気球搭載用超伝導スペクトロメータのための FADC Interfaceの開発

神戸大学自然科学研究科物理学専攻
987S115N　土岐 仁謙

2000年2月10日

概 要

BESS (Balloon-Borne Experiment with a Supercondacting Magnet Spectrometer) 実験では宇宙における素粒子現象の探究を目的に、超伝導ソレノイドコイルを用いた気球塔載型スペクトロメータで高度 35 km の上空及び地上において各種宇宙線流束の精密な測定を行っており、宇宙・素粒子・天文学的に貴重なデータを得ている。

BESS 実験では測定器の性能向上と飛行時間の長期化を目指して、研究開発が続けられている。この論文では現在開発が最終段階にさしかかっている、消費電力が従来の半分以下に削減された新しい FADC システムの内、FADC モジュールと従来のデータ収集系を仲介する FADC Interface について述べる。

この FADC Interface の特徴は、回路の大部分を FPGA (Field Programmable Gate Array) の内部に収めたことである。FPGA とは内部の回路を何度でも書き換え可能な IC である。FPGA を用いたことにより、大規模で複雑な回路を設計したにもかかわらず、部品数の大幅な削減と回路の変更が容易であるという柔軟性が得られた。

目 次

第1章　BESS実験について

1.1　BESS実験

BESS実験 (Balloon-Borne Experiment with a Supercondacting Magnet Spectrometer) は宇宙における素粒子現象の探究を目的とした、超伝導ソレノイドコイルを用いた気球塔載型スペクトロメータによる気球実験である。カナダの Lynn Lake (北緯 36°) で毎年夏に高度 35 km を 1 – 2 日間飛行して、陽子・反陽子・ヘリウム・μ 粒子などの各種宇宙線の流束を精密に測定している。なかでも反陽子流束のスペクトルの測定 (図 1.1 参照) については、質量の同定という確実な方法を用いての初めての測定であり、BESS 実験の最も重要な物理的成果の一つである。

1.1.1　低エネルギー領域の宇宙線反陽子の測定

BESSが主に測定しているのは運動量が 0.2 から 70 GeV の反陽子である。反陽子は宇宙空間にほとんど存在しない。したがって宇宙から地球へ降り注ぐ一次宇宙線の主成分は陽子であり、低エネルギーではその数は非常に多く、高エネルギーになるにつれて急激に減少していく。一次宇宙線の陽子が地球の大気と反応すると二次的な粒子が生成されることがあり、これは二次宇宙線と呼ばれる。地上で観測される宇宙線、特に宇宙線陽子はこの一次宇宙線と二次宇宙線が混ざったものである。また地上で観測される宇宙線反陽子の大部分は二次宇宙線として生成されたものである。

二次宇宙線として生成された反陽子は、元々高エネルギーの一次宇宙線陽子が地球大気と反応して生成されたものなので、ある程度のエネルギーを持つ確率が多い。図 1.1 のスペクトラムを見ると分かるが、二次宇宙線反陽子の数は運動量が 2 GeV 周辺が最大であり、より低エネルギーの領域では数が減少している。したがってもし低エネルギー領域での反陽子スペクトラムに異常に数が多い領域が見つかれば、それは外宇宙からの一次宇宙線反陽子の影響であると考えられる。

一次宇宙線反陽子の生成源としては、宇宙空間での陽子・陽子の衝突による反陽子生成、微小ブラックホールの蒸発による生成、ダークマターの対消滅、などが考えられている。宇宙空間での陽子・陽子衝突による反陽子の生成は宇宙線伝播モデルにより計算できるので、BESSの実測値によりモデルの検証・修正が可能となる。また微小ブラックホールの蒸発やダークマターの対消滅などの反陽子源は、それらの数の上限を決定することができる。

現在の BESS の測定では、まだ低エネルギー領域での宇宙線反陽子のデータ数が不十分であり、統計誤差が大きいため十分な検証ができない。そこで将来計画

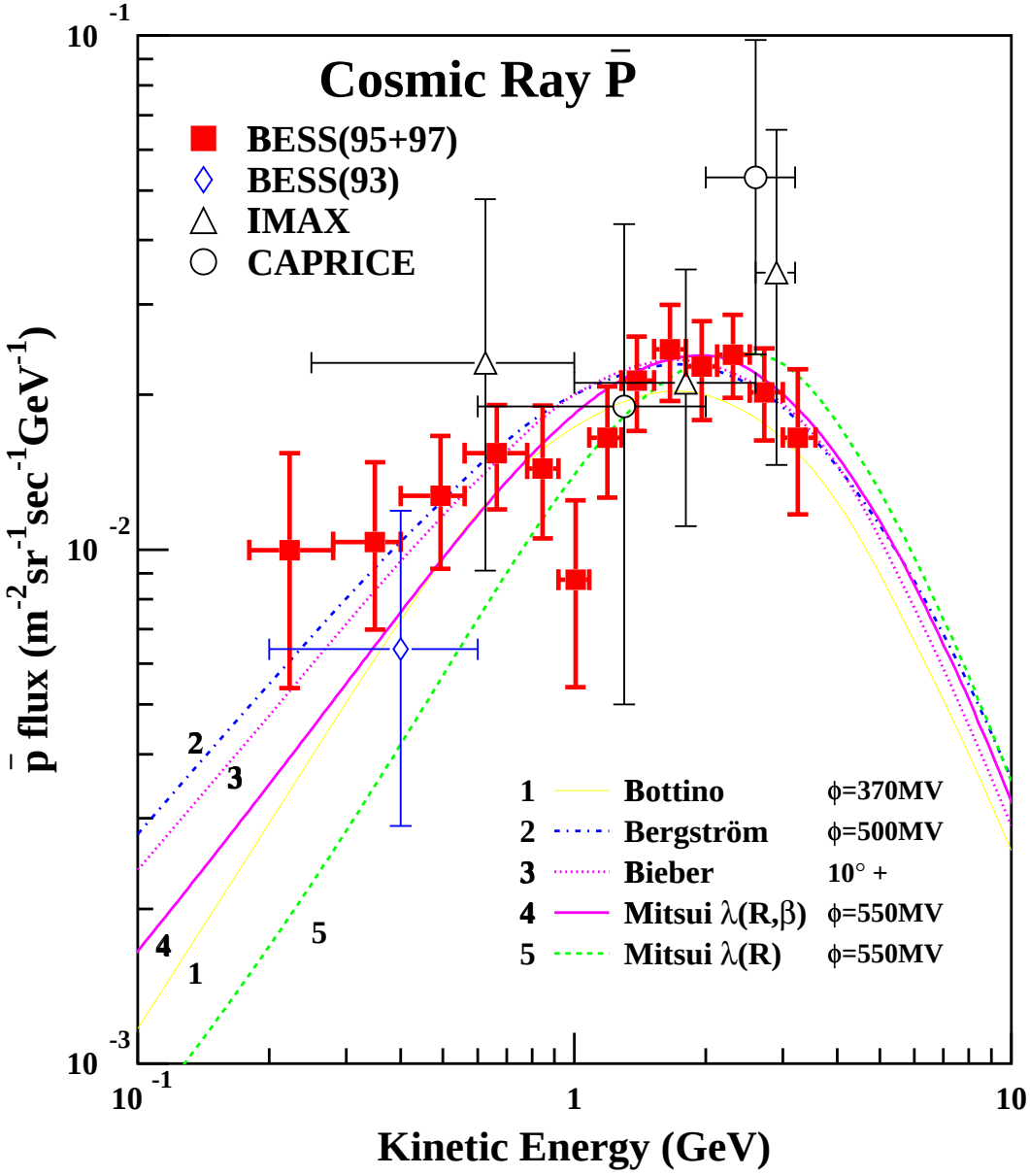


図 1.1: 反陽子流束のスペクトラム

として、BESS の飛行時間を現在の 1–2 日間から、5 日間および 20 日間に延長することが考えられている。BESS では測定器の電源として現在はリチウム電池を使用しているため、測定器の動作時間は限られており、現在の 2 日間の飛行時間は限界に近い。そこで現在 BESS では各測定器・電子機器の低消費電力化が進められており、BESS 全体で 1200 W の消費電力をその半分以下に削減することで 5 日間の飛行に対応する予定である。さらに 20 日間の飛行では、電源をリチウム電池から太陽電池に変更して対応する予定である。

1.1.2 高エネルギー領域の宇宙線陽子の測定

スーパーカミオカンデの大気ニュートリノの観測によりニュートリノ振動の存在が示唆されている。しかし大気ニュートリノを生成する元となる陽子・ヘリウムなどの一次宇宙線の強度はいまだ正確には測定されていない。例えば一次宇宙線の主成分である陽子の強度は、これまでの測定結果によると最大で 1.5 倍の違いがあるために、このデータを元に計算された大気ニュートリノの強度も 1.5 倍の不確定性を持ってしまうことになる。そこで BESS では広いエネルギー範囲に渡って一次宇宙線の強度を精度良く測定することが計画されている。

具体的には 2001 年に軌跡検出器の outer drift chamber (ODC) を BESS 測定器に増設し、さらに現在の軌跡検出器である inner drift chamber (IDC) と jet-type drift chamber (JET) を改良して、粒子の軌跡を精密に測定してより大きな運動量の粒子を測定可能にすることが計画されている。この改良により BESS で測定可能な粒子のエネルギー領域は、現在の 0.2 – 70 GeV から 0.2 – 500 GeV 以上にまで改善される予定である。

2001 年の軌跡検出器の増設・改良により、軌跡検出器の出力する信号数は現在の約 500 から倍の 1000 にまで増える予定である。一方、BESS では軌跡検出器の出力する信号をデータに変換するのに FADC (Flash Analog-to-Digital Converter) システムを用いているが、現在の約 500 の信号数はこの FADC システムが処理できる限界に近い。また FADC システムは大量の信号を処理するため大量の電力も消費しており、具体的には BESS 全体の消費電力 1200 W に対してその 1/4 の 300 W を FADC システムだけで消費している。このため単純に現在の FADC システムを 2 つに増やすだけでは、BESS 全体の消費電力を圧迫してしまう。BESS ではリチウム電池により電力を供給しているので消費電力の増加は即、飛行時間の削減すなわちデータ量の減少に繋がる。そこで現在次期 FADC システムの開発が続けられており、次期 FADC システムでは処理可能な信号数を増設するとともに低消費電力化され、2001 年の軌跡検出器の改良に対応する予定である。

1.2 BESS 測定器

BESS 測定器の構成を図 1.2 に示す。中心に配置されているのが jet-type drift chamber (JET) であり、その周囲に inner drift chamber (IDC) が配置されている。IDC の外側には超伝導コイル (COIL) が配置されており、この超伝導コイルは約 1 T の均

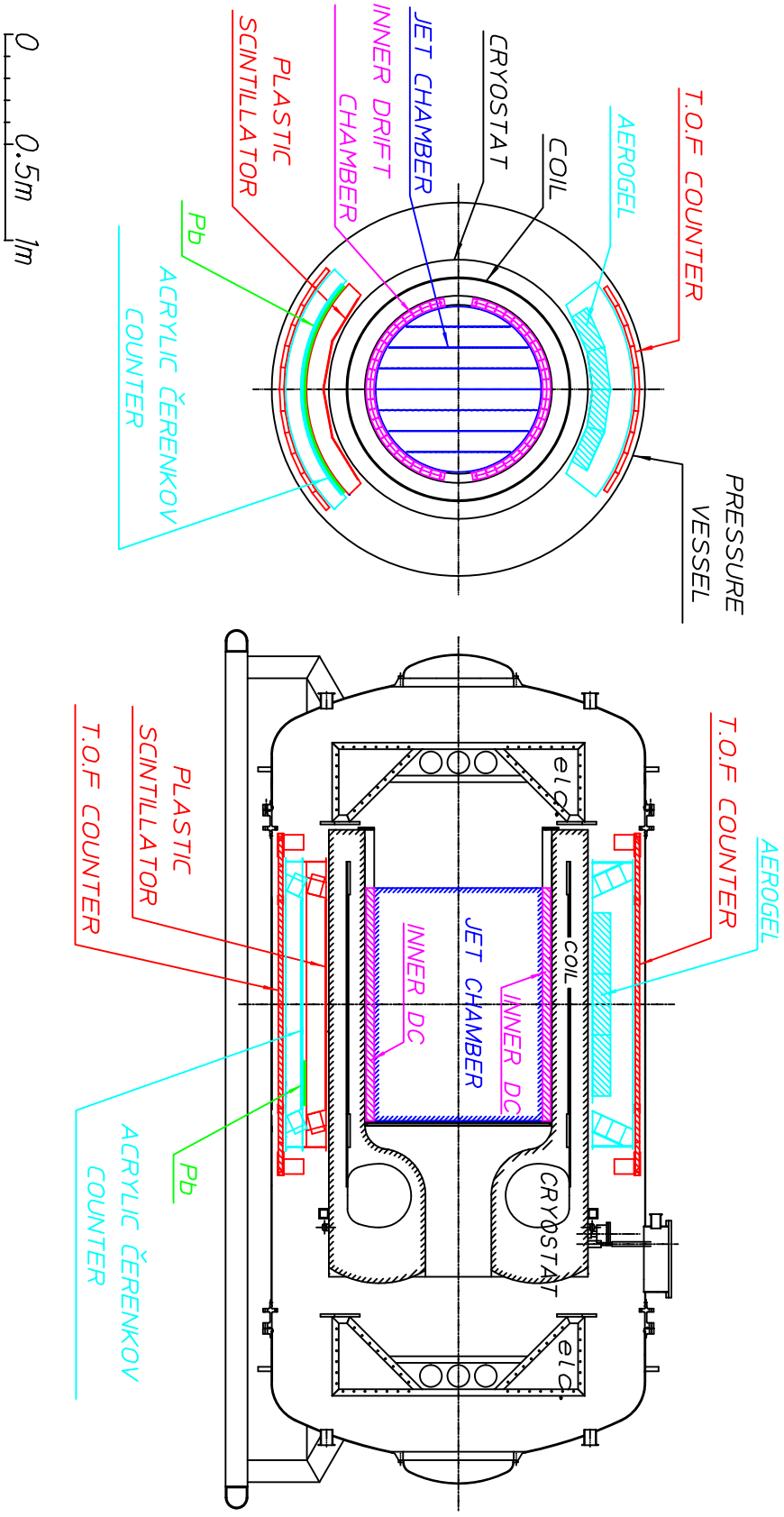


図 1.2: BESS 測定器 (1999 年)

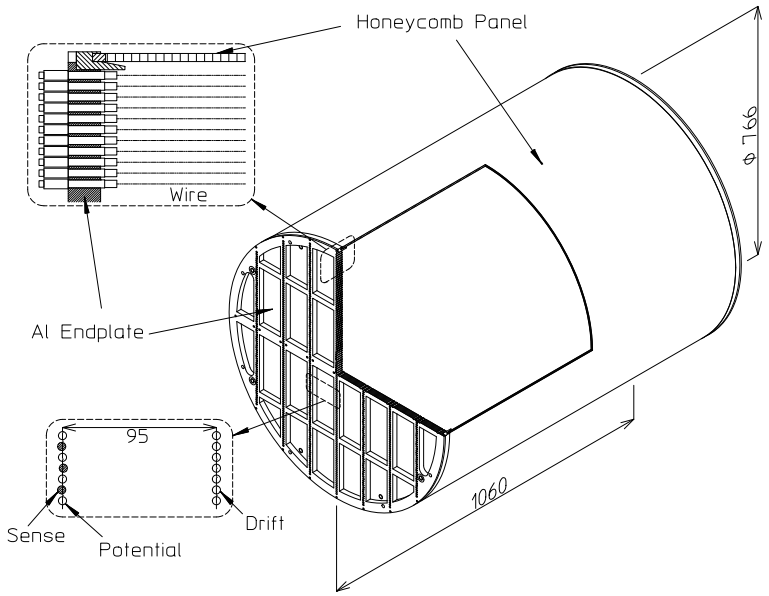


図 1.3: JET chamber

一な磁場を内側に発生させる。JET chamber はこの均一で強力な磁場の中で、通過する粒子の軌跡を測定する。IDC はセル (cell) 構造を持った円弧状のドリフトチェンバーで、粒子が通過した位置の検出を行う。超伝導コイルの上側には aerogel Čerenkov counter (AEROGEL) が配置され、コイルの下側には double decker detecter (DDD) が配置されている。AEROGEL は屈折率が約 1.02 のエアロジェルを輻射体として用いたチェレンコフカウンターで、運動量 5 GeV までの電子やμ 粒子のバックグラウンドから陽子や反陽子の判別を容易にする。DDD は二層構造の測定器で、上段にはプラスチックシンチレーションカウンターが配置され、下段にはアクリルチェレンコフカウンターが配置されていて、いずれも粒子が通過したときの光量を測定する。また二層の間の一部には鉛板が挟まれていて、シャワーカウンターとしての機能も合わせ持っている。さらにこれらの測定器を上下から挟む形で time of flight counter (TOF) が配置されている。TOF は上下に配置されたシンチレーションカウンターで、通過した粒子の飛行時間とエネルギーの損失を測定する。

以上の測定器の情報から入射粒子の電荷・質量・運動量を得ることができるので、BESS では原理的にはあらゆる荷電粒子の識別が可能である。入射した粒子のデータは、デジタル情報に変換されて大容量の磁気テープに保存される。

各測定器を制御する電子制御装置 (ELC) は測定器の両脇に配置されている。BESS 全体は圧力容器 (pressure vessel) で覆われており、圧力容器内は高度 35 km でも地上と同じ 1 気圧に保たれる。

1.2.1 JET chamber

JET chamber は直径が 0.766 m で長さが 1 m の、円筒形のドリフトチェンバーである。JET chamber の内部では両端のアルミニウムのエンドプレートの間にワイヤーが張られている。ワイヤーにはデータを読み出す sense ワイヤーと、電場を作

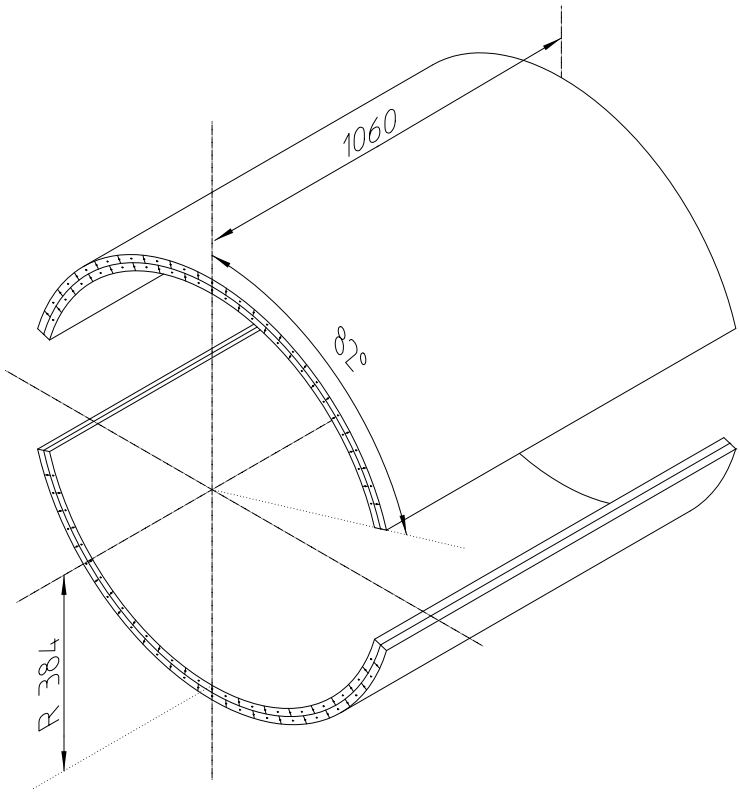


図 1.4: Inner Drift Chamber

る drift ワイヤーと potential ワイヤーの、3 種類がある。ワイヤーの配置は、sense ワイヤーと drift ワイヤーが交互に並んだ 4 枚の anode 平面の間に、potential ワイヤーが並んだ cathode 平面が 3 枚配置されている。各平面の中でワイヤーは 6 mm から 7 mm の間隔で並んでいる (以上、図 1.3 参照)。

