修　士　学　位　論　文

Zynq を用いた
FPGA ロジック監視システムの構築

令和 4 年 2 月 4 日

専攻名　　物理学専攻
学籍番号　　195S105S
氏名　　末田 皓介

神戸大学大学院理学研究科博士課程前期課程

概要

現在，我々の世界を記述することに最も成功している物理モデルは標準模型である．しかし，標準模型では説明できない物理現象も多く確認されている．それらの説明のためには，新しい物理モデルの導入が必要である．新しい物理モデルが予言する新物理の兆候を探すために，世界中で様々な手法を用いて物理実験が行われている．中でも，高エネルギー粒子の衝突により生じる二次粒子を巨大な検出器で検出することで高統計量の物理イベントを貯め，それらのデータを解析することによって新物理の発見を目指す高エネルギー実験という手法は，極低確率で起こると考えられる新物理の反応の発見に非常に有効であると考えられる．

ATLAS 実験を始めとする高エネルギー実験では，膨大な検出データの読み出しや処理に，多数の計算機や集積回路が用いられている．中でも，デジタル信号の高速並列処理を得意とするデバイス-FPGA で実装されるデータ処理ロジックは，実験の高エネルギー化・高ルミノシティ化が進み，処理する必要のあるデータのレートが増加する中で，ますます巨大かつ複雑なものとなっている．現状では，FPGA ロジックが正常に動作しているかを判断するためには，計測室から送られてきたデータを，複雑な条件分岐を含むデータ処理を得意とするデバイス-CPU を用いてオフライン解析する必要があるが，データ読み出しシステムの安定稼働のためには計測室内でのオンラインモニタリングシステムが不可欠である．

理想的なオンラインモニタリングシステムには，デジタル信号の高速並列処理の要素と，複雑な条件分岐を含むデータ処理の要素の，どちらも必要であると考えられる．私は，FPGA と CPU が 1 チップ上に集積されたデバイス-SoC が，このシステムの実現に適していると考え，Xilinx 社製 SoC デバイス Zynq を用いて，リアルタイム FPGA ロジック監視システムの構築を行った．本論文では，システムの概要及び，その開発と性能評価，応用について述べる．

目次

第 1 章　イントロダクション　6
- 1.1　標準模型と新物理　6
- 1.2　高エネルギー物理実験　7
- 1.3　FPGA　9
- 1.4　CPU　10
- 1.5　SoC　10
- 1.6　FPGA ロジック監視システム　10

第 2 章　Zynq　12
- 2.1　Programmable Logic (PL)　12
- 2.2　Processing System (PS)　14
- 2.3　インターコネクト　14
- 2.4　AXI4　17

第 3 章　Zynq を用いた FPGA ロジック監視システムの開発と性能評価　20
- 3.1　システムの開発環境　21
- 3.2　Zynq PS 上で作動する Linux OS の構築　25
- 3.3　PS-PL 間通信システムの開発と性能評価　29
- 3.4　監視対象ロジックの出力の可否判定アプリケーションの開発と性能評価　40
- 3.5　ロジック回復システムの開発　46
- 3.6　システムの応用例　47

第 4 章　結論　49

参考文献　53

付録 A　PTZ 回路図　55

図目次

1.1 標準模型で予言される素粒子. 6
1.2 一般的な高エネルギー物理実験のデータ取得システム. 8
1.3 理想的なリアルタイムモニタリングシステム. 8
1.4 4 入力 1 出力の LUT. 9

2.1 Zynq のブロック図. 13
2.2 GUI でのハードウェア設計. 14
2.3 Zynq のインターコネクトのブロック図. 15
2.4 AXI 規格の典型的な通信の様子. 17
2.5 AXI 規格での，マスターからスレーブのデータの読み出し. 18
2.6 AXI 規格での，マスターからスレーブのデータの書き込み. 18
2.7 AXI-Stream プロトコルの典型的な信号. 19

3.1 Zynq 搭載汎用 VME ボード PTZ. 21
3.2 NIM コネクタの出力テスト結果. 24
3.3 NIM コネクタの入力テスト結果. 24
3.4 GTX の IBERT 試験で得られた eye パターン. 26
3.5 PTZ の Linux OS の起動ログ 28
3.6 PTZ の Linux OS に ssh ログインした様子 29
3.7 CPU から PL FPGA ロジックの制御レジスタへのアクセスの通信経路および使用インターフェイス. 30
3.8 31
3.9 AXI_HP インターフェイスを使用した PS DDR アクセスの通信経路. 31
3.10 PL FPGA ロジックから PS DDR への DMA を実現する通信システム. 32
3.11 AXI4-Stream 信号をユーザーロジックで使用できる形式へ変換するロジック. . . . 33
3.12 ユーザーロジックの出力信号を AXI4-Stream 信号へ変換するロジック. 33
3.13 PL Logic から PS DDR への書き込みをロジックアナライザで確認した様子. . . . 36
3.14 PS DDR から PL Logic への読み出しをロジックアナライザで確認した様子. . . . 36
3.15 PL DDR を用いた PS-PL 間の通信システム. 37
3.16 AXI_GP_S インターフェイスを用いたタイマーのブロック図. 39

3.17 AXI_GP_S インターフェイスを用いたタイマーを使用する際のタイミング図. . . . 39
3.18 ルックアップテーブルの模式図. 41
3.19 bit 処理を行うテストロジック. 42
3.20 ルックアップテーブルを用いた正誤判定ロジックの概念図. 42
3.21 データ読み出しを模したテストロジック. 43
3.22 hit1 data をヒストグラミングした図. 44
3.23 Linux アプリケーションを用いた FPGA 監視システム. 45
3.24 エラーを検知した時のホスト PC のターミナルの様子. 45
3.25 NSL に実装されるトリガーロジックおよび読み出しロジックのブロック図. 48

A.1 PTZ 回路図，全体図. 56
A.2 PTZ 回路図，Zynq のコンフィギュレーションバンク (BANK0) と PL バンク (BANK9/10). 57
A.3 PTZ 回路図，Zynq の PL バンク (BANK11/12/13). 58
A.4 PTZ 回路図，Zynq の PL バンク (BANK33/34/35). 59
A.5 PTZ 回路図，Zynq の PL バンク (BANK109/110/111/112). 60
A.6 PTZ，Zynq の PS バンク (BANK500/501). 61
A.7 PTZ 回路図，Zynq の PS バンク (BANK502)．DDR3 メモリ接続. 62
A.8 PTZ 回路図，Zynq の電源バンク. 63
A.9 PTZ 回路図，Zynq の GND バンク. 64
A.10 PTZ 回路図，VME コネクター. 65
A.11 PTZ 回路図，VME アドレス線制御. 66
A.12 PTZ 回路図，VME データ線制御. 67
A.13 PTZ 回路図，メザニンコネクタ (A,B,C,D). 68
A.14 PTZ 回路図，メザニンコネクタ (E,F,G,H,J,K). 69
A.15 PTZ 回路図，メザニンコネクタ GND. 70
A.16 PTZ 回路図，JTAG コンフィギュレーション回路. 71
A.17 PTZ 回路図，Si5344 と外部/内部クロック周辺. 72
A.18 PTZ 回路図，NIM コネクタ. 73
A.19 PTZ 回路図，PS DDR3 メモリ. 74
A.20 PTZ 回路図，PS DDR3 メモリ. 75
A.21 PTZ 回路図，PL DDR3 メモリ. 76
A.22 PTZ 回路図，USB-UART モジュール. 77
A.23 PTZ 回路図，RJ45 コネクタと PHY モジュール. 78
A.24 PTZ 回路図，microSD モジュールと QSPI フラッシュメモリ. 79
A.25 PTZ 回路図，GTX 通信用 SFP+ モジュール．1 台はメザニンコネクタに接続. . . 80
A.26 PTZ 回路図，PS/PL コンフィギュレーションリセットスイッチとボード電源リセットスイッチ. 81
A.27 PTZ 回路図，電源レギュレーターその 1. 82

A.28 PTZ 回路図，電源レギュレーターその 2. 83
A.29 PTZ 回路図，電源レギュレーターその 3. 84

表目次

2.1 PS-PL 間通信の理論上の帯域幅 16
2.2 AXI 規格の通信の各チャンネルに用いられる主要な信号 19

3.1 Ethernet 通信テスト結果 25
3.2 GTX 通信 IBERT テスト結果 25
3.3 kernel image に含めた module，device driver 27
3.4 PS-PL 間通信速度 40
3.5 ルックアップテーブルの使用リソース 42

第 1 章
イントロダクション

1.1 標準模型と新物理

標準模型は，物質を構成する最小単位である素粒子とその相互作用を説明する物理モデルである．標準模型は，12 種類のフェルミオンと 4 種類のゲージボソン，そしてヒッグス粒子の計 17 種類の粒子の存在を予言する (図 1.1)．2012 年，LHC-ATLAS 実験と CMS 実験によりヒッグス粒子が発見され [1][2]，標準模型で予言されるすべての粒子が発見された．標準模型は現在までの物理実験の結果のほとんどを説明できるが，その一方で，強い CP 問題，ニュートリノ振動，物質と反物質の非対称性，ダークマター/ダークエネルギーの存在，ヒッグス粒子の階層性問題等，標準模型では説明できない現象も観測されている．これらの現象の説明のためには，標準模型を超えた新たな理論 “新物理” の導入が必要であり，世界中で様々な方法で新物理を探索する実験が行われている．

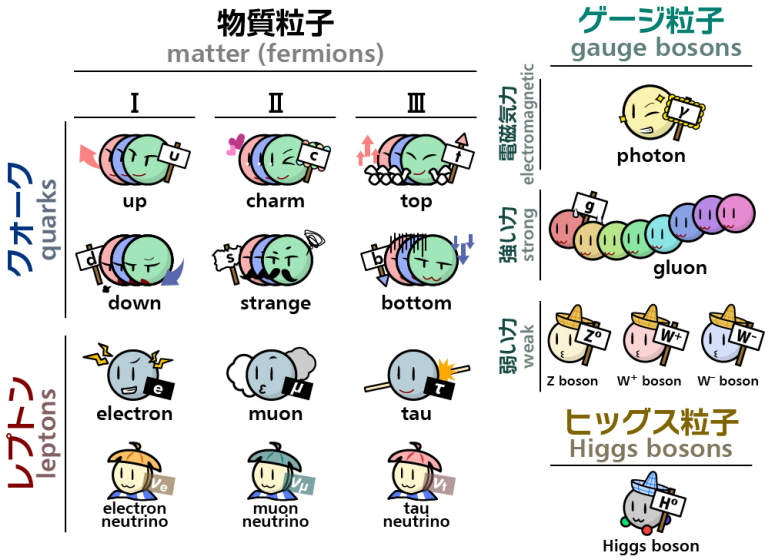

図 1.1　標準模型で予言される素粒子．[3]

1.2 高エネルギー物理実験

前述した“新物理”は，非常にクロスセクションが小さく，稀なイベントである．そのため，それら“新物理”の発見・精査のためには，極めて多くの反応事象を発生させ取得したデータから統計処理によりバックグラウンドを除く手法，もしくはバックグラウンドを極めて抑制した環境下で極低頻度で起こる反応事象を直接探索する手法が考えられる．例として，ヒッグス粒子の階層性問題やダークマターの存在の問題を解決する理論の一つである超対称性理論が予言する典型粒子の，重心系エネルギー 13 TeV の陽子-陽子衝突における生成クロスセクションは pb(pico barn) オーダーである [4]．多くの反応事象を発生させる一般的な新物理探索の手法として，加速した粒子ビーム同士を衝突させ，生じた二次粒子を観測する高エネルギー実験がある．例として ATLAS 実験では，加速された陽子ビーム同士を重心系エネルギー 13 TeV で衝突させることで，最大瞬間ルミノシティ 1.9×10^{34} cm^{-2}s^{-1} という非常に高い統計量を確保している [5]．陽子ビーム衝突頻度は 40.08 MHz で，1 回の衝突当たり平均 23 個の陽子が反応を起こす．そのため，イベントレートは 1 GHz 程度と計算できる．1 GHz のイベントの中で，後述するトリガーシステムにより物理的価値があると判断され，最終的にストレージに保存されるイベントレートは 1000 Hz 程度である [6]．瞬間ルミノシティから計算される超対称性粒子の生成レートは，10^{-6} Hz オーダーであり，この反応を捉えるためには膨大な観測データが必要であることがわかる．

1.2.1 高エネルギー実験のデータ取得システム

一般的な高エネルギー実験で，取得した観測データを保存する手法を図 1.2 に示す．検出器が粒子を捉えた際に出力されるアナログ信号は，デジタイザを通じてデジタル信号に変換され，様々な処理を施された後にリードアウトシステムにより実験室からモニタリングルームに送られ，さらにデータ加工されてストレージに保存される．この一連のシステムを DAQ システムという．前述したとおり，高エネルギー実験の時間当たりのイベント数は膨大で，それに伴って，観測データのデータサイズもまた膨大となり，それらすべてをストレージに保存することは現実的でない．例として ATLAS 実験の全検出器の観測データのレートは 60 GByte/sec に及ぶ [6]．

観測データから物理的価値のあるイベントの取捨選択を行うシステムをトリガーシステムという．中でも，実験室内でのオンライントリガーシステムは，非常に高いレートのデータを，バッファーに一時退避させている短い時間の中で処理する必要があるため，デジタル信号の高速並列処理に適しているデジタル回路により実装される．例として ATLAS 実験では，40 MHz で取得される観測データを，デジタル回路で実装された初段のオンライントリガーで 100 kHz まで落とす．このトリガーの処理は 2.5 μs 以内に行われる．

これらのデータ取得システムは数多くのハードウェアによって構築されている．例として, ATLAS 実験の内部の pixel 飛跡検出器では約 1,700 台のディジタイザモジュール，約 900 個の FPGA を用いて，約 80,000,000 チャンネルのデータを読み出す [7]．

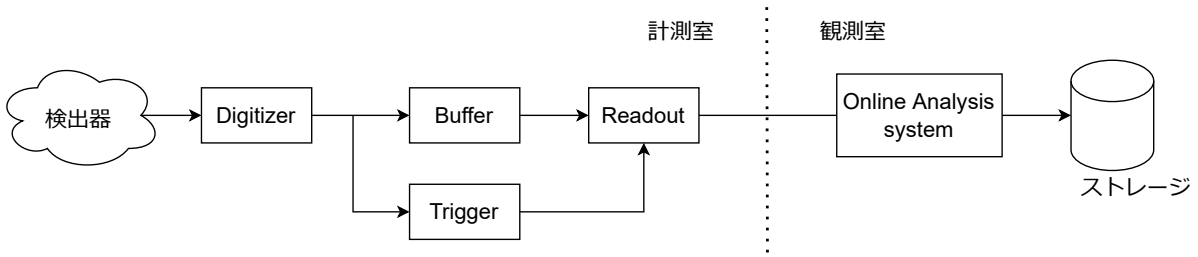

図 1.2　一般的な高エネルギー物理実験のデータ取得システム．

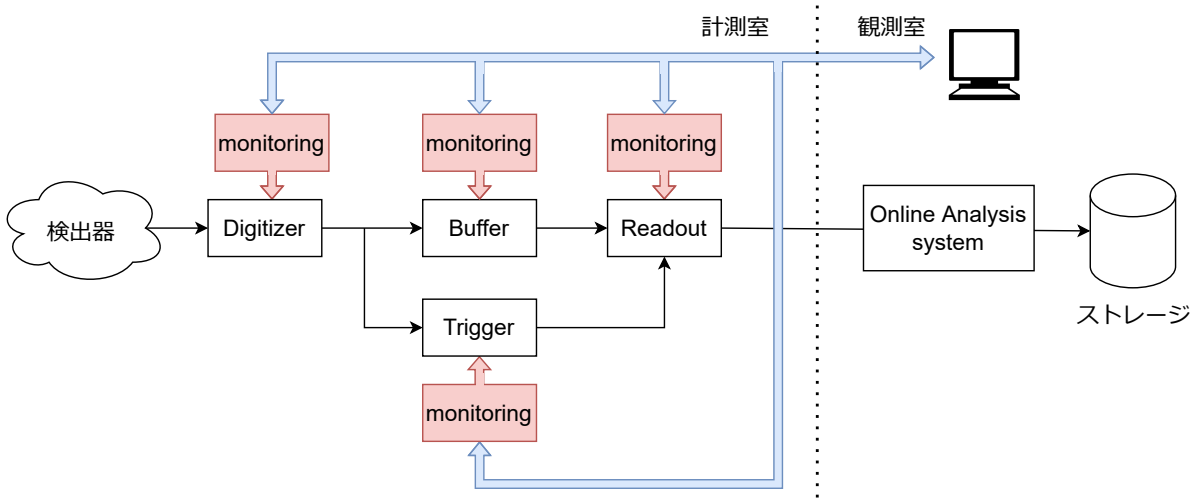

図 1.3　理想的なリアルタイムモニタリングシステム．

1.2.2 システムのモニタリング

高エネルギー実験のトリガー/DAQ システムは，多数の FPGA デバイスや IC デバイスにより構築される．これらのデバイス，およびそこに実装されるロジックは複雑に影響しあい，その一部の動作に異常が発生すると，システム全体の異常につながる恐れがあり，重要な物理データの損失につながる可能性がある．例として ATLAS 実験では，2018 年 4 月 30 日にミューオントリガーロジックファームウェアの更新を行った際，ファームウェアに問題が含まれていたためにトリガー発行レートが跳ね上がり，その結果数時間にわたりデータの取得が出来ない状態が続いた [8]．このような問題の回避のために，トリガー/DAQ システムを構築するデバイスやロジックのモニタリングシステムの開発は，高エネルギー実験において重要な課題であると言える．しかし，実験室でデータ処理を行っている FPGA デバイスでは高度な演算や条件処理を行うことは難しいため，リードアウトモジュールから得られたデータをモニタリングルームの CPU で解析し，間接的に FPGA デバイスや IC デバイスのモニタリングを行っているのが現状である．現状のシステムには，トリガー/DAQ システムに問題が起きた時にどのデバイスの異常なのか判別しにくいことや，問題が起こってからそれに気づくまでのタイムラグが大きいという課題がある．理想のモニタリングシステムは，各デバイスの入力信号や出力信号を実験室の中で個々に独立に監視し，その状態をリアルタイムでモニタリングルームに伝えることが出来るというものである（図 1.3）．本論文では図中の色がついている部分を総

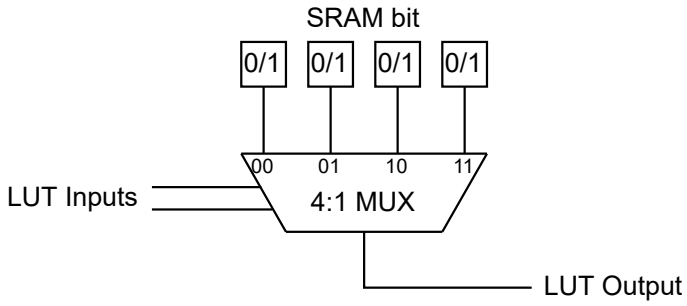

図 1.4　4 入力 1 出力の LUT.

称して，FPGA ロジック監視システムと呼称する．

1.3 FPGA

FPGA とは Field Programmable Gate Array の略で，書き換え可能なデジタル回路デバイスである．Look Up Table (LUT)，Frip Frop (FF)，マルチプレクサ (MUX)，Block Random Access Memory (BRAM) などが内部に設置されており，それらをつなぐ配線を外部から書き換えることで，様々な論理回路の実装が可能である．外部とのデータのやり取りのために，豊富な Input/Output (I/O) が用意されていることが多い．デジタル回路の利点として，大規模データの並列処理や，高速処理が得意であるということがあげられる．一方で，複雑な条件分岐のあるアプリケーションの実装は難しいため，FPGA だけで独立したシステムを作ることは難しい．また，放射線の影響などで，稀に Single Event Upset (SEU) という内部メモリなどの bit 反転によるエラーが生じることがあり，注意が必要である．例として ATLAS 実験では，上記のメリットを利用して，大規模データの高速処理が必要であるオンライントリガーロジックの実装，データ読み出し回路の実装，データの圧縮，などに用いられている．[9]

MUX

MUX とは 2 つ以上の入力を 1 つの信号として出力する機構である．組み合わせ回路に用いられる．

LUT

LUT とは，ある入力に対してある出力をする機構である．図 1.4 に 4 入力 1 出力 LUT のブロック図を示す．Static Random Access Memory (SRAM) に保存された 1bit 幅の 4 つの値をもとに，MUX が input に対し，output を決定する．組み合わせ回路に用いられる．

FF

FF とは 1bit 幅のデータを保持することのできる機構である．本論文ではレジスタと呼称することもある．順序回路の実装に用いられる．

1.4 CPU

CPU は Central Processing Unit の略で，外部メモリやストレージと連携して，複雑なアプリケーションを実現することが出来るデバイスである．また，オペレーティングシステムにより，外部デバイスの管理や，通信制御，アプリケーションの実行順序の管理などを行うことが可能で，独立したシステムとして稼働することが可能である．FPGA と比べ，複雑なアプリケーションの実行が可能であるが，逐次計算による処理のため，データの並列処理の速度の面では FPGA に劣る．例として ATLAS 実験では，上記のメリットを利用して，複雑な条件分岐を含むハイレベルトリガーの実装，タイミング調整などの機能を搭載したデータ読み出しの制御，FPGA ボード間通信の制御などに用いられている．[10]

1.5 SoC

SoC とは System on a Chip の略で，1 つのチップ上に CPU や I/O，FPGA などを集積し，連携してシステムとして機能するように設計された集積回路である．FPGA と CPU をボードに別々に実装するのに比べ，設計コストや部品コストが削減できることや，FPGA とプロセッサ間のインターコネクトがより密接になるため，FPGA と CPU の連携したアプリケーションの開発が容易になるなどの利点がある．また，CPU を搭載していることで，Linux OS をチップ上で走らせることが可能であり，オープンソースの kernel driver を用いて，様々な外部デバイスを容易に制御することが可能である．FPGA 上に実装した大規模データの高速並列処理機能を，CPU に搭載した Operating System (OS) で制御するシステムの実装や，FPGA で処理したデータをプロセッサを接続したネットワークで利用するなどの利用例が考えられる．例として ALTAS 実験では，Linux kernel による Ethernet 通信制御を用いた複数の FPGA ボード間の通信，Linux アプリケーションによる複数の FPGA ボードの一括制御などに用いられる予定である．

1.6 FPGA ロジック監視システム

1.2.2 項で述べた理想的な FPGA ロジック監視システムの要件は以下の 4 つと考えられる．

1. 監視対象 FPGA ロジックの入力信号や出力信号を捕捉可能であること．
2. 捕捉した信号を用いて監視対象 FPGA ロジックが正常に作動しているか否かを判定すること．
3. 監視対象ロジックの状態をモニタリングルームへ伝える，もしくはモニタリングルームからロ

ジックの回復を行うこと．

4. 監視システムが各々の FPGA に対して独立に稼働すること．

1 は，FPGA ロジックのモニタリングを行う上で，最も必要な要素といえる．FPGA に実装されるロジックは，ある入力に対し，何らかの処理を施し，ある出力を提供する．その入力信号および出力信号をロジックの外部から捕捉することで，FPGA ロジックの内部処理を間接的に把握することが可能となる．2 は，モニタリングシステムの核である．1 で取得した信号をもとに，監視対象のロジックが正常に働いているかを検証する．この検証は，FPGA ロジックを用いるもの，Linux アプリケーションを用いるものと様々考えられる．監視対象ロジックから取得するデータの形式や，監視対象ロジックのデータ処理方法などから総合的に判断し，それぞれのロジックに適したアプリケーションを開発する必要がある．3 は 2 で判断した結果をどう運用するかの部分である．モニタリングルームから FPGA の状態がリアルタイムで把握でき，なおかつエラーが起こった場合，その回復を速やかに行うことのできるシステムの構築が必要となる．4 は，以上の要素を持った監視システムが実験室内で独立して稼働するために必要である．FPGA をモニタリングするシステムが監視対象 FPGA ロジックの近くに独立して存在することで，通信リソースの削減や，真の意味でのリアルタイムモニタリングが可能となる．

1，2 は FPGA で実装できるが，3，4 の実装にはプロセッサが必要となる．逆に，1 の実装はデジタル回路でなければ難しいため，FPGA とプロセッサはどちらもシステムの構築に不可欠であると言える．よって，本研究では，以上 1-4 の要素をすべてを 1 チップで実現することのできるデバイス，SoC でのシステム開発を行った．

1.6.1 本論文の構成

第 1 章では，高エネルギー物理実験における，FPGA ロジックのリアルタイムモニタリングシステムの必要性，およびモニタリングシステムに求められる性質を示した．第 2 章では，監視システムを構築するデバイス Zynq の，基本的な性能および，監視システムに応用が出来る機能について述べる．第 3 章では，監視システムの開発と性能評価および，システムの応用性について述べる．最後に第 4 章で，本論文の結論を述べる．

第 2 章
Zynq

Zynq とは，Xilinx 社製の SoC である．ブロック図を図 2.1 に示す．Zynq は，FPGA を含む Programmable Logic (PL) と CPU を含む Processing System (PS) の 2 つのパートで構成されている [11]．PS と PL は複数のインターフェイス，インターコネクトで接続されており，FPGA と CPU が密に連携したアプリケーションの実装が可能である．

2.1 Programmable Logic (PL)

PL FPGA は，Configurable Logic block(CLB)，BRAM，Digital Signal Processor (DSP)，Programmable I/O Blocks で構成されている．CLB には 6 入力 LUT と 8 つのレジスタが MUX で接続され構成される “Slice”が 2 つ含まれる．Slice には 2 種類あり，1 つは，LUT に論理機能，シフトレジスタ機能，メモリ機能を実装可能な SLICEM で，もう 1 つは，論理機能のみを実装可能な SLICEL である．1 つの CLB には SLICEL が 2 つ，もしくは SLICEL と SLICEM が各 1 つずつ含まれる．DSP は，25×18 ビットの 2 の補数乗算器および，8 ビットのアキュームレータで構成される計算機で，多数の並列計算が可能である．Programmable I/O Blocks は，I/O ピンを持ったプログラム可能な FPGA セルである．I/O ピンには High Range (HR) I/O と High Performance (HP) I/O の二種類があり，HR I/O は，1.2 V $\sim$ 3.3 V の広範な I/O 電圧をサポートし，HP I/O は 1.2 V $\sim$ 1.8 V の電圧をサポートしている．HP I/O は最高性能の動作向けに最適化されており，1,866 Mb/s の DDR3 インターフェイスの制御が可能である．また，最高データレート 12.5 Gbps の高速光通信用インターフェイス GTX が複数個搭載されており，外部との大規模な高速データ通信が可能である．例として ATLAS 実験では，GTX を用いて観測データの読み出しなどを行う [12]．

PL FPGA リソースを利用するためには，1. ハードウェア記述言語でのコーディング．2. 論理合成．3. 配置配線．4. デバイスへのコンフィギュレーションの 4 手順が必要である．論理合成とは，ハードウェア記述言語で記述された抽象的な回路のふるまいを，実際の FPGA リソースに置き換える工程である．配置配線とは，論理合成され，ハードウェアレベルに落とし込まれたロジックを，実際の FPGA リソースのどの場所に配置するかを決定する工程である．最終的に，コンフィギュレーションという形で FPGA 内部の RAM に，配置配線されたロジックが書き込まれ，それをもとに FPGA 内部の配線が完了する．

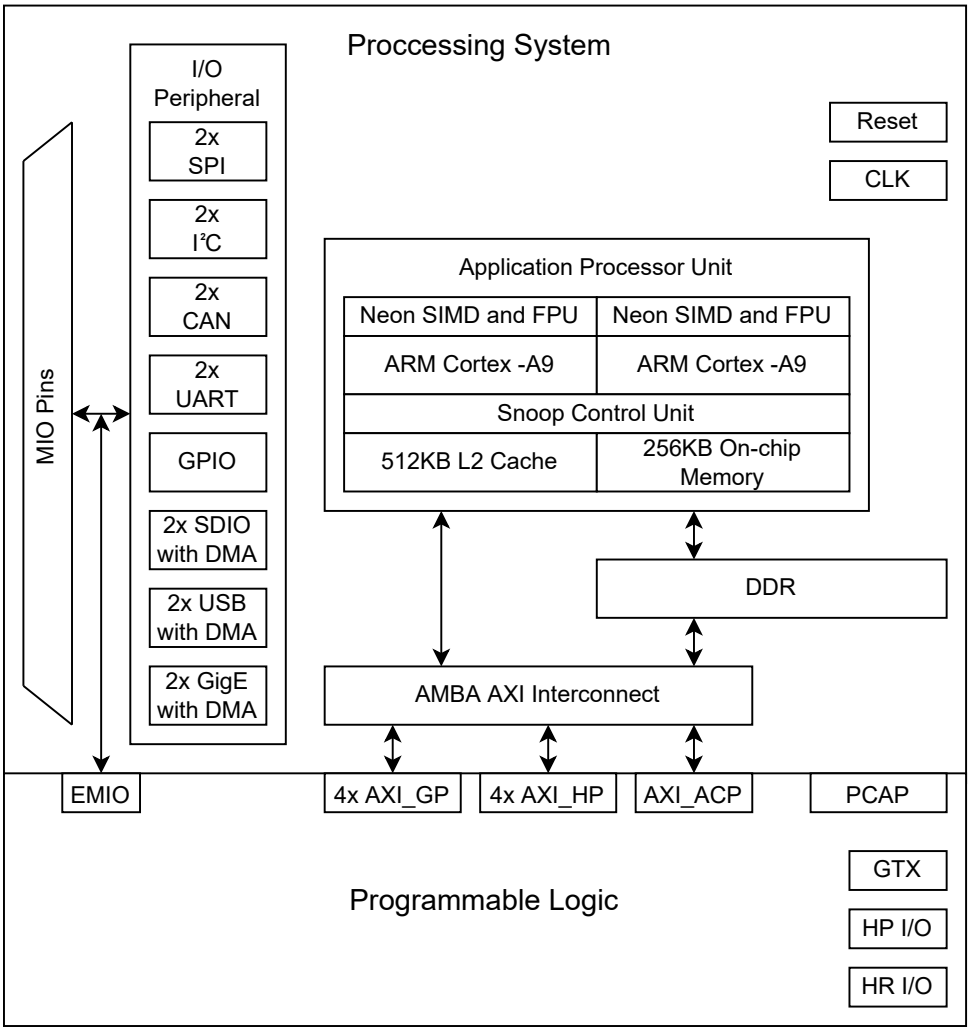

図 2.1　Zynq のブロック図.

本研究では，これらの工程を Xilinx Vivado で行った．Vivado では，従来のハードウェア記述言語でのプログラミングによるロジック開発はもちろん，GUI で行う，ブロックデザインという手法でロジックの開発を行うことが出来る (図 2.2)．ブロックデザインでは，1 つの RTL ファイルを 1 つのブロックとみなし, GUI で配線を行うことで, 複数の RTL ロジックを組み合わせてロジックを開発することが出来る．またブロックには，Xilinx 社により，様々な機能を持った Intellectual Property(IP) コアが多数用意されており，ユーザーはこの IP のパラメータを適切に設定し，組み合わせることで多様なロジックを開発することが可能である．もちろん，ユーザーも IP を作成することが可能である．また，PS-PL 連携アプリケーションを作成する際は，ZYNQ7 Processing System IP コア (図中右上) を設置し，ボードに搭載されている I/O ペリフェラルや DDR メモリ，AXI インターフェイスなどの情報を設定する必要がある．Vivado は，これらの情報をハードウェア定義ファイルとして出力することが可能で，ハードウェア定義ファイルは PS を利用したアプリケーションの開発や，Linux OS の動作に必要な Dvicce-tree ファイルの作成，Zynq 起動用ブートファイルの作成などに使用する．

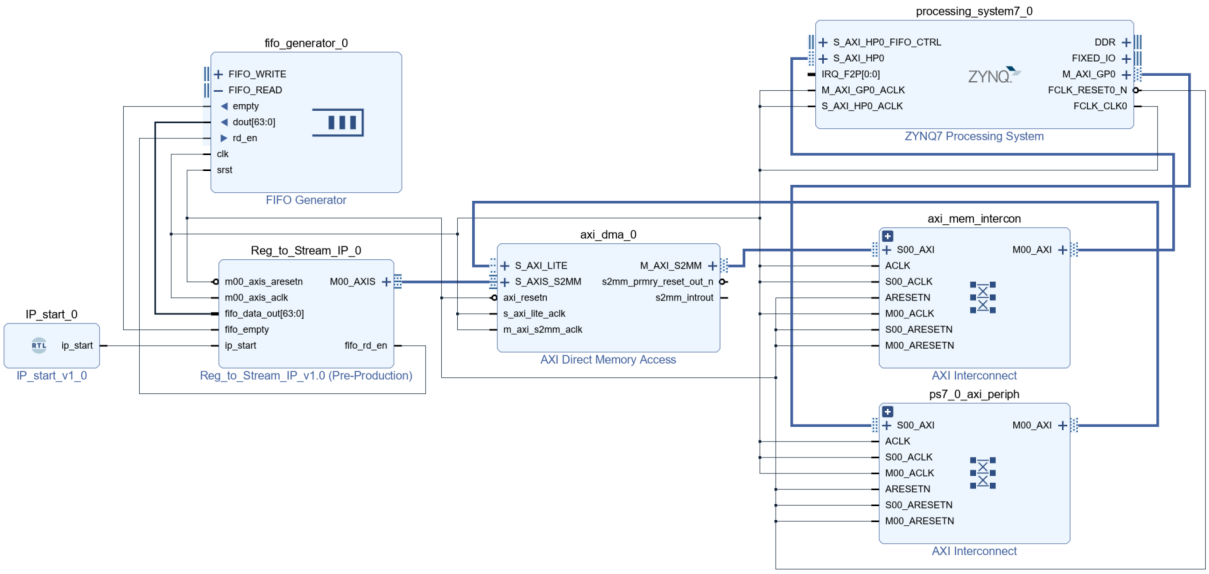

図 2.2　GUI でのハードウェア設計.

