修士論文

ATLAS実験ミューオン検出器用
データ読出システムの開発

東京大学大学院理学系研究科物理学専攻

76084
佐藤 構二

1999年1月

概 要

欧州原子核研究機構 (CERN) において、大型陽子・陽子衝突型加速器による実験 (LHC 実験) が 2005 年にスタートすることが予定されている。この LHC 実験のなかで、Higgs 粒子、超対称性粒子の探索、B メソンの崩壊における CP 非保存およびトップ・クォークの性質の精密測定などを目的として、ATLAS 検出器が現在建設準備中である。この ATLAS 実験では、高頻度のバックグラウンド・イベントのなかから重要なイベントを選びだすために、ミューオンの検出およびそのデータ読み出しが重要な課題となる。ミューオン・トリガー・チェンバーではヒット・レートがそれほど高くはないので、データを圧縮して読み出すのが効率的であり、本論文ではこの読み出しシステムの設計について述べる。また、エンドキャップ部ミューオン・トリガー・チェンバーの読み出しシステムでは、大部分の重要な回路を ASIC で実現することを予定している。この準備として、MWPC 読み出し用 ASIC を試作した。

目 次

1 ATLAS 実験

1.1 LHC 加速器実験

欧州原子核研究機構 (CERN) において、2005 年の実験開始にむけて大型陽子・陽子衝突型加速器 (LHC) の建設が進行中である。この LHC 加速器は、陽子・陽子衝突の加速器であり、現在稼働中の加速器のエネルギーを超える TeV 領域における素粒子物理の探索を可能にするものである。このエネルギー領域では、Higgs 粒子、超対称性粒子の探索、現在までに知られていない相互作用の発見など、素粒子物理の根源的な理解の鍵となる研究が期待されている。また、一方では、B メソンの崩壊における CP 非保存やトップ・クオークの性質の精密測定も可能になる。

LHC 加速器の主要なパラメータは表 1 のようになっている。

衝突粒子	陽子・陽子
陽子エネルギー	7TeV
ルミノシティー (低ルミノシティー運転時)	$10^{33}\mathrm{cm}^{-2}\mathrm{s}^{-1}$
ルミノシティー (高ルミノシティー運転時)	$10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$
バンチ・クロッシング・レート	40MHz
陽子・陽子衝突レート	$10^{9}/\mathrm{s}$
バンチ・クロッシングあたりのイベント数	~ 25
ビームの寿命	22 時間
トンネル周長	26.66km

表 1: LHC 加速器の主要なパラメータ。

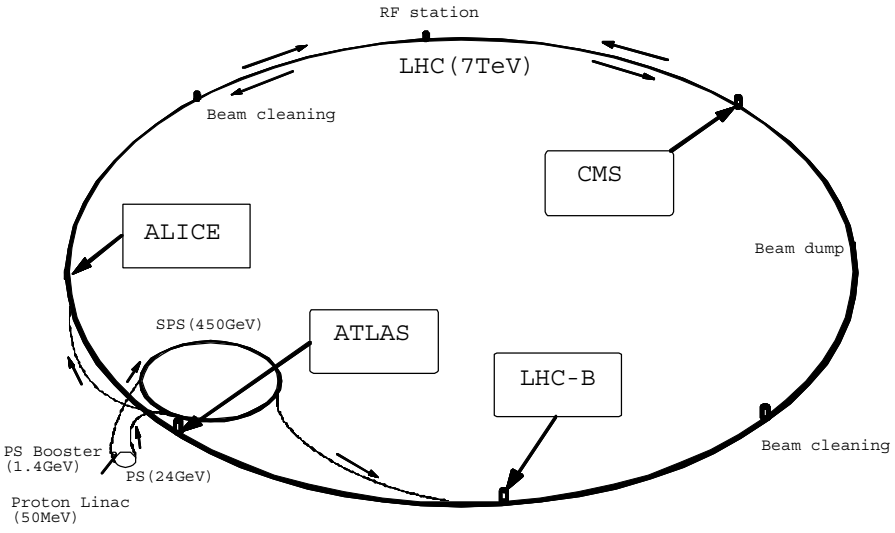

図 1: LHC の検出器の配置

LHC は、現在の LEP-II のトンネルに建設されることになっている。LHC では、図 1 のように、高ルミノシティー陽子・陽子衝突実験を目的とした ATLAS(A Troidal LHC Apparatus) と

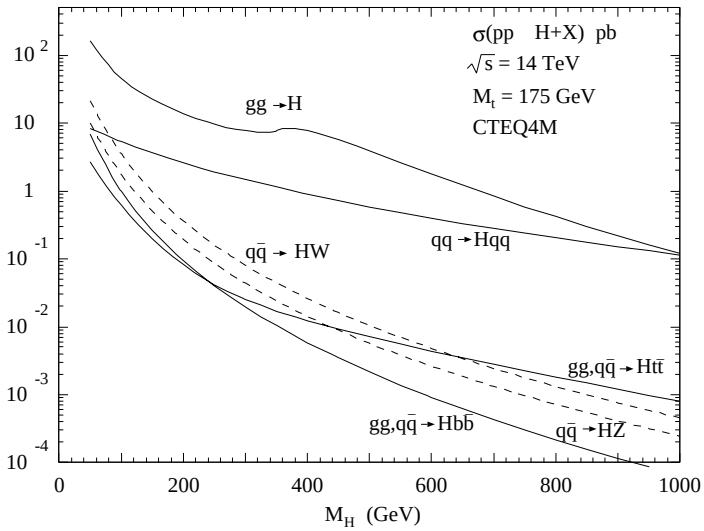

図 2: LHC($\sqrt{s} = 14$TeV) における標準 Higgs 粒子の質量と生成断面積の関係。トップ・クォークの質量を 175GeV として計算している。

CMS(The Compact Muon Solenoid)、b クォークの物理の精密測定を目的とする LHC-B、重イオン実験のための ALICE(A Large Ion Collider Experiment) の 4 検出器での測定が計画されている。

1.2 ATLAS 実験で期待されている物理

1.2.1 標準 Higgs 粒子の探索

Higgs 粒子の探索は、LHC 実験におけるもっとも重要な課題のひとつである。標準 Higgs 粒子が存在するのであれば、その質量は 1TeV 以下であることが理論により予想されている [3]。また、LEP-II によって探索が可能な標準 Higgs 粒子の質量は、約 100GeV 以下である [4]。LHC では、80GeV $\sim$ 1TeV までの質量領域で標準 Higgs 粒子を探索することが可能なので、標準 Higgs 粒子の存在が期待される質量領域を完全に網羅することができる。

図 2 に、標準 Higgs 粒子のさまざまな生成過程に対する生成断面積を、図 3 に、断面積の大きい生成過程の基本プロセスのダイアグラムをしめす [5]。

つぎに、標準 Higgs 粒子のさまざまな崩壊モードの分岐比を、図 4 に示す。

この図を見てわかるように、標準 Higgs 粒子の質量によって、崩壊モードの分岐比が変化する。したがって、探索する質量領域により、着目する崩壊モードを変える必要がある。

80 $\sim$ 120GeV もっとも分岐比が大きい崩壊モードは、$H \longrightarrow b\bar{b}$ であるが、このモードは、陽子・陽子衝突で生成された $b\bar{b}$ 対と区別することが困難である。

$H \longrightarrow \gamma\gamma$ のモードがこのエネルギー領域ではもっとも有効である。このモードは、エネルギー、角度を精密に測定すれば、不変質量分布のなかで標準 Higgs 粒子の質量が、幅の狭いピークとなってみえるはずである。

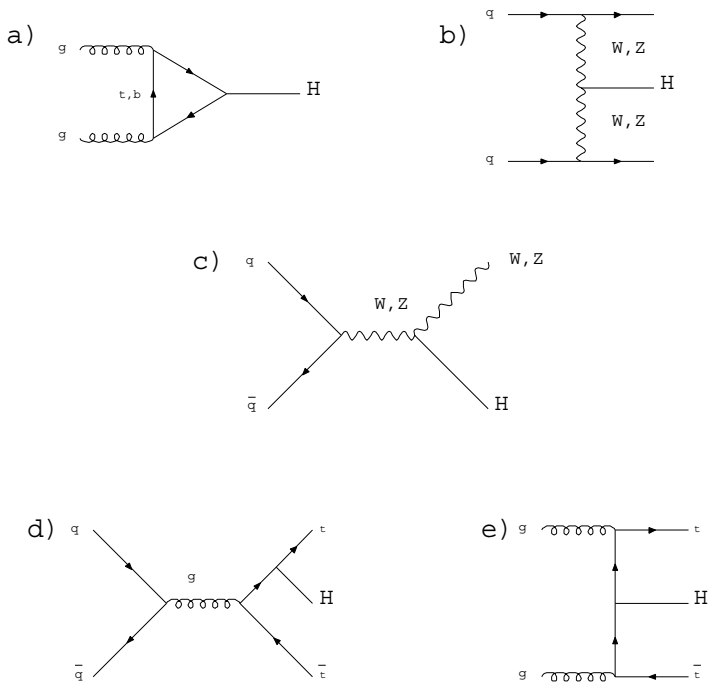

図 3: LHC における標準 Higgs 粒子の主な生成過程。a) グルオン-グルオン融合。b) WW、ZZ 融合、c) W、Z を伴う生成、d),e) トップ・クォークを伴う生成

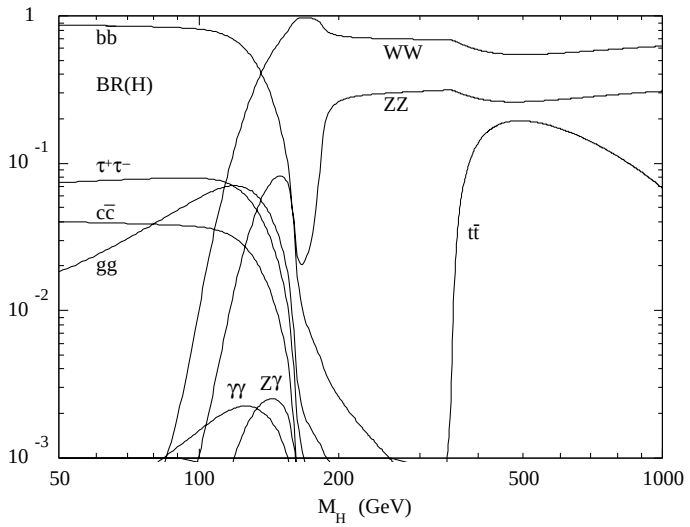

図 4: 標準 Higgs 粒子の崩壊モードの分岐比と質量の関係。トップ・クォークの質量を 175GeV として計算している。

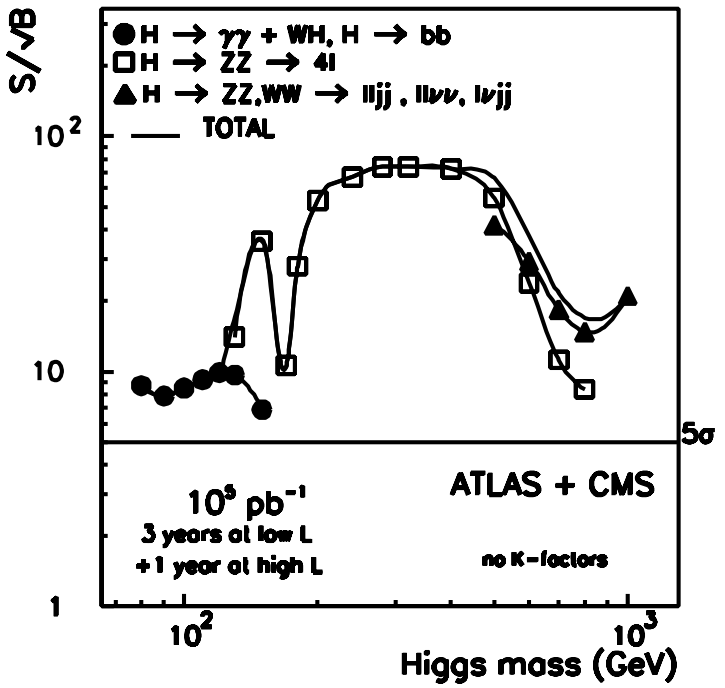

図 5: $10^5\,\mathrm{pb}^{-1}$ の積算ルミノシティーにおける Higgs 質量の関数としての ATLAS の Higgs 粒子に対する信号対雑音比。

さらに、W あるいは $t\bar{t}$ とともに標準 Higgs 粒子が生成された場合には、それぞれ終状態は $lb\bar{b}$、$lbb\bar{b}$ であり、バックグラウンド・イベントと識別することができる。

120 ∼ 180GeV このエネルギー領域以上では、おもに W 粒子対、Z 粒子対への崩壊モードが主なモードとなる。120GeV $< m_H <$ 180GeV では、$H \longrightarrow ZZ^* \longrightarrow llll$(一方の Z が off shell) のモードがもっとも有効な探索モードとなる。

180 ∼ 800GeV このエネルギー領域では、$H \longrightarrow ZZ$ のモードがレートも高く、バックグラウンドも少ないので、このモードがもっとも有効である。

800GeV **以上** このエネルギー領域では、Higgs 粒子の崩壊幅が広くなるので、信号とバックグラウンドの区別が困難になる。そこで、$H \longrightarrow ZZ \longrightarrow llll$ の 6 倍のレートが期待される $H \longrightarrow ZZ \longrightarrow ll\nu\nu$ を主に使うことになる。ただし、このモードによる Higgs 粒子の探索では、missing-E_T を精度良く測定することが必要となる。

このほか、この領域では、$qq \longrightarrow qqWW \longrightarrow qqH$, $qq \longrightarrow qqZZ \longrightarrow qqH$ のプロセスで生成された Higgs 粒子の $H \longrightarrow WW \longrightarrow l\nu jj$, $H \longrightarrow ZZ \longrightarrow lljj$ のモードによる崩壊も Higgs 粒子探索に用いられる。このモードは、散乱角前方の qq によるジェットを指標とすることでバックグラウンドと区別できる。

$10^5\,\mathrm{pb}^{-1}$ の積算ルミノシティーの時点での ATLAS の標準 Higgs に対する信号対雑音比 $S/\sqrt{B}$ は、図 5 のようになる。

1.2.2 超対称性 Higgs 粒子の探索

MSSM(Minimal Sypersymmetric extention of the Standard Model) 理論は、もっとも簡単な標準理論の拡張である。MSSM 理論では、2 つの Higgs 二重項が導入され、その結果、3 つの中性 Higgs 粒子 h、H、A と 2 つの荷電 Higgs 粒子 H^+、H^- が存在することになる。これら 5 つの Higgs 粒子の質量は 2 つのパラメータで記述することができる。

実際の探索は、以下のチャンネルを探すことで行なわれる。

- $pp \longrightarrow h, H, Wh, t\bar{t}h \qquad h, H \longrightarrow \gamma\gamma$
- $pp \longrightarrow WH \qquad h \longrightarrow b\bar{b} \quad W \longrightarrow l\nu$
- $pp \longrightarrow H \qquad H \longrightarrow ZZ^{(*)} \longrightarrow 4l$
- $pp \longrightarrow H, A \qquad H, A \longrightarrow \tau\tau$
- $pp \longrightarrow H, A \qquad H, A \longrightarrow \mu\mu$
- $pp \longrightarrow H \qquad H \longrightarrow hh \longrightarrow bb\gamma\gamma$
- $pp \longrightarrow H, A \qquad H, A \longrightarrow t\bar{t} \longrightarrow bjets$
- $pp \longrightarrow A \longrightarrow Zh \longrightarrow llbb$
- $pp \longrightarrow t \longrightarrow bH^+ \qquad H^+ \longrightarrow \tau\nu$

5 つの Higgs 粒子の質量を m_A と 2 つの真空期待値の比 $\tan\beta$ であらした場合に、ATLAS 実験で $3 \times 10^5 pb^{-1}$ の積算ルミノシティーにおいて、探索可能なパラメータ領域を図 6 に示す。

1.2.3 超対称性粒子の探索

R-パリティ保存則を仮定すると、超対称性粒子は必ずペアで生成される。また超対称性粒子は、生成されると、崩壊を繰り返し、最軽量の超対称性粒子 (LSP) で止まる。この LSP としては、最軽量ニュートラリーノ $\tilde{\chi}_1^0$ が考えられる。

ATLAS 実験では、以下のようなモードでの超対称性粒子の探索が予定されている。

Multijet + E_T^{miss} モード ATLAS 実験では、強い相互作用をするグルイーノ ($\tilde{g}$) の対、スクォーク ($\tilde{q}$) の対が多く生成される。たとえば、$m_{\tilde{g}} > m_{\tilde{q}} + m_q$ の場合には、$\tilde{g}$、$\tilde{q}$ の崩壊モードには、

$$\begin{aligned} \tilde{g} &\longrightarrow \tilde{q}q \\ \tilde{q} &\longrightarrow q\tilde{\chi}_1^0 \end{aligned}$$

などがある。

このように $\tilde{g}$、$\tilde{q}$ が崩壊する際にできる複数の high-P_T ジェットと、$\tilde{\chi}_1^0$ の生成による missing E_T の情報を使うことで、$\tilde{g}\tilde{g}$、$\tilde{q}\tilde{q}$ のイベントを同定する。

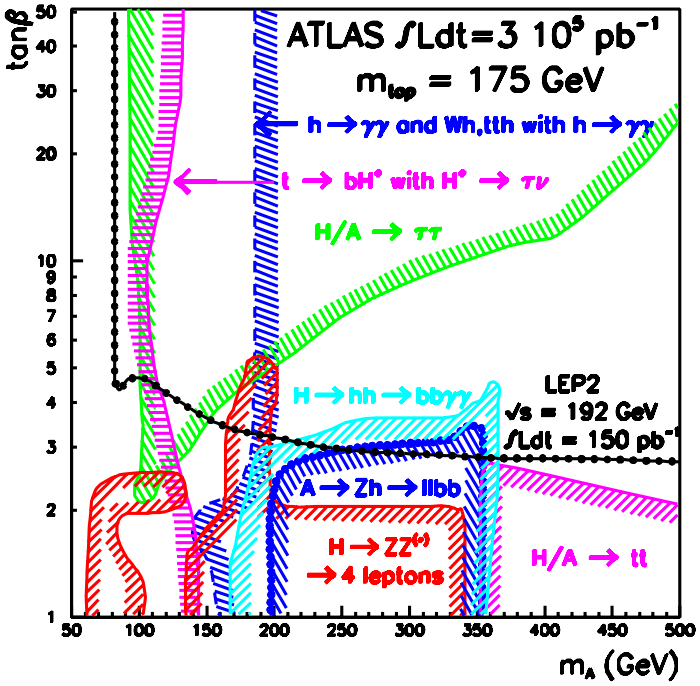

図 6: $3 \times 10^5 pb^{-1}$ の積算ルミノシティーにおける ATLAS 実験での MSSM-Higgs 粒子探索可能領域。

同符合の 2 レプトン・モード $\tilde{g}$、$\tilde{q}$ は、質量が大きいため、その崩壊過程では、最軽量ではないニュートラリーノ $\tilde{\chi}^0_{2,3,4}$ やチャージーノ $\tilde{\chi}^{\pm}_{1,2}$ が生成されることもある。これらの粒子を中間状態として $\tilde{g}$ や $\tilde{q}$ が崩壊する場合、

$$\tilde{g}\tilde{g}, \tilde{g}\tilde{q}, \tilde{q}\tilde{q}, \longrightarrow 2l^{\pm} + \text{n-jets} + E_T^{miss}$$

の終状態になることがある。この孤立した 2 つの同符合のレプトン、複数のジェット、missing$-E_T$ を要請することで探索を行なう。

3 レプトン・モード $\tilde{\chi}^0_{2,3,4}$ と $\tilde{\chi}^{\pm}_{1,2}$ が生成されたばあい、

$$\tilde{\chi_1^{\pm}}\tilde{\chi}^0_2 \longrightarrow lll\nu\tilde{\chi}^0_1\tilde{\chi}^0_1$$

がおこるが、この 3 つのレプトンと missing E_T を要請することで、終状態を同定する。

1.3 ATLAS 検出器

ATLAS 検出器は、LHC 加速器における新しい物理現象の発見の可能性を最大限に引き出すことを目指している。とくに、Higgs 粒子、超対称性粒子の探索、重い Z-や W-like の粒子の探索、B メソン崩壊における CP 非保存、トップ・クォークの精密測定を目的としており、これらのテーマからの要請を満足する設計となっている。この物理探索の要請をまとめるとつぎのようになる。

- 電子、光子を同定し、精度よく測定でき、それと相補的にジェット、missing-E_T を測定すること。
- レプトンの飛跡の再構成、ボトム・クォークが同定が効率よくできること。
- ミューオンの運動量を精度よく測定でき、さらに low-P_T ミューオン・トリガーを用意できること。
- 広い範囲の疑似ラピディティー[1]を測定できること。

さらに、以下のような LHC の環境からの要請がある。

- 大量のバックグラウンド放射線が予想されており、したがって検出器は放射線にたいして耐久性がなくてはならない。
- LHC は、ルミノシティーをあげるためにバンチ間隔を短くしているが、検出器の信号とバンチの対応がただしくとれなければならない。

ATLAS 検出器の全体図を図 7 に示す。直径 22m、長さ 42m であり、総重量は 7000 トンである。

ATLAS 検出器は、内部検出器、カロリメータ、ミューオン検出器から構成される。以下に、これらの特徴を挙げる。

内部検出器 内部にはバーテックスのまわりで高精度の位置分解能での測定を実現するピクセル検出器とシリコン・ストリップ検出器 (SCT) が配置される。そのまわりには、飛跡認識のために、直径 4mm のストロー・チューブ検出器を多層に積みかさねた TRT(Transition Radiation Tracker) を配置する。この TRT では、ストロー・チューブの間にラディエータをはさみ、そこからの遷移放射を測定することで、電子の同定をも行なう。内部検出器では、ソレノイドの磁場を利用して、運動量の測定も行なう。$|\eta| < 2.5$ のラピディティーの範囲をカバーする。

カロリーメータ 電磁カロリーメータは、耐放射線性にすぐれた液体アルゴン・カロリーメータをもちい、$|\eta| < 3.2$ のラピディティーの範囲をカバーする。とくに $H \longrightarrow \gamma\gamma$、$H \longrightarrow 4e$ のチャンネルの測定の測定のために、$\frac{10}{\sqrt{E(\mathrm{GeV})}}\%$ 程度の高いエネルギー分解能が必要である。一方、ハドロン・カロリーメータは、ジェットの再構成と missing E_T の測定をおこなう。バレル部 ($|\eta| < 3.2$) は、鉄の吸収体で囲まれたタイル状のプラスチック・シンチレータが配置され、放射線の多い $3.1 < |\eta| < 4.9$ の範囲には、銅の吸収体と液体アルゴンを組み合わせたカロリーメータを配置する。

ミューオン検出器 ミューオン検出器の特徴については、次節で詳細に述べる。

[1]高エネルギー散乱実験では、発生粒子の角分布を表現するのに疑似ラピディティーが用いられる。入射粒子の方向と発生粒子の方向のずれを θ で表した時、疑似ラピディティー η は $\eta = -ln(\tan\frac{\theta}{2})$ で定義される。

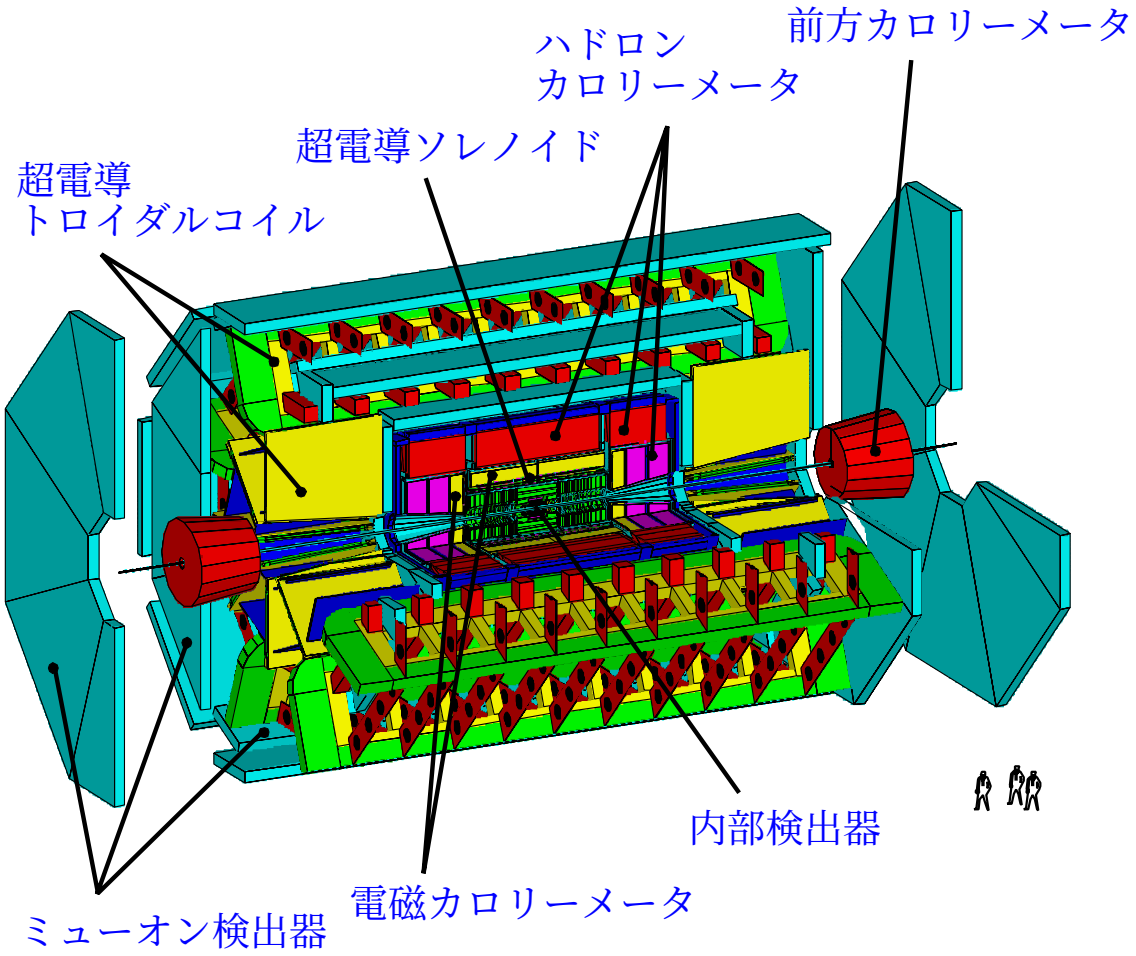

図 7: ATLAS 検出器の全体図。

1.4 ミューオン検出器

高運動量のミューオンは、LHC においては、新しい物理現象を探すためのもっとも確実な信号のひとつである。このため、ATLAS 検出器では広範囲の横運動量、ラピディティーをカバーするミューオン検出器を設計に入れている。図 8 にミューオン検出器の配置図を示す。

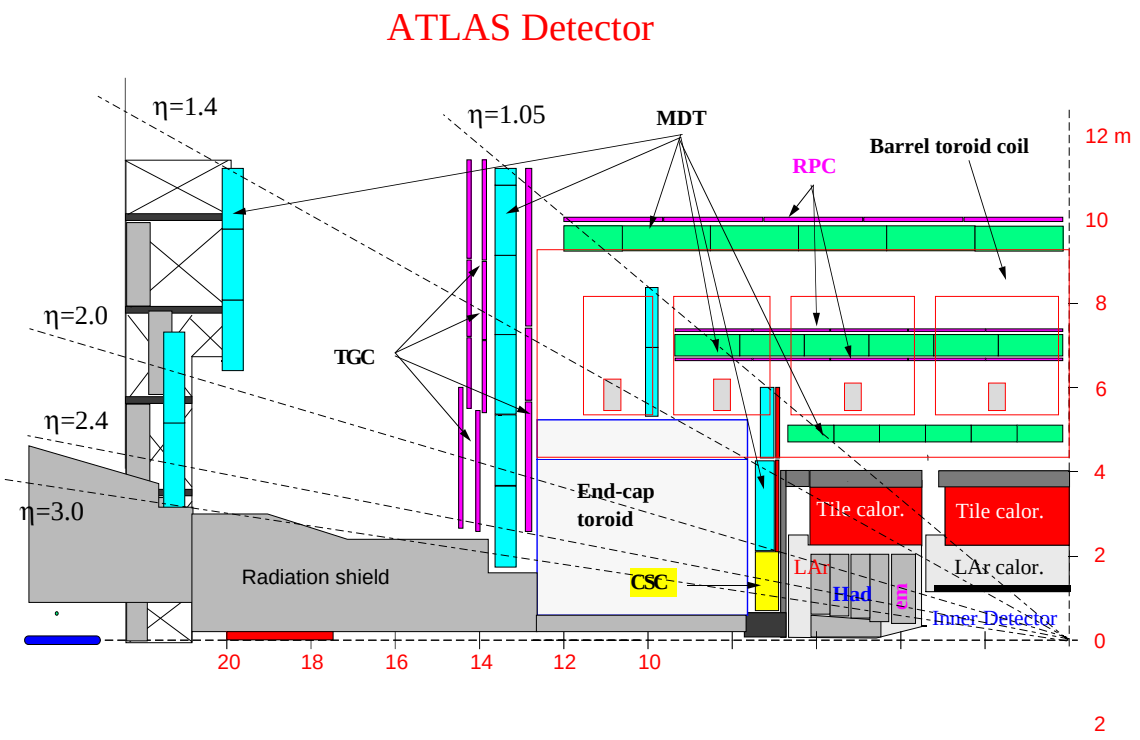

図 8: ATLAS 検出器の r-z 断面図。ミューオン検出器の配置が示されている。

バレル部 ($|\eta| \leq 1.0$) では、ハドロン・カロリーメータを包み込むように 8 個の空芯トロイド・コイルが配置され、このコイルのつくり出す磁場での飛跡の曲がりを測定することでミューオンの運動量を測定する。一方、エンドキャップ部 ($1.4 \leq |\eta| \leq 2.7$) には、2 個のトロイド・コイルが配置される。これらの中間の部分 (トランシション部) では、バレル・トロイドとエンドキャップ・トロイドがつくり出す合成磁場を利用する。

検出器は、運動量の精密測定のための MDT(Monitored Drift Tube) と CSC(Casode Strip Chamber) と、トリガーおよび第 2 座標測定[2] を行なう。RPC(Resisive Plate Chamber)、TGC(Thin Gap Chamber) で構成される。バレル部では、精密測定用検出器、トリガー用検出器とも 3 層の同心円状に配置され、エンドキャップ部においても、精密測定用検出器、トリガー用検出器ともに 3 層の構造をとる。以下、これらの検出器の特徴を挙げる [7]。

MDT チューブ径 30mm、ワイヤー径 50μm のドリフト・チューブを多層に積層した構造をしている。バレル部からエンドキャップ部にまたがるラピディティー領域をカバーし、磁場による曲がりの方向に平行な運動量第 1 座標方向成分を高精度で測定する。長い期間の使用に耐え得るように、圧の Ar-CH_4-N_2（混合率 91:4:5）を使用しガス・ゲインを低く (2×10^4) 押え

[2]磁場による曲がりの方向に平行な $r-z$ 面への投影を第 1 座標、$r-\phi$ 面への投影を第 2 座標と読んでいる。

ている。位置-ドリフト時間の線形性が非常によく、最大のドリフト時間は ~ 480ns である。80μm の位置分解能を達成している。

CSC 運動量精密測定用のストリップ読みだし用 MWPC(MultiWire Proportional Chamber) である。読みだし用ストリップは、ワイヤーと垂直に切ってあり、ワイヤー間隔とワイヤー・ストリップ間隔が等しく 2.54mm であり、ストリップの間隔は 5.08mm である。位置分解能は、隣接するストリップの電荷の重心をとることにより、60μm である。また、水素を含まない Ar-CO_2-CF_4(混合比 30:50:20) ガスを使用することにより、バックグラウンド中性子に対する感度をさげることも実現している。バックグラウンド放射線が多く、より高度の位置分解能が必要とされる高ラピディティー ($|\eta| \geq 2$) 領域の、最内部の運動量精密測定用検出器として使用される。

