2022 年度　修士論文

高輝度 LHC-ATLAS 実験に向けた
初段ミューオントリガーの開発と検証

(Development and verification of level-0 muon trigger for high-luminosity LHC-ATLAS Experiment)

東京大学理学系研究科物理学専攻
浅井研究室

修士課程 2 年

山下 恵理香

2023 年 1 月 26 日

概要

Large Hadron Colider（LHC）は欧州原子核機構（CERN）によって建設された世界最高エネルギーの陽子陽子衝突型加速器である。ATLAS 検出器は LHC の衝突点の一つに設置された大型汎用検出器であり、陽子同士の衝突事象を捉えることで新物理の兆候を探索している。現在第三期実験 (Run3) が進行中であり、Run3 の後さらに物理リーチを高めるべく高輝度化が予定されている。2029 年から開始される高輝度 LHC-ATLAS 実験において、ルミノシティが増強されるのに伴い、ミューオントリガーエレクトロニクスは全て刷新される。トリガー系の電子回路及び論理回路アルゴリズムは開発・試験が進められている最中である。

電子機器の一新に伴い、それらの複雑なケーブリング情報や、コンポーネント間のヒット情報転送に使用するデータフォーマットなどを管理する新たなシステムが必要となった。また、現在開発を進めている論理回路アルゴリズムの検証のためには、ファームウェアに入力する様々なテストパターンを生成する機構も不可欠である。さらに、実機、ソフトウェア間のコヒーレントな試験を行えるインフラストラクチャーは、ファームウェアの開発のみならず、運用時にも非常に有用なものである。本研究では以上のような要請を把握し、開発から運用までを見据えて、リレーショナル・データベースやビットワイズシミュレーションを基幹技術とした開発基盤の全体設計と実装に取り組んだ。

本研究ではリレーショナル・データベースの作成と、それを活用したテストパターン生成などのアプリケーションの開発、ビットワイズシミュレータの開発を行い、エンドキャップミューオントリガー系のエレクトロニクスの一つであるセクターロジックのファームウェアの検証システムを構築し、検証を行なった。本研究で開発したテストパターン生成機構は、無限運動量飛跡や、モンテカルロシミュレーションデータや実データから抽出した現実的な飛跡に対する入力ヒットパターンなどの様々なテストパターンを自在に生成することができ、エレクトロニクスの開発・検証研究の様々な段階で役に立つものである。また、ビットワイズシミュレータはファームウェアとテストパターンの入力の形式が一致しており、実機とコヒーレントな試験を行える。

本研究で検証に使用したテストパターンは主に無限運動量飛跡と MC データから生成したものの 2 種である。前者は実機およびファームウェアに入力してコヒーレントな試験を行い、トリガー系の正常動作を確認するとともに、実機およびファームウェアの出力が一致していることを確認した。続いて、後者の MC データから生成したテストパターンでビットワイズシミュレータの試験を行い、各モジュールの efficiency を検証した。

この検証システムは相互比較により全体パスの開発と安定運用を確立し、開発研究時のみならず、試運転、本番運転時でも有用な仕掛けに拡張していく予定であり、本研究はその基盤となるものである。

目次

図目次

表目次

第 1 章

序論

1.1 高輝度 LHC 実験

素粒子物理学における標準模型は、物質粒子である 6 種のクォークと 6 種のレプトン、及びその間に働く相互作用を記述するモデルである。標準模型は多くの実験結果を非常によく説明しているが、天文観測から明らかになった暗黒物質の存在など、標準理論では説明のつかない未解決問題が残っている。標準模型を超える様々な理論が提案されており、例えばヒッグスボソンの質量に対する階層性問題を解決する理論として、標準模型の素粒子とスピンが 1/2 異なるパートナー粒子を導入する超対称性理論がある。この超対称性理論で導入される粒子のうち、電気的に中性な最も軽い超対称性粒子は暗黒物質の候補として有力視されている。超対称性の実在を証明するための最重要なアプローチは、エネルギーフロンティアの衝突実験において、超対称性が予言するパートナー粒子を発見することである。

本研究の対象である ATLAS 実験は、世界最高エネルギーを持つ Large Hadron Colider(LHC) による陽子陽子衝突を観測することで、超対称性理論などの TeV 領域までの標準模型を超えた新しい物理事象を探索することや、標準模型の精密測定などを目的としている。上述した超対称性の実在を証明するパートナー粒子が LHC 加速器で到達可能な TeV 付近に存在している可能性があるため、探索が行われている。

現在 Run3 と呼ばれる実験期間中であるが、さらなる物理探索のために 2029 年からビーム輝度を向上させた高輝度 LHC が運転予定である (図 1.1)。重心エネルギーは 14 TeV、瞬間最高ルミノシティは $5 \sim 7 \times 10^{34}\ \mathrm{cm^{-2}s^{-1}}$ となり、10 年間の運転で積分ルミノシティは $3000 \sim 4000\ \mathrm{fb^{-1}}$ に到達する予定である。現在運転中の Run3 では積分ルミノシティが $450\ \mathrm{fb^{-1}}$ に到達する見込みであるので、高輝度化により桁が 1 つ上昇する。このように、LHC の高輝度化によって統計量が増大する。より断面積の小さな新事象の探索が可能になり、また、ヒッグス粒子など既知の粒子の高精度な精密測定が実現できる。一方で、高輝度 LHC 実験の瞬間ルミノシティは、ATLAS 検出器設計当初のデザイン ($1 \times 10^{34}\ \mathrm{cm^{-2}s^{-1}}$) の $5 \sim 7.5$ 倍となる。このような高輝度環境に対応するために検出器・トリガーの設計のアップグレードが必要となる。

1.2 ATLAS 実験・検出器

ATLAS 検出器は、LHC 加速器で加速した陽子同士の衝突事象を観測するための大型汎用検出器である。図 1.2 に ATLAS 検出器の断面図を示す。ATLAS 検出器を構成する検出器群は大きく 3 種類に分類することができる。最内層に配置される内部飛跡検出器は、ソレノイド磁石が作り出す磁場によって曲げられた荷電粒子の飛跡を再構成し、その曲率から運動量を求める。その外側に配置されているカロリメータはハドロン粒子、光子、電子の位置とエネルギーを測定する。最外層に設置されているミューオンスペクトロメータはカロリメータを透過したミューオンの飛跡を再構成し、運動量を測定する。本研究の対象となるのはミューオンスペクトロメータを使用したトリ

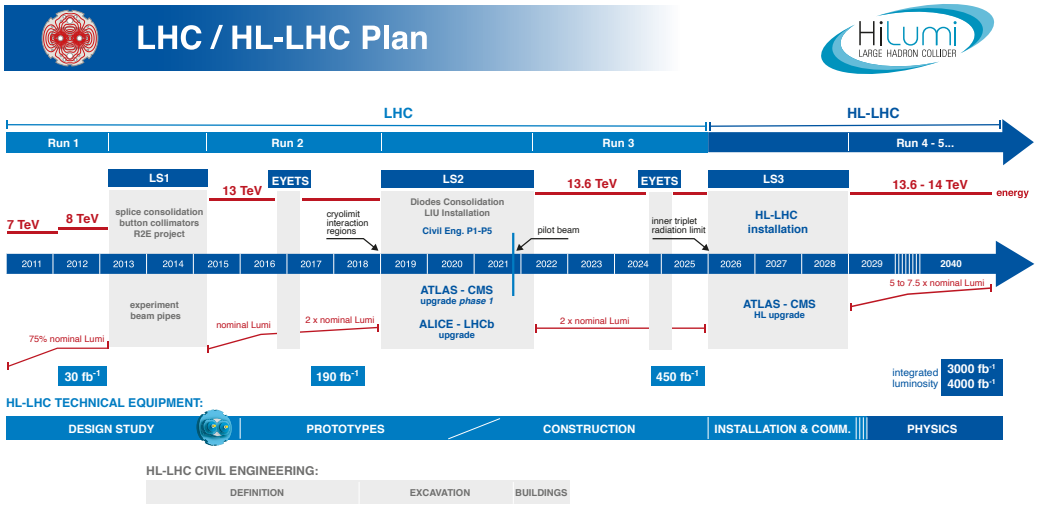

図 1.1　LHC 加速器の運転とアップグレード計画。高輝度 LHC の装置実装は 2026 年から、高輝度 LHC の運転は 2029 年からを予定している。(CERN 2022)

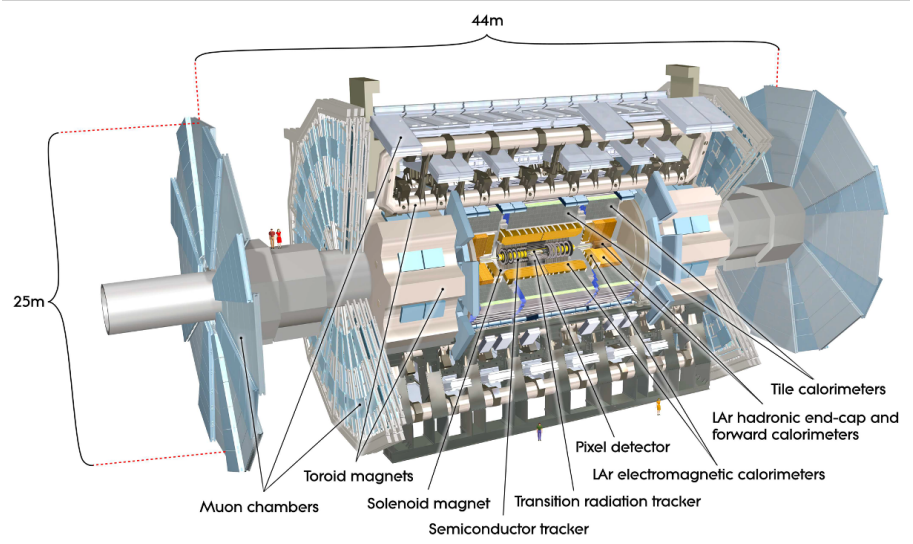

図 1.2　ATLAS 検出器の断面図。長さ 44m、直径 25m の円筒型検出器であり、内側から順に内部飛跡検出器、カロリメータ、ミューオンスペクトロメータが配置されている The ATLAS Collaboration et al. (2008)

ガーシステムである。

1.3　トリガー系とエンドキャップミューオントリガー

LHC-ATLAS 実験におけるビームバンチの交差頻度は 40 MHz である一方で、衝突事象のほとんどは陽子同士の非弾性散乱であり、興味のある精査対象となる衝突事象の割合は小さく、例えばヒッグス粒子の生成レートは、同じ条件で 1 Hz 以下である。さらに記録速度、記録媒体のコスト、データ処理のコストの面で衝突による全ての信号を記録することは現実的でないため、膨大な背景事象から興味のある事象のみを選別して記録することが望まし

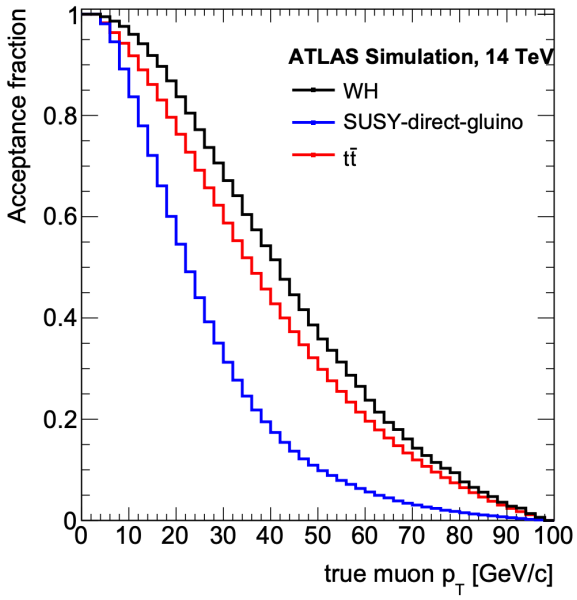

図 1.3　ミューオントリガーの p_{T} 閾値とアクセプタンスの相関 ATLAS Collaboration (2012)

い。ATLAS 実験ではこの選別をオンラインで行なっており、これをトリガーと呼ぶ。

LHC の高輝度化によってさらに瞬間ルミノシティは 5 ∼ 7.5 倍に増加する。この増加により、トリガーするべきミューオンの数と共に、パイルアップ (同一バンチ交差中の陽子の同時衝突) 事象による背景事象のトリガーレートも増加する。トリガーレートを実験がハンドルすることのできる範囲に抑えながら、物理事象に対するアクセプタンスを保つためには、従来の設計に比べて、トリガーにおける信号・ノイズの識別能力 (分解能) の向上と、読み出しにおける帯域の拡張が必要となる。もしミューオントリガーのアップグレードを行わない場合、p_{T} 閾値 20 GeV ではトリガーレートが強要量を超えてしまうので、p_{T} 閾値を 50 GeV まで上げなければならなくなる。具体的には、トリガーしたい事象の一つとして $WH \to \mu\nu b\bar{b}$ 過程があり、この過程では W/Z ボソンの崩壊により、高い p_{T} を持つミューオンが生成される。p_{T} 閾値を 50 GeV にしてしまうと、$WH \to \mu\nu b\bar{b}$ 過程の信号事象を約 50% も失ってしまう (図 1.3)。トリガー、読み出し系のアップグレードによりアクセプタンスの維持を実現するが、この高輝度 LHC に向けたアップグレードを Phase 2 アップグレードと呼ぶ。本研究ではその中でも特に、エンドキャップ部のミューオントリガーに関する Phase 2 アップグレードを研究対象とする。

ATLAS 実験のトリガーは二段階で構成される。初段トリガーでハードウェアによる高速な信号処理で粗く選別し、後段トリガーではソフトウェアでより時間をかけて精密なトリガー判定を行う。Run3 では初段トリガーレートと後段トリガーレートはそれぞれ 100 kHz と 1.7 kHz であったが、Phase2 アップグレードでは、それぞれ 1000 kHz と 10 kHz まで増強する。

エンドキャップミューオントリガー系

本研究の対象である初段エンドキャップミューオントリガーは、Thin Gap Chamber(TGC、詳細は 2 章) と呼ばれる検出器のヒット情報を用いて高速のミューオン再構成を行う初段トリガー系のハードウェアシステムである。

エンドキャップミューオントリガー系は、Phase 2 アップグレードで全てのデジタル電子回路を置き換え、また、高速ミューオン検出のアルゴリズムも刷新してこれに対応する。現在はハードウェア・ファームウェアの開発が急ピッチで進んでいるフェーズにある。

TGC 検出器から後段のエレクトロニクスであるセクターロジックの間では、多段のコンポーネントによってヒッ

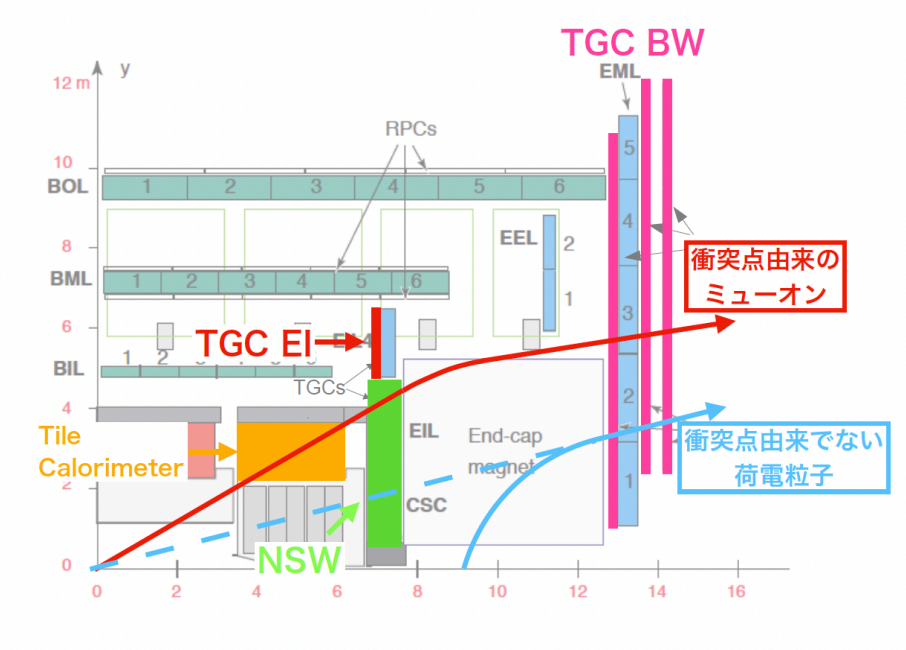

図 1.4　衝突点由来でない荷電粒子によるフェイクトリガーの概念図(小林蓮 2021)

ト情報の処理を行っている。Phase2 アップグレードでは、より高性能なトリガー判定を行うため、フロントエンドではヒット情報のデジタル化とタイミング調整のみを行い、コインシデンスによる情報の削減を行わない。ヒット情報は情報量を維持したままバックエンドのトリガー判定ボード (セクターロジック) へ送信され、コインシデンスとパターンマッチングによりミューオンの p_{T} が計算される。

本研究では現在開発が進められているセクターロジックの検証システムを構築するものであり、その一つとしてセクターロジックのテストパターンとなるビットマップ生成機構を作成したが、このテストパターンを生成するためには前段の多段かつ多数のコンポーネントの複雑なケーブリング情報を全て把握する必要があった。この要請から、本研究では開発研究のインフラストラクチャとして高輝度 LHC-ATLAS 実験におけるケーブリング情報を網羅したリレーショナル・データベースを作成した。

Phase2 アップグレードによるミューオン識別能力の向上

Phase2 アップグレード後の初段エンドキャップミューオントリガーシステムでは、トリガーの判定時間を 2.5μs から 10μs に延ばすことで、より複雑なトリガーアルゴリズムを導入する。TGC Big Wheel(BW) でのコインシデンスに加えて、磁場内側の検出器 (NSW、RPC BIS78、Tile calorimeter、TGC EI) における飛跡情報とマッチングをとることにより、衝突点由来ではない荷電粒子によるトリガー (フェイクトリガー) を削減する。このフェイクトリガーは陽子陽子衝突で生じた粒子がビームパイプと衝突することにより生じた荷電粒子が、あたかも衝突点由来であるかのようなヒットを TGC BW に残すことによって生じる。さらに、位置・角度分解能の良い NSW や RPC BIS78 の飛跡情報を TGC BW の飛跡情報を組み合わせることにより、p_{T} 計算の精度を上げることもできる。

Run3 のトリガーシステムを高輝度 LHC 環境下でそのまま使用した場合、p_{T} 閾値 20 GeV によるトリガーレートは 34 kHz となると予想される[*1] 。これに対し、先行研究(小林蓮 2021)で開発されたトリガーアルゴリズムの

[*1] Run3 ではルミノシティ $2.0 \times 10^{34}\ \mathrm{cm^{-2}s^{-1}}$ 環境下で、p_{T} 閾値 20 GeV によるトリガーレートは 9 kHZ と予想されている。高輝度 LHC ではルミノシティ $7.5 \times 10^{34}\ \mathrm{cm^{-2}s^{-1}}$ が予定されており、トリガーレートはルミノシティに比例することから、トリガーレートが 34 kHz になるだろうと計算できる。

p_{T} 閾値 20 GeV によるトリガーレートは 18 kHz だった。p_{T} 閾値以下のミューオンとフェイクトリガーを削減することにより、Run3 のトリガーアルゴリズムと比較してトリガーレートを 47% 削減できると見込まれている。

1.4　本研究の目的と論文の構成

本研究は、高輝度 LHC に向けた電子回路のアップグレードのための、開発から運用までを見据えたインフラストラクチャーの整備と、現在開発が進められているミューオントリガーシステムの一部であるセクターロジックファームウェアの開発・検証の全体設計から検証システムの開発・運用までを行なったものである。

高輝度 LHC 実験に向けて、全てのデジタル回路を置き換え、ファームウェアとして実装する論理回路も全て書き換える。この電子機器の一新に伴い、トリガー系を構成する多段かつ多数の電子機器の複雑なケーブリング情報や、コンポーネント間のヒット情報転送に使用するデータフォーマット、チャンネルの履歴情報など系統的に管理する機構が必要となった。また、新たに開発が進められているトリガー回路の検証機構として、様々なテストパターンを生成するシステムと、実機とコヒーレントな試験を行えるソフトウェアシミュレータの開発も重要である。本研究はこれらの要請に対し、開発から運用までを見据えて、リレーショナル・データベースやビットワイズシミュレーションを基幹技術とした開発基盤の全体設計と実装に取り組むものである。

本研究ではリレーショナル・データベースの作成と、それを活用したテストパターン生成などのアプリケーションの開発、ビットワイズシミュレータの開発を行い、エンドキャップミューオントリガー系のエレクトロニクスの一つであるセクターロジックのファームウェアの検証システムを構築し、検証を行なった。本研究で開発したテストパターン生成機構は、無限運動量飛跡や、モンテカルロシミュレーションデータや実データから抽出した現実的な飛跡に対する入力ヒットパターンなどの様々なテストパターンを自在に生成することができ、エレクトロニクスの開発・検証研究の様々な段階で役に立つものである。また、ビットワイズシミュレータはファームウェアとテストパターンの入力の形式が一致しており、実機とコヒーレントな試験を行える。

本論文の構成は以下の通りである。第 2 章では、本研究の対象となる、高輝度 LHC-ATLAS 実験における TGC 検出器システムと各エレクトロニクスの役割を示す。続いて、第 3 章では、セクターロジックの統合試験環境の開発における各要素の概要を述べる。第 4 章では、第 5 章・第 6 章の開発研究の基盤として構築したリレーショナル・データベースについて記述する。このリレーショナル・データベースは、エンドキャップ領域のミューオン検出器である TGC 検出器からセクターロジックまでのケーブリングや、チャンネルに付随する情報等を一元管理するためのものである。本研究の対象であるセクターロジックに入力するテストパターンとなるビットマップを作成するためには、TGC 検出器からセクターロジックまでの間に存在する多段のコンポーネントによる複雑なケーブリング情報を全て考慮しなければならないため、高輝度 LHC-ATLAS 実験におけるケーブリング情報を網羅したリレーショナル・データベースが必要となった。加えて、より合理的なテストパターンの生成や、ファームウェアの開発に使用する新たなチャンネル番号の定義も行い、リレーショナル・データベースに組み込んだ。以上のように、本研究ではリレーショナル・データベースやビットワイズシミュレーションを基幹技術とした開発基盤の全体設計と実装を行なった。

第 5 章ではセクターロジックのトリガー系のビットワイズシミュレータのロジックについて説明する。セクターロジックはエンドキャップミューオントリガーシステムの一部を担う電子機器である。セクターロジックのファームウェアに対して、ロジック及びモジュールごとの入出力が完全に一致するシミュレータを開発した。この新しく開発したビットワイズシミュレータは、実機および既存のファームウェアシミュレータとテストパターンの入力形式を揃えることにより、コヒーレントな試験を行えるインフラストラクチャとなっており、現在の開発のみならず、セクターロジック実機の運用時も見据えた設計となっている。

続いて、第 6 章ではセクターロジックの実機およびシミュレータの検証に使用するテスト入力パターンの生成機構の開発について述べる。この機構は無限運動量飛跡や、モンテカルロシミュレーションデータや実データから抽

出した現実的な飛跡に対する入力ヒットパターンなどの様々なテストパターンを生成することができる。前者の最も単純かつ重要でロジックの検証が比較的容易な無限運動量飛跡のテストパターンは開発研究の初期で役立ち、後者のテストパターンは本番の衝突実験での観測が期待される様々な事象に対する応答を満遍なく検証するために必要であるので、このような複数種類のテストパターンが生成可能な機構は、セクターロジックの開発・検証研究の様々な段階で役に立つ。

第 7 章では、第 5 章で記述したシミュレータに第 6 章のテストパターンを入力することで行なったトリガー系の検証について述べる。入力に使用したテストパターンは主に無限運動量飛跡と MC データから生成したものの 2 種である。前者は実機およびファームウェアに入力してコヒーレントな試験を行い、トリガー系の正常動作を確認するとともに、実機およびファームウェアの出力が一致していることを確認した。続いて、後者の MC データから生成したテストパターンでビットワイズシミュレータの試験を行い、各モジュールの efficiency を検証した。

最後に 8 章で本研究の結論と今後の展望についてまとめる。

本研究の構造については、第 3 章でさらに詳しく述べる。

第 2 章

LHC-ATLAS 実験におけるエンドキャップミューオントリガー系の検出器とエレクトロニクス

本章では、本研究の対象となる LHC-ATLAS 実験におけるエンドキャップミューオン検出器である Thin Gap Chamber（TGC）検出器と、この検出器によるヒット情報を用いたミューオントリガーシステムについて説明する。

本研究の対象であるエンドキャップミューオントリガーは、TGC 検出器で観測したヒット情報を主に用いることで、高速でミューオンの再構成を実現するものである。再構成されたミューオンの横運動量 p_{T} によってトリガー判定を行う。

2.1　エンドキャップミューオントリガーシステム

ATLAS 検出器郡の最も外側に位置するミューオンスペクトロメータは、カロリメータを透過してきたミューオンを捉え、検出器よりも内部にあるソレノイド磁場によって曲げられた飛跡を再構成することによって運動量を測定するための検出器である。ミューオン検出器には複数種類が存在しているが、TGC 検出器は時間分解能に優れ、トリガー検出器としての役割を果たす[*1] 。

図 2.1 に ATLAS 検出器の座標系を示す。ATLAS 実験では標準座標系として、直交座標系と円筒座標系の二種類を使用している。直交座標系は、衝突点を原点として、x 座標の正の方向を LHC 周回軌道の中心へ向かう法線方向、y 軸の正の方向を鉛直上向き (地上の方向) を指し、ビーム軸に平行に右手系で z 軸が定義される。円筒座標系では、ビーム軸からの動径方向を R、方位角を ϕ と定義する。また、天頂角 θ を用いて、擬ラピディティ η を以下のように定義する。

$$\eta = -\ln\left(\tan\frac{\theta}{2}\right) \tag{2.1}$$

ATLAS 実験を始めハドロンコライダー実験では、非弾性散乱によって生成される粒子数が η に対し一定 ($dN/\,d\eta = \mathrm{const}$) となることが経験的に知られているため、天頂角 θ ではなく η を運動方向や立体角の記述に用いる。本研究の対象である TGC 検出器も、$d\eta$ が一定となるように検出器のチャンネル幅が設計されている。初段ミューオントリガーは $|\eta| < 1.05$ のバレル部、$|\eta| > 1.05$ のエンドキャップ部で独立したシステムとなっており、本研究の対象はエンドキャップ部である。

[*1] Run2 まで使われていたミューオン検出器には Resistive Plate Chamber (RPC), Thin Gap Chamber (TGC), Monitored Drift Tube (MDT), Cathode Strip Chamber (CSC) の 4 種類の検出器がある。RPC と TGC は時間分解能に優れ、トリガー検出器としての役割を果たし、MDT と CSC は位置分解能に優れ、運動量の精密測定に使われる。Run3 以降では CSC と一部の MDT と TGC が、より位置分解能の高い Nw Small Wheel(NSW) に置き換わる。

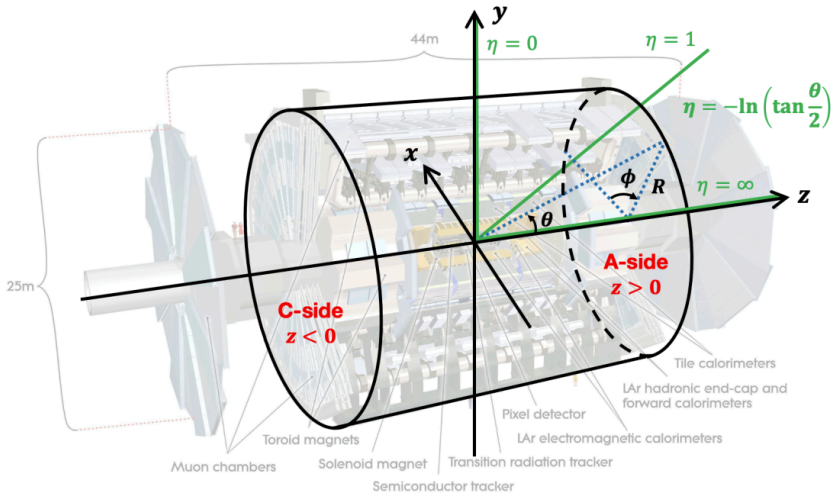

図 2.1　ATLAS 検出器で用いられている座標系(杉崎海斗 2021 から引用)

TGC は $1.05 < |\eta| < 2.4$ のエンドキャップ領域をカバーする検出器である。TGC は Endcap Toroidal Magnet より内側に位置する Endcap Inner (EI) と外側に位置する Big Wheel (BW) に大別される。

2.1.1　TGC モジュールからなる Big Wheel の構造とトリガーセクターの分割

TGC 検出器はアノードであるワイヤーと平板カソード電極で構成されるガス検出器であり、制限比例領域で動作させる。2 層 (Doublet) または 3 層 (Triplet) のチェンバーからなるチェンバーユニットを構造の最小単位とし、それらで構成される円盤状のものをステーションと呼ぶ。TGC BW は 3 つのステーションからなり、衝突点に近い側から、3 層構造のチェンバーユニットで構成される M1 ステーション、2 層構造のチェンバーユニットで構成される M2・M3 ステーションが設置されている。M1 ステーションの正面写真を図 2.2 に示す。

トリガー系ではチェンバーユニットの構造ではなく、TGC 検出器はほぼ同一の z 座標に並んだ複数枚のチェンバーからなるレイヤーという単位で取り扱う。レイヤーの番号はステーションと同様に衝突点側からの枚数で定義され、M1 ステーションはレイヤー 1〜3、M2 ステーションはレイヤー 4・5、M3 ステーションはレイヤー 6・7 で構成される。この 7 層でコインシデンスを行い、ミューオンの飛跡を再構成することによって p_{T} を計算する。

TGC Big Wheel のチェンバーは、η 座標によってフォワードとエンドキャップの 2 つの領域に分けられている。[*2] $1.05 < |\eta| < 1.92$ のエンドキャップ部は 48 回対称のチェンバーユニット構造で全方位角を覆い、一方、ビーム軸に近い $1.92 < |\eta| < 2.4$ のフォワード部では 24 回対称のチェンバーユニット構造で全方位角を覆う。24 回対称構造によるトリガーセクターについては、本章の最後に記述する。

2.1.2　7 層のコインシデンス論理によるミューオン飛跡再構成

TGC 検出器を使用したトリガーシステムでは M1〜M3 までのワイヤー 7 層、ストリップ 6 層のコインシデンスによってミューオンの飛跡を再構成する (図 2.3)。このコインシデンスは後述するエレクトロニクスの内の 1 つであるセクターロジックで行われる。

[*2] 円筒形をした ATLAS 検出器全体の側面をバレル部、底面をエンドキャップ部と呼ぶが、このバレル部に対する「エンドキャップ部」をさらに分割したものをエンドキャップ、フォワードと呼称している。η 範囲で考えるならば、前者が広義のエンドキャップ、後者が狭義のエンドキャップと言える。本研究は前者のエンドキャップのトリガーシステムを対象とするので、以降の議論で単に「エンドキャップ」と呼ぶ場合は、フォワードと対応する後者の方のエンドキャップを指す。

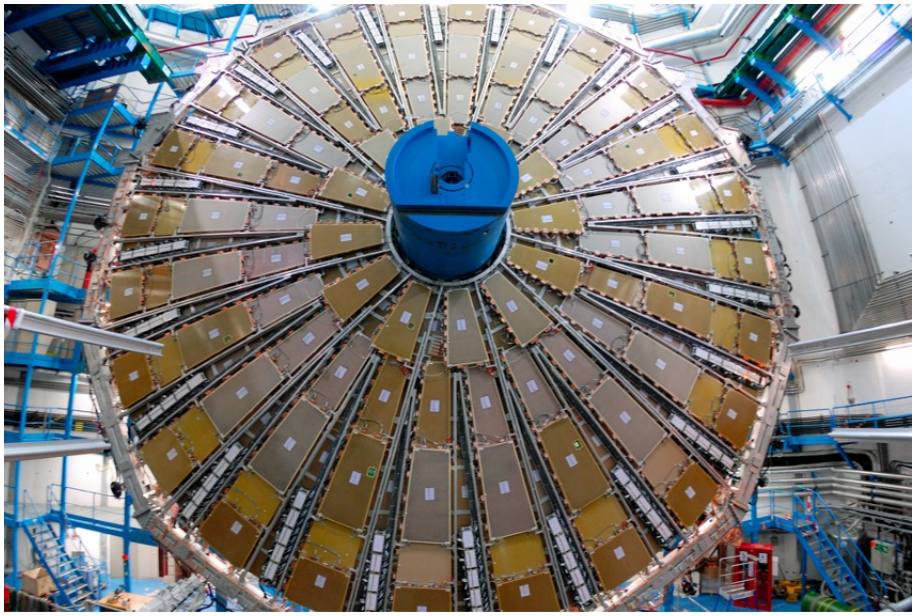

図 2.2　TGC 検出器 Big Wheel 構造体の正面写真 (M1)

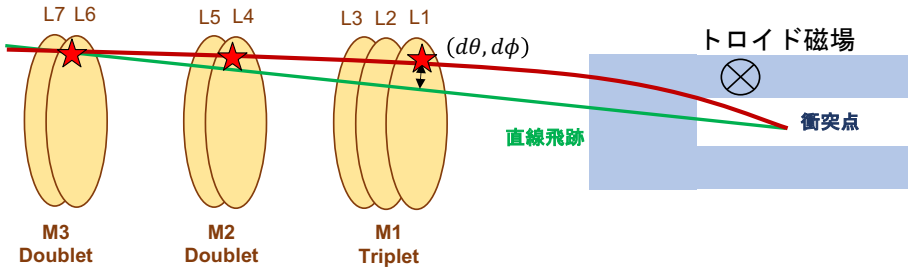

図 2.3　TGC 検出器 7 層のコインシデンスによる飛跡再構成。まずステーション内でコインシデンスを行い代表点 ($\star$) を得、代表点の組み合わせから飛跡を再構成する。

7 層のコインシデンスは 3 段階で行われる。最初の 2 段階はワイヤーとストリップで独立に計算を行う。

まず最初に、ステーション内でのコインシデンスを取り、ステーションごとに代表点を得る。代表点を得る計算は AND や OR、NOT からなる論理回路によって記述される。この計算については 5.2 節で詳しく述べる。

続いて、ステーション間でのパターンマッチングにより、M3 ステーションのヒットを通る直線飛跡 (無限運動量の飛跡) からの差分 $(\Delta\theta, \Delta\phi)$ を得る。この計算には Look Up Table(LUT) を用いる。この計算については 5.3 節で詳しく述べる。

最後に、ワイヤーから得た $\Delta\theta$ とストリップから得た $\Delta\phi$ を入力とする LUT を用いて p_{T} を概算する。ここで概算した p_{T} がトリガー判定に用いられる。この計算については 5.4 節で詳しく述べる。

2.1.3　Phase 2 アップグレード

高輝度 LHC 実験に向けた Phase 2 アップグレードでは、上述の全てのデジタル回路を置き換え、ファームウェアとして実装される論理回路も全て書き換える。また、ファイバーの数や帯域が飛躍的に大規模化し、一つの回路が担当する信号数がこれまでよりも圧倒的に多くなる。このような事情により、現在、開発から運用までを見据えたデザインの良いインフラストラクチャを作成しておくことが非常に重要となっており、これが本研究の動機となっ

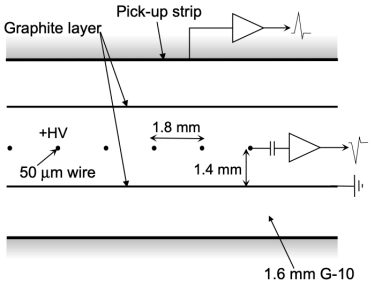

図 2.4　TGC チェンバーの断面図。ワイヤーチャンネルとストリップチャンネルが直交しており、ワイヤーには高電圧がかかっている。(The ATLAS Collaboration et al. 2008)

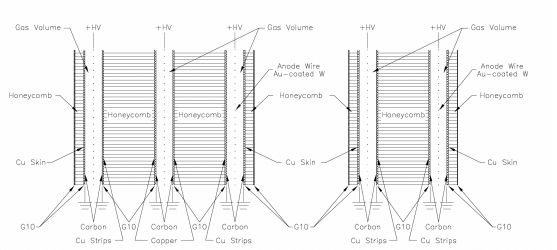

図 2.5　トリプレットとダブレットのステーションの断面図(The ATLAS Collaboration et al. 2008)