2.2 Processing System (PS)

PS は，Application Processor Unit (APU)，メモリインターフェイス，I/O ペリフェラル，インターコネクトで構成されている (図 2.1 中央)．APU はシステムの核となる部分で，Arm Cortex-A9 プロセッサを 2 つ含む．これらのプロセッサは，命令用とデータ用にそれぞれ 32 KB の L1 キャッシュを備えており，また 512 KB L2 キャッシュを共有している．スヌープ制御コントローラが 2 つのプロセッサと L2 キャッシュ間のコヒーレンシを保っている．L2 キャッシュと並列に存在する On-Chip メモリや DDR にシステムアドレスがマッピングされる．I/O ペリフェラルは，一般的なペリフェラルの制御を提供する．I/O ペリフェラルへのシステムアドレスのマッピングも可能である．また，一部の I/O ペリフェラルは DMA 転送を行うことが出来る．

PS を利用したアプリケーションの開発は Xilinx Vitis で行う．Vitis は，Vivado により作成したハードウェア定義ファイルを用いて，c 言語や c++ で記述されたアプリケーションを，搭載 I/O ペリフェラルや搭載メモリなどのボードの構成情報および，PL に実装された AXI インターフェイスの情報を含めてコンパイルすることが可能である．また CPU をベアメタルで使用するか，Linux kernel を用いて使用するかの選択が可能で，どちらの場合においても，Xilinx 社が多くのライブラリを用意しており，ユーザーは I/O ペリフェラルや AXI インターフェイスの操作を容易に行うことが可能である．

2.3 インターコネクト

Zynq の大きな特徴の 1 つが，PS-PL 間の密な接続である．図 2.3 に Zynq のインターコネクトのブロック図を示す [11]．これらのインターコネクトは全て，後述する AXI4 という通信規格で接続されている．

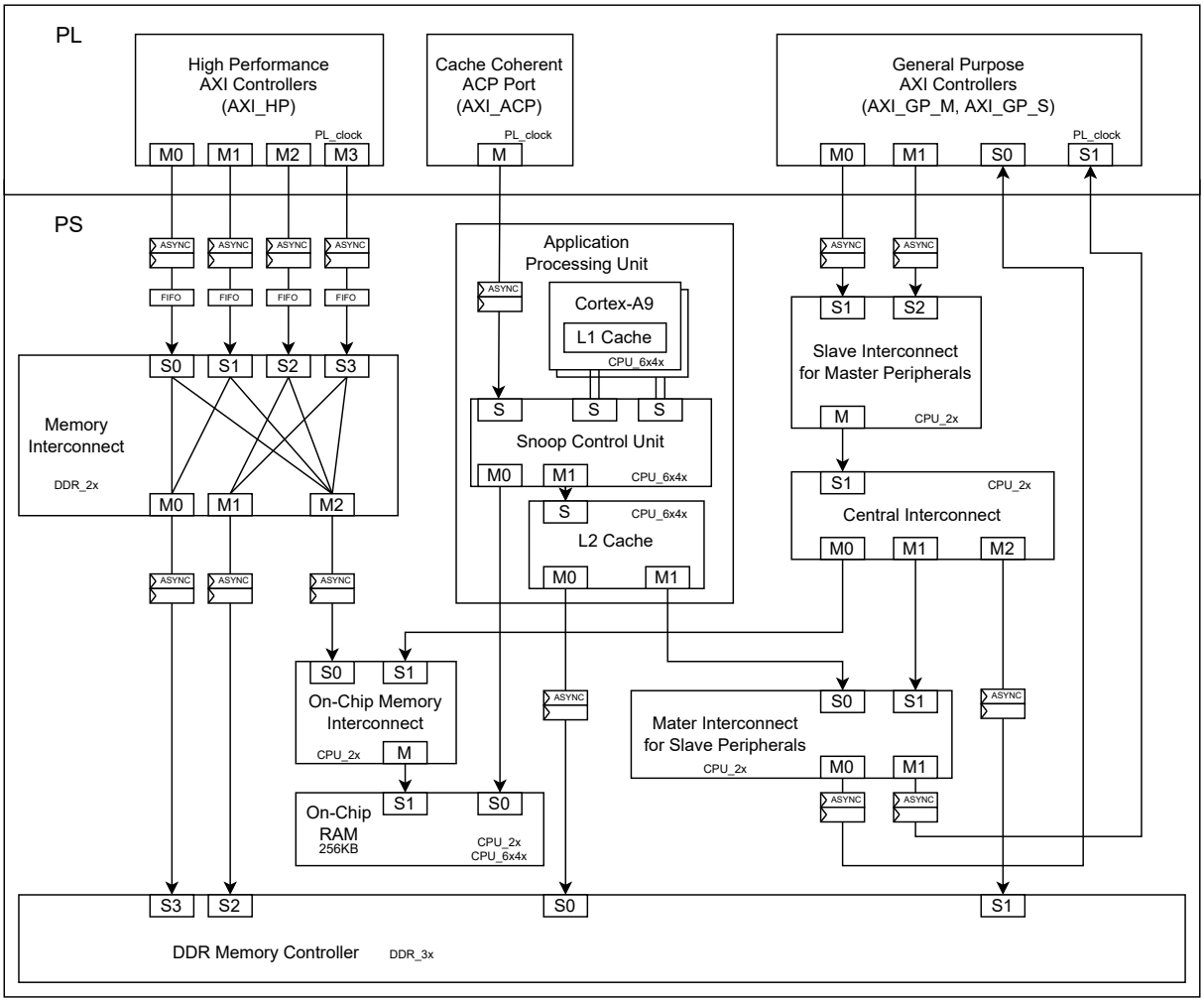

図 2.3　Zynq のインターコネクトのブロック図.

プロセッサと AXI_ACP インターフェイス

2 つのプロセッサと AXI_ACP インターフェイスはスヌープ制御コントローラに接続されており，L1 キャッシュとのコヒーレンシが保たれている．プロセッサと AXI_ACP インターフェイスは On-Chip メモリ，DDR メモリ，PL AXI_GP_S インターフェイスや，一部の I/O ペリフェラルにマッピングされたシステムレジスタにアクセスすることが可能である．ただし，CPU プライベートバスレジスタへのアクセスが可能なのはプロセッサだけである．CPU プライベートバスレジスタは APU サブシステムの制御に用いる．

AXI_HP インターフェイス

PL は AXI_HP インターフェイスを用いて，メモリインターコネクトを介して，DDR メモリと On-Chip メモリにアクセスできる．このインターフェイスは PL から PS メモリへの高スループットデータパスとして設計されており，各インターフェイスに読み出し，書き込みそれぞれのための FIFO が用意されている．主な仕様は以下のとおりである．

表 2.1　PS-PL 間通信の理論上の帯域幅

インターフェイス	マスター	バス幅 (bit)	書き込み/読み出し帯域幅 (MB/s)	インターフェイス数	総帯域幅 (MB/s)
AXI_GP	CPU	32	600	2	2,400
AXI_GP	AXI_GP	32	600	2	2,400
AXI_HP	AXI_HP	64	1,200	4	9,600
AXI_ACP	AXI_ACP	64	1,200	1	2,400

- AXI マスターインターフェイス.
- データ幅を 32 bit/64 bit で選択可能.
- 32 bit モードの時，アライメントされた転送を 64 bit に動的なサイズアップが可能.
- 32 bit モードの時，アライメントされていない 32 bit 転送を自動的に 64 bit に拡張可能.
- PS-PL 間のすべての AXI インターフェイスにおける非同期クロックに対応.
- 読み出し，書き込みの両方に 1 KB(128×64 bit) のデータ FIFO を使用し，レイテンシの大きい転送の平滑化が可能.

以上の仕様から，AXI_HP インターフェイスは大規模データの PS-PL 間転送に適していると言える．本研究で開発したシステムの PS-PL 間データ転送にはこのインターフェイスを用いた.

AXI_GP インターフェイス

AXI_GP インターフェイスは，マスターにもスレーブにもなりうる汎用の PL インターコネクトインターフェイスである．スレーブとして用いる際，PL の内部レジスタをシステムレジスタにマッピングし，CPU もしくは AXI_ACP インターフェイスから使用することが可能となる．また，マスターとして用いる際，インターコネクトを介して，On-Chip メモリ，DDR メモリ，AXI_GP_S インターフェイスにアクセスが可能である．AXI_GP インターフェイスは，マスターインターコネクトとスレーブインタコネクトのポートに直接接続されており，AXI_HP にあるような FIFO などは用意されていない．したがって，パフォーマンスはマスターインターコネクトとスレーブインターコネクトのポートの制約を受ける．よって，AXI_GP インターフェイスは大規模なデータ転送には適しておらず，主な用途としては，PS CPU からの PL FPGA ロジック内の制御レジスタの操作や，ステータスレジスタの値の読み出しなどがあげられる.

インターフェイスごとのデータレート

PS-PL 間の通信速度は，使用するインターフェイスおよび，マスターによって理論値が異なる．表 2.1 に理論値を示す．インターコネクトのクロック周波数はいずれも 150 MHz である．これらの数値は通信プロトコルのオーバーヘッドを考慮していないため，実際の通信帯域幅は理論値よりも小さくなると考えられる.

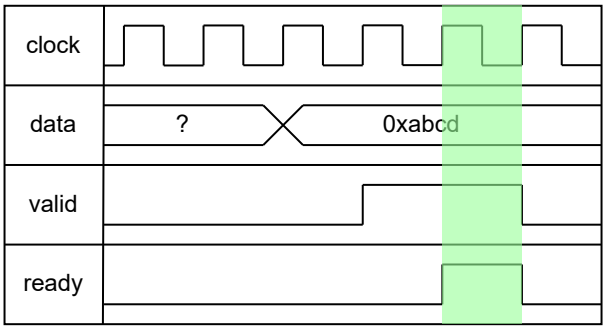

図 2.4　AXI 規格の典型的な通信の様子.

2.4 AXI4

Advanced eXtensible Interface 4 (AXI4) は，ARM 社が提供する 4 世代目の AMBA インターフェイス規格である [13]．Zynq のインターコネクトのインターフェイスの通信規格は AXI4 に統一されているため，ユーザーは一つの通信規格にのみ対応すればよく，PS-PL 間の連携したアプリケーションの開発を容易に行うことが出来る．また，Xilinx 社により多数の AXI 規格に対応する IP コアや Linux kernel driver が提供されているため，ユーザーはそれらをもとに，アプリケーションを開発することが可能である．AXI の基本的な通信プロトコルは valid 信号と ready 信号のハンドシェイクにより data 信号の送受信を制御するものである．典型的な通信の様子を図 2.4 に示す．図の黄緑の部分で通信が行われる．data 信号の送信側は，信号の送信準備ができた時に valid 信号をアサートする．受信側は，信号の受け入れ準備が出来た時に ready 信号をアサートする．valid 信号と ready 信号が同時にアサートされた時に通信が行われる．この際，valid 信号と ready 信号のアサートの順番は問わない．ready 信号のアサート条件に valid 信号の立ち上がりを含めてよいが，valid 信号のアサート条件に ready 信号の立ち上がりを含めることは禁止されている．これは，valid 信号と ready 信号がどちらもお互いの信号の立ち上がりを待機しているデッドロック状態の回避のためである．AXI4 には，用途に合わせて 3 つのプロトコル，AXI4，AXI4-Lite，AXI4-Stream が用意されている．

AXI4(フルスペック)

フルスペックの AXI4 はメモリマップ方式のインターフェイス用プロトコルであり，1 つのアドレスフェーズによって最大 256 データサイクルのバースト転送が可能である．データ幅は，8 bit，16 bit，32 bit，64 bit，128 bit，256 bit，1024 bit，2048 bit から選択可能である．マスターからスレーブの読み出しの様子を図 2.5 に，書き込みの様子を図 2.6 に示す．読み出しは，マスターが，読み出したいアドレスをスレーブに送信するための Read address channel (AR channel) と，スレーブが，受け取ったアドレスのデータをマスターに送信する Read data channel(R channel) の 2 チャンネルで行われる．書き込みは，マスターが書き込みたいアドレスをスレーブに送信する Write address channel (AW channel) と，マスターが書き込みたいデータをスレーブに送信する Write data channel (W channel)，および，書き込みを完了した信号をマスターに送るための

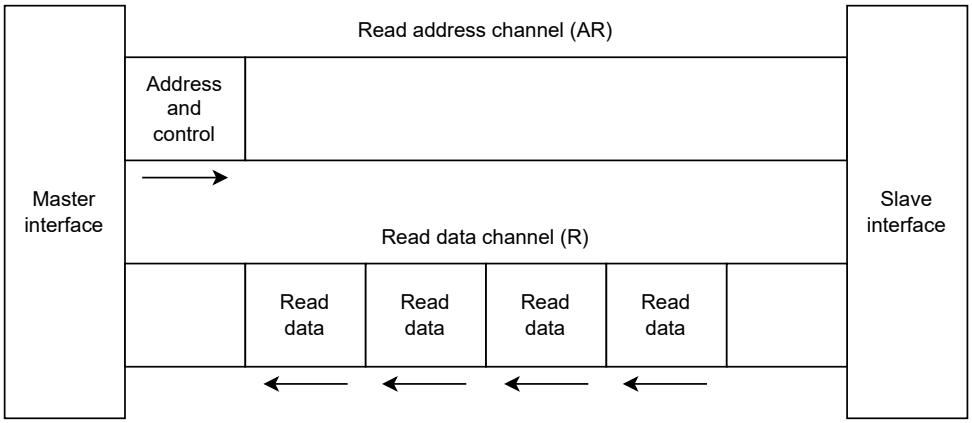

図 2.5　AXI 規格での，マスターからスレーブのデータの読み出し．

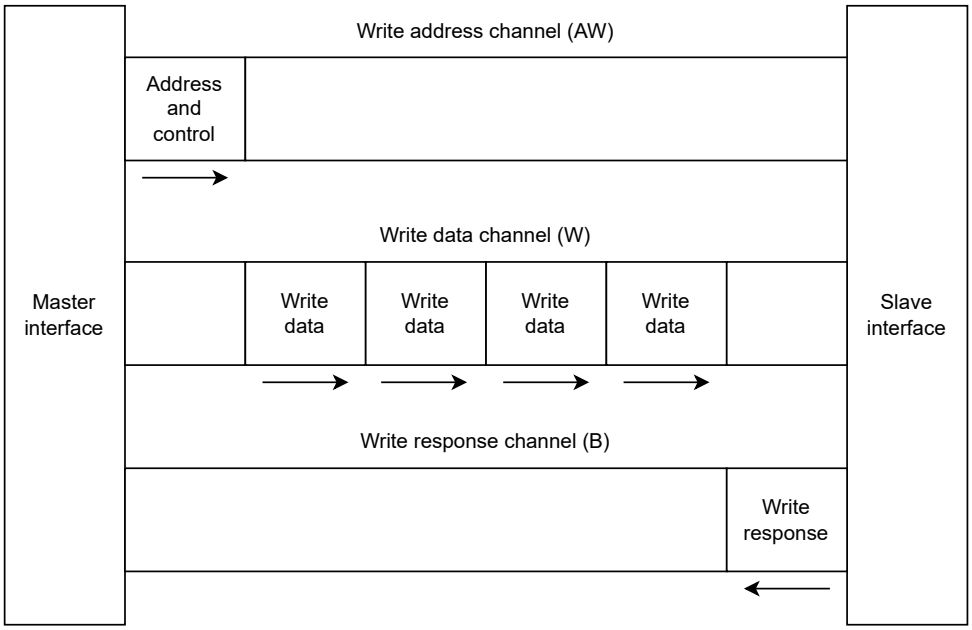

図 2.6　AXI 規格での，マスターからスレーブのデータの書き込み．

Write response channel(B channel) の 3 チャンネルで行われる．AR チャンネルおよび AW チャンネルでは，読み出したい/書き込みたいアドレスの他に，データ幅やバースト長の送信も行う．これらの 5 チャンネルは独立して存在するため，書き込みと読み出しは同時に行うことが可能である．各チャンネルの主要な信号を表 2.2 に示す．表に示した以外にも，オプションとして様々な信号が用意されている．Zynq インターコネクトでは，メモリ間の DMA や PS-PL 大規模通信などに用いられる．

AXI4-Lite

AXI4-Lite は，メモリマップ方式を用いた単一トランザクション専用の軽量インターフェイスである．ロジックのフットプリントが小さくすむため，設計時にも使用時にも扱いやすいシンプルなイン

表 2.2　AXI 規格の通信の各チャンネルに用いられる主要な信号

信号名	使用チャンネル	概要
valid	全チャンネル	データ，アドレスの有効性を示す． または制御信号がチャンネル上で利用可能であることを示す．
ready	全チャンネル	信号の受信準備が完了していることを示す．
addr	AW，AR	読み出し，書き込みを行うメモリ範囲の先頭アドレスの指定．
len	AW, AR	読み出し，書き込みのバースト長の指定．
size	AW, AR	読み出し，書き込みデータの byte 数の指定．
data	W，R	読み出し，書き込みを行うデータ．
strb	W，R	読み出し，書き込みデータのどの byte レーンが有効なデータかを示す．
last	W, R	読み出し，書き込みデータトランザクションの最終データであることを示す．
resp	B，R	読み出し，書き込みの成否を表す．0x00 が通信成功を示す．

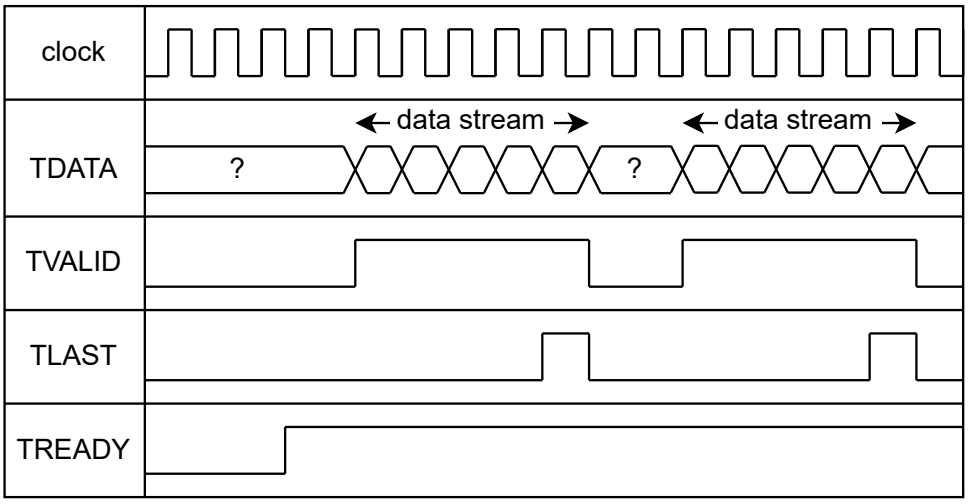

図 2.7　AXI-Stream プロトコルの典型的な信号．

ターフェイスとなっている．AXI4 との相違点は，バースト転送をサポートしていないという点のみである．Zynq インターコネクトでは，AXI_GP_S インターフェイスを使用し，PS から PL の制御レジスタを書き換えるときなどに使用する．

AXI4-Stream

AXI4-Stream は，アドレスの指定を必要とせず，データバースト長に制約のないプロトコルである．典型的な AXI4-Stream 信号を図 2.7 に示す．TDATA がデータ信号線である．TREADY 信号はスレーブのデータ受け入れ状況を示す．TVALID 信号でデータの有効性を示し，TLAST 信号でデータの区切りを表す．AXI4-Stream の通信にはバースト長の制限はないが，フルスペックの AXI4 との変換などの際に，AXI4 の最大バースト長である 256 words を超えるデータパケットを送信すると，オーバーフローになる可能性があるため注意が必要である．Zynq では PL 内部の IP コア間のデータ通信などに用いる．

第 3 章
Zynq を用いた FPGA ロジック監視システムの開発と性能評価

1.6 節で，理想的な FPGA ロジック監視システムの要件を述べた．この章では，監視システムを実際に Zynq を用いて構築する手法について述べる．理想的な監視システムの要件は，

1. 監視対象 FPGA ロジックの入力信号や出力信号を捕捉可能であること．
2. 捕捉した信号を用いて監視対象 FPGA ロジックが正常に作動しているか否かを判定すること．
3. 監視対象ロジックの状態をモニタリングルームへ伝える，もしくはモニタリングルームからロジックの回復を行うこと．
4. 監視システムが各々の FPGA に対して独立に稼働すること．

であったが，これらを Zynq で実装するためには，

A) PS-PL 間データ通信.
B) 監視対象 FPGA ロジックの正誤判定アプリケーション.
C) FPGA ロジック回復システム.
D) システムの全体制御のための Linux OS.

の 4 つの部分が必要になると考えられる．それぞれの部分を監視対象の FPGA ロジックに最適化することで，様々なロジックに対する監視システムの構築が可能となる．

A は，PS CPU からの PL FPGA ロジックの制御や，PL FPGA から PS DDR への大規模データ転送などに用いる．この開発と性能評価について 3.3 節で述べる．B は，システムの要件 2 そのものである．典型的な FPGA の異常に対する正誤判定の例について 3.4 節で述べる．C は，システムの要件 3 の一部である．3.5 節でその方法について述べる．D は，システム全体の制御のために必要である．3.2 節で，その開発について述べる．

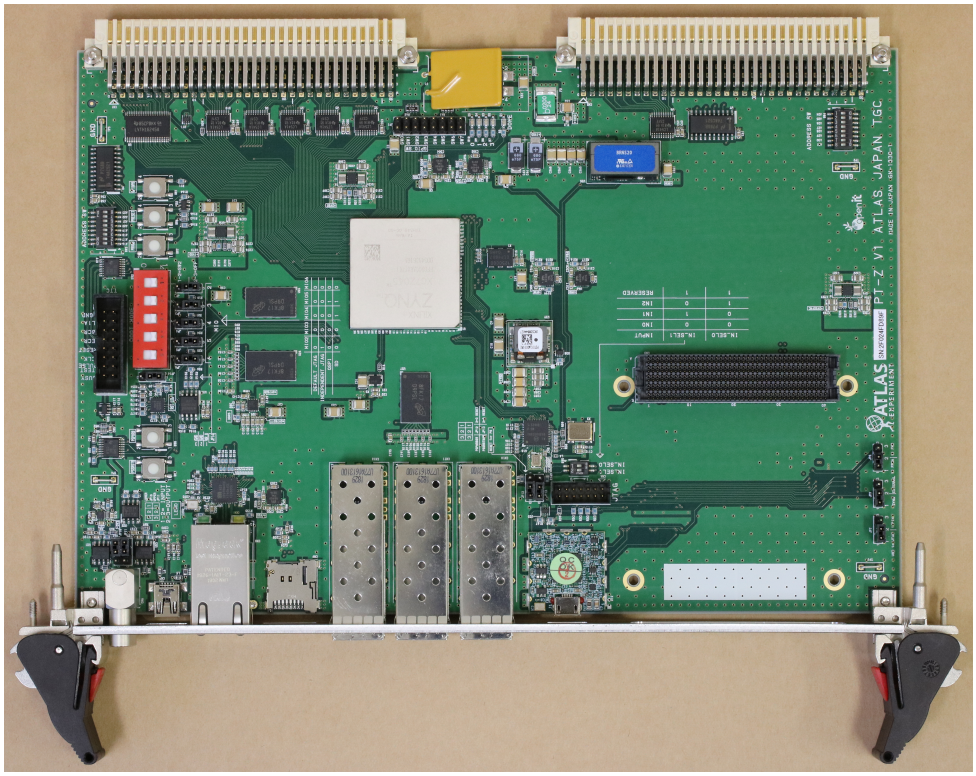

図 3.1　Zynq 搭載汎用 VME ボード PTZ[15].

3.1 システムの開発環境

Zynq を用いたシステムを構築するために，Zynq を搭載したボードを用意する必要がある．本研究では ATLAS 実験グループにより開発された PTZ を用いた (図 3.1) [14]．PTZ は，Prototype Zynq Board の略で Zynq の高エネルギー物理実験に対する有用性の確認のために開発された汎用の VME ボードであり，高エネルギー物理実験によく用いられる通信信号の入出力が可能となっている．また，大容量の DDR メモリを搭載しており，十分に CPU 性能が出せる構成となっている．

3.1.1 PTZ の機能と構成

XC7Z045-2FFG900I

ボードのメインチップとなる，Zynq-7000 本体である．PS 部の性能としては，プロセッサーコアとして Dual-core ARM Cortex-A9 MPCore^{TM} with CoreSight^{TM} を搭載．最大周波数 766 MHz．L1 キャッシュとして 32 KB，L2 キャッシュとして 521 KB，On-Chip Memory として 256 KB のメモリを搭載している．また，PL 部の性能としては，FPGA として，Xilinx Kintex-7 FPGA を搭載．Programmable Logic Cells が 350 K 個，LUT が 218,600 個，FF が 437,200 個，BRAM は 36 Kbit のものが 545 個で計 19.2 Mbit，DSP が 900 個搭載されいている．I/O は PS 部で 128 ポート，PL 部で HR I/O が 212 ポート，HP I/O が 150 ポート，高速光通信のための GTX ポートが

16 ポート搭載されている．

PS 周辺機器

- USB-UART

  USB-UART は，ホスト PC からの USB 接続による Zynq PS へのシリアル通信を介したアクセスのために設置されている．ビット転送速度は 115200 bps．USB コネクタは EDAC 社製 microUSB 690-005-299-043 を使用しており，規格は microUSB typeB である．USB-UART 変換 IC は Silicon Lab 社製 CP2103-GM を使用.

- Ethernet PHY / RJ45 コネクタ

  Ethernet PHY および RJ45 コネクタは，Zynq PS と外部ネットワークとの TCP/IP 通信のために設置されている．PS 部で動く Linux システムにより通信制御は容易に行うことが可能．スピードレートは 10/100/1000 Mbps モードで切り替え可能．コネクタは Bel Fuse 社製 RJ45 コネクタ 0826-1A1T-23-F．トランシーバー IC は MICROCHIP 社製 KSZ9031RNXCC を使用.

- microSD コネクタ

  microSD は Zynq PS のストレージ，Linux システムのルートファイルシステムの格納，電源投入と同時に Zynq のコンフィギュレーションを行うためのブートファイルの格納のために設置されている．コネクタは Molex 社製 5031821852．SD 規格の動作電圧は 3.3 V であるが，Zynq-7000 の PS の動作電圧は 1.8 V であるため，PTZ では MAXIM 社製レベルシフター MAX13035EETE+ を搭載し対応している.

- DDR3 メモリ

  DDR3 メモリは CPU を用いたアプリケーションのデータや関数の格納に用いる．Micron Technology 社製 MT41J128M16JT-125：KTR を 2 個搭載．合計容量 512 MB．データ転送速度 1600 MTps．PTZ ではデータ転送速度を最大限発揮させるために電源レギュレーターを搭載している.

PL 周辺機器

- GTX 対応 SFP ＋モジュール

  GTX 対応 SFP+ モジュールは，高速光通信に用いる．観測データなどの大規模なデータを実験室からモニタリングルームへ高速に送信することを想定している．GTX 対応 SFP+ 規格光ファイバートランシーバーモジュールを 3 台搭載．2 台は PL と，1 台は FMC コネクタを介してメザニンカードと通信する．コネクタは Molex 社製 074441-0010SFP+ 規格コネクタを使用．通信速度 10 Gbps で設計している．GTX 通信には，“綺麗な” クロックが必要であるため，PTZ にはクロックジッタークリーナを搭載している．クロックジッタークリーナ

は Silicon Labs 社製クロック調整デバイス Si5344 を搭載．Si5344 は I^2C 通信で PS に接続されており，内部レジスタを書き換えることでコンフィギュレーションが可能.

- JTAG コネクタ
  JTAG コネクタは，PL FPGA のコンフィギュレーションや内部ロジックアナライザを用いた FPGA デバッグに用いる. コネクタは Molex 社製 0878311420 を搭載.

- microUSB-JTAG 変換コネクタ
  USB 接続による，PL FPGA のコンフィギュレーションや内部ロジックアナライザを用いた FPGA デバッグに用いる．microUSB コネクタと変換 IC が一体となった Digilent 社製 210-251 を搭載.

- VME バスコネクタ
  3 V の電源供給に用いる．VME バスを通じた他ボードとの通信にも用いることが可能.

- NIM コネクタ
  同軸ケーブルによる通信に用いる．2 ポートでクロック，データの入出力を想定している．入力，出力は切り替え可能である．変換器を用いて，NIM レベルの信号を Zynq PL で使用可能な LVCOM レベルへ変換する．NIM コネクタとして，LEMO 社製 EPL.00.250.NTN を搭載．信号変換器として Texas Instruments 社製 SN65LVDS348PW を搭載.

- 40.08 MHz 水晶クロック
  PL FPGA 内部クロックおよび，Si5344 のクロックソースとして使用．ALTAS 実験の衝突周波数に対応している.

3.1.2 ボードの各機能のテストおよび性能評価

UART シリアル通信

Zynq の UART 通信は 115200 bps である．Zynq にシリアル通信を介して接続したホスト PC 上のターミナルの通信レートを 115200 bps に設定し，”Hello, World!”と表示させるプログラムを Zynq PS 上で実行した．コンソール上に文字が表示され，通信を確認した.

NIM 通信

NIM コネクタは IO-0 と IO-1 の 2 つが搭載されている．まず，出力テストについて述べる．内部水晶クロック (40.08 MHz) の出力を I0-0 と IO-1 に接続するロジックを作成，テストを行った．テスト結果を図 3.2 に示す．40 MHz のクロック信号の出力を確認した．次に，入力テストについて述べる．IO-1 の入力は Zynq PL に接続されているが，IO-0 の入力は，Si5344 にクロックソースとし

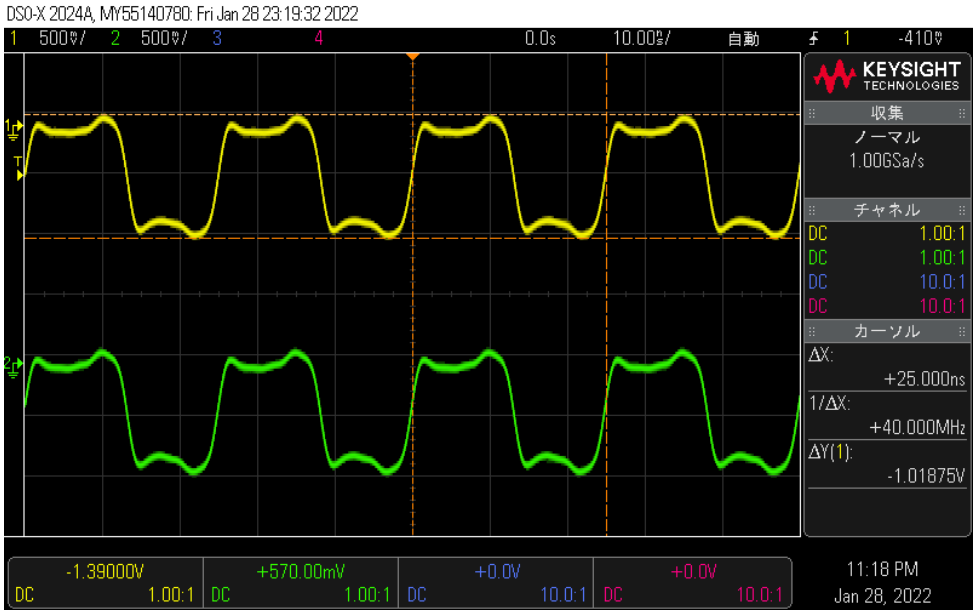

図 3.2　NIM コネクタの出力テスト結果．黄色が IO-0，緑が IO-1 の出力である．

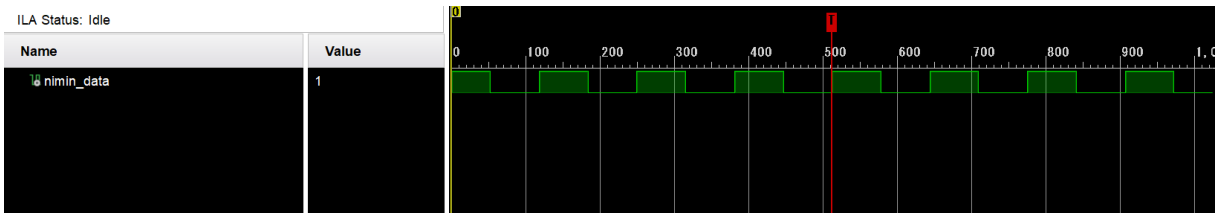

図 3.3　NIM コネクタの入力テスト結果．

て接続されている．IO-0 の入力テストに関しては，後述の GTX 通信試験の節で述べる．IO-1 の入力について確認を行う．パルス発生器で出力した 300 kHz クロック信号を IO-1 に入力し，内部ロジックアナライザで確認した．テスト結果を図 3.3 に示す．正しく 300 kHz のクロック信号を入力できていることを確認した．

Ethernet 通信

PTZ に搭載されている Ethernet PHY は 10/100/1000 Mbps モードでの通信が可能である．1000 Mbps モードについて，Zynq PS 上で作動する Linux OS を用いて通信試験を行った．PTZ と PC をネットワークスイッチを介して接続し，PTZ をホスト，PC をクライアントとして iperf3 コマンドで TCP テストを行った．試験結果を表 3.1 に示す．理論値の 70% 程度の通信性能を達成していることを確認した．これは，Linux OS で Ethernet PHY を制御するにあたり，妥当な数値といえる．

表 3.1　Ethernet 通信テスト結果

1 回目	2 回目	3 回目	4 回目	5 回目
707 Mbps	703 Mbps	697 Mbps	706 Mbps	702 Mbps

表 3.2　GTX 通信 IBERT テスト結果

通信レート	0.8 Gbps	1.6 Gpbs	3.2 Gbps
内部ループバック	$< 2.4 \times 10^{-12}$	$< 4.7 \times 10^{-13}$	$< 6.7 \times 10^{-14}$
GTX Channel 1	$< 2.4 \times 10^{-12}$	$< 4.7 \times 10^{-13}$	$< 6.7 \times 10^{-14}$
GTX Channel 2	$< 2.4 \times 10^{-12}$	$< 4.7 \times 10^{-13}$	$< 6.7 \times 10^{-14}$
通信レート	6.4 Gbps	8.0 Gbps	10.24 Gbps
内部ループバック	$< 4.1 \times 10^{-15}$	$< 7.3 \times 10^{-14}$	$< 1.7 \times 10^{-14}$
GTX Channel 1	$< 4.1 \times 10^{-15}$	$< 7.3 \times 10^{-14}$	$< 1.7 \times 10^{-14}$
GTX Channel 2	$< 4.1 \times 10^{-15}$	$< 7.3 \times 10^{-14}$	$< 1.7 \times 10^{-14}$

