修　士　学　位　論　文

Zynqを用いた
ワイヤーチェンバー試験用読み出しシステムの開発

令和4年2月4日

専 攻 名　物理学専攻
学籍番号　201S103S
氏　　名　池森 隆太郎

神戸大学大学院理学研究科博士課程前期課程

概要

多くの素粒子実験結果を説明する標準模型において、唯一未観測であったヒッグス粒子が 2012 年に初めて観測された。しかし標準模型では説明のつかない物理事象も確認されている。欧州原子核研究機構 (CERN) では、地下にある Large Hadron Collider (LHC) を用いて、それら標準模型を超える物理の探索実験が行われている。LHC では世界最高エネルギーでの陽子-陽子衝突実験が行われている。今後は興味のある物理事象のさらなる統計量増大に向け高輝度化アップグレードが予定されており、2029 年の運転開始を目指している。

ATLAS 実験はその LHC で行われている実験の一つであり衝突点を囲うように配置されている A Toroidal LHC ApparatuS (ATLAS) 検出器用いて新粒子の探索やヒッグス粒子の精密測定を行っている。Thin Gap Chamber (TGC) は ATLAS 実験のエンドキャップ部に配置されるミューオントリガー用のワイヤーチェンバーである。LHC の高輝度化アップグレードに伴い、ATLAS 実験 TGC 検出器システムにおいて新規ワイヤーチェンバーを導入予定である。本研究の目的はその新規導入予定である検出器の製造検査時試験を一つの使用用途とする、汎用的で簡易的な新しい読み出しシステムの開発を行うことである。

一般的な読み出しシステムには使用者が任意に内部ロジックを実装できる Field Programmable Gate Array (FPGA) が用いられる。FPGA を用いて信号のデジタル化やトリガー処理が行われた後、読み出されたデータはストレージサーバーに蓄えられる。蓄えられたデータは別の場所にある PC からインターネット通信を介して取得され、解析が行われる。本研究ではこのような読み出しシステムを汎用的・簡易的に実現するために Xilinx 社から提供される Zynq を用いる。Zynq は FPGA と CPU が一体化した System on Chip (SoC) デバイスである。FPGA を用いた読み出しデータ処理と CPU に搭載する Operation System (OS) により容易に実現するインターネット通信を用いて読み出しシステムを開発する。

FPGA を用いてデジタル信号の位置・時間情報とアナログ信号の波形情報の 2 つのデジタル化を行い、データの読み出しを行う。これらの読み出しは FPGA での任意なトリガー処理が可能である。取得されたデータは Zynq 上の CPU からアクセスを行い、データファイルとして外部 PC へ転送することを想定する。システム実現のために本研究では Zynq を中枢機能とする基板の設計を行った。回路図設計を終えた今は基板作成が行われている。本論文ではその読み出し用基板の詳細を説明した後、システムを実現するために必要である機能について説明する。またその機能の一部は既に検証済みであることを示す。

目次

図目次

表目次

第 1 章

序論

1.1　標準模型

素粒子とは物質の基本構成要素である。素粒子とその相互作用を記述した理論が標準模型である。標準模型では図 1.1 に示すように、12 種類のフェルミオンと 4 種類のゲージボソン、ヒッグス粒子の計 17 種類の素粒子が記述されている [1]。2012 年にヒッグス粒子が発見されたことにより、これら 17 種類全ての素粒子の存在が証明された。

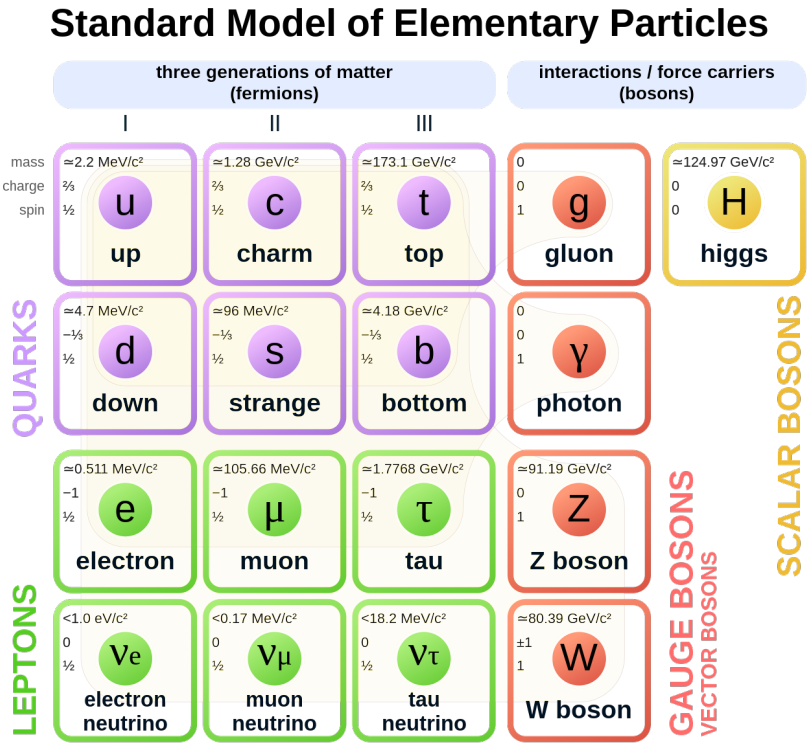


図 1.1: 標準模型を構成する粒子 [1]

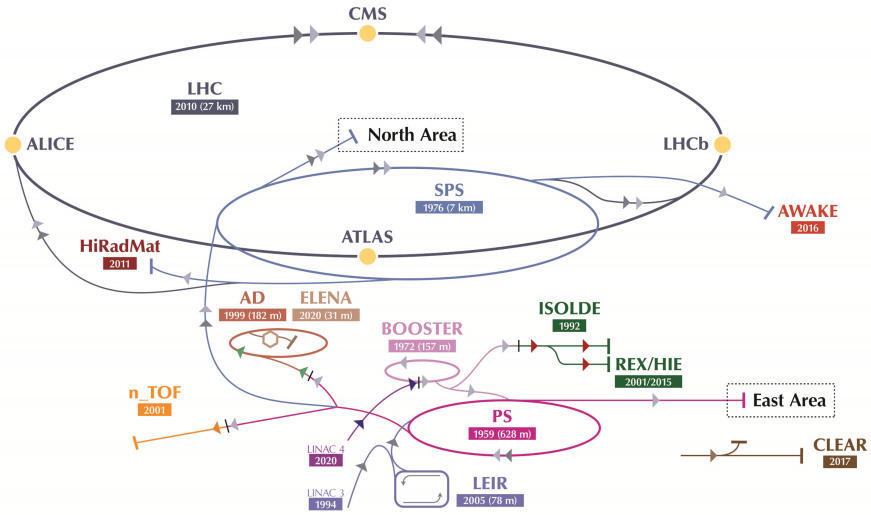


図 1.2: LHC 加速器の全体像 [4]。ATLAS のほかに ALICE、CMS、LHCb の計 4 つの検出器が配置されている。

1.2　標準模型を超える物理の探索

標準模型により、ほとんどの素粒子実験結果を説明することに成功している。その一方で標準模型での説明ができない現象も観測されている。ヒッグス粒子の階層性問題、ダークマターの存在、ニュートリノの質量起源などがその例である。これらの現象を説明するために標準模型を超える新物理の解明に向けた様々な実験が、現在世界中で行われている。

スイス・ジュネーブ郊外にある欧州原子核研究機構 (CERN)[2] では、地下にある Large Hadron Collider (LHC)[3] を用いて世界最高エネルギーである重心エネルギー 13 TeV の陽子-陽子衝突実験が行われている。LHC 加速器の全体像を図 1.2 に示す [4]。衝突点に設置されている検出器を用いて衝突の結果として生じる物理現象を観測している。LHC では加速器の輝度を向上させる高輝度 LHC アップグレードが予定され、2029 年の運転開始を目指している。高輝度 LHC アップグレードの目的は興味のある物理事象データの統計量増大であり、現行 LHC のおよそ 3 倍である 5〜 $7.5 \times 10^{34}\,\mathrm{cm^{-2}s^{-1}}$ の瞬間最高ルミノシティを達成する。ここでルミノシティの定義を式 (1.1) に示す。

$$L = \frac{N}{\sigma} \tag{1.1}$$

ここで L はルミノシティ、N はビーム粒子の単位時間あたりの反応数、σ は全反応断面積を示す。よってルミノシティとは加速器における衝突点での粒子同士の衝突頻度を表す指標である。瞬間ルミノシティは毎秒当たりの衝突頻度である。時間経過により蓄積した数値は積算ルミノシティと呼ばれ、単位には $\mathrm{fb^{-1}}$ が用いられる。高輝度 LHC での積算ルミノシティは運転期間全体で $3000\,\mathrm{fb^{-1}}$ 以上に到達する。また、高輝度 LHC での重心エネルギーは 14 TeV での運転が予定さ

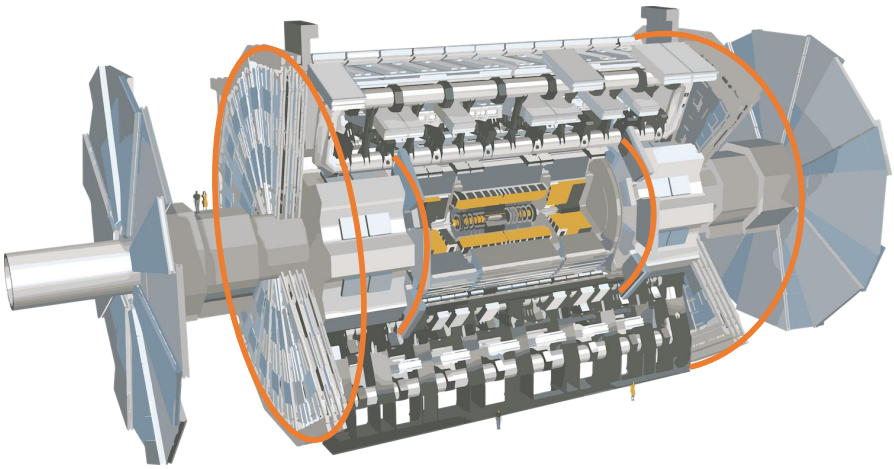


図 1.3: ATLAS 検出器 [6]。TGC 検出器の部分をオレンジ色の枠で示した。

れている。

1.3　ATLAS 実験と TGC 検出器

ATLAS 実験は LHC で行われている実験の一つであり、新粒子の探索やヒッグス粒子の精密測定を行っている [5]。ATLAS 実験には検出器衝突点を囲うように配置されている A Toroidal LHC ApparatuS (ATLAS) 検出器が用いられる。その ATLAS 検出器の概観を図 1.3 に示す [6]。ATLAS 実験でも LHC の高輝度化アップグレードに伴い、使用するエレクトロニクスの刷新や検出器の新規導入・高密度化が計画されている。図 1.3 にオレンジ色の枠で示される検出器が Thin Gap Chamber (TGC) 検出器である。TGC 検出器は ATLAS 検出器のエンドキャップ部に配置されるミューオントリガー用のワイヤーチェンバーである。エンドキャップ部外側の Big Wheel (BW) と呼ばれる検出器とエンドキャップ部内側の Endcap Inner (EIL4) と呼ばれる検出器で構成されている。

ここで TGC 検出器の構造について説明する。TGC 検出器は図 1.4 に示すようなガスギャップ 2.8 mm の Multi Wired Proportional Chamber (MWPC) である [7]。CO_2 55%、n-pentane 45% の混合ガスで動作し、アノードワイヤーには約 2.8 kV の印加電圧がかけられる。アノードワイヤーには直径 50 µm の金メッキタングステンワイヤーが使用され、1.8 mm の間隔で張られている。アノードワイヤーから 1.4 mm のところにあるカソードには片面に高抵抗なカーボンが塗布されたエポキシガラス板を使用している。エポキシガラス板の反対面にはストリップがアノードワイヤーと直交する向きに張られている。ガス中を荷電粒子が通過すると、経路上のガス分子が電離する。電離によって生成された一次電子はアノード側にドリフトしながら印加電圧によって加速し、ガス分子の電離エネルギーを超えると二次電子を生成する。このような電子生成が繰り返し発生することで電子雪崩が発生する。電子雪崩により電子雲と陽イオン雲が生成され、逆行する向きにド

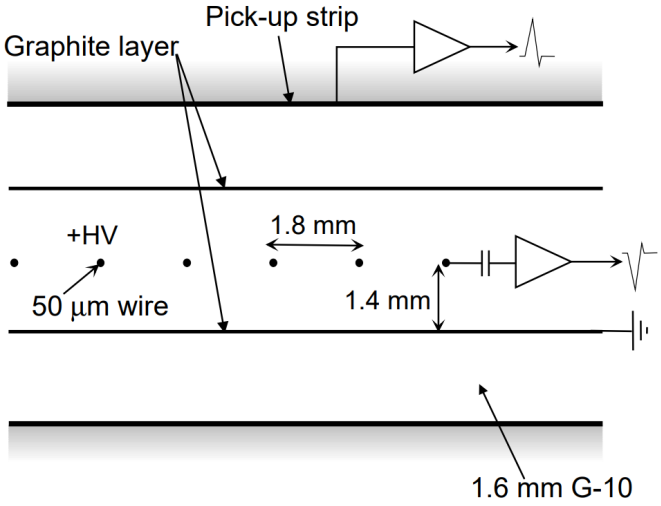


図 1.4: TGC 検出器の構造 [7]

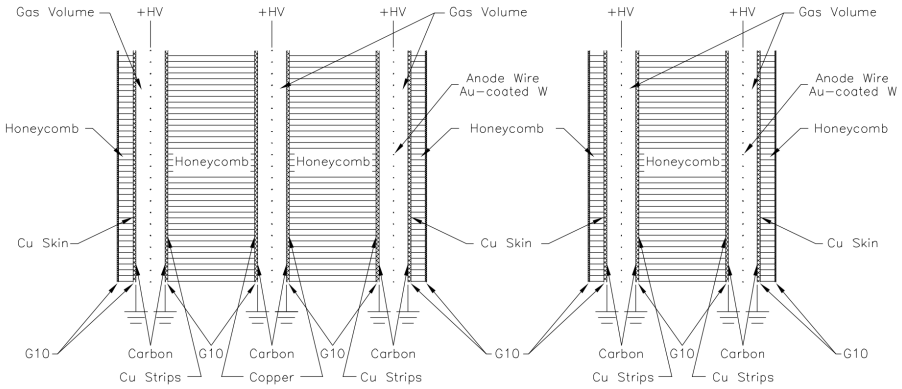


図 1.5: TGC 検出器の triplet・doublet の構成 [7]

リフトしていく。TGC 検出器ではこのような過程でできる電子雪崩をシグナルとしてアノードワイヤーから読み取る。一方、高抵抗のカーボン面が塗布されたカソード面では流れる電流が制限されており、カソードの外側に位置するストリップに電荷が誘起される。これをシグナルとして読み取ることで 2 次元の位置情報を読み出すことを可能にしている。

TGC 検出器には 3 層構造の triplet チェンバーと doublet チェンバーが存在する。それぞれの構成を図 1.5 に示す [7]。triplet チェンバーはワイヤー面 3 層・ストリップ面 2 層により構成される一方、doublet チェンバーはワイヤー面が 1 層少なくワイヤー面 2 層・ストリップ面 2 層で構成される。現行の EIL4 は 2 層構造であるが、高輝度化に伴い 3 層構造へのアップグレードが予定されている。BW よりも内側に存在する EIL4 の役割は陽子-陽子衝突点以外からのバックグラウンド事象を取り除くことにある。BW でのミューオン感知した場合に、EIL4 の doublet チェンバーのうち 1 層以上でのミューオン感知との同時計測条件をトリガー生成に課すことでバックグラウン

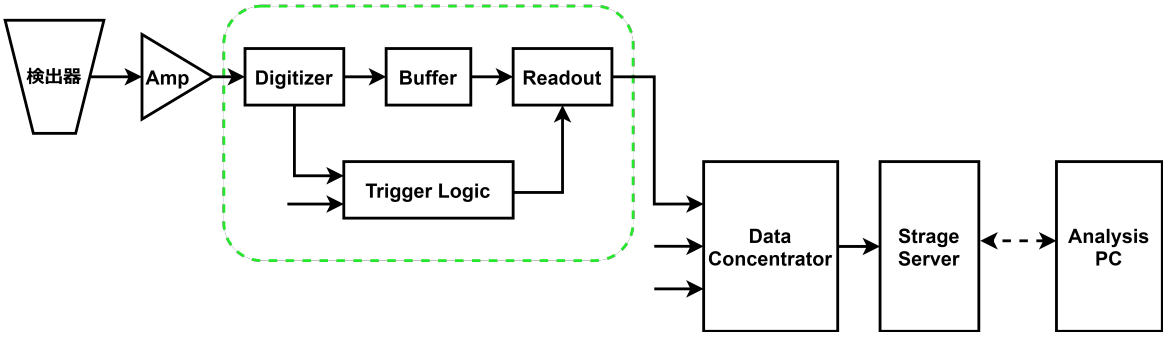


図 1.6: 検出器実験における一般的な読み出しデータの流れ

ド事象を削減している。EIL4 を triplet チェンバーに変更し、そのうち 2 層でのミューオン感知をトリガー条件に課すことでバックグラウンド由来のミューオンをさらに削減することがアップグレードの目的である。

1.4　一般的な検出器実験における読み出しシステム

一般的な高エネルギー実験における検出器読み出しデータの流れは図 1.6 のように構成される。中でも、図中点線で囲んだ部分を本論文では読み出しシステムと呼ぶ。以下でデータの流れに沿って各項目について説明する。検出器に粒子が飛来した際に生成されるアナログ信号は適度な電圧値に増幅・整形された後、デジタイザを用いてデジタル信号化 (数値化) される。ここでデジタル信号化について説明する。デジタル信号化はある一定の閾値電圧との比較が行われることで ‘1’ もしくは ‘0’ に振り分けられるだけではなく、振り分けられた情報から検出器へ飛来した位置情報や時間情報を取得するといった加工もデジタル化として表現できる。あるいは、元のアナログ信号の波形を得るために Analog to Digital Converter (ADC) を用いたデジタル化もこのデジタイザの役割として含まれる。デジタル化されたデータが膨大である場合、それらを全て保存することは処理を行うハードウェアの動作速度や保存先の容量による制限を考慮すると現実的ではない。そのため、必要に応じてデジタル化されたデータをバッファを用いて一時的に保存した後にデータの取捨選択を行う。データの取捨選択にはトリガーロジックが使用される。これにより興味のある事象を残しながら、データ量の削減を実現する。実際に ATLAS 実験のハードウェアを用いたトリガーロジックでは 40 MHz で発生する衝突事象データに対し 100 kHz 程度のレートまで削減している。選別された事象は、ATLAS 実験のような大規模な実験の場合、データコンセントレータを用いて他の検出器のデータと集約される。集約されたデータはストレージサーバーに蓄えられる。蓄えられたデータは解析を行う際に別の場所にある PC からアクセスされる。

これらの読み出しシステムには Field Programmable Gate Array (FPGA) がよく使用される。FPGA とは使用者により内部ロジックの作成が可能なデジタル集積回路である。FPGA を用いて粒子飛来の位置情報や時間情報、ADC からの波形情報を取得することが一般的である。自由に内部ロジックを決定できるため、任意なトリガーロジックの作成も可能にする。また、ストレージ

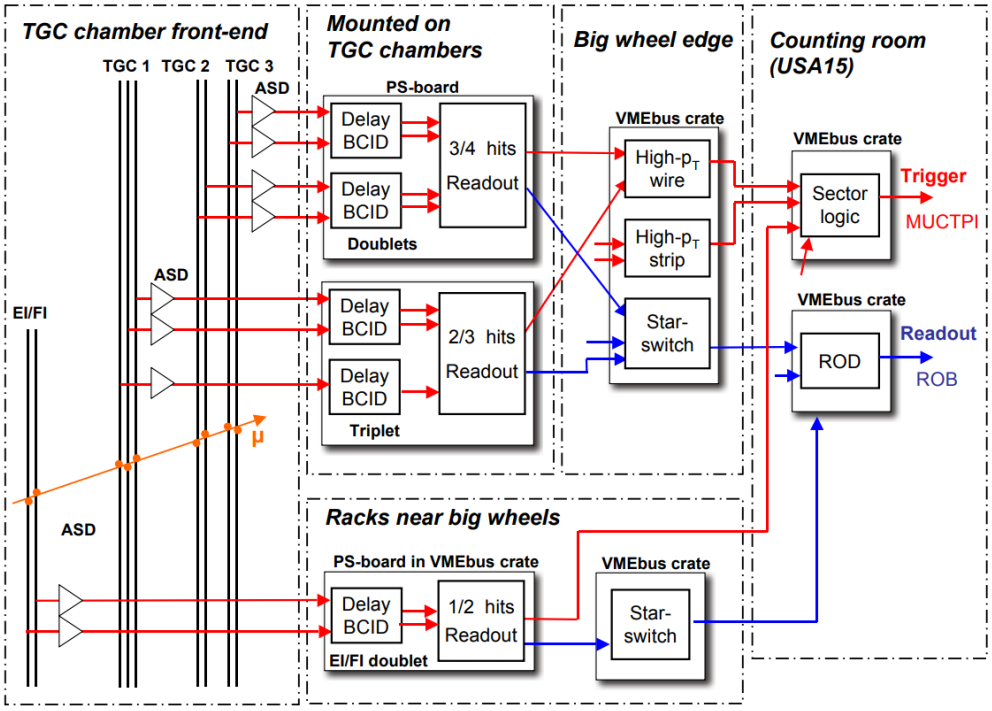


図 1.7: ATLAS 実験 TGC 検出器を用いたトリガーロジック及び読み出しシステムにおける電気回路の流れ [7]。主に各基板上の FPGA や ASIC により機能実装が行われている。

サーバーと解析用 PC の間の通信は TCP/IP 通信を用いるのが一般的である。TCP/IP 通信とはコンピュータ間の通信に標準的に使用される通信規格であり、インターネット通信においても用いられている。外部に位置する解析用 PC との TCP/IP 通信を行うことで実験者による解析が可能になる。

ATLAS 実験のような莫大な数のチャンネルを持つ大規模実験において、以上のような読み出しシステムの実現には多くの基板が用いられる。例として ATLAS 実験 TGC 検出器用に用いられているエレクトロニクスの全体像を図 1.7 に示す [7]。使用される基板にはそれぞれの機能を満たすために FPGA や特定用途向けのデジタル集積回路である Application Specific Integrated Circuit (ASIC) が搭載されている。一方、新規検出器の開発や製造検査時に行う検出器試験システムにおいてはここまで複雑化されているトリガーロジックや読み出しシステムは必要でなく、むしろ簡易的に検出器試験が可能な新しい読み出しシステムが必要になる。また、中小規模の実験においてもトリガーロジックと読み出しシステムを 1 つの基板に実装し、その基板と外部の解析用 PC のみで読み出しが完結することが理想である。本研究では LHC の高輝度アップグレードに伴い ATLAS 実験に新規導入されるワイヤーチェンバー試験を、一つの使用目的とする汎用性の高い読み出しシステムの開発を行う。

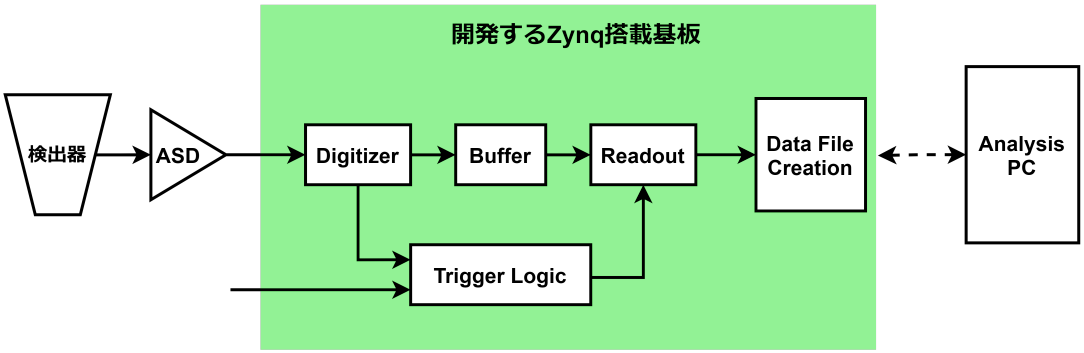


図 1.8: 開発する Zynq 搭載基板を用いた読み出しシステム

1.5　本研究において開発する読み出しシステムの概要

ATLAS 実験における読み出しシステムにも使用されているように、信号処理には FPGA を用いるのが便利であり一般的である。任意のロジックの組み込みが可能な FPGA を用いることで汎用性を大幅に高めることができるほか、FPGA が得意とする高速・大容量の通信の使用も可能である。また、「1 つの基板と外部 PC での読み出しシステムである」という簡易性を求める上で、基板上には Central Processing Unit (CPU) が求められる。CPU は汎用 Operation System (OS) の搭載により、容易な TCP/IP 通信やユーザープログラムでのデータ解析が可能になり、機能の導入が簡素化される。

Xilinx 社により開発・提供される Zynq は FPGA と CPU がワンチップ化した System on Chip (SoC) デバイスである [8]。Zynq は FPGA を担う Programmable Logic (PL) 部と、CPU と周辺機器操作用のコントローラーが含まれる Processing System (PS) 部により構成され、両者間の通信も可能である。本研究の内容として、この Zynq を機能の中枢とした基板の開発を行い、図 1.8 に示すようなシステムを開発する。開発を行う基板は、検出器からの信号が Amplifier-Shaper-Discriminator (ASD) を経由することを想定している。ASD とは検出器から信号を受け取り、増幅、波形整形、デジタル信号化を行う機能ブロックである。デジタル信号化は設定される閾値電圧値と入力信号の比較を行い、入力信号の電圧値が閾値電圧値よりも高い場合に ‘1’、低い場合には ‘0’ を返すことを想定する。デジタル化された検出器のヒット情報は開発する Zynq 搭載基板に到達し、位置情報・時間情報の取得を行うデジタイザを経由する。また、開発基板には ADC を搭載し、ASD でデジタル化 (‘1’ または ‘0’ の弁別) を行う前のアナログ信号の入力、波形のデジタル化も可能にする。これら 2 つの機能を図 1.8 ではデジタイザとして表記している。デジタイザを経由したデータは一時的に保存され、トリガーロジックを用いて取捨選択が行われた後に読み出しを行う。トリガーロジックには入力ヒット情報を用いた内部生成トリガー信号のほか、外部から開発基板に入力するトリガー信号を用いることを可能にする。以上の位置情報・時間情報のデジタル化、ADC からのデジタルデータの受信、トリガーロジックは Zynq PL 部の

FPGA に実装を行う。読み出されたデータは Zynq PL 部から PS 部へ転送され、PS 部の CPU を用いてデータファイル化を行う。PS 部は接続される Ethernet ケーブルを通して TCP/IP 通信を行い、外部からのデータファイルアクセスを可能にする。

前述したように、本研究において開発を行う読み出しシステムは高輝度 LHC アップグレードに伴い ATLAS 実験に新規導入予定であるワイヤーチェンバーの動作試験を一つの使用目的とする。ワイヤーチェンバーに飛来したミューオンが持つ位置情報・時間情報は検出器の正常な動作を確認する上で重要な指標となる。位置情報を取得することで想定しない挙動をするチャンネル位置の特定が可能になるほか、時間情報を用いて検出位置やケーブル長に依存するタイミングの調整も可能である。また、ADC を搭載しアナログ信号の波形取得も可能にすることで検出器からの信号を直接確認することも実現できる。FPGA を用いることでシステムとしての高い汎用性を持ち、中小規模程度の検出器実験読み出しシステムであれば本システムを用いて実現できる。これらのことを 1 つの基板と外部の解析用 PC で行うことが今研究における最大の目的である。

