ATLAS実験アップグレードに向けた新型シリコン検出器モジュールの読み出しシステムの開発

大阪大学大学院 理学研究科物理学専攻
山中卓研究室 博士前期課程２年

遠藤 理樹

February 6, 2012

概 要

欧州原子核研究機構 (CERN) に建設された衝突型円形加速器 Large Hadron Collider (LHC) には汎用粒子検出器である ATLAS 検出器が設置されている。2022 年頃に計画されている LHC のアップグレード計画では、ATLAS 検出器のストリップ型シリコン検出器 (SCT モジュール) が 4088 個から約 7000 個に増える。その増加に対応するために Buffer Control Chip (BCC) と呼ばれる ASIC を導入し、複数個のモジュールからの信号を同時に読み出す。本研究では BCC を経由したモジュールからの信号読み出しと、複数個のモジュールからの信号読み出しが可能な信号読み出しシステムを開発し、開発したシステムが正しく動作していることを示した。

目 次

図 目 次

表 目 次

第1章　序論

1.1　LHC-ATLAS

1.1.1　LHC

LHC は、スイスとフランスとの国境に位置する CERN 研究所にある円周 27km の陽子陽子衝突型円形加速器である。現在、重心系エネルギー 7TeV で稼働し、順調にデータを収集している。質量の起源であるとされている Higgs や階層性問題解決のために導入された超対称性の探索などを行っている。ルミノシティを $5 \times 10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ に上げる LHC アップグレードが 2022 年頃に計画されており、その後 10 年間で積分ルミノシティ $3000\mathrm{fb}^{-1}$ を目指す。図 1.1 に LHC の概要を、表 1.1 に LHC の主要パラメータ (設計値) を示す。表 1.2 に 2011 年における達成値を示す。

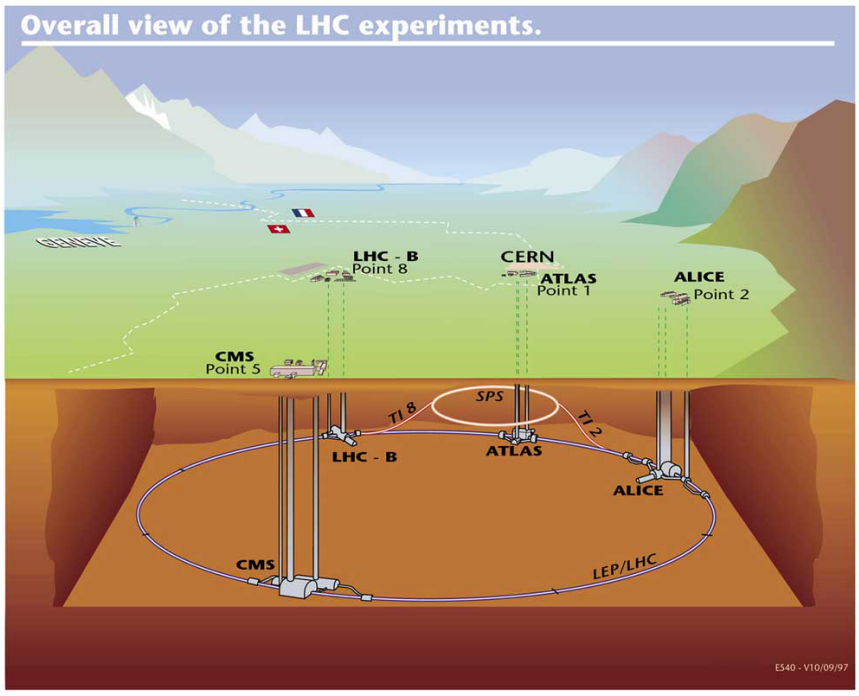

図 1.1: LHC (写真提供 CERN ATLAS 実験グループ)

表 1.1: LHC の主要パラメータ (設計値)

主リング周長	26.7km
重心系エネルギー	7TeV + 7TeV
ルミノシティ	$10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$
バンチ間隔	25nsec (40MHz)
バンチ当たりの陽子数	10^{11}
超伝導双極磁石	1232 台
双極磁石長	14.2m
総局磁石の磁場	8.33T
衝突点でのビーム半径	16μm
バンチ衝突当たりの陽子衝突数	約 20

表 1.2: LHC の主要パラメータ (達成値)

重心系エネルギー	3.5TeV + 3.5TeV
ルミノシティ	$3 \times 10^{33}\mathrm{cm}^{-2}\mathrm{s}^{-1}$

1.1.2 ATLAS 検出器

ATLAS 検出器は、LHC の 4 つあるビーム衝突点の 1 つに ATLAS グループが建設した汎用粒子検出器である。ビーム衝突点から外側に向かって内部飛跡検出器、超伝導ソレノイド磁石、電磁カロリメータ、ハドロンカロリメータ、ミューオン検出器という構成である。図 1.2 に ATLAS 検出器の概要を示す。内部飛跡検出器はビーム衝突点からピクセル型シリコン検出器、SemiConducter Tracker (SCT)、Transition Radiation Tracker から構成される。

1.2 SCT

SCT は、ビーム衝突点近傍に設置されている内部飛跡検出器を構成するストリップ型シリコン検出器である。内部飛跡検出器の外側に設置された超伝導ソレノイド磁石の磁場で荷電粒子は曲げられるため、SCT で測定した飛跡情報から荷電粒子の運動量を測定することができる。図 1.3 が示すようにビームラインに平行な方向に 4 層、ビームラインに垂直な方向に 9 層の検出器からできている。2022 年の LHC アップグレードに合わせて、SCT も、より放射線耐性が高く、かつパターン認識能力の高い検出器に交換される。

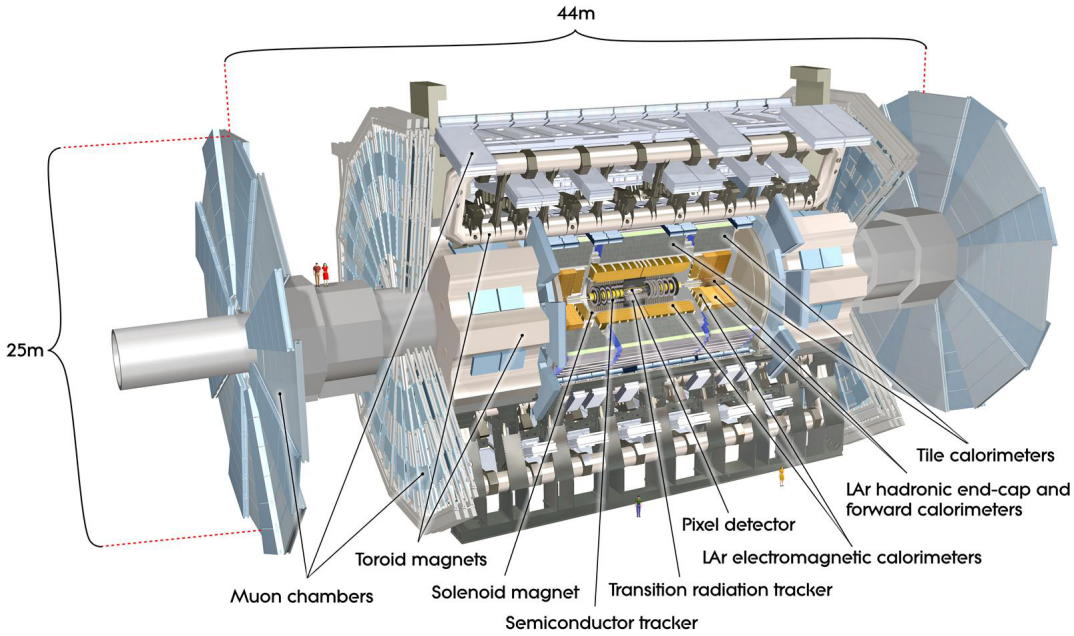

図 1.2: ATLAS 検出器 (写真提供 CERN ATLAS 実験グループ)

1.3 SCT モジュール

SCT の層には、裏表で 2 枚のストリップ型のシリコンセンサー (シリコンストリップセンサー) と読み出し ASIC[1]が搭載された基板から構成されるモジュール (SCT モジュール) が設置されている。高ルミノシティ下での運転でヒット占有率[2]を下げるために、アップグレード用 SCT モジュールでは、シリコンストリップセンサーの長さが従来の 12.8cm から 2.4cm と短くなり、ストリップの間隔は 80μm から 74.5μm となる。また、アップグレード用 SCT モジュールの読み出し ASIC である Atlas Binary Chip Next (ABCN) は 1 個当たり 128 本のシリコンストリップセンサーを読み出し、各ストリップのヒットの有無を返す。この ABCN は 10 個まとめて column と呼ばれる列を成しており、二つの column は hybrid と呼ばれる基板に搭載されている。この hybird 二つが SCT モジュールの片面のシリコンストリップセンサーの信号を読み出し、裏面と表面で合計四つの hybrid が一つのモジュールからの信号を読み出す。図 1.4 に、アップグレード用 SCT モジュールの片面を示す。1 個のモジュールでは、両面で合計 80 個の ABCN が 10240 本のシリコンストリップを読み出すことになる。

アップグレード用 SCT では、同時に読み出すモジュールは 12 個で、裏表で別系統

[1]Application Specific Integrated Circuit (ASIC) とは、特定の用途のための集積回路である。
[2]ヒット占有率 = ヒット数/全イベント数

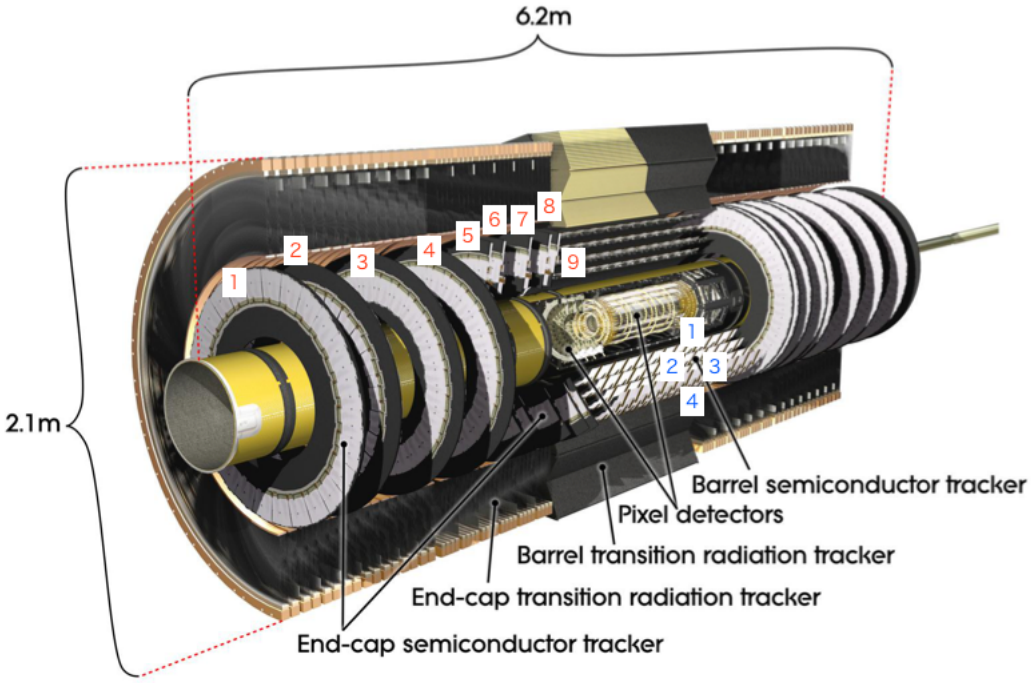

図 1.3: 内部飛跡検出器 (写真提供 CERN ATLAS 実験グループ)。9 層ある End-cap SCT には前方後方で合計 1976 個の SCT モジュールが、4 層ある Barrel SCT では 2112 個の SCT モジュールがそれぞれ設置されている。

の読み出しバスを持つ。モジュール片面あたり 40 個の ABCN があるため、一度に読み出す ABCN の数は 480 個になる。ABCN は自身の識別番号を持ち、読み出すストリップにヒットがあった場合、自身の識別番号とヒットのあったストリップの番号を返す。そのため、複数個の ABCN を同時に読み出す場合は、ABCN の識別番号が重複しないようにしなければならない。ABCN は自身の識別番号が 7bit で表されているため、480 個の同時読み出しでは識別番号の数が不足する。そこで、識別番号の数を増やすために BCC という読み出し ASIC を導入する。BCC は 6bit の識別番号を持ち、ABCN20 個からの信号を読み出す。ABCN と BCC に関しては第 2 章で説明する。

1.4 本研究の目的

ABCN からの信号を読み出すデータ収集 (DAQ) システムは既に開発されているが、BCC を介した ABCN の読み出しシステムは存在していない。そこで、本研究においては、BCC を介して ABCN からの信号を読み出せるシステムを開発することを第一の目的とした。SCT のアップグレードでは複数個のモジュールを同時に読み出

