

EASIROC テストボード仕様書

東北大学 本多良太郎

2013年3月30日

履歴

11.15.2010 開発開始 回路図作成開始

11.17.2010 回路図 電源関係、及び SPIROC 周り整理 (レギュレータ、入力電圧変更)

11.21.2010 第0版 回路図完成

- FPGA を SPARTAN6 XC6SLX25 FG(G)484 に変更
- PROM を XCF08P FS(G)48 に変更
- PROM 用に+1V8D 電源追加
- I/O を NIM レベルに変換
- LVDS 信号リピータ追加
- Power reset 用 IC を TPS3103 に変更
- 発振器の周波数を 40MHz から 25MHz (SOY 推奨値) に変更

12.4.2010 第1版 回路図完成

- FIN1108 を SN65LVDS389 (Line Driver) に変更
- AD8672 を AD8602 に変更
- 基準電圧を REF194ESZ (4.5V) に変更して 1.8V を抵抗分割で作るように変更
- PROM のパッケージを FSG48 から VOG48 に変更
- FPGA VCCO2 を +2V5 から +3V3 に変更
- LVDS 出力をバンク 0 に移動
- フィルタコンデンサ追加
- 回路図上のミスとパーツ番号整理
- コネクタやスイッチなど型番を変更
- 発振器の周波数を 25MHz から 50MHz に変更

2.10.2011 最終版 回路図完成

- 回路図のミスを修正
- 基板レイアウトにあわせて FPGA 配線を修正
- 外部クロックを受けるために、IN_FPGA1 を GCLK に接続するように変更

3.11.2011 東日本大震災 開発一次中断（震災後しばらくして納品されたらしい）

5月頭.2011 テスト再開

6.9.2011 回路図にミスが見付かる

Raz_ch が差動ペアにアサインされてなかったので、FPGA 側で test15,16 を Raz_ch として、碁盤状のバイパス配線で対処。

7.3.2011 ASIC 名が SPIROC-A から EASIROC に変更

8月末.2011 一通りの機能の実装が終了する

11月.2011 Firmware v3.0 公開。EASIROC 講習会で配布する。

- DAQ busy が出るように改良
- たまに ADC が止まる問題を修正

12月.2011 Firmware v4.0 公開。一部に配布。

- ユーザーが任意のタイミングで ADC を止めれるように改良。

4.5.2012 Firmware v4.1 公開。

- ISE 13 以上でコンパイルすると ADC のヘッダーが壊れる問題を修正。

6.1.2012 Firmware v4.2 公開。

- 内部回路修正。機能、性能は 4.1 と変わらず。

8.30.2012 Firmware v4.3 公開。

- FPGA のスピードグレードが -3 になっていたのを、-2 に直した。

8.30.2012 Firmware v7.0 公開。

- MHTDC を実装。
- RUN gate、BUSY 出力の実装。
- HOLD、Common stop、TRIGGER 入力を分けて実装。
- Data header を 64 bit 長から 96 bit 長へ変更。

12.1.2012 Firmware v7.3.1 (E10_v5) 実装と運用開始。

- Fast Clear を実装。
- EventTag の受信を実装。
- 各種 Bug Fix。

4.1.2013 Firmware v7.3.1 (E10_v5) 公開

- v4.4 の公開を終了。

目 次

第 1 章 概要	1
1.1 開発メンバー	1
1.2 スケジュール	1
1.3 概要	1
1.4 開発背景	2
1.5 検出器とシステム	2
1.6 基本動作	2
第 2 章 EASIROC	4
2.1 全体の構成	4
2.1.1 InputDAC	6
2.1.2 PreAMP	7
2.1.3 slow shaper と fast shaper	7
2.1.4 電荷測定方	9
2.1.5 時間測定 (discriminator)	10
2.1.6 信号プローブ	11
2.1.7 テスト入力	12
第 3 章 PCB	13
3.1 主な機能 (要求仕様)	13
3.2 形状	13
3.3 インターフェース	14
3.4 電源	14
3.5 主要部品	15
3.6 ブロック図と完成したボード	16
3.7 動作	16
3.7.1 信号モニター モード	17
3.7.2 DAQ モード	18
3.8 レイアウト時に気を付けたこと	19
3.9 基板レイアウト図と修正記録	19
第 4 章 FPGA	20
4.1 使用 FPGA	20
4.2 ブロック図	20
4.2.1 EASIROC 制御部	21

4.2.2	DAQ 部	21
4.2.3	インターフェース部	23
4.2.4	SiTCP 部	23
4.3	各部動作説明	24
4.3.1	SlowControl_Spirom2_ver2	24
4.3.2	ADC_AD9220dual_ver3	25
4.3.3	MHTDC	26
4.3.4	User IF	28
4.3.5	SiTCP_RECV	29
4.3.6	SiTCP_WRITE	29
4.4	設計時に注意したこと、工夫したこと	30
4.5	失敗したこと、修正が必要な事	30
4.6	開発ツール	31
第 5 章	部品リスト	32
第 6 章	初回電源投入時の試験項目	34
6.1	テスト入力を用いた試験	34
6.2	MPPC への電圧印加に関して	34
第 7 章	ユーザーガイド	36
7.1	各部説明	36
7.1.1	ジャンパピン	36
7.1.2	アナログ信号入出力	37
7.1.3	デジタル信号入出力	38
7.2	初回電源投入	39
7.2.1	ジャンパピンの設定	39
7.2.2	電源	40
7.2.3	ソフトウェア実行と EASIROC のコンフィグ	40
7.3	ソフトウェアの使い方	42
7.3.1	Transmit SC	43
7.3.2	Transmit Read SC	44
7.3.3	Transmit Probe	44
7.4	モニターモードでの使用	45
7.5	DAQ モードでの使用	46
7.5.1	DAQ 信号のタイミング調整	47
7.5.2	データ構造	48
第 8 章	レジスター一覧	50
8.1	ASIC Initialize で使用されているレジスタ	50
8.1.1	word_0.txt 内	50
8.1.2	word_1.txt 内	51

8.1.3 word_3.txt 内	51
8.2 Slow control register	52
8.3 Read slow control	55
第 9 章 既知の問題点	56
第 10 章 良くある質問	57

第1章 概要

1.1 開発メンバー

東北大学

本多良太郎 三輪浩司

KEK

家入正治 池野正弘 内田智久 斎藤正俊 田中真伸 中村勇 吉村浩司

1.2 スケジュール

- 2010年11月21日までに第0版の回路図完成。
- 2010年12月中に回路図の最終版完成。レイアウトを始めてもらう。
- 2011年1月中にレイアウトを含めて最終決定。発注。
- 2011年3月末までの納品を目指す。

1.3 概要

多チャンネル MPPC 読出し用 ASIC である EASIROC を用いた MPPC 読出し回路の開発を行う。EASIROC はフランスで開発された PPD 読出し用の 32ch の入力を持つ ASIC であり、新造のチップであるためその制御方法の確立が本回路を開発する最大の目的である。EASIROC は MPPC のバイアス調整、アンプ、discriminator といった MPPC を読出すための基本的な機能を備えており、ADC を基板上に配置し MHTDC を FPGA 内に実装することで、ひとつのボードで ADC、及び TDC の両方が取得出来る回路の開発を目指す。また、J-PARC K1.8 の DAQ システムがネットワークに基づいているため通信規格に SiTCP を採用し、ネットを介した制御、及びデータ転送の手法の確立を目指す。

1.4 開発背景

多チャンネルのMPPCの読み出しを可能にすることによって、様々な分野に応用することが出来る。シンチレーションファイバーをMPPCで読み出すことによってイメージングが可能になり、またファイバーによる飛跡検出器としても応用することが出来る。我々はこの多チャンネルファイバー読み出しの技術を用いてハイペロン陽子散乱実験に応用し、超高速イメージ撮像管システムとして用いたり、高ビーム強度下のトラッキング検出器として動作させることを考えている。しかしながらMPPC用の多チャンネル読み出し回路はクロック同期で動作するものしか存在せず、ハイペロン陽子散乱実験のような非同期条件下で利用できるような汎用回路が未だ開発されていない。そのため、Open-Itの枠組内でKEK測定器開発室およびエレクトロニクスシステムグループとの合同により上記要求を満たす多チャンネルMPPC用の読み出し回路の開発を行う。

1.5 検出器とシステム

本回路はテスト基板であるため、特定の検出器構成やDAQ構成に対する制約を持たない。一般的にはMPPCを読み出し、NIMレベルでDAQ信号をやりとりし、ネットワークでデータを取得するという使用法をとる。

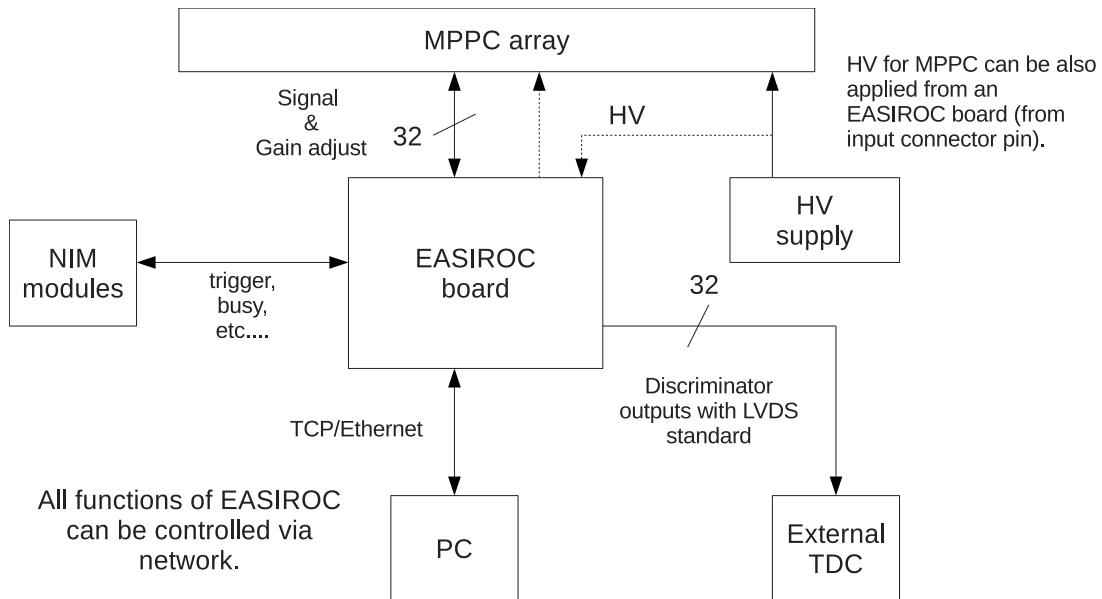


図 1.1: 本回路と検出器システムとの関係図

1.6 基本動作

読み出しボードのブロック図を図1.2に示す。本回路は主にアナログ処理用のASICであるEASIROCチップと、ASIC制御やDAQ制御のためのFPGA、及びADCで構成される。ア

ナログ入力にはツイストペアケーブルを用いる。片方が信号線であり、もう片方はMPPC用のバイアス供給としてもGNDとしても利用することが出この技術を広げることができる。来る。これは後述するEASIROCのバイアス調整機能が信号線を利用するためであり、MPPCへのバイアスの印加方法とは独立なためである。HVコネクタはペア線の片側をバイアスとして利用するために設置されている。MPPCのバイアス調整はEASIROCチップによってチャンネル毎に行うことが出来る。バイアス調整にはInputDACとうDACを利用する。その機能と利用法に関しては後ほど詳しく述べる。EASIROCチップに入力された信号は成形増幅されて電荷と時間測定に使用される。EASIROCの増幅部分は異なった増幅率を持つhigh gainとlow gainの2つに分かれており、160 fCから320 pCまでの入力レンジをカバーする。ADCが基板上に2つ設置されている理由は、このhigh gainとlow gainを同時にA/D変換するためである。電荷測定法の項で詳しく述べるが、ADCのラインはシリアルでの信号出力となる。また、EASIROCにはdiscriminatorも内蔵されており、その信号はパラレルで出力されFPGAに直結している。

FPGAはASICの制御と外界との通信を担う。本回路はSiTCPを利用してTCP/IP通信で制御及びデータ転送を行うが、SiTPCの機能は内蔵ではなく外部の基板(SOY BBY-006)に任せられているため、FPGAはSOYと会話し動作する。FPGA内にはMHTDCが実装されており、1 ch = 1 nsの精度で時間を測定することが出来る。EASIROCからのdiscriminator信号はそのままFPGAから外部へ出力される他に、MHTDCの入力、簡単なTrigger生成などに利用される。

本回路はNIMレベルの5つの入力と4つの出力を有しており、ユーザーが種々のDAQ信号のやりとりに利用できる。また、FPGAから出力されるdiscriminatorの信号はLVDSに変換され基板外へと出力される。そのため、LVDSが読めるTDCに接続すればそのまま時間情報を得ることが出来る。

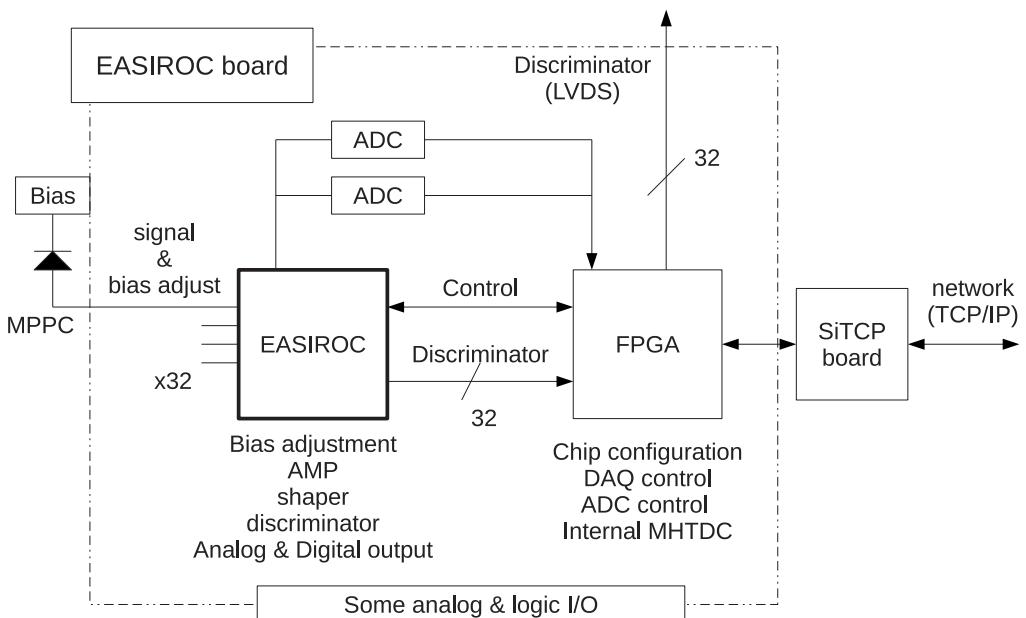


図 1.2: ボードブロック図

第2章 EASIROC

より詳しいEASIROCの仕様に関しては、開発元の仕様書を参照のこと。

2.1 全体の構成

ここではEASIROC全体の信号処理を簡単に説明する。各部位の詳細な動作に関しては後述する。EASIROCはOmega/IN2P3によって開発されたMPPC用の読み出しASICであり、1チップで32個のMPPCの読み出しが出来る。また、EASIROCはMPPCを読み出すための基本的な機能である、バイアス調整機能や、成形増幅器、discriminatorなどを全て有している。EASIROCの内部回路の概要を図2.1に示す。EASIROCへの入力信号の極性は正電圧にしなければならない。そのため、MPPCのカソードへ正電圧を印加しアノード側をEASIROCへの入力として使用する。入力ラインの最初に配置されているInputDACはMPPCのバイアス調整用に使用され、チャンネル毎に電圧を設定できる。入力信号は2つのキャパシタでhigh gain側(HG)とlow gain側(LG)に分割され、このキャパシタの容量比でhigh gainとlow gainの増幅比が決まっており、high gainとlow gainで10:1の比で電荷を分割する。キャパシタンスの後にはPreAMPが配置されており、ここで信号が増幅される。PreAMPのgainは可変になっており、high gain、low gainそれぞれ設定が可能である。

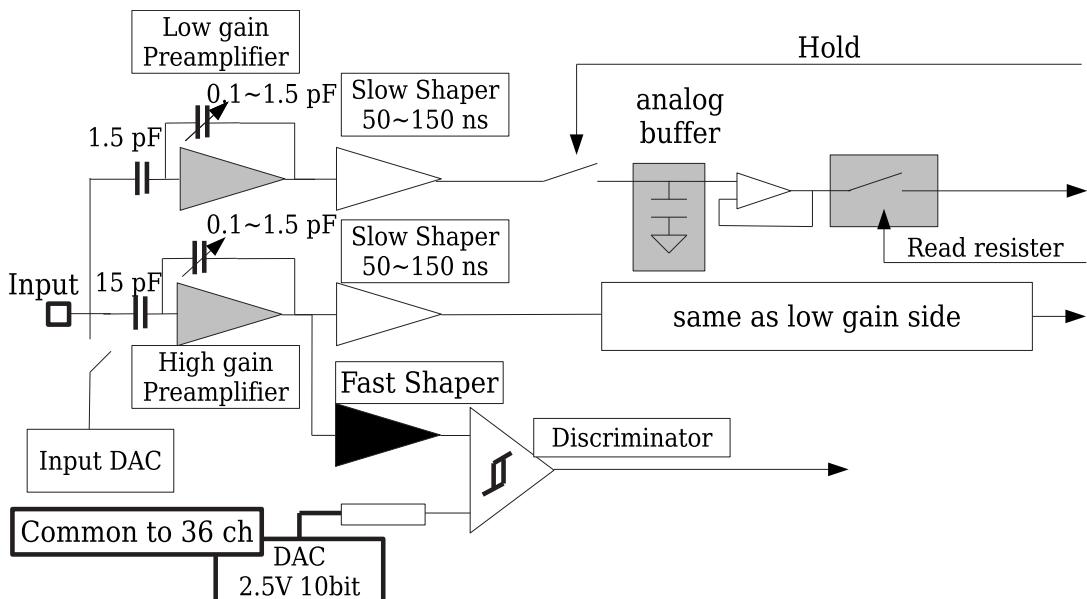


図 2.1: EASIROC 内部回路

る。EASIROC は成形器として電荷測定用の slow shaper と、時間測定用の fast shaper の 2 種類の成形器を内蔵している。slow shaper は high gain 側のみに配置されている。slow shaper で信号は peaking time (信号立ち上がりから最高点までに達する時間) 50 ns 程度で成形される。slow shaper の peaking time は可変であり、25 ns から 180 ns の間で設定可能である。fast shaper では peaking time 15 ns 程度で成形され、discriminator への入力となる。