1.6　本論文の構成

第 2 章では FPGA 及び CPU の機能を説明した後、それらを兼ね備える SoC デバイス Zynq について説明する。第 3 章では本研究において開発を行った Zynq を搭載した新規基板について説明する。第 4 章では開発した新規基板上の Zynq に組み込みを行うシステムについて説明し、技術的な要素の検証を行ったことを述べる。これらを第 5 章でまとめる。

第 2 章

SoC デバイス Zynq

Zynq はデジタル集積回路の製造を手掛ける Xilinx 社により提供されている SoC デバイスである。使用者が任意の論理回路を実装できる FPGA と OS を搭載することで容易な外部との通信やユーザーアプリケーションの開発・実行が可能な CPU をワンチップ化したものであり、両者の利点により様々なアプリケーションへの応用が可能である。本章では FPGA 及び CPU の機能や利点と Zynq の構造や特徴について説明する。

2.1 FPGA

Field Programmable Gate Array (FPGA) は、使用者が開発した論理回路を実装できるデジタル集積回路である。実装する論理回路は揮発性である。任意の回路を繰り返し実装することができ、柔軟な開発が可能である。論理回路の機能の記述には Verilog や VHSIC Hardware Description Language といったハードウェア記述言語が用いられる。ハードウェア記述言語により作成された回路機能を FPGA に実装するには論理合成・配置配線・FPGA へのコンフィギュレーションが必要となる。以下では主に Xilinx 社製の FPGA についてそれらの工程を説明する。

- 論理合成
  論理合成とはハードウェア記述言語により作成された論理の最適化を行い、ハードウェア素子の構成へ変換を行うことである。FPGA の内部には Configurable Logic Block (CLB) と呼ばれる基本素子が多数存在している。CLB は主に組み合わせ回路の論理を実現する Look Up Table (LUT) と順序回路の論理を実現するフリップフロップ (FF) の 2 つにより構成されている。LUT は 2 進数で表される複数の入力の組み合わせに対応して出力の状態が確定する真理値表の役割を果たす回路である。LUT を複数使用することであらゆる組み合わせ回路の実現が可能である。FPGA 内の LUT は書き換え可能な揮発性メモリである SRAM で作られており、任意な LUT の格納や書き換えができる。フリップフロップは入力の状態を保持する素子であり、同期信号の出力や順序回路の実現に使用される。論理合成ではハードウェア記述言語により定められた論理回路を LUT やフリップフロップに変換し、CLB

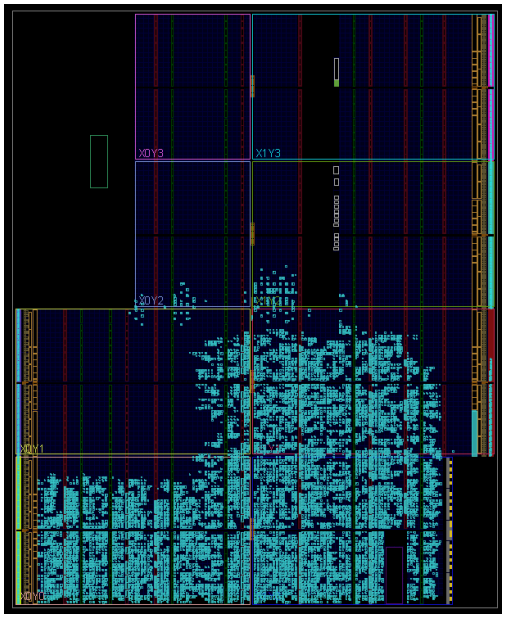


図 2.1: Vivado を用いて配置配線を行った際の様子

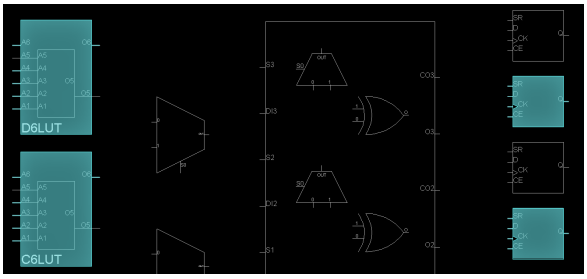


図 2.2: 配置配線に用いられる CLB の内部図。LUT やフリップフロップが確認できる。

の状態を決定する。

- 配置配線
  配置配線では各回路の電気的な接続や FPGA 内での物理的な実装位置の決定を行う。FPGA 内部は基本素子である CLB が規則正しく並んでおり、CLB 間を繋ぐための配線も用意されている。どの位置の CLB を論理回路の実装に用い、その間をいかに接続するかを決定するのが配置配線である。また、使用される FPGA ピンの決定が必要である。FPGA ピンには使用者が自由に割り当てが決定できるユーザー I/O ピンと、電源供給やグラウンドなど使用目的が定まっている専用ピンがあり、論理回路と外部との接続にどのユーザー I/O ピンを用いるかは使用者が決定する必要がある。ここで選択されたユーザー I/O ピンと論理回路実装に用いられる CLB との間も電気的な配線が行われる。Xilinx 社提供の Xilinx FPGA 設計ツールである Vivado を用いて配置配線を行った際の物理的な接続を図 2.1 に示す。水色の領域が実装に使用されている CLB の位置に該当する。また、実装に用いられる CLB の内部を図 2.2 に示す。左側の使用領域が LUT、右側の使用領域がフリップフロップである。

- コンフィギュレーション
  配置配線を行った回路情報から作成されるバイナリーファイルを用いて FPGA に論理回路の構築 (コンフィギュレーション) を行う。コンフィギュレーションは通常 Joint Test Action Group (JTAG) シリアル通信配線にて行われ、FPGA を構成する揮発性の SRAM に

データを書き込むことで回路情報の受け渡しが行われる。コンフィギュレーションにより構築された論理回路は電源を落とした際に消失してしまうため、電源を投入するたびにコンフィギュレーションが必要になる。ボード上の FPGA に付随するフラッシュメモリなどに回路情報を持つバイナリーファイルを格納しておくことで、電源を落とした場合にも基板上からコンフィギュレーションが行われ回路情報を保持することも可能である。

FPGA は使用用途は多岐に渡る。画像や音声のデータ処理や暗号解読などその使用用途は多岐にわたる。特定用途向けの集積回路である Application Specific Integrated Circuit (ASIC) のプロトタイプとして動作の確認に使用されることも多い。近年ではニューラルネットワークを用いた Artificial Intelligence (AI) モデルの FPGA 実装化も行われている。また第 1 章で述べたように、高エネルギー素粒子実験においても信号の処理や送受信、周辺機器操作に FPGA が使用されている。FPGA を用いたシステム開発の利点を以下に示す。

- 開発の柔軟性
  ユーザー任意の論理回路を実装することが可能であることに加え、論理回路の再構築が可能であるため汎用性が高い。素粒子実験において FPGA を用いる場合、取得するデータの取捨選択を決定するトリガーロジックの構築や、データ転送のケーブル長や検出位置などのカスタマイズが可能になる。また、ASIC に比べても、再構築可能な点から柔軟で汎用的な開発が可能である。

- 開発の簡易性
  メーカーにより提供される Intellectual Property (IP) の使用ができる。IP とは特定の機能を持つ回路のライブラリであり、IP を用いることで迅速な設計を可能にする [9]。IP のカスタマイズや自作論理回路との接続は図 2.3 に示すように GUI での操作も可能な場合も多い。図 2.3 では中央の自作ロジックに対して、FPGA 内部での容易なクロック調整を可能にする Clocking Wizard IP[10] からクロックを供給し、接続先の PC から内部ロジックの確認ができるロジックアナライザである Integrated Logic Analyzer (ILA) IP[11] を用いたロジックである。また、ハードウェアでありながらソフトウェア開発にも近い感覚で実装が可能であるため実装が比較的容易である。

- 大容量・高速なデータ通信
  FPGA は並列処理回路の構築が可能であるため、処理速度に優れている。そのため大容量で高速なデータ通信が可能である。Xilinx 社製の FPGA は消費電力とジッターを抑えた高速シリアル通信規格でのトランシーバーが搭載されており、効率良いリソース・電力消費で実現できる [12]。

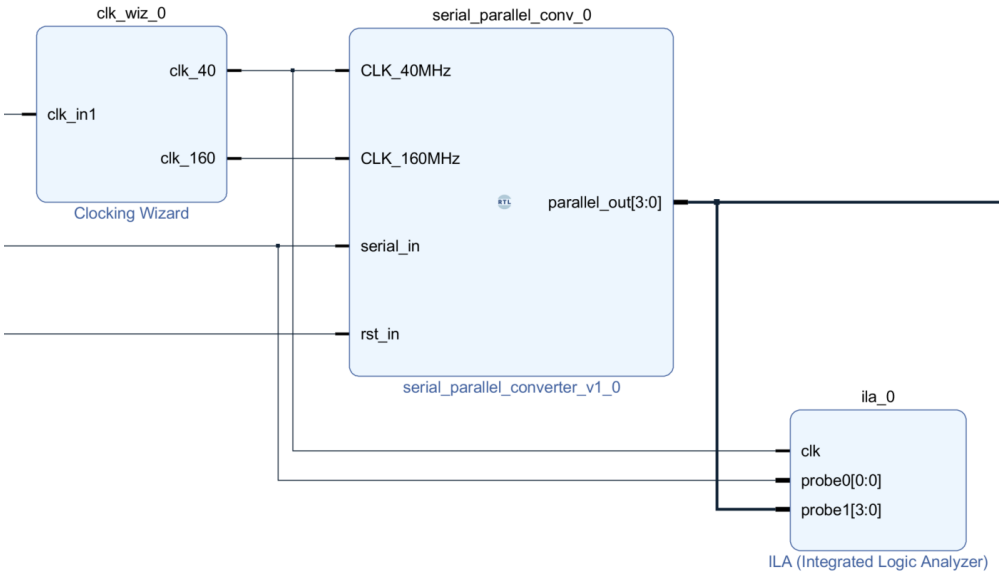


図 2.3: Vivado の GUI 上で IP を用いて設計を行う様子

2.2　CPU

Central Processing Unit (CPU) は主記憶装置上から順に読み込んだ命令の解読・実行を行うことでデータの演算を行う装置である。図 2.4 に CPU を用いた処理の流れを示す。演算対象であるデータは入力装置を介して主記憶装置に格納される。主記憶装置上には命令として演算を行うプログラムの実行内容も格納される。主記憶装置上のデータや命令にはそれぞれアドレスが割り振られている。プログラムの実行の際には命令アドレスに従って CPU 内のフェッチユニットが主記憶装置から命令を取り出す。取り出された命令はデコーダーによって解読される。このことからフェッチユニットとデコーダーは制御装置と呼ばれる。デコーダーにより解読された命令は制御情報として Arithmetic and Logic Unit (ALU) に送信される。ALU とは、算術演算 (四則演算) や論理演算などの計算を行う装置である。実行される命令内容にデータが必要であれば主記憶装置上から格納されているデータを取り出し、演算結果を主記憶装置に送信する。主記憶装置に送信された結果は出力装置を介して出力される。キャッシュメモリとは、主記憶装置とは異なり CPU 内部に配置されるメモリである。主記憶装置よりもアクセスがしやすいため、アクセスする頻度の高いデータや命令の保存に使われる。

- OS の搭載

  CPU の使用に関する重要な要素として Operation System (OS) の搭載が挙げられる。OS とは CPU のオペレーションを司るシステムソフトウェアである。他のアプリケーションプログラムと同様に、主記憶装置に保存される。OS はユーザーやアプリケーションプログラムとハードウェアの中間に位置し、ユーザーやアプリケーションプログラムに対して

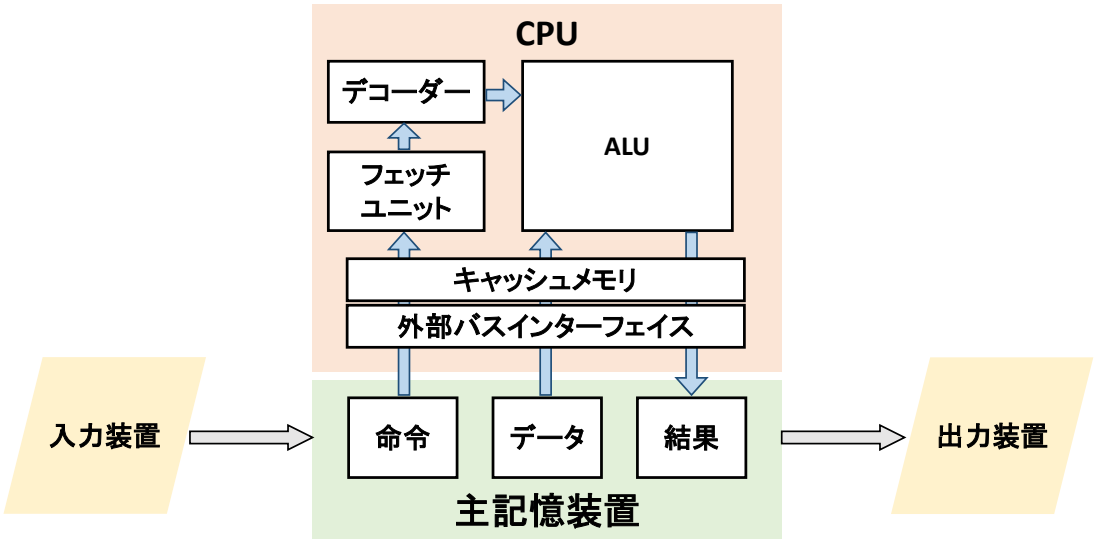


図 2.4: CPU を用いた処理の流れ

標準的なインターフェイスを提供すると同時に、ハードウェアの管理も効率的に行うことが可能である。また、ファイルシステムなどの補助記憶装置の管理や TCP/IP 通信などのネットワーク通信管理も OS の機能として挙げられる。一般的な OS として、Windows や MacOS、UNIX などが知られている。UNIX とは 1960 年代に AT&T 社により開発された OS である [13]。近年では UNIX そのものが使用されることは少なく、UNIX システムを源流として改良された UNIX 系 OS として使用されている。その代表が Linux である。Linux は厳密には OS そのものではなく OS 中枢部を担うカーネルのことを指す。カーネルとはファームウェア・デバイスドライバの情報を収集しシステム上のハードウェアの基本的制御を行うものであり、プログラムのメモリアクセスを管理し、どのプログラムがどのハードウェアへアクセスするかを決定する。、ファームウェアやデバイスドライバのカスタマイズを行うことで、用いられる環境に適したカーネルの構築が可能である。カーネルを元に構築された OS は Linux ディストリビューション (Linux OS) と呼ばれる。本論文ではカーネルを指す場合は Linux kernel、OS そのものを指す場合は Linux OS と表記する。

2.3　System on Chip デバイス Zynq-7000

Zynq-7000 シリーズ (以下 Zynq とする) は Xilinx 社により提供される System on Chip (SoC) であり、FPGA と CPU をワンチップ化したデバイスである [8]。任意の論理回路を構築できる FPGA と高度なデータ制御や OS の搭載が可能な CPU は Advanced eXtensible Interface (AXI) インターコネクトにより接続されている。Zynq は両者の汎用性と簡易性を活かしながら手軽に利用することができるデバイスであり、近年尽力されている Internet of Things (IoT) の発展にも貢献するものである。実際に Zynq は自動化運転技術や無線機器等への Edge Computing デバイスとして使用されている [8]。本節では Zynq の構造や機能について説明する。

2.3.1　Zynq の構成

Zynq は FPGA を担う Programmable Logic (PL) 部と、CPU を中心としたプロセッサである Processing System (PS) 部によって構成されている。図 2.5 に Zynq の構造図を示す [8]。構造図下部の黄色の部分は PL 部であり、上部の水色の部分が PS 部である。以下でそれぞれの詳細を説明する。

- Programmable Logic 部
  Programmable Logic (PL) 部は FPGA により構成される。FPGA は Xilinx 社製 7 シリーズ FPG に相当するものであり、比較的低価格な Zynq には Artix-7、高性能な Zynq には Kintex-7 に相当する FPGA が組み込まれている。FPGA 内部の Look Up Table の数やフリップフロップの総数はデバイスによって依存し、それぞれ 14,400〜444,000、28,800〜554,800 の幅がある [14]。本研究で使用する Zynq は中程度の性能に位置する XC7Z030-2FFG676I であり、内部 LUT 総数は 78,600、フリップフロップ総数は 157,200 である。使用する Zynq の性能に関する詳細は第 3.2.1 節で述べる。

- Processing System 部
  Processing System (PS) 部の中心を担うのは CPU である。プロセッサコアとして ARM Cortex-A9 ベースの CPU が 2 つ搭載されており、周辺機器インターフェイスやメモリコントローラへのアクセス、制御が可能である。プロセッサコアの動作周波数はデバイスにより異なるが性能の高いものでは最大 1 GHz である。PS 部には Multiplexed I/O (MIO) pin が付随している。MIO は PS 部と外部機器が直接接続されるピンであり、PS 内部に配置されている Serial Peripheral Interface (SPI) 通信や Ethernet 通信、Universal Asynchronous Receiver/Transmitter (UART) 通信などの各種インターフェイス用のペリフェラルを経由してプロセッサコアと接続される。これによりソフトウェアアプリケーションを用いたプロセッサコアの指令に応じて MIO 経由で接続されている周辺機器との通信を実現できる。また、同様の信号を MIO を介せずに PL 部のユーザー

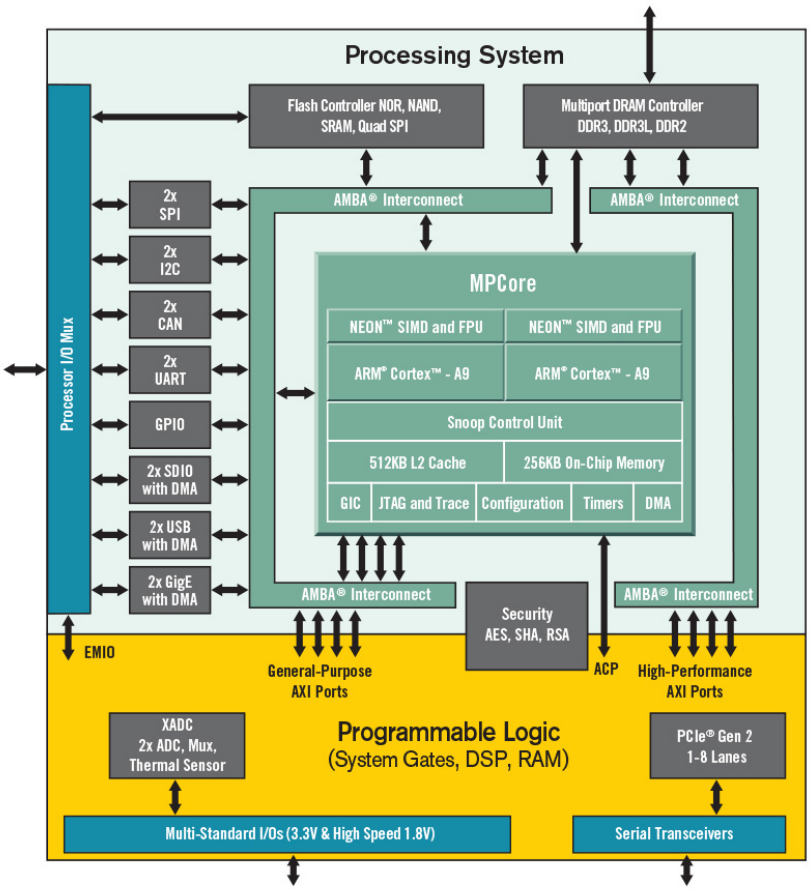


図 2.5: Zynq の構成図。上部水色部分が CPU を中心とする PS 部、下部黄色部分が FPGA を中心とする PL 部である [8]。

I/O ピンから出力する Extended MIO (EMIO) という機能も備わっており柔軟な設計が可能である。さらに外部メモリとの通信のために 2 種の外部メモリコントローラが存在する。1 つは Flash Controller であり、MIO を経由して接続することで使用可能である。接続先には主に Quad-SPI memory が採用され、ブートファイルの格納をしておくことで Zynq の電源投入時にブートファイルを参照し、起動することが可能である。もう 1 つは Multiport DRAM Controller であり、PS 部に付随する専用のピンから信号の入出力が可能である。接続先は主に DDR3 memory であり、主記憶装置として CPU のデータ入出力や命令の格納、ソフトウェアプログラムの実行、保存に使用する。

- PS 部-PL 部間接続

  PS 部と PL 部は 3 種類の内部接続ポートを用いて接続され通信が可能である。以下に 3 種類の内部接続ポートを示す。

  – General-Purpose (GP) Port

    PS 部内部 CPU と PL 部を接続するポート。メモリマップド I/O を用いた PS 部からの PL 部内ロジック制御や比較的少量なデータのやり取りに使用される。PL 部のロジックは典型的に PS 部からのリクエストを受けて動く設計が一般的であるため、Zynq を用いた開発時はデフォルトでこの GP ポートが有効になっている。

  – High-Performance (HP) Port

    PS 部のメモリコントローラと PL 部を接続するポート。PS 部はメモリコントローラを介して外部の主記憶装置 (DDR) に接続されている。PL 部から主記憶装置への高速アクセスを可能にし、高速な計算回路を設計する場合や PL 部が大量の記憶素領域を必要とする場合に用いる。AXI Direct Memory Access という機能を用いることで、PL 部で取得したデータや処理結果を DDR に転送し、PS 部からのアクセスが可能になる。

  – Accelerator Coherency Port (ACP)

    PS 部の L2 キャッシュメモリコントローラと PL 部を接続するポート。キャッシュメモリとは主記憶装置 (メインメモリ) とは異なり、CPU 内部に配置されるメモリである。CPU 側からは主記憶装置よりもアクセスがしやすいため、アクセスする頻度の高いデータや命令の保存に使われる。Zynq PS 部の CPU には 2 つのキャッシュメモリが存在し、容量よりも速度が求められるキャッシュメモリを L1 キャッシュメモリ、速度よりも容量が求められる方を L2 キャッシュメモリとして用いている。ACP を用いたメモリの読み書きは L2 キャッシュとの動作速度の一貫性 (コヒーレンス) が維持された状態で行われるが、その一方で通信帯域には制限が生じる。PS 部-PL 部間でより厳密な通信を行うときに使用される。

ここで PS 部-PL 部間の通信に用いられる AXI 通信について説明する。AXI とは Advanced Microcontroller Bus Architecture (AMBA) と呼ばれる ARM 社が定めたバス規格のうちの一部であり、Advanced eXtensible Interface の略称である [15]。Zynq の PL 部-PS 部接続のインターフェイスは AXI プロトコルに統一されている。AXI の基本的なプロトコルは送信側からの valid 信号と受信側からの ready 信号のハンドシェイク型通信である。送信側は信号の送信準備ができたときに valid 信号を立ち上げる。一方受信側も受信準備ができたときに ready 信号を立ち上げる。これら 2 つの信号が同時に立ち上がった際に data 信号の送受信が行われる。AXI 通信には用途に合わせて以下の 3 種類のプロトコルが用意されている。

– AXI4(フルスペック)
バースト転送が可能なメモリマップ方式のインターフェイス用のプロトコルである。バースト転送とは、データ転送ごとにアドレス情報を付加する通常転送とは異なり、1 度のアドレス情報を基にして複数のデータ転送を行うことである。フルスペックの AXI4 通信では最大 256 のサイクル分のバースト転送が可能である。データ幅は 32 bit、64 bit、256 bit、1024 bit、2048 bit から選択できる。大容量なデータを高速で転送できるため、Zynq においては PL 部-PS 部間の大規模なデータ転送である Direct Memory Access 転送に使用される。

– AXI4-Lite
フルスペックとは異なり、バースト転送をサポートしないメモリマップ方式のインターフェイス用プロトコルのである。フルスペックの AXI 通信のような大容量の転送には不向きであるが、軽量でシンプルな実装が可能である。Zynq においては PL 部ロジックの制御レジスタを PS 部より制御する際に使用される。

– AXI4-Stream
アドレス情報を用いず、バースト転送の制約を持たないプロトコルである。上述した ready 信号、valid 信号のほかに送信側からの転送データの終端を示す last 信号が用いられる。Zynq においては PL 部内の IP コア間の通信に使用される。

2.3.2　Zynq の機能

Zynq は複合デバイスである特徴から様々な機能が備わっている。本節では、本研究における Zynq の役割を述べる中で重要となる機能について説明する。

- ブートモード選択
電源投入時に Zynq のコンフィギュレーションを行うブートモードの選択が可能である。Zynq は電源投入時に PS 部の MIO ピンの状態を確認し、ブートモードを選択する。そのため Zynq 搭載モジュールでは基板上で PS 部 MIO ピンに接続されているジャンパピンや DIP スイッチの切り替えにより簡単にブートモードの選択が可能である。ブートモードは大きく以下の 2 種類に分けられる。

  – PS マスターブートモード
  フラッシュメモリを用いたブートである。フラッシュメモリに 3 種類のファイルを格納しておき、PS 部がファイルを参照することで起動時の状態を決定する。3 種類のブートファイルを以下で説明する。

① Initial Boot ファイル： PS が最初に読むブートファイル (BOOT.BIN)。PS の状態決定や設定をおこなう First Stage Boot Loader (FSBL) ファイル (zynq_fsbl.elf)、PL 部のファームウェア情報 (system.bit)、Linux OS の起動に必要なブートローダ (u-boot.elf) をまとめて 1 つのファイルとして格納。PS 部、PL 部の設定を行い、③を読んだ際の制御を行うためにブートローダ (u-boot) を PS 部上に構築する。

② スクリプトファイル： u-boot による制御が行われる際に使用されるファイル (boot.scr)。u-boot コマンドを 1 行 1 コマンドで記述したテキストファイルにヘッダー情報が付加されている。

③ Second Boot ファイル：PS が 2 番目に読むブートファイル (image.ub)。①による u-boot 制御下で、デバイスツリー (system.dtb)、Linux kernel (uImage)、Root File System(rootfs.ext4.gz) の構築が行われる。①と同様、1 つのファイルに複数のファイルの情報がまとめられている。デバイスツリーとは OS 側からアクセスが可能なハードウェア周辺機器について記述した情報である。Root File System とは一般にコンピュータファイルシステムにおけるファイル階層の最上位層のことを指し、Linux 上で使用したいライブラリやパッケージを設定しコンパイルを行うことで使用する Linux kernel をカスタマイズすることが可能である。

これらのブートファイル及びそれぞれに組み込まれているファイルは Xilinx 社によるクロスコンパイルツールである PetaLinux を用いて開発を行うことができる [16]。使用するデバイスや用途により起動に必要な情報が異なるため、PetaLinux 上で設定を行い、コンパイルを行う必要がある。ブートファイル作成に関する詳細は第 4.1.1 節で述べる。作成されたブートファイルはフラッシュメモリに格納することで Zynq からのアクセスが可能になる。フラッシュメモリは一般的に SD カードや Quad-SPI メモリが採用さる。どちらのフラッシュメモリを参照するのかの決定も PS 部 MIO ピンの状態で決定できるため、ボード上のジャンパピンや DIP スイッチを用いて用意に決定できる。