JET chamber に入射した荷電粒子は周囲のガスをイオン化させながら通過する。荷電粒子の通過により発生した電子は電場の中で sense ワイヤーへ移動 (drift) し、sense ワイヤー近傍で電子雪崩を起こし増幅される。r-ϕ 方向の座標は drift 時間から求められ、z 軸方向の座標はワイヤーの両端での電荷の比から求められている。JET chamber の位置分解能は、r-ϕ 平面では平均約 200 μm で、z 軸方向については約 2.5 cm である。

1.2.2 Inner Drift chamber

Inner drift chamber (IDC) はセル構造を持った円弧状のドリフトチェンバーである。IDC は JET chamber のすぐ外側に位置し、超伝導磁石の内部に収まっている。IDC は半径が 0.768 m で長さが 1.06 m あり、鉛直上向きを 90°として左右に 8° から 127° の角度に展開している。上側に 2 層、下側に 2 層の計 4 層の構造になっていて、各層の厚さは 36 mm ある (以上、図 1.4 参照)。

IDC の分解能は、r-ϕ 平面で約 200 μm であり、z 軸方向については約 350 μm である。

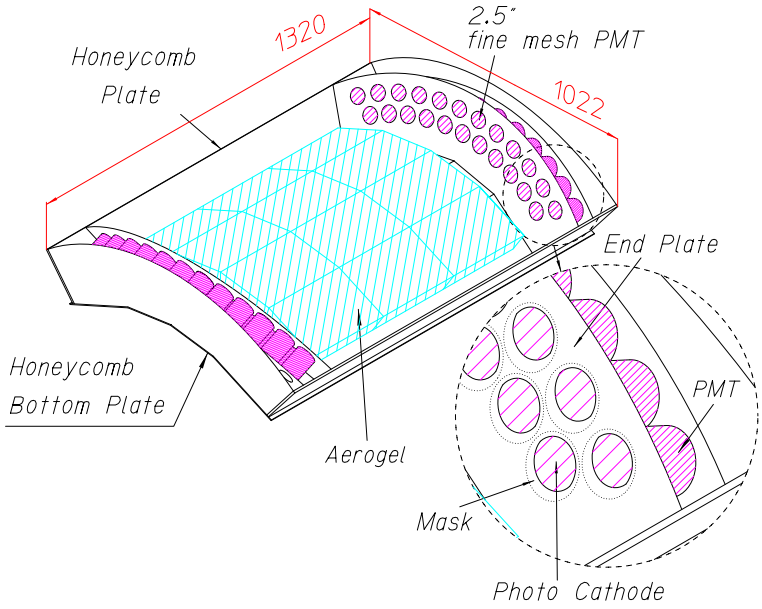


図 1.5: Aerogel Čerenkov counter

1.2.3 Aerogel Čerenkov counter

Aerogel Čerenkov counter (AEROGEL) は屈折率が約 1.02 のシリカエアロジェルを媒体に用いたチェレンコフカウンターである。大きさは長さが 1.32 m で幅が 1.022 m あり、内部には長さが 30 cm, 幅が 20 cm, 厚さが 2 cm のシリカエアロジェルが 4 枚重ねを 1 組として敷き詰められている。チェレンコフ光を計測する光電子増倍管 (PMT) は、合計 46 本が超電導磁石が作る磁場の向きを考慮して 24.5° の角度で両端に取り付けられている (図 1.5)。

AEROGEL は threshold 型のチェレンコフカウンターで、通過する粒子の速度の上限を特定することにより、粒子を識別するための情報を得ることができる。

1.2.4 Double Decker Detector

Double decker detector (DDD) は二層構造の測定器で、上段にはプラスチックシンチレーションカウンターが配置され、下段には屈折率が約 1.49 のアクリル樹脂を用いたチェレンコフカウンターが配置されている。上段と下段の両方とも粒子が通過したときの光量を、両端に取り付けられた光電子増倍管 (PMT) で測定する。また上段と下段の間の面積が約 1/5 の領域には厚さが 11.2 mm の鉛板が配置されていて、下段のチェレンコフカウンターと合わせることで DDD はシャワーカウンターとしても機能するようになっている (以上、図 1.6 参照)。

DDD は 1999 年度から BESS に搭載された新しい測定器で、従来 BESS が不得意であった high-charge 粒子の検出と 電子/μ 粒子の識別の向上を目的としている。

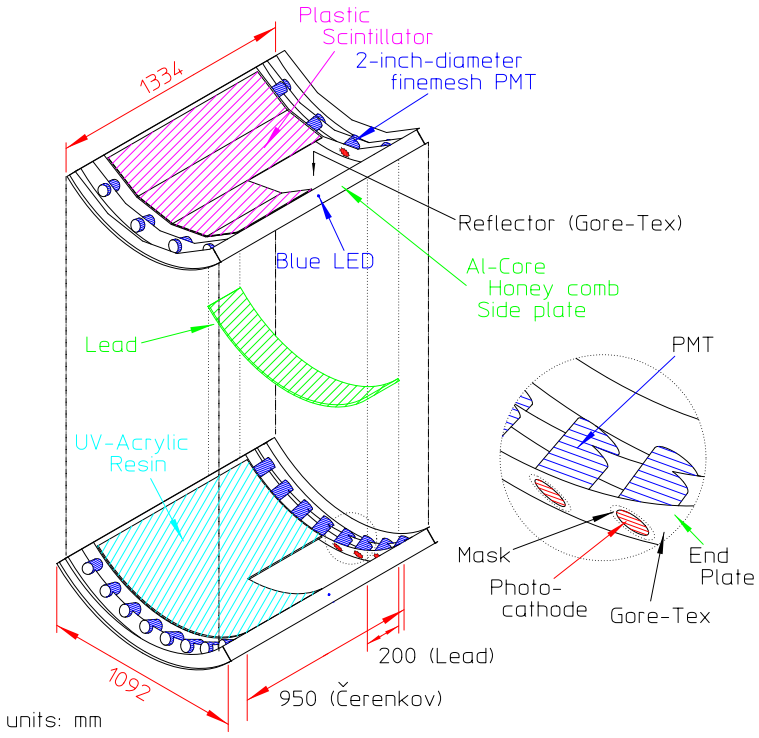


図 1.6: Double Decker Detector

1.2.5 Time of Flight counter

Time of flight counter (TOF) は内部の測定器を上下から挟む形で、BESS の耐圧容器 (pressure vessel) の内縁に沿ってプラスチックシンチレーションカウンターを配置したものである。個々のカウンターは、プラスチックシンチレータの大きさが長さ 950 cm, 幅 10 cm, 厚さ 2 cm あり、両端にはアクリルのライトガイドと光電子増倍管 (PMT) が取り付けられている (以上、図 1.7 参照)。このカウンターが BESS の上側には半径 0.804 m の円に沿って 10 個配置され、BESS の下側には半径 0.756 m の円に沿って 12 個配置されている。

TOF は粒子が上下のシンチレーションカウンターを通過する時間を測定する。また BESS では時間と同時に、粒子が通過したときにシンチレータが発っする光量についても測定しており、この光量により通過した粒子の dE/dx が計算されている。

図 1.7: Time of Flight counter

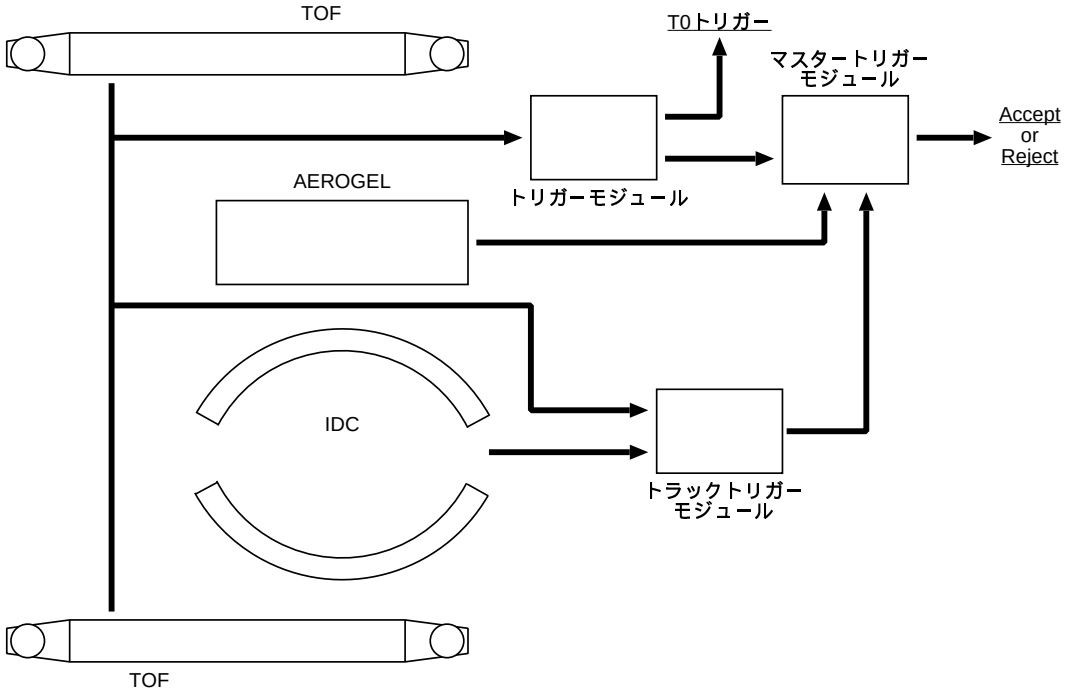


図 1.8: トリガーシステム

1.3 データ収集系

1.3.1 トリガーシステム

BESS 測定器ではトリガーシステムが粒子が入射されたことを検出して各測定器のデータ収集が始まる。トリガーシステムの概要を図 1.8 に示す。BESS のトリガーシステムは二段構えになっている。まず最初に上下の TOF カウンターの反応により、トリガーモジュールが first level トリガーである T0 トリガーを出力する。T0 トリガーの発生頻度は約 2 kHz である。T0 トリガーを受けると、BESS のデータ収集システムは各測定器のデータ収集を開始する。

T0 トリガーが発生すると、トラックトリガーモジュールは TOF と IDC の情報から入射粒子を受け入れるかどうかを判断して、その結果は約 10 μ second 後にマスタートリガーモジュールに渡される。マスタートリガーモジュールは、トラックトリガーモジュールの情報と AEROGEL の情報を使って、最終的に入射粒子を受け入れてデータを記録する (Accept) のかそれともデータを棄却する (Reject) のかを決定する。

マスタートリガーが入射粒子のデータを記録すると判断した場合は Accept 信号が発生し、T0 トリガーにより開始されたデータ収集の動作はそのまま続行される。逆に入射粒子のデータを棄却すると判断した場合は Reject 信号が発生し、データ収集の動作は直ちに中断されてシステムはクリアされ、約 40 μ second 後に BESS 測定器はトリガーが発生する前の初期状態に戻る。

1.3.2 データ収集システム

BESS に粒子が入射したときに各測定器が出力するアナログの信号は、FADC クレートと CAMAC クレートのモジュールにより適切なデジタル情報に変換され、さらにそれらのデジタル情報をイベントビルダー (event builder) が収集して 1 つの入射粒子のデータとしてまとめる。

JET chamber と IDC の出力するアナログの信号は、FADC クレート内の FADC システムによりデジタルの時刻と波高の情報に変換される。変換されたデータはそのままだと量が多過ぎて記録し切れないので、FADC システムは変換したデータをさらに圧縮してデータ量を削減する。JET chamber と IDC 以外の測定器の出力は、CAMAC クレートのモジュールによりデジタル情報に変換される。具体的には各検出器が信号を出力する時刻や出力された信号の電荷量を、CAMAC クレート内の TDC モジュールと ADC モジュールでデジタルの数値に変換する。CAMAC クレートと FADC クレートのモジュールによってデジタル情報に変換されたデータは、イベントビルダーに集められて一つの入射粒子のイベントデータとしてまとめられる。

イベントビルダーによって一つのイベントにまとめられたデータは、イベントフィルター (event filter) に渡される。イベントフィルターはトリガーシステムよりもさらに高度で柔軟なイベントの取捨選択の判断をオンラインで行う。イベントフィルターにより最終的に選択されたイベントデータが、磁気テープに記録され保存される。

1.3.3 新しい FADC システムの開発

気球実験である BESS では全ての測定器や電子機器の電力をリチウム電池で供給している。リチウム電池の重量は BESS 測定器全体の重量に対して無視できない割合を占めており、気球の飛行時間を制限する要因の一つとなるバラスト量や、到達高度やリスクに関係する使用気球の容積に大きな影響を与える。したがって消費電力の削減は非常に重要である。また BESS がデータを測定する大気上層では、外気の圧力が低いため圧力容器の外部への放熱は熱放射のみに制限され、また日中は太陽放射に晒される。したがって測定器や電子回路が正常に動作するように BESS の圧力容器の内部の温度を適切に保つためには、電力の消費を抑えて圧力容器の内部で発生する熱をできるだけ少なくしなければならない。

現在 BESS では測定器の高性能化とより長時間の飛行を目指して研究開発が進められている。そしてその研究開発の一環として消費電力の削減と処理できる信号数を増やすために、より少ない消費電力で多くの信号を処理できる FADC システムの開発が進められていて、最終段階にさしかかっている。JET chamber と IDC は BESS に入射した粒子の軌跡を判別するトラッキングシステムを構成し、非常に数多くの信号を出力する。このため JET chamber と IDC の出力する大量の信号を処理する FADC システムの消費電力はかなり大きく、BESS 全体が消費する電力のかなりの割合を占めている。また 2001 年に増設する予定の ODC により FADC システムの処理する信号の数がさらに増えるが、現在の FADC システムが処理し

ている信号の数は既にシステムの処理できる限界近くまで使い尽している。したがって FADC システムの拡張及び消費電力の削減は現在急務である。

FADC システムは主に複数の FADC モジュールにより構成されている。FADC モジュールは FADC クレートに設置する 1 枚の基板になっていて、この基板上に 16 個の FADC チャンネルが実装されている。FADC システムに入力された信号はこの FADC チャンネルによって変換される。FADC チャンネルは入力されたアナログ信号をデジタルの時刻と波高の情報に変換し、さらに波高が一定の閾値 (threshold) 以下のデータを削除する。FADC チャンネルによって変換されたデータは一台のデータ圧縮モジュールにより集められて波高情報を積分することにより圧縮され、イベントビルダーに送られる。

FADC システムは数多くの信号を処理するため FADC チャンネルの数も多くなり、その結果全ての FADC チャンネルが消費する電力は BESS 全体の消費電力に対して無視できない。具体的には BESS 全体の消費電力が約 1200 W であるのに対して、全 FADC チャンネルの消費電力は約 300 W である。BESS で使用している FADC モジュールは開発されてから既に 10 年近く経過しており、現在では当時よりも低消費電力で高性能な部品が入手可能になったので、新しく FADC モジュールを作り直すことで大幅な低消費電力化が見込めた。新しい FADC モジュールの消費電力は 1 チャンネル当りで現在の 1/2 から 1/3 程度になる予定である。

新しい FADC モジュールは現在のものと互換性を保つように作られているが、それでも幾つかの変更点がある。現在の FADC モジュールは個々の FADC チャンネルが threshold 以下のデータの棄却を行いデータ圧縮モジュールがデータを集めて圧縮を行っているが、新しい FADC モジュールでは個々の FADC チャンネルにデータ圧縮 IC が配置されていて、FADC チャンネルに入力された信号が変換されるのと並行してこのデータ圧縮 IC がデータの圧縮を行う。したがって新しい FADC モジュールでは独立したデータ圧縮モジュールは必要無く、また信号を変換するのと同時に圧縮してしまうので余分なデータ圧縮の時間も必要無くなる。このためデータの圧縮を行わずに FADC チャンネルの変換したデータを集めるモジュールが必要である。またデータ圧縮 IC は現在 BESS で使用しているデータ圧縮方法の他にも、新しいアルゴリズムを幾つか実装していて、この新しいアルゴリズムを利用するためにも新しい FADC モジュールと現在の BESS のデータ収集系を仲介する、FADC Interface (I/F) が必要である。

この論文では BESS の新しい FADC システムのために開発している、FADC I/F について述べる。この FADC I/F の最大の特徴は、FPGA (Field Programmable Gate Array) を採用したことである。FPGA とは内部の回路を何度でも書き換えることができる集積回路のことである。FADC I/F の回路の主要部分を FPGA の内部に収めることにより、ソフトウェア的な複雑な処理を実装しているにもかかわらず部品数を大幅に削減でき、またハードウェアでありながら回路の変更が容易になるというソフトウェアに近い柔軟性を備えることになった。なお、FADC チャンネルは信号の数に対応して大量に用意されているためその合計の消費電力は約 300 W にもなるが、FADC I/F はクレート毎に 1 台しか存在しないのでその消費電力はほとんど無視して構わない。このため FADC I/F では特に低消費電力化のための工夫は行っていないが、FPGA の使用により部品数が削減されたことと、FPGA は CMOS

の IC なので動作時以外は電力をほとんど消費しないことを考えると、FADC I/F の消費電力はそれほど大きくはならないと思われる。

第2章　FADCについて

2.1　FADCシステムに入力される信号

FADCシステムが扱う信号は、軌跡検出器であるJET chamberとinner drift chamber (IDC)の出力する信号である。JETとIDCはdrift chamberである。Drift chamberではガスで満たされた内部にワイヤーが張られていて、各ワイヤーには負の電位がかけられている。荷電粒子がchamber内部を通過すると周囲のガスを電離させ、電離した電子は各ワイヤーに集められる。ワイヤー近傍では $1/r$ (r はワイヤーからの距離)に比例して電場が強くなるので、集められた電子はなだれを起こして増幅されて、ワイヤーの出力する電気信号として観測される。荷電粒子の通過した位置から各ワイヤーまで電子が走る距離は異なるので、各ワイヤーはそれぞれ異なる時刻に信号を出力する。各ワイヤーが出力する時刻のパターンにより、通過した荷電粒子の軌跡を再構成することができる。

Drift chamberでは各ワイヤーが信号を出力する時刻を、正確に知ることが重要である。BESSではFADCシステムに負担を掛けないために、成分が CO_2 90 %, Ar 10 % のdrift速度が遅いガスを使用している。FADCシステムはJETとIDCの出力する信号を約30 nano second毎に8 bitの波高データに変換しており、この約30 nano secondの時間分解能で信号の立ち上がりを正確に捉えられるようになっている。信号の立ち上がりに要する時間は、FADCのデータにおいてせいぜい数サンプル(100 nano second以内)である。その後、信号は $\frac{1}{t_0+t}$ に比例して時間発展して行く。信号全体の幅は、典型的には10サンプル程度である(図2.1参照)。