EASIROC の電荷測定法は波高測定型であり、電荷積分型ではない事に注意しなければならない。よって一般的な QDC が持つ GATE という機能は存在しない。また、波高の最高点を自動検出しないため外から波高が最大となるタイミングを EASIROC へ知らせなければならない。slow shaper 後に配置されているキャパシタ (analog buffer)、HOLD 信号、及び read register 信号が電荷測定に使用される。HOLD 信号は low state でキャパシタに繋がっているスイッチを断線させる。そのため、キャパシタには HOLD 信号が high から low へ遷移した瞬間の電圧が保持されることになる。この電圧を ADC でデジタル情報に変換して電荷情報を知るのが EASIROC の基本的な使い方である。read register はキャパシタに保持された電荷を外へ出力する際に使用される。EASIROC はアナログ出力を high gain、low gain それぞれ 1 つずつしか有さないため、シリアルで電荷情報を出力する必要がある。バッファーアンプの後ろに配置されたスイッチは read register が high 状態の時導通となり、電圧を外へ出力する。そのため、ADC の動作クロックにあわせて read register を切り替えることで 32 チャンネル分の電圧を順番に A/D 変換することが出来る。HOLD 信号は全チャンネル共通の信号であるため、一度 HOLD を low にすると全てのチャンネルのキャパシタに電圧が保存されることに注意しなければならない。

チップ内部の discriminator の VTh は 10bit の DAC(以後 10bit DAC と表記する) が供給しており、全チャンネル共通である。discriminator の出力はアナログ信号と違いパラレルに出力される。

これらチップの特性を表 2.1 に列挙する。また、可変なパラメータを表 2.2 に示す。

表 2.1: EASIROC の特性

入力数	32
入力極性	正電圧
入力レンジ	160 fC ~ 320 pC

表 2.2: EASIROC の可変なパラメータ

InputDAC	0 - 4.5V (2.5V) 8 bit 精度
增幅率	10 ~ 150 倍 (HG), 1 ~ 15 倍 (LG)
slow shaper 時定数	25 ns ~ 175 ns (25 ns 毎)
Vth 用 DAC	0.26 - 2.5V 10 bit 精度

2.1.1 InputDAC

InputDAC は EASIROC の重要な機能の一つであり、MPPC のバイアスをチャンネル毎に調整する。InputDAC は 8bit 精度の DAC であり、デフォルト状態では 5V を 8bit で使いきるように設定されている。そのため、およそ 1 bit = 20 mV きざみでバイアスを調整できる。図 2.2 (a) に示すように、InputDAC は MPPC のアノード側に配置されている。そのため、図 2.2 (b) の様に InputDAC は正の電圧を出力することにより、バイアス電源と信号線間の電位差を調整して MPPC のバイアスを調整する。

InputDAC の出力電圧はその設定値が 0 の時に最大で、設定値をあげる毎に 1 bit = 20 mV の傾きに沿って下がっていく。そのため、InputDAC の設定値を 0 にすると MPPC へのバイアスは最低の状態となる。最大の出力電圧は参照電圧によって決まっており、EASIROC は内部と外部の両方の参照電圧をレジスタの設定によって使い分けることが出来る。内部参照電圧は 2.5 V となっており、外部参照電圧は VREF_DAC_5V に入力される電圧が使用される。本回路では VREF_DAC_5V には 4.5 V が接続されており、外部参照電圧を利用する際に必要な電源ピンである VDD_DAC_5V には 5 V の電源が接続されている。この参照電圧は各チャンネルで共通であり基本的に変動することは無いが、1 bit 辺りの傾きはチャンネル毎に変わりうる。図 2.3 はその模式図であり、3 本の線が描かれているがそれぞれ違うチャンネルの InputDAC の出力電圧を示している。チャンネル毎に傾きが異なるため、同じ設定値を与えても出力される電圧は異なるため、バイアス調整はチャンネル毎に MPPC のゲインを見ながら調整しなければならない。

前述のように EASIROC はデフォルトで 5 V を 8bit で使いきるように InputDAC の傾き (1bit 辺りの電圧変化) が調整されており、内部参照電圧を使用すると 8bit を使いきる前に

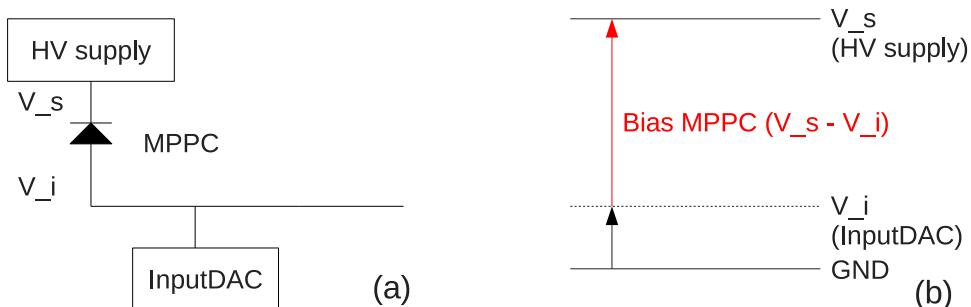


図 2.2: InputDAC によるバイアス調整機能。

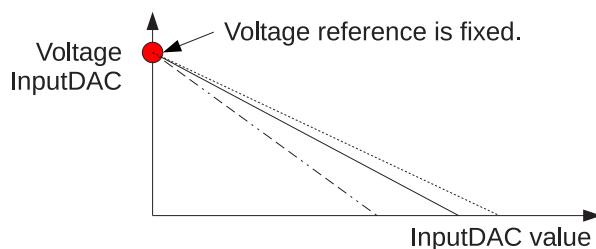


図 2.3: InputDAC の特性。

出力が 0 V になってしまう。もしも内部参照電圧のみを使用する場合は傾きを 2.5 V に合わせたほうが良く、その場合、IREF_DAC_5V を $M\Omega$ 程度の抵抗を介して GND に落とす必要がある。この際の推奨抵抗値は TBD である。また、外部参照電圧を使用しない場合は VREF_DAC_5V を N/C にし、VDD_DAC_5V を 3.3 V の電源に接続することが推奨される。

2.1.2 PreAMP

EASIROC には反転増幅の PreAMP が 2 個入っている。PreAMP の増幅率は可変であり、PreAMP の前段に配置されているキャパシタンスの容量比と合わせて、HG 側は 10 ~ 150 倍まで、LG 側は 1 ~ 15 倍まで増幅出来る。増幅率の変更はフィードバックキャパシタンスの容量を変えることで行い、キャパシタンスの容量と増幅率の対応を表 2.3 に示す。

また、PreAMP には波形が歪むのを防ぐための補償キャパシタンスが取り付けられている。補償キャパシタンスの容量も可変となっており、容量によって信号の波形も変化するが詳細な調査は行っていない。そのため本回路では開発者からの推奨値である、HG 側 500 fF、LG 側 3 pF を常用することにしている。

表 2.3: EASIROC の特性

容量	増幅率 (HG 側)	増幅率 (LG 側)
1.5 pF	10.0	1.00
1.4 pF	10.7	1.07
1.3 pF	11.5	1.15
1.2 pF	12.5	1.25
1.1 pF	13.6	1.36
1.0 pF	15.0	1.50
900 fF	16.0	1.60
800 fF	18.8	1.88
700 fF	21.4	2.14
600 fF	25.0	2.50
500 fF	30.0	3.00
400 fF	37.5	3.75
300 fF	50.0	5.00
200 fF	75.0	7.50
100 fF	150	15.0

2.1.3 slow shaper と fast shaper

shaper は波形を電荷と時間測定に適した形に成形するために利用される。信号反転するため shaper からの出力は正電圧となる。slow shaper は HG、LG 両方に配置されており、

電荷測定に使用される。fast shaper は HG 側のみに配置されており時間測定用に使用される。slow shaper は peaking time を 25 ns から 180 ns の間で変更可能であるが、fast shaper は peaking time を変更することは出来ない。shaper の peaking time を短くすると同じ電荷量でも信号波高が高くなり、長くすると低くなる。そのため、長い peaking time を利用すると基本的にベースラインノイズとの S/N は悪くなるが、後の電荷測定方で詳しく述べるように、HOLD 信号入力までの猶予が長くなる。利用者は目的の用途に合わせて peaking time を変える必要がある。EASIROC のデフォルト設定は 50 ns であり、仕様書内では特に断りのない限り peaking time 50 ns の測定結果を掲載している。

PreAMP、slow shaper、fast shaper によって成形増幅過程にある信号を図 2.4 に示す。図 2.4 (a) は PreAMP の出力を示しており、信号が反転している様子がわかる。PreAMP の信号は仕様上波形の一部分(ある電圧よりも高い部分)しか外へ出力できない出来ないようになっているため、ピーク付近しか我々は見ることが出来ない。図 2.4 (b) は slow shaper によって成形された信号を示している。peaking time は 50 ns に設定されており、信号の立ち上がりからおおよそ 50 ns で波高が最大となっていることが分かる。図 2.4 (c) は fast shaper の出力信号であり、図 2.4 (d) には slow と fast のタイミング差が示されている。fast shaper の信号の立ち上がりが slow shaper に比べ十分速いため、HOLD 信号に自分自身の discriminator 信号を使う self hold にも対応することが出来る。また、slow、fast 両方の信号にはアンダーシュートが見られる。アンダーシュートを含めた信号の幅が EASIROC が受けれる最大のレートを決めており、fast 側がおおよそ 4 MHz、slow 側がおおよそ 1 MHz が入力限界となる。

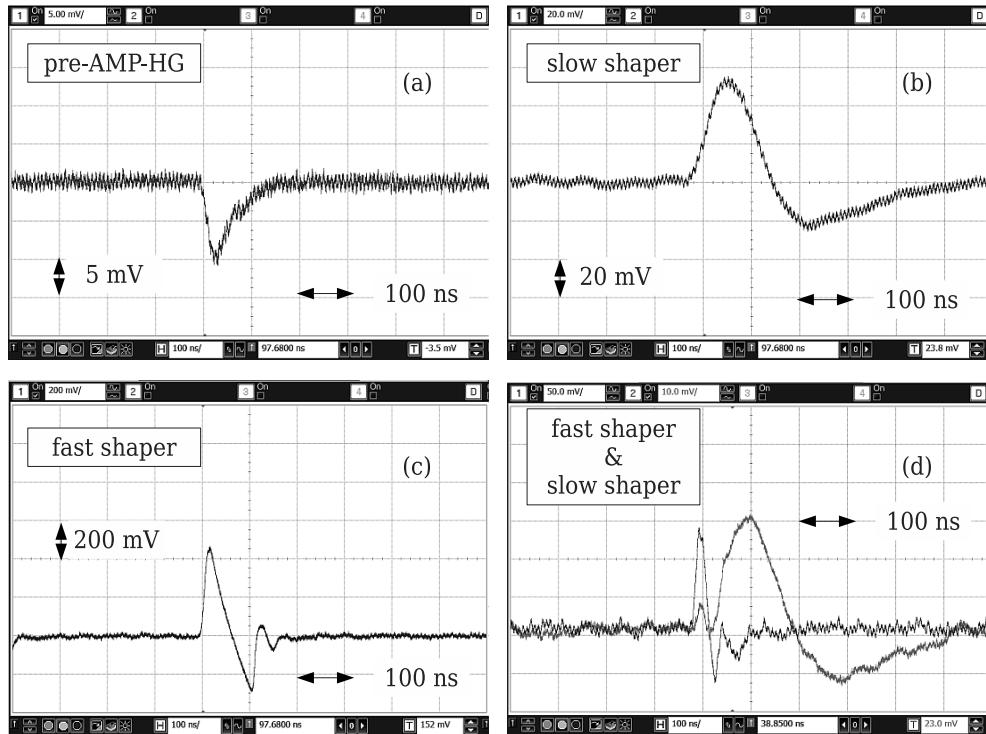


図 2.4: EASIROC のアナログ信号処理過程。

2.1.4 電荷測定方

前述の通り EASIROC の電荷測定方は波高測定型である。EASIROC はその機能として波形を外へ出力する事は出来るが、ADC 使用時には波形は読出さない事に気を付けなければならない。図 2.5 に slow shaper 後に配置されている回路系の模式図を示す。slow shaper 後には 1 つのキャパシタンスが配置されており、スイッチを断線させることでその時の電圧をキャパシタンスに保持することが出来る。スイッチを断線させるための信号が HOLD 信号であり、ロジック High 状態で導通、ロジック Low 状態で断線となるため、HOLD 信号が 1 から 0 へ遷移する瞬間の波高がキャパシタンスに保存されることとなる。HOLD は全チャンネル共通の信号であるため、HOLD が 0 になると 32 チャンネル全てのキャパシタンスに電荷が保存される。信号を HOLD する際の概念を図 2.5 の下段に示す。

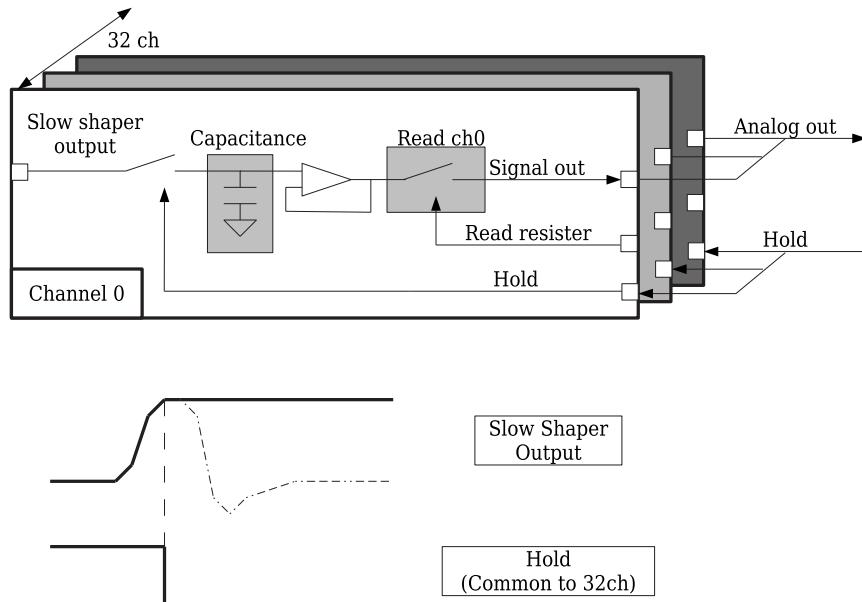


図 2.5: EASIROC の電荷測定方法。

概要で述べた通りアナログ出力線は HG,LG それぞれ 1 つしかないので、キャパシタンスに保存した電圧はシリアルに ADC へ転送しなければならない。(multiplexed analog output) その役目を負うのが read register であり、単純な shift register で順番にチャンネルを切り替えていく。ADC 測定時にはクロックに合わせて DC 電圧が各チャンネルから出力され、ADC はその電圧を A/D 変換し波高を知る。read register がチャンネルを切り替えてから DC 電圧が安定するまでには一定の時間がかかり、このスイッチの速度が EASIROC の読み出し速度を決めている。現在開発者の示す限界速度は 2.5 MHz である。(後述の Track and Hold の register を high にした場合) read register は波形を外へ出力する際にも使用でき、例えば、HOLD を 1 にして read register をチャンネル 10 に設定すれば、チャンネル 10 の波形が Analog output HG 及び Analog output LG から出力される。

ADC を使用している際の信号タイミングを図 2.6 に示す。図 2.6 は ADC に AD9220 を使用した場合のタイミング図である。Analog output とは前述の EASIROC からのアナログ信号出力であり、クロック信号に合わせて DC 電圧が切り替わる様子がわかる。clk slow

control は read register を送信するためのクロックであり、ASIC configuration 用の slow control クロックとは別物であるので注意しなければいけない。FIFO reset は FPGA 内部信号であり、ADC データをためるために FIFO のクリアを行う。read register in は read register を各チャンネルに送るための shift register への入力となる。read register out は EASIROC を daisy chain して使う場合の連結用の信号線で、入力した read register が押し出されてくる。hold は EASIROC への hold 入力であり、A/D 変換中はキャパシタの DC 電圧を保持しなければならないため、A/D 変換中 low state を出力するように幅を FPGA 内で変えている。FIFO enable は FIFO への書き込みを enable にする信号である。clk write FIFO は FIFO の書き込みクロック、clk ADC は ADC を駆動するためのクロックである。

