Open-It FPGA トレーニングコース（入門編）

受講前準備資料

第 2.1 版 2015 年 7 月 6 日

内田 智久

高エネルギー加速器研究機構

素粒子原子核研究所

エレクトロニクスシステムグループ

履歴

版	更新日	更新内容
1.4	2014 年 9 月 1 日	2014 年度トレーニングコース用に「デジタル回路の基礎知識」として作成
2.0	2015 年 6 月 18 日	章構成変更
2.1	2015 年 7 月 6 日	誤字訂正、一部文章書き直し

0.1　はじめに　第 2 版

2014 年度の FPGA トレーニングコースから予習用の準備資料を作り配布してますが、受講者の方々から「組み合わせ回路や順序回路などのデジタル回路についても資料に入れて欲しかった」と要望をいただきました。
そこで、内容を全面的に見直し、トレーニングコースで行っていた基礎知識に関しての講義内容を全て準備資料に含めました。準備内容が増えましたが、分からない事を事前に調べることができるので理解が進むと期待しています。受講前の限られた時間で全ての疑問点を解決する事は出来ないかもしれません。そのような時は分からなかったことを明確にしトレーニングコース受講時に質問してください。
皆さんとトレーニングコースでお会いできる事を楽しみにしています。

2015 年 7 月 6 日　筆者

0.2　はじめに　第 1 版

Open-It FPGA トレーニングコースを 5 年以上開催していますが、最近トレーニングコースで行う演習時間を長くして欲しいと頼まれる事が多くなりました。トレーニングコースの開催期間を 2 日間から 3 日間へ変更する事も検討したのですが、周りの方々に相談したところ 3 日間以上になると参加できなくなる人も増える事が分かりました。そこで、トレーニングコースでの説明内容からデジタル回路に関する基礎知識の部分を受講者の皆さんに予習していただく事で演習時間を延ばす方法を採用する事にしました。これまでも受講者の方に事前に予習しておく事を教えてほしいと聞かれた事も少なくありません。これは、初めて回路について学ぶので不安を感じているのだと筆者は想像しています。この様な不安もこの文書で払拭されると期待しています。受講者の皆さんには負荷をかけてしまいますがご協力お願いいたします。
この文章には Open-It FPGA トレーニングコースを受講する前に知っていて欲しい事をまとめてあります。この文章のみで完全に理解することは難しいかもしれません。ここで説明する内容はトレーニングコースで簡単に復習しますので完全に理解する必要はありませんが理解できなかった点や疑問点などを書き出してから受講してください。そして、トレーニングコース受講時に講師にその疑問点について質問して解決してください。
これから新しい事を学ぶときに大切なことは学ぶ対象全体に共通する原理原則を概念的、感覚的に理解することだと筆者は考えます。共通する原理原則を理解する事で理解が要素毎から要素間に拡張されるからです。この文章の目的はデジタル回路全体に共通する概念を伝えることです。筆者は概念を伝えるためには定性的な説明の方が適していると考えていますので、この文章での説明は全て定性的です。悪く言えば厳密な説明はありません。この文章の内容に物足りなさを感じる方も居ると思います。その様な方にとってこの文書が詳細な書籍を読むきっかけになれば筆者としては非常に嬉しく思います。
本文章に書かれている各章は独立です。しかし、前に書かれている章の知識を後ろの章ではすでに知っているとして書いていますので、先頭から順番に読み既に知っている節は読み飛ばす方法をお勧めします。

皆さんとトレーニングコースでお会いできる事を楽しみにしています。

2014 年 7 月 12 日　筆者

目次

第１章

導入～デジタル回路を学ぶ前に～

1.1　デジタル技術の特長

デジタル技術は私たちの日常生活の中でも広く使用されています。以前アナログ技術で実現されていた装置が年々デジタル化されています。例えば録音された音楽を聴くとき、以前はアナログ技術を用いたレコード盤を使用していました。しかし、今ではデジタル技術を用いた CD やデジタルプレーヤを使います。テレビ放送もアナログ技術を使用していましたがデジタル化されました。身の回りにある電気製品の多くがデジタル化され続けています。

この様に近年多用されているデジタル回路をこれから学んで行きますが、デジタル回路について学ぶ前に何故デジタル化が進むのかを確認します。何でもデジタル化すれ良いわけではなく、もちろん何でもアナログ化すれば良いわけではありません。どちらも長所と短所を持っています。より質の高い実験データを得るために両技術を適切な場所に使用して計測装置を開発する必要があります。ここではデジタル回路を用いる理由である長所を確認します。

デジタル回路の長所を音楽を聴くときに使用するレコード盤と CD を比較することで考えてみましょう。

図 1.1 はレコード盤を用いて音楽を聴く装置、レコード・プレ-ヤの概念図です。

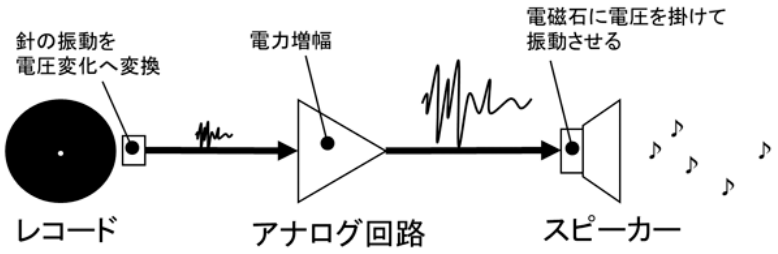


図 1.1　レコード・プレーヤの概念図

これはレコード・プレーヤを概念的に書いた図です。音楽はアナログ信号であり、音楽を聴く人間も音であるアナログ信号を受け取ります。音楽情報はレコードの溝として記録されています。その溝を針でなぞる事で針が振動します。針には磁石が付いていて、コイルの中で振動するようになっています。針の振動が磁石を動かし、その磁石がコイルに誘起する電圧を取り出します。小さな振動で発生させた電気信号なので電圧振幅は小さく電力も小さいのでアナログ回路で増幅します。大きな信号へ増幅した信号によりスピーカーを駆動することで空気を振動させ音楽を再生します。スピーカーは振動板、コイル、磁石で構成されています。磁石の周りに巻かれたコイルに電流を流すことでコイルが振動します。コイルには振動板が接続されているので音が再生されます。

この装置はとても分かりやすいと思いませんか？

アナログ信号をアナログ記録し、アナログ信号を読み出し、その信号をアナログ回路で増幅して再生する。しかし、今ではデジタル化されています。次はデジタル技術を使用している CD について考えます。図 1.2 は CD プレーヤーを概念的に表した図です。

先ほどのレコード盤と比較すると図中に信号処理と書いたデジタル回路が増えています。この事に注意してください。CD の場合は記録されている音楽情報が既にデジタル化

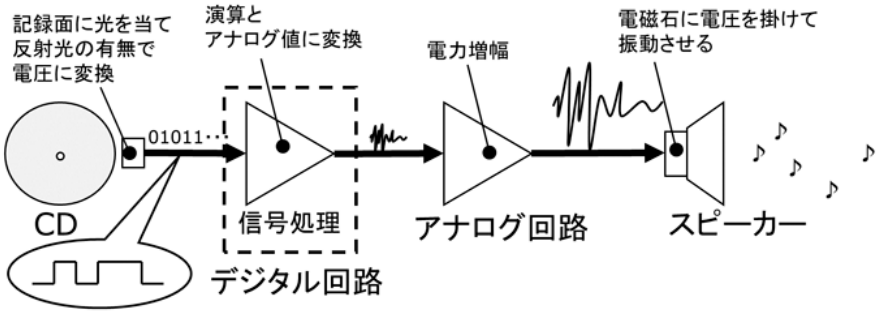


図 1.2 CD プレーヤーの概念図

されています。CD に光を当て反射光の有無で信号を再生します。図中の波高値と書かれている部分の波形が再生波形の例です。横が時間軸で縦が光の反射量です。光量を電圧へ変換して電気信号にします。すると時々刻々と変化する矩形波が出力されます。この信号をデジタル信号処理回路で信号処理を行い、その後、デジタル値をアナログ値へ変換する回路 Digital-to-Analog Converter (DAC) を用いてアナログ信号へ変換します。アナログ信号へ変換された後はレコードプレーヤと同じです。電力増幅してスピーカーを鳴らします。

ここで、皆さん、不思議に思いませんか？

デジタル化された CD は新たに回路が増えているのです。一見、物事を複雑にしただけのように見えます。しかし、デジタル技術の長所により CD の方が総合的な性能が良くなります。何故でしょうか？その理由がデジタル技術を採用する理由です。

デジタル回路（技術）の長所とは何でしょうか？

次を読む前に少し考えてください。

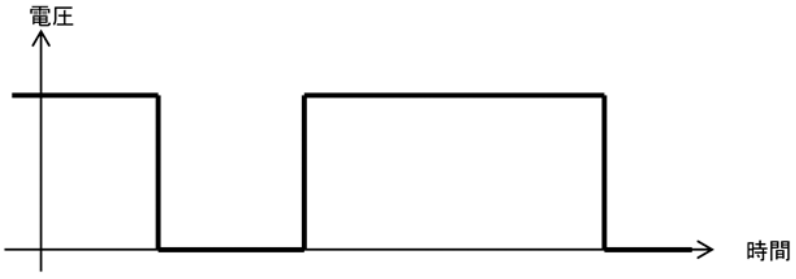


図 1.3 理想的なデジタル信号波形の例

長所を説明する前にデジタル信号について復習します。デジタル信号とは例えば図 1.3 のような信号です。電圧が時間変化しますが電圧が高い状態と低い状態の二つの状態間を行き来する信号です。一対の信号線で伝えることができる情報は 2 状態のみです[*1] 。このような信号を採用する長所はなんでしょうか？

それはノイズなどの外乱に強いことです。信号伝搬の実験経験がある方は実験を思い出し

[*1] ここでは電圧を状態値として使用しましたが電流を用いることもあります

てください。信号波形を変化させずに伝送する事はとても難しいです。先ほどのデジタル波形も電線を用いて伝送すると図 1.4 のような波形になるかもしれません。

図 1.4　実際のデジタル信号の例

この図はかなり大げさに書きましたがこのような波形の乱れを観測する事ができます。この様に波形が乱れてしまってもデジタル信号なら伝え誤ることはありません。それは図 1.5 に示すように電圧の高低を判断する電圧が一点ではなく範囲だからです。