トリガー用検出器は、膨大なバックグラウンド・イベントのなかから物理的に重要なイベントを効率よく選び出すために、正確に運動量を測定する必要があり、このためには、数 cm の測定分解能が要求される。また、ビーム・バンチとの対応づけ (Bunch-ID) も要請されており、このためには、LHC のバンチ・クロッシング周期 (25ns) よりも短い時間分解能が必要である。

RPC 2 枚の抵抗性ベークライト板の間にガス ($C_2H_2F_4$-$C_4H_{10} - SF_6$、混合比 97:2:1) を閉じ込めた構造の検出器を 2 層重ねたものである。2 層のうち 1 層は、トリガー用に MDT のワイヤーと平行にストリップが切ってあり、もう 1 層は、第 2 座標方向運動量を測定するために MDT のワイヤーとは垂直にストリップが切ってある。ストリップ間隔は、30.0 $\sim$ 39.5mm である。ガス中にできた初期電子を 5kV/mm の電場によって増幅し、0.5pC のパルスを出力する。時間分解能は、1.5ns で、$|\eta| \leq 1.0$ の領域をカバーする。

TGC ワイヤー間隔 (1.8mm) がワイヤー・ストリップ間隔 (1.4mm) よりも長いことを除けば、MWPC と同じ構造である。ワイヤーは、5 $\sim$ 32 本をまとめて読みだし、第 1 座標測定をおこなう。ワイヤーに垂直に切ったストリップから第 2 座標を測定する。クエンチ能力の高い $CO_2 - n$-$pentane$(混合比 55:45) ガスを用いることで、構造のゆがみ、入射角にあまり依存しない動作を実現している。ワイヤー間隔を短くすることで、短い時間分解能を実現しており、25ns のゲート幅で 99% 以上の検出効率である。$1.0 \leq |\eta| \leq 2.4$ をカバーする。

2 ATLAS 実験のトリガー/DAQ の方針

2.1 ATLAS 実験のトリガーの方針

2.1.1 トリガー・スキーム

ATLAS 実験では、バンチ・クロッシングごとに ∼ 25 イベントの陽子・陽子間相互作用がおこるため、イベント・レートは 10^9Hz にのぼる。ATLAS 実験においてもっとも困難な課題は、その膨大なイベントのなかから効率的に物理的に重要なイベントを選び出していくことである。

ATLAS では、レベル 1、レベル 2、イベント・フィルターの 3 段階のトリガー・レベルを経て、イベントを選択する方式をとる。この様子を図 9 に示す。

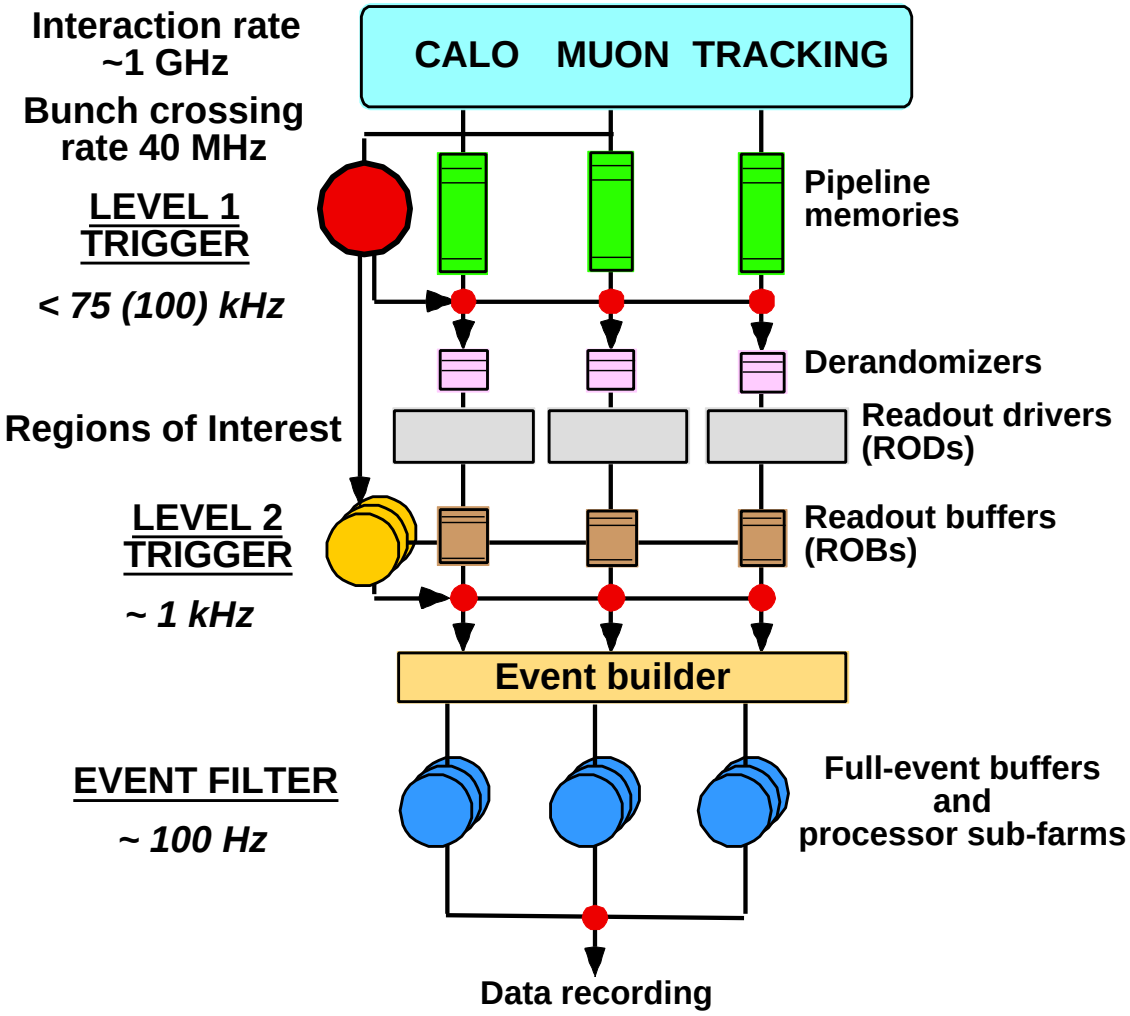

図 9: ATLAS 実験のトリガー・スキーム

レベル 1・トリガーは、LHC のバンチ・クロッシングの周期 (25ns) でデータを受け、最大 75kHz (100kHz までアップグレード可能) のレートでトリガーを出力をする。ATLAS 検出器を構成する検出器の読みだし回路系は、すべてこのレベル 1・トリガー・レートにあわせて設計される。レベル 1・トリガーは、バンチ・クロッシング後 2μs 以内に全検出器におくられることになっている。このレベ

ル 1・トリガーで選択されたイベントは、レベル 2 とイベント・フィルターで、さらに詳しいデータを使って最終的には ∼ 100Hz のイベント・レートまで絞られ、オフラインの解析のために記録される。

以下に、これら 3 段階のトリガー・レベルの特徴を述べる [8]。

レベル 1 全検出器に 2μs という短い時間の間にトリガーを配らなくてはならないので、レベル 1 では、高分解能のデータは用いず、また簡単なアルゴリズムでのイベント選択を行なう。イベントの選択には、ミューオン・トリガー用検出器、カロリーメータからのデータのみを用いる。ミューオン・トリガー検出器、カロリーメータでは、それぞれに、専用のトリガー回路が設けられ、ここでは、ミューオン、電磁クラスター、ジェットの P_T、missing-E_T、カロリーメータに残された全エネルギーが、設定されたしきい値よりも大きい場合に CTP(Central Trigger Processor) に信号を送る。なお、このしきい値は、複数用意されており、1 種類の物理量に対して、数種類のトリガー情報が用意できる。CTP では、ミューオン・トリガー用検出器、カロリーメータから送られてきたトリガー情報を総合しレベル 1・トリガーの判定を行ない、トリガーを TTC(Timing, Trigger and Control distribution) に伝える。トリガーをうけた TTC は、全検出器にトリガーを配る。このレベル 1・トリガーの判定の間、各検出器のデータは、それぞれに備え付けられたレベル 1・バッファ (L1B) と呼ばれるバッファに保持される。

また、ミューオンや電磁クラスタに関しては、その位置 (η, ϕ) の情報をレベル 2・トリガー・プロセッサに伝える。レベル 1・トリガーのプロセスの流れを図 10 に示す。

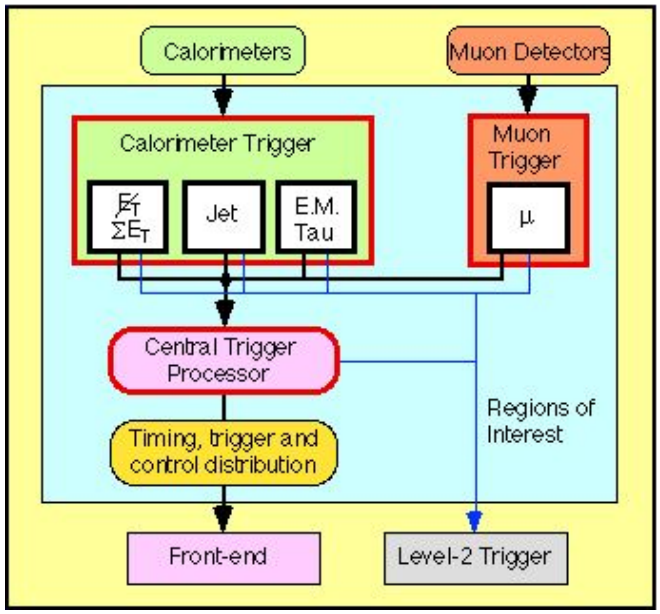

図 10: レベル 1・トリガーのプロセス。

レベル 2 レベル 2・トリガーは、レベル 1 から伝えられた位置のまわり (RoI(Region of Interest) とよばれる) のミューオン検出器、カロリーメータの高精度のデータ

にアクセスし、さらに、ピクセル、SCT、TRT のデータにもアクセスしながら、ミューオン、電子、光子、タウ、ジェットの候補を探す。この後、イベントの選択を行なう。レベル 2・トリガーのプロセスにかかる時間は、10ms 程度である。

イベント・フィルタ オンラインの最終段階のイベント選択である。ここでは、全検出器からの最高精度のデータを用い、バーテックスの再構成、トラックのフィッティングなどのイベントの再構成を行なった上でイベントの選択を行なう。このプロセスにかかる時間は、数秒である。

ここで、CTP、TTC についても簡単にふれておく [9]。

CTP ミュオン・トリガー用検出器とカロリーメータのデータから、レベル 1 トリガーの判断をくだすプロセッサ。レベル 1・トリガーは、以下のような制約がある。

- 1 つレベル 1・トリガーを出力したら、その後 4 クロックはつぎのトリガーを出さない。
- 16μs の間に出されるトリガーは、16 回を超えない。

TTC ATLAS 検出器全体に LHC のバンチ･クロッシングに同期したクロック (正確には 40.08MHz)、トリガー、コントロール信号を配る。以下に、TTC によって配られる信号のうち重要なものを挙げる。

- **バンチ・クロック** 全検出器のデータ読み出しをバンチ・クロッシングに同期して行なうためのクロック。
- **レベル 1・トリガー** レベル 1・トリガーの条件がすべて満たされた時に CTP から出されるトリガー。バンチ・クロックに同期。
- **BCR** すべてのフロント・エンドには、バンチ判別のためにバンチ・カウンタが用意される。このカウンタに対するリセット信号。クロックに同期。
- **ECR** すべてのフロント・エンドには、イベント判別のためにイベント・カウンタが用意される。このカウンタに対するリセット信号。クロックに同期。
- **テスト・コマンド** クロックに同期したテスト・コマンドを送ることができる。
- **パラメータ** アドレスを指定して送信することにより、個々のモジュールに対して動作パラメータを送ることができる。モジュールごとの信号の遅延を補正するためのパラメータやキャリブレーションのパラメータ。クロックとは非同期に送信される。

2.1.2 トリガー条件

表 2 にいくつかの重要な物理過程に対するレベル 1 とレベル 2 のトリガー条件を挙げる [2]。

物理過程	レベル 1・トリガー条件	レベル 2・トリガー条件
$(W \longrightarrow l\nu) + X$	$1\mu_{20}$,$1em_{30}$	$1\mu_{20}I$,$1e_{30}$
$(Z \longrightarrow l^+l^-) + X$	$2\mu_6$,$2em_{20}$, $1\mu_{20}$, $1em_{30}$	$2\mu_6I$,$2e_{20}$, $1\mu_{20}I$, $1e_{30}$
$H \longrightarrow \gamma\gamma$	$2em_{20}$	$2\gamma_{20}$
$H \longrightarrow ZZ^* \longrightarrow 4l$	$2em_{20}$,$2\mu_6$, $1em_{30}$, $1\mu_{20}$	$2e_{20}$,$2\mu_6I$, $1e_{30}$, $1\mu_{20}I$
$A \longrightarrow \tau^+\tau^-$	$1\mu_{20}$, $1em_{30}$	$1\mu_{20}I$, $1e_{30}$
$t\bar{t} \longrightarrow l^+l^- + X$	$2\mu_6$,$2em_{20}$,$1em_{30}$, $1\mu_{20}$	$2\mu_6I$,$2e_{20}$,$1e_{30}$, $1\mu_{20}I$
$t\bar{t} \longrightarrow l^{\pm}\nu + jets$	$1\mu_{20}$, $1em_{30}$	$1\mu_{20}$I, $1e_{30}$
$SUSY \longrightarrow leptons$	$1\mu_{20}$, $1em_{30}$	$1\mu_{20}I$, $1e_{30}$
$W', Z' \longrightarrow jets$	$1j_{150}$	$1j_{300}$,$1j_{200}$
$SUSY \longrightarrow jets$	$1j_{150}$, E_T^{miss}	$3j_{150}$, E_T^{miss}
B-physics	$1\mu_{20}$, $2\mu_6$	$1\mu_{40}$, $2\mu_{10}$, $B^0 \longrightarrow \mu^+\mu^-$

表 2: ATLAS 実験における重要な物理過程に対するトリガー条件。粒子名の添字は、その P_T のしきい値 (GeV)。

2.2 ATLAS 実験の DAQ の方針

図 11 に ATLAS 実験の DAQ の方針を示す [9]。

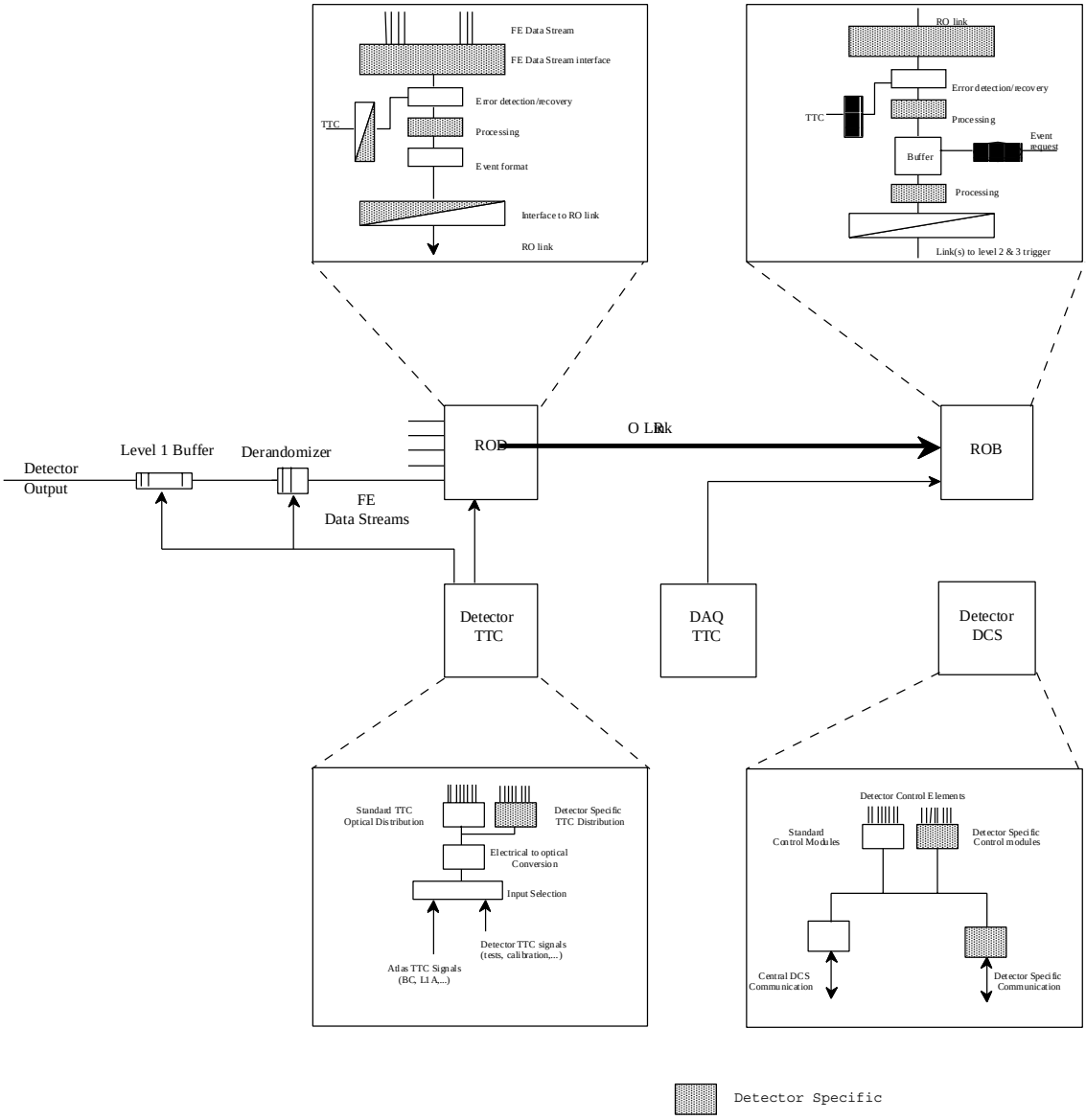

図 11: ATLAS 実験の DAQ・スキーム

まず、アナログ-デジタル変換された検出器のデータは、TTC からレベル 1・トリガーを受けとるまでの間、レベル 1・バッファ (L1B) と呼ばれるパイプラインに格納される。レベル 1・トリガーは、ビーム衝突から最大でも 2μs の間に各フロント・エンドに到達する。L1B は、回路系のデザインの変更などの事態による遅延の増加に備えて、この 2μs に 0.5μs を加えた、2.5μs 以上の間データを保持することになっている。

各フロント・エンドは、レベル 1・トリガーを受けとると、そのトリガーを出したバンチ・クロッシングに対応するデータを L1B から次の段階のバッファ (ROD) に送る。また、各フロント・エン

ドは、バンチ・クロッシング、レベル 1・トリガーの数を数えるカウンタを用意し、これらのカウンタの値 BID(Bunch Crossing ID) 、L1ID(Level 1 ID) をデータとともに送る[3]。

ただし、この L1B からのデータは、タイミングが一定でなく、データ量も多いので、デランドマイザ (Derandomizer) と呼ばれるバッファを中継する。

デランドマイザでは、データの圧縮を行なうことができ、この圧縮の形式は、各検出器ごとに決められる。ただし、データの圧縮をおこなう場合には、イベントごとにデータ・サイズが異なるので、デランドマイザのサイズを十分に大きくしてオーバーフローを防がなければいけない。なお、デランドマイザのサイズは、レベル 1・トリガー・レートが 75KHz、100kHz のときに、データ損失率がそれぞれ 1% 以内、6% 以内になるように設計する。また、キャリブレーションのためのオプションとして、レベル 1・トリガーに対応するバンチ、その前後のバンチの 3 クロック分までのデータを読み出せるように設計する。デランドマイザに貯められたデータは、ROD(Read Out Driver) に送られる。

ROD では、デランドマイザから集められたデータをイベントの順に整列し、最終的なデータ形式にフォーマットされる。この際、TTC から送られてくる 12bit の BID、24bit の L1ID と、データの BID、L1ID が一致しているかどうかをチェックし、一致していなければ、エラー・フラグを立てる。

ROD でフォーマットされたデータは、RO link(Read Out link) と呼ばれる全検出器で標準仕様の結線を通して ROB(Read Out Buffer) と呼ばれるバッファに送られる。この ROB の仕様も全検出器で統一される。ROD は、レベル 2・トリガーのプロセッサにデータを渡し、レベル 2・トリガーのプロセスの間、データを保持する。レベル 2・トリガーが出力されたイベントのデータに関しては、イベント・フィルタにデータを送る。なお、この ROB でも、ROD と同じように TTC から受けとる BID、L1ID とデータの BID、L1ID をチェックする。

このあと、データは、イベント・フィルタで選択を受けた後、記録される。

なお、全検出器を統一的に制御・監視するために、DCS(Detector Control System) と呼ばれるシステムが用意される。この DCS は、各検出器の制御パラメータの設定および監視だけでなく、検出器全体に共通な下部構造をも監視する。

[3]なお、これらのカウンタは、TTC から送られてくる BCR(Bunch Counter Reset)、ECR(Event Counter Reset) によってリセットされる。最終的に ROD で BID は 12bit、L1ID は 24bit の値に変換されるが、この時点では、カウンタの bit 数はこれよりも少ないものでよい。

2.3 エンドキャップ・ミューオン・トリガー条件

ミューオン・トリガーは、ミューオンの運動量を測定し、この値が設定されたしきい値よりも大きい場合に出力されるものである。このしきい値は 6 種類用意されており、したがってトリガーの種類は 6 種類である。ATLAS エンドキャップ部のミューオン・トリーガーは、TGC を使って判断する。図 12 にエンド・キャップ部のミューオン検出器の配置を示す [10]。

図 12 中、M1 は TGC を 3 枚、M2、M3 は TGC を 2 枚重ねたものであり、それぞれトリプレット、ミドル・ダブレット、ピボット・ダブレットと呼ばれる。このほか、I も Inner チェンバーと呼ばれる TGC であるが、これはトリガー条件には使用されない。また、$\eta < 1.9$ の領域をエンドキャップ、$\eta > 1.9$ の領域をフォワードと呼んでいる。なお、検出の効率をあげるために、同じ層のチェンバーの境界では、ギャップを避け、隣あうチェンバーの端を重ねている。

ミューオン・トリガーは、そのトリガー条件によって大きく low-P_T トリガーと、high-P_T トリガーの 2 種類に分類できる。代表的なしきい値は、low-P_T トリガー、high-P_T トリガーに対してそれぞれ 6GeV、20GeV であるが、さらにそのしきい値によってそれぞれ 3 種類ずつの low-P_T トリガー、high-P_T トリガーが用意される。

まず、low-P_T トリガーの条件を説明する。図 13 に示すように、ビーム・衝突点で生成されたミューオンは、トロイド磁場によって飛跡が曲げられた後に TGC に入射する。飛跡がどれだけ曲がったかを測定することにより運動量を測定することができる。エンドキャップ・ミューオン・トリガー・システムでは多層の TGC の間でのヒット位置のずれを測定することで運動量の測定を行なう。low-P_T トリガーの場合、ピボット・ダブレットのヒット位置を基準としてミドル・ダブレットのヒット位置の R 方向のずれ δR、ϕ 方向のずれ $\delta\phi$ を測定する (図 13)。さらに、高い検出効率を保持しながらバックグラウンドを取り除くために、2 組のダブレット (4 枚の TGC) のうちの 3 枚以上にヒットがあるコインシデンスを要請する最終的に、δR と $\delta\phi$ から 3 種類の low-P_T トリガーをつくるロジック (セクター・ロジック) の概念図を図 14 に示す。

つぎに、high-P_T トリガーの条件は、low-P_T トリガー条件を満足したうえで、トリプレット中の 3 枚の TGC のうち 2 枚以上でコインシデンスがあり、運動量がしきい値より大きいことである。運動量の測定には、ピボット・ダブレットとトリプレットのヒット位置のずれを用いる。

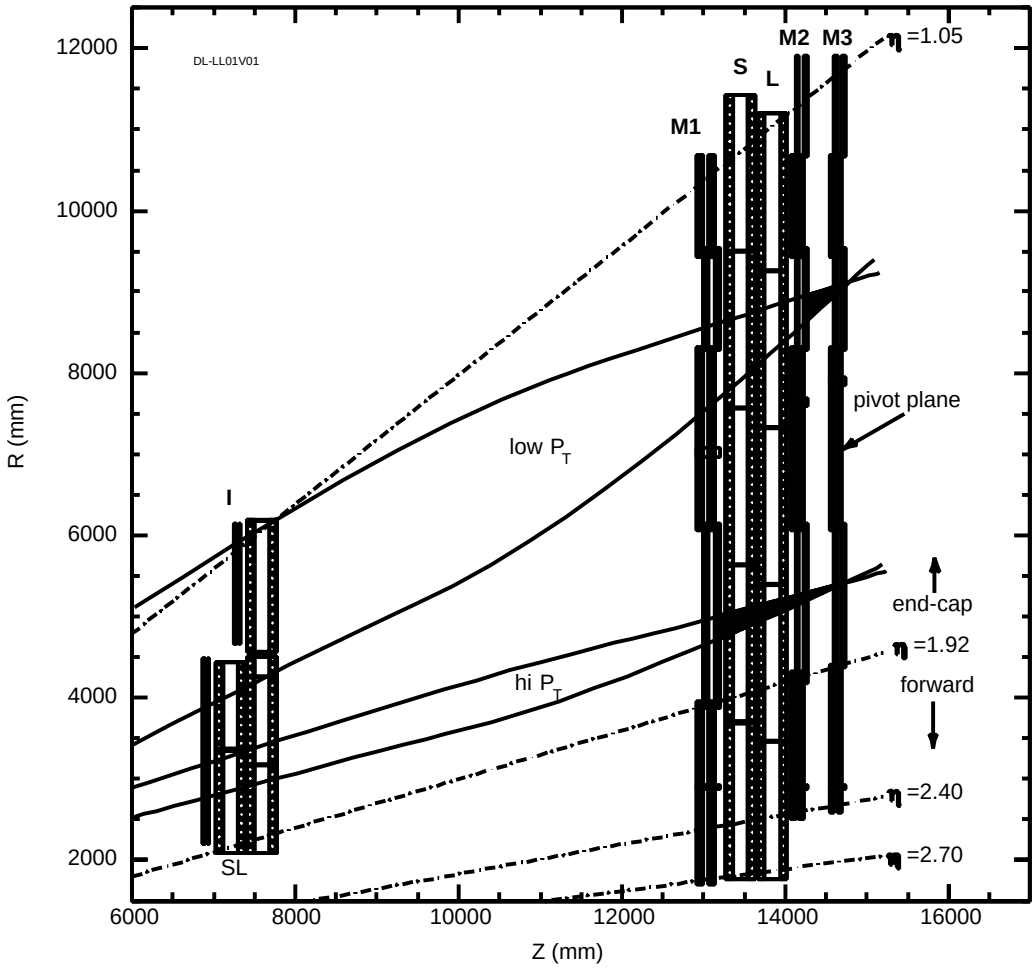

図 12: エンドキャップ部の TGC の配置およびミューオンの飛跡がトロイド磁場により曲がる様子が示されている。M1 は TGC 3 重層 (トリプレット)、M2 および M3 は TGC 2 重層 (ダブレット) であり、それぞれ、ミドル・ダブレット、ピボット・ダブレットと呼ばれる。これら 7 層の TGC でトリガーの判断をおこなう。I は内部チェンバー (TGC 2 重層) とよばれ、トリガーの判断には使われない。なお、S、L は MDT である。

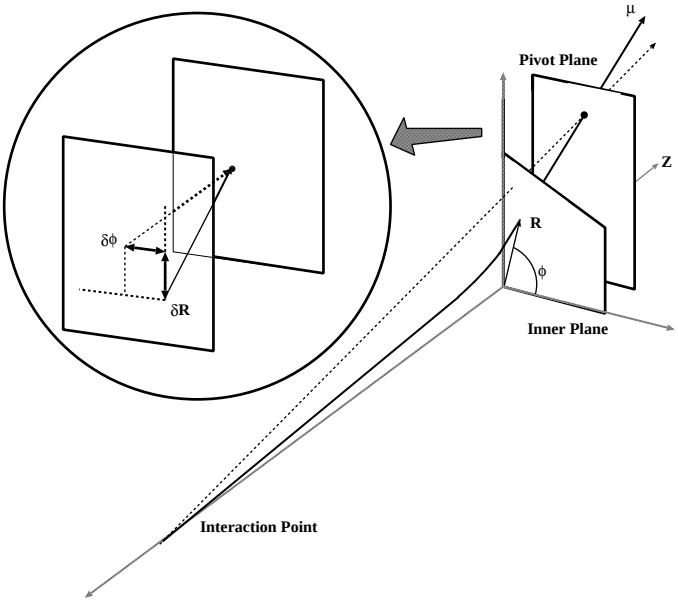

図 13: TGC によるミューオン運動量の測定。無限大の運動量をもつミューオンは、破線のような飛跡を描くが、実際の有限の運動量をもつミューオンは、実線のように、トロイド磁場によって飛跡が曲げられた後に TGC に入射してくる。ピボット・ダブレットのヒット位置を基準としてミドル・ダブレットのヒット位置のずれを測定する。なお、R 座標、ϕ 座標はそれぞれ wire、strip のヒット・チャンネルである。

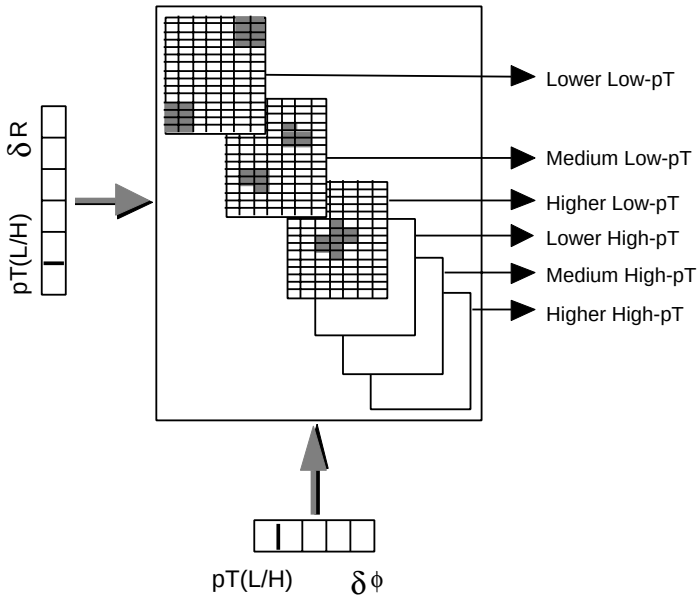

図 14: セクター・ロジック。2 つの TGC ステーションでのヒット位置のずれからトリガー low-P_T、high-P_T それぞれ 3 種類のトリガーをつくる。