I^2C 通信

I^2C 通信とは，周辺機器の設定のための低速シリアル通信である．GTX 通信のリファレンスクロック作成のためのデバイス，Si5344 は I^2C 通信で PS と接続されている．Si5344 は内部レジスタを持っており，I^2C 通信により，その値を書き換えることでクロックの値や，クロックソースの設定を行うことが出来る．なお，内部レジスタの設定値は Clock Builder Pro というソフトウェアで発行することが出来る．Clock Builder Pro で発行した値を I^2C で Si5344 に書き込むベアメタルアプリケーションを作成し，実行したところ，Si5344 は正しく設定され，設定した値のクロックの出力を確認できた．これにより，I^2C 通信の確認もできた．

GTX 通信

GTX とは，10 Gbps を超える非常に高速な通信規格である．内部ループバックおよび外部ループバックでの Integrated Bit Error Ratio Test(IBERT) 試験を様々な通信レートに対し行った．ボード外部から NIM コネクタを用いて入力した 40 MHz クロックをクロックソースとして，Si5344 で作成した 160 MHz クロックをリファレンスクロックとして用いた．bit error ratio の上限値を表 3.2 に示す．また，通信の eye パターンを図 3.4 に示す．eye パターンとは，通信信号の立ち上がりの鋭さや，ジッターの少なさを視覚的に表現し，通信のクオリティを示すものである．図の中央の “eye” が開いているほど通信のクオリティが高いと言える．1.6 Gbps にくらべ 10.24 Gbps の eye パターンは eye の開きが小さいが，エラーレートが極めて低い領域は十分に存在するため，これは十分クオリティの高い通信であると言える．10 Gbps 以上の高速光通信が可能であることを確認した．

3.2 Zynq PS 上で作動する Linux OS の構築

システム全体の制御やユーザーアプリケーションの実行を行う Linux Operating System(OS) を Zynq PS 上に構築した．Zynq PS 上で Linux OS を動作させるために必要なコンポーネントは，

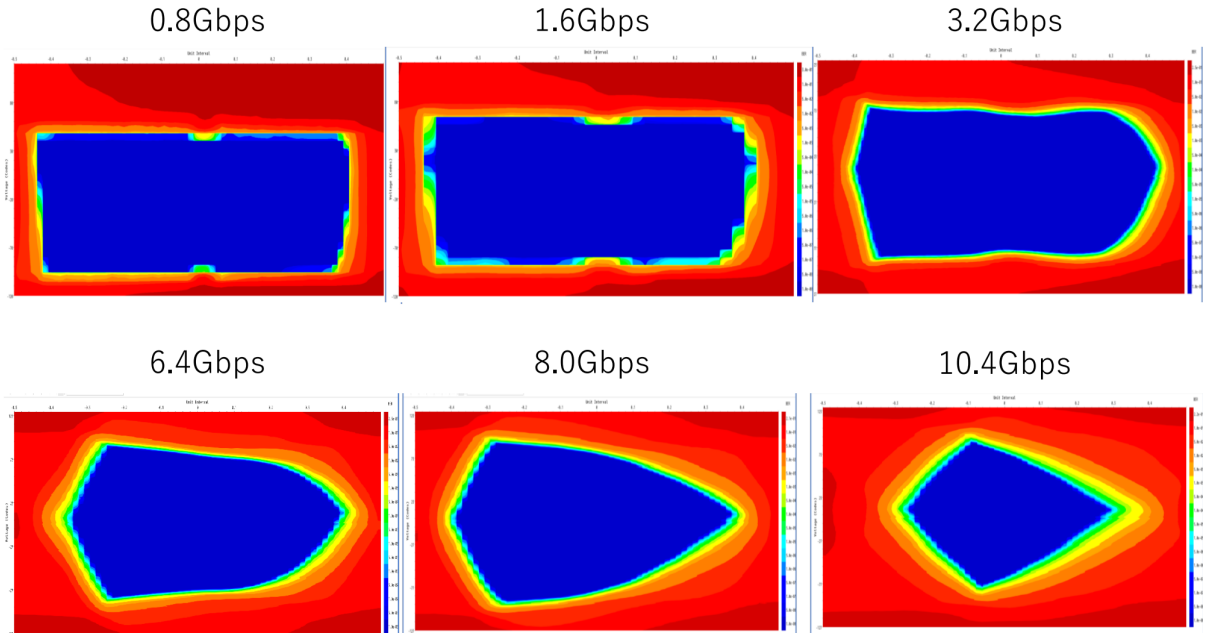

図 3.4　GTX の IBERT 試験で得られた eye パターン．

1. **Linux kernel**
   Linux OS の主要コンポーネント．システムのハードウェアレベルでのデータ処理とソフトウェアプロセスの間を取り持ち，可能な限り効率的にリソースを管理する．device driver や kernel module を使用して，ハードウェアの制御を行う．
2. **boot loader**
   Linux OS の起動に必要なコンポーネント．Linux kernel の起動に必要なハードウェアの初期化を行い，kernel image ファイルのロードを行う．また，ビットストリームファイルも含めることが出来，Zynq PL FPGA のコンフィギュレーションもここで行うことが出来る．
3. **Device-tree**
   ボードのハードウェア情報を記述するデータ構造体．ここに記述したハードウェアを，Linux kernel で利用することが出来る．ハードウェアの動作クロックや，制御レジスタの値，I/O，使用する device driver などを記述する．
4. **rootfs**
   Linux OS のルートディレクトリにマウントされるファイル構造．主要なコマンドファイルやライブラリの種類はこの構造に依存する．例として，Ubuntu や CentOS などのディストリビューションがある．

の 4 つである．これらを SD カードに入れ，Zynq を SD カードブートモードで起動することで，Zynq Linux を起動することが出来る．以上のコンポーネントについて，開発を行った．

表 3.3　kernel image に含めた module，device driver

コンフィギュレーション項目	用途
CONFIG MICREL PHY	PTZ に搭載された Ethernet PHY の制御
CONFIG FPGA REGION	
CONFIG FPGA BRIDGE	
CONFIG OF OVERLAY	Device-tree overlay を利用した
CONFIG OF CONFIGFS	FPGA 再コンフィギュレーション
CONFIG CMA	
CONFIG DMA CMA	

3.2.1 Petalinux

Zynq PS の CPU アーキテクチャと，Zynq Linux コンポーネントの開発環境の CPU アーキテクチャは一般に異なるため，通常のコンパイラでは Zynq Linux コンポーネントのビルドを行うことはできず，クロスコンパイラを用いる必要がある．クロスコンパイラとは，ある CPU 上で，異なるアーキテクチャの CPU で動作するアプリケーションをコンパイルすることのできるコンパイラである．Xilinx 社は Zynq Linux コンポーネント開発のために，Petalinux というクロスコンパイラを含む開発環境を用意している．Petalinux は，同じく Xilinx 社製の Vivado を用いて作成したハードウェア記述ファイルを読み込み，そのハードウェアに適した Linux kernel および boot loader のコンフィギュレーションファイルの作成，Device-tree の作成，Linux kernel，boot loader のビルド，rootfs の構築，kernel module のコンパイルなどが可能なツール群である．本研究では，Linux kernel および boot loader のビルドについて Petalinux を使用した．

3.2.2 Linux kernel

Linux kernel の開発は Petalinux で行った．Linux kernel のビルドに伴って，kernel image に含めたい kernel module や device driver についてコンフィギュレーションファイルに記述する必要がある．今回開発した kernel image に含めた kernel module，device driver の中で，本研究で開発した FPGA 監視システムの実現に対して特徴的なものを表 3.3 に示す．これらの項目をコンフィギュレーションファイルに記述し，ビルドを行い，kernel image ファイル image.ub を作成した．

3.2.3 boot loader

boot loader の開発は Petalinux で行った．デフォルトでは，SD カード内にある，rootfs.cpio.gz ファイルを展開してルートファイルシステムを構築する設定となっているが，本研究では，rootfs に Ubuntu.18.04 を使用したため，SD カード内の別パーティションに存在するルートファイルシステムを読み込むように設定した．ビルドを行い，boot loader ファイルを作成した．

```
U-Boot 2021.01 (Oct 12 2021 - 09:28:42 +0000)

CPU:   Zynq 7z045
Silicon: v3.1
DRAM:  ECC disabled 512 MiB
Flash: 0 Bytes
NAND:  0 MiB
MMC:   mmc@e0100000: 0
Loading Environment from FAT... *** Warning - bad CRC, using default environment

In:    serial@e0001000
Out:   serial@e0001000
Err:   serial@e0001000
Net:
ZYNQ GEM: e000b000, mdio bus e000b000, phyaddr -1, interface rgmii-id
eth0: ethernet@e000b000
Hit any key to stop autoboot:  0
switch to partitions #0, OK
mmc0 is current device
Scanning mmc 0:1...
Found U-Boot script /boot.scr
2710 bytes read in 15 ms (175.8 KiB/s)
## Executing script at 03000000
Trying to load boot images from mmc0
```

```
[  OK  ] Reached target Host and Network Name Lookups.
[  OK  ] Reached target Network.
         Starting Permit User Sessions...
         Starting Daily apt download activities...
         Starting OpenBSD Secure Shell server...
[  OK  ] Started Permit User Sessions.
         Starting Set console scheme...
[  OK  ] Started Serial Getty on ttyPS0.
[  OK  ] Started Set console scheme.
[  OK  ] Created slice system-getty.slice.
[  OK  ] Started Getty on tty1.
[  OK  ] Reached target Login Prompts.
[  OK  ] Started OpenBSD Secure Shell server.
[  OK  ] Started Dispatcher daemon for systemd-networkd.
[  OK  ] Reached target Multi-User System.
[  OK  ] Reached target Graphical Interface.
         Starting Update UTMP about System Runlevel Changes...
[  OK  ] Started Update UTMP about System Runlevel Changes.
         Starting Hostname Service...
[  OK  ] Started Hostname Service.

Ubuntu 18.04 LTS kosuke-OptiPlex-3020M ttyPS0

kosuke-OptiPlex-3020M login:
```

図 3.5　PTZ の Linux OS の起動ログ．左図が起動開始，右図が起動完了した際のログである．左図のログを見ると，Zynq 7z045 を CPU として正しく認識しており，また DDR も 512 MB とただしく認識していることがわかる．右図を見ると，Ubuntu18.04 が正しく起動していることがわかる．

3.2.4 Device-tree

Device-tree の開発は Petalinux で行った．AXI_DMA_IP の制御用の kernel module のための記述と，Zynq Linux を起動したまま Linux から FPGA を再コンフィギュレーションのための記述を Petalinux で作成したファイルに追記した．

3.2.5 rootfs

rootfs に関しては，前述の通り，Ubuntu.18.04 を使用した．ビルドのために，開発環境である Windows マシンの WSL 上で起動する Ubuntu にクロスコンパイラ，gcc-arm-none-eabi パッケージをインストールした．Ubuntu rootfs を SD カードにインストールするために，まず SD カードを 2 つのパーティション，BOOT と ROOT に分割し，それぞれを FAT32，EXT4 でフォーマットした．ROOT パーティションに rootfs を展開し，chroot コマンドで展開した rootfs に仮想的にログインすることに成功した．これにより，Zynq 上で Linux OS を起動せずとも，ホスト PC 上で Linux アプリケーションの開発や，パッケージの追加が可能となった．

3.2.6 Linux OS の起動確認

作成したコンポーネントを用いて Linux OS の起動確認を行った．前述の通り 2 つのパーティションに分割し，BOOT パーティションに，kernel Image ファイル，boot loader ファイル，Device-tree を格納し，ROOT パーティションに rootfs を展開した SD カードを PTZ に挿入し，SD カードブートモードで起動した．UART 通信を介してホスト PC のターミナルで Linux OS の起動ログを確認した様子が図 3.5 である．Linux OS の正常な起動を確認した．また，PTZ を Ethernet 通信を介し，ホスト PC が接続されているネットワークに接続し，ssh ログインの確認を行ったところ，正常なログインを確認した (図 3.6)．

```
sueda@LAPTOP-JUN5JMEE:~$ ssh -X sueda@10.37.1.196
sueda@10.37.1.196's password:
Welcome to Ubuntu 18.04 LTS (GNU/Linux 5.10.0-xilinx-v2021.2 armv7l)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Last login: Wed Feb  2 13:56:31 2022
sueda@PTZ-Linux:~$
```

図 3.6　PTZ の Linux OS に ssh ログインした様子．LAPTOP PC から，PTZ-Linux に user:sueda として，正しくログインできていることがわかる．

3.3 PS-PL 間通信システムの開発と性能評価

第 2 章で述べた通り，Zynq の PS と PL は様々なインターコネクト，インターフェイスによって，密接に接続されている．本システムにおける PS-PL 間通信の目的は，

1. PS CPU を用いた PL FPGA ロジックの制御.
2. PS-PL 間の大規模なデータの送受信.

の二つある．それぞれを実現するシステムの開発を行った．

3.3.1 PS CPU を用いた PL FPGA ロジックの制御

PS CPU を用いた PL FPGA ロジックの制御は，PS CPU から PL FPGA ロジックの内部レジスタへのアクセスにより実現される．PS CPU から PL FPGA の制御を行う際，高いスループットは必要ないため，最もシンプルな方法，つまり，マスターが CPU，スレーブが AXI_GP_S インターフェイスの AXI4-Lite プロトコルでの通信を用いる．(通信プロトコルの詳細は 2.4 節を参照．) 通信経路および使用するインターフェイス，インターコネクトを図 3.7 に示す．CPU から送信されたデータは，以下のように PL FPGA ロジックのレジスタに書き込まれる．

1. APU 部で，CPU からスヌープ制御ユニットを介して L2 キャッシュに書き込まれる．
2. PS インターコネクト部で，L2 キャッシュから Master Interconnect を介して AXI_GP_S インターフェイスに到達する．
3. PL 部で，AXI_GP_S インターフェイスから AXI スレーブポートを持った PL FPGA ロジックのレジスタに書き込まれる．

この方法でのスループットの理論値は 25 MB/s 程度となっている [11].

通信システムの実装とテスト

PL FPGA ロジックの内部レジスタを適切なシステムアドレスにマッピングし，その情報を Device-tree に記述することで，Zynq Linux でデバイスとして認識することが可能となり，Linux アプリケーションによる制御が可能となる．通信テストのために，1bit の内部制御レジスタを持つ，

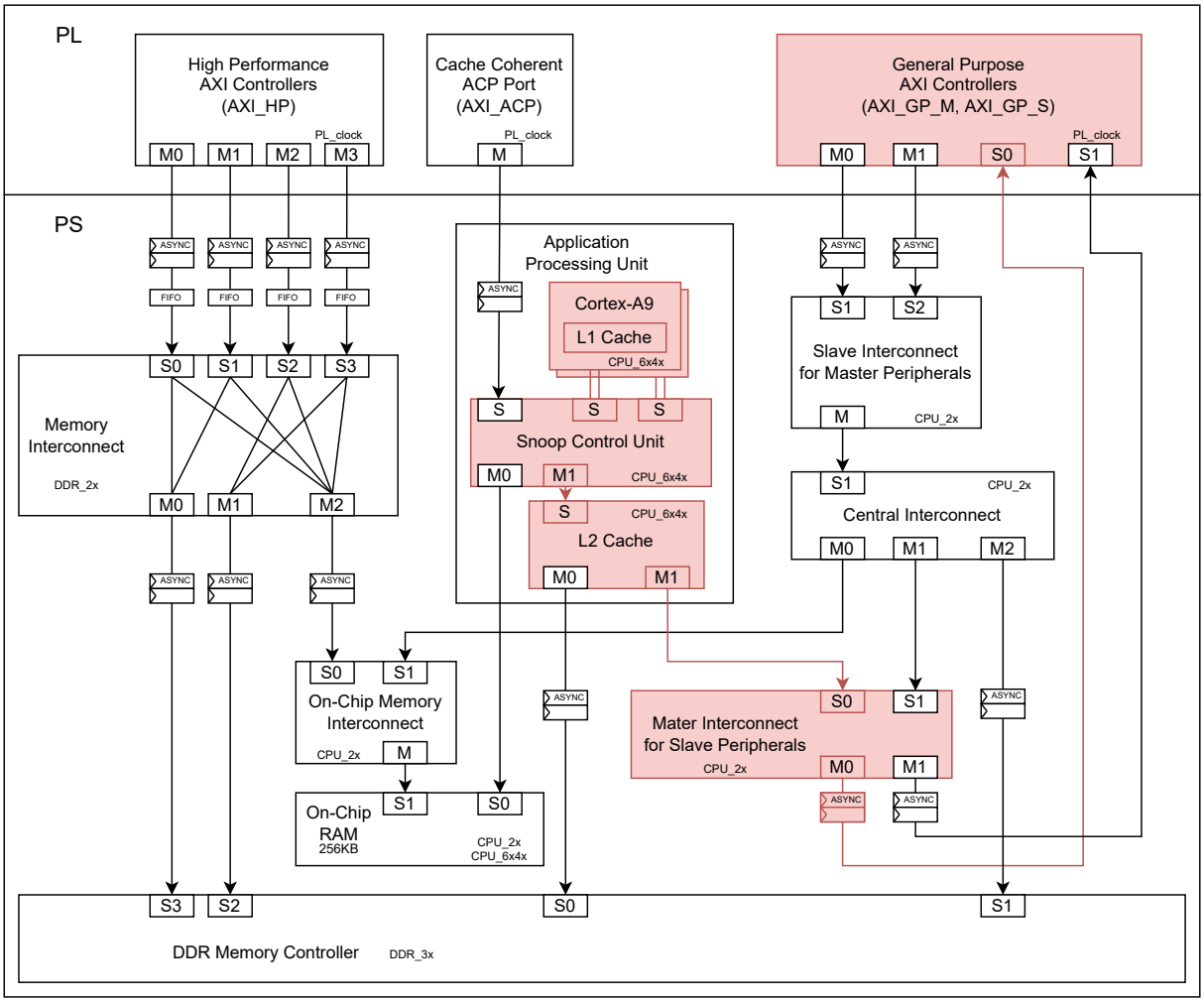

図 3.7　CPU から PL FPGA ロジックのレジスタへのアクセスの通信経路および使用インターフェイス．図中の赤で示した箇所が通信経路および使用インターフェイスである．通信プロトコルは AXI4 Lite．図中の M は AXI4 通信のマスター，S はスレーブを表す．

LED 制御ロジックを開発した．レジスタの値が 1 の時 LED が点灯し，0 の時消灯するものとした．1 秒おきに LED が点滅する Linux アプリケーションを作成し，実行したところ，LED の点滅を確認できた．テストに用いたシステムのブロック図を図 3.8 に示す．

3.3.2 PS-PL 間の大規模データ通信

PS-PL 間通信で最大のスループットを期待できる通信方法は，AXI_HP インターフェイスを使用した，PL から PS への Direct Memory Access(DMA) である．使用するメモリインターフェイスの選択によって，複数の通信経路が考えられるが，数 100 MB を超えるサイズのデータを送ることを想定し，PS DDR への DMA 通信について考える．通信経路および使用するインターフェイス，インターコネクトを図 3.9 に示す．データは PL FPGA に設置された DMA コントローラにより，PL の AXI_HP インターフェイスを用いて，メモリーインターコネクトを介し，PS DDR に書き込まれ

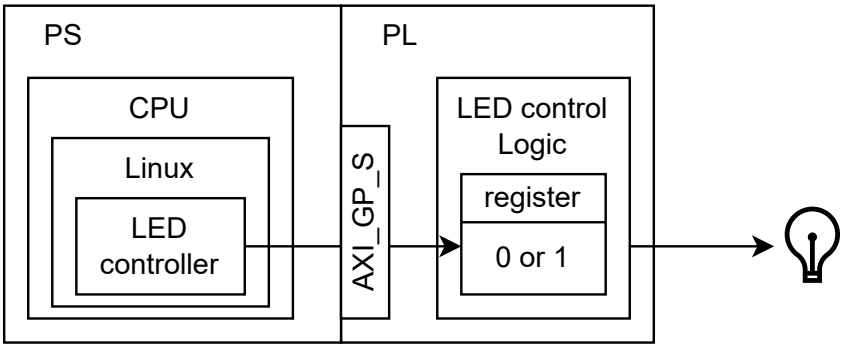

図 3.8　AXI_GP_S インターフェイスを使用した LED 制御システム．

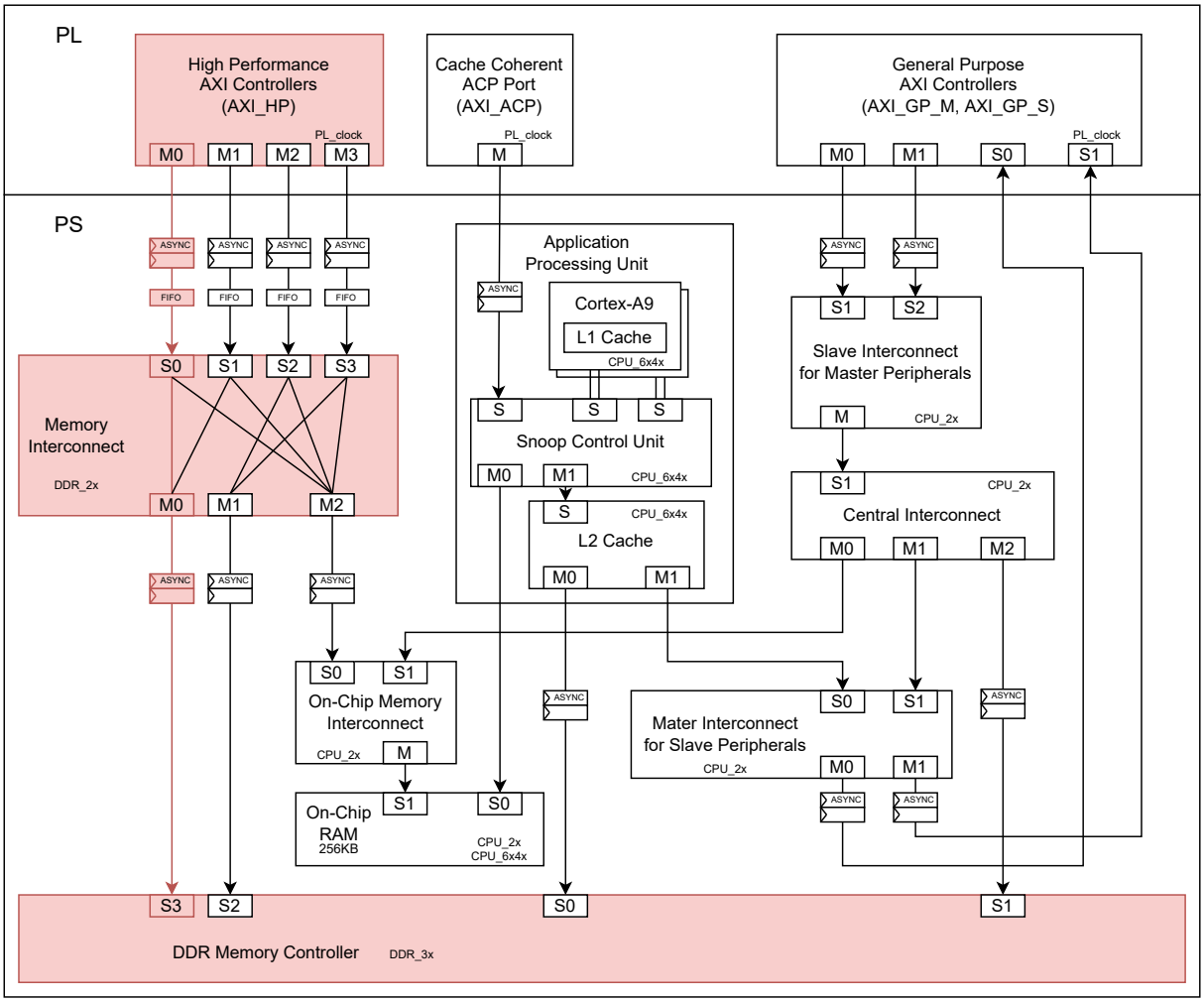

図 3.9　AXI_HP インターフェイスを使用した PS DDR アクセスの通信経路．図中の赤で示した箇所が通信経路および使用インターフェイスである．通信プロトコルは AXI4.

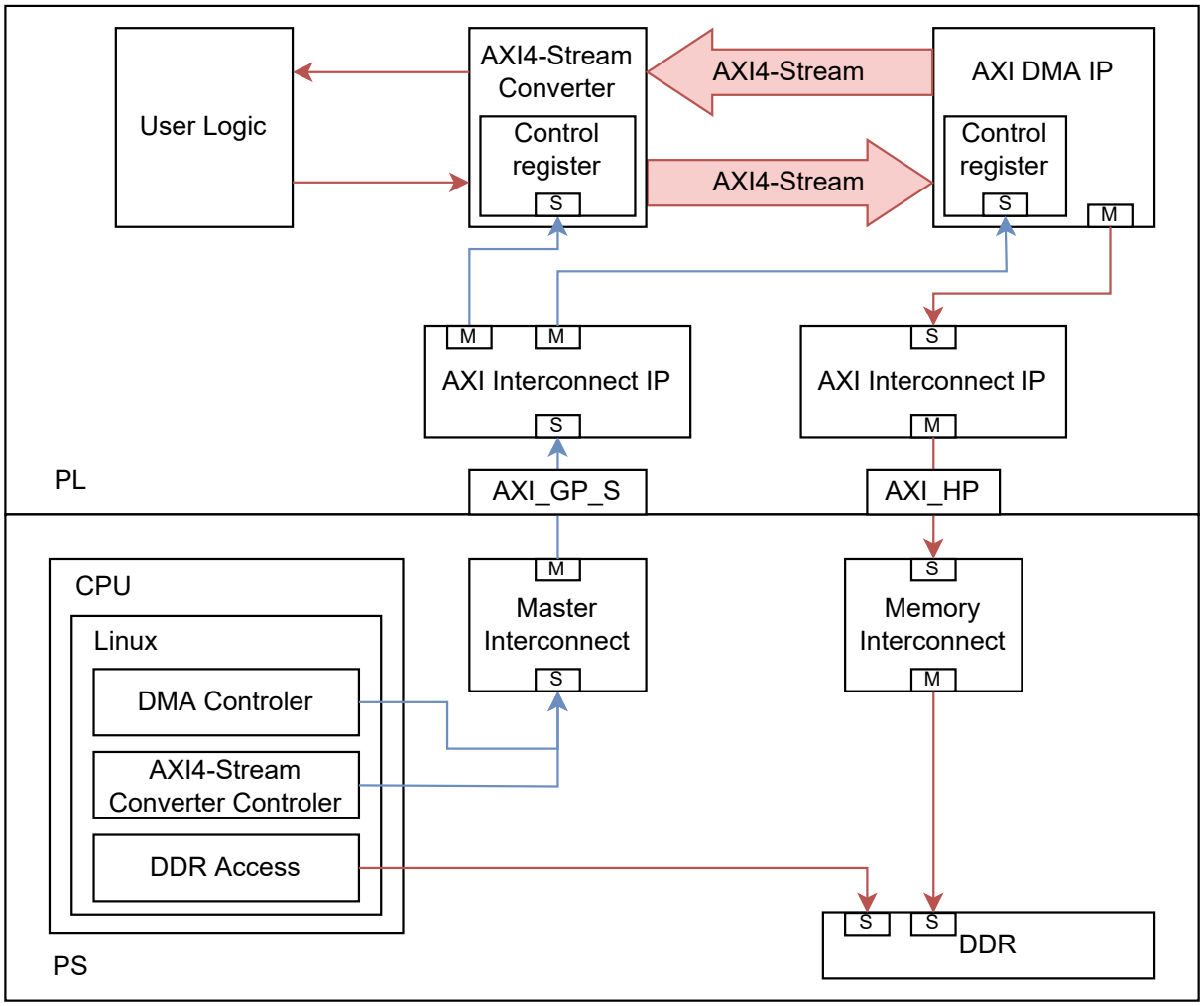

図 3.10　PL FPGA ロジックから PS DDR への DMA を実現する通信システム．赤がデータの経路を，青が制御の経路を示している．

る．また，同じ経路により読み出される．DMA コントローラは内部に制御レジスタを有しており，3.3.1 項に示した方法で PS CPU からの制御が可能である．

PL FPGA ロジックから PS DDR への DMA

PL FPGA ロジックから PS DDR への DMA を行うための通信システムを開発した．通信システムのブロック図を図 3.10 に示す．図中の右上にあるブロックが，システムの中核となる，AXI DMA IP コアである．AXI DMA IP コアは，Xilinx 社製 IP コアで，ユーザーロジックとメモリインターフェイス間の DMA を可能にする．ユーザーロジック側の通信プロトコルは AXI4-Stream，メモリーインターフェイス側の通信プロトコルはフルスペックの AXI4 である．よって，この IP コアを使用するためには，ユーザーロジックに AXI4-Stream プロトコルでの送受信を行うことのできる通信ポートを設置する必要がある．しかし，ある FPGA ロジックを監視するために，ロジックの内容を変更し AXI4-Stream 用のポートを追加することは，監視システムとして汎用性に

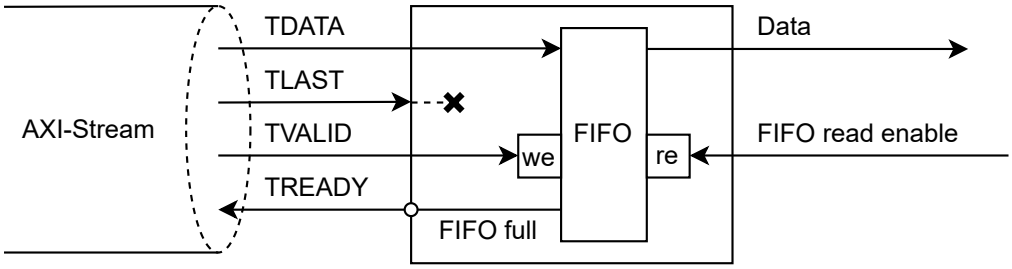

図 3.11　AXI4-Stream 信号をユーザーロジックで使用できる形式へ変換するロジック．

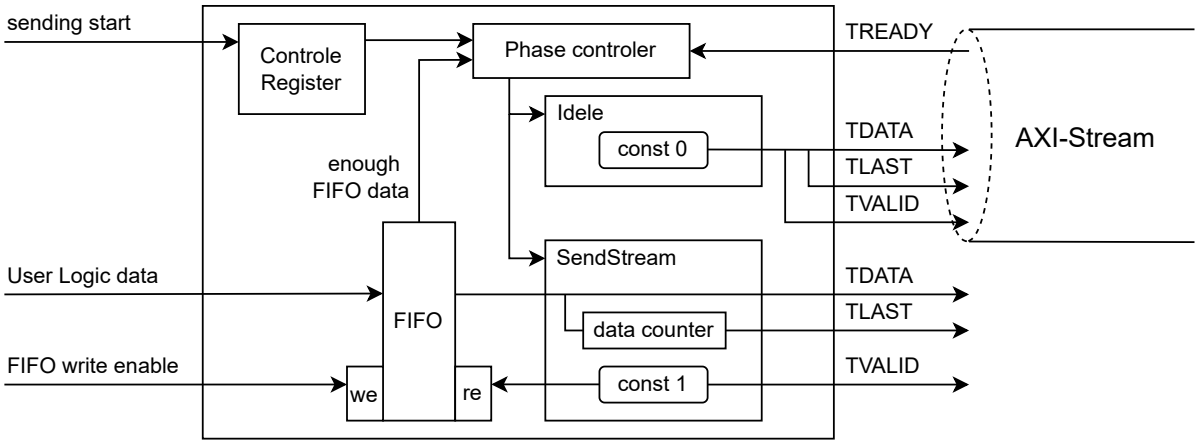

図 3.12　ユーザーロジックの出力信号を AXI4-Stream 信号へ変換するロジック．

欠ける．この問題の解決のために，AXI4-Stream 形式の信号を汎用ユーザーロジックで送受信するための AXI4-Stream コンバータを開発した．詳細な仕様については後述する．AXI DMA IP コアおよび AXI4-Stream コンバータは内部に制御レジスタを搭載しており，PS CPU から制御可能である．AXI Interconnect IP コアは，AXI4 プロトコル通信のためのバス機能をもった IP コアである．PS 側の実装としては，Linux アプリケーションとして，AXI DMA IP コントローラ，AXI4-Stream コンバータコントローラ，PS DDR へのアクセスアプリケーションを開発した．

AXI4-Stream コンバータの開発

AXI4-Stream 信号をユーザーロジックで使用できる形式へ変換するロジックおよび，ユーザーロジックの出力信号を AXI4-Stream 信号へ変換するロジックを開発した．概要図を図 3.11 および図 3.12 に示す．

まず，AXI4-Stream 信号をユーザーロジックで使用できる形式へ変換するロジックについて述べる．ロジックの動作を以下に示す．

- AXI4-Stream 信号のデータが有効であることを示す TVALID 信号がアサートされた時，TDATA 信号を内部の FIFO に詰める．
- AXI4-Stream のスレーブが受け入れ可能であることを示す TREADY 信号を，FIFO full がディアサートされている時にアサートし，FIFO full がアサートしてるときにディアサート

する．

- ユーザーロジックからのデータ読み出し信号がアサートされた時，FIFO からデータを取り出し，ユーザーロジックへ出力する．

なお，AXI-Stream 信号のデータパケットの末尾でアサートされる TLAST 信号は，本ロジックでは使用しない．

次にユーザーロジックが出力したデータを AXI4-Stream 信号へ変換するロジックについて述べる．ロジックは，2 つのフェーズで構成されている．その遷移の制御を，内部レジスタを搭載したフェーズコントローラが行う．各フェーズのロジックの動作および，遷移条件を以下に示す．

- **IDLE 状態**：出力する AXI4-Steam 信号の TDATA 信号，TLAST 信号，TVALID 信号をディアサートする．
  - ・FIFO に十分量のデータが存在する．
  - ・スレーブの AXI DMA IP コアがデータ受け入れ可能状態となったことを示す TREADY 信号がアサートされている．