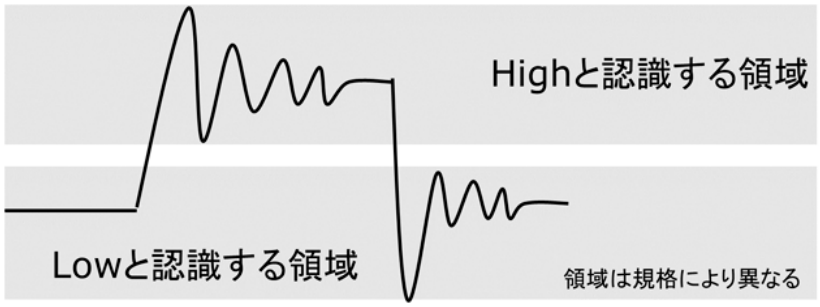


図 1.5　電圧の高低を判定する電圧範囲

仮にこの信号がデジタルではなくアナログ信号で矩形波を伝搬させようとしていたと考えると元の波形と全く異なる値を受け取ることになりますので誤った信号処理を行ってしまいます。ところがデジタル信号では間違えなく伝えることができるのです。
デジタル信号は情報量を二つに下げ、一度に伝送できる情報量を犠牲にすることでノイズ耐性を高めていると考えることができます。アナログ信号は信号の全ての点の電圧が信号情報なので情報量は非常に多い、点の数と考えると無限大！
デジタル回路はアナログ信号と異なり物理量（連続値）を伝送するのではなく数値を伝送していると考える方が自然です。アナログ信号は電圧や電流などの物理量（連続量）が信号情報となります。全ての点に意味があるため雑音と信号の識別はできません。デジタル信号も電圧や電流などの物理量を信号伝達に使用しますが、その物理量の値を使用するのではなく範囲を使用して 0 と 1 の値である離散値として情報を伝達します。アナログ信号処理の場合は電流や電圧をそのまま連続量として処理しますが、デジタル回路ではもはや電圧や電流などの物理量（連続量）の概念はなく数値（整数値）として扱い処理します。
デジタル技術の長所をまとめます。最大の長所はノイズに強いことです。ノイズに強いので信号をデジタル化する事で信号を間違って処理する確率をとても低くすることができます。検出器から出力される信号は微弱なアナログ信号です。そのまま伝送や処理を行うと

ノイズの影響を受けて本来の波形と異なってしまいます。そこで、計測システムでは検出器信号の劣化を最低限に抑えるために検出器が出力する信号をアナログ回路で必要最低限処理し、できるだけ早くデジタル信号へ変換します。一度デジタル化してしまえばノイズに強くなるので信号の扱いが簡単になります。例えば光ファイバを用いてデータを長距離伝送することも簡単に行うことができます。デジタル化の長所は様々な事がありますが、様々な機能を実現可能にしている背景にノイズに強いことが深く関係していることが多いです。

デジタル信号は数値ですからデジタル回路は 2 進演算を行う回路と考えることができます。数値演算を間違えることなく実行できるので数学的なテクニックを用いてアナログ技術では実現が難しい回路を作ることができます。例えば、先ほどの例で登場した CD で用いられているエラー訂正符号技術があります。CD の表面を見ると表面が汚れていたり、傷がついていることが少なくありません。そのような汚れや傷でデータを読み間違うことはないのでしょうか？実は間違えているのです。デジタルでも全てのノイズや外乱の影響を完全になくすことはできません。CD の読み出しが間違っていても正しく読み書きできる、これはとても不思議です。コンピュータはデータが少しでも間違えれば文字化けや暴走してしまうはずです、何故なら全く異なる値になってしまうのですから。実は CD では間違いが起こることを想定して本来のデータ以外にエラー訂正符号と呼ばれる特別なデータが書き込まれています。この特別なデータを用いてデータを計算するとエラーが起きている個所を特定する事ができます。デジタルが取る状態は 2 つですので間違っている場所が分かれば値を他方へ変換すれば正しい情報となります。例えば間違っている位置の値が 0 であれば 1 へ変換することで正しいデータとなります。この様な処理をアナログ回路で実現する事は容易ではありません。この様な機能を実現する為にデジタル回路は最適です。

この様な長所があるデジタル技術ですが短所もあります。デジタル処理の概念はとても古くトランジスタが発明される以前からありました。しかし、デジタル処理の短所により採用が見送られてきたのです。

その短所とは回路規模が大きくなることです。

デジタル信号は 1 対の信号で送ることができる情報が 2 状態のみです。例えば時々刻々変化する 0 から 255 の値を取る信号波形を送るなら多くの信号線（例えば 8 対の信号線）が必要です。信号の両端には必ずトランジスタを使用した回路が必要です。アナログ信号であれば 1 対で良い回路が複数対の回路が必要になるのです。これはアナログ回路と比較すると回路規模（例えば使用トランジスタの数）がとても多くなります。この理由により集積化が難しかった時代はデジタル回路を採用する事ができませんでした。しかし、近年の微細化加工技術の発展により大規模回路を数 mm 四方の大きさに作る込むことができるようになり問題ではなくなりました。短所が解決された事でデジタル回路が多数採用され現在のようにデジタル化が進みました。

ポイント：デジタル技術はノイズ耐性が高い

1.2　データ収集システムとデジタル回路

実験・観測で用いられる計測システムは非常に複雑な構成で各実験・観測プロジェクト毎に異なっています。ここでは、実験・観測で用いられる計測システム共通部分である、データ収集システム (Data acquisition system: DAQ システム) を例として用いてデジタル回路がどのような役割を担っているのか説明します。最後に私たちが設計する機会が多い部分について説明します。

1.2.1　データ収集システム

実験・観測する立場から見た場合、最も興味があるのは測定・観測データです。検出器・センサーにより物理事象が電気信号へ変換され最終的にコンピュータに送られ、解析やデータ蓄積が行われます。この様に検出器・センサーから出力された信号を処理し解析できる状態に変換するシステムを DAQ システムと呼びます。DAQ システムの定義は人により異なりますが、ここでは検出器・センサーから出力された信号をデジタルデータとして保存するまでを DAQ システムと呼ぶ事にします。
DAQ システムの目的は検出器・センサーから出力された信号を処理してコンピュータで解析可能なデジタルデータとして保存することです。最終的に保存されるデータの大部分は下の値により構成されています。

1. 検出器・センサーが出力するアナログ信号の波形情報
   （a）最大値
   （b）ある値を超えた部分の面積
   （c）ある値を超えている時間
2. 信号を検出した時刻

いつどのような信号が発生したかをデジタルデータとして保存する事ができればコンピュータにより解析する事で様々な現象を研究する事が可能になります。
実際のシステムでは検出器・センサーの数が数万を超えることも珍しくない、という事に注意してください。例えば画像を取得する事を想定してみましょう。VGA サイズの画素数は 30 万程度（640 × 480）になります。簡単に 10 万を超えてしまいました。実際のシステムはこのように非常に多くのデータ量を扱うことになるのです。
検出対象により計測システムは異なりますが、DAQ システム部を抜き出すと概念的に図 1.6 のように書くことができます。

具体的な検出器は実験・観測毎に異なりますが、多くの検出器はアナログ（連続）量を電圧、電流、電荷などとして出力します。このようなアナログ量として扱う必要がある電気信号をアナログ信号を呼ぶ事にします。これらの検出器が出力したアナログ信号はフロントエンド回路と呼ばれる回路で処理されデジタル信号へ変換されます。デジタル信号はバックエンドと呼ばれるコンピュータへ送られます。コンピュータへ送られてしまえばソフトウエアにより様々な処理を行うことができることを皆さんは良く知っていると思いま

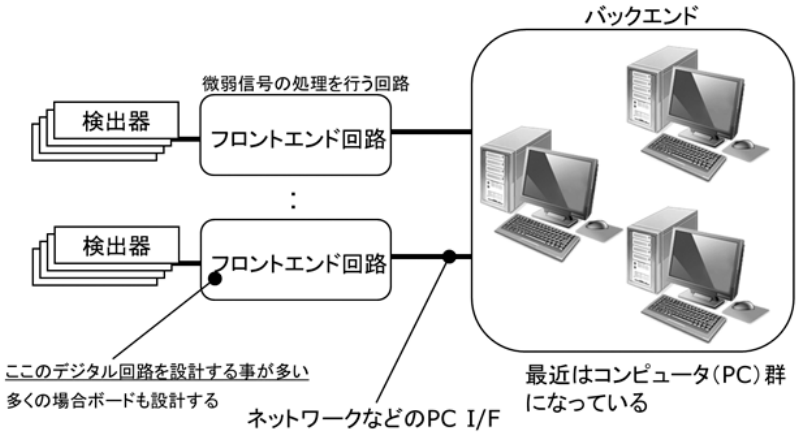


図 1.6　DAQ システムの概念図

す。実際のシステムではデータ量が膨大になるためコンピュータを複数使用して処理しますが概念的には同じです。フロントエンド、バックエンドは各システム内での呼び方なので一般的な決まりはありませんが検出器に近い方をフロント、遠い方をバックと呼ぶ事は共通しているように思えます。

デジタル回路はフロントエンド回路内のデジタル信号へ変換する部分からコンピュータまでで使われています。近年の装置ではデジタル回路の比率が高くなっています。デジタル回路の特長で説明しましたが、デジタル回路はノイズに強いため検出器信号をできるだけ早くデジタル化する事で信号劣化を最小限に抑える構成が採用されています。

測定・観測データは最終的にコンピュータに入力する必要があります。従って、フロントエンド回路の出力はコンピュータと接続できる一般的な通信形式（インターフェース、I/F）が有利です。例えば、イーサネットなどの通信規格、USB などの民生規格などがありますが、実験に適した I/F を選択する事が重要です。誤った I/F を採用すると期待した性能を得ることができなかったり、想定以上のコストが発生したりします。十分に注意して選択してください。

概念図中のバックエンドはコンピュータで構成されるので通常は一般に市販されている製品で構成します。私たちは残る箱である“フロントエンド回路”を開発ることが多いです。次にフロントエンド回路について説明します。

1.2.2　フロントエンド回路

フロントエンド回路を単純化したブロック図で表現すると図 1.7 のように書くことができます。

理解を助けるためにフロントエンド外部の装置である検出器とコンピュータも書き加えてあります。検出器とコンピュータで挟まれた 3 つの箱（ブロック）がフロントエンド回路の機能です。これらの機能を順番に説明します。

検出器が出力したアナログ信号はフロントエンド回路の入り口であるアナログ信号処理回路に入力されます。入力されたアナログ信号を増幅および波形整形を行いデジタル化に適

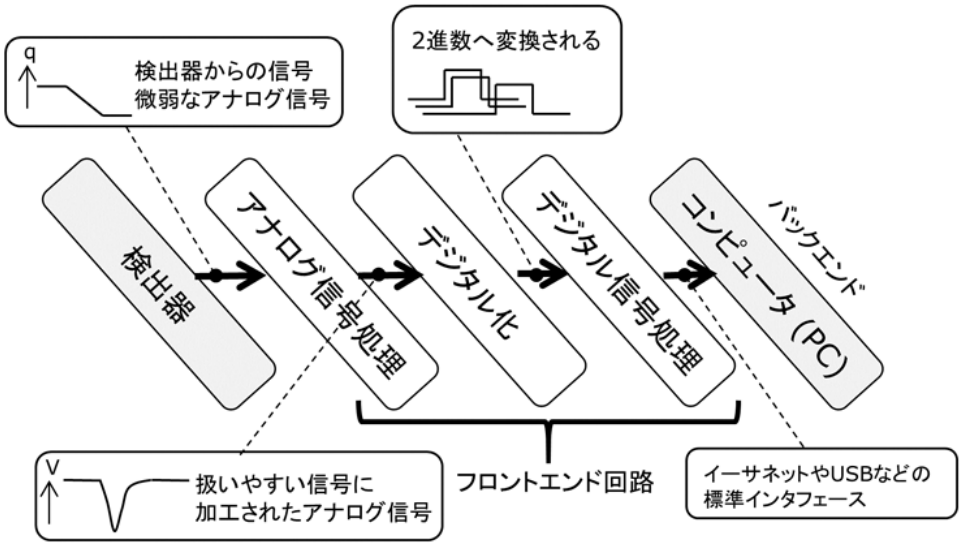


図 1.7　フロントエンド回路のブロック図

した信号へ変換する事がアナログ信号処理回路の役目です。
検出器が出力する信号は非常に小さく微弱な信号です。検出器を工夫する事で出力信号を大きくする場合もありますが、電子回路での増幅が有利であることが多いです。有利な点と欠点が複数ありお互いが関係しているので、ここで一つの長所を以て電子回路での増幅採用理由を示すことは難しいですが、電子回路は集積回路技術により小型化する事ができることやケーブルなどで装置間を配線できるので配置の柔軟性が高い事を代表的利点として挙げることができます。例えば、増幅度を 10 倍にするために検出器の大きさ（容量）が 2 倍になるとすると、これは検出器を同じ空間に並べる場合に数が半分になるので解像度が落ちてしまいます。これでは価値が半減してしまいますので増幅度は 1/10 で解像度を上げて増幅は電子回路で行う方が魅力あります。