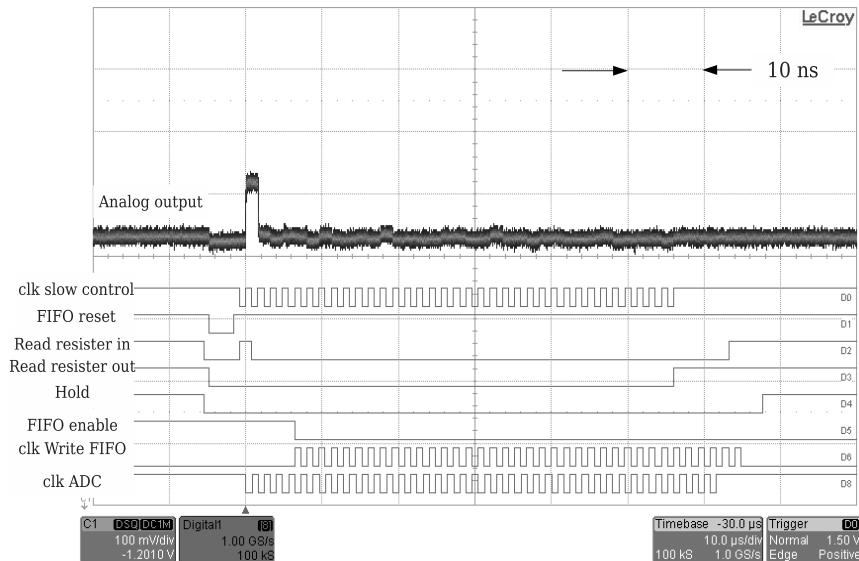


図 2.6: ADC を駆動する際の信号タイミング。

2.1.5 時間測定 (discriminator)

時間測定用のデジタル信号は図 2.4 (c) に示された fast shaper の出力を EASIROC 内部の discriminator に入力することで得られる。discriminator の信号は analog 側と異なり、1.2 V から 3.3 V まで任意の電圧 (VDD への供給電圧に依存する) でパラレルに出力される。discriminator への VTh は内部の 10bit DAC で供給する。10 bit DAC の出力電圧、fast shaper の base line 電圧、及び VTh の関係を図 2.7 に示す。VTh 用の DAC も InputDAC と同様に、基準電圧 (この場合 2.5 V) からある傾きをもって電圧が下がっていく。また、fast shaper の base line 電圧は GND ではなくある一定の値を持っている。よって、VTh は DAC の電圧とベースラインの電位差で表される。

10 bit DAC に関しては InputDAC と同様で、基準電圧はあまり動かないが傾きが変わりうる。InputDAC の場合は同一のチップ内でチャンネル毎に傾きが変わったが、10 bit DAC の場合異なったチップ間で傾きが変わりうる。そのため、あるチップで 1 p.e. に設定した VTh の値を、他のチップへそのまま適用することは出来ない。(たとえ MPPC のゲ

インを完全に同じにしても不可能である。) しかし後述のように基板上で電流制限抵抗を挿入することで 10 bit DAC の傾きは調整可能であり、チップ間の違いを吸収することは出来る。また、10 bit DAC の傾きは coarse と fine が DACslope.txt で設定でき、通常 fine で使用する。fine で利用中の場合、10 bit DAC の設定値が 850 付近でおよそ base line 電圧となる。coarse は fine よりも速く電圧が下がるため特に理由が無い限り使用することはない。

discriminator 信号はパラレル出力の他に digital_line という信号線からも出力される。digital_line は multiplexed digital output であり、アナログ信号と同様に read register を用いて出力するチャンネルを選択できる。本回路では digital_line は一度 FPGA を通った後、NIM レベルに変換されて出力される。

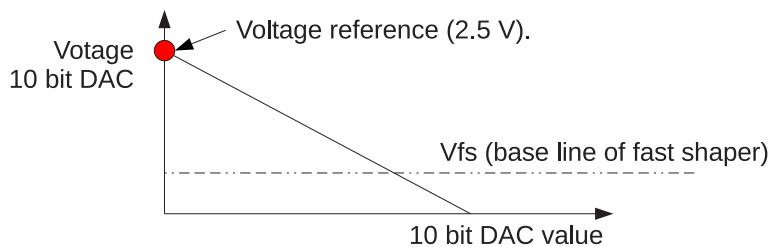


図 2.7: VTh の定義。

2.1.6 信号プローブ

EASIROC には analog output 用の Analog Out HG (LG) の他にもう 1 つアナログ信号を出力するための Probe という信号線が存在する。図 2.4 の 4 つの信号は全て Probe の信号である。Probe からは以下に示す成形増幅過程にある 5 つの種類の信号を出力することができる。

- OUT PA HG
 - HG 側の PreAMP 出力。PreAMP の信号は波形全体は出力されず、波形の高い部分のみ出力される。
- OUT PA LG
 - LG 側の PreAMP 出力。
- OUT SSh HG
 - HG 側の slow shaper 出力。この信号は Analog Out と違い電圧を HOLD で保存するためのキャパシタ以前の信号がプローブされるため、HOLD を入力しても OUT SSh の波形は変わらない。
- OUT SSh HG

- LG 側の slow shaper 出力。
- fast shaper
 - fast shaper からの波形を出力する。

Probe 信号を出力していると EASIROC 内部にノイズを乗せるため、データを取得する際にはリセットする事を推奨する。

2.1.7 テスト入力

EASIROC には MPPC 入力ラインをバイパスして直接 PreAMP に信号を入力するためのテスト入力ラインが用意されている。(pin 名で in_calib) テスト入力は HG と LG の信号強度比を決めている 15 pF と 1.5 pF のキャパシタンスをバイパスし、どちらも 3 pF のキャパシタンスでカップルするため、テスト入力に対しては HG も LG も同じように応答する事に気を付けなければならない。Function generator など任意波形を生成する機材を有していれば簡単に EASIROC へ信号を入力する事が出来る。

第3章 PCB

EASIROC の理解がまだ不十分であったこと、また回路設計が始めてでよく分かっていなかった事から本回路の構成はいい加減な部分が多い。参考にならない部分に関しては適宜注釈をいれていく。

3.1 主な機能 (要求仕様)

MPPC の制御は EASIROC が全て担うため、回路に求められる機能としては主に ADC、及び TDC である。本回路は LVDS で discriminator 信号を出力できるため、外部の TDC で時間を測ることができるが本来は FPGA 内部の MHTDC を使用することが望ましい。

- EASIROC チップを使用すること。
 - MPPC 32ch の同時駆動
 - 4.5 V 及び 2.5 V 8Bit 精度での MPPC の bias adjustment
 - double gain (high 側:10 - 150 倍、low 側:1 - 15 倍)
 - high low 側それぞれのアナログ及びデジタル出力
- NIM レベルでの I/O を有すこと (NIM モジュールに相当する速度を有する)
- ADC を有し電荷情報を得ることが出来ること。
- 32ch 分のディスクリ信号を LVDS で出力できること。
- FPGA 内に MHTDC を構築し、 LSB 精度 1 ns で時間を測れる事。
- SiTCP (SOY) による制御ができること

3.2 形状

テストボードなので準拠規格は無し。基板サイズは I/F によって制限される。

3.3 インターフェース

I/O に関してコネクタの製品番号と特性を示す。各 I/O の詳細な使い方に関してはユーザーガイド参照のこと。

- Analog 入力
 - MPPC 入力 : HIF-3BB-64PA-2.54DS (HIROSE) 雄 64pin 終端抵抗 50Ω
 - HV 入力 : LEMO (EPL.00.250.NTN) 電圧範囲 0 to +100V
 - テスト入力 : LEMO (EPL.00.250.NTN) 電圧範囲 0 to +0.5V
- Digital 入力
 - NIM input x5 : LEMO (EPL.00.250.NTN) 速度 400 Mbps
 - JTAG (878311420)
 - SOY I/F (FX2-100S-1.27DSL)
- Analog 出力
 - Analog Out x2 : LEMO (EPL.00.250.NTN) 電圧範囲 -5V to 5V 帯域 110 MHz
 - Probe x1 : 同上
- Digital 出力
 - NIM output x4 : LEMO (EPL.00.250.NTN) 速度 1 Gbps
 - ディスクリ出力 : HIF-3BA-34PA-2.54DSA (HIROSE) x2 速度 630 Mbps

3.4 電源

今後クレート挿しの基板を作る際に移行が楽になるように電源電圧は ± 6 V が選択されている。本回路が要求する最大の電圧は +5 V である。本回路では高速信号を扱わないため FPGA 電源は LDO である必要はなかったが、自分の知識不足のため無駄な LDO が多い。

- 入力電圧
 - +6V
 - -6V
- レギュレータ
 - +5V0A : LP38690SD-ADJ/NOPB (ASIC)
 - +3V3A : LP38690SD-ADJ/NOPB (ASIC)
 - +5V0D : LP38690SD-ADJ/NOPB (汎用)

- +3V3D : LMZ12003TZ-ADJ/NOPB (FPGA)
- +2V5D : LP38690SD-ADJ/NOPB (FPGA)
- +1V8D : LP38500TS-ADJ/NOPB (PROM)
- +1V2D : LP38500TS-ADJ/NOPB (FPGA)
- -3V3D : LM2991SX/NOPB (NIM 及びアナログバッファ)
- +4V5A : REF194ESZ (基準電圧 : 抵抗分割で+1V8も作成する)

- レギュレータ電流容量

- LP38690 : 電流出力 1A
- LP38500 : 電流出力 1.5A
- LMZ12003 : 電流出力 3A
- LM2991 : 電流出力 1A

* LP3855 は ADJ で 2.0V 以下の電圧を作ると電源投入時にオーバーシュートするらしいので、対策品の LP38500 に置き換え。

* REF194 の出力は汎用 OP アンプ (AD8602ARMZ-REEL) でバッファされる。

3.5 主要部品

このテストボードは一つの ASIC (EASIROC) と 1 つのコントロール用 FPGA、2 つの ADC で構成される。FPGA は EASIROC の制御と SOY との通信を担う。またトリガーロジックも FPGA 内部で組むことが出来る。EASIROC からの high gain (HG) 出力と low gain (LG) 出力は 2 つの ADC で同時に A/D 変換される。

- EASIROC (Extended Analogue SiPM Integrated Read Out Chip)
 - 32 チャンネルの MPPC の駆動
 - チャンネル毎のバイアス調整機能 (4.5 V 及び 2.5 V, 8 bit 精度)
 - 10 倍ゲインの違う 2 系統の読み出し (high gain, low gain)
 - ダイナミックレンジ 160 fC から 320 pC
 - Amp, Shaper, discriminator
 - 可変な gain, 成形時定数, discriminator 電圧閾値
 - アナログ情報のシリアルな出力
 - パラレルな discriminator 出力
 - slow control に制御
- SPARTAN6 (LX25 FGG484, speed grade -2)
- AD9220 (パイプライン ADC)

- 12 bit 精度
- 最大 10 MHz サンプリング (ただし EASIROC によって最高 2.5 MHz に制限される)
- SN65LVDS389BDT (LVCMOS to LVDS 変換)

3.6 ブロック図と完成したボード

ブロック図を図 3.1 に示す。入力された MPPC の信号は EASIROC で成形増幅され、アナログ信号とディスクリ信号にわけられる。EASIROC には MPPCへのバイアス供給機能は存在しないので、バイアスの印加は別ラインから行う事に気を付けなければならない。基板上の HV コネクタに電源をつなげば信号入力用のペア線の逆側を使って MPPC へバイアスを送ることが出来る。EASIROC からは HG、LG 両方の multiplexed analog output と digital_line (multiplexed digital output)、パラレルなディスクリ信号、及び 32 チャンネルの OR 信号が outputされる。これら digital_line、ディスクリ信号、OR 信号は前述の VHI で信号電圧を変えることが出来るため、本回路では試験の意味合いをかねて VHI をジャンパーで選択できるよう設計した。ジャンパーは 1.8V か 3.3V で選択でき、EASIROC のデジタル部分と FPGA のバンク 3 の電圧を変更できる。FPGA のバンク 3 は EASIROC の VHI と同じバンク電圧で駆動されるため、EASIROC のデジタル信号をバッファして 3.3V のバンクからもう一度出力するようになっている。ただし、ここでいうデジタル信号とは時間測定用の信号線であり、slow control 用の信号線などは VHI にかかわらず常に 3.3V であることに注意しないといけない。アナログ出力は 2 台の ADC (AD9220) で HG、LG 同時に A/D 変換される。また、ADC へ繋がってる信号線をそのまま基板外へ引き出す事でアナログ信号のモニターとして利用できる。EASIROC の制御は基板上の FPGA で行い、本回路では SiTCP の機能を SOY に委託しているため基板上の FPGA は EASIROC と SiTCP とのブリッジの役割を果たす。パラレルの discriminator 信号線は一度 FPGA から出力された後、SN65LVDS389 で LVDS へ変換されて外へ出力される。一度 FPGA でバッファする理由は前述のように VHI によって EASIROC からの出力電圧が変わるために、一度 FPGA でバッファすることによって出力は常に 3.3V となる。その他、いくつかのロジック I/O が FPGA へ接続されている。また、完成した PCB の写真を図 3.2 に示す。

3.7 動作

より詳しい実際の使用方法に関してはユーザーガイド参照の事。本回路の動作には大きく分けて 2 つのモードが存在する。1 つは DAQ を使わない単純なアンプディスクリとして動作するモードであり、もう 1 つは DAQ モードに入り ADC や TDC のデータを取得するモードである。

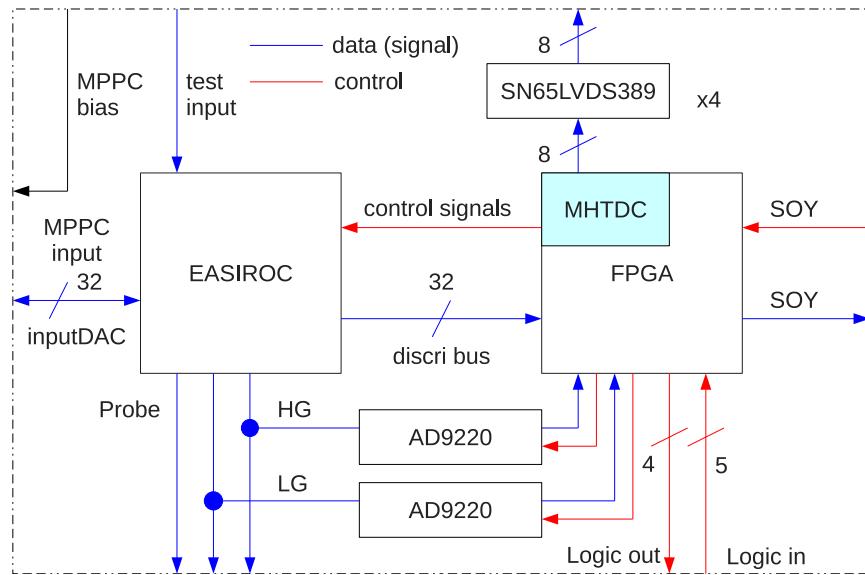


図 3.1: PCB ブロック図

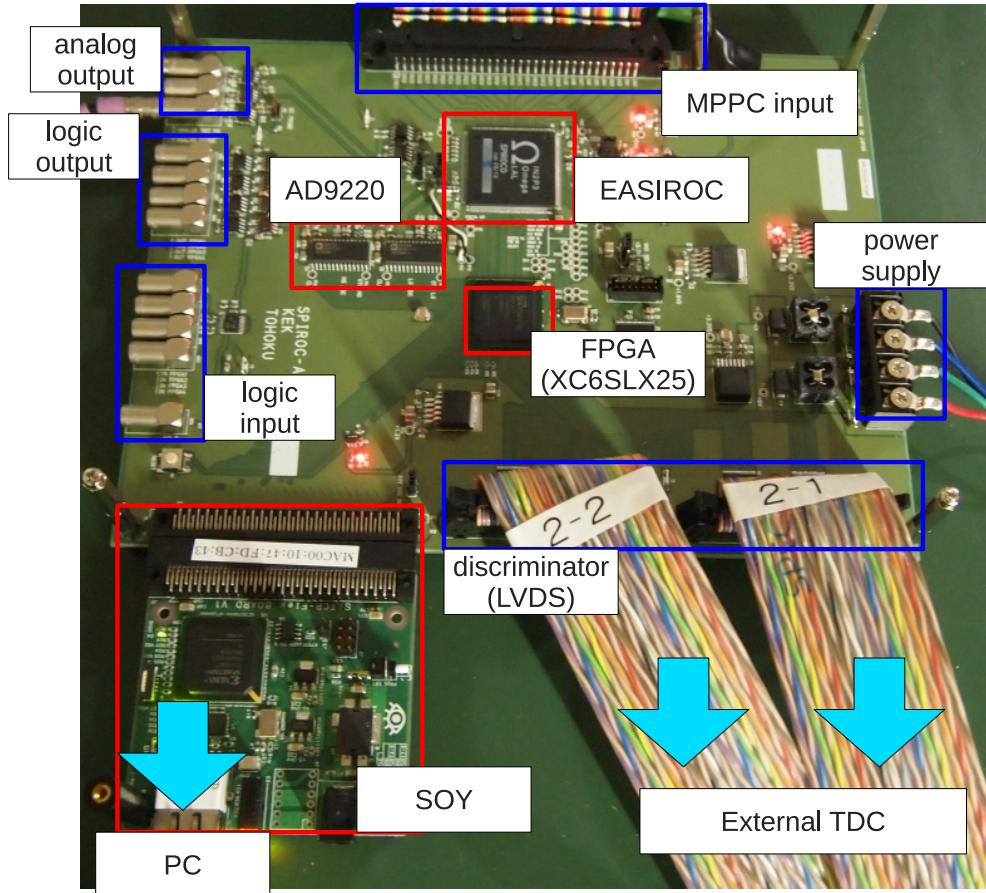


図 3.2: PCB 写真

3.7.1 信号モニター モード

DAQ を動作せず MPPC を駆動するのみのモードであり、DAQ の IDLE 状態とも言える。このモードで EASIROC のパラメータを変更しその結果を AnalogOut や Probe 出力などを

