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

(Development of a System-on-a-Chip based control board - Application to a large-scale electronics system in a high radiation area)

東京大学大学院
理学系研究科 物理学専攻
石野研究室
田中 碧人

2021 年 1 月

概要

2027 年開始予定の高輝度 LHC-ATLAS 実験に備え、ATLAS 検出器は大幅にアップグレードされる。ATLAS 検出器のエンドキャップ部に位置し、陽子衝突点から飛来するミューオンの飛来位置と運動量を測定する Thin Gap Chamber (TGC) 検出器においても、読み出し・トリガー回路システムがアップグレードされる。各フロントエンド回路では FPGA が使用され、放射線環境下に置かれる。FPGA のコンフィギュレーションに加え、Single Event Upset (SEU) を監視し必要に応じて自動修復を行う新たな制御系が必要となる。この要求に対しフロントエンド回路を監視・管理するハブモジュールを中心とした制御系の開発を進めている。

本研究では、FPGA と CPU を組み合わせた Xilinx 社製の Zynq SoC デバイスをメインドライバーとして搭載した制御回路を開発している。制御回路には主に 4 つの機能が実装される。

i. 制御回路の Zynq と ATLAS 実験室外を光ケーブルを介して接続し、イーサネット通信を行う機能
ii. 各フロントエンド回路の FPGA を JTAG 通信でコンフィギュレーションするハブ機能
iii. 回復不可能な SEU が起きたフロントエンド回路へ回復信号を自動で送り修復する機能
iv. 不揮発性メモリの放射線損傷を想定し冗長性を持って起動する機能

本制御回路を含む電気回路システムの実証試験を行うため、制御回路の試作機の設計と製造を行った。また、試作機にて制御回路の全機能の動作試験を達成した。更に、動作試験の結果を踏まえて必要な修正を加え、量産機の生産に向けた最終デザインの確立も実現した。

また、本制御回路の開発研究にて得られた成果を通して、大規模システムのコントロールと信頼性の高いシステム構築という点において、Zynq SoC デバイスの素粒子実験における有用性が明らかになった。

目次

図目次

表目次

第 1 章

序章

素粒子物理学において、標準理論は物理現象を説明する上で最も成功した理論体系である。標準理論は、6 種類のクオーク、6 種類のレプトン、4 種類のゲージボソン、そしてヒッグス粒子の全 17 種類の素粒子 (図 1.1) で構成されており、それらの相互作用を “電磁力”、“弱い力”、“強い力” の 3 つの力で記述した理論である。20 世紀の間に素粒子物理実験では数多くの素粒子を発見し、CMS 実験と後述する ATLAS 実験における 2012 年のヒッグス粒子の発見を最後に、標準理論が予言した全 17 種類の素粒子の存在が証明された。しかし、未だ標準理論では説明できない物理現象は存在する。例えば、暗黒物質の正体や、階層性問題、ニュートリノの質量などである。素粒子物理実験では、高エネルギーフロンティアの開拓を行ったり、測定の精度向上を行う等、実験的なアプローチから、これらの未解決問題に挑戦し続けて、標準理論を超える新物理の探索を行っている。

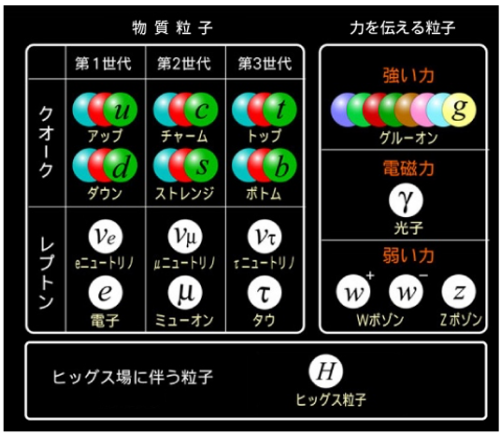


図 1.1　標準理論を構成する素粒子 [1]。

この実験的なアプローチを確固たるものにするには、大規模な素粒子物理実験を安定して遂行できる運用システムが必須である。その運用システムの中でも、本研究は、放射線環境下のシステム制御に焦点を置いた。更にその制御デザインを ATLAS の将来実験に応用した。本論文では、システム制御の設計とその実現に向けた開発研究について述べる。本章は本題に入る前の研究背景として ATLAS の将来実験について述べていく。

1.1　研究背景

スイスのジュネーブ郊外にある欧州原子核研究機構 (CERN) では、全長 27 km の陽子陽子衝突大型円形加速器である Large Hadron Collider (LHC) (図 1.2) が地下 100 m に設置されている。陽子同士を世界最高エネルギーである重心エネルギー 13 TeV で衝突させ、その結果として生じる物理現象を 4 つの衝突点に設置されている検出器で観測している。その 4 つの検出器とは、Large Hadron Collider beauty (LHCb) 検出器、A Large Ion CollidEr (ALICE) 検出器、Compact Muon Solenoid (CMS) 検出器、A Toroidal Lhc ApparatuS (ATLAS) 検出器である。本研究は ATLAS 実験に関わる内容である。

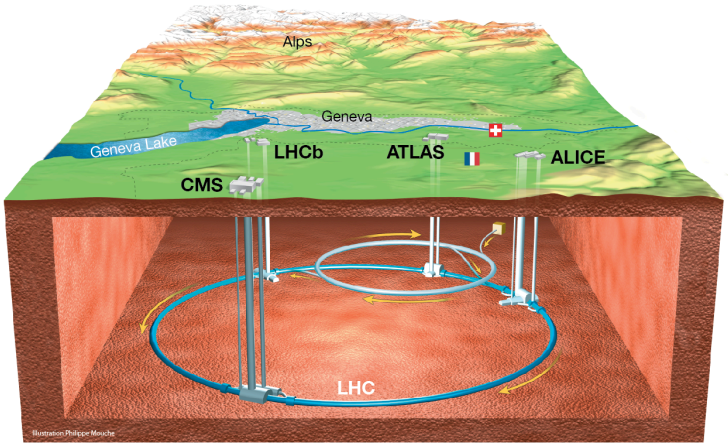


図 1.2　LHC の全体像と 4 つの衝突点 [2]。

1.1.1　ATLAS 実験と TGC 検出器

ATLAS 実験は、CMS 実験と同様に、広い領域での新物理探索とヒッグス粒子の精密測定を主な目的としている。ATLAS 検出器 (図 1.3) は全長 44 m、高さ 25 mの大型検出器である。本研究では両サイドの Endcap 部にある muon trigger 検出器である Thin Gap Chamber (TGC) 検出器 (図 1.3 の赤枠の検出器) のエレクトロニクスシステムの制御に焦点を当てている。

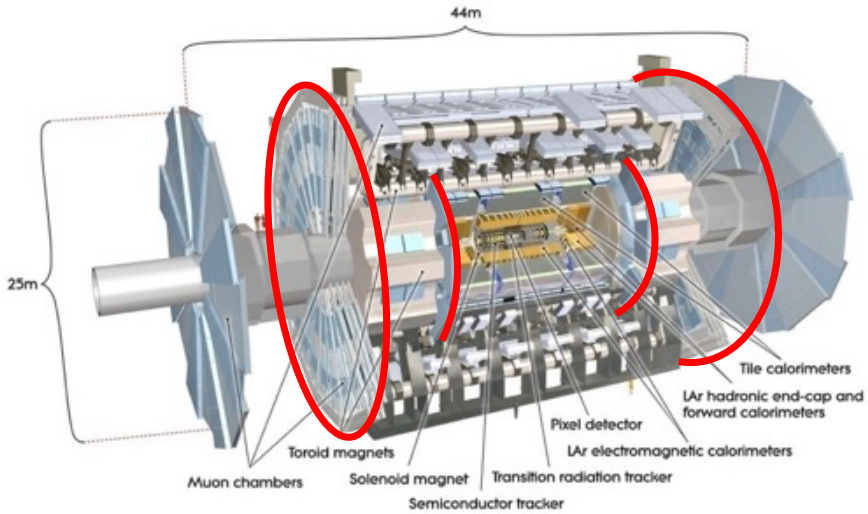


図 1.3　ATLAS 検出器と TGC 検出器 (赤枠)[3]。 Endcap 部外側の円盤状の検出器が Big Wheel (BW) と呼ばれる TGC 検出器、Endcap 部内側の円盤状の検出器が Endcap Inner (EIL4) と呼ばれる Inner TGC 検出器。

1.1.2 ATLAS 実験のオンライン事象選別

LHC では能動的に 40 MHz で陽子同士を衝突させるが、ほとんどの衝突事象は運動量移行の小さいパートンの弾性散乱であり、大きな運動量移行が伴うことが期待される ATLAS 実験の物理解析を行う上で興味の無い事象である。一方、興味のある衝突事象が発生した時は、その衝突で得られた生成物が崩壊して特徴量が出現する。そのため、その特徴量を検出し、重要な衝突事象データのみをオンラインで選別して有限な CERN の Permanent Storage に残す必要がある。また、ハドロンコライダーゆえ、興味のある衝突事象も多岐に渡る。検出する特徴量も様々で、それぞれの衝突事象に応じて包括的に事象選別を行う必要もある。

そこで、ATLAS 実験では、電子、ミューオン、光子、ジェット、消失横運動量などに注目し、2 段階でオンラインの事象選別 (Trigger) をしている。初段 Trigger では、高速で信号処理を行うために、専用の Hardware を用いている。後段 Trigger では、更に厳しい Trigger 判定を行うために、Software を用いている。

1.1.3 高輝度 LHC アップグレード計画

CERN では、現行の LHC 加速器の輝度を大幅に向上させるプロジェクト、「高輝度 LHC アップグレード[*1]」に取り組んでおり、2027 年の運転開始を目指している。高輝度 LHC は 10 年間の運転が計画されており、前運転の Run 3 が終了する 2025 年から新しい装置のインストールが始まる予定である。図 1.4 に高輝度 LHC(HL-LHC) の運転計画が記されてある。

高輝度 LHC アップグレードの特徴は、瞬間最高 Luminosity を現行の約 3 倍の $5\tilde{}7.5\times10^{34}$ $\mathrm{cm^{-2}s^{-1}}$ に向上させ、重心衝突エネルギー 14 TeV にて運転することにある。運転期間全体で 3000 $\mathrm{fb^{-1}}$ 以上の積分 Luminosity に到達する予定である。

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を増大させることである。統計量の増大により大幅な改善がみられる物理解析の例[*2]として、現在 LHC でのみ観測可能なヒッグスボソンの精密測定がある。1 例として、$H \to bb$ 崩壊の際の湯川結合定数の精密測定がある。様々な新物理の理論では、その結合定数が標準理論から逸脱することが示唆されている。そのため、高輝度 LHC の高統計量によって標準理論からの逸脱が精密に観測されれば、新物理モデルを絞り込めるので、素粒子物理の将来の方向性が決められると期待されている。

1.1.4 高輝度 LHC-ATLAS 実験の Trigger and DAQ system

ATLAS 実験でも高輝度 LHC アップグレードに合わせて、検出器やエレクトロニクスの多くが一新され、高輝度環境における物理事象観測に備える。ここで、現行システムの Trigger 性能のまま瞬間最高 Luminosity が増大すると、Trigger をかける横運動量 ($\mathrm{P_T}$) 閾値を上げなければならない。しかし、興味のある物理事象データを取得するためには、P_T 閾値を低く抑える必要がある。そこで、高輝度

[*1] “Phase 2 Upgrade”、“High Luminosity (HL)-LHC Upgrade” とも呼ばれている。本論文では “高輝度 LHC” と呼ぶ。
[*2] 他の例としては、余剰次元理論や超対称性理論にて予言された新粒子の直接探索もある。

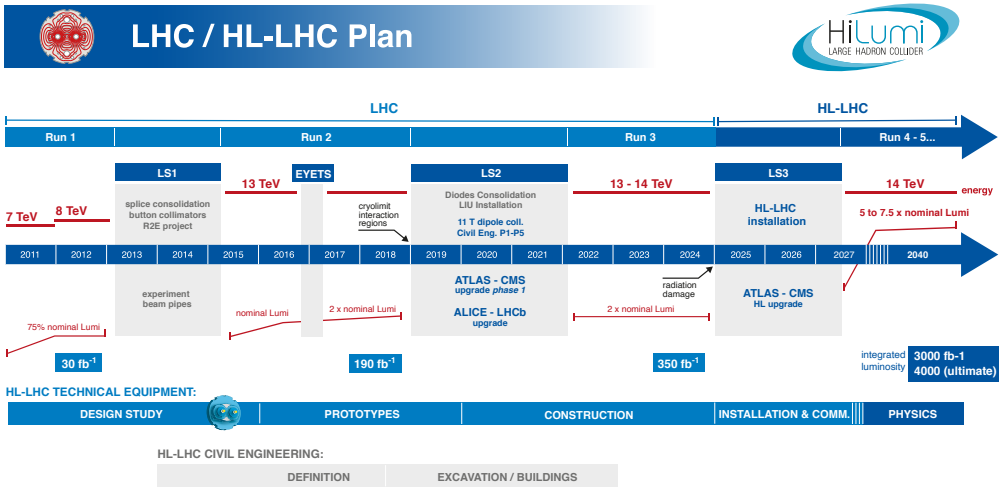


図 1.4 LHC と高輝度 LHC の運転計画 [4]。 高輝度 LHC の装置実装は 2025 年から行われる予定、高輝度 LHC の運転開始は 2027 年の予定。

LHC-ATLAS 実験では、Trigger and DAQ system のエレクトロニクスを一新し、Trigger 性能の向上を目指す。Trigger 性能向上の詳細は 2.1.8 節にて述べる。本節では、高輝度 LHC-ATLAS 実験における Trigger and DAQ system(図 1.5) について説明する。

高輝度 LHC-ATLAS 実験の Trigger and DAQ system は、Level-0 Trigger、DAQ、Event Filter(EF) に分かれる。

高輝度 LHC-ATLAS 実験における Hardware での初段 Trigger 処理は Level-0 trigger と呼ばれており、ミューオン、カロリーメータ、内部飛跡検出器の Trigger 情報を使用して、有益な衝突事象に対して Level-0 Accept 信号 (L0A) を発行する。包括的な Level-0 trigger 処理と L0A の発行は CTP で行われる。そして、CTP から、ATLAS 全体の読み出しを担う FELIX を経由して、L0A 信号と LHC クロックが各検出器システムに送られる。各システムでは、40MHz で捉えた衝突データを Buffer に保管しており、L0A 信号を受けた場合は、該当する衝突データを FELIX へ送る。FELIX は L0A によって送られた読み出しデータを受け取り、後段 Trigger へ送る。後段 Trigger では EF がソフトウェアベースで DAQ system 内のデータの Trigger 処理を行う。最後まで残った衝突事象データが CERN の Permanent Storage に保存される。

TGC 検出器は ATLAS 検出器の Endcap 部に飛来するミューオンを検出し、運動量と飛来位置を概算し、Trigger 情報を初段 Trigger システムに送る。そのため、図 1.5 のように高輝度 LHC-ATLAS 実験において Level-0 muon trigger system の検出器として機能する。このアップグレードに際して、TGC 検出器のエレクトロニクスシステムも一新される。Level-0 Muon Trigger System の Endcap 部をカバーしているので、TGC 検出器の読み出し・トリガーエレクトロニクスシステムは、本論文では Level-0 endcap muon trigger system と呼ぶ。第 2 章にて、Level-0 endcap muon trigger system について詳細に触れていく。

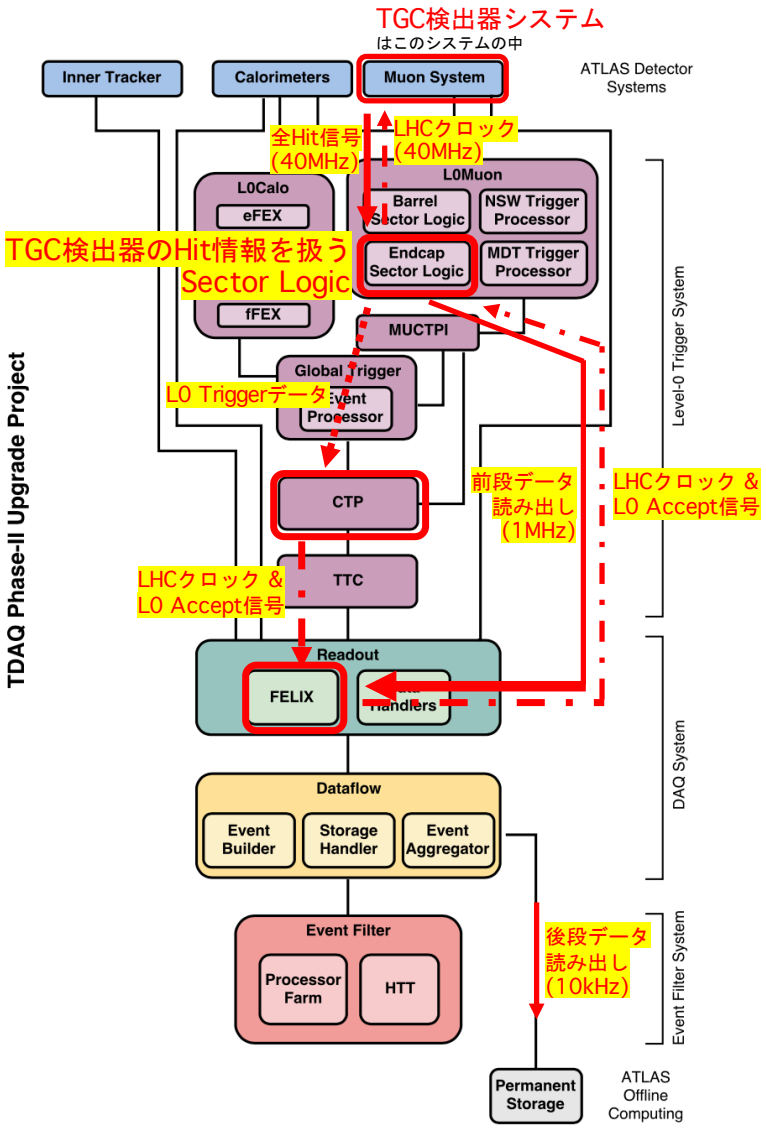


図 1.5　高輝度 LHC-ATLAS 実験の Trigger and DAQ system の概要 [5]。本研究対象の TGC 検出器エレクトロニクスに関わる箇所とデータの流れを赤字で示す。

1.2　研究目的

高輝度 LHC-ATLAS 実験における Level-0 endcap muon trigger system のフロントエンド回路では、TGC 検出器の全ヒット情報を高速光通信 (8 Gbps) でバックエンド回路へ送信する。その実現のため、新しくフロントエンド回路に FPGA を搭載している。この FPGA は、2.2 節にて議論するように、物理的にアクセスが難しい場所に設置される予定のため、外部から FPGA の Firmware のプログラムやデバック、FPGA の reconfiguration 用 flash memory のプログラムが行える必要がある。また、フロントエンドにおける FPGA の放射線損傷 Single Event Upset(SEU) は、高輝度化において更に厳しいものと

なるのだが、フロントエンド回路の安定した運用を保障するため、自己修復できない FPGA の放射線損傷を外部から対応する必要がある (2.2.2.1 節参照)。以上の要請を満たすような制御システムをデザインすることが、本研究の目的である。具体的には、モジュールの考案や Cabling の策定などシステムをデザインし、制御システムの中心として機能する制御回路を開発し、本番での制御回路の運用を可能にすることを目指す。

1.3 本論文の構成

第 2 章では、Level-0 endcap muon trigger system について詳細に説明し、その中でフロントエンド側の要請を明示し、その要請を満たすようデザインした TGC エレクトロニクス制御システムの全体像を示す。第 3 章では、デザインした制御システムの中心的な役割を担う制御回路の設計概要と回路の設計、試作機製作について説明する。第 4 章では、制御回路試作機の機能実装と動作試験を行い、自分が開発実装した機能が自分の想定通り動いたことを述べる。第 5 章では、動作試験によって浮かび上がった試作機の改善点をまとめた後、第 2 試作機の製作について述べる。最後に、第 6 章では、本論文の結論と今後の展望を述べる。

第 2 章

高輝度 LHC-ATLAS 実験における TGC 検出器エレクトロニクスシステム

本章では、本研究の対象である TGC 検出器のエレクトロニクスシステムについて説明する。また、TGC 検出器のエレクトロニクスシステムにおいて求められる機能を明確にし、制御系デザインの確立を実施したことも説明する。

2.1 Level-0 endcap muon trigger system

TGC 検出器では ATLAS 検出器の Endcap 部をカバーし、飛来する Muon 粒子を検出し、その Hit 情報を Hardware 的に処理した後に全体の Level-0 trigger 処理システムに送る。本節は、TGC 検出器の Hit 情報を全体の Level-0 trigger 処理システムへ送るまでの、読み出し・トリガーエレクトロニクスシステム (Level-0 endcap muon trigger system)[*1]について説明する。

このシステムの全体像を図 2.1 に示す。まず、複数層の TGC 検出器で荷電粒子が検出された際、アナログ電流信号が出力される。TGC 検出器に取り付けられている ASD でその電流信号を電圧信号に変換増幅し、閾値電圧との比較を行い LVDS 信号として出力する (以降 “Hit 信号”)。Hit 信号は TGC 検出器の円盤上に設置されている PS board に送られる。他の PS board に送られた Hit 信号とのタイミングのズレを考慮し、PS board 上の PP ASIC でタイミング調節が行われる。そして PS board 上の Field Programmable Gate Array(FPGA) から高速光通信を用いて、60 ˜100 m 離れたバックエンド回路室の Sector Logic(SL) へ光ケーブルで全 Hit 信号が送られる。SL では、TGC 検出器の各地点から送られてきた Hit 信号を元に荷電粒子の横運動量と位置を算出し、L0 Trigger データを全体の Level-0 trigger system に渡す。また、SL では、FELIX から LHC クロックと L0A 信号が配布されるので、L0A 信号に従って Hit 情報を FELIX に読み出す。更に、SL から PS board へ光ケーブルを用いて LHC クロックの配布を行ったり、SL から PS board 上の register の操作を行える設計になっている。以降、詳細を述べる。

[*1] 制御系は含まずに説明する

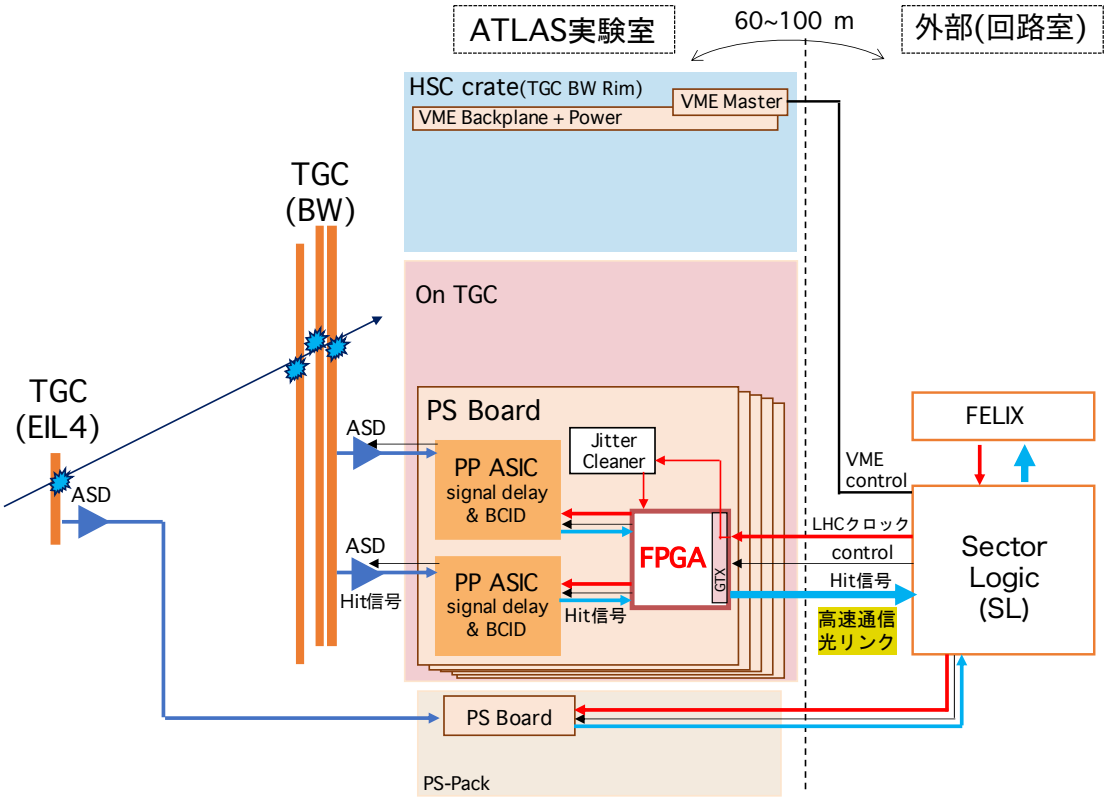


図 2.1 Level-0 endcap muon trigger system の全体概念図。

2.1.1 TGC 検出器とフロントエンド機器の位置関係

TGC 検出器の全体写真は図 2.2、TGC 検出器設置位置は図 2.3 にて示す。TGC 検出器は ATLAS 検出器の Endcap 部をカバーするようになっている。ATLAS 検出器の最外層には 3 層の円盤状の Big Wheel (BW) があり、衝突点に近い方から M1, M2, M3 と呼ばれている。内側には Endcap Inner (EIL4) が設置されている。外側の BW の直径は 25 m となっており、ビーム軸上の位置は衝突点から 13 m (M1)、14 m (M2)、14.5 m (M3) にある。内側の EIL4 は直径 12 m となっており、ビーム軸上の位置は衝突点から 7.5 m に設置されている。

図 2.2 に写っている検出器上に並べて設置された白い “PS-Pack” の中に後述する PS board がある。図 2.3 の M1、M3 の側面にある白い四角が、PS board が格納されている PS-Pack の設置位置である。また、図 2.2、図 2.3 の M1 の先端、つまり M1 の周縁部分には mini-Rack が片サイド 12 台設置されている。

2.1.2 TGC 検出器

TGC 検出器は、Multi-Wired Proportional Chamber (MWPC) 型のガス検出器である。TGC 検出器の断面図は図 2.4 の左図のようになっており、アノードワイヤーとピックアップストリップ電極と

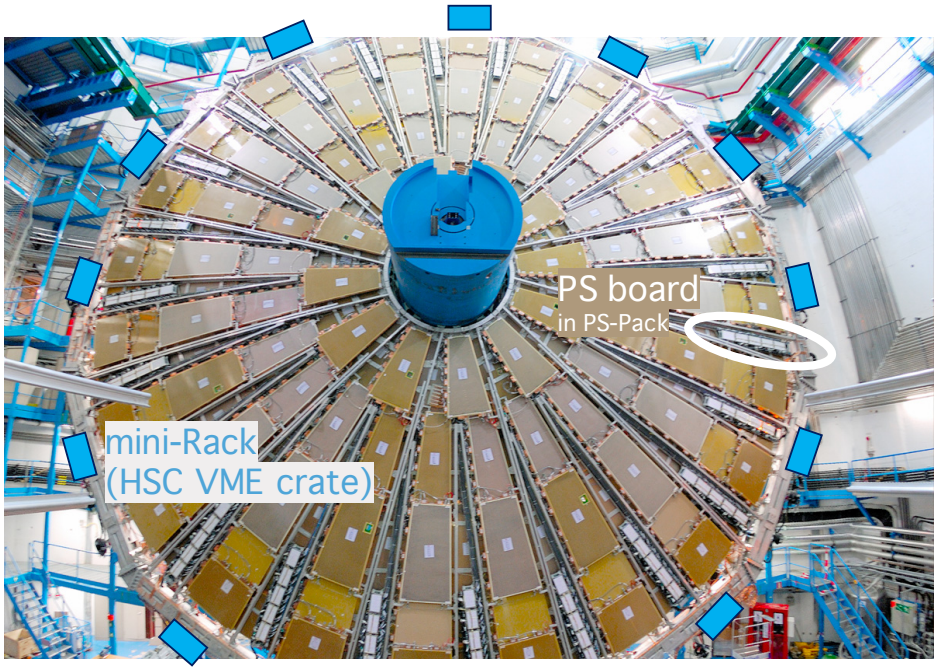


図 2.2 TGC 検出器の正面写真 (M1)。検出器の隙間に並んだ PS-Pack に PS board が設置される。M1 の周縁部には mini-Rack が片サイド 12 台設置されている。

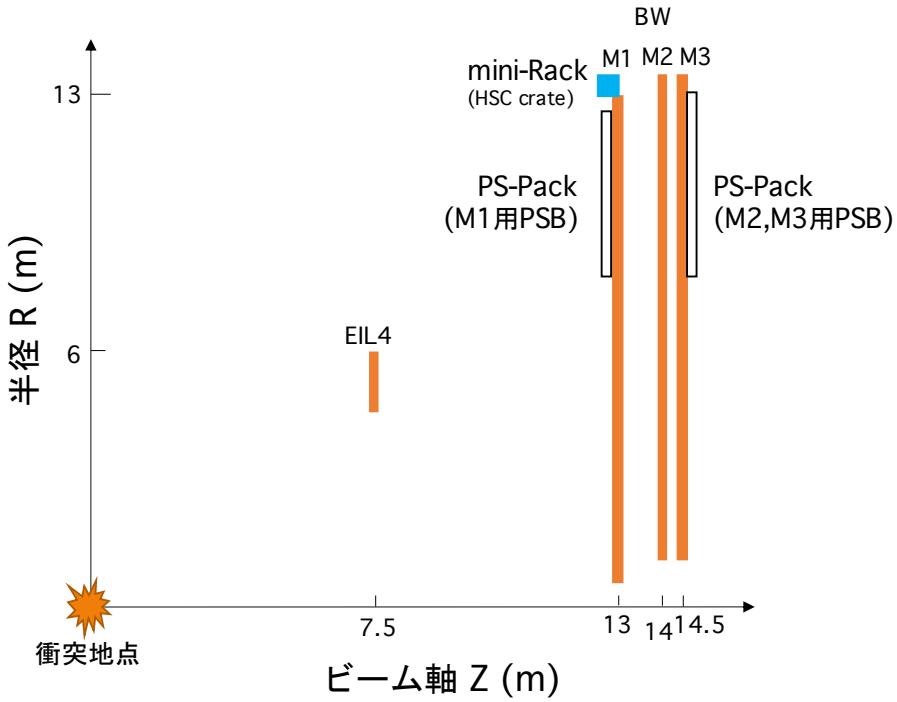


図 2.3 TGC 検出器の側面図。PS board は PS-Pack の中に複数台 (2.3.2 節参照) 設置される。M1 の周縁部には mini-Rack が設置されており HSC VME crate が収納されている。

Graphite の接地層の構造となっている。アノードワイヤーと Graphite 接地層の間には +2.8 kV の高電圧がかかっている。また、このチェンバー内は電子増幅用 CO_2 とクエンチャー用 n-ペンタンの混合ガス[*2]が充填されている。

TGC 検出器に荷電粒子が通過した際、ガス分子が荷電粒子により電離される。電離した電子は、アノード・カソード間の印加電圧による電場に従って、アノードワイヤーの方向に移動する。アノードワイヤー近傍では、強い電場により、電離した電子の通過を引き金にガスの電離が発生し、電子雪崩が起きて、電子が増幅される。この時発生した正イオンが陰極に向けてドリフトしていく電流信号がアノードワイヤーで検出される。また、鏡像電荷がカソードストリップで検出される。

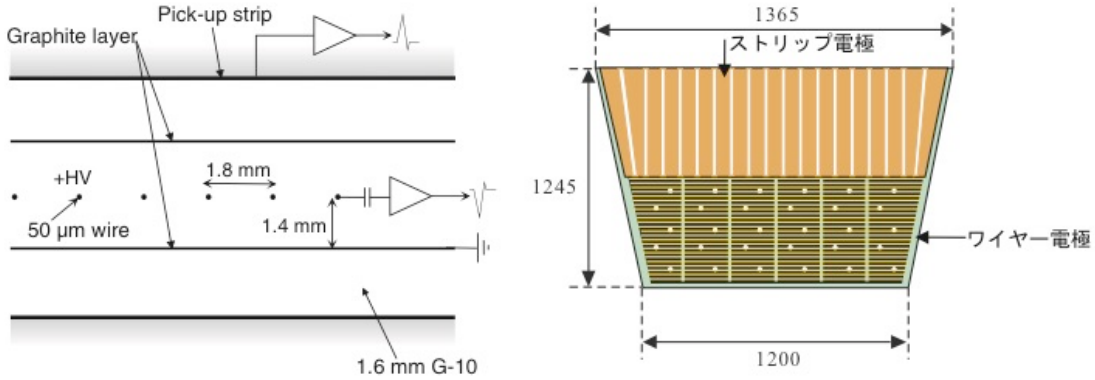


図 2.4 左:TGC 検出器の断面図、右:TGC 検出器の 1 チェンバー

図 2.4 の右図の通り、アノードワイヤーとピックアップストリップは直交して配置されているため、荷電粒子の位置を 2 次元的に読み出すことができる。この台形チェンバーの縁に ASD が設置されており、ASD から Hit 信号が PS board へ読み出される。TGC 検出器の BW や EIL4 は、図 2.4 の右図の台形チェンバーの集合体となっている。ATLAS 実験室内での TGC 検出器のチャンネル数は全体で約 32 万である。

2.1.3 Amplifier Shaper Discriminator (ASD) ボード

ASD は、TGC 検出器のチェンバーから出力されるアナログ電流信号を電圧信号へ変換、増幅、整形した後、閾値電圧以上の電圧信号を PS board の PP ASIC へ渡す。これらの処理は図 2.5 の写真にある 4 つの ASD Application Specific Integrated Circuit (ASIC)[*3]にて行われる。これらで TGC 検出器の 16 のチャンネル数をカバーする。PS board の PP ASIC との通信には、放射線などによるノイズに強く電力消費が少ない、Low Voltage Differential Signaling (LVDS) 通信を採用している。ATLAS 実験室内で実装される ASD は約 2 万台となる。

[*2] CO2: 55 %、n-pentane: 45 %
[*3] ASIC とは特定の用途のために信号処理などを設計し、製造される集積回路である。

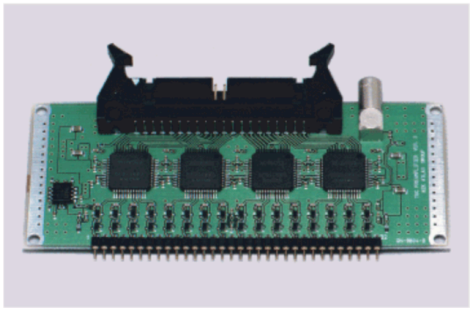


図 2.5 Amplifier Shaper Discriminator (ASD)

2.1.4 PS board

PS board は、Patch-Panel ASIC (PP ASIC) と Field Programmable Gate Array (FPGA) の 2 種類の集積回路が搭載されており、ASD から受けた全 Hit 信号を “Hit Bitmap” としてバックエンド回路室の SL へ高速光通信を用いて送る。ここで、FPGA とは、ユーザーが何度でも configuration[*4]でき、Firmware を編集して内部のデジタル回路を編集できる集積回路である。PS board に供給される電源は、PS board 駆動用 3.3 VD(デジタル) と、ASD との LVDS 通信に使用する +3 VA、-3 VA(アナログ) がある。また、M1 の PS-Pack にある PS board は、TGC 検出器 M1 をカバーし、M3 の PS-Pack にある PS board は、TGC 検出器 M2, M3 をカバーする。高輝度 LHC-ATLAS 実験室内で実装される PS board は合計で 1434 台である (小計は 2.3.2 節)。PS board の概念図は図 2.6 に示す。PS board の第 1 試作機の写真は図 2.7 に示す。

■PP ASIC ATLAS 検出器では各衝突事象毎に Bunch Crossing Identification(BCID) を割り振り、どの衝突事象でのデータであるかをラベル付けできるようにしている。そのため、各 Hit 信号を正しい BCID に振る必要がある。そしてその後、40 MHz で同期する回路にて信号が処理されていく。しかし、TGC 検出器のフロントエンドエレクトロニクスでは、衝突地点から荷電粒子が飛来した際、衝突点と検出器間の TOF の違いや、ASD と PS board 間などのケーブル長の違いにより、Hit 信号が PS board へ到達するタイミングが一致しない。そこで、PP ASIC では、LVDS 規格により ASD からの Hit 信号を受け取ると、Relayable Delay による Hit 信号のタイミング調節と、各 Hit 信号への BCID の割り振りが行われる。

LHC では 40 MHz の頻度で陽子を交差させているのだが、LHC 全システムで機器の動作を同期させるために 40 MHz (25 ns)[*5]の LHC クロックを生成し各システムに配布している。TGC 検出器のエレクトロニクスシステムでは、LHC クロックは、バックエンド回路室の SL から PS board の FPGA を経由して、PP ASIC まで送られてくる。PP ASIC では、この LHC クロックと Hit 信号が同期するように、Hit 信号に delay をかけてタイミング調節を行っている。具体的には、Hit 信号を受けるゲートが開くタイミングをチャンネル毎に 1 ns 以下の精度で微調節し、Hit 信号を LHC クロックの 1 波長分の長さに整える。

[*4] 集積回路中の register を設定すること。
[*5] 正確には 40.079MHz

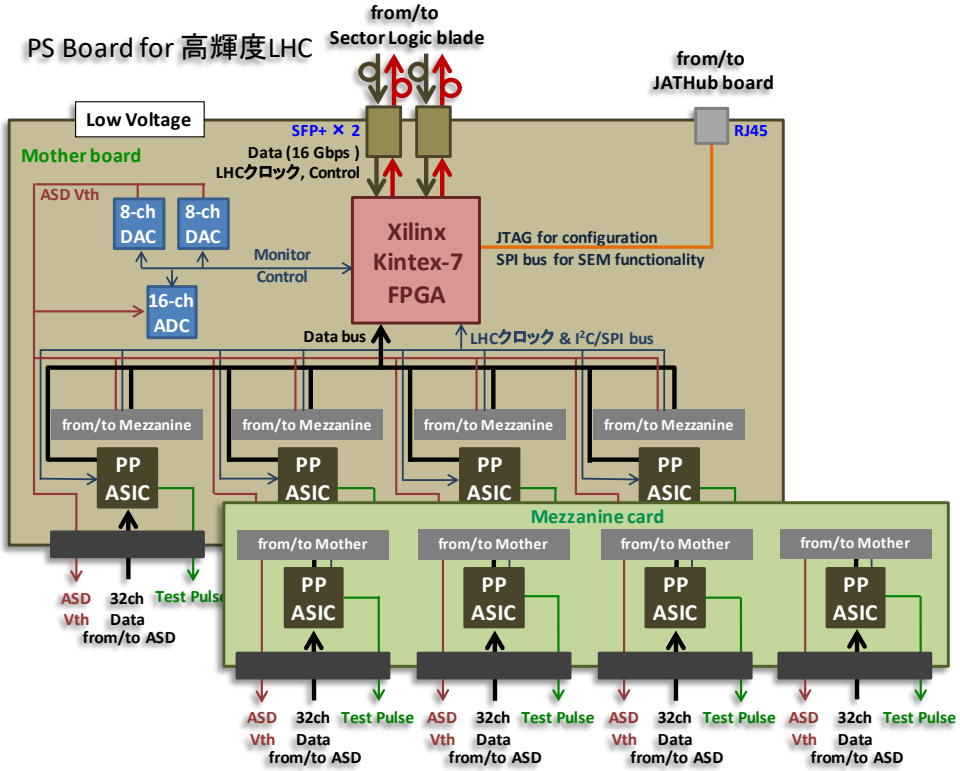


図 2.6　PS board 概念図

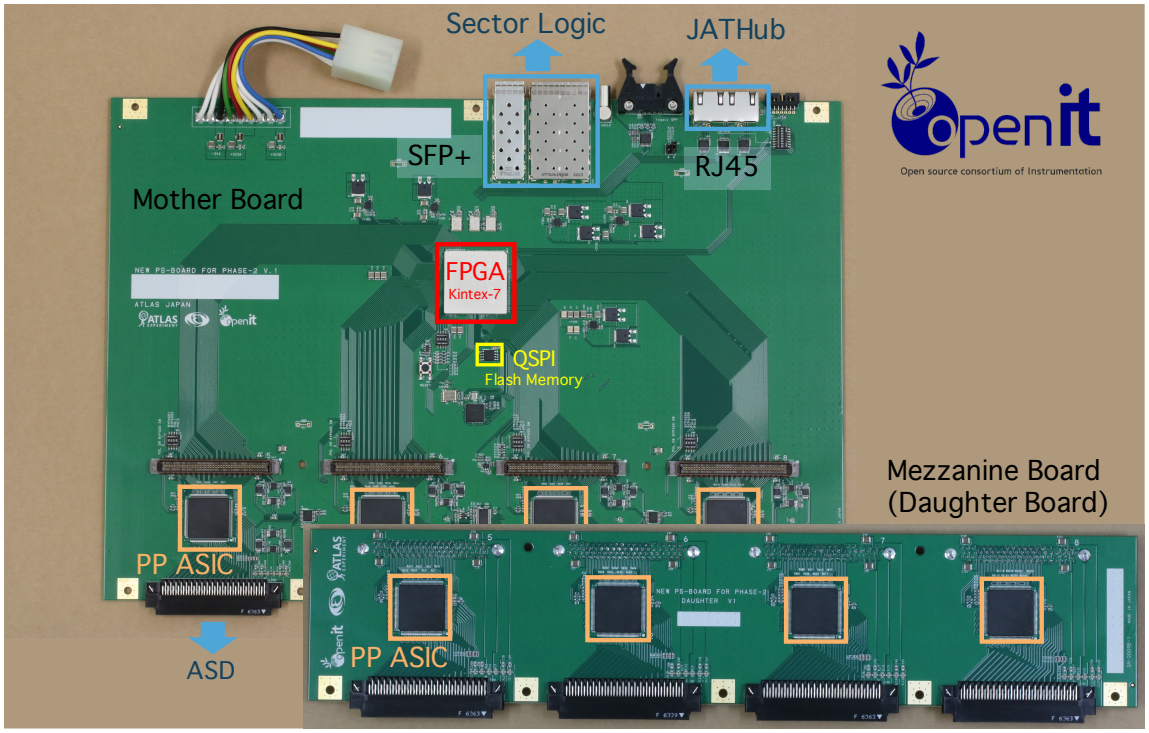


図 2.7　PS board 第 1 試作機の写真

ここで、LHC クロックの位相は全 PS board にて十分小さな精度で揃っていなければならないのだが、上記のタイミング微調節を行うべく、最低でも 1 ns 以下の精度で位相の一致が必要である。現行システムでは、300 ps 程の精度で位相を一致させてシステムが動いた実績がある。

また、PP ASIC には Test Pulse 試験用の機能がある。PP ASIC の register を操作することで ASD に Test pulse trigger を送ることができ、Test Pulse を増幅し AC-Coupling を経て ASD に決まったタイミングでチャージ信号を入力することで、Level-0 endcap muon trigger system に試験的な Hit 信号を通すことができる。

PP ASIC は PS board の mother board に 4 つ、mezzanine board に 4 つ、合計 8 つが設置されており、TGC 検出器の 128 チャンネルをカバーする。

■FPGA on PS board　PP ASIC で調節された全 Hit 信号は、FPGA に内蔵されている高速シリアル通信対応の GTX tranceiver から、PS board 上の SFP 光 tranceiver を経由して光ケーブルを繋げて、1 lane につき 8 Gb/s の高速光通信で 60 - 100 m 先のバックエンド回路室にいる SL へ送られる。また、SL から PS board の FPGA へ、同じ高速光通信で、LHC クロックが配布されたり、FPGA の register を操作する信号が送られる。FPGA の register を操作することで、ASD の閾値電圧の操作や監視、PP ASIC の register 操作を行える。

PS board は FPGA を 1 つ搭載し、FPGA は Xilinx 社の Kintex-7[*6]を使用している。高速光通信に対応する最新の確立された技術を活用するため、FPGA を使用する。

PS board には、ATLAS 実験全体で定められている放射線耐性の要求を満たした QSPI flash memory も載る。QSPI 上には、FPGA のデジタル回路を記述した Firmware が入った configuration file をプログラムして保持することができる。FPGA の configuration memory は Static Random Access Memory (SRAM) で実装されており、電源が切れると内部のデジタル回路は保存されない。そのため、FPGA の電源を入れ直したり、reconfiguration を行う際は、QSPI 上の file から Firmware を読み込み、FPGA に Firmware をプログラムしてデジタル回路を形成する。