– JTAG スレーブブートモード
外部 PC からの JTAG 通信を用いたブートである。Zynq は JTAG 通信のスレーブとして動作する。図 2.6 に示すように JTAG スレーブブートモードにはカスケードモードと独立モードの 2 種類がある [17]。Zynq 内部には 2 つの JTAG コントローラー (Test Access Port/Debug Access Port) が存在する。Test Access Port (TAP) は PL 部のコンフィギュレーションプロセスと PL 内部機能の制御ができる。Debug Ac-

cess Port (DAP) は PS 部に存在する JTAG 用のコントローラーである。2 つの JTAG スレーブブートモードはこれら JTAG コントローラーへのアクセスが異なる。
より一般的であるカスケードモードは、PL に付随する JTAG インターフェイスピンを介して、デイジーチェインの状態で TAP、DAP の両方へアクセスが行われ JTAG 通信が行われる。一方で、独立モードでは 2 つの JTAG コントローラーに対して別々のアクセスが行われる。独立モードでの JTAG 通信について説明する。まず、通常の JTAG インターフェイスピンを介して TAP へのアクセスが行われ、Zynq はまず PL 部のコンフィギュレーションを行う。このコンフィギュレーションで書き込まれる PL 部の回路は独立モードが可能である設定が必要である。独立モードでの JTAG 通信は Zynq の機能である Extended MIO (EMIO) の使用が必須である。EMIO は前述したように PS 内部と PL 部に付随するピンを接続する機能であり、これを用いて設定を行った PL 部付随ピンから DAP へのアクセスが可能になる。この独立モードは PL 部 PS 部それぞれでのデザイン決定が可能なため動作確認用のデバッグが容易になる。カスケードモードと独立モードのどちらの JTAG スレーブモードを選択するかは PS 部 MIO ピンの状態によって変わるため、この選択もボード上のジャンパピンや DIP スイッチを用いて容易に決定できる。

- 3 種類の PL 部 I/O
  Zynq PL 部のユーザー I/O は 3 種に分別され、それぞれ別のバンクに属している。バンクとは電源電圧を共有する I/O ピンの単位であり、同一バンク内の I/O ピンは FPGA 内での距離が近いため信号間のタイミングの差も小さい。そのため同じ対象の IC に対して機能する信号は同一バンクにまとめられることが推奨される。以下では本研究で開発をおこなう ASD Readout NIM Module に搭載する Zynq (XC7Z030-2FFG676I) に付随する PL 部 I/O について述べる。

  PL 部 I/O のバンクは High-Range (HR) バンク、High-Performance (HP) バンク、GTX バンクに分かれる。HR バンクは 1.2V〜3.3V の動作電圧範囲を持つバンクである。XC7Z030-2FFG676I では 2 つのバンク (BANK12,13) が該当し、計 100 本の I/O ピンを持つ。動作電圧範囲が広いため対応する規格の種類が多く、汎用的な使い方が可能である。HP バンクは 1.2V〜1.8V の動作電圧範囲を持つバンクである。XC7Z030-2FFG676I では 3 つのバンク (BANK33,34,35) が該当し、計 150 本の I/O ピンを持つ。HP バンクは高速な通信が可能であり、高速動作な DDR メモリとの通信を使用用途として持つ。GTX バンクは一般的なユーザー I/O とは異なり、第 2.1 節で述べたような並列処理を用いた低消費電力での大容量・高速な通信に用いられる。XC7Z030-2FFG676I では BANK112 が該当し、差動信号 4 チャンネル分の GTX 通信が可能である。

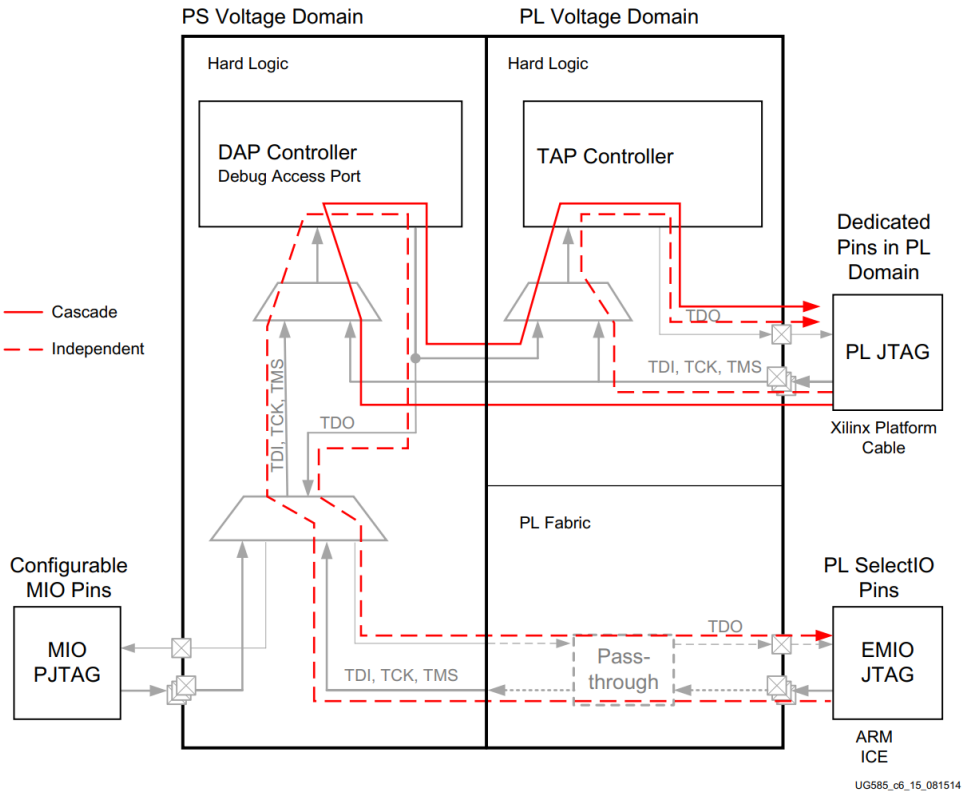


図 2.6: 2 種類の JTAG ブートモード [17]

- Linux OS を用いた MIO 操作
  Zynq は内部 CPU を用いた周辺機器操作が可能である。CPU と MIO は各種インターフェイス用のコントローラーを介して接続されているため、CPU によるアプリケーションの実行により MIO から出力される信号の操作が可能である。また、CPU では Linux OS を走らせることが可能である。Linux kernel に標準搭載されているデバイスドライバをすることで、よりユーザーフレンドリーな周辺機器操作環境を構築できる。なおデバイスドライバの設定にはデバイスツリーの編集が必要である。デバイスツリーとは OS 側からアクセスが可能なハードウェア周辺機器について記述した情報であり、クロスコンパイルツールである PetaLinux で編集が可能である。また、周辺機器操作だけではなく、MIO を経由する UART 通信を用いることで UART 通信先からの Zynq CPU のコマンド操作やソフトウェアアプリケーションの開発などが可能になる。

- AXI Directory Memory Access による PL 部-PS 部間データ転送
  Zynq の PL 部と PS 部は AXI インターコネクトにより接続されているため、データのやり取りが可能である。Xilinx 社より提供される AXI Direct Memory Access IP を用いると PS 部に付随している DDR メモリを介して送受信が可能である。これを利用することで、

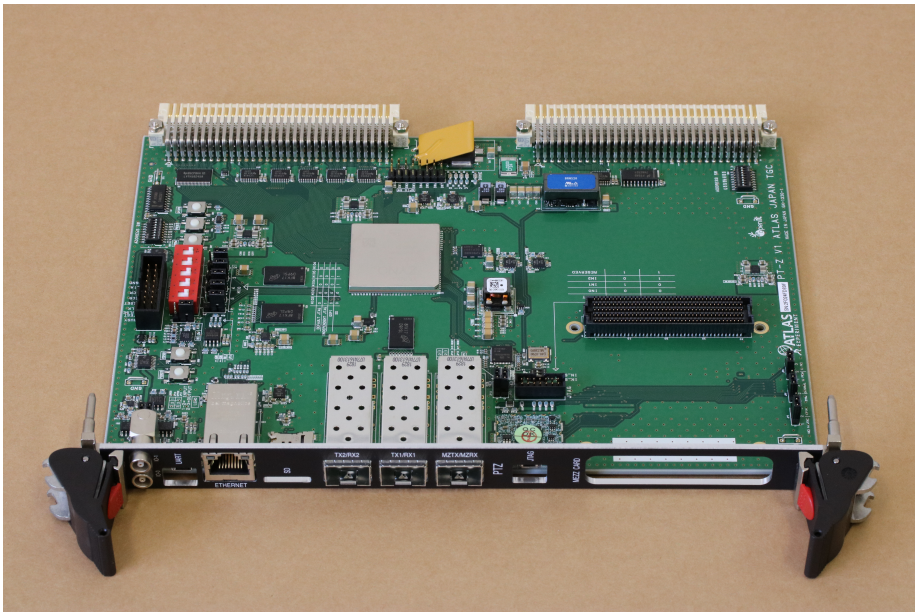


図 2.7: Zynq 搭載汎用モジュール PT-Z

PL 部で取得したデータや内部ロジックを用いて出力した結果を PS 部付随の DDR に転送することができる。

2.3.3　高エネルギー実験における Zynq の使用例

Zynq は 2011 年に Xilinx 社によって発表されてからその利便性や汎用性が評価され各種システムへ採用されてきた。近年では高エネルギー実験においてもその応用化が進み、エレクトロニクスのアップデートに大きく貢献し、高エネルギー実験への参加者にも身近な存在になっている。その例として LHC-ATLAS 実験において開発された 2 つの Zynq 搭載ボードについて紹介する。

汎用モジュール PT-Z

図 2.7 に示す汎用モジュール PT-Z は、LHC-ATLAS 実験における Zynq の応用に向けてプロトタイプ (PT) として開発された VME モジュールである [18]。Zynq-7000 シリーズである XC7Z045-2FFG900I を搭載しており、ATLAS Thin Gap Chamber (TGC) を用いたエンドキャップミューオンシステムにおける Zynq の有用性の検証と性能評価試験を行うために様々な機能が搭載されている。USB-UART 通信モジュールは PS 部上 Linux OS のコンソール表示に用いられる。これにより外部 PC から Zynq 上の Linux OS にアクセスし、アプリケーションの開発やコマンド操作が可能になる。また、Ethernet モジュールを搭載し、LAN ケーブルを用いたインターネット接続が確立されている。さらに、GTX 対応 SFP+ 規格光ファイバートランシーバーモジュールは、Zynq PL 部の GTX 機能用いて外部との光通信が可能な機能である。光通信は ATLAS 実験のエンドキャップミューオンシステムでも使用される通信であり Zynq を用いた

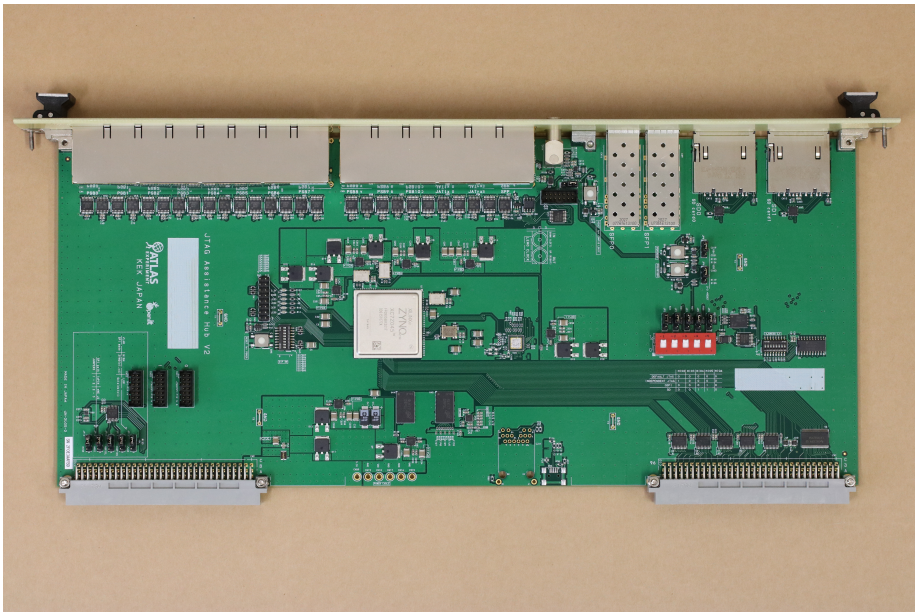


図 2.8: Zynq 搭載エレクトロニクス制御回路 JATHub

光通信の回路を実現した。また、FMC (FPGA Mezzanine Card) コネクタを搭載し、PT-Z にメザニンとして接続される別ボード上 FPGA との信号送受信や JTAG 通信を用いたコンフィギュレーションも可能にしている。そのほか、Zynq 搭載基板を開発する上で必要となりうる要素を搭載している。

エレクトロニクス制御回路 JATHub

図 2.8 に示すエレクトロニクス制御回路 JATHub は同じく LHC-ATLAS 実験のエンドキャップミューオンシステムにて、高輝度化アップデートに伴い新規導入される VME モジュールである [19]。PT-Z と同じく XC7Z045-2FFG900I を搭載している。エンドキャップミューオンシステムにおいて、フロントエンド回路ではヒット情報の収集・送信に FPGA が用いられている。この FPGA は物理的にアクセスしづらい場所に配置されているため、その近くからコンフィギュレーションやデバッグ、再コンフィギュレーション用のメモリアクセスを行うシステムが必要である。さらにフロントエンドは厳しい放射線環境下である。放射線環境下での FPGA の動作は Single Event Upset (SEU) による損傷によって保障されていない。この問題を解決するために高輝度化に伴い導入されるのが JATHub である。Zynq を FPGA の制御系統の中継点として用いることで PS 部上で走る Linux OS を用いたソフトウェアアプリケーションを用いて容易にデータ収集・送信用の FPGA のスローコントロールが可能になり、異常を感知した際には再コンフィギュレーションを行う機能を搭載している。

第 3 章

ASD Readout NIM Module の開発

ワイヤーチェンバー読み出しシステムの中枢として Nuclear Instrument Modules (NIM) 規格である ASD Readout NIM Module の設計開発を行った。ASD Readout NIM Module には FPGA と CPU プロセッサを組み合わせた Xilinx 社製の Zynq SoC をメインチップとして採用する。Zynq は FPGA の機能を担う PL 部と、CPU プロセッサの機能を担う PS 部で構成されており、ワイヤーチェンバーの動作試験時に読出し信号の時間情報や波高情報を取得する機能だけではなく、PS 部上で動作するソフトウェアを利用した周辺機器操作や取得データの解析、TCP/IP 通信を用いた外部 PC との通信など、1 つの基板上で試験を実現するという利便性・簡易性を求める上で最適であると判断した。本章では ASD Readout NIM Module に搭載する要素や機能を中心に、ハードウェアとしての概念を説明する。

3.1 ASD Readout NIM Module の役割

ASD Readout NIM Module の使用は図 3.1 のような状況を想定したものである。検出器からの信号は Amplifier-Shaper-Discriminator (ASD) IC を経由し、ASD Readout NIM Module に入力させる。検出器から出力された電流信号は、ASD IC により電圧信号への変換、増幅、整形、閾値電圧との比較による信号の弁別が行われ、Low Voltage Differential Signaling(LVDS) 規格の信号として ASD Readout NIM Module に到達することを想定している。

- LHC-ATLAS 実験用 Amplifier-Shaper-Discriminator (ASD) IC
  開発後に動作試験を行う LHC-ATLAS 実験における新規ワイヤーチェンバーに付随している ASD IC の回路図を図 3.2 に示す [20]。16 ns の時定数を持つ前段増幅器回路 (電圧変換効率は最大で 0.8 V/pC)、利得 7 倍の差動電圧増幅回路、外部より設定可能である閾値電圧とのコンパレータ回路から成り立っている。コンパレータ回路により、差動電圧増幅回路からの出力が閾値電圧を超過している時間をパルス幅とする LVDS 規格のデジタル信号が出力されるようになっている。また、外部より入力される信号 (Test Pulse Trigger) がハイパスフィルタを経由し前段増幅器回路に接続されており、疑似的な Hit 信号 (Test Pulse)

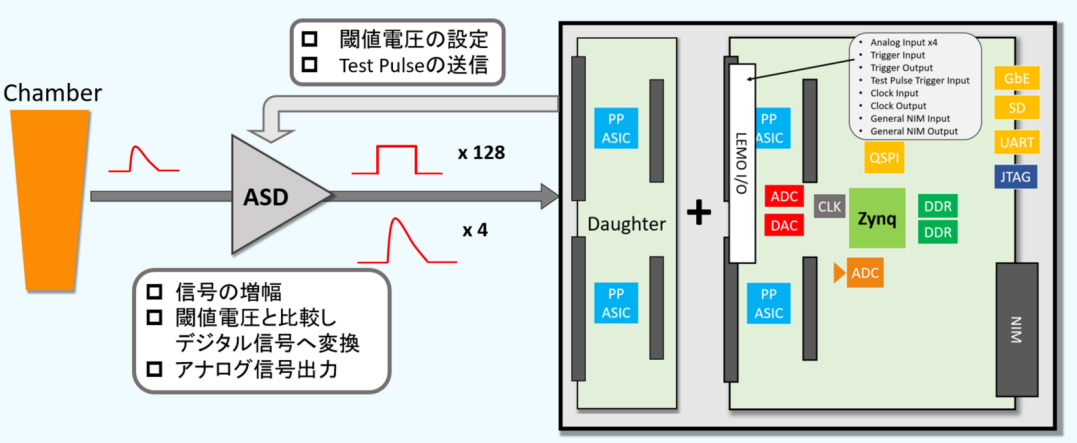


図 3.1: 想定する ASD Readout NIM Module の使用環境と部品配置図

を出力する機能が備わっているため、各検出器からの信号の時間較正が可能である。1 つの ASD IC は 4 チャンネル分の入力に対応しており、LHC-ATLAS 実験では ASD IC を 4 つ搭載した ASD Board を用いているため、全 16 チャンネルの信号に対し増幅、整形、閾値電圧との比較を行っている。さらに、16 チャンネルのうちの 1 チャンネルのみを対象に、LVDS 信号へ変換を行う前のアナログ信号を、付随している LEMO コネクタを経由して出力する機能が備わっている。

ASD Readout NIM Module の主な役割を以下に示す。

- 128 チャンネルの LVDS 信号受信
  - Hit 位置の特定
  - Hit の有無の確認
  - Hit の時間情報の取得
- 4 チャンネルのアナログ信号波形取得

主となる役割は LVDS 信号受信である。その接続チャンネル数は Zynq のピン数による制限から 128 と定めた。接続チャンネル数と基板の大きさや NIM ラックの幅サイズを考慮に入れ、サブ (ドーター) ボードの導入を決めた。メインボードとサブボードはコネクタにより接続を行い、どちらの基板とも入力されるデータ信号は Patch-Panel ASIC (詳細は第 3.2.2 節) を経由し Zynq の PL 部へ入力される。Zynq PL 部上では取得したデータの処理を行い、ASD からの Hit 信号の有無と Hit があった場合の位置情報・時間情報を取得する (詳細は第 3.2.3 節)。

さらに、4 チャンネルのアナログ信号の波形取得を可能にする (詳細は第 3.2.4 節)。LEMO コネクタを経由するアナログ信号を本ボード上の Analog to Digital Converter でデジタル信号に変換した後 Zynq へ入力する。ADC からの値を読み、アナログ信号の波形を取得する。

前段の ASD に対し ASD Readout NIM Module からは閾値電圧の設定を行うことが可能である。基板上に Digital to Analog Converter (DAC) を搭載し、Zynq 上で設定した電圧を ASD に

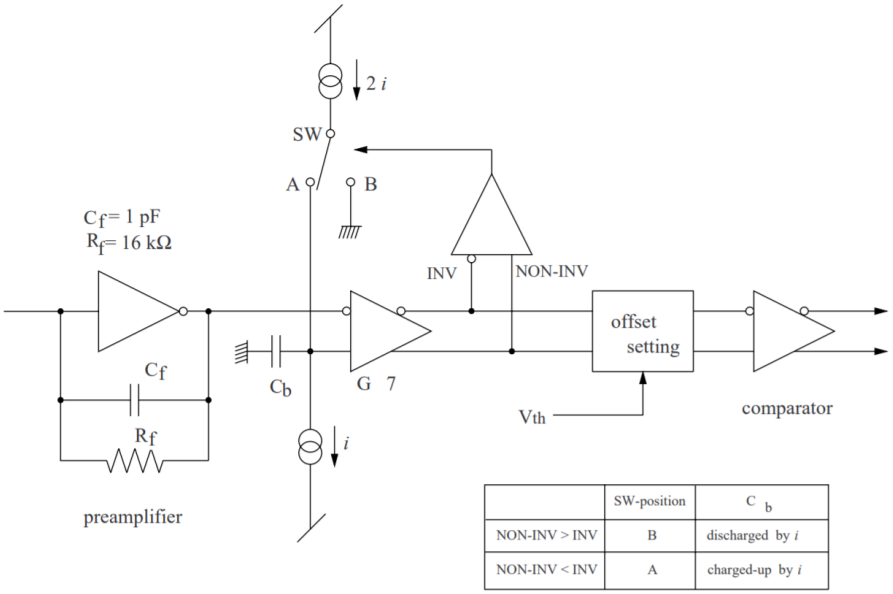


図 3.2: LHC-ATLAS 実験における ASD IC の回路図 [20]

向けて送信できる。また、時間較正を目的とした疑似的な Hit 信号 (Test Pulse) を発行させる Test Pulse Trigger を ASD に向けて出力することも可能である。

ASD とのデータ信号・閾値電圧・Test Pulse Trigger・ASD 駆動用アナログ電源 (+3V、-3.3V) のやり取りにはフラットケーブルの使用を想定し、ASD Readout NIM Module 上にはケル株式会社製の 82 pin コネクタ 8831E-080-170L を搭載する。また、メインボードとサブボードの接続にはケル株式会社製の 82 pin コネクタ 8901-080-178S-C-F を用いる。検出器の動作試験を行う上で、Zynq へのヒット信号到達タイミングが統一されていることが要求される。そのため、メインボードとサブボード間のコネクタを介する信号にはコネクタ間の配線を含め等長配線設計された。

3.2　要素と機能

ASD Readout NIM Module に搭載する IC・モジュールや Zynq に実装する機能を紹介する。

3.2.1　Zynq SoC

ASD Readout NIM Module には Xilinx 社製 Zynq-7000 シリーズ XC7Z030-2FFG676I をメインチップとして搭載する。総ピン数は 676 本である。各ピンの本数、及び詳細なスペックについては表 3.1 に示す。XC7Z030-2FFG676I は Xilinx 社製 FPGA Kintex-7 相当の PL 部 FPGA と 2 つの ARM Coretex-A9 を持つ PS 部で構成される。Patch-Panel ASIC(第 3.2.2 節) からのデータ時間情報取得 (第 3.2.3 節) やその制御、PL 部高速通信を用いたアナログ信号波形取得 (第 3.2.4

パッケージ名	XC7Z030-2FFG676I	
サイズ	27 mm x 27 mm	
ピン本数	PS I/O	128
	PL I/O	HR 100
		HP 150
		GTX 4
PS	Dual-Core ARM Cortex-A9 MPCore	
PL	搭載 FPGA	Kintex-7
	ロジックセル	125K
	LUTs	78,600
	フリップフロップ	157,200
	BRAM	9.3 Mb

表 3.1: XC7Z030-2FFG676I のピン本数及びスペック詳細。[14]
High-Range (HR) pin と High-Performance (HP) pin は それぞれ動作電圧 1.2V〜3.3V、1.2V〜1.8V で動作する PL I/O。

節)、その他周辺機器操作の制御やデータ送受信 (第 3.2.5 節) はこの Zynq が中心となって行う。

3.2.2　LHC-ATLAS 実験 TGC 用 Patch-Panel ASIC

ASD Readout NIM Module には高輝度 LHC-ATLAS 実験エンドキャップミューオントリガーシステム用に開発された Patch-Panel ASIC をメインボードに 2 つ、サブボードに 2 つ搭載する。Patch-Panel ASIC は LHC-ATLAS 実験においてミューオントリガーに使用される Thin Gap Chamber(TGC) で使用されている [21]。図 3.3 に Patch-Panel ASIC のブロック図を示す。Patch-Panel ASIC は、高輝度 LHC-ATLAS 実験においてミューオントリガーシステムで使用される ASIC である。高輝度 LHC-ATLAS 実験おいては、ASD ボードから送られてくる LVDS 信号に対し、ミューオン飛程時間やケーブル長に依存する遅延時間の較正、陽子バンチ衝突タイミングを示す LHC の 40MHz クロックとの同期を行い、1.8V の CMOS 信号として FPGA にデータ送信を行う。遅延時間の較正には内部可変遅延回路・Phase Lock Loop (PLL) 回路が、LHC クロックとの同期には Bunch Cross IDentification (BCID) 回路が使用される。また、第 3.1 節内で説明した LHC-ATLAS 実験用 ASD IC に対し Test Pulse を供給するためのテストパルスジェネレータが内蔵されており、FPGA (Zynq) から Test Pulse Trigger 信号を受け取ると特定の幅の差動矩形パルスを出力する機能が備わっている。

Patch-Panel ASIC の制御は Serial Peripheral Interface (SPI) プロトコルによる通信によって行われる。ここで SPI プロトコル通信について述べる。SPI プロトコル通信は信号の設定を行う

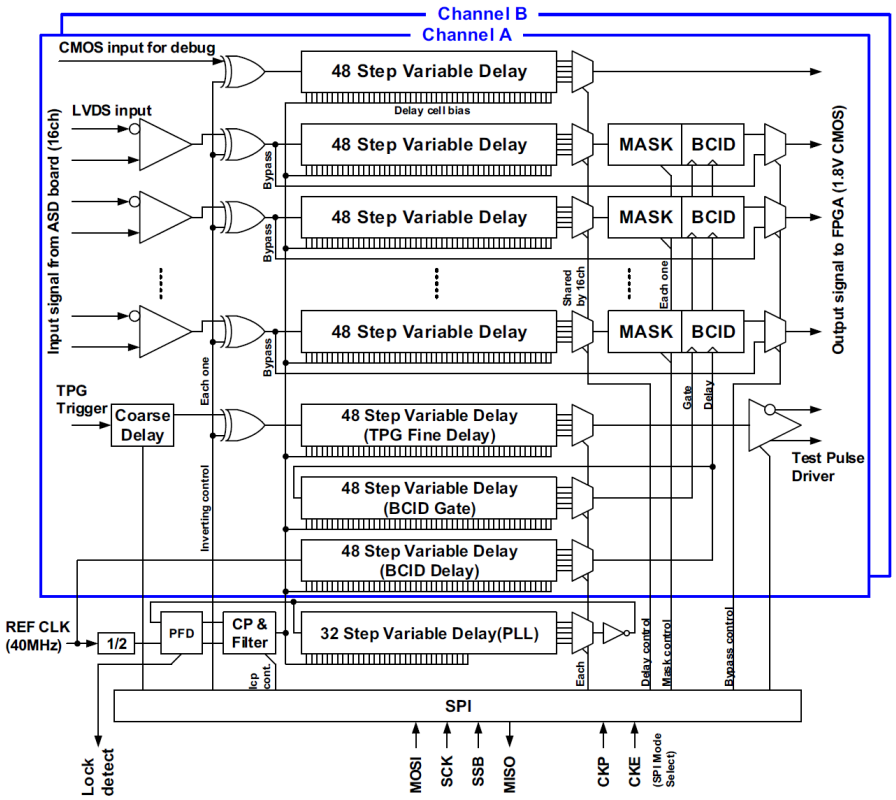


図 3.3: Patch-Panel ASIC のブロック図 [21]

