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

東京大学大学院理学系研究科物理学専攻
坂本研究室

結束 晃平
Kohei Kessoku

平成 21 年 2 月 9 日

概 要

LHC 加速器を用いた ALTAS 実験が本格的に始まろうとしている。日本グループは ATLAS 検出器の前後方ミューオントリガー検出器:TGC に携わってきた。ATLAS 実験の本格稼動へ向けて TGC はこれまで数々の検査を行い、品質を高めてきた。私は地下インストール前の最終検査である地上動作試験において、TGC の FI と呼ぶ部分に対して検査を行った。この検査によって FI の物理的な欠陥を 0.2%に抑えることができた。

一方で SuperLHC 計画に向けたアップグレード開発も次第に本格化してきた。ルミノシティ増強によって TGC のヒットレートも現行 LHC より上昇することが予想される。イスラエルグループによって開発された TGC 読み出しエレクトロニクスの一つである ROD モジュールは、現在最も動作懸念がされているモジュールであり TGC 読み出しシステムの不安要素となっている。そこで我々日本グループはその解決策の一つとして、SuperLHC を視野に入れた新たな ROD モジュールの開発に着手した。今回初期開発として、HDL を用いて論理回路の基本骨格を設計し ROD に要求される動作速度である 125MHz を十分上回る速度で動作することをシミュレーションによって確認した。

以上の研究成果は、これから本格稼動を迎える ATLAS 実験の成果に対し確実に貢献していくだろう。

目 次

序論

エネルギーフロンティア実験の完全稼動実現と更なる挑戦

1970 代に確立された標準理論はその後、LEP や SLC,Tevatron などを代表とする各種実験成果によって極めて高い精度で検証された。しかしながら標準理論において鍵となる Higgs 粒子は、存在が予言されてから 40 年以上を経た今も未発見のままである。またこれらの実験は標準理論の正しさを示すとともに、標準理論のその先にある大統一理論の可能性を示唆した。現代素粒子物理学は大きな成功を収めつつも深淵でより本質的な問題に直面している。

ATLAS 実験はこれら現代素粒子物理学が抱える問題に答えるために構想された大型汎用検出器である。私の所属する日本グループは、ATLAS の中のミュー粒子検出器—TGC 検出器を担当している。現在において ATLAS 実験は史上最高エネルギーのビームを用いた史上最大の検出器であり、それゆえの様々な量的、質的な挑戦が存在する。信用できる物理解析を行うにはこれら難関を乗り越え、高い水準で検出器を稼動させることが必須である。以上の背景から TGC グループが次の点を把握することは重要な意味を持つ。

- TGC 検出器はどの程度検査保証され、正常に稼動するか

また近い将来、さらにルミノシティを上げて物理探索を行おうとする Super LHC 計画が予定されている。当然 ATLAS 検出器もそれに応じるべくアップグレードさせる必要がある。TGC 検出器においてはデータ読み出しシステムがその対象部分であり、私達は本年度よりその研究開発に着手した。そこで本論では以下に対する報告を行う。

- 現状における新しい読み出しシステムの開発研究成果はどうか

以上の問いに答えるためにまず本実験とはどのようなものかを把握するべく、1 章で LHC 加速器、2 章で ATLAS 検出器について概要を述べる。ATLAS 検出器の中でも本論の中心となる TGC 検出器については 3 章で詳しく述べることにする。その上で 4 章において上記の一つ目の問いに対して述べ、5 章で二つ目の問いに対して述べる。そして最後に結論と総括を行うことにする。

第1章　LHC加速器

まずは本実験の心臓部と言えるLHC加速器について述べる。検出器の立場から考えたとき加速器の特徴として把握しておきたいことは、衝突させる粒子の種類、衝突エネルギー、衝突頻度などである。この章ではそれらに主眼をおいて述べることにする。またアップグレード計画であるSuperLHC計画についても、本論に特に関係しそうな点に絞って述べる。

1.1　現行のLHC

LHC(Large Hadron Collider)加速器はスイスとフランスの国境付近にある、CERN研究所の地下100mに建設された衝突型陽子陽子加速器である。周長は約27km、粒子衝突におけるその重心系エネルギーは14Tev、ルミノシティは最高で$10^{34}[cm^{-2}s^{-1}]$に達する。LHCには4箇所の衝突点が存在し、そのうちのPoint1と呼ぶ位置にATLAS検出器は設置されている。ATLAS検出器の詳細は次章で述べる。

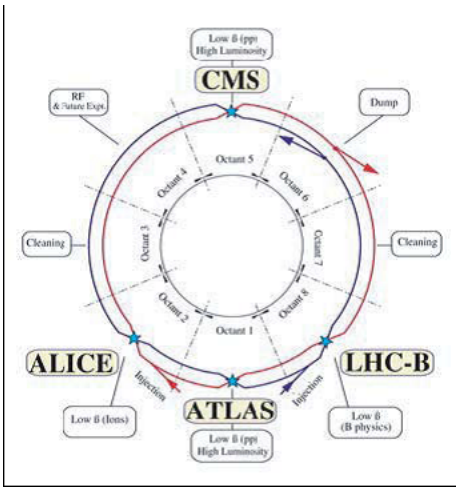


図 1.1: LHC加速器と4つの検出器
LHC加速器にはATLAS,CMS,ALICE,LHC-Bの4つの検出器が設置されている。

主リング周長	26.66[km]
衝突頻度	40.08[MHz]
重心系エネルギー	14[TeV]
(高) ルミノシティ	$10^{34}[cm^{-2}sec^{-1}]$

表 1.1: LHC加速器の主要なパラメータ

図 1.2: LHC 加速器

LHC は上記の他に次の 2 点の特徴を持つ。

- 陽子陽子衝突であること。
- 統計量を効率良くためるために 40MHz(25ns 間隔) で粒子衝突を繰り返すこと。

以上を考慮すると、LHC で起こる物理現象に関した情報量は莫大なものになることが容易に予想される。そのため LHC で有用な物理現象を探索するためには、これら膨大なデータから効率よく有用なデータを取得することが必要になる。そのような理由から ATLAS 検出器はデータ取捨選別システムを備えている。また、詳細は 3 章で述べるが ATLAS のミューオン検出器である TGC 検出器はこのデータ取捨選別システムを担う検出器の一つである。

1.2 Super LHC 計画

物理探索領域をさらに拡大するために、LHC 加速器はさらなる性能強化を目指した SuperLHC(SLHC) 計画を予定している。現時点でのスケジュールを図 1.3 に載せる。

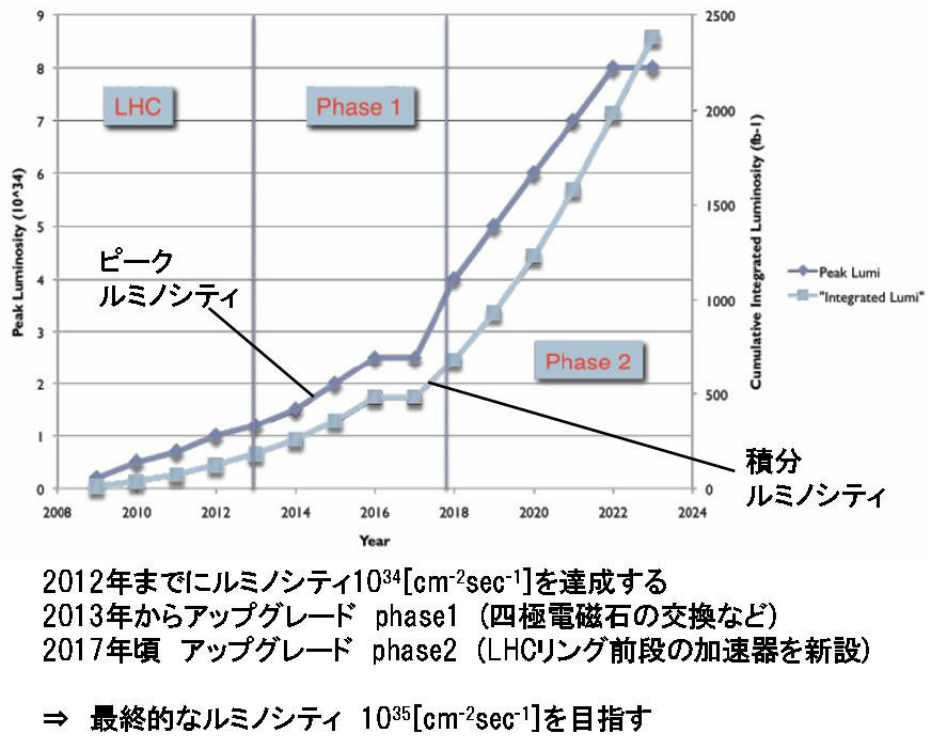


図 1.3: SuperLHC 計画
現時点でのスケジュール

現在までにエネルギーの増強とルミノシティの増強についての研究が進められているが、ここではより現実的なルミノシティ増強について、考えられているアップグレードプランを述べる。

ルミノシティ増強プランとしては、実験の観点から重要なバンチ間隔、つまり衝突頻度について分類すると以下の 3 つのシナリオが考えらる。

1: 現行と同じ間隔 (25[ns])

現行と同じ 25[ns] 間隔、つまり 40MHz である。現行の 2〜7 倍のルミノシティが達成可能である。具体的にはさらに次の 2 つのプランが考えられている。

1. バンチ内の陽子数を増やすのみ。このプランはハードウェアを変えることでほぼ達成できる。そのためコストは他のプランに比べかからない。ルミノシティは現行の 2 倍程度が達成可能である。

2. 上のプランに加え、新しい収束磁石を導入することで衝突点での横ビームサイズを小さくし、RF システムの修正によってバンチ長を半分程度にする。以上のアップグレードによってル

ミノシティを現行の 5 倍程度にすることができる。さらに Piwinski パラメターを大きくした場合ルミノシティは 7 倍程度まで達成可能である。

2: より短い間隔 (10[ns] or 12.5[ns] or 15[ns])

現行より短い間隔で 67MHz〜100MHz が考えられている。現行の 7〜11 倍のルミノシティが達成可能である。より強い収束を達成させるために、各種装置をアップグレードさせ現在より衝突点に近づける必要がある。短いバンチ間隔により電子雲効果の影響などが強くなるため、ビーム不安定性が大きくなってしまう。なお 12.5[ns] 間隔の場合は加速器の RF システムを新しくする必要があり、10[ns] 及び 15[ns] 間隔よりコストがかかる。

3: より長い間隔 (75[ns])

現行より長い 75[ns] 間隔、13.3MHz である。現行の約 9 倍のルミノシティが達成可能である。

上の 2 者に比べ新しいシナリオである。バンチ長を倍にすることで各バンチの利得を約 $\sqrt{2}$ 倍にする。そのためリング内のバンチ数は減るものの、ルミノシティとしては現行の 9 倍程度になる。

以上をまとめたものを表 1.2 に載せる

パラメータ	現行	シナリオ 1			シナリオ 2		シナリオ 3
		Ultimate	IR-upgrade	IR-upgrade-Piwinski			
バンチ間隔 [ns]	25	25			15	10	75
バンチあたりの陽子数 [$\times 10^{11}$]	1.15	1.7		2.6	1.7		6.0
リング内のバンチ数	2808	2808			4680	7020	936
バンチ長 [cm]	7.55	7.55	3.78	7.55	3.78		14.4
衝突角度 [μrad]	285	315	445	485	445		430
ルミノシティ [$\times 10^{34} cm^{-2} s^{-1}$]	1.0	2.3	4.6	7.2	7.7	11.5	8.9
衝突あたりの陽子反応数	19	44	88	132	88		510

表 1.2: SLHC 計画に関するパラメータ

このように現段階ではアップグレードプランがどのようなシナリオになるか確定していないが、加速器技術の面、コスト面、検出器側からの要求などあらゆる側面から検討した上で最適なものが決定される。

本論においてこのアップグレードシナリオに対する重要な点は次のことである。

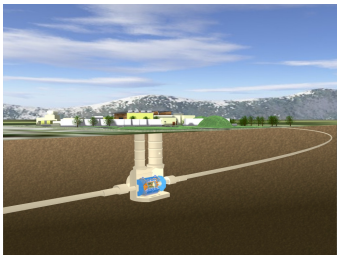
- いずれのシナリオにおいても情報量が現行の LHC よりも増大すること

衝突によって起こった事象は複雑である。それを意味のある現象として捉えるには高性能の検出器システムが必要である。次の章では LHC 加速器のビーム衝突による物理現象を広範に渡って測定する ATLAS 検出器について述べる。

第2章　ATLAS検出器

ここではATLAS検出器の概要について述べ、本論の主題に関わるTGC検出器の構造的、機能的位置付けを示す。また主題である読み出しシステムについても述べる。

前章で述べたLHCは史上初めてTevスケールでの直接物理探索を可能にする、エネルギーフロンティア加速器である。そこで、未踏の領域に潜んでいるであろう新物理現象を可能な限り網羅する検出器が望まれる。ATLAS(A Toroidal Lhc ApparatuS)検出器はその思想にたって設計された直径25m、長さ44mの円筒形大型汎用検出器である。ATLAS検出器が設置されているPoint1の構造を示す。



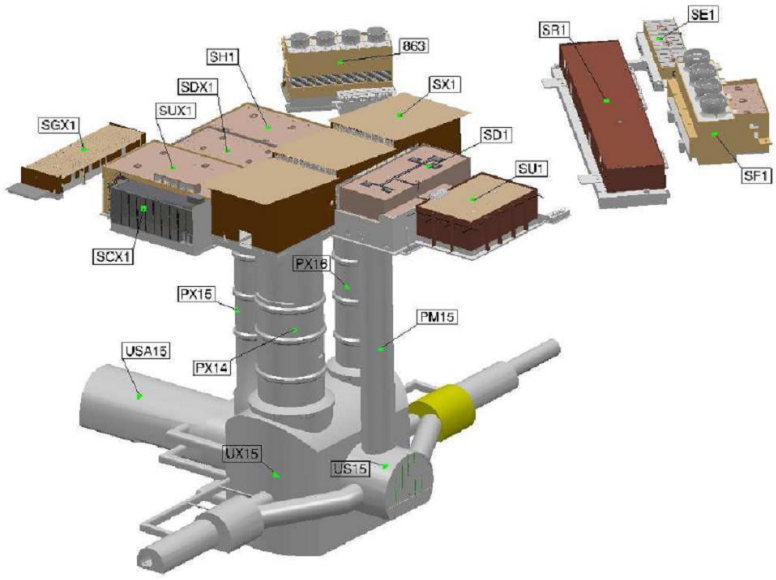


図 2.1: Pint1の構造

ATLAS検出器が設置されているのがUX15という空間。またUSA15は放射線から遮断されたエリアでエレクトロニクスが置かれる。

検出器自体は広大な空洞(cavern)内に設置される(図中ではUX15)。また空洞とは隔てられたUSA15と呼ばれるカウンティング・ルームにはデータの流れで言えば下流のエレクトロニクスが置かれ、上流エレクトロニクス(フロントエンド)、つまり検出器側(空洞内)のエレクトロニクスから来るデータを引き継いで処理する。USA15は加速器からの放射線が良く遮断されているエリアである。そのため加速器が稼働中でも人のアクセスが可能であり、また設置されるエレクトロニクスに対しても放射線耐性を考慮する必要がほとんどない。

ATLAS 検出器の全体像を載せる。

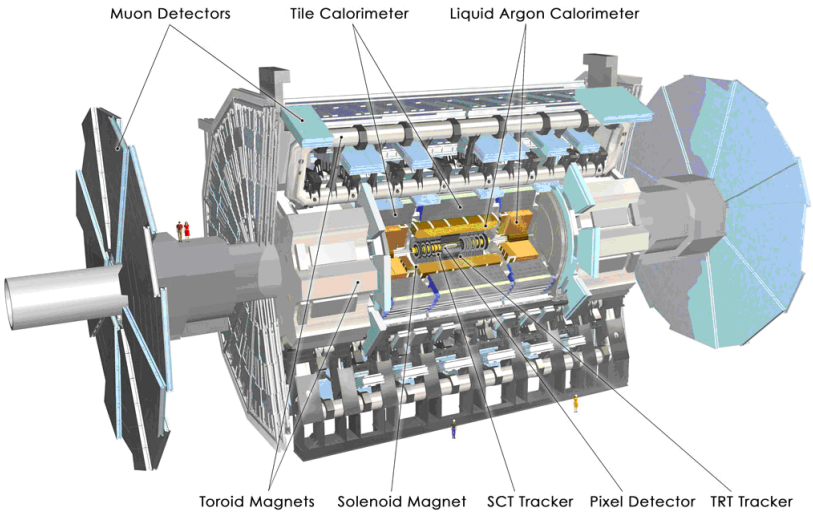


図 2.2: ATLAS 検出器の鳥瞰図
直径は 25m、長さは 44m ある。

本論の中心になる前後方ミューオン検出器は両端 2 箇所に分かれるので、それを識別するために A サイド、C サイドと ATLAS では区別している。

ATLAS 検出器は他の汎用検出器と同じく目的や検出領域の異なる複数の検出器から構成されている。以下ではまず各検出器の概略を述べ、その後に ATLAS でのデータ収集方法及び期待される物理について述べる。なおここで、これ以降頻出する ATLAS での座標系定義を記しておく。

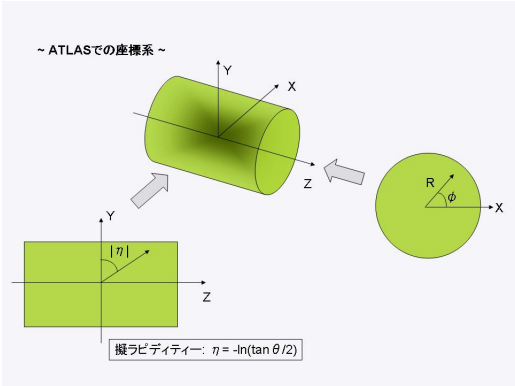


図 2.3: ATLAS での座標系

2.1 ATLAS を構成する検出器と磁石

ATLAS 検出器の構成を図 2.4 に載せる。

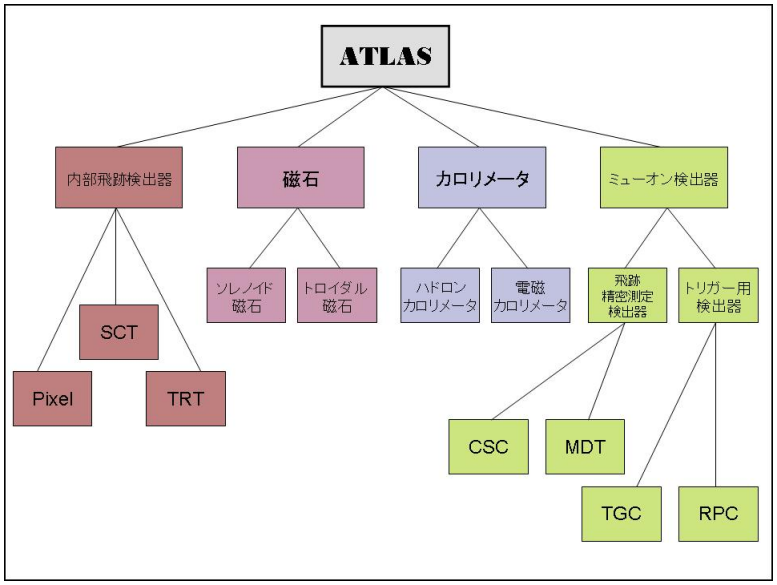


図 2.4: ATLAS 検出器の構成

ATLAS は内部飛跡検出器、カロリメータ、ミューオン検出器の 3 グループと磁石から構成される。

ATLAS では以下の、内側から大きく分けて次の 3 つの検出器群に分けられる。

- 内部飛跡検出器
- カロリメータ
- ミューオン検出器

磁石は検出器間に設置されている。各検出器及び磁石についてみていく。

2.1.1 内部飛跡検出器

ビームの衝突点に最も近い場所に設置され、ソレノイド磁場中におかれる。以下の 3 つの検出器から構成されている。

- ピクセル検出器 (Pixel)
- 半導体トラッカー (SCT)
- 遷移輻射トラッカー (TRT)

概要を述べる。

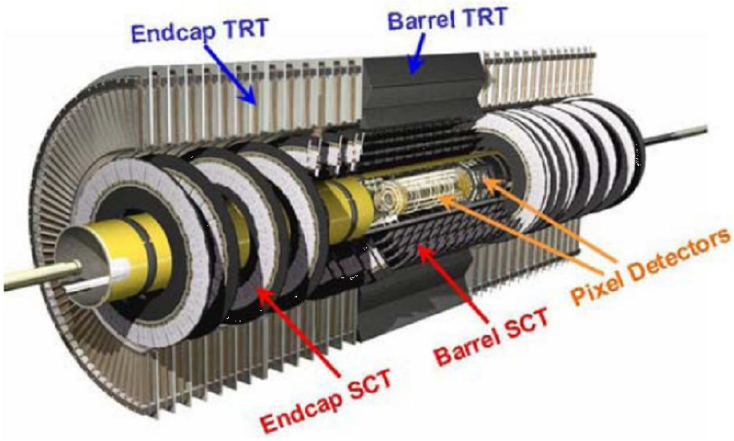


図 2.5: 内部飛跡検出器
内側から Pixel,SCT,TRT となっている。

1. Pixel
   内部飛跡検出器の中でも最内層にある高分解能半導体検出器である。衝突点と 2 次崩壊点の再構成に用いられる。

2. SCT
   細長い有感領域をシリコン上に施した半導体検出器である。Pixel とともにインパクトパラメータの再構成に用いられる。加えて r-φ方向に優れた位置分解能を持ち、それを利用して荷電粒子の横運動量を高い精度で測定する。

3. TRT
   ストロー型のドリフトチューブセンサーである。飛跡測定の他に遷移輻射を利用した電子の識別を行う。

2.1.2 カロリメータ

2 種類からなる。なおカロリメータはデータ取捨選別、すなわちトリガー判定を行う。

- 電磁カロリメータ
- ハドロンカロリメータ

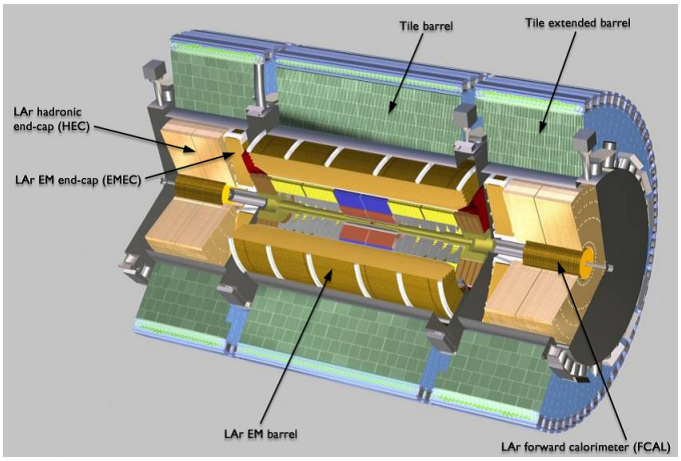


図 2.6: カロリメータ
内側に電磁カロリメータ、外側にハドロンカロリメータが存在する。

概要を述べる。

1. 電磁カロリメータ
   アコーディオン構造の鉛の吸収体と液体アルゴンからなり、電子と光子の同定に用いられる。放射線耐性に優れた検出器である。
2. ハドロンカロリメータ
   電磁カロリメータの外側に位置する。ハドロンの同定、エネルギー測定、ジェットの再構成に用いられる。

