2019 年度 修士論文

HL-LHC ATLAS 実験に向けた フロントエンド ASIC の読み出しシステムについての研究

2020 年 1 月 10 日

京都教育大学大学院 理科教育専修
基礎物理学研究室 修士 2 年

村田 大樹

指導教員
高嶋 隆一
谷口 和成

概要

ATLAS(A Toroidal LHC ApparatuS) 検出器とは，欧州原子核研究機構 (CERN) が所有している LHC(Large Hadron Collider) に設置されている検出器の一つである。ATLAS 実験の主な目的は，高エネルギー陽子・陽子衝突を用いて新粒子の発見や標準模型を超える物理の探索である。

LHC はより高いルミノシティを達成するために，2024 年からの Long shutdown を経て，2026 年頃に HL(High Luminosity)-LHC にアップグレードされる予定である。

ATLAS 検出器の内部飛跡検出器で実装されているシリコン検出器では，HL-LHC アップグレードに対応できるように高速データ転送が可能な新型フロントエンド ASIC を開発している。

京都教育大学では ASIC からのデータを読み出すシステムが構築されていなかったため，現在 ATLAS 検出器で用いられている ASIC のデータ読み出しが可能な DAQ システムを構築した。

現在の読み出しシステムでは 2 チップの ASIC 同時読み出しまでしか行われていなかったが，4 チップ ASIC のデモンストレータモジュールが開発される予定であることから 4 チップでの読み出しシステムの開発が求められている。4 チップ読み出しシステムの実現に向け，インターフェースカードを開発し，ファームウェアの修正を行った。そのシステムを用いて動作確認を行い，データを読み出せることを確認した。

新型 ASIC の読み出しに向けて，現在主に YARR と BDAQ53 の 2 種類のデータ読み出し用のソフトウェアが候補に挙がっている。これらを比較する際に ASIC の設定を同じにする必要があるが，設定を合わせるツールがなかった。そのため，ASIC の設定を記したコンフィギュレーションファイルを YARR から BDAQ53 へフォーマットを合わせて変換するコンバータスクリプトを製作した。

また，コンバータスクリプトを用いた BDAQ53 による読み出しに向けて，DAQ システムから ASIC へのケーブルを製作し，IBERT を用いたループバックテストを実行した。

目次

図目次

表目次

1　序論

1.1　LHC 実験

LHC は欧州原子核研究機構 (CERN) が所有する周長 27km の大型衝突型加速器でスイス・ジュネーブ郊外の地下 100m に設置されており，新粒子や，新物理の探索を行っている。LHC は陽子・陽子衝突点を設けており，その衝突点には ATLAS，ALICE，CMS，LHCb の 4 つの検出器が設置されている。図 1 に LHC の全体図を示す。

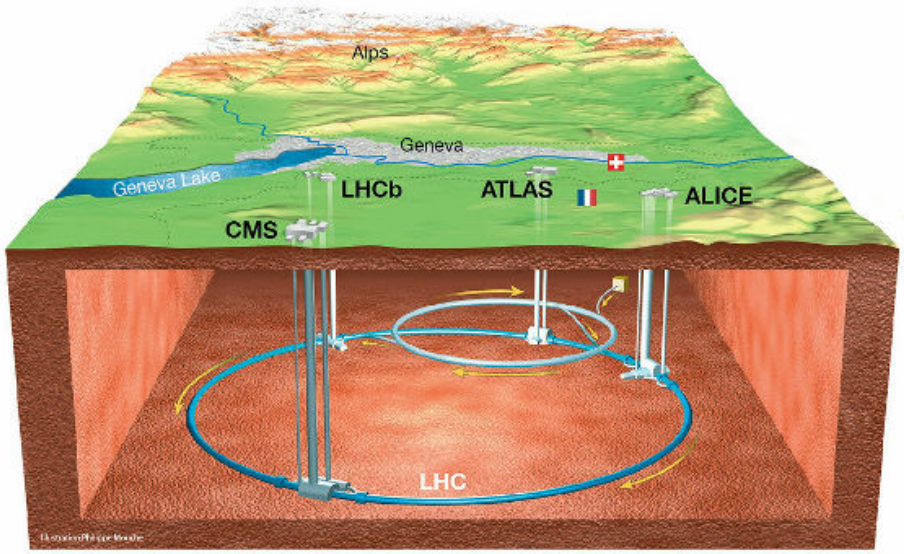


図 1: LHC の全体図

1.2　HL-LHC 計画

LHC は 2024 年からの Long shutdown を経て，2026 年頃に HL(High Luminosity)-LHC にアップグレードされ，ルミノシティが大幅に向上される予定である。これにより，取得できる事象数が増加することで，より詳細に新物理の探索が行える。瞬間ルミノシティは 2017 年に達成した $2.06 \times 10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ から $5\sim7 \times 10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ まで上昇する予定である。また，積分ルミノシティは 10 年間で $4000\mathrm{fb}^{-1}$ を目標としている。LHC と HL-LHC の比較を表 1 に示す。

1.3　ATLAS 検出器

ATLAS 検出器は LHC に設置されている検出器の一つで、新物理探索や新粒子の発見を目的とした検出器であり，ATLAS 検出器全体で，直径 25m，全長 44m になる。現行の ATLAS 検出器

	LHC	HL-LHC
重心系エネルギー	14TeV	14TeV
瞬間ルミノシティ	$2.06 \times 10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$	$5\sim7 \times 10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$
積分ルミノシティ	$300\mathrm{fb}^{-1}$	$4000\mathrm{fb}^{-1}$
バンチ当たりの衝突事象	60	140

表 1: LHC と HL-LHC の比較

内部では中央の陽子・陽子衝突点を覆うように内部飛跡検出器，電磁カロリーメータ，ハドロンカロリーメータ，ミューオン検出器が順に層を成して配置されている。ATLAS 検出器の全体図を図 2 に示す。

内部飛跡検出器 (Inner detector) は最内層から順に，格子状に配置された信号読み出しによる 2 次元での測定が可能なピクセル型検出器，1 次元での測定を行うストリップ型検出器の SCT(SemiConductor Tracker)，ストローチューブ型のガス検出器である TRT(Transition Radiateion Tracker) から構成される。

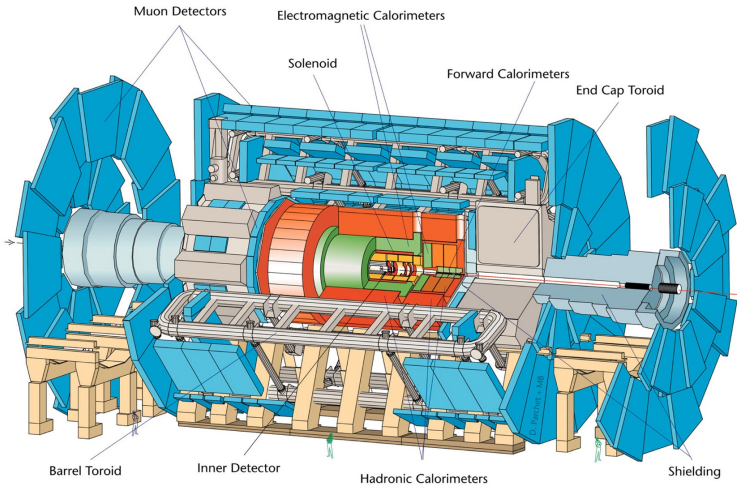


図 2: ATLAS 検出器

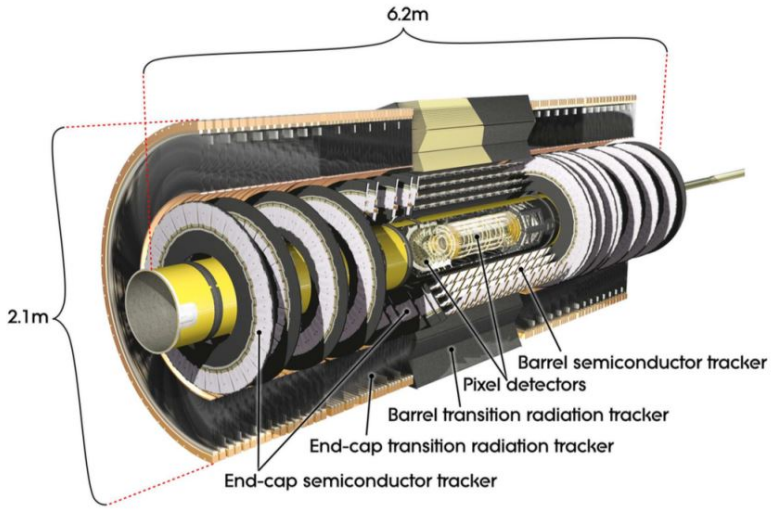


図 3: 内部飛跡検出器

内部飛跡検出器最内層のピクセル検出器では，円筒状のバレル部分に 4 層，ビームパイプから放射状に広がるディスク部分に 3 層配置されている。そのバレル部分最内層には Insertable B-Layer(IBL) が配置されている。図 4 にビームパイプに沿った方向からピクセル検出器を見た時の構造図を示す。

IBL は内部飛跡検出器のアップグレードのために，既存のピクセル検出器とは別にビームパイプごと独立で組み立てられ，後から挿入された検出器である。このような経緯から IBL で用いられている技術は既存のピクセル検出器と比べて，新しいものとなっている。

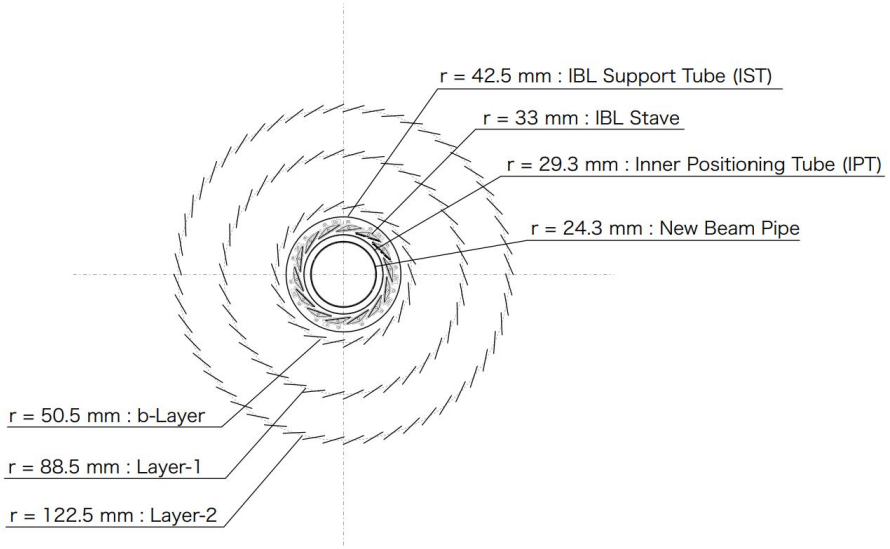


図 4: ピクセル検出器バレル部分の構造図。最内層の IBL Support Tube(IST) に囲まれた部分が IBL である。

HL-LHC アップグレードにより衝突事象数が増加することから放射線環境がさらに厳しくなる。これにより現行の検出器の耐えうる放射線損傷を超えてしまう。また，生成される粒子も増加するので検出器のヒット占有率が上がり，粒子の飛跡候補が増えることで飛跡再構築の精度が下がる。

これらの問題を解決するために，放射線耐性が高く，格子状に配置されているピクセルのサイズをより小さくした検出器が開発されており，2024 年に予定されている LHC のロングシャットダウンで，内部飛跡検出器のすべての検出器を新しく入れ替える予定である。図 5 にアップグレード後の内部飛跡検出器の構造を示す。青で示された部分がストリップ型検出器で，赤で示された部分がピクセル検出器であり，縦に配置されているのは End-cap である。また，ピクセル検出器の外側 3 層をアウターバレル，内側 2 層をインナーバレルと呼ぶ。

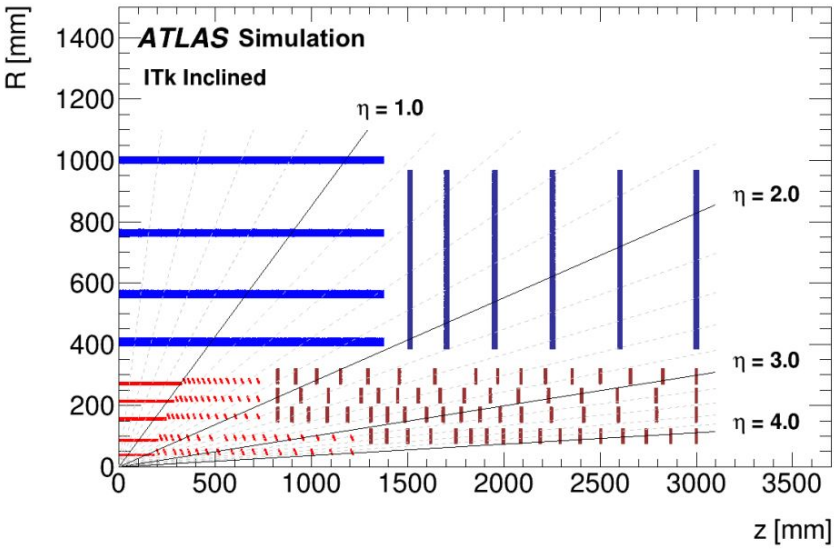
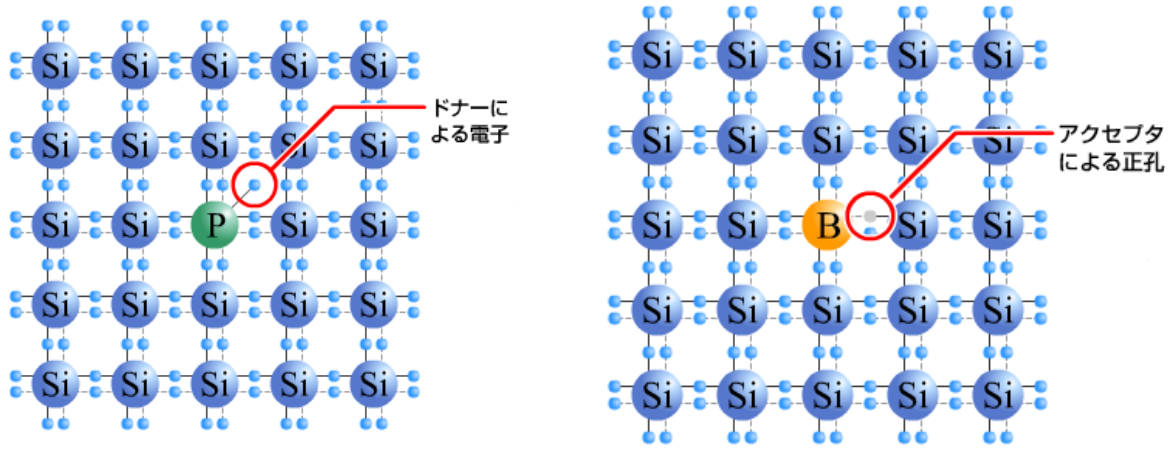


図 5: HL-LHC における内部飛跡検出器の模式図

1.4　シリコン検出器の原理

ATLAS 検出器内部のピクセル型検出器では，半導体検出器の一つであるシリコン検出器が用いられている。半導体とは導体と絶縁体の中間的な抵抗率を持つ物質のことであり，主にシリコンやゲルマニウムなどの 4 価の元素からなる。半導体には不純物を含まない純粋なシリコン結晶の真性半導体のほかに，5 価や 3 価の元素を不純物をして添加した n 型半導体や p 型半導体があり，n 型半導体に添加する不純物をドナー，p 型半導体に添加する不純物をアクセプターと呼ぶ。

n 型半導体の場合，5 価の元素をドナーとして添加することでドナーの持っていた電子のうちの一つが自由電子となる。この自由電子がキャリアとして移動することで電気伝導性が上昇する。また，p 型半導体の場合，3 価の元素をアクセプターとして添加することで，シリコンとアクセプターとの共有結合において電子が不足する状態となる。この電子が欠落したものを正孔（ホール）と呼び，正孔がキャリアとして移動することで電気伝導性が上昇する。



(a) n 型半導体（リンをドナーとして添加）　(b) p 型半導体（ホウ素をアクセプターとして添加）

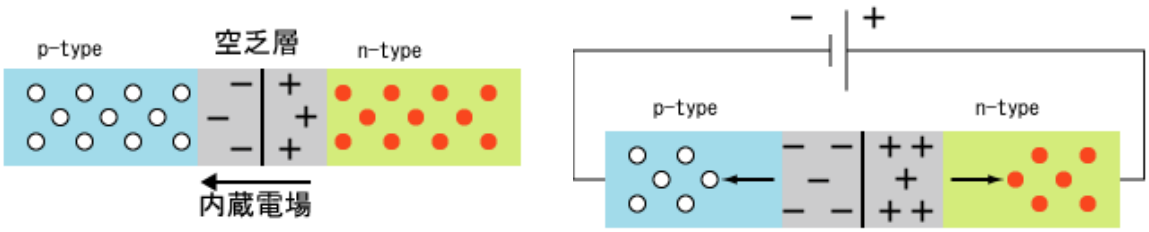
図 6: 不純物半導体の型と電子

n 型半導体と p 型半導体を接合（pn 接合）させると，接合面付近で自由電子と正孔が衝突し対消滅をする。これを再結合と呼び，再結合により自由電子と正孔が消滅することで n 型半導体は正に，p 型半導体は負に帯電する。これにより n 型半導体と p 型半導体の間に電場が発生する。また，再結合により自由電子も正孔も存在しない領域を空乏層と呼ぶ。

シリコン検出器に入射した荷電粒子はシリコンとの電離相互作用によるイオン化を引き起こし，電子正孔対を生成する。電子正孔対を生成するのに必要なエネルギーは既知であるため（1 個当たり 3.64eV），測定した電荷量から電子または正孔の数を求めることで荷電粒子のエネルギーを算出することができる。 だが，n 型半導体や p 型半導体内で荷電粒子が電子正孔対を生成しても自由電子やホールと結合してしまうため，検出することができない。一方で，空乏層内で電子正孔対を生成した場合，キャリアが存在しないので電子正孔対は消滅せず，pn 接合によって生じた電場により電子正孔対は空乏層内を移動するのだが，空乏化されていない領域に達すると自由電子またはホールと結合して消滅してしまう。そのため半導体領域をすべて空乏層とする全空乏化の必要がある。

n 型半導体に正，p 型半導体に負の電圧（逆バイアス）をかけることで n 型半導体に正孔，p 型半導体に電子を注入し，それぞれの半導体領域においてキャリアを減少させ，空乏層を広げることができる。これを用いて，半導体全体が空乏化するまで逆バイアスを高めることで全空乏化することができる，全空乏化したときの逆バイアスの値を全空乏過電圧と呼ぶ。

全空乏化することで半導体領域から荷電粒子が生成した電子正孔対を取り出すことができ，測定が可能になる。これがシリコン検出器の原理であり，位置分解能を高めるために縦横にシリコン検出器を並べたものをピクセル検出器と呼ぶ。



(a) pn 接合

(b) 逆バイアス電圧による空乏層拡大

図 7: pn 接合と逆バイアスによる空乏層の拡大

1.5 ASIC

ASIC（Application Specific Integrated Circuit：特定用途向き集積回路）とは特定用途向けに複数機能を実装した集積回路のことである。ASIC はピクセル検出器に直接接続され，検出器からの信号を最初に受け取り処理を行うフロントエンドエレクトロニクスとしての役割を果たしている。そのことからフロントエンド ASIC とも呼ばれることもあるが，この論文では ASIC と呼ぶ。

現行ピクセルセンサーには FE-I3 という ASIC が実装されており，Insertable B-Layer(IBL) には，性能を向上させた FE-I4 という ASIC が実装されている。FE-I4 はリビジョンが行われており，最終のバージョンとして FE-I4B が開発された。Insertable B-Layer の B をとって FE-I4B と呼ばれている場合もあるが，この論文では FE-I4B のことを FE-I4 と呼ぶ。また，アップグレード後の ATLAS に実装される予定の RD53A という新型 ASIC が現在開発されている。RD53A は，研究開発番号 53 の ASIC という意味があり，CMS，ALICE，LHC b を含むすべての検出器で実装される予定である。

FE-I4 は 50 × 250μm^2 の大きさのピクセル型読み出し回路が 80 × 336=26880 個配置されており，20 × 18.6mm^2 の一つのチップを成している。それに対して RD53A は 50 × 50μm^2 もしくは，25 × 100μm^2 のピクセルが，現段階では 400 × 192=76800 個配置され，20 × 11.8mm^2 のチップを構成しており，ピクセルの縮小，個数の増加がなされている。将来的にはカラム数が 192 から 384 になることが予定されている。そのほかにもデータ転送速度で Gbps[*1] 単位での読み出しを可能にするなど ATLAS アップグレードに対応できるような改良が施されている。

本論文では FE-I4 を用いた読み出しを行っていることから，FE-I4 の仕様を以下に記す。

1.5.1 電源規格

FE-I4 内ではリニアレギュレータという電源部品が電源装置からの入力電源を受け取り，制御回路を用いて出力電圧を監視することで，常に一定の電圧を内部回路へ出力することができる。これ

[*1] GigaBit Per Second の略で通信速度の単位を表す。

	FE-I3	FE-I4	RD53A
ピクセルサイズ	50 × 400μm^2	50 × 250μm^2	50 × 50μm^2
ピクセル数	18 × 160	80 × 336	400 × 192
チップサイズ	7.6 × 10.8mm^2	20.2 × 19.0mm^2	20 × 11.8mm^2
クロック周波数	40Mbps	40Mbps	160Mbps
コマンド転送速度	40Mbps	40Mbps	160Mbps
データ転送速度	40Mbps	160Mbps	∼5Gbps
トリガーレート	100kHz	200kHz	1MHz

表 2: 各 ASIC の仕様

により最大 2.50V の電圧まで稼働することができ，内部回路の保護を担っている。FE-I4 における電源規格を表 3 に記す。

項目	内部素子	リニアレギュレータ	単位
最小稼働電圧	1.20	1.50	V
通常稼働電圧	1.50	1.80	V
最大稼働電圧	1.50	2.50	V
最小電流	0.05	≈ 2	A
通常電流	0.42	0.42	A
最大電流	0.55	0.55	A

表 3: FE-I4 の電圧電流規格

1.5.2 アナログ回路

FE-I4 の各ピクセル内にはアナログ回路が実装されており，センサーからのアナログ信号をデジタル信号に変換している。アナログ回路の回路図を図 8 に示す。処理は左から右に向かって行われ，センサーからの信号は最初に図の入力パッド Q_{in} から回路に入る。アナログ回路に入った信号は AC 結合された 2 つのアンプ（図中の Preamp と Amp2）により整形，増幅されて三角波となり出力される。この三角波の波高はセンサーからの信号に比例し，出力された三角波は次にディスクリミネータ回路[*2]に入る。ディスクリミネータ回路では三角波の波高に閾値を設定し，その閾値を三角波が超えた時間を ToT(Time Over Threshold) と定め，閾値を超えなかった信号はノイズとして破棄する。ToT は三角波の波高に比例することから波高情報としての意味を持ち，ToT を図 9 のようにデジタル信号に変換することでコンピュータで処理可能になる。このとき，変換され

