修士論文

ATLAS TGC エレクトロニクス読み出し系の開発

京都大学大学院理学研究科 物理学宇宙物理学専攻
物理学第二分野 高エネルギー研究室
西田 昌平

2000 年 2 月 1 日

概要

LHC（大型陽子陽子衝突型加速器）は CERN 研究所に建設が予定されている世界最大のハドロンコライダーで、2005 年の運転開始を目指している。ATLAS は LHC の測定器の一つで、ヒッグス粒子や超対称性粒子の探索などエネルギーフロンティアでの新しい物理現象をとらえるための汎用の検出器である。

TGC（Thin Gap Chamber）は ATLAS 検出器のエンドキャップ部分を覆っているミューオントリガー用のチェンバーで、トロイダル磁場を用いてミューオンの横方向の運動量 p_{T} を測定し、トリガー信号を生成する。また、TGC は精密測定用のミューオンチェンバーでは測定されない ϕ 方向の情報と、各々のミューオンイベントがどのバンチに属しているかを識別するための時間情報を与える。

TGC のヒット情報は、デジタル化とバンチの識別が行われた後、Slave Board と呼ばれるモジュールでレベル 1 のトリガー判定が出るまでの 2.5 μs の間データを保持し続ける。そして、レベル 1 で採用されたイベントのデータは一旦デランダマイザと呼ばれるバッファに蓄えられたあと、シリアル変換されて Star Switch と呼ばれるモジュールに送られる。Star Switch は約 20 個の Slave Board から受け取ったデータを受け取り、圧縮してから 80 m 離れたエレクトロニクス室にある Local DAQ Master に送信する。

今回、TGC のエレクトロニクスのうち、これらの読み出し系に関わる部分の設計と検証を行った。まず、Slave Board や Star Switch に関して HDL（ハードウェア記述言語）を用いて論理回路の設計を行った。一方で、これらの論理回路の検証のために FPGA を実装したプロトタイプ用モジュールの製作を行った。

Slave Board は最終的には ASIC を用いて開発する。そこで、ASIC に実装すべき HDL 記述を FPGA に実装して、読み出しのスキームが正しいかどうかを確認した。その一方で VDEC（東京大学大規模集積システム設計教育研究センター）を利用して ASIC の試作を行った。

Star Switch については、特に Slave Board からデータを受け取る部分を HDL で記述した。そして、最終的に必要なロジックの 3 分の 1 程度の大きさのロジックを FPGA に実装して動作の確認を行った。その結果、30 MHz 以下では動作することが分かった。これをもとに、今後どのようなプロトタイプモジュールを作成すればよいかを考察した。

目 次

図 目 次

表 目 次

第 1 章

ATLAS 実験の概要

1.1 Large Hadron Collider

素粒子物理学の標準模型は素粒子の世界の様々な現象を極めて良く説明している。しかし、電弱相互作用での対称性の自発的破れがゲージ粒子やフェルミオンに質量を与えるというヒッグス機構の鍵となるヒッグス粒子はいまだに発見されていない。LHC 計画の最大の目的はこのヒッグス粒子の探索である。

LHC は Large Hadron Collider の略で、CERN 研究所に建設が予定されている周長 27 km の大型陽子陽子衝突型加速器である。LHC の主なパラメータを表 1.1 に示す。

	High Luminosity	Low Luminosity
Energy at collision	7 TeV + 7 TeV	
Luminosity	1×10^{34} cm^{-2}s^{-2}	1×10^{33} cm^{-2}s^{-2}
Current	0.56 A	0.087 A
Luminosity lifetime	10 h	
Bunch separation	24.95 ns	
Bunch crossing rate	40.08 MHz	
Circumference	26.66 km	

表 1.1： LHC 加速器の主なパラメータ。実験開始後の数年間は低ルミノシティで運転される。ここでは、高ルミノシティと低ルミノシティの両方の値を示す。

LHC 加速器は、現在行われている LEP 実験が 2002 年に終了した後、LEP 加速器を撤去した後のトンネル内に建設される。衝突点は 4 ヶ所で、汎用の検出器である ATLAS（A Troidal LHC Apparatus）と CMS（The Compact Muon Solenoid）、重イオン衝突実験用の ALICE（A Large Ion Collider Experiment）、B の物理のための LHC-B の 4 つの検出器が設置される。

ATLAS 検出器は、ヒッグス探索、超対称性粒子の探索、B の物理、トップクォークの物理などを視野に入れた汎用の検出器で、直径 22 m、長さ 44 m、重さ 7000 t という巨大な検出器である。そのためプロジェクト自体も 33 ヶ国の 150 以上の研究機関の共同研究という、かつてない大規模なものとなっている[1]。

[1] このような大規模プロジェクトがどのようにまとめられているかについては Appendix A で少し触れる。

1.2 ATLAS がめざす物理

ここでは ATLAS 実験がめざす物理について述べる。

1.2.1 標準理論ヒッグスの探索

標準理論における中性スカラーヒッグス粒子 H の探索が、ATLAS 実験の最大の目的である。

ヒッグス粒子の生成過程を図 1.1 に、それぞれの生成過程の散乱断面積を図 1.2 に示す。図 1.2 のように、ヒッグス粒子は主に gluon-gluon 融合によって生成されるが、ヒッグスの質量 m_H が大きいときには W-W 融合（$qq \to Hqq$）の寄与が大きくなる。

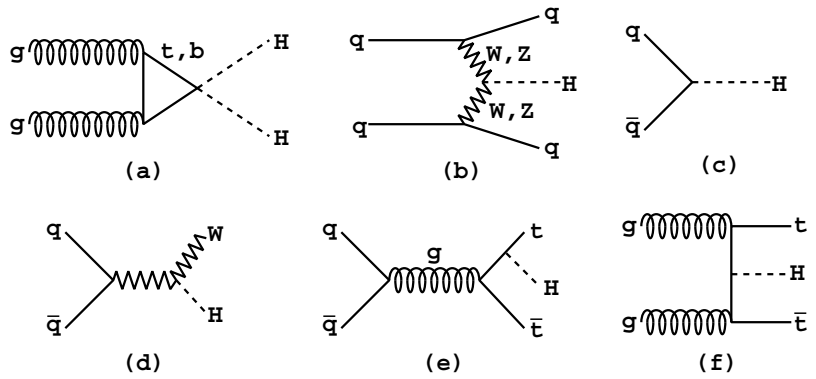

図 1.1： ヒッグス粒子の生成過程。(a) gluon-gluon 融合 (b) W-W 融合、Z-Z 融合 (c) q-$\bar{q}$ 融合 (d) W を伴う生成 (e) (f) $t\bar{t}$ を伴う生成

ATLAS では m_H が 80 GeV から 1 TeV の範囲の探索が可能である。ヒッグス粒子の分岐比を図 1.3 に示す。図からわかるように、ヒッグスの崩壊の仕方は m_H によって異なり、それに伴って探索の方法も m_H によって異なってくる。以下、探索に用いる崩壊モードを示す。

$t\bar{t}$ 又は W を伴って生成した H による $H \to b\bar{b}$

$H \to b\bar{b}$ は $m_H < 2m_W$ の領域では最も分岐比が大きい崩壊モードであるが、このモードを有効にトリガーする方法がない。そこで、W や $t\bar{t}$ を伴って H が生成した場合を考える。この場合、終状態は $l\nu b\bar{b}$ や $l\nu jjb\bar{b}b\bar{b}$ となるので、W や $t\bar{t}$ に由来する p_{T} が大きく孤立したレプトンでトリガーを行うことができる。これらのモードには、WZ（$t\bar{t}Z$）や $Wb\bar{b}$（$t\bar{t}b\bar{b}$）に由来するバックグラウンドなど、数多くのバックグラウンドが存在する。

これらのモードがヒッグス探索に有用なのは、おおよそ $80 < m_H < 125$ GeV の時である。

$H \to \gamma\gamma$

$H \to \gamma\gamma$ は $100 < m_H < 150$ GeV の場合に最も重要な崩壊モードである。このモードは、$q\bar{q} \to \gamma\gamma$ や $g\bar{g} \to \gamma\gamma$ から来る連続スペクトルが irreducible なバックグラウンドであり、このバックグラウンド上の狭い質量のピークを観測する必要がある。そのために、電磁カロリーメータには優れたエネルギー及び角度分解能が要求される。また、jj、$j\gamma$ 及び $Z \to ee$ の粒子識別を誤った場合、これらはバックグラウンドになる。それゆえ、検出器は粒子識別にも優れている必要がある。

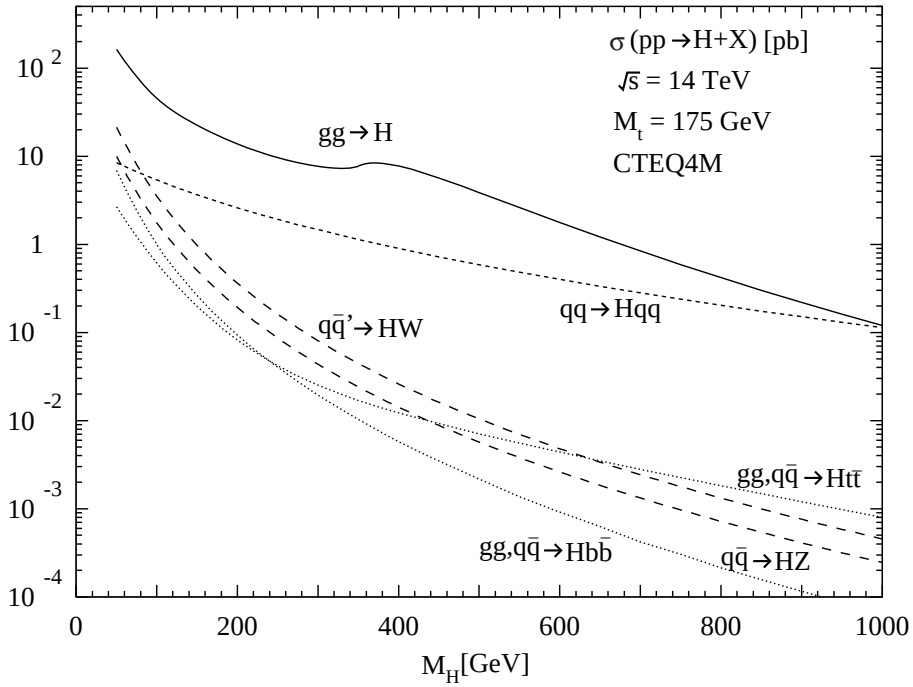

図 1.2： ヒッグス粒子の生成過程別の散乱断面積

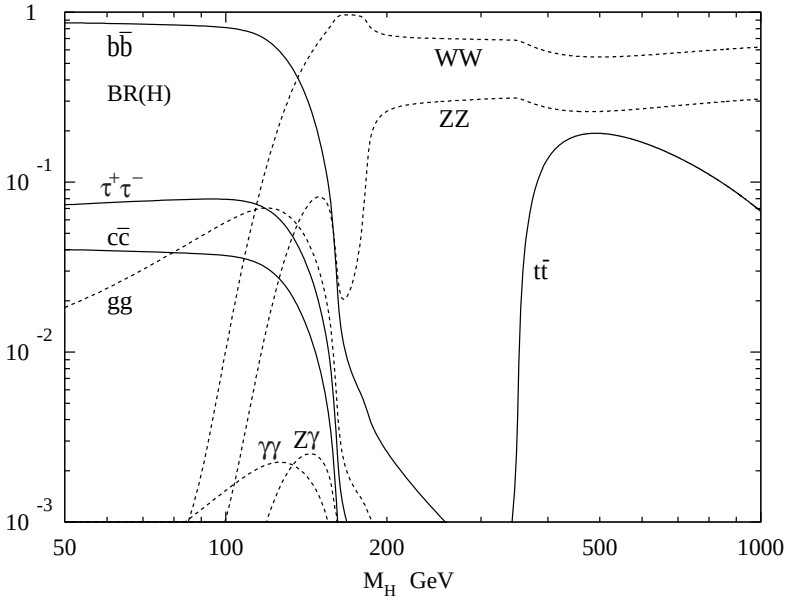

図 1.3： ヒッグス粒子の主な崩壊過程に対する分岐比

一方、H が生成する際に W や Z、$t\bar{t}$ も同時に生成するような反応は、断面積は小さくなるものの W などから来るレプトンをタグすることによって S/N 比が向上する。このような W などを伴う $H \to \gamma\gamma$ モードを利用することによって、おおよそ $80 < m_H < 140$ GeV の領域についてヒッグス探索を行うことが出来る。

$H \to ZZ^* \to 4l$

崩壊モード $H \to ZZ^* \to 4l$ は 120 GeV $< m_H < 2m_Z$ の領域で比較的きれいなシグナルとなる。但し、図 1.3 に示すように、$150 < m_H < 180$ GeV の領域では $H \to WW$ の崩壊モードが開く関係で $H \to ZZ^* \to 4l$ の分岐比は小さくなってしまう。そのため、$m_H = 170$ GeV 近辺ではこのモードは多少苦しくなってしまう。

このモードのバックグラウンドとしては、ZZ^* や $Z\gamma^*$ からの連続スペクトル（irreducible）、及び $t\bar{t}$ や $Zb\bar{b}$ の生成による reducible なものがある。

このモードに対する最も単純なトリガー条件は以下の様なものである。

- $|\eta| < 2.5$ の領域に $p_{\mathrm{T}} > 20$ GeV のレプトンが 2 つ存在すること[2]。
- これに加えて、$|\eta| < 2.5$ の領域に $p_{\mathrm{T}} > 7$ GeV のレプトンが 2 つ存在すること。
- 上記の 4 つのレプトンのうちのあるレプトンの対の不変質量が、分解能の範囲内で m_Z に等しいこと。レプトンの対を選ぶ際には当然電荷やフレーバーを考慮に入れなければならない。このカットを行うことによって $t\bar{t}$ のバックグラウンドを落とすことが出来る。
- 残りのレプトン対の不変質量が適当な閾値を越えていること。これにより、$Z\gamma^*$ や $Zb\bar{b}$ からの寄与を減らすことが出来る。

$t\bar{t}$ や $Zb\bar{b}$ に由来するレプトンは b クォークの崩壊で生じたものである。b クォークはブーストされているため、これらのレプトンは近接している。そのため、レプトンが孤立しているということ要求をすることによってバックグラウンドを減らすことが出来る。さらに、b クォークは寿命が長いので、レプトンのインパクトパラメータを利用してバックグラウンドを落とすことが可能である。

$H \to WW^{(*)} \to l\nu l\nu$

$m_H \cong 170$ GeV の時には $H \to WW$ のモードが開けてくるので、$H \to ZZ^* \to 4l$ のモードは suppress されてしまい、$H \to WW^{(*)} \to l\nu l\nu$ のモードの方が分岐比がはるかに大きくなる。しかし、このモードは終状態にニュートリノが含まれるために解析は容易ではない。このモードにも、$WW^* \to l\nu l\nu$ や $t\bar{t}$ など多数のバックグラウンドが存在する。

この他、ヒッグスの生成時に W も生成されたような反応に注目し、3 つのレプトンでトリガーをかけるという方法も補助的に用いることができる。

$H \to ZZ \to 4l$

$H \to ZZ \to 4l$ はヒッグス探索の最も基本的な崩壊モードで、$180 < m_H \lesssim 700$ GeV の時に使うことが出来る。バックグラウンドは連続スペクトルから来る ZZ であるが、さほど大きなバックグラウンドではない。ヒッグスの質量が大きい時にはヒッグスの質量幅が検出器のエネルギー分解能よりも大きくなるので、検出器の性能は探索にはさほど影響を与えない。このモードでの探索は積算ルミノシティーによって決まると言える。

[2] η については 15 頁の脚注参照

$H \to ZZ^{(*)} \to ll\nu\nu$

$m_H > 700$ GeV が大きい領域ではヒッグスの質量幅が広がってしまい、前述の $H \to ZZ \to 4l$ のモードではレートが不足してしまう。それゆえ、ヒッグスの質量が TeV に近い領域では、レートが大きな $H \to ZZ^{(*)} \to ll\nu\nu$ モードや $H \to WW^* \to l\nu jj$ を用いる。

$H \to ZZ^{(*)} \to ll\nu\nu$ のモードでは、Z を組める 2 つのレプトンが存在することと、大きな $E_{\mathrm{T}}^{\mathrm{miss}}$ があることがシグナルの条件となっている。このモードは $400 < m_H < 900$ GeV で観測可能だが、$E_{\mathrm{T}}^{\mathrm{miss}}$ があるために質量のピーク等があるわけではない。そのためには断面積のバックグラウンドからのずれを見ることになり、バックグラウンドに対する正確な理解が必要となる。

$qq \to qqH$ で生成した H の $H \to WW^* \to l\nu jj$

m_H が TeV の領域で最も有力なモードである。higs-p_{T} のレプトンが存在すること、大きな $E_{\mathrm{T}}^{\mathrm{miss}}$ が存在すること、近接した 2 つのジェットが存在することが条件となる。ジェットが近接しているのは W がブーストされているからである。この m_H の領域では W-W 融合が有効となってくるので、このときに前方に生じる 2 つのクォークジェットをとらえるとバックグラウンドを減らすことが出来る。

まとめ

図 1.4 に、積算ルミノシティが 30 fb^{-1} の時の ATLAS 検出器の標準理論ヒッグス探索に対する感度を示す [7]。この図から分かるように、ATLAS 検出器は 80 GeV $< m_H <$ 1 TeV の範囲を網羅している。積算ルミノシティ 30 fb^{-1} は、低ルミノシティ運転数年分に相当する。このことから、最初の数年間の低ルミノシティ運転で標準理論ヒッグスの探索が可能であることがわかる。また、図に示されるように、ほとんどの m_H 領域では 2 つ以上のモードを用いて探索が可能で、相互に確認し合うことによって十分な信頼性を得ることが出来る。

1.2.2 超対称性ヒッグスの探索

SUSY（超対称性）理論を導入し標準理論を最低限に拡張した MSSM（Minimal Supersymmetric extention of Standard Model）では、ヒッグスの 2 重項が必要で、最終的には $H^{\pm}$, H, h, A の 5 つのヒッグスが生じることになる。このうち、A は CP odd の中性ヒッグス、H, h は CP even の中性ヒッグスで、$m_h < m_H$ と定める。

これら 5 個のヒッグスの質量は、ツリーレベルでは 2 つの真空期待値の比 $\tan\beta$ と m_A だけで表すことができる。しかし、radiative correction を考慮すると実質的には自由な値になりうる。

超対称性ヒッグスも、$h \to \gamma\gamma$、$h \to b\bar{b}$、$H \to ZZ \to 4l$ のように標準理論ヒッグスと同じ崩壊モードで探索可能なものもある。超対称性ヒッグスでしか探索できないモードとしては、$A \to \tau\tau$ や $A \to \mu\mu$ などが存在する。前者は τ の少なくとも一方がレプトンに崩壊した場合を考え、レプトンでトリガーを行う。しかし、必ず ν が生成してしまうので、解析はさほど容易ではない。

その他、$H/A \to t\bar{t}$, $A \to Zh$, $H \to hh$ も注目されているモードである。

1.2.3 超対称性粒子の探索

SUSY（超対称性）粒子の探索も ATLAS 実験の重要な目的の一つである。SYSY によると、世の中に存在するフェルミオンに対してはボゾンの、ボゾンに対してはフェルミオンの SUSY 粒子（スーパーパートナー）が存在する。例えば、スピン 1/2 のクォーク q のスーパーパートナーはスピン 0 のスクォーク $\tilde{q}$ であり、スピン 1 のグルーオン g のスーパーパートナーはスピン 1/2 のグルイーノ $\tilde{g}$ である。ゲージ

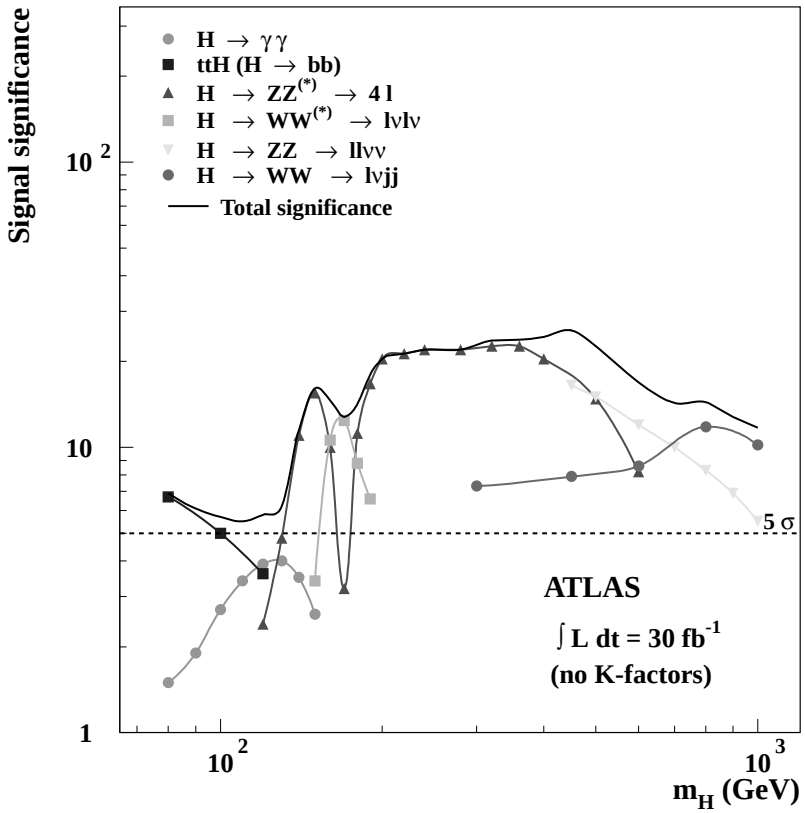

図 1.4： ATLAS 検出器のヒッグス粒子に対する感度。積算ルミノシティ 30 fb^{-1} を仮定している。

粒子とヒッグス粒子のスーパーパートナーは混合を起こしており、質量の固有状態としてはチャージーノ $\tilde{\chi}_i^{\pm}$ やニュートラリーノ $\tilde{\chi}_i^0$ となっているとされている。

ここで B をバリオン数、L をレプトン数、S をスピン量子数として $R = (-1)^{3(B-L)+2S}$ という量を考えると、通常の粒子に対しては $R = +1$、SUSY 粒子に対しては $R = -1$ となる。多くのモデルではこの R パリティは保存するとされている。R パリティが保存すると仮定すると、SUSY 粒子は必ず対になって生成することになる。生成した SUSY 粒子は不安定であると考えられるので、軽い粒子への崩壊を繰り返していく。しかし、R パリティが保存するので、最も軽い SUSY 粒子（LSP = Lightest Supersymmetric Particle）は安定でなければならない。LSP は宇宙論の要請から電荷やカラーをもたない中性の粒子でなければならず、ニュートラリーノ $\tilde{\chi}_1^0$、グラビティーノ $\tilde{g}_3/2$、グルイーノ $\tilde{g}$ が候補となっている。LSP は通常の粒子とは弱い相互作用しかしないので、検出することは出来ない。

実験では、生成した SUSY 粒子が LSP に至るまで繰り返し崩壊する際に生じるジェットやレプトン、及び LSP が検出器の外に逃げてしまうことによって生じる $E_{\mathrm{T}}^{\mathrm{miss}}$の検出が基本となる。現在、考えられている探索方法は、(a) マルチジェット + $E_{\mathrm{T}}^{\mathrm{miss}}$ (b) 2 つの同符号のレプトン + jet + $E_{\mathrm{T}}^{\mathrm{miss}}$ (c) 3 つのレプトン + $E_{\mathrm{T}}^{\mathrm{miss}}$ などがある。

1.2.4 B の物理

LHC では大量の b や t が生成される。$b\bar{b}$ の断面積は 100 μb であると見積もられており、10^{33} cm^{-2}s^{-2} の低ルミノシティ運転時でさえも 1 年間に 10^{12} 個の $b\bar{b}$ 対が生成されることとなる。これは e^+e^- コライダーを用いた B ファクトリ実験である Belle 実験や BaBar 実験よりもはるかに多い。この豊富に生成される b を用いて、CP 非保存の精密測定や CKM（Cabbibo-Kobayashi-Maskawa）行列の角度測定を行うことが出来る。また、Belle や BaBar 実験では生成されなかった B_S も大量に生成されるので、$B_S\bar{B_S}$ 振動の測定（Δm_S の測定など）も可能である。

以下のモードが CKM 行列の角度の測定に用いることが出来ると期待されている。

- $B_d^0 \to J/\psi K_S$ ：$\sin 2\beta$ の測定
- $B_d^0 \to \pi\pi$ ：$\sin 2\alpha$ の測定
- $B_s^0 \to J/\psi\phi$ ：γ の測定
- $B_d^0 \to D^0 K^{0*}$ ：γ の測定

その他、大量に生成される t を用いて t の質量の精密測定などが行われる。

1.3 ATLAS 検出器

1.3.1 ATLAS 検出器の概要

ATLAS 検出器は、前節で述べた物理をはじめエネルギーフロンティアーでの様々な物理事象を研究できるように設計されている。ATLAS 検出器に要求される性能をまとめると以下のようになる。

- LHC は最終的には 10^{34} cm^{-2}s^{-2}という高ルミノシティで運転される。ATLAS 検出器には数多くの電子、光子、ミューオン、ジェットなどが高いレートで到達する。検出器はこの高いレートに耐えられるものでなければならない。
- そのなかでもレプトンの検出は重要である。これらを確実に識別し精度良く測定できること。また低い $p_{\rm T}$ でもトリガーをかけることができること。
- ジェットや $E_{\rm T}^{\rm miss}$ も精度良く測定可能で、トリガーをかけることができること。
- b-tagging、運動量測定用のトラッキングが可能なこと。
- 高ルミノシティに由来する大量のバックグラウンド放射線に対して耐久性があること。
- LHC は 25 ns のバンチ間隔で運転されるが、各々の事象とバンチの対応がとれること。
- 検出器はできるだけ広い η を覆うこと[3]。また、支柱等が原因でできる穴はできるだけ少ないこと。

図 1.5 に ATLAS 検出器の全体図を載せる。これは全長 44 m 直径 22 m 重さ 7000 t という巨大な検出器である。検出器は内側から

- 内部検出器
- 半径 1.2m、長さ 5.3m の 2T 超伝導ソレノイド

[3] η は pseudo rapidity と呼ばれ、θ をビーム軸とのなす角とすると $\eta = -\ln(\tan(\theta/2))$ で定義される。ハドロンコライダーでは散乱断面積がおおよそ η に比例するので、この量が良く用いられる。

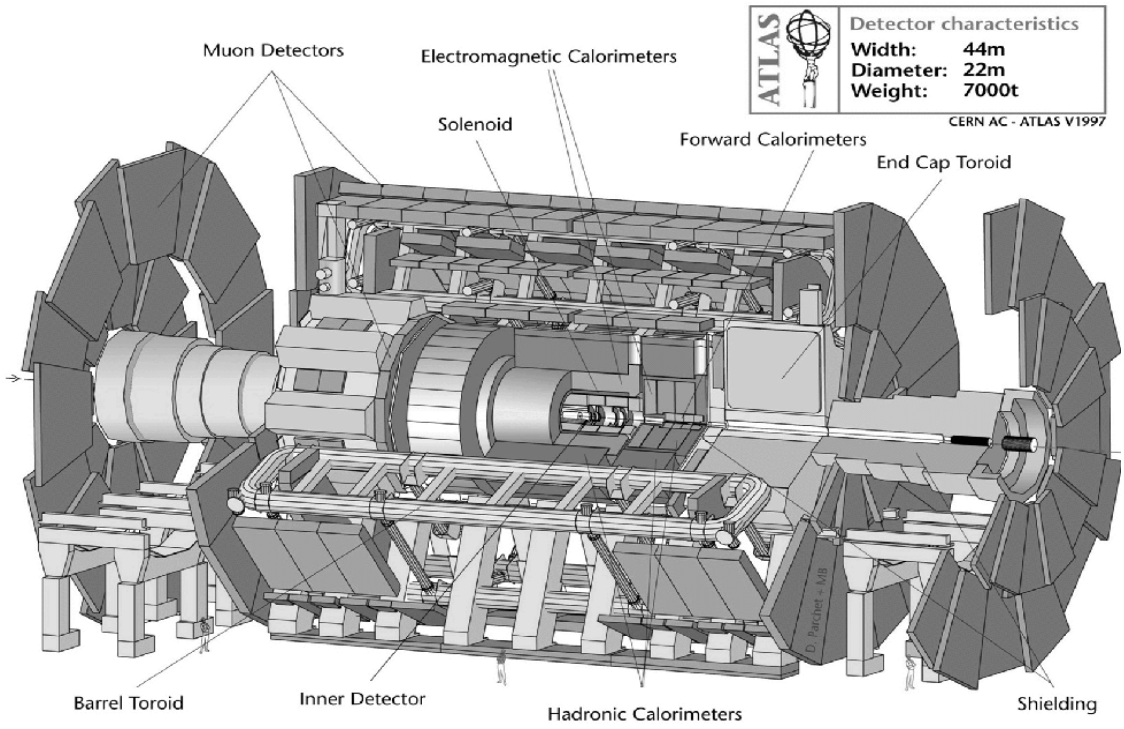

図 1.5： ATLAS 検出器

- カロリーメータ
- 空芯トロイダル磁石
- ミューオンスペクトロメータ

という構成になっている。

1.3.2 内部検出器

内部検出器は荷電粒子のトラッキングを主な目的としており、ソレノイド磁場と組み合わせて運動量の測定を行う。内部検出器の断面図を図 1.6 に示す。

最も内側にある検出器がピクセル検出器である。これは 1 つの要素が 50 μm × 300 μm という高分解能の半導体検出器である。この検出器の分解能がバーテックスの精度を決める。その外側にはシリコン・ストリップ検出器と呼ばれる半導体検出器がある。単に SCT（Semiconductor Tracker）と称されることもある。これは各々の要素が細長いストリップになっている。

最も外側には半径 4 mm のストロー検出器を束ねて作った TRT（Transition Radiation Tracker）と呼ばれる検出器を設置する。トラッカーとしての働きの他に、transition radiation を利用して電子識別を行う。

内部検出器のパラメータを表 1.2 に示す [5]。

1.3.3 カロリーメータ

ATLAS のカロリーメータは電磁用とハドロン用があるが、耐放射線性を考えて LAr（液体アルゴン）カロリーメータを基本としている。

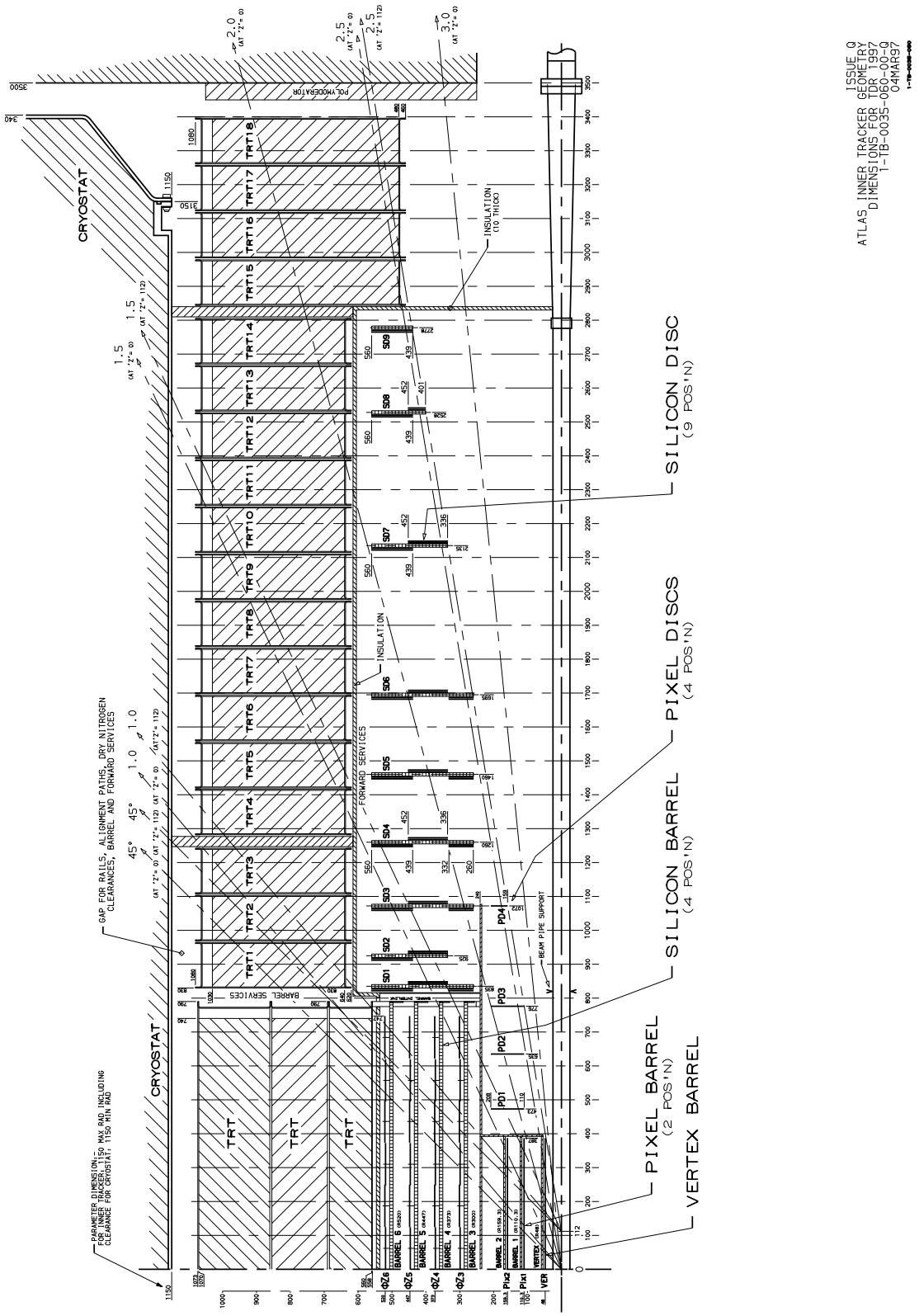

図 1.6： ATLAS 内部検出器の断面図

System	Position	Layer	σ (μm)	Channels	η coverage
Pixels	barrel‡	1	$R\phi = 12,\ z = 66$	16×10^6	± 2.5
	barrel	2	$R\phi = 12,\ z = 66$	81×10^6	± 1.7
	endcap	4	$R\phi = 12,\ z = 77$	43×10^6	$1.7 \sim 2.5$
Silicon Strips	barrel	4	$R\phi = 16,\ z = 580$	3.2×10^6	± 1.4
	endcap	4	$R\phi = 16,\ z = 580$	3.0×10^6	$1.4 \sim 2.5$
TRT	axial		170(per straw)	0.1×10^6	± 0.7
	radial		170(per straw)	0.3×10^6	$0.7 \sim 2.5$

‡バレル部分のピクセル検出器の最内層は、高ルミノシティー運転時に取り外せるように設計されている。

表 1.2： 内部検出器の主なパラメータ。

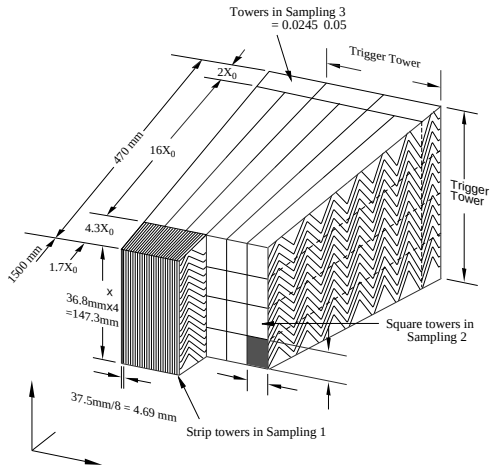

図 1.7： アコーディオン型電磁カロリーメータ

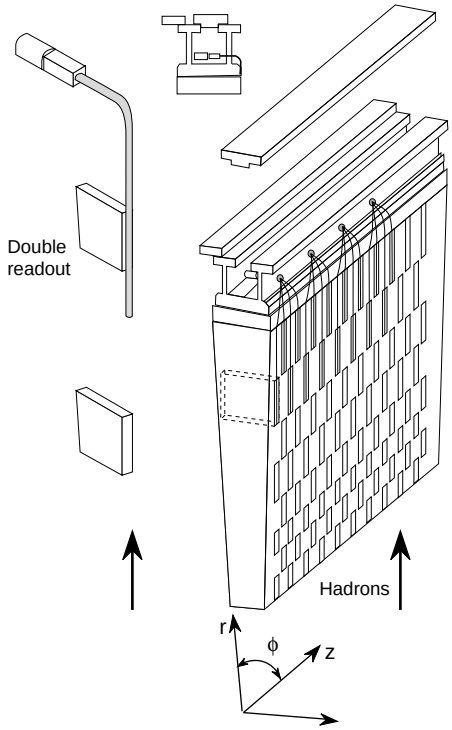

図 1.8： ハドロン・タイルカロリーメータ

電磁カロリーメータは、鉛の吸収体と LAr が図 1.7 のようにアコーディオン構造をなしたもので、バレル、エンドキャップ部をあわせて $|\eta| < 3.2$ の領域を覆っている。もっとも内側の部分はプリサンプラー（presampler）と呼ばれ、ストリップ状に区分けされており、η 方向の分解能を高めている。電磁カロリーメータは、$H \to \gamma\gamma$ や $H \to ZZ^{(*)} \to 4e$ などの反応から $10\%/\sqrt{E[\mathrm{GeV}]} + 1\%$ のエネルギー分解能、$40\mathrm{mrad}/\sqrt{E[\mathrm{GeV}]}$ の角度分解能が要求されている。

バレル部分のハドロン・タイル・カロリーメータ（図 1.8 ）は、鉄とシンチレータのサンドイッチ構造になっている。シンチレータの面の向きがハドロンの進行方向と平行になっている点が特徴である。読み出しは WLS（Wave Length Shift）ファイバーを用いて行なわれる。

エンドキャップ用のハドロン・カロリーメータは、耐放射線性を考慮して銅板を吸収体とした LAr カロ

リーメータである。これは $1.5 < |\eta| < 3.2$ の領域を覆っている。さらに、$3.2 < |\eta| < 4.9$ には前方カロリーメータがおかれる。

1.3.4 ミューオン検出器

ヒッグス粒子探索をはじめ、ATLAS 実験で標的となる崩壊モードには終状態にレプトンが含まれることが多い。レプトンの中でもミューオンは粒子識別が容易で、バックグラウンドも比較的少ない。それゆえ、高い p_{T} をもつミューオンを確実に精度良くとらえることが要求されている。

ATLAS 実験では、ミューオン測定器系だけでミューオンの位置と運動量を測定できるようにトロイダル磁場（ϕ 方向の磁場）が空芯トロイダル磁石によってかけられている。これにより、ミューオンの位置を 2 ヶ所以上で測定すれば、その曲がり具合によって p_{T} を知ることができる。

図 1.9 にミューオン検出器の R-z 断面図を示す。ミューオン検出器は、ミューオンの信号を最大限に活用するために精密測定用とトリガー用のものが別に用意されており、以下のような 4 種類の検出器が存在する。

MDT（Monitored Drift Tube） 精密測定用。

CSC（Cathode Strip Chamber） 精密測定用。エンドキャップ部のごく一部のみ。

RPC（Resistive Plate Chamber） バレル部トリガー用。

TGC（Thin Gap Chamber） エンドキャップ部トリガー用。

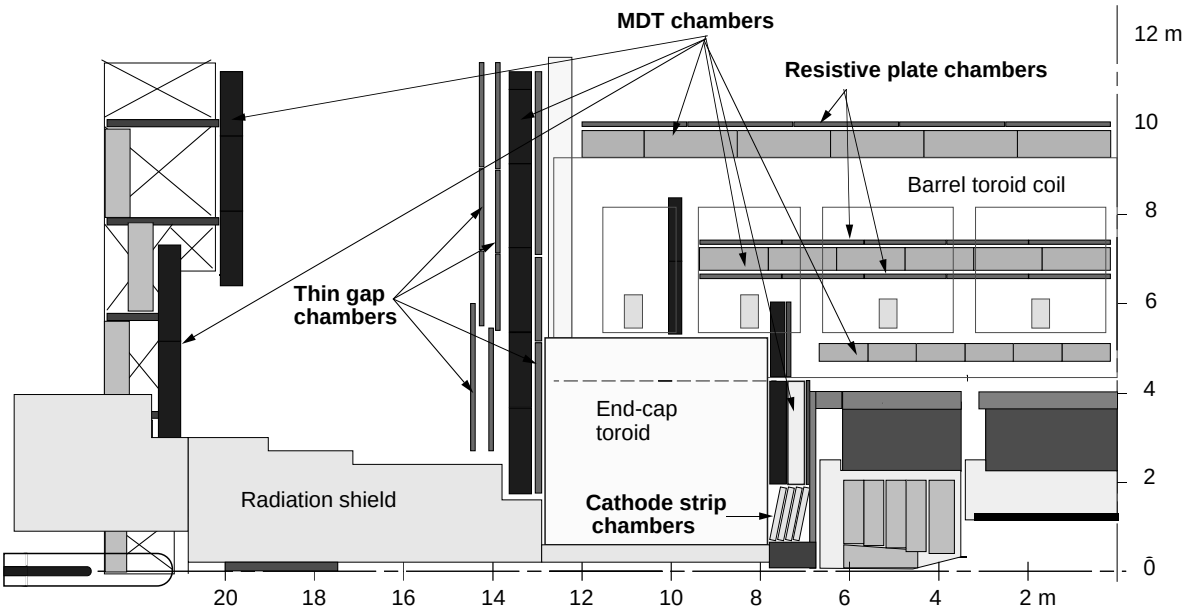

図 1.9： ATLAS のミューオン検出器の R-z 断面図。ATLAS のミューオン検出器は、精密測定用の MDT と CSC、トリガー用の RPC と TGC からなる。

MDT（Monitored Drift Tube）は、図 1.10 に示したとおり、チューブ系 30 mm ワイヤ系 50 μm のドリフトチェンバーを 3 又は 4 層重ねたものを、さらに 2 層に重ねたものである。チューブ内のガスは 3

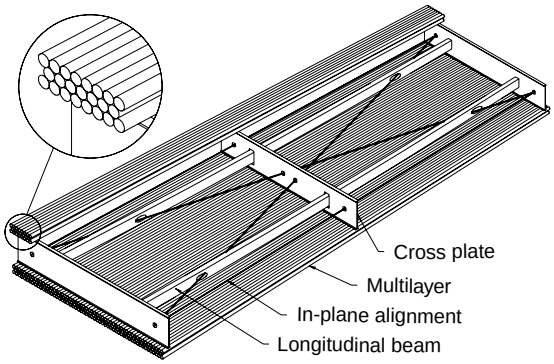

図 1.10： MDT（Monitored Drift Tube）

bar の Ar-CH_4-N_2 混合ガスである。チューブ一本あたりの分解能はおよそ 80 μm、ドリフト時間は最大 500 ns 程度である。

CSC（Cathode Strip Chamber）は、カソードストリップ読み出しの MWPC である。アノードワイア間隔とアノード・カソード間隔はともに 2.54 mm、カソード間隔は 5.08 mm である。位置分解能は 60 μm 程度である。ドリフト時間は 25 ns 以下と短く、時間分解能も 7 ns と良いので、バンチ識別が可能である。CSC は図 1.9 のように、高い放射線に曝される超前方領域の最も内側の部分に用いられる。

1.3.5 ミューオントリガーチェンバー

ミューオントリガーチェンバーの主な働きは以下の通りである。

- ミューオンを検出し、その $p_{\rm T}$ の大まかな値を測定する。6 段階の $p_{\rm T}$ の閾値に応じてトリガー信号を出す。
- バンチの識別。MDT の時間分解能は 500 ns 程度であるから、MDT だけではミューオンがどのバンチに属するかは判定できない。この判定のためにはトリガー用チェンバーの時間情報を用いる。
- 第二座標の読み出し。検出器の平面内において、理想的なトロイダル磁場によって曲がる方向を第一座標、曲がらない向きを第二座標と呼ぶ。従って、第二座標はバレル部分では z 座標、エンドキャップ部分では ϕ 座標のことである。MDT は第二座標は測定しないので、トリガーチェンバーからの情報だけが頼りとなる。

これからわかるように、トリガーチェンバーは、単にトリガー信号を作るだけでなく、他の検出器と同様にデータの読み出しと収集が必要とされている。

これらの目的のため、トリガーチェンバーとして、バレル部分には RPC を、エンドキャップ部分には TGC を置く。これらの検出器は図 1.11 のように各々 3 層からなる。但し、この各層が 2 または 3 層からなっている。同時に、図には運動量測定のスキームも示されている。理想的には ϕ 方向にトロイダル磁場がかかっているので、ミューオンは R-z 平面内で曲がる。この曲がり具合を 3 層のチェンバーで測定することによって運動量を測定することが出来る。現実には磁場の大きさは一定ではないし磁場には R や z 成分もある。そのためミューオンは ϕ にも曲がるので、ϕ 方向のずれも同時に測定しなければならない。そして、最終的にはミューオンの曲がり具合とその軌跡上での磁場の向きと大きさから、運動量の大きさを計算する必要がある。

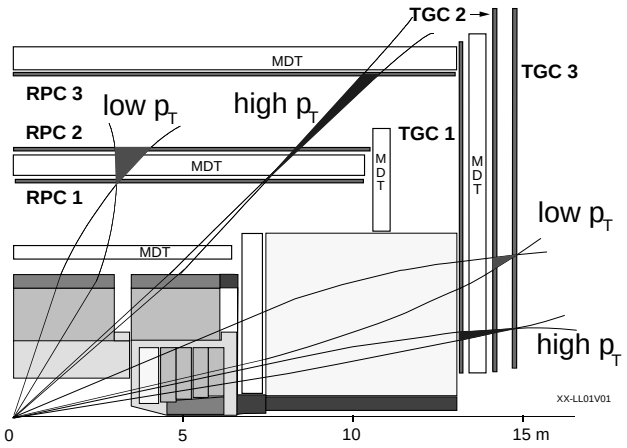

図 1.11： ミューオン検出器と運動量測定。ミューオンはトロイダル磁場によって曲がる。この曲がり方を 3 ヶ所のチェンバーで測定することにより、運動量が測定できる。

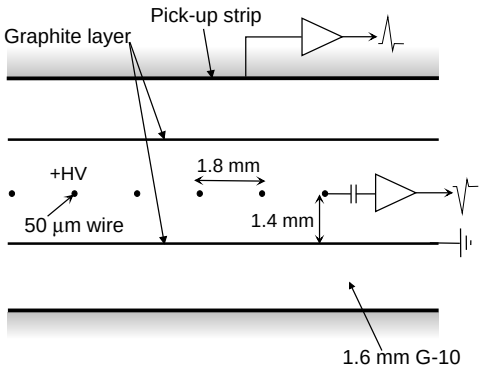

図 1.12： TGC の構造。1.8 mm 間隔のワイヤと、ストリップを用いた 2 次元読み出しが可能である。

RPC（Resistive Plate Chamber）

RPC は $|\eta| < 1$ のバレル部分を覆っているミューオントリガー用のチェンバーである。RPC は 2 枚のベークライト板の間にガスが封入されているものが 2 層重なっている構造をしている。ワイヤはなく、各層には直交するストリップが存在し、2 次元読み出しが可能になっている。ガスのギャップは 2 mm、ストリップ間隔は 30.0 mm から 39.5 mm である。時間分解能は 1.5 ns とよい。

TGC（Thin Gap Chamber）

TGC は $1 < |\eta| < 2.7$ のエンドキャップ部分を覆っているチェンバーである。このうちトリガーには $|\eta| < 2.4$ の部分が用いられる。TGC の構造を図 1.12 に示す。1.8 mm 間隔で設置された 50 μm のアノードワイアと、アノード面から 1.4 mm のところにあるカソードストリップにより 2 次元読み出しが可能になっている。ワイヤ間隔が 1.8 mm と短いことでドリフト時間が短くなり、時間分解能がよくなっている。用いるガスは CO_2-n pentan 55:45 である。このガスはクエンチ能力が高いので、構造のひずみや入射角にあまり依存しないガウス型に近い信号が得られる。また、レートが 20 kHz/cm^2 の状況下でも動作する。時間分解能の点では、25 ns のゲートで 99% 以上の検出効率があるとされている。

第 2 章

ATLAS 実験のトリガーとデータ収集

2.1 ATLAS のトリガー、DAQ

2.1.1 序論

ATLAS 実験では、40 MHz のバンチ交叉ごとに 20 個以上の何らかの事象が起こるとされている。しかし、最終的には 100 Hz 程度でしかデータを記録をできないため、$10^5 \sim 10^6$ の事象選択を行う必要がある。このように、数多くの事象から有用だと思われる事象を選びだし、膨大なチャネルのデータを収集、記録する必要がある。

ATLAS 実験ではヒッグスの探索を初めとする、1.2 で述べたような物理を標的としている。その一方で、LHC 加速器が作り出すエネルギー領域は未知のエネルギー領域であり、そこで見いだされるであろう様々な未知の物理に対しても対応できる必要がある。そのため、ATLAS のトリガーは 1.2 の物理に対して感度があり、かつ包括的なものにする必要がある。

具体的には、p_{T} の大きな e, μ, γ, τ, ジェット、あるいは大きな $E_{\mathrm{T}}^{\mathrm{miss}}$ などは、ヒッグスや超対称性粒子の探索の際に必要な情報であると共に新しい物理を示唆するものかもしれない。このようなものを含む事象については、少なくともトリガーの初期の段階では積極的に採用する。ただし、これだけでは十分にはイベントレートを落とすことが出来ないので、後の段階では、例えばレプトン候補の不変質量や $E_{\mathrm{T}}^{\mathrm{miss}}$ の情報を利用して W や Z がレプトンに崩壊した事象を選び出し、これについて包括的な選択を行う。

但し、特に最初の数年間の低ルミノシティ運転時には、新しい物理の探索用の包括的なトリガーと同時に、B の物理用のトリガーも必要になる。このときには精密測定用の特定の B の崩壊モードを標的にした排他的なトリガーも行うことになる。

2.1.2 トリガーのスキーム

図 2.1 に ATLAS のトリガー DAQ の概要を示す。図のように、ATLAS のトリガーは LVL1（レベル 1)、LVL2（レベル 2)、EF（Event Filter）の 3 段階からなる。

LVL1 では、カロリーメータやミューオン検出器（TGC, RPC）からの granularity の低いデータを用いて、各々のバンチ交叉に対して L1A（レベル 1 アクセプト：レベル 1 のトリガー判定）の決定を行う。この際、内部検出器のデータは用いない。レベル 1 トリガーの latency（バンチ交叉の後、L1A 信号が出されるまでの時間）は 2 μs である。この間、各検出器からのデータはバイプラインメモリに蓄えられる。L1A のレートは最大 75 kHz である。但し、各検出器の読み出し系は 100 kHz までアップグレード可能でなければならないとされている。そのため、DAQ や LVL2 トリガーは 100 kHz の L1A にも耐えられるように設計されている。

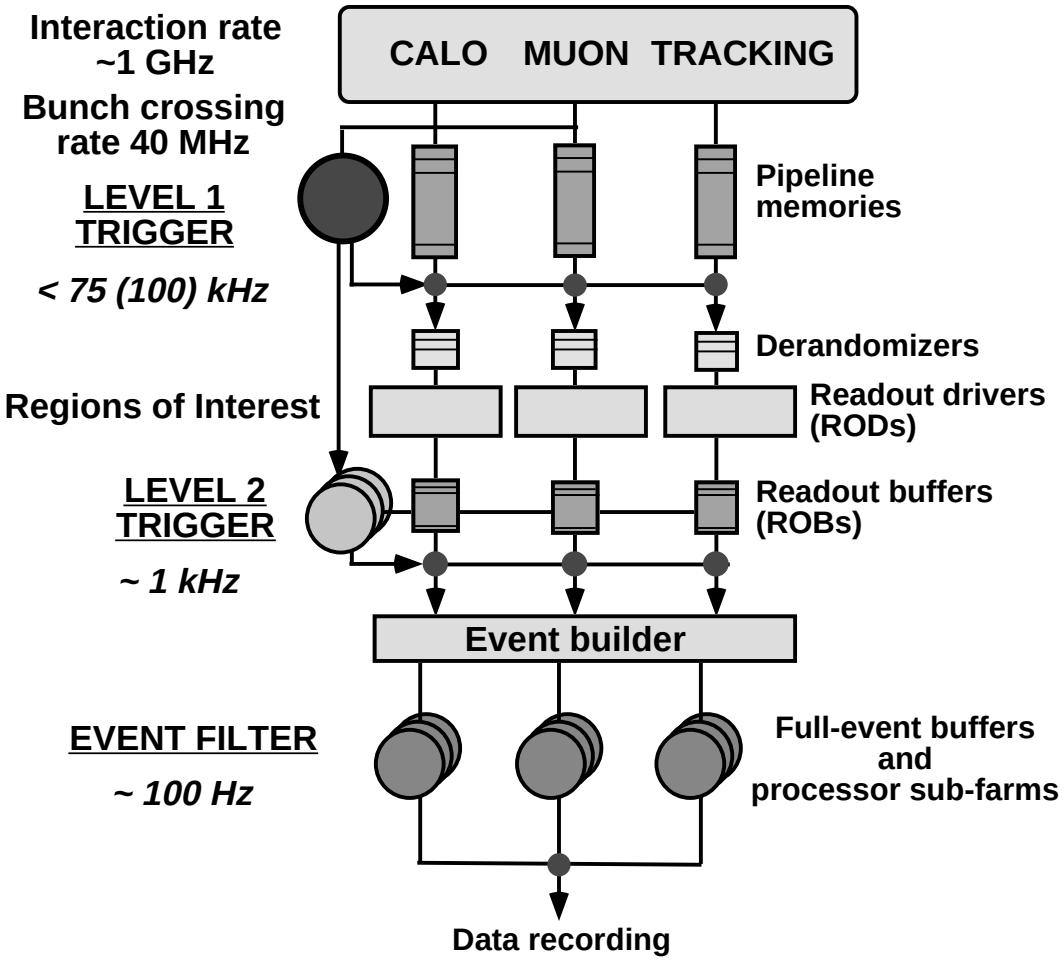

図 2.1： ATLAS のトリガー DAQ の概要。ATLAS では 3 段階のトリガースキームが採用されている。

LVL2 では完全な位置情報と精度を持つデータを用いて決定を行う。ただ、プロセッサーの処理能力とデータの転送能力の関係上、この段階では検出器の全てのデータを用いてトリガー判定を行うわけではない。LVL1 から与えられた情報から、High-p_{T} の電磁クラスタやジェット、ミューオンなどの重要な特徴がある領域 RoI（Region of Interest）を判定し、その領域のデータを用いて判定を行うのである。判定時間は可変で、平均 10 ms が想定されている。LVL 2 トリガーによりイベントレートは 1 kHz にまで減らされる。

LVL2 で採用されたイベントについては、イベントビルダーを経て EF に送られる。EF では完全な位置情報と精度を持つ全データを用いて判定が行われる。判定時間は約 1 s である。記録可能なデータ量は 100 MB/s と想定されており、EF ではこの範囲内になるまでデータ量を減少させなければならない。全ての検出器のデータを記録するとすると ATLAS のデータは 1 イベントあたり 1 MB 程度なので、EF 後のレートは 100 Hz ということになる。しかし、特定のイベントについては一部分のデータのみを記録するという方針をとれば、レートを増やすことも可能である。

2.1.3 DAQ のスキーム

ATLAS の各検出器のフロントエンドの部分でのデータ収集の流れを図 2.2 に示す。検出器からのデータは L1A 信号が TTC（2.2.5 参照）から来るまでの間、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。レベル 1 トリガーの latency は 2 μs であるが、レベル 1 バッファでは将来のケーブルの配置や回路の設計の変更に対応するために、500 ns の余裕をみて 2.5 μs 以上の間データを保持する。LVL1 で採用されたイベントについては、デランダマイザに送られる。デランダマイザは次の ROD（Readout Driver）に読み出されるまでの間データを溜めておくための FIFO である[1] 。デランダマイザの大きさはデータの損失が 1%（L1A が 100 KHz のときは 6%）以下であるようなものでなければならない。

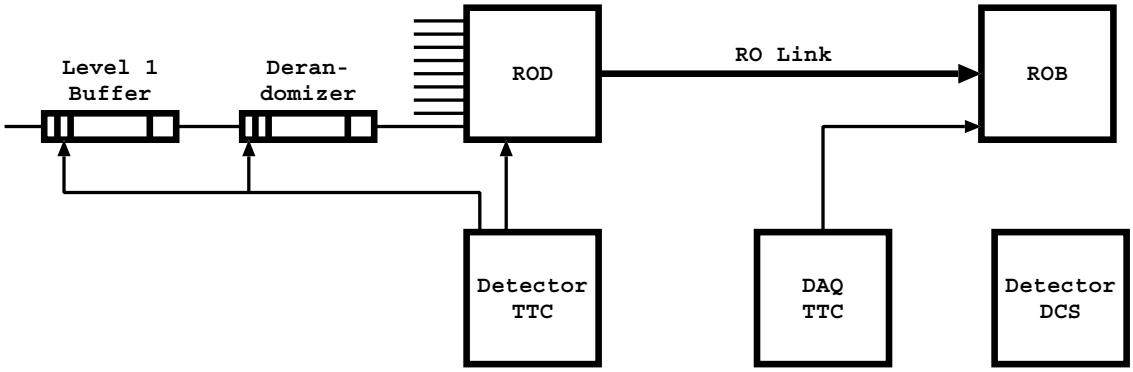

図 2.2： ATLAS 検出器のフロントエンドでの DAQ のスキーム。検出器から読み出されるデータは、ROD を経て ROB に送られる。ROB 以降は ATLAS 全体で共通のシステムが採用される。

デランダマイザに保持されたデータは ROD によって読み出される。この際、データと共に、そのデータがどのバンチに属するかという BCID（バンチ ID）、及びどの L1A 信号に対応するかという L1ID（レベル 1 ID）も送る。ROD はデランダマイザから送られてきたデータの整合性（BCID, L1ID など）を調べて、正しければデータを RO Link（Readout Link）を通じて ROB（Readout Buffer）に送る。表 2.1 に ATLAS 検出器のチャネル数とデータ量などを示す [6]。

LVL2 のトリガー判定では、ROB に保持されたデータのうち RoI のデータのみが利用される。この後は LVL2 で採用されればデータはイベントビルダーへ進み、さらに EF でも採用されればテープ等に記憶される。

DAQ のシステムのうち、ROB 以降は全ての ATLAS 検出器に共通のものとなる。LVL2 や EF は ROB のデータにしかアクセスしないので、これらの間のインターフェースは統一される。一方、ROD よりも検出器側については、各々の検出器に自由度があるので、検出器の特徴や用途に合わせた実装を行うことが可能である。

2.1.4 Detector Control System

ATLAS では検出器の制御と監視を統一的な方法で行うために、DCS（Detector Control System）という呼ばれるシステムが用意される。DCS では検出器の運転や物理解析に必要なパラメータだけでなく、検出器の安全性を確保するために必要なパラメータ（温度など）の監視を行う[2]。その他、換気や冷却などの

[1] デランダマイザ（Derandomizer）は、ランダムで起こるイベントを一定の間隔で読み出されるようにするためのもの、という意味でこのように名付けられている。

[2] 但し、最終的な安全性の確保は検出器側の責任であり、インターロックやハードワイヤなどを用意して検出器が破壊されないようにつとめなければならない。

Detector	Channels count	FE occupancy [%]	FE band-width[a] [Gb/s]	No. RODs
Pixel	1.4×10^8	0.01	50	81
SCT	9,314,304	1.0	224	256
TRT	424,576	20	204	256
EM calorimeter	173,952	100	417	704
Hadron calorimeter	25,714	100	62	120
Muon trigger chambers	789,704	0.12	1	32
Muon precision chambers	431,392	10	103	256
Total			979	1705[b]

[a]assuming 75 kHz LVL1 rate
[b]additional ($\sim$ 16) RODs will be used for LVL1 output data

表 2.1： ATLAS 検出器のチャネル数。同時に occupancy や ROD（Readout Driver） の数も示す。1 つの ROD あたりで 1 Gb/s のデータを転送できる。