  上記の状態で, 制御レジスタの通信開始ビットが PS CPU からアサートされた時, SendStream 状態へ移行する．
- **SendStream 状態**：FIFO からデータを取り出し，TDATA 信号として出力する．また，同時に TVALID 信号をアサートする．出力したデータの数をデータカウンターで数え，指定した数のデータを出力した際，TLAST 信号をアサートし，IDLE 状態へ移行する．

以上 2 つのロジックの開発により，AXI-Steream 形式の信号の送受信を，汎用の FIFO への書き込み/読み出しのように扱うことが可能となった．また，開発した AXI4-Stream コンバータは内部パラメータとして，データ幅を 8 bit，16 bit，32 bit，64 bit，128 bit，256 bit，512 bit，1024 bit から，バースト長を 1，2，4，8，16，32，64，128，256 から選択可能な IP コアパッケージングし，汎用性を高めた．

AXI DMA IP コントローラの開発

AXI DMA IP コアの制御には，PS CPU 上で動作する Linux OS を使用する．Linux アプリケーションから AXI DMA IP をデバイスとして使用するために，kernel module の開発を行った．kernel module とは，Linux kernel によるデバイスの制御関数などをまとめたもので，動的に Linux OS にロードすることが出来る device driver である．開発した kernel module には，汎用のシステムコールで AXI DMA IP の初期化や，通信の制御を行うことのできる機能を実装した．AXI DMA IP の制御シークエンスは以下の通りである．なお，PL FPGA ロジックから PS DDR へのデータ転送を送信，PS DDR から PL FPGA ロジックへのデータ転送を受信と定義した．

初期化シークエンス

1. **AXI DMA IP コアのリセット．**
   AXI DMA IP の制御レジスタのリセットビットをアサートする．AXI DMA IP の全ての内

部レジスタのビットが初期化される．kernel module のロード時に実行される．

データ送信シークエンス

1. **送信用 DMA チャンネルの開始．**
   送信制御レジスタの Run Start ビットをアサートする．送信チャンネルが開始され，送信チャンネルの停止状況を示すステータスレジスタの Halted ビットがディアサートされる．Linux アプリケーション内で，送信モジュールを “open” した時に実行される．
2. **送信アドレスの送信．**
   送信アドレスレジスタに，PS DDR のデータを書き込みたい領域の先頭アドレスを書き込む．
3. **データ長の送信．**
   送信バッファー長レジスタに送信するデータのサイズを書き込む．この際，実際に送信するデータサイズと送信バッファー長レジスタに書き込む値は必ず一致していなければならない．仮に，ここで書き込んだ値より送信データが小さければ，busy が解除されず，送信データが大きければ，エラーでチャンネルが強制停止する．
4. **PL のデータ送信開始．**
   AXI4-Stream コンバータの送信開始ビットをアサートし，データの送信を開始する．
5. **Busy 待機．**
   PS DDR への書き込みが完了した際，送信チャンネルのステータスレジスタの Halted ビットがアサートされ，送信チャンネルの停止が示される．このビットの立ち上がりの確認後，通信を終了する．2-5 はシステムコール “write” で実行される．

データ受信シークエンス

1. **受信用 DMA チャンネルの開始．**
   受信制御レジスタの Run Start ビットをアサートする．受信チャンネルが開始され，受信チャンネルの停止状況を示すステータスレジスタの Halted ビッがディアサートされる．Linux アプリケーション内で，受信モジュールを “open” した時に実行される．
2. **受信アドレスの送信．**
   ソースアドレスレジスタに，PS DDR のデータを読み出したい領域の先頭アドレスを書き込む．
3. **データ長の送信．**
   受信バッファー長レジスタに読み出すデータのサイズを書き込む．レジスタに値を書き込むと同時に通信がスタートされる，
4. **Busy 待機．**
   AXI_DMA_IP でのデータの受信が完了した際，受信チャンネルのステータスレジスタの Halted ビットがアサートされ，受信チャンネルの停止が示される．このビットの立ち上がりの確認後通信を終了する．2-4 はシステムコール “read” で実行される．

上記の手順で AXI DMA IP の制御を行う Linux kernel module を開発した．データ送信シークエンスおよび，データ受信シークエンスを実行した際の PL FPGA の信号をロジックアナライザで確

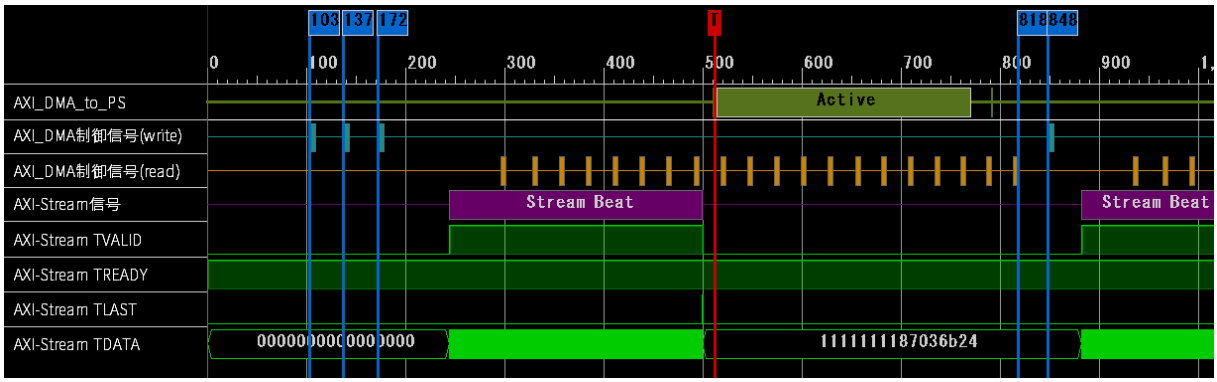

図 3.13　PL Logic から PS DDR への書き込みをロジックアナライザで確認した様子．103 clock 目が AXI DMA IP のリセット信号, 137 clock 目が送信用 DMA チャンネルの開始信号, 172 clock 目が通信の開始信号，818 clock 目が通信の終了信号である．848 clock 目で次の通信を開始している．

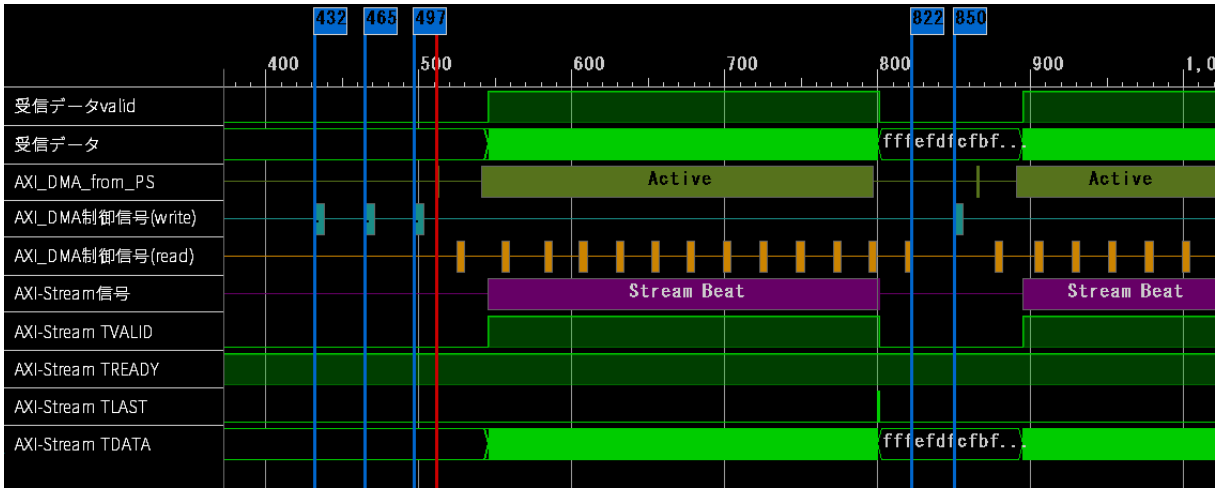

図 3.14　PL Logic への PS DDR からの読み出しをロジックアナライザで確認した様子．432 clock 目が AXI DMA IP のリセット信号，465 clock 目が受信用 DMA チャンネルの開始，497 clock 目が通信の開始信号，822 clock 目が通信の終了信号である．850 clock 目で次の通信を開始している．

認した様子を図 3.13 と図 3.14 に示す．期待通りに通信が行われていることが確認できる．

PL DDR を用いた PS-PL 間通信

以上の方法で PL FPGA ロジックから PS DDR 間への DMA 通信は可能になった．しかし，PL FPGA ロジックから PS DDR へデータを送る最大の目的は，PL FPGA ロジックの出力信号を PS CPU で解析することである．1.4 節で述べたように，CPU は複雑なデータ処理を行うことが出来る一方で，FPGA に比べて処理速度が劣る．そのため，PL FPGA ロジックで処理され出力される大規模なデータを，AXI DMA IP コアを用いる方法で PL FPGA ロジックから PS DDR へ転送可能な最大のデータサイズである 4096 byte ごと，つまり数 μs ごとに PS〜DDR へ送ると，CPU

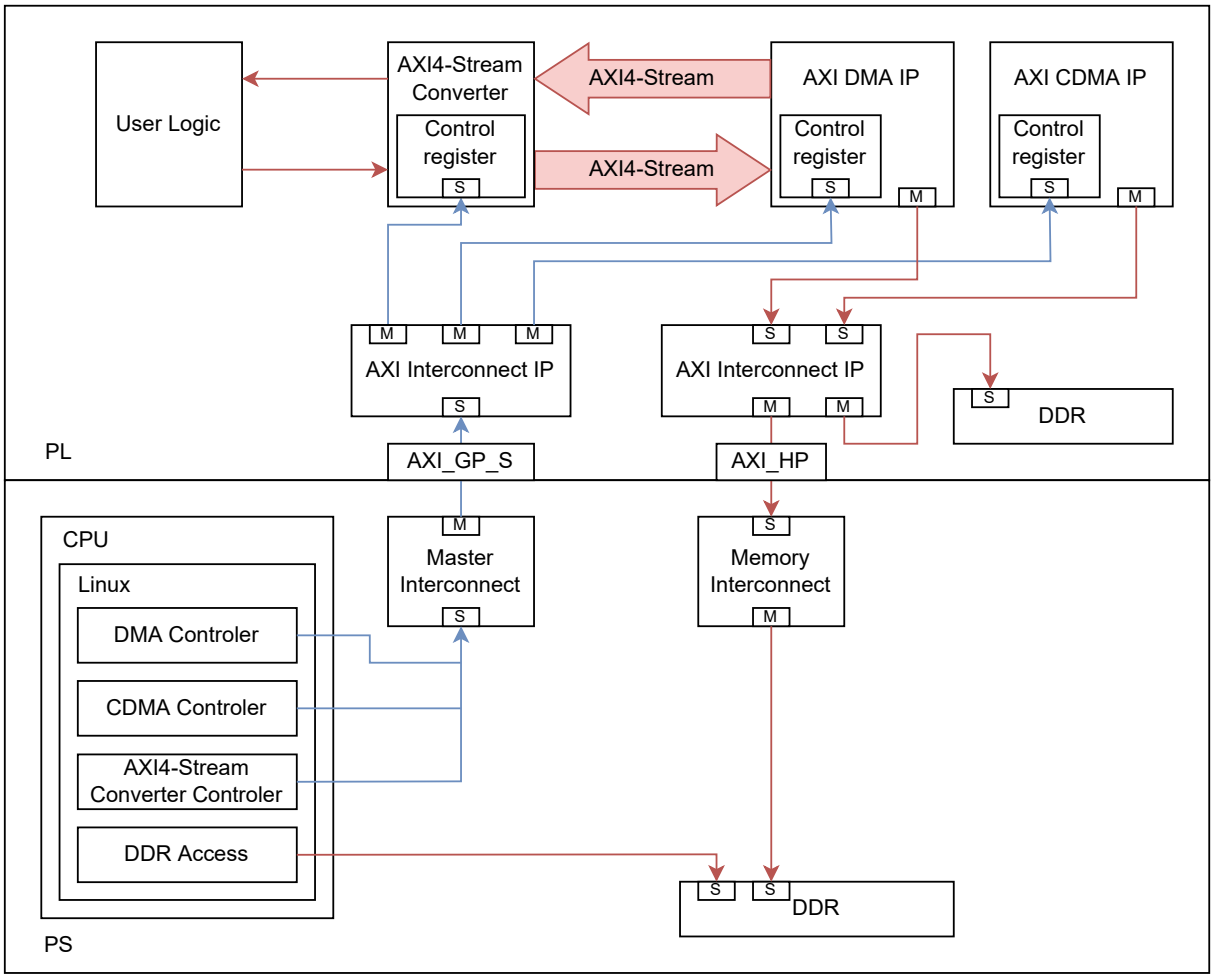

図 3.15　PL DDR を用いた PS-PL 間の通信システム．

でのデータ処理が間に合わず，PS CPU の処理能力を生かしたリアルタイムモニタリングシステムは実現できない．この問題の解決のために，PL から PS DDR への通信の間にバッファーを設ける必要がある．バッファーとして用いるメモリとして PS DDR や PL BRAM，PL DDR などが考えられる．しかし，PS CPU で行いたいデータ処理には多くの DDR メモリが必要になることも考えられるので，PS DDR リソースは使いたくない．また CPU の処理時間を稼ぐことが出来るほど PL BRAM は容量は大きくない．よって，PL DDR をバッファーにすることを考える．PL DDR をバッファとして使用するする通信システムの概要を図 3.15 に示す．

ユーザーロジックが出力したデータは，AXI DMA IP コアによる DMA 通信で PL DDR へ書き込まれ，PL DDR 内で一定のデータ量が貯まると，AXI CDMA IP コアにより PS DDR へ転送される．AXI CDMA IP コアは，Xilinx 社製 IP コアで，AXI4 プロトコルを使用した，メモリとメモリの間の DMA 通信を可能にする．AXI Interconnect IP コアを介して，AXI DMA IP コアおよび AXI CDMA IP コアは PL DDR および PS DDR へアクセスすることが出来る．PS 部の実装としては，Linux アプリケーションとして，AXI CDMA IP コントローラを追加で作成した．AXI CDMA IP の制御シークエンスを以下に示す．

初期化シークエンス

1. **AXI CDMA IP コアのリセット．**
   制御レジスタのリセットビットをアサートする．すべてのレジスタとビットが初期化される．

データ送信シークエンス

1. **ソースアドレスと書き込みアドレスの送信．**
   ソースアドレスレジスタに，転送元のメモリの転送したいデータが入っている領域の先頭アドレスを書き込み，送信アドレスレジスタに，転送を受けるメモリのデータを書き込みたい領域の先頭アドレスを書き込む．
2. **データ長の送信．**
   転送データ長レジスタに，転送するデータのサイズを書き込む．この書き込みと同時に，転送が開始される．
3. **Busy 待機．**
   AXI CDMA IP コアにる転送が完了した際，ステータスレジスタの，DMA チャンネルの IDLE 状態を示す IDLE ビットがアサートされる．このビットの立ち上がりの確認後通信を終了する．

PL FPGA ロジックから PL DDR への書き込みは，AXI DMA IP コアを使用して，3.3.2 項で示した方法と同様の方法で行うことが出来る．よって，PL DDR をバッファーとした，PL FPGA ロジックから PS DDR へのデータ転送システムが開発できた．

3.3.3 PS-PL 間通信システムの通信性能評価

開発したシステムを用いて PS-PL 間通信を行い，通信性能評価を行った．その手法，および結果をこの項で述べる．

時間測定手法

PL FPGA 内部の時間を測定するために，AXI_GP_S インターフェイスで PS CPU からアクセス可能なタイマーを PL FPGA ロジックとして開発した．また，そのコントロールアプリケーションの開発も行った．ブロック図を図 3.16 に示す．タイマーは Linux アプリケーションから，スタート，ストップ，値の読み出しが可能である．信号のタイミング図を図 3.17 に示す．PS CPU から，AXI_GP_S インターフェイスを用いて AXI4-Lite プロトコルで PL FPGA ロジックの制御レジスタやステータスレジスタにアクセスする際，その通信が完了するまで，次のアクセスは行えない．よって図のように，タイマーをスタートしてから，システムをスタートするまでの間と，システムの終了ステータスを確認してからタイマーをストップするまでにオーバーヘッドが生じる．次項以降で述べる通信時間測定結果は，このオーバーヘッドを引いた結果となっている．

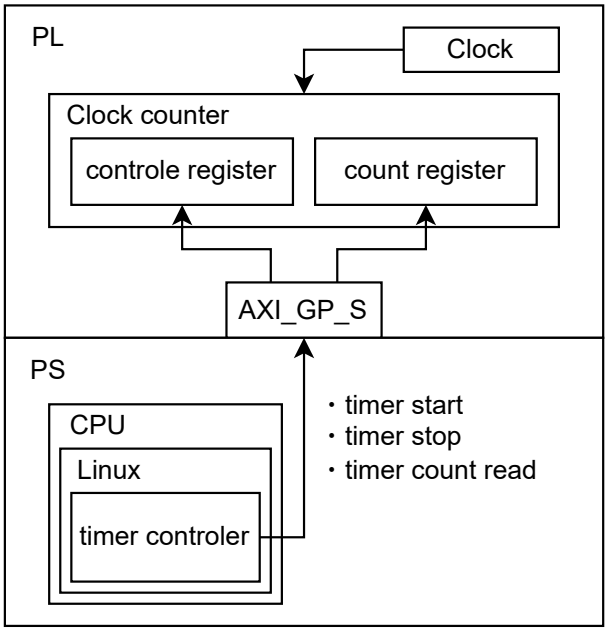

図 3.16　AXI_GP_S インターフェイスを用いたタイマーのブロック図.

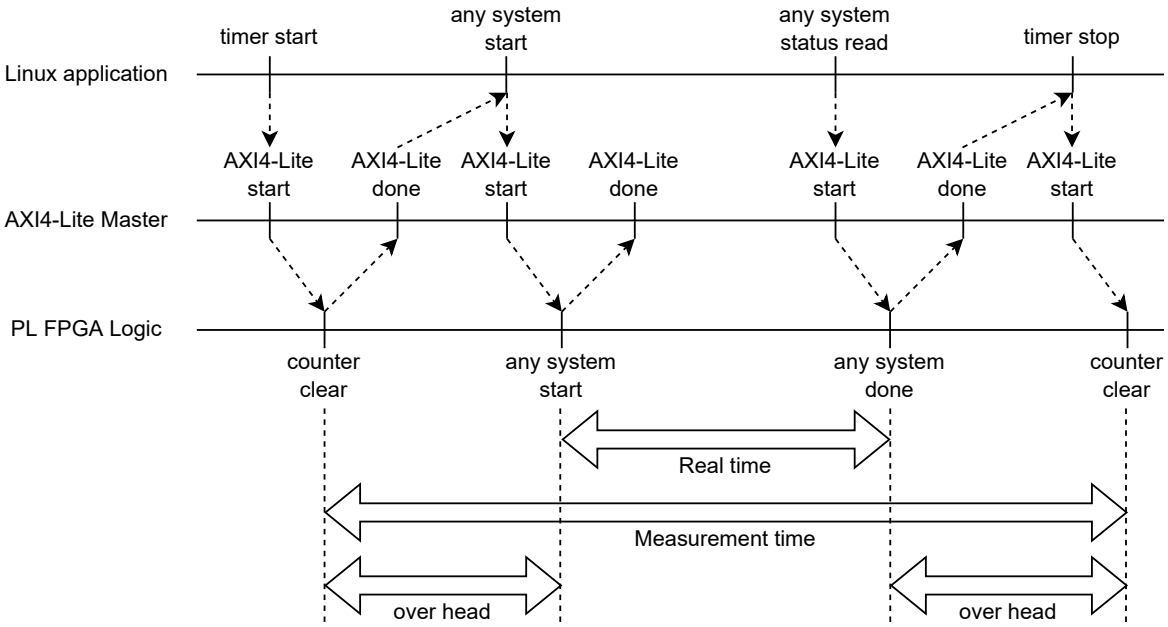

図 3.17　AXI_GP_S インターフェイスを用いたタイマーを使用する際のタイミング図.

表 3.4　PS-PL 間通信速度

送信側＼受信側	PL Logic	PL DDR	PS DDR
PL Logic		802 MB/s	633 MB/s
PL DDR	1551 MB/s		927 MB/s
PS DDR	1157 MB/s	994 MB/s	

3.3.4 通信時間測定結果

PS-PL 間通信の通信時間測定結果を述べる．PL FPGA ロジックと PL DDR 間の通信, PL FPGA ロジックと PS DDR 間の通信，PL DDR と PS DDR 間の通信について計測を行った．システム内のすべての PL FPGA のロジックは動作クロック 200 MHz で作動させた．また，通信に用いるデータは全て，200 MHz でインクリメントする 64 bit 幅のデータを用いた．データパケット長 4096 byte の通信を 256 回繰り返し，1 MB のデータを送受信し，その時間を測定した．結果を表 3.4 に示す．

AXI DMA IP コアによる DMA 通信 (PL FPGA ロジックと PL DDR 間の通信，PL FPGA ロジックと PS DDR 間の通信) では，FPAG ロジックからメモリへの書き込みと，メモリから FPGA ロジックへの読み出しで約 2 倍の通信性能の差が生じている．これは，AXI DMA IP コアの仕様上，ロジックからメモリへの書き込みの際は，AXI4-Stream プロトコルの信号ををすべて AXI4 プロトコルの信号に変換してから通信を開始するのに対し，メモリからロジックへの読み出しは，AXI4 プロトコル信号を逐一 AXI4-Stream 信号に変換して読み出すためである．これは図 3.13 および，図 3.14 からも確認できる．

PL FPGA ロジックから監視対象データを PL DDR へ貯め，PS DDR へ転送する通信について考察する．PL DDR のメモリ容量は 128 MB であるため，仮に監視対象データのデータレートが 300 MB/s であるとき，425 ms に 1 度 PL DDR から PS DDR へ転送を行えば，PL DDR の容量を最大限生かすことが出来る．128 MB のデータの PL DDR から PS DDR への転送は，150 ms 程度で完了するため，CPU の演算に使える時間は 275 ms 程度と分かる．これは，CPU での演算を行う上で十分な時間といえる．詳しくは，3.4.2 項で述べる

また, タイマーのオーバーヘッドを測定した結果, AXI_GP_S インターフェイスを用いた, PS CPU から PL FPGA ロジックのレジスタへのアクセスにかかる時間は，140 ns 程度であることが分かった．この時，データレートは 27 MB/s で，おおむね理論値と一致した．

3.4 監視対象ロジックの出力の可否判定アプリケーションの開発と性能評価

監視対象 FPGA ロジックの入力信号や出力信号をモニタリングし、ロジックが正常に作動しているかの判定を行うアプリケーションの開発について述べる．アプリケーションは監視対象 FPGA ロジックの異常の種類によって，様々な形態が考えられる．FPGA ロジックの異常には,

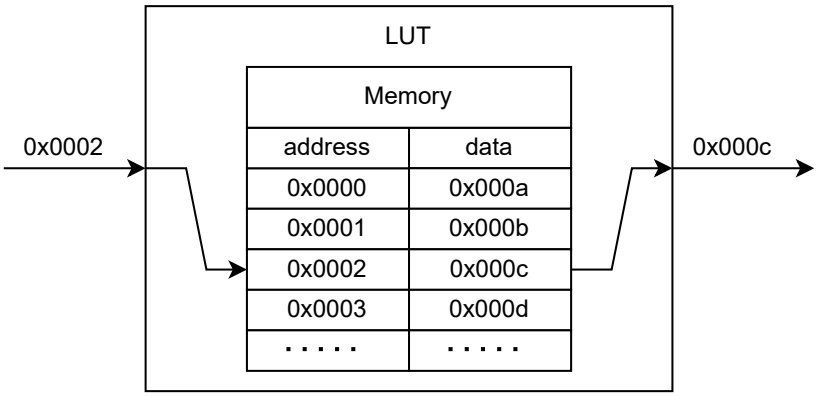

図 3.18　ルックアップテーブルの模式図．例として，0x0002 が入力された時の動作を示している．Memory の 0x0002 アドレスを参照し，格納されているデータ 0x000c を出力している．

1. ハードウェアの異常
2. 搭載アルゴリズムの異常
3. 検出器の異常

などが考えられる．1 の異常については，FPGA ロジックの出力の全てのビットを探索し，正しいデータパターンであるかの判定を行うことで検出を行うことが出来る．2,3 の異常に関しては，出力データパターンに異常が現れることは考えにくい．そのため，出力データを数万イベント貯め，統計処理を行うことで異常の検出を行うことを考える．それぞれの正誤判定アプリケーションを，テストロジックに対して開発を行った．

3.4.1 全ビット探索による正誤判定

全ビットの探索による正誤判定には，ルックアップテーブルを用いる方法が考えられる．ルックアップテーブルとは，ある値に対し，ある答えを返す表のようなもので，一般にはメモリインターフェイスに，入力をアドレス値，出力をそのアドレスに格納された値として実装される (図 3.18)．Zynq では，BRAM を用いた，PL FPGA ロジックとしての実装が考えられる．

監視対象 FPGA ロジックとしては，図 3.19 に示すテストロジックを用いる．16 bit 幅のデータを入力信号として受け取り，上位 8 bit と下位 8 bit を入れ替え，さらに bit 反転させ，Cyclic Redundancy Check (CRC) ビットを追加し出力するというものである．CRC とは，データをある値で割り算した余りをデータに付与することで，データの処理や転送の異常を検出する方法である．本ロジックでは，除算に用いる値として 0x7 を使用し，CRC ビットを 4bit 用意した．動作周波数は 160 MHz とした．入力データは，160 MHz クロックで内部 16 bit レジスタの値をインクリメントするロジックで作成したものを用いた．また，このテストロジックは論理回路にて実装し，レイテンシは hoge cycle であった．

このテストロジックに対する，ルックアップテーブルによる正誤判定を用いた監視システムの概念図を図 3.20 に示す．テストロジックの出力値としてあり得る値が入力された際に 0 を，そうでない時に 1 を出力する LUT を用いて，エラー判定を行う．出力されたエラー判定結果は，AXI_GP_S イ

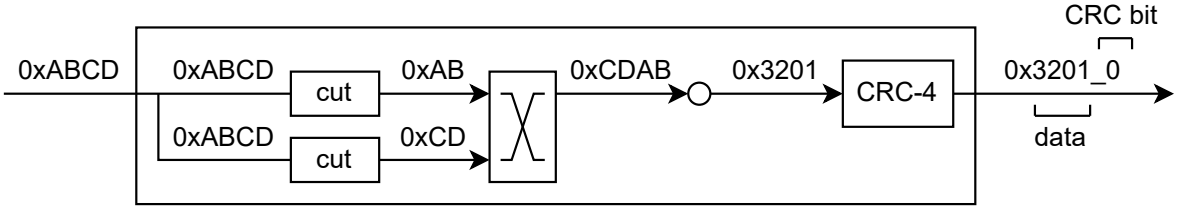

図 3.19　bit 処理を行うテストロジック．例として，入力が 0xABCD であった時の処理の様子を示している．

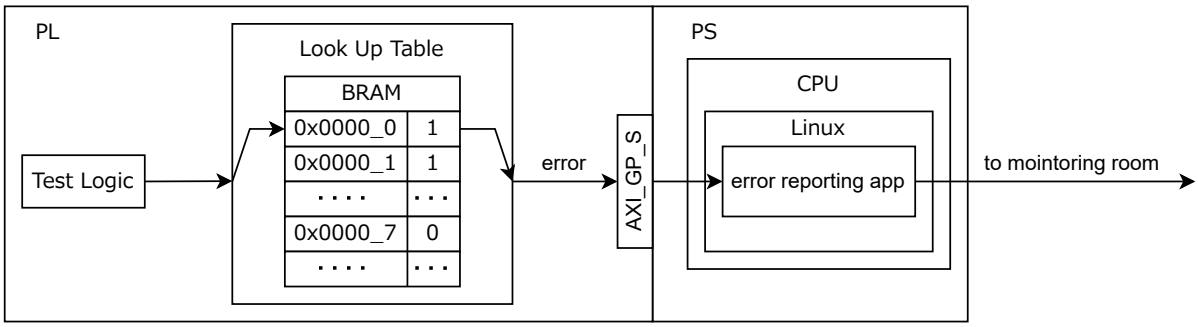

図 3.20　ルックアップテーブルを用いた正誤判定ロジックの概念図．ロジックが正常に動いている時，LUT は 0 を出力し，エラーが起こった時 1 を出力する．

表 3.5　ルックアップテーブルの使用リソース

Slice LUTs	Slice Registers	F7 Muxes	Slice	LUT	BRAM Tile
39	24	2	23	39	32

ンターフェイスを介して PS に転送され，モニタリングルームへ送られる．

テストロジックの出力信号が 20 bit であるので，実装には理論上 $2^{20} \times 1\,\mathrm{bit} = 1048576\,\mathrm{bit} = 1024\,\mathrm{Kbit}$ の BRAM を消費すると考えられる．Zynq の BRAM は 32 Kbit のタイルで分割されているので，理論上 32 個のタイルで実装できると考えられる．

この可否判定アプリケーションの実装に使用した PL FPGA リソースを表 3.5 に示す．ルックアップテーブルの実装に BRAM を 32 個のタイルを使用していることがわかり，理論値通りであった．

3.4.2 統計処理を用いた正誤判定

統計処理による正誤判定は，CPU を用いた演算により実現可能である．統計処理により判定できるエラーを再現するものとして，図 3.21 に示す，データ読み出しロジックを模したテストロジックを用意した．横方向 16 チャンネル，縦方向 16 チャンネルの読み出しチャンネルを持つ検出器を想定しており，hit があったチャンネルの bit を high にした 32 bit 信号を後段に送るデジタイザの出力を模したデータセットを用意した．データ分布は，横チャンネル，縦チャンネル，ともに $\mu = 8, \sigma = 1$ の正規分布にしたがうものとした．また，パイルアップの状況を再現するため，hit 数は $\mu = 10, \sigma = 3$ の正規分布に従うものとした．さらに，各データにデータのアイデンティファイのための 8 bit の data id 信号を付与している．テストロジックはこれらの信号を input として，hit 数が 1 の場合に

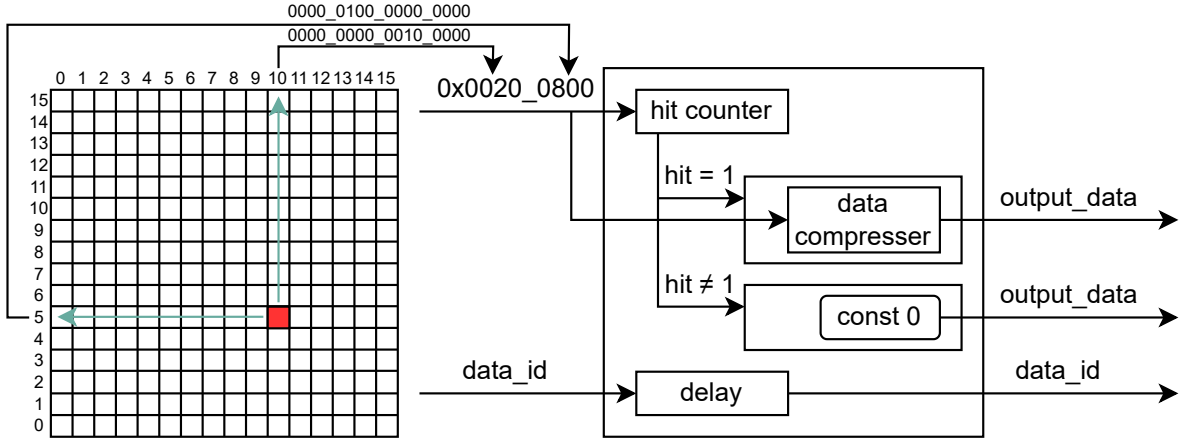

図 3.21　データ読み出しを模したテストロジック．

データを 8 bit に圧縮し，出力するものとした．8 bit のデータの内訳は，横チャンネルのヒットセルの値 (0 ∼ 15) が上位 4 bit，縦チャンネルのヒットセルの値 (0 ∼ 15) が下位 4 bit となっている．また，data id はテストロジックの処理レイテンシと同じ時間 delay をかけ，データのアイデンティティ情報が失われないようにする．テストロジックの動作周波数は 40 MHz とした．入力信号は合計 40 bit，出力信号は合計 16 bit である．これらのテストロジックに対して，正誤判定を行うアプリケーションの開発を行い，性能評価を行った．

可否判定アプリケーションおよび，監視システムの概要を以下に示す．

1. テストロジックで処理され，後段に送られたデータ (hit1 data) を DMA で PL DDR に保存する．
2. 1 秒間隔で，PL DDR から PS DDR へデータを送信する．
3. 256 MB 分の hit1 data を，シミュレーションで作成した予想統計分布と比較する．統計検定には K-S 検定を用い，得られた p 値を出力する．K-S 検定の p 値が 0.05 以下の時，TCP/IP 通信により，モニタリングルームへアラートを出力する．
4. K-S 検定が終わった hit1 data を SD カードに保存し．DDR 上のデータは削除し次のデータの受け入れを可能にする．
5. 1 分おきに，SD カードに保存されたデータセットを用いて hit1 data のヒストグラムを出力し，モニタリングルームへ送信する．
6. 送信後，データを SD カードから削除する．

hit1 data の予想されるイベントレートは，およそ 100,000 イベント/sec であり，データレートは，100 KB/sec 程度となる．PL DDR の容量は 256 MB であるため，1 秒間隔で PL DDR から PS DDR へデータ送信を行うことは十分可能である．PL DDR から PS DDR への DMA 転送は，927 MB/s であるため，100 KB のデータ転送にかかる時間は，0.1 ms 程度である．実際に 100,000 個のデータに対し，Zynq Linux アプリケーションで統計検定を実行した結果，実行時間は 200 ms 程度であった．よって，これはリアルタイムで十分実行可能なアプリケーションといえる．さらに，1 分間の出力データ，つまり 6,000,000 イベントのデータについてヒストグラミングを行っ

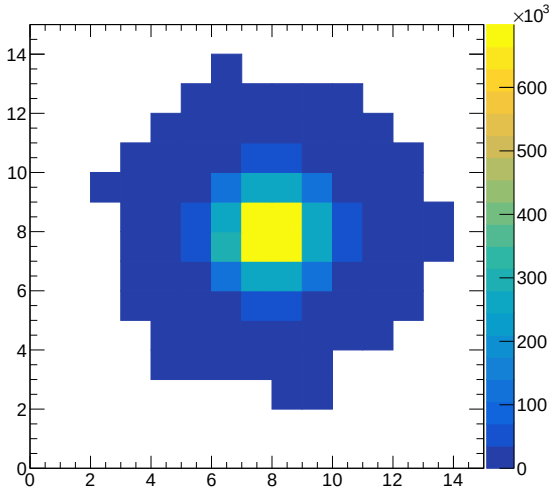

図 3.22　hit1 data をヒストグラミングした図.