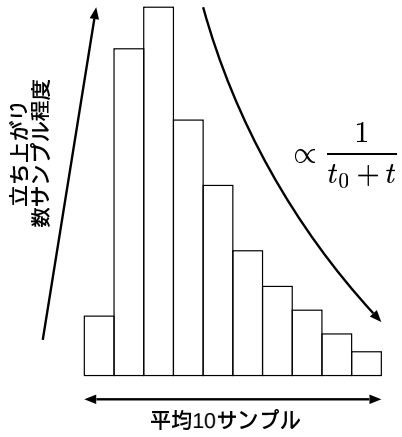


図 2.1: FADCで処理されたchamberの信号

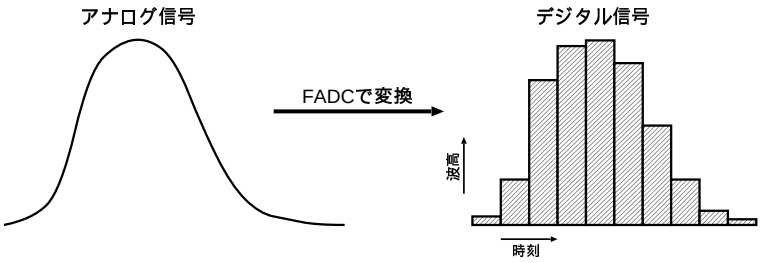


図 2.2: アナログ信号とデジタル信号

2.2 新しい FADC モジュールの概要

FADC とは Flash Analog-to-Digital Converter の略で、アナログ信号をデジタルの時刻と波高の情報に変換する (A/D 変換。図 2.2 参照)、一種の IC である。BESS ではこの FADC の IC に必要な回路を付けて chamber の出力する信号を処理できるようにしたひとまとまりの回路を FADC チャンネルと呼び、この FADC チャンネルを複数実装したモジュールを FADC モジュールと呼んでいる。現在開発中の新しい FADC モジュールの構造を図 2.3 に示す。FADC モジュールに入力された信号は、まずノイズを除去されて増幅されてから FADC に入力されてデジタル情報に変換され、その後データ圧縮 IC に入力されて圧縮によりデータ量を削減され、データ圧縮 IC 内部の RAM に保存される。開発中の FADC モジュールは、現在 BESS で使用している FADC モジュールと比較して、次の様な特徴を持っている。

- 消費電力が現在使用している FADC モジュールよりも半分以下に削減された。
- 実装密度が上がったことにより、同寸法のモジュールに今迄の 2 倍の 32 の FADC チャンネルが搭載されるようになった。
- 各 FADC チャンネル毎に専用 IC によるデータ圧縮回路を備えているので、FADC が A/D 変換を行うのと並行してデータを圧縮でき、データの圧縮に必要な時間が短縮された。
- 微調整が必要なアナログ回路をサブモジュールとして分離可能にし、メンテナンス性が向上した。

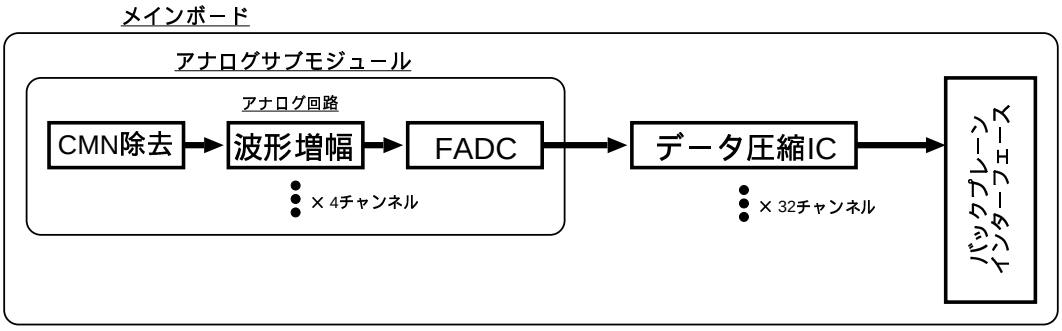
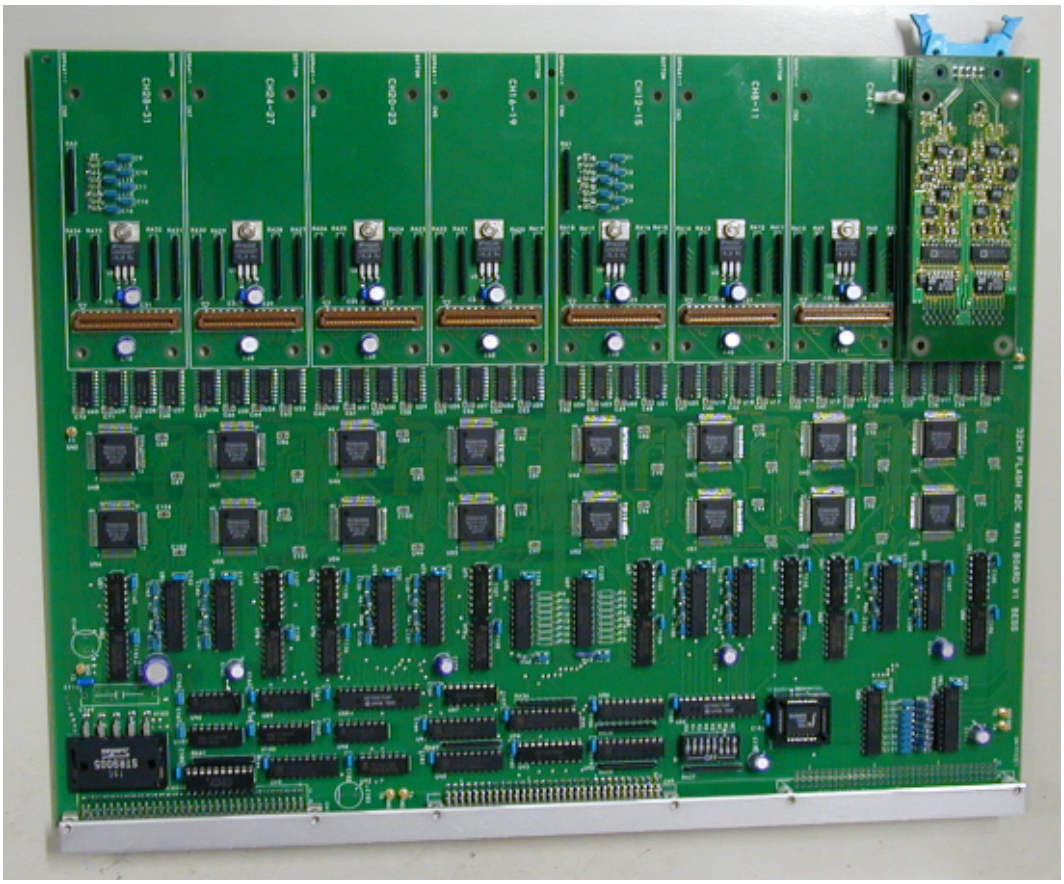
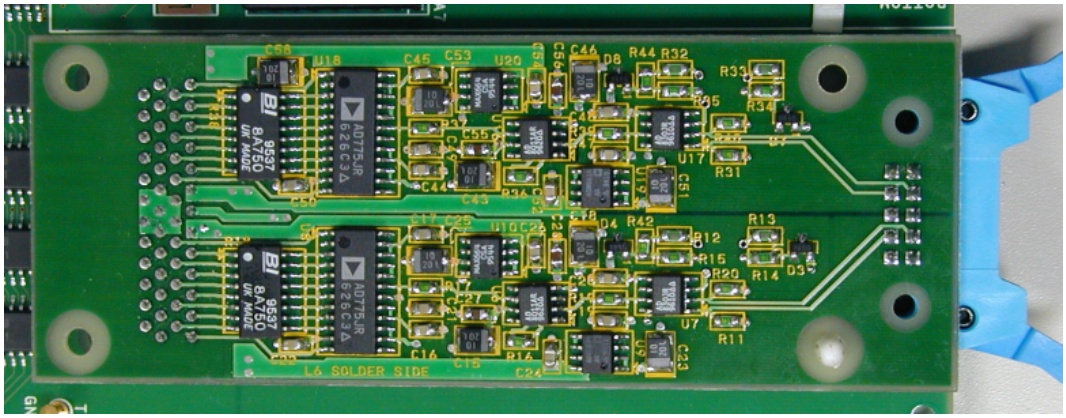


図 2.3: FADC モジュールの構成



(a) メインボード



(b) アナログサブモジュール

図 2.4: FADC モジュールの写真

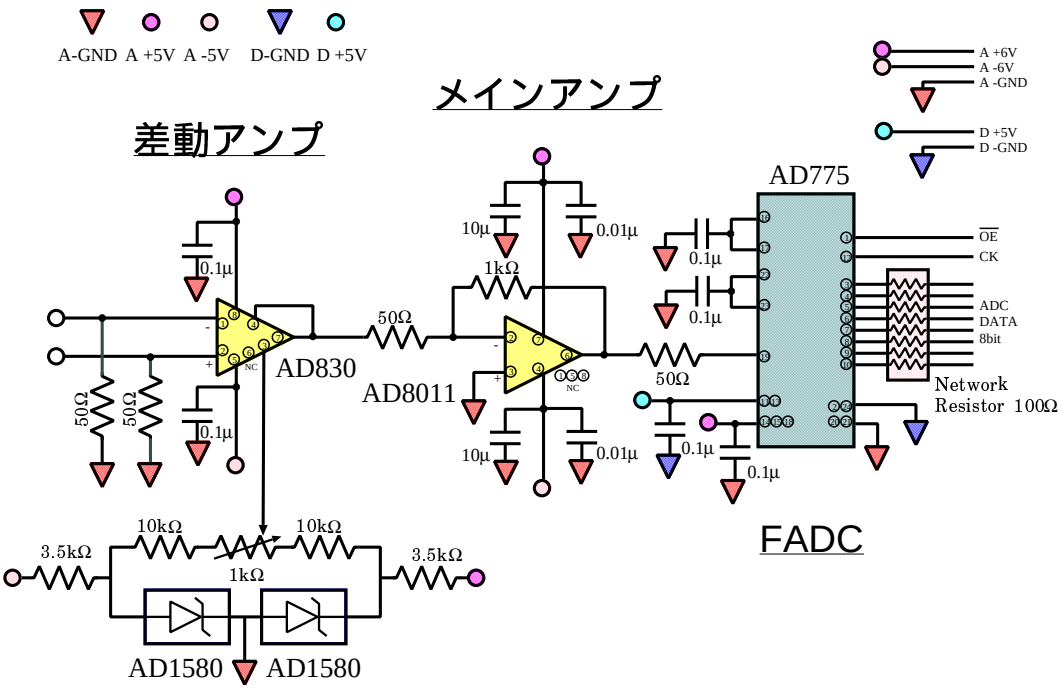


図 2.5: アナログ回路の回路図

新しい FADC モジュールでは現行のモジュールと同じく、高さが 9U (366.7 mm) で奥行きが 280 mm ある Eurocard 規格のメインボード上に、アナログサブモジュールを実装したサブボードを 8 枚搭載するようになっている。各アナログサブモジュールでは、4 チャンネル分のアナログ回路が実装されている。メインボード上には全部で 32 の FADC チャンネルのそれぞれに対応して 32 個の専用のデータ圧縮 IC が配置され、4 チャンネル毎にアナログサブモジュール上のアナログ回路とコネクタを通して接続されている。データ圧縮 IC は、アナログ回路上の FADC が A/D 変換を行うのと同期して FADC が出力するデータを圧縮し、データ圧縮 IC が内部に持つ RAM に記憶する。またメインボード上にはテストデータ用 FIFO が配置されていて、テストデータ用 FIFO に任意のテスト用データを書き込み、FADC が出力するデータの代わりにこのテストデータ用 FIFO のデータをデータ圧縮 IC に渡すこともできるようになっている。

2.3 アナログ回路

図 2.5 に現在開発中の次期 FADC モジュールのアナログ回路の回路図を示す。アナログ回路は主に次の 3 つの部分から構成されている。

- コモンモードノイズを除去する差動アンプ
- 信号の増幅が行われるメインアンプ
- 30 MHz のサンプリングレートで A/D 変換を行う FADC

以下、これらの部分について順番に説明する。

2.3.1 差動アンプ

JET chamber と IDC のプリアンプが出力した信号はツイストケーブルを通して、まず最初に差動アンプに入力される。差動アンプは、2 つの入力信号の差を増幅して出力するアンプである。差動アンプを使用する目的は、入力信号からコモンモード ノイズ (Common Mode Noise) を取り除いて信号成分だけを抽出することにある。コモンモード ノイズとは信号とグランドが同時に変化するノイズのことである。差動アンプでは 2 つの入力の差が出力されるので、2 つの入力が同時に変化するようなコモンモード ノイズは差動アンプを通すことで除去される。差動アンプはこのようにコモンモード ノイズの除去を目的としているので、ここでは信号の増幅はほとんど行われない。

現行の FADC モジュールでは、chamber 側のプリアンプのグランドと FADC モジュールのグランドとの間の意図的でない結合を避けるために、パルストランスを用いている。コモンモード ノイズに対してはこのパルストランスと、信号の増幅に使用している汎用の高速 OP アンプで構成した差動アンプを組み合わせて除去する設計になっている。しかしこのパルストランスは BESS が動作中に超伝導コイルの周辺に発生する 0.2 T という磁場中で使用するために、高磁場中で飽和してしまうような通常の強磁性体を芯材として使用できず、芯材には透磁率が低いけれども高磁場でも飽和しないカーボニル鉄を使用している。このためパルストランスはサイズが大きくてかさばり、また磁束が漏れるため回路がノイズを拾いやすい。メインアンプで構成した差動アンプもあまり効果的にコモンモード ノイズを除去できていなかった。

現行の FADC モジュールのパルストランスとメインアンプを組み合わせた回路では、グランドを完全に分離可能であるという長所はあるものの、以上で述べた欠点も色々ある。したがって開発中の新しい FADC モジュールでは、専用に設計された広帯域で消費電力の低い差動アンプを採用することにした。

2.3.2 メインアンプ

差動アンプでコモンモード ノイズを除去された信号は メインアンプで増幅される。ここで使用されるのは電流帰還型の OP アンプである。電流帰還型の OP アンプは電圧帰還型の OP アンプと比較して消費電力が少ないので、気球実験である BESS での使用に適している。ここで信号は 20 倍に増幅されて、次の FADC に入力される。電流帰還型の OP アンプは電圧帰還型と比べて直流特性を犠牲にしている面があり、特にオフセット電圧に個体差があるという欠点があるが、これは前段の差動アンプで各 FADC チャンネル毎に基準電位を可変抵抗で調節可能にすることで対処している。

電流帰還型の OP アンプの 2 つの入力端子は負入力が低インピーダンス、正入力が高インピーダンスになっており、通常は高インピーダンスの正入力側に信号を入力する非反転アンプとして使用することが推奨されている。しかし新しい FADC モジュールでは基準電位となるグランドに電流を流すことを嫌って、反転アンプとして使用して電流の流れない高インピーダンスの正入力側をグランドに接続し

型名	種類	電圧	電流 通常値 (最大値)	消費電力 通常値 (最大値)
AD830	Video Differential Amplifier	$+5$ V$-$$-5$ V	13.5 mA (16 mA)	135 mW (160 mW)
AD8011	Current Feedback Amplifier	$+5$ V$-$$-5$ V	1.0 mW (1.2 mW)	10 mW (12 mW)
AD775	Sampling A/D Converter	$+5.0$ V	9.5 mA (—)	47.5 mW (—)

MSPS: mega samples per second

表 2.1: アナログ回路の主な部品の消費電力 ([7], [8] の資料より抜粋)

ている。FADC チャンネルは 1 つのモジュール上に 32 チャンネル存在するので、グランドに大量の電流が流れることでグランドが不安定になる可能性を避けるために、このような設定で使用することにした。

2.3.3 FADC

FADC はアナログ信号をデジタルの波高情報に変換する IC であり、BESS では JET chamber や IDC が出力するアナログ信号をデジタル情報に変換するのに FADC を用いている。JET chamber では混合比が CO_2 90%, Ar 10%で drift 速度が約 7 mm/μ second の遅いガスを使用しているので、数十 MHz のサンプリングレートで十分な位置分解能が得られるようになっている。

FADC はトリガーが生じた直後に、512 サンプルのクロックが 30 MHz で入力されて動作する。クロックが入力される毎に、FADC は JET chamber や IDC の出力するアナログ信号を 8 bit のデジタル信号に変換して出力する。FADC が変換したデジタル信号は、メインボード上のデータ圧縮 IC に渡されて圧縮される。

2.4 メインボード

メインボード上にはテストデータ用 FIFO と各チャンネル毎にデータ圧縮 IC が配置されている。データ圧縮 IC は 4 チャンネル毎にアナログサブモジュール上のアナログ回路とコネクタを通じて接続されている。また FADC モジュールを収めるクレートのバックプレーンと接続するデータ圧縮 IC を選択するための回路も、メインボード上に配線されている。この回路によって、メインボード上の各データ圧縮 IC 毎のデータの読み書きや設定ができるようになっている。

メインボードとアナログサブモジュールが分離されているのは、故障しやすいアナログ回路を FADC モジュールから分離可能にしてメンテナンス性の向上を図るためと、消費電力の主な部分を占めるアナログ回路の部品を、将来さらに低消費電力の部品が入手可能になったときに交換できるようにしておくためである。

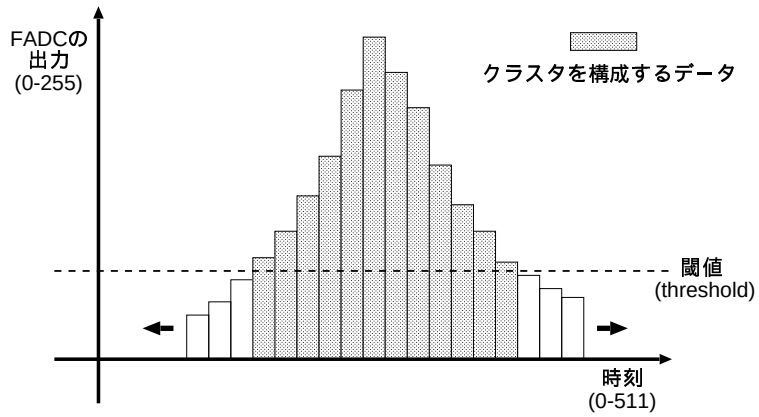


図 2.6: クラスタ

2.4.1 データ圧縮 IC

アナログ回路によりアナログからデジタルに変換された信号は、FADC モジュールのメインボード上で各チャンネル毎に配置されたデータ圧縮 IC に渡される。データ圧縮 IC は渡された信号のデータを、高速に圧縮して IC が内部に持つ RAM に保存する。IC 内部の RAM に保存されたデータは、IC の外部から読み出し信号を送ることでデータを非同期に読み出すことができ、全 FADC チャンネルのデータ圧縮処理が終了した後で、3 章で説明する FADC I/F により読み出される。このデータ圧縮 IC はカスタムメイドで製作された専用の ASIC (Application Specific Integrated Circuit) である。

データの圧縮モードは SUM, CRAW, HIST, RAW の 4 種類があり、それぞれデータの圧縮方法や出力されるデータの構造が異なる。各圧縮モードの詳細については、それぞれ 2.4.1.1 から 2.4.1.4 の各節で説明している。

SUM モードと CRAW モードでは、データ圧縮の処理は二段階で行われる。最初の段階では、8 bit で 512 サンプルの FADC データから threshold 以下のデータが捨てられる。この動作を zero suppress と呼ぶ。zero suppress の目的は、クラスタを構成するデータのみを抽出することである。クラスタとは、FADC の出力が threshold を越えてから再び threshold 以下に戻るまでの、一連のデータのことである (図 2.6 参照)。またデータ数が 1 個しかないクラスタは雑音であることが多いので、この段階で捨てられる。次の段階では、クラスタを構成する FADC のデータを各クラスタ毎にまとめて圧縮する。また SUM モードは BESS で主に使用される圧縮モードで、この SUM モードが出力するデータは、現在 BESS で使用されている FADC クレートのデータ圧縮モジュールが出力するデータと互換になっている。