状態の監視を行う他、LHC からルミノシティや放射線についての情報を得る。

DCS の構成を図 2.3 に示す。フロントエンドには LMB（Local Monitor Box）と呼ばれるモジュールを配置する。LMB には ADC や DAC が備わっており、センサーからの信号をデジタル化したり制御用のアナログ信号を作り出したり出来るようになっている。LMB は CAN[3] と呼ばれるバスシステムで接続されていて、CAN を通じてデータの読み書きが出来るようになっている。そのため LMB には市販の CAN マイクロコントローラが搭載されている。

2.2 レベル 1 トリガーシステム

2.2.1 トリガー条件

レベル 1 トリガーでは、カロリーメータとミューオントリガーチェンバーからの情報をもとにトリガーの判定を行う。高ルミノシティ運転時の主なトリガー条件と予測されるレートは表 2.2 に、低ルミノシティ運転時のものを表 2.3 に示す。レベル 1 では、基本的には p_{T} の大きなミューオン、エネルギーの大きい電磁クラスタ、ジェット、$E_{\mathrm{T}}^{\mathrm{miss}}$ でトリガーをかける。当然のことながら、1.2 で述べた物理を行うために必要なイベントはレベル 1 では採用される。低ルミノシティ運転時には、B の物理にも重点をおくため、ミューオンによるトリガーのレートが高くなっている。

2.2.2 レベル 1 トリガーエレクトロニクス

図 2.4 にレベル 1 トリガーのブロック図を示す。カロリーメータ側からは e/γ、ジェット、$E_{\mathrm{T}}^{\mathrm{miss}}$ の情報を、ミューオンスペクトロメータ側からは High-p_{T} のミューオンの情報が、それぞれ CTP（Central Trigger Processor）に送られる。CTP ではこれらの情報を用いてレベル 1 のトリガー判定を行い、その結果（L1A）を TTC（Timing, Trigger and Control）に送る。TTC は L1A 信号を各検出器のフロントエンドに分配する。

[3]Controller Area Network の略で、マルチマスターのシリアルバスシステムである。アドレスの概念はなく、送信側は識別子をつけてメッセージをブロードキャストし、受信側がその識別子をみてメッセージを受け取るかどうかを決める。CAN プロトコルは特にヨーロッパで自動車部品の制御などに用いられている。

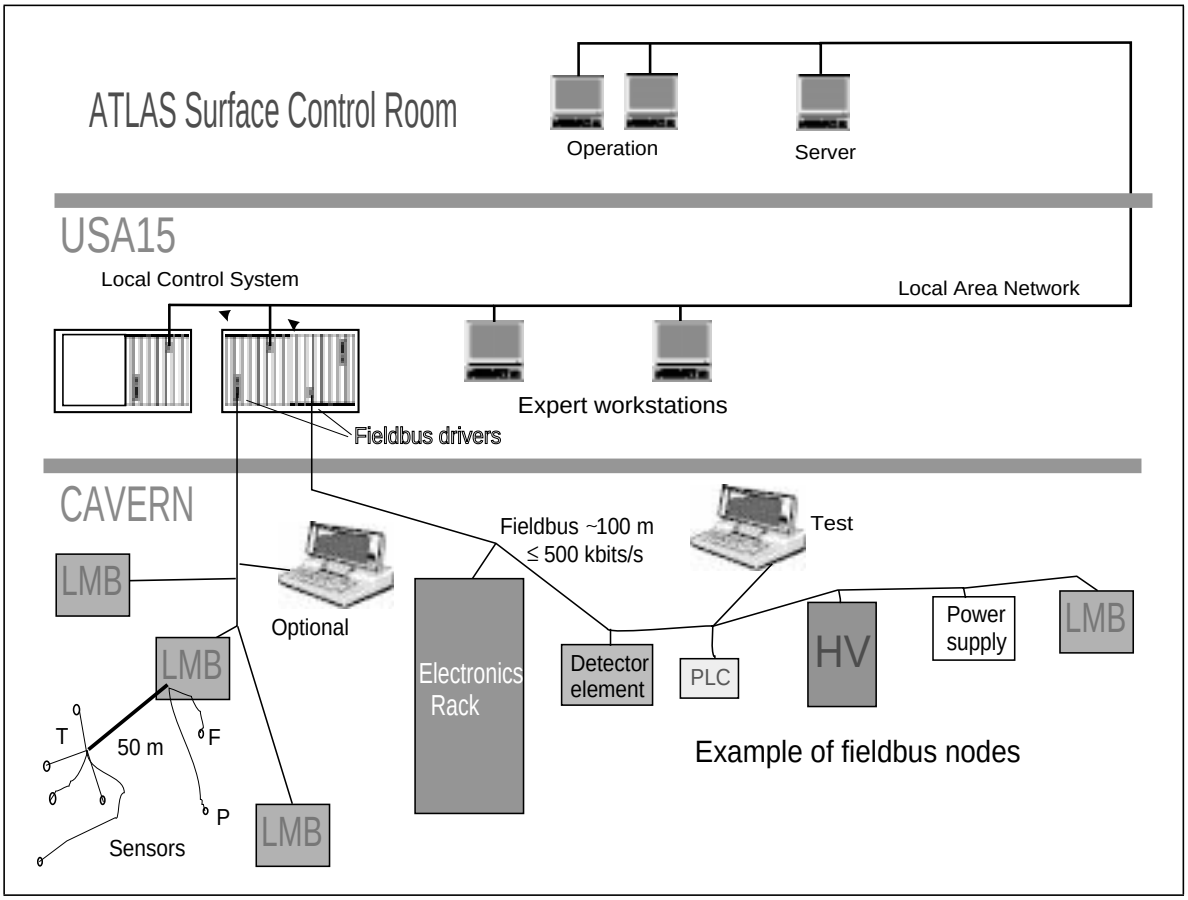

図 2.3： DCS（Detector Control System）の構成

このような機能を実現するためには、専用の処理を行うエレクトロニクスを用いることになるのであるが、これらのエレクトロニクスの設置場所としては、検出器のそばと USA15 と呼ばれるエレクトロニクス室がある。検出器のそばに置かれるエレクトロニクスは放射線や接近方法の関係上、運転中の保守、交換は困難である。だが、検出器から USA15 までは約 80 m あり、信号伝播の関係上、またケーブルの量を減らすためには、検出器のそばにエレクトロニクスを置かざるを得ない。それゆえ、一般的には検出器のそばのエレクトロニクスに信号のデジタル化やレベル 1 バッファ、デランダマイザなど部分を実装し、光ファイバーで USA15 に送信後 USA15 内の回路で複雑な処理を行う、という形になる。しかし、どこまでの回路を検出器のそばに置くのか、いかにケーブルの量を減らせるかということは重要な課題である。

2.2.3 Central Trigger Processor

CTP（Central Trigger Processor）は、カロリーメータとミューオンスペクトロメータからのトリガー情報を総合して、最終的なレベル 1 のトリガー判定を行う部分である（図 2.5 ）。カロリーメータでは e/γ、τ 又はハドロン、ジェットについて、ミューオンスペクトロメータではミューオンについて、それぞれ何段かの p_{T} の閾値が設けられており、閾値を越えたトラック数の情報が送られてくる。また、全エネルギー E_{T} とミッシングエネルギー $E_{\mathrm{T}}^{\mathrm{miss}}$ の値を受け取る。この他、キャリブレーションなどのためにその他の検出器からの入力もある。

Trigger Condition	Symbol	Rate[MHz]
Single Muon, $p_{\mathrm{T}} > 20$ GeV	MU20	4
Pair of Muon, $p_{\mathrm{T}} > 6$ GeV	MU6 × 2	1
Single isolated EM cluster, $E_{\mathrm{T}} > 30$ GeV	EM30I	22
Pair of isolated EM cluster, $E_{\mathrm{T}} > 20$ GeV	EM20I × 2	5
Single jet, $E_{\mathrm{T}} > 290$ GeV	J290	0.2
Three jet, $E_{\mathrm{T}} > 130$ GeV	J130 × 3	0.2
Four jet, $E_{\mathrm{T}} > 90$ GeV	J90 × 4	0.2
Jet, $E_{\mathrm{T}} > 100$ GeV AND missing $E_{\mathrm{T}} > 100$ GeV	J100 + XE100	0.5
Tau, $E_{\mathrm{T}} > 60$ GeV AND missing $E_{\mathrm{T}} > 60$ GeV	T60 + XE60	1
Muon, $E_{\mathrm{T}} > 10$ GeV AND isolated EM cluster, $E_{\mathrm{T}} > 15$ GeV	MU10 + EM15I	0.4
Other triggers		5
Total		**40**

表 2.2： 高ルミノシティ運転時のレベル 1 トリガー条件

Trigger Condition	Symbol	Rate[MHz]
Single Muon, $p_{\mathrm{T}} > 6$ GeV	MU6	23
Single isolated EM cluster, $E_{\mathrm{T}} > 20$ GeV	EM20I	11
Pair of isolated EM cluster, $E_{\mathrm{T}} > 15$ GeV	EM15I × 2	2
Single jet, $E_{\mathrm{T}} > 180$ GeV	J180	0.2
Three jet, $E_{\mathrm{T}} > 75$ GeV	J75 × 3	0.2
Four jet, $E_{\mathrm{T}} > 55$ GeV	J55 × 4	0.2
Jet, $E_{\mathrm{T}} > 50$ GeV AND missing $E_{\mathrm{T}} > 50$ GeV	J50 + XE50	0.4
Tau, $E_{\mathrm{T}} > 20$ GeV AND missing $E_{\mathrm{T}} > 30$ GeV	T20 + XE30	2
Other triggers		5
Total		**40**

表 2.3： 低ルミノシティ運転時のレベル 1 トリガー条件

これらの入力データは、内部で適当なディレイやシフトレジスタを導入することによって同一のバンチに合わせたあと、トリガー判定が行われる。CTP には最高 96 種類のトリガー項目が設定することが可能であり各々の項目に対して別々にプリスケール可能となっている。トリガー判定が完了すると、CTP は TTC に対して L1A（トリガー結果）とトリガーの特徴を表す 8 ビットの情報を送る。

レベル 1 トリガーシステムでは、フロントエンドの読み出しを容易にするために、一度 L1A 信号を出すとそれに続く 4 バンチ（100 ns）の間は L1A を出さないように決められている。また、フロントエンドのバッファが溢れかけた場合にも、L1A を出さないようにしたり、場合によっては優先度の高い項目のみトリガーを行ったりする必要がある。CTP ではこれらの処理も行う。

この他、CTP は TTC に対してトリガー情報を送ると同時に、DAQ やレベル 2 トリガーに対してデータを送信する。これらのデータは、レベル 2 トリガーの RoI を選ぶ基準として用いられたり、トリガー系自体のモニターなどに利用される。

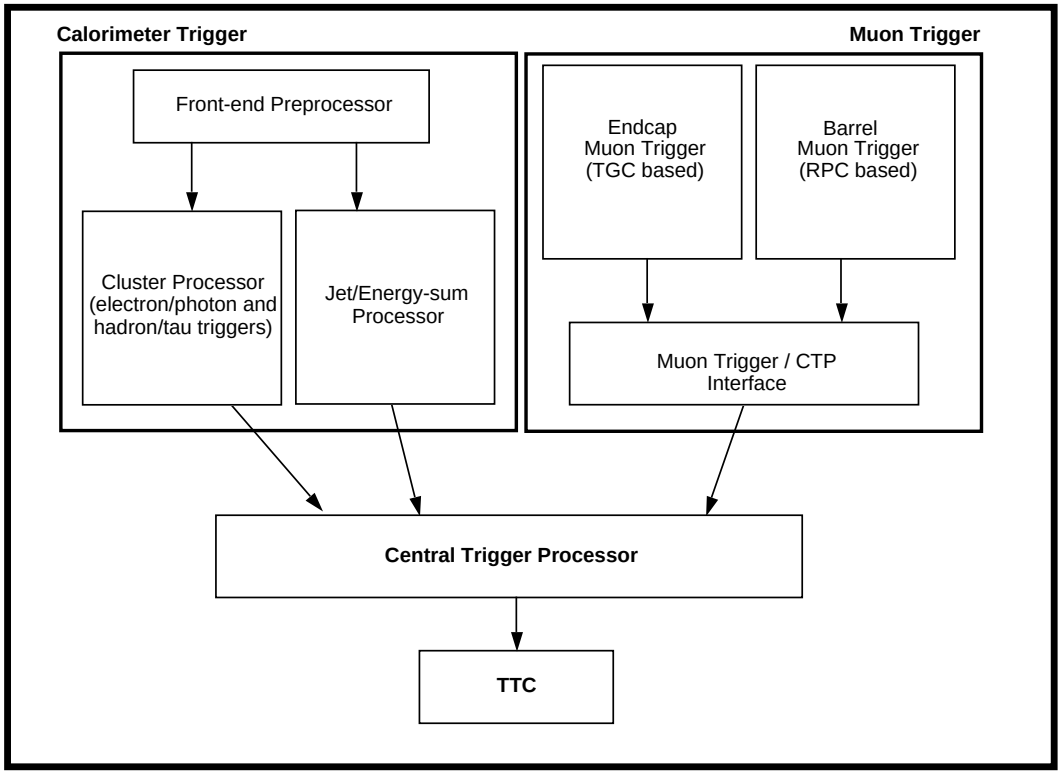

図 2.4： レベル 1 トリガーのブロック図。ミューオンとカロリーメータの情報は CTP で総合されてレベル 1 のトリガー判定が行われる。

2.2.4 MUCTPI

図 2.6 にミューオントリガー系のデータの流れを示す。RPC と TGC は各々セクタと呼ばれる単位ごとに p_{T} の大きなミューオンの候補を挙げる。MUCTPI（Muon Trigger Interface to CTP）はこれらの候補を集め、二重に数えることがないように境界部分での処理を行ってから、最終的なミューオンの候補数を CTP に送る役割を担っている。また、MUCTPI はミューオンのトラックの情報を LVL2 や DAQ に送る。LVL2 に送られたデータは RoI の判定に用いられる。

2.2.5 TTC

TTC（Timing, Trigger and Control）は、ATLAS 検出器のフロントエンドのエレクトロニクスに同期のための信号を分配するためのシステムである。TTC がフロントエンドで分配する信号には、

BC Clock 40.08 MHz の LHC クロック。

L1A Level 1 Accept Signal。レベル 1 のトリガー条件が満たされたことを示す信号。CTP から送られてくる。

BCR Bunch Counter Reset。88.924 s の LHC の軌道周期（ORBIT 信号）に同期した信号で、BCID はこの信号でリセットする。

ECR Event Counter Reset。L1ID はこの信号でリセットする。

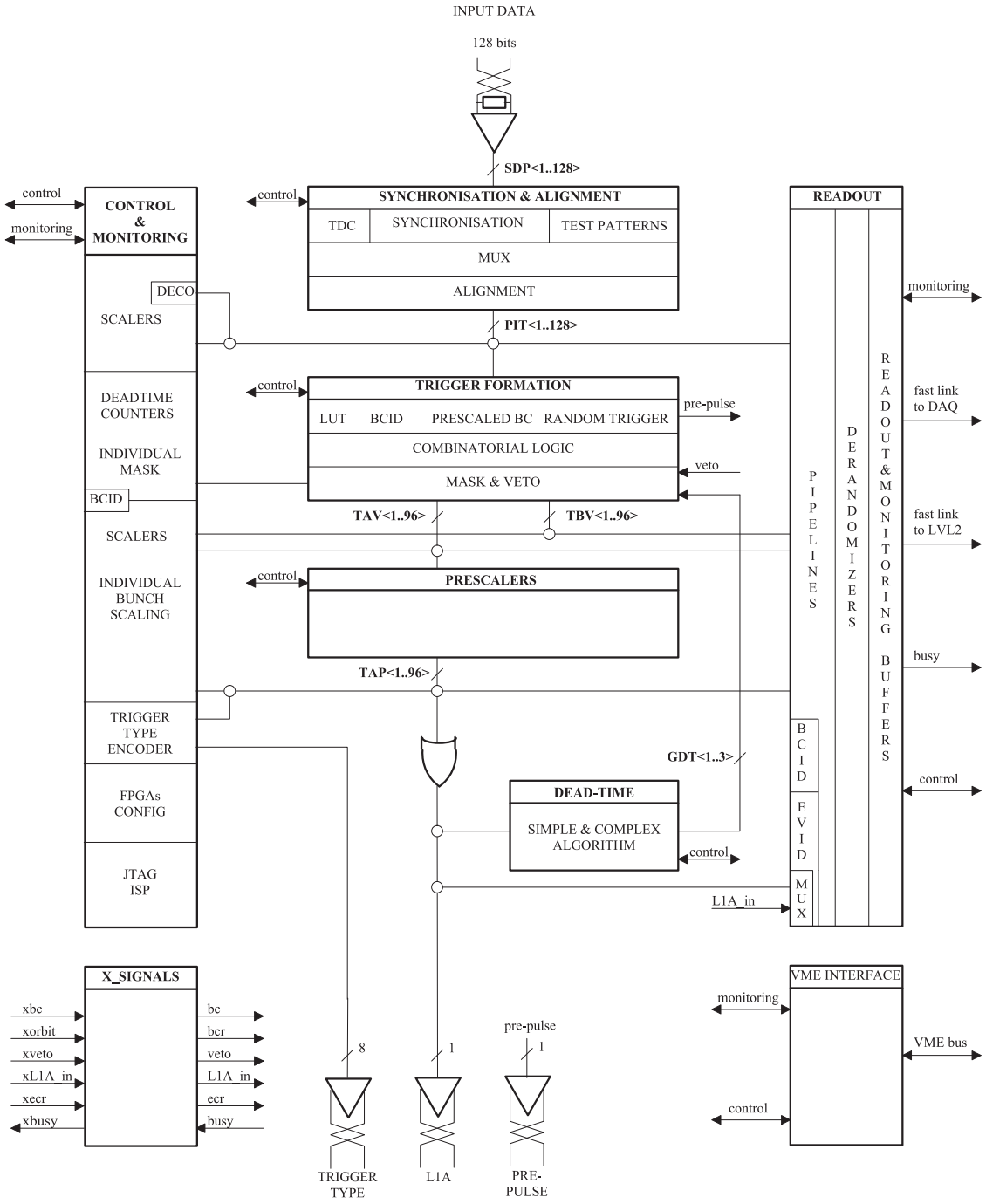

図 2.5： Central Trigger Processor のブロック図

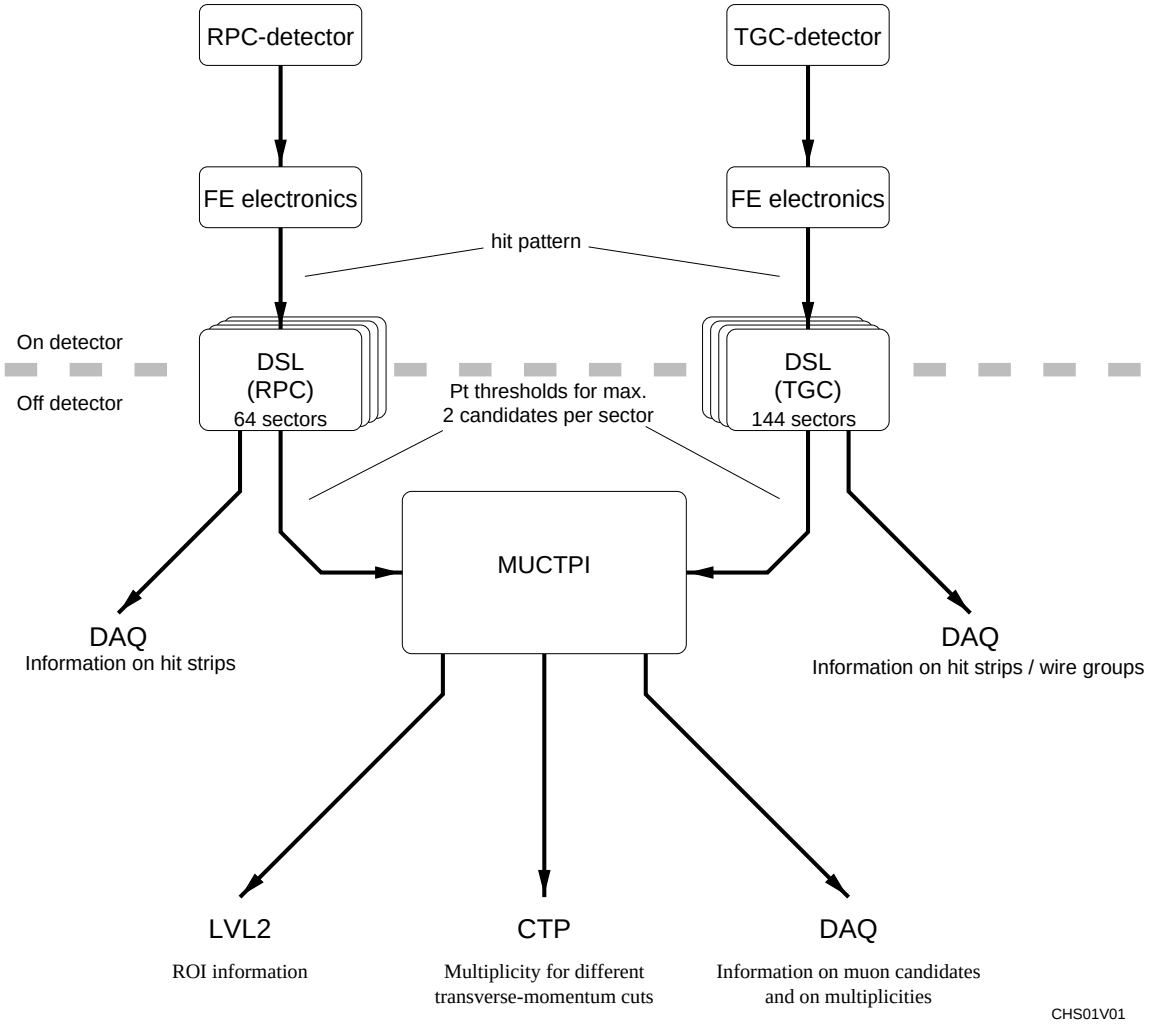

図 2.6： ミューオントリガー系のデータの流れ。MUCTPI は RPC と TGC からトラックの候補を受け取り、境界部分での処理を行ってから、ミューオンの候補についての情報を CTP や LVL2 トリガー、DAQ に送る。

がある。また、TTC は各々の検出器固有のテストやキャリブレーション用のコマンドを受け取り、それを実行する役割を担っている。

TTC は ATLAS 実験全体で見た場合、いくつかのパーティションに分割されている。分割は 1 つのサブシステムあたり数個で、TGC の場合は左右のエンドキャップが各々 1 つのパーティションをなしている。

図 2.7 に TTC のパーティションを示す。TTC のパーティションの中心となるのは、TTCvi と呼ばれる VME とのインターフェースの部分である。TTCvi は LHC から 40 MHz の BC Clock と周期 88.924 s の ORBIT 信号を、CTP からは L1A 信号を受け取る。また、VME 経由でキャリブレーション用のコマンド等を受け取る。これらの情報は TTC クレートに伝えられるのであるが、この際、信号は 25 ns の周期を時間分割して、A-channel B-channel と呼ばれる 2 つの多重化した信号の形で送信される。TTC クレートは TTCvi から受け取った情報を加工した後、ハイパワーの 1310 nm レーザーを用いてフロントエンドにある TTCrx と呼ばれる ASIC まで分配する。TTCrx はこれを解読して、フロントエンドに BC Clock や L1A、ECR、BCR などをフロントエンドに供給する。

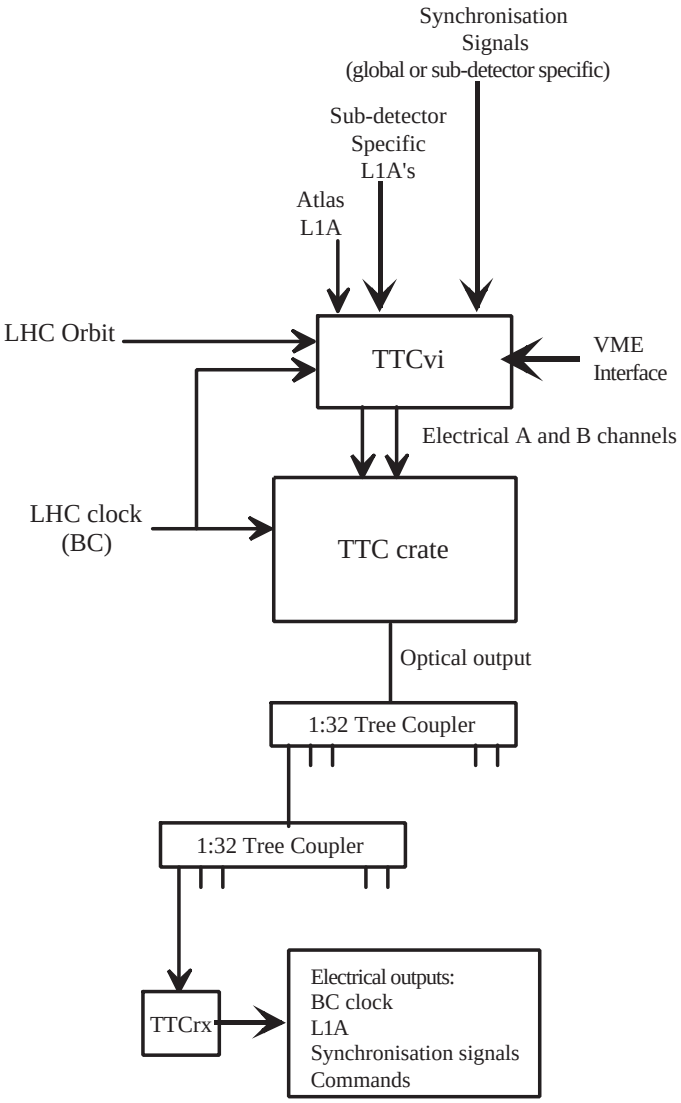

図 2.7： TTC のパーティション。フロントエンドには TTCrx というモジュールが設置される。ここにはレーザーを用いて信号を伝達する。そして、TTCrx がフロントエンドに L1A などタイミングが重要な信号を分配する。

A-channel で扱われるデータは L1A だけであるが、B-channel ではブロードキャストとして TTCrx に送付される同期コマンドと個々のアドレスを指定して TTCrx に送信する非同期コマンドを扱うことが出来る。前者はテストパルスの発生に用いられる。後者はパラメータの設定などに用いることが出来る。

第 3 章

TGC トリガーシステム

3.1 TGC トリガーシステムの概略

3.1.1 TGC トリガーシステムの概略

TGC は ATLAS 検出器のエンドキャップ部分を覆うミューオントリガーチェンバーである。両エンドキャップを合わせて 3600 枚のチェンバーがのべ 6296 m^2 の領域を覆っている。

図 3.1 に TGC の配置の R-z 断面図を示す。図において、M1 はトリプレット（ 3 重層）のチェンバー、M2 と M3 はダブレット（ 2 重層）のチェンバーからなる。M2 はインナーダブレット、M3 はピボットダブレットと呼ばれる。M1, M2 のような集まりをステーションと呼ぶこととする。最も内側の I はインナーステーション TGC でトリガーには補助的に用いられる。TGC は $1 < |\eta| < 2.7$ の領域を覆っているが、このうちトリガー用には $1.05 < |\eta| < 2.4$ の部分が用いられる。このうち $|\eta| < 1.9$ の領域をエンドキャップ領域、$|\eta| > 1.9$ の領域をフォワード領域と呼ぶ。

図中の軌跡が示すように、ミューオンはトロイダル磁場により基本的には R-z 平面内で曲がる。この曲がり具合からミューオンの p_{T} を測定することができる。TGC では 6 段階の p_{T} の閾値を用意することによりトリガーをかける。典型的な p_{T} の閾値は、low-p_{T} ミューオンと呼ばれるものに対しては 6 GeV、high-p_{T} ミューオンと呼ばれるものに対しては 20 GeV であるが、low-p_{T}, high-p_{T} に対して各々 3 種類ずつの p_{T} の設定可能な閾値が用意される。一方で、第二座標（ϕ 座標）測定、及びバンチ識別のための時間情報を与えるため、TGC のヒット情報は他の検出器と同様に読み出される[1]。

TGC については 1.3.5 で少し触れたが、1.8 mm 間隔のワイヤと幅 14.6 − 49.1 mm のストリップで読み出しが可能である。TGC のトリプレットと 2 つのダブレットだけを考えると、ワイヤ（R 座標を与える）は計 7 層、ストリップ（ϕ 座標を与える）はトリプレットにも 2 層しか読み出し面がないので計 6 層で構成されている。R 方向の読み出しは 6 本から 20 本のワイヤをまとめて読みだしている。このワイヤグループはダブレットやトリプレットの各層で粒子から見て互い違いに配置されているので、実効的な位置精度はダブレットでは 2 倍、トリプレットでは 3 倍になっている（図 3.2 ）。ϕ 方向の読み出しは、エンドキャップ領域では 4 mrad、フォワード領域では 8 mrad に相当するストリップパターンによって行われるが、これもワイヤグループと同様に互い違いに配置されており、実効的な位置精度を高めている。

p_{T} の求め方は以下の通りである。図 3.3 を参照してほしい。今、ミューオンがチェンバーを 2 層（layer 1 と 2）通過した。この場合、仮に粒子が無限大の運動量を持っていて直進してきたと仮定した場合の基準となる面（図の layer 2）でのヒット位置から、もう一つの面（layer 1）のヒット位置が求まる。この位置と実際の layer 1 のヒット位置とのずれを δR, $\delta\phi$ と呼ぶこととする。トロイド磁場が理想的な ϕ 方向

[1] これらは RPC でも同様である。

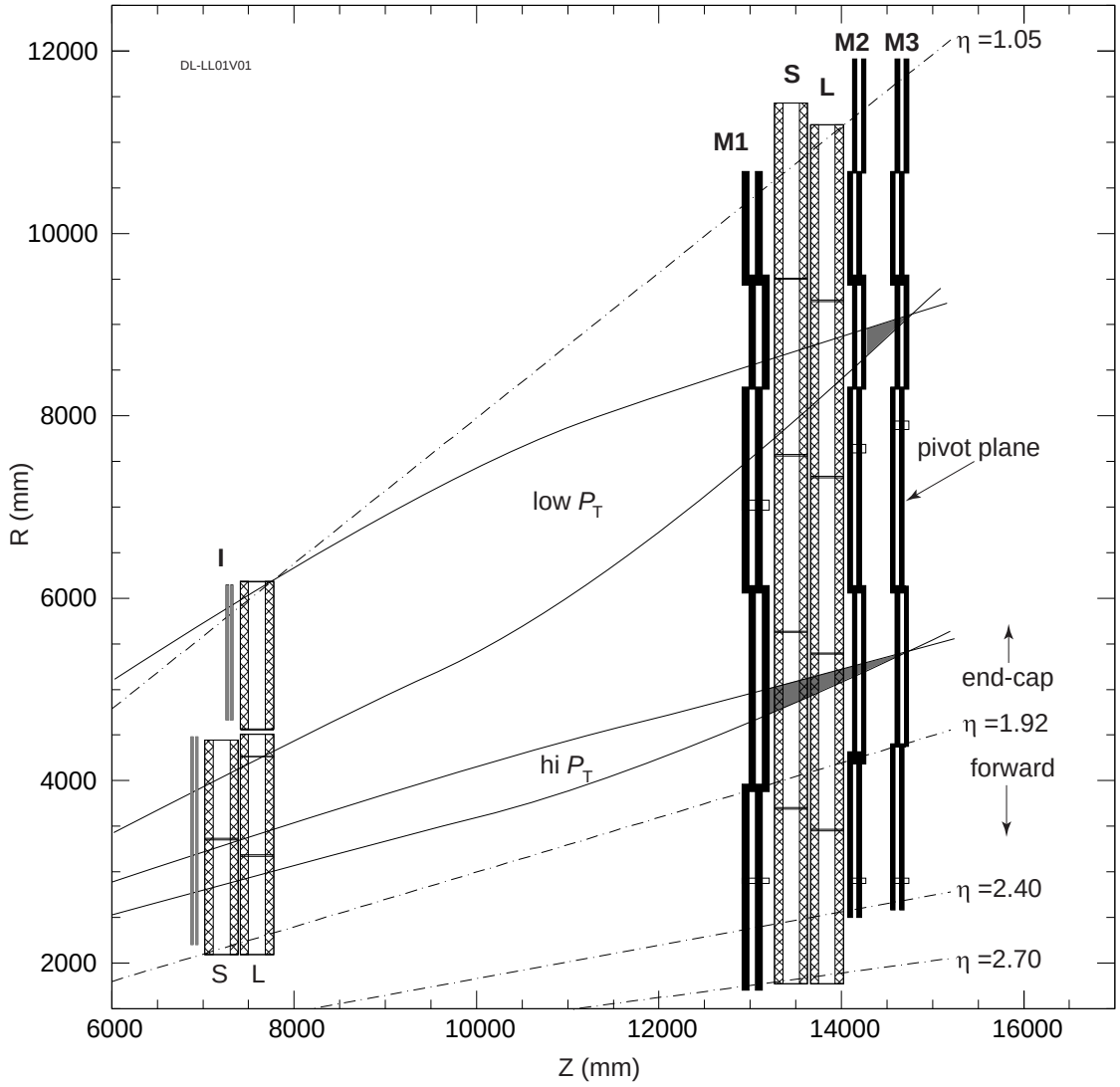

図 3.1： TGC の配置の R-z 断面図。M1 はトリプレット TGC。M2 はインナーダブレット TGC。M3 はピボットダブレット TGC。I はインナー TGC。S と L は MDT。M1, M2, M3 は境界で重なりあうようにするために、微妙に z 方向にずらされて配置されている。そのためこの図では二重に描かれている。

成分のみの磁場である場合には $\delta\phi = 0$ である。しかし、実際には磁場は理想的なものとは異なり一様ではなく、R 成分や ϕ 成分も存在する。そのため、実際にはミューオンは R 方向だけでなく ϕ 方向にも曲がるのである。それゆえ、$p_{\rm T}$ を求めるためには $\delta R, \delta\phi$ を測定すること、磁場分布が正確に知られていることが必要である。TGC では、図 3.3 の layer 2 に相当する基準となる面を最も外側のピボットダブレットに選ぶこととする[2]。そして、トリプレットの 3 層中 2 層（ストリップは 2 層中 1 層）、ダブレットの 4 層中の 3 層で検出されたものをミューオンとして扱い、ピボットダブレットとインナーダブレット、あるいはピボットダブレットとトリプレットとの間で $\delta R, \delta\phi$ を求める。一方、磁場を正確に測定するため、各所には 3 次元磁場測定用のセンサーが取り付けられており、実験中にも常にモニターされている。この磁場の情報と $\delta R, \delta\phi$ から $p_{\rm T}$ が算出される。

[2]というよりも、基準面であるからピボットダブレットと呼ばれている。

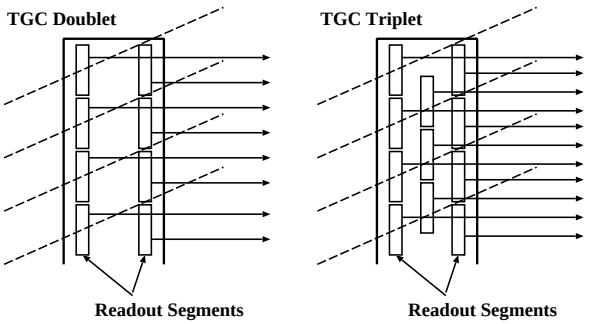

図 3.2： TGC のワイヤグループの配置。点線は反応点から直進してきた粒子。粒子に対してワイヤグループが互い違いに配置されているので、実効的な位置精度は高くなる。

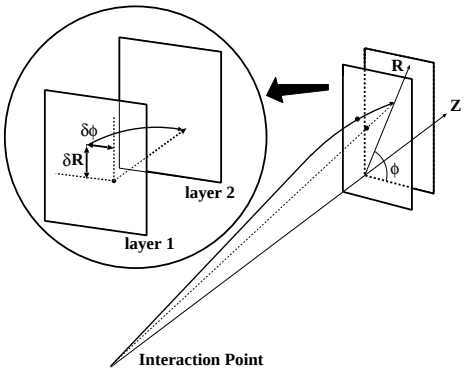

図 3.3： $\delta R, \delta\phi$ の定義。TGC ではピボットダブレットを基準として $\delta R, \delta\phi$ を測定し、p_{T} を算出する。

TGC はオクタントと呼ばれる 8 分の 1 円が一つの大きな単位となっており、チェンバーの建設からデータ収集まで全てこの単位で行われる。レベル 1 トリガーに関連する部分ではオクタントがさらにセクタと呼ばれる単位に分割されている。分割の仕方を図 3.4 に示す。図のようにエンドキャップ領域は 6 つのセクタに、フォワード領域は 3 つのセクタに分割する。TGC 全体では合計 144 のセクタが存在することになる。TGC では各セクタごとに 2 つのトラック候補を選びだし、MUCTPI に 6 段階の p_{T} 値と位置を送信する。

最後に、TGC の境界部分について触れておく。チェンバーの境界はデッドスペースをなくすために互いに重なるように配置されている。そのため、チェンバーの重なりの部分を通過したミューオンは、そのままでは二重に数えられしまう。そこで、r 方向の重なりに対してはワイヤ間で OR をとることにより、ϕ 方向の重なりに対してはストリップをマスクすることで二重に数えることを防ぐという方針をとる[3]。これらの処理を行った後は、TGC は各セクタごとにあたかも 1 枚の大きなチェンバーであるかのように扱うことが可能である。

3.1.2 TGC のパフォーマンス

トリガーの効率

ミューオンのトリガーは low-p_{T} トリガーと high-p_{T} トリガーに大別される。大まかにいって、low-p_{T} トリガーは 6 GeV 以上、high-p_{T} トリガーは 20 GeV 以上のミューオンのトリガーを行う。

low-p_{T} のトリガーでは、基本的には 2 つのダブレットの 4 層のうち 3 層以上のヒットを要求する（図 3.1 の low-p_{T} の軌跡を参照）。そして、2 つのダブレットでのヒットがある一定の幅のコインシデンスウィンドウに入っていたときに、low-p_{T} ミューオンの候補として扱う。4 分の 3 のトリガー条件を課すことにより、バックグラウンドによる偶発的なトリガーを抑えつつチェンバーの不感領域に起因する検出効率の低下を抑えることが出来る。

high-p_{T} のトリガーの場合には low-p_{T} の条件に加えて、トリプレットの 3 層のうちの 2 層のヒットが要求される（図 3.1 の hi-p_{T} の軌跡を参照）。そして low-p_{T} の場合と同様に、これらの 3 つのステーション（2 または 3 層のチェンバーの組）の間でヒットの位置が一定のコインシデンスウィンドウ内に収まっていたときにミューオンの候補として扱う。

[3] 一方で正しく処理を行えば、原理的には二重にトラックを数えることはない。

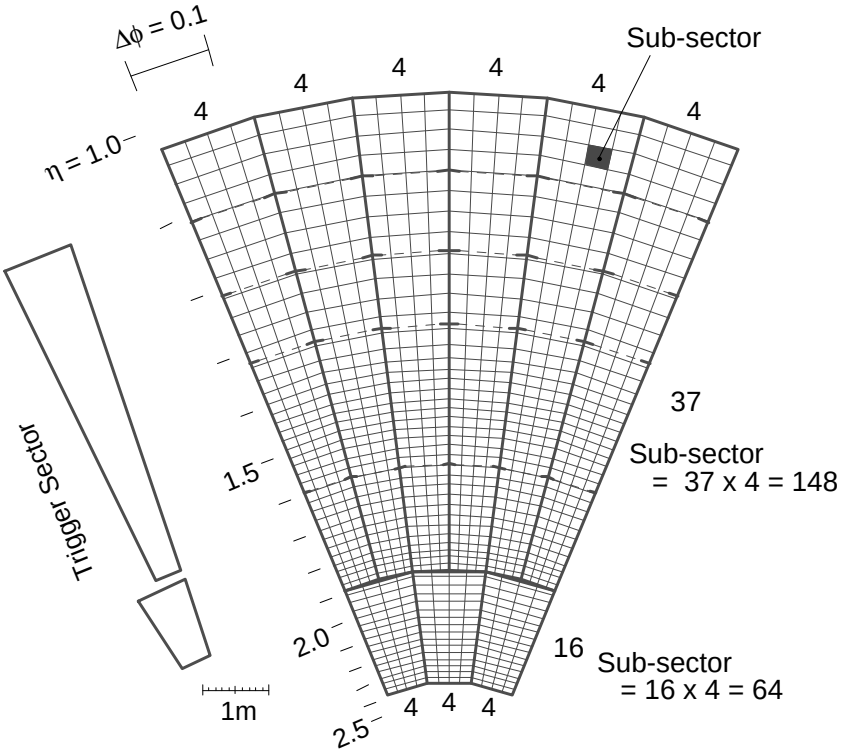

図 3.4： TGC オクタントのレベル 1 トリガー用の分割。TGC のオクタント（8 分の 1 円）は、エンドキャップ部分は 6 個、フォワード部分は 3 個の合計 9 個のセクタに分割される。

TGC ではコインシデンスウィンドウを 90 % 以上の閾値以上の粒子が検出できる領域として定義している。シミュレーションの結果、コインシデンスウィンドウの大きさは high-p_{T} low-p_{T} とも R 方向には 6 cm 程度であることが分かっている。これより TGC には 7 mm 程度の位置分解能が必要となる。それゆえ、1 つのワイヤグループは、ワイヤは互い違いに配置されていることによって実効的な位置分解能はそれぞれ 2 倍、3 倍になっていることを考慮して、ダブレットの場合には 14 mm、トリプレットの場合には 21 mm を目安としている。ワイヤグループに関しては、当初は $-15 \leqq |\delta R| < 15$ の 31 個のワイヤグループの間でコインシデンスを考えていたが、現在は $-20 \leqq |\delta R| \leqq 20$ に拡張されている。ストリップに関しては $-5 \leqq |\delta\phi| \leqq 5$ の範囲でコインシデンスを取っている。

以上のような条件のもとで TGC のトリガーの効率がシミュレーションされている。その結果を図 3.5 に示す。図は low-p_{T} または high-p_{T} のトリガー条件を用いた場合の、いくつかの p_{T} の閾値に対するトリガーの効率である。

ミューオンのヒットレートとバックグラウンド

表 3.1 に TGC でトリガーされるミューオンのレートを示す [3]。これらは L1A を発生すべき信号である。これに対して、以下のようなバックグラウンドが予測されている。

宇宙線ミューオン ATLAS 検出器は地下 75 m に位置するため、大方の宇宙線ミューオンは遮蔽されてしまうが、検出器に通じる通路からは宇宙線ミューオンが侵入してくる。宇宙線ミューオンに対するト

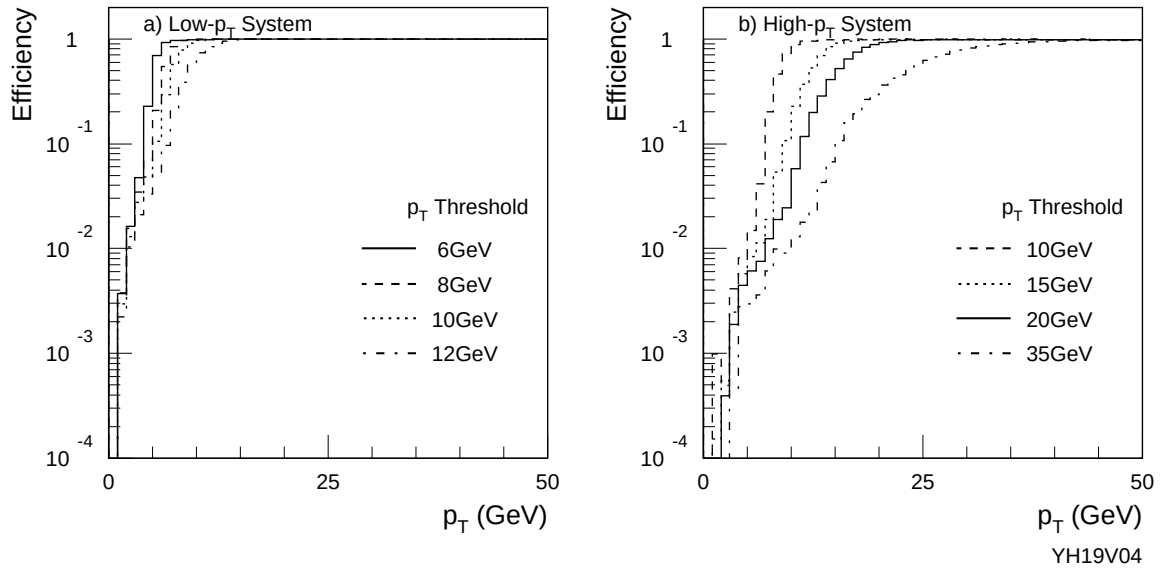

図 3.5： TGC トリガー効率。(a) low-p_{T} (b) high-p_{T} のトリガー条件を用いた場合の、いくつかの p_{T} の閾値に対する効率が記されている。

Process	Low p_{T} muons (6 GeV) in low luminosity	High p_{T} muons (20 GeV) in high luminosity
π/K decays	9.8	1.8
b	2.1	0.7
c	1.3	0.3
W	0.005	0.041
Total	13.2	2.8

表 3.1： 予想される TGC でのトリガーレート。単位は kHz。

リガーレートは、low-p_{T} で 200 Hz 程度、high-p_{T} で 10 Hz 以下と予測されている。これらは検出器のキャリブレーションに用いられる。

ビームハローミューオン 超前方領域では、加速された陽子がビームパイプ中のガスと衝突して生成されたミューオンが多く飛来する。これらは超前方領域でのトリガーを考えるためには考慮しなければならない。今のところ、このようなビームハローミューオンのレートは low-p_{T} high-p_{T} でそれぞれ 250 Hz、16 Hz と予測されており、無視できるとされている。

TGC を 1 枚だけヒットするバックグラウンド TGC を 1 枚だけヒットするバックグラウンドとしては、コンプトン効果で生成される 2 MeV 以下の電子や、中性子と TGC の間の相互作用で生成される低エネルギーの陽子にが考えられる。このようなバックグラウンドのレートは 4.5 Hz/cm^2 と予想されている。

TGC を 2 枚ヒットするバックグラウンド 隣り合う 2 枚の TGC をヒットするバックグラウンドとしては、コンプトン効果で生成される 2 MeV 以上の電子や、低エネルギーの陽子が考えられる。このようなバックグラウンドのレートは 3.1 Hz/cm^2 と予想されている。

TGC を 3 枚以上ヒットするバックグラウンド TGC を 3 枚の以上ヒットするバックグラウンドとしては、高エネルギーの陽子、パイオン、100 MeV 程度のミューオンが考えられる。このようなバックグラウンドのレートは 3.0 Hz/cm^2 と予想されている。

TGC を 1 枚または 2 枚ヒットするバックグラウンドが偶然 2 つのダブレットで起こったような場合にはトリガーがかかってしまうが、このようなレートはさほど大きくない。むしろ問題となるのは、TGC を 3 枚以上ヒットするバックグラウンドである。特に 100 MeV 程度のミューオンによるレートはかなり高いと懸念されている。これに対処するために、low-$p_{\rm T}$ ミューオンについても high-$p_{\rm T}$ の場合と同様のヒットを要求したり、補助的にインナーチェンバーの情報をトリガーに利用するという方針である。この他、CSC からの情報もトリガーに用いることも検討されたこともある。それぞれの場合に、予想されるレートを表 3.2 に示す [7]。

	low-$p_{\rm T}$(6 GeV) rates [kHz]	high-$p_{\rm T}$(20 GeV) rates [kHz]
TDR scheme	7.8	6.4
Three station logic	1.8	6.4
coincidence with EI/FI	0.05	0.02

表 3.2： トリガー条件をかえたときの 100 MeV ミューオンから予想されるトリガーレート。トリガー条件は、TDR のスキーム（low-$p_{\rm T}$ はトリプレットの情報を使わない）、Three station（low-$p_{\rm T}$ でもトリプレットの情報を使う）、EI/FI とのコインシデンス（インナーチェンバーを補助的にトリガーに利用）を想定している。

3.2 TGC エレクトロニクス

3.2.1 TGC エレクトロニクスの概略

図 3.6 に TGC のレベル 1 トリガーエレクトロニクスの概略を示す。TGC からの信号は ASD（Amplifier Shaper Discriminator）Board でデジタル化され、Patch Panel へ送られる。Patch Panel では、各々の信号が相対的にどの LHC クロックに属するかというバンチ識別が行われる。これ以降、信号は LHC クロックに同期したものとして扱われ、処理する電気回路も LHC クロックに同期したものとして設計される。

まず、最初の段階ではワイヤとストリップの情報は別々に扱われる。Slave Board では、ダブレットとストリップで別々にコインシデンスがとられる。図に示されているように、ダブレット用の Slave Board では 4 分の 3 のコインシデンス（4 層中 3 層にヒットがあるか）が、トリプレット用の Slave Board では 3 分の 2 のコインシデンスがとられる[4]。コインシデンス行列からの出力は、ヒット位置にエンコードされて High-$p_{\rm T}$ Board に送られる。ダブレット用の Slave Board の場合には δR や $\delta\phi$ の情報も同時に送られる。High-$p_{\rm T}$ Board では、Slave Board からの情報をもとにワイヤ及びストリップごとにコインシデンスがとられる。

High-$p_{\rm T}$ Board からの $R, \delta R$ と $\phi, \delta\phi$ の情報は Sector Logic に送られる。ここで初めて R-ϕ コインシデンスがとられ、ミューオン候補に対する $R, \delta R, \phi, \delta\phi$ が決定される。この情報をもとに 6 段階で $p_{\rm T}$ が決定される。Sector Logic は最終的に $p_{\rm T}$ が大きな 2 つのトラックを選び出し、その情報を MUCTPI に送る。

[4] ストリップには 2 層分しか読み出しがないので、2 分の 1 のコインシデンスすなわち単なる OR がとられる。

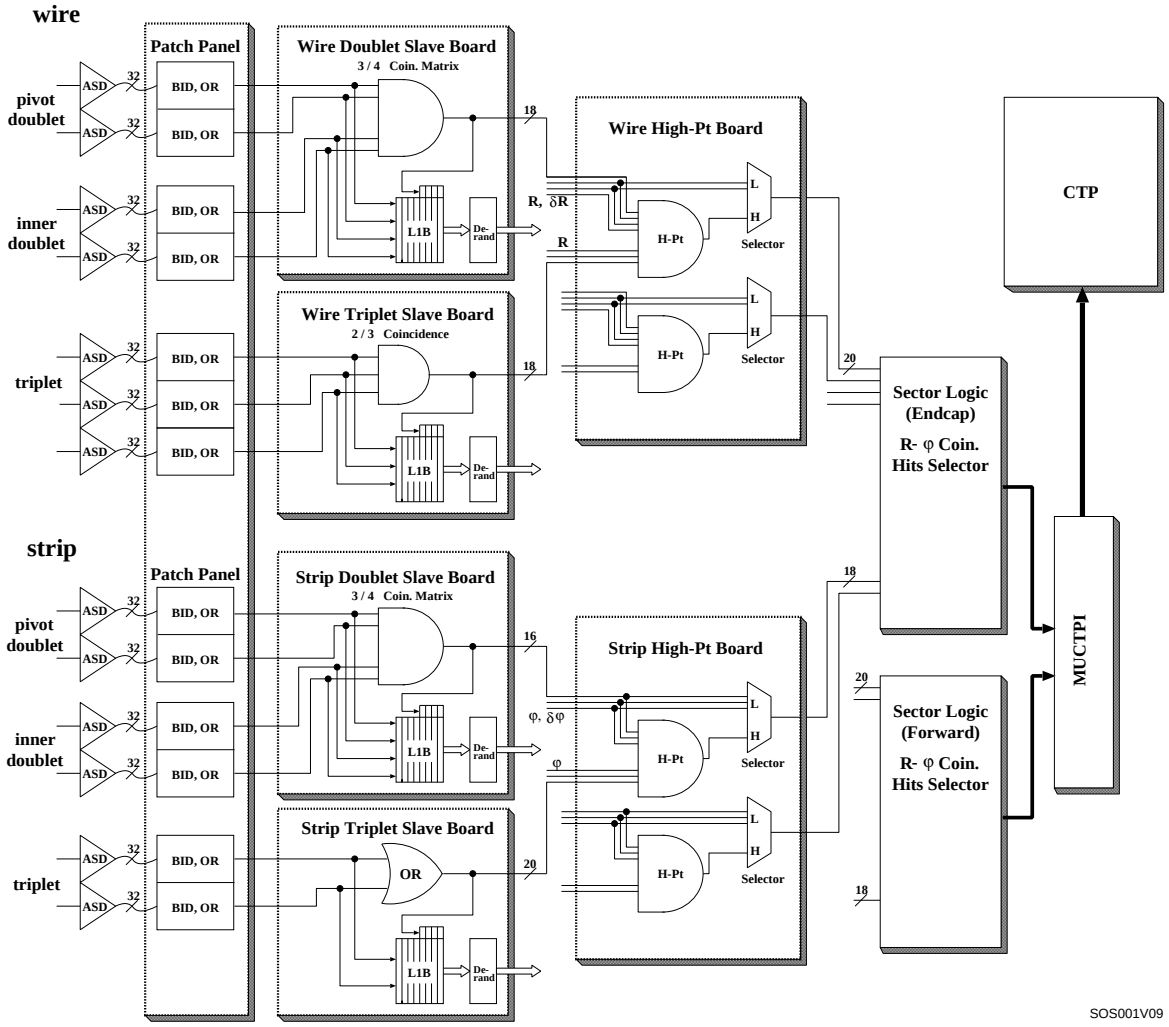

図 3.6： TGC レベル 1 トリガーエレクトロニクスの概略。High-p_{T} ボードまではワイヤとストリップの情報は別々に扱われる。

TGC のレベル 1 トリガーの流れを図 3.7 にのせておく。ダブレット用の Slave Board からはボード（32 チャネル）あたり 2 つのトラック候補が選ばれ、High-p_{T} Board に送られる。次の High-p_{T} Board では 6 つのダブレット Slave Board からの 384 チャネル分（ワイヤが互い違いに配置されていることにより、実効的に 2 倍のチャネル数になっている）とそれに対応するトリプレット用 Slave Board からの情報を合わせて、Board あたり 4 つのトラックの候補を選び出す。Sector Logic では R-ϕ コインシデンスをとったあと最終的にセクタあたり 2 つのミューオン候補を選び出す。この情報は MUCTPI に送られる。

なお、Slave Board に到達したデータはトリガーとは別経路で読み出しが行われる。基本的には Slave Board から読みだされたデータは一旦 Star Switch とよばれるモジュールに集められた後、USA15 内の Local DAQ Master を経て Readout Driver に至る。これに関しては第 4 章で述べる。

表 3.3 に TGC エレクトロニクスの総チャネル数と必要なモジュールの数を示す。総チャネル数は 320 k と極めて多い。数が多い ASD Board や Patch Panel、Slave Board、High-p_{T} Board は ASIC で開発する。それに対して、必要数が 200 個程度の Sector Logic や Star Switch などは、柔軟性の高い FPGA を用いて開発する予定である。

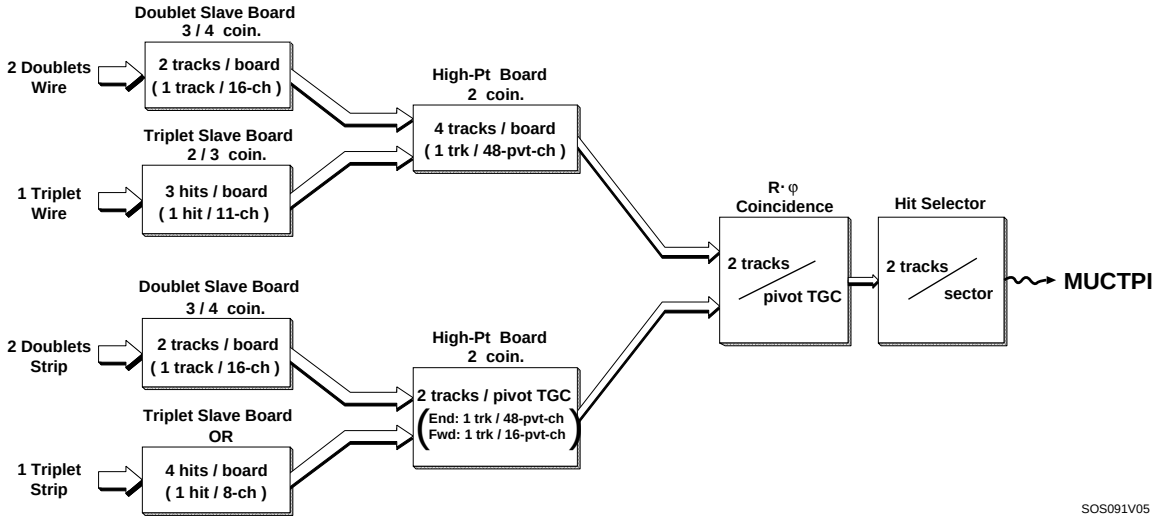

図 3.7： TGC レベル 1 トリガーでのデータの流れ

Item	Number	Details
Channel	321,824	Wire: 220,448　Strip: 101,376
ASD Chip	80,928	
ASD Board	20,784	
Patch Panel	1,600	Wire: 1,056　Strip: 480　Inner: 64
Slave Board	2,880	Wire: 2,016　Strip: 768　Inner: 96
High-p_{T} Board	480	Wire: 288　Strip: 192
Sector Logic	144	
Star Switch	208	
Local DAQ Master	208	
Readout Driver	16	

表 3.3： TGC エレクトロニクスに必要なモジュールの数。TGC の総チャネル数も同時に示す。

3.2.2　モジュールの配置

エレクトロニクスの設置場所は実験室内と USA15 エレクトロニクス室に大別できる。既に述べたように実験室内は放射線の影響を受ける上、保守などのために接近するのが困難である。

TGC エレクトロニクスの主な設置場所を図 3.8 に示す。図には示されていないが、ASD ボードはチェンバーに直接設置される。Patch Panel と Slave Board はチェンバーを支えるホイールのそばにおかれる。それに対して High-p_{T} Board や Star Switch はホイールの最も外側に設置される。ここは実験室内では比較的アクセスのしやすい場所であり、長期の運転停止を行っている時期以外でも保守は可能であるが、実験中には近付くことはできない。それに対して、Sector Logic などは USA15 内に設置されるので、放射線の影響は無くアクセスも可能である。

図において、Slave Board から High-p_{T} Board までのケーブルの長さは 11 m、Star Switch までは 9 m である。これらは典型的な値であるので、実際にはこれよりも若干長いケーブルが必要になるかも知れな

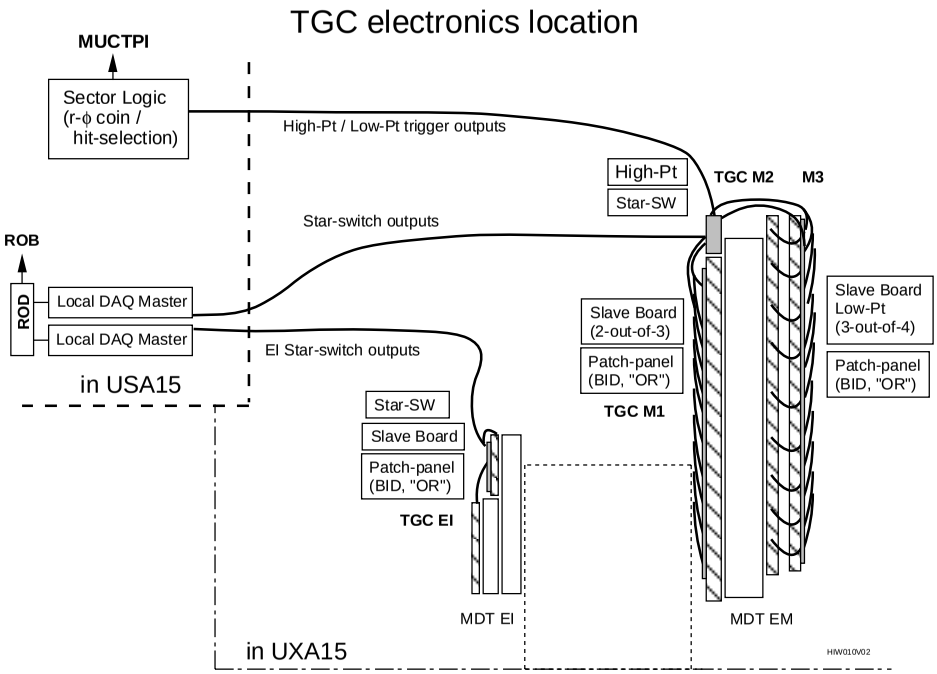

図 3.8： TGC エレクトロニクスの主な設置場所。ASD ボードはチェンバーに直接設置される。

い。一方、High-p_{T} Board などが設置されているホイールの縁から USA15 までは、ケーブルで 80 m の距離がある。そのため、ここは光ケーブルを用いてデータを送信する。