マスターデバイス及び設定されるスレーブデバイスを、Slave Select (SS)、Serial Clock (SCK)、Master Out Slave In (MOSI)、Master In Slave Out (MISO) の 4 本の信号線により接続される。図 3.4 に SPI 通信のタイミングチャートを、図 3.5 に 8 ビット送受信を行う際の一般的なハードウェア構成を示す [22]。Patch-Panel ASIC の場合は CPOL=0, CPHA=0 で動作をさせる予定である。Patch-Panel ASIC は SPI 通信のスレーブデバイスとして働き、内部の 224 ビットのレジスタの状態を決めることで制御が行われる。ASD Readout NIM Module においては Zynq の PS MIO を用いて PS 上 Linux OS でソフトウェアプログラムを走らせることで SPI 通信を行う予定である。

Patch-Panel ASIC には、バイパス機能が備わっている。この機能を使用すると、入力信号を可変遅延回路や BCID 回路を経由せずに 1.8V COMS 信号として送信することができる。一般に回路のテスト用に用いられる機能ではあるが、今回の使用用途では検出器自身の試 Patch-Panel ASIC には、バイパス機能が備わっている。この機能を使用すると、入力信号を可変遅延回路や BCID 回路を経由せずに 1.8V COMS 信号として送信することができる。一般に回路のテスト用に用いられる機能ではあるが、今回の使用用途では検出器からの直接の信号の時間情報を確認することで動作試験を行うため、このバイパス機能を常時用いて Patch-Panel ASIC を使用する。バイパス機能は上述の SPI 通信を用いて 1 チャンネルごとに設定が可能である。

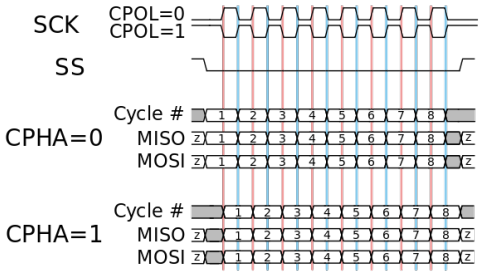


図 3.4: SPI 通信のタイミングチャート [22]

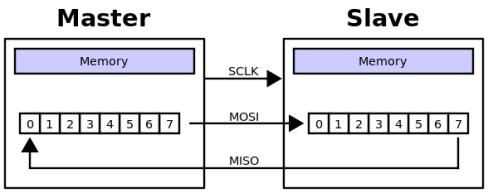


図 3.5: 一般的な SPI 通信のハードウェア構成。8 ビット送受信の場合。[22]

3.2.3　LHC-ATLAS 実験 MDT 用 Time to Digital Converter

Patch-Panel ASIC からの検出器 Hit 情報をもつ 1.8V CMOS 信号は Zynq の PL 部の High-Performance (HP) バンクに入力される。HP バンク は BANK33,34,35(全ピン数 150) に該当し、動作電圧 1.2V〜1.8V の高速 I/O バンクである。この検出器 Hit 情報から Hit の時間情報を取得する。そのために高輝度 LHC-ATLAS 実験 MDT 用に開発された Time to Digital Converter(TDC) を Zynq の PL 部に実装する。高輝度 LHC-ATLAS 実験では現在ミュー粒子の精密測定に使用されている Monitored Drift Tube(MDT) 検出器をトラッキングトリガーとして新しく採用する。現在の MDT には精密測定用の TDC ASIC が採用されているが、高輝度化に伴い時間情報の取得が可能な FPGA 実装用の TDC が名古屋大学及び Open-It により開発された [23]。実際に開発された TDC 回路のブロック図を図 3.6 に示す。周波数 f_{ref} のリファレンスクロックに同期した $8f_{ref}$ のクロックを用いる。位相が 90 度異なる 4 つの位相の $8f_{ref}$ クロックを用いることで、理想的な時間測定の刻み幅は $1/(8f_{ref} \times 4)$ となり、リファレンスクロックに 40 MHz のクロックを用いる場合、刻み幅は 0.78 ns となる。

また、ASD Readout NIM Module において、Patch-Panel ASIC から Zynq へ送信される CMOS 信号は全 128 チャンネルである。MDT 用に開発された TDC は 8 チャンネル用であり、当初はこの 8 チャンネル TDC を 16 個 Zynq の PL 部に実装することで 128 チャンネル全てをカバーする予定であった。しかし、8 チャンネル TDC を 16 個実装する際の FPGA リソース量に問題が生じることが判明した。そのため、開発を行った名古屋大学及び Open-It に対し全 128 チャンネルへの対応が可能な TDC への拡張を依頼し協力を頂き、開発された 128 チャンネル TDC を PL 部に実装することに決めた。8 チャンネル TDC の場合、TDC から出力されるデータは信号の有無、立ち上がり/立ち下がりの判別、チャンネル識別番号、信号時間情報を含め 22 bit であったが、128 チャンネル TDC への拡張後は FIFO 情報を含めた 104 bit に変更された。詳細は第 4.1.2 節で述べる

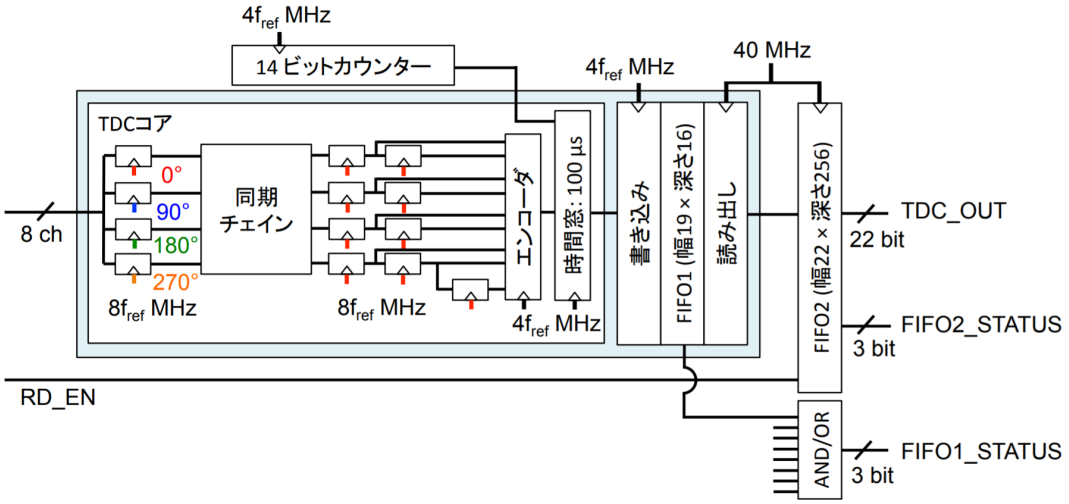


図 3.6: 開発された 8 チャンネル TDC 回路のブロック図。後に 128 チャンネルへの拡張が行われた。

3.2.4　高速通信 GTX を用いたアナログ信号波形取得

ASD Readout NIM Module はデジタル化された信号だけではなく、4 チャンネルのアナログ信号の波形取得も可能にする。LHC-ATLAS 実験で使用されている ASD Board には閾値電圧比較を行う前のアナログ信号を入力全 16 チャンネルのうち 1 チャンネルのみ出力する機能が備わっている。その出力アナログ信号をメインターゲットとし、波形取得が可能な機能を搭載する。ASD Board からは LEMO コネクタを経由し同軸ケーブルで信号が送信される。ASD Readout NIM Module にも LEMO コネクタを実装しこの信号を受け取れるように設計した。

入力されたアナログ信号はまず図 3.7 に示す回路を経由する。図中に示したのは 1 チャンネル分に対応する回路であり、同様な回路が ASD Readout NIM Module には他に 3 つ存在する。この図 3.7 に示した回路では受け取ったアナログ信号の波形整形を行う。ここで波形整形を行う回路について説明を行う。入力されたアナログ信号は TEXAS INSTRUMENTS 社製オペアンプ OPA820ID(図中 U30) を用いた非反転増幅回路を経由する。この非反転増幅回路の利得は約 5 倍に設計されている。その後 RC 積分回路 (図中 R144, C205) を経由し、TEXAS INSTRUMENTS 社製の差動アンプ THS4541IRGTT(図中 U31) を用いた積分回路に入力される。THS4541IRGTT の 9 番ピン (VOCM) に入力されている VCM という名前の信号は後段の ADC から配線されている。これは ADC に入力する差動信号の基準値であり (図 3.8 の水色の信号)、THS4541IRGTT もこの電圧値を基準値として入出力値が設定される。これら 3 段階の回路を経由することで ADC によるサンプリング性能が向上するように信号電圧の増幅を行うとともに、時間変化を滑らかにする目的がある。

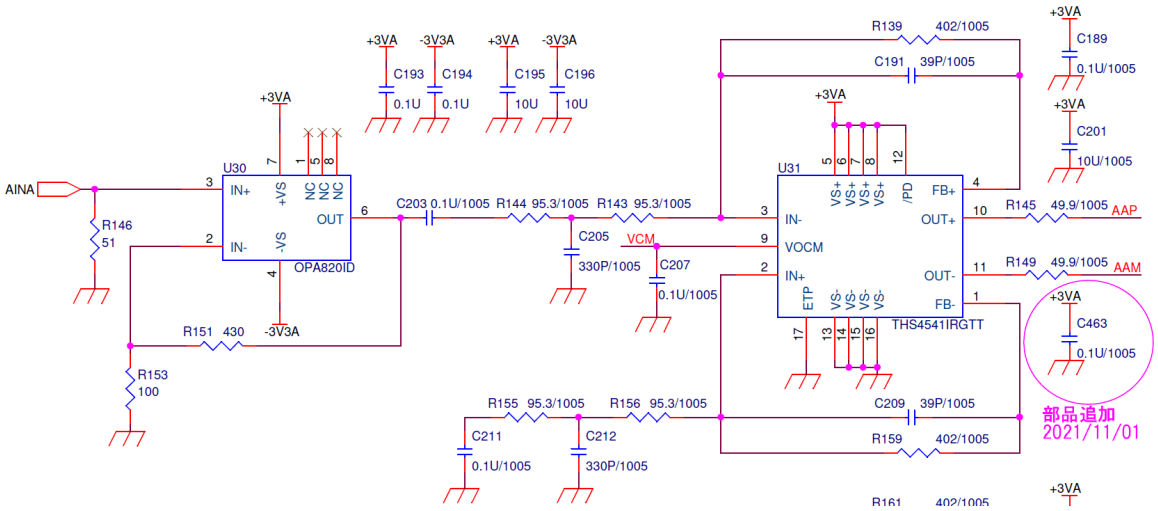


図 3.7: ASD からのアナログ信号を受け取り波形整形を行う回路

波形整形回路の挙動を確認するために、Cadence 社により提供される回路動作シミュレーションツール PSpice を用いて実証を行った [24]。PSpice とは、Cadence 社製電子回路設計専用 CAD ツールである OrCAD Capture 上で動作するシミュレータであり、実際の IC の性能情報から信号の遷移を確認できるツールである。PSpice を用いてシミュレーションを行った結果を図 3.8 に示す。入力信号 (黄緑色の信号) は 130 mV をピークに持つ時定数 16 ns で減衰する信号であり、ASD からのアナログ信号を想定している。これは LHC-ATLAS 実験における ASD Board における利得率 (0.8 V/pC) を考慮するとおよそ 0.16 pC 程度の電荷に相当し、実際には立ち上がりにも時定数を持つことが予想されるがおおよそ ASD Board の典型的なアナログ信号の出力電圧値と同程度のものである。この信号を入力とする後段の OPA820ID を用いた非反転増幅回路の出力 (赤色の信号) が RC 積分回路を経由し (青色の信号)、THS4541IRGTT を用いた積分回路に入力される。前述の通り、後段 ADC からの基準値電圧 (水色の信号) が THS4541IRGTT に入力されているため、その THS4541IRGTT の出力はその電圧値を基準に差動で出力 (ピンク、オレンジの信号) される。このようになだらかな波形に整形されたアナログ信号は TEXAS INSTRUMENTS 社製の Analog to Digital Converter である ADC34J45IRGZT に入力される。

ADC34J45IRGZT は 14 ビットの分解能を持ち最大 160 Megabits Sampling Per Second(MSPS) で動作する ADC である。ADC34J45IRGZT のブロック図を図 3.9 に示す [25]。入力電圧範囲は基準値電圧 (0.95 V) に対し ±0.5 V である。この入力電圧範囲を考慮に入れ、再度 PSpice を用いてシミュレーションを行ったところ、ASD Readout NIM Module へ入力されるアナログ信号の最大値はおおよそ 380 mV 程度であった。これは LHC-ATLAS 実験における ASD Board における利得率を考慮するとおよそ 0.48 pC 程度の電荷に相当する。また、ADC34J45IRGZT は Zynq と JESD204B インターフェイスでの通信が可能である。JESD204B とは半導体分野における規格標準化団体 Joint Electron Device Engineering Council(JEDEC) により制定された ADC や DAC などデータコンバータと FPGA(Zynq) や ASIC 間に用いられ

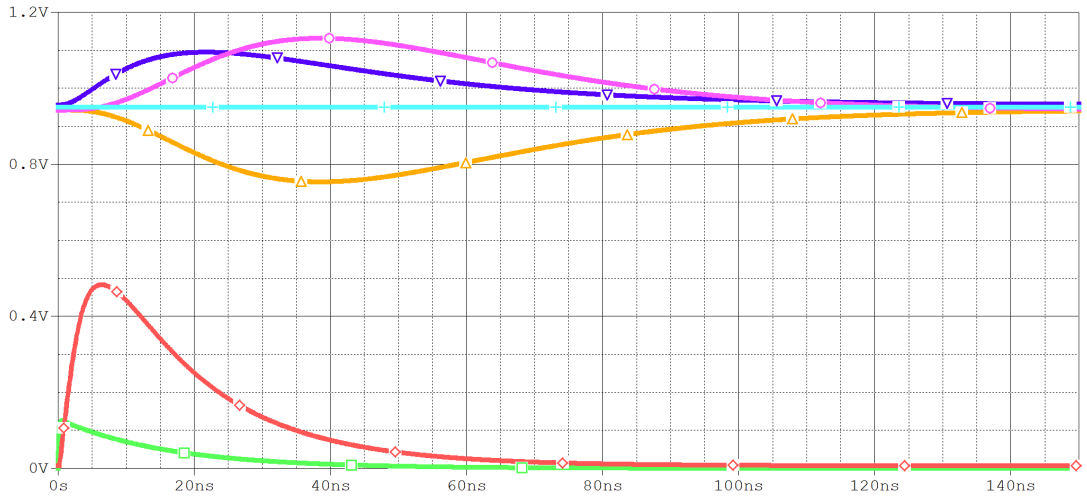


図 3.8: PSpice を用いたシミュレーション結果。130 mV、時定数 16 ns で減衰する信号を入力している。

る高速シリアルデータ通信の標準規格の名称である。CMOS 信号や LVDS 信号に比べ、大量のデジタルデータを正確に少ない信号数で伝送が可能であり、高速化・高伝送効率化 (最高で 12.5 Gbps の転送が可能) されたインターフェイスである。2011 年の発表以降 (前バージョンである JESD204 の発表は 2006 年、その後 2008 年に JESD204A が発表されている) 普及が広がり、現在では Xilinx 社や TEXAS INSTRUMENTS 社により FPGA 用の Intellectual Property (IP) コアも提供されている。そういった IP を Zynq の PL 部に実装した上で、Zynq の高速通信 GTX 第 2.1 節を用いて ADC34J45IRGZT とのデータの送受信を行う。ADC34J45IRGZT の設定は内部レジスタ状態で決定される。SEN (serial interface enable)、SCLK (serial interface clock)、SDATA (serial interface data)、SDOUT (serial interface data output) の 4 つのピンが付随しており、シリアル通信を用いて外部から設定が可能である [25]。ASD Readout NIM Module ではこれらのピンを Zynq PS 部の MIO ピンに接続を行い、Patch-Panel ASIC 同様、PS 部上 Linux OS でのソフトウェアプログラムによる Serial Peripheral Interface (SPI) プロトコルによる通信で制御を行う。

3.2.5　その他の周辺機器/インターフェイスと機能

その他の周辺モジュールについて説明する。表 3.2 にその一覧と使用される IC やコネクタの型番を示す。

PS 部

Zynq のプロセッサを担う PS 部には以下の周辺機器、インターフェイスを接続した。PS 部の Multiplexed Input Output(MIO) を用いて、テクニカルリファレンスマニュアル [17] に示された制限の下、選択と配線を行った。

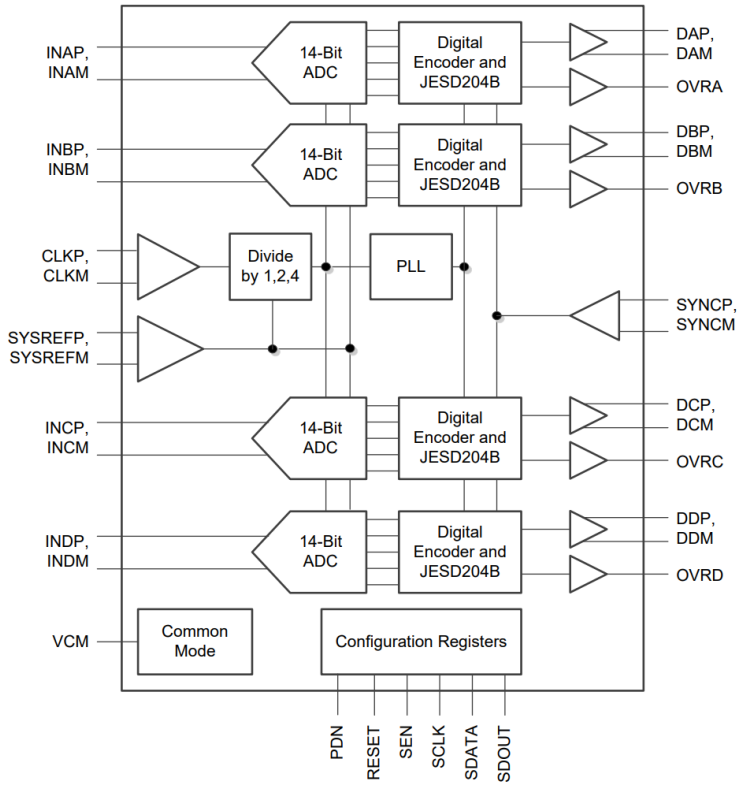


図 3.9: ADC34J45IRGZT のブロック図 [25]

	インターフェイス	型番
PS	USB-UART	690-005-299-043 CP2102N-A02-GQFN24
	Ethernet RJ45 + PHY	0826-1X1T-23-F KSZ9031RNXCC
	DDR3	MT41J128M16JT-125:KTR
	SD card connector	10067847-011RLF
	QSPI memory	MX25U51245GZ4I00
PL	14pin JTAG connector	87831-1420
	microUSB-JTAG conversion connector	JTAG-SMT2
	DAC	DAC7578SPWR
	Monitor ADC	ADS7951SBDBT
	LEMO	SN65LVDS348PW MC100EPT24DG SN65CML100D

表 3.2: 搭載モジュール一覧

- USB-UART
  Zynq PS のコンソール画面の表示に用いる。非同期シリアル通信である Universal Asynchronous Receiver/Transmitter(UART) を用い、接続先の Host PC 上で Zynq PS の CUI 表示を可能にする。使用する USB コネクタは USB-miniB 規格である EDAC 社製の 690-005-299-043 である。USB-UART 変換 IC には Silicon Labs 社製の CP2102N-A02-GQFN24 を採用しており、Host PC には Silicon Labs 社の USB-UART ブリッジ VCP ドライバの導入が必要である。

- Ethernet RJ45 + PHY
  LAN ケーブルを用いた Gigabit Ethernet 通信に用いる。TCP/IP 通信を確立、外部とのインターネット通信やデータ転送を可能にする。コネクタには Rj45 規格である Bel Fuse 社製の 0826-1X1T-23-F、送受信実装の物理層 (PHY)IC には MICROCHIP 社製の KSZ9031RNXCC を採用した。

- DDR3
  Zynq PS の ARM CPU のデータ入出力やソフトウェアプログラムの格納に用いる。Micron Technology 社製の MT41J128M16JT-125:KTR を 2 つ搭載する。合計容量は 4GB であり、32bit データ幅の Dynamic Random Access Memory(DRAM) である。

- SD card connector
  Zynq の Boot 時使用 ファイルや実行ファイルの保存に SD カード用いる。取り外し交換可能な不揮発性メモリであり、PetaLinux を用いた Linux kernel 作成時の設定により Zynq 上 Linux OS のハードディスクとしても使用できる。PS 部で走る Linux OS により中身の書き換えが可能である。コネクタには Amphel 社製 10067847-011RLF を採用する。また、SD メモリカードの標準信号電圧は 3.3V であるが、Zynq PS の動作電圧は 1.8V であるため、Maxim Integrated 社製のレベルシフター MAX13035EETE+ を搭載することで対応を行った。

- QSPI memory
  Zynq の Boot 用ファイルを保存する。取り外し不可能な不揮発性メモリである。Zynq PS との Quad Serial Peripheral Interface(QSPI) 通信により Boot 用ファイルの読み出しを行う。Boot 方法はユーザーが任意で設定可能である。 QSPI memory には Macronix 社製の MX25U51245GZ4I00 を採用する。

PL 部

Zynq の FPGA 部分を担う PL 部には前述の Patch-Panel ASIC からのデータ信号や高速通信 GTX を用いて波形を取得するアナログ信号以外にも、以下の周辺機器を接続する。Zynq の I/O ピンは機能や信号電圧によって BANK として分類されており、各信号の特性とテクニカルリファレンスマニュアル [17] に示された制限の下、選択と配線を行った。

- 14pin JTAG connector
  コンフィギュレーションに用いる。Xilinx 社製の FPGA, Zynq シリーズは、Xilinx 社製の ‘Platform Cable USB II’ を使用して、14 pin コネクタと HostPC の USB を接続し、HostPC からコンフィギュレーションを行うことが一般的であるため、本モジュールにも Molex 社製 14pin コネクタ 87831-1420 を搭載し、コンフィギュレーションバンクである ’BANK 0’ に配線した。

- microUSB-JTAG conversion connector
  上記の‘Platform Cable USB II’の有無に関わらずコンフィギュレーションを可能にするコネクタ。microUSB 規格コネクタと信号変換用 IC が一体化した Digilent 社製 JTAG-SMT2 を採用する。コンフィギュレーション方法はユーザーが任意で設定可能である。

- DAC
  本モジュールの前段回路として想定される ASD では入力アナログ信号をデジタル信号化する際、閾値電圧を用いる。ASD に対し閾値電圧を設定するために Digital to Analog Converter を用いる。Inter-Integrated Circuit(I2C) 規格の通信により PL 部から DAC の制御が可能であり、DAC からの出力はオペアンプを経由した後にコネクタ 8831E-080-170L から出力する。DAC には TEXAS INSTRUMENTS 社製 DAC7578SPWR を採用する。

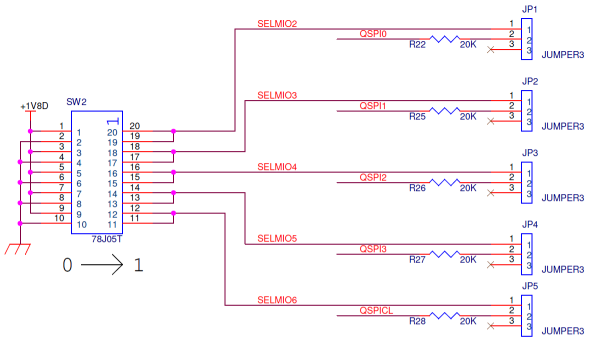
- Monitor ADC
  DAC を用いて ASD に対し設定する閾値電圧の値を監視するために Analog to Digital Converter を用いる。アナログデータ信号の高速な読み出しを行う ADC との区別化のため、‘Monitor ADC’ と表記した。Serial Peripheral Interface(SPI) 通信を用いて PL 部と通信を行い、データの送受信を行う。Monitor ADC には TEXAS INSTRUMENTS 社製 ADS7951SBDBT を採用する。

- LEMO
  LEMO コネクタを用いて外部との同軸ケーブルを用いた NIM 信号及びアナログデータ信号のやり取りを可能にする。フロントパネルに以下の 11 の信号を想定した LEMO コネクタを配置する。

- アナログデータ信号入力 (全 4 チャンネル)
  第 3.2.4 節で述べたアナログ信号の入力。A〜D の名前を付けた 4 チャンネルのコネクタを配置。
- Trigger 入力
  Zynq の PL 部へ入力される信号。内部ロジックによりデータ処理のトリガーに用いることを想定。
- Trigger 出力
  Zynq の PL 部から出力される信号。内部ロジックにより他のモジュールへのトリガー分配に用いることを想定。
- Test Pulse Trigger 入力
  Zynq の PL 部へ入力される信号。Patch-Panel ASIC にから ASD へ送信される Test Pulse 信号のトリガーに用いることを想定。
- クロック入力
  ボード上のクロック管理を行うクロックシンセサイザー及びジッタクリーナーである Si5395K-A-GM へ入力される信号。クロック源を他のモジュールより供給することを可能にし、他のモジュールとの同期を実現する。
- クロック出力
  ボード上のクロック管理を行うクロックシンセサイザー及びジッタクリーナーである Si5395K-A-GM から出力される信号。クロック源を他のモジュールに対して供給することを可能にし、他のモジュールとの同期を実現する。
- 汎用 NIM 信号入力
  Zynq の PL 部に入力される汎用的な信号。Zynq 内部ロジックへのリセット信号等への応用を想定。
- 汎用 NIM 信号出力
  Zynq の PL 部から出力される汎用的な信号。BUSY 信号等への応用を想定。

特徴

- ブートモード選択
  第 2.3.2 節で説明したように、電源投入時に Zynq のコンフィギュレーションを行うブートモードの選択が可能である。図 3.10 にブートモード選択を実現する回路図と、使用するジャンパピンのピンテーブルを示す。Zynq のブートモードには JTAG ブートモード、SD ブートモード、QSPI ブートモードの 3 種類を採用し、JTAG ブートモードにはカスケードモード (DEFAULT JTAG) と独立モード (INDEPENDENT JTAG) の 2 種類があるため、合計 4 種のブートモードの中から選択する設定になっている。
- クロック制御
  ASD Readout NIM Module 上には用途の異なるクロック信号が多数ある。中でも Patch-



	MIO2	MIO3	MIO4	MIO5	MIO6
DEFAULT JTAG	0	0	0	0	0
INDEPENDENT JTAG	1	0	0	0	0
QSPI	0	0	0	1	0
SD	0	0	1	1	0

図 3.10: ブートモード切替用ジャンパピン回路図とピンテーブル

Panel ASIC への 40 MHz クロック、高速読み出し用 ADC への 160 MHz クロック及びリファレンスクロック、Zynq PL 部への 40 MHz クロック及び 160 MHz クロック、さらに高速通信 GTX に用いる 160 MHz クロックの制御にはクロックシンセサイザー及びジッタクリーナーである Silicon Labs 社製の SI5395K-A13539-GM を用いて制御を行う。SI5395K-A13539-GM は Serial Peripheral Interface (SPI) プロトコルによる通信による制御が可能であるため、Zynq PS 部の MIO ピンと接続する。SI5395K-A13539-GM の制御は、同じく Silicon Labs 社製ユーザーユーティリティ Clock Builder Pro により容易なクロック制御が可能である [26]。Clock Builder Pro はプログラミング言語によって開発されているものであるので、従来の FPGA に搭載するのは困難であるが、OS 搭載可能な Zynq によりインストールを容易に行うことができる。また、SI5395K-A13539-GM のクロック源にはボード上に搭載する 40.079 MHz の水晶クロックだけでなく LEMO コネクタ経由で外部から入力されるクロックを用いることが可能である。クロック源はボード上 DIP スイッチにより容易に選択可能である。さらに生成したクロックを LEMO コネクタ経由で外部へ送信することも可能であり、これらによって外部の他のモジュールとの同期を実現できる。

