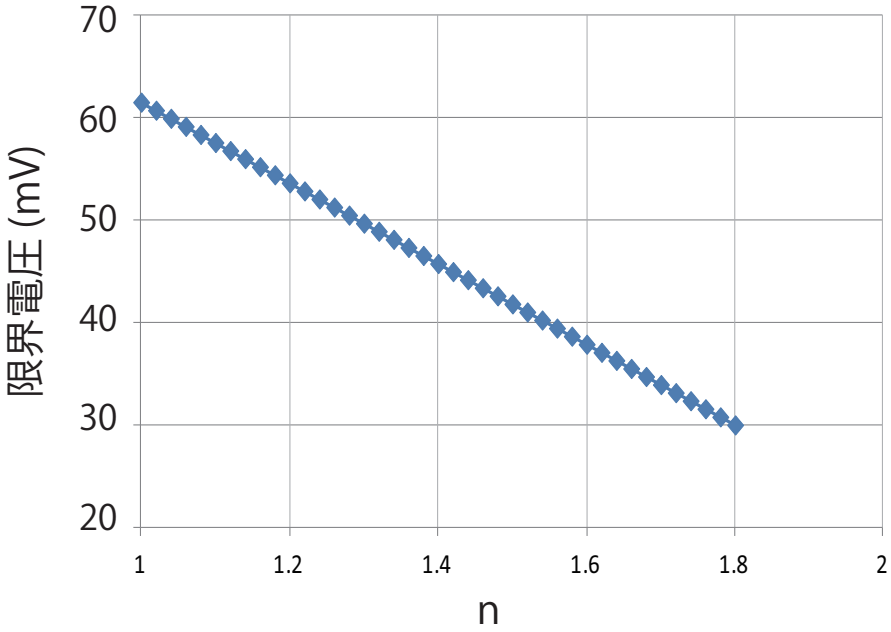


図 3.8 $exp(\frac{q(V_{gs}-V_{th})}{nk_BT}$ の n の値と動作限界理論値の関係

ら、トランジスタサイズを調節することで限界電圧は低くでき、実際の限界はばらつきや雑音等で決定されると考えると、CMOS 型と同等の限界電圧を有する回路の作成が可能であると想定される。

ここで、3 つのトランジスタのサイズが等しい ($W_p = W_{IN} = W_n$) と仮定したときについて考えた。演算時において値を維持できる限界電圧は 0.0458V であった。この時チップばらつきによってトランジスタ幅がそれぞれ 10% ばらついた時を想定して計算すると、限界電圧は $0.053V$ に上昇したが、その電圧以上であれば動作することがわかった。また、ばらつきがなかった場合の動作限界電圧値において、プリチャージの条件より nMOS 側のプリチャージトランジスタは pMOS 側のプリチャージトランジスタの 12.5 倍よりも小さくなければならない ($(\frac{W}{L})_n < 12.5(\frac{W}{L})_p$) という条件が得られた。また、IN=0 のときにリークによってプルダウンされてしまわない様な条件は、プリチャージトランジスタに対し IN のトランジスタサイズは 3.54 倍より小さくないといけない ($(\frac{W}{L})_{IN} < 3.54(\frac{W}{L})_p$) 結果が得られた。

次に、キーパーをつけた図 3.10 の (b) の場合につてい考えた。演算の nMOS トランジ

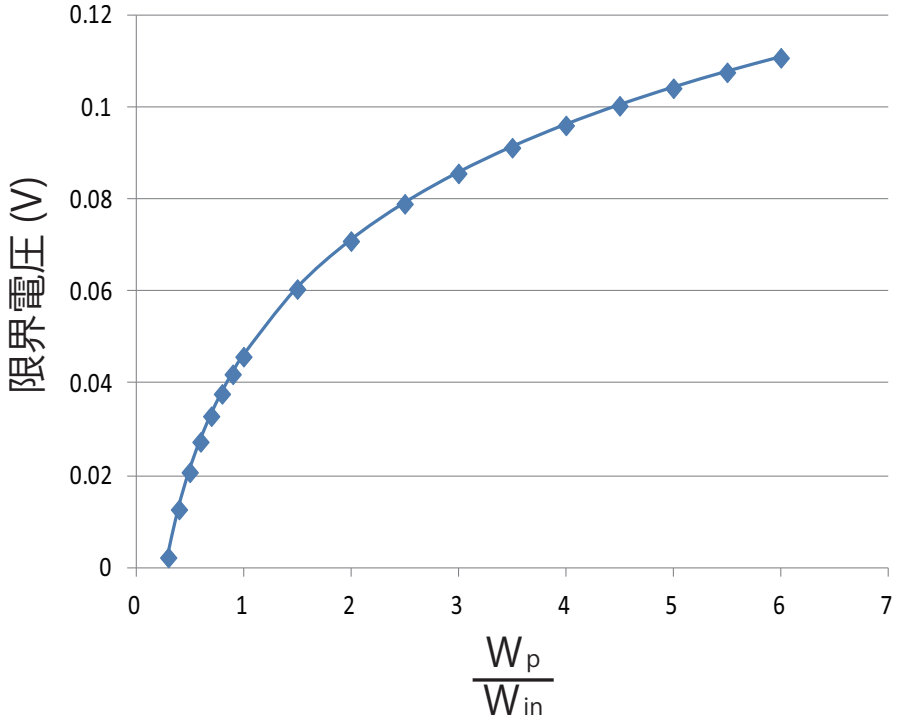


図 3.9 $\frac{W_p}{W_{IN}}$ を変化させた際の限界電圧値

スタは大きい程遅延は小さくなるが逆にリークによるプルダウンによって動作限界電圧を上げてしまう。そこでキーパーを用いることで、上記で述べた 3.54 倍より大きく遅延最小の nMOS トランジスタを用いても、リークによるプルダウンを防ぐことができる。キーパーのサイズは大きい程リークによるプルダウンを防ぐことができるが、大きくなりすぎると IN=1 の時のプルダウンを邪魔してしまう。その為、nMOS のサイズに依存してサイズの制約が決定し、nMOS が pMOS の 3.54 倍の大きさである時のキーパーの制約条件は、pMOS トランジスタの 0.717 より小さくないといけない ($(\frac{W}{L})_k < 0.717(\frac{W}{L})_p$) ということがわかった。ここで、キーパーは pMOS に比べて $\frac{W}{L}$ を小さな値にするために、通常は L を大きくすることで調節を行っている。

NAND 回路に関して同様に図 3.11 の様に電流値を定めて行ったところ、$W_p \geqq 0.5636 W_n$ であり、等号が成り立つ時の電圧は $V > 0.025$ という結果が得られた。しかし、ばらつきによって大きさが変化することで動作しなくなる危険がある。そこでトランジス

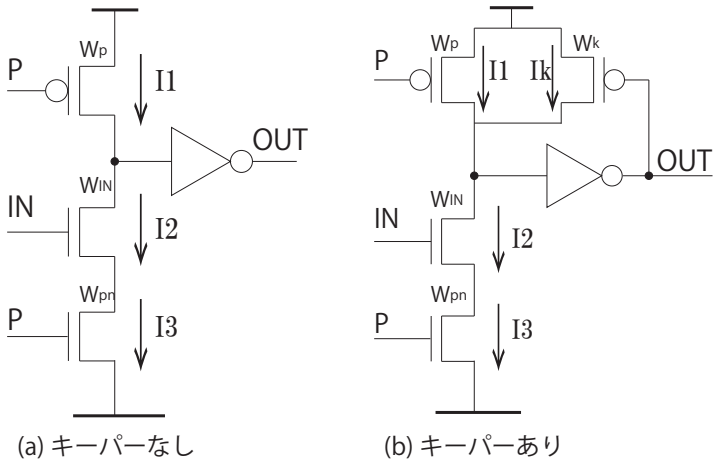


図 3.10 理論値計算のためのダイナミック型インバータ回路図

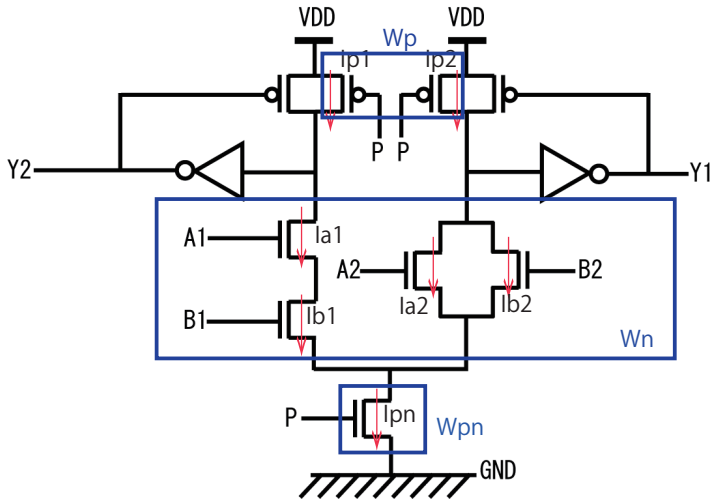


図 3.11 理論値計算のためのダイナミック型 NAND 回路図

タ幅が全て等しい時の限界電圧を求めると 0.0458V という結果が得られた。また、入力に”10”と”10”を入れた際の出力の立ち上がりが”11”のエラーとなる条件より、Y1 出力用並列の nMOS トランジスタサイズは Y2 出力用直列 nMOS トランジスタサイズの 1.77 倍より小さくなければならないという条件が得られた。今回行った最適化では演算用 nMOS トランジスタサイズは等しいものと仮定して最適化を行っているためこの等式を満たしている。

今回用いた式に関して妥当であるか確認するため、計算により求めた電流値 I を用いて遅延 $t = \dfrac{V_{DD}C_L}{nI}$ を用いて算出した。また、今回 C_L の値は 5f として、nMOS と pMOS

トランジスタサイズを等しいとした場合において計算した。この計算した遅延結果と、シミュレーションで求めた遅延結果を図 3.12 に示した。計算においては、電流の式がしきい値以下で成り立つため、0.4V 以下においての結果を示した。またシミュレーションにおいては、動作した電圧の 0.11V までプロットした。図 3.12 の結果より、0.2V 以下ではゲート電圧に加えてドレイン電圧による影響が大きくなるためシミュレーションの遅延は少し大きい値となっていたが、0.2V~0.4V にかけてはほとんど一致している為、この計算式はシミュレーションと比較しても近似できているということがわかった。

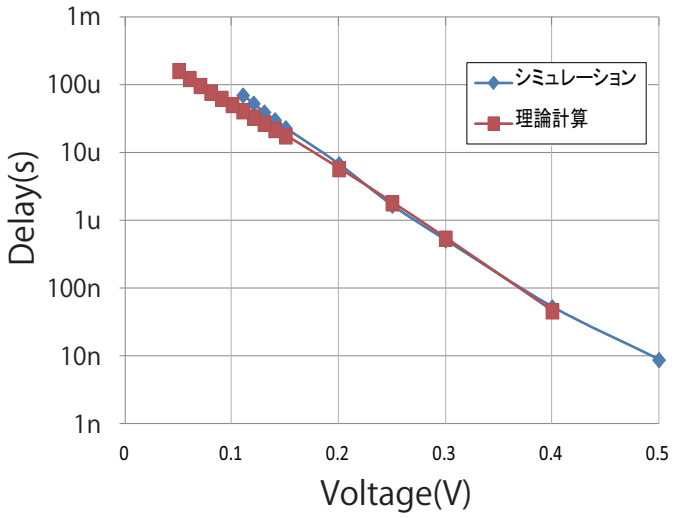


図 3.12 理論値計算のためのダイナミック型 NAND 回路図

しかしながら、動作限界に関してはシミュレーションと比べて差がある。これは、シミュレーションにおいて W の値によってモデルファイルが異なり、特に動作電圧の低い点において顕著にその差が出るため、W による誤差の補正を考えた。そこで、nMOS トランジスタ及び pMOS トランジスタにおいてシミュレーションにおける電流値から、その際の理論計算における W の値を求めた。動作電圧 0.2V の点において、$W_p = 0.22um$ の pMOS トランジスタに流れる電流値は 9×10^{-13} であり、これを今回用いた電流値の式に代入して W_p を求めると 0.213um と求まった。次に同様に $W_n = 0.22um$ の nMOS における電流値は 6.98×10^{-10} であり、計算式より求まった Wn は 2.34um と実際の 10 倍以上の値になることがわかった。また、nMOS トランジスタは動作電圧が低くなるにつれこの差は大きくなっていた。その為、入力 0 の時のリーク電流は今回計算で用いている値に比べて 10 倍の大

きさとなり、シミュレーションにおいて値がディスチャージされ出力が 1 になり限界電圧が高い点になっていたと考えられる。そこで、シミュレーションから得られる値を用いて限界電圧を求めた。限界電圧は $W_p\mu C_{oxp}exp(\frac{q(-V_{th})}{nK_BT}) < W_n\mu C_{oxn}exp(\frac{q(V_{gs}-V_{th})}{nK_BT})$ を用いて求める。今、pMOS トランジスタにおいて、リーク電流 $W_p\mu C_{oxp}exp(\frac{q(-V_{th})}{nK_BT}) = 900[fA]$ で左辺は決定する。また、動作電圧がしきい値と同じ 0.425V のときの nMOS トランジスタの電流値が 282[nA] から $W_p\mu C_{oxn}$ の値が、動作電圧が 0V の時の電流値 3.8[pA] から $exp(\frac{q}{nK_BT})$ の値が求まる。この値を代入して Vd の値を求めると、$V_{gs} > 0.121$ という結果が得られた。これより、シミュレーションより得られた動作限界である 0.11V と近い値が一次近似の式から得られた。また、この計算値とシミュレーションの値においての 10[mV] 程度の差は、ドレインーソース間による影響を無視していたためであると考えられる。さらに、実測で測定を行う場合は、kTC ノイズなどの雑音が動作の邪魔をするため、実測における動作限界電圧は、さらに数十 mV 程度は変わるであろうと想定される。

最適化をするにあたり、このように一次近似の理論計算によって制約条件や動作限界電圧を求めることができる。今回のシミュレーションにおける動作限界はモンテカルロシミュレーションを用いて行ったが、この様に簡略化した計算を用いることで時間を短縮できるだけでなく、動作電圧が低くシミュレーションの精度が曖昧な点においてもシミュレーションと近い値を求めることができるとわかった。以上の結果より、ここで用いた一次近似における計算方法は、最適化をするにあたり一つの役に立つ指標になるであろうと思われる。

3.2　遅延、電力、電力遅延積を指標として最適化した回路

3.2.1　最適化した回路のシミュレーション比較

次に、最適化の指標を遅延、電力、電力遅延積のそれぞれに設定し、同様に最適化を行った。2 線式 CMOS 回路はスタティック回路にすることで低電圧でも安定して動作すると考え比較をしたが、2 線式 DCVSL 回路も最適化で低電圧で動作が保証されることと、2 線式 CMOS 回路は回路が大きいため遅延や電力では不利なため、1 線式 CMOS 回路と 2 線式 DCVSL 回路でのみで上記の最適化を行った。