3.2.3 latency

latency は ATLAS のレベル 1 トリガーの設計上、重要なパラメータである。レベル 1 の latency とは、ビーム軸上で p-p 衝突が起こった瞬間から L1A 信号が検出器のフロントエンドシステムに分配されるまでの時間を指す。ATLAS のトリガーのスキームでは、レベル 1 の latency は 2.5 μs であるが、このうち 0.5 μs は今後のアップグレードなどに備えた予備であって、レベル 1 のトリガーシステムに割り当てられているのは 2.0 μs である。

latency の中には、回路内での処理時間は勿論のこと、反応点から検出器までの粒子の飛行時間（TOF）や信号の伝播時間も含まれる。特に信号の伝播時間は大きく、検出器から USA15 までの 80 m を伝わるのに要する時間はおよそ 0.5 μs である。信号が USA15 に送信された後、トリガー判定の結果（L1A 信号）は再びフロントエンドに分配される。その結果、トリガー用の時間の半分は伝播遅延で使われてしまうことになる。

レベル 1 のスキームでは、2.0 μs のレベル 1 トリガー latency のうち 1.15 μs（46 バンチ）が TGC に割り当てられている。すなわち、イベント発生後 1.15 μs 以内に MUCTPI にミューオンの候補を送らなければならないことになる。各モジュールでの latency は表 3.4 のように割り当てられている。表を見ても分かるように、多くの部分がケーブルでの伝播遅延で消費されてしまっている。さらに、各ボード上では基本的には入力信号を一旦ラッチしたり、信号を光ファイバーで送信するためにシリアル化されるのにもクロックを消費してしまうことを考えると、実質的には各々のボードでの処理は 1 から数クロックで行う必要がある。

Module/Transmission	Latency	Total
TOF to TGC	3	3
TGC response	1	4
ASD Board	1	5
Cable to Patch Panel	1	6
Patch Panel	2	8
Cable to Slave Board	1	9
Slave Board	4	13
Cable to High-p_{T} Board	3	16
High-p_{T} Board	5	21
Cable to Sector Logic (80 m)	16	37
Sector Logic	8	45
Cable to MUCTPI (5 m)	1	46

表 3.4： TGC LVL1 トリガーシステムの latency

3.3 TGC トリガーエレクトロニクスの構成要素

3.3.1 ASD Board

ASD（Amplifier Shaper Discriminator）Board には TGC 読み出し用の 4 チャネル ASD チップが 4 つ搭載されており、ボード 1 枚で 16 チャネル分の TGC 信号を処理できる。TGC からの信号は、ASD チップ内のアンプやコンパレータで読み出しとデジタル化が行われ、LVDS レベルに変換されて Patch Panel に送られる。デジタル化の際の threshold 電圧（V_{th}）は Patch Panel から受け取る。その他、Patch Panel からテストパルス信号を受け取り、テストパルスを発生する。テストパルスは時間較正などに利用される。ASD Board はチェンバー本体に直接取り付けられる。

3.3.2 Patch Panel

Patch Panel は ASD Board と Slave Board のバックプレーンのような働きをするモジュールである。Patch Panel の機能を図示すると図 3.9 のようになる。これらの機能の大半は Patch Panel 上に実装される ASIC で実現される。

まず、Patch Panel は ASD Board からの LVDS 信号を受け取り、40 MHz の LHC クロックに同期した信号への変換、すなわちバンチの識別（BCID）を行う。この信号は OR ロジックを経て Slave Board に送信される。すでに述べたように、R 方向にチェンバーが重なっている領域での処理として、異なるチェンバーのワイヤグループからの信号を OR をとることによってミューオン二重に数えることを防ぐ。OR ロジックとはこの処理をする部分のことである。この他、ASD Board には ±3 V の電源、threshold 電圧（V_{th}）[5]、テストパルスを、Slave Board には 3.3 V の電源、LHC クロック、L1A, BCR, ECR といった TTC 信号などを供給する。

チェンバーの境界には規則性はないので、OR ロジックを必要とする部分はボードによって異なる。また、1 つの Slave Board への出力がチェンバーをまたいでいる場合もあるため、Patch Panel 間に信号の行き来が必要である。TGC ではチェンバーの種類をできるだけ少なくするため、$|\eta| = const.$ の直線に対し

[5]threshold 電圧は Patch Panel ASIC からではなく、Service Patch Panel 上の LMB（2.1.4 参照）から供給される予定である。

て投影したような分割はされていない。一方、Slave Board 以降の回路はチェンバーの境界を意識しないで済むようになっている。その影響が Patch Panel にきており、19 種類のボードを作る必要がある。Patch Panel と呼ばれる所以である。

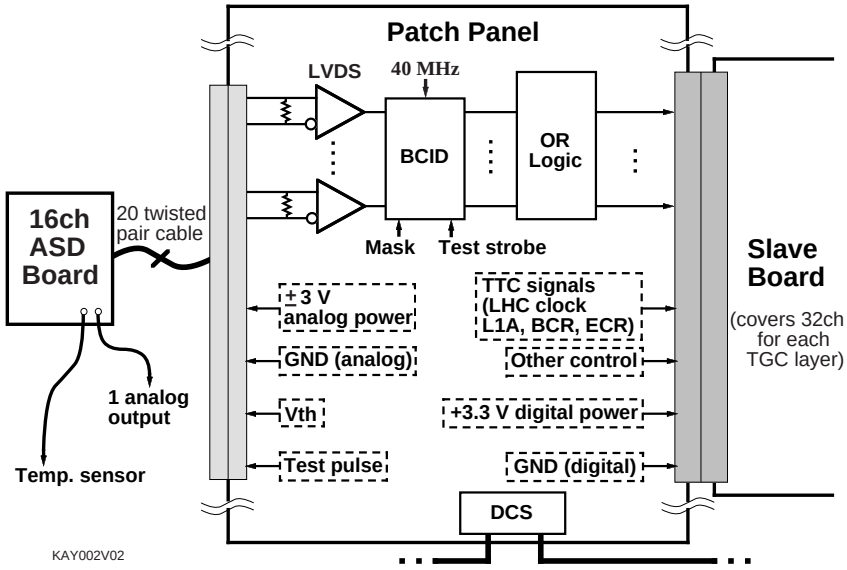

図 3.9： Patch Panel の機能ブロック図

なお、Patch Panel は Slave Board などと合わせて図 3.10 のような PS-Pack というものを形成している。1 つの Patch Panel には 2 つの Slave Board がつながっており、これが一つの単位になっている。さらに、各 PS-Pack には Service Patch Panel と呼ばれるモジュールが備えられており、TTC とのインターフェースである TTCrx（2.2.5 参照）や DCS とのインターフェースである LMB（2.1.4 参照）が備えられる。

3.3.3 Slave Board

Slave Board は Patch Panel から信号を受け取りコインシデンスをとる。図 3.6 に示すように、Slave Board の段階では、ワイヤとストリップ、ダブレットとトリプレットは別々にコインシデンスをとる。この他、インナーステーション（図 3.1 参照）用の Slave Board も存在する。結局、Slave Board は表 3.5 のように 5 種類が存在することとなる。Slave Board は ASIC を用いて開発されるが、開発費を抑えるためにこれらの 5 種に対して共通の ASIC を 1 つ開発し、トリガーマトリックスの部分を切替えて用いる予定である。

ワイヤダブレット用 Slave Board　図 3.11 にワイヤダブレット用の Slave Board の機能ブロック図を示す。ワイヤダブレット用の Slave Board は、Patch Panel から 2 つのダブレットの計 4 層分の信号を 32 チャネルずつ受け取る。これに加えてインナーダブレットについては 1 層につき 6 本ずつ、ピボットダブレットについては 1 層につき 2 本ずつ、隣接する両側の Slave Board に入っている信号を受け取っている。それゆえ、インナーダブレットは 1 層当たり 44 本の、ピボットダブレットは 1 層当たり 36 本の入力信号があることになる。これらの入力信号は、位相調節器（Phase Adjust）とトリガーマスクを経てコインシデンス行列に入る。

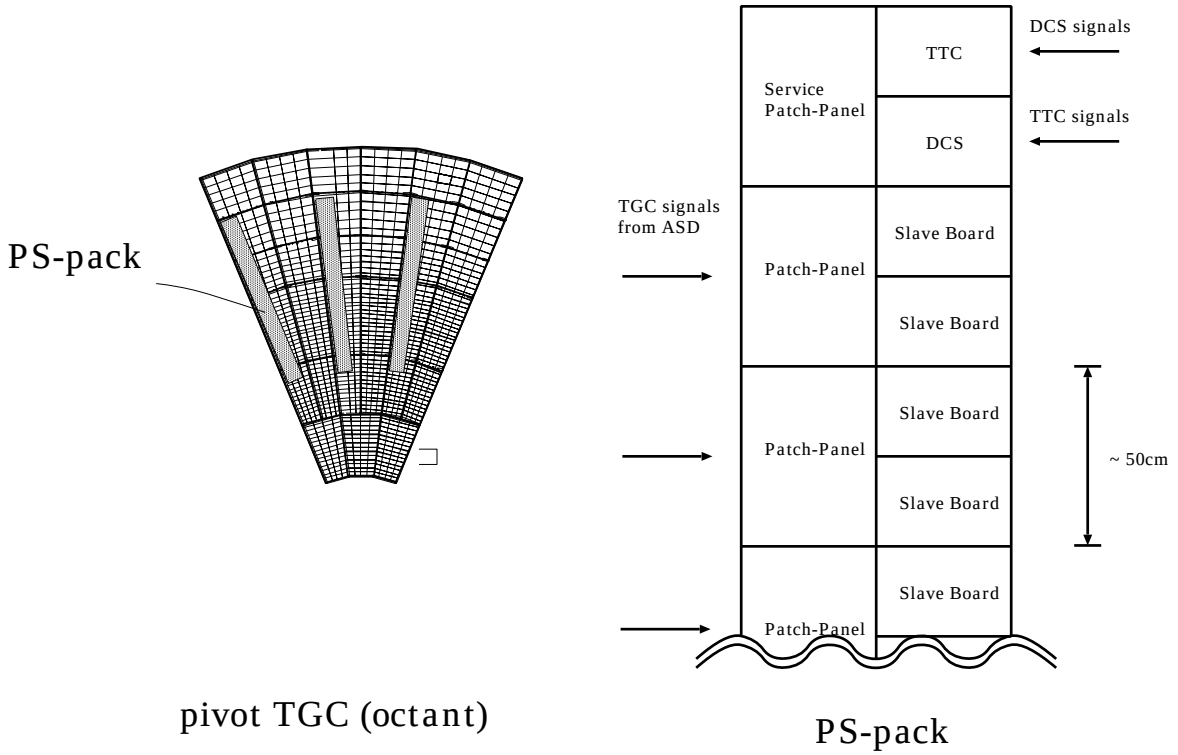

図 3.10： PS-Pack の配置と構成。PS-Pack は、1 つの Patch Panel と 2 つの Slave Board を単位としたものが、多数つながっている。各 PS-Pack には Service Patch Panel と呼ばれるモジュールが設置され、TTC や DCS とのインターフェースの役割を果たす。

Type	Input[/layer]	Layer	Input	Output	Coincidence
Doublet Wire	32	4	128	18	3/4
Triplet Wire	32	3	96	18	2/3
Doublet Strip	32	4	128	16	2/3
Triplet Strip	64	2	128	40	1/2
EI/FI	64	2	128	6	1/2

表 3.5： Slave Board の種類。この 5 種類の Slave Board は共通の ASIC で開発される。

そもそも入力信号は Patch Panel でバンチ識別されているが、これは相対的な識別にすぎない。チェンバーから Patch Panel までのケーブルの長さの差などが原因で、Slave Board に異なる Patch Panel から同時に到着した信号が必ずしも同じバンチに属しているとは限らない。それゆえ、位相調節器で半クロック単位のディレイで調節出来るようになっている。トリガーマスクはストリップ用のチェンバーの場合に ϕ 方向への重なりの部分のチャネルを止めたり不良チャネルの信号を止めてしまうためのものである。トリガーマスクはまたテストの際にも用いることが出来る。

ワイヤダブレット用のコインシデンス行列を図 3.12 に、その詳細を図 3.13 に示す。コインシデンスは 4 層中 3 層以上ヒットがあることが要求される。コインシデンス行列の中は 2 つに分かれている（図 3.12 の A と B ）。各々の部分で $-7 \leqq \delta R \leqq 7$ の領域でコインシデンスが行われる。コインシデンス結果はデ

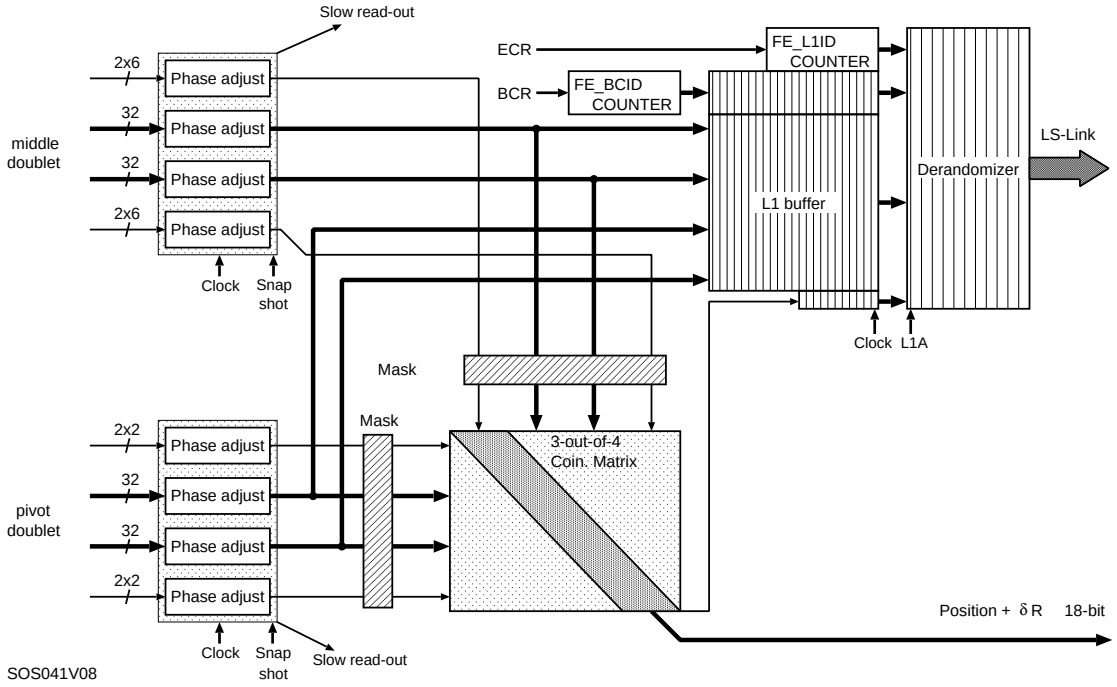

図 3.11： ワイヤダブレット用の Slave Board の機能図。トリガーに関連する部分としては、位相調節、トリガーマスク、トリガー行列がある。その他の部分は読み出しに関連する部分である。

クラスタリング（隣接する場所にヒットがあっても、れを 1 つの粒子が通っただけとみなし、1 箇所だけのヒット情報に直すこと）した後、2 つの領域で各々最も p_{T} が大きな、すなわち $|\delta R|$ が小さなトラックを 1 つ選び出す。そして、5 ビットの位置 R と 4 ビットの δR にエンコードされ、合計 18 ビットの情報が High-p_{T} Board に送られる。

ストリップダブレット用 Slave Board　ワイヤダブレット用の行列は $|\delta R| \leqq \pm 5$ の範囲で δR を求めたが、ストリップダブレット用の行列は $|\delta\phi| \leqq \pm 3$ の範囲で $\delta\phi$ を求める。したがって、ストリップダブレット用の Slave Board は出力ビット数が 18 ビットから 16 ビットに変わる以外は、ワイヤダブレット用の Slave Board と全く同じ構造をしている。

ワイヤトリプレット用 Slave Board　ワイヤトリプレット用の Slave Board とトリガー行列を図 3.14 と図 3.15 に示す。各層 32 本と、加えて隣接ボードとの共有信号各層 2 本ずつを受け取り、3 分の 2 のコインシデンスを行う。トリプレット用の Slave Board からはヒットの有無とヒット位置情報を High-p_{T} Board に送信する。ワイヤトリプレット用の Slave Board の場合は、図 3.15 のように、ヒット情報は 5 ビットで 3 つまで候補を送るので、出力ビット数は 18 ビットということになる。

ストリップトリプレット用 Slave Board　ストリップトリプレット用の Slave Board とトリガー行列を図 3.16 と図 3.17 に示す。トリプレット用チェンバーにはストリップは 2 層しかないので、コインシデンスは 2 分の 1（従って単なる OR 回路）になる。トリプレットストリップ用の Slave Board は他の Slave Board の 2 倍の領域を担当しており、一層当たり 64 本の入力がある。32 入力当たり 4 つのヒットを High-p_{T} Board に送るので、図に記してあるように出力ビット数は 40 ビットということになる。

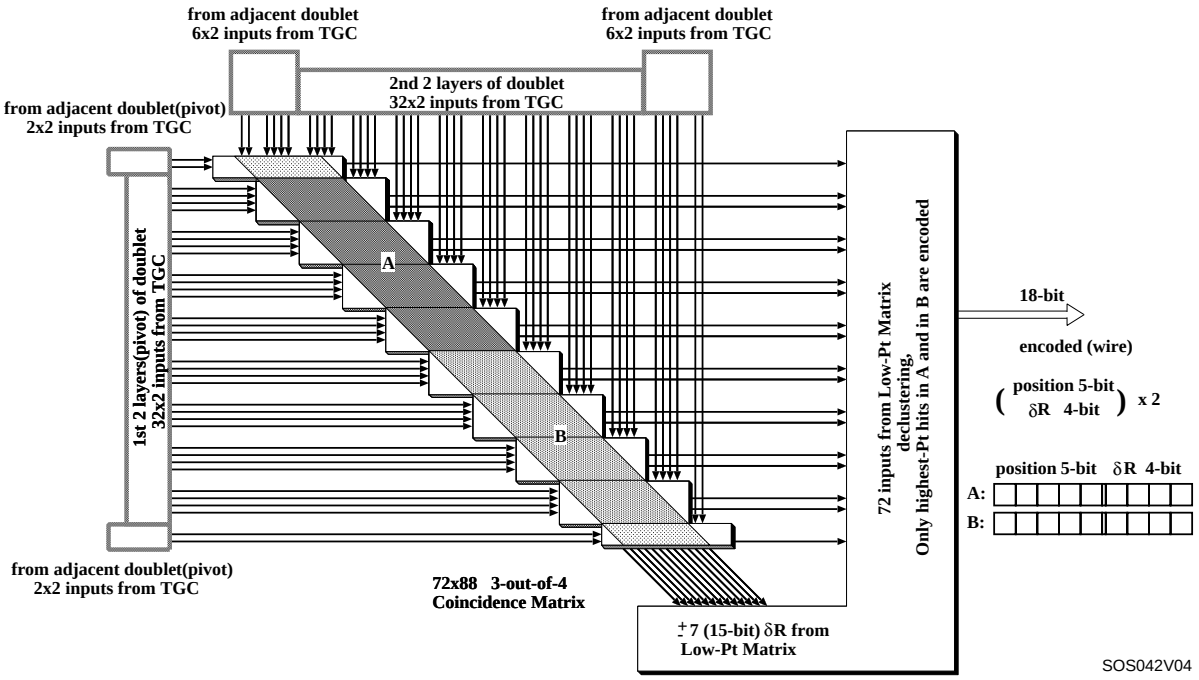

図 3.12： ワイヤダブレット用のコインシデンス行列。トリガー行列は 2 つの部分に分けられ、各々で最も $|\delta R|$ が小さなトラックが選び出される。

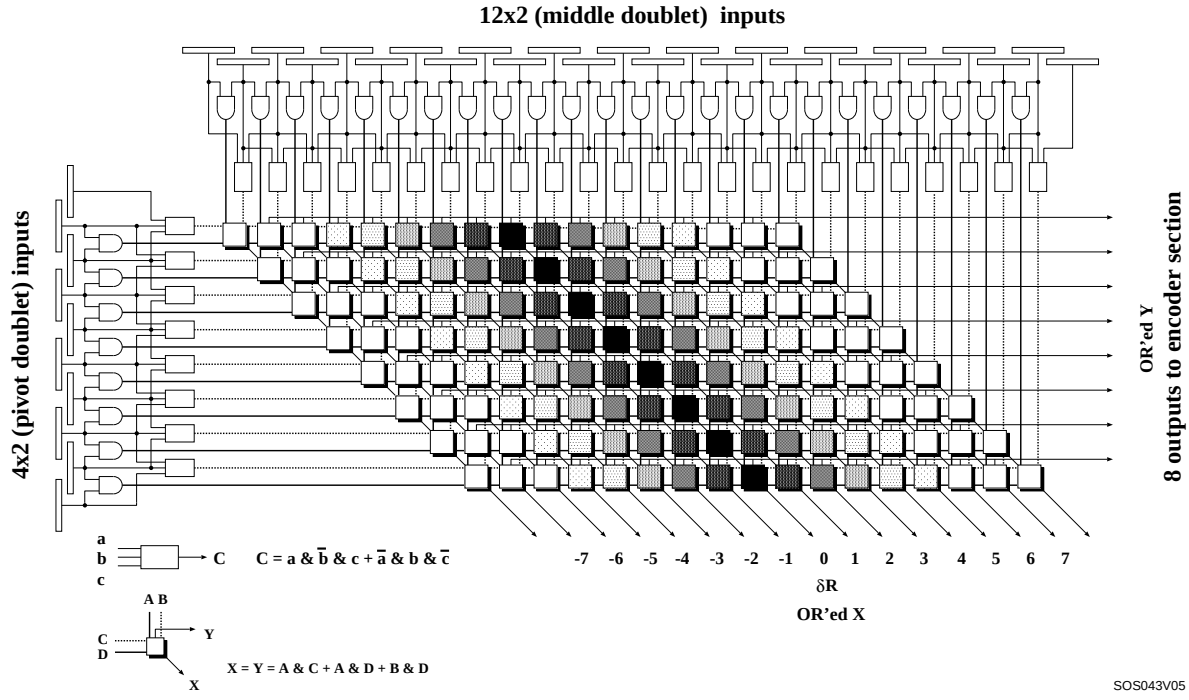

図 3.13： ワイヤダブレット用のコインシデンス行列の詳細。3 out-of 4 のコインシデンスが実現される。

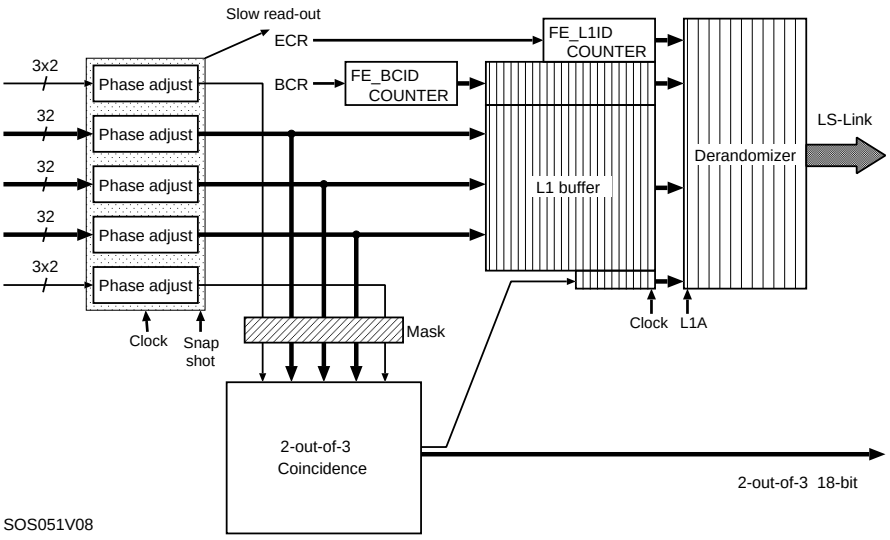

図 3.14： ワイヤトリプレット用の Slave Board

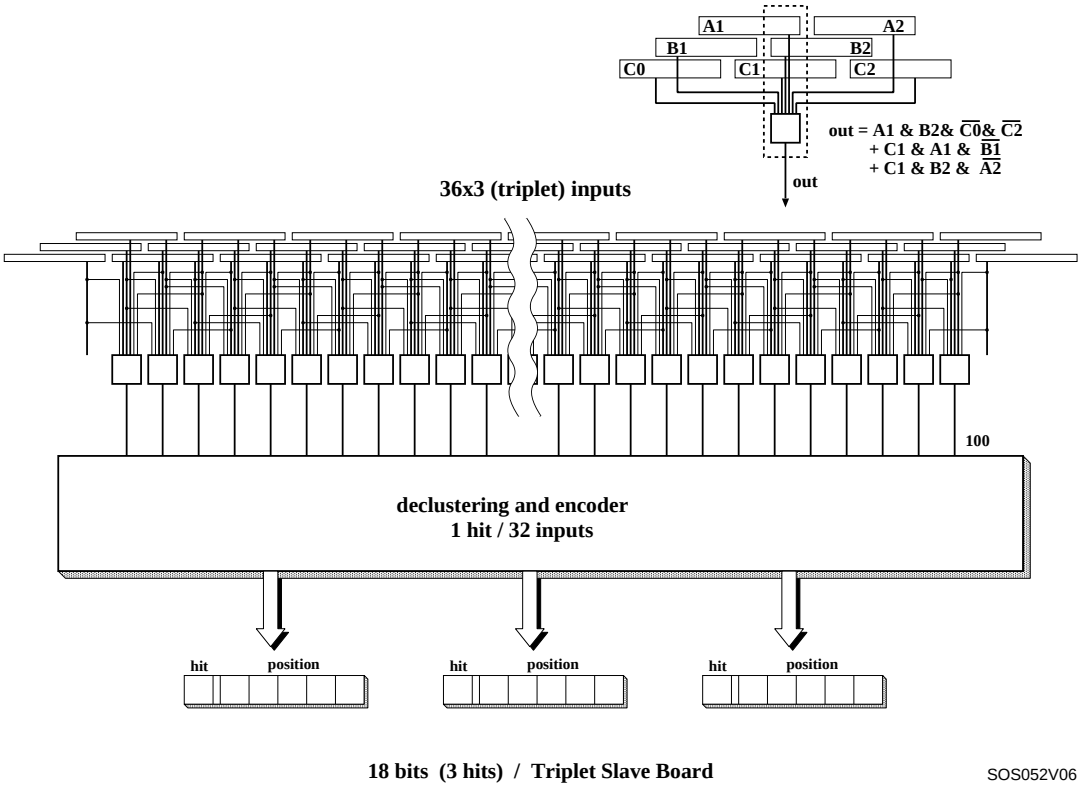

図 3.15： ワイヤトリプレット用のコインシデンス行列。2 out-of 3 のコインシデンスが実現される。

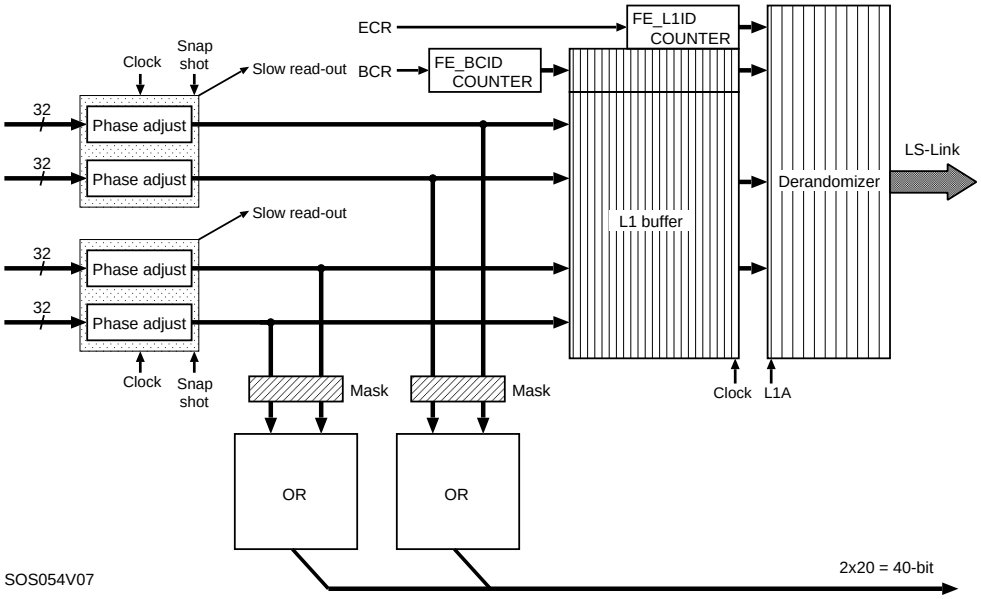

図 3.16： ストリップトリプレット用の Slave Board

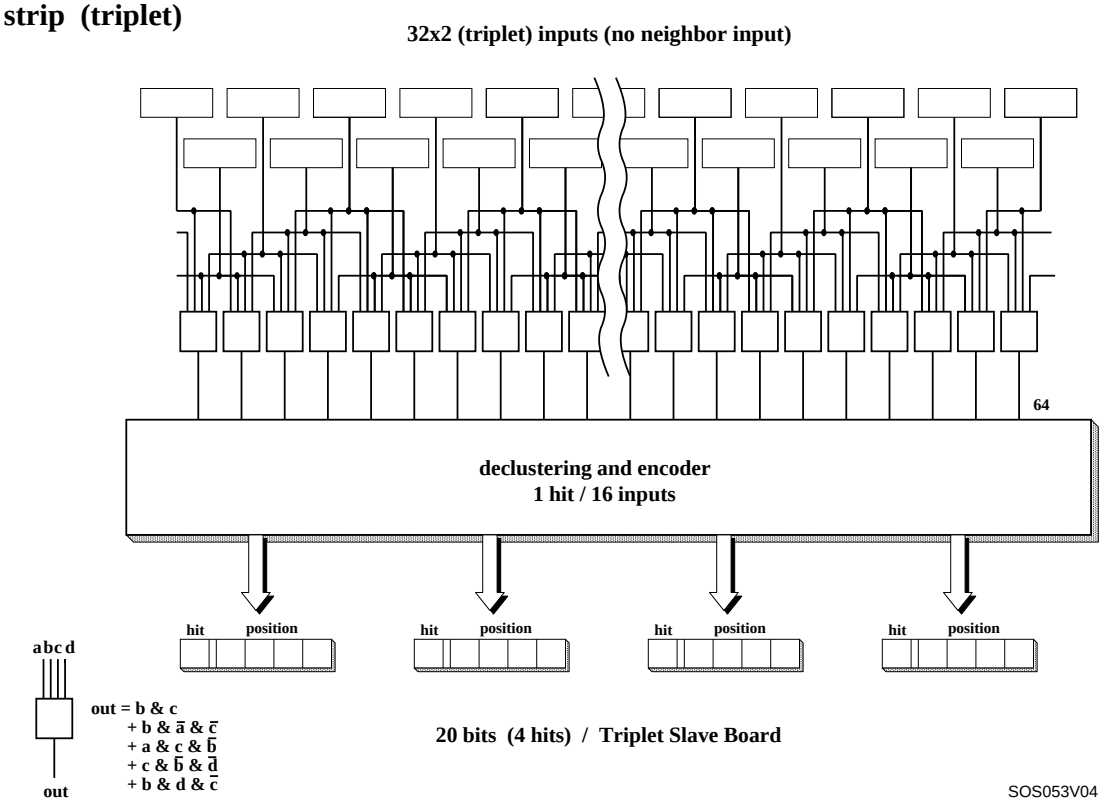

図 3.17： ストリップトリプレット用のコインシデンス行列。基本的には単なる OR である。

3.3.4 High-p_{T} Board

High-p_{T} Board はワイヤまたはストリップごとにコインシデンスをとるためのボードである。1 つの High-p_{T} Board には、2 つの同じ働きをする High-p_{T} Board ASIC が実装されている。

図 3.18 にワイヤ用の High-p_{T} Board ASIC の機能図を示す。図の様に、High-p_{T} Board ASIC は 3 つのダブレット Slave Board と 4 つのトリプレット Slave Board からの信号を受け取り、コインシデンスをとる。

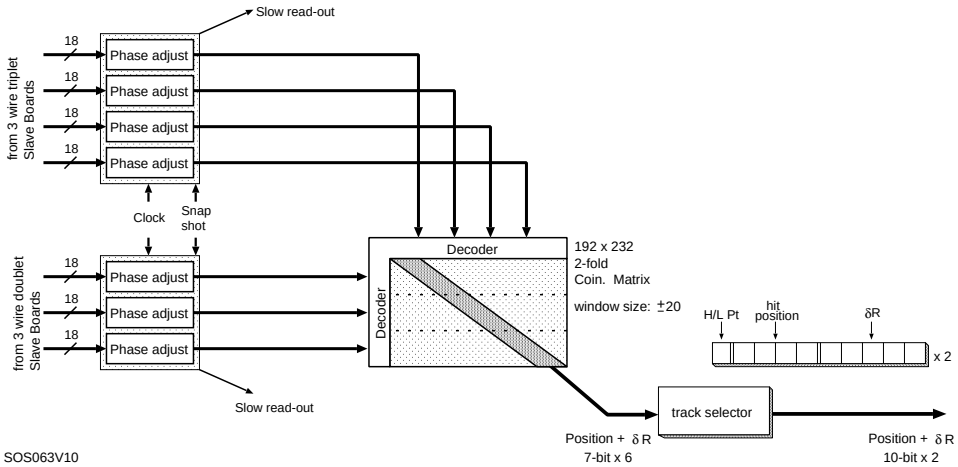

図 3.18： ワイア用 High-p_{T} Board ASIC の機能図。ワイア用 High-p_{T} Board ASIC では 3 つのダブレット Slave Board と 4 つのトリプレット Slave Board から信号を受け取り、192×232 のコインシデンスをとる。そして、トラックセレクタでトラックの候補を 2 つ絞る。

1 つのダブレット Slave Board は 32 チャネル分を処理していたが、コインシデンスを取る際にワイヤが互い違いに配置されている性質を利用して実効的に 2 倍に位置精度を高めることができるので、ダブレット側からのチャネル数は $32 \times 2 \times 3 = 192$ ということになる。これに対して、トリプレット側は実効的に位置精度を 3 倍に高められるので、2 つの Slave Board からの入力でダブレット側と対応するチャネルを賄うことができる。ただし、トラックが曲がることによって生じる δR を考えなければならないので、上記の 2 つの Slave Board の他にその両側の Slave Board からの入力も必要である。結局、トリプレット側は 4 つの Slave Board からの入力を受け取ることになる。また、トリガー行列は、δR の分を考慮してトリプレット側は片側あたり 20 チャネル分を余分に受け取るので、トリガー行列への入力は 232 入力ということになる。

Slave Board からの入力信号は Slave Board の出力時点でエンコードされているので、コインシデンスを取る前にデコードする必要がある。デコード後に 192×232 の大きさのコインシデンスが取られる。図 3.19 にこのうちの 4 分の 1 の部分を示す。図の部分では、コインシデンスの行列はさらに 2 つに分割され、各々から p_{T} が最大（δR が最小）となるトラックを探し出す。その部分のロジックは図 3.20 に示すとおりである。これは図 3.19 の 8 分の 1 のブロックに相当する。図 3.20 に示すように δR のウィンドウの大きさは $-20 \leqq \delta R \leqq +20$ であるが、これを 5 ビット（$-15 \leqq X \leqq +15$）の情報に収めるために、$|\delta R| > 10$ の領域では 2 つの δR をまとめて扱っている。

コインシデンスをとることにより全体から 8 個のトラックの候補が得られるが、トラックの数を減らすためにトラックセレクターで候補を 2 つにまで絞る。したがって、High-p_{T} Board 全体では最大 4 つのトラック候補を送ることが可能である。これらの情報は光ファイバーで Sector Logic に送られる。

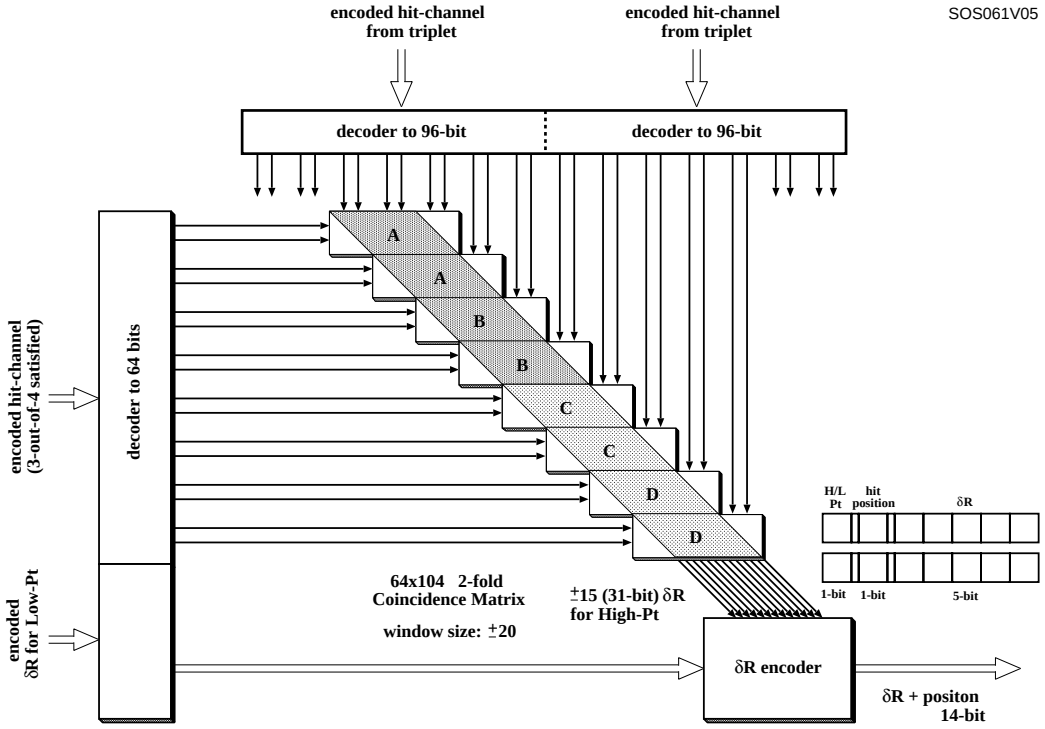

図 3.19： High-p_{T} Board コインシデンス行列。192×232 のコインシデンス行列の 4 分の 1 の部分。

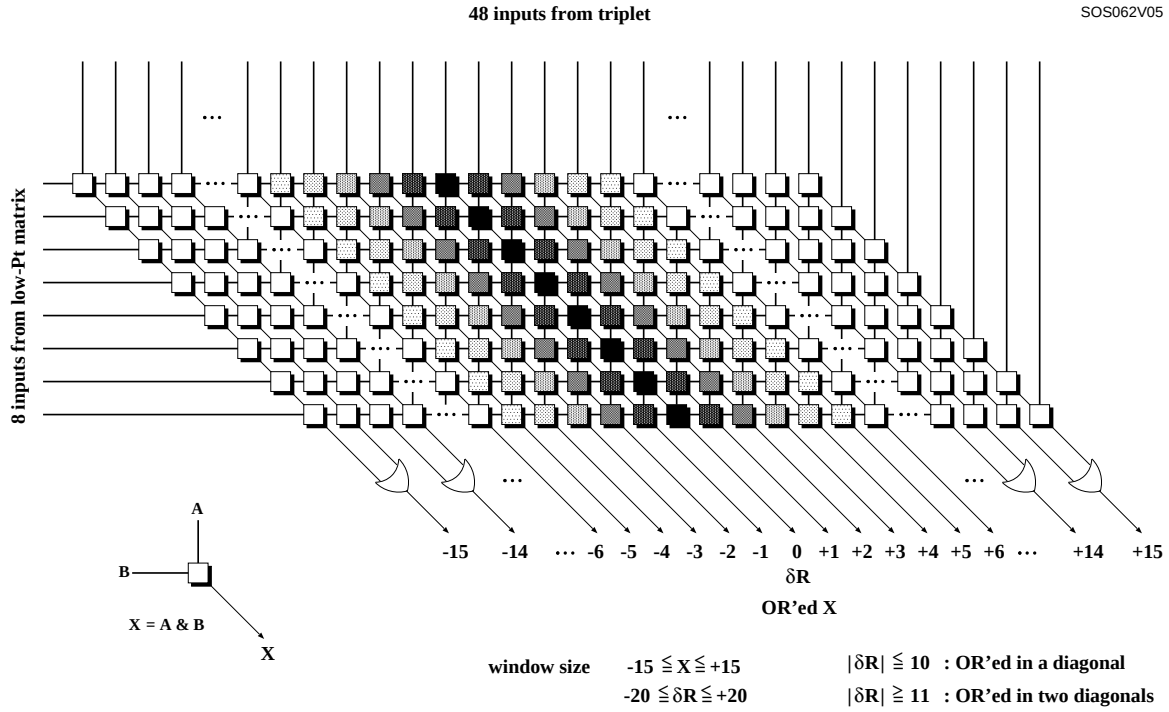

図 3.20： High-p_{T} Board コインシデンス行列の詳細。図 3.19 のさらに 2 分の 1 の部分。

以上はワイヤ用の High-p_{T} Board の場合であるが、ストリップ用の High-p_{T} Board についても入力信号の数や $|\delta\phi|$ の選択範囲が異なることを除けば基本的には同じであるので、ここでは図 3.21 に機能図を示すだけにとどめておく。

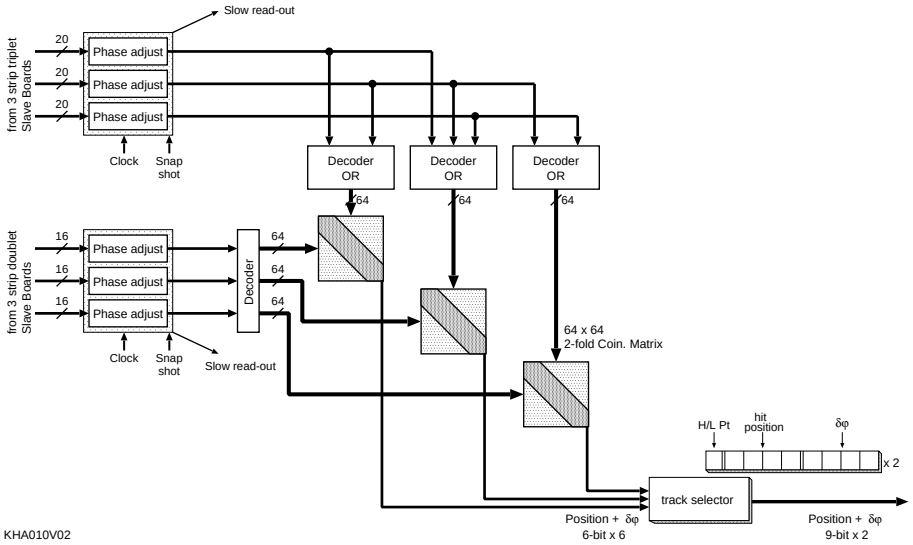

図 3.21： ストリップ用 High-p_{T} Board ASIC の機能図。

なお、当初は High-p_{T} Board の入力信号に対しても読み出しを行う方針であったが、High-p_{T} Board 入力は Slave Board 出力と同一なので Slave Board で読み出しが可能なことなどから、読み出しを行わないように方針を変更しつつある。これに伴い、High-p_{T} ASIC は Slave Board ASIC を切り替えて使う予定であったのを、別に小規模な ASIC を開発するように変更する予定である。

3.3.5 Sector Logic

Sector Logic はワイヤ用 High-p_{T} Board から $R, \delta R$ に関する情報を、ストリップ High-p_{T} Board からは $\phi, \delta\phi$ の情報を受け取る。そして、最終的な R-ϕ コインシデンスを行ったあと、各トラックを 6 段階の p_{T} に分ける。そして、p_{T} のおおきな 2 つのトラックを選択し、その情報を MUCTPI に送る。おおまかな機能図は図 3.22 のとおりである。

現在のところ次のような処理の流れを想定している。図 3.23 に示すように、エンドキャップの場合にはセクタを 19 の SSC（Sub-Sector Clustor）に分割する。そして各 SSC 内で R-ϕ コインシデンスをとり、p_{T} が最大のトラック候補を 1 つ決定する。この際、$\delta R, \delta\phi$ と p_{T} の関係はテーブルとしてメモリ内に記憶されており、このテーブルを参照することによって 6 段階の p_{T} 値を得る。この時点で、最高 19 のトラックの候補が存在する。次に、p_{T} ごとにトラックの候補を分類し、プリセレクタで p_{T} の値ごとに候補を 2 つに絞る。最後に、得られた 12 個の候補の中から p_{T} が大きな 2 つのトラック候補を選び、この情報を MUCTPI に送信するのである。

Sector Logic は全体で 144 個しかないため、FPGA を用いて開発される予定である。しかし、Sector Logic に与えられた latency は 8 クロックしかないので、1 クロックの間にかなりの複雑な処理を行う必要がある。そのためには高性能な FPGA を導入する必要がある。また、$\delta R, \delta\phi$ と p_{T} の関係を与えるテーブルをどのような方法で実現するかなど、解決すべき課題は多い。

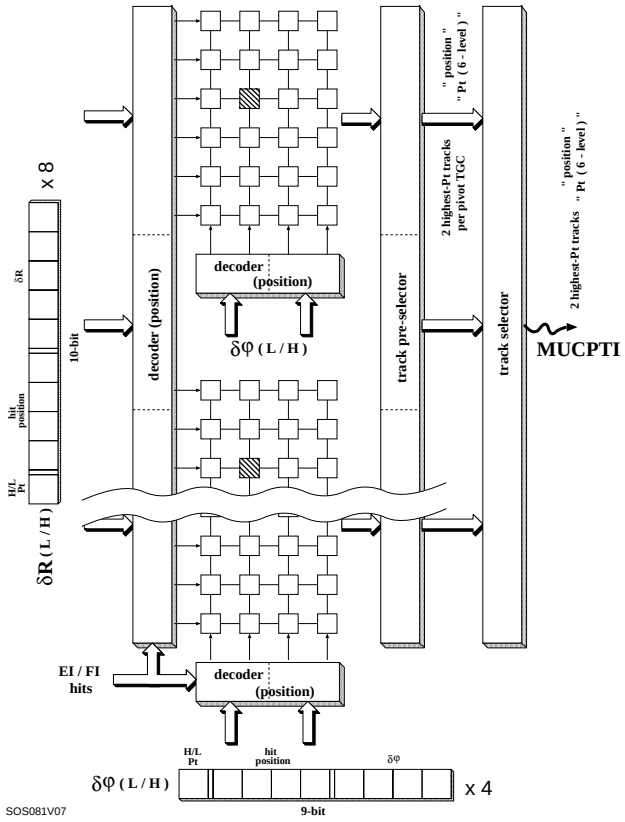

図 3.22： Sector Logic の機能図。Sector Logic では R-ϕ コインシデンスを行った後、最終的にトラック候補を 2 つ選び、それを MUCTPI に送る。

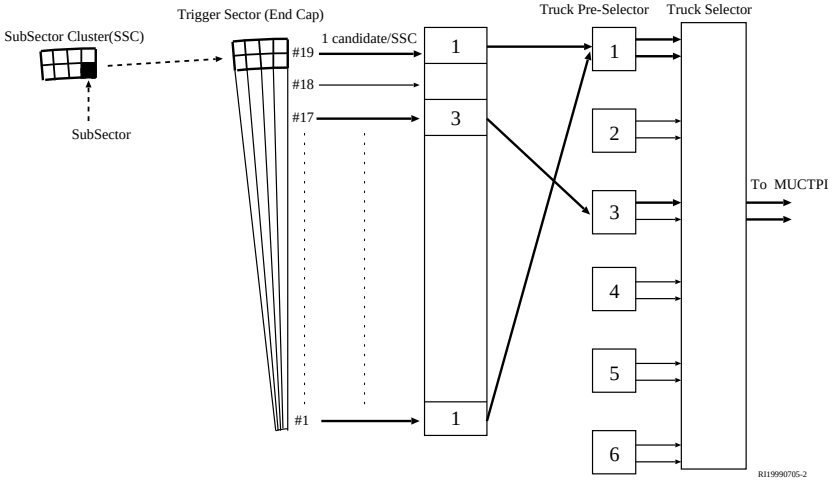

図 3.23： Sector Logic の処理の流れ。19 の SSC（Sub-Sector Clustor）ごとにトラック候補を選ぶ。次にトラック候補の p_{T} ごとにトラックの選択を行い、最後にその中から p_{T} の小さなトラックを 2 つ選ぶ。

第 4 章

TGC の読み出し系の設計

4.1 序論

ATLAS の TGC システムは、イスラエル、日本、中国が開発を行っているが、FE（Front-end）のエレクトロニクスの開発は日本グループが中心となって行っている。そのなかでも、京都大学は TGC エレクトロニクスの読み出し系の部分を担当している。今回、Slave Board と Star Switch の設計を行い仕様を定めた。また、プロトタイプ用のモジュールを製作しスキームが正しいかどうかを調べた。

この章では HDL（ハードウェア記述言語）を用いた大規模回路の設計法について述べた後、読み出し系の設計について述べる。次章では読み出し系の検証について述べる。

4.2 大規模電子回路の設計

4.2.1 大規模設計

ATLAS の TGC のエレクトロニクスに限定しても、膨大なチャネルの信号に対してトリガーの生成やデータの読み出しを行わねばならない。このような大規模なシステムはトップダウンに設計を行わなければ見通しが非常に悪く、しかも複雑になってしまう。ここでは、大規模なシステムを構築する際の方法について考えてみる。

まず最初に行うべきことはシステムに対する要求を調べることである。要求は、そのシステムが実現したい性能かもしれないし、それ以外の要因でシステムに加わる制限事項かもしれない。特に多人数で開発を行う場合には、これらの要求は URD（User Required Document）の形でまとめておく必要がある。

次に分析を行う。URD に基づいて、システムがどういう振舞をするべきかという点を考察し、システムの機能の仕様書を記述するのである。分析とはシステムの振舞のモデル化ということになる。もっとも、分析の段階でシステムの全ての振舞が分かるわけではない。むしろ、この後の設計を行うことによって新たに必要な振舞が見いだされたり、あるいは分析段階で考えた振舞が妥当かどうかがわかる。ある程度矛盾がない形でシステムの振舞の記述が出来たら、次の設計段階に進む。

設計では分析した振舞のモデルを実現するための手段を作り出す。この時点で機能の分割を行う。システムを小さな機能に分割することを下位の構造に向かって繰り返す。階層ごとに機能間の相互の接続（インターフェース）の仕様を決定する。これにより、全体としてトップダウンの階層構造ができあがる。

またハードウェアの調査が必要である。即ち、作ろうとしているシステムにどのような技術を用いるかという選択を行う。全く新しい技術を用いるのか既存の安定したものも用いるのかということを、分析や設計の結果を参考にしながら決定する。

設計がある程度終了すると、テストを行う。テストを行うためには、テスト環境やテスト方法、テスト項目などの詳細な仕様書を作って置く必要がある。これらの仕様書に従って、シミュレーションを行ったり、実際にプロトタイプの試作を行って機能の検証を行ったりする。十分にテストを行ったら、最終的な完成品の開発を行うこととなる。

大規模システムの開発の手順としては上記のようになるが、実際にはこのように直線的に物事が進むわけではない。既に述べたように、設計をすることによって不足している分析が見えてくることも多いし、テストを行った結果それ以前の作業に修正を加える必要が出てくる。また、トップダウンの設計を行った場合にでも、トップダウンの最下層の実装の際には諸々の制限が加わる。例えば、ASIC の製作の際には、トップダウンで階層構造化した結果を提供されているライブラリをうまく利用できる様に調節する必要がある。このように、ボトムアップの手法と付き合わせることによって、トップダウンの設計に変更を加えることもある。

それゆえ、上記の手順は何度も繰り返すことが不可欠になってくる。

4.2.2 ハードウェア記述言語

ハードウェア記述言語（Hardware Description Language：以下 HDL と略す。）は大規模な論理回路の開発の際に用いる非常に強力な道具である。まず、この章で HDL について説明する。HDL を用いた開発法については次節以降で述べる。

例えば ASIC[1]の開発について考えてみる。ASIC は数多くのトランジスタが集まって電子回路を形成している。原理的にはトランジスタを配置とそれらの間の配線を全て指定すれば目的の電子回路を作ることが出来る。しかし、この方法は少し複雑な回路を作ろうとするとすぐに破綻する。そこで、NAND や NOR といった論理ゲートを単位に設計するようになった。この方法では、開発者は回路図入力と論理シミュレータを用いて回路の作成検証を行う。そして、トランジスタの配置や配線は、ゲートレベルで記述された回路図をもとに自動生成されるのである。

が、半導体技術の進歩によってさらに大規模な電子回路の製作が可能になってくると、ゲートレベルの記述では開発や検証が困難になってきた。ゲート数が多くなってくると、問題点が見つかったときに信号線を一つ一つ追っていくのに膨大な時間がかかるし、それ以前に何万という論理ゲートを並べるということ自体、多大な労力である。

そこで、RTL（Register Transfer Level）で回路を記述する試みがなされた。RTL とはクロックやレジスタの位置を指定して、レジスタからレジスタへのデータの流れで回路の動作を記述することを意味する。HDL は RTL での記述を可能にしている。

HDL はデジタル回路の動作を言語レベルで設計、検証できるようにしたものである。HDL は前述の RTL を初め、さらに抽象的な動作レベル（Behavior Level）、あるいはゲートレベルに相当するネットリストのレベルで記述を行うことが出来る。HDL で記述したものは、合成ツールを用いてゲートレベルに変換しなければならないのだが、現時点では動作レベルで記述したものを変換してくれる合成ツールはないようである。従って、開発の際には RTL 以下のレベルで記述することになる。

目下標準的に用いられる HDL としては VHDL と Verilog-HDL があるが、ATLAS の TGC エレクトロニクスの開発には Verilog-HDL が用いられている。Verilog-HDL での記述の例を図 4.1 に示す。これは非同期リセット付き 4 ビットカウンタを Verilog-HDL で記述したものと、それを Synopsys 社の Design Analyzer で論理合成した結果である。

4.2.3 HDL を用いた回路の開発

HDL を用いて ASIC を設計したり、FPGA で目的の回路を実現するための手順は次のようになる。

[1]ASIC は Application Specific Integrated Circuit の略で、汎用の LSI に対して特定用途向けの LSI を指す。

```
module counter(clk, reset_, enable, q);

input clk, reset_, enable;
output [3:0] q;
reg [3:0] q;

always @(posedge clk or negedge reset_) begin
   if(!reset_) q <= 0;
   else if(enable) q <= q + 1'b1;
end

endmodule
```

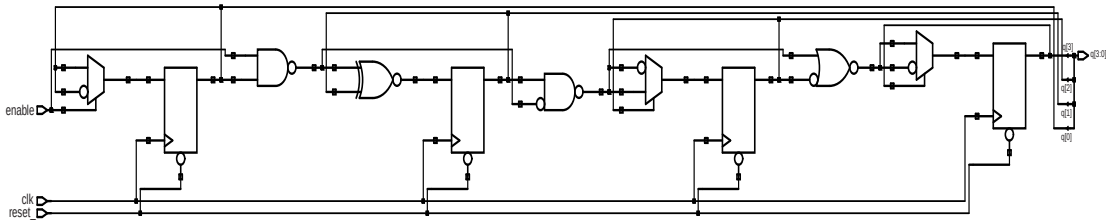

図 4.1： Verilog-HDL の記述例と合成結果

- 仕様の決定
- HDL 記述
- 論理シミュレーション
- 論理合成
- ゲートレベルでのシミュレーション
- レイアウト（配置配線）
- レイアウトの検証
- ASIC の製作、FPGA へのダウンロード
- 検証、性能評価

仕様が決定し HDL 記述が完了すると、まず論理シミュレーションを行う。HDL で記述したモジュールに対してテストパターンを入力し、それに対する出力結果を見ることによって、HDL の記述が論理的に正しいかを調べることが出来る。シミュレーションの際にはモジュール内部のレジスタや配線を見ることが出来る。論理シミュレーションの結果が仕様通りになるまで、HDL 記述と論理シミュレーションを繰り返すことになる。

論理シミュレーションが終了すると、論理合成を行って HDL の記述をゲートレベルのネットリストに変換する。変換した結果に対して、ゲートレベルでのシミュレーションを行う。このシミュレーションではゲートレベルの遅延時間と予想される配線遅延をもとに、モジュールがデバイス上で正しく動作するこ

とが出来るかをシミュレーションする。また、もっとも遅延時間の大きな経路（クリティカルパス）からデバイスの動作周波数が予測できる。

この後、ネットリストのレイアウト（配置配線）を行うのであるが、FPGA と ASIC、あるいは ASIC の中でもゲートアレーとカスタム IC では少し内容が異なる。カスタム IC の場合はゲートのトランジスタの配置を行わなければならないのに対し、ゲートアレーや FPGA ではゲートが固定されているので、ゲート間の接続だけを考えればよい。しかし、どちらの場合にもこれらの作業は自動で行うので、見た目には違いはあまり分からない。ASIC の開発時にはレイアウト終了後にレイアウトの検証が必要である。

以上の手順を繰り返しレイアウトがうまく行けば ASIC の場合は設計は終了で、数ヶ月の製作期間を経て ASIC が手に入る。これに対して FPGA の場合には、論理をすぐに書き込むことが出来るので、すぐに検証が可能である。従って FPGA の場合には、検証、設計を何度でも繰り返すことが可能である。

4.2.4 HDL と大規模設計

HDL の特徴として挙げることが出来るのは、RTL のレベルで記述を行っている限り HDL の記述はデバイスに依存しないということである。開発の途中でデバイスの変更があっても論理合成からやり直せばすむし、デバイスが確定しない段階でも HDL 記述を始めることが可能である。これらは大規模な設計の際に起こりうる状況であろう。また、最終的に ASIC で開発するものを FPGA で試作するというようなことが、非常に円滑に行うことが出来る。

また、HDL は振舞を抽象的に記述できる。大規模設計について述べたときに、分析段階でシステムの振舞のモデル化を行うと述べた。HDL を用いてこの振舞を記述することによって、分析の結果が妥当であったかどうかを検証できる。

最後に、HDL での記述はモジュール化されている。HDL ではプログラミング言語の関数のように、特定の機能を実現するための構成単位をモジュールとして扱う。そして、あるモジュールの中に他のモジュールを埋め込んだりすることができる。これはちょうど大規模設計における構造化、階層化に対応する。すなわち、大規模な回路をトップダウンに設計してモジュールに分割した後、HDL を利用すればそのモジュール群の構造を保ったまま設計、記述が可能である。これは、HDL 大規模な論理回路を実現するために開発されたので当然のことである。

このように、大規模な電子回路の設計を行う場合には、HDL は非常に有用なものであるといえる。

4.3 TGC の読み出し系の設計

4.3.1 読み出し系への要請

すでに述べたように、TGC のヒットデータは第二座標（ϕ 座標）と時間情報を与える。そのため、レベル 1 のトリガーで採用されたイベントに対しては、TGC のヒットデータを読み出す必要がある。ここでは、TGC の読み出し系に対してどのような要求、条件があるかを考えてみる。

