修士学位論文

High-luminosity LHCへ向けたATLASミューオントリガー検出器用読み出しシステムの開発

東京大学大学院 理学系研究科 物理学専攻
素粒子物理国際研究センター　坂本研究室

二ノ宮 陽一
Yoichi Ninomiya

2011年2月8日

要旨

2009 年末に本格的にスタートした LHC 加速器は、重心系エネルギー 7TeV にて順調に稼働し続けている。データ量も着々と増えてきており、ATLAS 実験でもさまざまな物理が見えるようになってきた。本論文は、約 10 年後に予定されている LHC 加速器のアップグレードに向けた研究について記述している。10 年後のアップグレードで行われるのはルミノシティの増加であり、現状のデザインルミノシティ$1\times10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ から、5 倍大きい、$5\times10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ へとすることによって、積分ルミノシティを 2030 年までに $3000\mathrm{fb}^{-1}$ 目指すものである。またトリガーレートである、L1 トリガーレートは現在の 75kHz から 150kHz になる予定であり、今までの 2 倍の処理速度が求められるようになる。

この論文では、ATLAS のミューオントリガー検出器である TGC の読み出しシステムについて、このシステムの現状の問題点を洗い出し、アップグレードに必要な性能を満たすための条件を求めた。ルミノシティ、トリガーレートの増加から、データサイズは現状の約 1.2 倍、処理速度は待ち行列理論を用いることで算出し、約 3.4μs 以内に処理することを目指した。シミュレーションを行い、内部クロックを 120 M Hz 以上にすることで目標とする数値以内で処理が終えることを確認した。またこの読み出しシステムのテストを行うために開発した汎用 VME モジュールの動作テストや、FPGA の評価用ボードを用いて、MicroBlaze というソフトプロセッサコアを使用したさまざまなテストを行い、読み出しシステムに活用できるかどうかをテストし、その性能を評価することができた。

目次

図目次

表目次

第1章　序論

1970 年代に確立された素粒子標準模型は 90 年代の数々の実験により非常に高い精度で検証され、力の源がゲージ対称性であることが示された。しかし標準模型で予言されている粒子のうちヒッグス粒子に関しては未発見であり、また O(100GeV) を大きく超えたエネルギーでは標準模型を上手く適用できない可能性が示唆されている。CERN はこの状況を鑑みて、2000 年、10 年以上稼働していた LEP 加速器の運転を終え、LHC 加速器建設へと移行したのである。それから度重なる事故や故障により、なかなか本格的な実験がスタートすることはなかったが、2009 年 11 月末から 900GeV での衝突実験が始まり、12 月 13 日には重心系エネルギー 2.36TeV での衝突に成功し、世界最高エネルギーを記録した。2010 年 3 月には重心系エネルギー 7TeV での衝突実験が始まり、本格的に始動したといえるだろう。

この章では LHC 加速器、ATLAS 実験、及び目指す物理について述べる。

1.1　LHC 加速器

LHC(Large Hadron Collider) は、スイスとフランスの国境付近に存在する CERN の地下 100m に建設された直径約 8.5km、リング円周約 27km の陽子陽子衝突加速器である。人類史上初めて TeV スケールでの直接物理探索を可能にした。また重イオン同士の衝突も可能であり、2010 年 12 月にこの重イオン同士の衝突実験が行われた。LHC のパラメータを表 1.1 に載せる。LHC の特徴は、重心系エネルギー 14TeV という世界最高エネルギーである (現在は重心系エネルギー 7TeV で稼働しており、14TeV で稼働するのは 2、3 年先のことになりそうであるが)。LHC は陽子陽子衝突であるため、シンクロトロン放射光によるエネルギー損失を質量の小さい電子に比べ非常に小さくすることができる。従って LEP 加速器と同様のリング長だが、エネルギーを格段に上げることができた。しかし一方で陽子は内部構造を有するハドロンであることと、LHC の衝突頻度が 25ns という高頻度衝突なために膨大な量のバックグラウンドが存在する。この膨大なバックグラウンドから効率よく有用なデータを得るためにさまざまな工夫がされている。

表 1.1: LHC のパラメータ

メインリング長	26.66km	最大ルミノシティ	$10^{34}\mathrm{cm}^{-2}\mathrm{sec}^{-1}$
ビームエネルギー (陽子)	7TeV	ビームエネルギー (重イオン)	2.8TeV
衝突頻度	40.08MHz	バンチ数	2808 個
バンチ当たりの陽子数	10^{11} 個	超伝導双極電磁石の磁場	8.33T

LHC には 4 箇所の衝突点が存在し、それぞれに異なる検出器が設置されている (図 1.1)。汎用検出器である ATLAS(A Toroidal Lhc AppratuS)、CMS(the Compact Muon Solenoid)(図 1.2)、重イオン衝突実験用検出器 ALICE(A Large Ion Collider Experiment)(図 1-3)、B-Physics に特化した LHC-B である (図 1-4)。

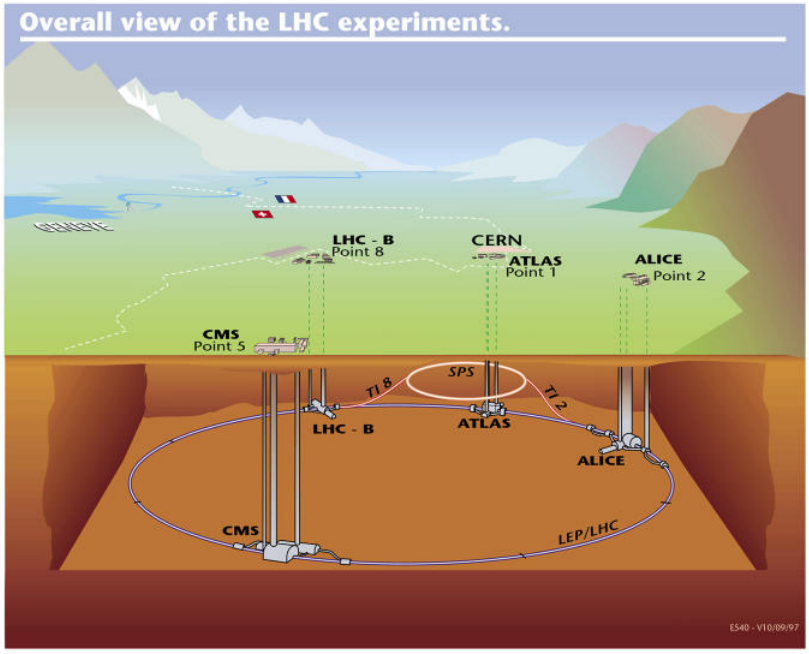


図 1.1: LHC 加速器と 4 つの検出器 [1]

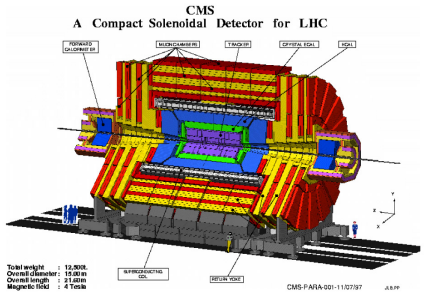


図 1.2: CMS 検出器 [2]

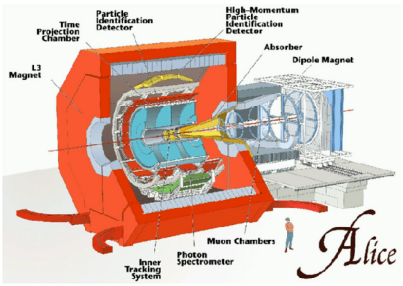


図 1.3: ALICE 検出器 [2]

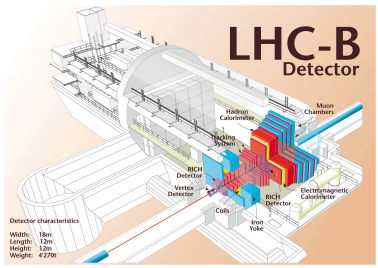


図 1.4: LHC-B 検出器 [2]

1.2 ATLAS 検出器

LHC には新粒子や新しい物理現象の発見を目的とした汎用検出器が 2 つあるが、日本グループが参加しているものは ATLAS 検出器である。検出器は長さ 44m、直径 25m の円筒形で、総重量は約 7000 トンである。ATLAS 検出器の鳥瞰図を図 1.5 に載せた。

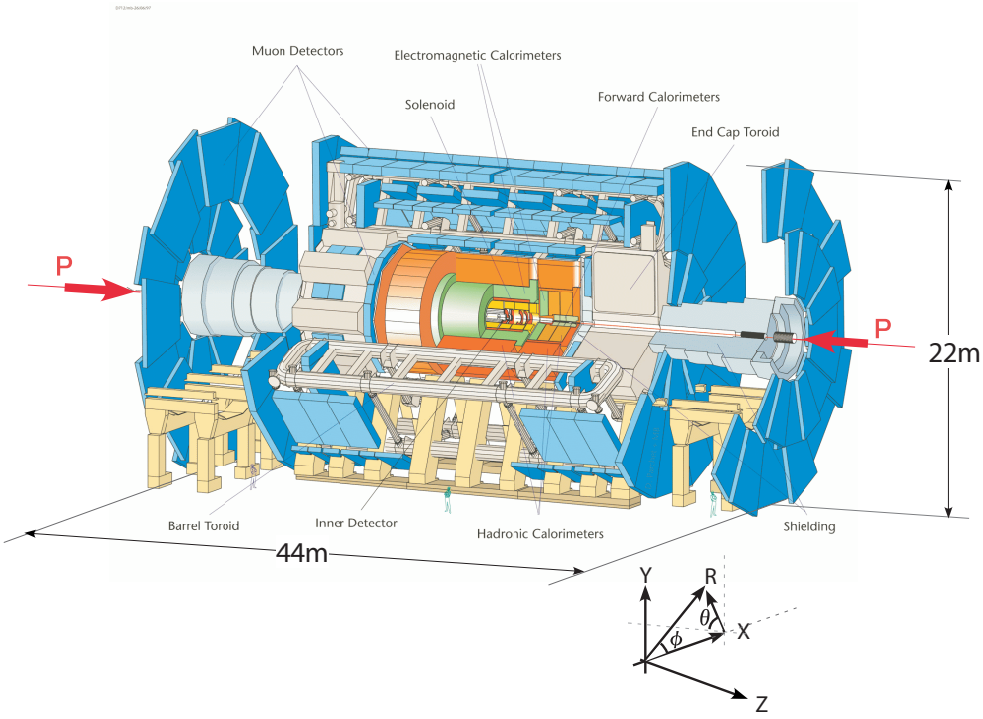


図 1.5: ATLAS 検出器 [2]

検出器は内側から内部飛跡検出器、カロリメータ、ミューオンスペクトロメータで構成され、ソレノイド磁石、トロイダル磁石を備えている。ATLAS 検出器の特徴を以下に述べる。

- カロリメータで、電子とフォトンのエネルギーと位置、ジェットと E_T^{miss} を高精度で測定
- ミューオンスペクトロメータのみでミューオンの p_T を測定できる
- 高頻度のイベントを逃すことなく測定可能
- 10 年以上の稼働を可能とする耐放射線性

次に各検出器について軽く説明する。

1.2.1 内部飛跡検出器

内部飛跡検出器 (図 1.6) は ATLAS 検出器のもっとも内側に設置されており、超伝道ソレノイド磁石の内部に位置する。内部飛跡検出器は以下の 3 種類で構成されている。もっとも衝突点に近いことから、いずれも高い放射線耐性を備えている。

- ピクセル検出器

  最内層にある半導体検出器。この検出器の精度によってバーテックスの精度が決まる。

- シリコントラッカー

  細長い有感領域をシリコン上に施した半導体検出器。r-ϕ 方向に優れた位置分解能を持ち、荷電粒子の p_T が測定可能。

- 遷移輻射トラッカー

  ストローチューブ検出器。飛跡検出の他に遷移輻射を利用した電子の同定を行う。

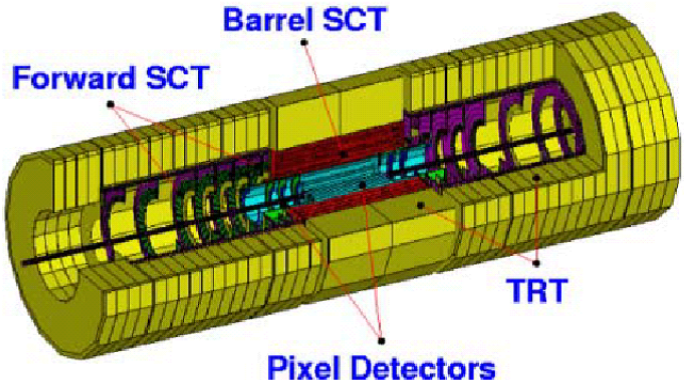


図 1.6: 内部飛跡検出器 [3]

1.2.2 カロリメータ

カロリメータ (図 1.7) は電子、γ のエネルギー、角度を測定する電磁カロリメータと、ジェットの測定を行うハドロンカロリメータが存在する。

- 電磁カロリメータ

  アコーディオン構造の鉛吸収体と放射線耐性に優れる液体アルゴンから成るサンプリングカロリメータを用いている。$|\eta|$ <1.5、1.4< $|\eta|$ <3.2 をカバーしている。電子、γ の同定を行う。

- ハドロンカロリメータ

  電磁カロリメータの外側に配置され、ハドロンの同定、エネルギー測定、ジェットの再構成を行う。$|\eta|$ <1.7 の領域と、1.5< $|\eta|$ <3.2 の領域で構造が異なる。前者は鉄の吸収体とタイル状のシンチレータを交互に重ね合わした構造から成り、後者は鋼の吸収体と液体アルゴンからなるシンチレータとなっている。

1.2.3 ミューオンスペクトロメータ

ミューオンスペクトロメータは ATLAS 検出器の最外部に設置されている、ミューオン検出器である。図 1.8 にミューオン検出器の全体図を載せた。ミューオンは物質の透過力が高く、弱い相互作用で崩壊するため寿命が長い。したがって最も外側でも他の検出器に響されることなく検出することができる。ミューオンスペクトロメータは軌跡検出器である MDT(Monitored Drift Tube)、CSC(Cathode Strip Chambers) と、トリガー用検出器である TGC(Thin Gap Chamber)、RPC(Resistive Plate Chamber) の 4 つ存在する。TGC については次章で述べる。

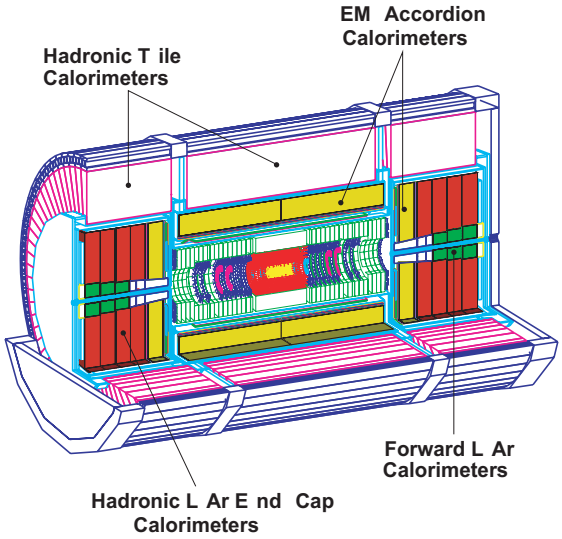


図 1.7: カロリメータ [3]

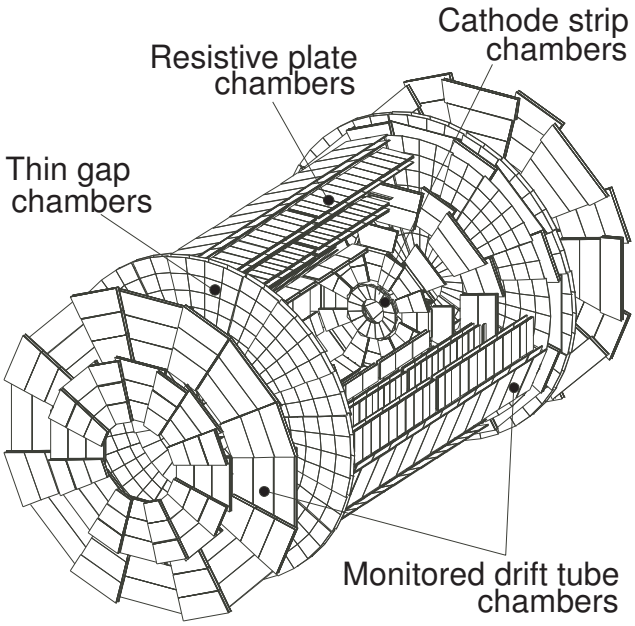


図 1.8: ミューオンスペクトロメータ [3]

- MDT(図 1.9)

  $|\eta|$ <2.7 をカバーする。30mmϕ のドリフトチューブを積層したもので、ドリフト時間とシグナルの大きさから位置を求める。位置分解能は 60μm、総チャンネル数は約 30 万である。

- CSC(図 1.10)

  2 放射線の多い領域である 2.0< $|\eta|$ <2.7 の領域をカバーする、運動量測定用カソードストリップ読み出し MWPC である。ワイヤ間隔 2.54mm、ストリップ間隔 5.08mm、ドリフト時間 30ns 以下、位置分解能 60μm。

- RPC(図 1.11)

  ストリップを用いた検出器を 2 層に重ねた構造で、$|\eta| < 1.05$ をカバーする。R-Z 方向、R-ϕ 方向の運動量を測定し、トリガー判定を行う。

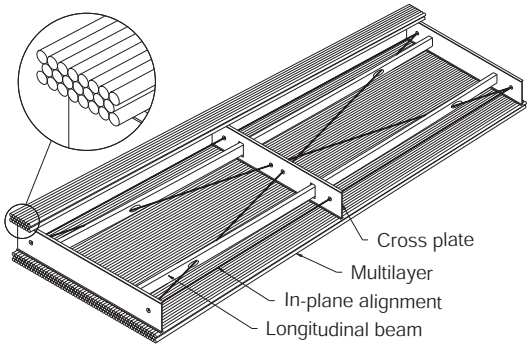


図 1.9: MDT[3]

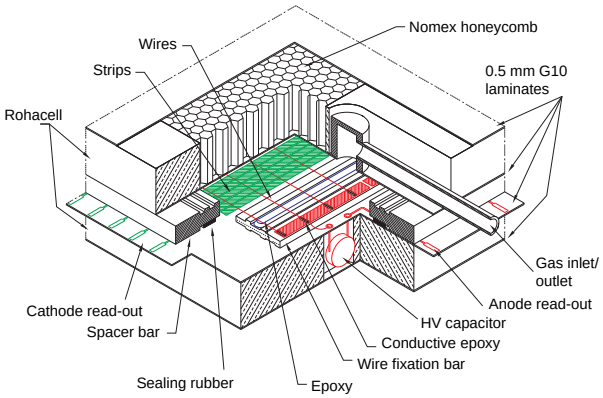


図 1.10: CSC[3]

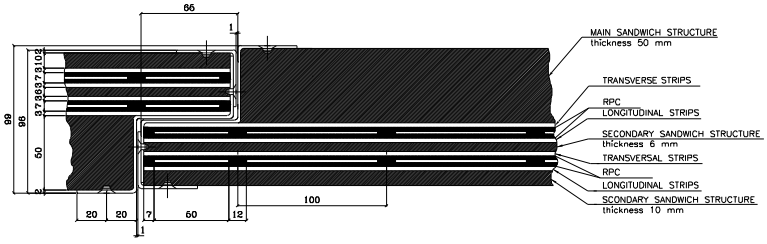


図 1.11: RPC[3]