図中での検出器出力は放射線検出器を想定して書いてありますので電荷です。放射線検出器の多くは検出対象の放射線エネルギーにより物質を電離させ、その電離された電荷を集めることで検出します。
検出器出力信号が電荷であれ、電流電圧であれ、アナログ信号処理回路にが入力されます。この信号をデジタル化しやすい信号波形に変換します。微弱な信号をノイズに埋もれないように増幅する必要があります。特別な例を除いて、入力が電圧でない（電流や電荷）場合は電圧へ変換します。電子回路は電荷や電流を扱うよりも電圧として扱う方が有利な場合が多いです。消費電力を抑える工夫ができるからです。電圧はポテンシャルですから静的にも動的にも扱うことができますが、電流は動的なものなので常に電荷を移動させるエネルギーが必要になり消費電力が増えます。この様に信号を増幅したり、電荷を電圧へ変化したりすることがアナログ信号処理回路の第一の役目です。
電圧へ変換された信号ですが、扱いやすい波形に形を整える必要があります。入力された信号に比例する信号が出力される際にどの程度の時間出力するのか決める必要があります。稀な事象を検出するのであれば長時間値を保持する回路が適していますし、頻繁に検出される事象は事象間の信号が重ならないように短いパルス状の信号波形の方が適しています。実験・観測の目的に合わせた波形を出力する必要があります。この適した波形に整

形する事がアナログ処理回路の第二の役目です。これでデジタル化の準備ができました。デジタル化とはある時刻の電圧をデジタル値へ変換する事です。必要な精度のビット数でデジタル値へ変換します。もっとも単純なデジタル化はアナログ信号がある基準値を超えたか超えていないかにより 0 または 1 のデジタル信号を出力する方法です。この場合、ビット数は 1 です。アナログ値を 256 段階で表現したければ 8 ビットのデジタル値へ変換します。アナログ信号をデジタル値へ変換する回路を Analog-to-digital converter (ADC) と呼びます。計測システムでは高速信号処理を行うことが多いのでアナログ信号を周期的にデジタル値へ変換する方法が採用されます。この変換周波数を表す単位として SPS (Samples par sec) が用いられます。例えば 20 ns 周期でデジタル値へ変換する場合は 50 MSPS と書きます。

デジタル値へ変換された信号はデジタル信号処理回路に入力されます。デジタル信号処理回路の役目は実験観測に必要な情報を抽出する事とコンピュータへデータを転送する事です。デジタル化した全ての信号情報をコンピュータへ転送することができるなら大きな問題はないですが、データ量が膨大になる場合は全データを送ることができません。そのような時は目的の信号あ含まれていると思われる部分のデータのみコンピュータへ転送する方法を採用します。データを抽出する方法は幾つかありますが代表的な方法の一つがゼロサプレスと呼ばれる方法です。目的の現象を検出した時のみ検出器の出力が変化します。通常は信号を受けると絶対値が大きな方向へ変化します。信号がない時はある値よりも小さくなっています。この特性を使って信号がない部分を削り取ります。ある基準値を設け、その基準値を超えた部分のデータのみ転送するように回路を設計します。この事でデータ量を削減する事ができます。実際は基準値を超えた部分のみではなく前後一定期間のデータをコンピュータへ送り、後に実験目的に合わせた処理を行います。コンピュータへ転送するデータが決まったら、そのデータをコンピュータ I/F を用いて転送します。コンピュータ I/F を用いてのデータ転送は I/F 規格で定められた動作を守ることで初めて転送できます。規格という決まり事を皆が守るから様々な装置がお互いに通信ができるのです。従って、規格に合わせて動作しなければいけません。これら規格に合わせて動作する作業もデジタル信号処理回路の役目です。これでやっと最終目的である実験観測データをコンピュータへ転送する事ができました。フロントエンド回路の仕事はここまでです。

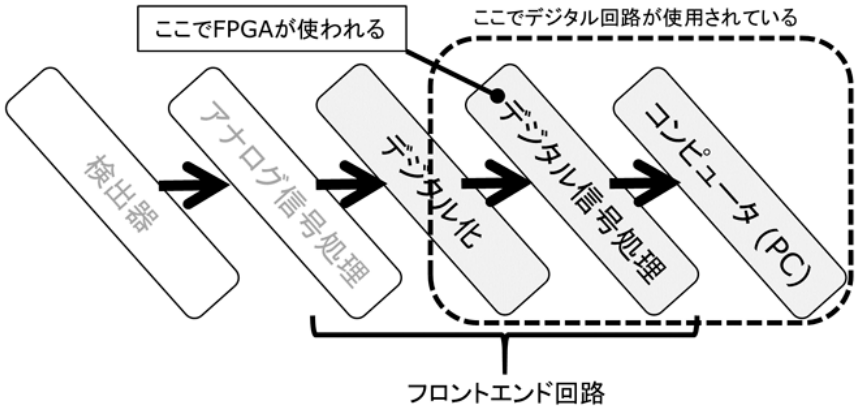


図 1.8　フロントエンド回路のブロック図

フロントエンド回路の役目を見てきましたが、デジタル化以降全てでデジタル回路が

用いられています。このデジタル回路部分で FPGA と名付けられたチップが近年多用され、私たちもこの部分を開発する機会が増えています（図 1.8 参照）。次にデジタル信号処理ブロックの中を詳細に説明します。

1.2.3　デジタル信号処理

DAQ システム内でのデジタル信号処理回路の役目はデジタル化したデータをコンピュータへ転送する事です。実際は実験内容に合わせた多種多様な処理を行っています。例えば、デジタル化した値から関数フィッテングにより特長パラメータを抜き出すこともできます。ここでは、必要最低限の機能について説明します。

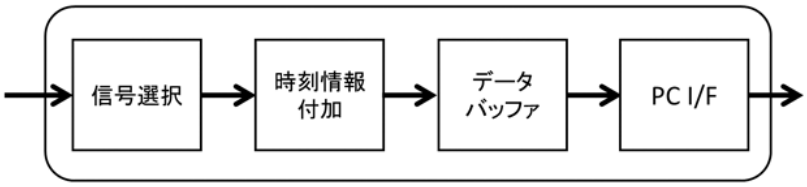


図 1.9　代表的なデジタル処理回路のブロック図

図 1.9 は代表的なデジタル処理回路のブロック図です。ここで扱う信号処理回路は次のブロックで構成されています：

- 興味があると思われる事象のデータを選択する信号選択ブロック
- 事象が起きた時刻をデータに付加する時刻情報付加ブロック
- データを一時保存するためのデータバッファ・ブロック
- PC へデータを転送するための PC I/F ブロックでデジタル信号処理回路

PC I/F を除くブロックの順番は入れ替えることができます。

1.2.4　信号選択ブロック

興味があると思われる事象のデータを選択する回路です。デジタル化したデータを全てコンピュータへ送ることができる時は不要となる回路です。デジタル化したデータを全て転送しようとするとデータ量が膨大になり全て送れない事があります。データを転送できたとしても現実的な時間で解析できないなどの理由でデータ量を削減しなければならくなることがあります。代表的なデータ量削減方法は前節で説明したゼロサプレスです。他に加速器科学で一般的に使われるトリガー機構による信号選択方法があります。トリガー機構の動きについて以下簡単に説明します。数多くの検出器で構成されている場合、その一部の検出器がある条件を満たした場合にトリガーと呼ばれる信号を各デジタル処理回路に対して発行し、トリガーを受けた時刻の前後の一定期間データを選択します。トリガーを発行する条件は測定対象となる物理に依存します。もっとも簡単なトリガー条件は全ての検出器の出力を監視して、一定数以上の検出器が信号を検出した場合に発行する方法で

す。この方法では雑音による検出器信号のデータ転送を抑えることができます。

1.2.5 時刻情報付加ブロック

複数のフロントエンド回路を使ってデータ収集する場合、異なるフロントエンド回路で収集されたデータを最終的にあるコンピュータに集めます。このデータからある特定の事象のコンピュータ上で再構成するためには事象が発生した時刻またはデータを受け取った時刻をデータに付加し、常に事象データと時刻を合わせて一つのデータとして扱う必要があります。この方法によりコンピュータ上での事象再構成が可能になります。

1.2.6 データバッファ・ブロック

データを一時記憶装置に保存するブロックです。コンピュータと通信する場合データバッファは必須です。データバッファは主に 2 つの役目を担っています。

1. データ転送速度変換機能
2. 待機機能

データ転送速度変換機能は汎用的なコンピュータ I/F を使用する場合必ず必要です。汎用的な I/F は世界標準規格で定められた転送速度で動作しなければいけません。通常は一定のデータサイズが必要です。例えば、ギガビットイーサネットを用いてデータを転送する場合、測定データの量に限らずコンピュータと通信する時はデータをギガビット速度で転送しなければいけません。そこで、コンピュータに転送するために必要なデータ量になるまで一時保存し、必要な量に達したらまとめてコンピュータに転送します。速度変換する場合、出力速度は入力速度と同じか大きい必要があります。さもないといずれデータが溢れてデータ損失が発生します。
待機機能はコンピュータが他の処理を行っているためにデータを受け取れない時にコンピュータが受け取ることができる状態になるまで待つ機能です。待っている間に発生する計測データを一時記憶装置に保存します。その後、コンピュータが受け取ることができるようになったら規格で定められた転送速度でデータをコンピュータへ転送します。

1.2.7 PC I/F ブロック

規格に従いデータをコンピュータへ転送するブロックです。規格はデータを転送する手順、一度に送るデータの量、データを送る順番など様々な事が定められています。それらを守りながらデータをコンピュータへ転送する機能です。
規格としてはイーサネット、USB、PCI express などが使用されることが多いです。規格により長所と短所がありますので適切な I/F を採用してください。

1.3　標準 IO 規格

皆さんが計測システムや計測装置を開発する場合、一つのチップで全ての処理を行うことはありません。目的の機能を実現するために複数のチップや装置を使用します。例えば、検出器やセンサー信号をデジタル化したデータを FPGA で処理するために、センサー信号をデジタル化するチップ（ADC）と FPGA を接続してデジタル化したデータを FPGA へ転送しなければいけません。