表 3.1 最適化した NAND 回路のトランジスタサイズ

回路	Wp[um]	Wn[um]	Winv[um]	Wpr[um]
[d1]	0.22	0.22	0.42	6
[d2]	0.22	0.5	1.9	5.7
[p1]	3	1	0.22	0.22
[p2]	3	0.22	0.22	3
[pd]	0.22	0.22	0.22	0.22
[lv1]	0.6	2.8	10.2	2.8
[lv2]	0.6	2.52	10.2	6

各々の指標において動作電圧を 1.8V、0.4V、0.2V で最適化を行った。遅延を指標に最適化を行ったものは、動作電圧によって一意に決まる。そこで、0.2V で最適化したものを [d1]、1.8V で最適化したものを [d2] とした。電力を指標とする最適化では、動作電圧が 0.2V と 0.4V のトランジスタ幅は等しい値になった。これは、1.8V の様に動作電圧が高い部分ではダイナミック電流が主流になるのに対し、動作電圧が低い場所ではリーク電流が主流になるため、高電圧と低電圧で最適な回路ができたと考えられる。同様に、0.2V で最適な回路を [p1]、1.8V で最適な回路を [p2] とした。電力遅延積で最適化したものは動作電圧に依存せず一意に決まった。その最適な回路を [pd1] とした。また、動作限界電圧で最適化した回路は 0.5V で最適化したものを l1、0.18V で最適化したものを l2 とした。最適化を行ったトランジスタサイズの結果を表 3.1 に示す。

各々の回路について、遅延、電力、電力遅延積でシミュレーションをした結果を図 3.13、図 3.14、図 3.15 に示す。図 3.13 より、遅延で最適化を行った回路は他の指標で最適化を行った回路に比べて、0.2V において 5∼10 倍の遅延のの大きさが小さくなっていることがわかる。図 3.14 からは、動作電圧 0.2V において電力で最適化を行った回路がその他の指標で最適化を行った回路よりも 4 割程度にすることができていることがわかった。また、電力遅延積に関しては、最適化からも得られたように動作電圧に依存せず最小の回路は一定であり、動作電圧 0.2V においては電力遅延積で最適化した回路は限界動作電圧で最適化した回路に比べて 8 割も電力遅延積を削減できていることがわかった。以上の結果からも、各々のシミュレーションにおいて遅延、電力、電力遅延

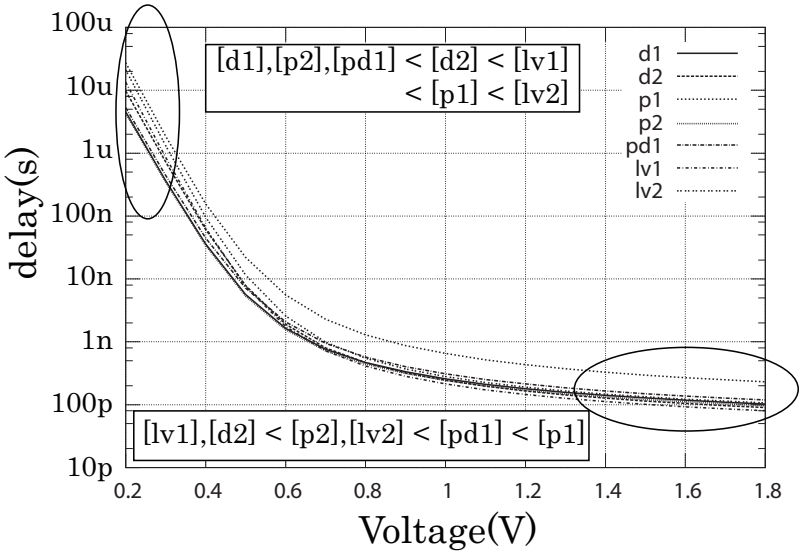


図 3.13 最適化した 2 線式 DCVSL 回路の遅延の比較.

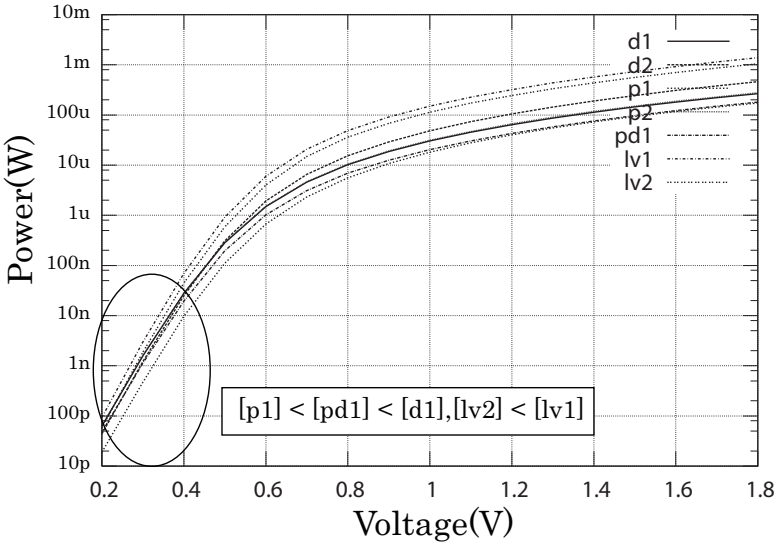


図 3.14 最適化した 2 線式 DCVSL 回路の power の比較.

積のそれぞれで最適化したものが、各指標において最適な結果であることを確認した。この結果より、求められたパラメータにおいて最適なトランジスタが一意に決定され、最適化をすることでより低電圧や低電力な回路を作成できることがわかった。

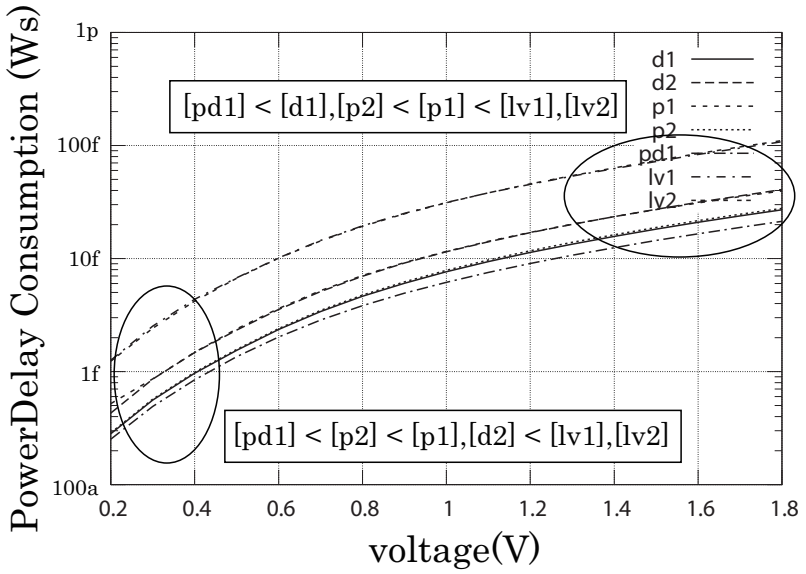


図 3.15 最適化した 2 線式 DCVSL 回路の powerdelay 積の比較.

3.2.2　最適化した回路の設計

前節において最適化を行った回路に関して、特に電流値に関しては低電圧動作におけるモデルファイルの傾向の確認と実測においての最適化の優位性を検討するために、実際にチップの作成を行った。作成したチップの回路図を図 3.16 に示す。

前節のチップは、NAND を 20 段並べた回路であったが、動作電圧が高く遅延が小さい場合においても平均に近い値を得れるように 100 直列に並べた回路を作成した。また、電力や電力遅延積において最適化した回路を比較するため、個々で電流値を測定できるように VDDMEAS を別入力としてそれぞれの回路において作成した。

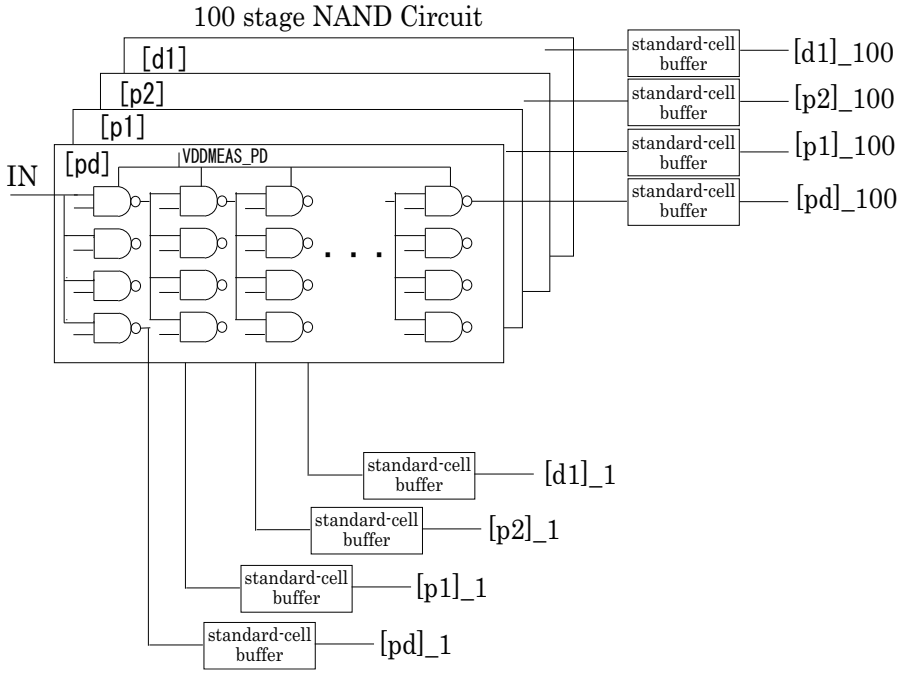


図 3.16 最適化した回路搭載チップの回路図

第4章

低電圧動作向けダイナミック回路の測定結果

4.1　測定手法

測定を行った環境を図 4.1 に示す。ポケットジェネレーター (JDS UPG2116) より各々のピンに波形信号を入力した。また、電源に関しては電力を測定するために必要な VDDMEAS に関してのみ半導体パラメータアナライザ (Agilent 4155C) を用いて電源を入力することで、電源から流れる電流値の測定を行った。また、温度特性を調べるために thermostream(TP04100A) を用いて温度の調節を行った。出力はオシロスコープ (Tektronix TDS3052 2ch 500MHz,Agilent DSO6102 2ch 1GHz 5Gs/s) により観測した。

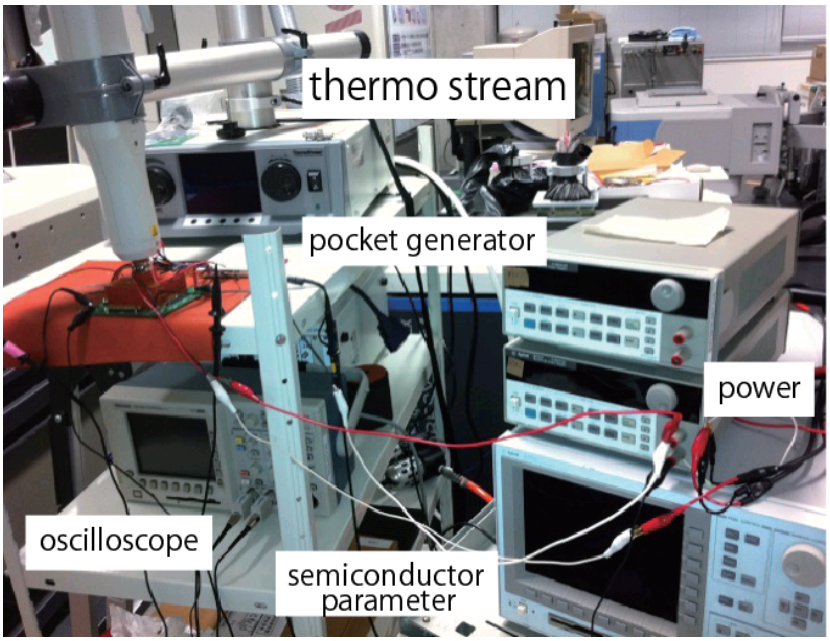


図 4.1 測定環境

電流値は各々の最高動作周波数にて測定した。2.2.2 節の図 2.10 で示したように、プ

リチャージ時間を回路が演算を行うときの遅延と同じ大きさにした場合であっても電流値の絶対値に変化はあるが傾向に変化はないため、シミュレーションと同様にプリチャージ時間は演算にかかる遅延と同じ時間だけ要して動作させた。さらに、低電圧においての電力値の変化および電力遅延積の極小値に関しては実測を行うことでそれぞれの変化について確認を行った。確認については、電力遅延積に関して最適化した NAND 回路を 100 段直列に並べた回路に関して行った。まず、電力値においてプリチャージ時間 t_P を変化させた場合の実測結果を図 4.2 に示す。図より、動作電圧が低くなるほどリークの割合が大きくなるため、プリチャージの待機時間を多くとった場合の電力値の値はプリチャージ時間を短くした場合と比べて差が小さくなっていくことがわかる。さらに、電力遅延積は 100 段もの段数になることでリークの影響が無視できなくなり、結果極小値が存在するグラフになるが、図 4.3 の実測結果より極小をとる電圧に関しても変わらないということがわかった。これは、動作電圧を低い値にすることでリークによる動作がより支配的になるため、プリチャージ終了後にリークのみの時間が含まれたとしても電圧依存特性に関しては変化がないということがわかった。

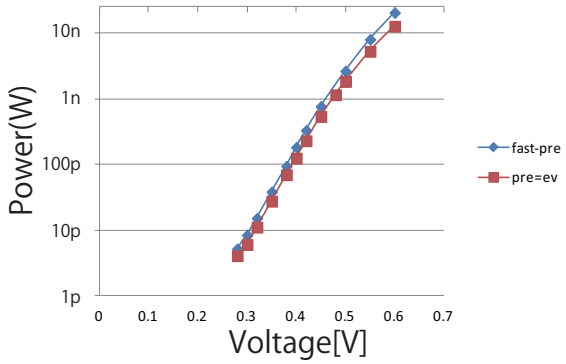


図 4.2 プリチャージ時間を変化させた時の電力値の変化の実測結果

4.2　最低動作限界電圧で最適化した回路の測定結果

4.2.1　試作したチップ

最適化したゲート幅を有するそれぞれの回路のチップを設計した。設計したチップ写真を図 4.4 に示す。既存の IO バッファを使用した回路と、シミュレーションで 0.1V まで動作した自作のバッファを入れた回路の 2 種類がチップに入っている。図 4.4 の左側