3.3　基板開発

ここまでに説明した要素や機能を実現するハードウェア基板の開発に取り組んだ。回路図の作成には Cadence 社製の電子回路設計専用 CAD ツールである OrCAD Capture を用いた [27]。回路図完成後には試作機のプリント基板 (PCB) 製作、フロントパネル製作、部品実装を外部業者 (有限会社ジー・エヌ・ディー http://www.gn-d.com) に発注した。現在は配線図面の作成が完了し基板作成の段階にある。図 3.11 に作成された配線図面の部品実装面を、図 3.12 に設計されたパネ

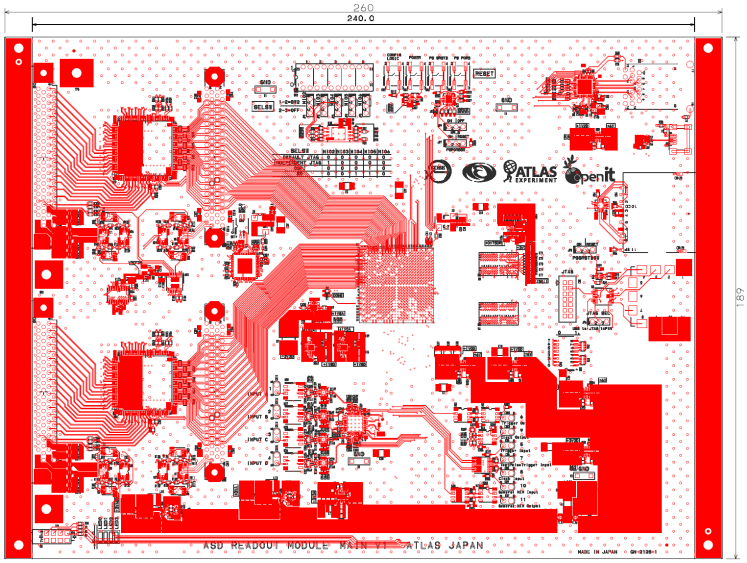


図 3.11: 作成された ASD Readout NIM Module メインボードの配線図。部品実装面。

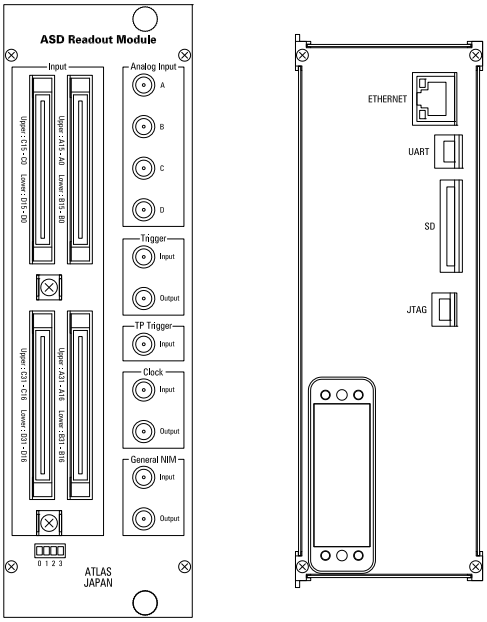


図 3.12: ASD Readout NIM Module のパネルデザイン。NIM2 口幅の規格である。

ルデザインを示す。

第 4 章

ワイヤーチェンバー試験システム

第 3 章で述べたハードウェアデザインを基に、ワイヤーチェンバー試験システムの概観を図 4.1 に示す。検出器からの信号は ASD を介して ASD Readout NIM Module に到達する。ASD は信号の増幅、整形、閾値電圧との比較を行いヒット情報をデジタル信号として送信する。また、デジタル信号化する前のアナログ信号の出力も行う。ASD Readout NIM Module はそれら 2 種のヒット信号を受け取り、それぞれのデジタル化 (数値化) を行う。ここでいうデジタル化とは、デジタルヒット信号の位置情報・時間情報取得とアナログ信号の波形取得である。これらのデジタル化されたデータはトリガーロジックによる取捨選択が可能である。トリガー信号は外部から NIM 信号で取得することが可能であるほか、内部でセルフトリガーとして生成することもできる。トリガー処理を経て取得されたデータは ASD Readout NIM Module 内部でファイルにまとめる。生

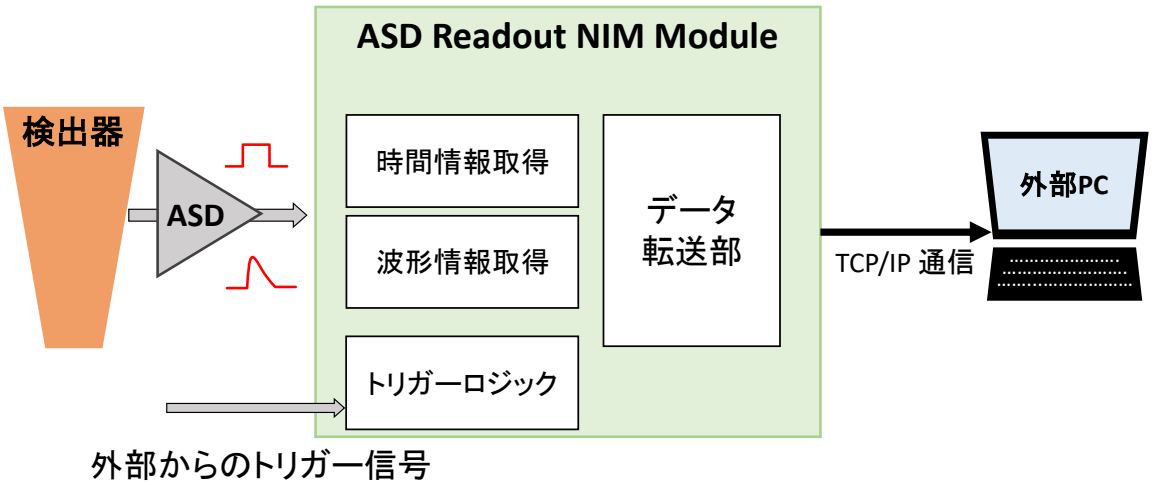


図 4.1: 読み出しシステム全体の概観

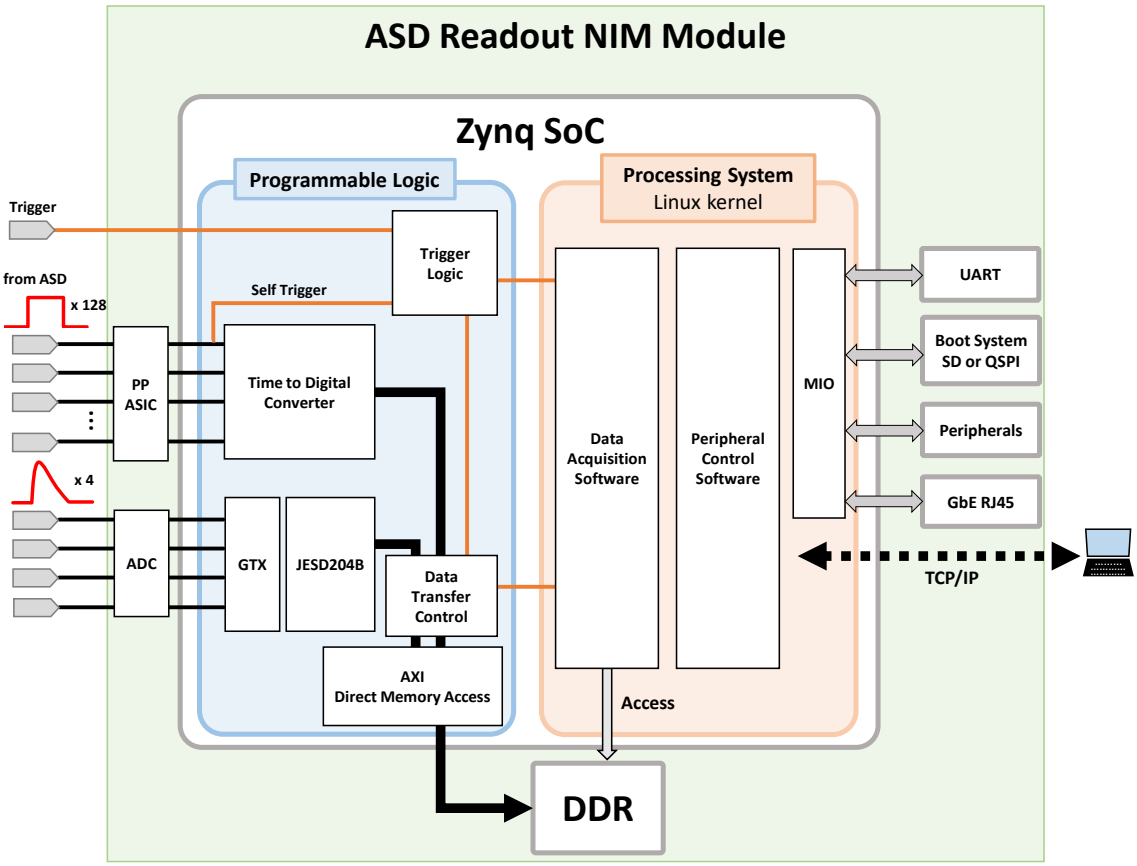


図 4.2: Zynq に必要な機能の概観

成されたファイルは必要により Ethernet ケーブルを介した TCP/IP 通信で外部 PC への転送を行う。

以上のような読み出しシステムを ASD Readout NIM Module を用いて実現するために、必要な機能を中枢である Zynq に実装する。本章では、Zynq に必要な機能の概観を述べ、開発に用いるツールや PL 部、PS 部を構成するファームウェア、PS 部に搭載する OS について述べた後、Zynq 搭載ボードを用いて既に検証を行った技術的要素について述べる。

4.1　Zynq に必要な機能

ASD Readout NIM Module に搭載される Zynq に求められる機能の概観を図 4.2 に示す。ボード上 Patch-Panel ASIC からの全 128 チャンネルの Hit 信号は Zynq PL 部の High Performance (HP) バンク (BANK 33〜35) に入力される。HP バンクは高速動作の DDR メモリとの通信を可能にするバンクであり、高速通信を得意とする。入力された Hit 信号は FPGA 内部に実装する Time to Digital Converter (TDC) へ入力される。TDC により得られた Hit 信号のチャンネル識別番号及び時間情報はデータ転送制御ロジックにより読み出される。また、アナログ信号の波

形取得を目的にボード上に搭載する ADC からの信号は JESD204B インターフェイス規格で到達し、Zynq PL 部の高速通信 GTX バンク (BANK 112) に入力される。高速通信 GTX を用いて受け取った後、PL 内部に搭載する Xilinx 社提供の JESD204B 受信コアによって受信される。受信された波形データはデジタル信号の時間情報と同様にデータ転送制御ロジックから読み出される。また、これらの読み出しを決定するトリガーロジックを実装する。トリガーロジックはモジュール外から入力されたトリガー信号もしくは Patch-Panel ASIC からの入力信号を用いてトリガー信号を内部で発行し、データの取捨選択を決定するロジックである。発行されたトリガー信号をデータ転送制御ロジックが受け取ると、上述の TDC 及び JESD204B 受信コアから取得データの読み出しを行う。また、トリガー信号はデータの取得開始のタイミング情報として PS 部のデータ取得アプリケーションにも送信される。データ転送制御ロジックにより読み出されたデータは AXI Direct Memory Access (DMA) IP を経由して PS 部に付随する DDR メモリに転送し、転送修了のタイミング情報を PS 部データ取得アプリケーションに送信する。これにより PS 部からのアクセスを実現にする。PL 部ハードウェア作成に併せて PS 部の MIO ピン (BANK 500, 501) の設定等を行い、ハードウェア情報として Hardware Design File (HDF) を作成する。

HDF を基に Linux kernel を作成する。Linux OS が走っているホスト PC 上で、ハードウェアに沿ったデバイスツリーと Root File System の設定を行い、Zynq 上で動作可能な Linux ernel をクロスコンパイルする。作成したブート用ファイルを SD カードまたは Quad SPI メモリに格納することで、電源投入時に Zynq PS 部上で Linux OS を走らせることが可能である。Linux OS 上では先述の AXI DMA 経由で PS 部からのアクセスが可能になった取得データの処理を行う。処理を行ったデータは、インターネット通信を介して外部 PC への送信が可能である。また、PS 部に接続されている Patch-Panel ASIC や ADC などの周辺機器の操作を行うソフトウェアアプリケーションを実行し、MIO を経由して通信を行う。

4.1.1　Zynq ファームウェア開発に用いるツール

以上のようなシステムの設計には Zynq ファームウェアの開発が必須である。本節ではそのファームウェア開発に用いられる Xilinx 社製ツールを 2 つ紹介する。

- Vivado
  Zynq の FPGA の役割を担う PL 部の設計や、CPU の役割を担う PS 部のハードウェア的な設計 (MIO ピンの接続設定やプロセッサの駆動用クロック、メモリの設定など) には Xilinx 社が提供するアプリケーションである “Vivado 2020.2” を用いる [28]。Windows マシン上に “Vivado 2020.2” をインストールし開発環境を構築した。Vivado では、Xilinx 社より提供される特定機能を実装可能にするパッケージである Intellectual Package (IP) の導入や、Verilog-HDL などのハードウェア記述言語を使用して任意の信号処理モジュールを配置することで Zynq のハードウェア部分に回路を実装するファームウェアのデザインが可能である。作成されたファームウェアは論理合成と配置配線を経て Bitstream と呼ばれ

るバイナリーファイルを生成し、その Bitstream を含めた Hardware Design File (HDF) としてハードウェア情報を出力する。

- PetaLinux
  Zynq の CPU の役割を担う PS 部で走る Linux OS の開発には、Xilinx 社が提供するクロスコンパイラー “PetaLinux 2020.2” を用いる [16]。Windows マシン上に Oracle VM VirtualBox 6.1 を導入し、仮想環境である Ubuntu 18.04.3 LTS 上で PetaLinux を使用する。クロスコンパイラーとは、開発環境とは異なる環境で実行可能なファイルをコンパイルするツールのことである。PetaLinux では、Vivado で作成したハードウェア情報 (HDF) を元に、ハードウェアに沿ったデバイスツリー、Root File System の設定を行うことで、Zynq PS 上で走る Linus OS をクロスコンパイルすることが可能である。デバイスツリーとは、Operation System(OS) 側からアクセス可能なハードウェアについてソフトウェア目線で記述したデータファイルであり、デバイスツリーを編集することで Linux kernel に組み込むデバイスドライバの設定が可能である。Root File System とは一般にコンピュータファイルシステムにおけるファイル階層の最上位層のことを指し、Linux OS 上で使用したいライブラリやパッケージを設定しコンパイルを行うことで使用する Linux kernel をカスタマイズすることが可能である。第 2.3.2 節で説明したように PetaLinux を用いてクロスコンパイルにより作成したブートファイルを、Zynq に付随するフラッシュメモリ (SD カードや Quad SPI メモリ) にプログラムすることで、電源投入時に Zynq のコンフィギュレーションが行われ Linux OS を起動することが可能になる。

4.1.2　Programmable Logic 部

128 チャンネル時間情報取得ロジック

第 3.2.3 節で ASD Readout NIM Module に搭載する要素として記載した Time to Digital Converter について述べる。ASD Readout NIM Module において、Patch-Panel ASIC から Zynq へ送信される CMOS 信号は全 128 チャンネルである。第 3.2.3 節ではじめに述べた MDT 用 TDC は 8 チャンネル用であり、当初はこの 8 チャンネル TDC を 16 個 Zynq PL 部に実装することで 128 チャンネル全てをカバーする予定であった。しかし、試験的に 16 個の TDC を Zynq PL 部に実装した場合の FPGA にて使用されるリソース量の過剰が確認された。搭載する TDC は 40 MHz クロック、 160 MHz クロック、そして位相が 90 度ずつずれた 320 MHz クロックと内部で合計 6 種類のクロックを用いる仕様になっている。これら 6 種類のクロックの生成には基板上から Zynq PL 部へ供給される 40 MHz クロックを PL 部内に実装する Xilinx 社提供の Clocking Wizard IP[10] を用いて逓倍を行うことで TDC ロジックへの供給を実現する。これら 6 種類のクロックは 8 チャンネル TDC ロジックモジュールの中で各々バッファリングが行われる。そのため、使用されるクロックバッファ (BUFGCTRL) の総数が Zynq(ASD Readout NIM

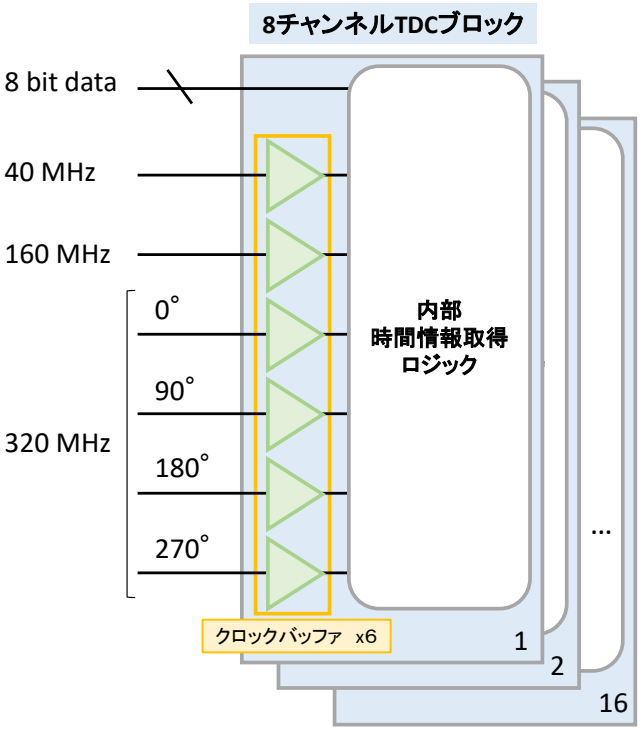


図 4.3: 8 チャンネル TDC を 16 個実装した際の使用リソース過剰原因

Module に搭載する XC7Z030-2FFG676I) のリソースとして内部で使用可能な数を上回る状況が確認され、実装が不可能であることが判明した。XC7Z030-2FFG676I で構築されるロジックにおける使用可能であるクロックバッファ数は 32 であるのに対し、8 チャンネル TDC を 16 個実装した際の使用数は最低でも 103 であった。また、同じく Zynq-7000 シリーズの Zynq において上位性能なデバイスであっても 8 チャンネル TDC を 16 個配置に必要なクロックバッファ数を内蔵しているものはなかったため、消費されるクロックバッファの数を押さえるために 128 チャンネル TDC への拡張が行われた。

128 チャンネル TDC も同じく 6 種類のクロックを用いて時間情報の取得を行う。8 チャンネル TDC の出力データは 22 bit での出力であり、信号の有無、立ち上がり/立ち下がりの判別、チャンネル識別番号 (3 bit)、信号時間情報 (17 bit) で構成されていた。128 チャンネル TDC への拡張後の出力データは 104 bit であり、0〜25、26〜51、52〜77、78〜103 bit 目がそれぞれで同じフォーマットの信号が出力される。出力信号のフォーマットは上位ビットから、2 つの内部 FIFO が FULL 状態である (1) か否 (0) か (2 bit)、信号がある (1) か否 (0) か、信号の立ち上がり (1) 立ち下がり (0)、チャンネル識別番号 (5 bit)、信号時間情報 (17 bit) で形成されている。出力信号はトリガー信号に応じてデータ転送制御ロジックから読み出される。

4 チャンネルアナログ信号波形取得ロジック

第 3.2.4 節で述べた Analog to Digital Converter からの信号を Zynq に入力し、アナログ信号の波形を取得する。ADC から出力される信号は高速通信 GTX を用いて Zynq に入力される。そのため、Zynq PL 部には GTX 受信のブロックが必要となる。GTX 受信ブロックは Xilinx 社より提供される 7 Series FPGA Transceivers Wizard [29] を用いて開発ロジックに組み込むことができる。受信ブロックには外部よりシリアル差動信号としてデータが入力される。パラレル信号への変換や、8b/10b 変換などを行いユーザーが使用できる形で出力する。8b/10b 変換とは、IBM 社が開発したシリアル通信の際に用いられるエンコード及びデコードの方式である。変換はあらかじめ定められたパターンに従って行われる。パターンはどのような 8 bit の信号であっても、同一な値の長期連続送信を避けるように定められており、シリアル信号の中にクロック情報を持たせるほか、信号の直流成分を減衰させる利点がある。GTX 機能及び受信ブロックを用いることで最大 12.5 Gbps でのデータ受信が可能になる

GTX 受信ブロックから出力された信号は JESD204B 受信ブロックに入力される。JESD204B 受信ブロックも GTX 受信ブロック同様 Xilinx 社より IP 化されて提供されている [30]。JESD204B ブロックは入力された GTX 受信ブロックの出力データを第 2.3.1 節で説明した AXI-Stream プロトコルに変換し出力する。また、ブロックの制御には AXI-Lite が用いられる。4 チャンネルの GTX 受信ブロック・JESD204B 受信ブロックを用いてデータ取得を行う場合、JESD204B 受信ブロックからの AXI-Stream プロトコルでの出力は 128 bit である。出力信号はトリガー信号に応じてデータ転送制御ロジックにより読み出される。

トリガーロジック

トリガーロジックを実装することでデータの取捨選択を行う。トリガーロジックには外部からのトリガー信号入力と Patch-Panel ASIC からの信号を入力する。Patch-Panel ASIC からの信号は 128 チャンネルの中から任意の信号を選択でき、検出器のジオメトリーを考慮したセルフトリガーの生成が可能である。トリガーロジックにより発行されるトリガー信号はデータ転送制御ロジックに送信される。また、データ取得開始のタイミング情報として PS 部アプリケーションにも送信する。この送信には第 2.3.1 節で述べた PS-PL 部間の汎用的な通信が可能な General-Purpose (GP) ポートを使用する。

データ転送制御ロジック

トリガーロジックからのトリガー信号を受け取ると、128 チャンネル TDC 及び JESD204B 受信コアに対してデータの読み出しを行う。読み出したデータは AXI DMA ロジックに送信するために AXI4-Strem プロトコルに変換し送信する。また、データを全て送信し終えたときに PS 部アプリケーションに対してデータ取得修了のタイミング情報を送信する。この送信にもトリガーロジック同様 GP ポートを用いる。

AXI Direct Memory Access ロジック

第 2.3.1 節で述べたように PS 部-PL 部間は 3 種のポートによって接続されている。時間情報・波形情報の 2 つのデジタル化により取得さたデータは今回のハードウェア設計では PL 部より直接外部へ転送するような回路を備えていないため、PS 部へ転送する機能を作成する。PS 部から取得したデータへのアクセスが可能になると、PS 部 CPU を用いた数値解析やデータの圧縮、データファイル化あるいはグラフとして画像ファイルを作成し、ユーザーが受け取りやすい形で転送することが可能になる。これを実現するために第 2.3.1 節で述べた High-Performance (HP) ポートを使用する。HP ポートは PS 部のメモリコントローラと PL 部を接続するポートであり、PS 部内メモリコントローラは外部主記憶装置である DDR3 に接続されているため、PL 部と DDR3 の接続を実質的に確立している。この通信には AXI Direct Memory Access (DMA) IP を用いる。AXI DMA IP コアは Xilinx 社により提供される IP コアであり、ユーザーが作成したロジックとメモリとの間の通信を提供している。これを用いることで PS 部付随の DDR からデータを PL 部ユーザーロジックへ転送したり、PL 部ユーザーロジックからデータを PS 部付随 DDR に転送することができる。ユーザーロジックと AXI DMA IP 間のデータ転送は AXI4-Stream プロトコルをサポートしている。また、メモリ側への転送は AXI4 プロトコルが用いられる。

4.1.3　Processing System 部

Zynq PS 部の MIO は各周辺機器に接続されている。MIO を用いて周辺機器との通信を行うには、回路的に接続されている周辺機器を設定しておく必要がある。また、使用する DDR メモリの型番もここで指定する。PL 部-PS 部間の通信を実現するための High-Performance (HP) ポートを有効にしておくことも必要である。これら PS 部のハードウェア的な設定は、図 4.4 のように Vivado を用いた GUI での設定が可能であり、プリセットとして保存しておくこともできる。

Linux kernel の構築

Linux kernel は第 2.2 節で述べたように、ファームウェアやデバイスドライバの情報を集約し、ハードウェアの基本的な制御を行うものである。今回新規基板である ASD Readout NIM Module 上で Zynq を用いるため、構成されるハードウェアの情報や用いるデバイスドライバを基に、Zynq PS 部上で走らせる Linux kernel を新しく構築する必要がある。Linux kernel の構築には第 4.1.1 節で述べたクロスコンパイラー PetaLinux を用いる。

PetaLinux では初めにハードウェア情報 (HDF) の取り込みを行う。HDF は第 4.1.2 節で述べたファームウェア情報を基に Vivado 上で作成されたバイナリファイルであり、PL 部、PS 部ともにハードウェア情報として組み込まれている。PS 部のハードウェア情報では図 4.4 のように、PS 部内のどのインターフェイスを用い、どの MIO ピンを介して通信を行うかが定められている。PetaLinux ではこれを基にデバイスツリーを作成する。自動的にデバイスツリーの作成が行われた後、ユーザーによるカスタマイズが可能になる。本研究においては主に SPI 通信インターフェ

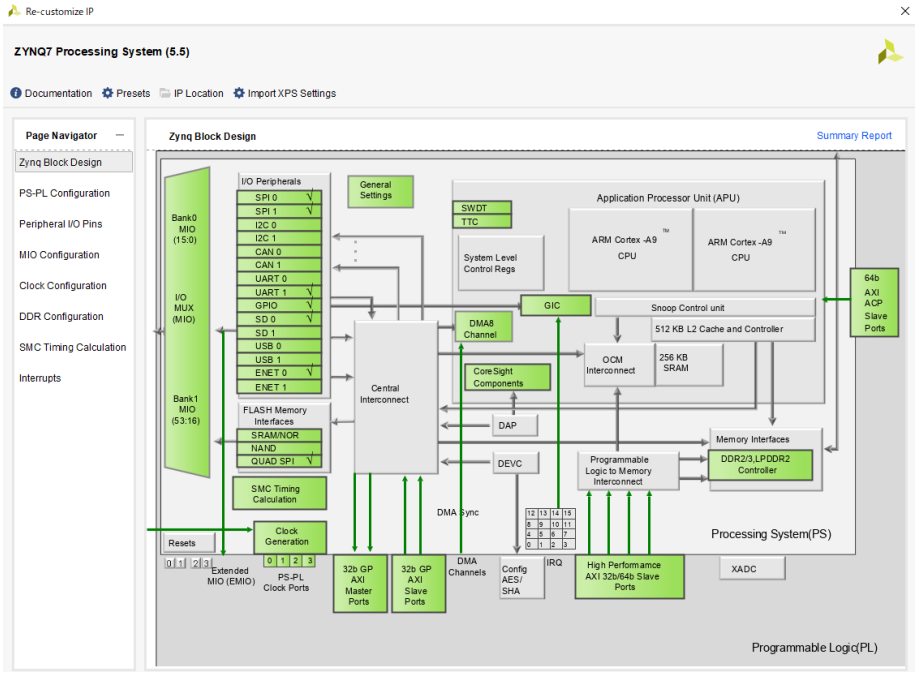


図 4.4: Vivado を用いた PS 部の設定の例

イスのカスタマイズを行った。今回開発するシステムにおいて SPI 通信は ASD Readout NIM Module 上の 4 つの Patch-Panel ASIC、高速読み出しを行う ADC、ボード上のクロック制御を行う SI5395K-A13539-GM の計 6 つの IC と Zynq PS 部間で用いられる。この SPI 通信を Zynq PS 部上の Linux OS から簡易的に行うために SPI 通信用ドライバである “spidev” を使用する。“spidev” に関する記述・設定をデバイスツリーに追加し、kernel への組み込みを行った。また、Linux アプリケーションとして AXI DMA を用いるためにデバイスドライバへの組み込みを行った。さらに構築した Linux OS 上で必須となるパッケージの導入や Root File System の設定を行い、クロスコンパイルを行う。これによって第 2.3.2 節で説明した、電源投入時に Zynq のコンフィギュレーションを行い Linux OS を走らせるための 3 種類のブートファイルを作成した。

