2024 年度 修士論文

高輝度 LHC-ATLAS 実験に向けたミューオントリガー回路の開発 ー動作検証システムの構築とトリガー性能の評価ー

(Development of the muon trigger for the ATLAS experiment at the High-Luminosity LHC - Construction of the validation system and evaluation of the trigger performance -)

東京大学大学院
理学系研究科 物理学専攻
石野研究室

修士課程 2 年
学籍番号 35236094

牧田 藍瑠

2025 年 1 月 27 日

概要

ヒッグス粒子の発見によって標準模型は完成したものの、未だ説明できない現象が多数確認されており、標準模型を超えた新物理の解明が待望される。LHC-ATLAS 実験では素粒子標準模型を超えた新物理の探索のために、世界最高エネルギーである 13.6 TeV での陽子陽子衝突事象において、標準模型の精密測定および新粒子の探索を行っているが、未だ新物理の解明には至っていない。さらなる精密測定の精度向上と新粒子探索領域の拡大のために LHC の高輝度化が予定されており、2029 年から瞬間最高ルミノシティは現行の約 3 倍の $5 - 7.5 \times 10^{34}\ \mathrm{cm}^{-2}\mathrm{s}^{-1}$ に増強される。高輝度化による衝突レートの増大に対応するため、ATLAS 実験の Trigger DAQ システムはアップグレードされ、初段トリガーレートは 1 MHz に、初段トリガーレイテンシーは 10 μs に増強される。このアップグレードに伴って、エンドキャップ部初段ミューオントリガーを担う TGC 検出器のエレクトロニクスも刷新される。

高輝度 LHC-ATLAS における TGC エレクトロニクスでは広帯域の通信技術を用いてヒットデータ全てを大規模 FPGA に集約し、包括的なトリガー演算・10 μs もの長時間の buffer・ヒットデータおよびトリガー中間出力の読み出しを行う。トリガー演算におけるロジックは TGC 検出器の全 7 層のヒットから飛跡を再構成し、横方向の運動量を概算することで構成される。このような大規模かつ精巧なトリガー論理回路の開発において、そのバリデーションは不可欠である。

本研究では、トリガー論理回路の検証システムを構築した。検証システムでは大統計のシミュレーションデータを入力として、トリガー論理回路を実装したファームウェアのシミュレーションとトリガー論理回路を再現したソフトウェアシミュレータの出力を bit レベルで比較し、不具合の精密な特定を進めた。前者は Vivado Simulator というシステムであり、Xilinx が提供する Vivado Simulation というファームウェアシミュレーションツールを用いて、トリガー論理回路に大統計 MC データから作成したテストパターンを入力してその出力を読み出すシステムを実装した。後者は先行研究によって開発された Bitwise Simulator であり、Vivado Simulator と同一の大統計 MC データを入力して、トリガー論理回路の出力を詳細に読み出せる。両者の出力を比較して差異があったイベントを抽出し、差異があったイベントにおいてはトリガーロジック上流から順に内部の中間出力を bit レベルで比較し、差異の原因となった不具合箇所を特定した。結果、全イベント中の差異があるイベントを 1 % 以下にまで低減した。引き続き、この差異を解消するために現在の検証システムを駆使していく。また、Vivado Simulator の開発によって SL 実機試験特有の問題を切り分け、今後のトリガー論理回路を洗練させていくベースラインを築いた。

次に、不具合を修正したトリガー論理回路においてトリガー性能評価を行なった。結果、修正前に見られていたトリガー効率の Inefficiency は多くが改善され、プラトー領域でのトリガー効率は修正前の 81.9 % から 94.6 % に回復した。この値は理想性能である Software Simulator の Efficiency に比肩する。

さらに、トリガー読み出しコンテンツの策定も行なった。現状はデバッグ用の仮実装の状態であるが、トリガー論理回路の検証が完了次第、本番運用のための本実装に移行する。本番運用時にはリソースと帯域幅の制限から必要最低限の読み出しコンテンツのみを選び出す必要がある。トリガー論理回路の重要な箇所を段階的に読み出せるようなコンテンツを抽出し、読み出しフォーマットを策定した。今後は読み出しフォーマットの形式通りにトリガー中間出力を成形する読み出し回路を実装していく。

本研究によって、トリガー論理回路はプラトー領域のトリガー効率 95 % を示し、設計思想通りのトリガー性能を至るという重要なマイルストーンに達した。また、確立した Wire Strip Coincidence までの検証システムは、今後開発される Inner Coincidence や Track Selector を含んだ完全な統合トリガー回路への拡張も視野に入れた形で実装している。そのため、今後のトリガー論理開発の中でも拡張され利用されていくことが期待される。

目次

図目次

表目次

第 1 章

序論

1.1　素粒子標準模型と標準模型を超えた物理

素粒子物理学における多くの実験結果は、標準模型に基づいて矛盾なく説明される。標準模型は、物質を構成するクォークとレプトン、これらの相互作用を媒介するゲージボソン、そしてヒッグス粒子で成り立っている。標準模型のフェルミオンとボソンの性質は、表 1.1 に示されている（質量の値は [1] より引用）。標準模型内のフェルミオンはすべて固有の自由度としてスピン 1/2 を持ち、クォークとレプトンに分類される。$SU(3)$ ゲージ対称性を課して構築される QCD では、クォーク間の強い相互作用をグルーオンが媒介する一方でレプトンはグルーオンと結合しない。また、左巻きのカイラリティを持つクォークとレプトンは $SU(2)$ ゲージ対称性によってウィークボソンを媒介して弱い相互作用で結合し、ダウンタイプとアップタイプで対になっている。加えて、クォークとレプトンはヒッグス場を介したカイラリティ右巻きと左巻きの結合の強さによって 3 世代に分かれている。標準模型のボソンの中で、スピン 1 を持つものには、$SU(3)$ ゲージ対称性に対応する強い相互作用を媒介するグルーオン、$SU(2)$ ゲージ対称性に対応する弱い相互作用を媒介するウィークボソン、そして $U(1)$ ゲージ対称性に対応する電磁相互作用を媒介する光子がある。スピン 0 を持つものはヒッグスであり、ヒッグスは 2 成分複素スカラー場の 4 つの自由度のうち、$SU(2) \times U(1)_Y$ 対称性の自発的破れの後に残る 1 つの自由度に対応する。他の 3 つの自由度は南部・ゴールドストンボソンとなり、その自由度は $\mathrm{W}^{\pm}$ ボソンと Z ボソンの質量となる。さらに、ヒッグス場は湯川型の結合を通じてクォークとレプトンに質量を与える。

標準模型は数多くの実験によって検証されているが、階層性問題や力の大統一、バリオン数生成、暗黒物質など、

表 1.1: 素粒子標準模型

分類		電荷 (e)	スピン ($\hbar$)	世代毎の表記			**質量 (GeV)**
クォーク	アップタイプ	+2/3	1/2	u	c	t	0.0022, 1.27, 173
	ダウンタイプ	-1/3	1/2	d	s	b	0.0047, 0.094, 4.18
レプトン	アップタイプ	0	1/2	ν_e	ν_μ	ν_τ	$< 10^{-9}$ (全て)
	ダウンタイプ	-1	1/2	e	μ	τ	0.000511, 0.106, 1.777
ゲージボソン	フォトン	0	1		γ		0
	グルーオン	0	1		g		0
	W ボソン	± 1	1		$W^{\pm}$		80.37
	Z ボソン	0	1		Z		91.18
スカラーボソン	ヒッグス	0	0		h		125

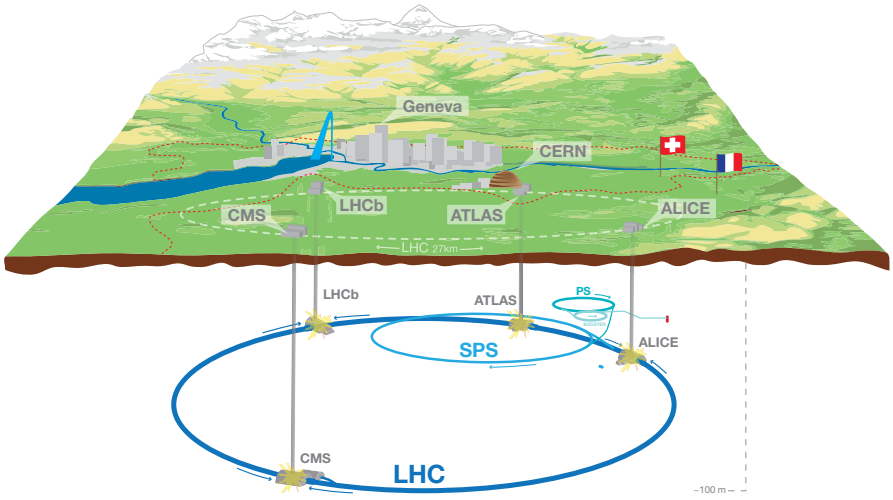

図 1.1: LHC の概観 [2]。スイスとフランスの国境付近地下 100 m に位置する。

標準模型で説明できない観測や理論的に不自然な問題も数多く見つかっている。これらの問題を解決するために、標準模型を超えたモデルが多く提案されている。標準模型を超える新物理モデルの検証のために標準模型の精密測定や新粒子探索は素粒子物理学の重要なテーマであり、これには重い新粒子探索や電弱スケール粒子の大量生成が可能なエネルギーフロンティア実験が重要である。標準模型を超えた物理として代表的なものは超対称性（SUSY）理論である。SUSY によって階層性問題を解決し、ゲージ相互作用の統一が期待されているのに加えて、モデルの中に暗黒物質となり得る粒子が存在する。また、ヒッグスセクターは未だ精密な測定が達成できておらず、標準模型を超えた多くの物理モデルはヒッグスの結合定数が標準模型の予想値から逸脱することを予言している。標準模型を超えた物理の解明のためにヒッグスの精密測定と SUSY 粒子の探索は大きなテーマであり、この章の以降では主にこの二つのテーマに焦点を当てて議論する。

1.2　LHC-ATLAS 実験

1.2.1　LHC

欧州原子核研究機構（CERN）では、世界最高エネルギーであり、世界で唯一のヒッグスファクトリーである粒子衝突型加速器 Large Hadoron Collider（LHC）での物理実験が進行中である。図 1.1 に示すように、LHC は周長 26.7 km のハドロンコライダーであり、衝突エネルギー 13.6 TeV（陽子衝突）、ビームバンチ交差頻度は 40 MHz である。ビームラインに 4 箇所のビーム交差地点が存在し、それぞれで ATLAS 実験、CMS 実験、ALICE 実験、LHCb 実験が行われている。

1.2.2　ATLAS 検出器

ATLAS 検出器は全長 44 m、直径 25 m の円筒形の汎用検出器である。図 1.2 に示すように、ATLAS 検出器は、内側から、内部飛跡検出器、カロリメータ、ミューオンスペクトロメータによって構成されている。内部飛跡検出器はソレノイド磁石によって覆われており、磁場による荷電粒子の曲がり具合から運動量を測定する。カロリメータは電磁カロリメータとハドロンカロリメータからなり、前者は電子と光子、後者はハドロンのエネルギーを測定する。最も外側に位置するのはするのはミューオンスペクトロメータであり、そのすぐ内側に位置するトロイダル磁場によるミューオンの曲がり具合から運動量を測定する。

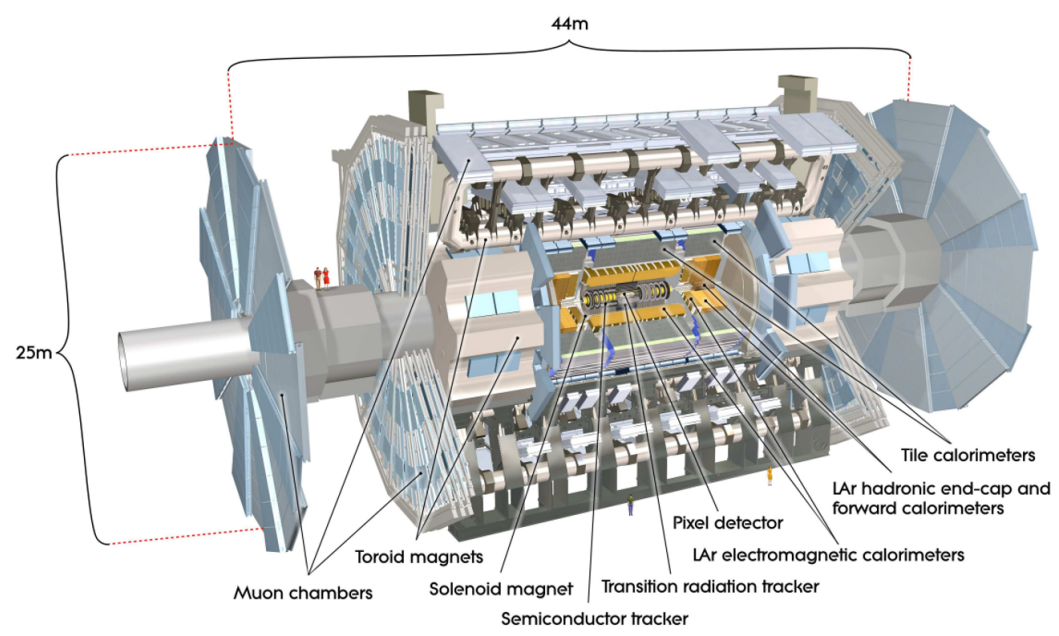

図 1.2: ATLAS 検出器の概観（[3] から引用）。

1.2.3　TDAQ システム

ATLAS 検出器では、40 MHz の陽子バンチ交差が発生し、1 回の陽子バンチ交差あたり $O(1\,\mathrm{MByte})$ ほどのデータ量が生じる。このサイズのデータ量全てを処理して記録するのはリソースの制限から不可能である。加えて、ほとんどの陽子バンチ交差に伴う物理事象は移行運動量がハドロンスケール程度の非弾性散乱であり、目標とする物理現象の測定の観点では価値が小さい。そのため、ATLAS 実験ではオンラインで物理現象の取捨選択を行い、重要な物理事象を選択し記録するトリガーシステムを採用している。トリガーシステムはハードウェアで実装された初段トリガーとソフトウェアで実装された後段トリガーの 2 段階からなる。本研究は初段トリガーにおけるエンドキャップ部[*1]ミューオントリガーの開発に関するものである。

1.2.4　LHC-ATLAS 実験における物理

LHC-ATLAS 実験では SUSY 粒子の探索が行われているが、未だ発見には至っていない。現状の各 SUSY 粒子の質量の棄却領域を図 1.3 に示す。また、ヒッグスセクターの測定に関して、例としてヒッグスの三点自己結合定数 λ の測定を上げる。λ はヒッグスポテンシャルの形状を決定する重要なパラメータである。λ はヒッグスの対生成事象の観測によって測定可能であるが、この断面積は非常に小さく、現状ではその断面積に上限を与えるまでにとどまる。（図 1.4）。いずれの例においても、感度向上が必要なケースであり、感度向上のための積分ルミノシティの蓄積が重要なアプローチである。

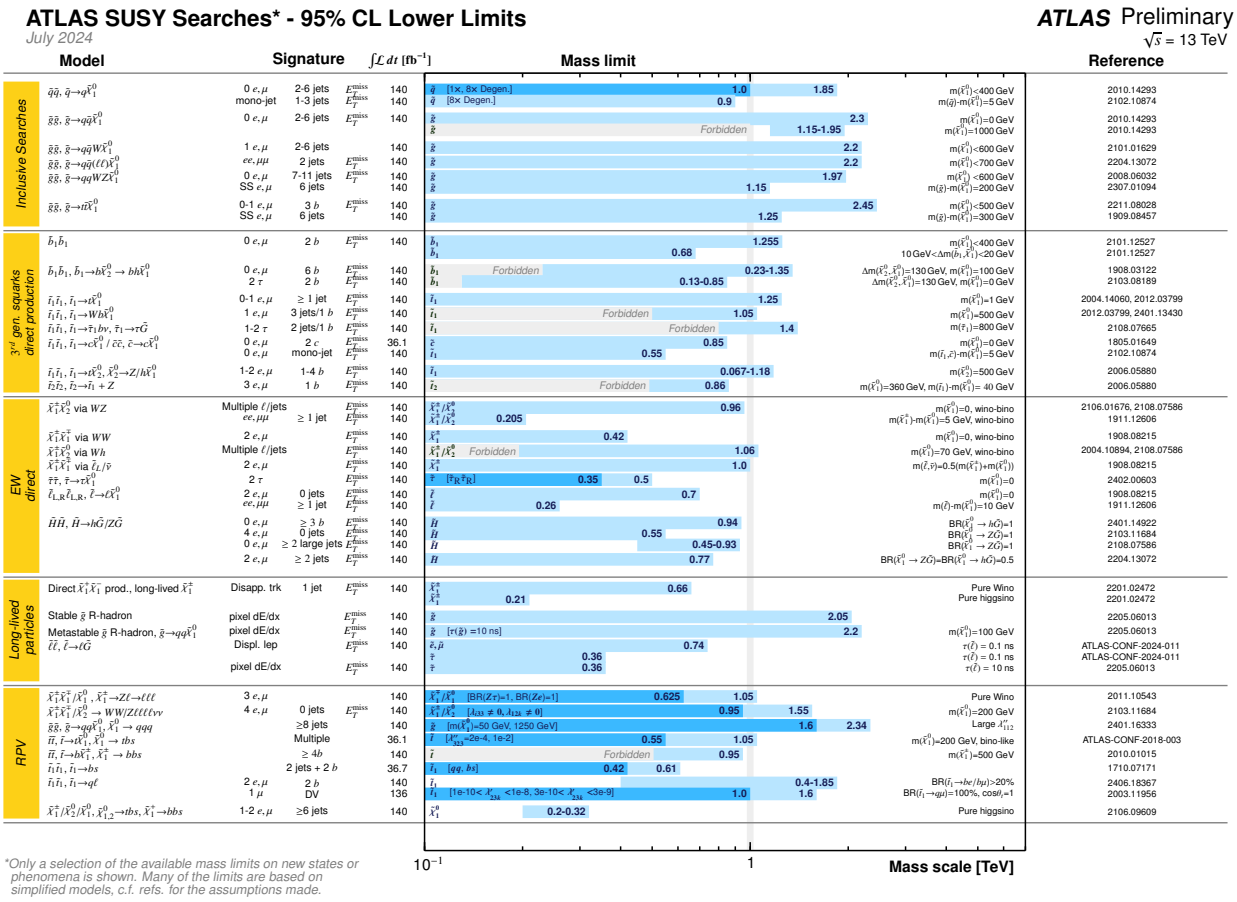

図 1.3: ATLAS 実験における SUSY 粒子の質量の棄却領域（[4] より引用）。

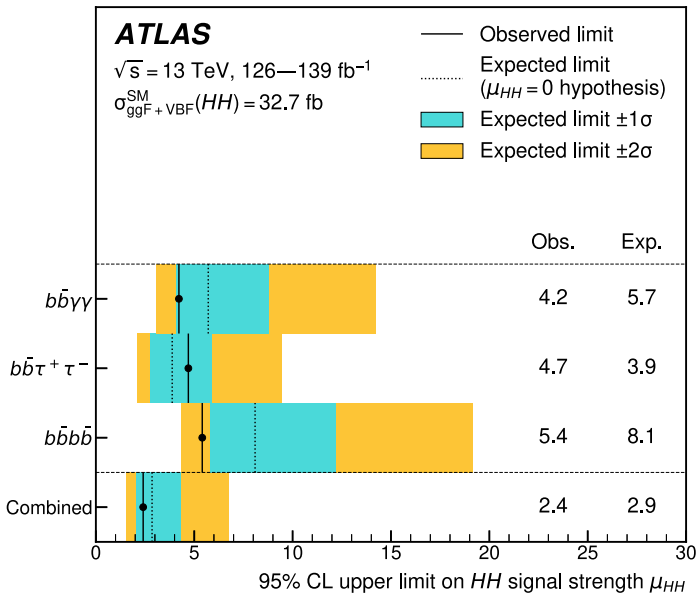

図 1.4: ATLAS 実験におけるヒッグス対生成事象の断面積の上限（[5] より引用）。μ_{HH} は標準模型による予想値に対する比である。

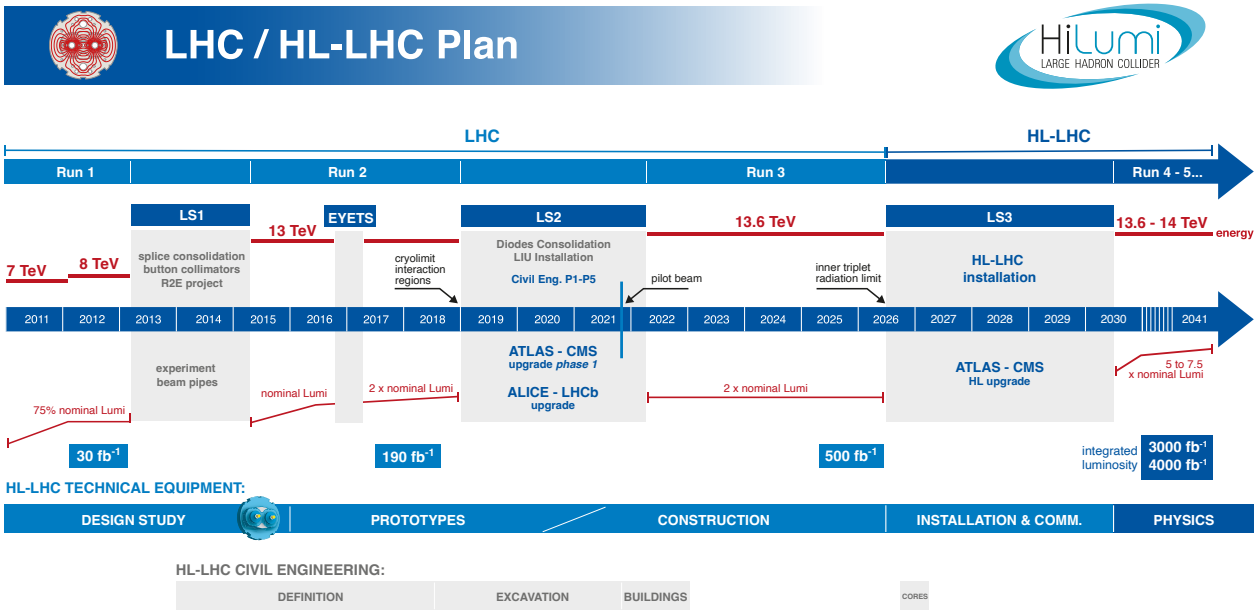

図 1.5: LHC/HL-LHC の計画（[6] より引用）。

1.3　高輝度 LHC に向けた Phase-II アップグレード

図 1.5 に示すように、LHC は現在進行中の第三期運転期（Run3）が完了後、3 年間の Long Shutdown（LS3）の期間で、高輝度化のためにアップグレードされる。瞬間最高ルミノシティは現行の 2×10^{34} cm^{-2}s^{-1} から $5 - 7.5 \times 10^{34}$ cm^{-2}s^{-1} に増強される。積分ルミノシティは 3000 − 4000 fb^{-1} まで蓄積される予定である。

1.3.1　初段トリガーにおける Phase-II アップグレードのモチベーション

瞬間ルミノシティの増大によるパイルアップの増加に伴い、ATLAS 実験においては検出器や TDAQ システムの大幅なアップグレード（Phase-II アップグレード）が予定されている。仮に Phase-II アップグレードを行わない場合でもトリガー発行の横方向運動量の閾値を上げることでトリガーレートを維持して既存の TDAQ システムを使い続けることは可能であるが、重要な物理事象のアクセプタンスが小さくなる。図 1.6 に、初段トリガーのレプトン横方向運動量と代表的な物理事象のアクセプタンスを示す。アップグレードがなかった場合、SUSY 粒子の探索やヒッグス対生成事象に対するアクセプタンスが著しく小さくなることがわかる。例えば、ヒッグス対生成事象において最も感度の高い崩壊チャンネルである $HH \to \tau\tau b\bar{b}$ では、55 % から 25 % にまで低下する。

1.3.2　初段ミューオントリガーにおける Phase-II アップグレード

初段トリガーのトリガーレートは 100 kHz から 1 MHz になり、初段トリガーレイテンシーは 2.5 μs から 10 μs に拡張される。初段トリガーの仕様変更に伴い、初段トリガーのエンドキャップ部ミューオンをカバーする Thin

[*1] エンドキャップとは ATLAS 検出器を円柱形と見た時の底面に当たる領域。詳細は 2.1.1 節

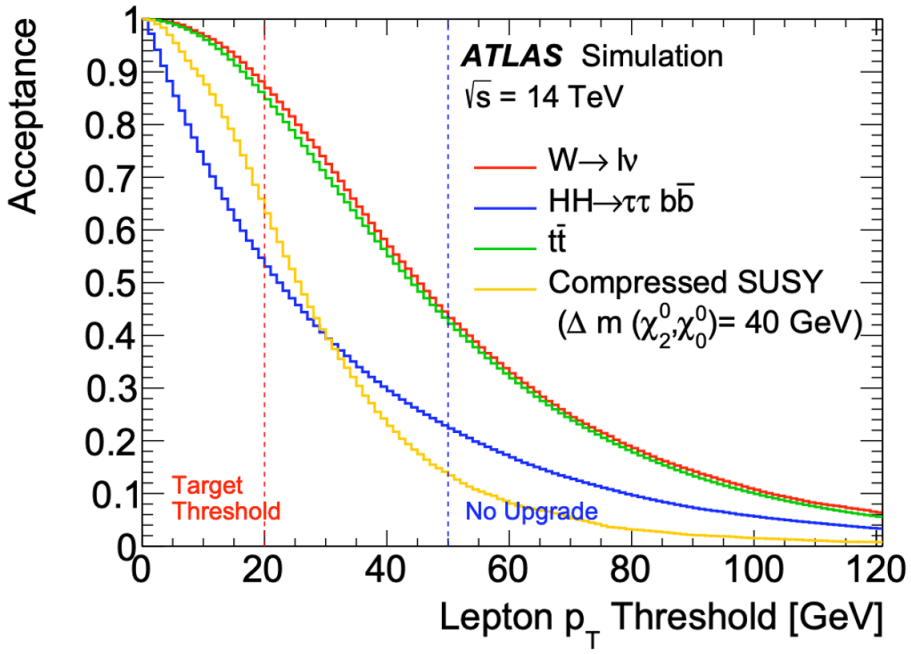

図 1.6: 初段トリガーにおけるレプトン横方向運動量と代表的な物理事象のアクセプタンス（[7] より引用）。

Gap Chamber（TGC）検出器のエレクトロニクスは刷新される。ヒット信号をデジタル化した後のエレクトロニクスは全て新しいものに置き換えられる。新しいエレクトロクスは最新の高速シリアル通信の広帯域を利用して、全てのヒット信号を大規模 FPGA をもつ後段回路に集約させ、トリガー判定が出る 10 μs だけヒット信号をバッファするデザインとなっている。高速ミューオン再構成アルゴリズムや読み出し機構も同一の後段回路に実装される。

1.4　本論文の目的と構成

本研究は、高輝度 LHC-ATLAS 実験に向けて、TGC 検出器エレクトロニクスにおいてトリガーアルゴリズムを担う論理回路を完成させ、理想的なトリガー効率を達成することを目的とする。そのために、トリガー論理回路を実装したシミュレータとファームウェアをビット単位で比較できる検証システムを構築した。その検証システムを駆使して、正確に不具合箇所を特定してファームウェアの修正を行った。

第 2 章では、高輝度 LHC-ATLAS 実験における TGC 検出器エレクトロニクスでの各モジュールの役割を説明する。第 3 章では、トリガーアルゴリズムが実装される論理回路の概観を示し、各トリガーモジュールについて説明する。第 4 章では、トリガー論理回路の検証システムの設計と実装、開発について述べる。第 5 章では、検証と修正を行う前後のトリガーの性能評価の結果を示す。第 6 章では、トリガー論理回路完成への展望を議論する。

第 2 章

高輝度 LHC-ATLAS 実験における TGC 検出器システム

2.1　ミューオンスペクトロメータ

2.1.1　ATLAS 検出器における座標系

ATLAS 検出器では、図 2.1 のように直交座標をとる。x 軸は LHC のリングの中心方向、y 軸は地上方向、z 軸はビーム軸方向である。$z > 0$ の領域を A side、$z < 0$ の領域を C side という。これに合わせて極座標と円柱座標も定義され、ϕ を方位角、θ を天頂角、動径方向を R と定義する。加えて、θ 方向に対応する座標として、擬ラピディティ（Peudorapidity）が定義され、以下のように表される。

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

ミューオンスペクトロメータにおいては $|\eta| < 1.05$ の領域をバレル部といい、$|\eta| > 1.05$ の領域をエンドキャップ部という。LHC は重心系エネルギー 13.6 TeV の陽子衝突であるものの、パートン散乱であるため、素過程に注目したときに z 軸方向の運動量は事象ごとに異なる値を持つ。一方 z 軸と垂直な方向の運動量は 0 であり、終状態粒