2.1.3 ミューオン検出器

ミューオン検出器群はトロイド磁場内を通過するミュー粒子の軌跡情報を基にした検出器システムである。以下の 2 グループ 4 種類の検出器から構成されている。

- 軌跡精密測定用検出器
  - モニタード・ドリフト・チューブ (MDT)
  - カソード・ストリップ・チェンバー (CSC)
- トリガー用検出器
  - レジスティブ・プレート・チェンバー (RPC)
  - 薄隙型多線式比例計数管 (TGC)

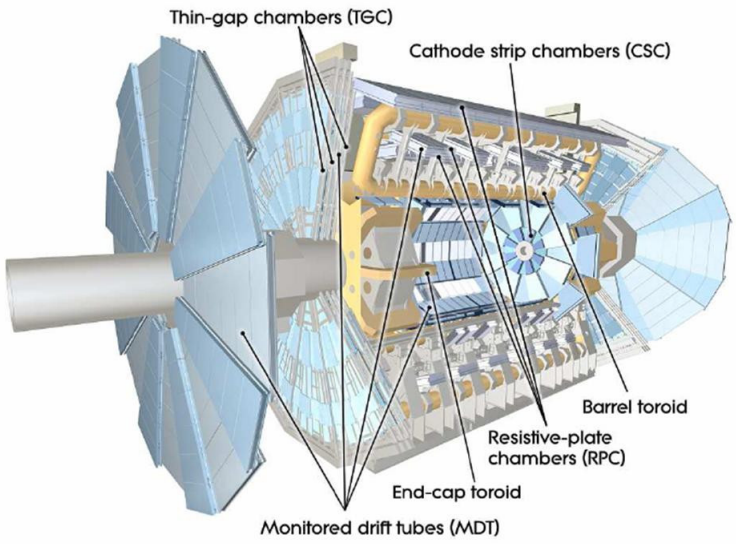


図 2.7: ミューオン検出器
バレル部に RPC、エンドキャップ部に TGC と CSC が存在し、
MDT はバレル部及びエンドキャップ部にある。

Monitored drift tubes	MDT
- Coverage	$\mid \eta \mid < 2.7$ (innermost layer: $\mid \eta \mid < 2.0$)
- Number of channels	339000 (354000)
- Function	Precision tracking
Cathode strip chambers	CSC
- Coverage	$2.0 < \mid \eta \mid < 2.7$
- Number of channels	31000
- Function	Precision tracking
Resistive plate chambers	RPC
- Coverage	$\mid \eta \mid < 1.05$
- Number of channels	359000 (373000)
- Function	Triggering, second coordinate
Thin Gap chambers	TGC
- Coverage	$1.05 < \mid \eta \mid < 2.7$ (2.4 for triggering)
- Number of channels	318000
- Function	Triggering, second coordinate

表 2.1: ミューオン検出器の主要なパラメータ

軌跡精密測定用が MDT と CSC。MDT が $\mid \eta \mid < 2.7$,CSC が $2.0 < \mid \eta \mid < 2.7$ の領域をカバーする。トリガー用が RPC と TGC。RPC が $\mid \eta \mid < 1.05$,TGC が $1.05 < \mid \eta \mid < 2.7$ の領域をカバーする。MDT は R 方向の測定のみであるため、RPC 及び TGC は ϕ 方向の測定も行う。

主要なパラメータに関する表を表 2.1 に載せる。
各検出器の概要を述べる。

1. ミューオン軌跡精密測定
   バレル部及びエンドキャップ部に MDT が用いられ、フォワード部に CSC が用いられる。MDT は幅広い $\mid \eta \mid$ 領域 ($\mid \eta \mid < 2.7$) に渡り、磁場によって曲がったミューオンの軌跡を精密測定する。CSC は $\mid \eta \mid$ の大きな領域 ($2.0 < \mid \eta \mid < 2.7$) を覆い、多量のバックグラウンド環境下でミューオンを精度良く測定する。

2. トリガー用検出器

   データ取捨選別、すなわちトリガー判定を担う検出器である。バレル部に RPC が用いられ、エンドキャップ部及びフォワード部に TGC が用いられる。TGC は $\mid \eta \mid < 2.4$ の領域を覆う。ミューオントリガー検出器は以下の役割を持つ。

   (a) 第一段データ取捨選別 (レベル 1 トリガー判定) を行う。

   (b) バンチの識別を行う。

   (c) 検出したミューオンを横運動量によって区別する。

   (d) 高速データ処理により大まかなミューオンの軌跡情報を測定し、第二段データ取捨選別システム (レベル 2 トリガー判定) に提供する。

(e) 検出したミューオンの r 方向、φ 方向の測定情報を測定する。MDT は r 方向のみの測定であり、トリガー検出器による φ 方向の情報が加わることでミューオンの 2 次元としての測定位置が得られる。

データ取捨選別システムについては次節、TGC は次章で改めて取り上げることにする。

2.1.4 磁石

ATLAS には次の 2 種類がある。

- ソレノイド磁石
- トロイド磁石

構造を載せる。

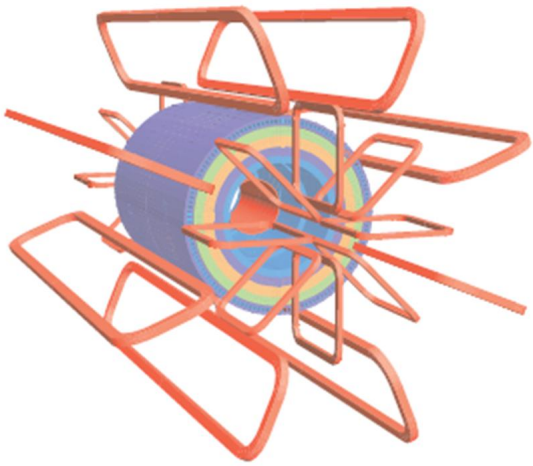


図 2.8: 磁石の構造
中央にソレノイド磁石があり、その外側にバレル部トロイド磁石とエンドキャップ部トロイド磁石が置かれている。

概要を述べる。

1. ソレノイド
   内部飛跡検出器での荷電粒子の運動量測定のために用いられる。磁場強度は 2T である。

2. トロイド
   ミューオン検出器でのミューオンの運動量測定のために用いられる。バレル部 ($0 < |\eta| < 1.4$) とエンドキャップ部 ($1.6 < |\eta| < 2.7$) からなる。磁場強度はバレル部で 1.5〜5.5Tm、エンドキャップ部で 1〜7.5Tm である。ただしバレル部からエンドキャップ部への遷移領域 ($1.4 < |\eta| < 1.6$) における磁場強度は他に比べ低くなる。

高品質の粒子衝突情報を得るには、検出器自体の性能に加え緻密に設計されたエレクトロニクス及びソフトウェアが伴うことによって達成される。次では ATLAS におけるデータ収集システムについて述べることにする。

2.2 ATLAS 実験におけるトリガー・データ収集システム (TDAQ : Trigger and Data Acquisition)

現行 LHC の最高ルミノシティ時の陽子－陽子反応レートは 1GHz に達する。しかし記憶装置、計算機資源の制限からこれら全イベント全情報を記録することはできず、最終的なデータ収集レートを 200Hz まで落とす必要がある。そのため ATLAS 実験では物理現象として興味あるイベントを選別するために、レベル 1 トリガーとハイレベルトリガー (レベル 2 トリガーとイベントフィルターからなる) によって多段的に情報を絞り込むことになっている。TGC はこのレベル 1 トリガーを担う検出器である。

ATLAS の TDAQ システムの概略図を図 2.9 に載せる。

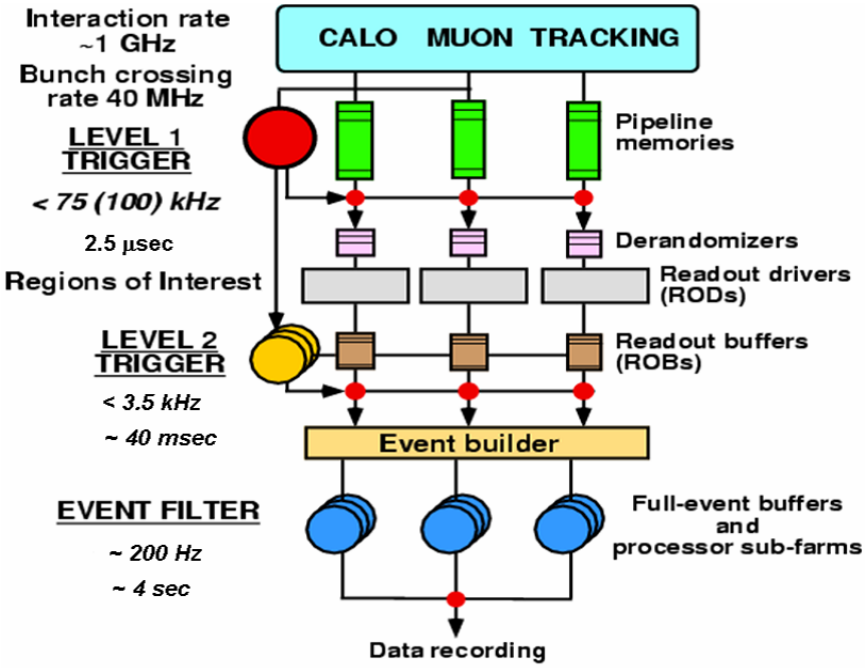


図 2.9: ATLAS の TDAQ システム

バックグラウンドを落とすために、ATLAS は段階的なトリガーによってデータ選別をする。第 1 段階トリガー (レベル 1 トリガー) ではハードウェアによって選別し、それに続くハイレベルトリガー (レベル 2 トリガー及びイベントフィルター) ではソフトウェアによって選別する。

まずトリガーシステムについて述べた後で、そのトリガーを用いたデータ収集アーキテクチャについて述べる。

2.2.1 トリガーシステム

レベル 1 トリガーとハイレベルトリガーに分けて考える。

レベル 1 トリガー

レベル 1 トリガーはカロリメータとミューオントリガー検出器の情報のみを用いた、ハードウェアによる高速トリガーである。このレベル 1 トリガーによってレートは 75kHz 程度にまで落とされる (オプションで 100kHz までアップグレードが可能である)。またレベル 1 トリガーの判定に許されている時間は 2.5 μ秒以内と決められている。

レベル 1 では高い横運動量を持ったミューオン、電子、フォトン、ジェット、ハドロンに崩壊する τ 粒子などに加え、大きな全消失横エネルギーを見つけ出す。そのうち高い横運動量を持つミューオンは RPC と TGC によって選別され、その他はカロリメータの精度を落とした情報によって選別される。

カロリメータと RPC、TGC によってそれぞれ処理されたそれらのトリガー (候補の) 情報は CTP(Central Trigger Processor) に集められる。CTP にはトリガー判定のためのトリガーメニューが実装されている。トリガー情報がメニューの条件を満たし、最終的なレベル 1 トリガー判定を通過すると、それまで各検出器のエレクトロニクス上に保持させていたデータを出力させる信号、L1A(Level 1 Accept) 信号が TTC(Timing,Trigger and Control distribution system) に向けて出力される。TTC はその L1A を各検出器に出力する。

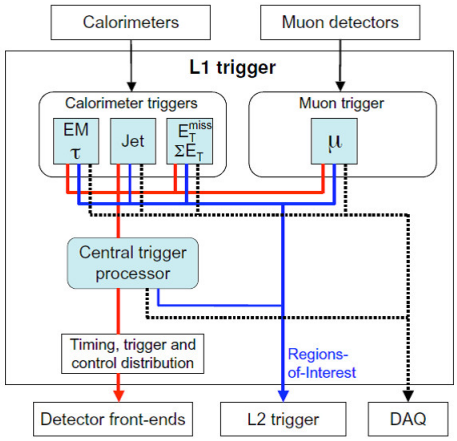


図 2.10: レベル 1 トリガー

レベル 1 トリガーはカロリメータとミューオントリガー検出器の情報を用いて行われる。最終的なトリガー判定は CTP(Central Trigger Processor) によってなされ、判定を通過すると TTC(Timing,Trigger and Control distribution system) から各検出器へとデータ取得信号 (Level1 Accept 信号) が配信される。またレベル 1 トリガーで得られた粒子の存在領域を Regions of Interests(RoI's) と呼び、続くレベル 2 トリガーへと送られレベル 2 トリガー判定に利用される。

その他にレベル 1 トリガーでは、イベントに対して RoI(Regions of Interest) と呼ぶ η とφによる座標領域を定義する。RoI は各トリガー検出器でのレベル 1 トリガー判定において選ばれた領

域を示すものである。RoI 情報は続くハイレベルトリガーで利用されることになる。

ここで簡単に TTC について述べておく。

- TTC システム
  TTC システムは主に次の役割を持つ。
  - 各検出器のエレクトロニクスの同期をとるための信号を分配する。
  - 各検出器独自のテストやキャリブレーション用のコマンドを受信し実行する。

  TTC システムは TTCvi と LTP(Local Trigger Processor),TTCvx,ROD(Read Out of Driver)busy 4 つのモジュールから構成される。また検出器側に TTCrx が設置される。

  TTC システムは TTCvi を中心に構成される。LTP は TTC システム外部から来る信号を受信する。そして L1A 信号は TTCvi へ、クロックは TTCvx へ送信する。TTCvi は LTP から受けた L1A、テスト信号を TTCvx に送信する。TTCvx は各信号を、光ファイバーによって各検出器の上流エレクトロニクス (フロントエンド) に設置される TTCrx に分配する。そしてさらに TTCrx が各エレクトロニクスへその信号を分配する。

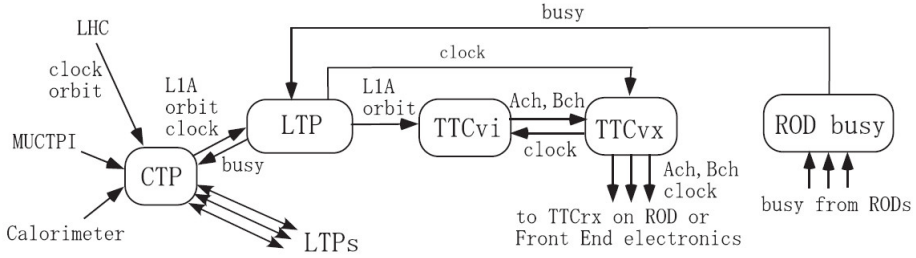


図 2.11: TTC システムの構成
TTCvi と LTP(Local Trigger Processor),TTCvx,ROD(Read Out of Driver)busy の 4 つのモジュールからなっている。

ハイレベルトリガー

ハイレベルトリガーはソフトウェアによってトリガー判定を行い、レベル 2 トリガーとイベントフィルターの 2 段階からなる。レベル 2 トリガーによってレートを約 3.5kHz まで落とし、さらにイベントフィルターで最終的な収集レートの 200Hz にまで落とす。トリガー判定に許されている時間は、レベル 2 トリガーでは約 40ms、イベントフィルターでは 4 秒程度が許されている。

レベル 1 トリガーを通過すると、レベル 2 トリガーに RoI の情報を送る。レベル 2 トリガーでは与えられた RoI に対応する領域内の、利用可能な検出器全ての情報を使ってトリガー判定を行う。

レベル 2 トリガーを通過したイベントはさらにイベントフィルターに送られる。イベントフィルターでは全検出器の全情報を用いてトリガー判定がされる。

2.2.2 データ収集アーキテクチャ

データ収集の流れは以下のようになる。(図 2.10 参照)

1. 検出器から得られた粒子情報はまず、各検出器が持つレベル 1 バッファと呼ばれるパイプラインメモリに保持される。その間にレベル 1 トリガーの判定が行われる。レベル 1 トリガーによって情報取得の判定がなされると L1A(Level 1 Accept) 信号が全検出器に向けて発行され、各検出器はその L1A 信号を受け取る。

2. L1A 信号を受け取ると粒子情報は、その検出器の粒子情報を取りまとめる ROD(Read Out Driver) へと送られる。ただし L1A は不規則に発行されるのでパイプラインメモリから流れてくる粒子情報も不規則である。そのために粒子情報は一度デランダマイザと呼ぶバッファでタイミングを整えられてから順次 ROD(Read Out Driver) に送られる。

3. ROD からの情報は ROB(Read Out Buffer) へと送られる。ROB を複数持つシステムを ROS(Read Out System) と呼ぶ。ここで粒子情報はハイレベルトリガーの前半、レベル 2 トリガー判定を待つ。

4. レベル 2 判定を通過した粒子情報はイベントビルダーに送られ、EF(Event Filter) によって最終的なトリガー判定がされる。

以上で述べたように各検出器を統合して ATLAS 全体として動作させるためには、次のような ATLAS 共通の規則に従ってエレクトロニクスを構築しなければならない。

- レベル 1 バッファを備えること
- デランダマイザを備えること
- ROD を備え、ATLAS 共通のデータフォーマットに変換すること
- データを転送する専用のリンクまたはバスを持つこと

全ての検出器データ収集システムはこれらの規則に則って設計されている。本論の中心の一つである読み出しシステムのアップグレードは、TGC に組み込まれるこの ROD に関してのものである。TGC のデータ収集エレクトロニクスについての詳細は次章で改めて取り上げる。

ATLAS 検出器全体に関する章の最後に、ATLAS で期待される物理について触れておく。

2.3 ATLAS で期待される物理

ATLAS は前述の通り汎用検出器であるために、測定可能な期待される物理現象は多岐に渡る。代表的なものには以下のものが挙げられる。

1. 標準理論の検証
    - W ボソンやトップクォークの精密質量測定

2. Higgs 粒子の探索
    - 標準理論 Higgs 粒子の探索
    - Higgs 粒子の質量、生成断面積の測定
3. 標準理論を越える物理の探索
    - 超対称性粒子の探索
    - 余剰次元の探索

上記の中でも特に注目される、標準理論 Higgs 粒子について簡単に述べることにする。

2.3.1 標準理論 Higgs 粒子

Higgs 粒子はボソンとフェルミオンに質量を与える未知の粒子である。ATLAS 実験の主要な目的のひとつはこの Higgs 粒子の発見である。Higgs 粒子の生成過程と崩壊過程について述べる。

Higgs 粒子の生成過程

Higgs 粒子は重い粒子と結合しやすいため、主に 4 つの生成過程が考えられる。それぞれのファインマンダイアグラムを図 2.12 に示す。

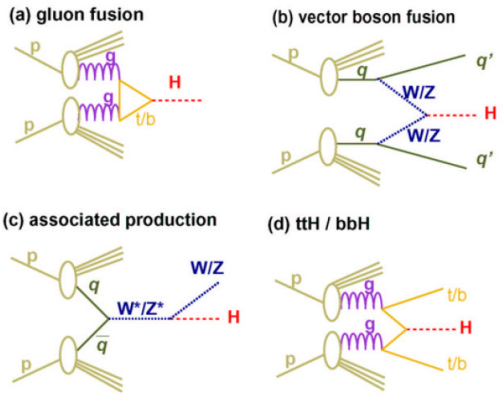


図 2.12: Higgs 生成のファインマンダイアグラム

1. gluon fusion
   トップクォークやボトムクォークを介した過程で最も断面積が大きい。しかし大きな横運動量を持つ粒子がないため、バックグラウンドとの選別が難しい。この生成の場合は崩壊過程で生成される粒子によってバックグラウンドを選別することが重要で、$H \to \gamma\gamma or ZZ or W^+W^-$ が有望な崩壊過程である。

2. vector boson fusion(VBF)
   クォークから放出されたゲージボソンによって生成される。断面積は比較的大きく、また反跳したクォークに起因する大きな横運動量を持つジェットが 2 本観測されるのが特徴である。さらに 2 つのクォークの間ではカラー交換が行われないので QCD バックグラウンドによる影響も少ない。

3. W/Z associate production
   クォークの対消滅で生成されたゲージボソンから更に Higgs 粒子が放射される過程である。終状態に W/Z を含む。これらがレプトンに崩壊した場合はバックグラウンドとの識別が容易にできる。

4. top associate production
   対生成されたトップクォークから Higgs 粒子が放出される。断面積は小さいが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。またこの反応には、トップクォークの湯川結合 (Higgs とクォークとの結合) という重要な情報を含んでいる。

Higgs 粒子の崩壊過程

崩壊過程は Higgs の質量に依存しており各領域で特徴的な崩壊過程が存在する。また、最近 Tevatron によって 170GeV 以上の Higgs の存在が排除された。170GeV 以下での Higgs の主要な崩壊は次の 3 つである。

1. $H \to \gamma\gamma$
   150GeV 以下で有効なチャネルである。この崩壊ではエネルギー及び角度分解能の優れた電磁カロリメータが必要である。

2. $H \to \tau\tau$
   150GeV 以下で有効なチャネルである。$\gamma\gamma$ よりも崩壊確率が高く、W/Z fusion の生成過程を考えることでバックグラウンドと区別することができる。τ の崩壊にはニュートリノが含まれるので消失横エネルギーの精度が重要になる。

3. $H \to ZZ^*$
   120GeV から 180GeV で有効なチャネルである。このモードは綺麗なピークが得られる。一つのレプトン対に対しては不変質量 m_Z に等しいという条件を課すことが出来るが、Z^* が仮想粒子であるためもう一つのレプトン対の不変質量には制限がない。そのため、検出器には運動量及びエネルギーに対する高い分解能が求められる。

ミューオンは物質の透過力が高く寿命が長いため検出がしやすく、また多くの終状態に含まれる。そのためミューオンを検出して確実にデータとして読み出すことは物理解析において重要である。

次の章では前後方ミューオン検出器、特に読み出しシステム部を中心に述べる。

第3章　前後方ミューオン検出器:TGC

この章では TGC システムの読み出しシステムを中心に述べる。まず TGC の構造及び配置について述べ、その後に TGC エレクトロニクス及びデータ処理システムについて述べることにする。

3.1　TGC 検出器の構造及び配置

3.1.1　構造

TGC(Thin Gap Chamber) は MWPC(Multi-wire Proportional Chamber) の一種であり、アノードのワイヤとカソードのストリップを用いて 2 次元読み出しが可能である。内部構造を図 3.1 に示す。

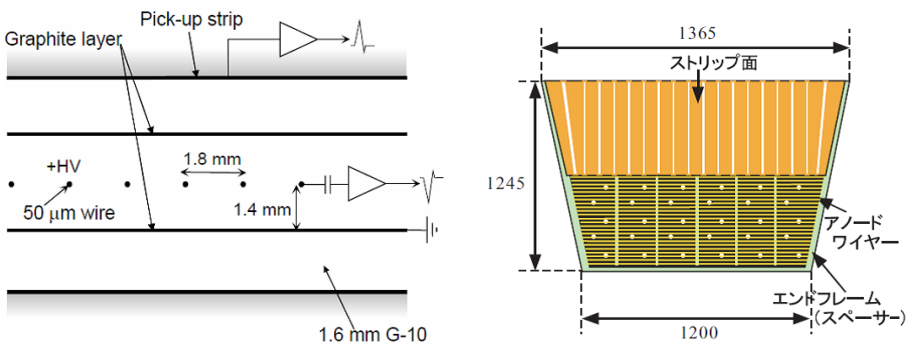


図 3.1: TGC の内部構造　(左)　断面図 、　(右)　平面図

典型的な MWPC は、ワイヤの間隔が 2mm 程度、アノードとカソードの距離は 7,8mm 程度であるが、LHC の 40MHz という衝突頻度で動作させると同時に検出効率を落とさないように、ワイヤ間隔を 1.8mm、アノード-カソード間を 1.4mm として設計されているのが特徴である。

1 チャンネルとして扱われるのは、ワイヤが 4〜20 本 (幅にして 10.8〜36mm)、ストリップは 1 本で 1 チャンネルに対応し、エンドキャップ領域で 4mrad、フォワード領域で 8mrad に相当する幅 (15.1mm〜53.4mm) である。