ソフトウェアアプリケーション

PS 部上で走る Linux で実行するソフトウェアアプリケーションについて述べる。AXI DMA を用いて転送されたデータは PS 部に接続される DDR に格納される。TCP/IP 通信によって取得したデータを PS 部から出力する場合、DDR に格納されているデータにアクセスを行いデータファイルとしてまとめる必要がある。DDR へのデータ転送の開始と終了のタイミングは GP ポートを介して PL 部から PS 部へ伝達される。PS 部で実行するデータ取得アプリケーションではこの情報を基に DDR へのアクセスを行い、取得データのファイル化を行う。作成したデータファイルから解析を行い、グラフとしてファイル化する手段も可能である。また、PS 部 CPU では取得データの処理に加え、周辺機器操作が可能である。周辺機器操作を行うには、Linux kernel に組み込ん

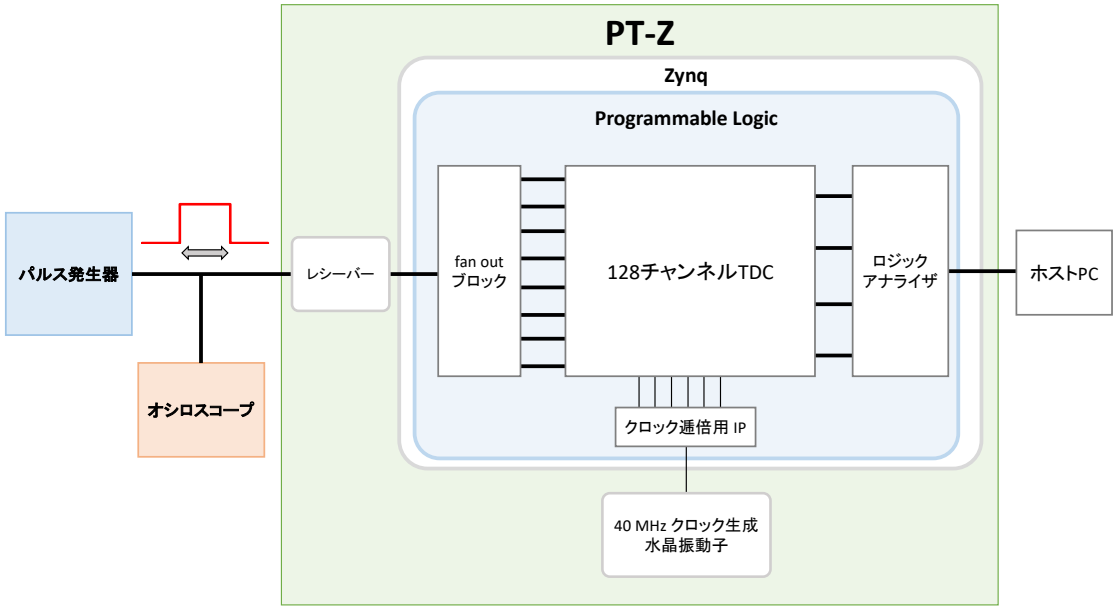


図 4.5: PT-Z を用いた 128 チャンネル TDC 動作検証環境

だデバイスドライバを用いて PS 部 MIO から通信を行うアプリケーションが必要である。

4.2　機能の検証

第 4.1 節で述べたような機能の実現にはいくつもの技術的要素が必要である。本節ではそのいくつかの要素の検証を行った結果を述べる。なおこの検証は本研究で開発を行った ASD Readout NIM Module の完成よりも前に行ったため、第 2.3.3 節で紹介した Zynq 搭載汎用モジュール PT-Z や、Xilinx 社製の Zynq 評価ボードである ZC706[31] を用いている。以下の各項目に対して検証を行った PT-Z と ZC706 にはどちらも Zynq-7000 シリーズである XC7Z045 が搭載されている。

128 チャンネル TDC の動作検証

第 4.1.2 節で述べた、使用するクロックバッファの数により 128 チャンネルへの拡張が行われた TDC の動作検証を行う。128 チャンネル TDC の元となる 8 チャンネル TDC おいても、本来 FPGA 実装用に開発されたものであるため、Zynq においての動作は確認されていない。今回の検証には汎用モジュール PT-Z を用いる。PT-Z は外部からの信号を Zynq PL 部に入力し、内部ロジックに組み込む機構が備わっている。この外部からの入力信号を用いて 128 チャンネル TDC の挙動を検証する。図 4.5 を用いて動作検証の内容を説明する。128 チャンネル全てで時間情報の確認を行うのは現実的ではないため、128 チャンネルの中から 8 チャンネルのみ選択し、動作検証を行う。128 チャンネル TDC は 8 チャンネル TDC を拡張したものであるため、検証には 8 チャン

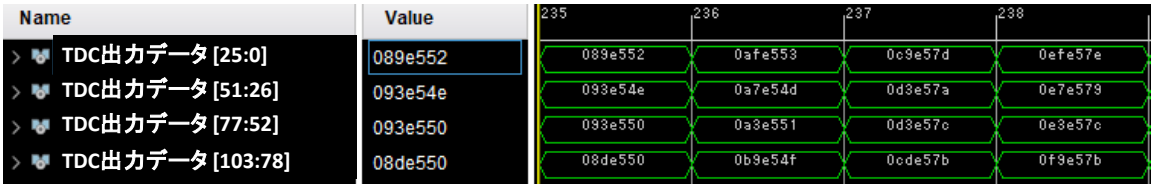


図 4.6: ホスト PC 上から確認できるロジックアナライザ。128 チャンネル TDC からの 104 bit 出力を同じフォーマットごとに 4 つに分割しそれぞれを確認している。

ネル以上離れた入力チャンネルを無作為に選出した。

今回の検証では、パルス発生器を用いて TDC への入力信号を生成する。PT-Z には LEMO コネクタ経由で入力され、レシーバーを経て Zynq PL 部に入力される。入力された信号は fan-out ロジックで 8 本の信号に分配される。8 本の信号は TDC に入力され、時間情報の読み出しが行われる。TDC の動作には前述した 6 種類のクロックが用いられる。基板上の水晶振動子から 40 MHz クロックを入力し、Xilinx 社より提供される Clocking Wizard IP[10] を用いた逓倍・位相調整により 6 種類のクロックを生成し、TDC へ入力した。TDC で読み出された時間情報は同じく Xilinx 社より提供されるロジックアナライザ Integrated Logic Analyzer (ILA) IP[11] を用いて時間情報データの取得を行った。なお、今回の検証においては 128 チャンネル TDC からの 104 bit 出力を同じフォーマットごとに 4 つに分割しそれぞれを確認している。ホスト PC 上でロジックアナライザを確認する様子を図 4.6 に示す。

以上のような環境で、パルス発生器を調整することでパルス幅を変更していく。実装する TDC は信号の立ち上がりと立ち下がりの時間情報が取得可能であることから、それぞれの時間情報の差分が入力する信号のパルス幅の時間情報を示すことになる。ロジックアナライザを用いて取得した出力データから時間情報を抽出し、10 進数に変換を行う。その後チャンネルごとに信号の立ち上がり時間情報と立ち下がり時間情報の差分を計算する。今回の検証では 20 ns から 100 ns までの間を 10 ns 刻みにパルス幅の変更行った。各入力信号パルス幅に対し、3 度ずつ差分値を確認し、その平均値を計算した。

図 4.7 に、ある 1 チャンネルでの検証結果を示す。横軸は TDC 出力値から計算を行ったパルス幅に対応する時間情報、縦軸はパルス発生器を用いて生成した入力信号のパルス幅である。一次関数を用いたフィットを行っており、図中に表記したパラメータ ‘p0’、‘p1’ はそれぞれフィットに用いられた一次関数の傾きと切片を表している。また、別の 7 チャンネルを含め、検証で得られたフィット関数の傾きと切片を表 4.1 に示す。

フィット一次関数の傾きは TDC の出力値 1 幅に対応する時間 (ns) である TDC 刻み時間と解釈することができる。拡張前の 8 チャンネル TDC の TDC 刻み時間は 0.78 ns であった [23]。今回の検証により Zynq 上にも実装可能であること及び無作為に選出した 8 チャンネル全てにおいて 0.78 ns を下回る TDC 刻み時間が確認され、Zynq においての時間情報取得も問題なく行えることを確認した。

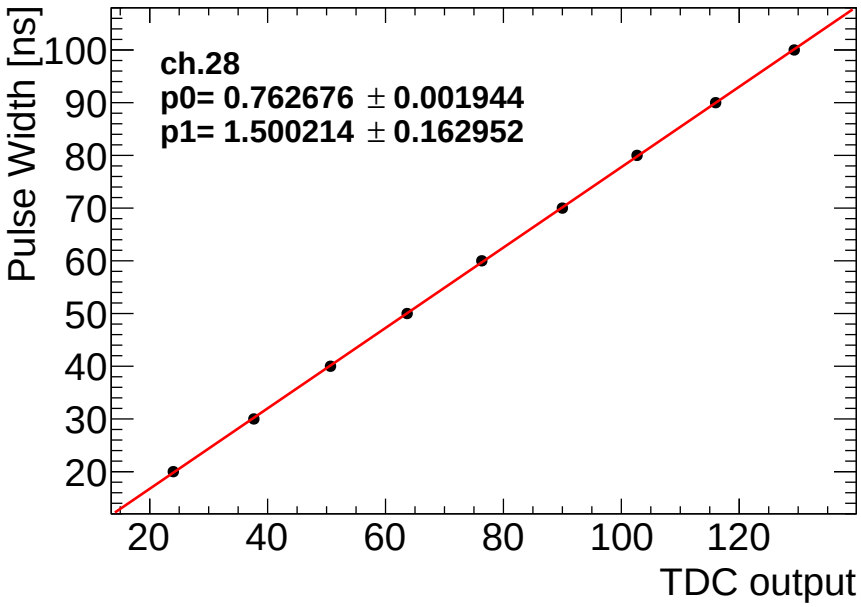


図 4.7: 128 チャンネル TDC におけるある 1 チャンネルでの検証結果。入力信号のパルス幅を縦軸に、TDC を用いて算出した入力パルス幅に対応する値を横軸にプロットしている。プロットには 3 度の計測の平均値を採用している。p0、p1 はそれぞれフィットにもちいた一次関数の傾きと切片。

チャンネル番号	傾き	切片
6	0.7601 ± 0.0028	1.62 ± 0.24
28	0.7627 ± 0.0019	1.50 ± 0.16
41	0.7613 ± 0.0035	1.58 ± 0.30
49	0.7578 ± 0.0035	2.02 ± 0.29
73	0.7600 ± 0.0033	1.79 ± 0.28
83	0.7592 ± 0.0016	1.77 ± 0.13
100	0.7584 ± 0.0035	1.85 ± 0.29
119	0.7594 ± 0.0027	1.72 ± 0.23

表 4.1: 検証した 8 チャンネルでの一次関数フィットのパラメータ。検証した全チャンネルで 0.78 ns 以下の刻み幅を計測。

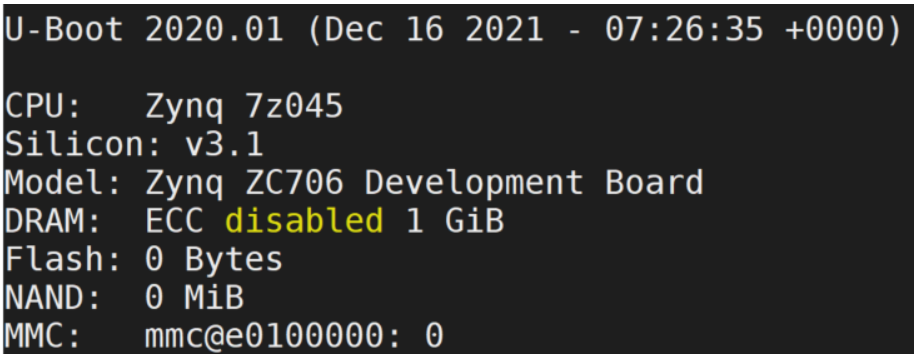


図 4.8: Linux OS の起動時の起動メッセージ最上部。適切に CPU を対象に選んでいることが分かる。

Linux OS の起動検証

Linux OS の起動検証を行う。今回は評価ボード ZC706 を用いて検証を行った。まず、Vivado を用いてハードウェア情報の作成を行う。今回の検証では PL 部は使用せず、Linux OS の起動確認と、SPI 通信用デバイスドライバの kernel への組み込みについて検証を行うため、Zynq PS 部のセッティングのみのハードウェア情報を作成する。ZC706 上で PS 部 MIO ピンに接続されている周辺機器に関しても適切に設定を行った。作成したハードウェア情報を基にして PetaLinux を用いて Linux kernel の構築を行う。デバイスツリーを編集することにより、前述した SPI 通信用のデバイスドライバである “spidev” の Linux kernel への組み込みも行った。作成した 3 種類のブートファイルを SD カードに格納する。このとき SD カードは 2 つのパーティションに分割する。第 1 パーティションはファイルシステムを FAT32 に設定し、ブートファイルの格納に用いる。第 2 パーティションはファイルシステムを EXT4 に設定し、Root File System に用いる。データを格納した SD を ZC706 上 SD カードコネクターに接続し起動を行った。起動の確認は Zynq PS 部との UART 通信を用いて行う。ホスト PC 上に UART 通信用のドライバを導入し、コンソール上で確認を行った。通信速度は 115200 bps に設定している。コンソール上の起動検証画面の最上部を図 4.8 に、最下部を図 4.9 に示す。

問題なく Linux OS の起動が成功し、UART 通信にも成功していることが分かる。また、図 4.10 に示すように、使用可能なデバイスドライバが表示される /dev 階層への SPI 通信用デバイスドライバである “spidev” の導入にも成功した。

```
Starting Dropbear SSH server: random: dropbear: uninitialized urandom read (32 bytes read)
dropbear.
random: crng init done
Starting internet superserver: inetd.
Starting syslogd/klogd: done
Starting tcf-agent: OK

PetaLinux 2020.2 xilinx-zc706-2020_2 ttyPS0

xilinx-zc706-2020_2 login: root
Password:
root@xilinx-zc706-2020_2:~#
```

図 4.9: Linux OS の起動時の起動メッセージ最下部。異常なく Linux OS が起動し、root でのログインが成功した。

```
root@xilinx-zc706-2020_2:~# ls /dev | grep spidev
spidev1.0
```

図 4.10: /dev 階層に使用可能なデバイスドライバとして “spidev” が導入された様子。デバイスツリーの編集が反映されていることが確認できた。

```
root@xilinx-zc706-2020_2:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0A:35:00:1E:53
          inet addr:10.37.1.90  Bcast:10.37.1.255  Mask:255.255.254.0
          inet6 addr: fe80::20a:35ff:fe00:1e53/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:845 errors:0 dropped:510 overruns:0 frame:0
          TX packets:43 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:44295 (43.2 KiB)  TX bytes:7868 (7.6 KiB)
          Interrupt:28 Base address:0xb000
```

図 4.11: UART 通信先の Zynq に割り振られた IP アドレスを確認する様子。確認した IP アドレスをホスト PC からの TCP/IP 通信に用いる。

TCP/IP 通信検証

Linux OS が搭載されている ZC706 上の Zynq PS 部に対し、外部からの TCP/IP 通信を検証する。TCP/IP 通信の検証には SSH コマンドを用いる。Secure SHell (SSH) とはネットワーク上に存在する UNIX コンピュータに対し、別のコンピュータから遠隔操作での UNIX コマンド実行を可能にするコマンドである。この遠隔操作アクセスには TCP/IP 通信が利用されるため、SSH を用いて ZC706 上 Zynq PS 部にアクセスが可能になることは TCP/IP 通信が成功していることを示す。

先述の Linux OS が搭載された ZC706 を Ethernet ケーブルを介してネットワーク通信環境下に配置する。まず、Linux OS を起動検証時のように UART 通信を用いて ZC706 上の Zynq に割り振られる IP アドレスを確認する。その時の様子を図 4.11 に示す。得られた IP アドレスを用いて

```
ikemori@PS200016:~$ ssh root@10.37.1.90
root@10.37.1.90's password:
root@xilinx-zc706-2020_2:~#
root@xilinx-zc706-2020_2:~# ls /dev | grep spidev
spidev1.0
```

図 4.12: SSH コマンドを用いて遠隔操作でのログインと UNIX コマンドの実行が成功した様子。TCP/IP 通信に成功していることが確認できる。

ホスト PC からの SSH ログインを検証する。このとき、ホスト PC には Windows 10 を搭載したコンピュータを用いている。Windows Subsystem for Linux (WSL)[32] を用いて Windows マシンで Linux 動作環境を構築している。OS には Ubuntu を使用している。検証した結果を図 4.12 に示す。遠隔操作での SSH コマンドによるログインと UNIX コマンドの実行が成功したため、TCP/IP 通信によるアクセスが可能であると結論付けられる。

第 5 章

結論と展望

標準模型を超える新物理探索に向け、LHC では高輝度化アップグレードが予定される。それに伴い ATLAS 実験においても新規ワイヤーチェンバーが導入される。その新規ワイヤーチェンバー試験を一つの使用目的とする ASD Readout NIM Module と、その要素機能の開発を行った。

Zynq は FPGA と CPU が一体となった SoC デバイスである。汎用性と利便性を求め、Zynq を ASD Readout NIM Module の中枢として基板設計を行った。Zynq PL 部では 2 つのデジタル化を行う。1 つ目は ASD から入力されるデジタル信号の時間情報取得である。粒子のヒットが持つ位置情報・時間情報はワイヤーチェンバー試験において重要な指標である。また、飛来位置やケーブル長に依存して調整が必要なタイミング情報も ASD Readout NIM Module を用いて収集することが可能である。2 つ目はアナログ信号の波形情報取得である。ボード上 ADC に入力される想定信号を用いてアナログ回路シミュレーションを実行することで、波形整形回路の設計を行った。この ADC と Zynq の機能である高速通信 GTX を用いることで最大 160 MSPS のサンプリングレートを達成する。入力は LEMO コネクタ経由の同軸ケーブル転送を想定し、使用用途も汎用性が高いものであると言える。

以上 2 つのデジタル化後のデータは内部トリガーロジックを用いて取捨選択が可能である。読み出されたデータは Zynq PS 部からのアクセスを実現する。PS 部には CPU が内蔵され、汎用的な OS を用いたユーザーアプリケーション開発が可能である。ユーザーアプリケーションによるデータ読み出しや MIO を用いた周辺機器操作が可能な設計になっている。また、外部との TCP/IP 通信を行い、外部 PC と PS 部間のデータ転送も可能である。

ASD Readout NIM Module は現在基板作成中であるが、本研究の中では既に、同じく Zynq が搭載された PT-Z や ZC706 を用いて要素機能の開発を行った。本システム開発により 128 チャンネル入力への拡張が行われた TDC は、初めて Zynq に実装可能があることが実証でき、その性能も FPGA 用に開発された 8 チャンネル TDC と変わりないことを示した。また、PS 部に搭載する Linux OS 用の kernel の構築を行い、ZC706 上 Zynq での動作検証を行った。想定した Linux OS が異状なく起動し、デバイスツリー編集による SPI 通信用デバイスドライバである “spidev” の導入にも成功した。また、外部マシンから Ethernet ケーブルを介しての TCP/IP 通信が可能であることを確認した。

ASD Readout NIM Module の完成後は読み出しシステムの確立に向け、動作試験及び機能開発を行う予定である。また、システム完成後には新規導入ワイヤーチェンバーの試験に本システムを用いることで、ATLAS 実験アップグレードへの貢献だけでなく、その高い汎用性と簡易性を活かして様々な新物理探索に使用されることに期待する。

付録

A　ASD Readout NIM Module の回路図

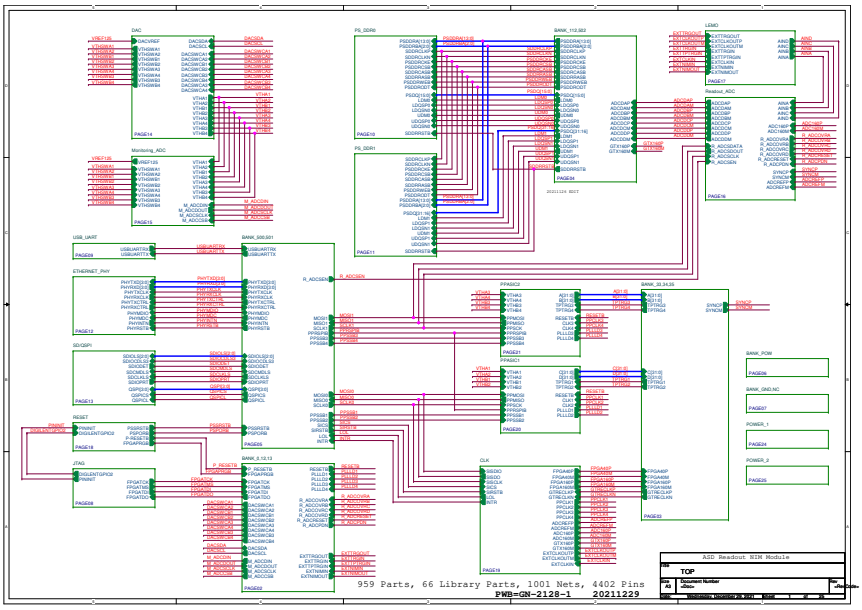


図 A.1: ASD Readout NIM Module メインボードの回路図。TOP ページ。TOP ページ配下には各ページがモジュールとして存在しており、TOP ページにて各ページ間の配線を行っている。

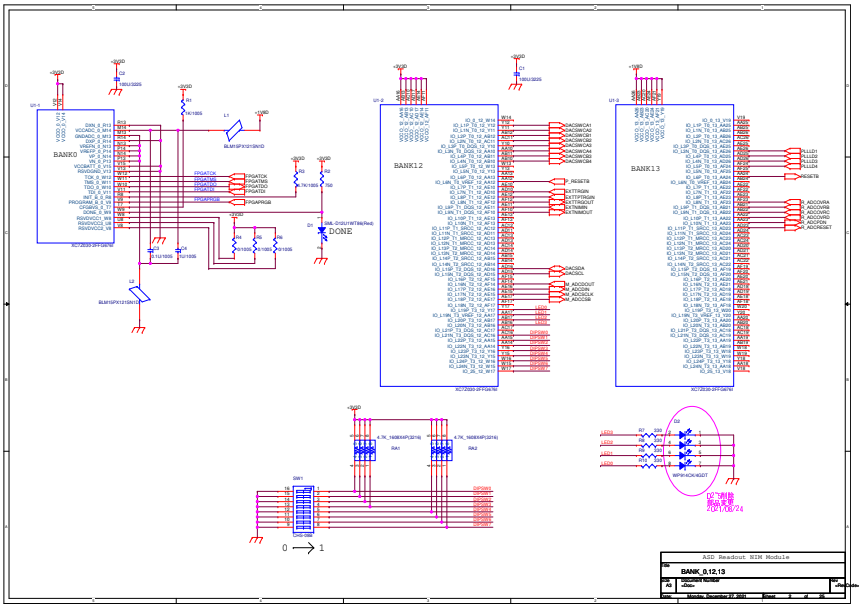


図 A.2: ASD Readout NIM Module メインボードの回路図。BANK0,12,11 ページ。BANK0 はコンフィギュレーション用、BANK12,11 はそれぞれ 3.3 V,1.8 V の動作電圧を持つ PL バンク。主に DAC や外部との NIM 信号送受信に使用。

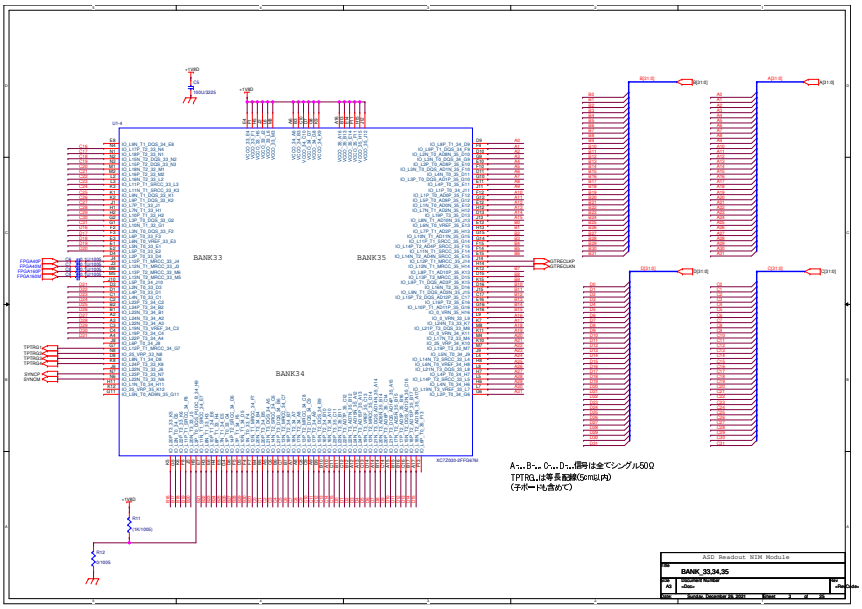


図 A.3: ASD Readout NIM Module メインボードの回路図。BANK33,34,35 ページ。1.8 V 動作の PL バンク。主に Patch-Panel ASIC からのデータ受信を行う。配線時スワップ用に各バンクを統合している。PL 用 40 MHz、160 MHz のクロック受信も行う。

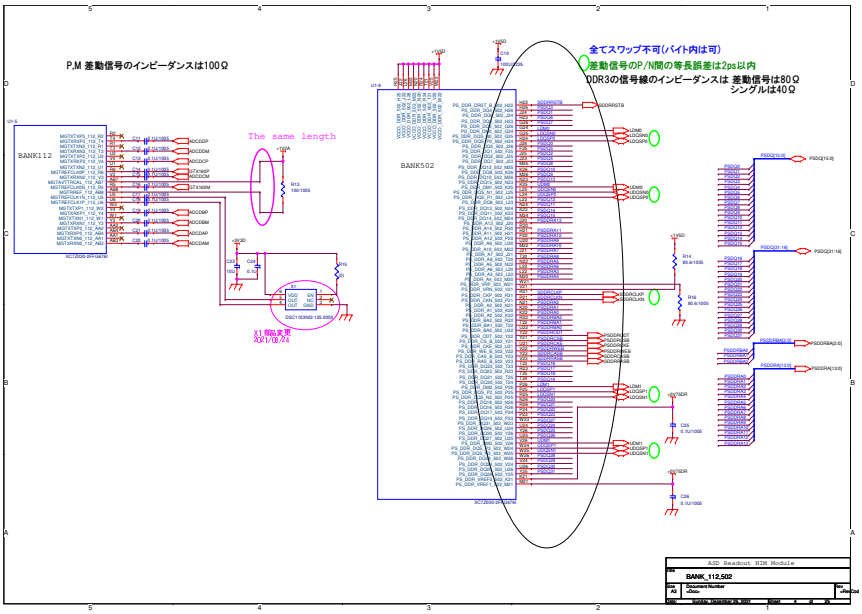
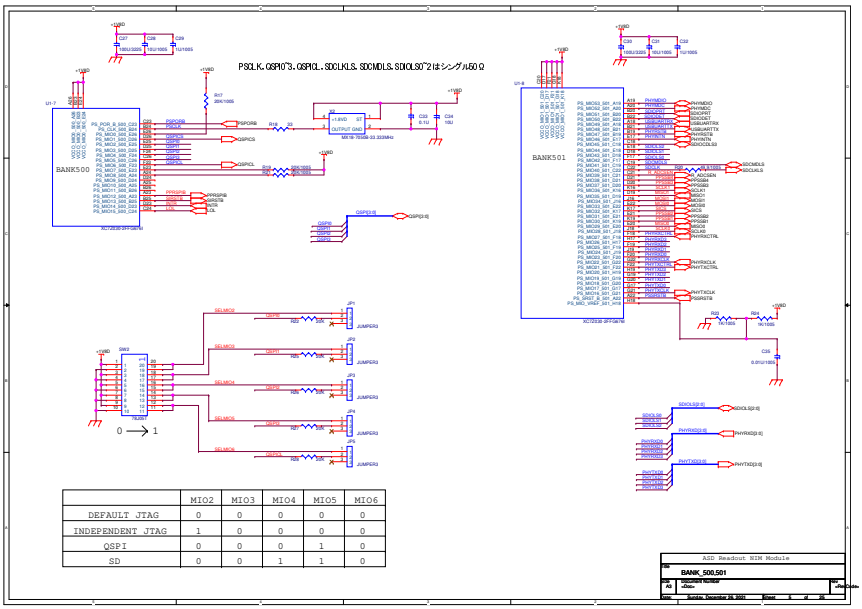