[*2] 分別器として動作する。必要な信号とノイズを分別する回路のこと。

たデジタル信号の波高は三角波の波高に比例しない。また，閾値は TDAC という 5 ビットのレジスタを用いて調整することができる。

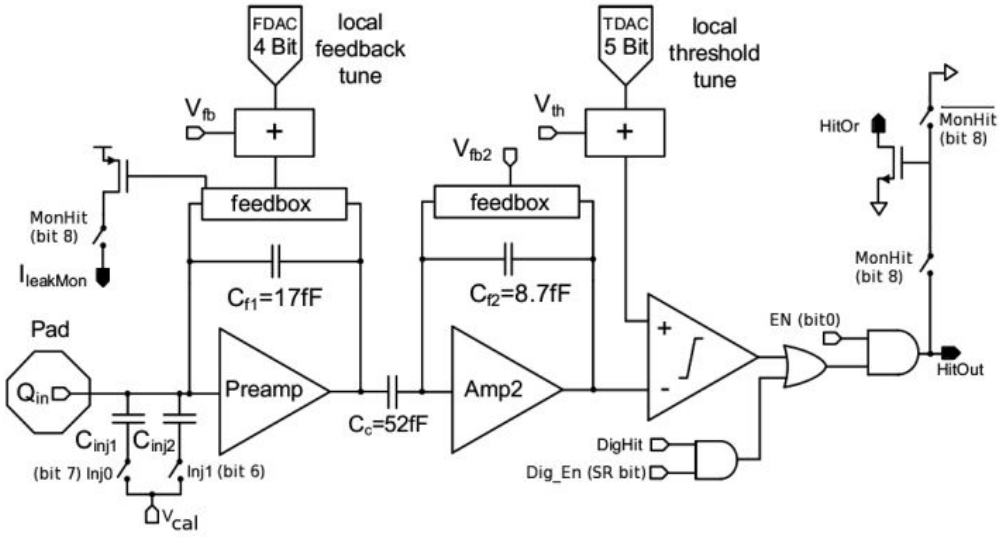


図 8: アナログ回路

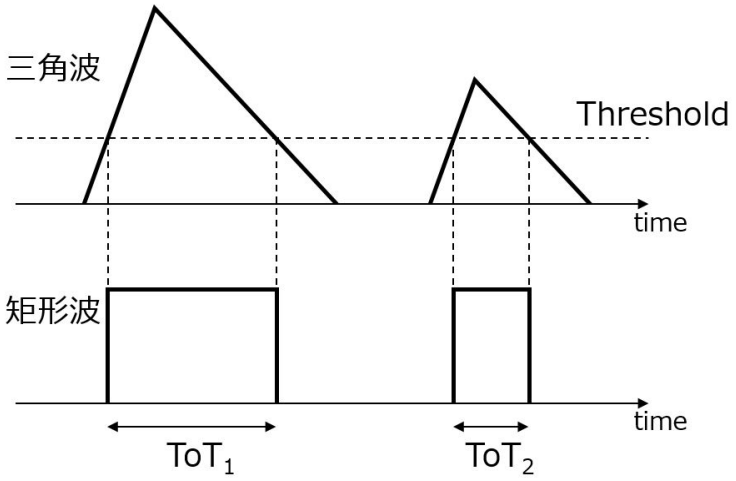


図 9: ToT(Time Over Threshold) の概念図

1.5.3　デジタル回路

デジタル回路は外部とのデジタル信号の入出力を担う回路であり，アナログ回路から出力された ToT の波高情報はデジタル回路を介して外部へ出力される。また，FE-I4 の入出力はクロック，コマンド，データ[*3]の 3 種類であり，データ出力のほかにもクロックとコマンドもデジタル回路に入力される。これらの信号はそれぞれポジティブとネガティブの差動信号で通信される。

デジタル回路から入出力されるデジタル信号は外部と Ethernet[*4]ケーブルで接続される。そのため FE-I4 の外部入出力インターフェースは RJ45 コネクタ[*5]が実装されている。FE-I4 は 3 種類の入出力が差動信号で通信されるため，計 6 本の伝送線が使われる。RJ45 コネクタは 8 本の接続箇所があるので，そのうち 2 本は使用しないことになる。

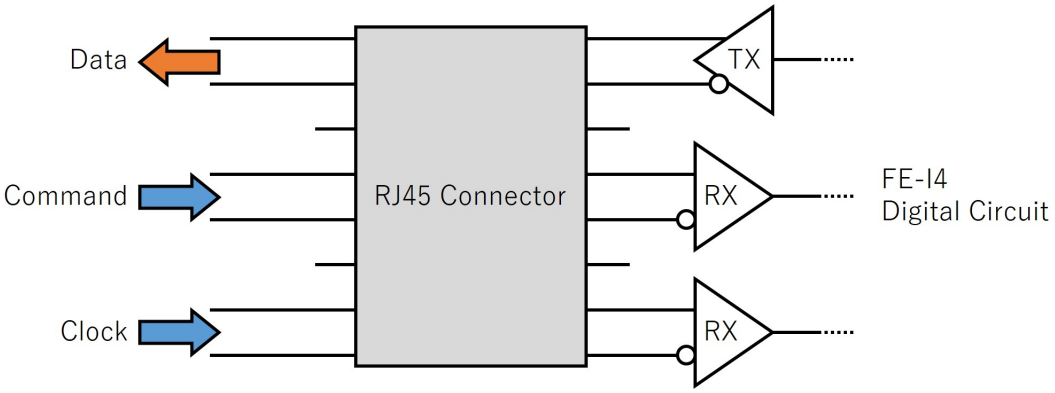


図 10: デジタル入出力信号

1.6　デモンストレータモジュール

HL-LHC アップグレードに向けて ATLAS 検出器内部のピクセル検出器を再現したデモンストレータを製作し，低温や高放射線環境下での動作確認や品質管理を行う必要がある。デモンストレータは図 11 のような構造となっている。上から FPC(Flexible Printed Circuits) という ASIC への電源供給とデータ読み出しシステムへの信号の引き渡しを兼ねたプリント基板が存在し，絶縁性接着剤でセンサーに接着されている。またセンサーはバンプボンディングで ASIC と接着されており，センサーと ASIC が一体となったものをベアモジュールと呼ぶ。ASIC には FPC からアルミワイヤーで電気的に接続されている。ASIC から発生する熱を冷却するために熱伝導性グラファイト (TPG:Thermal Pyrolytic Graphite) が ASIC の下に接着されており，TPG の下に炭素繊維

[*3] クロックとはデジタル回路で同期をとるための基準となる周期的な信号。コマンドとは機能の実行を指示するための信号。データはコマンドの動作をした結果を返す信号。

[*4] ネットワーク規格のこと。有線 LAN で構築されるネットワークにおいて，世界で最も使用されている規格でもある。

[*5] 通信ケーブルを差し込むコネクタの 1 つ。主に LAN ケーブルの接続に用いられる。

強化プラスチック (CFRP:Carbon Fiber Reinforced Plastic) が接着されている。TPG と CFRP は熱伝導性が高く，CFRP はクーリングパイプという液体の二酸化炭素が流れているパイプに接着されており，クーリングパイプによる冷却を行う。

図 11: デモンストレータ概略図

1.7 ベアモジュール

本研究では主に FE-I4 が実装されたベアモジュールを使用した。ASIC 及び FE-I4 については 1.5 で述べたが，本節では今回使用したベアモジュールについて述べる。

ベアモジュールは試験用に九州大学，高エネルギー加速器研究機構（KEK）で複数台製作され，その中の KEK101 を大阪大学からお借りして実験を行った。製作されたベアモジュールは様々な試験に用いられており，正常な動作をしないものもある。KEK101 は大阪大学にあるものでは最も状態が良かったが，それでも chipID1[*6]のチップが読み出せなかったり，chipID4 のチップにノイズが入ったりする。また，KEK101 は FE-I4 が 4 チップ実装されていることから，クアッドモジュールと呼ばれている。

[*6] ベモジュールに実装されているチップを識別するための ID。

図 12: KEK101

1.8 デジタルスキャン・アナログスキャン

ASIC の各ピクセルが正常に動作しているのか確かめる方法として，デジタルスキャンとアナログスキャンがある。ASIC の各ピクセルからデータを読み出すことをスキャンと呼ぶ。

デジタルスキャンとは各ピクセルのアナログ回路におけるディスクリミネータ回路出力部 (図 8 の DigHit) に疑似パルスを入力し，これを読み出すことでピクセルに不具合がないか検証できる。アナログ回路を通らず，次のデジタル回路を通ることからデジタルスキャンと呼ばれている。

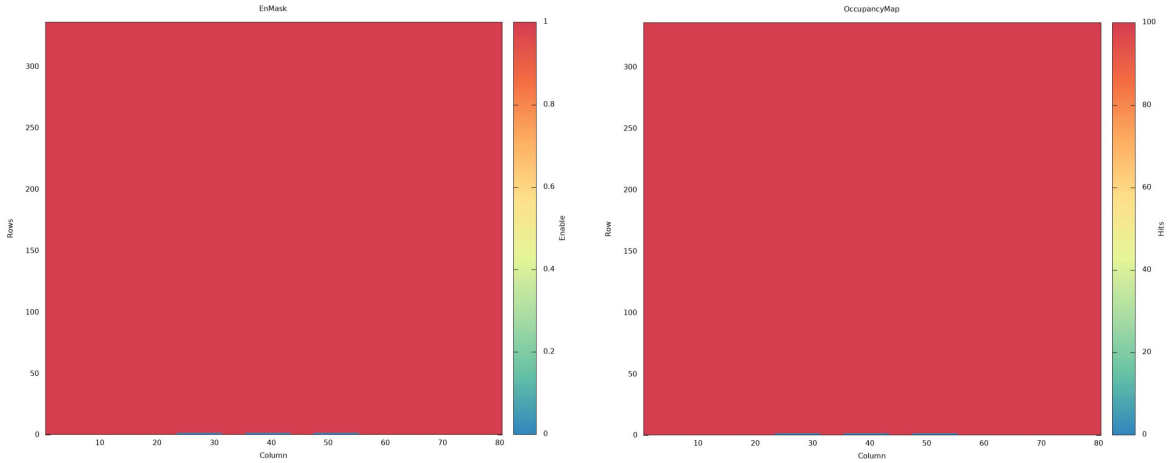
アナログスキャンとは各ピクセルのアナログ回路におけるプリアンプの入力前 (図 8 の V_{cal}) に疑似パルスを入力する。この疑似パルスはディスクリミネータ回路で設定されている閾値を超える電荷量を持つので，ノイズとして処理されることはない。アナログスキャンはアナログ回路を通過するため，アナログ回路の動作を検証することができる。

本研究は ASIC のデータ読み出しシステムに関するものであり，アナログ回路の検証等は必要ない。そのためデジタルスキャンを実行し，各ピクセルのデジタル回路からの疑似パルスを検出することでデータ読み出しシステムとしての機能を検証する。

デジタルスキャン及びアナログスキャンを実行すると，EnMask と OccupancyMap という 2 種類のマップが画像ファイルとして出力される。EnMask は読み出し可能 (enable) なピクセルを表

すマップで，OccupancyMap は読み出すことができた疑似パルスの数と個数である。ここで読み出すことのできた疑似パルスのことは” ヒット” と呼ばれる。なお，アナログスキャン，デジタルスキャン共に 100 個の疑似パルスを打ち込む。

図 13 に正常にスキャンできた時の EnMask と OccupancyMap を示す。どちらのマップも縦軸が約 330，横軸が約 80 となっている。これは FE-I4 のチップ 1 つを構成するピクセル数が 336 × 80 個ということからもわかる通り，2 種類のマップは FE-I4 のチップを表している。図 13a の EnMask は 1 が読み出し可能，0 が読み出し不可を示しており，右端にあるカラーバーによると 1 を赤，0 を青で示す。また，図 13b の OccupancyMap のカラーバーは 0 から 100 のヒット数を色で表している。



(a) EnMask　　(b) OccupancyMap

図 13: 正常にスキャンできた時の EnMask と OccupancyMap

2　データ読み出しシステムの構築

京都教育大学ではここ数年 ATLAS JAPAN グループにおいて，モジュールの組み立てや検証をするアセンブリグループでの活動を主としており，データ転送技術等を研究する DAQ グループとしての参加はなかった。そのため，最新のデータ読み出しシステム[*7]についての実験環境が整っていなかった。このような中で村田が DAQ グループに参加したことから京都教育大学でのデータ読み出しシステムを構築するに至った。この章ではデータ読み出しシステムの各機能と，研究室レベルでの読み出しシステムの構築について述べる。

2.1　データ読み出しの流れ

ピクセル検出器は通過した粒子により生成された電子正孔対から，微弱なアナログ信号を発生させる。この微弱なアナログ信号はそのままでは計測可能な信号ではないので，アンプ回路や波形成型回路などで計測に適した波形に調整し，アナログデジタル変換をする必要があり，この処理を ASIC が行う。

ASIC からのデジタル信号をコンピュータで読み出してデータ保存や処理をするためには，ASIC-コンピュータ間での入出力インターフェースや通信規格を合わせなければならない。そのため，ASIC からのデジタル信号をコンピュータの入出力規格に適合するように処理し，コンピュータと ASIC のデータのやり取りを仲介するシステムが必要である。これを DAQ(Data AcQuisition) システムと呼ぶ。これら一連のデータ読み出しの流れを図 14 に示す。

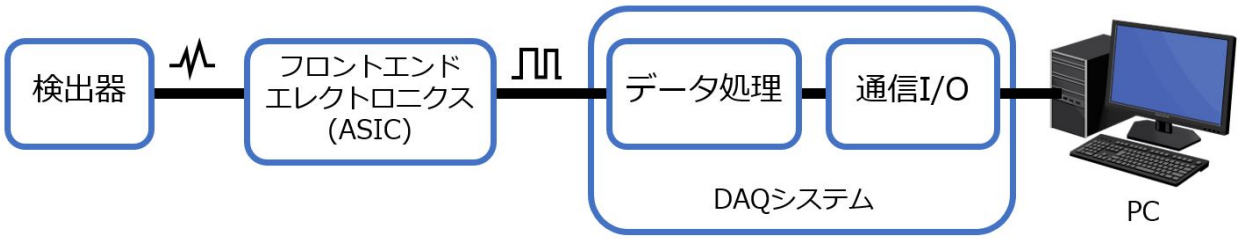


図 14: データ読み出しシステム概略図

[*7] 粒子を検出した検出器からの信号をコンピュータのような記憶媒体が処理できる形のデータに調整や変換を行うシステムのことをデータ読み出しシステムと呼ぶ。

2.2　DAQ システム

2 にも記したように DAQ システムとは ASIC からのデジタル信号をコンピュータが処理できるように通信規格や I/O を仲介するシステムである。DAQ システムは FPGA と読み出し用のソフトウェア，及びファームウェアによって構成される。

2.2.1　FPGA

FPGA とは Field Programmable Gate Array の略で，プログラムで書き換え可能な集積回路のことを指す。FPGA は汎用的な論理回路を一つの小規模なパッケージに集約した集積回路であるロジック IC という分類に属している。ロジック IC は標準 IC とカスタム IC に分別され，カスタム IC は IC を用いる目的ごとで専用に作られる IC のことである（文献 [1] を参照）。 1.5 で説明した ASIC がここに分類される。

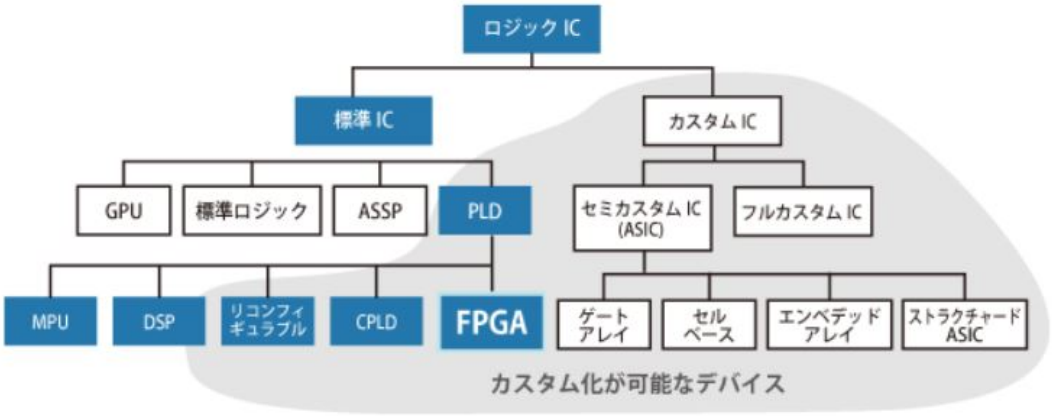


図 15: IC の分類分け

今回の読み出しシステム構築に際して，Xilinx, Inc. の Kintex-7 FPGA 搭載ボードの KC705 を用いた。KC705 は Xilinx, Inc. から一般的に購入できるといった手軽さや，多様な入出力インターフェースによる通信が可能であることから，汎用ボードとして適しており複数の読み出しソフトウェアや ASIC に対応している。以下に KC705 に搭載されている FPGA 及び記憶素子を記す。

- Kintex-7 XC7K325T-2FFG900 FPGA
- 1GB DDR3 メモリ S.O.DIMM
- 128MB パラレルフラッシュメモリ
- Quad シリアルフラッシュメモリ

さらに，KC705 の入出力インターフェースには Xilinx, Inc. が提供する高速シリアルトラン

シーバー[*8]の 1 つである GTX トランシーバーによる通信に対応しているものがある。GTX とは Gigabit Transceiver の略称[*9]で最大 12.5Gbps の通信が可能なトランシーバである。GTX を用いることにより現行 ASIC の FE-I4 はもちろん，新型 ASIC の RD53A のデータ転送速度にも対応できる。以下に GTX トランシーバーによる通信が可能な主な入出力インターフェースを挙げる（文献 [2] を参照）とともに，図 20 に KC705 における配置を示す。

SMA コネクタ

同軸ケーブル[*10]を接続できるコネクタ。KC705 には通常の User SMA と GTX SMA トランシーバーが備わっている。GTX SMA トランシーバーには TX(Transmitter)，RX(Receiver)，REFCLK(Reference Clock) 用のコネクタが，ポジティブとネガティブの 2 つずつ実装されている。

Ethernet コネクタ

Ethernet でネットワーク接続をするための RJ45 コネクタ。10/100/1000 Mbps の 3 つの規格に対応している。

PCI Express

PCI Epress は送信と受信の差動信号のペアを単位とした「レーン」で構成されるシリアルインターフェースであり，1 レーンの PCI Express を PCI Express × 1 と表す。KC705 は 8 レーンでの通信が可能である。PCI Express 主にコンピュータの拡張スロット規格として用いられ，インターフェースの増設やグラフィックボードの接続などに用いられる。また，PCI Express には現在 6 つのリビジョンが発表されており，KC705 は PCI Express 1.1(Gen1) と PCI Express 2.0(Gen2) に対応している。

”Express” を省略して”PCIe” と表記されることが多い。

FMC コネクタ

FMC(FPGA Mezzanine Card) とは FPGA が搭載されている基板専用で用いられるインターフェースカードのことで，FMC のコネクタのことを FMC コネクタと呼ぶ。FMC は最大 10Gbps の信号処理速度が保証されている。また，シンプルな構造や多くのインターフェース変換基盤が開発されていることから，I/O インターフェースとしての高い汎用性を持つ。

KC705 には HPC と LPC という 2 つの FMC コネクタ実装されている。どちらのコネクタも同じ 10 × 40 のフォームファクタで接続されるが，HPC は 400 すべてのピンを使用し，LPC は部分的に 160 のピンを使用するようになっている。[3]

[*8] トランシーバーとはトランスミッタ (送信機) とレシーバー (受信機) 両方の機能を備えた送受信機のこと。

[*9] ”X” に意味は無い。

[*10] 同軸ケーブルとは，中心の内部導体を軸として順にポリウレタンの絶縁体，アルミ線で編まれた外部導体，外部被膜のシースが同心円状に配置されている。外部導体が電磁シールドの役割を果たし，外部からの影響を受けにくい特徴がある。

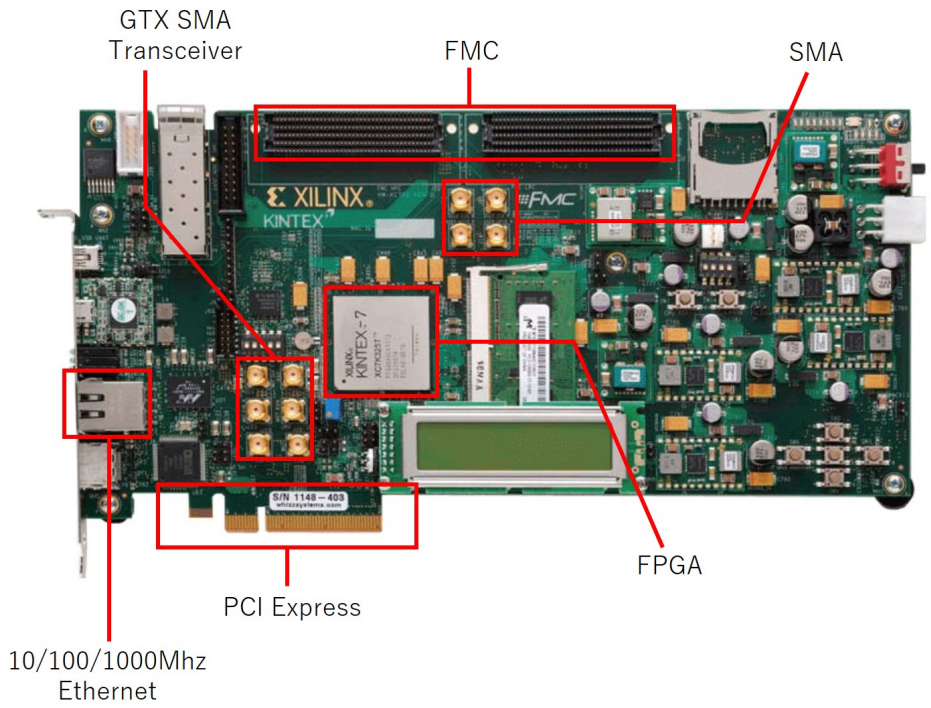


図 16: KC705

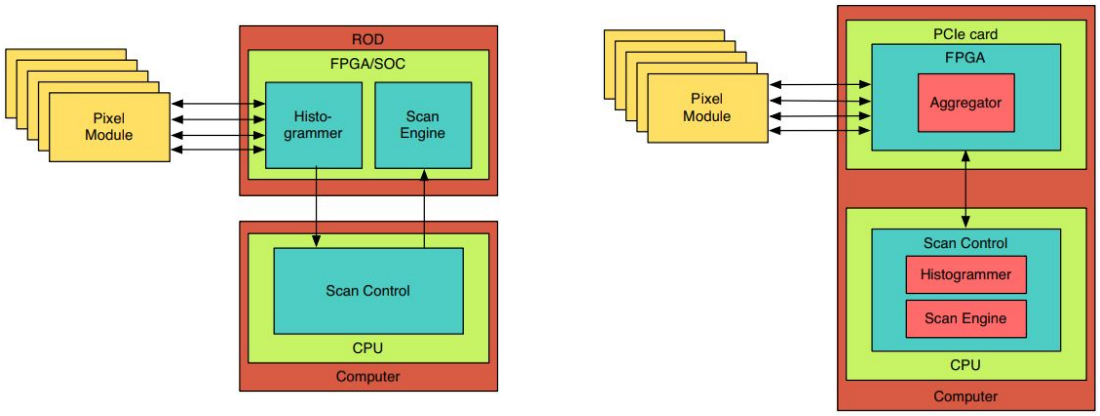
2.2.2 データ読み出しソフトウェア・ファームウェア

データ読み出しソフトウェア及びファームウェアには YARR(Yet Another Rapid Readout)[4] を用いた。ファームウェアとはハードウェアの動作を制御するソフトウェアのことであり，YARR はソフトウェアとファームウェアの両方を提供している。本章では YARR が提供しているソフトウェアとファームウェアを用いた。

