卒業論文

FPGA を用いたトリガー回路の開発

神戸大学理学部物理学科
粒子物理研究室 ATLAS グループ
0593118s　徳永香

平成 21 年 3 月 13 日

概要

ATLAS 実験は 2009 年秋、スイスのジュネーブ郊外にある CERN（欧州素粒子原子核研究機構）で実験再開予定である。ATLAS 検出器は LHC（Large Hadron Collider:大型ハドロン衝突型加速器）の衝突点に設置されている。この加速器は陽子陽子衝突型である。重心系で 14TeV の世界最高のエネルギーを実現する。この ATLAS 実験では Higgs 粒子や超対称性粒子の発見などを目指している。

神戸大では ATLAS 実験において、主に LVL1 トリガーに組み込まれる前後方ミューオントリガーの構築に取り組んできた。このうち、最後段に位置する Sector Logic（SL）では FPGA（Field Programmable Gate Allay）を使用している。FPGA は書き換え可能な論理素子である。今回はその FPGA を用いて 7seg-LED にトリガーレートを表示する回路の作成を行なった。

目 次

図 目 次

表 目 次

1 Introduction

1.1 LHC

LHC(Large Hadron Collider) はスイス・ジュネーブ郊外の CERN 研究所にある現在稼働中の陽子陽子衝突型加速器である。LHC は 2000 年まで使用されていた LEP(Large Electron Positron Collider) の地下 100m のトンネル内に建設されており、周長はおよそ 27km である。

LHC の全体像を図 1 に載せる。

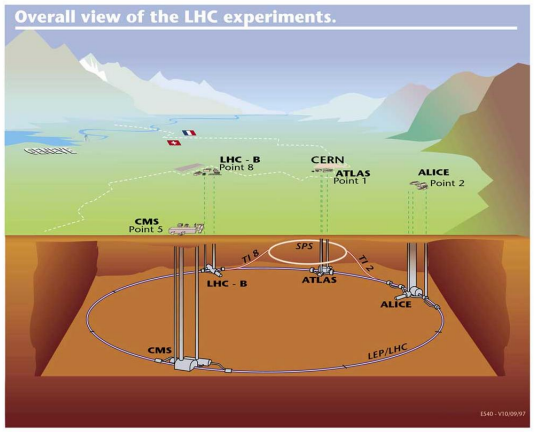

図 1: LHC 全体像

LHC の最大の特徴は 14TeV の世界最高の重心系エネルギーである。陽子を 7TeV まで加速させ、正面衝突させている。LHC はハドロンコライダーであり、円形加速器でもシンクロトロン放射によるエネルギー損失が少ないため、電子より高いエネルギーまで加速できる。そのため重心系で 14TeV を実現できる。これにより、Higgs 粒子の探索、超対称性粒子や未知の相互作用などにおいて、TeV 領域の物理の発見が期待される。

一方、陽子は電子と違って内部構造を持ち、その構成粒子であるクォークやグルーオン間の反応断面積は数 mb と非常に大きく、莫大な量のバックグラウンドが予想されている。そのため、物理現象を解析するために、必要なデータを効率よく正確に収集することが重要である。

LHC には 4 つの衝突点がある。それぞれ、大型汎用検出器 ATLAS(A Toroidal LHC AppratuS)、ATLAS より小型の汎用検出器である CMS(the Compact Muon Solenoid)、重イオン衝突実験用検出器の ALICE(A Large Ion Collider Experiment)、B-Physics に特化した検出器 LHC-B が設置される。

1.2 ATLAS

ATLAS 検出器は、直径 22m、長さ 44m の円筒形で、総重量は 7000t という巨大な検出器である。構成は衝突点に近いほうから、内部飛跡検出器、電磁カロリメータ、ハドロン

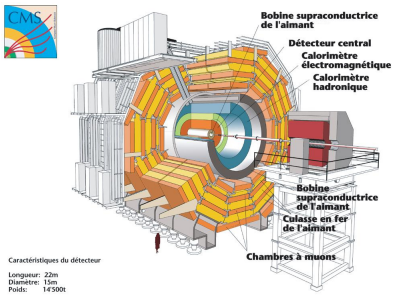

図 2: CMS

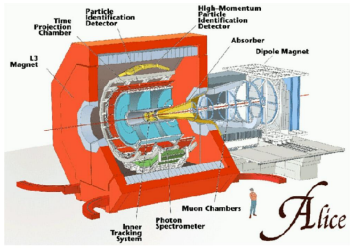

図 3: ALICE

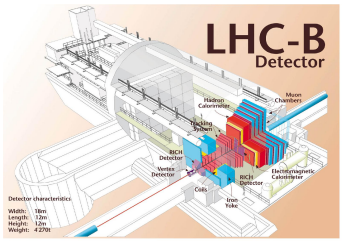

図 4: LHC-B

カロリメータ、ミューオン検出器である。

ATLAS 検出器における座標系は、ビーム軸を z 軸にとり、これに垂直な方向を r 方向、ビーム軸周りを周回する方向にφ方向をとった、円筒座標系を採用している。

ATLAS 検出器は大きく分けて、Barrel と Endcap に分けられる。Barrel は円筒の筒に相当する領域であり、Endcap は円筒の蓋に相当する領域である。さらに、Endcap は円筒の蓋の円の中心付近を Forward と呼び、それより外側を Endcap と呼ぶこともある。