読み出すべき総チャネル数は TGC 全体で 320 k チャネル、オクタントあたり 20 k チャネルである（表 3.3 参照）。以下オクタントあたりで考えると、L1A のレートは最高 100 kHz であるから、読み出すべきデータ量を単純に計算すると 2 Tb/s ということになる。ただし、TGC のヒットレートはかなり低いと見積もられている。表 4.1 に予想されるオクタントあたりのバックグラウンドのレートを示す [12]。表のように、オクタント全体でも 1 バンチあたりのヒットは 3 個以下である。したがって、読み出しているデータはほとんど 0 であり、データの圧縮を行うことによって、データ量ははるかに少なくなる。

次に、エレクトロニクスの設置場所の問題がある。エレクトロニクスなどの機器の設置場所としては、

- チェンバーに直接搭載

	Rates [Hz]		Hits per event	
	Wires	Strips	Wires	Strips
Inner	1.43×10^6	1.44×10^6	0.36	0.36
Triplet	2.44×10^6	2.63×10^6	0.61	0.66
Doublet	3.77×10^6	6.37×10^6	0.94	1.59

表 4.1： TGC で予想されるオクタントあたりのバックグラウンドのレート。Hits per event は、1 バンチあたりのヒットレート。

- 実験室内でチェンバーの近く。TGC エレクトロニクスの場合には、TGC を支えるホイールの表面または縁
- USA15 と呼ばれるエレクトロニクス室

が考えられる。最終的にはデータは USA15 内の ROB と呼ばれるモジュールに送らなければならない。

実験室内に設置した場合には放射線が問題になる。TGC のホイール上での放射線レベルとエレクトロニクスに要求される耐放射線性を表 4.2 に示す [3]。表中の放射線レベルは 1 年間に浴びると見積もられている放射線量である。TGC エレクトロニクスは 10 年間運転を続けることを想定されているので、この表の 10 倍の放射線量に耐えられなければならないのだが、CMOS プロセスで製造されたエレクトロニクスの場合には安全係数 4 を乗じた放射線量に耐えられることが要求されている。この値も表 4.2 に示す。なお、ATLAS では検出器やエレクトロニクスの全ての部品に対して、十分な耐放射線性を有するかのテストを行うことが要求されている。

	Radiation Level		Radiation Tolerance	
	Worst Location	Best Location	CMOS	Bipolar
Neutrons [n/cm^2]	9.7×10^{10}	3.1×10^{10}	3.9×10^{12}	5.9×10^{12}
1 MeV Equivalent Neutrons [n/cm^2]	2.0×10^{10}	3.6×10^{9}	8.1×10^{11}	1.2×10^{12}
Dose [Gy]	6.2×10^{-1}	2.1×10^{-1}	2.5×10^{1}	1.3×10^{2}

表 4.2： TGC ホイール上での放射線レベルと耐放射線性の基準。放射線レベルは 1 年間での値で、最もレベルが高いところと低いところでの値が記されている。耐放射線性の基準は総被曝量で、ここにかかれた値の被曝に耐えられるものでなければ使用できない。

実験室内に設置した場合にもう一つ問題になるのが、保守などのためのアクセスが制限されることである。チェンバーやチェンバー直接取り付けたエレクトロニクスに近付くのは、実験を止めない限り難しいであろう。これに対して、TGC のホイールに取り付けたエレクトロニクスについては、検出器のもっとも外側に位置するために比較的アクセスしやすい。特に、ホイールの縁に取り付けたモジュールについては、放射線などの関係からラン中に近付くことはできないが、それ以外の時間はアクセスができるかもしれない。

TGC の読み出し系の開発の上で考えなければならないことが、収集するチャネルが多いだけでなく、地理的にも広い範囲に渡っているということである。実験室内（ホイール上）と USA15 の間をケーブルでつなぐと 80 m になる。また、チェンバー自体が大きく、10 m 程度の範囲からデータを収集する必要がある。チャネル数が多いことに起因する問題でもあるのだが、距離が長いためにケーブルの量も多くなって

しまい、場所、重さ、価格の点で問題がある。そのため、これらのケーブルの数はできるだけ少なくすべきである。一方で、距離が長いため、信号がケーブルを伝播する時間も大きくなる。設計の上ではこれらも考慮にいれる必要がある。

最後に、設計の上では価格についても考慮しなければならない。チャネル数が膨大であるので、それを処理するエレクトロニクスの数も膨大になる。そのため、1 つ 1 つのモジュールをできる限り安く作ることが要求されてくる。

4.3.2 読み出し系の基本構成

以上の要請から、読み出し系の構成をどのようにすればよいのかを考えてみる。

まず、読み出すべきデータはデジタル化されバンチ識別されている必要があるので、Slave Board から読み出しを行う[2]。もっとも単純には図 4.2 のように Slave Board と USA15 内のマスタモジュールまでの間を直接ケーブルで接続すればよい。しかし、Slave Board が 3000 枚近くも存在するので、この方法だと 80 m のケーブルが 3000 本近く必要となり、ケーブルの量を考えると現実的ではない。

そこで、ケーブルの量を減らす方法として図 4.3 のようなバス構造が考えられる。バス構造では、マスタあるいはアドレスを指定された Slave Board がバスライン上にデータを載せることによって通信を行う。この方式では Slave 側はアドレスをデコードするだけなので、さほど複雑なロジックは必要でない。しかし、バスラインが長くなるとバスの動作速度が遅くなってしまう。今回のように Slave Board と USA15 内のマスターが 80 m も離れている場合には、この方式は困難である。また、バス構造の場合には、Slave Board が重度の故障を起こした場合にバス全体が機能しなくなることもありうる。

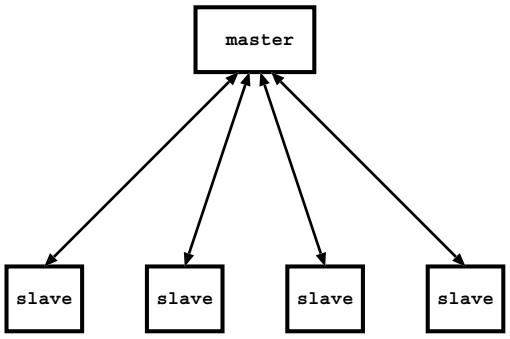

図 4.2： 直接接続によるデータの送信

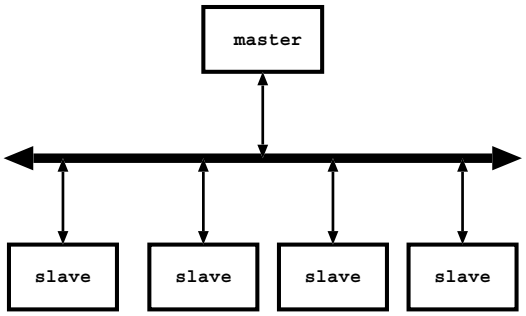

図 4.3： バス構造によるデータの送信

一方、図 4.4 のようなリング構造では、Slave Board は順々に次の Slave Board にデータを送信していき、最終的にマスタのところにデータが到達する。このような方式では、マスターとスレーブの距離が長くても問題が起こらない。しかし、データの送信をどのようなプロトコルで制御するにしても、Slave Board のロジックが複雑になる。また、リング構造の最大の欠点は、 Slave が 1 つ故障するとリング全体が機能しなくなることである。これはリングを二重化するなどして別の経路を用意することである程度は解消されるが、このような処置を行うとさらに Slave Board のロジックが複雑になる。

図 4.5 はスイッチ構造である。これはマスターと Slave Board の間にスイッチを中継させる方式である。この方式では Slave Board からみるとスイッチと 1 対 1 で通信していることになるので Slave Board の

[2]原理的には Patch Panel から読み出すのでもよい。Slave Board から読み出すと Patch Panel 上で OR がとられているチャネルについては直接の読み出しはできなくなる。その意味では Patch Panel から読み出した方がいい。しかし、Patch Panel ASIC は 16 チャネルしか扱わない、Patch Panel ASIC はアナログ回路を含むのでレベル 1 バッファなどで規模が大きくなる読み出し系のデジタル回路を入れるのは望ましくない、Patch Panel の時点での相対的なバンチのずれが Slave Board で調節されている、などを考慮すると Patch Panel から読み出すのは得策ではない。

ロジックは単純になる。しかも、この方式では Slave Board が故障しても他の Slave Board に与える影響は少ない。また、Slave Board とマスタの間の長い距離は、スイッチとマスタ間の 1 対 1 の接続が担うのであまり問題にならない。欠点としては、新たにスイッチが増えること、スイッチが故障するとシステム全体が動作しなくなることが挙げられる。

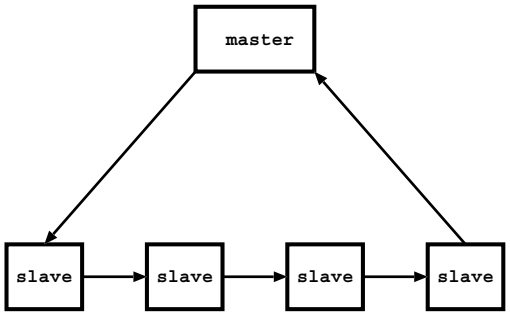

図 4.4： リング構造によるデータの送信

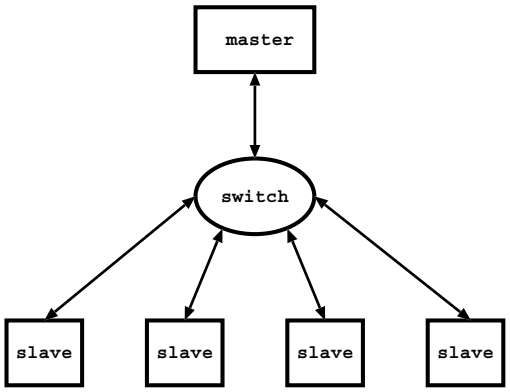

図 4.5： スイッチ構造によるデータの送信

TGC のエレクトロニクスの場合には、Slave Board のアクセスが制限されるので、Slave Board の故障が他の Slave Board の読み出しに影響を与えるリング構造は適当でない。また、Slave Board とマスターモジュールが 80 m も離れている状況で、80 m にも及ぶバスラインを動作させるのは容易ではない[3]。それゆえ、TGC エレクトロニクスの場合には、スイッチを用いた構造が最適であると考えられる。

4.3.3 TGC エレクトロニクス読み出し系の概要

以上の点を踏まえて設計された TGC エレクトロニクス読み出し系の概要を図 4.6 に示す。TGC の全チャネルのデータは Slave Board から読み出され、先ほど 議論したマスタとの間を仲介するスイッチに送られる。このスイッチを Star Switch と呼ぶ。Star Switch は 20 個程度の Slave Board からデータを収集する。Star Switch は Slave Board から集めたデータを USA15 内の LDM（Local DAQ Master）に送信する。ROD は LDM が受け取ったデータを読み出して、RO-Link を経て ROB に送信する。ROB 以降は ATLAS 全体の DAQ のスキームに従って設計されており、レベル 2 の RoI 情報として用いられたり、イベントフィルタで採用されたデータは記録されたりする。

1 つの Star Switch が担当する範囲は LDB（Local DAQ Block）と呼ばれる。従って、1 つの LDB 内には 1 つの Star Switch が存在する。図 4.6 のチェンバーのオクタントの図で、実践で区切られている領域が 1 つの LDB である。一方、点線で区切られている領域はトリガーのセクタである。この図から分かるように、1 つの LDB は 1 つ又は複数のセクタを含むこととなる。

Slave Board は TGC のホイール上、Star Switch は TGC のホイールの縁にあり、これらの間の接続は 10 m 程度である。この接続を LS-Link（Local Slave Link）と呼ぶ。LS-Link でのデータの送信の際には、Slave Board が自分のタイミングで次々にデータを Star Switch に送信するという、Source Synchronous Data Transmission 方式が良いと思われる。10 m もの距離でハンドシェークを行うと、その度にこの距離の伝播遅延の 2 倍に相当する時間を無駄にしてしまい、結果的にデータの転送レートを下げてしまうからである。LS-Link の詳細は 4.6 でのべる。Star Switch から Local DAQ Master までは約 80 m である。

[3]当初は X-Bus という双方向バスを用いてデータ収集を用いることが検討されていた。

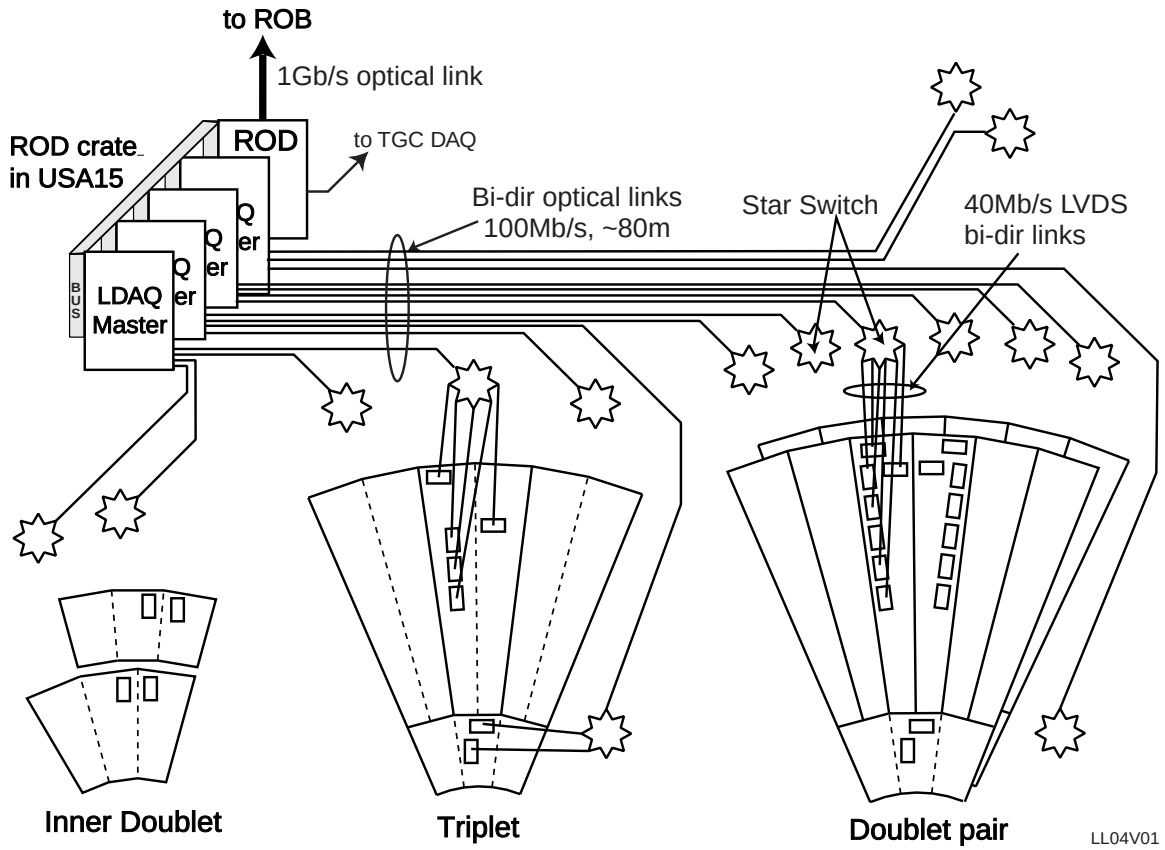

図 4.6： TGC の読み出し系の概略。Slave Board からのデータは Star Switch を経て USA15 内の LDM に送られる。チェンバーの図の中の実践で区切られた領域が LDB で、各 LDB には 1 つの Star Switch が存在する。

この接続は FE-Link（Front End Link）と呼ばれる。ここで必要な帯域は 100 Mb/s と予測されており、光ファイバーを用いる。

Slave Board から読み出すべきデータとしては TGC のヒットデータの他、後で述べるように L1ID や BCID がある。また、当初は High-p_{T} Board から Slave Board でのコインシデンス結果（R, ϕ, δR, $\delta\phi$ に関する情報）を読み出す予定であったが、これらは Slave Board からでも読み出しが可能である。そこで、すでに述べたように Slave Board からのみ読み出しを行い High-p_{T} Board には読み出しの機構を備えない、という方針で開発を行っている。

最後に、TGC の occupancy はかなり低いので、TGC のヒットデータは圧縮することによって大幅にデータ量を減らすことが出来る。従って、エレクトロニクス室にデータを送信する前にデータの圧縮を行う必要がある。データの圧縮は早い段階で行った方がデータの量が減って望ましいが、データの長さが固定でなくなるという弊害がある。そこで、LS-Link を経て Star Switch に送信する段階ではデータ圧縮を行わず、固定長のデータを送信することとする。このようにすることによって、Star Switch にはどの Slave Board からもデータが同時に届くので、ロジックが簡単になるからである。そして、データの圧縮は Star Switch で行うこととする。

最終的に予想されるデータ量は表 4.3 と表 4.4 のとおりである [12]。表 4.3 は LDB あたりのデータ量、表 4.4 はオクタントあたりのデータ量である。圧縮前のデータと、ヒットレートから予測される圧縮後の

データが示されている。圧縮後のデータは、1 つのヒットが 24 ビットで表せると仮定して計算している。24 ビットの内訳は、Slave Board のアドレスが 12 ビット、後述するゼロサプレスを用いたデータ圧縮を行う際のセルのアドレスが 4 ビット、セルのヒット情報が 8 ビットである。実際は、これらのヒットデータの他に L1ID や BCID やステータスといった情報を送る必要がある。これらはヒットの有無に関わらず送信する必要がある。これらの固定長のデータ量はデータのビット数に L1A のレートを乗じたものであるから、ビット数を 100 と仮定すれば 1 MB/s 程度になり、ヒットデータによるデータ量と同じくらいになる。したがって、実際には表の値に加えて十分な余裕を見ておく必要がある。

		Wires		Strips		Total				
LDB Type	No.	SB	Chan.	SB	Chan.	SB	Chan.	Raw[MB/s]	hits/evt	Cmp[kB/s]
Doublet **F**	1	12	1518	3	384	15	1902	23.8	0.11	31.6
Doublet **E**	6	10	1214	5	640	15	1854	23.2	0.37	110.9
Triplet **F**	1	12	1008	3	192	15	1200	15.0	0.07	20.4
Triplet **E**	3	14	1212	4	512	18	1724	21.6	0.19	56.9
Inner **F**	1		188		192	3	380	4.8	0.20	59.1
Inner **E**	1		144		192	3	336	4.2	0.13	37.9

No. = Number per Octant
SB = Slave Board, Chan. = Channels, Raw = Raw Data, Cmp = Compressed Data
F = Forward, **E** = Endcap

表 4.3： TGC の LDB あたりのデータ量。LDB の種類ごとに分けて示す。Slave Board の数やチャネル数なども同時に示す。データレートは 100 kHz のレベル 1 トリガーを仮定している。圧縮データのデータ量は 1 イベントを 24 ビットとして計算している。

LDB Type	No. LDB	No. SB	Chan.	Raw[MB/s]	hits/evt	Cmp[kB/s]
Doublet **F**	1	15	1902	23.8	0.11	31.6
Doublet **E**	6	90	11124	139.1	2.22	665.3
Triplet **F**	1	15	1200	15.0	0.07	20.4
Triplet **E**	3	54	5172	64.7	0.57	170.7
Inner **F**	1	3	380	4.8	0.20	59.1
Inner **E**	1	3	336	4.2	0.13	37.9
Total	13	180	20114	251.4	3.28	985.0
2 Sides	208	2880	321824	4022.8	52.54	15760.6

LDB = Local DAQ Block, SB = Slave Board, Chan. = Channels
Raw = Raw Data, Cmp = Compressed Data
F = Forward, **E** = Endcap

表 4.4： TGC のオクタントあたりのデータ量。Slave Board の数やチャネル数なども同時に示す。データレートは 100 kHz のレベル 1 トリガーを仮定している。圧縮データのデータ量は 1 イベントを 24 ビットとして計算している。最下段の 2 Sides は TGC 全体での値。

4.3.4 ハードウェアの選択

複雑な論理回路を実現可能なデバイスとしては ASIC や FPGA が考えられる。ASIC は容量が大きく動作速度も早いが、一旦作った論理を変更することはできない。また、カスタム化されているので製作には時間と手間と費用がかかるが、大量生産には向いている。一方、FPGA は動作速度、容量に問題点があるが、自由に論理を書き換える柔軟性がある。費用の面では、およそ 1000 個以上の製作した場合には ASIC の方が安くなる。

なお、ASIC の中にも、ゲートの位置まで設計できるフルカスタム IC とゲートの位置が固定されているゲートアレーがある。前者はアナログ回路の設計が可能であるが、後者は主にデジタル回路の設計が中心となる。

ATLAS の実験室内に設置するモジュールについては耐放射線性が要求される。耐放射線性は製造プロセスによって異なるが、一般的には ASIC の方が FPGA よりも耐放射線性に優れている。しかし、最近は耐放射線性のよい FPGA も登場しはじめていて、実験室内でも FPGA を用いることは可能である。

まず、Slave Board について考えてみる。Slave Board は TGC 全体で 2880 枚必要である。また、Slave Board の設置場所は TGC のホイール上で放射線の影響を考える必要がある。さらに、読み出し系には L1A の信号が来るまで 2.5 μs の間全てのヒットデータを保持するためのレベル 1 バッファが必要であるが、これは Slave Board に実装しなければならない。そうでなければ、40 MHz で生成されるヒットデータの全てを Star Switch に送信しなければならないからである。レベル 1 バッファは単なるシフトレジスタであるが、必要なフリップフロップが 10 k 程度とかなり大きな回路になってしまう。それゆえ、Slave Board の回路規模はかなり大きなものとなってしまう。

以上のことを考えると、Slave Board は ASIC を用いて開発するのが適当である。Slave Board には基本的にはデジタル回路しか存在しないので、ゲートアレーを用いることとする。Slave Board には、既に述べたように 5 種類が存在するが（表 3.5）、共通の ASIC を 1 種類用意して切替えて用いることとする。

これに対して、Star Switch の必要数は 208 である。また、Star Switch の設置場所は TGC を支えるホイールの最も外側の縁の部分で Slave Board よりも放射線量は少ない。そこで、柔軟性も考えて Star Switch は FPGA を用いて開発する予定である。

4.3.5 ATLAS フロントエンドエレクトロニクスとしての要求

すでに述べたように、読み出しのスキームとしては Slave Board から Star Switch を経て USA15 内の LDB にデータを送るのであるが、各モジュールの仕様を考える前に読み出し系に求められているその他の詳細な要求を示す。ATLAS の FE（Front-end）のエレクトロニクスが ATLAS 全体のデータ収集の枠組みの中で設計されるようにするために、FE エレクトロニクスには様々な基準や要求がある [10, 11]。これらのうち TGC に読み出し系に関連するものを以下に挙げる。

- 2.5 μs の LVL1 トリガーの latency の間、すなわち L1A 信号が来るまでの間、パイプラインメモリ（以下レベル 1 バッファと呼ぶ）にデータを保持しなければならない。レベル 1 バッファの深さは可変でなければならない。（但し、L1A は TTC で遅らせて発生させることも可能である。）

- L1A によって選択されたイベントについては、データはデランダマイザに保持した後、ROD を経て ROB に読み出さなければならない。このとき、前後のバンチのデータについても記憶及び読み出しが可能でなければならない。

- デランダマイザのデッドタイムは、L1A のレートが 75 kHz のとき 1% 以下、100 kHz の時 6% 以下でなければならない。ここでいうデッドタイムとは、デランダマイザが溢れたことによるデータの

損失とデータの収集が不可能な時間を指す。L1A 発生後の 4 バンチの間は、新しい L1A 信号の生成は禁止されるがこれもデッドタイムとして扱われる。

- 全てのデータには 24 ビットの L1ID、12 ビットの BCID をつける必要がある。但し、上記のビット数は ROD での要求であって、FE においてこれ以下のビット幅のタグをつけ、その後の段階で 24 ビット、12 ビットに変換することは許されている。
- ROB までのデータの転送は 100 μs 以下である。
- データは ROB に送るまでにゼロサプレス（データ圧縮）を行わなければならない。
- デランダマイザが溢れた場合には BUSY 信号を出す。
- 読み出しの順番はイベント順でなければならない。

L1ID と BCID について少し補足しておく。これらの値は TTC などから与えられるわけではない。TTC からは L1A, ECR（Event Counter Reset）, BCR（Bunch Counter Reset） の信号が供給されるだけである。従って、フロントエンドでは常に BCID や L1ID の値を数え続けていなければならない。

次の要求はパラメータ等に関連するものである。

- ソフトウェア的にパラメータの設定が可能でなければならない。
- パラメータは読み出し可能でなければならない。
- コントロールの経路は、読み出しの経路と独立でなければならない。
- タイミングに関連するパラメータ（LVL1 バッファの深さなど ）は設定が可能でなければならない。
- ROB に送るデータの種類（前後のバンチのデータを送るかなど ）は設定可能でなければならない。

これらの要求から、読み出し系に必要な機能の最も基本的なものをまとめると図 4.7 のようになる。ただし、実際には読み出すべきチャネルは極めて多く、また検出器からエレクトロニクス室までは距離があることはすでに述べた通りである。

4.3.6 制御プロトコルの選定

4.3.5 で触れたように、パラメータの設定と読み出しができることが FE エレクトロニクスに要求されている。パラメータの設定以外にも、ランコントロールやテスト、システム診断など、いろいろな制御が必要になって来る。

ATLAS では DCS が検出器の制御を統一的に行う。現時点では、TGC の FE エレクトロニクスの制御をどのような経路で行うのかは確定していないが、次のような経路が考えられる。

- Service Patch Panel 経由。Service Patch Panel は Patch Panel と Slave Board を含む PS-Pack（図 3.10 ）の中にある、TTC や DCS とのインターフェース用のボードである。ここでは CAN というプロトコルを用いてスローコントロールやモニターを行う。ASD ボードの threshold 電圧の設定などはこの経路を用いる予定である。原理的には、全ての Slave Board のパラメータ設定などをこの経路だけで行うように設計することも不可能ではない。
- Star Switch 経由。Star Switch から Slave Board や Patch Panel にコマンドを分配するという経路も可能である。この際、Star Switch に DCS とのインターフェースを設ける方法と、さらに上流（Local DAQ Master など ）に DCS とのインターフェースを設けて Star Switch と Local DAQ Master の間の接続を双方向にするという方法が考えられる。

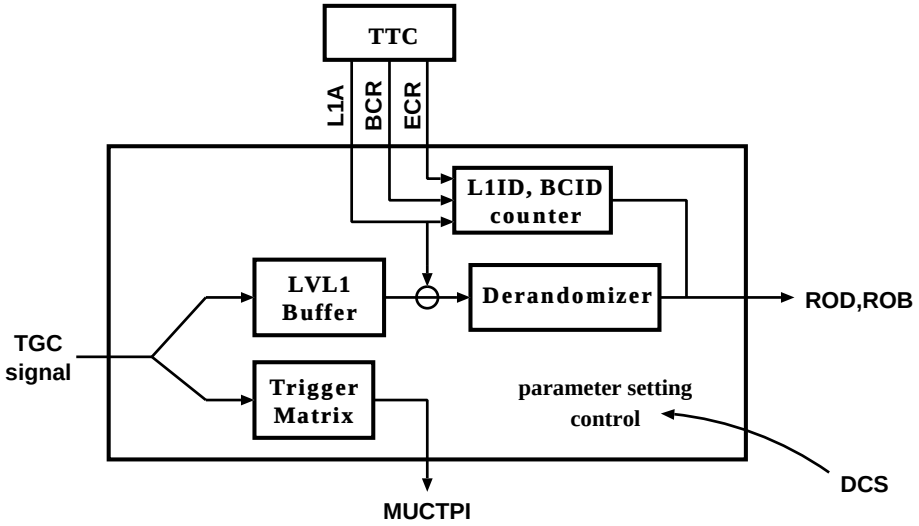

図 4.7： TGC 読み出し系に必要な機能

前者の CAN による制御の経路は、十分な帯域が確保できるかに疑問が残る。また、ランコントロールのように速い制御とパラメータ設定のように遅い制御を、同じ経路で行うのが適切かどうかは定かではない。そこで、前者の経路は ASD の threshold 電圧の設定や温度等のモニターに用いることとし、その他のパラメータ設定やランコントロールは Star Switch 経由で行うことを提案している。このことを前提として今回の設計を行った。

DCS とのインターフェースをどこに置くかに関わらず、Star Switch からの制御信号は Slave Board に分配されなければならない。このなかでパラメータ設定やシステム診断は Slave Board だけでなく Patch Panel に対しても行う必要がある。これに対して、ランコントロールのような速い制御は Slave Board に対してのみ必要である。そこで、制御のための系統としては速い制御用と遅い制御用の 2 つを用意することにした。

速い制御の場合には Slave Board に対して必要な命令を送るだけでよい。しかも、Star Switch から Slave Board は 1 対 1 の接続があるだけなので、標的のデバイスの指定などは必要ない。それゆえ、速い制御についてはプライベートにプロトコルを定義することにする。この場合は、単に Star Switch から適当な長さのコマンドをシリアルに Slave Board に転送するだけである。Slave Board から Star Switch へはデータを送るための接続があり、これを LS-Link（Local Slave Link）と呼んでいるが、以後速い制御用の Star Switch から Slave Board への接続も LS-Link と呼ぶことにする。

これに対し、遅い制御の方は書き込みと読み出しが出来なければならないし、経路中の複数のデバイスのなかから制御を行うデバイスを指定しなければならない。このため、制御のためのプロトコルはそれなりに複雑になる。そこで、遅い制御については規格化されたプロトコルを用いることとした。

遅い制御用のプロトコルとして考えられたのは、以下のようなものである。

JTAG 基板や IC のテストのためのプロトコルで、4 本のテスト用の信号線を用いてシステム診断をすることが可能である。デバイスをシリアルに接続することによって複数の IC を診断することができる。プライベートな命令（Instruction）を定義することによって、パラメータの設定などを行うことも可能である。

I^2C フィリップス社の開発したプロトコルで、SDA（Serial Data）と SCL（Serial Clock）2 本のシリア

ルバスを介して IC 間の情報伝達を行う。バス上の複数のマスターが存在可能で、バス調停の機能も規定されている。TTCrx が I^2C のプロトコルを解読できる。

I^2C は CAN の小型版のようなものであるが、TGC の遅い制御においてはバス調停などは不要であり、プロトコルも JTAG より若干複雑そうである。また、JTAG は個別の IC に JTAG 用の回路を実装するだけで特に複雑なセットアップを用意しなくてもテストや診断を行うことが可能である。それゆえ、TGC の遅い制御には JTAG を用いることにした。

JTAG についての詳細は Appendix B に記した。ここでは、TGC エレクトロニクスでの JTAG の経路に触れておく。JTAG によってパラメータの設定等が必要なのは読み出し系では、Slave Board と Star Switch だけであるが、その他にも Patch Panel や High-p_T Board もパラメータ設定や診断のために JTAG による制御がなければならない。Patch Panel のためには、JTAG の信号線を Slave Board ASIC と Patch Panel ASIC にシリアルに接続する必要がある。ところが、1 つの Slave Board ASIC に対して最高 8 つの Patch Panel ASIC が接続されている。これらを全てをシリアルに接続すると、Patch Panel ASIC のうちの 1 つが故障しただけで、他の Patch Panel ASIC と Slave Board ASIC はパラメータの設定が不可能になってしまう。そこで、図 4.8 のように、Patch Panel ASIC は 2 個だけをシリアルに接続し、8 個のうちどの 2 個を選ぶかは Slave Board ASIC で選択出来るようにする。こうすると、Patch Panel ASIC のうちの 1 つが故障しても使用不能になる Patch Panel ASIC は 2 つだけになり、被害が少なくてすむ。その分、JTAG の経路を Slave Board ASIC で選択しなければならないので、Slave Board ASIC のロジックが少しだけ複雑になる。

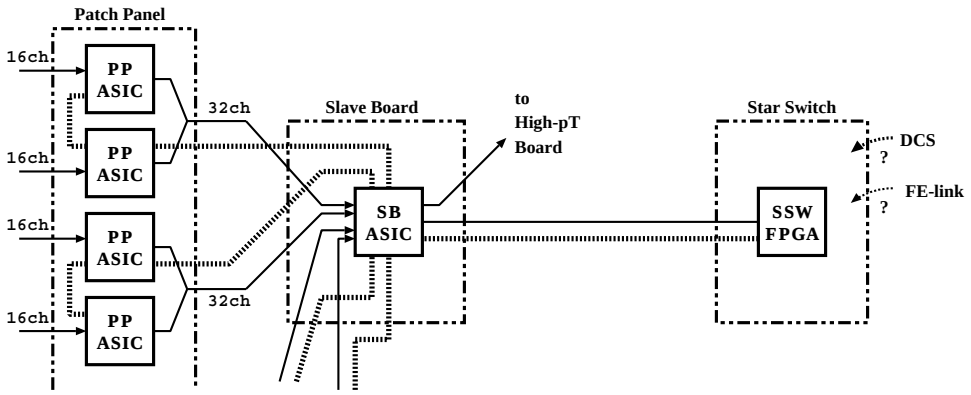

図 4.8： TGC エレクトロニクスにおける JTAG の経路。点線が JTAG 信号の流れで、実線がデータの流れ。本来は JTAG 信号線は一直線にシリアル接続するのであるが、Patch Panel の故障に備えて Slave Board で分配することとする。

4.4 Slave Board の仕様

4.4.1 Slave Board の概要

今回、Slave Board に必要な機能を分析し、その仕様を定めた。以下に Slave Board の仕様について述べる。なお、Slave Board の仕様についてここで述べなかったことについては Appendix C で述べる。また、この仕様に基づいて Slave Board ASIC の HDL 記述を行った。これについても Appendix C で述べる。

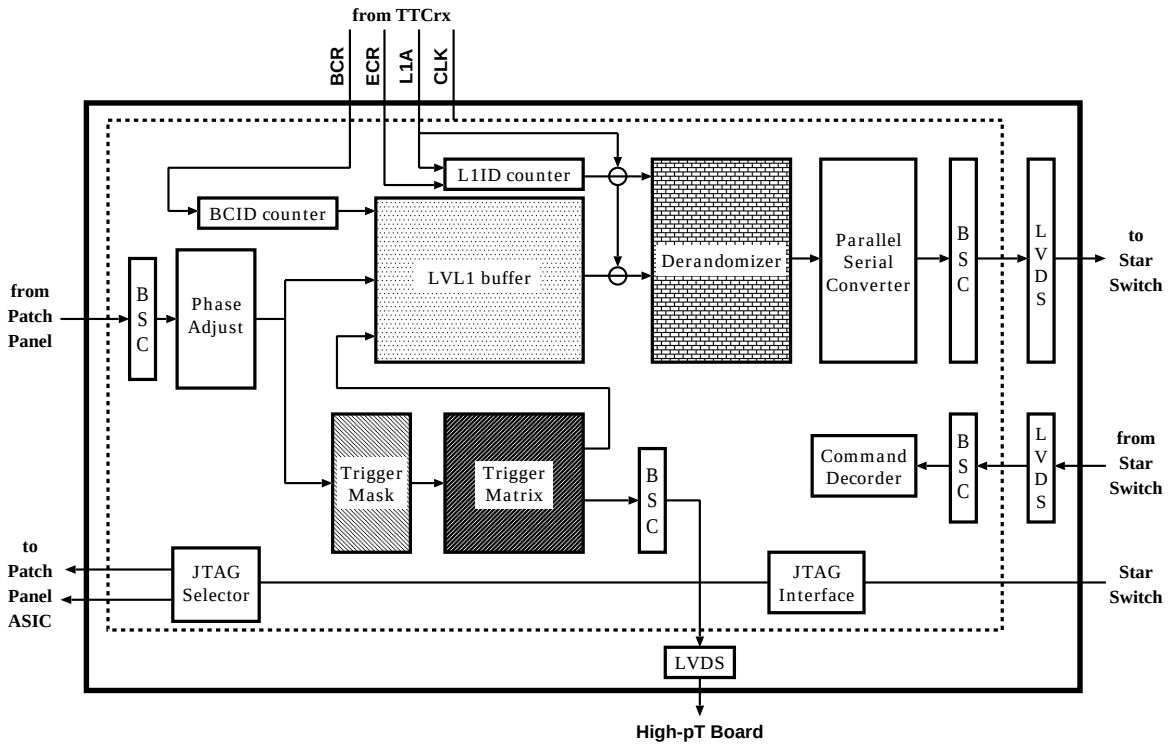

図 4.9： Slave Board の機能のブロック図。BSC は Boundary Scan Cell。TGC のヒットデータは L1A の判定が行われるまでの間レベル 1 バッファに保持される。L1A で採用されたデータはデランダマイザにコピーされ、シリアル化の後 LVDS に変換されて Star Switch に送信される。図の中の点線の部分（TTL-LVDS 変換以外）は ASIC（ゲートアレー）で実現される。

図 4.9 に Slave Board の機能のブロック図を示す。以下、読み出し系に関連する部分について考える。Patch Panel から入力された TGC のヒットデータは位相調節（Phase Adjust）を経た後、レベル 1 バッファに入力される。L1A の判定が行われるまでの約 2.5 μs の間、データはここで保持される。L1A で採用されたデータはデランダマイザと呼ばれるバッファにコピーされる。これらのデータはシリアル化され、LVDS レベルに変換された後、LS-Link を通じて Star Switch に送信される。

Star Switch に送信すべきデータとしては、ヒットデータの他に L1ID、BCID とトリガー行列出力がある。L1ID や BCID は、TTC から供給される L1A, ECR, BCR 信号をもとに Slave Board 内部で計数する必要がある。すでに述べたように、最終的には L1ID は 24 ビット、BCID は 12 ビットのビット幅が要求されているが、Slave Board の段階ではこれらのタグのビット幅は 8 ビットとして、後の段階で各々 24 ビット 12 ビットに変換することとする。なお、BCID タグはヒットデータと対応するものなので、ヒットデータと同様にレベル 1 バッファを経る必要がある。トリガー行列出力は、従来は High-p_{T} Board で入力信号として読み出す予定であったが、High-p_{T} Board からの読み出しはやめる方針であるので、かわりに Slave Board から読み出すこととする。

この他、Slave Board には Star Switch からのコマンドを解読する部分や、JTAG による遅い制御を処理する部分がある。

これらの回路は、基本的にはデジタル回路であり、ゲートアレーを用いて開発することはすでに述べたが、TTL レベルと LVDS レベルの変換だけはアナログ回路である。そこで、この変換だけはゲートアレー

の外で行うこととする（図 4.9）。

4.4.2 Slave Board の機能ブロック

以下、図 4.9 に示された各機能ブロックの機能を簡単に説明する。

位相調節器（Phase Adjust） Patch Panel から入力されるデータはバンチの識別が行われているが、これらは相対的なものである。例えば、ダブレット Slave Board の場合にはインナーダブレットからの信号とピボットダブレットからの信号が Slave Board に到達した時には 1 バンチ分位相がずれているということが起こりうる。検出器から Slave Board までのケーブルの長さが一定でないからである。また、これらの入力信号と TTC から供給されるクロックとの間の相対位相も保証されていない。そこで、Phase Adjust では入力信号を必要に応じて半クロック単位で遅延させ、位相を揃える。遅延はチェンバーの層ごとに設定することが可能で、設定は JTAG によって行なう。

Phase Adjust は図 4.10 のような回路で実現できる。なお、入力信号は Slave Board 内でラッチを行う必要があるので、Phase Adjust 内に含める。

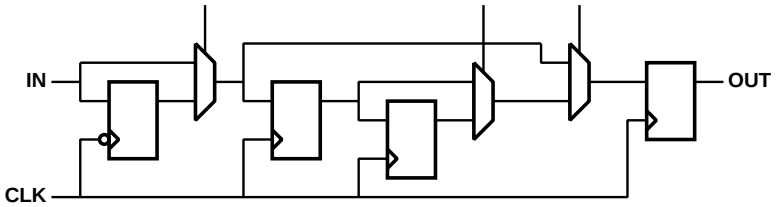

図 4.10： Phase Adjust 回路。入力信号を半クロック単位で調節する。最終段のフリップフロップは入力信号を Slave Board 内で一旦ラッチするためのものである。

トリガーマスク Trigger Matrix の入力直前にあるマスク。較正や、チェンバーの重なりあった部分の不要なチャネル、不良チャネルを取り除くのに使う。マスクは各信号ごとに設定が可能で、マスクをかけるとトリガー行列には常に 0 が入力されるようになる。設定は JTAG で行い、読み出しは JTAG および LS-Link の両方で行う。

トリガー行列 Slave Board の種類によって 5 種のトリガー行列が存在する。トリガー行列については 3.3.3 を参照のこと。

レベル 1 バッファ L1A 信号が得られるまでの間データを保持しているためのパイプラインメモリ。LVL1 の latency である 2.5 μs の間データを保持しなければならないため、深さはおよそ 100 段である。保持するデータは、TGC のヒットデータの他に、BCID とトリガー行列出力がある。ヒットデータと TTC からの信号は別の経路から来ているために伝播遅延の時間は同じでない。また、トリガー行列出力はトリガー行列でクロックを消費しているため、やはりヒットデータとはクロックがずれている。従って、これらのバッファの深さは各々独立にパラメータ設定可能でなければならない。いまのところ、1 段から 128 段まで 1 段ごとに設定可能という仕様になっている。パラメータの読み書きは JTAG で行なう。

デランダマイザ　レベル 1 で採用されたイベントを保持しておく FIFO。保持するデータは、L1ID、BCID、TGC ヒット情報、トリガー出力である。ヒット情報は前後のバンチについても必要である。Appendix H で述べるように、ATLAS FE エレクトロニクスに課せられた条件を満たすためにはバッファの深さは最低 5 段必要である。現在のところ少し余裕をみて 8 段で設計を行っている。

Parallel Serial Converter　デランダマイザに保持しておいたデータは、LS-Link を通して Star Switch に送られる。そのため、データは一旦ここでシリアル化される。出力されるフォーマットについては 4.6 で述べる（表 4.6 ）。

コマンド解読部（Command Decorder）　Star Switch から LS-Link を通して送られるコマンドを解読する部分。コマンドは 4 ビット程度の固定長のものを想定している。LS-Link を通じて送られる命令は、ランの開始終了といったランの制御に関連するものである。LS-Link については、4.6 述べる。

TTL - LVDS 変換　Slave Board からの出力信号は、High-p_{T} Board に行くトリガー出力も、Star Switch に行くデータも、ともに LVDS レベルへの変換が必要である。

JTAG インターフェース　Slave Board では JTAG によるバウンダリスキャン、パラメータの設定と読み出しを行う。そのための JTAG のプロトコルを解読する部分である。JTAG については Appendix B 参照。

BSC（Boundary Scan Cell）　JTAG によるバウンダリスキャンを行うためのシフトレジスタ。入出力ピンの値を読み出したり、入出力ピンに特定の値を書き込んだりすることが可能である。これについても、Appendix B 参照。

JTAG セレクタ　すでに述べたように、JTAG のシリアル信号線の経路は Slave Board で 4 つに分岐して Patch Panel に及ぶ（図 4.8 ）。そのため経路を切替えるためのセレクタが必要である。セレクタ自体も JTAG で制御する。

4.5　Star Switch の仕様

4.5.1　Star Switch の機能

Star Switch は Slave Board から Local DAQ Master へ転送されるデータを中継するモジュールである。1 台の Star Switch は 20 台程度の Slave Board からデータを受け取り、これらのデータをまとめ、適当なフォーマットに変換する。この際、データの BCID や L1ID を確認し、Slave Board 間で矛盾がないか調べる必要がある。その後、これらのデータを光信号に変換し、FE-Link（Front-end Link）を経て Local DAQ Master に送信する。この間、データ量を減らすために Star Switch 内でデータ圧縮を行なう。

一方、Star Switch は DCS 又は FE-Link から制御やパラメータ設定のコマンドを受け取り、これを解釈して Slave Board に分配しなければならない。これらを総合すると Star Switch の機能としては図 4.11 に示したもののようになる。

改めて Star Switch の機能を整理すると、以下のようになる。

- Slave Board から LS-Link を経てデータを受け取る。信号は LVDS レベルなので、まず TTL への変換を行う必要がある。

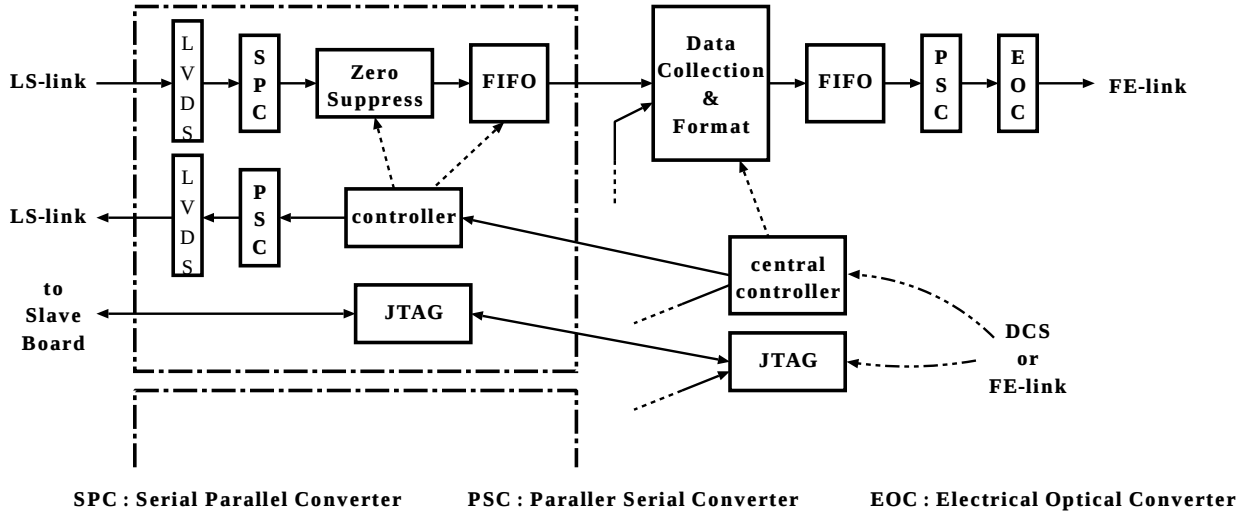

図 4.11： Star Switch の機能。20 個程度の Slave Board からデータを収集し、データの圧縮を行う。これらのデータをまとめ、適当なフォーマットで光ファイバーで Local DAQ Master に送信する。

- SPC（Serial Parallel Converter）で LS-Link のシリアルデータを復元を行う。
- データの圧縮を行う。
- 接続されている Slave Board からのデータを収集し、整形する。
- 異なる Slave Board からのデータの BCID L1ID などを比較、検査する。
- PSC（Parallel Serial Converter）でシリアルに変換後、さらに EOC（Electrical Optical Converter）で光に変換する。FE-Link を通じてデータを Local DAQ Master に送る。
- 必要に応じて FIFO を用意しなければならない。
- 命令をうけとり、各 Slave Board に分配する。
- JTAG 命令を受け取り、または生成し、Slave Board に分配する。

4.5.2 Star Switch の構成

Star Switch はモジュールによって接続される Slave Board の数が異なる。また 1 つの Star Switch にはおよそ 20 個の Slave Board が接続される。後で述べるように、Star Switch と Slave Board 間の接続には 20 ピンのコネクタを使う予定なので、1 つのモジュールで Star Switch を作ることは、コネクタの配置を考えても困難である。そこで、Star Switch 自体をバックプレーンと以下のようなのモジュールで構成する方法を考えている。

レシーバモジュール Slave Board との接続部分（LS-Link, JTAG）。1 つのモジュールが数個の Slave Board に接続される。

データ転送モジュール データをレシーバモジュールから収集し、整形し、FE-Link を通じて Local DAQ Master に送る。

制御モジュール 制御を行なう。データ転送モジュールに含まれてもよい。

バックプレーンとしては、コントロール用とデータ用のバスを用意する。概略図を図 4.12 に示す。レシーバが受け取ったデータはデータバスを介してデータ転送モジュールに送られる。データ転送モジュールや、制御モジュールに関しては現時点では仕様の詳細は定まっていない。次節では、レシーバモジュールについて述べる。

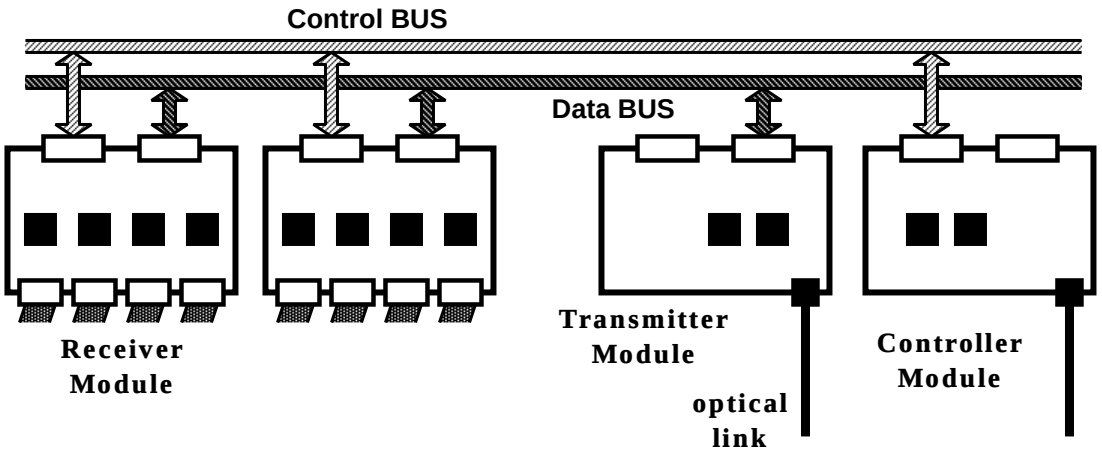

図 4.12： Star Switch の概略図。Star Switch は Slave Board からデータを受け取るレシーバモジュールと Local DAQ Master に光ファイバーを介してデータを転送するデータ転送モジュール、DCS などからコントロールの命令を受け取る制御モジュールからなる。これらはバックプレーンのバスラインを介して通信を行う。

4.5.3 レシーバモジュールの設計

レシーバモジュールのブロック図を図 4.13 に示す。各々のブロックの機能は以下のとおりである。

LVDS-TTL 変換 LS-Link は LVDS レベルであるから、入出力信号に対しては LVDS TTL 間の変換が必要である。

SPC（Serial Parallel Converter） Slave Board からのデータはシリアル化されて送信されてくる。したがって、レシーバモジュールではこれを復元するする機能が必要となる。

ゼロサプレス ゼロサプレスとはデータの圧縮方法の一つである。TGC のヒットレートは低いので、ヒットデータは通常はほとんどのチャネルで 0 である。そこで、図 4.14 (a) に示されるようにヒットがあるチャネルのアドレスを並べれば、データ量を大幅に減らすことが出来る。しかし、TGC のヒットの場合には隣のチャネルをならすようなヒットは頻繁におこるので、ヒットデータを適当な大きさのセルに区切り、ヒットのあるセルのアドレスとそのセルのヒット情報に変換するという方法がさらに効率的である。この方法によるゼロサプレスを図 4.14 (b) に示す。現在のところ、TGC のヒットデータの圧縮には後者の方法でセルの大きさを 8 ビットにして行う予定である。

デランダマイザ（FIFO） ゼロサプレスの出力結果は、データバスを介して読み出されるまでの間バッファに蓄えておく必要がある。デランダマイザはこのためのものである。ゼロサプレスの出力結果はビッ

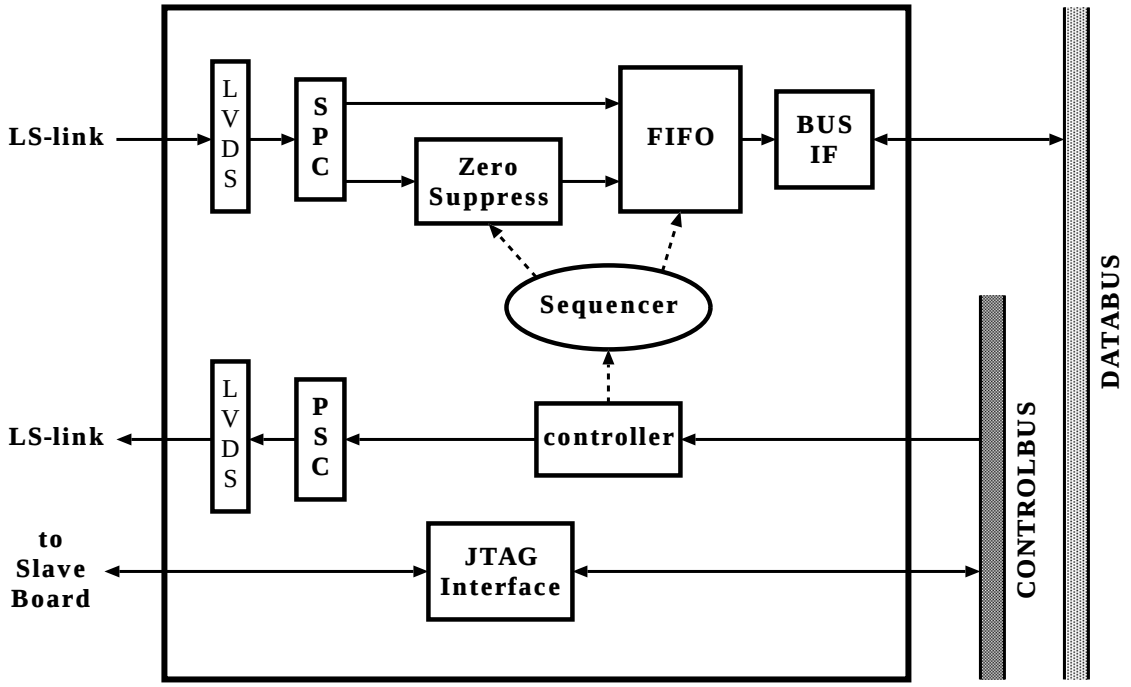

図 4.13： レシーバモジュールのブロック図

ト幅が可変なので可変長の FIFO が必要であるが、データバスを介して読み出される際に固定長のデータに分割しなければならないので、ここでも固定長の FIFO を用意して、数回に分けて書き込むという方式をとる。

バスインターフェース（BUSIF） データバスとのインターフェース部分である。現時点では、データバスのプロトコルが定まっていないので詳しい仕様は定まっていない。

その他 この他に、コントロールバスを介して制御用の命令を受け取り、これを Slave Board に分配する機能が必要である。

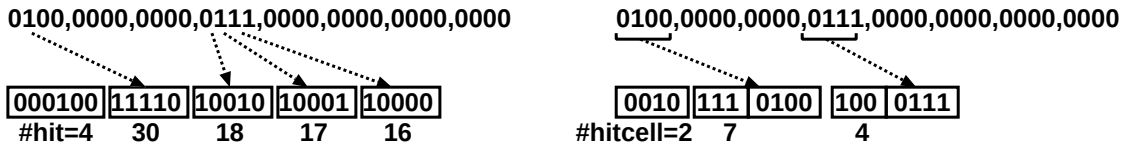

図 4.14： ゼロサプレスの論理。(a) はヒット位置をアドレスに変換するだけの単純な論理。(b) はセル（この場合は 4 ビット幅）に区切って、セル内にヒットがあるときにセルのアドレスとヒット情報に変換する論理。いずれの場合にも、アドレスは LSB（右側）を 0 に選んでいる。ゼロサプレス出力には、ヒットのあるチャネル又はセル数も必要である。

レシーバモジュールのこれらの機能のなかで、アナログの回路は TTL LVDS 間の変換だけである。既に述べたように、デジタル回路の部分は柔軟性を考えて FPGA を用いて開発を行う予定である。それ故、

レシーバモジュールは基本的には図 4.15 に示されるように、TTL LVDS 変換用のデバイスと FPGA から構成される[4]。図は 1 つのレシーバモジュールが、4 つの Slave Board と接続されている場合である。図では 1 つの FPGA が 1 つの Slave Board からのデータを処理しているが、1 つの FPGA が複数の Slave Board を担当するようにすると、FPGA の数を減らすことが出来る。

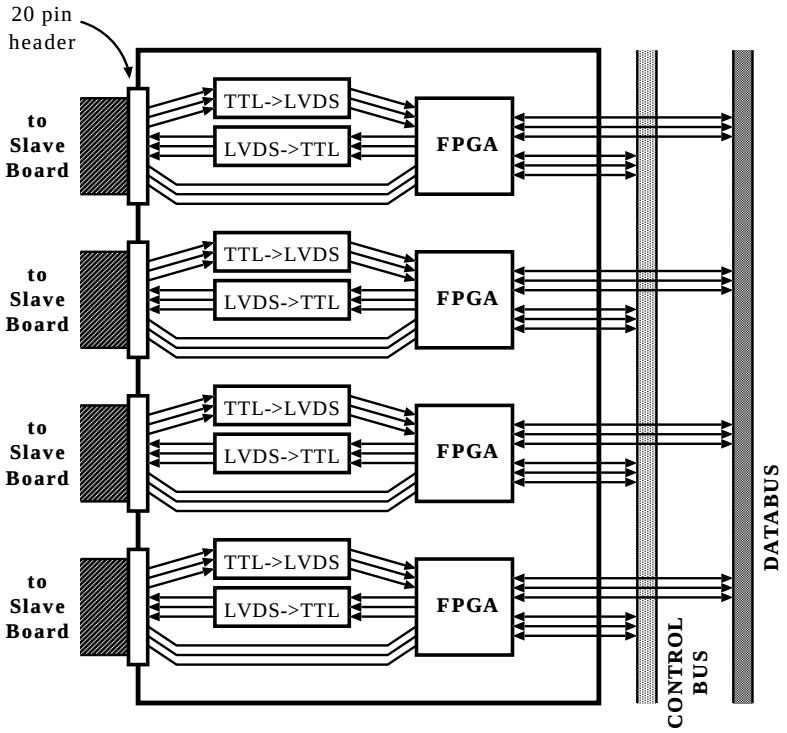

図 4.15： レシーバモジュールの構成。1 つのレシーバモジュールが 4 つの Slave Board からのデータを受け取る場合の図である。基本的には LVDS-TTL の変換以外は FPGA での実装を行う。

4.6 Local Slave Link

4.6.1 LS-Link の概要

LS-Link（Local Slave Link）は Slave Board と Star Switch の間の接続である。この間を行き来する必要がある情報としては以下のものがある。

- TGC から読み出すべきデータ。ヒットデータは Slave Board で BCID や L1ID を付加された後、LS-Link を経て Star Switch に送られる。
- Star Switch 経由で行われる速い制御。Star Switch から Slave Board へコマンドを送る。
- Star Switch 経由で行われる遅い制御。遅い制御用のプロトコルとしては、JTAG を採用している。このための Slave Board と Star Switch 間の接続としては、JTAG 用の 5 本の信号線（TCK, TMS, TDI, TDO, TRST*）[5] が必要になる。

[4]但し、Xilinx 社 FPGA の Virtex シリーズでは、LVDS レベルでの入出力が可能なものが利用可能になりつつある。このようなデバイスを用ることができれば、外付けの TTL LVDS 変換は不要になる。

[5]TRST* を用いなければ 4 本になる。

Slave Board から Star Switch へと送るデータはデータ量が多いので、40 MHz の LHC クロックと同期して送る必要がある。また、LHC のクロック単位で制御を行うことを考えると、速い制御用の接続も LHC クロックと同期している必要がある。しかし、Slave Board と Star Switch の間は約 10 m あるので、単にケーブルで接続しただけ（TTL レベル）では、40 MHz で信号を送ることは不可能である。そこで、一旦 LVDS レベルに変換することとする。一方、JTAG 用の信号は TCK で指定したタイミングで送る。こちらは TCK を十分に小さくすれば、TTL レベルのままの接続でも制御は可能である。

以後、単に LS-Link と言った場合、LVDS に変換して 40 MHz でデータの伝達を行う Slave Board と Star Switch 間の接続（JTAG 以外の接続）を意味することとする。