た結果，実行時間 7.1 秒程度であった．こちらも，十分実行可能であるといえる．実際に PTZ 上で走る Linux アプリケーションで出力したヒストグラムを図 3.22 に示す．なお，統計処理，及びヒストグラミングにはデータ解析ライブラリ ROOT を用いた．

また，上記の可否判定アプリケーションを用いた監視システムおよび，これまでの節で述べた通信システムなどを組み合わせ，監視システムを構築した．全体図を図 3.23 に示す．PS CPU 上の Linux アプリケーションで，システム全体の制御，およびデータ処理を行う．テストロジックから出力された hit1 data は，3.3.2 項で述べた AXI4-Stream コンバータで AXI4-Stream 形式に変換され，AXI DMA IP コアにより，PL DDR へ書き込まれる．PL DDR のデータは 1 秒間隔で，AXI CDMA IP コアにより，AXI_HP インターフェイスを用いて PS DDR に DMA 転送される．AXI DMA IP は，3.3.2 項で述べた AXI DMA IP 制御アプリケーション (図中の AXI DMA controller) により制御される．また，AXI CDMA IP は，3.3.2 項で述べた AXI CDMA IP 制御アプリケーション (図中の AXI CDMA controller) により制御される．統計検定による正誤判定アプリケーション (図中の Judgement app) は，DDR の hit1 data と Simulation data を用いて K-S 検定を行い，結果の p 値をモニタリングルームの PC に送信する．また，hit1 data は SD カードコントローラにより，SD カードに保存される．SD カードの hit1 data を用いてヒストグラミングアプリケーション (図中の Histograming app) がヒストグラムを作成し，モニタリングルームの PC に送信する．Ethernet コントローラがモニタリングルームの PC とのネットワークへ接続を可能にする．実際にこのシステムを PTZ 上に構築し，実行した．意図的にエラーを起こしたデータセットを用いてシステムを検証したところ，正しくエラー判定を行うことが出来た．エラーを検知した際の，ホスト PC 上の PTZ に接続されたターミナルの様子を図 3.24 に示す．

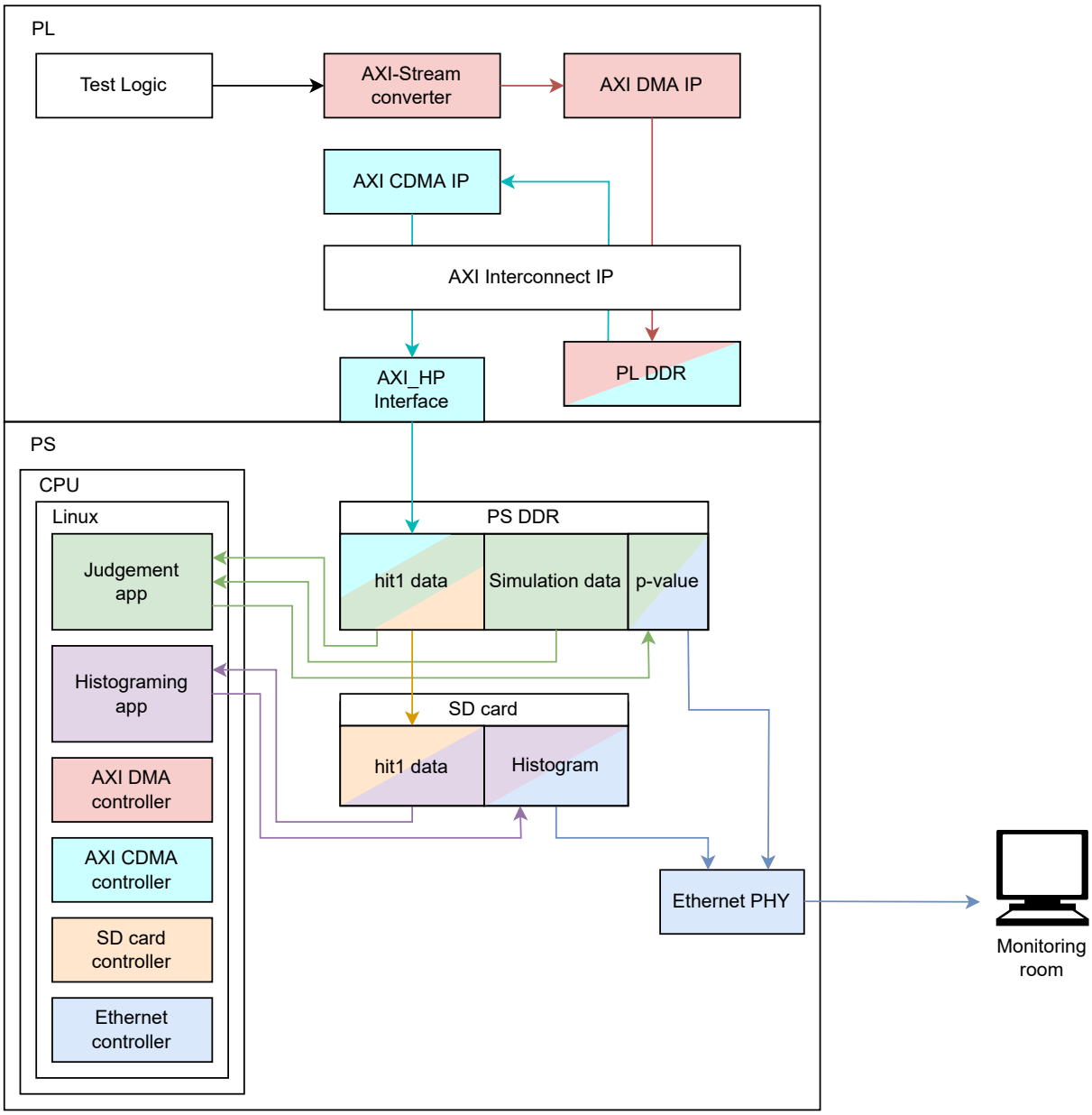

図 3.23　Linux アプリケーションを用いた FPGA 監視システム.

```
sueda@PTZ-Linux:~/work/root$ ./judgement
pvalue: 0.263381
pvalue: 0.950657
pvalue: 0.655101
pvalue: 0.066043
pvalue: 0.406748
pvalue: 0.952376
pvalue: 0.797837
pvalue: 0.0608289
pvalue: 0.536054
pvalue: 0.364054
pvalue: 6.1266e-28 error!
```

図 3.24　エラーを検知した時のホスト PC のターミナルの様子.

3.5 ロジック回復システムの開発

正誤判定アプリケーションがエラーを検知し，それがモニタリングルームに伝えられた後，モニタリングルームからロジックを回復するシステムが必要である．PL FPGA のコンフィギュレーションには複数のモードがあり，モードによってはボードの再起動をすることなく再コンフィギュレーションすることが可能である．これにより，モニタリングシステムはエラーが起こった際も稼働し続け，リアルタイムモニタリングを続けることが出来る．

3.5.1 PS から PL のコンフィギュレーション

PL コンフィギュレーションには主に使用される 2 つのパスがあり，選択可能である．特徴と共に，以下に示す．

- JATG デバッグ：デバッグで一般的に用いる．TAP コントローラを介して PL コンフィギュレーションを行う．
- PCAP パス：PS がマスターである．デバイスコンフィギュレーションインターフェイス (DevC) を介して PL コンフィギュレーションを行う．

また，PL の再コンフィギュレーションは一部の書き換えとすべての書き換えのモードが存在する．本システムでは，PCAP パスを用いた PS から PL の全コンフィギュレーションを行う．このパスを用いた PL コンフィギュレーションは，Linux kernel の device driver を用いて容易に実現が可能である．具体的に使用する device driver は，FPGA manager，FPGA bridge，FPGA region，Device-tree overlay である．Dvice-tree に，再コンフィギュレーションに使用するビットストリームファイルと，Zynq の PL コンフィギュレーションに関する制御レジスタを割り当てた FPGA manager モジュールを記載し，Device-tree overlay により Device-tree を書き換えることで，PS は PL FPGA のコンフィギュレーションを行う [16]．具体的な手順を以下に示す．

1. PL の全コンフィギュレーションフラグを立てる
   - echo 0 >/sys/class/fpga manager/fpga0/flags
2. コンフィギュレーション用のビットストリームファイルと，Device-tree ファイルを firmware フォルダにコピーする．
   - cp /media/new-bitstream.bin /lib/firmware/new-bitstream.bin
   - cp /media/new-devicetree.dtbo /lib/firmware/
3. Device-tree ファイルを適応する
   - mkdir full
   - echo -n ”new-devicetree.dtbo” >full/path

以上の手順により，Device-tree が書き換えられ，それに伴い，PL は再コンフィギュレーションされる．なお，再コンフィギュレーションによりデバイスや AXI インターフェイスに変更が生じる際は，変更にともなった Device-tree の記載が必要となる．これらの手順は，モニタリングルームから監視

システムの Zynq PS Linux OS に ssh 接続することで実現可能である．

実際に PTZ とホスト PC を同一ネットワーク上に接続し，PTZ 上で走る Linux OS にホスト PC から ssh 接続し，上記の手順で PL を再コンフィギュレーションできるか確認を行った．PL のビットストリームファイルが書き換えられたことがわかりやすいように，書き換えを行うビットストリームファイルでは，テスト用 LED が点灯するものを用意した．上記の手順で Device-tree overlay を実行した結果，LED の点灯を確認，PL が正しくコンフィギュレーションできたことを確認した．

3.5.2 外部ボードのコンフィギュレーション

Zynq PL には豊富な外部 I/O ポートが用意されているため，これまでに述べたシステムで，同チップ内の PL FPGA ロジックの監視だけでなく，外部 FPGA ボードの監視も可能であると言える．ここでは，外部 FPGA ボードのコンフィギュレーションの可能性について述べる．Xilinx 社製 FPGA はスレーブモードを選択することで，外部プロセッサや，マイコンからシリアル通信を介してコンフィギュレーションをすることが出来る [17]．Zynq PS CPU ももちろん，このコンフィギュレーションのマスターになることが出来るため，Zynq から外部 FPGA のコンフィギュレーションをすることは可能といえる．よって本研究で開発した FPGA 監視システムは外部 FPGA ボードに対しても適応可能である．

3.6 システムの応用例

以上に，Zynq を用いた FPGA ロジック監視システムに必要な要素を示した．これらを組み合わせることで，FPGA 監視システムは多様な FPGA ロジックに対し適応できる．この節では，実際の高エネルギー実験で用いられる FPGA ロジックに対する監視システムの可能性の提案を行う．

例として採用するのは，ATLAS 実験の New Sector Logic(NSL) のトリガーロジックおよび，リードアウトロジックである．NSL は，ATLAS 実験のエンドキャップミューオン検出器からのデータの読み出し，およびそのデータを用いたトリガー発行に用いられる FPGA ボードである [12]．NSL に実装される FPGA ロジックのブロック図を図 3.25 に示す．この FPGA ロジックを本研究で開発したシステムで外部から監視することを想定する．トリガーロジックの入力データレートは 7.37 GB/s，出力データレートは 763 MB/s である．また，読み出しロジックの入力データレートは 8.34 GB/s，出力データレートは 305MB/s である．これらすべてのデータを PL から PS へ転送することは不可能なので，ロジックの処理内容から，モニタリングするデータを決定する．このロジックは，高いレートのデータを処理し，データレートを落として後段に送る，という点で，3.4.2 項で提示したテストロジックと似た構造である．よって，同様の方法を用いた監視システムを構築することを考える．監視対象データは，読み出しロジックの出力信号である．この信号を PL から PS へ送り，統計検定およびヒストグラミングしてエラー判定を行う．監視対象データのレートは 305 MB/s であることから，3.3.4 項の議論により，CPU での演算に使える時間は 275 ms 程度である．**??** 項の正誤判定アプリケーションの処理時間から予想すると，十分統計処理が可能と考えられる．よっ

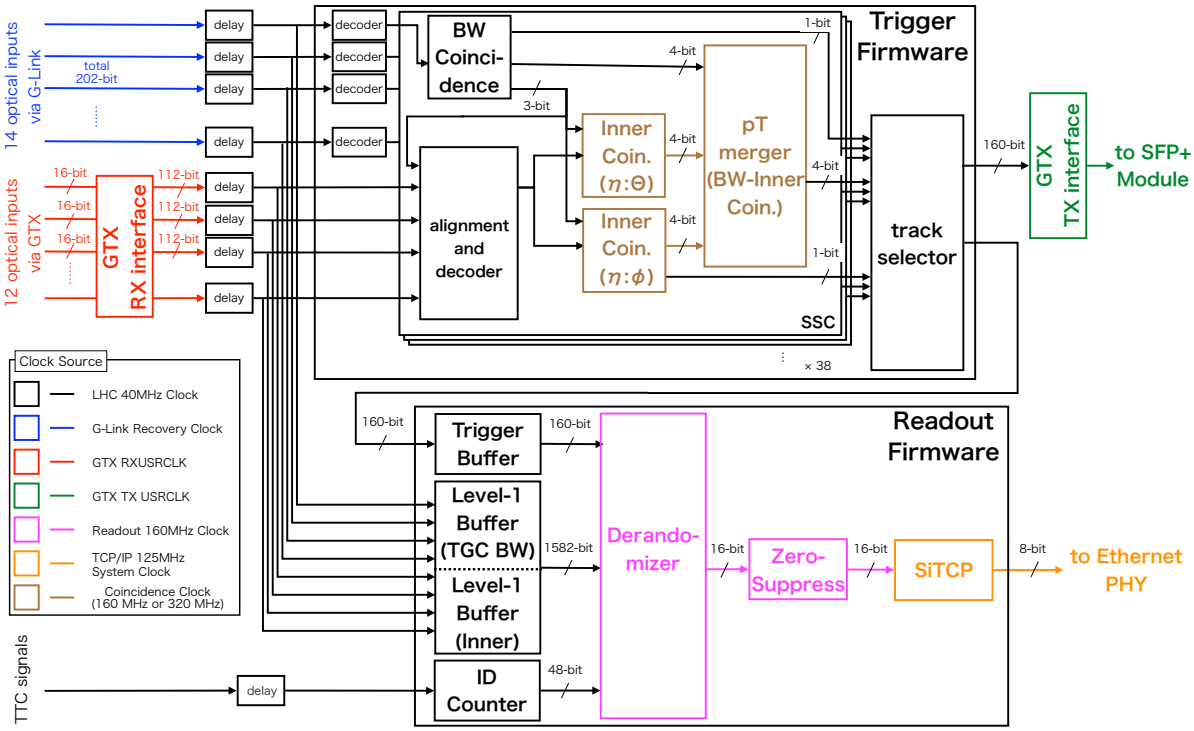

図 3.25　NSL に実装されるトリガーロジックおよび読み出しロジックのブロック図 [12].

て，NSL の読み出しロジックのリアルタイムモニタリングは，本研究で作成した監視システムによって実現可能であると言える．

第 4 章
結論

標準模型を超える物理の探索のために，現在，世界中で物理実験が行われている．新物理探索の実験手法の一つである高エネルギー物理実験では，加速した粒子を高い重心系エネルギーで衝突させて非常に多くの物理事象を起こし，生じた二次粒子を検出器で捉えることでそれを観測する．検出器からのデータの読み出しには，非常に多くの FPGA デバイスが使用されており，そのオンラインモニタリングは早急の課題である．本研究では，Xilinx 社製 SoC デバイス Zynq を用いて，リアルタイム FPGA ロジック監視システムの開発を行った．

最初に，リアルタイム FPGA ロジック監視システムの要件を以下のように定義した．

- 監視対象 FPGA ロジックの信号の捕捉が出来ること．
- 捕捉した信号をもとに，監視対象 FPGA ロジックに対する可否判定を行えること．
- 観測室へのエラー報告および観測室からの FPGA ロジックの回復の実行が可能なこと．
- 以上 3 つの要素を持ったシステムが測定室内で独立稼働すること．

また，それぞれの要件に対し，実装レベルでのシステムの構成要素の決定を以下のように行った．

1. システムの全体制御のための Linux OS.
2. PS-PL 間通信システム．
3. 監視対象ロジックの可否判定システム．
4. FPGA ロジック回復システム．

それぞれの要素について開発および性能評価を行った．

まず，1 のシステムの全体制御のための Linux OS の開発を行った．Zynq デバイスで動作する Linux OS に必要なコンポーネントについてそれぞれ開発を行い，作成した Linux OS の起動確認および，外部ネットワークとの接続確認を行った．

次に，2 の PS-PL 間通信システムの開発を行った．

- PS から PL FPGA の制御レジスタにアクセスするための，AXI_GP_S インターフェイスを用いた通信システム．
- PL FPGA ロジックと PS DDR 間および，PL DDR と PS DDR 間大容量データ通信のための，AXI_HP インターフェイスを用いた通信システム．
- PL FPGA ロジックと PL DDR 間の通信システム．

を開発した．また，それぞれの通信システムの制御用 Linux アプリケーションの開発も行った．それぞれの通信システムについて通信速度測定を行い，そこから予想される，PS CPU でのデータ処

理の可能性を示した.

さらに，3 の監視対象ロジックの可否判定システムの開発を，典型的な FPGA の異常のパターンに対して行った.

- ハードウェアの異常の検出のための，PL での LUT を用いた，全ビット探索による正誤判定アプリケーション.
- アルゴリズムの異常や検出器の異常の検出のための，PS での統計処理およびヒストグラミングを用いた正誤判定アプリケーション.

を開発した.

そして，4 の FPGA ロジック回復システムの開発を行った.

- Zynq PL FPGA の Zynq PS からのコンフィギュレーション手法の確認.
- 外部ボード FPGA の Zynq PS からのコンフィギュレーションの可能性の提示.

を行った.

最後に，本研究で開発したシステムの，実際の ATLAS 実験に用いられる読み出しロジックに対する適用可能性を示した.

今後, 実際の DAQ システムの FPGA ロジックを監視するために, 起こり得るエラーやその FPGA ロジックの形態に対し最適な可否判定アプリケーションの開発を行っていく必要がある.

本研究で開発したリアルタイム FPGA ロジック監視システムは，非常に汎用性の高いものであり，今後様々な高エネルギー実験の FPGA ロジックの監視に使用することが可能である．本システムが今後の新物理探索において重要な役割を果たすことは想像に難くない.

謝辞

本修士論文の執筆にあたり，非常に多くの方の力添えをいただきました．ここに大いなる感謝を示したいと思います．

指導教員の藏重久弥先生には，3 年もの間，研究指導はもちろんのこと，生活面や体調面でもご心配をかけ，大変お世話になりました．先端融合研究環長という役職柄，大変忙しいにもかかわらず，質問のために居室を訪れるたびに親身になってご指導していただきました．感謝してもしきれません．大変ありがとうございました．

神戸 ATLAS グループの前田順平先生，山﨑祐司先生，越智敦彦先生には，毎週のミーティングで，研究に対するご意見，ご指導をいただき，そのたびに研究を前に進めることが出来ました．前田先生は，学会出張や CERN 出張などで多くご同道してくださり，非常に助けになりました．また，ハードウェアやソフトウェアの基礎的な知識を一から指導してくださり，その指導が無ければこの研究は成しえませんでした．山崎先生は，日中のコーヒータイムで，示唆に富んだ含蓄のある話をしてくださり，楽しませていただきました．越智先生は，活力を持って研究に挑む姿勢をその背中で語ってくださり，研究者としての生きざまを学ばせていただきました．皆様，大変ありがとうございました．

KEK の池野正弘先生，佐々木修先生，ICCEP の坂本宏先生，特殊電子回路株式会社の内藤竜治様には，PTZ のテストに伴い，技術的な指導をしていただきました．坂本先生には，一度 KEK で会食をさせていただいた際，「君はハードウェア顔だね」という言葉をいただいたことを覚えています．その言葉を胸に，現在までハードウェアを用いた研究を続けてくることが出来ました．ありがとうございました．

粒子物理学研究室の竹内康雄先生，身内賢太朗先生，鈴木州先生，中野佑樹先生，東野聡先生には，毎週のミーティングで，コロキウム発表に対する質問の仕方などから，良い発表とは何ぞやということを学ばせていただきました．身内先生は，会食のたびに，いい意味で上品ではない言葉を高らかに叫ばれており，元気をいただきました．中野先生には，多くの会食に連れて行っていただきました．記憶に強く刻まれた会（記憶には残っていない）も少なくありません．ありがとうございました．

粒子物理学研究室の先輩の竹田康亮さん，石浦宏尚さん，水越彗太さん，日比宏明さん，堀口楠日さん，京都大学高エネルギー物理学研究室の先輩の岡崎佑太さんには，日々の研究の相談に乗っていただきました．竹田さんは，CERN 出張の際，慣れないフランス生活のサポートをしてくださいました．また，多くの会食を開催してくださり，何度も貴重な体験をさせていただきました．会食後，朝起きた時に竹田さんの車の側面がピンクに染まっていたことは記憶に鮮明です．石浦さんは，私が B4 のころから，非常にフレンドリーに接してくださり，研究室に馴染むことを手伝ってくださいました．水越さんは，本当に何度も会食に連れて行ってくださり，そのたびにくだらない議論をして

は，楽しい時間を過ごすことが出来，日々の活力になりました．日比さんは，日々の研究室生活で大変友好的に接してくださいました．また，血中アルコール濃度が高まった日比さんには，非常にお世話になりました．現在流行している感染症が収まり，羽を伸ばした日比さんと会食をする日を待ち望んでいます．ありがとうございました．

粒子物理学研究室の同期の網本圭輔くん，塩沢知晃くん，説田暉くん，中村拓馬くん，山下和輝くん，網本圭輔くん，上野龍一くん，塩見崇宏くん，島田拓弥くん，角源一郎くん，谷口浩平くん，山本祥太郎くん，安部草太くん，池森隆太郎くん，尾崎博紀くん，窪田諒くん，谷口大悟くん，寺村七都くん，長崎大智くん，野口健太くん，前田剛志くん，ICEPP の同期の田中碧人くん，杉崎海斗くん，京都大学高エネルギー物理学研究室の同期の辻川吉明くん，小林蓮くんは，日々の生活で，研究について議論したり，たわいもない話をしたりと，人生を豊かにしてくれる貴重な存在でした．ありがとうございました．

最後に，8 年間の学生生活を支えてくださった母，祖父，祖母に多大なる感謝の意を表し，筆を擱きたいと思います．

参考文献

[1] ATLAS Collaboration. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Phys. Lett. B*, Vol. 716, No. 1, pp. 1–29, 2012.

[2] CMS Collaboration. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Phys. Lett. B*, Vol. 716, No. 1, pp. 30–61, 2012.

[3] ひっぐすたん. https://higgstan.com/.

[4] LHC SUSY Cross Section Working Group. Cross-section plots. https://twiki.cern.ch/twiki/bin/view/LHCPhysics/SUSYCrossSections.

[5] ATLAS Collaboration. Luminosity Determination using $Z \to \ell\ell$ events at $\sqrt{s} = 13$ TeV with the ATLAS detector. ATL-DAPR-PUB-2021-001, Feb, 2021.

[6] Trigger and Data Acquisition, CERN. https://atlas.cern/discover/detector/trigger-daq.

[7] Nicolai Schroer. Design and Evaluation of the IBL BOC for the ATLAS Experiment at CERN, Feb 2013.

[8] T.Saito. TDAQ Status, ATLAS Week(Waseda/Tokyo, Japan), June 2018.

[9] ATLAS Collaboration. Technical Design Report for the Phase-I Upgrade of the ATLAS TDAQ System. Technical report, Sep 2013. Final version presented to December 2013 LHCC.

[10] Peter Jenni, Marzio Nessi, Markus Nordberg, and Kenway Smith. *ATLAS high-level trigger, data-acquisition and controls: Technical Design Report*. Technical design report. ATLAS. CERN, Geneva, 2003.

[11] Xilinx Inc. Zynq-7000 SoC Technical Reference Manual, UG585(v1.13), April 2021.

[12] 岡﨑 佑太. LHC-ATLAS 実験 Run-3 に向けたミューオントリガーの改良とハードウェアへの実装，京都大学 修士論文, 2018. https://www-he.scphys.kyoto-u.ac.jp/theses/master/okazaki_mt.pdf.

[13] arm. AMBA AXI and ACE Protocol Specification. https://developer.arm.com/documentation/ihi0022/hc/.

[14] 堀口 楠日. Zynq 搭載汎用モジュール PT-Z の開発と LHC-ATLAS 実験への応用，神戸大学 修士論文, 2019. https://ppwww.phys.sci.kobe-u.ac.jp/seminar/pdf/Horiguchi_thesis.pdf.

[15] 池野正弘. PTZ-Zynq 搭載汎用 VME モジュール開発プロジェクト.

[16] Xilinx Wiki. Solution Zynq PL Programming With FPGA Manager. https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841645/.

[17] Xilinx Inc. 7 Series FPGAs Configuration, UG470(v1.13.1), August 2018.

付録 A PTZ 回路図 [14]

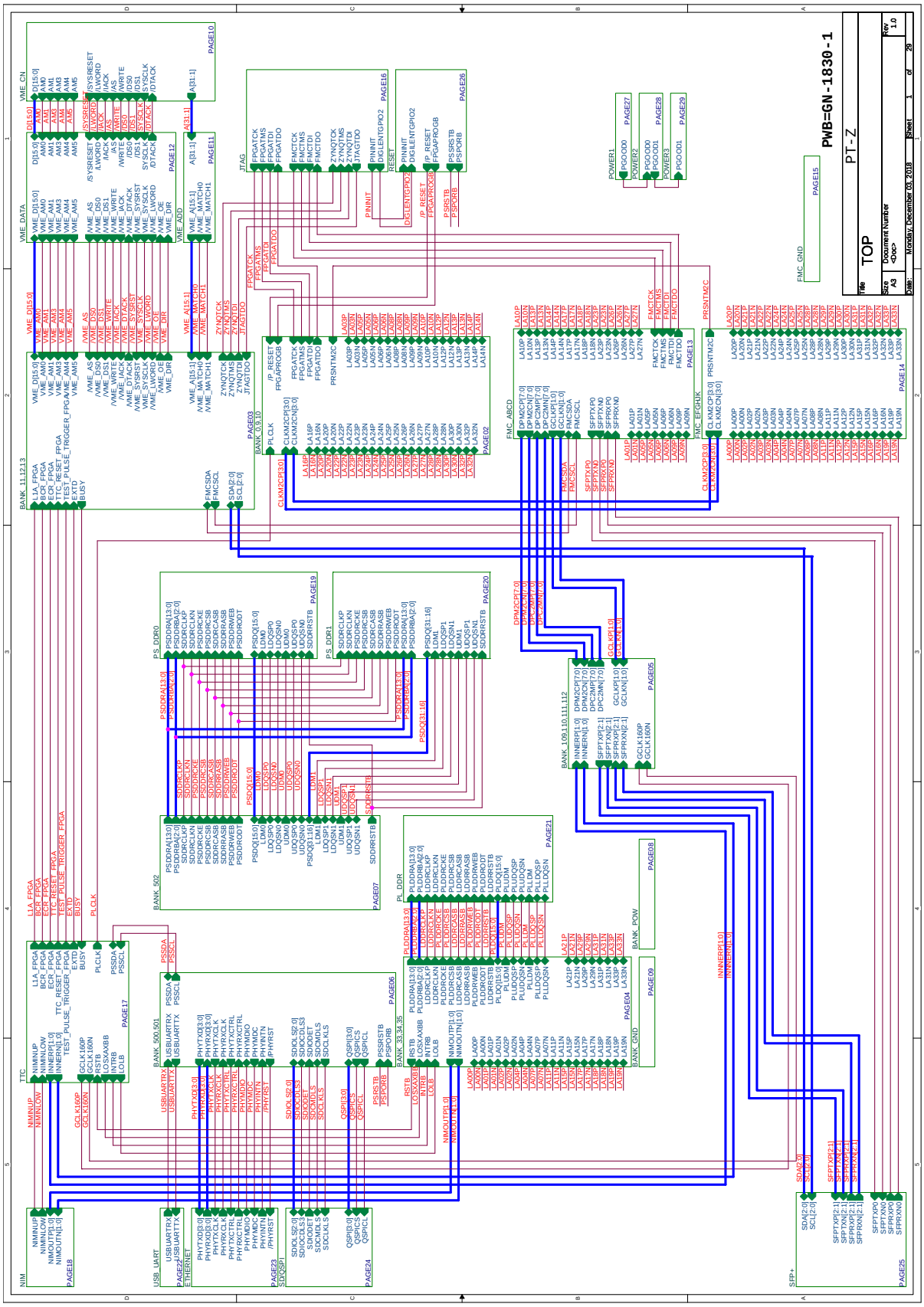

図 A.1　PTZ 回路図，全体図.

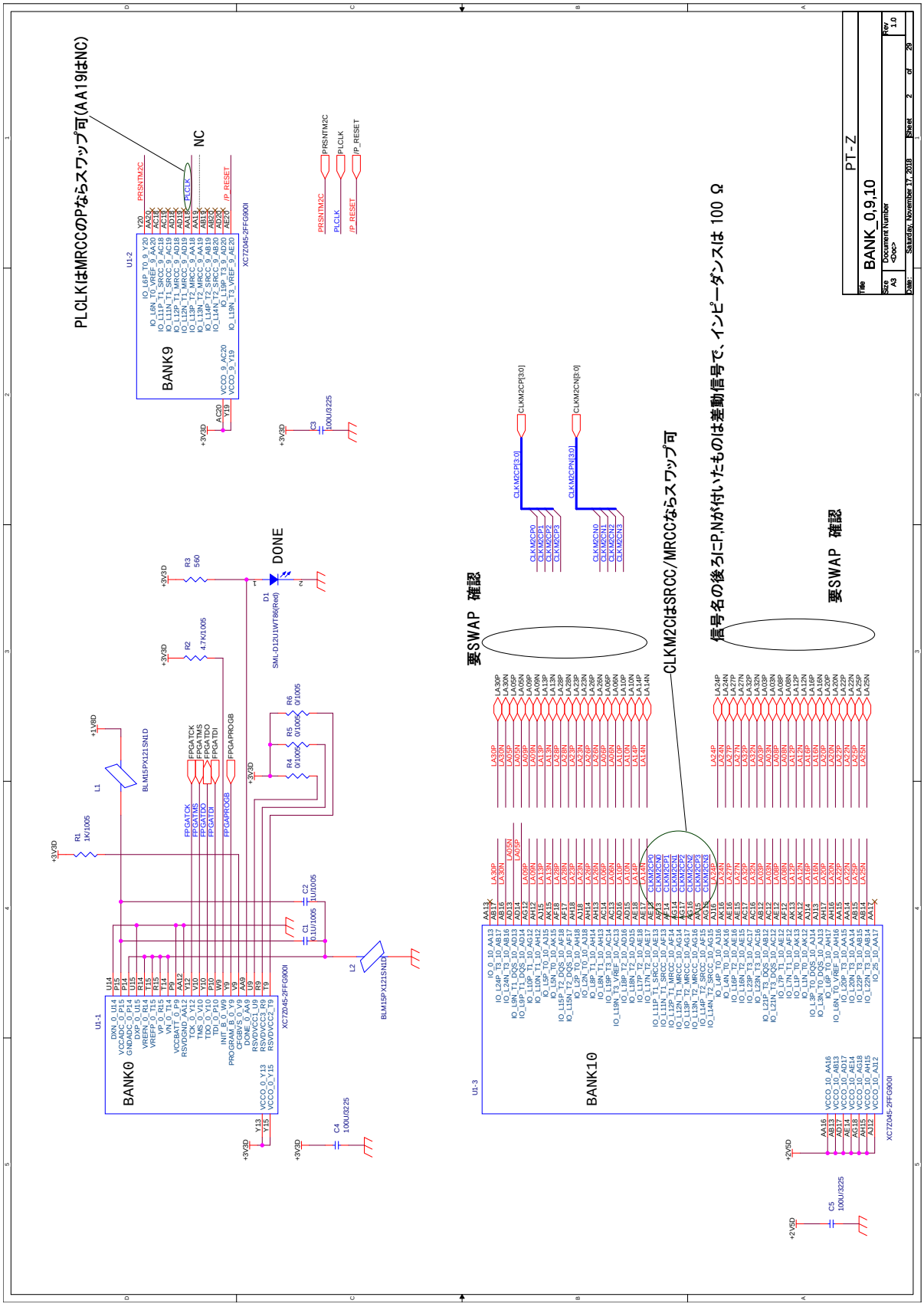

図 A.2　PTZ 回路図，Zynq のコンフィギュレーションバンク (BANK0) と PL バンク (BANK9/10)．BANK10 はメザニン接続．

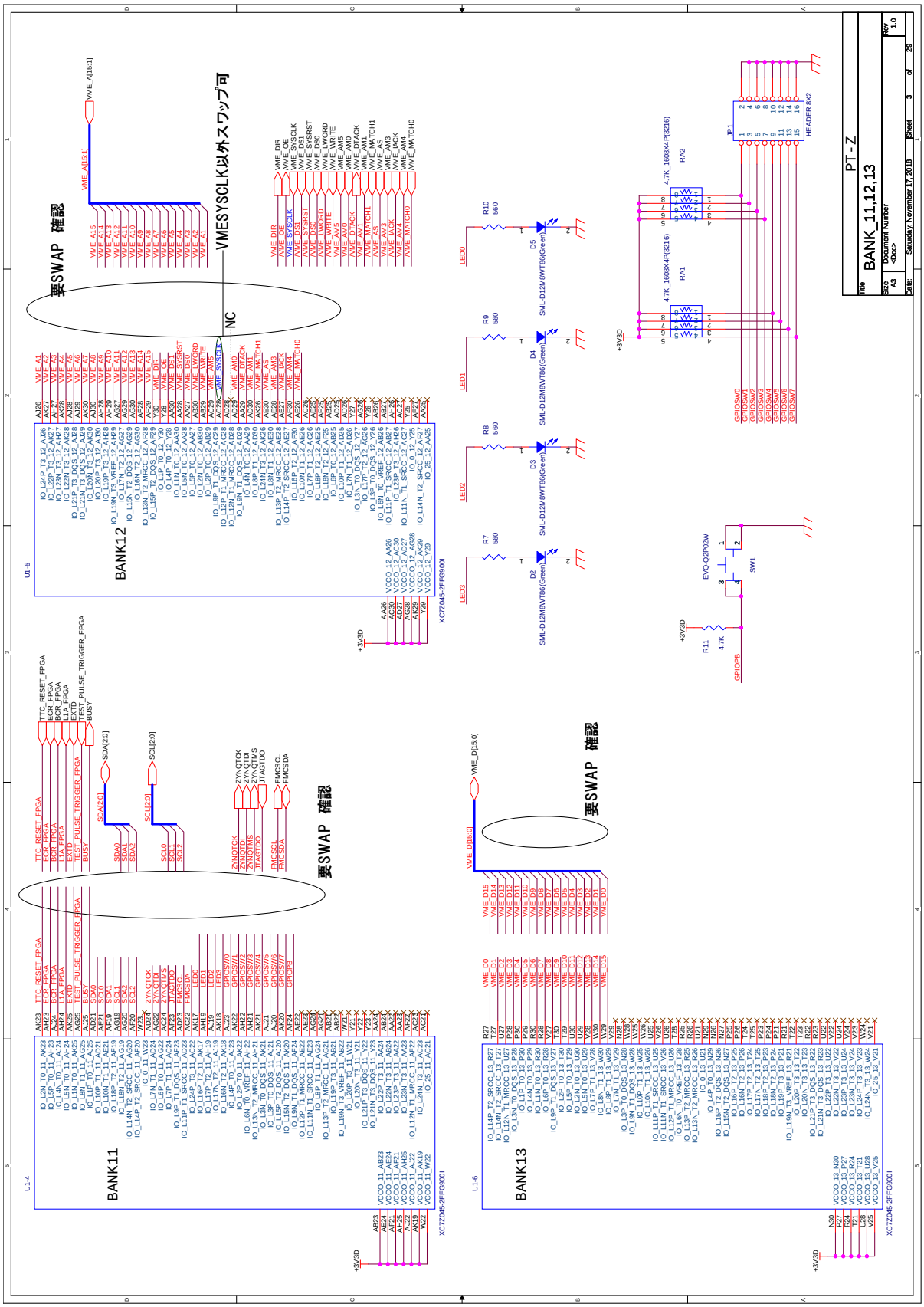

図 A.3　PTZ 回路図，Zynq の PL バンク (BANK11/12/13)．BANK12/13 は VME バス接続.