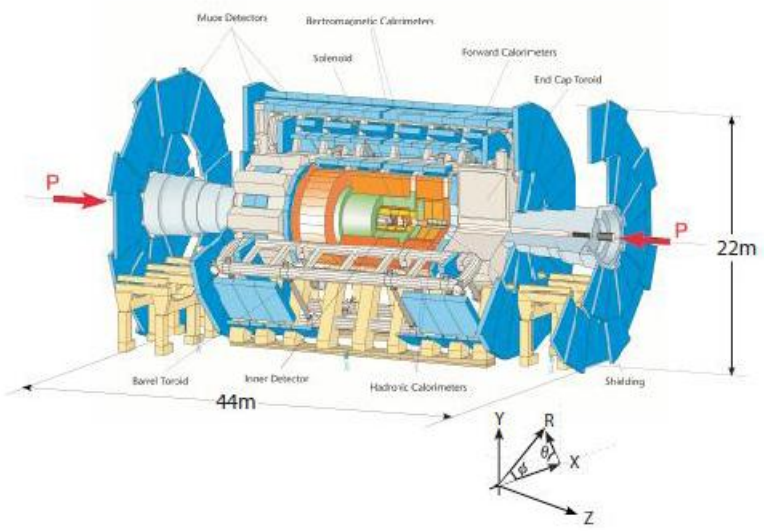

図 5: ATLAS 検出器

1.2.1 ATLAS 実験のトリガーシステム

ATLAS 実験のトリガーシステムは、LVL1、LVL2、EF(Event Filter) の 3 段階のトリガーを設け、段階的にレートを落としている。データは各検出器からの情報をもとに、各

システムで段階的に処理され、最終的に興味のあるイベントのみが記憶装置に保存される。

1.2.2 TGC システム

TGC(Thin Gap Chamber) は LVL1 におけるトリガー用ミューオン検出器の 1 つである。ATLAS 実験において、TGC システムは LVL1 ミューオントリガーシステムとしてのトリガー判定の役割と、ミューオンの R, φ方向の座標の測定という役割を担っている。TGC エレクトロニクスシステムのトリガーのデータは SL(Sector Logic) に最終的に集められる。

1.2.3 Sector Logic

SL(Sector Logic) は TGC システムの最終段に位置し、トリガーのデータを最終的に集めている。

SL は閾値の変更や、コミッショニング時にイレギュラーな使い方にもできるだけ対応できるように、これらの回路を FPGA(Field Programmable Gate Allay) や CPLD(Complex Programmable Logic Device) といった、内部回路を変更することができる IC によって構成される。これらの設計は HDL(ハードウェア記述言語) によって行なわれている。

2 FPGA(Field Programmable Gate Alley)

2.1 PLD(Programmable Logic Device)

PLD とは、機能をユーザがプログラムできる LSI である。目的のデザインや要求機能に応じてプログラムすることができるようになっている。PLD には大きく分けて CPLD(Complex PLD) と FPGA(Field Programmable Gate Alley) の 2 つがある。

CPLD は、複数の PLD ブロックとそれらを接続するためのひとまとまりの配線領域で構成される。PLD ブロックは、マクロ・セルと呼ばれる AND-OR ゲート、D 型フリップフロップ、I/O ピンで構成される。1 つのマクロ・セルはある程度まとまった機能を実現することが可能である。また、CPLD は EEPROM(Electrically Erasable and Programmable Read Only Memory) のアーキテクチャで実装され、書き込んだ回路データは電源を切っても保持される。

FPGA は、多数の論理ブロックと縦横方向に張り巡らされた配線領域で構成される。論理ブロックは CPLD のマクロ・セルと同じようなものであるが、規模がより小さい点で大きく異なっている。そのため、ひとまとまりの機能を実現するために、多くの論理ブロックを必要とする。また CPLD と異なり、FPGA は SRAM のアーキテクチャで実装されることが多く、書き込んだ回路データは電源を切ると消去される。そのため、電源起動ごとにコンフィギュレーション (書き込み) する必要がある。

最近は FPGA が普及し、先端のプロセスで製造されるようになったため、高速化と低価格化を両立できるようになった。回路規模が小さい場合や電源 ON と同時に動作しなくてはならない回路では CPLD が用いられ、それ以外ではだいたい FPGA が用いられている状況である。

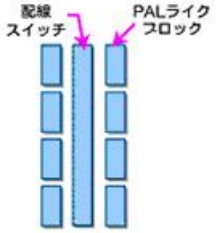

図 6: CPLD

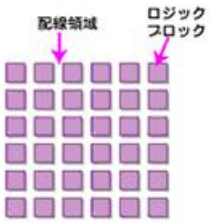

図 7: FPGA

2.2 FPGA

FPGA は書き換え可能な論理素子である。ハードウェアであるため、高速に動作し低消費電力であるという特徴をもつ。内部構造が SRAM(Static Random Access Memory) になっているものが多い。また前述したように、FPGA は SRAM のアーキテクチャで実装されることが多いので、いったん電源を切ると書き込んだ回路データは失われてしまうという欠点ももつ。