HIST モードと RAW モードは先に説明した SUM モードと CRAW モードとは異なり、実はデータの圧縮は行わない。HIST モードはオンラインでの簡単なデータ解析を補助するためのモードであり、RAW モードは FADC のデータを加工せずにそのまま出力するデバッグ用のモードである。

データ圧縮 IC は外部から参照できる MODE, CHANNEL, THRESHOLD の 3 つのレジスタを持っている。これらのレジスタの内容によってデータ圧縮 IC の動作が決まり、3 章で説明する FADC I/F はこれらのレジスタの内容を設定することが

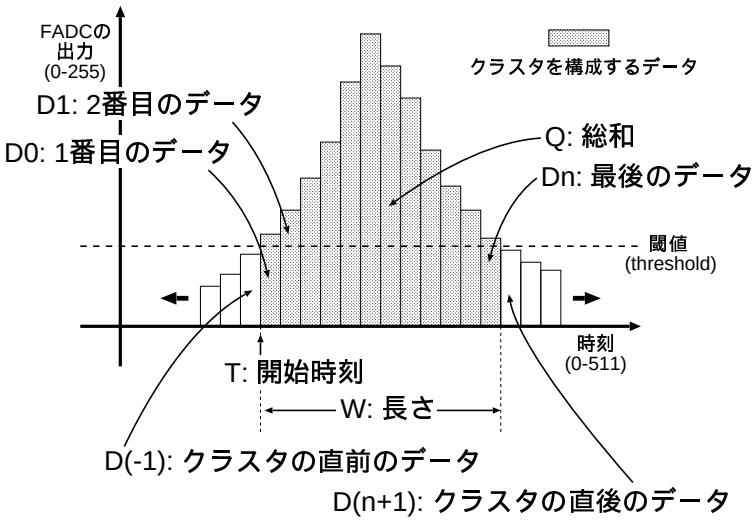


図 2.7: SUM モードと CRAW モードで記録されるクラスタのデータ

できる。MODE レジスタはデータ圧縮のモードや、外部から参照・設定できるデータ (CHANNEL レジスタ, THRESHOLD レジスタ, 内部 RAM) の種類などを決定する 7 bit の値を設定する。CHANNEL レジスタは、SUM モードと CRAW モードの出力するデータに付加する 8 bit のチャンネル番号を設定する。THRESHOLD レジスタは zero suppress 処理で使用する 8 bit の threshold の値を設定する。

データ圧縮 IC はトリガー発生直後に 512 サンプルのクロックが 30 MHz で入力されるときしか動作せず、また CMOS の IC なので動作時以外は電力をほとんど消費しない。よって FADC モジュール全体におけるデータ圧縮 IC の消費電力の割合は無視できる程度である。

2.4.1.1 SUM モード

SUM モードはクラスタの各データの総和を取る (つまり積分する) ことにより圧縮を行う。SUM モードで記録されるデータは次の通りである (図 2.7 参照)。

Q クラスタの総和

T クラスタの開始時刻

D0 クラスタの 1 番目のデータ

D1 クラスタの 2 番目のデータ

C FADC のチャンネル番号

W クラスタの長さ

O オーバーフロー

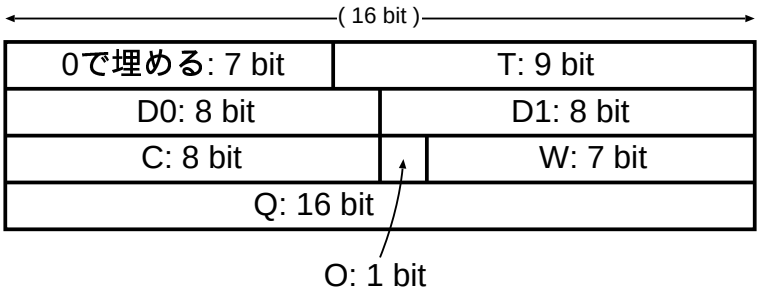


図 2.8: SUM モードで出力されるデータの構造

オーバーフロー (O) はクラスタの長さ (W) が 128 以上のときか、またはクラスタを構成するデータの内少なくとも 1 つのデータの値が 255 であったときに 1 の値をとり、そうでなければ 0 の値をとる。このオーバーフローはデータ圧縮 IC の OF ピンの出力とは別物であることに注意する必要がある。

SUM モードでデータ圧縮 IC の RAM に書き込まれるデータの構造を図 2.8 に示す。1 クラスタのデータの大きさは $2\ \text{byte} \times 4 = 8\ \text{byte}$ の固定長である。ここで 1 byte は 8 bit のことである。データ圧縮 IC が内部に持つ RAM の容量は 512 byte なので、1 回のトリガーで 64 個よりも多くのクラスタが発生すると、65 個目以降のクラスタを RAM に記録することができない。このときはデータ圧縮 IC の OF ピンに High が出力され、全てのクラスタを記録し切れなかったことを外部に知らせる。

SUM モードは BESS が動作中に主に使われる圧縮モードである。SUM モードでは 1 クラスタのデータが常に 8 byte の固定長で出力されるので、元の $8\ \text{bit}\ (1\ \text{byte}) \times 512$ sample の生データにおいてクラスタの数が 64 個よりも少なければ、圧縮後のデータサイズは圧縮前よりも少なくなる。実際には 1 トリガーにおける FADC 1 チャンネル当りの平均的なクラスタ数は 0.5 個で、1 チャンネルに 1 個のクラスタがあるかどうかという程度であり、1 クラスタの典型的なサンプル数は約 10 サンプルである。ただしクラスタのデータにはクラスタの開始時刻 (T), クラスタの流さ (W), FADC のチャンネル番号 (C) が必要なので、付加的な情報が 3 byte 必要である。したがって 1 クラスタのデータは平均して 13 byte になり、それが圧縮後は 8 byte の固定長データになるので、約 60% の圧縮率になる。

2.4.1.2 CRAW モード

CRAW モードでは FADC の出力からクラスタを構成するデータのみが取り出される。CRAW モードで記録されるデータは次の通りである (図 2.7 参照)。

T クラスタの開始時刻

C チャンネル番号

W 圧縮データのワード数

D(-1) クラスタの 1 つ前のデータ

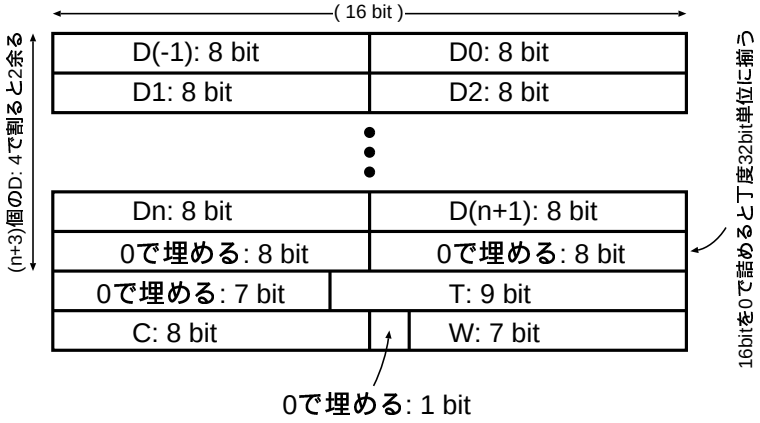


図 2.9: CRAW モードで出力されるデータの構造

D0$\cdots$**Dn** クラスタを形成するデータ

D(n+1) クラスタの 1 つ後のデータ

CRAW モードでデータ圧縮 IC の RAM に書き込まれるデータの構造を図 2.9 に示す。CRAW モードのクラスタのデータは可変長であり、クラスタの数が多過ぎるかまたは最後のクラスタのデータ量が大き過ぎて RAM に収まり切らない場合は、RAM に正常に収められる範囲のクラスタまで RAM に記録され、OF ピンに High が出力される。クラスタのデータ Di $(i = -1, 0, \cdots, n, n+1)$ は 32 bit 単位に揃えられ、余った領域には 0 が詰められる。クラスタのデータが丁度 32 bit 単位に収まるときは、クラスタデータの最後に 32 bit の 0 が追加される。これはデータ圧縮 IC の内部で 32 bit 単位でデータを処理していることと、0 を詰めた領域を検索することで後でクラスタデータの終端を検出可能にするために、このような仕様になっている。また CRAW モードでデータのワード数を記録する領域 (W) の大きさは 7 bit しかないので、クラスタのデータ数が 128 を越えたときは記録し切れないが、クラスタのデータの終端を検出することが可能なのでデータを読み出すことは可能である。

2.4.1.3 HIST モード

HIST モードでは、FADC データからヒストグラムを作成してデータ圧縮 IC の RAM に記録する。データ圧縮 IC は通常、一回の動作で 512 サンプルの FADC データを取り込むが、HIST モードではヒストグラムの計算に 4 clock 必要なため、512 サンプルの約 $1/4$ の 125 サンプルのデータを使ってヒストグラムを生成する。値が 127 以上の FADC データは同じ値とみなされ、ひとまとめにして数えられる。HIST モードでデータ圧縮 IC の RAM に生成されるヒストグラムのデータ構造を図 2.10 に示す。

HIST モードは何度も繰り返すことでヒストグラムのサンプル数を増やすことが可能であるが、サンプルの個数は 16 bit で記録されるので、$2^{16}/125 \gtrsim 524$ 回以上

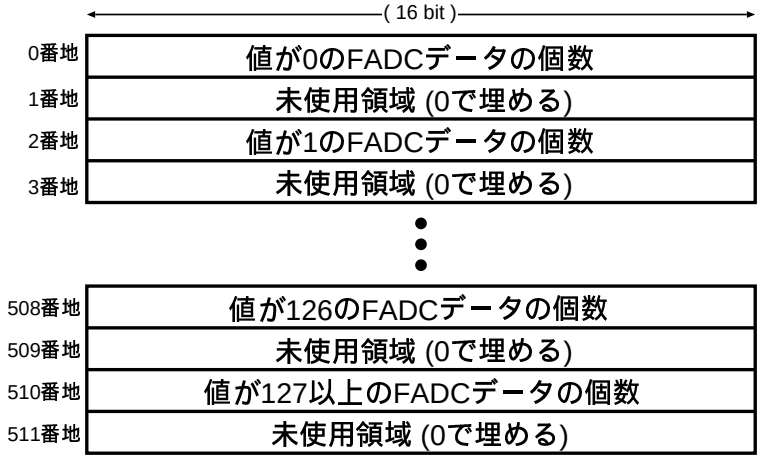


図 2.10: HIST モードで出力されるヒストグラムの構造

繰り返すことはできない。また HIST モードを使うときは、あらかじめデータ圧縮 IC の RAM を外部からの書き込みにより 0 で埋めておく必要がある。

HIST モードは BESS が動作中に定期的に実行される、FADC の pedestal スキャンを補助するためのモードである。このためデータの記録を目的としていないので、HIST モードの出力するデータにはチャンネル番号は付加されない。

2.4.1.4 RAW モード

RAW モードは 1 回の動作で得られる 512 サンプルの FADC データを、そのままデータ圧縮 IC の RAM に記録するモードである。RAW モードの出力するデータは素朴に考えると非常に単純であるように思えるが、実際には図 2.11 を見ると分かるように、少々ややこしいデータ構造になっている。これは ASIC のゲート数を削

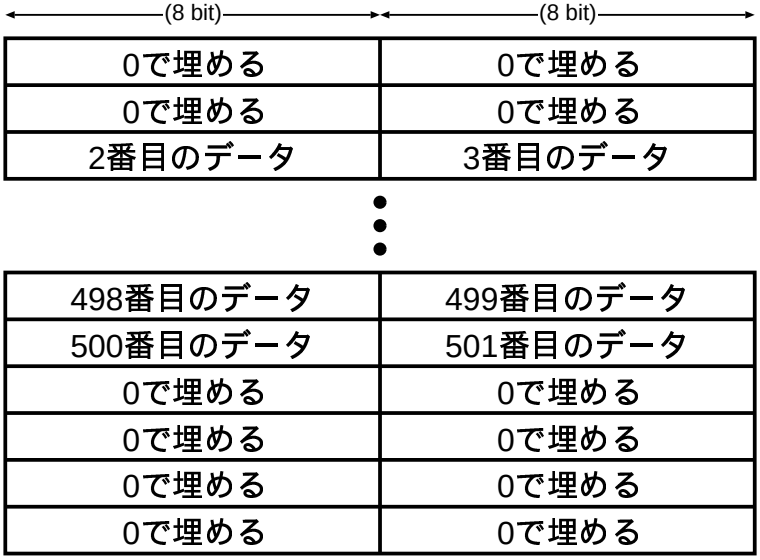


図 2.11: RAW モードで出力されるデータの構造

減するために回路を単純化したことで、RAW モードにデータ圧縮 IC 内部の動作がそのまま反映されているためである。以下、この理由を簡単に説明する。

データ圧縮 IC は SUM モードと CRAW モードにおいて、順番に入力される 512 サンプルのデータからクラスタを見つけて圧縮しなければならない。512 サンプルのデータ中でクラスタが現れ得るパターンは、次の 3 通りである。

1. 512 サンプルのデータを取り始めた時点で、既にクラスタが始まっている場合。この場合、0 番目のデータは既に threshold を越えている。
2. クラスタが 512 サンプルのデータの中に収まっている場合。この場合、データは threshold 以下の値から threshold 以上の値になり、しばらくてまた threshold 以下の値になる。
3. クラスタが 512 サンプルのデータの最後からはみ出ている場合。この場合、512 サンプルの最後のデータは threshold を越えている。

この 3 つのパターンのクラスタの中で、本当に記録したいのは 2 番目のパターンのクラスタである。なぜなら FADC モジュールに入力される chamber の信号は 512 サンプルの中にうまく収まるようにタイミングを調節されているので、512 サンプルの端に位置する 1 番目と 3 番目のパターンのクラスタは明らかにノイズだからである。

このため 3 つのパターンのクラスタを全て判別する回路を設計して ASIC のゲート数の増加を招くよりは、2 番目のパターンのクラスタだけを圧縮する単純な回路で設計した方が実装上の観点からは望ましかった。ASIC のゲート数の増加はそのまま値段に反映されるので、ゲート数の削減は切実な問題である。最も簡単な解決方法は、512 サンプルのデータの内、最初のデータと最後のデータを強制的に 0 として扱うことである。こうすることで 1 番目と 3 番目のパターンのクラスタを、内部のクラスタを処理する回路にはあたかも 2 番目のパターンのクラスタであるかのように見せかけることができ、クラスタを処理する回路は常に 2 番目のパターンのクラスタだけを処理すれば良いので回路を単純化できる。

実際には 512 サンプルのデータの内、最初の 2 つのデータと最後の 8 つのデータが強制的に 0 として扱われている。最初の 2 つのデータが 0 なのは、1 番目のクロックでトリガーの入力を確認して、2 番目のクロックで RAM の初期化を行うからである。RAM の初期化では、データを書き込むアドレスのレジスタを初期化する。最後の 8 つのデータが 0 なのは、データ圧縮 IC の圧縮回路はパイプラインになっているので、パイプラインの全ての回路を終了させるにはそれだけのクロックが必要だからである。

RAW モードでは、以上で説明したデータを加工せずにそのまま RAM に書き込むので、RAM の最初と最後には 0 が書き込まれて図 2.11 の様なデータ構造になっている。512 サンプルのデータをそのまま RAM に記録しないのは、データ圧縮 IC には 512 サンプルのデータの先頭と最後を強制的に 0 として扱う回路が必要でありそれは既に組み込まれているので、この回路を迂回して RAM にデータを書き込むのはゲート数の増加を招くので望ましくなく、またデータ圧縮 IC の動作の確認という意味では他の動作モードに入力されるデータを出力できる方が望ましかったからである。

図 2.11 のデータ構造を注意して見ると、RAM の最後は確かに 8 サンプル分のデータに 0 が書き込まれているが、RAM の最初は 2 サンプルではなく 4 サンプル分のデータに 0 が書き込まれている。これはデータ圧縮 IC の内部の RAM へのデータの書き込みが 32 bit 単位になっていることが原因である。512 サンプルのデータを全て記録しようとすると、RAM の書き込みが 32 bit 単位であるため 4 クロック毎に RAM へデータを書き込まねばならない。しかしクロックが入力され始めてトリガーの確認と RAM の初期化を終えてデータが取れる状態になった 3 番目のクロックの時点では、まだ RAM へデータを記録する準備ができておらず、この時点で無理をしてデータを RAM に書き込む回路を設計してゲート数の増加を招くよりは、単純にデータを捨てて最初の 4 サンプル分のデータを 0 で埋めてしまうほうが、回路が単純化されて望ましかったからである。

最後に注意しなければならないのが、RAW モードが出力するデータの配置と、SUM モードと CRAW モードでクラスタ毎に記録される時刻 T の関係である。データ圧縮 IC に入力される最初の 2 つのデータは強制的に 0 と見なされ捨てられてしまうので、時刻 T の基準となる位置は実は最初から 3 つ目のデータが入力された時刻のことであり、この時点で時刻 T は 0 に初期化される。さらに RAW モードでは最初から 3 つ目と 4 つ目のデータは捨てられるので、時刻 T が 2 のデータから記録し始めることになる。つまり RAW モードでは、時刻 T が 2 から 501 までのデータを RAM の図 2.11 の位置に記録するのである。

2.4.2　テストデータ用 FIFO

メインボード上には、512 個の 8 bit データを格納できるテストデータ用 FIFO が配置されている。テストデータ用 FIFO の役割は、データ圧縮 IC にテスト用のデータを入力して、データ圧縮 IC の動作を確認することである。

データ圧縮 IC には 8 bit のデータ入力用ピンが 2 セットあり、一方はアナログサブモジュール上にあるアナログ回路の FADC の出力に接続されていて、もう一方はテストデータ用 FIFO の出力に接続されている。データ圧縮 IC は MODE レジスタの設定により、FADC の出力とテストデータ用 FIFO の出力のどちらか一方を、データ圧縮 IC の入力データとして使用することができるようになっている。データ圧縮 IC は通常、FADC の出力を圧縮するように設定されている。データ圧縮 IC の動作を確認するときは、テストデータ用 FIFO に適当なデータを設定してからデータ圧縮 IC の入力をテストデータ用 FIFO に切り替えて、データ圧縮 IC にテストデータ用 FIFO のデータを圧縮させてデータ圧縮 IC の出力が適切かどうかを確認する。データ圧縮 IC がデータを圧縮する 4 つのアルゴリズムは、それぞれ 2.4.1.1 から 2.4.1.4 の各節で説明した通りなので、データ圧縮 IC に入力されるデータがあらかじめ分かっていれば計算することができる。したがってテストデータ用 FIFO に設定したデータに対して、圧縮後のデータを計算してデータ圧縮 IC の出力と比較することにより、データ圧縮 IC が正常に圧縮を行っているかどうかを確認することができる。

第3章　FADC I/Fについて

3.1　FADC I/Fの概要

FADC I/Fは最大32個のFADCモジュールを制御するためのモジュールである。各FADCモジュールには32個のFADCチャンネルが実装されており、FADC I/FはFADCモジュール上の各FADCチャンネルに対して個別に制御することが可能である。したがって見方を変えれば、FADC I/Fは最大で $32 \times 32 = 1024$ 個のFADCチャンネルを制御可能であるとも言える。