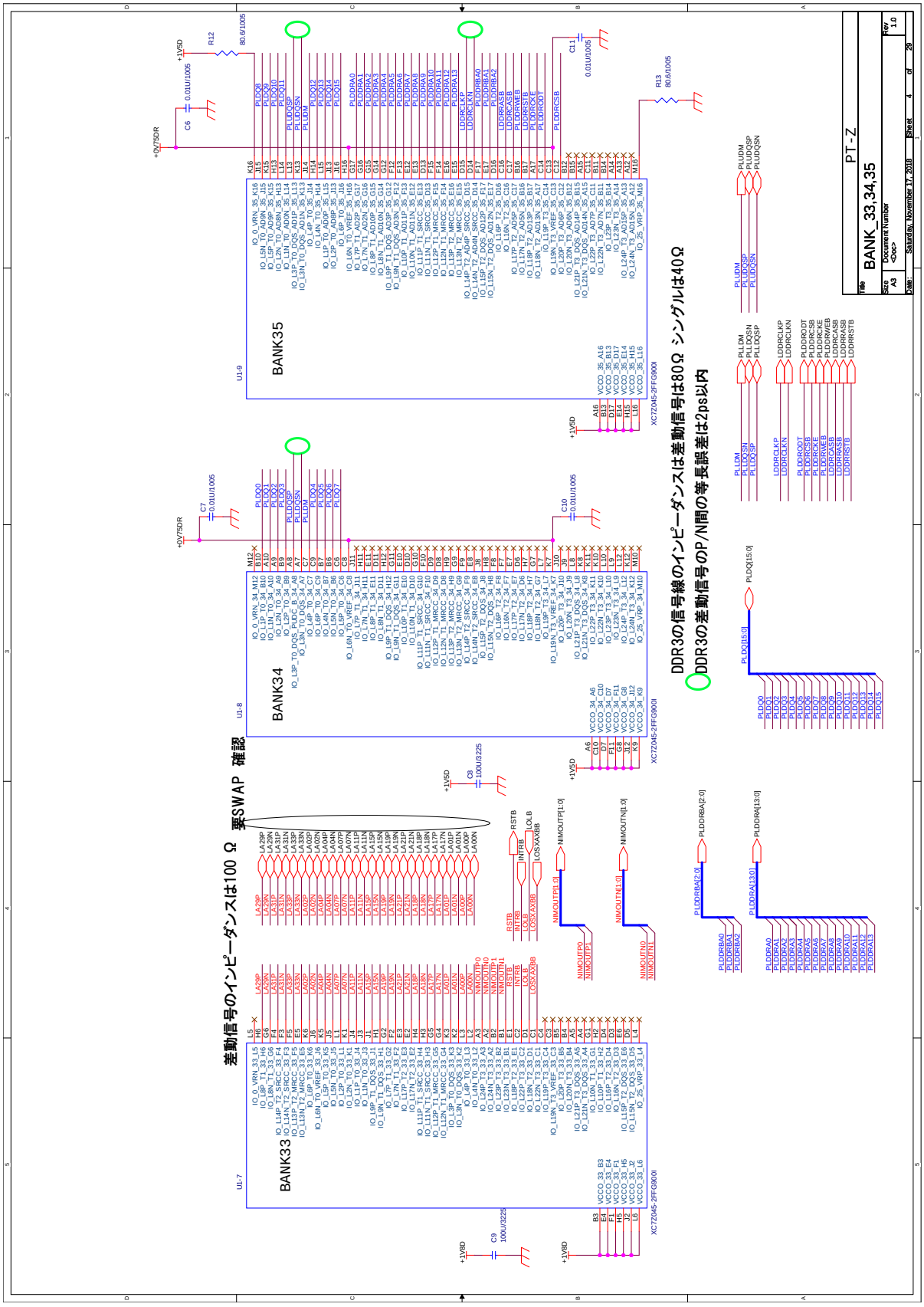

図 A.4　PTZ 回路図，Zynq の PL バンク (BANK33/34/35)．BANK33 は主にメザニン接続．BANK34/35 は PL DDR3 メモリ接続．

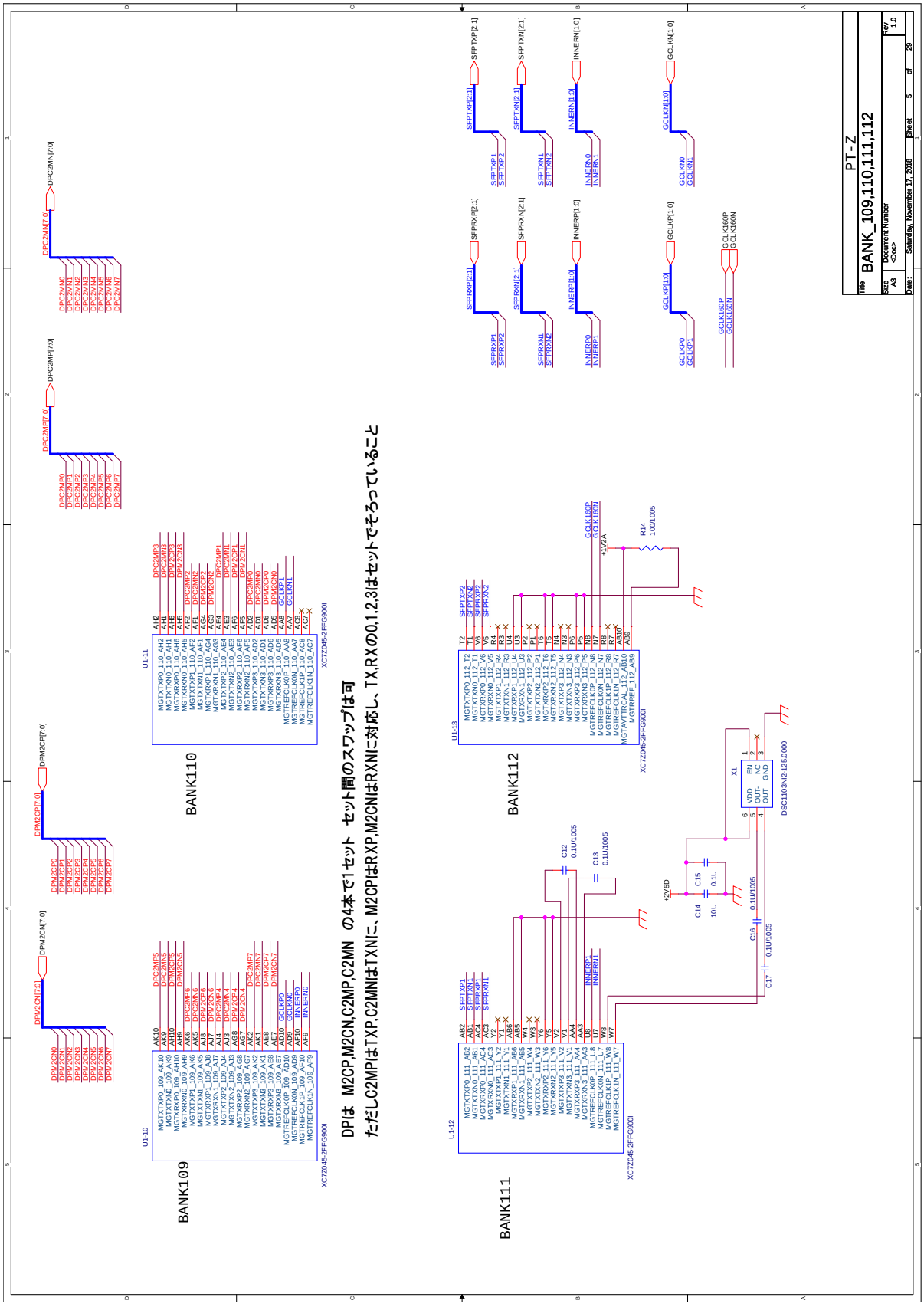

図 A.5　PTZ 回路図，Zynq の PL バンク (BANK109/110/111/112)．BANK109/110 は主に Disply Port のメザニン接続．BANK111/112 は GTX 通信用．

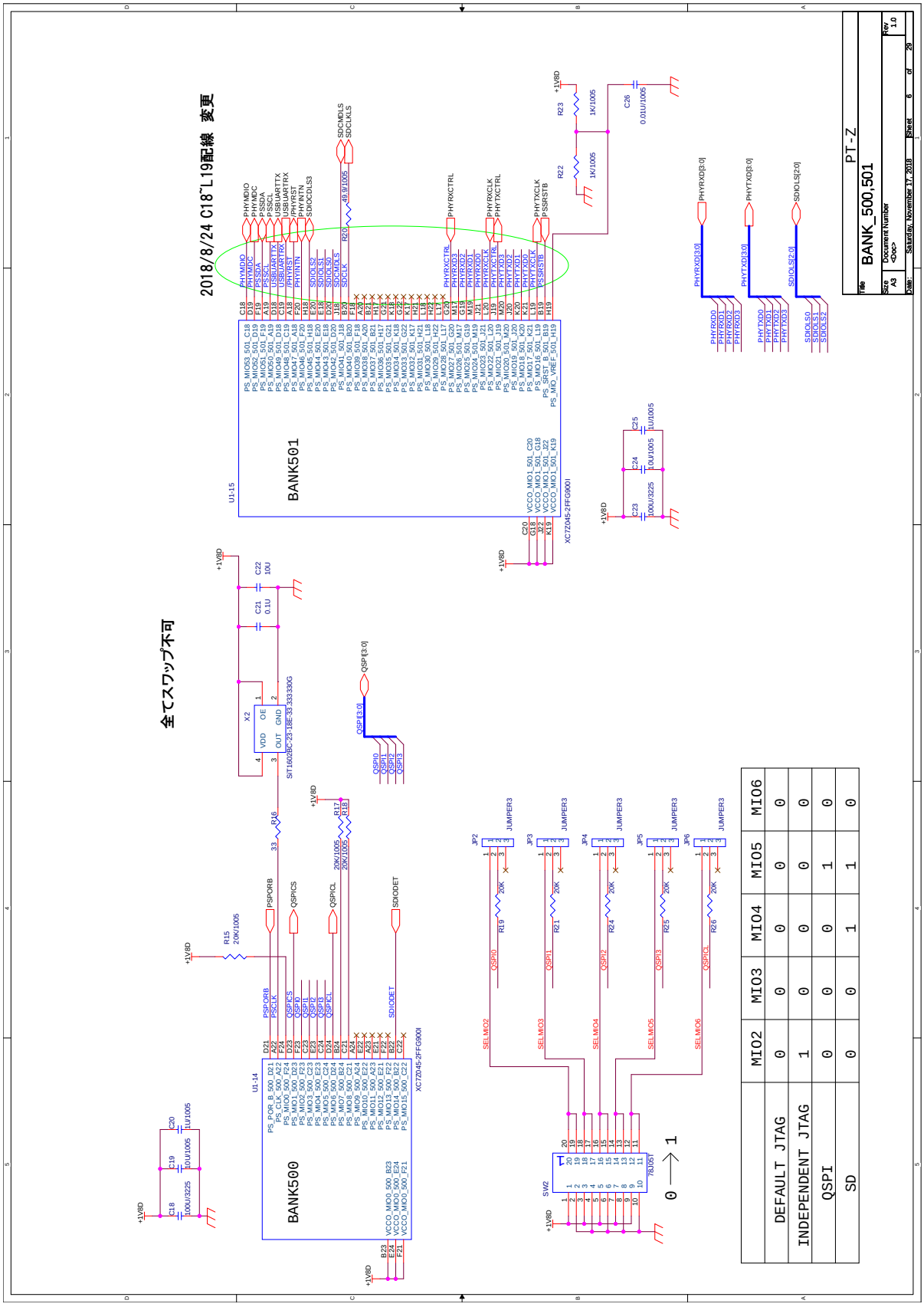

図 A.6　PTZ，Zynq の PS バンク (BANK500/501)．BANK500 は主に QSPI フラッシュメモリへの接続．BANK501 は主に SDIO，Ethernet PHY，I2C，USB-UART への接続．

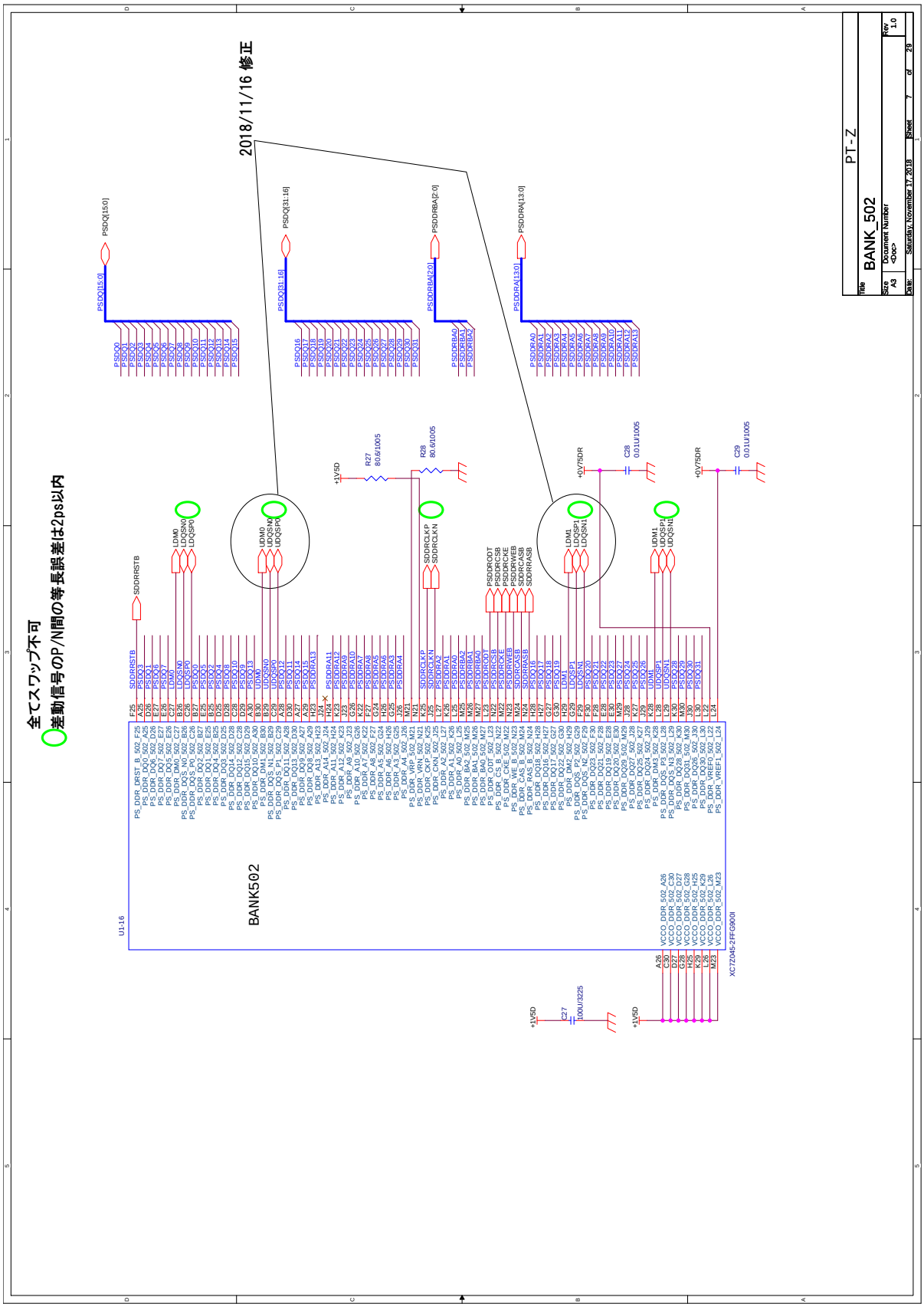

図 A.7　PTZ 回路図，Zynq の PS バンク (BANK502)．DDR3 メモリ接続．

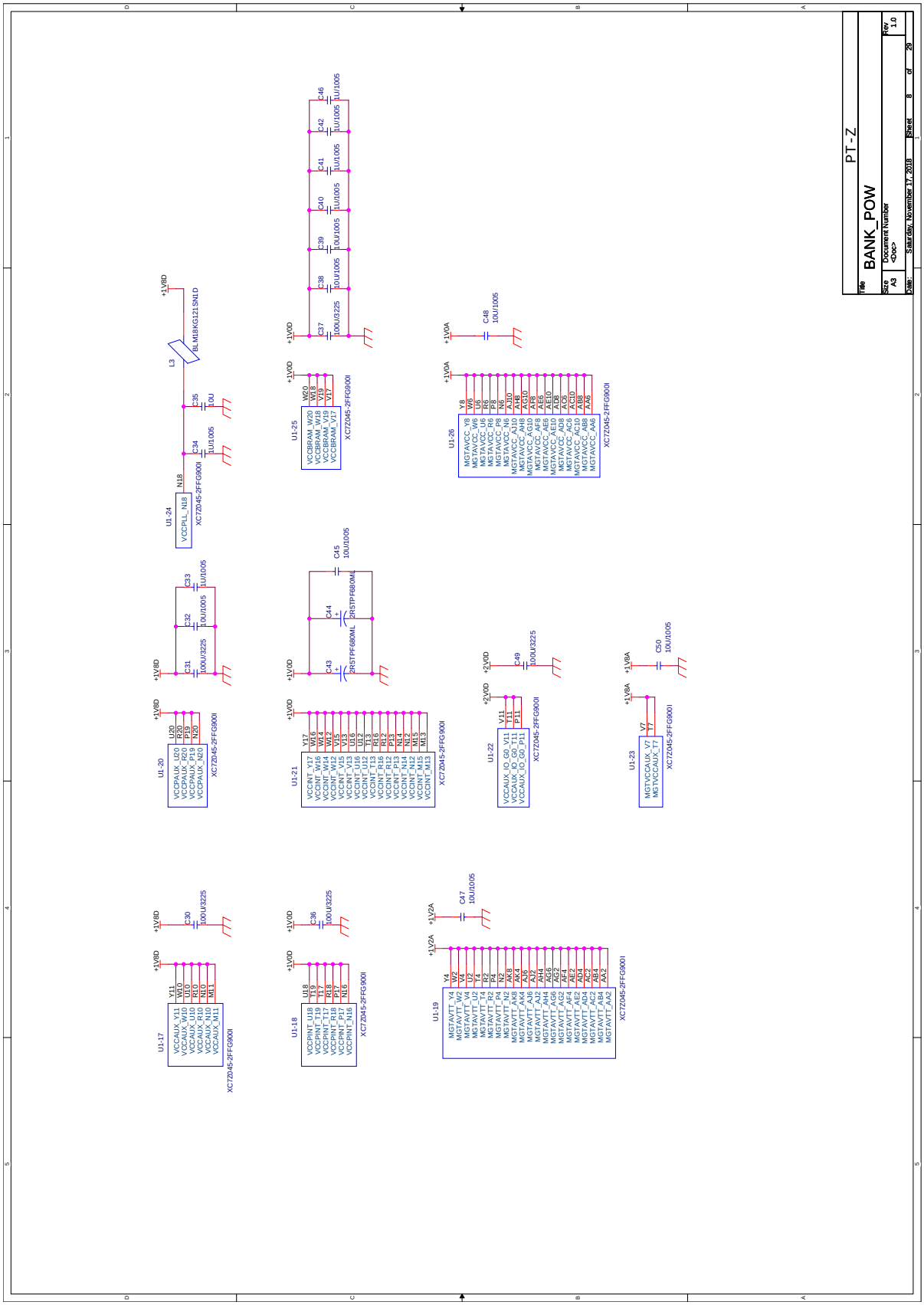

図 A.8　PTZ 回路図，Zynq の電源バンク．

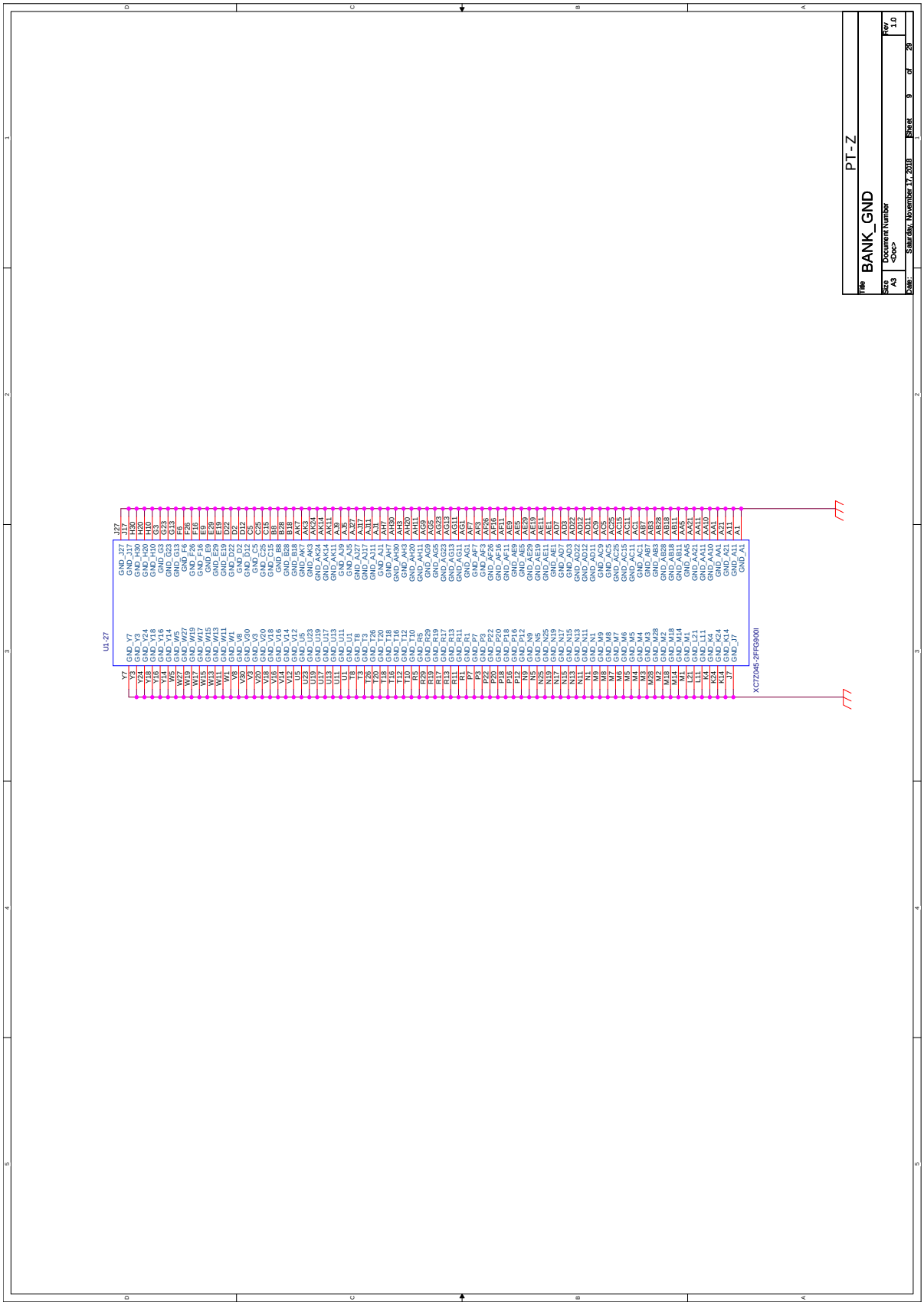

図 A.9　PTZ 回路図，Zynq の GND バンク．

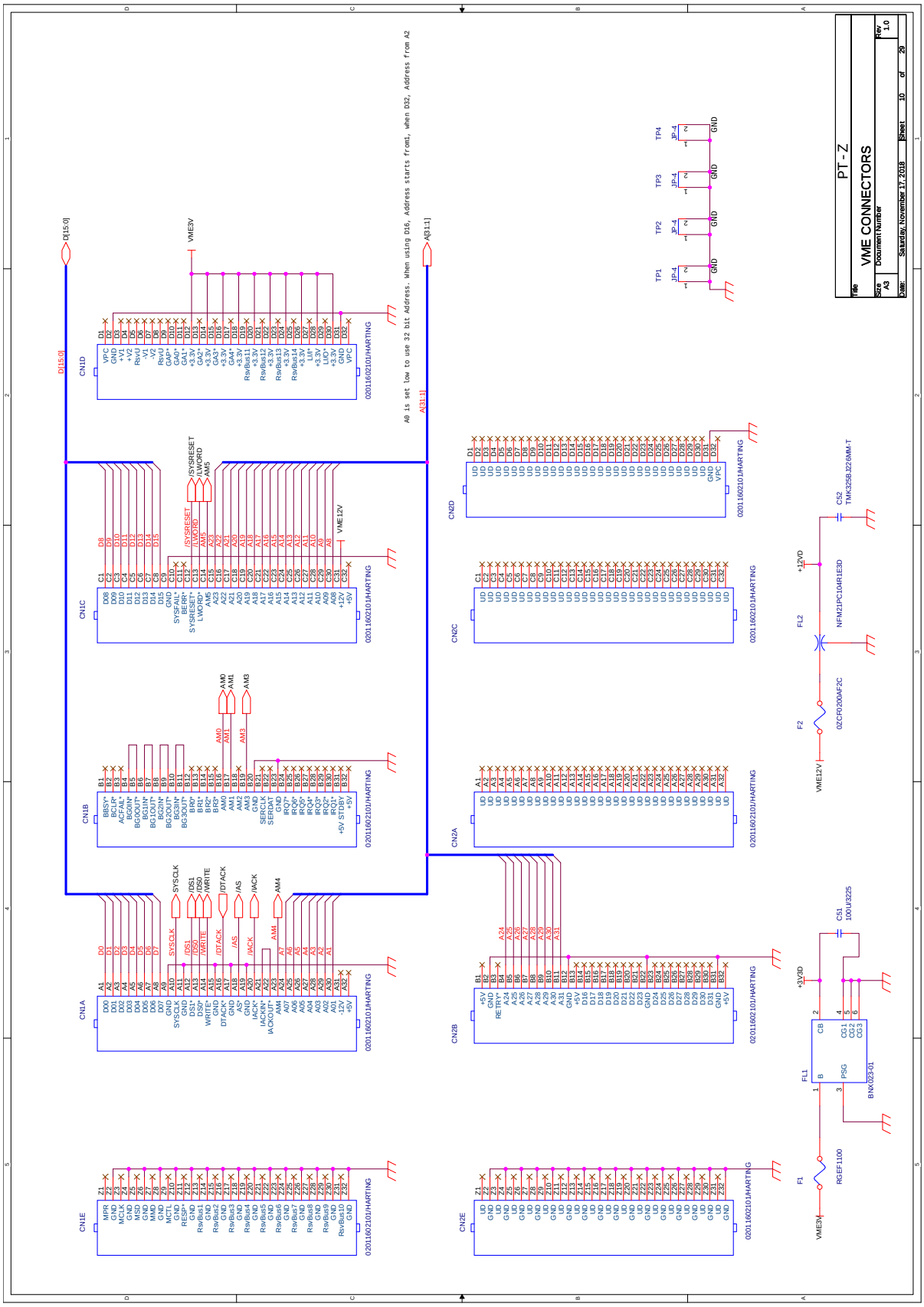

図 A.10　PTZ 回路図，VME コネクター.

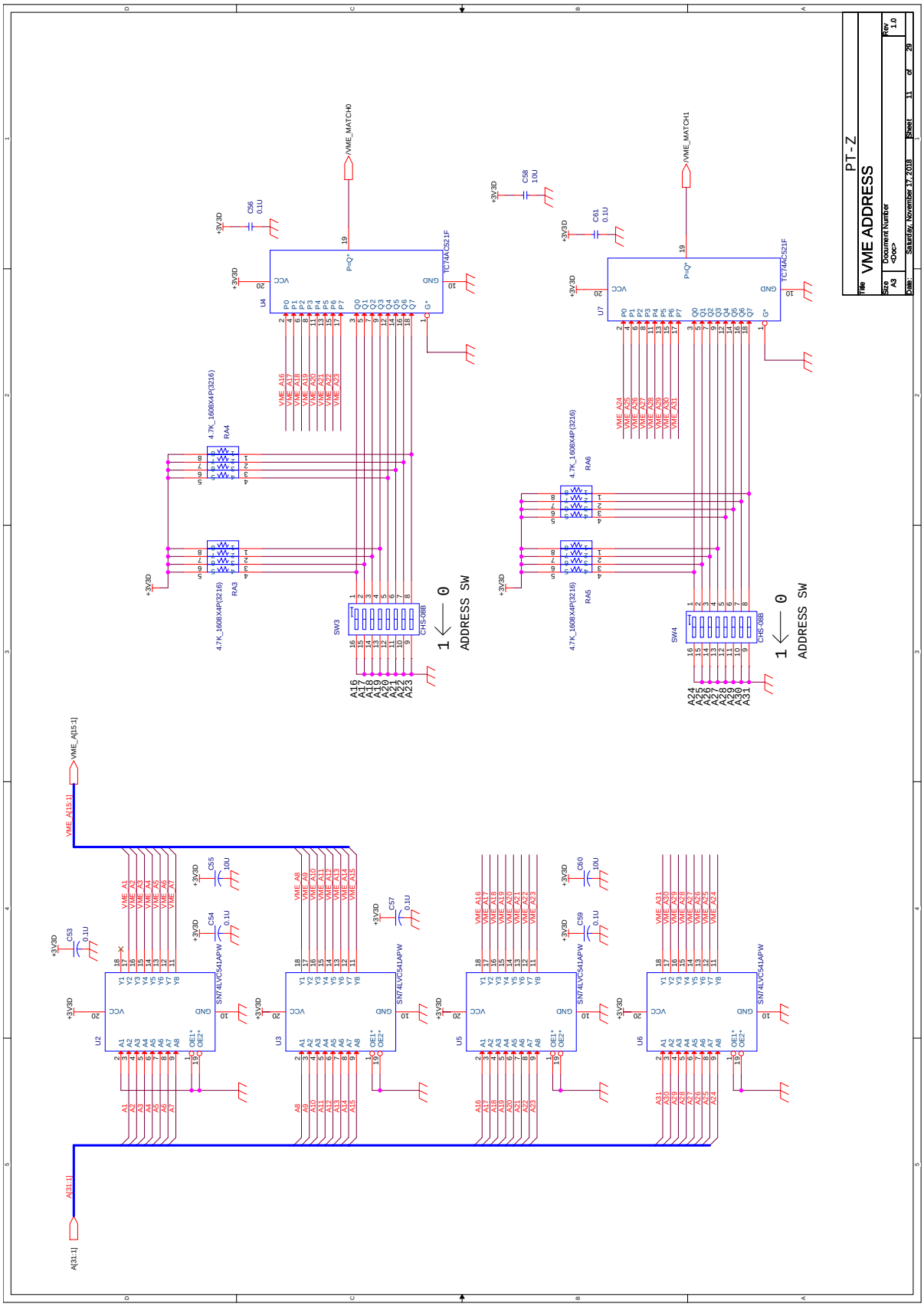

図 A.11　PTZ 回路図，VME アドレス線制御.