ファームウェアは広義的にはソフトウェアでもあるので，以降はファームウェアも含めて「データ読み出しソフトウェア」と呼ぶ。

YARR は現行 ASIC の FE-I4 や，FE-I4 の技術を用いた RD53A のデモンストレータである FE65，新型 ASIC の RD53A の読み出しにも対応したデータ読み出しソフトウェアであり，複数の FPGA ボードに対応するファームウェアを備えている。また，YARR では再構築可能な入出力インターフェースとして市販の FPGA を用いることで汎用性を高めている。

従来の読み出しシステムは図 17a のように，ROD(Read Out Driver) というハードウェアをピクセルモジュールに接続し，ROD でデータ処理やスキャンを行っていた。この読み出しシステムだと ROD とコンピュータ間の通信インターフェースがボトルネックとなるため，ROD 側でデータ圧縮を行いコンピュータへ圧縮されたデータを転送していた。



(a) 従来までの読み出し

(b) YARR による読み出し

図 17: YARR と従来のソフトウェアの読み出し概念図

従来の読み出しシステムでは以下のような問題 [5] が発生する。

- データ圧縮によりデータの損失が発生する。
- データ処理や圧縮などの多くの機能を ROD が担うことで ROD の専門性が高くなり，専用のボードを利用することになるため，ボードの入手性が悪い。
- 読み出しの実行やデータの処理はファームウェアにより動作しており，高水準言語[*11]に慣れている研究者が多い中ファームウェアの開発ができる研究者が今後少なくなる。

これらの問題を解決するために，YARR では PCIe での高速シリアル通信を用いてすべてのデータ処理とスキャンをコンピュータで制御している。これにより FPGA のファームウェアはピクセルモジュールと通信するためだけの必要最低限のもので済み，構造が分かりやすくなっている。また，FPGA ファームウェアの単純化により市販の FPGA ボードを利用することが可能になり，読み出しシステムの構築や部品の調達も容易である。

2.2.3 インターフェースカード

1.7 で述べたように KEK101 は Ethernet ケーブルで外部との通信を行う。そのため FPGA ボードのインターフェースを Ethrenet ケーブル対応の RJ45 コネクタに変換しなければならない。FPGA ボードからの信号は LVDS(Low Voltage Differential Signaling) という差動信号の規格で出力される。LVDS とは図 18 の上図のような，ポジティブ信号 (V_{OH}) とネガティブ信号 (V_{OL}) の 2 本の低電圧差動信号を，V_{OH} と V_{OL} の平均であるコモンモード電圧 (V_{OC}) を軸に，スイングさせながら伝送させる通信技術である。LVDS のコモンモード電圧は 1.2V，差動信号の電位差は 350mV が規格とされているがある程度のマージンが設けられている。差動信号は一見”Low”

[*11] 高水準言語とは言語自体が人間にとってわかりやすく，メモリ制御や入出力制御のを意識しなくてよいプログラミング言語のこと。対義語は低水準言語。

と”High” の区別がわからないが，ポジティブ信号とネガティブ信号の差分 ($V_{OH} - V_{OL}$) をとることで”Low” と”High” を得ることができる。IC などでこの処理を行い，GND(0V) を基準として”High” と”Low” が決まるように波形を調節することでシングルエンド信号が得られる。シングルエンド信号の中で＋ 2.0V 以上を”High”，＋ 0.8V 以下を”Low” として基準を定めたものを LVTTL(Low Voltage Transistor Transistor Logic) とよぶ。インターフェースカードには電源電圧 3.3V がかけられており，インターフェース上での LVTTL のスイングは最大 3.3V となる。

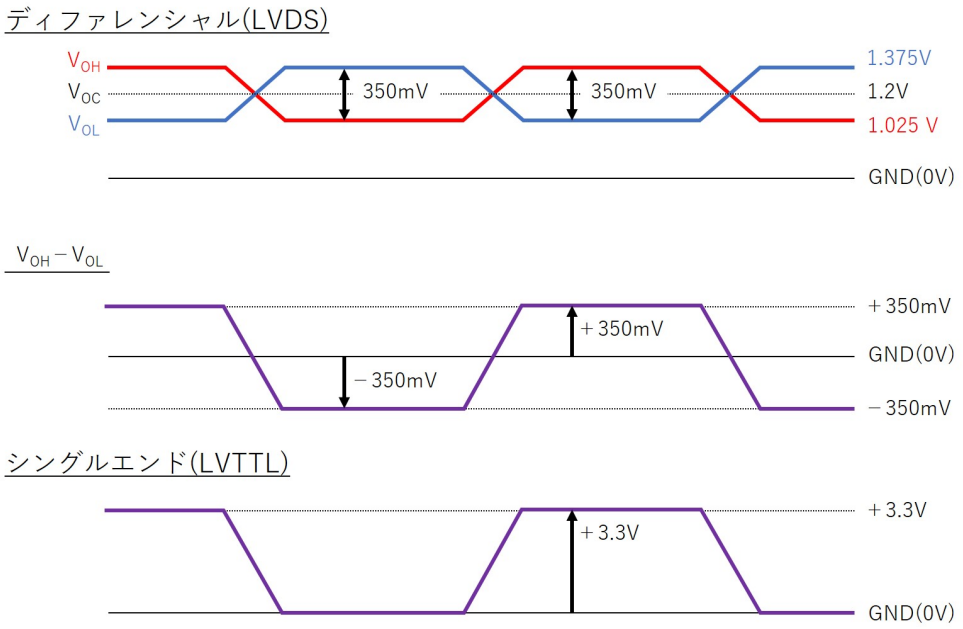


図 18: ディファレンシャル信号，ディファレンシャル信号の差分（$V_{OH} - V_{OL}$），シングルエンド信号

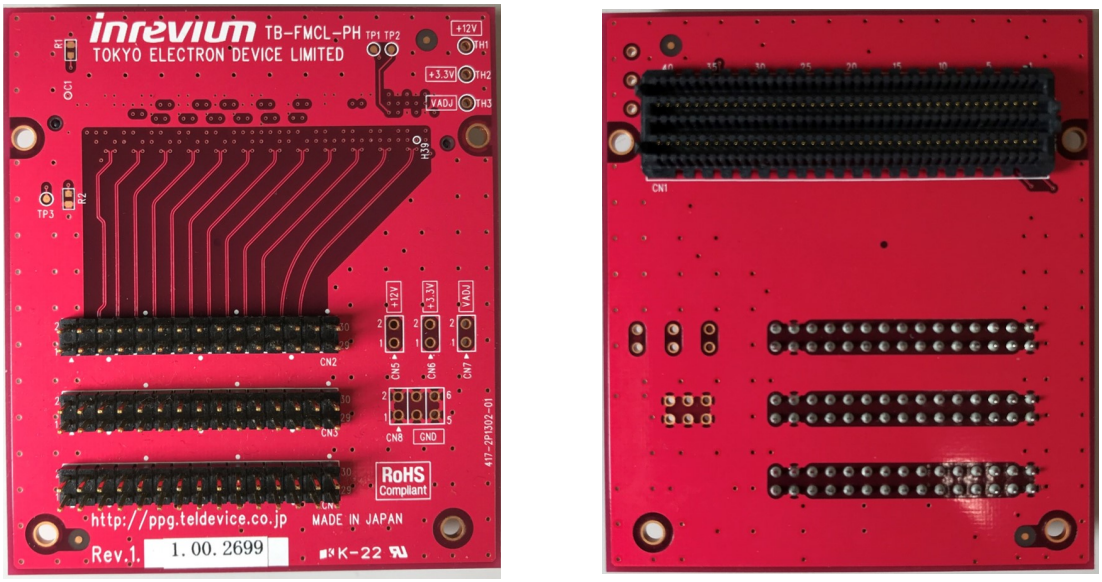
差動信号はシングルエンド信号と比べて，以下のような利点がある。

- コモンモード電圧と信号振幅が小さいため消費電力が少ない。
- 振幅が小さいことから論理状態間の遷移時間が短く，帯域幅[*12]を広くとることができるのでデータレートが高い。
- 同じく振幅の小ささから高速伝送が可能。
- 信号がノイズによる影響を受けても，2 つの信号が同じような影響を受けるので差分をとった時の影響が少ない。
- 差動信号では 2 線に異なる方向に電流が流れるので，磁力線が打ち消しあう方向に発生し，ノイズの発生を抑えられる。

[*12] データを転送する際に用いる周波数の幅。最高周波数と最低周波数の差で求められる。この値が大きいほどデータ転送量が大きい。

FE-I4 の信号規格は LVDS よりも低い電圧レベルでの差動信号に対応している。これらの理由からインターフェースを変換し，かつ FPGA からの電圧レベルを調整するインターフェースカードが必要となる。

まず，インターフェースカードを FPGA ボードに接続できるように FMC を用いる。FMC は 2.2.1 でも述べたように，高い汎用性から用途に合わせてインターフェースを変換できる特徴があるので，FMC を用いてピンヘッダにインターフェースを変換する。これには東京エレクトロンデバイス株式会社の inrevium シリーズ TB-FMCL-PH[7] を使用した。TB-FMCL-PH は FMC コネクタと 2.54mm ピッチのピンヘッダを変換する変換基盤であり，基板上の配線に差動配線が採用されているので高速な差動信号を通信させることが可能である。ピンヘッダは表側に 3 列，FMC コネクタは裏面上部に配置されており，FMC コネクタの各ピンは 3 つのピンヘッダに分かれて配線されている。



(a) 表　　(b) 裏

図 19: inrevium TB-FMCL-PH 基板両面図

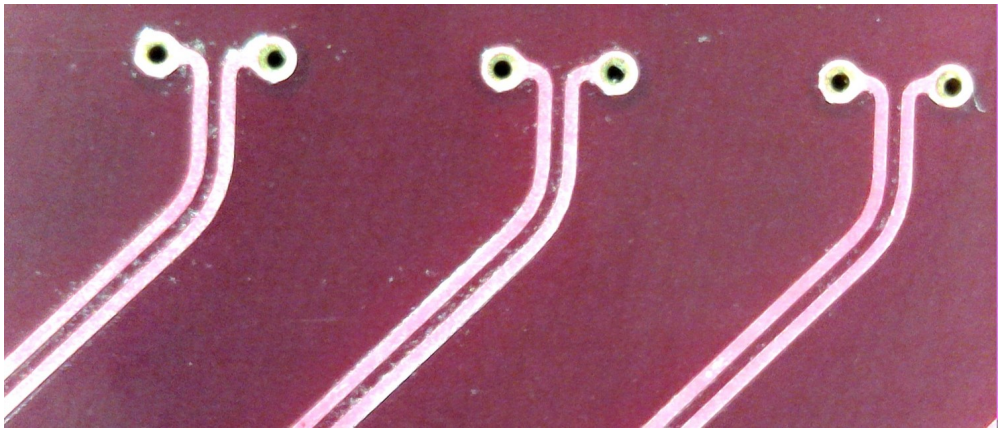


図 20: inrevium TB-FMCL-PH の拡大図。差動配線が裏面の FMC コネクタのピンにつながっている。

変換されたピンヘッダに RJ45 へのインターフェース変換と，電圧レベルの降下を担うインターフェースカードを接続する。このインターフェースカードは後述の 4 チップ同時読み出し用に作成したインターフェースカードを用いた。図 21 にインターフェースカードの回路図を示す。

図 21 の上図はコマンドとクロックが FPGA から FE-I4 へと流れる際に通る回路である。FPGA から LVDS で送られてきた信号は DS90LV028ATM という論理 IC へ流れる。DS90LV028ATM は LVDS 信号を受け取ると，LVTTL(Low Voltage Transistor Transistor Logic) というシングルエンド信号に信号を変換する。DS90LV028ATM から出た LVTTL 信号は DS90LV027M に入り，もう一度 LVDS 信号として出力される。この 2 つの論理 IC は伝送路が長くなることによる信号のなまりを防ぐものである。さらにこの LVDS 信号を 1k Ω抵抗でコモンモード電圧と信号振幅を降下させることで FE-I4 に対応した信号となる。

また，図 21 の下図は FE-I4 からデータが返されるときに通過する回路である。DS90LV001TM は入力された LVDS 信号を同じ LVDS で出力するバッファ回路で，図 21 の上図と同じ意味がある。

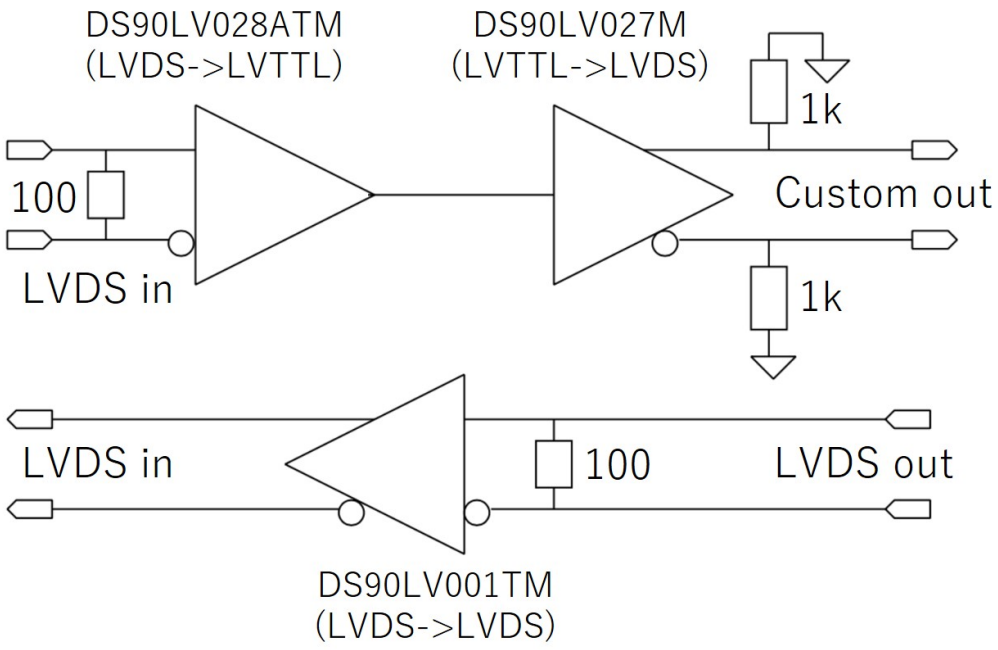
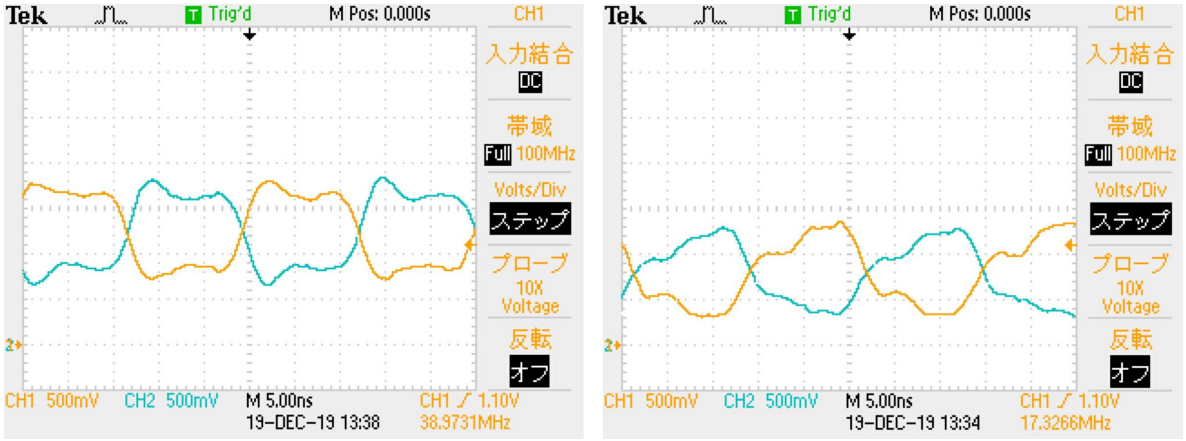


図 21: インターフェースカード回路図

図 22a，22b にインターフェース回路通過前後の差動信号をプローブし，オシロスコープで確認したものを示す。回路通過前と回路通過後のパルスを比べると，通過前の差動信号のコモンモード電圧は 1.2V あたりであるのに対し，通過後の差動信号のコモンモード電圧は 800mV 程度まで低下しているのがわかる。



(a) 回路通過前　　(b) 回路通過後

図 22: インターフェース回路通過前後の差動信号

2.3　コンピュータへの実装

これまでに述べた DAQ システムをコンピュータに実装した。コンピュータの環境を以下に記す。また，コンピュータに実装した写真を図 23 に示す。写真では分かりづらいが，KC705 とコンピュータは PCIe で接続されている。

CPU	Intel(R) Core(TM) i5-4570 CPU @ 3.20GHz
Memory	Komputerbay 4GB DDR3 PC3-12800 1600MHz × 2
Motherboard	Gigabyte Z87MX-D3H-CF
Storage	TS240GSSD220S
OS	CentOS Linux release 7.5.1804

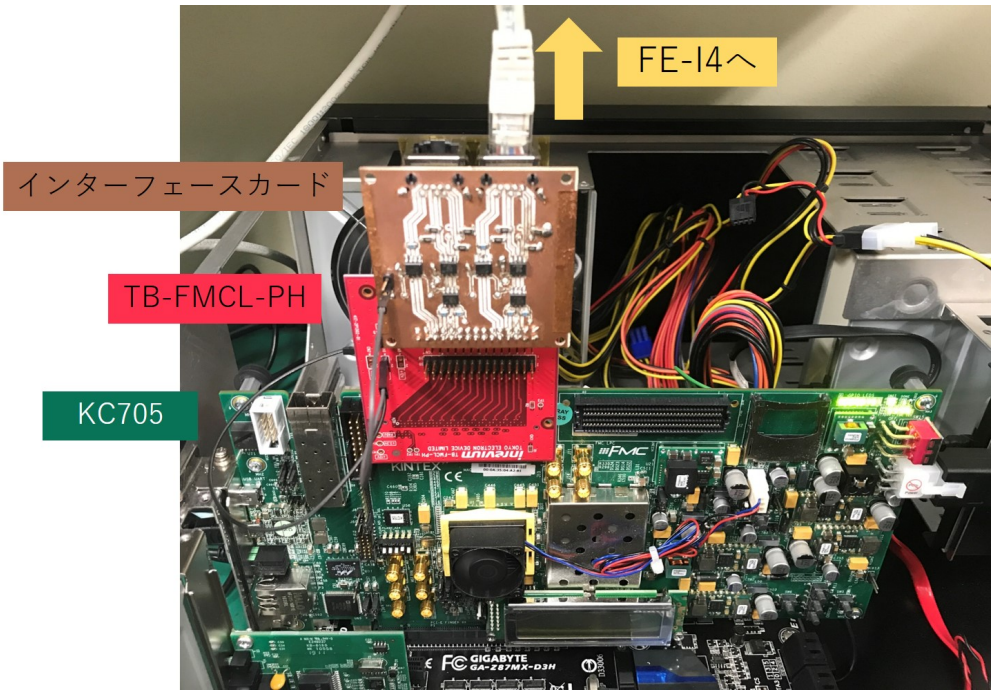


図 23: 実装した DAQ システムの写真

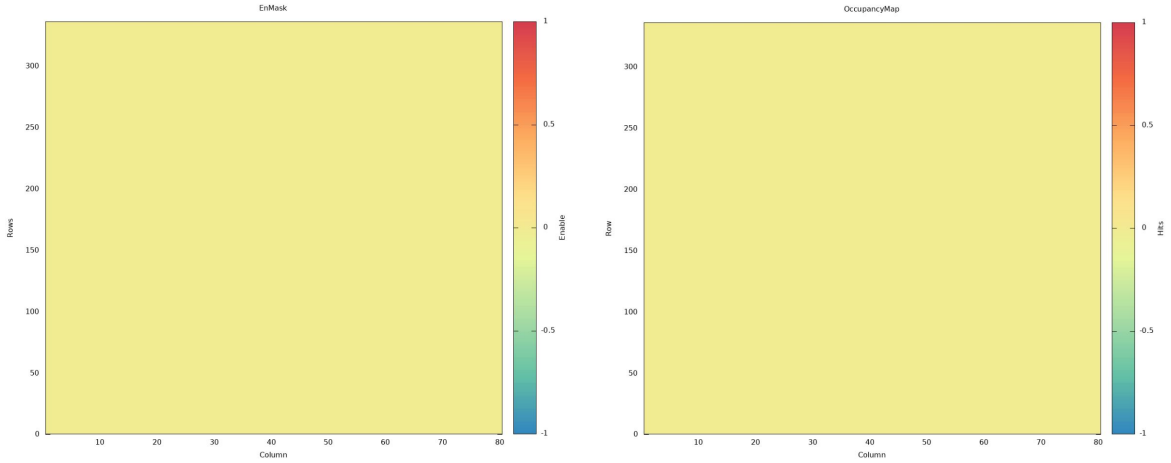
2.4　スキャン結果と考察

上記のデータ読み出しシステムを用いてスキャンを行ったが，スキャン自体は実行され EnMask と OccupancyMap が生成されたものの，図 24 のように EnMask，OccupancyMap ともにすべてのピクセルが”0” を表しており，疑似パルスを読み出すことができなかった。

YARR ソフトウェアの動作は正常に行われてスキャンを終了し，マップを生成していることか

らソフトウェアの問題ではないと考えられる。また，インターフェースカードも図 22 に示したように正常な動作をしている。正常なスキャンができない原因として，ファームウェアの不整合が考えられる。今回使用したファームウェアは YARR が提供しているものをそのまま用いたことから，本大学で構築したデータ読み出しシステムのピン配置や配線が異なることが考えられる。この問題を解決するため，現在新たなファームウェアを開発中である。

また，次章に記す大阪大学で実施した 4 チップ読み出しでは正常なスキャンができた。



(a) EnMask　　(b) OccupancyMap

図 24: 構築したデータ読み出しシステムによる EnMask と OccupancyMap

3　4 チップ読み出し

3.1　研究背景

現行 ATLAS 内部飛跡検出器において，最大 2 チップの読み出しモジュールが用いられている。ATLAS アップグレードに向けて開発されている新型 ASIC の RD53A には 4 つのチップが搭載されており，読み出しシステムも 4 チップでの読み出しに対応しなければならない。

4 チップ読み出しは，以前 SEABAS と SiTCP を用いて行われていた。SEABAS(Soi EvAluation BoArd with Sitcp)[8] とは SiTPC を搭載した汎用 DAQ ボードであり，SiTPC は FPGA 上でハードウェアによる TCP/IP の処理を可能にしたネットワークプロセッサである。すべてのプロトコルをハードウェア上で処理するので，帯域上限 100Mbps という速度で安定して転送が可能である。SEABAS ボードによる 4 チップ読み出しシステムを図 25 に示す。