ている。

2.2　TGC 検出器の仕組み

TGC 検出器は Multi-Wire Propotional Chambers(MWPC) の一種である。アノードワイヤー間隔が 1.8mm、ワイヤー直径 50μm、アノードワイヤーとカーボングラファイトによるカソードの距離が 1.4 mm である (図 2.4)。TGC 検出器のチェンバーは図 2.5 のように、ステーション内で 3 層または 2 層の組として設置される。ワイヤーとストリップは直行しており、R 方向 (ワイヤー) と ϕ 方向 (ストリップ) の 2 次元読み出しを行う。チェンバーの内部は CO_2 55 % とノルマンペンタン 45 % が充填されている。ミューオンがチェンバーを通過すると、ガスが電離し、電場によってワイヤー近傍で指数関数的に増幅される。この検出器ではワイヤーには約 2.8 kV の電圧をかけており、ガスゲインは約 3×10^5 となっている。

このようにワイヤーには高電圧がかかっているが、チェンバーで得られた電流信号を ASD に入力する前に直流高電圧を切るためのコンデンサを設置している。一方、ストリップ電極の方には高電圧はかかっていないため、コンデンサを設置していない。

この検出器は時間応答を重視しているため、ワイヤー間隔を狭くとり (図 2.4 より 1.8mm)、ドリフト時間を短く抑えている。後段でヒット情報を処理するエレクトロニクスのコストは、チェンバーから読み出すチャンネルの数に依存する。ワイヤー間隔を狭くするためにワイヤーの数が多くなっているが、個々のワイヤーを区別するほどの位置分解能は不要であることから、ワイヤー電極は 1 つのチャンネルが担当する η 幅 ($d\eta$) が一定となるように 4〜20 本をグルーピングして読み出している。また、薄いガス層により、十分なレート耐性が確保される。

2.3　エレクトロニクス

2.3.1　Amplifer Shaper Discriminator (ASD) ボード

TGC 検出器のワイヤー・ストリップチャンネルで得られたアナログ信号は、Amplifer Shaper Discriminator (ASD) ボードへ送られ、デジタル信号へ変換される。入力されたアナログの電流信号は、ASD 上の ASIC で電圧信号に変換・増幅され、閾値電圧によりデジタル化をされたのち、最終的に LVDS 信号として出力される。図 2.6 のように、1 枚の ASD ボードには 4 枚の ASD チップが存在しており、各 ASD チップが 4 チャンネルを担当するため、1 枚の ASD ボードでは 16 チャンネル分の検出器信号を処理することができる。前述の通り TGC には約 32 万チャンネルが存在しているため、システム全体では ASD ボードは約 2 万枚設置される(ATLAS Collaboration

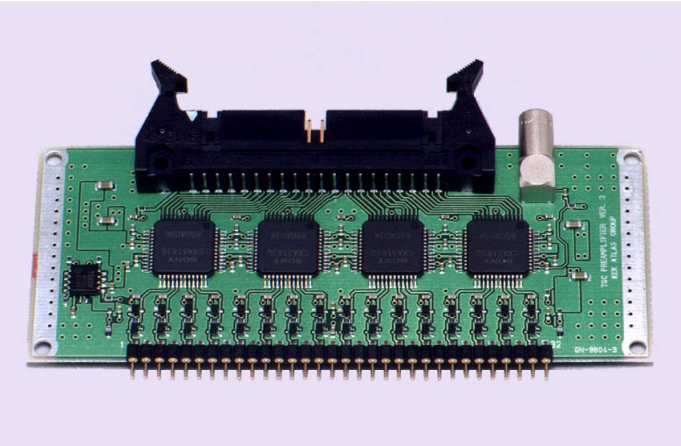

図 2.6　ASD ボードの写真

1999)。

2.3.2　Primary Processor (PS) ボード

ASD ボードでデジタル化された検出器信号は PS ボードに入力される。PS ボードには 8 つの Patch-Panel ASIC (PP ASIC) と Xilinx 社製の Kintex-7 FPGA [*3] という 2 種類の集積回路が搭載されており、PS ボードへ入力された検出器信号はまず PP ASIC へ入力される。

PP ASIC の機能は、信号のタイミング調整 (Bunch Crossing Identification、BCID) である。衝突点で生じたミューオンがチェンバーを通過することによって生じる TGC 検出器からの信号は、衝突点からチェンバーまでの飛行時間の違い (45〜64 ns) や、ASD から PP ASIC までのケーブル長の違い (1.8〜12.5 m) により、到着時間に分布を持つ。各信号線に対し、ケーブル長と飛行距離に応じた fine delay をかけることで、信号を LHC クロックと同期させる。また、ドリフト時間の違いや、検出器上の入射位置によって、同一信号線のヒットでもイベント毎で到着時間はばらつきをもつ。これらの事象毎にランダムにかかる時間分布については BCID におけるゲート幅を十分に広げることで、重複は許すが効率は落とさないように対応する。1 つの PP ASIC は 2 枚の ASD ボードと接続されるため、最大 32 チャンネルの LVDS 信号を処理する。PPASIC で BCID され、LHC バンチ交差クロックに同期した検出器信号は FPGA によってまとめられ、2 本の光リンク (転送レート 8 Gbps[*4]) でセクターロジックに出力する。ここでは情報の削減は行わず、1 つの PS ボードの受信する最大 256 チャンネルの BCID 後のビットマップ情報を 25 ns 毎に、全て後段のセクターロジックへ送信する(ATLAS Collaboration 2019)。

2.3.3　セクターロジック

セクターロジックはバックエンドに設置される、p_{T} の概算を行うボードである。SL は PS ボードでタイミング調整された TGC ヒット情報に加えて、磁場領域よりも内側の検出器である NSW、RPC BIS78、Tile カロリメータから受信した情報を受け取り、ミューオンの飛跡再構成と p_{T} の概算を行う。TGC 検出器の 1/24 構造につき 1 枚の SL が用意されるため、システム全体では 48 枚の SL が設置される。ヒット情報の処理の詳細については、5 章で記述する。

[*3] Field Programmable Gate Array の略。内部の論理回路をプログラムによって生成可能なデバイスである(木村 2012)。

[*4] 転送レートの要求は 256 チャンネル分のヒット信号とヘッダー (64 bit) を、40 MHz で、8B/10B プロトコルの元で送るので 16 Gbps となる。PS board ではこの転送レートを、それぞれ 8 Gbps で動く 2 本の光ファイバーを用いることで実現する(青木匠 2022)。

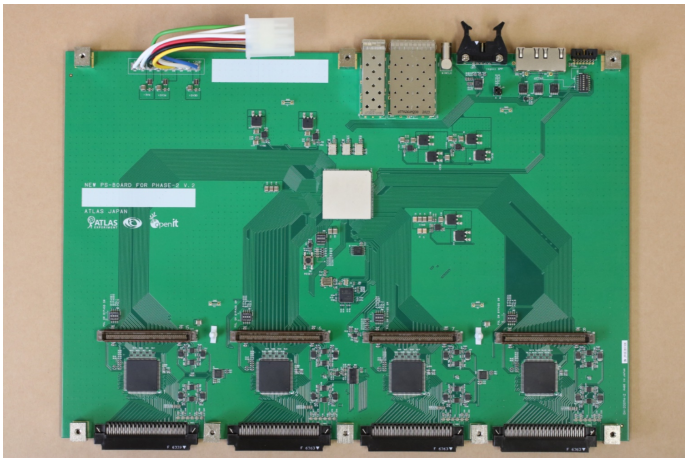

図 2.7　PS ボードの写真

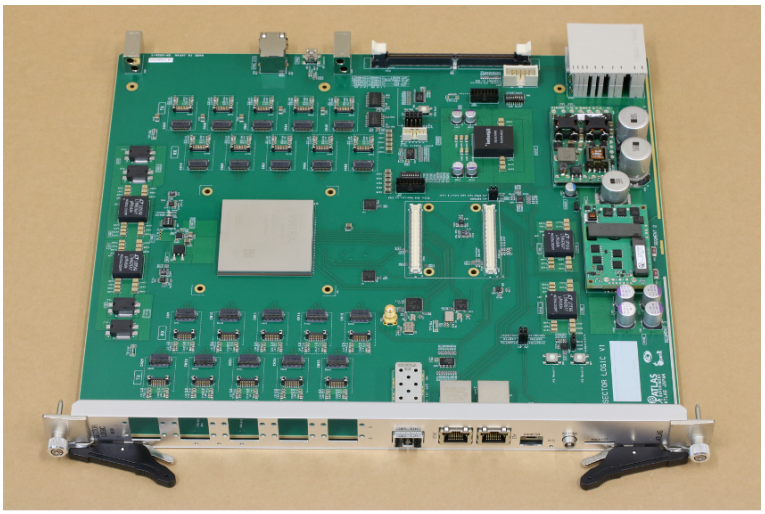

図 2.8　SL 第一試作機の写真

2.4　1 つのセクターロジックが担当する領域と Big Wheel の繰り返し構造

TGC Big Wheel のチェンバーは、ϕ 方向にエンドキャップでは 48 回、フォワードでは 24 回繰り返し構造をしているので、1/24 セクターは 1 枚のフォワードチェンバーと、2 組ずつのエンドキャップのチェンバー（M1 では 4 枚 $\times$2 の 8 枚、M2・M3 では 5 枚 $\times$2 の 10 枚）で構成される。図 2.9 に示すように、この範囲には 6408 の検出器チャンネルがあり TGC 検出器全体では約 32 万チャンネルが存在している。この範囲を 1/24 セクターと呼び、セクターロジックは Big Wheel の 1/24 セクターにつき 1 枚配置される。トリガー論理回路は全て 1/24 セクターで閉じた構造をしており、その内部でのケーブリングやハードウェアの配置は全セクターで対称な構造を持つ。この構造内及び検出器全体におけるエレクトロニクスの数を表 2.1 にまとめた。

1/24 セクター内でエンドキャップ部の対称な構造は、ϕ 方向の座標によって、$\phi 0$、$\phi 1$ と呼んで区別する。エレクトロニクスにおいては、ASD ボードは完全に $\phi 0$、$\phi 1$ が対称な構造となっている。

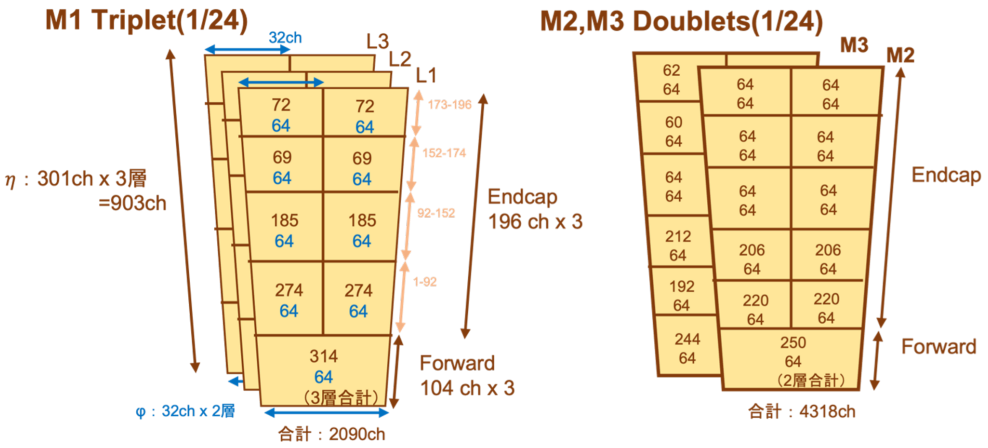

図 2.9　1/24 セクター内における検出器チャンネル総数。左の図では、M1 ステーションに存在するレイヤー 1 ～レイヤー 3 のチャンネル数を、各チェンバーごとに Wire チャンネル・ストリップチャンネルで区別して書き出した。右の図では、レイヤー 4・5 からなる M2 ステーションと、レイヤー 6・7 からなる M3 ステーションに関して、2 レイヤー合計のチャンネル数をチェンバーごとに記述した。

表 2.1　TGC エレクトロニクスの規模。TGC 検出器全体の数の列で、検出器チャンネル及び PS ボードの枚数に関してのみ、EIL4 検出器を含む値となっている。それ以外の数字は Big Wheel のみで計上した値である。

	1/24 構造	TGC 検出器全体
検出器チャンネル	6408	320k
ASD ボード	419	20k
PS ボード	29	1434
セクターロジック	1	48
ファイバー（PS ボード ～ セクターロジック間）	87	4k

サブセクター

フォワード、エンドキャップ $\phi 0$、エンドキャップ $\phi 1$ をサブセクターと呼び、トリガー論理はサブセクター内で閉じた論理として実現される。PS ボードの段階では、一部の例外[*5]を除き、基本的には 1 枚の PS ボードに接続する ASD は、同じサブセクターに属するものである。セクターロジックのトリガーファームウェアは、内部検出器である EIL4 とのコインシデンスを取るまで、サブセクターごとに独立なものとなっている。セクターロジックへの入力が $\phi 0$ と $\phi 1$ で対称的なものとなるように PS ボードからセクターロジックへのケーブリングが設計されており、ファームウェアは $\phi 0$ と $\phi 1$ で対称的な構造となる。

図 2.10 にセクターロジックへの入力からトリガー系前段部分のモジュールの構成を表す。1/24 セクター内に、TGC Big Wheel の PS ボードは 29 枚あり、各 PS ボードから 2 本の光リンクによってそれぞれ 128 bit の検出器情報を受け取るため、セクターロジックへ入力されるデータは 128 bit×58 links の形式となる。1 本の光リンクでは同じサブセクターの情報のみを送信する。トリガー系最前段部のモジュールである「Channel Mapping」からサブセクターごとに分かれており、$\phi 0$ と $\phi 1$ で対称的な構造である。

ただし、TGC よりも内部にある磁場構造が 1/24 対称ではないため、図 2.10 の最後弾に当たる磁場による曲率

[*5] ボードの枚数を削減するため、1 枚の PS ボードに対してエンドキャップ領域の ASD とフォワード領域の ASD の両方と接続するものなどがある。

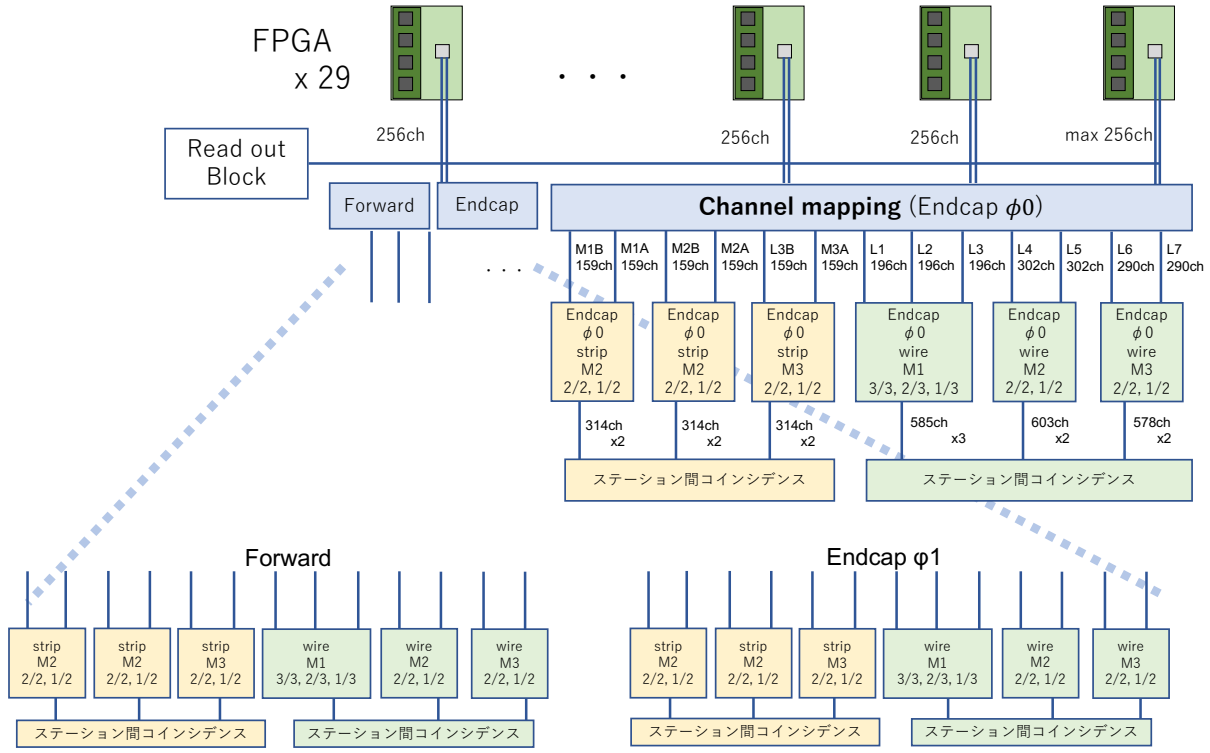

図 2.10　セクターロジックのトリガー系における Channel Mapping からステーション間コインシデンス (Segment Reconstruction) までのダイアグラム。緑で塗りつぶした箱がワイヤーチャンネルの処理回路、黄色で塗りつぶした箱がストリップの処理回路を表している。回路全体は大きく 3 つのサブセクター (Endcap ϕ0、Endcap ϕ1、Forward) で分かれており、対称的な構造をしている。

から p_{T} を計算する部分 (ステーション間コインシデンス) では対称性が崩れ、ϕ0 と ϕ1 で曲率から p_{T} への変換は各々で最適化される。p_{T} への変換はファームウェア中の Random Access Memory を用いた Look Up Table で実装されるが、このメモリに書き込む情報を変えることで、ファームウェア設計は変えずに磁場の違いを吸収できる設計となっている。ファームウェアの詳細は 5 章に記述する。

第 3 章

セクターロジックの開発及び統合試験環境の全体設計

この章では本研究で行なった Phase2 アップグレードのためのインフラストラクチャの開発と、セクターロジックのトリガー系の検証機構の全体設計を説明する。

2022 年度現在、高輝度 LHC 実験にむけた初段ミューオントリガーシステムの改良のための Phase2 アップグレードが進められており、開発から運用までを見据えたインフラストラクチャや、新たなトリガーシステムの検証システムなどの開発基盤の作成が必要である。開発基盤を新しく設計するにあたり、開発から、試運転、本番運用までを見据えて、検証のシステムをよいデザインで実装していくことは非常に価値があることである。

本研究で検証の対象としたセクターロジックは Phase2 アップグレードで開発が進められている新たなトリガー判定ボードである。このトリガーロジックのファームウェアは複数のモジュール[*1] から構成され、各モジュールは日本国内の複数の大学で分担して開発された。現在はそれらのモジュールの統合試験が行われているが、この試験のためには検証機構の開発が急務であった。

本研究ではこの開発・統合が進むセクターロジック実機統合試験等に使用する検証機構の全体設計を行った。本研究では特にセクターロジックのトリガー系を再現するシミュレータと、実機とシミュレータの両方で使用するためのヒットパターンの生成システムを開発した。また、ヒットパターン生成システムやファームウェア自動生成の実現のため及び、Phase2 アップグレードでのインフラストラクチャとして、チャンネルに付随する情報やケーブリングなどを一元管理するデータベースの作成も行なった。

図 3.1 に本研究で設計した検証システムの全体像を示す。様々な検討を重ねた結果、検証のためにこの図に示されるような仕掛けを備えることが最善であると判断し、研究に取り組んだ。本研究で開発した検証システムの主な要素は①ビットワイズシミュレータ、②テストパターン作成システム、③各出力の照合による検証/Truth 情報を使用した解析の 3 つに分けることができる。以下ではこのシステムを構成する各要素について述べる。

この検証システムを実用した結果と得られた知見については、7 章で述べる。

[*1] TGC Big Wheel のワイヤー 7 層 + ストリップ 6 層の合計 13 層によるコインシデンスは複数の段階に分割されて行われる。HDL でファームウェアを記述する際、機能単位で分類した HDL を記述し、それらを部品として組み合わせることにより全体を設計する。この各部品をモジュールと呼ぶ(木村 2012)。このようにファームウェアは階層構造で記述されており、ロジックによってはモジュールは更に複数のモジュールで構成され、多段の階層構造をもつ。本論文で単にモジュールと呼ぶときは、「ステーション内コインシデンス」、「ステーション間コインシデンス」、「ワイヤーとストリップ間のコインシデンス」等のコインシデンスの段階に対応する、最上位層に近い大枠のモジュールのことを指す。

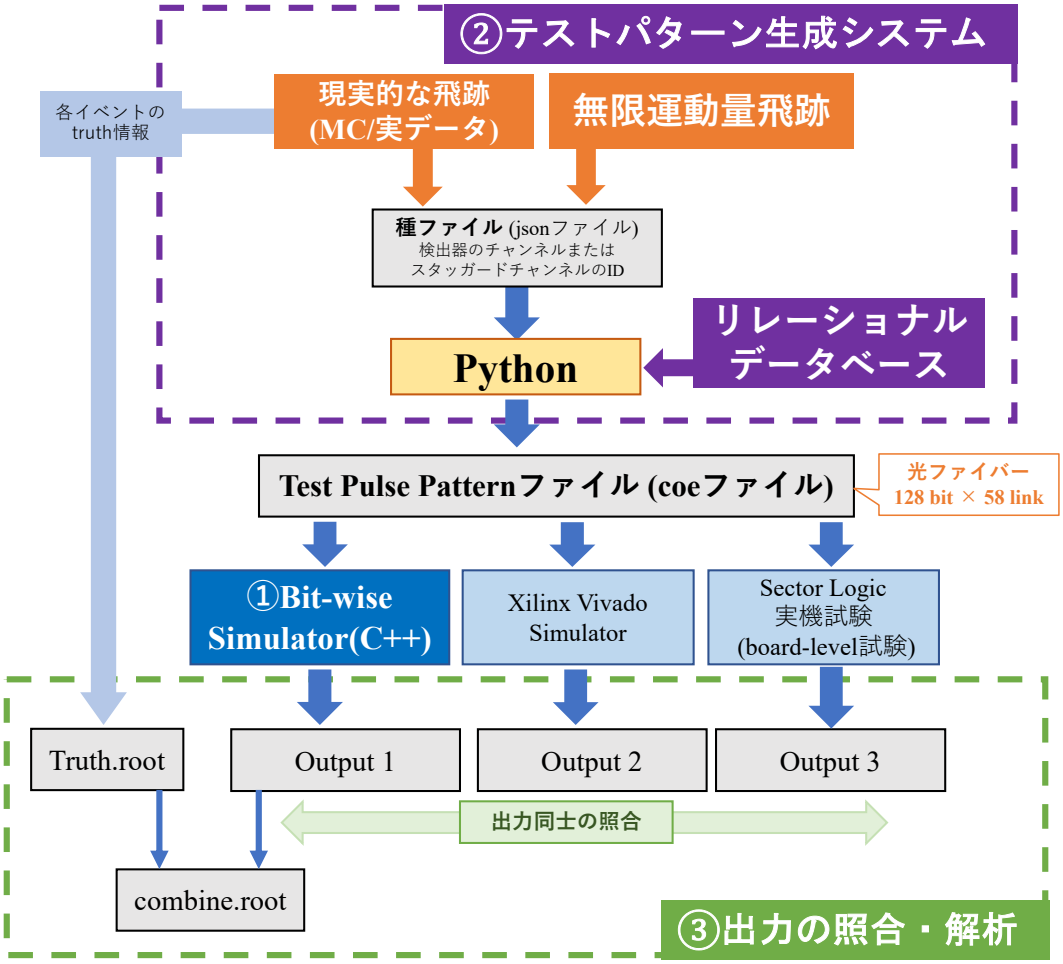

図 3.1　研究の全体設計：ヒットパターンの生成・処理・SL トリガー出力の経路。中央よりやや下にある 3 つの青い箱が SL 実機と 2 種類のシミュレータを表している。最も右の箱が SL 実機による試験、中央の「Xilinx Vivado Simulator」がファームウェアシミュレータ、左の「Bit-wise Simulator(C++)」が執筆者が作成したシミュレータを表す。上部のテストパターン生成システムで生成した同一のテストパルスパターンファイルを実機・シミュレータに入力し、その出力を照合することで堅固な検証を行うことができる。

3.1　①セクターロジックのビットワイズシミュレータ

セクターロジックはバックエンドに設置されるボードである。TGC 検出器と内部検出器のヒット情報を受け取り、トリガー判定に使用する p_{T} の概算を行う。

セクターロジックのトリガー系は複数のモジュールで構成され、ファームウェアは Verilog HDL や VHDL などのハードウェア記述言語 (hardware description language、HDL) で記述される。HDL で記述されたコードをシミュレートするためのアプリケーションとして Xilinx 社が Vivado シミュレータ（図中央の青箱、XILINX 2022）を提供しているが、各モジュールの全ての結線についてクロック毎の状態遷移を厳密にシミュレートするものであるため、多くのヒットパターンを試験するのには時間がかかる[*2]。このため、Vivado シミュレータとは別に、トリ

[*2] URAM への初期設定を行うのにかかるオフセットを含めて、63 イベント (1 度の試験の単位となるイベント数。ヒットパターンを仕込む URAM の深さによる制限) のシミュレーションに、約 10 時間程度かかる。

ガー系のロジックのみを再現する速度の速いシミュレータが必要となった。また、デバッグのためには、多段で構成されるアルゴリズムの途中経過も容易に取り出すことができるツールも不可欠である。このビットワイズシミュレータは、モジュールの単位で、実機と全く同じロジックと入出力を有するものである。このソフトウェアシミュレーション実装の研究について 5 章で述べる。

3.2　②テストパターンファイルの生成機構

図 3.1 の上半分は、実機またはシミュレータに入力するヒットパターンファイルの生成フローを示している。この仕掛けでは、無限運動量トラックのように人為的なものから、実データ・MC データを起点とした任意のパターンに対しての実機の応答を試験できることが重要である。

後述するリレーショナル・データベースを用いることで、セクターロジックのためのテストベクター作成機構を構築した。これらのテストベクター生成の詳細は 6 章に記述する。

リレーショナル・データベースの構築

テストベクター生成機構の開発及びセクターロジックトリガーシステムの開発のために、検出器からセクターロジックまでの複雑なケーブリングを系統的にハンドルする仕掛けが不可欠である。TGC 検出器は既存のものを活用するため、現行システムの精査とチャンネル番号を再定義し、ケーブリングデータベースへの実装を行なった。

前述の通り、セクターロジックへの入力は 128 bit×58 link の形式のビットマップであり、トリガー系で取り扱うためには、まず最初にロジカルなチャンネル番号に並び替える必要がある。この機能を担うのがセクターロジックトリガー系の最前段部に設置される「Channel Mapping」モジュールである。この「Channel Mapping」モジュールの自動生成システムと、テストベクターの生成システムを作成するために、ケーブリング情報と現行システムにおける Wired OR (η 方向における、チャンネルのオーバーラップや、ステーション間での各チェンバーを覆う η の範囲に違いを考慮するために必要な OR 処理) を精査し、チャンネル番号の再定義を行った。このトリガーロジックを元に再定義した情報とケーブリング情報をデータベースへ実装した。ここで作成したデータベースはケーブリング及びチャンネルに関わる情報を一元管理するためのものであり、Channel Mapping のファームウェアもこの定義にしたがって作成されている。

また、データベースはオープンソースのリレーショナルデータベース管理システムである MySQL を用いて作成したが、MySQL には Python との API が存在している。この API を利用することでファームウェアの自動生成とテストベクターの生成システムが実現した。データベースの詳細は 4 章及び付録 B に記述する。

3.3　③出力同士の検証・解析

3 つのパスの出力の比較検証

図 3.1 の検証システムでは、上記のテストベクターによる試験がシミューレーション、ファームウェア、実機に渡って横断的に実現されることが肝要である。この要請から、新たに開発したビットワイズシミュレータへのテストパターンの入力はファームウェアと全く同一のファイル形式を採用した。同一の入力を 3 つのパスに流すため、出力同士の比較が可能になり、実機や各シミュレータの系統的な検証を行うことが可能となっている。

MC データを用いた試験と検証

MC データから生成したテストパターンを用いた試験では、Truth 情報 (ミューオンの飛跡情報) を用いた解析を行うことができる。この検証システムでは、MC データの入った ROOT ファイルから json ファイル (テストパルスパターンファイルを生成するための前段階のファイル) を作成する際に、同時に Truth 情報を保持したファイル

(図 3.1 内、「Truth.root」) を生成し、シミュレータ等の出力を解析する際に使用する[*3]。

[*3] MC データから json ファイルを生成する際にミューオンの p_{T} や η でイベントを選別することができるため、選別されたイベントの入った json ファイルと 1 対 1 対応となる Truth のファイルを生成しておく必要がある。

第 4 章

リレーショナル・データベースによるケーブリング情報・コンポーネント情報の一元管理

エンドキャップミューオントリガーシステムは 32 万チャンネルから構成される。各チャンネルのヒット情報は専用エレクトロニクスによって実現された多段の論理回路によって処理されるが、これらすべての接続情報を一元的に管理するリレーショナル・データベースを MySQL によって作成した。このリレーショナル・データベースは L0 エンドキャップミューオントリガー系のケーブリングの他、各チャンネルの性質に関する履歴情報や、コンポーネント間のヒット情報転送に使用されるデータフォーマット、コインシデンスロジック等の、様々な種類の情報を一元管理するものである。

この章ではリレーショナル・データベースに含まれている情報の範囲と、データベースを作成するにあたり定義したチャンネル番号について記述する。データベース内の表や構造の詳細については付録 A、B に記述する。

このデータベースによって、5 章で記述するモジュールの一つである Channel Mapping モジュールの生成と、6 章で記述するテストパルスパターンの生成システムが実現した。また、このデータベースは本研究での利用目的であるファームウェアとテストベクターの生成に限らず、幅広い目的に応用できるものであり、ATLAS 実験共同研究者が活用可能なインフラストラクチャーとして設計されている。

4.1 リレーショナル・データベースを使用する利点

MySQL（MySQL 2022）はオープンソースのリレーショナルデータベース管理システム (Relational DataBase Manegement System、RDBMS) である。リレーショナルデータベースは 1 つのデータを複数の項目の集まりで表す。例えば、検出器上のチャンネルと ASD の対応関係を表すものでは、「検出器チャンネル」「ASD の名前」「ASD のチャンネル番号 (0-15)」の 3 つの項目からなる。この 3 つの項目の組をレコードと呼び、項目をカラムと呼ぶ。レコードの集合はテーブルと呼ばれ、1 つのデータベースは複数のテーブルによって構成される。レコードを一意に決める項目（一つまたは複数の項目の組み合わせ）を主キーと呼び、主キーの中身が重複するレコードをテーブルに入れることはできない（西沢 2017）。

データベース内で表現したいケーブリングには階層性がある。前述の「検出器上のチャンネルと ASD の対応関係」を表すテーブルでは、「検出器チャンネル」と「ASD の名前」+「ASD のチャンネル番号 (0-15)」が 1 対 1 対応の情報となり、レコードはチャンネルごとの単位となる。しかし、ヒット情報の流れの下流に当たる「ASD と PS ボードの対応関係」を表すテーブルでは、「ASD の名前」が「(PS ボード上の)FPGA 番号[*1] 」+「PPASIC 番号[*2] 」の組み合わせが 1 対 1 対応となるレコードによって記述される。ASD 1 つは 16 チャンネルに相当するの

[*1] FPGA は PS ボード 1 枚につき 1 つなので、PS ボードの番号と言い換えられる。定義域は 1 から 29。
[*2] PS ボードにつき 8 つあるので、定義域は 1 から 8。

表 4.1　TGC 検出器内のチャンネルと ASD チャンネルの対応関係を表す表の抜粋。縦に同じカラムの情報が並んでおり、一番上の行がカラムのタイトルまたは内容を示している。2 行目以降は、1 行が 1 つのレコードを示す。TGC 検出器側の情報を黄色、ASD 側の情報をオレンジで表している。最も左のカラムは エンドキャップのワイヤーのレイヤー 1 を示しており、定義域は 1 から 196 である。これがこのテーブルの主キーである。中心のカラムは ASD のチャンネル番号を表しており、一つの ASD ボードが処理できるのは 16 チャンネルなので、定義域は 0 から 15 である。右のカラムは ASD の名前を表している。

Endcap レイヤー 1 のチャンネルの通し番号 (196-1)	ASD のチャンネル番号 (0-15)	ASD_name
196	7	EW0-1
195	6	EW0-1
194	5	EW0-1
193	4	EW0-1
192	3	EW0-1
191	2	EW0-1
190	1	EW0-1
189	0	EW0-1
188	15	EW1-1
187	14	EW1-1

で、前者のテーブルと後者のテーブルではレコード 1 つあたりの単位が 16 倍異なっている。これがケーブリングの有する階層性である。

このような階層性を正しく取り扱うためには、情報を複数のテーブルに分割して処理することのできるリレーショナル・データベースを用いるのが最適である。表 4.1、表 4.2 に前述した二つのテーブルの抜粋を示す。もしこの二つのテーブルを表計算ソフト上で 1 枚の表として取り扱おうとする場合、表 4.3 のように、より大きな構造である前者に属する項目「ASD の名前」、「FPGA 番号 (1-29)」、「PPASIC 番号 (1-8)」は、チャンネルに関する部分 (「検出器チャンネル」、「ASD のチャンネル番号 (0-15)」) のみが違う同じ組み合わせが重複して記述されることになる。このような一つの表のみによる情報の管理方法は冗長性が高く、情報を更新する過程でミス[*3] が生じやすい。このような理由から、階層性のある情報を取り扱うとき、情報を管理する立場では、最小限の表に分割するのが合理的である。このように、どのカラム (一つまたは複数のカラムの組み合わせ) の情報によってどのカラムの情報が決まるか[*4] という構造を整理し、データの冗長性を廃し、データ全体を矛盾なく一貫して取り扱えるようにデータベースを設計することを、データベースの正規化と呼ぶ。

しかし、情報を利用する上では、「検出器のチャンネルとセクターロジックの入力」[*5] などの、複数の対応関係を跨いだ関係性の検索が重要になる。リレーショナル・データベースは、上記のように分割した表をシステム上で繋ぎ合わせ、仮想的な一枚の表として検索することができる。表の結合は、表 4.1、表 4.2 から表 4.3 を作るときのように、2 つの表が共通して持っているカラム (この例では「ASD_name」) を利用する。この複数の表をまたがる検索は、SQL 単体で検索を行うときと、Python との API を用いて検索を行うときの、どちらでも利用可能である。この研究では、Channel Mapping モジュールを生成するときは「検出器のチャンネルとセクターロジックの入力」の対応関係を用い、無限運動量飛跡のテストパルスパターンを生成するときはさらに「検出器のチャンネルと座標 ID」を加えて「η、ϕ 座標 ID とセクターロジックの入力」の対応関係を用いた。

[*3] 具体例: ASD と PS ボード間の接続情報が変更した場合を考える。ASD と PS ボードの間のケーブルが 2 本、交換して接続されるようになった場合、リレーショナル・データベースで情報を管理しているときは、表 4.2 のレコードを 2 つだけ変更すればよい。対して、表 4.3 のように 1 つの表で管理している場合は、ASD チャンネルの数だけ倍増したレコードを変更しなければならなくなる。

[*4] これは関数従属性と呼ばれる (吉川 2019)

[*5] これは 3 種類の階層の表を結合することで得られる対応関係である

表 4.2　ASD と PS ボードの接続情報を表すテーブルの抜粋。ASD 側の情報をオレンジ、PS ボード側の情報を緑色で表している。最も左のカラム「ASD_name」は表 4.1 と同じく、ASD の名前を表す (NC は ASD が接続していないことを示す)。カラム「phi(0/1)」は ϕ 方向の対称性を区別するために存在している (この表は 1/24 繰り返し構造内の対応関係を表している。この領域ではエンドキャップのチェンバーが 2 回対称構造になっているため、ϕ0 の ASD と ϕ1 の ASD を区別する)。カラム「port」は PS ボード上の PPASIC のポート A/B を区別し、カラム「PPASIC」は PS ボード上にある 8 つの PPASIC の内のどれであるかを示す。カラム「FPGA」は PS ボードごとに 1 つ設置されている FPGA を示す。
この表ではオレンジ色のカラム 2 つの組み合わせと、緑色のカラム 3 つの組み合わせが 1 対 1 対応になる。ただし、ASD と接続していないポートが存在するため、緑のカラム 3 つが主キーとなる。

ASD の名前	phi (0/1)	port (A/B)	PPASIC (1-8)	FPGA (1-29)
EW0-1	0	A	1	1
EW0-2	0	B	1	1
EW1-3	0	A	2	1
EW0-3	0	B	2	1
EW3-3	0	A	3	1
EW2-3	0	B	3	1
EW3-1	0	A	4	1
EW3-2	0	B	4	1
EW1-1	0	A	5	1
EW1-2	0	B	5	1
EW2-1	0	A	6	1
EW2-2	0	B	6	1
NC	NULL	A	7	1
NC	NULL	B	7	1

表 4.3　表 4.1、表 4.2 を一つの表にまとめたもの。右側の黒線で囲った部分では、ASD と PS ボード間の接続情報で同じ内容 (上は「EW0-1、A、1、1」、下は「EW1-1、A、5、1」) が繰り返されている。