1.3 ATLAS の目指す物理

ATLAS 検出器は汎用検出器なので、測定可能な物理現象は多岐にわたる。ここでは発見が期待されている Higgs 粒子と超対称性粒子について触れたいと思う。

1.3.1 標準模型 Higgs 粒子

標準模型で確証されたゲージ対称性のもとではすべての素粒子の質量はゼロであり、素粒子の質量起源は未だ解決できていない問題である。これを解決する有望な理論が真空を満たすヒッグス場と自発的対称性の破れである。この理論の直接的な証拠となるのが Higgs 粒子であり、ATLAS 検出器は、質量 100GeV から 1TeV の広範囲で Higgs を探索する能力を持つ。

Higgs 粒子生成過程

Higgs 粒子は重い粒子と結合しやすいので、主に 4 つの生成過程が考えられる (図 1.12)。

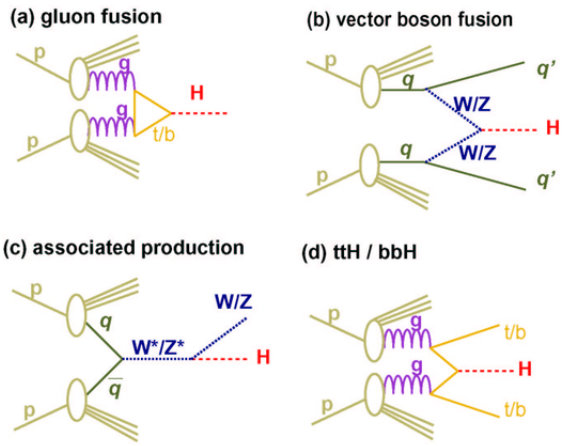


図 1.12: Higgs 生成過程 [4]

1. $gg \to H$ (図 1.12(a))

   Higgs 粒子との結合は質量の大きな粒子ほど強くなるので、主にトップクォークのループを介して生成され、断面積が最も大きくなる。反面、Higgs 粒子の崩壊粒子以外に大きな p_T を持つ特徴的な粒子が存在しないので、バックグラウンドとの識別が困難である。

2. $qq \to qqH$ (図 1.12(b))

   LHC は陽子陽子コライダーであるため、大きな運動量を担うクォーク同士の反応が起こり易く、断面積が大きい。またゲージ粒子を放出して反跳したクォークに起因する大きな p_T を持つジェットが 2 本観測される特徴があり、イベントの選別が行いやすい。

3. $qq \to (W/Z)H$ (図 1.12(c))

   クォークの対消滅で生成されたゲージボソンから、さらに Higgs 粒子が放射される過程。粒子、反粒子コライダーで有望な生成過程。

4. $gg \to ttH$ (図 1.12(d))

   対生成されたトップクォークから Higgs 粒子が放出される過程。トップクォークペアを終状態に含んでいるので、QCD バックグラウンドを減らすことができる。またトップクォークの湯川結合という情報を含んでいる。

Higgs の生成断面積と質量の関係を図 1.13 に示す。

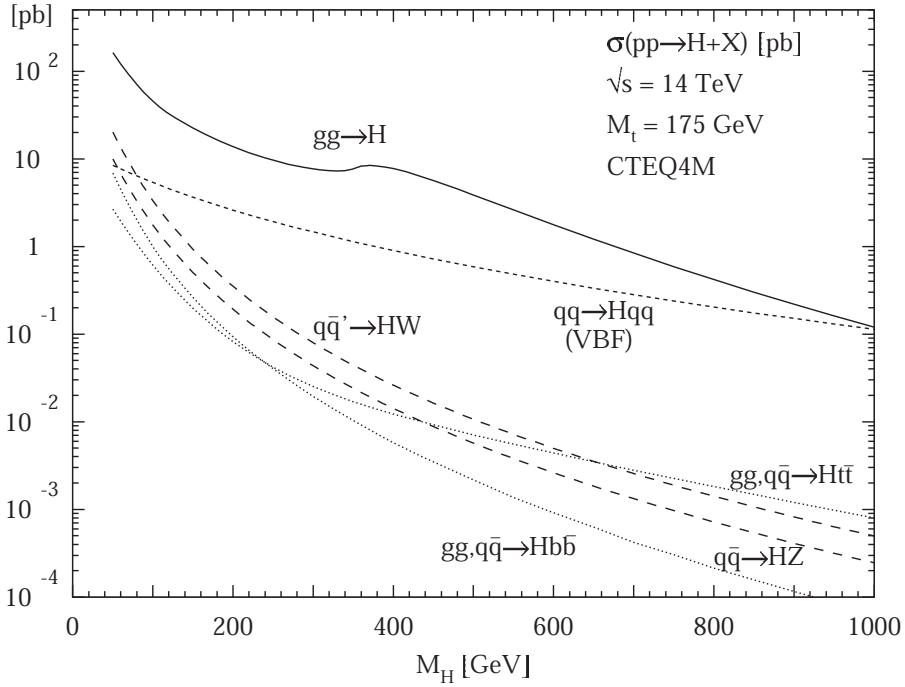


図 1.13: Higgs 粒子生成断面積 [5]

Higgs の崩壊過程

Higgs 粒子は生成後すぐに崩壊する。図 1.14 は、Higgs 粒子崩壊分岐比の質量依存を表したもので、このようにさまざまな崩壊モードが存在する。Higgs の質量によって、分岐比が異なるので、質量が小さい場合と大きい場合に分けて考えてみたい。

1. Higgs の質量が小さい場合 (<140GeV)

   質量が小さいときに有効なチャンネルは、

   - $qq \to qqH \to qq\tau\tau$
   - $gg \to H \to \gamma\gamma$
   - $qq \to qqH \to qq\gamma\gamma$

   で、始めのチャンネルは、どちらかの τ がレプトン崩壊した場合に、そのレプトンをトリガーとしてデータを取得する。また E_T^{miss} の情報を用いて τ の再構築が可能で、さらに Higgs の再構築が可能となる。$\gamma\gamma$ 崩壊は、1 本または 2 本の高い p_T のジェットを要求することで格段に発見能力が上がる。

2. Higgs の質量が大きい場合 (>140GeV)

   このときの主要チャンネルは、

   - $H \to WW$
   - $H \to ZZ$

   で、2 つの W が共にレプトンに崩壊場合、2 つのレプトンの運動量と横方向の消失運動量から計算される横方向質量が HIggs の質量に関係しており、Higgs の質量をエッジとする分布ができる。また ZZ が 4 つのレプトンに崩壊する過程はゴールデンチャンネルと呼ばれ、不変質量を組むときれいなピークが見える。

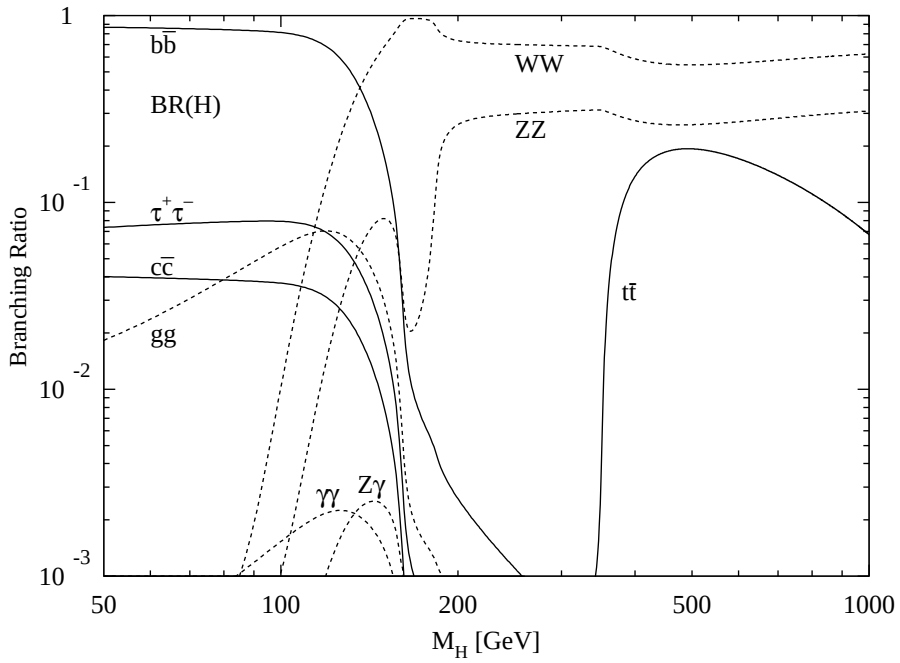


図 1.14: Higgs 粒子崩壊分岐比、質量依存分布 [5]

1.3.2 超対称性粒子

素粒子物理学の究極の目標は自然界の 4 つの力の統一である。超対称性はこの究極の理論への第一歩となりえるものとして有力視されている。

自然界の基本粒子はボソンと、フェルミオンの 2 つに分けることができるが、これはスピンの違いからくる。超対称性はこのボソンとフェルミオンにスピンが 1/2 違うスーパーパートナーと呼ばれる超対称性粒子を予言するものである。この理論が正しければ、ボソンはフェルミオンのパートナーが存在し、フェルミオンはボソンのパートナーが存在することになる。陽子は主にクォークとグルーオンで形成されているので、強い相互作用を通じて、スクォークとグルイーノが生成される。スクォークとグルイーノは運動学的に可能なら強い相互作用を通して 2 体に崩壊し、不可能なら電弱相互作用で 2 対もしくは 3 対に崩壊する。このとき出てくる電弱ゲージーノは 2 つのフェルミオンと最も軽く、観測にかからない LSP(Lightest Susy Particle) に崩壊する (図 1.15)。解析においては、ジェットとともに E_T^{miss} を指標として探索を行う。

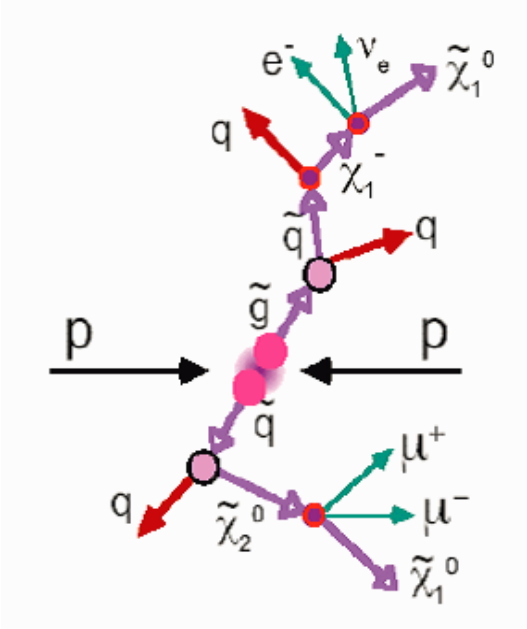


図 1.15: 超対称性粒子の崩壊 [6]

超対称性 Higgs

超対称性が存在する場合、2 つの Higgs 場 2 重項が必要になり、結果的に 5 つの Higgs 粒子がでてくる $(H^{\pm}, h, H, A)$。これらの Higgs 粒子の質量を決めるのが $(\tan\beta, M_A)$ の 2 つのパラメータである。$\tan\beta$ は 2 つの Higgs 場の真空期待値の比で、M_A は重い Higgs 粒子の質量である。以下に MSSM 中性 Higgs 粒子の崩壊モードで観測が期待されるものを示す。

1. $H/A \to \tau\tau$

   MSSM では、高い分岐比が期待できる。標準 Higgs のときの τ に崩壊するチャンネルの解析を用いることができる。

2. $H/A \to \mu\mu$

   1 ほど分岐比は高くないが、精度良く測定が行える。

3. $H \to hh$

   $hh \to \gamma\gamma b\bar{b}$ チャネルを用いる。

4. $A \to Zh$

   Z がレプトンに崩壊する過程で、レプトンをトリガーとして行う方法が有効。

1.4 LHC のアップグレード計画

2011 年現在、LHC は本格的に稼働し始めたばかりである。しかしながら LHC のアップグレードに向けた議論は、LHC が稼働する前から既に行われていた。計画は、現状の検出器で放射線による劣化が始まるとされている 10 年後に検出器の再インストールとともに行う。図 1.16 は LHC アップグレードの現状の予定である。まず最初の 10 年でルミノシティを 5 倍上げる (High-Luminosity LHC)。現状の最大デザインルミノシティは、$10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ なので、アップグレード後は、$5\times10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ になる。さらにその 10 年後にビームエネルギーを 16.5TeV に上げる (Higher Energy LHC) の計画も出てきているが、今回は High-Luminosity LHC について考える。

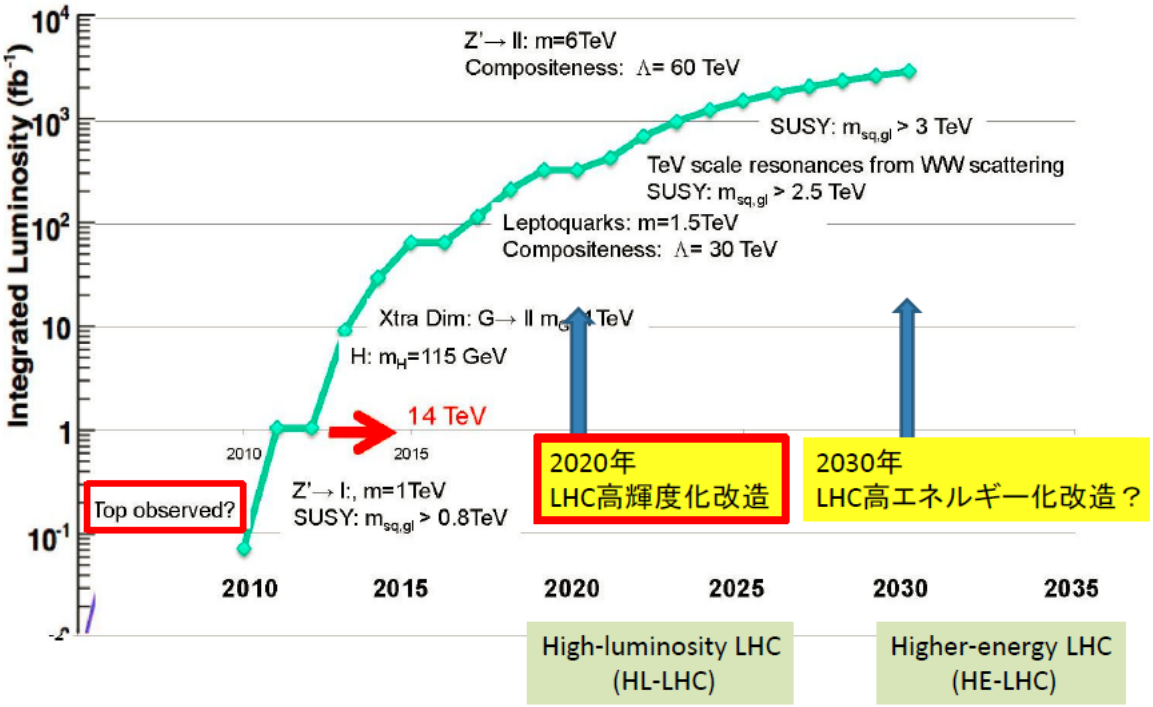


図 1.16: LHC のアップグレード計画 [8]

1.4.1 物理的モチベーション

ルミノシティが上がると、実質的に高いエネルギーのクォークやグルーオンを増やせるので、それはつまり、より質量の大きい粒子の探索を可能にするということである。また既に発見された粒子の精密検査も可能になる。この High-Luminosity LHC 計画は 2030 年までに積分ルミノシティで $3000\mathrm{fb}^{-1}$ 記録することを目指している。この $3000\mathrm{fb}^{-1}$ の物理は Higgs 粒子、超対称性粒子の精密測定、超対称性粒子のさらなる発見を目標にしている。

1.4.2 加速器のシナリオ

どうやって Luminosity を上昇させるのかということについてさまざまな議論がされ、IR(Interaction Region) の 4 重極電磁石、クラブ空洞の取り替えやインジェクターの開発に焦点が置かれている。いろいろな設計がさ

れたが、現在一度白紙に戻し、1 から考えられている。図 1.17 は現状考えられている High-Luminosity へ向けての案である。

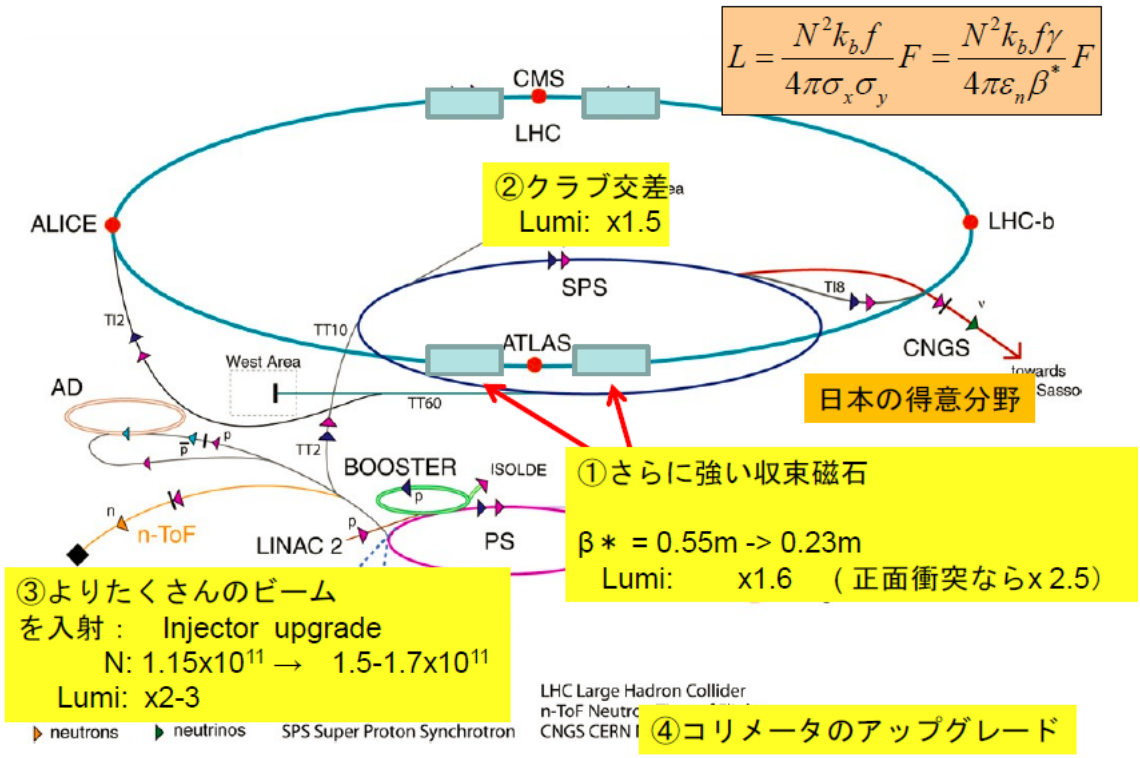


図 1.17: High-Luminosity LHC への考察 [8]

1.4.3 検出器の開発

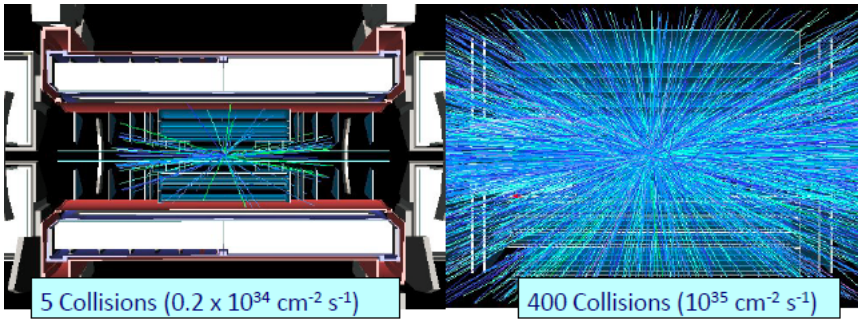


図 1.18: アップグレード前と後のイベント数 [8]

図 1.18 は些か過剰ではあるが、アップグレード後の衝突の様子である。この膨大なイベントの中でバックグラウンドと必要なデータをどのように区別し、取得していくのかということが現状の課題である。各検出器でアップグレードに向け研究開発がおこなわれているので、ここで紹介したいと思う。