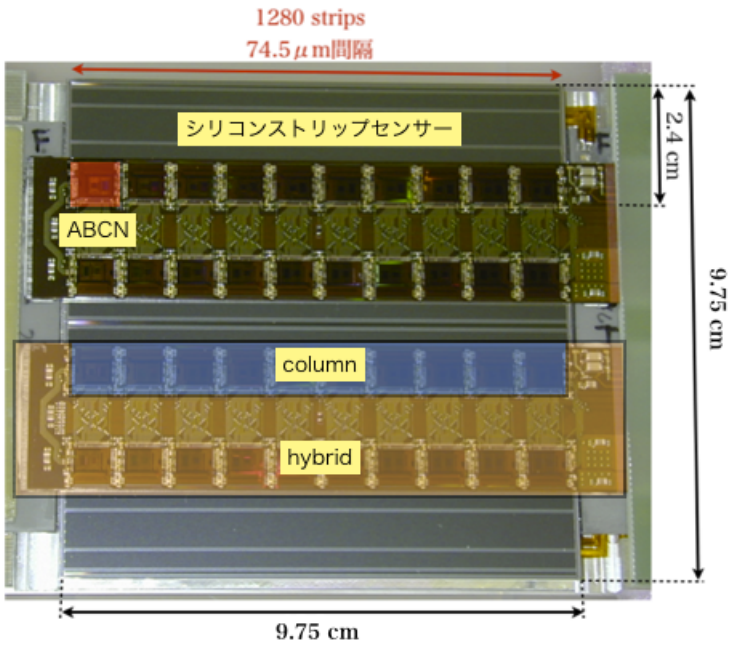

図 1.4: アップグレード用 SCT モジュールの片面。9.75cm 四方のシリコンストリップセンサー上に二つの hybrid が搭載されている。hybrid は二つの column から、column は 10 個の ABCN から構成され、ABCN は 2.4cm のシリコンストリップセンサーを読み出す。

さなければならないが、複数個のモジュールの同時読み出しはこれまでになされてないため、複数個のモジュールの同時読み出しをすることを第二の目的とした。複数個のモジュールからのデータは量が多いため、テストに用いる DAQ システムには速い処理速度が必要である。その足がかりとして DAQ の開発に際してシステムの処理速度向上を試みた。

第2章　信号読み出しASIC

ABCNは、シリコンストリップセンサーからの信号の読み出し用に設計されたASICである。また、BCCはABCN20個からの信号をまとめて読み出すためのASICである。これら二つのASICについて説明する。

2.1　ABCN

ABCN [1] [2] はシリコンストリップセンサーからの信号を増幅、波形整形をした後、閾値を超えたかどうかでヒットの判定をし、そのヒットの有無をバイナリでバッファメモリに保存する。L1トリガー[1]を受け取ると自身の識別番号やヒットのあったストリップの番号などの情報を出力する。ABCNには閾値の値などのパラメータを保存するためのレジスタがあり、これらには外部からビットストリームをABCNのcommand線 (COM) に送って書き込む。ABCNはこのレジスタ設定に基づいて動作する。図2.1にABCNのブロックダイアグラムを示す。外部からの入力信号線には次のものがある。

- BCO：LHCのビーム衝突頻度である40MHzのクロックを入力する。
- DCLK：ABCNの出力ビットストリームを同期させるクロックを入力する。
- L1：L1トリガーを入力する。
- RESETB：ABCNを電源入れた直後の状態にリセットするための信号を入力する。
- COM：レジスタ書き込みなどのためのビットストリーム入力。
- DATAOUT：ABCNからの出力線。

以下で、センサーからの信号読み出しにおける重要な機能について説明する。

[1]ATLAS検出器では事象を選別するためのトリガーは三段階で構成される。最初の一つ目をL1トリガーと呼ぶ。

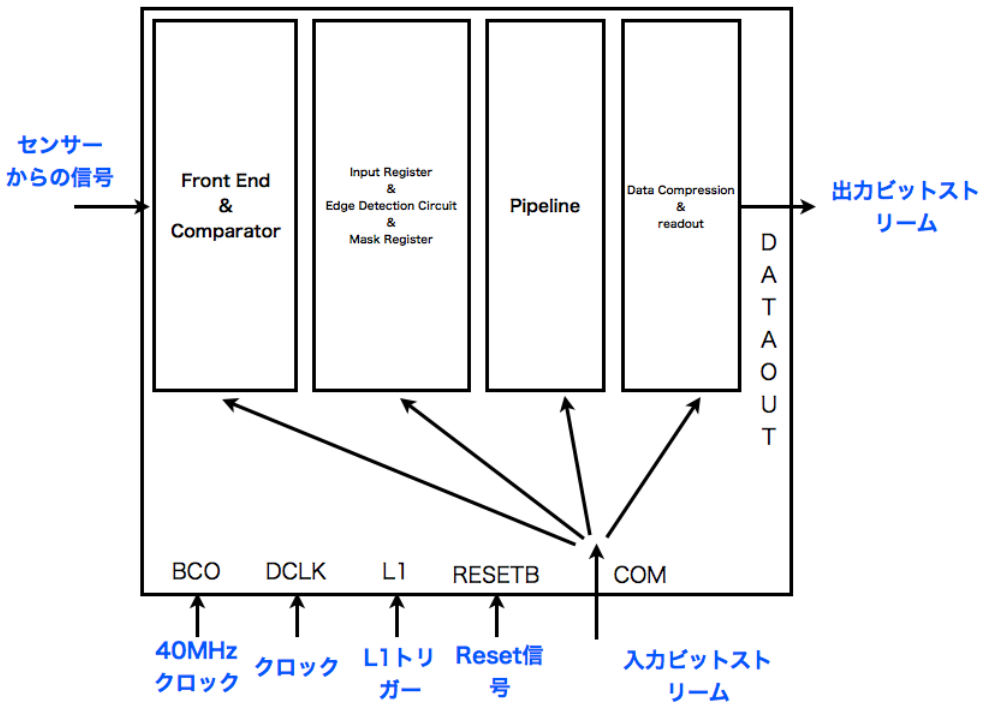

図 2.1: ABCN ブロックダイアグラム

2.1.1 Front End & Comparator

ABCN の Front end は正負両方の信号を受け取ることが出来る。クロストークは 5%以下である。Front End に入力された信号を Preamplifier が増幅し、Shaper が波形整形を行う。ゲインは 100mV/fC、ノイズは $370 + 76.7$ C [electrons] である [3] (C は電気容量 [pF])。増幅された信号が設定した閾値より大きい場合に Comparator が信号を出力する。

2.1.2 Input Register & Edge DetectionCircuit & Mask Register

Input Register は Comparator からの信号を幅 128bit、深さ 1bit のレジスタに保持し、40MHz のクロックに同期して信号を出力する。Edge Detection Circuit は設定し

たヒットの検出方法に基づいて、Comparator からの信号からヒットの有無を判定する。また、Mask Register は指定したストリップからの信号を遮断する (mask する)。

2.1.3 パイプライン

L1 トリガーは対応するバンチ衝突事象 (イベント) に対して 2 から 3μs 遅延する。ABCN は自身が読み出す 128 本のストリップからのヒットの有無を LHC におけるバンチ衝突事象間隔である 25nsec 毎に保存するパイプラインと呼ばれるバッファを持っている。このパイプラインは深さが 256bit あるため $25nsec \times 256 = 6.7 \times 10^3 nsec = 6.7\mu sec$ 前までのヒット情報を保持できる。L1 トリガーを受け取った ABCN は予め設定された遅延時間 (L1 delay) の分だけさかのぼってパイプラインに書き込まれたヒット情報を出力する。この時、L1 トリガーの遅延に該当するイベントと時系列的に前後のイベントのヒット情報の合わせて三つ分のヒット情報を出力する。

2.1.4 Data Compression

パイプラインには 128 本のストリップ全てのヒット情報が保持されるが、L1 トリガーを受け取ってヒット情報を出力する際には、ヒットのあるストリップのみのアドレスとヒット情報を送信することによりデータ量を減らす。

2.1.5 Readout

ABCN に対して、Master、Slave、End の三つの役割を与えることが出来る。DAQ は、Master を先頭に、次に Slave をはさみ、End までの ABCN から順番にデータを読み出す。三つの役割の割り振りは設定用レジスタの値で決定する。Master と End の役割を 1 個の ABCN に重複して設定した場合、column から読み出す ABCN はそれ 1 個だけになる。

2.1.6 Registers

ABCN では様々な回路がセンサーからの信号の読み出しやデータ圧縮などの処理を行っている。その各処理を外部からコントロールするために、様々な設定用パラメータがレジスタに保持される。複数個あるレジスタにはそれぞれアドレスが割り振られている。

2.1.7 Calibration Pulse

ABCN は自分で Front End に Calibration 用の信号を入力できる。

2.1.8 モード

ABCN には以下のモードがあり、モードによって L1 トリガーを受け取ったときに出力するデータが変わる。

- Send_ID

  このモードの時、L1 トリガーを受け取るとあるレジスタの値を出力する。

- Enable Data Taking

  このモードの時、L1 トリガーを受け取るとヒット情報を出力する。

- Read Register

  このモードの時、L1 トリガーを受け取ると指定したレジスタの値を出力する。

- Clock Feed Through

  このモードの時、L1 トリガーの入力とは無関係に、ABCN に入力したクロックをそのまま出力し続ける。clock feed through というレジスタの値が 0 の時に ON、1 の時に OFF となる。ABCN の電源を入れた時、仕様上は自動的に Send_ID モードになるが、レジスタ clock feed through が 0 で初期化されているため、電源を入れた後は結局このモードになる。

2.1.9 Command

ABCN へ送信するビットストリームを command と呼び、以下の、L1 トリガー command、リセット、レジスタ設定用 command (ABCN の各レジスタの設定を行う)、モード変更 command、Calibration Pulse 発行用 command がある。リセットは、レジスタ以外の全てのバッファが持つ値をゼロにリセットする Soft Reset と、ビーム衝突回数をゼロにリセットする BC Reset の 2 種類がある。例として、ABCN に Calibration Pulse を発行させて、ヒットデータを取得する時に送信する command とその順番を記す。

1. Soft Reset を送信してレジスタ以外のバッファをリセットする[2]。

[2]ABCN は電源入力直後において、外部から最初に送信された command を正しく受信しないことがある。原因はわからないが、正しく受信しなくとも問題のない command (今回は Soft Reset) を最初に送信すれば、次の本命の command は正しく受信する。

2. レジスタ設定用 command を送信して ABCN のレジスタを設定する。

3. Soft Reset を送信してレジスタ以外のバッファをリセットする。

4. Enable Data Taking モードに入るための command を送信してヒットデータを出力するモードに変更する。

5. Calibration Pulse 発行用 command を送信し、テスト電荷を Front End に入力する。

6. L1 トリガー command を送信して、ABCN にヒットデータを出力させる。

2.1.10 データフォーマット

ABCN がデータとして出力するビットストリームには、データの始まりを示すヘッダとデータの終わりを示すトレーラーの間に、ヒットのあったストリップの address やヒットパターンなどの情報がある。表 2.1 はビットストリームの全体を示す。最初にデータの始まりを意味する Preamble (11101) がある。次に、Master の ABCN から、DT (常にゼロ)、受け取った L1 トリガーの数を示す LVL1、そしてバンチの衝突回数を示す BC が続く。その後に Sep (1) を置いて、ヒット情報の Data Block が、Mater から順に Slave、最後に End まで読み出す ABCN の数だけ並ぶ。最後に、End の ABCN からデータの終わりを意味する Trailer (1 0000 0000 0000 0000) が置かれる。

表 2.1: データフォーマット、n : 4bit 長で、受け取った L1 トリガーの回数を表す。b : 8bit 長で、バンチ衝突回数を表す。

Preamble	DT	LVL1	BC	Sep	Data	Data	Data	Trailer
11101	0	nnnn	bbbbbbbb	1	block_1	.....	block_n	1 0000 0000 0000 0000

次に Data Block について説明する。Data Block は ABCN column の Master から、Slave をはさんで、End までの ABCN が出力するデータが並んでいる。Data Block の内容は ABCN のモードによって変わるが、ここでは ABCN が Enable Data Taking の時についてのみ説明する。ABCN が出力するデータの内容は、隣り合うストリップにヒットがなくそのストリップだけにヒットがあることを示す Isolated Hit Data Packet (Iso) と、隣り合うストリップの 2 本以上にヒットがあることを示す Non Isolated Hit Data Packet (Non_Iso) の 2 種類から構成される (Iso と Non_Iso の区別方法は節 3.4.1 を参照)。表 2.2 のように、Iso と Non_Iso が並ぶ。表 2.3 は Iso を示し、表 2.4 は Non_Iso を示す。

表 2.2: Data Format

Data Packet	Data Packet	...	Data Packet
Iso or Non_Iso	Iso or Non_Iso	...	Iso or Non_Iso

表 2.3: Isolated Hit Data Packet

Header	Chip Address	Channel Address	Sep	Hit Pattern
01	aaaaaaa	ccccccc	1	ddd

- a : 7bit 長で、ABCN のアドレスを表す。
- c : 7bit 長で、ヒットのあったストリップのアドレスを表す。
- d : 3bit 長で、ヒットパターンを表す。真ん中の bit が L1 に対応するイベントのヒットで、最上位 bit が一つ前のイベントのヒット、最下位 bit が一つ後のイベントのヒットをそれぞれ表している。

2.2 BCC

BCC [4] [5] は hybrid 一個分に相当する二つの ABCN column (合計で 20 個の ABCN) とビットストリームの送受信を行う。BCC は、前半が BCC 用のビットストリーム (BCC com) で後半が ABCN 用のビットストリーム (ABCN com) で構成されるビットストリームを COM で受信すると、BCC com を取り除いた ABCN com を ABCN の COM に送信する。また、BCC は二つの ABCN column から受信したビットストリームをマルチプレクス (後述) したものと自身の識別番号 (address) を一緒に DATAOUT から出力する。図 2.2 に BCC の概要を示す。外部の読み出しデバイスと BCC の間の信号線について説明する。

- BCO：LHC のビーム衝突頻度の 40MHz のクロックを入力する。
- COM：command を入力する。

表 2.4: Non Isolated Hit Data Packet

Header	Chip Address	First Hit Channel Address	Sep	First Strip Hit Pattern	Sep	Next Strip Hit Pattern	Sep	...	Sep	Last Strip Hit Pattern
01	aaaaaaa	ccccccc	1	ddd	1	ddd	1	...	1	ddd