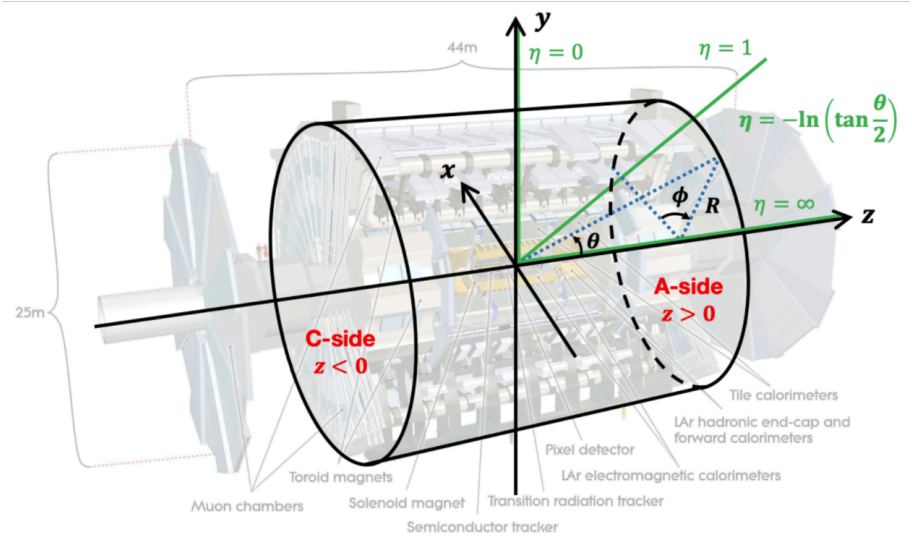

図 2.1: ATLAS 検出器の座標系（[8] から図を引用）。

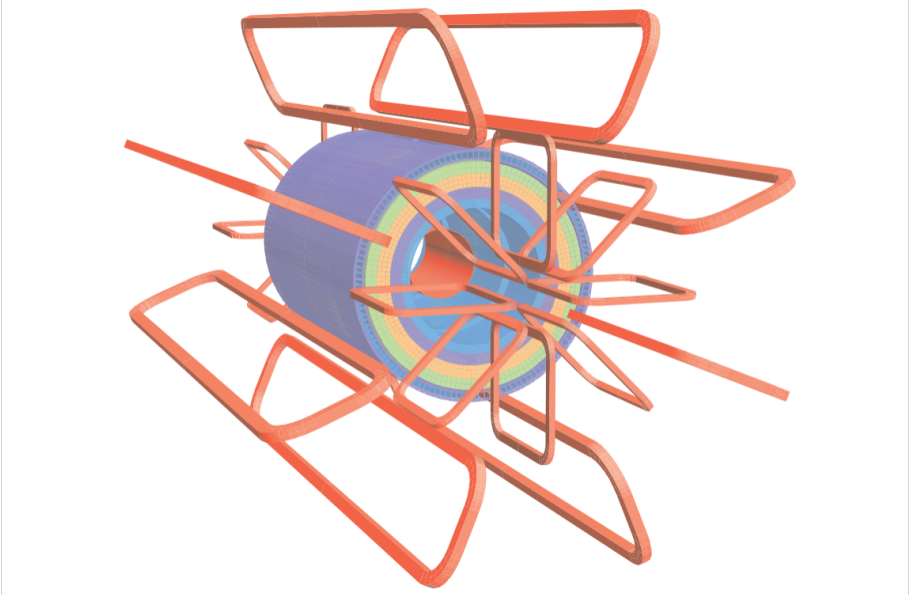

図 2.2: トロイド磁石（[5] から図を引用）。

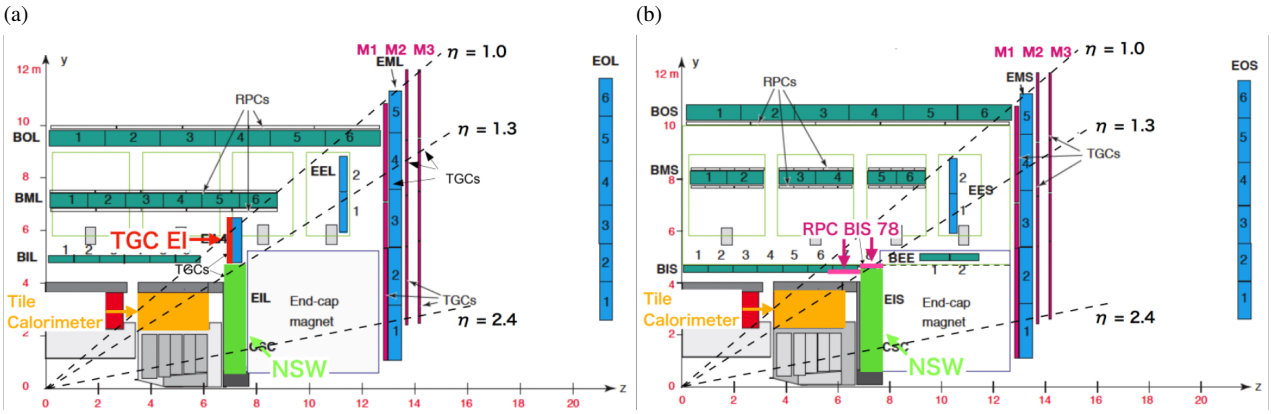

図 2.3: R-z 方向で見たミューオンスペクトロメータの配置（[9] から図を引用）。トロイド磁石が存在する領域（Small Sector）とその隙間（Sector Sector）で配置が異なる。(a)Large sector。(b)Small sector。

子の全運動量が保存するため、この方向の運動量（横方向運動量）が重要となる。本研究で対象とする初段ミューオントリガーでも横方向運動量の測定が重要なテーマである。

2.1.2　トロイド磁石

ATLAS 検出器には内部飛跡検出器を透過してきた荷電粒子の運動量を測定するためのトロイド磁石が設置されている（図 2.2）。トロイド磁石は ϕ 方向磁場を生成し、荷電粒子を θ 方向に曲げる。その曲がり度合いにより、運動量を測定する。トロイド磁場は ϕ 方向に 8 回対称になっており、完全に円柱形となって一様であるわけではなく、その磁場の周期性に対応して不均一になっている。

(a) (b)

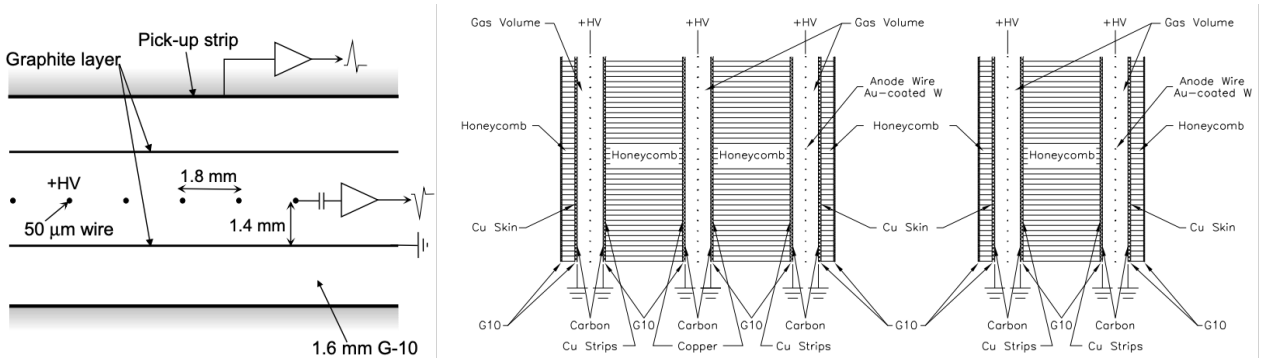

図 2.4: TGC の断面図（[3] より図を引用）。(a) ガスギャップの構造 (b)Triplet と Doublet の構造

2.1.3 ミューオンスペクトロメータ

図 2.3 に示すように、ATLAS 検出器でのミューオンスペクトロメータは New Small Wheel（NSW）、Resistive Plate Chamber（RPC）、Monitor Drift Tube（MDT）、そして TGC で構成されている。TGC と PRC は高速応答によりトリガー発行に特化した検出器であり、TGC はエンドキャップ部、PRC はバレル部を担当する。MDT はミューオン精密測定用の検出器であるが、Phase-II 以降の初段トリガーシステムに組み込まれる。これは MDT の時間応答の長さに対してレイテンシーを伸ばしたことによって可能となる。NSW はトリガーおよび精密測定用の検出器であり、small-strip TGC（sTGC）と Micromegas から構成される。

2.2 TGC 検出器

TGC 検出器は本研究の対象となる検出器である。エンドキャップ部（$1.05 < |\eta| < 2.4$）をカバーし、トロイド磁場の内側に位置する Endcap Inner（EI）と外側に位置する Big Wheel（BW）からなる。TGC BW は 3 つのステーションからなり、内側から M1、M2、M3 ステーションという。

2.2.1 ガスチェンバー

TGC 検出器は、図 2.4(a) のような構造の Multi Wire Propotional Chamber（MWPC）である。チェンバー内中空に張られたワイヤーとチェンバー内壁に塗布されたストリップは直交しており 2 次元読み出しが可能である。CO_2 と n-C_5H_{12} が 55:45 の割合で封入されており、ワイヤには 2.8k̇V の電圧が印加されている。25 ns 毎の陽子バンチ交差に対してミューオンがどのバンチ交差由来かを識別するバンチ交差識別（Bunch Crossing Identification, BCID）を行うために、高い時間分解能が要求されており、入射荷電粒子によるガス分子の電離電子のドリフト時間を小さくする必要があるため、ワイヤー間隔は 1.8 mm である。また、読み出し間隔を短くするという要請からワイヤーとストリップの間隔は 1.4 mm となっている。

ステーション毎にガスチェンバーは 2 層または 3 層が重ねられ、ペーパーハニカムで支持されている。図 2.4(b) はその断面構造である。M1 のみガス層が 3 つあり、ワイヤーが 3 層、ストリップが 2 層存在する（Triplet）。M2、M3 はガス層は 2 つで、ワイヤーもストリップも 2 層である（Doublet）。

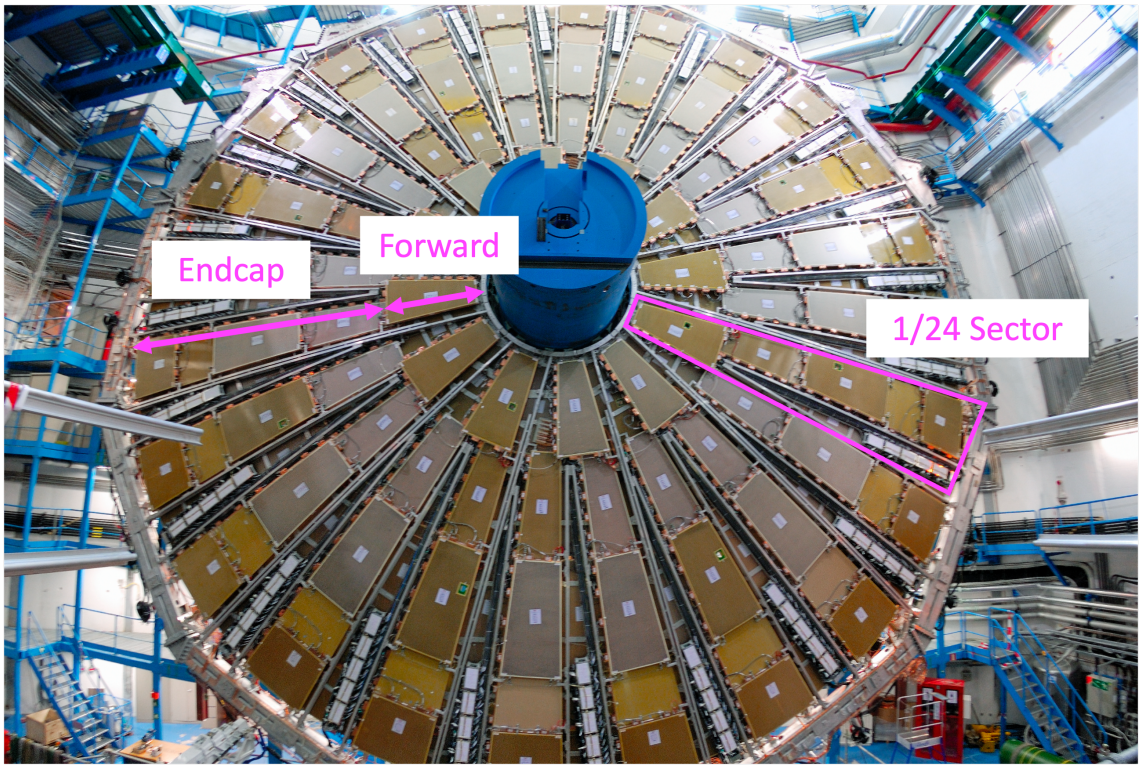

図 2.5: TGC の全体写真（[10] を一部編集）。

2.2.2　セクター構造

TGC 検出器は図 2.5 で示すように、$1.05 < |\eta| < 1.92$ の領域をエンドキャップ領域[*1]、$1.92 < |\eta| < 2.4$ の領域をフォワード領域という。エンドキャップ領域は 48 回対称、フォワード領域は 24 回対称であり、エンドキャップ領域 2 つとフォワード領域 1 つで構成される 1/24 セクターの領域をトリガーセクターという。トリガーセクター 1 つでトリガー論理回路が閉じており、トリガーセクターに含まれるこの三つの領域のうち、エンドキャップのうちの二つをエンドキャップ ϕ0、エンドキャップ ϕ1 と呼んで区別する。

2.2.3　横方向運動量の再構成

TGC 検出器のヒット情報から運動量を再構成する方法の概要を図 2.6 に示す。ミューオン飛跡は理想的には M1/2/3 の各ステーションにヒットを残す。TGC 検出器はトロイド磁場の外側にあるので、その飛跡は直線である。ビームの衝突点と M3 のヒット点（ピボット）を結んだ直線と M1 のヒット点のズレを $(d\theta,d\phi)$ とする。すると、高運動量ミューオンほどトロイド磁場で曲がりにくいので、$d\theta$ は小さくなる。このように、$d\theta$ の大きさから運動量を計算する。実際にはトロイド磁場は不均一であるので、横方向運動量は $d\phi$ とも相関を持つ。

実際にはあらかじめシミュレーションで、ヒットのパターンと $(d\theta,d\phi)$ の対応関係とある位置での $(d\theta,d\phi)$ と横方向運動量の対応関係を調べておき、ヒットパターンからその対応関係を元に横方向運動量を求める、という手法をとる。この対応関係をまとめたテーブルを Look Up Table（LUT）という。LUT をハードウェアの大容量メモリに書き込んでおき、入力ヒットパターンに対応するアドレスを参照することで、高速で横方向運動量を再構成する

[*1] バレルと対となるエンドキャップという呼称とは別である。

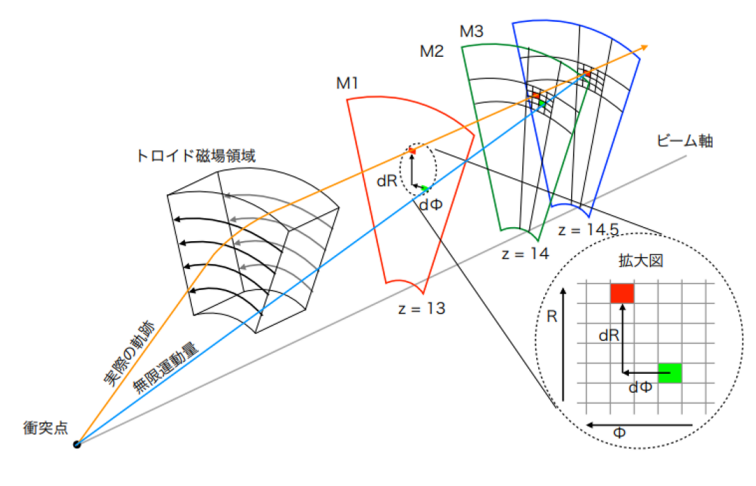

図 2.6: TGC におけるミューオン運動量再構成の手法（[11] から引用）。図中では dR と表しているが、本論文中の $d\theta$ と同一方向の座標を使った表現である。実際にはミューオンは衝突点からビーム方向側にずれた箇所で発生することがあるが、この分のずれは LUT で吸収させている。

ことができる。

2.3　高輝度 LHC-ATLAS 実験での TGC 検出器エレクトロニクス

ATLAS 検出器の Trigger DAQ（TDAQ）システムは初段のハードウェアトリガー（Level-0,L0）と後段のソフトウェアトリガー（Event Filter）によって構成される。TGC 検出器エレクトロニクスは L0 のエンドキャップ部ミューオントリガーを担当する。

2.3.1　TDAQ システム

LHC の高輝度化に伴い、陽子衝突事象が増加するのに合わせて、TDAQ システムはアップグレードされる。初段トリガーレートは 100 kHz から 1 MHz に、後段トリガーレートは 3.3 kHz から 10 kHz となるように改修される。加えて、初段トリガーレイテンシーは 2.5 μs から 10 μs に延長され、MDT も利用した柔軟で高精度なトリガーを実現する。ATLAS 検出器の TDAQ システムの概要を示したのが図 2.7 である。Global Trigger は複数のオブジェクトを含む事象をトリガー対象として L0 Calo や MUCTPI からの情報を総合してそれらのトポロジーが整合するかどうかを判定し、CTP に送る。CTP では、それらの入力に対して、図 2.8 で示されるトリガーメニューを満たすかどうか判断し、トリガー判定を行う。トリガー判定を通過したバンチ交差に対しては、Level-0 Accept（L0A）を各フロントエンド回路に分配する。バンチ交差時間から L0 トリガー判定までの処理は固定レイテンシー 10 μs で行われる。L0A を受け取ったフロントエンド回路は該当するバンチ交差のヒットデータを Front-End Link eXchange（FELIX）経由で読み出し、Event Filter に送る。Event Filter でのトリガー判定を通過したバンチ交差はそのヒットデータが記録される。L0Muon は MUCTPI に送るためのミューオン飛跡再構成を行い、L0Muon における Endcap Sector Logic（SL）が TGC 検出器の入力を元にミューオン再構成を主に行う回路であり、以降の節で詳述する。

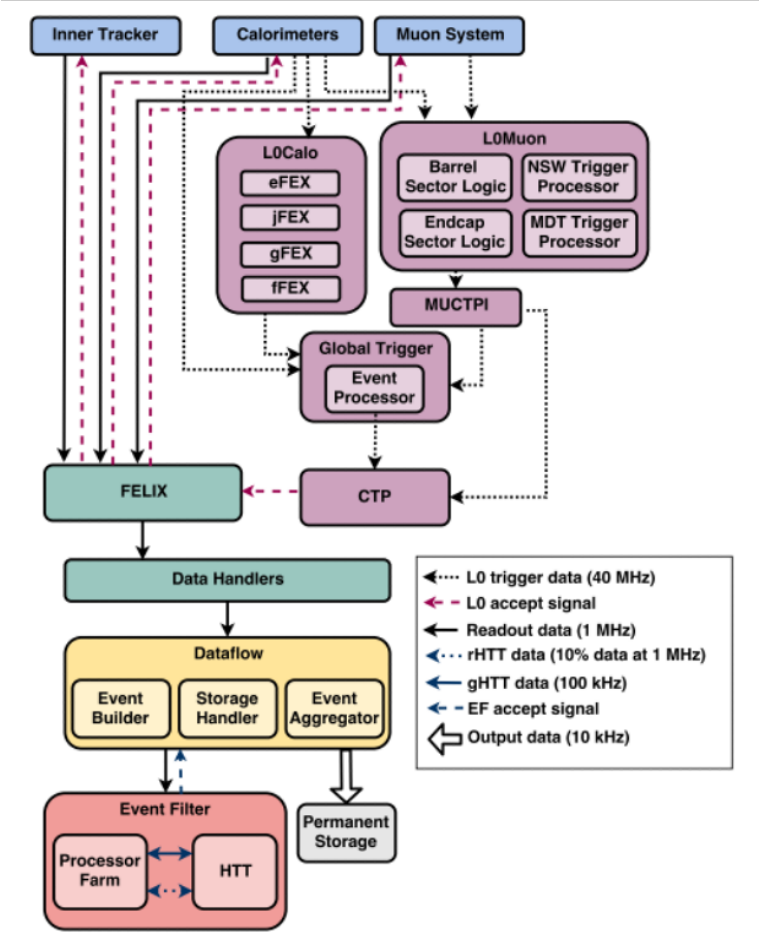

図 2.7: Phase-IIupgrade での TDAQ システム（[7] から引用）。

Trigger Selection	Run 1 Offline p_{T} Threshold [GeV]	Run 2 (2017) Offline p_{T} Threshold [GeV]	Planned HL-LHC Offline p_{T} Threshold [GeV]	L0 Rate [kHz]	After regional tracking cuts [kHz]	Event Filter Rate [kHz]
isolated single e	25	27	22	200	40	1.5
isolated single μ	25	27	20	45	45	1.5
single γ	120	145	120	5	5	0.3
forward e			35	40	8	0.2
di-γ	25	25	25,25		20	0.2
di-e	15	18	10,10	60	10	0.2
di-μ	15	15	10,10	10	2	0.2
$e-\mu$	17,6	8,25 / 18,15	10,10	45	10	0.2
single τ	100	170	150	3	3	0.35
di-τ	40,30	40,30	40,30	200	40	$0.5^{\dagger\dagger\dagger}$
single b-jet	200	235	180	25	25	$0.35^{\dagger\dagger\dagger}$
single jet	370	460	400			0.25
large-R jet	470	500	300	40	40	0.5
four-jet (w/ b-tags)		$45^{\dagger}$(1-tag)	65(2-tags)	100	20	0.1
four-jet	85	125	100			0.2
H_{T}	700	700	375	50	10	$0.2^{\dagger\dagger\dagger}$
$E_{\mathrm{T}}^{\mathrm{miss}}$	150	200	210	60	5	0.4
VBF inclusive			2x75 w/ ($\Delta\eta > 2.5$ & $\Delta\phi < 2.5$)	33	5	$0.5^{\dagger\dagger\dagger}$
B-physics††				50	10	0.5
Supporting Trigs				100	40	2
Total				1066	338	10.4

図 2.8: Level-0 trigger における代表的なトリガーメニュー（[7] から引用）。初段トリガーでの電子の運動量閾値が RUN3 と比べて、引き下げられていることに加えて、di-τ や four-jet のトリガーメニューも拡張およびレートの維持がなされている。後者は dihiggs 解析における感度を伸ばすためである。

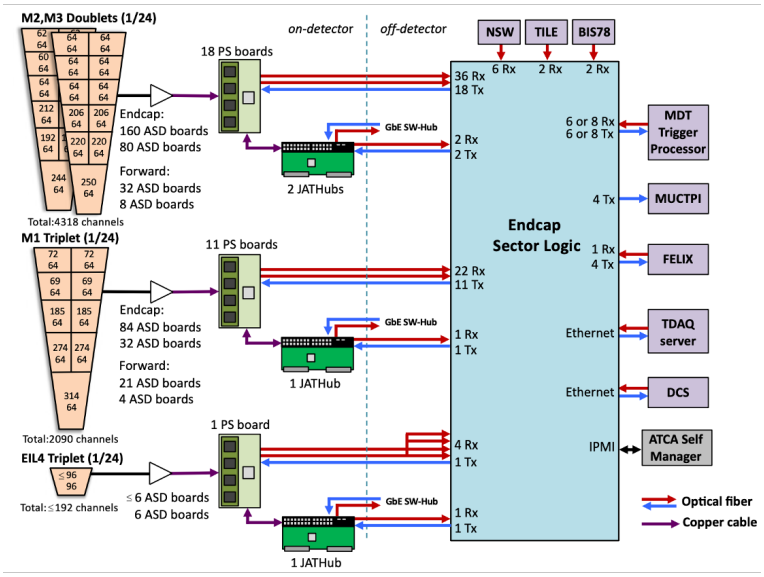

図 2.9: TGC エレクトロニクスの全体像（[7] から引用）。

2.3.2 TGC 検出器エレクトロニクスの全体像

TGC 検出器エレクトロニクスの全体像を示したのが図 2.9 である。この構造は片側 24 回対称の一つであり、ATLAS 検出器全体で 48 個存在する。TGC 検出器とそのすぐ後段の Amplifier-Shaper-Discriminator（ASD）は現行の RUN3 と同じものが使用され、それ以降のエレクトロニクスは新しいものである。TGC の電流信号は ASD でデジタル化され、後段の Primary ProceSsor board（PS board）に送られる。そこでは 25 ns 毎の陽子バンチ交差に対してミューオンがどのバンチ交差由来かを識別するバンチ交差識別（Bunch Crossing Identification, BCID）が行われる。BCID されたヒット信号はさらに後段の SL に送られ、ヒットパターンから飛跡と横方向運動量が再構成されるほか、CTP から L0 トリガー判定が出るまでの間バッファーされ、L0A が出た場合は FELIX 経由で読み出される。ヒットデータの読み出しおよびミューオン飛跡再構成は固定レイテンシーで行われる。SL から出力されたミューオン飛跡候補の情報は MDT Trigger Processor に送られ、MDT による高分解能の横運動量測定の情報を付加して、MUCTPI に送られる。CTP がトリガー判定の結果、L0A を出した場合は SL はバッファしていたヒット信号を FELIX に送る。詳細は後述するが、SL は NSW や RPC BIS7/8、Tile カロリメータといった磁場内部の検出器のヒット信号とミューオン飛跡のコインシデンスを取るトリガーアルゴリズムを備えているので、そのためのインターフェイスを持つ。

データパス以外のコントロールやコンフィギュレーションのためのエレクトロニクスは以下で説明する。PS board のコンフィギュレーションやリブートを行うための制御モジュールとして、JTAG AssisTance Hub（JATHub）が存在する。また、JATHub は PS Board 個体毎のクロック位相測定機能も担っており、その位相情報をもとに全 1400 台の PS Board のクロック位相を揃える。JATHub のコントロールや JATHub へクロック位相測定のための基準クロックを分配するモジュールが Timing Alignment Master（TAM）である。LHC と同期したクロックやトリガー判定の情報を分配する信号を Timing, Trigger and Control（TTC）信号といい、TTC 信号は FELIX を通じて SL に分配され、SL はこれらの TGC エレクトロニクス全体に分配する。

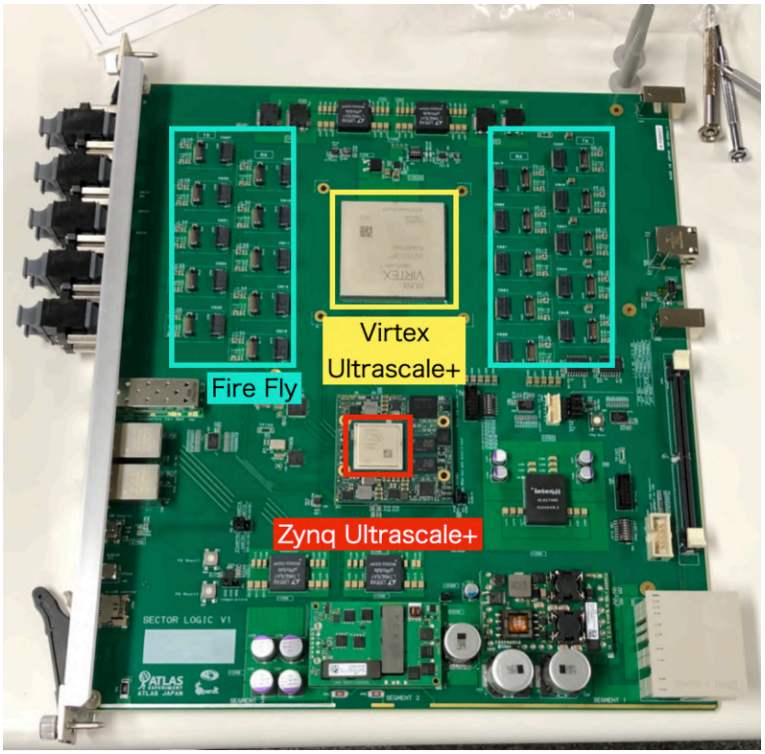

図 2.10: SL 第一試作機の写真（[9] から引用）。

2.3.3　Endcap Sector Logic（SL）

SL は、PS board からのヒットデータを受けてトリガー演算を行いミューオン飛跡再構成を行う、L0 トリガー判定までヒット信号をバッファーするなど多くの役割を持つ。そのために Virtex UltraScale+ という大規模 FPGA（以後、SL FPGA）と Zynq UltraScale+ MPSoC という System on Chip（以後、MPSoC）を搭載している。SL 第 1 試作機の写真が図 2.10 である。光リンクをデジタル電気信号に変換する FireFly やイーサネットポートとなる RJ45 コネクタを備えている。

MPSoC は SL FPGA や PS board へのレジスタアクセスを行いコントロールする。MPSoC には FPGA 領域（Progammable Logic、PL）とプロセッサー領域（Processing System、PS）があり、PS の特定のメモリは PL のメモリに接続されている。ユーザーはイーサネットで PS にアクセスし、PS 上でアプリケーションを操作することにより、MPSoC を介した SL FPGA の制御やデータ読み出しを行うことができる。これを利用し、PS board や TAM、SL FPGA のレジスタアクセスを行う。また、本研究では SL ハードウェアを使用したトリガー試験テストベンチの構築に応用している。

SL FPGA には読み出し回路とトリガー回路が実装される。読み出し回路は PS board から受け取ったヒット信号とトリガー演算の中間出力を L0 トリガー判定が出るまで L0 バッファーに格納する。L0A が出た場合はそれに対応するバンチ交差のデータとしてパックして成形し、FELIX とのインターフェースへ出力する。トリガー論理回路については、第 3 章で詳述する。SL FPGA は Super Logic Region（SLR）という 4 つのシリコンダイで構成されている大規模 FPGA である。SLR 間は Super Long Line（SLL）という専用のワイヤーで信号の伝達が行われる。SLR 毎にリソースの制限があることや SLL を介した通信にはタイミング制約が厳しくなることを考慮し、各 SLR ごとに効率的にモジュールが配置されている。SLR ごとのモジュールの配置を示したのが図 2.11 である。SLR0/2 はそれぞれエンドキャップ ϕ0/1、SLR3 はフォワードに対応しており、読み出し回路と前段のトリガー回路（BW

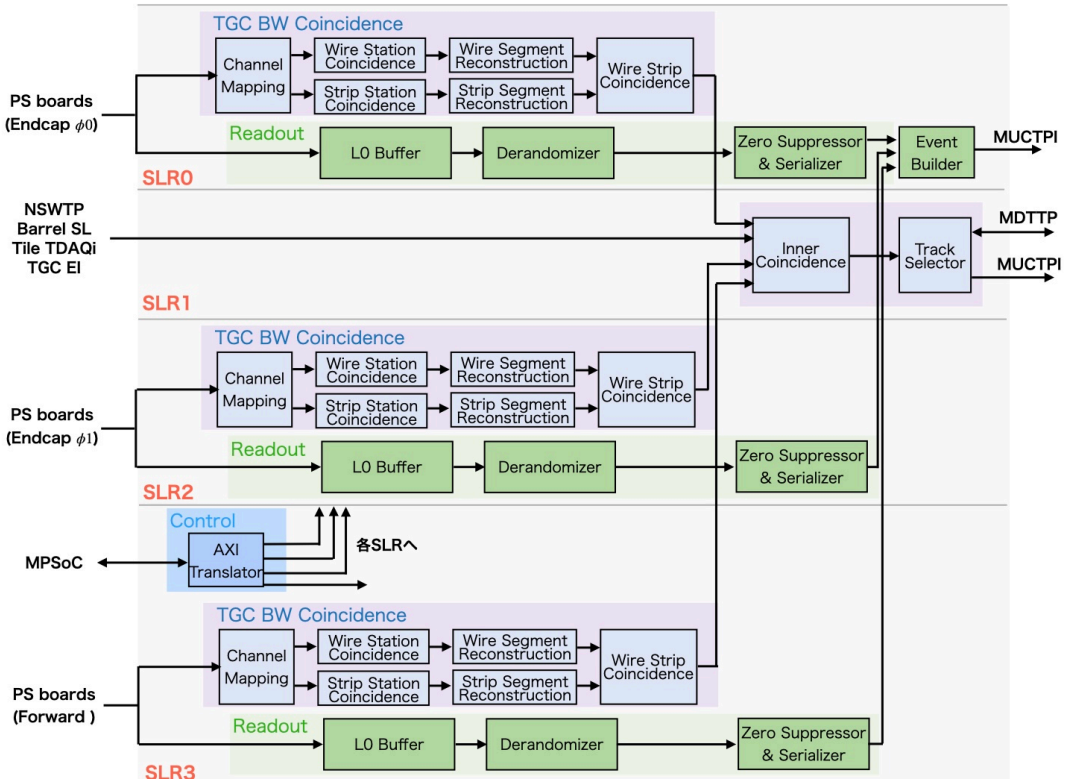

図 2.11: SL FPGA に実装されるモジュールの配置（[9] から引用して一部修正）。ピン配置の変更により、Event Builder の位置が SLR3 から SLR0 に変更される。

までのコインシデンスロジック）が配置される。SLR1 には、他の検出器とのコインシデンスをとる後段のトリガー回路が配置されるため、NSWTP や RPC BIS7/8、Tile カロリメータ、TGC EI からの信号は SLR1 に入力される。また、FELIX とのインターフェースは SLR0 と接続している[*2]ため、読み出し回路は最終的に SLR0 に集約する。MPSoC とのインターフェースは SLR3 に位置しており、ここが起点となってレジスタアクセスや MPSoC へのデータ読み出しが行われる。