使用されるガスは $CO_2/n-pentane$(55 %/45 %) の混合ガスである。CO_2 が活性ガスとして働き $n-pentane$ が抑制ガスとして働く。CO_2 を活性ガスとして用いているのは CO_2 の電子捕獲断面積が小さいため高電場でも電子のドリフト速度が飽和せず、ドリフト時間を短く抑えることが可能だからである。それゆえ TGC は粒子通過による信号の立ち上がりが 5 ナノ秒以内と高速であり、LHC での高衝突頻度でもバンチを識別できる。$n-pentane$ は入射粒子によって励起された分子が基底状態に戻るときに発生する紫外線を吸収する効果があり、放電を起こしにくくしている。なお初期のコミッショニングは CO_2 ガスのみで行われた。

3.1.2 TGC 検出器の配置

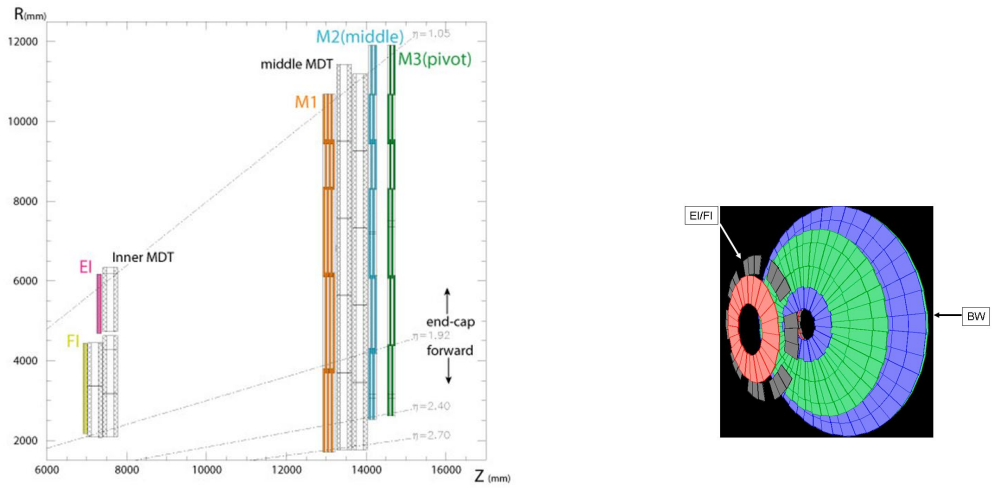


図 3.2: (左) TGC の配置 、(右) TGC の配置 (3D)

TGC システムは衝突点からみて、内側から EI/FI(Endcap Inner 及び Forward Inner)、TGC1(M1)、TGC2(M2)、TGC3(M3) の順に 4 ステーションが配置されている。反対側のサイドにも 4 ステーション配置されているので、つまり ATLAS 全体では 8 ステーションである。このうち TGC1,TGC2,TGC3 をビッグウィール (BW) と呼ぶ。EI/FI はビッグウィールよりも内側 (衝突点寄り) にあることから Inner Station と呼ぶことがある。また FI のみを指してスモールウィール (SW) と呼ぶ。なお FI 部は SLHC に対して高レートが予想されるため、アップグレードの必要があり現在 R&D が行われている。

各ステーションにある TGC は単層ではなく、TGC1 は 3 層 (Triplet) からなり、EI/FI、TGC2 及び TGC3 は 2 層 (Doublet) からなっている。ただし Triplet のストリップは 3 層ではなく 2 層である。つまり BW は片サイドだけで wire は 7 層、strip は 6 層である。

各層は 1/2 もしくは 1/3 チェンネル分ずらして配置させる。そのような多層構造にすることの理由は大きく次の 2 つがある。

- 各層でコインシデンスを取ることで、バックグラウンドになる偽信号の影響を減らせる。
- 各層間の位置をずらすことで、実質の位置分解能をチャンネル間隔より 2 倍もしくは 3 倍にできる。

BW は $1.05 < |\eta| < 2.7$ の領域をカバーする。$|\eta| < 1.9$ を後方部、$1.9 < |\eta|$ を前方部としている。EI/FI は $1.05 < |\eta| < 1.9$ の領域をカバーする。つまり BW の後方部の範囲をカバーする。EI に関しては構造上 acceptance が約 70 パーセント程度になっている。

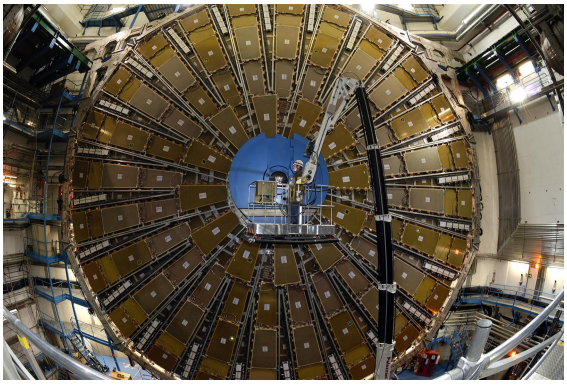


図 3.3: TGC(ビッグウィール)

ビッグウィールは 1/12 円のまとまりとして、それぞれ独立して動作させることが可能でありデータ処理もこの単位で行うことが可能である。この 1/12 円をセクターと呼ぶ。一つの BW のセクター例を図 3.4 に示す。

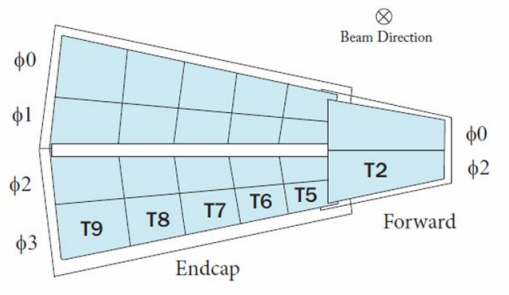


図 3.4: セクターの構成

一つのセクターは 4 φに分かれている。1 φは数枚の TGC からなり、その一枚一枚をチェンバーと呼ぶ (図中で T9...T2 のこと)。

TGC のチャンネル総数は約 32 万チャンネルあり、それらから情報を処理するために複数のエレクトロニクスを用いている。そこで次では TGC のエレクトロニクスとデータ処理について述べる。

3.2 TGC エレクトロニクス及びデータ処理システム

この節ではまず TGC でのエレクトロニクスによるデータ処理について述べる。その後にそれら各種エレクトロニクスについて具体的に述べる。

TGC システムには大きくわけて次の 3 つの流れがある。

1. トリガー系:
   TGC でのレベル 1 トリガー情報を処理する。なお他検出器を含めた最終のレベル 1 トリガー判定に利用されるのはビッグウィールのトリガー情報のみである。(ただし EI/FI もトリガー系のラインは持っている。)
2. 読み出し系:
   TGC のヒット情報を収集する。
3. 制御系:
   TGC をコントロールする。

TGC 検出器から流れてきたヒット情報はトリガー系及び読み出し系エレクトロニクスによって処理される。具体的には以下の順で通っていく。ATLAS 内で設置されている場所も併記しておく。

- トリガー系
  - ビッグウィール
    1. ASD(Amplifier Shaper Discriminator)-Board (TGC Chamber と接続)
    2. PS-Board (ビッグウィールの表面)
    3. Hpt(High-Pt)-Board (ビッグウィールの外周にあるミニラック)
    4. SL(Sector Logic)-Board (カウンティング・ルーム (USA15))
  - EI/FI
    1. ASD-Board (TGC Chamber と接続)
    2. PS-Board (EI/FI Chamber からやや離れて設置されているミニラック)
    3. SL (カウンティング・ルーム (USA15))
- 読み出し系
  - ビッグウィール
    1. ASD-Board (TGC Chamber と接続)
    2. PS-Board (ビッグウィールの表面)
    3. SSW-Board (ビッグウィールの外周にあるミニラック)
    4. ROD (カウンティング・ルーム (USA15))
  - EI/FI
    1. ASD-Board (TGC Chamber と接続)
    2. PS-Board (EI/FI Chamber からやや離れて設置されているミニラック)
    3. SSW-Board (EI/FI Chamber からやや離れて設置されているミニラック)
    4. ROD (カウンティングルーム (USA15))

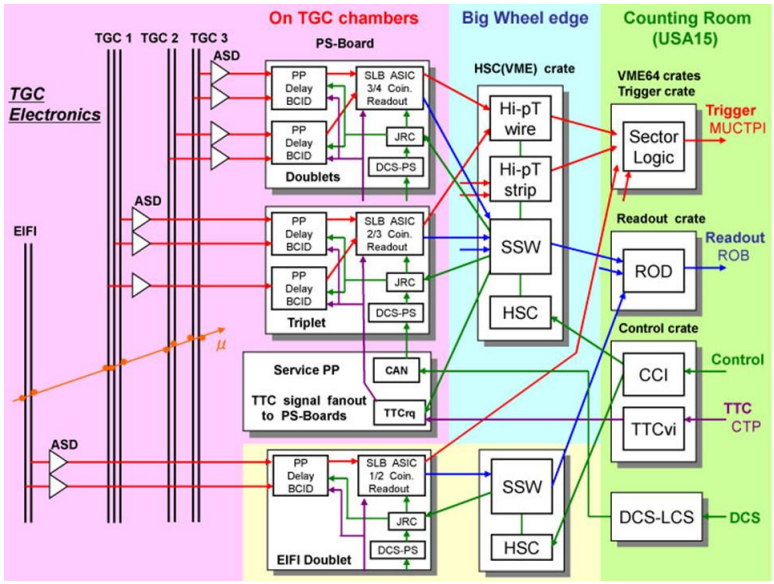


図 3.5: TGC エレクトロニクス

2 章で述べた TDAQ システムと図 3.4 の TGC エレクトロニクス図を踏まえて TGC でのデータ収集の大きな流れを述べると以下のようになる。

1. ビッグウィールでミューオンによる信号が発生するとその信号はデジタル化され、SLB(Slave Board) ASIC 内に到達する。この内部でトリガー系と読み出し系に情報は分岐する。

2. 読み出し系のためのヒット情報は SLB ASIC 内のレベル 1 バッファにおいて、トリガー判定による L1A 信号が来るまで待つことになる。その間、トリガー系に入ったヒット情報はトリガー判定を行うために SLB ASIC 内でデータ処理を施した後、下流のシステムへと流れていく。

3. トリガー系によって最終的にトリガー判定がなされると、L1A 信号は制御系を経て、SLB ASIC に到達する。このとき、該当するバンチデータだけでなく、その前後のバンチデータを含めた 3 バンチ分の情報が送られる。

4. SLB ASIC は L1A 信号を受けた後、ヒット情報を ASIC 内部でレベル 1 バッファからデランダマイザへ送り、次の読み出し系エレクトロニクスへと順次データを流し収集していく。

また TGC のレベル 1 トリガーシステムについては次のように行われる。

1. TGC で発生した信号が SLB ASIC に到達する。
2. TGC2 及び TGC3 の Doublet 情報は 4 層まとめて処理される。TGC2 と TGC3 を担当する SLB は、ワイヤとストリップ別々に 3/4 コインシデンス処理 (3 out of 4) を行う。TGC1 の Triplet 情報はワイヤでは 2/3 コインシデンス処理 (2 out of 3)、ストリップでは 1/2 コインシデンス処理 (1 out of 2) が行われる。情報は次の Hpt ボードへ送られる。
3. Hpt では SLB で別々に処理された Triplet 情報と Doublet 情報を統合してコインシデンス処理が行われる。Hpt ではワイヤとストリップとではまだ別々に扱われる。情報は次の SL へと送られる。
4. SL でワイヤ部とストリップ部の情報が統合されコインシデンス処理が行われる。ここで TGC のトリガー系の最終的な情報として、セクターごとに横運動量の大きな 2 つの軌跡が選び出される。SL での結果は次の MUCTPI へ送られる。
5. MUCTPI では RPC の情報と統合され、ミューオントリガー検出器としての最終的なレベル 1 トリガー判定が下される。

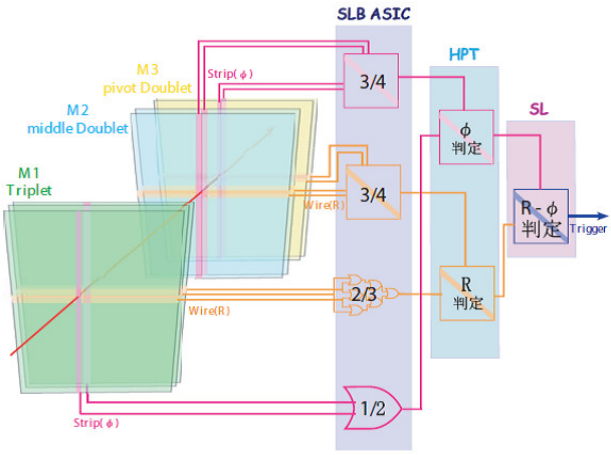


図 3.6: TGC でのレベル 1 トリガー

次ではトリガー系と読み出し系のシステムを構成する具体的なエレクトロニクスについて言及し、そしてそれらに伴うデータ処理について述べる。

3.2.1 各エレクトロニクス

各エレクトロニクスについて説明していく。本論の主題と深く関わる PS ボードと SSW、ROD については特に詳しく述べる。

ASD

ASD ボードは TGC Chamber に接続され、4 チャンネル分の処理を行う ASD ASIC が 4 つ搭載されている。つまり ASD ボードは 1 枚あたり 16 チャンネル分の信号を処理する。

ASD ASIC は次の機能を持つ。

- TGC Chamber から来るアナログ信号を増幅、整形し、設定されている閾値電圧を越えた信号のみを LVDS レベルの信号として出力する。
- トリガー信号を受けて擬似的な TGC 信号 (テストパルス) を出力できる。

テストパルス用トリガーは PS ボードから供給される。

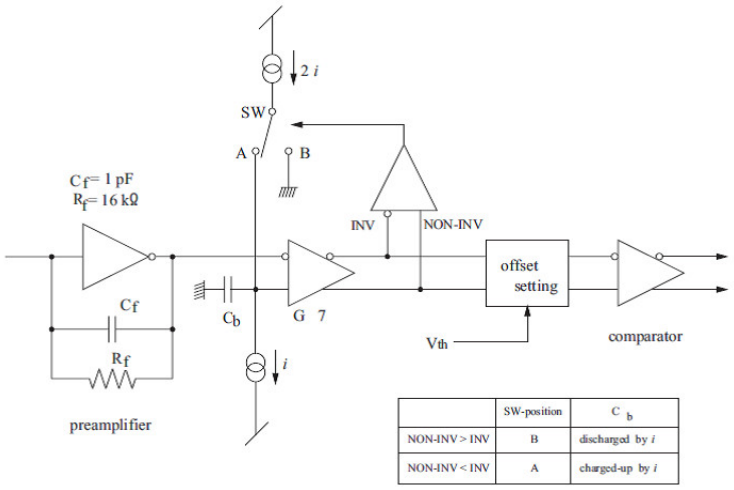


図 3.7: ASD ASIC のブロック図
信号の増幅やデジタル化を行う。

PS Board

ASD の信号は PS ボードに届けられ、担当するステーション及び領域に応じて多層コインシデンス情報を生成する。また PS ボードは各種類の TGC チェンバーからの読み出しチャンネル数のバラつきを吸収し、データフォーマットを整える役割を担う。その結果、その後の情報処理が画一化される。以上の理由から PS ボードには複数のタイプがある。表に示す。

Big Wheel	Triplet	Doublet
Endcap Wire	EWT0,1,2	EWD0,1,2,3,4
Endcap Strip	EST	ESD0,1
Forward Wire	FT0,1	FWD0,1
Forward Strip		FSD

EIFI	Doublet
EI Wire	EIFI
EI Strip	
FI Wire	
FI Strip	

表 3.1: PS ボードの種類

これらは配線が異なるだけで使用する IC は同じである。PS ボードには Patch Panel(PP) ASIC 及び SLB(Slave Board) ASIC が搭載されている。トリガー系情報と読み出し系情報は SLB 内で分岐する。これら ASIC についてそれぞれ説明する。

- PP ASIC
  PP ASIC は最大 32ch の処理を行う。TOF や各 ASD-PS ボード間をつなぐケーブル長の違いによって、ASD からの信号がそれぞれの PP に到達するのは同じタイミングではない。またチェンバーの重なる部分では粒子情報をダブルカウントしてしまう可能性がある。そこで ATLAS のような検出器には、タイミング調整、バンチ情報の識別、信号のダブルカウントを防ぐ、などの機能を持ったエレクトロニクスが必要であり、TGC では PP ASIC がそれにあたる。

  具体的な機能は以下のとおりである。

  - LVDS レベルの信号を CMOS レベルの信号に変換した後、variable delay 回路にて各チャンネルに 0 25ns の delay をかけることで、タイミングを揃える。
  - タイミングの調整がされた信号は BCID(バンチクロッシング ID) 回路にて、シグナル系の TTC から供給される LHC clock と同期が取られ、バンチの識別が行われる。すなわちこれ以降、信号は LHC クロックと同期して扱われるようになる。
  - TGC チェンバーが重なる部分でのワイヤ方向信号のダブルカウントを防ぐため、その領域でのワイヤ信号の OR を取り (OR ロジックと呼ぶ)SLB ASIC へ送る。
  - ASD に対してテストパルストリガーを出力する。
  - SLB に対して LHC クロック、L1A などの信号を供給する。

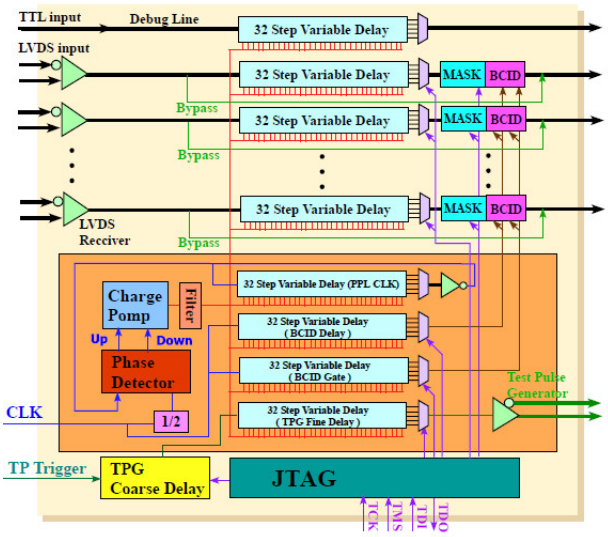


図 3.8: PP ASIC のブロック図

Delay 調整やテストパルス回路が搭載されている。1 つの PP ASIC で 32 チャンネル分を処理することができる。

- SLB(Slave Board) ASIC
  SLB ASIC ではトリガー系と読み出し系に分かれる。SLB で処理されたトリガー系情報は Hpt ボードへ (EI/FI の場合は SL へ)、読み出し系情報は SSW ボードへそれぞれ送られる。ブロック図を図 3.9 に載せる。

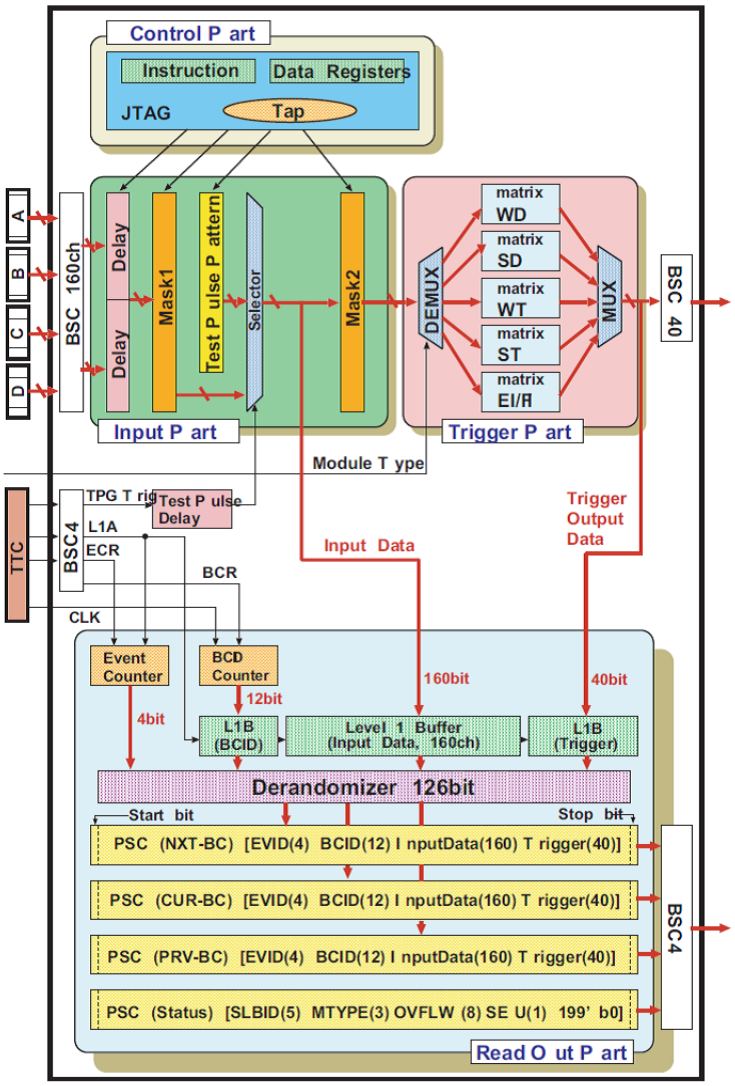


図 3.9: SLB ASIC のブロック図
PP からの入力を受け、トリガー部分と読み出し部分に分かれている。

SLB 内のトリガー系部分と読み出し系部分についてそれぞれ述べる。

- トリガー系部分
  PPASIC から届いた信号によってコインシデンス情報を生成する。その情報は Hpt ボードへと届けられる。
  なお、EI/FI に関しては Hpt ボードを介さずに直接 SL ボードへと情報が送られる。
- 読み出し系部分
  レベル 1 トリガーの判定によって L1A 信号が届けられるまで信号情報は一旦ここで保