2.4 TGC のデータ読み出しの概要

図 15 に、デランダマイザまでの TGC の読み出し系の概要を示す [10]。

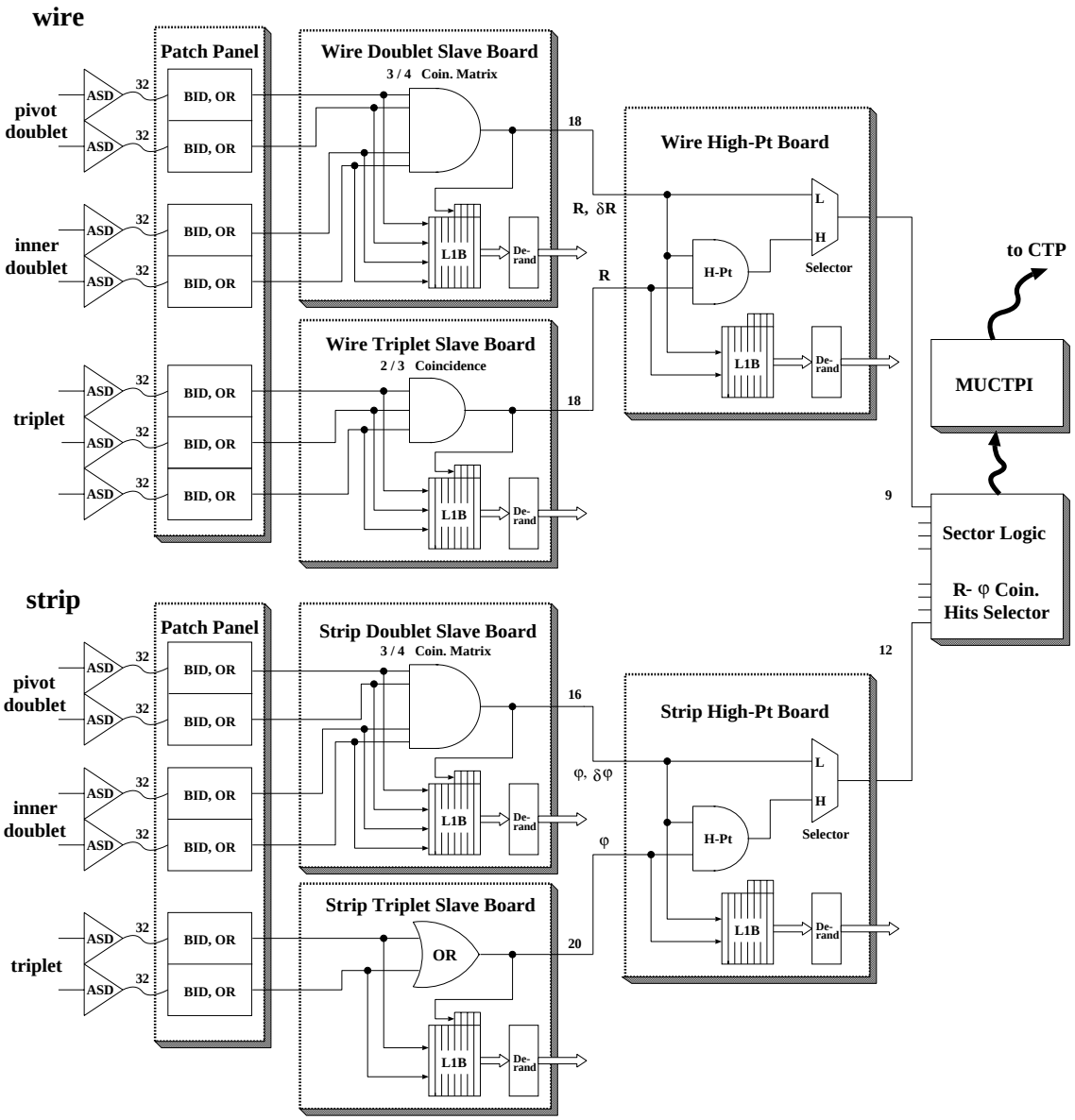

図 15: デランドマイザまでの TGC 読み出し系の概要。

TGCからからの出力信号は、まず、ASD(Amplifier Shaper Discriminator) ボードという TGC に直接搭載されるボードで LVDS(Low Voltage Difference Signal) レベルのデジタル信号に変換される。

ASD ボードの出力信号は、パッチ・パネル (Patch Panel) と呼ばれるボードに送られる。このパッチ・パネルは、LVDS レシーバを搭載しており、これを通して ASD ボードからの信号を受信し、この入力信号をバンチ・クロックに同期させる。その後、隣あうチェンバーの端でのダブル・カウントの処理を行なう。

このほか、また、パッチ・パネルには、DCS へのインターフェイス、ASD、スレーブ・ボードの電源なども搭載される。

パッチ・パネルの出力は、スレーブ・ボード (Slave Board) に送られる。スレーブ・ボードは、その扱うデータがトリプレットのものかダブレットのものか、また、ワイヤーのものかストリップのものかで 4 種類あり、それぞれ入出力のチャンネル数が異なる。

ダブレット・スレーブ・ボードでは、2 組のダブレットの 4 枚の TGC でコインシデンスをとりピボット・ダブレットでのヒット位置と δR、$\delta\phi$ を読み出す。図 16 に、ダブレット読み出し用のコインシデンス・ロジックを示す [10]。

トリプレット・スレーブ・ボードでは、トリプレットのの 3 枚の TGC でコインシデンスをとり、ヒット位置を読み出す。図 17 に、トリプレット・ワイヤー読み出し用のコインシデンス・ロジックを示す [10]。

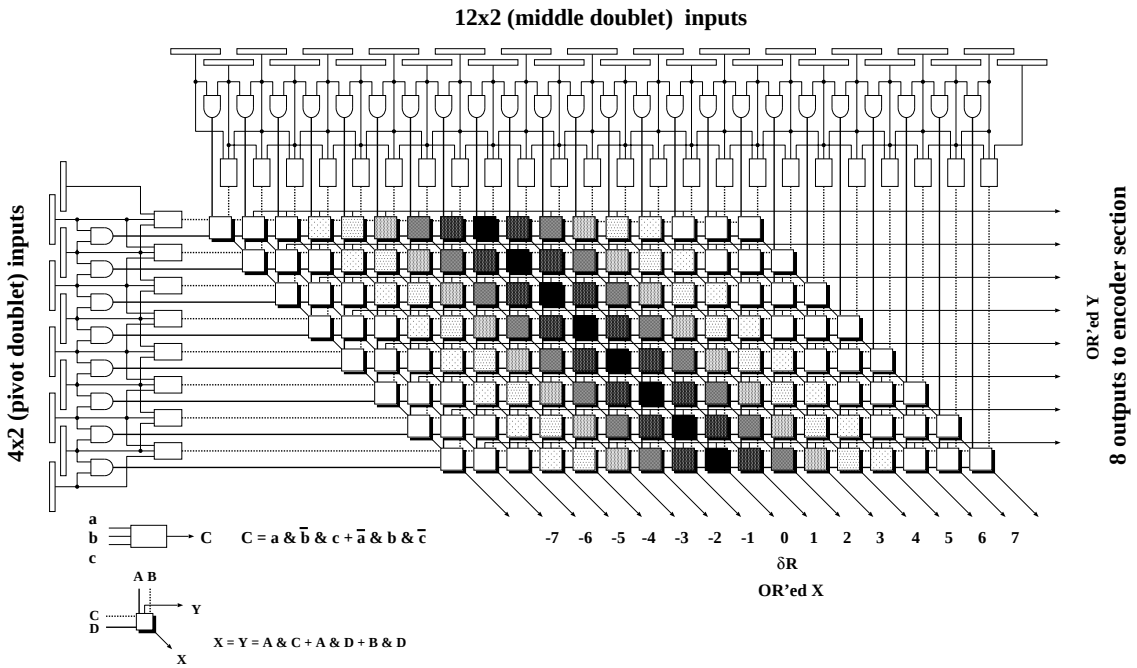

図 16: ダブレット・ワイヤー・スレーブ・ボードの Low P_T コインシデンス・マトリクス。図中、各セルが 3-out-of-4 のコインシデンスをとっている。ピボット・ダブレットのヒット位置とミドル・ダブレットのヒット位置がずれていなければ、図中、$\delta R = 0$ の対角線上にあるセルでコインシデンスが起こり、ヒット位置が 1 チャンネルずれた場合は、2 番目に濃く塗りつぶされたセルでコインシデンスが起こる。同じピボット・ダブレットでのヒット位置、あるいは、同じ δR のものは、OR をとり出力する。ストリップ用のものも同様の構成だが、読み出し $\delta\phi$ の読み出しは、-3 から +3 である。

スレーブ・ボードの出力信号は、high-P_T・ボードに送られるが、ここでは、ダブレット・スレーブ・ボードのコインシデンス・マトリクスと同様のマトリクス (図 18) を通してピボット・ダブレットとトリプレットのあいだの δR、$\delta\phi$ ををを測定する。

high-P_T・ボードのトリガー・ロジックの出力は、セクター・ロジックに送られる。ここでワイヤーとストリップのデータが合成され、ミューオン・トリガーとしてレベル 1・トリガーのプロ

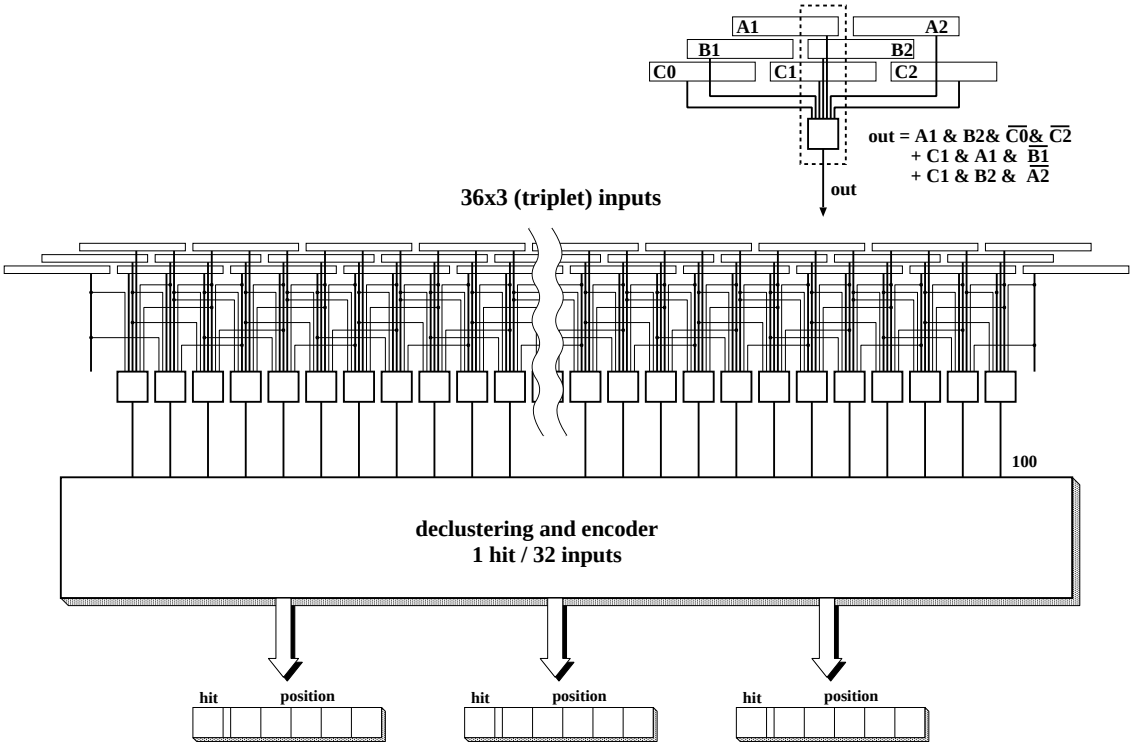

図 17: トリプレット・ワイヤー・スレーブ・ボードのコインシデンス・マトリクス。図中、各セルが 2-out-of-3 のコインシデンスをとっている。ストリップ用のものも、1-out-of-2 コインシデンスであることを除けば同様の構成である。ヒット位置のみを出力する。

セッサである CTP に送られる。

一方、スレーブ・ボードと high-P_T・ボードには、レベル 2 以降のイベント選択に送るためのレベル 1・バッファとデランドマイザも実装され、BID、L1ID、コインシデンス・ロジックへの入力 (各 TGC のヒット・パターン) と出力のデータ (δR、$\delta\phi$) がここから読み出される。

TGC 読み出しのスキームにおいて、TGC は、ローカル・DAQ・ブロック (LDB) と呼ばれるブロックにグループ分けされる。TGC の読み出しの方針は、スレーブ・ボードから ROB に至るまで、このローカル・DAQ・ブロックごとに、独立に読み出しを行なうというものである。ローカル・DAQ・ブロックは、トリプレット、2 組のダブレット、内部チェンバーの 3 つの読み出しレイヤーのうち 2 つのレイヤーにまたがらないように決められている。

図 19 に、オクタント[4]あたり、読み出しレイヤーごとの TGC のローカル・DAQ・ブロックへの分割の仕方、およびスレーブ・ボード以降の読み出し系の概要を示す [10]。

スレーブ・ボード、high-P_T ボードのデランドマイザに格納されたデータは、ローカル・DAQ・ブロックごとに 1 つずつ用意されるスター・スイッチと呼ばれる、データの中継点に集積される。このデータ転送は、LVDS レベルのリンクで行なう。なお、このリンクを LS-リンク (Local Slave link) と呼ぶ。

[4]TGC の各円盤を 8 分割したものをオクタントと呼ぶ。

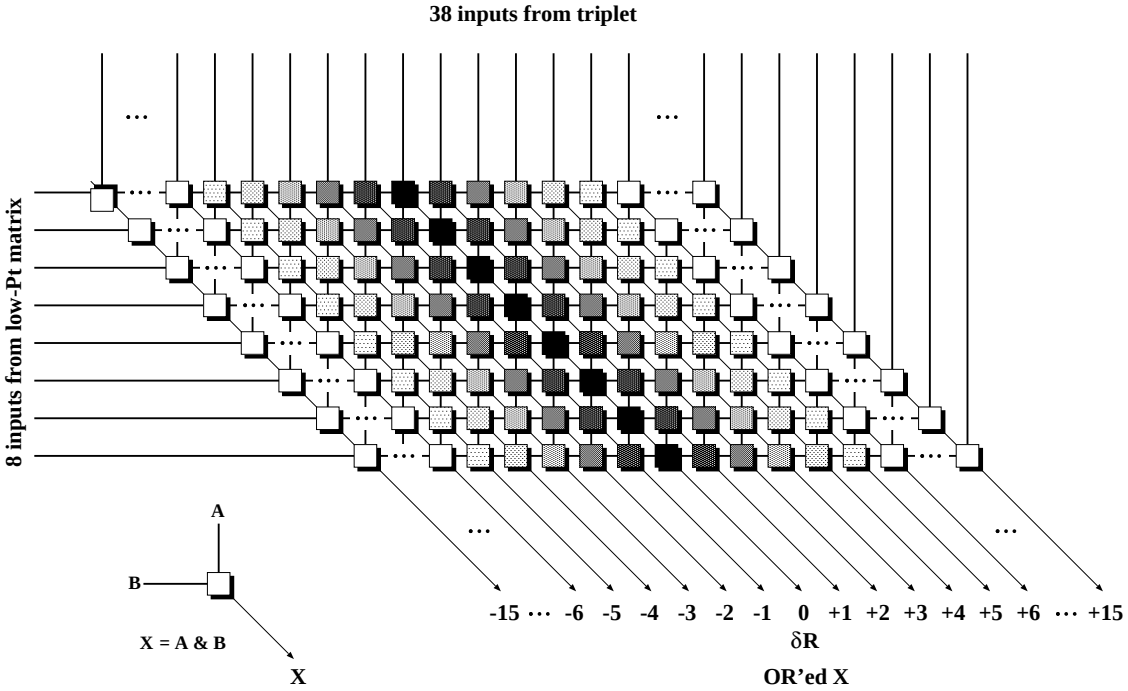

図 18: high-P_T・ボードののコインシデンス・マトリクス。各セルは、図中、各セルが 2-fold のコインシデンスをとっている。ストリップ用のものも、1-out-of-2 コインシデンスであることを除けば同様の構成である。

スター・スイッチに集積されたデータは、光ファイバーで ROB と同じクレートにあるローカル・DAQ・マスターに送られる。

ローカル・DAQ・マスターは、スター・スイッチからのデータ転送の制御、BID、L1ID によるエラー検出、データ形式の ROB に送るデータ形式に変換を行なう。

この後は、データは、ROD、RO-link を経て ROB に送られ、レベル 2、イベント・フィルターのデータ選択を受ける。

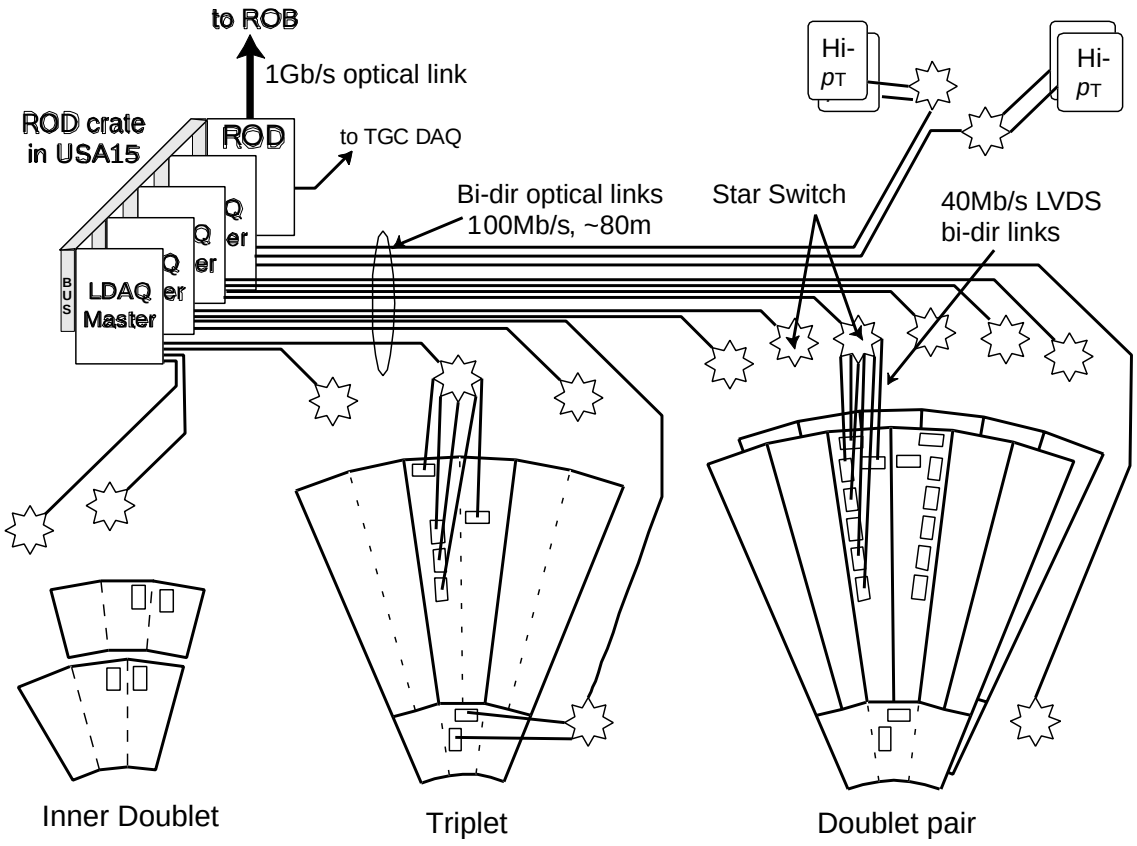

図 19: スレーブ・ボード以降の TGC 読み出し系の概要。各読み出しレイヤーのローカル・DAQ・ブロックが実線で示されている。

3 TGC 読み出しシステムの設計

3.1 データ集積方法の選択

TGC の読み出しには、一度スター・スイッチにデータを集積する構造をとるが、本節では、このスター構造がほかの構造よりも TGC の読み出しに適している理由を議論する。

TGC の読み出しの場合は、スレーブ・ボードは TGC 上にじかに配置されるので、一つのローカル・DAQ・ブロックが扱うスレーブが 10m 程度の範囲に配置されている。また、スレーブとマスターの距離が、約 80m であるため、1 つ 1 つのスレーブを直接ローカル・DAQ・マスターに接続することは、ケーブルの量からいって困難である。

そこで、各スレーブ・ボードからローカル・DAQ・マスターにデータを送信する方法として、スター・スイッチ構造のほかに、バス構造、リング構造が挙げられる。これらの特徴を以下に挙げる。

バス構造 図 20 にバス構造の概念図を示す。データ線とは別にアドレス線を用意し、マスターからスレーブのアドレスを特定することでマスターと目的のスレーブの接続を行なう。すなわち、各スレーブにアドレス・デコーダが必要となる。データ線を複数用意し、1 つのデータ線に 1 つの機能 (いまの場合は、1 種類のデータ) を割り付けるので、デコードが非常に簡単である。一方、一つのスレーブが故障した場合、その故障の症状によっては全体が正しく機能しなくなってしまう。また、マスターとスレーブの距離が長い場合には、通信は困難である。

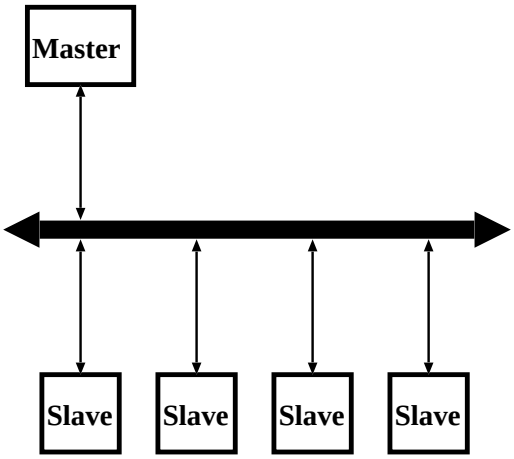

図 20: バス構造の概念図。

リング構造 図 21 にリング構造の概念図を示す。データは、パケットにして送信する。また、マスターからは、データとともにスレーブ・アドレスを送信する。各スレーブにはアドレス・デコーダが用意される。したがって、デコードのロジックは複雑になる。データ・パケットは、指定されたアドレスのスレーブに到達するまでスレーブからスレーブへと転送される。したがって、マスターとスレーブの距離が長くても問題はないが、故障したスレーブが存在すると、全体が正しく機能しなくなってしまう。

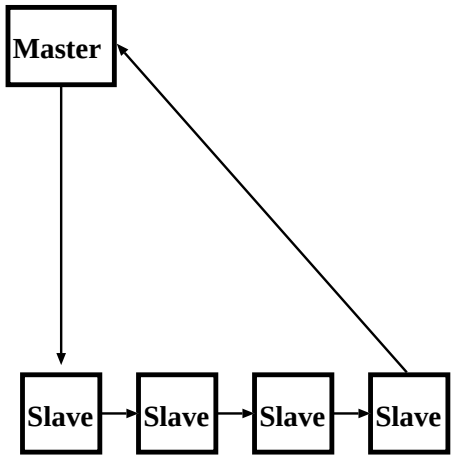

図 21: リング構造の概念図。

スター構造 図 22 にスター構造の概念図を示す。スイッチを中継してスレーブからマスターにデータを送る。スイッチは、各スレーブと直接通信するので、アドレス・デコーダは、スイッチにだけ用意すればよく、各スレーブに用意する必要はないので、スレーブのロジックは簡単になる。また、故障したスレーブが存在しても、ほかのスレーブとの通信に支障をきたさない。

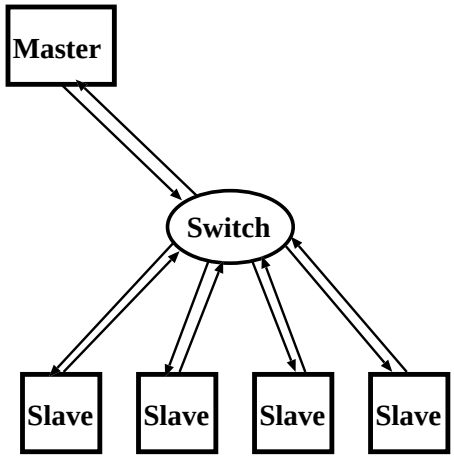

図 22: スター構造の概念図。

TGC 読み出しの場合には、スレーブが修理のために簡単にアクセスすることができないカウンター・ホール内に配置されるので、1 つのリンクの故障がほかのスレーブの読み出しにまで影響する構造は適当でなく、スター構造が適している。また、3.2 節に述べるが、TGC のデータ量は少ないので、データ圧縮を行ない、シリアル通信でデータ転送を行なうことが可能であり、ケーブルの量を効率的に節約することができる。

3.2 データの特徴

各ローカル・DAQ・ブロックには、読み出しレイヤーごとにスター・スイッチが 1 つずつ用意される。ワイヤー、ストリップともすべてのデータは、このスター・スイッチを中継して読み出される。

表 3 に、各種ローカル・DAQ・ブロックの特徴を示す

LDB の種類		オクタントあたりの LDB 数	LDB あたりのスレーブ・ボードの数	LDB あたりの読み出しチャンネル数
ダブレット	フォワード	1	15	1902
	エンドキャップ	6	15	1854
トリプレット	フォワード	1	15	1200
	エンドキャップ	3	18	1724
high-P_T	フォワード	1	6	
	エンドキャップ	1	24	

表 3: ローカル・DAQ・ブロックに含まれるスレーブ・ボードの数。TGC の 2 枚の円盤全体でたしあわせるとスレーブ・ボード、High-P_T・ボードの総数はそれぞれ 2784 枚、480 枚にのぼる。また、2 枚の円盤の読み出し総チャンネル数は、326496 チャンネルである。

スター・スイッチが読み出すデータは、ボードの種類ごとに以下のようになっている。

ダブレット・スレーブ・ボード コインシデンス・マトリクスへの入力の 4 枚の TGC のヒット・パターン。TGC1 枚あたり 32 チャンネルである。

トリプレット・スレーブ・ボード コインシデンス・マトリクスへの入力の 3 枚の TGC のヒット・パターン。TGC1 枚あたり 32 チャンネルである。

high-P_T・ボード high-P_T・ボードは、4 枚のダブレット・スレーブ・ボードと 3 枚のトリプレット・スレーブ・ボードからデータを受けとるが、これらのデータを読み出す。ダブレット・スレーブ・ボード、トリプレット・スレーブ・ボードは、データを送る際にデータの圧縮を行なう[5] が、high-P_T・ボードの側でデコードするまえのものを読み出す。

表 4 に、スレーブからスター・スイッチへ読み出す読み出しチャンネル数をまとめる。

ミューオンによる信号は、レベル 1・トリガーと相関があるが、イベントあたり、ローカル・DAQ・ブロックあたりのデータ量としては少ない。このことは 3.7 節で議論する。このため、ここでは、読み出しデータとしてバックグラウンドによるヒットのみを議論する。

TGC のバックグラウンド・レート (TGC の円盤全体で平均した値) は、シミュレーションの結果により以下のように見積もられている [10]。

a. TGC を 1 枚だけヒットするバックグラウンド。TGC の中で起きるコンプトン効果で生成される 2MeV 以下の電子や中性子と TGC との相互作用で生成される低エネルギーの陽子によるバックグラウンドで、4.5Hz/cm^2 のカウント・レートである。

[5] ダブレット・ワイヤー・スレーブ・ボードでは、1 つのトラックに対して位置を 5 ビットで、δR は、4 ビットでエンコードする。1 スレーブボードのなかではトリガーは 2 つまでしか出さないことになっているので、ダブレット・スレーブ・ボードからの送信は、イベントあたり 18 ビットになる。

スレーブの種類		スレーブあたり
ダブレット	ワイヤー	4×32
	ストリップ	4×32
トリプレット	ワイヤー	3×32
	ストリップ	2×32
high-P_T	ワイヤー	$4 \times 18 + 3 \times 18$
	ストリップ	$4 \times 16 + 3 \times 20$

表 4: 読み出しのチャンネル数。

b. 隣合う 2 枚の TGC をヒットするバックグラウンド。コンプトン効果で生成される 2MeV 以上の電子や低エネルギーの陽子によるバックグラウンドで、3.1Hz/cm^2 のカウント・レートである。

c. 3 枚の TGC をヒットする可能性のあるバックグラウンド。高エネルギーの陽子、パイオン、$\sim$ 100MeV のミューオンで、カウント・レートは、3.0Hz/cm^2 である。

これらとローカル・DAQ・ブロックの TGC の全面積の積をとり、ローカル・DAQ・ブロックあたりのバックグラウンド・ヒット・レートを計算すると、表 5 のようになる。

バックグラウンドの種類		トリプレット		ダブレット	
		フォワード	エンドキャップ	フォワード	エンドキャップ
a	ヒット・レート (kHz)	750	2000	970	1700
	hits/event	0.019	0.051	0.024	0.042
b	ヒット・レート (kHz)	340	940	330	570
	hits/event	0.0086	0.023	0.0084	0.014
c	ヒット・レート (kHz)	160	450	160	280
	hits/event	0.0041	0.011	0.0040	0.0069
計	ヒット・レート (MHz)	1.3	3.4	1.5	2.5
	hits/event	0.031	0.086	0.037	0.063
圧縮前のデータ量 (MBytes/s)		18	23	24	24

表 5: ローカル・DAQ・ブロックあたりのバックグラウンド・ヒット・レート。なお、hits/event は、レベル 1・トリガーのイベントに対する平均のヒット数である。

表 5 の hits/event の合計に見るように、レベル 1・トリガーのバンチにヒットがある確率は、どのローカル・DAQ・ブロックにおいても 10% 以内と小さい。したがって、データを圧縮し、シリアル通信でスター・スイッチに送信することが可能であると考えられる。

3.3 データの形式

最も簡単なデータの圧縮方法は、表 4 に示したスレーブ内の各読み出しチャンネルに対してひとつずつアドレスを割り当て、ヒットがあったチャンネルのアドレスを送信する方法である。ただし、この方法は、ひとつのスレーブ内のチャンネル同士に相関があるとデータ量が増えてしまううえ、回路の構造が複雑になる。

そこで、TGC での読み出しでは、読み出しチャンネルを 8 チャンネルごとのセルにまとめて以下のようにデータを圧縮 (ゼロ・サプレス) する。1 チャンネルもヒットがないセルのデータは、読み出さず、ヒットがあったセルに関してのみセルのアドレス (8 ビット) とセルのヒット・パターン (8 ビット) を読み出す。

実際の送信では、さらにスレーブ・ボードのアドレス (8 ビット)、BID(8 ビット) と L1ID(8 ビット) をヘッダとして、ターミネータ (8 ビット) をフッタとして付け加えたデータ形式で、シリアル転送する。ヘッダとフッタを付け加えた送信データ形式を図 23 に示す。ATLAS 実験のデータ読み出しでは、キャリレーションのためのオプションとして 1 つレベル 1・トリガーに対して、3 バンチ・クロッシング分のデータまで読み出せる設計にする。図 24 に 3 バンチ・クロッシング分のデータを読み出す際のデータ形式を示す。

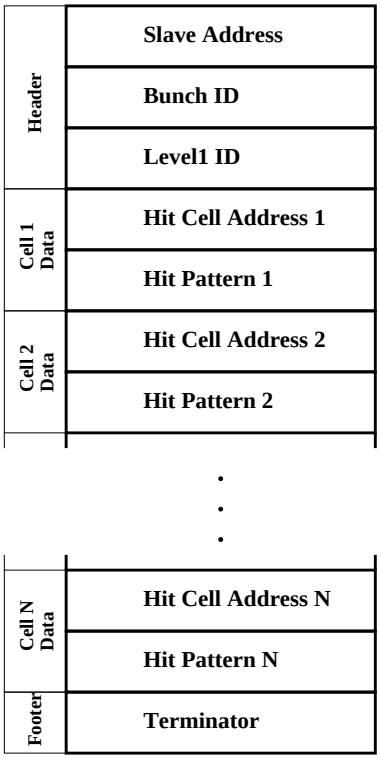

図 23: n 個のセルにヒットがあった場合のスレーブからスター・スイッチへの転送データの形式。1 段が 1 バイトであり、ヒットがあるセル数を n とすれば、データ長さは、$8(2n+4)$ である。

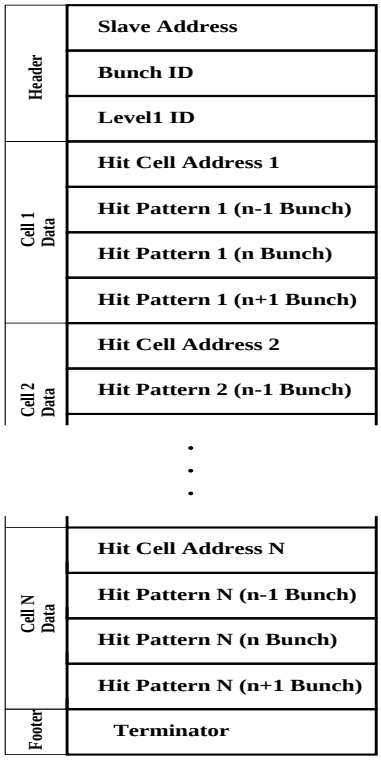

図 24: レベル 1・トリガーの前後のバンチ・クロッシングのデータも読み出す場合のスレーブからスター・スイッチへのデータ転送データの形式。

つぎに、このデータ形式が誤認識される可能性を議論する。この 8 ビットを単位とした形式は、ROD でのデコード まで保持されるので、結局、ターミネータを誤認識する可能性を考えればよい。ターミネータは全 8 ビットとも High のパターンとする 。まず、スレーブ・アドレスから L1ID までの 3 バイトは確実にヘッダなので、前のイベントのターミネータが正しく認識されれば、誤認識される可能性はない。1 つのレベル 1・トリガーに対して読み出すバンチの数は ROD にも制御パラメータとして与えられているので、各セルからのデータの長さは正しく認識される。したがって、誤ってターミネータと認識される可能性のあるものは、セル・アドレスだけである。表 4 を