Endcap レイヤー 1 のチャンネルの通し番号 (196-1)	ASD のチャンネル番号 (0-15)	ASD_name	port (A/B)	PPASIC (1-8)	FPGA (1-29)
196	7	EW0-1	A	1	1
195	6	EW0-1	A	1	1
194	5	EW0-1	A	1	1
193	4	EW0-1	A	1	1
192	3	EW0-1	A	1	1
191	2	EW0-1	A	1	1
190	1	EW0-1	A	1	1
189	0	EW0-1	A	1	1
188	15	EW1-1	A	5	1
187	14	EW1-1	A	5	1
186	13	EW1-1	A	5	1
185	12	EW1-1	A	5	1
184	11	EW1-1	A	5	1
183	10	EW1-1	A	5	1

以上のように、正規化されたリレーショナル・データベースは合理的なデータ構造と優れた検索性を有するため、従来の Excel などの表計算ソフトのファイルの集合よりも、データを一元管理するにあたり優れている。

さらに、MySQL では上記の検索で用いた表の結合の仕方もデータベース内に保存することができる。このデータベースでは MySQL の「ビュー」という機能によって、複数枚のテーブルを仮想的につなげたものを簡単に取り扱えるようにしている。このビューについては付録 C で記述する。

4.2　現行システムの精査とチャンネル番号の再定義の研究とケーブリングデータベースへの実装

この節では新しく定義したチャンネル番号について記述する。

リレーショナル・データベースを作成するにあたり、複数のエクセルファイルや回路図の形で共有されていたコンポーネント同士の 1 対 1 対応のケーブリング情報をまとめるだけでなく、現行のトリガーシステムの精査を行い、チャンネル番号の再定義も行った。高輝度 LHC に向けたアップグレードでは、トリガー論理回路やケーブリングは全て一新されるため、新しい論理回路に即したチャンネル再定義を行うことは大変重要である。一方で、TGC 検出器は既存のものを活用するため、現行システムの検出器・トリガー論理回路の設計を精査し、設計の意図を正確に理解し内在する知見を、すべて活用しきる形でチャンネルを再定義することが肝要である。このチャンネルの再定義では、特に現行システムにおける Wired OR (η 方向における、チャンネルのオーバーラップや、ステーション間での各チェンバーを覆う η の範囲に違いを考慮するために必要な OR 処理) やレイヤー間でのチャンネル番号の相関に注意した。再定義に加え、新しいエレクトロニクスのピンアサインメント、ファイバーケーブリング、データフォーマット等の入出力の情報を実装してリレーショナル・データベースを構築した。このリレーショナル・データベースにより、5.1 節で記述する Channel Mapping モジュールの自動生成と、6 節で記述するセクターロジックへのテストベクターの自動生成システムの構築が実現した。

この研究では「スタッガードチャンネル」と「ステーション内コインシデンス入力」の 2 種類のチャンネルの定義を行った。定義の詳細を記述する前に、以下にそれぞれのチャンネル定義の目的を述べる。

前者の「スタッガードチャンネル」定義の主な目的は、テストベクターの自動生成システムへの利用である。これはファームウェア内での処理で使用されるチャンネルの単位に対して新しく番号を振ったものだが、同じ座標のチャンネルに対して同じ番号を振ることによって、無限運動量飛跡を簡単に生成できるようにした。

後者の「ステーション内コインシデンス入力」の定義は、Channel Mapping モジュールの制作を主目的として、検出器上のチャンネルに対して物理的な配置順と Wired OR を元に番号を振ったものである。Channel Mapping モジュールはトリガー系最前段部に配置するファームウェアであり、セクターロジックへ入力される 128 bit×58 link の形式のビットマップをトリガー系で取り扱いやすいロジカルなチャンネルに変換するものである。この変換は規則性に乏しいため、ファームウェアでは変換前後のチャンネルの対応関係を (for ループ文などを使用せず)1 チャンネルにつき 1 行ずつ記述する形でリレーショナル・データベースから自動生成した。これに対し、Channel Mapping モジュールの次に配置するモジュールは、1 つのステーション内に存在する 3 層または 2 層のレイヤーに対してコインシデンスを行うものである。このコインシデンスは規則性が良く、(サブセクターの端を除き、) 同じ関数を使用してサブセクター内の全てのチャンネルを処理することができる (トリガーロジックの詳細は 5.2 節に記述する)。よって、チャンネルの検出器上での物理的な配置に起因する、トリガーシステム上での規則的ではない処理は、Channel Mapping モジュールで担う方が都合が良い。以上の理由で、再定義したチャンネルをリレーショナル・データベースに組み込むことによって Channel Mapping モジュールに吸収した。

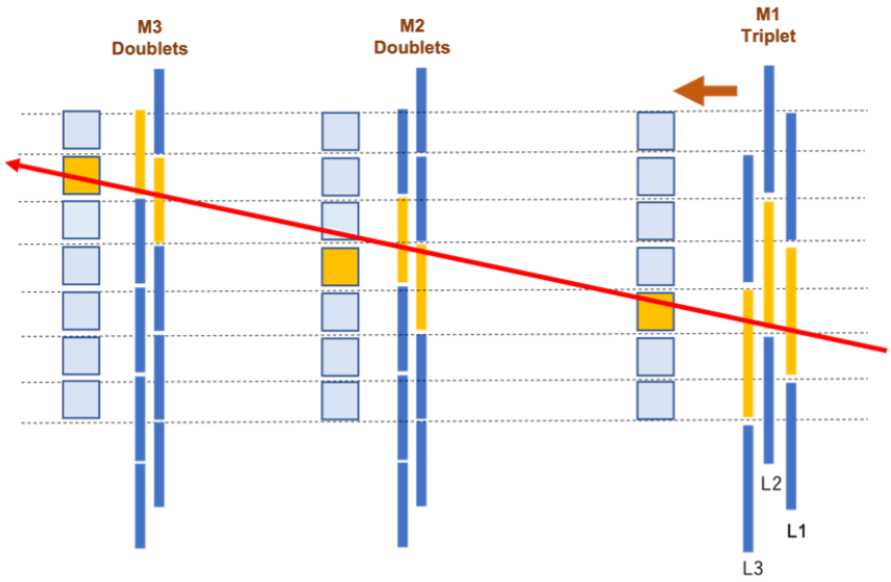

図 4.1　検出器チャンネルのスタッガリング構造。縦棒が一つの検出器チャンネル、四角がコインシデンスをとった後のスタッガードチャネル、赤の矢印はミューオンを表す。ミューオンが通過したチャンネルが発火 (黄色) し、発火したチャンネル同士でコインシデンスを取ることでスタッガードチャンネルの代表点を得る。

4.2.1　スタッガードチャンネル

TGC 検出器は 3 つのステーションで構成される。ヒット情報は、ステーション間での座標の差分から曲率を計算する前に、ステーション内でコインシデンスにかけられる。このステーション内コインシデンスで得られた代表点は、検出器内のチャンネルよりも細かな位置情報を持つ。MySQL で製作したデータベース内では、この位置情報をスタッガードチャンネルと呼称している。

ステーション内において、検出器チャンネルは、隣のレイヤーのチャンネルと意図的にずらした状態で配置されている（図 4.1 の縦棒部分)。具体的には、レイヤーが 3 層重なっている M1 では 1/3 ずつ、レイヤー 2 層からなる M2、M3 では半分ずつ検出器チャンネルがずれてグルーピングされている。この構造により、ステーション内で 2 層中 2 層または 3 層中 3 層のコインシデンスを取ると、ヒット情報を得た検出器チャンネルの組み合わせから、M1 では検出器チャンネル自体の 1/3 の位置分解能、M2M3 では検出器チャンネルの半分の位置分解能での位置情報を得ることができる（図 4.1 の四角部分)。この代表点をスタッガードチャンネルと呼ぶ。このスタッガードチャンネルにおける η の位置分解能が全てのステーションで等しくなるように設計されているため、4.1 に示すように、検出器チャンネル一つあたり η の幅は M1 : M2 : M3 で 3 : 2 : 2 となっている。

スタッガードチャンネルの通し番号を明示的に定義したドキュメントは存在していなかったため、本研究で新しく定義を行なった。定義にあたり、全ステーションにおいてスタッガードチャンネルのグラニュラリティが同一であるという検出器の設計を尊重し、同じ番号のチャンネルが同じ η になるようにした。この定義によって、スタッガードチャンネルをデータベースの検索条件にすることで直線飛跡となるチャンネルの組み合わせを簡単に得ることができるようになり、6 章のテストベクター作成機構において、無限運動量飛跡を作れるようになった。

図 4.2 は定義の際に作成したエクセル表の抜粋である。この番号はそれぞれエンドキャップ内、フォワード内での通し番号であるため、後述の SQL によるデータベースでは、ワイヤー/ストリップ、エンドキャップ/フォワードの 4 通りの組み合わせに応じた 4 枚の表を作成した。M1（右)、M2（中央)、M3（左）のステーションごとにま

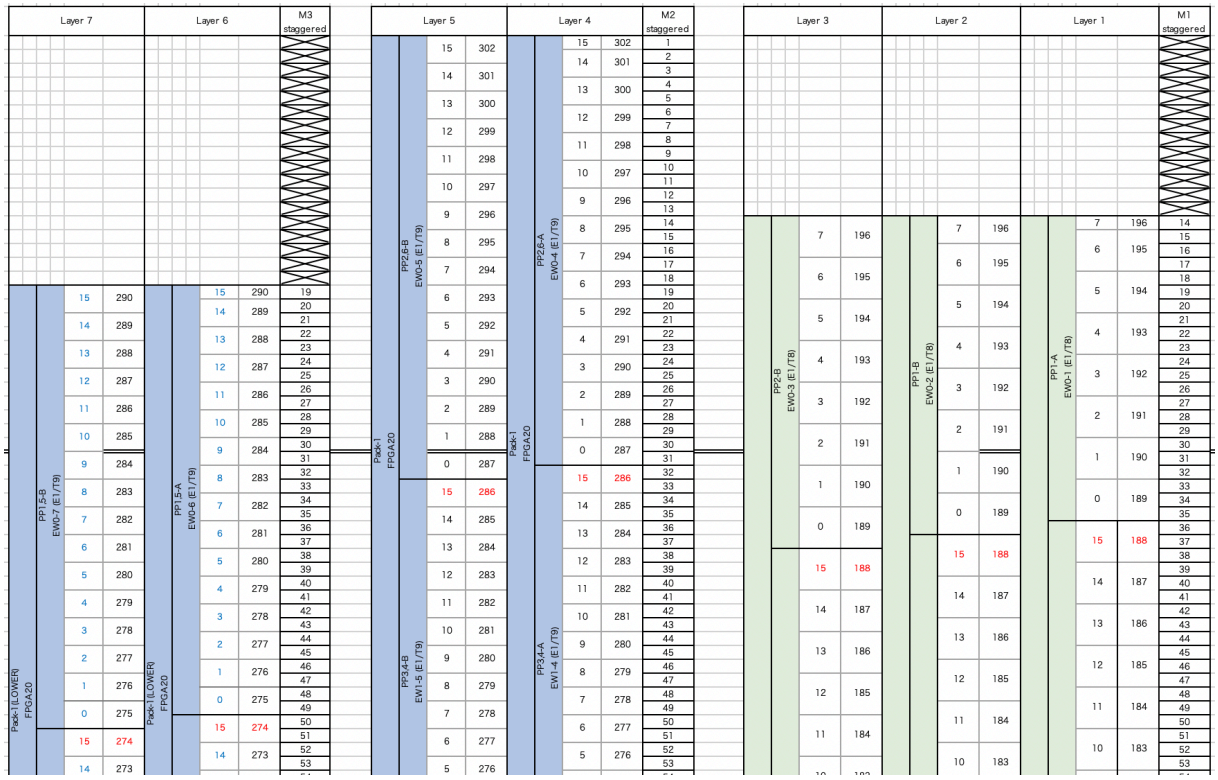

図 4.2　検出器チャンネルとスタッガードチャンネルの対応表（エクセル）。エンドキャップワイヤーチャンネルにおける、η が最も小さい部分の抜粋

とめて記述している。各ステーション内の一番右端、最も細かいセルの中の番号がスタッガードチャンネルである。その次に細かい白いセルの中の番号はレイヤー内の通し番号であり、これは Run3 のエレクトロニクスのものを踏襲している。色のついたセルの塊の中には、Pack、FPGA 番号、PPASIC、ASD 名を記入した。このエクセル表はリンク (ATLAS Internal webpage) の「TGC-ASD: Staggered Channel」で公開している。

TGC エレクトロニクスの設計では、慣習上、ビーム軸側（$\mathrm{R} = 0$, $\eta = \infty$）が 0 番になるようにチャンネルの番号が振られているため、Run3 のエレクトロニクスのものを踏襲したレイヤー内の通し番号もビーム軸側が若い番号になっている。対して、解析やファームウェアのデザイン研究で使用されるソフトウェアの方ではビーム軸から遠い側（$\eta = 0$）が若い番号になるように慣習上定義されている。この違いを吸収するため、レイヤー内の通し番号までをエレクトロニクス準拠のビーム軸側が若い番号の振り方にし、次の段階となるスタッガードチャンネル（= ステーション内コインシデンス出力）からビーム軸から遠い側から番号が振られるように定義し、新しいファームウェアの設計もその番号付けを尊重する形で信号線名を定義することに定めた。

4.2.2　ステーション内コインシデンスへの入力

「ステーション内コインシデンスへの入力」は、ワイヤーに関しては、現行システムで使用されている「レイヤー内でのチャンネルの通し番号」を踏襲して使用したが、ストリップに関しては新たに番号の定義を行った。

Wired OR とは、η 方向におけるチャンネルの重複を考慮した、チャンネルの OR 処理のことである。この Channel Mapping ではこの処理を含めた出力として「ステーションコインシデンス入力チャンネル」を定義した。この処理はワイヤーチャンネルとストリップチャンネルで考え方が異なる。ワイヤーチャンネルではステーション内でのコインシデンスを行う前に Wired OR が必要になり、ストリップチャンネルではステーション間でのコイン

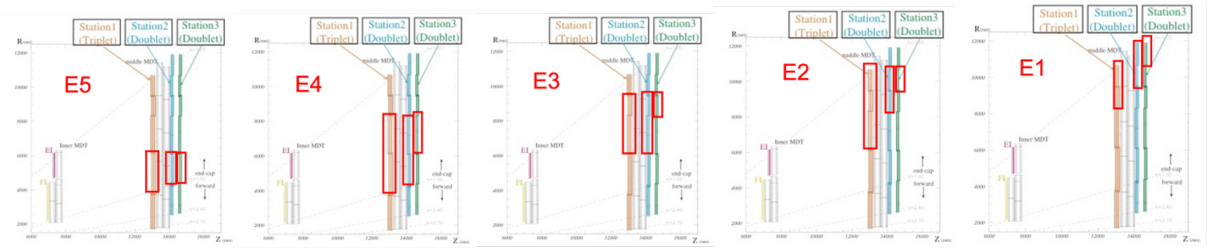

図 4.3　M3 のチェンバーとコインシデンスをとるべき M1・M2 の領域の対応関係(大町千尋 2006)。

シデンスを考えるにあたり、Wired OR の考慮が必要になる。

ワイヤーのステーション内コインシデンスへの入力

ワイヤーに関しては、Run3 で定義されていたレイヤー内でのチャンネルの通し番号が Wired OR が考慮されたものであったため、これを踏襲してワイヤーのステーション内コインシデンスへの入力とした。

ワイヤーチャンネルにおける Wired OR は、チェンバー境界のオーバーラップしたチャンネルに対する OR である。TGC 検出器のエンドキャップ領域のサブセクターは複数のチェンバーで構成されるが、チェンバー境界に不感領域を作らないために、チェンバー端が重なり合うような形で設置されている。この重なりあった、同じ η を見ているチャンネル同士で OR を取る処理を行っている。Run3 のエレクトロニクスで定義されているワイヤーチャンネルのレイヤー内通し番号も、Wired OR を取った後のものであり、ワイヤーの Channel Mapping モジュールに関してはこの定義を引き継いでいる。

ストリップのステーション内コインシデンスへの入力

ストリップでは、ステーション間でのコインシデンスを取る時、ステーション間でのチェンバー境界の η 座標が一致しないことと、内部のトロイド磁場により飛跡が η 方向に曲がることを考慮して、M1、M2 では複数のチェンバー同士で OR を取り、チェンバーをまたがる仮想的な長いチャンネルとして取り扱う必要がある (図 4.3)。

ストリップの各チャンネルは、チェンバーの長さの分の η をカバーしている。同じ ϕ を観測するストリップチャンネルがチェンバーごとに分断されており、またステーションごとに各チェンバーが覆う η の範囲が異なっている。さらに、磁場によってミューオンの飛跡が曲がるため、ピボットの M3 ステーションのチェンバーよりも広い範囲の η 領域に対してコインシデンスを行う必要がある。このため、ステーション間のコインシデンスでは 1 枚の M3 チェンバーにつき、M1M2 では複数のチェンバーを参照する必要がある[*6]。図 4.4 は M3 のチェンバーに対し、コインシデンスを取る M1・M2 のチェンバーの範囲を表している。これらのコインシデンスの関係性は、M2・M3 間のコインシデンス範囲に関しては Run3 の PS ボードの回路図(ATLAS Collaboration)から、M1・M3 間のコインシデンス範囲に関しては Run3 の HPT の結線から確認し、それを踏襲した。

前述の通り、ワイヤーに関しては Run3 のエレクトロニクスで定義されている番号を踏襲し、ステーション内コインシデンスチャンネルに用いた。一方、ストリップではチェンバーごとの 1-32 の通し番号しか定義されていないので、新たに「ステーション内コインシデンス入力チャンネル」の定義を行い、これを Channel Mapping モジュールの出力とした。この定義を行う際に、上述の Wired OR を考慮した。

M3 ステーションは 5 枚のチェンバーで構成されているため、1 チェンバーあたりストリップチャンネルが 32 本

[*6] 飛跡再構成を行う後段のモジュール「Segment Reconstruction」では各ステーションの代表点の組み合わせによって、ワイヤーとストリップでそれぞれ飛跡の再構成を行うが、このモジュールの出力は「M3 の位置情報 + 飛跡の角度 $\Delta\theta$ (または $\Delta\phi$)」である。このように、飛跡の座標はトリガー系において M3 のピボットを基準にして取り扱われるため、ストリップにおける「ステーション内コインシデンス入力」の番号は「どの M3 ステーションのチャンネルと対応するか」で定義した。

図 4.4　M3 のチェンバーとコインシデンスをとるべき M1・M2 の領域の対応関係およびステーション内コインシデンス入力のチャンネル番号とスタッガードチャンネルのチャンネル番号の定義

あることから、全体では 160 までの通し番号を振ることができる。これに対し、M1・M2 ではチェンバーを跨いで OR をとった、仮想的な長いチャンネルに対して 1-160 の通し番号 (＝ステーション内コインシデンスインプット) を振る（図 4.4)。具体例を挙げると、M3 の 64 番は E3 チェンバーに存在するため、M2 の 64 番は E3・E4 で OR を取ったものに対して振られる。このストリップチャンネルの OR の処理は、トリガー論理回路の最初段である Channe Mapping モジュールにおいて行われるため、以降の回路は、検出器境界の処理や、ステーション間の構造の違いを意識せず正確な設計をすることが可能になる。

第 5 章

セクターロジックトリガー系のビットワイズシミュレータの開発

セクターロジックのトリガー系では、TGC 検出器のワイヤーの 7 レイヤー、ストリップの 6 レイヤーで観測したヒット情報からミューオンの飛跡を再構成し、Look Up Table を用いて運動量を推定して後段へ $(\eta, \phi, p_{\mathrm{T}})$ を送

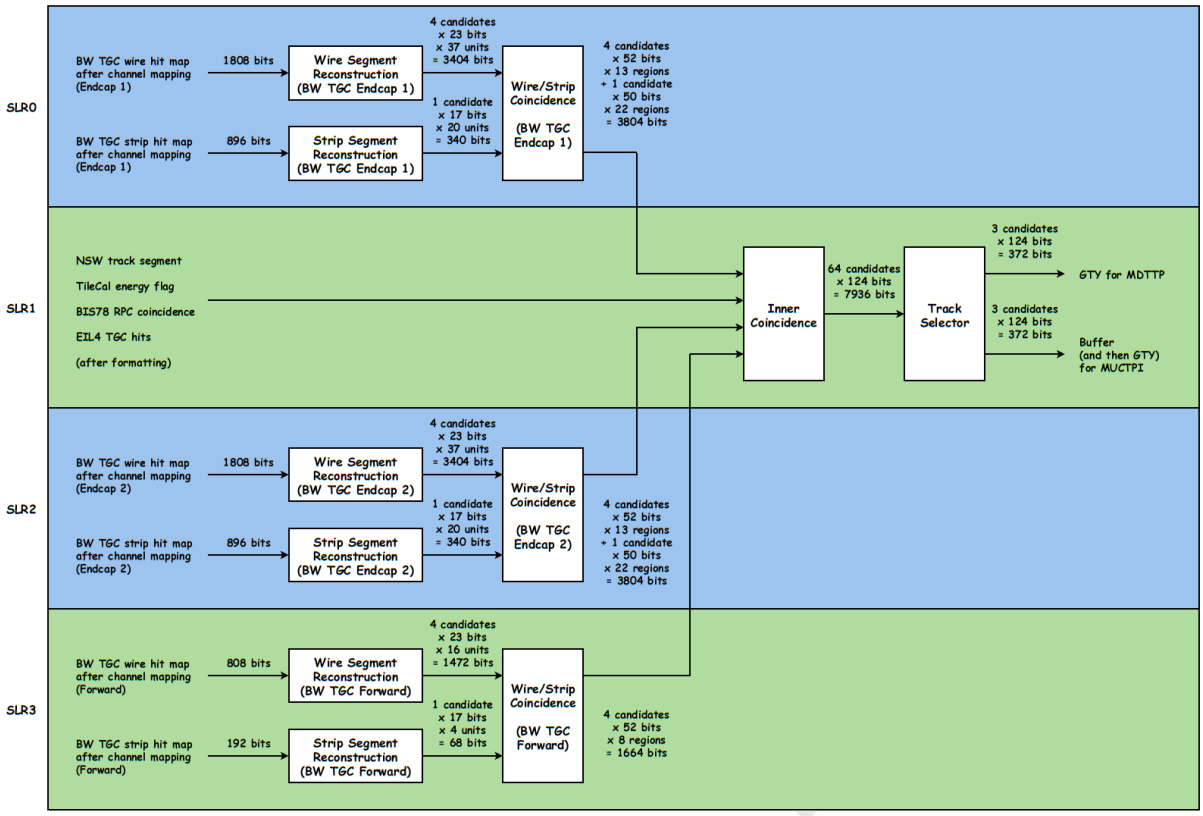

図 5.1　セクターロジックトリガー系のダイアグラム（ATLAS Collaboration 2021 の figure2.7）。セクターロジックボードで使用する FPGA の XCVU13P は 4 つの Super Logic Region (SLR) で構成されており、各 SLR が一つのサブセクター[*2] を担う。Wire-Strip Coincidence までは各 SLR での処理が 2 本に分かれているのは、ワイヤーチャンネルとストリップチャンネルがそれぞれ独立に処理していることを示している。Wire-Strip Coincidence の後に Inner Coincidence があり、この段階から 3 つのサブセクターの情報と EIL4 などの内部の検出器の情報を併せて処理する。

信する。この SL トリガー系は複数のモジュールで構成されており (図 5.1)、各モジュールは京都大学や名古屋大学などの複数の大学で分担して作成されている。現在はそれらのモジュールの統合試験が行われている(三島章熙他)。

本研究ではこのセクターロジックのトリガー系を再現するビットワイズシミュレータを C++ で開発した。このシミュレータは、モジュールのロジックと入出力がファームウェアと一致するように作成しており、C++ 上では各モジュールに対応するクラスを作成した。このシミュレータはフロートによる近似ではなくビットによる計算を行うものであり、また、トリガーの制約に起因するファームウェアの構造も全て一致するように実装した。具体的には、レイテンシーの制約により、ファームウェアにおけるステーション間でのコインシデンスではありうる全てのヒット点の組み合わせを計算することはできず、優先順位をつけて上位複数のみの計算をしているが、この優先順位の付け方と上限の数がファームウェアと一致している。また、ステーション間コインシデンスをとるモジュール (Segment Reconstruction、Wire-Strip Coincidence) ではモジュール内ではサブセクターを複数の領域に更に分割し、独立に並列計算を行なっているが、この領域の分割も同じである。

以下ではこのシミュレータの各クラスの機能について、前段から後段の順で記述する。

5.1　Channel Mapping

Channel Mapping モジュールはこのシミュレータの最前段に位置するクラスである。アップグレードでは、エンドキャップミューオン初段トリガーを全てバックエンドで行うため、全ヒット情報が SL へ入力される。BW の 1/24 繰り返し構造につき 1 枚の SL が存在し、各セクターロジックは 29 枚の PS ボードから各 2 本のリンクによってヒット情報が送信される。1 つのリンクから 25 ns につき 128 bit の情報を受信するデータフォーマットとなっており、セクターロジックが TGC 検出器から受け取る情報は 128 bit × 58 links のビットマップの形式になる。Channel Mapping モジュールは 128 bit×58 links のビットマップの情報からステーション内コインシデンスへの入力を作る工程を担う。この工程を Mapping と呼ぶ。変換後のチャンネルは、「サブセクター[*3] 」×「レイヤー[*4] 」×「レイヤー内でのチャンネルの通し番号[*5] 」で表される。この Mapping 後の整理された情報を「ステーションコインシデンス入力チャンネル」と呼ぶ。この工程では単にチャンネルを並び替えるだけでなく、この変換の際に、並べ直しだけではなく、検出器を跨いだ信号処理となる Wired OR と呼ばれる処理も行う。最初段のハードウェアの制約に起因する不規則を全て吸収するデザインをとることで後のコインシデンスロジックの設計が、検出器境界等のハードウェアの制約を意識せずに遂行できるように工夫した。

ファームウェア及びシミュレータの作成

このモジュールに限り、この研究で c++ シミュレータだけでなくファームウェアを作成した。このファームウェア及びシミュレータは、どちらも前述のリレーショナル・データベースと Python の API によって自動生成した。図 5.2 にこのモジュール及びクラス作成のフローを示す。主体となるのは Python であり、既存の API によってデータベースから Channel Mapping モジュール作成及びクラスに必要な情報を引き出す。引き出した情報を Python スクリプト内で整形し、VHDL で記述したファームウェア及び、シミュレータに読み込ませるデータファイルを作成した。トリガー系のファームウェアはサブセクターごと (=SLR ごと) に配置されるため、エンドキャップとフォワードの 2 種類を作成した。

Listing 5.1　Channel Mapping 部分のファームウェア (VHDL) の抜粋

```
1 OUTPUT_ENDCAP_PHI0_WIRE_L1(151) <=  INPUT_BITMAPS_E_phi0(11)(124);
```

[*3] Endcap ϕ0、Endcap ϕ1、Forward の 3 つ
[*4] ワイヤーの 7 レイヤーとストリップの 6 レイヤーの合計 13 枚
[*5] レイヤーによりチャンネル数が異なる

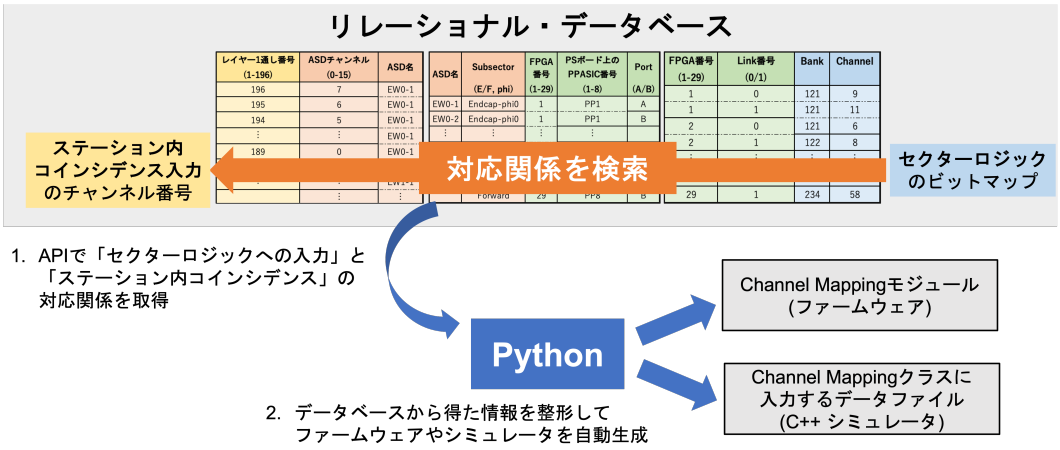

図 5.2　Channel Mapping モジュール及びクラス用ファイル作成のフロー。

```
2 OUTPUT_ENDCAP_PHI0_WIRE_L1(152) <= ( INPUT_BITMAPS_E_phi0(11)(53) or
  INPUT_BITMAPS_E_phi0(11)(125) );
3 OUTPUT_ENDCAP_PHI0_WIRE_L1(153) <=  INPUT_BITMAPS_E_phi0(11)(54);
```

Listing5.1 に、Endcap ϕ 0 領域のファームウェアの抜粋を示す。右辺の「INPUT_BITMAPS_E_phi0」が入力されたビットマップを表す二次元配列であり、左から一つ目の括弧の内側のインデックスがリンク番号[*6] 、二つ目の括弧の内側のインデックスが bitmap 内のポジション番号 (0-127) を表す。左辺は「OUTPUT_ENDCAP_PHI0_WIRE_L1」が出力となる変換後のチャンネルであり、レイヤーによってチャンネル数が異なるため、レイヤーごとに 4 章で議論した定義に従い、1 次元の配列を用意した。Listing5.1 は、ワイヤーのレイヤー 1 に相当する部分であり、第 2 行で後述する Wired OR を取っている。Endcap ϕ 0 領域には約 3000 チャンネルが存在しているため、このファームウェアは約 3000 行の Listing5.1 に抜粋したような記述によって構成される。自動生成されるため、「3000 行に及ぶハードコードが必要となる箇所」も間違いなく準備が可能であり、将来の何らかの変更 (ケーブリングスキームの変更等) にも柔軟に対応できる。このように、複雑なケーブリングが付随する大量のチャンネル情報をリレーショナル・データベースと Python の API によって合理的に取り扱える基盤を作成した。

5.1.1　ビットワイズシミュレータへの入力データの自動生成

一方、図 5.2 に示した二つ目の出力ファイルは、csv 形式の C++ シミュレータへ入力するファイルである。このファイルはファームウェアにおける入力にあたるリンク数、bitmap 内のポジション番号と、出力にあたるステーション内コインシデンス入力の組み合わせを記録したものである。シミュレータを走らせるときは、イベントの処理をする前のセットアップのタイミングでこのファイルを読み込む。

5.2　Intra station coincidence(ステーション内コインシデンス)

Channel Mapping で整形されたヒット情報に対して、ステーション内でコインシデンスをかけてノイズを落とすとともに、チャンネルの組み合わせによって 4.2.1 節で記述したスタッガードチャンネルによる代表点と、コインシ

[*6] エンドキャップでは 0~23、フォワードでは 0~9

デンスが成立したレイヤー数の情報を出力する。コインシデンスが成立したレイヤー数の情報は、次の「Segment Reconstruction」で飛跡の再構成を行う時の優先順位として使用する。

図 2.10 に示したように、ワイヤーとストリップに対して、それぞれ M1、M2、M3 のステーション内でコインシデンスの処理を行うので、一つのサブセクターの Intra station coincidence モジュールは、大まかに 6 つに分けることができる。ただし、ステーション間の対称性から、Intra station coincidence の処理を行う関数として作成したのは、ワイヤー 3 種、ストリップ 1 種の合計 4 種類のみである[*7]。

Run3 までは、例えば 3 枚のレイヤーからなる Wire の M1 ステーションでは、3 層中の 2 層以上にヒットがあった場合 (2/3 コインシデンス)、コインシデンスが成立していた[*8]が、Phase2 アップグレードからは、「1/3 コインシデンス」、「2/3 コインシデンス」、「3/3 コインシデンス」の 3 種類を独立かつ排他的にに取り扱う、より柔軟なトリガーになった。[*9]

このコインシデンスロジックの詳細については、付録 D に記述した。

5.3　Segment reconstruction

M1、M2、M3 の 3 つのステーションの代表点の組み合わせから、無限運動量飛跡からの差 $\Delta\theta$(ワイヤー)、$\Delta\phi$(ストリップ) と、M3 における座標情報を得る。Segment reconstruction から出力された $\Delta\theta$、$\Delta\phi$ は後段で運動量推定のために使用される。代表点の組み合わせが複数ある時は、コインシデンスの際のレイヤー数で優先順位がつけられる。この処理はワイヤーとストリップで独立に行い、再構成された情報を segmenet と呼ぶ。

前述の Channel Mapping や Intra station coincidence は AND や OR、NOT のみを使用した論理回路であったが、この工程では Look Up Table(LUT) を使用して直線飛跡の情報を得る。具体的には、ピボットステーションである M3 上の検出スタッガードチャンネルを起点とした無限運動量飛跡からの差に対応する $\Delta\theta$、$\Delta\phi$ の角度情報を得る (図 5.3)。$\Delta\theta$、$\Delta\phi$ はトロイド磁場中の曲率を反映し、運動量の大きさの決定に用いられる。シミュレータでは c++ の標準ライブラリに含まれるクラスの一つである map によって LUT を実装した。ファームウェア上では、LUT は UltraRam(URAM) と呼ばれる Random Access Memory を用いて実装される。本シミュレータは、ファームウェアで URAM に LUT を書き込む時に使用するものと同じファイルを入力として使用する。

ワイヤーとストリップで詳細は異なるが、unit、subunit という構造を持つことと、各ステーションの代表点の組み合わせから map の key (ファームウェアでは URAM のアドレス) を作成することは共通している。詳細は付録 E に譲り、以下ではストリップを例として大まかなロジックの流れを記述する。

5.3.1　Strip Segment Reconstruction

ストリップは各チェンバーに 32 チャンネルずつが存在しており、それらに対してコインシデンスをとることで得られる各ステーションの代表点はチェンバーにつき 63 個存在する。Channel Mapping の段階で図 5.4 のようなコインシデンスを取るべきチェンバーの組み合わせが考慮されているため、Segment Reconstruction でステーション間でのコインシデンスを取るときはチェンバーを考慮する必要はない。よって、Segment Reconstruction で M3 のチェンバー 1 枚に対するコインシデンスを考えるときは、M1、M2、M3 ステーションの各 63 代表点から、63^3

[*7] ストリップは全てのステーションがダブレットであり、チェンバー 1 つあたりのチャンネル数も等しく、対称性が良いため、全てのステーションを同一の関数で処理することができた。ワイヤーは M1 はトリプレット、M2・M3 はダブレットであり、2 種類の構造が存在している。さらに、フォワードでは一部の領域のみスタッガードチャンネルの構造が変化するため、「ワイヤーのダブレット」、「ワイヤーのエンドキャップ M1」、「ワイヤーのフォワード M1」の 3 種が必要になった。

[*8] Run3 ではワイヤーの M1 ステーションでは 2/3 以上、ストリップの M1 ステーションでは 1/2 以上、M2・M3 ステーションでは 3/4 以上がコインシデンス条件だった。

[*9] Phase2 アップグレード後のファームウェア内では、ヒット情報の信号線はコインシデンスの種類ごとに、各ステーションに存在するスタッガードチャンネルの数の長さの 1 次元配列が用意される。