図 A.4: ASD Readout NIM Module メインボードの回路図。BANK112,502 ページ。BANK112 は GTX 用バンクであり ADC からのデータ受信に用いる。BANK502 は PS の DDR 接続バンク。



	MIO2	MIO3	MIO4	MIO5	MIO6
DEFAULT JTAG	0	0	0	0	0
INDEPENDENT JTAG	1	0	0	0	0
QSPI	0	0	0	1	0
SD	0	0	1	1	0

図 A.5: ASD Readout NIM Module メインボードの回路図。BANK500,501 ページ。PS 周辺機器操作に使用される MIO 用バンク。

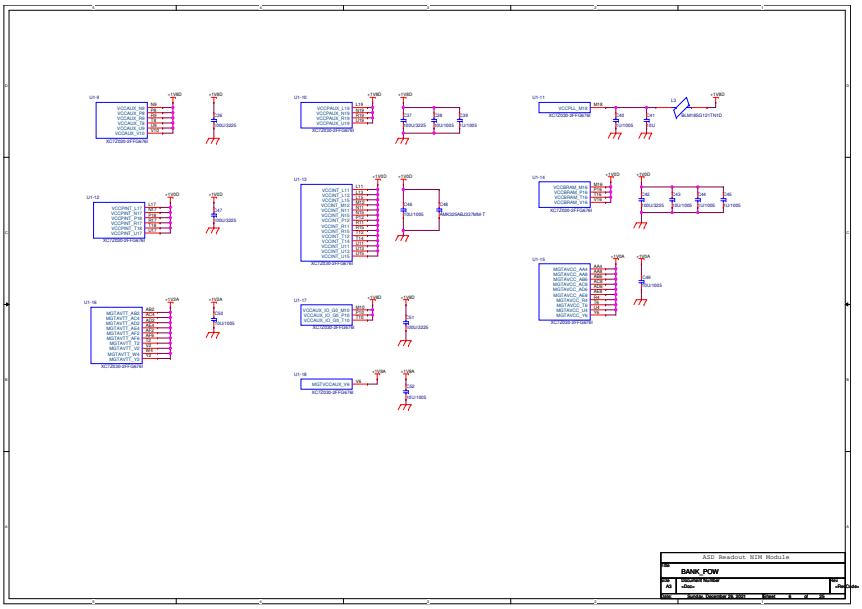


図 A.6: ASD Readout NIM Module メインボードの回路図。BANK POWER ページ。Zynq 動作に投入が必要な電圧値を設定。

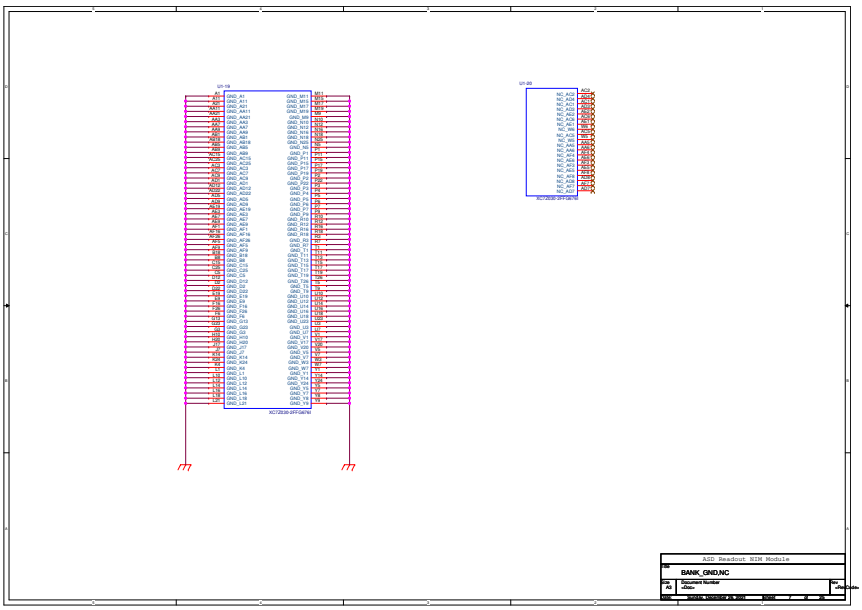


図 A.7: ASD Readout NIM Module メインボードの回路図。BANK GND,No connect ページ。BANK GND は Zynq の GND 用に使用。BANK No connect には何も接続されていない。

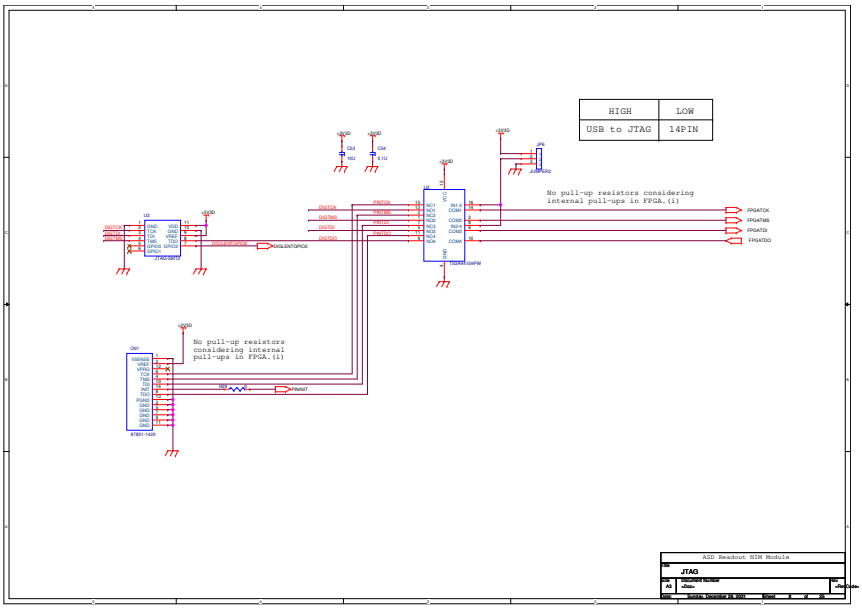


図 A.8: ASD Readout NIM Module メインボードの回路図。JTAG ページ。14 pin コネクタと USB コネクタの両方からコンフィギュレーションを行えるように設計。

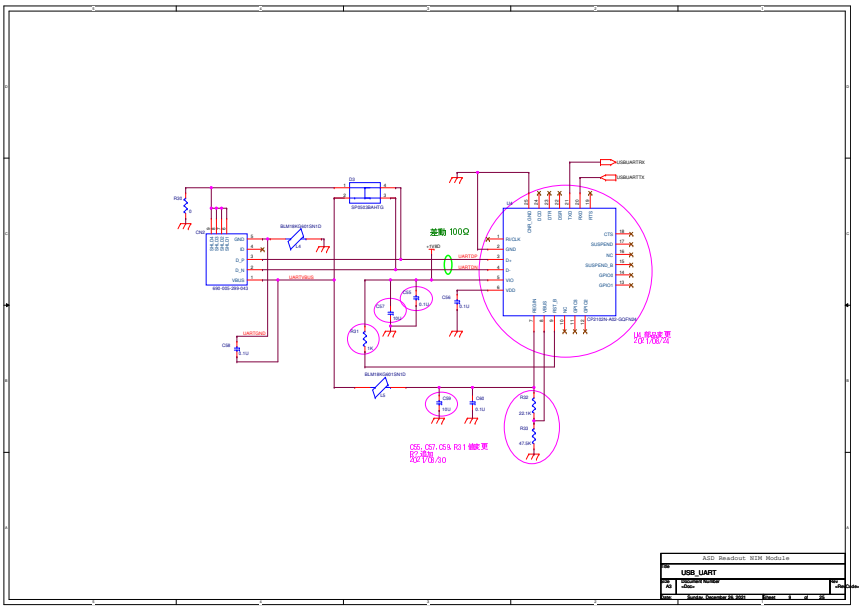


図 A.9: ASD Readout NIM Module メインボードの回路図。USB-UART ページ。UART 通信用素子と USB コネクタにより構成。

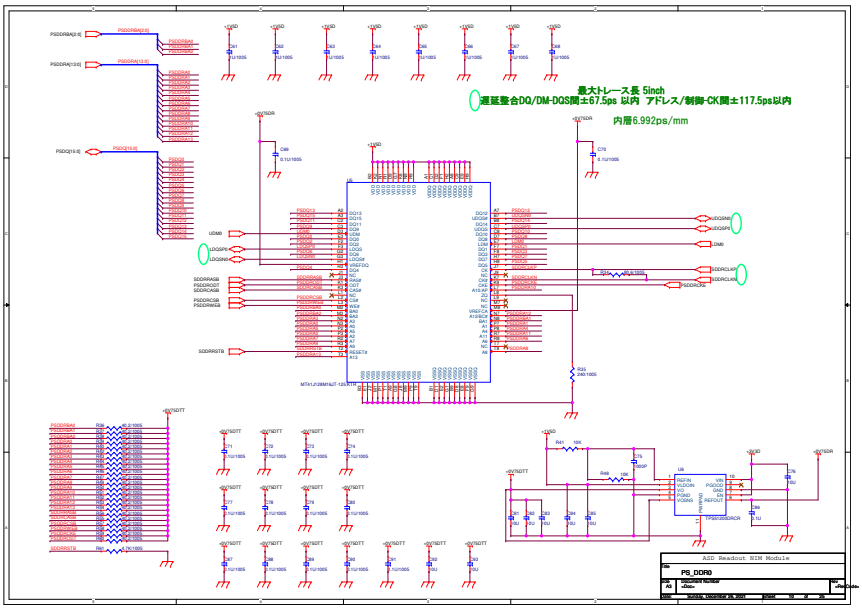


図 A.10: ASD Readout NIM Module メインボードの回路図。PS DDR0 ページ。PS 用の DDR[15:0]

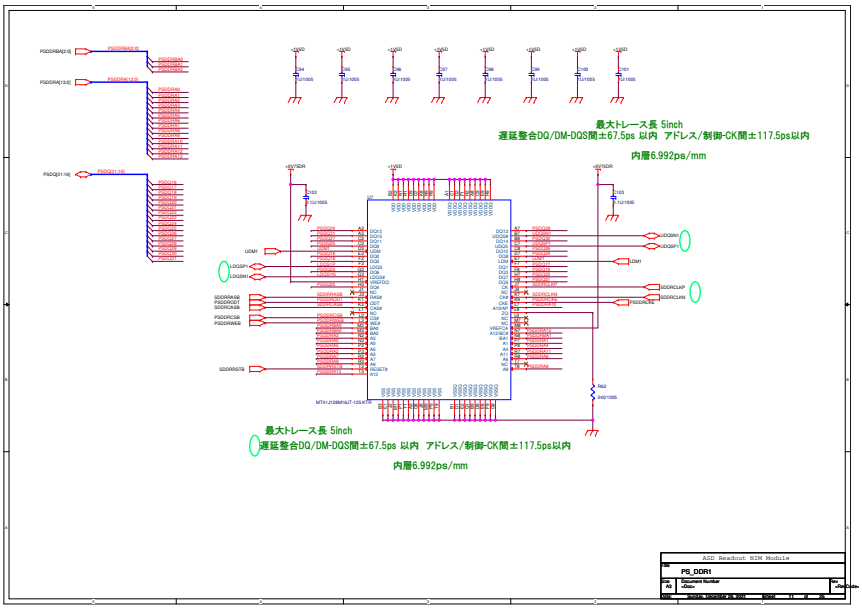


図 A.11: ASD Readout NIM Module メインボードの回路図。PS DDR1 ページ。PS 用の DDR[31:16]

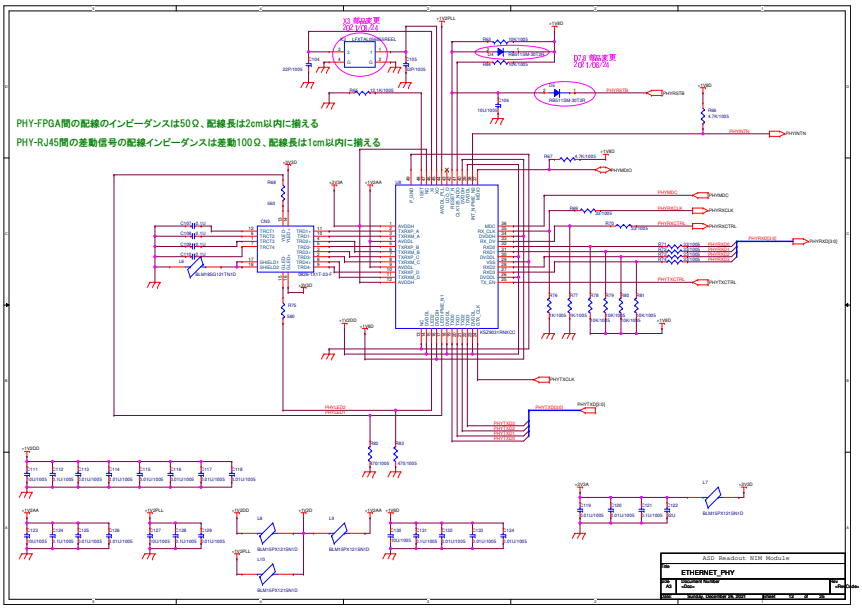


図 A.12: ASD Readout NIM Module メインボードの回路図。Ethernet PHY ページ。Ethernet 用 RJ45 と PHY 素子により構成。

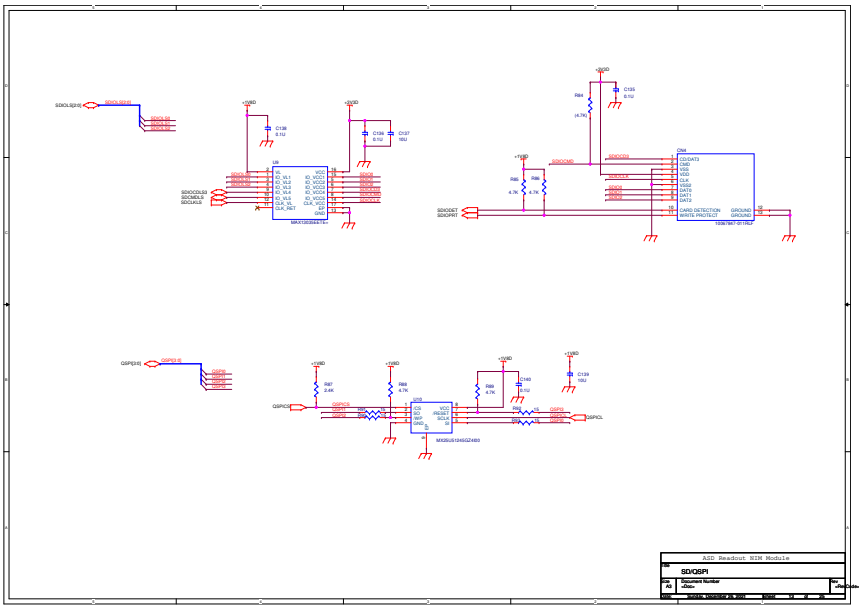


図 A.13: ASD Readout NIM Module メインボードの回路図。SD/QSPI ページ。Zynq のブートファイル格納用。

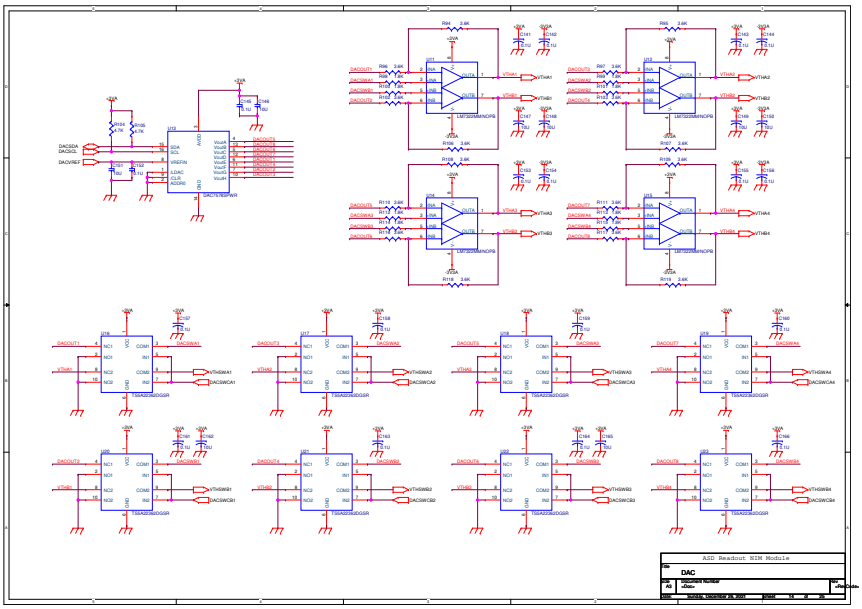


図 A.14: ASD Readout NIM Module メインボードの回路図。DAC ページ。ASD への閾値電圧設定用。

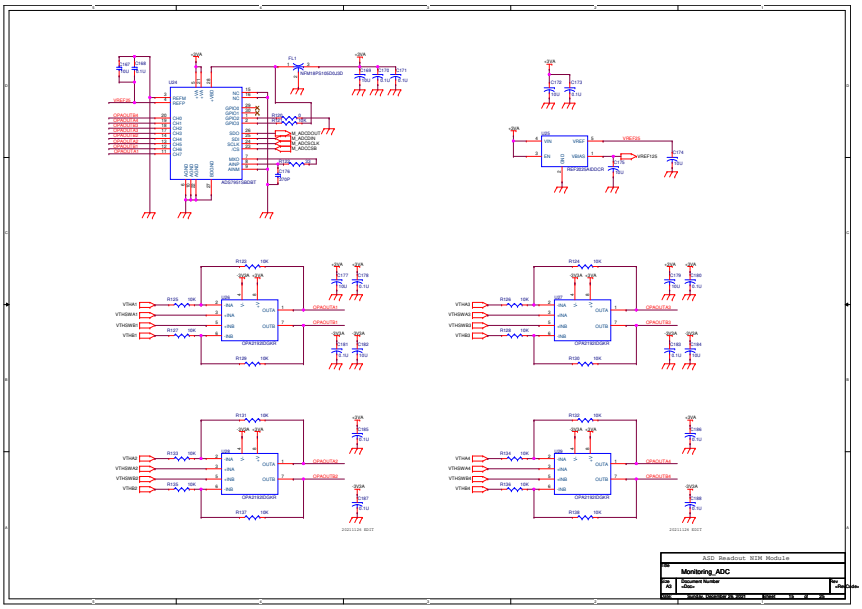


図 A.15: ASD Readout NIM Module メインボードの回路図。Monitor ADC ページ。DAC による設定電圧値モニター用。

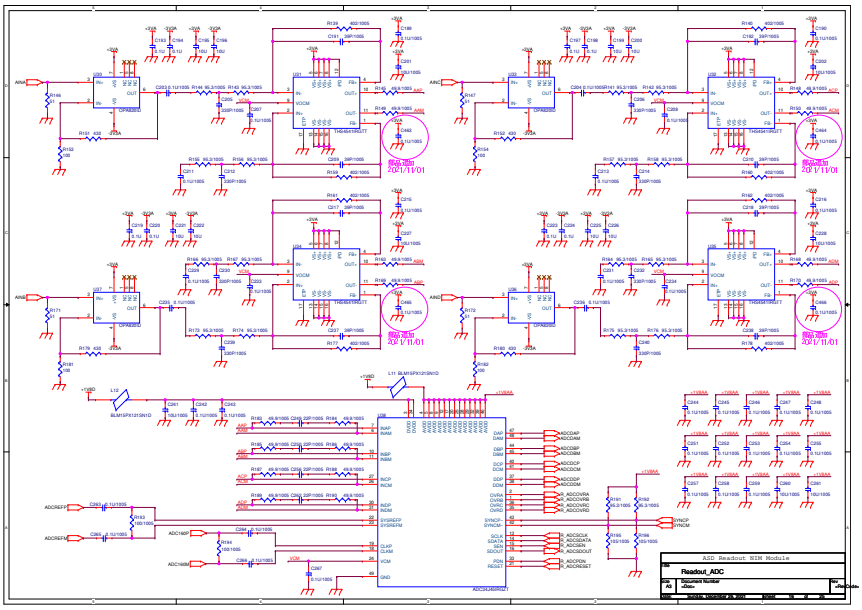


図 A.16: ASD Readout NIM Module メインボードの回路図。Readout ADC ページ。アナログ信号読み出しに使用。オペアンプを用いた波形整形回路と ADC により構成。出力は Zynq の GTX バンクに接続。

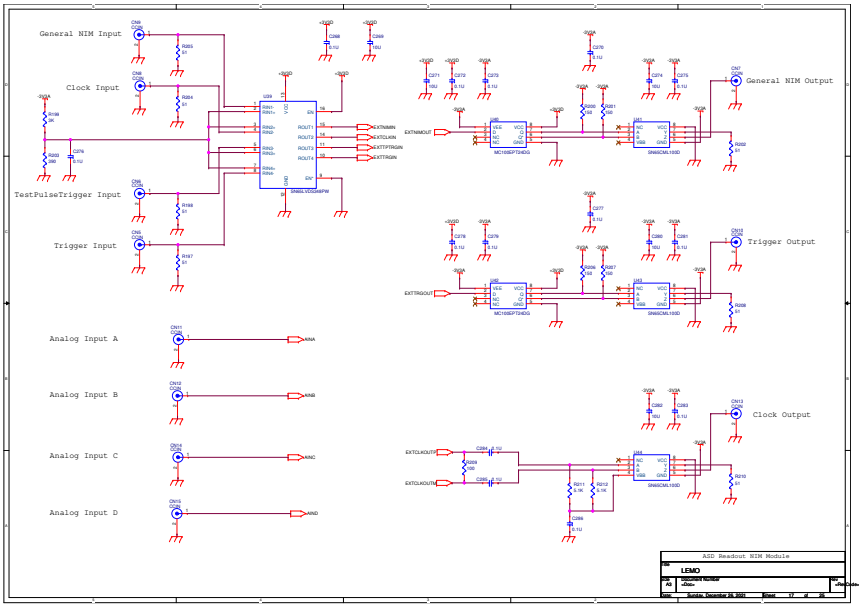


図 A.17: ASD Readout NIM Module メインボードの回路図。LEMO ページ。外部との NIM 信号の送受信に使用。アナログ信号の入力も行う。クロック送受信、トリガー送受信も可能。

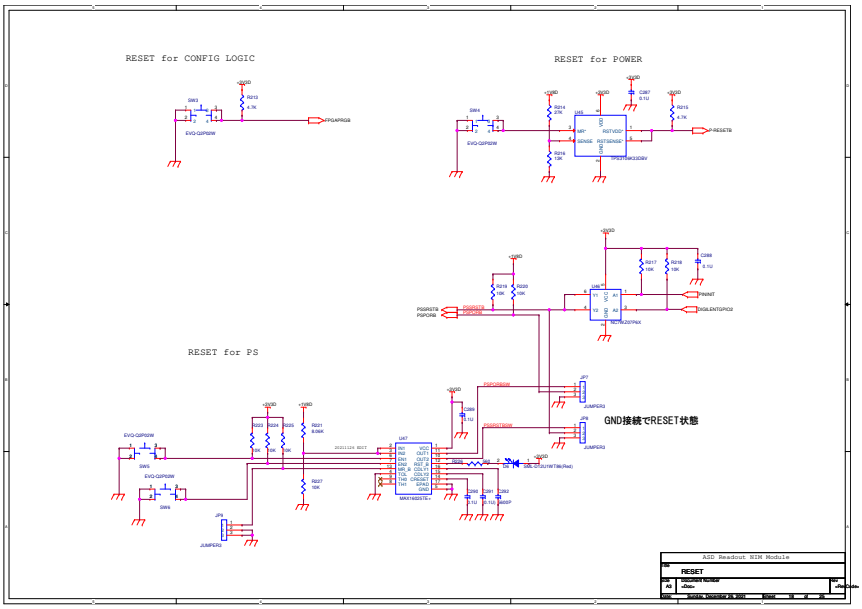


図 A.18: ASD Readout NIM Module メインボードの回路図。RESET 回路。スイッチとジャンパピンにより各 RESET 信号を Zynq に送信。

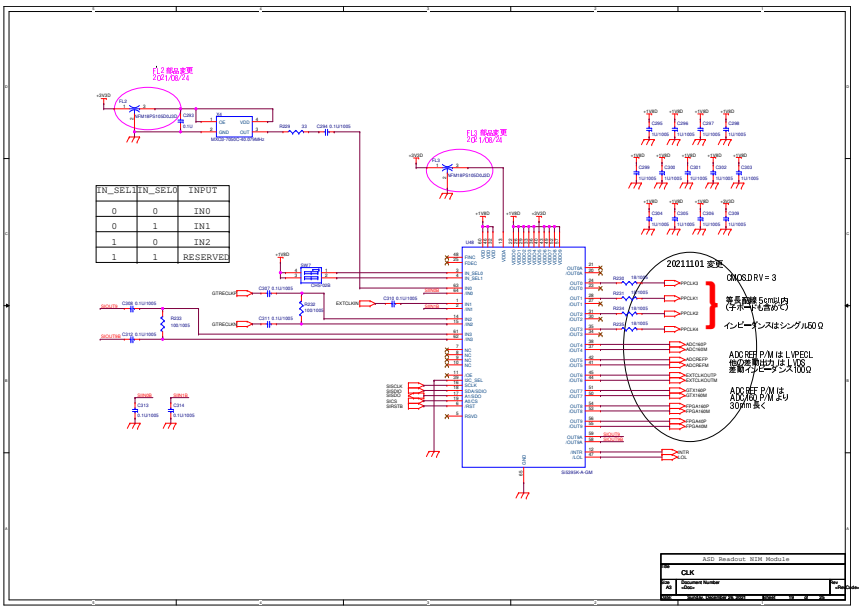


図 A.19: ASD Readout NIM Module メインボードの回路図。CLK ページ。クロック制御及びジッタクリーナーを配置。水晶 40.079 MHz、外部クロック入力をクロック源に PPASIC、ADC、Zynq、外部 NIM 信号などにクロックを分配。