使って確認する。DAQ が動いていない状態ではいつでも Slow control を行う事が出来る。出力可能なアナログ信号は Analog out HG、LG 及び Probe である。デジタル信号に関しては NIM OUT1 から 32 チャンネル分の discriminator の OR が出力され、NIM OUT2 からは ReadRegister で選択した discriminator の信号が出力される。ただし、Analog out に関しては ADC 駆動中は ADC へ転送される DC 電圧がそのまま外へ出力されるので MPPC の波形のモニターとしては使えない。Probe は Hold される前の信号を外へ出力するため ADC の駆動と関係なくいつでもモニターすることが出来る。また、discriminator は常時駆動しているためこのモードが事実上の ASD モードとなる。TCP 接続を切断すると FPGA の内部は初期化されるが EASIROC のレジスタは初期化されないため、一度設定を固定してしまえば SiTCP との接続を切ってしても問題ない。

3.7.2 DAQ モード

DAQ を使用して ADC や TDC のデータを TCP 通信を介して収集するモードである。DAQ モードに HOLD や common stop 信号で A/D 変換を行う conversion phase とデータをネットワークへ送信する transmit phase の 2 つの phase が存在する。本回路は DAQ モードになると HOLD と common stop の入力を待ち、それぞれの信号の入力があった時点で A/D 変換を開始する。ADC と MHTDC は独立の機能であるためそれぞれ正しいタイミングで HOLD と common stop を入力しなければならない。ADC、MHTDC 片方でも conversion phase へ入ると本回路は BUSY を出力し transmit phase 終了まで出力し続ける。conversion phase ではデータを ADC、MHTDC それぞれの FIFO へ書き込むところまでしか処理を行わない。データを転送するための transmit phase へ移行するためには TRIGGER の入力が必要となる。本回路は TRIGGER の入力を記憶しているため、TRIGGER 入力のタイミングに関しては特に制限がないが HOLD や common stop と同時にそれよりも遅いタイミングでの入力を推奨する。実際の transmit phase への移行は、ADC 及び MHTDC の conversion phase が終了していること（内部レジスタを参照しているためユーザーからは分からない）、TRIGGER が入力されている事、の両方が満たされているが条件となる。全データを SOY の FIFO へ書き込み終わると transmit phase を抜け再び HOLD と common stop 入力待ち状態となり、BUSY の出力を解除する。

DAQ モード終了時にはプログラム側から FPGA の DAQ シーケンスを止めるための信号を送信するが、プログラムが終了信号を送信した時点から後にネットワークから送られてきたデータは dummy data として捨てられる。（ファイルにも書き込まれない）ネットワークによって FPGA を制御しているため CPU のタイミングと実際に FPGA の処理が止まるまでのタイミングに大きな差が生じるためこのような処理を行っている。また、SOY には内部 FIFO をクリアする機能がないため、次の RUN へデータを残さないために RUN がクローズした時点で SOY からデータを吸い出しきる必要がある。その動作も dummy data 処理は兼ねている。全データを吸い出し TCP の recv 関数からタイムアウトが返ってきた時点でプログラムはファイルをクローズして DAQ の IDLE 状態へと戻る。

3.8 レイアウト時に気を付けたこと

アナログ入力のカップリング防止のためにMPPC入力は偶奇で使用レイヤーを分けた。EASIROC開発者推奨は内層3層に分けることだったが表層と内層の2層でも十分な性能を得られた。

3.9 基板レイアウト図と修正記録

レイアウト修正1.pdf及び、レイアウト修正2.pdfを参照。

主な基本的なチェック項目は、

- 50Ω 線と作動線のインピーダンスに気を付ける。
- パスコンの位置やスルーホールの数を確認する。
- 電源層の構造や配置を最適化する。
- 信号の間隔や受動素子の位置を確認する。
- パターン走行をなるべく真直にする。
- アナログとデジタルが混在する場合、なるべく隣り合わないようにする。アナログパターンの裏などにも気を付ける。

第4章 FPGA

当初は MHTDC の全機能を実装する予定は無く、単に EASIROC と ADC の動作テストと MHTDC の実装テストを行う予定だった。そのため本回路で採用されている FPGA は非常に小さく MHTDC の機能を実装するとスライスの占有率が 70 % を越えてしまう。そのため ADC の zero supress や MHTDC の time window の設定などの補助機能を組み込む事が出来なかった。ある程度実戦投入を見据えた開発をしないと後で困るという教訓となる。

4.1 使用 FPGA

本回路上の FPGA は Xilinx SPARTAN6 ファミリ XC6SLX25-2FGG484C である。選定理由は以下のような理由である。

- Altera の Cyclone (EP1C6Q240C6) よりもロジックセル数が多いこと。
- 300 以上の pin を持ち尚且つ出来るだけ安価であること。
- 1 ns の TDC を作ることが出来る程度に速いクロックが使えること。

回路容量の異なる SPARTAN6 ファミリを実装することは考慮されていない。この FPGA とピンコンパチな SPARTAN6 は XC6SLX150 まで存在せず、尚且つ XC6SLS150 を実装してしまうと PROM 容量の関係で MCS ファイルをダウンロードすることが出来ない。

FPGA 固有の機能 (ギガビットトランシーバなど) の機能を使用することはない。

MHTDC を実装するために 250 MHz のクロックで DFF がドライブ出来ることが条件である。具体的には 256 個 ($4 \times 32 \times 2$) の DFF をドライブできなければならない。

MHTDC 用の ring buffer の実装とデータ FIFO のために最低でも 70 個程度の 9Kb block RAM が必要である。またスライスの占有率を決めているのは殆どが MHTDC であり、MHTDC を実装しない場合スライスの占有率は 5 % 程度である。全機能を実装した場合スライスの占有率は 75 % となる。

4.2 ブロック図

FPGA 全体のブロック図を 4.1 に示す。このブロック図は FPGA コードの TopLevel 構造を反映しており、TopLevel に配置されている全てのモジュールと主要な信号線が記載されている。TopLevel の構成は 4 つに大きく分けることができる。EASIROC を制御するパート、EASIROC や ADC からデータを取得する DAQ パート、FPAG の機能全体を制御する

パート、外部と通信を行うパートの4つとなる。FPGAコードは全てVHDLで作成されている。

各パートの説明の前に一つこの回路の設計ミスの点を挙げなければならない。SiTCPにはTCPとUDPの両方の通信方法が用意されているが、TCPは基本的にデータ送信用の一方通行が推奨されており、コンフィグやハンドシェイク動作などはUDP通信を使って行う事が推奨されている。しかし、この回路を設計した頃SOYを使ったUDP通信の方法が全く分からず(今でもよく分かっていない)、結局TCP通信を使って全ての制御を行う実装方法をとってしまった。そのため、TCP関連のモジュールの動作やDAQモードとコンフィグモードの切り替え部分などに若干複雑な事をさせてしまっている。その点が最終的にモジュールの不安定さに繋がってしまい、開発期間を伸ばす結果となってしまった。そのためTCP通信部分は全く参考にならないので注意が必要である。各パートの概要を以下に述べる。

4.2.1 EASIROC制御部

EASIROCのコンフィグ全般を受け持つ部分で、以下のモジュールで構成される。

- SlowControl_Spiroc2_ver2

この部分ではユーザーが送信したビット配列を slow control 様に deserialize して EASIROC へ送信する。slow control にはチップの動作を決める slow control とどのチャンネルをアナログ出力するかを決める Read slow control の2種類があり、両方をこのモジュールから送信する。EASIROC はレジスタをロードするまではただの shift register 的な振舞しかしないため複数の EASIROC を実装した場合でも本モジュールの Instance は一つで良い。Read slow control は ADC を使用する際にも使うため、ADC モジュールからも同様に Read slow control の制御線が出ており TopLevel 上でどちらを EASIROC へつなぐか選択されるようになっている。また、EASIROC にはいくつか slow control を利用しないで直接チップの状態を制御するピンが存在し、そのピンのドライブも本モジュールが行う。ユーザーが送信したビットを DFF へラッチし、直接 EASIROC を制御する。モジュール名が SPIROC2 となっているのはこの部分は EASIROC の元となっている SPIROC チップと互換であるためである。

4.2.2 DAQ部

A/D変換したデータを収集するパートであり、以下のモジュールで構成される。

- ADC_AD9220dual_ver3
- MHTDC

本回路のDAQは独立したADCとMHTDCを内蔵しており、それぞれ、ADC_AD9220dual_ver3とMHTDCモジュールが制御及びデータ収集を行う。ADC_AD9220dual_ver3はAD9220ヘクロックを送信しA/D変換されたデータをFIFOへ格納する。DAQサイクル中はASIC

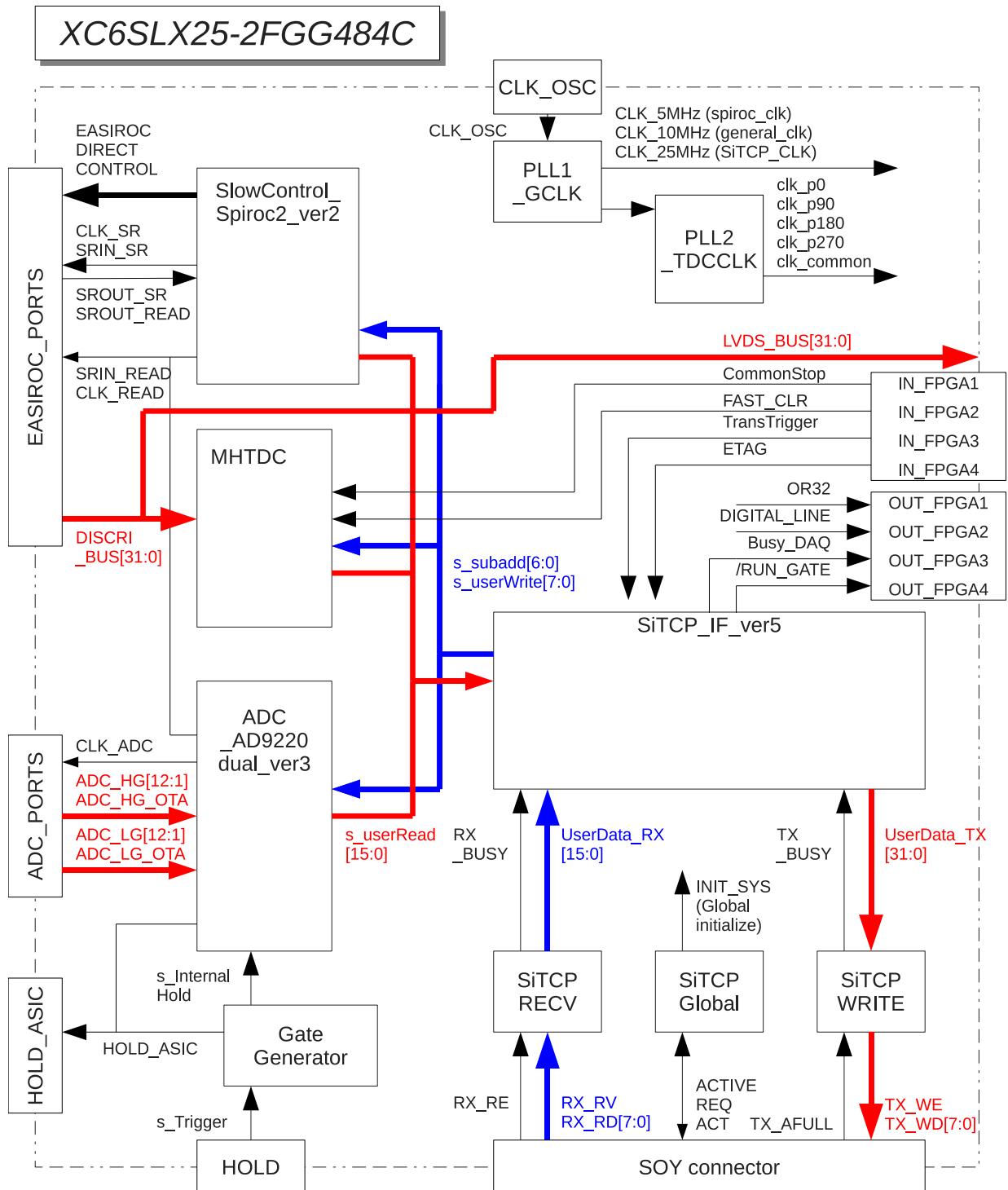


図 4.1: FPGA TOP LEVEL のブロック図

への hold 信号と Read SC 信号の線を本モジュールが駆動する。MHTDC は EASIROC から出力される各チャンネルの discriminator 信号を受け、1 bit = 1 ns の精度で時間を測定するモジュールである。測定できる時間窓は 1 us で、信号の立ち上がりと立ち下がりの両方を測定することが出来る。MHTDC 最終的に変換したデジタル情報をモジュール内の FIFO

へと格納する。これらのモジュールに対する信号として、ADCにはhold信号が、MHTDCにはcommon stop信号がそれぞれタイミング信号として必要となる。また、MHTDCはfast clearに対応しておりシーケンサステートを初期状態へ強制的に戻すことが出来る。

4.2.3 インターフェース部

FPGA全体を制御するパートであり、各モジュールへのアクセスと制御を行う。以下のモジュールで構成される。

- SiTCP_IF_ver5

本回路の制御を全て行うモジュールであり、SiTCPとの会話、各モジュールへのアドレッシングと読み書き、DAQの制御を行う。コンフィグモード時はSiTCPからのデータ待ち状態を維持し、各モジュールへの書き込み、読み出しを行う。書き込みが一回8bitずつ、読み出しは一回16bitずつとなっており、ハンドシェイクのため速度はでないが確実にモジュールの試験を行うことが出来る。この操作を前述の通りTCP回線を利用して行っているため、シーケンサの分岐が若干複雑になっている。

DAQモード時には本モジュールのデータ転送系をハンドシェイクモードから一方的にSiTCPへデータを送信するモードに切り替えて運用する。DAQ用のシーケンサを実装しており、DAQ部へのアクセスして状態の監視とデータの吸い出し、及びSiTCPへの送信を行う。また、1イベントのデータサイズの確認とイベント数のインクリメントを行い、ヘッダーを付与する。ヘッダーにデータサイズが書き込まれているため、ユーザーは読み出すデータ長を先に知ることが出来る。本回路のDAQはイベント毎読み出しとなっておりmulti-event bufferには対応していない。

4.2.4 SiTCP部

SOYとの会話をを行い、データの送受信を行う。通信は全てTCP回線を使って行われる。以下のモジュールから構成される。

- SiTCP_RECV
- SiTCP_WRITE
- SiTCP_Global

SiTCP部の主な役割はユーザー回路側からSiTCPの信号線を隠すことにある。本回路では32bitを1ワードとしているため、SiTCPからの読み出しあればSiTCPの読み出しを4回、書き込みであっても同じく4回繰りかえす必要がある。この操作をユーザーから隠し、ユーザーは32bit長のデータの読み書きとBUSYの監視のみを行えばよく、AFULLの管理やRX_RVなどの監視を行う必要がない。SiTCP_RECVはSiTCPからの読み出しを行い、32bitのデータが利用可能となった時点でBUSYを解除しユーザー回路へ知らせる。SiTCP_WRITEはSiTCPへの書き込みを行い、ユーザーが32bitのデータの書き込み要求を出した時点から

SOY へ送信を完了するまでの間 BUSY を出力する。TCP_AFULL が出力されている場合は BUSY が解除されない仕様となっている。SiTCP_Global は TCP_CLOSE_REQ と ACT の管理を行っており、TCP が Active になったときと切断されたときに、それぞれ 1 回初期化信号を全モジュールへ配布する。

4.3 各部動作説明

4.3.1 SlowControl_SpiRon2_ver2

本モジュールには 8bit 長の DFF が 2 つと、8bit 長の FIFO が 1 つ実装されており、それ、word0,1,3 の出力及び SlowControl 用のレジスタのバッファに使用されている。word0,1,3 の更新はアドレスを指定し DFF の内容を書き換えることによって行われる。SlowControl のサイクルは、まずユーザーが FIFO へ必要なレジスタ情報を全て記録する。この時のアドレスによって現在 FIFO へ書き込んでいるレジスタが SlowControl 用なのか ReadSlowControl 用なのかを記録するようになっている。この記録はシーケンサの EndCycle が発行されるとクリアされる。FIFO への書き込みが終了すると word1(1) を 1 にしてシーケンサを st0 から st1 へ遷移させて EASIROC への転送を開始する。シーケンサステートを図 4.2 に示す。FIFO の bit 長が 8 であるため 8 回毎に FIFO からデータをロードして deserialize してクロックに同期させて EASIROC へ送信する。この方式は SlowControl と ReadSlowControl 共に共通である。FIFO が empty になると EndCycle を発行しサイクルを終了する。その後 LoadSC にビットを立てて (word1(0) に 1) EASIROC へ送信したレジスタを反映させる。