FADC I/Fの主な仕事は、BESSが動作中にトリガーが発生したら、FADCモジュールを動作させてJET chamberの信号をデジタル情報に変換し、FADCモジュールがデータの変換を終えたら、FADCモジュールから変換されたデータを全て読み出してイベントビルダーへ送ることである。トリガーが発生したらFADC I/Fは、30 MHzで512サンプルのクロックをFADCモジュール上の各FADCチャンネルに配置されたFADCとデータ圧縮ICに送って動作させて、JET chamberから各FADCチャンネルに入力される信号をデジタル情報に変換して圧縮する。アナログ信号からデジタル情報に変換されて圧縮されたデータは、データ圧縮ICが内部に持つRAMに保存される。全てのFADCチャンネルが信号の変換と圧縮を終えたら、FADC I/Fは全FADCチャンネルのデータ圧縮IC内部のRAMに保存されたデータを、順番に読み出してデータ出力用FIFOに出力する。データ出力用FIFOはFADC I/F上に配置されたFIFOで、データの取り出し口はフラットケーブルでイベントビルダーに接続されており、データ出力用FIFOに書き込まれたデータはイベントビルダーが自動的に読み出して処理する。

FADC I/FはFADCモジュールを動作させて出力されたデータをイベントビルダーへ送るという主な仕事の他にも、FADCモジュール上の各FADCチャンネルの各種設定などを必要に応じて行える必要がある。以上から、FADC I/Fに必要とされている機能を簡単にまとめる。

- **非同期に入力されるT0トリガーに反応してFADCモジュールを適切に動作させる機能。**

  気球実験であるBESSでは電池を使用しているため電源の容量に限りがあるので、FADCモジュールを常に動作させて電力を消費するのは望ましくない。そこでトリガーが発生してJET chamberの信号を処理する必要が生じたときにだけ、FADCモジュールを動作させる必要がある。

  またT0トリガー発生後にMasterトリガーでイベントが棄却されてReject信号が発生した場合は、FADC I/FはFADCモジュールの駆動やデータの読み出しを中断して待機状態に戻らなければならない。そして関連したモジュー

ルに Fast Clear 信号を発行してイベントが棄却されたことを伝えなければならない。

- **複数の FADC チャンネルのデータを連続して読み出す機能。**

  トリガーが発生して FADC モジュールが JET chamber の信号を処理した後、FADC I/F は全ての FADC チャンネルのデータを読み出して、イベントビルダーへ読み出したデータを渡す必要がある。イベントビルダーへのデータの引き渡しはデータ出力用 FIFO を通じて行われるので、FADC I/F は読み出したデータをデータ出力用 FIFO へ出力すればよい。またこのとき FADC の出力するデータに仕様上の不都合な部分があるので、この不都合な部分を修正する必要がある。

- **各 FADC チャンネルの設定を行う機能。**

  FADC I/F は個々の FADC チャンネルの各種設定ができなければならない。FADC チャンネルの各種設定とは、例えばデータ圧縮 IC の各レジスタの内容やテストデータ用 FIFO のデータの設定などである。また FADC チャンネルを設定するための命令や情報などは、FADC I/F の外部から送られて来るため、FADC I/F は外部から送られて来る命令や情報を受け取ってを判別し、実行することができなければならない。

3.2 FADC I/F の実装について

FADC モジュールと FADC I/F は、高さが 9U (366.7 mm) で奥行きが 280 mm ある Eurocard 規格のボード上に実装されている。FADC モジュールと FADC I/F を収めるクレートは 21 のスロットを持ち、1 枚の FADC I/F と最大 20 枚の FADC モジュールを収納できるようになっている。FADC I/F はトリガー時に FADC データを収集する役割の他に、各 FADC モジュールやそのモジュール上の各 FADC チャンネルを制御するクレートコントローラの役割も担っている。クレートの裏側には、クレートのスロットに収められた各モジュールを接続する 2 枚のバックプレーンが取り付けられている。2 枚のバックプレーンは下から順にそれぞれ DIN41612 型の 64 ピンと 96 ピンのものが使用されている。

図 3.1 に FADC モジュールと FADC I/F の内部構造のブロック図を示す。FADC モジュールの内部構造については、2 章で述べたのでここでは触れない。FADC I/F の内部構造に注目すると、中央の FPGA という部品に FADC I/F の入出力がすべて集まっていることが分かる。次の 3.3.1 節で詳しく述べるが FPGA とは内部の配線を変更可能な IC のことであり、FADC I/F の複雑な動作を制御・実行する回路はすべてこの FPGA の内部で実装されている。図 3.2 に開発中の FADC I/F の基板の写真を示す。図 3.2(a) の写真の基板表面の中央に配置されている正方形の大きな IC が FPGA である。FPGA の外部に配置されている部品はすべてそれ自身で複雑なロジックを実行するようなものではなく、FADC I/F 外部との入出力を補助する部品や、クロック生成回路のように FPGA では実現できない動作を行なう部品などの、単純なものばかりである。基板上には実質的に FPGA が 1 個載っているだけ

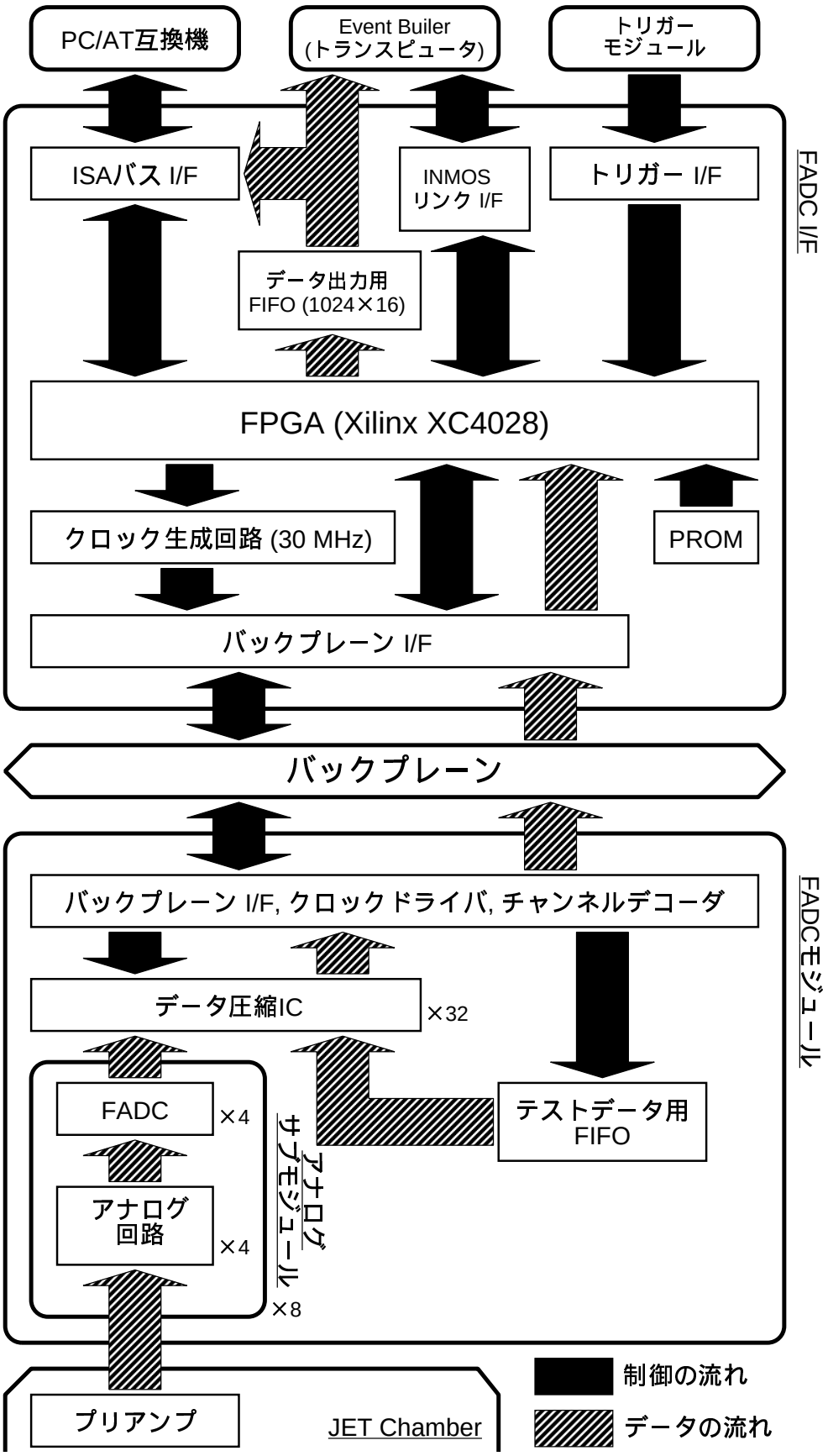
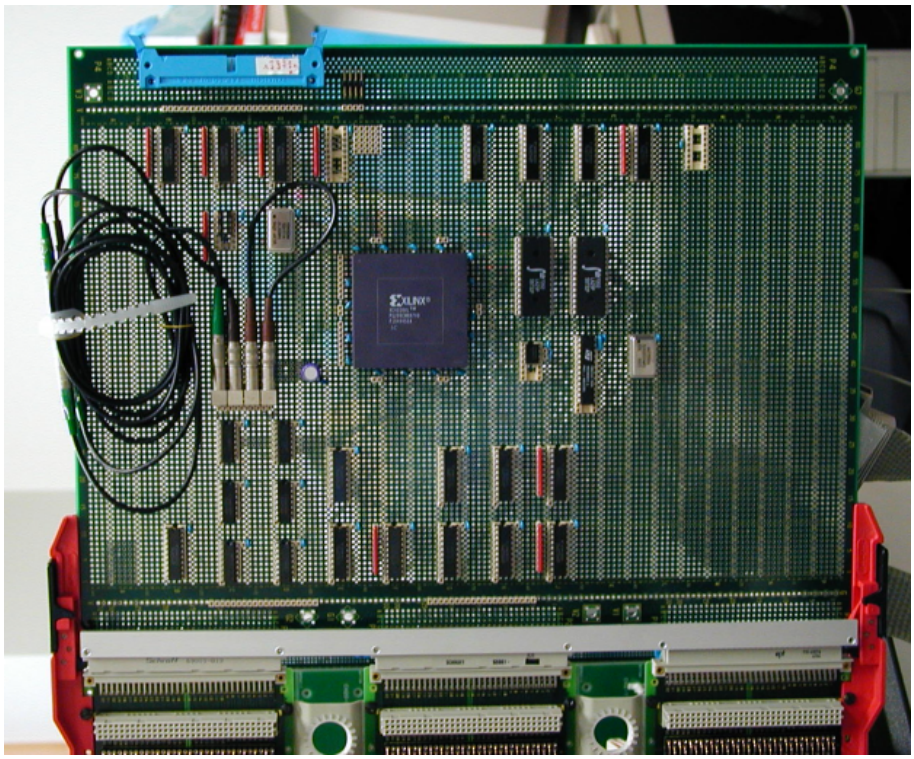
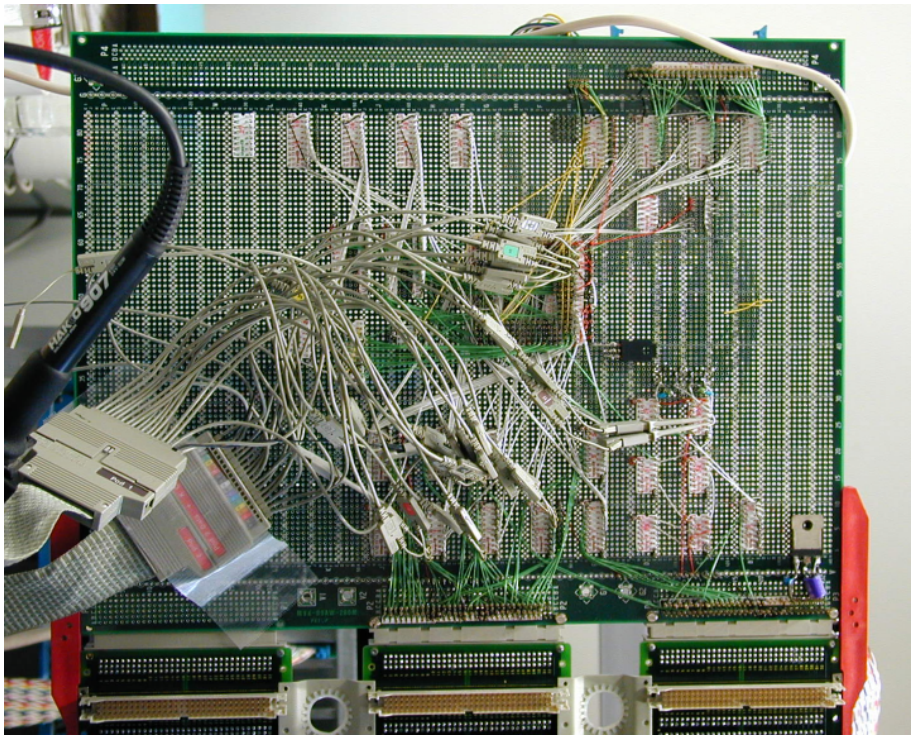


図 3.1: FADC モジュールと FADC I/F の内部構造のブロック図



(a) 基板の表



(b) 基板の裏 (デバッグ中)

図 3.2: 開発中の FADC I/F の写真

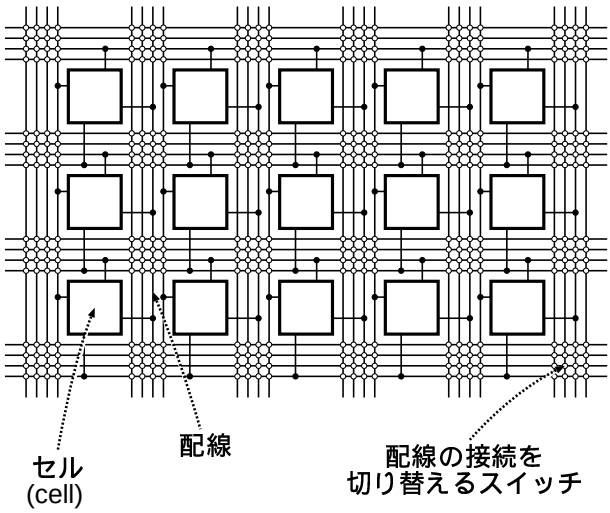


図 3.3: FPGA の構造

であり、クロック生成回路以外の全ての電子回路は FPGA の中に収まっていることが分かる。

3.3 開発方法について

3.3.1 FPGA

FADC I/F の開発では、FPGA を使うことにした。FPGA とは Field Programmable Gate Array の略で、内部の配線を何度でも書き換えることのできる集積回路のことである。集積回路は通常、マスクと呼ばれる一種のフィルムに回路を描き、それをシリコンの基板に感光させて複写することによって作られる。大量に生産される集積回路ならばこの方法で効率良く安価に作れるが、開発中の集積回路だと仕様の変更やバグの修正などで頻繁に回路の修正が必要であり、一度マスクを作る方式ではかえって効率が悪い。そこで内部の配線を何度でも書き換え可能な FPGA が考案された。FPGA ではセル (cell) と呼ばれる小規模な集積回路を図 3.3 の様にあらかじめ格子状に配置し、電気的なスイッチで配線やセルの機能を変更できるようになっている。集積回路の開発段階では FPGA が使われることが多い。FPGA で実際に回路を動かして検証し、仕様を満たすことを確認してから、マスクを作って集積回路の大量生産に入るのである。

我々は電子回路の設計を専門的に学んだわけではないので、限られた開発期間で求められた仕様を忠実に満たすバグの無い回路を設計することは難しい。また開発当初では想定していなかった要因により、仕様が変更されることもめずらしくない。そこで何度も書き換え可能な FPGA を用いて設計することにより、バグの修正や仕様の変更への追随に柔軟に対応できることが期待できた。

FPGA はもともと大量生産を目的としていなかったので安価ではなかった。また回路を何度でも書き換え可能な汎用性を持つ代わりに、専用に設計された集積回路と比較して、どうしても速度や集積度の点で劣っていた。しかし、今日では技

術競争により高速度・高密度の集積回路の製作が可能となり、十分な性能を持ったFPGAが作れるようになった。またPHSや携帯電話などの高度な電子回路を内臓した小型の製品が、一般に広く出回り激しい競争を繰り広げ、短期間で何度も世代交代を繰り返したため、新製品に組み込む集積回路の開発期間が非常に短くなり、集積回路のマスクを製作して大量生産する時間が無くなくなってきたので、大量生産される製品にもFPGAが使われるようになり、FPGA自体も大量生産されるようになって価格も下がってきた。このため、十分な性能を持ったFPGAが手頃な値段で[1]入手可能となり、物理実験機器の製作でFPGAを使用する条件が整ってきたと言える。

3.3.2 HDL

HDLとはHardware Description Languageの略であり、主にデジタル回路を設計するためのプログラミング言語である。FADC I/Fの製作では、このHDLを使うことにした。HDLを使用することにより、デバイスの詳細な知識が無くても、ある程度の性能で動作する回路の設計が可能となるからである。

集積回路の設計は、従来はAND (論理積), OR (論理和), NOT (論理否定), Flip-Flop (記憶素子)などの基本的な論理ゲートを配置配線することで行われていた。計算機上でCAD(Computer Aided Design)を使って論理ゲートを配置配線することにより、あらかじめシミュレーションで回路の動作を検証することも可能であった。しかし近年の集積回路の高密度・大規模化により、手作業による論理ゲートの配置配線では設計が手に負えくなってきた。また論理ゲートを単位とする方式では、ある程度構造を持ったセルを単位とするFPGAと、より基本的な論理ゲートで構成される専用集積回路との間で、互換性を持たせて設計することが難しい。開発段階ではFPGAを使い、大量生産では専用集積回路を使うことが多いので、両者の間での設計の共通化は重要である。そこでハードウェアの詳細を抽象化して記述できるHDLが開発され、使われるようになってきた。HDLで抽象的に回路を記述し、各ハードウェアの特性に合った回路を合成するソフトウェアを使って、HDLの記述を各々のハードウェアで動作する回路に変換するのである。比喩的にはFortranやC言語などの高級言語で記述されたプログラムを、コンパイラで変換して計算機が実行可能な機械語を生成するのに似ている。

HDLは本来、高密度・大規模な高性能集積回路を効率良く開発するために作られたのであるが、その高い抽象度のおかげで、ハードウェアの詳細を意識せずに設計できるという特徴を持っている。このためハードウェアの詳細な知識を持たない我々のような素人でも、ある程度の設計が可能である。HDLといえども、使用するハードウェアの性能を上限まで使い切るには、ハードウェアの構造を意識して設計する必要がある。しかし素人の設計した冗長で低速な回路でも、高性能なハードウェアで動かすことにより、必要な性能を出せると判断した。物理実験

[1]開発環境が数十万円程度。今回FADC I/Fの開発に使用したFPGAは約5万円程度。ただしこれは流通量の少ないパッケージのものを使用している。流通量の多いパッケージなら数万円程度である。

機器の製作では、営利目的の企業とは異なり価格競争の必要は無いので[2]、闇雲に性能を追及するよりも限られた時間と人的資源で目的を達成することの方が重要である。

現在主流の HDL には、VHDL と Verilog-HDL の 2 種類があるが、今回の FADC I/F の開発では Verilog-HDL を使用した。これは ASIC であるデータ圧縮 IC の開発で Verilog-HDL を使用したため、Verilog-HDL での開発に関するノウハウを得ていたためである。