内部飛跡検出器

内部飛跡検出器は、ATLAS 検出器のもっとも内側なので、その分放射線損害も他より大きく受けることになる。計算によるとあと 6 年くらいで放射線損傷による寿命がくると言われている。そこで 6 年後、内部飛跡検出器は IBL(Insertable B-Layer) と呼ばれるピクセル検出器を ATLAS 検出器に組み込むことで、データ取得のパフォーマンスを落とすことなしに、2020 年ごろに予定されている大幅なアップグレードへ向かおうとしている。そして 2020 年には、ID トラッカーすべてをアップグレード用に交換する。その際に高イベントにも耐えられるように、ピクセルレイヤー、ストリップレイヤーの数が増える予定である。また日本グループでも図 1.19 のようにさまざまな R&D が行われている。

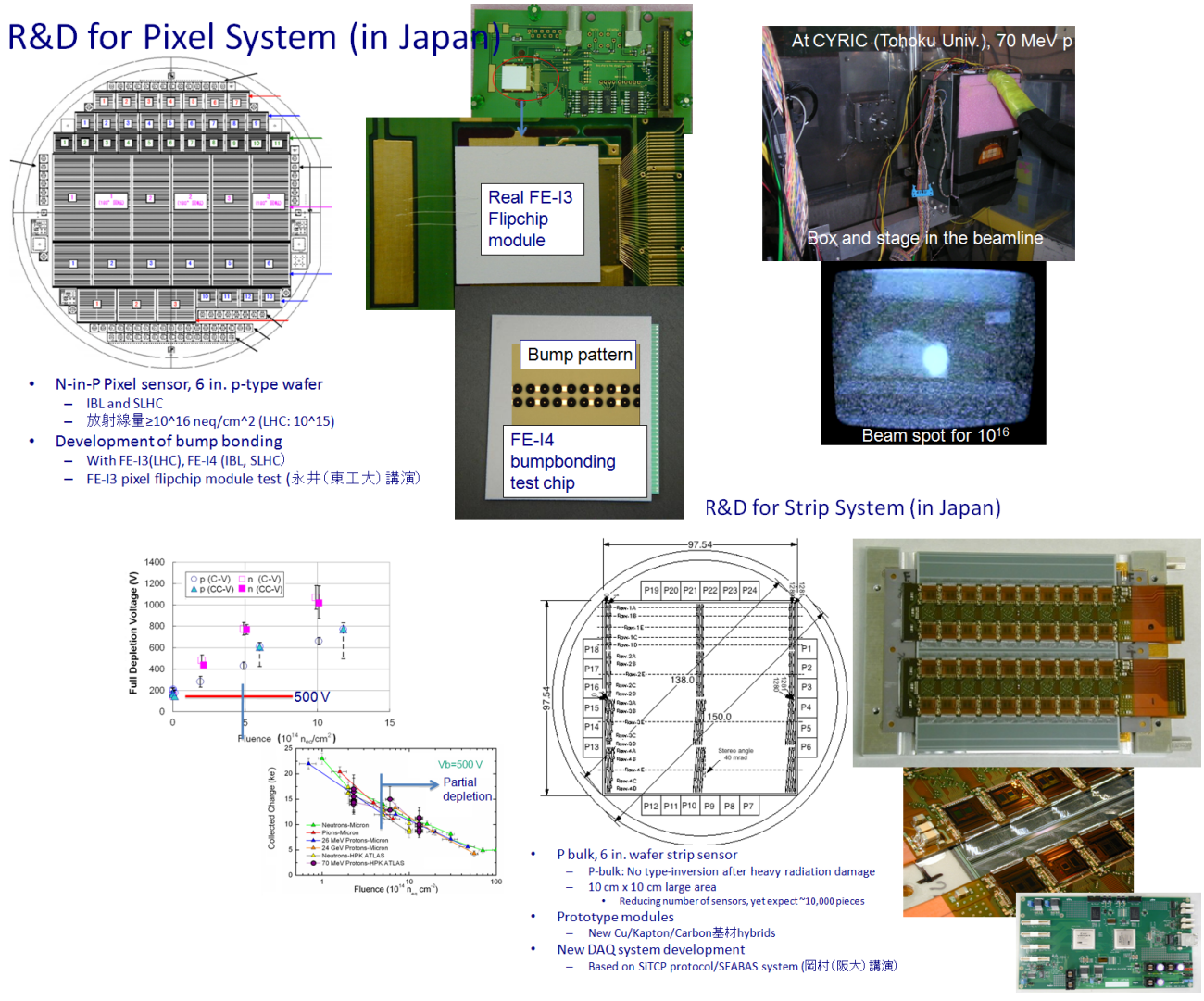


図 1.19: アップグレード検出器開発 [20]

カロリメータ

カロリメータの大部分がルミノシティ10^{35} でも耐えられる構造をしており、フォワードの液体アルゴンカロリメータだけに問題があるようである。

ミューオンスペクトロメータ

ミューオンスペクトロメータの主な問題はバックグラウンドである。バックグラウンドがあまりにも大きいと、正しくデータが得られなくなってしまう。そこでミューオンスペクトロメータもアップグレードにむけ、MDT や TGC などのさまざまなミューオン検出器が R&D されているところである (図 1.20)。

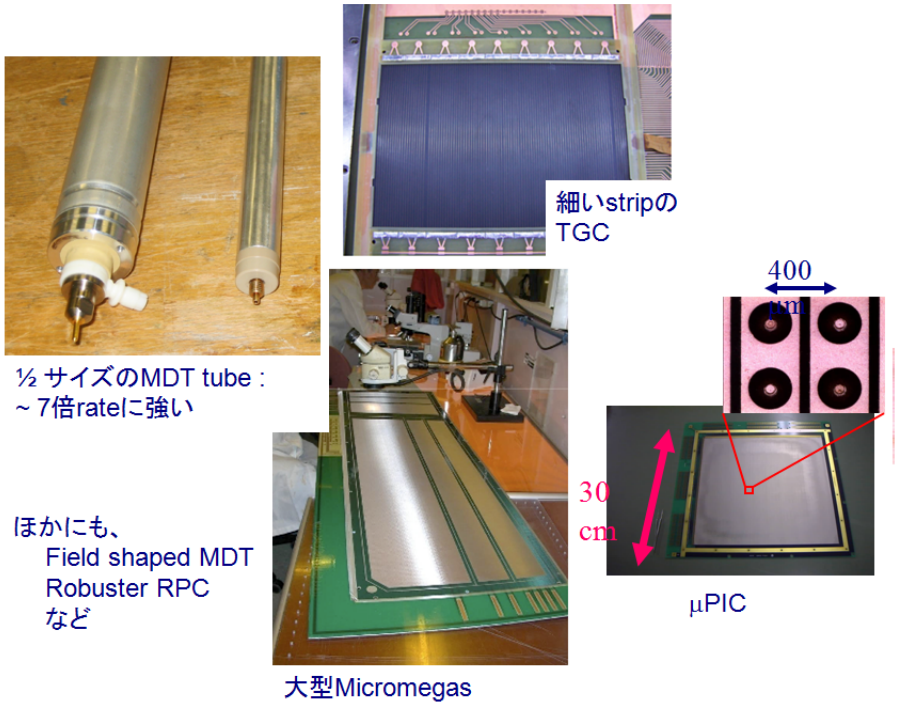


図 1.20: ミューオン検出器のアップグレード [20]

トリガー

トリガーについても多くの議論がなされている。TGC はミューオンの p_T を用いてトリガー判定を行っている。アップグレードによって、ルミノシティが 5 倍になるが、このとき、現状と同様の p_T 閾値を用いたのでは L1A も 5 倍の 375kHz になってしまうだろう。しかし 375kHz という L1 トリガーレートでは現状の TGC エレクトロニクスでは全く処理が間に合わない。かといって、ミューオンの p_T の閾値を上げるのにも限界があることがわかっている (図 1.21)。

そこで、トポロジー情報を用いるなどの案もでているが、まだ議論の域を出ていないと言えるだろう。

1.4.4 アップグレードに関してのまとめ

現在 LHC は稼働中だが、10 年後のアップグレードの関した議論も活発におこなわれている。ルミノシティを増加させたときの衝突イベントは、膨大なバックグラウンドをもたらし、それらに耐えられる、処理できる検出器は現状稼働しているものにはほとんど存在しないことがわかった。またアップグレードに対応できるように様々な研究が行われていることもわかった。これからまだまだ多くの議論がされ、検出器やトリガーレートも決まっていくのだろうが、これからの議論ではアップグレードの目標数値である、ルミノシティは $5\times10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$、L1 トリガーレートは 150kHz で計算していきたいと思う。

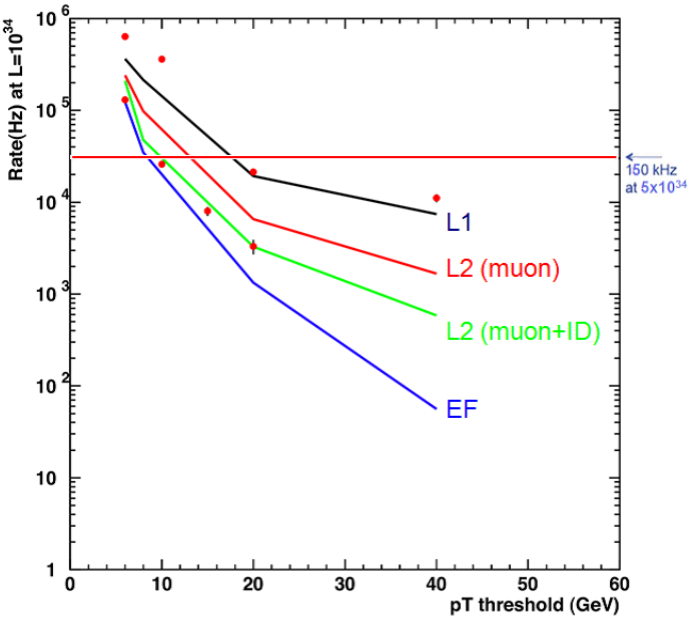


図 1.21: ミューオンの運動量とトリガーレートの関係 [20]

第2章　TGCのデータ読み出しシステム

TGC(Thin Gap Chamber) は、ATLAS 検出器の数あるミューオンスペクトロメータの中で物理的に有用な事象のデータを効率よく選出するために使用されるトリガーシステムの 1 つである。この TGC は主に日本グループとイスラエルグループによって作成された。まず TGC システムについては簡単に触れた後、本研究に関係のあるレベル 1 トリガーシステムと TGC のリードアウトライン、リードアウト系のエレクトロニクスについて述べていく。

2.1　TGC

TGC は高エネルギー実験でよく用いられる MWPC(Multi-Wire Proportional Chamber) の一種であるが、ワイヤー間隔 1.8mm よりもアノードワイヤー、カソードストリップ間が 1.4mm と狭くなっていることに特徴がある。図 2.1 に TGC の断面図、図 2.2 に TGC のパラメータを示す。TGC は総チャンネルが 32 万で、チェ

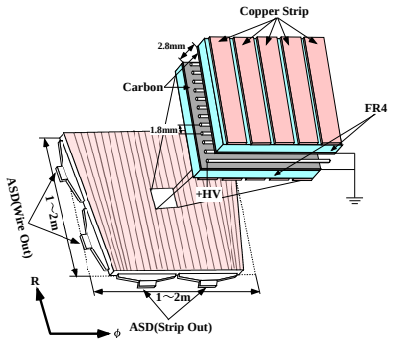


図 2.1: TGC の断面図 [4]

陽極ワイヤーピッチ	1.8mm
陽極ワイヤー径	$\phi 50\mu$m
陰極カソード面厚	10μm
ガス成分	$CO_2 + n-pentane(55:45)$
陽極－陰極面間隔	1.4mm
ワイヤー張力	350g 重
印加電圧	3.1kV
カーボン面抵抗	~1MΩ/□

図 2.2: TGC のパラメータ

ンバー数は 3600 枚。ワイヤーが R 方向、ストリップが ϕ 方向の情報を出力することで 2 次元での読み出しが可能となっている。カソード面に挟まれた領域に CO_2/n-pentane(55/45) が封入されており、TGC を通過した荷電粒子は CO_2 を電離して、電子・陽イオン対を作り出す。ワイヤー間が 1.8mm と短いのはこの電子のドリフト時間を短くするためであり、アノードワイヤーとカソード面の間隔が 1.4mm と狭いのは、陽イオンのドリフト距離を短くし、高レートの粒子の入射時の検出効率の低下を抑えるためである。

2.1.1　TGC の配置

TGC は衝突点から約 7m 離れたところに EI/FI、約 14m 離れたところに Big Wheel が設置されている。TGC の配置を図 2.3 に示す。Big Wheel は End-cap トロイド電磁石の後方に設置され、$1.05 < |\eta| < 2.7$ の領域を網羅する。内側から M1、M2、M3 と呼ばれ、M1 は 3 層、M2 及び M3 は 2 層で計 7 層の TGC チェンバーで Big Wheel を構成している (図 2.4) 。また M2 では 2 つのレイヤーのチャンネルが 1/2、M3 では 3 つのレ

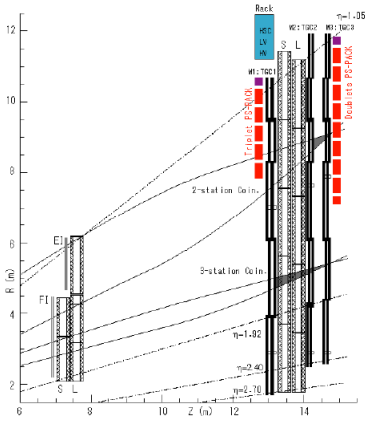


図 2.3: TGC の配置 [3]

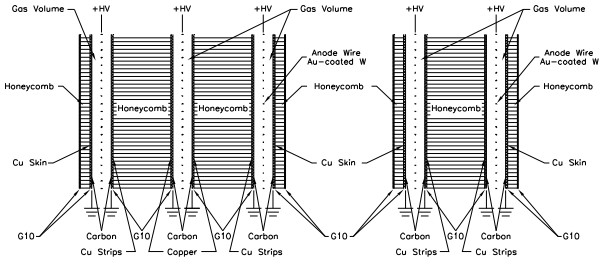


図 2.4: トリプレット (左)、ダブレット (右)[3]

イヤーがのチャンネルが 1/3 ずつずれており、実効的に位置分解能を上げるようにデザインされている。P_t 値は M3 のヒットを基準として、衝突点と M3 のヒットを結んだ直線からのずれを M1、M2 で測定して求める。Big Wheel は 12 分の 1 を 1 セクターと呼び、1 セクターのみで単独動作することができる。

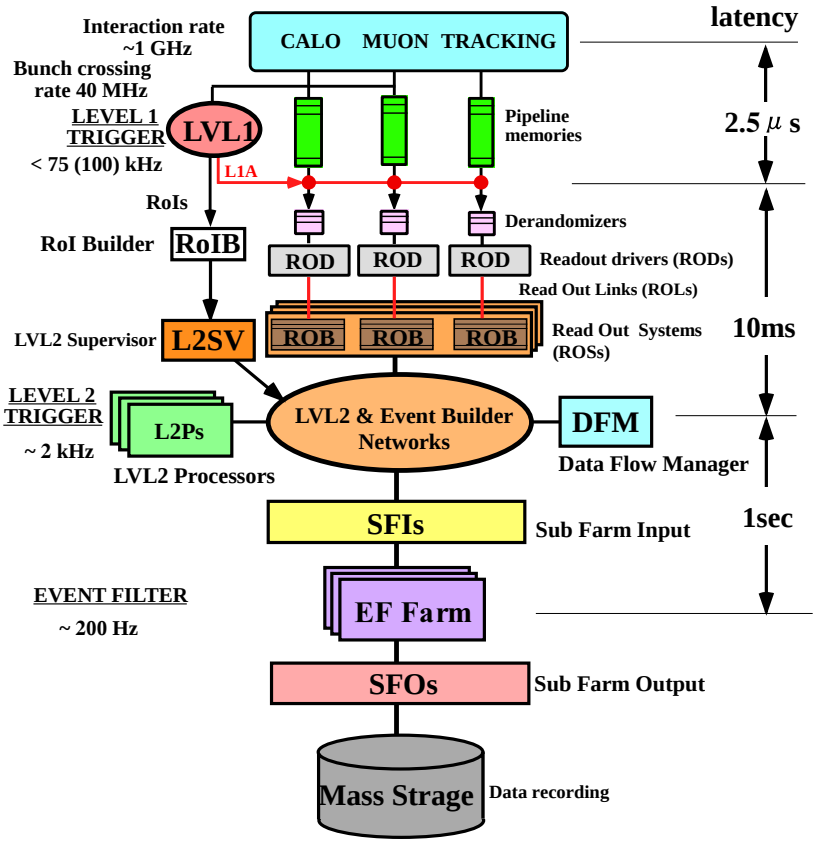


図 2.5: Trriger & DAQ システム [9]

2.2 レベル 1 トリガーシステム

ATLAS 実験は、Higgs 粒子や超対称性粒子などの未発見の粒子探索を行うための汎用検出器であるが、LHC が陽子陽子ハドロンコライダーであるために膨大な量のバックグラウンドが存在する。このバックグラウンドの中から興味のあるイベントを選別するために多段階のトリガーシステムを採用している。その一番始めのトリガーシステムあるレベル 1 トリガーシステムは、カロリメータで得られる e、γ、ジェット等のシグナルと TGC、RPC で検出されるミューオンを用いてバックグラウンドを排除していくものである。TGC は ATLAS 検出器の前後方部を担当しており、$1.05 < |\eta| < 2.7$ の領域をカバーしている。またレベル 1 トリガーは LHC の衝突頻度 25ns という高頻度に対応するためにすべてハードウェアで処理される。

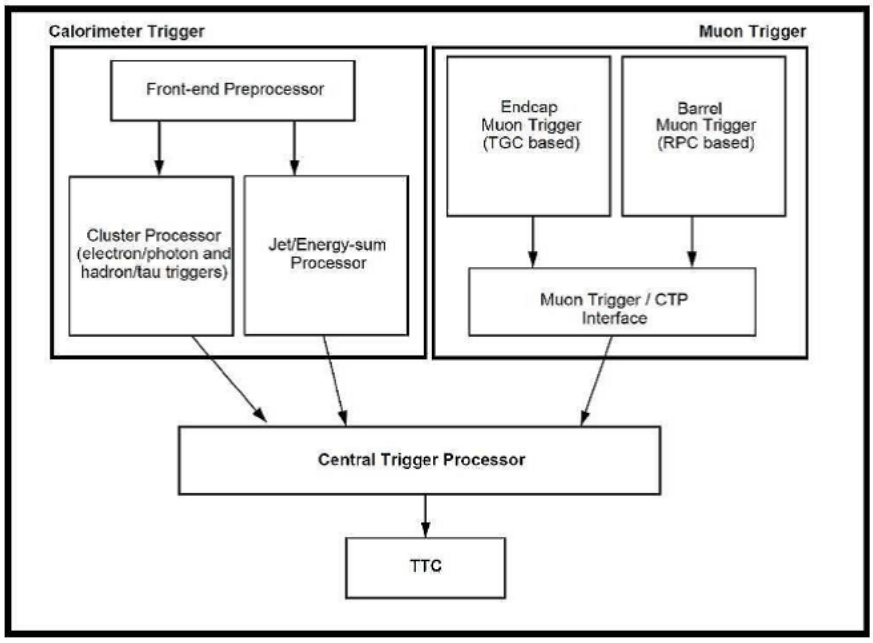


図 2.6: レベル 1 トリガーブロック図 [10]