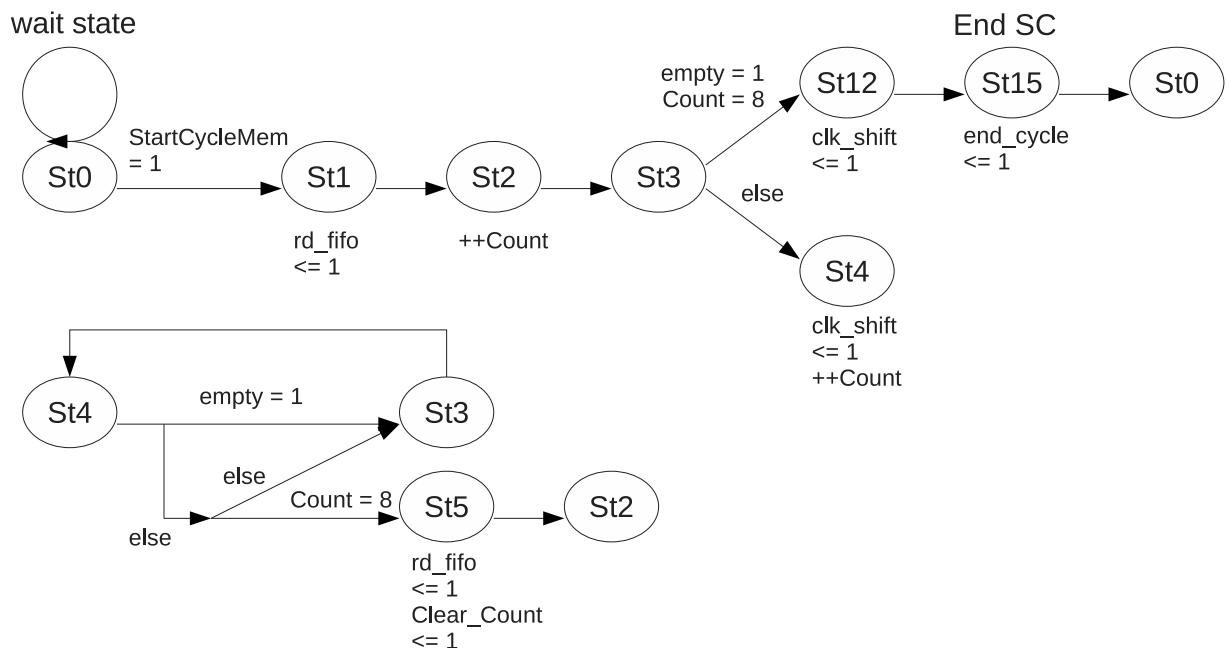


図 4.2: SlowControl のシーケンサステート図

4.3.2 ADC_AD9220dual_ver3

本モジュールは EASIROC に ReadSC の信号を送信し ADC への電荷入力を制御し、同時に AD9220 を駆動して A/D 変換を行い自身の FIFO へ記録する。本回路には AD9220 が 2 台実装されているため 2 台同時に駆動する。本モジュールには IDLE(もしくは Data ready)、Trigger 待ち受け、A/D 変換中の 3 つの状態が存在し、これらを順番に繰り返すことでデータを取得する。DAQ が走っていない時は本モジュールは IDLE 状態で待機しており、DAQ モードへ入ると Trigger 待ち受け状態へと移行する。この時に hold が入力されるとその信号を非同期ラッチして EASIROC のアナログ出力の状態を固定し、シーケンサの st0 を st1 へ遷移させ A/D 変換フェイズに入る。本モジュールの BUSY の定義は hold 入力からシーケンサが st0 へ戻るまでの間である。A/D 変換が終了し EndCycle が発行されたときに FIFO 以外の全ての状態がリセットされ Data ready 状態に移行する (IDLE と同義)。User_IF がデータを吸い出し終わると、再び Trigger 待ち受け状態へと戻る。

ADC 読出しのシーケンサステート図を図 4.3 に示す。st1,2 で EASIROC の ReadSC の初期化を行った後、AD9220 と EASIROC にクロックを送り A/D 変換を行う。clk_read と clk_adc の位相は 180 度ずれており、各ステートで 0 を代入していることに注意が必要である。st4,5 で srin_read を入力するのは EASIROC の ch0 から順番にアナログ出力を切り替えていくためである。AD9220 のレイテンシ分だけクロックを進めてから FIFO への書き込みを開始し、最後に EndCycle を発行してシーケンサを抜ける。st16 で s_EndConvert が発行されると Uesr_IF へ A/D 変換終了が伝えられる。Count が 37 になるまでループ処理を抜けないのは SPIROC2 の頃のなごりであり、dead time を無意味に長くしているだけなので修正が必要である。

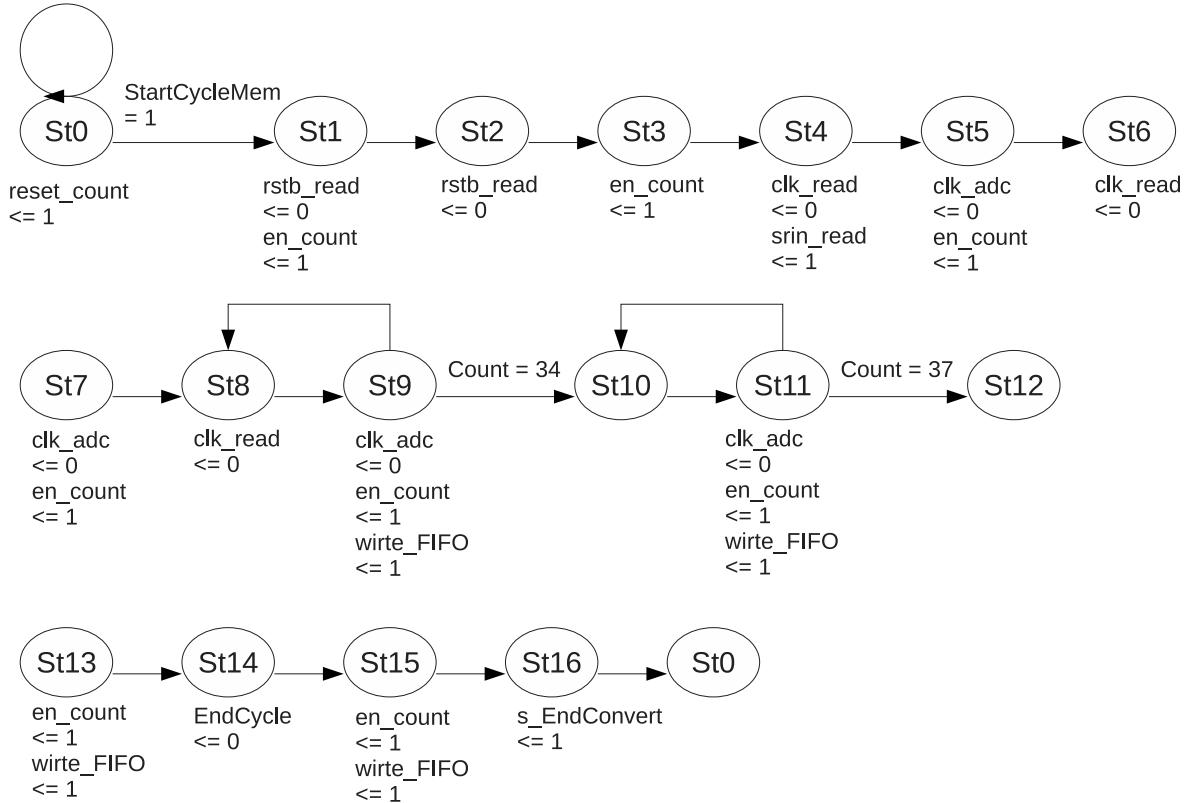


図 4.3: ADC 読出しのシーケンサステート図

4.3.3 MHTDC

MHTDC は主に discriminator の信号を 1 bit = 1 ns でデジタル情報に変換するための TDC Comparator、1 us 分の長さを持つ RingBuffer、各チャンネルの Hit 情報を格納するための L1 Buffer、全チャンネル分の Hit 情報を格納するための EventBuffer の 4 つで構成される。1 GHz のクロックで discriminator の信号をサンプリングする事は不可能であるため、250 MHz で 90 度毎に位相の違う 4 相のクロックでサンプリングし擬似的に 1 GHz を達成する。本回路の MHTDC は Leading と Trailing 両方の情報を取得することが出来る。

TDC comparator の概念図を図 4.4 に示す。4 相の 250 MHz クロックで 4 つの free run counter を走らせておく。(各カウンタは PLL lock 時に 0 度から順番に初期化されるよう設計されている) 入力された discriminator 信号は 4 相のクロックでサンプリングされ、DFF にカウンタの値をラッチするための enable 信号を生成する。それらの値は比較回路に入力されどのクロックが一番早く信号をサンプリングしたかを判定する。この時 0 度のクロックでサンプリングされた enable 信号を引き延ばし、遅いクロックに同期させることで Hit があった事を示す Hit ビットを生成し比較回路からの出力と結合して RingBuffer へ書き込む。250 MHz から遅いクロック (66 MHz) への受渡しで不定性が生まれるため、マージンをとるために 30 ns 程度不感時間が存在する。

また、この構造は Leading、Trailing 共通であるが Trailing の Hit ビットは Leading 側の Hit ビットがそのまま利用される。言い替えると Leading と Trailing のデータサイズは常に同じになるように設計されている。

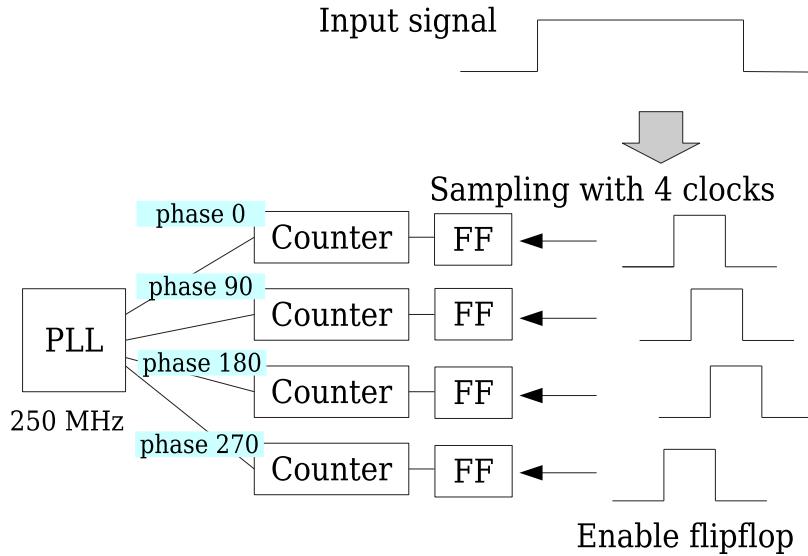


図 4.4: TDC comparator 部の概念図

TDC comparator から EventBuffer までのプロック図を 4.5 に示す。RingBuffer は 64 の深さを持つ BlockRAM であり、66 MHz のクロックで駆動されているため 1024 ns 分の長さを持つバッファとなっている。RingBuffer は常に TDC comparator の出力線の値を記録しつづけており、Hit があったかどうかは同じ配列内に埋め込まれている前述の Hit ビットを参照して判定する。L1 Buffer は 16 の深さを持つバッファであり、RingBuffer から Hit の有った際の値のみを取り出して格納するためのバッファである。16 を越えて Hit が有った場合新しい Hit は読み出されなくなる。EventBuffer は全てのチャンネルの L1 Buffer の情報を格納するためのバッファであり 256 の深さを持つ。最終的に EventBuffer への書き込みが終了すると MHTDC は Data ready を発行する。この構造は Leading と Trailing 両方が備えており、最大データサイズは 512 ワードとなる。

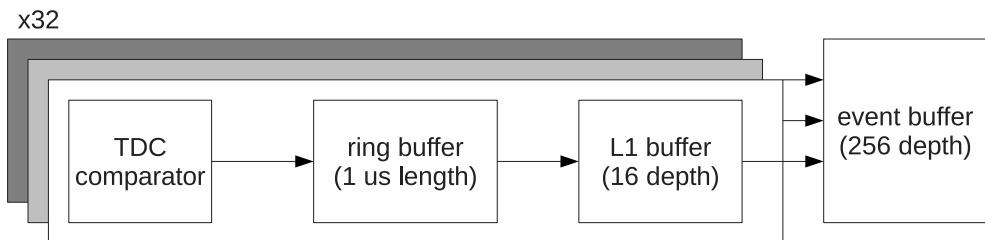


図 4.5: MHTDC のバッファ構造のプロック図

MHTDC のシーケンサステートを図 4.6 に示す。初期状態では st0 で待ちつづけている。StartTDC の信号により強制的に st1 へ遷移する。st1 への遷移はどのステートからでも StartTDC 信号によって即座に行うことが出来る。これは StartTDC が clear によって発行される事もあるためである。st2,3,4 でバッファと common stop manager 及び各 TDC instance の初期化を行い、common stop 入力があるまで st5 で待機する。common stop manager に

common stop 時のカウンタの値が保持されると RingBuffer から L1 への吸い出しを行う。この作業は各チャンネル並列で行える代わりに固定長の時間がかかる。RingBuffer を読み書きで同じクロックを使ってアクセスしているためこの処理には 1 us かかる。L1 への吸い出しが終わると ch0 から順番にアクセスしていき EventBuffer への書き込みを行う。この際に common stop との引き算を行う。EventBuffer への書き込みには SubSequencer を呼んで行わせる。32ch 分全てが終わると st11 へ遷移して Data ready 信号を発行して次の StartTDC に対して待機状態となる。

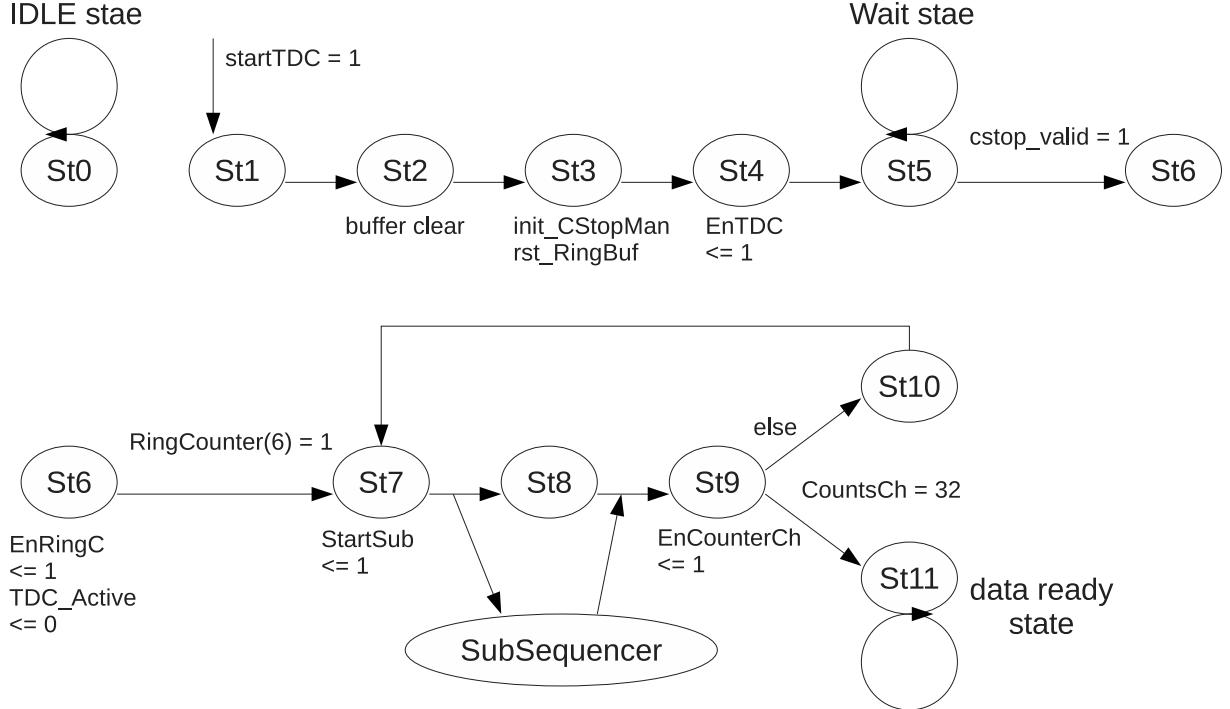


図 4.6: MHTDC のシーケンサステート図

4.3.4 User_IF

User_IF は各モジュールへのハンドシェイクでの読み書きと DAQ 制御の両方を担当するが、ここでは DAQ サイクルのみに関して述べる。DAQ サイクルのシーケンサステートを図 4.7 に示す。DAQ シーケンサは DAQGate が 1 の間サイクルを続ける。st1 で発行される StartDAQ は ADC と MHTDC のモジュールに配られ、それぞれを Trigger 待ち受け状態と common stop 待ち受け状態へ遷移させる。st3 は wait state であり、ADC 及び MHTDC の conversion が終了したこと、データ転送トリガーが入力されている事、SiTCP が Busy で無いことがデータ転送を開始する条件となる。まずデータ転送は Header 部分からはじまり、96bit 長の固定長の Header を送信する。Header 内部には EASIROC を示す Magic word、データサイズ、イベント番号が格納されている。次に TDC、ADC の順番に読出して転送を行うが、DAQ モードで使用しない(EnTDC、もしくは EnADC が 0) ように選択されて