最近の FPGA は論理ブロックだけでなく、多くの専用機能も内蔵するようになってい

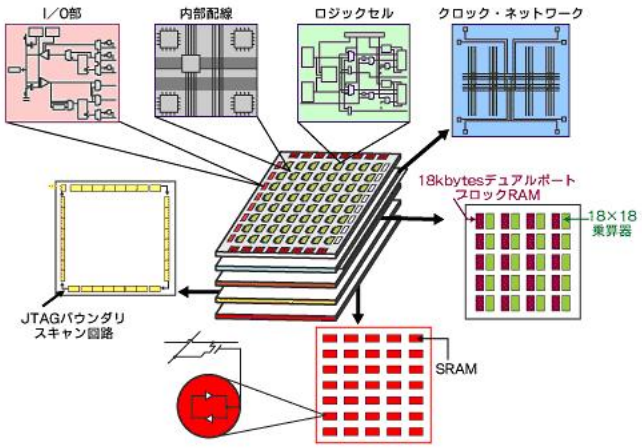

図 8: FPGA アーキテクチャ概念図

る。例えば、メモリ、I/O 回路などである。このように、従来は FPGA の周辺にあった素子たちが FPGA の内部に取り込まれるようになったため、近年 FPGA は論理システムへの適用以外にも、その用途が拡大している。

この取り込みは、より高度なレベルにも及んでおり、組み込みプロセッサの実装もされている。現在、各 FPGA ベンダが提供するプロセッサや汎用プロセッサが FPGA に実装されており、その上で Linux などの汎用 OS などがサポートされている。FPGA に汎用プロセッサを搭載することにより、大きなシステム内でも簡易なサブシステムや、FPGA のみのシステム構築が可能になる。

2.2.1 論理設計

FPGA の論理設計は VHDL（VHSIC HDL）または Verilog HDL などの HDL（Hardware Description Language）で設計するのが一般的である。

2.3 論理回路

論理回路とは、デジタル信号によって論理演算を行なうデジタル回路である。

2.3.1 論理演算

論理演算はブール演算とも呼ばれ、1(真) か 0(偽) かの 2 通りの入力値に対して、1 つの値を出力する演算である。論理演算で基本となるものに、論理積 (AND)、論理和 (OR)、否定 (NOT) といわれる 3 つの演算がある。論理積 (AND) は入力値がどちらも真のとき出力値が真となるものである。論理和 (OR) は入力値のどちらかが真のとき出力値が真となるものである。否定 (NOT) は入力値の値を反転して出力するものである。これらを用い

ている回路をそれぞれ AND 回路、OR 回路、NOT 回路という。一般に使用されているデジタル回路は、この 3 つの回路を組み合わせることによって様々な機能を実現している。

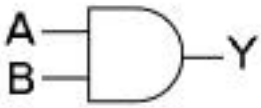

図 9: AND 回路

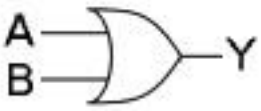

図 10: OR 回路

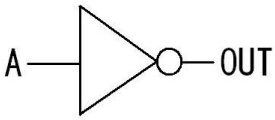

図 11: NOT 回路

入力		出力
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

表 1: AND 回路の真理値表

入力		出力
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

表 2: OR 回路の真理値表

2.3.2 正論理・負論理

デジタル回路には電圧が高い状態”H”(High) と低い状態”L”(Low) の 2 つの状態がある。そして、それぞれの状態に”1”と”0”を対応させる。”1”は信号がある状態、”0”は信号がない状態を表す。”H”に”1”を、”L”に”0”を対応させることを正論理といい、反対に”H”に”0”を、”L”に”1”を対応させることを負論理という。つまり、信号があるとき電圧が高くなることを正論理といい、信号があるとき電圧が低くなることを負論理という。多くのデジタル回路の場合、正論理と負論理が混在しているが、これらは NOT 回路を用いることによって変換することができる。

2.3.3 フリップフロップ

デジタル回路の中で、信号を一時的に保持する回路のことをフリップフロップという。フリップフロップの中にも、RS フリップフロップ、D フリップフロップ、JK フリップフロップ、T フリップフロップと様々なものがある。フリップフロップには、クロック信号の入力によらない非同期式フリップフロップと、クロック信号の入力による同期式フリップフロップがある。同期式フリップフロップはクロック信号の立ち上がり (立ち下り) に同期して、その時点での入力に対応する信号を出力する。この出力は次にクロックが立ち上がる (立ち下がる) まで更新されない。