図 2.5 はトリガーと、DAQ のシステム一覧である。データは上から下に流れていく。カロリメータの情報は CTP(Central Trigger Processor) に送られ、TGC、RPC の情報は MUCTPI(MUon triger CTP Interface) で統合されてから CTP に送られる。CTP で最終的な判定を行い、L1A(Level-1 Accept) と呼ばれるトリガー信号を TTC(Timing Trigger and Control system) を通じて ATLAS の各検出器のフロントエンドに運ばれる (図 2.6)。この L1A は 2.5μs 以内に発行されることが決まっており、各検出器には L1A の判定を受ける間データを溜めておくバッファが実装されている。このバッファは L1 バッファと呼ばれ、TGC は 3.2μs の間データを保持できるバッファを備えている。各検出器はこの L1A を受け取るとデータを後段のリードアウトラインに流し、ROD(Read Out Driver) で ATLAS 共通のフォーマットにイベントビルドして ROS(Read Out System) へと送る。

2.3 TGC のリードアウトライン

図 2.7 は TGC のリードアウトラインを表している。TGC で観測されたデータは、ASD(Amplifier Shaper Discriminator) によって増幅、整形、デジタル化し、設定している閾値電圧を超えた信号を LVDS(Low Voltage Defferential Signal) で PS Board(Patch panel and Slave board asic board) の中の PP(Patch panel board)

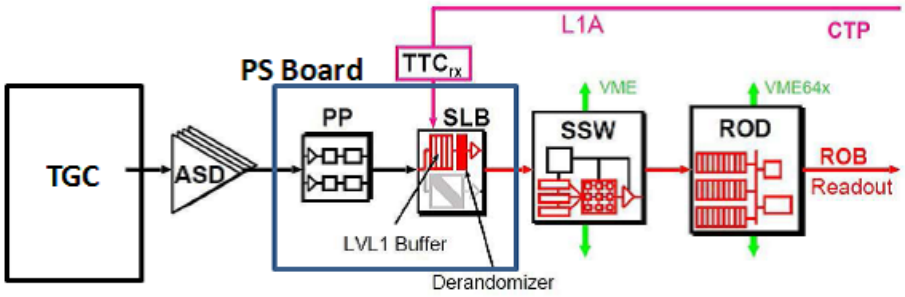


図 2.7: TGC のリードアウトラインのエレクトロニクス

ASIC に送る。そこで各チェンバー内のチャンネルに粒子が飛来するまでにかかる時間 TOF(Time Of Flight) や信号が PS Board に来るまでのケーブル遅延を調節し、TTC から供給される LHC clock と同期が取られ、バンチ識別が行われる。その後 SLB(SLave Board) 内のレベル 1 バッファにデータが一時的に保存される。TGC のレベル 1 バッファは 128 clock(3.2μs) の間 L1A 信号を待つことができる。L1A 信号を受け取った信号のみが次の SSW(Star SWitch) に送られ、データの圧縮を行い、ROD へと送られる。

次に SLB、SSW、ROD について詳細を述べる。

2.3.1 SLB

SLB 内に搭載されている SLB ASIC は、データの読み出しに関する部分だけでなくトリガーに関する部分も担っているが、ここではデータ読み出しに関する部分だけ触れるとする。

SLB のリードアウト部は、レベル 1 バッファとデランダマイザにより成る。レベル 1 バッファは 128 段のシフトレジスタで 1 clock、1 段に対応している。

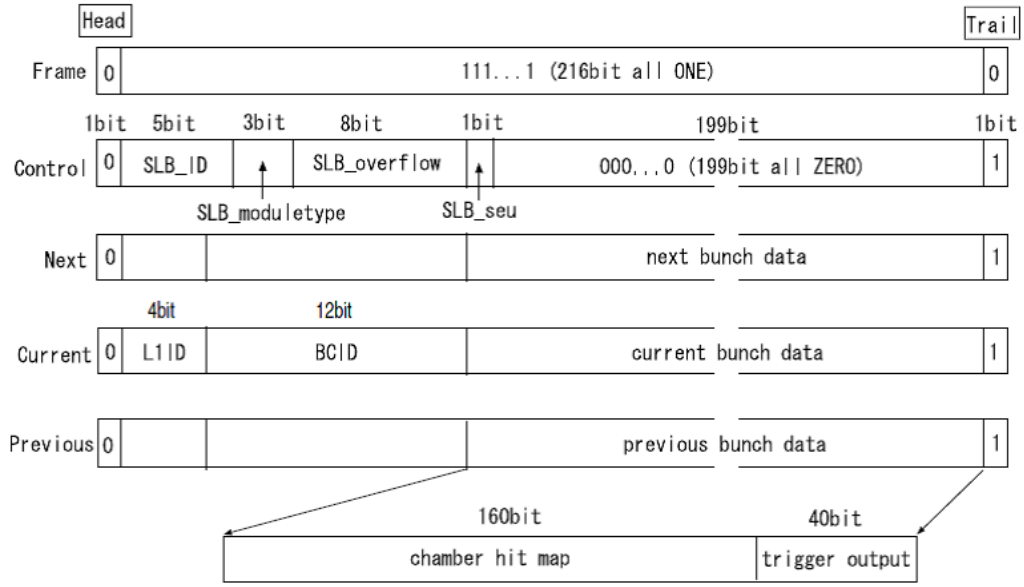


図 2.8: SLB の出力データフォーマット [11]

図 2.8 は SLB の出力データフォーマットである。ATLAS 実験では、L1A を受け取った前後のバンチデー

タも出力することになっており、Frame データ、Control データ、Hit データ ×3 の計 1090bit のデータがデランダマイザを通して SSW へと送られる。SLB から SSW への送信は LVDS 規格が用いられているが、この送信速度は 40Mbps である。現状は 2 つの送信ラインを用いることで実質的な送信速度を 80Mbps にしているが、この送信速度で対応できるのはレベル 1 トリガーレートが 130kHz までであり、もしアップグレードにより 150kHz まで上がった場合データの処理が間に合わず、溢れてしまうことになる。また現 SLB のレベル 1 バッファの容量では小さすぎて、アップグレードに対応できないことがわかっており、現在 SLB についてもアップグレード研究がなされている。

2.3.2 SSW

SSW は ROD の前段階に位置するモジュールであり、SLB からのデータを圧縮するのが主な役割である。表 2.1 は 1ROD あたりに接続される SSW の個数 10 個がいくつの SLB と接続しているかを表にしたものである。

表 2.1: SSW に接続させるＳＬＢの個数

SSW	0	1	2	3	4	5	6	7	8	9
SLB 数 (個)	18	18	10	15	15	15	15	10	11 or 12	6

圧縮はゼロサプレスという方法を用いている。図 2.9 にゼロサプレスの方法を示した。これはバンチデータ 200bit を 8bit ずつ 25 個に分け、0 でない部分だけ取り出すものである。SLB からのデータはほとんどが 0 であり、これにより大幅にデータを圧縮することができる。しかしこのような圧縮方法はヒット数が増えると圧縮効果が格段に減ってしまう。アップグレード後ヒット数が多くなったときにはほとんど圧縮効果が無くなってしまうことがわかっており、SSW もまたアップグレードにむけて研究が必要な部分である。

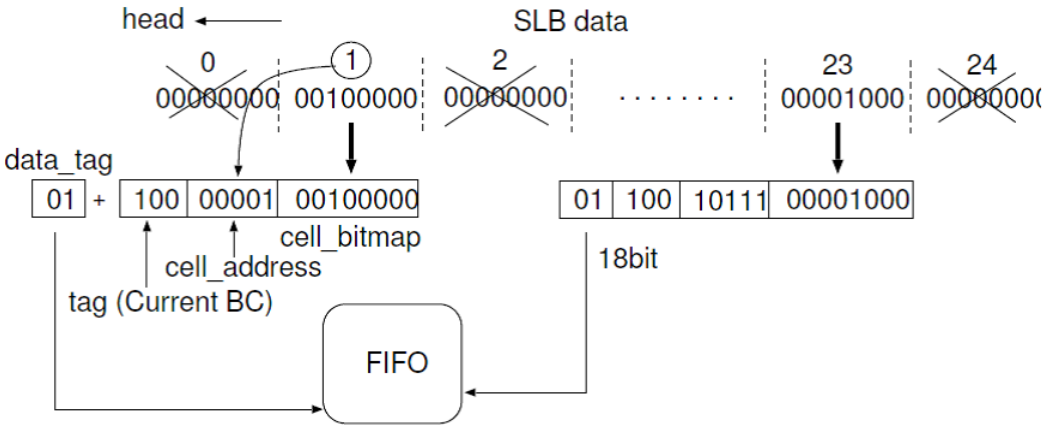


図 2.9: ゼロサプレスによるデータ圧縮 [11]

さらに SSW は SLB からの情報と SSW の情報を合わせ、ROD に送る。SSW からの出力フォーマットを図 2.10 に掲載する。SSW からのデータサイズの考察は 3 章で述べることにする。SSW から ROD への出力は Glink という光通信規格が用いられる。

Version 01 of the TGC Front End link data format

Dec.12,2005 (after PRR) version

Event Header 000 Now, Record Type=01

31 30 29	28 27	26 25 24 23	22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
000	RecType	SSWID	RX mask pattern (1=enabled, 0=disabled)	

Record Type (RecType) is 01 in this format version, hard-wired in FPGA
SSWID is arbitrarily set by a dip-switch on each SSW board

SLB header 010

31 30 29 28 27 26 25 24	23	22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
010 SLBID	0	BCmap Mod Type	0 L1ID BCID

BCmap shows 3BC data lines taken by RX. 3bit shows {next, current, previous} events. 1=adopted. 0=discarded
SLBID, Mod Type, L1ID and BCID are all SLB's data. See SLB documents.

SLB header 011 0

31 30 29	28	27	26 25 24 23 22 21 20	19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
011	0	0	RXID	0 RX FIFO status	SLB-OVF RX-OVF

RXID is RX identified number from 0 to 22.
RX FIFO status tells what amount of data are stored in RX-FIFO then.
SLB-OVF is SLB's data. See SLB documents.
RX-OVF is RX-FIFO overflow counter. This tells the snapshot value when this word is sent from RX to TX.

SLB trailer 011 1 This word appears after SLB data words only when there is an error

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
011 1 SEU OVF LVDSlink RX error state

LVDSlink=LVDS links status. 2bits are {now,old}. 1=Not linked. 0=Linked.
SEU = SLB SEU flag. See SLB documents.
OVF = RX-FIFO overflow flag. If OVF=1, some overflows have happened in this RX data.

SLB data 100, 101, 110

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	
100 cell address	cell bitmap
101 cell address	cell bitmap
110 cell address	cell bitmap

In any order:
Cell data for Current BC data
Cell data for Previous BC data
Cell data for Next BC data

PAD word 110

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
110 11111	0

i.e. 0xDF00

Event Trailer 111

31 30 29 28 27 26 25 24 23 22 21 20	19	18	17	16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
111 0x1CA	Glnk	T1C	NRC	T2C	XOR check sum

Glnk = Glink TX status. "Locked" signal of Glink Tx. 1=Not locked. 0=Locked
T1C = Timeout1_count_flag. Time-out to collect the event fragment from all the enabled input ports.
NRC = Nores_count_flag. No response from RX FIFO of "enabled" (not masked) input port.
T2C = Timeout2_count_flag. Time-out to collect the event fragment from each enabled RX FIFO.
These three flags are reset at every event.

The XOR operation includes the first word(16bits) of the event header through the first word of the event trailer. When the result is XOR'ed with the XOR checksum word, the result becomes zero. (the XOR does not include the 0x0B0F an 0x0E0F framing words)

Framing

Each event is preceeded by the 32-bit word 0x0000'0B0F and followed by the 32-bit word 0x0E0F'0000, both words are sent in Glink control mode.

図 2.10: SSW の出力フォーマット [11]

2.3.3 ROD

ROD は TGC のデータが最終的に集まるモジュールであり、イスラエルグループによって作成された。ROD までは ATLAS の各検出器独自の設計が許されているが、ROD 以降は ATLAS 検出器で共通のフォーマットが用意されており、ROD はそのフォーマットに整形する役目を行う。まず複数の SSW からのデータを FIFO に格納し、トリガー情報を元にパラレルで受け取ったデータをシリアルにし、さらに ATLAS のフォーマットに従ってヘッダーとトレイラーをつけ、そしてデータを Slink という CERN で開発された光通信規格で ROS へとデータを送信する。図 2.11 は TGC チェンバーと ROD の接続を示している。図のように ROD は TGC の 1 セクターに 1 つの割合で接続されている。つまり片サイドで 12 個、全部で 24 個の ROD が実験時には稼働している。

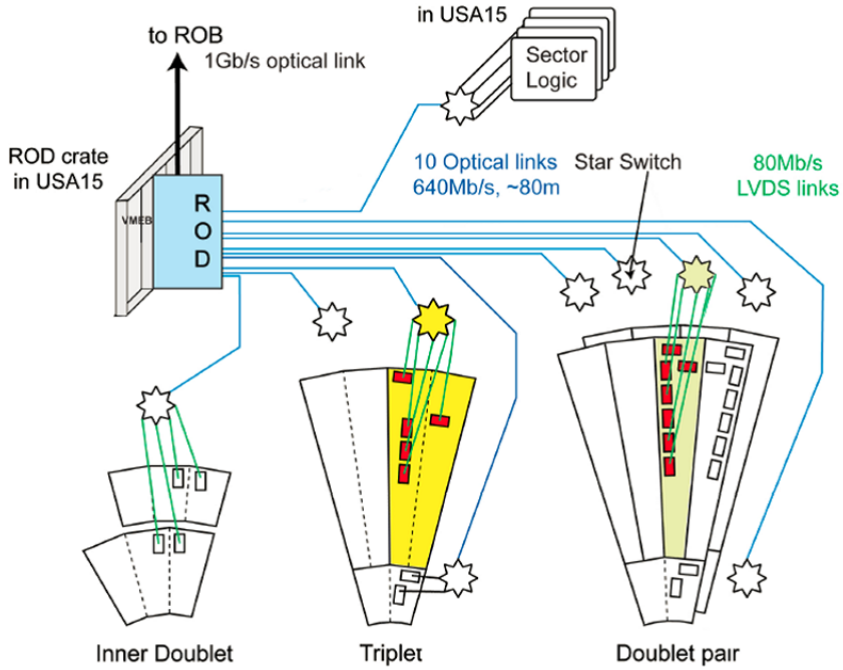


図 2.11: ROD の接続

エレクトロニクス

ROD は ATLAS 検出器から離れたカウンティングルームに設置されている、6U サイズの VME モジュールである (図 2.12)。4 つのメザニンカードを挿すことができ、このうち 2 つのメザニンカードは SSW からのデータを受け取る Glink コネクターが 1 つのボードにつき 6 個搭載されている (図 2.13)。従って、最大 12 個の SSW からデータを受け取ることが可能だが、現在は 12 個すべて使うことはなく、最大 10 個となっている。さらに 1 つは Slink メザニンカードで、データを送信する際に使用し、残りの 1 つは TTC からのシグナルを受け取るために使用する。

ROD に搭載されている FPGA は Xilinx 社製の XC2V2000FG676 を用いている。この FPGA に HDL デザインをダウンロードすることで、ROD に必要な処理を行っている。また CPLD も搭載されており、これにより VME からのアクセスも可能となっている。図 2.14 に ROD のブロック図を掲載した。

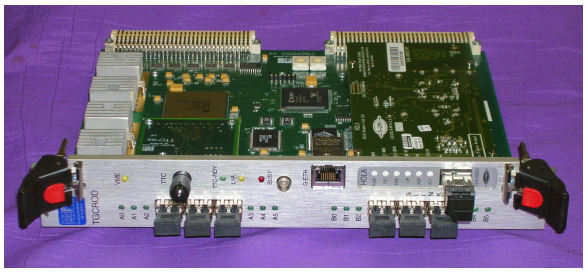


図 2.12: ROD

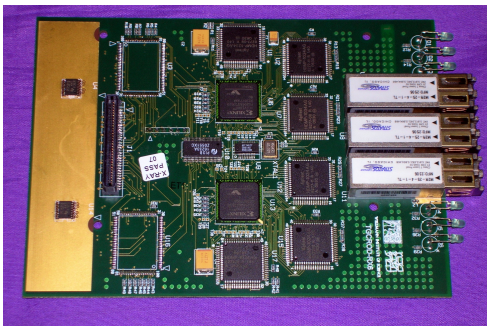


図 2.13: Front End メザニンカード [12]

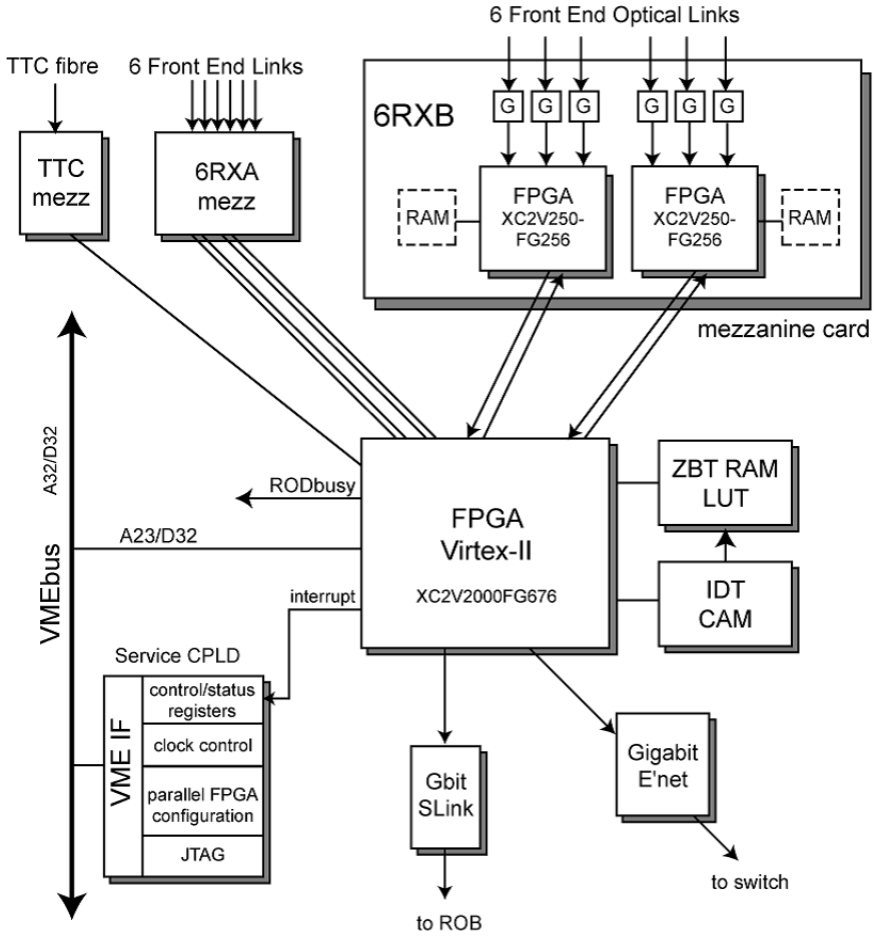


図 2.14: ROD の接続 [12]

出力フォーマット

ROD の出力フォーマットを図 2.15 に載せる。データフォーマットについては後の章で説明する。

	Data word				Comments
	31..24	23..16	15..8	7..0	
Frame	x'B0F0xxxx'				event frame word (control mode word)
Hdr0	x'EE1234EE'				start of header marker for ROD data
Hdr1	*reserved*	*reserved*	header size = 9		words (excluding the x'B0F0xxxx' word)
Hdr2	ATLAS format version=3.0		TGC format version=3.0		i.e.: ATLAS=0x03'00, TGC=0x03'00
Hdr3	0	x'67' or x'68'	0	sector[12..1]	source id: x'67' / x'68' = A / C endcap;
Hdr4	Run type	Run number			
Hdr5	Level-1 ID				High byte is Extended Level-1 ID
Hdr6	*reserved*	*reserved*	Bunch crossing ID[11..0]		
Hdr7	*reserved*	*reserved*	*reserved*	Trigger type	
Hdr8	Detector event type				not used yet
Status	First status word: specific | generic				≠0: event is **not** OK. See Table 2, & [ref. 1]
Status	TGC ROD event status				See Table 3.
Status	ROD VME filter bits		Star Switch timeout or dropped status		one bit per SSW; Filter:1 = accepted. SSW: 1 = dropped or timed-out (see Table 4)
Status	Local status word		presence		Presence indicates which of the following fragments are present[a]. See Tables 5 & 6.
Status	orbit count				orbit count; zero for first L1AID.[b]
Data	Fragment ID	"raw" data word count[c]			fragment ID =1, length in words
Data	Fragment ID	"readout format" hit data word count			fragment ID =2, length in words[d]
Data	Fragment ID	"readout format" tracklet data word count ("tracklet"= 3/4 or 2/3 coincidence)			fragment ID =3, length in words
Data	Fragment ID	"chamber format" hit data word count			fragment ID =4, length in words
Data	Fragment ID	"chamber format" tracklet data word count			fragment ID =5, length in words
Data	Fragment ID	HipT output word count			fragment ID =8, length in words
Data	Fragment ID	Sector Logic word count			fragment ID =9, length in words
Data	raw data, hit, tracklet, sector logic, etc. fragments, in the order of the word counts.				See [ref. 6] and[ref. 8](raw) and Tables 7 to 10.
Data	. . .				
Data	last raw data, hit or tracklet word				
Trail0	number of status elements = 5				
Trail1	number of data elements				
Trail2	Status block position = 0, i.e. data follows status				
Frame	x'E0F0xxxx'				event frame word (control mode word)

a. The number of fragment ID \| WC words and fragments is equal to the number of `H` i bits in this pattern.

b. 32 bits give >100 hrs

c. For debugging; not usually present

d. Fragment ID=6 is used for hits in testpulse format. Normally runs with this format are not sent to the ROB. In this case, ID=6 replaces ID=2 and the order will be 1,6,3,8,9, i.e. the fragments are not in order.