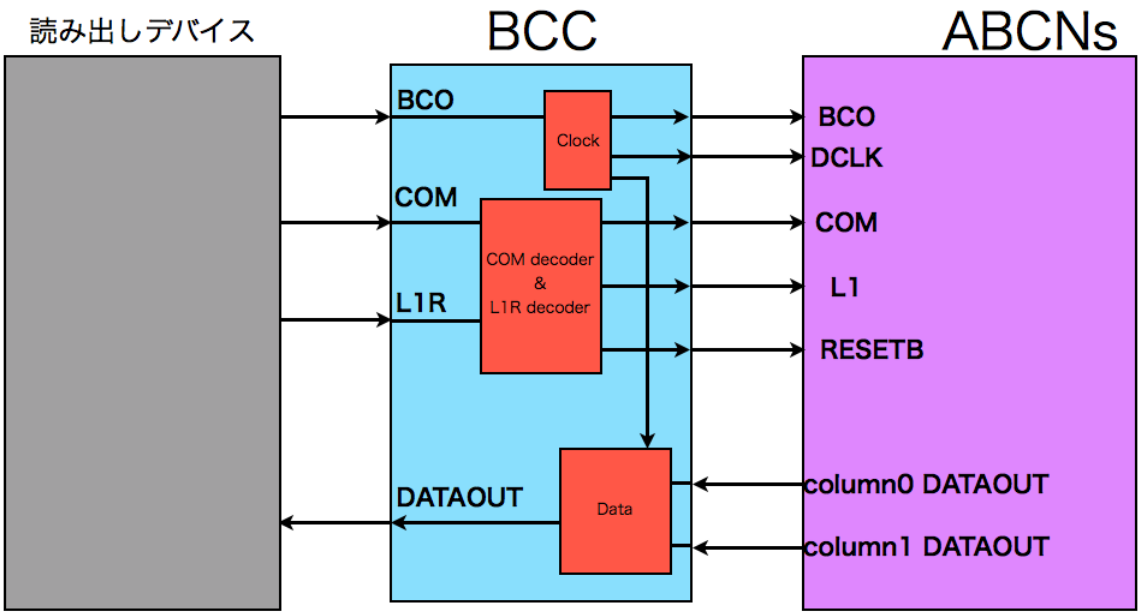

図 2.2: BCC 概要

- L1R：L1 と RESETB を入力する。(節 2.1 を参照)
- DATAOUT：BCC からの出力。

2.2.1 clock

BCC は、BCO からの入力クロック (40MHz) をもとに、ABCN の BCO 用、DCLK 用、BCC の出力データ同期用の三つのクロックを生成する。ABCN の BCO へは 40MHz を入力する。ABCN の DCLK へは、BCO の入力クロックから 2 倍の振動数のクロック (80MHz) を生成し、40MHz と 80MHz のどちらかを出力する。BCC からの出力データが同期するクロックとして、DCLK への入力クロックに応じて 80MHz と 160MHz のどちらかを BCC 内部に出力する。

2.2.2 L1R decoder & COM decoder

COM、L1R で受け取った command から、レジスタ設定用のパラメータや ABCN へ送信するビットパターンを取り出すことを decode という。

L1R decoder は L1R で受信した L1 や Reset 用 command を decode して ABCN へ送信したり、BCC 自身の内部リセット信号を得る。表 2.5 は L1R Decoder が L1R への入力ビットストリームをどのように decode して、出力ビットストリームをどの線に送信するのかを示したものである。X は不定値 (0 or 1) を表す。

表 2.5: L1R Decoding

入力ビットストリーム	出力信号線	出力ビットストリーム
X0100	ABCN_L1	100
X0110	ABCN_L1	110
01110	ABCN_RESETB	111
01111	internal BCC reset	

COM decoder は BCC の COM に入力されたビットストリームを decode して、BCC 自身の持つレジスタの設定や ABCN へ送信する ABCN command を取り出す。BCC の COM への入力は、どの address の BCC に対しても送信する command (Broadcast commands) と特定の address の BCC に送信する command (BCC Addressed Transfer) に分類される。表 2.6 は、COM Decoder が COM への入力ビットストリームをどのように decode して、出力ビットストリームをどの線に送信するのかを示したものである。

表 2.6: COM Decoding

command	入力ビットストリーム					出力信号線	出力ビットストリーム
Broadcast L1Command	110					ABCN_COM	0110
Broadcast Fast Command	101	CCCC				ABCN_COM	CCCC
BCC Addressed Transfer	111	AAAAAA	TTT	LLLLLLLLLLLLLLL	DDD...DD		
- Send Data on ABCN_COM			000	LLLLLLLLLLLLLLL	DDD...DD	ABCN_COM	DDD...DD
- Send Data on ABCN_L1			001	LLLLLLLLLLLLLLL	DDD...DD	ABCN_L1	DDD...DD
- Send Data on ABCN_RESETB			010	LLLLLLLLLLLLLLL	DDD...DD	ABCN_RESETB	DDD...DD
- Write BCC Config Reg			101	RRRRRRRRRRRRRRR			
- Readback BCC Config Reg			100			DATAOUT	1RRRRRRRRRRRRRRR1
- Read BCC ID Reg			111			DATAOUT	10000000000AAAAAA1

- C: 4bit で表される Command。そのまま、ABCN へ送信される。
- A: 6bit で表される BCC の識別番号 (address)。111111 の場合は全ての BCC 共通の address を意味する。
- T: データの出力先を示す。MSB が 1 の場合は BCC 内部で信号が処理される。

- D: ABCN へ送信するビットストリーム。
- L: ABCN へ送信するビットストリームのビット長を指定する。
- R: BCC のレジスタへ書き込むパラメータ。

2.2.3 Registers

ABCN と同様に BCC もレジスタを持ち、その設定に基づいて動作する。

2.2.4 Data

BCC は、二つの ABCN column からのデータ (それぞれ DATA0、DATA1 とする) を読み出し、それらを 1bit ずつ交互に出力する。これをマルチプレクスするという。マルチプレクスは、ABCN の出力が同期しているクロックの 2 倍のクロックに同期して行われる。BCC は、マルチプレクスしたデータの前に 4bit ゼロを置いて自身の address を並べたもの (BCC DATA) を DATAOUT から出力する。図 2.3 に ABCN から BCC への入力ビットパターンと、BCC から外部の読み出しデバイスへの出力ビットパターンを示す。

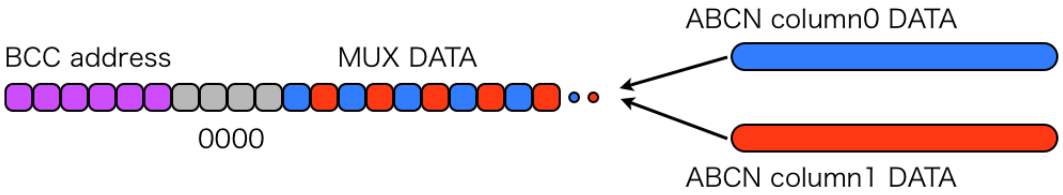

図 2.3: ABCN から BCC への入力ビットパターンと、BCC からの外部の読み出しデバイスへの出力ビットパターン

第3章　DAQシステム

この章では、本研究で開発したDAQシステムに関して説明する。モジュールからの信号は大容量FPGAを搭載した汎用読み出しボードとPCで読み出す構成とした。本システムは田窪洋介氏 (KEK) と Sergio Gonzalez Sevilla 氏 (University of Geneva) との共同開発である。

3.1　概要

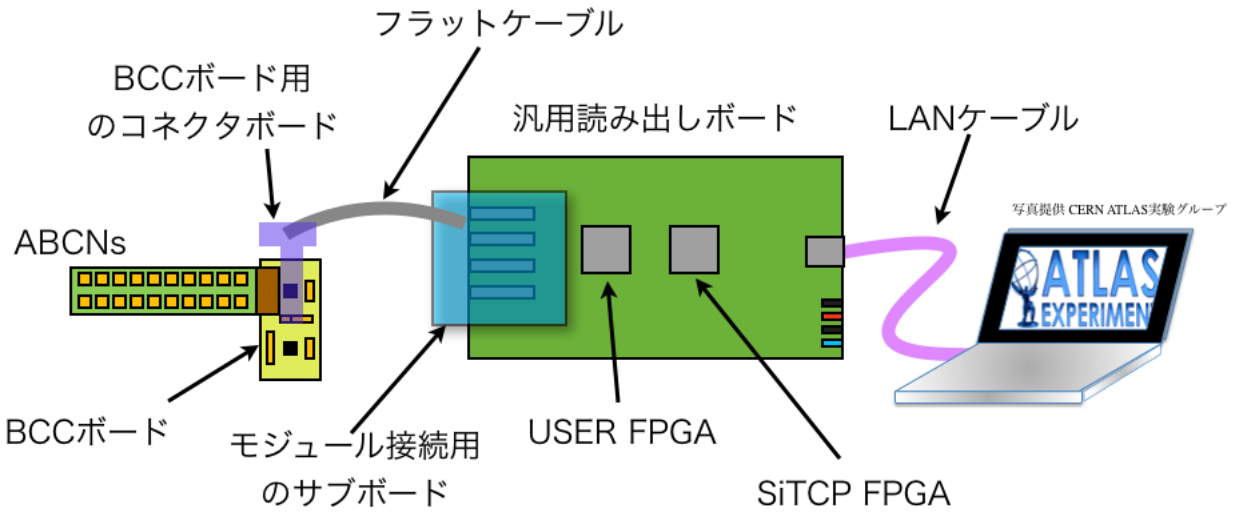

図 3.1: DAQ システムの概要、ABCN20 個を BCC1 個経由で読み出す時の構成

図3.1にDAQシステムの概要を示す。ABCN20個からの信号を、BCC1個とSEABASと呼ばれる汎用読み出しボードを経由して PC で読み出す。汎用読み出しボードには PC との通信用の FPGA (SiTCP FPGA) とモジュールからの信号の読み出し用の FPGA (USER FPGA) が搭載されている。USER FPGA は、ABCN/BCC のレジスタ設定用 command、モード変更用 command、L1 トリガー command などを ABCN/BCC に送信し、ABCN/BCC から出力されたデータを受信する。SiTCP

FPGA は、イーサネット経由で PC と TCP/IP 通信を行う。PC からの ABCN/BCC への command を受信し、ABCN/BCC から出力されたデータを PC に送信する。

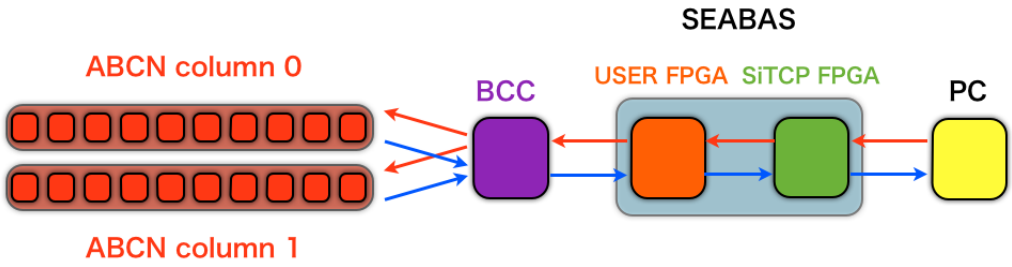

図 3.2: 信号の流れ

図 3.2 は、図 3.1 の重要な箇所を示した図で、BCC 経由で ABCN からの信号を読み出すシステム全体の信号の流れを示す。ABCN は 10 個ずつ column というグループにまとめられている。赤色の矢印は PC から ABCN へ流れている信号を表し、command 送信の流れである。PC からの出力信号が SiTCP FPGA を経て USER FPGA へ送信される。USER FPGA は、PC からの信号を元にして BCC へ command を送信する。青色の矢印は、ABCN から PC へ流れている信号を表し、ABCN/BCC からのデータの流れである。ここで、二つの ABCN column が出力するデータをそれぞれ DATA0 と DATA1 とする。DATA0 と DATA1 を受信した BCC は、BCC DATA (節 2.2.4 を参照) を出力する。USER FPGA は、BCC から送られた BCC DATA を SiTCP FPGA を介して PC へ送信する。

3.2 Soi EvAluation BoArd with Sitcp (SEABAS)

Soi EvAluation BoArd with Sitcp (SEABAS [6]) は、Silicon-On-Insulator (SOI) と呼ばれる技術を用いたピクセル型半導体検出器を開発している KEK 測定器開発室が開発した汎用読み出しボードで、本 DAQ システムの核ある。SEABAS 上の I/O を整えれば、様々なデバイスの読み出しが可能となる。TCP/IP 通信でネットワーク経由のデータの送受信を可能にする SiTCP を実装した SiTCP FPGA、デバイス制御のためにユーザーが自由にカスタマイズできる USER FPGA、そして、本システムでは使用していないが DAC、ADC、NIM I/O を搭載している。図 3.3 に SEABAS の写真を示す。