3.3.3 CVS

CVS とは Concurrent Version System の略であり、大規模なソフトウェアの開発などでよく使われている、テキストファイル用の変更履歴管理システムである。FADC I/F の開発では、東京大学折戸研究室の松井氏が外部から送られて来る命令の解釈・実行部分の開発を担当し、わたし (土岐) が FADC からの連続読み出し部分の開発を担当して、複数人による開発形態を取ったので、CVS を用いて HDL のソースファイルの変更を管理することにした。

CVS は複数人で効率良くファイル[3]の一群を編集するために、主に次の機能を提供している。

複数人の変更を調停する CVS では repository と呼ばれる場所でファイルの一群を一括して管理し、各編集者はこの repository から checkout という操作でファイルを自分の手元に複製して、この複製したファイルに対して編集を行う。ファイルの編集が終了したら、commit という操作で複製ファイルの変更点を repository に通知し、この commit 操作の時点で各複製ファイルの変更点が repository のファイルに登録・反映される。各編集ファイルは各々の編集者の手元に複製されるので各編集者が同時に変更することが可能であり、また変更個所が重なった場合は commit 時に CVS が編集者に自動的に通知してくれるので、複数人でも安心してファイルを編集することが可能である。

任意の時点でのファイルの一群を取り出す CVS は各ファイルの変更を全て記録しているので、いつでも過去の任意の時点でのファイルの一群を取り出すことが可能である。また任意の二つの時点でのファイルを比較して、自動的に相違点を検出することも可能である。このためテキストにバグが混入した場合でも、過去のテキストと比較してバグを検出し、テキストをバグが混入する前の状態に戻すことが容易にできる。

任意の時点でのファイルの一群に名前を付ける CVS は任意の時点のファイルの一群に tag と呼ばれる名前を付けることができる。例えば実際に実験で使用したときのファイルの一群に tag を付けておくと、実験で使用したファイルに対する比較・参照が後で容易になり、機能の改善やバグの修正などがやりやすくなる。

[2] むろん予算には限りがあるので出来るだけ安いに越したことはないが、予算の範囲内であれば妥協できる。

[3] ここでのファイルとはテキストファイルのことを指す。

主流から分岐したファイルの一群を生成・管理できる CVS は repository で管理しているファイルの任意の時点での一群から、branch と呼ばれる独立した変更履歴の系列を生成できる。Branch は元のファイル群の変更履歴系列とはまったく独立しており、branch に対して編集操作を行っても元の変更履歴系列に影響を与えることはない。また branch に加えた変更点を merge という操作で、元の変更履歴系列に反映させることも可能である。実際に試してみなければうまく動作するかどうか分からない実験的な機能を、branch を生成して試しに実装して本当にうまく動作するか確認し、うまく動作することが確認できたら元のソースコード群に mergeして反映させる、というようなことが可能である。

CVS は「GNU 一般公有使用許諾書 (GNU General Public License)」に従って配布されるフリーソフトウェアである。CVS に関する日本語の詳細な情報は、次の WEB ページから得ることができる。

- `http://www.cyclic.com/cvs/lang-jp.html`
  (Cylic Software 日本語)
- `http://www-vox.dj.kit.ac.jp/nishi/cvs/cvs.html`
  (バージョン管理システム CVS を使う [西本卓也氏])
- `http://www-vox.dj.kit.ac.jp/nishi/cvs/cvs-manual/cvs-jp_toc.html`
  (CVS 1.9 日本語マニュアル [廣保誠氏])

3.3.4 開発工程

FADC I/F の開発は、Xilinx 社の HDL 統合開発環境である Foundation Express を使用した。開発言語は Verilog-HDL を使用したが Foundation Express は元々VHDL 用の開発環境なので、Verilog-HDL で記述された回路の論理合成は可能であるがシミュレーション用のソフトウェアが Verilog-HDL に対応していない。そこでシミュレーションはワークステーション上で Cadence 社の Verilog-XL というソフトウェアを使って行った。以下に、開発の工程について簡単に説明する。

1. **FADC I/F の仕様の決定。**BESS の現在のデータ収集系との互換性や、新しい FADC の機能などを考慮して、FADC I/F に必要な機能を洗い出す。
2. **HDL によるコーディング。**1 で決定した仕様を満たす回路を Verilog-HDL で設計する。
3. **論理シミュレーション。**計算機上でシミュレーションを行い、2 で設計した回路を実際に動作させてみて、本当に仕様通りに動作するか確認する。
4. **論理合成・配置配線。**HDL で設計した抽象的な回路を、IC の基本的な構成要素である論理ゲートに変換する作業を論理合成と呼び、変換したゲートの IC 上での物理的な位置を決定して各ゲートを配線する作業を配置配線と呼ぶ。通常の開発行程ではこの 2 つの行程が独立していることが多い。FADC I/F の開

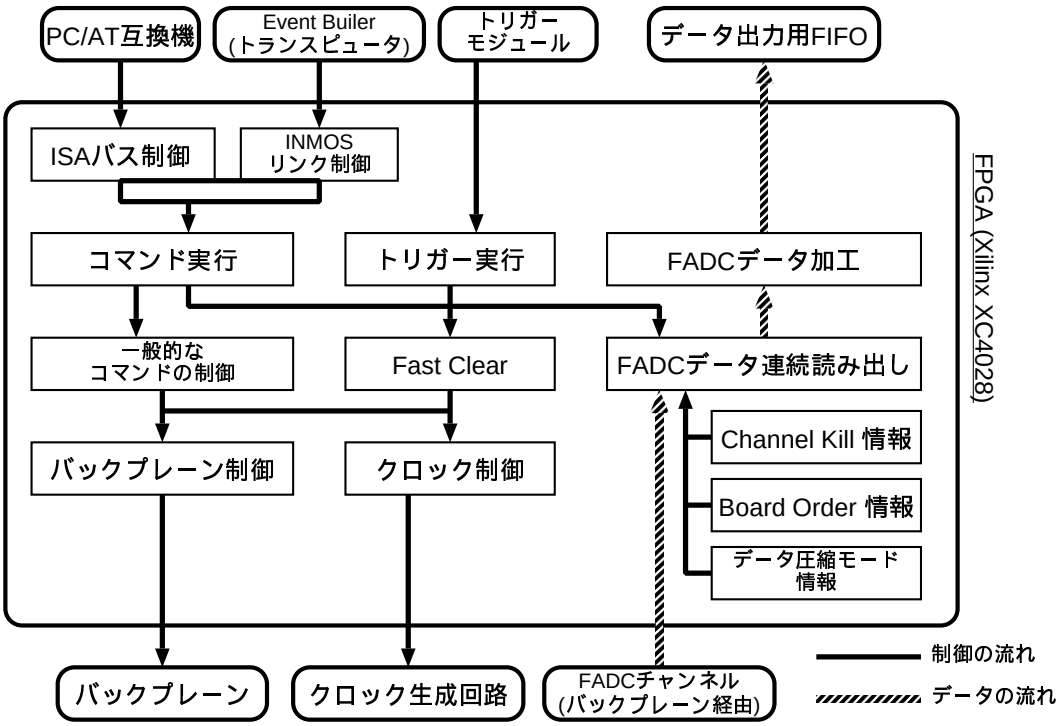


図 3.4: FPGA の内部構造のブロック図

発では、Foundation Express が PC 上で動作するので CPU の計算能力を独占できたことと、配置配線後でなければ 5 の遅延シミュレーションができないこともあって、論理合成と配置配線の作業は連続して行った。

5. **遅延シミュレーション。**配置配線後に出力される遅延データをもとに、回路が IC 上に物理的に配置されたときの遅延情報を考慮したシミュレーションを行い、回路が物理的な制約条件を満たして動作するかどうかを確認する。

実際の開発では、これらの行程が流れるように順番に進むのではなく、途中で仕様の問題や HDL コードのバグが判明したり、論理合成・配置配線後の回路の物理的な性能が不十分なことが判明したりして、何度も前の行程を反復している。なお HDL コードの修正では、前 3.3.3 節で説明した CVS の機能が非常に役立った。すべての行程を終えて回路が完成したら、回路を FPGA が理解できるデータ形式に変換し、FPGA にデータを書き込んで使用する物理デバイスに装着し、動作テストを行う。

3.4 機能と動作の説明

FADC I/F の機能と動作の大部分は FPGA 内部で実装されている。FPGA の内部構造のブロック図を図 3.4 に示す。FPGA の内部で実装されている回路は、主に次の 4 つの機能に関連するものとして分類できる。

- トリガー発生時の自動処理

- FADC の情報を記憶する RAM
- FADC データの連続読み出し
- 外部からの命令の実行やその他の細々とした処理

以下では、これらの機能について順番に説明していく。なお FADC の状態を記憶する RAM の部分と FADC データの連続読み出しを行う部分は、互いに密接に関係している。

3.4.1 トリガー

BESS は上下に配置された TOF カウンターが同時に反応する (coincidence) ことにより T0 トリガーが発生し、各検出器のデータ収集を開始する。T0 トリガーは FADC I/F 上の FPGA にも入力される。T0 トリガーが入力されると FPGA は自動的に、FADC モジュールを動作させてそのデータの収集を行う。また Master トリガーでイベントが棄却された場合は、FADC モジュールの動作やデータ収集を中断し、関連したモジュールの Fast Clear を行う。

3.4.1.1 FADC クロックの供給

FADC モジュール上の FADC とデータ圧縮 IC は外部クロックの供給を必要とする同期回路であるが、消費電力を抑えるためにトリガーの発生していない待機状態ではクロックが供給されていない。このためトリガーが発生してデータ収集を開始するときに、FADC I/F がクロックを供給してやらねばならない。具体的には、FADC I/F はトリガー発生直後に 30 MHz のクロック信号を 512 サンプル、FADC とデータ圧縮 IC に送る必要がある。

FADC I/F の主要部分を構成する FPGA は基本的に同期回路なので、非同期な処理を必要とするクロックの生成回路を FPGA 上に作るのは非常に難しい。そこでクロック生成回路は FPGA の外部に設けられた。クロック生成回路は 30 MHz の High と Low が均等に分配された 50 % duty のクロック信号を生成する。FADC I/F では FPGA がこの外部回路に指示を出すことによりクロックの生成を行う。トリガーが発生したら、FADC I/F はこのクロック生成回路を使って、自動的に 512 サンプルのクロックを FADC とデータ圧縮 IC に供給する。なおクロック生成回路の回路図は巻末の付録 A にある。

現在テスト用に製作している FADC I/F は、FADC モジュール上の FADC とデータ圧縮 IC に同一のクロックを送っている。しかし実際に BESS に搭載する FADC I/F では 2 つのクロック生成回路を実装して、FADC とデータ圧縮 IC のクロックを独立に生成できるようにする予定である。これは FADC の出力したデータがデータ圧縮 IC に伝わるのに少し時間が掛かり、この遅延時間を FADC とデータ圧縮 IC に送るクロックをずらすことによって調節するためである。

3.4.1.2 FADC データの連続読み出し

512 サンプルのクロックを送って FADC モジュールが JET chamber の信号を処理したら、FPGA は有効な FADC チャンネルのデータを全て読み出して、イベントビルダーに接続されている FADC I/F 上のデータ出力用 FIFO に読み出したデータを送る作業を行う。FADC モジュールでは各チャンネルのデータ圧縮 IC が内部の RAM に圧縮された FADC データを保存しているので、FADC I/F はこの RAM からデータを読み出す。データ圧縮 IC 内部の RAM は、外部からデータ読み出しのための非同期信号を送ることにより動作するので、クロックを供給しなくても RAM のデータを読み出すことができる。

連続読み出しにおいて FPGA は、FPGA が内部に持つ RAM の情報を参照しながら、FADC の有効なチャンネルのデータを指定された順番に読み出し、読み出したデータを適切に加工してからデータ出力用 FIFO に送る、という作業を行う。このように FADC データの連続読み出しは非常に複雑な作業なので、これらの作業の詳細は後の 3.4.2 から 3.4.4 の節で説明する。

3.4.1.3 Fast Clear

T0 トリガーによりデータ収集が開始されたイベントは、オンラインでマスタートリガーモジュールがデータを解析して、そのイベントを受け入れて記録する (Accept) のかそれとも棄却する (Reject) のかを決定する。マスタートリガーモジュールがイベントを受け入れると判断した場合は BESS 測定器全体に Accept 信号が送られ、逆にイベントを棄却すると判断した場合は Reject 信号が送られる。Reject 信号が送られて来たら、各測定器はデータ収集の動作を中断して、トリガーが発生する前の待機状態に戻らなければならない。

Reject 信号は FADC I/F 上の FPGA にも入力される。Rejct 信号が入力されると FPGA はまず最初に、FPGA 内部の FADC データ連続読み出しを行う回路をクリアして待機状態に戻す。次に FPGA は、FADC モジュールへ供給しているクロックを止めてから FADC モジュールにクリア信号を送って、FADC モジュールの状態をトリガーが発生する前の待機状態に戻す。次に FPGA は、CAMAC モジュールにクリア信号を送る。CAMAC モジュールは JET chamber 以外の TOF や IDC などの測定器の信号を処理するためのものなので、FADC I/F とは直接的には関係が無い。しかし現在の FADC I/F 上にはこれらの処理を行う Fast Clear モジュールが搭載されているため、互換性を保つために FPGA もこの Fast Clear モジュールと同等の動作を行うようになっている。CAMAC モジュールにクリア信号を送ったら、最後に FPGA は T0 トリガーモジュールに Fast Clear が終了したことを通知する信号を送り、Fast Clear の動作を終える。Fast Clear の動作を終えたら、FPGA はトリガーが入力される前の待機状態に戻る。

3.4.2 RAM

3.4.2.1 RAM の記憶する情報

FPGA が内部に持つ RAM では各 FADC チャンネルの情報が記憶されていて、連続読み出し中に参照される。全 FADC チャンネルのデータを連続して読み出すために必要な情報は、次の通りである。

- **それぞれの FADC を読み飛ばすための情報。**この論文では channel kill 情報と呼ぶ。FADC I/F には 32 の FADC チャンネルを持つ FADC モジュールを最大 32 枚接続することが可能なので、これはつまり最大で $32 \times 32 = 1024$ の FADC チャンネルが接続されることを意味するから、1 bit $\times$ 1024 address の RAM が必要である。

- **FADC モジュールを読み出す順序。**この論文では board order 情報と呼ぶ。本当は各 FADC チャンネルの読み出す順序を記憶したいのだが、データ量が多くて FPGA に収めることができないので、妥協して FADC モジュールの読み出す順序を記憶することにした。FADC モジュールは最大で 32 枚接続することが可能なので、5 bit $\times$ 32 address の RAM が必要である。

- **各 FADC モジュール毎の FADC データ圧縮モード。**FPGA は各 FADC チャンネルのデータ圧縮 IC のデータ圧縮モードに応じてデータを適切に加工する必要があるので、各 FADC チャンネル毎 (すなわち各データ圧縮 IC 毎) のデータ圧縮モードを記憶する必要がある。しかし全 FADC チャンネルのデータ圧縮モードを記憶しようとすると、データ量が多くて FPGA に収めることができないので、妥協して各 FADC モジュール毎にデータ圧縮 IC のデータ圧縮モードを記憶することにした。2 bit $\times$ 32 address の RAM が必要である。

それぞれの RAM には、個々のデータの読み込みと書き込みを行う回路と、RAM の全データを連続して読み出すための専用の回路が接続されている。また channel kill 情報を記憶する RAM と FADC チャンネルのデータ圧縮モードを記憶する RAM には、RAM の全データを一つの値で初期化する回路が接続されている。

3.4.2.2 RAM データの連続読み出し

全 FADC チャンネルのデータを連続して読み出すには、前 3.4.2.1 節で説明した RAM のデータを、各々の内容に従って連続して読み出す必要がある。Board order 情報の順序に従い、channel kill 情報の設定された FADC チャンネルを跳ばしながら、ボード・チャンネル番号と FADC モジュール毎の FADC データ圧縮モードを取り出さなければならない。

ここでボード番号とは各 FADC モジュールを特定する番号のことであり、チャンネル番号とは FADC モジュール上の各 FADC チャンネルを特定する番号のことである。FADC モジュールは最大で 32 枚なのでボード番号は 5 bit ($2^5 = 32$) のデータであり、FADC チャンネルは FADC モジュール上に 32 チャンネルあるのでチャ

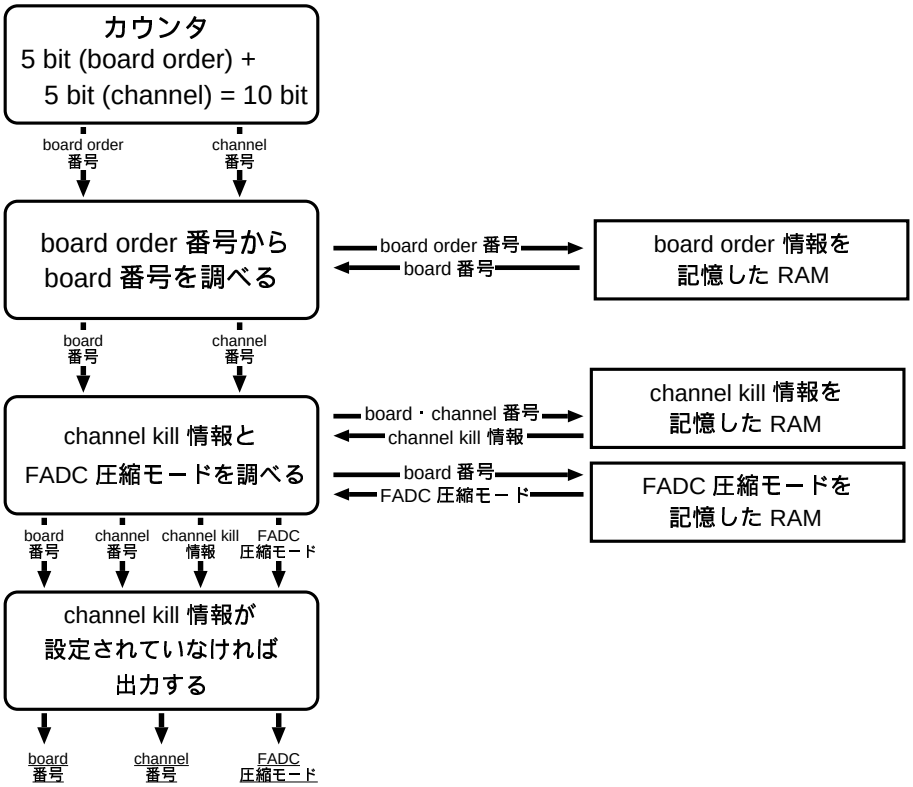
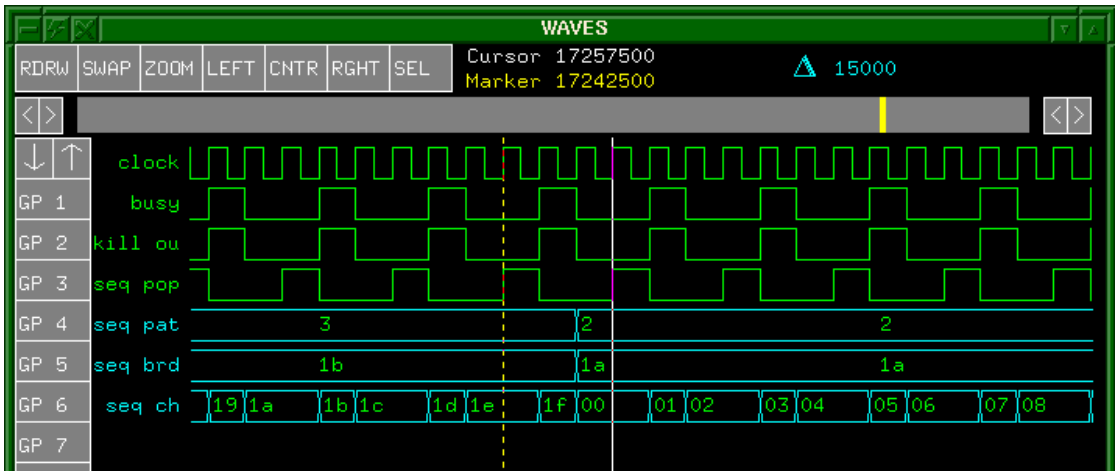