2.1.5 HSC (VME) crate

図 2.2 や図 2.3 に示す mini-Rack の中には図 2.8 のような VME crate が収納されており、VME master module と最大 20 枚の VME slave module が挿入できるようになっている。そして、VME master module が VME slave module に対して A24/D16 (AM=0x39) の register 操作を行える。Backplane は VME 規格の J1 とユーザー定義の J3 で構成されている。この VME crate は “HSC crate” と呼ばれている。この HSC crate には 3.3V の電源が供給されている。この Crate は現行システムでも使用[*7]されている。高輝度 LHC-ATLAS 実験でも再利用するが詳細は後述する。

[*6] 型番: XC7K325T-2FFG900C

[*7] 現行のシステムでは crate 内に HPT、SSW と呼ばれる 2 種類の module が実装されているため、“Hpt Ssw Crate(HSC) crate” と呼ばれている。

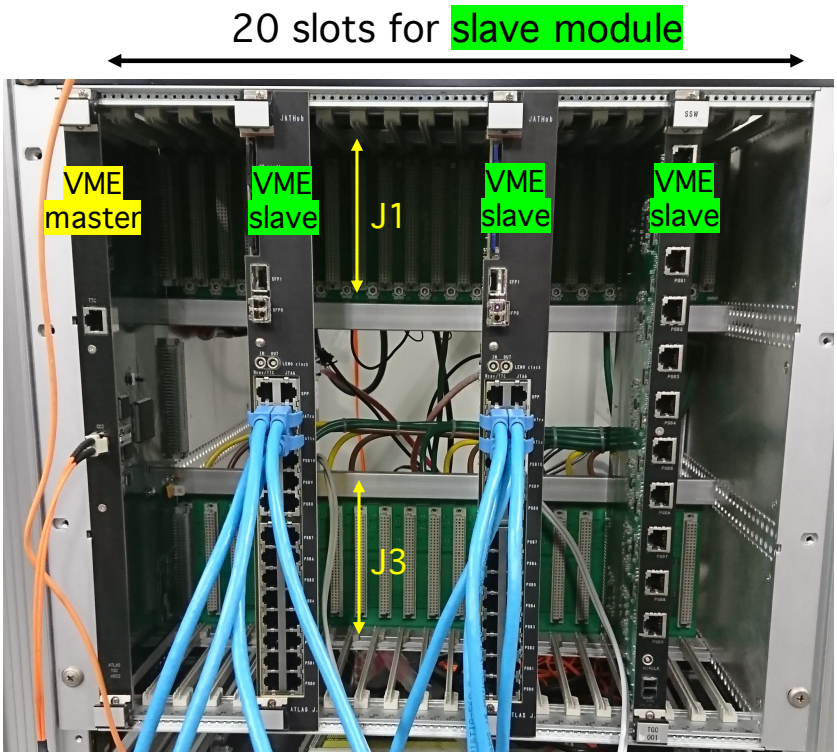


図 2.8　mini-Rack 内にある HSC VME crate。Backplane は J1 と J3 の部分がある。一番左が VME master module で、J1 を通じて他の VME slave module の register 操作を行える。J3 から 3.3V の電源供給を行う。

2.1.6　Sector Logic (SL)

SL の概念図を図 2.9 に示す。SL では、PS board から高速光通信で送られてきた TGC 検出器 BW の Hit 信号を使用して、Firmware による信号処理を行い、Muon 粒子の候補となり得る荷電粒子の飛跡を再構成する。飛跡再構成を行う際、FPGA の RAM に予め TGC 検出器での Hit パターンを記録しておき、BW からの Hit 信号を照らし合わせて荷電粒子の飛跡を特定する手法を取っている。補助的な情報として、ATLAS 実験の endcap 部の muon に関わる他の検出器からの Hit 情報も加味して、飛来した荷電粒子の横運動量と位置などの Hit 情報が算出される。そして、後段の L0 muon trigger 回路へ Level-0 Trigger 情報を送る。

SL は FELIX から Level-0 Accept (L0A) 信号と LHC クロックを受け取り、PS board へ LHC クロックを配布し、L0A 信号に従って該当する BCID の Hit 情報を FELIX へ読み出す。

SL と HSC crate の VME master module を光ケーブルで接続し、SL から VME master module を高速光通信を使用して操作する。

SL は ATLAS 実験室から 60 - 100m 離れた回路室に全 48 台設置されている。ATLAS 実験室は LHC 加速器運転中において放射線環境である一方、この回路室は、実験室と回路室の間に用意されたケーブル用の穴は小さく放射線対策が施されているので、LHC 加速器運転中でもアクセスできる。

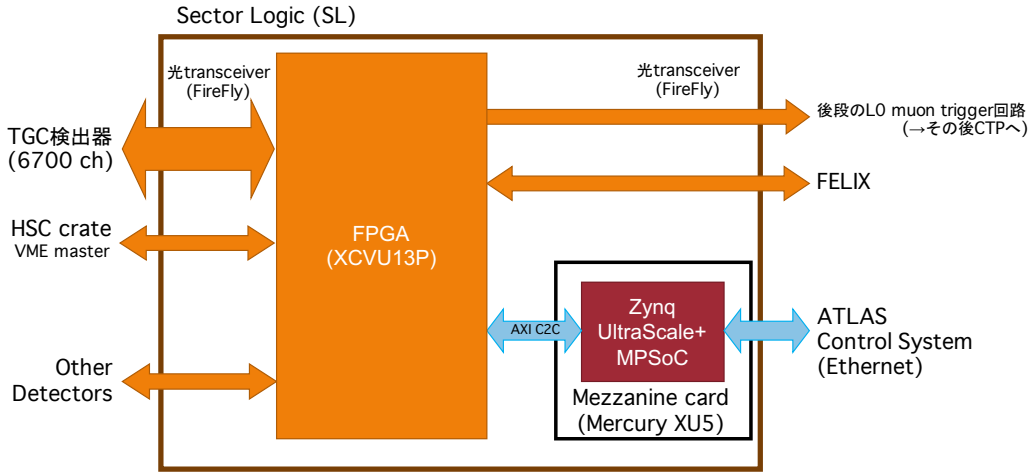


図 2.9 Sector Logic(SL) の概念図。SL と他回路との接続の様子を示す。

SL には、Xilinx 社の FPGA Virtex UltraScale+[*8]を搭載し大規模処理を可能にした。また、ボード上のコントロールを行うため、Xilinx 社の Zynq UltraScale+ MPSoC[*9]を使用する。Zynq UltraScale+ MPSoC は Mezzanine card となっている既存の商品を使用する。SL の試作機は 2020 年 12 月現在製作段階にある。

2.1.7 Front-End LInk eXchange(FELIX)

CTP にて全体の Level-0 trigger 処理が行われると、どの BCID の Hit 情報を解析データとして保存するか検出器全体に教えるための Level-0 Accept (L0A) 信号が発行される。FELIX は、CTP から送られる L0A 信号と LHC クロックを ATLAS 検出器全体に配布する。そして、ATLAS 検出器全体から L0A 信号で許可された BCID の Hit 情報が FELIX へ読み出されるので、FELIX はこれを更に後段へ送る。Software による事象選別が行われる後段の回路へは、1 MHz の頻度で Hit 情報が送られる。

2.1.8 アップグレードによる Performance の向上

高輝度 LHC-ATLAS 実験の Trigger and DAQ system 全体では、エレクトロニクスを一新したことにより、以下 2 つの観点から Trigger 性能の向上を実現する。

- 最新技術の導入により、通信性能を向上させたり、初段 Trigger の処理時間 (レイテンシー) を伸ばす。それにより、Hardware ベースの初段 Trigger で、Software ベースの後段 Trigger と同等レベルの処理が行えるようになり、複雑な Trigger アルゴリズムを実装可能となる。そして、ターゲットとしている物理事象を更に記録することができるようになる。
- LHC では多様な物理過程を解析の対象としている。そこで、現行システムより 10 倍の頻度で Trigger を発行できるようにし、読み出し頻度を向上させる。それにより、可能な限り多くの興味

[*8] 型番: XCVU9P-1FLGA2577E
[*9] 型番: Zynq ZU+ C784

のある物理事象データを残すことができるようになる。

レイテンシーと読み出し頻度の向上は表 2.1 に示す通りである。尚、‘初段読み出し頻度’ は初段 Trigger 後 DAQ システムに読み出される頻度を示し、‘後段読み出し頻度’ は Permanent Storage に読み出される頻度を示す。

表 2.1 Trigger and DAQ system のレイテンシーと読み出し頻度

実験	レイテンシー (μs)	初段読み出し頻度 (kHz)	後段読み出し頻度 (kHz)
高輝度 LHC-ATLAS 実験	10	1000	10
Run3(現行) 実験	2.5	100	1

そして、Trigger 性能を向上させることにより、図 2.10 のように、P_T 閾値を低く抑えることが可能となる。図 2.10 はシミュレーションにより得られた Trigger 性能向上の結果の一例である。横軸は Single lepton の P_T 閾値を示し、縦軸は Trigger 後取得できる Single lepton の量を Trigger をかけない場合と比較した割合 (Acceptance) で示している。尚、凡例はどの物理事象に由来する Single lepton であるかを示す。Trigger 性能が不変の場合、P_T 閾値は 50 GeV に上がってしまうが、本アップグレードにより P_T 閾値は 20 GeV にまで抑えられ、Acceptance が向上する。

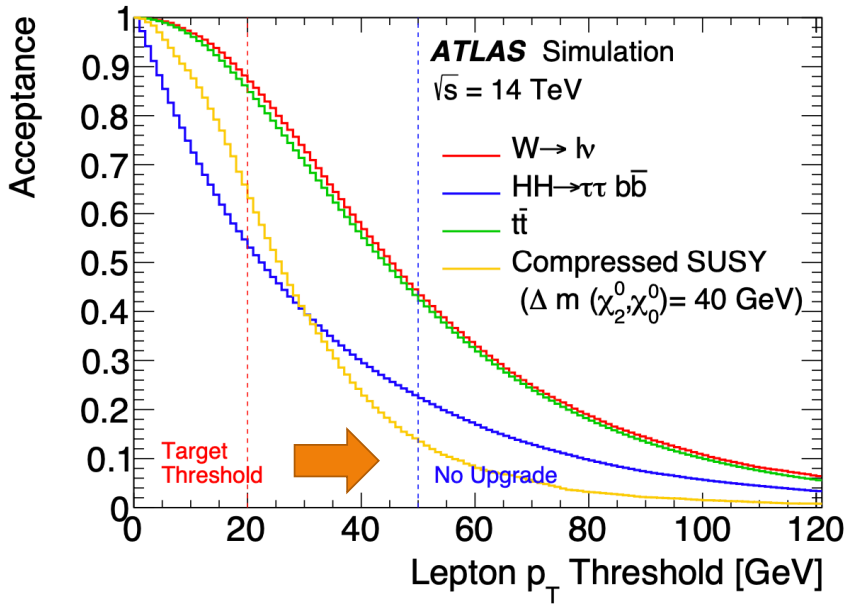


図 2.10 Trigger 性能が向上することにより、物理 Acceptance が向上する例。

Level-0 endcap muon trigger system では、高速データ転送技術を用いて PS board から全 Hit 信号を SL へ送り、そして SL にて一手に Endcap 部の Level-0 muon trigger 処理を行うという機構にすることで、上記 Trigger 性能の向上に貢献している。具体的には、PS board から全 Hit 信号を SL に送る際、性能が上がった最新の FPGA や高速光通信を利用することで、バンド幅による通信制限をかけないように

している。また、PS board ではなく、SL にて Trigger 処理を行うことで、Trigger 処理回路の修正や調節がしやすくなる。そして、SL に読み出し用の大型 Buffer を設けることで、L0A 信号が来るまで十分な量の読み出しデータを保存できる。

2.2 フロントエンド側エレクトロニクスシステムの要請

Level-0 endcap muon trigger system のフロントエンド側にて FPGA が新設される。このエレクトロニクスシステムを高い信頼性を持たせて運転させるためには、ATLAS 実験室内に設置される FPGA を安定して稼働させる必要がある。そのため、Level-0 endcap muon trigger system は、フロントエンドの FPGA の安定した動作を確保するために、外部から以下 2 点の制御が行えることを要請する。

- ATLAS 実験室内の FPGA に対して遠隔で configuration を実行する
- フロントエンドに堅牢で信頼性の高いシステムを構築する

2.2.1 ATLAS 実験室内の FPGA に対して遠隔で configuration を実行

PS board は図 2.13 の PS-Pack に設置される。よって、ATLAS 実験室内に設置される FPGA は、物理的に人のアクセスが容易ではない場所に存在することになる。一般的に FPGA を configuration し内部にデジタル回路を実装するためには、ホスト PC を FPGA の近くまで持っていき、短い JTAG ケーブルでホスト PC と FPGA を接続しなければならない。しかし、PS board の FPGA[*10]では、一般的な方法で configuration を行うことが実質的に不可能である。そのため、ATLAS 実験室外から遠隔で PS board の FPGA を configuration し、FPGA へ Firmware[*11]をプログラム、debug できる仕組みが必要である。

2.2.2 フロントエンドに堅牢で信頼性の高いシステムを構築

2.2.2.1 ATLAS 実験室内の放射線対策

LHC 運転中は ATLAS 実験室内は高い放射線環境となる。TGC 検出器の位置では、1 衝突 (25 ns) 毎に 300˜500 回の Hit が検出される程荷電粒子が飛来してくる。一方、ASIC や FPGA などの集積回路は、イオン化を行う放射線によって内部 register や memory の bit が反転してしまう Single Event Upset (SEU) 事象や、多量の放射線入射により永久的に素子が機能しなくなる損傷など、放射線損傷を被る場合がある。一般に ASIC は放射線に強いと知られているが、フロントエンドに置かれた Level-0 endcap muon trigger system のエレクトロニクスである ASD、PS board は、この放射線損傷への対策が必要である。

ASD ASIC は放射線照射試験にて耐久性に問題がないことを確認した。PS board の PP ASIC に関しては、高輝度 LHC-ATLAS 実験における PS board 設置位置の想定放射線レベル (6 Gy) に安全係数 4.5

[*10] デジタル回路の編集の自由度を担保する必要がある

[*11] FPGA に実装するデジタル回路を記述する回路情報体を指す言葉。TGC 検出器エレクトロニクス開発グループでは Xilinx 社の Vivado ソフトウェアで Hardware Design Language からデジタル回路を記述し Firmware のコンパイルを行っている。

をかけた 27 Gy の中でも安定して稼働するよう耐性のある素子を選択している。試作 PP ASIC による放射線照射試験を行った際は 100 Gy まで異常検知されなかった。そのため、両 ASIC ともに、永久的な素子の損傷に関しての対策は問題ない。Level-0 endcap muon trigger の trigger logic では、荷電粒子の飛跡再構成を行う際、各 TGC 検出器での Hit 信号のコインシデンスを取り検出位置を判断する。そのため、ASD ASIC や PP ASIC での SEU 事象により予期せぬデータが数 bit のみ SL に送られても、SL では問題なく飛跡再構成を行うことが可能である。

PS board の FPGA は、FPGA の configuration memory で SEU が発生した場合、最悪その PS board から出力される全データが途切れる可能性もある。そのため、FPGA 内の SEU を監視し、即座に回復できる仕組みが必須となる。Xilinx は FPGA 上の SEU などの放射線損傷を自動回復できるロジックパッケージ (Intellectual Property(IP))、“Soft Error Mitigation (SEM)” IP core [6] を提供しており、PS board の FPGA にも実装する。この SEM は内部クロックを使用して常時 FPGA configuration memory の bit flip を監視しており、FPGA 内の 1 つの register で SEU が発生した場合、自動で該当の bit を回復させる。だが、例えば FPGA に対して放射線が斜め方向から照射された時など、同時に 2 つ以上の連続する bit で SEU が発生する場合、SEM でも発生したビット反転を回復できない。このような “SEM でも回復不可能な SEU 事象”が発生した時に備え、外部回路で該当する事象を常に監視し、外部から FPGA を reconfiguration する仕掛けが必要となる。また、SL と PS board 間の高速光通信のパスから FPGA の reconfiguration をトリガーすることは、Kintex-7 の仕様上行えないため、別のパスから reconfiguration のトリガーを行う必要がある。

2018 年の ATLAS 実験 Run2 運転の際、高輝度 LHC-ATLAS 実験用の PS board 第 0 試作機[*12]を製作し、PS board 上の FPGA における SEU の発生頻度を調査した。PS board 第 0 試作機は本番と同じ型番の FPGA を使用し、ATLAS 実験室内の TGC 検出器上に設置した。結果は図 2.11 の通り、積分 Luminosity 12.6fb^{-1} で合計 16 回の SEU を検出した [7]。高輝度 LHC 実験での瞬間最高 Luminosity は現行約 3 倍の 5~7.5$\times 10^{34}$cm^{-1}s^{-1} であるため、0.8fb^{-1} あたりに 1 回の SEU が発生するということは、高輝度 LHC-ATLAS 実験における 1 台の PS board の FPGA の SEU 発生頻度は 3 時間に 1 回となる。従って、ATLAS 実験室内全体の 1434 台の PS board の内、10 秒に 1 回、どれかの PS board にて SEU 事象が発生していることになる。

2018 年の ATLAS 実験での SEU 頻度調査試験の際、全ての SEU 事象は SEM によって修正された。しかし、“自動回復不可能な SEU 事象 (UnRecoverable Error (URE))” は観測されなかった。

FPGA Virtex-5 に中性子照射試験を行った実験では、URE 事象が観測され、SEU 事象と URE 事象の比 ~300SEUs/URE が得られた [8]。この比と 2018 年の SEU 頻度調査試験から、高輝度 LHC 実験における PS board 上の FPGA の URE 発生頻度を概算した。1 台の FPGA に対して URE が約 900 時間 (約 37 日) に 1 回、ATLAS 実験室内のどこかの FPGA に対して URE が約 40 分に 1 回発生することが予測できる (詳細は B 章)。

従って、Level-0 endcap muon trigger system の安定稼働のためには、“回復不可能な SEU 事象 (URE)” への対処が必須となる。

[*12] 第 1 試作機との違いは本研究で開発している制御回路への接続口が無い点。

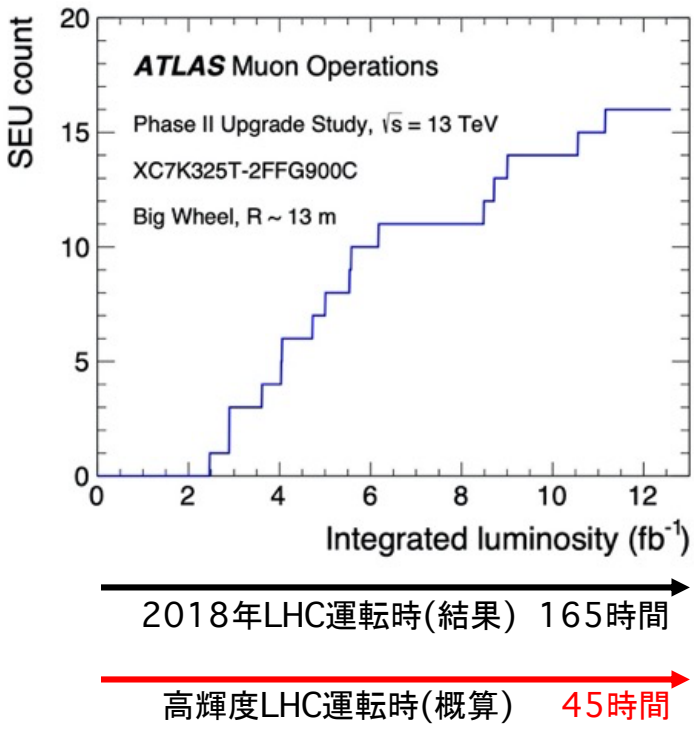


図 2.11 SEU 回数試験の結果 [7] 2018 年の試験。合計 16 回の SEU が発生し、全 SEU 事象は SEM にて修正された。

2.2.2.2 高速光通信の安定的な確立

PS board の FPGA に組み込まれている GTX tranceiver は、PS board に供給される LHC クロックをソースクロックとして稼働している。リンクが途絶えた時、光通信外部からリンク再確立の手続きをトリガーできる仕組みを作っておくことが必須である。そのため、外部から高速光通信をリセットし、リンクを再確立する機能が必要である。

2.2.2.3 LHC クロック位相のモニター

PS board は、BCID を振る際に LHC クロックを使用しており、その LHC クロックの位相が他 PS board と O(100) ps の精度で揃って供給されていることを前提として機能する。そのため、PS board へ配布された LHC クロックを O(100) ps の精度でモニターする必要がある。また、複数の PS board において使用されている各 LHC クロックを同時に比較しモニターできる仕組みも必要となる。

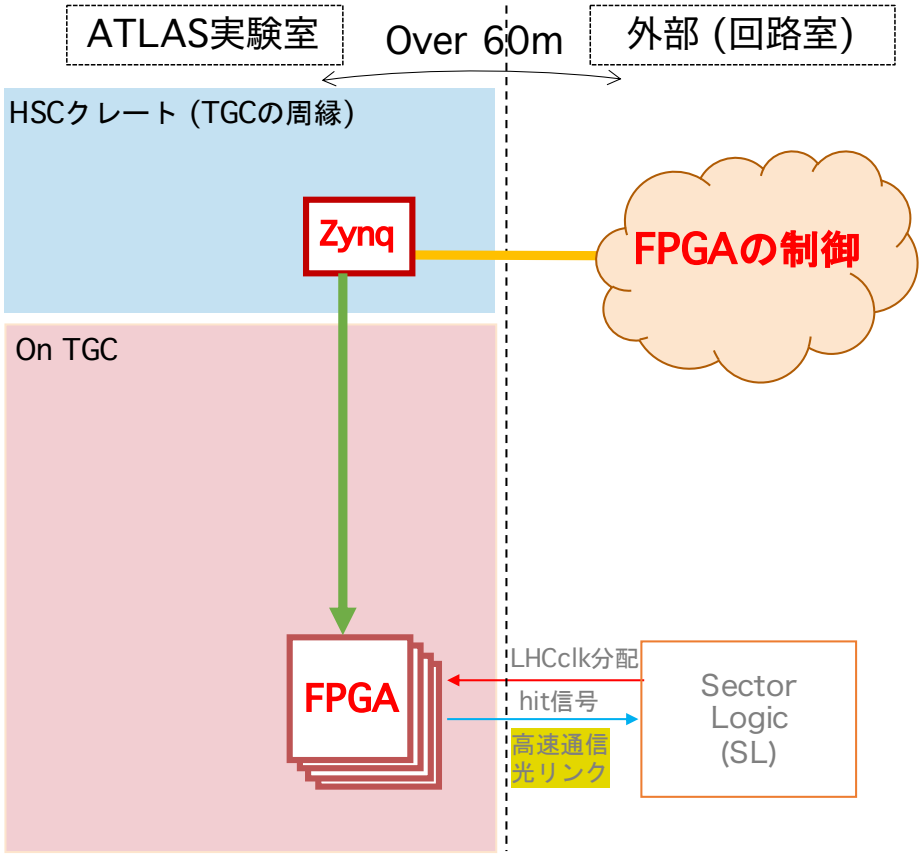


図 2.12 制御系の概念。Zynq を搭載した制御回路を VME crate に挿入し、中継地点として、PS board を制御する。

2.3 制御系の確立

2.3.1 制御系の概要

2.2 節の要請を満たすように、本研究では制御系をデザインした。最適化の結果、TGC 検出器フロントエンドエレクトロニクスでは、図 2.12 のように、FPGA とプロセッサーを組み合わせた System-on-a-Chip(SoC) デバイスを搭載した制御回路を中継地点におき、PS board の FPGA を外部からスローコントロール[*13]で監視、制御できる制御システムを導入することにした。また、実験中と実験外では以下のように用途を使い分けることで、ハイブリッドな制御システムとした。

- **実験時間外**：Linux のソフトウェアアプリケーションにより、実験時間外にて必要な制御のハブとして稼働すること。
- **実験中**：FPGA の Firmware ロジックにより、実験中に必要な機能のハブとして稼働すること。

この制御系に導入される制御回路には、SoC デバイスとして Xilinx 社の Zynq-7000 を搭載した。Zynq

[*13] LHC クロックと同期しないで動作するコントロール。

SoC デバイスを採用した理由として主に以下 3 点が上げられる。

- 4.3.1 節にて登場する Xilinx Virtual Cable という Xilinx 社開発の Zynq を使った既存技術をこの制御系に応用すること
- Zynq のプロセッサーで走る Linux を用いることで、自作アプリによる自由な操作、監視を行うこと
- Zynq の FPGA により、要求される信号処理を自由度をもたせて実装すること

また、この制御回路の設置位置として、図 2.13 の mini-Rack にある HSC VME crate を活用した。

そして、本研究の研究開発の対象である、この制御回路は、**JTAG Assistance Hub (JATHub)**[*14] と名付けられた。第 3 章以降にて、JATHub の詳細と JATHub 試作機の動作試験について述べる。

2.3.2 制御系回路の必要数

TGC 検出器の円盤を 1/12 した扇形を、1/12 セクターと呼ぶ。図 2.13 の黄色扇形で囲まれた区域が 1/12 セクターである。この 1/12 セクター単位で、制御系にある回路の必要数を表 2.2 に示す。TGC 検出器は円周座標を ϕ で表しており、1/12 セクターに 2 つある PS-Pack は $\phi 0, 1, \phi 2, 3$ と分けて呼ぶ。(この制御系を含めた、TGC 検出器エレクトロニクスシステムのスケールは図 A.1 を参照。)

また、ATLAS 実験室内全体の必要数を表 2.3 に示す。

表 2.2 制御系にある回路の必要数 (BW1/12 セクター単位)

回路	設置場所	必要台数	CAT6 本数
PS board	PS-Pack (M1 $\phi 0, 1$)	11 台	22 本
	PS-Pack (M1 $\phi 2, 3$)	11 台	22 本
	PS-Pack (M3 $\phi 0, 1$)	18 台	36 本
	PS-Pack (M3 $\phi 2, 3$)	18 台	36 本
	必要数合計	58 台	116 本
JATHub	HSC crate(mini-Rack)	6 台	12 本

[*14] JTAG 通信と補助線によるフロントエンド電子回路を支える Hub module のため、この名前にした。Zynq は Edge Computing として自動車の自動運転技術にも貢献しているデバイスだが、その自動運転技術は Advanced Driver-Assistance Systems(ADAS) と呼ばれている。JATHub の “Assistance” はこの名称を参考にした。

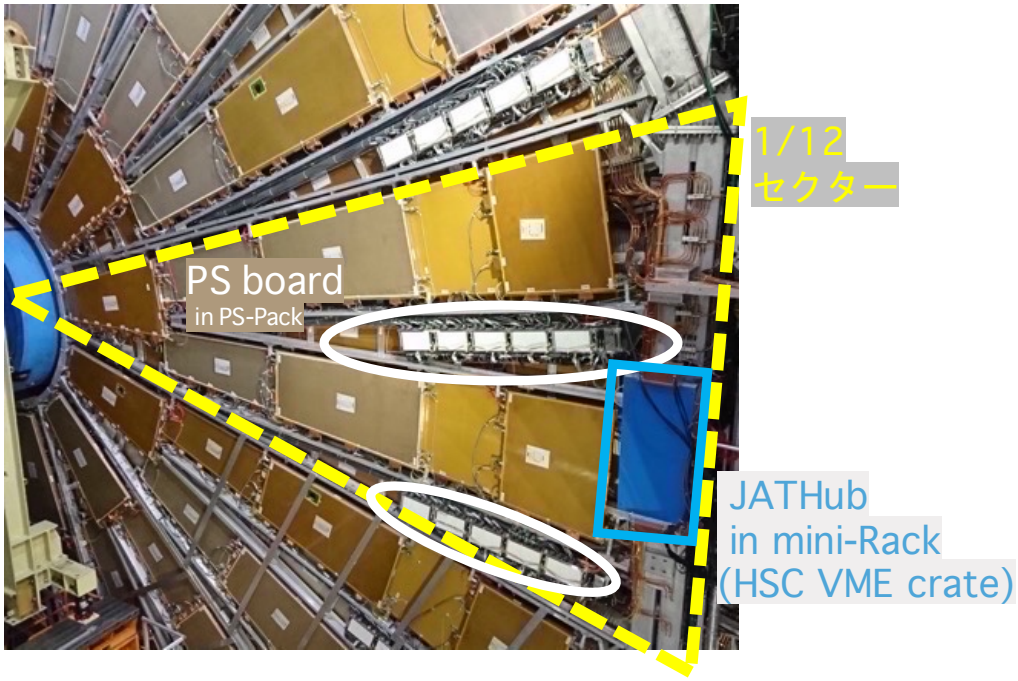


図 2.13　TGC 検出器の写真 (BW1/12 セクター)。PS-Pack の列に PS board が設置される。mini-Rack 内の VME crate に制御回路 (JATHub) が設置される。1/12 セクターにつき、4 つの PS-Rack の列と 1 つの青 Rack が設置されている。

表 2.3　制御系にある回路の必要数 (ATLAS 実験室全体)

回路	設置場所	必要台数
PS board	PS-Pack (Big Wheel)	58 × 12 × 2 台
	PS-Pack (EIL4)	21 × 2 台
	必要数合計	1434 台
JATHub	HSC crate (BW mini-Rack)	6 × 12 × 2 台
	HSC crate (EIL4 用)	2 × 2 台
	必要数合計	148 台

第 3 章

制御回路 JTAG Assistance Hub (JATHub)

最初に 2.2 節で述べられた TGC 検出器フロントエンドエレクトロニクスからの要請を満たす制御システムとその中心で機能するモジュール JTAG Assistance Hub (JATHub) の概念設計を決めた。そして、概念設計に沿ってハードウェアの設計を行い、試作機を製作した。その後、JATHub 試作機で機能を実装し、動作試験を行った。本章では、JATHub の概念設計と、JATHub 試作機の製作について述べていく。なお、本 JATHub 開発研究は Open-it プロジェクト [9] 支援のもとで進められている。

3.1 概念設計

JATHub は外部とネットワークで通信し、複数台の PS board と接続して、各 PS board の FPGA を操作・モニターする。また、JATHub には、プロセッサーと FPGA を組み合わせた Xilinx 社製の Zynq SoC デバイスをメインドライバーとして搭載する。Zynq は、プロセッサー部分である Processing System(PS) と、FPGA 部分である Programmable Logic(PL, FPGA 部分) で構成されている。この CPU + FPGA の構造が JATHub の必要な機能を実装する上で最適であった。図 3.1 は JATHub を使用した制御システムについて簡易的に示している。

本節では、JATHub の概念設計として、‘JATHub に必要な機能の詳細’ と ‘ATLAS 実験室に実装する上で必要なハードウェア設計への要請’ を説明していく。

3.1.1 必要な機能

本節では、2.2 節の要請を満たす制御システムを動かす上で、JATHub に必要な機能を述べる

3.1.1.1 光イーサネット通信

JATHub は Zynq のプロセッサー部分である PS で Linux を起動し、ATLAS 実験室外部と Ethernet 通信を確立し、PS board の制御とモニターを行う。そこで、JATHub は回路室の JATHub 用 Switching Hub と有線で接続して、Switching Hub を介した外部との Ethernet 通信を確立する。このパスを実現するには、JATHub と回路室の Switching Hub の間 約 60 - 100m を有線で接続させる必要がある。また、ATLAS 実験室と回路室を有線で繋ぐ際、回路室を放射線区域外にすべく実験室と回路室の間の穴は最小

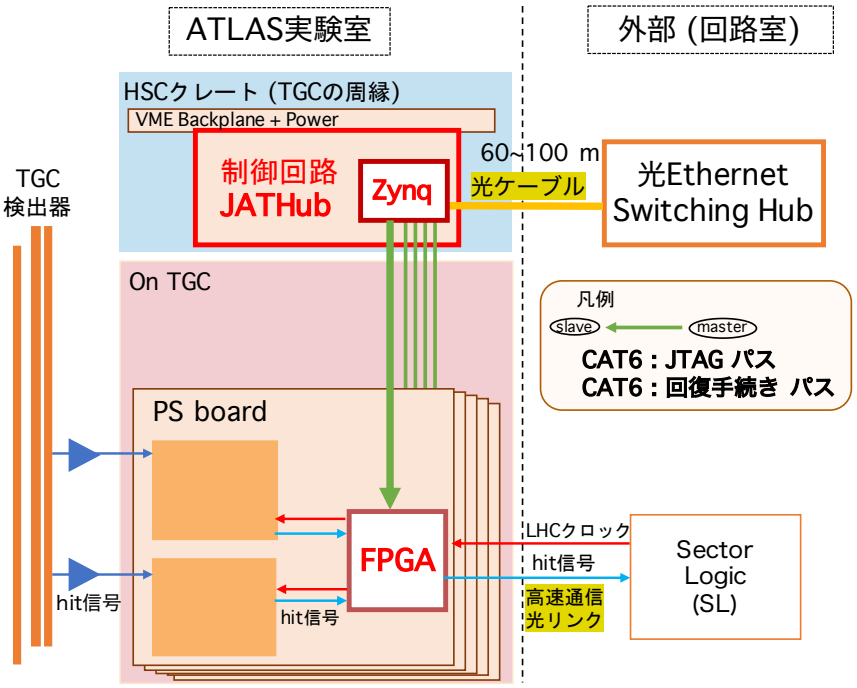


図 3.1 JATHub の設計概要図。Zynq をメインドライバーとして、Linux を走らせ、複数の PS board の FPGA を操作・モニターする。

限にするため、幅を取らないケーブルを使用する必要がある。従って、Ethernet 通信用のケーブルには、一般的によく使用させる Copper(LAN) ケーブルではなく、長距離でも安定して高速通信が行え、且つ幅を取らない光ケーブルを使用する。

そのため、RJ45 jack や Ethernet 用 PHY chip、Copper ケーブルを使用した従来の Ethernet 通信方法は本番環境では使用できないので、JATHub は、光ケーブルで回路室と接続し、光信号によって Ethernet 通信を確立する機能が必要である。

3.1.1.2 JTAG 通信による Slave module の制御

PS board に搭載されている FPGA Kintex-7 は内部のデジタル回路[*1]を何度でもユーザーが書き直すことができる。しかし、一般的に Kintex-7 へ Firmware を書き込むためには、書き込み用のアプリケーションが走っている Host PC を Kintex-7 の近くまで持っていき、JTAG4 線用の短い Copper ケーブルで Host PC と Kintex-7 を接続させる必要がある。

2.2 節でも示した通り、PS board は TGC 検出器上に設置されており、実験外時間であっても実質的にアクセスできない場所に置かれている。よって、JATHub は、PS board と Copper ケーブルで接続し、JTAG 通信により PS board の Kintex-7 内のデジタル回路を書き込む機能が必要である。また、複数の PS board の Kintex-7 を一同に操作するハブ機能も必要である。

さらに、同じ HSC VME crate 内の隣の JATHub の Zynq も外部から configuration を行う必要があるので、相互的に master JATHub から slave JATHub へ JTAG 通信する機能が必要である。

[*1] ‘論理回路’ とも言う

3.1.1.3 PS board で自己修復不可能な SEU 事象が発生した際の対処

2.2 節でも説明した通り、高輝度 LHC-ATLAS 実験では、PS board に FPGA を新設するため、FPGA Kintex-7 の放射線損傷へ対処する必要がある。特に注意すべき放射線損傷は、内部 register のフリップさせる Single Event Upset (SEU) 事象である。通常、PS board 上の Kintex-7 は、内部 register の 1 つで SEU 事象が発生しても、Xilinx が提供する Soft Error Mitigation (SEM) により自動修復する。しかし、たまに隣合う 2 個以上の内部 register が同時に SEU を起こした時など、フリップレジスタを修復できない状態に陥ることがある。この “回復不可能な SEU 事象”が発生した場合、外部から Kintex-7 のリセットを掛ける必要がある。また、PS board は全 Hit 信号を SL へ送るデータパスとしての役割があるため、Kintex-7 にて回復不可能な SEU 事象が発生したら自動的にリセットを行う必要がある。本論文では、この対処を “Recovery 手続き”と称する。

そこで、JATHub は PS board と Copper ケーブルで接続し、PS board を常時監視する。そして、Kintex-7 内の自動回復不可能な SEU 事象を検知し、自動的にリセット信号を送り外部から Kintex-7 の機能を回復させるという Recovery 手続き機能が必要である。

3.1.1.4 隣の JATHub で自己修復不可能な SEU 事象が発生した際の対処

PS board 上の Kintex-7 と同様に、JATHub に搭載される Zynq SoC に関しても、放射線損傷への対策が必要である。FPGA 領域である PL では、同様に SEM 機能が働いており、ほとんどの SEU 事象が自動修復される。しかし、プロセッサー領域である PS では、SEU 事象への対策が存在せず、SEU 事象が起きたらプロセッサーのクラッシュが発生する。JATHub の設置位置において、1 台の Zynq の PS がクラッシュする頻度は 2 時間に 1 回程度になると算出されている [10]。Zynq SoC では PS と PL の動作は電力系統も含めて独立しているため、PS がクラッシュしても、PL は動き続けることができる。また、JATHub はデータパスに関わっていないため、高輝度 LHC-ATLAS 実験中は、PL が機能している限り、直ぐに PS の回復を行う必要はない。そのため、JATHub は PL で回復不可能な SEU 事象が起きた時のみ、外部から Zynq の再起動を行ってもらう必要がある。

そこで、同じ HSC VME crate 内の JATHub と相互的に Recovery 手続きを行えるようにし、隣の JATHub 上の Zynq PL で自動回復不可能な SEU 事象が起きた時のみ、JATHub が隣の JATHub へ Recovery 手続きを行う機能が必要である。尚、PL の configuration は、Zynq の起動シークエンスの中に含まれているため、PL のみを reconfiguration することはできず、Zynq の再起動を行う必要がある。

また、高輝度 LHC-ATLAS 実験中、JATHub は PL のみ使用できるため、Slave module の Recovery 手続き機能は PL の Firmware 内で完結させる。そして実験中 JATHub の機能は Slave modules の Recovery 手続きのみに制限する。

3.1.1.5 LHC クロック位相のモニター

LHC クロックの位相は、ATLAS 実験室内の全ての PS board にて O(100) ps の精度で一致している必要がある。LHC クロックは回路室の SL から高速光通信の 8b/10b プロトコルでコーディングされて PS board へ届けられる。そこで、ケーブルの長さ等の要因で、個々の PS board にて LHC クロックの位相は変位する。

そこで、JATHub は PS board に供給された LHC クロックの位相が ATLAS 実験室内全体で一致していること確かめるために、25ns(40.079MHz) の LHC クロックの位相を O(100) ps の精度でモニターする機能が必要である。また、複数の PS board と JATHub の間は、JATHub の PCB レイアウトも含めて等長配線にし、LHC クロックの伝送距離を全線にて揃える必要がある。

3.1.1.6 冗長性のある BOOT システム

JATHub の Zynq は常に再起動が行える状態でいるのが必須で、有事の際にも予備の BOOT システムによって再起動が行える必要がある。

そこで、JATHub は、複数の Boot ファイルを回路上の flash memory に用意し、ある Boot ファイルが放射線損傷を受けても、Zynq が有効な Boot ファイルを選択できるような、冗長性のある Boot システムが必要である。

3.1.2 ATLAS 実験室内の設備との親和性に関する考察

ATLAS 実験室では、Run3(2021-2024) 運転が終了する翌年の 2025 年から、高輝度 LHC-ATLAS 実験の準備が始まるが、そこで効率の良い実験装置の実装が求められている。新しい機材を実装するにしても、一から設備を作り変えるよりも、現行のシステムの設備を再利用する方が、コストも時間も削減できる。本節では、そういった観点から ATLAS 実験室の現行システムの設備を踏まえて JATHub の設計を考察する。

3.1.2.1 HSC (VME) crate の活用

TGC 検出器の現行システムでは、Hit 信号は PS board でコインシデンスを取り HSC crate 内の module に送られる。HSC 内の module は、読み出しデータの buffering とコインシデンスを取る信号処理を行い、回路室にコインシデンスを取ったデータや trigger された読み出しデータを送る。しかし、高輝度 LHC-ATLAS 実験における TGC 検出器のフロントエンドエレクトロニクスでは、全 Hit 信号を PS board が回路室へ直接送るため、この HSC crate が空く。

そこで、この HSC crate に JATHub を置くことで、フロントエンドエレクトロニクスの制御システムを構築した。HSC crate に JATHub を実装する利点について以下の要点が上げられる。

- LHC 運転休止中 ATLAS 実験室に入った際、HSC crate は人が簡単にアクセスできる位置に設置されているため、簡単に crate 内の module を交換できる。
- HSC crate は、衝突点からも遠い位置にある。そのため、crate 内の module の放射線損傷のリスクを軽減できる。
- HSC crate 内の module と PS board は、最大 15m の Copper ケーブル (Category-6 (CAT6) ケーブル) で安定して接続できる距離にある。

上記の点を踏まえて、JATHub が HSC crate に実装できるようハードウェアを設計した。図 2.8 と図 3.2 のように、この HSC crate は縦 9U、奥行き 16 cm の大きさで、VME master module と 20 slots 分の VME slave module が挿入できる構造になっている。また、VME backplane の J1 には、VME 規格

の信号線がある。VME backplane の J3 には、電源供給線と JTAG 通信線、slave module 個別に操作できる信号線がある。JATHub は HSC crate の J3 から電源供給を行う。

3.1.2.2 ケーブルの再利用

TGC 検出器の現行システムにおいて、最大 15m[*2]の CAT6 ケーブルを使用している。

高輝度 LHC-ATLAS 実験における TGC 検出器制御システムでは、CAT6 ケーブルは再利用して JATHub と PS board の接続に使用する。なお、光ケーブルは新設する。

現行システムの CAT6 ケーブルの本数を踏まえて、現行システムの CAT6 ケーブルを取り外さずに再利用して JATHub と PS board を実装できるようにした。高輝度 LHC-ATLAS 実験において、2.3.2 節の必要数をカバーすること、JATHub, PS board 間は 8 対 16 線の LVDS 通信を行えるのが望ましいことも、JATHub 設計に加味した。その結果、JATHub は最大 11 枚の PS board と接続し、各 PS board に対して 2 本の CAT6 ケーブル (4 対 8 線 x2) を使用する設計にした。加えて、1/12 セクター毎に 1 つの HSC crate が置かれているので、図 3.2 のように、その中に 6 台の JATHub を実装して、1/12 セクター内全ての PS board を操作できるように設計した。現行システムでは 1/12 sector 毎に 122 本の CAT6 ケーブルを使用しており、高輝度 LHC-ATLAS 実験では JATHub, PS board 間にて 116 本の CAT6 ケーブルを使用する予定となっている。

JATHub と PS board の配線図は C 章にて示す。

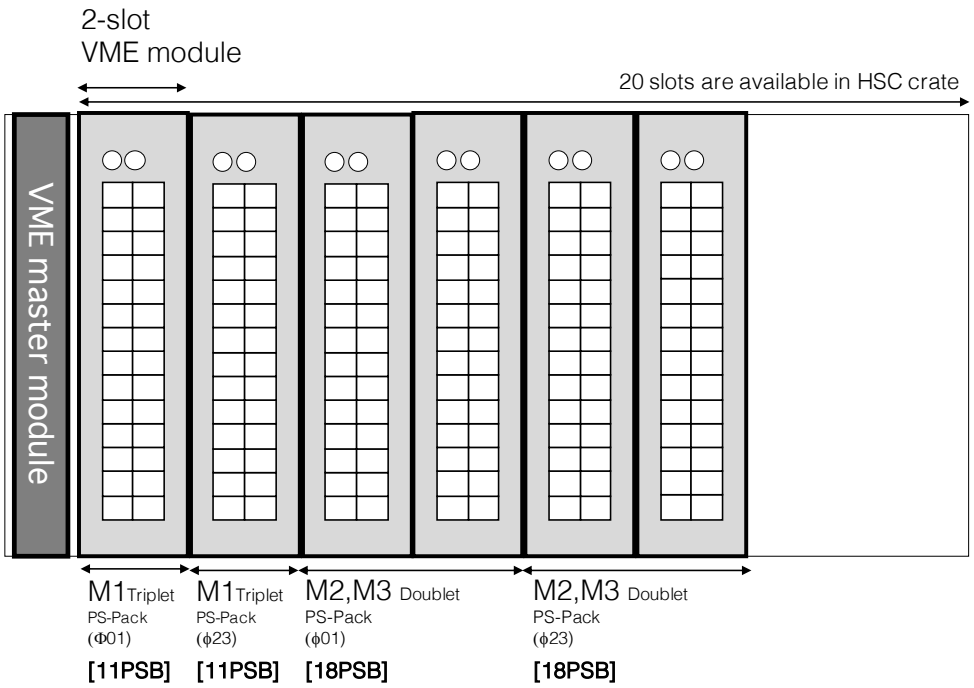


図 3.2 HSC crate 内の JATHub の実装。VME 規格の 20slots 中、各 JATHub は 2slots を使用し、12 slots 分 6 台の JATHub で 1/12 sector の PS board を制御する。