SiTCP FPGA

Xilinx Virtex-4 (XC4VLX15-10FF668)
100BASE-T 規格の SiTCP

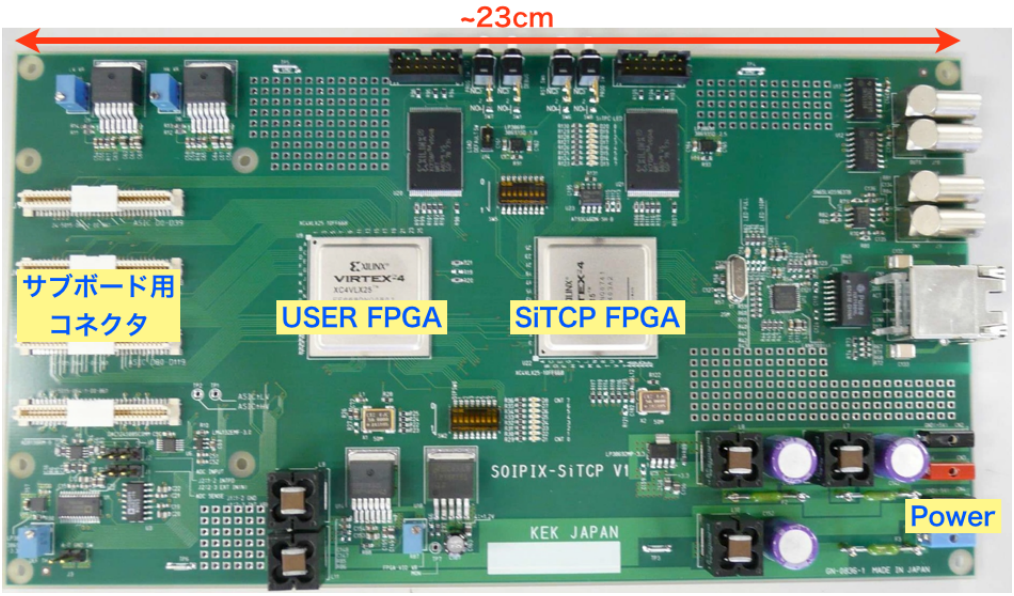

図 3.3: SEABAS

USER FPGA

Xilinx Virtex-4 (XC4VLX25-10FF668)
120 本の I/O がサブボード接続用コネクタに配線されている。接続した外部デバイスとのデータ送受信の他、SAEBAS 上の DAC、ADC、NIM I/O の制御も可能。

Power

± 5V を入力する。

3.3 ファームウェア

3.3.1 SiTCP FPGA

SiTCP では、TCP と UDP による通信が可能である。TCP 通信は専用のインターフェース (TCP I/F) を用いて行う。TCP I/F は同期 FIFO のインターフェースと同様の構造になっているため、FIFO のデータを読み書きするのと同様の方法で使用できる。SiTCP FPGA からデータを読み出す場合は、SiTCP FPGA がデータを持っていることを示す data valid 信号がある時に、read enable 信号を SiTCP FPGA に送ってデータを読み出す。SiTCP FPGA にデータを送る場合は、SiTCP FPGA に空き容量がある時 (almost full 信号がない時) に、write enable 信号とデータを SiTCP

FPGA に送る。UDP 通信も Remote Bus Control Protocol (RBCP [7]) と呼ばれる専用のインターフェースを使う。RBCP を用いて PC から USER FPGA 上の専用のレジスタへのアクセスを行う。複数個用意されているレジスタにはそれぞれアドレスが割り振られており、PC からデータを送信する際にどのレジスタにデータを書き込むのかをアドレスで指定すると、指定されたレジスタにデータが書き込まれる。一般的に TCP 通信よりも UDP 通信の方がデータ転送速度は速いが、UDP 専用に用意されたレジスタへのアクセスに時間がかかるため、SEABAS での UDP 通信は TCP 通信よりもデータ転送速度が遅い。本 DAQ システムでは ABCN/BCC に送信するパラメータの転送は全て TCP で行う。USER FPGA に実装したファームウェア用のパラメータは UDP で送信する。PC からの信号の受信や、PC への信号の送信は、SiTCP FPGA が自動的に処理する。これらの処理を行うファームウェアは、KEK 内田智久氏によって開発されたもので、SEABAS ユーザーは特別な要求がなければ変更を加える必要はない[1]。本研究でも内田氏開発のものをそのまま用いた。

3.3.2 USER FPGA

本研究では、USER FPGA 用のファームウェアを開発した。DAQ システムの中で USER FPGA が関わる部分は、SiTCP FPGA から信号を受け取りそれを元に BCC へ command を送信することと、BCC からの信号を受信してそれを SiTCP FPGA へ送信することである。PC は、ビットストリーム送信時に、BCC の COM と L1R どちらの信号線を使うのか指定するパラメータ (job_index) も送る。これらの信号を受け取った USER FPGA は、job_index で指定された信号線にビットストリームを送信する。

BCC からのデータ受信では、データを一旦保存しておくために、読み出す BCC の数に対応して 32 個の FIFO を USER FPGA 上に用意しておく。PC は UDP 通信で FIFO の読み出しを制御するためのパラメータ (bcc_mask) を送信し、読み出す FIFO の指定と読み出しの順番を指定する。データ転送可能な場合、FIFO からは SiTCP FPGA を介して PC へデータが送信される。以上の機能を verilog HDL で実装した。図 3.4 にファームウェアのブロック図を示す。

以下で図 3.4 に示した各モジュールについて説明する。

TCP reader

PC から TCP で送信されたビットストリームを SiTCP FPGA を介して受け取る。受け取ったビットストリームから、BCC に送信するビットストリームと job_index を

[1] SiTCP FPGA に搭載されたファームウェアのソースコードは著作権の関係上非公開のため、改変は基本的に推奨されない。

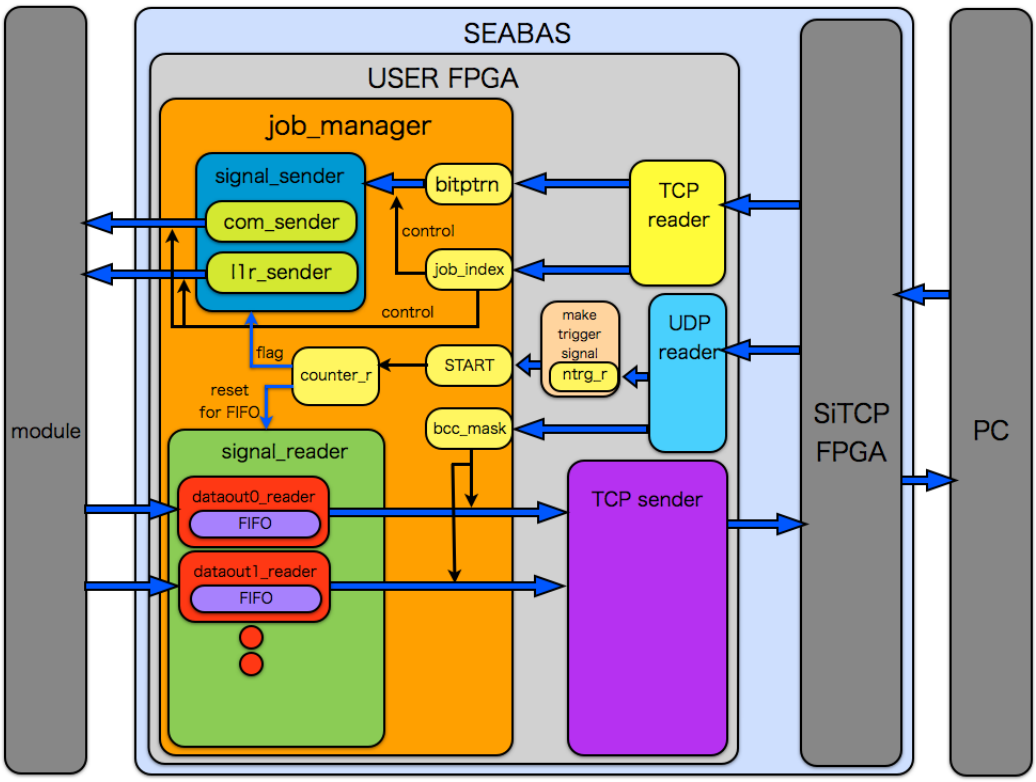

図 3.4: ファームウェアのブロックダイアグラム

抜き出す。ビットストリームは bitptrn というレジスタに保持、job_index は job_index というレジスタにそれぞれ保持する。

UDP reader

PC が送信した FIFO の読み出し制御やファームウェア上の処理で用いるパラメータを UDP 通信で受信する。bcc_mask は 32 個の FIFO の bitmask 用のパラメータである。clk_mode は BCC からの出力データ読み出しを同期させるクロックの周波数を決定する。L1 トリガーの数は ntrg_r というレジスタに保持される。

make trigger signal

DAQ を開始するためのフラグ (DAQ 開始フラグ) を ntrg_r の数だけ生成する。DAQ 開始フラグは後述の job_manager に渡す。

signal_sender

signal_sender は、BCC へビットストリームを送信する機能を持つ。L1R_sender と COM_sender の二つがあり、L1R_sender は BCC の L1R にビットストリームを送信し、COM_sender は BCC の COM にビットストリームを送信する。どちらのモジュールを使うかは job_index により制御する。各 sender には BCC へ送信するビットストリーム用のレジスタがあり、bitptrn の値を COM_sender のレジスタと L1R_sender のレジスタのどちらに入れるかも job_index で決定する。後述の job_manager から送信開始フラグを受け取ると信号線へのビットストリーム送信を開始する。

signal_reader

signal_reader には BCC から読み出したデータ保存用の FIFO があり、1 個の signal_reader が 1 個の BCC から一対一対応でデータを受け取る。signal_reader は、dataoutX_reader (X = 0,1,2,,,,31) の 32 個あり、最大 32 個の BCC からデータを受信できる。FIFO に保持されたデータは順次 TCP sender へ送られる。BCC からの出力データの読み出しは clk_mode によって決定した周波数のクロックに同期する。

job_manager

job_manager は、signal_sender や signal_reader を制御する USER FPGA の動作の要となるモジュールである。TCP reader から受け取った job_index の値に基づき COM_sender か L1R_sender のどちらを使用するか選び、なおかつ、選んだ方の送信ビットストリーム用のレジスタに bitptrn に保持されているビットストリームを送る。job_manager は counter_r という 40MHz のクロックに同期するカウンタをもっている。counter_r は make trigger signal から DAQ 開始フラグを受け取ると、dataoutX_reader の FIFO リセット用フラグの作成や、COM_sender や L1R_sender のビットストリーム送信開始フラグの作成を制御する。また、bcc_mask の値に対応した signal_reader の FIFO に入っているデータを順番に TCP sender へ送信する。

TCP sender

TCP_sender は、32 個の dataoutX_reader の FIFO に入っているデータを SiTCP FPGA を介して TCP 通信で PC へ送信する。dataoutX_reader の FIFO が読み出し可能、かつ SiTCP FPGA と PC が通信確立、かつ SiTCP FPGA に空き容量がある場合にデータを送信する。

3.4 DAQ用ソフトウェア

PC 上で走らせる DAQ 用のソフトウェアを C、C++、ROOT を用いて開発した。DAQ は、大きく分けると、ソケットを用いて ABCN/BCC のレジスタ設定用のパラメータを SEABAS へ送信すること、ABCN/BCC からのデータを SEABAS から受信すること、そしてビットストリームである受信データから ABCN/BCC のデータフォーマットに従ってヒット情報を取り出すことの三つの機能を持つ。図 3.5 は DAQ のフローチャートである。以下にソフトウェアが行う処理の手順を示す。

1. ABCN/BCC のレジスタに設定するパラメータを記述した xml 形式のファイルを読み込む。

2. ソケットプログラムにより SEABAS の SiTCP FPGA と TCP 通信を確立する。

3. 先に読み込んだ xml ファイルの ABCN/BCC 用パラメータと job_index を TCP 通信で、同じく xml ファイルから読み込んだ bcc_mask や L1 トリガーの数などのファームウェア用のパラメータを UDP 通信で送信する。

4. BCC を介して ABCN に L1 トリガーコマンドを送信する。L1 を受け取った ABCN はヒット情報であるビットストリームを出力する。ABCN の出力ビットストリームは、SEABAS を経由して PC の TCP ソケットへ送信され、カーネルバッファに保存される。ファームウェア上で設定した L1 トリガーの数だけこの処理を繰り返す。

5. カーネルバッファに入っているビットストリームを読み出し、ABCN/BCC の識別番号 (address) やヒットの情報などを取り出す。カーネルバッファに入っている全データを読み出すまでこの処理が行われる。

6. PC と SEABAS 間の接続を切断する。

3.4.1 decode

BCC からのビットストリームは最初に BCC の address、次に 4 個のゼロ、そして二つの ABCN column からのビットストリームをマルチプレクスしたものが続く。ABCN/BCC のビットストリームには、BCC や ABCN の address や受け取った L1 トリガーの数、ABCN のヒットストリップの address、ヒットパターンなどの情報がバイナリで決まった順序でならんでおり、これを順序に従って抽出していくことを decode と呼ぶ。BCC からのビットストリームには先頭に 6bit 長の BCC の address があるが、この値は BCC によって変わるため直接探し出すことはできない。代わりに ABCN のデータフォーマットの開始を表す preamble (11101) を探すことで ABCN/BCC から