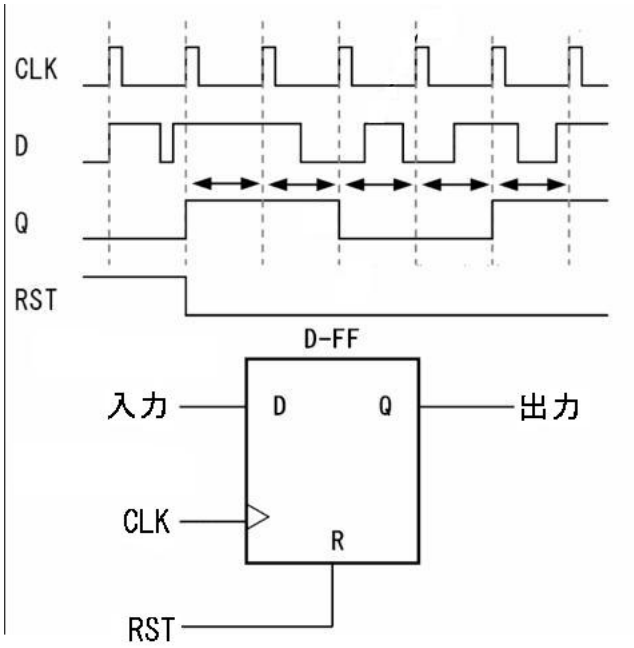

図 12: D フリップフロップの動作

2.3.4 カウンタ

カウンタは数を数えるためのものである。そして一定の周期で元の状態に戻るものである。最も簡単なカウンタは、フリップフロップ 1 個で作ることができる。これは、クロックが 2 回入るともとの状態に戻るので、2 進数の 1 桁の動作をしていることから 2 進カウンタと呼ばれる。フリップフロップを接続していくことにより、簡単に 2 の n 乗カウンタが構成できる。カウンタはフリップフロップで構成されるため、非同期式と同期式がある。各フリップフロップの出力が、クロック入力と同時に変化するように構成されたカウンタを、同期式カウンタという。

2.3.5 レジスタ

レジスタとは、プロセッサが内部に保持する一時記憶装置である。メインメモリを読み書きする際のアドレスや計算結果などを保持することができる。

2.4 SUZAKU

今回は Xilinx 社の FPGA（Spartan3）が搭載されている学習用キット「SUZAKU」を使用した。SUZAKU は FPGA 上にプロセッサと周辺ペリフェラルコアを構成し、オペレーティングシステムとして Linux を採用している。CPU として低コストで資産継承性の高いソフトプロセッサの MicroBlaze を採用している。

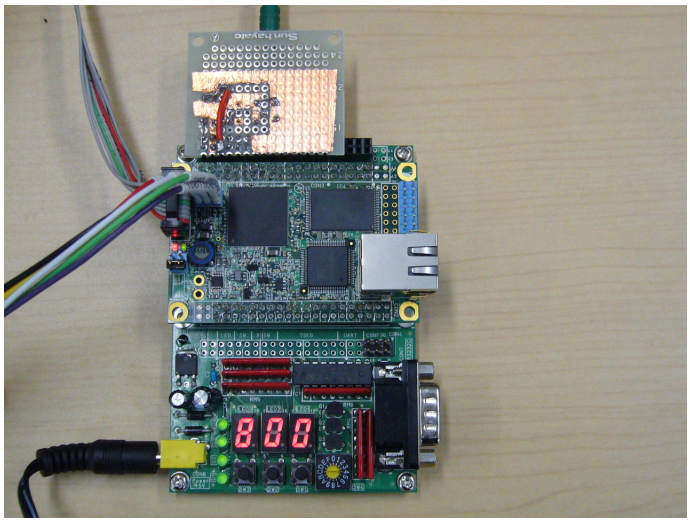

図 13: SUZAKU 写真

2.4.1 全体ブロック図

- プロセッサ
  FPGA 内部で MicroBlaze を使用している。
- バス
  3 種類のバスで構成されている。
  - FPGA 内部 LMB(Local Memory Bus)
    MicroBlaze と BRAM(FPGA 内部メモリ) を接続する専用バス。
  - FPGA 内部 OPB(On-Chip Peripheral Bus)
    複数のペリフェラル IP コアを接続するバス。
  - FPGA 外部バス
    OPB EMC(External Memory Controller) および OPB SDRAM(Synchronous DRAM) を介し、外部メモリデバイスなどを接続するバス。
- メモリ
  3 種類のメモリで構成されている。
  - FPGA 内部 BRAM
    ブートプログラム用として使用している。
  - FPGA 外部フラッシュメモリ
    ブートローダ Hermit や Linux,FPGA コンフィギュレーションデータなどの保存に使用している。
  - FPGA 外部 SDRAM
    Linux のメインメモリとして使用している。OPB SDRAM を使用し、OPB と接続している。

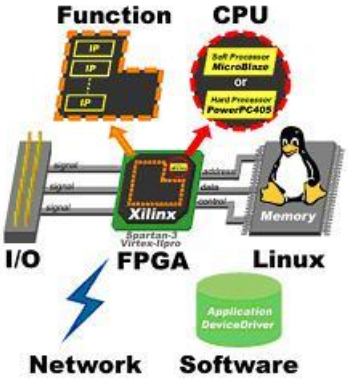

図 14: SUZAKU とは

モデル	SZ030
FPGA デバイス	Xilinx Spartan-3 (XC3S1000)
CPU コア	MicroBlaze
CPU クロック	51.6096MHz
水晶発振器周波数	3.6864MHz
DRAM	16MB
フラッシュメモリ	4MB
Ethernet	10BASE-T/100BASE-TX
拡張 I/O ピン	86
シリアル	1ch(UART : 115.2kbps)
タイマ	2ch
コンフィギュレーション	TE7720
基板サイズ	72×47mm
電源	電圧:+3.3V±3%
リセット機能	ソフトウェアリセット
標準 OS	μClinux

図 15: SUZAKU の仕様

- シリアルコンソール
  OS 用シリアルコンソールに OPB UART Lite を使用している。
- LAN
  LAN コントローラに 10/100Mbps Ethernet コントローラ (LAN91C111) を実装している。
- FPGA コンフィギュレーション
  FPGA コンフィギュレーション用として、フラッシュメモリ (TE7720) を実装している。