[*2] この仕様は変更を経た、2024.12 現在最新のものである。

第 3 章

トリガー論理回路の全体像

トリガー論理回路は Channel Mapping、Station Coincidence、Segment Reconstruction、Wire Strip Coincidence、Inner Coincidence、Track Selector という主に 6 種類の論理回路から構成される。その全体像を示したのが図 3.1 である。6 種類の論理回路の入力部には、複数の入力情報の各タイミングを揃えるための可変長シフトレジスタを配置している。[*1]Channel Mapping は PS board からのヒット信号を以降のトリガー論理回路の入力に合わせた形式に並べ替える。Station Coincidence は各ステーションの中の 3 層または 2 層の発火チャンネルでコインシデンスをとって代表点を出力する。Segement Reconstruction は各ステーションからの代表点の組み合わせに対して、直線飛跡を成すかどうかの判定を行う。直線飛跡と判別されたものに対しては無限運動量飛跡（ビームの衝突点と M3 ピボットを結んだ直線）からの角度のズレ $(d\theta, d\phi)$ を出力する。Wire Srtip Coincidence は各 $(d\theta, d\phi)$ から横方向運動量、飛跡の代表点番号から座標の情報を求め、ミューオン飛跡候補を出力する。Inner Coincidence は NSW、RPC BIS7/8、Tile カロリメータ、TGC EI のヒット信号と飛跡候補の位置関係の整合性からフェイクミューオンかどうかを判断する。Track Selector は出力した飛跡候補を運動量の高い順に優先順位をつけて、後段に送る。以降でそれぞれの論理回路の動作と実装を説明する。

3.1 Channel Mapping

Channel Mapping は PS board からのヒット信号を並び替えてトリガー論理回路に合う形に並べ替える。加えて、特定のチャンネルに対しては OR 回路で発火チャンネルを複製させる。ワイヤーチャンネルに関しては、隣り合うチェンバーが重なりあっているため、オーバーラップしている領域では複数のチェンネルが同一の η 座標を持つ。このようなチャンネルの組み合わせでは OR の処理によって、どちらか一方が発火した際には処理後のワイヤを発火させて、その後の論理回路内では同一のチャンネルとして扱う。次にストリップチャンネルに関しては、低運動量ミューオンは η 方向に大きく曲がり、M1/2/3 ステーションで異なる番号のチェンバーにヒットを残すことを想

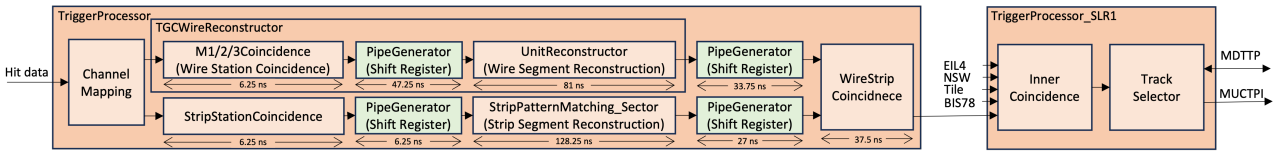

図 3.1: トリガー論理回路の全体像。現状のレイテンシーとファームウェア上でのモジュール名を明記している。

[*1] レイテンシーを揃えるのに加えて、タイミング制約を満たす（ネガティブスラックを解消する）という目的もある。FPGA 内での実質的な経路が長く、コード上の要求レイテンシーに間に合わない場合はパイプラインを挿入して論理経路を短くすることで解消できる

図 3.2: 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（[12] から引用）。4 個（または 5 個）のチャンバーが並んでいるが上から順に E1、E2、E3、E4（、E5）である。

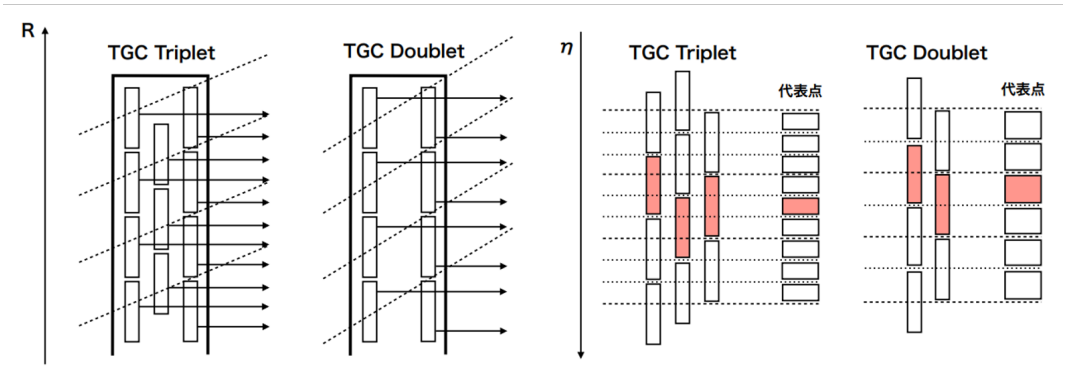

図 3.3: ワイヤーチャンネルにおける staggering 構造（[13] から引用）。

定して、M1/2 において一部の隣り合うチェンバーの同一 ϕ 座標のストリップチャンネルを重複して発火させる。[*2] それを表したのが図 3.2 である。これは、後段の Segement Reconstruction でのステーション間コインシデンスで、M3 のチェンバーとコインシデンスが取られるチェンバーが M1/2 ステーションで一つずつのみであることによる。例えば、図 3.2 で図中央上段に示すように、M3 の E3 にヒットがあった場合は、ミューオン飛跡が M2 の E3 と E4、M1 の E2 と E3 が通った飛跡とコインシデンスをとるため、Channel Mapping の段階で M2 の E3 と E4、M3 の E2 と E3 に OR の処理を施す。

3.2　Station Coincidence

Station Coincidence はステーション内で 3 層または 2 層のチャンネルに対してコインシデンスをとり代表点を選ぶ。図 3.3 のように、ステーション内のチャンネル位置はズレた（staggered）構造をなしている。3 層では 3/3、2/3、1/3 という 3 種類のコインシデンス、2 層では 2/2、1/2 という 2 種類のコインシデンスの代表点を出力する。これにより、位置分解能を上げながらデータ量を削減し、ヒット位置を出力することができる。ワイヤーとスト

[*2] 別の立場から言い換えると、ϕ 座標を求めるだけで良いならば全てのチェンバーで OR をとるべきであるところを、無意味なステーション間コインシデンス（トリガー判定をパスしない超低運動量ミューオン飛跡など）を取らないように、最大で隣合うチェンバーの間だけで OR 処理をしているとも言える。

(a)

A レイヤー			A1		
B レイヤー		B1		B2	
C レイヤー	C1		C2		
代表点番号	N-3	N-2	N-1	N	N+1	N+2	N+2

(b)

αレイヤー		α1		α2
βレイヤー	β1		β2	
代表点番号	N-2	N-1	N	N+1	N+2

図 3.4: TGC Triplet/Doublet での Station Coincidence における N 番目の代表点に対するレイヤーの定義。N 番目の代表点に対して、A/B/C レイヤーおよび α/β レイヤーは図のように定義される。その図中の上下の順序は一例であり、隣の代表点であれば A/B/C レイヤーおよび α/β レイヤーの順序が入れ替わる。(a)3 層の場合。A は代表点が中央、B は B1 が代表点の左側、C は C2 が代表点の右側に位置するようなレイヤーである。(b)2 層の場合。α は $\alpha 1$ が代表点の左側、β は $\beta 2$ が代表点の右側に位置するようなレイヤーである。

リップが別々の論理回路で動作する。以下の論理回路[*3]について先行研究で議論されており詳細は [12] を参照されたい。

3.2.1 Wire Station Coincidence（M1/2/3Coincidence1/2/3OutOf2/3）

Wire Station Coincidence[*4]は M1 では 3 層、M2/3 では 2 層であるので、以下の 5 種類の論理回路が並列に動作する。まず M1 つまり triplet の場合を説明する。図 3.4(a) は staggering 構造を持ったチャンネルの一例であり、以降の説明で用いる A/B/C レイヤーの定義を表す。3/3、2/3、1/3 のコインシデンスがあり、N 番目代表点のそれぞれの出力 $O_N^{3/3}$、$O_N^{2/3}$、$O_N^{1/3}$ は以下の論理式で表せる。

$$O_N^{3/3} = A1 \times B1 \times C2 \tag{3.1}$$

$$O_N^{2/3} = A1 \times B1 \times \overline{C1} \times \overline{C2} + A1 \times \overline{B1} \times \overline{B2} \times C2 + \overline{A1} \times B1 \times C2 \tag{3.2}$$

$$O_N^{1/3} = A1 \times \overline{B1} \times \overline{B2} \times \overline{C1} \times \overline{C2} \tag{3.3}$$

ここで $\times$ は AND を、$+$ は OR、オーバーラインは NOT を意味する。2/3 と 1/3 コインシデンスに関しては単純な 3 層中の 1、2 層の発火で代表点を出力するのではなく、連続したヒットチャンネルがあったときに無闇に代表点を増やさない論理になっている。

具体例として 2/3 コインシデンスにおける代表点の出力を説明する。図 3.5(a) は 2/3 コインシデンスによる発火の例である。図中の N 番目の代表点は 3.2 式の右辺第 1 項が適用され出力される。一方で、無闇に代表点を増やさない実装の例を図 3.5(b) に示す。図中 N 番目の代表点は一見、2/3 コインシデンスが成立しているが、3.2 式のどの項にも当てはまらず、出力されない。C1 が発火しているため、隣の N-1 の代表点で 3/3 コインシデンスが成立していて、連続した代表点の出力を防いでいる。

次に M2 と M3 の doublet の場合を説明する。図 3.4(b) は staggering 構造を持ったチャンネルの一例であり、以降の説明で用いる α/β レイヤーの定義を表す。2/2、1/2 のコインシデンスがあり、N 番目の代表点のそれぞれの出力を $O_N^{2/2}$、$O_N^{1/2}$ は以下の論理式で表せる。

$$O_N^{2/2} = \alpha 1 \times \beta 2 \tag{3.4}$$

$$O_N^{1/2} = \alpha 1 \times \overline{\beta 1} \times \overline{\beta 2} + \beta 2 \times \overline{\alpha 1} \times \overline{\alpha 2} \tag{3.5}$$

[*3] この論理式はまだ最適化の余地があり、今後仕様変更となる可能性がある。
[*4] ファームウェア上では TGCWireReconstructer_XX の内部モジュールである MXCoincidence_XOutOfX_EC である。

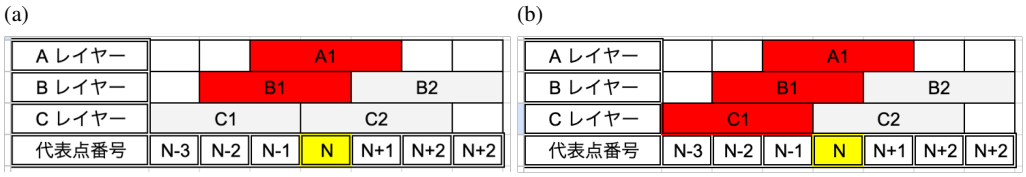

図 3.5: Triplet wire station coincidence における 2/3 コインシデンスの代表点出力の例。(a)2/3 コインシデンスが出力される例。(b)2/3 コインシデンスを出力しない例

ここで × は AND を、＋ は OR、オーバーラインは NOT を意味する。1/2 コインシデンスに関しては単純な 2 層中の 1 層の発火で代表点を出力するのではなく、連続したヒットチャンネルがあったときに無闇に代表点を増やさない論理になっている。

Wire Station Coincidence は 160 MHz クロック[*5]の 1 tick で駆動し、そのレイテンシーは 6.25 ns である。

3.2.2　Strip Station Coincidence

Strip Station Coincidence はすべてのステーションで 2 層構造である。図 3.4(b) は staggering 構造を持ったチャンネルの一例であり、以降の説明で用いる α/β レイヤーの定義を表す。2/2、1/2 のコインシデンスがあり、N 番目の代表点のそれぞれの出力を $O_N^{2/2}$、$O_N^{1/2}$ は以下の論理式で表せる。

$$O_N^{2/2} = \alpha 1 \times \beta 2 \tag{3.6}$$

$$O_N^{1/2} = \alpha 1 \times \overline{\beta 2} \times (\overline{\beta 1} + \alpha 2) + \beta 2 \times \overline{\alpha 1} \times (\overline{\alpha 2} + \beta 1) \tag{3.7}$$

ここで × は AND を、＋ は OR、オーバーラインは NOT を意味する。1/2 コインシデンスに関しては単純な 2 層中の 1 層の発火で代表点を出力するのではなく、連続したヒットチャンネルがあったときに無闇に代表点を増やさない論理になっている。

Strip Station Coincidence は 160 MHz クロック[*6]の 1 tick で駆動し、そのレイテンシーは 6.25 ns である。

3.3　Segment Reconstruction

図 3.6 で示されるように、Segement Reconstruction は前段回路から受け取った代表点の組み合わせからそれに対応する ($d\theta$,$d\phi$) を出力する。[*7]その対応関係をまとめているテーブルである LUT は FPGA 上の RAM に書き込まれ、代表点の組み合わせはそのアドレスに対応する。[*8]この実装により複雑な再構成過程を短い時間で完了できる。出力した ($d\theta$,$d\phi$) のうち、ヒット層数が多いものを選び後段に送る。M3 の代表点を起点にして M1 と M2 の代表点を探しコインシデンスをとり、それに対応する $d\theta$ や $d\phi$ を出力する、というロジックである。Segment Reconstruction はワイヤーとストリップで独立なモジュールで動作する。以下でそれぞれについて詳細に解説する。

[*5] トリガー論理回路のクロックドメインを 160 MHz に統一してタイミング制約を緩和するために、40 MHz から変更されて 160 MHz となった

[*6] トリガー論理回路のクロックドメインを 160 MHz に統一してタイミング制約を緩和するために、40 MHz から変更されて 160 MHz となった

[*7] 各ステーション間でコインシデンスをとって代表点を選び、すべてのステーションから代表点が出力されていない限りは $d\theta$ や $d\phi$ は出力できない。

[*8] 厳密には代表点の組は複数で一つのアドレスに対応し、そのアドレスに格納されたデータ列にそれらの代表点の組すべてに対応するデータが格納される。

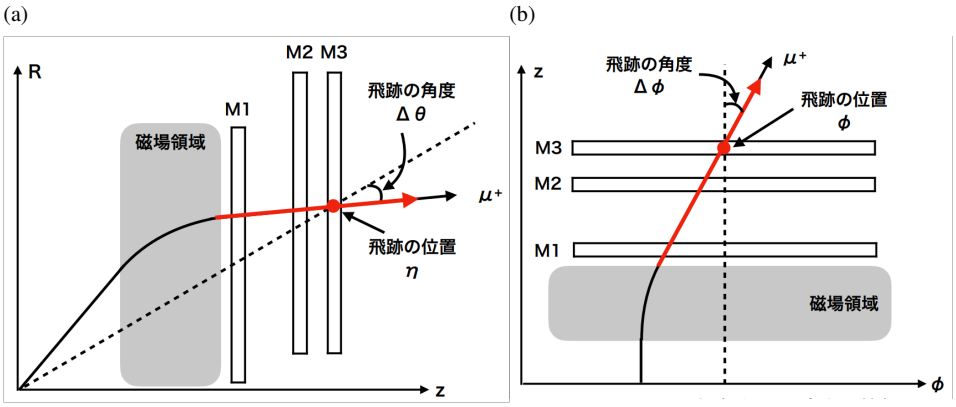

図 3.6: Segment Reconstruction での ($d\theta$,$d\phi$) の再構成（[13] から図を引用）。(a)θ 方向の場合。(b)ϕ 方向の場合。

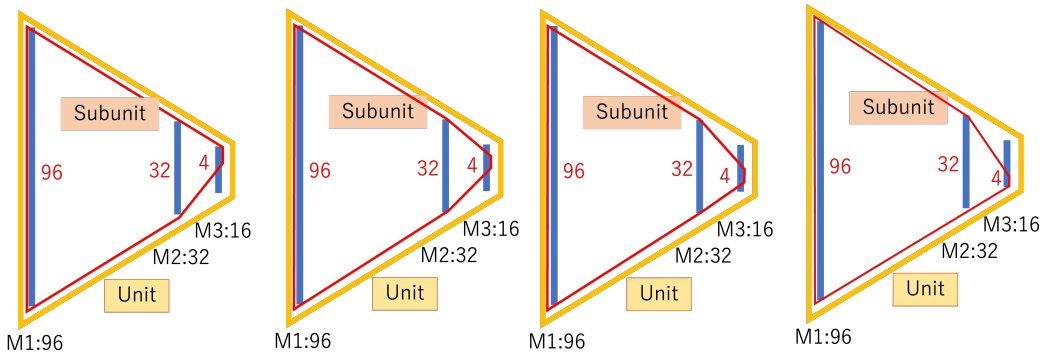

図 3.7: ワイヤーにおける unit/subunit 構造（[12] から引用）。

3.3.1 Wire Segment Reconstruction（UnitReconstructor）

Wire Segment Reconstruction[*9]はワイヤーの代表点の組に対して直線飛跡かどうかの判定を行い、直線飛跡である場合には $d\theta$ を出力する。ここで unit、subunit という区画を導入する。subunit は連続した M3 上の代表点 4 つと、それらと位置的に対応する M1 96 点と M2 32 点の領域で構成される代表点の組み合わせを指す。unit は subunit 4 つにより構成される領域である。トリガーセクターの片側のエンドキャップ領域では M3 の代表点は 579 個あるので、対応する Unit 数は 37 個あり、フォワード領域では M3 の代表点は 243 個あるので、対応する unit は 16 個ある。[*10]具体的な代表点と subunit、unit の対応関係はテーブル [14] にまとまっている。1 つの unit に対応する各ステーションの代表点の個数を示したのが、図 3.7 である。内側のステーションほど unit に含まれる代表点が多くなるのは、横方向運動量の閾値の最低値として 5 GeV のミューオンを想定しており、それら低運動量ミューオンの飛跡に対してのアクセプタンスを確保するためである。1 つの subunit に対してヒット層数の多い順に選び、最大 1 つの $d\theta$ を再構成し出力する。1 つの subunit 単位で、$d\theta$ の再構成には、Address Specifier（RAMaddressGenerator）、Segment Extractor、Segment Selector というサブモジュールによって行われ、以下でそれぞれについて詳述する。

[*9] ファームウェア上では、TGCWireReconstructer_XX の内部モジュールである UnitReconstructor_X である。
[*10] 最後尾の Unit は余りが発生して、最初の subunit のみ代表点が存在しており、残りの 3 つの subunit に対応する代表点は無い。

Address Specifier[*11]は代表点の組からアドレスを生成する。1 つの subunit に対応する代表点は M1 が 96 個（<7 bit)、M2 が 32 個（=5 bit)、M3 が 4 個（=2 bit）であり、M1 の代表点番号のみ下位 2 bit を潰して、アドレスは{M1 代表点番号 [6:2], M2 代表点番号 [4:0], M3 代表点番号 [1:0]}の 12bit となる。RAM アドレスと subunit の対応関係は [15] にまとまっている。代表点の組み合わせの数だけアドレスは存在しうるが、固定レイテンシーという条件を満たすにはアクセスする回数が制限されるので、そのうち 8 つを選抜して後段に送る。その優先順位は、M1/2/3 の計 7 層のうちヒットがあった層数が多い組み合わせほど高く、表 3.1 で上位ほど優先度が高い。ヒットがあった層数が同じ場合は subunit 中心により近い代表点の組み合わせのアドレスが優先される。

表 3.1: Wire Station Coincidence におけるヒットパターンの優先順位

Hit Pattern	M1	M2	M3
7/7	3/3	2/2	2/2
6/7A	2/3	2/2	2/2
6/7B	3/3	1/2	2/2
6/7C	3/3	2/2	1/2
5/7A	2/3	1/2	2/2
5/7B	3/3	1/2	1/2
5/7C	3/3	2/2	0/2
5/7D	1/3	2/2	2/2

Segment Extractor は生成したアドレスに元に対応するデータ（segment）を LUT から取り出し、$d\theta$ を出力する。LUT は FPGA の Ultra RAM（URAM）に格納されている。M1 の代表点番号の下位 2 bit を潰してアドレスを生成したため、その M1 の下位 2 bit（4 表現）の分、1 つのアドレスに対して、4 パターンの代表点の組み合わせに対応する segment 情報が入っている。4 パターンの代表点の組み合わせは M1 の下位 2 bit の値が小さいものほど、上位 bit に配置する。つまり、1 アドレスの格納データは{下位 2 bit=0 の segment, 下位 2 bit=1 の segment, 下位 2 bit=2 の segment, 下位 2 bit=3 の segment}である。それぞれの segment は{フラッグ 1 bit, 0, M3 代表点番号 4 bit, $d\theta$ 9 bit, 横方向運動量 4 bit}という 18 bit[*12]で表される。一つのアドレスに入っているデータは 18 bit×4=72 bit であり、これは URAM が格納できる最大値である。最大で $d\theta$ は 4×8 個だけ出力される

Segment Selector は最大 32 個の $d\theta$ のうち、各 Unit で 1 つ選びだす。その優先順位はヒット信号のチェンバーの層数が多さが多いものほど優先され、層数が同じ場合は $d\theta$ が小さいものを選ぶ。加えて $d\theta$ が同じ場合はアドレスの生成順序が早いものを選ぶ。

3.3.2　Strip Segment Reconstruction（StripPatternMatching）

Strip Segment Reconstruction[*13]はストリップの代表点の組に対して直線飛跡かどうかの判定を行い、直線飛跡である場合には $d\phi$ を出力する。ここで Unit、subunit という区画を導入する。subunit は M3 代表点 8 つとそれに対応する M1/M2 の代表点により構成される領域で、unit は subunit2 つにより構成される領域である。M3 の代表点は M3 のチェンバーあたり 63 個あるので、対応する Unit 数はチェンバーあたり 4 個ある[*14]。$\phi 0$ と $\phi 1$ のうちの片側のエンドキャップ領域でチェンバーは 5 枚、フォワード領域で 1 枚あるので、トリガーセクターあたり

[*11] ファームウェア上では RAMaddressGenerator である。
[*12] このうち、後段で利用されるのは $d\theta$ のみで、他は不必要である。この無駄なデータを省略し、表現できる代表点の組を低運動量側に広げるための研究が進んでいる。
[*13] ファームウェア上では、StripPatternMatching である。
[*14] 最後尾の Unit は余りが発生して、最初の subunit のみ代表点が存在しており、残りの 3 つの subunit に対応する代表点は無い。

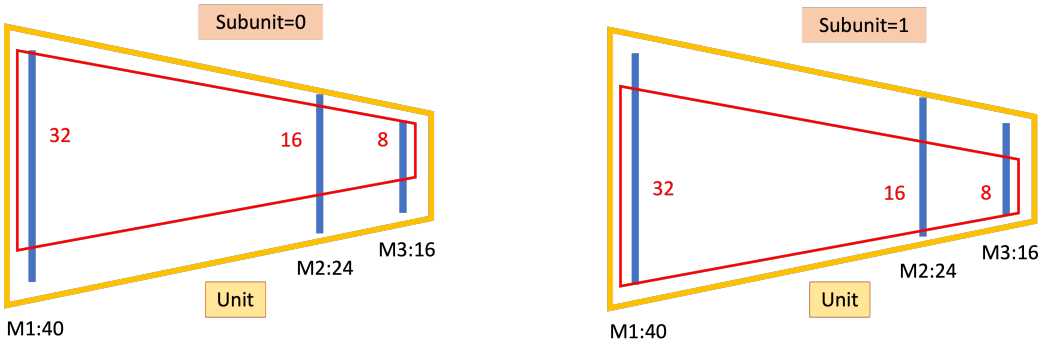

図 3.8: ストリップにおける unit/subunit 構造（[12] から引用）。

unit は 4×5+4×5+4×1=41 個ある。具体的な代表点と subunit,unit の対応関係はテーブル [14] にまとまっている。1 つの unit に対応する各ステーションの代表点の個数を示したのが、図 3.8 である。内側のステーションほど Unit に含まれる代表点が多くなるのは、曲がった飛跡を取りこぼさないためである。strip segment reconstruction は 160 MHz の 19 tick で駆動し、レイテンシーは 118.75 ns である。1 つの unit に対して、ヒット層数が多い順に選び、最大 1 つの $d\phi$ を再構成し出力する。1 つの unit 単位で、$d\phi$ の再構成には、Address Specifier、Segment Extractor、Segment Selector（LocalSelector）というサブモジュールによって行われ、以下でそれぞれについて詳述する。

Address Specifier は subunit に一つ存在し、代表点の組からアドレスを生成する。1 つの subunit に対応する代表点は M1 が 32 個（=5 bit）、M2 が 16 個（=4 bit）、M3 が 8 個（=3 bit）であり、M1 の代表点番号の下位 2 bit と M2 の代表点番号の下位 1 bit を潰してアドレスは{subunit 番号 [0:0], M1 代表点番号 [4:2], M2 代表点番号 [3:1], M3 代表点番号 [2:0]}の 10bit となる。RAM アドレスと subunit の対応関係は [15] にまとまっている。代表点の組み合わせの数だけアドレスは存在しうるが、固定レイテンシーという条件を満たすには LUT にアクセスする回数が制限されるので、そのうち 4 つ[*15]を選抜して後段に送る。その優先順位は、M1/2/3 の計 6 層のうちヒットがあった層数が多い組み合わせほど高く、表 3.2 で上位ほど優先度が高い。ヒットがあった層数が同じ場合は subunit 中心により近い代表点の組み合わせのアドレスが優先される。

表 3.2: Strip Station Coincidence におけるヒットパターンの優先順位

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

Segment Extractor は unit に一つ存在し、subunit 二つで生成したアドレスを元に対応するデータ（segment）を LUT から取り出し、$d\phi$ を出力する。LUT は FPGA の Ultra RAM（URAM）に格納されている。M1 の代表点

[*15] Strip Segment Reconstruction のクロックドメインを 160 MHz にした変更に伴って、LUT へのアクセス回数は 6 回から 4 回に変更になった。

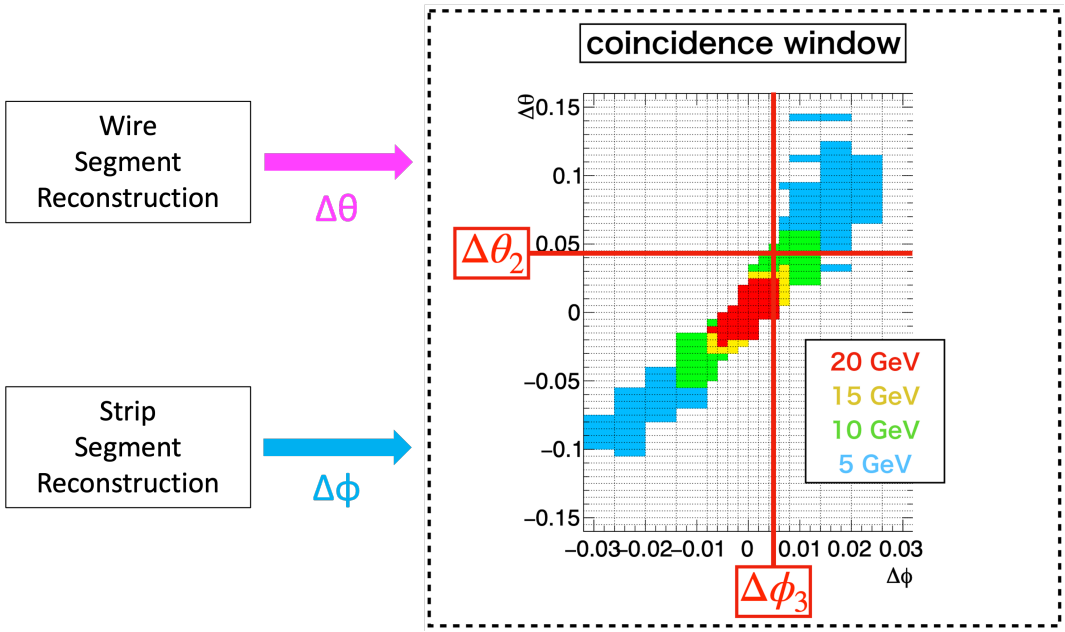

図 3.9: Coincidence Window を使った横方向運動量再構成の概念図（[13] から引用し一部修正）。

番号の下位 2 bit、M2 の下位 1 bit を潰してアドレスを生成したため、その M1 の下位 2 bit と M2 の下位 1 bit の全通り（8 表現）の分、1 つのアドレスに対して、8 パターンの代表点の組み合わせに対応する segment 情報が入っている。8 パターンの代表点の組み合わせは M2 の下位 1,bit の値が 0 であるならば、下位 bit 半分になり、1 ならば上位 bit 半分となる。そのうち、M1 の下位 2 bit の値が大きいものほど、上位 bit に配置する。つまり、1 アドレスの格納データは{M2 下位 1 bit=1 で M1 下位 2 bit=3 の segment, M2 下位 1 bit=1 で M1 下位 2 bit=2 の segment, ... , M2 下位 1 bit=0 で M1 下位 2 bit=0 の segment}である。それぞれの segment は $d\phi$ の情報 9 bit が格納されている。一つのアドレスに入っているデータは 9 bit×8=72 bit であり、これは URAM が格納できる最大値である。1 つの unit では最大で $d\phi$ は 8×4×2 個だけ出力される。

Segment Selector[*16]は unit に一つ存在し、最大 64 個の $d\phi$ のうち、1 つ選びだす。その優先順位はヒット信号のチェンバーの層数が多さが多いものほど優先され、層数が同じ場合は $d\phi$ が小さいものを選ぶ。加えて $d\phi$ が同じ場合はアドレスの生成順序が早いものを選ぶ。

3.4　Wire Strip Coincidece

Wire Strip Coincidence では Segment Reconstruction で出力された ($d\theta$,$d\phi$) からそれに対応する横方向運動量の閾値（p_{T} threshold）とその座標を出力する。p_{T} threshold の出力はあらかじめシミュレーションによって出力した対応関係のテーブルである LUT を利用する。p_{T} threshold を出力する LUT は Coincidence Window といい、図 3.9 がその例である。Wire Strip Coincidence では、Region は 8 Unit Region と 32 Unit Region の 2 種類の Region で区切られる。8 Unit Region は wire subunit 2 つ ×strip unit 4 つ =8 つという segment の組み合わせからなり、32 Unit Region は wire subunit 8 つ ×strip unit 4 つ =32 個という segment の組み合わせからなる。図 3.10 で示すように、エンドキャップ ϕ0、ϕ1 のそれぞれは $|\eta| < 1.3$ の領域では 8 Unit Region 22 個がカバーし、$|\eta| > 1.3$ の領域では 32 Unit Region 13 個がカバーする。フォワード領域は 32 Unit Region8 個がカバーする。8 Unit Region では最大 8 個の飛跡候補が想定できるが、そのうちから 1 個の飛跡候補を出力し、32 Unit Region で