いる場合はこの部分をスキップする。最後に DAQGate の状態を確認して、0 で無ければ再び StartDAQ を発行する。

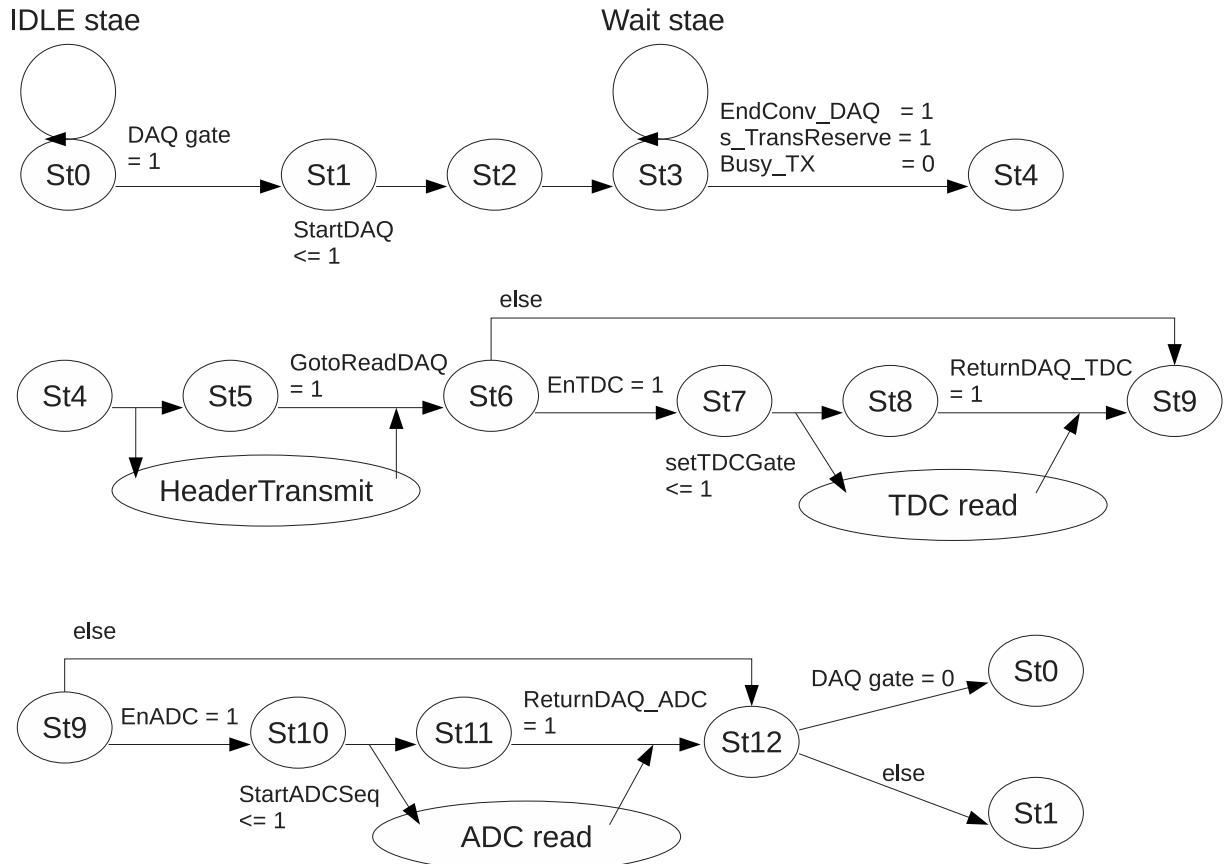


図 4.7: DAQ のシーケンサステート図

4.3.5 SiTCP_RECV

SiTCP から 1 ワード (32bit 長) のデータを読出すためのモジュールであり、読出しを要求する UserReq_RX を送ると Busy_RX が 1 となって返ってくる。1 ワード分のデータを読出した場合 Busy が解除され、その時 SiTCP_RECV の出力バスに流れているデータが有効となる。モジュールからの出力は再び UserReq_RX を送るまでは更新されない。

4.3.6 SiTCP_WRITE

SiTCP へ 1 ワード分のデータを送信するモジュールである。UserReq_TX と共にモジュールの入力バスへ 32bit 分のデータを流すと、送信要求から 1 クロック後にモジュール内の DFF にラッチされ自動的に送信が始まる。この時に同時に Busy_TX も発行される。32bit 分のデータが全て SOY のバッファへ書き込まれ、尚且つ AFULL が立っていないことが

Busy_TX が解除される条件である。Busy_TX が解除されたらユーザーは次のデータを送信することが出来る。

4.4 設計時に注意したこと、工夫したこと

MHTDC の設計が一番大変で工夫が必要であった。まず、250MHz のクロックで 256 個の DFF がドライブできないといけないが、discriminator 信号をサンプリングする回路からカウンタの値を保持するための DFF までの経路上に何も余分なロジックを挟まない工夫が必要であった。1ch のみでテストしていた段階では safety 用の非同期ロジックがいくつが挟まれていたが、チャンネル数を増やした際にその部分が timing error を出し正しく動かすことが出来なかった。そのため、その部分を直結にした所なんとか timing を満たすようになった。

次に大変だっ多部分が 1 us 分のデータをどうやって保持しておくかという点であった。当初使用される RAM の量を抑えるために、1 us でデータが自動的に消える制限時間付きの L1 buffer を実装してたが、EASIROC が発振した際に一周前のデータが消えてくれないという致命的なバグがあることが開発中に分かった。そのため BlockRAM を使用し 1 us 分のデータをとりあえず全て保持できるように作りかえた所動作が安定した。使える資材を無理に制限するのは望ましくないという教訓となった。

最後に MHTDC の resolution を出すために必要だった作業として ISE14.2 への更新と非同期バスの経路長を揃えるという作業が必要であった。どうしても MHTDC の timing resolution がでないという問題を開発段階で抱えていた。平均で 700 ps (rms) 程度の分解能しか出ず、時間分解能が 0.8 ns (σ) 必要な検出器を読出す回路としては不十分であった。まず、改善策として行ったことが非同期バスの経路長を揃えるということであった。MHTDC は discriminator 信号を 4 相のクロックでサンプリングするために、どこかで信号を 4 つに分ける必要がある。しかし ISE はこの経路長 (DFF までの距離) を揃えるようにコンパイルしてくれない。そのため、PlanAhead を用いて手動で経路が揃うように配置制約を与えた。(詳しくは Open-It の Tips 「SPARTAN6 で 1ns の MHTDC を作る」を参照の事) しかし、これだけでは不十分で何回コンパイルしても時間分解能が改善されなかった。これには ISE のバージョンが関係しており、当初開発は ISE12.2 で行つたが、これを ISE14.2 にアップグレードしただけで時間分解能が 700 ps から 450 ps まで改善された。routing アルゴリズムの違いによるものと思われるが詳しい理由は分からない。

4.5 失敗したこと、修正が必要な事

最大の失敗は試験基板であるのに SiTCP を分けて実装してしまったことである。SiTCP をユーザー FPGA に内蔵していればもっと性能があがったはずである。

FPGA の外部へ信号を出す際には最終段の DFF の位置を PAD の一番近くにするように配置制約を与えるのがセオリーらしいが、それを知らなかつたため最初はコンパイルするたびに動作が変わる回路が出来てしまった。

4.6 開発ツール

開発環境

- ISE version 14.2.

Design Goal

- Balanced

Design Strategy

- Xilinx Default

その他環境設定に関しては Xilinx_System_Setting_Report_E10_v5.htm を参照の事。

第5章 部品リスト

基板上で使用予定の部品リスト。

FPGA : Spartan6 LX25 FG(G)484

PROM : XCF08P VOG48

- AD8602ARMZ-REEL (汎用 OP アンプ)
 - 帯域 10MHz
 - 出力電流 30mA/channel
- SN65LVDS389DBT (Line Driver)
 - 8 channel
 - Input 電圧許容値 VCC(+3.3V) まで
 - LVCMOS to LVDS 変換
 - 630Mbps
- KC7050B50.0000C31A00
 - 50 MHz
 - 一般品
- SN65LVDS34 (LVDS 信号レシーバ)
 - 出力電圧 +3.3V
 - 入力電圧 -4 - 5V (Common mode)
- MC100EPT24DTG (ロジックトランスレータ)
 - LVTTL to LVECL
 - 入力電圧 3.3V
 - 電源電圧 -3.3V +3.3V
- SN65CML100 (ロジックトランスレータ)

- LVECL to CML
 - 電源電圧 -3.3V 、 GND で使用
- TPS3103K33
 - リセット信号 (Active Low)
 - コンパレータ内蔵

第6章 初回電源投入時の試験項目

初回電源投入時の試験は非常に重要なため、ユーザーガイドよりも先に説明を行う。実際に試験する際には必ずユーザーガイドにも目を通してから行う事。

6.1 テスト入力を用いた試験

ユーザーガイドを参照して正しくジャンパピンを設定する。SOYへの電源供給はEASIROCボードから行うのが標準である。ジャンパが設定できたら $\pm 6\text{ V}$ を印加する。 $+6\text{ V}$ 側は電流値が安定するまで2秒ほどかかることがある。この時点で全てのLEDが光らなければ問題が有り、最も可能性が高いトラブルはヒューズの半田不良である。デジボル等で導通を確認する。

次に試験用のソフトウェアを起動する。読み込むレジスタはデフォルト状態の物を使用する。ソフトウェアを起動すると自動でSlowControlが完了する。この時点でエラーが出た場合通信関連のトラブルであることが多い。問題が無ければAnalogOut HGからベースラインノイズが見えるはずである。この時 $+6\text{ V}$ 側の電流値が0.5 A程度になるはずであり、これより極端に低い場合はASICの不良が可能性がある。

最後にテスト入力を使って実際に信号を確認する。PreAmpCTest.txt内のA30(初期読み出しチャンネル)に1を設定してTransmit SCを行う。次に20 mV程度の正電圧矩形波を(EASIROCは正電圧読み出し。普段利用する負電圧を読み出す機器とは逆なので注意)In_Calib(HVコネクタの隣)に入力するとAnalogOut HGから信号が見える。ここまで確認が出来ればASICの健全性が確認できる。

6.2 MPPCへの電圧印加に関して

初回電源投入の項目とは若干内容が異なるが、非常に重要な注意事項なのでユーザーガイドと両方に同じ内容を掲載する。EASIROCのInputDACを破損するというトラブルが続出している。恐らくMPPCへバイアス印加と関連があるという所までは分かってるが、完全にトラブルの原因を理解できているわけでは無い。特に気を付ける事柄として、

- 絶対にHVと信号線をショートさせない。(端子を手で触らない)
- EASIROCの電源オフ時にMPPCの信号を入れない。
- MPPCへ定格電圧を印加するのはEASIROCへレジスタを設定してから。
- HVはゆっくり上げる。(スイッチ一つでいきなり70 V印加しない)

MPPC に電源を印加するための安全と思われる手順を示す。

- MPPC やその他ケーブルを電源オフ時に全て接続しておく。
- MPPC へ 5 V を印加する。(弱い逆バイアスにしておく)
- ボードの電源を投入しレジスタを設定する。
- バイアスを定格まで上げる。

この方法を守れば少なくとも我々は ASIC を故障させていない。

第7章 ユーザーガイド

本回路の実際の使い方を説明する。

7.1 各部説明

EASIROC ボードのジャンパピン、及び入出力に関して説明を行う。ボードの各部名称を図 7.1 に示す。

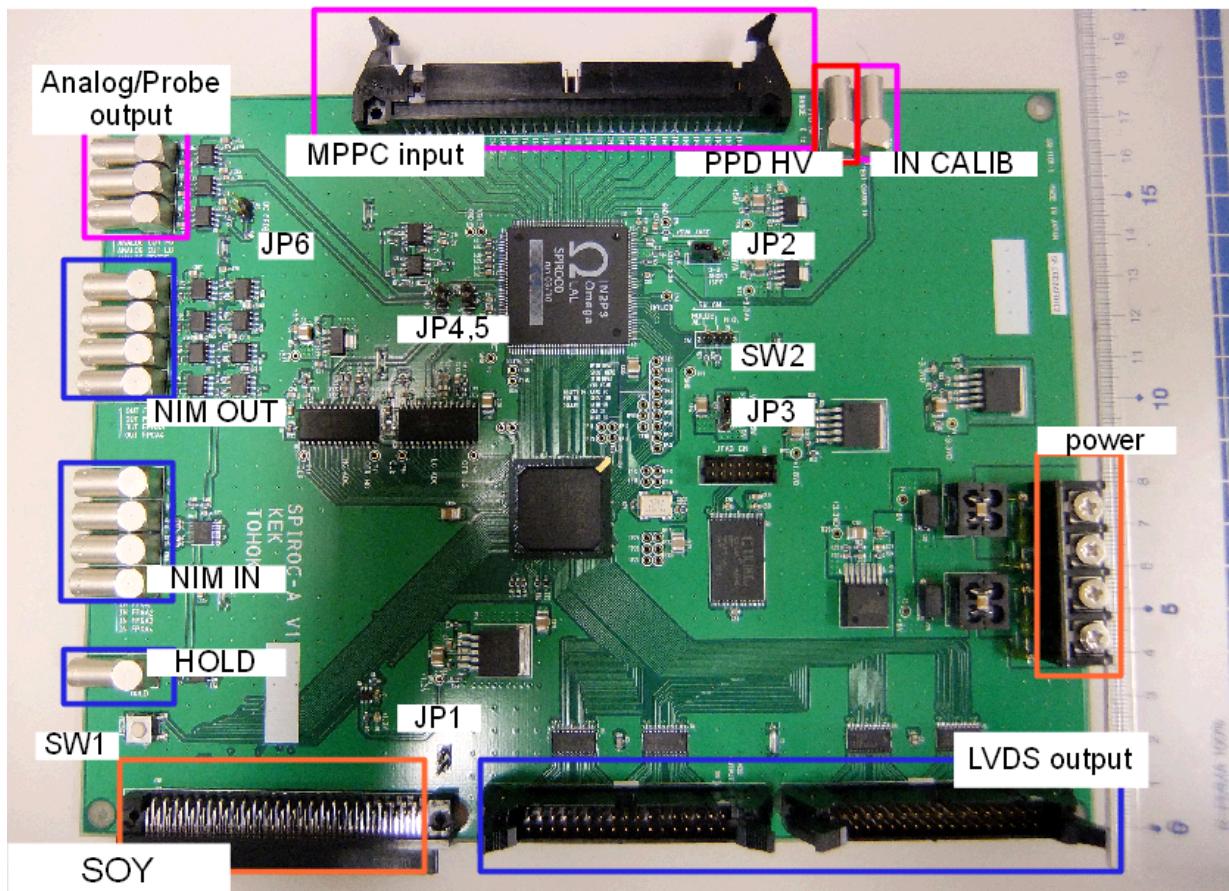


図 7.1: EASIROC ボードの各部説明

7.1.1 ジャンパピン

- JP1 SOY VCC SW

- SOY に EASIROC から電源を供給するかを選択。ショートすると SOY へ電源が供給される。
- JP2 2-3 SHORT 15PF
 - MPPC 入力信号を hig gain に通すかを選択。2-3 をショートで high gain 側が使用不可となるが、基本的に 1-2 ショートで使用する。
- JP3 VHI SELSW
 - FPGA の digital part に供給する電圧を選択する。1-2 ショートで +3.3V, 2-3 ショートで +1.8V を供給。3.3V で通常使用する。
- JP4
 - EASIROC からの hign gain 側出力を ADC へ送るかどうかを決める。ショートで ADC が利用可能となる。
- JP5
 - EASIROC からの low gain 側出力を ADC へ送るかどうかを決める。ショートで ADC が利用可能となる。
- JP6 DC PROBE
 - マルチメータなどで直接 EASIROC から出力される V_{th} などの DC 電圧を測定したいときに使用する。
- SW1 RESET
 - SlowControl モジュールを初期化する。現在は殆ど機能していない。
- SW2 HOLD
 - HOLD を有効にするかを選択。1-2 ショートで HOLD 信号ラインが接続され、2-3 ショートで high state ヘプルアップ。

7.1.2 アナログ信号入出力

アナログ入出力として 32 個の MPPC インプット、1 つのテスト入力、2 つのアナログ出力、1 つのプローブ出力を有する。以下はその詳細。EASIROC の analog output をオシロスコープで確認する際には、設定を終端抵抗 $10M\Omega$ の AC カップリングにする事。

- MPPC input
 - MPPC の信号を入力。コネクタの下部のピンから PPD HV へ入力した電圧が出力され、MPPC へのバイアスとする事が出来る。GND を接続すれば単純なペア線入力として利用できる。

- IN CALIB (TEST CHARGE IN)
 - テスト信号入力。EASIROC の入力用キャパシタンスをバイパスして直接信号を入力できる。正電圧を入力する。
- ANALOG OUT HG
 - High gain の整形増幅後 (slow shaper 後) のアナログ信号を出力。出力するチャネルは ReadSC で選択する。
- ANALOG OUT LG
 - Low gain の整形増幅後 (slow shaper 後) のアナログ信号を出力。出力するチャネルは ReadSC で選択する。
- ANALOG PROBE
 - 整形増幅過程の任意のアナログ信号をプローブする。ユーザーは PreAMP 出力 (HG,LG)、Slow shaper 出力 (HG,LG),Fast shaper 出力の 5 つから見たい信号を選択できる。
- PPD HV
 - この端子に電圧を印加すると、MPPC に本ボードからペア線の片方を使って電圧を供給できる。MPPC input の下側のピンへ印加される。

7.1.3 デジタル信号入出力

デジタル信号の入出力として FPGA へつながる汎用ロジック入出力それぞれ 4 つ、HOLD 入力が 1 つ存在する。これらの信号規格は NIM である。また、EASIROC の discriminator 出力が LVDS 規格で個別に出力される。

- OUT FPGA1 (OR32)
 - 32ch の discriminator output の OR を出力。
- OUT FPGA2 (Digital line)
 - ReadSC で選んだチャネルの discriminator 信号が出力される。
- OUT FPGA3 (DAQ BUSY)
 - BUSY 出力。BUSY の定義は common stop か HOLD 入力からデータのネットワークへの転送完了まで。
- OUT FPGA4 (/RUN_GATE)
 - DAQ モード時に 0 となる信号。複数ボードで同期を取りたいときに使用する。

- IN FPGA1 (common stop)
 - MHTDC の stop 入力。80 ns 程度の以上の幅を推奨。
- IN FPGA2 (Fast clear)
 - MHTDC のシーケンサステート初期化する信号。ADC は応答しないので注意。
- IN FPGA3 (Trigger)
 - データの転送用 Trigger。この信号が入力されるまではデータが転送されない。
- IN FPGA4 (予備)
 - 実際には機能が割り当てられているが、限定的な使用なのでここでは未実装扱いとする。
- HOLD
 - EASIROC への HOLT タイミング信号。NIM レベルに於ける 0 で TRACK、1 で HOLD 状態となる。
- LVDS OUTPUT (Discriminator 出力)
 - 各 channel の discriminator output をパラレルに出力。

7.2 初回電源投入

EASIROC 基盤へ初めて電源をかける際の手順とチェック項目。初回電源投入時の試験項目と内容が重なる部分も大きいが、非常に重要なので再度説明する。

7.2.1 ジャンパピンの設定

以下の設定がデフォルトでの使用方法となる。基本的に JP2-6 は全てデバッグや性能試験用なのでユーザーが触ることは殆ど無い。

- JP1... ショート
- JP2...1-2 をショート
- JP3...1-2 をショート
- JP4... ショート
- JP5... ショート
- JP6... 断線
- SW2...1-2