40 MHz の LHC クロックと同期して情報を伝達するため、LS-Link のプロトコルを次のようなものと定める（図 4.16)。LS-Link は基本的には一方向当たり CLOCK, DATA, SYNC の 3 本の信号線からなっている。図のように受信側は CLOCK の立ち下がりでデータを取り込むようにする。SYNC は通信を制御するための信号線で、SYNC = 0, DATA = 0 が送信開始、SYNC = 0, DATA = 1 が送信終了を表す。これらのコマンドを送るとき以外は SYNC = 1 に保っておく。

データを取り込む CLOCK エッジの向きは立ち上がり側を利用しても、問題はない。SYNC については、SYNC = 1 の時に開始又は終了を意味するように定めても大きな支障はないが、LVDS の規格からリンクのエラー発生時は信号線は 1 になるので、開始、終了のコマンドには SYNC = 0 を割り当てることとする。

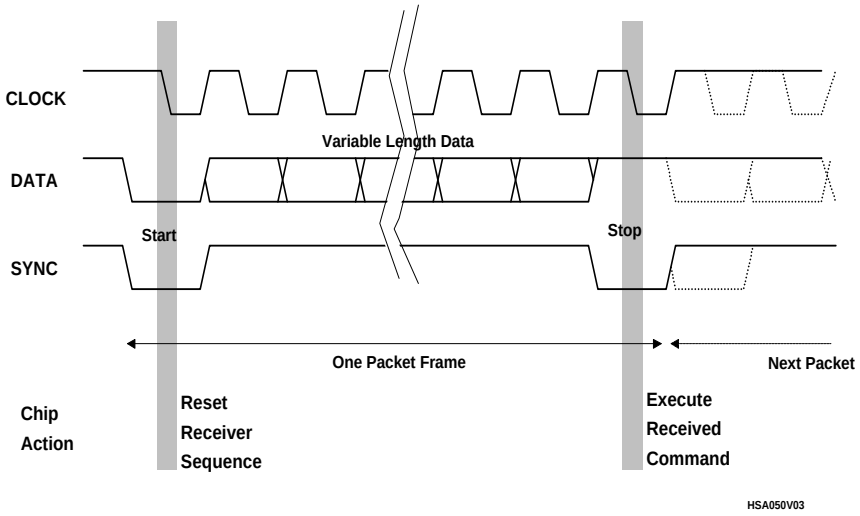

図 4.16： LS-Link のプロトコル

4.6.2 Slave Board から Star Switch に向かう LS-Link

Slave Board から Star Switch に向かう接続には、TGC ヒット等のデータが流される。このデータの中には TGC のヒットデータの他に、L1ID や BCID、Slave Board のアドレス等が入らなければならない。さらに、ヒットデータはレベル 1 で採用されたバンチだけでなく、その前後のバンチを含めた、計 3 バンチ分のデータについても収集が可能でなければならない。これらを考慮した上で、Slave Board から Star Switch に伝えるべき 1 イベントあたりのデータ量を見積もると表 4.5 のようになる。

一方、CLOCK は 40 MHz なので、L1A が 75 kHz で来た場合のデータサイズの上限は 533、100 kHz の場合は 400 となる。これから、100 kHz で L1A が来た場合は 3 バンチ分のデータを送ることは不可能

send mode	1BC per event	2 BC per event	3 BC per event
start code	1	1	1
header	8	8	8
address	8	8	8
BCID	8	8	8
L1ID	8	8	8
hitdata	128	256	384
trigger output	18	18	8
trailer	8	8	8
terminator	1	1	1
total	188	316	444

表 4.5： LS-Link の 1 イベントあたりのデータ量
Slave Board はダブレットワイヤ用を想定している。

ということになってしまう。

この問題を解決するために、Slave Board から Star Switch への接続に関しては、DATA 線を 2 本用意することとする。そして、Slave Board から Star Switch へは、データを表 4.6 のようなフォーマットで送信することとする。括弧内の数字はビット数である。

Header (8)	Address(8)	BCID(8)	L1ID(8)	Data(128)	Status(128)	Checksum
Header (8)	Address(8)	BCID(8)	L1ID(8)	Prev. Data(128)	Next. Data(128)	Checksum

表 4.6： Slave Board から Star Switch に向かう LS-Link のフォーマット

4.6.3 Star Switch から Slave Board に向かう LS-Link

Star Switch から Slave Board に向かう LS-Link は速い制御を担っている。そのため、制御の際にだけコマンドを送信すれば良いので、DATA 線は 1 本で十分である。単純化するため、コマンドは固定長とする。速い制御としては

- ハードウェアやカウンタの初期化
- ランの開始や終了
- 通常のデータ以外のパラメータ（トリガーマスクの値など）の読み出し。

などが必要であるがこれらのコマンドを表すためにはとりあえず 4 ビットあれば十分である。現在のところ、各コマンドと対応する命令コードは表 4.7 のように定めている。

4.6.4 LS-Link のまとめ

以上をまとめると、Slave Board と Star Switch 間の接続は以下の通りになる。

Command	Code	Command	Code
Initialize	0000		
Reset	0001	Read Parameter	0101
Standby	0010	Read Counter	0110
Run	0011	Read Status	0111

表 4.7： LS-Link の 4 ビットのコマンド

- Slave Board から Star Switch へは CLOCK, SYNC, DATA0, DATA1 の 4 つの信号で、TGC のヒットデータなどを送信する。レベルは LVDS である。フォーマットは表 4.6 参照。
- Star Switch から Slave Board へは CLOCK, SYNC, DATA の 3 つの信号で、4 ビット固定長コマンドを送信する。レベルは LVDS である。
- 遅い制御用に JTAG の 4 ないし 5 本の信号線を接続する。LVDS への変換は行わない。

LVDS では 1 つの信号を送るために 2 本の信号線が必要である。その結果、Slave Board と Star Switch 間は 19 本の信号線が必要になるので、20 ピンのツイストペアケーブルなどを用いることとなる。

第 5 章

プロトタイプの製作と検証

5.1 プロトタイプモジュールの製作

5.1.1 pt3 モジュールの製作

最終的なモジュールの開発に先立って、プロトタイプ用のモジュールの作成を行い、設計に問題がないか必要な性能を満たすことが出来るかを調べなければならない。特に ATLAS 実験のような大規模な実験においては、大量生産の前に何段階ものプロトタイプの製作を行い、入念に検証を重ねる必要がある。

今回、TGC の読み出し系の最初のプロトタイプとして、FPGA を用いたプロトタイプモジュールの製作を行い、これを pt3 モジュールと命名した。図 5.1 に製作した pt3 モジュールの写真をのせる。

pt3 モジュールは Xilinx 社製の FPGA を 4 つ搭載した 6U の VME ボードである。pt3 モジュールの機能ブロック図を図 5.2 に示す。FPGA は Spartan シリーズの XCS40（Plastic Quad Flat Package の 208 ピン）を用いた。VME のコネクタとしては P1 コネクタのみを使用している。そのため可能なアクセスは A24D16 又は A16D16 などに限られる。フロントパネルには 34 ピンのヘッダが二段重ねになったものが 2 個つけられていて、合計 64 本の入出力ピンを形成している。FPGA には VME 経由でロジックを書き込むことが出来るのだが、VME のプロトコルを解読は主に CPLD XC95216 が担っている。

クロックは、モジュールに装着されている 40 MHz のクロックの他、外部クロック用の NIM 端子が 2 つ用意されている。これらのクロックは CPLD に供給されており、CPLD で FPGA のクロックの切り替えが可能になっている。

Spartan シリーズ FPGA XCS40 は内部に 784 の CLB（Configurable Logic Block）を有し、2016 のフリップフロップを有している。また、CLB は RAM として利用可能であり、全ての CLB を RAM として使用したとすると 25 kB の容量がある。Spartan シリーズの FPGA は、容量や動作速度という点ではさほど優れているわけでない。にもかかわらずこのデバイスを選んだのは、Spartan シリーズの FPGA が他の種類の FPGA に比べてかなり安価であるからである。それゆえ、pt3 モジュールを用いて Slave Board ASIC や Star Switch の完全なプロトタイプを実現することはできない。pt3 モジュールの目的は Slave Board ASIC や Star Switch のロジックの確認やスキームの検証、最終的に開発されるモジュールに必要な性能の見積もりである。

Slave Board ASIC に関して言えば、最終的には ASIC を作成するわけであるから当然 ASIC の試作が不可欠である。しかし、ASIC の試作には費用と時間がかかるので、何度も繰り返して行うというわけにはいかない。そこで、ASIC の試作を進める一方、FPGA のような柔軟性の高いデバイスで検証を行う必要がある。

なお FPGA についての詳細は Appendix D で、pt3 モジュールの仕様については Appendix E で述

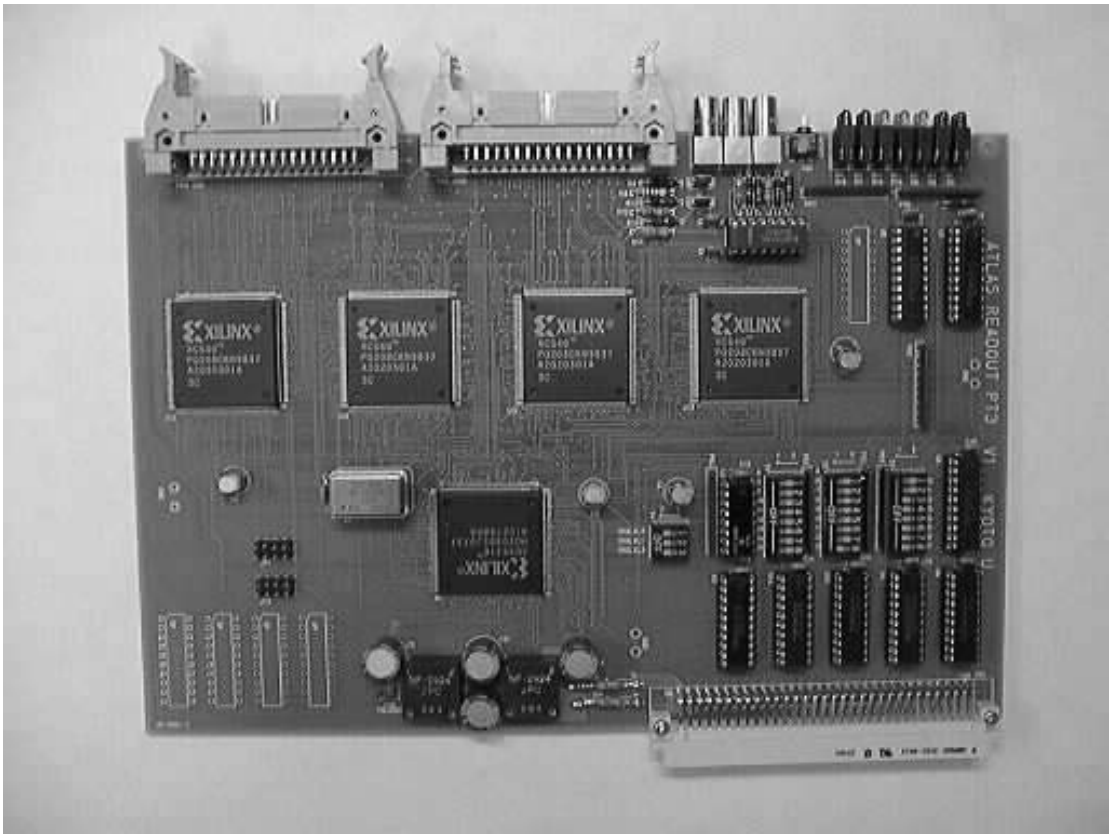

図 5.1： pt3 モジュールの写真。真ん中に 4 つ並んでいるデバイスが Xilinx 社製 Spartan FPGA。その下にあるのが CPLD。

べる。

5.1.2 パターンジェネレータとレコーダ

pt3 モジュールで読み出し系の検証を行う前に、次のようなパターンジェネレータとレコーダを作成し、これらをモジュールの FPGA に書き込みどの程度の周波数で動作するかを調べてみた。なお、これらのロジックを作る時には、CLB を Selected RAM として利用した（Appendix D 参照）。

- 幅 16 ビット、深さ 256 段のパターンジェネレータ。VME からのアクセスで、順番に FPGA 内部のメモリにパターンを書き込む。VME 経由でパターン生成命令を受け取ると、256 クロックの間書き込まれていたパターンを出力する。クリティカルパス（回路内で最も遅延が大きい経路）から算出したこのロジックの動作限界は 40.80 MHz であった。
- 幅 16 ビット、深さ 256 段のパターンレコーダ。16 ビットの入力信号をシフトレジスタ内に取り込む。取り込まれたデータは VME から読み出すことが可能である。クリティカルパスから算出したこのロジックの動作限界は 38.33 MHz であった。

これらのロジックは Selected RAM にシフトレジスタ役割をさせた単純なもので、配線量も少ない。従って、これらのロジックは動作周波数は読み出し系のプロトタイプ用に作成するロジックよりも高速になっている。

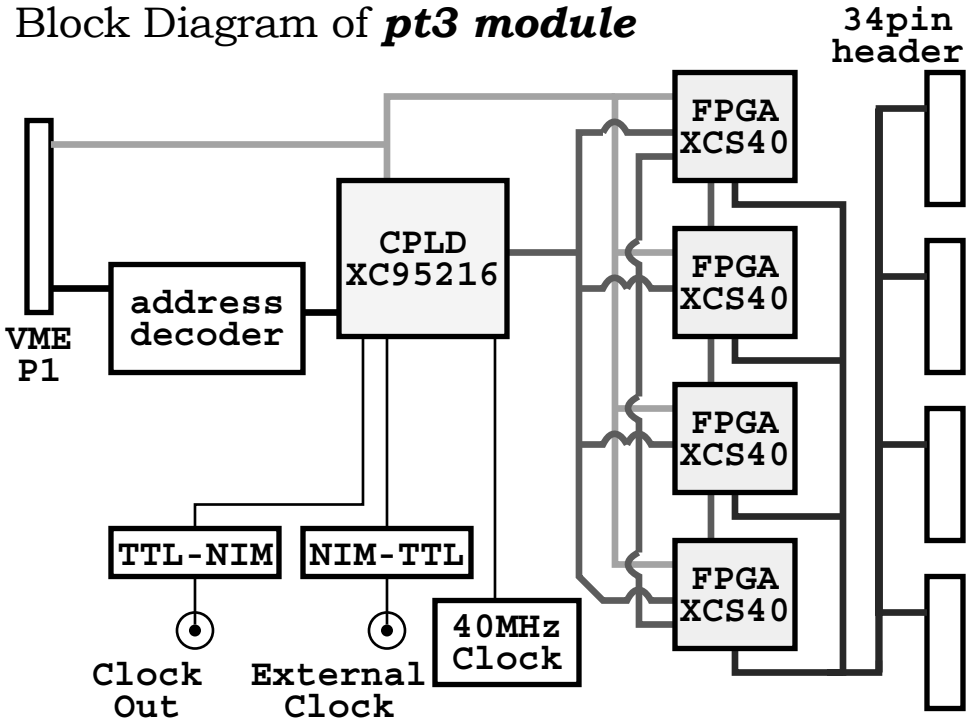

図 5.2： pt3 モジュールのブロック図

実際にこのロジックがどの程度の周波数で動作するか調べるために、これらのロジックをモジュールの 2 つの FPGA に書き込んで、ジェネレータで発生させた 16×256 のパターンをレコーダで取り込み、発生したパターンと取り込んだパターンが一致するかをクロックの周波数を変化させながら測定した。

測定結果を図 5.3 に示す。図の横軸は pt3 モジュールに外部から与えたクロックの周波数、縦軸はその際のエラーレートである。このロジックの場合にはクリティカルパスから得られた動作周波数の限界は 40 MHz 程度であった、実際には 53 MHz まで動作した。

遅延を見積もる際には静的なモデルが用いられている。このモデルでは、論理合成された回路内に含まれる基本ゲート（FPGA の場合には CLB）及び配線ごとに遅延を算出し、これらの合計をフリップフロップ間の遅延としている。ゲートごとの遅延はそのゲートに入力される信号のパターンによっても変化するが、静的なモデルでは典型的な値に安全計数を乗じたものを用いている[1]。このようにして求めた遅延が最大になる経路がクリティカルパスであり、この逆数が動作周波数である。

実際には、入力パターンによって遅延は変化するし、ゲートや配線の間の影響も考える必要がある。この意味で、クリティカルパスによる見積もりは回路内の全ての効果を忠実に再現できているわけではない。また、この値は入力信号の到着時間にある程度のずれがあることを想定して算出されているので、実際の入力信号の到着時間によって動作周波数の限界は変化しうる。したがって、この程度のシミュレーションと実測値のずれは問題がない。逆にいうと、53 MHz まで動くという結果が得られても。使い方によってはこの周波数で動作しないということがあり得る。それゆえ、実際にロジックを用いる際には動作周波数の点でも十分な余裕を持っておく必要がある。

パターンジェネレータとパターンレコーダはかなり高速に動作した。そこで、Slave Board や Star Switch のプロトタイプ用のロジックの性能の評価の際に用いることとした。

[1] 今回用いた Foundation では、信号が 50 % の確率で通過可能な時間の 2 倍をゲートでの遅延として用いている。

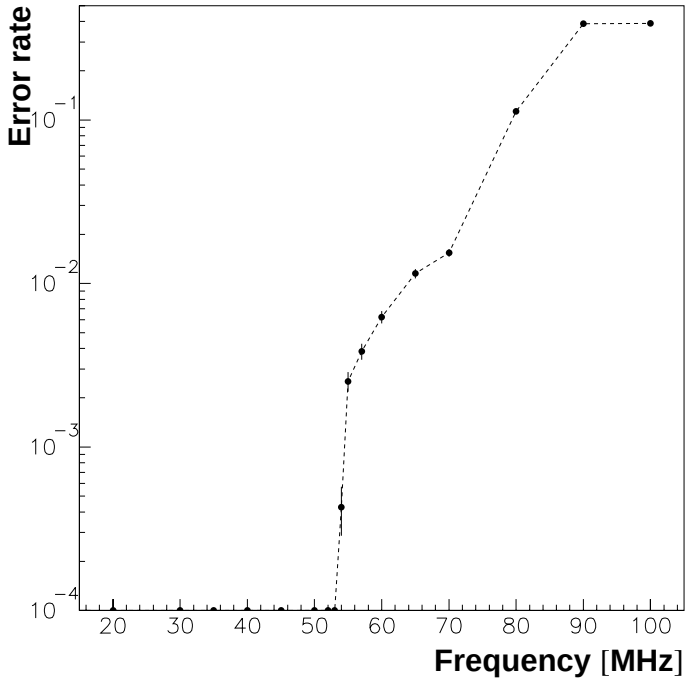

図 5.3： pt3 モジュールでパターンジェネレータとパターンレコーダを実現したときの周波数とエラーレートの関係。このロジックでは 53 MHz まで動作した。

5.2 Slave Board ASIC 読み出し系の検証

5.2.1 Slave Board ASIC 読み出し系のプロトタイプの作成

Slave Board ASIC の読み出し系の仕様は第 4 章に述べたように決定した。この仕様に従って Slave Board ASIC の読み出しの部分の HDL 記述を Verilog-HDL を用いて行った。そして、この HDL 記述のチャネル数やバッファの深さを減らしたものを pt3 モジュールの FPGA に実装して、読み出しのスキームの検証を行った。

HDL を用いて記述した回路のブロック図は Appendix C の図 C.1 と図 C.2 に載せてある。Appendix C には HDL 記述の一部も載せてある（113 頁）。

FPGA の容量の関係上、実装したロジックは最終的に必要なものよりも表 5.1 の様に規模を縮小してある。しかし、Verilog での記述を行う際には入力信号数やバッファの深さなどはほとんどがパラメータとして宣言してある。ソースレベルでは最終的に必要な規模のものとプロトタイプ用の規模のものの違いは、基本的にはパラメータの値だけである。

この他、今回 HDL 記述で記述したロジックと Slave Board ASIC の仕様との相違は以下の通りである。これらは Slave Board ASIC の読み出しの検証という点では特に問題となるものではない。

- トリガー行列は実装されていない。
- 隣接する Slave Board との共有信号は存在しない。
- モジュールのアドレスは固定されている。

パラメータ	本来の値	プロトタイプ用の値
Patch Panel ASIC からの入力信号数	128	8
BCID カウンタ等のカウンタのビット数	8	4
モジュールのアドレス	8	4
トリガー出力	18	2
レベル 1 バッファの最大の深さ	128	34
デランダマイザの深さ	8	6

表 5.1： Slave Board ASIC プロトタイプ用のパラメータ。本来の値はダブレットワイヤの場合の値。

- SEU（Single Event Upset）対策は施されていない。SEU は放射線によってレジスタのビットが反転してしまう現象である。これについては Appendix F でのべる。
- LS-link から入力されるクロックとシステムクロックは共通である。
- JTAG の一部の機能は実装されていない。
- 実際の Slave Board ASIC は Slave Board の種類に対応した複数のロジックを切替えて用いるが、そのようなことは考慮されていない。

Verilog で各モジュールを記述する際には、必要に応じてこれらのモジュールに対して論理シミュレーションを行って論理に間違いがないかを確かめた。論理シミュレーションには Cadence 社の Verilog-XL を用いた。

各々のモジュールの記述が終了したあと、これらのモジュールを組み合わせ、全体としての論理シミュレーションを行った。論理シミュレーションは以下のような手順で行った。

- 乱数を用いて Slave Board ASIC への入力のパターンを作成する。
- Cadence 社の Verilog-XL を用いて、このパターンに対する出力をシミュレーションする。
- この出力と、入力したパターンから予測される出力が一致するかを確認する。
- 不具合がみつかった場合には、HDL 記述を修正して上記の手順を繰り返す。

図 5.4 にシミュレーション結果を示す。図に示された信号のうち、UDAT0, UDAT1, USYN, UCLK は Slave Board の LS-Link への出力信号で、その他は Slave Board への入力信号である。このシミュレーションの際にはレベル 1 のバッファの深さは 6 段に設定してある。LS-Link への出力では UDAT0 = 0, UDAT1 = 0, USYN = 0 がパケットのはじまり、UDAT0 = 1, UDAT1 = 1, USYN = 0 がパケットの終りを意味している。L1A が真になると、その 6 クロック前とその前後のバンチの TGC 信号がシリアル化されて、LS-Link に出力されている。このシミュレーションで得られた出力のパターンは、入力パターンから予測されるものと正しく一致した。

HDL 記述が一通り完成すると、記述されたロジックを pt3 モジュールの FPGA の 1 つに実装した。FPGA に実装するための論理合成や配置配線等は Xilinx 社の Foundation ver 2.1 を用いて行った。このロジックのゲート数は 9800 で、FPGA の CLB を 72 %（782 個中の 567 個）使用した。また、クリティカルパスから得られる動作周波数は 22.8 MHz であった。このうち、内部配線による最大の遅延は 17 ns であった。

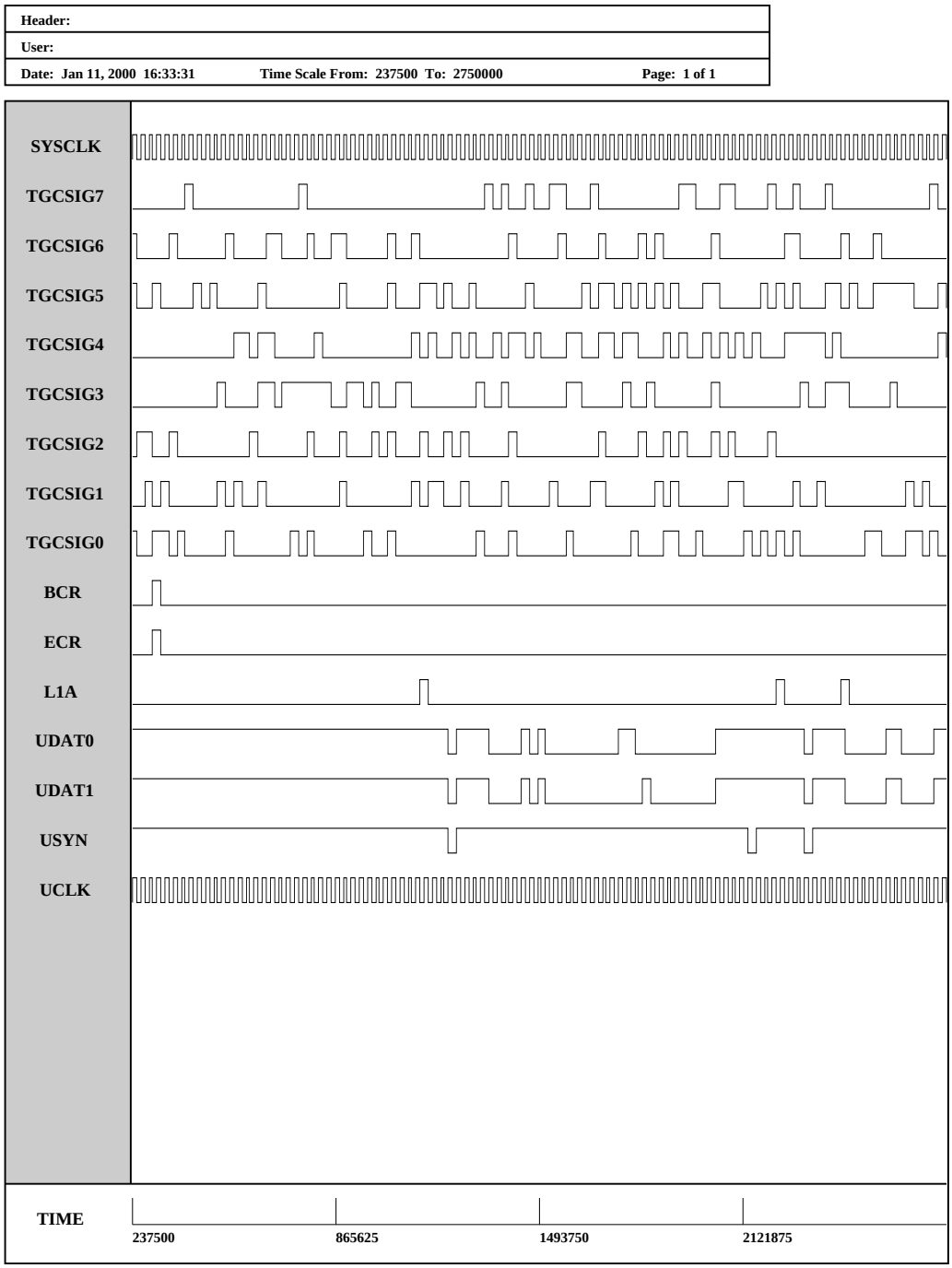

図 5.4： Slave Board ASIC のプロトタイプ用のロジックのシミュレーション結果。UDAT0, UDAT1, USYN, UCLK は Slave Board の LS-Link への出力信号。その他は Slave Board への入力信号。レベル 1 のバッファの深さは 6 段に設定してある。

5.2.2 プロトタイプの動作結果

今回作成したロジックを実装した FPGA が正しく動作しているかどうかをたしかめるために、pt3 モジュールの残りの FPGA に 5.1.2 で作成したパターンジェネレータとパターンレコーダを実装した。そして、パターンジェネレータからはシミュレーションで用いたパターンと同じものをプロトタイプ用のロジックを実装した FPGA に対して入力し、ここからの出力をパターンレコーダで記録した。図 5.5 はこの時の入出力のパターンをロジックアナライザーで観測したものである。このときのクロックは 20 MHz である。図 5.4 からもわかるように、得られたパターンはシミュレーションで得られたものと同じであった。

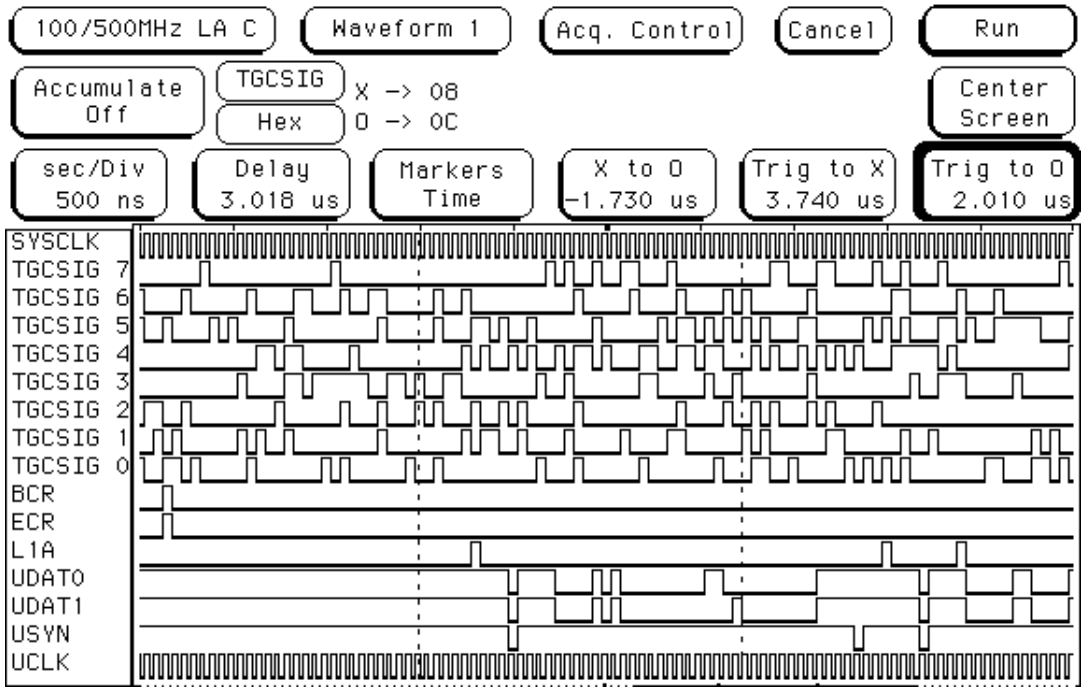

図 5.5： FPGA に実装した Slave Board ASIC のプロトタイプ用のロジック動作結果。クロックは 20 MHz。図 5.4 のシミュレーション結果と一致している。

参考までにこのロジックがどの程度のクロック周波数まで動作可能か調べるために、周波数を変えながらパターンの生成と記録を行った。エラーの発生率を図 5.6 に示す。このロジックは 43 MHz まで動作した。

この他、LS-Link から読み出しの停止やパラメータの読み出し命令を送るなどしたが、期待されるとおりの働きをした。以上から、Slave Board ASIC のプロトタイプは仕様通りの機能を実現できたことが分かった。今後は、今回の検証の際に実装しなかった一部の機能を付け加えて、ASIC 等の試作を行う必要がある。

この FPGA のプロトタイプは 43 MHz まで動作した。しかし、必要な回路の規模はプロトタイプのものよりもはるかに大きいので、これを仮に同じ性能の FPGA で製作したとすると動作周波数は遅くなるであろう。しかし、実際には Slave Board は ASIC を用いて開発するので、この結果から動作周波数を議論してもあまり意味はない。

ASIC の場合には、目的の論理回路を実現するために必要な論理ゲートを配置し、その間をアルミやポリシリコンの配線で接続する。ゲートアレーの場合にはゲートは最初から配置されているが、それらの中から必要なものだけを選んでアルミ配線で接続する。これに対して、FPGA の場合はあらかじめ用意された

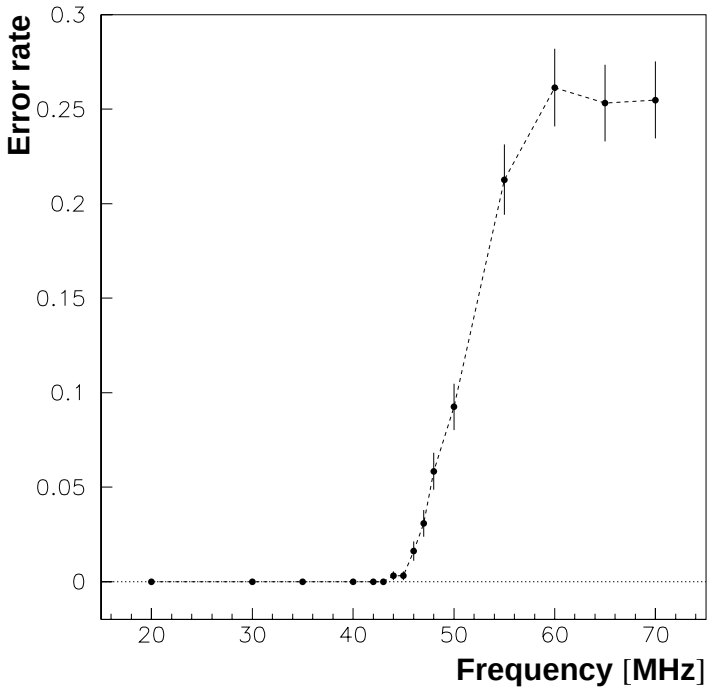

図 5.6： FPGA に実装した Slave Board ASIC のプロトタイプ用のロジック動作周波数。縦軸は出力データに含まれるエラー率。このロジックは 43 MHz まで動作した。

内部配線がいたるところで交差しており、その交点にある PSM（Programmable Switch Matrix）と呼ばれる回路がこれらの内部配線を接続している（Appendix D 参照）。PSM はトランジスタで形成されており、信号がここを通過するたびに伝播遅延が生じてしまう。また、FPGA の場合には CLB（Configurable Logic Block）が基本単位となっている。Appendix D の図 D.2 に示したように、CLB は LUT（Look-up Tabke）とフリップフロップからなるが、利用する LUT を切替えたりするために、各所にマルチプレクサが配置されている。そのため、例えば単なるフリップフロップであっても、その直前にある LUT とマルチプレクサを経なければならないため、ASIC で実現した場合よりもはるかに多くの遅延の要素を含む。

このように、ASIC と FPGA はその構造が異なるため、直接性能を比較することはできない。今回、FPGA を用いて機能の検証が行えたので、今後は ASIC を試作して性能の検証を行う必要がある。

5.2.3 Slave Board ASIC に対する見積もり

FPGA を用いたプロトタイプで可能だったものは、入力用の信号数やバッファの深さを減らした規模の小さなものであった。最終的に必要な Slave Board ASIC の規模は、今回のプロトタイプ用のものよりはかなり大きいものになる。そこで、最終的に必要な Slave Board ASIC がどの程度の規模になるのかを HDL 記述の論理合成を行うことによって見積もった。なお、規模の指標となる量としてゲート数がある。デジタル回路では、基本論理ゲート（2 入力 NAND）を基準にしたときのロジックを実現するのに必要なデバイスの大きさを指す。

論理合成の際に用いたツールは、Synopsys 社の Design Analyzer と Xilinx 社の Foundation である。前者は VDEC での ASIC の試作の際に用いるプログラム、後者は FPGA の実装用に用いるプログラム

である。いずれもターゲットとなるデバイスを指定しなければならないので、前者についてはローム社の CMOS 0.6 μm フルカスタムを、後者は Xilinx 社 FPGA Virtex シリーズを指定した。

論理合成した HDL 記述は、プロトタイプの際に用いた HDL 記述のパラメータを最終的に必要な値に変えたものである（表 5.1 参照)。但し、全てのパラメータを最終的な値に変えるとロジックが大きくなりすぎて Foundation での論理合成が不可能であったので、入力信号の数（TGC のチャネル数）を変えながら論理合成を繰り返した。論理合成にはかなりの時間を要し、Design Analyzer を用いて 128 チャネルの ASIC の論理合成を行うのに、UltraSPARC-II 300 MHz のマシンで CPU 時間で 105 時間かかった。

図 5.7 に Design Analyzer による論理合成から得られたゲート数を、図 5.8 に Foundation を用いた論理合成から得られたゲート数を示す。前者は論理合成の結果から得られた必要な全面積を基本論理ゲートの面積で割ったものである。後者は 2 入力ゲートを 1 ゲートと数えているのは同じであるが、内部のファンクションジェネレータを 8 ゲートというように使用している要素に適当なゲート数を割り当てて換算している。Design Analyzer を用いた論理合成の際には、Slave Board ASIC のロジックの中で大きな容量を必要とするレベル 1 バッファ、デランダマイザ、トリガーマスク、PSC 等の占めるゲート数の内訳も知ることができたので、同時に示す。グラフの横軸は入力信号数である。最終的には 128 チャネルの ASIC を開発する必要がある。

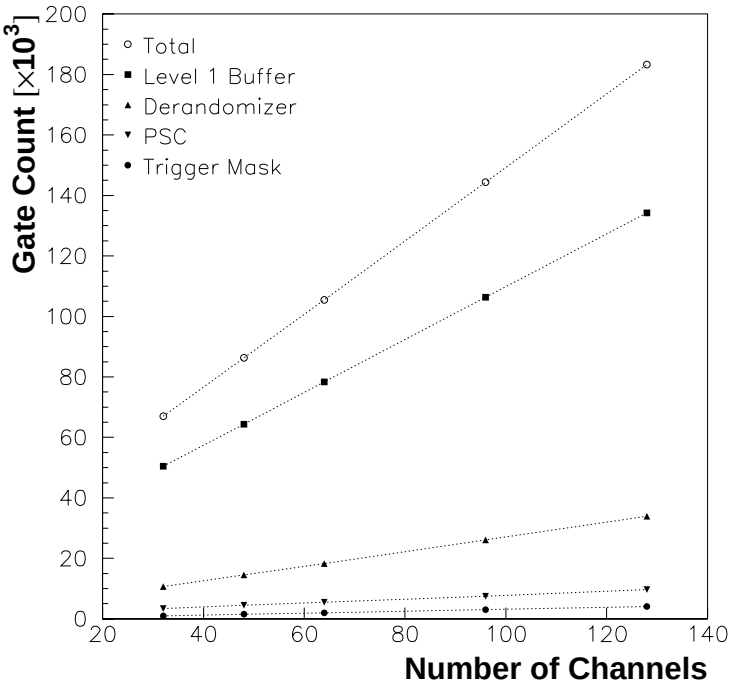

図 5.7： Design Analyzer を用いた Slave Board ASIC のゲート数の見積もり。Rohm CMOS 0.6 μm フルカスタムをターゲットデバイスにしている。同時に、大きな容量を必要とするレベル 1 バッファ、デランダマイザ、トリガーマスク、PSC が占めるゲート数についても示す。

図から分かるようにチャネル数が増えるに従ってゲート数も直線的に増加した。これはバッファの大きさが基本的にはチャネル数に比例することから当然予想されることである。Design Analyzer を用いた見積もりでは、128 チャネルの場合にはゲート数は 183 k であった。一方、Foundation を用いた見積もりでは 72 チャネル以下の論理合成しか行えなかったので、その結果から外挿すると 205 k ゲートという結果

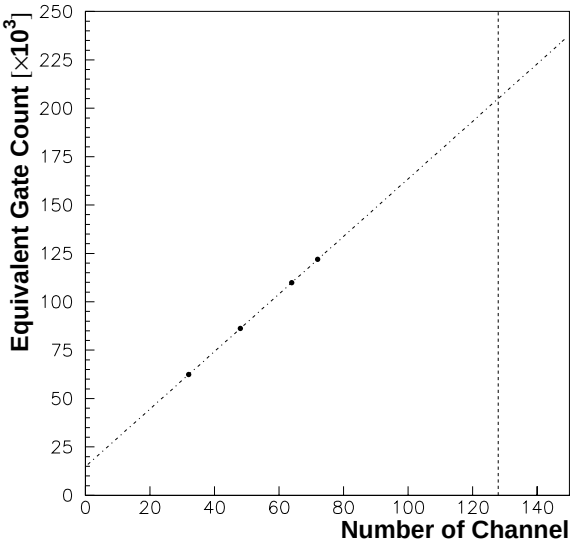

図 5.8： Foundation を用いた Slave Board ASIC のゲート数の見積もり。Virtex シリーズをターゲットをターゲットデバイスにしている。チャネル数が 72 以下のものしか合成できなかった。

が得られた。見積もりの際に対象としたデバイスが異なるため論理合成結果が異なる、ゲート数の定義が異なる、などが原因で多少の不一致が生じている。いずれにせよ、最終的には読み出し回路でおよそ 200 k ゲートの規模になることがわかった。

今回の見積もりにはトリガー行列が入っていないが、実際には 5 種類の Slave Board ASIC に対応した 5 つのトリガー行列を実装する必要がある。また、チップの製作の際には容量に少し余裕がある方がよい。従って、最終的には 300 k から 500 k ゲート規模の ASIC（ゲートアレイ）を用いて開発することになると思われる。

図 5.7 あるいは表 5.2 からゲート数の内訳をみると、レベル 1 バッファが全ゲート数の 7 割を、デランダマイザが 2 割を占めており、両者を合わせると 9 割に達する。従って、回路規模を小さくしたければこれらのバッファの規模を減らすことが必要になる。現在の仕様ではレベル 1 バッファの深さは最大 128 段、またデランダマイザの深さは 8 段となっているが、これらは各々 100 段 5 段まで減らすことが可能ではあるので、あと 2 割程度は回路規模を減らすことは可能である。

今回行ったのは論理合成だけであり、配置配線は行っていないので、配線リソースをどの程度消費するかという見積もりは完全にはできないが、ネット（信号線）の数からおおまかな目安を得ることはできる。表 5.2 には Slave Board ASIC の各要素のゲート数とともにネット数とその比も示す。その他のネット数が多いのは、レベル 1 バッファなどの機能単位同士の間の接続も含まれているからである。

表からデランダマイザのネット数が他の機能単位に比べて相対的に多いことが分かる。それに対してレベル 1 バッファはネット数が相対的に少ない。レベル 1 バッファは基本的には単なるシフトレジスタである。図 5.9 (a) に概念図を示したが、データはレジスタ内を順番にシフトするだけでよいので、配線としては隣同士のレジスタを接続するものだけでよい。従って、レベル 1 バッファは最も簡単な配線で実現可能な回路の一つである。これに対して、デランダマイザは FIFO である。FIFO の場合には、全てのレジスタは FIFO への入力信号と接続されていなければならない。出力信号に関しても同様で、全てのレジスタの出力が FIFO の出力になる。よって FIFO の概念図は図 5.9 (b) のようになる。この図からもデランダマイザは多くの配線リソースを使用することが予想される。実際に、表 5.2 のようにネット数のゲート

	Gate Count	Number of Net	Net/Gate Ratio
Trigger Mask	4031	1297	0.32
Level 1 Buffer	134200	29102	0.22
Derandomizer	33928	15290	0.45
Parallel Serial Converter	9700	2731	0.28
Others	1427	1922	1.35
Total	183286	50342	0.27

表 5.2： Slave Board ASIC の各要素のゲート数とネット数。Design Analyzer を用いて 128 チャネル用の回路を論理合成した結果から得られたもの。比較のためネットとゲートの比もしめす。その他（Others）のネット数が多いのは、各機能単位間を繋ぐネットも含まれているため。

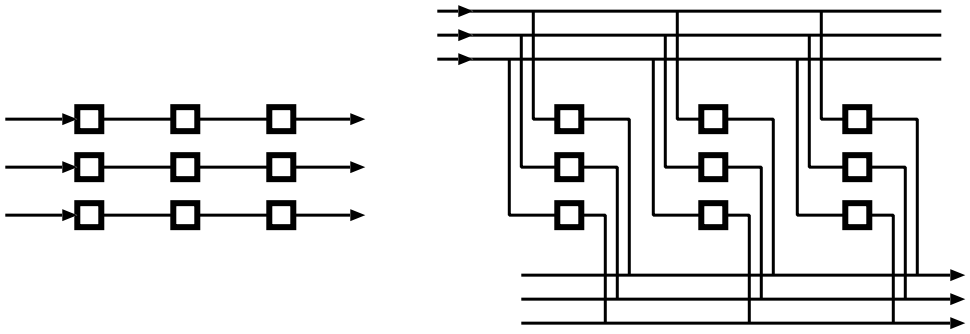

図 5.9： レベル 1 バッファとデランダマイザの概念図。(a) はレベル 1 バッファ、(b) はデランダマイザ。四角がレジスタを表す。

数に対する比はレベル 1 バッファが最も小さく、デランダマイザが最も大きかった。このことから、配線リソースという点ではデランダマイザが課題となる[2]。

最後に動作速度という点では、ゲートによる遅延ではもっとも遅延が大きい場所でも 3.5 ns であった。実際にはこの値に配線での遅延も加わるが、これらを含めても 40 MHz では十分に動作すると思われる。この点については今後の ASIC の試作で確認する必要がある。

5.3 VDEC での ASIC の試作

5.3.1 VDEC とは

FPGA を用いて Slave Board ASIC の読み出し系の機能を検証を行う一方で、実際に ASIC を試作をおこなって性能を確認することが重要である。そこで、東京大学の VDEC（VLSI Design and Education Center, 大規模集積システム設計教育研究センター）を利用して ASIC の試作を行った[3]。

[2]最終的に ASIC を製作する際には、FIFO などはベンダーからライブラリとして提供されるかもしれない。

[3]本チップ試作は東京大学大規模集積システム設計教育研究センターを通しローム（株）および凸版印刷（株）の協力で行われたものである。

VDEC は大学や高専での VLSI（Very Large Scale Integration：大規模集積回路）の設計教育、試作の支援を目的としている。VDEC では複数の大学や高専からの別々の VLSI チップの設計を集め、これらを一枚のシリコン・ ウェーハの上に相乗りで配置した設計データにまとめた上で企業に試作を依頼し、完成したチップを各大学ごとに切り分けて返送する。こうすることにより、比較的安価に ASIC の試作を行うことができる。ただし、VDEC は VLSI の設計教育を目的としているため全ての設計の工程をユーザが行う必要がある。

今回の試作で用いたプロセスはローム社の CMOS 0.6 μm フルカスタムである。チップのサイズは 4.5 mm × 4.5 mm である。このチップは 87 個の入出力ピンを備えており、42 k ゲートまでのロジックが作成可能である。

今回はこのチップの一部分を使って、Slave Board ASIC の読み出し系の部分の試作を行った。

5.3.2 設計工程

HDL を用いた ASIC の設計は、 4.2.3 で述べたように仕様の決定と HDL 記述から始めて、論理シミュレーション、論理合成、ゲートレベルでのシミュレーション、レイアウト（配置配線）、レイアウトの検証という順に行われる。今回の試作では以下のような開発ツールを用いて行った。

論理シミュレーション 作成した HDL 記述は最初に Cadence 社の Verilog-XL シミュレータを用いて論理を確認する。

論理合成 次に Synopsys 社の Design Analyzer を使用してターゲットデバイス（この場合はローム社の CMOS 0.6 μm フルカスタム）固有のライブラリを使用した論理合成を行う。合成結果はネットリストにして保存する。

ゲートレベルでのシミュレーション 論理合成結果のネットリストにテスト信号を入力してゲート遅延や予想される配線遅延を考慮したシミュレーションを行う。このために再び Cadence 社の Verilog-XL シミュレータを用いる。

レイアウト ネットリストをもとにゲートの配置と配線を行う。レイアウトはベンダーに委託されることが多いが、VDEC では利用者が行う。ここでは Avant 社製の自動配置配線ツールである Apollo を用いる。

レイアウトの検証 設計したレイアウトは配線の線幅や間隔などが実現可能でなければならない。そのために定めた設計ルールを破っていないことを確認する DRC（Design Rule Check）を行う必要がある。さらに、レイアウトが正しくネットリストを反映しているかを検査する LVS（Layout vs Schematic）という作業も行わねばならない。これらのために Cadence 社の Dracula というツールを用いた。

5.3.3 読み出し系の ASIC の試作

今回は読み出し系としては初めての試作であったので、読み出し系のうちもっとも基本的な部分だけのかなり小規模なものとした。

ASIC 試作を行った回路は、Slave Board の読み出し系のうちデランダマイザとパラレルシリアル変換の部分である（図 5.10 ）。デランダマイザへの入力信号は 8 チャネル、デランダマイザの深さは 5 段である。論理合成を行ったところおよそ 900 ゲートであった。この回路内では最も遅延の多い経路でも 1.2 ns であった。同じ回路を Spartan シリーズの FPGA で実装した場合の最大の遅延 9.1 ns に比べれば、十分に小さな値である。

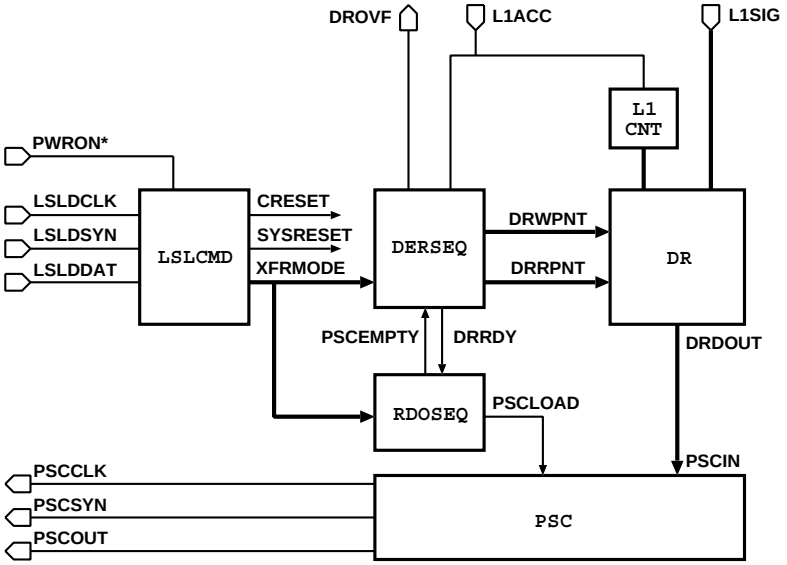

図 5.10： VDEC で試作した Slave Board 読み出し回路

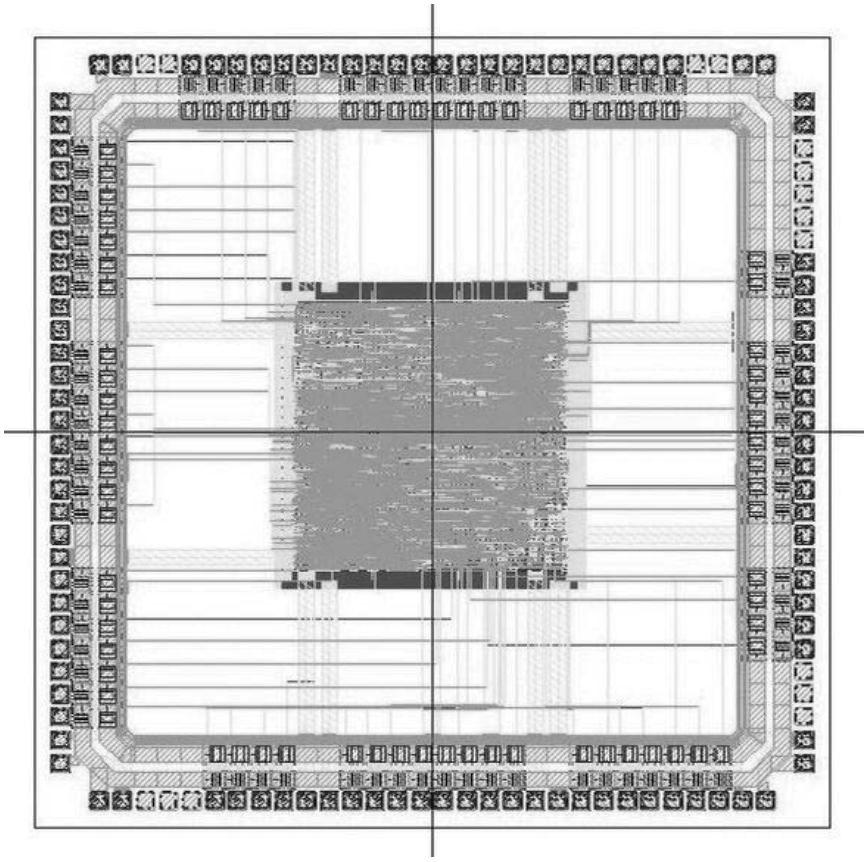

図 5.11： VDEC で試作した ASIC のレイアウト図

この回路を同じ ASIC の中にいれる他のロジックとあわせて上記の手順に従って設計を行った。途中の ASIC のレイアウト結果を図 5.11 に示す。

この ASIC は 1999 年 10 月中に完成する予定であったが、製造工程上の問題から完成が遅れている。完成後には期待どおりの機能を実現できているかを検証する予定である。

また、現在 Slave Board ASIC の読み出し系の機能をのほとんどを搭載した ASIC を VDEC で試作するために開発を行っている途中である。この試作はピンの数や容量の制限からやはり完全な大きさのものではないが、動作速度などの性能についての検証が行える。この ASIC は 2000 年 4 月頃に完成する予定である。

5.4 Star Switch の検証

5.4.1 レシーバ回路の HDL 記述

Star Switch についても Slave Board ASIC の読み出し系と同様に Verilog HDL での記述を行い、これを pt3 モジュールに実装した。今回、特に Star Switch のレシーバの部分の検証を行った。

図 5.12 に Star Switch のレシーバ回路を示す。Slave Board から LS-Link を経て到達したデータはゼロサプレス後、デランダマイザの中に保持される。ゼロサプレスされた後のデータは固定長のデータではないので、プリフォーマッタという部分で数回のクロックに分けて固定幅の FIFO（デランダマイザ）に書き込んでいる。FIFO 内のデータは、やはり数クロックの間にデータバス上に流される。これらのデータの流れを制御する部分がレシーバシークエンサである。

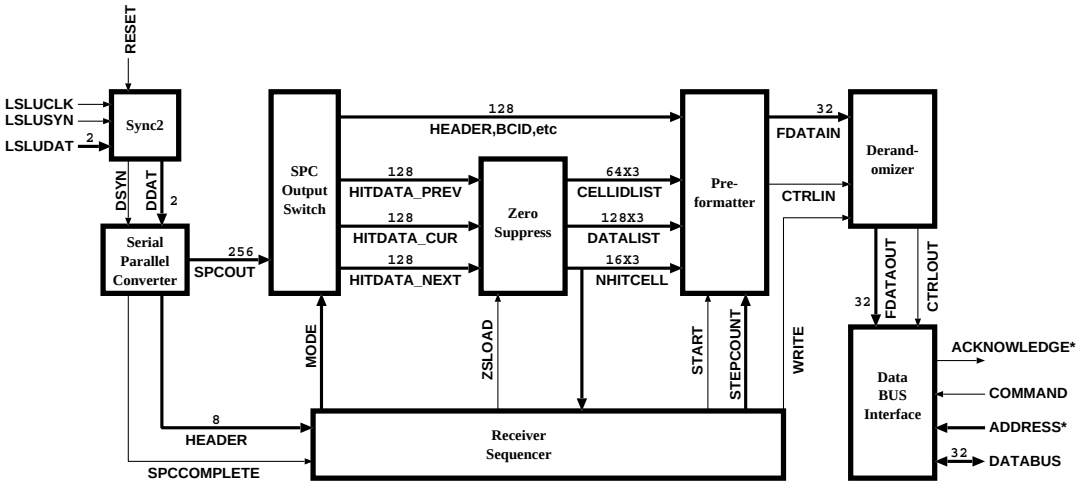

図 5.12： Star Switch のレシーバ回路。但し、プロトタイプ用に作成したものは、規模を小さくするためビット幅が異なる。

データバスのプロトコルは、今回のプロトタイプの段階では以下の様な単純なものを用いた。8 ビットのデータバスの他に、アドレスラインと COMMAND と ACKNOWLEDGE* というラインを用意する。マスタは COMMAND を high にしながら、アドレス線にレシーバのアドレスを流す。これに対し、該当するモジュールは ACKNOWLEDGE* を low に下げてからデータを送信するというものである。

Star Switch のプロトタイプにおいても、回路規模を小さくするために次のような制限を加えた。

- Star Switch では本来は 3 バンチ分のデータを処理するが、2 バンチ分しか処理しないものとする。

- ヘッダ、アドレス、BCID などゼロサプレスしないデータの長さは合計 16 ビットとして扱った。本来は 32 ビットある。
- 本来はダブレットワイヤ Slave Board からは 128 ビットの TGC データが送られて来る。しかし、プロトタイプでは 16 ～ 64 ビットのデータが送られて来ると仮定した。
- ゼロサプレスは 8 ビットごとに圧縮を行うが、それを 4 ビットごとにした。

TGC データのデータ幅が 64 ビットの場合、2 バンチ分しか処理しない関係から回路の規模としては最終的に必要なものの 3 分の 1 程度となる。なお、TGC のデータは Slave Board からシリアル化されて送信されて来るので、データの幅に関わらず入出力ピンの数は同じである。以上のような回路を Verilog で記述して、論理シミュレーションを行い、シミュレーション段階で問題が無いことを確認した。

5.4.2 Star Switch プロトタイプの動作

前節で述べたプロトタイプ用のロジックを pt3 モジュール内の FPGA に実装し、動作の検証と動作周波数の測定を行った。

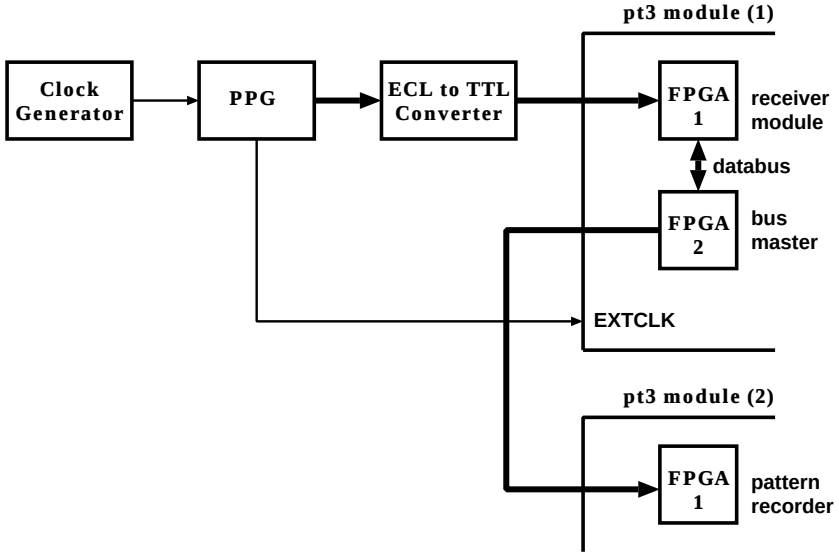

図 5.13： Star Switch レシーバモジュールの動作周波数の測定のセットアップ。

このときのセットアップを図 5.13 に示す。図のように PPG からの出力は TTL に変換されたあと、pt3 モジュールに入力される。pt3 モジュール内には、前述のレシーバモジュール用のプロトタイプ用ロジックと、データバスからデータを読み出すバスマスタ用のロジックが入っている。バスマスタ用のロジックは適当な間隔で COMMAND 信号線を high にしてデータを収集するだけの単純なロジックである。バスマスタは収集したデータを単にラッチして出力するだけであり、この出力をもう一台の pt3 モジュールへ入力した。もう一台の pt3 モジュールには、5.1.2 で作成したパターンレコーダを実装し、出力されたパターンを記録した。そして、Slave Board ASIC の動作テストの場合と同様に、入力したパターンと出力されたパターンの間でどの程度のエラーレートがあるかを、クロックジェネレータの周波数を変えて測定した。

レシーバモジュール用に用意したロジックの規模と、クリティカルパスから算出した動作周波数の一覧を表 5.3 に示す。TGC のヒットデータの幅が大きくなると、規模は当然大きくなる。それに伴って動作周

波数も遅くなる。但し、この数値は論理合成や配置配線の仕方に依存するので、必ずしも回路の規模だけで決まるものではないようである。これらの回路はクロックの周波数が低いとき（10 MHz）には、入力したデータは正しくゼロサプレスが行われて出力された。

NBit	CLB usage	Number of Flip Flop	Frequency [MHz]
16	293 (37%)	413 (26%)	32.5
24	445 (56%)	568 (36%)	32.6
32	526 (67%)	669 (42%)	18.3
40	572 (72%)	816 (52%)	33.2
48	653 (83%)	940 (59%)	20.3
54	721 (91%)	1064 (67%)	21.5
64	760 (96%)	1188 (75%)	14.5