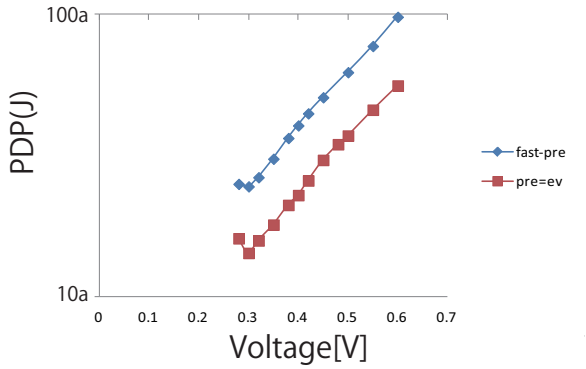


図 4.3 プリチャージ時間を変化させた時の電力遅延積の変化の実測結果

に自作のバッファを入出力に用いた低電圧測定用の回路を、右側には既存のバッファを用いた回路が載っている。

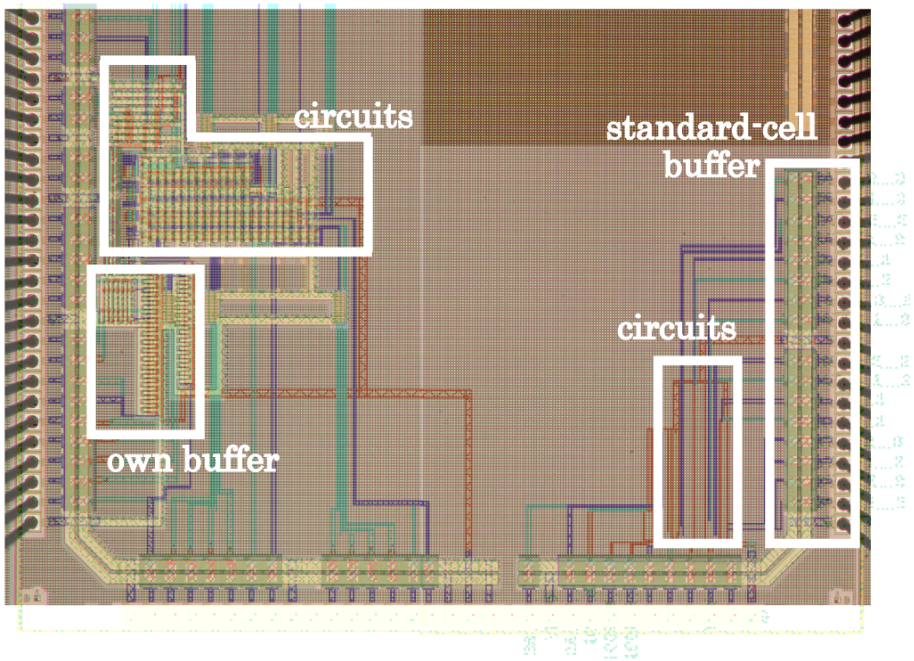


図 4.4 作成したチップ写真.

4.2.2　測定結果

1.8V から 1.0V までの電圧において 19 段分の遅延を測定し、((OUT20-2)-(OUT2-2))/19 とすることで平均の 1 段分の遅延を算出した。また、1.0V 以下では遅延は十分大きい

ので、(OUT2-1)-(OUT1-1) より 1 段分の遅延を測定した。誤差を補正する為に、ともに前後 0.2V 程度かぶらせて測定を行っている。

作成した各々の回路のシミュレーション結果は図 4.5 の様であった。作成したチップについて、測定を行った結果を図 4.6 に示した。実測においては、遅延時間のグラフの形状はシミュレーションと似たものとなった。

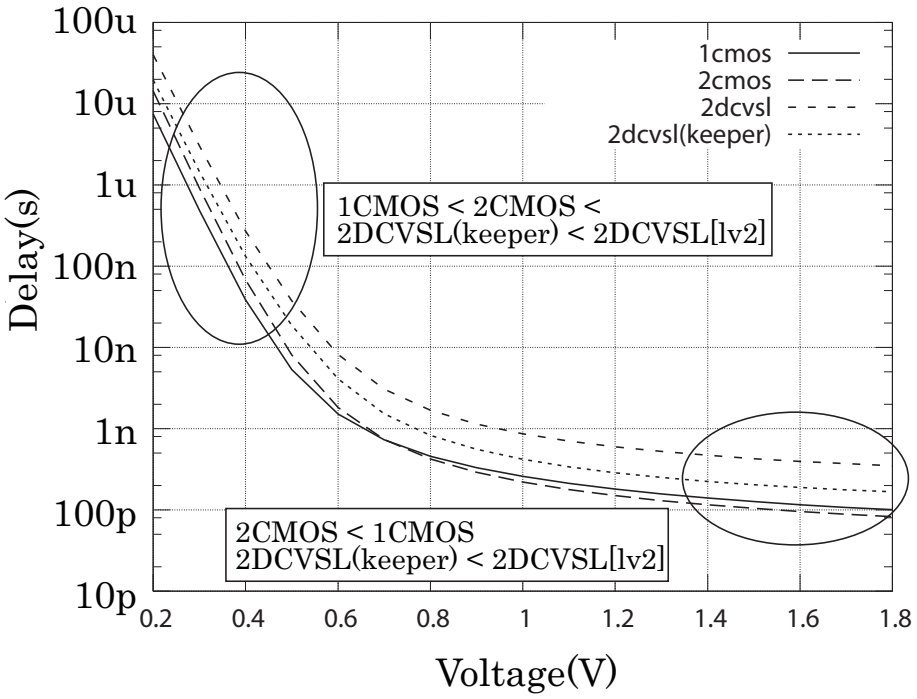


図 4.5 作成したチップのシミュレーション結果.

既存のバッファは 0.5V 以下で正しい出力を得れなかった為、バッファの電圧の調整を行い遅延時間を測定した。その為、バッファの調節した動作電圧が 0.6V 以下の部分で差が生じている。高い動作電圧については、キーパーを用いない 2 線式 DCVSL 回路を除く 3 回路については、シミュレーションと近い測定結果を得た。2 線式 DCVSL 回路は, 別のチップや温度を調節し再度測定を行ったが大小関係に変化がなかった。つまり、シミュレーションとの差異は、レイアウトかトランジスタのサイズにより使用されたモデルファイルが異なったことにより生じたものだと考えられる。既存のバッファで測定した回路はバッファの調節を行うことで、各々の回路において 0.25V まで動作した。

自作のバッファについては, バッファの電圧を調節する必要はなかったが、1 線式 CMOS 回路で 0.3V まで、2 線式の 2 回路では 0.4V までしか動作しなかった。また、0.4V に

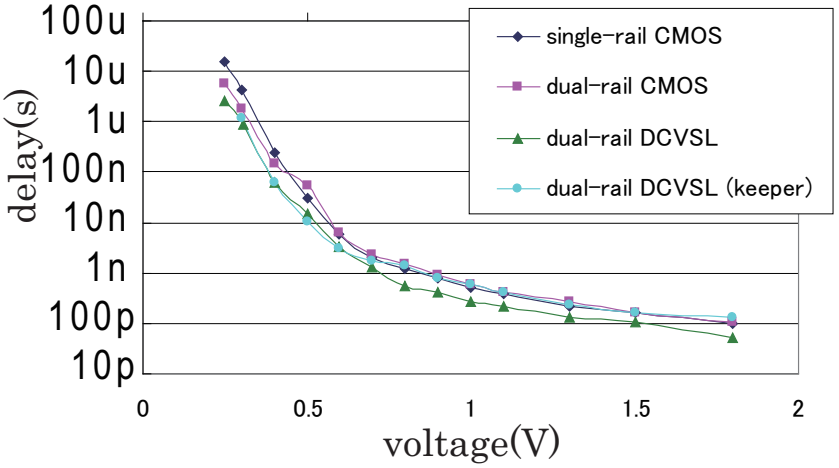


図 4.6 チップの測定結果.

おいて遅延時間の大きさは少し小さい値であったがオーダーは変わらない為、既存のバッファの測定結果のグラフでバッファの動作電圧を平滑化することで補正できることがわかった。補正後のグラフを図 4.7 に示す。低電圧下においてキーパーのない 2 線式 DCVSL 回路、2 線式 CMOS 回路、キーパーをつけた 2 線式 DCVSL 回路の順で遅延時間が大きくなっており、シミュレーションの結果とも合致していることがわかる。

4.3　遅延、電力、電力遅延積を指標として最適化した回路の測定結果

4.3.1　試作したチップ

0.18umCMOS プロセスで作成したチップの写真を図 4.8 に示す。それぞれの指標において最適化を行った回路が搭載されており、各々の回路の VDDMEAS を別にとり、1 段目と 100 段目の出力がそれぞれの回路にある構成となっている。

試作したチップにおいて、1 段目の出力と 100 段目の出力の差を 99 段分で割った値を 1 段分の平均遅延とした。また VDDM を流れる電流値を 100 段分で割った値に動作電圧をかけた値を 1 段分の回路に要する電力とした。

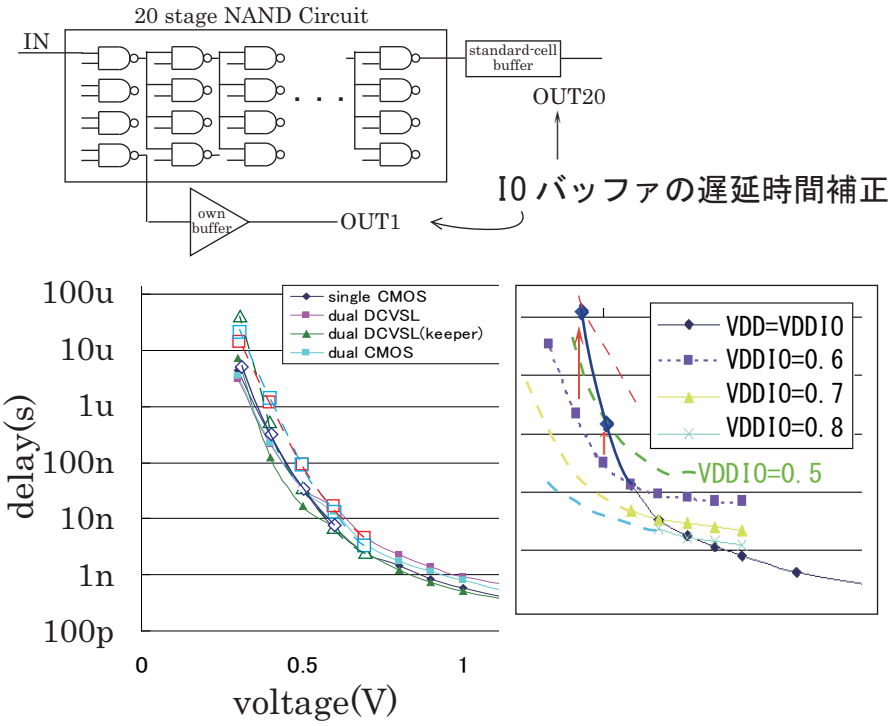


図 4.7 補正後の測定結果.

4.3.2　回路のばらつき

まず、回路のばらつきに関しての測定を行った。3 チップ間のばらつきに関しての測定結果を図 4.9 に示す。動作電圧 1.8V の点においては、300p〜380p での遅延ばらつきが見受けられた。しかし今回の測定はオシロスコープで行っているため、動作電圧 1.8V において 100 段分の遅延は数十 ns 程度であり、数 ns 程度の誤差がある可能性があることを考えると、ばらつきの大きさはもう少し小さい値になると考えられる。動作電圧 0.3V においては、605ns〜466ns のばらつきが見受けられた。この結果より、電圧を低くするにつれて、チップ間におけるばらつきの影響が大きくなるなることが実測からも示された。

4.3.3　回路の温度特性

作成した回路の温度特性について測定した。今回作成した回路はサイズが異なるだけであり、トランジスタの特性の変化はないはずであるので、1 つの回路において測定を行った。また、その温度依存性に関しての妥当性を理論解析によっても検討を行った。

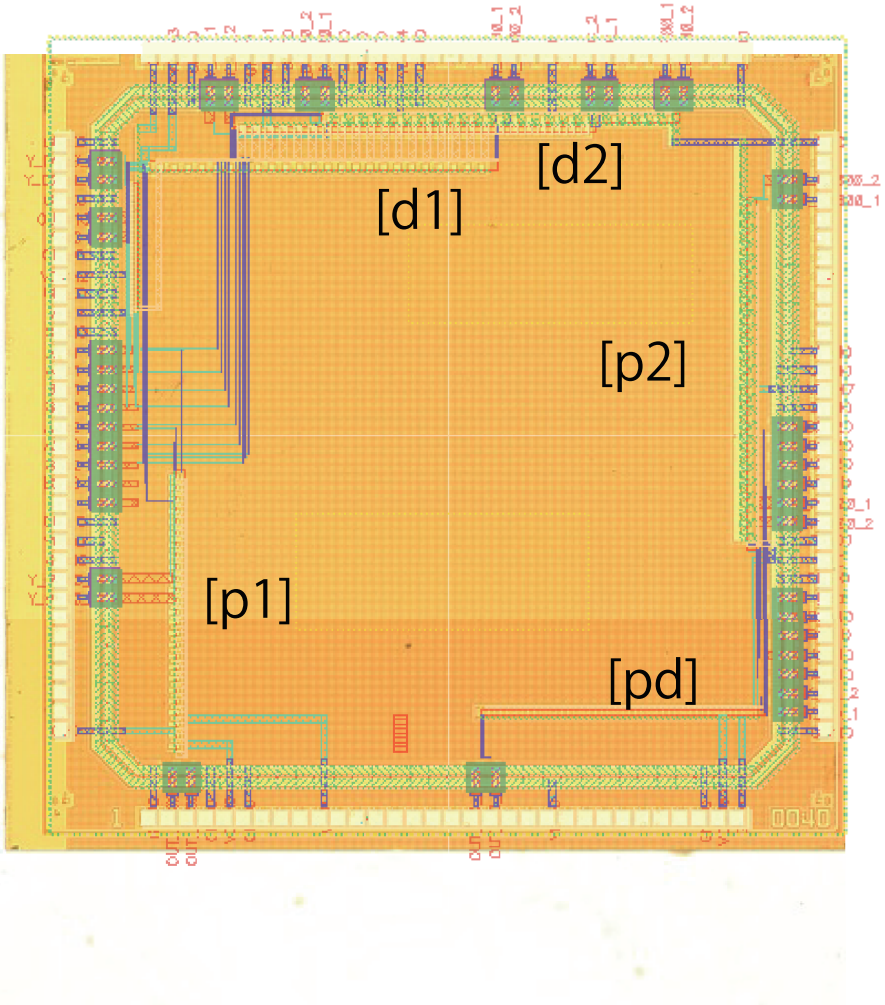


図 4.8 各指標において最適化した回路のチップ写真

計算に関しては、次に示す式を用いている [14]。

まず、遅延の温度特性を図 4.10 に示した。図より、高い電圧では温度が低い方が速く、逆に高い電圧においては温度が高い方が速くなっていた。また、回路の動作限界は、演算による遅延とダイナミックの欠点であるリークによるディスチャージで出力が変化することにより、データの遷移が同じとなるエラー状態になってしまう点を限界とした。そのため、温度が低い方が低電圧まで動作しているのは、温度が低い方が電子の移動が遅くリークによる出力の立ち上がり時間も遅くなったためと考えられる。遅延は式 (a) で表されるように、移動度と、ゲート電圧からしきい値電圧を引いた値に反比例する。動作電圧の高い点においては、ゲート電圧に比べてしきい値の変動は小さい