みると、読み出しチャンネルの多いスレーブはダブレット・スレーブボードだが、8ビットごとにまとめると16セルであり、下位4ビットでアドレスを表現できる。したがって、残っている上位4ビットをすべてLowに保てば、ターミネータとの誤認識を避けることができる。

このデータ圧縮の後でのバックグラウンド・ヒットによるデータ量を表6に示す[6]。

バックグラウンドの種類		トリプレット		ダブレット	
		フォワード	エンドキャップ	フォワード	エンドキャップ
a	hits/event	0.019	0.051	0.024	0.042
	bits/hit	48 or 96	48 or 96	96	96
b	hits/event	0.0086	0.023	0.0084	0.014
	bits/hit	144	144	192	192
c	hits/event	0.0041	0.011	0.0040	0.0069
	bits/hit	240	240	384	384
計	bits/event	3.7	10	5.5	9.4
	kBytes/s	46	130	69	120
圧縮前のデータ量	MBytes/s	18	23	24	24

表6: データ圧縮後のローカル・DAQ・ブロックあたりのデータ量。ただし、kBytes/sは100kHz/sのレベル1・トリガー・レートでの値である。比較のために圧縮前のデータ量が示されている。なお、トリプレットでのバックグラウンドaによるヒットのイベントあたりのデータ量が2通り示されているのはトリプレットの3枚のTGCのうち真中のTGCはストリップ読み出しがないため、データ量が他の2枚とは異なるからである。

3.4 LS-リンクの通信プロトコル

スター・スイッチとスレーブは、LVDSレベルのリンクでパケット通信を行なう。スター・スイッチからスレーブへの送信用とスレーブからスター・スイッチへの送信用のリンクが用意され、それぞれが、CLOCK、DATA、SYNCの3信号でなっている。CLOCKは、パケット読みとりのためのクロックである。DATAは、送信データであると同時に、SYNCとともに通信を制御する。図25に示すように、SYNC= 1&DATA= 1でパケットの始まりを、SYNC= 1&DATA= 0でパケットの終りを指定し、この間に3.3節で述べたデータやスター・スイッチからスレーブへのコマンドなどを挟んだ構造のパケットを転送する。このLS-リンクは、主には、データの読み出しをおこなうが、このほかにも、スレーブへのパラメータのダウンロードなどの目的にも使用する。

[6]ただし、データ量は多めに見積もっている。例えば、バックグラウンド b. では、2枚のTGCにヒットがあるが、ヒット位置によってデータ量が異なる。

1). 2枚のTGCでのワイヤー・ヒット・チャンネルが同じスレーブ・ボードで読み出される場合、ワイヤーの読み出しデータ量は、56ビットである。。

2). 2枚のTGCでのワイヤー・ヒット・チャンネルが別々のスレーブ・ボードで読み出される場合、ワイヤーの読み出しデータ量は、2×48ビットである。。

この場合、2) の方がデータ量が多い。表6の計算では、2) を使って計算している。ストリップに関しても同様である。

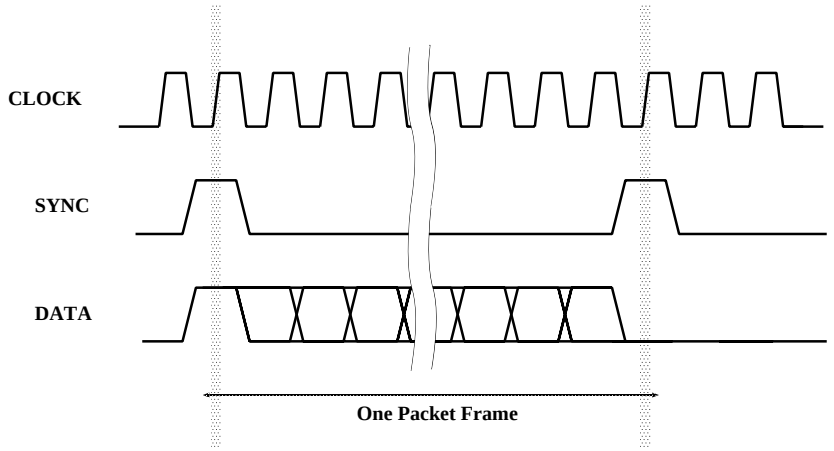

図 25: LS-リンクの 3 種類の信号線。SYNC と DATA の組み合わせでパケットの開始と終了を指定する。

3.5 スレーブ・ボード

図 26 にスレーブ・ボードのデータ読み出し部分のブロック図を示す。まず、データは、レベル 1・バッファを通して、デランドマイザに格納される。その後、シリアル変換を受けて、スター・スイッチに送信される。

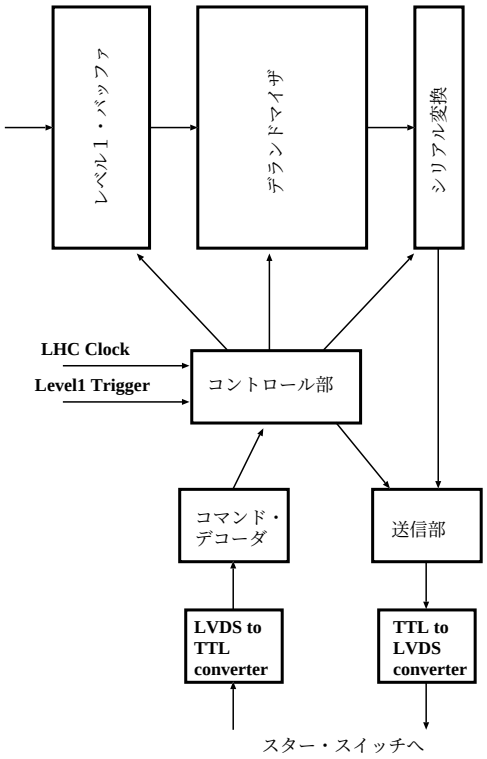

図 26: スレーブ・ボードのデータ読み出し部分。

図 27 に読み出しセル 1 つ分のレベル 1・バッファからデランドマイザまでのブロック図を示す。

レベル 1・トリガーに対応するバンチとその前後の合計 3 バンチまで読み出しできるよう設定可能にする必要があるが、各バンチの読み出し制御は図 27 中の Bunch readout control 信号で行なう。

レベル 1・バッファは、イベント発生から 2.5μs の間、データを保持しておくパイプラインであり、40MHz で動作する 100 段のシフト・レジスタで実現する。ただし、各レベル 1・トリガーに対してその前後のバンチに対応するデータまで読み出せるようにしている。

デランドマイザは、レベル 1・トリガーのタイミングで TGC からのヒット・パターンをレベル 1・バッファから入力し、これをスター・スイッチからのデータ送信コマンドのタイミングで読み出すことを可能にするバッファである。また、デランドマイザではシリアル変換時の圧縮のために読み出しセルごとにデータを扱い、各セルでのヒットの有無を判断する。まず、バッファの部分に関しては、書き込みと読み出しのタイミングが独立であるので、FIFO にすることが適当である。この FIFO への入力には、TGC からのヒット・パターンとともに、各セルの読み出しを制御する信号を用意する。FIFO の前の段階でセル内の全チャンネルで論理和をとり、データの圧縮の際にこのセルを読み出すかどうかを判断している (図 27 中 Has Data Logic)。この Has Data Logic を

図 28 に示す。Has Data Logic では、セル内の読み出すべきバンチに 1 箇所でもヒットがあれば、Has Data 信号を High にする。さらに、FIFO の後の段階でこの Has Data と各バンチの Bunch readout control 信号を合成し、このセル内の各バンチの読み出し制御に用いる。また、各セルに対して、Has Data 信号を読み出し制御信号とするセル・アドレスが用意されている。

デランドマイザの出力は、シリアル変換した後にスター・スイッチに送信する。このシリアル変換の際にデータの圧縮を行なう。デランドマイザの出力は、8 ビットのデータとこのデータの読み出し制御の信号の組み合せの繰り返しである。そこで、データの圧縮の方針としては、読み出し制御信号が High になっていればそのデータを読み出し、読み出し制御信号が Low になっていればそのデータの読み出しをスキップする。また、スター・スイッチに対して送信する信号は、DATA のほかに SYNC も用意する必要があるが、この SYNC 信号もシリアル変換の際に用意する。

図 29 にデランドマイザの出力をシリアル変換する部分のブロック図を示す。図 29 中、左のシフト・レジスタの列は、スレーブからスター・スイッチに送信する DATA を用意する。デランドマイザから出力される 8 ビットのデータは、スター・スイッチからのデータ送信コマンドのタイミングでシフトレジスタにロードされ、そのあと、40MHz のクロックで図の下の方向に読み出される。シフトレジスタには、セルの区切りごとに読み出し制御信号によって制御されるセレクタがあり、このセレクタにより、読み出す必要のないセルはスキップされる。スレーブのアドレス、BID、L1ID、ターミネータに関しては、スレーブ全体でデータがある場合に読み出すべきものなので、これらの制御信号としては、全セルの Has Data の論理和をとったものを用いる。また、パケットのヘッダの次のビットには、スター・スイッチが読み出していないイベント・データがデランドマイザの FIFO に残っているかどうかの情報として FIFO のエンプティー・フラグをロードする。さらにその次のビットには、ROD でデランドマイザが溢れているかどうかをモニターするために、FIFO のフル・フラグをロードする。

一方、図 29 中の右のシフト・レジスタの列は、スター・スイッチに送信する SYNC を用意する。SYNC のシフト・レジスタには、DATA のシフトレジスタと同時に、パケットのヘッダとフッタを除いてすべてのシフトレジスタに Low をロードする。DATA を用意するシフト・レジスタと同じ構造のシフトレジスタを用意し、またデータのスキップを制御する信号を共有することで、DATA 信号の長さにあわせた SYNC 信号をつくり出している。

このようにしてスレーブからスター・スイッチに送信されるパケットを図 30 に示す。

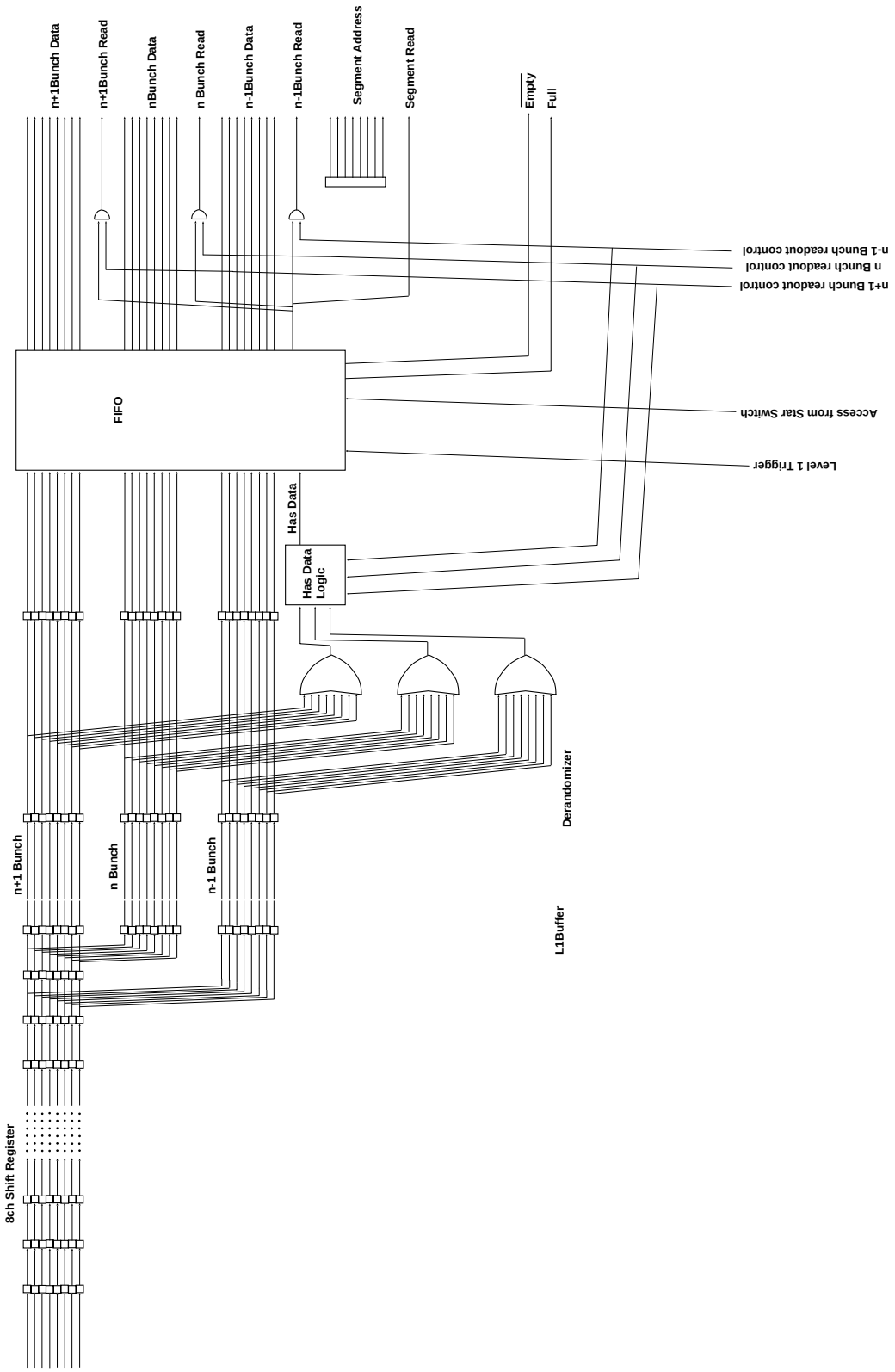

図 27: 読み出しセルあたりのレベル 1・バッファからデランドマイザのブロック図。小正方形ひとつひとつが 40MHz で動作するフリップ・フロップである。

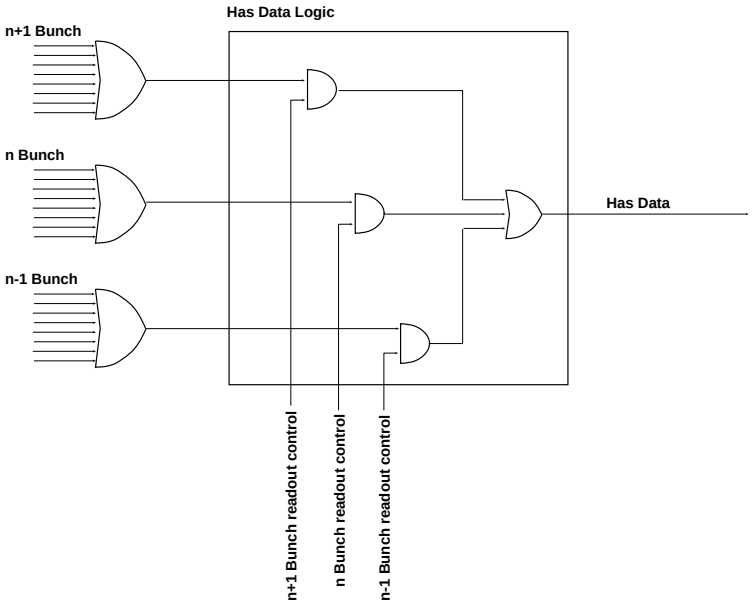

図 28: Has Data Logic。

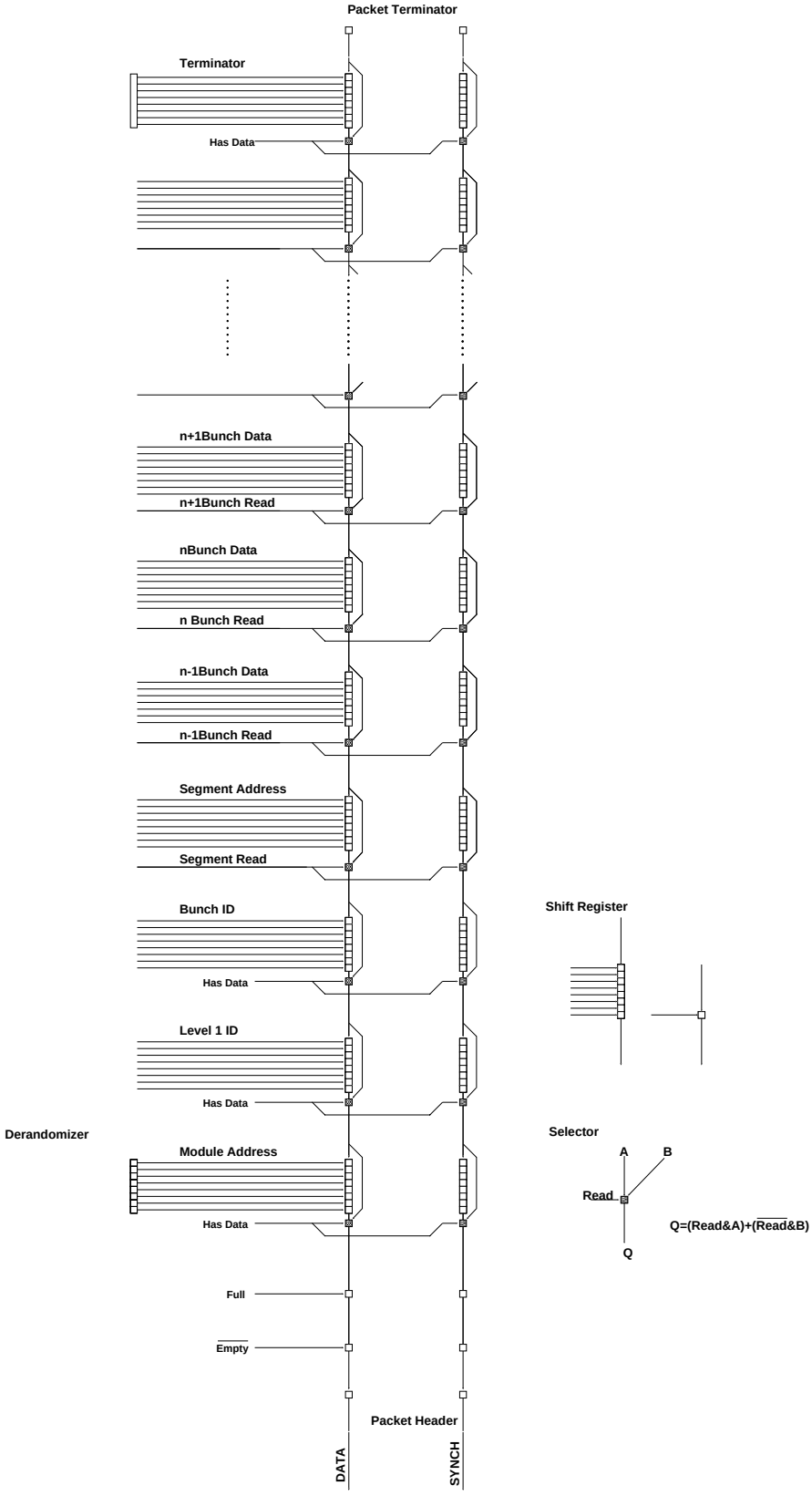

図 29: デランドマイザの出力をシリアル変換する部分のブロック図。

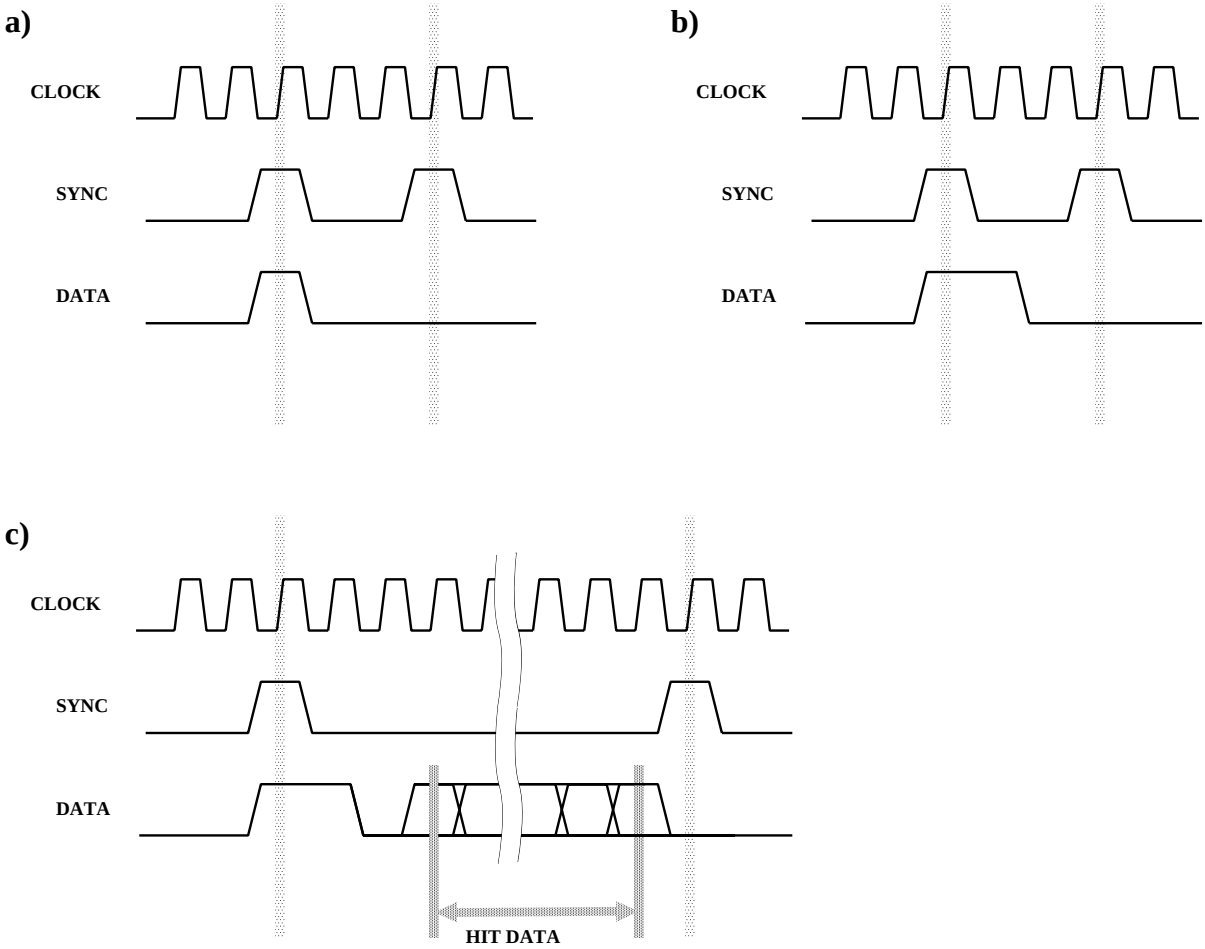

図 30: スレーブからスター・スイッチに送信されるパケット。a) デランドマイザが空の場合。b) デランドマイザは空ではないが、トリガーに対応するバンチ・クロッシングに対してスレーブ内のどのセルにもヒットがない場合。c) トリガーに対応するバンチ・クロッシングに対してスレーブ内にヒットがある場合。データの 1 ビット目にデランドマイザが空かどうか、2 ビット目にデランドマイザが溢れているかどうかの情報を送信する。

3.6 スター・スイッチ

図 31 にスター・スイッチのブロック図を示す。スター・スイッチは、最大 32 のスレーブ・ボードまたは High-P_T・ボードからのデータを扱うことのできる構造であり、マルチプレクサで通信するスレーブをつなぎかえる。スレーブから読み出したデータは、スター・スイッチ内の FIFO に格納される。この後、データには、ヘッダとフッタが付加され、光ファイバーのリンクでローカル・DAQ・マスターに送信される。ローカル・DAQ・マスターからの光ファイバー・リンクは、スター・スイッチからローカル・DAQ・マスターへのデータ送信の制御やスレーブ、スター・スイッチへのパラメータのダウンロードなどに使用される。

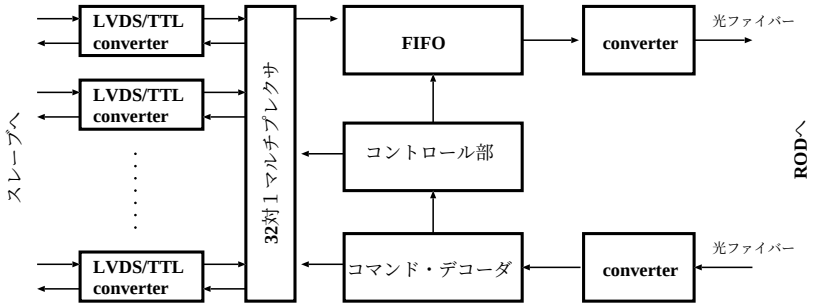

図 31: スター・スイッチのブロック図。

たとえば、100kHz のレベル 1・トリガー・レートでは、スター・スイッチは平均 10μs 以内にローカル・DAQ・ブロック内の全スレーブの読み出しを行なわなくてはならない。この読み出しの制御は、$\sim$ 80m も離れたローカル・DAQ・マスターから行なうのは困難であり、したがって読み出しのシークエンスを制御する回路はスター・スイッチに搭載する。

図 33 にスター・スイッチの読み出しシークエンスを示す。読み出すべきスレーブのアドレスの一覧を保持するためのレジスタがスター・スイッチ内に用意される。故障しているスレーブはこの一覧から取り除かれる。スター・スイッチ内のシークエンサは、まず、アドレス一覧のなかの一番はじめのスレーブにマルチプレクサを接続する。このスレーブにデータ送信コマンドを繰り返し送信し、デランドマイザにデータが溜る (デランドマイザの FIFO が空でなくなる) まで待つ。はじめのスレーブのデランドマイザにデータが溜ったらデータを読み出し、次のスレーブに接続し、データの読み出しを行なう。このつなぎかえ、読み出しの操作を繰り返し、最後のスレーブの読み出しが終ったら、再びはじめのスレーブに接続し、データが溜るのを待つ。

なお、スレーブへのデータ送信コマンドの長さは、イベントあたりのデータ読み出しの所要時間に直接影響するので、できるかぎり短くする必要がある。データ送信コマンドを図 34 に示す。

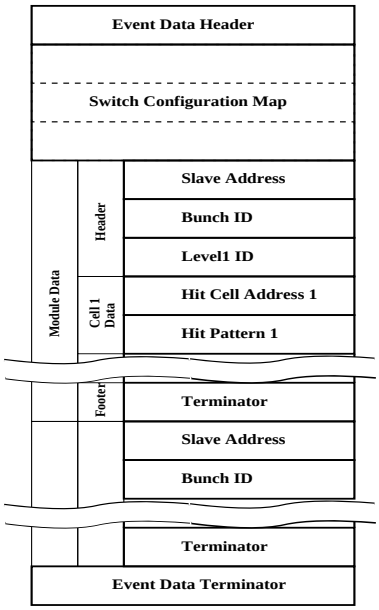

図 32: スター・スイッチからローカル・DAQ・マスターに送信されるデータの形式。

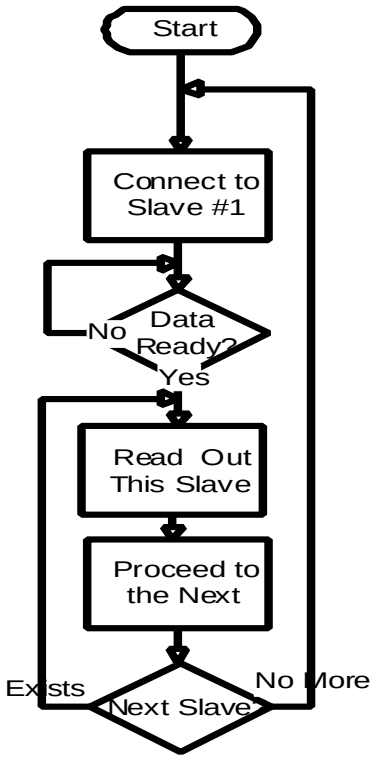

図 33: スター・スイッチの読み出しシークエンス。

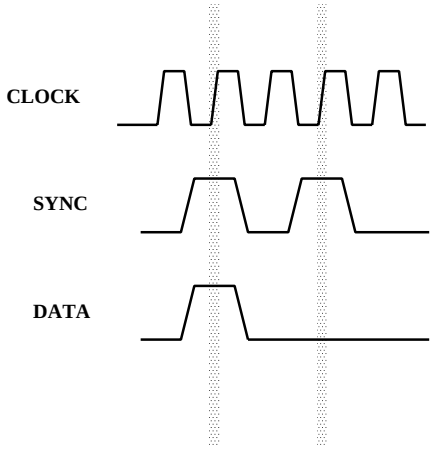

図 34: スレーブへのデータ送信コマンド。

3.7 デランドマイザのサイズ

ATLAS 検出器では、各検出器の読み出しシステムをレベル 1・トリガー・レートが 75kHz、100kHz のときのデッド・タイムまたはデータ損出率がそれぞれ 6%、1% 以内になるように設計する。このデッド・タイムあるいはデータの損失の主な原因はデランドマイザが溢れていてレベル 1・トリガーが到達してもデランドマイザにデータを書き込むことができない状況になってしまうことである。TGC の読み出しシステムでのデランドマイザのサイズとデッド損失率の関係をみるためにシミュレーションを行なった。

3.7.1 シミュレーションの仮定

シミュレーションの入力データとしては、レベル 1・トリガー・レート、データ量およびデータを処理する処理系の能力 (いまの場合は LS-リンクのデータ送信速度) を仮定する必要がある。

デタンドマイザのサイズに対する要請は 75kHz と 100kHz のレベル 1・トリガー・レートに対するものなので、シミュレーションもこれらのトリガー・レートを仮定して行なった。

つぎにデータ量であるが、入力データには、3.2 節で議論したバックグラウンド・ヒットとミューオン・トリガーを出力するミューオン・ヒットがある。

3.2 節で議論したバックグラウンド・ヒット・レートは、シミュレーションで求めたバックグラウンド放射線量をもとに計算した。ここでは、このシミュレーションの不定性を考慮し、3.2 節で計算したバックグラウンド・ヒット・レートに安全係数 5 を掛けた値を用いた。

レベル 1・トリガーとミューオン・トリガーの間にはもちろん相関がある。しかし、CTP がレベル 1・トリガーを出力するためのミューオン検出器、カロリーメータからのデータに対する要請は、興味の対象となっている物理現象によって変わるので、レベル 1・トリガーとミューオン・トリガーの間の相関もこれとともに変動する。ここでは、最悪の場合としてすべてのレベル 1・トリガーがミューオン・トリガーを要請していることを仮定する。すなわち、すべてのレベル 1・トリガーに対応するバンチ・クロッシングに対してミューオンによるヒットがバレルまたはエンドキャップのどこかにあると仮定する。なお、ここでは、簡単のため、Low-P_T ミューオンのみを考えることにする。ミューオン・トリガー・レートは、シミュレーションにより表 7 のように予想されている [10]。

	バレル部	エンドキャップ部	計
Low-P_T($P_T > 6$GeV)	10.0	13.2	23.2
High-P_T($P_T > 20$GeV)	0.9	2.8	3.8

表 7: ミューオン・トリガー・レート。単位は kHz。Low-P_T、High-P_T でそれぞれ、$1\times10^{33}\mathrm{cm}^{-2}\mathrm{s}^{-1}$、$1\times10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$、の加速器ルミノシティーを仮定している。

この表 7 から、各レベル 1・トリガーに対して、エンドキャップ部にミューオン・ヒットがある確率は、$13.2/23.2 = 0.569$ である。ここではエンドキャップ内のミューオン・レートの η 依存性を無視し、エンドキャップ全体で平均して単位面積あたりの入射ミューオン数を計算した。これと、各ローカル・DAQ・ブロックの面積から、ローカル・DAQ・ブロックごとのミューオン・ヒットによるデータ量を計算した。このミューオン・ヒットのデータ量を 3.2 節で議論したバックグラウンドによるデータ量に安全係数 5 を積算したものとともに表 8 にまとめる。

ヒット の種類		トリプレット		ダブレット	
		フォワード	エンドキャップ	フォワード	エンドキャップ
バックグラウンド a	hits/event bits/hit	0.093 48 or 96	0.26 48 or 96	0.12 96	0.21 96
バックグラウンド b	hits/event bits/hit	0.043 144	0.12 144	0.042 192	0.072 192
バックグラウンド c	hits/event bits/hit	0.021 240	0.057 240	0.020 384	0.035 384
ミューオン	hits/event bits/hit	0.0039 240	0.011 240	0.0031 384	0.0055 384
計	bits/event 平均処理時間	20 230	54 306	29 239	49 288