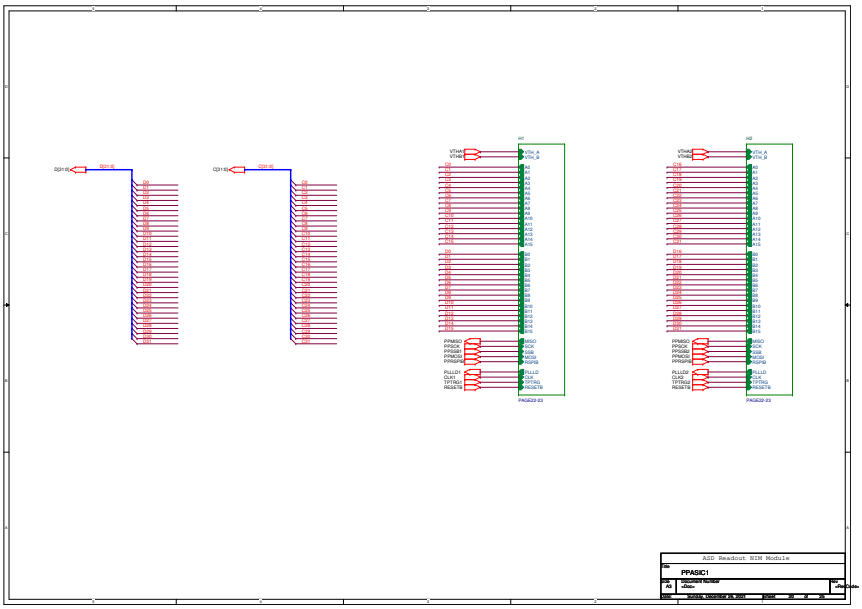


図 A.20: ASD Readout NIM Module メインボードの回路図。Patch-Panel ASIC TOP ページ。以下の階層に A.22,A.23 が存在し、Zynq と接続される。

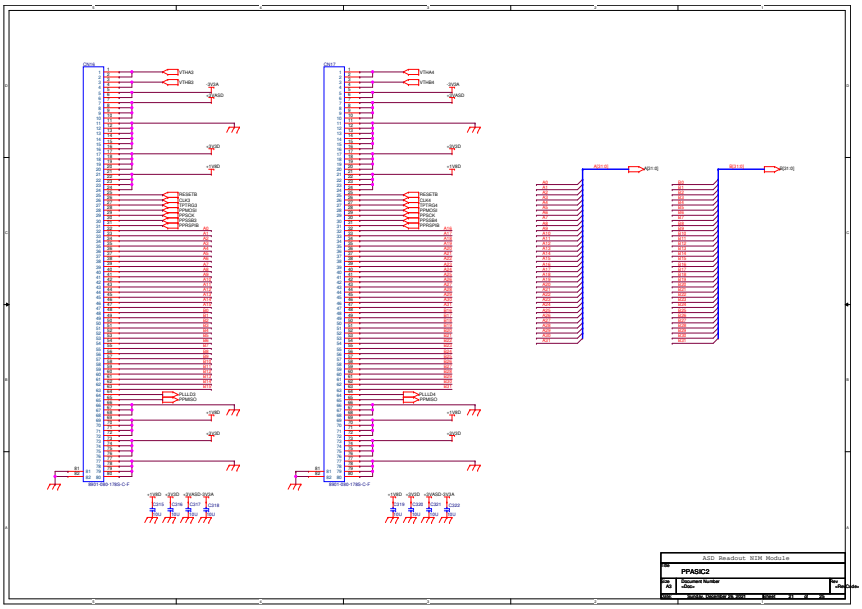


図 A.21: ASD Readout NIM Module メインボードの回路図。サブボード接続ページ。主にサブボードからデータ受信、電源供給。

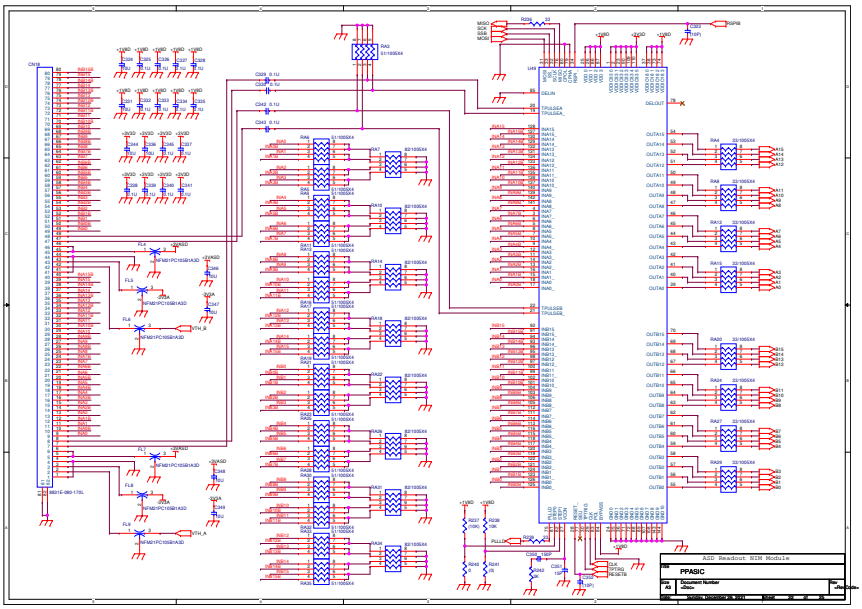


図 A.22: ASD Readout NIM Module メインボードの回路図。Patch-Panel ASIC 1 ページ。フラットケーブルコネクタも配置。

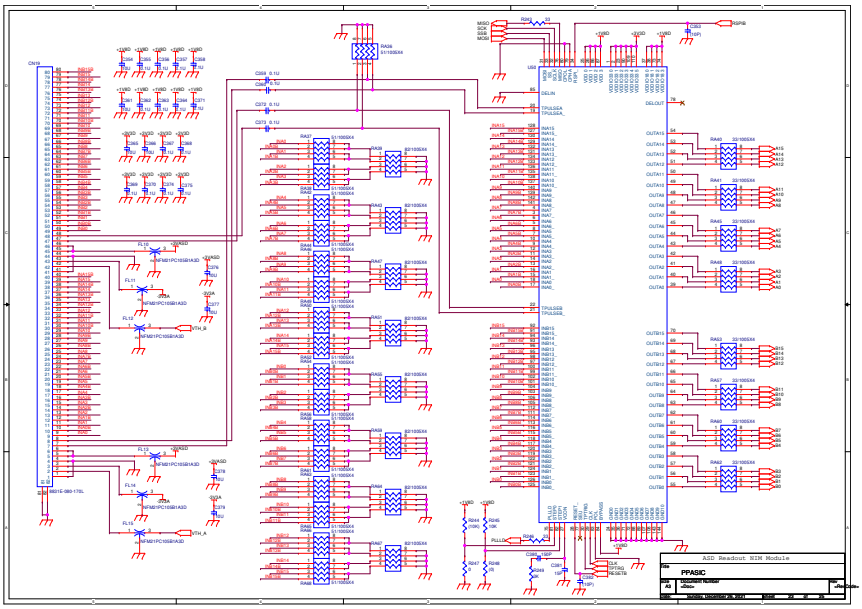


図 A.23: ASD Readout NIM Module メインボードの回路図。Patch-Panel ASIC 2 ページ。フラットケーブルコネクタも配置。

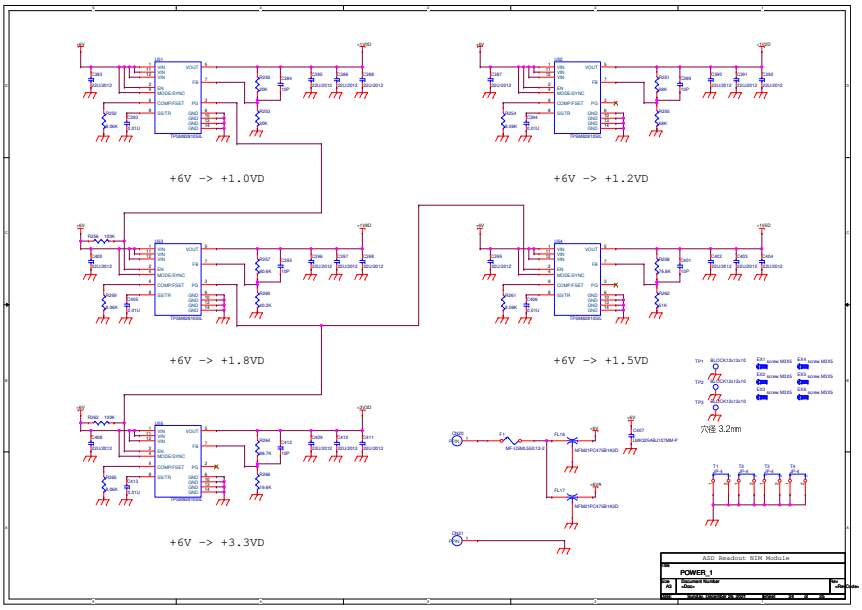


図 A.24: ASD Readout NIM Module メインボードの回路図。POWER 1 ページ。デジタル電源の生成と NIM ラックからの電源取得。Zynq への電源投入シーケンスに従い各ジェネレータを接続。

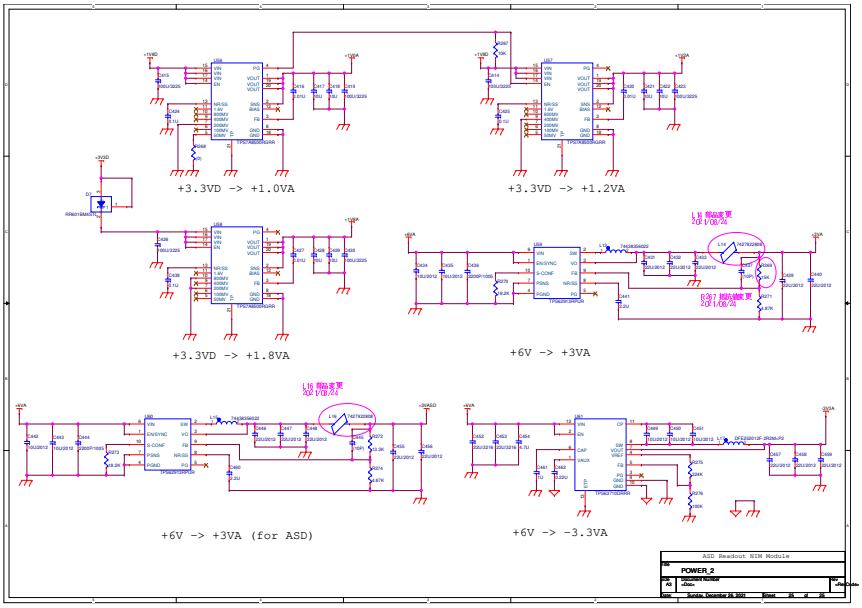


図 A.25: ASD Readout NIM Module メインボードの回路図。POWER 2 ページ。アナログ電源の生成。Zynq への電源投入シーケンスに従い各ジェネレータを接続。

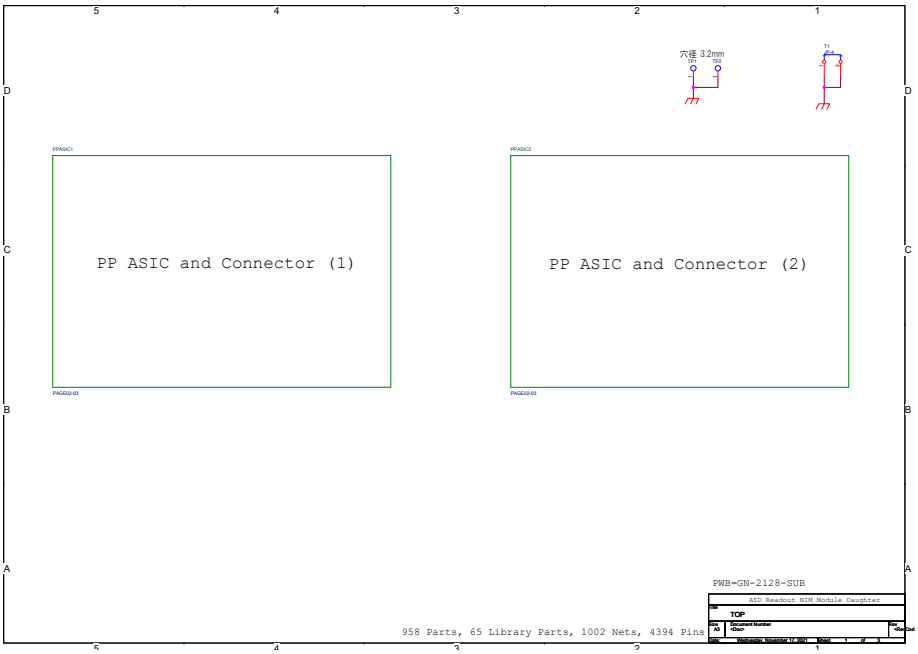


図 A.26: ASD Readout NIM Module サブボードの回路図。TOP ページ。以下 2 枚の回路図を統合。

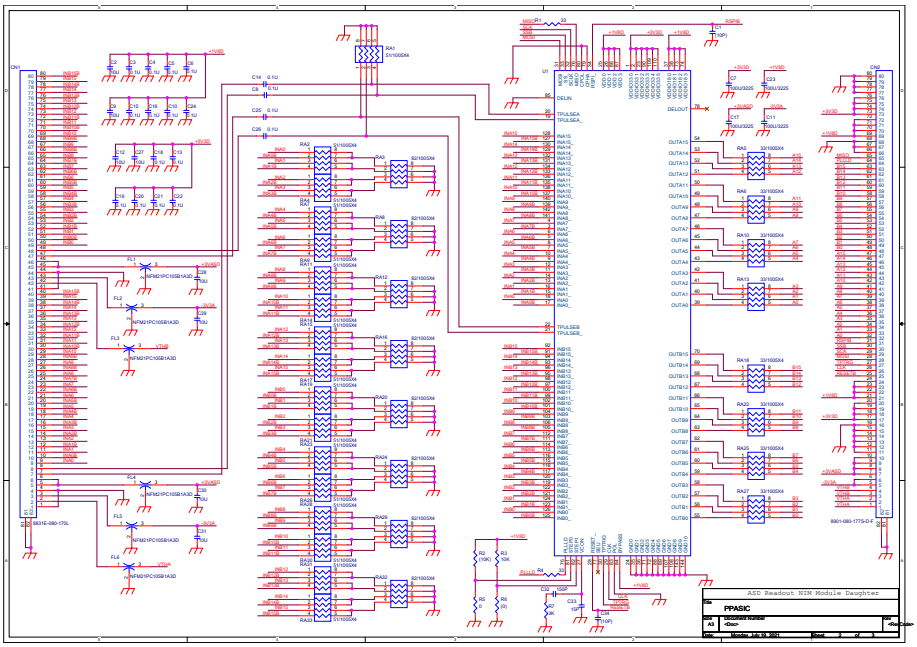


図 A.27: ASD Readout NIM Module サブボードの回路図。Patch-Panel ASIC 1 ページ。フラットケーブルコネクタとメインボードへのコネクタも配置。

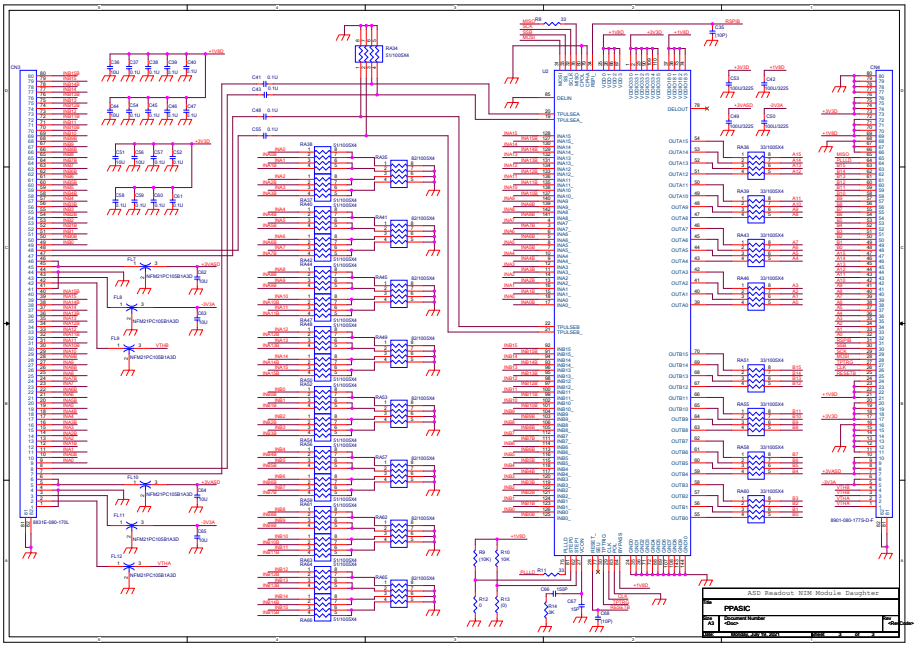


図 A.28: ASD Readout NIM Module サブボードの回路図。Patch-Panel ASIC 2 ページ。フラットケーブルコネクタとメインボードへのコネクタも配置。

謝辞

本研究を行うにあたり、2 年間多くの方々に多大なるご支援を頂きました。この場を借りて深くお礼申し上げます。

指導教員である藏重久弥先生には、ハードウェアに関する初歩中の初歩からご教授を頂きました。本当にありがとうございました。アナログ回路も回路図設計も論理設計も何も分からなかった 2 年前から、少しは成長できたのかと感じるようになれたのは藏重先生のおかげです。研究者としての姿勢だけではなく、普段の研究室生活の中で感じられる真面目さ、優しさ、その他多くのことを学ばさせて頂きました。本当にお世話になりました。

粒子物理学研究室の教員の方には数多くの場で研究に関する助言、ご指導を頂きました。なかでも神戸 ATLAS グループである山﨑祐司先生、越智敦彦先生、前田順平先生にはミーティングを始め、多くのご指導を頂きました。山﨑先生には修士に入る前の卒業実験のときから大変お世話になりました。EASIROC ファームウェア作成やデータ解析を行う中で研究の楽しさを教えて頂きました。思えばあの時に FPGA を触ったことがこの研究をしたいと思うきっかけでした。良い機会をありがとうございました。越智先生には検出器ゼミで多くのことを教えて頂きました。また、アナログ回路設計の際に質問やアドバイスを頂いたおかげでより一層理解が深まりました。前田先生には回路設計やスライド作成において本当に多くのご指導を頂きました。また、本研究を進めるにあたってのミーティングにもヨーロッパ時刻では朝早い中参加いただき、たくさんの助言を頂きました。多くの場で気にかけて頂きありがとうございました。また、竹内康雄先生、身内賢太朗先生、鈴木州先生、中野佑樹先生、東野聡先生には、研究グループが異なる中、ミーティング、ゼミや授業、コロキウムの中で多くのことを教えて頂いたおかげで物理に関することだけでなく多くのことを学ぶことができました。先生皆様方のいる研究室で学ぶことができて本当に良かったと思います。

高エネルギー加速器研究機構の佐々木修先生には、本研究のコンセプトから用いる IC、回路シミュレーションなど本当に多くの知識や技術を教えて頂きました。知識の至らない私にも真摯に対応していただき、KEK に出張にいった際には直接回路設計のご指導を頂きました。お忙しい中、隔週のミーティングに参加頂き、私に対して多くの道筋を立てて頂いたように感じます。また、なによりもこのような研究を行うきっかけを頂けたことに心から感謝しています。本当にありがとうございました。同じく KEK の池野正弘先生には、回路図設計、配線図面設計の場で本当にお世話になりました。細部の細部まで私のミスや勘違いの修正を頂きました。回路図設計はこうやるん

だ、という姿勢を何度も学ばせて頂きました。本当にありがとうございました。基板開発のお願いをしている有限会社ジー・エヌ・ディーの皆様にもお世話になりました。細部までこだわった設計、特に等長配線を行っていただき、また素敵なパネルデザインを頂き誠にありがとうございました。東京大学の坂本宏先生には Zynq ファームウェアを触る際にお世話になりました。急なメールにも真摯に返信を頂き、なおかつしっかりとまとめられた研究ログのおかげで効率よく作業を行うことができました。また、同じく東大の奥村恭幸先生にははじめのコンセプト段階からミーティングに同席頂き、何も分からなかった私に優しく教えて頂きました。また、回路設計用の OrCAD や PSpice のインストールやサーバーに関して何度もメールでやり取りをさせて頂き、ありがとうございました。名古屋大学の堀井泰之先生にも本研究においてネックであった TDC の拡張や ZC706 の貸出など至る所で面倒なお願いを引き受けて頂き、心から感謝しています。また、奥村先生、堀井先生には Phase-Ⅱミーティングの中でも発言の機会や質問、助言などをたくさん頂きました。良い機会をありがとうございました。東京大学田中碧人様は困難な状況で連絡を送らせて頂いたときに、いつも適切な助言を頂きました。名古屋大学に在籍されていた山田敏大様には Patch-Panel ASIC や PS ボードについて尋ねた際に多くの資料や回路図を共有いただきました。同じく名古屋大の鍋山友希様には TDC の拡張を行っていただき、本当にお世話になりました。複数回お願いをしてしまったにも関わらず分かりやすい資料をありがとうございました。

粒子物理学研究室では多くの方々とのコミュニケーションの中で、色々なことを学ばせて頂きました。神戸 ATLAS グループの先輩である日比宏明さん、末田皓介さんには同じ居室で色々なお話をさせて頂きました。日比さんには日比さんゼミをはじめ物理の内容や ATLAS に関すること、そして楽しいお酒の嗜み方を教えて頂きました。末田さんには限られていてしかも離散的な期間でしたが、Zynq を中心にハードウェアをたくさん教えて頂きました。特に本論文執筆期間中は些細な疑問や質問に対し、面倒くさがらずに返答頂き、たまには一緒に考え、一緒に調べて頂きました。お二人の後輩で良かったです。同じ神戸 ATLAS グループ後輩の丸本君、中村君はたまに席の近くへ行くと雑談をしてくれたおかげで研究の息抜きになりました。同期の谷口大悟君、長崎大智君、窪田諒君、尾崎博紀君、前田剛志君、Kotsor Yurii 君には研究グループは違えど研究室生活の中でたくさんお世話になりました。皆さんと楽しい研究室生活を送れてよかったです。同じ神戸 ATLAS グループの 3 人には特にお世話になりました。安部草太君にはいつ話に行っても相手をして頂きました。技術的なことも ATLAS のことも詳しく、それでいて謙虚な安部君の姿勢には学ぶことが多かったです。野口健太には学部の頃からお世話になりました。B4 の時の卒業研究や就活など様々な場面で相談しにいくといつも協力してくれる野口君はとても頼もしかったです。寺村七都君には 3 年生のときから、とても友達のいない私とも仲良くしていただき、さすがにありがとうございました。いつもくだらない話に耳を貸してくれてありがとう。楽しかったです。

最後に、サークルをはじめこの六甲で出会ったすべての人と、6 年間大学に通える環境をくれ、いつも遠くから支えてくれた両親、祖母、姉に心からの感謝を申し上げます。

参考文献

[1] Wikipedia. Standard Model. https://en.wikipedia.org/wiki/Standard_Model.

[2] CERN:Home. https://home.cern/.

[3] The Large Hadron Collider. https://home.cern/science/accelerators/large-hadron-collider.

[4] Immersive tour of the accelerator complex. https://home.cern/science/accelerators/accelerator-complex/panoramas.

[5] ATLAS Experiment. https://atlas.cern/.

[6] CERN ATLAS HP. Detector and Technology. https://atlas.cern/discover/detector.

[7] ATLAS Collaboration. The ATLAS Experiment at the CERN Large Hadron Collider. JINST 3 S08003, 2008.

[8] Zynq-7000 SoC. https://japan.xilinx.com/products/silicon-devices/soc/zynq-7000.html.

[9] IP-Xilinx. https://japan.xilinx.com/products/intellectual-property.html.

[10] Clocking Wizard-Xilinx. https://japan.xilinx.com/products/intellectual-property/clocking_wizard.html.

[11] Integrated Logic Analyzer (ILA)-Xilinx. https://japan.xilinx.com/products/intellectual-property/ila.html.

[12] 7 Series FPGAs GTX/GTH Transceivers User Guide. https://www.xilinx.com/support/documentation/user_guides/ug476_7Series_Transceivers.pdf.

[13] Wikipedia. UNIX. https://ja.wikipedia.org/wiki/UNIX.

[14] Zynq-7000 SoC Family Product Selection Guide. https://japan.xilinx.com/content/dam/xilinx/support/documentation/selection-guides/zynq-7000-product-selection-guide.pdf.

[15] arm Developer. AMBA AXI and ACE Protocol Specification Version H.c. https://developer.arm.com/documentation/ihi0022/hc/.

[16] Xilinx. PetaLinux ツール. https://japan.xilinx.com/products/design-tools/embedded-software/petalinux-sdk.html.

[17] Zynq-7000 SoC Technical Reference Manual. https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf.

[18] 堀口楠日. 修士学位論文 Zynq 搭載汎用モジュール PT-Z の開発と LHC-ATLAS 実験への応用. 2019.

[19] 田中碧人. 修士学位論文 System-on-a-Chip を用いたエレクトロニクス制御回路の開発- 高放射線環境での大規模システムへの応用. 2021.

[20] ATLAS Thin Gap Chamber Amplifier-Shaper-Discriminator ICs and ASD Boards. https://twiki.cern.ch/twiki/pub/Atlas/TgcDocument/ASD-PRR_v19991001.pdf.

[21] TGC Electronics Group. TGC Patch-Panel ASIC Design Report for Production Readiness Review. 2019.

[22] Wikipedia. シリアル・ペリフェラル・インタフェース. https://ja.wikipedia.org/wiki/シリアル・ペリフェラル・インタフェース.

[23] Open-It アトラス実験 MDT μ粒子検出器トリガー用 TDC. http://openit.kek.jp/project/atlas-mdt-trigger-tdc/atlas-mdt-trigger-tdc.

[24] INNOTECH. OrCAD PSPICE. https://www.innotech.co.jp/products/orcad/products/orcad-ee-designer/.

[25] ADC34J4x Quad-Channel, 14-Bit, 50-MSPS to 160-MSPS, Analog-to-Digital Converter with a JESD204B Interface datasheet (Rev. B). https://www.tij.co.jp/jp/lit/ds/symlink/adc34j45.pdf?ts=1641826972018&ref_url=https%253A%252F%252Fwww.ti.com%252Fstore%252Fti%252Fja-jp%252Fp%252Fproduct%252F%253Fp%253DADC34J45IRGZT.

[26] Silicon Labs. ClockBuilder Pro Software. https://pages.silabs.com/clockbuilder-pro-software.html.

[27] CADENCE. OrCAD Capture. https://www.orcad.com/jp/products/orcad-capture/overview.

[28] Xilinx. Vivado. https://japan.xilinx.com/developer/products/vivado.html.

[29] 7 series FPGA Transceivers Wizard. https://japan.xilinx.com/products/intellectual-property/7-series_fpga_transceivers_wizard.html.

[30] JESD204-Xilinx. https://japan.xilinx.com/products/intellectual-property/ef-di-jesd204.html.

[31] Xilinx Zynq-7000 SoC ZC706 Evaluation Kit. https://www.xilinx.com/products/boards-and-kits/ek-z7-zc706-g.html.

[32] Microsoft. Windows Subsystem for Linux Documentation. https://docs.microsoft.com/en-us/windows/wsl/.