2.4.2 LED/SW ボードについて

LED/SW ボードには単色 LED が 4 つ（D1、D2、D3、D4)、押しボタンスイッチが 3 つ（SW1、SW2、SW3)、ロータリコードスイッチが 1 つ（SW4)、7 セグメント LED が

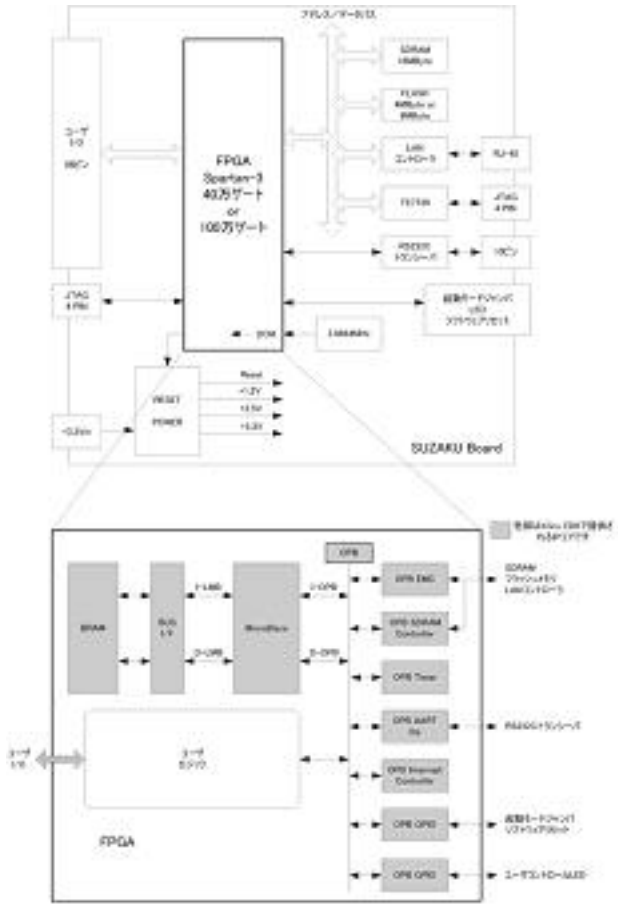

図 16: SZ030 の全体ブロック図

３つ（LED1、LED2、LED3）シリアルポートが 1 つ実装されている。安定した＋ 3.3V を得るため、AC アダプタ 5 V から 3 端子レギュレータで＋ 3.3V を作っている。

2.4.3 ISE

FPGA 側から SUZAKU の開発をするために、ISE（Integrated Software Environment）を用いる。ISE は Xilinx 社が提供する FPGA の統合型設計環境である。GUI 統合ツール Project Navigator で、FPGA に必要な論理合成、配置配線、bit ファイルの書き込みのツールなど、トータルな環境開発を提供している。

- プロジェクトの新規作成
  デバイスの選択、ソースファイルの作成・追加などを行なう。
  今回の FPGA デバイスは次のようになっている。

- 論理合成
  プログラムで記述された回路を and や or の実際の論理回路に変換する。

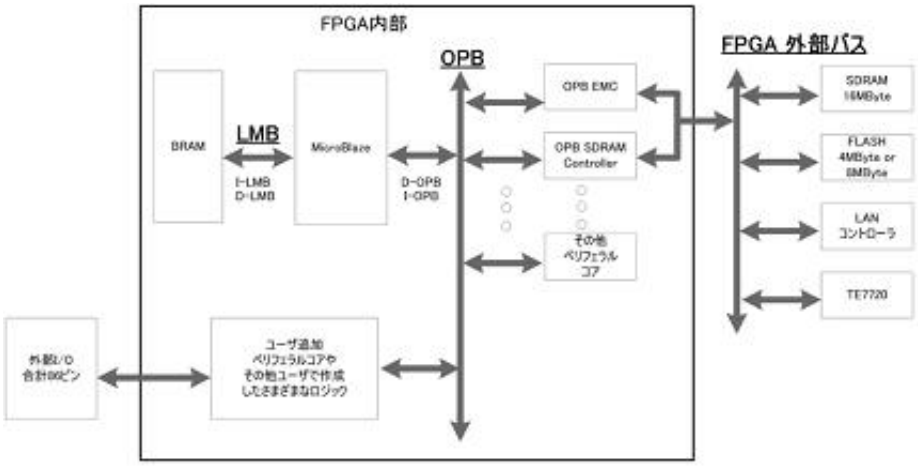

図 17: バス

型式	SZ030
Project Category	All
Family	Spartan3
Device	XC3S1000
Package	FT256
Speed	-4
Synthesis Tool	XST(VHDL/Verilog)
Simulator	ISE Simulator(VHDL/Verilog)

表 3: SUZAKU

- シミュレーション
  作成した回路をシミュレーションする。
- インプリメンテーション
  変換、マップ、配置配線などを行なう。
- プログラムファイル作成
  FPGA に書き込むためのファイルを作成する。
- コンフィギュレーション
  FPGA に作成したファイルを書き込む。