7.2.2 電源

ジャンパの設定が出来たら次に電源を投入する。ボード上部の電源供給部に $\pm 6V$ とGNDを接続する。電源ケーブルの接続は図7.2の参照の事。電源を供給するとボードについている赤色LEDが5つ点灯する。正常なボードではこのとき $+6V$ 側に約0.45Aの電流が流れる。これより極端に少ない場合ハード的な問題を抱えている可能性が高い。

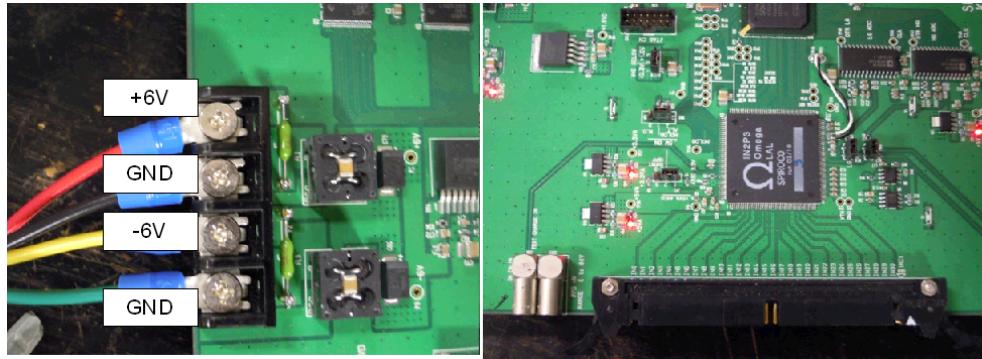


図 7.2: EASIROC ボードの電源供給部

7.2.3 ソフトウェア実行とEASIROCのコンフィグ

SOYに接続しボードを制御するためのプログラムを起動する。今回路の制御用プログラムはLinux環境で作成、及び試験がされておりMacOSでの動作は保証しない。cygwinでの動作報告は挙がっているが推奨はしない。実行体を構成するファイルはeasiroc_for_v7.tar.gzを展開した際に生成される以下のファイルによって構成される。

- ConnectSiTCP.cc, .hh
- test.cc, .hh
- class_FileManager.cc, .hh
- class_ParameterManager.cc, .hh
- class_StringManager.cc, .hh

gcc, g++が導入されている環境であれば make すれば easiroc という実行体が生成されるはずである。制御用のレジスタの設定のしかたや、他のファイルの詳細についてはソフトウェアの項目で詳細に述べる。

生成された実行体の起動するには以下のコマンドを入力する。

./easiroc [IP address] [DAQ mode]

IPはSOYのIPを指す。DAQ modeは今は無関係なのでとりあえず3を設定する。プログラムを起動すると以下のメッセージがプリントされるはずである。

SiTCP Master :: Connection Done

bit0 << >> bit7

...!..!.... word_0

....!!!! word_1

..... word_2

!..... word_3

..... word_5

ASIS Initialize : Done

Slow Control : Done

ReadSC_Channel

Ch 32 << >> Ch 0
..!.....

Read Slow Control : Done

#D : DAQ mode is 3

I'm : 192.168.11.100

Please select

1. Transmit SC
 2. Transmit Read SC
 3. ASIC Initialize
 4. Transmit Probe
 5. Connection Close
 6. Start DAQ
 7. Debug

最初の SiTCPMaster :: Connection Done が TCP 接続が完了したことを示す。この表示がそもそもでない場合 PING も通らない可能性が高いので、ネットワークの設定を再度確認する。その後に表示される ASCII Initialize, Slow Control, Read Slow Control Done という表示はそれぞれプログラムを立ち上げた際の EASIROC のコンフィグが行われていることを示す。この画面が表示されれば EASIROC のコンフィグは完了している。

EASIROC のコンフィグが完了すると +6V 側に流れる電流量も 0.7A へ増加するはずである。プログラムはエラーを出さないが全く電流が変化しない場合、ASIC の不良が考えられる。また、ANALOG OUT HG の信号をオシロで確認して ASIC の健全性を確認する事

も可能である。図 7.3 にプログラム起動前後の ANALOG OUT HG の出力の変化を示す。コンフィグ前はこの信号線は浮いているため何も表示されないが、コンフィグ後はデフォルトであれば ch30 の信号が出力されている。EASIROC へ何も接続していない状態でも図 7.3 の右図の様に 2mV 程度のベースラインノイズが見えるはずである。この場合も、コンフィグ前後で何も変化が無い場合 ASIC の不良が考えられる。

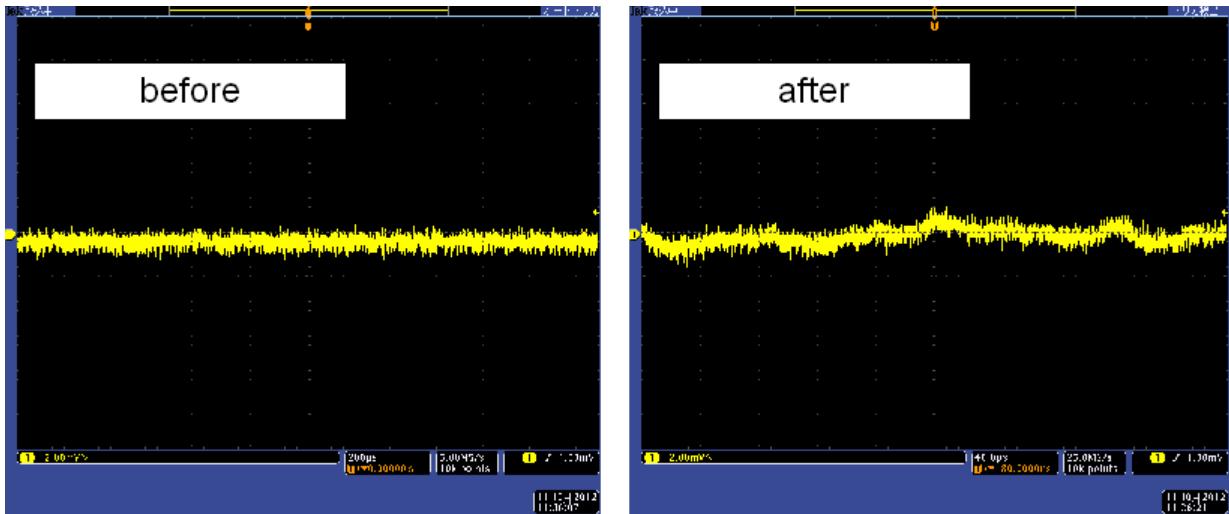


図 7.3: ANALOG OUT HG の変化

7.3 ソフトウェアの使い方

より詳細なソフトウェアの使用方法について述べる。

まず、実行体を構成するソースコード以外の展開されたファイルやディレクトリについて述べる。fullcheck.C は CERN ROOT のマクロファイルであり、DAQ モードで作製されたファイルを引数に取り実行すると histogram と tree を生成するので、簡易的なチェックに利用できる。非常に簡単なマクロなので生成される histogram と tree の名前などはソースコードを読めば分かるはずである。register_labview.txt と SlowControlSP-A.txt はデバッグ用の default register set が格納されたファイルである。本来ユーザーにとっては必要なファイルであるが、はずすのが面倒なため残してある。easiroc_for_v7.tar.gz を展開すると autoconfig, setup, tmp の 3 つのディレクトリが更に展開される。この内ユーザーが変更を加えるのは setup 内のレジスタファイルである。

次に、プログラム実行に表示されるメニューについて述べる。先ほど述べたようにプログラムが正しく起動すると以下のようないいメニューが表示される。

```
I'm : 192.168.11.100
```

```
Please select
```

1. Transmit SC
2. Transmit Read SC
3. ASIC Initialize

```

4. Transmit Probe
5. Connection Close
6. Start DAQ
7. Debug
input # =====>

```

先頭の IP は今接続している SOY の IP を示している。それ以下に表示されているメニューを実行するためには、番号を入力してエンターを押す。Transmit SC は Slow Control の実行を行う。setup ディレクトリ以下のレジスタを転送する。Transmit Read SC は Analog Out と Digital line の出力チャンネルの設定を行う。ReacSC_Channel.txt の内容が転送される。ASIC Initialize は普段ユーザーは使用しない。Transmit Probe は Probe 出力の内容を設定する。Connection Close は TCP 接続を切断しプログラムを抜ける。Start DAQ は DAQ モードへ切り替え、データを取得する。Debug コマンドはユーザーは使用しない。

7.3.1 Transmit SC

Transmit SC は Transmit Slow Control の略である。このコマンドを実行すると自動的にレジスタが EASIROC へ転送される。Transmit SC は現在保存されているファイルをロードして転送するため、プログラムを立ち上げた状態でもテキストエディタでファイルを変更してから、再び Transmit SC を選択すれば変更内容が EASIROC へ反映される。ここではレジスタファイルの設定法について述べる。setup ディレクトリ以下にあるレジスタファイルを開くと図 7.4 のような文字の羅列が見える。この部分は各レジスタファイル共通の項目で、レジスタの定義部分にあたる。前 3 つの BitNum, ArrayNum, BitOrder は使用する bit 長、配列の数、bit order を示すキーワードでユーザーが編集することは無い部分である。その後の Axx という部分が実際のレジスタの値となる。A は Array の頭文字で、A 何番まで有効なのかが ArrayNum に記述されている。例えば、図 7.5 には DAC.txt の内容が示されている。EASIROC の Vth DAC は 32ch で共通のため値は一つしかない。そのため、ArrayNum が 1 で A0 のみ有効となる。図 7.6 では Input8bitDAC.txt の内容が示されているが、DAC.txt と違い各チャンネルの値を設定しなければならないため、ArrayNum は 32 で全ての配列が有効となる。どのような数字を設定すべきかは大体ファイル内部にコメントが書かれているが、詳しい説明はレジスタテーブルの項目で説明する。Enble 系のレジスタで負論理の場合コメントで Active low と記されているので、0 と 1 を間違えないように気を付ける事。

```

BitNum ArrayNum BitOrder A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11
A12 A13 A14 A15 A16 A17 A18 A19 A20 A21 A22 A23 A24 A25 A26 A27 A28
A29 A30 A31

```

図 7.4: レジスタ定義部分

例 1.

```
#Register definition DAC
StartDefinition
BitNum ArrayNum BitOrder A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11
A12 A13 A14 A15 A16 A17 A18 A19 A20 A21 A22 A23 A24 A25 A26 A27 A28
A29 A30 A31
EndDefinition
10  1    0    850  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  ←その他は無視
1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
```

図 7.5: DAC.txt の例

例 2.

```
#Register definition Input8bitDAC
StartDefinition
BitNum ArrayNum BitOrder A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11
A12 A13 A14 A15 A16 A17 A18 A19 A20 A21 A22 A23 A24 A25 A26 A27 A28
A29 A30 A31
9   32    0    289 287 287 370 350 287 284 287 286 287 284 285 283 282
282 406 287 285 291 289 290 291 293 288 295 295 294 296 294 298 386 350
```

図 7.6: Input8bitDAC.txt の例

7.3.2 Transmit Read SC

Read SC は Slow Control とは違う信号線を使うためコマンドが分かれている。ここではどのチャンネルを ANALOG OUT HG、LG、及び Digital Line へ出力するかを決める。ロードされるファイルは ReadSC_Channel.txt であり、ファイル構造は Slow Control 用の物と同じである。出力したいチャンネルに 1 を書き込んで Transmit Read SC を選ぶと出力チャンネルが変更される。

7.3.3 Transmit Probe

Probe 出力から何を出力するかを決めるコマンドである。Probe は Slow Control と Read Slow Control とは独立の機能で、ユーザーが何も設定しなければこの信号線は浮いている。Transmit Probe を選択すると以下の様な選択肢が現れるので、どの種類の信号を出力するか選択する。

```

Probe select
1. OUT PA HG
2. OUT PA LG
3. OUT SSh HG
4. OUT SSh LG
5. OUT FS
6. Next (未実装)
7. Previous (未実装)
8. Reset
input # =====>

```

PA は PreAMP の略、SSh は Slow Shaper の略、FS は Fast Shaper の略である。Reset は Probe の状態を初期化する。出力する信号の種類を選択したら次にチャンネルを選択する。ここで選んだチャンネルは Read SC とは独立である。

7.4 モニターモードでの使用

DAQ を使用しないモードでの簡単な使用法について述べる。この状態は単純に EASIROC をコンフィグしただけの何もしない状態であるが、便宜上モニターモードと呼ぶことにする。EASIROC を ASD カードとして使う場合もこの状態で放置するだけである。MPPC を使って信号を確認するために必要なセットアップの例を図 7.7 に示す。このモードでは Input8bitDAC と 10bit DAC の値の変更が必要となる。

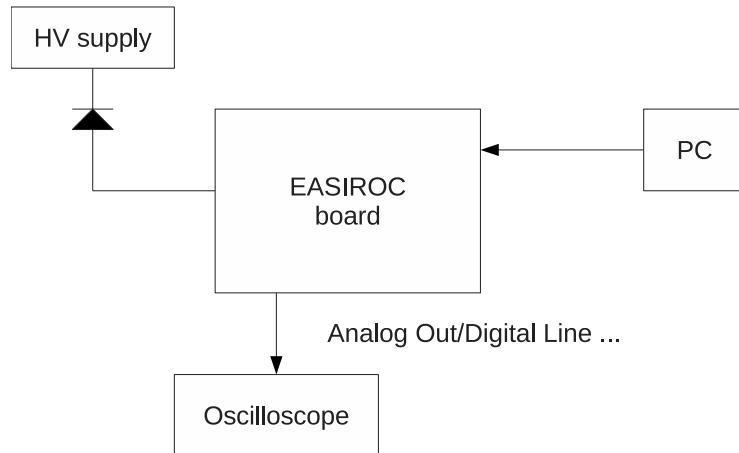


図 7.7: モニターモードで使用するときの例

Input8bitDAC は前述の通り MPPCへのバイアスをチャンネル毎に設定できる EASIROC の重要な機能である。Input8bitDAC のレジスタファイルは図 7.6 で示したようになっており、BitNum は 9 である。この部分のレジスタは InputDAC On + register value となっており、9bit 目は DAC の On/Off をになっている。基本的に常に InputDAC は On なので、 $256 + \text{設定したい値} = \text{書き込む数字}$ 、となる。また更に、Input8bitDAC は信号線上へ DC

電圧を乗せるので、図 7.8 の様に MPPC bias との差を縮める事でバイアスを調整する。設定したい値を 0 とした時 InputDAC からの出力は最大となり MPPC へのバイアスは最も小さくなる。逆に設定したい値を 255 とした時 InputDAC からの出力は最小となり、HV 電源の電圧がそのまま MPPC へ印加される。デフォルトでは Input8bitDAC の外部参照電源を使っているため、0 (256) で 4.5V が出力されており、255 (511) でほぼ 0V が出力されることになる。

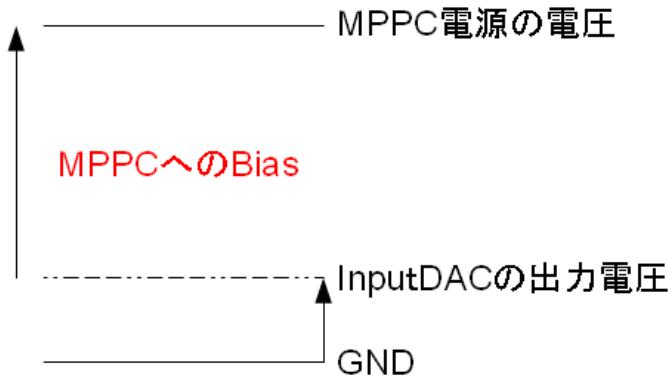


図 7.8: Input8bitDAC の概念図

V_{th} DAC の値は DAC.txt で変更する。 V_{th} は各チャンネル共通のため InputDAC で各 MPPC のゲインを揃えてから値を設定する。DAC.txt 内の数字が大きい程 V_{th} は低くなり、小さい程高くなる。ベースラインは 850 付近に存在するがチップによって個体差があるため、毎回探す必要がある。

7.5 DAQ モードでの使用

DAQ を使用しデータを取得する場合について説明する。本回路は ADC と MHTDC が独立しているため、どちらか片方のみを使用するという使用法も可能であるが、ここでは同時に使う場合を例に述べる。Start DAQ を選択すると出力するデータファイルのファイル名の入力を求められる。次に取得するイベント数を入力する。目標イベント数に達するか Ctrl-C で強制的に止めると、プログラムはデータファイルを閉じて DAQ モードの終了処理に入る。プログラムが DAQ の終了を検知してから FPGA が実際に止まるまではラグが存在するため、SOY からのデータ転送が無くなるまではプログラムはデータを吸い出し続ける。この際、dummy data という表示がしばらくプリントされ続けるが異常ではない。データファイルは既に閉じるのでこの時のイベントは全て捨てられる。TCP recv からタイムアウトが返ってきた時点でプログラムは DAQ プロセスを抜けメニュー一覧へ戻る。

DAQ を使用する際のセットアップ例を図 7.9 に示す。(あくまで一例である) HOLD、Common Stop、Trigger は独立の信号のため別の delay line が必要になる。Coincidnce モジュール上で DAQ_BUSY を VETO として入力しているが、3 つの信号が入力される順番を守るためにも、可能であれば /RUN_GATE も VETO に参加した方が良い。これは、例えば DAQ モードに入ったタイミングによっては、HOLD は認識されなかつたが Trigger は

認識された、という状況になる場合がある。その場合常に一つ前のイベントの Trigger でデータ転送が行われることになるので、望まない動作をする可能性が高い。/RUN_GATE が VETO に参加していれば、必ず DAQ モードに入ってから信号が出るのでこのような不具合が無くなる。