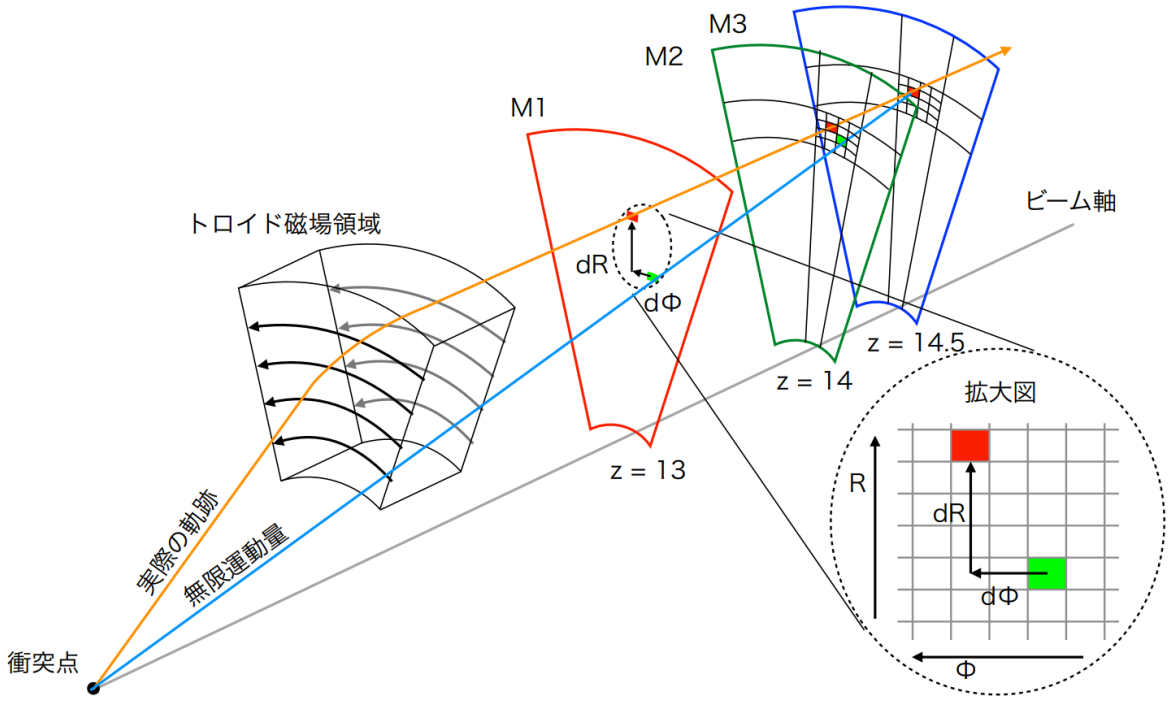

図 5.3　現行のエンドキャップ部のトリガーのコンセプト（赤塚駿一 2017）。黄色の線が実際の軌跡を、水色の線がピボットとなる M3 のヒット点を通る無限運動量飛跡を表し、M1・M2 ステーションにおけるこの 2 点の差分 $(dR, d\phi)$ から、ミューオンの p_{T} を概算する。Phase2 アップルグレード後の Segment Reconstruction における $(\Delta\theta, \Delta\phi)$ は $(dR, d\phi)$ に対応する情報である。

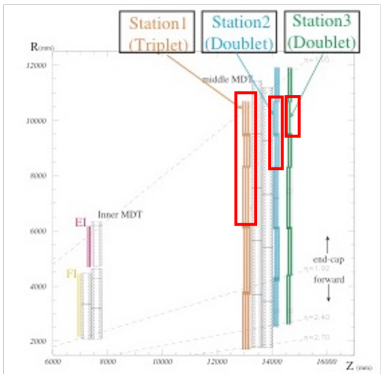

図 5.4　M3 の E2 チェンバーとコインシデンスを取るべきチェンバーの範囲（大町千尋 2006）

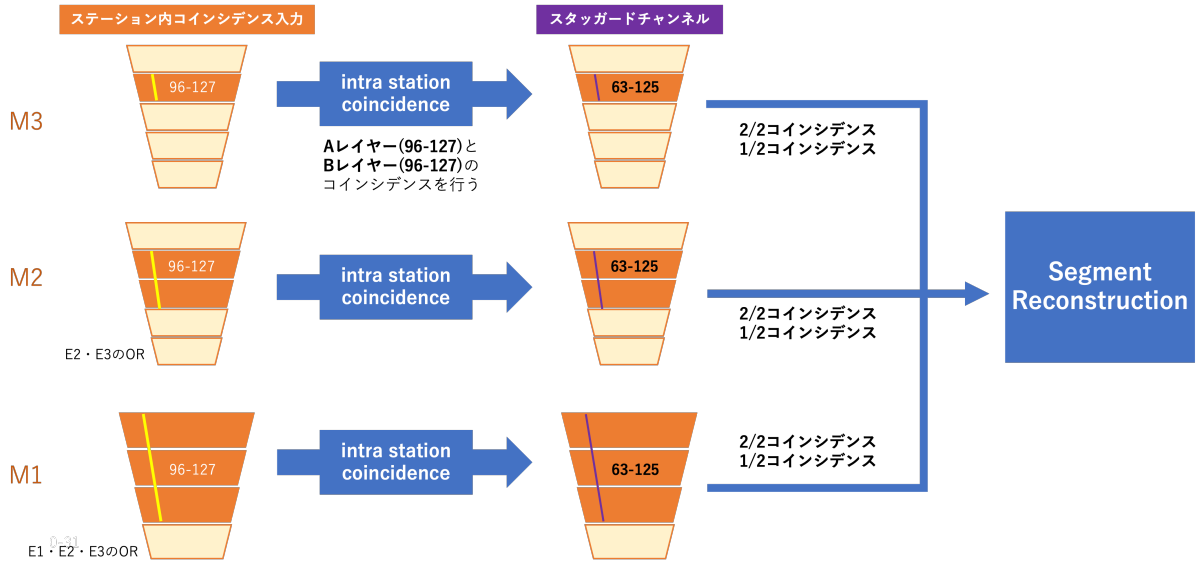

図 5.5　M3 の E2 チェンバーをピボットとする時の Segment Reconstruction の入力。ステーション内コインシデンス入力を Wired OR を考慮して定義したことにより、Segment Reconstruction では近い番号の代表点同士が入力として使えるようになっている。

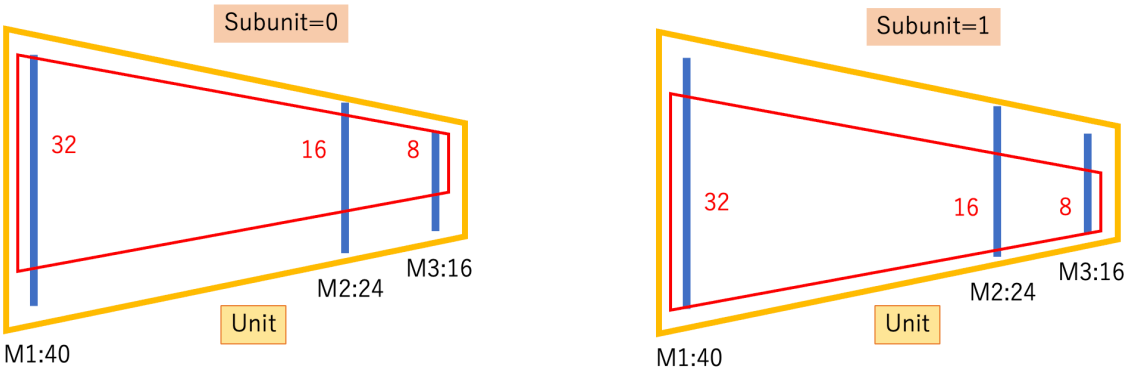

図 5.6　ストリップの Segment Reconstruction における unit/subunit の構造。黄色の枠で囲われた範囲が 1 つの unit を、赤枠で囲われた範囲が subunit を表す。左の図が Subunit=0、右の図が Subunit=1 の範囲であり、隣接する Subunit が担う範囲は全ステーションで代表点 8 つずつ並行移動したものである。

の組み合わせを考慮すればよい。例えば、図 5.5 であれば、M3 の E2 チェンバーをピボットとする飛跡のトリガーを行うためには、M1、M2、M3 ステーションの代表点 63 番から 125 番で Segment Reconstruction を行う。

さらに、63^3 の組み合わせの中で、大きく曲がる p_T の小さいトラックは物理的な興味が薄く、組み合わせとしてフェイクである可能性も高いため、そもそもトリガーにかけられないことから、LUT として用意すべき代表点の組み合わせのパターンをさらに絞り込むことができる。ここで導入するのが unit と subunit という概念である。

パターンマッチングは subunit の中で行われる。一つの subunit は、M1 の代表点 32 チャンネル (=5 bit)、M2 の代表点 16 チャンネル (=4 bit)、M3 の代表点 8 チャンネル (=3 bit) からなる[*10]。隣接する 2 つの subunit をまとめて 1 つの unit という単位として取り扱う (図 5.6)。同じ unit 内で異なる subunit が担当する M3 の代表点は被

[*10] ワイヤーでは、1 つの subunit が担当する領域は M1 の代表点 96 チャンネル (=7 bit)、M2 の代表点 32 チャンネル (=5 bit)、M3 の代表点 4 チャンネル (=2 bit) である。

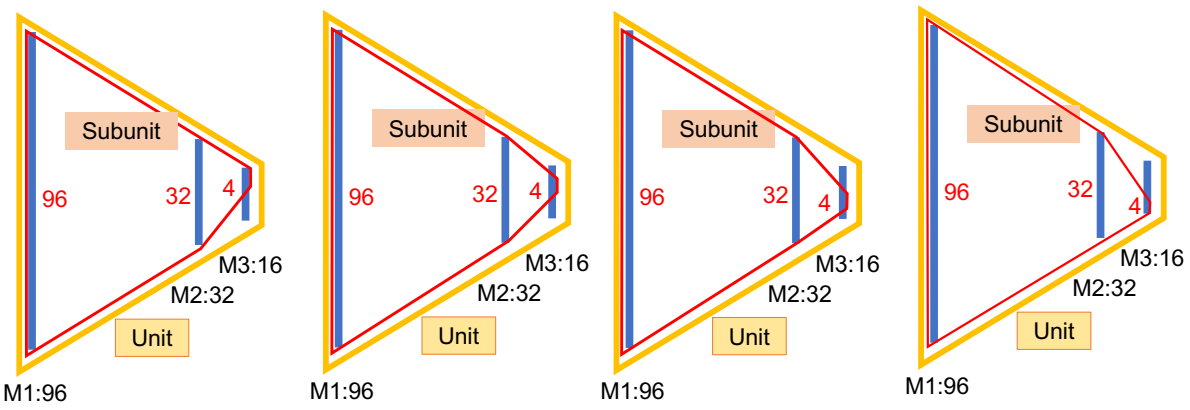

図 5.7　ワイヤーの Segment Reconstruction における unit/subunit の構造。黄色の枠で囲われた範囲が 1 つの unit を、赤枠で囲われた範囲が subunit を表す。左から順に subunit 番号 0〜3 番を表したものである。同じ unit 内の各 subunit は、M1・M2 は同一の代表点を使用し、M3 のみ排他的に代表点 4 つずつを担う。

りがないが、M1・M2 は M3 の領域に対して広がりを持っているため、同じ代表点のヒットが複数個の subunit に入力される。各チェンバーには 63 代表点が存在しているので、チェンバーごとに 8 つの subunit、4 つの unit が作られる。チェンバー端の subunit、例えば「unit 番号=0、subunit 番号=0」などは、M1 や M2 の広がり部分が実際に存在するチャンネルからはみ出すが、はみ出た部分に相当する信号線にはモジュール内で「0」が入力される (ファームウェアでは GND に接続される)。

LUT への入力は代表点の組み合わせであり、ファームウェアでは subunit 内における代表点の番号から RAM の Adress を生成する。ここで、トリガーにかけられる時間が決まっている都合上、URAM から取り出せる飛跡の数に上限がある[*11]。よって、一つの subunit の中で同じステーションの中に複数の代表点が存在しており、複数の組み合わせを作ることができる場合、より優先度の高い代表点を使った組み合わせからアドレスを生成し、LUT からデータを取り出す。ヒットがあったレイヤー数が多い代表点が優先度が高く、同じ枚数であった場合は、subunit の中心により近い代表点が優先される。

LUT から複数のデータを取り出した場合は、その中で最も優先順位の高い segment1 つのみに絞り込んで後段へ出力する。この時の数の制約は、ストリップからは 1unit あたり最大 1segment、ワイヤーからは 1subunit あたり最大 1segment である。ヒットがあったレイヤー数が多い代表点が優先度が高く、枚数が同じものが複数存在していた場合は、角度 $\Delta\theta$、$\Delta\phi$ が小さいものが優先される。

ワイヤーとストリップの両方で、代表点を表す全ての bit をアドレス生成には使用せず、bit の足を潰している。このため、URAM の一つのアドレスには複数の segment の情報が含まれ、トリガー系では URAM からデータを取り出したのちに、潰した足の情報を使って segment を選ぶ[*12] 。これは UltraRam の入力アドレス幅が 12 bit, 出力データ長が 72 bit で固定されているためである。(小林蓮 2021)

5.3.2　Wire Segment Reconstruction

大まかなロジックはストリップと同様であるが、ワイヤーは unit、subunit の構造がストリップと異なり、図 5.7 のように、一つの unit の中に 4 つの subunit が存在する。一つの subunit は、M1 の代表点 96 チャンネル (=7 bit)、M2 の代表点 32 チャンネル (=5 bit)、M3 の代表点 4 チャンネル (=2 bit) からなり、同じ unit 内の subunit は M1・

[*11] Wire/Strip で制限のかけられ方が異なる。Strip では最大 6 アドレス/subunit、Wire では最大 8 アドレス/subunit。

[*12] アドレス生成で潰される bit は、ワイヤーでは M1 の下位 2 bit、ストリップでは M1 の下位 2 bit と M2 の下位 1 bit である。このため、ワイヤーでは一つのアドレスに 4 つ、ストリップでは一つのアドレスに 8 つの segment が入っている。

M2 ステーションの代表点に関しては全く同一のものが入力となる。

フォワード領域では M1 には 312 代表点、M2 には 249 代表点、M3 には 248 代表点が存在し、これらを 16 unit で処理する。エンドキャップ領域には 37 unit が存在する。

5.4　Wire-Strip coincidence

Segment Reconstruction で得たピボットにおける検出位置情報 (η, ϕ) と、飛跡の角度情報 $(\Delta\phi, \Delta\theta)$ を用いて、検出した飛跡の運動量 $(\eta, \phi, p_{\mathrm{T}})$ を計算する。ワイヤーチャンネルとストリップチャンネルのヒット情報は、Segment Reconstruction まで独立に処理されていたが、このモジュールで統合される。

Wire-Strip coincidence には、飛跡の M3 ステーションにおける位置情報 (ϕ, θ) と飛跡の角度情報 $(\Delta\phi, \Delta\theta)$、ヒットのあったレイヤー枚数が入力される。

ファームウェアにおいては、Segment Reconstruction から入力された情報は、「pT Calculator」、「Wire Position Corrector」、「Block Selector」の 3 つのモジュールに並列的に入力・処理される。「pT Calculator」にはワイヤーとストリップのそれぞれの Segment Reconstruction から出力された角度情報 $\Delta\theta$、$\Delta\phi$ が入力され、この二つからアドレスを生成して Look Up Table から横運動量 p_{T} を得る。「Wire Position Corrector」にはワイヤーとストリップの位置情報が与えられ、ワイヤー、ストリップのチャンネル番号から η, ϕ へと情報が変換される。$(\eta, \phi) \rightarrow p_{\mathrm{T}}$ へ変換する LUT は、η, ϕ 領域ごとに独立に準備されるため、変換の η, ϕ 依存性 (磁場構造による) が吸収される設計となっている。

5.4.1　Block Selector

Wire-Strip coincidence に入力される情報は、ワイヤーでは subunit あたりに 1 つ、ストリップでは unit あたりに 1 つである。このワイヤーの subunit 1 つとストリップの unit 1 つの組み合わせを block と呼ぶ。Wire-Strip coincidence は、複数の block からなる region という単位で計算され、Block Selector では角度情報 $(\Delta\phi, \Delta\theta)$ を用いて、region 内のどの block で得られた p_{T} を出力するべきかを選択する。region には、ワイヤーの subunit 2 つとストリップの unit 4 つの掛け合わせからなる、8 個の block を内包するものと、ワイヤーの subunit 8 つとストリップの unit 4 つの掛け合わせからなるものと、32 個の block を内包するもののの 2 種類がある。ここではこれらをそれぞれ「Region8」と「Region32」と呼称する。エンドキャップは 22 個の Region8 と 13 個の Region32 からなり、フォワードは 8 個の Region32 からなる (図 5.8)。

「Region8」と「Region32」では出力される block の数が異なる。「Region8」では 1 つの block からの情報のみを選択して出力する。これに対し、「Region32」では最大 4 つの block の計算結果を出力する。

5.4.2　pT Calculator

飛跡の角度情報 $(\Delta\phi, \Delta\theta)$ を用いた Coincidence Window(CW) で飛跡の p_{T} を出力する (図 5.9)。CW はワイヤー方向には subunit ごと、ストリップ方向には unit ごとの単位で用意する。

RAM のアドレスとなるのは { ワイヤーの $\Delta\theta$（7 bit)、ストリップの $\Delta\phi$ の領域（4 bit）} の合計 11 bit である[*13] 。また、Segment Reconstruction から入力される $\Delta\theta$ と $\Delta\phi$ の bit 幅はそれぞれ 8 bit、9 bit であるが、ウィ

[*13] ファームウェアで RAM に CW を書き込むために使用するテキストファイルでは、アドレスの部分の上に Region 内の URAM 番号 0-3(2 bit) がついている。ビットワイズシミュレータへの CW の入力にも同じテキストファイルを使用しており、シミュレータ内では wire の subunit ごとに配列化した map を CW として使用している。このため、シミュレータにおける key は Region 内の URAM 番号 0-3(2 bit) も加えた {Regeon 内の URAM 番号 0-3（2 bit)、ワイヤーの $\Delta\theta$（7 bit)、ストリップの $\Delta\phi$ の領域（4 bit）} の計 13 bit である。

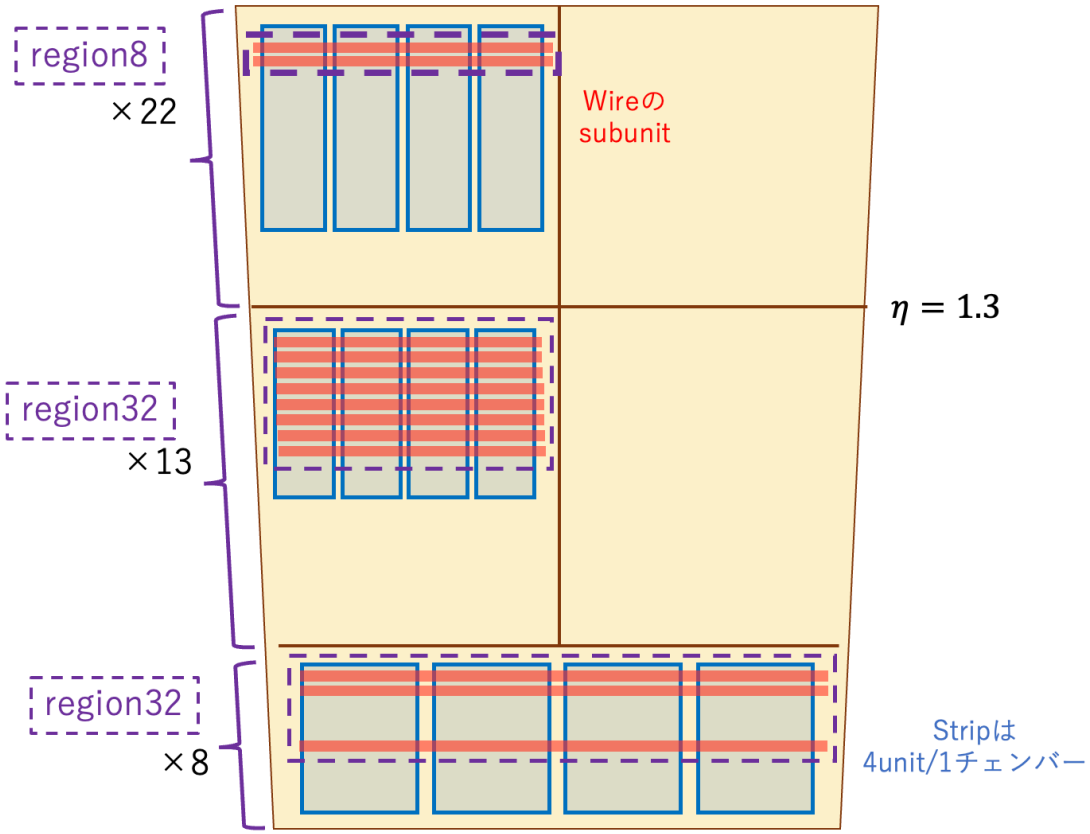

図 5.8　Wire-Strip Coincidence における region8 と region32 の配置。チェンバーにつき横に 4 つ並ぶ青い四角がストリップの unit を表し、赤い横線がワイヤーの subunit の単位を表す。紫色の波線で囲まれた領域が 1 つの region を表し、図の左にあるように、エンドキャップ領域には円板の外側に region8 が 22 個、ビーム軸側に region32 が 13 個存在している。フォワード領域は 32 個の region32 が存在する。

ンドウの数を抑えるために、アドレスを作る時は bit 幅を減少させる。$\Delta\theta$ は単純に最下位の 1 bit を無視した 7 bit をアドレスとして使用する。対して、ストリップの $\Delta\phi$(最上位 1bit が符号、下位 8bit が絶対値) は 0 付近では解像度を高くするために、表 5.1 の法則に従い、9 bit の情報を 4 bit に変換する (表は符号に相当する最上位 1 bit を除く、絶対値部分 8 bit から 3 bit への変換)。

CW からは横運動量 p_{T} が 4 bit で出力される。現在の LUT における出力は 0 ～ 4 の 5 種類であり、0 以外の出力は数の小さい方から 5、10、15、20 GeV の閾値を表すものとなっている。将来的には、Window を 15 段階に増設し、CW の作成手法も現在のものから変わる予定である[*14] 。

5.4.3　Wire Position Corrector

このモジュールはワイヤーの位置情報を補正するためのものである。

ワイヤーの飛跡検出で得る飛跡の動径 (R/η) 方向の位置情報はワイヤーチャンネルのコインシデンスによって得られる ID 番号であるが、この ID 番号は η を正確に示すものではない。この原因は 2 つある。一つ目の要因は、ワイヤーのチャンネルが η に対して完全に線形に配置されたものではないためである (図 5.10)。二つ目の理由は、ワイヤーチャンネルはチェンバー内で直線に張られているため、ϕ 座標により、出力される ID 情報が示す η と実際

[*14] 本論文執筆時点においては、増設後の閾値の設定と CW の作成手法は(吉村宣倖 2022)の 4 章と同様のものとなる見込みである。

表 5.1　pt calculator における、入力される $\Delta\phi$ の絶対値 8 bit とアドレスに使われる絶対値部分 3bit の対応関係

入力される $\Delta\phi$ の絶対値 (8 bit)	アドレスに使われる 0~5(3 bit)
0-3	入力と同じ 0-3
4-6	4
7-9	5
10-12	6
13-	7

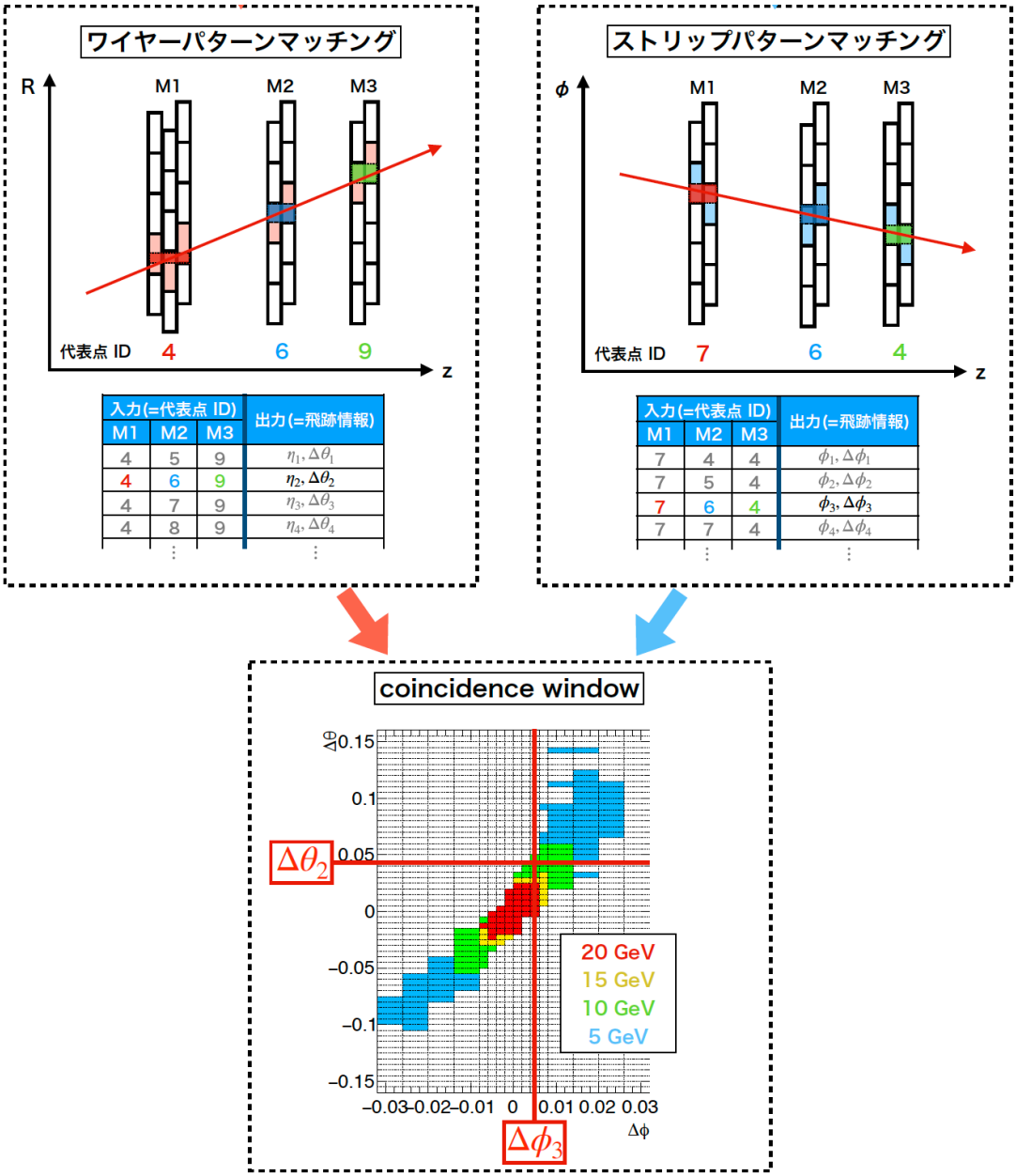

図 5.9　Wire-Strip coincidence のパターンマッチングの概念図(三野裕哉 2020)。ワイヤーとストリップで独立にパターンマッチングを行い、飛跡の位置情報と角度情報 $(\Delta\theta, \Delta\phi)$ を抽出する．ワイヤーとストリップで求めた飛跡の角度情報から、CW を用いて p_{T} 閾値を求める。

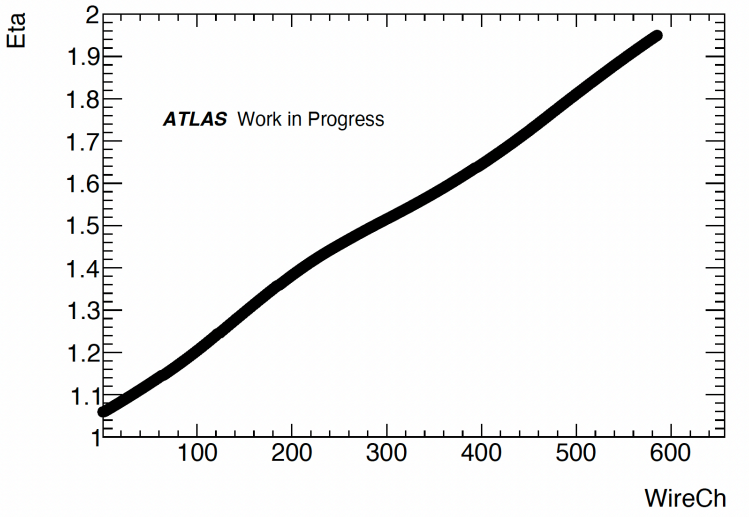

図 5.10　Wire チャンネルと η 座標の非線形性(河本地弘他)

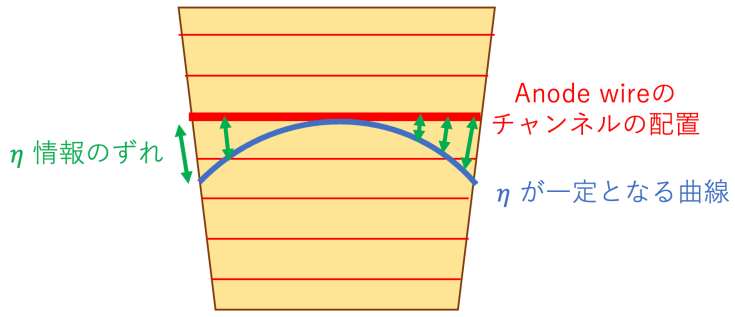

図 5.11　Wire チャンネルの組み合わせで定義される η-ID と実際の η のずれ

の η の間でずれが生じてしまうためである (図 5.11)。

このモジュールではワイヤーとストリップの代表点 ID(スタッガードチャンネルの分解能) を入力として LUT を用いてより高精度な η 位置情報を得る。入力となるアドレスは {Wire ID(2bit), Strip ID(6bit)} の計 8bit であり、8bit のデータが出力される。LUT はワイヤーの subunit ごとに定義されるため、ワイヤーの ID は subunit 内にある M3 スタッガードチャンネル 4 つの中で一意に決められれば良いので 2bit が入力される。ストリップの 6bit はチェンバー内での 0~62 のスタッガードチャンネルを一意に示す。

5.4.4　ビットワイズシミュレータとファームウェアにおける処理の違い

上記の 3 つのモジュールはファームウェア上で同時に走るので、全ての block に対する pT Calculator と Wire Position Corrector の出力を一度得てから、Block Selector によって出力するべきものを選別する。一方、ビットワイズシミュレータでは各モジュールは並列化されておらず、逐次的に走るため、最初に Block Selector で block の選択を行なってから、その block のみ pT Calculator と Wire Position Corrector の計算を行う。

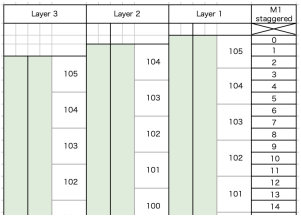

図 5.12　修正前の M1 ステーション (フォワード領域) の代表点の定義。代表点の数は 316 となる。

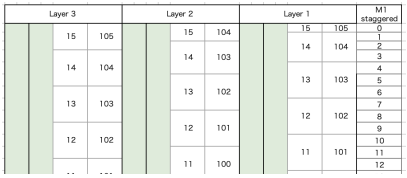

図 5.13　修正後の M1 ステーション (フォワード領域) の代表点の定義。代表点の数は 312 となる。

5.5　intra station coincidence 部分のファームウェアへの提言

以上のシミュレータを作成するにあたり、ファームウェアのロジックを深く理解する必要があったが、その過程でファームウェアの改善点を発見したため、ファームウェア側へのフィードバックを行った。

ワイヤーとストリップのファームウェアは違う研究者によって作成されたため、ワイヤー部分とストリップ部分のモジュールは独立して存在しており、提言した改善もそれぞれ異なる。

5.5.1　Wire

ワイヤーに関しては、代表点の数についての提言を 2 つ行った。この提言はチェンバー境界の処理に起因するものである。

1 つ目は、エンドキャップ・フォワードの端のチャンネルの処理が検出器内でのチャンネル配置と不整合であったことへの指摘である。修正前のファームウェアでは、図 5.12 に示すように、レイヤーによってチェンバーの端におけるチャンネルの端の位置が異なるものとして取り扱われていた。しかし、実際のチェンバーの構造は図 5.13 に示すように、チャンネルの終端は同じ η であり、チャンネル幅が異なっているため、チェンバー構造に則したファームウェアに修正が必要であった。

2 つ目の提案は、エンドキャップではチェンバーごとに代表点を分割していたところを、連続的に処理することである。前段のモジュールである Channel Mapping で、チェンバーのオーバーラップで同じ η を観測しているチャンネルに対して OR をとる Wired OR の処理を行っているため、intra station coincidence への入力はチェンバー境界の位置を考慮する必要がなく、レイヤー内のチャンネルを一続きのものとして取り扱うことができる。しかし、ファームウェア内ではチェンバー境界で例外処理を行なっていたため、その必要はないと指摘した。

5.5.2　Strip

ストリップに関しては、2/2 コインシデンスと 1/2 コインシデンスの処理についての指摘を行った。

2/2 コインシデンスと 1/2 コインシデンスの出力は排他的なものである (デクラスタリング)。修正前のロジックでは、2 段階で処理を行なっており、まず各代表点に対して代表点を直接構成するチャンネルの情報のみを使って 2/2 コインシデンスと 1/2 以上のコインシデンスを一時的に取ってから、隣接する代表点で 2/2 コインシデンスが成立している 1/2 コインシデンスを除外していた。これに対して、Run3 で使用されているエレクトロニクスの論理回路を模倣し、代表点を直接構成するチャンネルの情報のみではなくその隣のチャンネルまで使用することで 1 段階で 2/2 コインシデンスと 1/2 コインシデンスを独立かつ排他的に行える (付録 D) と提案した。

第 6 章

検証のためのテスト入力パターンの生成機構の開発

4 章で述べたリレーショナル・データベースを用いてセクターロジックへ入力するためのテストパターンを自動生成する仕組みを作成した。この仕組みの主体は Python である。MySQL と Python の API を用いて、チャンネルの情報と SL に入力される bitmap 内でのポジションの対応関係を取り出し、整形して coe ファイルやテキストファイルとして出力した。

以下に述べるテストベクター生成用のプログラムは ATLAS のインターナルページで公開している(山下 2022)。

6.1　実機試験とシミュレーションの全体設計

この研究で開発したテストパターンは実機、ファームウェアシミュレータ、5 章で開発したシミュレータの全てに入力するものである。このように同一のテストパターンを実機およびシミュレータという複数の対象に使用することによる、コヒーレントな検証が可能となっている。この節ではファームウェアおよびシミュレータのテストパターンの入力部分について記述する。

6.1.1　ファームウェア上のトリガー試験用の設計

ファームウェアには、試験のためにテストパターンをテストパルスに同期して、トリガー系に入力する仕掛けがある。図 6.1 に示すように、前段回路からヒットパルスパターンを入力する代わりに、ファームウェア内の RAM の中からテストパターンを読み出し、トリガー系に入力する。RAM にテストパターンを書き込む仕掛けとして、コンパイル時に初期値として仕掛ける方法及び、RAM の情報を上書きすることの 2 種類がある。前者の、初期値として設定する場合は coe ファイルと呼ばれるファイル形式を用い、後者の RAM の上書きでは、SLR ID、RAM ID、address、データを指定されたフォーマットに沿って記述したテキストファイルを使用する。このように RAM へ仕込むテストパターンを変更することにより、任意のパターンによる試験を行うことができる。これは実機およびファームウェアシミュレーションの両方で同じ仕掛けとなっている。

この研究では、6.2.3 節の方法論により指定される任意のヒットの組み合わせの入力をエミュレートするための入力パターンを生成し、初期値の設定に使用する coe ファイルと、上書きに使用するテキストファイルの両方の形式でテストパターンを生成する仕組みを開発した。テストパターンを入力する RAM の深さは 64 であり、Address=0 にはデータを入れることができないので、一度に書き込めるヒットパターンの数は 63 に制限されている。

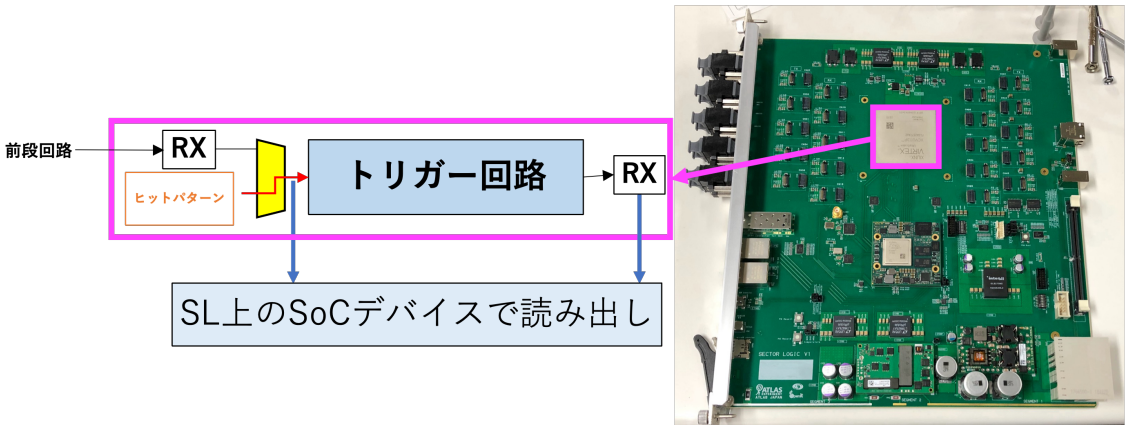

図 6.1　セクターロジック実機試験(三島章熙他)のセットアップ。前段階ろからヒットパルスパターンを入力する代わりに、URAM に予めテストパターンを書き込んでおき、テストパルスに同期して、URAM からトリガー系へテストパターンを入力する。トリガー系の入出力はセクターロジック上の SoC デバイスで読み出し、ファームウェアシミュレータやビットワイズシミュレータの出力との比較を行う。

6.1.2　シミュレータにおける入力

5 章で開発したビットワイズシミュレータは、ファイルの読み込みを行う C++ の関数を作成することで、任意の入力ファイルに対応することができる。この研究では、ファームウェアとコヒーレントな研究をするために、上述の RAM の初期設定を行うのに使用していた coe ファイルを入力する機構を用意した。