図 3.5: RAM データの連続読み出し

ンネル番号も 5 bit のデータである。ボード番号とチャンネル番号を指定することで、最大 1024 の FADC チャンネルから特定のチャンネルを指定することができる。

RAM には、専用の読み出し回路を通じてこの動作を行う回路が接続されている。RAM データの連続読み出しの動作は、次の様な手順で行う (図 3.5, 図 3.6 参照)。

1. ボード番号用に 5 bit, チャンネル番号用に 5 bit, 合わせて 10 bit 幅のカウンターを昇順に数え上げて、0 をオフセットとして最大 1023 までの FADC チャンネルのアドレスを生成する。アドレスの上位 5 bit を board order 番号として使い、下位 5 bit をチャンネル番号として使う。
2. 1 で生成した board order 番号をアドレスとして board order 情報を記憶した RAM を参照し、実際に読み出すボード番号を得る。
3. 2 で得られたボード番号と 1 で得られたチャンネル番号をアドレスとして用いて、channel kill 情報を記憶した RAM と FADC データ圧縮モードを記憶した RAM のデータを読み出す。
4. 3 で得られた channel kill 情報を調べて、チャンネルが kill されてなければ、2 で得られたボード番号と 1 で得られたチャンネル番号と 3 で得られた FADC データ圧縮モードを出力する。



clock 20 MHz のクロック。

busy この信号が High のとき、データを読み込んでいる最中であることを表している。

kill out channel kill 情報が出力されている。この信号が High のとき、ボード・チャンネル番号が無効であることを表している。

seq pop この信号を High にすると、次のデータの読み出しを実行する。

seq pat データ圧縮 IC の圧縮モードが出力されている。

seq brd ボード番号が出力されている。

seq ch チャンネル番号が出力されている。

FPGA 内部の回路が RAM から各種データを連続して読み出している様子を、シミュレーションで確認している。図を垂直に横切っている緑の点線から白線までの間が、データの読み出しに必要な 1 サイクルである。テスト用に与えたデータは奇数番号のチャンネルが全て kill されて無効になっており、kill out 信号が High のときbusy 信号が High になり、kill されたチャンネル番号が読み飛ばされていることが分かる。

図 3.6: FADC I/F 内部の RAM を連続して読み出しているシミュレーションの様子

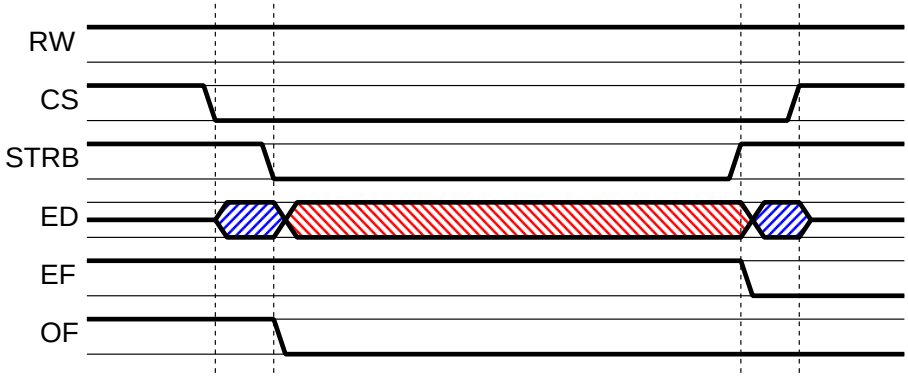


図 3.7: FADC のデータを読み出すタイミング

3.4.3 FADC データの連続読み出し

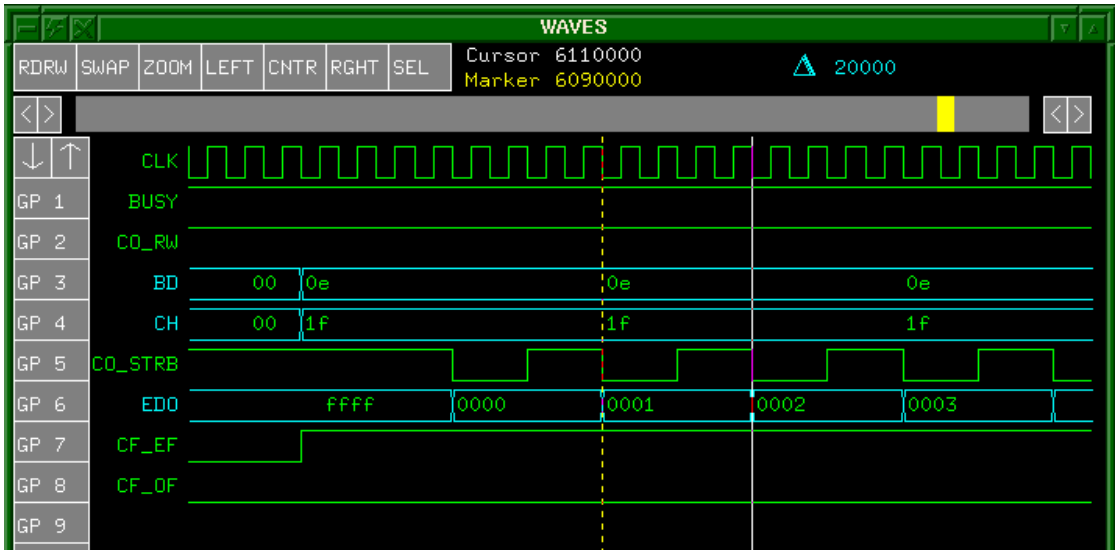
FADC データの連続読み出しでは、FPGA 内部の RAM に設定された情報に従って最大 1024 の FADC チャンネルのデータを連続して読み出す。前 3.4.2.2 節で説明したように RAM データの連続読み出し回路からは、kill されていない FADC チャンネルのボード・チャンネル番号が board order 情報に従った順序で出力される。FADC データの連続読み出しを実行する回路は、このボード・チャンネル番号に従って順番に各 FADC チャンネルのデータの読み出しを行う。

3.4.3.1 FADC データの読み込み

FADC チャンネルが処理したデータは、各チャンネルのデータ圧縮 IC が内部に持つ RAM に保存されている。このデータ圧縮 IC 内部の RAM のデータを読み込むには、次の手順で行なう必要がある (図 3.7, 図 3.8 参照)。

1. データ圧縮 IC の RW ポートの信号を High に設定してから、データ圧縮 IC の CS ポートの信号を Low に設定してデータ圧縮 IC への操作を有効にする。
2. 続いてデータ圧縮 IC の STRB ポートの信号を High から Low に下げると、STRB 信号の negative edge によりデータ圧縮 IC の非同期 RAM が作動して、RAM のデータがデータ圧縮 IC の ED ポートに出力され、オーバーフローフラグの値がデータ圧縮 IC の OF ポートに出力される。
3. 続いて STRB ポートの信号を Low から High に上げると、STRB 信号の positive edge によりデータ圧縮 IC の非同期 RAM が作動して、ED ポートのデータが無効になる。このとき RAM の全データを読み出していたら、empty フラグの値がデータ圧縮 IC の EF ポートに出力される。
4. EF ポートの値が Low になるまで 2 と 3 の手順を繰り返す。
5. RAM のデータを全て読み出したら、データ圧縮 IC の CS ポートの信号を High に設定してデータ圧縮 IC への操作を無効にする。

各 FADC チャンネルのデータ圧縮 IC の入出力ポートと FPGA はバックプレーンで接続されているので、データ圧縮 IC と信号の遣り取りをするときは信号の遅延



CLK 20 MHz のクロック。

BUSY この信号に常に High が出力されているのは、FADC I/F が動作中であることを表している。

CO_RW データ圧縮 IC のRW ポートへ送る信号が出力されている。

BD ボード番号が出力されている。

CH チャンネル番号が出力されている。

CO_STRB データ圧縮 IC のSTRB ポートへ送る信号が出力されている。

EDO データ圧縮 IC の RAM のデータが出力されている。

CF_EF データ圧縮 IC の empty フラグが出力されている。

CF_OF データ圧縮 IC のオーバーフローフラグが出力されている。

FPGA がデータ圧縮 IC のデータを連続して読み出している様子を、シミュレーションで確認している。図を垂直に横切っている緑の点線から白線までの間が、データの読み出しに必要な 1 サイクルである。データ圧縮 IC のCS ポートの信号は、BD 信号とCH 信号をバックプレーンへ送ることにより、FADC モジュールが適切な FADC チャンネルのデータ圧縮 IC を選択してCS 信号を設定している。

図 3.8: データ圧縮 IC 内部の RAM のデータを読み出しているシミュレーションの様子

を考慮する必要がある。当初は FADC I/F を 20 MHz で動作させて、STRB 信号の幅を 1 クロック (50 nano second) にする予定であった。しかしバックプレーンや FADC モジュール上のチャンネル選択回路を往復する遅延や、FPGA が内部に信号を取り込むときの遅延が予想以上に大きく、信号の往復時間が 50 nano second では間に合わない可能性が出てきたため、STRB 信号の幅を 2 クロック (100 nano second) に変更した。さらに Verilog-HDL のソースコード中でSTRB 信号の幅をパラメータで指定し、STRB 信号の幅を 2 クロック以上の任意の大きさに変更できるようにした。

3.4.3.2　一時停止のアルゴリズム

FADC チャンネルから読み込まれたデータは、データ圧縮 IC のデータ圧縮モードに応じて適切に加工された後、イベントビルダーに接続されている FADC I/F 上のデータ出力用 FIFO へ出力される。このデータ出力用 FIFO が full になったときに備えて、FADC データの連続読み出しを行う回路は連続読み出しの動作を一時停止する必要がある。

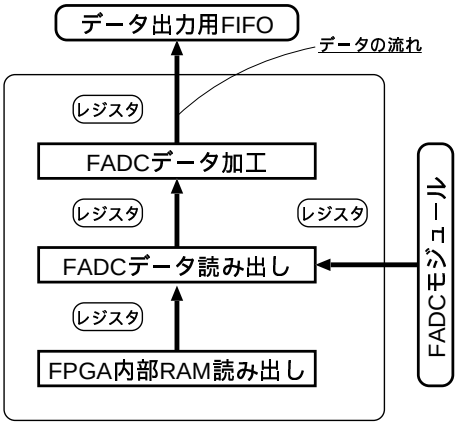
FPGA 内部の回路では、FPGA の動作速度の遅さをカバーするために「RAM データの連続読み出し」・「FADC データの連続読み出し」・「FADC データの加工」の 3 つの処理を分割して独立した回路で行い、並列に動作するそれぞれの回路に連続してデータを流すことで高速化を図っている。このように流れ作業でデータを処理する回路全体のことを一般にパイプラインと呼び、データ処理を分担されて並列に動作するそれぞれの部分回路のことをステージと呼ぶ。

パイプラインは同期回路のデータ処理速度を高速化するのに非常に有効な手段ではあるが、その一方で、各ステージの回路が独立しているためパイプライン全体を一斉に停止させることが非常に難しいという問題がある。しかし先程説明した通りデータ出力用 FIFO が full 状態のときは、FADC データ連続読み出しの処理を行うパイプラインを一時停止させる必要がある。

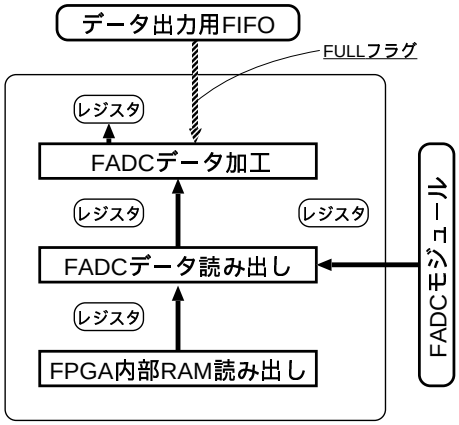
開発当初はパイプラインの動作と一時停止を決定する信号をパイプラインの回路全体に配って、回路全体を一斉に一時停止させていた。しかし開発が進み回路の規模が大きくなるにつれて、パイプライン全体に停止信号を配ることによる遅延が増して行き、ついには遅延が 50 nano second を上回り予定された動作速度である 20 MHz で回路を動作させることができなくなってしまった。

そこでこの問題を解決するためにパイプラインの各ステージの間に、一時停止中にデータを退避するレジスタを設けた。データ出力用 FIFO が full になり一時停止信号が送られて来たら、データ出力用 FIFO に近いステージから順番に一時停止して行き、一時停止の間は処理中の信号を各ステージ間のレジスタに退避するのである。この方法により各ステージ毎に回路の独立性を保つことができるので、パイプライン全体を一時停止させる必要が無くなり、回路規模の増加によって遅延が増大するという問題から逃れることができた。図 3.9 で実際にパイプラインが稼働中にデータ出力用 FIFO が full になったときの動作を説明している。

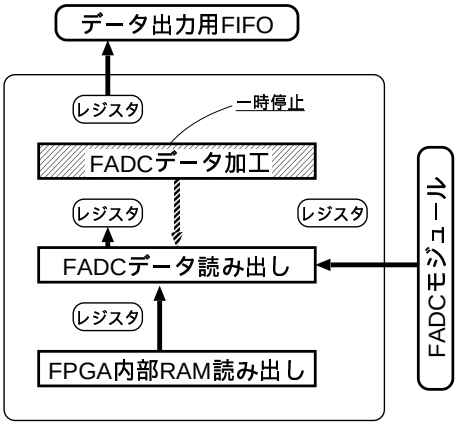
図 3.9(a) の説明 パイプラインの各ステージが並列に動作してデータを処理している。



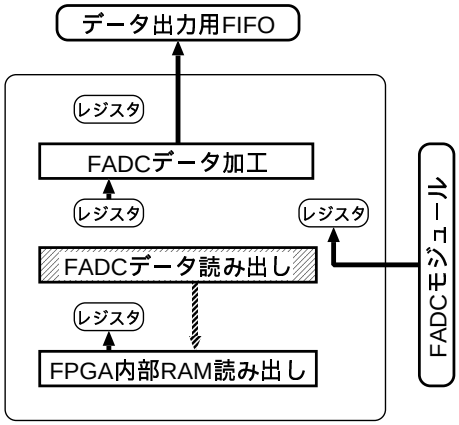
(a) データ処理中



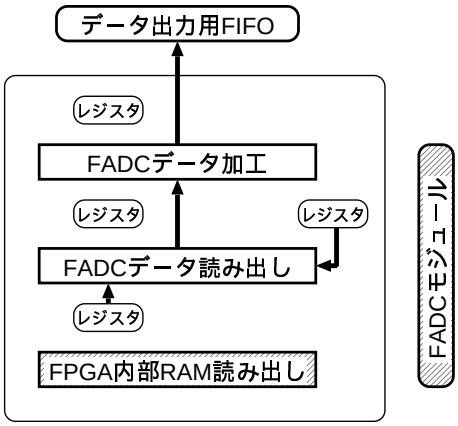
(b) データ出力用 FIFO が full になった



(c) FADC データ加工の一時停止



(d) FADC データ読み出しの一時停止



(e) FPGA 内部 RAM 読み出しの一時停止

図 3.9: パイプラインの一時停止

図 3.9(b) の説明 データ出力用 FIFO が full 状態になると、まず最初にデータ出力用 FIFO に最も近い FADC データを加工するステージに full フラグが伝えられる。full フラグを受け取ると、FADC データ加工ステージはその時データ出力用 FIFO に出力しようとしていたデータをレジスタに退避してから一時停止に入る。

図 3.9(c) の説明 一時停止した FADC データ加工ステージは、次の FADC データの読み出しを行うステージに full フラグを伝える。FADC データ加工ステージがデータ出力用 FIFO に対して行ったのと同様に、FADC データ読み出しステージは、データを FADC データ加工ステージに渡す代わりにレジスタに退避してから一時停止に入る。このときデータ出力用 FIFO が full 状態から再び書き込み可能な状態になると、FADC データ加工ステージが一時停止中なので、データ出力用 FIFO はレジスタから退避されていたデータを受け取り、FADC データ加工ステージの一時停止が解除される。

図 3.9(d) の説明 一時停止した FADC データ読み出しステージは、次の FPGA 内部 RAM の読み出しを行うステージに full フラグを伝える。FPGA 内部 RAM 読み出しステージは、データを FADC データ読み出しステージに渡す代わりに、レジスタに退避してから一時停止に入る。また FADC データ読み出しステージが一時停止に入る直前に、読み出そうとしていた FADC チャンネルのデータもレジスタに退避される。再び動作を始めた FADC データ加工ステージは、FADC データ読み出しステージが一時停止中なので、レジスタから退避されていたデータを受け取り、FADC データ読み出しステージの一時停止が解除される。

図 3.9(e) の説明 再び動作を開始した FADC データ読み出しステージは、FPGA 内部 RAM 読み出しステージと FADC モジュールが一時停止中なので、レジスタから退避していたデータを受け取る。そして FPGA 内部 RAM 読み出しステージと FADC の一時停止が解除され、次のクロックからは再び図 3.9(a) の状態に戻る。

3.4.3.3 外部入力信号を高速に読み込むアルゴリズム

FADC チャンネルのデータは、FADC モジュール上のチャンネル選択回路やバックプレーンを経て送られて来るので、FPGA に到達した時点でかなりの遅延が生じている。このためバックプレーンを経て送られて来た FADC データを FPGA に引き込んで直接参照しようとすると、FPGA 内部での遅延にさらにバックプレーンを経由して来た遅延が加わり、特に高速動作が要求される FADC データの連続読み出しの動作に間に合わなくなってしまう。

この問題を解決するために、FADC データの連続読み出しで使用されるデータ圧縮 IC の CF_EF ピン (empty フラグ), CF_OF ピン (オーバーフローフラグ), EDO ピン (RAM データ) の信号に関しては、バックプレーンを経て FPGA に入力された信号を一旦レジスタに記憶してから FPGA 内部で参照している。一旦レジスタに

記憶してから参照することで、FPGA 内部で信号を参照するタイミングが最大で 1 クロック遅れてしまうことになるが、その代わり FPGA 内部の回路でバックプレーンを経由して来たために生ずる遅延を考慮する必要が無くなる。また FPGA 内部のバックプレーンの信号と直接接続されている回路は、常にバックプレーンから送られて来た信号をレジスタに保存するだけの単純なものになるので、バックプレーン経由の遅延に加わる内部の遅延は必要最小限になる。

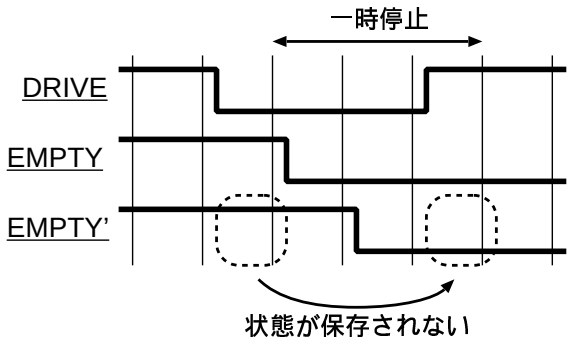
しかしながら、バックプレーンの信号を一旦レジスタに記憶してから内部で参照するというアルゴリズムには、実は 1 つ問題点がある。前 3.4.3.2 節で説明したが、FADC データの連続読み出しの回路は出力先のデータ出力用 FIFO が full 状態になったとき一時停止しなければならない。この一時停止の間にバックプレーンからの入力信号が変化すると、一時停止から復帰したときの信号の状態が一時停止前と異なってしまうのである。この現象は FPGA 内部の FADC データ連続読み出し回路に矛盾を生じさせる。