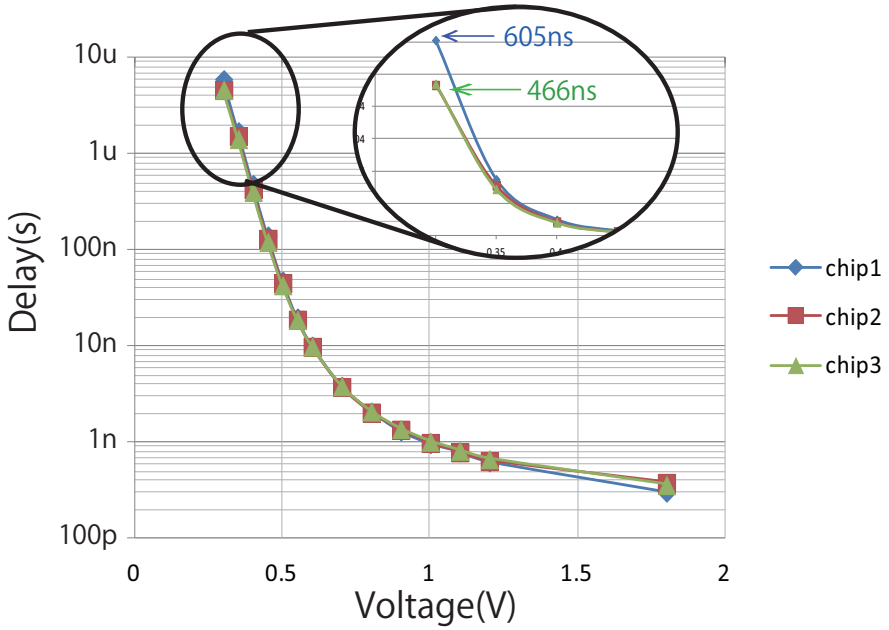


図 4.9 動作電圧 1.8V で遅延最適化した回路の遅延ばらつき

$$\tau \propto \frac{1}{\mu(V_G - V_T)} \quad \cdots (a)$$

$$I_D = \frac{W}{L}\mu\, Cox[(V_G - V_T)V_D - \frac{1}{2}V_D^2] \quad (b)$$

$$I_{leak} \propto W \exp(\frac{-V_T}{nU_T}) \quad \cdots (c)$$

$$\text{ただし} \quad V_{th} = V_{th0} - \kappa T \qquad \mu = \mu_0 \left(\frac{T_0}{T}\right)^m$$

ため、移動度の傾向に依存する。移動度は $u = u_0(\frac{T_0}{T})^m$ で表されるため、温度が大きくなるにつれて移動度は小さくなることが予測される（ただし、m は正の値とする）。移動度が小さくなると、遅延は移動度に反比例するので、温度が高くなるにつれて遅延は増加すると考えられる。図 4.11 の (a) に動作電圧 1.8V における、温度を変化させた時の遅延変化を示している。図からも同様の結果であることがわかる。また、図より式における定数を求めると、$m = 0.497$ であることがわかった。次に、動作電圧が小さい点においては、逆にしきい値の変動が大きく依存する。しきい値は $V_{th} = V_{th0} - \kappa\, T$ という式からも分かるように、温度の上昇に伴い小さくなる（ただし κ は正の値とす

る)。遅延は、ゲート電圧にしきい値を引いた値に反比例するので、温度の上昇につれて低い電圧では小さくなることが予想される。図 4.10 においても動作電圧が約 1V より小さい電圧においてその傾向が見られる。図 4.11 の (b) に動作電圧が低い 0.4V において遅延の温度依存グラフを示した。図からも同様に温度の上昇に伴って遅延が減少する傾向が見られる。動作電圧においても同様に比例定数を求めると、$\kappa = 2.53 \times 10^{-4}$ であることがわかった。

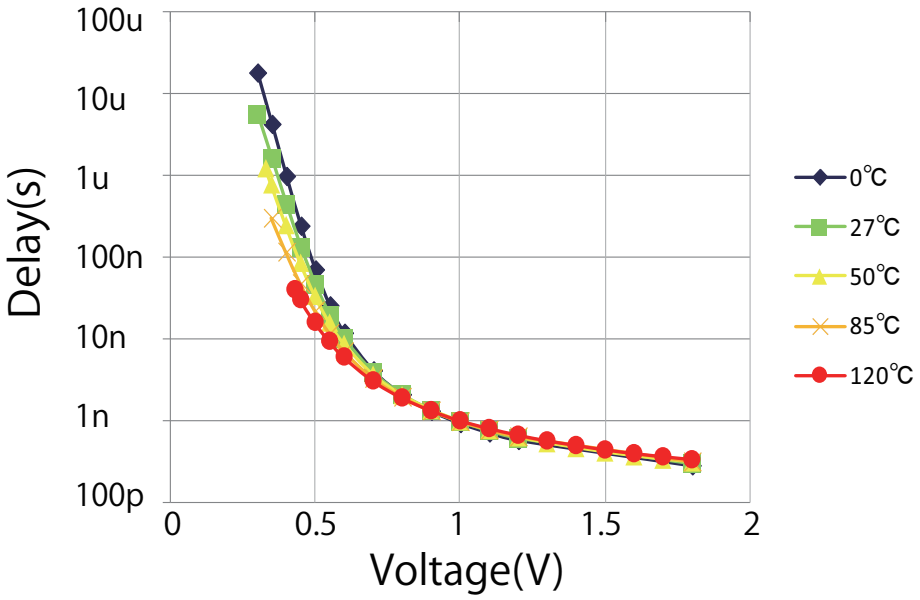
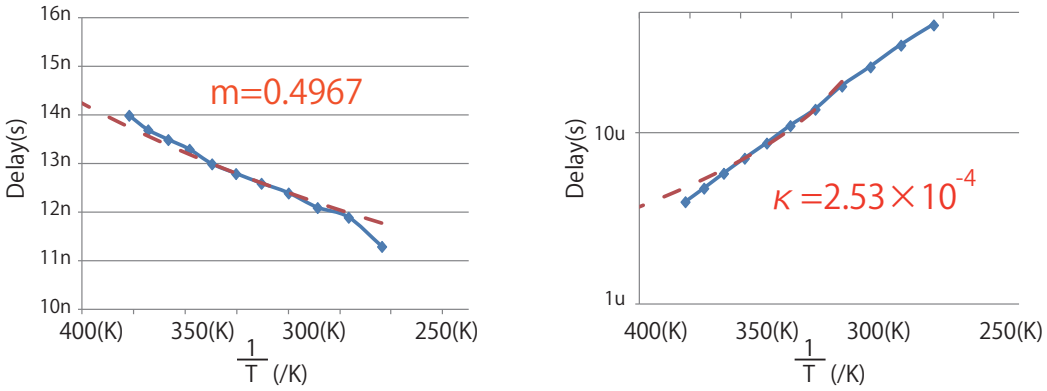


図 4.10 動作電圧 1.8V で遅延最適化した回路の遅延の温度特性

次に、電力値における温度依存性も測定した。温度を変化させた場合の各動作電圧での電力のグラフを図 4.12 に示す。図からもわかるように、遅延と同様に動作電圧の高い点と低い点において、温度による電力値の傾向が逆転する。動作電圧の高い点においては温度が高い程電力は小さくなっている。逆に動作電圧の低い点においては温度が低いほど電力は小さくなっている。電力に関しても上記に示した式において傾向の解析を行った。

まず、動作電圧が高い点においては、遅延のときと同様に移動度の温度依存の方が大きく見えると考えられる。式 (b) より電流値は移動度に比例するので、温度が高くなるにつれ移動度は小さくなり結果として電力も小さくなる方向にシフトすると考えられる。逆に、動作電圧の低い点においては、しきい値の温度変化の方により近くなると思



(a) 動作電圧 1.8V における遅延の温度依存性　(b) 動作電圧 0.4V における遅延の温度依存性

図 4.11 動作電圧 1.8V と 0.4V における遅延の温度依存性

われる。式 (b) より電流値はゲート電圧にしきい値をひいた値に比例しているので、温度の上昇と伴いしきい値は小さくなり結果的に電力値は大きくなると考えられる。また、動作電圧がしきい値以下の場合は電流の式が変わり、リーク電流での動作になるため、式 (c) で与えられるような値をとる。しかしながら、この式においてもしきい値が小さくなるにつれて電流値は大きくなるので、しきい値以下においても同様の傾向を示すことがわかった。

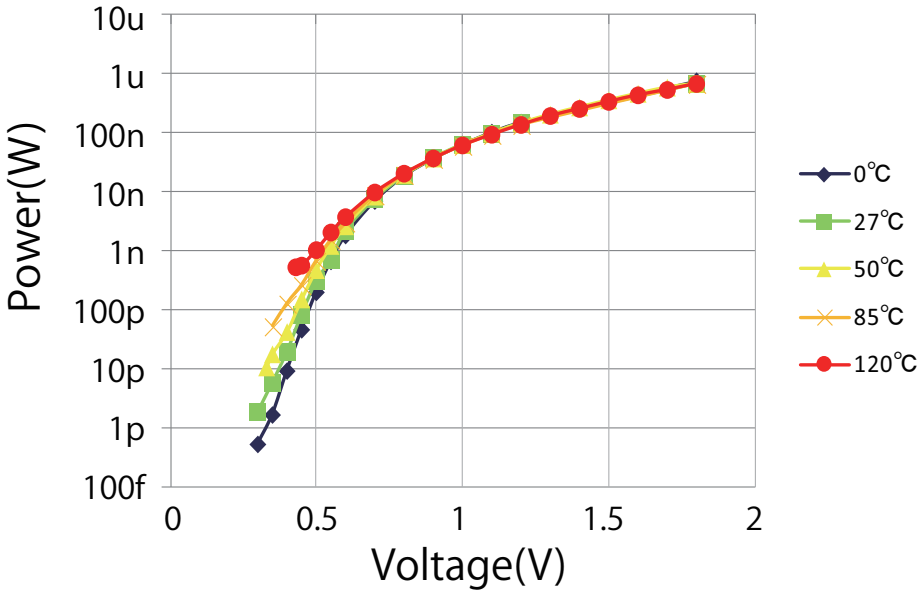


図 4.12 動作電圧 1.8V で遅延最適化した回路の電力の温度特性

最後に、電力遅延積に関しても、同様に温度を変化させた場合の実測値を電圧ー遅延グラフを図 4.13 に示す。電力遅延積に関しては、上記の遅延および電力値をかけたものになっている。ここで、遅延と電力の温度による傾向は、両者は共にトレードオフの関係になっている。そのため、それぞれが打ち消し合うことによって、ある電圧における電力遅延積は温度に依らず一定となっていることがわかる。また、動作限界の点においては、温度が低い方が動作を行っており、電力遅延積の極小値においても温度が低い点がより低電圧においてより小さな電力遅延積の動作を行っていることがわかった。

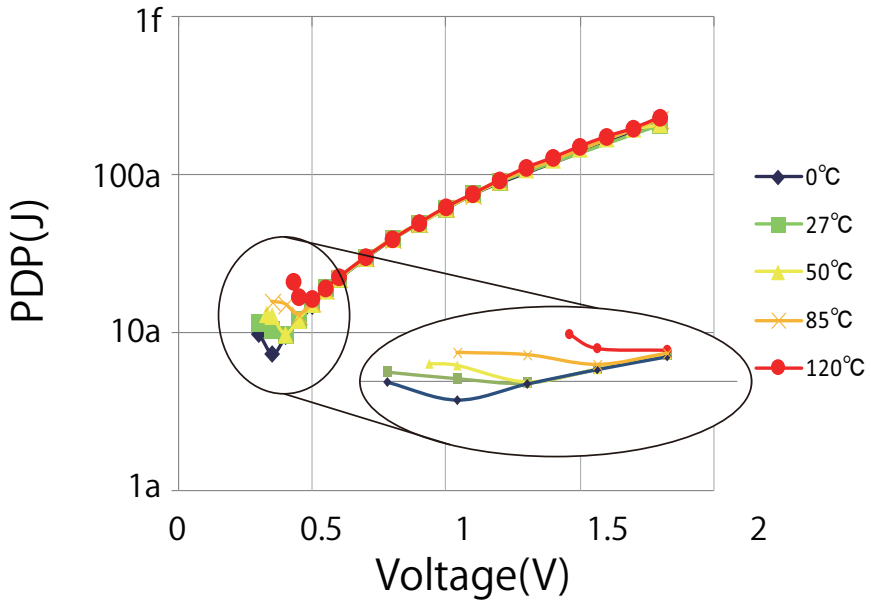


図 4.13 動作電圧 1.8V で遅延最適化した回路の電力遅延積の温度特性

また、回路を 225 度で 10 分程度温めた際、まわりのソケットやコンデンサは熱で溶けてしまったが、回路自体は動作したので測定を行った。その結果を図 4.14 に示す。図より、加熱前後の変化は温度が 0 度と 25 度の遅延の差に比べても小さい点から、加熱前後で遅延の変化がほぼなかったことがわかる。この結果から、チップ自体は 225 度の環境下におかれていても回路が壊れることはなく、温度耐性があることがわかった。

4.3.4　各指標における最適化の優位性

各指標において、最適化を行った各回路の測定結果を以下に示す。

まず、遅延において最適化した各回路の測定結果の比較を図 4.15 に示す。動作電圧

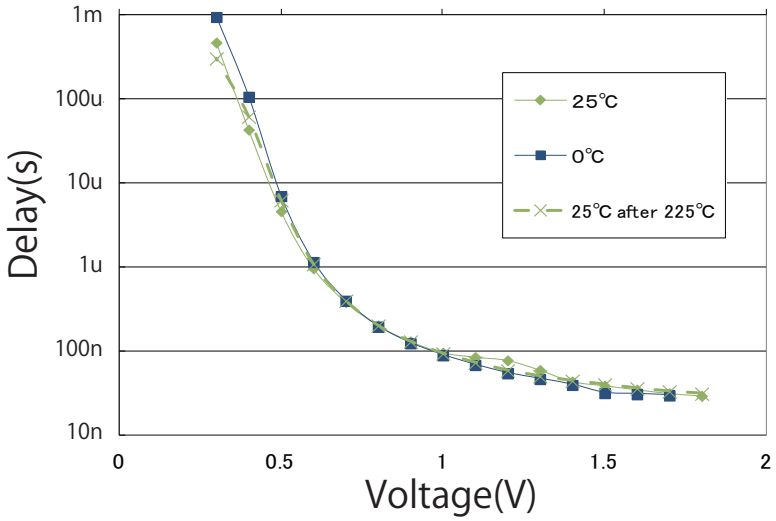


図 4.14 225 度で 10 分程度熱した前後の電圧遅延測定

の高い点においては、動作電圧 1.8V において最適化をした回路が最速であった。また、動作電圧の低い点においても動作電圧 1.8V で最適化した回路の方が速い結果ではあったが、動作電圧 0.2V において最適化した回路との差が小さくなってきていることから、動作電圧がもっと低い点において逆転する可能性があることが伺える。遅延で最適化した回路は、動作電圧 0.35V において、1.8V で電力最適値の遅延が 1.84us であるのに対し、同じ電圧で遅延最適値の遅延は 657ns となっており、最適化を行うことで実測においても約 3 倍もの速さでの動作が可能であることがわかった。動作電圧によらず遅延によって最適化をした回路はその他の指標で最適化した回路に比べて共に速いため、最適化による優位性は示された。