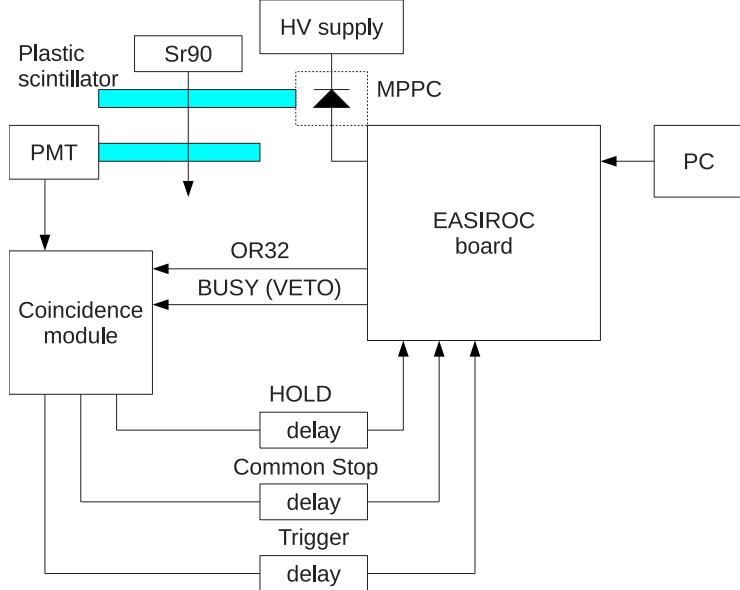


図 7.9: DAQ 使用時のセットアップ例

7.5.1 DAQ 信号のタイミング調整

QDC では波形が Gate 内に入るように Gate タイミングを調整しなければならない様に、本回路でも ADC を使うためには信号を Hold するタイミングを調整しなければならない。調整はモニターモード時にオシロスコープを使って行う。モニターモード時は Hold へ信号を入力すると FPGA 内部の Gate Generator もどきで信号幅を 2 us 程度に調整してから EASIROC へ入力する。そのため、どのような信号を Hold 入力しても 2 us 程度は HOLD 状態が維持されるためタイミングの調整が行いやすい。図 7.10 に示した信号は、上から HOLD された Analog Out HG、Probe 出力の SSh HG、及び Digital Line である。信号が最大の時に平坦になっているのが正しいタイミングで HOLD されている状態であり、図 7.10 の様になるように HOLD タイミングを調整する。

また、MPPC の生信号を起点とした DAQ 全体の信号タイミングチャートを図 7.11 に示す。HOLD 信号は、当然 Slow Shaper の成形時間に間に合うように入力しなければならない。MHTDC の時間窓が 1 us であるため、Common Stop は必要な信号から 1 us までがレンジとなる。Trigger は一つ入力しても良いが、想定上は HOLD や Common stop よりも後である。これらの信号の幅は、100 ns 程度以上が望ましい。

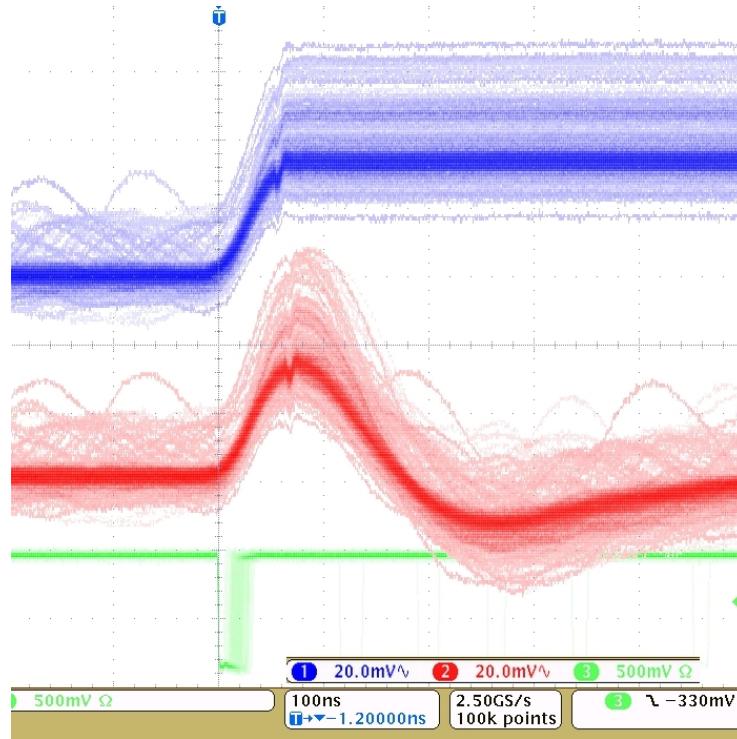


図 7.10: HOLD が正しいタイミングで入力されている時の信号

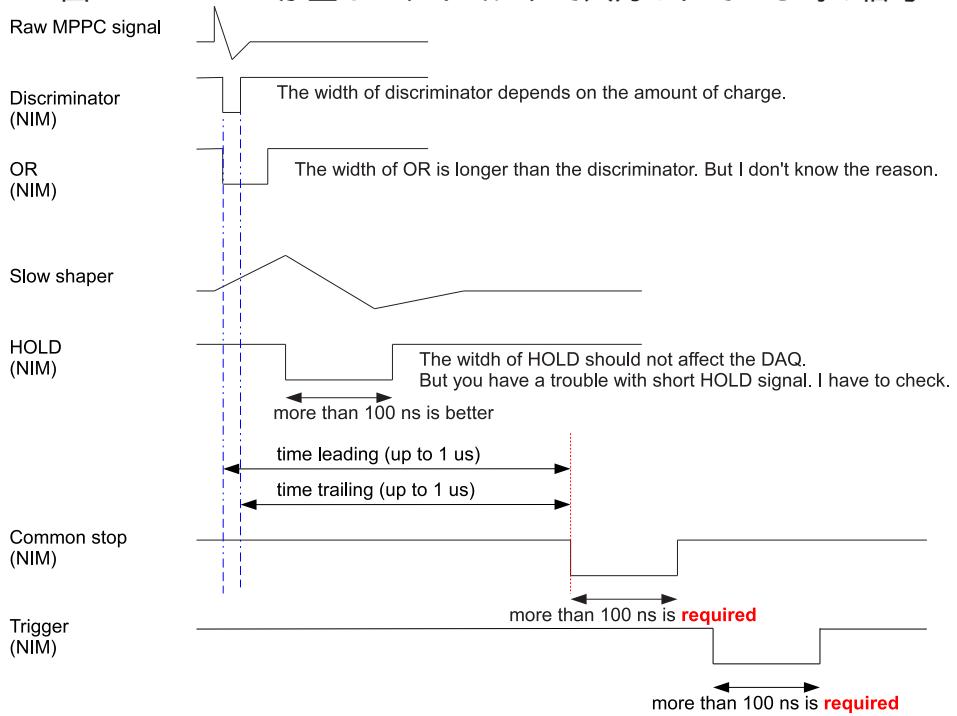


図 7.11: DAQ 全体のタイミングチャート

7.5.2 データ構造

本回路から転送されてくるデータは 96bit 長の header + データである。footer は存在しない。1 word を 32bit として扱う。

Header part	31 bit	0bit
	[8bit][8bit][8bit][8bit]	
Header1	[0xfffff] [0xea0c]	
Header2	[reserved] [Number of word]	
Header3	[EventCounter (12bit)] [reserved]	

Header 部は Magic word 0xfffffea0c、そのイベント内のワード数、イベント番号で構成される。ワード数は Header のワード数を含まない。EventCounter は FPGA 内部のセルフカウンターであり、イベントを転送した回数を数えている。

ADC part	31 bit	0bit
	[8bit][8bit][8bit][8bit]	
data1	[0x81,0x60] [ch (5bit)] [000 + X +data(12bit)]	
data2	[0x81,0x60] [ch (5bit)] [000 + X +data(12bit)]	
.....		
MHTDC part		
data1	[0xcc] [ch (5bit)] [LT+00000+data(10bit)]	
data2	[0xcc] [ch (5bit)] [LT+00000+data(10bit)]	
.....		

ADC のデータは先頭の 8bit が High gain か Low gain かを示しており、0x81 で High、0x60 で Low である。次の 8bit の内下位 5bit がチャンネル番号を示す。下位 16bit の内、13bit 目がオーバーフロービットで、1 の場合 ADC がレンジオーバーを起こしている。最後の 12bit 分が ADC のデータである。

MHTDC のデータは 0xcc から始まる。チャンネルの部分は ADC と共通である。下位 16bit のうち、16bit 目が LT ビットで、0 で Leading、1 で Traling を示す。最後の 10bit が MHTDC のデータである。

第8章 レジスター一覧

EASIROCで使用されているレジスターについての一覧である。

8.1 ASIC Initializeで使用されているレジスタ

ASIC Initializeでは word_x.txt が全てロードされる。実際に FPGA 内で使われているのは word0, 1, 3 のみである。これは元々は FPGA のデバッグ用に使用されていた名残で、現在は殆ど機能していないレジスタがほとんどである。

8.1.1 word_0.txt 内

- bit 0 (none)
 - 未使用。
- bit1 (Raz_Ch)
 - analog buffer に hold されている情報を強制的に reset する。discriminator output のラッチ状態を解除するのにも使用。現在は常時 0。
- bit2 (Val_E)
 - valid event の略。discriminator の動作を規定。1 だと discrimination、0 だと全ての discrimination の動作が off。現在は常時 1。
- bit3 (none)
 - 未使用。
- bit4 (en_link)
 - 正式には enable serial link。slow control 等シリアル転送ラインをアクティブにする。常時 1。
- bit5, bit6, bit7
 - 未使用。

8.1.2 word_1.txt 内

- bit0 (load)
 - 正式には slow control load。 slow control 等で送った情報を ASIC 内で反映させるために使用。ユーザーが触る事は無い。
- bit1 (cycle)
 - 正式には start slow control cycle。この信号が 1になると bit の転送が開始される。ユーザーが触る事は無い。
- bit2,bit3
 - 未使用。
- bit4 (rst_pa)
 - reset PreAMP。プリアンプの設定をリセットする。負論理信号。
- bit5 (select)
 - 今転送している bit が slow control register なのか prove register なのかを規定する。1 で slow control、0 で prove を意味する。ユーザーが触れる事は無い。
- bit6 (rst_r)
 - reset read register を意味する。transmit read SC で送った情報をリセットする。負論理。
- bit7 (rst_sc)
 - reset slow control。transmit SC で送った情報をリセットする。負論理。

8.1.3 word_3.txt 内

- bit0 (pwr_on)
 - ASIC power on。1 で ASIC がアクティブ状態になる。常時 1。
- bit1-7
 - 未使用。

8.2 Slow control register

Slow Control 用のレジスタには一つだけ命名規則が存在する。En と先頭に付くものは enable を意味する。1 で電圧をかけてアクティブし、0 で電圧を落として使用不可にする。電圧をかけるかどうかを規定するのでもっとも強力な register となる。pp と付くものは power pulsing mode を意味する。元々は ILC 用に作られたので、ビームがきていない間は idle 状態にして消費電力を抑えられるように設計されている。1 で常時 on, 0 で power pulsing mode となるが、power pulsing するための信号を送っていないので常時 1 で使用する。

- DAC.txt
 - discriminator の閾値を設定する。閾値は 32ch 共通。850 付近がペデスターの電位となる。数字が小さいほど閾値が上がる。チップ毎に個性があるので、あるボードで設定した値が他のボードで使えるとは限らない。
- DACslope.txt
 - discriminator 用の DAC の電圧スロープを切り替える。0 だと 1 の場合よりも DAC.txt の 1bit でより大きく閾値電圧が変動する。またオフセットも変わる。通常は 1 で使用する。
- DiscriMast.txt
 - discriminator output を遮断する。チャンネルごとに動作を規定可能。1 で on, 0 で output を遮断。発振チャンネルが出た場合等に使用する。
- En32Trig.txt
 - 32 チャンネルの parallel discriminator output を enable にする。負論理。discriminator 自体の動作を規定するわけではないので注意。これを off しても discriminator 自体は動き続ける。
- EnBgap.txt
 - 常時 1 で使用。
- EnDAC.txt
 - discriminator の電圧閾値を決める DAC を enable にする。
- EnDigOut.txt
 - enable digital line output。常時 1。
- EnDiscri.txt
 - enable discriminator。常時 1。
- EnFSh.txt

- enable fast shaper。常時 1。
- EnHGOTAq.txt、EnLgotaq.txt、EnProveOTAq.txt
 - 常時 1。 HG は high gain, LG は low gain を意味する。
- EnHGPA.txt, EnLGPA.txt
 - enable PreAMP。常時 1。
- EnHGSSh.txt, EnLGSSh.txt
 - enable slow shaper。ADC を使用しないで ASD カードとして使用する際は 0 にしても良い。
- EnInputDAC.txt
 - MPPC の bias 調整に使う DAC の動作を規定。
- EnLVDS.txt
 - Raz_Ch と Val_E は LVDS で受信している。その受信部分を enable にする。
- EnOR32.txt
 - OR32 信号を enable にする。負論理。
- EnTandH.txt
 - track and hold、つまり analog buffer に電荷情報をストックする部分の状態を規定する。常時 1。
- HGPAComp.txt, LGPAComp.txt
 - PreAMP の feed back capacitance の補償コンデンサの値を決める。出力信号がゆがむのを防ぐらしいが、未確認。デフォルトは HG が 500 fF、LG が 3 pF。
- HGPAFback.txt, LGPAFback.txt
 - PreAMP の feed back capacitance の値を決める。容量が大きいほど gain は小さくなる。デフォルトでは HG/LG 共に 200 fF が設定されている。各コンデンサ容量に対する増幅度は easiroc のデータシートに記載されている。
- Input8bitDAC.txt
 - MPPC の bias 調整の DAC の出力電圧をチャンネルごとに設定する。9bit 構造で、9bit 目が DAC の on/off。動作が不安定になるので off にしないこと。下位 8bit 部分でかける電圧を決める。0 で最大電圧が出力され、255 でほぼ 0V となる。また、このボードでは InputDAC の 1bit は 20 mV で固定されている。そのため、ref8bitDAC が 0(internal 2.5 V) の場合、150 程度で InputDAC の出力

が 0V になり、逆に 1(external 4.5 V) の場合、240 程度で 0V を出力する。稀にこの値を越えると信号が出なくなる ASIC が存在する。0V ぎりぎりでの使用は避けた方が良い。

- LatchDirect.txt
 - Discriminator 出力の方法を切り替える。出力方法には一般的なモード (overlap 型) とラッチモードがあり、ラッチモードでは Raz_Ch を受け取るまで high state を保ち続ける。現在は一般的なモードで利用している。
- PreAMPCtest.txt
 - PreAMP にテストチャージを入れるための設定、及び PreAMP の disable 設定を行う。2bit 構成で、下位 bit がテストチャージ入力を入れるチャンネルを決める。上位 bit が PreAMP の disable 設定である。上位 bit はいくつ複数のチャンネルで立てて良いが、下位 bit は複数チャンネルに立ててはいけない。その場合の動作は未定義である。
- TimeCHGSSh.txt, TimeCLGSSh.txt
 - slow shaper のピーキングまでの時間を 25 ns から 175 ns の間で変更。デフォルトは 50 ns。
- biasLGPA.txt
 - low gain 側の PreAMP にかける bias 電圧をデフォルトよりも下げる事が出来る。これにより消費電力を抑える事が出来る。デフォルトでは 0。下げたい場合は 1 を入力。
- biasTandH.txt
 - analog buffer にかける電圧を下げる。電圧を下げるとき立ち上がりがなまるので Conversion の際に送る CLK_READ の周波数を下げる必要があるが、消費電力を抑えられる。この 2 つのレジスタは消費電力を抑える必要のある気球実験のために実装された。
- pp*.txt
 - 未使用。
- ref8bitDAC.txt
 - InputDAC の参照電位を切り替える。0 で ASIC 内部のレギュレータが生成する 2.5V を参照し、1 でボード上に乗っている石から供給される 4.5V を参照する。内部の方がノイズが少ないとのこと。

8.3 Read slow control

ReadSC_Channel.txt を使用してどのチャンネルの信号を analog out HG, LG に出力するかを決める。analog probe とはレジスタが別。読み出したいチャンネルを 1 にして転送する。2bit 以上 1 を立てて送ろうとした場合プログラムが強制終了する。

第9章 既知の問題点

問題として理解されているが解決策の無い物をリストする。

- InputDAC が壊れる。
 - InputDAC が破壊され bias の調整が出来なくなるだけでなく、ASIC そのものが使用不可となる。解決策は今のところ分かっていないが、初回起動時の試験項目で述べた手順を守って MPPC へ電圧を印加すれば壊れないようである。(我々は MPPC HV を EASIROC ボードから供給するのをやめたが、それも関係あるかもしれない)
- Low gain 側の線型性がでない。
 - ASIC そのものの問題で Low gain 側の線型性はよくない。使用する場合は校正関数を知っておく必要がある。
- InputDAC に大きい値を設定すると信号が消える場合がある。
 - InputDAC の出力が 0V 付近になるような大きな値を設定すると信号がでなくななるというバグを持つ石が存在する。全ての石がそうではないが、注意しなければならない。
- Probe で Fast shaper を選ぶとノイズが乗る。
 - その様な仕様らしくあきらめるしかない。
- Probe で OutPA を見たいが殆ど見えない。
 - 信号の頭の部分の極一部しか出力されない仕様らしい。
- MHTDC の 1000ch 以上の部分は信用できない。
 - 各 TDC モジュールと CommonStopManager のタイミング差の関係で 1000ch を越える部分のデータは信用できない。

第10章 良くある質問

- InputDAC の電圧を測るには?
 - InputDAC は殆どドライブ能力が無いので、通常のテスター位の内部抵抗だと正しい電圧が測れない。マルチメータのように $G\Omega$ 程度の抵抗を持った測定器が必要となる。
- 複数のボードを同時に使うには?
 - 簡単な方法としてはプログラムを複数立ち上げて、全てのボードの /RUN_GATE の OR と BUSY の OR を Trigger モジュールの VETO に入れる事である。
- FPGA 内部で複雑な Trigger を組むには?
 - 可能であるが、MHTDC が動かなくなる可能性が高いのでやめた方がいい。