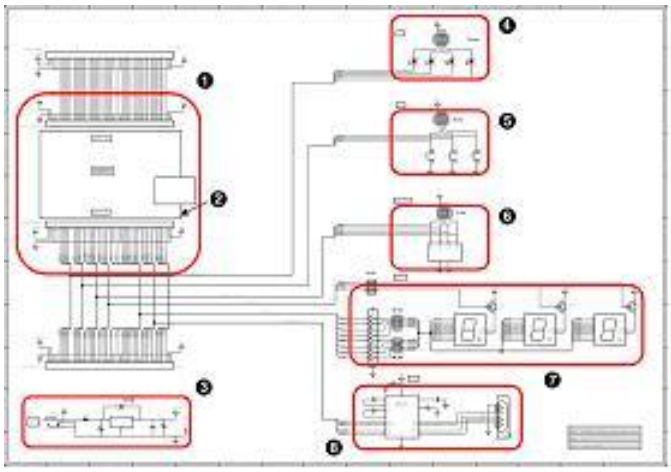

図 18: LED・SW 回路図

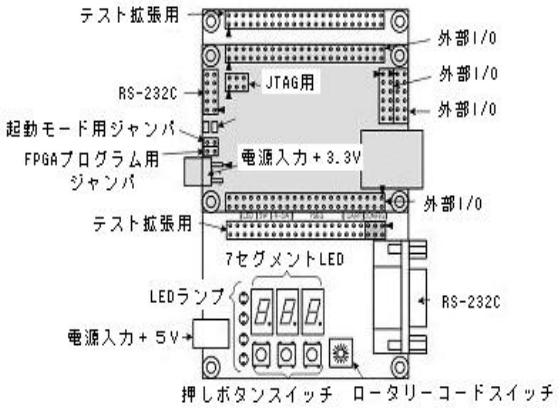

図 19: 各種インターフェースの配置

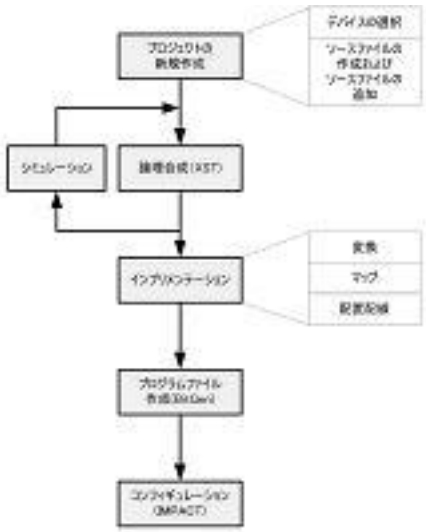

図 20: ISE 開発フロー

3　レートモニター

SUZAKU を用いて 7seg-LED にトリガーレートを表示する回路の作成を行なった。今回は SUZAKU の機能のうち、ユーザー I/O、ユーザーロジック部のみ使用した。

3.1　レートモニター表示

1 秒ごとに表示が切り替わるように設計した。5 桁まで数えられるように設計し、上位 3 桁を 7seg-LED に表示するようにした。そして、4 桁以上のものには小数点をつけ、キロ Hz 単位で表示するようにした。また、6 桁以上になる場合は LED ランプを点灯させ、オーバーフローしたことを明確にするようにした。

3.2　FPGA 内部設計

FPGA の回路設計は HDL(Hardware Description Language) の一種である VHDL(VHSIC HDL) によって行なった。HDL は論理回路を言語によって設計することができるというものである。ここで記述したコードを ISE の論理合成で、実際の論理回路へと変換する。

今回、記述した回路は次のモジュールにわかれている。各モジュールの番号は図 22 にかかれている番号と対応しており、各モジュールは図 22 に示している働きをしている。

1. top
   ここで、以下のモジュールをすべてまとめている。

2. counter
   内部クロックに同期させ、1 秒経過すると RST 信号を出力するように設定している。

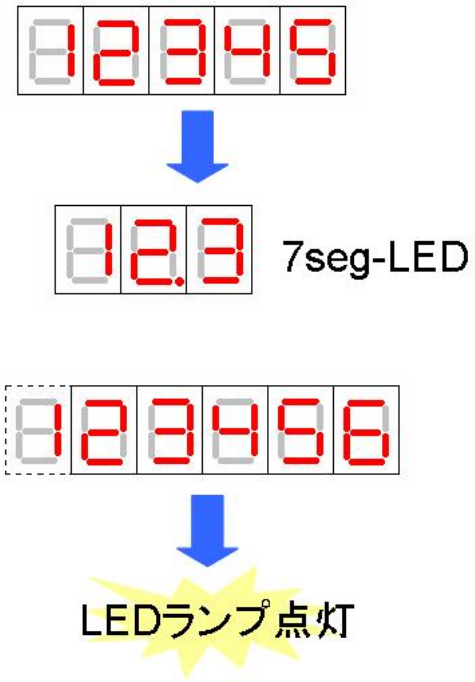

図 21: レートモニター仕様

SUZAKU のクロックは 3.6864MHz なので 1 秒のタイミングはカウンターを用いて計る。

3. seq blink
   入力された外部信号を数える。この際、RST 信号が入力されたら 0 から数えるように設計している。今回数えるのは 1 桁につき 0(0000) から 9(1001) までなので、1 桁ずつ 4bit の counter を 5 つ作成している。これによって 5 桁まで数えられる。またそれぞれの counter が 9(1001) より大きくなったとき 0 に戻り、それより 1 桁大きい位が 1 増えるように設定し、きちんと数えられるようにした。そして 5 桁目が 9(1001) より大きくなったときは、すべての counter を 0(0000) に RST するようにしている。そしてまた、はじめから数えなおすようにしている。