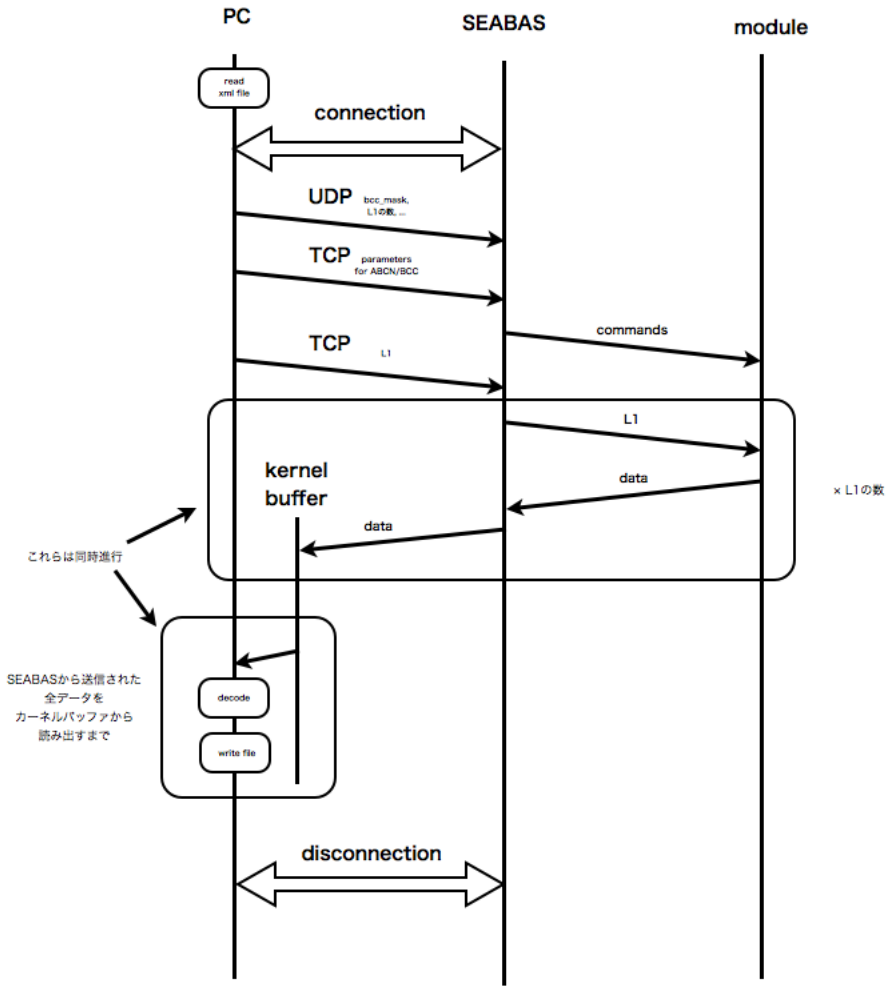

図 3.5: DAQ フローチャート

の出力の先頭を探す。具体的にはまず図 3.6 のように、BCC からの受信ビットストリームを 1 ビットずつ交互にデータ 1 とデータ 2 として並び替える。次にデータ 1 を 1 ビットずつ確認していき、preamble (11101) を見つけ出したら DATA1 の preamble の最下位ビットから数えて 13bit さかのぼって BCC address を取り出す (図 3.7)。

次に、図 3.8 に示すように、ABCN の preamble の後ろから ABCN が受け取った L1 トリガーの回数とバンチ衝突回数を取り出す。さらに、それらの後ろにあるヒット情報を以下の手順で識別する。

第 2 章で説明したように、ABCN のヒットデータ情報は、Isolated と Non Isolated の 2 種類がある。ABCN の address、ヒットのあったストリップの address、ヒットパターンを抽出した後に、ヒットデータ情報のデータの始まりを示す header (01) が続けば Isolated で、1X と続く場合は Non Isolated である。その様子を示したのが図

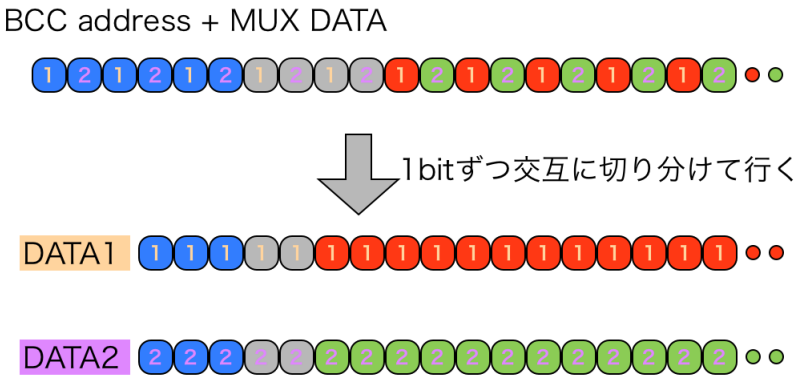

図 3.6: decode：切り分け

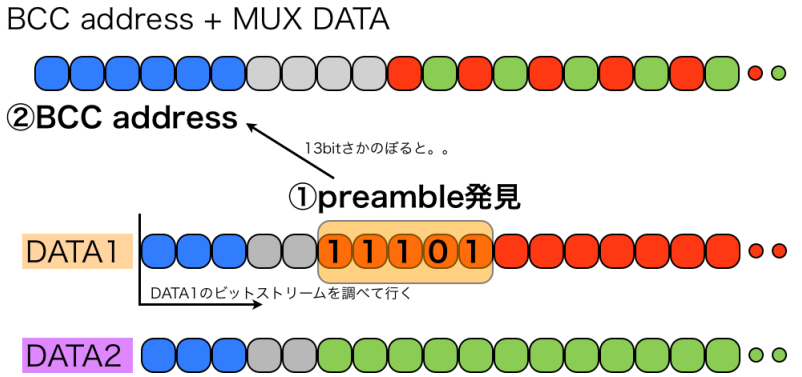

図 3.7: decode：preamble 探し

3.9 である。Non Isolated の場合はセパレータ (1)、隣のストリップのヒットパターン (XXX)、という並びがヒットのあるだけ続くのでそれらの decode を次のヘッダが見つかるまで行う。

3.5 DAQ 速度向上の試み

複数個のモジュールの同時読み出しでは、大量のデータを取得するため DAQ に時間がかかる。そこで、DAQ システムの読み出し速度を向上させるためにソフトウェアに改良を加えた。ソースコード記述内容を元にデータ送受信などの個々の処理にかかる時間を測定し、特に時間がかかる部分の改善を試みた。この章では試みた速度向上に関する結果について述べる。

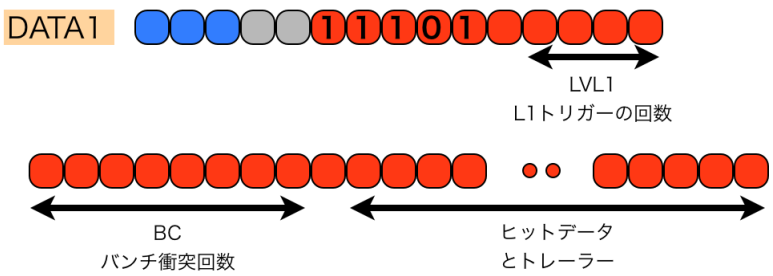

図 3.8: decode：L1 回数とバンチ衝突回数の抽出

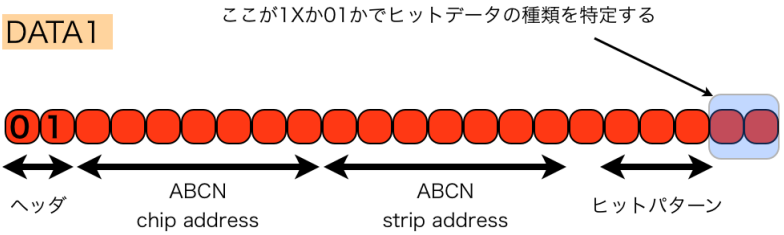

図 3.9: decode：ヒットデータ抽出

3.5.1 ソフトウェアの構造

ソフトウェアの処理を以下の六つに分類し 、それぞれにかかる時間を測定した。

オーバーヘッド

ABCN/BCC のレジスタ設定用のパラメータを記述した xml ファイルの読み込み部分、ユーザーが指定したプログラム用の引数の読み込み、SEABAS との TCP 通信のためのソケット確立、xml ファイルから読み込んだ ABCN/BCC のレジスタ用パラメータの SEABAS への送信、などの部分をオーバーヘッド部分とした。

L1 トリガー

L1 トリガー command のビットストリームを SEABAS へ送信する。

データ受信

ソケットを生成すると PC のシステム資源を用いてソケット用のカーネルバッファが生成される。SEABAS から送られて来たデータは、そのカーネルバッファに随時

蓄積される。ソケットプログラムでデータを受信するということは、PC のカーネルバッファからそこに入っているデータを読み出すことである。ここでデータ受信と呼ぶのは select 関数でソケットのカーネルバッファにデータが来るのを待ち、データが来たらそれを receive 関数で読み出す、という二つの処理である。

BCC DATA の Trailer カウント

ABCN はデータの最後に Trailer (1 0000 0000 0000 0000) (節 2.1.10 を参照) を送るため、BCC 経由では、ABCN 二つ分の Trailer がマルチプレクスされたビットストリーム (11 00000000 00000000 00000000 00000000) が BCC DATA の Trailer となる。この BCC DATA の Trailer を探し出す処理を Trailer カウントと呼ぶ。

decode

受信したデータの decode を行う。

ファイル出力

decode したデータを外部ファイルへ出力する。ヒットのあったストリップ毎にヒット情報を書き出す。ファイルフォーマットは root ファイルとバイナリファイルの二つを検証した。

3.5.2 DAQ 速度測定

時間測定には、ABCN が 20 個搭載された hybrid1 台と BCC1 個を用いた。1 台の hybrid を使って hybrid 複数個からの同時読み出しにかかる時間を測定するために、一つの BCC からのデータを USER FPGA 上に用意した 32 個全ての FIFO に書き込んだ。時間測定には C の標準関数である getimeofday 関数を用いた。

表 3.1: 1000 事象のデータ収集の各部分にかかる時間

部分	全体	オーバーヘッド	L1 トリガー	データ受信	decode	トレーラーカウント
時間 (sec)	418	5.1	5.1×10^{-3}	305	42.4	65

表 3.1 と表 3.2 に、1000 イベントのデータ収集を 10 回行った時の平均値をまとめた。ファイル出力については、バイナリ形式と root 形式の両方で書き出し、かかる時間

表 3.2: 1000 事象のファイル出力にかかる時間

ファイル形式	バイナリ	root
時間 (sec)	160	279

を比較した。二つの表からわかるように、特に遅かったのは TCP/IP 通信によるデータ受信部分とファイル書き出し部分であった。受信したデータサイズが 340Mbit だったため、表 3.1 のデータ受信部分の時間から、データ受信部分の処理速度は 1.1Mbps だったことがわかる。また、表 3.2 から、データファイルはフォーマットがバイナリの場合の方が速いためバイナリを採用することにする。

3.5.3 データ受信部分

簡易テスト

ここでは最も時間のかかっているデータ受信部分を詳細に検証する。正確にデータ受信の速度を測るためには、ファームウェアのデータ送信のタイミングを把握しなくてはならない。しかし、それは困難なので、クロック毎に 8bit のデータを SEABAS から PC へ TCP 通信で送り続けるファームウェアと、そのデータを receive 関数を用いて受信するだけのソフトウェアを準備した。ファームウェアでのデータ送信タイミングは常に一定なため、PC でデータを受け取る速度を測るだけで TCP 通信の速度を測ることが出来る。

この手法を用いたところ、7.54 秒で 20Mbit のデータを読み出せたため、データ収集速度は 2.65Mbps であった。receive 関数の中ではデータを受け取る変数が char 型なので、カーネルバッファから 8bit のデータを何度も読み出さなければならない。receive 関数にはオーバーヘッドがあるため、receive 関数が呼び出される回数に比例してデータ読み出しに時間がかかってしまう。そこで、より大きいサイズの変数を用意して一度にたくさんのデータを受け取るように変更した。

一度により多くのデータを読み出すためには大きいサイズの C の変数を用意すれば良いが、ABCN からのデータのサイズは可変、つまりイベント毎にカーネルバッファに入るデータサイズが変わってくる。そのため、カーネルバッファのデータを受け取る前にカーネルバッファに入っているデータのサイズを知る必要がある。そこで以下の手順でデータを読み出すことにした。

1. select 関数でソケットにデータが来るのを待つ。

2. データが来たら、システムコール関数の ioctl 関数を用いてソケット生成時に用意されたカーネルバッファにアクセスして、カーネルバッファに入っている

データサイズを調べる。

3. データの大きさに等しいCのchar型の配列を生成し、カーネルバッファに入っているデータを一度に読み出す。

上記の変更を加えたところ、データ収集速度は90Mbpsになった。SEABASのTCP/IP通信でのデータ転送速度は仕様上では100Mbpsなので、これを上手く利用できている。ここで、select関数、ioctl関数、receive関数を個別にかかる時間を測定したところ、稀にselect関数で時間がかかることがわかった。20,000回データ読み出しを行った時にselect関数でかかる時間を測定したところ、平均値が4.7×10^{-4}秒であった一方、0.1秒以上かかる事象は全体の0.25%であった。稀な事象のため、その原因の特定には至っていない。

3.5.4 改良後の結果

前項で説明した方法をシステムに組み込んで改めてテストをした。

改良前後での比較

改良後のシステムのデータ受信速度は、340Mbitのデータの受信に9.57秒かかったので36Mbpsであった。向上前の1.1Mbpsの33倍の速度に向上している。

読み出しBCC数依存性

3.5.2節で説明した方法で、仮想的に読み出すBCCの数を変えてデータ収集速度のデータ量依存性を測定した。オーバーヘッド部分にかかる時間はシステム全体の初期化なので読み出しデータ量に依存しないはずである。他の部分はデータ量に比例して処理時間が増えるはずである。

図3.10にデータ収集時間を仮想的に読み出しするBCCの数の関数として示した。オーバーヘッドではデータ量に依存しないという予想を図3.10の左上で確認できた。図3.10の他の部分はBCCの数に比例して線形に時間が増大している。