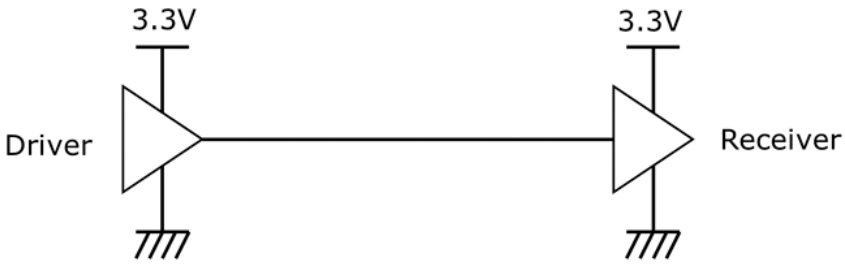


図 1.10　チップ間通信

例として図 1.10 のような状況を考えます。この図は異なるチップ間の配線を表しており、信号を駆動する Driver と信号を受信する Receiver 回路は異なるチップに搭載されています。デジタル信号をチップ間で通信する場合、その信号について規則を設ける必要があります。デジタル信号は 2 つの状態を伝送する事を説明しましたが、この 2 つの状態を区別するための条件を決め、接続する全てのチップがその条件を守る必要があります。定めるべき主な条件は下の二つです。

- High level（H レベル）と認識するための条件
- Low level（L レベル）と認識するための条件

電圧変化により状態を伝える場合、何ボルト以上が H レベルであり、何ボルト以下が L レベルと認識するのか予め決める必要があります。
条件を独自に決めるとお互いが通信できなくなってしまいます。そこで、世界的に統一規格が定められています。この規格を使用する事でお互いを接続して通信する事ができます。この規格は用途毎に多種多様なものが定められています。FPGA は様々な規格に対応する事が出来ますので接続されているチップの規格に FPGA 側を合わせて使用してください。FPGA で設定できる標準 IO 規格はデータシートに書かれています。
規格例として頻繁に使用する LVCMOS33 と LVDS と呼ばれる規格について簡単に説明します。

1.3.1　LVCMOS33 の信号波形

LVCMOS33 は図 1.11 と同じように一つの信号線の両端に送受信回路が接続されます。各チップは基準電位（0 V）となる共通のグランドと電圧 +3.3 V の電源が接続されてい

ます。受信側で 2 つの状態を区別するための条件は図 1.11 の様に定められています。

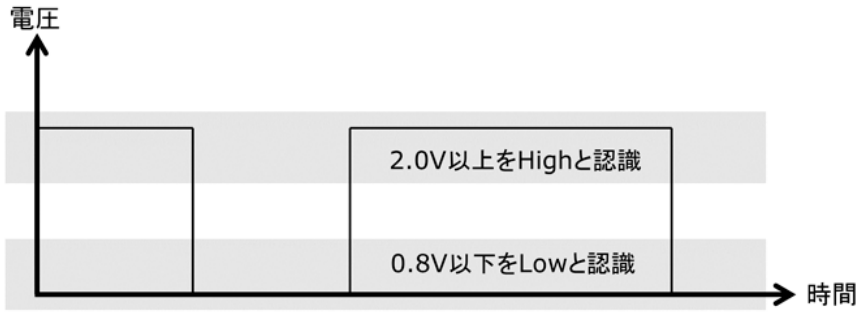


図 1.11 LVCMOS33 規格

図中の実線が信号を表しています。この規格では 2V 以上を電圧が高い状態と認識し 0.8V 以下を電圧が低い状態と認識します。注目してもらいたいのは 2 つの状態間にどちらの状態でもない隙間がある事です。この隙間は禁止領域と呼ばれ、この隙間に信号がとどまる事は許されていません。もちろん変化時にこの領域を通過する事は許されています。モノづくりの観点ではこの隙間はとても重要です。この様な隙間を設けることができる事がノイズに強いデジタルらしい工夫です。
この隙間にどのような意味があるのでしょうか？
この隙間は送受信回路の特性ばらつきを吸収するために設けられています。デジタル回路はトランジスタにより構成されています。この回路はある電圧より高い場合は High レベルと認識し低い場合は Low レベルと認識します。この判断電圧は判定点なので High レベルでもなく Low レベルでもない状態になります。その様な信号を回路が受信すると中間電位を出力してしまいデジタル値を正しく伝達できなくなってしまいますので禁止しなければいけません。さらに、この判定電圧はトランジスタの製造ばらつきや電源電圧値によりある範囲でばらつきます。回路を工夫してばらつき範囲を小さくする事は出来ますがゼロとすることはできません。そこで、ある範囲を禁止領域と決める事で特性ばらつきを吸収しお互いの通信を保証しているのです。

1.3.2 LVDS

LVCMOS はとても分かりやすく広く使われている標準規格ですが高速通信は苦手です。100MHz 程度が限界です。高速動作に適した信号規格として頻繁に使われるのが LVDS（Low Voltage Differential Signaling）です。こちらは 500MHz 以上まで動作します。
この規格を説明する前に皆さんに質問です。信号をどのように送れば高速化できると思いますか？少し考えてください。例えば、先の LVCMOS33 規格をどのように修正すれば高速動作可能となるでしょう？
答えは以下の 2 つです。

1. 振幅を小さくする
2. 信号波形の立下り、立ち上がり時間を短くする

この条件を満たすために作られた規格の一つが LVDS です。
先ずは振幅から考えてみます。図 1.11 での矩形波の立ち上がりと立下りは垂直に書かれていますが実際は垂直にはならず必ずある時間をかけて変化します。この遷移速度の単位は V/sec であり、この値は有限です。時間をかけて変化するので振幅が小さい方が遷移時間が短くなります。従って、高速化する一つの方法は信号の振幅を小さくすることです。遷移速度を大きくして信号波形の立下り、立ち上がり時間を短くするのが次の高速化方法です。変化速度は回路上に存在する静電容量の充放電で使われる時間です。従って速くするためには単位時間当たりの充放電量を大きくする、すなわち信号に流す電流を大きくすれば良いのです。
上の 2 つの方法、充放電時間を短縮するために信号に流す電流を大きくする、電圧変化にかける時間を短縮するために信号振幅を小さくする、これらを採用します。ところが一つ問題があります。振幅を小さくしたために、LVCMOS で説明した禁止領域が狭くなり雑音耐性が低下してしまうのです。

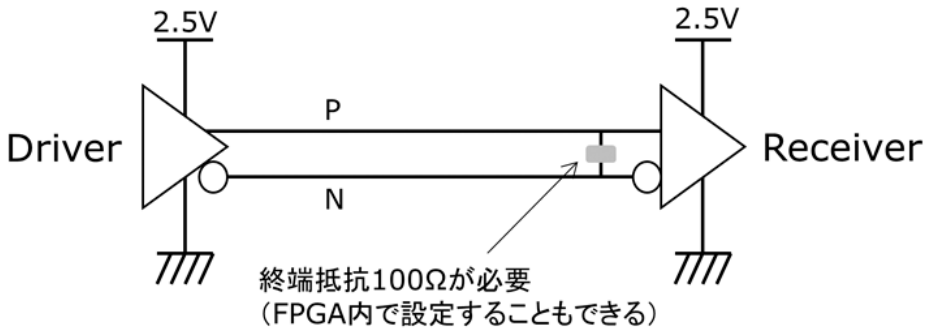


図 1.12　LVCMOS33 規格

この問題を解決するために LVDS 規格は差動伝送という方法を採用しました。図 1.12 が LVDS 規格で接続されたチップ間の配線です。共通電位のグランド以外に 2 本信号線があります。一つが P（正側）、他方が N（負側）と名付けられることが多いです。この 2 つの信号 P と N の電圧差 (P-N) を使用する事で見かけ上の振幅を 2 倍にして使用します。第二の戦略を用いて大きな電流を流すために波形の反射を抑える 100 Ωの終端抵抗を 2 つの信号間に接続する事も規格で定められています。この終端抵抗は FPGA で設定する事ができるので基板上に付けない事も多いです。電源電圧は通常 2.5 V です。他の電圧で動作するチップもありますが、標準は 2.5 V ですので他の電圧で用いる場合は規格が微妙に異なっていないかなどをデータシートで確認してください。

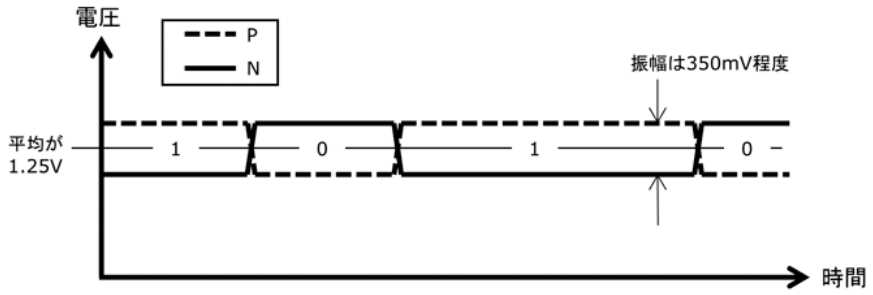


図 1.13 LVDS 信号波形

図 1.13 は LVDS の信号波形です。信号は 1.25 V を中心に振幅 350 mV 程度で変化します。図中の点線は P 側の信号、実践は N 側の信号を示しています。P-N が正の時は 1 、負の時は 0 と認識する事になっています。詳細な仕様はデータシートなどで確認してください。

1.4　記数法

ここではデジタル技術で使用される数字の表現について説明します。デジタル技術は 2 進数の演算や通信を行いますので数字を 2 進数表現が基本ですが、読み難いので 16 進数などを使用します。

1.4.1　2 進数

物や要素の個数などの数自体は普遍な概念ですが、その表現方法は多数あります。日常では個数などを数えるために 10 進数表現を使用していることは皆さんご存知だと思います。デジタル技術では 2 状態の信号をやり取りするので処理は 2 進数の演算が中心になります。頻繁に 2 進数表現を使用するので復習しておきましょう。
数 Z の N 進数表現を式で表現すると、数 Z を下式の様に表した時の a_i を $_i$ の小さい方から右から左へ順番に並べる表現方法のことです。

$$Z = \sum_{i=0}^{m} a_i N^i, a_i \in \{0, 1, ..., N-1\}$$

2 進数表現とは a_i の母集合が 0 と 1 で構成される場合です。例えば 10 進数の 6 を 2 進数で表現すると 110 となります。2 進数表現の 1 桁の事をデジタル技術ではビット（bit）と呼びます。また式の m の事をビット数と呼びます。m 桁の 2 進数は 0 から $2m-1$ まで表現できることを確認してください。2 進数で表現した時、$i = m$ の桁を最上位ビット (Most Significant Bit, MSB)、i=0 の桁を最下位ビット (Least Significant Bit, LSB) と呼びます。2 進、16 進表現した場合、通常は右端が LSB、左端が MSB となります。16 進表現については次の節で説明します。

1.4.2　16 進数

2 進数のビット数が増えるととても見難くなります。例えば、最近のコンピュータは 64 ビットの数値を使って動作しますが、64 桁の 2 進数の並びでは数字が分からないばかりか、メモを間違えなく取ることさえ慎重になります。
そこで少しでも見やすくするために 16 進表現が導入されました。2 進数を 4 ビット毎にまとめて分け、4 ビットを一つの桁として表現する方法です。この方法を採用するために 16 個の要素を新しく導入します。通常私たちが使用する数の要素は 10 個です。一桁で表現できる数字は 0 から 9 までの 10 個です。これでは足りません。そこで、アルファベットを使って要素を 16 個へ増やします。0 から 9 までの数字に加えて A, B, C, D, E, F の 6 個のアルファベットを加えます。アルファベットは A からそれぞれ十進数の 10, 11, 12, 13, 14, 15 と対応させます。例えば、2 進数で表現した場合の 10110010 は 16 進表現では 0xB2 となります、B2 の前の 0x は数が 16 進表現であることを示すために先頭に付ける識別記号です。
この方法で幾分か読みやすくなりましたが、見て直ぐに数字へ変換できる人は少ないです。しかし、この方法は頻繁に使用されます。その理由はデジタル回路で値を数値として使用することより符号として使用する機会の方が多いからです。符号として使用する場