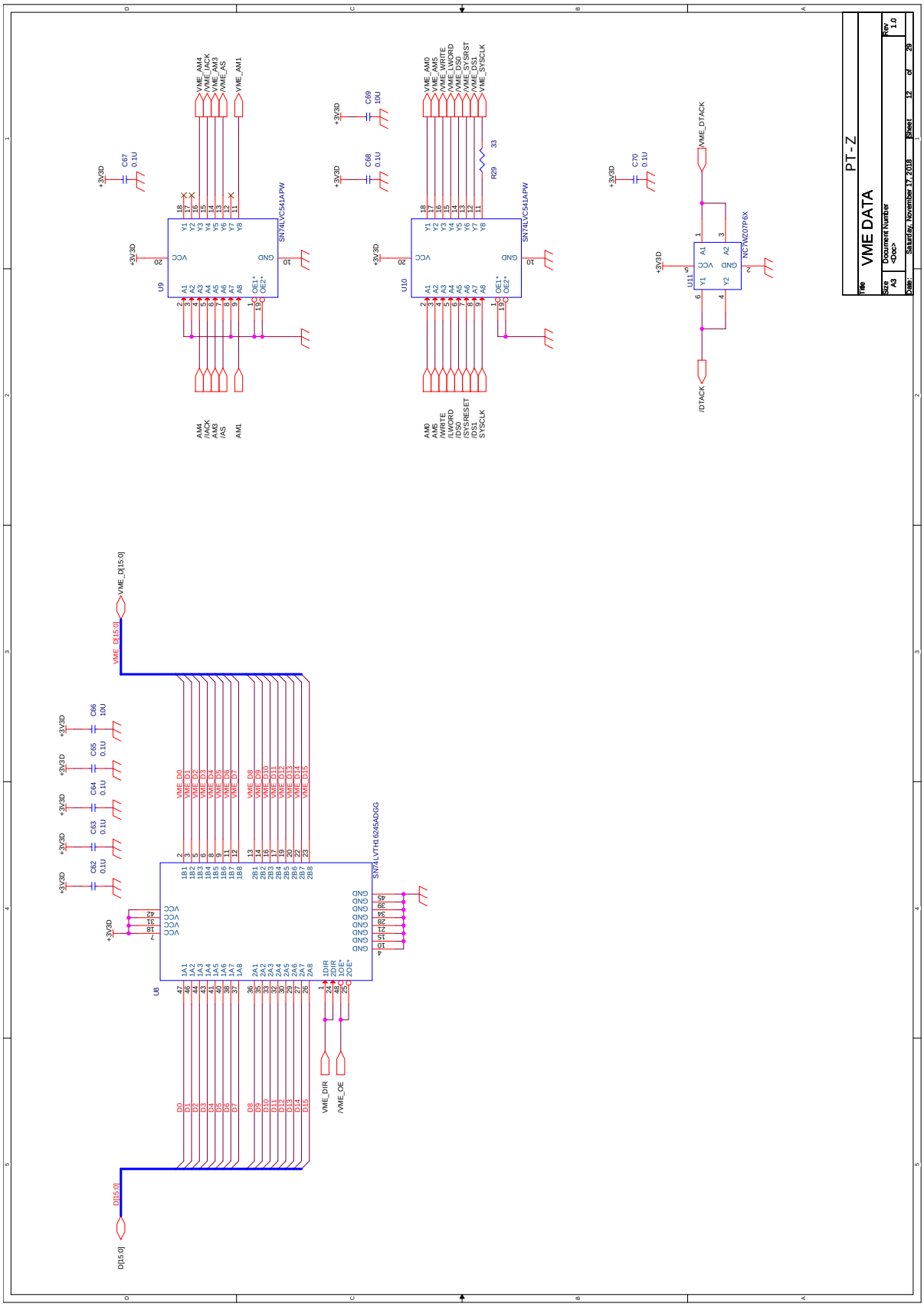

図 A.12　PTZ 回路図，VME データ線制御.

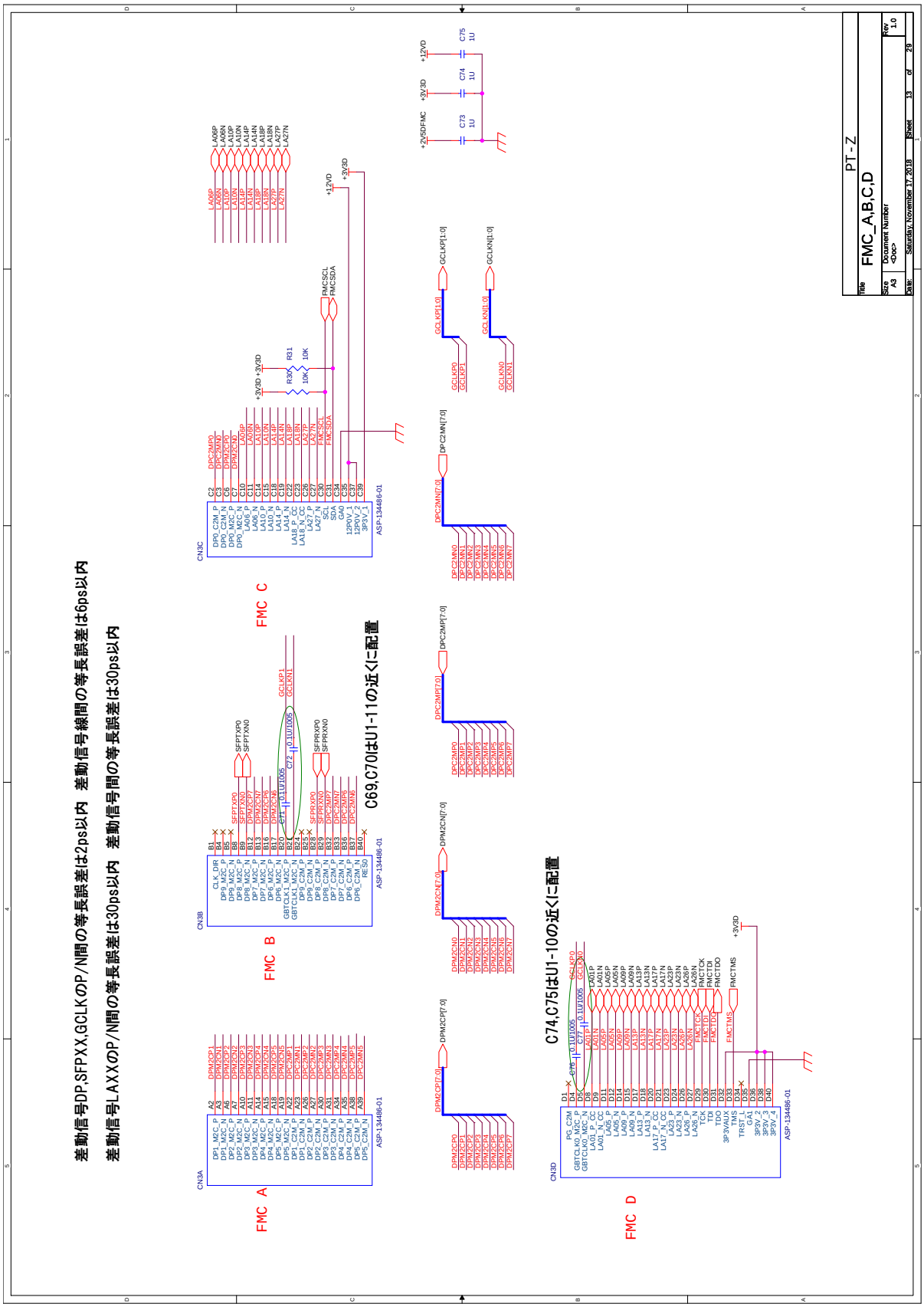

図 A.13　PTZ 回路図，メザニンコネクタ (A,B,C,D).

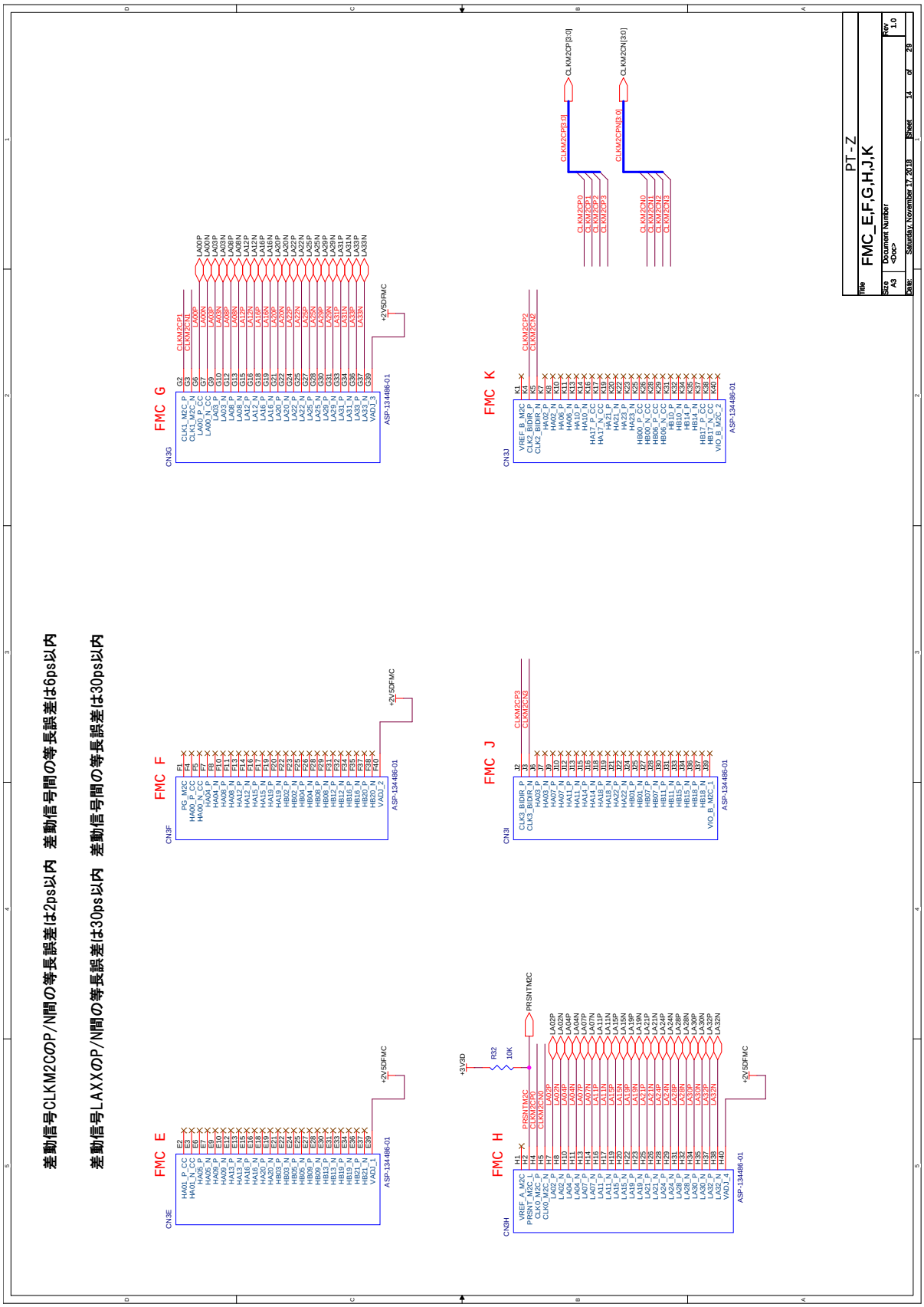

図 A.14　PTZ 回路図，メザニンコネクタ (E,F,G,H,J,K).

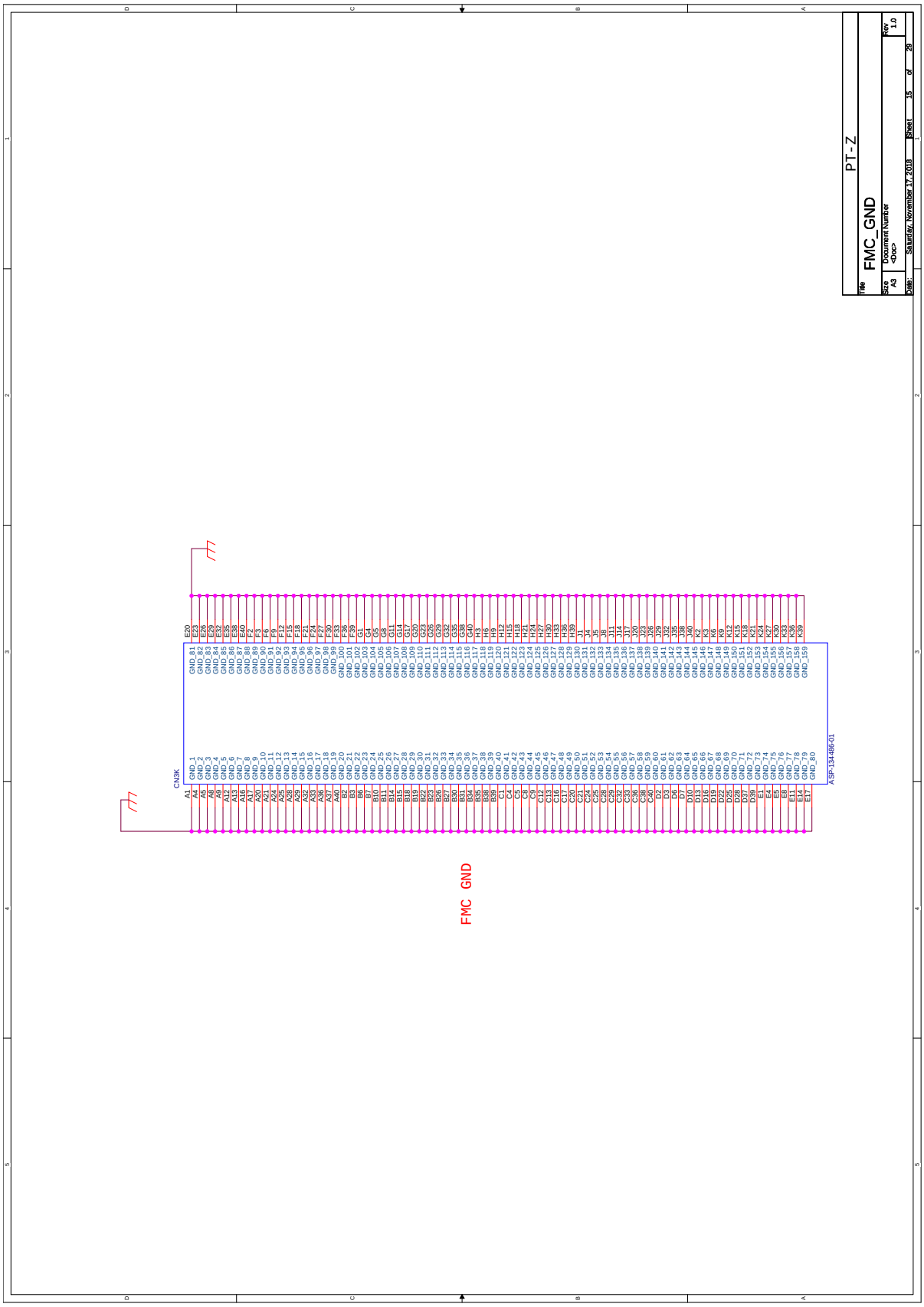

図 A.15　PTZ 回路図，メザニンコネクタ GND.

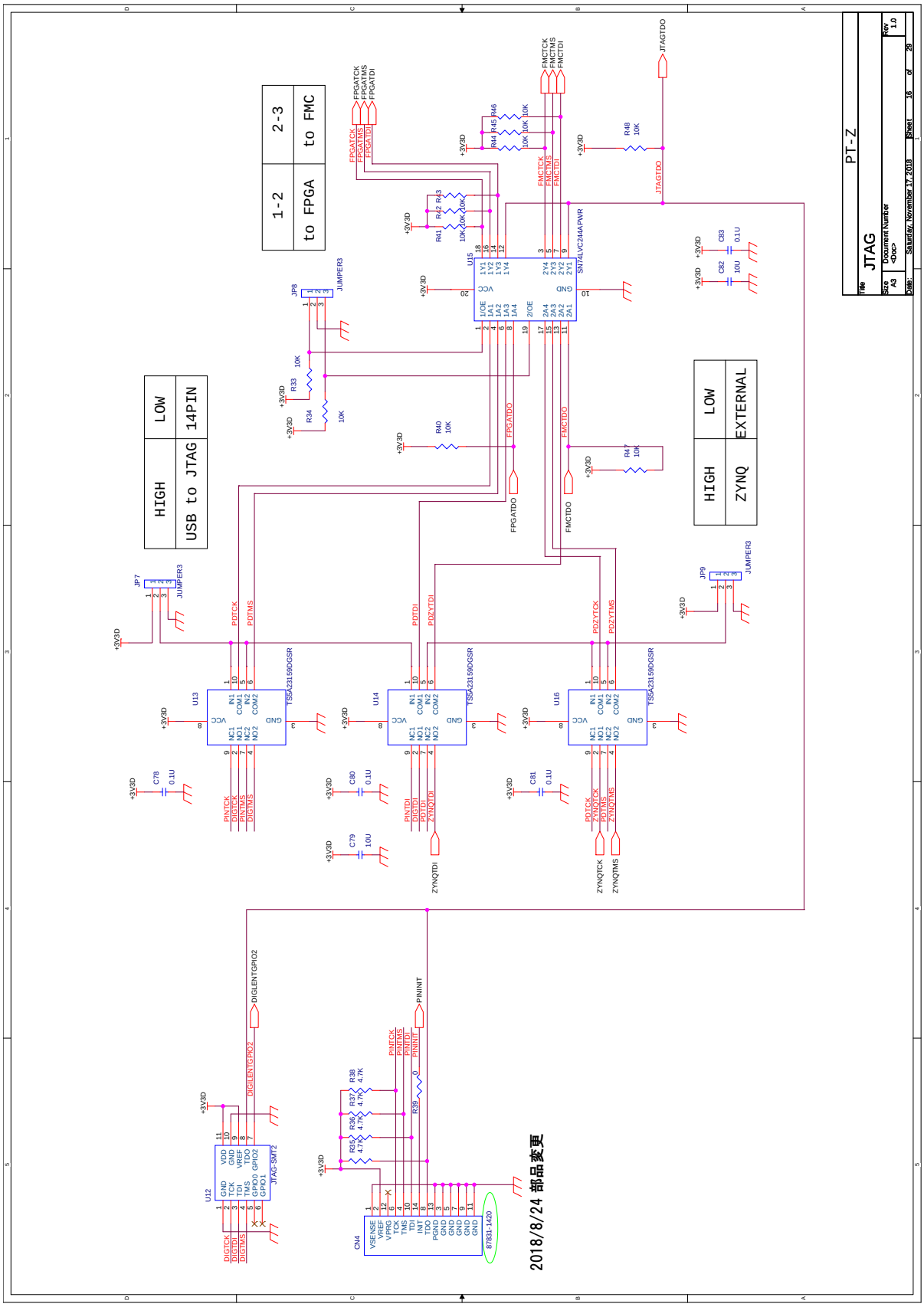

図 A.16　PTZ 回路図，JTAG コンフィギュレーション回路.

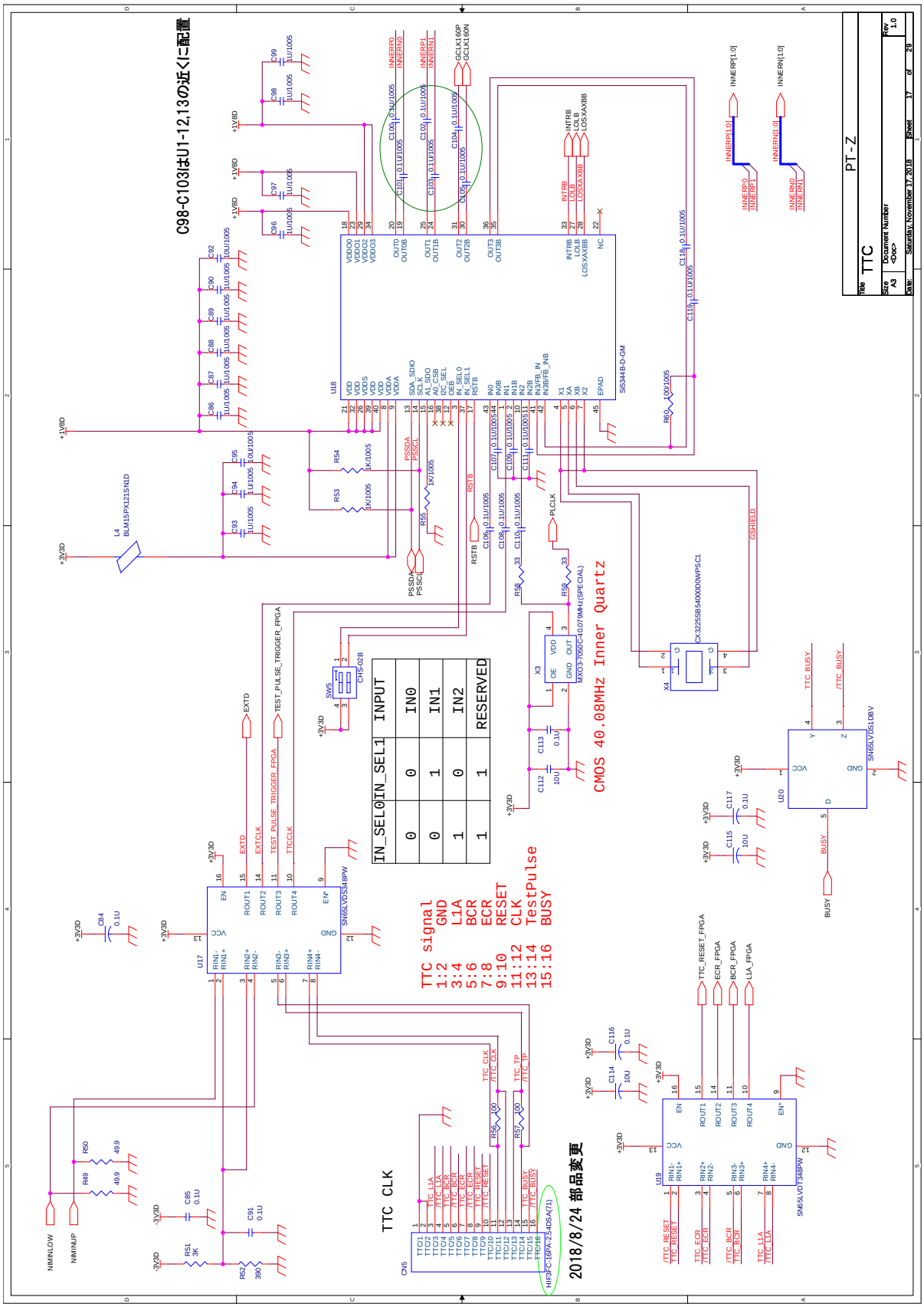

図 A.17　PTZ 回路図，Si5344 と外部/内部クロック周辺.

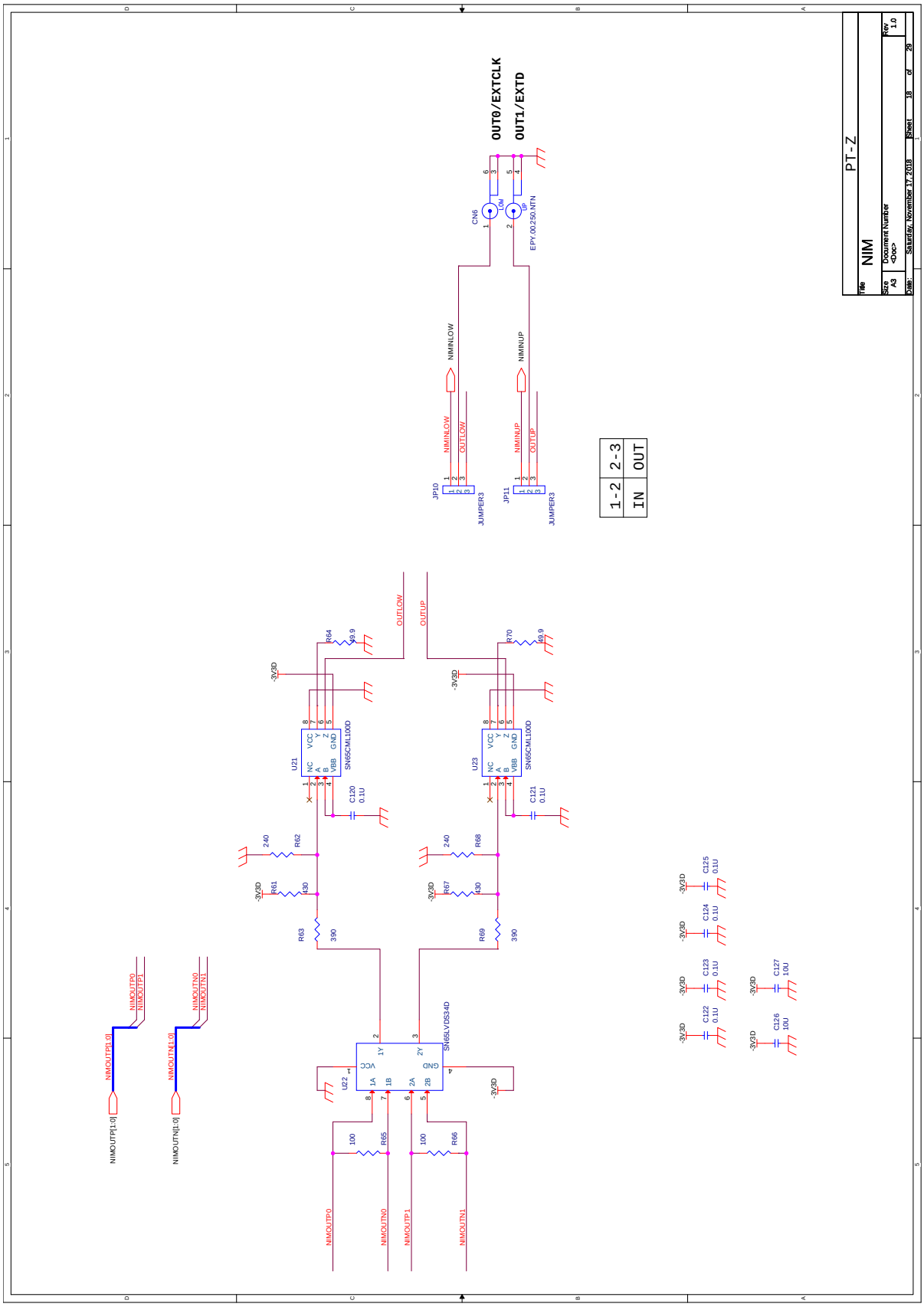

図 A.18　PTZ 回路図，NIM コネクタ.