今回作成した NAND 回路の動作限界は、2 線のうち 0 であるべき値がリークにより立ち上がった点とした。そのため、pMOS 側でも演算を行いリークによる出力の立ち上がりが起こらない 2 線 CMOS 型 NAND 回路の方が低い電圧まで動作する結果となった。しかしながら、動作電圧 0.3V において遅延は約 4 倍になるというデメリットもあることがわかった。

次に、電力値において各回路を比較した測定結果を図 4.16 に示す。動作電圧によらず最も電力値が小さいものは 2 線 CMOS 回路であった。これは、pMOS 側でも演算を行っているため、リーク電流が少なくなることに起因していると考えられる。最適化

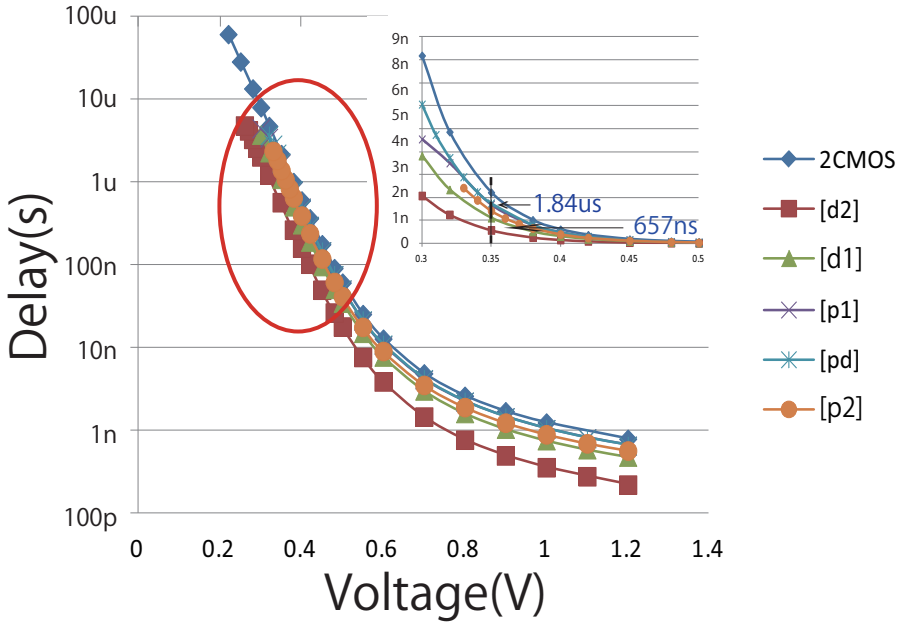


図 4.15 最適化した NAND 回路測定結果の遅延比較

した 2 線 DCVSL 回路の中で最も電力が小さかったのは電力遅延積で最適化した回路であった。電力で最適化を行った回路は動作電圧の高い点においては動作電圧 1.8V で最適化をした回路が電力値が小さく、動作電圧が低い点においてはほぼ同じ値となっている。これは、遅延と同様に動作電圧がもう少し低い点においてこの 2 回路の値が逆転することが予想される。動作電圧 0.35V において、動作電圧 1.8V での遅延最適値での電力が 27.4pW であるのに対し、同電圧において電力最適化を行った回路は 8.58pW となっており、最適化を行うことで実測において消費電力を 1/3 に削減できていた。この結果より、遅延による最適化と電力による最適化では各指標においてそれぞれの優位性があることが示された。

最後に、電力遅延積において各回路を比較した測定結果を図 4.17 に示す。電力遅延積においては、電力遅延積を用いて最適化した回路が電圧に依らず小さい値をとっていた。また動作電圧の低い部分での拡大図から、電力遅延積が極小をとる値も最適化した回路であり、極小値は動作電圧が 0.32V において 11aJ であることがわかった。この結果より、電力遅延積の値に関しては、ある動作電圧において最適値が最も小さなエネルギーであるだけでなく、動作電圧を変えた際の極小の値をとる回路も最適化をした回路

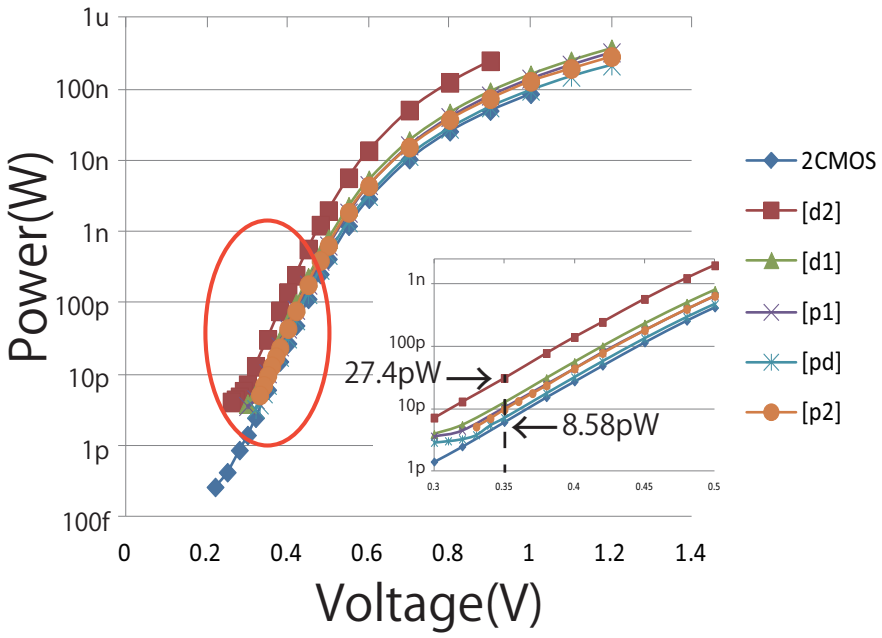


図 4.16 最適化した NAND 回路測定

であることがわかった。

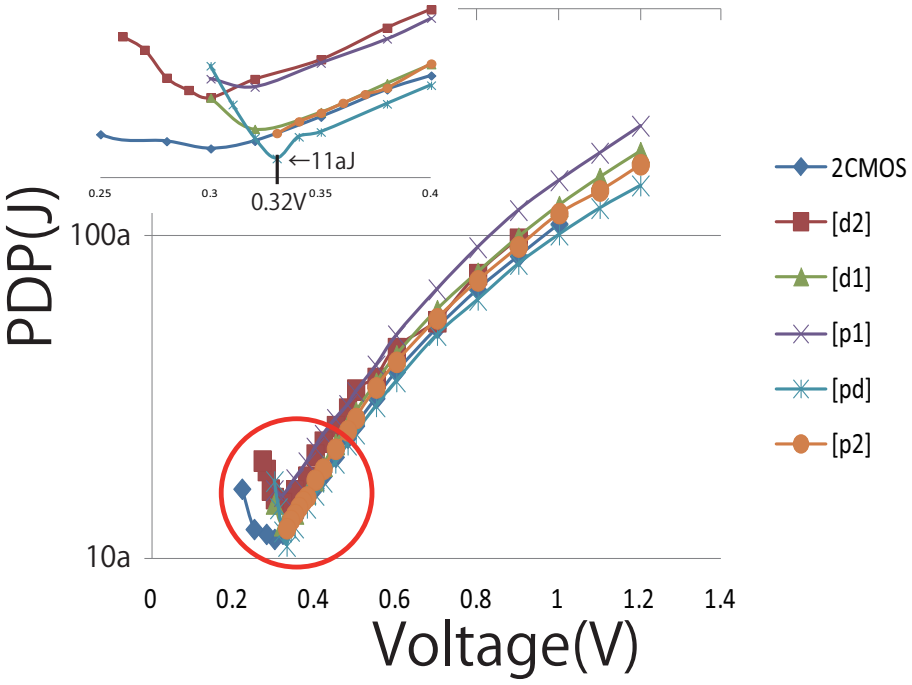


図 4.17 最適化した NAND 回路測定結果の電力遅延積比較

2 線 CMOS 回路の極小値は動作電圧のより低い点に存在し、その値は 2 線 DCVSL 回路のそれより小さい。そのため、一概に 2 線 DCVSL 回路の方が優れているとは言えない。ところで、2 線 DCVSL 回路の動作限界はリークによる出力の立ち上がりにより決まっていた。その為、温度を下げることでリークによる出力の立ち上がりが遅くなり、結果低い電圧まで動作していた。それに対し、2 線 CMOS 回路は出力が立ち上がらない点が動作限界となっていた。つまり、温度を変化させたとしても動作限界に変化はないと考えられる。そこで、2 線 DCVSL 回路の動作限界を低くするために、温度を 0 度の点において回路の比較を行った。その結果を図 4.18 に示す。図 4.18 より、2 線 DCVSL 回路は温度を低くすることで PDP に関しても下側にシフトするのに対し、2 線 CMOS 回路の極小値に変化はなかった。以上の結果より、2 線 DCVSL 回路は温度を下げることで 2 線 CMOS と電力遅延積において同等の極小値をとることができ、極小値をとる遅延の値も 2 線 DCVSL 回路の方が速いため、2 線 DCVSL 回路の方が優位であることが示された。

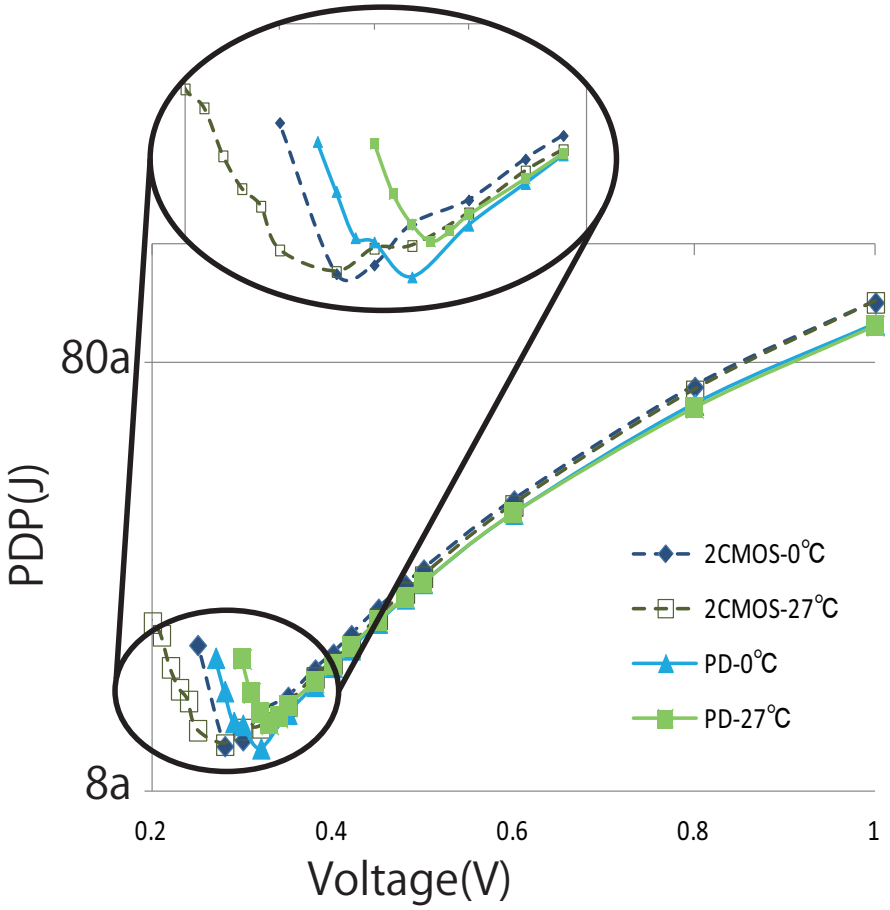


図 4.18 最適化した NAND 回路測定結果の電力遅延積比較

第5章

最適化手法の8ビット加算器への応用

5.1　8ビット加算器

汎用性のある一般的な回路の一つに、加算器がある。加算器には、大きく分けて2種類の演算方法がある。通常の演算のように下位ビットから順に演算を行うプルキャリー型と、桁上げを予測して演算を行うことで高速化を図ったキャリールックアヘッド型である。以下に、この2種類の加算器について説明をする。

5.1.1　リプルキャリー型加算器

リプルキャリー型加算器は、和$SUM_{i+1} = A_i \oplus B_i \oplus Ci$、および繰り上げ$CARRY_{i+1} = A_iB_i + A_iC_i + B_iC_i$でそれぞれ表される。Ciは$(i-1)$段からの繰り上がりである。前の段の繰り上げが決定されてから次の段の演算が行われ、図5.1のようなシンプルな構成になっているため、段数に比例した時間を演算に要する。自己同期回路においては平均遅延で動作するため、同じ桁においてのAiとBiの値が等しい場合は前の段を待たずに次の段へ桁上げができるため、最速([h]00000000[l]+[h]00000000[l]、[h]11111111[l]+[h]11111111[l]など)はCARRY1段分とSUM1段分の和と程度の遅延で演算を終了することができる。自己同期回路は演算の終了を検出して動作できるため、速く演算を終了できることもメリットとなる。逆に同期回路においては、最悪遅延を見なければならないため、段数に応じて遅くなる。

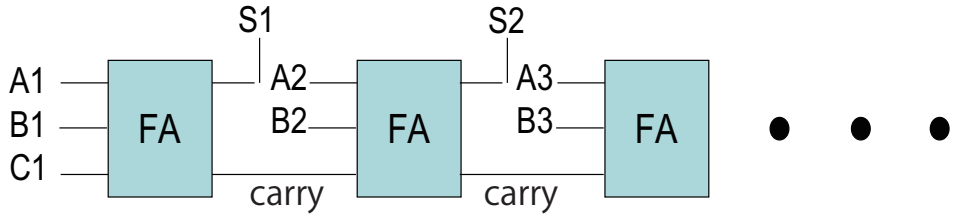


図 5.1 リプルキャリー型加算器

5.1.2 キャリールックアヘッド型加算器

キャリールックアヘッド型 (桁上げ先読み) 加算器は、桁上げの部分を別個に計算させることで、演算時間を短くする加算器である。加算器の構成は図 5.2 のようになっており、演算時間が段数の log に比例する加算器になっている。そのため、段数が多くなるほど、より高速な演算が可能な回路の実現が可能になる。しかしながら、桁上げのみを演算する部分を必要とするため、回路全体が大きくなるという点とそれにより消費電力が大きくなるというデメリットがある。また、演算において桁上げの部分の演算を行って出力が出ているため、最速の演算であっても 8 ビット回路であれば約 4 段分の遅延を必要とするため、終了を検出して動作する自己同期回路において最速値はリプルキャリーの方が速いといえる。

同期回路では、キャリールックアヘッド型が一般的な加算器としてはよく用いられる。それに対し、自己同期回路は最悪遅延で動くわけではなく演算が終了すれば次に進むことができるため、一概にどちらが良いとは言えない。そこで、自己同期的に動作させることができる 2 線 DCVSL 回路に関してはリプルキャリー型とキャリールックアヘッド型の 2 種類について作成することとした。また、4 ビット加算器では、くりあげが少ないため、キャリールックアヘッド型加算器において優位性があまりないため、差がより出るように 8 ビットの加算器において作成を行った。