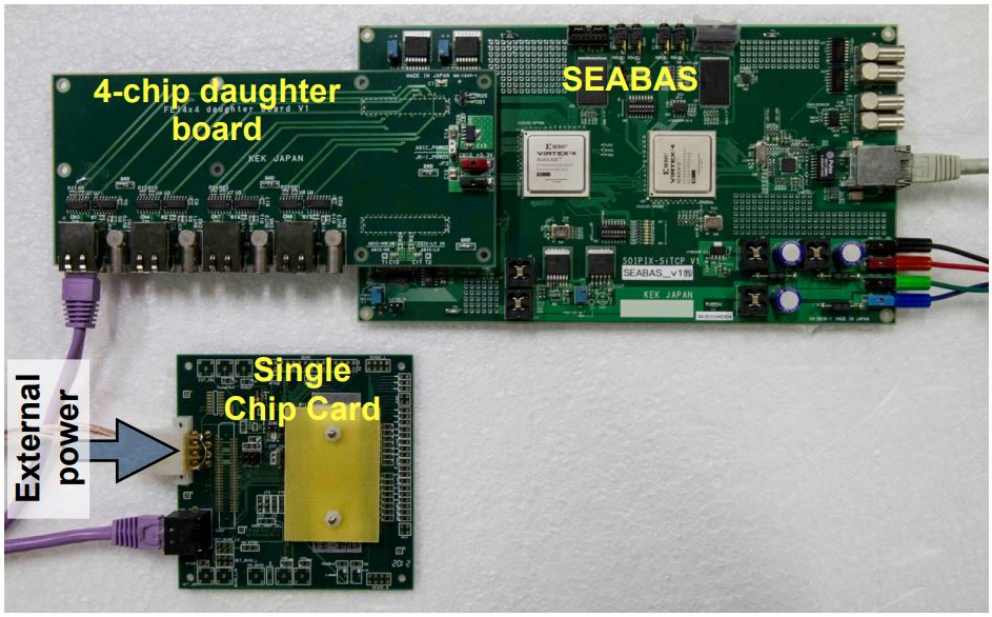


図 25: SEABAS ボードによる 4 チップ読み出しシステム [9]

SEABAS と SiTCP は FE-I4 や RD53A の最大転送速度を満たせないことから，これら 2 つの ASIC 読み出しには用いられておらず，ATLAS 日本グループでは主に YARR と KC705 を用いている。だが，FE-I4 と RD53A の 4 チップ読み出しシステムは開発されていなかった。また，ジュネーブ大学が 4 チップ読み出し用のデモンストレータモジュールを開発予定であることから 4 チップ読み出しシステムの開発を行った。

3.2　4 チップ読み出しシステムの開発

4 チップ読み出しシステムの構造は 2 章で述べた読み出しシステムにおける DAQ システムと基本的に同じである。だが，これまで開発されてきた FE-I4 のスキャンが可能なインターフェースカードは，図 21 のコマンド，クロック，データの送受信を仲介する回路が 1 本しか実装されておらず，1 チップの読み出ししかできなかった。そのため，4 チップ読み出し用のインターフェースカードを新たに作成するとともに，4 チップ読み出しに対応したファームウェアを開発した。

4 チップ読み出し用のインターフェースカードは RJ45 コネクタを計 4 口実装し，FE-I4 クアッドモジュールの各チップと通信ができるようにする。図 26 に 4 チップ読み出し用インターフェースカードを用いた読み出しシステム開発の模式図を示す。

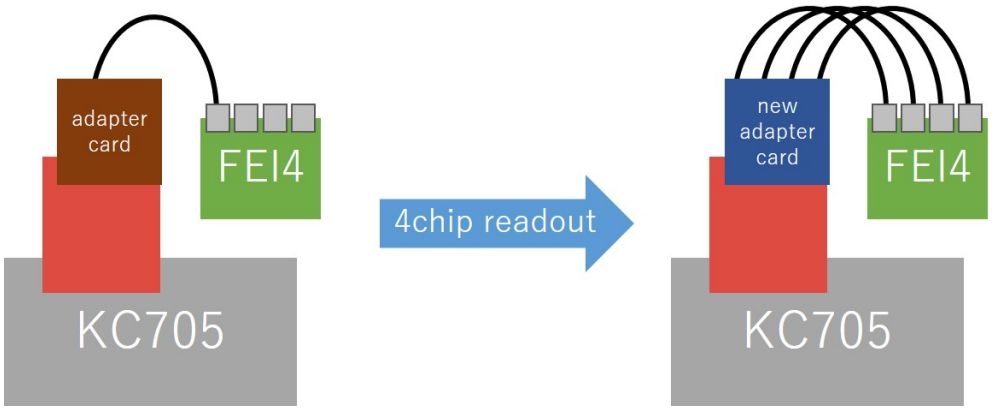


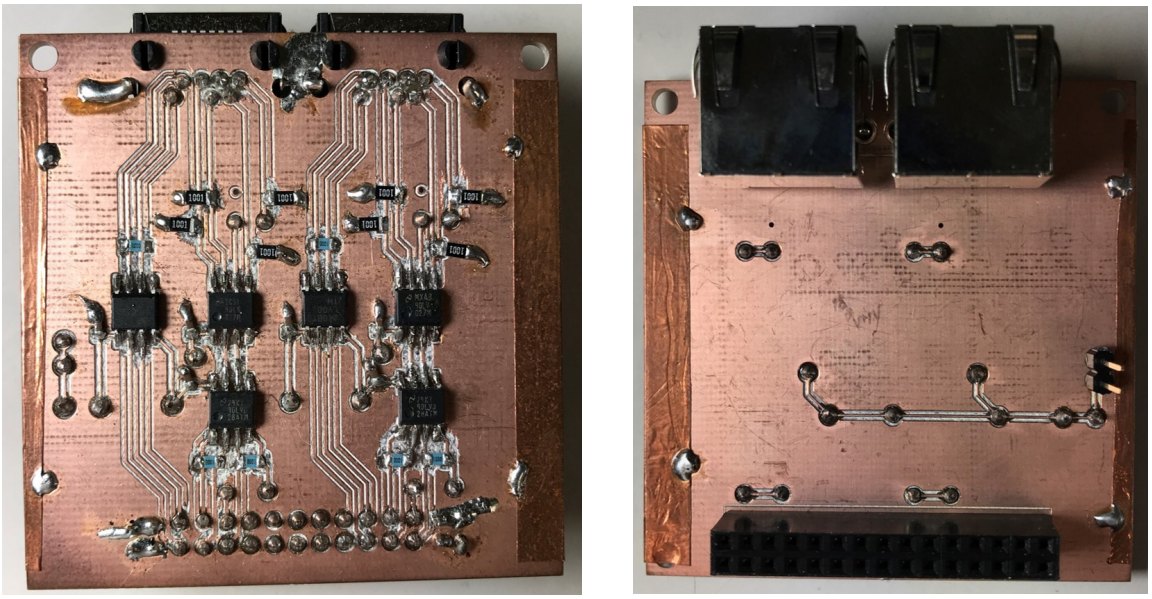
図 26: 4 チップ読み出し用インターフェースカードを用いた読み出しシステムの模式図

3.2.1　4 チップ読み出し用インターフェースカードの作成

4 チップ読み出し用のインターフェースカードのデザインとしてまず考えられるのは，1 つの基板に 4 本の読み出し回路を実装するものである。だが 1 つの基板に図 21 の回路を 4 本実装すると，基板自体の大きさが大きくなり GND[*13]の値が揺らいでしまうことや，回路同士で干渉をおこす問題などで正確な読み出しができない。そこで，図 21 の回路を 2 本実装したインターフェースカードを 2 つ作成し，メザニンカードの TB-FMCL-PH にその 2 つの基盤を取り付けることで計 4 口のインターフェースを設けることにした。

2 本の回路が実装されるインターフェースカードは大阪大学の南條氏の保持する基板加工機 (MITS AutoLab) で基板を製作し，その基盤に必要な IC や抵抗，コネクタ等をはんだ付けした。図 27 にはんだ付けをして完成させたインターフェースカードを示す。

[*13] 信号電圧の基準となる電圧。0V と設定されることが多い。



(a) 表　　(b) 裏

図 27: 4 チップ読み出し用インターフェースカード

このインターフェースカードを TB-FMCL-PH メザニンカードに接続するのだが，TB-FMCL-PH のピンヘッダは図 28 のようにすべて高さが同じであり，インターフェースカードを接続しようとしてもお互いが接触してうまく配置できない。そこで，通常の読み出しに用いる図 28 の CN4 からは今まで通りの方法でインターフェースカードを接続し，もう 1 枚のインターフェースカードは CN2 から基板連結用のロングピン型ピンソケットを 2 個用いてピンヘッダから高さを出して接続した。図 29，30 に基板連結用ピンソケットと，高さを出して接続したインターフェースカードを示す。

図 31 に TB-FMCL-PH メザニンカードを用いて KC705 に実装した写真を示す。高さを出した方のインターフェースカードが下のインターフェースカードに接触してショートしないように，ポリイミドテープを RJ45 コネクタに張り付けて絶縁している。

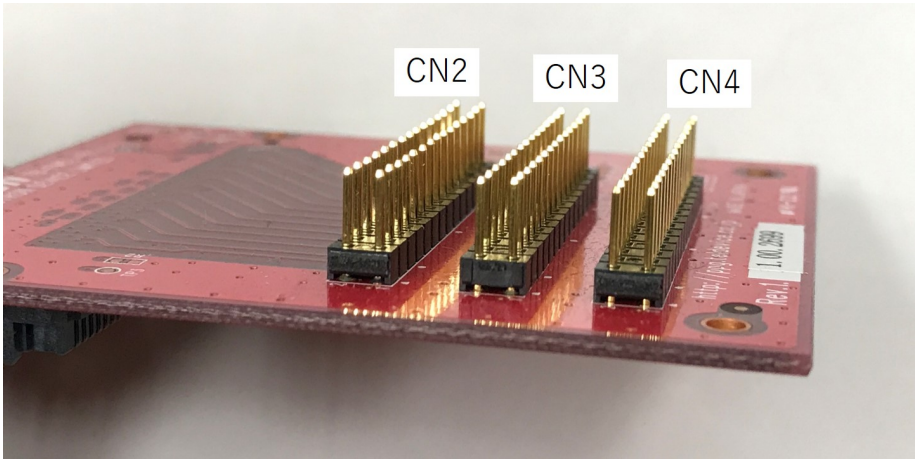


図 28: TB-FMCL-PH メザニンカードのピンヘッダ部分

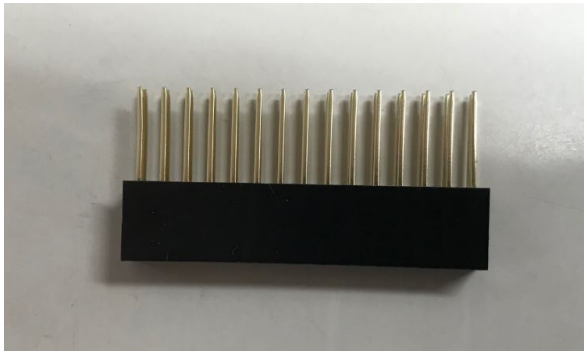


図 29: 基板連結ピンソケット

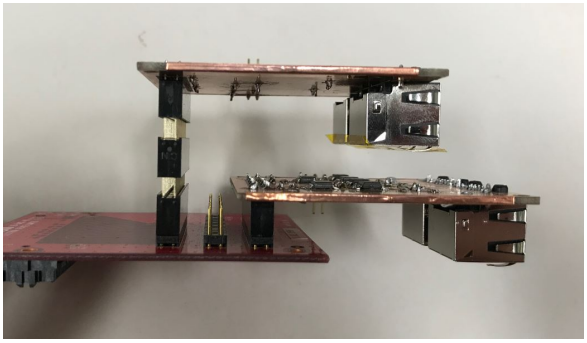


図 30: インターフェースカード配置

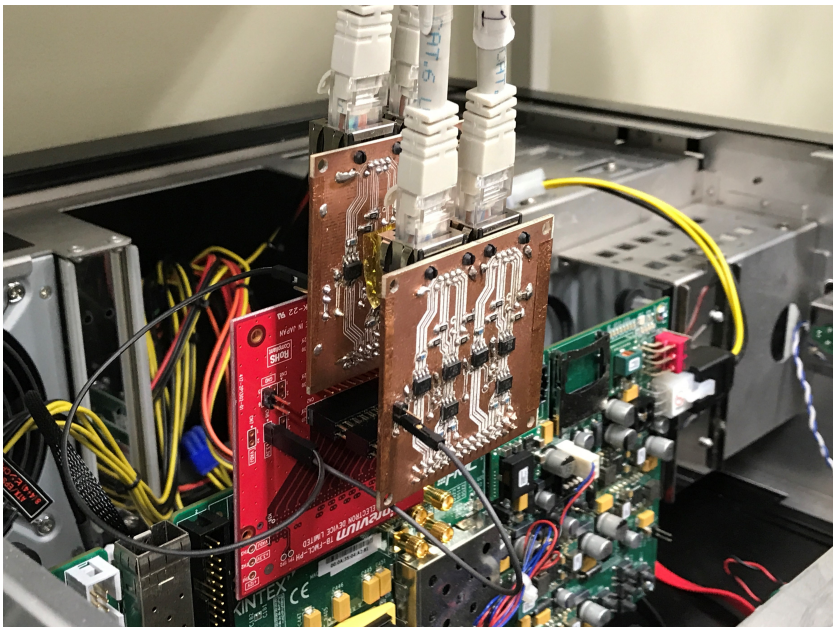


図 31: TB-FMCL-PH メザニンカードを用いて KC705 に実装した写真（京都教育大学で撮影）

図 30 の高さを出すために用いた基板連結用ピンソケットはあくまで複数基板の連結に用いられるものであり，今回の目的のようなピンを延長するためのものではない。また，ピンの長さが延長されたことなども含め，これらがどのような影響を与えるのか検証するために，ピンソケットの数を変えて 2 チップ読み出しをデジタルスキャンで行った。図 32，33，34 に示すように，ピンソケットを 0 個，1 個，2 個と増やしてスキャンを実行した。インターフェースカード以外の DAQ システムは大阪大学のものを用いて，KEK101 ベアモジュールの chipID3 と chipID4 のチップをスキャン対象とした。また，ファームウェアは 3.2.2 で作成した 4 チップ読み出し用ファームウェアを用いた。スキャンの結果を図 35，36，37 に示す。

chipID4 の OccupancyMap は一部のピクセルの Hit 数が 100 を超えて最大 400 という値を出してしまっている。ゆえにマップが 400 を最大ヒットの赤に指定してしまい，全体が緑色になっている。chipID4 ではいつもこのようなスキャン結果が出ることから一部ピクセルの不具合であると考えられる。よって，ピンソケットの影響を受けずに正しくスキャンできていると考える。

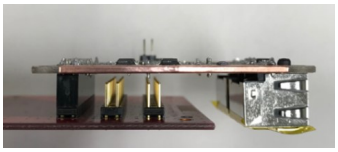


図 32: ピンソケット 0 個

図 33: ピンソケット 1 個

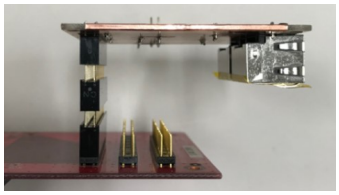
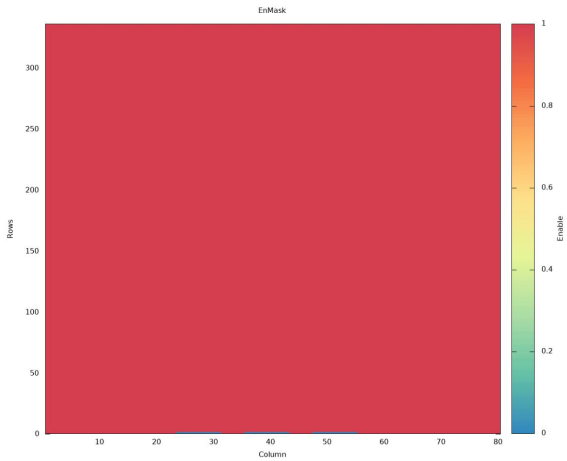
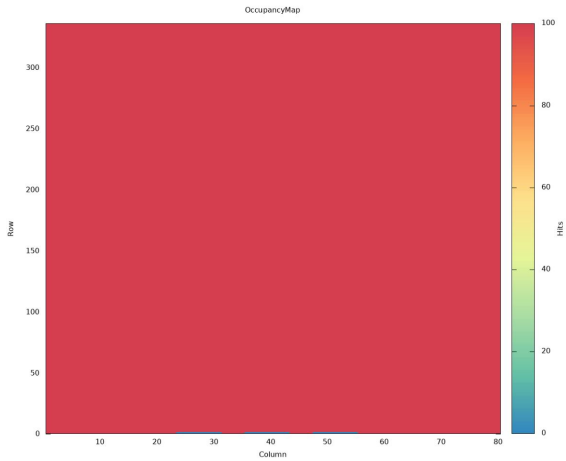


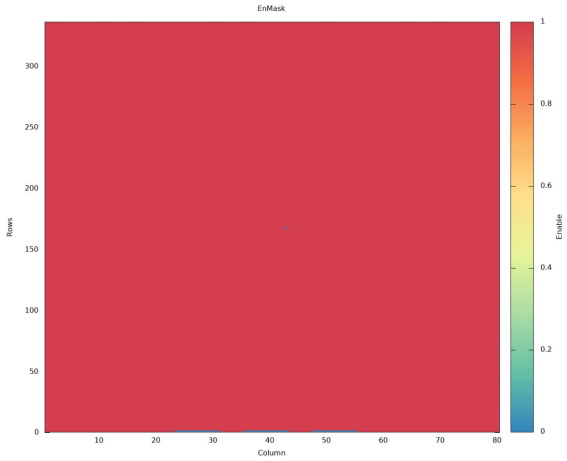
図 34: ピンソケット 2 個



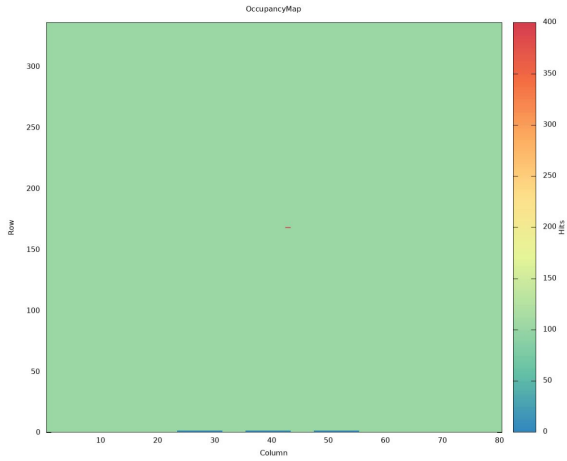
(a) chipID3，EnMask



(b) chipID3，OccupancyMap

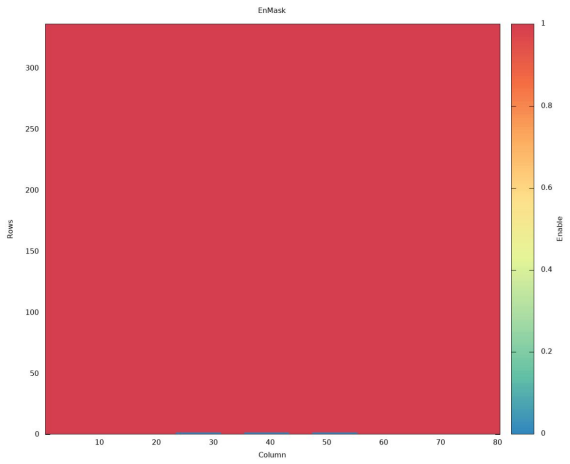


(c) chipID4，EnMask

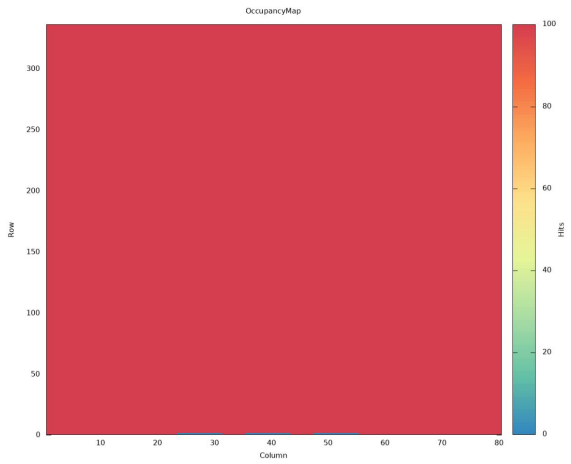


(d) chipID4，OccumapcyMap

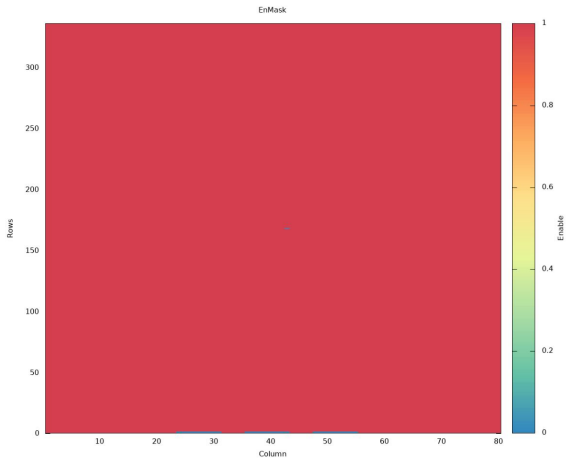
図 35: ピンソケット 0 個でのスキャン結果



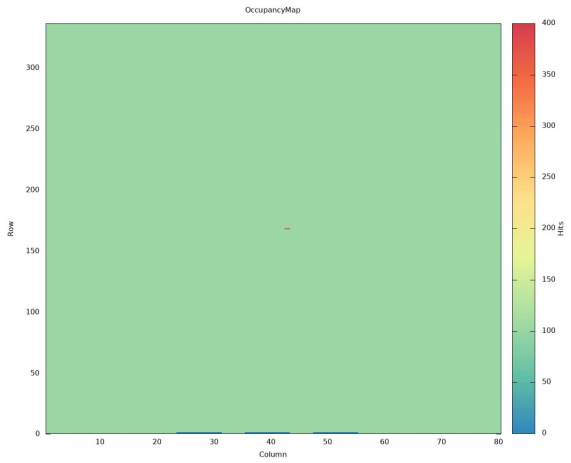
(a) chipID3，EnMask



(b) chipID3，OccupancyMap

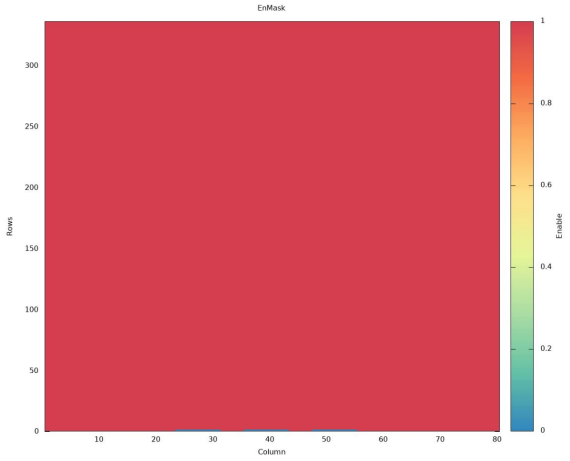


(c) chipID4，EnMask

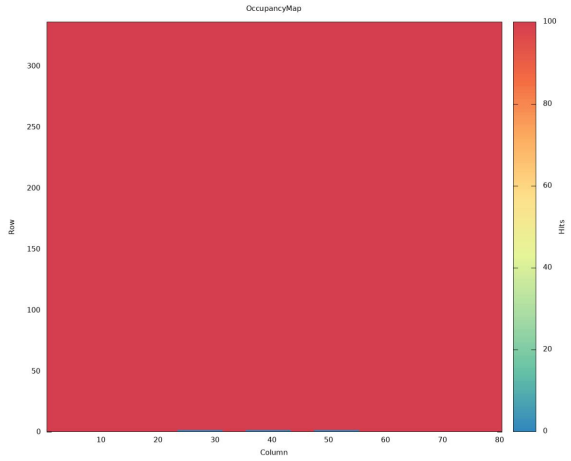


(d) chipID4，OccumapcyMap

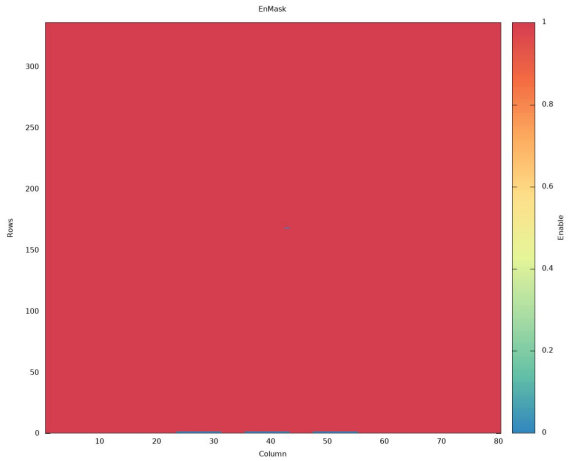
図 36: ピンソケット 1 個でのスキャン結果



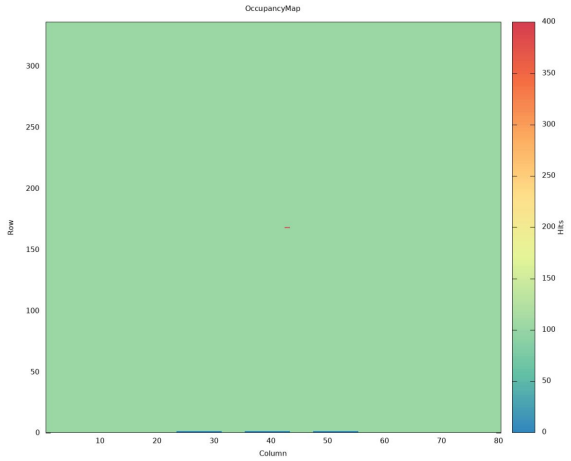
(a) chipID3，EnMask



(b) chipID3，OccupancyMap



(c) chipID4，EnMask



(d) chipID4，OccumapcyMap

図 37: ピンソケット 2 個でのスキャン結果

3.2.2　ファームウェア

これまで YARR で 4 チップ読み出しが行われていなかったことから，ファームウェアもそれに対応したものが無かった。そのため「Vivado」という Xilinx FPGA 向けの統合開発環境を用いて，ファームウェアを作成した。