表 8: 各種のヒットによるイベントあたりのローカル・DAQ・ブロック内のヒット数 (hits/event) とヒットあたりのデータ量。イベント・レートはローカル・DAQ・ブロックあたりの値である。処理時間の単位はクロック (25ns) である。

つぎに、処理系の能力、すなわち、スター・スイッチが 1 イベントを処理するのにかかる時間を計算する。この処理時間とは、スター・スイッチがひとつひとつのスレーブ・ボードに順次接続をつなぎかえて全スレーブとの間でコマンドの送信とデータの受信を行なう時間である。まず、スター・スイッチが 1 つのスレーブに接続し、そのスレーブからデータを読み出し終るまでにかかる時間を計算する。マルチプレクサで目的のスレーブに接続するのに 1 クロックかかる。スレーブへのデータ送信コマンドの送信に 3 クロックかかり、スレーブに読み出すべきデータがない場合、スレーブからの信号の受信に 4 クロックかかる。これらのほかに、ケーブルの遅延が往復で 4 クロック、スレーブとスター・スイッチでコマンドをデコードするのにそれぞれ 1 クロックかかる。以上、合計 14 クロックが読み出すべきデータを持たないスレーブとの応答にかかる時間である。読み出すべきデータがある場合には、14 クロックにデータの長さを付加したクロック数だけの読み出し時間がかかる。したがって、各スター・スイッチが 1 イベントを処理するための所要時間は、ローカル・DAQ・ブロック内の全データ量 (ビット数) と $14 \times$ スレーブ数 の和である。表 8 に 1 イベントを処理するのに必要な平均クロック数を付加する。

ここでのデランドマイザのサイズの議論は、イベントあたりの平均処理時間が最も長いエンドキャップ部のトリプレット・スレーブ・ボードに関して行なう。

3.7.2 シミュレーションの方法

シミュレーションは、各バンチ・クロッシングに対してレベル 1・トリガーが発生するかどうか、またトリガーが発生した場合には、そのイベントにたいしてローカル・DAQ・ブロック内の総データ量を乱数を発生して決定し、デランドマイザに残っているデータを処理するための処理時間およびデランドマイザのなかに溜っているデータ量の時間変化を評価した。

シミュレーションのフロー・チャートを図 35 に示す。

バンチ・クロックの数を数える時刻 t とその時点までにデランドマイザに溜ったデータをスター・

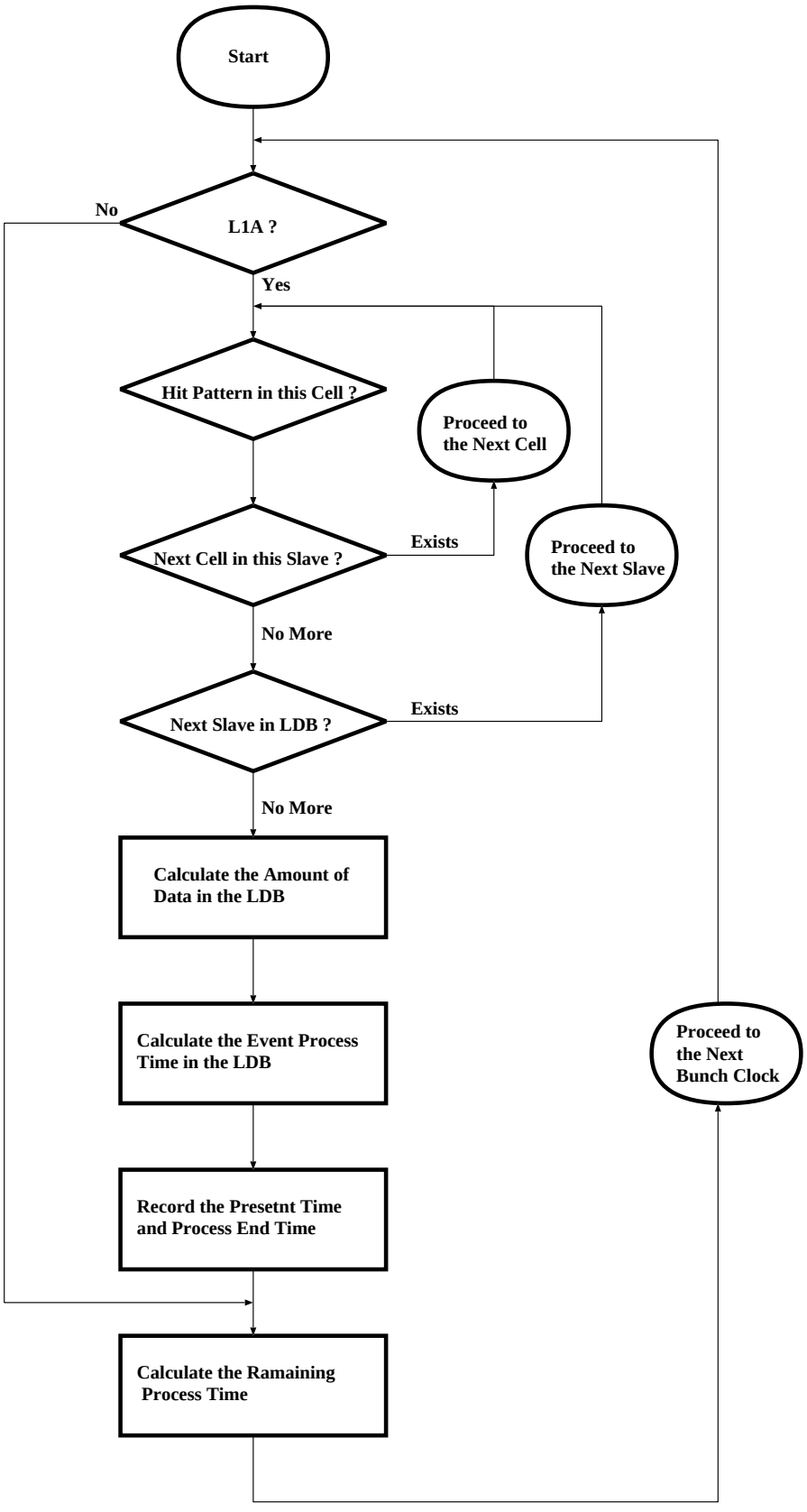

図 35: シミュレーションのフロー・チャート。

スイッチに送信するための残りの処理時間 $Drest$ を変数として用意する。

まず、各バンチに対して乱数を発生させ、そのバンチに対してレベル 1・トリガーが発生するかどうかを決める。このバンチに対してレベル 1・トリガーが発生しなければ、残余処理時間 $Drest$ から 1 を差引き、つぎのバンチ・クロッシングにすすむ (時刻 t に 1 加える)。

レベル 1・トリガーが発生した場合には、ローカル・DAQ・ブロック内の全データ量を計算する。各ワイヤー・スレーブ・ボードには 8 チャンネルの入力セルが 12 セルあるが、それぞれに対して乱数を発生し、これら各入力セルにヒットがあるかどうか、さらにヒットがあった場合には表 8 中のどの種類のヒットであるかを決め、このヒットに対するデータ量を決める。この際、各入力セルにヒットがある確率は、表 8 のローカル・DAQ・ブロック全体での hits/event をローカル・DAQ・ブロック内のワイヤー・スレーブ・ボードの数で割り、さらにスレーブ内の各入力セルの数で割って算出した。このようにして決定した各セルのデータ量を足し合わせ、ローカル・DAQ・ブロック全体でのデータ量を計算する。このデータ量に、スター・スイッチが全スレーブをスキャンする時間 (スレーブの数 $\times$ 14 クロック) を加算し、このイベントのデータをスター・スイッチに読み出す処理時間 D を計算する。このイベントのデータ読み出しが完了するのは、$Drest + D$ 後である。時刻 t とイベント・データの読み出しが終了する時刻 $t + Drest + D$ を記録する。このあと、残りの処理時間 $Drest$ を $Drest + D - 1$ に置き換え、つぎのバンチ・クロッシングにすすむ。

以上のようにしてレベル 1・トリガーのタイミングとそのイベント処理終了のタイミングの配列をつくった。ある時刻 t にデランドマイザに残っているイベント・データの数は、それより早く発生したレベル 1・トリガーの数からそれまでに処理が終了したイベントの数を差し引いて計算した。

3.7.3 シミュレーションの結果

前節の方法で計算した 75kHz、100kHz のレベル 1・トリガー・レートでのデランドマイザ内のデータの数の時間的推移をそれぞれ図 36、図 37 に示す。

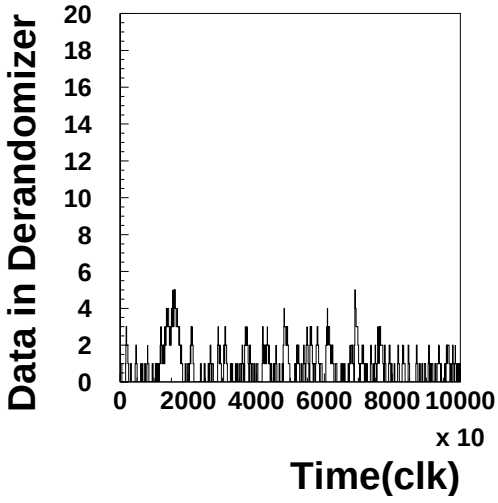

図 36: 75kHz のレベル 1・トリガー・レートでのデランドマイザ内のデータの数の時間的推移。

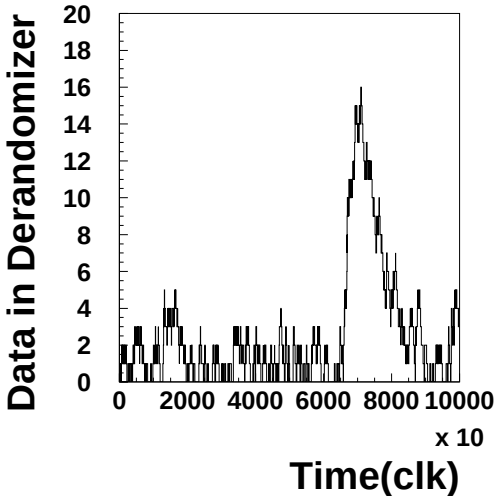

図 37: 100kHz のレベル 1・トリガー・レートでのデランドマイザ内のデータの数の時間的推移。

シミュレーションは 10^9 バンチ・クロック分行ない、75kHz、100kHz のレベル 1・トリガー・レートでそれぞれレベル 1・トリガーは 1877516、2501259 回発生した。このときのデランドマイザ内のデータの数の分布が、それぞれ図 38、図 39 である。

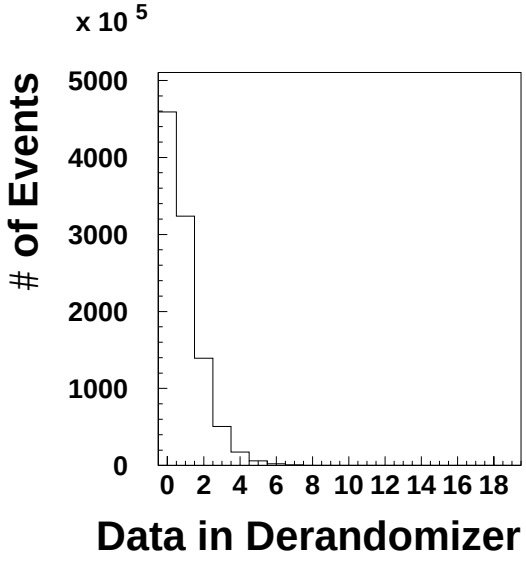

図 38: 75kHz のレベル 1・トリガー・レートでのデランドマイザ内のデータの数の分布。

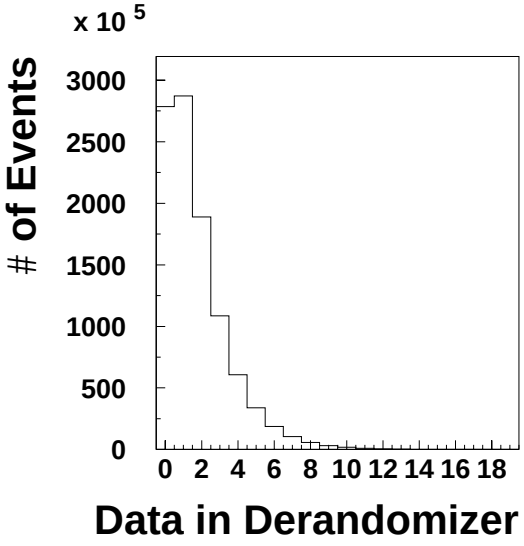

図 39: 100kHz のベル 1・トリガー・レートでのデランドマイザ内のデータの数のの分布。

これらの分布から、任意の時刻において処理待ちのデータ数が n である確率 P_n を求めることができる。デランドマイザ内の処理待ちのデータ数が N より多くなる確率は,

$$\alpha = P(n > N) = 1 - \sum_{n=0}^{N} P_n$$

である。この式は、デランドマイザが保持できるデータの数 (デランドマイザのサイズ)N と、デランドマイザからデータが溢れてしまう確率 (データの棄却率) の関係を表している。この関係を 75kHz、100kHz のそれぞれに対してグラフにしたものが図 40 である。

デランドマイザのサイズを 4 以上にすれば 75kHz のレベル 1・トリガー・レートの場合にデータの棄却率を 1% 以下にできる。また、100kHz のレベル 1・トリガー・レートの場合にデータの棄却率を 6% 以内にするためには、デランドマイザのサイズを 5 以上にする必要がある。したがって、デランドマイザのサイズは、5 以上にする必要がある。

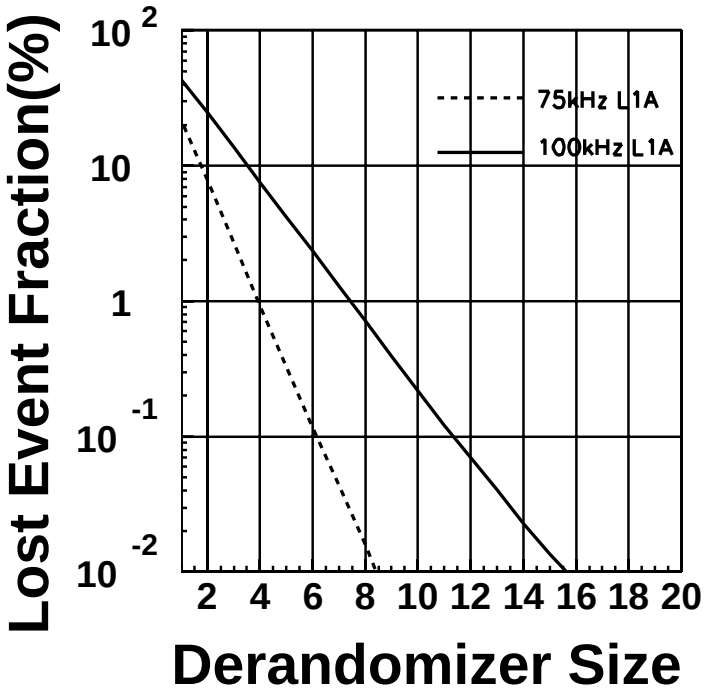

図 40: デランドマイザのサイズとデータの棄却率の関係。

3.8 verilogHDL によるシミュレーション

3.8.1 verilog 言語と RTL 記述

一般に、デジタル回路は複雑であり、とくにその詳細な記述レベルであるロジック・ゲートの集合体としてみる場合、非常に多くの要素の集合体である。したがって、デジタル回路を設計する場合、ロジック・ゲートをひとつひとつ結線する作業を繰り返す設計手法は困難になりがちである。デジタル回路をより抽象化し、システムの機能・振舞いを記述することにより設計することを実現するのがハードウェア記述言語 (HDL) である。ハードウェア記述言語では、具体的な実装工程によらずに記述をすすめることができ、この記述を出発点として自動合成することで実際のハードウェアに実装することが可能である。現在標準的に回路設計に使用されているハードウェア記述言語には VHDL 言語と verilog 言語があるが、ここでは、verilog 言語を利用して設計を行なった。

verilog 言語では、回路の動作を記述する際のさまざまな抽象化レベルをサポートしている。高い抽象化のレベルでは、システムの動作のみを簡単に記述することでおおよその機能を理解することができるが、実際に合成を行なってハードウェアに実装することができない。また、もっとも低い抽象化レベルでは、ロジック・ゲートどうしの結線 (ネットリスト) を記述することで、回路動作を記述することも可能である。自動合成して実際のハードウェアに実装するためには、各レジスタ転送に対してクロックのエッジなどの条件を指定する RTL(レジスタ・トランスファー・レベル) で回路動作を記述する必要がある。

TGC 読み出しシステムを RTL レベルで動作記述し、これを verilog シミュレータを用いて性能評価した。

3.8.2 スレーブ・ボードの読み出し回路

デランドマイザからのデータをシリアル変換する部分を RTL 記述し、この機能を verilog シミュレータで評価した。ただし、ここでは簡単のため、図 41 のように入力チャンネルを 2×8 ビットとした。また、スター・スイッチからのデータ送信コマンドをデコードし、シフト・レジスタに対してデータ・ロード・コマンドを発信するコマンド・デコーダも組み入れた。この verilog ソース・コードを A.1 に載せる。

回路の入力は、スター・スイッチからの SYNC(sw_sync)、DATA(sw_data) の信号 (CLOCK はスレーブのクロックを使う)、各セルの入力データ (cell0、cell1)、その読み出し制御の信号 (read0、read1) とデランドマイザのエンプティー・フラグ (emptyb)、フル・フラグ (full) と HasData 信号である。これらは、シミュレーションの入力テスト・ベクトルとして与える。

出力は、スレーブからスター・スイッチに送信する SYNC(sl_sync)、DATA(sl_data) の信号である。

図 42 にこのシミュレーションの結果を示す。1 回目のスター・スイッチからのデータ送信コマンドのタイミングでは、HasData、read0、read1 すべてが Low なので、スレーブからスター・スイッチに送信するパケット中のデータは、emptyb、full の 2 ビットである。2 回目のデータ送信コマンドのタイミングでは、read0 が High になっておりしたがって cell0 のデータが読み出される。図 42 中、cell0、cell1 のデータは 16 進数表示となっており、08 は下位 3 ビット目が High で残りの 7 ビットは Low であることを表している。この 2 回目のデータ送信では、パケット中 1 ビット目、2 ビット目がそれぞれ emptyb、full である。この次の 8 ビットが cell0 のデータであるが、確かに 5 個目のクロックで High に立ち上がっている。cell0 のデータのあとには、8 ビットのデータ・

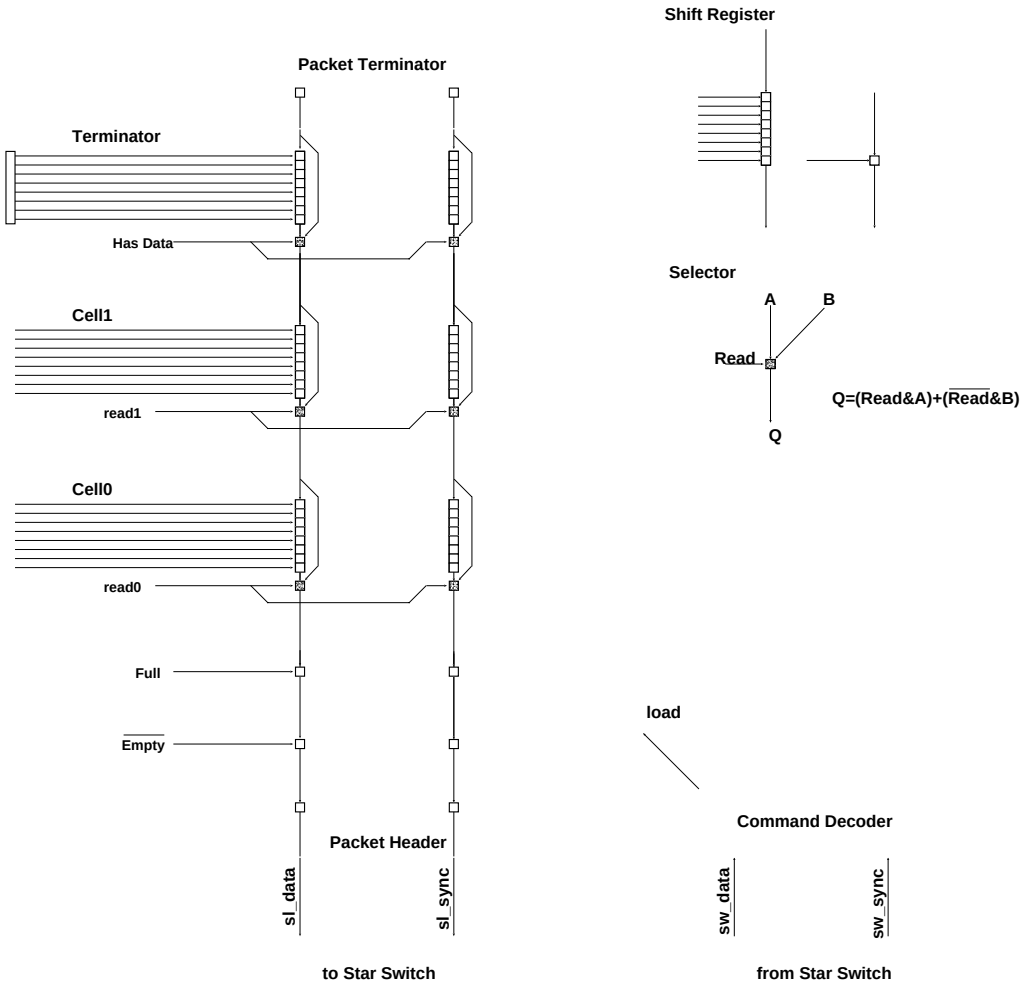

図 41: verilog シミュレータで性能評価したスレーブ・ボード読み出し回路。

ターミネータが付加され、パケットを閉じている。

3.8.3 スター・スイッチのシークエンサ

スター・スイッチのシークエンサを RTL 記述し、verilog シミュレータでこの回路の性能を評価した。

図 43 中ハッチがかかっている部分を HDL 記述した。この verilog ソース・コードを A.2 に載せる。

シークエンサは、マルチプレクサが接続すべきスレーブのアドレス addr、スレーブへの信号 sw_sync、sw_data とスター・スイッチ内の FIFO への書き込み許可 write を出力する。また、入力は、スレーブからの信号 sl_sync、sl_data である。ここでは簡単のため、スレーブの数を 4 つに限り (addr の値は 0 以上 3 以下)、スレーブからのデータも 3.8.2 節と同様に簡略化してシミュレーションを行なった。

このほかに、FIFO へのデータ入力のタイミングを write 信号に合わせるために sl_data を遅ら

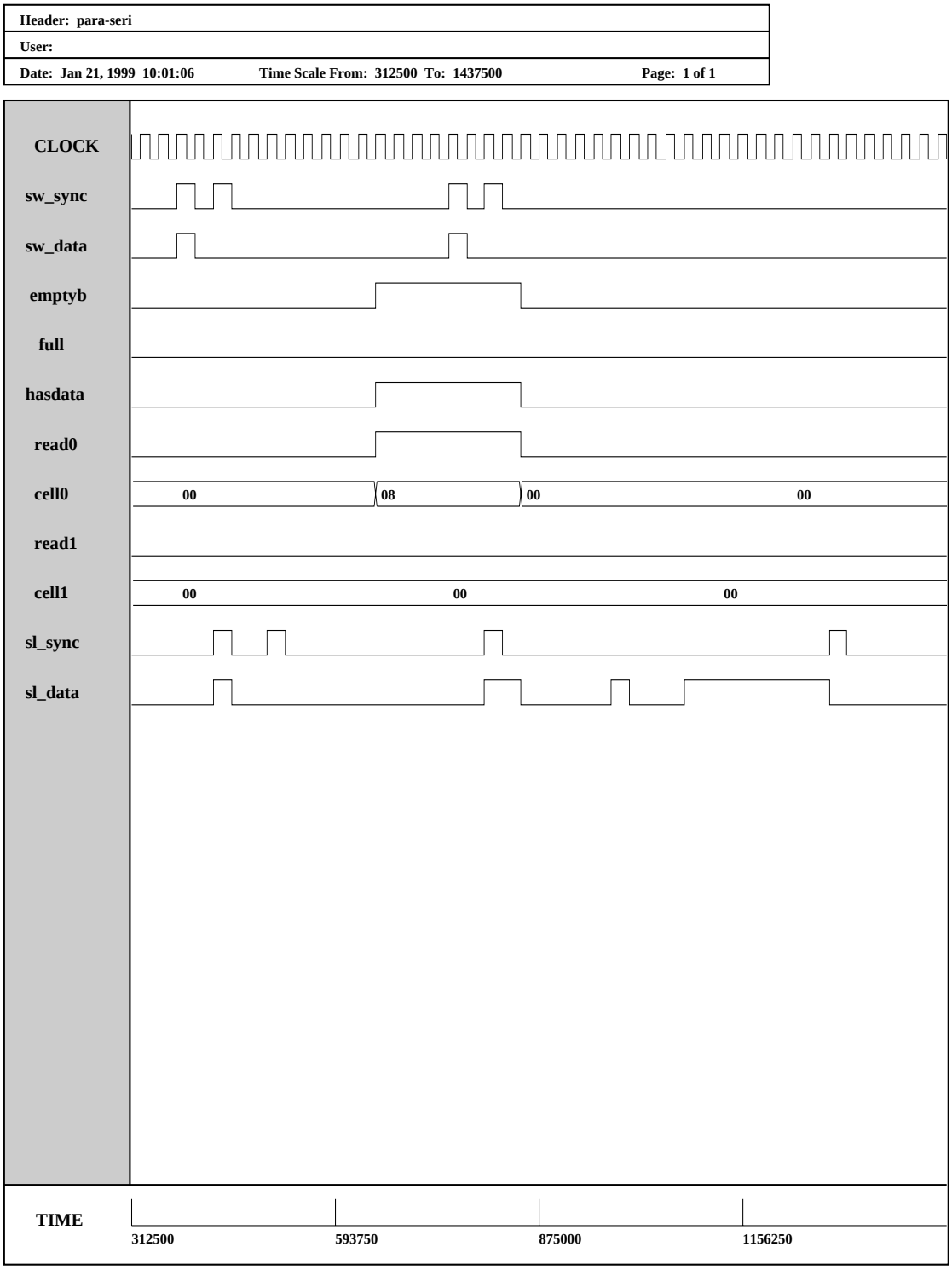

図 42: スレーブ・ボード読み出し回路の verilog シミュレーション結果。

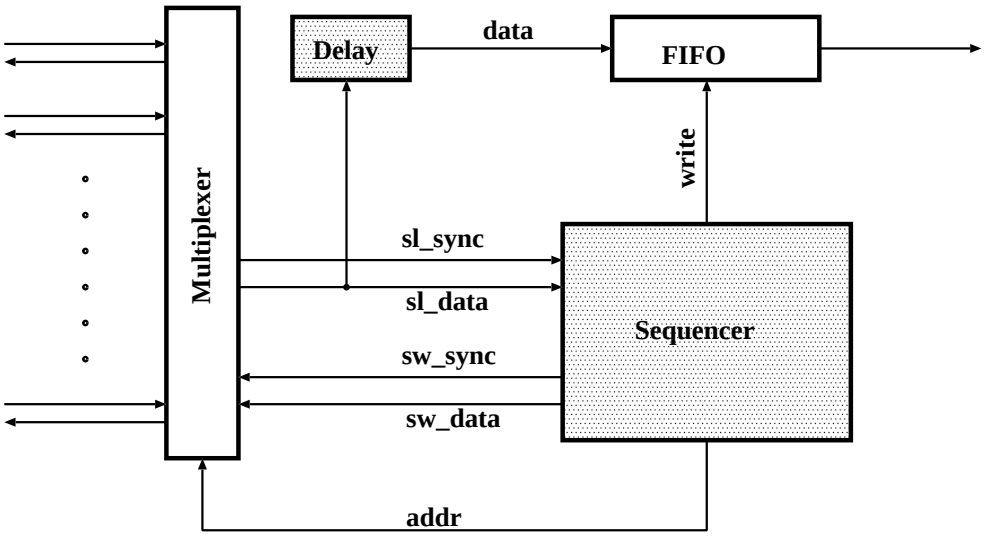

図 43: verilog シミュレータで性能評価したスター・スイッチのシークエンサ。図のハッチのかかっている部分を性能評価した。

せた data 信号を FIFO への入力データにしている。

図 44 にこのシミュレーションの結果を示す。

まず、初期状態では、接続するスレーブのアドレス addr は 0 である。1 回目のデータ送信コマンドに対するスレーブの応答は、emptyb=0 であり、したがってスター・スイッチは addr を繰り上げない。2 回目もデータ送信コマンドはスレーブ 0 に送信されるが、今度の応答は emptyb=1 で、デランドマイザにデータが溜ったことを示している。スレーブ 0 にはヒットはないので、こんどは addr を繰り上げ、スレーブ 1 にデータ送信コマンドを送信する。このスレーブ 1 には、ヒットがあったので、パケットのデータの 3 ビット目以降に 16 ビットのヒット・データが付け足されている。これを FIFO に格納するためにシークエンサは 16 クロックの間、write を High に保持する。また、この write 信号は、スレーブからのデータのタイミングより送れるので、sl_data をレジスタで遅延させた data 信号を FIFO の入力データにする。このようにして、スレーブ 2、3 とも通信する。スレーブ 3 との通信が済んだ時点で再び初期状態にもどる。

3.8.4 verilogHDL シミュレーションの結論と今後の予定

スレーブ・ボードのシリアル変換部分とスター・スイッチのシークエンサを verilogHDL で記述し、シミュレーションを行なったが、両回路とも仕様どおりの機能を実現できた。また、RTL 記述で設計を行なったので、実際にデバイスに実装できるレベルの設計でもある。ただし、今回のシミュレーションでは配線や論理ゲートにおける信号の遅延情報は採り入れていないので、実装の際にこれらの遅延を解消するために一部設計 (ソース・コード) を変更することはあり得る。そこで、これらの遅延が実装時にこの設計に対してどの程度影響するかを理解するために今回の設計を FPGA に実装しての性能評価を行なう予定である。なお、試験用のデバイスとして FPGA を選定している理由は、FPGA の場合には回路が要求の性能に達するまで設計を変えながら繰り返しコンフィギュレーションを行うことが可能だからである。