[*16] ファームウェア上では LocalSelector である。

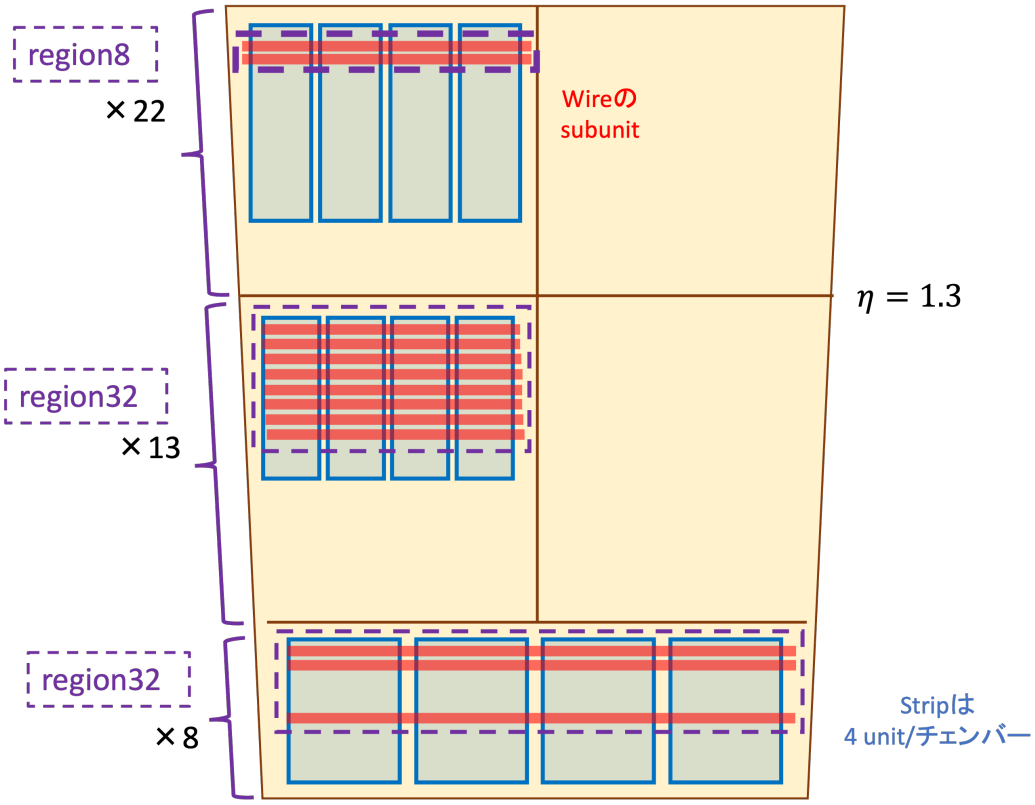

図 3.10: 8/32 Unit Region の配置（[12] から引用）。

は最大 32 個の飛跡候補が想定できるが、そのうち 4 個の飛跡候補を出力する。それぞれの領域では strip segment はチェンバー単位で出力されるため、チェンバー境界に位置する 8/32 Unit Region では 2 枚のチェンバーからの strip segment を受け取る。Wire Strip Coincidence は 160 MHz の 6 tick で駆動し、レイテンシーは 37.5 ns である。8 Unit Rrgion/32 Unit Region 単位で、p_{T} calculator、Wire Position Corrector、Block Selector、Duplicate Selector というサブモジュールで構成される。以下でそれらのサブモジュールについて、詳述する。

p_{T} calculator は wire subunit に一つ存在し、($d\theta$,$d\phi$) からアドレスを生成し、そのアドレスの対応する p_{T} threshold を出力する。アドレス生成のためにまず、8 bit の $d\theta$ の下位 1 bit を潰して 7 bit にし、9 bit の $d\phi$ は特定の規則に従って、4 bit にする。後者は具体的には符号を意味する MSB を除いた 8 bit を表 3.3 に従って 3 bit に変換し、その後 MSB を復元する。これは $d\phi$ の大きさが小さいところで分解能を落とさないようにしながら圧縮するという思想である。アドレスは strip segment4 つ分の 4 通り出力され、{strip unit 番号 [1:0], $d\theta$[6:0], $d\phi$[3:0]} である。p_{T} threshold と ($d\theta$,$d\phi$) の対応関係を記した LUT は BRAM に書き込まれ、1 つのアドレスに対して一つの p_{T} threshold を出力するので、p_{T} calculator 単位では strip unit4 つ分の 4 通りの p_{T} threshold を出力する。

Wire Position Corrector はワイヤーとストリップの代表点番号から η 座標に変換し、飛跡候補に座標情報を付加する。図 3.11 に示すように、ワイヤーの代表点番号が同一であっても、ストリップの位置によっては η 座標が異なる。これを補正するため、ワイヤーとストリップの代表点番号と η 座標の対応関係を記した LUT を参照する。アドレスは{ワイヤー代表点番号 [1:0], ストリップ代表点番号 [5:0]}の 8 bit である。

Block Selector は 8/32 Unit Region に一つ存在し、飛跡候補を選ぶ。その優先順位はヒットのあった層数が多い順であり、層数が同じ場合は ($d\theta$,$d\phi$) が小さいものを選ぶ。8 Unit Region では、最高 8 個の segment の組み合わせから飛跡候補を 1 つ選ぶ。32 Unit Region では、最高 32 個の segment の組み合わせから、$d\theta$ が正のもの 2 つと $d\theta$ が負のもの 2 つ、合計 4 つ選ぶ。この実装は、高運動量親粒子がミューオン対に崩壊し、近傍に残される異符号

表 3.3: Wire Strip Coincidence における $d\phi$ の変換則

変換前 $d\phi$[7:0]	変換後 $d\phi$[2:0]
0-3	0-3 (そのまま)
4-6	4
7-9	5
10-12	6
13-	7

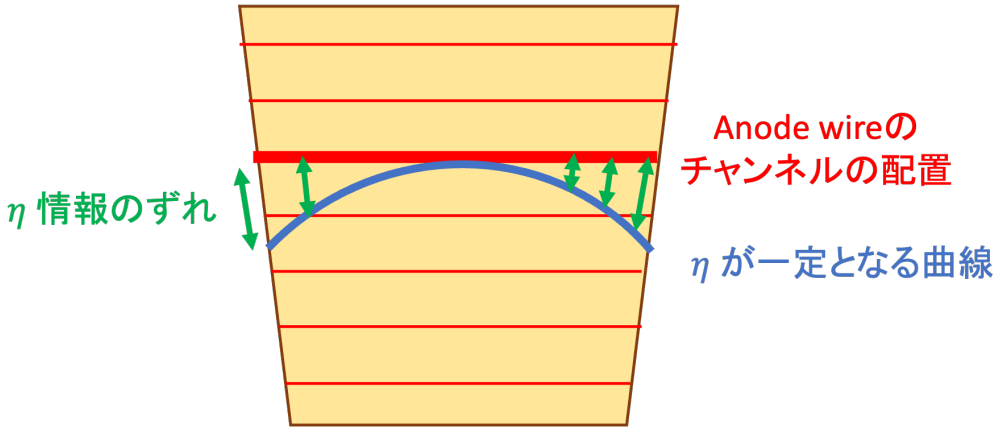

図 3.11: wire channel と実際の η 座標のズレ（[12] から引用）。

ミューオン飛跡を優先的に選択するためのものである。

Duplicate Selector はチェンバー境界に位置する Region に対して、2 枚のチェンバーからの処理を並行して行い、どちらの出力を選ぶかを決める。2 つのチェンバーからの strip segment をヒットがあった層数が多い方を選び、層数が同じ場合は $d\phi$ が小さいものを選ぶ。

3.5　Inner Coincidece

Inner Coincidence は TGC BW で再構成した飛跡候補がトロイド磁場内部の検出器のヒット信号とコインシデンスをとり、衝突点以外の箇所で生成された荷電粒子であるフェイクミューオンを除外する。その概要を図 3.12 に示す。また、多重散乱によってミューオン飛跡が折れ曲がる場合に対して、それを補正するロジックにより、運動量分解能の低下を抑制する。コインシデンスをとるトロイド磁場内部の検出器は NSW、TGC EI、RPC BIS7/8、Tile カロリメータである。これらの検出器がカバーする領域は図 3.13 の通りである。8 Unit Region では、TGC BW で出力する飛跡候補 1 つを受け取り、Region ごとに決まっている内部検出器とコインシデンスを取ったのちに飛跡候補 1 つを出力する。32 Unit Region では、TGC BW で出力する飛跡候補 4 つを受け取り、内部検出器とコインシデンスを取ったのちに飛跡候補 2 つを出力する。図 3.14 のように、Inner Coincidence は Decoder、NSW/EI/Tile/RPC Coincidence、Which Inner というサブモジュールで構成されている。

各内部検出器とのコインシデンスロジックは検出器ごとに異なっており、現在共同研究者によって実装・検証中である。具体例として、開発が進んでいる NSW のコインシデンスロジックについて詳述する。NSW と TGC BW で再構成したミューオン飛跡の η 座標をそれぞれ η_{NSW}、η_{TGC} として、その差 $d\eta$ を

$$d\eta = \eta_{\mathrm{TGC}} - \eta_{\mathrm{NSW}} \tag{3.8}$$

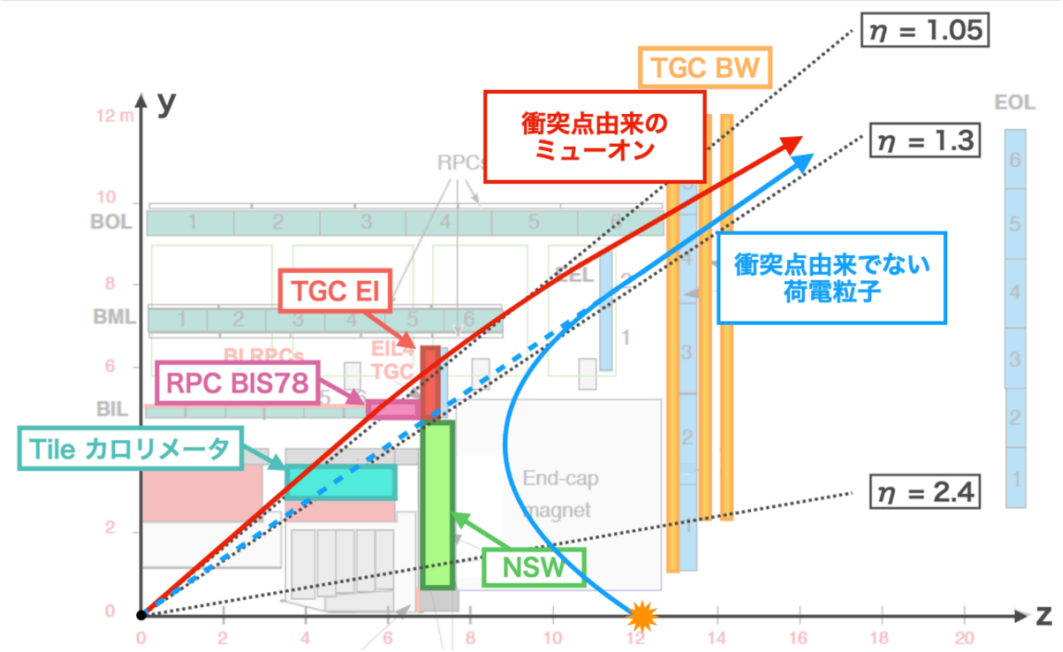

図 3.12: Inner Coincidence の概要図（[16] から引用）。NSW や RPC BIS7/8、Tile カロリメータ、TGC EI などの内部飛跡検出器と TGC BW のコインシデンスをとって、運動量測定の精度の向上とフェイクミューオンの除外を狙う。

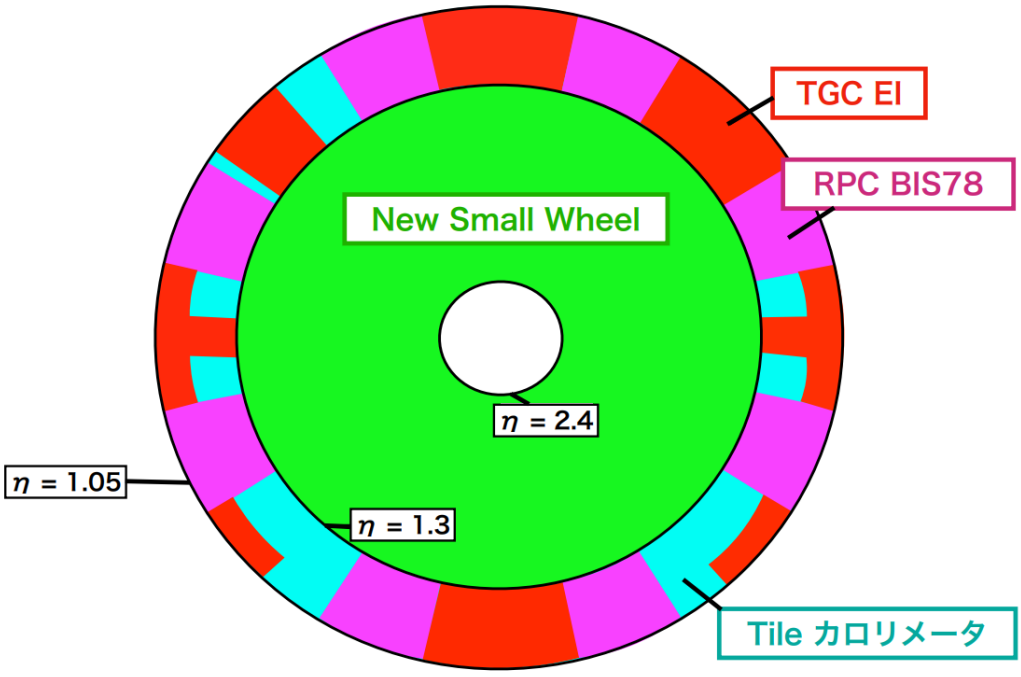

図 3.13: 磁場内部の検出器がカバーする領域（[13] から引用）。

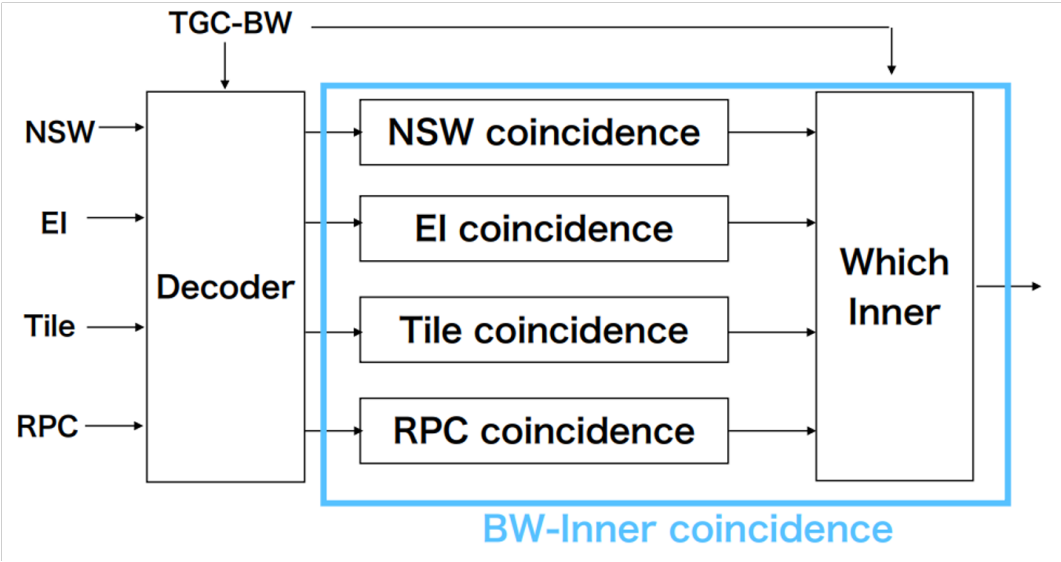

図 3.14: 磁場内部の検出器と TGC BW のコインシデンスロジックの全体像（[17] から引用）。

と定義する。$d\eta$ はトロイド磁場による曲率のため、横方向運動量が大きいほど小さくなる。しかし、NSW より内側の領域でミューオンが多重散乱を起こして飛跡が折れ曲がった場合（図 3.15）、$d\eta$ の測定による横運動量分解能は低下する。これを補正するために、多重散乱により折れ曲がりの角度と相関のある NSW への入射角度 $\Delta\theta_{\mathrm{NSW}}$ を利用する。$\Delta\theta_{\mathrm{NSW}}$ と $d\eta$ に対する横方向運動量の対応関係を記した LUT によって、多重散乱に堅牢な運動量測定が可能となる。

Decoder では NSW から受け取った最大 16 個の飛跡候補の中から、$|d\eta|$ が小さいものを優先的に 4 つ選ぶ。NSW Coincidence では、TGC BW の飛跡候補 1 つと NSW の飛跡候補 4 つの組み合わせでコインシデンスをとって、$\Delta\theta_{\mathrm{NSW}}$ と $d\eta$ から横方向運動量 4 bit を出力する。そしてその 4 つの飛跡候補から横方向運動量が大きいもの 1 つ選ぶ。Which Inner では、複数の内部検出器とのコインシデンスを取った飛跡候補において、いずれの検出器とコインシデンスを取るかを選択する。

3.6　Track Selector

Track Selector は再構成した飛跡候補を p_{T} threshold の大きい順に並び替える。飛跡候補を 2 つ出す 32 Unit Region が 34 個、飛跡候補を 1 つ出す 8 Unit Region が 44 個あるので飛跡候補は全部で最大 112 個あり、これら全てを p_{T} threshold が高い順位並び替えるために図 3.16 で示すようなソーティングロジックで実装されている。[*17]加えて、一部のチャンバー領域のオーバラップによる飛跡再構成の重複を補正するロジックが Track Selector の中間論理に実装される予定である。これは、エンドキャップ $\phi 0$、$\phi 1$、フォワードを構成するチェンバーはそれぞれの境界でオーバラップが存在し、理想的に再構成された場合には一つの飛跡に対して複数の飛跡候補を出力してしまうことによる。最終的に p_{T} threshold の大きな 6 つの飛跡候補を MUCTPI に送るものとして選ぶ。また、160 MHz の 5 tick で駆動し、レイテンシーは 31.25 ns である。

[*17] Batcher の奇遇マージソート法を論理回路として実装した形になっている。

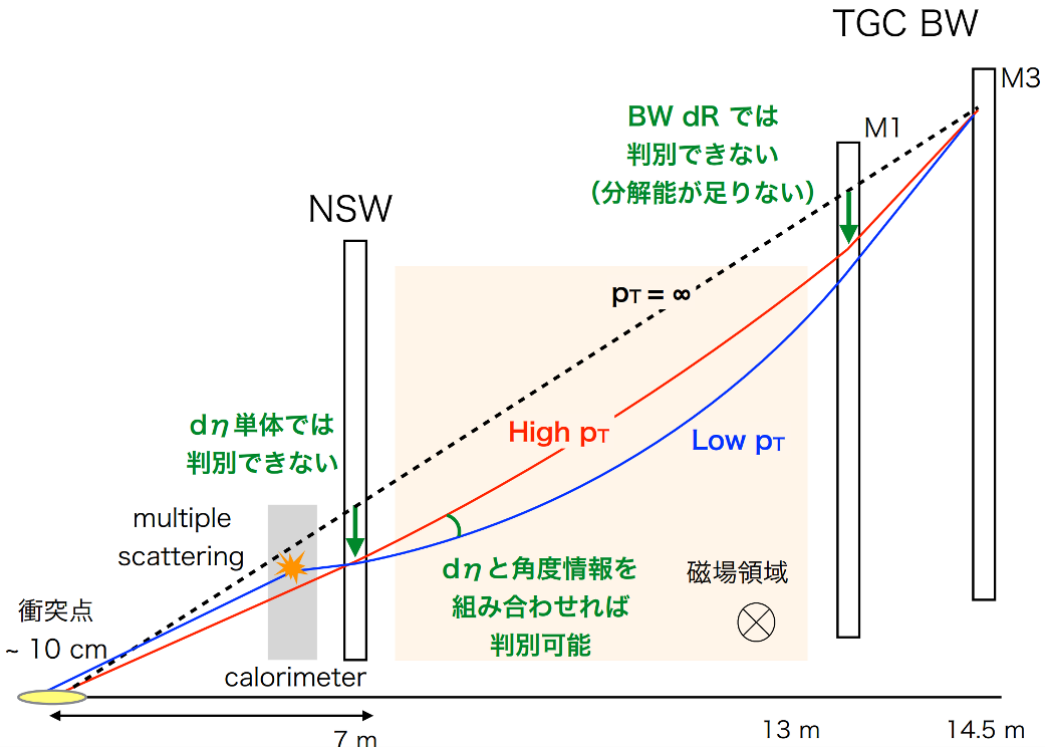

図 3.15: 磁場内部で多重散乱を起こしたミューオン飛跡（[11] から引用）。

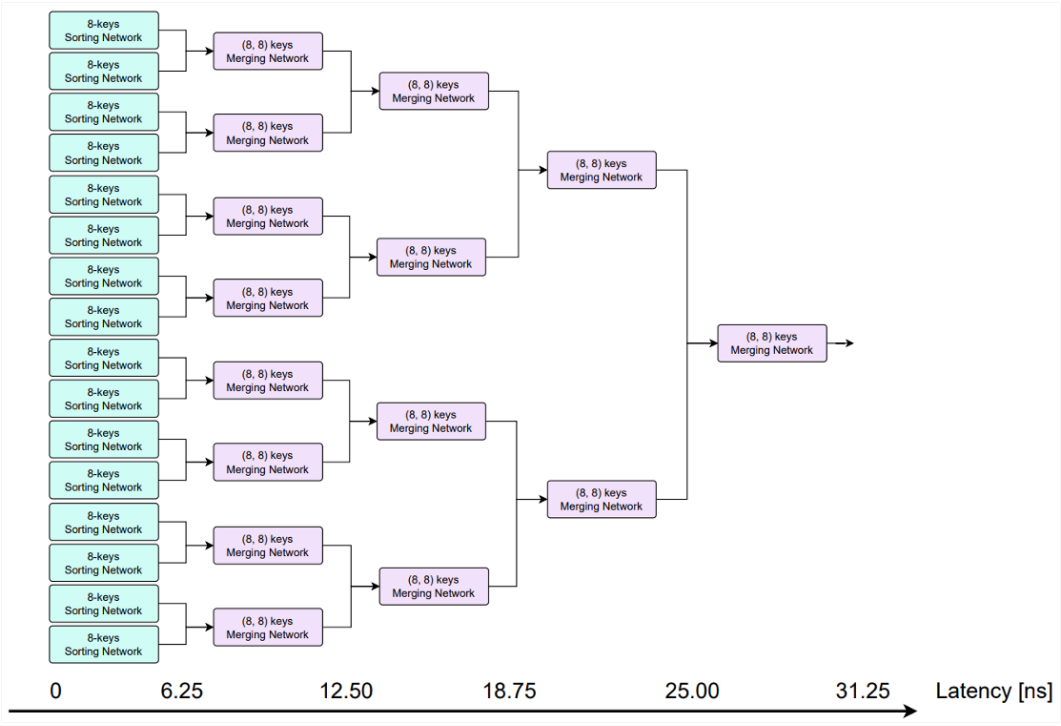

図 3.16: Track Selector の概要（[9] から引用）。

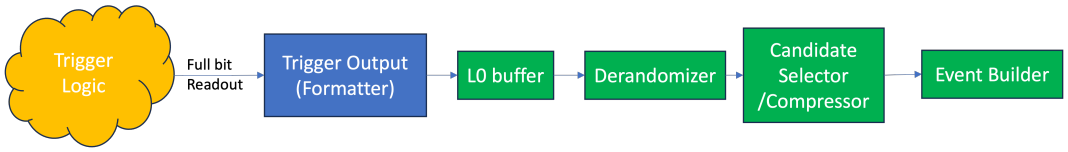

図 3.17: トリガー論理回路の読み出し回路。

3.7　トリガー論理回路のコンフィギュレーションと読み出し

ここではトリガー論理回路のコンフィギュレーションを担うモジュールと読み出し回路について説明する。前者は Segment Reconstruction と Wire Strip Coincidence の URAM/BRAM のコンフィギュレーションを行うモジュールは LUT Manager である。以下でそれぞれについて説明する。

3.7.1　LUT Manager

LUT Manager はそれぞれの URAM/BRAM に LUT のデータを書き込むモジュールである。MPSoC から SL FPGA のレジスタアクセスにより LUT のデータを書き込むパスと Simulation のテストベンチから LUT データに対応する信号をエミュレートするパスがある。データの分配は駆動クロック 160 MHz で駆動する一方、MPSoC からの書き込みはオンボード発振器からのクロック 50 MHz で駆動するため、クロックドメイン境界を跨ぐ仕掛けが実装されている。

3.7.2　トリガー読み出し回路

トリガー論理回路の読み出し回路の全体像を図 3.17 に示す。この構造は複数のトリガー論理回路の中間出力に対して並列に存在する。ここでは、実装と検証が完了している、Channel Mapping から Wire Strip Coincidence におけるトリガー読み出しについて述べる。

L0 Buffer はトリガー論理回路の出力を L0 判定までバッファする。深さ 512 列の BRAM で実装され、40 MHz の LHC クロックで駆動する書き込みポインタに従って、トリガー論理回路週間出力が書き込まれる。読み出しは 240 MHz で行われ、Derandomizer に送られる。

Derandomizer は後段の Candidate Selector/Compressor での処理を待つためのバッファである。読み出し命令を受け取るとバッファしている信号を後段の Candidate Selector/Compressor に送る。

Candidate Selector/Compressor は Derandomizer から受け取った信号の圧縮処理を行う。Channel Mapping および Station Coincidence の出力は Compressor によって、Segment Reconstruction と Wire Strip Coincidence の出力は Candidate Selector によって圧縮される。前者はチャンネル・代表点を一定長の bit ごとのセルに分割し発火チャンネルがあったセルの情報のみを、後者は飛跡再構成できた unit/region 情報のみを取り出し、後段に送る。

Event Builder は Candidate Selector/Compressor から受けたデータを読み出し用のフォーマットに成形する。このフォーマットについては現在はデバッグ用の仮実装になっており、本番用フォーマットとその実装が予定されており、詳細は 6.2 節で議論する。

第 4 章

トリガー論理回路検証システムの構築

先行研究 [9] により、第 3 章で示したようにトリガー論理回路の統合が完了している。全ての論理回路が実装された統合論理回路は大規模であり、大統計のシミュレーションデータを用いたトリガー性能評価に基づく検証は必要不可欠である。この検証により、開発したトリガー論理回路が実際の運用に対する要求を満たす実装になっていることを保証することができる。また、設計思想から外れて性能が落ちるケースがあった場合には、不具合箇所を精密に探知して改善するということが求められる。さらに、すでに既知の不具合が見られており、めトリガー性能が大きく劣化していた（図 5.4 を参照）ため、その不具合の特定が必要であった。以上の目的から本研究では、トリガー論理回路の動作を検証し、不具合箇所を特定するための検証システムを構築した。図 4.1 にトリガー論理回路の開発状況を示しており、本検証システムは Channel Mapping から Wire Strip Coincidecen までの統合回路を対象とする。

4.1　検証システムの全体像

検証システムの全体像を図 4.2 に示す。開発されたテストパターン生成システムにより、大統計 MC データからハードウェアのコンフィギュレーションに用いる形式（coe ファイル形式）のテストパターンを生成する。テストパターンは Bitwise Simulator、Xilinx Vivado Simulator、Sector Logic 実機試験の三つの試験システムに共通に入力できる。Bitwise Simulator はトリガー論理回路を再現した C++ のプログラムであり、bit 単位で論理回路の中間出力を高速でシミュレートできる。Xilinx Vivado Simulator は、Xilinx が提供するファームウェアシミュレーションツール（Vivado Simulation）で実装した、テストパターンを入力できるテストベンチである。Sector Logic 実機試験は実機単体で入力テストパターンからテストパルスをエミュレートし、トリガー論理回路に入力してその出力を読み出すシステムで、実装の工程を終えたファームウェアの最終動作試験として役割を持つ。Software Simulator はトリガーアルゴリズムを実装した C++ プログラムであり、トリガー論理回路の開発に先行してアルゴリズム自体の性能評価を行うシミュレーターである。Bitwise Simulator は高速でトリガー論理回路の挙動をシミュレートする

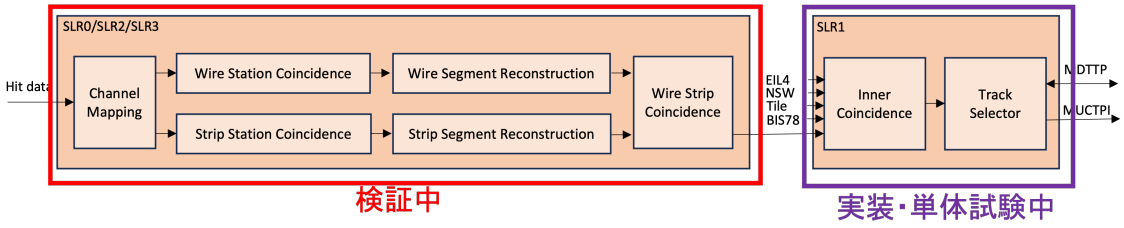

図 4.1: トリガー論理回路の開発状況。

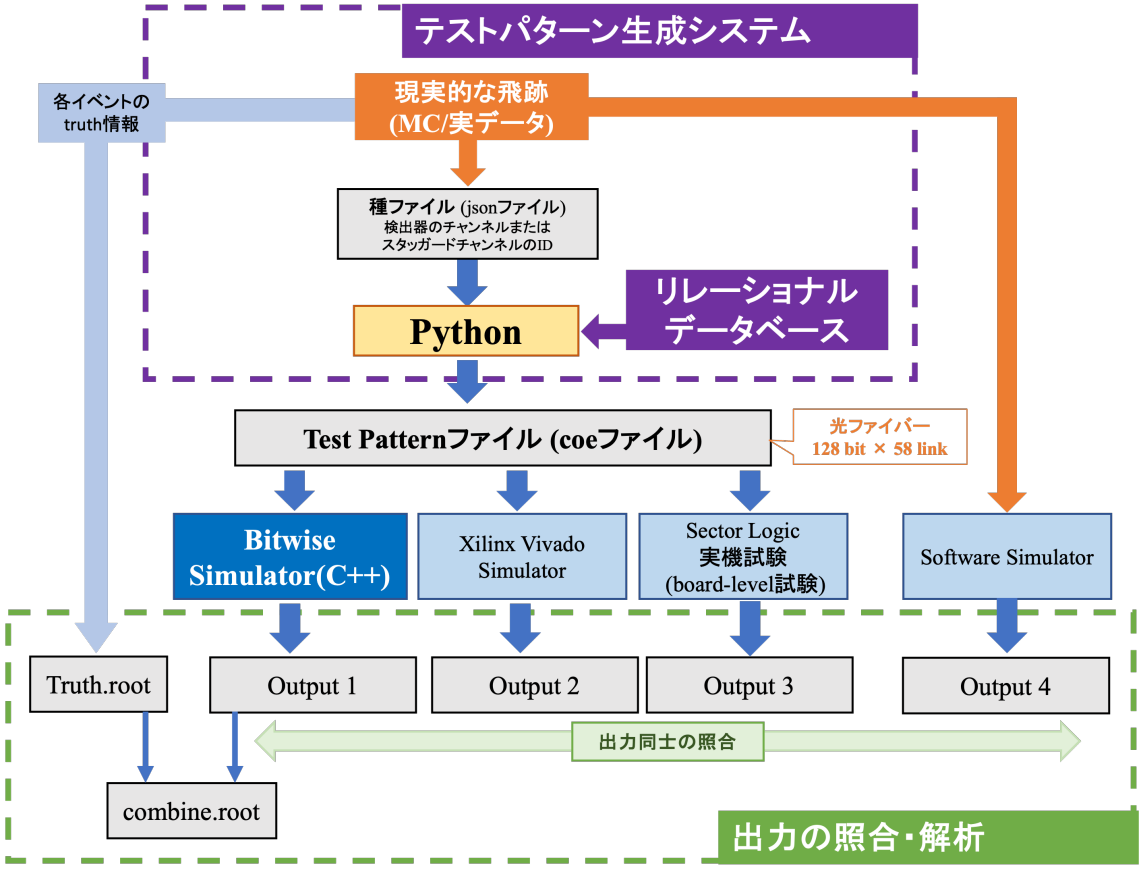

図 4.2: トリガー論理回路の検証システムの全体像（[12] から引用して一部修正）。

ことが可能で、任意の信号線をプローブして詳細なファームウェアの挙動を調査できる Vivado Simulator と bit レベルで比較することで不具合の特定の高速化が可能である。