Vivado は FPGA の回路データを開発するための専用ツールであり，YARR ファームウェアの開発で用いられている。Vivado は，FPGA の動作や構造，回路デザインが記述された回路記述[*14]と，回路記述のポートと FPGA の外部端子の割り当てや各端子の信号レベルなどを決定する制約ファイルから bit ファイルという FPGA のコンフィギュレーション用のデータを生成する。コンフィギュレーション用のデータを生成することをコンパイルと呼ぶ。図 38 にコンパイルの流れを示す。[6]

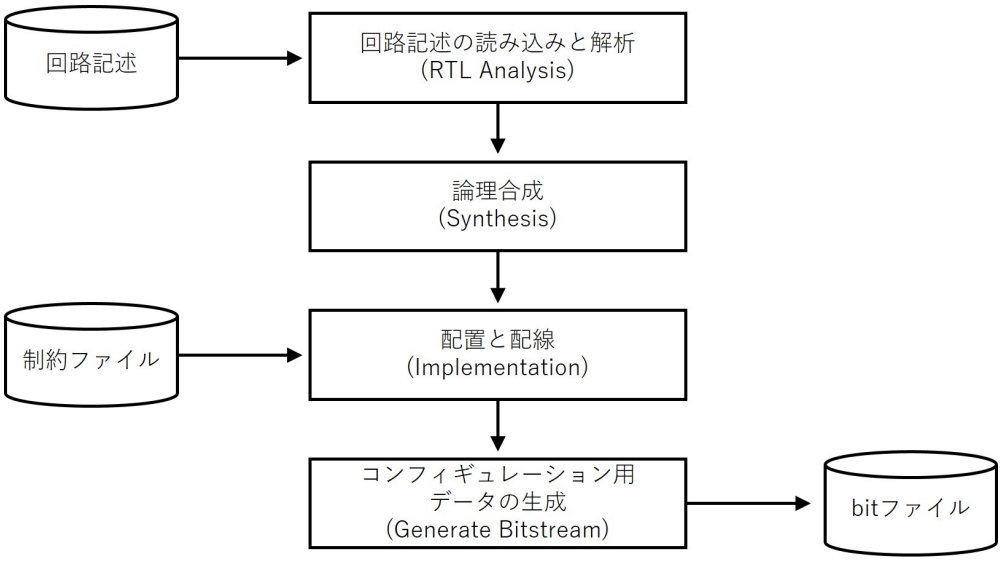


図 38: bit ファイル生成のコンパイル流れ

コンパイルにおける各処理工程について以下に記す。

RLT Analysis

回路記述ファイルを読み込み，解析するとともに分布のエラーをチェックする。

Synthesis

回路記述の内容から回路素子の接続情報を作り出す。この工程を論理合成と呼ぶ。

Implementation

読み込んだ制約ファイルに基づいて FPGA 内の回路や I/O の配線を行う。

[*14] HDL(Hardware Description Language) という言語を用いて記述されており，HDL は Verilog と VHDL の 2 種類が主に使われている。

Generate Bitstream

コンフィギュレーション用ファイルの bit ファイルを生成する。

YARR では bit ファイルそのものを提供しており，さらに bit ファイルを生成するための回路記述と制約ファイルも備えてある。今回 4 チップ読み出し用のファームウェアを作成するにあたって，回路記述は元から用意されていたものをそのまま利用し，制約ファイルを書き換えることで FPGA の外部端子を 4 チップ読み出し用に設定した。

KC705 からの入出力は FMC コネクタで行うため，FMC コネクタのどのピンをインターフェースとして設定するかを制約ファイルに記述する。TB-FMCL-PH メザニンカードのどのピンヘッダから信号が送られているかを図 40 に示す。これは図 39 のように実際にインターフェースカードと TB-FMCL-PH メザニンカードを接続して，どのピンヘッダから信号が送られているのか確認した。

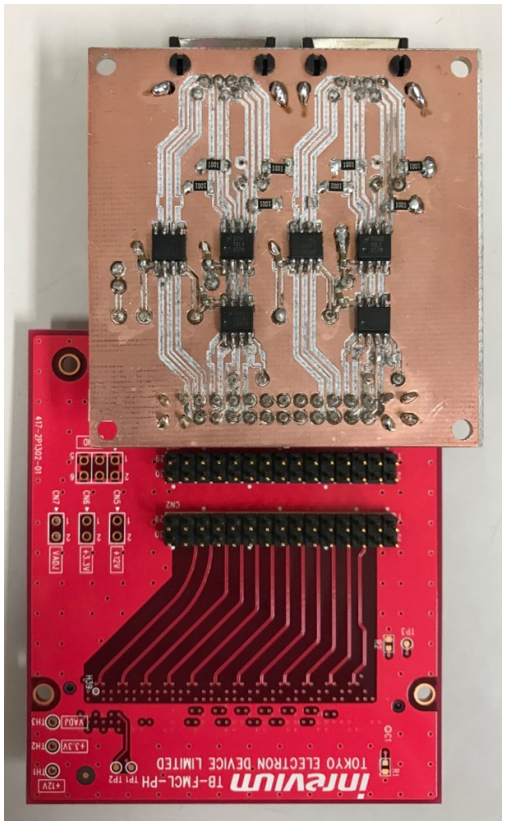


図 39: インターフェースカード接続写真

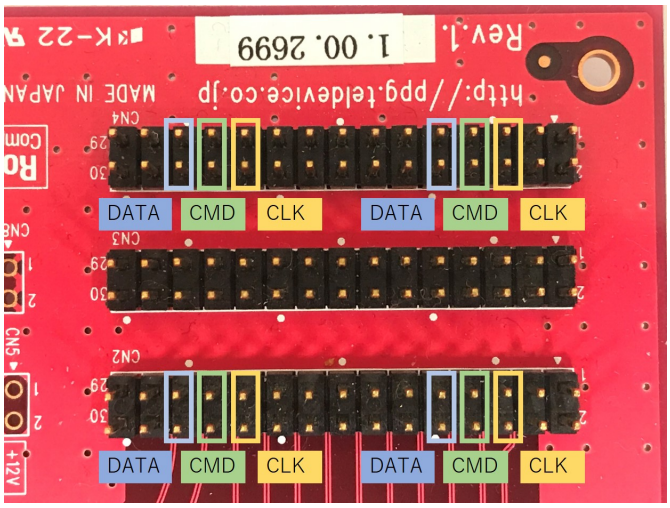


図 40: TB-FMCL-PH メザニンカードピンヘッダの信号配置図。CLK はクロック，CMD はコマンド DATA はデータの入出力を行う。

どのピンヘッダが FMC コネクタのピンにつながっているかは TB-FMCL-PH ハードユーザマニュアル [7] にある PinHeader 配置表からわかる。図
reffig:pincn2，42 に CN2 と CN4 の配置表を示す。

FMC Pin	Signal Name	PinHeader(CN2)		Signal Name	FMC Pin
-	GND	1	2	GND	-
H5	CLK0_M2C_N	3	4	CLK0_M2C_P	H4
H8	LA02_N	5	6	LA02_P	H7
H11	LA04_N	7	8	LA04_P	H10
H14	LA07_N	9	10	LA07_P	H13
H17	LA11_N	11	12	LA11_P	H16
H20	LA15_N	13	14	LA15_P	H19
H23	LA19_N	15	16	LA19_P	H22
H26	LA21_N	17	18	LA21_P	H25
H29	LA24_N	19	20	LA24_P	H28
H32	LA28_N	21	22	LA28_P	H31
H35	LA30_N	23	24	LA30_P	H34
H38	LA32_N	25	26	LA32_P	H37
-	GND	27	28	GND	-
-	NC	29	30	NC	-

図 41: PinHeader 配置表 (CN2)

FMC Pin	Signal Name	PinHeader(CN4)		Signal Name	FMC Pin
-	GND	1	2	GND	-
D9	LA01_N_CC	3	4	LA01_P_CC	D8
D12	LA05_N	5	6	LA05_P	D11
C11	LA06_N	7	8	LA06_P	C10
D15	LA09_N	9	10	LA09_P	D14
C15	LA10_N	11	12	LA10_P	C14
D18	LA13_N	13	14	LA13_P	D17
C19	LA14_N	15	16	LA14_P	C18
D21	LA17_N_CC	17	18	LA17_P_CC	D20
C23	LA18_N_CC	19	20	LA18_P_CC	C22
D24	LA23_N	21	22	LA23_P	D23
D27	LA26_N	23	24	LA26_P	D26
C27	LA27_N	25	26	LA27_P	C26
-	GND	27	28	GND	-
-	NC	29	30	NC	-

図 42: PinHeader 配置表 (CN4)

この配置表から FMC コネクタのピンの名前がわかるのだが，制約ファイルにピンの配置を記述するときは各ピンに割り当てられらピン番号を用いる必要がある。このピン番号は KC705 Evaluation Board for the Kintex-7 FPGA User Guide[3] に記載されており，これを参考に制約ファイルを作成した。作成した制約ファイルを Listing1[*15] この制約ファイルを用いてコンパイルを行うことで，4 チップ読み出しに対応したファームウェアを生成した。

Listing 1: constrs_kc705_fmccard-fei4b-kyokyo-quad.xdc

```
1  ##############################################################################################
2  ## Target Device : Kintex-7 KC705 Evaluation Platform
3  ## Yarr-fw with the custom kyokyo quad FEI4 adapter
4  ## fe_clk_x[0],fe_cmd_x[0],fe_data_x[0] are only available (others are connected but
     meaningless)
5  ##############################################################################################
6
7  # FMC LPC LA
8  ######
9
10 set_property IOSTANDARD LVDS_25 [get_ports fe_clk_*]
11 set_property IOSTANDARD LVDS_25 [get_ports fe_cmd_*]
12 set_property IOSTANDARD LVDS_25 [get_ports fe_data_*]
13
14 #CLK
15 ###CN2
16 #FMC_HPC_LA02_P
17 set_property PACKAGE_PIN H24 [get_ports {fe_clk_p[7]}]
18 #FMC_HPC_LA02_N
19 set_property PACKAGE_PIN H25 [get_ports {fe_clk_n[6]}]
20 #FMC_HPC_LA28_P
21 set_property PACKAGE_PIN D16 [get_ports {fe_clk_p[5]}]
22 #FMC_HPC_LA28_N
23 set_property PACKAGE_PIN C16 [get_ports {fe_clk_n[4]}]
24
25 ###CN4
26 #FMC_HPC_LA05_P
27 set_property PACKAGE_PIN G29 [get_ports {fe_clk_p[3]}]
28 #FMC_HPC_LA05_N
29 set_property PACKAGE_PIN F30 [get_ports {fe_clk_n[2]}]
30 #FMC_HPC_LA23_P
```

[*15] Program Listing のことである。

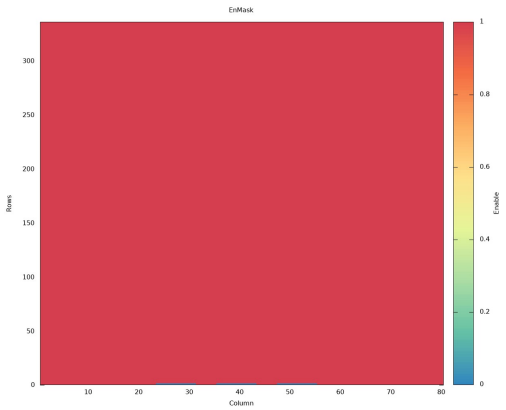
```
set_property PACKAGE_PIN B22 [get_ports {fe_clk_p[1]}]
#FMC_HPC_LA23_N
set_property PACKAGE_PIN A22 [get_ports {fe_clk_n[0]}]

#CMD
###CN2
#FMC_HPC_LA04_P
set_property PACKAGE_PIN G28 [get_ports {fe_cmd_p[7]}]
#FMC_HPC_LA04_N
set_property PACKAGE_PIN F28 [get_ports {fe_cmd_n[6]}]
#FMC_HPC_LA30_P
set_property PACKAGE_PIN D22 [get_ports {fe_cmd_p[5]}]
#FMC_HPC_LA30_N
set_property PACKAGE_PIN C22 [get_ports {fe_cmd_n[4]}]

###CN4
#FMC_HPC_LA06_P
set_property PACKAGE_PIN H30 [get_ports {fe_cmd_p[3]}]
#FMC_HPC_LA06_N
set_property PACKAGE_PIN G30 [get_ports {fe_cmd_n[2]}]
#FMC_HPC_LA26_P
set_property PACKAGE_PIN B18 [get_ports {fe_cmd_p[1]}]
#FMC_HPC_LA26_N
set_property PACKAGE_PIN A18 [get_ports {fe_cmd_n[0]}]

#DAT
###CN2
#FMC_HPC_LA07_P
set_property PACKAGE_PIN E28 [get_ports {fe_data_p[7]}]
#FMC_HPC_LA07_N
set_property PACKAGE_PIN D28 [get_ports {fe_data_n[6]}]
#FMC_HPC_LA32_P
set_property PACKAGE_PIN D21 [get_ports {fe_data_p[5]}]
#FMC_HPC_LA32_N
set_property PACKAGE_PIN C21 [get_ports {fe_data_n[4]}]

###CN4
#FMC_HPC_LA09_P
set_property PACKAGE_PIN B30 [get_ports {fe_data_p[3]}]
#FMC_HPC_LA09_N
set_property PACKAGE_PIN A30 [get_ports {fe_data_n[2]}]
#FMC_HPC_LA27_P
set_property PACKAGE_PIN C19 [get_ports {fe_data_p[1]}]
#FMC_HPC_LA27_N
set_property PACKAGE_PIN B19 [get_ports {fe_data_n[0]}]


##Other properties

###
set_property PULLUP true [get_ports {fe_clk_p[7]}]
set_property PULLUP true [get_ports {fe_clk_n[6]}]
set_property PULLUP true [get_ports {fe_clk_p[5]}]
set_property PULLUP true [get_ports {fe_clk_n[4]}]
set_property PULLUP true [get_ports {fe_clk_p[3]}]
set_property PULLUP true [get_ports {fe_clk_n[2]}]
set_property PULLUP true [get_ports {fe_clk_p[1]}]
set_property PULLUP true [get_ports {fe_clk_n[0]}]
set_property PULLUP true [get_ports {fe_cmd_p[7]}]
set_property PULLUP true [get_ports {fe_cmd_n[6]}]
set_property PULLUP true [get_ports {fe_cmd_p[5]}]
set_property PULLUP true [get_ports {fe_cmd_n[4]}]
set_property PULLUP true [get_ports {fe_cmd_p[3]}]
set_property PULLUP true [get_ports {fe_cmd_n[2]}]
set_property PULLUP true [get_ports {fe_cmd_p[1]}]
set_property PULLUP true [get_ports {fe_cmd_n[0]}]
set_property PULLUP true [get_ports {fe_data_p[7]}]
set_property PULLUP true [get_ports {fe_data_n[6]}]
set_property PULLUP true [get_ports {fe_data_p[5]}]
set_property PULLUP true [get_ports {fe_data_n[4]}]
set_property PULLUP true [get_ports {fe_data_p[3]}]
set_property PULLUP true [get_ports {fe_data_n[2]}]
set_property PULLUP true [get_ports {fe_data_p[1]}]
```

```
104 set_property PULLUP true [get_ports {fe_data_n[0]}]
```

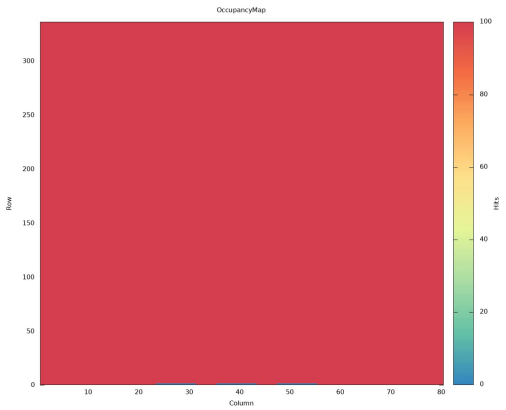
3.3 4 チップ読み出し結果

4 チップ読み出しを実行した結果を図 43 に示す。スキャンはデジタルスキャンを実行し，KEK101 ベアモジュールを用いた。KEK101 に実装されている chipID1 のチップは故障しており，読み出すことができないためスキャンできていない。

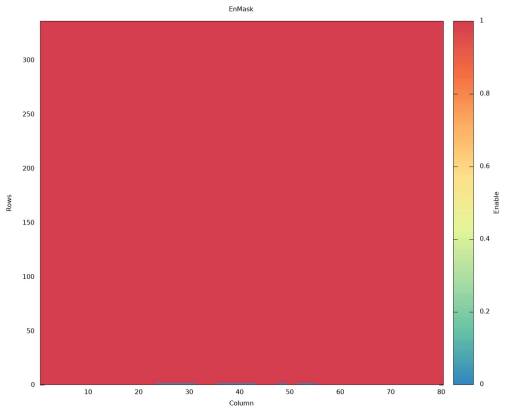
図 43f の色が他と異なるのは，図 35，36，37 と同じで一部ピクセルで 100 を超える Hit が検出されているからである。



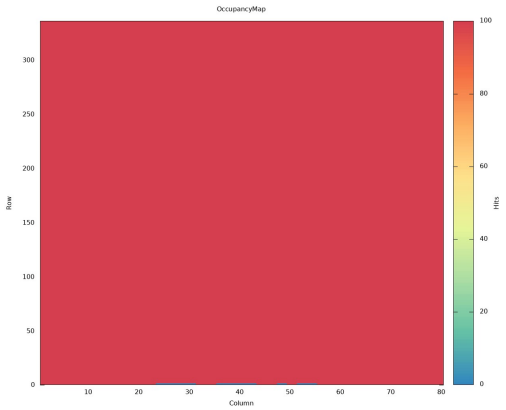
(a) chipID2，EnMask



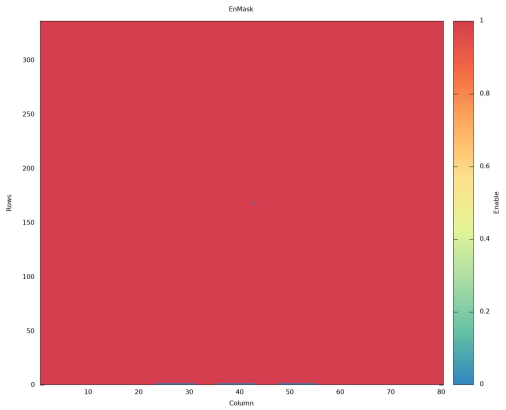
(b) chipID2，OccupancyMap



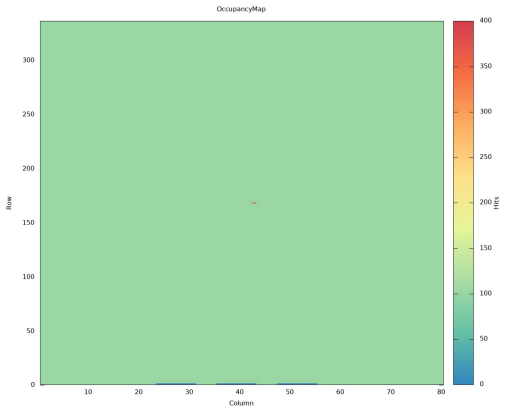
(c) chipID3，EnMask



(d) chipID3，OccumapcyMap



(e) chipID4，EnMask



(f) chipID4，OccumapcyMap

図 43: デジタルスキャンによる 4 チップ読み出し (chipID1 のチップは故障により読み出し不可)

3.4　4 チップ読み出しシステム製作についてのまとめと課題

以上に述べたように，2 個のインターフェースカード製作し，それに合わせたファームウェアを作成することで 4 チップ読み出しシステムを開発したが，KEK101 のチップが故障により 3 つしか動作していなかったため 3 チップの読み出ししかできていない。だが，2 個のインターフェースカードで同時に読み出しが行えていることやインターフェースカード上の回路は同じものであることから，読み出しシステムとしては 4 チップ読み出しが可能であると考えられる。

今後の課題としては 4 チップすべての読み出しが可能なベアモジュールを用いて読み出しを行うことが挙げられるが，今現在そのようなベアモジュールを入手できる目途が立っていない。

また，現在インターフェースカードの配置は図 30 のようになっているが，インターフェースカードの高さを出すために使っている基板連結用ピンヘッダが目的外の利用のため，ソケットの長さが足りずピンがむき出しになっている。これによりピンを固定する部分が短く強度的に不安が残るので，現段階では以下のような解決策を考えている。