[*2] M1 用 PS board へは 10m、M2, M3 用 PS board へは 15m の CAT6 ケーブルを使用している。

3.1.2.3 EIL4 区域への JATHub 実装

EIL4 区域の TGC 検出器上に PS-Pack が設置され、EIL4 用 PS board はその PS-Pack に収納される。また、30 m 程離れた場所に EIL4 用の HSC crate が設置されている。JATHub は EIL4 用の HSC crate にも挿入し、30 m の CAT6 ケーブルを使って、EIL4 用 PS board に接続する。

3.1.3 ハードウェア設計への要請

制御系の中心的回路である JATHub の必要な機能を思案し、ATLAS 実験室内の設備を考察して、TGC 検出器エレクトロニクスの制御システムの最終設計を図 3.3 のように決めた。その上で、JATHub のハードウェア設計に対して、以下の要請を取り決めた。

- Arm Cortex-A9 プロセッサーと Kintex-7 FPGA を組み合わせて高速光通信に対応する GTX transceiver も載せた Zynq SoC デバイス (Zynq-7000 シリーズ中の Zynq-7045 型) を搭載すること。
- 光ケーブルによる回路室との光通信に対応するため、光信号と電気信号を相互に変換する SFP (Small Form-factor Pluggable) 光 transceiver をインターフェイスとして使用すること。
- SFP から Zynq の GTX transceiver に伸びる差動線 (+, -) は等長配線であること。
- 最大 11 枚の PS board に対して、CAT6 ケーブルで接続し、JTAG 通信と Recovery 手続きと LHC クロックモニターができるように、RJ45 multijack をインターフェイスとして使用すること。更に、各 PS board に対して、2 本の CAT6 ケーブルを使用すること (JTAG 通信に 1 本、Recovery 手続きとモニターに 1 本)。
- LHC クロック位相のモニターのために、O(100) ps の精度で PCB レイアウトの等長が保障されていること。
- 隣の JATHub1 台に対して、CAT6 ケーブルで接続し、JTAG 通信と Recovery 手続きができるように、RJ45 multijack をインターフェイスとして使用すること。こちらも JTAG パス用、Recovery パス用に 2 本の CAT6 ケーブルを使用すること。
- Slave module の reset 操作に関して、active low の reset 信号線を設けて、一定の信号幅の 0 信号を受信することで slave module が reset されるような仕掛けを作ること。
- 各 PS board から送られる LHC クロックの位相をモニターするにあたって、参照比較用の独立した LHC クロックを取り入れるために、LEMO をインターフェイスとして使用すること。
- PS board からの信号線は等長配線にすること。CAT6 ケーブルは等長として、JATHub の PCB レイアウトでも等長であること。
- 冗長性のある BOOT システムを実現するために、1 つの QSPI flash memory と 2 つの SD カードを使用すること。詳細は 4 章の BOOT システム実装で説明する。
- HSC crate に入るように、回路の大きさは縦 9U、奥行き 16cm とすること。また、VME backplane である J1, J3 に接続するコネクターを使用すること。
- VME 規格に沿ったアクセスが行えること。

- RJ45 multijack で横幅を取るので、HSC crate の slave module 用 20 slots のうち、JATHub1 台につき 2 slots 使う横幅にすること。
- 電源供給は、HSC crate の J3 backplane から、もしくは、デバック用に外付け電源ケーブルから行うこと。

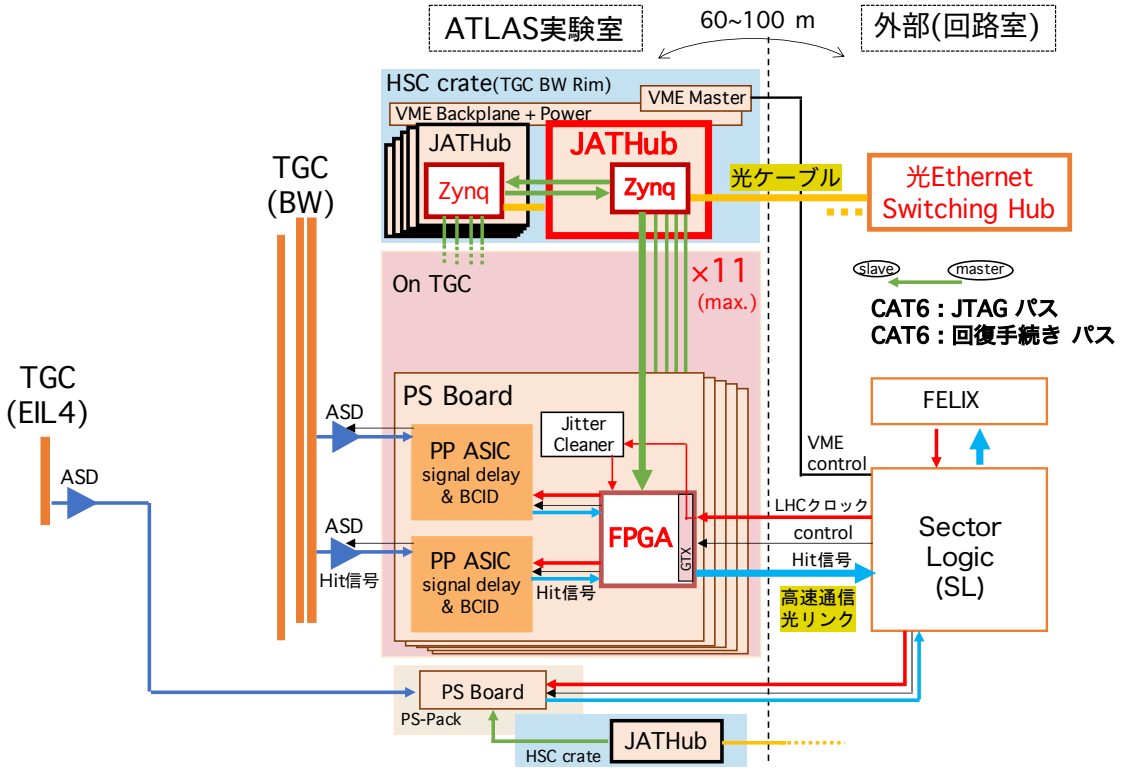


図 3.3　JATHub を制御系の中心として組み込んだ TGC 検出器エレクトロニクスシステム。TGC 検出器読み出し・トリガーエレクトロニクスシステムの詳細絵 (図 2.1) に、JATHub を含む制御系の詳細絵を組み込んだ。

ここで、Zynq-7045 型を採用した理由について述べる。

第 1 に、JATHub が扱う信号線の必要数分、I/O pin が揃っている Zynq を選んだ。最大 11 枚の PS board の操作用信号線で 88 本、VME slave module としての信号線で 49 本、他にも 50 本程など、JATHub において PL に繋げる信号線は多く、pin 数が多い FPGA が必要になってくる。

第 2 に、必要十分な処理能力を持つ Zynq を選んだ。Zynq には Zynq MPSoC と呼ばれる複数のプロセッサーを積んだ種類のデバイスも存在する。しかし、今回 JATHub が Linux を起動させて行うのは、Ethernet 通信機能と、PL から伸びる信号線の操作である。つまり、そこまで高度でないアプリケーションの実行に限られているので、Arm 32bit CPU を搭載した Zynq-7000 シリーズのデバイスを選んだ。また、Zynq SoC に組み込まれている FPGA には高速通信 transceiver が優良なデバイスも存在する。しかし、JATHub は、GTX transceiver を利用する機会が光 Ethernet 通信などに限られる。

結果、適切な Zynq SoC デバイスは Zynq-7045 型 FFG900 パッケージであった。

3.2 ハードウェアの設計

JATHub の概要設計に従って、JATHub 第 1 試作機を製作し、JATHub の動作検証を行う。そのため、JATHub 第 1 試作機の回路図 (D.1 節参照) を作成した。本節では、下記に示すような、ハードウェア設計の技術的特徴を説明する。また、その後評価ボードを使用した JATHub の機能のデモンストレーションについて説明する。

i. JATHub のインターフェイスと周辺機器
ii. CAT6 ケーブルを使用した他回路との LVDS 通信
iii. 回路上の等長配線
iv. Zynq に対応した電源機構

3.2.1 JATHub のインターフェイスと周辺機器

ハードウェア設計ではまず、JATHub に載せるインターフェイスと周辺機器を洗い出した。そして、メインドライバーとして Zynq SoC が JATHub のインターフェイスや周辺機器を操作できるように回路図の設計を行った。本節では、Zynq の PS 部と PL 部に分けてインターフェイスや周辺機器を述べる。Zynq と JATHub 上のインターフェイスや周辺機器の接続関係は図 3.4 に示す。Zynq は、3.1.3 節の要請に従って、'XC7Z045-2FFG900I' 型番を使用した。

■PS 部　Zynq のプロセッサー部である PS 領域から伸びるピン; Multiplexed Input Output(MIO) ピンには、以下の 5 つの周辺機器・インターフェイスを接続させた。MIO に接続できる周辺機器は Zynq の User Guide[13](Table2-4) にて使用できるピンも含めて指定されている。ユーザーはその制限の元 MIO に接続する周辺機器を選択する。

- **SD カード x2**：Zynq の Boot ファイルや実行ファイルを保存する、取り外し交換可能な不揮発性メモリである。Zynq Linux のハードディスクとして使用する。プロセッサー (PS) で走る Zynq Linux により中身の書き換えを行う。
- **QSPI flash memory**：Zynq の Boot 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。PS で走る Zynq Linux などにより Quad Serial Peripheral Interface(QSPI) アクセスで中身の書き換えを行う。(5.1.3 節にて説明する通り、バイパスとして VME からの直接書き換えを行える仕様も追加した。)
- **DDR3**[*3] **memory**：32bit データ幅の Dynamic Random Access Memory(DRAM) である。Zynq Linux のメモリとして使用する。
- **Ethernet RJ45 + PHY(Debug 用)**：LAN(Copper) ケーブルによる Gigabit Ethernet(GbE) 通信を行うためのインターフェイスである。Zynq Linux によるネットワーク通信の実績は当初 Copper ケーブルによるもののみだったため、JATHub 第 1 試作機にはネットワーク通信のデバッ

[*3] Double-Data-Rate3

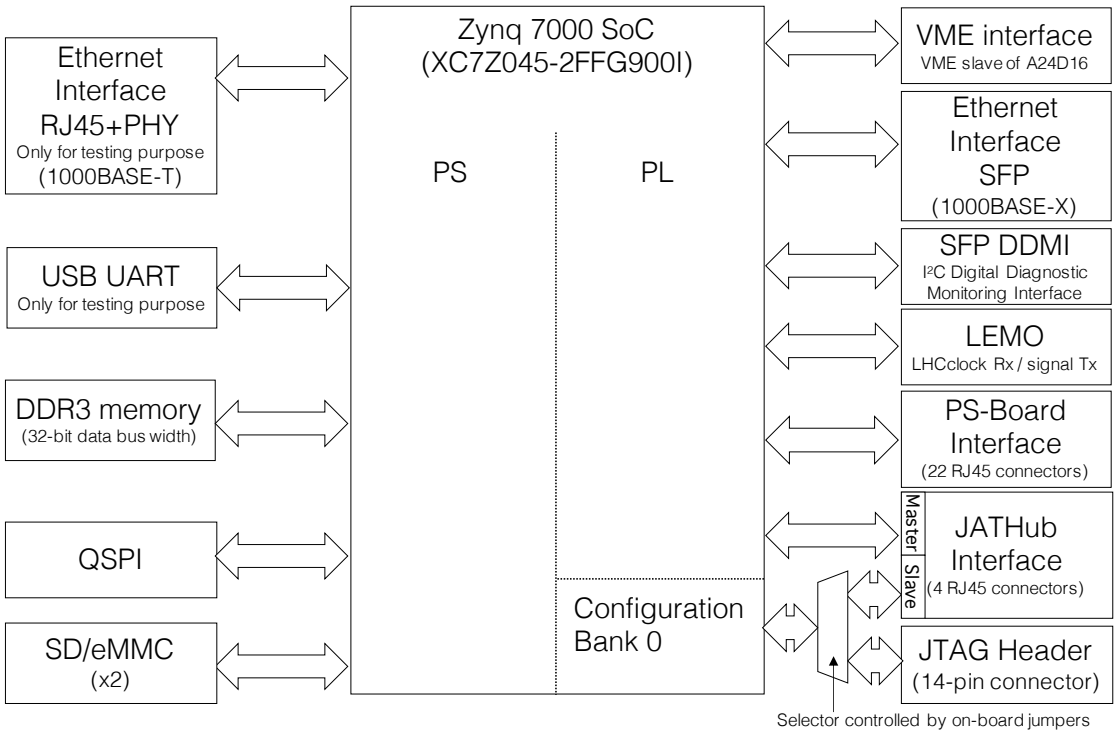


図 3.4 Zynq にドライブされる JATHub の周辺機器。周辺機器をそれぞれ Zynq の適切な Bank のピンに接続した。

ク用として実装した。

- **USB UART(Debug 用)** : Zynq PS のコンソール画面を開くためのインターフェイスである。Universal Asynchronous Receiver/Transmitter(UART) は非同期シリアル通信を行うデバイスであり、接続相手の HostPC にて Zynq PS の Command User Interface(CUI) を開くことができる[*4]。それにより、Zynq Linux が起動する前の PS configuration や Linux 起動の様子が確認できる。JATHub 第 1 試作機には PS 稼働の様子を見るためのデバック用として実装した。

■PL 部 Zynq の FPGA 部である PL 領域から伸びるピンには、主に以下の 7 つのインターフェイスを接続させた。

Zynq の I/O ピンは機能によって ‘Bank’ と呼ばれるグループに分けられており、ピンの特性も Bank によって違う。Firmware でデザインすることでユーザーが自由にピンの設定を行える Bank には、自由度の高い User I/O ピンが付いている。GTX transceiver 用の Bank には、高速通信用の二極信号線と GTX 用水晶発振器のクロック信号線用 I/O ピンが付いている。そして、Zynq を configuration するための Bank(‘Bank 0’) には、JTAG 信号線用のピンや、Zynq の電源サイクルを行うリセット信号線 ‘Power on Reset(POR)’ 用のピンや、PL の reset のみを行う信号線 ‘Program_b(PROGB)’ 用のピンなど、Zynq の configuration に関わる信号線用のピンが付いている。以下で説明するインターフェイスは、それぞれ

[*4] コンソール画面にて、‘screen’ コマンドが非常に使いやすかった。

の信号線の特性に合わせて、適切な PL のピンに接続した。

- **VME** : VME 規格の J1, J3 backplane に接続するためのインターフェイスである。HSC (VME) crate に挿入された際、VME slave module として、Zynq 内の PL の簡単な register 操作を行うために設けた。アドレス 24bit 幅、データ 16bit 幅で設計した。User I/O ピンに接続しており、PL の Firmware にて、VME 規格の信号を処理する。
- **Ethernet (SFP)** : 光 Ethernet 用のインターフェイスである。SFP は光信号と電気信号を相互変換する光 transceiver である。JATHub では、外部からの 1000BASE-X 規格の光信号を SFP で電気信号に変え、PL の GTX transceiver に渡す。機能実装の詳細は 4 章の光 Ethernet 通信にて説明する。
- **SFP DDMI** : SFP は環境情報も内部 register に記録しており、I2C 通信によりその register を読むことができる。これを Digital Diagnostic Monitoring Interface(DDMI) と呼ぶ。JATHub では DDMI により SFP のモニタリングを行う。
- **LEMO jack(tx/rx)** : LEMO 規格 copper ケーブルのインターフェイスである。JATHub では、PS board から送られた LHC クロックをモニターするために、PS board とは独立した LHC クロックを受け取るための LEMO rx を実装した。また、PL で生成した信号を送り出すための LEMO tx も実装した。User I/O ピンに接続した。
- **PS board 用 RJ45 multi-jack** : 対 PS board 用のインターフェイスである。最大 11 台の PS board に 2 つの CAT6 ケーブルを繋げられるように、2×11 の RJ45 ポートを実装した。この信号線は User I/O ピンに接続されており、Firmware によって自由度を保って信号処理が行える。詳細は 3.2.1.1 節。
- **隣 JATHub 用 RJ45 multi-jack** : 隣の JATHub と相互通信を行うためのインターフェイスである。master として 2×1 の RJ45 ポートを、slave として 2x1 の RJ45 ポートを実装した。slave JTAG(‘JTAG rx’) 用 RJ45 ポートの信号線や slave reset 線は configuration bank 0 へ接続し、他の信号線は User I/O ピンに接続した。詳細は 3.2.1.1 節。
- **14pin JTAG rx connector(Debug 用)** : JATHub が JTAG slave として Host PC と接続するためのインターフェイスである。一般的に Xilinx 社製の FPGA, Zynq デバイスでは JTAG rx 通信による configuration を行う場合、14pin connector を使用している。Xilinx 社製の ‘Platform Cable USB II’ を使用して、14 pin connector と HostPC の USB を接続し、HostPC から直接 FPGA, Zynq を configuration する仕組みになっている。そこで、JATHub にも Zynq configuration の Debug 用に 14pin connector を実装した。また、slave JTAG 用のピンは bank 0 に 1 つしかないため、隣 JATHub からの JTAG rx 信号線と干渉しないように、Jumper pin を使ったスイッチを設けた。

3.2.1.1 Front-panel に設けたインターフェイス

図 2.8 の HSC crate に隙間なく全ての module を挿入するため、簡単に VME slave module のインターフェイス機器[*5]を取り扱うことができるのが手前側のみとなる。他にインターフェイス機器を取り扱うことができるスペースは、HSC crate の裏側の J1, J3backplane の間にあるスペースのみであるが、アクセスが難しく、ATLAS 実験室の本番の HSC crate では手が届かない。そのため、HSC crate に実装することに留意しつつ JATHub にインターフェイス機器を設置した。また、JATHub は手前側に Front-panel を用意し、crate 挿入状態でも手前側のどこに何のインターフェイスが用意されているか分かるようにした。Front-panel のデザイン詳細は 3.3.1 節にて述べる。JATHub を HSC crate に挿入した状態でも取り扱う必要のあるインターフェイス機器は、2 つの SD card スロット、2 個の SFP、LEMO tx/rx、RJ45 multi-jack である。これらのインターフェイス機器は Front-panel 側に設けた。特に、RJ45 multi-jack は図 3.5 の通りに 2x6 の部品と 2x8 の部品を使用した。これにより、最大 11 台の PS board と JTAG パス, Recovery パスで接続し、隣の JATHub とも相互的に JTAG パス, Recovery パスで接続できるようにした。図 3.5 は RJ45 multi-jack の部品を示しており、表 3.1 に従って各ポートの信号線を RJ45 multi-jack に繋いだ。

表 3.1 RJ45 multi-jack の使用ポートの割り当て

本 JATHub	接続先の他回路	パスの種類	図 3.5 中の RJ45 のポート番号
Master	11 台 PS board	JTAG パス	CN4 の A,B,C,D,E,F,G,H, CN5 の A,B,C
Master	11 台 PS board	Recovery パス	CN4 の I,J,K,L,M,N,O,P, CN5 の G,H,I
Master	隣の JATHub(S)	JTAG パス	CN5 の D
Master	隣の JATHub(S)	Recovery パス	CN5 の J
Slave	隣の JATHub(M)	JTAG パス[*6]	CN5 の E
Slave	隣の JATHub(M)	Recovery パス	CN5 の K

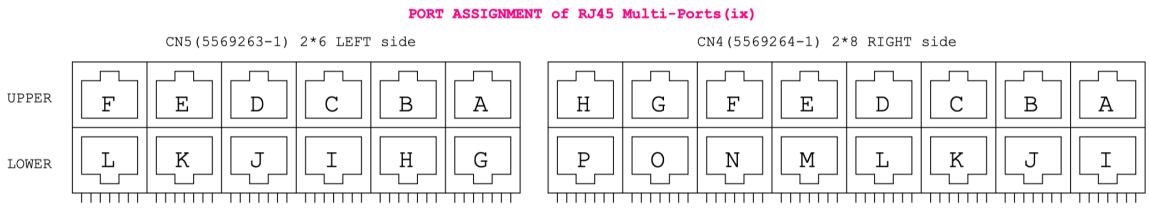


図 3.5 RJ45 multi-jack の概要図。図の左側が JATHub の上向きとなるように設計した。この並びで Frontpanel に設置した。

[*5] VME インターフェイスは裏側にあり、VME backplane に接続する。
[*6] 文中の ‘JTAG rx’

3.2.2 CAT6 ケーブルを使用した他回路との LVDS 通信

CAT6(Category-6) ケーブルは 4 対 8 線の導線で構成されており、4 種類の差動信号線 (+ , -) を通すことができる。JATHub では、LVDS(Low Voltage Differential Signaling) 通信を採用し、他回路と CAT6 ケーブルで信号線を接続する設計にした。

JATHub を master として、slave の各回路に対して 2 本の CAT6 ケーブルを接続するが、その信号線と RJ45 ピン配置は図 3.6、図 3.7、図 3.8、図 3.9 にて示している。ここで、灰色の三角 LVDS 素子が driver 素子を示しており、白色三角 LVDS 素子が receiver 素子を示している。

■対 PS board の場合

- JTAG パスでは 4 つの信号線、TCK (Master → Slave)、TMS (M → S)、TDI (M → S)、TDO (S → M) を配線した。(図 3.6)
- Recovery パスでは、Recovery Request[‘Recov’] (S → M)、Monitor[‘MON’] (S → M)、PROGB reset (M → S)、GTX reset (M → S) を配線した。詳細は 4 章。(図 3.7)

■対隣 JATHub の場合

- JTAG パスでは 4 つの信号線、TCK (Master → Slave)、TMS (M → S)、TDI (M → S)、TDO (S → M) を配線した。(図 3.8)
- Recovery パスでは、Recovery Request[‘Recov’] (S → M)、Monitor[‘MON’] (S → M)、PROGB reset (M → S)、PORB reset (M → S) を配線した。詳細は 4 章。(図 3.9)

3.2.2.1 LVDS 通信

LVDS 通信は低電圧差動信号による伝送技術を指している。図 3.10 のような、(a) driver と (b) receiver の素子の間で、1.4 V - 1.0 V の範囲の低電圧差動信号を作り出し、一方通行の信号伝送を行っている。‘1’ 信号 (3.3 V) を送るとき、driver 素子は (+) 差動線にて 1.4V、(-) 差動線にて 1.0 V を receiver 素子に流す。‘0’ 信号 (0 V) を送るとき、driver 素子は (+) 差動線にて 1.0 V、(-) 差動線にて 1.4 V を receiver 素子に流す。この LVDS 通信による利点は 4 点ある。

i. **安定したデータ伝送**：LVDS 信号は receiver 素子にて差動線 (+)(-) の差分を評価して、信号を出力する。そのため、外的電気ノイズが両差動線に発生しても、差分は不変なので、出力される信号が一定に保たれる。また、差動線の電流方向は逆であるので、並走する差動線から発生する磁場は安定する。そのため、内的電気ノイズは発生しにくい。従って、伝送の安定性が保たれる。
ii. **長距離データ伝送**：LVDS 信号は差動線 (+)(-) に電位差が生じていればよいので、長距離伝送中の電圧降下によって信号振幅が減衰しても正しく信号を伝送できる。
iii. **低消費電力**：LVDS の素子は定電流回路 (3.5 mA) を内蔵しているため、終端抵抗 (100 Ω) により常に一定の低電力消費を行っている。スイッチ動作によって電力を多く消費する CMOS(1 線) 信号と比較すると、消費電力は抑えられている。

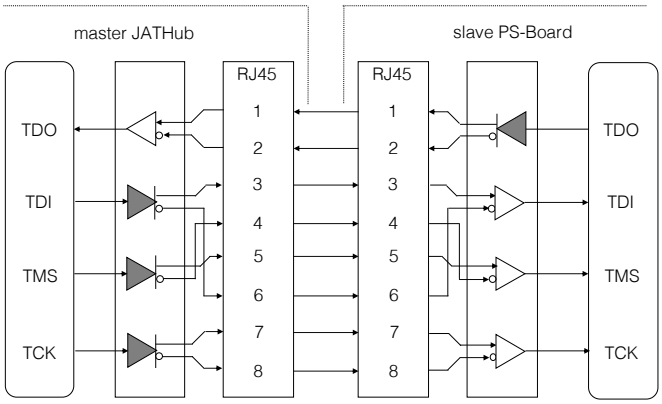


図 3.6 LVDS 接続 JATHub/PSB JTAG パス

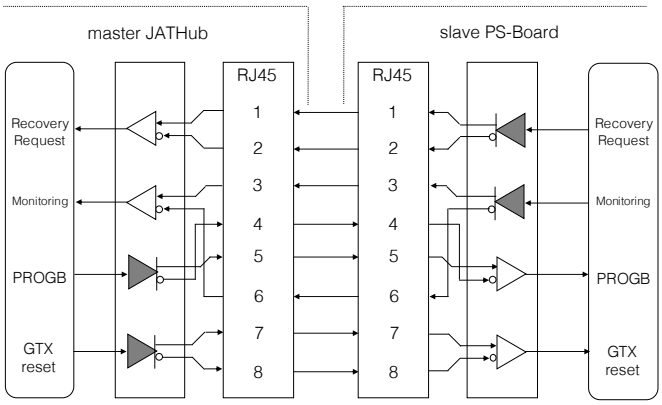


図 3.7 LVDS 接続 JATHub/PSB Recovery パス

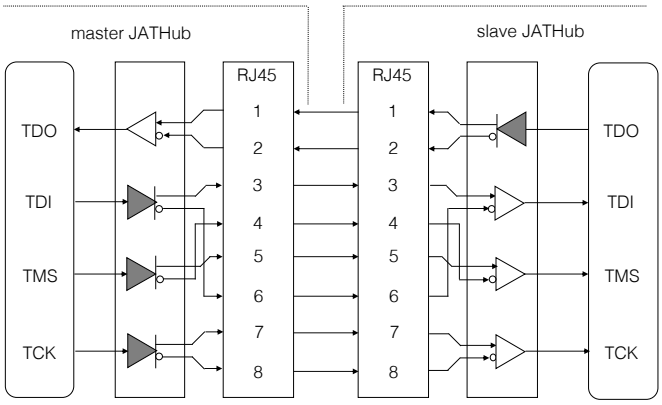


図 3.8 LVDS 接続 JATHub/JATHub JTAG パス

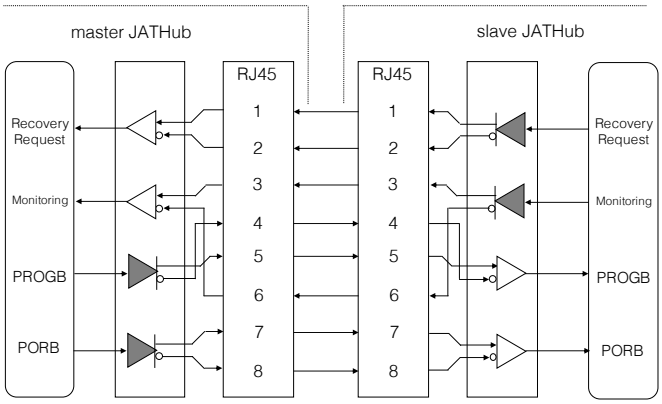


図 3.9 LVDS 接続 JATHub/JATHub Recovery パス

iv. **大量高速データ伝送**：LVDS の差動信号の振幅は 0.4 V 程と狭いため、3.3 V 振幅などの信号と比べると高周波数の伝送が可能となる。

利点 i.、ii. は、ATLAS 実験室内における JATHub と PS board 間 (最大 15 m) の安定したデータ伝送を実現する上で都合がよい。そのため、CAT6 ケーブルの差動線を LVDS 通信に使用した。尚、利点 iv. に関して、JATHub は LHC クロックに同期しないスローコントロールで他回路を制御するため、今回この利点は重要ではない。

3.2.2.2 LVDS 素子による Fail Safe 機能

信号の伝送を行う際、意図しない状況で信号が意図しない動きをすることがある。JATHub は特に他回路の reset を取り扱うため、誤って active low の reset 線に 0 信号を送ることは避けたい。そこで、意図しない状況で安全装置が作動し slave module が 0 信号を受信しないようにするという Fail Safe 機能の実装がハードウェア設計のレベルで重要になってくる。LVDS 素子は Fail Safe 機能を実現できる組み合わせを選定した。

図 3.10 の (a) の driver 素子は、'EN' ピン (TXENABLE 信号線) に '1' 信号 (3.3V) が入力されると、Zynq からの入力信号 (0 or 1) は対応する LVDS 差動信号 ('0' or '1') に変わり RJ45 ポートへ出力される。しかし、EN ピンに '0' 信号 (0V) が入力されると、Zynq からの信号関係無しに、High impedance の LVDS 差動信号が出力される仕様になっている [12]。High impedance 信号とは、信号線が切断された状態と同様の状態であることを表している。

図 3.10 の (b) の receiver 素子は、'0' or '1' の LVDS 差動信号を入力すると、対応する '0' or '1' 信号を出力する。しかし、LVDS 差動信号線にて High impedance 信号が入力されると、自動的に出力線が pull up されて、'1' 信号が出力される仕様になっている [11]。

これにより、driver 素子と receiver 素子が CAT6 ケーブルで接続していない場合、receiver 素子には High impedance 信号が入力されている状態なので、Fail Safe 機能によって常に '1' が出力される。また、driver 素子の EN ピンは Zynq の PL から伸びている TXENABLE 信号線によって操作できるよう設計した。そのため、CAT6 ケーブルで素子間が接続されていても、回路の電源が入っていなかったり、Zynq が configuration されていない時 TXENABLE 信号線は '0' 信号になっているので、driver 素子は High impedance 信号を出力し、receiver 素子は同様に Fail Safe 機能によって '1' 信号が出力される。

このように、CAT6 ケーブルで 2 つの回路が接続されていても、一方が積極的に '0' 信号を送らない限り、もう一方に '0' 信号が届かないよう、Fail Safe 機能を実装した。また、同時に、どの信号線もアイドル状態は '1' 信号になる設計となった。また、回路間でこの LVDS 通信の Fail Safe 機能が働くように、JATHub はもちろんの事、LVDS 通信相手である PS board の LVDS 素子も図 3.10 の素子で統一した。

3.2.2.3 CAT6 ケーブルの種類

CAT6 ケーブルには、ストレートケーブルとクロスケーブルが存在する。図 3.11 のように、ストレートケーブルは 8 つの線それぞれが同じ番号のピンに繋がるが、クロスケーブルは違う番号のピンにスワップして繋がる。Ethernet 通信では master-slave でピンの配置を変え、その間でストレートケーブルが使えるようになっている。クロスケーブルは、昔の Ethernet 通信の master module 同士を繋ぐ際に同種のピ

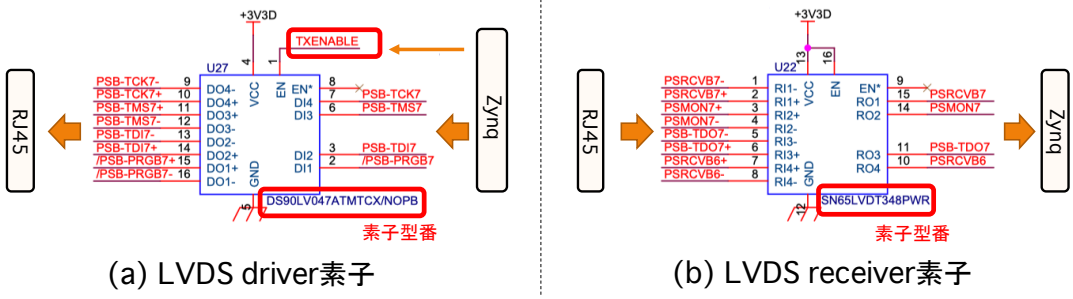


図 3.10　LVDS 素子。(a) が driver の素子で、(b) が receiver の素子。(a) の driver 素子の EN ピンは Zynq PL から操作できるようになっている。

ン配置でも通信できるように使用していた物であった。現在の Ethernet 通信では master module 同士でも、module 側で自動でピンのスワップを行っているので、ストレートケーブルも使用できる。そのため、最近ではクロスケーブルは市場に出回っていないが、運悪く拾ってしまうこともある。

しかし、JATHub で採用している LVDS 通信では、ストレートケーブルを使用することを想定して設計した。従って、使用時は CAT6 ケーブルの種類を慎重に選ばなければならない[*7]。見極め方としては、CAT6 ケーブルのプラグの透明な部分にて信号線に色が付いているのが確認できるので、両端で配色順が一致していることを確認する方法が簡単である。

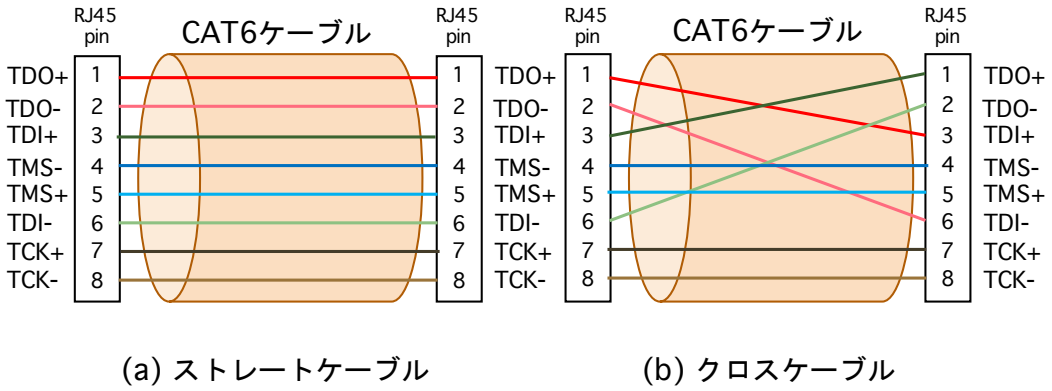


図 3.11　CAT6 ケーブルの種類。(a) ストレートケーブル or (b) クロスケーブル。JATHub の LVDS 通信ではストレートケーブルを使用しなければならない。ここでは、例として JTAG 通信における RJ45 のピン配置を示している。

3.2.3　等長配線

各回路との Recovery パスには、Recovery Request(Recov) 信号線と Monitor(MON) 信号線がある。この 2 本の信号線において等長配線が要請されている。そのため、図 3.12 のように、PCB にて 300 ps

[*7] Copper ケーブルの Ethernet(Debug) はクロスケーブルでも繋がる

(60 mm) 精度で等長配線にするように、Zynq PL の同じ Bank に信号線を固めて、それらの信号線がお互いに近い Zynq ピンに繋がるように設計した。

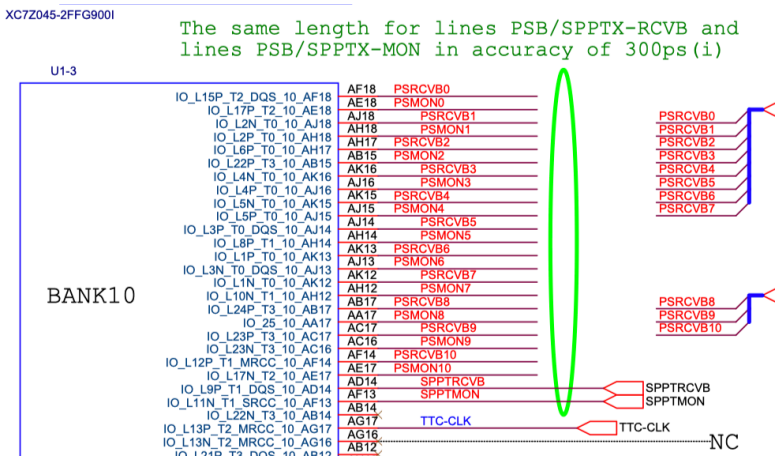


図 3.12 等長配線を指示した信号線。緑色の信号線。他回路から同時に送られる信号は同時に受信されなければならない。

また、ハードウェア設計の要請に従って、SFP と Zynq 間の差動線は PCB レイアウトにて等長配線になるように設計した。詳細は 3.3.2 節にて述べる。

3.2.4 電源周り

Zynq にはいくつもの Bank が存在するが、それぞれ個別の電源がある。これら Bank への電源供給には順序が存在する [13]。PS と PL は独立した電源系統となっている。以下、電源供給の順序である。それぞれの Bank に必要な電圧は図 3.13 の 'FPGA POWER' に示してある。

i. PS PINT | PL INT、PL BRAM、(PL Vcco)
ii. PS PAUX、PS PLL、PS MIO、PS DDR | PL AUX、PL Vcco、MGT(GTX bank) AUX
iii. MGTAVCC
iv. MGTAVTT

ここで、放射線耐性が確認されている Linear Regulator という電圧制御素子を使用して、電源供給を調節する。Linear Regulator は降圧のみ可能な素子で、電源と負荷の間に直列に接続させ、定電圧直流電源を出力する。また、今回選定した素子は Enable ピンに '1' が入力されると降電圧を出力し始め、出力が定まったら Open-drain の PG ピンから '1' を出力する。そこで、JATHub では Linear Regulator の PG ピンと EN ピンを繋げて、図 3.13 の 'POWER OF SEQUENCE' のように、3.3 VD → 1.0 VD → 1.5 VD, 1.8 VD, 1.8 VA → 1.2 VD, 1.0 VA → 1.2 VA の順で電源が上がるように設計した。ここで、3.3 VD のみ外部から JATHub へ供給し、それ以外の電圧電源はダイオードで電圧を可能な限り下げて Linear Regulator での熱の発生を抑える設計にした。

3.3 VD は、VME J3、もしくは、外付け電源ケーブルから電源供給できるように設計した。合流地点にてフューズを設けて、ホールド電流値 12A、トリップ電流値 24A に設計した。ここで、トリップ電流値と

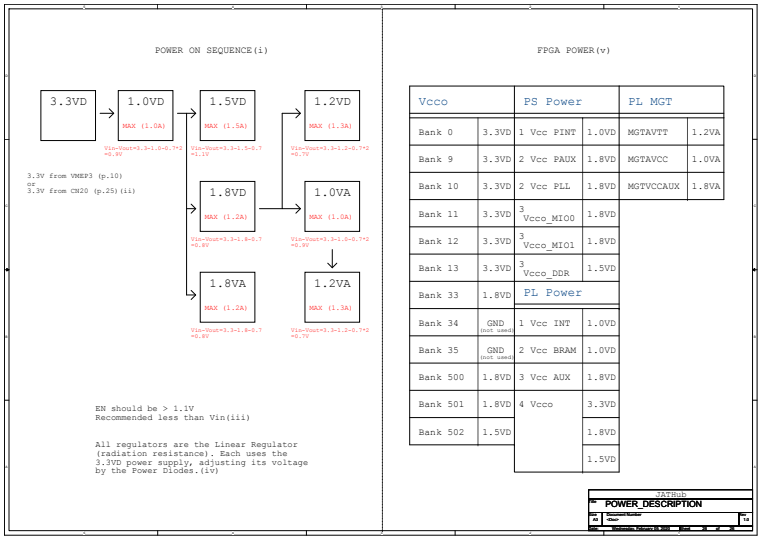


図 3.13 Zynq 電源の供給順序。Linear Regulator を使用している。

は、それ以上の電流が流れ込む場合、フューズが切れて電源供給を切断する電流値を指す。ホールド電流値は、フューズが切れることなく、安定して電源供給できる上限の電流値を指す。(5.1.6 節で議論する通り、第 1 試作機の動作試験の結果により Fuse 容量の最適化を行い、最終的にホールド電流 6 A、トリップ電流 12 A とした。)

3.2.5 評価ボードによる機能のデモンストレーション

回路図作成と同時並行で、Xilinx 社製の ZC706 評価ボードを使用して、JATHub に実装予定の機能をデモンストレーションした。Zynq の開発に慣れて JATHub 第 1 試作機の動作試験にスムーズに入れるように知見を蓄えることを目的とした。デモンストレーションにあたり、Zynq の基本的な使い方に関して、Zynq に関わる開発作業ページ [14] や、同研究室の過去の作業マニュアル [15] を参照した。

ZC706 評価ボードには、JATHub 第 1 試作機と同様の型番の Zynq-7000 SoC が搭載されており、更に、SFP や JTAG 用 4 本テストピンが備わっていた。そのため、JATHub の光 Ethernet 通信機能と、JATHub の Zynq Linux が JTAG 通信をして slave module を操作する機能 (図 3.14) をデモンストレーションした。ここでは、他回路との接続は CAT6 ケーブルではなく、リボンケーブルを使用した。

結果、無事に機能が動作することを ZC706 評価ボードにて確認できたので、評価ボードの回路図も参考にしながら JATHub のハードウェア設計を進めた。

また、Zynq の組み込みデザイン (Zynq PS の組み込み Linux と Zynq PL の Firmware のデザイン) の開発手続きも、このデモンストレーションの研究で完成させた。

3.3 JATHub 第 1 試作機の製作

回路図を作成し、JATHub 第 1 試作機のハードウェアの設計を決定した後に、JATHub 第 1 試作機の回路製作を行った。回路の部品配置の大まかなデザインと、フロントパネルのデザインを決めた

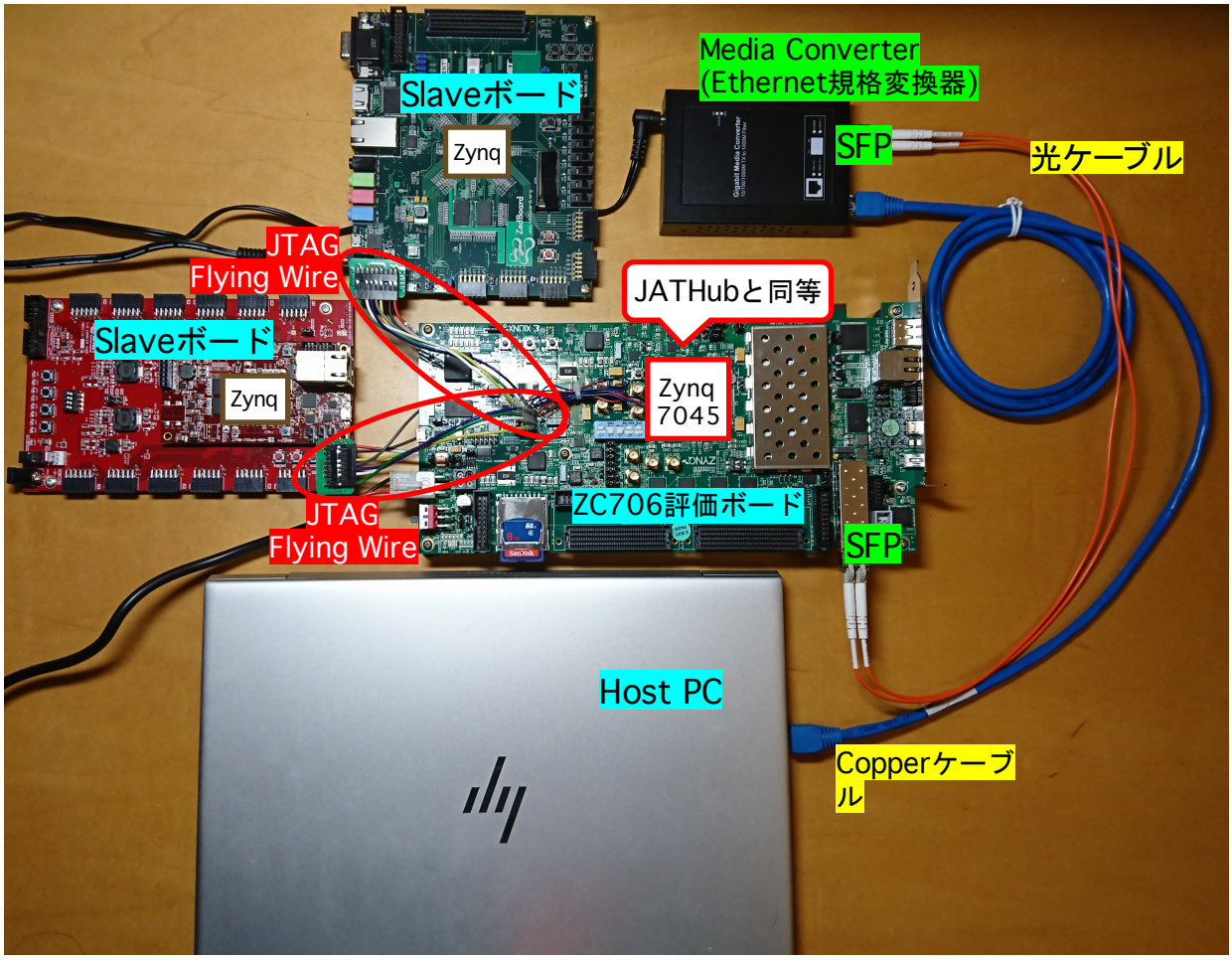


図 3.14　ZC706 評価ボードによるデモンストレーション。HostPC から Media Converter を経由して光 Ethernet 通信で ZC706 の Zynq にアクセス、そして Slave module の Zynq を JTAG 通信で操作した。