関数のみの処理速度

カーネルバッファからデータを読み出すための三つの関数、select関数、ioctl関数、receive関数にかかる処理速度を測定した。ここで、select関数は、カーネルバッファにデータが入るのを待ち、カーネルバッファにデータが入ったら、そのことを示す返り値を出力するという処理を行う関数である。ソフトウェアでは、select関数のデータ待ちを回避するため、L1トリガーを送った後に3ms待ってからデータ受信を開始

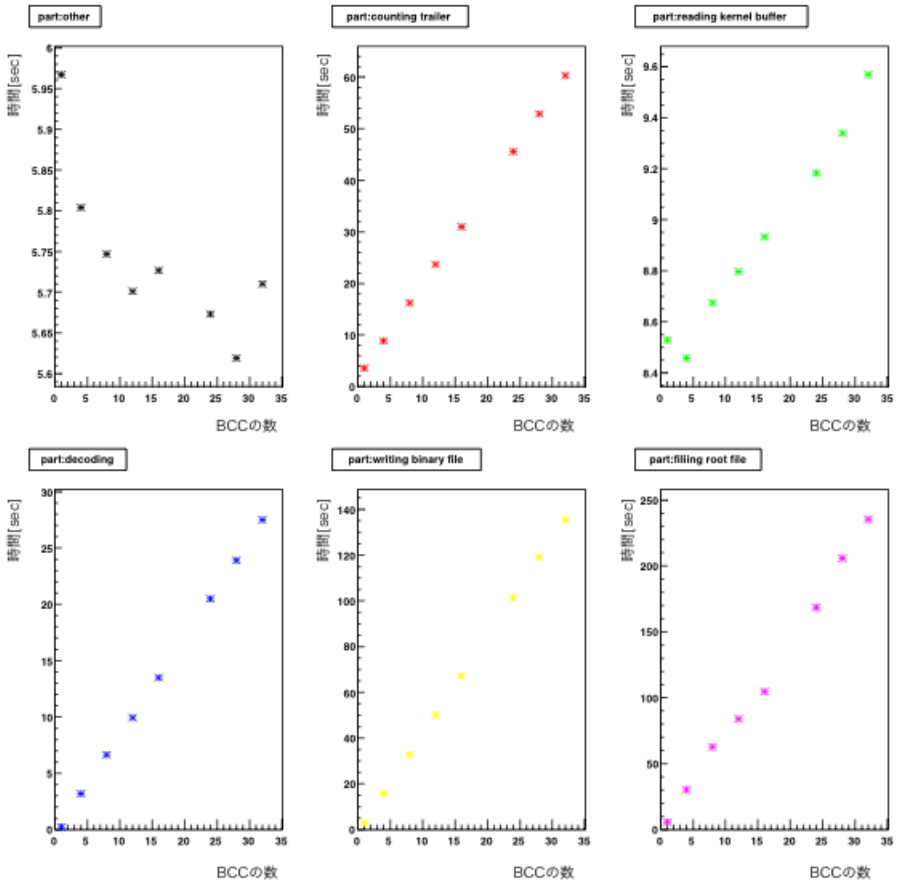

図 3.10: 1000 イベントの処理時間のデータ量依存性。左上がオーバーヘッド、真ん中上が Trailer のカウント、右上がデータ受信、左下が decode、真ん中下がバイナリファイル出力、右下が roor ファイル出力についてそれぞれ測定した時間を BCC の関数としてプロットした図。

するが、SEABAS が送信するデータ量によっては 3ms を超えても SEABAS がデータ転送を続けることがある。そのため、select 関数の処理にかかる時間は、SEABAS のデータ転送を待っている時間が含まれている可能性があるが、それでも三つの関数の処理速度 (=データ量/三つの関数の処理時間の合計) は SEABAS のデータ転送速度の 100Mbps (設計値) よりも速い 642Mbps であった。

3.5.5 まとめと考察

DAQ システムの速度向上のための検証を行い、特にソケットプログラムのデータ受信部分について大幅な速度向上に成功した。現状のバージョンのソフトウェアでは受信したデータの decode を行っているが、これはモジュールからの信号読み出し

においては必須の処理ではない。受信したデータを解析する際に decode も行うようにすれば、信号を読み出す時の decode を省くことができ、更なる速度向上も可能である。

また、データ受信部分の select 関数、ioctl 関数、receive 関数のみの処理速度は SEABAS のデータ転送速度を上回っている。SEABAS よりもデータ転送速度の速い読み出しボードを用いれば、さらにデータ受信部分の処理速度が向上する可能性がある。現在 SEABAS の後継である SEABAS2 が開発、試験されている。通信速度が 1Gbps になり、USER FPGA もより大容量のものに換装された一方で、USER FPGA から SiTCP FPGA を介して PC と通信するなどの基本的な機能は SEABAS と同様である。開発した DAQ システムは SEABAS2 の環境へ簡単に移植できるため、SEABAS2 を使用すれば更なる速度向上が期待できる。

第4章　DAQ動作確認

この章では、開発したシステムの動作試験について記す。まずは、BCC 経由で ABCN から信号を読み出せることを以下の手順で示す。

- BCC 経由での ABCN へのビットストリーム送受信
- ABCN のレジスタへのパラメータ書き込み
- 上記 2 項目に加えデータ収集や decode も含めた総合動作確認
- 既存システムとの比較

その後、複数個のモジュールからの信号を正しく読み出せることを確認する。

4.1　BCC 経由での ABCN 読み出し

4.1.1　送信ビットストリーム

SEABAS の USER FPGA から BCC へは BCC com と ABCN com を連続して送信する。BCC はそのビットストリームの中から ABCN com だけを ABCN へ送信する。図 4.1 は USER FPGA から BCC へ送ったビットストリーム (青色) と BCC が ABCN

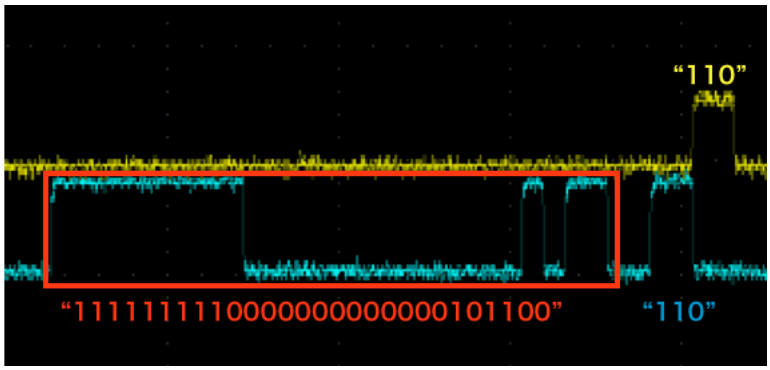

図 4.1: 青色が USER FPGA から BCC への送信ビットストリーム 、黄色が BCC から ABCN への送信ビットストリーム

へ送信したビットストリーム (黄色) をオシロスコープで見た図である。まず、青色で表示されたBCCへの入力ビットストリームは、PCから送信したBCC com + ABCN comのビットストリームと一致していることを確認した。また、赤枠で囲っている部分がBCC com (表2.6を参照) であり、このBCC comを取り除いた青色部分が黄色で示したABCNへの入力と等しいことがわかる。よって、BCC経由でABCNへビットストリームを正しく送信できていることがわかった。

4.1.2 ABCNからの出力ビットストリーム

二つのABCN columnは0と1と呼ぶことにする。ここでは、データはcolumn 0からのみ読み出す。BCCのアドレスはゼロとした。ABCNからは40MHzでデータが出力される。column 1からの出力は常に0なので、BCCはcolumn 0のデータを1ビットおきに出力するはずである。また、BCCの出力はABCNの出力の2倍の速度であるので、ここでは80MHzの出力が期待される。これらの挙動を確認するために、ABCNからBCCへの入力ビットパターンとBCCからの出力ビットパターンをオシロスコープで観察し比較した。その様子を図4.2に示す。紫色がABCNのcolumn0からBCCへの入力、黄色がcolumn 1からBCCへの入力、緑色がBCCからの出力である。ABCN column 0のデータの最初の7ビットは1110101で、column 1は常にゼロなので、BCCからの出力は10101000100010が期待される。緑色のBCCの出力は確かに10101000100010になっていることがわかる。また、BCCからの出力がABCN columnの2倍のレートになっていることも図からわかる。よって、ABCNからの出力ビットストリームは、BCCを介して正しく出力されていることを確認できた。

4.1.3 レジスタ設定

前節でUSER FPGAからBCCへBCC com + ABCN comを送信すると、BCCはABCN comをABCNへ送信することを確認した。ここでは、ABCNが自身のレジスタに保持しているパラメータを出力できることを利用して、ABCNのレジスタに送ったビットストリームがレジスタ内の値と一致しているかを確認する。

図4.3は、オシロスコープで見たUSER FPGAからBCC経由でABCNへ送信したビットストリーム (黄色で示した部分) と、ABCNがBCC経由で出力したUSER FPGAへの入力 (青色で示した部分) を示す。送信したABCNのレジスタ設定のためのビットストリーム部分を緑枠で、ABCNへ送ったL1トリガーのビットストリームを赤枠で、BCC経由で出力されたABCNの出力ビットストリームを青枠で示した。緑枠部分はBCC comとABCN comが連続している。ABCN com部分はヘッダ部分を除くとABCNのレジスタ設定のパラメータがバイナリで記述されており、その部

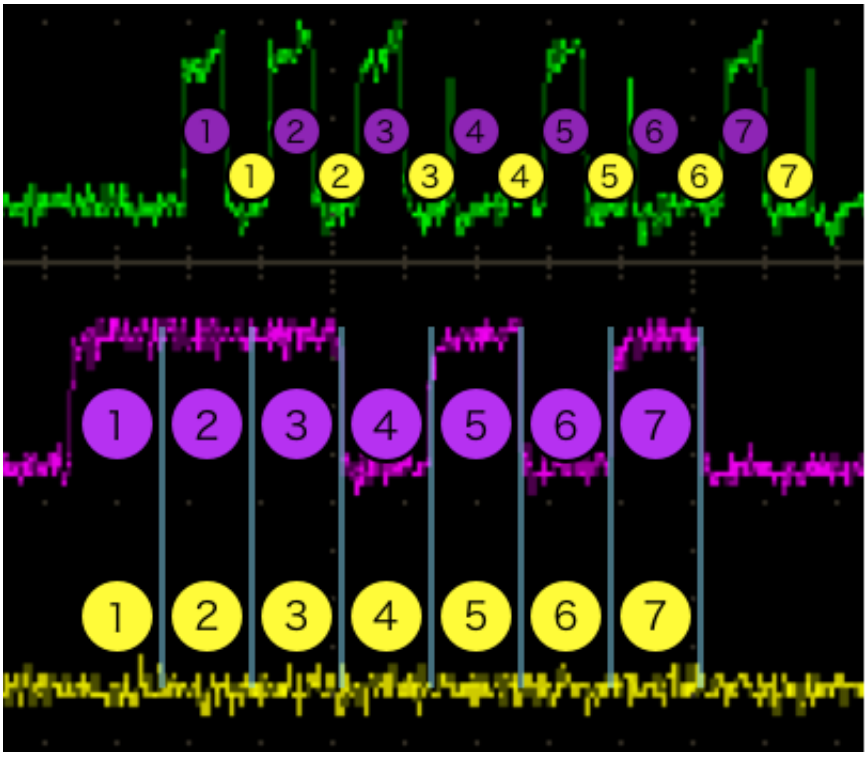

図 4.2: 緑色が BCC から USER FPGA への出力ビットストリーム、紫色が ABCN column 0 から BCC への出力ビットストリーム、黄色が ABCN column 1 から BCC への出力ビットストリーム

分のビットストリームは 0011000001000001 となっている。これは PC から送信したビットストリームと一致する。次に、青枠で示された ABCN からの出力中のレジスタのパラメータが記述されている部分を見ると、0011000001000001 となっている。これは PC から送信したビットパターンと一致している。よって、ABCN のレジスタの設定が正しく出来ていることがわかる。

4.1.4 総合動作確認

前節までで PC から SEABAS、BCC を経由して ABCN へ正しくビットストリームが送信できることと、ABCN のレジスタへの書き込みが正しくできることを確認した。この節では、BCC からのビットストリームを USER FPGA が読み出し、SiTCP を介して PC へ送信し、PC で読み出したビットストリームを deocde するという動作を含めて DAQ システム全体が正しく機能しているか試験した結果を記す。

まず、BCC 経由で ABCN1 個を使い、2.0fC の Calibration Pulse を入力したときのヒット数の変化を、Comparator の閾値を変えながら観察した。Front End のアン