- ソケット部が長いピンソケットを新たに探す。
- はんだ付けにより，ピン同士を完全に固定する。
- 接着剤やホットボンドを用いて固定する。

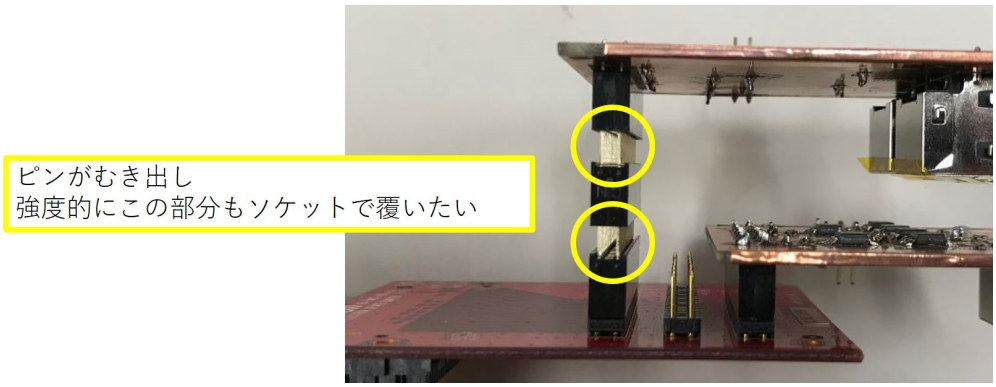


図 44: 基板連結用ピンソケットを用いる問題点

4 YARR BDAQ Converter

現在 ASIC の読み出しシステムにおいて用いられているデータ読み出しソフトウェアは YARR の他に BDAQ53[10] がある。YARR はローレンスバークレー国立研究所 (Lawrence Berkeley National Laboratory:LBL)，BDAQ53 はドイツのボン大学 (University of Bonn) でそれぞれ独立して開発されており，新型 ASIC の RD53 を実装する際にこの 2 つの読み出しソフトウェアのどちらを採用するか検討がなされている。

どちらもスキャン実行時に ASIC のチップやピクセルに対してコンフィギュレーションという設定をするのだが，それぞれが独立に開発されたために同一条件での読み出しができなかった。そこで，コンフィギュレーションの条件を合わせるために，コンフィギュレーションファイルというスキャン実行時に読み出され，コンフィギュレーションの条件を設定するファイルを YARR のフォーマットから BDAQ53 のフォーマットへ変換するコンバータスクリプトを作成した。

4.1 BDAQ53

BDAQ53 は新型 ASIC，RD53A 専用の読み出しソフトウェアであり，ボン大学で開発されていることから，Bonn Data AcQuisition の文字をとって BDAQ と呼ばれている。BDAQ53 は YARR と同様にソフトウェアとファームウェアの両方を提供し，複数枚の FPGA ボードに対応している。BDAQ53 はファームウェアを除くスキャン実行部分の多くが Python[*16]で動作しており，Python3 に対応している。

BDAQ53 による DAQ システムは，YARR と同様に KC705 FPGA ボードを用いることができるが，BDAQ53 は KC705 とコンピュータの通信に Ethernet ケーブルを用いる。また，RD53A が Displayport[*17]コネクタを採用していることから，DAQ システムと RD53A 間は SMA-Displayport 規格で接続される。

4.2 レジスタ

ASIC にはチップの動作設定を保存する記憶回路が備わっており，それをレジスタと呼ぶ。RD53A にはグローバルレジスタとピクセルレジスタの 2 種類のレジスタが備わっている。グローバルレジスタはチップ全体の制御を設定するレジスタであり，ピクセルレジスタはチップを構成するピクセルの制御を設定するレジスタである。

YARR，BDAQ53 どちらの読み出しソフトウェアにもレジスタにチップ動作の設定を書き込む

[*16] Python とはオブジェクト指向言語の 1 つであり，シンプルなコードで記述できることから可読性に優れている。2008 年に Python3.0 が公開され，Python2.x の最後のバージョンである Python2.7 は 2020 年 1 月 1 日にサポートが終了する。

[*17] ディスプレイ等への映像出力を前提に設計されたインターフェース規格。二次的な利用で汎用データなども転送できる。

ためのコンフィギュレーションファイルが含まれているが，BDAQ53 のコンフィギュレーションファイルにはピクセルレジスタについての記述がない。そのためグローバルレジスタについてのコンバータを作成した。表 4 に RD53A のグローバルレジスタの一覧を示す。

グローバルレジスタは合計で 137 個あり，それぞれ名前とアドレスが割り振られている。また，最大 16 ビットの情報を記憶することができる。表 4 の”Bit Map” に各グローバルレジスタのビット長を，英数字の 0∼9 と A∼F の 16 進表示を用いて表す。例えば”PIX_PORTAL” であれば Bit Map の値は [F:8,7:0] とある。これは，16 ビットレジスタを下位の 0 番目のビットから 7 番までを使った 8 ビットの変数と 8 番目のビットから 15 番目を使った 8 ビットの変数に分割して使用しているという意味である。RD53A には SELF_TRIG_EN は実装されていない。

Address	Name	Bit Map
0	PIX_PORTAL	[F:8,7:0]
1	REGION_COL	[7:0]
2	REGION_ROW	[8:0]
3	PIX_MODE,B_MASK	[5:3,2:0]
4	PIX_DEFAULT_CONFIG	[F:0]
5	IBIASP1_SYNC	[8:0]
6	IBIASP2_SYNC	[8:0]
7	IBIAS_SF_SYNC	[8:0]
8	IBIAS_KRUM_SYNC	[8:0]
9	IBIAS_DISC_SYNC	[8:0]
10	ICTRL_SYNCT_SYNC	[9:0]
11	VBL_SYNC	[9:0]
12	VTH_SYNC	[9:0]
13	VREF_KRUM_SYNC	[9:0]
30	CONF_FE_SYNC	[4:0]
14	PA_IN_BIAS_LIN	[8:0]
15	FC_BIAS_LIN	[7:0]
16	KRUM_CURR_LIN	[8:0]
17	LDAC_LIN	[9:0]
18	COMP_LIN	[8:0]
19	REF_KRUM_LIN	[9:0]
20	Vthreshold_LIN	[9:0]
21	PRMP_DIFF	[9:0]
22	FOL_DIFF	[9:0]
23	PRECOMP_DIFF	[9:0]
24	COMP_DIFF	[9:0]
25	VFF_DIFF	[9:0]
26	VTH1_DIFF	[9:0]
27	VTH2_DIFF	[9:0]
28	LCC_DIFF	[9:0]

次ページに続く

前ページからの続き

Address	Name	Bit Map
29	CONF_FE_DIFF	[1:0]
31	SLDO_ANALOG_TRIM	[9:5]
31	SLDO_DIGITAL_TRIM	[4:0]
32	EN_CORE_COL_SYNC	[F:0]
33	EN_CORE_COL_LIN_1	[F:0]
34	EN_CORE_COL_LIN_2	[0]
35	EN_CORE_COL_DIFF_1	[F:0]
36	EN_CORE_COL_DIFF_2	[0]
37	LATENCY_CONFIG	[8:0]
38	WR_SYNC_DELAY_SYNC	[4:0]
39	INJ_MODE_DEL	[5,4,3:0]
41	VCAL_HIGH	[B:0]
42	VCAL_MED	[B:0]
46-49	CAL_COLPR_SYNC	4×[F:0]
50-53	CAL_COLPR_LIN	4×[F:0]
54	CAL_COLPR_LIN5	[3:0]
55-58	CAL_COLPR_DIFF	4×[F:0]
59	CAL_COLPR_DIFF5	[3:0]
40	CLK_DATA_DELAY	[8,7:4,3:0]
43	CH_SYNC_CONF	[11:10,9:5,4:0]
44	GLOBAL_PULSE_RT	[F:0]
60	DEBUG_CONFIG	[1:0]
61	OUTPUT_CONFIG	[8:7,6,5:2,1:0]
62	OUT_PAD_CONFIG	[D:0]
63	GP_LVDS_ROUTE	[2:0]
64	CDR_CONFIG	[D:0]
65	VCO_BUFF_BIAS	[9:0]
66	CDR_CP_IBIAS	[9:0]
67	VCO_IBIAS	[9:0]
68	SER_SEL_OUT	[7:6,5:4,3:2,1:0]
69	CML_CONFIG	[7:6,5:4]
70-72	CML_TAP_BIAS	3×[9:0]
73	AURORA_CC_CFG	[7:2,1:0]
74	AURORA_CB_CFG0	[7:4,3:0]
75	AURORA_CB_CFG1	[F:0]
76	AURORA_INIT_WAIT	[A:0]
45	MON_FRAME_SKIP	[7:0]
101-102	AUTO_READ_A0,B0	2×[8:0]
103-104	AUTO_READ_A1,B1	2×[8:0]
105-106	AUTO_READ_A2,B2	2×[8:0]
107-108	AUTO_READ_A3,B3	2×[8:0]

次ページに続く

前ページからの続き

Address	Name	Bit Map
77	MONITOR_MUX	Monitor
78-81	HITOR_MASK_SYNC	4×[F:0]
82,84,86,88	HITOR_MASK_LIN1	4×[F:0]
83,85,87,89	HITOR_MASK_LIN2	4×[0]
90,92,94,96	HITOR_MASK_DIFF1	4×[F:0]
91,93,95,97	HITOR_MASK_DIFF2	4×[0]
98	ADC_CONFIG	[A:6,0:5]
99-100	SENSOR_CONFIG	2×[B:0]
109	RING_OSC_ENABLE	[7:0]
110-117	RING_OSC	8×[F:0]
118	BC_CTR	[F:0]
119	TRIG_CTR	[F:0]
120	LCK_LOSS_CTR	[F:0]
121	BFLIP_WARN_CTR	[F:0]
122	BFLIP_ERR_CTR	[F:0]
123	CMD_ERR_CTR	CMD
124-127	FIFO_FULL_CTR	4×[F:8,7:0]
128	AI_PIX_COL	[7:0]
129	AI_PIX_ROW	[8:0]
130-133	HitOr_Cnt	4×[F:0]
134	SKP_TRIG_CNT	[F:0]
135	ERR_MASK	[D:0]
136	ADC_READ	[B:0]
137	SELF_TRIG_EN	

表 4: グローバルレジスタ一覧

以上

4.3 コンフィギュレーションファイル

YARR と BDAQ53 の両方にコンフィギュレーションファイルが用意されている。コンフィギュレーションファイルはスキャン実行時に読み込まれ，記述されたコンフィグパラメータからグローバルレジスタのコンフィギュレーション（設定）が行われる。

4.3.1 YARR コンフィギュレーションファイル

YARR のコンフィギュレーションファイルは default_chip.json という名前であり，json フォーマットで記述されている。json とは (JavaScript Object Notaion) という JavaScript のデータ定義文をベースとした，簡易的なデータ形式である。Listing2 に default_chip.json のグローバルレ

ジスタのコンフィギュレーションに関わる部分を示す。

default_chip.json では { } で囲まれた中にコロンで区切られた文字列と数字のペアが並んでいる。{ } で囲まれた範囲はオブジェクトと呼ばれ，データが入る場所を指す。オブジェクト内のコロンで区切られた文字列と数字はそれぞれ key と value と呼ばれ，value の値がグローバルレジスタのコンフィギュレーションに関わるパラメータであり，コンフィグパラメータと呼ばれている。また，コンフィグパラメータの key と value が記述されているオブジェクトは，”GlobalConfig”の value であり，”GlobalConfig”とその value は”RD53A”の value である。このように key と value のネスト（入れ子）構造が用いられている。

コンフィグパラメータの数を見ると RD53A のグローバルレジスタの数と比べて多くなっていることが分かる。これはグローバルレジスタの PIX_PORTAL のような複数に分かれた Bit Map に対し，1 つずつコンフィグパラメータを用意しているからである。

Listing 2: default_chip.json

```
{
    "RD53A": {
        "GlobalConfig": {
            "AdcRead": 0,
            "AdcRefTrim": 12,
            "AdcTrim": 5,
            "AiPixCol": 0,
            "AiPixRow": 0,
            "AuroraCbSend": 0,
            "AuroraCbWaitHigh": 15,
            "AuroraCbWaitLow": 15,
            "AuroraCcSend": 3,
            "AuroraCcWait": 25,
            "AuroraInitWait": 32,
            "AutoReadA0": 136,
            "AutoReadA1": 118,
            "AutoReadA2": 120,
            "AutoReadA3": 122,
            "AutoReadB0": 130,
            "AutoReadB1": 119,
            "AutoReadB2": 121,
            "AutoReadB3": 123,
            "BcCounter": 0,
            "BflipErrCounter": 0,
            "BflipWarnCounter": 0,
            "CalColprDiff1": 65535,
            "CalColprDiff2": 65535,
            "CalColprDiff3": 65535,
            "CalColprDiff4": 65535,
            "CalColprDiff5": 15,
            "CalColprLin1": 65535,
            "CalColprLin2": 65535,
            "CalColprLin3": 65535,
            "CalColprLin4": 65535,
            "CalColprLin5": 15,
            "CalColprSync1": 65535,
            "CalColprSync2": 65535,
            "CalColprSync3": 65535,
            "CalColprSync4": 65535,
            "CdrCpIbias": 50,
            "CdrEnGck": 0,
            "CdrPdDel": 8,
            "CdrPdSel": 0,
            "CdrSelDelClk": 0,
            "CdrSelSerClk": 3,
            "CdrVcoGain": 3,
            "ChSyncLock": 16,
```

```
            "ChSyncPhase": 0,
            "ChSyncUnlock": 8,
            "ClkDelay": 0,
            "ClkDelaySel": 0,
            "CmdDelay": 0,
            "CmdErrCounter": 0,
            "CmlEn": 15,
            "CmlEnTap": 1,
            "CmlInvTap": 0,
            "CmlTapBias0": 600,
            "CmlTapBias1": 0,
            "CmlTapBias2": 0,
            "DebugConfig": 0,
            "DiffComp": 1000,
            "DiffFbCapEn": 0,
            "DiffFol": 500,
            "DiffLcc": 20,
            "DiffLccEn": 0,
            "DiffPrecomp": 400,
            "DiffPrmp": 500,
            "DiffVff": 50,
            "DiffVth1": 250,
            "DiffVth2": 50,
            "EnCoreColDiff1": 65535,
            "EnCoreColDiff2": 1,
            "EnCoreColLin1": 65535,
            "EnCoreColLin2": 1,
            "EnCoreColSync": 65535,
            "ErrMask": 0,
            "FifoFullCounter0": 0,
            "FifoFullCounter1": 0,
            "FifoFullCounter2": 0,
            "FifoFullCounter3": 0,
            "GlobalPulseRt": 16640,
            "GpLvdsRoute": 0,
            "HitOr0MaskDiff0": 0,
            "HitOr0MaskDiff1": 0,
            "HitOr0MaskLin0": 0,
            "HitOr0MaskLin1": 0,
            "HitOr0MaskSync": 0,
            "HitOr1MaskDiff0": 0,
            "HitOr1MaskDiff1": 0,
            "HitOr1MaskLin0": 0,
            "HitOr1MaskLin1": 0,
            "HitOr1MaskSync": 0,
            "HitOr2MaskDiff0": 0,
            "HitOr2MaskDiff1": 0,
            "HitOr2MaskLin0": 0,
            "HitOr2MaskLin1": 0,
            "HitOr2MaskSync": 0,
            "HitOr3MaskDiff0": 0,
            "HitOr3MaskDiff1": 0,
            "HitOr3MaskLin0": 0,
            "HitOr3MaskLin1": 0,
            "HitOr3MaskSync": 0,
            "HitOrCounter0": 0,
            "HitOrCounter1": 0,
            "HitOrCounter2": 0,
            "HitOrCounter3": 0,
            "InjAnaMode": 0,
            "InjDelay": 0,
            "InjEnDig": 0,
            "InjVcalDiff": 0,
            "InjVcalHigh": 1000,
            "InjVcalMed": 1000,
            "LatencyConfig": 64,
            "LinComp": 110,
            "LinFcBias": 20,
            "LinKrumCurr": 50,
            "LinLdac": 100,
            "LinPaInBias": 300,
            "LinRefKrum": 300,
            "LinVth": 400,
```

```
            "LockLossCounter": 0,
            "MonFrameSkip": 200,
            "MonitorEnable": 0,
            "MonitorImonMux": 63,
            "MonitorVmonMux": 127,
            "OutPadConfig": 5124,
            "OutputActiveLanes": 15,
            "OutputDataReadDelay": 0,
            "OutputFmt": 0,
            "OutputSerType": 0,
            "PixAutoCol": 0,
            "PixAutoRow": 0,
            "PixBroadcastEn": 0,
            "PixBroadcastMask": 0,
            "PixDefaultConfig": 0,
            "PixPortal": 1796,
            "PixRegionCol": 199,
            "PixRegionRow": 135,
            "RingOsc0": 0,
            "RingOsc1": 0,
            "RingOsc2": 0,
            "RingOsc3": 0,
            "RingOsc4": 0,
            "RingOsc5": 0,
            "RingOsc6": 0,
            "RingOsc7": 0,
            "RingOscEn": 0,
            "SelfTrigEn": 0,
            "SensorCfg0": 0,
            "SensorCfg1": 0,
            "SerSelOut0": 1,
            "SerSelOut1": 1,
            "SerSelOut2": 1,
            "SerSelOut3": 1,
            "SkipTriggerCounter": 0,
            "SldoAnalogTrim": 22,
            "SldoDigitalTrim": 22,
            "SyncAutoZero": 0,
            "SyncFastTot": 0,
            "SyncIbiasDisc": 300,
            "SyncIbiasKrum": 55,
            "SyncIbiasSf": 80,
            "SyncIbiasp1": 80,
            "SyncIbiasp2": 120,
            "SyncIctrlSynct": 100,
            "SyncSelC2F": 0,
            "SyncSelC4F": 1,
            "SyncVbl": 400,
            "SyncVrefKrum": 450,
            "SyncVth": 300,
            "TrigCounter": 0,
            "VcoBuffBias": 400,
            "VcoIbias": 500,
            "WrSyncDelaySync": 16
        }
```

4.3.2 BDAQ53 コンフィギュレーションファイル

BDAQ53 におけるグローバルレジスタに関するコンフィギュレーションファイルは 2 種類あり，rd53a_default.cfg.yaml と rd53a_registers.yaml である。これらのファイルは yaml(Yet Another Markup Language) というフォーマットで記述されている。

yaml は可読性に優れ，json とは異なり，コメント[*18]を挿入できるという利点がある。Listing3

[*18] プログラムの動作に影響を与えずに，プログラム内に記述することができるメモ。人間のために書かれた注意書き。yaml の場合は”#” を行頭に用いることでコメント化できる。

に rd53a_default.cfg.yaml を示す。

rd53a_default.cfg.yaml はハッシュを用いて記述されている。ハッシュは default_chip.json でのコンフィグパラメータの記述と同じで，コロンを挟んだ key と value によりデータを記述する方法である。ハッシュで記述されたデータをオブジェクト指向言語でオブジェクトとして扱う場合，そのオブジェクトのデータ型は辞書型と呼ばれる。json での記法と異なり，スペースのみを用いてネスト構造を記述する。

rd53a_default.cfg.yaml において，グローバルレジスタに関する記述は 32 行目の rgisters の value にある。

Listing 3: rd53a_default.cfg.yaml

```yaml
# Default trimbit values obtained from statistical analysis.
# If you need more exact values, please trim your individual chip.
trim:
  VREF_A_TRIM       : 24
  VREF_D_TRIM       : 23
  MON_BG_TRIM       : 12
  MON_ADC_TRIM      : 5

# Default calibration values should be ok to get an order of magnitude reading.
# If you need more exact measurements, please calibrate your individual chip.
calibration:
  ADC_a             : 0.2     # Conversion factors ADC counts - voltage: slope
  ADC_b             : 15.5    # Conversion factors ADC counts - voltage: offset
  Nf                : 1.24    # Chip specific conversion factor for on-chip T sensors:
    voltage - temperature

  # Default charge calibration is based on 50x50um modules and should be ok for bare chips
    as well.
  # For optimal results, perform a charge calibration on your individual chip!
  e_conversion:
    # [slope] = Electrons / Delta VCAL
    slope           : 10.4
    slope_error     : 0.10
    # [offset] = Electrons
    offset          : 180
    offset_error    : 60

# Disable specific pixels
#disable           :
#                  - [0, 0]

# DAC settings (inner layer, bare chip, 5 uA operating point)
# For other settings, have a look at the individual FE guides.
registers:
  IBIASP1_SYNC      : 90
  IBIASP2_SYNC      : 140
  IBIAS_SF_SYNC     : 80
  IBIAS_KRUM_SYNC   : 55
  IBIAS_DISC_SYNC   : 300
  ICTRL_SYNCT_SYNC  : 100
  VBL_SYNC          : 380
  VTH_SYNC          : 390
  VREF_KRUM_SYNC    : 450
  CONF_FE_SYNC      : 0x0002 # High gain, normal ToT

  PA_IN_BIAS_LIN    : 350
  FC_BIAS_LIN       : 20
  KRUM_CURR_LIN     : 32
  LDAC_LIN          : 130
  COMP_LIN          : 110
  REF_KRUM_LIN      : 300
  Vthreshold_LIN    : 415

  PRMP_DIFF         : 511
  FOL_DIFF          : 542
  PRECOMP_DIFF      : 512
  COMP_DIFF         : 1023 # (FPM), for FEM use 528
  VFF_DIFF          : 40   # (FPM), for FEM use 140
  VTH1_DIFF         : 600
  VTH2_DIFF         : 50
  LCC_DIFF          : 20
  CONF_FE_DIFF      : 0 # Default: LCC off, if leakage current higher than 1 nA, LCC can be
    switched on.
```

Listing4 に rd53a_registers.yaml の一部を示す。全体を記載すると膨大な量となるので一部を示した。BDAQ53 のコンフィギュレーションファイルが rd53a_default.cfg.yaml と rd53a_registers.yaml の 2 種類ある理由としては，rd53a_registers.yaml ですべてのグローバルレ

ジスタの初期設定をし，rd53a_default.cfg.yaml でユーザーが設定をするコンフィグパラメータをピックアップして 1 つのファイルにまとめることで，多くのコンフィグパラメータから必要なものを探す手間をなくすためである。

yaml の記法で”-”(ハイフン) を用いると配列を表すことができる。rd53a_registers.yaml は registers が key となり，value に配列構造が記されている。その配列の要素に address から reset までのハッシュが含まれている。配列の要素 (address∼reset) のうちコンバータ作成に用いたものを以下に記す。

address	RD53A に対して割り当てられているアドレスが 16 進数で記述されている。表 4 の Address に対応している。
default	レジスタのコンフィグパラメータ。2 進数で記述されている。
description	レジスタについての説明。
name	レジスタの名前。
size	レジスタのビット長。分割して使用されている場合は合計が記述されている。

RD53A に実装されていないグローバルレジスタの SELF_TRIGGER_ENABLE は#でコメントアウトされており，それを除いた最後の要素の address は 16 進数で 0x88 となっている。これを 10 進数に直すと 136 になり，表 4 のグローバルレジスタの個数と一致することから，表 4 に示したグローバルレジスタと rd53a_registers.yaml のコンフィグパラメータは一致していることがわかる。また，name についても同じ名前が用いられているコンフィグパラメータが多いが，すべて完全に同じではなく，意味合いは同じだが違う言葉が使われている場合もある。