5.2 加算器を構成する回路の最適化

加算器の作成にあたって、それぞれの加算器を構成する一番小さい回路に関してそれぞれ FO4 の回路を作成することで最適化を行った。

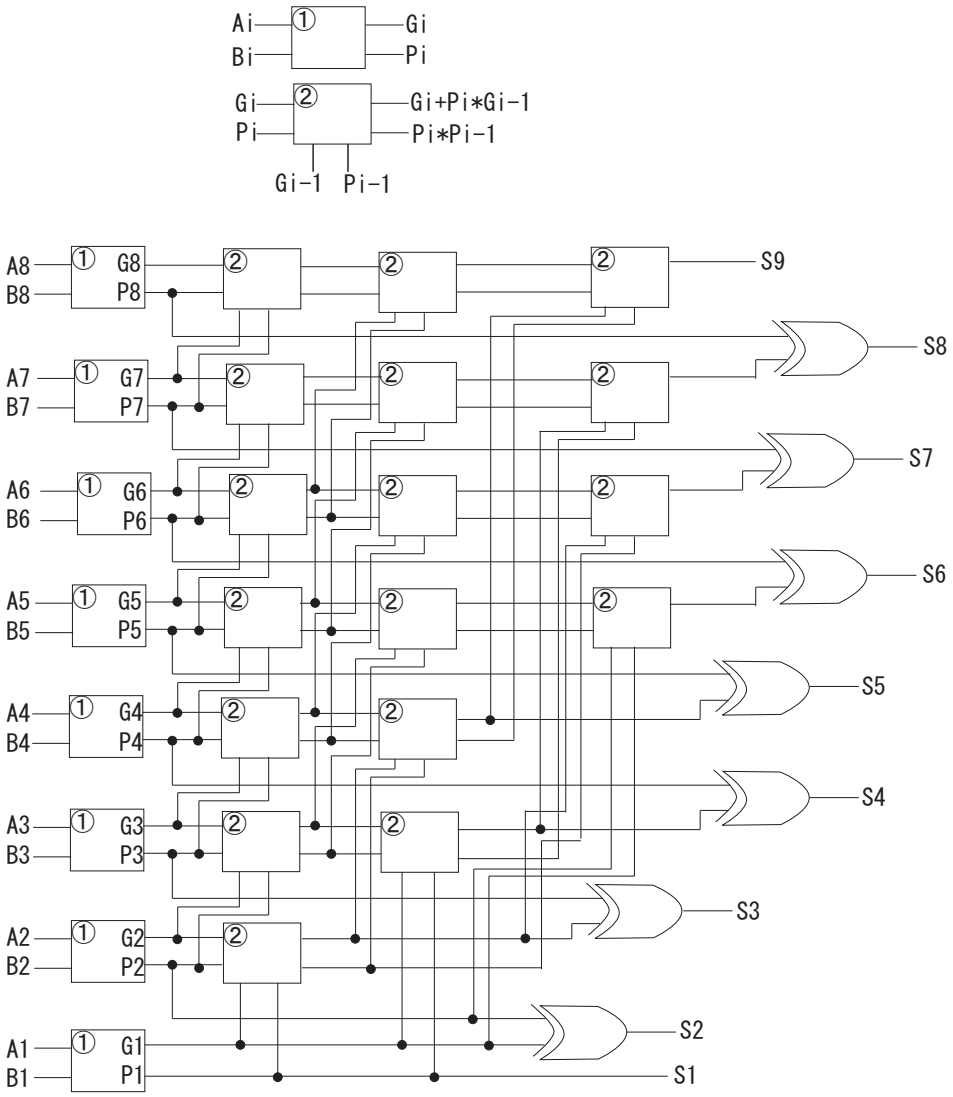


図 5.2 キャリールックアヘッド型加算器

5.2.1　リプルキャリー型加算器の最適化

リプルキャリーは、SUM と CARRY の回路のみでできている。SUM は、加算する 2 つの入力および下の位からの桁上げの 3 値の XOR により得られる。そこで、まず図 5.3 の (a) に示すような 3 入力 XOR 回路を用いて SUM とした。同様に、桁上げの値である CARRY に関しても 3 入力の演算回路を図 5.3 の (b) に示す回路を用いることにした。この両回路に関して、それぞれ動作電圧 0.2V において最適化を行った。

また、この回路に関して最速遅延、平均遅延、最悪遅延の比較のグラフを図 5.4 に示す。平均は、全通りの遅延をシミュレーションより測定し、その平均遅延に近い値を

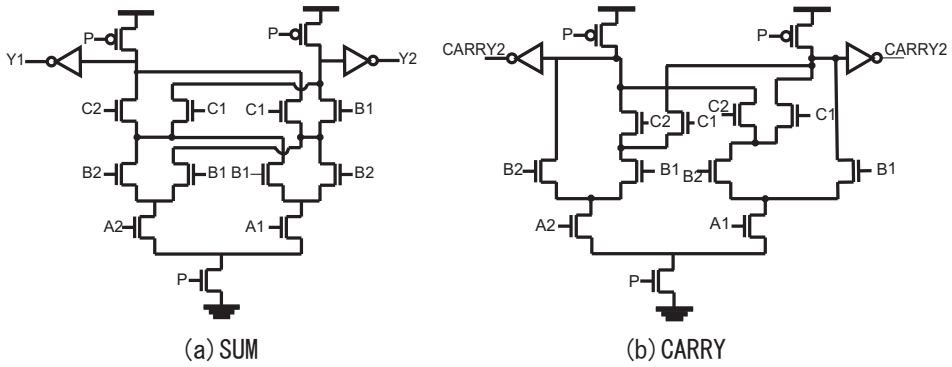


図 5.3 リプルキャリー型加算器を構成する縦積み 3 段の (a)SUM(b)CARRY

持つ演算の遅延を用いている。前節のリプルキャリーでも述べた通り、遅延は前の段における A と B の値が同じであれば速く、異なると遅い。そのため、8 ビット全てが同じ [h]11111111[l]+[h]11111111[l] や [h]00000000[l]+[h]00000000[l] は最も速く、逆に 8 ビット全ての値が異なる [h]11111111[l]+[h]00000000[l] の様な演算は遅くなっていることがシミュレーションの結果からもわかる。

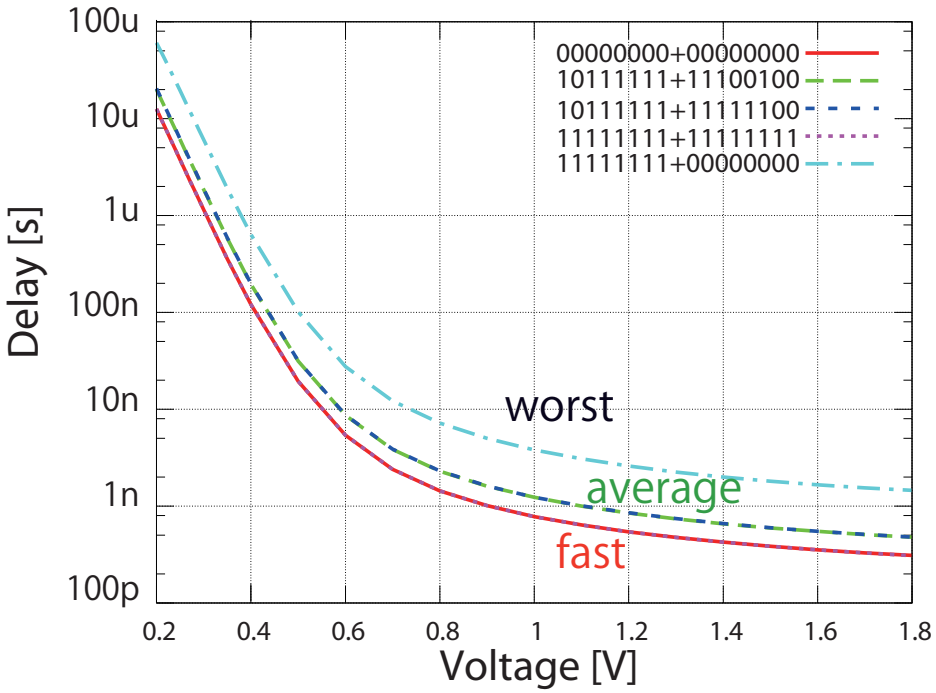


図 5.4 縦積み 3 段のリプルキャリー加算器の遅延のシミュレーション結果

次に、先に示した回路は縦積み 3 段の回路であるため、動作電圧の限界という点では

2 段に劣ると考えた。そこで、同様の演算回路を縦積み 2 段で構成した回路図を図 5.5 にそれぞれ示す。(a) に示す演算は 2 値の XOR を示すものであり、3 桁の XOR は次段にこの結果と下の段からの CARRY をたす演算が入る。ここで、前段の桁上げ部分の演算を行っている間に、A と B の XOR の演算を行っていると考えられるため、演算時間の差はあまり生じないと考えられる。この回路についても比較を行うため、それぞれの回路について最適化を行った。最適化の際、トランジスタの分類は縦積みの段が並列のもののみ同じサイズとして、それぞれ Wn を 4 種類に分けて動作電圧 0.2V において最適化を行った。

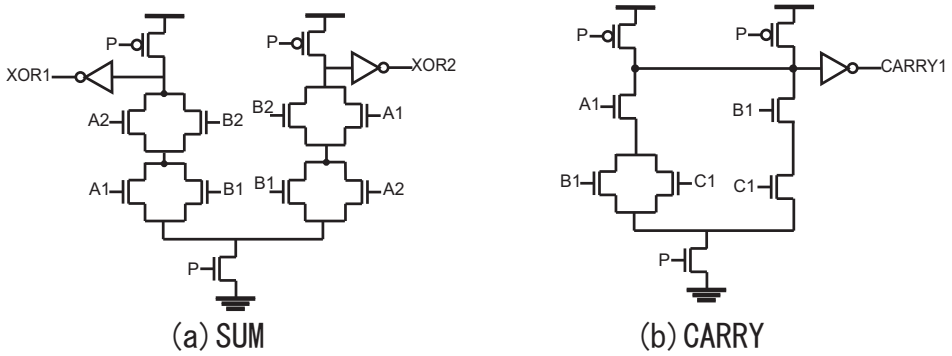


図 5.5 リプルキャリー型加算器を構成する縦積み 2 段の (a)SUM(b)CARRY

5.2.2　キャリールックアヘッド回路の最適化

キャリールックアヘッド回路に関しても、同様に個々の単位に分類して最適化を行った。キャリールックアヘッドは、AND-NOR 複合ゲート、OR-NAND 複合ゲート、XOR、NAND、インバータの 5 つを組み合わせて構成されていることがわかった。そこで、各々の回路に分けて動作電圧 0.2V において遅延および電力遅延積に関して最適化を行った。また、2 線式回路においてのインバータは出力の 2 線を入れ替えることで逆の値を得るため、最適化はインバータ以外の 4 回路に関して行った。

5.3　スタティック CMOS 回路との比較

スタティック CMOS に関しては最悪遅延で動作するため、キャリールックアヘッド型加算器にメリットがある。そこで、スタティックの 1 線 CMOS に関しても同様に 5 種

類の回路に分けてそれぞれにおいて動作電圧 0.2V で回路の最適化を行った。この際、1 線 CMOS 回路に関しては入力 A,B の立ち上がりから出力の立ち上がり/立ち下がりまでの時間を遅延とした。2 線 DCVSL 回路に関しては、

最適化を行った 1 線 CMOS 回路と、2 線 DCVSL 回路の 3 種類の加算器に関してのシミュレーション結果を図 5.6 に示す。図からわかるように、1 線 CMOS 回路は最悪遅延によって動作するので、2 線 DCVSL 型回路は自己同期システムにすることで平均遅延で動作させると仮定すると 1 線 CMOS の加算器の方が 2 線 DCVSL で作成した加算器に比べて動作速度が遅いという結果になっていた。

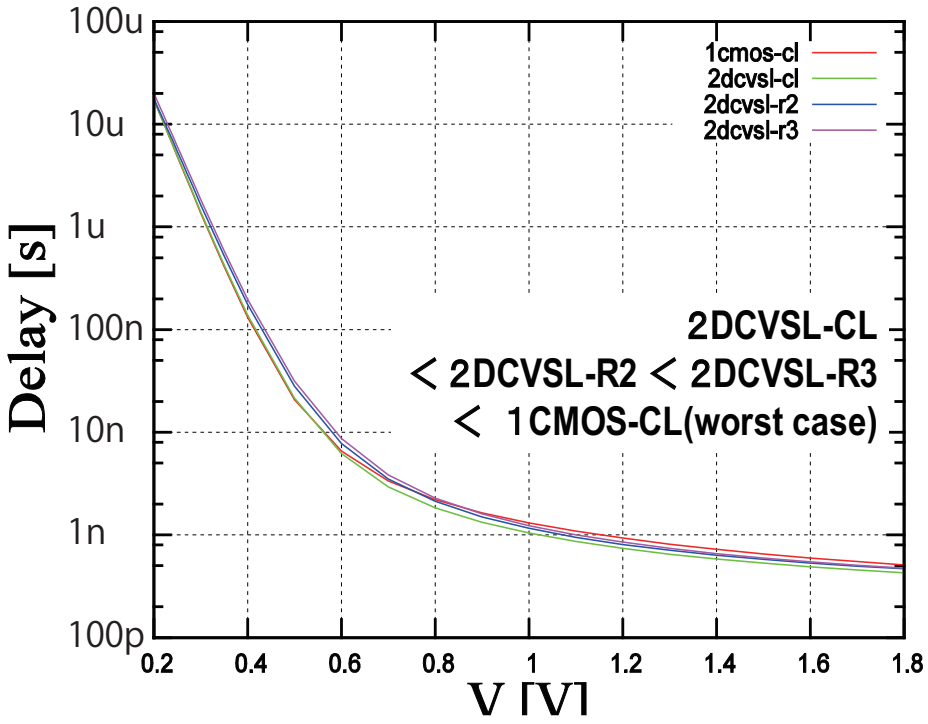


図 5.6 4 種類の加算器の測定結果

5.4　最適化した回路の実現とその測定結果

5.4.1　加算器搭載チップの実現

実際にそれぞれの回路をレイアウトし、チップを作成した。作成したチップの回路図を図 5.7 に、チップ写真を図 5.8 にそれぞれ示す。1 線 CMOS および 2 線 DCVSL のキャリールックアヘッド型加算器、2 線 DCVSL のリプルキャリー型で縦積み 2 段と 3 段で