6.1.3　テストベクター生成機構の全体設計

上述の coe ファイル形式、テキストファイル形式での入力データを出力するための機構を開発した。同一のヒットの組み合わせに対し、統一的な枠組みで準備できるため、ハードウェア、ソフトウェアの横断的かつ系統的な検証がこれにより実現できる。

1 枚のセクターロジックへ入力されるヒットデータは、29 枚の PS ボードにつき 2 本ずつ用意したファイバーによって、128 ビットのビット列の形式をしている。本質的に必要な機能は、トリガー論理に入力したいヒットの組み合わせをこの 128 bit×58 ファイバーの形式に整形することであり、この目的のためには Channel Mapping モジュールを生成したときと同様に、リレーショナル・データベースと接続した機構を作成するのが最も合理的な解になる。以上の考察により、テストベクター生成機構の全体設計を行い、リレーショナル・データベースと Python の API を利用して開発した。

6.2　テストベクター生成機構への入力フォーマットと 3 種の方法論

この研究で開発したテストベクター生成機構は種ファイル (json ファイル) とリレーショナル・データベースを入力とするものである。json ファイルに作成したいヒットパターンの情報を書き込み、リレーショナル・データベースのケーブリング情報を用いて、チャンネル情報をセクターロジックの入力となる 128 bit×58 ファイバーのテストベクターの形式に翻訳する。この json ファイルのフォーマットを複数用意した。ここでは主に使用している 2 種

類のフォーマットについて述べる[*1]。入力する json ファイルには「イベント」の概念が存在しており、1 つのイベントに対し任意の数のヒットを仕込むことが可能であり、任意の数のイベントに対して独立に指定できる構造を持つ。また、6.2.1 節、6.2.2 節で示す 2 つのフォーマットの自由な組み合わせで 1 つのイベントのヒットの組み合わせを指定できる。この構造により、複数の飛跡が存在するイベントや、高いパイルアップ環境下のノイズヒットの混入等を柔軟に表現することが可能である。

6.2.1　個々のチャンネルを指定するフォーマット

1 つ目のフォーマットはチャンネルを一つずつ指定するものである。このフォーマットでは、以下の情報を指定する。

- エンドキャップ/ フォワード領域
- (エンドキャップであれば)ϕ 0/1
- レイヤー名: ワイヤーでは「1」～「7」、ストリップでは「M1-A」～「M3-B」
- レイヤー内でのチャンネルの通し番号

この情報の組み合わせによって、1 枚のセクターロジックが担う TGC 検出器の領域の中で一意にチャンネルを定めることができる。任意のヒットの組み合わせの入力を指定することができるため、6.2.2 節で導入する無限運動量飛跡と組み合わせることにより、飛跡の付近にノイズが混入した場合の回路の応答等の調査にも活用できる。

6.2.2　無限運動量飛跡を生成するフォーマット

2 つ目のフォーマットは無限運動量飛跡を生成するものである。4.2.1 節で述べたデータベース内のスタッガードチャンネルは同じ η 座標または ϕ 座標のチャンネルに対して同じ番号が振られているため、スタッガードチャンネル番号を使うことによって、簡単に無限運動量飛跡を作成することができる。

このフォーマットでは、サブセクターの指定と、ワイヤーのスタッガードチャンネル番号とストリップのスタッガードチャンネル番号を入力とする。このフォーマットで 1 つの無限運動量飛跡の座標となるスタッガードチャンネル番号を指定すると、ワイヤー・ストリップ計 13 層のヒットの組み合わせが自動的に定まる[*2]。無限運動量飛跡は確実にトリガーしなければならない最も簡素なパターンに対応し、このフォーマットで指定される入力パターンは、運転中、試運転中のファームウェアの最初の検証や、システムの診断に最適なパターンとなる。

6.2.3　テストパターン生成機構利用における 3 種の方法論

手入力によるチャンネル指定

一つ目の方法論は、入力ヒットチャンネルを手で指定して準備するものである。上述した入力フォーマット一つは、json ファイル及びテストベクター生成機構の主体である Python スクリプト内では一つの辞書として取り扱う。図 6.2 のように、json ファイルは全体としてリストの形をしており、リスト内にさらにリスト化された辞書が入る入れ子構造となっている。大枠のリストの何番目であるかが、何番目のイベントであるかに対応しており、一つのリストに入れられる辞書の数には制限がないため、1 イベントの中に任意の数のヒットを入れることができる。

[*1] この研究では使用していないが、セクターロジックの入力となる 128 bit×58 ファイバーのファイバーの番号と bit position を直接使用するフォーマットや、ランダムに任意の数のノイズを生成するためのフォーマットも用意した。

[*2] 基本的には 13 層に 1 点ずつの 13 点のヒットになるが、ヒット数の増減する例外もある。もし入力したスタッガードチャンネル番号がチェンバー境界であった場合、Wired OR を取るワイヤーチャンネル両方が入力となり、ヒット数は 13 よりも多くなる。また、M1～M3 まで全てのステーションで覆えていない η 領域にあたるワイヤーのスタッガードチャンネルを入力とした場合は、ヒット数は 13 よりも少なくなる。

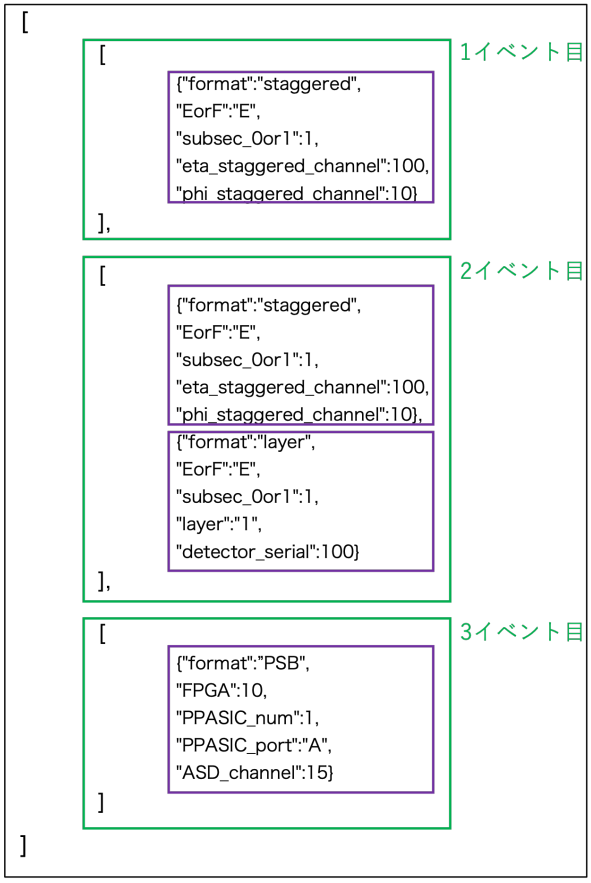

図 6.2　テストパターン生成プログラムへ入力する json ファイルの例。第 1 イベントには無限運動量飛跡が 1 トラック入っており、第 2 イベントには、第 1 イベントと同一の無限運動量飛跡に加えて、単一のヒットが 1 つ入っている。第 3 イベントには単一のヒットが第 2 イベントと異なるフォーマットで記述されている。

よって、このフォーマットによるチャンネルの指定を、1 つのイベントの中に任意の数記述することにより、任意のヒットパターンを作成することができる。

無限運動量飛跡

二つ目の方法論は、上述した二つ目の入力フォーマットを用いて、無限運動量飛跡に対応するヒットの組み合わせを生成する。これにより、最もよく制御された入力パターンが生成できる。

本研究では無限運動量飛跡のテストパターンをトリガーロジックの最初の検証としてこれは実機、Vivado Simulator、ビットワイズシミュレータの 3 つ全てに入力した。最も優先度の高いヒットのパターンが確実に再構成されるかを確認すると同時に、この 3 種のトリガー系それぞれから得られた出力を比較することによって、シミュレータ及びファームウェアのデバッグを行った。ワイヤーの 7 レイヤーとストリップの 6 レイヤー全てにヒットがある無限運動量飛跡は、トリガーするべきトラックの中で最も単純かつ優先度の高いものであり、ロジックの検証も比較的容易なので、トリガーロジックの最初の検証として重要かつ最適である。

MC または実データ

三つ目の方法論は、MC または実データの情報を保持する ROOT ファイルを利用するものである。このテストパターンファイルは、一つ目の方法論のように、ROOT ファイルから 6.2.1 節のフォーマットで記述した json ファイ

ルを作成し、これをテストパターン生成機構に入力することによって、機械的にヒットパターンファイルを生成することができる。

この方法ではトロイド磁場によって現実的な有限の曲率を持った飛跡に対する入力パターンが準備できるため、様々な運動量に対する応答を確認するなどの、より詳細なトリガーの検証が可能となる。実際の衝突実験では、横運動量 p_{T} が大きくないため磁場の影響で曲がる事象や、確率的な問題で (またはチェンバーの不具合によって)13 層全てでヒット情報が得られない事象などがあり、それらのトリガーも行わなければならない。特に、磁場による飛跡の歪曲でミューオンの横運動量を測る検出器であるため、直線飛跡から少しだけ曲がった飛跡もトリガーし、どの程度曲がったかまでを含めて正しく処理できることがトリガーシステムとして重要である。MC シミュレーションや実データから抽出した飛跡に対するヒットパターンを使用することで、本番の衝突実験での観測が期待されるこのような様々な事象に対する応答を満遍なく検証することができ、より精度の良いトリガーの性能試験が行える。

6.2.4　現在のテストベンチにおけるテストベクターの活用

この研究で開発したテストパターンファイルは、現在の研究ですでに活用している。それぞれの結果については 7 章で述べる。また、このフレームワークは今後のトリガーファームウェア開発について活用可能であるほか、高輝度 LHC 実験開始後のトリガーシステムの診断機構としてや、機能実装前の試験などにも使用することができる。

第 7 章

検証の研究と議論

5 章で開発したビットワイズシミュレータと、6 章で生成したテストパターンを使用して、セクターロジックトリガー系の性能検証を行った。

セクターロジックの検証システムを確立できたことが本研究の成果であり、この章ではこの検証システムを使用した 3 種の具体的な運用方法とその結果を記述する。

7.1　シミュレータの開発状況と実機の統合試験の状況

この検証を行った段階でのビットワイズシミュレータの作成状況を図 7.1 に示す。intra station coincidence[*1] まではエンドキャップ、フォワードの両方の領域が完成している。Segment Reconstruction[*2]及び Wire-Strip Coincidence はフォワード領域のみ作成した。

ファームウェア側もトリガー系のモジュールの統合が進んでいる最中であり、フォワード領域は Wire-Strip Coincidence までの統合が終わっている。

7.2　無限運動量飛跡 63 イベントを使用した出力照合

無限運動量飛跡 63 イベントをセクターロジック実機、ファームウェアシミュレータ (vivado)、ビットワイズシミュレータに入力し、各出力の照合を行なった。この試験では、全レイヤーにヒットがある直線飛跡という最も単

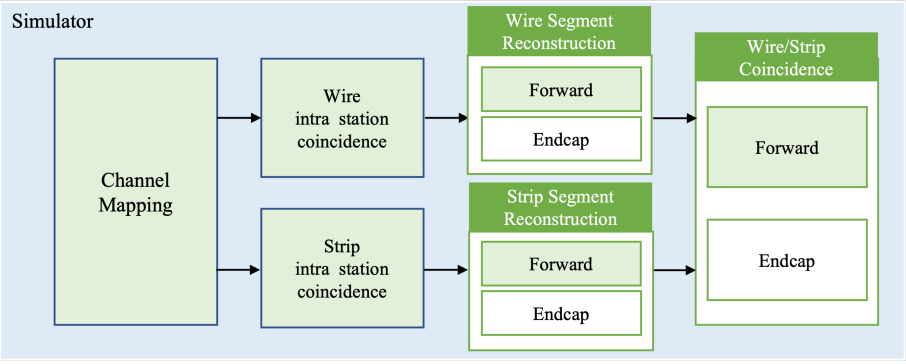

図 7.1　この検証を行った段階でのビットワイズシミュレータの作成状況。緑色に塗った箱はシミュレータ内に作成済のモジュールである

[*1] ステーション内での代表点を決めるモジュールに対応するクラス

[*2] ワイヤー、ストリップそれぞれで飛跡が無限運動量飛跡からどれほど曲がったかを LUT を用いて得る

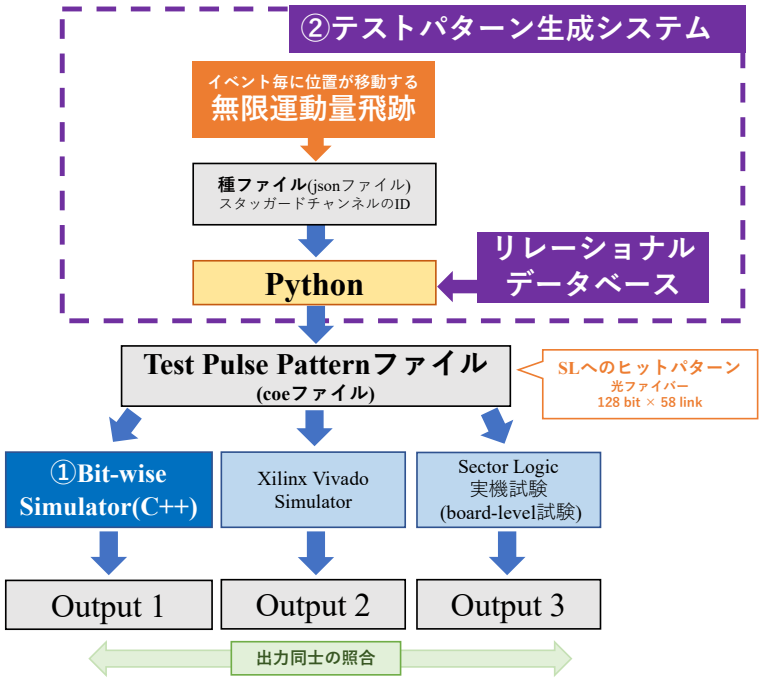

図 7.2　無限運動量飛跡 63 イベントを使用した出力照合の流れ: ヒットパターンの生成・処理・SL トリガー出力の経路。

純かつ重要なテストパターンを用いてトリガー系の応答を簡単に確認すると共に、実機とシミュレータの整合性も確認した。

図 7.2 に無限運動量飛跡を使用した検証試験の流れを示す。まず、Python ファイルによって無限運動量飛跡のフォーマット (6.2.2 節) で記述した json ファイルを作成した。この json ファイルをリレーショナル・データベースと API で接続されたテストパターン生成機構に読み込み、テストパターンファイルとして coe ファイル[*3]を出力した。この coe ファイルを実機、vivado シミュレータ (ファームウェアシミュレータ)、ビットワイズシミュレータに入力し、それぞれの出力を比較することでトリガー系の検証を行った。

入力したヒットパターン

実機試験でトリガー系にヒットパターンを入力するために用いる RAM の深さは 63 であるため、この試験で使用できるイベント数は 63 イベントに制限される。63 イベントでセクター全体を満遍に検証するため、以下の条件でヒットパターンを作成した。これを図に書き表したのが図 7.3 である。図のように、ヒットはそれぞれのサブセクター内でチェンバーを斜めに横断する。生成した 63 イベント分のテストパターンは以下の条件を満たす。

- Endcap-Phi0、 Endcap-Phi1、 Forward の 3 つのサブセクターにつき、イベント毎に 1 本ずつ、合計 3 本の無限運動量飛跡
- 1 本の無限運動量飛跡はワイヤー 7 層 + ストリップ 6 層の、合計 13 ヒットで構成した
- イベントごとにチャンネル番号が一定値ずつずれていく

[*3] ファームウェアの試験において、RAM に初期値としてテストパターンを書き込む時に使用するファイル

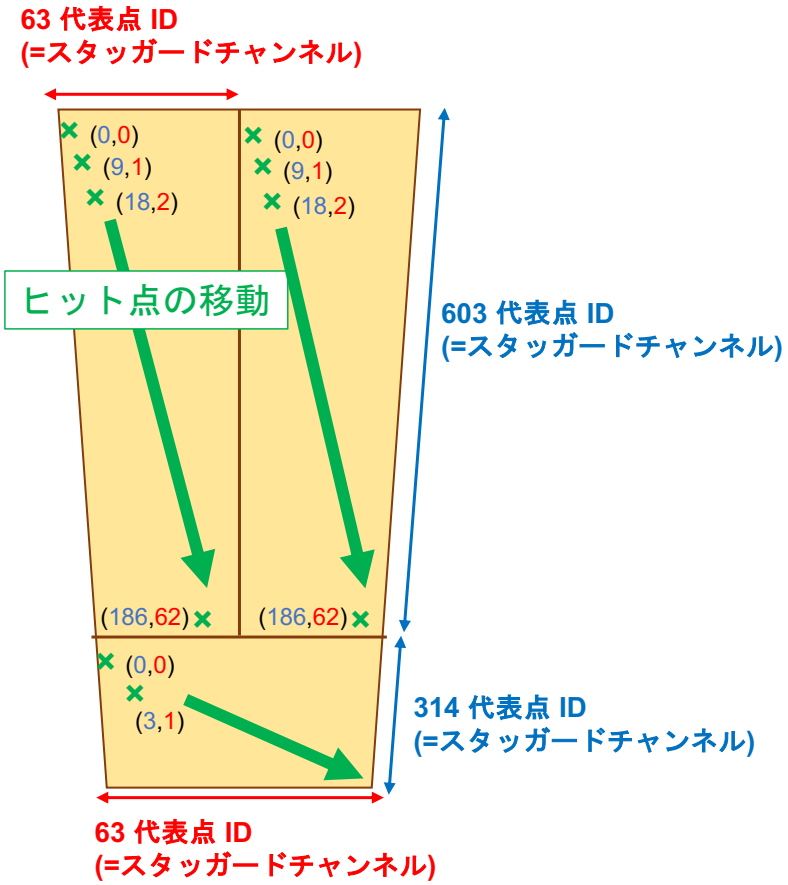

図 7.3　無限運動量飛跡 63 パターンの試験。1 イベントにつき、各サブセクターに無限運動量飛跡が一つずつ入力した。イベントごとに、入力されるヒットパターンの η ID、ϕ ID に相当するスタッガードチャンネルの値が一定値ずつずれ、イベント 1 番からイベント 63 番にかけて、サブセクターを斜めに横断する。

7.2.1　各出力の照合による検証

実機の統合試験では、ボード単体でのトリガー回路の試験フレームワークが実現され、動作検証が行われており（三島章熙他）、ファームウェアシミュレータと実機試験の出力照合は、統合研究側で一致が確認されている。この研究ではビットワイズシミュレータの出力とファームウェアシミュレータの出力の比較について論じる。

vivado シミュレータで波形コンフィギュレーション (WCFG) ファイルを開いたものを図 7.4 に示す。vivado シミュレータはファームウェアの挙動を時間に沿って忠実に再現するものであり、図の横方向は時間の進行に対応している[*4]。

ファームウェアは VHDL や Verilog HDL、System Verilog などの HDL で記述されているが、この研究では System Verilog のテキスト出力用の機能を使用して vivado シミュレータの出力ファイルを作成した。

ファームウェア側では Segment Reconstruction までの vivado シミュレーションの出力が得られているので、Segment Reconstruction までの出力の比較を行った。

[*4] この画面ではテストパルスパターンの入力とそれによる応答を視覚的に見ることができる他、あるタイミングにおける任意の信号線 (ベクター) の出力をクリップボードにコピーすることもできるため、研究の初期段階では、この画面から vivado シミュレータの出力ファイルを作成していた。本研究は共同研究として行なっており、特に intra station coincidence までの箇所は共同研究者の三島さんに出力ファイルを作成してもらっていた。

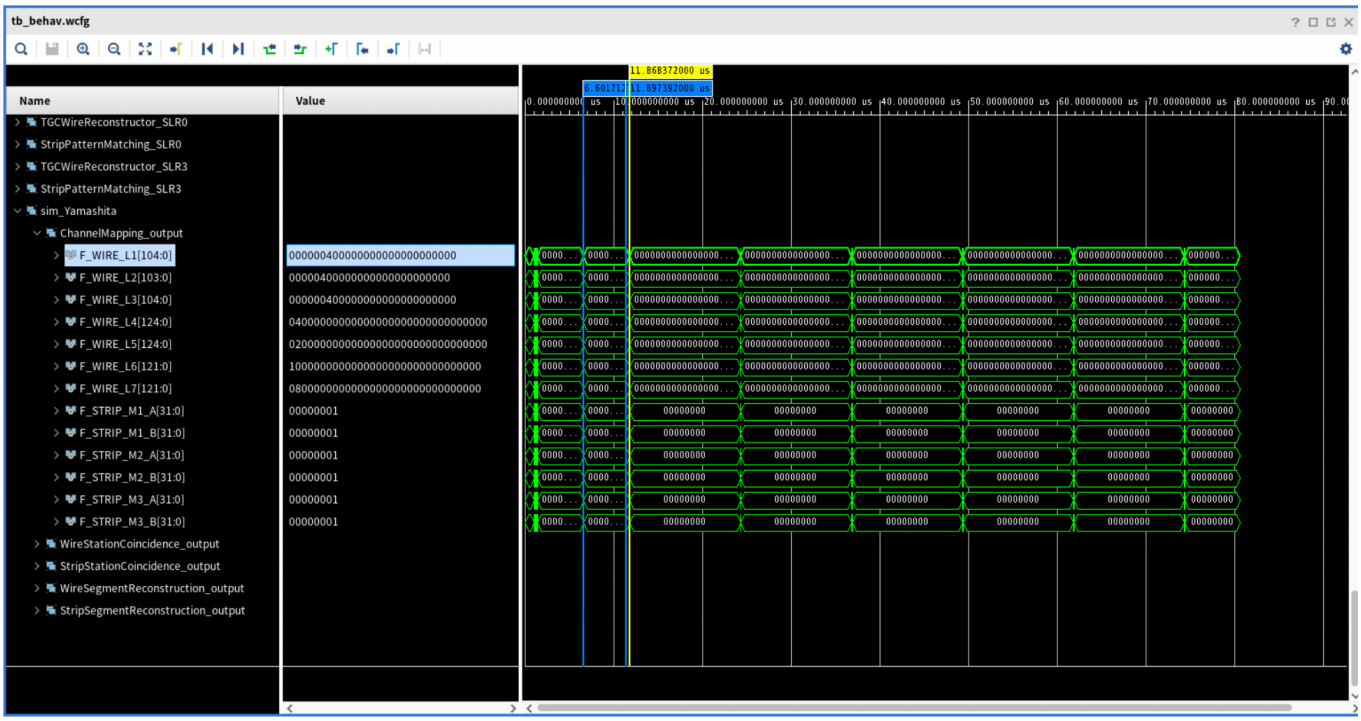

図 7.4　vivado シミュレーション

intra station coincidence まで

フォワード、エンドキャップを含む全領域で、セクターロジック実機、ファームウェアシミュレータ (vivado シミュレータ)、ビットワイズシミュレータの全てで全く同じ出力が得られた。

Wire Segment Reconstruction (フォワード領域)

セクターロジック実機、ファームウェアシミュレータ (vivado シミュレータ)、ビットワイズシミュレータの全てで全く同じ出力が得られた。

Strip Segment Reconstruction (フォワード領域)

セクターロジック実機とファームウェアシミュレータ (vivado シミュレータ) は同じ出力となった。しかし、ビットワイズシミュレータとの比較では、入力した 63 パターンのうち、1 パターンのみ出力が一致しなかった。入力パターンは、フォワード領域に存在する 63 個の代表点を網羅しているが、一致しなかったパターンはチェンバーの最端の代表点による無限運動量飛跡であった。このテストパターンでは、ファームウェアではイベントが再構成されず (LUT からの出力がないことを、ここでは「再構成されない」と表現する) 、シミュレータではイベントが再構成された。無限運動量飛跡はトリガーしなければならないパターンであるため、現在、ファームウェアで再構成が成立しない理由を調査中である。

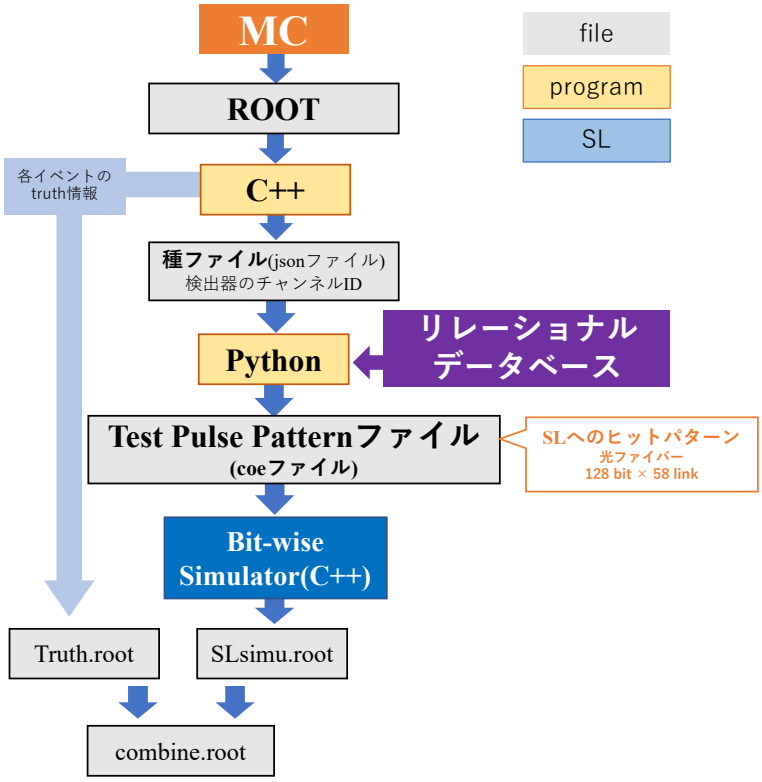

図 7.5　MC データを使用したトリガー検証の流れ。MC データの入った ROOT ファイルから、試験用のイベントを選別し、ヒットパターンを表す json ファイルと、各イベントの Truth 情報をまとめた ROOT ファイルを作成した。テストパターン生成機構によって json ファイルから 128 bit×58 links 形式のテストベクターを生成し、ビットワイズシミュレータで試験を行なった。最後にシミュレーションの結果と Truth 情報を統合して、各モジュールの efficiency を解析した。

7.3　無限運動量飛跡のテストパターンによる Segment Reconstruction の LUT の検証

Segment Reconstruction の LUT を検証するために、Foward 領域のワイヤーチャンネルについて全代表点に対する無限運動量飛跡を生成し、ビットワイズシミュレータへ入力した。この検証ではファームウェアへテストパターンは入力せず、ビットワイズシミュレータ単体で試験を行った。

この試験では 1 イベントにつき 1 本の無限運動量飛跡を入力した。M1 から M3 まで揃っているスタッガードチャンネルは 242 個であるため、イベント数は 242 である。この試験の目的は LUT のデバッグであったため、時間あたりのイベント処理数に優れたビットワイズシミュレータを使用した。

この試験の結果、スタッガードチャンネル 248 番、249 番による 2 本の無限運動量飛跡が再構成できなかった。無限運動量飛跡はトリガーしなければならないパターンであるため、LUT 製作者にフィードバックを行い、LUT が修正された。

7.4　MC データを使用した efficiency の検証

MC シミュレーションデータの ROOT ファイルを使用して、テストパターンを作成し、ビットワイズシミュレータによる LUT の試験を行った。この試験は一つ目の検証でビットワイズシミュレータが実機と同一のロジックになっていることを確認した後に行なった。

図 7.5 にこの試験の設計を示す。MC シミュレーションデータには粒子の情報 (Truth 情報) とヒットのあった TGC チャンネルを示す情報が入っている。MC データの入った ROOT ファイル内ではチャンネルを示す形式がリレーショナル・データベースでの表現方法と異なっていたため、適切に解釈した上で、ROOT ファイルの情報から json ファイルを生成するプログラム (c++) を作成した。ここで、ヒットのあるチャンネルを記述する json ファイルと同時に、テストパターンに対応する粒子の Truth 情報を別ファイル (Truth.root) として作成した。json ファイルをもとに生成したテストパターンファイル (coe ファイル) をビットワイズシミュレータに入力し、シミュレータで再構成された飛跡の角度情報 $\Delta\theta$、$\Delta\phi$、横運動量 p_{T} の情報を ROOT ファイル形式 (SLsimu.root) で得た。最後に、シミュレータの出力 (SLsimu.root) と Truth 情報 (Truth.root) を統合したファイル (combine.root) を作成し、このファイルを使用してトリガー系の efficiency を得た。この検証試験では解析に Truth 情報を使用することにより、ミューオンの再構成率の p_{T} 依存性など、様々な条件でトリガーロジックの性能を検証することが可能である。

7.4.1　ヒットデータの選別

ROOT ファイルに入っているイベントのうち、フォワード領域で確実にトリガーされるイベントのみについて検証を行うために、解析には以下の条件を満たす 6152 イベントを使用した。

- ミューオンが一つだけのシングルミューオンイベント
- Truth の η が $1.95 < |\eta| < 2.4$ (フォワード領域において、M1 から M3 ステーションまでのチェンバーが全て揃っている領域)
- Truth の ϕ が $2.87 < |\phi| < 3.14$ (アライメントホール[*5]が存在していない領域)

7.4.2　各モジュールの efficiency

Strip Segment Reconstruction (フォワード領域)

図 7.6 に efficiency を示す。この試験における平均 efficiency は 0.98 であり、η を横軸に efficiency をプロットしたところ、満遍に再構成ができていることがわかった。

Wire Segment Reconstruction (フォワード領域)

図 7.7 に efficiency を示す。この試験における平均 efficiency は 0.97 であり、Strip Segment Reconstruction よりも efficiency が悪い。次の節で inefficiency の生じるメカニズムの精査を行う。

Wire-Strip Coincidence (フォワード領域)

図 7.8 に efficiency を示す。この試験における平均 efficiency は 0.92 だった。

Strip Segment Reconstruction の inefficiency2 % と Wire Segment Reconstruction の inefficiency3 % が独立な

[*5] ミューオン検出器の一つである Monitored Drift Tube 同士の相対位置補正をするのにレーザーを使っている。TGC を貫通させてレーザーを確認し合う必要があるため、TGC 検出器には穴があいている。この穴をアライメントホールと呼び、アライメントホールがある部分は不感領域となる。

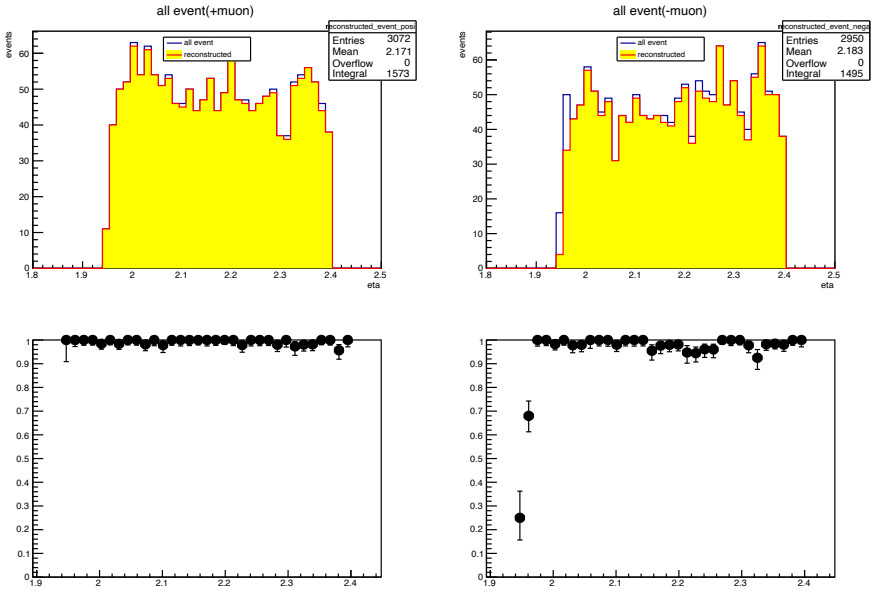

図 7.6　MC シミュレーションデータを使用した Strip Segment Reconstruction の efficiency。横軸は Truth の η。4 つの図のうち、左の図は正電荷、右の図は負電荷のミューオンによるイベントを表す。上の図は青線がイベントの全体数、赤線に黄色の地で塗られた部分は LUT から何らかの $\Delta\phi$ が出力されたイベントを示しており、下の図は efficiency を表している。

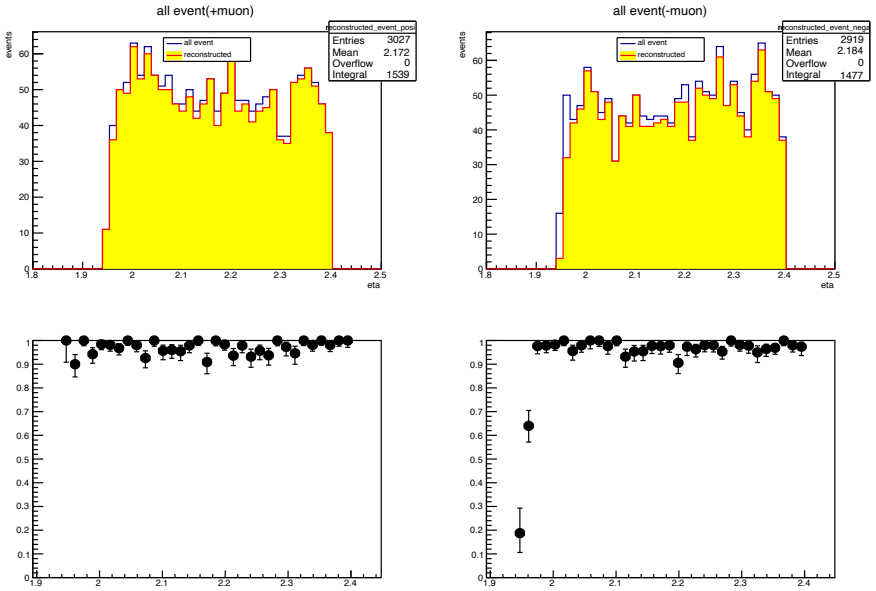

図 7.7　MC シミュレーションデータを使用した Wire Segment Reconstruction の efficiency。横軸は Truth の p_{T}。左の図は正電荷、右の図は負電荷のミューオンによるイベントを表す。上の図は青線がイベントの全体数、赤線に黄色の地で塗られた部分は LUT から何らかの $\Delta\phi$ が出力されたイベントを示しており、下の図は efficiency を表している。

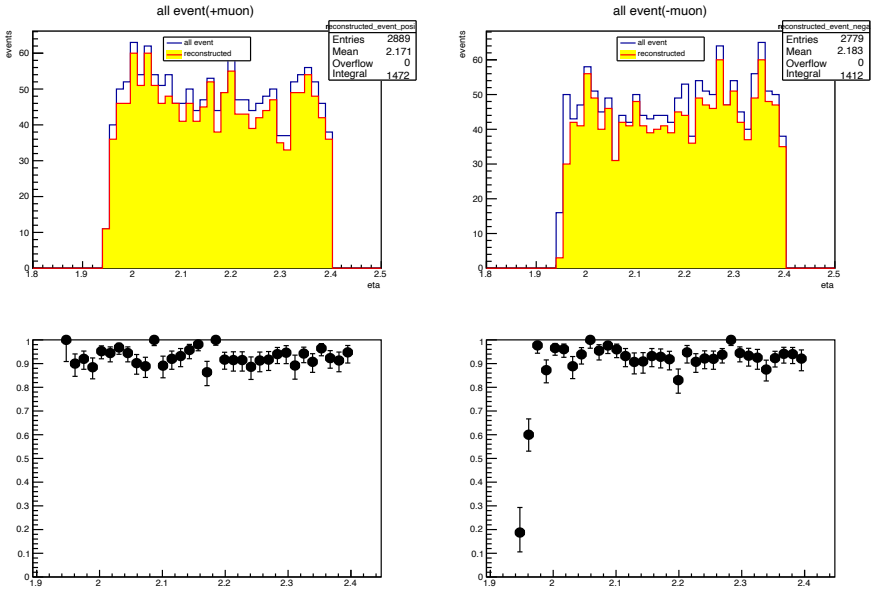

図 7.8　MC シミュレーションデータを使用した Wire Strip Coincidence の efficiency。横軸は Truth の p_{T}。左の図は正電荷、右の図は負電荷のミューオンによるイベントを表す。上の図は青線がイベントの全体数、赤線に黄色の地は LUT から何らかの $\Delta\phi$ が出力されたイベントを示しており、下の図は efficiency を表している。

inefficiency である場合でも、入力の欠けによる inefficiency は合計で 5 % であるので、3 % 以上は Wire-Strip Coincidence を由来とする inefficiency である。ファームウェアのモジュールを開発した先行研究(小林蓮 2021)では閾値以上のミューオンに対して 94 % の検出効率が見込まれていたため、ビットワイズシミュレーションのロジックに不備がある可能性が考えられる。この inefficiency については精査が必要であるが、本研究では時間の都合上、前段のモジュールである Segment Reconstruction の inefficiency の解析のみを行った。