Listing 4: rd53a_registers.yaml

```yaml
registers:
- address: '0x0'
  default: '0b0'
  description: Virtual register to access pixel matrix
  mode: 1
  name: PIX_PORTAL
  size: 16
  reset: 0

- address: '0x1'
  default: '0b0'
  description: Region Column Address
  mode: 1
  name: REGION_COL
  size: 8
  reset: 1

- address: '0x2'
  default: '0b0'
  description: Region Row Address
  mode: 1
  name: REGION_ROW
  size: 9
  reset: 1

- address: '0x3'
  default: '0b111'
  description: 'Mode bits: Broadcast, AutoCol, AutoRow, BroadcastMask'
  mode: 1
  name: PIX_MODE
  size: 6
  reset: 0
  .
  .
  .
  .省略

  .
  .
  .
  .
- address: '0x88'
  default: '0b0'
  description: Contains the value of the ADC to be read back
  mode: 0
  name: MonitoringDataADC
  size: 12
  reset: 0

#- address: '0x89'
#  default: '0b0'
#  description: Enable Self Triggering Mode
#  mode: 1
#  name: SELF_TRIGGER_ENABLE
#  size: 8
#  reset: 1
```

4.4 コンフィグパラメータ対応表

YARR と BDAQ53 コンフィギュレーションファイルの違いをまとめると表 5 ような違いがあり，コンバータスクリプトはこれらを違いを YARR から BDAQ53 へ変換する必要がある。この中でコンフィグパラメータの数だけはコンバータスクリプトだけで対応できなかったので，YARR と BDAQ53 のコンフィグパラメータを対応付ける対応表を作成した。

	YARR	BDAQ53
ファイル名	default_chip.json	rd53a_default.cfg.yaml，rd53a_registers.yaml
ファイル数	1	2
フォーマット	json	yaml
位取り記数法	10 進数	2 進数，10 進数
コンフィグパラメータ数	171	136

表 5: YARR・BDAQ53 コンフィギュレーションファイル比較

BDAQ53		YARR	
Name	Value	Name	Value
PIX_PORTAL	0b0	PixPortal	1796
REGION_COL	0b0	PixRegionCol	199
REGION_ROW	0b0	PixRegionRow	191
PIX_MODE	0b111	PixAutoCol	0
PIX_MODE	0b111	PixAutoRow	0
PIX_MODE	0b111	PixBroadcastEn	0
PIX_MODE	0b111	PixBroadcastMask	0
PIX_DEFAULT_CONFIG	0b1001110011100010	PixDefaultConfig	0
IBIASP1_SYNC	0b1100100	SyncIbiasp1	100
IBIASP2_SYNC	0b10010110	SyncIbiasp2	150
IBIAS_SF_SYNC	0b1100100	SyncIbiasSf	100
IBIAS_KRUM_SYNC	0b10001100	SyncIbiasKrum	40
IBIAS_DISC_SYNC	0b11001000	SyncIbiasDisc	300
ICTRL_SYNCT_SYNC	0b1100100	SyncIctrlSynct	100
VBL_SYNC	0b111000010	SyncVbl	400
VTH_SYNC	0b100101100	SyncVth	300
VREF_KRUM_SYNC	0b111101010	SyncVrefKrum	450
PA_IN_BIAS_LIN	0b100101100	LinPaInBias	300
FC_BIAS_LIN	0b10100	LinFcBias	20
KRUM_CURR_LIN	0b110010	LinKrumCurr	50

次ページに続く

前ページからの続き

BDAQ53		YARR	
Name	Value	Name	Value
LDAC_LIN	0b1010000	LinLdac	100
COMP_LIN	0b1101110	LinComp	110
REF_KRUM_LIN	0b100101100	LinRefKrum	300
Vthreshold_LIN	0b110011000	LinVth	400
PRMP_DIFF	0b1000010101	DiffPrmp	500
FOL_DIFF	0b1000011110	DiffFol	500
PRECOMP_DIFF	0b1000100111	DiffPrecomp	400
COMP_DIFF	0b1000010000	DiffComp	1000
VFF_DIFF	0b10100100	DiffVff	50
VTH1_DIFF	0b1111111111	DiffVth1	250
VTH2_DIFF	0b0	DiffVth2	50
LCC_DIFF	0b10100	DiffLcc	20
CONF_FE_DIFF	0b10	DiffFbCapEn	0
CONF_FE_DIFF	0b10	DiffLccEn	0
CONF_FE_SYNC	0b100	SyncAutoZero	0
CONF_FE_SYNC	0b100	SyncSelC2F	0
CONF_FE_SYNC	0b100	SyncSelC4F	1
CONF_FE_SYNC	0b100	SyncFastTot	0
VOLTAGE_TRIM	0b1100010111	SldoAnalogTrim	26
VOLTAGE_TRIM	0b1100010111	SldoDigitalTrim	20
EN_CORE_COL_SYNC	0b0	EnCoreColSync	65535
EN_CORE_COL_LIN_1	0b0	EnCoreColLin1	65535
EN_CORE_COL_LIN_2	0b0	EnCoreColLin2	1
EN_CORE_COL_DIFF_1	0b0	EnCoreColDiff1	65535
EN_CORE_COL_DIFF_2	0b0	EnCoreColDiff2	1
LATENCY_CONFIG	0b111110100	LatencyConfig	64
WR_SYNC_DELAY_SYNC	0b10000	WrSyncDelaySync	16
INJECTION_SELECT	0b100000	InjAnaMode	0
INJECTION_SELECT	0b100000	InjDelay	0
INJECTION_SELECT	0b100000	InjEnDig	0
CLK_DATA_DELAY	0b0	ClkDelay	0
CLK_DATA_DELAY	0b0	ClkDelaySel	0
CLK_DATA_DELAY	0b0	CmdDelay	0
VCAL_HIGH	0b111110100	InjVcalHigh	1000
VCAL_MED	0b100101100	InjVcalMed	1000
CH_SYNC_CONF	0b1000001000	ChSyncLock	16
CH_SYNC_CONF	0b1000001000	ChSyncPhase	0
CH_SYNC_CONF	0b1000001000	ChSyncUnlock	8
GLOBAL_PULSE_ROUTE	0b0	GlobalPulseRt	16640
MONITOR_FRAME_SKIP	0b110010	MonFrameSkip	200

次ページに続く

前ページからの続き

BDAQ53		YARR	
Name	Value	Name	Value
EN_MACRO_COL_CAL_SYNC_1	0b1111111111111111	CalColprSync1	0
EN_MACRO_COL_CAL_SYNC_2	0b1111111111111111	CalColprSync2	15
EN_MACRO_COL_CAL_SYNC_3	0b1111111111111111	CalColprSync3	240
EN_MACRO_COL_CAL_SYNC_4	0b1111111111111111	CalColprSync4	3840
EN_MACRO_COL_CAL_LIN_1	0b1111111111111111	CalColprLin1	61440
EN_MACRO_COL_CAL_LIN_2	0b1111111111111111	CalColprLin2	0
EN_MACRO_COL_CAL_LIN_3	0b1111111111111111	CalColprLin3	15
EN_MACRO_COL_CAL_LIN_4	0b1111111111111111	CalColprLin4	240
EN_MACRO_COL_CAL_LIN_5	0b1111	CalColprLin5	0
EN_MACRO_COL_CAL_DIFF_1	0b1111111111111111	CalColprDiff1	240
EN_MACRO_COL_CAL_DIFF_2	0b1111111111111111	CalColprDiff2	3840
EN_MACRO_COL_CAL_DIFF_3	0b1111111111111111	CalColprDiff3	61440
EN_MACRO_COL_CAL_DIFF_4	0b1111111111111111	CalColprDiff4	0
EN_MACRO_COL_CAL_DIFF_5	0b1111	CalColprDiff5	15
DEBUG_CONFIG	0b0	DebugConfig	0
OUTPUT_CONFIG	0b100	OutputActiveLanes	15
OUTPUT_CONFIG	0b100	OutputDataReadDelay	0
OUTPUT_CONFIG	0b100	OutputFmt	0
OUTPUT_CONFIG	0b100	OutputSerType	0
OUT_PAD_CONFIG	0b1010000000100	OutPadConfig	5124
GP_LVDS_ROUTE	0b0	GpLvdsRoute	0
CDR_CONFIG	0b10000011000	CdrEnGck	0
CDR_CONFIG	0b10000011000	CdrPdDel	8
CDR_CONFIG	0b10000011000	CdrPdSel	0
CDR_CONFIG	0b10000011000	CdrSelDelClk	0
CDR_CONFIG	0b10000011000	CdrSelSerClk	3
CDR_CONFIG	0b10000011000	CdrVcoGain	3
CDR_VCO_BUFF_BIAS	0b110010000	VcoBuffBias	400
CDR_CP_IBIAS	0b110010	CdrCpIbias	50
CDR_VCO_IBIAS	0b111110100	VcoIbias	500
SER_SEL_OUT	0b1010101	SerSelOut0	1
SER_SEL_OUT	0b1010101	SerSelOut1	1
SER_SEL_OUT	0b1010101	SerSelOut2	1
SER_SEL_OUT	0b1010101	SerSelOut3	1
CML_CONFIG	0b111111	CmlEn	15
CML_CONFIG	0b111111	CmlEnTap	1
CML_CONFIG	0b111111	CmlInvTap	0
CML_TAP0_BIAS	0b111110100	CmlTapBias0	600
CML_TAP1_BIAS	0b0	CmlTapBias1	0
CML_TAP2_BIAS	0b0	CmlTapBias2	0

次ページに続く

前ページからの続き

BDAQ53		YARR	
Name	Value	Name	Value
AURORA_CC_CONFIG	0b1100111	AuroraCcSend	3
AURORA_CC_CONFIG	0b1100111	AuroraCcWait	25
AURORA_CB_CONFIG0	0b11110000	AuroraCbSend	0
AURORA_CB_CONFIG0	0b11110000	AuroraCbWaitLow	15
AURORA_CB_CONFIG1	0b1111	AuroraCbWaitHigh	15
AURORA_INIT_WAIT	0b100000	AuroraInitWait	32
MONITOR_SELECT	0b1111111111111	MonitorImonMux	63
MONITOR_SELECT	0b1111111111111	MonitorVmonMux	127
MONITOR_SELECT	0b1111111111111	MonitorEnable	0
HITOR_0_MASK_SYNC	0b0	HitOr0MaskSync	0
HITOR_1_MASK_SYNC	0b0	HitOr1MaskSync	0
HITOR_2_MASK_SYNC	0b0	HitOr2MaskSync	0
HITOR_3_MASK_SYNC	0b0	HitOr3MaskSync	0
HITOR_0_MASK_LIN_0	0b0	HitOr0MaskLin0	0
HITOR_0_MASK_LIN_1	0b0	HitOr0MaskLin1	0
HITOR_1_MASK_LIN_0	0b0	HitOr1MaskLin0	0
HITOR_1_MASK_LIN_1	0b0	HitOr1MaskLin1	0
HITOR_2_MASK_LIN_0	0b0	HitOr2MaskLin0	0
HITOR_2_MASK_LIN_1	0b0	HitOr2MaskLin1	0
HITOR_3_MASK_LIN_0	0b0	HitOr3MaskLin0	0
HITOR_3_MASK_LIN_1	0b0	HitOr3MaskLin1	0
HITOR_0_MASK_DIFF_0	0b0	HitOr0MaskDiff0	0
HITOR_0_MASK_DIFF_1	0b0	HitOr0MaskDiff1	0
HITOR_1_MASK_DIFF_0	0b0	HitOr1MaskDiff0	0
HITOR_1_MASK_DIFF_1	0b0	HitOr1MaskDiff1	0
HITOR_2_MASK_DIFF_0	0b0	HitOr2MaskDiff0	0
HITOR_2_MASK_DIFF_1	0b0	HitOr2MaskDiff1	0
HITOR_3_MASK_DIFF_0	0b0	HitOr3MaskDiff0	0
HITOR_3_MASK_DIFF_1	0b0	HitOr3MaskDiff1	0
MONITOR_CONFIG	0b01100000101	AdcRefTrim	0
MONITOR_CONFIG	0b01100000101	AdcTrim	0
SENSOR_CONFIG_0	0b0	SensorCfg0	0
SENSOR_CONFIG_1	0b0	SensorCfg1	0
AutoRead0	0b10001000	AutoReadA0	136
AutoRead1	0b10000010	AutoReadA1	118
AutoRead2	0b1110110	AutoReadA2	120
AutoRead3	0b1110111	AutoReadA3	122
AutoRead4	0b1111000	AutoReadB0	130
AutoRead5	0b1111001	AutoReadB1	119
AutoRead6	0b1111010	AutoReadB2	121

次ページに続く

前ページからの続き

BDAQ53		YARR	
Name	Value	Name	Value
AutoRead7	0b1111011	AutoReadB3	123
RING_OSC_ENABLE	0b0	RingOscEn	0
RING_OSC_0	0b0	RingOsc0	0
RING_OSC_1	0b0	RingOsc1	0
RING_OSC_2	0b0	RingOsc2	0
RING_OSC_3	0b0	RingOsc3	0
RING_OSC_4	0b0	RingOsc4	0
RING_OSC_5	0b0	RingOsc5	0
RING_OSC_6	0b0	RingOsc6	0
RING_OSC_7	0b0	RingOsc7	0
BCIDCnt	0b0	BcCounter	0
TrigCnt	0b0	TrigCounter	0
LockLossCnt	0b0	LockLossCounter	0
BitFlipWngCnt	0b0	BflipWarnCounter	0
BitFlipErrCnt	0b0	BflipErrCounter	0
CmdErrCnt	0b0	CmdErrCounter	0
WngFifoFullCnt_0	0b0	FifoFullCounter0	0
WngFifoFullCnt_1	0b0	FifoFullCounter1	0
WngFifoFullCnt_2	0b0	FifoFullCounter2	0
WngFifoFullCnt_3	0b0	FifoFullCounter3	0
AI_REGION_COL	0b0	AiPixCol	0
AI_REGION_ROW	0b0	AiPixRow	0
HitOr_0_Cnt	0b0	HitOrCounter0	0
HitOr_1_Cnt	0b0	HitOrCounter1	0
HitOr_2_Cnt	0b0	HitOrCounter2	0
HitOr_3_Cnt	0b0	HitOrCounter3	0
SkippedTriggerCnt	0b0	SkipTriggerCounter	0
ErrWngMask	0b0	ErrMask	0
MonitoringDataADC	0b0	AdcRead	0
SELF_TRIGGER_ENABLE	0b0	SelfTrigEn	0

表 6: 対応表

以上

表 6 の対応表は YARR コンフィギュレーションファイルの複数に分割されたコンフィグパラメータを 1 つにまとめるためであるが，表形式のデータではコンバータスクリプトでファイルを読み込むことができない。そのためコンバータスクリプトが読み込めるようなフォーマットで，必要なコンフィグパラメータの名前とビット長の情報を含んだファイルを作成した。このファイル

は BDAQ53 のコンフィギュレーションファイルに合わせて yaml フォーマットで記述した。構造としては，BDAQ53 コンフィグパラメータの名前を key として，value に YARR コンフィグパラメータの名前を key，そのコンフィグパラメータのビット長を value としたハッシュを入れている。Listing5 に作成した yaml ファイルを示す。行数が膨大になるので 2 列で表示している。

Listing 5: correspondence.yaml

```
PIX_PORTAL:
 PixPortal: 16
REGION_COL:
 PixRegionCol: 8
REGION_ROW:
 PixRegionRow: 9
PIX_MODE:
 PixBroadcastEn: 3
 PixAutoCol: 8
 PixAutoRow: 9
 PixBroadcastMask: 3
PIX_DEFAULT_CONFIG:
 PixDefaultConfig: 16
IBIASP1_SYNC:
 SyncIbiasp1: 9
IBIASP2_SYNC:
 SyncIbiasp2: 9
IBIAS_SF_SYNC:
 SyncIbiasSf: 9
IBIAS_KRUM_SYNC:
 SyncIbiasKrum: 9
IBIAS_DISC_SYNC:
 SyncIbiasDisc: 9
ICTRL_SYNCT_SYNC:
 SyncIctrlSynct: 10
VBL_SYNC:
 SyncVbl: 10
VTH_SYNC:
 SyncVth: 10
VREF_KRUM_SYNC:
 SyncVrefKrum: 10
PA_IN_BIAS_LIN:
 LinPaInBias: 9
FC_BIAS_LIN:
 LinFcBias: 8
KRUM_CURR_LIN:
 LinKrumCurr: 9
LDAC_LIN:
 LinLdac: 10
COMP_LIN:
 LinComp: 9
REF_KRUM_LIN:
 LinRefKrum: 10
Vthreshold_LIN:
 LinVth: 10
PRMP_DIFF:
 DiffPrmp: 10
FOL_DIFF:
 DiffFol: 10
PRECOMP_DIFF:
 DiffPrecomp: 10
COMP_DIFF:
 DiffComp: 10
VFF_DIFF:
 DiffVff: 10
VTH1_DIFF:
 DiffVth1: 10
VTH2_DIFF:
 DiffVth2: 10
LCC_DIFF:
 DiffLcc: 10
CONF_FE_DIFF:
 DiffLccEn: 1
 DiffFbCapEn: 1
CONF_FE_SYNC:
 SyncSelC2F: 1
 SyncSelC4F: 1
 SyncFastTot: 1
 SyncAutoZero: 2
VOLTAGE_TRIM:
 SldoAnalogTrim: 5
 SldoDigitalTrim: 5
EN_CORE_COL_SYNC:
 EnCoreColSync: 16
EN_CORE_COL_LIN_1:
 EnCoreColLin1: 16
EN_CORE_COL_LIN_2:
 EnCoreColLin2: 1
EN_CORE_COL_DIFF_1:
 EnCoreColDiff1: 16
EN_CORE_COL_DIFF_2:
 EnCoreColDiff2: 1
LATENCY_CONFIG:
 LatencyConfig: 9
WR_SYNC_DELAY_SYNC:
 WrSyncDelaySync: 5
INJECTION_SELECT:
 InjAnaMode: 1
 InjEnDig: 1
 InjDelay: 4
CLK_DATA_DELAY:
 ClkDelaySel: 1
 ClkDelay: 4
 CmdDelay: 4
VCAL_HIGH:
 InjVcalHigh: 12
VCAL_MED:
 InjVcalMed: 12
CH_SYNC_CONF:
 ChSyncPhase: 2
 ChSyncLock: 5
 ChSyncUnlock: 5
GLOBAL_PULSE_ROUTE:
 GlobalPulseRt: 16
MONITOR_FRAME_SKIP:
 MonFrameSkip: 8
EN_MACRO_COL_CAL_SYNC_1:
 CalColprSync1: 16
EN_MACRO_COL_CAL_SYNC_2:
 CalColprSync2: 16
EN_MACRO_COL_CAL_SYNC_3:
 CalColprSync3: 16
EN_MACRO_COL_CAL_SYNC_4:
 CalColprSync4: 16
EN_MACRO_COL_CAL_LIN_1:
 CalColprLin1: 16
EN_MACRO_COL_CAL_LIN_2:
 CalColprLin2: 16
EN_MACRO_COL_CAL_LIN_3:
 CalColprLin3: 16
EN_MACRO_COL_CAL_LIN_4:
 CalColprLin4: 16
EN_MACRO_COL_CAL_LIN_5:
```

```
  CalColprLin5: 4
 EN_MACRO_COL_CAL_DIFF_1:
  CalColprDiff1: 16
 EN_MACRO_COL_CAL_DIFF_2:
  CalColprDiff2: 16
 EN_MACRO_COL_CAL_DIFF_3:
  CalColprDiff3: 16
 EN_MACRO_COL_CAL_DIFF_4:
  CalColprDiff4: 16
 EN_MACRO_COL_CAL_DIFF_5:
  CalColprDiff5: 4
 DEBUG_CONFIG:
  DebugConfig: 2
 OUTPUT_CONFIG:
  OutputDataReadDelay: 2
  OutputSerType: 1
  OutputActiveLanes: 4
  OutputFmt: 2
 OUT_PAD_CONFIG:
  OutPadConfig: 14
 GP_LVDS_ROUTE:
  GpLvdsRoute: 16
 CDR_CONFIG:
  CdrSelDelClk: 1
  CdrPdSel: 2
  CdrPdDel: 4
  CdrEnGck: 1
  CdrVcoGain: 3
  CdrSelSerClk: 3
 CDR_VCO_BUFF_BIAS:
  VcoBuffBias: 10
 CDR_CP_IBIAS:
  CdrCpIbias: 10
 CDR_VCO_IBIAS:
  VcoIbias: 10
 SER_SEL_OUT:
  SerSelOut3: 2
  SerSelOut2: 2
  SerSelOut1: 2
  SerSelOut0: 2
 CML_CONFIG:
  CmlInvTap: 2
  CmlEnTap: 2
  CmlEn: 4
 CML_TAP0_BIAS:
  CmlTapBias0: 10
 CML_TAP1_BIAS:
  CmlTapBias1: 10
 CML_TAP2_BIAS:
  CmlTapBias2: 10
 AURORA_CC_CONFIG:
  AuroraCcWait: 6
  AuroraCcSend: 2
 AURORA_CB_CONFIG0:
  AuroraCbWaitLow: 4
  AuroraCbSend: 4
 AURORA_CB_CONFIG1:
  AuroraCbWaitHigh: 4
 AURORA_INIT_WAIT:
  AuroraInitWait: 11
 MONITOR_SELECT:
  MonitorEnable: 1
  MonitorImonMux: 6
  MonitorVmonMux: 7
 HITOR_0_MASK_SYNC:
  HitOr0MaskSync: 16
 HITOR_1_MASK_SYNC:
  HitOr1MaskSync: 16
 HITOR_2_MASK_SYNC:
  HitOr2MaskSync: 16
 HITOR_3_MASK_SYNC:
  HitOr3MaskSync: 16
 HITOR_0_MASK_LIN_0:
  HitOr0MaskLin0: 16
 HITOR_0_MASK_LIN_1:
  HitOr0MaskLin1: 1
 HITOR_1_MASK_LIN_0:
  HitOr1MaskLin0: 16
 HITOR_1_MASK_LIN_1:
  HitOr1MaskLin1: 1
 HITOR_2_MASK_LIN_0:
  HitOr2MaskLin0: 16
 HITOR_2_MASK_LIN_1:
  HitOr2MaskLin1: 1
 HITOR_3_MASK_LIN_0:
  HitOr3MaskLin0: 16
 HITOR_3_MASK_LIN_1:
  HitOr3MaskLin1: 1
 HITOR_0_MASK_DIFF_0:
  HitOr0MaskDiff0: 16
 HITOR_0_MASK_DIFF_1:
  HitOr0MaskDiff1: 1
 HITOR_1_MASK_DIFF_0:
  HitOr1MaskDiff0: 16
 HITOR_1_MASK_DIFF_1:
  HitOr1MaskDiff1: 1
 HITOR_2_MASK_DIFF_0:
  HitOr2MaskDiff0: 16
 HITOR_2_MASK_DIFF_1:
  HitOr2MaskDiff1: 1
 HITOR_3_MASK_DIFF_0:
  HitOr3MaskDiff0: 16
 HITOR_3_MASK_DIFF_1:
  HitOr3MaskDiff1: 1
 MONITOR_CONFIG:
  AdcRefTrim: 5
  AdcTrim: 6
 SENSOR_CONFIG_0:
  SensorCfg0: 12
 SENSOR_CONFIG_1:
  SensorCfg1: 12
 AutoRead0:
  AutoReadA0: 9
 AutoRead1:
  AutoReadA1: 9
 AutoRead2:
  AutoReadA2: 9
 AutoRead3:
  AutoReadA3: 9
 AutoRead4:
  AutoReadB0: 9
 AutoRead5:
  AutoReadB1: 9
 AutoRead6:
  AutoReadB2: 9
 AutoRead7:
  AutoReadB3: 9
 RING_OSC_ENABLE:
  RingOscEn: 8
 RING_OSC_0:
  RingOsc0: 16
 RING_OSC_1:
  RingOsc1: 16
 RING_OSC_2:
  RingOsc2: 16
 RING_OSC_3:
  RingOsc3: 16
 RING_OSC_4:
  RingOsc4: 16
 RING_OSC_5:
  RingOsc5: 16
 RING_OSC_6:
  RingOsc6: 16
 RING_OSC_7:
  RingOsc7: 16
 BCIDCnt:
```

```
 BcCounter: 16
TrigCnt:
 TrigCounter: 16
LockLossCnt:
 LockLossCounter: 16
BitFlipWngCnt:
 BflipWarnCounter: 16
BitFlipErrCnt:
 BflipErrCounter: 16
CmdErrCnt:
 CmdErrCounter: 16
WngFifoFullCnt_0:
 FifoFullCounter0: 16
WngFifoFullCnt_1:
 FifoFullCounter1: 16
WngFifoFullCnt_2:
 FifoFullCounter2: 16
WngFifoFullCnt_3:
 FifoFullCounter3: 16
AI_REGION_COL:
 AiPixCol: 8
AI_REGION_ROW:
 AiPixRow: 9
HitOr_0_Cnt:
 HitOrCounter0: 16
HitOr_1_Cnt:
 HitOrCounter1: 16
HitOr_2_Cnt:
 HitOrCounter2: 16
HitOr_3_Cnt:
 HitOrCounter3: 16
SkippedTriggerCnt:
 SkipTriggerCounter: 16
ErrWngMask:
 ErrMask: 14
MonitoringDataADC:
 AdcRead: 12
SELF_TRIGGER_ENABLE:
 SelfTrigEn: 8
```