図 2.15: ROD の出力フォーマット [12]

第3章　High-luminosity LHCへ向けたRead Out Driver開発

この章ではアップグレードRODについて、その必要とされる性能とそれを満たすためにどのように設計していくかについて述べたいと思う。

3.1　新RODについて

まず最初に新RODのデザインについて説明する。現在は図3.1のようなものを考えている。

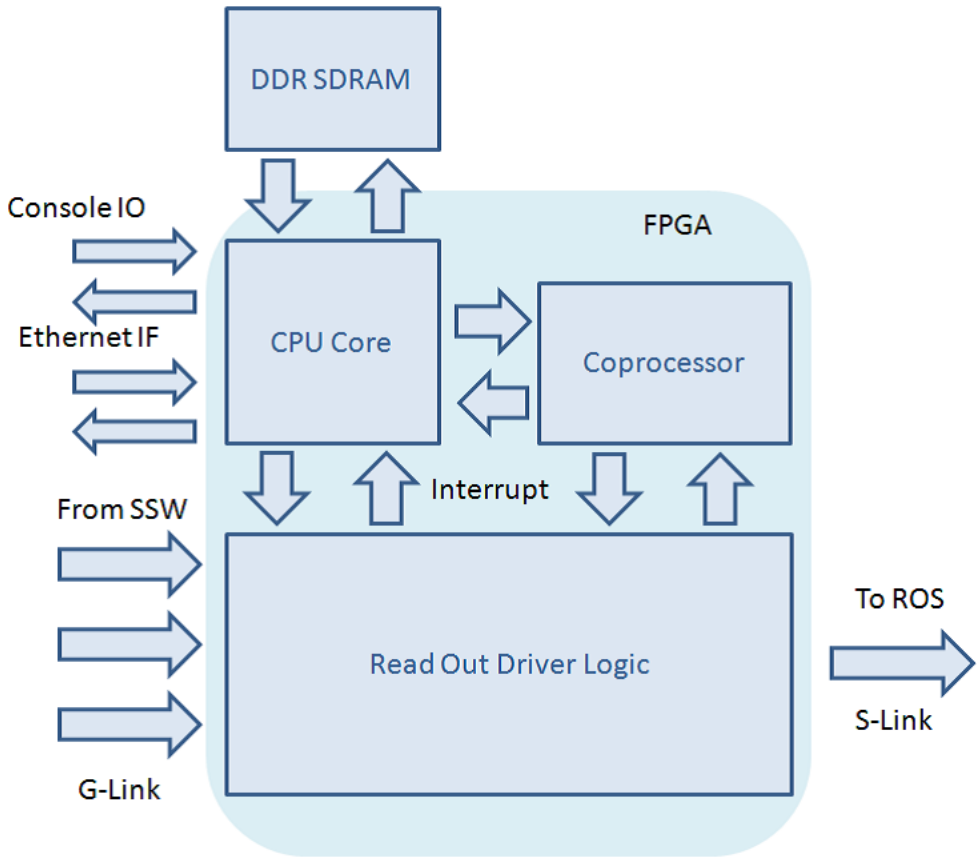


図 3.1: 新RODデザイン

以下にこのデザインの大きな特徴を述べる。

3.1.1 FPGA チップの使用

FPGA は Field Programmable Gate Array の略で、PLD(Programmabel Logic Device) の一種である。基本的な論理回路をあらかじめ敷き詰めておき、それをユーザが HDL(ハードウェア記述言語) を用いて自由に構成することができる。このように FPGA は内部要素だけが確保されており、その内部構成を変えることで高い柔軟性を確保している。この FPGA を用いる大きなメリットは、モジュールが完成した後でも変更が可能という点である。実験の初期におけるコミッショニングの段階で修正したい点があった場合やバグが発見された場合、通常の LSI では回路変更はすることができない。しかしながら FPGA を用いることで、設計を変更することができる。また実験が進んだ後に予定にない処理を追加したい場合でも、新しいモジュールを製作すること無しに後から回路変更ができるというのが大きな強みである。

3.1.2 CPU コア搭載

現在の ROD のデザインと新 ROD のデザインでもっとも大きな違いは、FPGA 内に CPU コアと同様の回路を作ることで、ソフトウェアを用いた処理がリアルタイムで行えるということである。現 ROD も PC 側から VME 経由でソフトウェアを用いた処理をすることができるが、通常はすべてハードウェア記述による処理を行っている。新 ROD はこのすべて HDL を用いて処理することの大きなデメリットを考慮して CPU コアを組み込むことにした。ソフトウェア、HDL で処理することのメリット、デメリットについては後述する。

⟶ CPU コアである MicroBlaze について (35 ページ)

⟶ HDL で処理することのメリット、デメリット (32 ページ)

⟶ CPU コアを搭載させることのメリット、デメリット (36 ページ)

⟶ 組み込みの必要性 (37 ページ)

3.1.3 Read Out Driver Logic

ROD のコアになる部分である。ここで各 SSW から来たデータを受け取り、ATLAS 共通のフォーマットに整形して、ROS へデータを送信する。ROS へのデータ出力フォーマットは、raw data format というものと、readout format というものが存在するが、現状では raw data format を用いて出力できる処理能力をこの新 ROD には求めるとする。このデータを受信し、出力する部分は回路を用いて処理する。そしてこの処理で起こるエラーやシステムの診断など複雑な処理はソフトウェアを用いて処理する。具体的なデザインは後述する。

⟶ raw data formt、readout format について (33 ページ)

⟶ Read Out Driver Logic の具体的な内容 (38 ページ)

3.2 アップグレード ROD の必要性

LHC がアップグレードするに従って、現行の ROD のデザインでは対処できない可能性が示唆されている。現 ROD に搭載されている FPGA は XC2V2000FG676 だが (図 3.2)、このチップは古く、現在はこれよりも数倍リソースがあるチップも発売されている (図 3.3)。また動作クロックも 1.5 倍ほど早く稼働できるようになっている。このように FPGA の発達と、新技術を導入することによって、現状の ROD が抱えている問題点を排除し、高速な処理が可能になる。まず現状の ROD の問題点について述べたいと思う。

デバイス	システム ゲート	CLB (1 CLB = 4 スライス = 最大 128 ビット) アレイ 行 X 列	CLB スライス	CLB 最大分散 RAM (Kb)	乗算 ブロック	SelectRAM ブロック 18Kb ブロック	SelectRAM ブロック 最大 RAM (Kb)	DCM	最大 I/O パッド(1)
XC2V2000	2M	56 x 48	10,752	336	56	56	1,008	8	624

図 3.2: 現 ROD 搭載 FPGA の機能

デバイス	ロジック セル	CLB (コンギャブル ロジック ブロック) スライス (1)	CLB 最大分散 RAM (Kb)	DSP48E1 スライス (2)	ブロック RAM ブロック 18Kb (3)	36Kb	最大 (Kb)	MMCM (4)	PCI Express 用インター フェイス ブロック	イーサ ネット MAC(5)	トランシーバ 最大数 GTX	GTH	総 I/O バンク (6))	最大 ユー ザー I/O(7)
XC6VLX550T	549,888	85,920	6,200	864	1,264	632	22,752	18	2	4	36	0	30	1200
XC6VLX760	758,784	118,560	8,280	864	1,440	720	25,920	18	0	0	0	0	30	1200
XC6VSX475T	476,160	74,400	7,640	2,016	2,128	1,064	38,304	18	2	4	36	0	21	840
XC6VHX565T	566,784	88,560	6,370	864	1,824	912	32,832	18	4	4	48	24	18	720

図 3.3: 最新の FPGA の機能

3.2.1 現状 ROD の問題点

ATLAS 実験は現在のように正常に稼働するまでさまざまな試行錯誤があった。ROD についてもさまざまな問題があったが、現在は概ね正常に動いているといってもいいだろう。しかし、現状も多くの問題を抱えている。具体的な内部デザインはイスラエルグループが担当しているため、日本グループに情報は入ってこないが以下のような処理が問題であることがわかっている。

- エラーなどの複雑な処理を含め、すべてを HDL 記述で行っている

この問題点を順を追って説明したいと思う。

HDL を用いた処理

HDL は FPGA 内部の配線をテキストで記述することができ、これにより開発効率を格段に良くすることができる。C 言語によく似た文法体系も存在し、ソフトウェアプログラム経験者でも馴染み易い。しかしながらその動作はソフトウェアとは大きく異なるものであり、ソフトウェアプログラムのように記述することはできない。またテキストで記述できるが実態は回路であり、処理もさまざまな論理回路の組み合わせによって行う。

従って、複雑な処理をさせるには深い専門的な知識が必要になってくる。また処理に不具合がありHDLを修正するということは、回路を組みかえるということである。単純な回路ならばさほど問題にはならないが、複雑な回路になればなるほど思わぬところでバグが出たり、またそのバグを修正するときに今まで問題無かった箇所でバグが発見されたりという難しい部分が顔を出してくる。もちろんこれはHDLに限ったことではないが、HDLを用いた処理ではより顕著に表れる。しかしながらこのHDLによる処理はソフトウェアによる処理に比べて格段に処理速度が速いというメリットも存在する。また同様の処理でもソフトウェアよりもHDLの方が簡単に記述できるというものも存在する。

すべての処理を HDL を用いて行う問題点

HDLによる処理のメリット、デメリットについては上記の通りである。RODは次々とデータが流れてくるため、そのデータの処理にある一定以上の速度が求められる。現RODはその速度を出すためにすべての処理をハードウェア記述で行っている。問題は複雑な処理が必要な部分を含め、すべてHDL記述で行っている点である。すべてをHDLで行っているためデザインが巨大化し、その回路構成はとても複雑なものになっていることが予想される。デザインが複雑になればなるほど多くのバグや、そのデバッグに悩まされることになる。現実にイスラエルグループのデザインは多くのバグに悩まされ、そのデバッグにも多くの時間が費やされてきた。またその複雑な回路をデザインしているHDLプログラムもかなり難解なものになっているだろう。ATLAS実験のように大規模な実験では実験開始から終わるまで１人の人が担当するというのは現実的ではないし、実際実験を計画し、各モジュールを製作してきた世代から次の世代に交代が始まっている。このように何代にも渡って実験は続けられていくのだが、人が書いたプログラムというものはソフトウェアによるプログラムでさえ理解するのに多くの時間がかかる。ましてこの分野におけるHDLを理解できるユーザは限られているため、HDLによる複雑なプログラムはこれからのことを考慮しても避けるべきである。

データフォーマット

RODから出力されるフォーマットは、raw data formatというものとreadout formatというものが存在する。

- raw data format

  SSWからのヒットデータそのまま。データサイズはreadout formatに比べ大きいが、RODは各SSWからのデータを集めて流すだけでよい。

- readout format

  イスラエルグループによる整形されたフォーマット。raw data formatに比べデータサイズがかなり小さくなるが、ROD内部でさまざまな処理が必要。

2010年11月までの運転では、2つのフォーマットが出力されていた。というのも、readout formatではデータが正しく出てこないという問題があったためである。しかし2011年からの実験ではreadout formatのみで出力する予定である。この理由として、現状ATLASのL1トリガーレートは50kHzで動作してきたが、これから当初の予定である75kHzで動くとこになるだろう。つまり、現RODでは75kHzのレートではraw data formatではサイズが大きすぎて処理できないのである。そこで、複雑な処理が必要になるがサイズの小さいreadout formatを用いることになった。2010年12月のテクニカルストップ時に、readout formatのテストが行われ、概ね正しく動くことが確認されたが、このフォーマットでどのレートまで対処できるかは不明である。

3.2.2 アップグレードに向けて

現 ROD でどのレートまで耐えれるか不明なため、アップグレードした後でも十分に処理できる可能性もありえる。しかしながら、L1 トリガーレートが 75kHz で動くかどうかも怪しい状態で、アップグレード後の 150kHz ではとても動作するとは思えない。従って、将来を見据え今のうちから研究開発する必要性は十分にあると言える。そこで我々日本グループはイスラエルグループとは別にアップグレードに向けて動いている。現 ROD のすべて HDL 記述による処理ではデザインの巨大化が起こり、これによりバグが見つかる頻度が上がり、また問題点の特定がしにくいなどデバッグも困難になる。さらにこのような記述方法は機能の拡張が難しいという問題も存在する。そこで HDL 部はモジュール化し、単機能化した各 HDL 部を CPU とソフトウェアを用いて動的制御を行うことで、デバッグが容易で拡張性が高いデザインとなる。また複雑な処理をソフトウェアに任せることで HDL デザイン部の負担を軽減することができる。

3.3 アップグレードで求められる性能

ではアップグレードによって、ROD にはどのような性能が求められるだろうか。これは、

- どれだけデータ量が増えるか。
- 何秒以内に処理しなければならないか。

という 2 つの要素で決まるだろう。この考察は現状のリードアウトラインのままアップグレードした場合を考えている。アップグレードではリードアウトラインの変更、SSW からのフォーマットの変更など考えられるが、現状よりもデータサイズが増える変更にはならないだろう。そこで何も変更なかったときを考える。これよりデータサイズが小さくなる変更ならば何も問題はない。それぞれについて考えてみる。

3.3.1 データ量増加について

ROD に来るデータがどれだけ増加するかということについて考える。まずヒットデータ以外の部分 (Header や trailer) は、SSW が 8byte 分、SLB が 8byte 分存在する。それぞれの SSW でどれだけのデータサイズになるかというのを表 3.1 に記載した。ヒットデータは、1 ヒットあたり 2byte である。例えば Current、Previous、

表 3.1: SSW からのデータサイズ

SSW	0	1	2	3	4	5	6	7	8	9
SLB 数 (個)	18	18	10	15	15	15	15	10	11 or 12	6
データ (byte)	152	152	88	128	128	128	128	88	96 or 104	56

Next にそれぞれ 1 ヒットずつあるならばヒットデータは 6byte になる。シミュレーション結果から、ルミノシティ $10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ のとき、TGC の 1 セクターにトラック数が 10 本と見積もった。このときのヒットデータは 80byte で、SSW から ROD にくるデータは 1232byte になる。アップグレードでルミノシティが $5\times10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ になったとき、陽子衝突が 5 倍起きやすくなると考え、単純に 10^{34} のときの 5 倍ヒットがあるとして計算するとヒットデータは 400byte。従って計 1552byte となり、アップグレード後は 1.26 倍になることがわかった。

3.3.2 処理時間について

次に L1 トリガーレートの上昇による処理時間について考えてみる。現状の ATLAS 実験の最大 L1 トリガーレートは 75kHz である。これがアップグレードによって 150kHz になったとすると、トリガーレートが 2 倍になったので、処理時間は 2 分の 1 になる。つまり現在の 2 倍の処理速度が求められる。具体的な数値としては、75kHz では平均データ到着時間は 13.2μs、150kHz では 6.7μs である。あくまで平均なので、最短では、75ns で次のデータが来る場合もありえる。

まず SSW 側からの転送速度を考える。SSW からのデータ転送には Glink という規格が用いられているが、この Glink の転送速度は 80Mbyte/s である。これは一度に転送できるデータ量が 2byte で、40MHz で転送していることから導かれる。ROD は 10 個の SSW からデータを受け取るので、実質転送速度は最大 800MHz になるが、実際は常にデータが転送されているわけではないので、平均の転送速度はこの値よりも小さいはずである。平均の転送速度は、L1 トリガーレートと、1 バンチあたりの SSW からのデータ量から決まり、

$$\lambda = (\text{L1 trigger rate}) \times (\text{SSW data /バンチ}) \tag{3.1}$$

で与えられる。L1 トリガーレートを 150kHz、SSW からのデータを 1552byte とすれば、λ は 230Mbyte/s となる。一方送信部は、Slink[13] という規格を用いて ROD から ROS へと送られる。この Slink の現状の転送速度は最大 125Mbyte/s だが、最新のものは最大 250Mbyte/s である。これからさらに高性能の Slink が開発される可能性もある。

では具体的に M/M/1(N) 待ち行列理論を用いて必要とされる処理速度を見積もってみたいと思う。単位時間あたりに到着するデータ量の平均を λ で表し、単位時間あたりに処理されるデータ量を μ で表す。ROD を深さ N のバッファと考え、データの到着はポアソン分布に従い、データの処理は指数分布に従うと仮定する。このとき ρ を

$$\rho = \frac{\lambda}{\mu} \tag{3.2}$$

と定義すれば、SSW から来たデータがバッファに収まらず失われる確率は、

$$P_{\text{Lost Event}} = \frac{(1-\rho)\rho^N}{1-\rho^{N+1}} \tag{3.3}$$

で表すことができる。しかしこの式で求められる値自体にあまり意味はない。この式における変数は ρ と N であるが、処理速度を決めるのは ρ であり、以下この ρ についてのみ考える。

Slink の転送速度は最大 250Mbyte/s だが、まず slink の転送速度を考慮せずに処理速度を考えよう。従って処理速度はデータが SSW から来て、それを ATLAS 共通のフォーマットに変形して Slink で出力するまでの時間を表す。データの単位時間当たりの平均サイズを 230Mbyte/s とすると、ρ の値を定めてしまえばそれに必要な処理速度がおのずと定まる。イスラエルグループは現 ROD のシミュレーションを $\rho = 0.7$ として行ってきた。仮に 0.7 とすれば処理速度は、

$$\mu = \frac{230}{0.7} \approx 330\text{Mbyte/s}$$

である。これは 1 イベント分のデータを 4.7μs で処理しなければならないことを意味する。しかし現状の ROD が高レートに対処できないことを考慮して ρ が 0.5 以下であるのが望ましい。このとき処理速度は、

$$\mu = \frac{230}{0.5} = 460\text{Mbyte/s}$$

となり、これは 3.37μs 以下で処理しなければならないことを表す。従って新 ROD はこの 3.37μs 以内を目標にして開発していく。しかしこの処理速度は Slink の転送速度を超えているので、データ送信の際はなにかしらの工夫が必要である。

3.4 MicroBlazeとそのメリットについて

アップグレード ROD には、Xilinx 社製ソフト CPU コアである MicroBlaze を搭載させることを考えている。ここでは、MicroBlaze の性能及び搭載によるメリットについて述べていきたいと思う。