持される。複数の ASD からの粒子ヒット情報は、ここで 160 ビットのヒットマップになる。ヒットマップの例を図 3.10 に載せる。

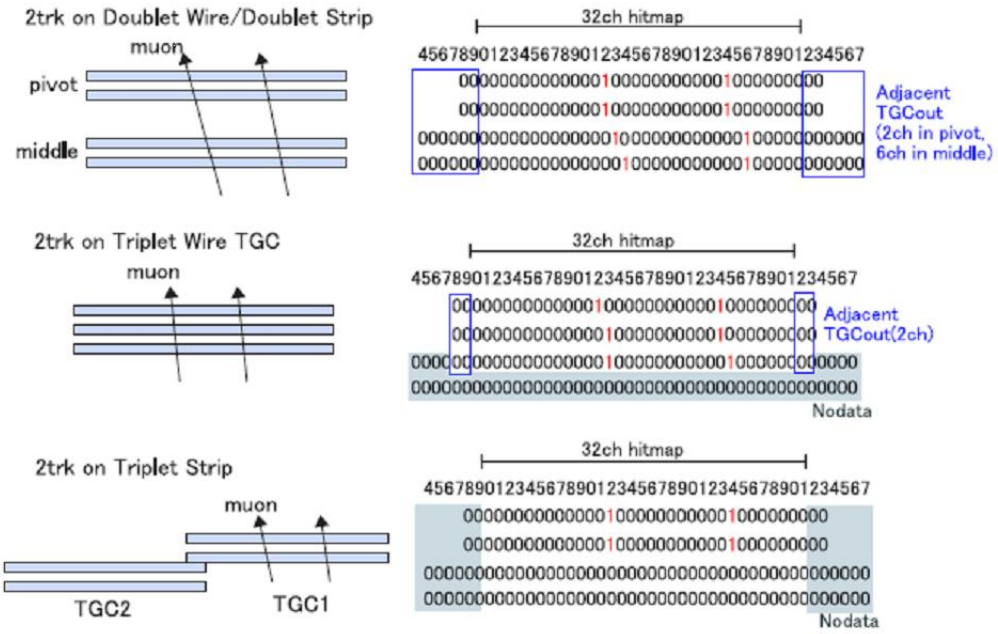


図 3.10: ヒットマップ
ヒット情報は同じ層の隣接するチェンバーの
チャンネル情報を加え、それぞれ 160 ビットのヒットマップを作る。

L1A 信号が届くと対応したデータが、その前後 1 バンチを含めた 3 バンチ分の情報が SSW ボードへと送られる。図 3.11 に示す。

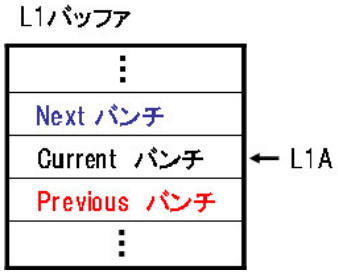


図 3.11: L1 バッファと L1A 信号
L1A 信号に対応したバンチ情報に加えその前後のバンチ情報が読み出される。

SLB からのデータ出力フォーマットを図 3.12 に載せる。1 つの SLB から読み出し系に流れる情報はヒットマップ情報 3 バンチ分 (384 ビット) にトリガー系のデータ処理結果 40 ビット、そしてステータス情報などを加え計 572 ビットになる。送信には LVDS 信号が使われ 40Mbps で送信される。しかし L1 トリガーレートが最大 100kHz になった場合、この送信速度では処理しきれなくなるため、実際は 2 つの送信ラインを利用することで実質的な送信速度を 80Mbps にしている。このことにより PS ボードの送信能力は 130kHz までのレベル 1 トリガーレートに耐えられるようになっている。

前述のように PS ボード以降はトリガー系と読み出し系で通るエレクトロニクスが異なる。まずトリガー系エレクトロニクス部を述べてから読み出し系エレクトロニクス部について述べる。

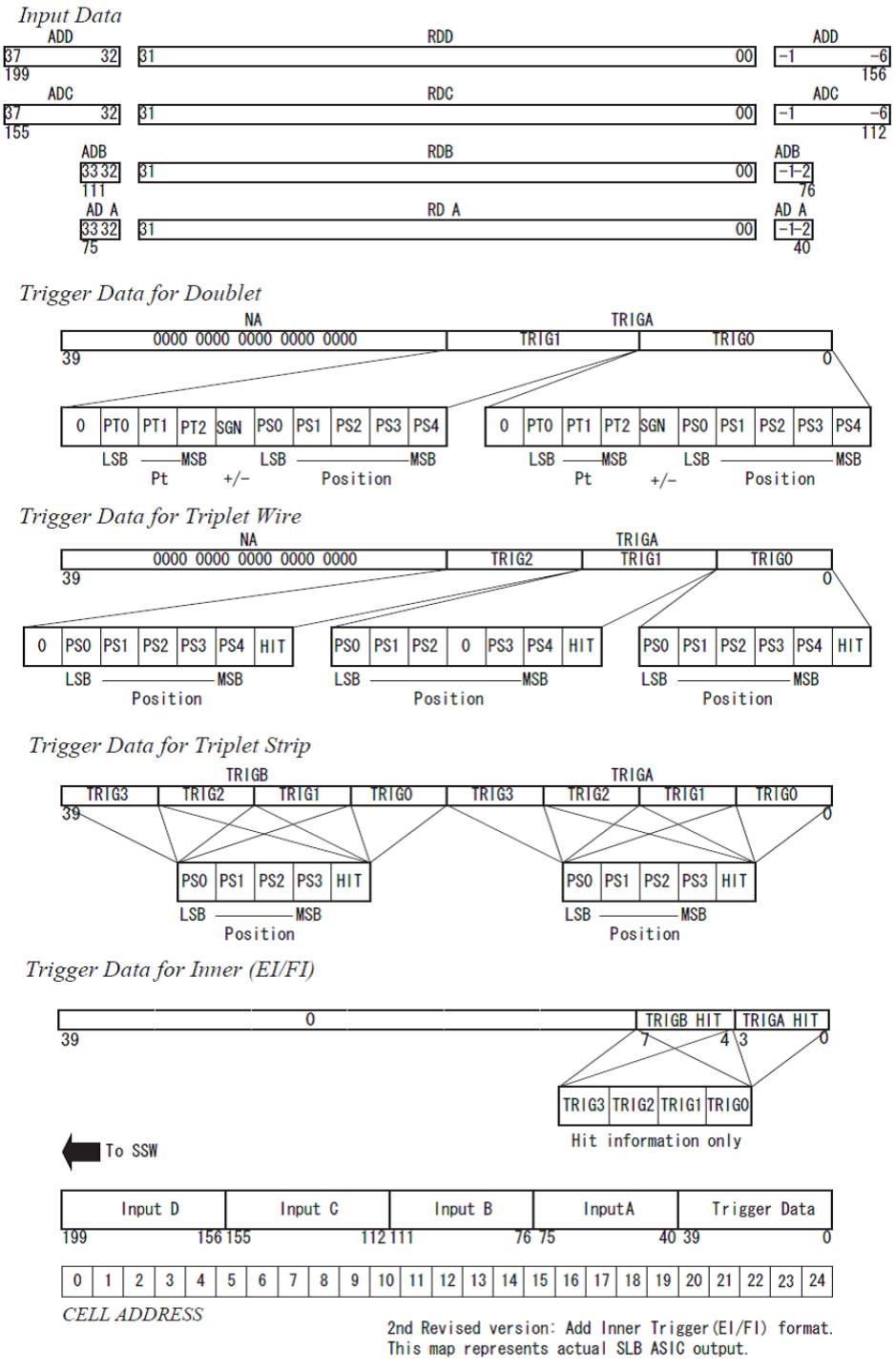


図 3.12: SLB の出力フォーマット

Hpt(High-Pt(トリガー系))

Hpt は Doublet と Triplet の情報を用いてコインシデンス情報を生成する。この段階ではまだ r 方向の情報と φ 方向の情報は分けられて処理される。コインシデンス情報は続いて SL へと送られるが、SL は 100m 程離れたカウンティング・ルーム (USA15) にあるため、出力には G-Link と呼ぶ光通信で転送される。G-Link については後の SSW の項で述べる。

SL(Sector Logic(トリガー系))

SL は Hpt ボードから届けられた r 方向の情報と φ 方向の情報を用いてコインシデンス情報を生成する、TGC のトリガーデータが最終的に集まるモジュールである。SL は生成したコインシデンス情報によってミューオンの軌跡を構成し、それらのミューオンを 6 段階の横運動量に分けることで分類する (Pt 判定)。そして最終的にそれらの中から横運動量が大きなミューオンを 2 つ選び、その 2 つのミューオン情報の Pt 判定と位置情報 (RoI) を MUCTPI と呼ぶ、RPC と TGC のトリガー情報を CTP へまとめて送るシステムへ渡す。

SSW(読み出し系)

PS ボードからの情報は SSW に届けられる。SSW の主な役割は複数の SLB からのデータを圧縮してまとめ、続く ROD へと情報を送ることである。圧縮する利点は PS ボード上の SLB で生成されたヒットマップの中身がほとんどがゼロであり、その不必要な部分を落とすことで大幅に情報量を軽減できるからである。

具体的には以下のように圧縮を行う。

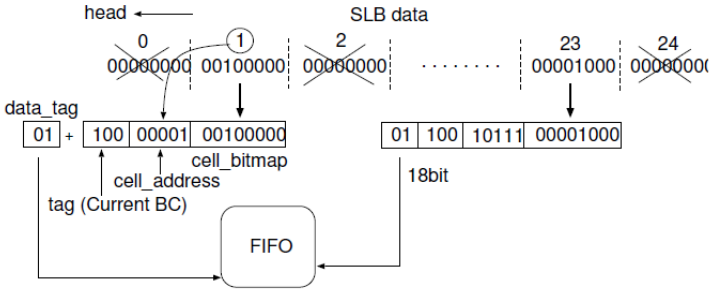


図 3.13: SSW でのデータ圧縮:ゼロサプレス

1. ヒットマップを 8 ビットごとに区切りながら読む。
2. その 8 ビットが全て 0(Low) の場合、そのデータを捨てる。
3. 全て 0 以外のとき、すなわちヒットがある 8 ビット列は、前後含めた 3 バンチのどれかを示すタグ (3 ビット) と区切った 8 ビット列群の中の位置を示すセルアドレス (5 ビット) を対象の 8 ビットの頭に付加し、合計 16 ビットのデータとして扱う。

1 セクターで最大 10 枚の SSW が使用される。セクター内で各 SSW が担当する領域を示した対応表を載せる。

	SSW0	SSW1	SSW2	SSW3	SSW4	SSW5	SSW6	SSW7	SSW8	SSW9
SLB数	18個	18個	10個	15個	15個	15個	15個	10個	11個 又は 12個	6個
担当領域	Endcap ϕ 0/1	Endcap ϕ 2/3	Forward	Endcap ϕ 0	Endcap ϕ 1	Endcap ϕ 2	Endcap ϕ 3	Forward	EI/FI (セクター 2/5/8/11)	SL
	TGC1			TGC2/3					EI/FI	SL

図 3.14: セクター内で各 SSW がカバーする領域及び処理する SLB の数

圧縮したデータはヘッダーとトレイラーが付加され ROD へと送られる。ROD は SL と同じくカウンティング・ルームにあるため出力には G-Link による光通信で転送される。

- G-Link
  Agilent 社の HDMP1032A(送信用:Tx)/HDMP1034A(受信用:Rx) の IC を使用している。TGC の場合は Tx チップが SSW 及び Hpt、Rx チップが SL そして次に述べる ROD に搭載されている。Tx へ入力された 16bit のパラレル信号はシリアルに変換された後、光信号として Rx チップに向けて出力される。このとき外部クロックに対して 20 倍の速度でデータは出力されるため、シリアルにすることによる遅延は起こらない。Rx はその光信号を受け取り元のパラレル信号に戻す。データ転送速度は最高 1120Mbps である。TGC においては LHC クロックが 40MHz なので転送速度 640Mbps で使用されている。

SSW が出力するデータフォーマットを図 3.15 に載せる。

ヒット情報量によって SSW から出力されるデータサイズは当然変化するが、ヒット情報を持たない、いわば最小のデータサイズは各 SSW の受け持つ SLB の数に比例する。つまり最大の最小データサイズは SSW0 及び SSW1 であり約 160Byte になる。また SSW10 個での最小データサイズは 1272Byte である。現行 LHC で予想される衝突あたりの TGC ヒットレートはおよそ 50Hits(チャンネル) であり、これから予想される 1 枚の SSW が持つ平均データ量は約 130byte と見積もることができる。SLHC の場合はバンチあたりの陽子数が現行と同程度と仮定するとヒットレートはおよそ 500Hits であり、それに伴う SSW の平均データ量は 170byte 程度と見積もられる。

またゼロサプレスの性質を考えると、総チャンネル数に占めるヒット情報の割合が増えるにつれゼロサプレスによるデータ圧縮効果は小さくなる。粗く見積もると 1 つの SLB 内の 160 ビットのうちランダムに 10 箇所程度のヒット情報がある場合、タグやセルアドレスによる情報付加もありデータ圧縮効果はほぼ望めないことになる。しかし以前行われたシミュレーションによると 1 イベントあたりのヒット情報は総チャンネル数の約 0.02 パーセントであり、ゼロサプレスによるデータ圧縮は極めて有効であると言える。

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	LVDSInk	RX error state

LVDSInk=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_frag. 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 3.15: SSW の出力フォーマット

ROD(Read Out Driver(読み出し系))

SSW の情報は ROD に届けられる。TGC の読み出しデータが最終的に集まるモジュールである。ROD までは各検出器の特性に合わせたデザインが許されているが、それに続く ROB 以降は ATLAS 共通である。すなわち ROD は複数の SSW から来る情報をまとめ ROS が要求するフォーマットに整形して出力する役割を担う。出力には CERN で開発された S-Link と呼ぶ光通信によって ROB へ転送する。現行の ROD は 1 枚で 1 セクターを担当しており、つまり SSW の出力を最大 10 個扱う。

- S-Link
  S-Link は CERN で開発されている光信号のリンクモジュールである。1 クロックで 32bit のパラレルデータを送信、受信できる。現在 ATLAS で使用されている S-Link の最高データ転送速度は 1Gbps である。最新の S-Link は 2Gbps の性能を持つ。

ROD の出力が従わなければならないイベントフォーマットは以下の通りである。

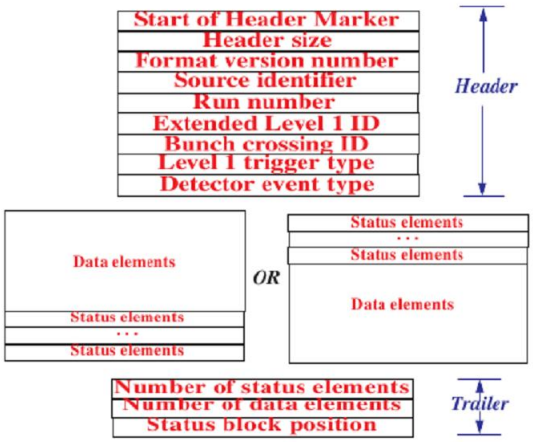


図 3.16: ROD に要求されるデータフォーマット

ヘッダーは次のものからなる。

1. Start of Header Marker: ROD データの先頭であることを示す。0xee1234ee と決められている。
2. Total Header Size: ヘッダーの行数を示す。
3. Format Version Number: フォーマットバージョンを示す。現在は上位 16bit を ATLAS の format version、下位 16bit を TGC の format version とし、どちらも 0x0300 と決められている。
4. Source Identifer: データの起源を示す。現在は 8bit の sub-detectorID と ModuleID からなる。sub-detectorID は TGC データの場合、A-side が 0x67、C-side が 0x68 となっており、ModuleID は TGC のセクター番号を示す。

5. Run Number: 実験回数に応じて割り振られる値。
6. Extended Level 1 ID: 24bit の L1ID と 8bit の ECRID からなる。
7. Bunch Crossing ID: 12bit のバンチ識別値。
8. Level 1 Trigger Type: 8bit
9. Detector Event Type: 追加情報を載せられるヘッダー行である。

データ部はステータスの部分とヒット情報の部分からなる。検出器によってステータスとデータの順番は異なり、TGC では現在ステータス部を先にしている。ヒット情報の部分は TGC の場合、複数の SSW の出力をつなげてまとめた構造になっている。

トレイラーは次のものからなる。

1. Number of Status Elements: ステータスの行数を示す。
2. Number of Data Elements: データの行数を示す。
3. Status Block Position: データとステータスの前後関係を示す。ステータスが先であることを示すのは 32 ビット全てが 0 の場合である。

ROD は SSW から受け取った後、以上の情報を付加し整えた上で ROB へ出力しなければならない。SSW10 個の入力を考えると 1 つの ROD が処理しなければならないデータ量は 1300byte 程度である。また SLHC の場合は 1700byte 程度と考えることができる。

ここまで述べてきたように検出器とは、検出器自体に加え複数の種類からなるエレクトロニクスから構成されている。これらを安定に動作させるには、段階的に動作検査を繰り返し確実性を高めていかなくてはならない。
次章では読み出しシステムの動作検査について述べることにする。

第4章　TGC読み出しシステムの動作検証

ここでは地上で行った読み出しシステムの各種動作検査について述べる。

4.1　地上におけるTGC動作試験

ATLAS 実験は地下 100m に設置されるため、地下配置後の検出器へのアクセス環境が良くない。そのため検出器システムの不具合は地下へ設置する前になるべく発見し、修復・対処を行うことが効率の点及び実験開始後の ATLAS のパフォーマンスを下げないために必要である。

そこで地上において、TGC システムのうち未試験部であった EI/FI 用 PS ボード検査と FI の動作検査を行った。EI/FI は衝突点以外から発生するバックグラウンドの除去やトラッキングのための Inner Station におけるϕ情報を提供する役割を担う。FI は 2007 年の暮れには地下へインストールされる予定であったが、その年の夏の時点でまだ検査がされておらず、早急に検査ステーションを構築し、検査をする必要があった。

段階的に次の動作検査を行った。

1. EI/FI 用 PS ボードの動作検査
2. FI のテストパルスを用いた動作検査
3. FI の宇宙線を用いた動作検査

チェンバーとエレクトロニクスを統合して検査する前に、まず別々に検査して単体できちんと動作するかを検査する。EI 及び FI に関して、PS ボードの単体検査がまだ行われていなかったため行う必要があった。

PS ボード検査完了後に、チェンバーとエレクトロニクスを接続した上でその接続が間違っていないかをテストパルスを用いて確認した。特に ASD から PS ボードへと向かうケーブルは構造体の周囲に張り巡らされており、そのままの配線で地下のインストールされるためこの検査できちんとその配線を確認することは重要であった。

配線が確認できた上で、宇宙線を用いてチェンバー及びエレクトロニクスを統合させた検査を行った。

それぞれについて述べる。

4.1.1　EI/FI 用 PS ボードの動作検査

ここでは PS ボード動作検査 (及び地上動作検査) に利用したテストパルスの概要について述べ、その後検査システム及び検査方法について、最後に検査結果について述べる。

テストパルス

2 章で述べたように TGC エレクトロニクスには擬似信号出力、つまりテストパルスを生成する機能が搭載されている。これは名前の通り TGC システムを検証 (テスト) する際に使用される機能である。TGC テストパルスには 2 種類あって、それぞれ SLB テストパルス (SLBTP) と ASD テストパルス (ASDTP) と呼んでいる。

SLBTP と ASDTP にわけて述べる。

- SLBTP
  テストパルストリガーが発行されると擬似信号、すなわちテストパルスが SLB 内で生成され、そのまま SLB 内のパイプラインメモリに入る。その後 L1A が発行されると、パイプラインメモリからデランダマイザへとテストパルスが送られ、データとして読み出される。

- ASDTP
  テストパルストリガーが発行されると、PPASIC が ASD にテストパルスを生成して出力させる信号を出す。それを受けて ASD はテストパルスを生成し出力する。テストパルスは PPASIC を経て SLB に到達しパイプラインメモリに入る。その後 L1A が発行されると、パイプラインメモリからデランダマイザへとテストパルスが送られ、データとして読み出される。

それぞれのテストパルスの経路を図 4.1 に示す。なお、より詳細な図は 4 章の SLB の項に掲載してある。

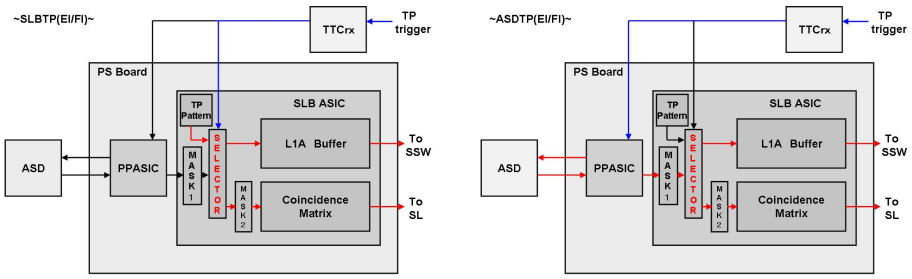


図 4.1: (左) SLBTP(EI/FI)、 (右) ASDTP(EI/FI)

まず TTC がテストパルストリガーを出す。SLBTP の場合はテストパルストリガーを直接 SLB ASIC で受け、L1A バッファにテストパルスが入る。ASDTP の場合にはテストパルストリガーを初めに PP ASIC で受ける。その後 ASD へテストパルスを発生させる信号を送る。結果、ASD からテストパルスが出力され PP ASIC を再び通り、SLB 内の L1A バッファに入る。

読み出し検査におけるテストパルストリガー信号と L1A 信号との関係図を図 4.2 に示す。

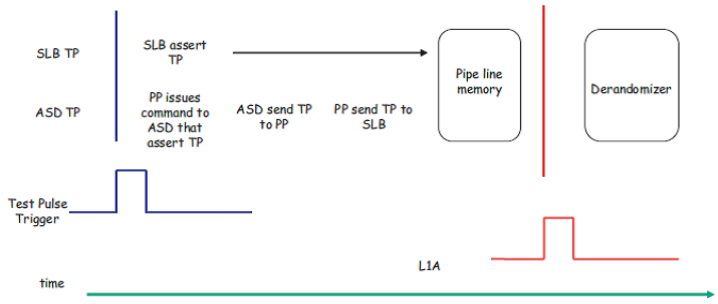


図 4.2: テストパルスと L1A の関係
テストパルスはまずパイプラインメモリ (L1A バッファのこと) で保持される。L1A が来るとデランダマイザへ送られる。

両パルスとも SlB 以降の経路は同じであり、両者の違いのうちこの検査で重要な意味を持つのは PPASIC を信号が通過するかしないかという点である。すなわち SLBTP は検査対象 IC が SLBASIC のみであるのに対して、ASDTP は PPASIC 及び SLBASIC の 2 種を含む。この違いを活かすことで検査領域を絞ることができ、不良の疑いがある場合に不良箇所特定までの時間を短縮させることができる。この 2 種のテストパルスは SLBASIC の TestPulseVETO というレジスタ設定によって選択される。図 4.3 に示しておく。

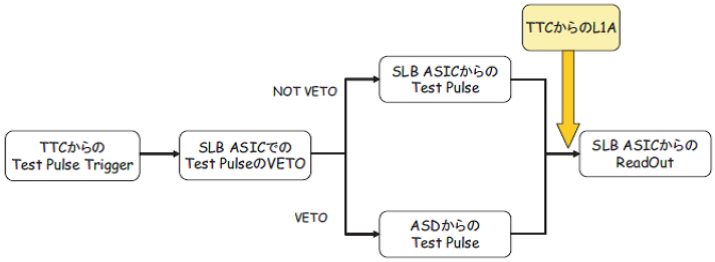


図 4.3: テストパルスの種類の決まり方

PS ボード検査システムの構築及び検査方法

検査は主に CERN 研究所 Bld.188 の 2F にあるテストベンチスペースで行った。ここでは検査システム及び検査方法について述べる。検査システムのセットアップ図を図 4.4 に示す。PS ボードは SSW から制御される。

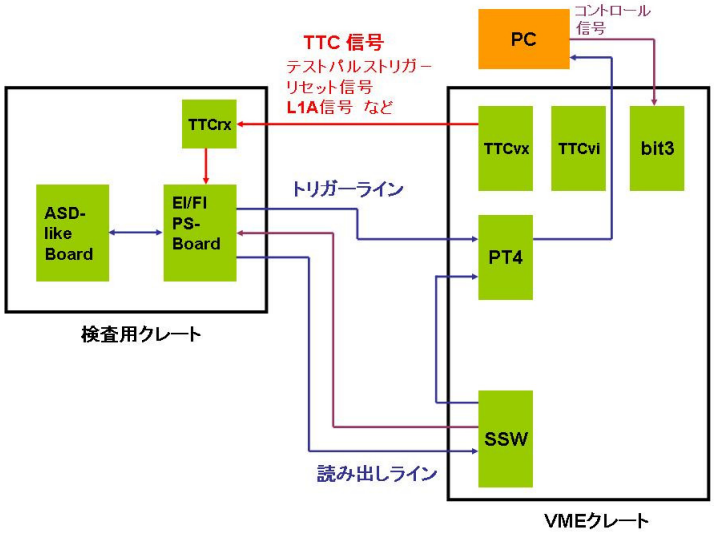


図 4.4: PS ボード動作検査システム

VME クレートに TTC や SSW などのモジュールを集め、それらのモジュールを bit3 を通じて PC から操作する。bit3 とは PC と VME クレートをつなぐバスアダプタである。また PT4 とは CPLD と FPGA を搭載した汎用モジュールである。