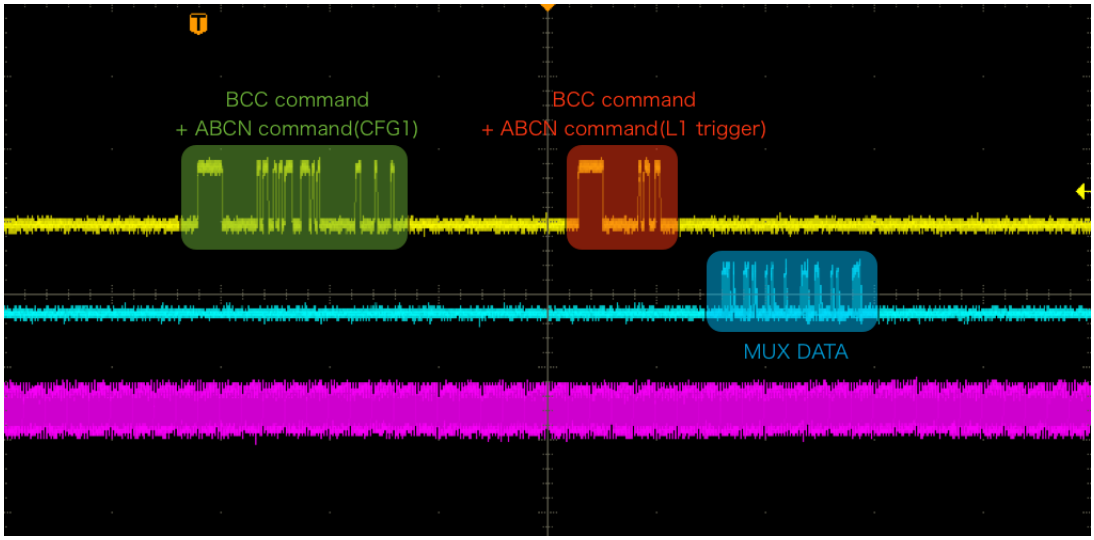

図 4.3: 黄色が USER FPGA から BCC へのビットストリーム、青色が BCC から USER FPGA へのビットストリーム

プのゲインは 100mV/fC (設計値) なので出力電圧は 200mV になる。閾値を 160mV、192mV、208mV と変えて、それぞれ L1 トリガーを 100 回送った時の各ストリップのヒット数を図 4.4 に示す。閾値が高くなるにつれヒット数が減少して、閾値が 200mV を超えるとほとんどヒットがなくなることがわかる。開発したシステムで ABCN の閾値を正しく変えることが出来ていることを示している。また、閾値が十分低い 160mV の時のヒット数が、送った L1 トリガーの数と等しいことから、L1 トリガーを指定した回数だけ過不足なく送信、かつ ABCN からのデータを全て受信できていることがわかる。これらの挙動を示す図 4.4 は、取得したデータを decode して解析することで得られた図であることから、USER FPGA が BCC からビットストリームを読み出し SiTCP FPGA を介して PC へ送信し、PC 上で decode するという一連の機能が正しく動作していることも示している。

次に、一つの ABCN column から読み出す ABCN の数を変えて試験した。図 4.5 は、読み出す ABCN の個数を ABCN column の先頭から数えて 1 個、6 個、10 個と変えたときの L1 トリガー 100 回に対する全 ABCN の各ストリップのヒット数を示したものである。指定した ABCN からのみデータを読み出していることがわかる。

4.1.5 既存システムとの比較

BCC を経由せず ABCN のみ読み出すシステムは既に存在する。開発したシステムで BCC 経由で ABCN を読み出した結果を、既存の DAQ システムで読み出した結果と比較する。なお、既存 DAQ での測定結果は本研究とは別の研究にて行われたもの

を参照する [8]。

表 4.1: ゲインとノイズ

システム	既存 DAQ		開発した DAQ	
column	0	1	0	1
Gain [mV/fC]	99.28	99.47	97.43	96.95
RMS (Gain)	3.215	3.083	1.062	1.512
Noise [electrons]	397.8	393.9	382	387.1
RMS (Noise)	28.96	33.33	30.73	32.41

二つの ABCN column を使い ABCN20 個のゲインとノイズを測定した結果を表 4.1 にまとめた。Gain [mV/fC] と Noise [electrons] は各 column (=1280 本のストリップ) の平均値で、RMS は標準偏差を表す。ABCN の設計値ではゲインは 100 mV/fC で、6fC までの入力に対して 3%の線形性なので、測定値は設計値をほぼ再現していることがわかる。図 4.6 は各ストリップ毎の測定値の差 (=開発 DAQ 測定-既存 DAQ 測定) を示すもので、図 4.7 は各 column 毎の差分の度数分布を示す。図 4.6 からは、column0 のストリップ 0 番から 1023 番までの ABCN では既存システムの測定と今回の測定が非常に良い一致を示していることがわかる。図 4.7 からノイズの違いは 50electrons 程度におさまっていることがわかる。Minimum Ionizing Particle が SCT のシリコンセンサーを通過した時に作る信号の大きさが 22,000electrons であることや、2.4cm のストリップセンサーを付けたときの ABCN のノイズが 750electrons 程度であることから、50electrons は十分に小さなものであり、今回開発したシステムに理解不能なノイズは乗っていないと言える。

4.2 複数モジュールの同時読み出し

2 個のモジュールを用意し、それぞれ片面のみ (すなわち hybrid4 個) を読み出した。まずは hybrid を 1 個ずつ読み出し、ゲインとノイズを測定した。次に 2 個のモジュールからの信号を同時に読み出し、ゲインとノイズを測定、一つずつ読み出した場合の結果と比較した。モジュールはぞれぞれ M2、M4 と呼び、M2 の hybrid をそれぞれ h0、h1、M4 の hybrid をそれぞれ h2、h3 と呼ぶ。

表 4.2 は hybrid を 1 個ずつ読み出した時の各 column 毎のゲインとノイズの平均値と RMS を、表 4.2 は hybrid4 個同時に読み出した時の各 column 毎のゲインとノイズの平均値と RMS をまとめたものである。表 4.2 と表 4.3 を比較するとゲイン、ノイズ共にほぼ一致していることがわかる。図 4.8 は、hybrid1 個ずつ測定した時と hybrid4 個同時に測定した時の、ストリップ毎のゲインの差分の度数分布で、図 4.9

表 4.2: hybrid1 個ずつ読み出し

module	M2				M4			
hybrid	h0		h1		h2		h3	
line	0	1	0	1	0	1	0	1
Gain [mV/fC]	114.1	113.7	113.0	113.5	115.2	114.8	114.7	114.2
RMS (Gain)	1.630	1.402	1.884	1.541	1.596	1.441	1.363	1.398
Noise [electrons]	563.8	569.0	650.4	584.6	574.7	576.3	561.4	558.5
RMS (Noise)	35.79	38.66	47.95	39.38	36.55	38.09	35.65	34.54

表 4.3: hybrid4 個同時読み出し

module	M2				M4			
hybrid	h0		h1		h2		h3	
column	0	1	0	1	0	1	0	1
Gain [mV/fC]	114.2	113.8	114.0	114.6	115.0	115.0	114.9	114.1
RMS (Gain)	1.771	1.494	1.990	1.517	1.529	1.459	1.472	1.361
Noise [electrons]	565.6	568.2	637.7	572.1	564.5	568.6	559.7	552.1
RMS (Noise)	35.57	36.70	45.01	37.65	37.61	38.48	36.80	37.38

はノイズの差分の度数分布を示す。図 4.8 を見ると、6fC までの電荷の入力信号に対して 3%の線形性、10fC までの電荷の入力信号に対して 10%の線形性という ABCN のスペックを考慮すると、複数読み出しの結果は一つずつ読み出した時の結果と良く一致していることがわかる。また、図 4.9 を見ると RMS は 50electrons なので、複数個のモジュールから同時に信号を読み出しても 1 個ずつ読み出したときに比べてノイズが増えていないことがわかる。よって、開発したシステムで複数個のモジュールの同時読み出しを正しく行えたと結論付ける。

4.3 まとめと考察

開発したシステムに関して以下のことを確認した。

- BCC 経由で ABCN 読み出しを行ったところ、ABCN/BCC のレジスタ設定に基づく期待通りの挙動を確認した。
- 開発した DAQ システムで測定した ABCN のゲインとノイズは既存 DAQ での測定結果と良く一致することを確認した。
- 複数個のモジュールを同時に読み出した時の結果が、一つずつモジュールを読み出した時の結果と良く一致した。

よって、開発した読み出しシステムは、BCC 経由での ABCN 読み出しと複数個の SCT モジュールの同時読み出しを正しく行えている。

今回、2 個のモジュールの片面だけから同時読み出しに成功したが、モジュール 16 個の片面からの読み出しができるようにシステムを開発した。第 3 章で DAQ の速度向上について説明した時、BCC1 個分のビットストリームを他の BCC 用の FIFO に入力することで仮想的に 32 個の BCC からの信号を読み出せることを確認した。よって、この章では 32 個の FIFO のうち 4 個を用いた読み出ししか確認していないが、他の 28 個の FIFO でも同様に同時に読み出せるはずである。今後の課題として、実際に 12 個のモジュールを用意して、それら片面分の 24 個の BCC からの信号を同時に読み出せることを確認しなければならない。

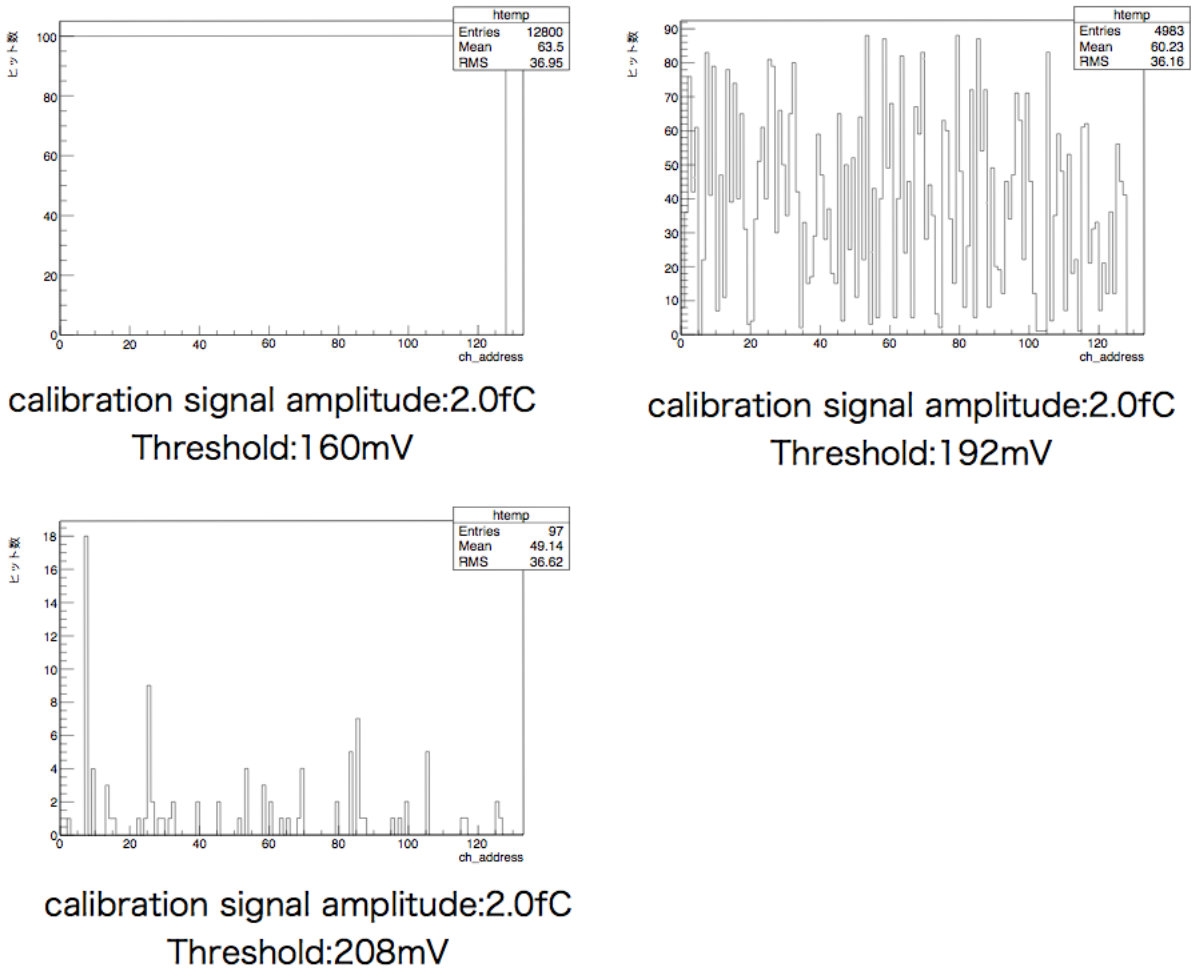

図 4.4: 各ストリップのヒット数。左上が閾値 160mV、右上が閾値 192mV、左下が閾値 208mV に設定したときの、100 回 L1 トリガーを送ったときの各ストリップにおけるヒット数を度数分布にした図。

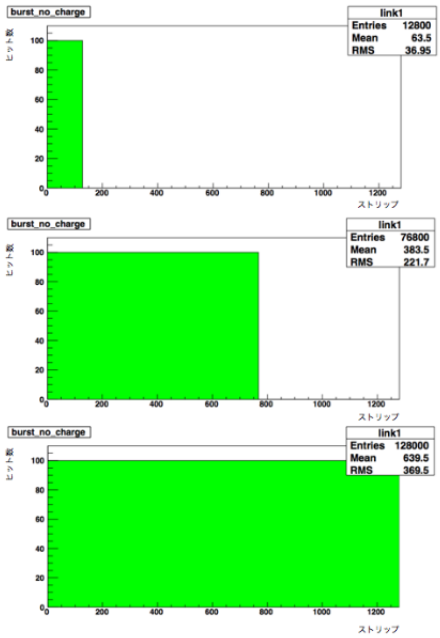

図 4.5: ABCN column 読み出し。上が column の 1 個だけ、中央が column の最初から 6 個目まで、下が column 全部の ABCN を読み出したときの図。