3.4.1 CPU コアを搭載することの利点

集積回路の代表格である CPU は FPGA とはまた別次元の柔軟性を持ち合わせている。CPU は内部回路そのものに柔軟性はないが、その上でソフトウェアを動作させることができ、これにより柔軟性を確保するものである。FPGA に搭載できる CPU は一般の商用コンピュータに搭載されているような CPU と比べて動作クロックに大きな性能の差があり、とても低速であるが、それでもソフトウェアを用いることができるメリットはそのデメリットを大きく上回るものである。ソフトウェアを用いる一番の大きなメリットはデバックの容易さである。ソフトウェアはプログラムの上から下に順に処理される。従って、どこに問題があるか比較的発見しやすい。ところがハードウェア記述ではそうはいかない。HDL プログラムを見てデバッグするのだが、ソフトウェアのように上から順ではなく、すべて並行に処理される。従って、どこに問題があるのか把握するのにとても時間がかかる。また回路による処理とソフトウェアによる処理は仕組みがことなるためにソフトウェアのプログラムを書き換えたところで回路に影響を与えることはない。MicroBlaze 自体は HDL デザインによって生成されるので、その MicroBlaze を動かしている以上、全くないとは言えないが、ほぼ無いといえるだろう。逆に、すべて HDL で記述している場合はエラー処理に関する部分だけを修正したつもりでも、他にどのような影響がでるかはわからない。

以上のような理由により、ソフトウェアによる処理はハードウェア記述による処理に比べて低速であるというデメリットも存在するが、ソフトウェアを用いることのメリットから CPU コアを搭載することにした。

3.4.2 MicroBlaze

MicroBlaze は、Xilinx 社製 FPGA に構築できるソフトプロセッサコアである。32bitRISC 型アーキテクチャで、データ記述はビッグエンディアン形式である。図 3.4 に MicroBlaze のコアブロック図を載せる。

また MicroBlaze は、ユーザの用途によって

- 動作クロック数
- パイプラインの段数 (3 or 5)
- 周辺インターフェース
- バスインターフェース
- メモリ管理ユニット

の要素を設定することができる。これらは Xilinx 社が提供する組み込みプロセッサ開発環境ツールである EDK(Embedded Development Kit) によって、GUI で簡単に操作することができる。

3.4.3 MicroBlazeのメリット

- インターフェースを自由に接続することができる

  MicroBlaze を使うメリットとしてインターフェースを自由に変更できる点を挙げることができる。Xilinx 社が提供するツールを用いることによって、専門的な知識が無くてもハードウェアとソフトウェアの I/O 部分を簡単に連結することができる。

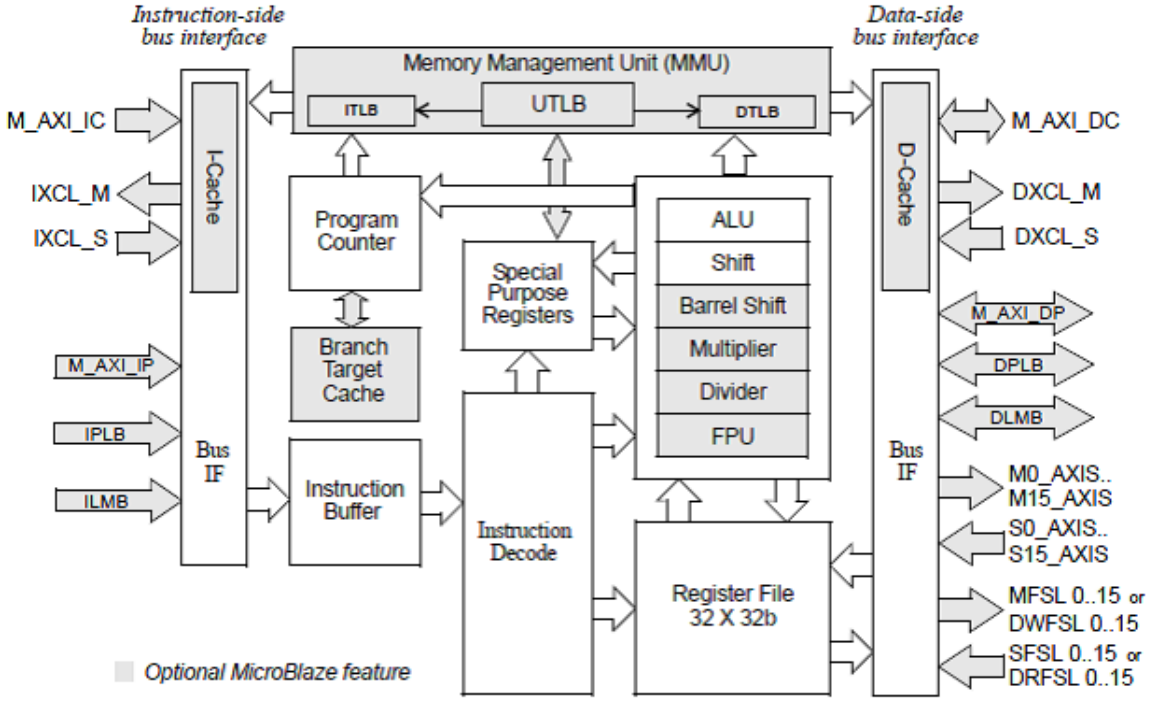


図 3.4: MicroBlaze コアブロック図

サンプルデザインとして、図 3.5 のデザインを用いる。色分けしてあるが、青文字は HDL を用いて記述した部分で、緑文字は MicroBlaze を用いてソフトウェアで管理できるようになっている。MicroBlaze からバスを通じて各インターフェースに配線されているのがわかると思う。各インターフェースは HDL で書かれたものであるが、ソフトウェアを用いて制御することができる。また配線も自由にできる。

- FPGA の容量が許す限り何個でも CPU を設置できる

  近年の FPGA の大容量、高密度集積化に伴い、使用できるリソースが格段に多くなった。これにより並列に CPU を設置していき、１つの処理に１つの CPU を用いることもできる。これにより MicroBlaze の動作周波数の低さをカバーすることができる。

以上のように設計のし易さ、FPGA の持つ柔軟さを引き出すために MicroBlaze を用いることを検討している。しかしながらこのような FPGA 内に CPU 回路を組み込むことは若干の違和感を抱くことがあるかもしれない。なぜなら FPGA 内にわざわざ CPU を載せなくても、FPGA と CPU を別々に載せることで FPGA と CPU のそれぞれの柔軟性を確保できるからである。しかしながらこれには、金銭面の問題、基板の広さの問題などあるが、実際どのようにするかについてはこれからの議論によるだろう。

3.4.4 組み込みの利点

ROD は VME モジュールであるため、FPGA 内に CPU コアを搭載しなくても VME を経由してコンピュータからソフトウェアによる処理をすることができる。そこで敢えて組み込むことのメリットについて考えてみたい。

先ほども述べたが、商用コンピュータに搭載されている汎用 CPU と MicroBlaze には大きな性能の差がある。表 3.2 に空の for loop を 1000 万回処理したときに要する時間を示した。今回 MicroBlaze の動作周波数は 50MHz を用いたが、Spartan6 では最大でも 250MHz 程度が限界である。処理が終わるのに要する時間で約 150

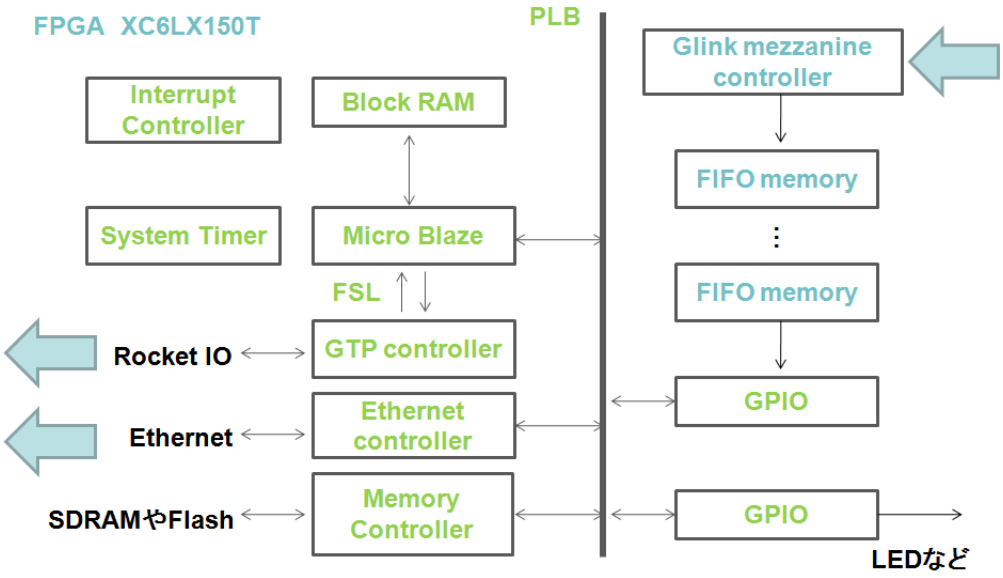


図 3.5: MicroBlaze から各インターフェースへのアクセス

表 3.2: for loop を一千万回行ったときの処理時間

	時間	クロック回数	1 ループあたり
MicroBlaze (50MHz)	2.6s	1 億 3000 万	13
Pentium4 (2.4GHz)	16.8ms	4000 万	4

倍差がある。これは動作周波数で約 50 倍、1 ループあたりにかかるクロック数で約 3 倍の性能差からくることがわかる。このように for による処理や、演算などによる処理は汎用 CPU の方がはるかに早く処理が終わる。

次に MicroBlaze によるオンボード上のメモリへのアクセスと、VME クレートのスレーブモジュール上に搭載されているメモリへの商用コンピュータからのアクセスにおける処理時間を表 3.3 に示す。このように動

表 3.3: ボード上のメモリへのアクセスにかかる時間

	時間	クロック回数
MicroBlaze (50MHz)	0.94μs	47
Pentium4 (2.4GHz)	4.97μs	(11200)

作周波数に 50 倍の差があっても、距離的に近い MicroBlaze の方が約 5 倍の早さで処理できることがわかる。従って、オンボード上のメモリや周辺装置を操作する場合などは MicroBlaze を用いて、組み込むことに大きなメリットがあることがわかる

3.5 新 ROD のデザイン

新 ROD のデザインは、HDL 部とソフトウェアプログラム部の 2 つに分けることができる。HDL 部である Read Out Driver Logic はある程度完成しているが、ソフトウェアプログラムは未だ MicroBlaze の性能評価の

段階である。ここでは Read Out Driver Logic の詳細を述べたいと思う。

3.5.1 Read Out Driver Logic

Read Out Driver Logic のブロック図を図??に載せる。

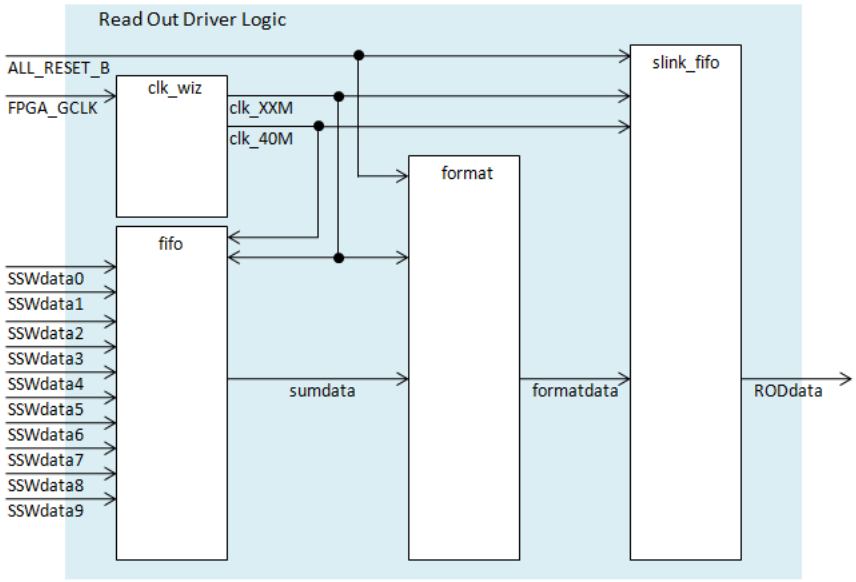


図 3.6: Read Out Driver Logic のブロック図

このロジックの各ブロックについて簡単に説明する。

clk_wiz

この Clock Wizard は xilinx 社が提供する IP コアを用いている。これは供給クロックを何倍かしてさまざまな周波数のクロックを生成することができる。供給クロックは ATLAS 検出器で供給される 40MHz。ここで SSW からの信号及び ROS への信号に同期させるため 40.08MHz のクロックと、またこの FPGA 内で用いるために X 倍したクロックを生成している。

fifo

このブロックでは SSW からのデータをまず FIFO に溜め、それを順に読み出し、パラレルのデータを 1 つのシリアルデータにする。FIFO バッファには、SSW からのデータとコントロールワードの 18bit が保存される。書き込み用のクロックは 40.08MHz だが、読み出し用のクロックは clk_wiz で X 倍したものを用い、高速処理を行う。FIFO バッファは Xilinx 社提供の IP コアを用いた。FIFO バッファについては後述する。

format

fifo ブロックにてシリアルにされたデータに図 2.15 に記載されているような各情報を含んだ header と trailer をつける作業を行う。

slink_fifo

format で成形されたデータを FIFO バッファに溜め、Slink によって送信される。この FIFO バッファは書き込みは clk_wiz で X 倍したクロックで 18bit 幅で保存され、読み出しは 40MHz で 36bit で行われる。Slink の送信は 34bit である。これは Glink のコントロールワードが 2bit あり、2 データ読みだされるので 4bit 分のコントロールワードが存在することになるのだが、Slink のコントロールワードは 2bit で良いので、2bit は使わなくてよい。

第4章　検証実験

新 ROD の開発のために、汎用モジュール PT5 を用いて、今回作成した PT6 のテスト、及び Spartan6 評価ボードを用いてさまざまなテストを行った。Spartan6 評価ボードでは MicroBlaze を用いてどのようなことができるのかということを検証した。まず使用した機器についての説明をし、行ったテストについて記述していく。

4.1　PT5、PT6、Spartan6 評価ボード

まず今回用いた機器について簡単に説明しておく。

4.1.1　PT5

PT5 とは、FPGA を搭載した VME 汎用モジュールで、LHC に於ける ATLAS 実験において、 エレクトロニクスのテストベンチを構成するために開発された。図 4.1 に PT5 の概要、図 4.2 に PT5 を載せる。

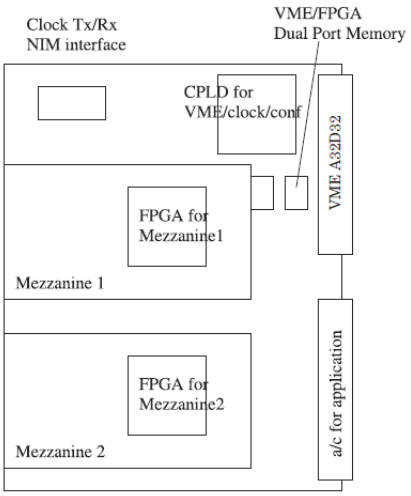


図 4.1: PT5 の概要

図 4.2: PT5

搭載されている CPLD は XC2C256PQ208、FPGA は XC3S400FGG320 でメザニンカードのスロットが 2 つあることに特徴がある。この新 ROD 開発研究では PT5 を ROD の前段階モジュールである SSW と見立て、Glink メザニンカードを用いてデータを送る役割をさせる。

4.1.2　PT6

PT6 は PT5 の後継機であり、TGC の読み出しシステムのアップグレード機開発のために制作された VME モジュールである。図 4.3 に概略図、図 4.4 に PT6 の写真を載せる。CPLD は PT5 と同様の XC2C256PQ208

を用いているが、FPGA は XC6SLX150TFGG676 と現状の FPGA では最新の Spartan6 を用い、規模も最大のものを用いている。この Spartan6、XC6SLX150TFGG676 の特徴は、そのロジックセル数の多さと GTP トランシーバが最大 8 個使えること、さらにそのブロック RAM の多さである。ブロック RAM はプリミティブとして用意されているデュアルポートメモリで、FPGA の CLB を消費することなく使用することができる。このブロック RAM は FIFO バッファなどで使用することが多い。また MicroBlaze のソフトウェアプログラムを保存するためのローカルメモリとしても使用される。PT5 に搭載されている XC3S400FGG320 はこのブロック RAM の容量が少なく、MicroBlaze を用いてサイズの大きいプログラムを稼働させることができなかったが、PT6 に搭載されている XC6SLX150TFGG676 は特に容量を気にすることなく使用することができる (図 4.5)。そのために容量が許す限り、MicroBlaze をいくつも作成して並列に動かすことで、マルチコアとして使うこともできる。

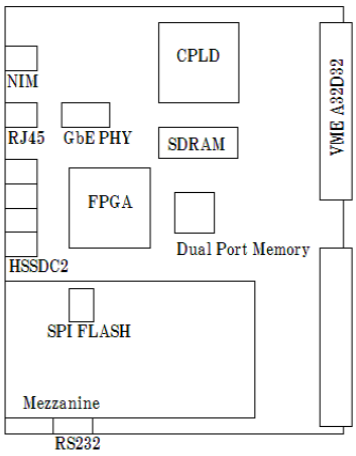


図 4.3: PT6 の概要

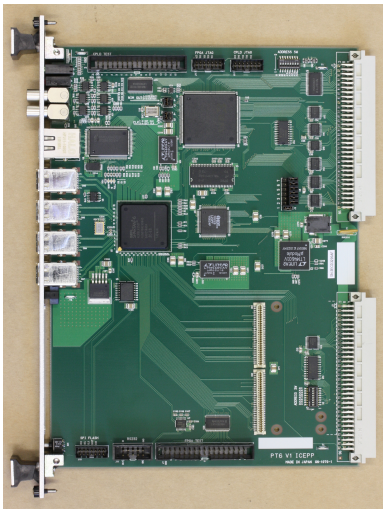


図 4.4: PT6

デバイス	ロジックセル	ブロック RAM ブロック		最大 GTP トランシーバ	最大ユーザー I/O
		18Kb	最大 (Kb)		
XC6SLX150T	147,443	268	4,824	8	540
XC3S400	8,064	16	288	0	264

図 4.5: PT5 と PT6 の FPGA の機能 [21]

また PT6 上には SDRAM、FLASH メモリーが搭載されているので、MicroBlaze を用いることで OS を搭載することができるのだが、今回はそこまで触れなかった。

4.1.3 Spartan6 評価ボード

Spartan6 評価ボードは、Xilinx 社が制作した Spartan6 FPGA の評価用ボードである。このテストでは SP605 という型を用いた。この評価ボードはこのボードだけでさまざまな開発ができるように、機能も充実している。

搭載されている FPGA は、XC6SLX45TFGG484 であり、PT6 に搭載されている FPGA よりは若干規模が小さいものだが、それでも特に容量を気にすることなく設計できる。図 4.6 に写真を、図 4.7 に PT6 との FPGA の規模の違いを示した。

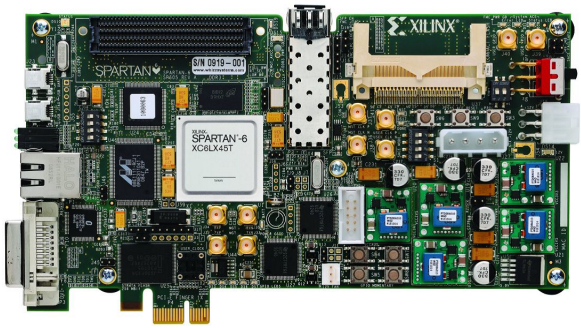


図 4.6: Spartan6 評価ボード [14]