検査手順は以下である。

1. まず PS ボードに載っている SLBASIC や PPASIC へアクセスできるかどうかを SSW を用いて検査する。アクセスできなければ電源ラインが PS ボード上でショートしていたり、SSW-PSB 間のコネクションがしっかりできていないことが疑われる。

2. 続いて SLBTP によって SLBASIC 及びそれ以降の読み出しラインの検査を行う。チェンネルに欠けが見られたり、信号の予期しない遅延がある場合には SLBASIC の不良であることが考えられる。

3. 次に ASDTP によって PPASIC 及び SLBASIC までのラインの検査を行う。SLBTP による検査に合格してもここでチャンネルの欠けや信号遅延などが確認できた場合には PPASIC の不良が考えられる。

4. そして最後に ASDTP によってトリガーラインの検査を行う。Mask を利用してパルスデータをトリガー用、つまり実際のヒットのような形に成形する。L1A シグナルが発行されない場合は SLBASIC のトリガー部の不良が考えられる。

検査の方法の流れを図 4.4 に示す。

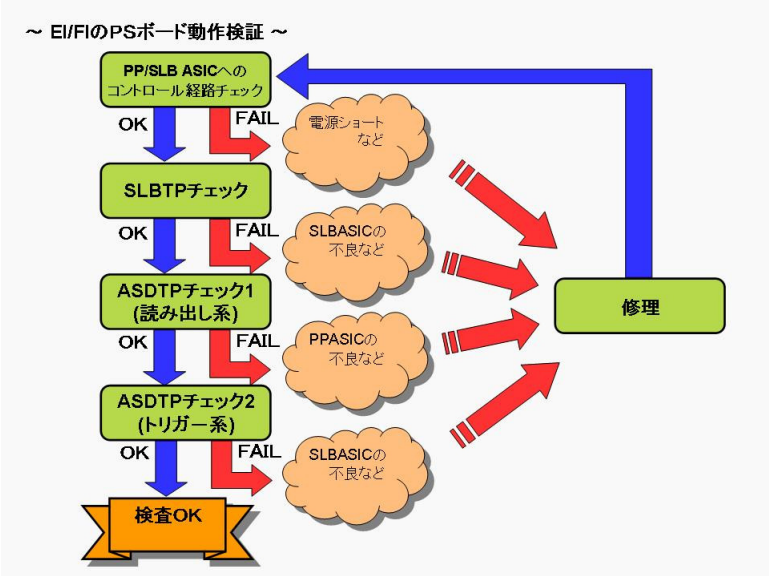


図 4.5: EI/FI の PS ボード動作検査の流れ

結果

予備も含めた EI/FI 用 PS ボード 54 枚に対して各検査を行った結果、SLB ASIC の不良など 6 枚の不良ボードを発見することができた。その成果により地下へのインストール前にそれらボードに対して修理を施すことができた。

4.1.2　FI のテストパルスを用いた動作検査

ここでは ASD テストパルスを用いた動作検査について述べる。EI はスケジュールの都合からすでに地下へインストールされていたため、FI のみの動作検査を行った。まず検査システム及び検査方法について述べ、その後結果を述べる。

検査システムの構築及び検査方法

FI は片サイドに 24 チェンバーあり、全部で 48 チェンバーある。地上でそれら FI チェンバーを SW 用の円盤構造体上に設置し、ASD から PS ボードへと続くケーブルを配線した。

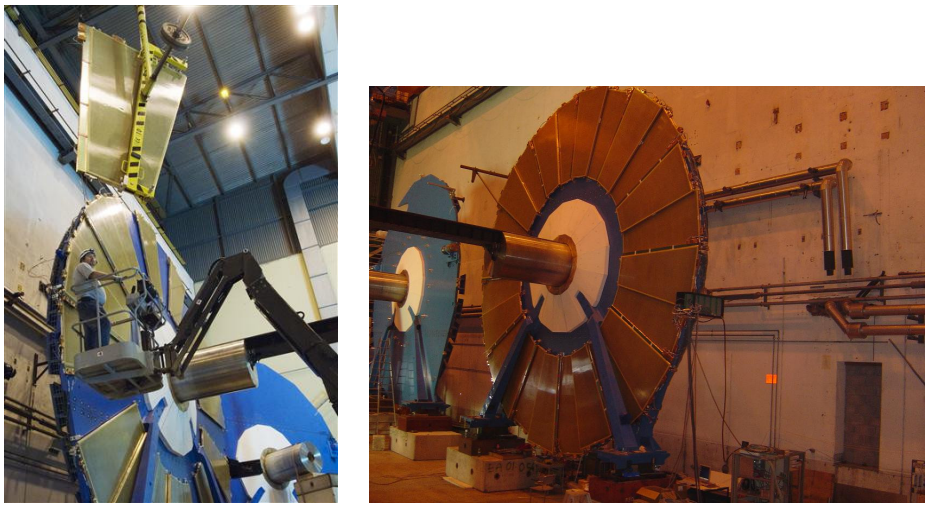


図 4.6: (左) SW セットアップ中の FI、 (右) 構造体への設置が完了した FI

エレクトロニクスのセットアップを図 4.7 に載せる。

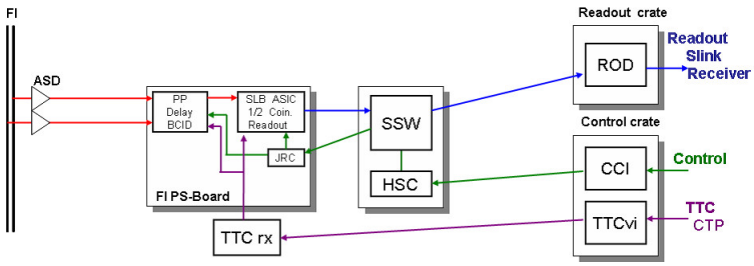


図 4.7: FI 地上検査のセットアップ概要

エレクトロニクスが全体として正しく動作するかを確認する。そしてケーブルの配線チェックを行う。ASD テストパルスを出し、そのデータを SSW を通じて ROD で読み出す。

ASD テストパルスによって検査可能な項目は主に次の点である。

- ASD が正しく動作するか
- ASD-PS ボード間のケーブルが断線していないか

- ASD-PS ボード間のケーブルが正しく配線されているか

検査は CERN 研究所の Bld.191 において行った。

結果

全チャンネルに対して ASDTP を行った。その結果、問題のあった箇所を示す。

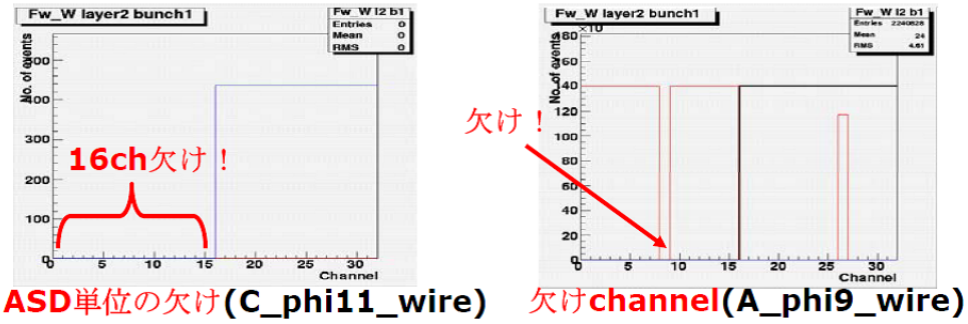


図 4.8: ASDTP 結果　(左)16 チャンネルの欠損　(右)1 箇所の欠損チャンネル
縦軸がヒット数、横軸がチャンネル数である。

A サイドにおいて欠損チャンネル 1 箇所を発見した (右の絵)。これについては Mask をするなどの対処をすることにした。他には 16 チャンネルのまとまった欠けが見られたが、多くの場合はケーブルの挿しかたが甘いなどが原因で、C サイドの 1 箇所 (左の絵) 以外は全て解決することができた。なお、この欠けについても宇宙線の検査で解決された (次節)。

ここまでの検査で ASD 以降の検査は終了したことになる。続いては検査検査領域を拡張して、チェンバーの動作までを検査対象にする宇宙線動作検査について述べる。

4.1.3　FI の宇宙線を用いた動作検査

ここでは宇宙線荷電粒子を用いた FI 動作検査について述べる。検査システム及び検査方法について述べ、その後結果を述べる。

検査システムの構築及び検査方法

セットアップ図を載せる。検査システムは基本的にはテストパルスでのセットアップと同じであ

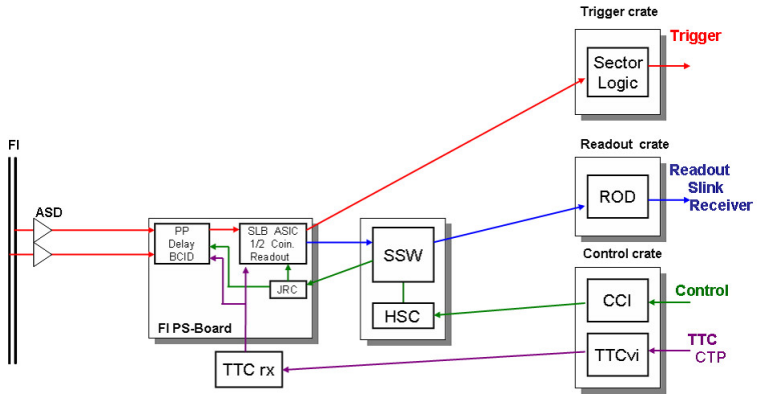


図 4.9: 宇宙線検査のセットアップ概要

るが、セルフトリガーとして SL を用いている。またチェンバーにかけたハイボルテージは 2700V、閾値電圧は 140mV で行った。宇宙線を用いて検査可能な項目は次の点である。

- 正常にチェンバーからデータが読み出せるか
- 欠損チャンネルの洗い出し
  2 層ともハイボルテージをかけて、その 2 層ともにヒットがあった場合データを読み出す。その結果ヒットが見られなかったチェンネルを欠損チャンネルとする。
- ケーブルがスワップしていないか
  チェンバーのハイボルテージを 2 層のうち一方にだけかけ、かけたほうの層のみヒットが見えるか確認する。

FI は片サイドで wire,strip ともに 768 チャンネルある。つまり片サイドで 1536 チャンネル、全部で 3072 チャンネルである。

結果

全チャンネルに対して行った。

スワップテストにおいてスワップが確認された箇所については、ケーブルを配線しなおし、解決した。

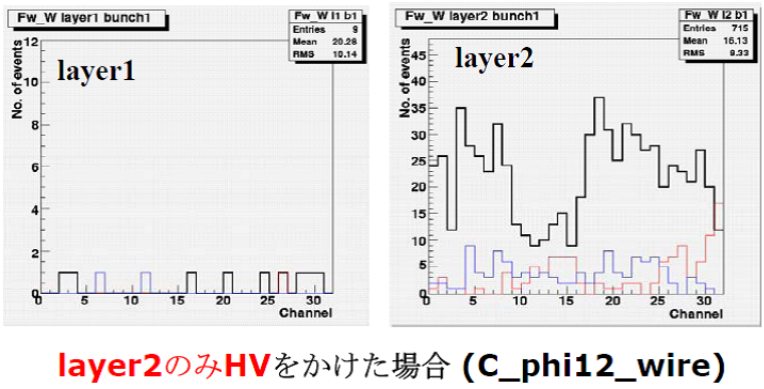


図 4.10: スワップテスト

図中のレイヤーとは FI チェンバーの 2 層のどちらかを指す。縦軸がヒット数、横軸がチャンネル数である。

また両サイドにおいて 2 チャンネルずつ発振する箇所を発見した。これはマスクで対処する予定である。

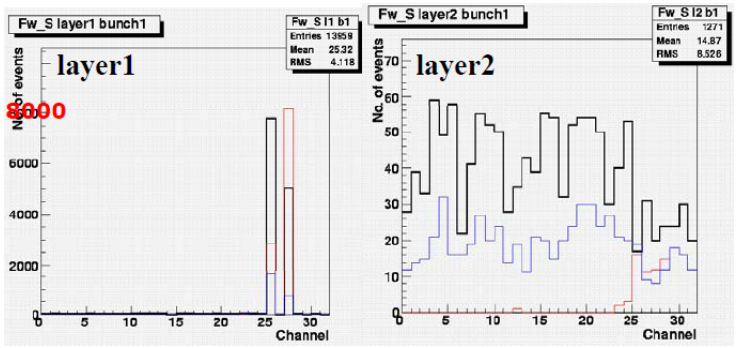


図 4.11: チャンネルの発振

縦軸がヒット数、横軸がチャンネル数である。

ASDTP において 16 チャンネルのまとまった欠けが確認された箇所は、宇宙線検査では欠けが確認されなかった。(図 4.12)

これは、ASD において TP を出力する経路が故障していると考えられる。そのため図の左 2 チャンネルについては欠損箇所であるが、他のチャンネルについては実際の実験の際に問題にはならないと考えられる。

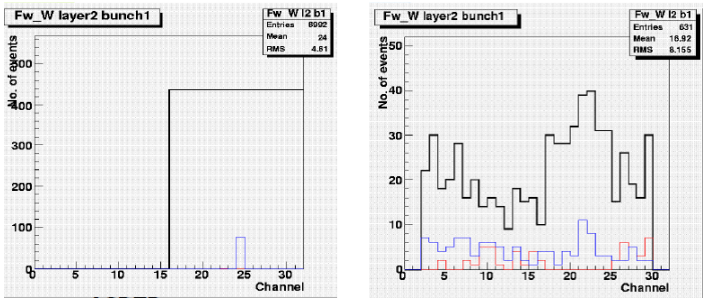


図 4.12: (左) ASDTP、 (右) 宇宙線

同チェンバーに対する ASDTP と宇宙線検査結果。縦軸がヒット数、横軸がチャンネル数である。宇宙線の左側 2 チャンネルは欠損チャンネル。右側の 2 チャンネルはこのチェンバーに関しては存在していないチャンネルであるため欠損ではない。

以上の検査により、FI チェンバー及び PS ボードの動作が検査され、また FI チェンバーと PS ボード間の接続も解決できた。その結果 FI での最終的な欠陥箇所を 7 箇所にまで減らすことができた。FI の全チャンネル数が 3072 チャンネルであることを考えると、欠損箇所は検査を通して 0.2%まで抑えることができたと言える。ここでは EI/FI についての動作試験について述べたが、ビッグウィールについても同様に検査が行われていることを付け加えておく。

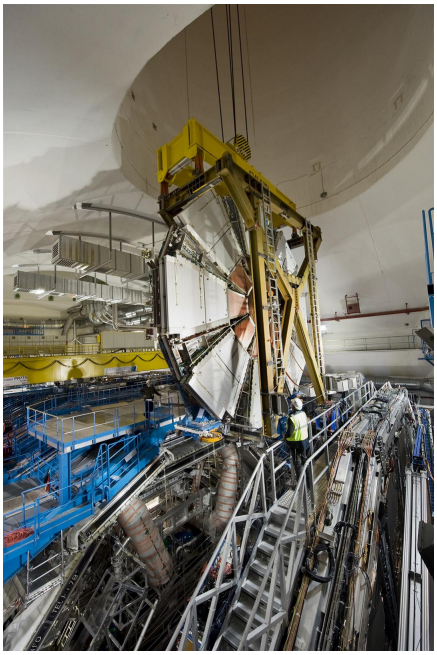


図 4.13: 地上検査が終了し Point1 へインストールされるスモールウィール

地上動作検査の後、FI を含む TGC は地下へインストールされ、そこで再び同様の検査を行い、そして最終的に無事 ATLAS に組み込まれた。

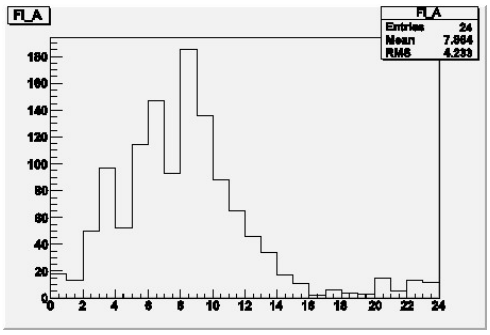


図 4.14: FI が初めて ATLAS に組み込まれた後、実際に得られた初期宇宙線データ
縦軸がヒット数、横軸がチェンバーの場所である。

現行の LHC に対して TGC 各種エレクトロニクスは、個別の動作検証そして ATLAS 全体での統合動作試験を経て現在高い水準で動作している。しかしながら対応できるデータ処理能力の限界はエレクトロニクスそれぞれで異なり、TGC エレクトロニクスの中で最初に限界が懸念されているのは ROD である。LHC の本格稼動を目前に控えそして中期計画として SuperLHC が視野に入ってきた現在、日本グループは現状 ROD に対する対応策を早期に講じておく必要性を感じ、新 ROD 開発プロジェクトを本年度より開始した。次章ではそのような背景のもとで行われた、ROD アップグレード初期開発の研究成果について述べることにする。

第5章　読み出しエレクトロニクス SuROD(Super ROD)の開発研究

この章では SuperLHC を視野に入れて開始したアップグレード ROD の開発状況について述べる。ここでは現行の ROD と区別するため、アップグレード ROD のことを SuROD として扱う。まず開発の方向性について述べ、それに基づいた開発状況及び結果について述べる。最後に考察を行う。

5.1　開発研究の方向性

ここでは次の 3 点を述べる。

1. エレクトロニクス設計手法
2. SuROD に要求されるもの
3. SuROD 開発方針

5.1.1　エレクトロニクス設計手法

エレクトロニクス設計に用いるハードウェア記述言語について、LSI である ASIC 及び FPGA について、そしてエレクトロニクス開発の流れについて述べる。

ハードウェア記述言語

ハードウェア記述言語 (Hardware Description Language:HDL) とは大規模な論理回路の開発の際に用いる言語である。論理回路が大規模になるにつれ、回路図による論理回路設計に限界が見えてきた。そこで回路図に代わる効率的な設計手法としてこの HDL が用いられるようになった。HDL での設計による特徴は主に次の点がある。

- デバイスを決定することなく回路設計ができること
- シミュレーションを用いた検証が詳細に行えること
  具体的には次の点である。
  - デバイスを設定した検証ができること
  - 設計の途中段階で検証できること
    つまり細部に渡る設計をしてからではなくても、骨格となるような設計部分のみで検証可能であり、その時点で設計にフィードバックさせることができる。検証対象が絞れるため開発効率が良くなる。

  - 検証用の記述も同じ文法で可能であり、きめ細かい記述ができること

以上の点から、HDL を用いることでモジュールを試作せずに精度の良い動作検証ができる。またデバイスを選択したシミュレーションによって最適なデバイスを決定することができ、試作するコスト等を抑えることができる。さらには、もし開発途中でデバイス変更の必要が生じても比較的容易に対応できる。このように HDL は開発効率が良い他、開発に対する柔軟性が高い点でも魅力的である。

HDL の代表的なものには、Verilog-HDL 及び VHDL の 2 種類がある。本開発では Verilog-HDL を用いている。Verilog-HDL の特徴には次のようなものがある。

- 記述が VHDL に比べ簡潔。
- C 言語をベースにした文法体系であり、構文や演算子が C 言語とほぼ同じなので (ソフトウェア) プログラム経験者にとってなじみ易い。なお、C 言語で関数にあたるものを Verilog-HDL ではモジュール (module) と呼ぶ。
- シミュレーション用言語として誕生した経緯から、VHDL に比べシミュレーションの記述力が充実している。

HDL 設計によって利用できるデバイスとして FPGA と呼ぶ LSI がある。次ではそれについて述べる。

ASIC と FPGA

ここでは良く FPGA を比較される ASIC について触れながら、FPGA の特徴を述べることにする。

ASIC とは Application Specific Integrated Circuit(特定用途向け集積回路) の略である。FPGA とは Field Programmable Gate Array(現場でプログラム可能なゲートアレイ) の略であり、つまり設計した論理回路を焼き付けることですぐ LSI として利用できるデバイスのことである。それぞれの特徴及び長所・短所を簡単に述べておく。

- ASIC
  - (特徴) チップ単価が安く大量生産に向いている。
  - (長所) FPGA に比べ高速、放射線に強い。
  - (短所) 開発コスト (金銭面、マンパワー面など) が大、設計後から製作までの時間がかかる。
- FPGA
  - (特徴) 回路構造をプログラムできる。少量生産向き。
  - (長所) 開発コストが小、設計後すぐ利用できる。
  - (短所) ASIC に比べ低速、ASIC に比べ放射線に弱い。

最終的な実用化でどちらを採用するかは上の特徴等を考慮した上で、目的に適したほうを選ぶべきである。しかしながら、論理設計を効率よく実機動作として反映させることのできる FPGA は ASIC 製作のプロトタイプとしても用いられる。

次では HDL を用いたエレクトロニクス設計の流れについて述べる。

開発の流れ

開発の流れは以下のようになる。

1. 対象システムに要求される条件の把握
2. 条件に基づいた上で仕様 (開発方針) の決定
3. HDL による論理設計
4. ビヘイビアシミュレーション
5. 論理合成
6. 配置配線
7. タイミングシミュレーション
8. 製作 (ASIC) またはダウンロード (FPGA)
9. 実機検証
10. 実用化

エレクトロニクス設計を行う際にまずすべきことは、対象システムに要求される条件を把握することである。その条件には、実現すべき機能から課される条件と物理的要因や外部要因から制限される条件がある。開発の初めにこれらをまず明確にする。

その次に行うことは、要求条件を考慮した上で対象回路の仕様を考えることである。この時点で全てを確定できるわけでないが大局的な枠組みを決定することは重要である。

仕様が確定した段階で実際に設計を行う。そして HDL を用いて設計した論理回路の評価をビヘイビアシミュレーションを用いて行う。その結果を踏まえ仕様及び設計の改良を繰り返す。

続いて論理合成を行う。論理合成とは HDL での記述をゲート回路の記述に変換することでありHDL 記述をもとに最適化する。さらに続いてテクノロジ・マッピングという、指定した FPGA デバイスのテクノロジ条件に合致する素子を割り当てる処理を行う。これらは自動的に行われる。

続いて配置配線を行い、タイミングシミュレーションによって検証を行う。その結果を考慮してさらに仕様及び設計を改良する。なお配置配線とは実際に素子を配置しそれらを配線することである。以上からもわかるようにタイミングシミュレーションとは論理合成及び配置配線を経てから行うもので、それゆえ FPGA のデバイスの性能に依存するとともに配線・配線状況による動作速度への影響も考慮されたシミュレーションであり実際の動作をかなり正確に表現するものである。この配置配線処理は多少のパラメータを設定する以外は自動的に行われる。

シミュレーションで期待通りの動作が確認できたら、実機で動作検証を行う。ASIC の場合は、数ヶ月の製作期間を経て出来上がる。一方で FPGA の場合は直接論理を書き込んで使用できるようになる。この検証を経て実用段階になる。

本論ではタイミングシュミレーションまでを行った。以上の流れに沿って、まず SuROD を開発するにあたって ATLAS から要求される条件、またそれを考慮した設計方針を述べる。