実現した加算器の 4 種類の回路を搭載している。入力には tree 型にしたバッファを出力には既存の IO バッファを用いて作成した。

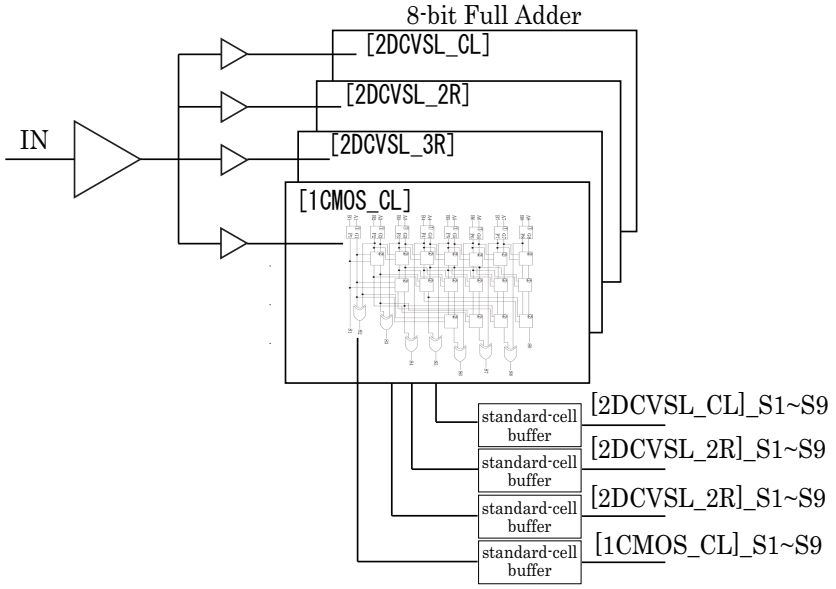


図 5.7 最適化した加算器のチップの回路図

5.4.2　作成したチップの測定方法

2 線 DCVSL 回路の遅延は、プリチャージが立ち上がってから演算が終了するまでの時間を遅延と定義した。しかしながら、作成したチップはプリチャージの入力をそのままバッファを通して出力するピンを作成していなかった。そこで、ポケットジェネレータからの出力と、入出力バッファと加算器を通ってきた出力の差をオシロスコープで測定し、以下の方法で計算した入出力バッファ分の遅延をひくことにより、シミュレーションと比較できるようにした。

バッファの測定は、2 段縦積みのリプルキャリー型加算器にて行った。リプルキャリー型は独立に SUM と CARRY の回路があるため、[h]11111111[l]+[h]11111111[l] の入力を入れたとき、1 ビット分の出力の遅延が CARRY 回路の遅延になる。また、最終段は td8 が SUM を通った出力、td9 が CARRY を通った出力であるので、そこから SUM の遅延が求まる。さらに、i 段目の出力の遅延は CARRY を i 個分と SUM および入出力バッファの遅延の和であるから、そこから逆算をすることで入出力バッファの遅延を算出した。バッファのばらつきがあり、CARRY が一意に決定できなかったため、それぞ

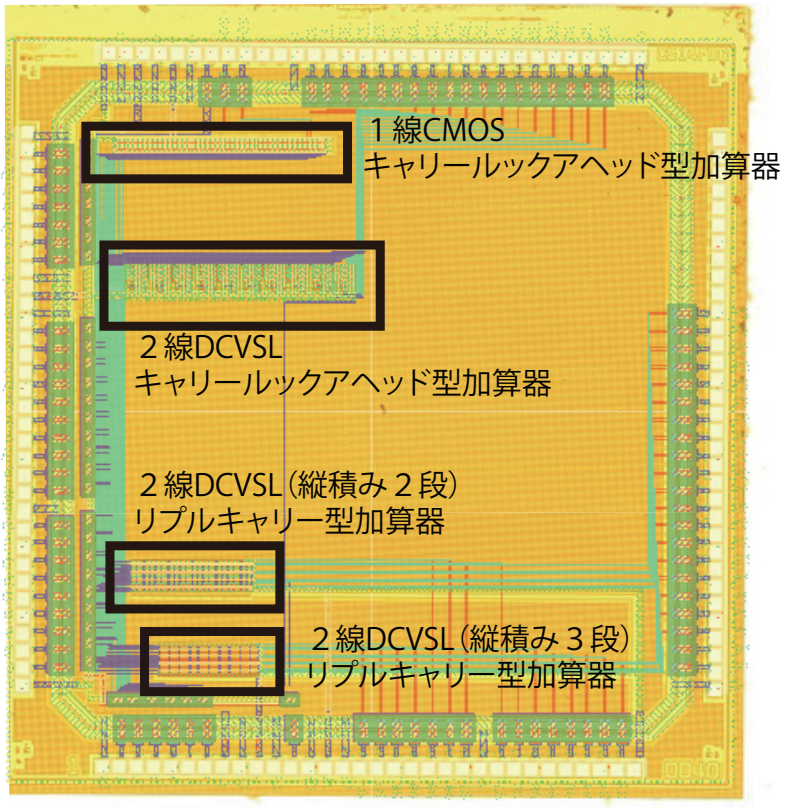


図 5.8 最適化した加算器のチップ写真

れの段においての遅延を測定し、最小二乗法を用いて平均の CARRY の値を求めた。また、SUM に関してはバッファのばらつきに比べて小さい値であることが予測されたので、今回のバッファ遅延においては微小とみなし無視した。

上記の方法で測定をした結果を図 5.10 に示す。薄い水色の一番遅延が小さいものは、スケマレベルでのシミュレーションにより求めた遅延である。それよりもう少し遅い紫色の線はレイアウトからのシミュレーションであり、レイアウトにより発生する抵抗値の影響によりスケマレベルでのシミュレーションに比べて遅延が大きくなっていた。濃い青色の線は実際のチップの測定結果である。レイアウトからのシミュレーション結果に対して動作電圧の高い点ではあまり差はないものの、動作電圧の低い点においては遅延の差が大きい。そこで、演算における最高動作周波数を求めることで、回路の演算にのみかかっている時間の測定を動作電圧が低くレイアウトからのシミュレーションと差が発生する点において行った。その結果が緑色の実線である。緑色の実線より、

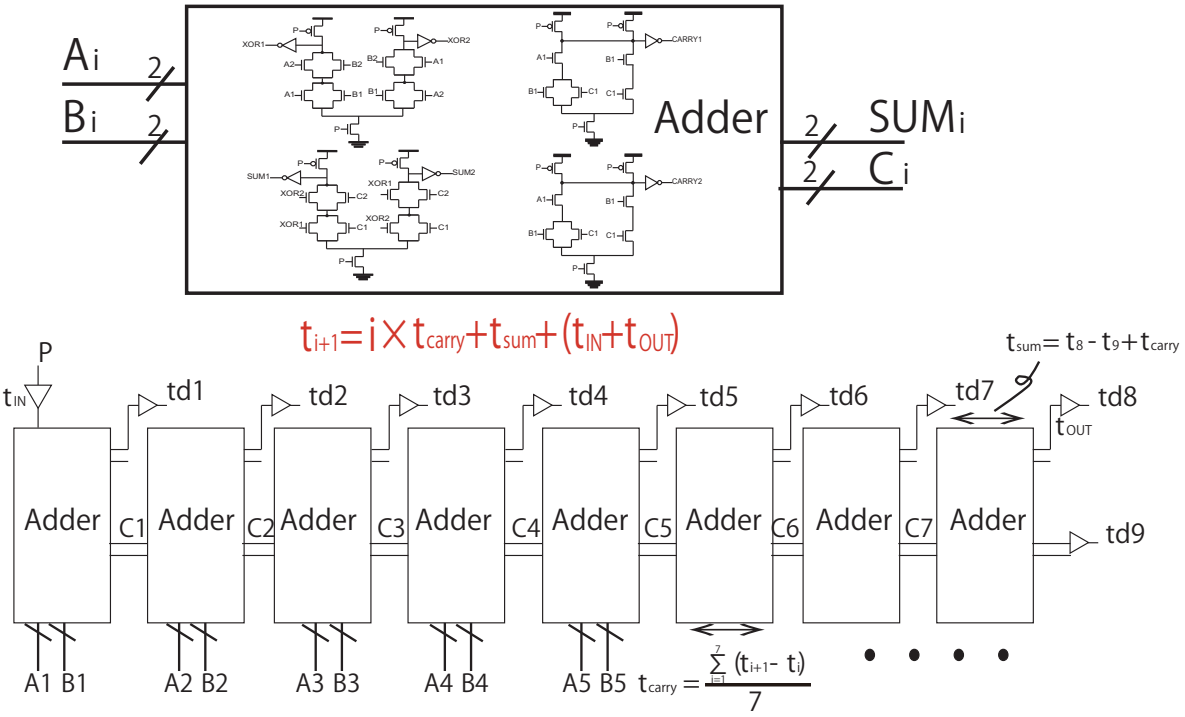


図 5.9 入出力バッファの遅延算出方法

入出力バッファの演算は妥当であることがわかった。以上より入出力バッファの遅延を省いた値の妥当性が示せた。以下の測定は、ここで求めたバッファの遅延をひくことで測定値を出している。

5.4.3 加算器の温度特性

縦積み 2 段のリプルキャリー型加算器において、遅延の温度依存性に関して測定を行った。結果を図 5.11 に示す。NAND を 100 段直列に並べた回路と同様に、動作電圧が 1.8V では温度上昇に伴い遅延も大きくなっている。それに対し、動作電圧が 0.4V においては温度上昇に伴い遅延も小さくなるという逆の傾向を示していた。

ここで、動作電圧 1.8V においては、NAND 回路に比べて直線的な傾向が見られなかった。しかし測定値が 10ns 以下であるため、バッファによる誤差やオシロスコープが 2 桁分しか見れなかったという精度による問題で誤差があると判断し、1 割のエラーバーをつけた。最小 2 乗法により引いた近似線はこのエラーバーの中に入っているため、赤い線で引いた実線は妥当であると考えた。この直線より、NAND 回路同様に移動度の

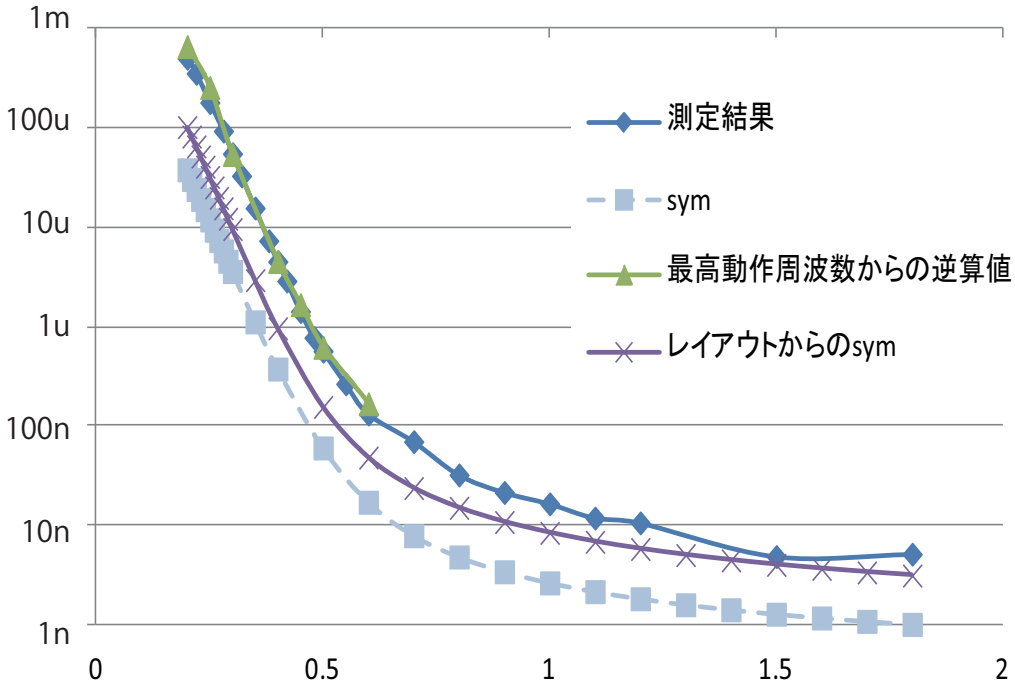
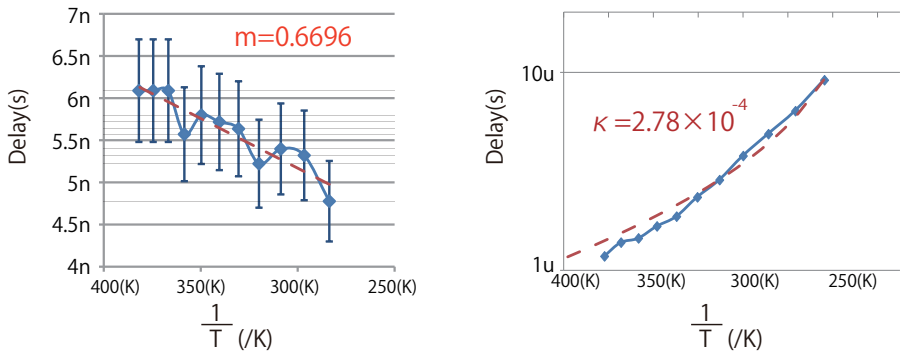


図 5.10 バッファ分を引いた測定結果とその妥当性



(a)1.8Vでの温度と遅延のグラフ　(b)0.2Vでの温度と遅延のグラフ

図 5.11 リプルキャリー型加算器の動作電圧 1.8V と 0.4V における遅延の温度依存性

温度依存係数を求めた。その計算値は m=0.6696 であった。同様に、動作電圧 0.4V においてもしきい値の温度依存係数を求めると、$\kappa = 2.79 \times 10^{-4}$ であることがわかった。

5.4.4　最適化した加算器の比較

まず、3 段縦積みのリプルキャリー型加算器において、最速遅延、平均遅延、および最悪遅延の 3 種類の遅延について測定を行った。その結果を図 5.12 に示す。電圧の低い点においては 3 つの遅延の傾向が正確に見えているのに対し、高い電圧においてはガタガタの実測結果になっている。これは、動作電圧の高い点では遅延が小さく、バッファの遅延を計算した際のばらつきによる誤差によって生じている。そのため、誤差を考慮して黒い点線のように平均の値をとって引くことで、実際の回路遅延の傾向が見られている。この結果は、図 5.4 のシミュレーション結果の傾向とも合致しており、妥当であると考えた。