デバイス	ロジック セル(1)	コンフィギャブル ロジック ブロック (CLB)			DSP48A1 スライス (3)	ブロック RAM ブロック		CMT(5)	メモリ コントローラ ブロック (最大数)(6)	PCI Express 用エンドポイント ブロック	最大 GTP トランシーバ	合計 I/O バンク	最大 ユーザー I/O
		スライス (2)	フリップフロップ	最大 分散 RAM (Kb)		18Kb(4)	最大 (Kb)						
XC6SLX45T	43,661	6,822	54,576	401	58	116	2,088	4	2	1	4	4	296
XC6SLX150T	147,443	23,038	184,304	1,355	180	268	4,824	6	4	1	8	6	540

図 4.7: PT6 との規模の違い [14]

4.2 シミュレーション

現 ROD は 1 つにつき 10 個の SSW からデータを受けている。汎用モジュールではテストができないので、シミュレーションでアップグレード ROD に必要な性能を満たすよう設定していきたいと思う。用いたシミュレーションソフトは Xilinx ISim[16] を用いた。

今回行ったシミュレーションは、論理シミュレーションであり、配置配線行い、信号の遅延を考慮したタイミングシミュレーションではないので、論理シミュレーションの結果のように正しくデータが出てくるとは限らない。しかしこのシミュレーションではアップグレードに必要な処理速度を備えているかということに関して調べているので、タイミングシミュレーションのような正確さはないが、論理シミュレーションでも十分目安にはなるだろう。表 4.1 は、FIFO から Slink で出力するまでにかかる時間をクロック数を変えて測定したも

表 4.1: 処理時間

	40MHz	80MHz	120MHz	160MHz	200MHz
FIFO から Slink 出力まで	863ns	488ns	363ns	313ns	263ns

のである。用いたデータはほとんどがフレームでデータは数ヒットである。この FIFO から Slink で出力するまでのデータ整形に関する部分は 40MHz でも約 900ns、200MHz では約 300ns で処理が終わる。しかしこれ以外の部分である SSW からのデータを FIFO で受け取る部分、Slink による出力は、前後のモジュールに同期

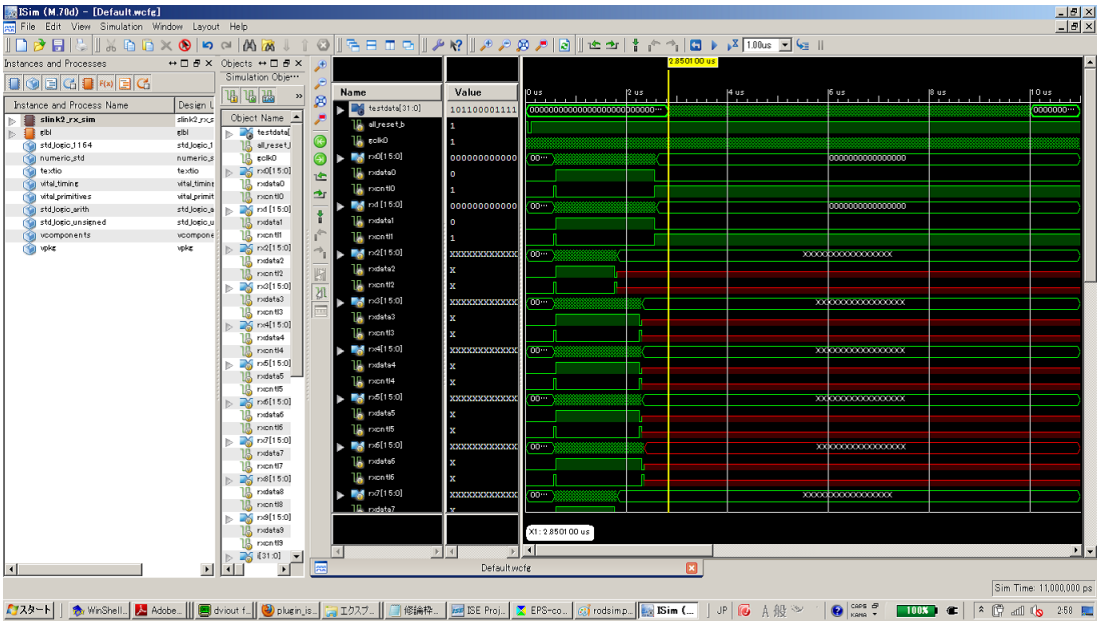


図 4.8: シミュレーション画面

させるために 40MHz で動作させなければならない。そしてこの部分がもっとも時間がかかる。例えば、もっとも SLB 数の多い SSW は SSW0、SSW1 であるが、このデータを受け取り、FIFO から出力するまでに 2.5μs 以上かかる。

アップグレードにあたって、slink で出力するまでを 3.37μs 以内にすることが目標であった。シミュレーションで用いたデータはアップグレード後のデータサイズではないので、1.26 倍する (3 章でアップグレード後のデータサイズが現状の 1.26 倍になることは説明した)。このとき 3.37μs 以内にするには、clk_wiz で生成するクロックを 120MHz 以上にすればよい。

4.3 MicroBlaze の性能評価

MicroBlaze を用いてどのようなことができるのか、またどのような性能を持っているのか検証してみた。この性能評価は Spartan6 評価ボードを用いて行った。

4.3.1 テスト 1：FIFO バッファからの読み出し

新 ROD では MicroBlaze にエラー処理やシステム診断などを行わせたいと考えている。もし ISE で生成した FIFO バッファの値を MicroBlaze で読み出すことができればさまざまなことに活用できるだろう。そこでまず FIFO バッファの値を MicroBlaze 側で読み出すというテストを行った。FIFO バッファはすべてのクロックを統一するものも存在するが、今回使用した FIFO バッファは図 4.9 のような書き込み用のクロックと読み出し用のクロックが独立しているものを用いた。

この HDL デザインは、ISE 側で FIFO バッファを作成し、そこに 1 から順にカウントアップしていくデータを保存する。そのデータを MicroBlaze を用いて読み出し、RS232 コネクタから Tera Term 上にデータを出力するというものである。

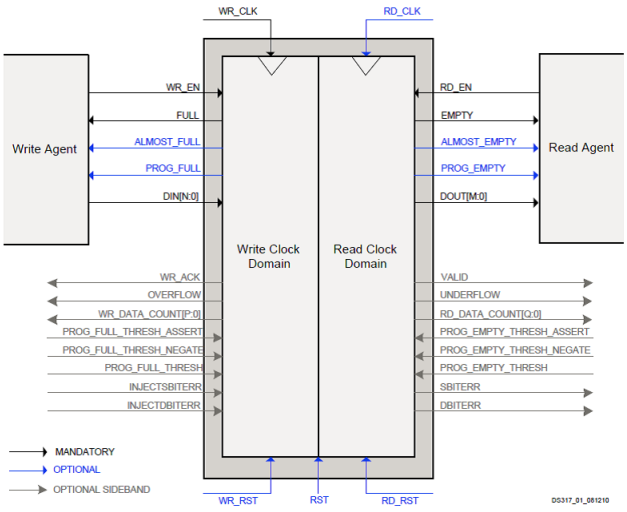


図 4.9: 使用した FIFO バッファ[17]

MicroBlaze と FIFO バッファの接続

まず始めに MicroBlaze と FIFO バッファとの接続であるが、これは General Purpose IO という IP コアを用いた。この IP コアを用いて FIFO バッファの DOUT 線と、RD_EN 線を接続する。MicroBlaze の動作クロックは Clock Generaror によって入力クロックを定数倍した値を生成できる。評価ボードは 200MHz のオシレーターが搭載されており、このクロックを用いて生成が、このオシレーターで生成でき、かつ MicroBlaze が動作するクロック周波数帯は、4MHz から 230MHz であることがわかった。Clock Generator の output 線はデフォルトの状態では MicroBlaze のクロックのみだが、これを増やして MicroBlaze の動作クロックとは別に使用することもできる。

データの読み出し方法

このテストでは書き込みに関してはすべて HDL デザインに任せるが、読み出しに関しては MicroBlaze で操作する。つまりデータをプルする。この方法のメリットはソフト側ですべて操作できる点である。従って、データを読み出すタイミングもソフトで制御できる。Appendix にプログラムのコードを置いたが、GPIO の操作には、

- XGpio_DiscreteWrite()
- XGpio_DiscreteRead()

という特殊な関数を用いる。まず関数のオーバーヘッドを計測してみた。

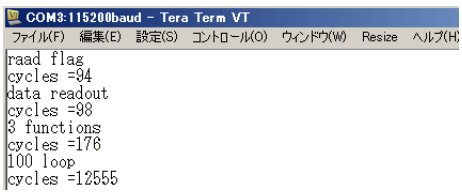


図 4.10: 関数のオーバーヘッドの計測

データの書き込みには 94 クロック、読み込みには 98 クロック、この GPIO を操作する関数は 95 クロック前後で行われる。またこの MicroBlaze のパイプラインの段数は 3 に設定してあるので、3 つの関数の操作は 1 つの関数の操作にかかるクロック数の足し算ではなく、それよりも早く処理を終えることができる。ちなみに RD_EN を High にし、データを読み込み、RD_EN を Low にする処理にかかるクロック数は 176 であった。また for loop を用いて 100 回、このデータ読み込みのサイクルを繰り返したところ 12555 クロックかかったので、1 つのデータ読み込みに約 126 クロックで終えることができることがわかった。従って、このデータをプルする方法は MicroBlaze の動作クロックを 230MHz で動かした場合、連続的にデータを取り出す場合は 7.3Mbyte/s での速度でデータを通信することができる。Vertex6 を用いた場合、最大 600MHz で動作可能なので 19Mbyte/s という通信速度でやり取りが可能である。MicroBlaze の動作クロック 230MHz で正しくデータが得られることを確認した (図 4.11)。

図 4.11: テスト 1 出力結果

4.3.2 テスト 2：データをプッシュする

次にテスト 1 とは異なり、HDL デザイン側がデータをプッシュしてくるのをタイミングよく MicroBlaze 側で受け取れるかを試してみた。この HDL デザインは、テスト 1 のデザインとほぼ変わらないが、HDL デザイン側と MicroBlaze での処理によるタイミングを合わせるためにハンドシェイクを行い、上手くタイミングを合わせることで図 4.12 のように正しくデータが得られることを確認した。

図 4.12: テスト 2 出力結果

データを for loop で 100 回取り出すときにかかるクロック数は図 4.13 のように 17256 かかったので、1 つのデータ取得に約 173 クロックかかっていることがわかる。従ってデータをプッシュして MicroBlaze で受け取る方法は、230MHz で動かしたときは 5.3Mbyte/s で、600MHz では 14Mbyte/s で通信できることがわかった。

図 4.13: クロック数の計測

4.3.3 テスト 3：HDL デザイン側からの信号に応答するまでのオーバーヘッドの計測

HDL デザイン側のある線の値をトリガーとして、MicroBlaze がその値を受け取り、返答するまでに何クロックかかるかをこのテストで計測してみた。これを応用すれば HDL デザイン側でエラービットを立てて、MicroBlaze でなにかしらの処理をさせた後に値を返すことなどに使えるだろう。もちろん MicroBlaze で何を処理させるかによって、返答にかかるクロック数は依存するが、データを受け取り、返答するだけで何クロックかかるかをここでは計測してみた。MicroBlaze の動作クロックは 200MHz を用い、HDL デザイン側で Flag が High になってから信号が返ってくるまでの時間を計測した。その結果 91 クロック必要であることがわかった。今回のテストでは回路側のクロックは MicroBlaze の動作クロックと同じ 200MHz を用いたので、Flag を読み込み、返答するのに 460ns 必要なことがわかる。

4.3.4 テスト 4：MicroBlaze を用いたメモリへのアクセス

MicroBlaze を用いたメモリへのアクセスは前章で触れているが、オンボード上のメモリへのアクセスは VME 経由よりも高速に行える。Spartan6 評価ボード上には 128MB、DDR3 コンポーネントメモリが搭載されており、IP コアでメモリコントローラーも提供されている。これを組み込むことで簡単にアクセスが可能となっている。

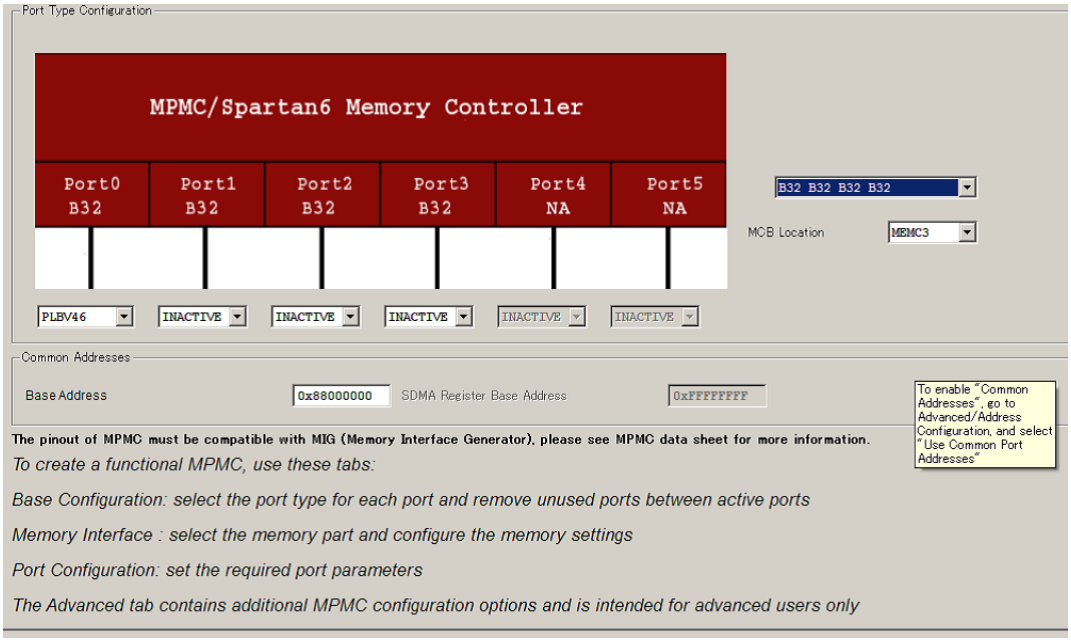
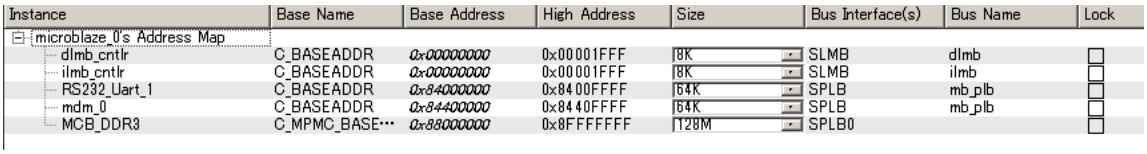


図 4.14: IP コアのコンフィグレーション画面

図 4.14 ように GUI 上でメモリの種類、バスの設定、データ線の bit 幅を変えることができる。このメモリコントローラーは Xilinx 社製 FPGA ならば対応しているが、FPGA の種類によっては使用できないメモリも

存在する。例えばこの Spartan6 は、LPDDR、DDR、DDR2、DDR3 には対応しているが、SDRAM には対応していない。従って、PT6 の SDRAM へのアクセスはこの IP コアを用いることができなかった。

このメモリの操作プログラムはいたって簡単である。EDK には図 4.15 のように各インターフェースに自動でアドレスを振ってくれる機能がある。あとはポインタを作成し、そのポインタのアドレスをメモリのアドレスにすれば、そのポインタに格納される値はメモリに書き込まれることとなる。GPIO の操作などのように専用の関数を使わなくてよいので非常に使い勝手がよい。



Instance	Base Name	Base Address	High Address	Size	Bus Interface(s)	Bus Name	Lock
microblaze_0's Address Map							
dlmb_cntlr	C_BASEADDR	0x00000000	0x00001FFF	8K	SLMB	dlmb	
ilmb_cntlr	C_BASEADDR	0x00000000	0x00001FFF	8K	SLMB	ilmb	
RS232_Uart_1	C_BASEADDR	0x84000000	0x84000FFF	64K	SPLB	mb_plb	
mdm_0	C_BASEADDR	0x84400000	0x8440FFFF	64K	SPLB	mb_plb	
MCB_DDR3	C_MPMC_BASE…	0x80000000	0x87FFFFFF	128M	SPLB0		

図 4.15: MicroBlaze のアドレスマップ

4.3.5 テスト 5：デュアルコアを用いたメモリへのアクセス

Spartan6 評価ボード上の DDR3 をデュアルコアの共通メモリとして片方の MicroBlaze で書き込んだ値を別の MicroBlaze で読み込むという作業を行った。MicroBlaze を複数個作成し、マルチコアで作業させるとき図 4.16 のような共通メモリが必要だろう。

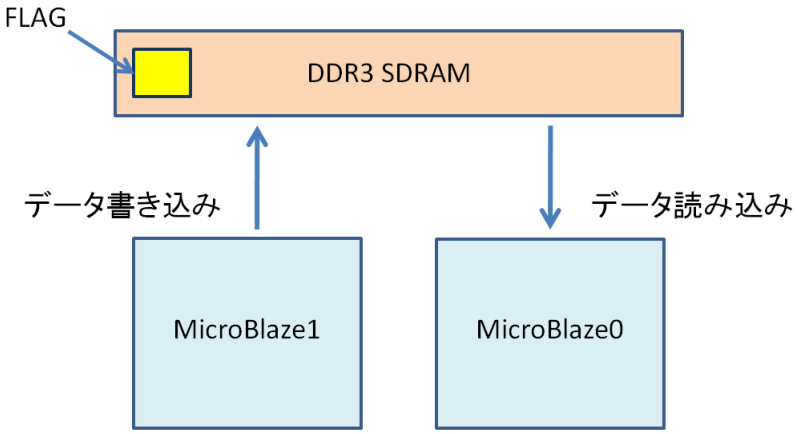


図 4.16: 共有メモリへのアクセス

メモリ上に書き込みと読み込みを制御する flag を作り、この値によって書き込みと読み込みが同時に行われることを防いだ。ここで重要なのが flag に volatile 修飾子をつけ、余計な最適化を防ぐことである。このようにマルチコアで共通の変数を使用する場合などは、コンパイル時に最適化によって本来意図した動作と異なることになることがあるので、注意が必要である。

またオンボード上のメモリだけでなく、FPGA 内に用意されているブロック RAM を用いても同様のことが行える。図 4.17 は新 ROD のテストのために作成した SSW エミュレータのブロック図である。MicroBlaze1 で SLB のデータを作成し、そのデータを共通ブロック RAM に保存、MicroBlaze0 でデータを取り出し、SSW のフォーマットに整形するものである。このようにボード上にメモリが存在しなくても共通のメモリは作成することが可能だが、DDR3 などのメモリコントローラとは違い、メモリへの書き込み、読み込みに専用の関数を用いなくてはならない。

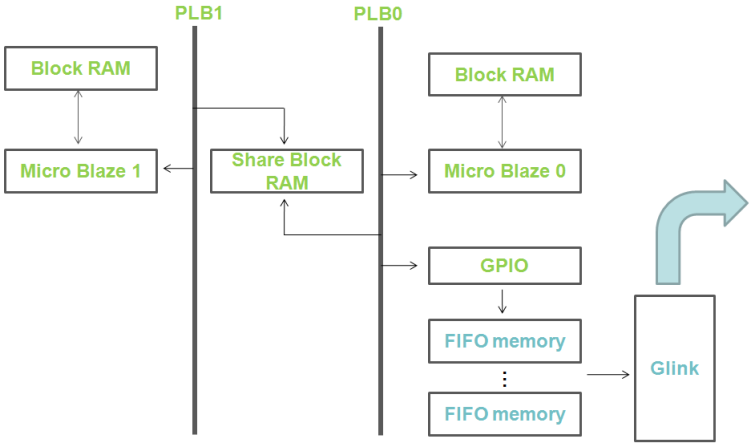


図 4.17: SSWemu のブロック図

またこのデザインで、シングルコアですべて処理させる場合と、デュアルコアで処理させる場合にどれだけ処理速度がことなるかを調べた結果が表 4.2 である。この際、MicroBlaze の動作クロックは 66.67MHz を用