5.1.2 SuROD に要求されるもの

ここでは SuROD の開発方針を決定するために、SuROD に要求される機能及び動作速度を考えてみる。まずは現行 ROD と置き換えることを想定して、SuROD も現行と同じ入出力として考えることにすると 10 個の SSW の入力を受け取り、それを 1 本化にして出力するという構成になる。ここでは SuROD の手前にある SSW 出力データフォーマットそして SuROD の後にある ROS から要求されるデータフォーマットについて考え、続いて達成すべき動作速度を見積もることにする。

SSW のデータ出力フォーマット

フォーマットの詳細は 3 章にある。SuROD はこれらのフォーマットデータを複数の SSW から受け取る。SSW の出力には G-Link が用いられており、そのため SuROD は G-Link を用いて 1 クロックあたり 16 ビットのデータをパラレルに受信する必要がある。

ROS からの要求

フォーマットの詳細は 3 章にある。フォーマットの細かい点は将来変更される可能性があるが、おおよそこのフォーマットに従うと考えられる。現時点では SuROD もこのフォーマットに従ってデータを出力するものとする。また出力には S-Link を用いて 32 ビットのパラレルデータとしなければならない。

フォーマットに従って正しい出力できたとしても、ATLAS のデータ処理に制限を付けてしまうような動作速度は望ましくない。次では SuROD が達成すべき動作速度について考えてみる。

要求される動作速度

動作する速度は速ければ速いほうが望ましいが、コストや開発時間などを考慮すると必要以上の速度を求めるのは有益とは限らない。達成すべき動作速度を決定する方法として、送るべきデータサイズを見積もった上で、ここでは外部から与えられる制限を考えて見積もることにする。考えられる制限は 2 つあって、入力側の G-Link 及びレベル 1 トリガーからの制限、そして出力側の S-Link からの制限である。それぞれ述べる。

- ROB へ出力しなければならないデータサイズの見積もり
  3 章で述べたが、SSW は 1 セクターあたり最大 10 個使用される。1 つの SSW の 1event あたりの潜在的な最大ヒットデータサイズはおおよそ 2000byte 程度であるが、予想されるヒット情報は総チャンネル数の 0.02 パーセント程度である。SuperLHC でルミノシティが 10 倍とし、ヒット情報も比例して 10 倍の 0.2 パーセント程度になるとして粗く見積もると、一つの SSW のデータサイズは平均 170byte 程度だと考えられる。つまりセクター単位では最大で 1700byte 程度となり、100KHz の L1 トリガーレートを仮定すると 170M[bytes/sec] が SuROD の送るべきデータ量として見積もられる。

これを踏まえてデータ転送の上限を見積もってみる。

- G-Link 及びレベル 1 トリガーからの制限
  レベル 1 トリガーのレートは最大 75KHz(オプションで 100kHz) である。また SSW 上の G-Link は 40MHz のクロックで動作しており、1 クロックあたりの出力データサイズは 2Byte である。つまり 1 つの SSW の送信能力は最大で 80M[bytes/sec] であり、セクター単位では 800M[bytes/sec] であると見積もられる。

- S-Link からの制限
  現在 ATLAS で使用されている S-Link の出力帯域は 1Gbps、最新のものは 2Gbps である。これは換算するとそれぞれ 125M[bytes/sec]、250M[bytes/sec] の送信能力があるということであり、SLHC でのデータ量を送信するためには 2Gbps を使用する必要があると言える。

以上からデータ処理速度を制限するのは出力側の S-Link であることがわかる。このことから論理回路がデータ処理においてボトルネックにならないためには S-Link の動作速度を上回る必要がある。SuROD は 2bytes の入力を受け 4bytes で出力することから、内部ロジックでは 125MHz 以上で処理し、そして 62.5MHz 以上で出力する必要があると言える。
次ではここまでの条件を考慮した上での開発の方針を述べる。

5.1.3 SuROD 開発方針

ここでは使用する初期開発での使用デバイスについて述べ、SuROD の担う役割、そして HDL の設計方針を述べる。

使用デバイス (ASIC と FPGA)

初期開発では、まず FPGA を利用して開発を行う。最終的に利用するデバイスには ASIC も考えられるが、前述のように実機として製作するまでに時間を要するために、初期開発段階においては FPGA を用いて実機動作検証までを行うほうが、特に時間的側面において有益だからである。
なお SuROD は最終的に FPGA を使用する可能性が高いことも付け加えておく。それは以下の理由による。

- 必要とするチップ総数はどんなに多く見積もっても数百個であるため、ASIC よりもコスト面で有利であること

- 検出器のある空洞とは隔てられたカウンティング・ルームに置かれるため、放射線耐性をほとんど必要としないこと

FPGA と一口に言っても性能や大きさなどにより、様々な種類がある。これは回路設計後のシミュレーション結果の後に決定することができる。

続いて SuROD モジュールの構想を述べる。

SuROD モジュール構想

改めて整理すると、SuROD は次の役割を担う。

- 複数の SSW からの 16bit データ列を G-Link によって受け取る
- 受け取った SSW データをまとめる
- ROS が要求する情報を付加する
- S-Link を通して 32bit データ列にして出力する
- 動作速度は 62.5MHz 以上、最新の S-Link を使用する場合は 125MHz 以上が必要である。

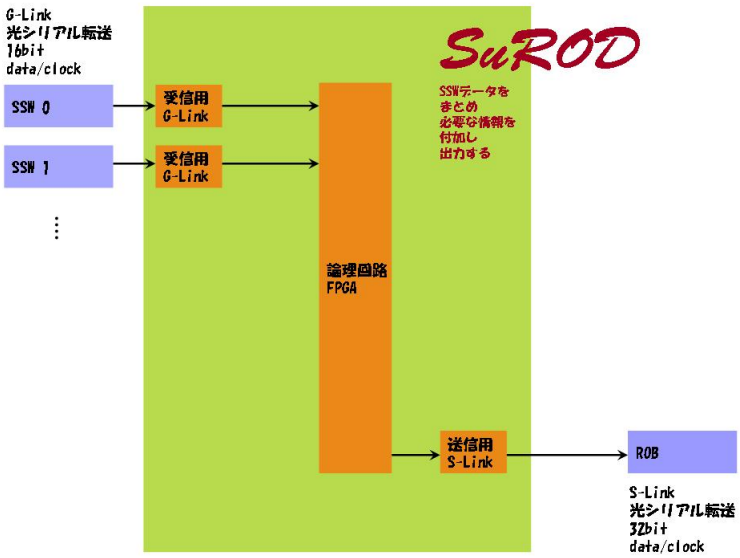


図 5.1: SuROD でのデータの流れ

16 ビットの SSW データを複数受け、それを一本化して ROB に 32 ビットデータで出力する。

次にこれらの役割を持つ HDL による論理設計方針を述べる

HDL 構成

論理回路は大きくデータ受信用 (Rx と呼ぶ) とデータ送信用 (Tx と呼ぶ) の 2 つに分けて考えることにする。

- Rx 論理
  前述のように、Verilog-HDL の記述は C 言語での関数に相当するモジュールによって構成される。モジュール構成を示す。

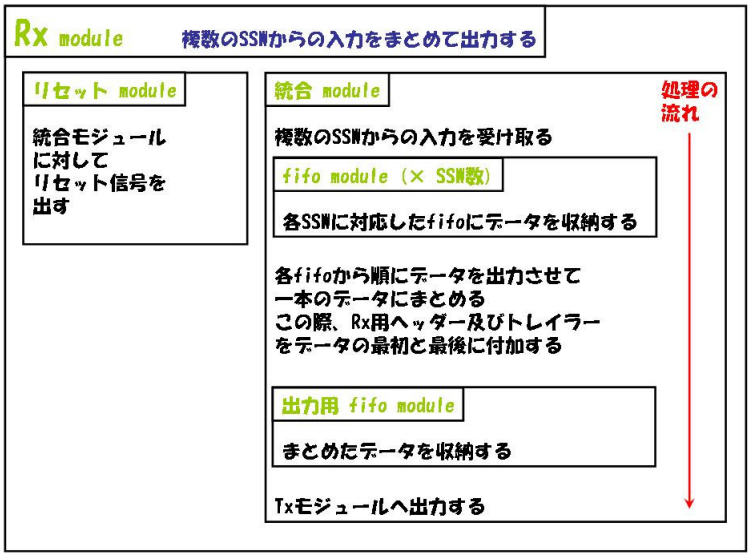


図 5.2: Rx 論理回路の構成

- Tx 論理
  モジュール構成を示す。

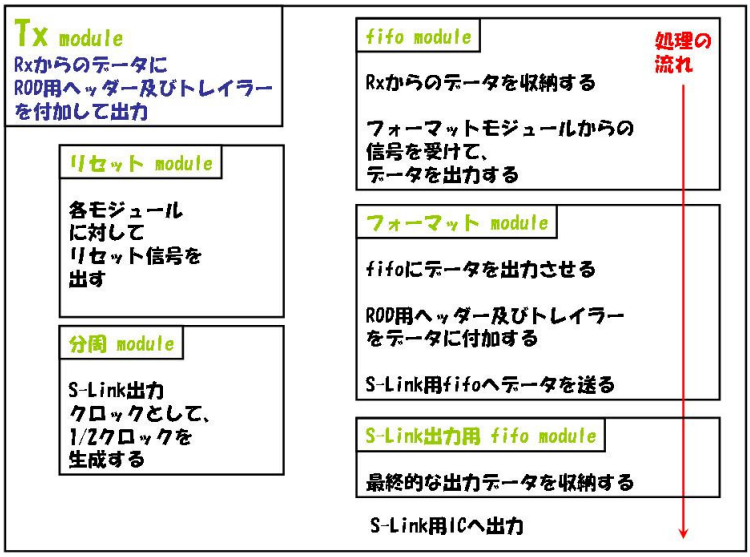


図 5.3: Tx 論理回路の構成

以上のように Rx と Tx で処理機能が閉じているので別個として設計することが可能である。

次節では設計に対して行ったシミュレーションでの検証について述べる。

5.2 検証

この節では仕様に基づいて設計した HDL に対しての検証について述べる。検証方法について述べた後に検証結果を述べる。

5.2.1 検証方法

まず設計したものが、論理的に仕様に沿っているかをまず検証する必要がある。そこでビヘイビアシミュレーションによって評価してみる。シミュレータには ISE Simulator を使用した。シミュレーション環境を載せる。また入力データと出力データの中身を示したものを載せる。なお Rx ロジックの入力にあたる SSW のデータは、現行の ROD で得た宇宙線データを SSW の出力データにまで戻すプログラムを作成し生成した。

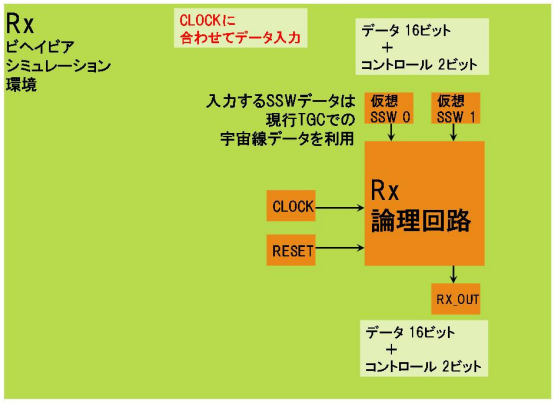


図 5.4: Rx 用ビヘイビアシミュレーション環境

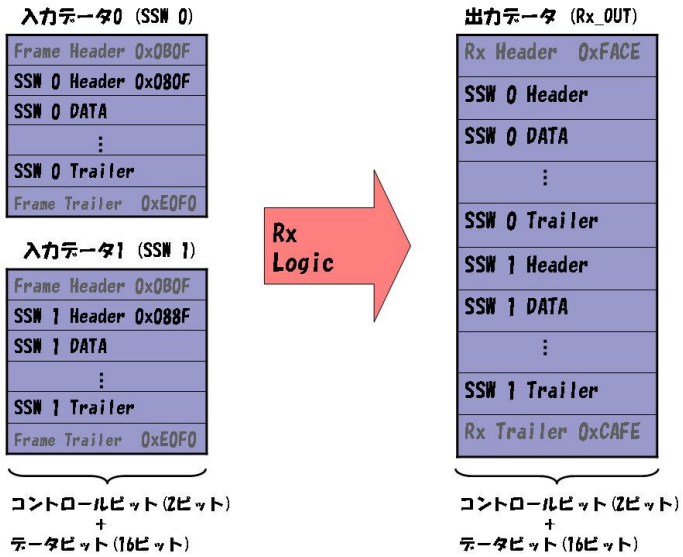


図 5.5: SSW データから Rx 出力データ

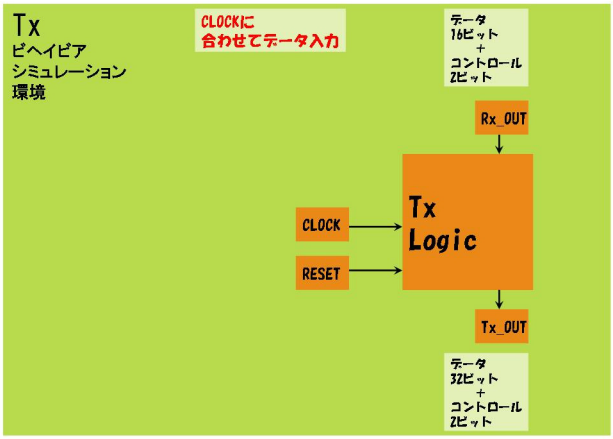


図 5.6: Tx 用ビヘイビアシミュレーション環境

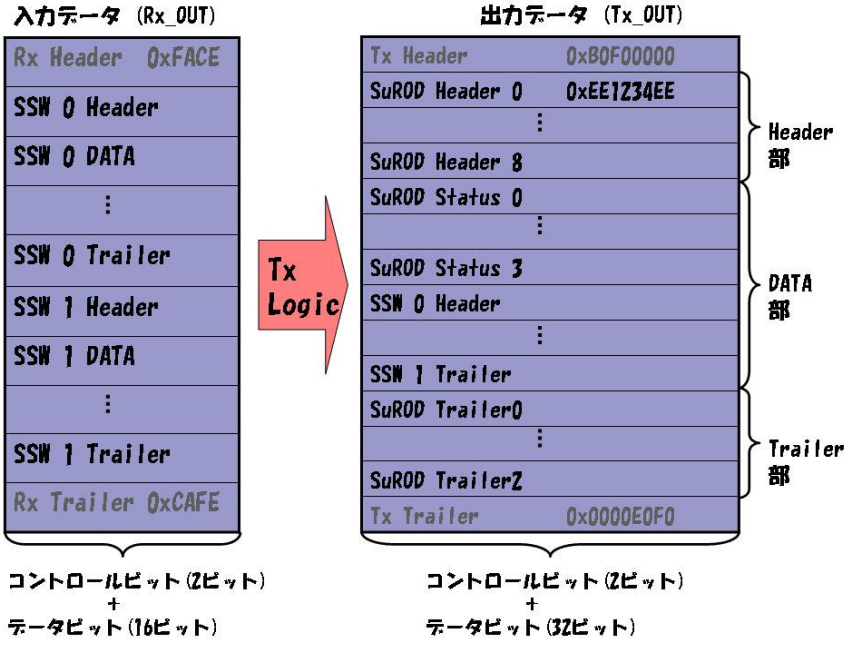


図 5.7: Rx データから Tx 出力データ

ソースコードを付録として載せる。

ビヘイビアシミュレーションによって検証した後にタイミングシミュレーションで検証を行う。タイミングシミュレーションによって論理回路の実際の動作速度を見積もることが可能である。さらに FPGA の種類を変更しながらシミュレーションを行うことで、デバイスによる速度変化を検証することも可能である。

テストベンチはビヘイビアシミュレーションで用いたものを使用した。そしてクロック周波数を変化させていき、出力が確認できた最大のクロック周波数を限界動作速度とした。配置配線は自動配線を用いて行い、タイミング制約はクロック周波数のみを課した。

デバイスには Xilinx 社の FPGA 群を用いた。その中でタイミングシミュレーションを行ったデバイスは次のものである。

デバイス ファミリ名	スピード	デバイス名	スライス数 (※1)	最大ユーザI/O
Spartan3	-4, -5	XC3S400	3584	264
Spartan3A	-4, -5	XC3SZ00A	1792	112
	-4, -5	XC3S400A	3584	142
	-4, -5	XC3S700A	5888	165
Virtex4	-10, -11, -12	XC4VLX15	6144	320
Virtex5	-1, -2, -3	XC5VLX30	4800 (※2)	400

※1 ロジック、四則演算、およびROM機能を実現させる単位。
フリップフロップ、LUT (Look Up Table)などで構成される。
なお、1CLB = 4スライスである。

※2 Virtex5のスライスはLUTとフリップフロップが各4個含まれている。
(他のデバイスファミリは各2個)

図 5.8: タイミングシミュレーションに用いたデバイス

Spartan シリーズとは量産向けの低コストな FPGA シリーズである。それに対して Virtex シリーズは高性能の FPGA シリーズである。それぞれのシリーズの一般的なものを選択した。スピードは絶対値として大きいものほど高速である。ただしファミリ間での単純比較はできない。

以上のデバイスに対して、タイミングシミュレーションで次の点を検証してみる。

- Rx 及び Tx 論理に対して

  – デバイスごとの限界動作速度の見積もり
  設計した論理の動作速度をそれぞれのデバイスについて検証してみる。

  – 同ファミリ内でのデバイスサイズを変えたときの動作速度変化

- Rx 論理に対して

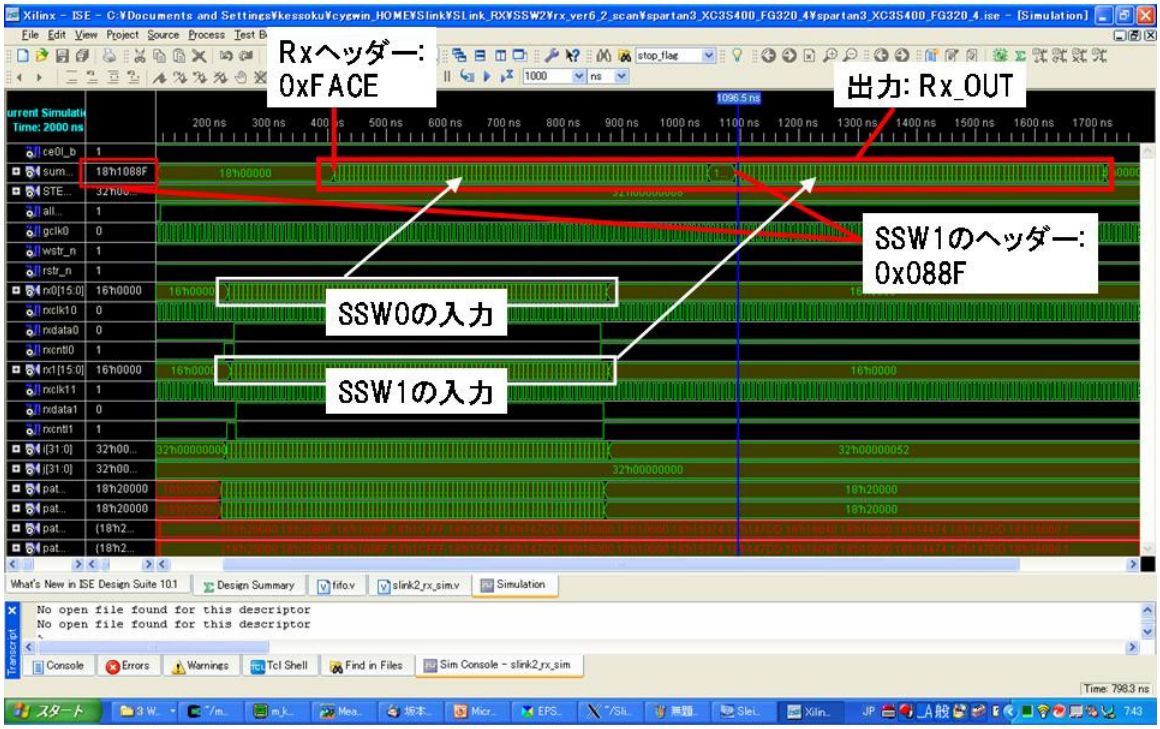
  – 入力数を変化させていったときの速度及びスライス使用率の変化
  入力数が変化するにつれて動作速度及びスライス使用率がどう変化するかを検証してみる。なお Tx 論理の入力数は 1 つの Rx 出力データのみと仮定して行っているため入力数の変化は検証しない。

5.2.2 検証結果

ビヘイビアシミュレーション

Rx と Tx のビヘイビアシミュレーションの結果である。論理として設計通りに動作していることが確認できる。

Rx ビヘイビアシミュレーション結果



125MHz と仮定した Rx ビヘイビアシミュレーション結果。SSW データを入力している。232 ナノ秒で SSW0 の入力が開始、236 ナノ秒で SSW1 の入力が開始している。その後 873 ナノ秒で SSW0 の入力が終了し、877 ナノ秒で SSW1 の入力が終了する。
出力は 413 ナノ秒で開始する。その後 1045 ナノ秒で SSW0 までの出力が完了して 1091 ナノ秒から SSW1 の出力が始まる。そして 1716 ナノ秒で Rx トレイラーによって出力が終了している。

図 5.9: Rx ビヘイビアシミュレーション結果

Tx ビヘイビアシミュレーション結果

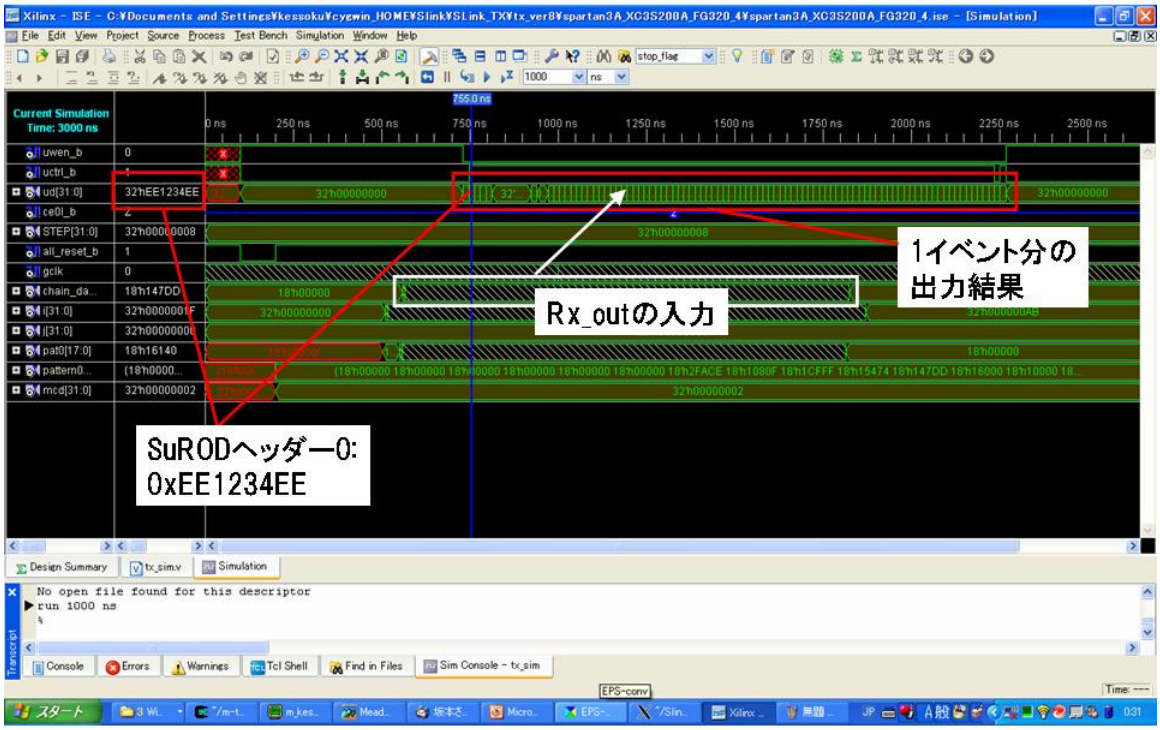


図 5.10: Tx ビヘイビアシミュレーション結果

125MHz と仮定した Tx ビヘイビアシミュレーション結果。Rx の出力結果を入力している。559 ナノ秒に Rx データ入力が始まり 732 ナノ秒から出力が始まっている。また 1824 ナノ秒で Rx データ入力が終了し 2268 ナノ秒で出力が終了している。ヘッダーなどがきちんと付加されて出力されているのがわかる。