7.4.3　Wire Segment Reconstruction の inefficiency の精査

後段の efficiency は前段の efficiency の影響を受けるため、まず Segment Reconstruction の解析を行った。ストリップよりもワイヤーの方が efficiency が悪かったため、ストリップとワイヤーの比較から、inefficiency の原因を解析した。

Segment Reconstruction では LUT を用いて $\Delta\theta$ または $\Delta\phi$ を再構成しており、再構成できない場合のメカニズムは 2 種類に分類される。一つ目は、subunit 内に M1〜M3 の 3 つのステーション全ての代表点が揃っていないため、map の key（ファームウェアでは URAM のアドレス）自体を作れないものである。2 つ目は、M1〜M3 ステーションに代表点があり、代表点の組み合わせから key 自体は生成できたが、LUT の中にデータが用意されていないものである。粒子の飛跡としてあり得ないパターンや、p_{T} が小さすぎる (飛跡の曲率が大きい) トリガーする必要のないパターンには URAM の中にデータを入れないため、LUT からデータを得られないパターンが存在する。

上述の 2 つの原因の分布を、ワイヤーとストリップで比較した (表 7.1、表 7.2)。この 2 つの表のうち、項目「全てのステーションにヒットがあるイベント」が後者の LUT の中にデータがないため再構成できなかったイベントに対応し、それ以外のものが代表点から key を作成できないため再構成できなかったイベントに対応する。この表からは、代表点の欠けにより再構成できなかったイベントの数はワイヤーとストリップでほぼ差がないが、「全てのステーションにヒットがあるイベント」は、ストリップの 71 イベントに対し、ワイヤーでは 145 イベントと、約 2 倍存在していたことがわかった。

表 7.1　Strip Segment Reconstruction において、MC データから生成した 6152 イベント中で再構成できなかったイベントの内訳

パターン	再構成されなかったイベント数	割合
全てのステーションにヒットがあるイベント	71	0.55
M1・M2 ステーションにヒットがあるイベント	45	0.35
M2・M3 ステーションにヒットがあるイベント	1	0.01
M1・M3 ステーションにヒットがあるイベント	7	0.05
M1 ステーションのみにヒットがあるイベント	5	0.04
M2 ステーションのみにヒットがあるイベント	0	0
M3 ステーションのみにヒットがあるイベント	0	0
どのステーションにもヒットがないイベント	1	0.01

表 7.2　Wire Segment Reconstruction において、MC データから生成した 6152 イベント中で再構成できなかったイベントの内訳

パターン	再構成されなかったイベント数	割合
全てのステーションにヒットがあるイベント	145	0.82
M1・M2 ステーションにヒットがあるイベント	43	0.21
M2・M3 ステーションにヒットがあるイベント	3	0.01
M1・M3 ステーションにヒットがあるイベント	7	0.03
M1 ステーションのみにヒットがあるイベント	5	0.02
M2 ステーションのみにヒットがあるイベント	2	0.01
M3 ステーションのみにヒットがあるイベント	0	0
どのステーションにもヒットがないイベント	1	0

また、トリガーの計算時間に制限があるため、Segment Reconstruction では subunit ごとに LUT に問い合わせられる代表点の組み合わせの数の上限が定められている。Wire Segment Reconstruction の場合、最大で 8 つの組み合わせに対応するデータを LUT から引き出すことができる。この構造から、代表点の数が多すぎたために LUT に問い合わせるべき真の飛跡の組み合わせが埋もれてしまい、問い合わせができていない可能性も考えられる。図 7.9 は横軸をイベントあたりの代表点の数[*6] にとり、縦軸を (表 7.2 の「全てのステーションにヒットがあるイベント」)/ (再構成できたイベント数 + 表 7.2 の「全てのステーションにヒットがあるイベント」) にとったものである。図 7.9 からは、代表点の数が増えるほど再構成に失敗する確率が上昇する傾向が見える。

ただし、LUT に問い合わせられる組み合わせの最大数は 8 であるため、(M3 に代表点が複数あった場合を除き) 代表点が 6 つまでのイベントでは、全ての組み合わせを LUT へ問い合わせられている。つまり、代表点が 6 つ以上のイベントの inefficiency は LUT にパターンのある組み合わせが優先順位が低く設定されてしまったためにデータを取り出せていないというメカニズムである可能性があるが、代表点が 3 つから 6 つまでの場合で inefficiency が生じている場合については、LUT 自体にパターンが存在していないと考えられる。

以上の結果から、さらに LUT からデータを得られなかったパターンをイベントごとに精査した。図 7.10 は、試験したイベントセットの中で再構成できなかったイベントのうち、最も番号が若いイベントにおける Segment Reconstruction へ入力された代表点の組み合わせを表している。このイベントではスタッガードチャンネル番号で

[*6] 理想的には subunit 内の代表点の数を横軸にするべきであるが、本試験ではパイルアップによるノイズのない MC データを用いているため、最初の簡単な解析としてこの解析ではイベントあたりの代表点の数を横軸にしている。

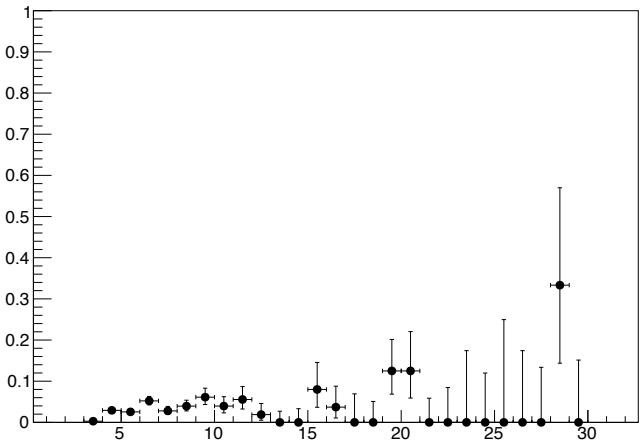

図 7.9　Wire Segment Reconstruction における、代表点の数に対する「全てのステーションにヒットがあるが再構成できなかったイベント」の割合。横軸は代表点の数、縦軸は (表 7.2 の「全てのステーションにヒットがあるイベント」)/ (再構成できたイベント数 + 表 7.2 の「全てのステーションにヒットがあるイベント」) である。

M1	M2	M3
224	224	224
225	225	225
226	226	226
227	227	227
228	228	228
229	229	229

図 7.10　Wire Segment Reconstruction で LUT からデータを得ることができなかったパターン 1。数字はスタッガードチャンネルであり、同じ数字のチャンネルは同じ η 領域を観測する。左から順に M1、M2、M3 ステーションを表し、黄色で塗られたチャンネルが Segment Reconstruction へ入力された代表点である。M1 に 228 番、M2 に 225 番、M3 に 225 番と 226 番にヒットがあり、全て 3/3 コインシデンスまたは 2/2 コインシデンスの出力であった。

M1	M2	M3
187	187	187
188	188	188
189	189	189
190	190	190
191	191	191

図 7.11　Wire Segment Reconstruction で LUT からデータを得ることができなかったパターン 2。数字はスタッガードチャンネルであり、同じ数字のチャンネルは同じ η 領域を観測する。左から順に M1、M2、M3 ステーションを表し、黄色で塗られたチャンネルが Segment Reconstruction へ入力された代表点である。

M1 に 228 番、M2 に 225 番、M3 に 225 番と 226 番にヒットがあった。Wire Segment Reconstruction では、M3 の 2/2 コインシデンスは subunit につき 1 つの代表点しか Address 生成に使用しないため、このイベントで LUT に問い合わせたのはは M1=228 番、M2=225 番、M3=225 番の組み合わせのみである。シミュレータへの LUT の入力に使用しているオリジナルデータファイルを確認したところ、この組み合わせに対応するデータは入っていなかった。また、M1、M2 の番号を固定したまま、M3 を 222、223、224 に変更した組み合わせのデータは LUT 内に存在していた。同様に、LUT への入力は作れたがデータが入っていなかったために再構成できなかったイベントのうち、2 番目に最も番号が若いイベントの精査も行った (図 7.11)。このイベントでは (M1,M2,M3 のスタッガードチャンネル番号) =(189,188,189)、(190,188,189) の 2 つの組み合わせが LUT に入力されるが、どちらの組み合わせも対応するデータが LUT 内に存在していなかった。

以上のことから、再構成できなかったイベントは、検出器自体がミューオンを観測する段階や LUT に問い合わ

せる過程で問題が生じているのではなく、LUT 自体にデータが入っていないために再構成に失敗していることが理解できた。また、図 7.10、図 7.11 をはじめとして他の再構成できなかったイベントも確認した結果、再構成されない飛跡は若干の折れ曲がりのあるパターンであることもわかった。

物理として、MC データのミューオンの飛跡に若干の折れ曲がりが生じる理屈については、Truth の情報を簡単に辿れないため推測になるが、ミューオンと検出器等の物質との相互作用で散乱している (多重散乱) ためであると考えられる。則ち、多重散乱事象に対応するデータが LUT 内に用意されていないための inefficiency と結論付けられる。この inefficiency は LUT にデータを入れることによって原理的に回復することができる。回復する場合は resolution と efficiency の間でのトレードオフになるため、ノイズやアライメント等の影響に対してロバストになるように、精度良くチェックする必要がある。

7.5　結論

以上のように、様々なテストベクターが生成できるようになったことと、ファームウェアと全く同じロジックのビットワイズシミュレータを開発したことにより、セクターロジックの検証システムを確立できたことが本研究の成果である。これにより、トリガー系の正常動作の検証 (7.2.1 節)、LUT の抜けの確認 (7.3 節)、MC データによる efficiency の評価 (7.4 節) のような様々な検証が行えるようになった。加えて、ビットワイズシミュレータではクラス (ファームウェアでのモジュール) ごとに出力を ROOT ファイル (図 7.5 の「SLsimu.root」) に記録するため、LUT で再構成できなかったイベントがどのようなヒットの組み合わせであったかについても、表 7.1、表 7.2 のように簡単に解析することができる。この情報はファームウェアの修正において有用な情報となる。

さらに、今後の展望として、LUT の変更のみならず、トリガーロジックの変更 (例：付録 D.4) による efficiency の変化についても、ファームウェアの変更をする前にこのビットワイズシミュレータによって確認できるようになる。

第 8 章

結論と今後の展望

2029 年から運転開始予定の高輝度 LHC-ATLAS 実験に向けて、Phase2 アップグレードが進行しており、エンドキャップミューオントリガーシステムではエレクトロニクスを全て刷新する予定である。本研究では Phase2 アップグレードの開発フレームワークとして初段トリガーシステムの一部であるセクターロジックのための検証機構の全体設計とその構成要素の開発を行った。開発した構成要素はケーブリング等の情報を一元管理するリレーショナル・データベースと、セクターロジックのトリガー系をビットレベルで再現したシミュレータ、セクターロジックへ入力するテストベクターの生成機構である。本研究で開発した検証機構は実際に運用も行なっており、ファームウェアの統合の検証や、MC データのイベントに対する応答のビットレベルでのシミュレーション試験などで、セクターロジックの開発研究に役立てている。

本研究ではシミュレータはエンドキャップ領域では intra station coincidece まで、フォワード領域では Wire-Strip Coincidence まで開発したが、今後も開発を続け、後段にある内部検出器とのコインシデンスを取る Inner Coincidence や Track Selector も開発し、セクターロジック内のトリガー系全体をシミュレートできるものにする予定である。トリガー系全体の完成により、先行研究で示されたトリガーレートの改善見込みの値をビットレベルの試験で検証することが可能になる。

また、テストベクター生成機構に関しても、現在は無限運動量飛跡のフォーマットではワイヤー・ストリップ合わせて 13 層全てにヒットがあるイベントしか生成できないが、今後は json ファイルのフォーマットを拡張することで任意のレイヤーのヒットを抜くこともできるようにし、2/3 コインシデンスなどを使用する場合の検証にも対応していく予定である。その他、本研究では使用しなかったランダムノイズ生成機能を使った efficiency 検証なども行えば、セクターロジックの開発において有意義な知見が得られると見込まれる。

本検証機構は現在のセクターロジックの開発段階のみならず、セクターロジック実機の運用時も見据えた設計となっているため、将来的には、運転中や、試運転中のファームウェアの最初の検証、トリガーシステムの診断などに用いられる予定である。

謝辞

本研究を遂行するにあたり、多くの方にお世話になりました。まず、指導教員である浅井祥仁教授には、研究を始めるにあたり、幅広い選択肢を用意していただきました。ご多忙の中でも快く相談に乗ってくださったり、事務的な部分のサポートもしてくださいました。博士課程進学後もよろしくお願いいたします。石野雅也教授と奥村恭幸准教授には、日頃から研究活動の様々な面で大変お世話になりました。研究開始までの丁寧な導入指導に始まり、研究内容の提案、技術面での詳細な助言など、幅広い面でお世話になりました。特に私は研究発表が苦手でしたが、これに関しても、御二方は根気強くご指導くださいました。

夏の学校等では、多くの ICEPP スタッフの皆様や学生の方々にお世話になりました。TGC に限らない ATLAS の検出器や、解析等について、幅広い知識をいただきました。また、秘書の皆様にも出張手続きなどの様々な面で支えていただきました。

主にセクターロジックのファームウェア関係では、同期の学生の方々にお世話になりました。石野研究室の三島さんには、実機試験・ファームウェアシミュレーションの試験の出力を提供していただき、HDL に関する技術的な助言もいただきました。名古屋大学の鍋山さんには、Wire Segment Reconstruction のファームウェアや LUT の詳細について教えていただきました。京都大学の河本さんには、Strip Segment Reconstruction および Wire-Strip Coincidence のファームウェアのロジックを丁寧に教えていただきました。また、リレーショナル・データベースのスタッガードチャンネルの番号の定義では、京都大学の三野さんに各チャンネルの座標の情報を提供していただきました。

石野研究室の青木先輩、奥村研究室の林先輩にもお世話になりました。青木さんにはファームウェアシミュレーションに関する技術的な助言をいただきました。Run3 の研究をしている林先輩との会話の中では、現行のシステムの理解を深めることができました。

また、他の ICEPP の同期の方々にもお世話になりました。古川さんにはプログラムに関する一般的なことや発表資料の見せ方など、幅広いことについて相談に乗っていただきました。同じ浅井研究室の寺尾さんには TA などでお世話になりました。

最後に、研究生活を支えてくださった家族に感謝します。

付録 A

リレーショナル・データベースの構造

このリレーショナル・データベースは TGC 内のチャンネルから後段のエレクトロニクス内でのチャンネルの情報までを統一的に取り扱うためのものであり、TGC Big Wheel の円盤内で ϕ 方向に 24 回繰り返される TGC 検出器とエレクトロニクスのケーブリングを記述する。

A.1　1/24 繰り返し構造の配線

TGC Big Wheel のチェンバーは、24 回繰り返し構造をしており、この中には Forwrad のチェンバーが 1 枚、Endcap のチェンバーが 2 組ずつ（M1 では 4 枚 ×2 の 8 枚、M2・M3 では 5 枚 ×2 の 10 枚）存在している。コインシデンスの計算を行う Sector Logic はこの 1/24 セクター単位の情報を取り扱うため、このデータベース内でケーブリングを記述する部分は（24 回繰り返し構造ではない EIL4 を除き）、1/24 セクター構造で取り扱うのが合理的である。この範囲には 6408 の検出器チャンネルが存在している。ここからは 1/24 の構造を記述する。

この 6408 個の TGC 検出器チャンネルから得られたヒット情報は、まず ASD によって増幅され、閾値をかけられてデジタル化される。この ASD が載っているボードでは、最大 16 チャンネルを処理することができる。1/24 セクターにつき、417 枚の ASD ボードがある。

ASD でデジタル化されたヒット情報は PS ボード上の PPASIC に入力される。PPASIC では、ヒット情報にケーブル長に応じた fine delay や BCID ゲートをかけて LHC クロックと同期させる。PPASIC で整形された情報は FPGA でまとめられ、2 本のリンクで Sector Logic に出力する。1/24 セクターにつき、PS ボードは 29 枚あり、1 枚の PS ボードには 8 個の PPASIC と 1 個の FPGA が乗っている。PPASIC には A/B の 2 つのポートが存在しており、それぞれ 1 枚の ASD ボードの情報（16 チャンネル分）を処理できるので、1 枚の PS ボードでは最大で 16 枚の ASD ボード、256 個の検出器の情報を取り扱うことができる。

FPGA から伸びる 2 本のリンクは、12 本ずつにまとめられて USA15 にある Sector Logic まで配線される。これは MPO24 という、出力信号線 12 本と入力信号線 12 本が束になったケーブルを使っているため、12 本の単位で取り扱っている。Sector Logic には 24 個の Bank が存在し、その中にある 4 チャンネルの GTY で情報の入出力を行なっている。GTY のチャンネル 1 つにつき FPGA から伸びるリンク 1 本が対応するため、ヒット情報が入力されるチャンネルは FPGA 29 個 ×2 で 58 チャンネルであり、これは Bank15 個に対応する。

Sector Logic に入力された情報は、まずサブセクターごと、レイヤーごとに並べ直されて整理される。この過程をここでは Mapping と呼称する。この時、η 方向の重複を考慮した OR をとる。これを Wired OR と呼ぶ（詳細は 4.2.2 節および 4.2.2 節）。整理された情報は、ステーション内でコインシデンスをとり、デクラスタリングをかけた上でステーション間コインシデンスへ入力される。ヒット情報はさらに処理をされ後段へと流れていくが、このデータベースはステーション内コインシデンスまでを取り扱うものなので、ここでは踏み入らない。

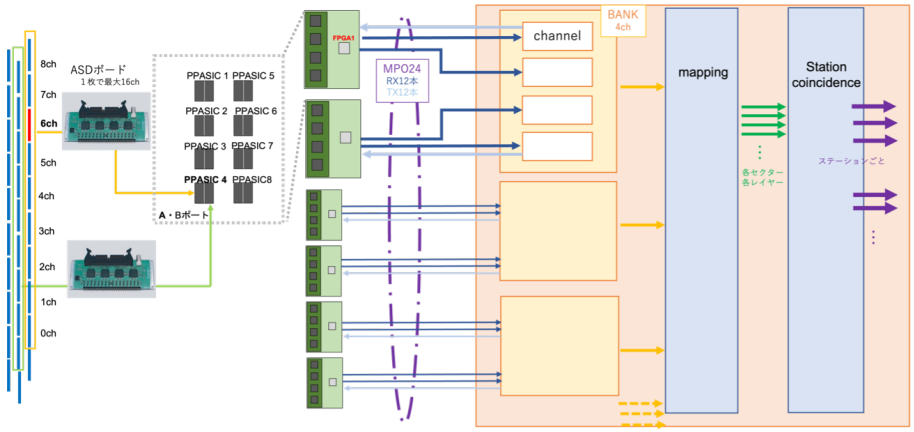

図 A.1　検出器チャンネルから Sector Logic 入力までの模式図

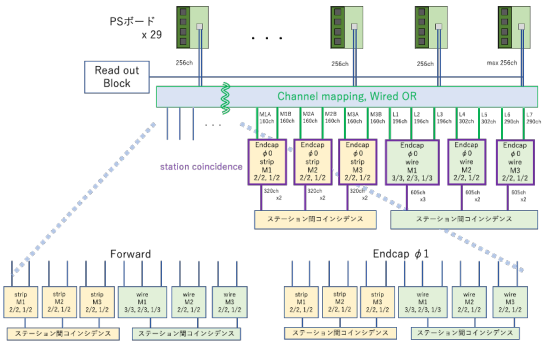

図 A.2　Sector Logic 内のトリガー系で行われるヒット情報の処理の一部のダイアグラム

A.2　レイヤーの表記法

このデータベースでは、Wire と Strip で異なるレイヤーの表記法を用いる。

Wire の ASD およびチャンネルを取り扱う時は、衝突点側から数えて何枚目かという意味で、「L（レイヤー）1」、「L2」、. . . 「L7」の形で表記する。一方、Strip の情報を取り扱う際には、M_ ステーションの A(/B) レイヤーという意味で、「M1-A」、「M1-B」、. . .「M3-B」と表記する。A/B レイヤーの定義は以下の通りである。Layer2 には Strip のチャンネルは存在しない。

- A レイヤー
  - Forward チェンバーの L1,4,6（lower-Z）
  - Backward チェンバーの L3,5,7（higher-Z）
- B レイヤー
  - Forward チェンバーの L3,5,7（higher-Z）

– Backward チェンバーの L1,4,6（lower-Z）」

Strip でこの表記法を用いる理由の詳細は付録 F に記述する。

付録 B

リレーショナル・データベースを構成する表

データベースは複数の表で構成されている。検出器チャンネルの単位、ASD の単位など、チャンネルの有する属性や配線の規模には階層性があるため、階層ごとに分割して表を作成した。

SQL の表では列（カラム）ごとに「FPGA」などの項目名がつき、数値や文字列などの属性（データ型）が定められる。例として下の表「3_1_Cabling」を用いて説明する。一つの行には FPGA、PPASIC、. . . 、ϕ の組み合わせ（レコード）が入る。レコードを一意に決める項目（一つまたは複数の項目の組み合わせ）を主キーと呼び、下の表では FPGA、PPASIC、Port が主キーである。

この表で使用している属性は以下の 5 種類である。

INT	整数
DOUBLE	小数
CHAR(n)	n 文字の文字列
VARCHAR(n)	n 文字までの可変長文字列
TINYINT(1)	真理値

以下ではそれぞれの表に含めた、個々の項目及び主キー、属性などの情報を記述する。

1. ステーション内コインシデンスの input-output(スタッガードチャンネルと検出器チャンネル)
2. ASD・ASD ピンとステーションコインシデンス入力チャンネル
3. ASD 単位での各属性/各コンポーネントの情報/コネクタの情報
4. 各コンポーネントの運用状況
5. EIL4
6. データフォーマット

セクション 1・2・3 は、1/24 セクター内での各属性の構造を記述したものである。これらは各チャンネル、コンポーネント間の接続状況の検索に用いることができる。1/24 セクター内でのファームウェアの作成に用いるほか、

FPGA	PPASIC	Port	ASD_name	phi
1	1	A	EW0-1	0
1	1	B	EW0-2	0
⋮	⋮	⋮	⋮	⋮

図 B.1 「3_1_Cabling」の抜粋

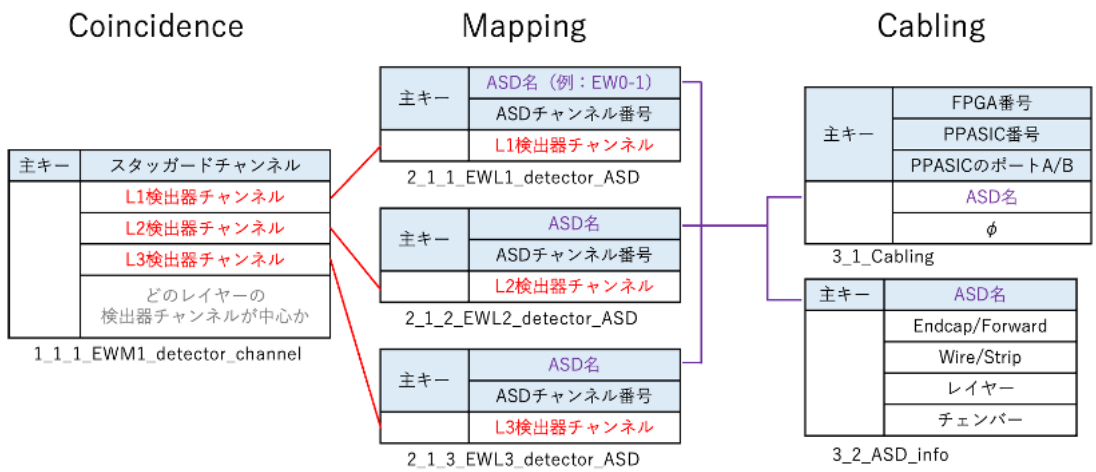

図 B.2　表 1〜3 群における表同士の関係の構造（EW、M1 部分を抜粋）

下記の 4 セクションの表を結合するための補完情報としても使用する。

表の中の一つのレコードが示す観測領域（チャンネル数や $\eta \cdot \phi$ のカバー域）は、セクション 3 ＞ セクション 2 ＞ セクション 1 の順で広くなる。セクション 1 に属する表（以下、セクション X に属する表の集合を表 X 群と呼称する）には、スタッガードチャンネル単位での情報が入っており、これはこのデータベースの中で最も細かい単位である。表 2 群のレコードは検出器チャンネルの単位での情報を取り扱っている。表 3_1, 表 3_2 には ASD 単位での情報が入っており、表 3_3 はコネクタ単位での情報が入っている。

セクション 4 に記述したのは、TGC 検出器全体でのコンポーネントの運用状況を管理するための表である。これはチャンネル、チェンバー、ASD をそれぞれ管理するための 3 枚の表からなっている。上の 3 つのセクションで用いた、1/24 セクター内でユニークなチャンネル名に加えて、サイド、セクター、ϕ の情報を追加することで全検出器内でチャンネルやチェンバーをユニークに指定する。これらは運用中のマスク情報やスレッショルドの管理などに用いることを想定している。

セクション 5 は EIL4 のケーブリング等の構造を記述したものである。EIL4 は Big Wheel とは異なり、24 回繰り返し構造ではないため、独立したデータベースとして項目を作成した。Big Wheel の表 1・2・3 群に相当する 3 枚の表で記述した。

セクション 6 は、データフォーマットを SQL の表に落とし込んだものである。これらはセクション 1・2・3 と結合させることで、ファームウェアの記述に用いる。

B.1　ステーション内コインシデンスの INPUT-OUTPUT

表 1 群はステーション内コインシデンスの input-output 対応を記述した表である。Endcap/Forward のサブセクターごと、Wire /Strip ごと、スターションごとに表を分けて作成した。Strip に限り、どのステーションでも構造が同じであるため、M1・M2・M3 を一つの同じ表で取り扱っている。よって、表 1 群は Wire で 6 枚（Endcap/Forward で 3 枚ずつ）、Strip で 2 枚の合計 8 枚の表からなる。

下の表はそのうちの一枚、Endcap Wire M1 ステーションを記述した表「1_1_1_EWM1 detector channel」の項目を並べたものである。

この表の主キーは、ステーション内コインシデンスの出力となる「スタッガードチャンネル」であり、これはこのデータベースの中で最も単位の細かい要素である。

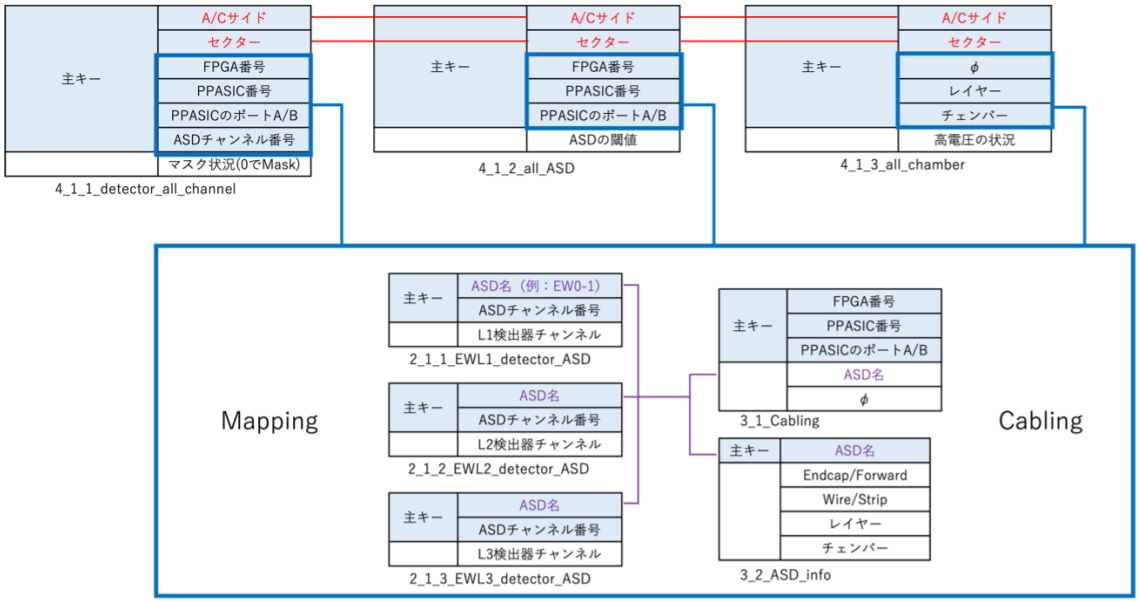

図 B.3　上部の表 4 群同士は下の枠内の表 2-3 群を介して参照可能な関係になっている

		セクター		レイヤー/ステーション		表名		属性			
								属性	名前	変数型	定義域
1	スタッガードチャンネルと検出器チャンネル	1-1	EW	1-1-1	M1	1_1_1_EWM1_detector_channel	主キー	スタッガードチャンネル	EW_staggered	INT	1-605
								L1検出器チャンネル	EWL1_detector_serial	INT	1-196
								L2検出器チャンネル	EWL2_detector_serial	INT	1-196
								L3検出器チャンネル	EWL3_detector_serial	INT	1-196
								どのレイヤーの検出器チャンネルが中心か	Center	INT	44199
				1-1-2	M2	1_1_2_EWM2_detector_channel	主キー	スタッガードチャンネル	EW_staggered	INT	1-605
								L4検出器チャンネル	EWL4_detector_serial	INT	1-302
								L5検出器チャンネル	EWL5_detector_serial	INT	1-302
				1-1-2	M3	1_1_3_EWM3_detector_channel	主キー	スタッガードチャンネル	EW_staggered	INT	1-605
								L6検出器チャンネル	EWL6_detector_serial	INT	1-290
								L7検出器チャンネル	EWL7_detector_serial	INT	1-290
		1-2	FW	1-2-1	M1	1_2_1_FWM1_detector_channel	主キー	スタッガードチャンネル	FW_staggered	INT	1-316
								L1検出器チャンネル	FWL1_detector_serial	INT	1-105
								L2検出器チャンネル	FWL2_detector_serial	INT	1-104
								L3検出器チャンネル	FWL3_detector_serial	INT	1-105
				1-2-2	M2	1_2_2_FWM2_detector_channel	主キー	スタッガードチャンネル	FW_staggered	INT	1-317
								L4検出器チャンネル	FWL4_detector_serial	INT	1-125
								L5検出器チャンネル	FWL5_detector_serial	INT	1-125
				1-2-3	M3	1_2_3_FWM3_detector_channel	主キー	スタッガードチャンネル	FW_staggered	INT	1-311
								L6検出器チャンネル	FWL6_detector_serial	INT	1-122
								L7検出器チャンネル	FWL7_detector_serial	INT	1-122
		1-3	ES			1_3_ES_detector_channel	主キー	スタッガードチャンネル	ES_staggered	INT	1-320
								Aレイヤー_ステーション内コインシデンス入力	A_Layer_detector_serial	INT	0-159
								Bレイヤー_ステーション内コインシデンス入力	B_Layer_detector_serial	INT	0-159
		1-4	FS			1_4_FS_detector_channel	主キー	スタッガードチャンネル	FS_staggered	INT	1-64
								Aレイヤー_ステーション内コインシデンス入力	A_Layer_detector_serial	INT	0-31
								Bレイヤー_ステーション内コインシデンス入力	B_Layer_detector_serial	INT	0-31

図 B.4　表 1 群のリスト

	項目	名前	変数型	定義域
主キー	スタッガードチャンネル	EW_staggered	INT	1-605
	L1 検出器チャンネル	EWL1_detector_serial	INT	1-196
	L2 検出器チャンネル	EWL2_detector_serial	INT	1-196
	L3 検出器チャンネル	EWL3_detector_serial	INT	1-196
	どのレイヤーの検出器チャンネルが中心か	Center	INT	1-3

図 B.5　表「1_1_1_EWM1_detector_channel」の項目

EWL3_detector_serial （レイヤー3通し番号） (1-196)	EWL2_detector_serial （レイヤー2通し番号） (1-196)	EWL1_detector_serial （レイヤー1通し番号） (1-196)	EW_staggered （スタッガードチャンネル） (4-593)	Center （どのレイヤーが中心か） (1-3)
⋮	⋮	⋮	⋮	⋮
196	196	195	15	1
196	195	195	16	2
195	195	195	17	3
195	195	194	18	1
195	194	194	19	2
194	194	194	20	3
194	194	193	21	1
194	193	193	22	2
⋮	⋮	⋮	⋮	⋮

図 B.6　表「1_1_1_EWM1_detector_channel」の抜粋

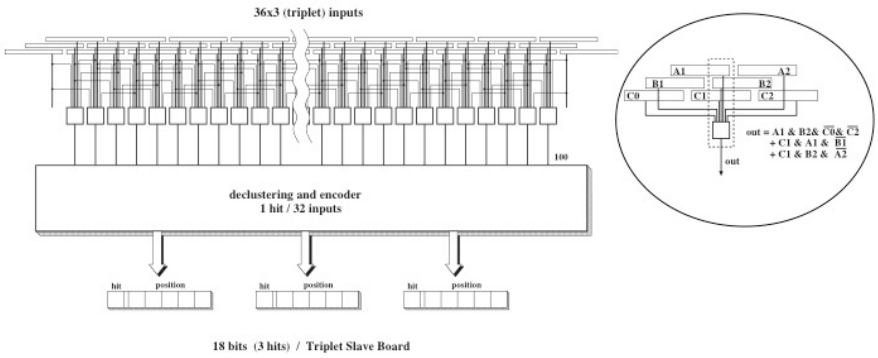

図 B.7　Run2 時点におけるトリプレットのステーション内コインシデンス回路（ATLAS Collaboration 1998）

スタッガードチャンネルは、ヒット情報を得た検出器チャンネルの組み合わせによって定義されるものである。基本的にステーション内で扱うものであるので、M1,M2,M3 のステーション毎に、「スタッガードチャンネル」と 2〜3 列の「L_ 検出器チャンネル（＝ステーション内コインシデンス入力）」からなる表を作成した。

ワイヤーの方での「ステーション内コインシデンス入力」はチェンバー境界で WiredOR を取った後の、レイヤー内通し番号 (図 4.2) に対応する。ストリップの場合は、4.2.2 節で定義した「ステーション内コインシデンス入力」と同義である。

ストリップはどのステーションでも、「ステーション間コインシデンス入力」と「スタッガードチャンネル」の番号の組み合わせが同じであるため、1 つの表のみで取り扱うことができる。

また、トリプレットのチャンネルには「Center」列を追加している。これはスタッガードチャンネルがどのレイヤーの検出器の中心に位置しているかを示すものである。トリプレットでは、スタッガードチャンネルの中心か、左右どちらかにずれているかで、コインシデンスの論理回路中の ABC レイヤーのどれに当てはまるかが変わる。「Center」列は、レイヤー 123 と ABC レイヤーの対応をとるときの参考に用いる。

B.2　ASD & ASD ピンとステーションコインシデンス入力チャンネル

表 2 群は Mapping に用いる対応表である。Mapping とはサブセクター/レイヤーごとにチャンネルを一列に並べる工程であり、このとき、Wired OR を考慮してチャンネルの OR をとる。Endcap/Forward のサブセクターごと、Wire /Strip ごと、レイヤーごと（Wire7 枚、Strip6 枚）で作成したため、合計で 26 枚の表が存在している。

下の表はそのうちの 2 枚、Endcap Wire Layer1 を記述した表「1_2_1_EWL1_detector_ASD」と Endcap Strip Layer1 を記述した表「1_3_1_ESM1A_detector_ASD」の項目を並べたものである。