後に、第 1 試作機のプリント基板 (PCB) 製作と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。

3.3.1　部品配置とフロントパネルデザイン

3.2.1.1 節の Front-panel 側のインターフェイスでも述べた通り、図 3.15 の (a) のように Front-panel を設計した。JATHub の上から SD card 0、SD card 1、SFP 0、SFP 1、LEMO tx/rx、RJ45 multi-jack を設置した。HSC crate に挿入できるよう、Front-panel の縦は crate 9U の大きさに、幅は 2 slots 分の大きさに設計した。

また、Front-panel の設計と並行して、JATHub 第 1 試作機上の主要な部品の配置デザインも行った。図 3.15 の (b) がその部品配置のデザインである。HSC crate に挿入できるよう、奥行きは 16cm で描いた。HSC crate への挿入を考慮した結果、縁の赤斜線領域がインターフェイス機器を置けない位置となった。左側が Front-panel 側で、Front-panel のデザインに合わせてインターフェイス機器を置いた。右側

が VME backplane 側で、3.2.1.1 節で言及した通り、VME connector P1, P3 と、Debug 用の Ethernet RJ45 と USB UART を置いた。VME backplane を支えるためのバーがインターフェイス機器と干渉してしまうため、backplane 側にも赤斜線領域がある。RJ45 と UART の機器はこの領域を避けて配置した。

Zynq I/O はその機能により Bank と呼ばれるグループに分かれており、(c) は JATHub に搭載する Zynq7045 のそれぞれの Bank から伸びるピンの位置を示している [16]。Zynq の右上にある Bank500, 501, 502 が PS 領域から伸びたピン。左下の Bank109, 110, 111, 112 が GTX tranceiver から伸びたピン。Bank33, 34, 35 が PL 領域のクロックを取り扱う Bank から伸びたピン。そして、それ以外の Bank 番号が PL 領域の様々な信号を取り扱う Bank から伸びたピンとなっている。

部品の配置デザインでは図 3.15 の (c) にある Zynq-7000 SoC の設置向きに注意した。この Zynq7045 の PS 領域と接続するインターフェイス機器は Zynq よりも上側に設置した。GTX transceiver に繋がる SFP や、Bank109 に繋がる LEMO は、Zynq の左横に近い位置に設置した。そして、RJ45 multi-jack の信号線は Bank9-12 に接続するので、RJ45 multi-jack は Front-panel の下側に設置した。

図 3.15 の (b) では、reset 線を on-board の押しボタンでも操作できるようにするため、スイッチを導入した。PL の reset を行う PROGB reset (SW4) や、Zynq の再起動を行う PORB reset (SW7) もスイッチで行えるように設計した。また、JATHub の状態確認を目的として、LED も設置した。具体的には、電源供給されると点灯する LED (D17) や、Zynq の PL が configuration されたことを示す LED (D1)[*8]、PL の User I/O から伸びる Firmware で制御可能な LED (D2-5) を設置した。

また、VME P3 近くの CN15-20 から外付け電源ケーブルが伸びるように設計した。EIL4 区域では、こちらのコネクタを用いて電源供給する。

3.3.2 プリント基板 (PCB) レイアウト概要

回路図を作成し、部品の配置デザインや Front-panel のデザインを決定した後は、外部業者に委託して PCB の製作と部品実装を実施した。委託先業者と協議を重ねていき PCB レイアウトは図 3.17 のようにした。PCB レイアウトは等長配線も考慮して実施した。特に、SFP 光 transceiver と Zynq GTX transceiver 間の信号差動線 (P: +, N: -) は、等長配線にするためのスペースを取ることが難しかった。そのため、SFP0 の TX 線の差動線 (P, N) は、図 3.16 のようにスワップさせた。この差動線の極性の不一致は Zynq PL の Firmware にて修正できるので、Zynq は SFP0 を問題なく使用できる。

また、PCB は 14 層構造にした。階層構成は表 E.1 に示した。インピーダンス値は、単線の信号線に関しては特性インピーダンス (Zo) : (40 Ω or 50 Ω) となり、極性がある差動信号線に関しては差動インピーダンス (Zdiff) : (80 Ω or 100 Ω) となるように設計した。

3.3.3 試作機納品

2020 年 4 月 17 日に、JATHub 第 1 試作機 (図 3.18) が 2 台、KEK へ納品された。納品直後は部品が正しく実装されているか 1 つずつ確認した後に、外付け電源ケーブルから 3.3V を電源供給して正常に動くか検査した。この時、消費電流値は 0.8A であった。また、部品から煙が立つこともなく、正常に電

[*8] Zynq の PS と PL は独立に機能する。この LED は PL の configuration のみを示している。

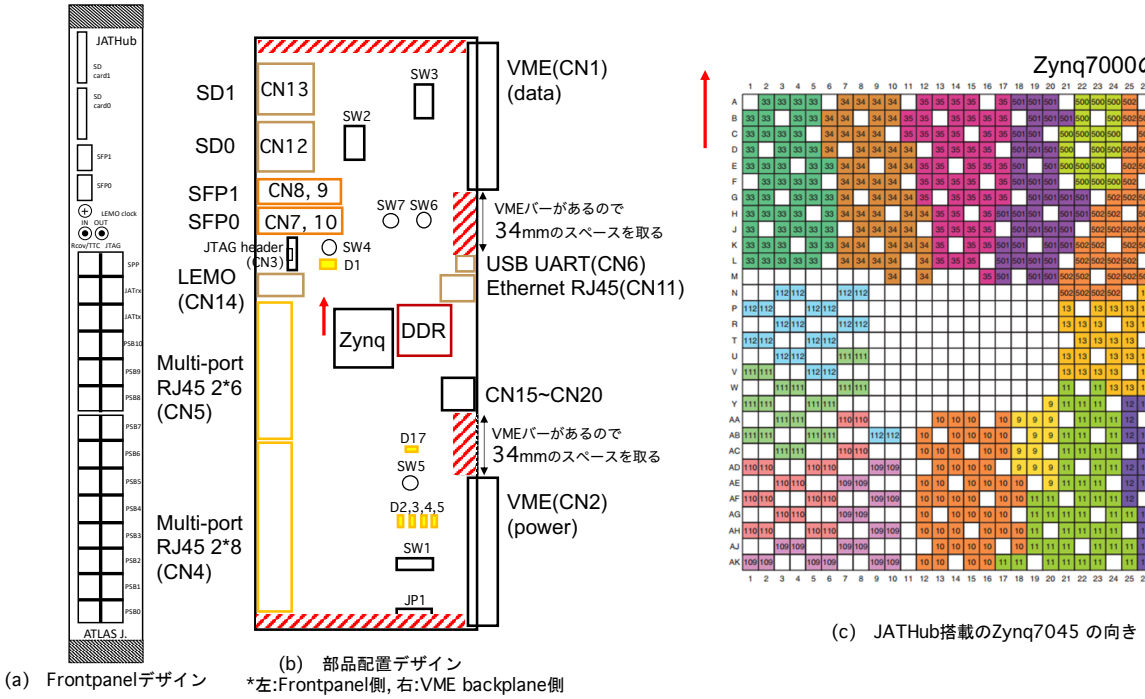


図 3.15 第 1 試作機での部品配置とフロントパネルデザイン。

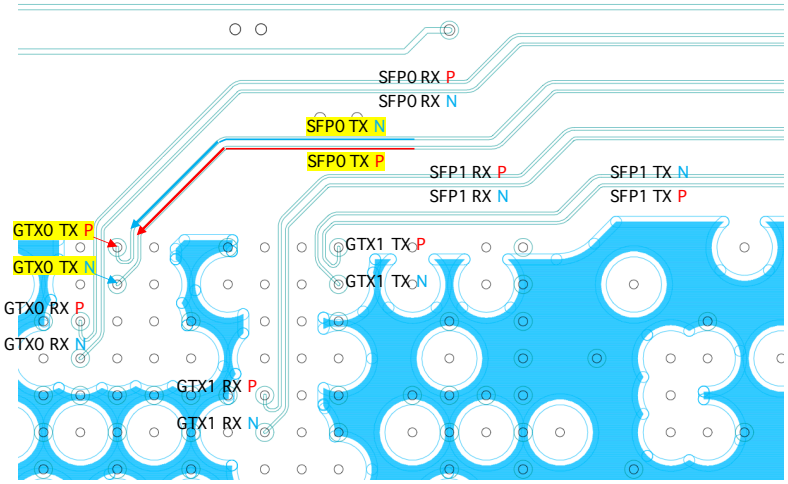


図 3.16 SFP0 の TX 差動線の極性のスワップ。Zynq PL の Firmware にて “TXPORLARITY” を High と設定すると、Firmware がスワップ差動線の極性を更にスワップして扱う。それにより、Zynq は問題なくこの信号線を操作できる。

源供給できたことを確認した。更に、全てのレギュレーターから出力された電圧が正しいこと、Zynq の Programming 等基礎動作が正常に行えたことも確認した。

手前側が Front-panel で、奥側が VME backplane 用 connector である。中心に Zynq-7000 SoC が搭載されており、Front-panel の SFP、もしくは奥側の RJ45 から Ethernet で外部と通信できるようになっている。Zynq の PL に Firmware を焼き、Frontpanel の RJ45 multi-jack の信号線を Hardware 的に処

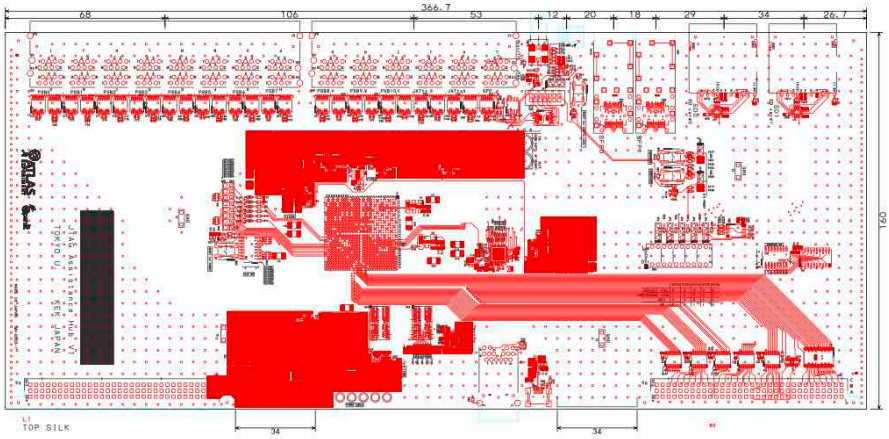


図 3.17　第 1 試作機の PCB レイアウト。業者に委託して製作してもらった。

理できるようになる。また、Zynq の PS で起動する Zynq Linux 上で適切なドライバーやアプリを用意すると、それらの信号線を操作することができるようになる。また、Zynq の起動には、2 つの SD card と 1 つの QSPI flash memory を使用する。

次章では、この JATHub 第 1 試作機の Zynq に機能を実装し、動作試験を行って、設計通りに稼働することを確認した話を述べる。

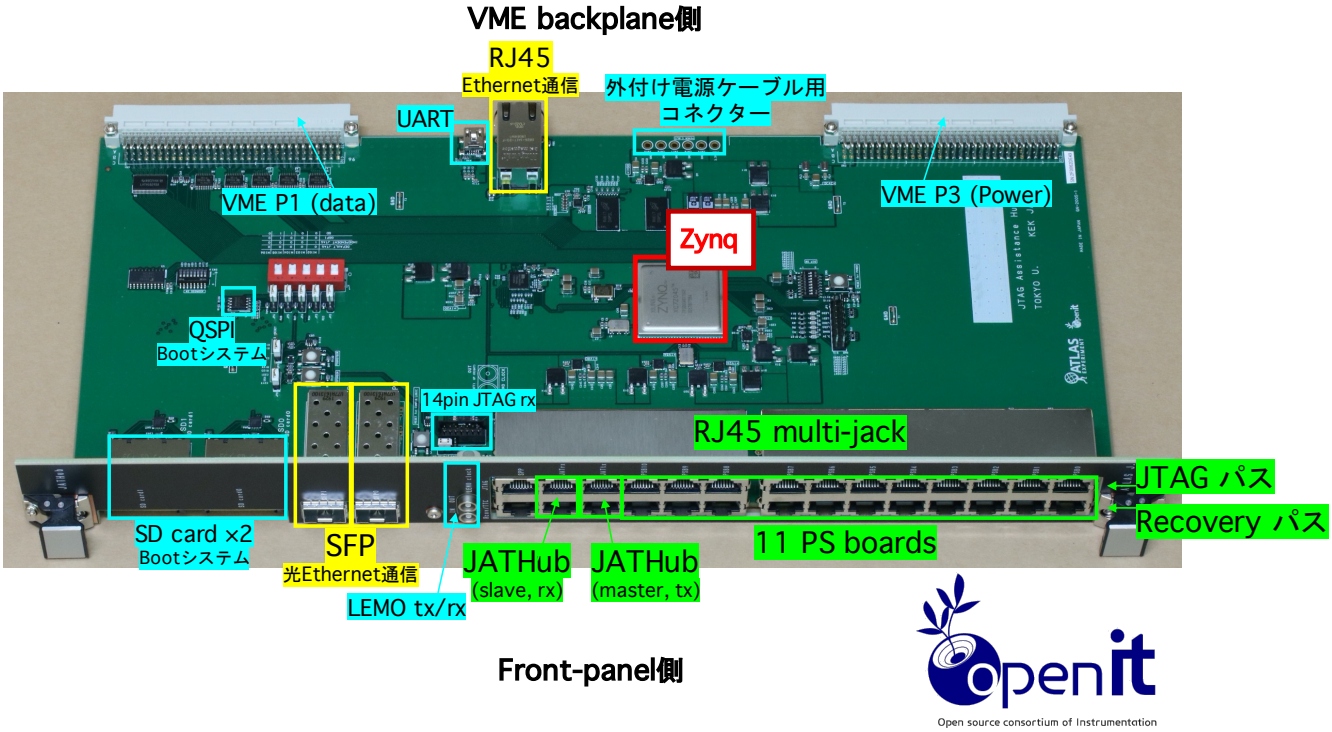


図 3.18　JATHub 第 1 試作機の写真。2020/4/17 に KEK へ 2 台納品された。

第 4 章

JATHub 第 1 試作機の機能実装と動作試験

JATHub のメインドライバーである Zynq SoC に機能を実装し、製作した JATHub 第 1 試作機の動作試験を行った。

Zynq への機能実装において重要となるのは、FPGA 部分である PL のデジタル回路を構築する Firmware のデザインと、プロセッサー部分である PS で走る Zynq Linux や自作アプリケーションなどの組み込み Linux (Software) のデザインである。本研究では、これらの Zynq 組み込みデザインを開発し、JATHub の機能を実装した。実装した機能は 3.1.1 節で述べた “JATHub の必要な機能” に加え、‘VME 操作’ も含めた全 6 項目であり、これらの機能を使ってフロントエンド回路の制御やモニターを行う。本番環境においてもこの 6 項目の機能を持って、JATHub を稼働させる予定である。この章では、開発環境のセットアップについて述べた後、各機能の実装における Zynq デザインの開発詳細と、試作機による動作試験の詳細を説明する。最終的な Zynq デザインの Diagram は F 章に載せている。また、開発した Zynq デザインは Open-it の JATHub プロジェクトページ [9] にて公開予定である。

4.1 開発環境

4.1.1 Zynq 組み込みデザインの開発

Zynq 組み込みデザインの開発に使用したツールを表 4.1 にまとめた。

Zynq 組み込みデザインの開発には Windows 10 Pro の開発 PC を使用した。Zynq の PL 部にプログラムする Firmware の開発や PS 部のハードウェア的設定 (MIO ピンの設定や、プロセッサーの駆動クロック、メモリの設定など) は、Xilinx 社が提供するアプリケーション “Vivado IDE 2018.2” を Windows 10 上で使用し行った。Vivado IDE では、Xilinx 社から提供された特定の機能を実装するための Package(IP(Intellectual Package)) を導入したり、Verilog などの HDL(Hardware Design Language) を使用して自作の信号処理を書くことで、Firmware をデザインする。また、Zynq の User I/O ピンがどの信号線に使用するかなどの制約を行う。最後に Firmware をコンパイルし、バイナリーファイル “Bitstream” を生成する。そして、Bitstream を含む HDF(Hardware Design File) を出力する。

Zynq の PS 部で走る Zynq Linux の開発は、Xilinx 社が提供するクロスコンパイラー “petalinux tool 2018.3” を Ubuntu16.04 上で使用し行った。ここで、Ubuntu16.04 は Windows 10 上で走る仮想マシン

(VMware) で起動している。また、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を指す。Petalinux では、Vivado で出力した HDF を読み込ませて、ハードウェアの設定に沿ったデバイスドライバー、デバイスツリー、Root File System の設定を行うことで、Zynq Linux という Zynq 上で走る OS をクロスコンパイルすることができる。また、自作アプリケーションを Zynq Linux に組み込むこともできる。そして、クロスコンパイルした Zynq Linux 用の Boot ファイルを、Zynq の flash memory(SD card and QSPI flash memory) にプログラムし電源を入れることで、Zynq を configuration して Zynq Linux を起動することができる。Boot ファイルには Bitstream も含まれているので、Firmware の PL へのプログラムは一連の Zynq 起動 sequence の中で行われる。Boot ファイルなどの詳細な説明は 4.6 節で行う。尚、Zynq Linux 上で走る自作アプリケーションは “Vivado SDK 2018.2” 上でデバックやコンパイルを行い開発した。

HostPC には Windows 10 Pro を使用し、Xilinx 社提供のソフトウェア “Vivado Hardware Manager(Vivado HM)” と Xilinx Software Commandline Tool(XSCT) を走らせた。GUI で操作する Vivado HM では、ターゲット回路の FPGA、Zynq の configuration やプログラム、‘デバック’ を行い、QSPI のプログラムも行った。また、CUI で操作する XSCT では、FPGA, Zynq, QSPI の configuration とプログラムを行った。

表 4.1 Zynq 組み込みデザイン開発環境

OS	開発アプリ	出力ファイル	Zynq 組み込みデザイン開発詳細
Windows 10	Vivado IDE 2018.3	HDF(Bitstream など)	Zynq PL Firmware 開発や Zynq PS の設定
Windows 10	Vivado SDK 2018.3	自作アプリの実行ファイル	Zynq Linux 上の自作アプリ開発
Ubuntu16.04[*1]	petalinux 2018.3	Zynq Boot ファイル	Zynq Linux のクロスコンパイル
Windows 10	Vivado HM(GUI)		FPGA, Zynq の configuration や ‘debug’, QSPI flash memory の program
Windows 10	XSCT(CUI)		FPGA, Zynq, QSPI の configuration, program

4.1.2 テストベンチ

動作試験を実施するに際して、東京大学と KEK(高エネルギー加速器研究機構) に試作機を稼働させるためのテストベンチを準備した。テストベンチは ATLAS 実験室の TGC 検出器周辺のエレクトロニクス設備に近い環境になっており、実際の本番設備に出向かなくても開発回路の動作試験や TGC 検出器エレクトロニクスシステムの実証試験が実施できるようになっている。

テストベンチの概要は図 4.1 の通りである。回路室側には、6U/9U[*2]の crate を置き、HSC crate 内の master module を操作するために Bit3 と Crate Control Interface (CCI) を挿入する。本論文では述べないが SL もこの crate に置く。また、その他に HSC VME 操作用の LinuxPC や光 Ethernet 用の Switching Hub を設置する。実験室側には、HSC crate を置き、VME master と 2 台の JATHub 第 1 試

[*1] Windows 10 上で走る仮想マシン (VMware) で起動している。
[*2] 6U や 9U とは crate の縦の大きさを表す。

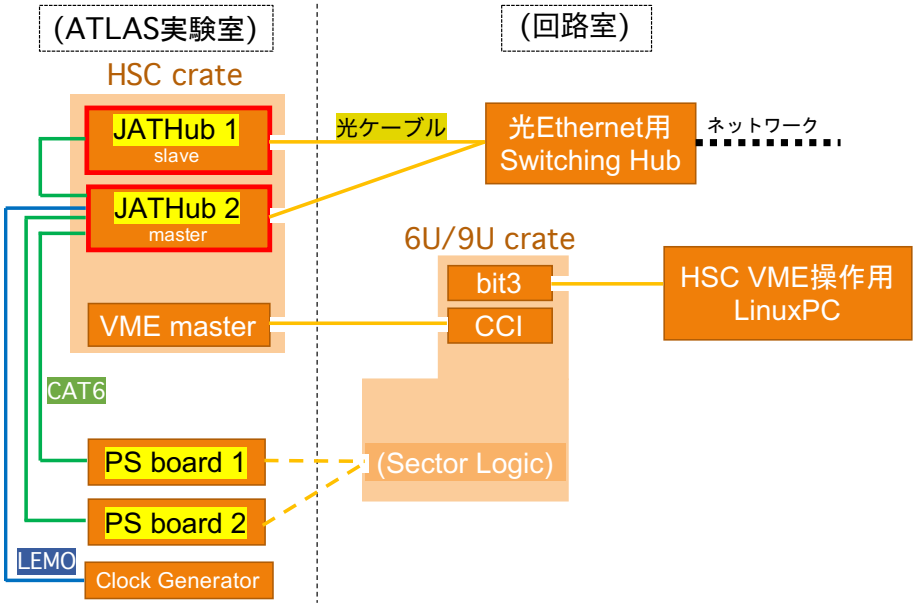


図 4.1 テストベンチの概要。本番の TGC 検出器エレクトロニクスとほぼ同様のシステムを構築し、開発回路の動作試験を行う。今後は SL や ASD なども加えて TGC 検出器エレクトロニクスシステムの実証試験も行う予定である。

作機を挿入する。また、PS board 試作機や Clock Generator も設置する。

Switching Hub から JATHub へ光ケーブルを接続する、もしくは、LAN ケーブルを直接 JATHub に接続することで、JATHub 第 1 試作機がネットワークに繋がるようにする。そして、JATHub から CAT6 ケーブルで slave module に接続する (詳細は本節後半)。また、HSC VME master module を操作するために、LinuxPC から Bit3 へ光ケーブルを伸ばし、Bit3 が VME backplane 経由で CCI を操作し、CCI から HSC VME master module へ光ケーブルを繋げるパスを準備する。更に、本来 Level-0 trigger system の上流から LHC クロックを配布するが、当テストベンチでは LHC クロックを擬似的に Clock generator から出力し LEMO ケーブルで配布する。

実際に東京大学と KEK にテストベンチを作成した。図 4.2 は KEK のテストベンチの様子である。JATHub 第 1 試作機を稼働させるための手段として、卓上で LV 電源から電源供給する方法、もしくは、当テストベンチの HSC crate に挿入し VME J3 backplane から電源供給する方法があり、どちらの手段も実施できるようにする。また、KEK テストベンチでは、図 4.2 の (b) のように隣の作業台に 2 台の PS board 試作機を置いた。

なお、必要な機能を全て実装し本番環境に近い状態で JATHub 第 1 試作機を稼働させた際の、最終的な消費電流値は表 4.2 に示す。JATHub には 3.3V の直流電源を供給しており、電流値測定にはクランプメーターを使用した。

■JATHub 第 1 試作機 2 台間の接続 JATHub は隣の JATHub とお互いに制御できる設計になっており、図 4.3 のように、当 JATHub が隣 JATHub の master として JTAG 通信や Recovery 手続きを行うための RJ45 ポート、そして、隣 JATHub の slave module として JTAG 通信や Recovery 手続きをされるための RJ45 ポートが取り付けられている。今回両テストベンチで 2 台の JATHub 第 1 試作機を稼働

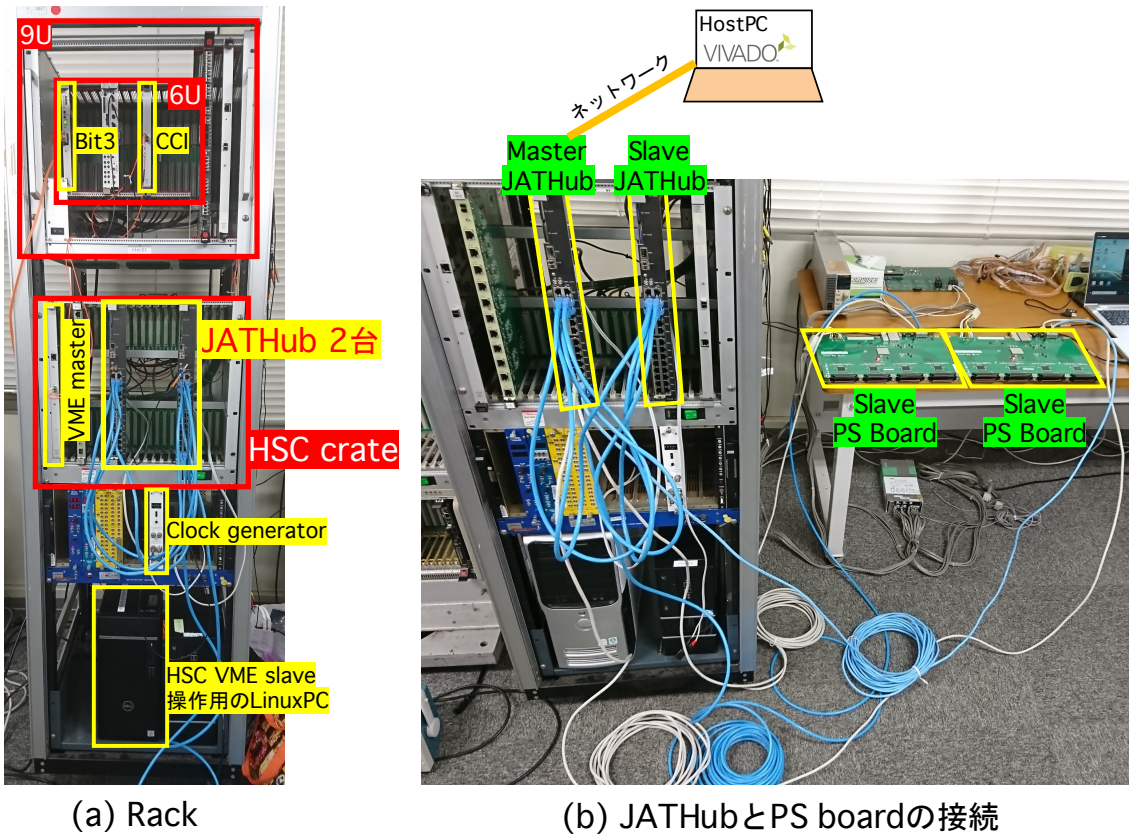


図 4.2 KEK テストベンチの写真。(a)6U/9U と HSC crate を収納した Rack の様子と、(b)Rack の隣の机に置いた PS board 試作機と JATHub 第 1 試作機の接続の様子を示す。(b) では、master JATHub から 2 台の PS boards へ 15m の CAT6 ケーブルで接続している。青色ケーブルは JTAG パス、白色ケーブルは Recovery パスである。

表 4.2 本番環境に近い状態での JATHub 第 1 試作機の消費電力. 電源供給は直流 3.3VD。

JATHub 第 1 試作機の状態	消費電流値 (A)
Zynq(PS+PL) configuration 中	1.61
PS 上の Zynq Linux 起動中	2.25
定常待機状態	2.08
JTAG 通信機能 実行中	2.09
最大使用電流値 (A)	2.25

させるにあたり、2 台間でお互いに制御できるかも試験した。その際、1m の CAT6 ケーブル 4 本を使用して JATHub 第 1 試作機同士を接続した。繋げたポートは図 4.3 に示されており、CAT6 ケーブルで接続する RJ45 ポート同士は同じ配色で示した。slave として接続する場合は上の RJ45 ポートを、master として接続する場合は下の RJ45 ポートを使用した。

本番環境では、1m の CAT6 ケーブルは長いため、0.5m 程の CAT6 ケーブルを準備する予定である。

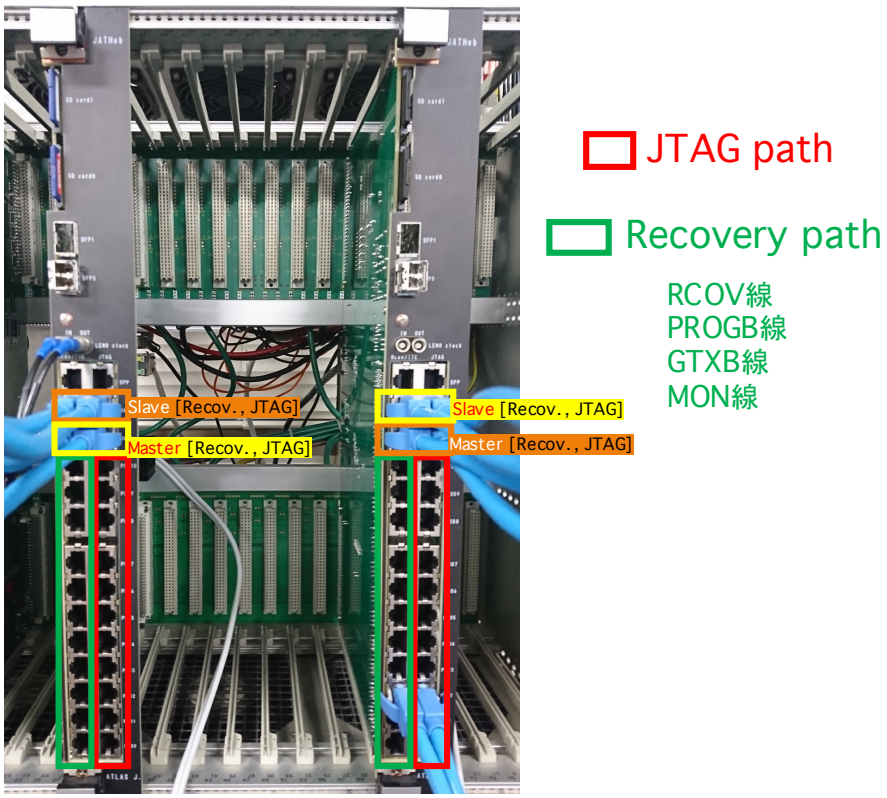


図 4.3　JATHub 第 1 試作機の RJ45 multi-jack. JATHub 間相互接続用 RJ45 ports は CAT6 ケーブルで接続する ports 同士を同じ配色にした。

■JATHub 第 1 試作機と PS board 試作機の接続　今回準備したテストベンチには、図 4.2 のように、卓上にて LV 電源を使用して 2 台の PS board 試作機を稼働させた。また、master JATHub(左) から本番環境を想定した 15m の CAT6 ケーブルを 4 本伸ばし、2 本ずつを各 PS board 試作機に接続した。青色ケーブルは JTAG パス、白色ケーブルは Recovery パスである。JATHub 第 1 試作機において、JTAG パス用の青色 CAT6 ケーブルは図 4.3 の ‘JTAG path’ へ、Recovery パス用の白色 CAT6 ケーブルは図 4.3 の ‘Recovery path’ へ繋げた。

また、図 4.2 の KEK テストベンチでは、1 台の master JATHub 第 1 試作機の配下に、1 台の slave JATHub 第 1 試作機と 2 台の PS boards 試作機がある状態で接続試験を行った。

4.2　光イーサネット通信

JATHub が外部とネットワークで通信できるように、JATHub と 60m-100m 程離れた回路室の光 Ethernet 用 Switching Hub を光ケーブルで繋げる。そのために、光 Ethernet 通信の機能を Zynq デザインに組み込んだ。この光 Ethernet 通信機能は、Xilinx が提供している光 Ethernet 通信用の Zynq デザイン例 [17] を参考に JATHub 第 1 試作機へ実装した。

4.2.1 Zynq における光 Ethernet 通信の仕組み

Zynq はプロセッサー PS を搭載していることから、TCP/IP 規格の Ethernet 通信を行い、外部からアクセスすることができる。CPU が Ethernet 通信を行うためには、物理層 (PHY) において Ethernet 信号を処理し、CPU が取り扱える信号に変換する必要がある。一般的には、PHY の集積回路 (PHY chip) を同回路上に実装する。しかし、Zynq は PHY の信号処理を PL 領域で行うことができる。そのため、Zynq の CPU は、PHY chip を使用しなくても、PL の GTX transceiver に繋がる SFP connector を経由してネットワークに接続できる。SFP connector には、SFP 光 transceiver と SFP-RJ45 adapter どちらでも設置することができ、SFP-RJ45 adapter を使用すると PHY chip を用いない LAN ケーブルによる Ethernet 通信も可能となる (詳細は G 章)。JATHub 第 1 試作機では、図 4.4 のように、SFP 光 transceiver と光ケーブルを使用して Ethernet 通信の確立を行った。

JATHub 第 1 試作機の Zynq PS は、Ethernet 通信を行う MAC(Media Access Controller) を搭載しており、GMII 規格による通信が可能である。GMII 規格は TX、RX、TX 用クロック、RX 用クロック、MDIO、MDC の 6 種類の信号線を使用する。この MAC には 2 つのインターフェイス ‘eth0’, ‘eth1’ があり、2 種類のパスから Ethernet 通信を同時に確立することができる。‘eth0’ は、MIO ピンと接続する回路上素子の PHY chip を介して、一般的な LAN(Copper) ケーブルによる Ethernet 通信に使用した。そして、‘eth1’ は、PS-PL 間を繋ぐ Extended MIO(EMIO) ピンを介して、光 Ethernet 通信に使用した。

Zynq の PL には GMII 規格のインターフェイスを設けた PHY(1000BASE-X PCS/PMA と呼ばれる IP ブロック) を実装した。この PHY は 1000BASE-X 規格の信号への変換を行う。1000BASE-X 規格は上下 2 対 (tx,rx) の信号線を使用した場合の Ethernet 通信の規格であり、SFP から伸びる 2 対の光ケーブルの信号線に使用できる。よって、この PHY で MAC の eth1 と GTX transceiver を接続すると、Zynq の MAC は光ケーブルで外部と 1000BASE-X 規格の Ethernet 通信を確立できる。

またテストのために、eth0 側では、MAC と PHY chip 間は (R)GMII 規格、PHY chip からは 1000BASE-T 規格の信号で Ethernet 通信を行っていた。Copper ケーブルの場合 4 対 8 線の信号線があるので、1000BASE-T 規格の信号が使用できる。

4.2.2 光 Ethernet 通信の動作試験

セットアップとしては、図 4.5 のようにした。HostPC には光ケーブル用のポートがなかったため、Media Converter を介して通信試験を行った。HostPC から伸ばした Copper ケーブルを Media Converter に繋げ、規格変換を行い、Media Converter の SFP ポートから光信号による通信を行った。Media Converter の SFP ポートと JATHub 第 1 試作機の SFP ポートを光ケーブルで繋げ、JATHub 第 1 試作機が 1000BASE-X 規格の光信号と通信できるようにした。IP address を表 4.3 の通りそれぞれ設定し、ローカルネットワークを HostPC と JATHub 第 1 試作機の間で構築できるようにした。

結果、図 4.6 の通り、HostPC から JATHub 第 1 試作機へ ping, ssh を送ることができた。また、ssh によって JATHub 第 1 試作機上で稼働する Zynq Linux にアクセスすることもできた。以上の事から、JATHub 第 1 試作機と光 Ethernet 通信を確立することに成功した。

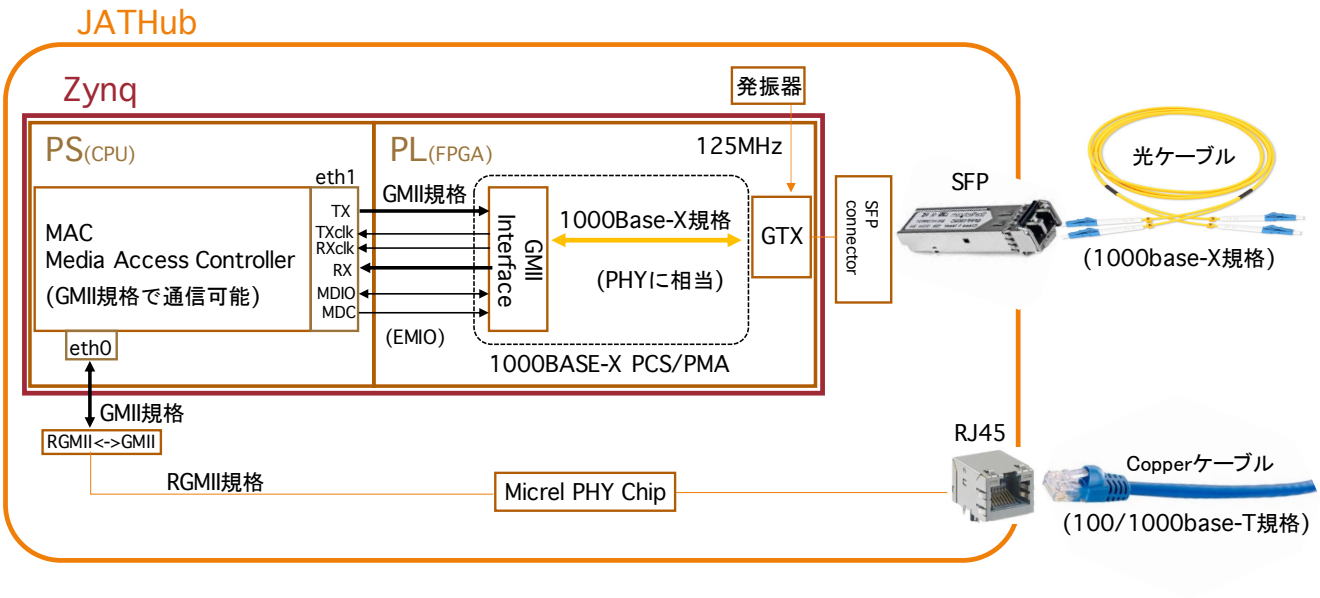


図 4.4　光 Ethernet 通信の仕組み。PL 領域にて PHY を実装し、SFP からの 1000BASE-X 規格 Ethernet 通信を CPU の MAC に繋げる Zynq デザインにした。

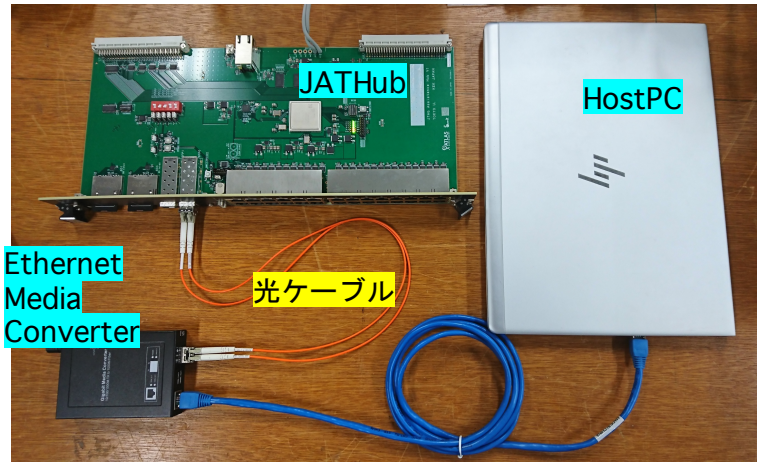


図 4.5　光 Ethernet 通信試験のセットアップ。HostPC に光ケーブル用のポートがないため、Media Converter を介して光ケーブルによる接続を行った。

4.3　JTAG 通信による Slave module の制御

一般的に、FPGA や Zynq を制御するためには、Vivado HM が走る HostPC を近くに置き、短い JTAG ケーブルを繋げなければならない。しかし、ATLAS 実験室で実装される FPGA や Zynq 搭載の回路は実質的にアクセス不可能な場所に置かれている。そこで、JATHub は中継役となって、CAT6 ケーブルで繋がる複数の遠隔回路を JTAG 通信により制御できる設計になっている。JTAG 通信で行う動作は以下 3 点である。

表 4.3　光 Ethernet 通信試験でのネットワーク設定

デバイス	IP address
HostPC (Windows 10)	192.168.10.2
JATHub 第 1 試作機	192.168.10.1

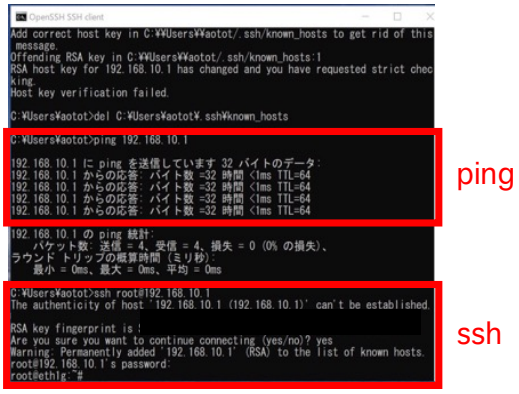


図 4.6　光 Ethernet 通信試験の結果。HostPC から JATHub 第 1 試作機へ ping, ssh を送れた。

- FPGA に直接 Firmware をプログラムする。
- QSPI flash memory にプログラムする。FPGA 用 QSPI には Firmware 情報が入った configuration file をプログラムし、Zynq 用 QSPI には Boot ファイルをプログラムする。
- Zynq、FPGA 内の Firmware のデバックをする。

module にリセットをかけた際、QSPI 内のファイルが読み込まれて、reboot や reconfiguration が行われる。そのため、フロントエンドでの本番運用時に一番重要な動作は QSPI プログラムである。

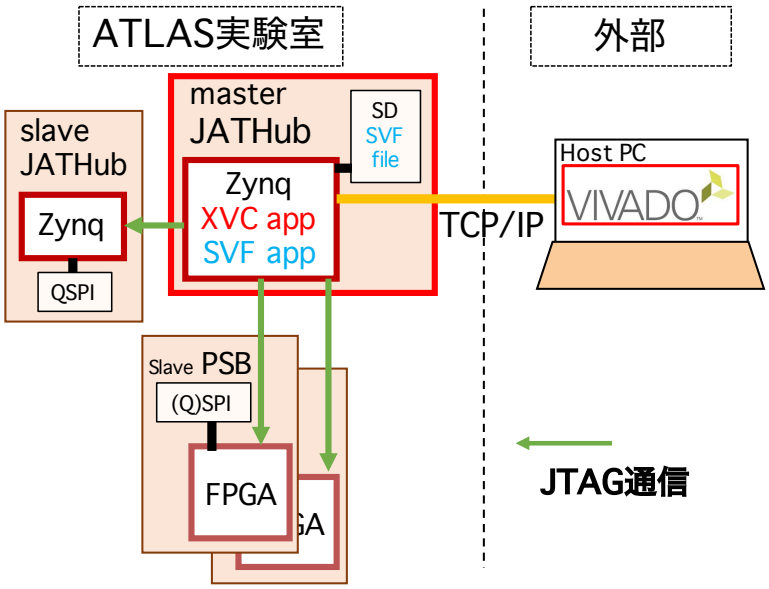


図 4.7　JATHub による JTAG 制御概要図。XVC による操作と SVF による操作が行える。

JATHub 第 1 試作機には、以下の 2 種類の JTAG 通信手段を実装した。

i. **Xilinx Virtual Cable(XVC)** : Vivado HM の GUI や Xilinx Software Commandline Tool(XSCT) の CUI を使用して、TCP/IP 通信を介して遠隔の slave module を制御する機能
ii. **SVF player** : JATHub の Zynq Linux にアクセスして、Zynq Linux の CUI を使用して遠隔の slave module を制御する機能