4.5　コンバータスクリプト

スキャン実行時，BDAQ53 のコンフィギュレーションファイルの書き出しや，読み込みは Python で書かれたプログラムで行われることと、コンバータスクリプトは BDAQ53 のコンフィギュレーションファイルを生成する必要があることから，コンバータスクリプトは Python で作成した。Listing6 にコンバータスクリプトを示す。

Listing 6: converter.py

```
import os
import yaml
import json


#get absolute path of 'rd53a_registers.yaml'
bdaq_config = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'rd53a_registers.yaml')

#get absolute path of 'correspondence.yaml'
bdaq_yarr = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'correspondence.yaml')

#get absolute path of 'default_rd53a.json'
yarr_config = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'default_rd53a.json')

#get absolute path of 'rd53a_default.cfg.yaml'
bdaq_default_cfg = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'rd53a_default.cfg.yaml')


with open(bdaq_config, 'r') as fy:
    b_conf = yaml.full_load(fy)

with open(bdaq_yarr, 'r') as fy:
    by_conv = yaml.full_load(fy)

with open(yarr_config, 'r') as fj:
    y = json.load(fj)
    y_conf = y['RD53A']
    y_regs = y_conf['GlobalConfig']

with open(bdaq_default_cfg, 'r') as fy:
    b_def_conf = yaml.full_load(fy)


for i in b_conf['registers']:
    by_hash = by_conv[i['name']]
    a = 0
    for idx,j in enumerate(by_hash.keys()):
        if idx == 0:
            a = y_regs[j]
        else:
            a = a * 2 ** by_hash[j] + y_regs[j]
    i['default'] = bin(a)

    for k in b_def_conf['registers']:
        if i['name'] == k:
            b_def_conf['registers'][k] = a

with open('rd53a_registers_converted.yaml', 'w') as file:
    yaml.dump(b_conf, file, default_flow_style=False)

with open('rd53a_default.cfg_converted.yaml', 'w') as file:
    yaml.dump(b_def_conf, file, default_flow_style=False)
```

コンバータスクリプトは同一ディレクトリに，以下のファイルが存在するときに動作する。

- コンバータスクリプト (converter.py; Listing 6)
- 対応表を変換した yaml ファイル (correspondence.yaml; Listing 5)
- YARR コンフィギュレーションファイル (default_chip.json; Listing 2)
- BDAQ53 コンフィギュレーションファイル 1(rd53a_default.cfg.yaml; Listing 3)
- BDAQ53 コンフィギュレーションファイル 2(rd53a_registers.yaml; Listing 4)

BDAQ53 のコンフィギュレーションファイルを生成することから，これらのソースコードを配置するディレクトリは元のコンフィギュレーションファイルがあるディレクトリと同じがよいと考えられる。

以下に BDAQ53 のディレクトリのツリー構造を示す。青文字で書かれているのがディレクトリであり，黒文字はファイルを表す。また，BDAQ53 のトップディレクトリ”bdaq53” 以下のすべてのディレクトリ，及びファイルを示すと膨大な量になってしまうのでコンバータの作成において関連するディレクトリ構造のみを示す。BDAQ53 コンフィギュレーションファイルの rd53a_default.cfg.yaml と rd53a_registers.yaml は以下に示したツリー構造の同一ディレクトリに存在するので，この 2 つのファイルがあるディレクトリに残りの 3 つのソースコードを配置する。

```
bdaq53
├── LICENSE
├── MANIFEST.in
├── README.md
├── bdaq53.egg-info
├── data
├── example
├── firmware
├── setup.cfg
├── setup.py
└── bdaq53
    ├── README.md
    ├── __init__.py
    ├── __pycache__
    ├── analysis
    ├── aurora_rx.py
    ├── bdaq53.py
    ├── bdaq53.yaml
    ├── bdaq53_cli.py
    ├── chip_base.py
    ├── cmd_rd53.py
    ├── fifo_readout.py
    ├── global_pulse_route.yaml
    ├── logger.py
    ├── logo.bmp
    ├── manage_configuration.py
    ├── manage_firmware.py
    ├── measurements
    ├── periphery.py
    ├── periphery.yaml
    ├── rd53a.py
    ├── rd53a_default.cfg.yaml ..... コンフィギュレーションファイル
    ├── rd53a_registers.yaml ..... コンフィギュレーションファイル
    ├── register_utils.py
    ├── scan_base.py
    ├── scans
    ├── testbench.yaml
    └── tests
```

コンバータスクリプトによって生成されたコンフィギュレーションファイルを，Listing 7 と Listing 8 に示す。どちらのコンフィギュレーションファイルもコンバータスクリプトにより，コンフィグパラメータの値が変換されていることがわかる。また，Listing 7 の rd53a_default.cfg_converted.yaml では，コメントアウトが消えているとともに，行順がアルファベット順に変換されている。これは辞書型のデータオブジェクトが要素の格納順を意識しないということから，yaml.dump() 関数[*19]によって yaml ファイルとして辞書型オブジェクトが出力される際にアルファベット順に並び変えられてしまうからである。Python パッケージ[*20]を導入すること等で解決できるのだが，この程度のコンフィギュレーションファイルの変化では BDAQ53 の動作に影響を及ぼさないことや，実行環境によらない汎用性を重視したことから導入しないことにした。

[*19] 関数とは，引数を受け取り，その引数をもとに処理を行い結果を返す機能を持つものである。yaml.dump() は yaml モジュールにおける dump 関数であり，() 内のオブジェクトを引数として受け取り yaml ファイルとして出力する関数である。

[*20] モジュールという Python ソースコードを複数まとめたものであり，導入することでモジュールに記述されているクラスや関数を利用することができるようになる。

Listing 7: rd53a_default.cfg_converted.yaml

```yaml
calibration:
  ADC_a: 0.2
  ADC_b: 15.5
  Nf: 1.24
  e_conversion:
    offset: 180
    offset_error: 60
    slope: 10.4
    slope_error: 0.1
registers:
  COMP_DIFF: 1000
  COMP_LIN: 110
  CONF_FE_DIFF: 0
  CONF_FE_SYNC: 8
  FC_BIAS_LIN: 20
  FOL_DIFF: 500
  IBIASP1_SYNC: 80
  IBIASP2_SYNC: 120
  IBIAS_DISC_SYNC: 300
  IBIAS_KRUM_SYNC: 55
  IBIAS_SF_SYNC: 80
  ICTRL_SYNCT_SYNC: 100
  KRUM_CURR_LIN: 50
  LCC_DIFF: 20
  LDAC_LIN: 100
  PA_IN_BIAS_LIN: 300
  PRECOMP_DIFF: 400
  PRMP_DIFF: 500
  REF_KRUM_LIN: 300
  VBL_SYNC: 400
  VFF_DIFF: 50
  VREF_KRUM_SYNC: 450
  VTH1_DIFF: 250
  VTH2_DIFF: 50
  VTH_SYNC: 300
  Vthreshold_LIN: 400
trim:
  MON_ADC_TRIM: 5
  MON_BG_TRIM: 12
  VREF_A_TRIM: 24
  VREF_D_TRIM: 23
```

Listing 8: rd53a_registers_converted.yaml

```yaml
registers:
- address: '0x0'
  default: '0b11100000100'
  description: Virtual register to access pixel matrix
  mode: 1
  name: PIX_PORTAL
  reset: 0
  size: 16
- address: '0x1'
  default: '0b11000111'
  description: Region Column Address
  mode: 1
  name: REGION_COL
  reset: 1
  size: 8
- address: '0x2'
  default: '0b10000111'
  description: Region Row Address
  mode: 1
  name: REGION_ROW
  reset: 1
  size: 9
- address: '0x3'
  default: '0b0'
  description: 'Mode bits: Broadcast, AutoCol, AutoRow, BroadcastMask'
  mode: 1
  name: PIX_MODE
  reset: 0
  size: 6
 .
 .
 .
 .省略

 .
 .
 .
 .
 - address: '0x87'
  default: '0b0'
  description: Mask single Error Warning messages
  mode: 1
  name: ErrWngMask
  reset: 0
  size: 14
- address: '0x88'
  default: '0b0'
  description: Contains the value of the ADC to be read back
  mode: 0
  name: MonitoringDataADC
  reset: 0
  size: 12
```

4.6　BDAQ53 での読み出しに向けた SMA-Displayport ケーブルの製作と検証

コンバータスクリプトを用いて YARR と BDAQ53 の条件を合わせた状態での読み出しを行いたいが，RD53A シングルチップカード[*21]を用意できる目途が立っていない。そこで BDAQ53 での読み出しに向けて，現在できることとして SMA-Displayport ケーブルの製作と，そのケーブルを用いた IBERT(Integrated Bit Error Ratio Test) を実行し，ケーブルの通信可能データレートを検証した。

4.6.1　SMA-Displayport ケーブル

RD53A と KC705 を接続するため，片方の端子が SMA コネクタで，もう片方の端子が Displayport のケーブルを文献 [10] を参考にして製作した。

差動信号が 2 レーン必要なことから，両端が SMA コネクタの同軸ケーブルを 2 本用意し，それをちょうど半分の長さで切断して差動信号のポジティブ信号とネガティブ信号のレーンを 2 組用意する。この半分に切断した同軸ケーブルを Displayport コネクタにはんだ付けする。同軸ケーブルは図 45 のように中心に芯線が通っており，順に絶縁体，外部導体，外部被膜が同心円状に芯線を覆っている。芯線は信号を伝送するためのものであり，外部導体は GND として利用する。

Displayport コネクタにはんだ付けした同軸ケーブルのピンアサインを図 46 に示す。また，図 47 に製作した SMA-Displayport ケーブルを示す。

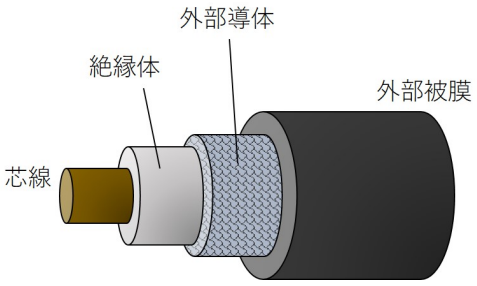


図 45: 同軸ケーブル

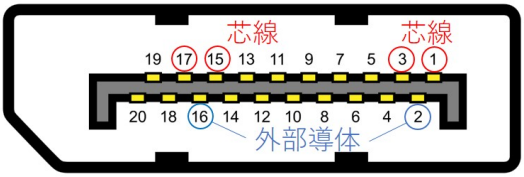


図 46: Displayport ピンアサイン

[*21] チップが 1 つだけ実装されたモジュールをシングルチップカードと呼ぶ。

図 47: SMA-Displayport ケーブル

4.6.2　IBERT を用いたアイスキャンの観測

Xilinx, Inc. は IP(Intellectual Property：知的財産) という，あらかじめ最適化された機能を備えた回路データを提供している。IBERT とは GTX トランシーバのデバッグ IP のことで，エラーレートや，アイスキャンの観測が可能である。

アイスキャンとは図 48 に示すように，ユニットインターバル (UI)[*22]単位の波形を重ねることで波形に囲まれた部分がアイ (目) の形に浮かび上がってくることから名づけられた，伝送特性を評価する方法のことである。ジッタという信号の立ち上がり時間の揺らぎが大きいと，アイの領域が小さくなることから，アイが大きいほど伝送特性が良いことになる。

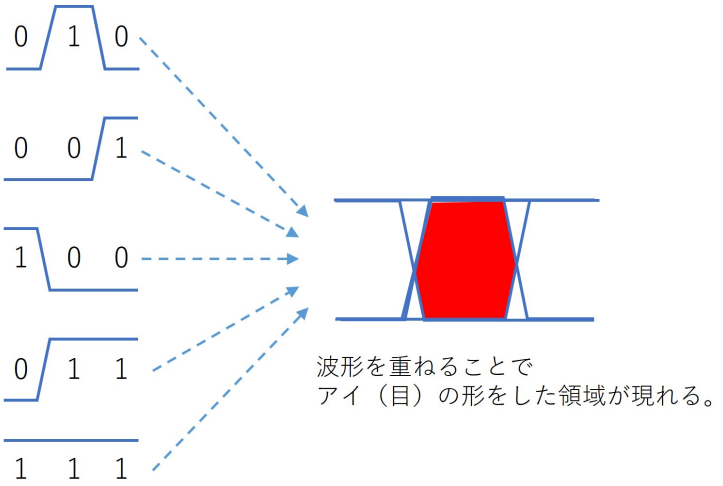


図 48: アイスキャン説明図

[*22] デジタル信号の 1 ビット分の範囲

製作した SMA-Displayport ケーブルの最大データレートを検証するために，IBERT を用いてループバック試験を行った。その結果，1.5Gbps までのデータレートではアイが開いているものの，1.6Gbps 以上のデータレートではアイが閉じてしまうことが分かった。1.5Gbps でのアイと 1.6Gbps でのアイを図 49，50 に示す。図 49，50 は，BER(Bit Error Rate：ビット誤り率) という，全ビット数に対する誤りビット数の割合からアイを求めている[*23]。

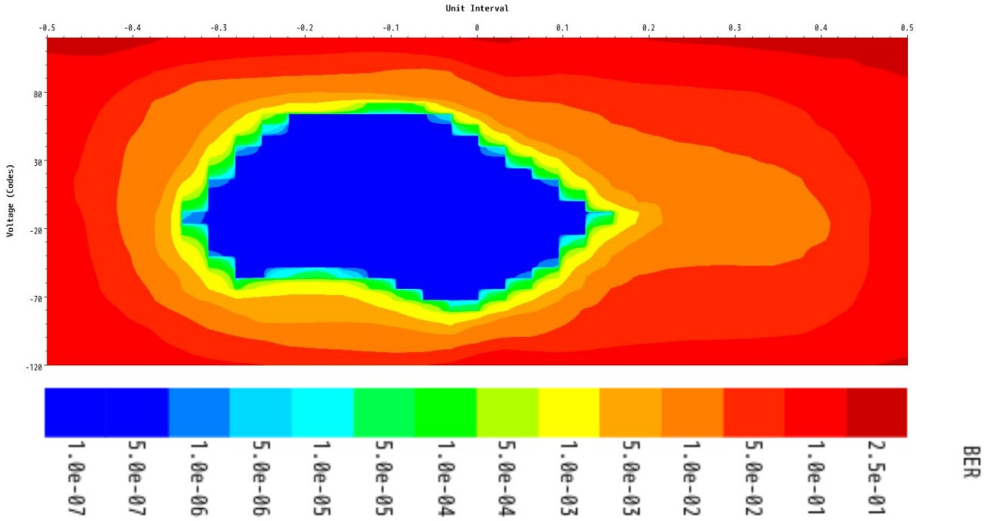


図 49: 1.5Gbps でのループバック試験実行結果

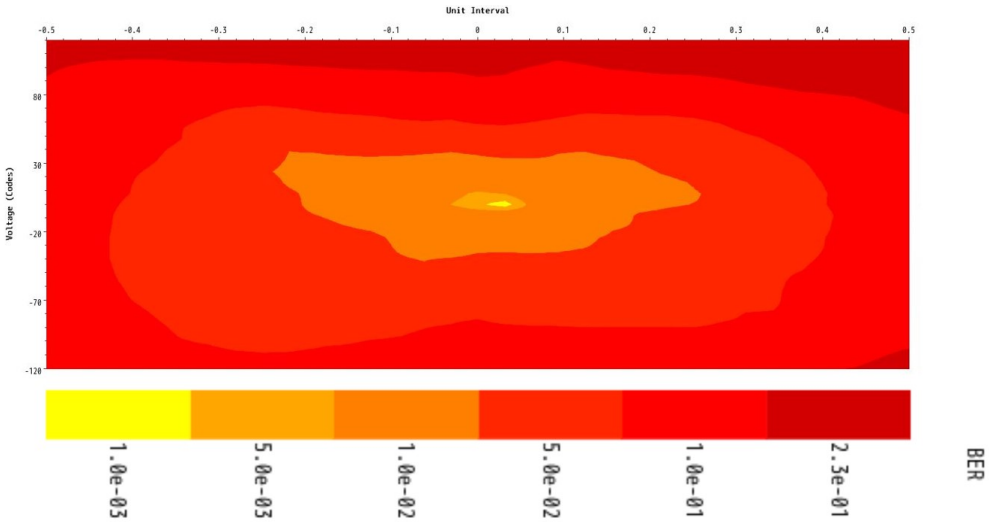


図 50: 1.6Gbps でのループバック試験実行結果

[*23] 文献 [11] によると，地上デジタル放送では BER が 2×10^{-4} を超えなければ，ほとんど劣化のない画質で視聴することができる。

4.7　本章における課題と結論

製作したコンバータスクリプトはコンフィグレーションの対応表を yaml 形式に変換したファイルと共に使用する必要があり，ユーザビリティにかける部分がある。そこで，対応表の情報をコンバータスクリプトそのものに埋め込めるようにしたい。

また，コンバータスクリプトにより生成されたコンフィギュレーションファイルは，コメントが消えたり，辞書型のデータがアルファベット順に並び変えられたりと，スキャンの実行に影響を与えない記述の変化が発生した。生成されたコンフィギュレーションファイルの可読性や，元のコンフィギュレーションファイルとの整合性を保つために，システムの動作に影響を与えないようにこれらを解決したい。

YARR と BDAQ53 のコンフィグパラメータ対照表を製作して，両方のコンフィギュレーションファイルのデフォルト値に大きな違いがあることが分かった。今後，YARR と BDAQ53 を比較することがあるのであれば，コンフィグパラメータを十分意識する必要がある。

BDAQ53DAQ システムでの読み出しに向けて，必要なケーブルを製作し，ループバックテストを行った。その結果 1.5Gbps 程度までなら，アイが閉じない正常な通信ができることが分かった。文献 [10] より，BDAQ53 ファームウェアの通信速度は 1.28Gbps が最大であることから，製作したケーブルを用いるうえで問題はないと結論付ける。

5　本研究における結論

HL-LHC ATLAS 実験に向けたフロントエンド ASIC の読み出しシステムについての研究として，本研究の結論を以下に記す。

- 京都教育大学で現行 ASIC，FE-I4 の読み出しシステムを構築した。だが，スキャンの実行は成功したものの，疑似パルスの読み出しには至っていない。
- 4 チップ読み出しシステムを構築するために，専用のインターフェースカードとファームウェアを開発した。開発した 4 チップ読み出しシステムで 1 つのチップが故障しているクアッドモジュールのスキャンを実行したところ，故障しているチップを除いた 3 チップすべての読み出しに成功した。読み出しシステムとしては 4 チップも読み出しが可能であると考える。
- YARR から BDAQ53 へののコンフィグファイルコンバータを製作した。
- BDAQ53 での読み出しに向けて SMA-Displayport ケーブルのループバック試験を行い，1.5Gbps までのデータレートで通信が可能であることを確認した。

6　謝辞

本研究を進めるにあたって，基礎物理学研究室の高嶋隆一教授には積極的に助言や提言をしていただくとともに，自由に研究活動をさせていただきました。ありがとうございました。研究中におやつタイムと称してシュークリームをご馳走していただくなど，ご配慮いただき大変感謝しております。

ATLAS-J QA/QC グループの大阪大学山中卓研究室の皆様には大変お世話になりました。南條創准教授には 4 チップ読み出しに向けたインターフェースカードの配置などで助言を頂いたり，基板加工機を使わせていただりしました。廣瀬穣助教授には YARR や BDAQ53 での読み出しに必要な知識を丁寧に教えていただくとともに，勉強不足な私の質問に対しても親切にお答えいただきました。お忙しい中ご迷惑おかけして申し訳なく思っておりますが，そんな中対応していただき大変助かりました。大西裕二さんにはコンバータ製作でプログラムの方針を助言していただきました。また，大阪大学へ出張したときに声をかけていただき，緊張していたところで気が休まりました。山家谷昌平君には YARR の読み出しについて丁寧に教えていただいたり，ITK ミーティングでの発表についての助言など大変お世話になりました。皆さま大変ありがとうございました。

基礎物理学研究室で同期の大川賢悟君には，研究のことも含めて色々と話に付き合ってもらいました。たわいのない話など研究以外のことを話せたことで，楽しい研究生活を過ごせたとともに良い気分転換になりました。

京都教育大学理科教育専修の皆さまや，基礎物理学研究室の皆さまとは，授業や研究で共に学ぶだけでなく，同じ学生として楽しく過ごさせていただきました。

深草寮，並びに露草寮の皆さまには，大学，大学院含め 6 年間お世話になるとともに，皆様のおかげで学生生活において非常に充実した日々を送ることができました。

最後になりましたが，学生生活 6 年間を支えていただいた両親には深く感謝します。ありがとうございました。

参考文献

[1] ザイリンクス FPGA 講座 2019/12/24 閲覧
https://japan.xilinx.com/japan/fpga-koza.html

[2] KC705 リファレンスデザイン ユーザーガイド UG883 (v1.0) 2012 年 1 月 13 日

[3] Xilinx, KC705 Evaluation Board for the Kintex-7 FPGA User Guide UG810 (v1.9) February 4，2019

[4] Yet Another Rapid Readout https://gitlab.cern.ch/YARR/YARR

[5] Timon Heim， YARR - A PCIe based Readout Concept for Current and Future ATLAS Pixel Modules， Journal of Physics 898 032053 (2017)

[6] 小林 優，FPGA プログラミング大全 Xilinx 編 株式会社秀和システム 2016/12/20 発行

[7] 東京エレクトロンデバイス株式会社，TB-FMCL-PH ハードユーザマニュアル

[8] Tomohisa Uchida，SEABAS(Soi EvAluation BoArd with Sitcp) User's Manual

[9] Teoh Jia Jian，Development of SiTCP Based Readout System for The ATLAS Pixel Detector Upgrade

[10] BDAQ53 software repository，https://gitlab.cern.ch/silab/bdaq53

[11] 総務省 東海総合通信局 地デジ用語集 2020/1/8 閲覧
https://www.soumu.go.jp/soutsu/tokai/housou/digital/yougo/yougo.html