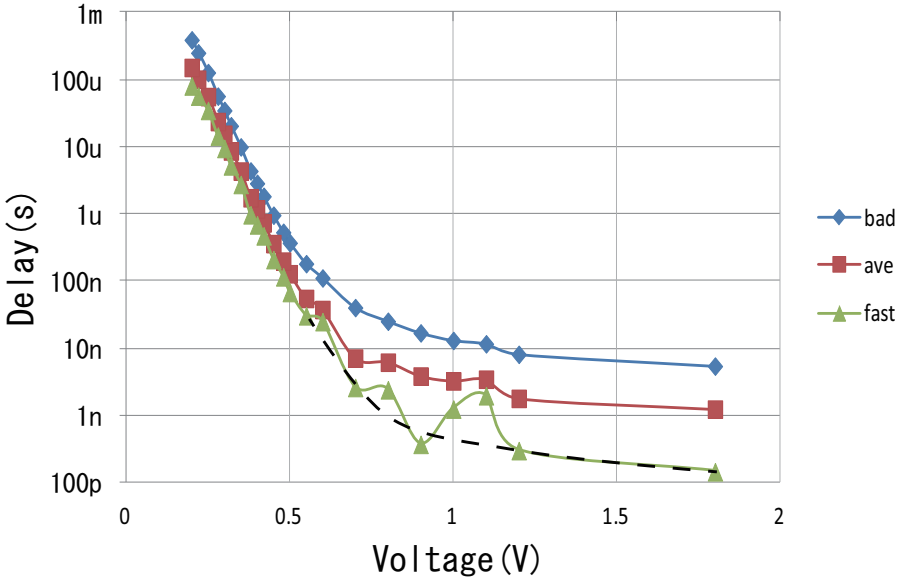


図 5.12 3 段縦積みリプルキャリー加算器の遅延の実測結果

作成した 4 種類の加算器において、遅延の実測値を比較した結果を図 5.13 に示した。1 線 CMOS 加算器においては、最悪遅延を考えたクロックで同期する回路になるため、最悪値のみを示した。2 線 DCVSL 型で示した加算器は自己同期的に動作させることができるため、平均遅延を採用しエラーバーとして最速の場合と最悪遅延について示した。回路のみで最悪遅延を比較すると、1 線 CMOS 回路が速く、2 線 DCVSL 型に関してはキャリールックアヘッド型の最悪遅延が速い結果になっている。最悪遅延の場合、リプルキャリーは下の段から桁上げを全て伝搬しなければならないため 8 段分の遅延

がかかってしまうのに対し、キャリールックアヘッド回路に関しては桁上げを先に演算しているので前の段の演算を待たない分最悪遅延は小さくなることが考えられる。そのため、キャリールックアヘッド型加算器に関しては最悪遅延と平均遅延の差があまりない。

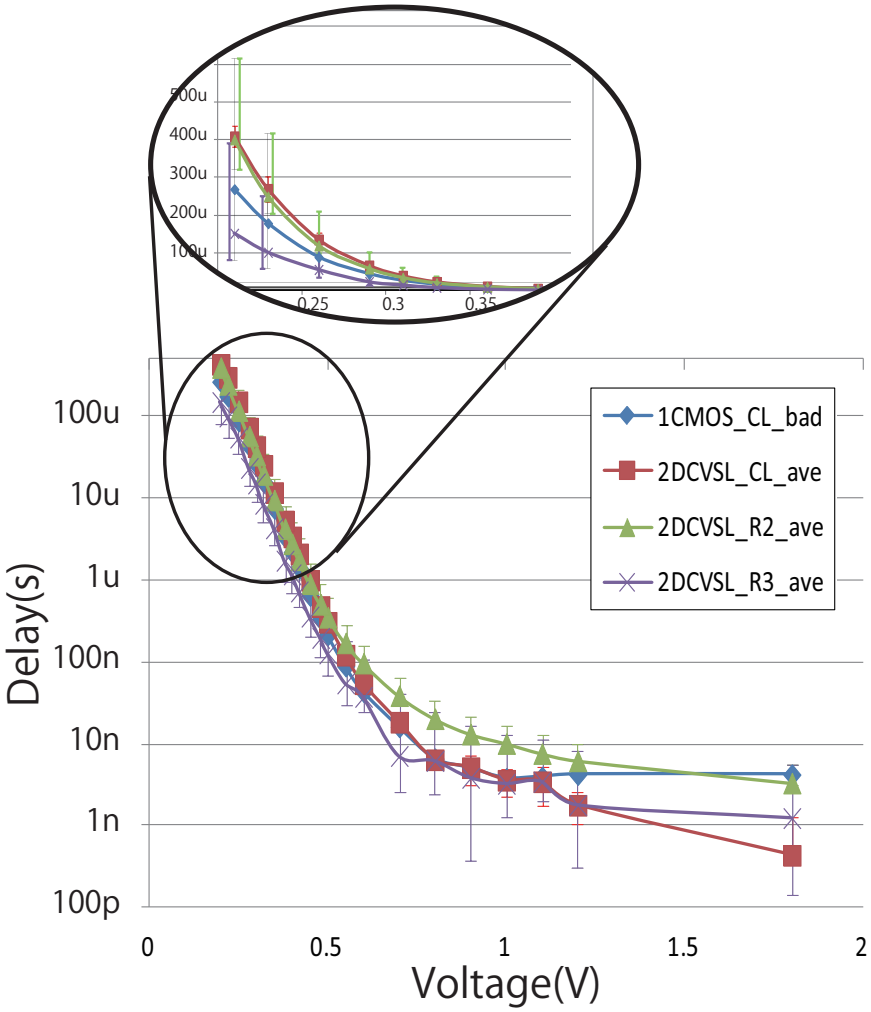


図 5.13 最適化した 4 種類の加算器の比較

リプルキャリー型加算器は、最も遅延が小さいものは前段の CARRY および SUM によって演算が終了するため、平均遅延で動作させることを想定すると優位性が出てくる。そこで、2 線 DCVSL 回路に関しては自己同期的に動作させることを想定し平均遅延にて比較をした。動作電圧の低い点においては縦積み 3 段のリプルキャリー型加算器の平均遅延が最も速く最適化を行った動作電圧 0.2V において縦積み 3 段のリプルキャ

リー型加算器は 151us であり、同電圧におけるキャリールックアヘッド型の平均遅延の 437us に比べ約 3 倍の速さであることがわかった。また、最適化を行った 1 線 CMOS の加算器の同電圧における最悪遅延は 269us であり、こちらと比べても 3 段縦積みのリプルキャリー型加算器は約 1.8 倍の速度で動作させることができることがわかった。動作電圧の高い点において異なる傾向が得られているのは、前章でも述べたように動作電圧によって遅延の最適値は異なるためである。

以上の結果より、2 線 DCVSL 回路で最適化を行った加算器を比較すると、動作電圧の低い点においてはリプルキャリー型の方が自己同期的に動作させることを考慮すると有利であることがわかる。また、キャリールックアヘッド型加算器は桁上げを行う回路が余計につくため消費電力も大きくなることを考慮すると、リプルキャリー型の方が電力の面でも優位であることが伺える。実際に 3 段縦積みリプルキャリー型加算器の電力遅延積の極小値は動作電圧 0.3V で 6.9[fJ] であるのに対し、リプルキャリー型加算器の極小値は動作電圧 0.3V で 34.7fJ と約 5 倍の大きさであることがわかる。さらに、極小値における遅延に関しては、3 段縦積みリプルキャリー型加算器が動作電圧 0.3V において 15.3[us] であったのに対し、リプルキャリー型加算器は動作電圧 0.22V において 303[us] と約 20 倍の速度であることもわかった。

以上の結果より、速度の面で考えると 1 線 CMOS 型加算器に比べて平均動作が可能な 2 線 DCVSL 回路の方が優位であることがわかった。また、2 線 DCVSL 型加算器の中では縦積み 3 段のリプルキャリー型加算器が最も高速に動作させることができ、電力遅延積の極小の値も最小であるため、2 線 DCVSL 型で構成する加算器はリプルキャリー型が良い回路であることがわかった。

第6章

結論

本論文では、2線式論理を用いて自己同期システムに応用できるダイナミック回路に関して最適化を行い、最適化に優位性をシミュレーションから示した。また0.18umCMOSプロセスを用いて実際に最適化を行った回路を用いて実測による最適化の評価を行った。

まず、ダイナミック型NAND回路において、動作限界電圧、遅延、電力、電力遅延積の4つの指標に関して最も小さい値をとるようなトランジスタサイズをとるように、FO4の回路を作成しいくつかの部分にトランジスタを分けてサイズを調節することで最適化を行った。まず動作限界電圧において最適化した回路はモンテカルロシミュレーションにより0.18Vまで動作する回路の作成に成功した。また、動作電圧0.2Vにおいて、最適化を行うことで遅延は5~10倍高速に、電力は4割程度の削減に、電力遅延積は8割も値を小さくなっており、シミュレーション上において最適化の優位性を示せた。

次に、0.18umCMOSプロセスにて作成したチップの実測においても優位性の評価を行った。その結果、遅延においては動作電圧0.35Vにおいて遅延で最適化を行った回路は電力で最適化を行った回路の3倍の速度であり、電力においては動作電圧0.35Vにおいて電力で最適化を行った回路は遅延で最適化を行った回路の1/3程度の電力であることがわかった。また、電力遅延積においても、それで最適化を行った回路が最も小さくなり、極小点は動作電圧0.32Vにおいて11aJであることがわかった。以上より、動作電圧の低い点でも、最適化を行うことの優位性を測定結果から示すことができた。

最後に、汎用性のある回路の一つとして8ビット加算器を取り上げ最適化を行い、最適化回路を試作後測定した。加算器はリプルキャリー型とキャリールックアヘッド型加算器の2種類を取り上げ最適化を行った。1線CMOS回路は同期回路であるため最悪

遅延で動作することを考慮すると、動作電圧 0.2V において最適化したキャリールックアヘッド回路における演算時間は 269us であった。それに対し、2 線 DCVSL 回路で構成した加算器は自己同期的な動作により平均遅延で比較を行うと縦積み 3 段で構成したリプルキャリー型加算器が最も速く、動作電圧 0.2V において最適化した回路の演算時間は 151us であり、1 線 CMOS 型加算器に比べて約 1.8 倍の速度で動作していることが示された。以上の結果から、2 線ダイナミック回路に関して、最適化回路を組み合わせることで動作電圧の低い点においても優位性のある回路の作成ができたと考えられる。

参考文献

[1] N. Chabini, E. M. Aboulhamid, and Y. Savaria, “Determining Schedules for Reducing Power Consumption Using Multiple Supply Voltages,” $International\ Conference\ on\ Large\ Scale\ Integration\ Systems$, 2001, $pp.546-552$.

[2] A. V. Mezhiba and E. G. Friedman, “Scaling Trends of On-Chip Power Distribution Noise,” $IEEE\ Transactions\ on\ Very\ Large\ Scale\ Integration\ Systems$, $Vol.12$, $Issue4$, $Apr.2004$, $pp.386-394$.

[3] Y. Liu and T. Hwang, “Crosstalk-Aware Domino-Logic Synthesis,” $IEEE\ Transactions\ on\ Computer-Aided\ Design\ of\ Integrated\ Circuits\ and\ Systems$, $Vol.26$, $Issue6$, $Jun.2007$, $pp.1155-1161$.

[4] N. Karaki, T. Nanmoto, H. Ebihara, S. Utsunomiya, S. Inoue and T. Shimoda, “A Flexible 8b Asynchronous Microprocessor based on Low-Temperature Poly-Silicon TFT Technology,” $Proceedings\ of\ IEEE\ International\ Solid-State\ Circuits\ Conference$,2005, $Feb.2005$, $pp.272-598$.

[5] A. Bink, and R. York, “ARM996HS: The First Licensable, Clockless 32-Bit Processor Core,” $Micro$, $Vol.27$, $Issue2$, $Mar-Apr.2007$, $pp.58-68$.

[6] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger and L. Alvisi, “Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic,” $Proceedings\ of\ International\ Conference\ of\ Dependable\ Systems\ and\ Networks$ 2002, $Nov.2002$, $pp.389-398$.

[7] Martin Simlastik, Viera Stopjakova, Libor Majer, and Peter Malik, ”Clockless Implementation of LEON2 for Low-Power Applications” $Design\ and\ Diagnostics\ of\ Electronic\ Circuits\ and\ Systems$, 2007, $April11-13$, $pp.1-4$.

[8] A. J. Martin, and M. Nystrom, “Asynchronous Techniques for System-on-Chip Design,” $Proceedings\ of\ the\ IEEE, Vol.83, Issue1, Jun.2006, pp.1089-1120.$

[9] Al Davis, and Steven M. Nowick, “An Introduction to Asynchronous Circuit Design,” $UUCS-97-013, Sep.1997, pp.1-58.$

[10] L. G. Heller, W. R. Griffin, J. W. Davis, and N. G. Thoma, “Cascode Voltage Switch Logic: A Differential CMOS Logic Family,” $Proceedings\ of\ International\ Solid-State\ Circuits\ Conference, Feb.1984, pp.16-17.$

[11] 曽我部拓, “二線式ダイナミック論理を用いた自己同期プロセッサの信頼性評価 “ 平成 20 年度東京大学大学院工学系研究科電子工学専攻修士論文, 2009

[12] 中里輝希, “二線式スタティック論理を用いた自己同期回路の設計と評価 “ 平成 22 年度東京大学電子情報工学科卒業論文, 2010

[13] J. MyeongGyu, T. Nakura, M. Ikeda, K. Asada, “Mebius Circuit: Dual-Rail Dynamic Logic for Logic Gate Level Pipeline with Error Gate Search Feature“ $VLSI2009, Proceedings.NineteenthGreatLakesSymposiumon, May2009$

[14] Yusuke Tsugita, Ken Ueno, Tetuya Hirose, Tetsuya Asai, and Yoshihito Amemiya, “Process compensation techiques for low-voltage CMOS digital circuits“ $TheInstituteofElectronics, InformationandCommunicationEngineers, May2009$

本研究に関する発表

神谷歩未, 池田誠, ” 自己同期システムに向けた低電圧動作回路の最適化” 集積回路研究会,2011 年 8 月,

神谷歩未, 池田誠, ” 最適化した自己同期向け回路の比較” 電子情報通信学会 ,2012 年 3 月,$C-12-63(submitted)$,

謝辞

本研究を進めるにあたり、日頃から暖かいご指導をして頂き、また良好な研究環境を与えて下さいました、東京大学大学院工学系研究科　池田誠准教授、浅田邦博教授、名倉徹准教授、飯塚哲也講師に心より深く感謝致します。

研究活動のみならず、他の様々な場面においても多くの貴重なご意見を頂いた研究員の Tse-Wei Chen 氏、大学院生の門馬太平氏、Bushnaq Sanad Saleh 氏、Devlin Benjamin Stefan 氏に深く感謝致します。

日頃から研究活動を共にし、多くの助言を頂いた大学院生の斎藤総氏、胡興華氏、吉川俊之氏、中里輝希氏、王楠氏、久保田透氏、児玉和俊氏、矢部紘貴氏、卒論生の中村陽二氏、宮崎耕太郎氏、研究生の Parit Kanjanavirojkul 氏に深く感謝致します。

様々な場面において、数多くのご助言、ご支援を頂きました佐々木正浩准教授 (芝浦工業大学)、Nguyaen Ngoc Mai Khanh 博士、金鎮明博士 (SumSung)、萬代新悟氏 (TU Delft)、服部慶士氏 (グーグル)、程在鉉氏 (LG 電子)、玉置裕基氏 (任天堂株式会社)、宋暁旭氏 (ゴールドマンサックス)、秘書の横地順子氏、山中知歩氏，吉田直美氏，猪股啓子氏に深く感謝致します。

研究で行き詰まったときに、話を聞いてくれたり励ましてくれた同電気系工学専攻の作田真理子氏、鈴木麗菜氏にも深く感謝致します。

本チップ試作は東京大学大規模集積システム設計教育研究センター (VDEC) を通し、富士通株式会社、ローム株式会社、および凸版印刷株式会社の協力で行われました。

ご協力頂いたことに深く感謝致します。