表 4.2: デュアルコアのテスト結果

	処理時間	総クロック数
シングルコア	22ms	1478 万
デュアルコア	19ms	1290 万

いた。今回作成したプログラムは単純なものなので大した差は見られないが、それでも 14%の速度上昇が見られた。

テスト 6：PT6 を用いた Tera Term 上への出力

このテスト以降は Spartan6 評価ボードではなく PT6 を用いての動作テストを行った。PT6 も RS232 コネクタを用いて、データを Tera Term 上へ出力することができる。しかし PT6 はピンがむき出しの状態なので、PT6 のピンと D-SUB9 ピンコネクタと接続してあげなければならない。各ピンとの接続は図 4.18 に記載する。

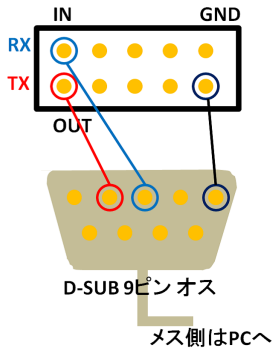


図 4.18: コネクタとのピン接続

このように接続し、RS232IP コアを用いることで出力できることを確認した。

テスト 7：PT5 を用いた Glink 出力

ROD は SSW からのデータを Glink 光通信で行う。そこで PT5 と PT6 を用いて、Glink の送受信テストを行った。将来的には SSW のデータを PT5 から送り、PT6 で受け取るということを行いたいのだが、今回は PT5 で作成したデータを PT6 で受け取るだけに止めた。テストの結果正しくデータが送受信されていることを確認した。図 4.19 に Glink メザニンカードと、図 4.20 にテスト風景を載せる。

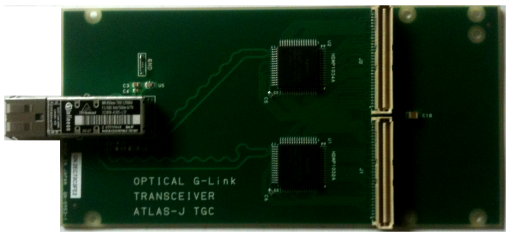


図 4.19: Glink メザニンカード

図 4.20: テスト風景

テスト 8：PT6 上の SDRAM へのアクセス

メモリへのアクセスはテスト 4 で行ったが、IP コアであるメモリコントローラーが Spartan6 では SDRAM は対応していない。そのために渡部謹二氏が作成した SDRAM コントローラー [15] を用いて、MicroBlaze で SDRAM を操作するデザインを作成し、テストを行った。図 4.21 が渡部氏が作成した SDRAM コントローラーである。これを MicroBlaze で操作するために図 4.22 のように加筆した。systemflag、readflag、address、data などの線は GPIO を用いて HDL デザイン側とやり取りをするようになっている。つまりソフトウェア側でアドレス、データ値を設定できる。また systemflag、readflag でデータの書き込み、読み込みのタイミングを設定する。このデザインを用いて、SDRAM の全アドレスにアクセスし、書き込み、読み込みができることを確認した。ただし、リフレッシュやプリチャージの設定をまだ組み込んでいないので、書き込んだ値を保持することはできない。

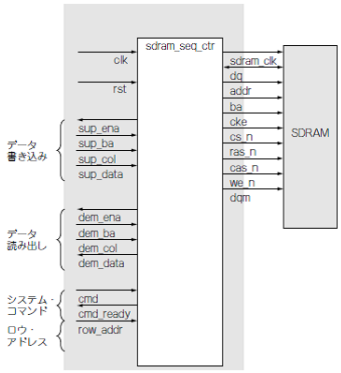


図 4.21: SDRAM コントローラ

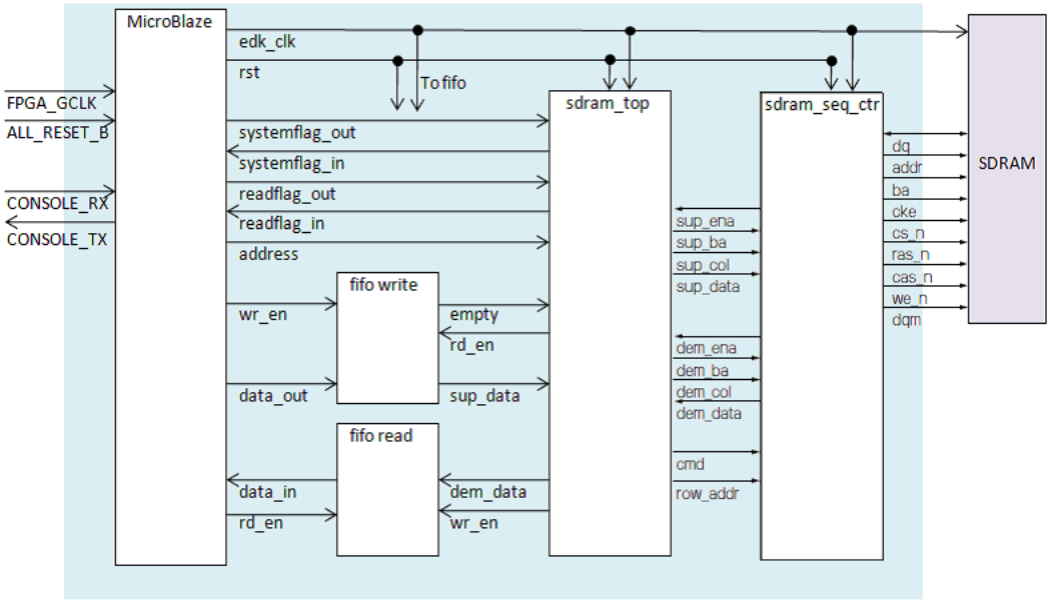


図 4.22: テストデザイン

第5章　まとめ

LHC のアップグレードに向けて、TGC のデータ読み出しシステムの研究を行った。本研究は読み出しシステムの FPGA にダウンロードするプログラムを現状の HDL だけを用いたデザインではなく、ソフトウェアを用いた処理をリアルタイムで行うことを目指し、そのために MicroBlaze という Xilinx 社製 FPGA に搭載可能なソフトプロセッサコアを用いて新 ROD に組み込むことに意義があるかどうかを検証した。以下にそれぞれのまとめとこれからの展望を述べる。

5.1　Read Out Driver Logic の検証

新 ROD に搭載させる HDL デザインで、データを整形する Read Out Driver Logic を作成し、シミュレートした。SSW からのデータ、ROS への送信は 40MHz で行い、前後の SSW と ROS にタイミングを合わせる。データ整形を待ち行列理論で算出した 3.37μs 以内で行うために Clock Wizard という IP コアを用いて、120MHz で動かすことでアップグレード後のデータ量増加にも耐えられることを確認した。データ整形だけでなく、MicroBlaze と組み合わせたエラー処理やシステム診断に関して組み込んでいくことが今後の課題である。

5.2　MicroBlaze の検証

MicroBlaze を用いた検証では、HDL による処理と組み合わせて上手く対応できるか、また作成した汎用モジュールが MicroBlaze を用いて操作できるかをテストした。その結果、HDL デザインからのデータ取得では 5.3Mbyte/s 以上の転送速度で通信できることがわかった。また HDL デザインからの信号を感知してプログラムを起動させたり、メモリへのアクセスなどのテストから

- データ収集のエラーにより起動
- データフローの停止
- エラー発生時のデータの保管
- エラー内容の解析
- データフローシーケンスの初期化
- データフローの再起動

などのエラー処理に MicroBlaze を用いることが有用であるということがわかった。さらに IP コアを使用し、メモリへのアクセスやシリアル転送が行うことができた。これはモジュール化した HDL デザインを MicroBlaze で動的制御したことに他ならない。今後は IP コアだけでなく、自作の HDL デザインをモジュール化し、制御できるようにすることが求められる。

5.3 これからの展望

これから新 ROD 開発へ向け、具体的にどのようなエラー処理、システム診断を行うかを検討し、HDL デザイン、ソフトウェアプログラムを開発し、CERN にて実際にテストを行いたいと考えている。また今回は触れることができなかったが、ソフトウェアプログラムを OS 上で管理することもこれから行っていきたいと考えている。

Appendix

MicroBlazeの検証で用いたソースコード

テスト 1、テスト 2 の FIFO からのデータ取得で用いた HDL コード。

```
‘timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: The University of Tokyo
// Engineer: Yoichi Ninomiya
//
// Create Date: 21:11:24 02/05/2011
// Design Name:
// Module Name: FIFOtest_top
// Project Name:
// Target Devices: XC6SLX45T-3FGG2484
// Tool versions: ISE12.3
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module FIFOtest_top(
  input CLK_N,
  input CLK_P,
  input U_rx,
  output U_tx,
  input RST
  );

reg [2:0] r_timer = 3'b0;
reg [6:0] counter;
reg wr_en;
reg rf_reg;

reg ready;
```

```
always@(posedge clk_edk or posedge RST)begin
  if(RST) begin
    r_timer <= 3'b0;
  end
  else if (wr_en) begin
    r_timer <= 3'b0;
  end
  else begin
    r_timer <= r_timer +1'b1;
  end
end

always@(posedge clk_edk or posedge RST)begin
  if(RST) begin
    counter <= 7'b0;
  end
  else if (wr_en) begin
    counter <= counter + 1'b1;
  end
  else begin
    counter <= 7'b0;
  end
end

always@(posedge clk_edk or posedge RST)begin
  if(RST) begin
    wr_en <= 1'b0;
  end
  else if(r_timer == 3'b111) begin
  wr_en <= 1'b1;
  end
  else if(counter == 7'b1100100) begin
    wr_en <= 1'b0;
  end
end

always@(posedge clk_edk or posedge RST)begin
  if(RST) begin
    rf_reg <= 1'b0;
  end
  else if(counter == 7'b1100101) begin
    rf_reg <= 1'b1;
  end
end
```

```
always@(posedge clk_edk or posedge RST)begin
   if(RST) begin
      ready <= 1'b0;
   end
   else if(rd_en) begin
      ready <= 1'b1;
   end
   else begin
      ready <= 1'b0;
   end
end

wire [31:0] dout;

fifo_generator_v7_2 fifo_i(
   .rst(RST),
   .wr_clk(clk_edk),
   .rd_clk(ready),
   .din(counter),
   .wr_en(wr_en),
   .rd_en(rd_en),
   .dout(dout),
   .full(full),
   .empty()
);

// XST black box declaration
// box_type "black_box"
// synthesis attribute box_type of system_i is "black_box"

system system_i(
   .fpga_0_RS232_Uart_1_RX_pin(U_rx),
   .fpga_0_RS232_Uart_1_TX_pin(U_tx),
   .fpga_0_clk_1_sys_clk_p_pin(CLK_P),
   .fpga_0_clk_1_sys_clk_n_pin(CLK_N),
   .fpga_0_rst_1_sys_rst_pin(RST),
   .dataIO_GPIO_IO_I_pin(dout),
   .flagIO_GPIO_IO_I_pin(rf_reg),
   .flagIO_GPIO_IO_O_pin(rd_en),
   .clock_generator_0_CLKOUT1_pin(clk_edk)
);

endmodule
```

データをプルする場合のソフトウェアコード

```
#include ”stdio.h”
#include ”xparameters.h”
#include ”xgpio.h”

int main(void){

   unsigned int i;
   unsigned int data;
   unsigned int flag;

   XGpio dataIO;
   XGpio flagIO;

   XGpio_Initialize(&dataIO, XPAR_DATAIO_DEVICE_ID);
   XGpio_Initialize(&flagIO, XPAR_FLAGIO_DEVICE_ID);

   data = 0;

   XGpio_DiscreteWrite(&flagIO,1,0);

   while(1){
      flag = XGpio_DiscreteRead(&flagIO,1);
      if(flag == 1){
         break;
      }
   }

   for(i=0;i<100;i++){
      XGpio_DiscreteWrite(&flagIO,1,1);
      data = XGpio_DiscreteRead(&dataIO,1);
      XGpio_DiscreteWrite(&flagIO,1,0);
      xil_printf(”data =%d\n\r”,data);
   }

   return 0;
}
```

データをプッシュする場合のソースコード

```
#include ”stdio.h”
#include ”xparameters.h”
#include ”xgpio.h”

int main(void){

   unsigned int i;
   unsigned int data[100];
```

```
  unsigned int flag;

  XGpio dataIO;
  XGpio flagIO;

  XGpio_Initialize(&dataIO, XPAR_DATAIO_DEVICE_ID);
  XGpio_Initialize(&flagIO, XPAR_FLAGIO_DEVICE_ID);

  for(i=0;i<100;i++){
    XGpio_DiscreteWrite(&flagIO,1,0);
    while(1){
      flag = XGpio_DiscreteRead(&flagIO,1);
      if(flag == 1){
        data[i] = XGpio_DiscreteRead(&dataIO,1);
        XGpio_DiscreteWrite(&flagIO,1,1);
        break;
      }
    }
  }

  for(i=0;i<100;i++){
    xil_printf("data = %d\n\r",data[i]);
  }

  return 0;
}
```

関数のオーバーヘッドを計測 (xps_timer という IP コアが必要)

```
#include "stdio.h"
#include "xtmrctr.h"

#define TMRCTR_DEVICE_ID XPAR_TMRCTR_0_DEVICE_ID
#define TIMER_COUNTER_0 0

XTmrCtr TimerCounter;

int main(void){

  int cycles;

  XStatus Status;
  XTmrCtr *TmrCtrInstancePtr = &TimerCounter;
  Xuint8 TmrCtrNumber = TIMER_COUNTER_0;

  Status = XTmrCtr_Initialize(TmrCtrInstancePtr, TMRCTR_DEVICE_ID);
  if(Status != XST_SUCCESS){
```

```
    return XST_FAILURE;
  }

  XTmrCtr_SetResetValue(TmrCtrInstancePtr, TmrCtrNumber, 0x00000000);
  XTmrCtr_Reset(TmrCtrInstancePtr, TmrCtrNumber);

  XTmrCtr_Start(TmrCtrInstancePtr, TmrCtrNumber);
  (調べたい関数をここに挿入)
  XTmrCtr_Stop(TmrCtrInstancePtr, TmrCtrNumber);
  cycles = XTmrCtr_GetValue(TmrCtrInstancePtr, TmrCtrNumber);
  xil_printf("cycles =%d\r\n",cycles);

  return 0;
}
```

共有メモリへのデータ書き込み (Multi-Port Memory Controller という IP コアが必要)

```
#include "stdio.h"

int main(void){

  volatile int *flag;
  flag = 0x88000012;
  int *datain;
  datain = 0x88000004;
  int data;
  *flag = 0;

  if(*flag == 0){
    *datain = 100;
    *flags = 1;
  }
  return 0;
}
```

共有メモリへのデータ読み込み (Multi-Port Memory Controller という IP コアが必要)

```
#include "stdio.h"

int main(void){

  volatile int *flag;
  flag = 0x88000012;
  int *dataout;
  dataout = 0x88000004;
  int data;

  while(1){
    if(*flag == 1){
```

```
        data = *dataout;
        xil_printf("data= %x\r\n",data);
        *flags = 0;
        break;
      }
    }
    return 0;
}
```

謝辞

本研究を進めるにあたり、多くの方々から御指導と御支援を頂きました。ここに深く感謝の意を表します。指導教員の坂本宏教授には、2 年間に渡り格別なる御指導と御高配を賜わりました。数多くの研究者と交流する機会を提供していただき、自分の知見を広げ、また充実した研究生活を送れたことに心より感謝致します。

佐々木修氏、蔵重久弥氏には研究面において的確な助言を頂きました。深く感謝致します。内田智久氏、池野正弘氏、宮沢正和氏にはハードウェアに関する多くの指導を頂きました。同じ研究室の織田勧氏、結束晃平氏、神谷隆之氏にも様々なアドバイスを頂きました。深謝致します。

また, ATLAS 日本グループの徳宿克夫氏、小林富雄氏、川本辰男氏、福永力氏、戸本誠氏、岩崎博行氏、石野雅也氏、菅谷頼仁氏、越智敦彦氏、松下崇氏、杉本拓也氏、他 ATLAS 日本グループの皆様には研究, 生活両面で大変お世話になりました。心よりお礼申し上げます。

研究生活の多くの場面で手助けをしていただいた、 久保田隆至氏、東裕也氏、山口博史氏、道前武氏、石田明氏、奥山豊信氏、越前谷陽佑氏、早川俊氏、西山知徳氏、鈴木友氏、奥村恭幸氏、高橋悠太氏、長谷川慧氏、伊藤悠貴氏、岸木俊一氏、廣瀬穣氏、管野貴之氏、先輩方にも大変感謝しています。

また, 同期である佐々木雄一氏、宮崎彬氏、風間慎吾氏、Khaw Kim Siang 氏、井上竜一氏、吉原圭亮氏、藤井祐樹氏、山口洋平氏、飯山悠太郎氏、小森雄斗氏、橋本直氏、Katarina Bendtz 氏、谷和俊氏、徳永香氏、吹田航一氏、宮崎一樹氏、志知秀治氏、前島亮平氏、若林潤氏、高橋将太氏、中野浩至氏、齋藤智之氏、岡村航氏 らの存在は研究を行う上で大きな励みとなりました。

事務員として研究生活をサポートしてくださったの安蒜律子氏、塩田雅子氏、森田智恵子氏、鈴木恵美氏、片岡直子氏にもとてもお世話になり、感謝しています。

最後に、様々な面で支えてくださった友人、そしてこの 2 年間で出会った全ての方々に感謝します。特に、あらゆる面で支え続けてくれた両親に、心から感謝致します。

参考文献

[1] ATLAS Experiment, "ATLAS Photos", [http://www.atlas.ch/photos/index.html]

[2] CERN Document Server [http://cdsweb.cern.ch/]

[3] ATLAS Muon Spectrometer Technical Design Report, CERN/LHCC/97-22, 1999

[4] 桑原隆志. 東京大学修士学位論文「ALTAS 前後方ミューオントリガーシステムの構築」2007 年 1 月

[5] M.Spira, Higgs Production and Decay at Future Machines, CERN -TH/97-323, hep-ex/9711394 1997

[6] 浅井祥仁 講義ノート [http://www.icepp.s.u-tokyo.ac.jp/ asai/Lecture/4nensei3.pdf]

[7] 浅井祥仁 LHC で期待されている物理 2006, 3, 1

[8] 徳宿克夫 2010 年日本物理学会秋季大会「LHC upgrade 計画概要」

[9] ATLAS High-Level Trigger, Data Acquisition and Controls, ATLAS Technical Design Report-016, 2003.

[10] ATLAS Level1 Trigger Technical Report, ATLAS TDR 12, 1999.

[11] 野本裕史 Star Switch Specification, 2004, Jun. 8

[12] Lorne Levinson ATLAS Endcap Muon Trigger Read Out Driver [https://atlas-proj-tgc.web.cern.ch/atlas-proj-tgc/ROD/TGCROD.html]

[13] CERN S-LINK homepage [http://hsi.web.cern.ch/HSI/s-link/]

[14] Xilinx homepage [http://japan.xilinx.com/]

[15] 渡部謹二 Design Wave Magazine 2009 January P122-P127

[16] Xilinx ISim ユーザガイド [http://japan.xilinx.com/support/documentation/sw_manuals/xilinx12_1/plugin_ism.pdf]

[17] Xilinx LogiCORE^{TM}IP FIFO Generator v7.2 User Guide

[18] 結束晃平　東京大学修士学位論文「ATLAS ミューオン検出器における読み出しシステムのコミッショニング及び SuperLHC に向けたアップグレードの研究」

[19] 越前谷陽佑 東京大学修士学位論文「STLAS 前後方ミューオントリガーシステムのコミッショニングとアップグレードに向けた研究開発」

[20] 川本辰男 2010 年日本物理学会秋季大会 「LHC 測定器のアップグレード計画」

[21] spartan6 ファミリ概要 v.1.5 [http://japan.xilinx.com/support/documentation/data_sheets/j_ds160.pdf]