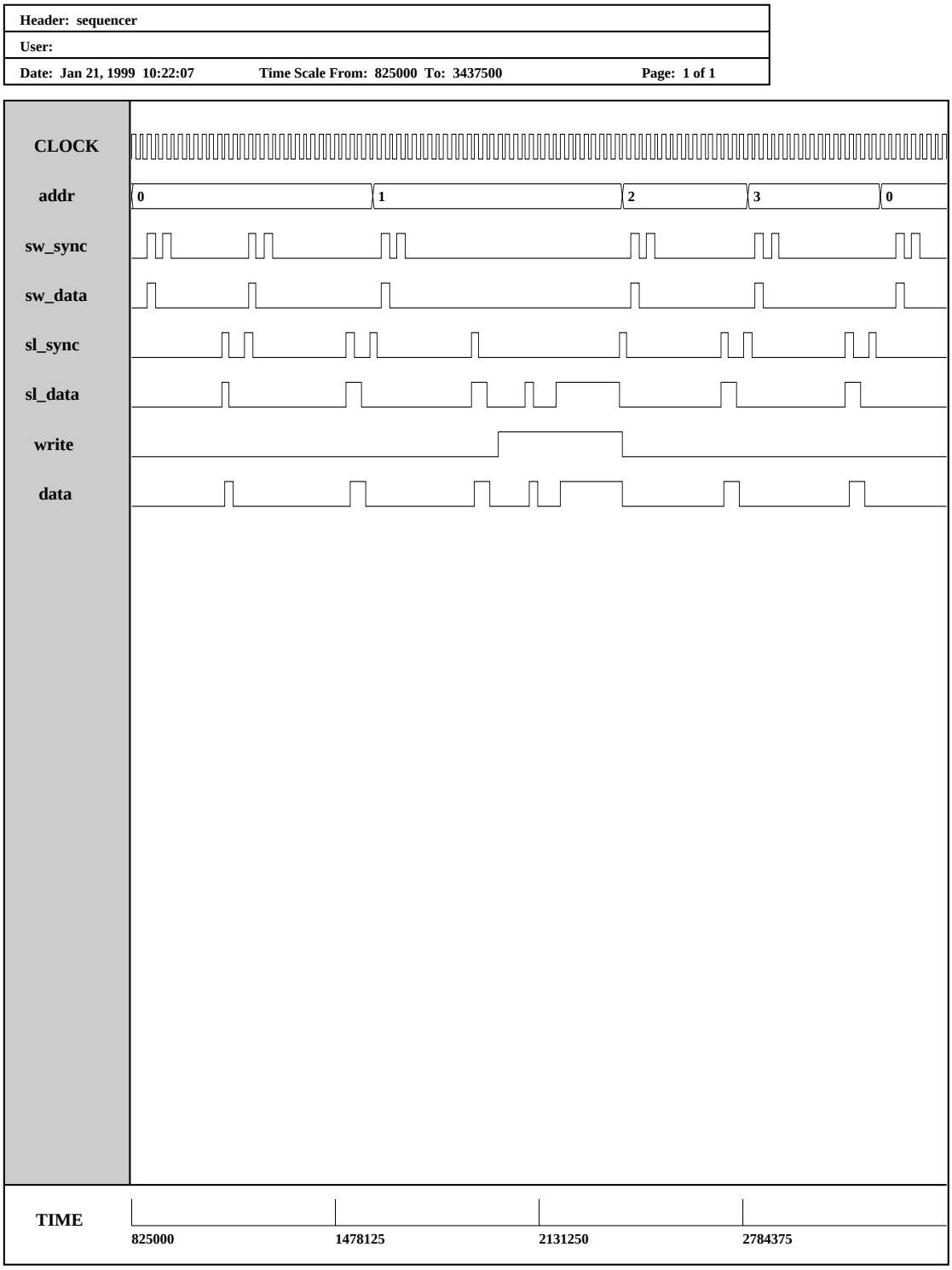

図 44: スター・スイッチ・シークエンサの verilog シミュレーション結果。

4 MWPC読み出し用ASICの試作

TGCの読み出しシステムのうち、スレーブ・ボードとHigh-P_T・ボード内の論理回路はASIC(Application Specific Integrated Circuit)での実装を予定している。このほか、スター・スイッチの回路も放射線耐性のよいFPGA(Field Programmable Gate Array)またはASICでの実装が検討されている。とくに論理回路に関しては、近年の急速なハードウェア記述言語(HDL)、論理合成ツールの充実にともない、ASICの設計手法は大きく変貌を遂げつつある。また、素粒子実験の分野では、大規模な論理回路のASICでの実装は、ほとんど前例がない。したがって、TGCの読み出しシステムを開発していく上で論理回路をASICに実装するための設計手法を理解することは非常に重要な課題である。

ここでは、ASICの論理回路設計手法に対する理解を深めるためにASICの試作を行なった。スレーブ・ボードとHigh-P_T・ボードのレベル1・バッファは、ビーム・クロッシングからレベル1・トリガーを受信するまでの間データを保持しておくためのパイプラインである。このデータを保持しておくべき時間は2.5μsであり、25nsのクロックで動作する100段のシフト・レジスタで実現する。そこで、ここでは100段のシフト・レジスタをASIC化することにした。ただし、シフト・レジスタだけを製作してもあまり実用性がない。そこで、シフト・レジスタを遅延パイプラインとして内蔵し、TGCの読み出しを行なえるシステムを開発することにした。ここでは、とくにその用途をTGCの読み出しのみに限らず、より汎用性のシステムとして、一般のMWPCで使用が可能な読み出しシステムをASIC化した。

一般に、ASICの設計においてレイアウトの工程は技術的に難しいのでベンダーが行ない、ユーザはHDL(Hardware Design Language)記述および設計の各段階でのシミュレーションのみ行なう場合が多いが、ここでは、レイアウトの工程まで含めた全設計工程を行なった。また、実装されたサンプルICの性能評価し、その問題点を検討した。

4.1 デバイスの選定

スレーブ・ボードとHigh-P_T・ボードはASICで開発することを予定しているが、本節では、これらの回路に対する要請を検討し、ターゲット・デバイスの選定の経緯を述べる。

表3にあるように、TGCでは読み出し総チャンネル数は、全体で約33万チャンネルにのぼる。スレーブ・ボードの数は2784枚、High-P_T・ボードの数も480枚である。したがって、スレーブ・ボードとHigh-P_Tボード上の回路は、単価が安いデバイスで実装し、大量生産する必要がある。また、これらの回路の規模も大きいので、実装面積を小さくするために規模の大きい回路を実現できるデバイスを選択する必要がある。さらに、これらのボードは、カウンター・ホールに配置されるために多量の放射線を浴びる。とくに最悪の位置では、$9.7 \times 10^{10}\mathrm{cm}^{-2} \cdot \mathrm{year}^{-1}$の中性子フラックス、$6.2 \times 10^{-1}\mathrm{Gy} \cdot \mathrm{year}^{-1}$のガンマ線量がシミュレーションにより予想されている[12]。このため、放射線耐性のよいデバイスを選定することは非常に重要である。

ロジック・デバイスには、以下のような種類がある。なお、ここでは、ゲート・アレイおよびフルカスタムをASICと呼んでいる。

PLD Programmable Logic Device。数百ゲート程度の集積度を実現する小規模のロジック・デバイスである。非常に手軽に実装することが可能である。

FPGA Field Programmable Gate Array。内部には論理ブロックと内蔵メモリがある。論理ブロックの配置は固定されており、内蔵メモリに書き込まれたコンフィギュレーション・プロ

グラムにより論理ブロックどうしの結線を行ない、論理回路を実現する。このコンフィギュレーション・プログラムの書き込みは、電源投入時またはシステム初期化時におこなう。このため、工場におけるカスタム化は必要ない。ゲート数は、数千から数万程度である。概して放射線耐性は良くないが、放射線耐性のよいものも開発されている。コストは、大量生産されている標準 IC と同じ習熟曲線に沿った傾向を示す。

ゲート・アレイ その製造の工程から大きくわけてゲート・アレイとカスタム IC がある。ゲート・アレイは、ゲートの配置は固定されており、製造の最終段階でユーザがゲート間を接続し、論理を実現する IC である。標準 IC とは異なり、コストにはユニットあたりの製造費のほかに固定費が含まれる。論理ゲート数は大きいものでは 100000 ゲート以上のものまである。

カスタム IC 一方、カスタム IC は、配線だけでなくゲートの配置を含めた全レイアウトをユーザが設計する。したがって、レベルの高いユーザ論理の構築が可能であるほか、アナログ回路の実現も可能である。固定費はもっとも高いが、大量のアプリケーションに対する製造コストは最も安くなる。

ASIC の場合には、製造プロセスによって放射線耐性が大きく異なるが、概して FPGA よりよい。製造プロセスによっては放射線耐性を保証するものもある。

TGC のスレーブ・ボード、High-P_T は生産量、また放射線耐性に対する要請からも ASIC での実現が適当である。

スター・スイッチの場合には、その総数は 200 程度であるが、カウンター・ホール内に配置されるのでやはり放射線耐性は要求される。そこで、放射線耐性のよい FPGA で実装するか ASIC で実装するかは今後検討して決定する。

4.2 プロセスの選定

1996 年に東京大学に全国共同利用施設「大規模集積システム設計教育研究センター」(VLSI Design and Education Center, VDEC) が設置された。VDEC は大学および高専における大規模集積回路 (Very Large Scale Integration, VLSI) の設計教育の充実、VLSI チップの試作の支援を目的としている。とくに、複数の大学・高専による別々のチップの設計を集め、1 枚のシリコン・ウェーハ上に相乗りで配置した設計データにまとめて企業に試作を依頼するため、あまり経費をかけずに ASIC の試作をすることを可能にしている。ただし、VDEC は LSI チップの設計教育を目的としているため、設計の全段階をユーザが行ない、ベンダーへの委託はいっさい行なわない。また、量産はサポートしていない。今回のチップ試作の最大の目的はチップ設計工程の理解を深めることであるので、この VDEC を利用して ASIC の試作を行なった。

プロセスは、VDEC で試作がサポートされているローム社の CMOS 0.6μm フルカスタムを選択した。チップ・サイズは 4.5mm 角、利用可能の最大信号ピン数は 87 ピンである[7] 。

4.3 設計の工程

本節では、MWPC 読み出し用 ASIC の設計の詳細を説明する前に論理回路を ASIC で実装する際の設計の工程について説明する。

[7]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) の協力で行なわれたものである。

設計工程は、構築すべきシステムの概念および最終的な実装上の制約からスタートする。設計は、各設計段階でいくつかの解を提案し、それが与えられた制約を満たすか否かを試していく操作の反復である。この設計段階には、HDL 記述、この HDL 記述をもとにゲート・レベルのネット・リストを生成する論理合成、レイアウトの 3 段階がある。

論理システムの設計工程の一覧を図 45 に示す。

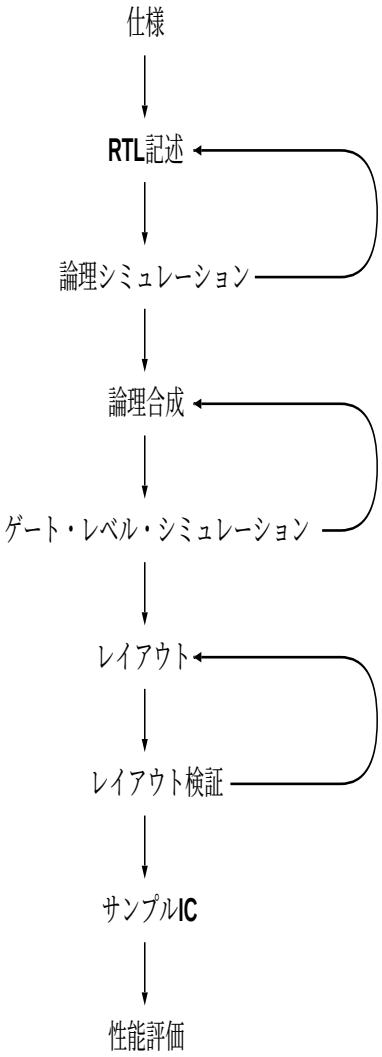

図 45: ASIC 設計のフロー・チャート。

設計の各工程を MWPC 読み出し ASIC の設計に即して以下に説明する。

RTL(Register Transfer Level) 記述 システムをハードウェア記述言語 (HDL) で記述する。この HDL 記述をもとに合成を行なって IC への実装を行なうためには、各レジスタ転送に対してクロックのエッジなどの条件を指定する RTL(レジスタ・トランスファー・レベル) で回路動作を記述する必要がある。MWPC 読み出し用 ASIC は、verilogHDL で記述を行なった。

論理シミュレーション HDL で記述したシステムにテスト信号を入力し、仕様どおりに動作するかどうかを確認する。このシミュレーションで仕様どおりの動作が得られるまで RTL 記述

をやりなおす。MWPC 読み出し用 ASIC では、CADENCE 社 verilogXL シミュレータを用いて行なった。

論理合成 HDL 記述をゲート・レベルのネット・リストに変換する工程である。この際、ゲートはターゲットのデバイスによって異なるため、論理合成ツールにターゲット・デバイス固有のライブラリを指定する必要がある。論理合成は、SYNOPSYS 社 Design Analyzer を使用して行なった。

ゲート・レベル・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、仕様どおりに動作することを試験する。このシミュレーションでは、ゲートでの遅延時間情報と、合成ツールが予想するレイアウトでの配線遅延の情報も組み込まれる。CADENCE 社 verilogXL シミュレータを用いて行なった。

レイアウト 論理合成結果のネット・リストをもとにターゲット・デバイス上にゲートを配置し、また配線を行なう。レイアウトは、そのツールを使いこなすことが難しいため、一般にユーザ自身が行なうことは少なく、ベンダーに委託することが多い。最近では、自動配置配線ツールが発達してきている。ここでも、Avant 社製の自動レイアウト・ツール AquariusXO および Cadence 社 LayoutEditor をを用いて行なった。

レイアウト検証 レイアウトが設計ルール[8] を破っていないことを確認する作業 DRC(Design Rule Check) とレイアウトが正しくネット･リストを反映しているかどうかを検査する作業 LVS(Layout vs Schematic) である。DRC、LVS ともツールは Cadence 社 Dracula を用いた。

設計工程では、ターゲット・デバイスのデータをライブラリとして設計ツールに入力する必要があるが、このライブラリすべて、VDEC から提供された。

4.4 仕様

今回設計した MWPC 読み出し用 ASIC は、トリガー信号が外部から入力された後にゲートを開き、このゲート内に MWPC の各チャンネルにヒットがあるかどうかを判断するものである。読み出しチャンネルは、IC あたり 16 チャンネルである。動作クロックの周波数は、40MHz を想定して設計を行なった。

MWPC 読み出し用 ASIC の仕様は、つぎのようになる。外部入力のトリガー信号のタイミングでゲートを開き、ゲート内での MWPC のヒットの有無を出力する。ゲート幅は、IC 全体の動作クロックの 1 ～ 15 周期に設定可能である。また、トリガー信号のタイミングに合わせるために MWPC の出力信号を 100 段のシフト・レジスタで遅延している。シフト・レジスタの出力を直接出力するように設定を切替えることもできる。アノード、カソードとも読み出すことが可能であり、またセルフ・トリガーのためのトリガー信号の出力も用意している。このほか、イベントの処理中にはビジー信号を、イベントの処理終了時にはレディー信号を用意している。ASIC の動作電圧レベルは、TTL レベルであり、したがって全入出力の信号レベルは TTL レベルである。

図 46 に MWPC 読み出し用 ASIC のブロック図を示す。

図 46 中の機能ブロックの機能を以下に説明する。

[8] 製造プロセスを通化したウェーハ上のパターンが設計時の幾何形状と寸法を保持するためには、設計レイアウトの配線の線幅、線間隔などをプロセスで実現できる寸法にしなくてはならない。このプロセス上許容される寸法に関する一連の規則のこと。

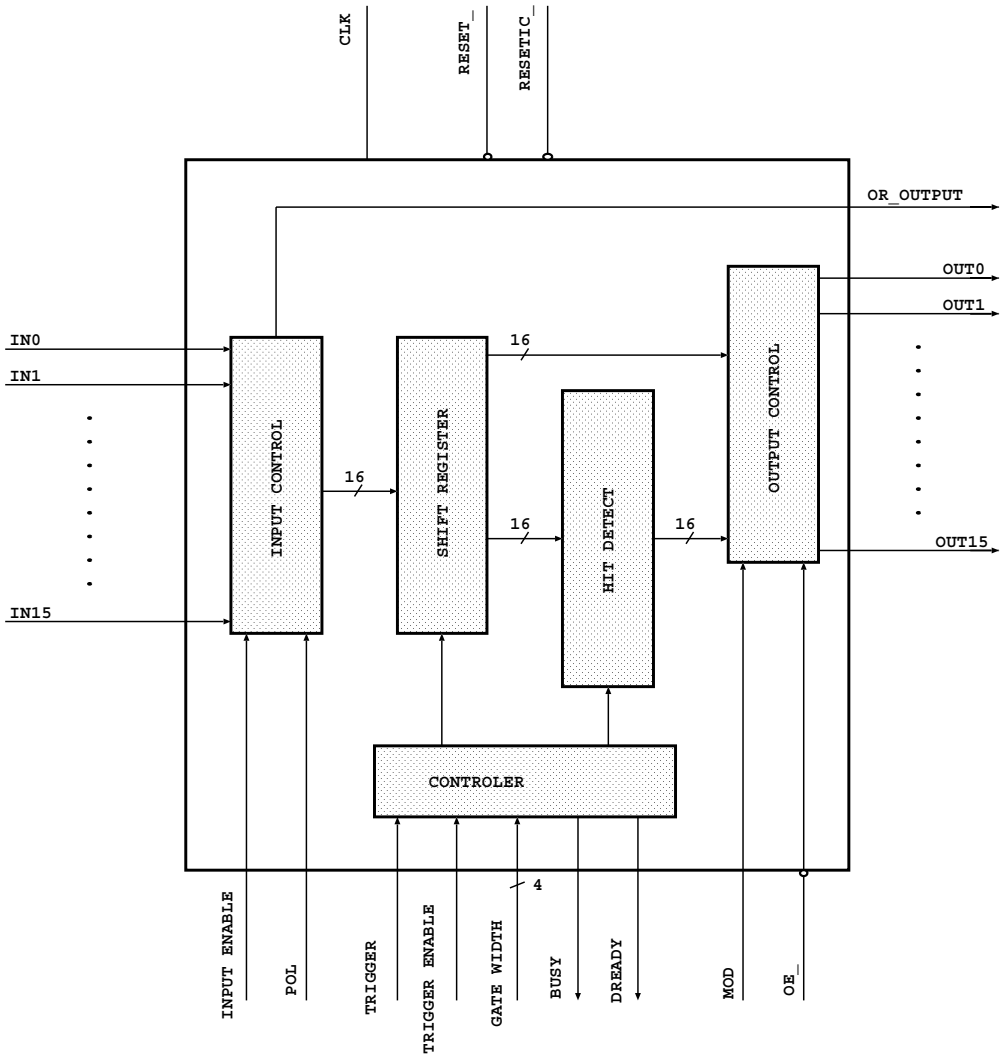

図 46: MWPC 読み出し用 ASIC のブロック図。

INPUT CONTROL INPUT ENABLEが High の場合に、IN0~IN15 の立ち上がり (POL=Low の場合) あるいは立ち下がり (POL=High の場合) を検出し、1 クロック分だけパルスを出力する。また、全 16 チャンネルの入力信号の論理和をとって OR OUTPUT に出力する。

SHIFT REGISTER 100 段のシフト・レジスタ。RESETIC_でリセットされる。イベントごとのリセット (RESET_による) ではリセットされないので、リセット後、100 クロック待たないでつぎの TRIGGER を入力できる。

CONTROLER TRIGGER 入力のタイミングで GATE 信号を High にドライブする。クロックの数を数えるカウンタを内蔵しており、TRIGGER を受けてからクロックを数えはじめ、カウンタの値が GATE WIDTH になった時点で GATE を Low に戻す。また、ビジー信号 BUSY、データのレディー信号 DREADY の制御も行なう。

HIT DETECT GATE が High にドライブされている間にシフト・レジスタからパルスが出力されるかどうかを判断することにより、ヒットを検出する。このヒットの有無の情報を出力

する。この出力は、RESET_またはRESETIC_信号によりリセットされるまで保持される。

OUTPUT CONTROL MOD信号の値により、SHIFT REGISTERの出力(MOD=Low)あるいはHIT DETECTの出力(MOD=High)をOUT0∼OUT15に出力する。また、OUTPUT ENABLE_がHighのときにはOUT0∼OUT15をHigh Impedanceにする。

つぎに図46中の信号を説明する。スロー・コントロール入力信号は、ICの動作パラメータであり、信号レベルを固定して入力する。スロー・コントロール以外の入力信号はすべてクロックと非同期に入力することが可能である。なお、信号名に“_”がついている信号は、負論理である。

CLK(入力) ASIC全体のクロック。

IN0∼IN15(入力) MWPCからのTTL入力。16チャンネルの読み出しができる。

INPUT ENABLE(入力、スロー・コントロール) 入力許可。これがHighになっていないと入力を受け付けない。

POL(入力、スロー・コントロール) MWPCのアノードとカソードの出力は、極性が異なる。この極性が＋の場合は信号の立ち上がりを検出し、極性がーの場合は立ち下がりを検出する必要がある。入力の極性が＋の場合には0に、入力の極性がーの場合には1に設定する。

OR OUTPUT(出力) 全16入力チャンネルの論理和。実用時にはトリガー信号等に使用する。

TRIGGER(入力) 外部入力トリガー。

TRIGGER ENABLE(入力、スロー・コントロール) トリガー入力許可。

GATE WIDTH(入力、スロー・コントロール) ゲート幅設定値。クロック周期を単位として4ビットで設定する。

BUSY(出力) TRIGGERのタイミングからRESET_のタイミングまでの間Highにドライブされるビジー信号。

DREADY(出力) GATEの立ち下がりで、出力データは確定する。このGATEの立ち下がりからRESET_までの間Highにドライブされるレディー信号。

GATE(内部信号) ゲート。TRIGGER入力からGATE WIDTHの時間だけ、Highにドライブされる。

MOD(入力、スロー・コントロール) 出力データの選択を設定する信号。LowでSHIFT REGISTERの出力を、HighでHIT DETECTの出力を選択する。

OUTPUT ENABLE_(入力、スロー・コントロール) ヒット・データの出力のコントロール。この信号がLowならばICの出力OUT0∼OUT15を出力ピンに出力し、Highならば出力ピンはHigh Impedanceとなる。

OUT0∼OUT15(出力) ヒット・データのトライステート出力。

RESET_(入力) イベントごとのリセットを行なうためには、全システムをリセットする必要はなく、出力信号を保持しいるレジスタのみをリセットすれば十分である。このイベントごとのリセットを行なう。

RESETIC_(入力) 全システムのリセット。初期化に用いる。

4.5 HDL 記述

今回の設計は verilogHDL で RTL 記述を行ない、その記述を合成、レイアウトして行なった。

ここでは、verilogHDL で記述する前に、全体をコントロール部 (前節の CONTROLER) と読み出し部 (前節の CONTROLER 以外の機能ブロック) の 2 つのモジュールに分割し、それぞれに対して、あらかじめ簡単な回路図を作成した。この回路図を作成する際に、フリップ・フロップや AND、OR などの基本的な論理要素を結線して設計するほうが簡単な回路部分に関しては、これらのネット・リストのレベルまで設計のレベルを落した。動作記述のほうが設計が簡単になる回路部分に関してはモジュールとして残しておき、それ以上の詳細な記述は verilogHDL で動作記述して設計した。

図 47 に読み出し部 (前節の CONTROLER 以外の部分) の回路図を示す。前節で述べた論理ブロックがそれぞれハッチで示されている。

MWPC からの入力データは、まず POL 信号との排他的論理和をとられる。この排他的論理和の出力をシステムの入力データとすることで入力の極性によらずにデータを扱うことが可能になる。この排他的論理和の出力は、つぎに 2 段のフリップ・フロップに入力されるが、ここでは OR OUT_信号をつくる組合せ回路に信号を送るとともに、入力データをクロックと同期をとり、1 クロック分のパルスに変換している。この出力は、遅延用パイプラインであるシフト・レジスタに送られる (シフト・レジスタは動作記述で記述した)。シフト・レジスタの出力は、AND ゲートを通してフリップ・フロップに送られるが、このフリップ・フロップで GATE が High になっている間のヒットの有無を判定している。また、このフリップ・フロップの出力では、リセットされるまでヒット情報が保持される。この後、セレクタでヒット情報とフリップ・フロップの出力のどちらを IC の出力とするかを選択する。

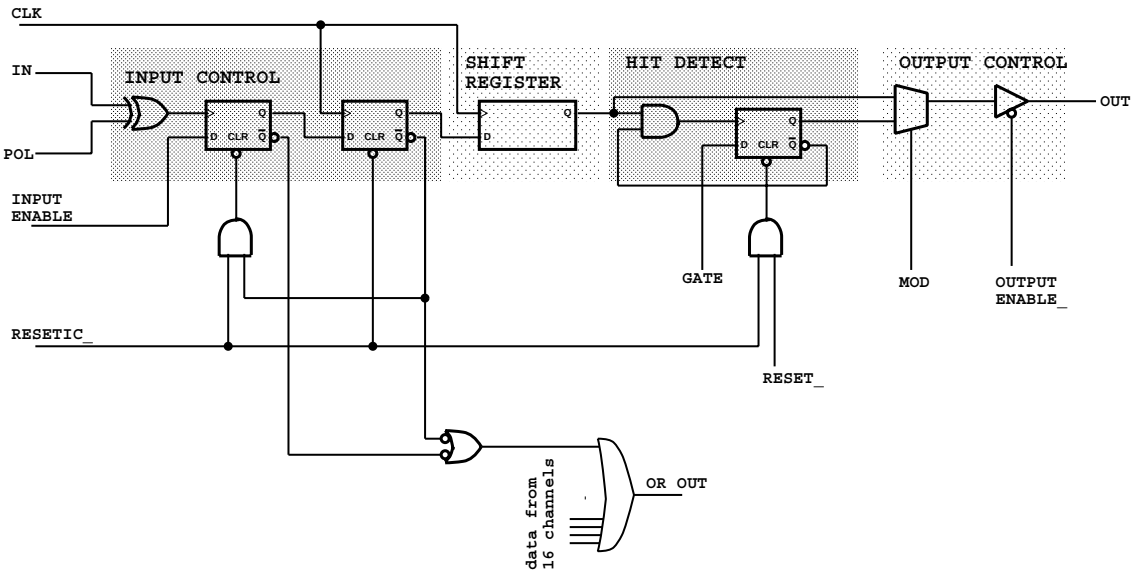

図 47: 読み出し部の回路図。

コントロール部は、カウンタを内蔵しており、トリガー入力後のクロック数を数えてゲートを

開き、またトリガー、ゲート信号の遷移のタイミングでビジー信号、レディー信号を出力する。

図 48 にコントロール部 (前節のの CONTROLER) の回路図を示す。トリガー入力があると、COUNTER ENABLE 信号は High にドライブされ、リセットされるまで High のまま保持される。COUNTER(動作記述で記述した) は、クロックを数えるカウンタを内蔵している。GATE は、COUNTER ENABLE が High になったタイミングから、このカウンタがゲート幅だけのクロックを数えるまでの間 High にドライブされる。BUSY、DREADY はそれぞれ、COUNTER ENABLE の立ち上がり、GATE の立ち下がりのタイミングからリセットされるまでの間 High の状態を保つ。

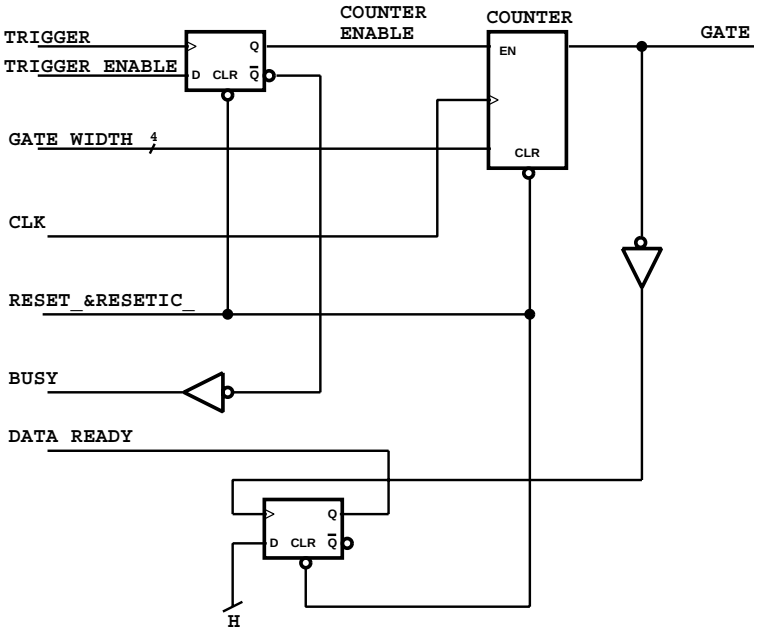

図 48: コントロール部の回路図。

設計は、これらの回路図をもとに verilogHDL でシステムを記述し、これを合成することで行なった。この verilog ソース・コードを B に載せる。

4.6 サンプル IC

図 49 は製作したチップの顕微鏡写真である。

チップの大きさは、4.5mm 角で、コアの大きさは約 4mm^2 である。配置されているゲート総数は、約 4000 である。図中、チップの外周とコアを結んでいる太い線は電源供給線であり、細い線が入出力信号線である。チップの外周には 120 のピンが配置されており、このうち信号入出力ピンとして 48 ピン、電源ピンとして 32 ピンを使用している。

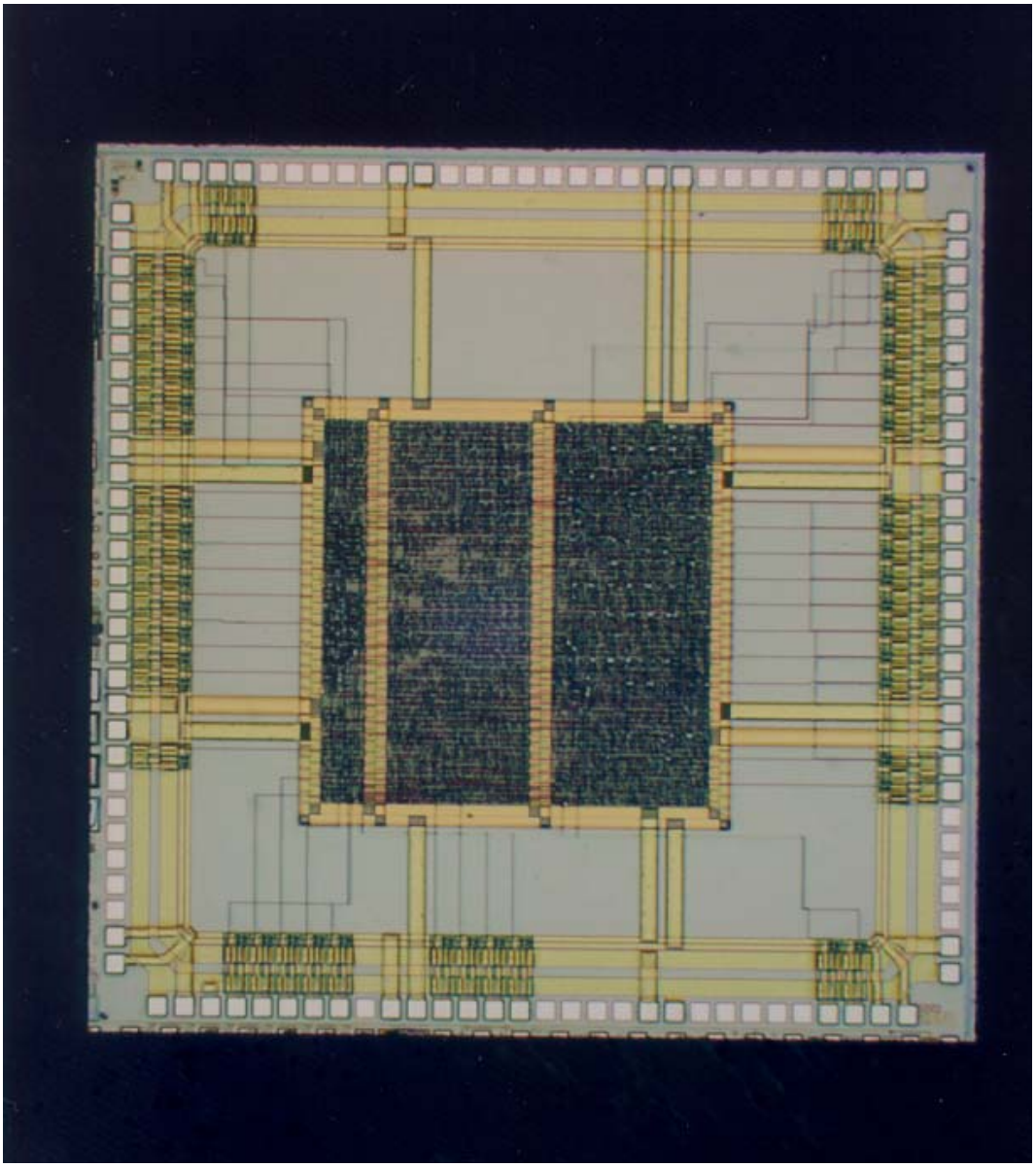

図 49: 製作した IC の顕微鏡写真。

4.7 動作試験のセットアップ

実装されたサンプル IC の動作試験を行なった。

動作試験では、ASIC への入力データ信号とトリガー信号をさまざまなタイミングで入力し、正しくヒットの有無を判断するかどうかを試験した。また、40MHz の動作クロックで試験を行なった。

図 50 に動作試験のセットアップを示す。

以下に図 50 中のいくつかのモジュールについて説明する。

PC VME の制御は、PCI-VME インターフェースを通して PC から行なった。システム OS は、Linux である。

PPG Pulse Pattern Generator。6U の VME ボードである。VME から書き込むことのできる 16 ビット幅、深さ 4096 の DATA RAM をもち、この RAM 内のビット・パターンを 16 チャンネルの ECL レベル・パルス・パターンにして出力する。クロックは NIM レベルで入力する。