4. select
   1 秒間に入力された外部信号の桁数を求め、3 つの 7 セグメント LED のうち小数点をつけるかつけないか、またつけるとしたらどこにつけるかを判断する。これは先ほどの seq blink で数えられた 5 つの counter を、1 秒経過した時点でそれぞれの位が 0 かどうかを見ることによって行なわれる。例えば、seq blink の counter が上位から
   0000 0000 0111 1001 0100
   ならば、小数点はつかず、7 セグメント LED には 794 と表示される。
   また例えば

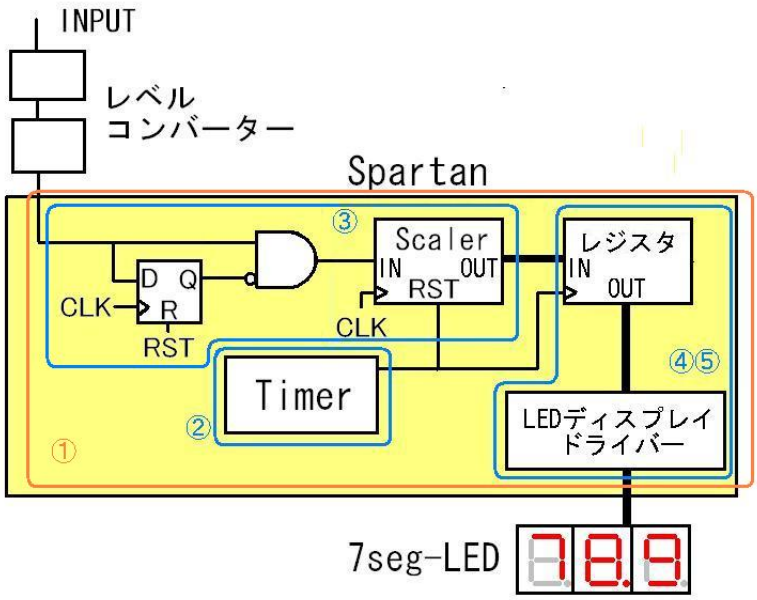

図 22: FPGA 内部回路

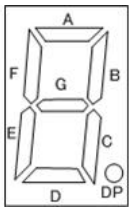

図 23: セグメントの配置

0001 0011 0111 1001 0100

ならば、本来は 13794 だが 5 桁もの精度は必要ないので、下 2 桁は切り捨てて考えられる。そのため 7 セグメント LED には上位 3 桁が残り、小数点も最上位につけられ、1.37 と表示される。

5. seg7 decoder

桁数や小数点の位置などを考慮しつつ、7 セグメント LED に表示の仕方を指定している。7 セグメント LED は数字を構成する 7 つの LED と小数点の部分の 1 つの LED でで構成されている。それらを点灯させるかさせないかを、ここで 1 つ 1 つ 2 進数 (”1”または”0”) で指定している。適当な発光ダイオードを光らせることによって、数字を表すことができる。その例を表 4 に載せる。

7 セグメント LED ではダイナミック点灯という方式を用いている。ダイナミック点

数字	DP(SEG7)	G(SEG6)	F(SEG5)	E(SEG4)	D(SEG3)	C(SEG2)	B(SEG1)	A(SEG0)
0	0	0	1	1	1	1	1	1
1	0	0	0	0	0	1	1	0
2	0	1	0	1	1	0	1	1
3	0	1	0	0	1	1	1	1
4	0	1	1	0	0	1	1	0
5	0	1	1	0	1	1	0	1
6	0	1	1	1	1	1	0	1
7	0	0	1	0	0	1	1	1
8	0	1	1	1	1	1	1	1
9	0	1	1	0	1	1	1	1
A	0	1	1	1	0	1	1	1
B	0	1	1	1	1	1	0	0
C	0	0	1	1	1	0	0	1
D	0	1	0	1	1	1	1	0
E	0	1	1	1	1	0	0	1
F	0	1	1	1	0	0	0	1

表 4: 7 セグメント LED デコーダ（正論理）

灯とは配線の本数を減らすための手法である。複数の 7 セグメント LED に同じデータ線が接続されているので、3 つの 7 セグメント LED を同時に点灯させることができない。しかし高速で 7 セグメント LED を順次点灯することにより、複数の 7 セグメント LED が同時に点灯しているように見せている。そのため、データ線を切り替えるプロセスが必要になっている。

3.3 外部との接続

Spartan で読み取ることのできる信号は、LVTTL(+3.3V/0V) である。一方、外部から入力される信号は NIM 規格 (0V/-0.8V) である。そのため、外部信号を読み取るためには、電圧を変換する必要があった。今回は以下のようにして、電圧の変換を行なった。

まず、レベルアダプタを用いて NIM から TTL(+5V/0V) への変換を行なった。次に回路を自作して、TTL から LVTTL へ変換した。この際、自作した回路を図 26 に載せる。

3.4 動作検証

クロックジェネレータとスケーラーを用いて作成した回路が正しく動作しているかを確認した。様々な周波数 (1Hz,10Hz,100Hz,1kHz,10kHz,100kHz) に変えて確認した。これらはいずれも正常に動作した。

また実際の検出器として、Mesh 付き μ-PIC に接続してレートを測った。この Mesh 付き μ-PIC は、Fe55 を線源としている検出器である。これは同研究室 μ-PIC グループの宮崎一樹氏が実験で用いていたものである。この時は主に、レートの変化について確認し、正常に動作していることを確認した。