4.2　テストパターン生成

テストパターン生成の工程では MC データ等を入力として、検証に用いるすべてのシステムに共通して入力可能なテストパターンを生成する。詳細の説明は開発を行った先行研究 [12] に譲り、ここでは簡単な説明に留める。

MC データから設定ファイルに記載したカット条件に応じてイベントを選択し、MC データに格納された TGC ヒットデータ情報を中間情報として json 形式の種ファイル（図 4.2 中）を出力する。種ファイルのフォーマットは図 4.3 の構造となっており、イベント[*1]ごとにそのヒットデータ情報が順に格納される。ヒットデータ情報は発火チャンネルの位置を特定できる情報である。その種ファイルを元に、配線情報がまとめられたリレーショナルデータベースを参照して coe 形式のファイルのテストパターンを生成する。テストパターンはヘッダーを除いて 1 行あたり 32 文字の 16 進数が記載されたテキストファイルとなっていて、行はイベントを表す。32 文字の 16 進数（=128 bit）は特定の配線での 128 チャンネルの発火を表している。一つのトリガーセクターでは 58 個のテキストファイルで全てのチャンネルをカバーするテストパターン 1 セットとなる。配線情報は [14] のテーブルにまとまっており、具体的な配線情報を復元方法は付録 A で説明する。検証システムでは MC データのシングルミューオンイベントを用いてテストパターンを生成し、Bitwise Simulator、Xilinx Vivado Simulator、SL 実機試験に共通の

[*1] イベントとはバンチ交差に対応する。

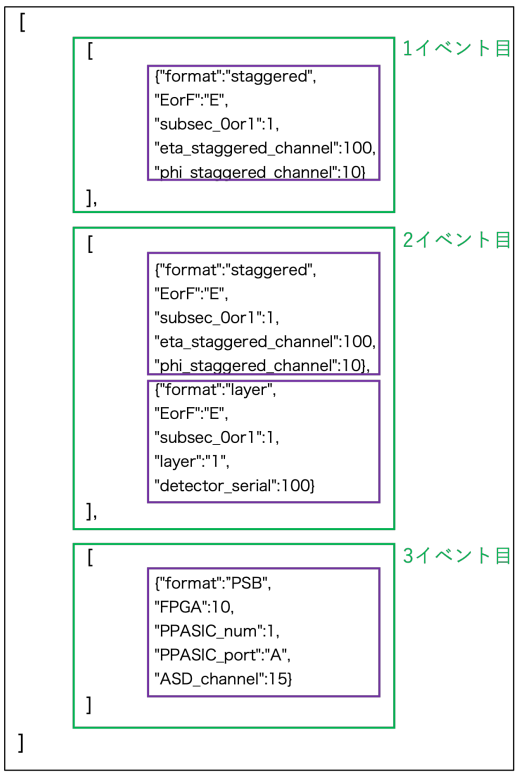

図 4.3: テストパターン生成システムの中間情報の構造（[12] から引用）。イベント毎の発火チャンネルの情報が格納されている。図は発火チャンネルの種類を例示しており、実際には発火チャンネルの数だけ羅列される。

テストパターンを入力するのに用いた。

4.3 Bitwise Simulator

Bitwise Simulator は先行研究 [12] により開発された bit 単位でファームウェアの動作をシミュレートできる C++ プログラムである。入力テストパターンからトリガー論理回路、出力の読み出し回路までファームウェアの動作をトレースしているため、相互の動作を精密に比較できる。Vivado Simulation というファームウェアのシミュレーションツールよりも高速で動作し、10000 event の MC データの入力に対して、数分で結果が得られる。設定ファイルによって様々なデバッグフラッグが用意されており、トリガー論理回路の中間出力が容易にプローブできる機能を備えている。

検証システムにおいては、Vivado Simulator と bit 単位での比較により不具合箇所を精細に特定するのに利用した。特にそのデバッグ表示機能は複雑なトリガー論理回路の挙動を確認するに有用であり、デバッグ表示機能の確認方法は付録 B で説明する。

4.4 Xilinx Vivado Simulator

Xilinx Vivado Simulator は、Xilinx の Vivado Simulation というファームウェアシミュレーションツールを使って実装した、SL 実機試験や Bitwise Simulation と共通の大統計テストパターンを入力可能なテストベンチである。図 4.4 で示すように、トリガー論理回路全体が組み込まれた形でテストベンチを実装している。テストパターンの coe ファイルをシステム関数を用いて 1 行ずつ読み取り、それに対応するテストパルスをエミュレートしてテスト

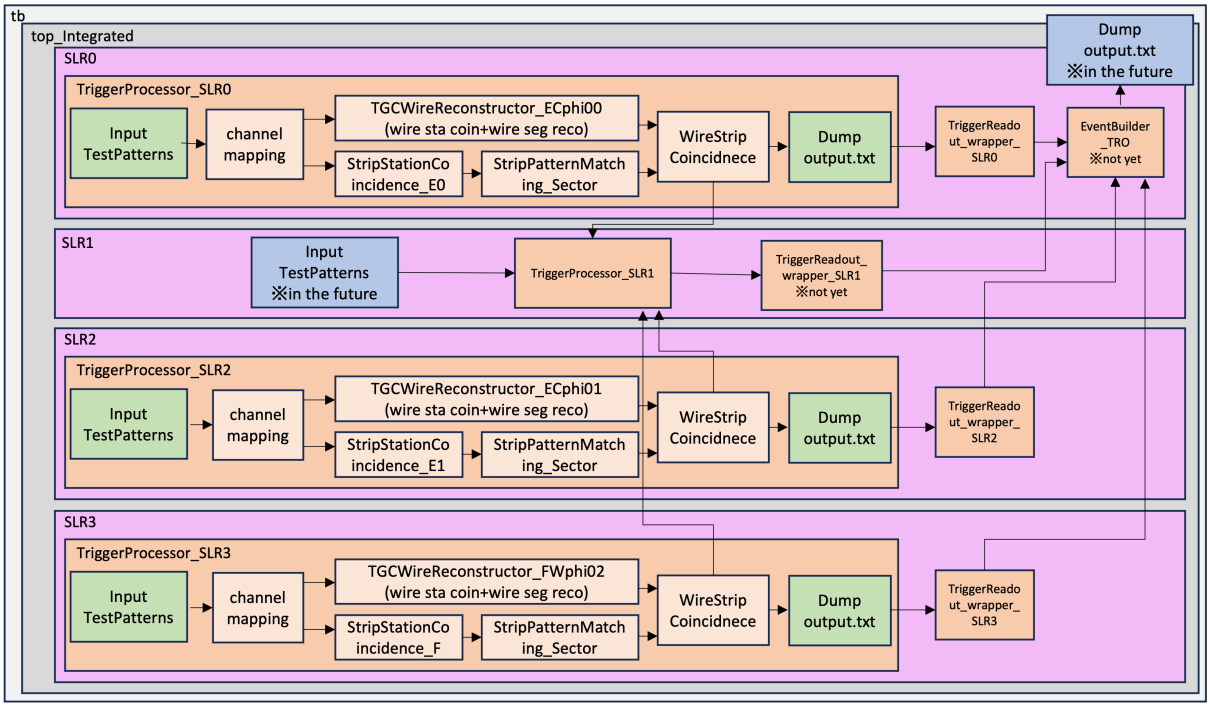

図 4.4: Vivado Simulation テストベンチ。tb がテストベンチのモジュール名である。緑のボックスが現在実装中のテストパターンを入力する機構およびトリガー中間出力の読み出し箇所である。青のボックスは磁場内部検出器のテストパターンやその読み出しであり、今後の開発によって拡張されていくべき機能である。

ベンチからトリガー論理回路に入力する。そのトリガー論理回路の中間出力は設定した固定時間だけ経過した後に読み出され、SL 実機試験や Bitwise Simulator と全く同一の形式でテキストダンプされる。

また本研究において、テストパターンをトリガー論理回路に入力する部分を大きく改善したことにより、大統計テストパターンを短時間で処理できるようになった。既存の実装では Test Pattern Generator は BRAM の深さ分の 63 events しかテストパターンの入力ができないが、Test Pattern Generator ではなくシミュレーションテストベンチ上でテストパターンの coe ファイルの 1 行ずつ読み取り、対応するテストパルスを 40 MHz 毎に入力する機能を実装した。大統計テストパターンの入力が可能になっただけでなく、Test Pattern Generator とそれを駆動させるモジュールをシミュレーションテストベンチから除外したことで、コンパイルやシミュレーション動作を最適化することができた。これにより大統計 MC でも現実的な時間でのシミュレーションが可能となり、10,000 event のプロセスが約 30 分ほどで完了する。

検証システムにおいては実機試験システムと比較して SL 実機特有の問題切り分けるのに加えて、Bitwise Simulator と bit 単位で比較してコーディングレベルの細かい不具合の探索に利用した。

4.5　SL 実機試験

SL 実機試験は SL 実機単体でテストパルス信号を入力し、トリガー論理回路で処理した後、出力を読み出す試験システムである。詳細の説明は開発を行った先行研究 [9] を参照されたい。

SL 実機試験の全体像を示したのが図 4.5 である。試験システムは以下のように動作する。まず、MPSoC から Test Pattern Generator という Virtex Ultrascale+ FPGA に実装されたモジュールにテストパターンを書き込む。Test Pattern Generator は Test Pulse Trigger（TPT）信号を受けたタイミングでテストパターンをトリガーロジッ

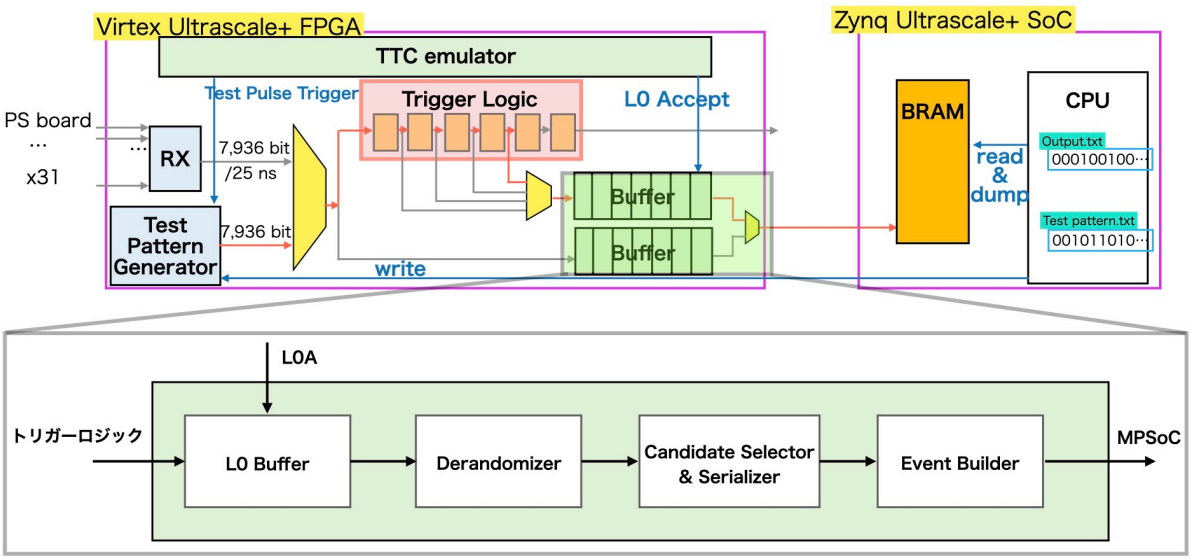

図 4.5: SL 実機試験の全体像（[9] から引用）。

クに入力する。TPT 信号は TTC emulator という Virtex UltraScale+ FPGA に実装されたモジュールによって生成される。トリガー論理回路の出力は L0 Buffer で保持し、L0A 信号を受けたタイミングで読み出されるが、このテストベンチでは L0A 信号を TTC emulator に生成させる。読み出した信号は deradomizer という処理待ち buffer で保持され、Cadidate Selector で圧縮処理されて、Event Builder でフォーマットされた後、MPSoC の BRAM に格納しテキストファイルとしてダンプする。これらの MPSoC 側の動作は MPSoC の PS で走る C++ プログラムによって制御され、10000 event のプロセスに数分かかる。

このテストベンチではリソースの観点から、全てのトリガー論理回路の中間出力を読み出すことは合理的でない。そのため、Wire Strip Coincidence、Wire Segement Reconstruction、Strip Segement Reconstruction のうち、1 種類の中間処理情報を選択して出力する別個のファームウェアとして実装している。[*2]

検証システムにおいては、Xilinx Vivado Simulator の出力と比較し、実機特有の問題の切り分けに利用した。

[*2] トリガー論理回路を含んだファームウェアではリソースの逼迫により、タイミング制約が厳しい状況にあるからである。

第 5 章

トリガー性能評価

4 章で説明した検証システムを用いてトリガー論理回路の不具合の特定と修正を行なった。任意の信号線をプローブできる複数のシステムを bit 単位で比較することで精密な不具合の特定が可能となった。また、大統計シングルミューオン MC データを用いてトリガー性能評価を行った。[*1]MC データを使用することで本番の運用に近い環境を再現して、そこでのトリガー性能を保証することができる。

5.1 検証システムを使った不具合の特定

この節では検証システムを使った検証の具体的な方法論を述べる。まず、共通の大統計テストパターンを用いた Bitwise Simulator と Xilinx Vivado Simulator の出力を同一の解析マクロを用いて、Segment Reconstruction または Wire Strip Coincidence 不一致イベントをリストアップした。次に、その不一致イベントのうちある 1 つのイベントに焦点を当て、そのイベントにおける両者のトリガー論理回路の信号線を比較した。図 5.1 のように Channel Mapping から順に後段の論理回路へ、両者の信号線を bit 単位で比較していき、ズレが生じた箇所を不具合箇所として特定し修正した。この作業をリストした不一致イベントに対して繰り返すことで多くの不具合を解消した。使用したテストパターンはパイルアップ無しのシングルミューオン MC データから生成した 8603 events で、クロストークや多重散乱を含んだイベントも存在する。

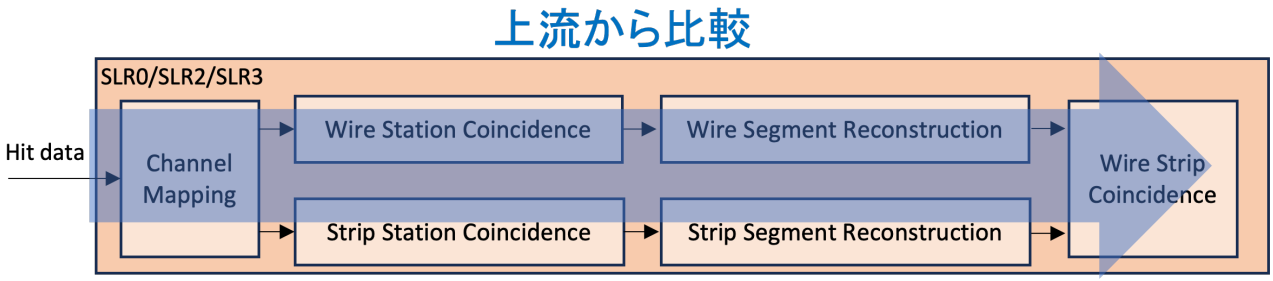

図 5.1: 特定の出力不一致イベントに対する不具合探索の流れ。

[*1] トリガー論理回路はクロックドメインを 160 MHz に統一する開発が並行して進んでおり、Strip Segment Reconstruction の駆動クロックは 240 MHz から 160 MHz に変更された。この性能評価は Strip Segment Reconstruction の駆動クロックの変更前の値である。この変更によりトリガー性能はほとんど変化しないことが先行研究で示されている。

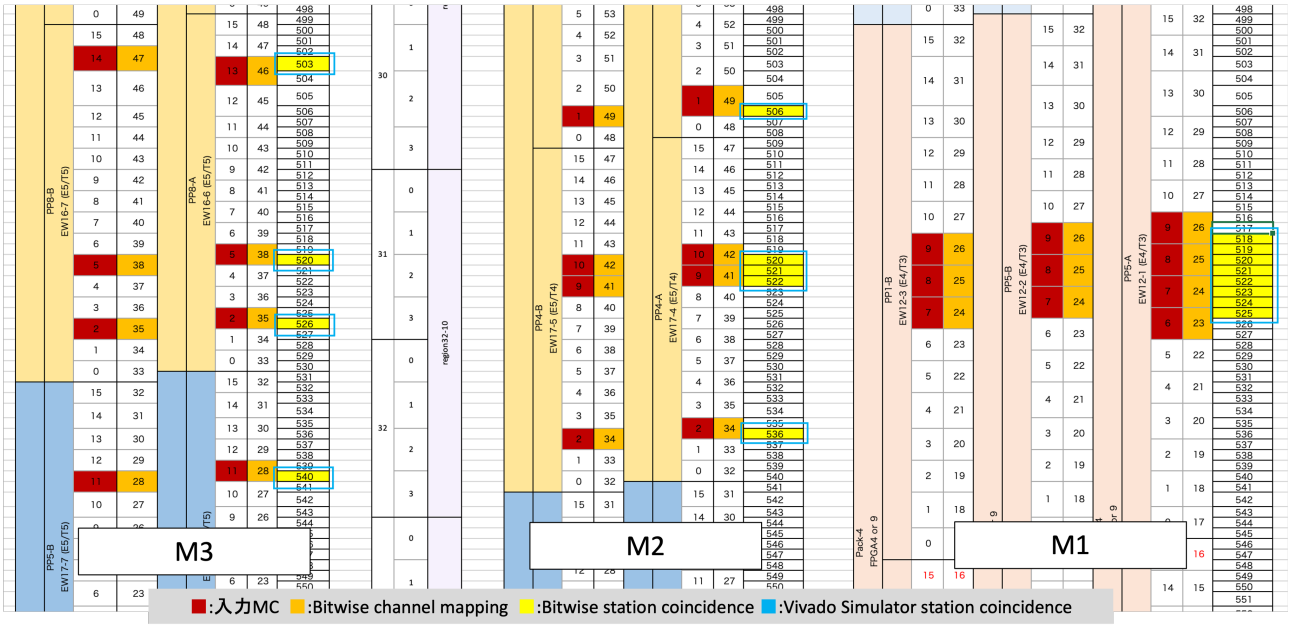

図 5.2: 特定の出力不一致イベントにおける Wire Station Coincidence での Bitwise Simulator と Vivado Simulator の出力。ワイヤ 7 層のそれぞれにチェンバー内チャンネル番号とチャンネルの通し番号、ステーションそれぞれに代表点番号が書いている。

5.1.1　Station Coincidence までの検証

Channel Mapping、Station Coincidence の入力を Bitwise Simulator と Vivado Simulator で比較した。比較には最上流である入力 MC データの中の TGC ヒットデータも利用した。ある不一致イベントにおけるそれぞれのシステムのワイヤー出力をテーブルにまとめたものを図 5.2 に示す。このテーブルは [14] からアクセスでき、TGC のチャンネルと代表点の対応関係が網羅されているものである。赤が入力 MC、橙が Bitwise Simulator の Channel Mapping、黄色が Bitwise Simulator の Station Coincidence、青が Vivado Simulator の Station Coincidence であり、いずれも整合していて Station Coincidence では不一致がないことがわかる。このイベントに限らず、Station Coincidence までででトリガー論理回路の不具合は見つかっていない。

5.1.2　Segment Reconstruction の検証

Segment Reconstruction では複数の不一致があり、その 1 例を図 5.3 に示す。この例では $d\phi$ に注目しているが、$d\phi$ を出力するまでの途中出力もすべて確認することができる。途中出力の bit 単位の比較によって Bitwise Simulator と Vivado Simulator の両者において複数箇所での不具合を特定した。ここでは、ファームウェアで見つかった不具合を紹介する。

まず、Segment Extractor における不具合である。3.3.2 節で示したように、Segment Extractor では LUT から $d\phi$ 9 bit を取り出し出力するが、その時に M3 の代表情報 6 bit を出力する。M3 代表点情報 6 bit={チェンバー内 unit 番号 2 bit, unit 内 subunit 番号 1 bit, subunit 内代表点番号 3 bit}である。この subunit 内代表点番号 3 bit を出力する際に、配線の補正の処理（付録 E に詳述）を行った後に出力するが、出力後の bit width を指定していなかった。そのため、不当に大きい bit width として処理され、残りの上位 3 bit を埋め尽くし、正しく M3 代表点情報が出力されていなかった。bit width を指定すると改善した。

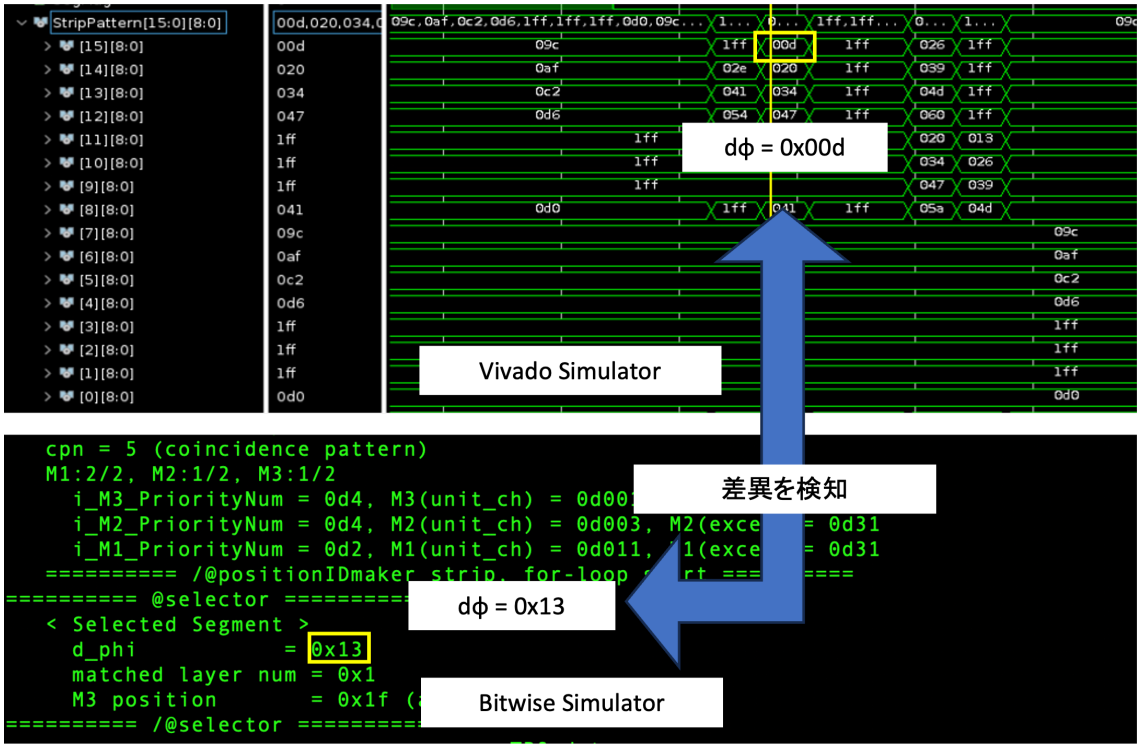

図 5.3: 特定の出力不一致イベントにおける Strip Segment Reconstruction での Bitwise Simulator と Vivado Simulator の出力。

次に、Wire Segment Reconstruction における不具合である。Wire Segment Reconstruction から出力した Segment は Wire Strip Coincidence へ送られる。その接続は for 文で記述されていたが、ループの数が不当な値であり、信号線の数を下回っていた。つまり、部分的に断線していた。適切に接続すると改善した。

さらに、Wire Segment reconstruction において LUT を書き込むパスにも不具合を発見した。LUT 書き込みパスは LUT を subunit ごとに存在する URAM に分配する形で書き込む。Vivado Simulator においては一つの subunit の LUT を書き終わった後にすべての subunit に対しての LUT 書き込みを終了する実装となっていた。LUT のデータ数は subunit 毎に異なるため[*2]、短いデータ数の LUT の書き込みが終了した時点で長いデータ数の LUT の書き込みが途中で終了してしまっていた。LUT 書き込みを行うモジュールをそれぞれの subunit での書き込み終了を検知して終了する仕様に変更したところ改善した。

5.1.3　Wire Strip Coincidence の検証

Wire Strip Coincidence でも複数の不一致が見つかった。途中出力の bit 単位の比較によって Bitwise Simulator と Vivado Simulator の両者において複数箇所での不具合を特定した。ここでは、ファームウェアで見つかった不具合を紹介する。

まず、Block Selector の不具合である。Block Selector はヒット層数が多い順に飛跡を選び、ヒット層数が同じ場合は $(d\theta, d\phi)$ の絶対値が小さいものを選ぶ。$(d\theta, d\phi)$ はそれぞれ最上位 bit が符号を表し、それ以降の下位 bit が絶対値を表す。$(d\theta, d\phi)$ の大小比較を行う条件式が最上位 bit を含んだままとなっていて絶対値の大小評価が適切に

[*2] Wire Segment Reconstuction の LUT では飛跡を再構成できない不当なデータは省略して格納されておらず、省略される不当なデータの数は位置によって異なる。

行えていなかった。絶対値を意味する bit のみを抜き出し大小評価を行うように修正すると改善した。

次に、Duplicate Selector である。Duplicate Selector は 1 つの Region に対して 2 つのチェンバー由来の segment が入力される場合にいずれを選ぶか判定する回路である。ヒット層数が多い順に選び、ヒット層数が同じ場合は $(d\theta, d\phi)$ が小さいものを選ぶが、ここでも条件式に符号 bit が含まれていた。絶対値を意味する bit のみを抜き出し大小評価を行うように修正すると改善した。

また、ファームウェアの不具合以外に Coincidence Window の不具合も発見した。共同研究者によって運用される Coincidence Window が出力するシステムとファームウェアの間で $d\theta$ の符号の定義が反転していた。両者をデザインレポートの定義に揃える形で修正した。

5.1.4　現時点での検証結果

Channel Mapping から Wire Strip Coincidece までの 8603 events の比較によって、不具合箇所を特定し修正を行なった。現時点での結果を表 5.1 に示す。残りのわずかな不一致イベントに対して今後も本検証手法による特定と修正を進めていき、不具合の完全な解消を目指す。

表 5.1: 8603 events 中の不一致イベント数

Wire Segment Reconstruction	Strip Segment Reconstruction	Wire Strip Coincidence
7 events	7 events	65 events

5.2　検証システムによる修正前後の性能評価

この節では検証システムによる修正前後のトリガー性能を見る。ここでは先行研究 [9] と比較するため、SL 実機試験におけるトリガー性能、特に、トリガー効率の運動量依存性について比較をする。まず、Efficiency を次のように定義する。Wire Strip Reconstruction の場合は横方向運動量の閾値 p_{T}threshold を用いて

$$\mathrm{Efficiency} = \frac{p_{\mathrm{T}}\mathrm{threshold} = 20\ \mathrm{GeV}\text{と判定されたイベント数}}{\text{入力したイベント数}} \tag{5.1}$$

と定義する。

使用したシングルミューオン MC データは表 5.2 の通りであり、パイルアップは無く、運動量・位置に対してミューオンイベントが均等に分布している。η や ϕ に対するカット条件は A-side のある 1 つのトリガーセクターを選んだことによる。

表 5.2: トリガー性能評価に使用した MC データの性質

Parameter	カット条件
p_T	$0 < p_T < 50$ GeV （均等に分布）
η	$1.1 < \eta < 2.4$ （均等に分布）
ϕ	$0.03 < \phi < 0.25$ （均等に分布）

プラトー領域をここでは $p_{\mathrm{T}} > 20\mathrm{GeV}$ と定義する。プラトー領域での Efficiency は修正前で **81.9 %** であったが、修正後で **94.6 %** となり、修正前後で Efficiency は **12.7 %** も回復した。検証システムによる修正前後の実機試験出力における Efficiency の p_{T} 分布を図 5.4 に示す。修正前に見えていた Turn On Curve の立ち上がり部分が

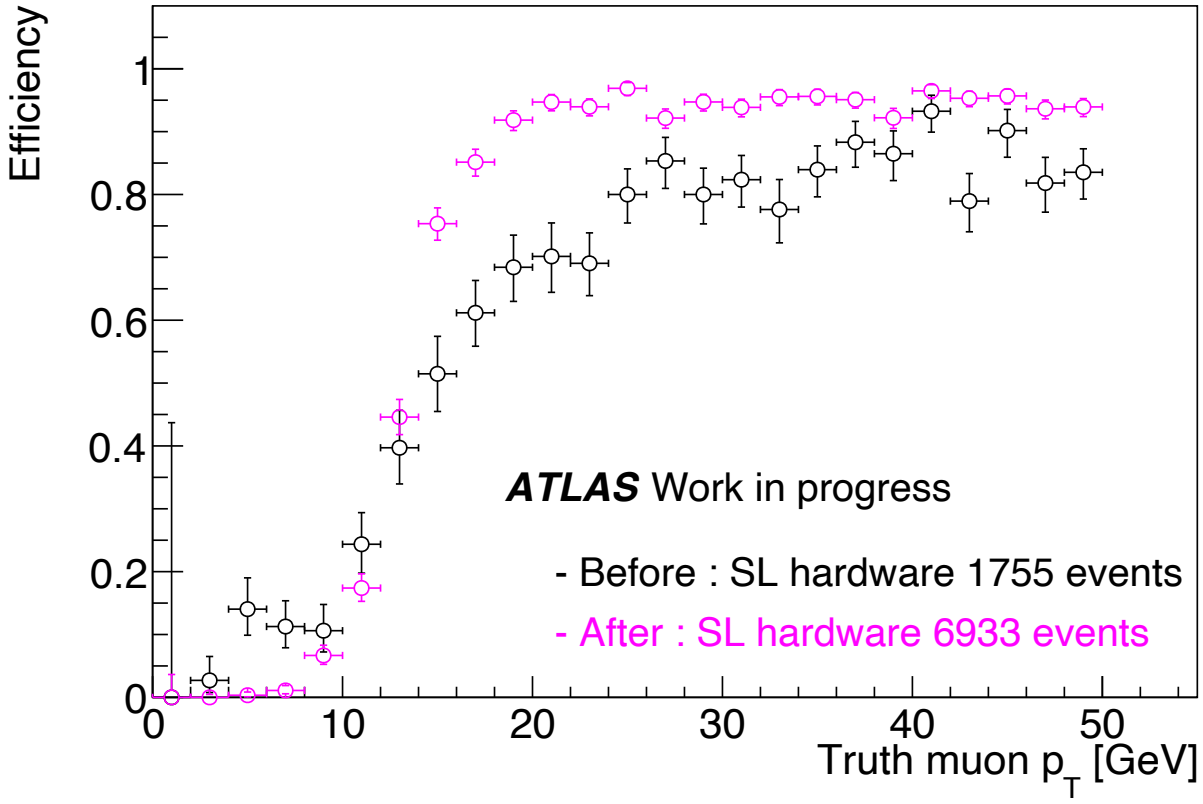

図 5.4: 修正前後の SL 実機試験出力における Efficiency の p_{T} 分布。

鈍っているところが改善し、プラトー領域も全体的に Efficiency が回復しているのがわかる。ただし、修正前に使用したイベントセットはイベント数が少ないが表 5.2 に示した条件を満たすものである。