JTAG 通信の動作試験では、図 4.7 と図 4.2 の通りセットアップし、1 台の master JATHub 第 1 試作機から、1 台の slave JATHub 第 1 試作機と 2 台の slave PS board 試作機へ JTAG 通信した。PS board FPGA(Kintex-7) 用の 2 種類 Firmware を用意し、一目で Firmware の Program が成功したか確認できるようにした。2 つの Firmware の違いは LED が点灯、もしくは点滅するかである。そして、PS board の QSPI flash memory へ program 試験も実施するために、それぞれの Firmware の情報を入れた configuration file も準備した。

以下、それぞれの JTAG 通信手段の実装と動作試験、そして、本番時の運用スタイルについて述べる。

4.3.1 Xilinx Virtual Cable(XVC)

Xilinx Virtual Cable(XVC) は、図 4.7 のように、TCP/IP 通信で Zynq SoC にアクセスし、専用のアプリケーション (XVC app) を Zynq SoC 内で走らせることで、Zynq SoC が別回路の FPGA や Zynq に JTAG 通信するプロトコルである。この XVC プロトコルを利用することにより、Xilinx ソフトが走る HostPC がネットワークで master Zynq SoC にアクセスできれば、遠距離の slave 回路上の FPGA や Zynq に対してあたかも近くで繋いでいる様に Xilinx ソフトの機能を使用することが可能となる。Xilinx ソフトには Vivado HM と XSCT があるが、Vivado HM の機能は使い勝手が良く、具体的には以下のような機能が XVC 下で使用できる。

- **FPGA のプログラムや QSPI flash memory のプログラム** : FPGA や Zynq PL 部に Firmware をプログラムすることができる。また、回路上の FPGA に接続する QSPI flash memory に configuration file をプログラムすることもできる。尚、Zynq に接続する QSPI は、Zynq PS 領域にある QSPI controller が管理しており、FPGA による QSPI 操作と仕様が大幅に異なるため、Zynq 用 QSPI への Boot ファイルのプログラムは XVC ではサポートされていない。
- **Integrated Logic Analyzer(ILA)** : FPGA や Zynq の User I/O pin に繋がる信号線の挙動を、オシロスコープでプローブするように見ることができる。Firmware 開発では信号処理のデバッグ用ツールとして重宝されている。
- **Integrated Bit Error Ration Test(IBERT)** : GTX transceiver の監視及び性能評価を行うことができる。SFP などを使った高速光通信の信号中のノイズや、通信速度などを測ることができる。光通信のパフォーマンス試験に使用する。

4.3.1.1 XVC の開発と実装

Xilinx 社は XVC の Zynq デザイン例 [18] を提供している。Zynq Linux で XVC app が実行され、Zynq PL 内の Buffer を通って PL から伸びる JTAG4 線をドライブする仕掛けになっている (図 4.11)。しかし、この Zynq デザイン例で使用される評価ボードは違う型番の Zynq デバイスを使用しているため、JATHub の Zynq に適合するようにパラメータやソースコードを編集して実装した。具体的な変更点は以下の通りである。

- XVC app にて記述されていた Zynq 内のメモリの大きさを、JATHub に搭載した Zynq の値に直した。
- Zynq デザイン例では、master Zynq1 台に対して slave 1 台を JTAG 制御するデザインとなっていた。これを最大 12 台 (PS board: 11 台, JATHub: 1 台) の slave module に対して JTAG 制御できるように大幅な変更を加えた。
- PS board との接続では 15m の JTAG ケーブルを使用するので、ケーブルのデータ伝送によって遅延時間が増え、同一のクロック位相内で Master と Slave 間のデータ通信ができなくなる。そのため、JTAG のクロック (TCK) 周波数を遅くしてクロック位相の幅を広げ、データが遅延しても同一位相内で読めるようにした。尚、通信状態が安定する最も速い JTAG の TCK 周波数は、slave PS board: 2.5MHz、slave JATHub: 3.125MHz であった。TCK 周波数はこの値で設定した。また、EIL4 用 PS board との接続に 30 m の CAT6 ケーブルを使用するので、今後 30 m の CAT6 ケーブルでの JTAG 通信試験も行う。
- master Zynq が自らの configuration bank に接続し、master Zynq 内の信号線の様子を debug できる “Debug Bridge” という機能も、XVC の仕組みを利用することで実装できた。詳細は H 章。

4.3.1.2 XVC の動作試験

図 4.2 の KEK テストベンチを使用して、XVC の動作試験を実施した。master JATHub で、XVC app を走らせ、HostPC で開いている Vivado HM から slave JATHub と slave PS board の操作を試みた。図 4.8、図 4.9、図 4.10 が Vivado HM の GUI 画面である。

■Hub 機能を持った slave module の configuration XVC を使用して slave module を Vivado HM に認識させるには、まず master JATHub の ‘IP address’ を指定し、master JATHub のどのポート (図 4.3) を開けるか ‘Port No.’ を指定した。すると、Vivado HM にて、図 4.8 のように slave module 全 12 台分のポートで接続準備が完了した。ここで、一度に接続して JTAG 通信を行えるポートは 1 つだけであった。

また、実際に slave PS board 試作機を CAT6 ケーブルで繋げているポートを開けると、PS board の FPGA(Kintex-7) を正常に認識した。この Kintex-7 に新たな Firmware を ‘Program’(図 4.8) することもできた。CAT6 ケーブルを他のポートに付け直して同様の試験を行い、PS board 用の全 11 ポートにおいて XVC の機能が使えることが確認できた。

■slave JATHub の認識と ILA の使用 slave JATHub 第 1 試作機も PS board と同様に Vivado HM において認識した。更に、この接続では、slave JATHub の Zynq 内で駆動している信号線の様子を、図 4.9

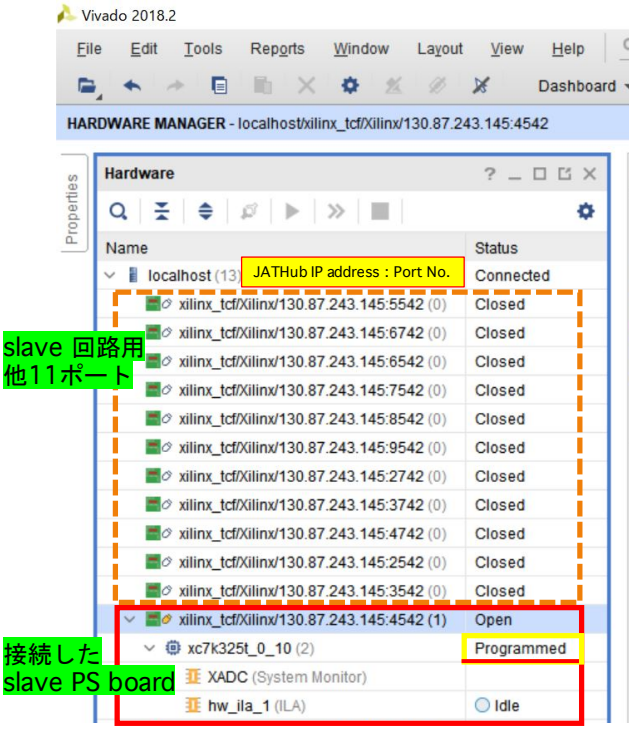


図 4.8　XVC によるハブ機能付き configuration。Vivado HM において、Hub 機能を持ってポートを開通でき、接続した slave module は正常に configuration した。

のように、ILA にて確認し、成功した。図 4.9 の ILA では slave JATHub がドライブできる JTAG パスの信号線をプローブしてみた。TCK(JTAG のクロック) の立ち上がりをトリガーし、JTAG 線の信号の変動を捉えることができた。また、JTAG4 線の下にある 2 つの信号線は、JATHub 間の Recovery パスの MON 線の上り線と下り線を捉えている。JATHub の MON 線は特に PS から操作しない限り 0 信号をお互いに出力するようにしており、この設定を ILA で確認することができた。

従って、遠隔でもターゲット回路の信号線を ILA でプローブし、設計通りの挙動をしているか確認できることがわかった。

■QSPI flash memory をプログラム　PS board 上の Kintex-7 に繋がっている QSPI flash memory に対して、XVC を使用して Vivado HM から configuration file をプログラムできるか確認した。結果、図 4.10 のように設定を行い、QSPI flash memory に新たな configuration file をプログラムすることができた。ここで、Hardware 欄で認識されている Kintex-7 を右クリックし QSPI device を追加してからプログラムを行った。また、図 4.10 の設定は、PS board のハードウェア設計に沿って決めたものである。

ただし、JATHub 上の Zynq-7000 に繋がっている QSPI flash memory に対しては、XVC 経由でプログラムすることができなかった (Zynq QSPI の使用詳細は 4.6 節)。14pin Ribbon Cable が付いた Platform Cable USB II で HostPC と JATHub を直接接続すると成功したが、この条件では本番環境にて遠隔で Zynq の QSPI flash memory をプログラムできない。XVC は Zynq QSPI のプログラムをサポートしていないので、遠隔で Zynq QSPI をプログラムする方法を確立する必要がある (5.1.3 節)。

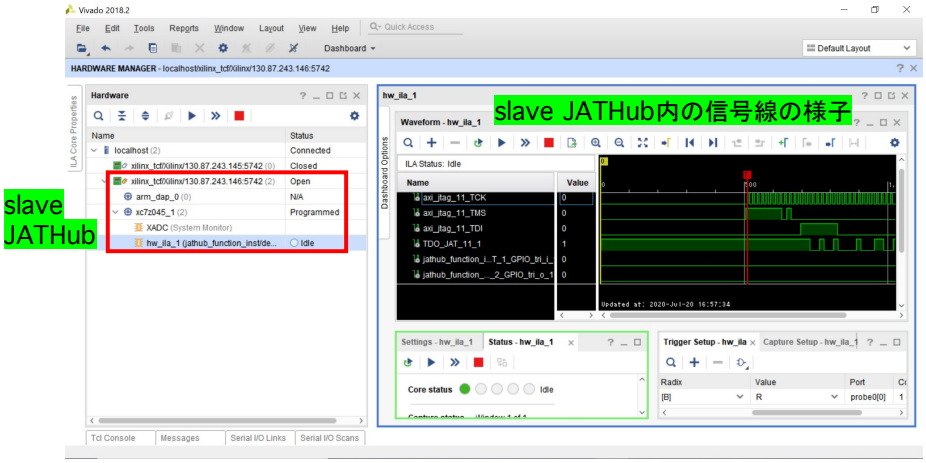


図 4.9 XVC による ILA の使用。Vivado HM だからこそ使える機能の 1 つである ILA を使用して、実際に slave JATHub の Zynq 内の信号線をプローブした。

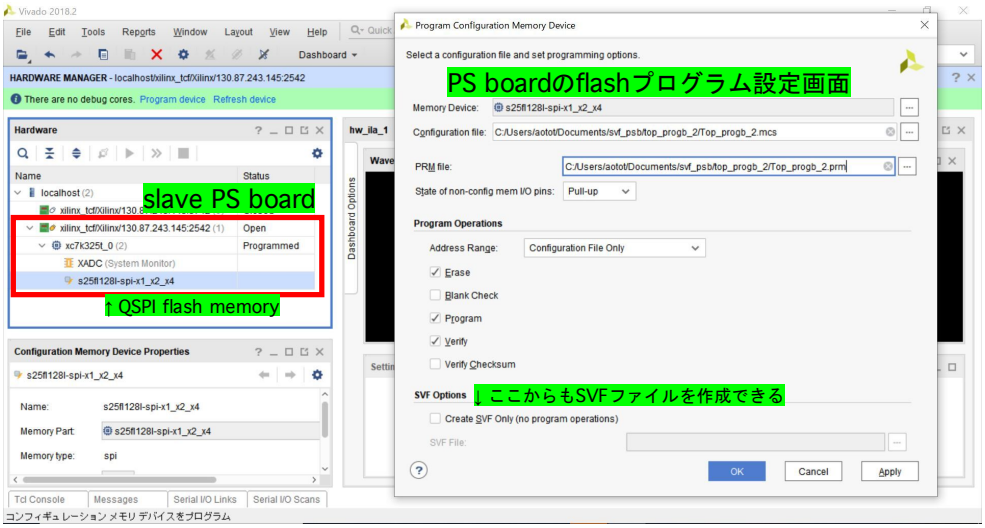


図 4.10 XVC による QSPI flash memory のプログラム。PS board に対しては成功した。JATHub に対しては、Zynq QSPI Program が XVC でサポートされてないので失敗した。

4.3.2 SVF player

SVF(Serial Vector Format) ファイルは JTAG4 線をドライブするパターンを記述した ACSII(テキスト) ファイルである。JATHub では TCP/IP 通信を介さない JTAG 操作を可能にするために、図 4.7 の青字のように、SVF ファイルと SVF player を使って JTAG4 信号線をドライブし、slave module を操作する機能を設けた。ネットワーク経由 (SCP) で SD card に SVF ファイルを置き、Zynq Linux 上で走る SVF player アプリケーションがその SVF ファイルに従って bit-banging 形式で JTAG4 線を操作する。

4.3.2.1 SVF player の開発と実装

C. Wolf[19] の “SVF and XSVF JTAG player” を参考に、C 言語コードを書き直し、メモリサイズや 12 ポート対応など JATHub に最適化した専用の SVF player を開発した。RJ45 の JTAG ポートは XVC と共有しているが、この SVF player は XVC とは独立したアプリケーションである。そのため、Zynq PL 内で XVC による信号線と干渉しないように、合流点でスイッチできる Firmware も開発した (図 4.11)。SVF player がこの Firmware のスイッチを操作できるようにしており、SVF player 実行中のみ SVF のパスを開通させる仕組みにした。

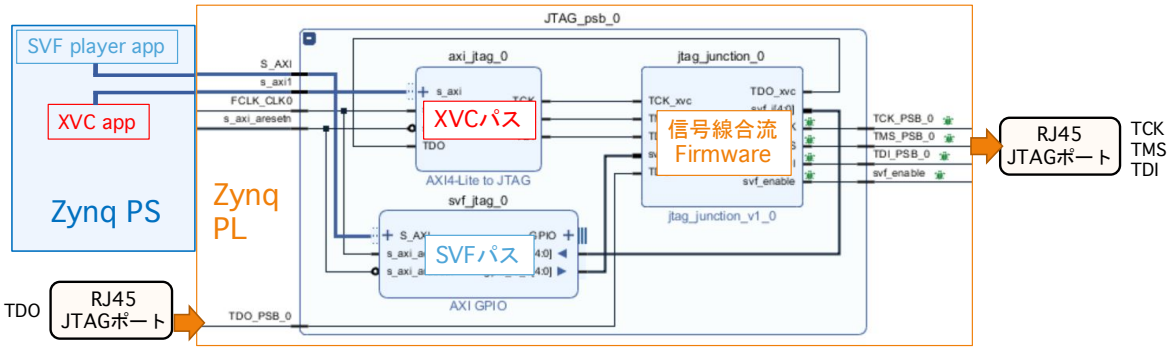


図 4.11 XVC と SVF を実装した Zynq デザイン。左側が Zynq PS とそこで走るアプリケーション、右側が Zynq PL とそこに展開したデジタル回路を示している。XVC と SVF のパスを合流させ RJ45 JTAG ポートに繋げた。

4.3.2.2 SVF player の動作試験

SVF ファイルは図 4.10 のように Slave module を認識した状態でも作成できたが、スタンドアローンで走る Vivado HM だけでも作成できた (詳細は I 章)。図 4.12 のように、master JATHub 上にて作成した SVF ファイルを SVF player で実行すると、PS board の FPGA Kintex-7 の Firmware プログラムと PS board の QSPI flash memory のプログラムが成功した。JTAG のクロック周波数は 0.44 MHz だった。

尚、XVC と同様に、slave JATHub の QSPI flash memory へのプログラムはできなかった。Zynq は PS が QSPI をドライブしており、FPGA の QSPI プログラムとは仕組みが異なるため、Vivado HM は Zynq に繋がる QSPI 用の SVF ファイル作成に対応してない仕様になっている。遠隔で Zynq QSPI をプログラムする方法を確立する必要がある (5.1.3 節)。

4.3.3 開発した JTAG 通信の本番時の運用

JTAG 通信の動作種類毎に、動作試験の結果を踏まえた本番時の運用について述べる。

■FPGA のプログラム、FPGA 用 QSPI のプログラム XVC により、Host PC で稼働する Vivado の GUI からのプログラムに成功した。また SVF player によるプログラムも成功した。本番運用時は、通信の安定性を考慮し、JTAG 通信中に TCP/IP 通信を介さない SVF player を主に使用する予定である。

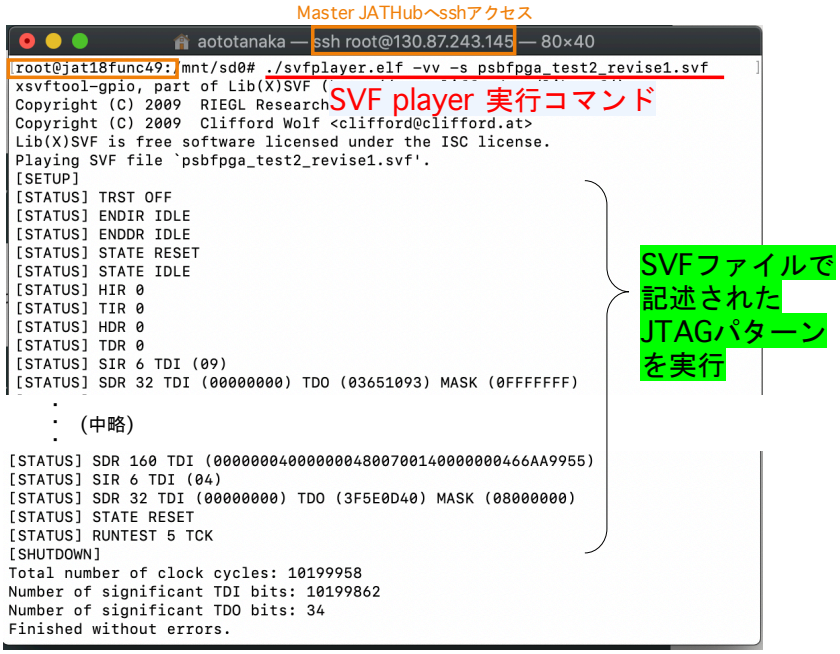


図 4.12　SVF player 実行時の CUI 画面。Zynq Linux に ssh でアクセスし、SD card 内の SVF ファイルを引数として参照しながら、SVF player を実行した。PS board Kintex-7 の Firmware をプログラムする際の CUI 画面。

■FPGA 内 Firmware のデバック　XVC により、Vivado の GUI から ILA などのデバックツールを用いて成功した。SVF player にデバック動作は無いので、本番運用時は、XVC を使用する予定である。

■Zynq 用 QSPI のプログラム　XVC でも、SVF player でも失敗した。5.1.3 節にて詳しく説明するが、遠隔でプログラムする方法を第 2 試作機で確立し、本番ではその方法を使用する予定である。

4.4　Slave module の Recovery 手続き

Slave module は SEM IP を使用しており、自動回復不可能な SEU 事象が発生した時に master JATHub へ active-low の救難信号を送る。その際、master JATHub は救難信号を受信すると自動で対象の slave module を reset するための信号を送る。ここで、reset 信号線は active low のため、0 信号を送ることで slave module の reset が掛かる仕様になっている。JATHub 第 1 試作機では、この Recovery 手続きを行えるよう Zynq PL の Firmware を開発して、テストベンチにて動作検証を行った。

4.4.1　PS board に対して

PS board には 2 種類の reset 線、PROGB と GTXB がある。PROGB 線に 0 信号を送ると、PS board は QSPI flash memory から configuration file を読み込み Kintex-7 の reconfiguration を行う。GTXB 線に 0 信号を送ると、Kintex-7 の GTX transceiver を reset し光通信の再確立を行う。図 4.13 のように、master JATHub では、PS board から伸びる RcvB 線を常時監視しており、0 信号を受信したら両方の

reset 信号を PS board に送信する。また、PS の自作アプリを実行することによっても各 reset 信号を送信する。reset 信号を送る上で、以下の機能を実装した。

- Idle 状態の際は PROGB 線と GTXB 線へ 1 信号を送り続ける機能。
- PROGB 線と GTXB 線へ 100ms 信号幅 (詳細は 5.1.1 節) の 0 信号を送る機能。
- Kintex-7 の reconfiguration 後に光通信の再確立を行うため、PROGB 線へ 0 信号を送った 5 秒後に GTXB 線へ 0 信号を送る機能。(図 4.13 の 'Timing control')
- reset 信号を送信した後 10 秒間は、RcvB 線からの救難信号も、PS アプリからの reset 命令信号も一切受け付けずに、両 reset 信号線を Idle 状態にする機能。(図 4.13 の 'Timing control')
- 救難信号の受信拒否を行えるように、RcvB 線の開閉を VME data パス (J1) によって操作する機能。
- 本番実験中に PS アプリから誤った信号が送られないように、PS に繋がるパスの開閉を VME data パス (J1) によって操作する機能。

Zynq PL 内の SEU 対策のため、パスの各所に多数決ロジックも実装した。MON 線に関しては、4.5 節にて説明する。

Kintex-7 で回復不可能な SEU 事象が発生している間、PS board は SL との光リンクが切れてしまい、もし ASD から Hit 信号が来ても SL へ送信できない状態が続く。ここで、PS board と SL 間の光通信で使用している "Bit Map" 中には Kintex-7 の不動作もしくは光リンクの切断を示す flag がある。Kintex-7 が機能していない間は、この flag が立ち、SL 側が該当の PS board の不動作を確認できる。よって、L0 trigger system は PS board がいつ機能不全を起こしたかオンラインで把握することができる。

この Recovery 手続きを全 11 ポート分実装し、各ポートに順番に PS board 試作機を接続して動作試験を行った。PS board 試作機では押しボタンで RcvB 線に救難信号を送る仕掛け、Kintex-7 の configuration が完了すると DONE LED が点灯する仕掛け、GTXB 線の 0 信号を受け取るとテスト用 LED が点滅する仕掛けを準備した。結果、全ポートでの動作試験において、上記の機能が正常に動作した。しかし、master JATHub のパワーサイクルが行われた際、PS board の PROGB 線へ意図せず 0 信号が送られた。その原因究明と対策は 5.1.1 節にて説明する。

4.4.2 隣の JATHub に対して

JATHub には 2 種類の reset 線、PROGB と Power on Reset(PORB) がある。PROGB 線に 0 信号を送ると、Zynq PL のデジタル回路が消され初期状態に戻る。PORB 線に 0 信号を送ると、Zynq が再起動を行う。Zynq PL の reconfiguration は、Zynq の再起動シークエンスに含まれている。そのため、Zynq PL にて自動回復不可能な SEU 事象が発生した場合、Zynq の再起動を行う必要がある。そこで、master JATHub では、図 4.14 のように、slave JATHub の RcvB 線からの 0 信号を受信したら、PORB 線へ 0 信号を送信する。また、PS の自作アプリから両 reset 線を操作できる。reset 信号を送る上で、以下の機能を実装した。

- PROGB 線と PORB 線が Idle 状態の際は 1 信号を送り続ける機能。

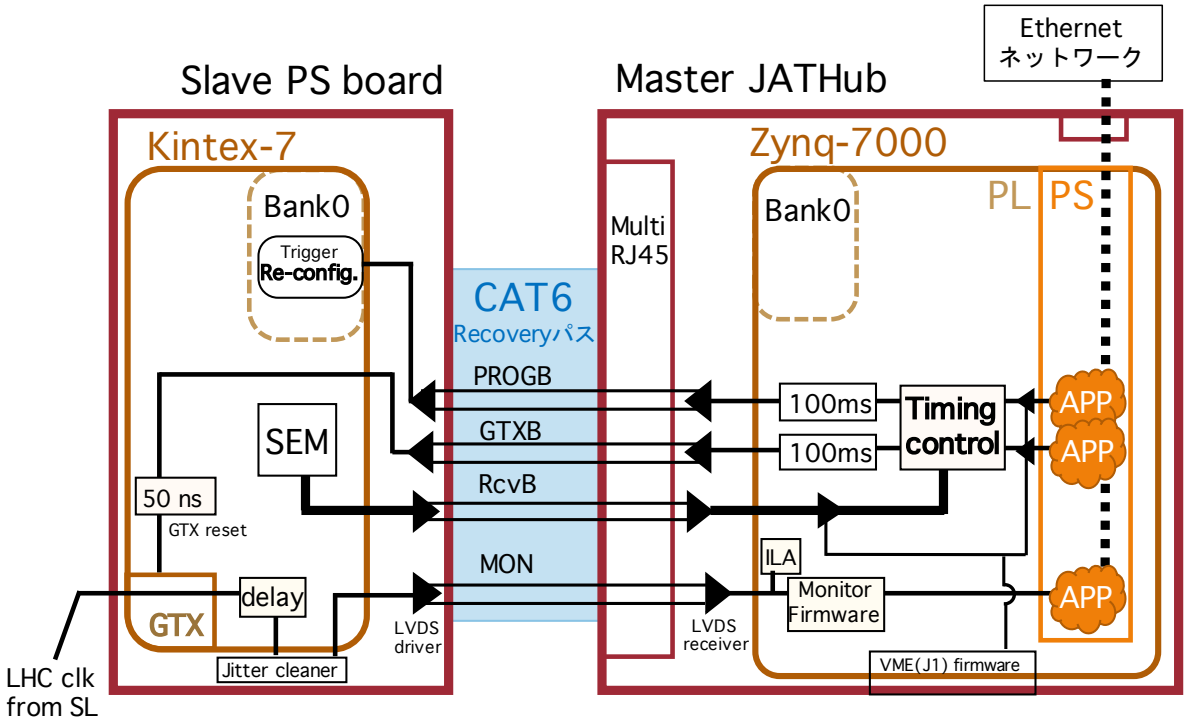


図 4.13　PS board に対する Recovery 手続きの概要。これを全 11 ポート分実装し、正常に動作したことを確認した。

- PROGB 線と PORB 線へ 100ms 信号幅 (5.1.1 節) の 0 信号を送る機能。
- reset 信号を送信した後 10 秒間は、RcvB 線からの救難信号も、PS アプリからの reset 命令信号も一切受け付けずに、reset 信号線を Idle 状態にする機能。(図 4.14 の ‘Timing control’)
- 救難信号の受信拒否を行えるように、RcvB 線の開閉を VME data パス (J1) によって操作する機能。
- 本番実験中に PS アプリから誤った信号が送られないように、PS に繋がるパスの開閉を VME data パス (J1) によって操作する機能。

こちらもパスの各所に多数決ロジックを実装した。MON 線は、slave 側から 0 信号を送り続け、master 側で 0 or 1 を PS アプリで読む仕組みにした。LVDS の Fail Safe 機構 (3.2.2.2 節) により、slave JATHub の Zynq がクラッシュした場合に 1 信号が受信できるので、遠隔で slave JATHub の状態を確認できる。

もう一台の JATHub 第 1 試作機と接続して、この Recovery 手続きの動作試験を行った。もう一台の JATHub 第 1 試作機では、PS アプリから救難信号を送信する仕掛けを準備し、再起動を確認するため UART 通信による Linux のコンソール画面を用意した。また、JATHub 第 1 試作機は、Zynq PL の configuration が完了すると DONE LED が点灯する仕様である。結果、上記の機能が正常に動作した。しかし、master JATHub が再起動を行うと、slave JATHub の PROGB 線へ意図せず 0 信号が送られた。その原因究明と対策は 5.1.1 節にて説明する。

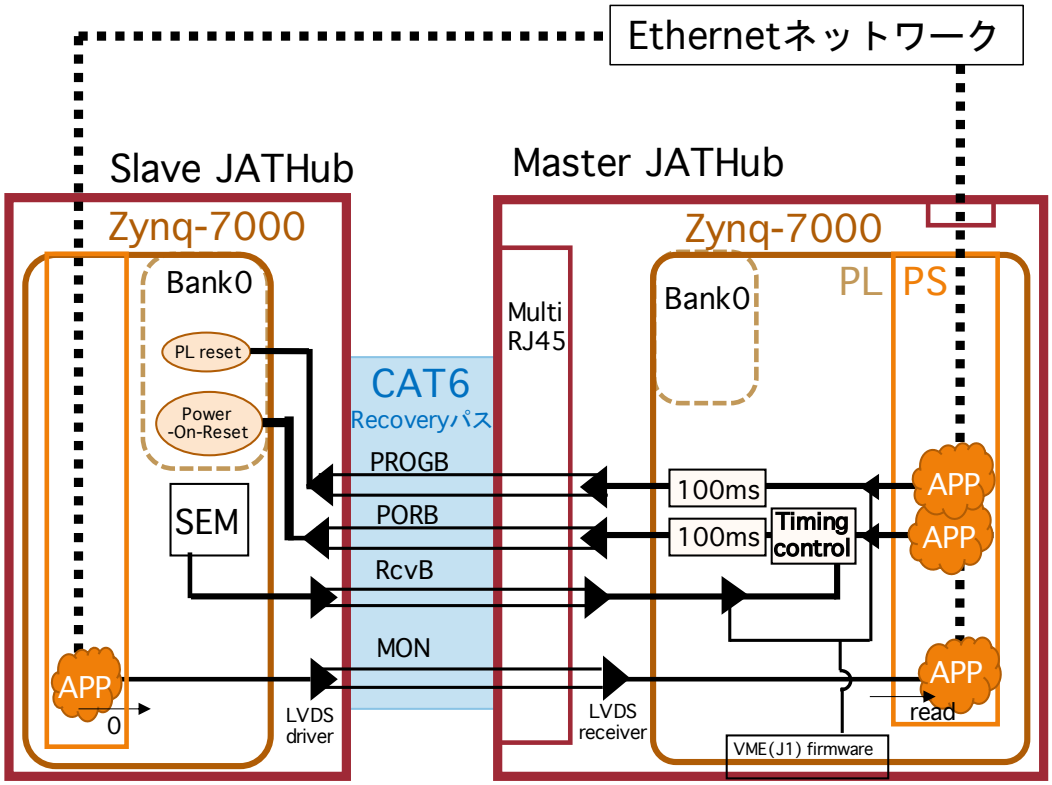


図 4.14　隣の JATHub に対する Recovery 手続きの概要。正常に動作したことを確認した。

4.5　LHC クロック位相のモニター

図 4.13 のように、PS board は、SL から 8b/10b プロトコルの光通信によりデータを受信し LHC クロックをデコードした後、Jitter cleaner 素子でノイズを除き、Kintex-7 の GTX transceiver の参照クロック及びシステムクロックとして LHC クロック (25 ns) を使用する。この LHC クロックの位相を全 PS board において O(100) ps の精度で揃えるために、JATHub は MOS 線を使用して最大 11 台の PS board の LHC クロックをモニターする。

4.5.1　モニター方法の実装

モニター方法は、図 4.15 のように 2 種類実装した。1 つ目が ILA による MON 信号線のプローブで、最高精度は 1.25 ns となる。2 つ目が専用の自作 Firmware によるモニターで、最高精度は 1/56 ns となる。

ILA による MON 信号線のプローブでは、JATHub 上の水晶発振器から参照クロックを取り入れ、そのクロックの立ち上がりで MON 信号を測定している。Zynq が扱える参照クロックの最大周波数は 799 MHz のため、最高精度は 1.25 ns と 3.1.3 節の要求を満たせない。

専用の自作 Firmware では、図 4.15 のように、PS board からの LHC クロック (①) と、LEMO rx から取り入れた外部の独立した参照用 LHC クロック (②) を比較する。②が立ち上がるタイミングで、①が 0 or 1 であるかを、Monitor アプリで読み出す。ここで、②は Clock Wizard IP[20] の “Dynamic Phase

Shift”機能によって、1/56 ns ずつ位相をずらすことができる。したがって、②の位相を 1/56 ns × N 回変位させる毎に①を読み出し、これを 1 波長 (25 ns) 分行い、図 4.16(b) のようにまとめると、N/56ns の精度で①の 1 波長の波形を測定できる。

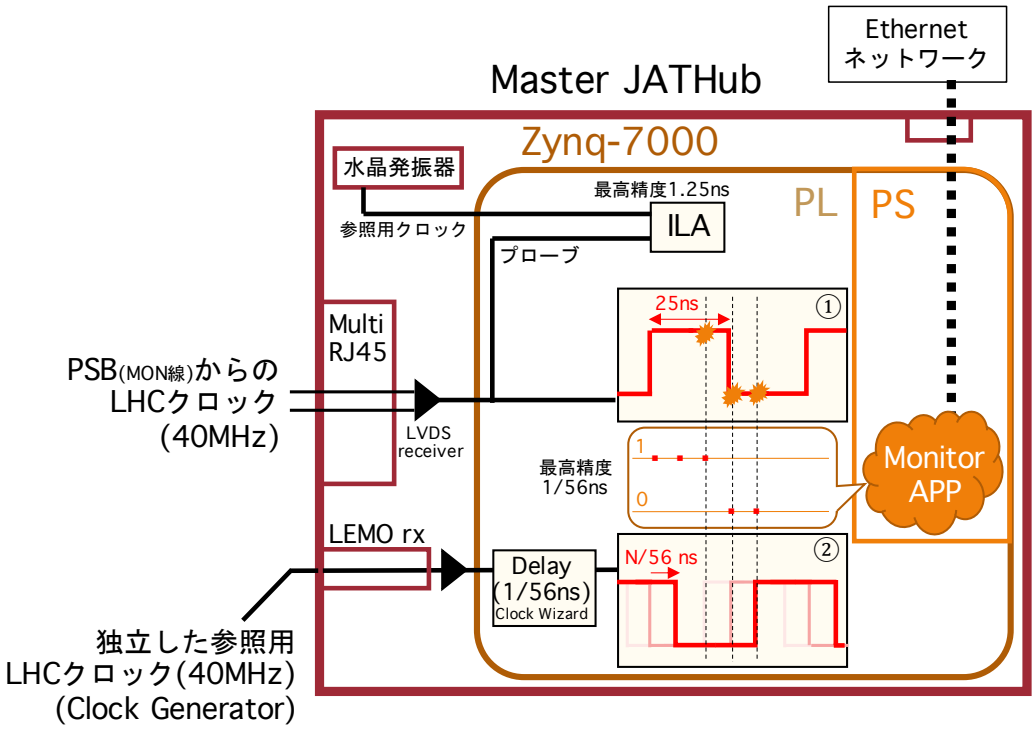


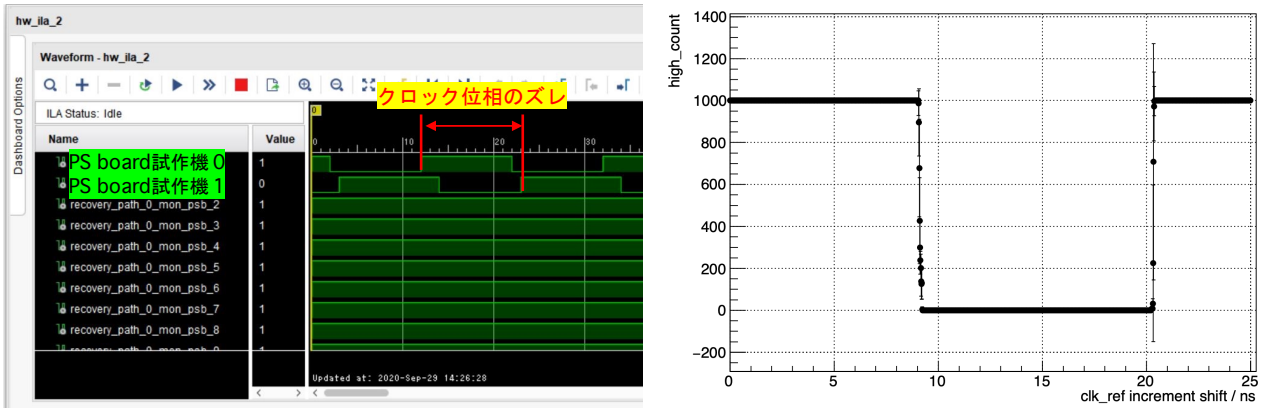
図 4.15　JATHub にて LHC クロックをモニターする方法。1 つは ILA を使用する。もう 1 つは専用の自作 Firmware を使用する。

4.5.2　モニター試験

LHC クロック位相の上記のモニター機能を JATHub 第 1 試作機に実装し、動作試験を行った。

ILA によるモニターの結果は図 4.16 の (a) の通りであった。2 台の PS board 試作機を接続し、それぞれの MON 線から 12.5 ns (20 MHz) のテストクロックを受信した。ILA の参照クロックを 2.5 ns (400 MHz) とした。そして、それぞれのテストクロックの位相は 27.5 ns(11 目盛) ほどの差ができていることをモニターできた。

自作 Firmware によるモニターの結果は図 4.16 の (b) の通りであった。②の LHC クロックは、図 4.2 の “clock generator”から受け取った。1/56 ns 毎に位相を変位し、各位相で 1000 回①の読み出しを行った。図 (b) の縦軸は 1000 回中何回 ‘1’ を読み出したかを示し、横軸は①を読み出した時点で変位した位相 (per 1 波長 (25 ns)) を示す。そして、PS board から送られる LHC クロックを 1/56 ns の精度でモニターできた。



(a) LHCクロック位相のズレ　(b) 1/56ns精度のLHCクロックモニタ

図 4.16　LHC クロックのモニター試験結果。(a) が ILA によって PS board 毎にテストクロックを比較した様子。試験では 12.5 ns のテストクロックを 2.5 ns の精度で測定した。(b) が自作の Monitor Firmware によって LHC クロックを 1/56 ns の精度で測定した様子。

4.6　冗長性のある BOOT システム

JATHub は、3.1.1 節にて説明した通り、高い放射線環境下においても安定して Zynq の起動を行う必要がある。

ここで、Zynq の起動に使用する Boot ファイルは flash memory に保管される。もちろん JATHub の flash memory 素子は、ATLAS 実験が定めた放射線耐性の要求値を満たす素子を使用する予定である。しかし、一般的に flash memory は放射線耐性が弱く、永久的な放射線損傷を受けやすいと知られている。

そのため、JATHub では、QSPI flash memory と SD card flash memory により複数の Boot ファイルを用意して、PS が自由に有効な Boot ファイルを選択できる BOOT システムを確立した。この BOOT システムにて、安定して起動したこと、JATHub の機能が正常に機能したことを確認した。もし、Boot ファイルもしくは flash memory が損傷した時は、この冗長性のある BOOT システムによって Zynq の再起動を行い、ATLAS 実験室に入れる実験時間外の時期に JATHub ごと取り替える。本節では、この BOOT システムの実装について説明する。

4.6.1　BOOT システムの仕組み

Zynq-7000 SoC の Boot ファイルは 2 種類に分けられ、図 4.17 のように使用される。

① **Initial Boot ファイル**:PS が最初に読む Boot ファイル。ファイル Header や、PS の configuration 情報、②の展開方法が書かれてある[*3]。PS がどちらの SD card(0 or 1) を読むかもここで指定できる。また、SD card 内のテキストファイル (uEnv.txt) も PS に読ませ Zynq の環境設定に反映させ

[*3] BOOT.BIN(fsbl.elf, u-boot.elf)

る。ファイルサイズは 550 KB ほどである。同じ内容の Initial Boot ファイル (①-a,①-b) を QSPI flash memory にプログラムするのだが、①-a は 0x0、①-b は 0x0 以外 (動作試験では 0x90000) の offset 値の場所にそれぞれの Header の最初の Byte が来るようプログラムする。デフォルトでは、Zynq の仕様により、offset 値が最小の①(①-a) が読まれる。

② **Second Boot ファイル**：PS の configuration 後に PS が読む Boot ファイル。Linux イメージや PL Firmware などが書かれてある[*4]。ファイルサイズは 25MB ほどである。同じ内容の Second Boot ファイル (②-a,②-b) をそれぞれの SD card に置く。デフォルトでは、①-a の設定により、SD 0 にある②(②-a) が読まれる。

①と②の Boot ファイルは Petalinux でクロスコンパイルして作成するが、コンパイルには時間と手間がかかる。回路毎にコンパイルし直すことを避けるため、JATHub 回路毎に異なる環境設定を行う場合は、uEnv.txt という簡単に作成できるテキストファイルを作成して、SD card に置く。つまり、150 台ほどある JATHub 回路に対して、①と②の Boot ファイルは共通の物を使用し、uEnv.txt のみ異なる物を使用する。デフォルトでは、SD 0 にある uEnv.txt が読まれる。

この BOOT システムでは、QSPI 内の Boot ファイルを先に読む “QSPI Boot Mode” を実行している。ところで、Zynq-7000 は SD card 内の Boot ファイルのみを参照する “SD Boot Mode” も実行できるが、Zynq-7000 の仕様により Boot 時は SD 0 の Boot ファイルしか読めない [13]。そのため、SD Boot Mode では次節のように冗長性を持たせることができない。

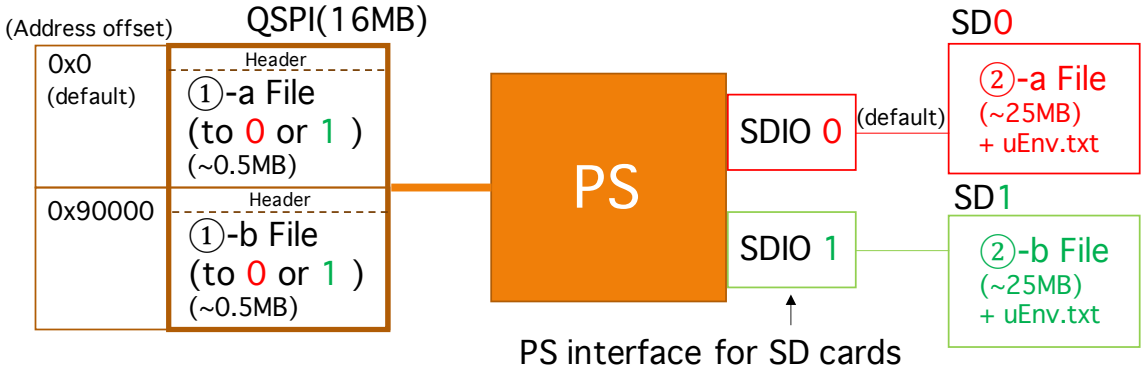


図 4.17 冗長性のある BOOT システム概要。複数の Boot ファイルを PS(プロセッサー) が自由に選択して、Zynq を起動する。

4.6.2 冗長性を保障する動作

デフォルトで読まれる Boot ファイルに問題が発生した場合、どのように予備の Boot ファイルを読み込んで Zynq を起動させるかを図 4.18 に沿って説明する。

[*4] image.ub(kernel, device-tree, zynq linux image), system.bit

■①-a の読み込みに失敗した場合　①-a の Header が放射線損傷により読めない場合、PS で稼働する“Boot Header Searcher”が異常を検知し、次に小さい offset 値にプログラムされた Boot ファイル (①-b) を自動で探し読み込む。動作試験では、①-a の Header を意図的に傷つけ再起動させて、Zynq の環境設定を確認すると、offset 値 0x90000 の①-b ファイルを読んでいることが確認できた。

しかし、①-a の Header 以外の中身が損傷して読めない場合、異常が検知できず、Zynq が正常に起動しない。そのため、外部から遠隔手動で QSPI 内の①-a の上書きが必要になる。4.3 節より隣の JATHub から QSPI の上書きが行えないため、5.1.3 節の対策を投じた。

なお、Zynq が正常に起動している場合は、①ファイルを一度 SD card に保管し、Zynq Linux のコマンド[*5]によって QSPI にプログラムすることができる。

■②-a, uEnv.txt の読み込みに失敗した場合　SD 0 内の Boot ファイルには、同一 SD card 内に予め複製した参照用 Boot ファイルを置いている。Zynq の起動中、PS は SD 0 内に Boot ファイルが存在していることと、参照用 Boot ファイルと一致していることを確認してから、Boot ファイルを読み込む。一致していなければ SD 1 の Boot ファイルを自動で読み込み始める。そのため、放射線損傷や SD card 認識の失敗などで、②-a や uEnv.txt ファイルの読み込みに失敗した場合、PS は自動的に SD 1 内のファイルを参照する。動作試験では、②-a が入っている SD 0 を SD スロットから取り外したり、②-a のファイルを一部損傷させてから、Zynq を再起動させた。結果、Zynq が自動で②-b を読んで正常に Zynq が起動したことを確認した。

バックアップとして、②-a や uEnv.txt の読み込みに失敗し、上記の自動読み込みも失敗した場合、SD card の読み出し先を SD 1 に指定変更した新しい①-a を上書きする手段も設ける。外部から遠隔で QSPI のプログラムを行うので、5.1.3 節の対処が必要となる。