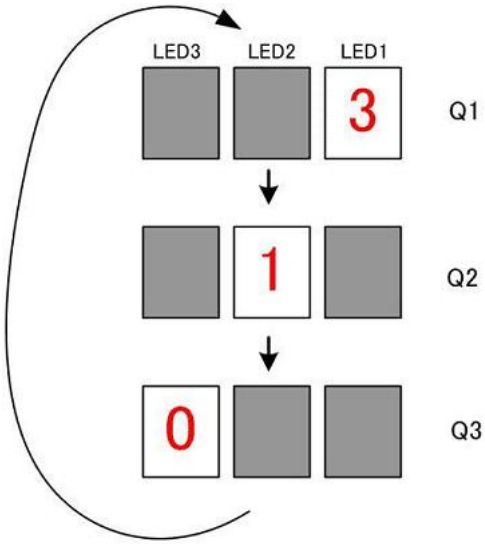

図 24: 7 セグメント LED ダイナミック点灯

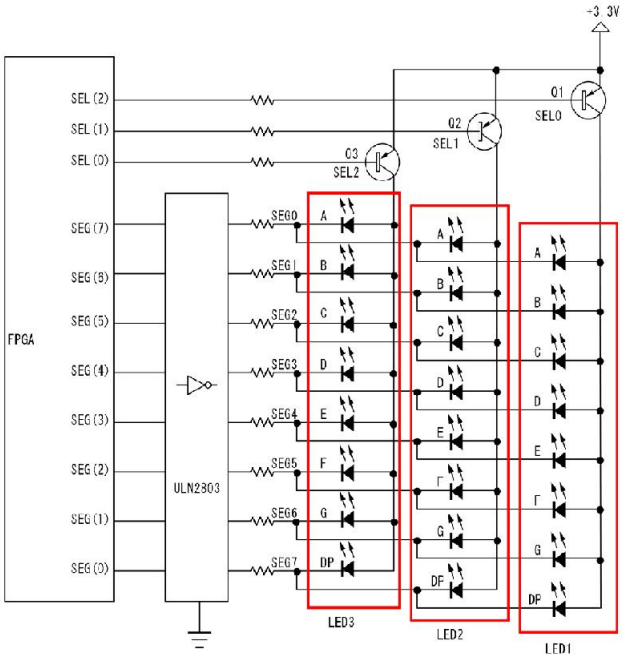

図 25: 7 セグメント LED 周辺回路

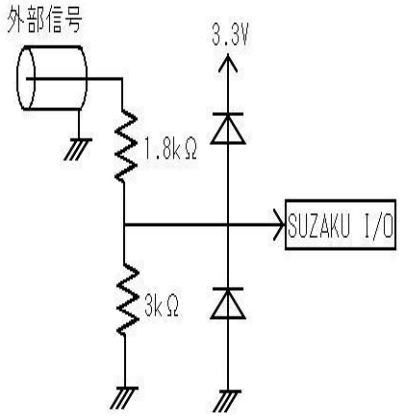

図 26: 自作した回路

3.5 まとめ

今回、Xilinx 社の FPGA（Spartan3）が搭載されている学習用キット「SUZAKU」を使用し、外部信号を読み取りトリガーレートを 7seg-LED に表示する回路を作成した。

そして動作検証をし、正常に動作することを確認した。

また実際の検出器として、Mesh 付き μ-PIC にも接続しレートを測った。こちらも正常に動作していることが確認できた。

3.6 今後の課題

今後の課題としては次の 3 つが挙げられる。

まず 1 つ目は、プログラムを整え汎用性を持たせることにより、改良・変更しやすいようにすることである。今回はきちんとした動作をさせるまでしかできておらず、整ったプログラムにすることができなかった。誰が見てもどこにどのような働きをするモジュールがあるか、分かるようにすることが必要である。実際に、ATLAS の Sector Logic FPGA のコードはいくつかの機能的にモジュール化されており、論理の変更や追加にも柔軟に対応できるようになっている。

2 つ目は、データの読み出しができるようにすることである。「SUZAKU」には Linux も搭載されているが、今回は使用しなかった。これを有効に利用することによって、様々なことができるようになる。

3 つ目は、複雑なロジックに対応できるようにすることである。実際に ATLAS 実験などではかなり複雑なものを扱っている。このようなものに対応していくことが必要である。

4　謝辞

本研究を行なうにあたり、優しく丁寧にご指導いただきました指導教官の藏重久弥准教授に深く感謝いたします。

また ATLAS グループの川越清以教授、山崎祐司准教授、越智敦彦助教には本研究のほかにも様々な助言や指導をしていただきました。感謝しております。

さらに、ハードウェアに関してその時々に応じて助言、指導をしてくださった早川俊氏、中塚洋輝氏、西山知徳氏に感謝いたします。また ATLAS グループの同期である岡村航氏、吹田航一氏、谷和俊氏には研究の際、いつも支えていただきました。こちらも感謝しております。最後に粒子物理研究室の先生、先輩方と同級生の皆さまと、その他の皆さまに感謝しております。

参考文献

SUZAKU Starter Kit　スターターキットガイド (FPGA 開発編) Version 2.1.8
FPGA 活用チュートリアル　　CQ 出版社
ディジタル IC 回路の基礎　　松田勲／伊原充博　著