タイミングシミュレーション

Rx と Tx のシミュレーション結果をそれぞれ述べる。

- Rx FPGA
  各デバイスに対する、SSW からの入力が 2 つの場合の Rx ロジックの大きさを示す。他の入力数については後で述べる。どのデバイスに対しても比較的余裕があることがわかる。

Rx

デバイス ファミリ名	デバイス名	A 総スライス数	B 使用スライス数	B/A [%]
Spartan3	XC3S400	3584	567	15%
Spartan3A	XC3S200A	1792	567	31%
	XC3S400A	3584	567	15%
	XC3S700A	5888	567	9%
Virtex4	XC4VLX15	6144	572	9%
Virtex5	XC5VLX30	4800	315	6%

図 5.11: デバイスごとのスライス使用率 (SSW からの入力が 2 つの場合)

  - デバイスによる限界動作速度の変化
    デバイスごとの限界動作速度を図 5.12 に示す。Virtex4 が最も高い周波数で動作する。他のデバイスファミリでも 125MHz 以上で十分動作している。

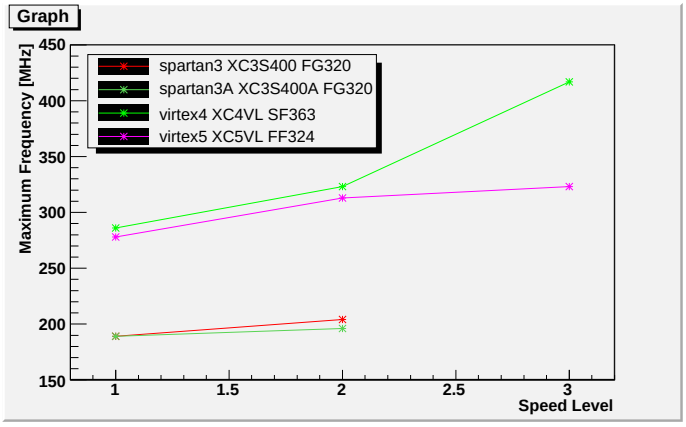


図 5.12: Rx ロジックにおけるデバイスの種類と限界動作速度
各デバイスファミリで最もスピードレベルが小さいものを図中横軸の 1 にしている。
(例えば Spartan シリーズは speed -4 を図中横軸の 1、-5 を 2 としている。)
つまり右へ行くほど同デバイスの中で性能が高い。

– 入力する SSW 数の変化に対する動作速度と使用スライス数の変化
SSW の入力数を 3 つまで増やして検証してみた。図 5.13 に入力数と動作速度を示す。SSW の入力数が 3 つまでは入力数の変化による動作速度の低下はほとんど見られない。これはデバイスに対してロジックの大きさが比較的小さいためと思われる。

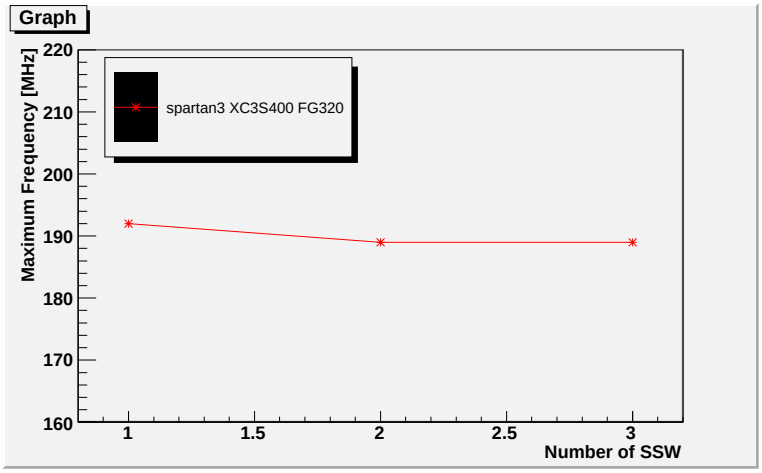


図 5.13: 入力数変化に対する限界動作速度

入力数に対するスライス数の変化を図 5.14 に示す。SSW からの入力が一つ増えるとスライス数にして 200 個程度増えることがわかる。

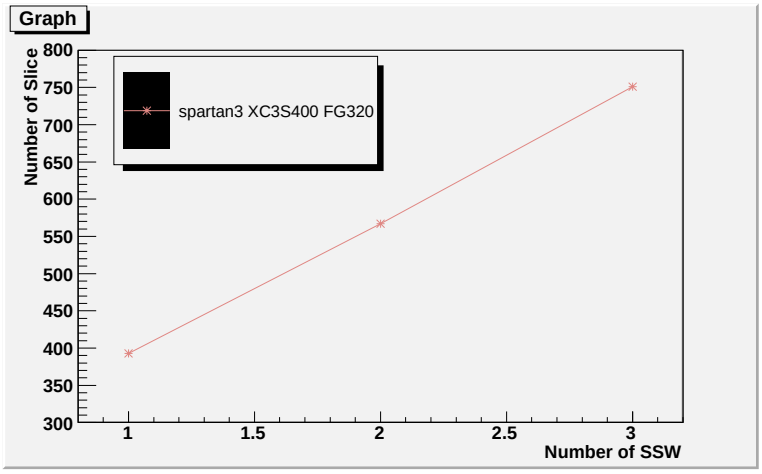


図 5.14: 入力する SSW 数と使用スライス数

– 同じファミリ内でのデバイスサイズの違いによる変化
デバイスサイズの違いによる動作速度の変化を図 5.15 に示す。

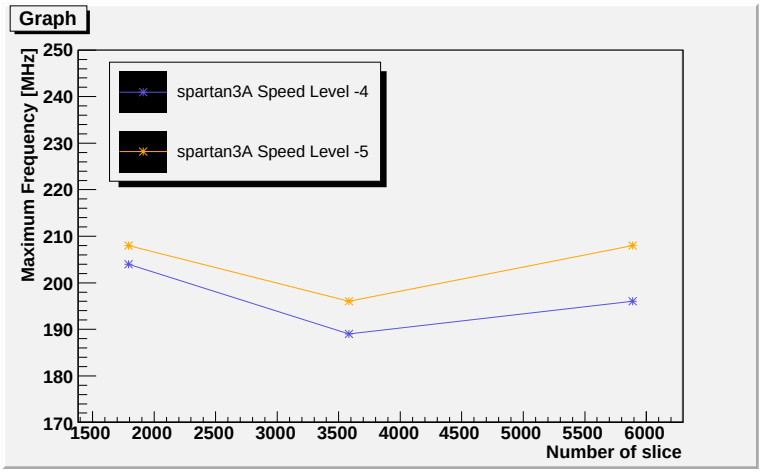


図 5.15: デバイスサイズによる最大動作速度

- Tx FPGA
  各デバイスでの Tx ロジックの大きさを示す。どのデバイスも Rx ロジック同様、比較的余裕がある。

Tx

デバイス ファミリ名	デバイス名	A 総スライス数	B 使用スライス数	B/A [%]
Spartan3	XC3S400	3584	608	16%
Spartan3A	XC3S200A	1792	612	34%
	XC3S400A	3584	612	17%
	XC3S700A	5888	612	10%
Virtex4	XC4VLX15	6144	629	10%

図 5.16: デバイスごとのスライス使用率

  - デバイスによる限界動作速度の変化
    各デバイスでの限界動作速度を図 5.17 に示す。Spartan3 シリーズでも十分な速度で動作していることがわかる。

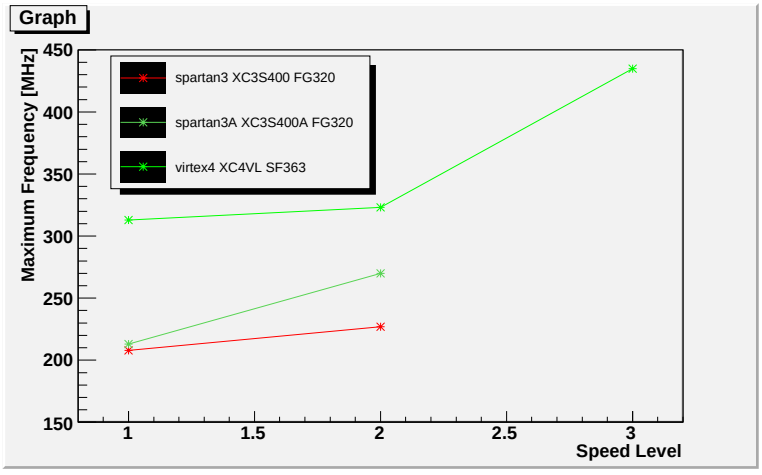


図 5.17: Tx ロジックにおけるデバイスの種類と限界動作速度

– 同じファミリ内でのデバイスサイズの違いによる変化
デバイスサイズの違いによる動作速度の変化を図 5.18 に示す。

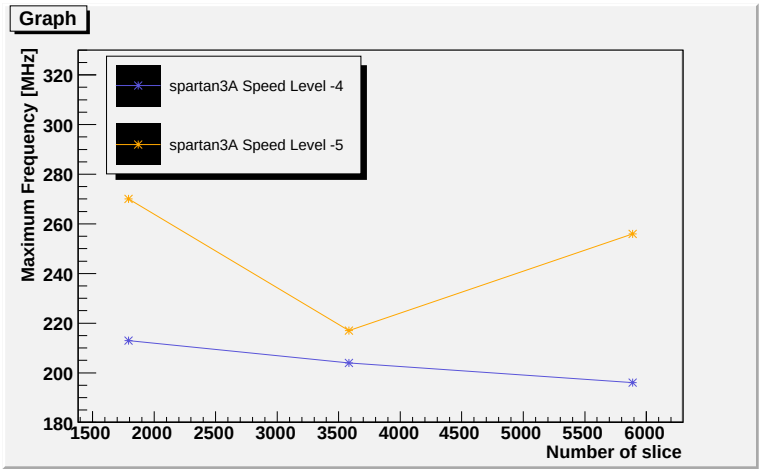


図 5.18: デバイスサイズによる最大動作速度

これらのタイミングシミュレーションは自動配置配線によって行ったものである。配線を行った後に動作の遅い部分の再配線を行ったり、ロジック自体を改良させることで、動作速度は向上する。現時点のようにデバイスがどれほどの速度で動作させることができるのかを把握したい場合、このような微調整を施すことのなく行った動作速度は最大動作速度の下限値の目安となる。以上の検証によると、安価な Spartan シリーズでも要求される水準、125MHz 以上で動作すると言える。これを踏まえてこれからの開発プランについて述べることにする。

5.3 これからの開発プラン

ここでは検証結果を踏まえて、これからの SuROD 開発のプランを考察してみる。

SuROD 上の FPGA 構成

論理の骨格ができた時点で次に考えるべき点は論理処理をどのように分散させるかである。構成を図 5.19 に示す。

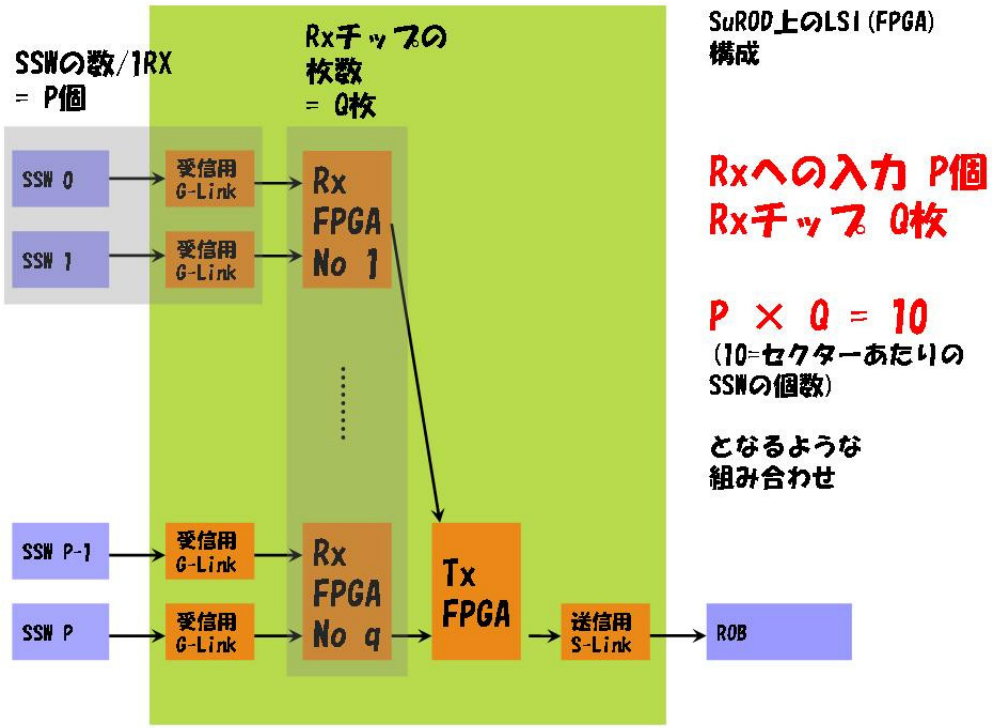


図 5.19: SuROD モジュールのデータ処理 LSI の構成

1 セクター分のデータを SuROD1 枚で処理する場合、最大 10 枚の SSW を受付け、それを出力しなければならない。1 枚の Rx チップではロジックサイズの制限や動作速度の制限などでそれら全てを処理できない可能性が考えられる。その場合は Rx チップを複数枚使用することを考えなくてはならない。その組み合わせは 1 枚の Rx が受ける SSW の数と Rx チップの枚数の積が 10 となるような組み合わせがある。今後 Rx についてさらに入力数を増やしたタイミングシミュレーションを行い、それに伴って Tx の入力数を変えたタイミングシミュレーションを行っていく必要がある。

今後の予定

その他、SuROD 開発を行っていくにあたって今後すべきことは次のようなものがある。

- シミュレーションによる検証
  SSW 入力数などをさらに変化させてみるなどしてさらなる検証を行う。

- より具体的な設計
  エラー検知アルゴリズムなどの機能をつけるなど、より具体的な設計を行っていく。
- 実際の ATLAS 衝突実験データを用いた性能の検証
- 汎用モジュールを用いた実機動作検証
  実際のデバイスでどれほどの速度で動作するかを検証していく。

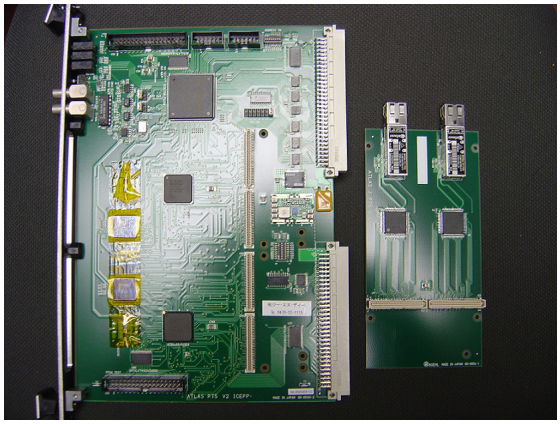


図 5.20: 汎用モジュール (Spartan3 搭載)

来年度までにデザインを完了しプロトタイプの製作に着手する。2010 年度に ATLAS に仮に組み込み、動作確認及び動作検証を行う。そして 2011 年度には LHC アップグレードに関する Technical Design Report に寄与する予定である。

結論

本論において TGC 読み出しシステムの検証は EI/FI を中心であったが、BW においても同等の綿密な検査が行われてきた。その結果 TGC システムは動作が保証され、これまでの統合動作試験において概ね正常に稼動していることが確証されている。また問題箇所についても把握が進み、その対処についても日々進展している。

SuROD については初期の重要なステップと言える HDL デザインの骨格が完成し、シミュレーションの結果において要求水準を十分に満たす設計であることが確認された。そしてこの先、本論の結果に基づいた上で詳細な設計を行い、より信頼性のあるミューオン検出器読み出しシステムを構築していく。

以上の成果はこの先の ATLAS 実験に貢献していくだろう。

謝辞

この二年間は多くの人に出会いそして刺激を受けた日々でした。指導教官である坂本宏教授には、研究内容だけに留まらず研究生活についても多くの助言を頂きました。ありがとうございます。

またCERNでの生活では本当に多くのスタッフの方々に支えてもらいました。佐々木修氏、池野正弘氏、岩崎博行氏、近藤敬比古氏、田中秀治氏、蔵重久弥氏、松下崇氏、越智敦彦氏、石川明正氏、福永力氏、竹下徹氏、菅谷頼仁氏、戸本誠氏、杉本拓也氏、石野雅也氏、川本辰男氏、小林富雄氏、織田勧氏、皆様どうもありがとうございました。

先輩方々にも大変お世話になりました。奥村恭幸氏、高橋悠太氏、久保田隆至氏、野本浩史氏、門坂拓哉氏、丹羽正氏、どうもありがとうございました。

そして多くの刺激をもらった同期、後輩の皆様。長谷川慧氏、鈴木友氏、平山翔氏、金賀史彦氏、中塚洋輝氏、早川俊氏、越前谷陽佑氏、東裕也氏、鈴木拓也氏、どうもありがとう。

研究生活をサポートしてくださった秘書の皆様にも感謝いたします。本当にありがとうございました。

参考文献

[1] The ATLAS Experiment at the CERN Large Hadron Collider, 2008 JINST 3 S08003

[2] ATLAS Level1 Trigger Technical Design Report, CERN/LHCC/97-22,1999

[3] Outline of R&D activities for ATLAS at an upgraded LHC, January 23rd 2005

[4] The raw event format in the ATLAS Trigger & DAQ, January 23rd 2008

[5] Agilent Technologies Agilent HDMP-1032A/1034A Transmitter/Receiver Chip Set Data Sheet

[6] S-Link: A Prototype of the ATLAS Read-out Link

[7] Endcap Muon trigger system: ROB input buffer format, August 5th 2006

[8] Hiroshi Nomoto, "Star Switch Spec " April 7th 2006

[9] 桑原隆志、東京大学大学院修士学位論文「ATLAS 前後方ミューオントリガーシステムの構築」, 2007 年 1 月

[10] 西田昌平、京都大学大学院修士論文 「ATLAS TGC エレクトロニクス読み出し系の開発」, 2000 年 2 月

[11] 永井義一、筑波大学大学院博士前期過程　数理物質科学研究科修士論文「ATLAS 実験 SCT バレルシリコン飛跡検出器の宇宙線を用いた性能評価」, 2007 年 2 月