また、同じく SL 実機試験のプラトー領域における Efficiency の η 分布、ϕ 分布は図 5.5 である。$1.2 < \eta < 1.6$ の領域[*3]で見えていた大きな Inefficiency が改善されていることがわかる。

検証システムによる不具合を修正によって、修正前に見えていた諸々の Inefficiency が解消し、Turn on curve が正常な形になっているのを確認できた。

5.3 理想性能との比較

この節では理想性能と現状のトリガー性能を比較する。図 4.2 に示したように、本研究のセットアップでは Software Simulator と共通の MC データを用いており、トリガー性能に関して対等な比較が可能である。Software Simulator は先行研究（[11, 13, 17, 16] 等）によって開発された、トリガーアルゴリズムを実装した C++ プログラムであり、トリガー論理回路の開発に先行してアルゴリズム自体の性能評価を行うシミュレーターである。つまり、トリガー論理回路の開発においては理想性能としてのリファレンスとなる。

Segment Reconstruction で理想性能と同程度のトリガー効率出ていることを確認する。Software Simulator と SL 実機試験、Bitwise Simulator の Wire Segment Reconstruction と Strip Segment Reconstructions での出力を

[*3] トロイド磁場が強い領域である。

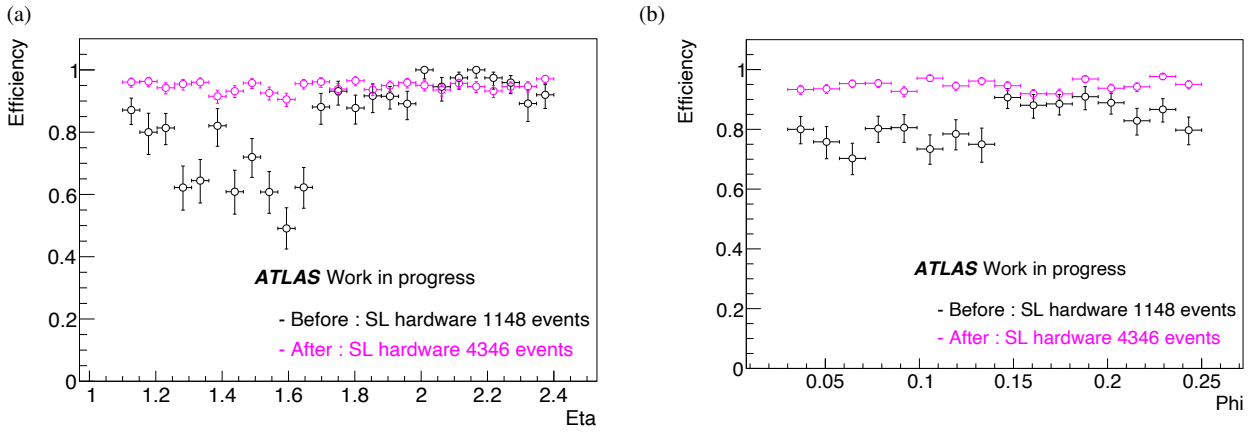

図 5.5: SL 実機試験のプラトー領域における Efficiency。(a)η 分布。(b)ϕ 分布。

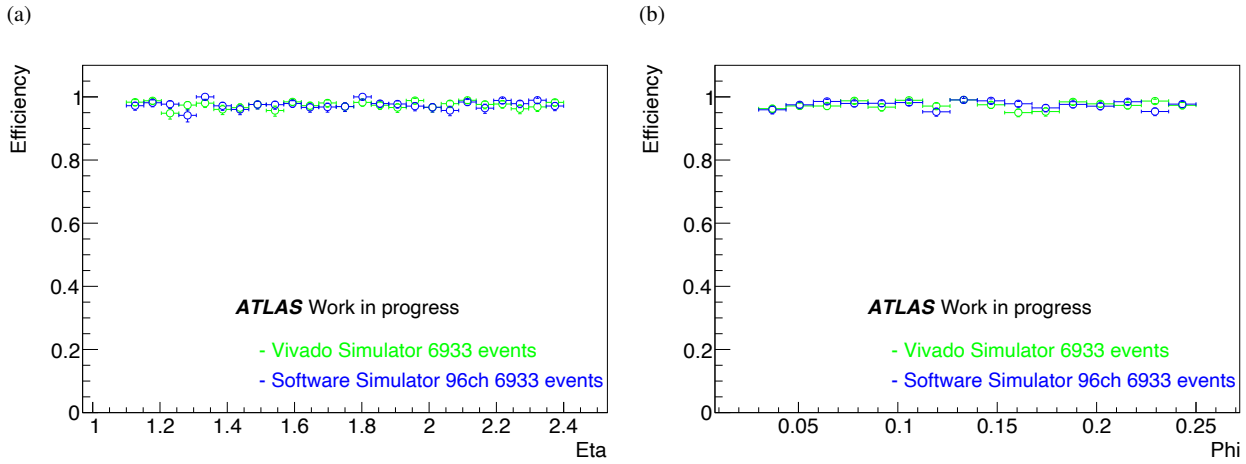

図 5.6: Wire Segment Reconstruction - 修正後の各システムのプラトー領域における Efficiency。(a)η 分布。(b)ϕ 分布。

比較したのがそれぞれ図 5.6 と図 5.7 である。ただし、Segment Reconstruction における Efficiency は

$$\text{Efficiency} = \frac{\text{SegmentReconstruction において直線飛跡を再構成したイベント数}}{\text{入力したイベント数}} \tag{5.2}$$

と定義する。Segment Reconstruction で、Software Simulator と同程度の Efficiency を示していることがわかる。ただし、$1.2 < \eta < 1.35$ で他のシステムと比較して SL 実機試験のみに数 % の Inefficiency がある。これについては 5.4 節で議論する。

次に Wire Strip Coincidence の出力について議論する。Software Simulator と SL 実機試験、Bitwise Simulator の Wire Strip Coincidence での出力を比較したのが図 5.8 と図 5.9 である。現在の SL 実機試験は、Software Simulator の Efficinecy に比肩する Eficciency を有していることがわかる。ただし、図 5.9(a) では、$1.95 < \eta < 2.2$ で他のシステムと比較して SL 実機試験のみに数 % の Inefficiency がある。これについては 5.4 節で議論する。

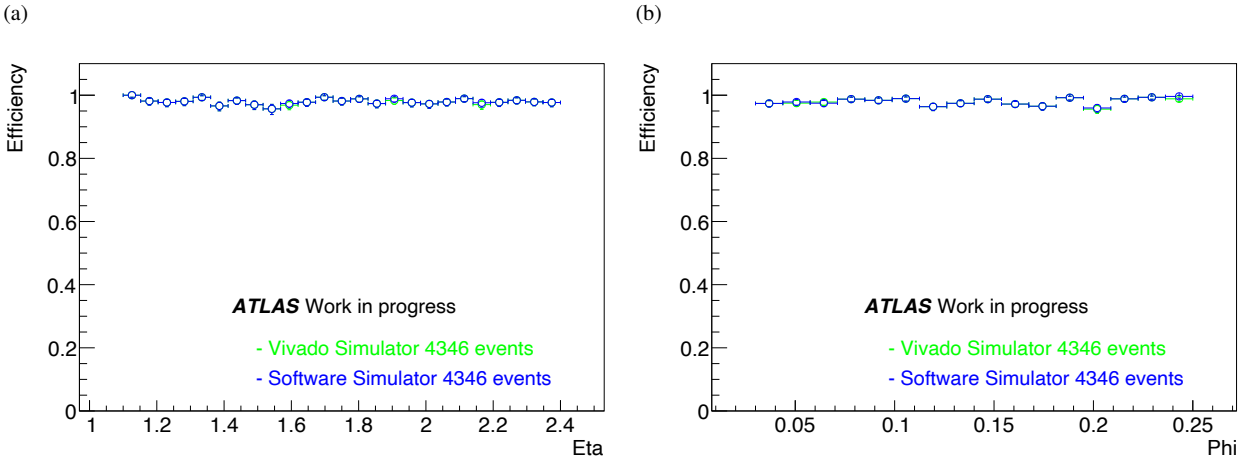

図 5.7: Strip Segment Reconstruction - 修正後の各システムのプラトー領域における Efficiency。(a)η 分布。(b)ϕ 分布。

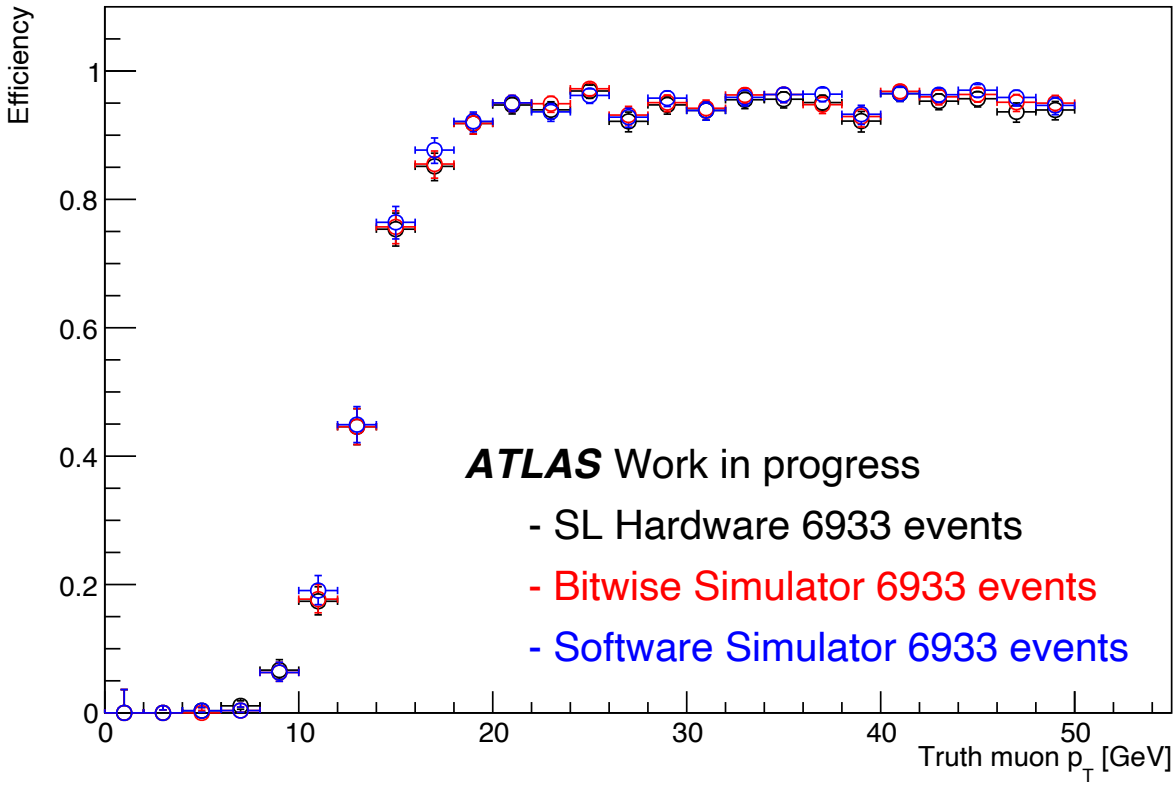

図 5.8: Wire Strip Coincidence - 修正後の各システムにおける Efficiency の p_{T} 分布。

(a) (b)

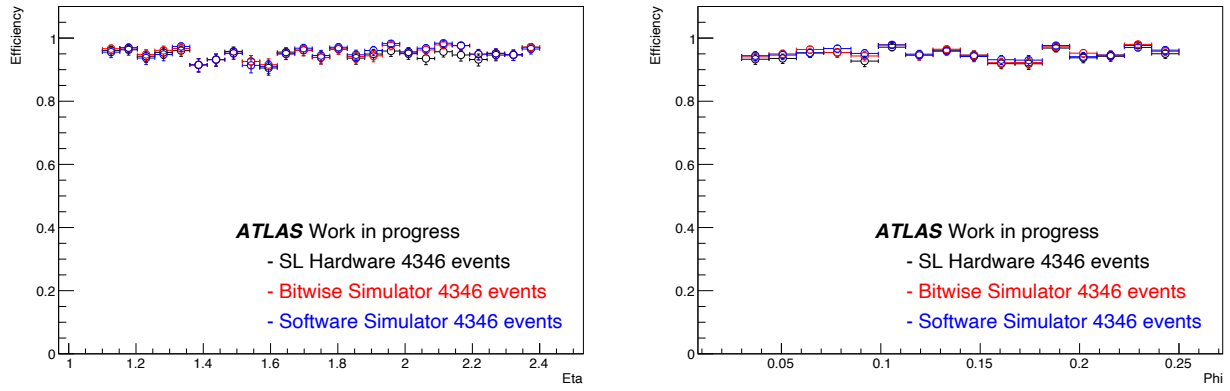

図 5.9: Wire Strip Coincidence - 修正後の各システムにおける Efficiency。(a)η 分布。(b)ϕ 分布。

5.4　SL 実機試験での Inefficiency

SL 実機試験と Vivado Simulator を比較して、その差異について議論する。図 5.12 と図 5.13 に示すように、Segment Reconstruction において出力に差異がある。また、図 5.10 と図 5.11 に示すように、Wire Strip Coincedence において出力に差異がある。この性能評価に用いた全 6933 events 中の 271 events（3.9 %）が Wire Strip Coincidence および Segment Reconstruction のいずれかで一致していない。いずれにおいても SL 実機試験の出力が Vivado Simulator の出力を下回っている。加えて、SL 実機試験では試験アプリケーションの run by run に対して p_{T} threshold の出力が約 0.1 % だけ変化することが確認されている。以上のことから、SL 実機試験において何らかの Inefficinecy が発生していると考えられる。両者はいずれも同一のトリガー論理回路ファームウェアを用いており、その差異は LUT を書き込むパスとリードアウトパスである。

LUT の書き込みパスについて、SL 実機試験は MPSoC から SL FPGA の URAM/BRAM に書き込むのに対して、Vivado Simulator ではプロジェクトに格納された LUT を参照して書き込む。SL 実機試験における LUT 書き込みパスでは MPSoC とのインターフェースとなるモジュールとトリガー論理回路の間でクロックドメインクロッシング（CDC）が存在する。クロックドメイン境界においてはタイミング不一致の際にデータロスを発生させないために適正な仕掛けを実装する必要がある。現時点ではこの実装に関する要求を完全に満たせていないことを認識しており、わずかなデータロスの発生による Efficiency の低下がここで発生していると推察している。現在、CDC の実装にアップデートが施され、その検証を行なっている。さらに、今後 LUT の書き込みエラーを検知するために LUT の読み取り機能を実装する予定である。

また、リードアウトパスについて、SL 実機試験の出力は読み出し回路を介して MPSoC に送られるのに対して、Vivado Simulator ではリードアウトパスを省略してトリガー論理回路の出力をそのままテキストダンプしている。今後はリードアウトパスを Vivado Simulator のテストベンチに含めて検証を進める予定である。

以上のように SL 実機試験特有の問題がいくつか観測されているが、問題解決のための取り組みはいくつか提案・着手しており、戦略的に検証していく。

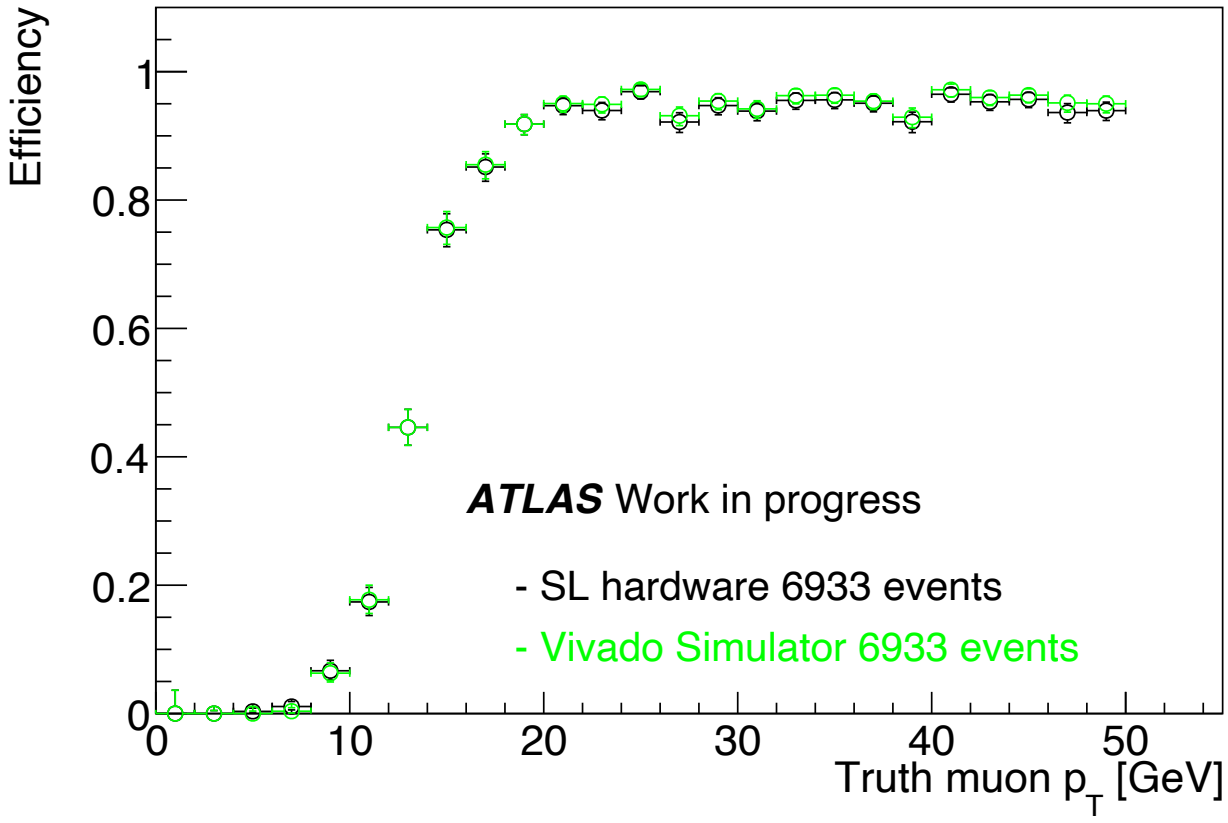

図 5.10: Wire Strip Coincidence - SL 実機試験と Vivado Simulator における Efficiency の p_{T} 分布。

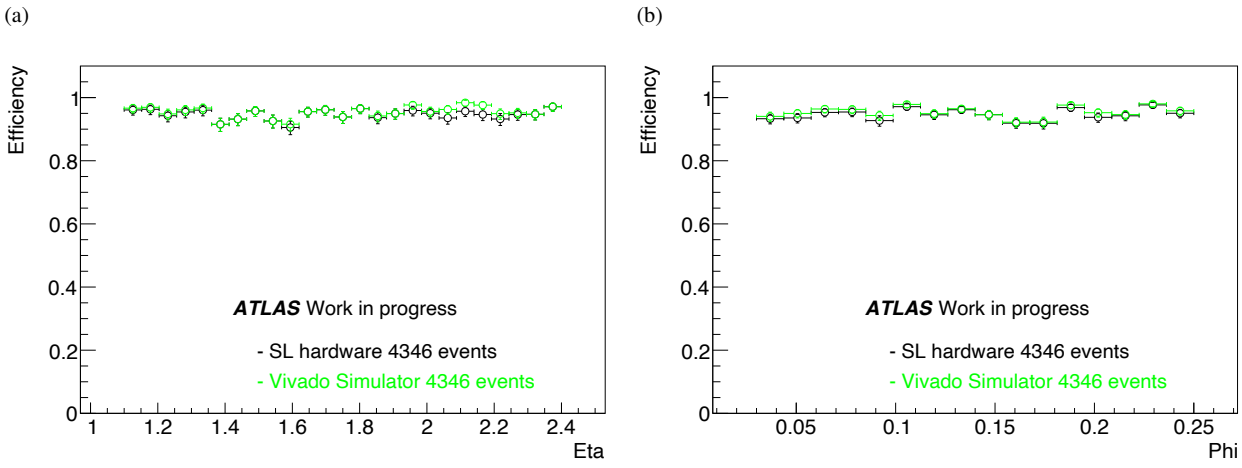

図 5.11: Wire Strip Coincidence - SL 実機試験と Vivado Simulator のプラトー領域における Efficiency。(a)η 分布。(b)ϕ 分布。η 分布を見ると、$1.95 < \eta < 2.25$ の領域、つまりフォワード領域で顕著に Inefficiency が生まれているのがわかる。

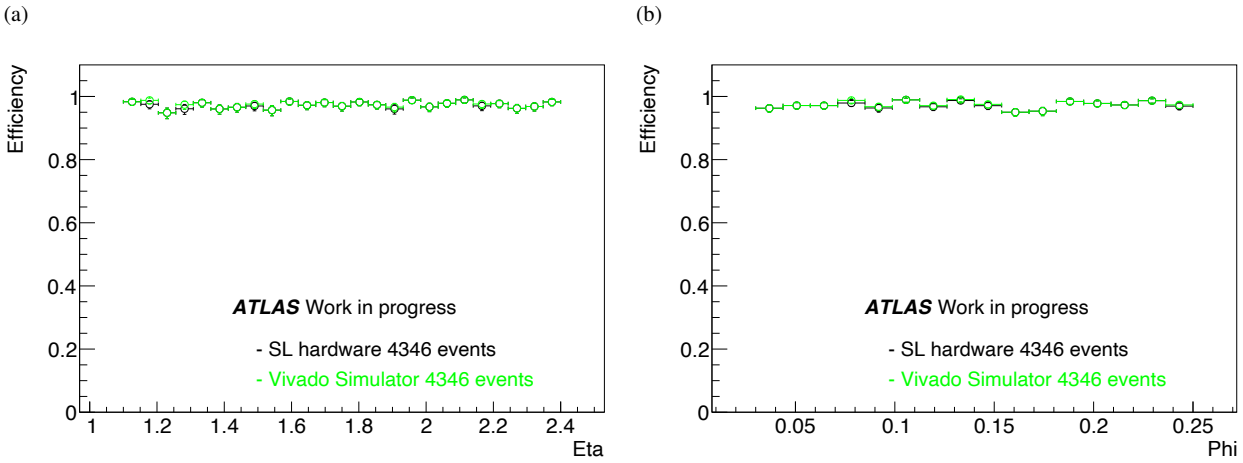

図 5.12: Wire Segment Reconstruction - SL 実機試験と Vivado Simulator のプラトー領域における Efficiency。(a)η 分布。(b)ϕ 分布。

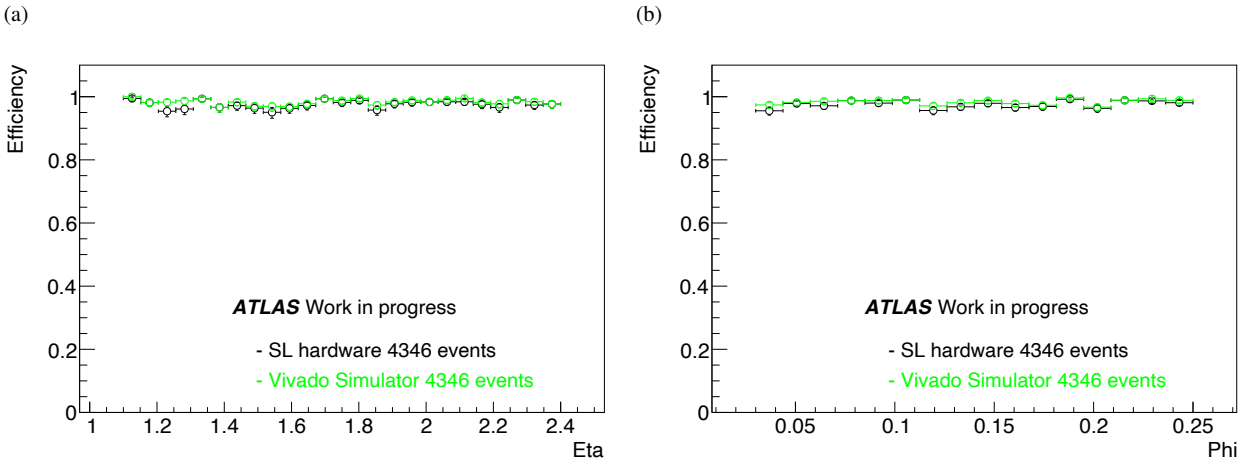

図 5.13: Strip Segment Reconstruction - SL 実機試験と Vivado Simulator のプラトー領域における Efficiency。(a)η 分布。(b)ϕ 分布。η 分布を見ると、$1.2 < \eta < 1.35$ での領域で大きく SL 実機試験の Efficiency が落ちている。。

第 6 章

今後の開発の展望

この章では、今後の開発の展望について述べる。現状のトリガー論理回路の開発・検証状況は図 6.1 で示すように、本研究によって Channel Mapping から Wire Strip Coincidence までの回路はよく検証できた。次のステップとして、Inner Coincidence と Track Selector を含んだ統合トリガー回路での検証を行う。Inner Coincidence から Track Selector までの回路は部分的に仮実装の箇所が残っていたり、追加の機能を実装しているところであり、実装でき次第検証を行うため検証システムを拡張する。また、Trigger Readout コンテンツはデバッグ用の仮実装となっているので、デバッグが終盤に差し掛かった現状では本実装に着手できる段階にある。以下ではそれぞれに開発項目について詳しく述べる。

6.1 Inner Coincidence と Track Selector を含んだ統合トリガー回路の検証

本研究で構築した検証システムは Wire Strip Coincidence までしか対応していない。検証システムを構成する各要素について拡張すべき項目をまとめる。

6.1.1 テストパターン生成システム

現状の実装では、MC データに TGC BW のヒットデータしか格納されていないため、TGC BW のテストパターンしか生成できない。今後 Inner Coincidence に入力するためのテストパターンを生成するために、NSW、RPC BIS7/8、Tile カロリメータ、TGC EI のヒットデータが格納された MC データを用意することとそれに対応したテストパターン生成機能を追加する。

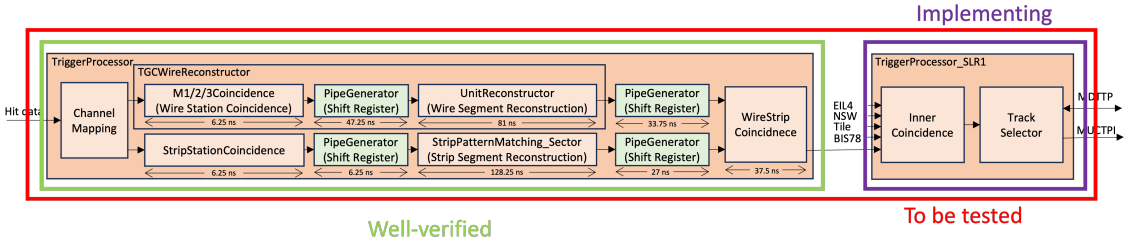

図 6.1: 現状の開発・検証状況。

6.1.2　Bitwise Simulator

現状の実装では、Wire Strip Coincidence までのトリガー回路しか実装されていない。トリガー論理回路ファームウェアの開発に合わせて、Inner Coincidence と Track Selector を追加で実装する必要がある。

6.1.3　Xilinx Vivado Simulator

現状の実装では、Inner Coincidence へのテストパターン入力パスとその読み出しは実装していないため、新たに実装する必要がある。ただし、図 4.4 に示したように、現在のテストベンチでは任意の信号線を用意にプローブできる実装になっているので、簡単に拡張することができる。

6.1.4　SL 実機試験

現状の実装では、Inner Coincidence への Test Pulse Generator は実装されておらず、制御するアプリケーションも未実装である。Test Pulse Generator の SLR1 に複製・応用して、その制御アプリケーションを開発する。

6.2　Trigger Readout の実装

現状、Trigger Readout はデバッグ用の仮実装となっており、トリガー中間出力を全 bit 読み出す実装になっている。リソースとバンドウィズスの観点からトリガー中間出力を全て読み出すのは不可能であり、適切なトリガー中間情報のみを読み出す本実装に移行する必要がある。

6.2.1　現状の Trigger Readout

SL 実機試験および Xilinx Vivado Simulator はこの Trigger Readout にあった実装になっている。トリガー中間出力は 51 種類のデータフレーバーに分けられて、それぞれ全ビット読み出される。全てのデータフレーバーを同時に読み出すことはできないため、実機を用いる際には必要なデータフレーバーの読み出し回路のみをアクティベートする。それぞれのデータフレーバーのコンテンツは付録 C にまとまっている。

6.2.2　本実装の Trigegr Readout の Data format の策定

本番運用での Trigger Readout は FELIX 経由で読み出され、その後ストレージされる。オペレーションにおけるモニターやデバッグ、パフォーマンススタディに役立てる目的のものである。FELIX との接続は 9.6 Gbps の 1 link でありバンドウィズスや読み出しモジュールのリソース削減の観点から、必要最低限なデータ量に抑えたい。以上を踏まえて、本実装の Trigger Readout の Data format を策定した。フォーマットの外枠は図 6.2 で表される。読み出しコンテンツは 29 種類のデータタイプに分けられ、データタイプごとに candidate という概念を定義する。例えば、Wire Strip Coincidence であれば、飛跡候補を candidate に割り当てる。1 つの candidate に対して、最大 4 words を使ってトリガー中間情報を表現する。具体的なデータタイプとそのフォーマットの定義は [18] および付録 D を参照されたい。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Start of Packet	Start [31:24]								D5.6								D5.6								K28.1							
Header 0	Format version																															
Header 1	Trigger Type																Rsvd						Fiber ID		Rsvd			Sector Logic ID				
Header 2	Run Type								Run Number																							
Header 3	L0ID [31:0]																															
Header 4	Rsvd																BCID								Rsvd			L0ID [37:32]				
Header 5 (meta switch)	Enable Data Type bitmap [31:0]																															
Header 6 (meta switch)	Enable Data Type bitmap [63:32]																															
Trigger Data Header(meta data)	Data type (Data flavor)						Words per cand		BC readout enable bitmap (NN,N,C,P)				Number of candidates (NN)				Number of candidates (N)						Number of candidates (C)					Number of candidates (P)				
Data	Data																															
Trailer 0	Data loss flag bitmap [31:0]																															
Trailer 1	Data loss flag bitmap [63:32]																															
Trailer 2	Rsvd																								Number of PP ASIC hit words (max 992)							
Trailer 3	Error bitmap																															
End of packet	BUSY status								CRC20																K28.6							