合、2 進数のビット列パターンが重要となります。現時点では納得できないかもしれませんが、ソフトウエアや回路設計を体験すると実感できると思います。

第 2 章

デジタル回路

2.1　基本 4 要素

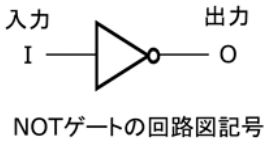
コンピュータに代表されるデジタル回路は次の 4 つの基本要素から構成されています。

- NOT ゲート（インバータ）
- AND ゲート
- OR ゲート
- 記憶素子

どんなに複雑な動きをする回路でもこの 4 つの要素へ分解する事が出来ます。各要素の動作を確認しましょう。

2.1.1　NOT ゲート

NOT ゲートはデジタル回路の基本形です。別名インバータとも呼ばれます。図 2.1 に回路図記号と真理値表を示します。



真理値表

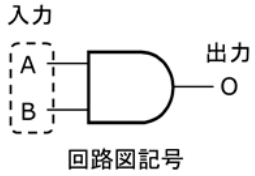
I	O
L	H
H	L

図 2.1　NOT ゲートの回路図記号と真理値表

真理値表とは入出力の論理関係を表した表です。NOT ゲートの場合、入力の状態と異なる状態が出力に現れる事を意味しています。H レベルを入力すれば L レベルが出力され、L レベルを入力すれば H レベルが出力されます。この様な動作を（論理）反転動作と呼びます。会話時は「信号を反転する」などと言います。
回路図記号は三角形を使用します。入力または出力線の隅に丸印をつけます。この丸印が信号の論理反転を表します。
NOT ゲートは論理を反転させる事が出来るので注目する論理を変更したいときに使用します。例えば、ある機能が動作しているときに H レベルになる信号 A がある時にある回路で信号 A を使用する事を想定します。動作が止まっている時、すなわち信号 A が L の時に何らかの動作を行う回路を設計する場合、信号 A に NOT ゲートを挿入する事で動作が止まっている時に H レベルになる信号を作ることができます。この様に注目する状態を反転する機能と後述する AND ゲートや OR ゲートと組み合わせて使用する事で様々な条件を検出する回路を設計する事ができます。
ポイント：NOT ゲートは注目する状態を入れ替える事ができる

2.1.2　AND ゲート

図 2.2 に AND ゲートの回路図記号と真理値表を示します。
AND ゲートは全ての入力が H レベルの時のみ H レベルを出力します。他の場合は L レベルを出力します。これは入力の掛け算と考える事もできます。ここでは入力が 2 つで



真理値表

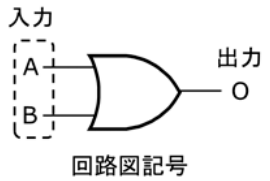
A	B	O
L	L	L
H	L	L
L	H	L
H	H	H

図 2.2 AND ゲートの回路図記号と真理値表

すが入力信号数が 3 以上の AND ゲートもあります。入力が増えても動作は同じです。全ての入力が H レベルの時のみ H レベルを出力します。
補足：3 入力以上の AND ゲートは 2 入力 AND を複数接続する事でも実現できます。
実際の回路では AND ゲートの入力に NOT ゲートを適当に挿入して特定入力パターンを検出するために使用します。例えば、２進数の０を L レベル、１を H レベルと考えて、２桁の２進数信号を２入力 AND ゲートに入力すると入力が３（１０進数表現）の時のみ出力が H レベルになります。この回路は値３を検出する回路と考える事が出来ます。入力に NOT ゲートを適当に挿入する事で他の値を検出する回路を設計する事が出来ます。
ポイント：AND ゲートは条件検出時に使用する

2.1.3 OR ゲート

図 2.3 に OR ゲートの回路図記号と真理値表を示します。



真理値表

A	B	O
L	L	L
H	L	H
L	H	H
H	H	H

図 2.3 OR ゲートの回路図記号と真理値表

OR ゲートはいずれか一つの入力が H レベルの時に H レベルを出力します。OR ゲートも入力信号数が 3 以上の OR ゲートを考える事が出来ます。いずれか一つの入力が H レベルの時に H レベルを出力します。
補足：AND ゲート同様に複数の 2 入力 OR ゲートを複数接続しても作ることができます。
実際の回路では前述の AND ゲートと NOT ゲートを用いて作られる条件検出回路と組み合わせて使用します。例えば、２進数２桁の信号の値３または値１を検出する回路などを作る事が出来ます。AND ゲート節で説明した値３を検出する回路と新たに値１を検出する回路を AND ゲートと NOT ゲートを用いて設計し OR ゲートの入力に接続すると値３または値１を検出する回路となります。
ポイント：OR ゲートは条件を足し算する時に使用する

2.1.4　記憶素子

デジタル回路内で使用される記憶素子の種類は複数あります。ここでは回路設計で重要な役割を担う D タイプ・フリップフロップ（DFF）について説明します。
図 2.4 に DFF の回路図記号と動作説明のためのタイムチャートを示します。

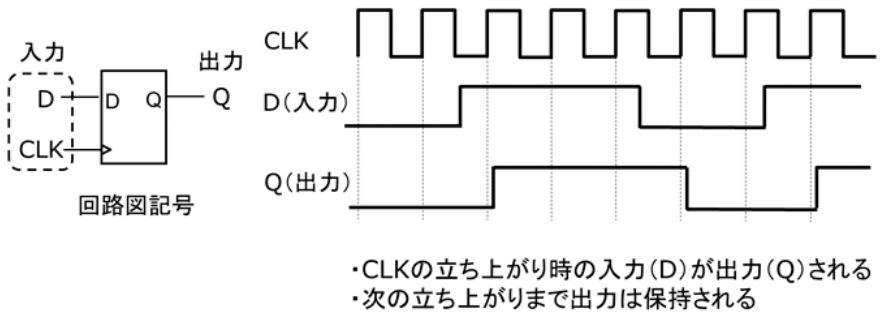


図 2.4　DFF の回路図記号とタイミング・ダイアブラム

前述した他ゲートと比較すると動作が複雑です。DFF の入力は 2 つあります。D 端子と CLK（クロック）入力です。出力は Q 端子です。タイミング・ダイアグラムは時間変化による信号変化を表した図で横軸が時間で左から右へ流れています。縦軸は各信号の電圧変化を表しています。
DFF の動作を説明します。CLK 端子に入力している信号の L レベルから H レベルへの立ち上がり遷移時に D 端子入力の状態を取り込み（記憶して）Q 端子へ出力します。CLK 端子に入力している信号の立ち上がり時のみ動作するので、CLK 信号の立ち上がり時以外に D 端子の信号が変化しても出力 Q 端子の状態を記憶しているので Q 端子の状態は変化しません。この DFF と前節まで説明したゲートを使用する事で全てのデジタル回路を作る事が出来ます。
現時点では 4 つの要素と CPU などの複雑な回路の関係を想像する事は難しいかもしれませんが、学習を進めるに従って様々な事がつながると思います。
ポイント：CLK 端子に入力している信号の立ち上がり遷移時に D 端子入力の状態を取り込み（記憶し）Q 端子へ出力する

2.1.5　回路の種類

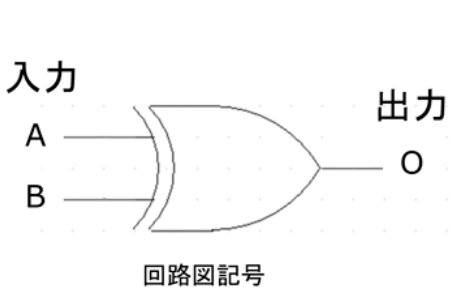
回路は入力と出力信号を持っています。スイッチやディスプレイなど入出力信号どちらかありませんが、これらは動作や視覚として入出力すると考えると入出力双方持っています。回路は与えた入力を回路で処理し希望（設計）の値を出力します。デジタル回路は基本 4 要素を組み合わせで構成されています。膨大な数の要素を用いて回路は構成されますが、あるまとまり毎に回路を分解すると以下 2 種類に分類できます。

- 組み合わせ回路
- 順序回路

組み合わせ回路とは記憶素子を含まない回路であり、順序回路とは記憶素子を含む回路です。次節でこれらについて説明します。

2.2　組み合わせ回路

組み合わせ回路とは記憶素子を含まない回路です。記憶素子がないので過去の事を記憶する事ができません。現在入力されている入力信号の 0, 1 のパターンにより出力の値 0, 1 が一意に決定します。2 入力の XOR (排他的論理和、Exclusive OR) 回路を例に説明します。XOR はデジタル回路で重要な役割を担っているので立派な回路図記号が与えられています。図 2.5 に回路図記号と入出力の関係を表としてあらわした真理値表を示します。



真理値表

A	B	O
L	L	L
H	L	H
L	H	H
H	H	L

図 2.5　XOR の回路図記号と真理値表

真理値表の H と L は電圧の高い状態と低い状態を意味しています。H=1, L=0 と考えてください。真理値表には入力の全ての組み合わせが書かれています。各入力組み合わせに対して出力が対で書かれています。入力信号の組み合わせで出力が一意に決まっていることを確認してください。デジタル回路では 0 でも 1 でもない状態は扱いません。0 または 1 どちらかの状態を必ず取ります。ある入力が 0 でも 1 でも良い場合は、そのどちらに対しても出力を決めます。XOR の真理値表を見ると OR ゲートの動作に似ていますが、2 つの入力が同時に H になった時に出力が H にならず L になっている点が異なります。XOR ももちろん AND, OR, NOT ゲートで表現する事ができます。ここでは、XOR を設計する事で組み合わせ回路の設計方法を説明します。考えずに解答を見てしまうより、一度考えてから解答を見る方が理解しやすいと思いますので AND, OR, NOT ゲートを使って XOR 回路を設計してください。5 分程度考えていただければ十分だと思います。

デジタル回路を回路図記号（論理式）で表現する場合、答えが複数ある事があります。特定の制約を課せば答えは一意になるのですが、通常はそのようなことはしません。これから XOR の一つの回答例を挙げますが、あくまで例です。自分で設計した回路が XOR の動作をするのなら、それは正解です。自信がない場合は論理シミュレータを用いて動作を確認すると勉強になると思います。ここではある設計方法（考え方）を使います。真理値表の 3 行目と 4 行目に注目してください（図 2.6 参照）。

出力 O が H(=1) になっている行に注目します。後の説明を進めやすくするために 3 行目の条件を「条件 1 」、同様に 4 行目を「条件 2 」とします。デジタル回路が取る状態は 0 と 1 の 2 つのみなので、出力 H(=1) になる状態を全て考えることは、同時に出力 L(=0) の状態を考える事と同じ意味になります。従って、出力が H または L のどちらか片方を考えれば必要十分です。通常は 1 の状態、ここでは H 状態を考えます。