なお、Zynq が正常に起動している場合は、SD card へネットワーク経由でファイルを転送[*6]することができる。

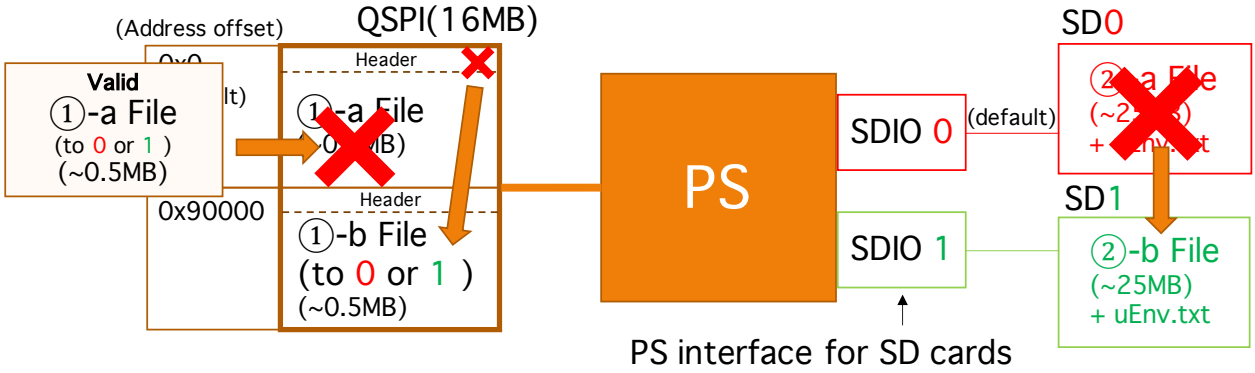


図 4.18　冗長性を保障する BOOT 動作。Boot ファイルが損傷した場合に備えて予備の Boot ファイルを読み込める機構になっている。

[*5] flashcp (hoge.BIN or hoge.mcs) /dev/mtd0
[*6] scp hoge.* {IP address}:/mnt/sd0 or 1

4.7 VME 操作

Big Wheel の JATHub は HSC VME crate に挿入されるので、JATHub の register 操作の backup パスとして、VME パスによる操作を実装する。第 1 試作機では、図 4.2 の通り、VME 操作用 Linux PC → Bit3 → CCI → HSC crate の master module → JATHub 第 1 試作機というパスで動作試験を行った。

動作試験では、第 1 試作機の PL に VME slave module の protocol 処理用 Firmware とテスト register を実装し、VME backplane を経由して操作できるようにした。テスト register には 0x2020 を初期値として設定した。結果、図 4.19 の (a) のように、JATHub までの VME パスを開通していき、テスト register の値を読むことができた。また、図 4.19 の (b) のように、テスト register を 0x2021 へ上書きすることもできた。

したがって、VME による JATHub の register 操作は成功した。

図 4.19 VME による register 操作試験。VME 操作用 Linux PC が VME slave module である JATHub 第 1 試作機の PL 内の register 値を操作する。

第 5 章

JATHub 第 2 試作機の製作

JATHub 第 1 試作機の動作試験を行った結果、修正すべき問題点が数カ所発見された。これらの問題を修正した JATHub 第 2 試作機を製作し、再び他回路との接続試験を行い、修正デザインに問題が無いことを確認する。また、第 2 試作機のデザインをそのまま JATHub 量産機に使用する予定である。

本章では、JATHub 第 1 試作機における問題点を整理し、具体的な対処方法と修正したデザインについて述べていく。

5.1 JATHub 第 1 試作機からの修正点

本節では、主に PCB レイアウトの修正に関わる点や、本番時の設計に合わせた修正点について述べる。

5.1.1 reset 線における LVDS 通信の動作改善

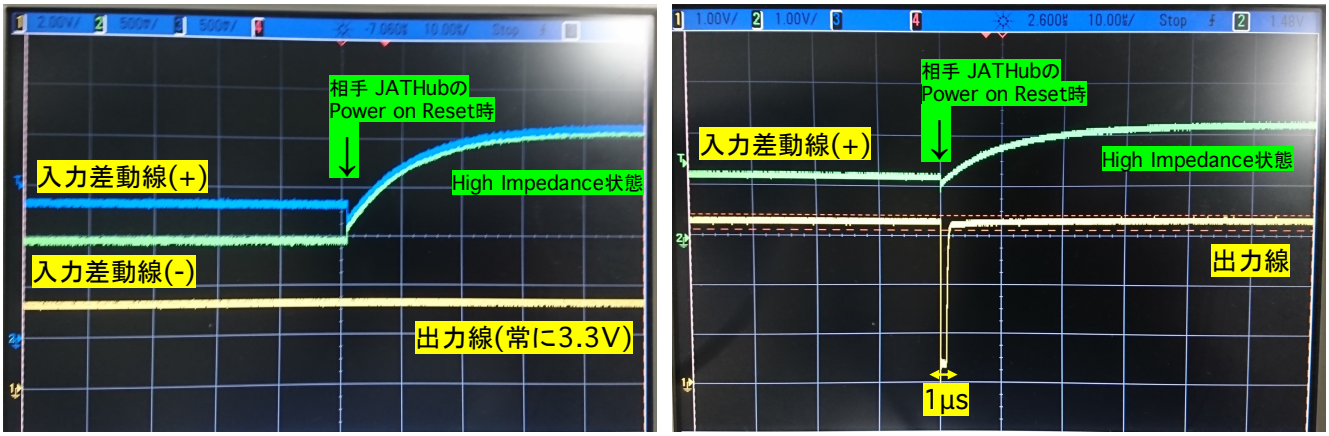
master JATHub は slave module を reset できる設計になっている。全 module の reset は active low になっているため、JATHub はアイドル状態時、常に 1 信号を reset 線へ送り続けなければならない。しかし、4.4 節でも言及した通り、JATHub 第 1 試作機では、master JATHub が Zynq の configuration 中に slave module へ意図せぬ 0 信号を送ってしまうという問題が発生していた。master JATHub と slave module の間の LVDS 通信には、3.2.2.2 節で述べた設計通り、“Fail Safe”機能が実装されていた。しかし、原因は、その LVDS 通信を行う回路上の receiver 素子の仕様にあった。

master JATHub では、reset 線は Zynq の User I/O pin から操作できるようになっているが、configuration 中はその User I/O pin が high impedance 状態となっていた。また、configuration 中 JATHub 試作機上の LVDS driver 素子の TXENABLE pin も high impedance 状態となるため、LVDS driver 素子は high impedance 信号を slave module の LVDS receiver 素子へ送信していた。そのため、master Zynq configuration 中も Fail Safe 機能が働き、reset 線には 1 信号が送られていた。

しかし、slave module の LVDS receiver 素子で、受信する信号が 1 から high impedance へ遷移した時、問題が発生した。設計通りならば、図 5.1 の (a) のように、遷移時でも receiver 素子は常に 1 信号を reset 線に出力することを期待していた。だが、図 5.1 の (b) のように、素子は 0 信号 (reset 信号) を 1μs 程出力してしまっていた。その結果、意図せぬ 0 信号が生じて、勝手に slave module が reset されてし

まっていた。実は、この LVDS receiver 素子では、受信信号が high impedance 状態に遷移してから 600 ns 後に、high impedance 状態を検知して 1 信号を出力する仕様となっていた [11]。

ところで、図 5.1 の (b) にて、receiver 素子のマイナス差動入力線をオシロでプローブすると、オシロの終端抵抗により、同一素子の出力線で 0 信号が観測されなくなった。



(a) 想定していた
LVDS receiver周りの信号挙動

(b) 意図せぬ0信号を出力した
LVDS receiver周りの信号挙動

図 5.1 LVDS receiver 素子の入出力信号の挙動をオシロスコープで測定した結果。LVDS driver を操作する FPGA or Zynq が configuration をした時、driver の TXENABLE pin が high impedance となり、driver の出力差動線も入力信号と関係なく high impedance 状態となる。そして、この時、LVDS receiver 素子では、図中の入力差動線も 1 信号から high impedance 信号に変化する。LVDS receiver 素子は入力信号が high impedance に遷移した瞬間は 0 信号が入力されたと検知することがあり、図中の通り 0 信号が出力されることがある。この現象は receiver 素子で個体差があるが、どの receiver 素子においても起こりうるため、対策が必要となる。

この意図せぬ挙動を解決するために、以下 2 点の変更を第 2 試作機に施した。

i. LVDS driver 素子から open drain 信号を出力しない設計にすること。
ii. reset 線にて、LVDS receiver 素子と FPGA or Zynq の間にデバウンサー (Debouncer) を実装し、デバウンサーに一定時間以上 0 信号が入力されると reset 用 0 信号が出力される仕組みにすること。

■(i.) の対処 configuration 中も Zynq の User I/O pin から 1 信号が出力されるように、Zynq-7000User Guide[16] に従って、図 D.28 のように、Bank34 の PUDC_B pin を GND に落とすことにした。括弧書きされた (部品名) は製作時その部品を実装しないことを示している。

PS board 試作機では既に PUDC_B pin が GND に落とされていたため、configuration 中の LVDS driver 素子から open drain 信号が出力されないか、出力差動線の様子をオシロスコープで図 5.2 の通り、確認した。この時、差動線は 40MHz のクロックを出力していた。図 5.1 のような open drain 信号は観測されなかったため、PUDC_B pin を GND に落とす処置は有効だと確認できた。

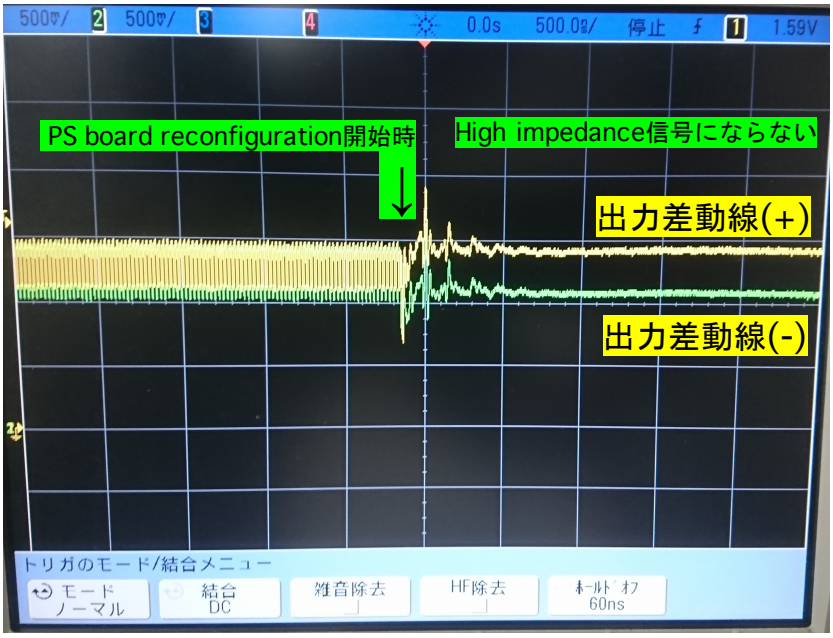


図 5.2 PUDC_B を GND に落とした際の LVDS driver 差動出力線の様子. FPGA の reconfiguration 中、FPGA の User I/O pin から LVDS driver 素子の EN pin に 1 信号が送られ続けているため、差動出力信号は high impedance 状態にはならずに、FPGA の別の User I/O pin から出力されている 1 信号を出力する。

この修正を反映した JATHub における LVDS driver 素子の差動出力線の様子は、今後 JATHub 第 2 試作機の動作試験で確認する。

■(ii.) の対処 slave module 上の reset 線において、LVDS receiver 素子から誤作動で 1μs 程の 0 信号が FPGA or Zynq に出力されても、FPGA or Zynq が 0 信号を検知して reset を行わないように、デバウンサーを導入した。

実装するデバウンサーは MAX6816[21] にした。この素子は最大 80 ms 以上の 0 信号が入力された時のみ、0 信号を出力する機能を有する。そのため、今回の誤作動によって発生した 1μs 程の 0 信号は無視してくれる。JATHub 第 2 試作機の回路図では図 D.33 のように追加した。この修正を反映した結果は、今後 JATHub 第 2 試作機の動作試験で確認する。

デバウンサーを導入した reset 線は、slave JATHub の PL reset を行う PROGB 線と、slave JATHub の Power on Reset を行う PORB 線、そして、slave PS board の PL reset と reconfiguration を行う PROGB 線である。これら 3 線は FPGA or Zynq デバイスの決められた I/O pin に繋がっているため、デバイス外にてデバウンサーによる 0 信号幅の調節が必要となる。

残りの reset 線、slave PS board の GTX transceiver の reset を行う GTXB 線は、PS board の FPGA の User I/O pin に繋がるので、FPGA 内のデジタル回路で信号処理し 0 信号幅の調節が行える。そのため、reset 線のうち GTXB 線のみ、PS board 上の FPGA 外にてデバウンサーを実装しない設計となっている。

5.1.2 LHC クロックと同周波数の水晶発振器の実装

JATHub には 2 つの SFP port が存在する。1 つは光 Ethernet 通信に使用するが、もう 1 つの port は SL との高速光通信に使用する設計へと変更になった。しかし、SL との高速光通信ではシステムクロックに LHC クロックである 40.079MHz を使用しているのだが、FPGA Zynq 内では LHC クロックを生成することができず、外部から FPGA 内蔵の GTX transceiver へ jitter の無い綺麗なシステムクロックとして供給する必要がある。PS board では、SL から光通信ように encode されて供給された LHC クロックを decode し、jitter cleaner 素子を通して GTX transceiver にシステムクロックとして供給する。JATHub では、jitter cleaner 素子を搭載していないため、PS board と同様の実装方法は採用する場合大幅な PCB レイアウト修正が必要になる。そこで、JATHub では簡易的な実装として、GTX transceiver に、水晶発振器から供給される綺麗なクロックを直接供給する方法を採用した。水晶発振器から直接 GTX transceiver にクロックを供給する場合は、GTX の厳しいクロック要求を満たすために、4 倍細かい 160.316MHz のクロックを使用する必要がある。従って、JATHub 第 2 試作機の回路図では、160.316MHz の水晶発振器を GTX の Bank に導入した (図 D.29)。

5.1.3 QSPI flash memory へのアクセスパス追加

JATHub では、独自の BOOT システムのために、遠隔で JATHub の QSPI flash memory の上書きする必要がある。普段 Zynq Linux が稼働していれば、専用のコマンドにより遠隔での上書きが可能である。しかし、放射線により QSPI 内の Boot ファイルが損傷して、Zynq がクラッシュした際に再起動が実施できない場合、外部からの QSPI の上書きが必要になる。当初の設計では、PS board と同様に、隣の master JATHub から JTAG 通信によって QSPI の上書きを行う予定であった。しかし、4.3 節の JTAG 通信試験で明確になった通り、第 1 試作機の設計では上書きが不可能であった。

そこで、JATHub 第 2 試作機では、図 5.3 のように、VME のパスを利用して VME master module によって直接 QSPI の上書きを行う機構を導入した。VME の J3 backplane には、3.3V 電源供給パス以外にも、それぞれの 20 slave slot に対して VME master から独立に操作できる Look-At-Me(LAM) 線[*1]、JTAG4 線 (Input x3, Output x1) があり、この信号線を活用した。また、QSPI flash memory へのアクセスは表 5.1 の通り 2 種類あるが、JTAG 線と親和性が良い SPI アクセスを採用した。これは、VME 通信では回路上に信号線用の buffer を設けなければならないので信号線の方向が 1 つに固定されていること、JTAG4 線と同様の方向であることが理由である。更に、VME からのパスを試験するために、SPI 用 test-14pin を導入した。

通常は PS 内の SPI controller によって、QSPI アクセスにて QSPI flash memory を操作する。そして、Zynq を介さず直接 QSPI flash memory を操作する場合、LAM 線を使用して SPI 信号線のパスを開通させて、buffer を介して VME or test-14pin からの SPI アクセスを使用する。

[*1] CAMAC 規格の用語

表 5.1 (Q)SPI flash memory へのアクセス方法。素子ピンは図 5.4 を参照

アクセス	素子ピン	信号線	QSPI への方向
SPI x1	6	CLock(CL)	Input
	1	Chip Select(CS)	Input
	5	MOSI	Input
	2	MISO	Output
	3	使用しない	
	7	使用しない	
QuadSPI	6	CLock(CL)	Input
(SPI x4)	1	Chip Select(CS)	Input
	5	QSPI 0	In-Out
	2	QSPI 1	In-Out
	3	QSPI 2	In-Out
	7	QSPI 3	In-Out

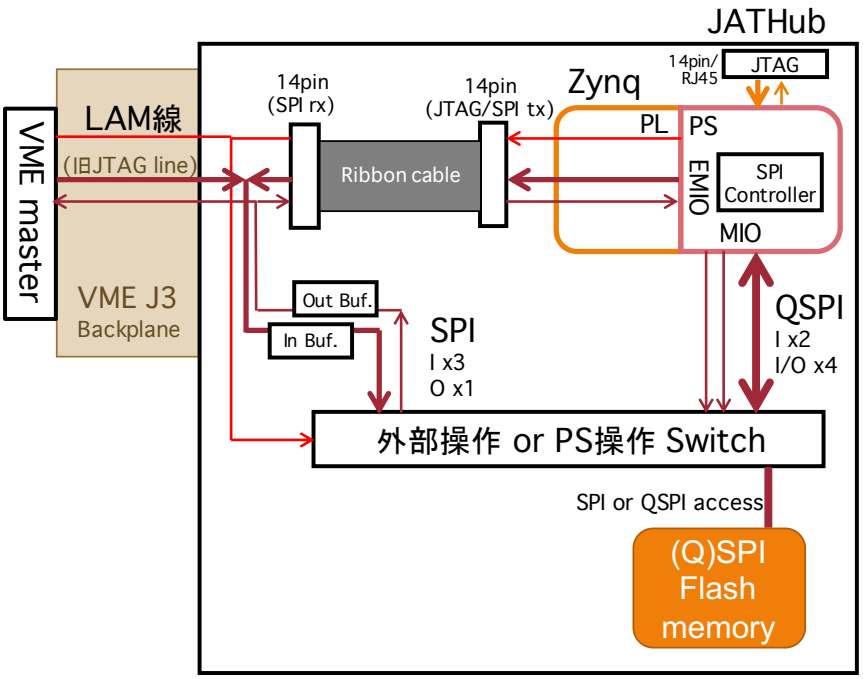


図 5.3 JATHub 第 2 試作機における、(Q)SPI flash memory へのアクセスパス概要。Zynq からの QSPI アクセスと、VME からの SPI アクセスを導入した。

また、QSPI 素子は、自由に SPI or QSPI アクセスの変更設定ができる。デフォルトで SPI アクセスになっており、素子内部の status register を変更すると QSPI アクセスできるようになる。そこで、図 5.4 のように、QSPI 素子へ QSPI アクセス用の信号線が伸びていても、SPI アクセスも行えることを第 1 試作機で試験した。結果、PS が SPI アクセスを行った際、赤枠の “QSPI 2”, “QSPI 3” の信号線は仕様通り使用されてなかった。

保険のため、JATHub 第 2 試作機では QSPI flash memory の容量を 16MB から 64MB に増大させた

(図 5.4)。量産機用の QSPI 素子の選定は、放射線耐性において ATLAS 実験が定める水準を満たす物を使用する。最終的な QSPI 周りの回路図は図 D.32 とした。

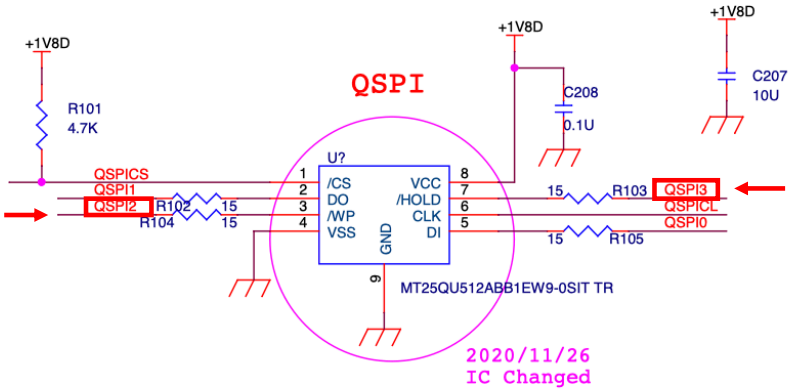


図 5.4 QSPI 素子の回路シンボル。‘QSPI 2’, ‘QSPI 3’ の信号線は SPI アクセス時使用されなかった。

5.1.4 master としての test-14pin を実装

JATHub 第 1 試作機では XVC や SVF player などの JTAG 操作を行える対象 m odule が、JTAG slave port として RJ45 jack を実装している回路 (PS board) に限定されていた。しかし、一般的に Xilinx 社の評価ボードや、Xilinx FPGA を使用する手作り回路には、JTAG slave port として 14 pin connector が実装してある。それら PS board 以外の回路に対しても JATHub が master として JTAG 操作を行えるように、JATHub 第 2 試作機では、JTAG master port として 2 つの 14 pin connector を導入した。

また、JTAG の 4 線は、SPI アクセスの 4 線と I/O 関係において親和性が良い。そのため、この 14pin connectors を SPI アクセス master port としても使用する。 JATHub 第 2 試作機の SPI test-14 pin へ接続して、Zynq から flash memory へ SPI アクセスが行えるように、14 pin に LAM 線 (pin 名 : PGND) も導入した。回路図は図 D.31 と描いた。

14 pin による JTAG 操作、SPI 操作の動作試験は JATHub 第 2 試作機にて行う予定である。また、単純なテストピンとしてデバックにも活用する。

5.1.5 Debug tool の不実装

JATHub 第 1 試作機では、RJ45 Ethernet と UART をデバック用に取り付けていたが、本番では SFP を使用して外部と光 Ethernet 通信を行うので、量産機では該当部品を外す予定である。そのため、JATHub 第 2 試作機でも該当の部品を実装せずに動作試験を行う予定である。これらの部分の PCB レイアウトは変更しない。

5.1.6 Fuse 回路の最適化

JATHub 第 1 試作機では、Fuse の IC を 2 つ並列で繋ぎ、ホールド電流が 12A となるように設計していた。しかし、動作試験によって、本番環境における JATHub の推定最大電流値は 2.5A も満たないことがわかった。そのため、JATHub 第 2 試作機では、Fuse の IC を 1 つにして、ホールド電流を 6A となるように設計した。回路図は図 D.30 と描いた。

5.2 回路図修正とレイアウト修正

5.1 節の修正点を元に回路図を修正した (D.2 節参照)。また、部品配置のデザインも修正し図 5.5 のように描きなおした。JATHub 第 1 試作機と同様に、JATHub 第 2 試作機の PCB 製作と部品実装は外部業者 (有限会社ジー・エヌ・ディー http://www.gn-d.com) に発注した。

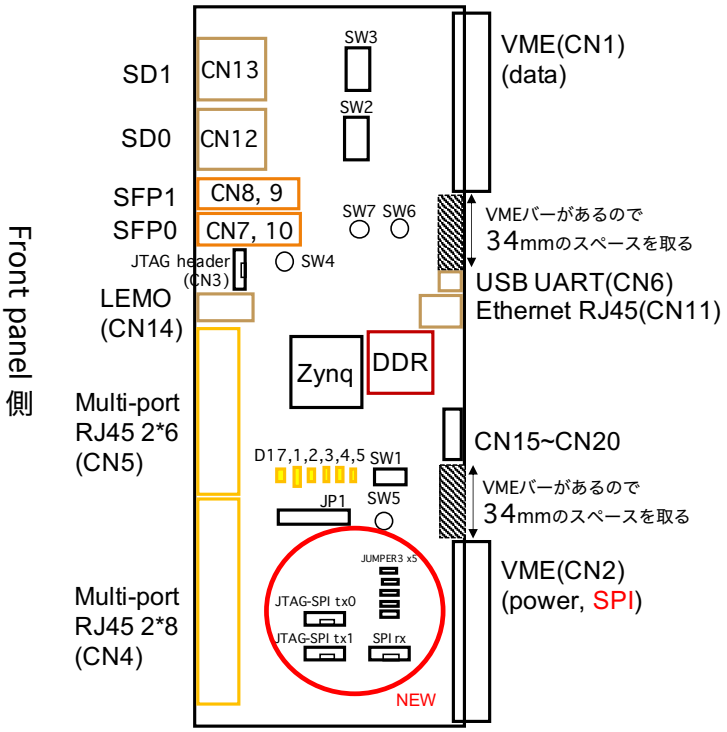


図 5.5 JATHub 第 2 試作機の簡易部品レイアウト．flash memory への SPI アクセスを追加したので、追加するコネクターや Jumper pin の位置を赤枠内に決めた。

第 6 章

結論と今後の展望

6.1　本研究の結論

本研究では、大規模な高エネルギー物理実験を高放射線環境下でも安定して稼働できる制御システムの開発に取り組んだ。その応用として、エネルギーフロンティアでの素粒子物理研究を目的とする 2027 年開始予定の高輝度 LHC-ATLAS 実験に着目し、その将来実験の Level-0 endcap muon trigger system の制御系をデザインし、開発した。

Level-0 endcap muon trigger system からの以下 2 点の要請を満たすように制御システムの仕様を決め、その制御系の中心的な役割を担うハブモジュール - JATHub - を設計した。

i. 大規模な実験室内に設置されている 1500 個程の FPGA を遠隔操作・モニターする
ii. 高エネルギー物理実験室内の高い放射線環境における運転で高い信頼性を持つ

高輝度 LHC-ATLAS 実験の TGC 検出器エレクトロニクスシステムの試運転を行うために、JATHub 試作機を 2 台製作した。また、JATHub の各種機能の開発と実装を行い、JATHub 試作機の動作試験と、JATHub 試作機と PS board 試作機の接続試験を完了させた。そして、動作試験の結果を踏まえて試作機のデザインに必要な修正を施し、JATHub 第 2 試作機のデザインを決定した。(高輝度 LHC-ATLAS 実験における TGC 検出器エレクトロニクスシステムの試運転は現在遂行中である。)

高輝度 LHC-ATLAS 実験で使用される 150 台程の JATHub 量産機の生産では、JATHub 第 2 試作機のデザインをそのまま使用する予定となっている。そのため、本研究によって、JATHub モジュールの最終デザインが完了した。

6.2　JATHub 開発研究の今後の展望

今後 JATHub の開発は、量産機の生産と高輝度 LHC-ATLAS 実験での実用を目指し、表 6.1 のスケジュールで進行していく計画となっている。以下、各計画の詳細を記す。

①：**JATHub 第 2 試作機の製作と動作試験**　完成したデザインに従って JATHub 第 2 試作機を 2020 年度中に製作し、2021 年の夏前までに第 2 試作機の動作試験を完了する。

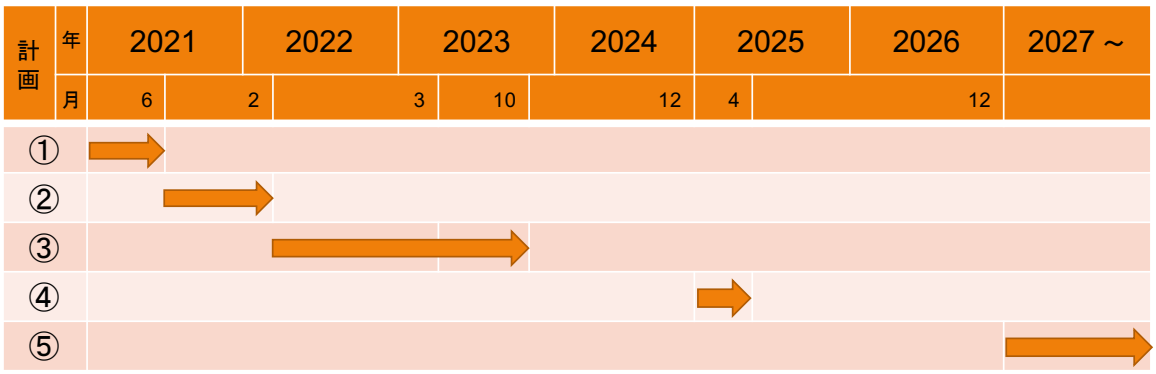
②：**JATHub 量産機設計と初号機の生産、動作試験**　JATHub 量産機のデザインは第 2 試作機のデザインをそのまま使用する。放射線照射試験を行い量産機の部品が ATLAS 実験の要求する放射線耐性を満たすことを確認する。量産機の生産を請け負ってもらう製作所を選定し、初号機を生産する。初号機の動作試験を行い本番環境でも問題なく稼働することを確認する。

③：**JATHub 量産機の生産**　JATHub 量産機を生産する。PS board 量産機の生産と同時並行で進めるため、2023 年 10 月を目標に全 148 台 + 予備台を生産する。

④：**ATLAS 実験室に JATHub 量産機を実装**　2024 年までの Run3 運転が終了し、年が明けたら、TGC 検出器のフロントエンドエレクトロニクスを実験室内に実装していく。ケーブル接続も含めて、2025 年 4 月までの実装完了を目標にしている。

⑤：**高輝度 LHC-ATLAS 実験の開始**　2027 年から高輝度 LHC-ATLAS 実験が始まる。

表 6.1　JATHub モジュールの今後の開発計画



謝辞

本研究を遂行するにあたり、指導教員である石野雅也教授には、毎週の研究報告会議や、学会の発表練習などで、多くの助言をいただき、大変お世話になりました。私の頓珍漢な日本語に対しても、要点が伝わるまで耳を傾けてくださりありがとうございました。また、奥村恭幸准教授には、本開発回路の技術的な助言や、効果的な説明の指導をしていただき、多くのことを学ばせていただきました。CERN 出張では頻繁に質問できる環境にいたため、スピード感を持って研究を進めることができました。お二人のおかげで、若輩の身ながらも、本研究を着実に進めていき、本研究の価値を最大限に引き出した形で修士論文として発表することができました。心より感謝いたします。また、私生活面におきましても、CERN への送迎や生活のアドバイスなど度重なる場面にてご助力いただき充実した研究生活を過ごすことができました。誠にありがとうございました。

本開発回路製作にあたり、高エネルギー加速器研究機構素粒子原子核研究所の佐々木修教授、池野正弘先任技師、庄子正剛准技師、田内一弥先任技師には、多くの技術的な助力をいただき大変感謝しております。おかげさまで、大きな失敗もなく、スケジュール通りに開発を進めることができました。また、ハードウェア関連の様々な専門知識を会得することができました。ありがとうございました。

共同で Phase II アップグレード研究を行っている、TGC グループの皆様にも大変お世話になりました。戸本誠氏、堀井泰之氏には、JATHub の開発進捗の相談に乗ってくださったり、PS board 開発者側とのやり取りの橋渡しをしていただくなど、多くの助力をいただきました。青木雅人氏には、KEK のネットワーク設定や Testbench 作成で多くのお力添えをいただきました。水上さん、日比さん、三野さん、末田くん、綿井くん、山田くん、皆川くんには幾つもの技術的な相談に乗ってもらっただけでなく、休憩時間に他愛ない雑談の相手になってくださったこと、感謝しています。直属の後輩である青木くんには、JATHub の開発を手伝ってもらえてとても感謝しています。直ぐに技術面で相談できる即戦力となってくれたので、今まで一人で悩んでいた問題を二人で悩めるようになり、多くの最適解を導けるようになりました。

ICEPP の他の皆様にも大変お世話になりました。増渕達也助教には主に物理解析の相談に乗っていただきました。齊藤真彦特任助教にも ICEPP シンポジウムなどで物理解析で相談にのっていただきました。齋藤智之助教には発表資料に対する助言をくださりました。秘書の皆様には、出張手続きや経費購入、納品など様々なサポートをしていただきました。他にも ICEPP のスタッフ、学生に度々助けていただきました。皆様、誠にありがとうございました。

ICEPP の同期にも大変お世話になりました。杉崎、CERN 出張が重なったり、下宿部屋も近かったりと何かと一緒に行動していて、見分けがつかないとよく言われましたね。これからもよろしく。島田、同

じスイスでもなかなか会えなかったですね。日帰り旅行 (with 杉崎、米山) 楽しかったです。ビールは程々に。仲間、橋立、成田さん、ICEPP 夏の学校ではしゃぎまくったこととても楽しかったです。山本くん、米山くん、増田くん、たまに会ったときに大部屋で雑談するのがとても心地よかったです。皆様、またばったり会ったら思い出 (or 無駄) 話に華を咲かせましょう。

最後に、これまで支えてくださった両親及び親戚の皆様に感謝いたします。やりたいことができているのはとても幸運だと感じています。これからもよろしくお願いいたします。

Appendix A

TGC 検出器エレクトロニクスシステムの全体像とスケール

制御系も含めた、TGC 検出器フロントエンドエレクトロニクスの全体像を図 A.1 にて示す。この図は片サイドの $\frac{1}{24}$ セクター単位で必要な ASD、PS board、JATHub、SL の台数とケーブル数を示している。

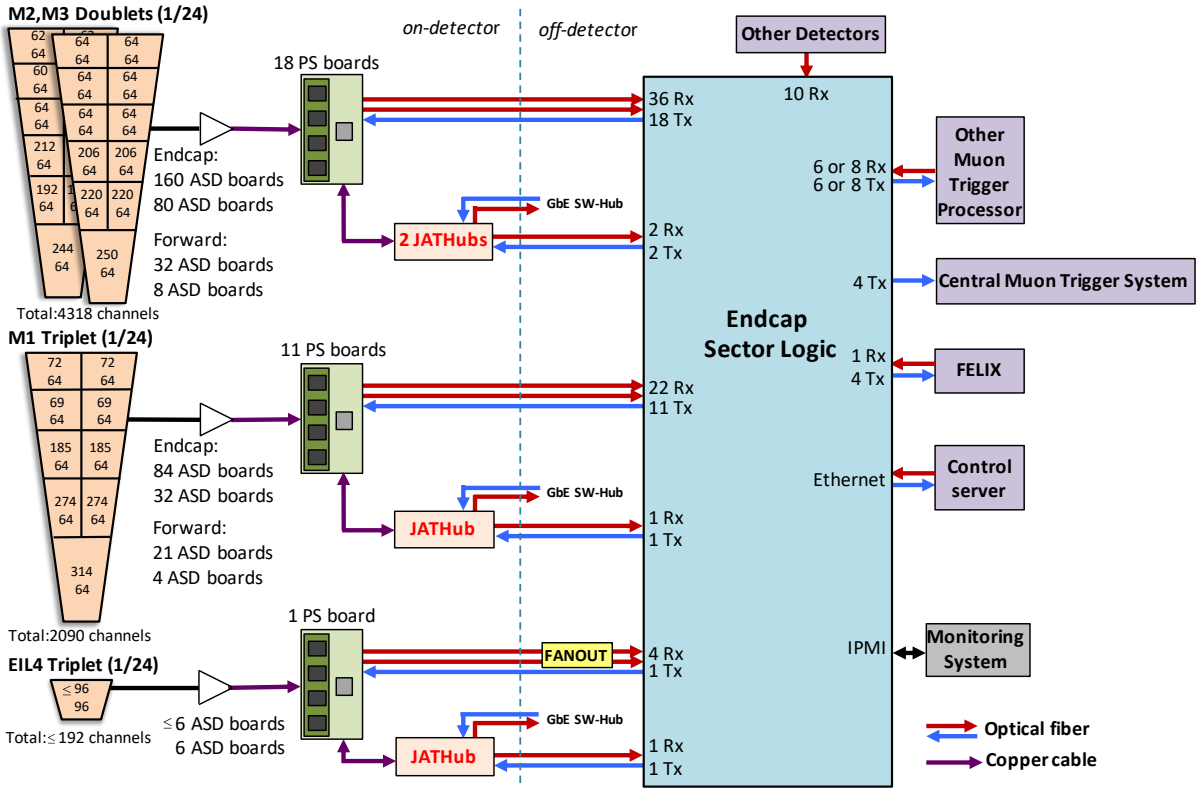


図 A.1　TGC 検出器エレクトロニクスの全体像概念図。1/24 セクター分の台数とケーブル数が記載されている。

Appendix B

自動回復不可能な SEU 事象の発生頻度の概算

以下、“自動回復不可能な SEU 事象”を “UnRecoverable Error (URE)”と略称する。

FPGA Virtex-5 に中性子照射試験を行った実験では、表 B.1 のような結果が得られた [8]。参考文献 [8] では、URE 発生頻度の概算に、SEU と URE の比を使用していた。そのため、2.2.2.1 節でも、実験中の中性子流量と SEU 頻度、URE 頻度の関係を考慮せずに、SEU 事象と URE 事象の比から高輝度 LHC-ATLAS 実験での URE 発生頻度を概算した。(1 台の FPGA の URE:約 900 時間に 1 回、ATLAS 実験室内のどこかの FPGA の URE:約 40 分に 1 回)

本章では、中性子流量と SEU 発生頻度、中性子流量と URE 発生頻度が比例関係にあると仮定して、高輝度 LHC 実験における PS board 上の FPGA の URE 発生頻度を表 B.2 の通り概算した。表 B.2 より、1 台の FPGA に対して URE が約 703 時間に 1 回、ATLAS 実験室内のどこかの FPGA に対して URE が約 30 分に 1 回発生することが予測できる。

表 B.1　FPGA の中性子照射試験の結果 [8]

実験	SEU 頻度 (/s)	URE 頻度 (/s)	中性子流量 (N/s/FPGA)
1st Run	2.51	8.84×10^{-3}	5.31×10^{8}
2nd Run	0.96	2.63×10^{-3}	1.43×10^{8}

表 B.2　PS board 上 FPGA における SEU、URE の発生頻度と表 B.1 の結果に相当する中性子流量

実験	SEU 頻度 (/s)	URE 頻度 (/s)	相当する中性子流量 (N/s/FPGA)
高輝度 LHC-ATLAS 実験	[*97] 9.88×10^{-5}	[*98] 3.95×10^{-7}	(2.47×10^{4})
Run2(2018) 実験	[*99] 2.69×10^{-5}	[*98] 1.08×10^{-7}	(6.73×10^{4})

[*97] 2018 年の ATLAS 実験で行った SEU 発生頻度調査の結果から、概算したもの
[*98] 中性子照射試験の結果から概算したもの
[*99] 2018 年の ATLAS 実験で行った SEU 発生頻度調査の結果

Appendix C

JATHub と Slave module の CAT6 Cabling

TGC 検出器エレクトロニクスの制御系を構築し、JATHub の概念設計を決定した後、現行システムで利用する CAT6 ケーブルをこの制御系で再利用できるか検証を行った。

結果、M1 PS board 用 10 m ケーブル、M3 PS board 用 15 m ケーブルどちらも足りていた。従って、ケーブルラベルを現場で張り替えて使用することで、CAT6 ケーブルの再利用が可能となる。

検証の際に、JATHub と PS board、そして JATHub 同士の配線図 C.1(1/12 セクター単位) を決めた。この配線図に従って ATLAS 実験室への回路実装を行う。

配線図 C.1 の (a) は mini-Rack 内の JATHub の配置を示している。(b) は PS-Pack 内の PS board の配置を示している。それぞれ、欄内に CAT6 接続先のポート名が書かれてある。JATHub 同士は白黒線のように、1HSC crate 内の 6 台をディジーチェーンで繋ぐ。また、JATHub には (a) にて各台に色分けがされており、(b) の PS board は同じ色の JATHub と接続する仕様になっている。

JATHub#	JATHub0		JATHub1		JATHub2		JATHub3		JATHub4		JATHub5	
phi Section	M1-phi0/1		M1-phi2/3		M3-phi0		M3-phi1		M3-phi2		M3-phi3	
Colour												
HSC address												
Destination	PS Ladder M1 phi0		PS Ladder M1 phi2		PS Ladder M3 phi0		PS Ladder M3 phi0		PS Ladder M3 phi2		PS Ladder M3 phi2	
RJ45 Jack	Recovery/TTC	JTAG/I2C	Recovery/TTC	JTAG/I2C	Recovery/TTC	JTAG/I2C	Recovery/TTC	JTAG/I2C	Recovery/TTC	JTAG/I2C	Recovery/TTC	JTAG/I2C
SPP	SPP-m1-TTC	SPP-m1-I2C	SPP-m1-TTC	SPP-m1-I2C	SPP-m1-TTC	SPP-m1-I2C	N/A	N/A	SPP-m1-TTC	SPP-m1-I2C	N/A	N/A
JATrx	JAT5-0-R	JAT5-0-J	JAT5-1-R	JAT5-1-J	JAT3-0-R	JAT3-0-J	JAT4-0-R	JAT4-0-J	JAT5-0-R	JAT5-0-J	JAT5-2-R	JAT5-2-J
JATtx	JAT1-0-R	JAT1-0-J	JAT2-0-R	JAT2-0-J	JAT3-0-R	JAT3-0-J	JAT4-0-R	JAT4-0-J	JAT5-0-R	JAT5-0-J	JAT5-2-R	JAT5-2-J
PSB10	PS6-m1-R	PS6-m1-J	PS6-m1-R	PS6-m1-J	PS10-m1-R	PS10-m1-J	N/A	N/A	PS10-m1-R	PS10-m1-J	N/A	N/A
PSB9	PS5-m2-R	PS5-m2-J	PS5-m2-R	PS5-m2-J	PS9-m2-R	PS9-m2-J	N/A	N/A	PS9-m2-R	PS9-m2-J	N/A	N/A
PSB8	PS5-m1-R	PS5-m1-J	PS5-m1-R	PS5-m1-J	PS9-m1-R	PS9-m1-J	N/A	N/A	PS9-m1-R	PS9-m1-J	N/A	N/A
PSB7	PS4-m2-R	PS4-m2-J	PS4-m2-R	PS4-m2-J	PS8-m2-R	PS8-m2-J	PS8-m1-R	PS8-m1-J	PS8-m2-R	PS8-m2-J	PS8-m1-R	PS8-m1-J
PSB6	PS4-m1-R	PS4-m1-J	PS4-m1-R	PS4-m1-J	PS7-m2-R	PS7-m2-J	PS7-m1-R	PS7-m1-J	PS7-m2-R	PS7-m2-J	PS7-m1-R	PS7-m1-J
PSB5	PS3-m2-R	PS3-m2-J	PS3-m2-R	PS3-m2-J	PS6-m2-R	PS6-m2-J	PS6-m1-R	PS6-m1-J	PS6-m2-R	PS6-m2-J	PS6-m1-R	PS6-m1-J
PSB4	PS3-m1-R	PS3-m1-J	PS3-m1-R	PS3-m1-J	PS5-m2-R	PS5-m2-J	PS5-m1-R	PS5-m1-J	PS5-m2-R	PS5-m2-J	PS5-m1-R	PS5-m1-J
PSB3	PS2-m2-R	PS2-m2-J	PS2-m2-R	PS2-m2-J	PS4-m2-R	PS4-m2-J	PS4-m1-R	PS4-m1-J	PS4-m2-R	PS4-m2-J	PS4-m1-R	PS4-m1-J
PSB2	PS2-m1-R	PS2-m1-J	PS2-m1-R	PS2-m1-J	PS3-m2-R	PS3-m2-J	PS3-m1-R	PS3-m1-J	PS3-m2-R	PS3-m2-J	PS3-m1-R	PS3-m1-J
PSB1	PS1-m2-R	PS1-m2-J	PS1-m2-R	PS1-m2-J	PS2-m2-R	PS2-m2-J	PS2-m1-R	PS2-m1-J	PS2-m2-R	PS2-m2-J	PS2-m1-R	PS2-m1-J
PSB0	PS1-m1-R	PS1-m1-J	PS1-m1-R	PS1-m1-J	N/A	N/A	PS1-m1-R	PS1-m1-J	N/A	N/A	PS1-m1-R	PS1-m1-J

(a) mini-Rack内のJATHubの配置とCAT6ケーブル接続先

	Large R									small R
<PS Pack M1 phi0>										
	PS1	PS2	PS3	PS4	PS5	PS6				
m2	JAT0-1	JAT0-3	JAT0-5	JAT0-7	JAT0-9					
m1	JAT0-0	JAT0-2	JAT0-4	JAT0-6	JAT0-8	JAT0-10				
<PS Pack M1 phi2>										
	PS1	PS2	PS3	PS4	PS5	PS6				
m2	JAT1-1	JAT1-3	JAT1-5	JAT1-7	JAT1-9					
m1	JAT1-0	JAT1-2	JAT1-4	JAT1-6	JAT1-8	JAT1-10				
<PS Pack M3 phi0>										
	PS1	PS2	PS3	PS4	PS5	PS6	PS7	PS8	PS9	PS10
m2		JAT2-1	JAT2-2	JAT2-3	JAT2-4	JAT2-5	JAT2-6	JAT2-7	JAT2-9	
m1	JAT3-0	JAT3-1	JAT3-2	JAT3-3	JAT3-4	JAT3-5	JAT3-6	JAT3-7	JAT2-8	JAT2-10
<PS Pack M3 phi2>										
	PS1	PS2	PS3	PS4	PS5	PS6	PS7	PS8	PS9	PS10
m2		JAT4-1	JAT4-2	JAT4-3	JAT4-4	JAT4-5	JAT4-6	JAT4-7	JAT4-9	
m1	JAT5-0	JAT5-1	JAT5-2	JAT5-3	JAT5-4	JAT5-5	JAT5-6	JAT5-7	JAT4-8	JAT4-10