図 6.2: Trigger Readout の Data format（[18] および付録 D にまとまっている。）。

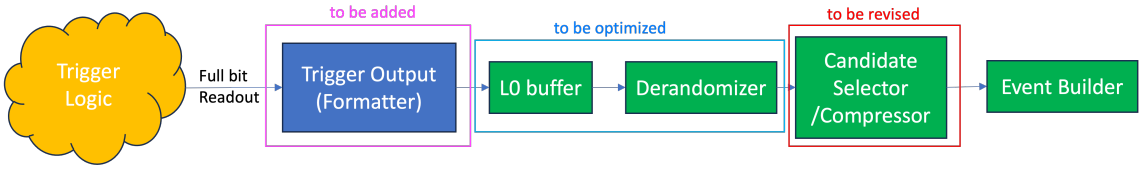

図 6.3: Trigger Readout の実装案。

6.2.3 実装の見通し

策定した Trigger Readout Data format は図 6.3 のように実装する計画である。L0 buffer 以降の読み出しモジュールは既存のものであり、その上流に策定したフォーマット通りにトリガー出力を成形する Formatter を実装する。全 bit 読み出しの実装から formatter による bit 幅の削減が見込まれるのでそれに応じて L0 Buffer および Deradomizer（処理待ち用の Buffer）の幅を最適化する。Candidate Selector/Compressor では圧縮処理が行われるが、これも Formatter による成形後の出力に適した実装に変更する。以上のような計画で Trigger Readout を実装する。

第 7 章

結論

瞬間最高ルミノシティが現行の約 3 倍となる高輝度 LHC が 2029 年から始まる。増大する衝突レートに対してアクセプタンスを落とさないようにするため、ATLAS 実験の TDAQ システムはアップグレードされる。初段トリガーレートは 1 MHz、初段トリガーレイテンシーは 10 μs に増強される。それに伴って、エンドキャップ部ミューオントリガーを担う TGC 検出器のエレクトロニクスも刷新される。

高輝度 LHC-ATLAS における TGC エレクトロニクスでは、広帯域通信技術を用いてヒットデータ全てを大規模 FPGA に集約する。そこでは、大規模かつ柔軟なトリガー演算、10 μs のヒット信号のバッファリング、ヒットデータや中間出力の読み出しを行う。このトリガー演算は TGC 検出器の 7 層すべてのヒットを利用して飛跡を再構成し、横方向の運動量を概算するものである。開発したトリガー論理回路は大規模かつ複雑であるので、詳細な検証が不可欠である。

本研究では、トリガー論理回路の検証システムを構築した。このシステムは大規模なシミュレーションデータを用い、新たに実装した Xilinx の Vivado Simulator と先行研究による Bitwise Simulator を組み合わせることで、トリガー論理回路の出力をビットレベルで比較し、不具合を特定するというものである。不具合が見つかった場合には、トリガーロジックの上流から順に中間出力を詳細に調査し、その原因を特定した。その結果、差異が生じるイベントを全イベント約 8600 イベント中、約 80 イベント以下に抑えることができた。この検証システムは、SL 実機試験特有の問題の切り分けにも役立ち、今後のトリガー回路開発における基盤を構築したと言える。

また、修正後のトリガー論理回路の性能評価を行い、トリガー効率の大幅な向上を確認した。プラトー領域での効率は修正前の 81.9 % から 94.6 % に改善され、理想的なソフトウェアシミュレーションの効率に匹敵する値を達成した。

さらに、トリガー読み出しフォーマットの策定を行い、必要最低限のデータを選択的に読み出す効率的な設計を考案した。今後このトリガー読み出し回路の実装を進める。

本研究により、TGC 検出器のトリガー論理回路は設計通りの性能を示し、プラトー領域でのトリガー効率 95 % という重要な成果を達成した。また、Wire Strip Coincidence までを網羅した検証システムは、Inner Coincidence や Track Selector を含む統合トリガー回路への拡張を視野に入れており、今後の開発においても活用されることが期待される。

謝辞

本研究を進めるにあたり、様々な方々にお世話になりました。

指導教員である石野雅也教授にはお忙しい中で、精力的なご指導をいただき、大変お世話になりました。本論文やその他のプレゼンテーション資料の作成にあたり、コメントと修正を何往復も行って、私の拙い研究発表能力に対して根気強く指導していただきました。また、研究を進めるにおいても、重要な分岐点において何度もクリティカルなご指摘をいただき、正しいアプローチで研究を進めることができました。研究に対する姿勢というマインド面でも多くのことを学ばせてもらいました。奥村恭幸准教授にも大変お世話になりました。研究を進めるにあたってテクニカルな知識に関するご助言を多くいただきました。また、研究進捗に関しても気にかけていただき、私が何らかの障壁に詰まっているときには抜本的な方向転換や効率的アプローチに関するアイデアを提言していただき、非常に助かりました。お二方には大変お世話になりました。博士課程進学後もよろしくお願いします。

またトリガー論理回路の開発における共同研究者として、山下恵理香氏、成川佳史氏、中川徹郎氏、Wanotayaroj Chaowaroj、須江祐貴氏、河本地弘氏、三野裕哉氏、水落永遠氏にもお世話になりました。山下さんには Bitwise Simulator およびテストパターン生成システムの開発者として、それらのシステムの使い方などを手取り足取り教えていただきました。また、CERN にいながらも ZOOM でトリガーアルゴリズムの詳細部分に関することに関しても詳しく教えていただきありがとうございました。成川さんには SL 実機試験の開発者としてその使い方や SL ファームウェアの挙動について詳しく教えていただきました。また、高い視座から SL ファームウェア研究開発におけるビッグピクチャーのアドバイスもいただき、大変参考になりました。京都大の中川には Software Simulator におけるトリガーアルゴリズムの挙動および設計思想について様々なことを教えてもらいました。昼夜問わず飛んでくる私の質問や要望に根気強く答えていただいて、大変感謝しております。I would like to express my gratitude to Wanotayaroj Chaowaroj. He provided me with a lot of technical advice on HDL and Vivado when developing the test bench of Vivado Simulator. His idea for implementing the test pattern input is truly smart. 須江祐貴氏にはトリガーリードアウトのデータフォーマットの策定に関して相談させてもらい、MUCTPI や NSWTP とのインターフェースに関して情報提供していただきました。河本地弘氏には研究発表の場でトリガーアルゴリズムとトリガー論理回路に関しての精密で的確な議論を展開していただき、そのおかげで研究を進める上で正しい理解に至ることができました。水落永遠氏には本研究で構築した検証システム使ってファームウェアおよび Bitwise Simulator のロジックの一部の検証と修正をしてもらいました。今後、不具合の完全撲滅とトリガー読み出しの実装を一緒に頑張りましょう。

他の ATLAS 日本ミューオントリガーグループの方々にお世話になりました。開発ミーティングや研究発表の場においては、前田順平准教授、堀井泰之准教授、齋藤智之助教には TGC エレクトロニクスだけでなく、ATLAS の TDAQ システム全体を俯瞰した上での多角的な指摘や議論をしていただき、大変参考になりました。近藤翔太氏、長坂錬氏、田中碧人氏には、PS Board や TAM、JATHub、その他エレキやエンジニアリングに関する多大なアドバイスをいただきました。

東京大学理学系研究科修士課程学生支援制度、理学系修士卓越 RA のおかげで経済的余裕ができ研究に専念できました。

家族には関西から遠方ながらも東京での研究生活に対して様々な支援をしていいただきました。
皆様のご協力があってこそ本研究を遂行できました。ありがとうございました。

付録

A　テストパターンのフォーマットの詳細

ここでは coe ファイル形式のテストパターンのフォーマットについて説明する。具体例として、pattern_link0.coe について説明する。以下がデータベースより出力した coe ファイル番号と PS Board のリンクの対応関係である。

coe_number	RX_FPGA	RX_Link
NULL	NULL	NULL
0	11	1
NULL	NULL	NULL
1	4	1
2	10	2
NULL	NULL	NULL
3	2	1
4	3	2
5	2	2
6	1	1
7	3	1
8	1	2
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
24	10	1
25	9	2
26	9	1

27	8	1
28	8	2
29	7	1
30	7	2
31	6	1
32	6	2
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
9	13	2
10	15	1
11	13	1
12	15	2
13	14	1
14	14	2
15	16	2
16	16	1
17	17	1
18	17	2
19	18	1
20	18	2
21	12	1
22	20	2
NULL	NULL	NULL
23	12	2
33	20	1
34	21	1
35	21	2
NULL	NULL	NULL

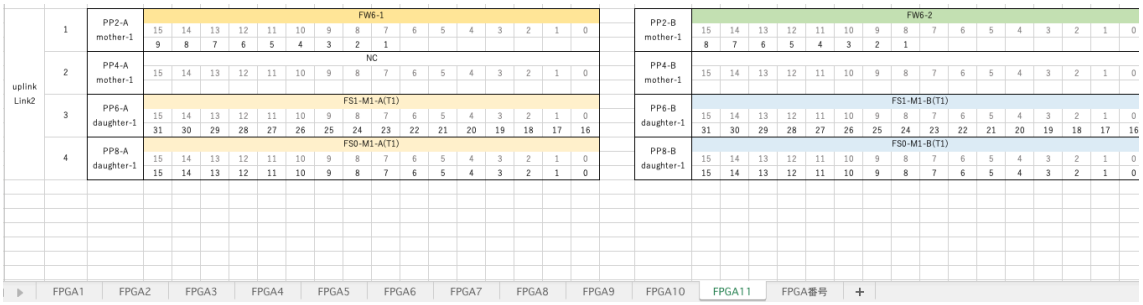

図 A.1: FPGA Table の具体例。

```
|          40 |       22 |        2 |
|          41 |       22 |        1 |
|          42 |       23 |        2 |
|          43 |       23 |        1 |
|          44 |       24 |        2 |
|          45 |       24 |        1 |
|          46 |       25 |        1 |
|          47 |       25 |        2 |
|          48 |       26 |        1 |
|          49 |       26 |        2 |
|          50 |       27 |        1 |
|          51 |       27 |        2 |
|          52 |        5 |        1 |
|          53 |        4 |        2 |
|          54 |       11 |        2 |
|          55 |        5 |        2 |
|          56 |       19 |        2 |
|          57 |       19 |        1 |
|          58 |       29 |        1 |
|          59 |       28 |        1 |
|          60 |       28 |        2 |
|        NULL |     NULL |     NULL |
|          61 |       29 |        2 |
|        NULL |     NULL |     NULL |
|        NULL |     NULL |     NULL |
|        NULL |     NULL |     NULL |
|        NULL |     NULL |     NULL |
|        NULL |     NULL |     NULL |
+-------------+----------+----------+
```

対応関係によると、pattern_link0 に対応するのは RX FPGA-11 の RX LINK-1 である。

次に、[14] にある、FPGA table（excel ファイル）を参照する。FPGA11 のシートを選び、uplink Link2 を選ぶと

図 A.1 のようになる。このテーブルは 32 bit の行で構成されており上から順に LSB であるので、pattern_link0.coe の 1 行 128 bit に対応するのは{FS0-M1-A,FS0-M1-B,FS1-M1-A,FS1-M1-B,NC,NC,FW6-1,FW6-2}である。他のテストパターンの coe ファイルも同様に対応関係を調べることができる。

B Bitwise Simulator のデバッグ表示

この節では Bitwise Simulator でのデバッグ表示の意味を解説する。Channel Mapping と station coincidece においてはロジックが単純で直感的に理解できるようになっているためスキップし、Segment Reconstruction と Wire Strip Coincidence に限る。

B.1 Wire Segment Reconstruction

図 B.2 はあるイベントのデバッグ表示であり、このイベントについて具体的に説明する。注目する segment は (unit,subunit)=(15,2) である。よって [15] における Unit-15 SubUnit-2 を参照すると、RAM ID は 0x0000013E である。

一方でアドレス生成について、unit 内での代表点番号（mhitINFO）で書いて、(M1,M2,M3)=(0x2f,0x10,0x0)（2 進数で (0b1011_11,0b10000,0b00)）となる代表点に注目する。M1 ではそれぞれ下位 2 bit を落としてアドレス生成することに注意すると、アドレス={M1 代表点番号（mhitINFO）[5:2], M2 代表点番号（mhitINFO）[4:0], M3 代表点番号（mhitINFO）[1:0] }=0b1011_10000_00=0x0x5c0 となる。

以上の情報を踏まえて、LUT を参照する。Wire Segment Reconstruction の LUT を参照し、RAM ID が 0x13E でかつ、アドレスが 0x5c0 の箇所を探すと、図 B.3 のように、格納されたデータは 0x0000288c4a22528844 とわかる。潰した M1 の足は 3 であるので、該当する segment 情報は下位から 1 個目に位置する（3.3.1 節を参照）。ゆえに該当する segment は格納されたデータの [18:0] つまり 0b00000_1101=0x28844=0b1_0_1000_10000100_0100 となる。wire segment reconstruction における格納されたデータの内訳は{フラッグ 1 bit, 0, M3 代表点番号 4 bit, $d\theta$ 9 bit, 横方向運動量 4 bit}であるので、$d\theta$ は 0b10000100=84（最上位が符号 bit であるので、符号を明示的に書く場合は $d\theta = +4$）となる。

B.2 Strip Segment Reconstruction

図 B.5 はあるイベントのデバッグ表示であり、このイベントについて具体的に説明する。注目する segment は (chamber,unit,subunit)=(3,2,0) である。よって [15] における Chamber-3 Unit-2 port-A を参照すると、RAM ID は 0x0101C である。

一方で、このイベントで発火チャンネルおよび発火代表点は図 B.4 であり、代表点は M1 が 2 通り、M2 が 2 通り、M3 が 1 通りあるので、合計で 2×2=4 通りの代表点の組み合わせが考えられる。そのうち、代表点番号が (M1,M2,M3)=(224,223,223) となる組に着目する。この組の代表点番号を unit 内での代表点番号（unit_ch）に書き直すと (M1,M2,M3)=(3,2+4,2+12)=(0b011_10,0b011_0,0b011)（unit_ch）となる。この時、M2 と M1 でそれぞれ +4 と +12 しているのは、unit 内に含まれる代表点番号数が多いからである（3.3.2 節を参照）。M1 と M2 ではそれぞれ下位 2 bit、1 bit を落としてアドレス生成することに注意すると、アドレス={subunit 1 bit, M1 代表点番号（unit_ch）[4:2], M2 代表点番号（unit_ch）[3:1], M3 代表点番号（unit_ch）[2:0] }=0b0_011_011_011=0x0xdb となる。

以上の情報を踏まえて、LUT を参照する。Strip Segment Reconstruction の LUT を参照し、RAM ID が 0x0101C でかつ、アドレスが 0xdb の箇所を探すと、図 B.6 のように、格納されたデータは 0x0004c4c39828344034 とわかる。潰した M1、M2 の足は (M1,M2)=(2,1) であるので、該当する segment 情報は下位から 3 個目に位置する

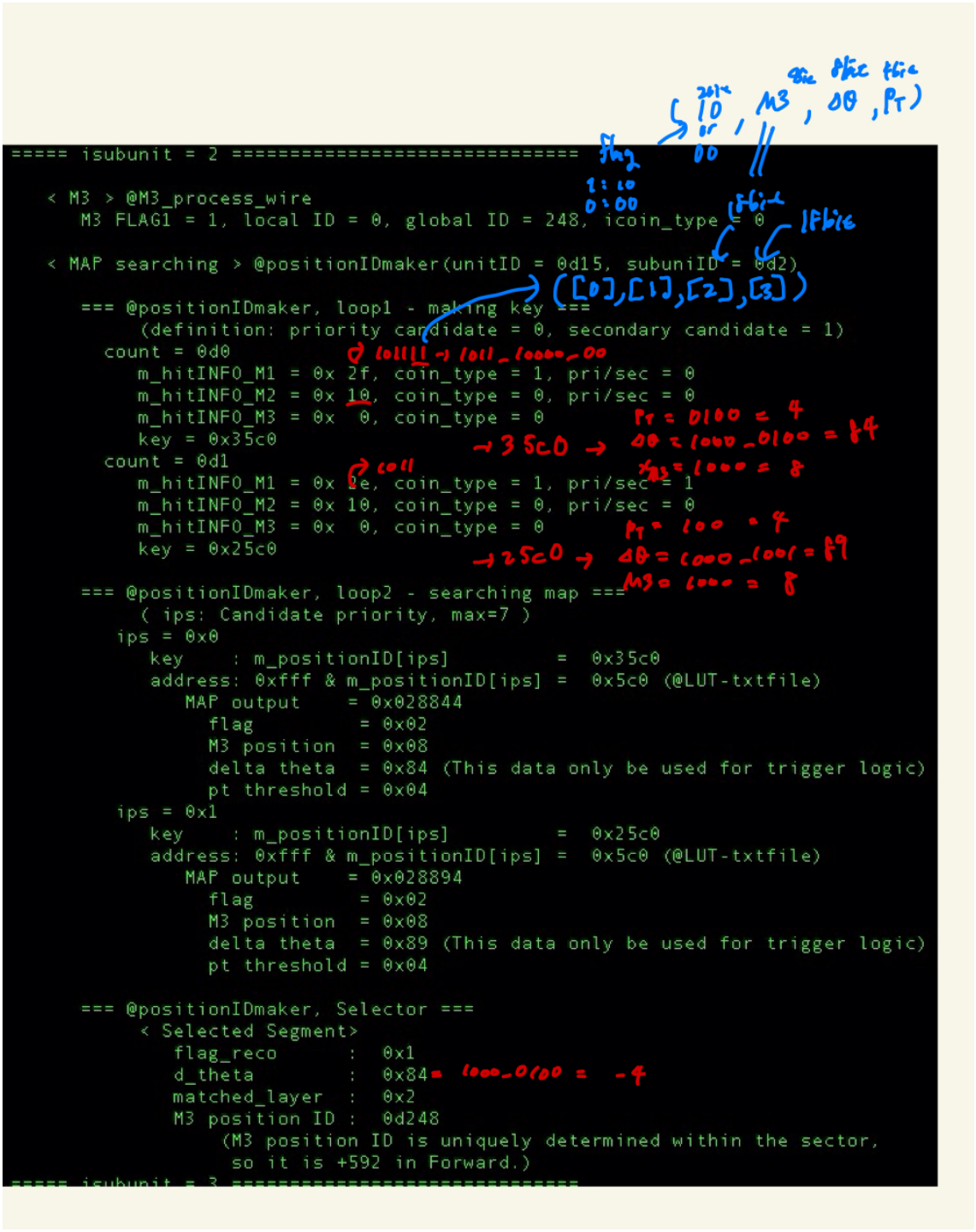

図 B.2: Bitwise Simulator のデバッグ表示:Wire Segment Reconstruction。$d\theta$ の符号は画像中では-で書かれているが、＋が正しい。

```
9371    0 13E 5b8 a23500000000000000
9372    0 13E 5bc a239288a4a21d28834
9373    0 13E 5bd a649298d4000000000
9374    0 13E 5c0 0000288c4a22528844
9375    0 13E 5c1 a649298e4a62d29874
9376    0 13E 5c2 aa552a904aa312a884
9377    0 13E 5c5 000000000000029874
```

図 B.3: Bitwise Simulator のデバッグ表示:LUT の一部（wire segment reconstruction の例）。

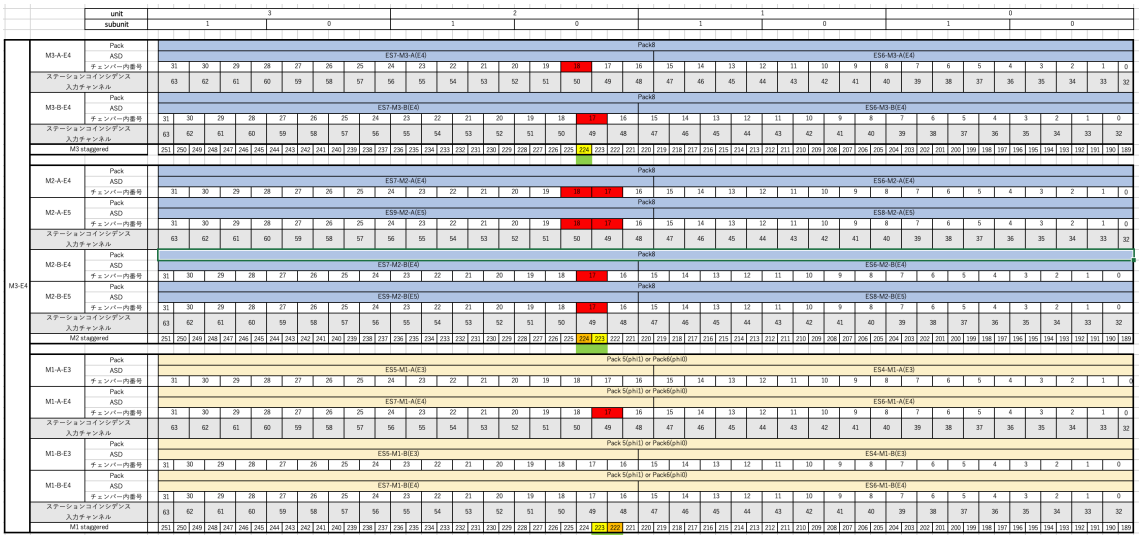

図 B.4: Bitwise Simulator のデバッグ表示:Strip Segment Reconstruction のヒットテーブル

(3.3.2 節を参照)。ゆえに該当する ϕ は格納されたデータの [9+6 × 9:0+6 × 9] つまり ϕ=0b00000_1101=0xd となる。

B.3 Wire Strip Coincedence

図 B.7 はあるイベントのデバッグ表示であり、このイベントについて具体的に説明する。wire の (unit,subunit)=(0,1) であり、連番での subunit 番号は $4 \times 0 + 1 = 1$ である。よって [15] における Eta001 を参照すると、RAM ID は 0x10001 である。

一方で、$d\theta$=0x82=0b10000010、$d\phi$=0x0=0b000000000、strip の unit=0b00 であるので、アドレス={strip unit[1:0], 変換後の $d\theta$[6:0], 変換後の $d\phi$[3:0]}=0b00_100001_0000=0x410 となる。

以上の情報を踏まえて、Wire Strip Coincidece の LUT を参照し、RAM ID が 0x10001 でかつ、アドレスが 0x410 の箇所を探すと、図 B.8 のように、p_{T} threshold は 4 とわかる。

C 現状のデータフレーバーのリスト

この節では現状（仮実装）のデータフレーバーを表 C.1 と表 C.2 にまとめる。Strip Segment Reconstruction での unit/subunit/m3ch は現状は反転処理を補正した後の値である。今後反転位置を Wire Strip Coincidence まで後段にする変更が予定されている。また、データフレーバーの 50 番以降は利用していないので省略した。

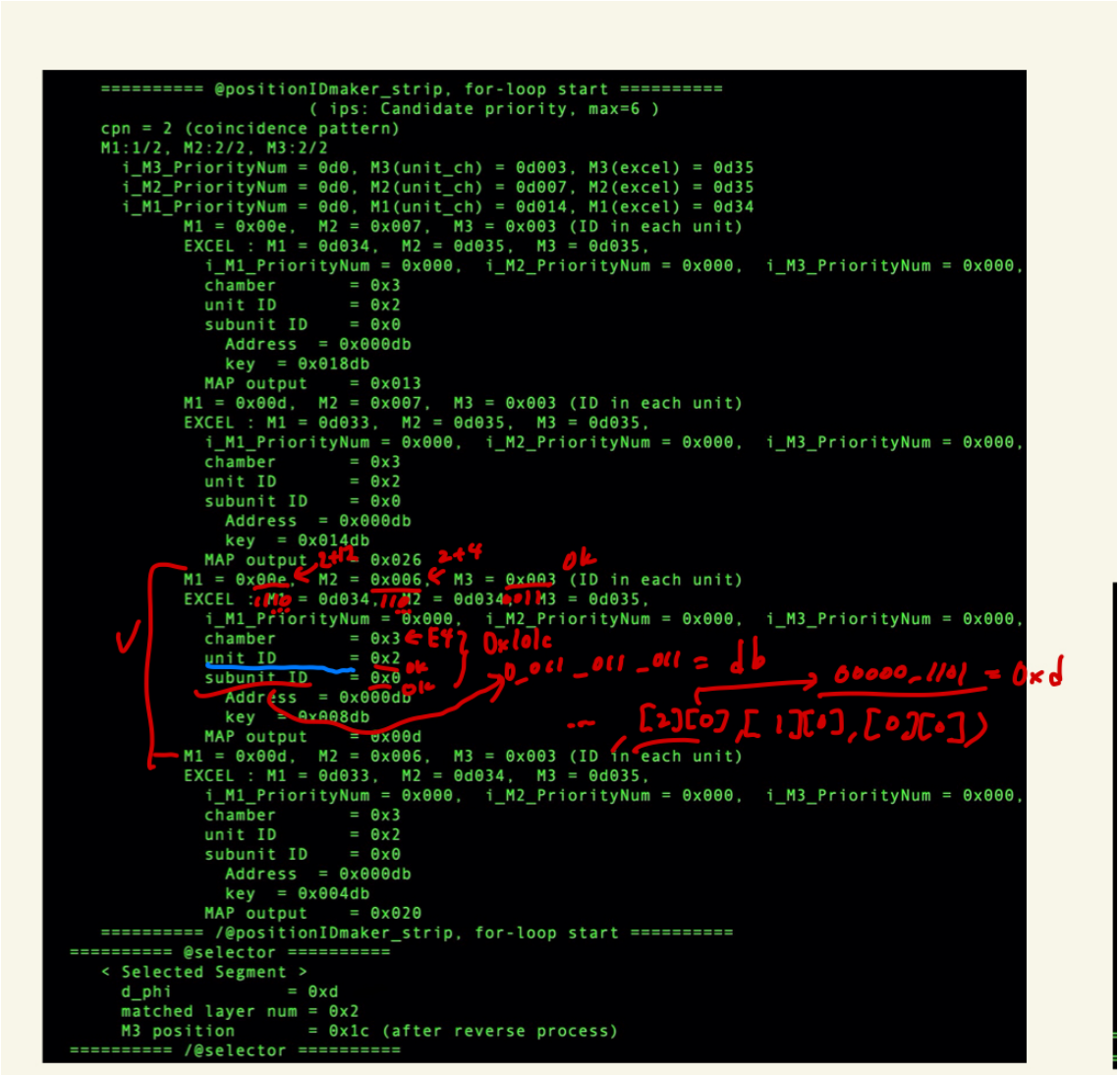

図 B.5: Bitwise Simulator のデバッグ表示:Strip Segment Reconstruction

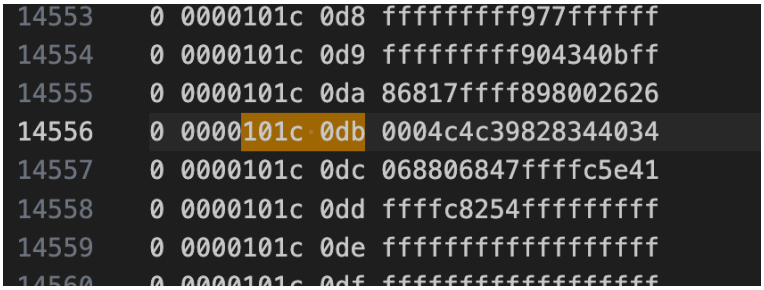

図 B.6: Bitwise Simulator のデバッグ表示:LUT の一部（strip segment reconstruction の例）。

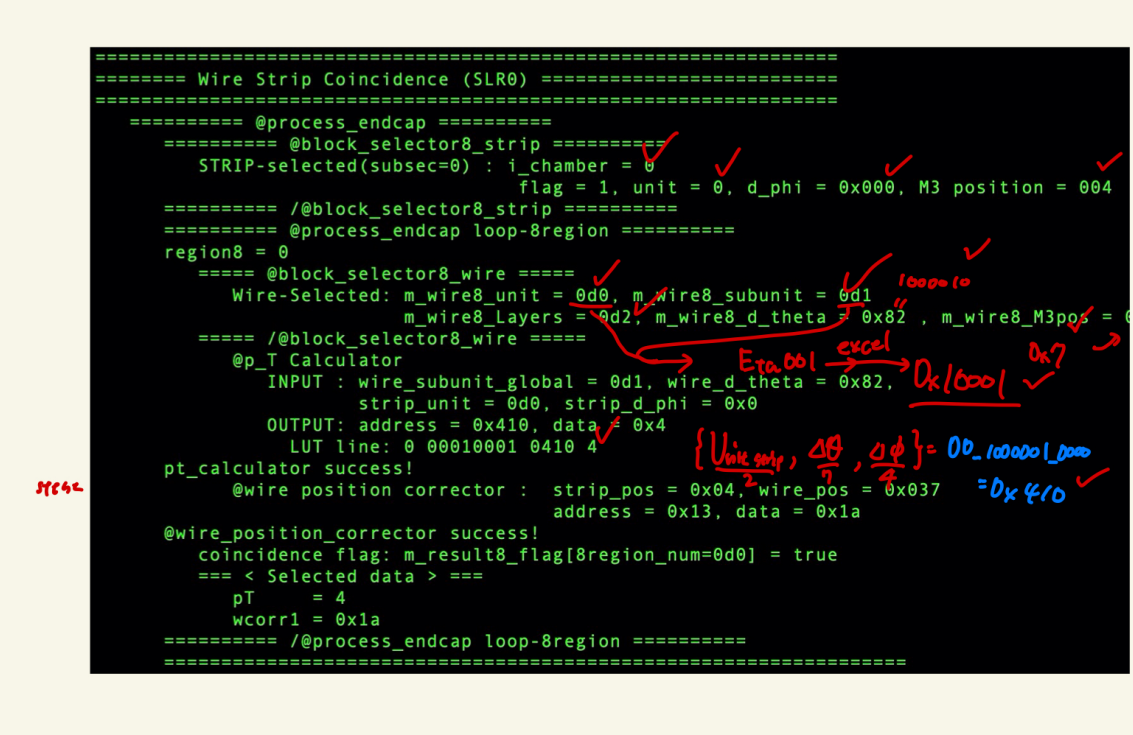

図 B.7: Bitwise Simulator のデバッグ表示:Wire Strip Coincidece

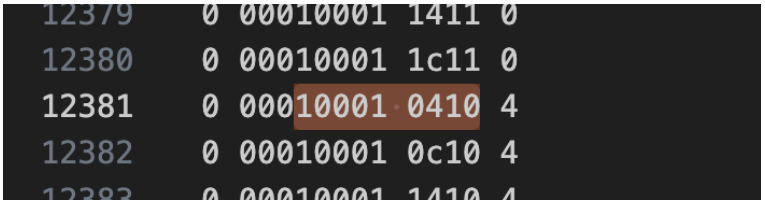

図 B.8: Bitwise Simulator のデバッグ表示:LUT の一部（例）。

D トリガーリードアウトフォーマットの策定

この節では全てのデータタイプ（データフレーバー）に対するフォーマットについて詳細に説明する。フォーマット自体は [18] にまとまっている。以下で各データタイプごとに具体的に説明するが、そのうち他の検出器から送られてくる等して、本研究による考案以前にすでに策定が済んでいるものに関してはスキップする。

D.1 Channel Mapping

図 D.9 のように、Channel Mapping は単純なヒットチャンネルの情報であるので、candidate という概念を無視して 1 word に対して 3 つまたは 5 つのヒットデータを格納するフォーマットにした。FW のストリップは最大チャンネル数が小さく、表現に必要な bit 幅が小さく済んだので、特別に最適化している。