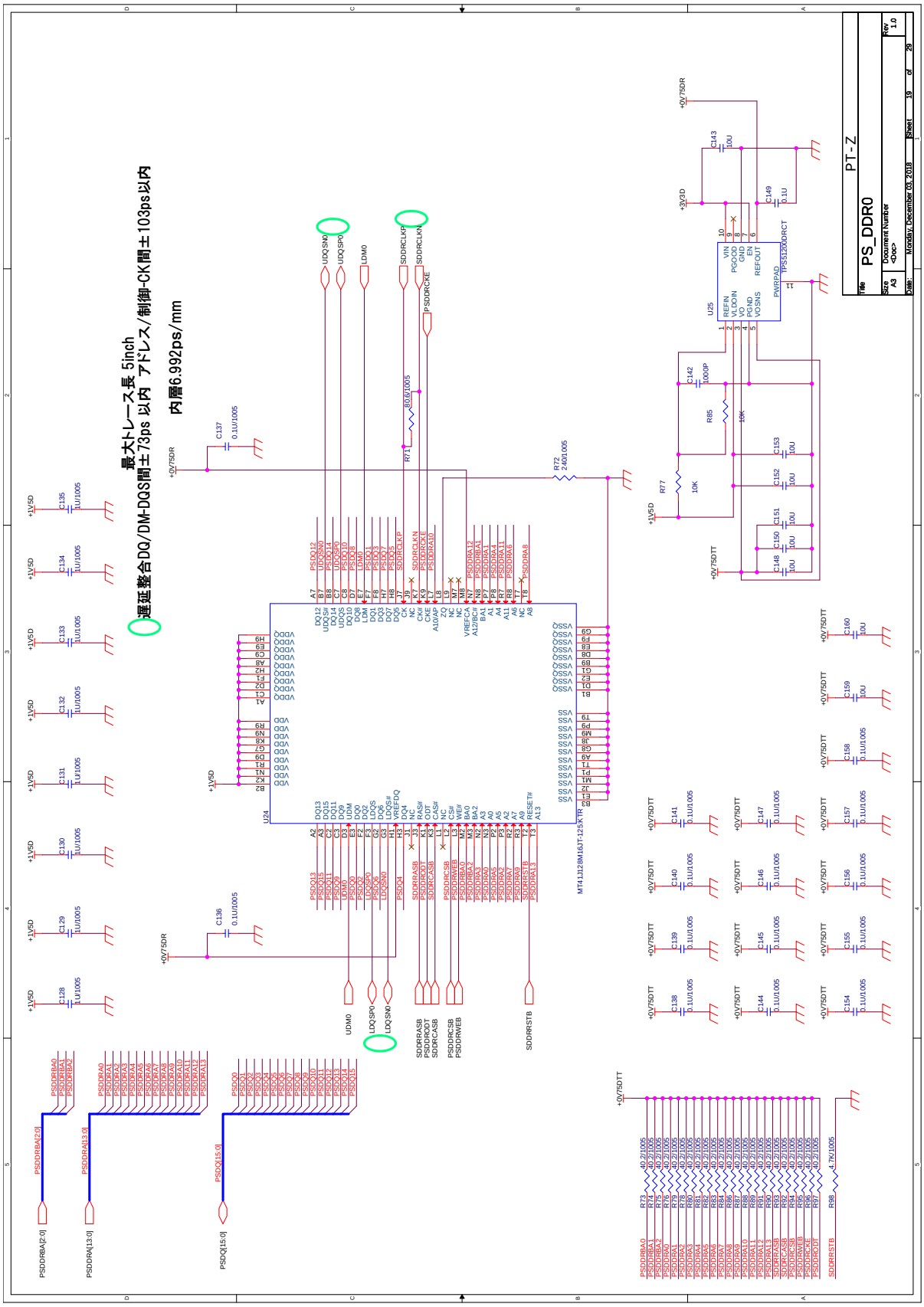

図 A.19　PTZ 回路図，PS DDR3 メモリ．

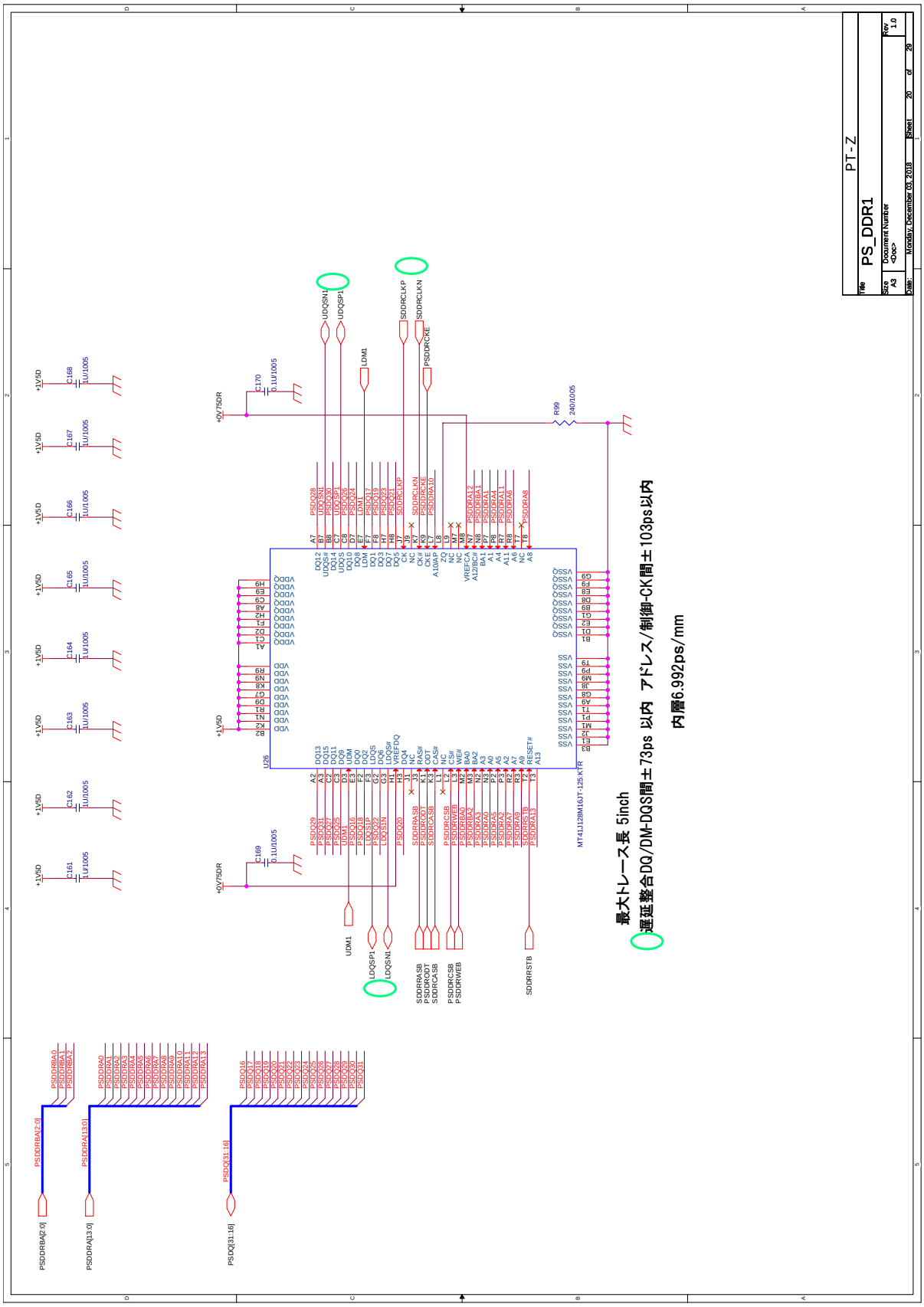

図 A.20　PTZ 回路図，PS DDR3 メモリ．

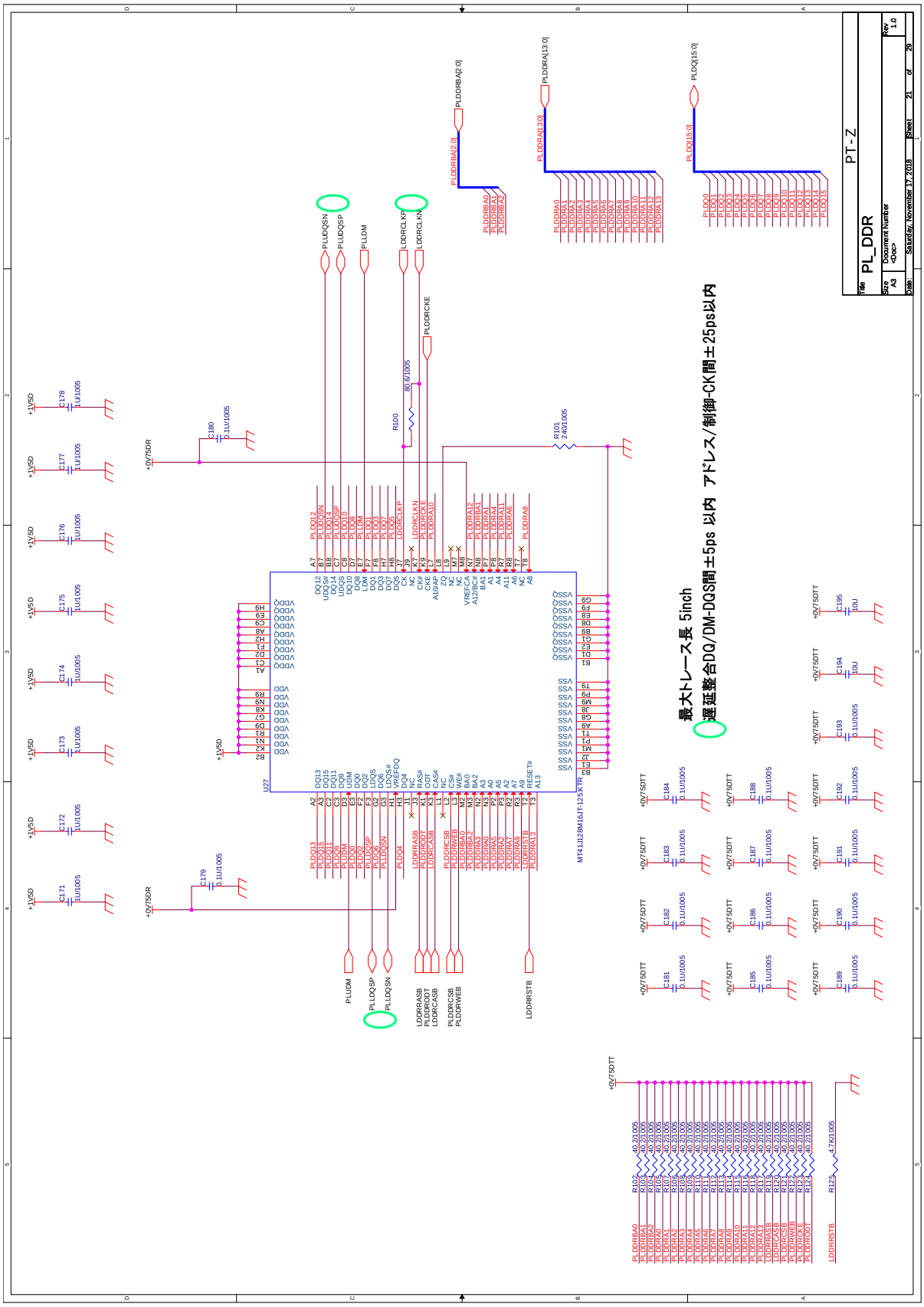

図 A.21　PTZ 回路図，PL DDR3 メモリ．

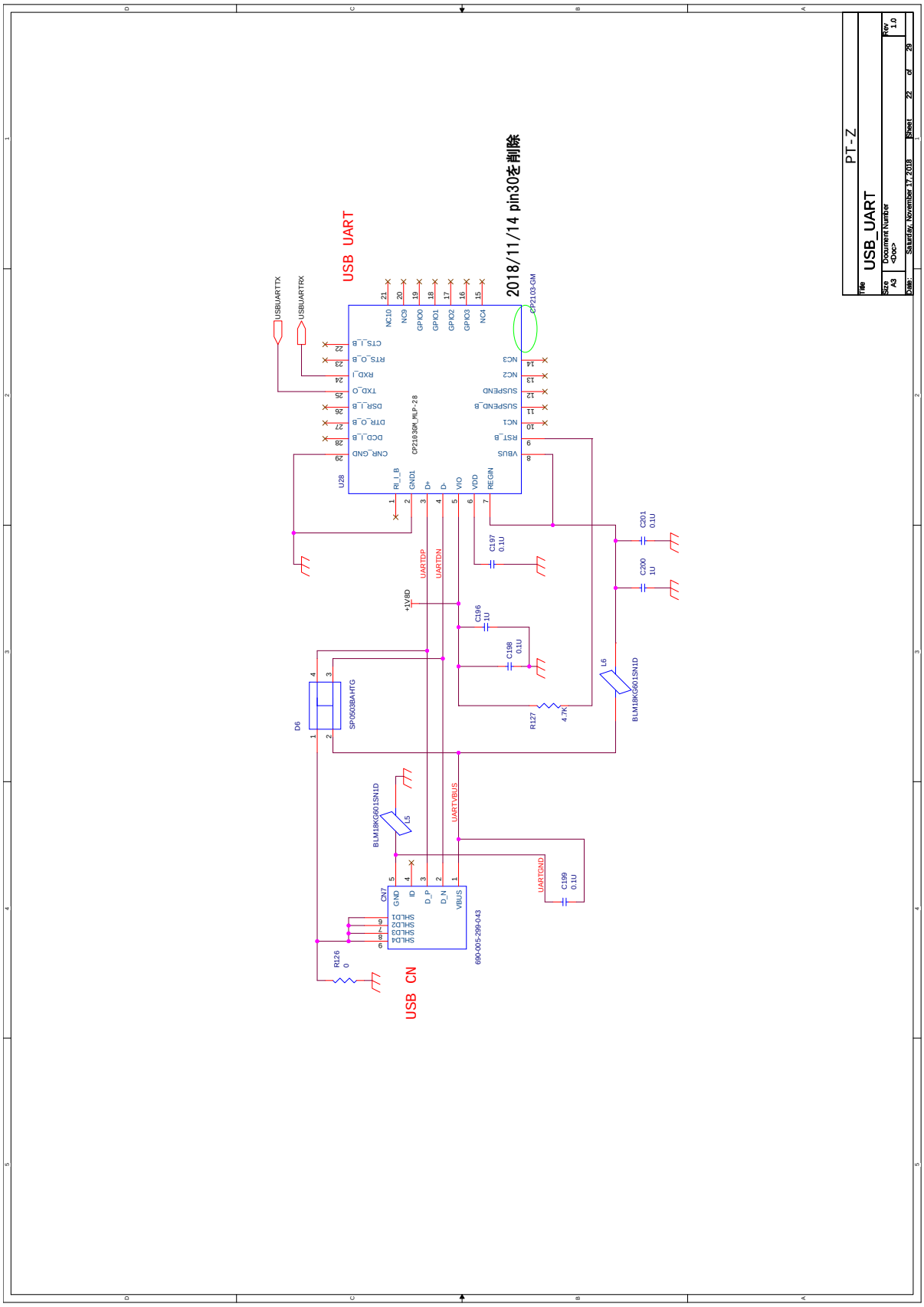

図 A.22　PTZ 回路図，USB-UART モジュール．

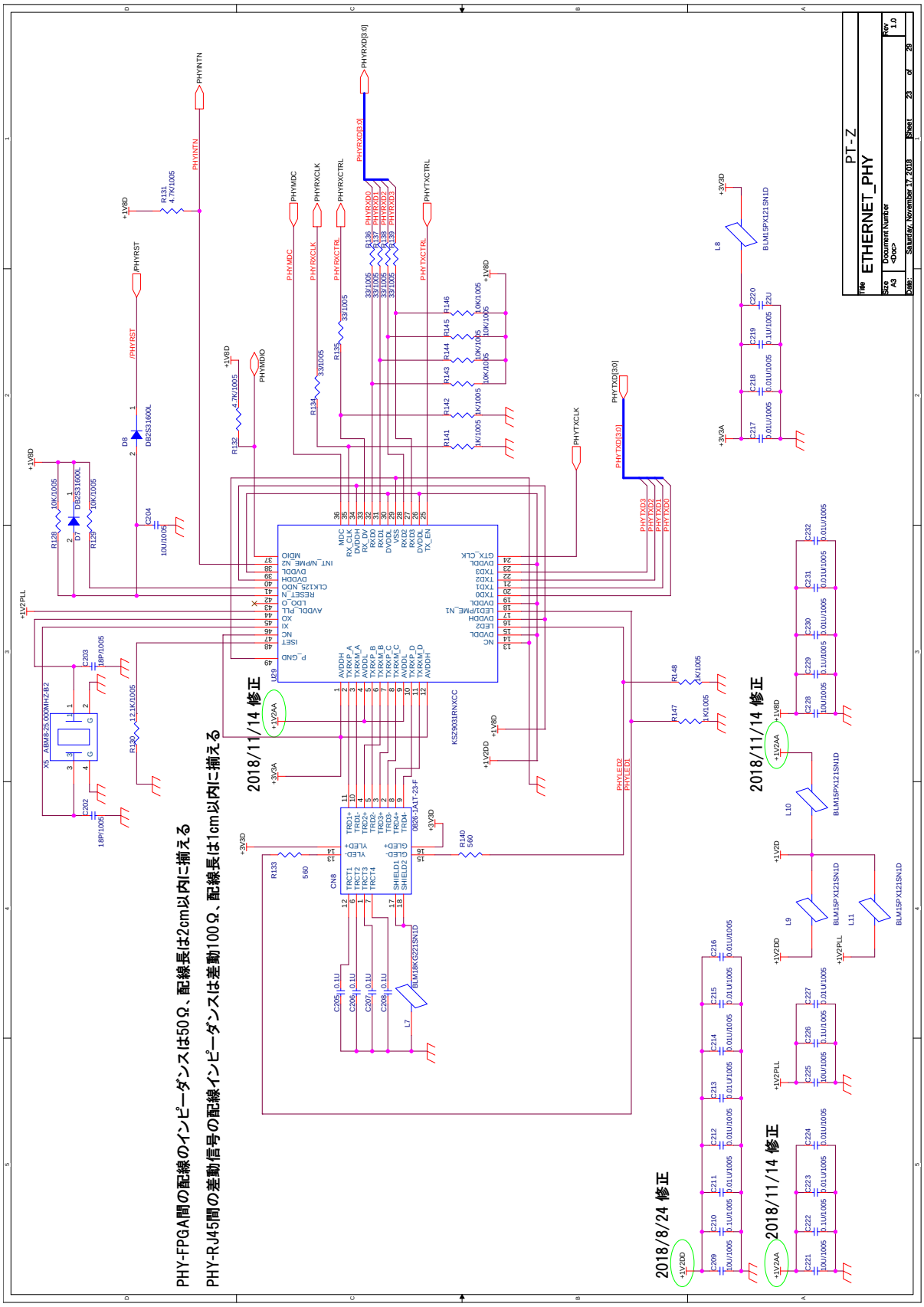

図 A.23　PTZ 回路図，RJ45 コネクタと PHY モジュール．

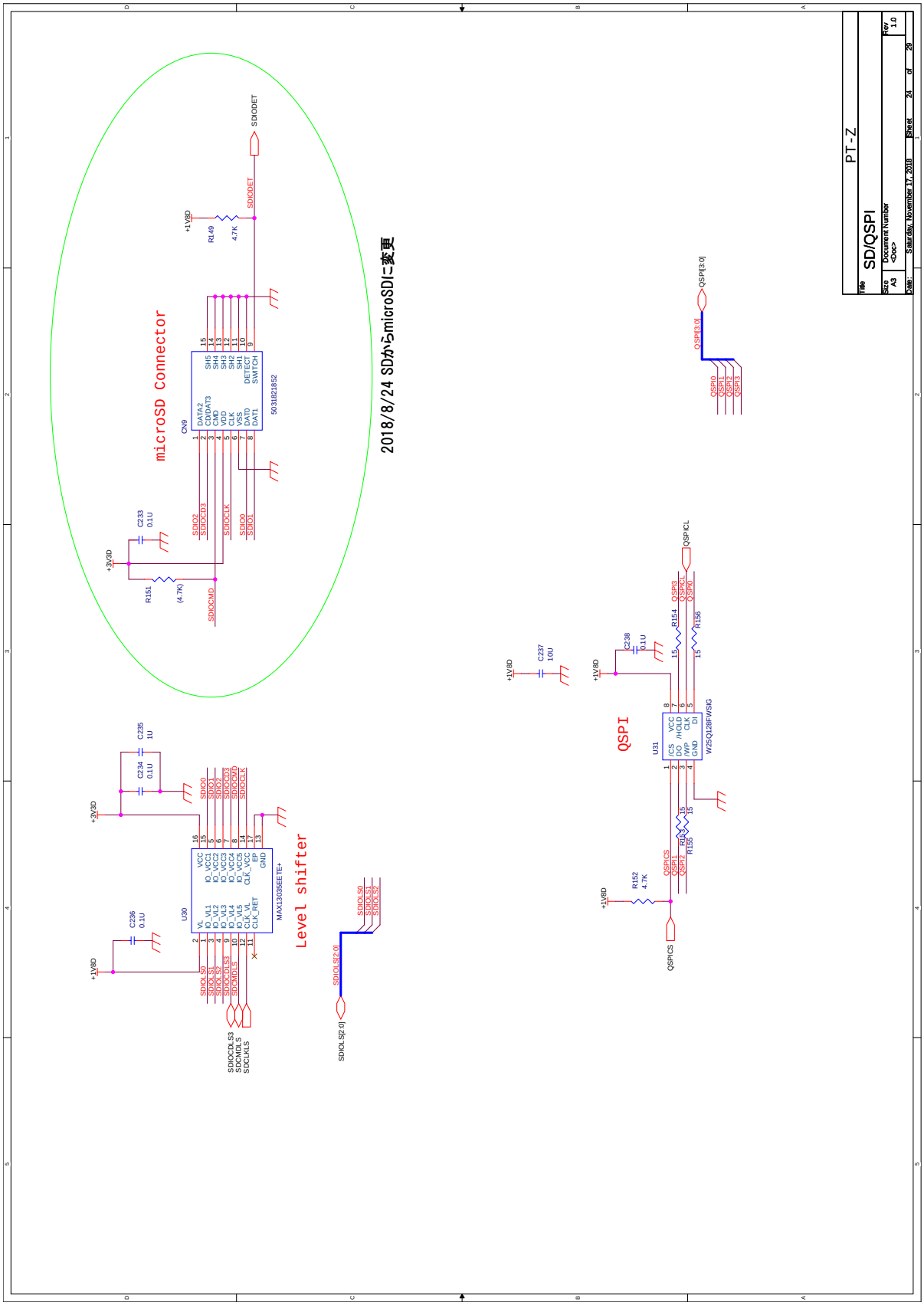

図 A.24　PTZ 回路図，microSD モジュールと QSPI フラッシュメモリ．

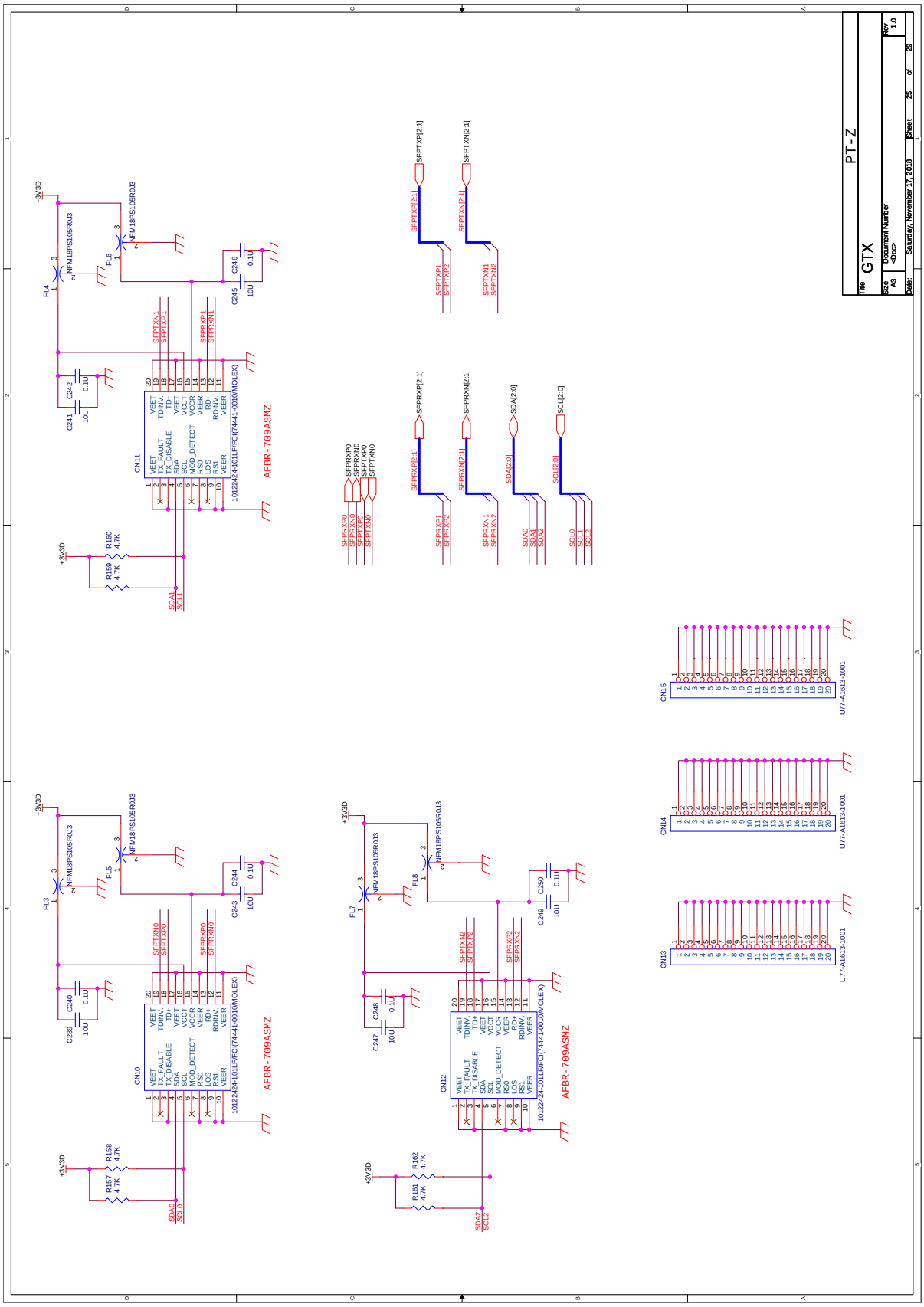

図 A.25　PTZ 回路図，GTX 通信用 SFP+ モジュール．1 台はメザニンコネクタに接続.

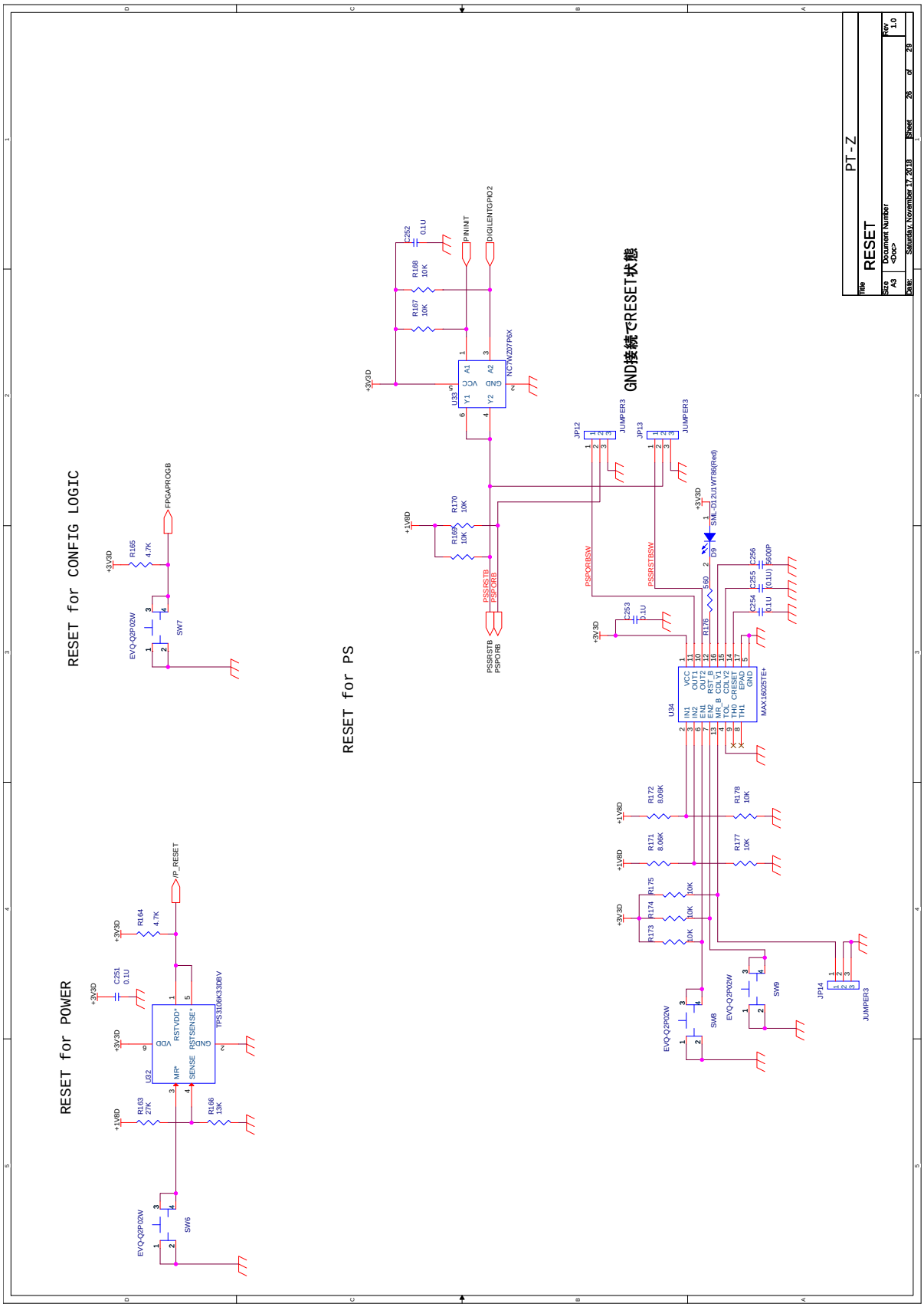

図 A.26　PTZ 回路図，PS/PL コンフィギュレーションリセットスイッチとボード電源リセットスイッチ．

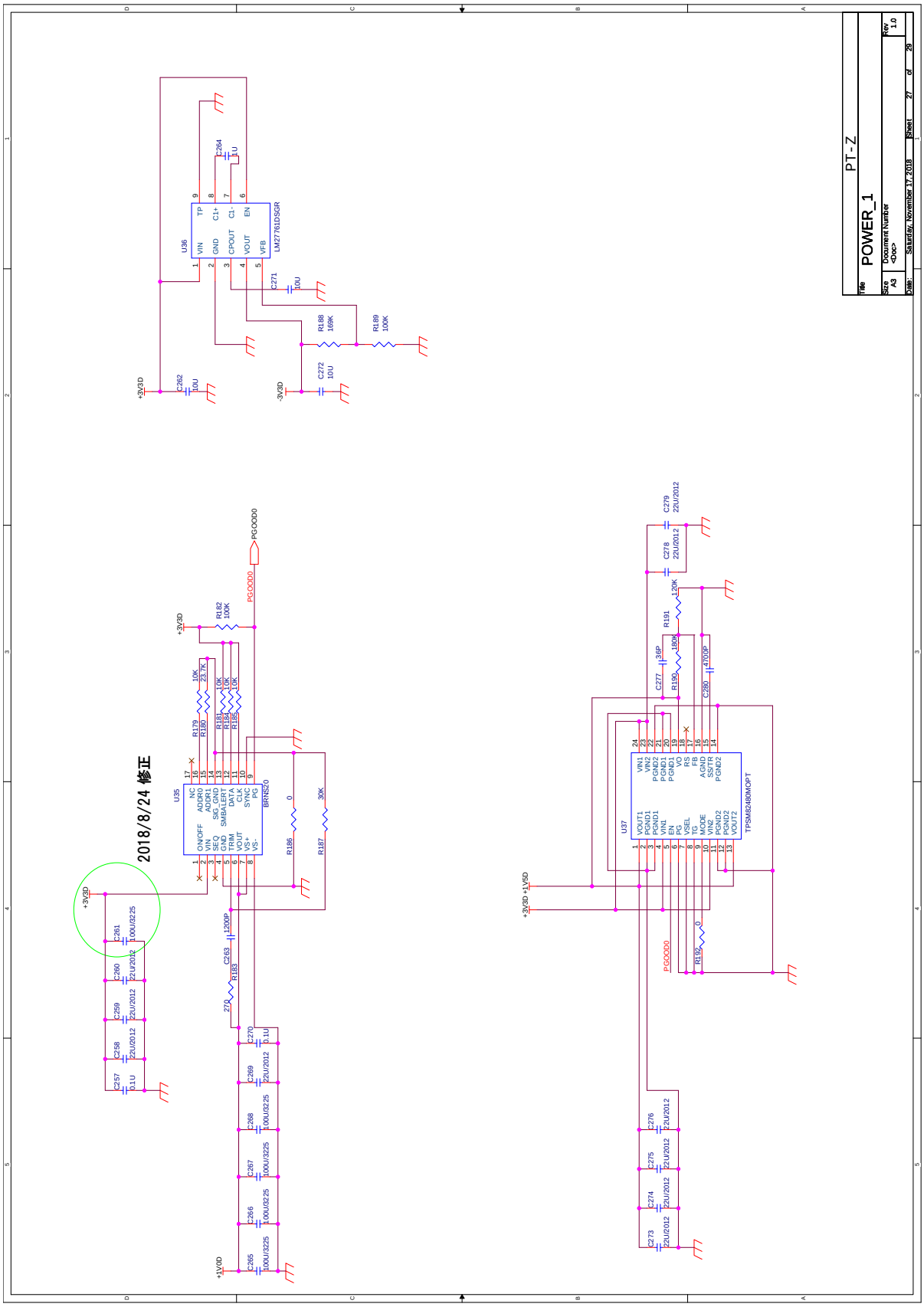

図 A.27　PTZ 回路図，電源レギュレーターその 1.

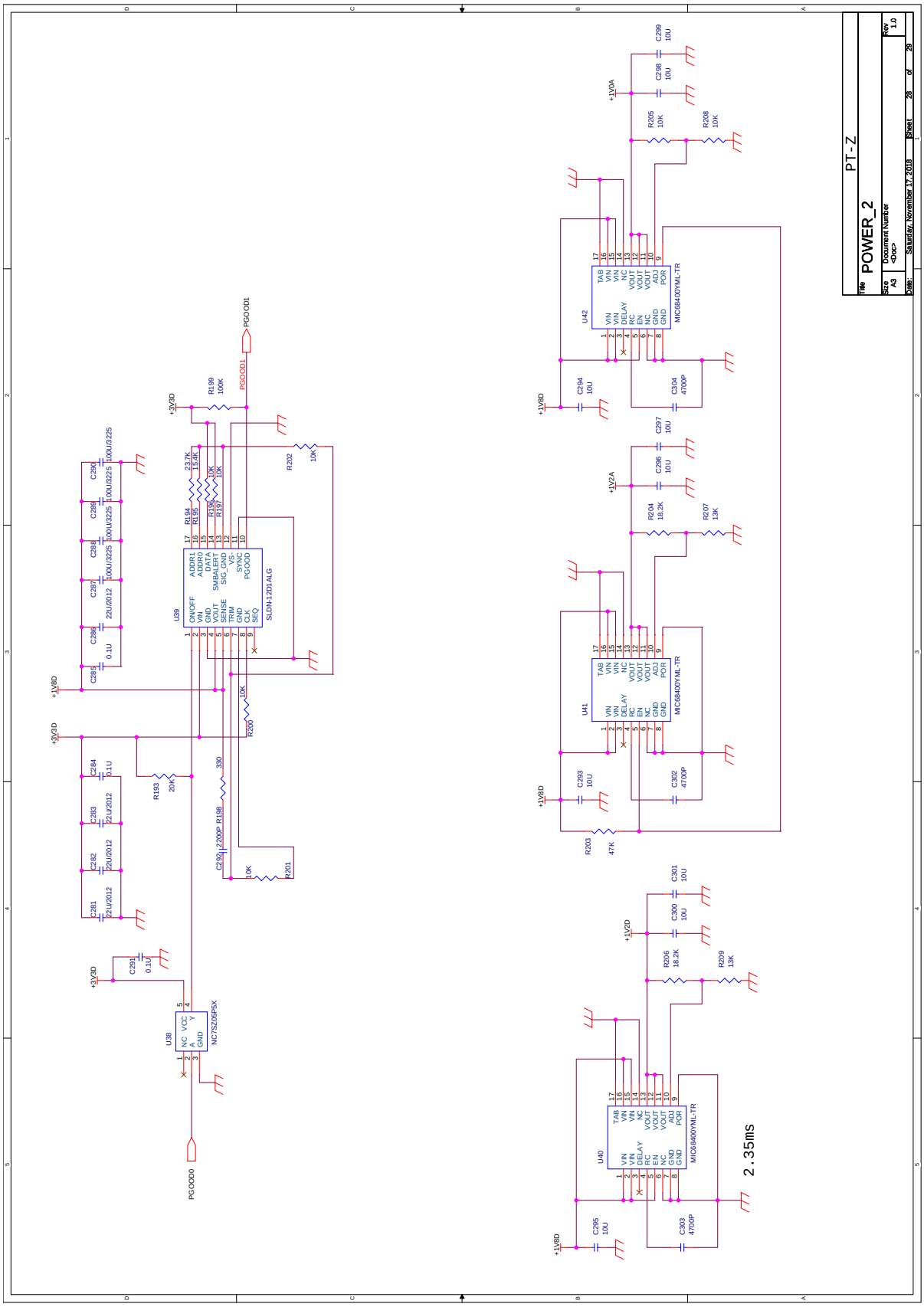

図 A.28　PTZ 回路図，電源レギュレーターその 2.

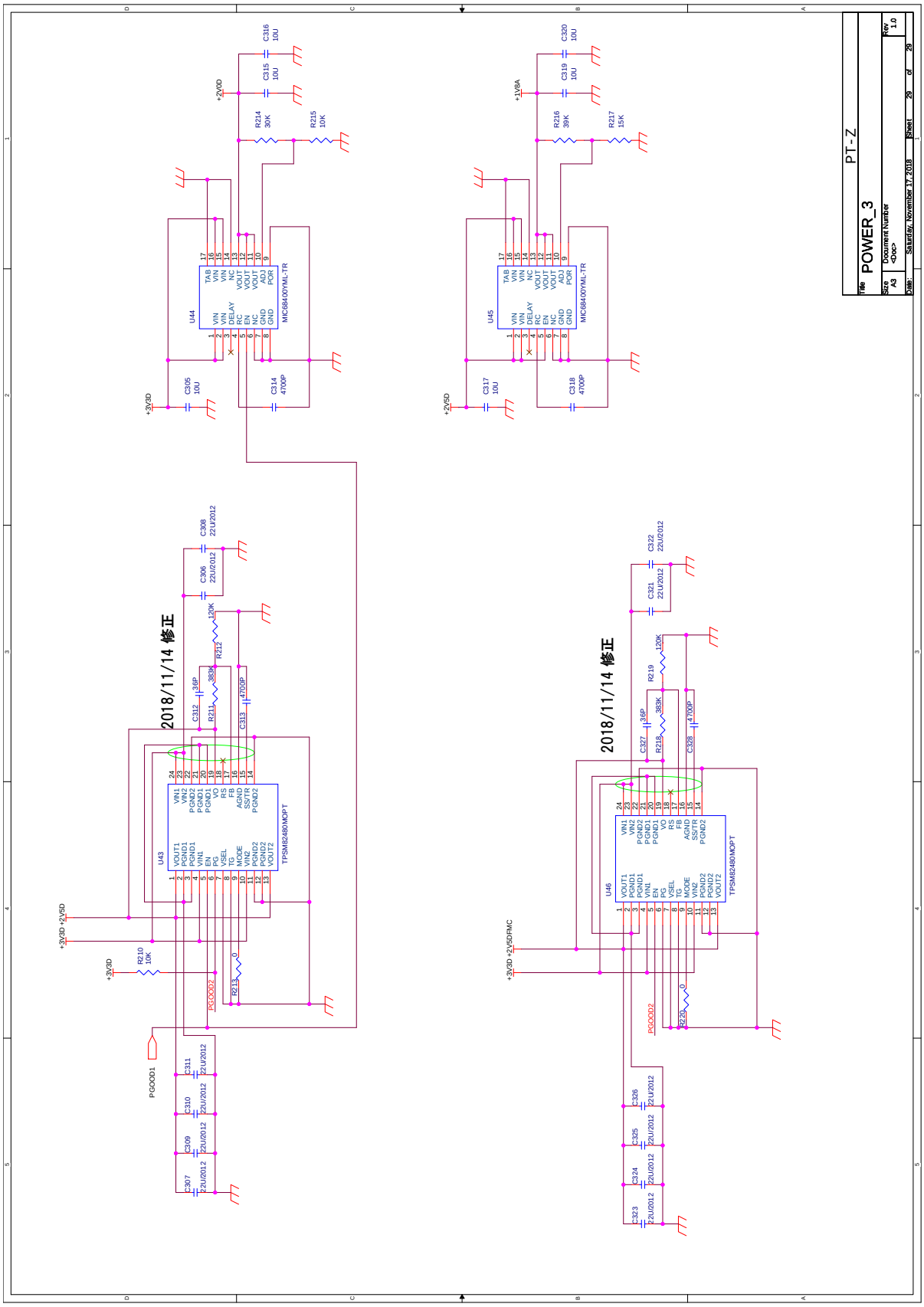

図 A.29　PTZ 回路図，電源レギュレーターその 3.