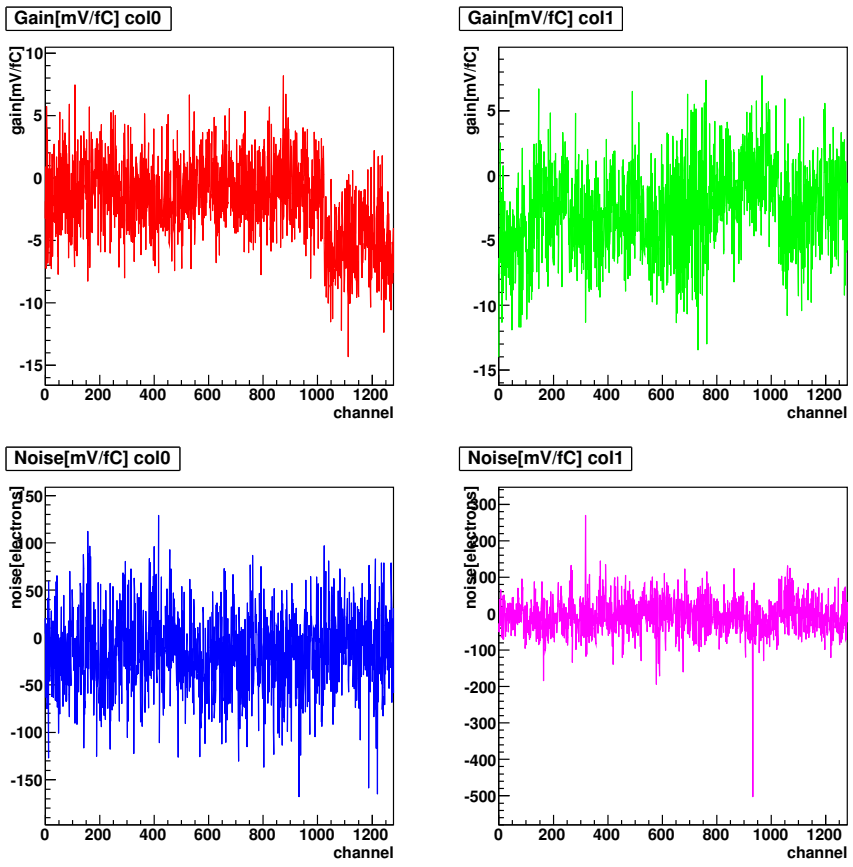

図 4.6: 二つの column の各ストリップにおけるゲインとノイズの差分 (=開発 DAQ の結果-既存 DAQ の結果) を横軸をストリップ番号にとり、縦軸をそれぞれゲイン [mV]、ノイズ [electrons] にとってプロットしたもの。左上が column0 のゲインの差分を、右上が column1 のゲインの差分を、左下が column0 のノイズの差分を、右下が column1 のノイズの差分を示す。

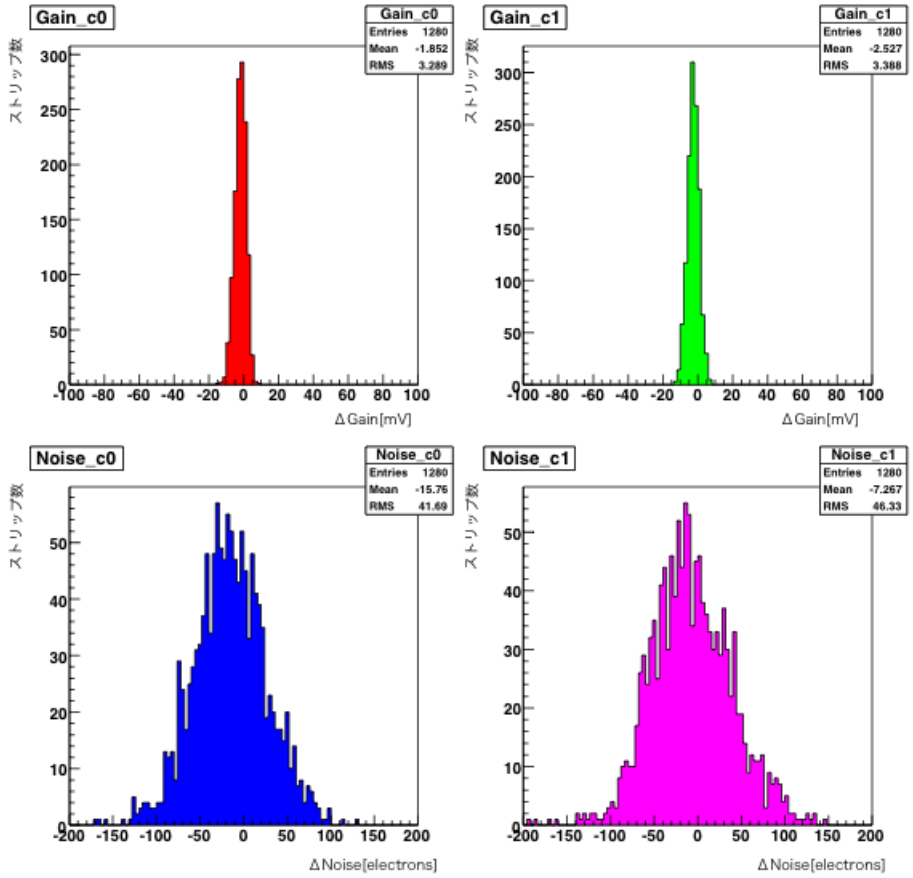

図 4.7: 二つの column の各ストリップにおけるゲインとノイズの差分 (=開発 DAQ の結果-既存 DAQ の結果) の度数分布。左上が column0 のゲインの差分を、右上が column1 のゲインの差分を、左下が column0 のノイズの差分を、右下が column1 のノイズの差分を示す。

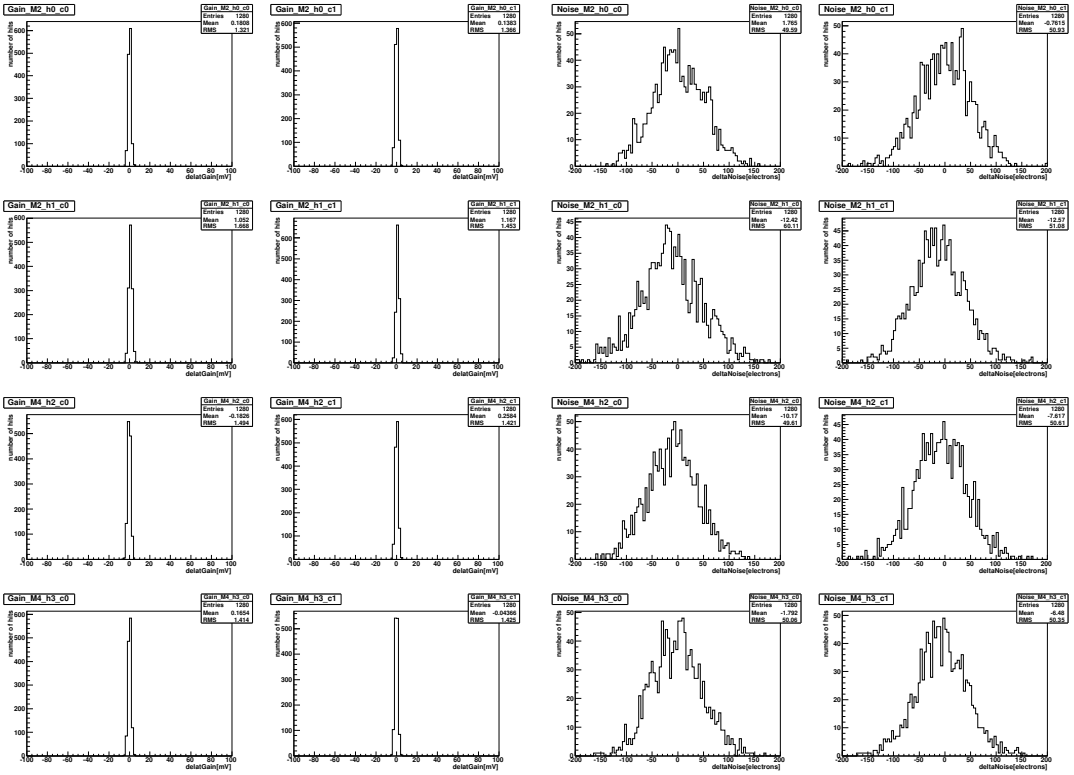

図 4.8: 8 個の column の各ストリップにおけるゲインの差分 (=同時読み出しの結果 - 一つずつ読み出しの結果) の度数分布。各度数分布のタイトルの M〜がモジュールの番号を、h〜が hybrid の番号を、c〜が column の番号をそれぞれ表す。例えば、一番上の左の図はモジュール M2 の hybrid0 番の column0 のゲインの差分の度数分布を示す。

図 4.9: 8 個の column の各ストリップにおけるノイズの差分 (=同時読み出しの結果 - 一つずつ読み出しの結果) の度数分布。各度数分布のタイトルの M〜がモジュールの番号を、h〜が hybrid の番号を、c〜が column の番号をそれぞれ表す。例えば、一番上の左の図はモジュール M2 の hybrid0 番の column0 のノイズの差分の度数分布を示す。

第5章　結論

BCC 経由で ABCN を読み出しと複数個の新型 SCT モジュールの読み出しができるシステムを開発した。ABCN/BCC から設定通りの挙動が得られたこと、開発したシステムの測定結果が既存システムの測定結果と良い一致を示したこと、モジュールを複数個同時に読み出した結果がモジュールを単独で読み出した結果と良い一致を示したことから開発したシステムが正しく動作していると結論付ける。

開発したシステムのソフトウェアの処理速度向上を試みた。TCP 通信でデータを受信する処理部分の処理速度をある程度向上することができた。

謝辞

本研究を行うに当たって、まず、素晴らしい研究環境を与えて下さった山中卓教授に深く感謝致します。物理に限らず、様々なテーマに対して、考察し答えを追求する姿勢からは多くのことを学びました。本当にありがとうございました。

指導教官である花垣和則准教授には、ATLAS 実験に関わる機会を下さっただけでなく、研究室に配属された当初から今日にいたるまで、要領を得ない私に熱心な御指導と御助力を下さいましたことを深く感謝致します。また、折に触れて、物理だけでなく、研究生活に関する助言やご本人の体験談など興味深い話をして下さいました。本当にありがとうございました。

KEK の田窪洋介さんは、お忙しい中、物理やソフトウェアなどの知識、研究者としてのあり方、海外での研究生活における様々な助言など、数えきれないほどの助言を下さいました。共同での研究で、熱心に御指導と御助力下さいましたこと深く感謝致します。本当にありがとうございました。

KEK の海野義信さん、池上陽一さん、安芳次さん、University of Geneva の Sergio Gonzalez Sevilla さんにはお忙しい中、本研究において様々な御助言や御助力を下さいましたこと深く感謝致します。ありがとうございました。

ATLAS グループの先輩である、廣瀬穣さん、岡村航さん、後輩の東野聡くん、Teoh JiaJian くん、辻嶺二くん、元阪大 ATLAS グループの目黒立真さん、内田潤さん、松本良雄くんには研究だけでなく様々な場面で本当にお世話になりました。廣瀬さんにはいろいろなことで相談に乗って頂き、特に海外生活に関しては廣瀬さんのおかげで無事に過ごすことが出来ました。岡村さんと内田さんには M1 の頃から、研究に関する御助言や御助力だけでなく、研究生活における有益な話をして頂きました。特に岡村さんには、研究室や KEK での共同研究でいろいろなことを教えて頂きました。本当にありがとうございます。東野くん、JiaJian くん、辻くんからはいつも良い刺激を受けています。ありがとうございます。

助教の外川学さん、特任研究員の李栄篤さん、Jason Sang Hun Lee さんには、研究に関する御助言や日常生活における興味深い話をして下さいました。秘書の亀井彰子さんには、事務に関する様々な面で御助力頂きました。研究生活に集中できたのも亀井さんのおかげです。本当にありがとうございました。

研究室の先輩の岩井瑛人さん、佐藤和史さん、Lee Jong-won さん、村山理恵さん、柳田陽子さん、中谷洋一さん、杉山泰之さん、伴野真太郎くん、元 KOTO グループの宇井利昌くんには研究や研究生活における様々な御助言や御助力を頂いたり、他

愛のない話につきあって頂いたりしました。特に岩井さんにはよく研究の相談に乗って頂きました。ありがとうございます。学部生の頃から付き合いがある浅川直也くんにはよく話相手になって頂きました。高島悠太くん、豊田高士くん、保坂荘太朗くんが熱心に実験装置を作成している姿には良い刺激を受けました。ありがとうございました。

最後に研究生活を支えてくれた家族に感謝します。

参考文献

[1] Project Specification Project Name: ABC-N ASIC Version: 1.3.1 (2008).

[2] D. Campbell, *et al.* "THE ATLAS BINARY CHIP", 2nd Workshop on Electronics for LHC Experiments, CERN/LHCC/96-39(1997).

[3] J. Kaplon, *et al.* "The ABCN front-end chip for ATLAS Inner Detector Upgrade", Topical Workshop on Electronics for Particle Physics, Naxos, Greece,15-19 Sep 2008, pp.116-120(Published Version from CERN 2009).

[4] Josuah Wicht, Hervè Fischer, *et al.*"Test of Buffer Control Chip by FPGA", (2009) Ècole d'lnènieurs et d'Architectes de Fribourg part of the University of Applied Sciences Western Switzerland Bachelor thesis.

[5] https://twiki.cern.ch/twiki/bin/view/Main/BccChip

[6] Tomohisa Uchida, Yasuo Arai, SEABAS (Soi EvAluation BoArd with Sitcp) User's Manual (2008).

[7] Tomohisa Uchida, Member, IEEE, "Hardware-Based TCP Processor for Gigabit Ethernet",(2008) IEEE Transactions on Nuclear Science. vol. 55, no. 3, 2008.6, pp. 1631-1637.

[8] 岡村航, 「ATLAS 実験アップグレード用シリコン検出器テストシステムの開発およびプロトタイプ検出器の性能評価」 (2011) 大阪大学 修士論文