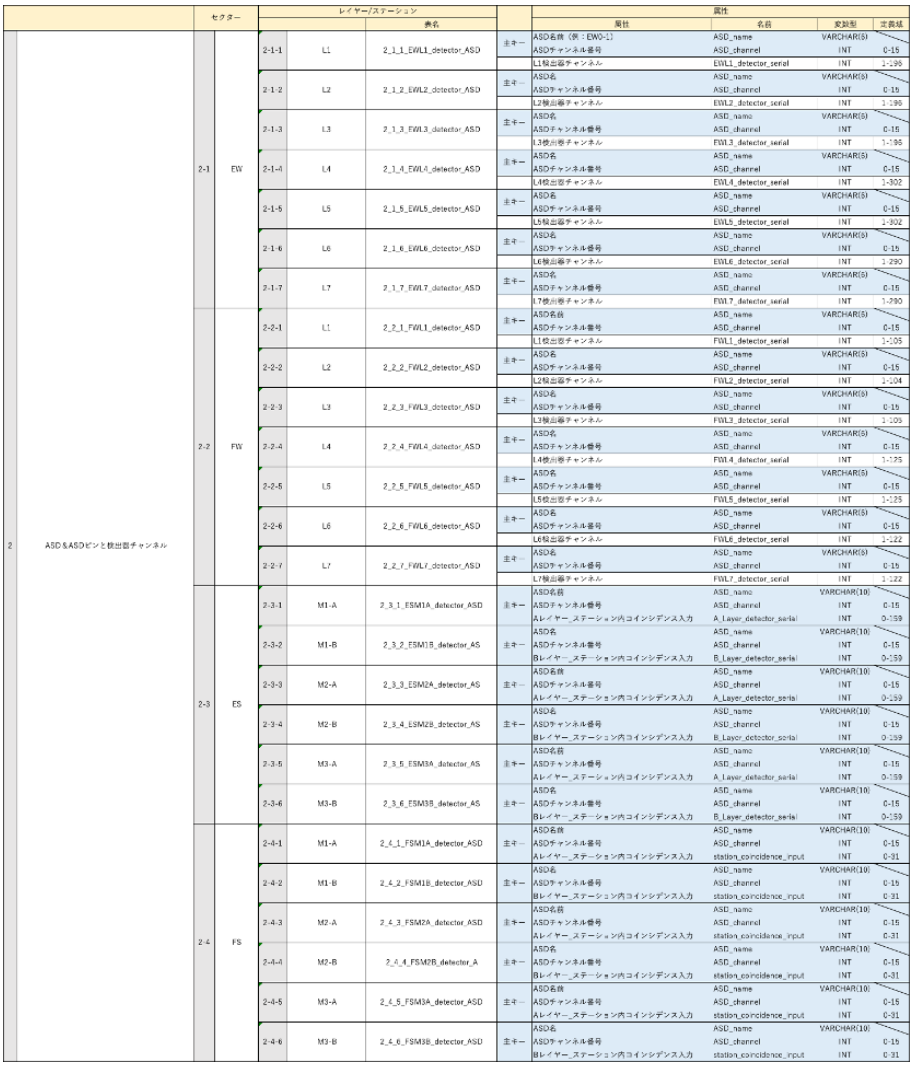

図 B.8　表 2 群のリスト

	属性			
	属性	名前	変数型	定義域
主キー	ASD名前（例：EW0-1）	ASD_name	VARCHAR(6)	
主キー	ASDチャンネル番号	ASD_channel	INT	0-15
	L1検出器チャンネル	EWL1_detector_serial	INT	1-196
	チェンバー内でのローカルなチャンネル番号	local_ID_per_chamber	INT	1-92
	モジュール内(VHDL)での信号線の番号	serial_ch_forVHDL	INT	0-195

図 B.9　表「1_2_1_EWL1_detector_ASD」の項目

	属性	名前	変数型	定義域
主キー	ASD 名（例:EW0-1）	ASD_name	VARCHAR(6)	
主キー	ASD チャンネル番号	ASD_channel	INT	0-15
	L1 検出器チャンネル	EWL1_detector_serial	INT	1-196
	チェンバー内でのローカルなチャンネル番号	local_ID_per_chamber	INT	1-92
	モジュール内 (VHDL) での信号線の番号	serial_ch_forVHDL	INT	0-195

表 B.1　表「1_2_1_EWL1_detector_ASD」の項目

	属性	名前	変数型	定義域
主キー	ASD名前	ASD_name	VARCHAR(10)	
	ASDチャンネル番号	ASD_channel	INT	0-15
	Aレイヤー_ステーション内コインシデンス入力	A_Layer_detector_serial	INT	0-159
	チェンバー内でのローカルなチャンネル番号	local_ID_per_chamber	INT	1-32
	モジュール内(VHDL)での信号線の番号	serial_ch_forVHDL	INT	0-159

図 B.10　表「1_3_1_ESM1A_detector_ASD」の項目

この表では ASD 名「ASD_nam」と ASD チャンネル番号「ASD_channel」の組み合わせがステーションコインシデンス入力チャンネル「EWL1_detector_serial」と対応している。このステーションコインシデンス入力チャンネルは、各レイヤー内での検出器チャンネルの通し番号と言い換えることができる。

Channel Mapping のファームウェア内（使用言語は VHDL）での信号線の番号「serial_ch_forVHDL」はステーションコインシデンス入力チャンネルと 1 対 1 で対応している。これはファームウェアの自動生成に用いるための列である。「ステーションコインシデンス入力チャンネル」は現行のエレクトロニクスから踏襲した番号であり、ビーム軸側（$\eta = \infty$）が 0 番になるように定義されているのに対して、「serial_ch_forVHDL」はビーム軸から遠い側（$\eta = 0$）が若い番号になるように定義されている。Channnel Mapping のモジュール（128bit $\times$ 58 links の情報から SLR、レイヤー、チャンネル番号の情報に読み直すモジュール）から、この順番で取り扱うことにより、ソフトウェアと同じ並び順（ビーム軸から遠い側（$\eta = 0$）が若い番号）を採用している後段のステーション内コインシデンスへ滑らかに接続できるようになっている。

チェンバー内でのローカルなチャンネル番号「local_ID_per_chamber」は、ステーションコインシデンス入力チャンネルと同じく、ビーム軸側を 0 番として振られている。これは Run3 から踏襲したデータである。

4.2.2 章で述べた通り、Wire に於いては Phase-1 で定義されていた通し番号を踏襲して使用したが、Strip では新しく定義を行った。Wire と異なり、Strip の表ではステーション内コインシデンス入力も主キーに含まれているのは、ここで Wired OR を考慮しているためである。Wire の表では、ステーション内コインシデンス入力は、ほとんどのチャンネルでは 1 チャンネルにつきひとつの検出器チャンネル（ASD/ASD チャンネルの組み合わせ）が入力され、チェンバー境界のオーバラップ部分にだけ 2 つの検出器チャンネルが入力される。一方 Strip で Wired OR を考慮すると、M1 及び M2 ステーション内のひとつの検出器チャンネルがコインシデンスをとる M3 ステーションのピボットは、最大で 3 点に及ぶ。よって、M1 と M2 の検出器チャンネルは、ステーション内コインシデンス入力の列に並べるときに 2 つあるいは 3 つに複製される。ASD/ASD チャンネルの組み合わせに対して、複数のステーション内コインシデンス入力が存在しうるため、Strip では 3 項目全てが主キーとなっている。下に示す表「2_3_1_ESM1A_detector_ASD」の抜粋に具体的な複製のされ方を示す。

ここでは同じチェンバーに属するチャンネルを同じ色で表記している。

M1 の同じチェンバーからステーション内コインシデンス入力に投影されるチャンネルを、同じ色で示した。このように M1 を基準にして見ると、一つのヒット情報が 2 つあるいは 3 つに複製されているのがわかる。

B.3　ASD 単位での各属性/各コンポーネントの情報/コネクタの情報

表 3 群は ASD または FPGA から Sector Logic へ伸びるリンク単位での表からなる。

ASD_name	ASD_channel	A_Layer_detector_serial
ES0-M1-A	0	1
⋮	⋮	⋮
ES0-M1-A	0	65
ES2-M1-A	0	65
⋮	⋮	⋮
ES2-M1-A	0	129
ES4-M1-A	0	129
⋮	⋮	⋮
ES2-M1-A	0	193
ES4-M1-A	0	129
ES6-M1-A	0	193
⋮	⋮	⋮
ES4-M1-A	0	257
ES6-M1-A	0	257

図 B.11　表「1_3_1_ESM1A_detector_ASD」の抜粋

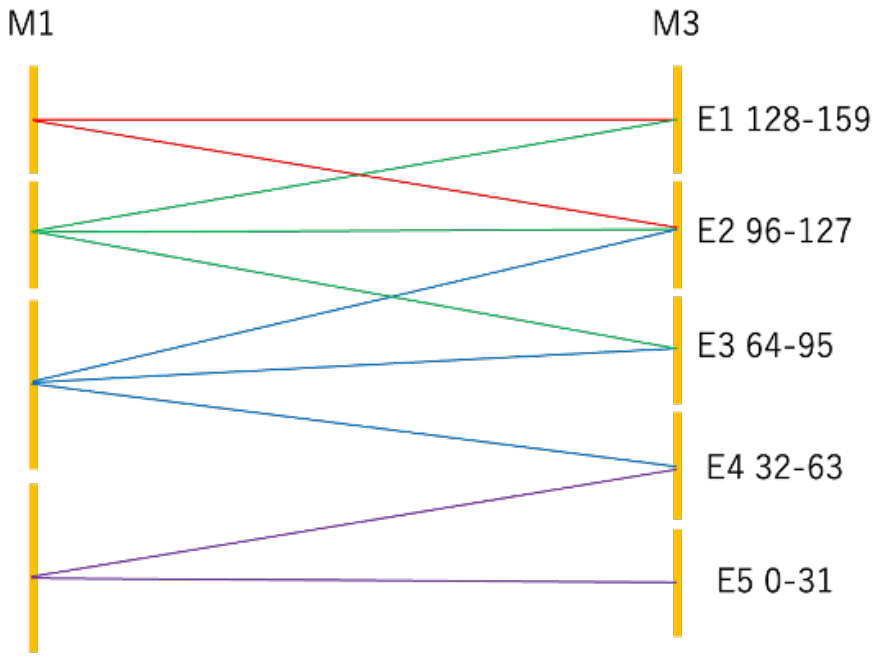

図 B.12　M1 チェンバーと M3 チェンバーにおける、コインシデンスをとる領域の対応図

	属性	名前	変数型	定義域
主キー	FPGA 番号	FPGA	INT	1-29
	PPASIC 番号	PPASIC	INT	1-8
	PPASIC のポート A/B	Port	CHAR(1)	A,B
	ASD 名	ASD_name	VARCHAR(6)	
	ϕ	phi	INT	0,1

図 B.13　表「3_1_Cabling」の項目

	属性	名前	変数型	定義域
主キー	ASD 名	ASD_name	VARCHAR(6)	
	Endcap/Forward	Endcap_or_Forward	CHAR(1)	
	Wire/Strip	Wire_or_Strip	CHAR(1)	
	レイヤー	Layer	VARCHAR(4)	
	チェンバー	Chamber_E	INT	1-5

図 B.14　表「3_2_ASD_info」の項目

表 B.2　サブセクターと SL 上の Bank やチャンネルの対応関係

リンク番号	サブセクター	SLR	bank,channel
0-23	Endcap ϕ0	SLR0	bank120-122,220-222 のそれぞれの 4 channels
24-51	Endcap ϕ1	SLR2	bank129-131,228-231 のそれぞれの 4 channels
52-61	Forward	SLR3	bank232-233 のそれぞれの 4 channels, および bank234 の 2 channels

B.3.1　ケーブリング

表「3_1_Cabling」は PS ボード上で、ASD がどの PPASIC に配線されているかを示すものである。1/48 セクターの Endcap には phi0 と phi1 のそれぞれに同じ ASD があるため、ASD/phi の組み合わせが FPGA/PPASIC/Port と 1 対 1 で対応する。

B.3.2　ASD の属性

表「3_2_ASD_info」は ASD の名前からわかる情報を開くためのものである。ASD の命名法は、1 文字目で Endcap/Forward、2 文字目で Wire/Strip を示し、その後に「そのレイヤー内で high eta 側から 0 から数えて何番目の ASD ボードか」、「何枚目のレイヤーか」を表す数字が続く。加えて、どの ASD がどのチェンバーに刺さるかも ASD 単体が主キーとなる情報なので、この表に含めた。

B.3.3　GTY_Bank

表「3_3_GTY_Bank」は FPGA から Sector Logic へ伸びるリンク単位で記述される表である。この表では GTY の Bank とチャンネルを主キーとした。主としてこの Google spreadsheet（リンク）の情報が記述されている。加えて、リンクの通し番号も追加した。

リンクの通し番号とは、テストパルスパターンを記述した coe ファイルに付けられている 0 から 61 の番号のことであり、この coe ファイルは Sector Logic にある FPGA の BRAM に書き込まれている。

	属性	名前	変数型	定義域
主キー		GTY_Bank	INT	
		GTY_Channel_X	INT	
		GTY_Channel_Y	INT	
	SLR	SLR	INT	0-3
		SLR_X	INT	
		TX_Signal_name	VARCHAR(10)	
	TX に Polarity_flip があるときは TRUE	TX_Polarity_flip	BOOLEAN	
	TX の FireFly のコネクタ	TX_FireFly_channel_CN	INT	
	TX の FireFly のチャンネル	TX_FireFly_channel_Ch	INT	
		RX_Signal_name	VARCHAR(10)	
	RX に Polarity_flip があるときは TRUE	RX_Polarity_flip	BOOLEAN	
	RX の FireFly のコネクタ	RX_FireFly_channel_CN	INT	
	RX の FireFly のチャンネル	RX_FireFly_channel_Ch	INT	
	何に出力するか（FPGA, etc）	TX_FROM	VARCHAR(20)	
	出力する FPGA の番号	TX_FPGA	INT	
	何から入力されるか（FPGA, etc）	RX_FROM	VARCHAR(20)	
	入力される FPGA の番号	RX_FPGA	INT	1-29
	入力される FPGA のリンク	RX_Link	INT	1,2
	データフォーマット。 α=1，β=2，γ=3	Data_format	INT	1-3
	出力速度（単位は Gbps）	TX_speed_Gps	DOUBLE	
	入力速度（単位は Gbps）	RX_speed_Gps	DOUBLE	

図 B.15　表「3_3_GTY_Bank」の項目

このうち、36 番から 39 番は Big Wheel ではなく EIL4 の PS ボードからの入力に対応する。また、channel Mapping のモジュール内で使われる 0-23 あるいは 0-9 のリンクの通し番号も、この 62 本の全体のリンクの通し番号と同じ並びになっている。channel Mapping のモジュールとは、ヒット情報がリードアウト系とトリガー系が分かれてから、最初にトリガー系リードアウト系から分かれてから最初に位置するトリガー系のモジュールである。このモジュールにはヒット情報がリンクごとに区切って入力されている。

ここで、GTY のチャンネルの中には、FPGA の RX と接続していないものがあることに注意されたい。このため、GTY の Bank とチャンネルの組み合わせと、FPGA 番号 (1-29) とリンク (1-2) の組み合わせは一対一で対応しているが、後者の組み合わせを主キーとして用いることはできない。

ここで、SQL の表に収めるにあたり、いくつかの項目を複数のカラムに分割して記述した。項目「FPGA GTY channel」は Google spreadsheet では「X0Y0」、「X0Y1」のように記述されているが、SQL の表では X の後に続く数字を項目「GTY_Channel_X」、Y の後に続く数字を項目「GTY_Channel_Y」に分けて記述した。同じく、項目「TX_FireFly_channel」も Google spreadsheet では「CN29 TX7」のように記述されているところを、SQL では CN の後の数字を「TX_FireFly_channel_CN」、TX の後の数字を「TX_FireFly_channel_ch」に分けて記述した。これは入力信号線「RX_FireFly_channel」も同様である。また、Google spreadsheet の項目「Fiber link」の TX・RX も、シート内では 1 列で表されるところを「何からの入 (出) 力か」、「(FPGA からの入出力ならば) FPGA 番号」、「リンクの番号」の 3 列に分割している。

	属性	名前	変数型	定義域
主キー	A/C サイド	Side	CHAR(1)	
	セクター	Sector	INT	1-24
	FPGA 番号	FPGA	INT	1-29
	PPASIC 番号	PPASIC	INT	1-8
	PPASIC のポート A/B	Port	CHAR(1)	A,B
	ASD チャンネル番号	ASD_channel	INT	0-15
	雑音状況	Noisy	TINYINT(1)	0,1

図 B.16　表「4_1_1_detector_all_channel」の項目

B.3.4　3_4_for_module_port_Number

表「3_4_for_module_port_Number」は PS ボードの FPGA 番号と Link を主キーとする表である。レコード 1 つあたりの規模は「PS ボードから SL に伸びる Link」であるため、表 3_3 と同じであるが、この Link に紐づけられる要素のうち、配線の変更を受けないものを表 3_4 にまとめた。これにより、配線の変更に対して、データベースの修正を最小限に抑えられるようにしている。

この表の主キーは「FPGA」(PS ボードの FPGA 番号:1-29) と「Link_12」(PS ボードから伸びる 1 または 2 の Link の番号) であり、データフォーマットの種類を示す列「Data_format」を NOT NULL で管理している。

使用されているデータフォーマットの種類は α, β, γ の 3 種類があるが、表の中ではこの 3 種を INT 型で取り扱い、1、2、3 の数字で区別している。データフォーマットの中身の詳細は B.6 章で記述されており、表「6_1_FPGA_dataformat」と結合するときにこのカラム「Data_lformat」を用いる。

B.4　個々のコンポーネントの性質の管理

表 1 群から表 3 群までは 1/24 セクター内での構造を記述するものだが、表 4 群は検出器内に実際に存在する個々のコンポーネントの性質を記録するものである。よって、表 1 群から表 3 群での記法に加えて、A/C サイドの区別とセクター番号を付け加えることによって、検出器全体でコンポーネントをユニークに表記している。

B.4.1　全検出器チャンネルの運用状況

表「4_1_1_detector_all_channel」は検出器チャンネルの情報を管理するためのものである。表 2 群で検出器チャンネルを表現するのに使った組み合わせ「FPGA 番号, PPASIC 番号, PPASIC のポート, ASD チャンネル番号」に「A/C サイド, セクター (1-24)」を掛け合わせることで、個々の検出器チャンネルを表現している。

この表の中には実際に検出器に接続しているチャンネルのみが入っているため、考えうる主キーの組み合わせ全ての FPGA 29 個 ×PPASIC 8 個 ×Port 2 個 ×ASD チャンネル 16 個 ×24 セクター ×2 サイド=356,352 行ではなく、6408 チャンネル ×24 セクター ×2 サイド=307,584 行からなる。

現在管理している検出器チャンネルの情報は雑音状況（Noisy）であり、これは正常に使用できているチャンネルが 1、雑音が多いためマスクされているチャンネルが 0 で表現される。よって、この値が 1 のチャンネルは BCID_Mask（PPASIC の register）が 1、この値が 0、あるいはこの表に存在していないチャンネルは BCID_Mask が 0 になる。これに加えて、未実装ではあるが、例えば PPASIC での「fine delay」の値などもこの表で管理が可能である。

	属性	名前	変数型	定義域
主キー	A/C サイド	Side	CHAR(1)	
	セクター	Sector	INT	1-24
	FPGA 番号	FPGA	INT	1-29
	PPASIC 番号	PPASIC	INT	1-8
	PPASIC のポート A/B	Port	CHAR(1)	A,B
	ASD の閾値	ASD_threshold	DOUBLE	

図 B.17　表「4_1_2_all_ASD」の項目

	属性	名前	変数型	定義域
主キー	A/C サイド	Side	CHAR(1)	
	セクター	Sector	INT	1-24
	ϕ	phi	INT	0,1
	レイヤー	Layer	INT	1-7
	チェンバー	Chamber_E	INT	1-5
	高電圧の状況	HV	TINYINT(1)	0,1

図 B.18　表「4_1_3_all_chamber」の項目

B.4.2　全 ASD の運用状況

表「4_1_2_all_ASD」は ASD の情報を管理するためのものである。上の表「4_1_1_detector_all_channel」と同じく、表 2 群で ASD を表現するのに使った組み合わせ「FPGA 番号, PPASIC 番号, PPASIC のポート」に「A/C サイド, セクター (1-24)」を掛け合わせることで、個々の ASD を表現している。この表で記録している性質は ASD のスレッショルドの値である。

B.4.3　全チェンバーの運用状況

表「4_1_3_all_chamber」はチェンバーの情報を管理するためのものである。1-7 のレイヤー、「E_」で表現されるチェンバーの番号（low eta 側から何番目のチェンバーか)、更にエンドキャップでは ϕ0 か ϕ1 かの情報を付け加えることによって、1/24 セクター内でチェンバーをユニークに表現する。この 3 属性に加えて「A/C サイド, セクター (1-24)」を掛け合わせることで、検出器全体の中での個々のチェンバーを表現している。この表で管理しているのは高電圧の状況であり、TINYINT(1) の 0/1 で高電圧がかかっているかいないかを表現する。

B.5　EIL4

EIL4 の情報を表 3_5_1　3_5_3 に記述した。これらは Big Wheel における表 1〜3 群に相当するものである。

データベース作成時点（2022/11/18）ではまだ EIL4 の詳細な構造が決定していないので、この部分のデータベースは仮版である。

B.5.1　Coincidence

表「5_1_1_EIL4_coincidence_wire」及び表「5_1_2_EIL4_coincidence_strip」にコインシデンスの input/output（レイヤー毎の検出器チャンネル通し番号/スタッガードチャンネル）対応を記述した。5_1_1 が Wire、5_1_2 が

	項目	名前	変数型	定義域
主キー	スタッガードチャンネル	EW_staggered	INT	1-94
	L1 検出器チャンネルの通し番号	EWL3_detector_serial	INT	1-32
	L2 検出器チャンネルの通し番号	EWL2_detector_serial	INT	1-32
	L3 検出器チャンネルの通し番号	EWL1_detector_serial	INT	1-32
	どのレイヤーの検出器チャンネルが中心か	Center	INT	1/3

図 B.19　表「5_1_1_EIL4_coincidence_wire」の項目

	項目	名前	変数型	定義域
主キー	ASD 名前（例：EW0-1）	ASD_name	VARCHAR(6)	
	ASD チャンネル番号	ASD_channel	INT	0-15
	レイヤー内での検出器チャンネルの通し番号	detector_serial	INT	
	Wire か Strip か	Wire_or_Strip	CHAR(1)	W/S
	どのレイヤーの検出器チャンネルか	Layer	INT	1-3

図 B.20　表「5_2_EIL4_detector_ASD」の項目

Strip を記述したものである。

下の表は表「5_1_1_EIL4_coincidence_wire」に入っている項目をまとめたものである。ここで、この表のスタッガードチャンネルは Big Wheel のスタッガードチャンネルとは独立したものであることに注意する必要がある。Big Wheel 内においては、同じ番号のチャンネルは無限運動量飛跡に対応するものだったが、この EIL4 の表は eta/phi の範囲、granularity において Big Wheel とは別物になっている。

よって、Big Wheel の表「1_1_1_EWM1_detector_channel」のスタッガードチャンネル 1 番と、EIL4 の表「5_1_1_EIL4_coincidence_wire」のスタッガードチャンネル 1 番は対応していない。

B.5.2　Mapping

表「5_2_EIL4_detector_ASD」にマッピングで必要になる情報をまとめた。これは Big Wheel の表 2 群に相当するものである。ASD_name, ASD_channel の組み合わせが主キーとなっており、これは detector_serial, Wire_or_Strip, Layer の組み合わせと一対一の対応になる。レイヤー内での検出器チャンネルの通し番号（detector_seria）は、表 5_1 で LX 検出器チャンネルの通し番号（E_LX_detector_serial）と同じものである。

チェンバーあたりの検出器の数が少ないため、Big Wheel のデータベースとは異なり、全てのレイヤーを一つの表の中にまとめた。このため、表の中に「Layer」の項目が存在している。

EIL4 には 16+16 検出器チャンネルからなる Normal Chamber と、それよりも短い、16+5 チャンネルからなる Special Chamber の 2 種類のチェンバーが存在している。この表では、どちらのチェンバーでも同じ ASD/ASD チャンネルが同じ η 領域を見ていると仮定して、共通した通し番号（detector_serial）を振っている。

B.5.3　Cabling

表「5_3_EIL4_Cabling」にケーブリングの情報をまとめた。これは Big Wheel の表 3_1 に対応するものである。

EIL4 は 24 回繰り返し構造ではないので、Big Wheel とは異なり、Wheel 全体の情報を一枚にまとめた。このた

	項目	名前	変数型	定義域
主キー	セクター	Sector	INT	0-15
	PPASIC 番号	PPASIC	INT	1-8
	PPASIC のポート A/B	Port	CHAR(1)	A/B
	ϕ	phi	INT	0/1/2
	ASD 名	ASD_name	VARCHAR(5)	

図 B.21　表「5_3_EIL4_Cabling」の項目

	属性	名前	変数型	定義域
主キー	データフォーマット α=1, β=2, γ=3	Data_format	INT	1-3
	PPASIC 番号	PPASIC	INT	1-8
	リンク	Link	INT	1,2
	ワード	Word	INT	1-4

図 B.22　表「6_1_FPGA_dataformat」の項目

	属性	名前	変数型	定義域
主キー	ビット番号（0~31）	bit_number	INT	0-31
	PPASIC のポート A/B	Port	CHAR(1)	A,B
	ASD チャンネル番号	ASD_channel	INT	0-15

図 B.23　表「6_2_ASDpin_and_bit_number」の項目

め、主キーにセクター番号が入っている。

この表はこのリンクのエクセル表を元に作成した。

B.6　データフォマットの表

表 6 群は PS ボード上の FPGA から Sector Logic へ送られるヒット情報のデータフォーマットを記述したものである。

B.6.1　Word との対応を記述する表

表「6_1_FPGA_dataformat」は、PPASIC 単位でデータフォーマット中のどの Link, Word に入れられるかを記述する表である。3 種類のデータフォーマットが存在しており、それぞれで Link, Word へのヒット情報の入れ方が変わるので、PPASIC の数 8 とデータフォーマットの 3 種類の、合計 24 行からなる。どの PS ボード（FPGA）でどのデータフォーマットが使われるかという情報は、表「3_3_GTY_Bank」に記載されている。

B.6.2　Bit position を記述する表

表「6_2_ASDpin_and_bit_number」は bit position を記述する表である。ヒット情報が各ワード内でどの bit position(0-31) に収まるかは、PPASIC のポート（A/B）と ASD のチャンネル番号（0-15）で一意に決まるものであり、データフォーマットに依らず共通である。よって、この表は 16 行で構成されている。

付録 C

検索用のビュー

リレーショナル・データベース内での検索に用いるため、付録 B の表を繋ぎ合わせたビューを作成した。

ビューとは検索の条件を記録しておき、それを表のように扱える機能である。例えば「とある表のうち、Phi0 のコンポーネントのみを抜き出した検索」に名前をつけて記録しておくことができる。ビューは表そのものではなく、「検索の仕方」を記録しているだけのものであるため、構成要素となる表に変更を加えると、ビューも変更される。

この研究で開発したデータベースでは、表 1〜3 群を繋ぎ合わせた検索条件をビューとして記録した。これらの表には同じものを表す列が群の間をまたがって存在しているため、この列の値（または文字列）が一致することを条件として表を繋ぎ合わせることができる。表同士の関係の例を B.2 であり、ビューはこの関係性を元に構築した。表 1 群と表 2 群では「ステーション内コインシデンス入力（=レイヤー内での通し番号)」の列を共通して持っており、表 2 群と表 3 群は「ASD 名」の列を共通して持っている。

表 1 群はコインシデンスの input/output、表 2 群は Wired OR を含めた Mapping、表 3 群は Cabling の情報を保持している。これらを繋ぎ合わせることで、「どの FPGA が故障したら、どのスタッガードチャンネルに影響が出るのか」や「あるチェンバーがカバーしているステーション内コインシデンス入力（＝レイヤー内での通し番号）の領域」などの情報を瞬時に引くことができる。

さらに PS ボードからセクターロジックへの送信に使用されるデータフォーマットを表す表を付け加えることで、無限運動量飛跡をのテストパターンを生成するために必要な情報を得ることができる。

付録 D

intra station coincidence のロジック

intra station coincidence は AND や OR、NOT などで記述される論理回路によって記述される。チェンバーの構造の違いから、

- ストリップのダブレット
- ワイヤーのトリプレット
- ワイヤーのダブレット

の 3 種類に大別できる。さらに、例えばダブレットでは 2/2 コインシデンスと 1/2 コインシデンスが独立に計算されるため、論理回路は合計で 2(Strip Doublet)+3(Wire Triplet)+2(Wire Doublet)=7 種類存在する。ただし、ワイヤーとストリップの 2/2 コインシデンスの論理回路は同じ式で記述されるため、式の形は 6 種類である。以下ではこの 7 つの式について記述する。

D.1　ストリップのダブレット

ストリップは全ステーションがダブレットなので、出力されるのは 2/2 コインシデンスと 1/2 コインシデンスの 2 種類である。ストリップのチャンネルの配置は Big Wheel の全てのステーションの全てのチェンバーで等しいため、全て同じ関数の繰り返しで処理することができる。

2/2 コインシデンス

2/2 コインシデンスは同じ代表点をカバーする検出器のチャンネル二つの両方にヒットがあった時に出力する。N 番目の代表点に対しての配置から、図 D.1 のように代表点の周辺にあるチャンネルを $\alpha 1$、$\alpha 2$、$\beta 1$、$\beta 2$ と定義する (代表点に対するチャンネルのずれ方で α レイヤーと β レイヤーを定義したので、代表点が 1 つずれるごとに α レイヤーに相当するレイヤーが A レイヤー、B レイヤーで交互に入れ替わる)。この時、代表点 N の 2/2 コインシデンスの出力 $O_N^{2/2}$ は次の式で書き表すことができる。

$$O_N^{2/2} = \alpha 1 \times \beta 2 \tag{D.1}$$

α レイヤー		$\alpha 1$		$\alpha 2$	
β レイヤー		$\beta 1$		$\beta 2$	
2/2コインシデンス	N-2	N-1	N	N+1	N+2

図 D.1　ストリップの 2/2 コインシデンス。橙のチャンネルにヒットがあった時、代表点 N が発火する

図 D.2　ストリップの 1/2 コインシデンス。橙のチャンネルにヒットがあり、黒のチャンネルにヒットがなかったとき、代表点 N が発火する。4 つの図が 4 つの項に対応しており、最終的にこれらの OR を出力する。

1/2 コインシデンス

1/2 コインシデンスは、片方のレイヤーのみに出力があった場合に出力される。ただし、図 D.1 のように 2 点のヒットによって 2/2 コインシデンスが成立しているときは、代表点 N-1、N、N+1 のどれからも 1/2 コインシデンスを出力しない。この仕組みは後段へ送る代表点の数を増やしすぎないための処理であり、この処理をデクラスタリングと呼ぶ。後段の Segment Reconstruction では intra station coincidence から出力された各ステーションの代表点の組み合わせを使って LUT から $\Delta\phi$ を得るが、トリガーは限られた時間で行わなければならないため、1 度のイベントに対し、LUT の使用回数に制限がある。デクラスタリングはこの制限のために必要な処理である。

1/2 コインシデンスの出力は次の式で書き表すことができる。シミュレータ内では、この式をインライン関数として実装した。

$$O_N^{1/2} = \alpha 1 \times \overline{\beta 2} \times (\overline{\beta 1} + \alpha 2) + \beta 2 \times \overline{\alpha 1} \times (\overline{\alpha 2} + \beta 1) \tag{D.2}$$

このデクラスタリングのために、1/2 コインシデンスの出力を表す式は、AND のみで表せた 2/2 コインシデンスの式よりも複雑になっている。この式を展開すると 4 つの項の和として表すことができる。図 D.2 は、この 4 つの項をそれぞれ図として書き表したものであり、この 4 つのどれかの条件が満たされた場合、1/2 コインシデンスが出力される。

2/2 コインシデンス及び 1/2 コインシデンスの出力の例

上記のように、ファームウェア及びシミュレータでは、2/2 コインシデンスと 1/2 コインシデンスを独立に計算し、出力する。ここでは具体的に、典型的なヒットパターンを例を挙げてコインシデンスの出力を解説する。

図 D.3 の場合は、2/2 コインシデンスが成立する代表点 1 つのみを出力する。2/2 コインシデンスの節で記述した通り、デクラスタリングによって、1/2 コインシデンスは出力しない。

図 D.4 のように 1 チャンネルのみにヒットがあった場合、そのチャンネルがカバーする代表点 2 つの 1/2 コインシデンスを出力する。

Aレイヤー	1	2		3	4
Bレイヤー	1		2		3
2/2コインシデンス	0	1	2	**3**	4	5
1/2コインシデンス	0	1	2	3	4	5

図 D.3　ストリップのコインシデンスの例 1。黄色で示されている、2/2 コインシデンスの 3 番目の代表点のみが発火する。

Aレイヤー	1	2		3	4
Bレイヤー	1		2		3
2/2コインシデンス	0	1	2	3	4	5
1/2コインシデンス	0	1	2	**3**	**4**	5

図 D.4　ストリップのコインシデンスの例 2。1 チャンネルのみにヒットがあった場合、そのチャンネルがカバーする 2 つの代表点である、3 番と 4 番の 1/2 コインシデンスが発火する。

Aレイヤー	1	2		3	4
Bレイヤー	1		2		3
2/2コインシデンス	0	1	2	**3**	**4**	5
1/2コインシデンス	0	1	2	3	4	5

図 D.5　ストリップのコインシデンスの例 3。黄色で示されている、2/2 コインシデンスの 3 番目と 4 番目の代表点が発火する。

Aレイヤー	1	2		3	4
Bレイヤー	1		2		3
2/2コインシデンス	0	1	2	**3**	4	5
1/2コインシデンス	0	1	2	3	**4**	**5**

図 D.6　ストリップのコインシデンスの例 4。黄色で示されている、2/2 コインシデンスの 3 番目の代表点と、1/2 コインシデンスの 4 番目と 5 番目の代表点が発火する。

図 D.5 は図 D.3 に、さらに 1 チャンネルのヒットを追加した場合の出力である。この場合は 2/2 コインシデンスを満たす二つの代表点両方を出力する。

図 D.6 は図 D.3 に、図 D.5 とは違う場所にヒットを 1 つ追加した場合の出力である。図 D.3、図 D.5 では 1/2 コインシデンスは出力されていなかったが、この場合は 1/2 コインシデンスの 5 番目の代表点が図 D.2 の 2 番目の条件を満たすために発火する。さらに、1/2 コインシデンスの 6 番目の代表点も、図 D.2 の 3 番目の条件を満たすために発火する。よって、最終的に 2/2 コインシデンスの代表点が 2 点と、1/2 コインシデンスの代表点が 2 点が出力される。ここで重要なのは、2/2 コインシデンスと 1/2 コインシデンスで共に代表点が出力されることはなく、余分なチャンネルが増えても正しくデクラスタリングされている点である。

D.2　ワイヤーのトリプレット

M1 のワイヤーのみ 3 つのレイヤーからなるトリプレットの構造を有しているため、出力は 3/3 コインシデンス、2/3 コインシデンス、1/3 コインシデンスの 3 種類であ理り、代表点の η 幅は検出器のチャンネルの η 幅の 1/3 となる。

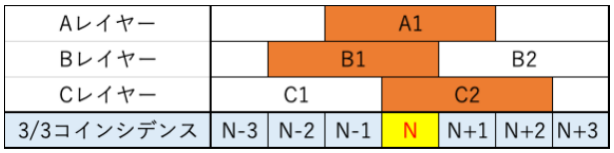

図 D.7　ワイヤーの 3/3 コインシデンス。図中の 3 つの橙のチャンネルにヒットがあった時、代表点 N が発火する

3/3 コインシデンス

3/3 コインシデンスは同じ代表点をカバーする検出器のチャンネル 3 つ全てにヒットがあった時に出力する。図 D.1 のように代表点の周辺にあるチャンネルを A1、B1、B2、C1、C2 と定義すると、N 番目の代表点の 3/3 コインシデンスの出力 $O_N^{3/3}$ は次の式で書き表すことができる。

$$O_N^{3/3} = A1 \times B1 \times C2 \tag{D.3}$$

ストリップのコインシデンスと同様に、レイヤー 1、2、3 のどれがレイヤー A、レイヤー B、レイヤー C に相当するかは代表点によって異なり、代表点が 1 つずれるにつれてレイヤーの役回りが入れ替わる。

2/3 コインシデンス

3 レイヤー中の 2 レイヤーにヒットがあった場合、2/3 コインシデンスを出力する。この時、ストリップの 1/2 と同様にデクラスタリングも考慮する。よって、2/3 コインシデンスを出力するのは以下の 3 つの条件を満たしているときである。

- その代表点を構成する 3 チャンネル中の 2 チャンネルにヒットがある
- 同じ代表点で 3/3 コインシデンスが成立していない
- 隣接する代表点で 3/3 コインシデンスが成立していない

このデクラスタリングは、図 D.8 の形で実現している。論理式は以下の形で書き表せる。

$$O_N^{2/3} = A1 \times B1 \times \overline{C1} \times \overline{C2} \tag{D.4}$$
$$A1 \times \overline{B1} \times \overline{B2} \times C2 \tag{D.5}$$
$$\overline{A1} \times B1 \times C2 \tag{D.6}$$

1/3 コインシデンス

1/3 コインシデンスを出す条件は、2/3 コインシデンスと同様に、以下のように書き表せる。

- その代表点を構成する 3 チャンネル中の 1 チャンネルにヒットがある
- 同じ代表点で 3/3 コインシデンス及び 2/3 コインシデンスが成立していない
- 隣接する代表点で 3/3 コインシデンス及び 2/3 コインシデンスが成立していない