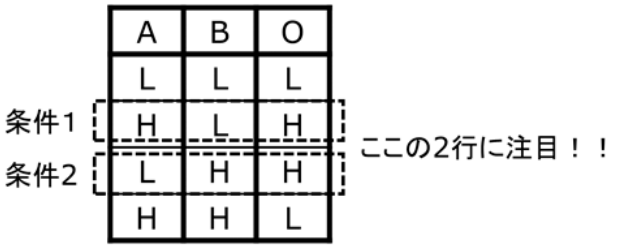


図 2.6　真理値表の注目すべき行

先ずはこの 2 条件を 1 つずつ設計します。条件 1 を見ると A が H で B が L の時のみ出力が H になる回路を設計すればよいことがわかります。ある入力信号の組み合わせの時“のみ”に H を出力するためには AND ゲートを用います。AND ゲートは入力が全て H の時“のみ”出力が H になります。AND ゲートそのままでは条件 1 を満たしません。そこで、片方の AND ゲート入力に NOT ゲートを挿入し、NOT ゲートの入力を入力 B とします。AND ゲートのもう片方の入力は入力 A に接続します。設計した条件 1 を満たす回路を図 2.7 に示します。

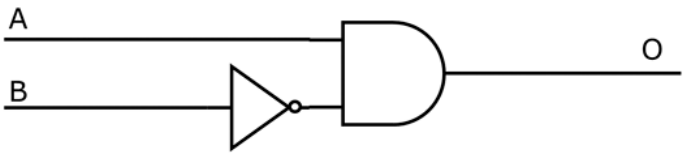


図 2.7　条件 1 を満たす回路

AND ゲートは特定の入力組み合わせの時のみ H を出力する性質があります。特定の入力組み合わせを選択するために AND ゲートの入力に適切に NOT ゲートを挿入します。では条件 2 の回路はどのようになるでしょうか？

出力が H になる入力条件が A,B で入れ替わっていますから NOT ゲートを A 側へ移動させた図 2.8 の回路になります。

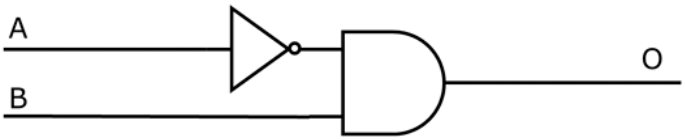


図 2.8　条件 2 を満たす回路

2 つの条件と満たす回路がそれぞれできました。しかし XOR の出力が H になるのは条件 1“または”条件 2 が成立する時です。ここで、“または”が登場しました。これは OR ゲートの動作です。OR ゲートは複数入力のいずれか 1 つが H の時に出力が H になります。条件 1 を満たす回路の出力と条件 2 を満たす回路の出力を OR ゲートの入力に接続する事で条件 1 または条件 2 が成立した時に出力が H になる回路が完成します（図 2.9)。

これが組み合わせ回路の基本的な設計手順です。まとめると下の様になります。

1. 真理値表を書く

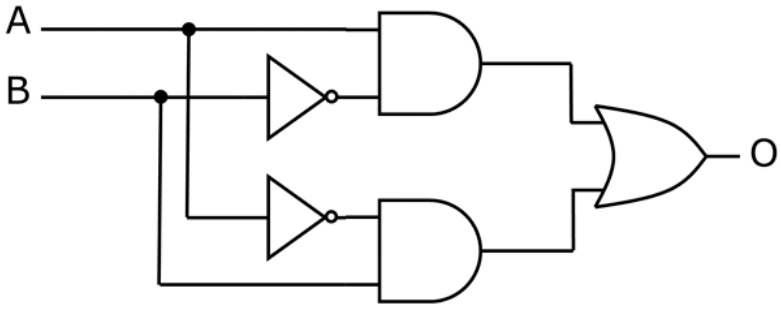


図 2.9　設計した XOR 回路

- 全ての入力状態に対する出力を書くこと

2. 出力が H または 1 の行毎に回路を設計する
   - AND ゲートの入力に適切に NOT ゲートを挿入する事で設計する
3. 行毎に設計した回路の出力を OR ゲートの入力に接続する

この設計方法を採用するなら、真理値表を書き終えた時点で設計が終わっているようなものです。皆さんの中には「本当にこんなやり方で良いの？」と感じている方もいるかもしれません。そのように思う方はデジタル回路について学んだ経験がある方だと思います。デジタル回路の教科書にはブール代数やカルノー図を用いた回路の最適化について必ず書いてあります。最適化とは AND, OR, INT などのゲート数を最小にする事です。2 進演算子の個数を最小にすると言ってもよいかと思います。コンピュータ上で動作する設計ツールが使えない時代やチップの性能が低かった時代は最適化は非常に重要でした。動作性能や使用チップ数（製造コスト）に強い影響を与えていたからです。しかし、現在ではデジタル回路をハードウエア技術言語で記述し、そのコードを開発ツールに入力し回路情報を生成します。開発ツールが非常に強力な最適化を行いますので人間が手で計算して最適化する必要ありません。特に FPGA を使う場合、その動作原理によりゲート数の最適化の影響は小さいです。詳細は「FPGA の構造」で説明します。従って、最適化はツールに任せ、設計者が最も理解しやすい表現方法で記述する事が重要です。

最後に例題として 3 bit の +1 演算回路を設計することで組わせ回路についての理解を深めましょう。

3 bit の +1 演算回路は入力信号 3 本、出力信号 3 本を持っています。ここでは入力を I、出力を R とします。図 2.10 は回路ブロックです。

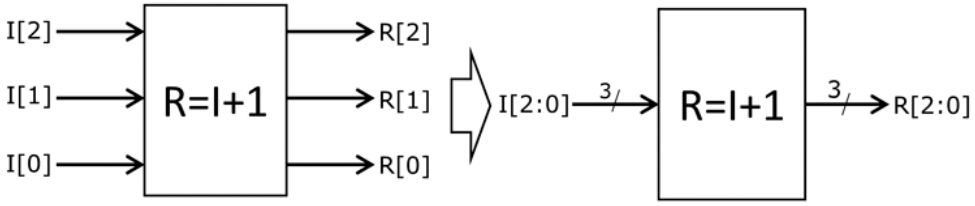


図 2.10　演算回路ブロック

左側の図を見てください。I または R の右の数字は桁番号を表しています。数字が小さい方が下位、大きい方が上位となります。この演算回路の入出力信号のように複数信号で

表 2.1　演算回路の真理値表

I[2]	I[1]	I[0]	R[2]	R[1]	R[0]
L	L	L	L	L	H
L	L	H	L	H	L
L	H	L	L	H	H
L	H	H	H	L	L
H	L	L	H	L	H
H	L	H	H	H	L
H	H	L	H	H	H
H	H	H	L	L	L

一つの意味を表す信号の場合、それぞれの桁を分けて図を書くと見難いので通常は右の図のように I[2:0]、R[2:0] などのようにまとめて書きます。右の図の矢印にスラッシュと数字が書かれていますが、これは図としては信号が 1 本だが実際は 3 本の信号線であることを表しています。次にこの回路の真理値表を書いてみましょう。

表 2.1 が設計する演算回路の真理値表です。出力が複数あるのでどのように設計を進めれば良いか躊躇する方もいるかもしれませんが、組み合わせ回路の出力は他の出力の状態に影響を与えませんので出力一つずつ独立に回路を設計します。

先ずは最下位桁出力の R[0] について設計します。真理値表の I[2:0] と R[0] 部分を抜き出して書くと図 2.11 のようになります。

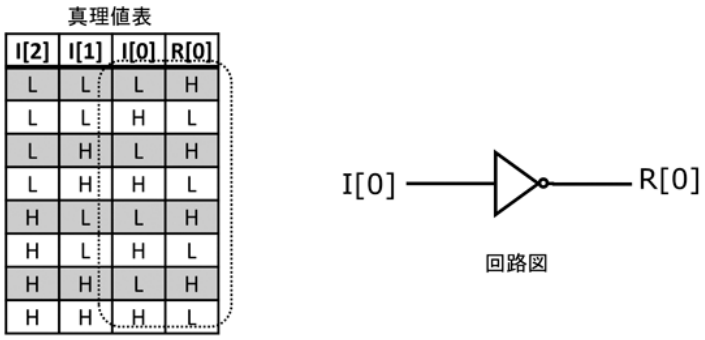


図 2.11　R[0] の真理値表と回路図

XOR を設計した時と同じ方法で設計しても問題ありませんが、ここではデジタル回路のパズル的側面を紹介するために違った設計方法で進めます。真理値表を良く見ると I[0] の反転が R[0] になっていることに気が付きます。従って、R[0] の回路は右の図のように NOT ゲートとなります。

次に R[1] について設計します。真理値表の I[2:0] と R[1] 部分を抜き出して書くと図 2.12 のようになります。R[1] の桁についても I[0] と I[1] について良く見ると、先ほど設計した XOR と全く同じ動作であることが分かります。従って、R[1] の回路は右の図ように XOR ゲートになります。

最後に R[2] ですが、これは少し難しいです。この様な場合は XOR を設計した際の方法を用いると良いですが、ここでは XOR の機能に注目して設計を進めます。図 2.5 の XOR の真理値表を見てください。入力 A を信号入力、入力 B を制御信号とみなして出力の変化をもう一度見直します。制御信号 B が L の時、入力 A はそのまま出力されます。

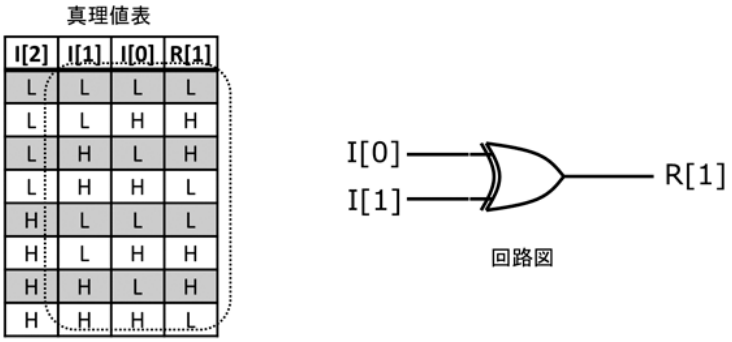


図 2.12　R[1] の真理値表と回路図

ところが制御信号 B が H の時、入力信号 A の反転が出力されます。制御信号 B により入力信号を反転したり、そのまま出力できます。XOR は信号反転制御回路としても動作するのです。この視点で、R[2] の真理値表を見てみましょう。

真理値表の I[2:0] と R[2] 部分を抜き出して書くと図 2.13 のようになります。

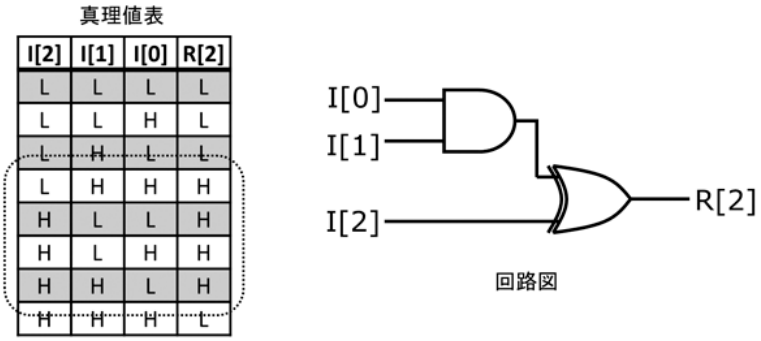


図 2.13　R[2] の真理値表と回路図

R[2]=H となる状態は 4 つありますが、そのうち I[2]=H の時に R[2]=H となる条件は 3 つあります。すなわち、大部分は I[2] をそのまま出力すれば良い事が分かります。ただし、2 か所異なる個所があります。それは両方ともに I[1]=H, I[0]=H の時です。これは I[1]=H, I[0]=H の時のみ I[2] を反転させて出力すれば良い事を意味しています。ここで XOR を信号反転制御回路として使うと目的の回路は右の図となります。