(b) 各PS-Pack内のPS boardの配置とCAT6ケーブル接続先

図 C.1　JATHub と Slave module の CAT6 Cabling。(a) は mini-Rack 内の JATHub の配置を、(b) は PS-Pack 内の PS board の配置を示しており、欄内に接続先のポート名が書かれてある。JATHub 同士は 6 台をディジーチェーンで繋いでいる。

Appendix D

JATHub 試作機の回路図

D.1　JATHub 第 1 試作機の回路図 (全 26 ページ)

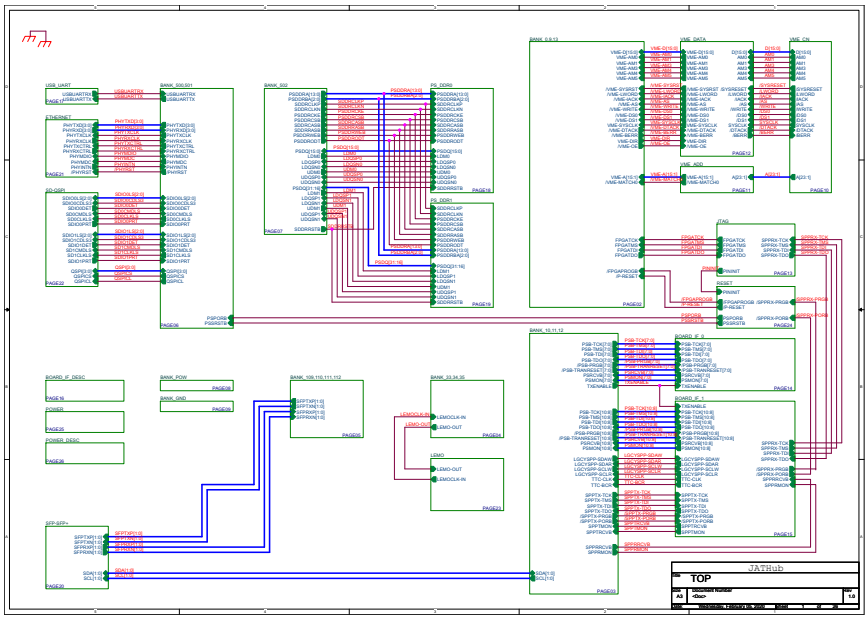


図 D.1　JATHub 第 1 試作機の回路図。TOP ページ。TOP ページの配下にはそれぞれの JATHub の機構を描いた回路ページが存在しており、TOP ページにて配下のページ間の線号線を配線している。そのため、TOP ページはその電子回路の全体像の把握に役に立つ。

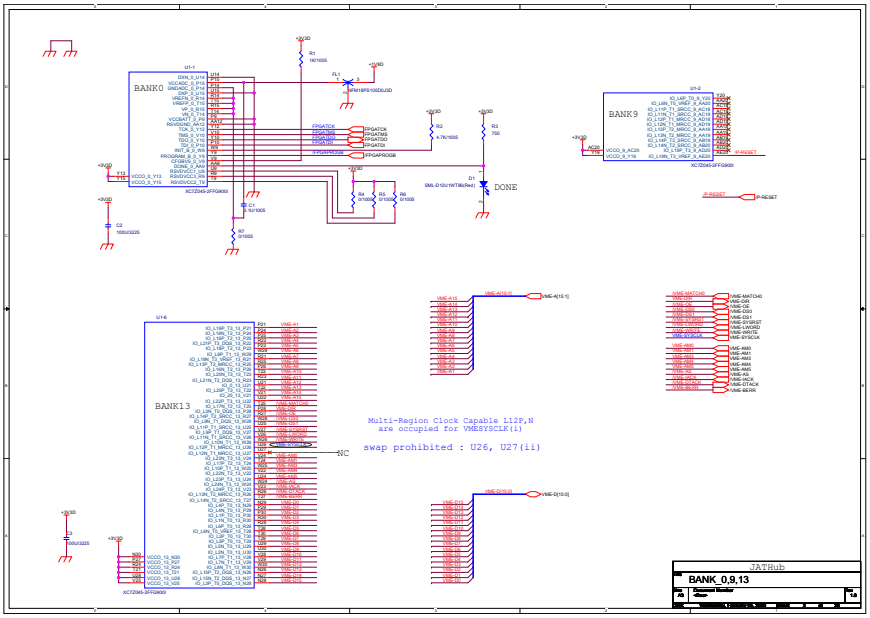


図 D.2　JATHub 第 1 試作機の回路図。BANK 0,9,13 ページ。configurationBank と、PL 用 Bank。Bank 13 は VME 通信に使用。

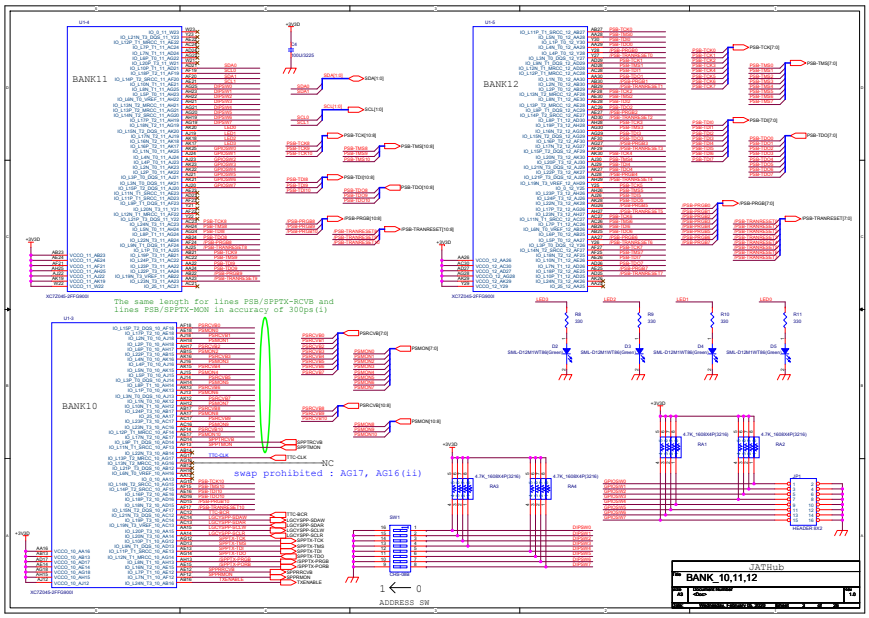


図 D.3　JATHub 第 1 試作機の回路図。BANK 10,11,12 ページ。PL 用 Bank。主に slave module のインターフェイスに使用。

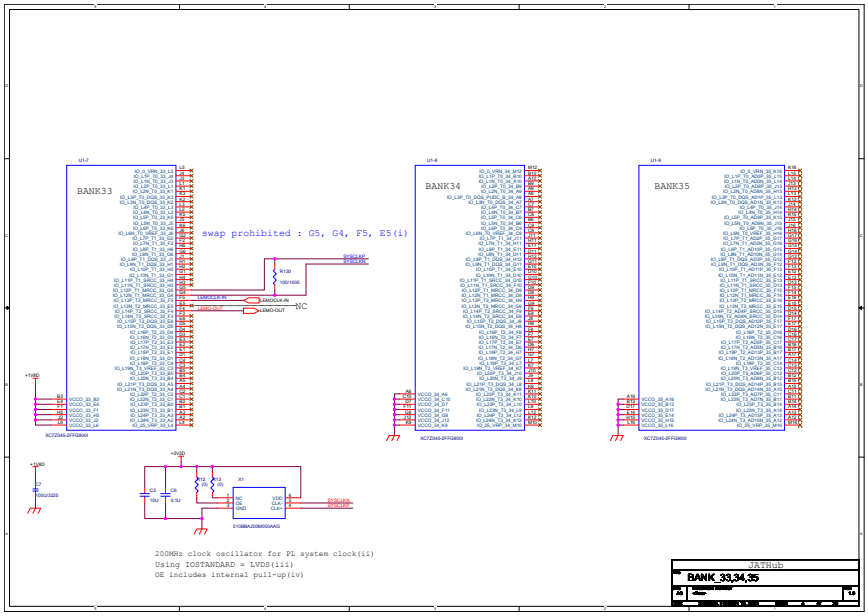


図 D.4 JATHub 第 1 試作機の回路図。BANK33,34,35 ページ。PL 用 Bank。クロック信号線を扱うために使用。200MHz の PL システムクロックは水晶発振器からこの Bank に供給される。

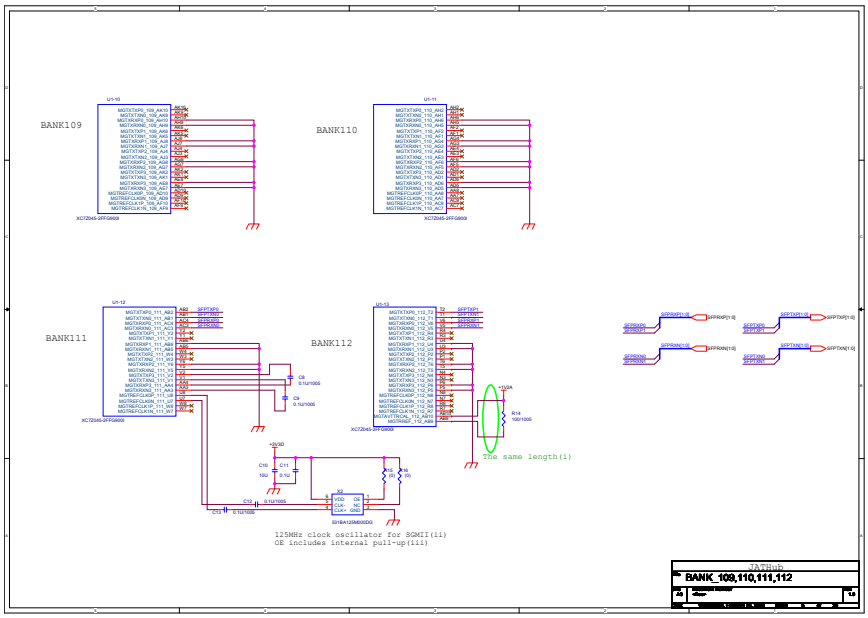
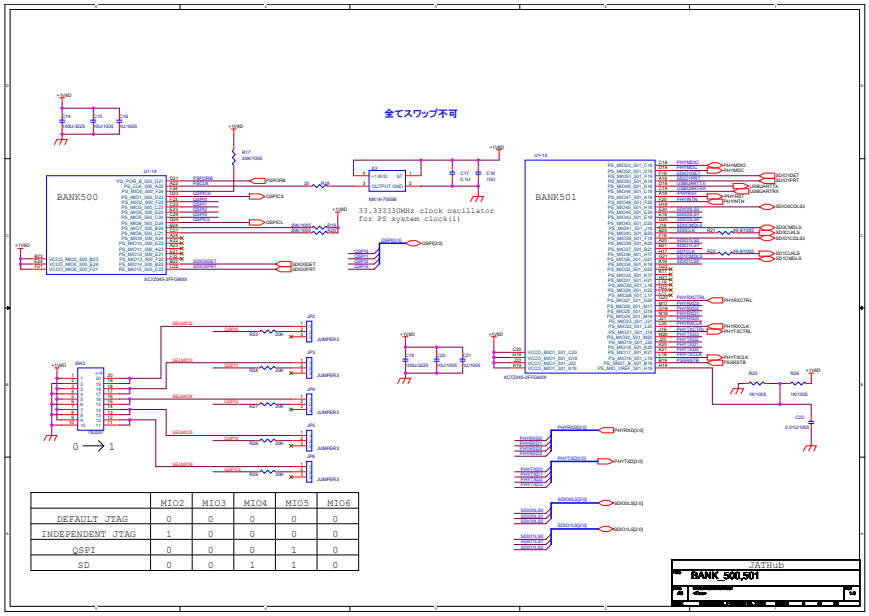


図 D.5 JATHub 第 1 試作機の回路図。BANK109,110,111,112 ページ。GTX transceiver 用の Bank。光 Ethernet 通信用の 125MHz は水晶発振器から Bank111 へ供給される。



	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 D.6　JATHub 第 1 試作機の回路図。BANK500,501 ページ。PS 用 Bank。PS が直接周辺機器を扱うために、MIO ピンが伸びている。UART や RJ45 Ethernet、SD、QSPI、Boot Mode 変更に使用。

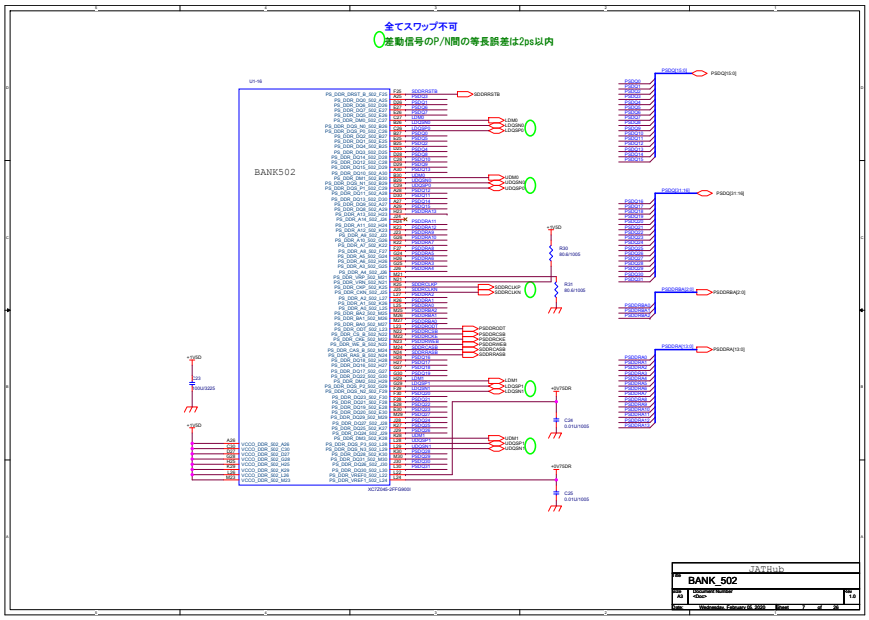


図 D.7　JATHub 第 1 試作機の回路図。BANK502 ページ。PS 用 Bank。PS が DDR を扱うために使用。

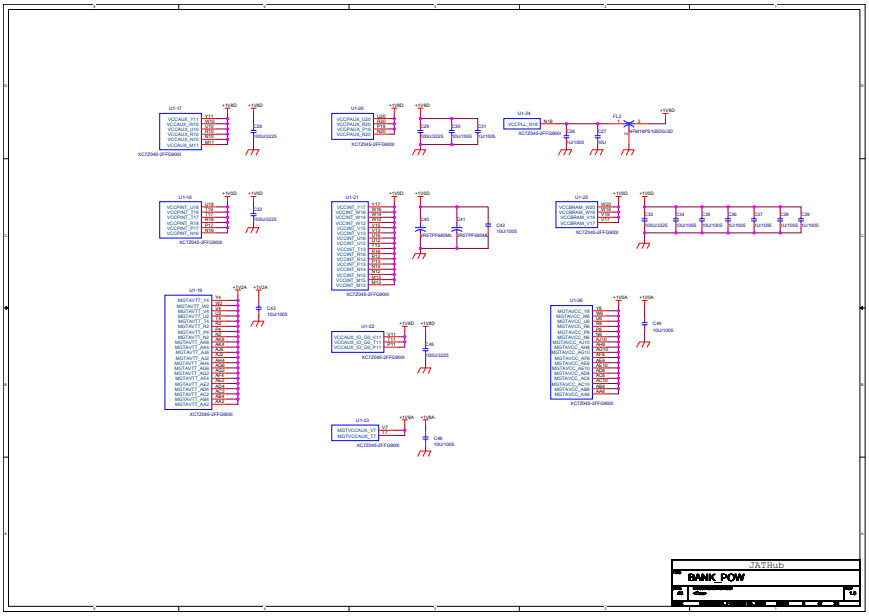


図 D.8 JATHub 第 1 試作機の回路図。BANK POWER ページ。Zynq の電源供給用 Bank。それぞれの電圧は図 D.25 の Linear Regulator にて出力。

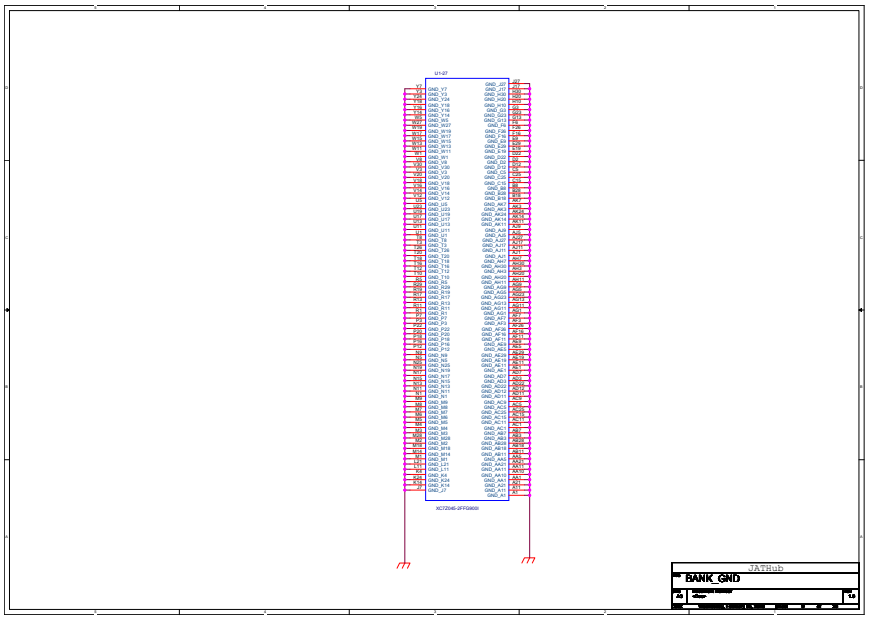


図 D.9 JATHub 第 1 試作機の回路図。BANK GND ページ。Zynq の GND 用 Bank。

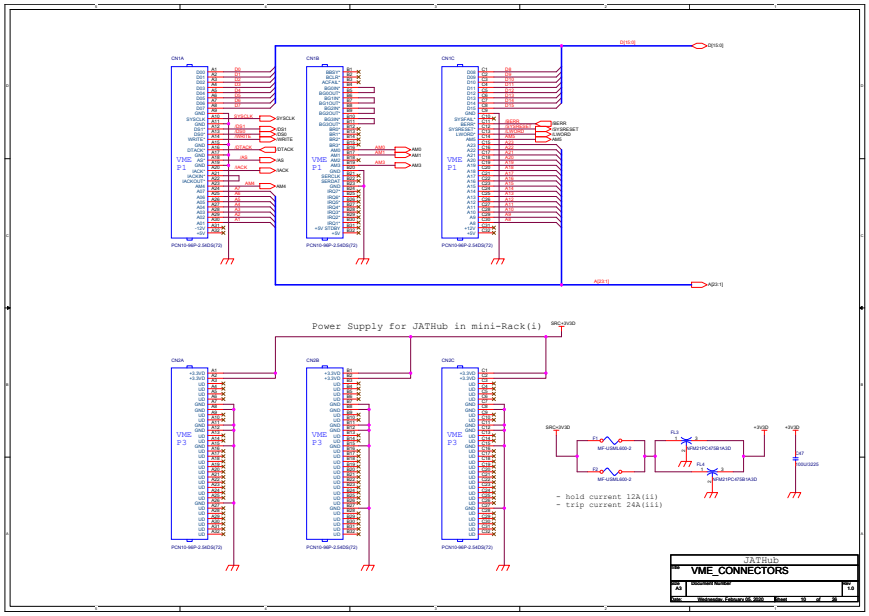


図 D.10　JATHub 第 1 試作機の回路図。VME CONNECTOR ページ。HSC VME backplaneJ1,J3 に接続するための、P1,P3 コネクタ。外部から 3.3V 電源供給を行う上で Fuse を設けて、ホールド電流を 12A にした。

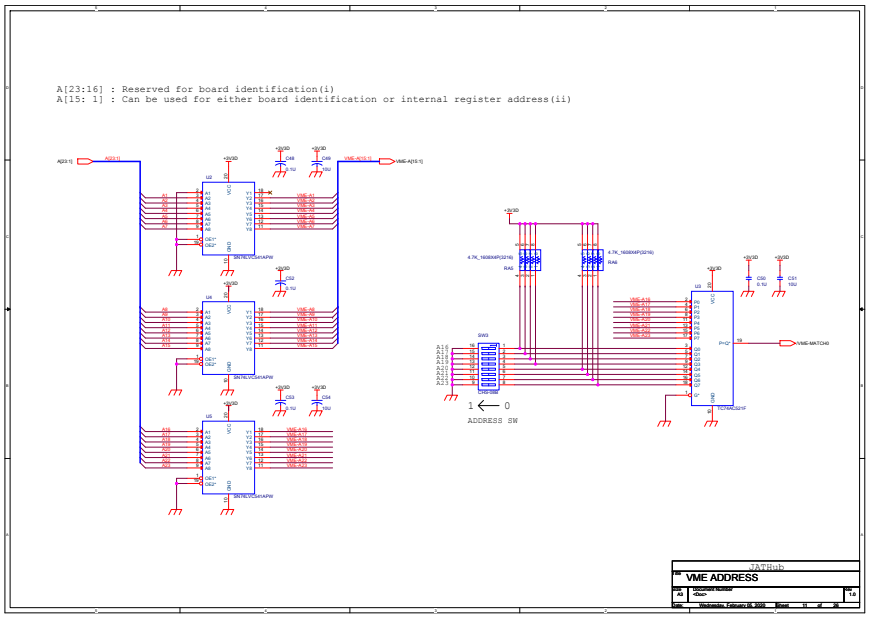


図 D.11　JATHub 第 1 試作機の回路図。VME ADDRESS ページ。VME のアドレス線用の Buffer。PL に繋がる。[23:16] のアドレスは回路上のスイッチで指定できる。[15:1] のアドレスは PL 内で指定できる。

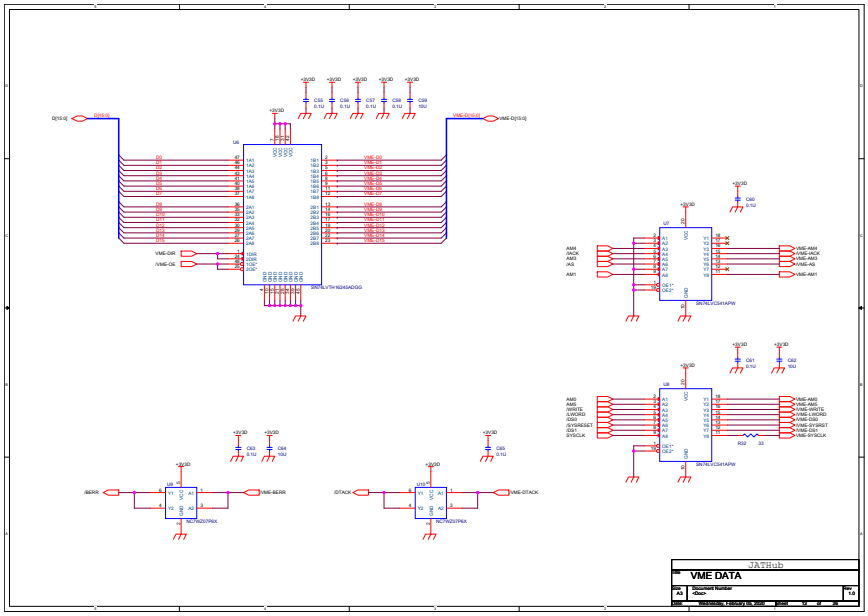


図 D.12 JATHub 第 1 試作機の回路図。VME DATA ページ。VME のデータ線用の Buffer。PL に繋がる VME プロトコルを扱う信号線もここの Buffer を使用。

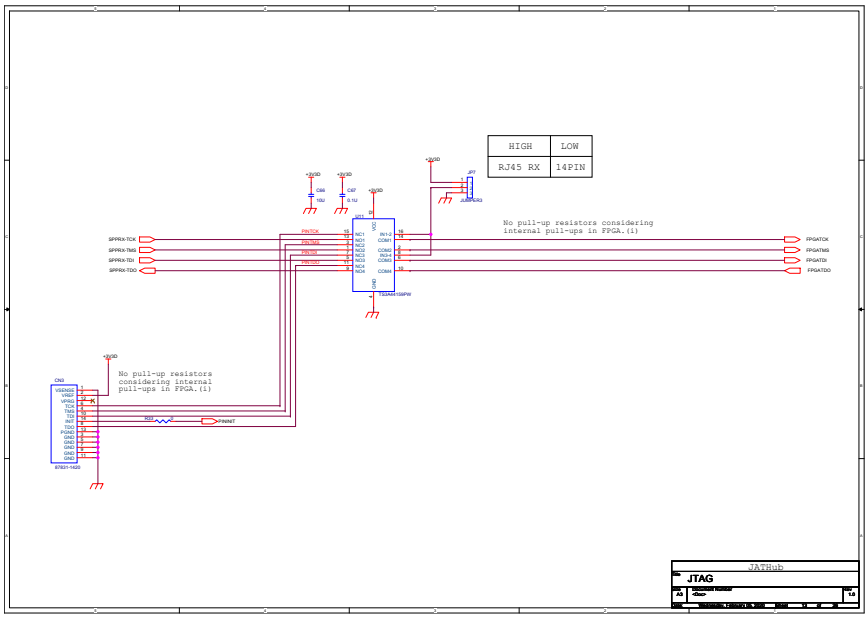


図 D.13 JATHub 第 1 試作機の回路図。JTAG ページ。JATHub が slave として JTAG 通信される際、RJ45 からの信号線と、14pin からの信号線、どちらを開通させるか JUMPER pin で選択できる。PL configuration Bank0 に繋がる

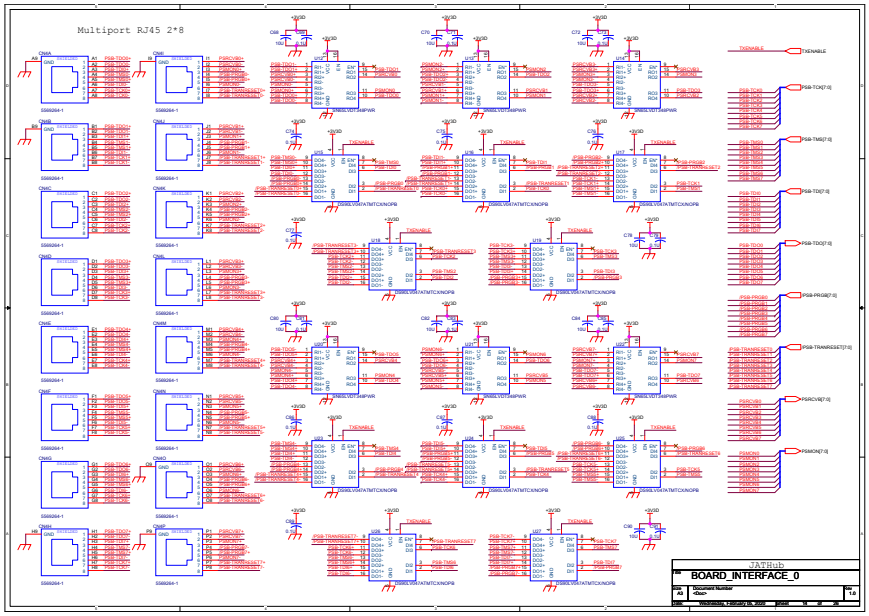


図 D.14 JATHub 第 1 試作機の回路図。BOARD INTERFACE0 ページ。RJ45 multi-jack と LVDS 素子。PSB[7:0] の JTAG パスと Recovery パスをカバーする。

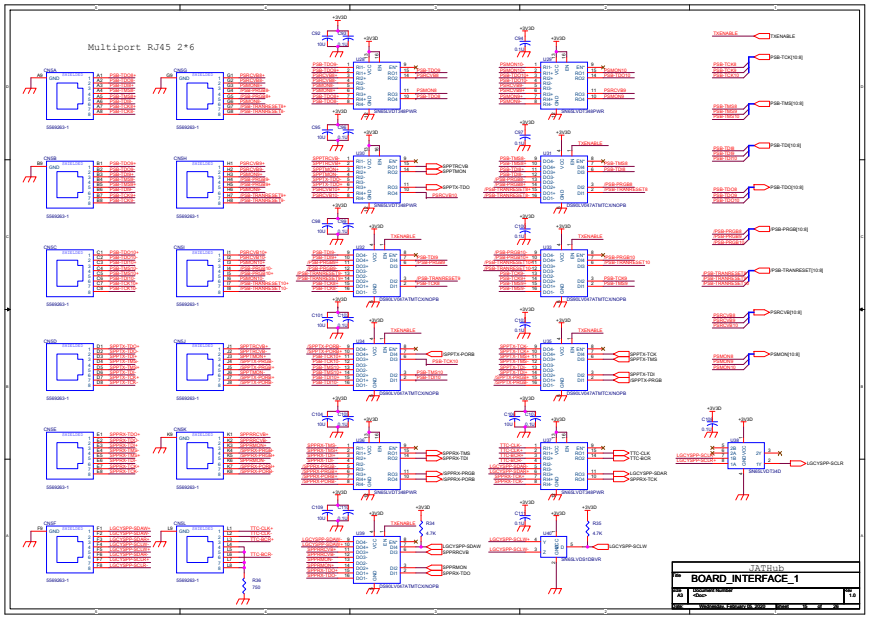
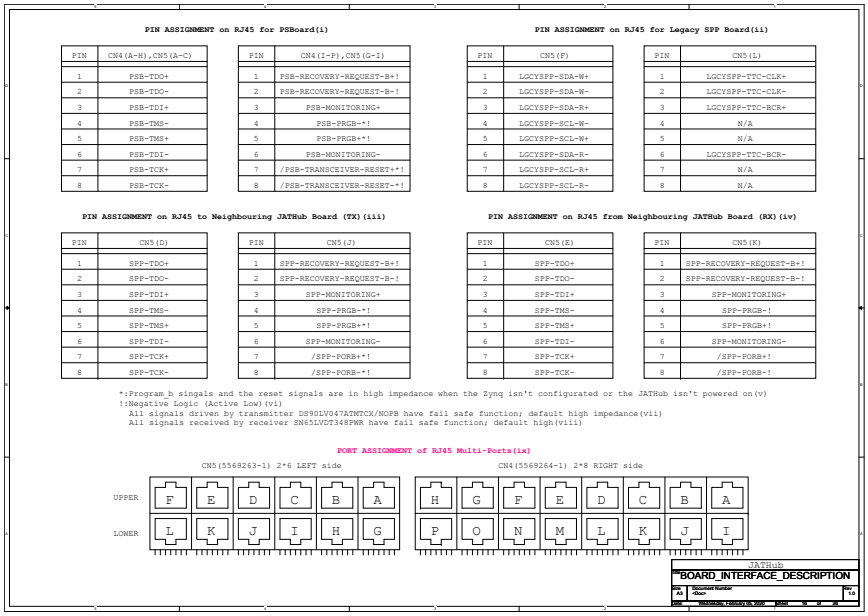


図 D.15 JATHub 第 1 試作機の回路図。BOARD INTERFACE1 ページ。RJ45 multi-jack と LVDS 素子。PSB[10:8] と master JATHub, slave JATHub の JTAG パスと Recovery パスをカバーする。



PIN ASSIGNMENT on RJ45 for PSBoard(i)

PIN	CN4(A-H),CN5(A-C)
1	PSB-TDO+
2	PSB-TDO-
3	PSB-TDI+
4	PSB-TMS-
5	PSB-TMS+
6	PSB-TDI-
7	PSB-TCK+
8	PSB-TCK-

PIN	CN4(I-P),CN5(G-I)
1	PSB-RECOVERY-REQUEST-B+!
2	PSB-RECOVERY-REQUEST-B-!
3	PSB-MONITORING+
4	PSB-PRGB-*!
5	PSB-PRGB+*!
6	PSB-MONITORING-
7	/PSB-TRANSCEIVER-RESET+*!
8	/PSB-TRANSCEIVER-RESET-*!

PIN ASSIGNMENT on RJ45 for Legacy SPP Board(ii)

PIN	CN5(F)
1	LGCYSPP-SDA-W+
2	LGCYSPP-SDA-W-
3	LGCYSPP-SDA-R+
4	LGCYSPP-SCL-W-
5	LGCYSPP-SCL-W+
6	LGCYSPP-SDA-R-
7	LGCYSPP-SCL-R+
8	LGCYSPP-SCL-R-

PIN	CN5(L)
1	LGCYSPP-TTC-CLK+
2	LGCYSPP-TTC-CLK-
3	LGCYSPP-TTC-BCR+
4	N/A
5	N/A
6	LGCYSPP-TTC-BCR-
7	N/A
8	N/A

PIN ASSIGNMENT on RJ45 to Neighbouring JATHub Board (TX)(iii)

PIN	CN5(D)
1	SPP-TDO+
2	SPP-TDO-
3	SPP-TDI+
4	SPP-TMS-
5	SPP-TMS+
6	SPP-TDI-
7	SPP-TCK+
8	SPP-TCK-

PIN	CN5(J)
1	SPP-RECOVERY-REQUEST-B+!
2	SPP-RECOVERY-REQUEST-B-!
3	SPP-MONITORING+
4	SPP-PRGB-*!
5	SPP-PRGB+*!
6	SPP-MONITORING-
7	/SPP-PORB+*!
8	/SPP-PORB-*!

PIN ASSIGNMENT on RJ45 from Neighbouring JATHub Board (RX)(iv)

PIN	CN5(E)
1	SPP-TDO+
2	SPP-TDO-
3	SPP-TDI+
4	SPP-TMS-
5	SPP-TMS+
6	SPP-TDI-
7	SPP-TCK+
8	SPP-TCK-

PIN	CN5(K)
1	SPP-RECOVERY-REQUEST-B+!
2	SPP-RECOVERY-REQUEST-B-!
3	SPP-MONITORING+
4	SPP-PRGB-!
5	SPP-PRGB+!
6	SPP-MONITORING-
7	/SPP-PORB+!
8	/SPP-PORB-!

*:Program_b singals and the reset signals are in high impedance when the Zynq isn't configurated or the JATHub isn't powered on(v)
!:Negative Logic (Active Low)(vi)
All signals driven by transmitter DS90LV047ATMTCX/NOPB have fail safe function; default high impedance(vii)
All signals received by receiver SN65LVDT348PWR have fail safe function; default high(viii)

PORT ASSIGNMENT of RJ45 Multi-Ports(ix)

CN5(5569263-1) 2*6 LEFT side　　CN4(5569264-1) 2*8 RIGHT side

UPPER　F E D C B A　H G F E D C B A

LOWER　L K J I H G　P O N M L K J I

JATHub
BOARD_INTERFACE_DESCRIPTION

図 D.16　JATHub 第 1 試作機の回路図。BOARD INTERFACE DESCRIPTION ページ。RJ45 multi-jack の配線説明。

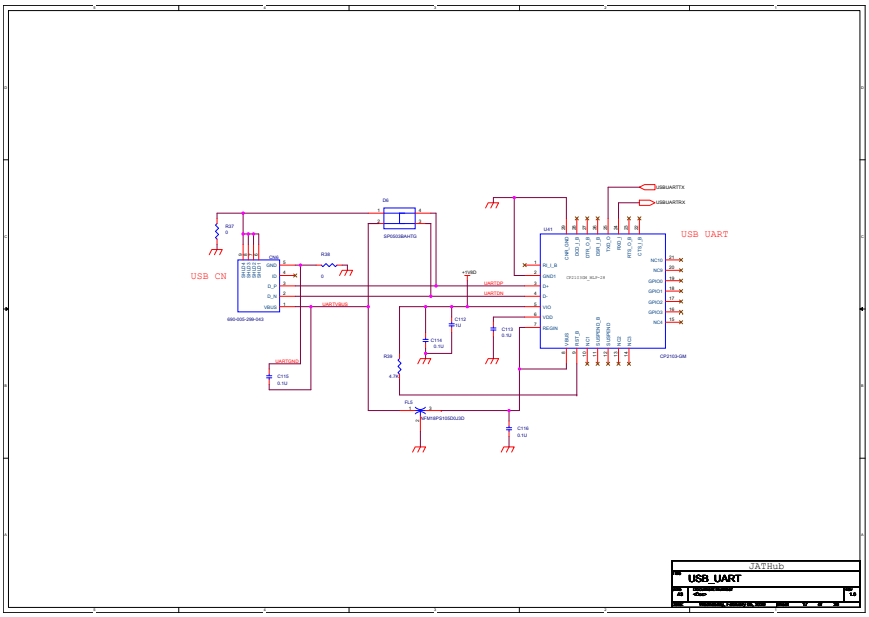


図 D.17　JATHub 第 1 試作機の回路図。USB-UART ページ。UART 通信用素子とインターフェイス。PS に繋がる。Debug 用のため、第 2 試作機では実装しない。

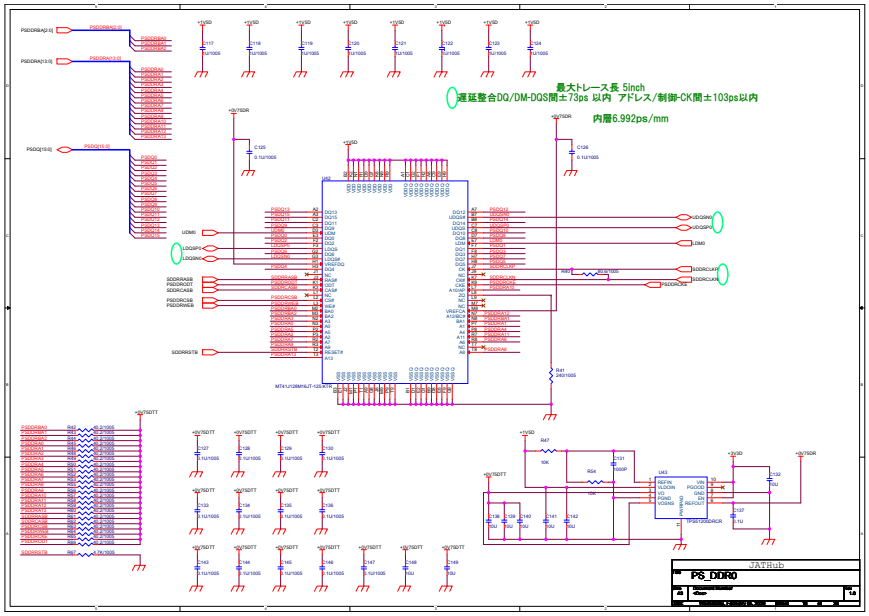


図 D.18　JATHub 第 1 試作機の回路図。PS DDR0 ページ。PS 用の DDR[15:0]。

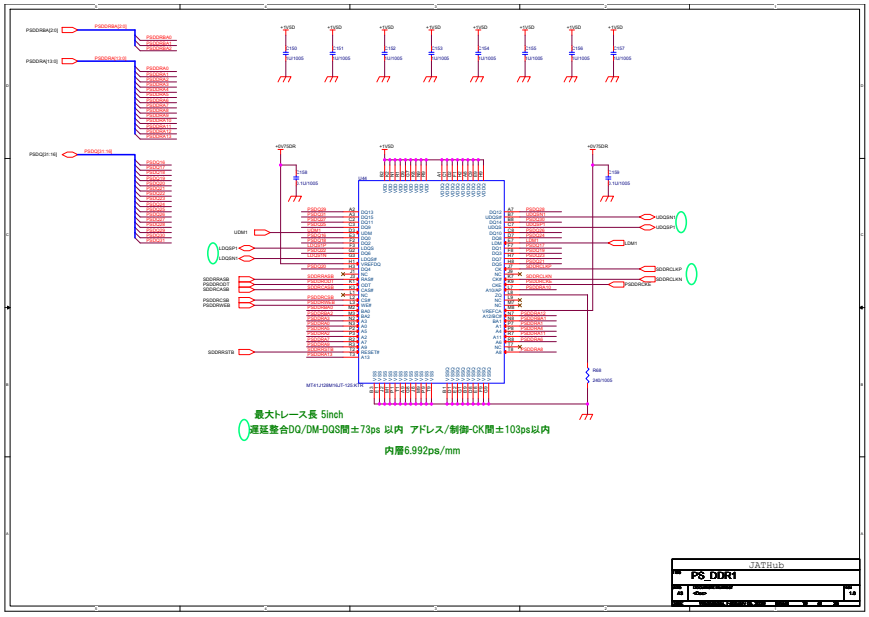


図 D.19　JATHub 第 1 試作機の回路図。PS DDR1 ページ。PS 用の DDR[31:16]

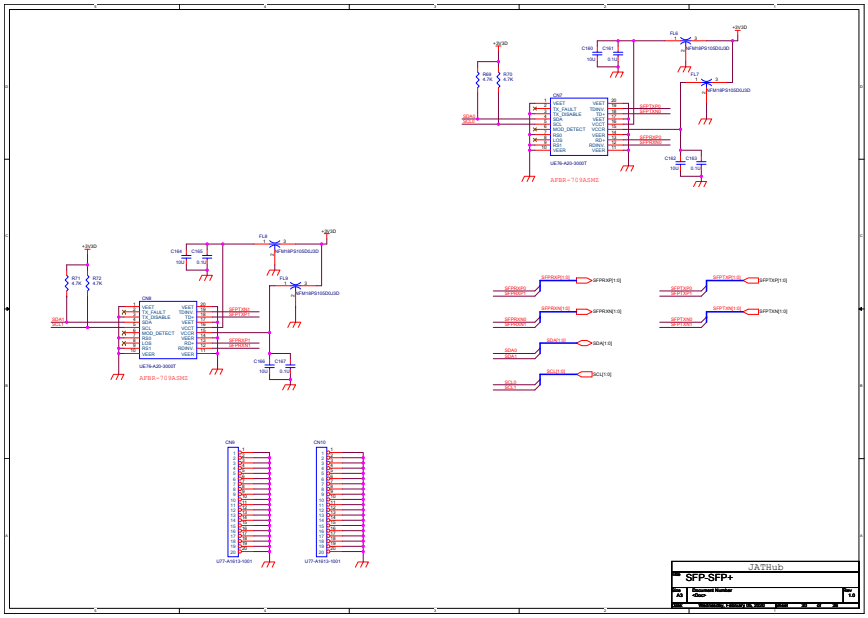


図 D.20　JATHub 第 1 試作機の回路図。SFP ページ。SFP 用素子と SFP connector。PL の GTX transceiver に繋がる。I2C 信号線は PL の user I/O pin に繋がる。

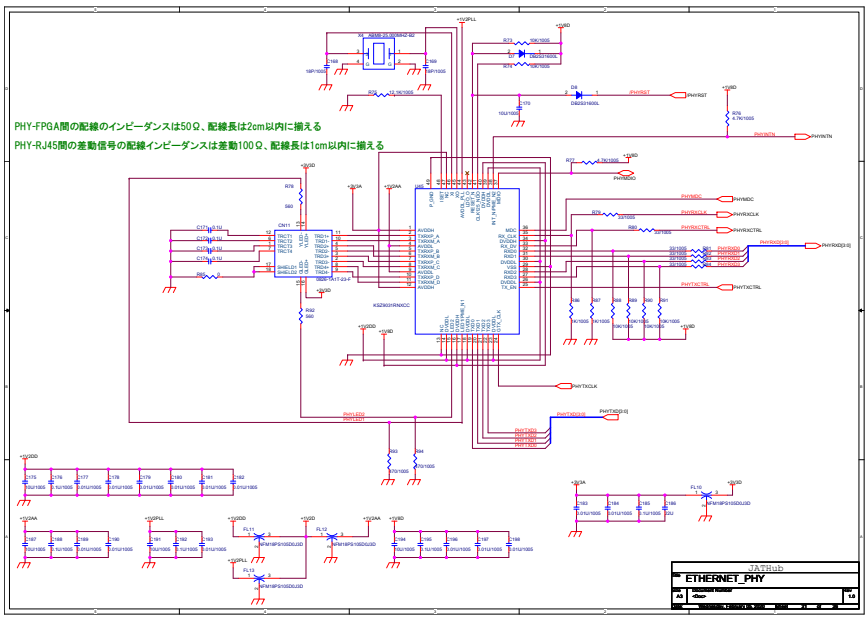


図 D.21　JATHub 第 1 試作機の回路図。ETHERNET PHY ページ。Ethernet 用 RJ45 と PHY 素子。PS に繋がる。Debug 用なので、第 2 試作機では実装しない。