この条件を図示したものが D.9 であり、論理式は一項のみとなる。

$$O_N^{1/3} = A1 \times \overline{B1} \times \overline{B2} \times \overline{C1} \times \overline{C2} \tag{D.7}$$

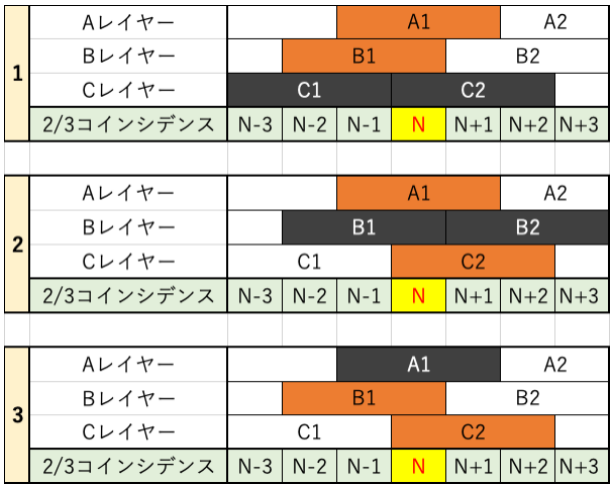

図 D.8　ワイヤーの 2/3 コインシデンス。橙のチャンネルにヒットがあり、黒のチャンネルにヒットがなかったとき、代表点 N が発火する。3 つの図が 3 つの項に対応しており、最終的にこれらの OR を出力する。1 番上の条件は A レイヤーと B レイヤーの 2 つのレイヤーにヒットがあるパターンを処理するためのものであるが、$A1 \times B1$ に対して、$\overline{C1}$ を乗算することで代表点 N での 3/3 コインシデンスがあるパターンを除外し、さらに $\overline{C2}$ を乗算することで代表点 (N+1) での 3/3 コインシデンスがあるパターンを除外している。

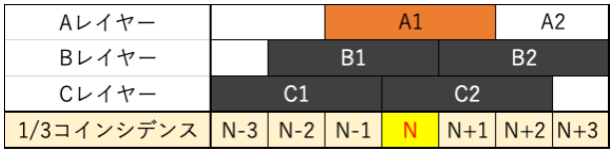

図 D.9　ワイヤーの 1/3 コインシデンス。橙のチャンネルにヒットがあり、黒のチャンネルにヒットがなかった時、代表点 N が発火する

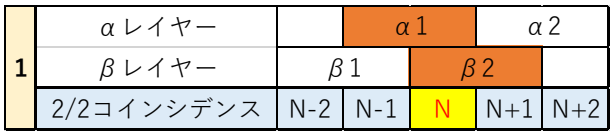

図 D.10　ワイヤーの 2/2 コインシデンス。橙のチャンネルにヒットがあった時、代表点 N が発火する

D.3　ワイヤーのダブレット

M2・M3 ステーションは、ワイヤーもストリップと同じくレイヤー 2 層によるダブレット構造であるので、出力は 2/2 コインシデンスと 1/2 コインシデンスの 2 種類である。

2/2 コインシデンス

ワイヤーの 2/2 コインシデンスは、ストリップの 2/2 コインシデンスと全く同じ条件である。2/2 コインシデンスの条件を図示した図 D.10 は以下の論理式になる。

$$O_N^{2/2} = \alpha 1 \times \beta 2 \tag{D.8}$$

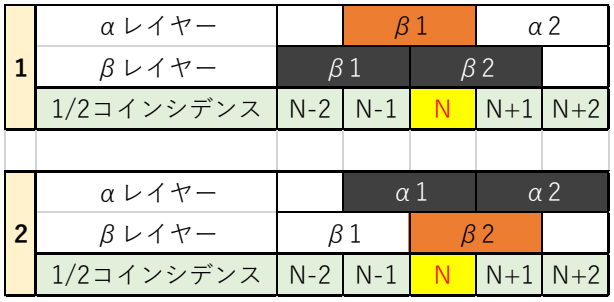

図 D.11　ワイヤーの 1/2 コインシデンス。橙のチャンネルにヒットがあり、黒のチャンネルにヒットがなかったとき、代表点 N が発火する。

レイヤー5	1	2		3		4
レイヤー4	1		2		3	
2/2コインシデンス	602	601	600	599	598	597
1/2コインシデンス	602	601	600	599	598	597

図 D.12　ワイヤーのコインシデンスの例 4。黄色で示されている、2/2 コインシデンスの 599 番目の代表点と、1/2 コインシデンスの 597 番目の代表点が発火する。この図は M2 の Endcap 部分のビーム軸に近い側を抜粋したものである

1/2 コインシデンス

ワイヤーの 1/2 コインシデンスは図 D.11 で表せるように、ストリップの 1/2 コインシデンスよりも項が少ない。この条件の違いは、図 D.2 における第 2、第 4 条件の欠落である。この条件を式で表したものが以下である。

$$O_N^{1/2} = \alpha 1 \times \overline{\beta 1} \times \overline{\beta 2} + \beta 2 \times \overline{\alpha 1} \times \overline{\alpha 2} \tag{D.9}$$

この条件の違いがコインシデンスの出力に現れるのは、図 D.12 のパターンが入力される場合である。これはストリップの例で挙げた図 D.6 と同じパターンであるが、コインシデンス条件の違いにより、598 番目の代表点で 1/2 コインシデンスが出力されない。

D.4　今後におけるコインシデンス論理式の変更の可能性

上記の議論は論文執筆時点のものである。セクターロジックのトリガー系は開発段階にあるため、変更する可能性が十分にある。むしろワイヤーとストリップの対称性を高めるためにダブレットの 1/2 コインシデンスの式をどちらかに統一する可能性が高い[*1]。この研究で開発したシミュレータは、このようなコインシデンス条件の違いによる efficiency 変化を見るのに役立つものであるため、今後のシミュレータの出力結果をもとに、このコインシデンスロジック部分に携わる共同研究者とロジックについて検討していく予定である。

[*1] intra station coincidence 及び Segment Reconstruction のファームウェアは、ワイヤーとストリップは別々の研究者によって開発されたものである。この経緯により、ファームウェアの構造において、ワイヤーとストリップで対称性が良くない部分が複数存在している。

付録 E

Segment Reconstruction のロジックの詳細

Segment Reconstruction はセクターロジックトリガー系のモジュールであり、サブセクター毎に、ワイヤーとストリップで独立に用意される。サブセクターは unit という単位で分割され、unit はさらに subunit という細かい構造に分割される (図 5.6、5.7)。subunit ごとに LUT が用意され、M1・M2・M3 ステーションの代表点の組み合わせを入力とし、$\Delta\theta$ または $\Delta\phi$ の情報を得る。

ファームウェアでは URAM によって LUT を実装しており、M1・M2・M3 ステーションの代表点の組み合わせが LUT のアドレスとなる。

ファームウェア

1 segment の bit 幅がワイヤーでは 18 bit、ストリップでは 9 bit であるのに対し、LUT のデータを入れている UltraRam の出力データ長が 72 bit で固定されているため、URAM の一つのアドレスに複数の segmennt の情報を入れている。ファームウェアのトリガーシステムでは、ワイヤーとストリップの両方で、代表点を表す全ての bit をアドレス生成には使用せず、bit の足を潰している。URAM からデータを取り出したのちに、潰した足の情報を使用して segment を選ぶ。

ストリップでは、一つのアドレスに 8 つの segment が入っている。潰した bit の足を「Local ID」と呼び、URAM データ内の segment の関係は、次のように書き表せる。

$$\mathrm{Pattern[M1LocalID][M2LocalID]} = \{[3][1], [2][1], [1][1], [0][1], [3][0], [2][0], [1][0], [0][0]\} \tag{E.1}$$

ワイヤーでは、潰す足は M1 の下位 2bit であり、こちらでは URAM データの最上位ビット側から順に、0b00∼0b11 と対応する。

シミュレータ

ビットワイズシミュレータではイベントの計算を行う前のセットアップの段階で LUT 用のテキストファイルを読み込み、map を作成する。イベントの計算では LUT としてこの map を使用する。ファームウェアでは URAM のデータ幅が決まっているため 1 つのアドレスに複数の segment を入れているが、シミュレータにはこの制約がないため、map の 1 つのデータにつき 1 つの segment になるように分割を行う。よって、map の key は LUT のアドレスに「前から何番目のデータか」を示す接頭辞を付け加えたものとなる。

map は配列化しており、ワイヤーでは unit、subunit による 2 次元配列[*1] ストリップではチェンバー、unit、subunit による 3 次元配列[*2] である。この構造により、トリガー系では該当する subunit の map のみの検索を行っているため、LUT 全てを 1 つの map に収める場合よりも処理時間が短縮が見込まれる。

[*1] データ型は std::vector<std::vector<std::map<int, uint32_t> > >

[*2] データ型は std::vector<std::vector<std::vector<std::map<int, uint32_t> > > >

表 E.1　Strip Segment Reconstruction におけるヒットのあったレイヤー枚数によるアドレス生成の優先順位（ATLAS Collaboration 2019）

coincidence pattern	Hit pattern		
	M1	M2	M3
6/6	2/2	2/2	2/2
5/6 A	2/2	1/2	2/2
5/6 B	1/2	2/2	2/2
5/6 C	2/2	2/2	1/2
4/6 A	1/2	1/2	2/2
4/6 B	2/2	1/2	1/2
4/6 C	1/2	2/2	1/2

ファームウェアでは全ての unit、subunit が並列計算されるが、C++ では直列的な構造となっている。unit 内の全てのステーションに代表点が存在していなければアドレスを生成できないことから、計算時間の無駄を省くために、unit の計算の最初に M3 ステーションに代表点が存在しているかを確認しており、M3 に代表点がなければその unit の計算をスキップするようにしている。

以下ではシミュレーション内での代表点の組み合わせから key の作成、最終的な出力の選別までについて記述する。

E.1　ストリップ

ストリップは 1 つの unit を 2 つの subunit に分割する。

トリガーシステムでは計算に使用できる時間が決まっているため、LUT から引き出せる情報の数には制限がある。ストリップでは subunit につき最大 6 つのアドレスの生成を行う。代表点の数によってはアドレスの候補が 6 つ以上となるが、ファームウェア及びシミュレータでは、その中で優先順位の高いものからアドレスの生成を行う。

最も重視される優先順位はレイヤー数であり、例えば 1/3 コインシデンスよりも 3/3 コインシデンスによる代表点を使用した組み合わせの方が優先される (表 E.1)。次に重視される優先順位は代表点の位置であり、同じ枚数のコインシデンスによる代表点が複数ある場合は、より subunit の中心に近い代表点を優先してアドレス生成に使用する。

一つの subunit は、M1 の代表点 32 チャンネル (=5 bit)、M2 の代表点 16 チャンネル (=4 bit)、M3 の代表点 8 チャンネル (=3 bit) からなる。各代表点の上位 3 bit を global ID と呼び、global ID に含まれない下位ビットを Local ID と呼ぶ。global ID を合わせた 9 bit = {M1 global ID (3 bit)、M2 global ID (3 bit)、M3 global ID (3 bit)} がアドレスとなる。ここで、図 E.1 左側で代表点のグループに振られている番号はアドレス生成における優先順位 (0 が最も優先順位が高い) であり、global ID そのものではないことに注意せよ。global ID はスタッガードチャンネル番号と同じ昇降順で定義される。

シミュレータではアドレスとなる 9 bit の最上位ビット側に M1 下位 2 bit, M2 下位 1 bit の計 3 bit を付け加えたものを作成し、これを map への入力とする。ミューオンの飛跡としてあり得ないパターンなど、トリガーする必要がない代表点の組み合わせによるアドレスにはデータが入っていない。よって、map から取り出せたデータ $\Delta\phi$ の中から、最もレイヤー数が多いものを選択し、unit につき最大 1 つの segment を後段の Wire-Strip coincidence に出力する。レイヤー数が同じものが複数あった場合は更に $\Delta\phi$ が最も小さいものを選択する。

また、ファームウェアおよびシミュレータにおいて、M3 の位置情報（0 ～ 63 の 5 bit）は、どのセクターでも ϕ

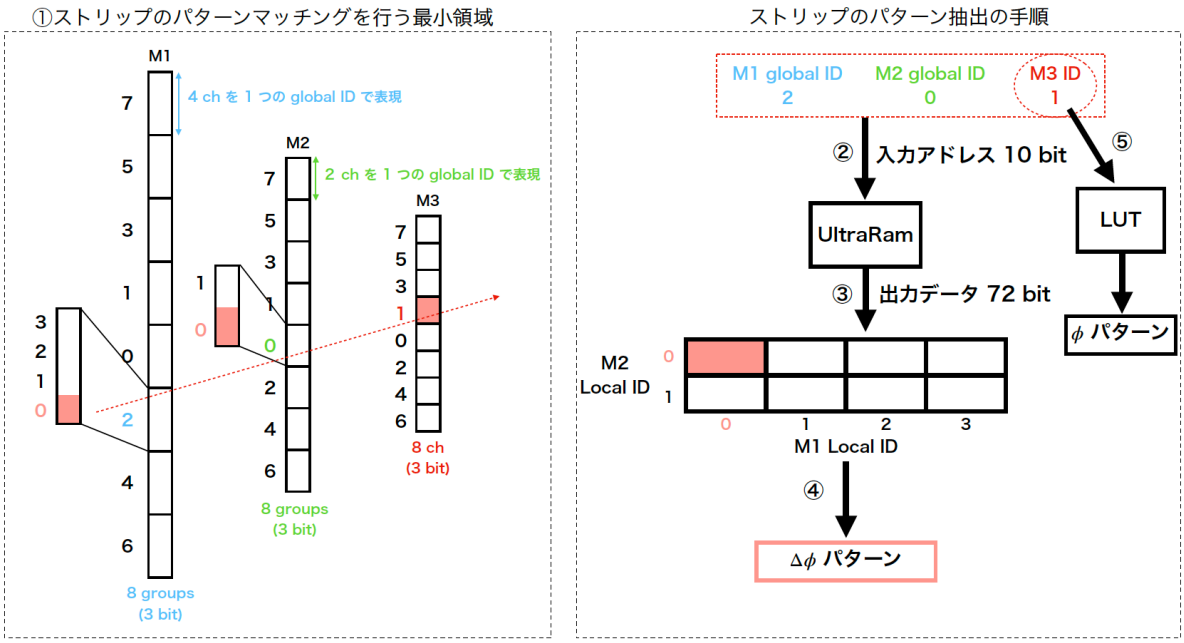

図 E.1　ストリップのパターン抽出の概念図。①M1 の代表点 32 チャンネル、M2 の代表点 16 チャンネル、M3 の代表点 8 チャンネルに対してパターンマッチングを行う。M1 の代表点を 4 チャンネルずつ、M2 の代表点を 2 チャンネルずつのグループに分けて、グループの番号を global ID、グループの中での位置を local ID とする。global ID はパターンマッチングを行う領域の中心から設定し、代表点の組み合わせを求める際に中心のものから使用されるようにする。② M1 global ID (= 3 bit)、M2 global ID (= 3 bit)、M3 ID(= 3 bit) の組み合わせは 9 bit で表現される。Ultra RAM は 2 つの領域のパターンリストを保存しているため、どちらの領域に対応するかを判別するための 1 bit を加えた 10bit を UltraRAM の入力アドレスとする。③ 入力アドレスに対応する 72 bit のデータを UltraRAM から出力する。④ M1、M2 の local ID を用いて 9 bit の $\Delta\phi$ のパターンを取り出す。⑤ M3 ID から、LUT を用いて ϕ を出力する。（小林蓮 2021）

の増加に対して順方向になるように定義されている。1 枚のチェンバーの中にあるユニットの番号 0 ～ 3 も同様である。このため、ϕ とスタッガード ID が逆行するファイセクターでは、位置情報がこのモジュール内で逆転し、これ以降のモジュールでは検出器のチャンネルの番号順ではなく座標に準じた並びになる。

E.2　ワイヤー

ワイヤーは 1 つの unit を 4 つの subunit に分割する。ストリップと異なり、subunit を構成するチャンネルは、M1・M2 は unit 内で同じものとなっており、M3 の代表点のみが subunit ごとに異なる。

シミュレータおよびファームウェアでは、まず各ステーションから代表点を取得・データメンバに記録し、記録された代表点を前述の優先順位に従って組み合わせることでアドレスを生成する。ワイヤーではストリップと異なり、データメンバに記録する代表点の数にも制限が存在する。

ピボットとなる M3 ステーションでは、2/2 コインシデンスの代表点と 1/2 コインシデンスの代表点[*3]を、最大で 1 つづつデータメンバに記録する。このとき、スタッガードチャンネル番号が最も大きい代表点が優先される。代表点の番号をデータメンバに保持するときに、別に用意した bool 値のデータメンバによってデータを記録したか否かのフラグ管理を行う。

[*3] subunit 内で一意にチャンネルを定めるため、0 ～ 3 の値を記録する

表 E.2　Wire Segment Reconstruction におけるヒットのあったレイヤー枚数によるアドレス生成の優先順位（ATLAS Collaboration 2021）

coincidence pattern	Hit pattern		
	M1	M2	M3
7/7	3/3	2/2	2/2
6/7 A	2/3	2/2	2/2
6/7 B	3/3	1/2	2/2
6/7 C	3/3	2/2	1/2
5/7 A	2/3	2/2	2/2
5/7 B	2/3	1/2	1/2
5/7 C	3/3	2/2	1/2
5/7 D	1/3	2/2	2/2

M1・M2 ステーションでは、ヒットのあったレイヤー数の数ごとに、unit の中心に近い代表点から最大 2 点ずつ記録する。例えば、M2 ステーションでは、2/2 コインシデンスが最大 2 点、1/2 コインシデンスの代表点が最大 2 点であるため、合計で最大 4 点の代表点をデータメンバに記録することができる。M3 ステーションと同じく、別に用意した bool 値のデータメンバによってデータを記録したか否かのフラグ管理を行う。M1・M2 ステーションの領域は、同一の unit 内の全ての subunit で共通しているため、unit につき 1 度のみ代表点取得の計算を行い、subunit ごとのループ処理では M3 の代表点の取得とアドレス生成の関数のみを反復する。

アドレス生成の過程では、ワイヤーもストリップと同様に、飛跡のヒットがあったレイヤー数が多い代表点の組み合わせが優先され (E.2)、レイヤー数が同じ場合は、より unit の中心に近い代表点を使用する。

付録 F

FORWARD/BACKWARD の構造と STRIP における A/B レイヤー表記

Strip のチャンネルのレイヤーを表記するときは、衝突点側から数えて何番目のレイヤーかではなく、ステーションの番号と A/B レイヤーの組み合わせで記述される。ここで、A レイヤーの定義は「Forward チェンバーの L1,4,6（lower-Z）および Backward チェンバーの L3,5,7（higher-Z）」であり、B レイヤーは「Forward チェンバーの L3,5,7（higher-Z）および Backward チェンバーの L1,4,6（lower-Z）」で定義される。以下ではこの表記法を使用する理由について記述する。

TGC のチェンバーには Forward/Backward の 2 種類が存在する。衝突点側からチェンバーを見たときに、右側の側面に Wire チャンネルの ASD ボードがついているのが Forward チェンバーであり、左側の側面に ASD ボードがついているのが Backward チェンバーである。Forward/Backward では Strip のチャンネル番号のつき方が異なっており、Wire の ASD が刺さっている側が小さいチャンネル番号になっている。

図 F.1 のように、チャンネル番号のつき方は Forward/Backward チェンバーで鏡面対称のようになっているが、内部のスタッガード構造はどちらのチェンバーでも同じであり、鏡面対称構造にはなっていない。

中心に衝突点をおいた時の A/C サイドそれぞれ隣接する 2 枚のチェンバーのスタッガー構造とチャンネル番号を図 F.4 に示す。図の中において、最も外側のチャンネルが短くなっているレイヤーは Forward では L5 だが、Backward では L4 である。よって、Strip のチャンネルを扱うときは、同じポジションを観測するチャンネルを同じ名称で取り扱いたいため、レイヤーを番号ではなく A/B レイヤーで記述した方が便利であることがわかる。

ここで、レイヤー番号の表記を A/B レイヤーの形に変換する際、1/24 セクター内では同じレイヤー番号に対し

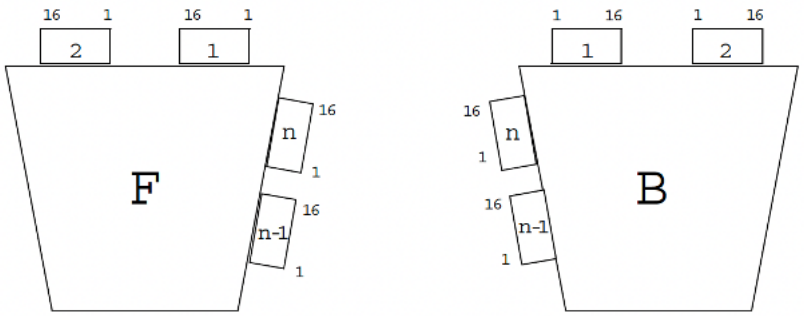

Figure 1: Forward and Backward Units, with their FE-electronics, ASD, boards, as seen from the Interaction Point. Refer also to Figure 5 to see how the 'F' and 'B' chambers are arranged on the wheels.

図 F.1　Forward/Backward チェンバーの模式図（ATLAS Collaboration 2005）

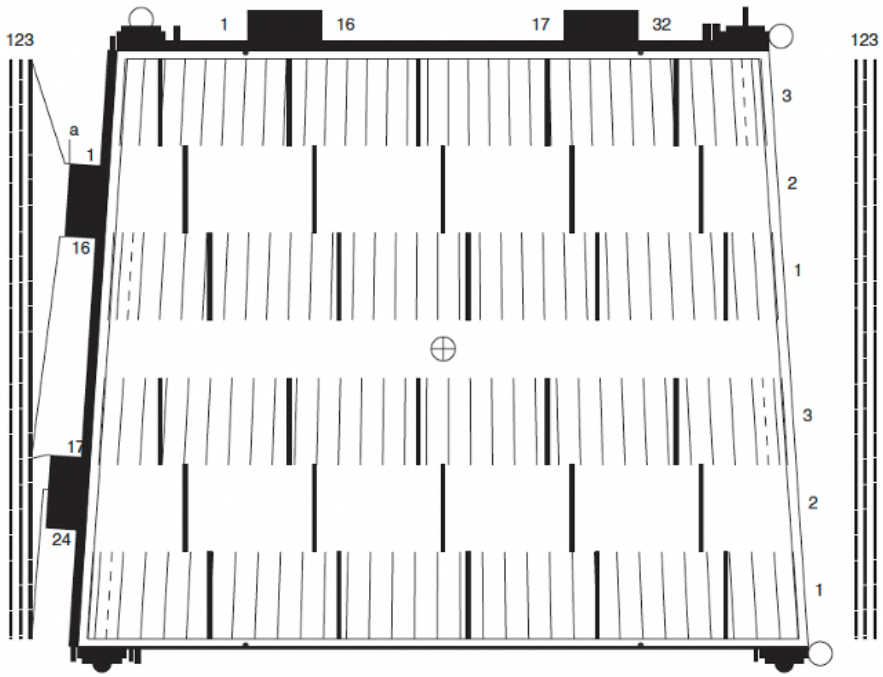

Figure 2: Tomograph of a Type 8 Forward (as in forward-backward) tripletunit (U08F11). For Side A, f increases from right to left; for side C from left to right.

図 F.2　Big Wheel の Forward チェンバーの Strip チャンネルの配置（ATLAS Collaboration 2005）

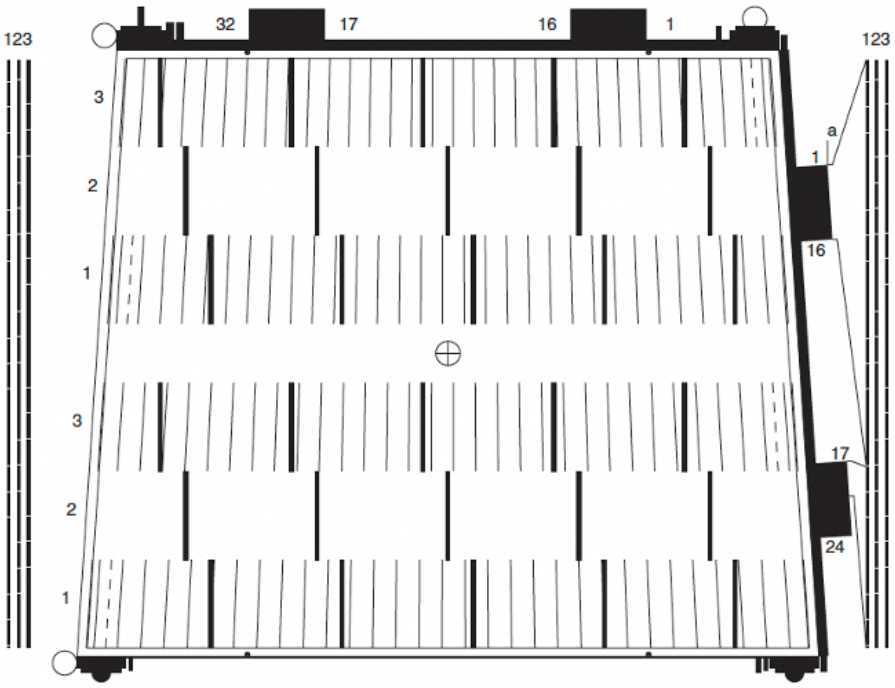

Figure 3: Tomograph of a Type 8 Backward (as in forward-backward) tripletunit (U08B11). For Side A, f increases from right to left; for side C from left to right.

図 F.3　Big Wheel の Backward チェンバーの Strip チャンネルの配置（ATLAS Collaboration 2005）

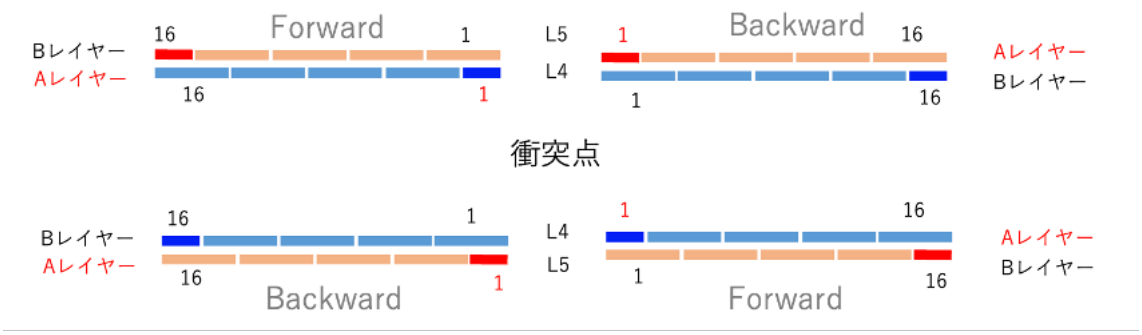

図 F.4　中心に衝突点をおいた時のチャンネルのスタッガー構造 (M2 ステーション)

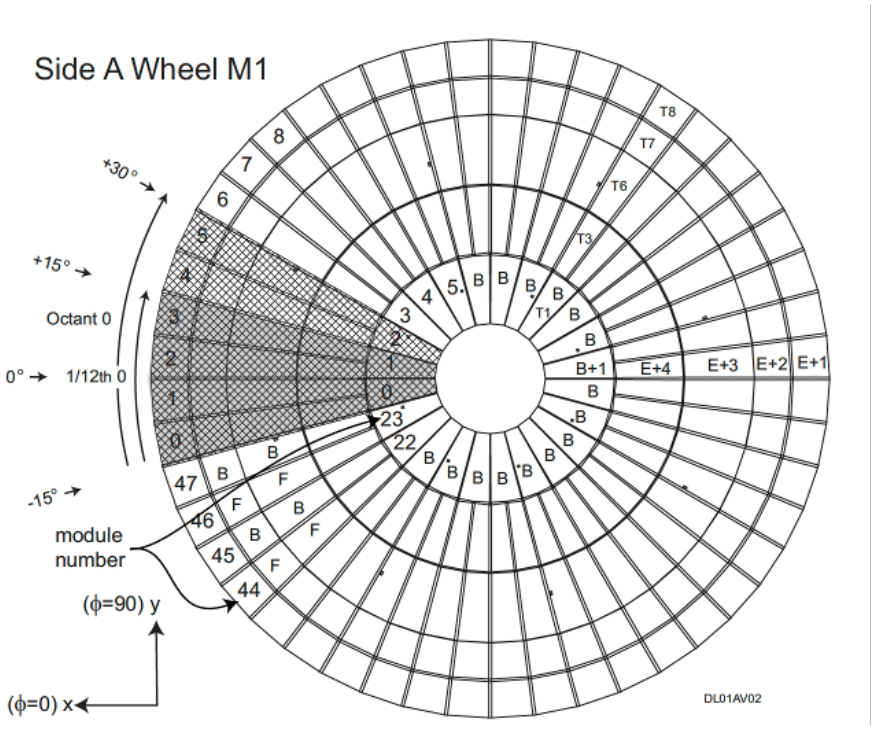

図 F.5　Side A Wheel M1 の Forward /Backward チェンバーの配置状況（ATLAS Collaboration 2005）

て、Phi0, Phi1 で A/B レイヤーが交換していることに注意する必要がある。Forward 領域においては、一つのサイド内で Forward/Backward チェンバーはどちらか一方しか存在しない（Side A は Backward、Side C は Forward のみ）が、Endcap 領域では Forward/Backward チェンバーは交互に並んで存在している。この並び方は、1/24 セクターで見た時、Endcap 領域の Phi0、Phi1 の 2 列のチェンバーの中心側に Wire チャンネルの ASD が刺さる形である。このため、1/24 セクター内では同じレイヤー番号に対して、Phi0、Phi1 で A/B レイヤーが異なる。

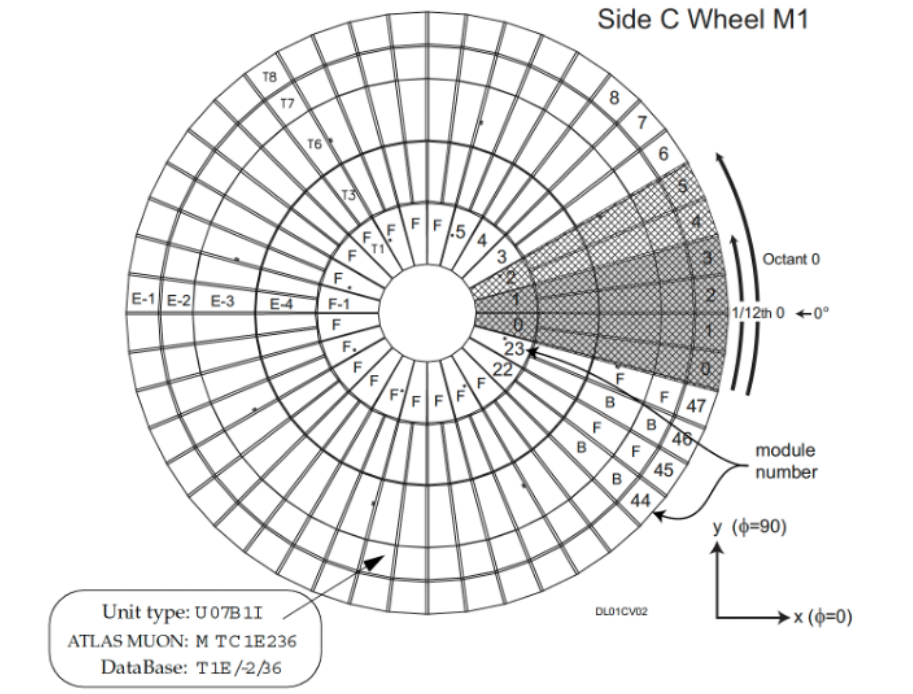

図 F.6　Side C Wheel M1 の Forward /Bacalward チェンバーの配置状況（ATLAS Collaboration 2005）

引用文献

[1] ATLAS Collaboration“PS Board Schematics,” https://twiki.cern.ch/twiki/bin/viewauth/Atlas/TgcDocument#:~:text=PS-,Board,-Schematics (ATLAS Internal webpage).

[2] ——— 1998 “ATLAS level-1 trigger : Technical Design Report,” https://cds.cern.ch/record/381429.

[3] ——— 1999 “Amplifier-Shaper-Discriminator ICs and ASD Boards,” https://twiki.cern.ch/twiki/pub/Atlas/TgcDocument/ASD-PRR_v19991001.pdf (ATLAS Internal webpage).

[4] ——— 2005 “Naming and numbering schemefor the Endcap muon trigger system,” https://twiki.cern.ch/twiki/pub/Atlas/TgcDocument/numbering.pdf.

[5] ——— 2012 “Letter of Intent for the Phase-II Upgrade of the ATLAS Experiment,” https://cds.cern.ch/record/1502664.

[6] ——— 2019 “TGC Patch-Panel ASIC Design Report for Production Readiness Review,” https://twiki.cern.ch/twiki/pub/Atlas/Ph2TgcPsb/PP-ASIC_PRR_v2.pdf (ATLAS Internal webpage).

[7] ——— 2021 “Endcap Sector Logic: PBS/WBS 1.1.5.1 (Preliminary Design Report),” https://twiki.cern.ch/twiki/pub/Atlas/Ph2TgcSectorLogic/EndcapSLPDR.Dec2021.pdf (ATLAS Internal webpage).

[8] CERN 2022 “The HL-LHC project,” https://hilumilhc.web.cern.ch/content/hl-lhc-project.

[9] MySQL 2022 “MySQL homepage,” https://www.mysql.com/jp/.

[10] The ATLAS Collaboration et al. 2008 “The ATLAS Experimet at the CERN Large Hadron Colider,” *JINST* **3**, No. S08003, 199–201, URL: https://iopscience.iop.org/article/10.1088/1748-0221/3/08/S08003/meta.

[11] XILINX 2022 “Vivado Design Suite,” https://japan.xilinx.com/products/design-tools/vivado.html.

[12] 三島章熙・青木匠・石野雅也他「高輝度 LHC-ATLAS 実験のミューオントリガー開発 : I/O・読み出し・トリガー回路の 大規模 FPGA への実装と実機動作試験」, , 2022 年度日本物理学会秋季大会 7pA422-10.

[13] 三野裕哉 2020「高輝度 LHC ATLAS 実験に向けた初段ミューオントリガーアルゴリズムの開発およびハードウェアへの実装」, 修士論文, URL : https://www-he.scphys.kyoto-u.ac.jp/theses/master/mino_mt.pdf.

[14] 吉川正俊 2019 『データベースの基礎』, 株式会社オーム社.

[15] 吉村宣倖 2022 「LHC-ATLAS 実験 Run-3 に向けた初段ミューオントリガーアルゴリズムの開発および性能評価」, 修士論文, URL : https://www-he.scphys.kyoto-u.ac.jp/theses/master/yoshimura_mt.pdf.

[16] 大町千尋 2006「ATLAS 実験におけるシミュレーションを用いたエンドキャップトリガーの性能評価」, 修

士論文，URL：https://ppwww.phys.sci.kobe-u.ac.jp/seminar/pdf/omachi-mron.pdf.

[17] 小林蓮 2021 「高輝度 LHC ATLAS 実験に向けた初段ミューオントリガーアルゴリズムの改良とハードウェアへの実装」，修士論文，URL：https://www-he.scphys.kyoto-u.ac.jp/theses/master/kobayashi_mt.pdf.

[18] 山下恵理香 2022 「Git Lab: TGC Channel Mapping(ATLAS Internal webpage)」，，URL：https://gitlab.cern.ch/eyamashi/tgc_channel_mapping.

[19] 木村誠聡 2012 『ハードウェア記述言語によるディジタル回路設計の基礎-VHDL による回路設計-』，株式会社数理工学社.

[20] 杉崎海斗 2021「LHC-ATLAS 実験 Run 3 の開始に向けたミューオントリガー回路系の高速読み出しと統合制御の実現」，修士論文，URL：https://www.icepp.s.u-tokyo.ac.jp/download/master/m2020_sugizaki.pdf.

[21] 河本地弘・隅田土詞・三野裕哉他「高輝度 LHC-ATLAS 実験のミューオントリガー開発: 運動量再構成回路の大規模化」，，2022 年度日本物理学会秋季大会 7pA422-9.

[22] 西沢夢路 2017 『基礎からの MySQL 第 3 版』，SB クリエイティブ株式会社.

[23] 赤塚駿一 2017 「LHC-ATLAS 実験 Run-3 に向けたミューオントリガーの改良」，修士論文，URL：https://www-he.scphys.kyoto-u.ac.jp/theses/master/akatsuka_mt.pdf.

[24] 青木匠 2022 「大規模エレクトロニクスシステムにおける次世代型オペレーションモデルの研究- 自律型ステートマシーンの導入及び SoC による遠隔型制御の実現-」，修士論文，URL：https://www.icepp.s.u-tokyo.ac.jp/download/master/m2021_aoki.pdf.