図 2.14 はこれまで設計した 3 つの回路をまとめて書いた回路図です。

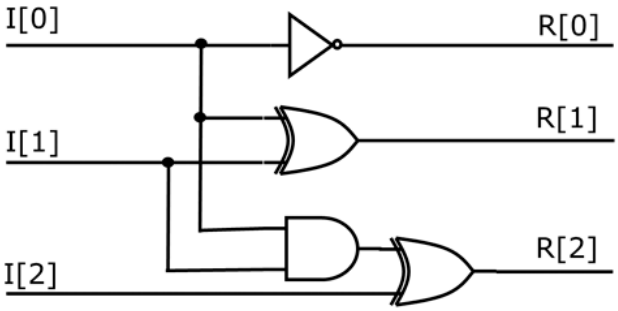


図 2.14　演算器の全回路図

以上で 3bit の +1 演算器が完成しました。

2.3　順序回路

順序回路とは記憶素子を含む回路です。記憶素子があるので過去の事象を記憶する事ができ、その記憶値と現在入力されている信号の組み合わせから出力が決まります。

2.3.1　順序回路の設計手法

順序回路を設計する方法は下の 2 種類あります：

1. 非同期設計
2. 同期設計

どちらも長所短所がありますが、現在は同期設計が主流です。古くは非同期設計が主流でした。現在は FPGA を含むほとんどの回路が同期設計を用いて設計され、FPGA は同期設計用の構造になっています。非同期設計と同期設計の違いを順序回路の代表回路であるカウンタ回路を用いて違いを説明します。設計方法の違いを説明する前にカウンタ回路（通常は略して単にカウンタと呼びます）の動作について説明します。

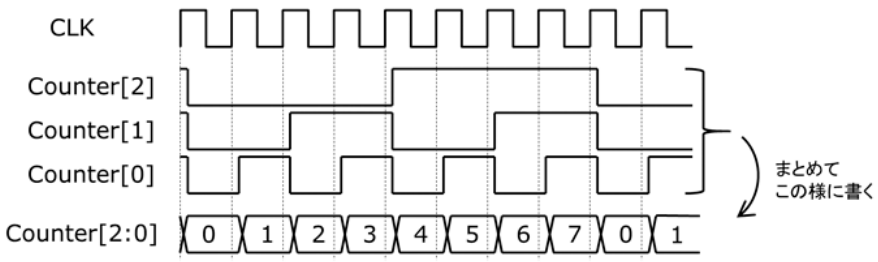


図 2.15　カウンタのタイミング・チャート

カウンタはクロックと呼ばれる入力信号と複数本の出力信号を持っています。出力をまとめて一つの値と見た場合、入力信号の立ち上がりエッヂ毎に値が 1 ずつ増加する回路です。図 2.15 はカウンタ回路のタイミングチャートです。ここでの入力は CLK 信号、出力は 3 bit の Counter 信号です。CLK 信号の立ち上がりエッヂで値が 1 加算されています。

2.3.2　非同期設計

非同期設計で設計されたカウンタは非同期カウンタと呼ばれます。図 2 が非同期カウンタの回路図です。

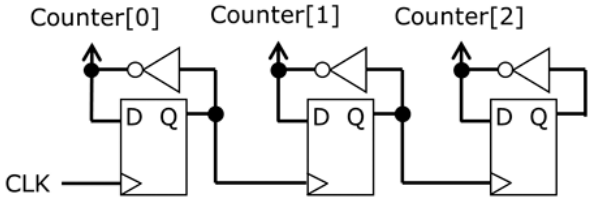


図 2.16　非同期カウンタの回路図

回路動作を考える前に非同期回路の回路を見た目だけで単純な回路だと感じませんか。上手に非同期設計すると回路が単純になる傾向があります。出力される値の桁毎に DFF と NOT ゲートが構成された回路で全体が構成されています。ここで図 2.15 に戻り各出力信号の波形を見てください。CLK の 2 倍の周期で最下位桁の Counter[0] が変化し、その上の桁はさらに 2 倍の周期で変化していることが分かります。この事を踏まえて改めて動作を見てみましょう。

図 2.16 を見てください。CLK の立ち上がりエッヂにより左の DFF の出力の反転値が Couonter[0] として出力されます。次の CLK の立ち上がりエッヂでも自分自身の出力の反転値が出力されます。図 1 のタイミングチャートの最下位桁の動作になることが確認できます。先に確認したように上の桁は下の桁の 2 倍の周期になっていますので、上位桁を生成する中央の DFF 入力に左の DFF の出力を接続し、最下位桁と同じ回路とすれば 2 倍の周期を生成できます。最上位桁も同じです。桁数が増えても同じです。とても分かりやすい回路です。
ところが、このような非同期設計はあまり使われません。何故でしょうか？
非同期設計の長所は以下です：

1. 回路規模（ゲート数）を小さくできる
2. 高速動作
3. 低消費電力

一般的に非同期設計で回路を設計すると回路規模を小さくすることが可能です。回路規模（ゲート数）を小さくできるので高速化される傾向があります。出力を変化させたい時のみ変化する回路のみ動作させることができるので同期回路と比較すると消費電力を削減できます。これらの長所は全て魅力的です。しかし、現在では使されることは稀です。それは、同期設計と比較して設計が難しくなる欠点があるからです。図 2.17 の非同期カウンタのタイミング・チャートを見てください。

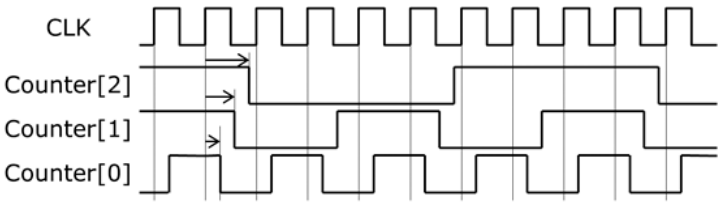


図 2.17 非同期カウンタのタイミング・チャート

図中の矢印に注目してください。非同期カウンタはその動作原理から、CLK の立ち上がりエッヂにより Counter[0] が変化、その変化により Counter[1] が変化、Counter[2] が変化、とドミノ倒しのように変化が伝搬することで動作します。ですから、上位の桁ほど入力信号 CLK から変化するまでの時間が長くなります。図はかなり大げさに書いてありますが、このタイミングチャートの回路はカウンタ機能、数を 1 つずつ加算する機能として使用するためには 3bit が上限のようです。4bit へ拡張すると拡張された Counter[3] の変化が Counter[0] の変化と同じになるか超えてしまうからです。Counter 信号をまとめて値として見た時に 0, 1, 2,…と正しく数字が加算されなくなってしまいます。 非同期設計を採用する場合、このような遅延を全て考慮して設計しなければいけません。言い換え

ると、非同期回路の動作は回路の論理動作に加えて遅延により動作が決定します。現在の回路は DFF の数が少なくとも 100 個程度、多ければ数千個になり、その間のゲートの遅延を考慮して設計する事はとても大変な事です。さらに遅延は様々な原因により変化します。例えば、ゲート間の配線長、温度、素子特性のばらつき、クロックジッタ、など色々あります。これらを全て考慮して設計することは簡単ではありません。回路規模が小さければ設計できますが、複雑な機能を搭載する場合、限られた時間内で設計する事がとても難しくなり現実的に設計不可能になります。
問題点を改めて書くと、回路の動作が論理動作だけでは決まらず、遅延により動作が変わることです。この問題を解決するために登場した設計手法が同期設計です。

2.3.3　同期設計

同期設計は一つのルールに従って設計する事で非同期設計の問題点である遅延を考慮する事を解決し設計を容易にします。
そのルールとは全ての DFF の CLK 入力にクロックと呼ばれる周期的に変化する信号を入力する事を設計の出発点に設定します（図 2.18 を参照）。

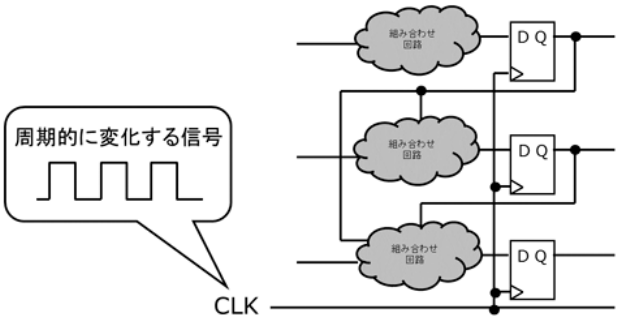


図 2.18　同期設計のルール

このルールに従って設計する事で非同期設計の問題である遅延について考慮する必要が実質なくなるのです。もちろん、実際はまったく遅延について考えなくて良いわけではありません。しかし、論理動作設計には考える必要が無くなります。同期カウンタを例に説明します。

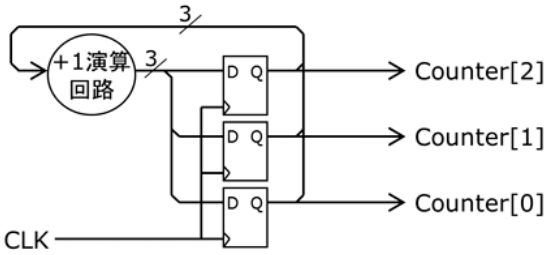


図 2.19　同期カウンタの概念図

図 2.19 は同期設計によるカウンタの概念図です。“+1 演算回路”は「組み合わせ回路」節で設計した回路です。この演算回路は入出力をそれぞれ 3 本持ち、3 本一組で一つの

値を表しています。入力値に 1 を加えた値を出力します。桁数が 3 桁に限られているため 7 を入力した場合、出力は 0 になります。常に下 3 桁が出力されると考えてください。DFF が出力している値に 1 を加えた値が常に DFF に入力されていますので、CLK 信号の立ち上がりエッヂで値が 1 つずつ加算するカウンタ機能が実現できています。図 2.19 を見ると順序回路は組み合わせ回路と DFF により構成されていることが分かります。
さて、この設計方法で遅延問題が解決される理由を説明します。図 2.20 が同期カウンタのタイミング・チャートです。

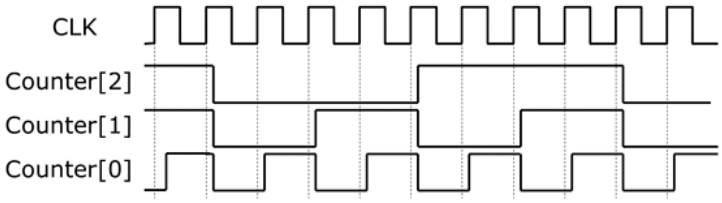


図 2.20 同期カウンタのタイミング・チャート

先ほどの非同期カウンタと何かが違いますね。CLK の立ち上がりエッヂから Counter[2:0] のそれぞれの出力が変化するまでの時間がほぼ同じです。何故なら Counter[2:0] は DFF の出力であり、その DFF の CLK 端子には共通のクロックが入力されているからです。同期回路が正しく動作する条件は CLK の立ち上がりエッヂから DFF の入力が変化するまでの時間がクロック周期より短い事です。この条件とここでは同期設計条件と呼ぶ事にします。図 2.19 では CLK の立ち上がりエッヂから DFF の出力が変化し、その信号が +1 演算回路へ伝搬し、通り抜けて DFF の入力に到達するまでの時間がクロック周期よりも短ければ正しく動作します（厳密には DFF 入力は CLK の立ち上がりエッヂ前の一定期間安定している必要があります)。正しく動作する条件を満たしていると仮定すると同期回路は論理動作のみで回路動作が決まります。遅延の影響を受けません。非同期回路の問題点を解決しました。