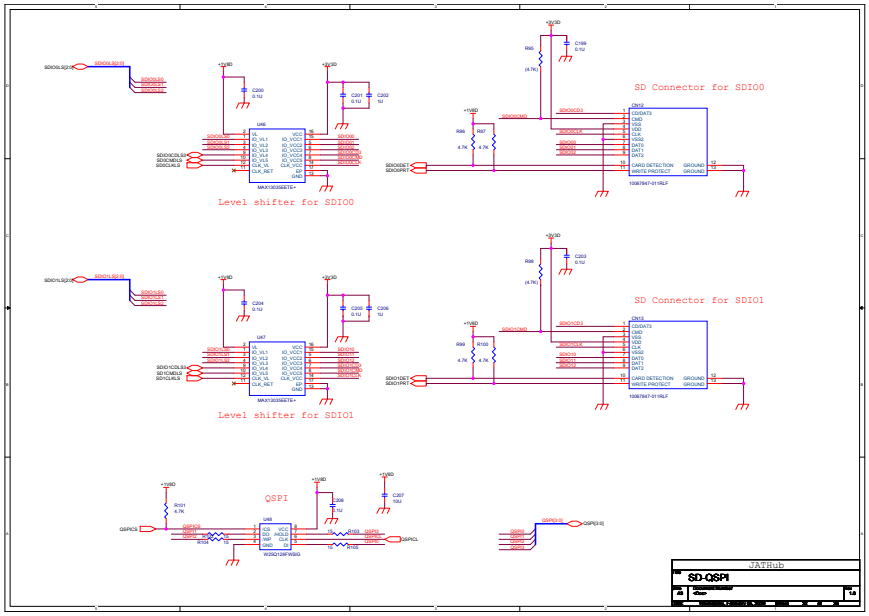


図 D.22　JATHub 第 1 試作機の回路図。SD QSPI ページ。flash memory。PS に繋がる。

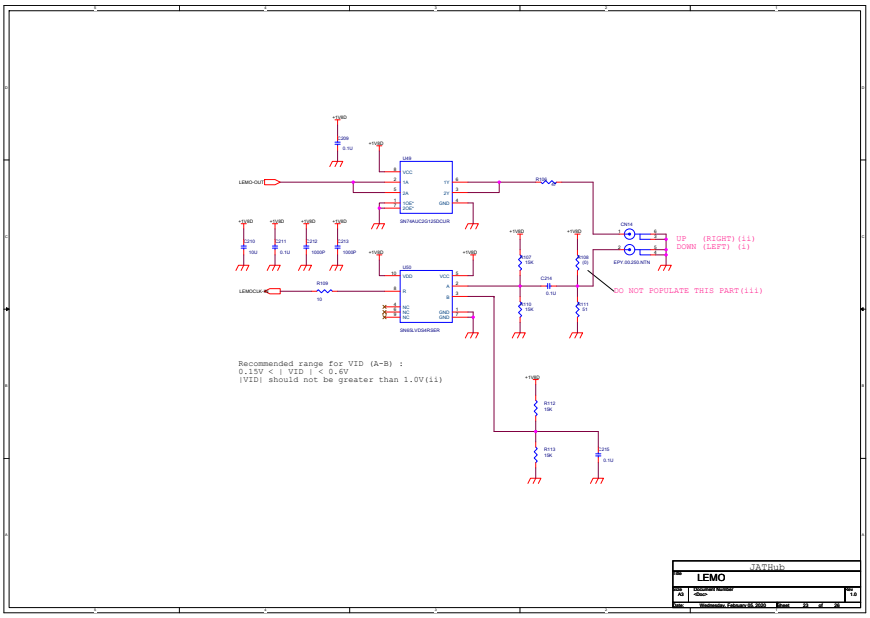


図 D.23　JATHub 第 1 試作機の回路図。LEMO ページ。LEMO の input, output 用の Buffer を設けている。PL に繋がる。clock の duty cycle は 50 %を想定している。

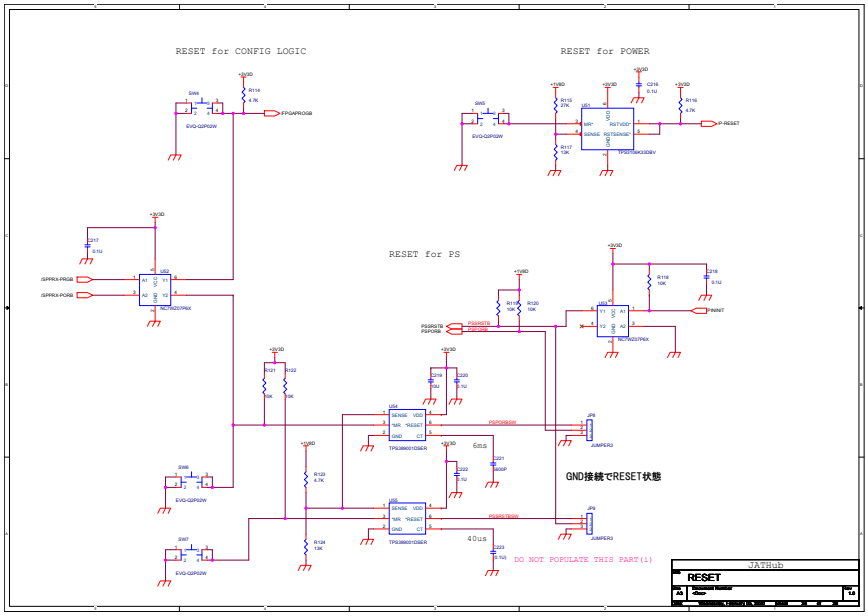


図 D.24　JATHub 第 1 試作機の回路図。RESET ページ。Reset 信号線を扱う。PROGB は PL configuration Bank へ、PORB は PS Bank へ繋がり、0 信号が検知されるとリセットがかかる。リセット用ボタンも実装している。

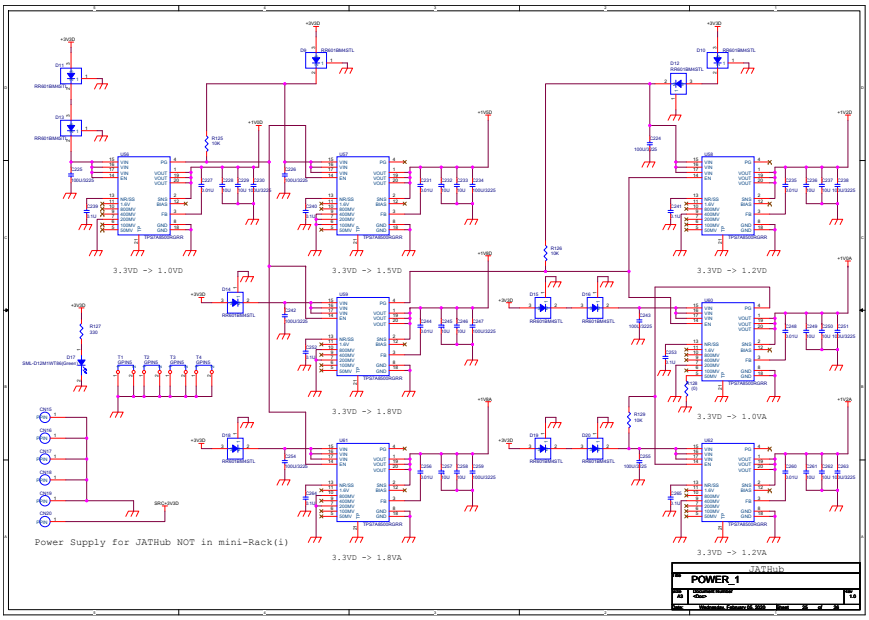


図 D.25　JATHub 第 1 試作機の回路図。電源ページ。Linear Regulator を使用して、電源を管理している。詳細は 3.2.4 節。

最終ページは図 3.13 の電源周りの説明ページである。詳細は 3.2.4 節にて説明している。

D.2 JATHub 第 2 試作機の回路図 (変更した主なページ)

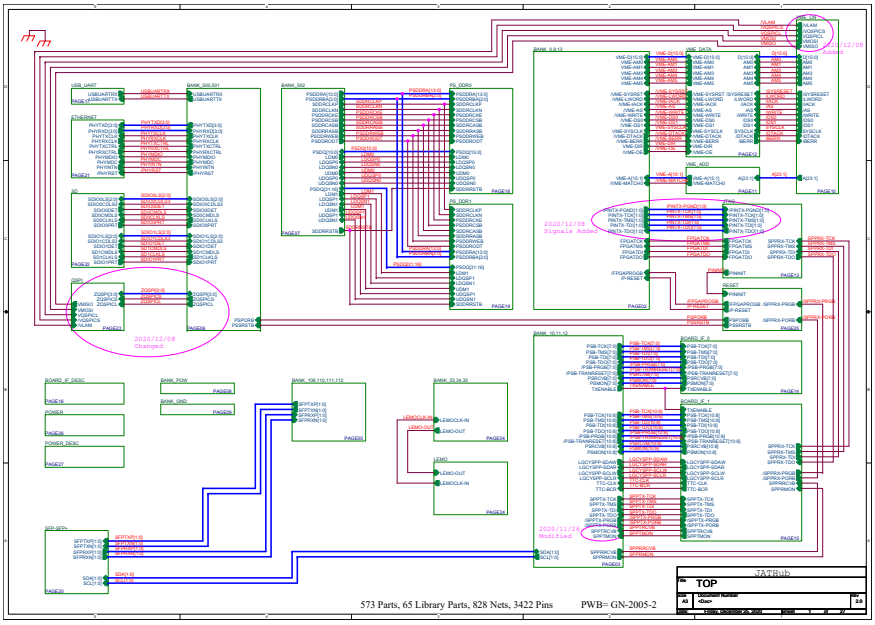


図 D.26 JATHub 第 2 試作機の回路図。TOP ページ。QSPI flash memory への SPI パスを導入したため、QSPI 用の回路図ページを追加した。

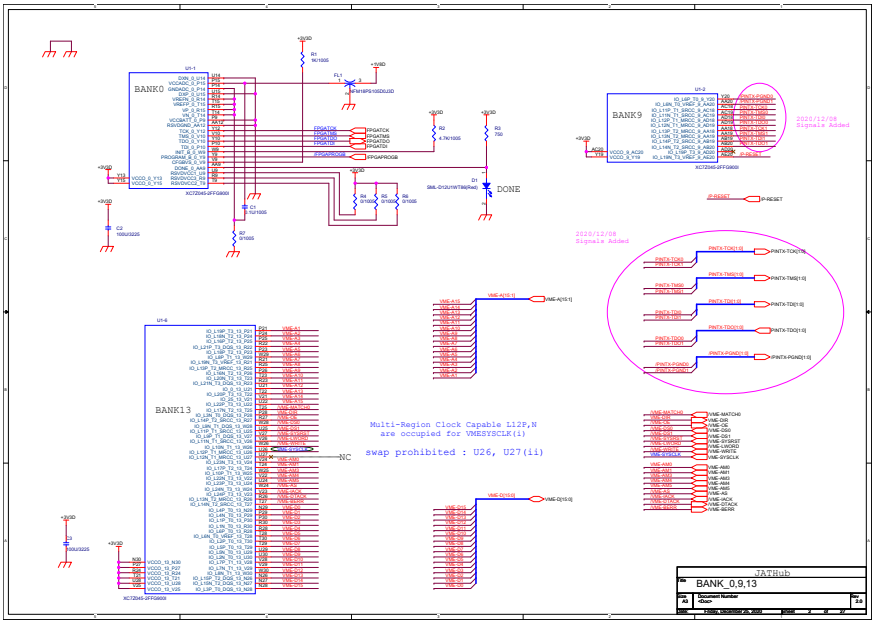


図 D.27 JATHub 第 2 試作機の回路図。BANK 0,9,13 ページ。2 つの test-14pin をドライブするために、Bank9 を使用。

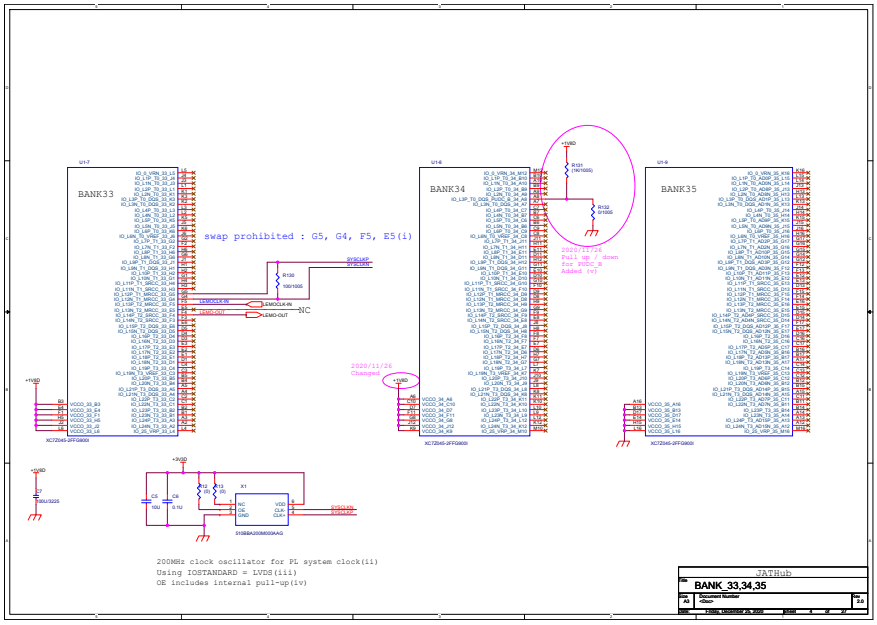


図 D.28 JATHub 第 2 試作機の回路図。BANK 33,34,35 ページ。PUDC_B を GND に落とした。Zynq は起動後の configuration 中に User I/O pin から 1 信号を出力するようになる。

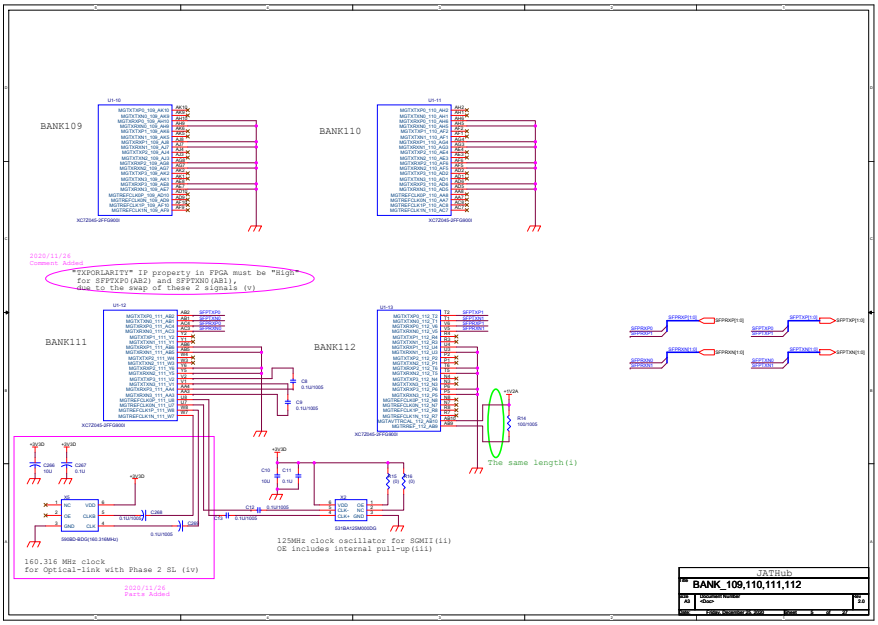


図 D.29 JATHub 第 2 試作機の回路図。BANK 109,110,111,112 ページ。Sector Logic との通信のために水晶発振器から参照クロック 160.316MHz を GTX transceiver へ供給する。

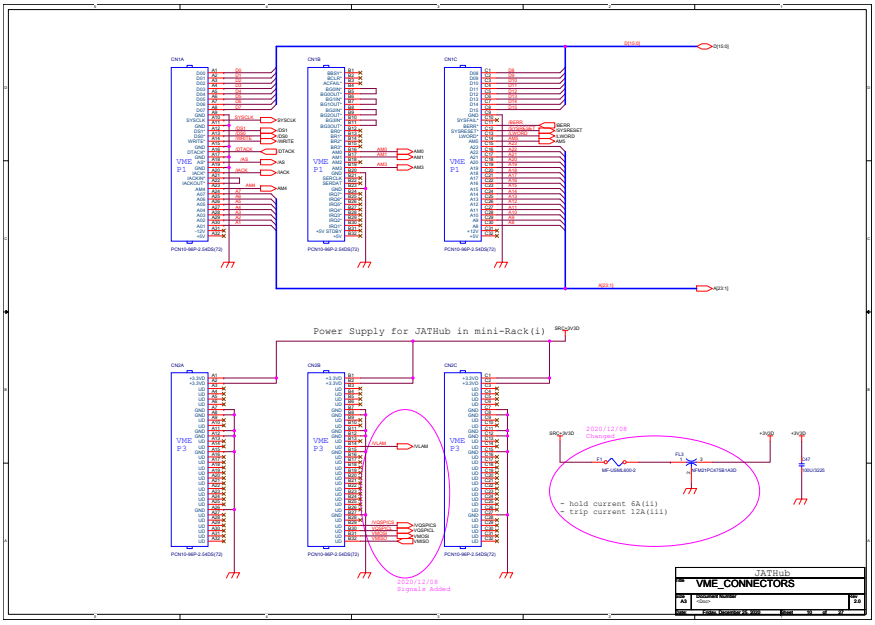


図 D.30　JATHub 第 2 試作機の回路図。VME CONNECTOR ページ。VME master からの SPI アクセスを導入するため、Backplane J3 の LAM 線と JTAG4 線を活用した。また、3.3V 電源供給用の Fuse はホールド電流を 6A に下げた。

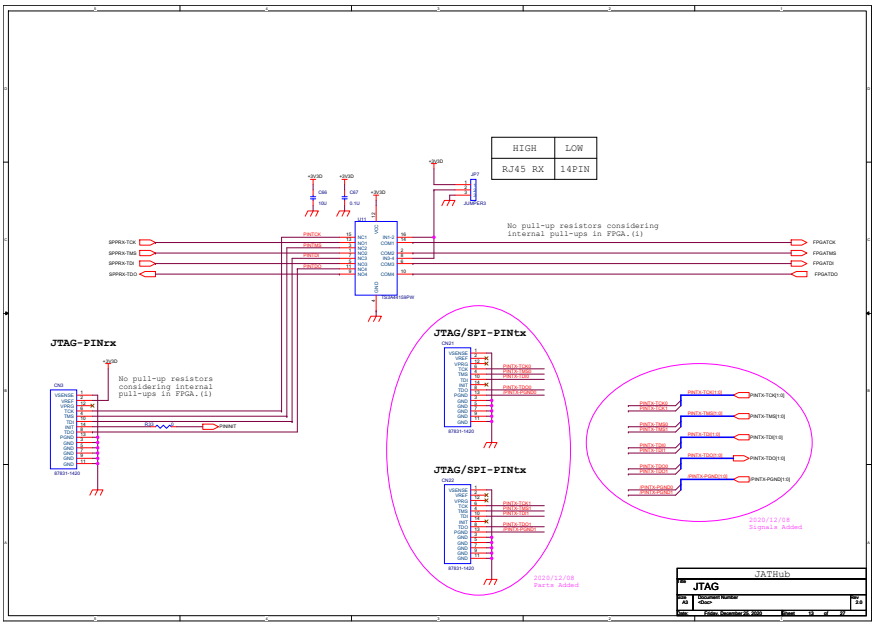


図 D.31　JATHub 第 2 試作機の回路図。JTAG ページ。JATHub が master として、JTAG 線 orSPI 線をドライブするための test-14pin を導入した。

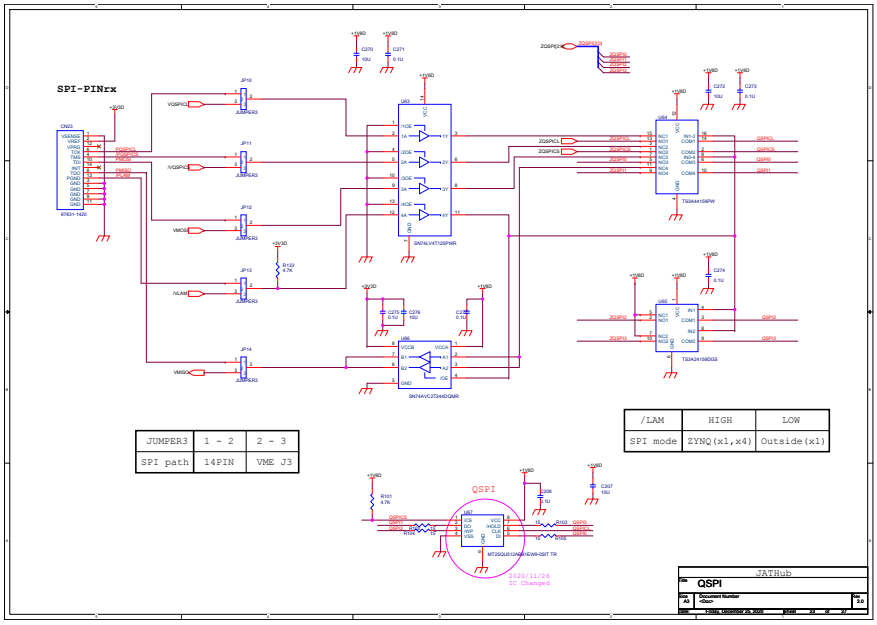


図 D.32 JATHub 第 2 試作機の回路図。QSPI ページ。VME から QSPI flash memory へ SPI アクセスするためのパスを導入した。また、SPI アクセスを試験するための test-14pin も設けた。

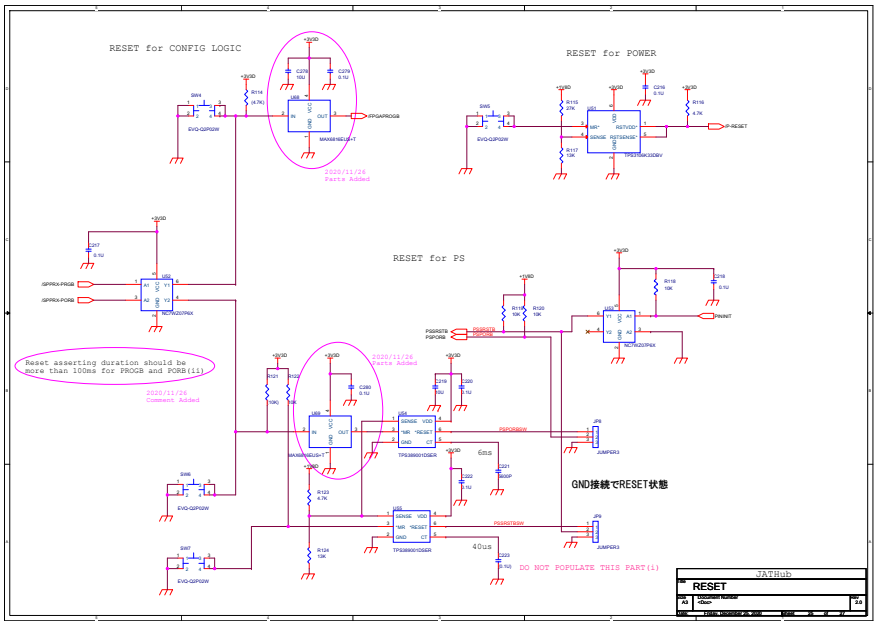


図 D.33 JATHub 第 2 試作機の回路図。RESET ページ。デバウンサー (MAX6816) を導入した。回路上にこの IC を実装し、信号幅 80ms 以上の 0 信号のみ Zynq or FPGA の reset 線に流す仕掛けにした。

Appendix E

JATHub 試作機の 14 層構造表

委託業者と協議した結果、第 1 試作機でも、第 2 試作機でも 14 層構造の PCB となった。以下、表 E.1 にて階層構成表を示す。各層にコアとプリプレグを挟んでいる。コアは基板を硬化させて両面に銅箔を張ったもの、プリプレグは基板を半硬化させて層間の絶縁としてしようするものである。

表 E.1: JATHub 第 1 試作機の PCB 階層構成。 14 層構造。特性インピーダンス (Zo) : 40 Ω or 50 Ω、差動インピーダンス (Zdiff) : 80 Ω or 100 Ω。表面には SILK で文字が書かれている。

層構成	厚み (μ m)	層説明
L1	46	Signal 層
プリプレグ	86	
L2	18	Ground 層
コア	100	
L3	18	Signal 層
プリプレグ	103	
L4	18	Ground 層
コア	100	
L5	18	Signal 層
プリプレグ	100	
L6	35	Ground 層
コア	100	
L7	35	電源層
プリプレグ	104	
L8	35	電源層
コア	100	
L9	35	Ground 層
プリプレグ	100	

表は次ページに続く

前ページの続き

層構成	厚み (μ m)	層説明
L10	18	Signal 層
コア	100	
L11	18	Ground 層
プリプレグ	103	
L12	18	Signal 層
コア	100	
L13	18	Ground 層
プリプレグ	86	
L14	46	Signal 層

表終わり

Appendix F

Zynq 組み込みデザインの Diagram

JATHub 第 1 試作機の Zynq には図 F.1 のような組み込みデザイン (PS と PL の全体的なデザイン) を実装した。実装した機能は以下の通りであり、詳細は 4 章を参照。

- 光 Ethernet 通信。
- slave JATHub として、configuration Bank 0 に触れられて、JTAG 通信、Power-on-Reset の再起動 (PORB)、PL reset(PROGB) を行ってもらう機能。
- XVC を用いて、自分の configuration Bank 0 に触れて、自分の PL をデバックする Debug Bridge 機能。
- master JATHub として、XVC、SVF player を用いて最大 11 台の PS boards と slave JATHub を JTAG 通信する機能。
- SEM で PL 内の SEU 事象を監視修復し、自己修復不可能な場合 slave JATHub として救難信号を発信する機能。
- 最大 11 台の PS boards から送られる LHC クロックをモニターする機能。
- master JATHub として、最大 11 台の PS boards と slave JATHub からの救難信号を受信し、Recovery 手続きを行う機能。
- slave module として、VME 通信により JATHub の register を操作してもらう機能。

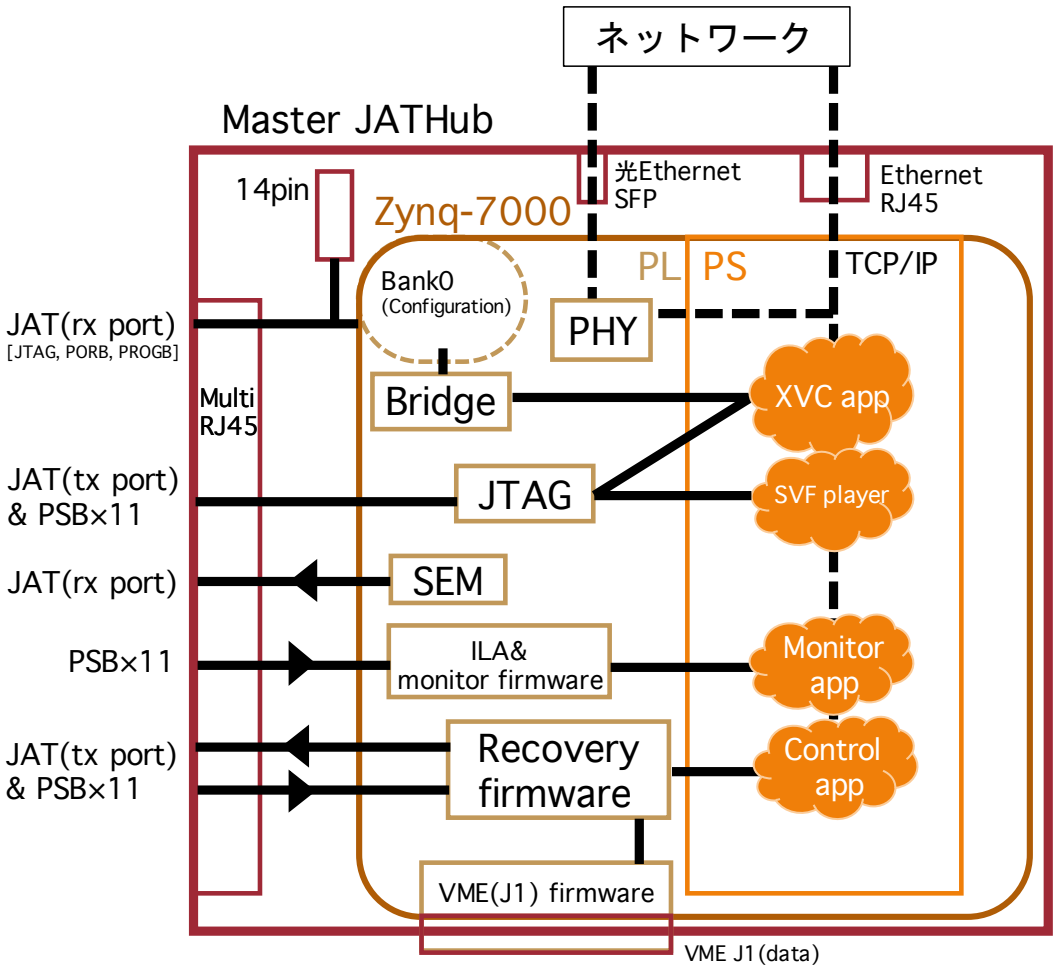


図 F.1 Zynq に実装した組み込みデザインの全体像

Appendix G

PHY chip 不要な Zynq の Ethernet 通信

ZC706 評価ボードで光 Ethernet 通信のデモンストレーションを行っていた際、SFP connector に SFP-RJ45 adapter を設置して、図 G.1 のように Zynq による Ethernet 通信も行えた。HostPC と評価ボードをローカルネットで接続すると、ping を送ったり、ssh でアクセスすることができた。

この技術により、Zynq では、PHY chip がなくても LAN(Copper) ケーブルの Ethernet 通信が可能であることがわかった。

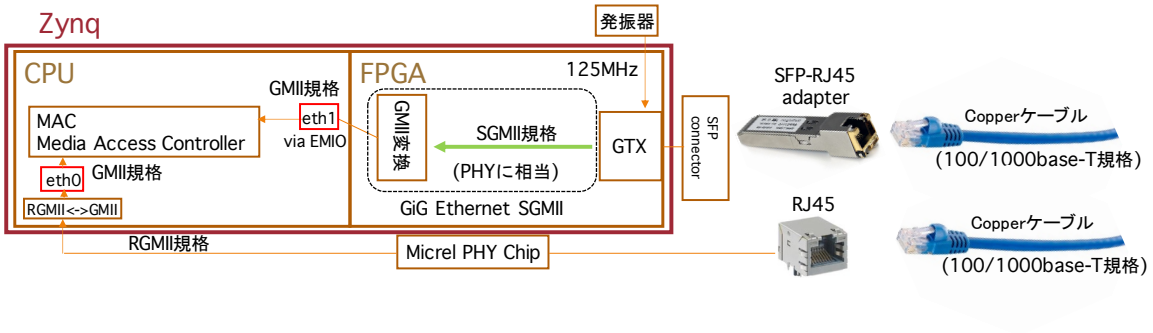


図 G.1　PHY chip 不要な Zynq の Ethernet 通信．SFP-RJ45 adapter を使用して、SGMII 規格の信号処理を Zynq PL 内で行う。すると、LAN ケーブルの 1000BASE-T 規格の Ethernet 通信を SFP、Zynq PL 経由で行えるようになる。

Appendix H

XVC を利用した Debug Bridge 機能

Zynq-7000 では、自分の PL 領域の信号線を XVC 機能によりデバックすることができる。図 F.1 の通り、TCP/IP 通信で Zynq PS にアクセスしたら、PS → PL(‘Bridge’) →自分の configuration Bank 0 と、Zynq の閉じた環境にて JTAG 通信が行える。すると、隣の JATHub から JTAG 通信するのと同じ要領で、自らの Zynq PL を configuration できる。

図 H.1 は、この Debug Bridge 機能を使用して、ネットワークアクセスした Zynq の PL の信号線を ILA によってデバックしている様子である。

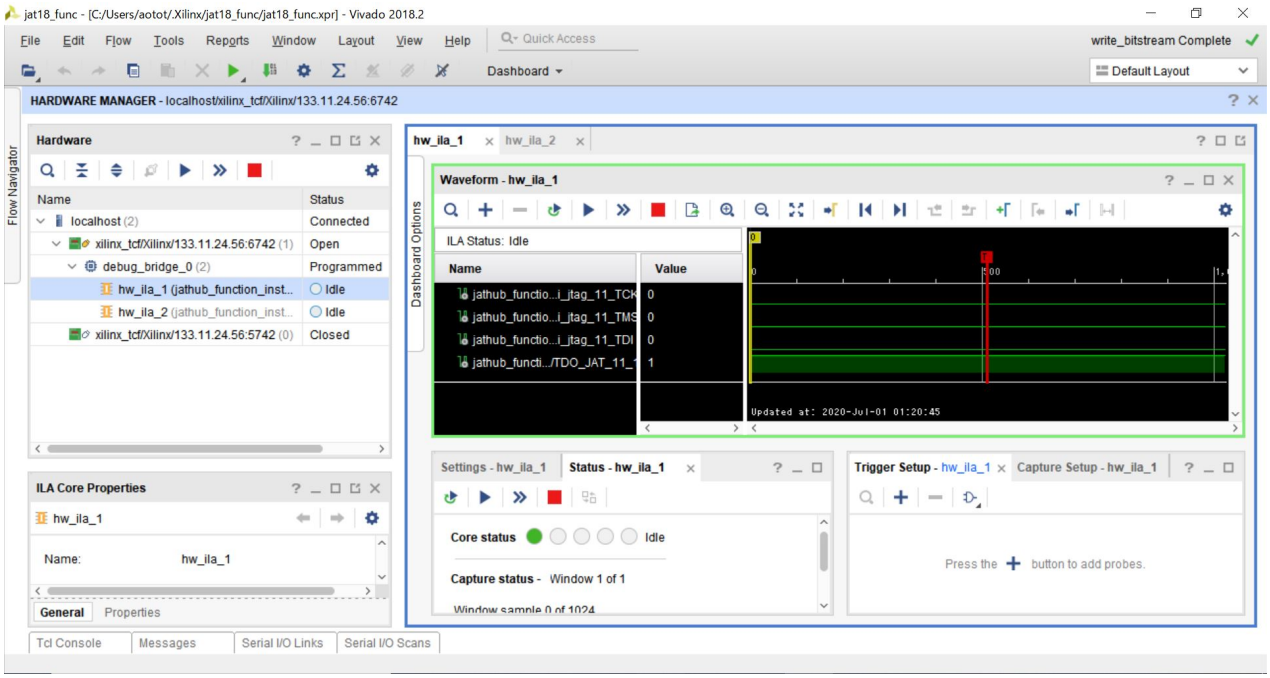


図 H.1　Debug Bridge 機能によって PL の信号線をデバックする様子

Appendix I

SVF ファイルの中身 (Kintex-7 QSPI program)

どの module にも接続していない Vivado HM 上でも、SVF ファイルは図 I.1 のように作成できた。

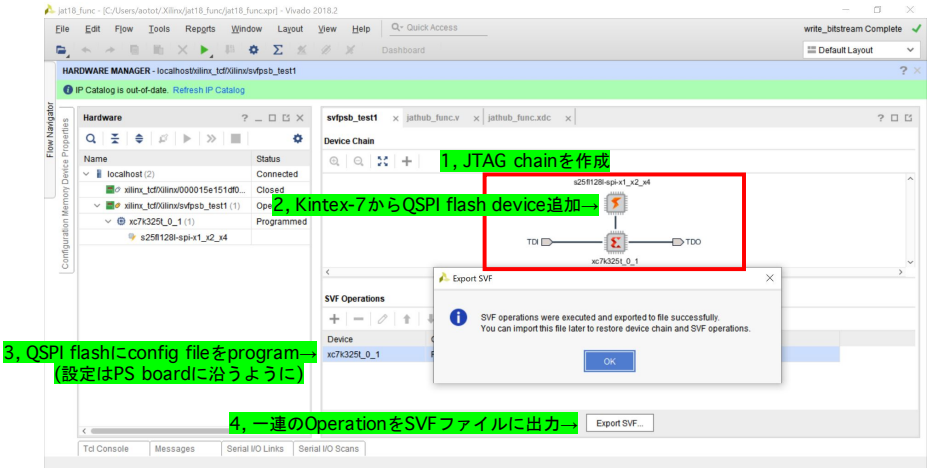


図 I.1　Vivado HM での SVF ファイル作成方法. 手順通りに作成していく。

FPGA Kintex-7 の QSPI に configuration file をプログラムする時、SVF ファイルには図 I.2 のような手順が書かれてあった。“State of non-config mem I/O pins”の設定は、FPGA Kintex-7 の設定と一致してなければならない。Kintex-7 の status register“End of Startup(EOS)”が 1 となるのは、FPGA の動作準備が完了したことを意味する。

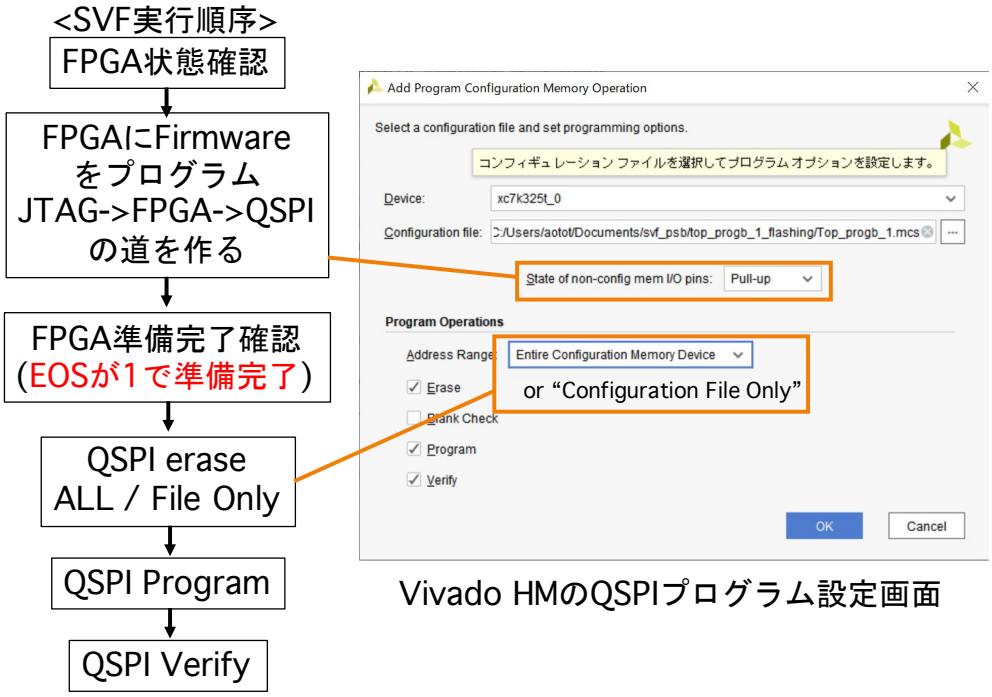


図 I.2 Kintex-7 の QSPI に configuration file をプログラムする時の SVF ファイルに書かれた手順内容

参考文献

[1] KEK ホームページ. ノーベル物理学賞でたどる標準理論 100 年の歴史. 2012/7/27. https://www.kek.jp/ja/newsroom/2012/07/27/1500/ .

[2] CERN Document Server. Overall view of the LHC. Dec. 6th, 2014. https://cds.cern.ch/record/1708847 .

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

[4] CERN, HL-LHC industry. Project Schedule. https://project-hl-lhc-industry.web.cern.ch/content/project-schedule

[5] The ATLAS Collaboration. *Technical Design Report for the Phase-II Upgrade of the ATLAS Trigger and Data Acquisition System*. CERN-LHCC-2017-020. 15 June 2018.

[6] Vivado Design Suite. *LogiCORE IP Soft Error Mitigation Controller v4.1*. Xilinx, PG036 (v4.1). Sep. 30, 2015

[7] Yasuyuki Horii, ATLAS Collaboration. *CERN-LHCC-2017-017*, ATLAS-TDR-026, 24 Apr. 2019. https://twiki.cern.ch/twiki/bin/view/AtlasPublic/ApprovedPlotsMuon#Phase_II_upgrade_study_for_TGC_i

[8] 中沢 遊. *COMET Phase-I CDC* 用読み出し回路のファームウェア開発及び性能評価. 2016/3/7. http://www-kuno.phys.sci.osaka-u.ac.jp/papers/m-thesis/fy2015_nakazawa_m-thesis.pdf

[9] A. Tanaka; M. Ikeno; M. Ishino; Y. Okumura; O. Sasaki; M. Shoji; K. Sugizaki; K. Tauchi. *Development Project -JTAG Assistance Hub-*, 26 Nov. 2019. http://openit.kek.jp/project/JATHub/JATHub

[10] C. Toner, et al. (Radiation Protection, European Organization for Nuclear Research (CERN), Geneva, Switzerland.) *Fault resilient FPGA design for 28 nm ZYNQ system-on-chip based radiation T monitoring system at CERN*. Elsevier Ltd., 9 August 2019. https://doi.org/10.1016/j.microrel.2019.113492

[11] Texas Instruments(TI.com). *QUAD HIGH-SPEED DIFFERENTIAL RECEIVERS*. https://www.ti.com/lit/ds/symlink/sn65lvds348.pdf?HQS=TI-null-null-digikeymode-df-pf-null-wwe&ts=1606289673235. May 2004.

[12] Texas Instruments(TI.com). *DS90LV047A3-V LVDS Quad CMOS Differential Line Driver*. https://www.ti.com/lit/ds/symlink/ds90lv047a.pdf?HQS=

dis-mous-null-mousermode-dsf-pf-null-wwe&ts=1608748068006&ref_url=https%253A%252F%252Fwww.mouser.tw%252F. May 2020.

[13] Xilinx. *UG585 Zynq-7000 SoC Technical Reference Manual*. July 1, 2018. https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf.

[14] 坂本 宏. *Vivado・Petalinux* 関連作業ページ. 24 Dec. 2020. http://www.icepp.s.u-tokyo.ac.jp/~sakamoto/research/atlas/tgcelex/vivado/

[15] 東田 旺大. 大規模高エネルギー実験における電子回路モジュール 制御への *system-on-a-chip* デバイスの応用研究. 2019/1/24. https://www.icepp.s.u-tokyo.ac.jp/download/master/m2018_higashida.pdf

[16] Xilinx. *UG865, Zynq-7000 SoC Packaging and Pinout Product Specification*. https://www.xilinx.com/support/documentation/user_guides/ug865-Zynq-7000-Pkg-Pinout.pdf. June 22, 2018.

[17] Anil Kumar A V, Radhey Shyam Pandey and Naveen Kumar Gaddipati. *PS and PL Ethernet Performance and Jumbo Frame Support with PL Ethernet in the Zynq-7000 SoC*, Xilinx, XAPP1082 (v5.0). July 16, 2018.

[18] Alvin Clark (Avnet Inc.), and Luis Bielich. *Xilinx Virtual Cable Running on Zynq-7000 Using the PetaLinux Tools*, Xilinx, XAPP1251 (v1.0). Apr. 30, 2015.

[19] Clifford Wolf, Lib(X)SVF: A library for implementing SVF and XSVF JTAG players. http://www.clifford.at/libxsvf/

[20] Vivado Design Suite. *Clocking Wizard v6.0 -LogiCORE IP Product Guide-*, Xilinx, PG065 (v6.0). Feb. 5, 2020.

[21] Maxim Integrated Products, Inc. ± *15kV ESD-Protected, Single/Dual/Octal, CMOS Switch Debouncers*. https://datasheets.maximintegrated.com/en/ds/MAX6816-MAX6818.pdf. Feb. 2020