PPG CLOCK CONTROLER PPG クロック入力用に開発されたクロック・ジェネレータで、6U の VME ボードである。周波数は 0.4MHz から 0.4MHz ステップで周波数は、最大 250MHz までと、その 10、100、…、100000 分の 1 の値に設定できる。また、必要な数だけのクロックを出力することも可能であり。クロックの数は、$2^{16}(= 65536)$ 個まで設定可能である。もちろんクロックの数を制限しないで動作することも可能である。出力は NIM レベルであり、周波数、クロック数の設定は、VME から行なう。

CLOCK FANOUT CLOCK CONTROLER の出力クロックを同期を保ったまま複数の PPG に分配するための VME6U ボードで、1 入力 16 出力である。

DELAY MODULE 250ps ごとに最大 48.750ns まで遅延を設定することのできる VME6U、NIM レベルのディレイ・モジュールである。遅延値の設定は、VME から行なう。

INTERRUPT REGISTER 8 ビットのインタラプト・レジスタ機能と I/O レジスタ機能を持つ 6U の VME モジュール。ここでは、アウトプット・レジスタの機能を用いた。VME からのアクセスのタイミングで NIM レベルのパルスを出力する。

動作試験は、テスト・ボードを製作して行なった。ASIC の入出力信号はすべて TTL レベルであるが、CLOCK、TRIGGER、RESET_、RESETIC_の制御信号は、NIM レベルで発生するのが便利である。そこで、テスト・ボードには、4 チャンネルの NIM to TTL コンバータを搭載し、CLOCK、TRIGGER、RESET_、RESETIC_の 4 信号を NIM レベルで受信して ASIC に入力するために使用した。また、GATE WIDTH をディップ・スイッチで設定し、POL、MOD、その他各種許可信号の動作パラメータをジャンパー・スイッチで設定することを可能にした。

入力データ IN0~IN15 と TRIGGER 信号それぞれを任意のタイミングで発生させるために 1 台ずつ PPG を用意した。これらの PPG は、PPG CLOCK CONTROLER で発生した 40MHz のクロックで動作させた。したがって PPG による信号のタイミング制御は、25ns ごとのものである。TRIGGER 信号のタイミングに関しては、さらに DELAY MODULE を使用することで 250ps ごとに制御した。RESET_、RESETIC_のリセット信号は、INTERRUPT REGISTER によって発生した。

実際の MWPC からの出力信号は、読み出しシステムのクロックとは非同期に発生するので、この動作試験でも PPG CLOCK CONTROLER とは別に ASIC の動作クロック用にクロック・ジェ

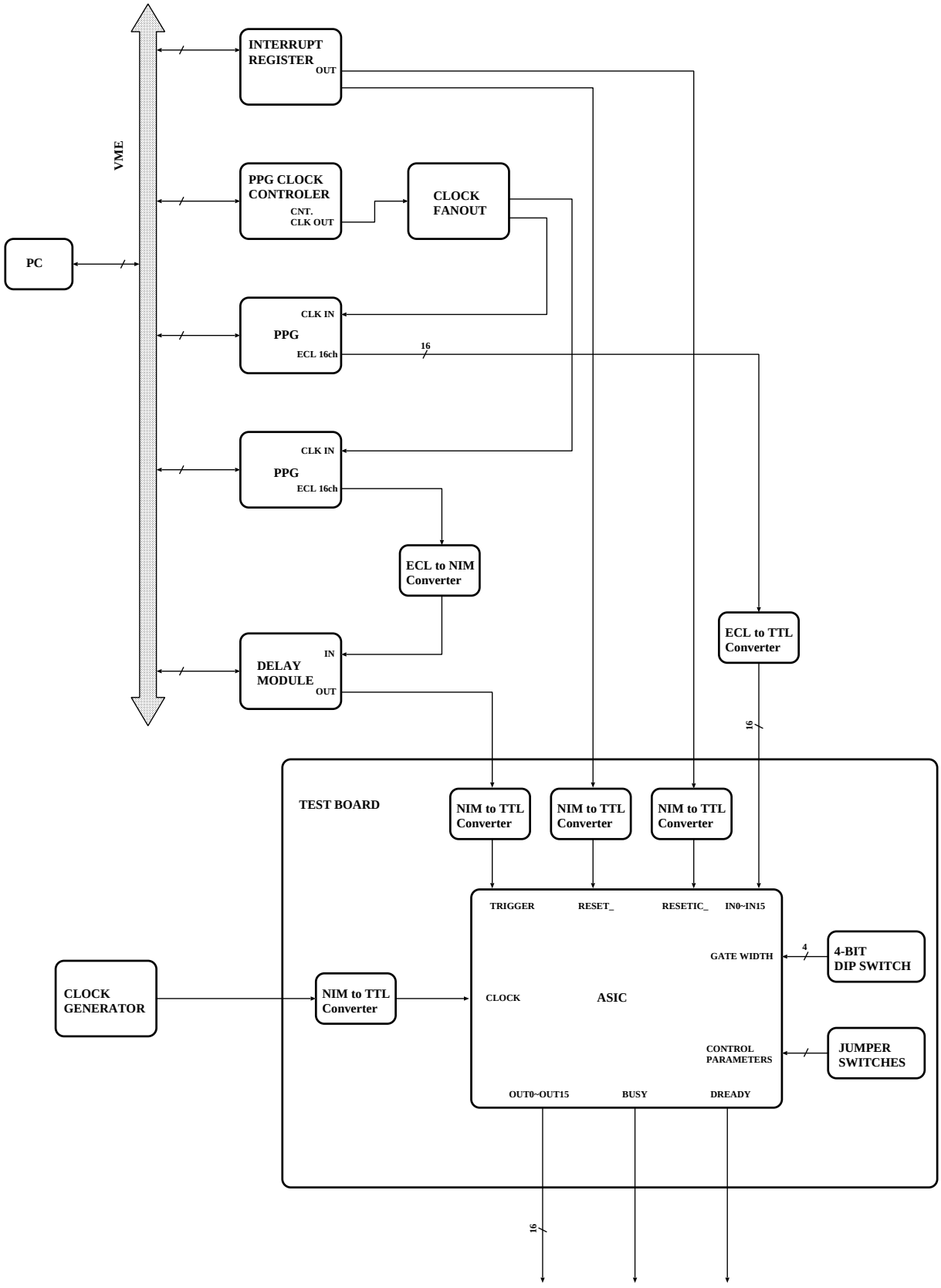

図 50: 動作試験のセットアップ。

ネレータを用意して入力データを読み出しシステムと非同期にした。なお、この動作試験は、ASICのクロックを 40MHz に設定して行なった。

4.8 動作試験の結果

4.8.1 タイミング・チャート

入力の極性を +、シフト・レジスタの出力を IC の出力に設定し、ゲート幅を 6 クロック分に設定したときにロジック・アナライザで測定したタイミング・チャートを図 51 に示す。100 クロック (2500ns) 分の遅延が生じており、シフト・レジスタが正しく動作していることが確認できる。

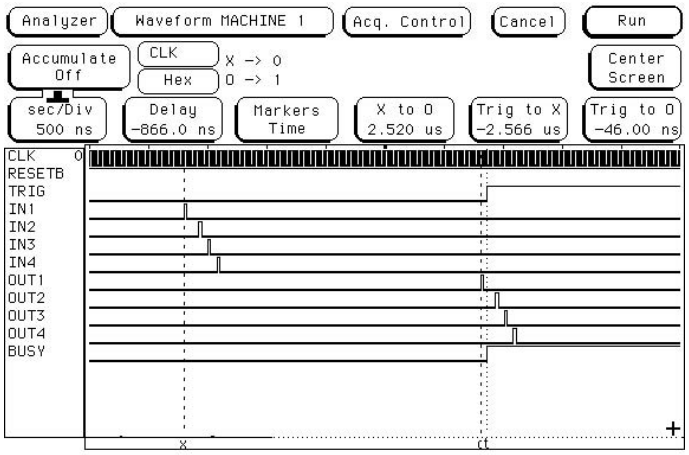

図 51: シフト・レジスタの出力を IC の出力に設定したときのタイミング・チャート。1 チャンネルの入力信号の立ち上がり (左の破線) と 1 チャンネルの出力信号の立ち上がり (右の破線) との間の時間間隔が図中の “X to O” のウィンドウに表示されている。

このタイミング・チャートの、出力信号の立ち上がり部分を拡大したものが図 52 である。トリガー入力から 2 つ目のクロックの立ち上がりエッジから 6 クロック分の時間、ゲートが開く。この様子がカーソル (破線) で示されている。

図 52 では、ゲートの間に OUT2、OUT3 の立ち上がりがある。同じセットアップで IC からヒット・パターンを出力させたものが図 53 である。シフト・レジスタの出力の立ち上がりがゲートの間にあるチャンネルでリセット信号のタイミングまで出力が High に保持されている。

IC のクロック周波数を 10MHz から大きくしながら測定してみたところ、74MHz まで図 51、図 52 および図 53 のタイミング・チャートの回路動作が得られ、これより大きいクロック周波数では出力信号は一切出力されなかった。したがって、上限の動作クロック周波数は 74MHz である。

また、シフト・レジスタの出力、ヒット情報それぞれの出力モードの出力波形を図 54、図 55 に示す。

4.8.2 システムの検出効率

図 53 のように、入力データ、トリガー信号を入力し、その後しばらく経った後にリセットを入力するという操作を繰り返し、IC からの出力をスケーラで数えた。こうして測定した IC からの

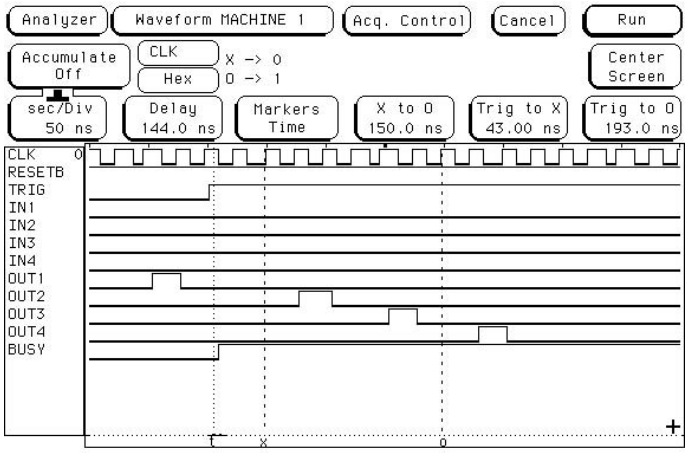

図 52: シフト・レジスタの出力とゲートのタイミング。カーソル (破線) はゲートが開いている時間を示している。

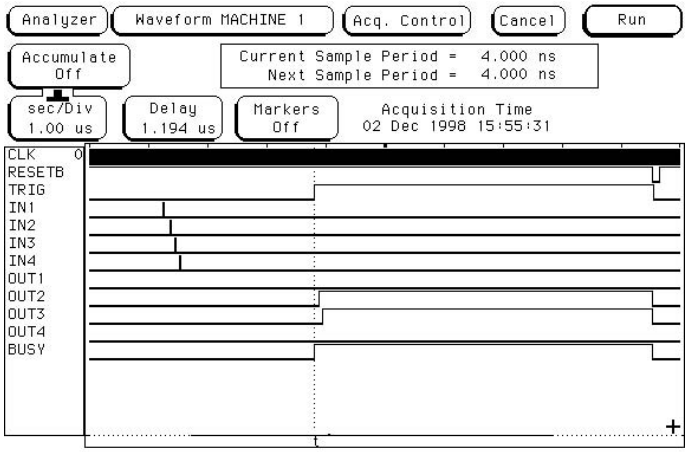

図 53: ヒット・パターンを IC の出力に設定したときのタイミング・チャート。

出力の回数と入力データ、トリガー信号を入力した回数との比をとり、システムの検出効率を計算した。ゲート幅を 4 クロック (100ns) に固定し、データ入力のタイミングからトリガー入力のタイミングまでの時間差を変化させながら検出効率を測定した結果が図 56 である。

入力データとトリガーの時間差が 2400 ～ 2500(ns) の間は検出効率が高いが、これはこの範囲の時間差の場合にはゲートが開いている間にシフト・レジスタの出力が立ち上がるからである。

ただし、検出効率が高い範囲の時間差に設定した場合でも検出効率は 100% を達成していない。また、シフト・レジスタの出力がゲートの開いている間に入らないはずの時間差の設定の場合にも 1% 程度の検出効率がある領域が存在している。これらの問題点は、次節で議論する。

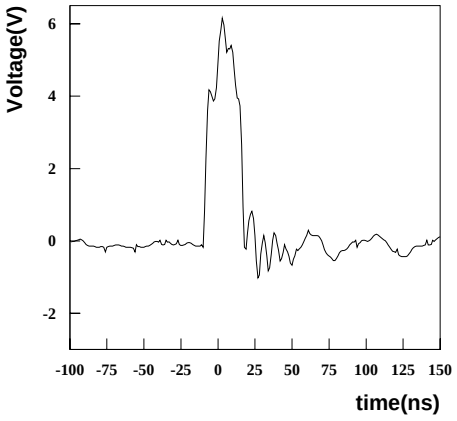

図 54: シフト・レジスタの出力の波形。

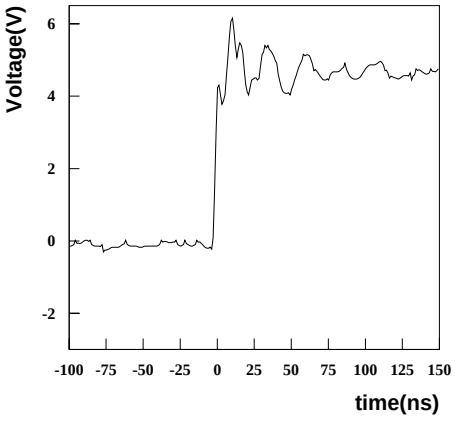

図 55: ヒット情報の出力波形。

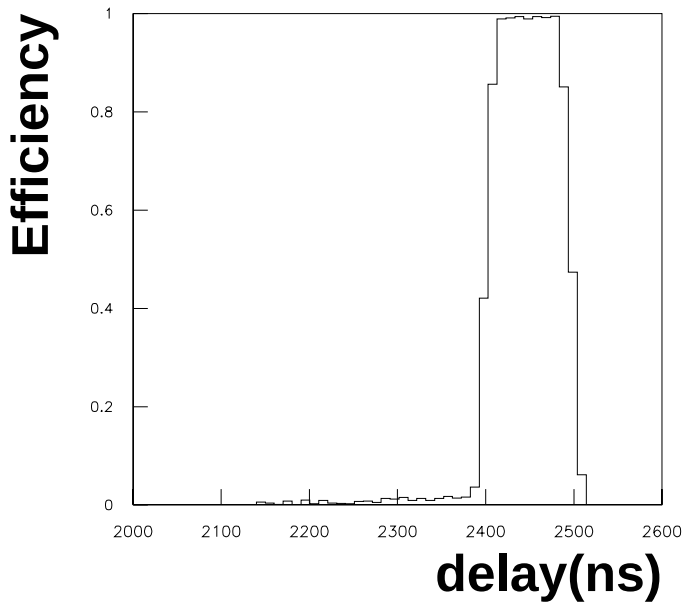

図 56: データ入力のタイミングからトリガー入力のタイミングまでの時間差 (delay) を変えた時の検出効率の変化。

4.8.3 問題点の検討

シフト・レジスタの出力をICの出力とするモードで測定を行なったが、ICに入力したパルスの数よりも出力されるパルスの数の方が少ない。まず、前節の問題を考慮するまえにこの原因を考察する。

このASICでは、MWPCからの入力信号をクロックに非同期に受信している。この部分の回路図を図57に示す。この部分で受信に失敗しているかどうかを確かめるためにセットアップを変更して測定を行なった。新しいセットアップでは、入力信号とクロックのタイミングのずれをいろいろと変えてみることで非同期・同期変換が正しく行なわれているかどうかを試験した。

図58に変更後のセットアップをしめす。

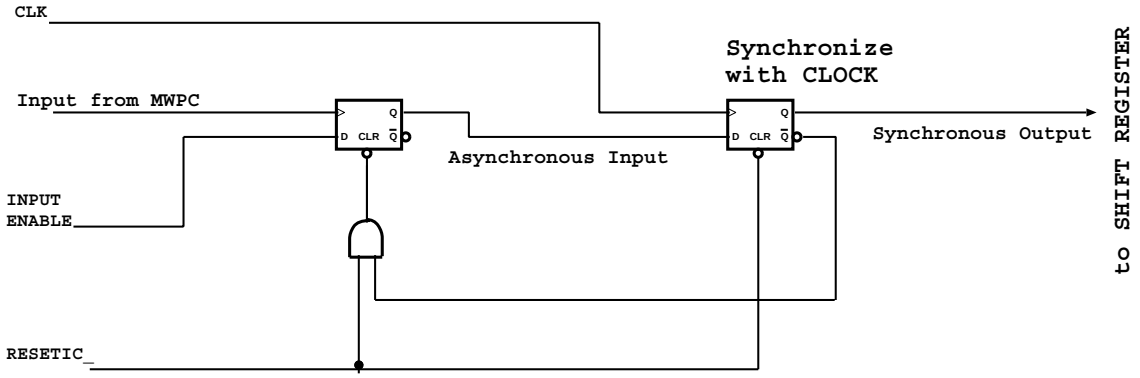

図57: MWPCからの信号をクロックと同期させる回路I。1段目のフリップ・フロップでの立ち上がりを検出し、2段目のフリップ・フロップでクロックと同期をとっている。また、2段目のフリップ・フロップの$\bar{q}$を1段目のフリップ・フロップのリセット入力CLRに入力することで1クロック分のパルスを出力する。

図50のセットアップでは、PPGでICへの入力データを生成するためのクロックとIC自体の動作クロックを別々に用意していた。ここでは、両方ともCLOCK CONTROLERのクロックをもちいる。ただし、ICのクロックはDELAY MODULEで位相をずらして入力した。これにより、入力データのタイミングとICのクロックのタイミングのずれを変化させながらシフト・レジスタの出力の振舞いを測定することで非同期から同期信号へ変換する部分のテストを行った。また、シフト・レジスタの出力を測定するので、トリガーは入力しなかった。

前節と同様にこのセットアップでDELAY MODULEの遅延時間設定値を250psごとに変化させながら検出効率(ICがデータ信号を受け付ける効率)を測定した結果が図59である。

この結果から、入力データのクロックに対するタイミングによっては、データを受け付けない不感時間があることがわかる。図56中、2400 ~ 2500nsの範囲で100%の検出効率を達成していない原因はこの入力データに対する不感時間であると考えられる。

そこで、図57の非同期・同期変換回路の論理合成結果のゲート構成をみてみると、2つのフリップ・フロップとも図60のようなゲート構成になっている。このゲートDFSRとWDFRP4の真理値表をそれぞれ図61と図62に示すが、ここでは、どちらも単なるフリップ・フロップとして使われている。すなわち、図57中のフリップ・フロップのq、$\bar{q}$出力は、べつべつのフリップ・フロップ・ゲートでクロックとの同期をとられている。とくに、図57中の2段目のフリップ・フロップに注目する。クロックの立ち上がりとD入力の立ち上がりのタイミング差が短い場合には、セット

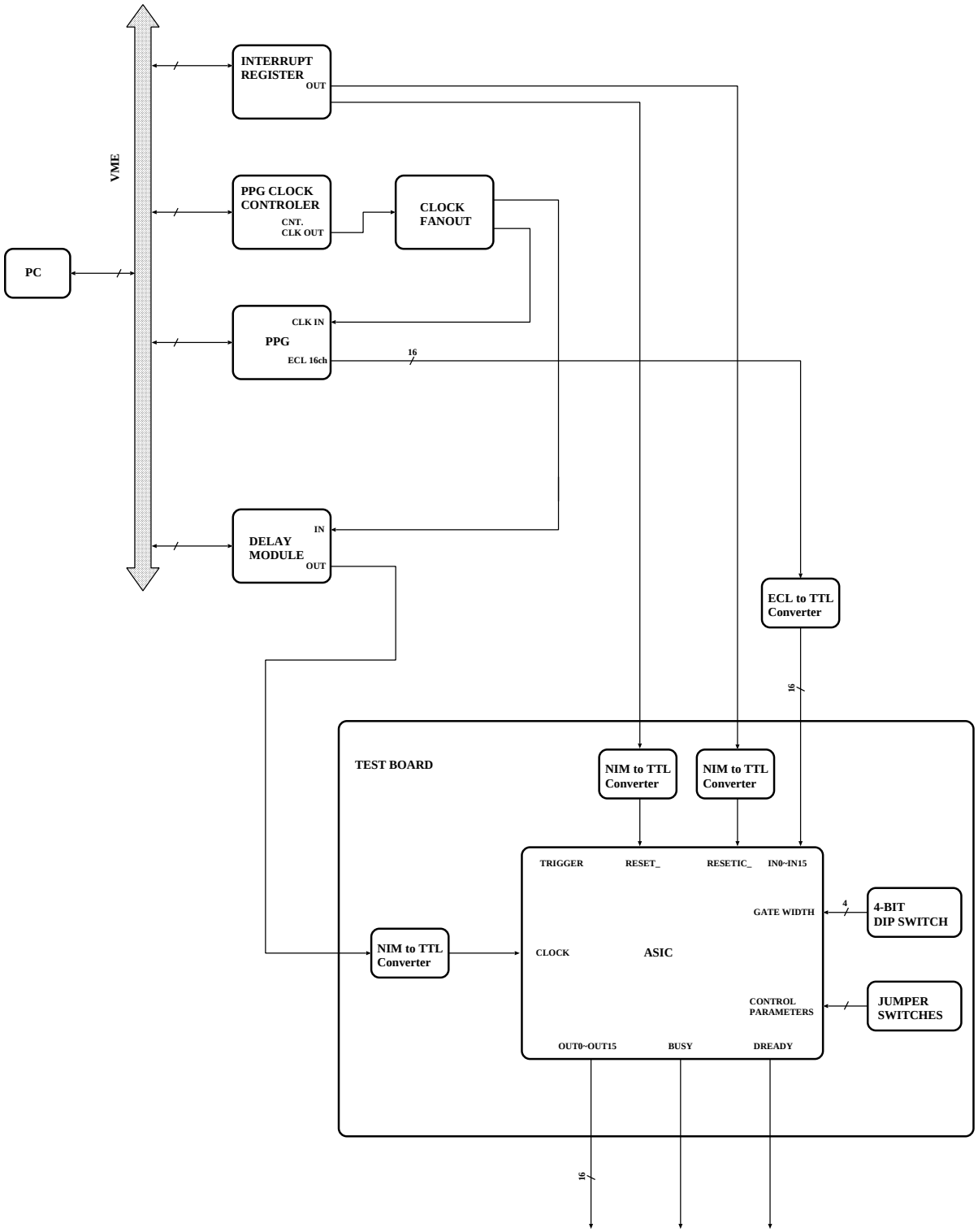

図 58: 動作試験のセットアップ II。

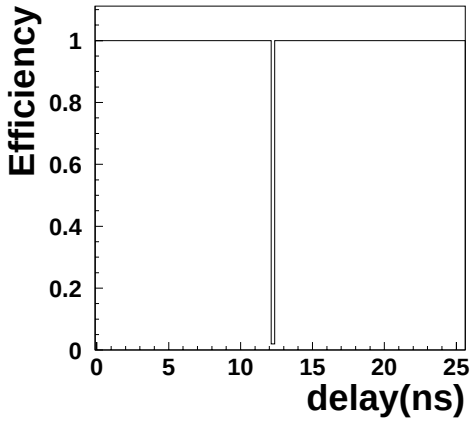

図 59: 入力データ信号と IC のクロックのタイミングのずれを変えたときのデータ受け付けの効率。

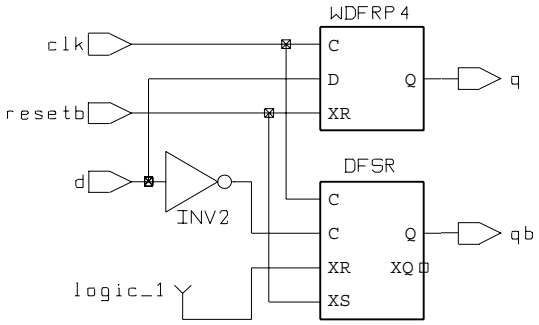

図 60: フリップ・フロップの論理合成結果。ここでは DFSR と WDRP4 は単なるフリップ・フロップとして使われている。なお、INV2 はインバータである。

アップ時間またはホールド時間が十分でないためにフリップ・フロップは D 入力を正しく読み取ることができず、したがって、q、$\bar{q}$ は正しく出力されない。とくに、図 60 のように q と $\bar{q}$ がべつべつのフリップ・フロップで出力されているので、q と $\bar{q}$ の出力が反転する保証はない。そこで、図 57 中の Asynchronous Input の立ち上がり直後のクロックで、q と $\bar{q}$ ともに Low になる場合もある。このとき、1 段目のフリップ・フロップはこの $\bar{q}$ を受けてクリアされてしまうので、つぎのクロック立ち上がりの後の 2 段目のフリップ・フロップの q 出力も Low のままである。結局、この場合には MWPC からの入力信号は立ち上がってもシフト・レジスタへの入力パルスは出ない。

ここでの問題の原因は、フリップ・フロップの q と $\bar{q}$ の 2 出力を別々のフリップ・フロップ・ゲートで作っていることである。そこで、この問題は、フリップ・フロップのゲート構成を図 63 のようにすれば解消できるはずである。

C	D	XR	XS	Q	XQ
X	X	0	0	0	0
X	X	1	0	1	0
X	X	0	1	0	1
1->0	X	1	1	hold	hold
0->1	0	1	1	0	1
0->1	1	1	1	1	0

図 61: DFSR の真理値表。

C	D	XR	Q
X	X	0	0
1->0	X	1	hold
0->1	0	1	0
0->1	1	1	1

図 62: WDFRP4 の真理値表。

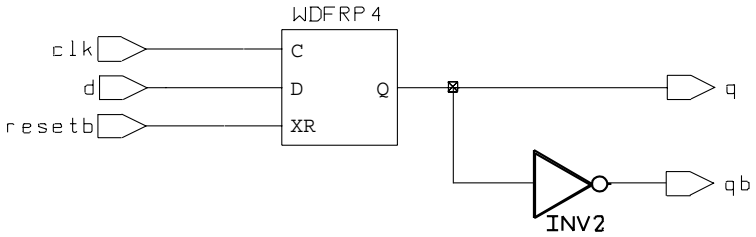

図 63: フリップ・フロップのゲート構成をこのようにすれば非同期・同期変換回路のバグは修正できるはずである。

つぎに、図 56 中、2100 ～ 2400(ns) の範囲で 1% 程度の検出効率があることについて考察する。小さくとも検出効率がある領域は全体で 15 クロック分程度である。一方、ゲート幅は 4 ビットの入力信号で設定するが、回路を簡単にするために設定値は 1 ～ 15 クロックにしてある。したがって、ゲート幅の設定に失敗していることが考えられる。

そこで、ゲート信号を作り出す際にクロックを数えるカウンタの構造を考察する。ゲート信号を作り出す際にクロックを数えるカウンタは、COUNTER ENABLE 信号 (トリガー入力のタイミングで立ち上がり、リセット入力まで High のレベルを保持する信号。クロックとは非同期) が Low のあいだに 16 から GATE WIDTH(4 ビットの信号で、0 ～ 15 を表現) を引いた値をロードし、COUNTER ENABLE が High になるとクロックを数えるものである。このカウンタの概念図を図 64 に示す。カウンタは、4 つのフリップ・フロップをレジスタとして用意している。また、このフリップ・フロップの出力に 1 加える操作を行なう回路がある。フリップ・フロップの D 入力には COUNTER ENABLE 信号のレベルにより、GATE WIDTH を論理反転させた信号 (COUNTER ENABLE=Low) またはカウンタの値に 1 加えた値 (COUNTER ENABLE=High) が入力される。

COUNTER ENABLE 信号の立ち上がり直後、セレクタの出力がまだ安定しないうちにクロックの立ち上がりがあると、フリップ・フロップは正しく入力信号を読み取れない。このため、クロックの立ち上がりの直前に COUNTER ENABLE 信号の立ち上がりがあるとカウンタの値に間違った値が入ってしまう。この結果、設定した値どおりのゲート幅が得られなくなってしまう。

このカウンタの問題は、COUNTER ENABLE 信号をクロックと同期の信号としてカウンタに入力すれば解消できるはずである。この同期をとる回路の回路図を図 65 に示す。

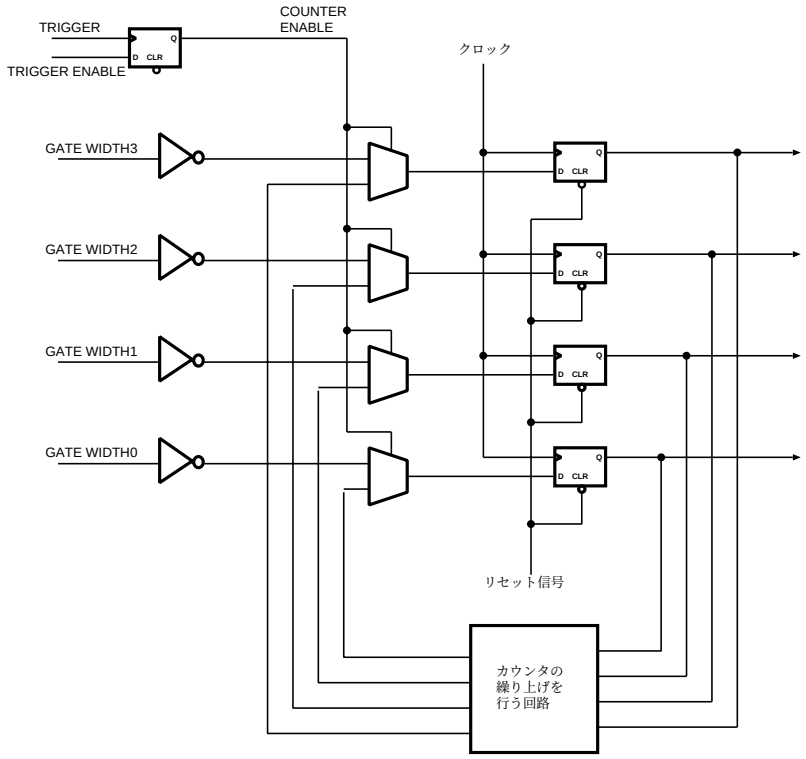

図 64: コントロール部のカウンタの概念図。

この他の問題としては、データのレディー信号 DATA READY がまったく出力されなかった。まず、図 66 にこの DATA READY 信号を作り出している部分の回路図を示す。

DATA READY が出力されなかった原因を考えるまえに、設計過程について考えてみる。ゲート・レベル・シミュレーションでは、論理合成結果のネット・リストが仕様を正しく実現していることを検証した。また、レイアウト検証では、LVS(Layout vs Schematic) を行ない、レイアウト結果と論理合成結果のネット・リストが同じ結線になっていることを確認した。これらのことから、もし設計に失敗しているとすれば、それはレイアウト配線遅延によるものであるといえる。

ところで、DATA READY 信号の場合には、リセット時以外は GATE 信号のタイミングのみで制御されるので、配線遅延は問題にならない。さらに GATE 信号は確かに出力されてことがわかっている。

これらのことから、設計工程には問題がなかったことがわかるが、ここで使ったツール、ライブラリにバグがあることは十分に考えられる。また、これらのツール、ライブラリを使用する際の使用方法に問題があることも十分に考えられる。

最後に、全 16 チャンネルのうち、1 チャンネルの出力が DATA READY 信号と同様にまった

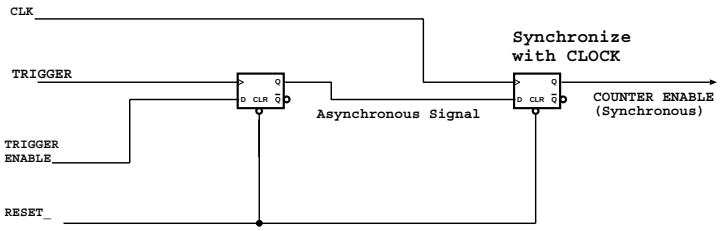

図 65: COUNTER ENABLE 信号をクロックと同期させる。1 段目のフリップ・フロップで TRIGGER の立ち上がりを検出し、2 段目のフリップ・フロップでクロックと同期をとっている。2 段のフリップ・フロップともリセットが入力されるまで High の状態を保持する。