表 5.3： レシーバ用ロジックのパラメータ。NBit は回路が処理する TGC のデータのデータ幅である。動作周波数はクリティカルパスから算出したもの。

これらのロジックを FPGA に実装し、先程のべたセットアップで周波数を変えながら正しく動作するか確かめた。測定結果を図 5.14 に示す。一方、Foundation では、特定の動作周波数を仮定し、論理合成の結果から配置配線に対する時間制約を課し、それにもとづいて配置配線を行うことが可能である。今回のロジックでは、30 MHz の動作周波数の時間制約下で配置配線を行うことが可能であったが、35 MHz の制約をつけると、TGC のデータ幅を 64 ビットにした際配置配線が行えなかった。そこで、30 MHz の時間制約をつけて配置配線を行って同様の測定を行った。そのときの測定結果を図 5.15 に示す。

これらのデータから回路の最大の動作周波数をまとめたのが図 5.16 である。配置配線に時間制約を課さなかった場合は、全体としてみればロジックが大きくなると動作速度は落ちているようであるが、むしろ配置配線の結果に大きく依存しているようである。30 MHz の時間制約を課した場合には、どのロジックも 30 MHz では動作した。

今回の測定から、次のようなことが分かる。まず、時間制約を行わずに配置配線した場合にクリティカルパスから算出される動作周波数は、配置配線に大きく依存する。そのため、必ずしもその回路の動作速度の限界を表しているわけではなく、時間制約を行って配置配線すればさらに早いクロックで動作する。逆に言えば、動作周波数の最悪値を表していることになるので、デバイスを選択する際の目安として用いることができる。

時間制約を課して配置配線を行った場合には 30 MHz で動作させることが可能であった。実験では 40 MHz で十分に動作可能な性能がないといけない。今回のプロトタイプモジュールでは Spartan シリーズの FPGA を用いたが、Appendix D の表 D.2 にも示したようにさらに高性能の FPGA が存在する。例えば、Virtex シリーズの FPGA は、表 D.2 からもわかるように Spartan の 3 倍程度の性能をもっている。よってこれらのデバイスを用いれば、今回用いたものと同じロジックを 40 MHz で動作させることは特に問題はないと思われる。実際、今回のロジックを Virtex シリーズの XCV600 をターゲットとして、時間制約なしで配置配線を行ったところ、クリティカルパスから見積もられる動作周波数は 54 MHz であった。

むしろ不確定な要素としては、今回用いたロジックが最大のものでも最終的に必要なロジックの 3 分の 1 程度のものということである。実際、回路の規模が大きくなると動作周波数は下がってしまう。しかし、今回の測定からは動作速度が回路規模に大きく依存するという結果は得られなかったし、後で述べるように、実際の回路規模から動作周波数を見積もった場合にも 40 MHz の動作については問題がなさそうである。

したがって、さらに高性能の FPGA を導入することにより、 FPGA を用いて Star Switch のレシーバ

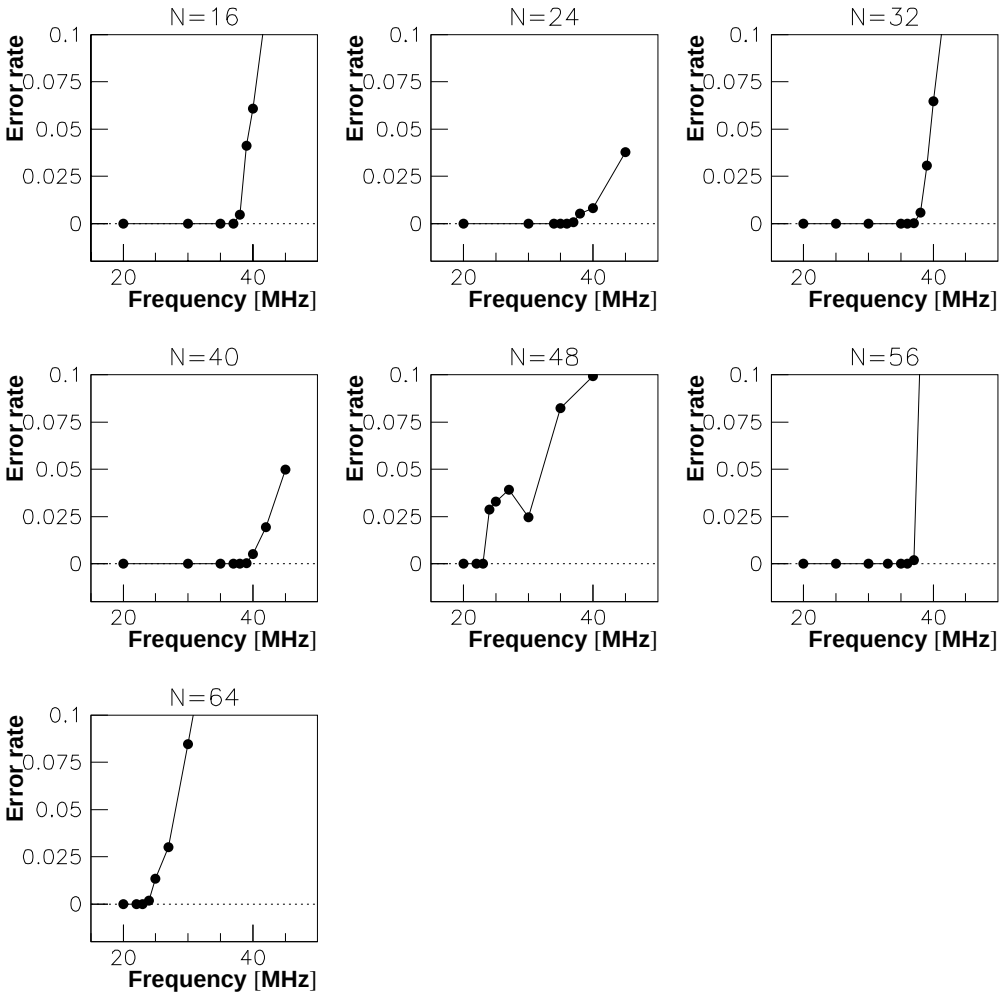

図 5.14： レシーバ回路の周波数とエラー率の関係 (1)。通常の配置配線を行った結果。N は回路が処理する TGC ヒットデータのデータ幅を表す。

回路を開発することが可能であるといえる。

5.4.3 今後の Star Switch の開発

今回の pt3 モジュールを用いたテストで、シリアルパラレル変換、ゼロサプレス、デランダマイザの部分は正しく機能することが分かった。しかし、次のような点でさらに検証が必要である。

- データバスやコントロールバス。今回は、今回は検証用に簡単なプロトコルを定めた。動作周波数を測定する際には、レシーバモジュールが 1 つだけで行ったので問題は無いが、pt3 モジュールを複数用い、その間をフラットケーブルなどで接続して動かそうとすると動作周波数はケーブルの長さに応じて急激に下がってしまう。これらのバスはクレートのバックプレーンを利用する予定であるので、実際にバックプレーンを用いて動作の検証を行い、最適なプロトコルを定める必要がある。

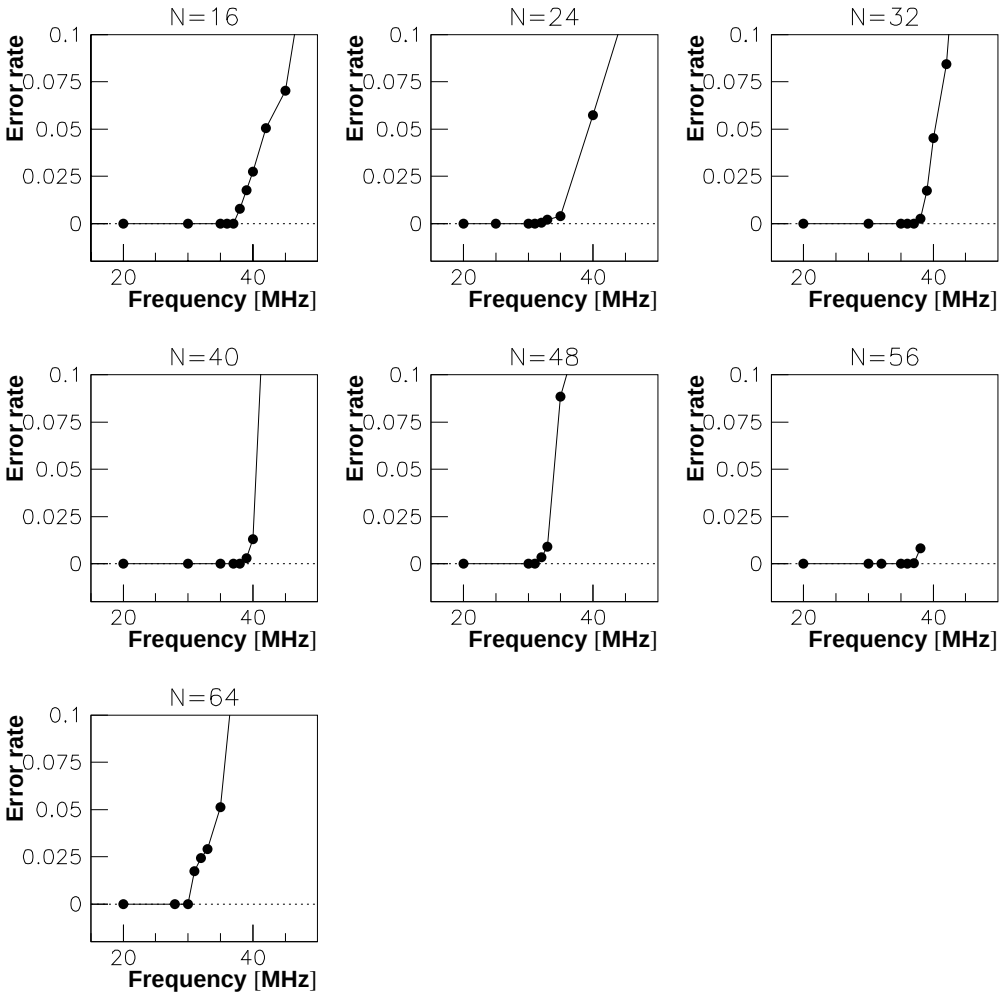

図 5.15： レシーバ回路の周波数とエラー率の関係 (2)。配置配線の際に 30 MHz の時間制約を課した場合。N は回路が処理する TGC ヒットデータのデータ幅を表す。

- データ転送モジュール。データ転送モジュールは、データバスを介して Slave Board からのデータを収集し、特定のフォーマットに整形して、エレキハット内の Local DAQ Master にデータを転送するモジュールである。この部分は、基本的な動作としては単純なものではあるが、BCID や L1ID の整合性の検査や、特定の Slave Board に問題が発生した場合の処理などを行う必要があり、実際にはかなり複雑な機能が要求されると思われる。
- コントロール系。JTAG 信号の発生や、LS-Link での早い命令の分配などについても検証が必要である。

これらを確認するために、今後、Star Switch 専用のプロトタイプモジュールの作成を行う予定である。このモジュールを仮に pt4 モジュールと呼ぶこととする。

pt4 モジュールの仕様としては次の様なものを考えている。

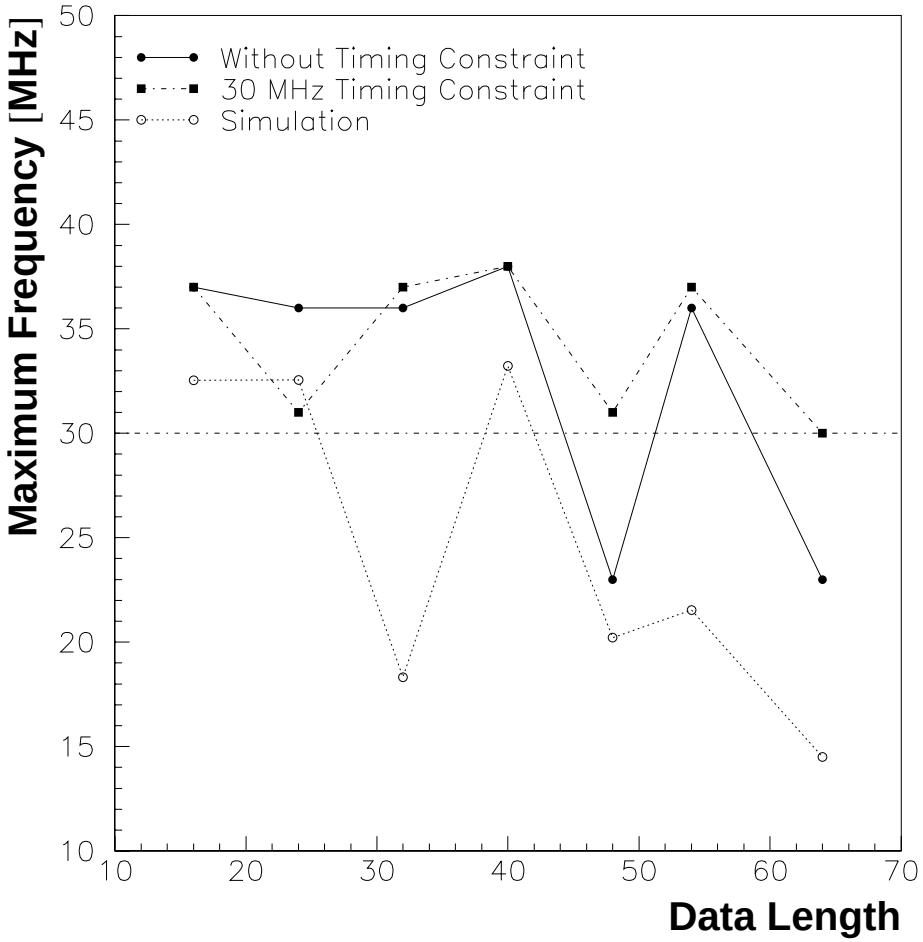

図 5.16： レシーバ回路の動作周波数。横軸は回路が処理する TGC ヒットデータのデータ幅でこれが大きい程ロジックの規模は大きくなる。時間制約なしで配置配線したときの動作周波数、30 MHz の時間制約つきで配置配線したときの動作周波数、時間制約無しで配線したときにクリティカルパスから見積もった動作周波数をプロットした。

- 完全な規模のロジックを実装可能な大規模な FPGA の搭載した 40 MHz のシステムクロックで動作する VME モジュール。
- データバスやコントロールバスとしては VME のバックプレーンを利用。
- レシーバへの入出力ピンは LVDS をサポート。しかし、TTL などでの入力も可能である方が便利であると思われるので、何らかの方法で切替えられるように設計すべきである。

5.4.4 次期プロトタイプの FPGA

pt4 モジュールの設計を行うためには、FPGA を用いてどの程度の性能が得られるかの見積もりを立てておく必要がある。今回 Xilinx 社 Foundation を用い、Xilinx 社の FPGA を用いた場合にどの程度の性能が得られるかをシミュレーションした。

まず、レシーバモジュールのロジックの HDL 記述を、Foundation を用いて論理合成、配置配線を行った。この操作により、FPGA 内のリソースをどの程度利用したか、クリティカルパスから算出される動作周波数の限界はいくらかが得られるので、HDL 記述、FPGA の種類や大きさを変えながら行った。

この際に用いたロジックは、Slave Board からのデータを受け取って、データ圧縮（ゼロサプレス）後、データバス上にデータを流すまでの部分である。動作テストの場合と異なり、L1A の前後の 3 バンチ分のデータの読み出しが処理可能なものとなっている。TGC のヒットデータ長は 128 ビットである。動作テストを行ったロジックは最大のもので 64 ビットの TGC データに対し 2 バンチ分の処理を行うので、今回の見積もりに用いたロジックはおよそ 3 倍の回路規模であることになる。なお、バスインターフェースの直前にある FIFO は、FIFO の深さに対する見積もりの場合を除いて含まれていない。また、制御用のロジックも含まれていない。実際には、これらを含めるとロジックの規模は大きくなり性能は悪くなると予測されている。しかし、Slave Board からのデータを処理する部分は、400 ビット程度のデータが行き来するのに対し、制御用のロジックは 10 ビット程度のデータを処理するだけなので、この部分が全体に寄与する影響は少ないと思われる。

FPGA の配置配線は通常の自動配線を用いて行った。実際には、時間制約を課した配置配線を行ったり、一旦配線を行った後に動作の遅い部分の再配線を行う機能などを利用すれば動作速度は向上する。勿論、ロジック自体を工夫することによってもよりよく動作させるようにすることも可能である。また、既に述べたように、これから与えられる動作速度などの性能は配置配線の結果に大きく依存しているようである。しかし、これ以上の動作周波数では動かすことができるという最悪値という意味では、十分な目安になる。しかも、毎年のように性能の良い種類の FPGA が登場しており、pt4 モジュールの設計のための見積もりとしては十分である。

FPGA の種類

レシーバモジュールの FPGA は、当然、40MHz の LHC クロックで動作させなければならない。まず、現在市販されている Xilinx FPGA で、動作速度の点で条件を満たしている FPGA があるかどうかを調べた。

用いたロジックは、レシーバ FPGA が 1 つの Slave Board を担当するとした場合のものである。また、FIFO は含まれていない。それでも、2550 個のフリップフロップが必要で、換算ゲート数が 36 k ゲートというかなり大きなロジックになった。このため、Spartan シリーズの FPGA に実装することは出来なかった。今回、対象とした FPGA は、XC4000XLA シリーズ、XC4000XV シリーズ、Virtex シリーズである。これらは、現在市販されている Xilinx 社 FPGA の中では最も規模が大きいものである（Appendix D 参照）。

図 5.17 にレシーバモジュールの FPGA の種類と動作周波数の関係を示す。縦軸はクリティカルパスから算出した動作周波数、横軸は対象となる FPGA の大きさである。FPGA の大きさの単位は内蔵されている CLB（Configurable Logic Block）の数で、1 つの CLB が 2 つのフリップフロップに対応する[4] 。FPGA の内部の詳細は Appendix D で述べる。

一般に FPGA が大きければそれだけ信号が通過する配線の長さが長くなるので、動作周波数は遅くなってしまう。が、あまりにも FPGA の規模が小さいと配線の自由度がなくなるので、やはり動作周波数は遅くなる。同じシリーズの中でも若干動作周波数に変化があるのはこのためであると思われる。今回の結果からは XC4000XLA シリーズや XC4000XV シリーズでは 20 MHz 程度でしか動作しないという結論になった。もちろん、これらを用いても配置配線等で工夫をすれば 40 MHz で動作させることが可能になるかもしれない。しかし、十分に余裕をみて 40 MHz で動作させるためには Virtex シリーズの FPGA を用

[4]Virtex シリーズでは実は CLB は 4 つのフリップフロップを含んでいて、かわりに SLICE という単位が 2 つのフリップフロップを含んでいる。以下、Virtex を用いた見積もりの中でも CLB と書いているが、正しくは SLICE という単位である。

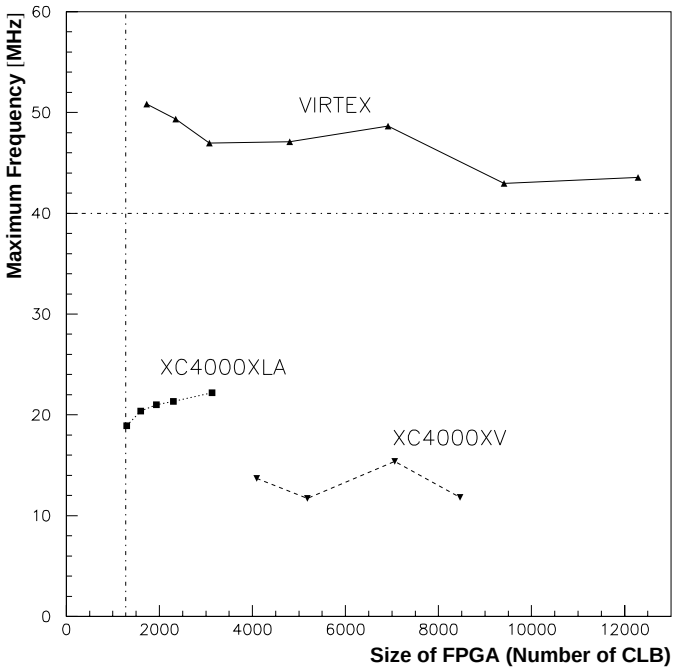

図 5.17： レシーバモジュールの FPGA の種類と動作周波数

いる必要がある。

FIFO の深さ

FPGA の種類に関するシミュレーションから、レシーバモジュールに用いる FPGA としては Virtex シリーズが適当であることが分かった。以下の見積もりでは、FPGA として Virtex シリーズの XCV600−4HQ240 を選んだ。この FPGA は 6912 個の CLB（SLICE）を内蔵しており、かなり規模の大きなものである。

FPGA の種類に関するシミュレーションでは、バスインターフェースの直前にある FIFO は含まれてなかった。そこで、この FIFO を実装した場合に FPGA の規模になるかの見積もりを行った。

図 5.18 と図 5.19 に FIFO の深さを変化させたときのロジックの大きさ（CLB の個数）と動作周波数の変化を示す。この FIFO に収められるデータの大きさは直前のゼロサプレスの結果に依存する。ここでいう FIFO の深さというのは、通常のデータ（TGC のヒットの殆どが 0 でゼロサプレスが有効に働く場合のデータ）を保持できる数、という意味である。

図から分かるように、ロジックの大きさは FIFO の深さに比例して増加する。これは、ロジックに必要なレジスタの個数が FIFO の深さに比例するので当然のことである。一方、FIFO の深さを大きくし、ロジックが大きくなるにつれて、動作周波数は遅くなっていく。単調減少しているわけではないが、これは配置配線がうまく行ったかどうかによって多少動作周波数が変化することによるものと思われる。

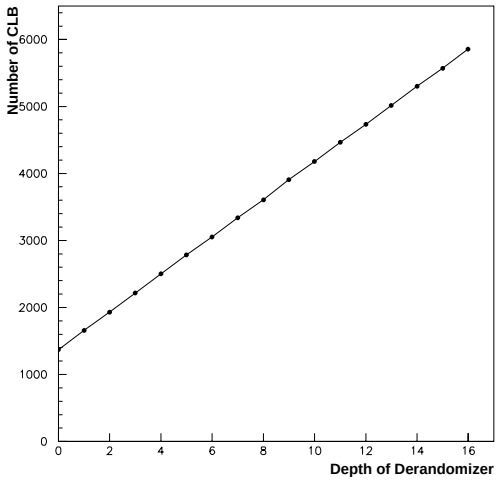

図 5.18： FIFO の深さとロジックの大きさ

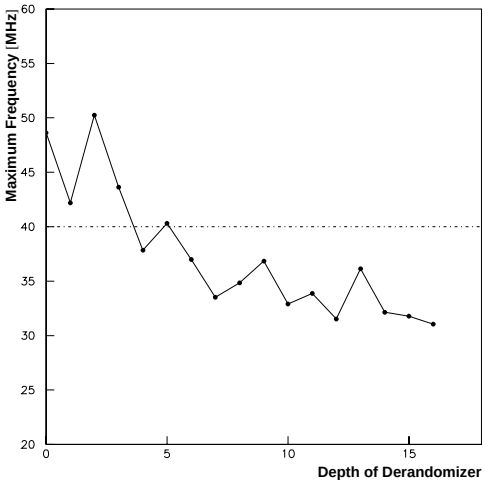

図 5.19： FIFO の段数と動作周波数

1 つのレシーバ FPGA が担当する Slave Board の数

最後に、1 つのレシーバ FPGA が担当する Slave Board の数を変化させてみた。FPGA が複数の Slave Board を担当するようにしたロジックは、Slave Board が 1 つの場合のロジックを並列に並べればよいのだが、この見積もりの際に利用したロジックでは、ゼロサプレスを共通にするなど多少工夫してある。

図 5.20 と図 5.21 に、1 つの FPGA が担当する Slave Board の数を変化させた場合の、ロジックの大きさと動作周波数を示す。基本的な傾向は FIFO の深さを変えた場合と同じである。これらの図から、あまり担当する Slave Board の数を多くするのは無理なことがわかる。

まとめ

FPGA としては Xilinx Virtex シリーズが 40 MHz で動作可能である。しかし、あまり大きな FIFO を実装したり、1 つの FPGA に複数の Slave Board を担当させようとすると、回路規模が大きくなりすぎてしまい動作速度が遅くなってしまう。規模の大きな FPGA は価格が高いこともあり、pt4 モジュールによるプロトタイプの製作の際には 1 つの FPGA に 1 つの Slave Board を担当させ、小さな規模の FPGA を採用することが得策であると思われる。また、FIFO に関しては、今回の見積もりの結果を見る限り数段のものであれば FPGA 内に実装することは可能のようであるが、段数が増えるになるにつれて実装することが厳しくなる。その場合は、FIFO は FPGA に実装しなくても、高速で動作し容量も大きい汎用 IC が存在するのでこれらを外付にすればよい。最終的には、費用や柔軟性を考えるとできれば FPGA 内に実装することが望ましいが、次回のプロトタイプでは FIFO を外付にする方法も試すのがいいかもしれない。

しかし、既に述べたように、今後さらに回路規模が大きく性能の良い FPGA がより安価に入手可能と予想される。さらに、FPGA には CLB を RAM として用いる機能がある。特に Virtex シリーズには、豊富な種類の RAM が用意されている。この機能を利用すれば、動作速度の点では落ちるかもしれないが、回路の規模を小さくできる。これらの結果、最終的には 1 つの FPGA に複数の Slave Board を担当させることも可能になるかもしれない。

最後に、今回は全く考慮に入れていなかったが耐放射線についても考えなければならない。Star Switch の設置場所は TGC を支えるホイールの縁であり、年間 10^{11} cm^{-2} の中性子が到達すると見込まれている。

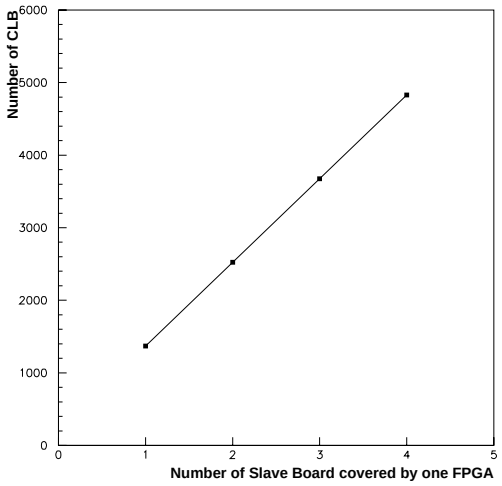

図 5.20： 1 つのレシーバ FPGA が担当する Slave Board の数とその時のロジックの大きさ

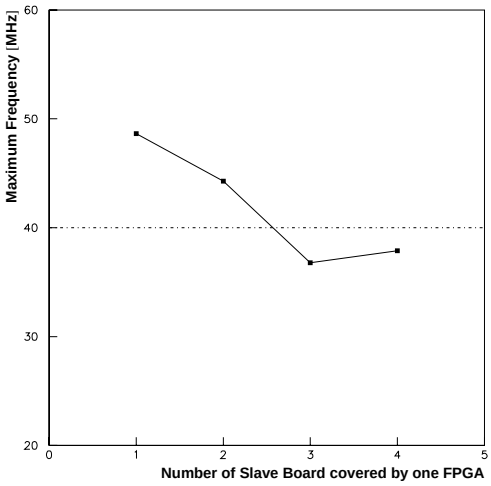

図 5.21： 1 つのレシーバ FPGA が担当する Slave Board の数とその時の動作周波数

FPGA は比較的放射線には弱いので、選んだデバイスが耐放射線性があるかどうかは十分に検証が必要である。ただ、宇宙開発などで耐放射線性のある FPGA の需要が高まるにつれて耐放射線性のある FPGA が市販されるようになってきている。また、通常の FPGA にも比較的 SEU には強いものがある。これらを利用することによって、要求される耐放射線性を満足できると思われる。これらについては Appendix F に記した。なお、耐放射線性のある FPGA が高価すぎるなどの理由でどうしても FPGA での実装が不可能な場合には、ASIC を用いて Star Switch を開発することも考える。

5.5 LS-Link の検証

Slave Board と Star Switch を結ぶ LS-Link では、平均 9 m の距離を LHC クロックと同じ 40 MHz でデータを送信する必要がある。このために、インターフェース規格として LVDS（Appendix G 参照）を用いる。今回、LVDS が LS-Link に要求される性能を満たしているかどうか検証を行った。

まず、LVDS ドライバ DS90C031 を用いて 4 チャネルの TTL → LVDS 変換器を、レシーバ DS90C032 を用いて 4 チャネルの LVDS → TTL 変換器を製作した。レシーバ側に 100 Ω の終端抵抗が取り付けてあることを除けば、基本的には変換用の IC がのっただけの回路である。この変換器を用い図 5.22 のようなセットアップでデータの転送が正しく行われるかを調べた。LVDS 信号を伝えるためのケーブルとして 34 ピンのツイストペアケーブルを用いたが、実際にデータを伝送したのはこのうちの 4 チャネル分である。LS-Link のプロトコルに従って、4 チャネルのうちの 1 チャネルはクロックを、残りの 3 チャネルにはデータを割り当てた。パターンジェネレータで発生させたパターンをロジックアナライザで取り込んでこれらが一致するかどうかを確認した。

この測定をケーブルの長さとクロックの周波数を変えて行った。周波数は 5 MHz 刻みで測定した。その結果を図 5.23 に示す。ケーブルの長さが 10 m 以下のときには 50 MHz が動作の限界であった。今回、TTL LVDS 間の変換に用いた DS90C031 DS90C032 は最高 77.7 MHz で動作するはずであるが、50 MHz でしか動作しなかったのは変換器が手配線で製作したものであったためと思われる。しかし、16 m のケー

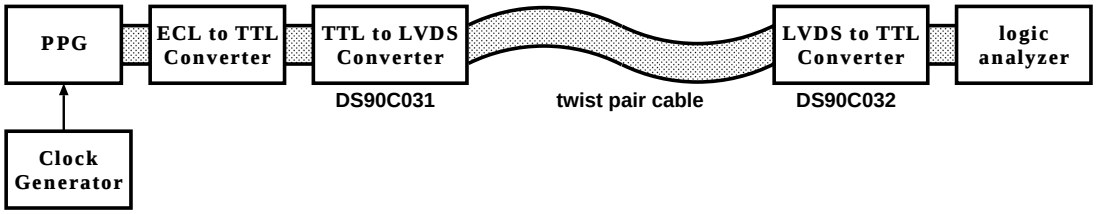

図 5.22： LVDS の性能の検証のためのセットアップ。ケーブルの長さを変えて、動作周波数の限界を測定した。

ブルでも 45 MHz で動作し、LVDS が LS-Link に要求される 40 MHz で 10 m の距離を伝送するという条件は満たすことが分かった。

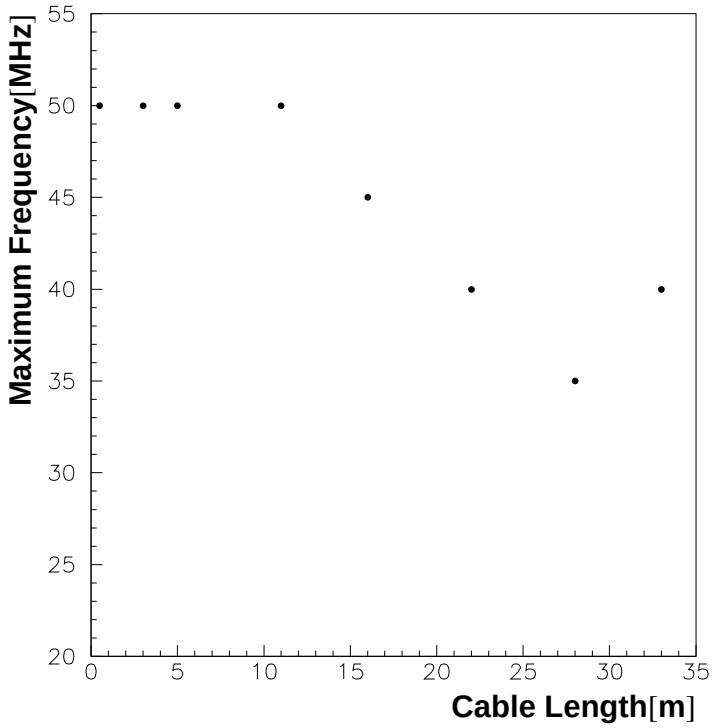

図 5.23： LVDS を用いた伝送テストの結果。ケーブルの長さ、動作周波数の限界の関係。周波数は 5 MHz おきに測定した。

しかし、28 m のケーブルを用いた時と 33 m のケーブルを用いた場合では結果が逆転するなど十分に理解できていない点も多い。これは、28 m のケーブルと 33 m のケーブルの種類がわずかに異なっているのが原因かも知れない。今後は、ケーブルの種類や形状による影響や信号間のクロストークなどの影響について調べ、さらに安定して動作させるにはどのようにすればよいかを調べる必要がある。

第 6 章

まとめと今後

ATLAS の TGC の読み出し系は広範囲にわたる極めて多くのチャネルのデータを収集する必要がある。今回、読み出し系のうちのフロントエンドの部分である Slave Board と Star Switch の設計を行った。

Slave Board については、今後開発する Slave Board ASIC の読み出し系に関連する仕様を決定した。そして、この仕様にしたがってプロトタイプ用の HDL 記述を行った。そして、この HDL 記述に対して論理シミュレーションを行って、記述が正しいことを確認した。また、読み出し系のプロトタイプモジュールである pt3 モジュールを製作し、ここに搭載された FPGA に Slave Board ASIC の読み出し系のロジックを実装した。その結果、必要な機能は正しく動作した。一方で、Slave Board ASIC の読み出し系のロジックを含んだ ASIC を VDEC で試作することによって、ASIC の設計に対する理解を深めることができた。現在、より多くの読み出し系の機能を含んだ ASIC の試作を行っている途中である。一方で、今回用意した HDL 記述のパラメータを最終的に必要な ASIC と同じ規模のものに設定して論理合成を行った。その結果、最終的な ASIC の回路規模は 200 k ゲートであることがわかった。

Star Switch に関しては、データを Slave Board からシリアルで受け取り、ゼロサプレス後にデータバスを介してデータを収集するまでの部分を HDL で記述した。そして、このロジックを FPGA に実装して動作を確認することによって、スキームに問題が無いことを確認した。pt3 モジュールを用いたプロトタイピングでは、完全な大きさのものの 3 分の 1 の大きさのロジックで、30 MHz で動作した。これより、高性能な FPGA を利用すれば LHC クロックの 40 MHz で動作させることは可能であることが分かった。

LS-Link については、LVDS が LS-Link に必要とされる、40 MHz で 10 m の距離を信号を伝送させるという条件を満たすことが分かった。但し、まだ十分に理解ができていないことも多いので、さらに詳しい測定を行う必要がある。

以上の結果から、現在考えている TGC の読み出し系のスキームは全て妥当なものであり、実現可能であることが分かった。

今後の読み出し系の開発予定について述べる前に、まず図 6.1 に TGC エレクトロニクスの開発スケジュールを示す。図の一番下に記されているように、ATLAS TGC エレクトロニクスでは、2000 年 9 月に 1 回目のフルシステムテスト、2001 年 9 月に大量生産前の最終テストを行うという予定になっている。それにあわせて、2000 年 9 月までに TGC のエレクトロニクスの全ての要素についてプロトタイプのボードを作ることになっている。この段階では、今回のようなチャネル数を減らしたりクロック周波数を落したりしたものではなくて、必要最低限の機能を持ったものでなければならない。

Star Switch に関しては、必要な規模の回路が実装可能で、40 MHz のシステムクロックで動作する FPGA を搭載したプロトタイプモジュールを作成する予定である。今回そのためにはどの程度の性能の FPGA が必要かを見積もった。その結果、Xilinx 社の Virtex シリーズは規模としては十分な大きさがあり、40 MHz で動作することが期待されることが分かった。Star Switch のレシーバ部分のロジックは 40 k ゲー

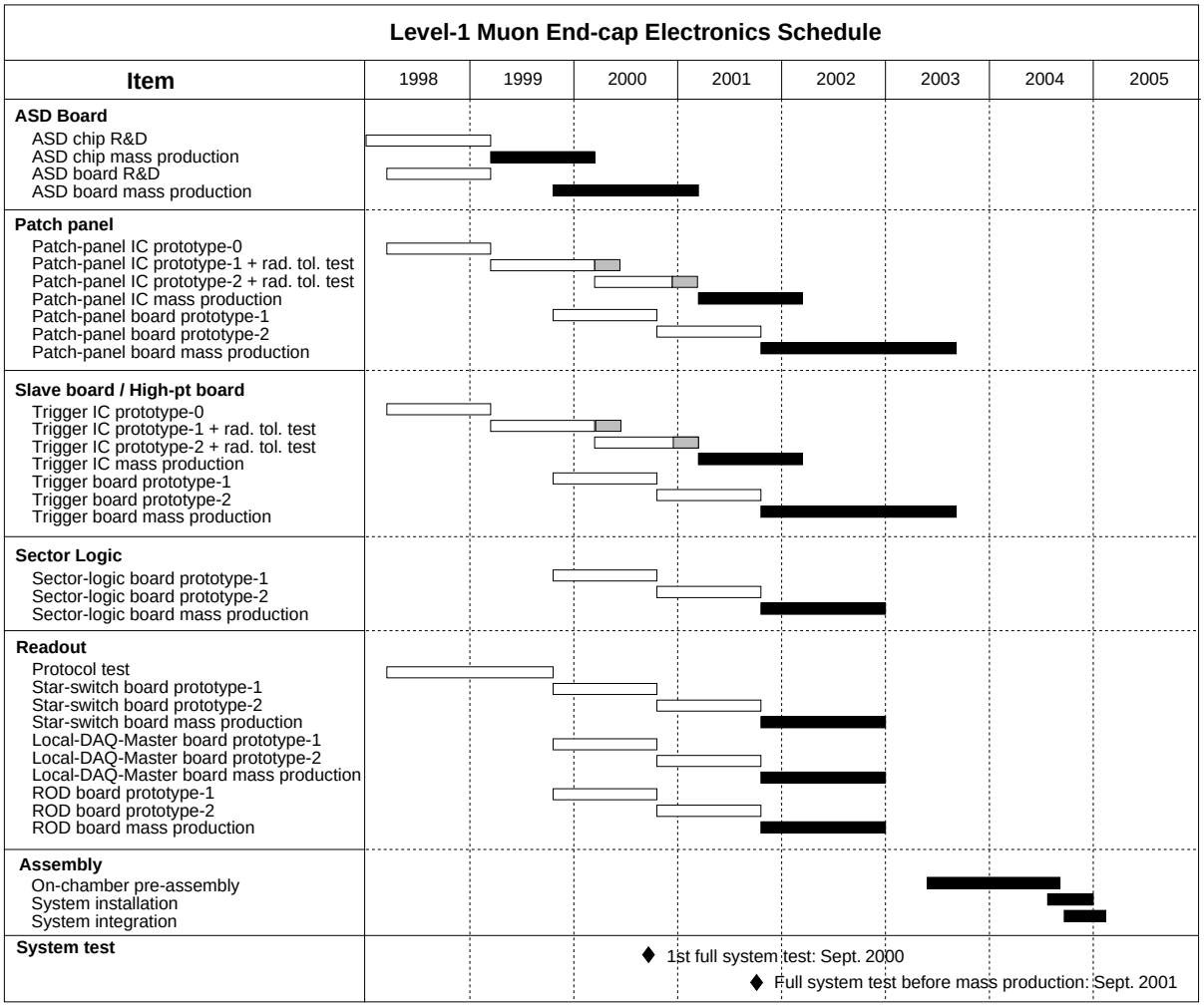

図 6.1： TGC エレクトロニクスの開発スケジュール

トと大きいので、次回のプロトタイプでは 1 つの Slave Board に対する処理を 1 つの FPGA で行うという方式を採用することとした。今後は、次回のプロトタイプでデータバス部分のプロトコルの検証を行いながら、さらに詳細な設計を行う必要がある。

Slave Board の読み出し系に関しては、前回試作した ASIC の検証を行って、設計に問題が無かったかを確認する。そして、再び VDEC を利用してもう少し規模の大きなプロトタイプ用 ASIC を作成して、性能を検証する予定である。その後は、2000 年度中にゲートアレーを用いて Slave Board ASIC の試作を行う予定である。このときには今回設計した読み出し系の部分にトリガー行列を加えた完全なプロトタイプを試作する。これが、1 回目のフルシステムテストのためのプロトタイプとなる。

現在の進行状況をみると、2000 年の 9 月に 1 回目のフルシステムテストを行うのは厳しい状況ではある。ただ、このスケジュールではエレクトロニクスの大量生産に 2 年近い期間を予定しているが、実際にはそれほどはかからない。したがって、2000 年 9 月からそう遠くないうちに 1 回目のフルシステムテストを行えば、エレクトロニクスの開発に関しては問題がないと思われる。それゆえ、今後は 1 年程度の時間をかけて、Slave Board 及び Star Switch の完全なプロトタイプを開発することが急務となる。

謝辞

本研究を最初から最後まで指導して下さいました指導教官の坂本宏助教授に深く感謝いたし ます。また、常に丁寧にご指導下さいました KEK（高エネルギー加速器研究機構）の佐々木修氏、研究を進める上で惜しみない協力をしてくださいました KEK 回路室の池野正弘氏には心よりお礼を申し上げたいと思います。また、モジュールの製作を行って下さいました株式会社ジー・エヌ・ディーの宮沢正和氏にお礼を申し上げたいと思います。ICEPP（東京大学素粒子物理国際研究センター）の小林富雄氏、蓮子和巳氏、神戸大学の藏重久彌氏、東京都立大学の福永力氏、KEK の近藤敬比古氏、岩崎博之氏、大須賀鐵雄氏、新井康夫氏、その他 ATLAS 日本グループの方には様々な面で支援をいただき、心より感謝しています。さらに、東京都立大学の狩野博之氏、東京大学の佐藤構二氏、戸谷大介氏、仁木太一氏、神戸大学の一宮亮氏とは共に TGC のエレクトロニクスの開発を行ってきましたが、多くの助言を頂き、多くの学ぶことがありました。深く感謝いたし ます。最後になりましたが、研究を手伝ってくれた本田洋介氏、新家英太郎氏をはじめとする高エネルギー研究室の方々には心よりお礼を申し上げたいと思います。

Appendix A

プロジェクト管理

A.1 プロジェクト管理の概要

ATLAS のような大規模な実験を統制するには「プロジェクト管理」と呼ばれる方法が必要となってくる。ここでプロジェクトとは

- 人によって実行されている。
- 有限なリソースに制限されている。
- 計画され、実行され、制御される。
- 始まりと終わりがある。
- 得られる成果が他の活動と何らかの形で異なる。

ような点で特徴づけられる一般的な活動を指す。
プロジェクト管理では以下のような観点からの管理が必要とされる [17]。

統合管理 プロジェクトの様々な要素を適切に統合する。

スコープ管理 スコープとはプロジェクトに伴う成果や作業のことである。プロジェクトを成功させるのにどのような作業が必要かということを見極める。

時間管理 プロジェクトの時間を管理する。

費用管理 予算内でプロジェクトが行えるように、経費を管理する。

品質管理 プロジェクトで得られる成果が十分な品質、性能を保つようにする。

人的資源の管理 プロジェクト内の人的資源を最も効果的に利用するように、作業に人を配置する。

情報伝達管理 プロジェクトに関する情報をよりよく伝達させるようにする。

リスク管理 プロジェクト内のリスクを分析し、それに対応する。

調達管理 プロジェクトに必要な物資などの調達を管理する。

これらの管理のために必要なプロセス（過程）が多く存在するが、これらのプロセスは独立しているわけではなく、相互に関連し合っている。
特にプロジェクトの計画の段階に関して言えば、図 A.1 のようなプロセスを経るのがよい。

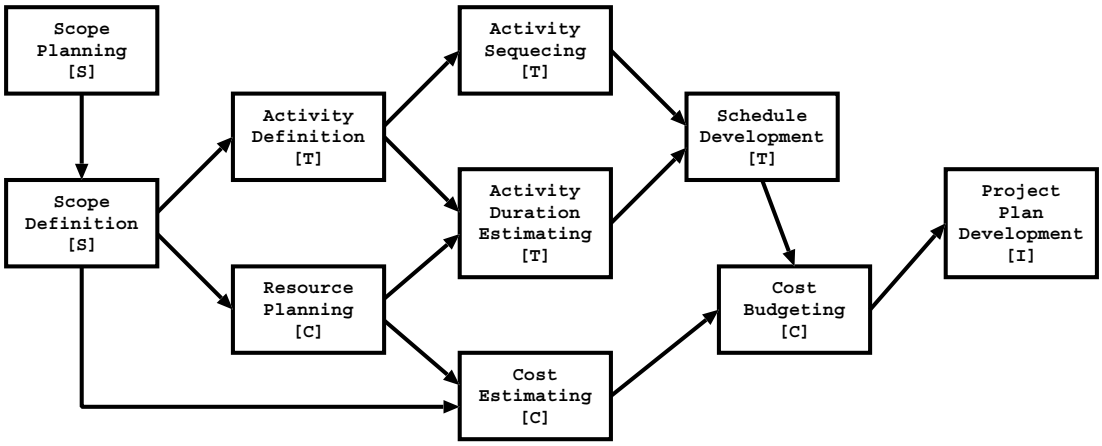

図 A.1： プロジェクトの計画の段階でのプロセス。括弧内の文字はそれぞれ S = Scope, T = Time, C = Cost, I = Integration に関する管理を表す。

Scope Planning プロジェクトの目的、プロジェクト成功の基準を文書化する。

Scope Definition その文書を元にプロジェクトを分割し、WBS（Work Breakdown Structure：作業階層構造）とよばれる階層構造に分割する。WBS については次節を参照

Activity Definition WBS をさらに具体的な作業に分け、作業リストを作成する。

Resource Planning WBS をもとにどういうリソースが必要か考え、必要なリソースのリストを作る。

Cost Estimating WBS 及びリソースのリストから必要な経費を見積もる。

Activity Sequencing 作業リスト内の作業の相互関係や依存性を調べ、これらをプロジェクトネットワークダイアグラムの形で図示する。ダイアグラムの例を図 A.2 に示す。

Activity Duration Estimating 作業リスト内の作業を完了するのに、どれくらいの期間を必要とするかの見積もりをたてる。

Schedule Development 上記の見積もりを基にプロジェクトのスケジュールをたてる。スケジュールは見やすいように、日付入りのプロジェクトネットワークダイアグラムやガットチャート（図 A.3 ）の形で表示する。

Cost Budjeting WBS やプロジェクトのスケジュールをもとに、各々の作業に対し予算の配分を行う。

Project Plan Development 以上の様々な計画をもとに、プロジェクトの計画を作成する。

これらのプロセスと並行して、品質保証や情報伝達についても計画をたてる必要がある。以上は、プロジェクトの計画の部分についての方針である。プロジェクトを実行したり、制御したりする際には、同様に多くのプロセスを経る必要がある。これについては文献 [17] などを参照のこと。

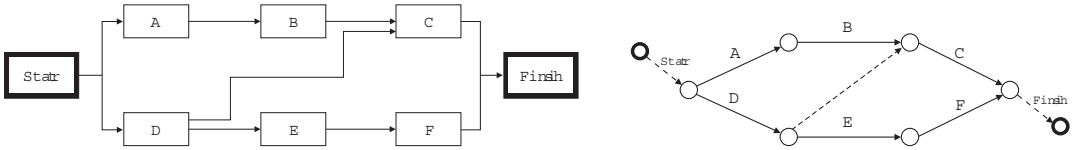

図 A.2： プロジェクトネットワークダイアグラムの例

左は PDM（Precedence Diagramming Method）で、四角が作業を表す。右は ADM（Arrow Diagramming Method）で矢印が作業を表す。

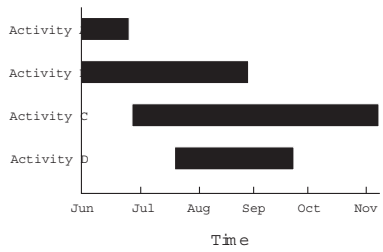

図 A.3： ガットチャートの例

A.2　ATLAS におけるプロジェクト管理

ATLAS 実験におけるプロジェクト管理も基本的な考え方は同じである。プロジェクトを階層構造に分割し、それぞれに必要な作業を調べ、それらについてスケジュールを決定するのである。ATLAS では複雑なプロジェクトをより単純化するために、図 A.4 に示すような 4 つの階層構造を導入している [9]。

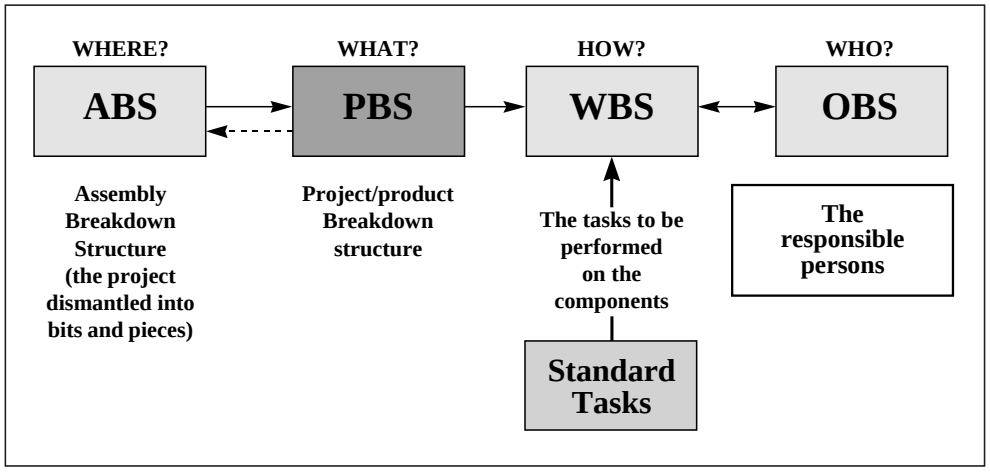

図 A.4： ATLAS の 4 つの階層構造

ABS（Assembly Breakdown Structure）は BOM（Bill Of Material）とも呼ばれ、物理的な物の集まりの階層を表す。ABS に分割することにより、実際にどういう材料、部品、物が必要かがわかる。PBS

（Product Breakdown Structure 又は Project Breakdown Structure）はプロジェクトを分解し、より細かい構成要素へと分割、階層化したものである。前節で登場した WBS は、ATLAS の 4 つの階層構造のなかではむしろ PBS に相当する。一方、ATLAS でいう WBS は具体的な作業を指していて、各々の PBS の要素に対し必要な作業を列挙したものである。この意味では、前節でのべた作業リストに近いものである。OBS（Organizational Breakdown Structure）は、どの作業をどの組織や人に割り当てるかを示す階層構造である。

このうち PBS はプロジェクトの根幹をなすもので、ATLAS では、ATLAS Detector, ATLAS General Facilities, ATLAS Assembly & Test Areas, ATLAS Offine Computing, ATLAS Technical Coordination の 5 つの主 PBS に分割されている。これらのそれぞれの内部でさらに分割は行われており、例えば、ATLAS Detector は図 A.5 のように分割されている。これらの木構造をたどって行くと、非公式ながら、Slave Board や Star Switch のような要素まで分解されている。これらの要素それぞれについて、必要な作業の割当とスケジュールの決定が行われる。ATLAS 実験の参加者は、これらのスケジュールにあわせて ATLAS 実験に必要な作業を行うこととなる。

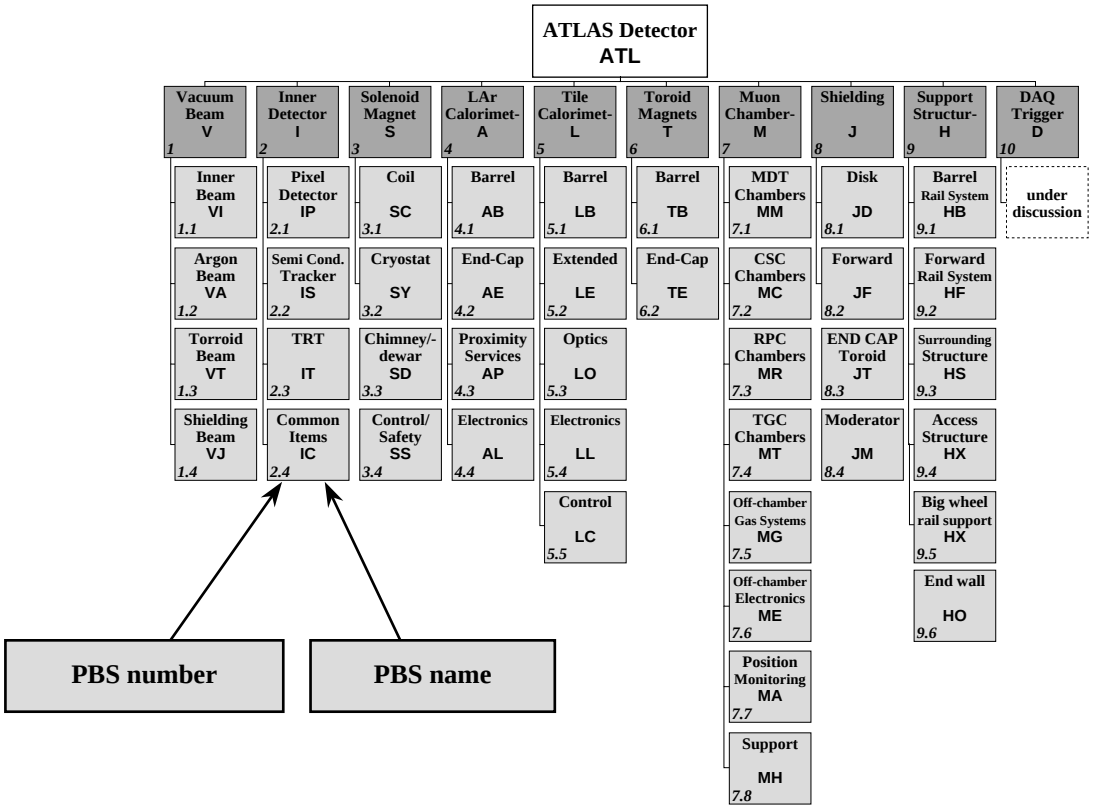

図 A.5： ATLAS Detector の PBS

Appendix B

JTAG（IEEE1149.1）

B.1　JTAG とは

従来の基板の試験法は、原理的には基板上に物理的にプローブをあてて行うものであった。しかし、IC のピン間隔が狭くなったこと、多層基板が開発されたことによってこの方法は困難になってきた。そこで考案された方法の一つが JTAG（Joint Test Action Group）という規格である。この規格は IEEE（Institute for Electrical and Electronics Engineering）1149.1 として定められている [18]。

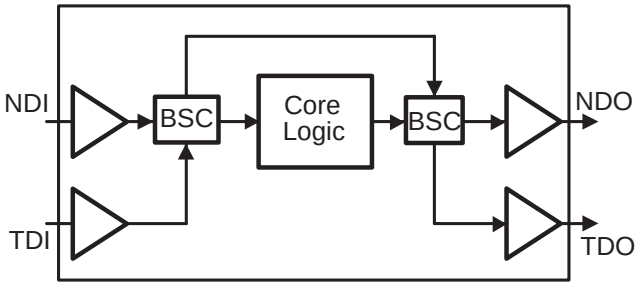

図 B.1： JTAG の基本的な考え方。IC の入出力ピンと論理を担う部分の間に、シフトレジスタを配置する。このシフトレジスタにデータを取り込んだりすることにより、診断を行う。

JTAG による試験法の概念は図 B.1 に示す通りである。IC の入出力ピンと論理を担う部分の間に、シフトレジスタを配置するというものである。通常は入出力ピンと論理部分は直接接続されているだけであるが、必要に応じて、シフトレジスタにピンのデータが取り込まれたり、逆にシフトレジスタ上の値を強制的に出力させることが出来る。そして、シフトレジスタをシフトさせていることにより、入出力ピンの値を読み出したり、設定したりするのである。なお、JTAG では、IC の境界（boundary）にシフトレジスタを配置するので、このような試験法をバウンダリスキャン（boundary scan）と呼ぶ。

B.2　JTAG の概要

JTAG（IEEE1149.1）に対応したデバイスは図 B.2 のような構造を持つ。バウンダリスキャンのためには、TCK TMS TDI TDO の 4 本のピンが必要である。このうち、TCK TMS TDI は入力、TDO は 3 ステートの出力ピンである。この他に、TRST* という 5 本目のピンがオプションとして用意されてい

る。TRST* は後述する TAP コントローラの状態を Test-Logic-Reset に戻すためのリセット用の信号線である。

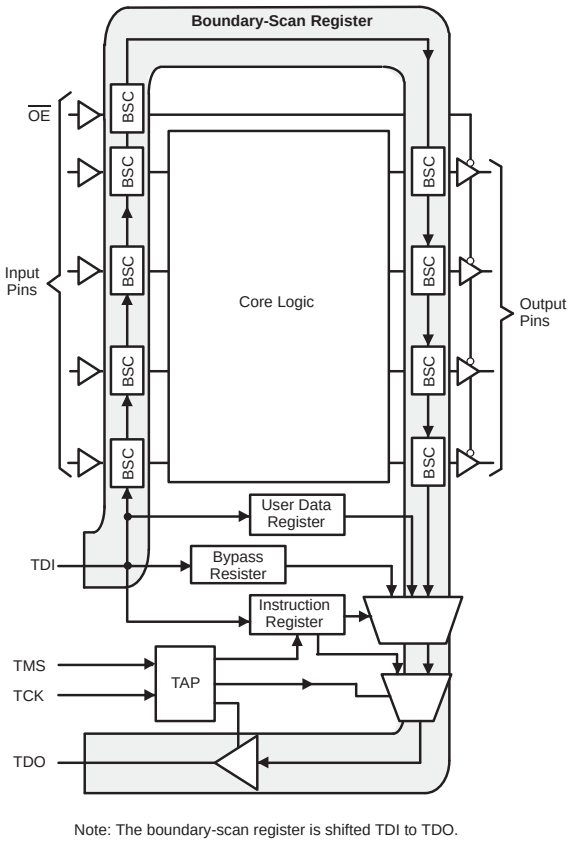

図 B.2： JTAG バウンダリスキャンの構成

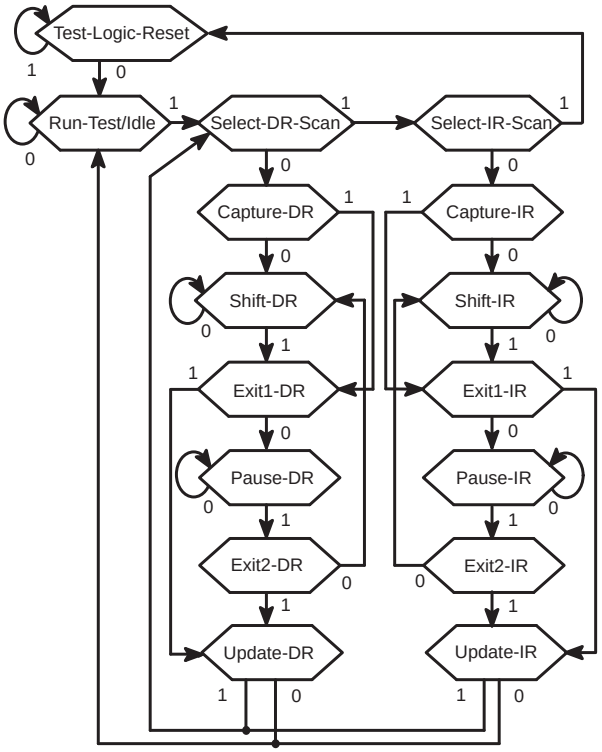

図 B.3： JTAG の TAP ステートダイアグラム

各入出力ピンの内部には Boundary Scan Cell（BSC）と呼ばれるブロックが配置され、これらは内部で接続されていて Boundary Scan Register（BSR）と呼ばれるシフトレジスタを構成している。TDI と TDO は、 BSR の他いくつかの経路のシフトレジスタで結ばれている。このうち Instruction Register（IR）はどのようなバウンダリスキャンを行うかというインストラクション（命令）を保持する。IR に対して、その他のシフトレジスタは Data Register (DR) と呼ぶ。DR には BSR の他、1 ビットのバイパスレジスタ (BYPASS Register)、デバイスコード識別のためのデバイス識別レジスタ (Device ID Register) などある。また、IC の製作者は新たにシフトレジスタを追加してもよい。