[12] 小林 優 CQ 出版社 「入門 Verilog HDL 記述」

[13] Xilinx データシート

[14] 市川保正、松本潔、岩瀬英治 東京大学工学系研究科知能機械情報工学専攻　「FPGA を用いたデジタル回路設計演習」

[15] 平山翔、日本物理学会春季大会　口頭一般講演「ATLAS 前後方ミューオントリガー Small Whell の総合試験」,2008 年 3 月

[16] 佐々木修、日本物理学会春季大会　口頭一般講演「ATLAS 用エレクトロニクス/TDAQ の開発」,2008 年 3 月

[17] 浅井祥仁、神前純一、田中純一、LHC が切り拓く 21 世紀の素粒子物理

Appendix

HDL ソースコード

Rx ロジック

SSW の入力が 2 つの場合のコードを載せる。

```
//////////////////// slink2_rx.v ////////////////////////////

`include "slink2_rx_param.v"

module slink2_rx
(
    all_reset_b, gclk0, wstr_n, rstr_n,

    rx0, rxclk10, rxdata0, rxcntl0,
    rx1, rxclk11, rxdata1, rxcntl1,

    ce0l_b,

    sum_out
    chain_data
);

// signal

input  all_reset_b;        // global reset
input  gclk0;         // global clock
input  wstr_n; // write strobe
input  rstr_n; // read strobe

// from/to glink chip
input  [15:0] rx0, rx1;
input  rxclk10, rxclk11;
input  rxdata0, rxdata1;
```

```
input  rxcntl0, rxcntl1;

output ce0l_b;

output [17:0] sum_out;

//----------------------------------------

assign ce0l_b = 1'b1;

//------------------------------------------------
// write clk
wire wclk0, wclk1;

assign wclk0 = rxclk10;
assign wclk1 = rxclk11;

//------------------------------------------------------
// making reset signal

wire rst_r_n;

rst_mod #( `RESET_PULSE_WIDTH ) rst_mod
(
    .gclk0( gclk0 ),
    .all_reset_b( all_reset_b ),
    .rst_r_n( rst_r_n )
);

//------------------------------------------------------


//-----------------------------------------------------
// fifo signals

wire [`FIFO_DATA_WIDTH-1:0] fifo_in0, fifo_in1;

assign fifo_in0 = { rxcntl0, rxdata0, rx0 };
assign fifo_in1 = { rxcntl1, rxdata1, rx1 };


//-------------------------------------------------------
// fifo dump, sum
```

```
reg [17:0] raw_data0, raw_data1;

always @ ( posedge wclk0 ) begin
raw_data0 <= fifo_in0;
end

always @ ( posedge wclk1 ) begin
raw_data1 <= fifo_in1;
end

combine combine
(
.raw_data0( raw_data0 ), .raw_data1( raw_data1 ),
.clk( gclk0 ),
.rst_n( rst_r_n ),
.wclk0( wclk0 ), .wclk1( wclk1 ),

.sum_out( sum_out )
);

endmodule



////////////////// rst_mod.v ///////////////////////

module rst_mod
(
    gclk0,
    all_reset_b,
    rst_r_n
);

parameter RESET_PULSE_WIDTH = 2;

input  gclk0;
input  all_reset_b;
output rst_r_n;

sync_rst #( RESET_PULSE_WIDTH ) sync_rst_r
(
    .rst_src_n( all_reset_b ),
```

```
    .clk( gclk0 ),
    .rst_n( rst_r_n )
);

endmodule




////////////////// combine.v ////////////////////////////

module combine
(
raw_data0, raw_data1,
clk,
rst_n,
wclk0, wclk1,

sum_out
);

parameter SSW_FRAME_HEAD = 36'b10_0000_0000_0000_0000_10_0000_1011_0000_1111;//0x00000B0F
parameter SSW_FRAME_HEAD_B = 18'b10_0000_1011_0000_1111;
parameter SSW_FRAME_TRAIL = 36'b10_0000_1110_0000_1111_10_0000_0000_0000_0000;//0x0E0F0000
parameter SSW_FRAME_TRAIL_F = 18'b10_0000_1110_0000_1111;
parameter RX_FRAME_HEAD = 18'b10_1111_1010_1100_1110; //0xFACE
parameter RX_FRAME_TRAIL = 18'b10_1100_1010_1111_1110; //0xCAFE
parameter LASTSSW = 2'b01;


// signal
input [17:0] raw_data0, raw_data1;
input wclk0, wclk1;
input clk;
input rst_n;
output [17:0] sum_out;

wire [17:0] dout0, dout1;
wire empty0, empty1;
wire full0, full1;
```

```
reg [17:0] order_data0, order_data1;

always @ ( posedge wclk0 ) begin
order_data0 <= raw_data0;
end

always @ ( posedge wclk1 ) begin
order_data1 <= raw_data1;
end


////////////////////////////////////////////////


reg [71:0] shift_order0, shift_order1;

always @ ( posedge wclk0 ) begin
shift_order0 <= { shift_order0[53:0], order_data0 };
end

always @ ( posedge wclk1 ) begin
shift_order1 <= { shift_order1[53:0], order_data1 };
end

////////////////////////////////////////////////

reg write_flag0, write_flag1;

always @ ( posedge wclk0 or negedge rst_n ) begin
if ( !rst_n ) begin
write_flag0 <= 1'b0;
end
else if ( shift_order0[35:0] == SSW_FRAME_HEAD ) begin
write_flag0 <= 1'b1;
end
else if ( shift_order0[71:36] == SSW_FRAME_TRAIL ) begin
write_flag0 <= 0;
end
end

always @ ( posedge wclk1 or negedge rst_n ) begin
if ( !rst_n ) begin
write_flag1 <= 1'b0;
```

```
end
else if ( shift_order1[35:0] == SSW_FRAME_HEAD ) begin
write_flag1 <= 1'b1;
end
else if ( shift_order1[71:36] == SSW_FRAME_TRAIL ) begin
write_flag1 <= 0;
end
end

//////////////////////////////////////////////////////

reg [1:0] sellect;

dump_fifo dump0
(
.din( shift_order0[53:36] ),
.rd_clk( clk ),
.rd_en( !empty0 & ( sellect == 2'b00 ) ),
.rst( !rst_n ),
.wr_clk( wclk0 ),
.wr_en( write_flag0 & ( shift_order0[53:52] != 2'b00 ) ),
.dout( dout0 ),
.empty( empty0 ),
.full( full0 )
);

dump_fifo dump1
(
.din( shift_order1[53:36] ),
.rd_clk( clk ),
.rd_en( !empty1 & ( sellect == 2'b01 ) ),
.rst( !rst_n ),
.wr_clk( wclk1 ),
.wr_en( write_flag1 & ( shift_order1[53:52] != 2'b00 ) ),
.dout( dout1 ),
.empty( empty1 ),
.full( full1 )
);

reg [17:0] fifo_out0, fifo_out1;//, fifo_out2;

always @ ( posedge clk ) begin
fifo_out0 <= !empty0 ? dout0 : 18'b0;
```

```
fifo_out1 <= !empty1 ? dout1 : 18'b0;
end


//-----------------------------------------
// sum

reg [3:0] add_fr_h;

always @(posedge clk or negedge rst_n) begin
if ( !rst_n ) begin
add_fr_h <= 4'b0;
end
else begin
   add_fr_h <= { add_fr_h[2:0], (!empty0 & ( sellect == 2'b00 )) };
end
end

reg [17:0] sum_data;
  always @ ( posedge clk ) begin
    case ( sellect )
      2'b00:  sum_data <= fifo_out0;
      2'b01:  sum_data <= fifo_out1;
    endcase
  end

reg [125:0] shift_data;

always @ ( posedge clk or negedge rst_n ) begin
if ( !rst_n ) begin
sellect <= 2'b0;
end
else if ( add_fr_h == 4'b0111 ) begin
shift_data <= { shift_data[107:0], RX_FRAME_HEAD };
end
   else if ( sum_data == SSW_FRAME_TRAIL_F ) begin
   if ( sellect == LASTSSW )begin
sellect <= 2'b0;
   shift_data <= { shift_data[107:0], sum_data };
end
else begin
   sellect <= sellect + 1;
   shift_data <= { shift_data[107:0], sum_data };
```

```
end
end
else begin
   shift_data <= { shift_data[107:0], sum_data };
end
end


reg [17:0] sum_fifo_in;
reg [1:0] sswid;
reg lastssw;

always @(posedge clk)begin
 if ( !rst_n ) begin
lastssw <= 1'b0;
 end
        else if ( sswid == LASTSSW) begin
   lastssw <= 1'b1;
 end
 else begin
   lastssw <= 1'b0;
 end
end

reg write_flag;

always @ ( posedge clk or negedge rst_n ) begin
if ( !rst_n ) begin
sum_fifo_in <= 18'b0;
sswid <= 2'b0;
write_flag <= 1'b0;
end
else if ( shift_data[53:36] == RX_FRAME_HEAD ) begin
write_flag <= 1'b1;
end
else if ( shift_data[125:90] == SSW_FRAME_HEAD ) begin
write_flag <= 1'b1;
end
else if ( shift_data[71:36] == SSW_FRAME_TRAIL ) begin
   sum_fifo_in <= lastssw ? RX_FRAME_TRAIL : 18'b0;
sswid <= sswid + 1;
end
   else if ( shift_data[89:54] == SSW_FRAME_TRAIL ) begin
```

```
sum_fifo_in <= 18'b0;
write_flag <= lastssw ? 1'b1 : 1'b0;
  if( lastssw == 1'b1 ) begin
  sswid <= 2'b0;
  end
end
else begin
   sum_fifo_in <= shift_data[71:54];
end
end

wire empty;
wire [17:0] dout;


dump_fifo sum_fifo
(
   .din( sum_fifo_in ),
.rd_clk( clk ),
.rd_en( !empty ),
.rst( !rst_n ),
.wr_clk( clk ),
.wr_en( write_flag ),
.dout( dout ),
.empty( empty ),
.full( full )
);

reg [17:0] sum_out;

always @ ( posedge clk ) begin
sum_out <= !empty ? dout : 18'b0;
end

endmodule




////////////////////// slink2_rx_param ////////////////////////////
`ifdef  GLFIFO_PARAM
`else
`define GLFIFO_PARAM
```

```
‘define FIFO_DATA_WIDTH 18
‘define FIFO_ADDR_WIDTH 13
‘define COARSE_COUNT_WIDTH 4
‘define FIFO_DEPTH 8192

‘define ADDR_FIFO_DATA   4’d0
‘define ADDR_FIFO_STATUS 4’d1
‘define ADDR_LINK_STATUS 4’d2
‘define ADDR_DIV         4’d3
‘define ADDR_SELLECT     4’d4

‘define RESET_PULSE_WIDTH 2

‘endif



////////////////// sync_rst.v ////////////////////////

module sync_rst
(
rst_src_n,
clk,
rst_n
);

parameter WIDTH_PULSE = 2;

input  rst_src_n;
input  clk;
output rst_n;

reg [WIDTH_PULSE:0] rst_sync_n;


always @( posedge clk or negedge rst_src_n ) begin
if ( !rst_src_n )
rst_sync_n <= { (WIDTH_PULSE+1){1’b0} };
else
rst_sync_n <= { rst_sync_n[WIDTH_PULSE-1:0], 1’b1 };
end
```

```
assign rst_n = rst_sync_n[WIDTH_PULSE];

endmodule



////////////////////////////// slink2_rx_sim.v ////////////////////////////

`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:   18:33:28 11/21/2008
// Design Name:   slink2_rx
// Module Name:   C:/Documents and Settings/kessoku/cygwin_HOME/Slink/rx2_ver4/slink2_rx_sim.v
// Project Name:  rx2_ver4
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: slink2_rx
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module slink2_rx_sim;

// Inputs
reg all_reset_b;
reg gclk0;
reg wstr_n;
reg rstr_n;
reg [15:0] rx0;
reg rxclk10;
reg rxdata0;
```

```
reg rxcntl0;
reg [15:0] rx1;
reg rxclk11;
reg rxdata1;
reg rxcntl1;

// Outputs

wire ce0l_b;
wire [17:0] sum_out;

   // input file
   integer i , j;
reg [17:0] pat0, pat1, pattern0[0:99], pattern1[0:99];

   //output file
reg [31:0] mcd;

   // clock
   parameter STEP = 5; //

   always #(STEP/2) begin
gclk0 = ~gclk0;
   rxclk10 = ~rxclk10;
rxclk11 = ~rxclk11;
   end

// Instantiate the Unit Under Test (UUT)
slink2_rx uut (
.all_reset_b(all_reset_b),
.gclk0(gclk0),
.wstr_n( wstr_n ),
.rstr_n( rstr_n ),
.rx0(rx0),
.rxclk10(rxclk10),
.rxdata0(rxdata0),
.rxcntl0(rxcntl0),
.rx1(rx1),
.rxclk11(rxclk11),
.rxdata1(rxdata1),
.rxcntl1(rxcntl1),
.ce0l_b(ce0l_b),
.sum_out(sum_out)
```

```
);

initial begin
// Initialize Inputs
all_reset_b = 1;
gclk0 = 0;
wstr_n = 1;
rstr_n = 1;
rx0 = 0;
rxclk10 = 0;
rxdata0 = 0;
rxcntl0 = 0;
rx1 = 0;
rxclk11 = 1;
rxdata1 = 0;
rxcntl1 = 0;
i = 0;
j = 0;

// Wait 100 ns for global reset to finish
#100;

// Add stimulate
      all_reset_b = 0;
      #(2*STEP);
all_reset_b = 1;

      $readmemb( "run91387_A09_ssw0.dat" , pattern0 );
      $readmemb( "run91387_A09_ssw1.dat" , pattern1 );
      mcd = $fopen( "rx_out.dat" );

      #100;
#(STEP/2);

      for( i=0; i<82; i=i+1) begin
  pat0 = pattern0[i];
         pat1 = pattern1[i];
   #(STEP/2) rxcntl0 = pat0[17];
             rxdata0 = pat0[16];
                   rx0 = pat0[15:0];
   #(STEP/2) rxcntl1 = pat1[17];
             rxdata1 = pat1[16];
                   rx1 = pat1[15:0];
```

```
end

   #3000 $fclose( mcd );
#100 $finish;
end

always begin
   #(STEP-1)$fdisplay( mcd, "%b", sum_out );
#1;
   end

endmodule
```

Tx ロジック

```
//////////////////// slink_tx.v//////////////////////////////

`include "slink_tx_param.v"

module slink_tx
(
    all_reset_b, //reset_signal
 gclk, //global clock

 uwen_b, //tx out for the 32bit
 uctrl_b, //tx out for the 0bit
 ud, //tx out for 32bits data

 ce0l_b, //for DPM
 chain_data //form rx
);

// signal

input  all_reset_b;
input  gclk;// global clock

// to Slink
output uwen_b;
output uctrl_b;
output [31:0] ud;

// to DPM
output ce0l_b;

// from DPM (rx)
input [17:0] chain_data;

//divided clock
wire gclk0;
wire gclk1; //for Slink_fifo_out

tx_out_clk tx_out_clk
(
    .CLKIN_IN( gclk ),
```

```
    .RST_IN( !all_reset_b ),
    .CLKDV_OUT( gclk1 ),
    .CLKIN_IBUFG_OUT(),
    .CLK0_OUT( gclk0 ),
    .LOCKED_OUT()
);

//------------------------------------------------------
// making reset signal

wire rst_local;
wire rst_r_n;     // negated synchronously to gclk0
                  // ( global and local reset )
wire rst_w_n;     // negated synchronously to wclk
                  // ( global and local reset )

rst_mod #( `RESET_PULSE_WIDTH ) rst_mod
(
    .gclk0( gclk0 ), .wclk( gclk0 ),
    .all_reset_b( all_reset_b ),
    .rst_r_n( rst_r_n ), .rst_w_n( rst_w_n )
);

//------------------------------------------------------


//------------------------------------------------------
// fifo signal

wire [35:0] fifo_out;
wire full;
wire empty;

//------------------------------------------------------
// fifo dump

reg [17:0] raw_data;
  always @ ( posedge gclk0 ) begin
 raw_data <= chain_data;
  end

wire [17:0] dump_out;
```

```
fifo fifo
(
.raw_data( raw_data ),
.clk( gclk0 ),
.rst_n( rst_r_n ),
.dump_out( dump_out ),
.read_flag( read_flag )
);

wire [17:0] format_data;

format format
(
.clk( gclk0 ),
.dump_out( dump_out ),
.read_flag( read_flag ), .write_flag( write_flag ),

.full( full ), .empty( empty ),

.rst_n( rst_r_n ),

.format_data( format_data )
);


//-------------------------------------------------------
// fifo

slink_fifo slink_fifo
(
   .din( format_data ),
.rd_clk( gclk1 ),
.rd_en( !empty ),
.rst( !rst_w_n ),
.wr_clk( gclk0 ),
.wr_en( ( format_data[17:16] != 2'b00 ) && ( format_data[17:16] != 2'b11 ) && write_flag ),
.dout( fifo_out ),
.empty( empty ),
.full( full )
);

assign { uctrl_b, ud, uwen_b } = !empty ? { fifo_out[34:18], fifo_out[15:0], 1'b0 } : 34'b1;
```

```
endmodule



//////////////////// fifo.v ////////////////////////////
module fifo
(
raw_data,
clk,
rst_n,
read_flag,

dump_out
);


// signal
input [17:0] raw_data;
input clk;
input rst_n;
input read_flag;
output [17:0] dump_out;

wire [17:0] dout;
wire empty;
wire full;


dump_fifo dump_fifo
(
.din( raw_data ),
.rd_clk( clk ),
.rd_en( !empty ),
.rst( !rst_n ),
.wr_clk( clk ),
.wr_en( raw_data[17:16] != 2'b00 ),
.dout( dout ),
.empty( empty ),
.full( full )
);
```

```
assign dump_out = ( !empty & read_flag ) ? dout : 18'b0;

endmodule




////////////////////// format.v //////////////////////////////////////////////

module format
(
clk,
read_flag, write_flag,

full, empty,

rst_n,

format_data,

dump_out

);

//from RX
parameter RX_HEAD = 18'b10_1111_1010_1100_1110; //0xFACE
parameter RX_TRAIL = 18'b10_1100_1010_1111_1110; //0xCAFE

// header parameter
//parameter SSW_FRAME_HEAD = 36'b10_0000_0000_0000_0000_10_0000_1011_0000_1111;
parameter SSW_FRAME_HEAD = 18'b10_0000_1011_0000_1111;
parameter FRAME_HEAD = 36'b10_1011_0000_1111_0000_10_0000_0000_0000_0000;
parameter HEADER0 = 36'b01_1110_1110_0001_0010_01_0011_0100_1110_1110;
parameter HEADER1 = 36'b01_0000_0000_0000_0000_01_0000_0000_0000_1000;
parameter HEADER2 = 36'b01_0000_0010_0000_0010_01_0000_0010_0000_0010;
parameter HEADER3 = 36'b01_0000_0000_0100_0011_01_0000_0000_0000_0000;
parameter HEADER4 = 36'b01_0000_0000_0000_0000_01_0000_0000_0000_0000;
parameter HEADER5 = 36'b01_0000_0000_0000_0000_01_0000_0000_0000_0000;
parameter HEADER6 = 36'b01_0000_0000_0000_0000_01_0000_0000_0000_0000;
parameter HEADER7 = 36'b01_0000_0000_0000_0000_01_0000_0000_0000_0000;
parameter HEADER8 = 36'b01_0000_0000_0000_0000_01_0000_0000_0000_0000;

// status parameter
```

```
parameter STATUS0 = 36'b01_0000_0000_0000_0000_01_0000_0000_0000_0000;
parameter STATUS1 = 36'b01_0000_0000_0000_0000_01_0000_0000_0000_0000;
parameter STATUS2 = 36'b01_0000_0000_0000_0000_01_0000_0000_0000_0010;
parameter STATUS3 = 36'b01_0000_0000_0000_0000_01_0000_0000_0000_0000;

// fragment parameter
// this parat will data elemnt.
// but we don't care this on soft ware. becouse I change that.
parameter FRAGMENT0 = 36'b01_0000_0000_0000_0000_01_0000_0000_0000_0000;

// trailer parameter
//parameter SSW_FRAME_TRAIL = 36'b10_0000_1110_0000_1111_10_0000_0000_0000_0000;
parameter SSW_FRAME_TRAIL = 36'b10_0000_1110_0000_1111;
parameter FRAME_TRAIL = 36'b10_1110_0000_1111_0000_10_0000_0000_0000_0000;
parameter TRAILER0 = 36'b01_0000_0000_0000_0000_01_0000_0000_0000_0100;//0x00000004
// trailer1 is number of data elements
// count at slink_tx
// now control word b0a0_c0a0
parameter TRAILER1 = 36'b10_1011_0000_1010_0000_10_1100_0000_1010_0000;//0xb0a0c0a0
parameter TRAILER2 = 36'b01_0000_0000_0000_0000_01_0000_0000_0000_0000;//
//parameter TRAILER1B = 18'b10_1100_0000_1010_0000;


// signal

input clk;
input [17:0] dump_out;
input rst_n;

input full, empty;

output read_flag, write_flag;
output [17:0] format_data;


////////////////////// make header and trailer ////////////////
////////////////// data count will be made at tx ///////////////

// sellect data
reg [17:0] fifo_in;


always @ ( posedge clk ) begin
```

```
fifo_in <= dump_out;
end

// remove idle
// shift data
reg [71:0] data_delay;

always @ ( posedge clk or negedge rst_n ) begin
if ( !rst_n ) begin
data_delay[71:0] <= 72'b0;
end
else begin
data_delay[71:0] <= { data_delay[53:0], fifo_in };
end
end



reg pre_header_flag;
reg pre_trailer_flag;


always @ ( posedge clk or negedge rst_n ) begin
if ( !rst_n ) begin
pre_header_flag <= 1'b0;
pre_trailer_flag <= 1'b0;
end

else if ( data_delay[17:0] == RX_HEAD ) begin
pre_header_flag <= write_flag ? 1'b0 : 1'b1;
pre_trailer_flag <= 1'b0;
end

else if ( data_delay[17:0] == RX_TRAIL ) begin
pre_header_flag <= 1'b0;
pre_trailer_flag <= write_flag ? 1'b1 : 1'b0;
end
else begin
pre_header_flag <= 1'b0;
pre_trailer_flag <= 1'b0;
end
end
```

```
reg [3:0] header_flag;
reg [31:0] trailer_flag;

always @ ( posedge clk ) begin
header_flag <= { header_flag[2:0], pre_header_flag };
trailer_flag <= { trailer_flag[31:0], pre_trailer_flag };
end



reg [575:0] raw_data;

always @ ( posedge clk ) begin

if ( header_flag[2:1] == 2'b01 ) begin
raw_data
<= { 72'b0, FRAME_HEAD,
     HEADER0, HEADER1, HEADER2, HEADER3, HEADER4, HEADER5, HEADER6, HEADER7, HEADER8,
  STATUS0, STATUS1, STATUS2, STATUS3, FRAGMENT0 };

end
else if ( trailer_flag[31:30] == 2'b01 ) begin
raw_data <= {18'b0, TRAILER0, TRAILER1, TRAILER2, FRAME_TRAIL, 414'b0 };
end
else begin
raw_data[575:0] <= { raw_data[557:0], data_delay[71:54] };
end
end



reg write_flag;

always @ ( posedge clk or negedge rst_n ) begin
if ( !rst_n ) begin
write_flag <= 1'b0;
end
else if ( raw_data[539:504] == FRAME_HEAD ) begin
write_flag <= 1'b1;
end
else if ( raw_data[575:540] == FRAME_TRAIL ) begin
```

```
write_flag <= 1'b0;
end
end



reg read_flag;

always @ ( posedge clk or negedge rst_n ) begin
if ( !rst_n ) begin
read_flag <= 1'b1;
end
else if ( data_delay[17:0] == RX_TRAIL ) begin
read_flag <= 1'b0;
end
else if ( !write_flag ) begin
read_flag <= 1'b1;
end
end

assign format_data = raw_data[557:540];

endmodule




////////////////////// rst_mod.v //////////////////////

module rst_mod
(
    gclk0, wclk,
    all_reset_b,
    rst_r_n, rst_w_n
);

parameter RESET_PULSE_WIDTH = 2;

input  gclk0;
input  wclk;

input  all_reset_b;
```

```
output rst_r_n;
output rst_w_n;

sync_rst #( RESET_PULSE_WIDTH ) sync_rst_r
(
    .rst_src_n( all_reset_b ),
    .clk( gclk0 ),
    .rst_n( rst_r_n )
);

sync_rst #( RESET_PULSE_WIDTH ) sync_rst_w
(
    .rst_src_n( all_reset_b ),
    .clk( wclk ),
    .rst_n( rst_w_n )
);

endmodule
```

```
////////////////// slink_tx_param.v ////////////////////

‘ifdef  GLFIFO_PARAM
‘else
‘define GLFIFO_PARAM

‘define FIFO_DATA_WIDTH 18
‘define FIFO_ADDR_WIDTH 13
‘define COARSE_COUNT_WIDTH 4
‘define FIFO_DEPTH 8192

‘define ADDR_FIFO_DATA   4’d0
‘define ADDR_FIFO_STATUS 4’d1
‘define ADDR_LINK_STATUS 4’d2
‘define ADDR_DIV         4’d3
‘define ADDR_FLAG        4’d4

‘define RESET_PULSE_WIDTH 2
```

```
‘endif



////////////////// sync_rst.v ///////////////////////

module sync_rst
(
rst_src_n,
clk,
rst_n
);

parameter WIDTH_PULSE = 2;

input  rst_src_n;
input  clk;
output rst_n;

reg [WIDTH_PULSE:0] rst_sync_n;


always @( posedge clk or negedge rst_src_n ) begin
if ( !rst_src_n )
rst_sync_n <= { (WIDTH_PULSE+1){1’b0} };
else
rst_sync_n <= { rst_sync_n[WIDTH_PULSE-1:0], 1’b1 };
end

assign rst_n = rst_sync_n[WIDTH_PULSE];

endmodule




////////////////////////////// tx_sim.v ////////////////////////////////

‘timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
```

```
//
// Create Date:   20:03:39 12/21/2008
// Design Name:   slink_tx
// Module Name:   C:/Documents and Settings/kessoku/cygwin_HOME/Slink/tx_ver4/tx_sim.v
// Project Name:  tx_ver4
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: slink_tx
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////
module tx_sim;


/////// Inputs //////////////
reg all_reset_b;
reg gclk;
reg [17:0] chain_data;

/////// Outputs /////////////
wire uwen_b;
wire uctrl_b;
wire [31:0] ud;
wire ce0l_b;

//////// clock ///////////////
parameter STEP = 20;

always #(STEP/2) begin
gclk = ~gclk;
end

/////// input file /////////////
integer i , j;
reg [17:0] pat0, pattern0 [0:199];// for rx0
```

```
/////// output file ////////////
reg [31:0] mcd;


////// Instantiate the Unit Under Test (UUT) ///////////////
slink_tx uut (
.all_reset_b(all_reset_b),
.gclk(gclk),
.uwen_b(uwen_b),
.uctrl_b(uctrl_b),
.ud(ud),
.ce0l_b(ce0l_b),
.chain_data(chain_data)
);


//////// begin initial ///////////////
initial begin
all_reset_b = 1;
gclk = 0;
chain_data = 0;
i = 0;
j = 0;

  ///// Wait 100 ns for global reset to finish ///////
      #100;
      all_reset_b = 0;
#100;
all_reset_b = 1;

      mcd = $fopen( "tx_out.dat" );


      $readmemb( "100M_rx_out.dat" , pattern0 );

#300;
      #(STEP/2);

 /////// for ppg_data ///////////////////////
for( i=0; i<171; i=i+1 )begin
   pat0 = pattern0[i];
#STEP chain_data = pat0;
end
```

```
#5000 $fclose( mcd );
      #100 $finish;
end
////////// end initial ////////////////////

   always begin
   #(2*STEP-1)$fdisplay( mcd, "%b", ud);
#1;
end

endmodule
```