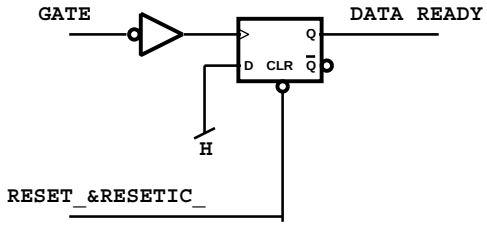

図 66: DATA READY 信号を発生する回路。ゲートの立ち下がりを検出し、リセット入力まで DREADY を High にドライブする。

く出力されなかった。この部分の回路図は、図 47 を見てほしい。これは、ヒット情報を出力するモードでもシフト・レジスタの出力を IC の出力とする場合でも同じであった。また、このチャンネルのみにデータを入力した場合も、OR OUT 信号の出力は得られたので、これはシフト・レジスタへの入力からシフト・レジスタ出力までの間か、あるいは出力を選択するセレクタ以後出力ピンまでのどこかで問題が起こっていることになる。一方、この場合もゲート・レベルのシミュレーション、LVS ではともにエラーがない。そこで、この原因が設計工程でのミスであるならば、それは、配線遅延が許容されないくらい大きいためである。とくに、遅延のために問題が起こり得る部分はシフト・レジスタの出力までの部分である。ところが、レイアウト後にレイアウトツールが出力した各ゲート間の配線遅延時間を調べたが、レジスタ間の配線遅延は 25ns のクロック周期に対してどれも 1ns 以下と小さく、問題となるものではなかった。

このことから、1 チャンネルの出力が得られなかった原因も使用したツール、ライブラリのバグ、またはこれらを使用する際の使用方法の問題であると考えられる。

4.8.4 シフト・レジスタの動作

シフト・レジスタの出力を測定したタイミング・チャート (図 51) をみると、入力信号が IC の中で 101 クロック分送れて出力されていることがわかる。図 47 のように IC の中では、100 段のシフト・レジスタの前に、もう 1 段フリップ・フロップがある。したがって、シフト・レジスタでの遅延時間は仕様どおりであり、100 段のシフト・レジスタは正しく実現できた。

4.8.5　まとめ

回路系は 10 ～ 74MHz まで動作した。100 段のシフト・レジスタが正しく動作した。設計の問題点は、入力データ受信部の非同期・同期変換にデッド・タイムが出てしまったこと、やはり非同期信号のためにゲート信号を作るためのカウンタが誤動作してしまったことであり、これらにたいしては解消法を考察した。このほか、データのレディー信号と 1 チャンネル分の出力が得られなかった。

5　まとめと今後の予定

TGC 読み出しシステムでは効率良くデータを読み出すためにデランドマイザの出力を圧縮・シリアル変換してスター・スイッチに集積した後に ROD にデータを送信する。ここでは、レベル 1・バッファからスター・スイッチへの読み出しのプロトコルを考察した。また、シミュレーションを行ない、デランドマイザ内のデータ量の振舞いを理解し、データの棄却率が許容範囲に収まるために必要なデランドマイザのサイズを評価した。また 、verilogHDL でシステムを設計した。さらに、論理シミュレーションを行ない、回路が仕様どおりの機能を実現していることを検証した。

また、TGC の読み出しシステムでは、その生産量、放射線耐性への要請から、大部分の論理回路を ASIC でに実装する。したがって、TGC の読み出しシステムを開発していく上で、ASIC の設計手法を理解することは非常に重要な課題である。とくに、ここでは、設計の全工程を行ない、ASIC への実装のための論理回路の設計手法への理解を深めた。ここでは、TGC 読み出しシステムの一部の回路を含み、TGC の読み出しを行なうこともできる汎用性、16 チャンネルの MWPC 読み出し用 ASIC を試作し、その性能評価を行なった。性能評価の結果は、以下のとおりであった。

- IC は 74MHz のクロック周波数まで動作した。
- TGC 読み出しシステムの一部である 100 段のシフト・レジスタが正しく機能した。
- 入力信号の非同期・同期変換を行なう回路が正しく機能せず、このために MWPC からの信号入力に対して不感時間がある。
- 非同期トリガー入力を受けてゲートを作り出す回路が誤動作をすることがある。
- 出力のレディー信号と 1 チャンネル分の出力が得られなかった。

設計上の問題点は、その原因を考察し、またその回避策を考察した。

今後の課題としては、まず、ここでおこなった TGC 読み出しシステムの HDL 記述を実際に FPGA に実装することでプロトタイピングを行なう。最終的には、スレーブ・ボードおよび High-P_T・ボードの回路は ASIC に実装する予定である。スター・スイッチの回路は、放射線耐性のよい FPGA または ASIC に実装する予定である。

MWPC 読み出し用 ASIC に関しては、MWPC からの入力信号を LVDS レベル[9] で受信できるよう、LVDS レシーバを搭載したチップを実用化する予定である。

[9]Low Voltage Differential Signaling。1.2V を中心とし、振れ幅 0.3V の信号レベル。

A　TGC 読み出しシステムの verilogHDL ソース・コード

A.1　スレーブ・ボード の読み出し回路の verilog ソース・コード

3.8.2 節で議論したスレーブ・ボード の読み出し回路の verilogHDL によるソース・コード を以下に示す。

```
module paraseri(sl_sync, sl_data, emptyb, full, hasdata, read0, read1, cell0,
 cell1, sw_sync, sw_data, clk, hdc);
//デランドマイザの出力をシリアル変換する。
  input emptyb, full,  hasdata, read0, read1;
  input sw_sync, sw_data;
  input [7:0] cell0, cell1;
  input clk;
  output sl_sync, sl_data;
  output hdc;
  assign hdc=0;
//8bits のターミネータおよびゼロ。
  wire [7:0] term, null;
//1bit の High および Low。
  wire onbit, offbit;
  wire [5:0] wire0, wire1;
  wire loadcom;

  assign term=8'b11111111;
  assign null=8'b00000000;
  assign onbit=1'b1;
  assign offbit=1'b0;

  command_decode decoder0(.loadcom(loadcom), .sw_sync(sw_sync),
 .sw_data(sw_data), .clk(clk));

// sl_sync をつくるシフト・レジスタの列。
    //パケットのヘッダ。
    reg_cell sync_head(sl_sync, onbit, wire1[0], loadcom, clk);
    reg_cell sync_empty(wire1[0], offbit, wire1[1], loadcom, clk);
    reg_cell sync_full(wire1[1], offbit, wire1[2], loadcom, clk);
    cell_8bits sync_cell0(wire1[2], null, wire1[3], loadcom, read0, clk);
    cell_8bits sync_cell1(wire1[3], null, wire1[4], loadcom, read1, clk);
    cell_8bits sync_term(wire1[4], null, wire1[5], loadcom, hasdata, clk);
    //パケットのフッタ。
    reg_cell sync_foot(wire1[5], onbit, offbit, loadcom, clk);

// sl_data をつくるシフト・レジスタの列。
    //パケットのヘッダ。
    reg_cell data_head(sl_data, onbit, wire0[0], loadcom, clk);
    //デランドマイザのエンプティー・フラグ。
    reg_cell data_empty(wire0[0], emptyb, wire0[1], loadcom, clk);
    //デランドマイザのフル・フラグ。
    reg_cell data_full(wire0[1], full, wire0[2], loadcom, clk);
    //セル 0 の読み出し。
    cell_8bits data_cell0(wire0[2], cell0, wire0[3], loadcom, read0, clk);
    //セル 1 の読み出し。
    cell_8bits data_cell1(wire0[3], cell1, wire0[4], loadcom, read1, clk);
```

```
    //データ・ターミネータ。
    cell_8bits data_term(wire0[4], term, wire0[5], loadcom, hasdata, clk);
    //パケットのフッタ。
    reg_cell data_foot(wire0[5], offbit, offbit, loadcom, clk);
endmodule


module command_decode(loadcom, sw_sync, sw_data, clk);
//スター・スイッチからのデータ送信コマンドを検出し、1クロック分のパルスを出
//力する。
  input sw_sync, sw_data;
  input clk;
  output loadcom;
  reg sw_sync1, sw_sync2, sw_data1, sw_data2;

  assign loadcom=(sw_sync2&sw_data2)&(!sw_sync1&!sw_data1)&(sw_sync&!sw_data);
  always @(posedge clk) begin
    sw_sync1 <= sw_sync;
    sw_data1 <= sw_data;
    sw_sync2 <= sw_sync1;
    sw_data2 <= sw_data1;
  end
endmodule


module cell_8bits(q, load, d, loadcom, read, clk);
//8ビットのセルの読み出し回路。
//readの値が1ならこのセルを読み出し、0ならばこのセルの読み出しをスキップする。
  input d;
  input [7:0] load;
  input loadcom, read;
  input clk;
  output q;
  wire qreg;

  reg_cell_8bits reg_cell_8bits0(qreg, load, d, loadcom, clk);
  selector selector0(q, qreg, d, read, loadcom, clk);
endmodule


module reg_cell_8bits(q, load, d, com, clk);
//8ビットのシフト・レジスタ。reg_cellを8個つなぎあわせて実現している。
  input d;
  input [7:0] load;
  input com;
  input clk;
  wire [6:0] node;
  output q;

  reg_cell reg_cell0(node[0], load[0],       d, com, clk);
  reg_cell reg_cell1(node[1], load[1], node[0], com, clk);
  reg_cell reg_cell2(node[2], load[2], node[1], com, clk);
  reg_cell reg_cell3(node[3], load[3], node[2], com, clk);
  reg_cell reg_cell4(node[4], load[4], node[3], com, clk);
```

```
  reg_cell reg_cell5(node[5], load[5], node[4], com, clk);
  reg_cell reg_cell6(node[6], load[6], node[5], com, clk);
  reg_cell reg_cell7(      q, load[7], node[6], com, clk);
endmodule


module reg_cell(q, load, d, com, clk);
//レジスタ。comが 1 のときに loadをレジスタにロードし、comが 0 のときには、dをレ
//ジスタにロードする。
  input d, load;
  input com;
  input clk;
  output q;
  reg q;

  always @(posedge clk)
    case(com)
      0: q <= d;
      1: q <= load;
    endcase
endmodule


module selector(q, d0, d1, com, loadcom, clk);
//loadcomのタイミングで comをレジスタにロードし、
//この値によって出力 qを d0あるいは d1に接続する。
  input clk;
  input d0, d1;
  input com;
  input loadcom;
  output q;
  reg com_reg;

  assign q = (com_reg&d0)||(!com_reg&d1);
  always @(posedge clk)
    if (loadcom == 1) com_reg <= com;
endmodule
```

A.2 スター・スイッチのシークエンサの verilog ソース・コード

3.8.3 節で議論したスター・スイッチのシークエンサの verilogHDL によるソース・コードを以下に示す。

```
module control_inc(write, data, sw_sync, sw_data, addr, sl_sync, sl_data, clk,
 status, reset, hdc);

  ‘define END_ADDR 3
  input clk;
//スレーブからの SYNC および DATA 信号。
  input sl_sync, sl_data;
//スレーブへの SYNC および DATA 信号。
  output sw_sync, sw_data;
  wire sw_sync, sw_data;
//スレーブへのデータ送信コマンド開始。
```

```
  wire ask;
//スター・スイッチ内 FIFO への書き込み許可。
  output write;
  wire write;
//スター・スイッチ内 FIFO への書き込みデータ。
  output data;
  reg data;
//接続するスレーブのアドレス。
  output [3:0] addr;
  wire [3:0] addr;
//シークエンサの状態。
  output [3:0] status;
  wire [3:0] status;
//シークエンスのリセット。
  input reset;
//スレーブからのパケットの開始。
  reg sl_start;

  output hdc;
  assign hdc=0;

  control control0(write, ask, addr, sl_sync, sl_data, clk, status, reset);
  send send0(reset, ask, sw_sync, sw_data, clk);
//write 信号にタイミングを合わせるために FIFO への書き込みデータを 1 クロック遅ら
//せる。
  always @(posedge clk)
    data <= sl_data;
endmodule


module control(write, ask, addr, sl_sync, sl_data, clk, status, reset);
//シークエンサ本体。
  ‘define END_ADDR 3
  input clk;
  input sl_sync, sl_data;
  output ask;
  reg ask;
  output write;
  reg write;
  output [3:0] addr;
  reg [3:0] addr;
  output [3:0] status;
  reg [3:0] status;
  input reset;
  reg sl_start;

  always @(posedge clk) begin
//スレーブからのパケットの開始を検出。
    sl_start <= sl_sync&sl_data;

    if (reset==1) status <= 0;
    else
      case(status)
//シークエンス・スタート
```

```
        0: begin
//シークエンスの初期化。
            addr <= 0;
            ask <= 0;
            write <= 0;
            status <= 1;
          end

//はじめのスレーブへのデータ送信コマンドを送信。
        1: begin
            ask <= 1;
            status <= 4'b0010;
          end

//はじめのスレーブのデランドマイザの情報を取得。
        2: begin
            ask <= 0;
//デランドマイザにデータが無い場合。
            if ((sl_start==1)&&(sl_data==0))
              status <= 1;
//デランドマイザにデータがある場合。
            if ((sl_start==1)&&(sl_data==1))
              status <= 3;
          end

        3: status <= 4;

//スレーブ内のデータの有無の情報を取得。
        4: begin
//スレーブに読み出しデータがない場合。
            if ((sl_sync==1)&&(sl_data==0)) begin
//もう読み出すべきスレーブが無い場合はシークエンスのはじめに戻る。
              case(addr)
                `END_ADDR: begin
                  status <= 0;
                end
//つぎのスレーブを読み出しに移る。
                default: begin
                  ask <= 1;
                  addr <= addr+1;
                  status <= 2;
                end
              endcase
            end
//スレーブに読み出しデータがある場合。
            else begin
//スター・スイッチ内の FIFO へ書き込み許可信号を送信。
              write <= 1;
              status <= 5;
            end
          end

//スレーブからのデータをスター・スイッチ内の FIFO に格納。
        5: begin
```

```
//スレーブからのパケット終了を検出。
            if ((sl_sync==1)&&(sl_data==0)) begin
              write <= 0;
              case(addr)
//もう読み出すべきスレーブが無い場合はシークエンスのはじめに戻る。
                `END_ADDR: begin
                  status <= 0;
                end
//つぎのスレーブを読み出しに移る。
                default: begin
                  addr <= addr+1;
                  ask <= 1;
                  status <= 2;
                end
               endcase
            end
           end

      endcase

  end
endmodule


module send(reset, ask, sw_sync, sw_data, clk);
//ask 信号が High になるタイミングで、スレーブに対するデータ送信コマンドを出力。
  input reset;
  input clk;
  input ask;
  reg prevask;
  output sw_sync, sw_data;
  reg sw_sync, sw_data;
  reg [1:0] status;

  always @(posedge clk) begin

    if (reset==1) status <= 0;
    else begin
     case(status)
      0: begin
//ask 信号の立ち上がりを検出。
        if ((ask==1)&&(prevask==0)) begin
//パケット開始
          sw_sync <= 1;
          sw_data <= 1;
          status <=1;
        end
        else begin
          sw_sync <= 0;
          sw_data <= 0;
        end
      end
      1: begin
          sw_sync <= 0;
```

```
          sw_data <= 0;
          status <=2;
      end
      2: begin
//パケット終了
          sw_sync <= 1;
          sw_data <= 0;
          status <=3;
      end
      3: begin
          sw_sync <= 0;
          sw_data <= 0;
          status <=0;
      end
     endcase
    end
    prevask <= ask;

  end
endmodule
```

B　MWPC 読み出し用 ASIC の verilogHDL ソース・コード

MWPC 読み出し用 ASIC の verilogHDL ソース・コードをいかに示す。

```
//IC 全体。
module wholeIC(QOUT, OROUT, BUSY, DREADY, CLK, IN, INENA, POL, TRIG, TRIGENA,
 GW, MOD, RESETB, OUTENAB, RESETICB);

//出力データ
  output [15:0] QOUT;
  output OROUT, BUSY, DREADY;
//入力データ
  input  [15:0] IN;
  input CLK, INENA, POL, TRIG, TRIGENA, MOD, RESETB, OUTENAB, RESETICB;
//ゲート巾
  input [3:0] GW;
  wire [15:0] monitor;
  wire GATE;

  wire CLK1, CLK2, CLK3, CLK4, CLK5;
  wire RESETICB1, RESETICB2, RESETICB3, RESETICB4, RESETICB5;
  wire GATE1, GATE2;
  wire INENA1, INENA2;

  wire DREADY0, BUSY0;
  wire [15:0] IN0, QOUT0;
  wire [3:0] GW0;

//ファンアウトの多い信号をバッファしている。
  buffer clkbuf1(CLK1, CLK);
  buffer clkbuf2(CLK2, CLK);
  buffer clkbuf3(CLK3, CLK);
  buffer clkbuf4(CLK4, CLK);
  buffer clkbuf5(CLK5, CLK);
  buffer rsticbuf1(RESETICB1, RESETICB);
  buffer rsticbuf2(RESETICB2, RESETICB);
  buffer rsticbuf3(RESETICB3, RESETICB);
  buffer rsticbuf4(RESETICB4, RESETICB);
  buffer rsticbuf5(RESETICB5, RESETICB);
  buffer gatebuf1(GATE1, GATE);
  buffer gatebuf2(GATE2, GATE);
  buffer inenabuf1(INENA1, INENA);
  buffer inenabuf2(INENA2, INENA);

//入力ピン
  buffer busybuf(BUSY,BUSY0);
  buffer drbuf(DREADY,DREADY0);
  buffer inbuf0(IN0[0],IN[0]);
  buffer inbuf1(IN0[1],IN[1]);
  buffer inbuf2(IN0[2],IN[2]);
  buffer inbuf3(IN0[3],IN[3]);
  buffer inbuf4(IN0[4],IN[4]);
  buffer inbuf5(IN0[5],IN[5]);
  buffer inbuf6(IN0[6],IN[6]);
  buffer inbuf7(IN0[7],IN[7]);
```

```
 buffer inbuf8(IN0[8],IN[8]);
 buffer inbuf9(IN0[9],IN[9]);
 buffer inbuf10(IN0[10],IN[10]);
 buffer inbuf11(IN0[11],IN[11]);
 buffer inbuf12(IN0[12],IN[12]);
 buffer inbuf13(IN0[13],IN[13]);
 buffer inbuf14(IN0[14],IN[14]);
 buffer inbuf15(IN0[15],IN[15]);
 buffer gwbuf0(GW0[0], GW[0]);
 buffer gwbuf1(GW0[1], GW[1]);
 buffer gwbuf2(GW0[2], GW[2]);
 buffer gwbuf3(GW0[3], GW[3]);

//出力ピン (IOバッファ)
 assign QOUT[0] = tribuff(OUTENAB, QOUT0[0]);
 assign QOUT[1] = tribuff(OUTENAB, QOUT0[1]);
 assign QOUT[2] = tribuff(OUTENAB, QOUT0[2]);
 assign QOUT[3] = tribuff(OUTENAB, QOUT0[3]);
 assign QOUT[4] = tribuff(OUTENAB, QOUT0[4]);
 assign QOUT[5] = tribuff(OUTENAB, QOUT0[5]);
 assign QOUT[6] = tribuff(OUTENAB, QOUT0[6]);
 assign QOUT[7] = tribuff(OUTENAB, QOUT0[7]);
 assign QOUT[8] = tribuff(OUTENAB, QOUT0[8]);
 assign QOUT[9] = tribuff(OUTENAB, QOUT0[9]);
 assign QOUT[10] = tribuff(OUTENAB, QOUT0[10]);
 assign QOUT[11] = tribuff(OUTENAB, QOUT0[11]);
 assign QOUT[12] = tribuff(OUTENAB, QOUT0[12]);
 assign QOUT[13] = tribuff(OUTENAB, QOUT0[13]);
 assign QOUT[14] = tribuff(OUTENAB, QOUT0[14]);
 assign QOUT[15] = tribuff(OUTENAB, QOUT0[15]);

 control control0(GATE, BUSY0, DREADY0, CLK1, TRIG, TRIGENA, GW0, RESETB,
RESETICB1);
 readout chan0(QOUT0[0], monitor[0], IN0[0], POL, INENA1, CLK2, GATE1,
RESETB, MOD, RESETICB2);
 readout chan1(QOUT0[1], monitor[1], IN0[1], POL, INENA1, CLK2, GATE1,
RESETB, MOD, RESETICB2);
 readout chan2(QOUT0[2], monitor[2], IN0[2], POL, INENA1, CLK2, GATE1,
RESETB, MOD, RESETICB2);
 readout chan3(QOUT0[3], monitor[3], IN0[3], POL, INENA1, CLK2, GATE1,
RESETB, MOD, RESETICB2);
 readout chan4(QOUT0[4], monitor[4], IN0[4], POL, INENA1, CLK3, GATE1,
RESETB, MOD, RESETICB3);
 readout chan5(QOUT0[5], monitor[5], IN0[5], POL, INENA1, CLK3, GATE1,
RESETB, MOD, RESETICB3);
 readout chan6(QOUT0[6], monitor[6], IN0[6], POL, INENA1, CLK3, GATE1,
RESETB, MOD, RESETICB3);
 readout chan7(QOUT0[7], monitor[7], IN0[7], POL, INENA1, CLK3, GATE1,
RESETB, MOD, RESETICB3);
 readout chan8(QOUT0[8], monitor[8], IN0[8], POL, INENA2, CLK4, GATE2,
RESETB, MOD, RESETICB4);
 readout chan9(QOUT0[9], monitor[9], IN0[9], POL, INENA2, CLK4, GATE2,
RESETB, MOD, RESETICB4);
 readout chan10(QOUT0[10], monitor[10], IN0[10], POL, INENA2, CLK4, GATE2,
```

```
 RESETB, MOD, RESETICB4);
  readout chan11(QOUT0[11], monitor[11], IN0[11], POL, INENA2, CLK4, GATE2,
 RESETB, MOD, RESETICB4);
  readout chan12(QOUT0[12], monitor[12], IN0[12], POL, INENA2, CLK5, GATE2,
 RESETB, MOD, RESETICB5);
  readout chan13(QOUT0[13], monitor[13], IN0[13], POL, INENA2, CLK5, GATE2,
 RESETB, MOD, RESETICB5);
  readout chan14(QOUT0[14], monitor[14], IN0[14], POL, INENA2, CLK5, GATE2,
 RESETB, MOD, RESETICB5);
  readout chan15(QOUT0[15], monitor[15], IN0[15], POL, INENA2, CLK5, GATE2,
 RESETB, MOD, RESETICB5);

  assign OROUT=|monitor[15:0];

function tribuff;
  input oeb;
  input d;
  case (oeb)
    1'b0: tribuff=d;
    1'b1: tribuff=1'bz;
  endcase
endfunction
endmodule


//読み出し部
module readout(OUT, MONITOR, IN, POL, INENA, CLK, GATE, RESETB, MOD, RESETICB);
  output OUT, MONITOR;
  input IN, POL, INENA;
  input CLK;
  input GATE, RESETB;
  input MOD;
  input RESETICB;
  wire xor0q, dff0q, dff0qb, dff1q, dff1qb, sregq, dff2q, dff2qb, and0q, sel0q;

  xor xor0 (xor0q, IN, POL);
  dff dff0 (dff0q, dff0qb, xor0q, INENA, dff1qb&&RESETICB);
  dff dff1 (dff1q, dff1qb, CLK, dff0q, RESETICB);
  buffer delay0(sregin, dff1q);
  shiftreg sreg0 (sreg0q, CLK, sregin,  RESETICB);
  and and0 (and0q, sreg0q, dff2qb);
  dff dff2 (dff2q, dff2qb, and0q, GATE, RESETB&&RESETICB);
  selector sel0 (sel0q, MOD, sreg0q, dff2q);
  assign OUT=sel0q;
  assign MONITOR=(~dff0qb)||(~dff1qb);
endmodule


//コントロール部
module control(GATE, BUSY, DREADY, CLK, TRIG, TRIGEN, GW, RESETB, RESETICB);
  input TRIG, TRIGEN, RESETB, RESETICB;
  input [3:0] GW;
  input CLK;
  output GATE, BUSY, DREADY;
```

```
  wire cnten, cutenb, high;

  assign high=1;

  dff dff0 (cnten, cntenb, TRIG, TRIGEN, RESETB&&RESETICB);
  dff dff1 (DREADY, , ~GATE, high, RESETB&&RESETICB);
  counter cnt0 (GATE, GW, cnten, CLK, RESETB&&RESETICB);

  assign BUSY=~cntenb;
endmodule


//100段シフト・レジスタ
module shiftreg(q, clk, d, resetb);
  `define REPEAT 9

  input  clk, d, resetb;
  output q;

  wire [`REPEAT:0] qsreg;
  wire [`REPEAT:0] clkbuf;
  wire [`REPEAT:0] rstbbuf;

  sreg sreg0(qsreg[0], clkbuf[0], d, rstbbuf[0]);
  buffer buf0(clkbuf[0], clkbuf[1]);
  buffer rstbuf0(rstbbuf[0], rstbbuf[1]);

  sreg sreg1(qsreg[1], clkbuf[1], qsreg[0], rstbbuf[1]);
  buffer buf1(clkbuf[1], clkbuf[2]);
  buffer rstbuf1(rstbbuf[1], rstbbuf[2]);

  sreg sreg2(qsreg[2], clkbuf[2], qsreg[1], rstbbuf[2]);
  buffer buf2(clkbuf[2], clkbuf[3]);
  buffer rstbuf2(rstbbuf[2], rstbbuf[3]);

  sreg sreg3(qsreg[3], clkbuf[3], qsreg[2], rstbbuf[3]);
  buffer buf3(clkbuf[3], clkbuf[4]);
  buffer rstbuf3(rstbbuf[3], rstbbuf[4]);

  sreg sreg4(qsreg[4], clkbuf[4], qsreg[3], rstbbuf[4]);
  buffer buf4(clkbuf[4], clkbuf[5]);
  buffer rstbuf4(rstbbuf[4], rstbbuf[5]);

  sreg sreg5(qsreg[5], clkbuf[5], qsreg[4], rstbbuf[5]);
  buffer buf5(clkbuf[5], clkbuf[6]);
  buffer rstbuf5(rstbbuf[5], rstbbuf[6]);

  sreg sreg6(qsreg[6], clkbuf[6], qsreg[5], rstbbuf[6]);
  buffer buf6(clkbuf[6], clkbuf[7]);
  buffer rstbuf6(rstbbuf[6], rstbbuf[7]);

  sreg sreg7(qsreg[7], clkbuf[7], qsreg[6], rstbbuf[7]);
  buffer buf7(clkbuf[7], clkbuf[8]);
```

```
  buffer rstbuf7(rstbbuf[7], rstbbuf[8]);

  sreg sreg8(qsreg[8], clkbuf[8], qsreg[7], rstbbuf[8]);
  buffer buf8(clkbuf[8], clkbuf[9]);
  buffer rstbuf8(rstbbuf[8], rstbbuf[9]);

  sreg sreg9(qsreg[9], clkbuf[9], qsreg[8], rstbbuf[9]);
  buffer buf9(clkbuf[9], clk);
  buffer rstbuf9(rstbbuf[9], resetb);

  assign q=qsreg[`REPEAT];
endmodule


//10段シフト・レジスタ
module sreg(q, clk, d, resetb);

  `define LENGTH 9

  input  clk, d, resetb;
  output q;

  reg [`LENGTH:0] sreg;

  always
    @ (posedge clk or negedge resetb)
      begin
        if (resetb==0)
          sreg <= 0;
        else
          begin
            sreg <= sreg << 1;
            sreg[0] <= d;
          end
      end

  assign q=sreg[`LENGTH];
endmodule


//コントロール部のカウンタ
module counter(GATE, GW, EN, CLK, CLRB);
  input [3:0] GW;
  input EN, CLK, CLRB;
  output GATE;

  reg [3:0] cnt;
  reg fullb;
  reg GATE;

  always
    @(posedge CLK or negedge CLRB)
        if (CLRB==0)
          begin
```

```
            fullb<=1;
            cnt<=0;
            GATE<=0;
          end
        else
          begin
            if (EN)
              begin
                if (fullb)
                  begin
                    if (cnt==15)
                      begin
                        GATE<=0;
                        fullb<=0;
                        cnt<=0;
                      end
                    else
                      begin
                        cnt<=cnt+1;
                        GATE<=1;
                      end
                  end
                else
                  begin
                    cnt<=0;
                    fullb<=0;
                    GATE<=0;
                  end
              end
            else
              begin
                cnt<=~GW;
              end
          end
endmodule


//セレクタ
module selector(q, sel, d1, d2);
  input sel;
  input d1, d2;
  output q;

  assign q=select(sel, d1, d2);

function select;
  input sel;
  input d1;
  input d2;
  case (sel)
    1'b0: select=d1;
    1'b1: select=d2;
  endcase
endfunction
```

```
endmodule


//D-フリップ・フロップ
module dff(q, qb, clk, d, resetb);
  input d, clk, resetb;
  output q, qb;
  reg q, qb;

  always
    @ (posedge clk or negedge resetb)
        if ( resetb==0 )
          begin
            q<=0;
            qb<=1;
          end
        else
          begin
            q<=d;
            qb<=~d;
          end
endmodule


//バッファ
module buffer(q, d);
  input d;
  output q;

  wire b;

  singnot not0(b, d);
  singnot not1(q, b);
endmodule


//NOT
module singnot(q, d);
  input d;
  output q;

  not (q,d);
endmodule
```

図 目 次

表 目 次

参考文献

[1] Review Document of ATLAS Japan Collaboration

[2] ATLAS Technical Proposal, CERN/LHCC 94-43(1994)

[3] T. Hambye and K. Riesselmann, SM Higgs bounds from theory, DESY 97-152(1997)

[4] J. Guinion, A.Stange, S. Willenbrock, Weakly-Coupled Higgs Bosons, preprint UCD-95-28(1995)

[5] M. Spira, QCD Effects In Higgs Phy sics, CERN-TH/97-68(1997)

[6] E. Richter-Was, D. Froidevaux, F. Gianotti, L. Poggioli, D. Cavalli, S. Resconi, Minimal Supersymmetric Standard Model Higgs rates and backgrounds in ATLAS, Atlas Internal Note PHYS-No-074(1996)

[7] ATLAS Muon Spectrometer Technical Design Report(1997)

[8] ATLAS Trigger Performance Status Report(1998)

[9] ATLAS Trigger-DAQ Steering Group, TRIGGER & DAQ INTERFACES WITH FRONT-END SYSTEMS: REQUIREMENT DOCUMENT VERSION2.0, Atlas Internal Note DAQ-NO103(1996)

[10] ATLAS First-Level Trigger Technical Design Report(1998)

[11] 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)

[12] ATLAS policiy on radiation tolerant electronics DRAFT2, Available from http://www.cern.ch/Atlas/GROUPS/FRONTEND/WWW/radtol2.ps

謝辞

本研究を進めるにあたり、適切な指導を与えてくださった指導教官、小林富雄東京大学教授に感謝します。また、常に丁寧にご指導くださいました高エネルギー加速器研究機構 佐々木修氏、京都大学 坂本宏氏に感謝します。研究を進めるうえで惜しみない協力をしてくださった池野正弘氏には心からお礼を申し上げます。ASIC 設計にあたり、さまざまな助言をいただいた新井康夫氏に感謝します。また、ASIC の設計を支援して下さった池田誠氏をはじめ、東京大学大規模集積システム設計教育センターの皆様に感謝します。さらに、近藤敬比古氏、岩崎博行氏、福永力氏、田中秀治氏、山内一夫氏、福井秀人氏をはじめとする ATLAS 日本グループのかたがたには貴重な意見をいただき、深く感謝します。また IC テスト用ボードをレイアウトしてくださった林栄精器の大川久氏に感謝します。松浦聡氏、東京都立大学の狩野博之氏の協力は、研究を進めるうえで不可欠のものでした。心から感謝します。さらに、研究生活を通して、惜しみない支援、協力をくださりました陣内修氏、吉田光宏氏、深津吉聡氏、津野総司氏、仁木太一氏、戸谷大介氏、広島大学 本間謙輔氏には深くお礼申し上げます。