一方、これらの各々の要素を制御するために、有限ステートマシンの TAP (Test Access Port) コントローラが用意されている。TAP コントローラには TCK と TMS が接続され、TCK の立ち上がりエッジでの TMS の値によって、状態を遷移する。TAP コントローラのステートダイアグラムを図 B.3 に示す。Test-Logic-Reset はバウンダリスキャンをしていない状態、すなわち通常の状態である。一般的には以下のような使い方となる。まず最初にインストラクションを設定する。IR の値を設定するためには、Shift-IR ステートに行き TMS を 0 に保った状態で TDI からデータをシフトさせる。次に、Shift-DR 状態に行き TDI からデータをシフトすると、IR にセットされた命令に応じた DR が選択され、そこにデータがシフトされる。これにより目的のバウンダリスキャンが実行される。

図 B.4 に出力ピン用 BSC の回路を示す。通常は IC の内部からの信号はそのまま外部に出力される。

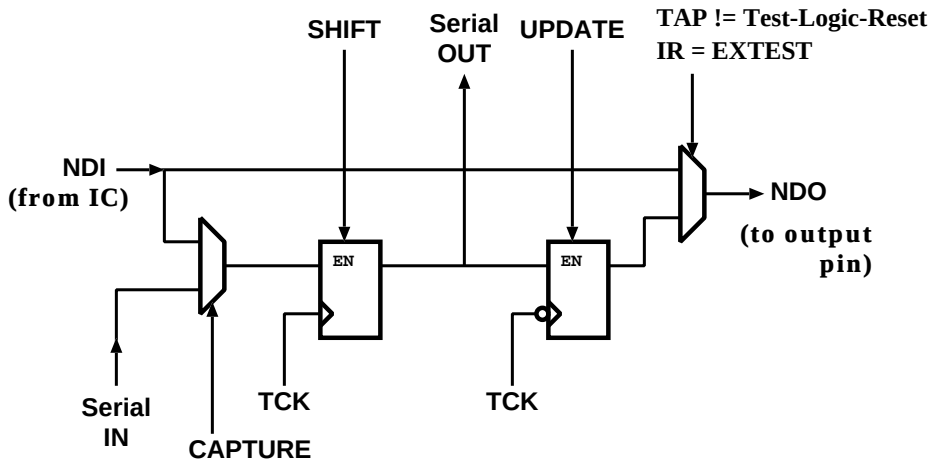

図 B.4： BSC 回路。出力ピン用。BSR に沿ってデータをシフトさせる際には、Serial IN から Serial OUT へと出力する。BSC を用いることによってデータを取り込んだり、特定の値を出力させることが可能である。

出力ピンの値を JTAG 経由で読み出したいときは、TAP コントローラを Capture-DR 状態にする。すると、IC の内部からの信号が Shift Register に取り込まれる。その後 TAP コントローラを Shift-DR 状態にすることにより、Shift Register は Serial IN から Serial OUT と書かれた経路に沿ってシフトされる。最終的にこれらのデータは IC の TDO ピンから出力される。

逆に出力ピンの値を特定の値に設定したいときは次のような操作を行う。まず、TAP コントローラを Shift-DR 状態にして、Shift Register に TDI からデータをシフトされる。次に、Update-DR 状態にすることによってデータは Update Register に取り込まれる。出力ピンに Update Register の値を出力すれば、出力ピンを特定の値に設定したことになる。実際に、Update Register の値をピンに出力するかどうかは TAP の状態と後述するインストラクションにより決まる。出力ピン用の BSC の場合には、TAP が Test-Logic-Reset 状態でなくインストラクションが EXTEST であれば出力される。

なお、一つの基板上に複数のデバイスが存在する場合には図 B.5 のようにシリアルデバイスを接続すればよい。それゆえ、デバイスの個数にかかわらず、基板に 4 本（TRST* を含めると 5 本）のバウンダリスキャン信号線を追加すればよいことになる。

B.3 JTAG のインストラクション

JTAG のインストラクションには、3 つの必須命令といくつかのオプション命令が定義されている。この他、設計者は新しいインストラクションを自由に定めてよいことになっている。例えば、TGC エレクトロニクスにおいてはパラメータの設定のために JTAG を利用するが、その際にはパラメータ設定用のインストラクションを定めればよい。以下に JTAG のインストラクションを挙げる。なお、以下「命令を実行する」とは、「その命令に対応するインストラクションコードを IR に設定する」ということを意味する。

BYPASS 必須命令。命令コードは全ビット 1 のコード。BYPASS 命令を実行しても、デバイスは通常どおり動作する。BYPASS 命令を実行すると、バイパスレジスタが選択され、TDI から入力されたデータはバイパスレジスタを経て TDO に出力される。要は、TDO の値は TDI の値の 1 クロック遅れたものになる。

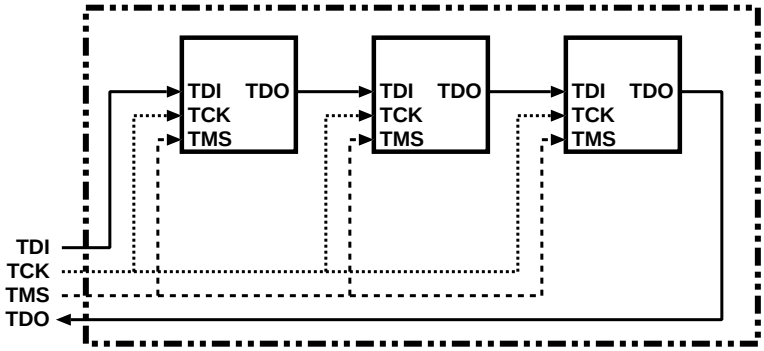

図 B.5： デバイスが複数存在する場合の JTAG 信号線の接続。TDI, TDO をシリアルにつなげばよいので、デバイスの数に関わらずボードには 4 つのピンがあればよい。

SAMPLE/PRELOAD 必須命令。命令コードは定義されていない。BYPASS 命令を実行しても、デバイスは通常どおり動作する。SAMPLE/PRELOAD 命令を実行すると BSR が選択される。TAP ステートが Capture-DR の時に、入出力ピンの値が BSC に取り込まれるので、入出力ピンの値をサンプルすることが可能である。また、この命令は論理には影響を与えないので、BSC に特定の値を前もって設定するときにも用いられる。

EXTEST 必須命令。命令コードはかつては全ビット 1 と定義されていた。EXTEST 命令は SAMPLE/PRELOAD 命令に似ているが、出力ピンには BSC に設定した値が強制的に出力される。そのため、EXTEST 命令を実行すると、デバイスはテスト状態に入ってしまう。EXTEST は配線試験等の際に用いられる。

INTEST オプション命令。INTEST 命令は EXTEST 命令に似るが、デバイスの内部試験を行うためのものである。

RUNBIST オプション命令。デバイスを自己診断状態にする。

CLAMP オプション命令。出力ピンは BSC の設定値になるが、DR としてはバイパスレジスタが選択される。

HIGHZ オプション命令。出力ピンはハイインピーダンスになるが、DR としてはバイパスレジスタが選択される。

IDCODE オプション命令。DR としてデバイス識別レジスタを選択し、32 ビットのデバイス情報を出力する。

USERCODE オプション命令。DR としてデバイス識別レジスタを選択し、32 ビットのユーザー情報を出力する。

B.4 TGC エレクトロニクスにおける JTAG

TGC のエレクトロニクスではパラメータを設定と読み出しを行う必要がある。このためには、パラメータ設定用の新しいインストラクションを定めればよい。TGC エレクトロニクスでは、Patch Panel ASIC や Slave Board ASIC など多くのデバイスで JTAG を用いることになるので、その際のインストラクショ

ンコード（命令コード）など設計者の裁量に任せられている部分を出来る限り統一すべきである。このような観点から、TGC エレクトロニクス内では以下の様な非公式の取り決めを行っている。

- インストラクションコード（命令コード）の幅は 8 ビット。
- ユーザレジスタは MSB（Most Significant Bit）を TDO 側に持ってくる。すなわち、シリアルにデータを入力する際には MSB から入力する。
- バウンダリスキャンレジスタは、入力ピンが TDI 側、出力ピンが TDO 側に来る。
- プライベートなインストラクションコードについて。下位 2 ビットが 00 のインストラクションは、パラメータの読み出し、01 のインストラクションは書き込みを意味する。また、プライベートなインストラクションの上位 5 ビットには 1 つ以上 1 が含まれていなければならない。
- 既に定義されているインストラクションについては、インストラクションコードを表 B.1 のように定める。

Instruction	ID Code	Comments
BYPASS	1111_1111	necessary
SAMPLE/PRELOAD	0000_0001	necessary
EXTEST	0000_0000	necessary
INTEST	0000_0010	optional
RUNBIST	0000_0011	optional
IDCODE	0000_0100	optional
USERCODE	0000_0101	optional
CLAMP	0000_0110	optional
HIGHZ	0000_0111	optional

表 B.1： TGC エレクトロニクスにおけるインストラクションコード

Appendix C

Slave Board についての補遺

Slave Board の仕様については 4.4 で触れたが、そこで省略したことについてはここで述べる。

C.1 Slave Board ASIC の仕様についての補遺

C.1.1 入出力信号線

表 C.1 に Slave Board ASIC の入出力信号線の一覧を示す。表中の数値は必要なピンの数である。これらの入出力信号線の働きは以下の通りである。

CLK, BCR, ECR, L1A TTC から Patch Panel 経由で受け取るクロック及び同期用の信号。これらの信号については 2.2.5 を参照。

TGCSIG, RADJSIG, LADJSIG Patch Panel から来る TGC のヒット情報。RADJSIG, LADJSIG は隣接する Slave Board との共有信号。

TRGOUTA, TRGOUTB High-p_{T} Board への出力信号（コインシデンスの結果）。ASIC の外で LVDS に変換される。

LSLDCLK, LSLDSYN, LSLDDAT Star Switch からの入力。LS-Link。ASIC の外部で LVDS からの変換が必要。詳細は 4.6 を参照。

LSLUCLK, LSLUSYN, LSLUDAT Star Switch への出力。ASIC の外部で LVDS への変換が必要。詳細は 4.6 を参照。

JTAG 関連信号 Star Switch への接続と Patch Panel への接続がある。詳細は 4.3.6 を参照。

ADDR Slave Board を識別するための 8 ビットのアドレス。外部にディップスイッチを用意する。

TYPE 5 種の Slave Board の切り替え。

アドレスについては、ここではとりあえずディップスイッチで設定するということにしているが、実際に 3000 個近い Slave Board に対してディップスイッチの設定をいちいち行うのは大変である。何かよい方法を考えるべきである。一方、5 種の Slave Board の切替えについては、いずれにせよボード自体は 5 種類用意することになると思われるので、特定のピンをプルアップしたりプルダウンすれば問題ない。

Signal Name	Abbreviation	Dir.	D/W	T/W	D/S	T/S
LHC Clock	CLK	IN	1	1	1	1
TGC Signal	TGCSIG	IN	128	96	128	128
Adjacent Signal (Left)	RADJSIG	IN	16	6	16	0
Adjacent Signal (Right)	LADJSIG	IN	16	6	16	0
Trigger Output A	TRGOUTA	OUT	9	9	8	20
Trigger Output B	TRGOUTB	OUT	9	9	8	20
Bunch Crossing Reset	BCR	IN	1	1	1	1
Event Counter Reset	ECR	IN	1	1	1	1
Level 1 Accept	L1A	IN	1	1	1	1
LS link Clock Input	LSLDCLK	IN	1	1	1	1
LS link Sync Input	LSLDSYN	IN	1	1	1	1
LS link Data Input	LSLDDAT	IN	1	1	1	1
LS link Clock Output	LSLUCLK	OUT	1	1	1	1
LS link Sync Output	LSLUSYN	OUT	1	1	1	1
LS link Data Output	LSLUDAT	OUT	2	2	2	2
JTAG TCK	TCK	IN	1	1	1	1
JTAG TMS	TMS	IN	1	1	1	1
JTAG TDI	TDI	IN	1	1	1	1
JTAG TDO	TDO	OUT	1	1	1	1
JTAG TRST*	TRST*	IN	1	1	1	1
JTAG TCKOUT	TCK	OUT	4	4	4	4
JTAG TMSOUT	TMS	OUT	4	4	4	4
JTAG TDIOUT	TDI	OUT	4	4	4	4
JTAG TDOIN	TDO	IN	4	4	4	4
JTAG TRSTOUT*	TRSTOUT*	OUT	4	4	4	4
Module Address	ADDR	IN	4	4	4	4
Module Type	TYPE	IN	3	3	3	3
Total			221	169	219	211

D/W = Doublet Wire　D/S = Doublet Strip
T/W = Triplet Wire　T/S = Triplet Strip

表 C.1： Slave Board ASIC の入出力信号一覧

C.1.2　パラメータ

Slave Board ASIC で扱わなければならないパラメータを表 C.2 に挙げる。既に述べたように、FE エレクトロニクスへの要求として、これらはソフトウェア的に設定と読み出しが可能でなければならない。

これらの他に、トリガーに関連するパラメータやトリガーに関する統計結果などが必要となるかもしれない。

Parameter Name	NBit	Setting	Readout
Module Type	3	EXT	–
Module Address	4	EXT	LS,JTAG
Phase Adjust	6*	JTAG	JTAG
Trigger Mask Bits	160*	JTAG	LS,JTAG
LVL1 Buffer Depth (Signal)	7	JTAG	JTAG
LVL1 Buffer Depth (BCID)	7	JTAG	JTAG
LVL1 Buffer Depth (Trigger Output)	7	JTAG	JTAG
Overflow Counter	8	–	LS
SEU Frequency	1	–	JTAG

EXT = External Pins, LS = LS-Link

* : Value for Wire Doublet Slave Board

表 C.2： Slave Board ASIC のパラメータ。Setting, Readout の欄で、EXT は外部ピンによるもの、JTAG は JTAG によって設定や読み出しを行うもの、LS は LS-Link を通して読み出しを行うもの。

C.2 Slave Board ASIC の回路と HDL 記述

Slave Board ASIC の仕様に基づいて HDL での記述を行った。その回路と HDL 記述の一部を示す。

図 C.1 と図 C.2 に Slave Board ASIC 内の回路のブロック図を示す。前者は JTAG によるバウンダリスキャンの部分で、この図の core logic と書かれた部分と後者（図 C.2 ）が対応する。ここで挙げたブロックは図 4.9 で挙げた構成要素と対応している。

以下、これらの各ブロックの Verilog ソースコードの一例を載せる。紙面の関係上、最初にパラメータの定義を載せた後、ここではデランダマイザーと PSC の部分のみを載せることとする。

```
//================================================================
//------------------------------
//  readout_h.v
//------------------------------

// definition of global parameter
‘define NSIG2W          128     // # input of Slave Board
‘define NLADJ2W         16      // # left side adjacent signal
‘define NRADJ2W         16      // # right side adjacent signal
‘define NADJ2W          (‘NLADJ2W+‘NRADJ2W)     // # adjacent signal
‘define NBMASK2W        (‘NSIG2W+‘NADJ2W)       // # signal to mask
‘define NBBCID          8       // width of BCID counter
‘define NBL1ID          8       // width of L1ID counter
‘define NBADDR          8       // width of module address
‘define NBHEAD          8       // width of message header
‘define NBTAIL          8       // width of message trailer
‘define NBTRGP          9       // trigger primitive bit
```

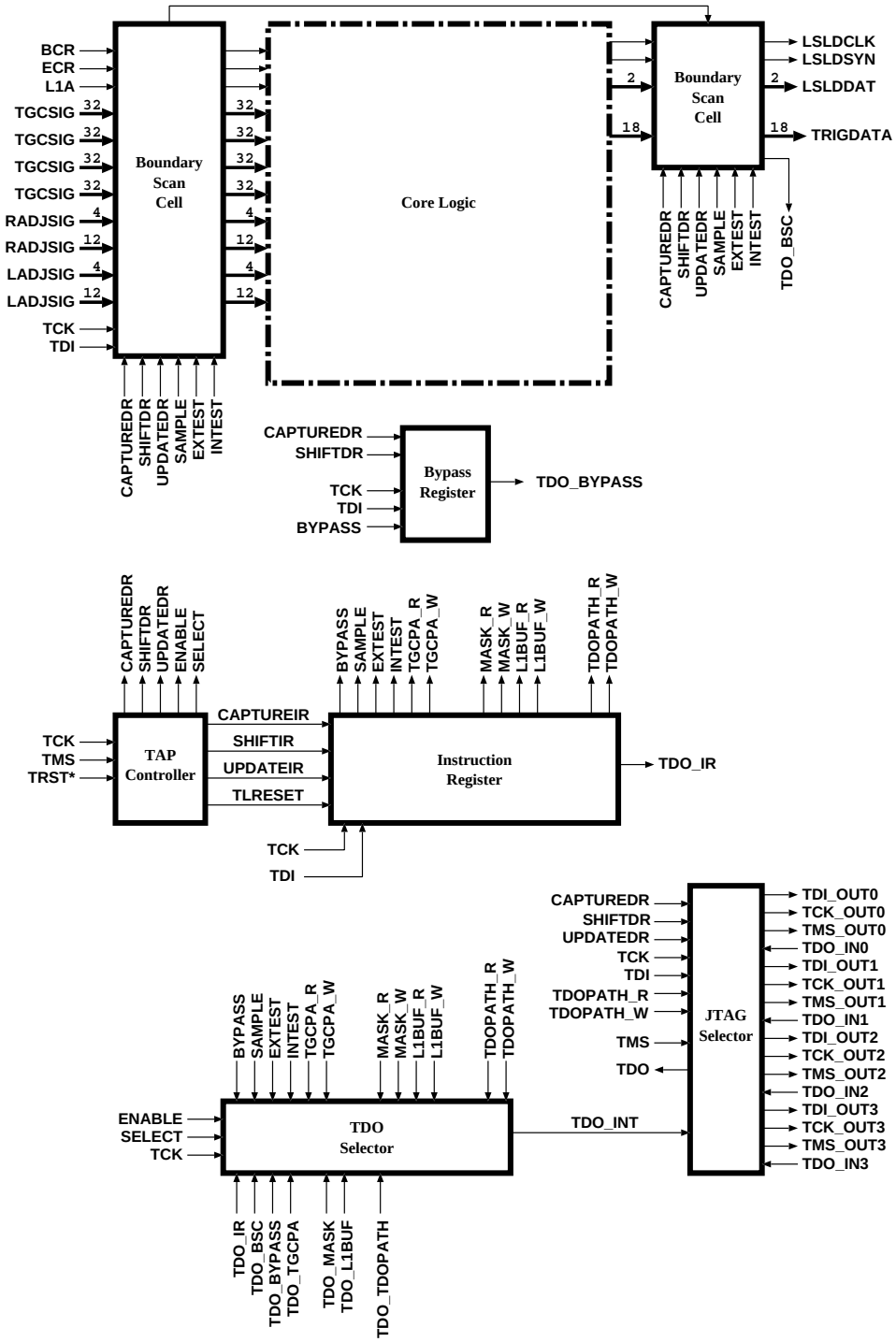

図 C.1： Slave Board ASIC 回路のブロック図（バウンダリスキャン）。矢印上にかかれた数字は信号線のビット幅である。JTAG selector は Patch Panel への JTAG の経路の選択を行う。この部分を除けば、通常の JTAG 用の回路と同じである。この図の core logic とかかれた部分には、図 C.2 に示される回路が対応する。

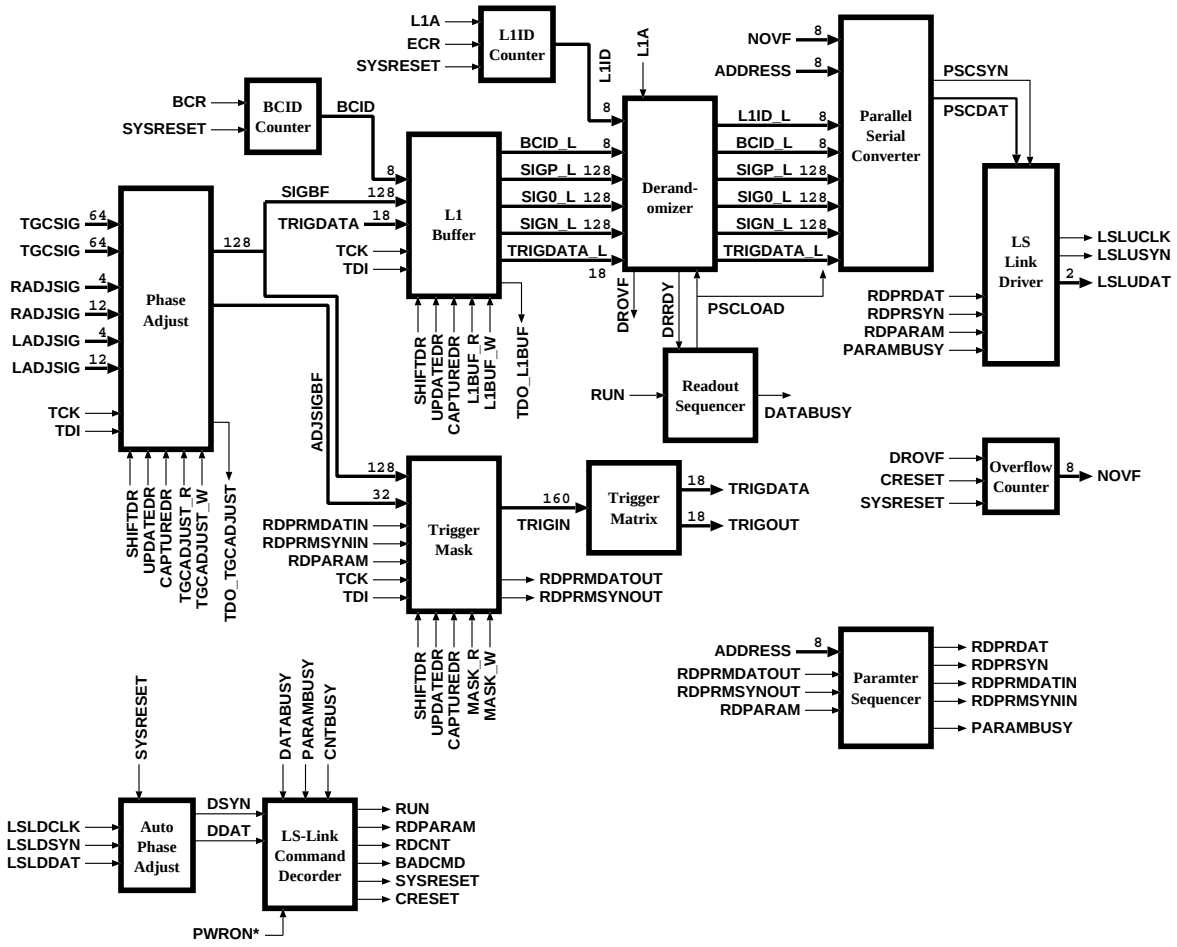

図 C.2： Slave Board ASIC 回路のブロック図（中心部）。ワイヤダブレット用のものを想定してある。矢印上にかかれた数字は信号線のビット幅である。

```
‘define NBTRGO          (‘NBTRGP*2)     // # output of trigger
‘define NBTRGL1         ‘NBTRGO
‘define NBL1SIG         (‘NSIG2W+‘NBBCID+‘NBTRGL1)
                                // # output of L1 buffer
//‘define NBDRDIN               (‘NBL1SIG+‘NBL1ID)      // width of derandomizer
‘define NBPSCIN         ‘NBHEAD+‘NBADDR+‘NBBCID+‘NBL1ID+2*‘NSIG2W+‘NBTAIL
                                // width of PSC input signal
‘define NBL1BPRM        7       // width of L1B length parameter
‘define NLDRDB          6       // depth of derandomizer
‘define NBOVF           8       // overflow counter

// auxiliary
‘define NBDRDIN_CNT     8       // width of counter to count NBDRDIN
‘define NLDRDB_CNT      3       // width of counter to count NLDRDB
```

```
//================================================================
//----------------------------
// sb_derand.v
//
// Derandomizer for Slave Board
//----------------------------

`include "readout_h.v"

module sb_derand(
        sysclk, sysreset, l1a, pscload, drovf, drrdy,
        bcid_l, l1id_l, trigdata_l, sigp_l, sig0_l, sign_l,
        bcid_d, l1id_d, trigdata_d, sigp_d, sig0_d, sign_d
);

parameter NBDRDIN = `NBBCID + 3*`NSIG2W + `NBL1ID + `NBTRGL1;

input  sysclk, sysreset, l1a, pscload;
output drovf, drrdy;
input [`NBBCID-1:0] bcid_l;
input [`NSIG2W-1:0] sigp_l, sig0_l, sign_l;
input [`NBL1ID-1:0] l1id_l;
input [`NBTRGL1-1:0] trigdata_l;
output [`NBBCID-1:0] bcid_d;
output [`NSIG2W-1:0] sigp_d, sig0_d, sign_d;
output [`NBL1ID-1:0] l1id_d;
output [`NBTRGL1-1:0] trigdata_d;

wire [NBDRDIN-1:0] drdin;
wire [NBDRDIN-1:0] drdout;

assign drdin = { bcid_l, sigp_l, sig0_l, sign_l, l1id_l, trigdata_l };
assign { bcid_d, sigp_d, sig0_d, sign_d, l1id_d, trigdata_d } = drdout;

derand #(NBDRDIN,`NLDRDB,`NLDRDB_CNT) u0(
        .sysclk(sysclk), .sysreset(sysreset), .write(l1a), .read(pscload),
        .drdin(drdin), .drrdy(drrdy), .drovf(drovf), .drdout(drdout)
);

endmodule


//----------------------------
```

```
//  derand.v
//
//  Derandomizer
//-----------------------------

module derand(
        sysclk, sysreset, write, read, drdin,
        drrdy, drovf, drdout
);

parameter WIDTH = 8;
parameter DEPTH = 5;
parameter NBCNT = 3; // # bit to count 0 to DEPTH

input  sysclk, sysreset, write, read;
input  [WIDTH-1:0] drdin;
output drrdy, drovf;
output [WIDTH-1:0] drdout;

wire   [DEPTH-1:0] drrpnt, drwpnt;

derbuf1 #(WIDTH, DEPTH) u0(
        .sysclk(sysclk), .drdin(drdin), .drdout(drdout),
        .drwpnt(drwpnt), .drrpnt(drrpnt)
);

derseq1 #(DEPTH, NBCNT) u1(
        .sysreset(sysreset), .sysclk(sysclk), .write(write), .read(read),
        .drrdy(drrdy), .drovf(drovf), .drrpnt(drrpnt), .drwpnt(drwpnt)
);

endmodule

//---------
// derbuf1
//---------
module derbuf1( sysclk, drdin, drdout, drwpnt, drrpnt );

parameter WIDTH = 16;
parameter DEPTH = 5;

input  sysclk;
input  [WIDTH-1:0] drdin;
input  [DEPTH-1:0] drrpnt, drwpnt;
```

```
output [WIDTH-1:0] drdout;

integer j,k;

reg [WIDTH*DEPTH-1:0] buffer;   // buf[d][w] -> buf[d*WIDTH+w]

// read
function [WIDTH-1:0] derand_out;
   input [WIDTH*DEPTH-1:0] buffer;
   input [DEPTH-1:0] drrpnt;
   reg   [DEPTH-1:0] tmp;
   integer j, k;
   begin
       for( j=0; j<WIDTH; j=j+1 ) begin
           for( k=0; k<DEPTH; k=k+1 ) tmp[k] = buffer[k*WIDTH+j] & drrpnt[k];
           derand_out[j] = | tmp;
       end
   end
endfunction

assign drdout = derand_out( buffer, drrpnt );

// write
always @(posedge sysclk) begin
    for( k=0; k<DEPTH; k=k+1 ) begin
        if(drwpnt[k]) for( j=0; j<WIDTH; j=j+1 ) buffer[k*WIDTH+j] <= drdin[j];
    end
end

endmodule

//---------
// derseq1
//---------
module derseq1 (
        sysreset, sysclk, write, read,
        drrdy, drovf, drrpnt, drwpnt
);

parameter DEPTH = 5;    // # length of the derandomizer buffer
parameter NBCNT = 3;    // # bit to count 0 to DEPTH

input  sysreset, sysclk;
input  write, read;
```

```
output drrdy, drovf;
output [DEPTH-1:0] drrpnt, drwpnt;

reg [NBCNT:0] n_data;
reg [DEPTH-1:0] drrpnt;
reg drovf;

reg [DEPTH-1:0] writepoint;
wire full = ( n_data == DEPTH );

assign drrdy = (n_data != 0);
assign drwpnt = writepoint & { DEPTH{ write & !full } };

always @(posedge sysclk or posedge sysreset) begin
    if( sysreset ) begin
        drrpnt <= { {(DEPTH-1){1'b0}}, 1'b1 };
        writepoint <= { {(DEPTH-1){1'b0}}, 1'b1 };
        n_data <= 0;
    end
    else begin
        if((write && !read && !full) || (read && write && !drrdy)) begin
            writepoint <= { writepoint[DEPTH-2:0], writepoint[DEPTH-1] };
            n_data <= n_data + 1;
        end
        else if( !write && read && drrdy ) begin
            drrpnt <= { drrpnt[DEPTH-2:0], drrpnt[DEPTH-1] };
            n_data <= n_data - 1;
        end
        else if( write && read ) begin
            writepoint <= { writepoint[DEPTH-2:0], writepoint[DEPTH-1] };
            drrpnt <= { drrpnt[DEPTH-2:0], drrpnt[DEPTH-1] };
        end
    end
end

always @(posedge sysclk or posedge sysreset) begin
    if( sysreset ) drovf <= 1'b0;
    else begin
        if( write && !read && n_data == DEPTH ) drovf <= 1'b1;
        else drovf <= 1'b0;
    end
end

endmodule
```

```
//================================================================
// ----------------------------
//  psc2d.v
//
//  Parallel Serial Converter
// ----------------------------

`include "readout_h.v"

module psc2d(
        sysclk, sysreset, pscload,
        address, bcid_d, l1id_d, sigp_d, sig0_d, sign_d,
        trigdata_d, novf,
        pscsyn, pscdat0, pscdat1
);

parameter NDUMMY = `NSIG2W-`NBTRGL1-`NBOVF;

input sysclk, sysreset, pscload;
input [`NBADDR-1:0] address;
input [`NBBCID-1:0] bcid_d;
input [`NBL1ID-1:0] l1id_d;
input [`NSIG2W-1:0] sigp_d, sig0_d, sign_d;
input [`NBTRGL1-1:0] trigdata_d;
input [`NBOVF-1:0]  novf;
output pscsyn, pscdat0, pscdat1;

wire [`NBHEAD-1:0] header;
wire [`NBTAIL-1:0] trailer;

reg [`NBPSCIN+1:0] datbuf0, datbuf1, synbuf;

assign header = { `NBHEAD{1'b1} };
assign trailer = { `NBTAIL{1'b1} };
assign pscsyn = synbuf[`NBPSCIN+1];
assign pscdat0 = datbuf0[`NBPSCIN+1];
assign pscdat1 = datbuf1[`NBPSCIN+1];

always @(posedge sysclk or posedge sysreset) begin
    if(sysreset) begin
        synbuf  <= {(`NBPSCIN+2){1'b1}};
        datbuf0 <= {(`NBPSCIN+2){1'b1}};
        datbuf1 <= {(`NBPSCIN+2){1'b1}};
```

```
    end
    else begin
        if(pscload) begin
            synbuf <= { 1'b0, {`NBPSCIN{1'b1}}, 1'b0 };
            datbuf0 <= { 1'b0, header, address, bcid_d, l1id_d, sig0_d,
                         {NDUMMY{1'b0}}, novf, trigdata_d, trailer, 1'b1 };
            datbuf1 <= { 1'b0, header, address, bcid_d, l1id_d, sigp_d,
                         sign_d, trailer, 1'b1 };
        end
        else begin
            synbuf  <= {  synbuf[`NBPSCIN:0], 1'b1 };
            datbuf0 <= { datbuf0[`NBPSCIN:0], 1'b1 };
            datbuf1 <= { datbuf1[`NBPSCIN:0], 1'b1 };
        end
    end
end

endmodule
```

Appendix D

Xilinx 社製 FPGA

D.1 Xilinx 社製 FPGA の概要

FPGA（Field Programmable Gate Array：現場書き込み可能なゲートアレー）は、SRAM 上にロジックデータを書き込むことによって何度でもプログラムが可能なデバイスである。今回、TGC エレクトロニクスの読み出し系のプロトタイプ用に製作した pt3 モジュールには FPGA が 4 つ搭載されている。また、Star Switch は FPGA を用いて開発される予定である。このように、FPGA は TGC のエレクトロニクスの開発において不可欠なデバイスである。ここでは、特に Xilinx 社製の FPGA について触れることとする。

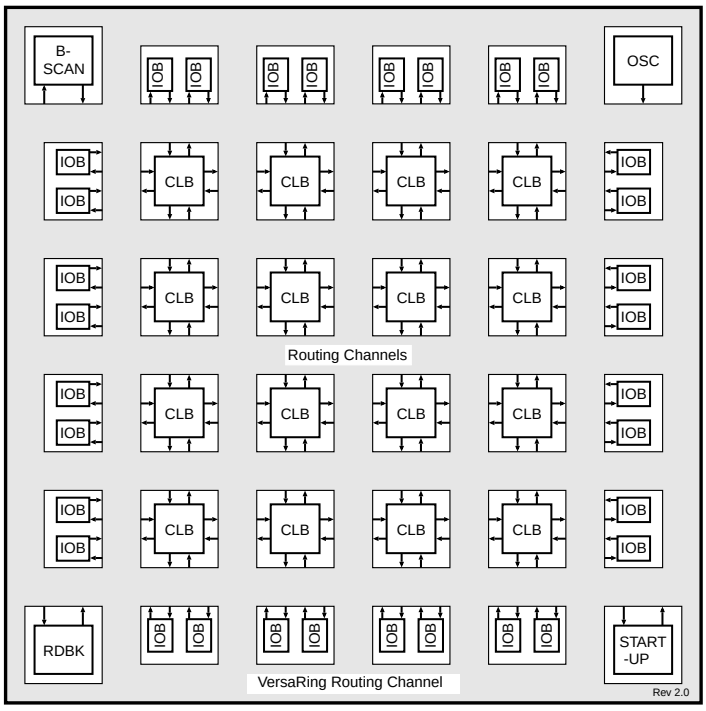

図 D.1： FPGA の構造概念図。CLB がロジックを担っている部分であり、その間を内部配線で接続することにより、目的のロジックを実現する。

Xilinx 社製の FPGA の構造を図 D.1 に示す。FPGA は以下の要素からなっている。

CLB (Configurable Logic Block) CLB は FPGA で実現されるロジックの殆どを担っている部分であり、その内部の構造を単純化した図を図 D.2 に示す。CLB の中には LUT（Look-up Table）と 2 つのフリップフロップが含まれている。これらの LUT は通常は自由な組み合わせ回路を実現するためのファンクションジェネレータとして用いられるが、FPGA の種類によっては、Selected RAM として用いることができる。この場合、CLB は 32 ビットのシングルポート RAM 又は 16 ビットのデュアルポート RAM として用いることが出来る。

IOB (Input/Output Block) IOB は、FPGA の内部のロジックと入出力ピンの間にある部分である。IOB には、出力ドライバ、入力バッファ、プルアップ抵抗、フリップフロップ等が含まれている。入力、出力、双方向ピンとしてコンフィギュレーションが可能である。

内部接続 CLB や IOB 間の信号伝達のための配線は PSM (Programmable Switch Matrix) と呼ばれるところで交差しており、ここで配線間の接続を切り替えている。PSM はトランジスタを含む回路であるため、これを通過する度に伝搬遅延が生じる。

配線には、シングル・レングス・ライン、ダブル・レングス・ライン、ロング・ライン等の種類がある。シングル・レングス・ラインは各 CLB に接続されている短い配線である。ダブル・レングス・ラインは、CLB を一つ飛び越して 2 つ隣の CLB と結ばれている配線である。ロング・ラインは FPGA の端から端まで通じている配線のことで、PSM をあまり通らずに遠くの CLB 間を接続できる。ロング・ラインはまた 3 ステートのバスラインを実現する。

D.2 Xilinx FPGA の種類

Xilinx 社からは目的、用途に応じて様々な種類の FPGA が市販されている。ここでは TGC のエレクトロニクスの開発に関連のある製品を挙げる。

Spartan シリーズ 5 V で動作する FPGA。動作速度の点で他の種類に劣るが、安価である。3.3 V で動作する Spartan-XL シリーズも存在する。

XC4000E シリーズ 5 V で動作するもののなかでは最速の FPGA。

XC4000XLA シリーズ 3.3 V で動作する FPGA で、最大 180 k のシステムゲートのものまで存在する。

XC4000XV シリーズ XLA シリーズをさらに大容量化をすすめたシリーズで、2.5 V で動作する。

Virtex シリーズ 最大 3.2 M ゲートの高性能 FPGA。2.5 V で動作するが、他に 1.8 V で動作する Virtex-E シリーズもある。一部、LVDS などの差動型信号に対応したものも登場し始めている。

これらのシリーズのうち、規模（内蔵される CLB 数）が最大のものと最小のものについてのデータを表 D.1 に示した [19, 20, 21, 22]。但し、表中の Virtex シリーズの CLB は他のシリーズの CLB と大きさが異なり、これらを比較しても無意味である。また、表中の Max RAM Bits は全ての CLB を RAM として利用した場合の値である。

表 D.2 に各シリーズの性能を示す。

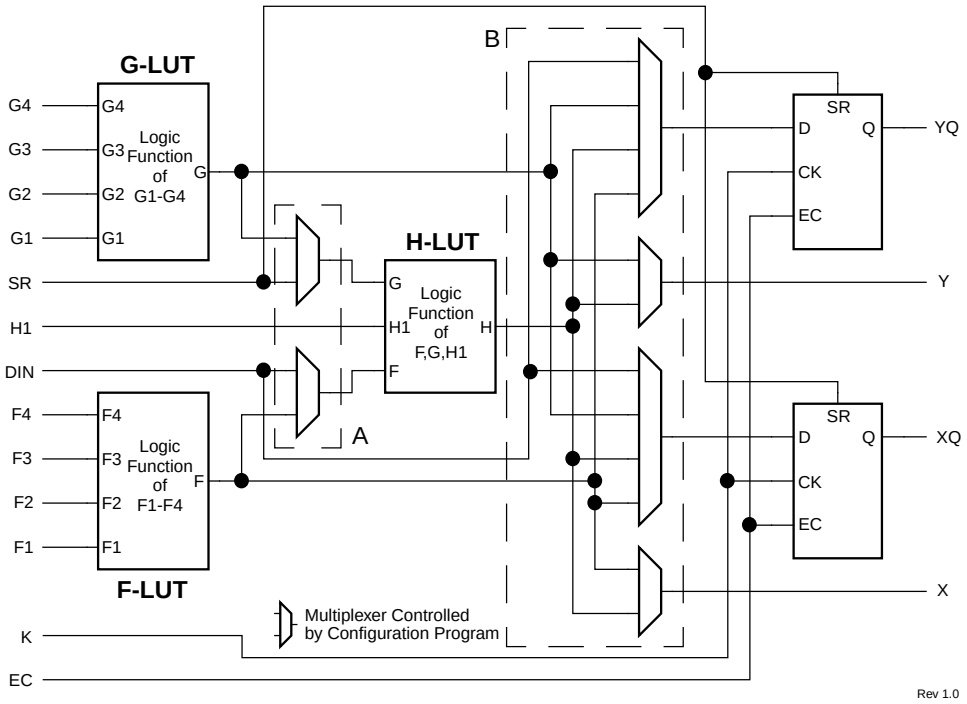

図 D.2： CLB の内部の詳細。基本的には LUT（Look-up Table）と 2 つのフリップフロップから形成される。各所に配置されたマルチプレクサによって、どの LUT を利用するか、フリップフロップを利用するかどうかを選択できるようになっている。

Series	Device	Max Logic Gate [k]	Max RAM Bits	CLBs
Spartan	XCS05	3	3200	100
Spartan	XCS40	20	25088	784
XC4000E	XC4003E	3	3200	100
XC4000E	XC4025E	25	32768	1024
XC4000XLA	XC4013XLA	13	18432	576
XC4000XLA	XC4085XLA	85	100352	3136
XC4000XV	XC40110XV	110	131072	4096
XC4000XV	XC40250XV	250	270848	8464
Virtex	XCV50	58	24576	384
Virtex	XCV1000	1124	393216	6144

表 D.1： 主な Xilinx 社製 FPGA の規模。なお、Virtex シリーズの CLB は他のシリーズのものと大きさが異なるので、直接比較は出来ない。

	Spartan	XC4000XLA	Virtex
System Performace [MHz]	80	100	200
CLB F/F Setup Time [ns]	1.8	0.7	0.6
Output Delay [ns]	3.6	2.6	1.2

表 D.2： 各シリーズの FPGA の主な性能。System Performance は動作速度の限界の目安。CLB F/F Setup Time は CLB のフリップフロップにおいてクロックの到着時刻よりもどのくらい入力信号が早く到達していなければならないかを表す。Output delay は I/O ブロック内での遅延。

D.3 FPGA への書き込み

ここでは pt3 モジュールに利用した Spartan シリーズの FPGA にロジックを書き込む方法を記す。

従来の Xilinx の FPGA への書き込みには、マスターモード、スレーブ・シリアルモード、ペリフェラルモードのモードがあり、同期・非同期などを区別すると計 6 種類のモードがあったが、Spartan シリーズではこのうちマスター・シリアルモードとスレーブ・シリアルモードだけが利用できる。pt3 モジュールではスレーブ・シリアルモードを利用して、コンフィギュレーションデータの書き込みを行っている。

FPGA の機能ピン

ここで、書き込みの方法を示すため Spartan シリーズにおいて、コンフィギュレーション等のために割り当てられた主な機能ピンの一覧を示す。但し、以下の説明は、スレーブ・シリアルモードを用いた場合にのみ有効である。なお、機能ピンの殆どはコンフィギュレーションに関連したものであるが、それ以外のものも存在する。都合上ここで同時に示すことにする。

VCC, GND 5V 電源ピンとグランドピン。

CCLK 入力ピン。コンフィギュレーション用のクロック。

DONE 出力ピン。コンフィギュレーション終了後にレベルを high にすることが可能で、その場合はコンフィギュレーションの終了を示す。

$\overline{\text{PROGRAM}}$ 入力ピン。$\overline{\text{PROGRAM}}$ を low にすると、コンフィギュレーションメモリがクリアされる。

MODE 入力ピン。マスター・シリアル及びスレーブ・シリアルの切り替えに用いる。high（または float）の時スレーブ・シリアルモードになる。

HDC 出力ピン。HDC はコンフィギュレーション中は high を出力する。コンフィギュレーション終了後はユーザー入出力ピンとなる。従って、HDC が low になるようなデータを書き込むと、このピンはコンフィギュレーション中は high、終了後は low になるので、このピンを見ることによって、コンフィギュレーション完了を知ることが出来る。

$\overline{\text{LDC}}$ 論理が反転していることを除いて HDC と同じ。

$\overline{\text{INIT}}$ コンフィギュレーション制御用の双方向ピン。low に下げるとコンフィギュレーションが WAIT 状態になる。また、コンフィギュレーション中にエラーが起こると low を出力する。

DIN 入力ピン。コンフィギュレーションデータのシリアル入力。コンフィギュレーション時には、CCLK の上りエッジで DIN のデータが取り込まれる。

PCLK1-4, SCLK1-4 クロック入力。コンフィギュレーションには無関係。

TCK,TDI,TMS,TDO バウンダリスキャン用のピン。これらを用いて、コンフィギュレーションを行うことも可能。

これらの機能ピンの多くは、コンフィギュレーション後には通常の I/O ピンとして利用可能である。ただし、pt3 モジュールの場合には、I/O ピンの数は足りているので、機能ピンを I/O ピンとしては使っていない。

FPGA に書き込むデータ

FPGA に書き込むデータは、データ長を表す 24 ビットの数字を含むヘッダ、実際のコンフィギュレーションデータ、終了コードからなる。シリアルモードで書き込んだ場合には、ヘッダは 40 ビット、終了コードは 24 ビットである。コンフィギューレーションデータはいくつかのフレームに分けられていて、1 フレーム分のデータを送るごとに 4 ビットのエラーチェックビットを送る。エラーチェックビットは、CRC (Cyclic Redundancy Check）法により計算されたものか、又は定数（0110）である。Spartan XCS40 の場合には、1076 フレームに分けられており、1 フレームあたり 306 ビットのデータがある。ヘッダも合わせた全データは 329312 ビットである。

書き込みの手順

FPGA への書き込みは、コンフィギュレーションメモリクリア、初期化、コンフィギュレーションの三段階からなる。書き込みの手順を図 D.3 に示す。

電源投入後、又は $\overline{\text{PROGRAM}}$ ピンが low の間、FPGA はコンフィギュレーションメモリを消去する。その後 $\overline{\text{PROGRAM}}$ ピンが high であれば、もう一度コンフィギュレーションメモリを消去した後、初期化を行う。

初期化状態になると、$\overline{\text{LDC}}$, DONE, $\overline{\text{INIT}}$ は low に、HDC は high になる。初期化が終了すると $\overline{\text{INIT}}$ は解放される（通常は外部プルアップされているので high になる）。FPGA は $\overline{\text{INIT}}$ が high であることを確かめた後、MODE ピンから書き込みのモードを決定し、コンフィギュレーションに移る。

スレーブシリアルモードの場合は、FPGA は DIN にシリアルに乗せられたコンフィギュレーションデータを CCLK の立ち上がりエッジで取り込む。コンフィギュレーションデータを全て送るとコンフィギュレーションは終了である。書き込みが成功した場合は、FPGA はスタートアップ状態を経た後、書き込んだロジック通りに動作する。一方、書き込み中にエラーが検出されると、$\overline{\text{INIT}}$ が low になる。このときは、$\overline{\text{PROGRAM}}$ を一旦 low に下げて書き込みをやり直せばよい。

書き込みが完了したかどうかを知るためには DONE ピンを用いる。DONE ピンは書き込み中は low、書き込み完了後は high になる。ただし、書き込み完了後 float にするようにコンパイルツールのオプションを設定することも可能で、このように設定すると DONE ピンを見ても書き込み終了を知ることはできない。また、書き込みの完了は HDC ピンによって知ることも可能である。HDC は書き込み中は high であるので、HDC が low であるようなロジックを FPGA に書き込むと、書き込みが成功すれば HDC は low になる。

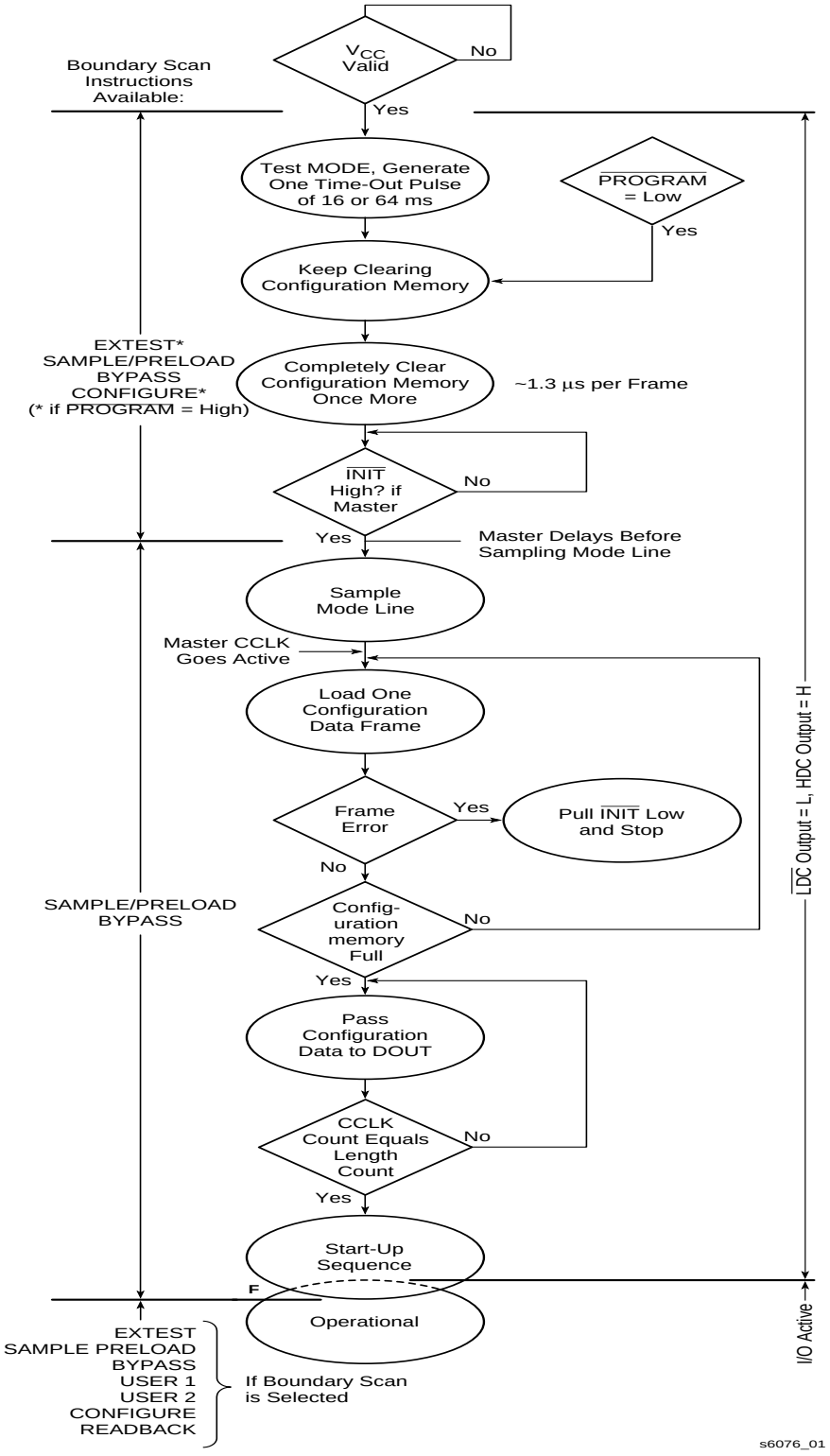

図 D.3： FPGA への書き込みの手順

Appendix E

pt3 モジュール

E.1 pt3 モジュールの概要

pt3 モジュールは ATLAS TGC の読み出し系のためのプロトタイプ用モジュールである。pt3 モジュールについては第 5 章で簡単に説明したが、ここではその仕様についてもう少し詳しく説明する。

pt3 モジュールは Xilinx 社の Spartan シリーズの FPGA（Appendix D 参照）を 4 つ搭載しており、VME 経由でここに必要なロジックを書き込んで使用する。ここに書き込むロジックを変えることにより、このモジュールは種々の用途に使うことが可能である。その意味では、単なるプロトタイプモジュールではなく、汎用モジュールと呼んでも差し支えない。

本文中で pt3 モジュールの写真（ 76 頁の図 5.1 ）と機能ブロック図を（ 77 頁の図 5.2 ）を示した。pt3 モジュールの回路図を図 E.1 と図 E.2 に示す。pt3 モジュールは 6U の VME モジュールであり、以下の部分からなる。

- FPGA（XCS40PQ208）
- VME のプロトコルを解読するための VME 解読部
- クロック関連部分
- 入出力用ピン、LED などその他の部分

pt3 モジュールに搭載されている FPGA は Spartan シリーズの XCS40PQ208 である。Xilinx 社製の FPGA の仕様や書き込み手順などは Appendix D に記したので、以下ではそれ以外の部分について説明する。

E.2 pt3 モジュールの構成部分

VME 解読部

pt3 モジュールは VME のコネクタとしては P1 コネクタのみを使用している。P2 コネクターは用いていないので、可能なアクセスは A24D16 や A16D16 などに限られる。

pt3 の VME 解読部分は、基本的には Xlinix 社製の CPLD XC95216HQ208 が担っている。但し、全ての VME の信号線をこの CPLD につなぐと CPLD の I/O ピンが不足する。また、CPLD ではそれほど大規模な論理回路を実現できるわけではないので、変更が必要のなく CPLD のリソースを大きく消費する

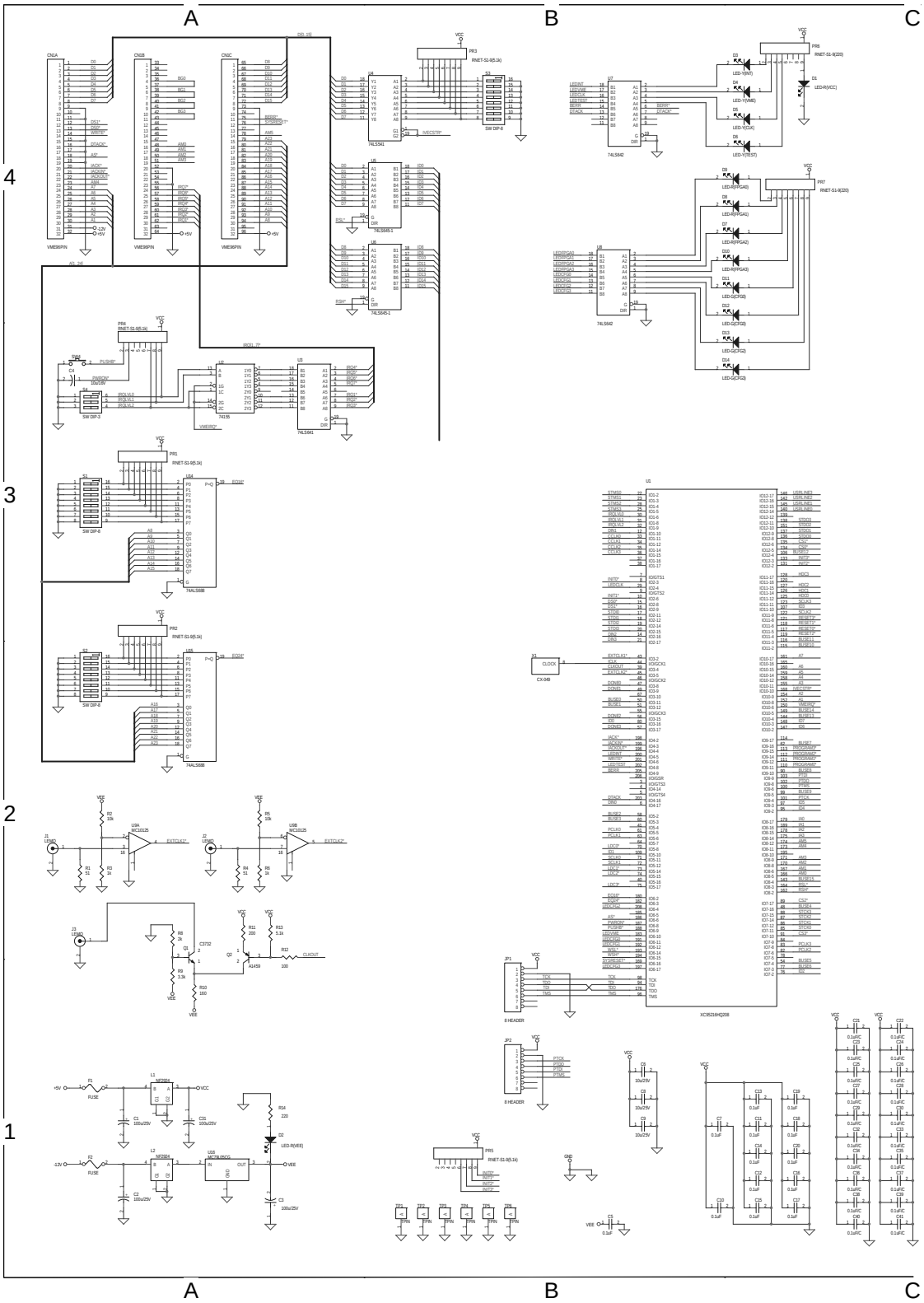

図 E.1： pt3 モジュールの回路図 (1)

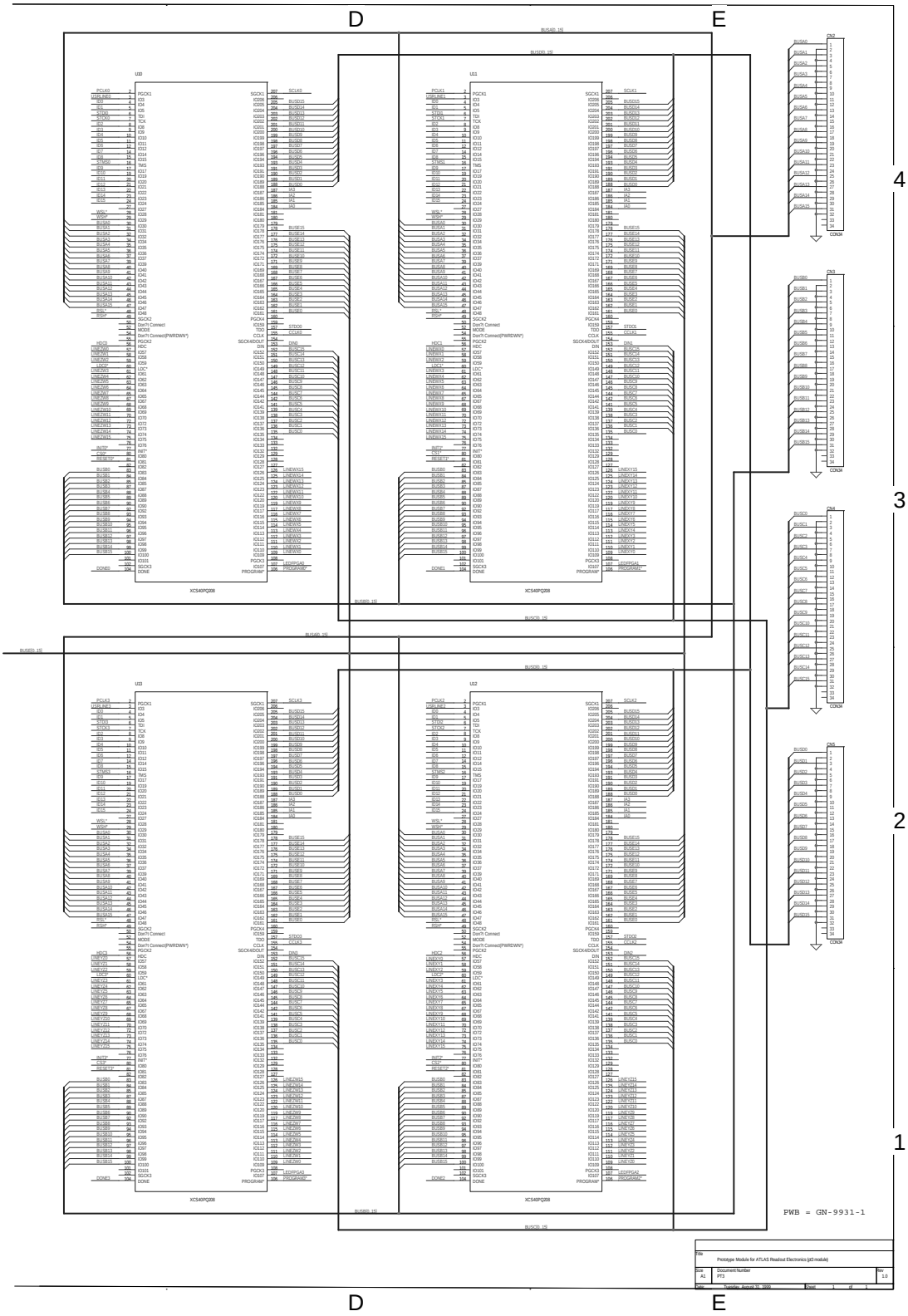

図 E.2： pt3 モジュールの回路図 (2)

部分は、出来る限り CPLD の外部で作り出した方がよい。そのために、補助的に 74 シリーズのデバイスを用いている。

pt3 モジュールが利用している VME の信号線は、以下の通りである。

A23-A08 アドレス線。SN74LS688 によりディップスイッチと比較している。

A07-A01 アドレス線。CPLD の I/O ピンに接続。

AM5-AM0 アドレス・モディファイア線。CPLD の I/O ピンに接続。CPLD でフルデコード。

D15-D00 データ線。74LS645-1 でバッファされる（ID15-ID00 と呼ぶ）。74LS645-1 のデータ転送方向は CPLD が制御。ID07-ID00 は CPLD の I/O ピンに接続。また、D07-D00 に関しては、ディップスイッチで設定した割り込みベクトルをながすことも可能。