実際の開発では同期設計条件を満たさなくなることがあります。その場合、条件を満たさない部分は他の回路に影響を与えないので問題個所のみ修正することで開発を進める事ができます。これは実用上非常に重要な長所です。非同期回路では遅延が積み重なるので問題発生個所の出力が接続される全ての部分が影響を受けるので修正範囲が広くなります。

同期設計において遅延計算は同期設計条件を満たすかどうかの判定のためだけに限られます。遅延では回路動作は決まりません。同期設計条件を満たすか否かはコンピュータツールで簡単に計算できます。全ての DFF の出力から入力までの遅延時間を全て計算し、その値がクロックの周期よりも小さい事を調べれば良いからです。もし、条件を満たさない部分を見つかったら、その経路をレポートとして出力します。設計者はその問題個所の回路を条件を満たすように修正します。

少し複雑になったのでまとめます。非同期回路は回路動作が論理動作と遅延の両方で決まるので、常に双方を考慮しながら設計を進めなければいけない。同期設計は遅延を忘れて論理動作のみ設計すればよい。ただし、設計後に同期回路動作するか否かを計算して確認する必要がある。

2.4　FPGA の構造

FPGA は回路を書き換えることが可能な構造を持っています。ここでは、回路変更可能な構造、仕組みを説明します。

2.4.1　FPGA の基本的概念

前節でデジタル回路は組み合わせ回路と順序回路に分かることを説明しました。順序回路は組み合わせ回路に記憶素子を組み合わせた回路である、ということを思い出してください。FPGA の基本的な考え方は、自由に変更可能な組み合わせ回路と記憶素子を多数並べ、それらを自由に配線する仕組みを作ることです。以下で自由に変更可能な組み合わせ回路を作るためにはどのような仕組みが必要なのか説明し、その後でそれらを自由に配線できるようにするためにはどのような仕組みが必要なのか説明します。

2.4.2　メモリー

メモリーについての知識が必要なので簡単に説明します。メモリーについて既に知ってい入る方はこの節を飛ばして次節へ進んでください。

メモリーとは複数値の記憶を目的に設計された回路です。ここでは概念的な動作を理解することが目的なので記憶素子として既に理解している DFF を使って説明します。メモリーは、例えば 16 個、1024 個など複数の DFF で構成されています。通常は 2 の N 乗個で構成されます。PC の DRAM という名前について聞いたことがある人がいるかもしれません。DRAM もメモリーの一種です。メモリーは各 DFF に値を書き込んだり、各 DFF からデータを読み出すことができます。ただし、同時に複数の DFF の値を読み書きすることはできません。各 DFF にはアドレスと名付けられた番地が 0, 1, 2,.... と与えられており、読み書きするアドレスを指定するための複数の信号線を用いてアドレスを指定します。メモリー内部に設けられた複数の DFF 入出力を切り替える回路によりアドレスで指定された DFF の入出力がメモリーの外部と接続される構造になっています。この仕組みによりアドレス信号で指定された DFF に対して読み書きすることができるようになります。一度書き込んだ DFF のデータは次の書き込みが行われるまで保持されるようになっています。このように多数のデータ記憶を目的とした素子をメモリーと呼んでいます[*1] 。

2.4.3　変更可能な組み合わせ回路

組み合わせ回路の動作は真理値表で表現することができることを思い出してください。例えば、入力信号 I[3:0]、出力信号 O を持つ 4 入力 AND ゲートについて考えます。真理値表を表 2.2 に示します。

この真理値表の動作を 4 入力 1 出力の書き換え可能メモリーを使って実現することができます。入力をメモリーのアドレスとみて真理値表の該当する出力にそれぞれ O,1 を書き込めばよいのです。ここでの例である 4 入力 AND を実現する場合、アドレス 15(0xF)

[*1] 実際のメモリーでは DFF は使用されていません。記憶に特化した回路を使用します

表 2.2 4 入力 AND ゲート

I[3]	I[2]	I[1]	I[0]	O
L	L	L	L	L
L	L	L	H	L
L	L	H	L	L
L	L	H	H	L
L	H	L	L	L
L	H	L	H	L
L	H	H	L	L
L	H	H	H	L
H	L	L	L	L
H	L	L	H	L
H	L	H	L	L
H	L	H	H	L
H	H	L	L	L
H	H	L	H	L
H	H	H	L	L
H	H	H	H	H

のみ 1 を書き込み、他は 0 を書き込めば良いのです。書き込み後、アドレスを入力だと思い、色々な値を入れて動作させると確かに 4 入力 AND と同じ動作する事が分かります。この方法を用いることで全ての 4 入力組み合わせ回路を表現可能です。なぜなら 4 入力の状態は 16 であり、それぞれの入力パターンに対応する出力値が設定されているからです。メモリーが心理帳票そのものになっていることに注意してください。このメモリーは FPGA ユーザーから見ると記憶素子として使用しないので混同しないように Look Up Table(LUT) と名付けられています。

2.4.4 変更可能な順序回路

LUT を使用する事で組み合わせ回路を表現できることが分かりました。ここでは、順序回路をどのように実現するか説明します。
順序回路は組み合わせ回路に記憶素子を追加した回路です。そこで、図 2.21 の回路を考えます。

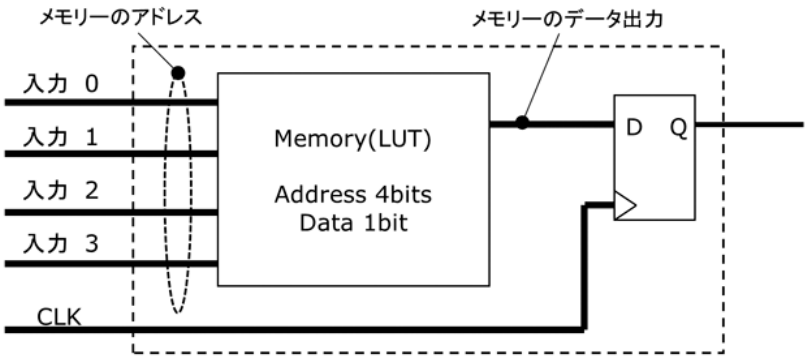


図 2.21 FPGA 内の順序回路

図中では LUT 出力が直接 DFF 入力に接続されていますが、ここに DFF をバイパスす

る経路を設けて設定データで ON/OFF できるスイッチで切り替えできるようにすれば順序回路または組み合わせ回路を表現することができます。これが FPGA の基本的な考え方です。この回路が FPGA を構成する基本単位となり、Xilinx 社では Slice、ALTERA 社では LE(Logic element) と名付けられています。基本単位を数多く並べ、お互いをスイッチにより自由に結線できる構造を持つデバイスが FPGA です。

2.4.5　FPGA の構造

前節で FPGA の基本単位は LUT と DFF であることを説明しました。FPGA は基本単位を下の図のように多数配置した構造を持っています。

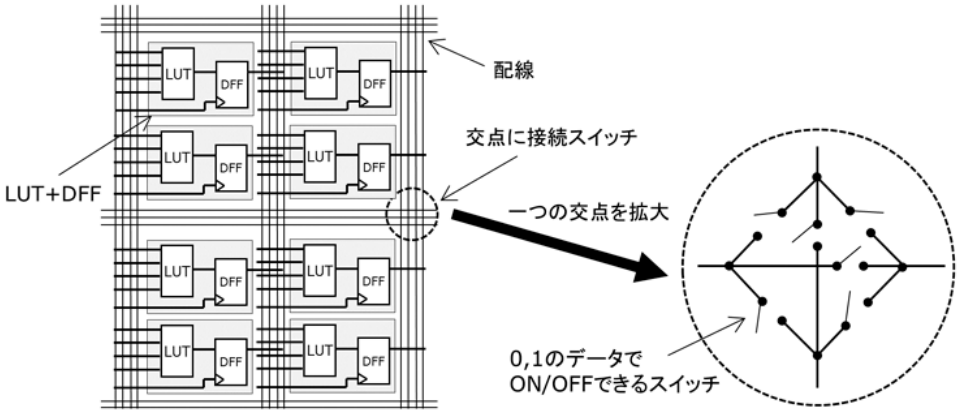


図 2.22　FPGA の構造

基本単位を囲むように予め縦横に配線が用意されています。この配線と基本単位の入出力は 0,1 のデータで ON/OFF できるスイッチで接続/切断ができます。また、配線が交差する点には配線の縦横の方向を変更するためのスイッチがあります。これらを組み合わせることで複数の基本単位を接続する多入出力回路を実現することができます。

FPGA にデータを書き込む、ダウンロードするとは FPGA 内の LUT とスイッチのデータを書き込むことです。このデータを変更することで FPGA は何度でも回路変更することが可能になっているのです。

最後に幾つか注目して欲しいことを書きます。最初は論理ゲートの最適化についてです。組み合わせ回路の節で回路の最適化はツールにより行うのでユーザーは気にする必要がないと書きましたが、FPGA の場合は最適化をユーザーが行うことはほとんど無意味であることが分かります。なぜなら、FPGA 内の組み合わ回路部分は論理ゲートではなく真理値表で回路を表現するので組み合わせ回路の入出力数と動作論理が決まれば使用する LUT がほぼ決まってしまうからです。このことからも FPGA を使用する際の設計では設計者が分かりやすい記述を優先することが重要であることが分かります。

次は FPGA の配線リソースについてです。FPGA の予め用意された配線には限りがあります。配線を使い切ってしまえは基本単位が余っていても使うことができません。このことから、FPGA の回路を設計する際は小さくまとまった回路を用いて構成する方が有利です。一つのまとまりが大きな回路では各基本単位間を行き来する信号数が増えてしまうため配線を使いきってしまう事があります。また配線長が長くなる傾向があります。配

線長が長くなると信号が行き来する時間が長くなりますので回路を高速にどうさせることが難しくなります。FPGA では小さな回路を用いて構成することで行き来する信号数が少なりまとまり毎に独立配置することができるので配線リソースと動作速度の点から有利になります。

最後にクロック信号の配線についてです。FPGA は同期設計で回路設計することが前提の構造になっています。同期設計の節で説明したように全ての記憶素子のクロック入力にクロック信号が入力され、全ての記憶素子の入力でクロック位相がそろっていなければいけません。信号が伝搬するためには時間が必要です。FPGA のクロック入力端子から各基本単位までの距離にかかわらず位相をそろえるのは簡単ではありません。そこで、FPGA にはクロック信号を全ての基本単位へ同時に分配するクロック信号専用配線が設けられています。これは同時にクロックが到着するよう特別な工夫がされた配線です。このような特別な配線の数は多くても 20 本程度までです。従って、複数のクロック信号を使うときは上限を超えないように注意してください。また、このような特別な配線を駆動するためには特別な配線とチップ内のバッファ（増幅器）を使用しなければいけません。複雑なクロック構造を持つ回路を設計する際は FPGA のデータシートなどを十分に理解し、何がどの程度までできるのか理解してから設計を始めてください。