表 C.1: 現状のデータフレーバーの一覧（1）

番号	モジュール	コンテンツ
0-3	ϕ0 Channel Mapping	{20'b0, wire L1[195:0], wire L2[195:0],wire L3[195:0], wire L4[301:0], wire L5[301:0], wire L6[289:0], wire L7[289:0], strip M1 A[159:0], ... , strip M3 B[159:0]}
4-7	ϕ1 Channel Mapping	{20'b0, wire L1[195:0], wire L2[195:0],wire L3[195:0], wire L4[301:0], wire L5[301:0], wire L6[289:0], wire L7[289:0], strip M1 A[159:0], ... , strip M3 B[159:0]}
8-9	FW Channel Mapping	{24'b0, wire L1[104:0], wire L2[103:0],wire L3[104:0], wire L4[124:0], wire L5[124:0], wire L6[121:0], wire L7[121:0], strip M1 A[31:0], ... , strip M3 B[31:0]}
10	ϕ0 Wire Station Coincidence	M1 代表点番号に対応
11	ϕ0 Wire Station Coincidence	M2 代表点番号に対応
12	ϕ0 Wire Station Coincidence	M3 代表点番号に対応
13	ϕ1 Wire Station Coincidence	M1 代表点番号に対応
14	ϕ1 Wire Station Coincidence	M2 代表点番号に対応
15	ϕ1 Wire Station Coincidence	M3 代表点番号に対応
16	FW Wire Station Coincidence	M1 代表点番号に対応
17	FW Wire Station Coincidence	M2 代表点番号に対応
18	FW Wire Station Coincidence	M3 代表点番号に対応
19	ϕ0 Strip Station Coincidence	M1 代表点番号に対応
20	ϕ0 Strip Station Coincidence	M2 代表点番号に対応
21	ϕ0 Strip Station Coincidence	M3 代表点番号に対応
22	ϕ1 Strip Station Coincidence	M1 代表点番号に対応
23	ϕ1 Strip Station Coincidence	M2 代表点番号に対応
24	ϕ1 Strip Station Coincidence	M3 代表点番号に対応
25	FW Strip Station Coincidence	M1 代表点番号に対応
26	FW Strip Station Coincidence	M2 代表点番号に対応
27	FW Strip Station Coincidence	M3 代表点番号に対応

D.2　Station Coincidence

図 D.10 のように、Station Coincidence は単純な代表点番号の情報であるので、candidate という概念を無視して 1 word に対して 2 つから 4 つの代表点番号を格納するフォーマットにした。M2/M3 のワイヤおよび FW のストリップは代表点数が少なく、表現に必要な bit 幅が小さく済んだので、特別に最適化している。

D.3　Segment Reconstruction

図 D.11 のように、Segment Reconstruction は candidate を segment に対応させるフォーマットにした。ワイヤーの場合のみ、1 word に収まらなかったため data type を二つに分割している。これはデータタイプの分けるこ

表 C.2: 現状のデータフレーバーの一覧（2）

番号	モジュール	コンテンツ
28-32	ϕ0 Wire Segment Reconstruction	最初の subunit が LSB で、最後尾の subunit が MSB。subunit 単位では{valid flag[0:0], matched layer[1:0], $d\theta$[7:0], M3position[9:0], 2'b0}の 23 bit。
33-37	ϕ1 Wire Segment Reconstruction	最初の subunit が LSB で、最後尾の subunit が MSB。subunit 単位では{valid flag[0:0], matched layer[1:0], $d\theta$[7:0], M3position[9:0], 2'b0}の 23 bit。
38-39	FW Wire Segment Reconstruction	最初の subunit が LSB で、最後尾の subunit が MSB。subunit 単位では{valid flag[0:0], matched layer[1:0], $d\theta$[7:0], M3position[9:0], 2'b0}の 23 bit。
40	ϕ0 Strip Segment Reconstruction	E1 chamber が LSB で、E5 chamber が MSB。その中で unit-0 が LSB で unit-3 が MSB。unit 単位では{valid flag[0:0], $d\phi$[8:0], chamber 内 unit[1:0], unit 内 subunit[0:0], subunit 内 M3ch[2:0], matched layer[1:0],}の 18 bit。
41	ϕ1 Strip Segment Reconstruction	E1 chamber が LSB で、E5 chamber が MSB。その中で unit-0 が LSB で unit-3 が MSB。unit 単位では{valid flag[0:0], $d\phi$[8:0], chamber 内 unit[1:0], unit 内 subunit[0:0], subunit 内 M3ch[2:0], matched layer[1:0],}の 18 bit。
42	FW Strip Segment Reconstruction	その中で unit-0 が LSB で unit-3 が MSB。unit 単位では{valid flag[0:0], $d\phi$[8:0], chamber 内 unit[1:0], unit 内 subunit[0:0], subunit 内 M3ch[2:0], matched layer[1:0],}の 18 bit。
43	ϕ0 Region8 Wire Strip Coincidence	Region8-0 が LSB で Region8-21 が MSB。Region 単位では{valid flag[0:0], p_{T} threshold[3:0], 3'b0, compressed $d\theta$[6:0], 2'b0, compressed $d\phi$[3:0],3b'0}の 24 bit。
44-45	ϕ0 Region32 Wire Strip Coincidence	Region32-0 が LSB で Region32-12 が MSB。その中で{正ミューオン候補 1, 正ミューオン候補 2, 負ミューオン候補 1, 負ミューオン候補 2}の順で格納されている。ミューオン候補単位では{valid flag[0:0], p_{T} threshold[3:0], 3'b0, $d\theta$[6:0], 2'b0, $d\phi$[3:0],3b'0}の 24 bit。$d\theta$ および $d\phi$ は変換後の値。
46	ϕ1 Region8 Wire Strip Coincidence	Region8-0 が LSB で Region8-21 が MSB。Region 単位では{valid flag[0:0], p_{T} threshold[3:0], 3'b0, compressed $d\theta$[6:0], 2'b0, compressed $d\phi$[3:0],3b'0}の 24 bit。
47-48	ϕ1 Region32 Wire Strip Coincidence	Region32-0 が LSB で Region32-12 が MSB。その中で{正ミューオン候補 1, 正ミューオン候補 2, 負ミューオン候補 1, 負ミューオン候補 2}の順で格納されている。ミューオン候補単位では{valid flag[0:0], p_{T} threshold[3:0], 3'b0, $d\theta$[6:0], 2'b0, $d\phi$[3:0],3b'0}の 24 bit。$d\theta$ および $d\phi$ は変換後の値。
49	FW Region32 Wire Strip Coincidence	Region32-0 が LSB で Region32-7 が MSB。その中で{正ミューオン候補 1, 正ミューオン候補 2, 負ミューオン候補 1, 負ミューオン候補 2}の順で格納されている。ミューオン候補単位では{valid flag[0:0], p_{T} threshold[3:0], 3'b0, $d\theta$[6:0], 2'b0, $d\phi$[3:0],3b'0}の 24 bit。$d\theta$ および $d\phi$ は変換後の値。

Data type-0 / Data type-1	EC0/EC1 channel mapping						
cand-0 word-0	# hits per word	layer (3'b)	IsStrip		3rd	2nd	1st least hit ch (0-max301) (8'b)
cand-0 word-1	# hits per word	layer (3'b)	IsStrip		6th	5th	4th
cand-0 word-2	# hits per word	layer (3'b)	IsStrip		9th	8th	7th
cand-0 word-3	# hits per word	layer (3'b)	IsStrip		12th	11th	10th
cand-1 word-1	# hits per word	layer (3'b)	IsStrip		15th	14th	13th
...							

Date type-2	FW channel mapping								
cand-0 word-0	# hits per word (3'b)	layer (3'b)	IsStrip			3rd	2nd	1st least hit ch (0-max124) (7'b)	
cand-0 word-1	# hits per word (3'b)	layer (3'b)	IsStrip			6th	5th	4th	
cand-0 word-2	# hits per word (3'b)	layer (3'b)	IsStrip			9th	8th	7th	
cand-0 word-3	# hits per word (3'b)	layer (3'b)	IsStrip			12th	11th	10th	
cand-1 word-1	# hits per word (3'b)	layer (3'b)	IsStrip			15th	14th	13th	
...									
cand-0 word-0	# hits per word (3'b)	layer (3'b)	IsStrip	5th	4th	3rd	2nd	1st least hit ch (0-max31) (5'b)	
cand-0 word-1	# hits per word (3'b)	layer (3'b)	IsStrip	10th	9th	8th	7th	6th	
cand-0 word-2	# hits per word (3'b)	layer (3'b)	IsStrip	15th	14th	13th	12th	11th	
cand-0 word-3	# hits per word (3'b)	layer (3'b)	IsStrip	20th	19th	18th	17th	16th	
cand-1 word-1	# hits per word (3'b)	layer (3'b)	IsStrip	25th	24th	23rd	22nd	21st	
...									

図 D.9: Trigger Readout Format : Channel Mapping。

Data type-3 / Data type-4 / Data type-5	EC0/EC1/FW wire station coincidence					
cand-0 word-0	# hits per word	coin type	station		2nd	1st least staggeredID (min0-max602) (10'b)
cand-0 word-1	# hits per word	coin type	station		4th	3rd
cand-0 word-2	# hits per word	coin type	station		6th	5th
cand-0 word-3	# hits per word	coin type	station		8th	7th
cand-1 word-0	# hits per word	coin type	station		10th	9th
...						
cand-0 word-0	# hits per word	coin type	station	3rd	2nd	1st least staggered ID (min3-max251) (8'b)
cand-0 word-1	# hits per word	coin type	station	6th	5th	4th
cand-0 word-2	# hits per word	coin type	station	9th	8th	7th
cand-0 word-3	# hits per word	coin type	station	12th	11th	10th
cand-1 word-0	# hits per word	coin type	station	15th	14th	13th

Data type-6 / Data type-7	EC0/EC1 strip station coincidence					
cand-0 word-0	# hits per word	coin type	station	3rd	2nd	1st least staggered ID (0-314) (9'b)
cand-0 word-1	# hits per word	coin type	station	6th	5th	4th
cand-0 word-2	# hits per word	coin type	station	9th	8th	7th
cand-0word-3	# hits per word	coin type	station	12th	11th	10th
cand-1 word-0	# hits per word	coin type	station	15th	14th	13th
...						

Data type-8	FW strip station coincidence						
cand-0 word-0	# hits per word	coin type	station	4th	3rd	2nd	1st least staggered ID (0-62) (6'b)
cand-0 word-1	# hits per word	coin type	station	8th	7th	6th	5th
cand-0 word-2	# hits per word	coin type	station	12th	11th	10th	9th
cand-0word-3	# hits per word	coin type	station	16th	15th	14th	13th
cand-1 word-0	# hits per word	coin type	station	21th	19th	18th	17th
...							

図 D.10: Trigger Readout Format : Station Coincidece。

Data type-9 / Data type-11	EC0/EC1 wire segment reconstruction		
cand-0 word-0	Matched layer(0-3)	delta theta (8'b)	M3 global ch (10'b)
...			

Data type-10 / Data type-12	EC0/EC1 wire segment reconstruction for debug			
cand-0 word-0		priority (6'b)	M2 ch per unit (6'b)	M1ch per unit (8'b)
...				

Data type-13	FW wire segment reconstruction					
cand-0 word-0	Matched layer(0-3)	delta theta (8'b)	M3 global ch (6'b)		M2 ch per unit (5'b)	M1ch per unit (8'b)
...						

Data type-14 / Data type-15 / Data type-16	EC0/EC1/FWのstrip segment reconstruction				
cand-0 word-0	matched layer(0-3)	delta phi (9'b)	M3 global ch (6'b)	M2ch per unit (4'b)	M1ch per unit (5'b)
...					

図 D.11: Trigger Readout Format : Segment Reconstruction。

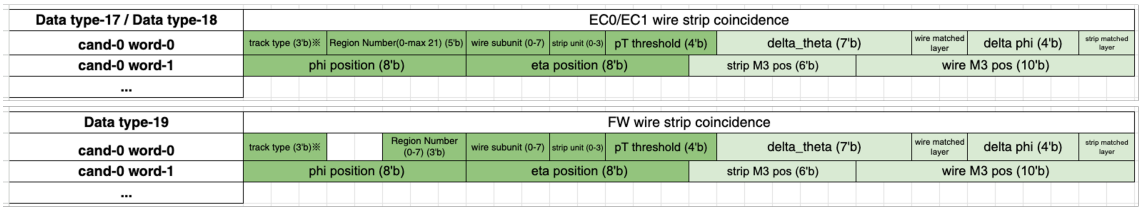

図 D.12: Trigger Readout Format : Wire Strip Coincidence。

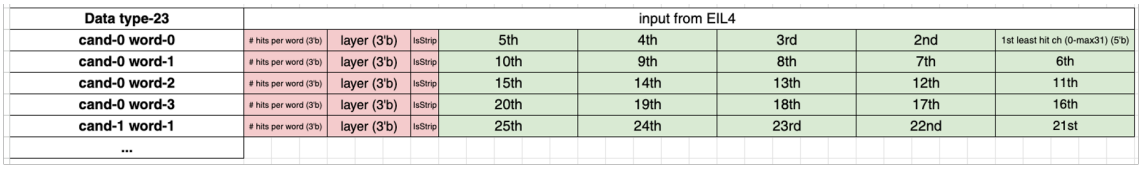

図 D.13: Trigger Readout Format : TGC EIL4。

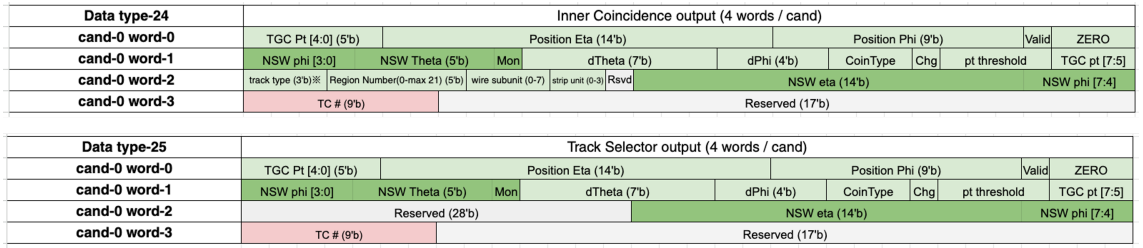

図 D.14: Trigger Readout Format : Inner Coincidence。

とで別々に enable/disable できる実装にしてあり、情報の優先順位として低いものを disable できるようにするためである。

D.4 Wire Strip Coincidence

図 D.12 のように、Wire Strip Coincidence は candidate を segment に対応させるフォーマットにした。ワイヤーの場合のみ、1 word に収まらなかったため data type を二つに分割している。これはデータタイプの分けることで別々に enable/disable できる実装にしてあり、情報の優先順位として低いものを disable できるようにするためである。

D.5 TGC EIL4

図 D.13 のように、TGC EIL4 は TGC BW の Channel Mapping と同様のフォーマットにした。TGC EIL4 のコインシデンスロジックは現在実装中であり、実装でき次第、その中間出力ロジックを新たにデータタイプを追加して読み出すことになる。

D.6 Inner Coincidence および Track Selector

図 D.14 のように、Inner Coincidence と Track Selector は candidate と飛跡候補が対応するフォーマットにした。両者の二つのフォーマットは似通っているが、Track Selector は飛跡候補を並び替えるだけであるからである。Inner Coincidence のみ、 Wire Strip Coincidece で飛跡候補も出力する際の wire/strip の subunit/unit 情報も追加

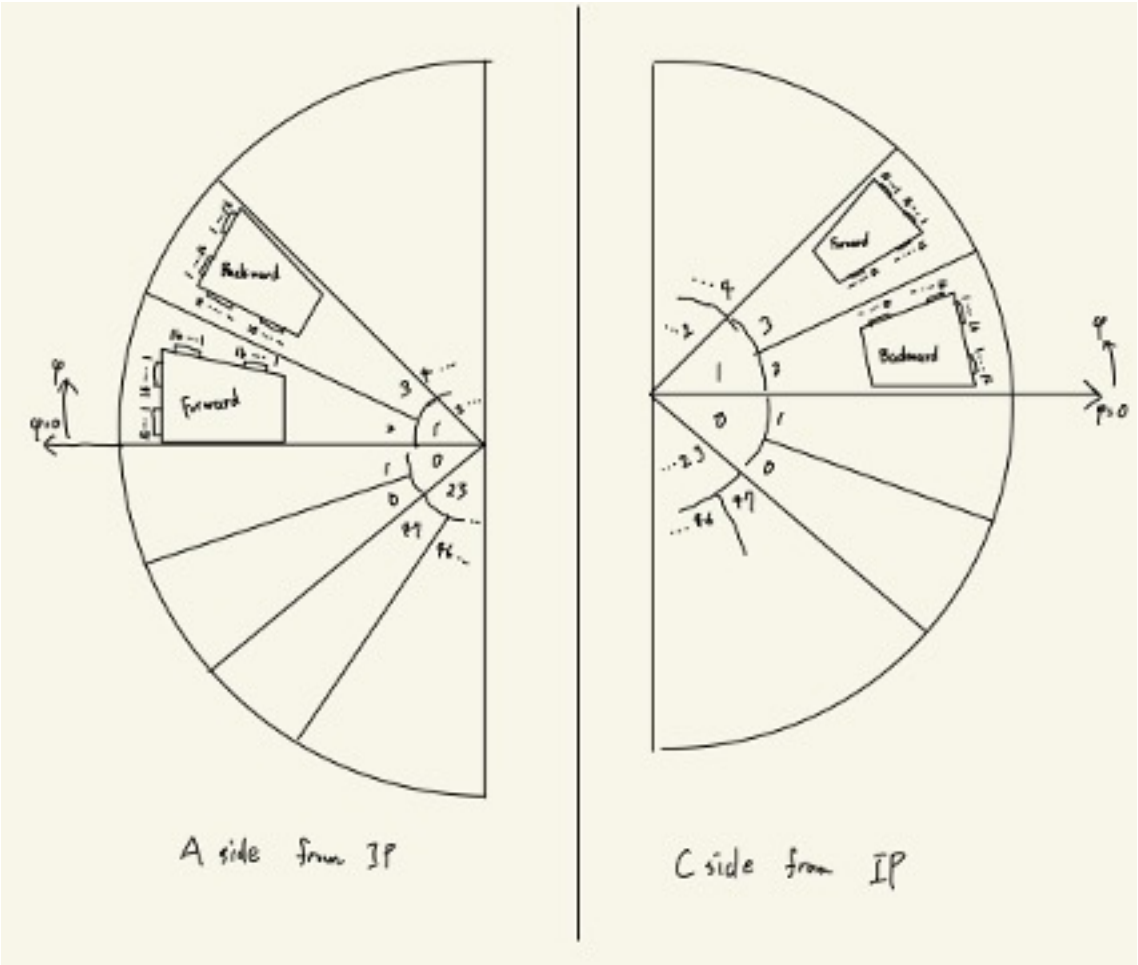

図 E.15: Backward と Forward の構造

している。これは Wire Strip Coincidence - Inner Coincidece 間の座標情報の伝達の検証を目的としている。

E　チェンバー配置としての Backward/Forward の構造とその補正

チェンバーは表と裏に対応する方向性を持ち、Backward と Forward という。図 E.15 に示したように、衝突点（IP）から見て、strip のチャンネル番号が反時計回りのものが Forward で時計回りのものが Backward である。これにより、strip チャンネルは ϕ 座標の順番と同じ順番である箇所と反対の順番である箇所が存在し、その補正を行う必要がある。ϕ 座標で扱われる MC データとチャンネル番号の順番で扱われるエレキの対応関係を図 E.16 に正確にまとめた。

F　プラトー領域以外の Segment Reconstruction での Software Simulator との比較

5.3 節で見たのは、プラトー領域だけの Segment Reconstruction のトリガー効率であった。実際にはプラトー領域以外では Software Simulator との性能に大きな違いがみられているという問題点を発見した。それを図 F.17 と図 F.18 である。Wire の場合は、Segment Reconstruction の段階で開く M1 の窓が 96 代表点のみになっている

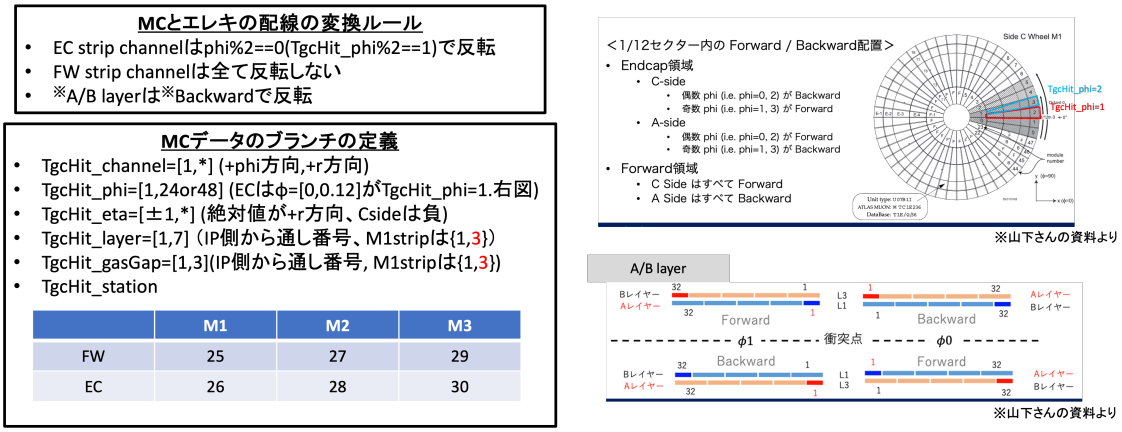

図 E.16: ストリップチャンネルのエレキと ϕ 座標の対応関係

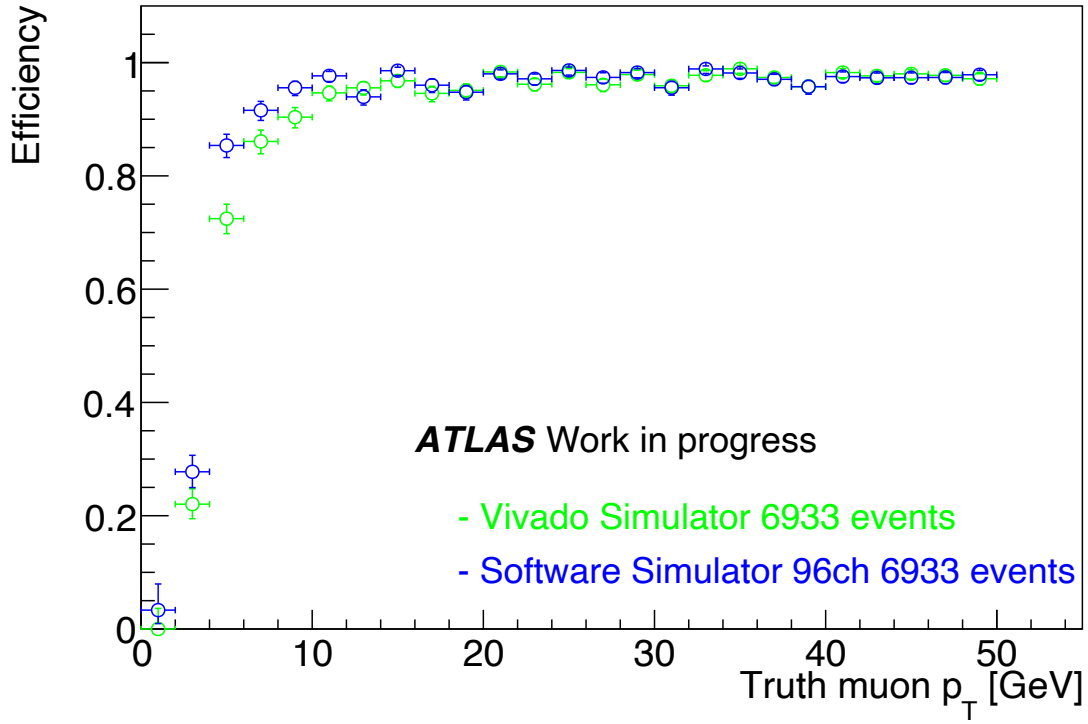

図 F.17: プラトー領域以外の Wire Segment Reconstruction での Software Simulator との比較

が、Software Simulator では 128 代表点の仕様となっているという事情があるため、それを補正して、SOftware Simulator を 96 代表点の仕様にして検証した。しかし、Efficiency の差は以前として残っている。Strip の場合も、Efficiency の差の原因は不明で今後の調査が必要である。

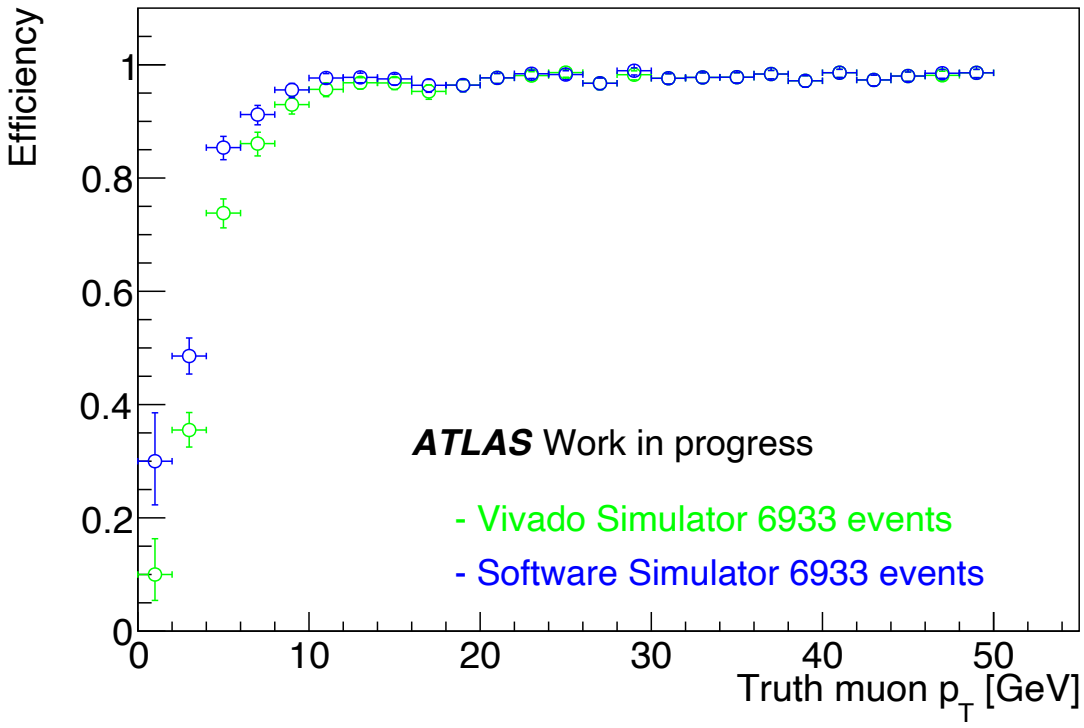

図 F.18: プラトー領域以外の Strip Segment Reconstruction での Software Simulator との比較

引用文献

[1] Navas, S. et al. (2024) “Review of particle physics,” *Phys. Rev. D*, **110**, No. 3, 030001, DOI: 10.1103/PhysRevD.110.030001.

[2] Service graphique, C. (2014) “Overall view of the LHC. Vue d’ensemble du LHC,” URL: https://cds.cern.ch/record/1708849, General Photo.

[3] Aad, G., Bentvelsen, S., Bobbink, G. J. et al. (2008) “The ATLAS Experiment at the CERN Large Hadron Collider,” *JINST*, **3**, S08003, DOI: 10.1088/1748-0221/3/08/S08003, Also published by CERN Geneva in 2010.

[4] ATLAS Collaboration (2024a) “SUSY July 2024 Summary Plot Update,”Technical report, CERN, Geneva, URL: http://cds.cern.ch/record/2904978, All figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2024-014.

[5] Aad, G., Abbott, B. K., Abbott, D. et al. (2023) “Constraining the Higgs boson self-coupling from single- and double-Higgs production with the ATLAS detector using pp collisions at $\sqrt{s} = 13$ TeV,” *Phys. Lett. B*, **843**, 137745, DOI: 10.1016/j.physletb.2023.137745, All figures including auxiliary figures are available at http://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PAPERS/HDBS-2022-03.

[6] CERN (2024) “The HL-LHC project,” https://hilumilhc.web.cern.ch/content/hl-lhc-project, Accessed: 2025-1-5.

[7] ATLAS Collaboration (2017) “Technical Design Report for the Phase-II Upgrade of the ATLAS TDAQ System,”Technical report, CERN, Geneva, DOI: 10.17181/CERN.2LBB.4IAL.

[8] 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，東京大学大学院理学系研究科物理学専攻，URL：https://www.icepp.s.u-tokyo.ac.jp/download/master/m2020_sugizaki.pdf.

[9] 成川佳史（2024）「高輝度 LHC-ATLAS 実験に向けたミューオントリガー論理回路の実装及び性能評価—トリガー系の統合と試験システムの構築—」，修士論文，東京大学大学院理学系研究科物理学専攻，URL：https://www.icepp.s.u-tokyo.ac.jp/download/master/m2023_narukawa.pdf.

[10] Marcelloni, C. (2006) “Installation of the first of the big wheels of the ATLAS muon spectrometer, a thin gap chamber (TGC) wheel. Installation de la première des grandes roues du ATLAS spectromètre à muons, une roue TGC,” URL: https://cds.cern.ch/record/986163.

[11] 赤塚俊一（2017）「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」，修士論文，京都大学大学院理学研究科 物理学・宇宙物理学専攻 物理学第二分野 高エネルギー物理学研究室，URL：https://www-he.scphys.kyoto-u.ac.jp/theses/master/akatsuka_mt.pdf.

[12] 山下恵理香（2023）「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」，修士論文，東京大学大学院理学系研究科物理学専攻，URL：https://www.icepp.s.u-tokyo.ac.jp/download/master/m2022_yamashita.pdf.

[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, 京都大学大学院理学研究科 物理学・宇宙物理学専攻 物理学第二分野 高エネルギー物理学研究室, URL：https://www-he.scphys.kyoto-u.ac.jp/theses/master/mino_mt.pdf.

[14] ATLAS Collaboration (2024b) "Ph2TgcChannelMapping < Atlas < TWiki," https://twiki.cern.ch/twiki/bin/viewauth/Atlas/Ph2TgcChannelMapping, Accessed: 2025-1-4.

[15] ATLAS Collaboration (2023) "phase-II Endcap Sector Logic RAM Table," https://docs.google.com/spreadsheets/d/1yteY64jz4FXAK1qC6_ppIHkdF-rhPAN3RWpxPOhREd4/edit?gid=0#gid=0, Accessed: 2025-1-6.

[16] 河本地弘 (2023)「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実装と検出器全体への拡張」, 修士論文, 京都大学大学院理学研究科 物理学・宇宙物理学専攻 物理学第二分野 高エネルギー物理学研究室, URL：https://www-he.scphys.kyoto-u.ac.jp/theses/master/kawamoto_mt.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, 京都大学大学院理学研究科 物理学・宇宙物理学専攻 物理学第二分野 高エネルギー物理学研究室, URL：https://www-he.scphys.kyoto-u.ac.jp/theses/master/kobayashi_mt.pdf.

[18] ATLAS Collaboration (2024c) "Trigger Readout Dataformat," https://docs.google.com/spreadsheets/d/1tMZV-stnDUDsrSoBsyMhFgc9jz1Mwoa-97NwfdMIBZQ/edit?gid=0#gid=0, Accessed: 2025-1-5.