AS* アドレスストローブ。CPLD の I/O ピンに接続。

DS0*, DS1* データストローブ。CPLD の I/O ピンに接続。

IACK* CPLD の I/O ピンに接続。

IACKIN* CPLD の I/O ピンに接続。

IACKOUT* CPLD の I/O ピンに接続。

IRQ7*-IRQ1* CPLD が VMEIRQ* ピンを low にすると、割り込みレベルディップスイッチで設定したレベルに相当する IRQ* 線が low になり、割り込みを要求する。

WRITE* CPLD の I/O ピンに接続。

SYSRESET* CPLD の I/O ピンに接続。

DTACK* オープンコレクタでなければならないので、74LS642 を介して CPLD の I/O ピンに接続。

BERR* オープンコレクタでなければならないので、74LS642 を介して CPLD の I/O ピンに接続。

pt3 モジュールでは A24-A08 の値はディップスイッチで設定する。それゆえ 256 バイトのアドレス空間を消費する。また、割り込みレベルもディップスイッチで固定される。これらの点を除けば、CPLD の内部を書き換えることによって VME の解読方法を自由に決定することが出来る。

TGC の読み出し系のプロトタイプとして使う場合に必要な機能としては、FPGA へのコンフィギュレーションデータのダウンロード、FPGA に対して VME から読み込みと書き込み、内部クロックと外部クロックの切り替え、などが挙げられる。このための CPLD のロジックについては E.3 でのべる。pt3 モジュールを利用する場合、通常はこのロジックで十分であるが、少し複雑なことをしようとすれば目的に応じて CPLD の中を書き換える必要がある。

モジュール内部の信号線

pt3 モジュール内の信号線には、前節で述べた VME 関連の信号線の他に以下のものが存在する。なお、以下では FPGA0, FPGA1 などは、それぞれ 0 番目の FPGA、1 番目の FPGA を意味する。

BUSA,BUSB,BUSC,BUSD 4 つの FPGA と外部ピンをつなぐ各々 16 ビットのバスライン。モジュールへの入出力用の信号線である。

BUSE 4 つの FPGA と CPLD をつなぐ 16 ビットのバスライン。

PCLK0 CPLD から FPGA0 にクロックを供給する信号線。PCLK1 なども同様。

RSL*,RSH* VME の読み込みのストローブを意味する。

WSL*,WSH* VME の書き込みのストローブを意味する。

CS0* CPLD と FPGA0 を結ぶ信号線。VME のアクセスで FPGA0 が選ばれている時、この信号線は CPLD によって low にされる。CS1*, CS2*, CS3* も同様に FPGA1, FPGA2, FPGA3 に対するアクセスを示す。

RESET0* CPLD と FPGA0 を結ぶ信号線で FPGA0 のリセットに用いる。その際には FPGA 側に RESET0* がリセットを意味するようなロジックを実装しなければならない。RESET1* なども同様。

クロック関連部

pt3 モジュールには内部クロックとして 40 MHz のクロックが装着されている。他の周波数のクロックと取り替えることも可能である。その他に、外部クロック用の NIM 端子が 2 つ用意されている。これらのクロックは CPLD に供給されており、CPLD で FPGA のクロックを切り替えが可能なようになっている。4 つの FPGA のクロックはそれぞれ独立である。

また、クロック出力用の NIM 端子も 1 つ用意されている[1]。この信号線も CPLD の出力ピンとつながっていて、内部クロックなどを出力することが出来る。

その他の部品

pt3 モジュールのその他の構成要素について説明する。

電源関連 pt3 モジュールのほとんどの素子は VME から供給される 5 V の電源で動作している。NIM と TTL の変換部分に −5 V の電源が必要な部分があるが、これは VME の −12 V の電源をレギュレータで −5 V にまで落として利用している。

コネクタ pt3 モジュールのフロントパネル側には 34 ピンヘッダが二段重ねになったものが二つ取り付けられている。これらは 4 つの FPGA と接続されており、64 本の入出力線となっている。一方、バックプレーン側には VME の P1 コネクタが用いられている。このほか、CPLD のコンフィギュレーション用の JTAG ピンなどが、基板上に装着されている。

LED pt3 モジュールのフロントパネルには LED が 14 個ついている。内訳は、電源表示用 2 つ（5 V, −5 V 用）、CPLD に接続されているもの 8 つ（うち 4 つは各 FPGA のダウンロード完了を示すためのもの）、各々の FPGA に接続されているものが 4 つである。

E.3 VME 解読用 CPLD のロジック

ここでは pt3 を TGC の読み出し系のプロトタイプとして用いるために作成した CPLD のロジックについて述べる。なお、このロジックの Verilog のソースについては最後に挙げる。このロジックを用いると以下のことができるようになる。

[1] TTL を NIM に変換する回路の抵抗値に問題があり、出力レベルが不足している。

- コンフィギュレーションデータの FPGA へのダウンロード
- FPGA への VME からの書き込みと読み出し
- クロックの切り替え

pt3 モジュールは既に述べたように、VME の A24D16 でのアクセスをサポートしている。このうち A23-A08 の値はディップスイッチにより設定しているので、A07 以下の 256 バイトのアドレスを占有している。

pt3 モジュールに対するアクセスのうち、A07 が 1 であるようなアクセスは、FPGA に対するアクセスと見なされる。この際、A06, A05 の値によってどの FPGA に対するアクセスか区別される。すなわち、A07 = 1, A06 = 0, A05 = 0 の時は 0 番目の FPGA へのアクセスであり、A07 = 1, A06 = 0, A05 = 1 の時は 1 番目の FPGA へのアクセスである。0 番目のアクセスが行われた場合、CS0* が low に下げられる。A04-A01 の値は CPLD から供給されているので、その値に応じた振舞をするように FPGA 側で設計すればよい。

一方、A07 = 0 であるアクセスは、CPLD に対するアクセスとして扱っている。このときは、CPLD がアドレスに応じた処理を行う。表 E.1 に、アドレス（A07-A01）の機能の一覧を示す。

最後に、CPLD 内に書き込んだロジックの Verilog ソースを載せておく。

```
//------------
// pt3_h.v
//------------
‘define NCHIPS          4

// address  A[6:1]
‘define ADR_RESET       6’b000000
‘define ADR_PROGRAM     6’b000001
‘define ADR_INIT        6’b000010
‘define ADR_DONE_HDC    6’b000011
‘define ADR_IRQLVL      6’b000100
‘define ADR_LED         6’b000101
‘define ADR_VMECLK      6’b000111
‘define ADR_JTAGSRC     6’b001000
‘define ADR_VMEJTAG     6’b001001
‘define ADR_CLKOUT      6’b001100

// address group A[6:3]
‘define ADRG_CONFIGURE  4’b1000
‘define ADRG_PCLK       4’b1010
‘define ADRG_SCLK       4’b1011

// clock source
‘define CLK_WIDTH       3
‘define CLK_INT         3’b000
‘define CLK_INT2        3’b001
‘define CLK_INT4        3’b010
```

A07-A01	方向	種類	働き
0000000	RW	A	RESET3*-RESET0* 値の設定、読み出し。
0000001	RW	A	PROGRAM3*-PROGRAM0* 値の設定、読み出し。
0000010	R	A	INIT3*-INIT0* 値の読み出し。
0000011	R	-	DONE3-DONE0, HDC3-HDC0 値の読み出し。 D7-D4 に DONE が、D3-D0 に HDC の値が乗る。
0000100	R	-	D2-D0 に IRQLVL2-IRQLVL0 の値を乗せる。
0000101	RW	-	LEDTEST の点灯、読み出し。D0 のみ利用。
0000110			なし
0000111	RW	A	VMECLK。VME で FPGA のクロックを作るときに利用。
0001000	RW	-	JTAGSRC。JTAG 線のつなぎかた。
0001001	RW	A	VMEJTAG。VME で JTAG を使うときに利用。
0001010			なし
00011xx			なし
01000??	W	B	CONFIGURE（本文参照）
01001xx			なし
01010??	RW	B	PCLK。FPGA の PCLK の切り替え。
01011??	RW	B	SCLK。FPGA の PCLK の切り替え。
011xxxx		B	なし
1??xxxx			A06,A05 の値によって CS0-3 を発生。 FPGA への書き込み、読み出しのため。

表 E.1： pt3 モジュールのアドレス一覧。方向は VME のアクセスの向きで、R は読み込み、W は書き込みである。種類が A と表記されているアドレスについては、D3-D0 がそれぞれ FPGA3-FPGA0 に対応する。例えば、アドレス 0 に 0x03 を書き込むと、RESET3* = 0, RESET2* = 0, RESET1* = 1, RESET0* = 1 となる。一方 B と表記されているアドレスについては、アドレスのうち ?? と記されている部分に FPGA の番号を入れることによって特定の FPGA に対するアクセスが可能である。例えば、アドレス 0100001 は FPGA1 に対する CONFIGURE である。

```
‘define CLK_EXT1        3’b100
‘define CLK_EXT2        3’b101
‘define CLK_VME         3’b110

// clock enable
‘define CLKEN_WIDTH     2
‘define CLKEN_NONE      2’b00
‘define CLKEN_EXT1      2’b01
‘define CLKEN_EXT2      2’b10
‘define CLKEN_BOTH      2’b11

// JTAG source
‘define JTAGSRC_WIDTH   4
```

```
‘define JTAG_NULL       4’b0000
‘define JTAG_VME        4’b0001

//------------
// pt3cpld.v
//------------
‘include "pt3_h.v"

module pt3cpld (
        ICLK, EXTCLK1_, EXTCLK2_, CLKOUT,
        A, AM, EQ16_, EQ24_, AS_, WRITE_, DS0_, DS1_,
        PUSHB_, PWRON_, SYSRESET_, IACK_, IACKIN_, IACKOUT_, IRQLVL,
        DTACK, BERR, WSL_, WSH_, RSL_, RSH_, VMEIRQ_, IVECSTR_, IA, ID,
        INIT_, HDC, LDC_, DONE, PCLK, SCLK, CS_, RESET_, DIN, CCLK, PROGRAM_,
        LEDCFG, LEDINT, LEDVME, LEDCLK, LEDTEST, STCK, STDI, STMS, STDO
);

parameter CNT_WIDTH = 3;
parameter CNT_MAXIMUM = 5;  // 2^3-1

input  ICLK, EXTCLK1_, EXTCLK2_;
input  [7:1] A;
input  [5:0] AM;
input  EQ16_, EQ24_, AS_, WRITE_, DS0_, DS1_, PUSHB_, PWRON_, SYSRESET_;
input  IACK_, IACKIN_;
input  [2:0] IRQLVL;
output CLKOUT, DTACK, BERR, IACKOUT_, WSL_, WSH_, RSL_, RSH_;
output VMEIRQ_, IVECSTR_;
output [4:1] IA;
output [‘NCHIPS-1:0] LEDCFG;
output LEDINT, LEDVME, LEDCLK, LEDTEST;
inout [7:0] ID;

input  [‘NCHIPS-1:0] INIT_, HDC, LDC_, DONE;
output [‘NCHIPS-1:0] PCLK, SCLK;
output [‘NCHIPS-1:0] CS_, RESET_;
output [‘NCHIPS-1:0] DIN, CCLK, PROGRAM_;
output [‘NCHIPS-1:0] STCK, STDI, STMS;
input  [‘NCHIPS-1:0] STDO;

reg PROGRAM_, CCLK, DIN;
reg LEDTEST;

// -------------------------
```

```
// internal signal, register
// -------------------------
integer i, j;
reg [`NCHIPS-1:0] vmeReset_;
reg [2*`NCHIPS-1:0] vmeClk;
reg clk2, clk4;
reg [`JTAGSRC_WIDTH-1:0] jtagSrc;
reg [`CLK_WIDTH*`NCHIPS-1:0] pclkSrc, sclkSrc;
reg [`CLK_WIDTH-1:0] clkoutSrc;
reg [`CLKEN_WIDTH*`NCHIPS-1:0] pclkEn, sclkEn;
reg [CNT_WIDTH-1:0] counter;
reg vmeTck, vmeTdi, vmeTms;
wire vmeTdo, selDt_, selErr_, selIrq_, extReset_;
reg dtackDt;
wire dtackInt;

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

//----------------
// RESET signal
//----------------
assign extReset_ = SYSRESET_ & !PUSHB_ & PWRON_;
assign RESET_ = vmeReset_ & { `NCHIPS{extReset_} };

//---------------
// VME selection
//---------------
assign selDt_ = !( IACK_ & (
        ( ((AM==6'h2D)||(AM==6'h29)) & !EQ16_) ||
        ( ((AM==6'h39)||(AM==6'h3A)||(AM==6'h3D)||(AM==6'h3E)) & !EQ16_ & !EQ24_)
        ));
assign selErr_ = !( selDt_ & IACK_ &
        ((AM!=6'h0F)||(AM!=6'h0E)||(AM!=6'h0D)||(AM!=6'h0B)
                    ||(AM!=6'h0A)||(AM!=6'h09)) & !EQ16_ & !EQ24_ );
assign selIrq_ = !( !IACK_ & !AS_ & (A[3:1]==IRQLVL) );

//---------------
// VME signal
//---------------
assign BERR  = !selErr_ && (!DS0_ || !DS1_ );
//assign BERR = 1'b0;
assign IACKOUT_ = IACKIN_;
assign IA = A[4:1];
assign WSL_ = !( !selDt_ & !DS0_ & !WRITE_ );
```

```
assign WSH_ = !( !selDt_ & !DS1_ & !WRITE_ );
assign RSL_ = !( !selDt_ & !DS0_ & WRITE_ );
assign RSH_ = !( !selDt_ & !DS1_ & WRITE_ );
assign DTACK = dtackDt | dtackInt;
assign dtackInt = 1'b0;

assign LEDINT = 1'b0;
assign LEDVME = 1'b0;
assign LEDCLK = ICLK;
assign LEDCFG = DONE;  // ~HDC;

function [`NCHIPS-1:0] chipselect_;
  input [7:5] A;
  input selDt_;
  begin
    chipselect_ = { `NCHIPS{1'b1} };
    if( !selDt_ & A[7] ) chipselect_[ A[6:5] ] = 1'b0;
  end
endfunction

assign CS_ = chipselect_( A[7:5], selDt_ );

//------------------
// internal counter
//------------------
always @( posedge ICLK )
begin
    if( DS0_ || DS1_ ) counter <= {CNT_WIDTH{1'b0}};
    else begin
        if( !selDt_ && counter != {CNT_WIDTH{1'b1}} ) counter <= counter + 1;
    end
end

//---------------
// make CLK
//---------------
always @( posedge ICLK or negedge extReset_ ) begin
    if( !extReset_ ) clk2 <= 1'b0;
    else clk2 <= ~clk2;
end

always @( posedge clk2 or negedge extReset_ ) begin
    if( !extReset_ ) clk4 <= 1'b0;
    else clk4 <= ~clk4;
```

```
end

//-------------------
// return DTACK (DT)
//-------------------
always @( posedge ICLK ) begin
    if( DS0_ || DS1_ ) dtackDt <= 1'b0;
    else if( counter==7 ) dtackDt <= 1'b1;
end

//----------------------------------------
// VME write operation ( normal address )
//----------------------------------------
always @( negedge DS0_ or negedge extReset_ ) begin
    if( !extReset_ ) begin
        vmeReset_ <= { `NCHIPS{1'b1} };
        PROGRAM_  <= { `NCHIPS{1'b1} };
        LEDTEST   <= 1'b0;
        vmeClk    <= { 2*`NCHIPS{1'b0} };
        vmeReset_ <= { `NCHIPS{1'b1} };
        jtagSrc   <= `JTAG_NULL;
        clkoutSrc <= `CLK_INT;
        for( i=0; i<`NCHIPS; i=i+1 ) begin
            pclkSrc[(i+1)*`CLK_WIDTH-1:i*`CLK_WIDTH] = `CLK_INT;
            pclkEn[(i+1)*`CLKEN_WIDTH-1:i*`CLKEN_WIDTH] = `CLKEN_NONE;
        end
    end
    else begin
        if( !selDt_ && !WRITE_ && A[7]==1'b0 ) begin
            casex ( A[6:1] )
                `ADR_RESET:     vmeReset_ <= ID[`NCHIPS-1:0];
                `ADR_PROGRAM:   PROGRAM_  <= ID[`NCHIPS-1:0];
                `ADR_LED:       LEDTEST   <= ID[0];
                `ADR_VMECLK:    vmeClk    <= ID[2*`NCHIPS-1:0];
                `ADR_JTAGSRC:   jtagSrc   <= ID[3:0];
                `ADR_CLKOUT:    clkoutSrc <= ID[`CLK_WIDTH-1:0];
                { `ADRG_PCLK, 2'bxx }:
                    for( i=0; i<`NCHIPS; i=i+1 ) if( A[2:1] == i ) begin
                        pclkSrc[(i+1)*`CLK_WIDTH-1:i*`CLK_WIDTH]
                           = ID[`CLK_WIDTH-1:0];
                        pclkEn[(i+1)*`CLKEN_WIDTH-1:i*`CLKEN_WIDTH]
                           = ID[`CLK_WIDTH-1+4:4];
                    end
            endcase;
```

```
        end
    end
end

always @(posedge ICLK or negedge extReset_) begin
    if( !extReset_ ) begin
        vmeTck <= 1'b0;
        vmeTdi <= 1'b0;
        vmeTms <= 1'b0;
    end
    else begin
        if( DS0_ ) vmeTck <= 1'b0;
        else if( !WRITE_ && A=={ 1'b0, `ADR_VMEJTAG } ) begin
            if( counter == 1 ) begin
                vmeTms <= ID[1];
                vmeTdi <= ID[0];
            end
            else if( counter == 3 ) vmeTck <= 1'b1;
        end
    end
end

//---------------------------------------
// VME write operation ( address group )
//---------------------------------------
always @(posedge ICLK) begin
    if( DS0_ ) CCLK <= { `NCHIPS{1'b0} };
    else if( !selDt_ && !WRITE_ && A[7]==1'b0 ) begin
        if( A[6:3] == `ADRG_CONFIGURE ) begin
            if( counter == 1 ) DIN[A[2:1]] <= ID[0];
            if( counter == 3 ) CCLK[A[2:1]] <= 1'b1;
        end
    end
end

//--------------------
// VME read operation
//--------------------
function [7:0] read;
  input selDt_, WRITE_, DS0_;
  input [7:1] A;
  input [`NCHIPS-1:0] vmeReset_, PROGRAM_, INIT_, DONE, HDC;
  input [2:0] IRQLVL;
  input LEDTEST;
```

```
  input [2*‘NCHIPS-1:0] vmeClk;
  input [‘JTAGSRC_WIDTH-1:0] jtagSrc;
  input vmeTDO, vmeTCK, vmeTMS, vmeTDI;
  input [‘CLK_WIDTH*‘NCHIPS-1:0] pclkSrc;
  input [‘CLKEN_WIDTH*‘NCHIPS-1:0] pclkEn;
  integer i;
  begin
    read = 8’hzz;
    if( !selDt_ && WRITE_ && !DS0_ && A[7]==1’b0 ) begin
        casex( A[6:1] )
            ‘ADR_RESET:         read = { {(8-‘NCHIPS){1’b1}}, vmeReset_ };
            ‘ADR_PROGRAM:       read = { {(8-‘NCHIPS){1’b1}}, PROGRAM_ };
            ‘ADR_INIT:          read = { {(8-‘NCHIPS){1’b1}}, INIT_ };
            ‘ADR_DONE_HDC:
                begin
                    read = 8’hff;
                    read[‘NCHIPS+3:4] = DONE;
                    read[‘NCHIPS-1:0] = HDC;
                end
            ‘ADR_IRQLVL:        read = { 5’b11111, IRQLVL };
            ‘ADR_LED:           read = { 7’b1111111, LEDTEST };
            ‘ADR_VMECLK:
                begin
                    read = 8’hff;
                    read[2*‘NCHIPS-1:0] = vmeClk;
                end
            ‘ADR_JTAGSRC:       read = { {(8-‘JTAGSRC_WIDTH){1’b1}}, jtagSrc };
            ‘ADR_VMEJTAG:       read = { 4’hf, vmeTDO, vmeTCK, vmeTMS, vmeTDI };
            ‘ADR_CLKOUT:        read = { {(8-‘CLK_WIDTH){1’b1}}, clkoutSrc };
            { ‘ADRG_PCLK, 2’bxx }:
                begin
                    read = 8’hff;
                    for( i=0; i<‘NCHIPS; i=i+1 )
                      if( i == A[2:1] ) begin
                            read[‘CLKEN_WIDTH+3:4]
                               = pclkEn[(i+1)*‘CLKEN_WIDTH-1:i*‘CLKEN_WIDTH];
                            read[‘CLK_WIDTH-1:0]
                               = pclkSrc[(i+1)*‘CLK_WIDTH-1:i*‘CLK_WIDTH];
                      end
                end
        endcase;
    end
  end
endfunction
```

```
assign ID = read( selDt_, WRITE_, DS0_, A, vmeReset_, PROGRAM_, INIT_,
                  DONE, HDC, IRQLVL, LEDTEST, vmeClk, jtagSrc, vmeTdo,
                  vmeTck, vmeTms, vmeTdi, pclkSrc, pclkEn );

//--------------------
// CLK
//--------------------
function [`NCHIPS-1:0] clkselect;
  input [`CLK_WIDTH*`NCHIPS-1:0] clkSrc;
  input [`CLKEN_WIDTH*`NCHIPS-1:0] clkEn;
  input ICLK, CLK2, CLK4, EXTCLK1_, EXTCLK2_;
  input [`NCHIPS-1:0] VMECLK_HALF;
  integer i;
  reg [`CLKEN_WIDTH-1:0] ce;
  begin
    clkselect[i] = 1'b0;
    for( i=0; i<`NCHIPS; i=i+1 ) begin
        ce = clkEn[(i+1)*`CLKEN_WIDTH-1:i*`CLKEN_WIDTH];
        if( (ce==`CLKEN_NONE)
          || ((ce==`CLKEN_EXT1)&&(!EXTCLK1_))
          || ((ce==`CLKEN_EXT2)&&(!EXTCLK2_))
          || ((ce==`CLKEN_BOTH)&&(!EXTCLK1_)&&(!EXTCLK2_)) ) begin
            case ( clkSrc[(i+1)*`CLK_WIDTH-1:i*`CLK_WIDTH] )
            `CLK_INT:   clkselect[i] = ICLK;
            `CLK_INT2:  clkselect[i] = CLK2;
            `CLK_INT4:  clkselect[i] = CLK4;
            `CLK_EXT1:  clkselect[i] = !EXTCLK1_;
            `CLK_EXT2:  clkselect[i] = !EXTCLK2_;
            `CLK_VME:   clkselect[i] = VMECLK_HALF[i];
            default:    clkselect[i] = 1'b0;
            endcase;
        end
    end
  end
endfunction

assign PCLK = clkselect( pclkSrc, pclkEn, ICLK, clk2, clk4, EXTCLK1_, EXTCLK2_,
                vmeClk[`NCHIPS-1:0] );
assign SCLK = { `NCHIPS{ ICLK } };
// assign SCLK = clkselect( sclkSrc, sclkEn, ICLK, HALFCLK, EXTCLK1_, EXTCLK2_,
//                 vmeClk[2*`NCHIPS-1:`NCHIPS] );

assign CLKOUT = ( clkoutSrc==`CLK_INT ) ? ICLK :
```

```
                ( clkoutSrc==‘CLK_INT2 ) ? clk2 :
                ( clkoutSrc==‘CLK_INT4 ) ? clk4 :
                ( clkoutSrc==‘CLK_EXT1 ) ? !EXTCLK1_ :
                ( clkoutSrc==‘CLK_EXT2 ) ? !EXTCLK2_ : 1’b0;

//--------------------
// JTAG
//--------------------

assign STCK = ( jtagSrc==‘JTAG_VME ) ? vmeTck : 1’bz;
assign STMS = ( jtagSrc==‘JTAG_VME ) ? vmeTms : 1’bz;
assign STDI = ( jtagSrc==‘JTAG_VME ) ? vmeTdi : 1’bz;
assign vmeTdo = ( jtagSrc==‘JTAG_VME ) ? STDO : 1’b0;
```

Appendix F

Single Event Upset

F.1 積算吸収線量と Single Event Effect

大量の放射線を浴びるような環境のもとでは、半導体は次第にその特性を失って劣化していく。したがって IC が受ける放射線の影響を考える場合、一般的には積算吸収線量（総被曝量）が基準となる。例えば、TGC のエレクトロニクスは表 4.2 に示されているように一年間に 6.2×10^{-1} Gy の線量を吸収する。従って、エレクトロニクスに用いるデバイスには、10 年間分の被曝量に安全計数を乗じた 25 Gy の線量を吸収しても劣化が起こらない、という条件が課される。

一方で、集積度が高い IC では信号として内部を移動する電荷の量は小さい。このような IC にある一定以上のエネルギーをもつ放射線が通過した場合、その時に生じる電荷が回路を流れる電荷量と同程度となり一時的な誤動作が起こる。このように積算吸収線量とは関係なく一度放射線が通過しただけでおこる現象を SEE（Single Event Effect）と呼ぶ。SEE の中で代表的なものが SEU（Single Event Upset）と呼ばれる現象である。これは、IC に入射した粒子によってフリップフロップのビットが反転してしまう現象である。

SEE は宇宙空間のような大量の荷電粒子を浴びる環境で特に問題となるため、特に人工衛星に搭載する IC の開発などの際に研究されてきた。しかし、ATLAS 実験のように高いバックグラウンドが存在し、なおかつ実験室内に電気回路を設置しなければいけない場合にもこれらの効果を考慮に入れた設計を行う必要がある。

F.1.1 Slave Board ASIC の SEU 対策

幸い TGC は ATLAS 検出器の最も外側に位置するため、総被曝量から来る影響は比較的少ない。例えば、TGC エレクトロニクスに対する積算吸収線量の基準は 25 Gy であるが、これは耐放射線用でない通常の ASIC や FPGA が使用可能な程度の被曝量である。一方、大量に到達する中性子の影響で SEU などがおこりうるので、SEU によって受ける影響とその対策について述べる。

中性子には電荷が無いので、SEU を起こすためには一旦物質内のイオンをたたき出す必要がある。そのため、荷電粒子よりも SEU は起こしにくい。中性子による SEU については一部のデバイスについてのみ影響が調べられており、14 MeV の中性子に対する SEU の断面積は $10^{-14} \sim 10^{-13}$ cm^2/bit である [23]。一般に、SEU の影響はデバイスによって大きく依存するので、実際に TGC で用いるデバイスがどの程度 SEU の影響を受けるかは別にテストを行う必要がある。これに対して、TGC エレクトロニクスの中性子の被曝量は、56 頁の表 4.2 にも示した通り最大 1×10^{11} cm^{-2}y^{-1} 程度である。

これらの値から Slave Board に対する SEU の影響を考えてみる。SEU の影響が問題となるのは Slave Board ASIC に設定されているパラメータの類である。データも SEU の影響を受けるが、バックグラウ

ンドや電気ノイズなどの影響に比べれば無視できるからである。これらのパラメータは、後述するように Slave Board ASIC の中に約 200 個ある。Slave Board ASIC 自体は約 3000 個ある。これより 1 日あたりに SEU によって反転するビットの数は、

$$10^{-13}\ [\mathrm{cm}^2/\mathrm{bit}] \times \frac{(1\times 10^{11})\ [\mathrm{cm}^{-2}\mathrm{y}^{-1}]}{365\ [\mathrm{d/y}]}(3000\times 200)[\mathrm{bit}] \cong 20$$

となる。この見積もりは 1 桁以上の誤差を含んでいる上に、デバイスの SEU の断面積を最大限に見積もっている、エネルギーの低い中性子からは SEU の影響を受けないことを考慮していない、という点でかなり過大評価しているが、SEU の効果が無視できないことが分かる。そこで、Slave Board ではパラメータ用のレジスタに SEU 対策を施すことを考えている。

SEU 対策としては、図 F.1 のような Voting Logic が考えられている。この回路は、1 ビットの情報をを記憶するのに 3 つのレジスタを用い、これらの多数決をとる。これらのレジスタには常に出力の値がロードされているので、仮に 3 つのレジスタのうちの 1 つが SEU により反転しても次のクロックで正しい値に書き換えられる。イネーブル付きのフリップフロップをこのような回路に置き換えることによって、SEU によってパラメータが書き変わってしまうことは事実上なくなる。

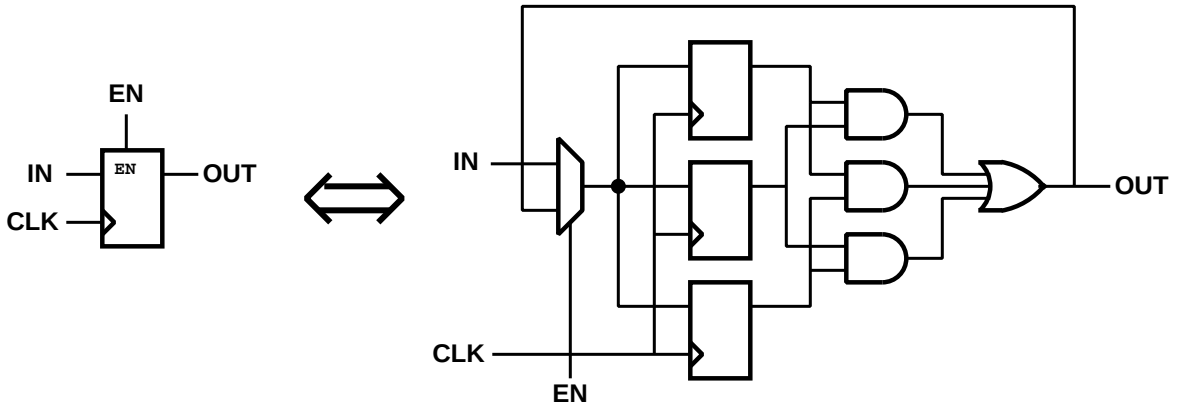

図 F.1： SEU 対策の Voting Logic。左の図のイネーブル付きのフリップフロップを右のような回路に置き換えることによって SEU の影響を受けなくする。右の回路は、1 ビットの情報をを記憶するのに 3 つのレジスタを用い、これらの多数決をとる。これらのレジスタには常に出力の値がロードされているので、仮に 3 つのレジスタのうちの 1 つが SEU により反転しても次のクロックで正しい値に書き換えられる。

ただし、JTAG で設定するパラメータについてはすこし注意が必要である。JTAG の制御は TCK に同期して行われるが、TCK はパラメータ設定時以外は出力されていないかも知れないからである。それゆえ、TCK をクロックとするフリップフロップを用意して一旦ここにデータを保持した後、このフリップフロップの出力を図 F.1 の回路に入力するという形になる。

実際にどのレジスタに SEU 対策を施すかどうかについては検討の余地がある。レベル 1 バッファに施すことが不必要で不可能であることは明らかであるが、パラメータが設定されているレジスタだけで十分であるかと言えばそうではない。例えば、JTAG の TAP コントローラのステート（Appendix B 参照）を保持しているレジスタに SEU が起こってしまうと、ASIC 全体が一瞬にして診断モードに入ってしまうこともありうる。また、L1ID や BCID のカウンタでビットが反転すると、ECR や BCR が来るまでの間はその Slave Board からのデータは不当なものになってしまう。今後は、SEU 対策のための回路を準備する一方、実際に用いる ASIC の耐放射線性についての新しい情報を参考にしながら、SEU 対策を施すレジスタを決定する必要がある。

F.2 FPGA の耐放射線性

FPGA は SRAM 上にコンフィギュレーション用のデータを書き込むことによってロジックを実現している。従って、SEU などによって SRAM 上のコンフィギュレーションデータが書きかわってしまうと正しく動作しなくなってしまう場合がある。これが原因で FPGA は比較的放射線には弱いとされてきた。

しかし、近年、航空産業や宇宙開発などで耐放射線性のあるデバイスの需要が高まるにつれ、FPGA の耐放射線性について研究されるようになってきた。それに伴い、耐放射線性のある FPGA も市販されるようになってきた。例えば、Xilinx 社の XQR シリーズは 600 Gy の線量を吸収しても安定に動作するとされている [24]。

一方で、耐放射線用でない通常のデバイスについても放射線の影響が調べられてきた。文献 [25] によれば、耐放射線用ではない XC4000 シリーズの FPGA の中性子に対する SEU 断面積は 10^{-15} cm^{-2} 程度である。これは、通常のデバイスとしてはかなり小さな値である。このような FPGA を利用すれば、TGC のエレクトロニクスに要求されている耐放射線性を満足するであろうと思われる。

但し、ASIC の場合はパラメータ等の重要なレジスタだけが SEU の影響を受けたが、FPGA の場合はコンフィギュレーションデータ全体、すなわちロジック全体が影響を受ける。従って、コンフィギュレーションデータを読み出してデータが書きかわっていないかを調べ、書きかわっていればデータをロードし直すなどの措置をとって SEU の影響を減らす、などの方策を講じる必要があるかもしれない。しかし、そのためには SEU チェック用のデバイスが新たに必要となってしまう。これが不可能な場合でも、定期的にコンフィギュレーションデータを FPGA にダウンロードするなどによって影響を減らすことができる。

以上を考慮すると、Star Switch を FPGA で開発することは耐放射線性からも可能であると思われる。ただ、耐放射線用の FPGA は価格が高いので、通常の FPGA を用いて開発するのが望ましい。最終的に採用する FPGA が決定すれば、実際に被曝試験を行って十分に動作することを確認する必要がある。

Appendix G

LVDS

LVDS（Low Voltage Differencial Signaling）は差動型のインターフェース規格である [26, 27]。LVDS を定めている規格としては 2 つ存在し、1 つは IEEE（Institute for Electrical and Electronics Engineering ）1596.3、もう 1 つは ANSI/TIA/EIA（American National Standars Institute / Telecommunications Industry Association / Electronic Insustries Association）-644 である。LVDS の特徴としては、高速、低消費電力、ノイズや EMI（Electromagnetic Interface）に強い、比較的安価である、などが挙げられる。

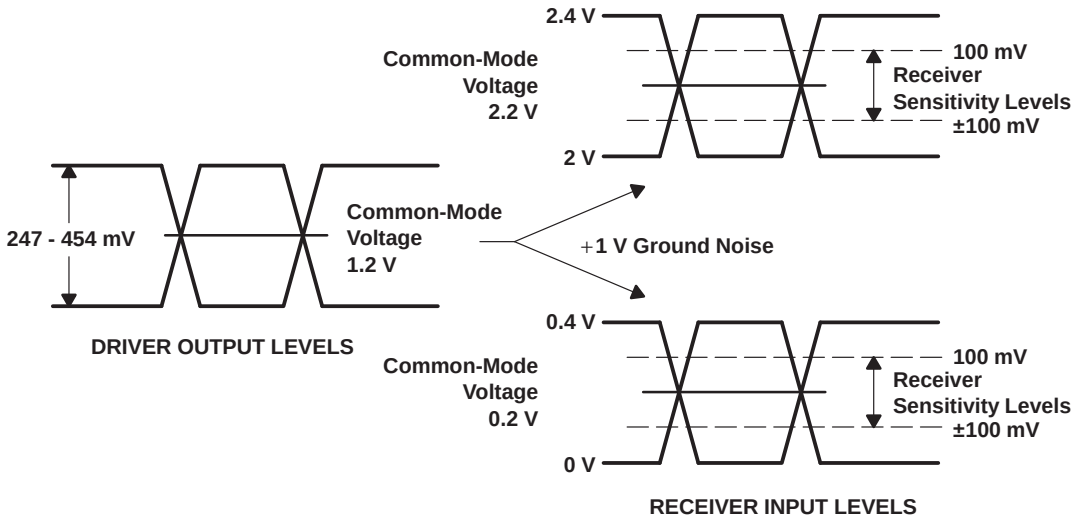

図 G.1： LVDS の電圧レベル。ドライバ側は 1.2 V を中心とした 400 mV （規格では 247 mV 以上 454 mV 以下）の差動電圧を生成する。受信側はグランドレベルが 1 V 変化しても正しく受信できる。

LVDS の電圧レベルを図 G.1 に示す。ドライバ側では 2 本の信号線に 1.2 V を中心として 400 mV（ANSI/TIA/EIA-644 の規格では 247 mV 以上 454 mV 以下と定められている） の差動電圧を与えることによってデータの伝送を行う。レシーバ側は ±1 V のグラウンドのシフトに対応できるようになっている。すなわち、データの送信側と受信側でグラウンドレベルが 1 V 異なっていても問題はない。逆に言えば、レシーバ側は中心電圧が 0.2 V から 2.2 V の間でさえあれば、2 本の信号線の電圧の差から正しくデータを受信出来るのである。このような、差動型のスキームを用いることにより、伝送線にのる信号を大幅に減らすことが出来る。さらに、スイング幅の 400 mV というのは他の規格と比べれば小さいが、これによって、転送速度を向上し、消費電力を抑え、EMI の効果を減らすことが出来る。LVDS を用いると、

500 Mbps の転送速度を実現できるとされている。

LVDS での伝送の構成を図 G.2 に示す。基本的には、送信側と受信側との間を 1 対 1 で接続する。伝送用のケーブルに制限はなく、反射を防ぐために特性インピーダンスに等しい終端抵抗を受信側に接続すればよい。このように、ターミネーションのスキームが単純なことも LVDS の特徴である。LVDS は定電流のドライバを用いており、ドライバの 1 つである DS90C031 の場合には最大電流値は 4.5 mA である。レシーバには伝送線に異常が起こった場合にでも出力を論理 1 に保つ failsafe（安全装置）機構が備わっている。ここでいう異常とは、レシーバの入力に何も接続されていない、レシーバの入力（2 本の伝送線）が短絡している、ドライバが存在しない又は電源が入っていない、などを指す。この様な場合にでも、レシーバ出力は不定値なったり振動したりすることなく、論理 1 を出力する。これにより、利用していないレシーバに伝送線を接続しないまま放置しておくことが可能なだけでなく、外部にバイアス抵抗などを用意しなくてもすむ。この他、1 組のケーブルで双方向の伝送を行ったり（図 G.3 ）、1 つの伝送線上に複数のレシーバを載せることも可能である（図 G.4 ）。

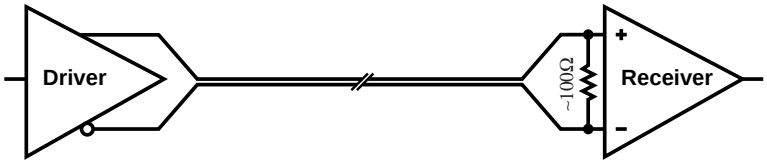

図 G.2： ドライバとレシーバが 1 つずつ存在する場合（point-to-point）の LVDS の接続。受信側には終端抵抗が必要である。

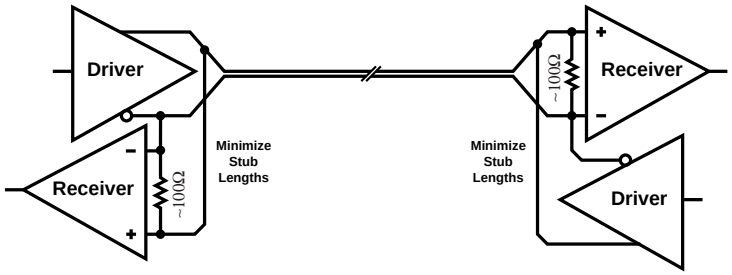

図 G.3： 一対の信号線で双方向の通信を行う際の LVDS の接続

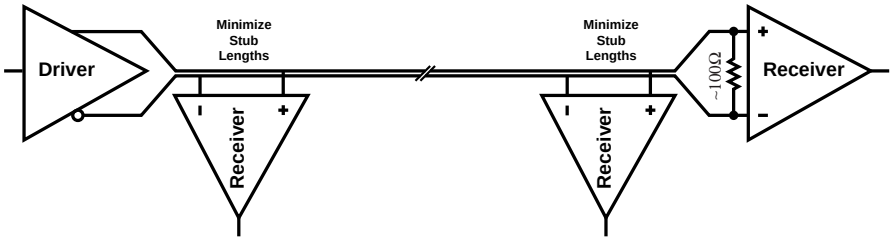

図 G.4： 1 つの伝送線上に複数のレシーバが存在する場合（multidrop）の LVDS の接続

最も一般的な LVDS のドライバとレシーバは National Semiconductor 社の DS90C031 / DS90C032 で

ある。ドライバの DS90C031 は TTL/CMOS の入力レベルを 350 mV の差動電圧に変換する。レシーバの DS90C032 は LVDS レベルを CMOS レベル（TTL 互換）に変換する。ともにチップ 1 つ当たり 4 チャネルの変換が可能で、155.5 Mbps（77.7 MHz）以上のスイッチングレートを有する。これらのチップの主な性能を表 G.1 と表 G.2 に示す[1]。

Parameter	Min	Typ	Max
Differencial Output Voltage [mV]	250	345	450
Offset Voltage [V]	1.125	1.25	1.35
Input Voltage High [V]	2.0		V_{CC}
Input Voltage Low [V]	GND		0.8
Differencial Propagation Delay from High to Low [ns]	1.0	2.0	3.0
Differencial Propagation Delay from Low to High [ns]	1.0	2.0	3.0
Supply Voltage V_{CC} [V]	4.5	5.0	5.5

表 G.1： LVDS ドライバ DS90C031 の主な性能。スイッチングレートは 155.5 Mbps（77.7 MHz）である。

Parameter	Min	Typ	Max
Differencial Input High Threshold [mV]			+100
Differencial Input Low Threshold [mV]	-100		
Input High Voltage [V]	2.0		
Input Low Voltage [V]	GND		0.8
Differencial Propagation Delay from High to Low [ns]	1.5	3.40	5.0
Differencial Propagation Delay from Low to High [ns]	1.5	3.48	5.0

表 G.2： LVDS レシーバ DS90C032 の主な性能。スイッチングレートは 155.5 Mbps（77.7 MHz）である。

[1] これらのデータシートは National Semiconductor 社のホームページ `http://www.national.com` から入手できる。

Appendix H

デランダマイザの深さ

ATLAS の FE エレクトロニクスに対する要請から、デランダマイザでのデッドタイム（データの損失）は L1A のレートが 75 kHz のとき 1% 以下、100 kHz の時 6% 以下でなければならない、と定められている。この要求を満たすためにデランダマイザバッファの深さをどの程度にするべきかを見積もった。

特に Slave Board のデランダマイザについて考えてみる。L1A 信号が来るとデランダマイザ内にデータを取り込む。L1A は完全にランダムに到達すると考えてよい[1]。一方、デランダマイザ内のデータは順次 PSC でシリアル化され、LS-Link を通じて Star Switch に送られる。シリアル化に要する時間は一定である。このような、ランダムに到着するイベントをバッファに蓄え一定時間の処理を行ったあと出力するという系は、待ち行列理論では M/D/1 の系と呼ばれる。

今、バッファが無限に深いと仮定し、このような系が十分時間がたって平衡状態になったとする。このときにバッファ内に n 個のデータが蓄えられている確率を P_n とする。このとき、$P(n > N) = \sum_{i=N+1}^{\infty} P_i$ という量を考えると、これはバッファの深さが N であると仮定した場合にバッファがオーバーフローする確率の目安となる。厳密にはバッファの深さが有限であるモデルを用いて解析を行う必要があるが、モジュールの設計上はこの近似で十分である。従って、バッファの深さを見積もるためには、P_n を求めればよいことになる。

M/D/1 の系では、処理時間の分布（処理中のイベントに対する残りの処理時間）が時刻 t に依存する。そこで、処理が終了した時間を特別に扱う。i 番目の処理が終了した時のバッファの占有数を n_i とし、i 番目の処理の途中に到着したイベントの数を V_i とおくと、

$$n_j = \begin{cases} (n_{j-1} - 1) + V_j & (n_{j-1} \geqq 1) \\ V_j & (n_{j-1} = 0) \end{cases} \tag{H.1}$$

となる。簡単のため、

$$n^* = \begin{cases} n & (\mathrm{n} \geqq 0) \\ 0 & (\mathrm{n} ¡ 0) \end{cases}$$

と表記すると、(H.1) は

$$n_j = (n_{j-1} - 1)^* + V_j \tag{H.2}$$

とかける。

ここで、母関数

$$G_n(Z) = \sum_{k=0}^{\infty} P(n = k) Z^k \tag{H.3}$$

[1]実際には一旦 L1A が発生するとその後の 4 クロックの間は L1A が抑制されるので完全にはランダムではない。

を定義する。このとき、独立な変数 X, Y の母関数を $G_X(Z), G_Y(Z)$ と書くと、$W = X + Y$ の母関数は $G_W(Z) = G_X(Z)G_Y(Z)$ となる。従って、(H.2) において、n_j, $(n_{j-1} - 1)^*$, V_j の母関数を $G_{n_j}(Z)$, $G_{a_j}(Z)$, $G_{V_j}(Z)$ と書くと、各々の母関数の間に、

$$G_{n_j}(Z) = G_{a_j}(Z)G_{V_j}(Z)$$

という関係があるが、さらに定常状態を考えると

$$G_n(Z) = G_a(Z)G_V(Z) \tag{H.4}$$

が成り立つ。

ところで、

$$\begin{aligned} G_a(Z) &= P[(n-1)^* = 0] + P[(n-1)^* = 1]Z + P[(n-1)^* = 2]Z^2 + \cdots \\ &= P_0 + P_1 + P_2 Z + \cdots \\ &= P_0 + \sum_{k=1}^{\infty} P_k Z^{k-1} \end{aligned}$$

に対して $G_n(Z) = \sum_{k=1}^{\infty} P_k Z^k$ を代入すると、

$$G_a(Z) = P_0 + \frac{G_n(Z) - P_0}{Z_0}$$

が得られるので、これを (H.4) に代入して変形すると

$$G_n(Z) = \frac{P_0(Z-1)G_V(Z)}{Z - G_V(Z)} \tag{H.5}$$

が得られる。

ここで母関数の性質から、$E(n)$, $E(n^2)$ をそれぞれ平均、二乗平均とすると、

$$\begin{aligned} G_n(Z) &= \sum_k P_k Z^k \\ \frac{dG_n(Z)}{dZ} &= \sum_k k P_k Z^k \qquad \therefore \ \left.\frac{dG_n(Z)}{dZ}\right|_{Z=1} = E(n) \\ \frac{d^2G_n(Z)}{dZ^2} &= \sum_k k(k-1) P_k Z^k \qquad \therefore \ \left.\frac{d^2G_n(Z)}{dZ^2}\right|_{Z=1} = E(n^2) - E(n) \end{aligned}$$

という関係があるので、

$$G_n(Z) = 1 + E(n)(Z-1) + \frac{1}{2}\{E(n^2) - E(n)\}(Z-1)^2 + \cdots$$

が成り立つ。これを (H.5) に代入すると

$$G_n(Z) = \frac{P_0(Z-1)\{1 + E(V)(Z-1) + \cdots\}}{(Z-1) - E(V)(Z-1) - \cdots}$$

となり、$n \to 1$ とすることにより、

$$P_0 = 1 - E(V) \tag{H.6}$$

が得られる。

V は処理時間中に到達するイベントの数であるから、$P(V=n)$ はポワソン分布に従う。処理時間を τ_0 単位時間あたりのイベントの到達確率を λ とすると、

$$P(V=n) = \frac{(\lambda\tau_0)^n e^{-\lambda\tau_0}}{n!} = \frac{(\mu)^n e^{-\mu}}{n!}$$
$$E(V) = \lambda\tau_0 = \mu$$

となる。但し、$\mu = \lambda\tau_0$ とおいた。これより、(H.3) (H.6) から $G_V(Z)$ と P_0 は

$$G_V(Z) = e^{-\mu(1-Z)}$$
$$P_0 = 1-\mu$$

求まる。以上から (H.5) は、

$$G_n(Z) = \frac{(1-\mu)(Z-1)}{Ze^{(1-Z)\mu} - 1} \tag{H.7}$$

と計算できる。

P_n を求めたければ、(H.7) を Z についてテーラー展開を行えばよい。この計算は煩雑になるので省略する。結果は、

$$P_n = (b_n - b_{n-1})(1-\mu) \quad \left(\text{但し} \quad b_n = \sum_{k=0}^{n} \frac{e^{\mu k}}{(n-k)!}(-\mu k)^{n-k} \right) \tag{H.8}$$

となる。

さて、以上の計算にもとづいて Slave Board のデランダマイザのオーバーフローの確率を見積もってみる。デランダマイザのバッファの深さを N とすると、オーバーフローの確率は近似的に、$P(n>N) = \sum_{i=N+1}^{\infty} P_i$ と考えてよい。この式にもとづいて、N とオーバーフローの確率の関係を求めた。Slave Board から出力されるシリアルデータのフォーマットは表 4.6 を仮定する。この表でチェックサムビットが 8 ビットであると仮定すると、シリアルデータの長さは開始コードと終了コードを合わせて 298 ビットである。従って、データを PSC でシリアル化するのに必要な処理時間は 298 クロックである。一方、L1A 信号は 75 kHz または 100 kHz で到達する。100 kHz の時には

$$\mu = \lambda\tau_0 = (100\times10^3)\times298 \div (40\times10^6) = 0.745$$

となる。これを (H.8) に代入して計算した結果を図 H.1 に示す。L1A が 75 kHz の場合の結果も同時に示す。この図を見る上で注意しなければならないのは、これはあくまでもバッファが無限の深さをもつと仮定した場合の計算であって目安に過ぎないこと、図中のバッファの深さには Slave Board の PSC にある 1 段分のバッファ（シリアル化を行うために一旦保存しておくバッファ）が含まれていることである。

FE エレクトロニクスに要求される性能としては、デランダマイザのデッドタイム（データの損失）が L1A のレートが 75 kHz のとき 1% 以下、100 kHz の時 6% 以下でなければならないということである。図 H.1 から、そのためにはバッファの深さが 6 段以上は必要であることが分かる。

但し、実際には TGC 読み出し系のデランダマイザは Slave Board の他、Star Switch にも分散されている。このようにデランダマイザが分散されている場合の各デランダマイザの性能については文献 [10] などにも特に触れられていないようだが、上記のデッドタイムの条件は、全てのデランダマイザでのデッドタイムの合計と解釈しておくのが無難である。その意味では、Slave Board の段階でのデッドタイムはさらに少なくするべきである。さらに Star Switch は FPGA で実装する関係上、十分な深さのバッファを用意できない可能性もある。そこで、100 kHz でも 1 % 程度しかデッドタイムがないという深さを当面の目安として考えることにした。図 H.1 からその時のバッファの深さは 9 段である。デランダマイザのバッ

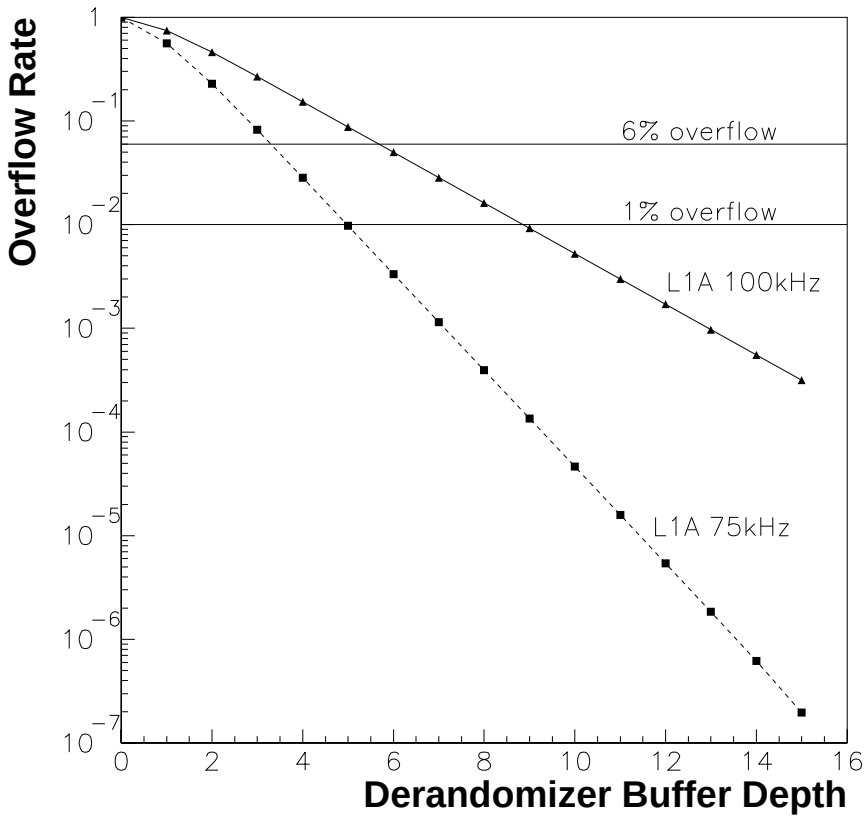

図 H.1： Slave Board のデランダマイザのオーバーフローの確率の見積もり。バッファが無限に深い系をモデルにして計算した。L1A が 75 kHz の場合と 100 kHz の場合を示す。

ファの深さとしては、PSC のバッファを除いて 8 ということになる。これに基づいて、今回のシミュレーションや検証の際には、デランダマイザの深さを 8 段とした。

一方、Star Switch のデランダマイザの深さを見積もるためには、Star Switch のデータバスのプロトコルが確定する必要がある。しかし、Star Switch は FPGA で実装する場合にはあまり深いデランダマイザは用意できないかもしれない。ただ、Slave Board から到達するデータは、Slave Board の PSC の処理時間に等しいクロックの間隔をおいて到達する。すなわち、上記の見積もりで用いた値を用いると、Star Switch には一旦データが到達すると、それに続く 297 クロックの間はデータは来ない。従って、この間にデータバスを通じてデータの収集が可能であれば、原理的には Star Switch にはデランダマイザが必要ないということになる。実際には、なんらかの原因でヒットが多くてゼロサプレスが有効で無かった場合などを想定すると用意しないわけには行かないが、とりあえずは数段のデランダマイザがあれば十分と思われる。深いデランダマイザが必要になれば FIFO を外付にする方法もあるので、どの程度の深さにするかを確定するのはデータバスのプロトコルを確定してからで良いと思われる。

略称と表記法

略称

略称	正式名称	参照章
ABS	Assembly Breakdown Structure	A
ADC	Analog to Digital Converter	
ASD	Amplifier Shaper Discriminator	3
ASIC	Application Specific Integrated Circuit	4
BCID	Bunch Crossing Identifier	2
BCR	Bunch Counter Reset	2
BSC	Boundary Scan Cell	B
BSR	Boundary Scan Register	B
CAN	Controller Area Network	2
CLB	Configurable Logic Block	D
CSC	Cathode Strip Chamber	1
CTP	Central Trigger Processor	2
DAC	Digital to Analog Converter	
DAQ	Data Acquisition	2
DCS	Detector Control System	2
DR	Data Register	B
ECR	Event Counter Reset	2
EF	Event Filter	2
FE	Front-end	4
FE-Link	Front-end Link	4
FPGA	Field Programmable Gate Array	D
HDL	Hardware Description Language	4
IR	Instruction Register	B
JTAG	Joint Test Action Group	B
LAr	Liquid Argon	1
LDM	Local DAQ Master	4
LDB	Local DAQ Block	4
LHC	Large Hadron Collider	1
LMB	Local Monitor Board	2

略称	正式名称	参照章
LSI	Large Scale Integrated Circuit	
LS-Link	Local Slave Link	4
LVDS	Low Voltage Differential Signaling	G
LVL1	Level 1	2
LVL2	Level 2	2
L1A	Level 1 Accept Signal	2
L1ID	Level 1 Trigger Accept Identifier	2
MDT	Monitored Drift Tube	1
MUCTPI	Muon trigger / CTP Interface	2
OBS	Organization Breakdown Structure	A
PBS	Product(Project) Breakdown Structure	A
PSC	Parallel Serial Converter	4
PSM	Programmable Switch Matrix	D
ROB	Readout Buffer	2
RoI	Region of Interest	2
ROD	Readout Driver	2
RPC	Resistive Plate Chamber	1
RTL	Register Transfer Level	4
SCT	Semiconductor Tracker	1
SEU	Single Event Upset	C
SPC	Serial Parallel Converter	4
SUSY	Super Symmetry	1
TGC	Thin Gap Chamber	1
TOF	Time Of Flight	4
TRT	Transition Radiation Tracker	1
TTC	Timing, Trigger and Control	2
WBS	Work Breakdown Structure	A

表記法

$E_{\mathrm{T}}^{\mathrm{miss}}$	transverse missing energy
H	Higgs
j	jet
l	lepton
m	mass
p_{T}	transverse momentum
η	pseudo rapidity

参考文献

[1] ATLAS Letter of Intent for a General Purpose pp Experiment at the Large Hadron Collider at CERN, CERN/LHCC/92-4, LHCC/I2 (1992).

[2] ATLAS Technical Proposal for a General Purpose pp Experiment at the Large Hadron Collider at CERN, CERN/LHCC/94-43, LHCC/P2 (1994).

[3] ATLAS First-Level Trigger Technical Design Report, CERN/LHCC/98-14 (1998).

[4] ATLAS Muon Spectrometer Technical Design Report, CERN/LHCC/97-22 (1997).

[5] ATLAS Inner Detector Technical Design Report, Volume 1, CERN/LHCC/97-16 (1997).

[6] ATLAS DAQ, EF, LVL2 and DCS Technical Progress Report, CERN/LHCC/98-16 (1998).

[7] ATLAS Detector and Physics Performance Technical Design Report Volume 1, CERN/LHCC/99-14 (1999).

[8] ATLAS Detector and Physics Performance Technical Design Report Volume 1, CERN/LHCC/99-15 (1999).

[9] ATLAS Technical Co-ordination Technical Design Report, CERN/LHCC/99-01 (1999).

[10] Trigger and DAQ Interfaces with Front-End Systems: Requirement Document (version 2.0), ATLAS note DAQ-NO-103 (1998).

[11] ATLAS Muon Trigger User Requirements Document (Draft version 1.4), ATLAS working document ATL-DA-ES-0002 (1998).

[12] ATLAS/TGC Master Database `http://www.cern.ch/Atlas/project/TGC/www/design/tgc.pdf`, Daniel Lellouch, Lorne Levinson, Meir Shoa

[13] 東京大学 陣内修 修士学位論文「ATLAS 実験ミューオン検出器用トリガーエレクトロニクスの開発」(1997).

[14] 東京大学 佐藤構二 修士学位論文「ATLAS 実験ミューオン検出器用データ読出システムの開発」(1999).

[15] 東京大学 松浦聡 修士学位論文「ATLAS 実験前後方部ミューオントリガーシステムの開発」(1999).

[16] Michael Spira, QCD Effects in Higgs Physics, CERN-TH/97-68, hep-ph/9705337 (1997).

[17] PMI Standards Committee, A Guide to the Project Management Body of Knowledge (1996).

[18] Texas Instruments, IEEE Std 1149.1 (JTAG) Testability Primer (1996).

[19] Xilinx http://www.xilinx.com, An Introduction to Xilinx Products (1999).

[20] Xilinx, Spartan and SpartanXL Families Field Programmable Gate Arrays (Version 1.4) (1998).

[21] Xilinx, XC4000XLA/XV Families Field Programmable Gate Arrays (Version 1.2) (1999).

[22] Xilinx, Virtex^{TM} Families Field Programmable Gate Arrays (Version 1.8), DS03 (2000).

[23] M.Hihtinen, F. Faccio, Computational method to estimate Single Event Upset rates in an accelerator environment, CERN CH-1211 (1999)

[24] Peter Alfke and Rick Padovani, Radiation Tolerance of High-Density FPGAs, Xilinx Inc

[25] Neutron Single Event Upsets in SRAM-based FPGAs, available from http://www.xilinx.co.jp/products/hirel_qml.htm

[26] National Semiconductor http://www.national.com, An Introduction of LVDS Technology, Application Note 971 (1996).

[27] National Semiconductor, LVDS Owner's Manual & Design Guide (1997).

[28] 野口正一・木村英俊・大庭弘太郎, 岩波講座情報科学 5「情報ネットワークの理論」, 岩波書店