以下、情報が 1 bit で最も単純な、データ圧縮 IC の empty フラグを例に説明する。図 3.10(a) はさきほど説明した素朴なアルゴリズムで、一時停止中に empty フラグが変化したときの動作の様子である。図中の細い縦線はクロックにより回路が動作する時刻を表し、太い横線は信号が時間経過により変化していく様子を表している。時間は左から右に経過し、時間の経過に従って信号の太線が上下に変化している。ここで説明する信号は全て 1 bit で値を 2 つしかとらないので、太線が上側にある状態が High (1) を表し、下側にある状態が Low (0) を表すことにする。FPGA は同期回路なので、クロックにより起動された瞬間にしか回路は動作しないことに注意する必要がある。

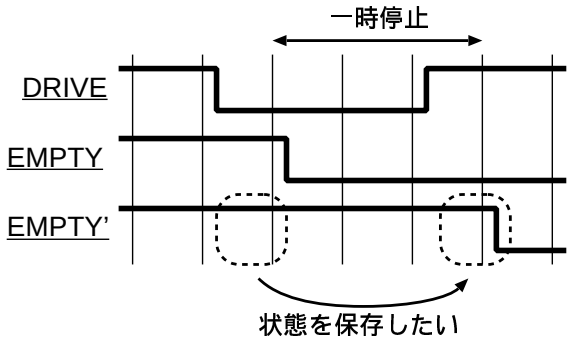
図中の DRIVE は連続読み出しの稼働状態を決定し、この信号が High のとき連続読み出し動作を行い、Low のとき一時停止する。前 3.4.3.2 節で説明した一時停止のアルゴリズムで、データ出力用 FIFO が full 状態になったときに送られてくる full フラグは、FPGA の内部では DRIVE 信号に変換される。図中の EMPTY はバックプレーンを経て送られてくるデータ圧縮 IC の empty フラグである。データ圧縮 IC の empty フラグは負論理なので、EMPTY 信号は High のとき偽の状態 (データ圧縮 IC の RAM が空ではない) を表し、Low のとき真の状態 (データ圧縮 IC の RAM が空である) を表している。EMPTY' は FPGA 内部でバックプレーンから送られて来た EMPTY 信号を記憶するレジスタの内容を表している。

図 3.10(a) ではまず DRIVE 信号が High から Low に切り替わり、連続読み出し動作が一時停止に入る。次に EMPTY 信号が High から Low に変わり、続いて 1 クロック遅れて EMPTY' レジスタの内容が EMPTY 信号の変化に追随する。その後、DRIVE 信号が Low から High に戻り再び連続読み出しの動作が始まると、EMPTY' レジスタの内容が High から Low に変わってしまっているので矛盾が生じてしまう。

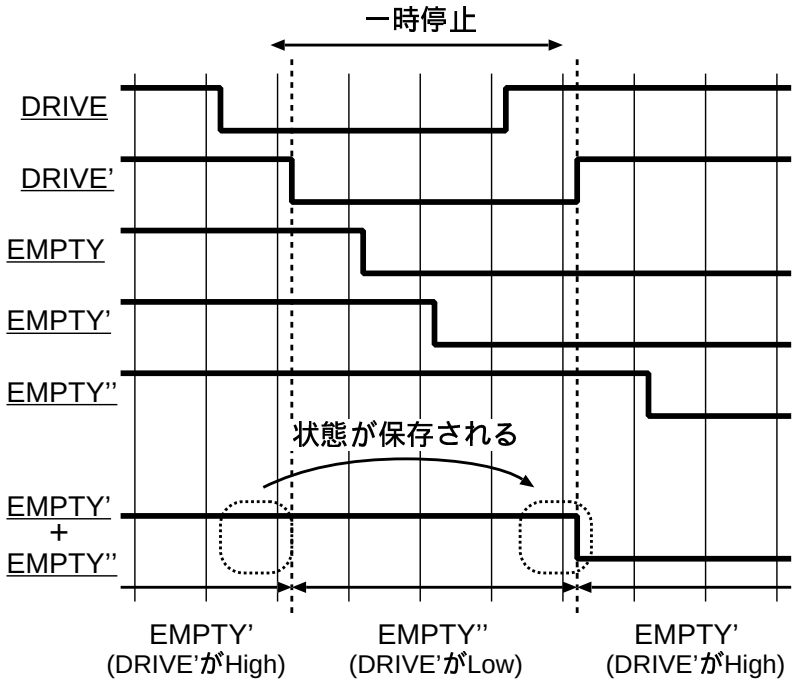
この問題を解決するためには、図 3.10(b) のように DRIVE 信号が High に戻ってから EMPTY' レジスタを変化させる必要がある。この動作を行うための最も単純な方法は、DRIVE 信号が High のときだけ EMPTY 信号を EMPTY' レジスタへ保存することであるが、この方法には問題がある。DRIVE 信号はパイプラインの各ステージの動作を決定する信号であり、各々のステージの回路全体から参照されているため、切り替えに時間がかかる。したがって DRIVE 信号を参照すると DRIVE



(a) 高速だが一時停止前の状態が保存されない



(b) 低速だが一時停止前の状態が保存される



(c) 高速でかつ一時停止前の状態が保存される

図 3.10: empty フラグの読み込み

信号の遅延に引っ張られて、このためEMPTY 信号をEMPTY'レジスタへ記憶する動作も遅くなってしまうのである。これではEMPTY 信号をEMPTY'レジスタに記憶させて高速化を図った意味が無い。

これらの問題を根本的に解決する方法を図 3.10(c) に示す。今度は図 3.10(a) と図 3.10(b) とは異なり、DRIVE'とEMPTY"が新たに加わっている。DRIVE'はDRIVE 信号を記憶するレジスタの内容で、DRIVE 信号の変化に 1 クロック遅れて追随する。EMPTY' は図 3.10(a) と同じでEMPTY 信号を常に記憶し続けるレジスタである。EMPTY"はEMPTY'レジスタの内容を記憶するレジスタで、DRIVE'レジスタが High のときEMPTY'レジスタの内容を記憶する。バックプレーンからの信号を直接参照しているのはEMPTY'レジスタだけであり、その動作は図 3.10(a) とまったく同じで、他の信号はすべて FPGA 内部のものを参照していることが重要なポイントである。これは図 3.10(a) の回路の外部信号に対する遅延が、図 3.10(b) の回路とまったく同じになることを意味している。

図 3.10(c) の動作の様子を順を追って説明する。まずDRIVE 信号が High から Low に変化し、続いて 1 クロック遅れてDRIVE'レジスタが High から Low に変化して、連続読み出し動作が一時停止状態に入る。次にEMPTY 信号が High から Low に変化し、続いて 1 クロック遅れてEMPTY'レジスタが High から Low に変化するが、EMPTY"レジスタはDRIVE'レジスタが Low なので、High の状態を保つ。DRIVE 信号が Low から High に戻って一時停止状態から動作が再開すると、1 クロック遅れてDRIVE'レジスタが Low から High に戻り、さらに 1 クロック遅れてEMPTY"レジスタがEMPTY'レジスタの内容をコピーして High から Low に変化する。

以上の動作により、結果としてEMPTY"レジスタは一時停止中に、一時停止に入る直前のEMPTY'レジスタの内容を保存することになる。したがってDRIVE'レジスタが High のときはEMPTY'レジスタの内容を参照し、Low のときはEMPTY"レジスタの内容を参照すれば、図 3.10(b) のEMPTY'レジスタの動作を図 3.10(a) の回路と同じ遅延で実現することが可能である。データ圧縮 IC の RAM データ (EDO ピン) とオーバーフローフラグ (CF_OF ピン) の信号も、empty フラグと同様のアルゴリズムによって FPGA に高速に読み込まれる。

遅延シミュレーションの結果ではこのアルゴリズムを使用することにより、外部入力データを読み込む速度が 30 nano second から 18 nano second に改善された。データの読み出しは 2 クロックすなわち 100 nano second で行われるので、これは FPGA がバックプレーンを通してデータ圧縮 IC にSTRB 信号を送ってからデータが返って来るまでに、最大で $100 - 18 = 82$ nano second まで待てることを意味している。ただしこれはデータを連続して読み出す場合のみで、FADC チャンネルを切り替えた直後は、このアルゴリズムでは最初のデータを記憶するのに 1 クロック余計に消費するので、合計で 3 クロック待たなければならない。

3.4.4 FADC データの加工

2 章の 2.4.1 節で説明したように、FADC チャンネルのデータ圧縮 IC が出力するデータには SUM, CRAW, HIST, RAW の 4 種類がある。これらのデータの中には、データの構造に問題のあるものや余分なデータを含むものがあるので、各々のモー

ドによってデータの構造を修正したり余分なデータを削除する必要がある。このため FPGA は FADC チャンネルの圧縮モードによって適切にデータを加工しなければならない。

なお FADC データを加工する回路は完全にパイプライン化されていて、1 クロック毎に連続してデータを流しても問題は無い。したがってデータ圧縮 IC のデータの読み出しに 2 クロックを使うのはデータの読み出しの遅延が問題であるからであり、将来バックプレーンおよび BESS 搭載用 FADC モジュールのチャンネル選択回路の改良、およびより高速で上位互換な FPGA の使用により遅延が問題で無くなれば、データの読み出しを 1 クロックで行うことも可能である。ただ現在のデータ圧縮 IC 内部の RAM を読み出す回路は、データを読み出す遅延が 2 クロック以上であることに完全に依存しており、この部分を書き直さなければ 1 クロックの遅延でデータ圧縮 IC のデータを読み出すことはできない。ODC の導入により FADC チャンネル数が増加することを考えると、1 クロックの遅延でも読み出し可能にしておくことが望ましいが、これは将来の課題である。

3.4.4.1 SUM モードで出力されるデータの加工

データ圧縮 IC の SUM モードで出力されるデータの構造は 2 章 2.4.1.1 節の図 2.8 の様になっているが、このデータ構造には次の 2 つの問題点がある。

1. チャンネル番号の幅が 8 bit しかない。
2. RAM のオーバーフローを記録できない。

まず 1 番目の問題について説明する。2 章 2.4.1.1 節の図 2.8 を見ると分かるように、SUM モードで出力されるクラスタのデータに付けられるチャンネル番号の幅は 8 bit になっている。これは現在使用している FADC のデータ構造をそのまま使用しているのであるが、チャンネル番号を記録する領域が 8 bit しかないことが開発中の FADC I/F では問題になる。

現在 BESS で使用している FADC I/F に接続できる FADC の最大数は 256 個 (8 bit) なので、チャンネル番号の幅は 8 bit で問題ない。しかし開発中の FADC I/F に接続できる FADC の最大数は、ボードの数が 32 個 (5 bit) で 1 ボード上の FADC チャンネル数が 32 個 (5 bit) なので、$32\ (5\text{ bit}) \times 32\ (5\text{ bit}) = 1024$ 個 (10 bit) すなわち 10 bit のチャンネル番号が必要である

データ圧縮 IC を設計した段階では FADC チャンネルの数を現在より増やすことを考慮していなかったのでこのような仕様になってしまったのであるが、データ圧縮 IC は既に量産体制に入っており設計を変更することは事実上不可能である。幸いにもクラスタのデータ構造は最初の 7 bit が空いているので、この場所にチャンネル番号に不足している 2 bit を FPGA が書き込むことで対処が可能である。

次に 2 番目の問題について説明する。クラスタの数が 64 個以上発生して RAM の容量を越えた場合、データ圧縮 IC は OF ピンに High を出力して FADC I/F にオーバーフローを知らせるが、クラスタのデータには記録されないので、後で保存したデータを参照しても FADC の RAM のオーバーフローを知ることはできない。そこで FPGA がクラスタのデータにデータ圧縮 IC の OF ピンの出力を記録すること

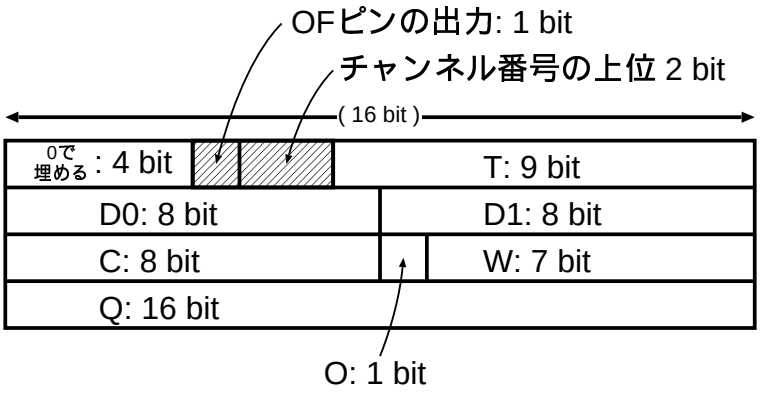


図 3.11: 加工された SUM モードのクラスタデータの構造

でこの問題に対処することにした。

図 3.11 に FPGA が加工した後のデータ構造を示す。クラスタデータの先頭の 7 bit の空領域の内、後の 3 bit を新たに使用している。3 bit の内、上位 1 bit はデータ圧縮 IC の OF ピンの出力が記録され、下位 2 bit にはチャンネル番号の上位 2 bit が記録される。

3.4.4.2 CRAW モードで出力されるデータの加工

データ圧縮 IC の CRAW モードで出力されるデータの構造は 2 章 2.4.1.2 節の図 2.9 の様になっているが、このデータ構造には SUM モードと同様に次の 2 つの問題点がある。

- チャンネル番号の幅が 8 bit しかない。
- RAM のオーバーフローを記録できない。

CRAW モードのデータ構造は SUM モードと異なり可変長でデータの長さが決まっておらず、またデータの終端の検出方法も単純ではないので、FPGA でデータの終端を検出する複雑なアルゴリズムを実行するのは難しい。そこでデータ圧縮 IC が出力するデータの前に 10 bit のチャンネル番号と 1 bit のオーバーフローフラグを記録したヘッダを追加することにした。

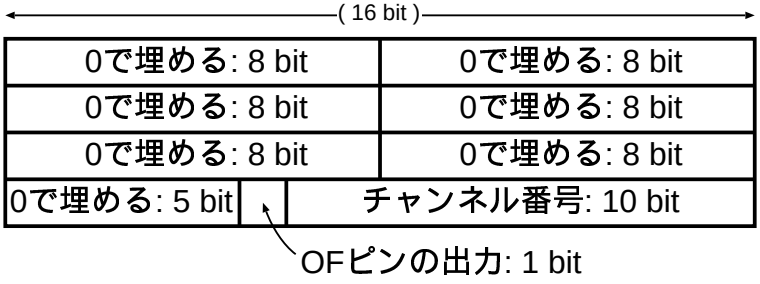


図 3.12: CRAW モードのデータに付加されるヘッダのデータ構造

CRAW モードで出力されるデータに FPGA が付加するヘッダのデータ構造を図 3.12 に示す。ヘッダではまず最初に 8 bit × 6 = 48 bit の連続した 0 が出力される。CRAW モードの可変長クラスタデータ中に埋め込まれる 0 の最大長は 8 bit × 4 = 32 bit なので、48 bit の連続した 0 をヘッダ中に埋め込むことにより可変長クラスタのデータとヘッダを区別できる。

続いて次の 16 bit でチャンネル番号とオーバーフローフラグが記録される。16 bit の内上位 5 bit は使用されずに 0 で埋められ、次の 1 bit でデータ圧縮 IC の OF フラグの出力が記録され、残りの 10 bit でチャンネル番号が記録される。

3.4.4.3 HIST モードで出力されるデータの加工

データ圧縮 IC の HIST モードで出力されるデータの構造は 2 章 2.4.1.3 節の図 2.10 の様になっている。図 2.10 を見れば分かる様に、データの半分は 0 で埋められているだけであり、この部分を出力するのは無駄である。このため FPGA は HIST モードの出力データの内、0 で埋められている奇数番号のデータは読み捨てて、偶数番号のヒストグラムデータのみを出力する。

HIST モードは、SUM モードや CRAW モードのように記録するためのデータを生成するモードではなく、BESS が稼働中に定期的に実行される FADC の pedestal スキャンの実行を補助するためのモードである。このため各 FADC チャンネル毎にデータを読み出すので、SUM モードや CRAW モードのようにチャンネル番号や RAM のオーバーフローを記録する必要は無い。

3.4.4.4 RAW モードで出力されるデータの加工

データ圧縮 IC の RAW モードで出力されるデータの構造は 2 章 2.4.1.4 節の図 2.11 の様になっている。RAW モードはデータ圧縮 IC が FADC のデータを正常に処理できているかどうかを確認するためのデバッグ用のモードなので、FPGA はデータ圧縮 IC から読み出したデータをそのまま出力する。

3.4.5 命令の実行

FADC I/F は通常、起動すると待機状態に入り、トリガーが入力されると FADC モジュールを動作させてからデータを全て読み出してイベントビルダーに接続されたデータ出力用 FIFO へ送り、データの読み出しが終了すると再び待機状態に戻る、という動作を行う。しかし 3.4.2.1 節で説明した channel kill 情報や board order 情報を設定したり、個々の FADC チャンネルの設定を変更したりするためには、外部から FADC I/F に何らかの方法で情報を送る必要がある。また FADC I/F の動作を確認するためにも、FADC I/F の持つ個々の機能を外部からの操作で起動できることが望ましい。

このため FADC I/F は、外部から命令を受け取って実行することができるようになっている。待機中の FADC I/F に命令を送ることによって、FPGA を動作させて channel kill 情報や board order 情報を設定したり、FADC チャンネルの設定を変更

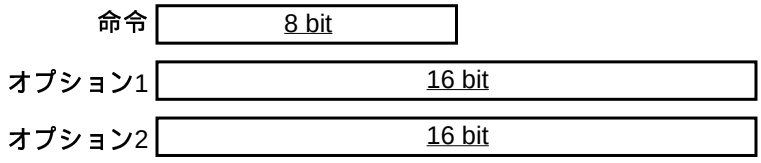


図 3.13: FADC I/F へ送る命令のデータ構造

したり、FADC I/F の各種機能を呼び出したりすることができるようになっている。FADC I/F に送ることのできる各種命令は、付録 C で説明している。

FADC I/F に命令を送る方法は二つある。一つは ISA バスを使って PC から命令を送る方法であり、もう一つは INMOS リンクを使ってトランスピュータから命令を送る方法である。どちらの方法も FADC I/F へ送る命令のデータ構造は共通である。命令のデータ構造は常に固定長で

1. 8 bit の命令部分。
2. 16 bit の 1 番目のパラメータ (オプション 1)。
3. 16 bit の 2 番目のパラメータ (オプション 2)。

という構成になっている (図 3.13 参照)。オプション 1 とオプション 2 は常に必要で、命令がオプションを使用しない場合でも 0 で埋めた値を渡す必要がある。

FADC I/F に送られて来た命令はいずれの場合も、FPGA が I/O を補助する IC を使って自動的に読み出して実行する。

3.4.5.1 ISA バス

ISA バスは PC で周辺機器の接続などに使われる 16 bit のバスである。ISA バスは主に FADC I/F を単体でテストするときに使用する。ISA バスからの入出力は、次の 5 つの信号を使って行われる (図 3.14 参照)。

ISA_SEL 1 bit の入力。ISA バスを使ってデータの入出力を行う間、この信号を

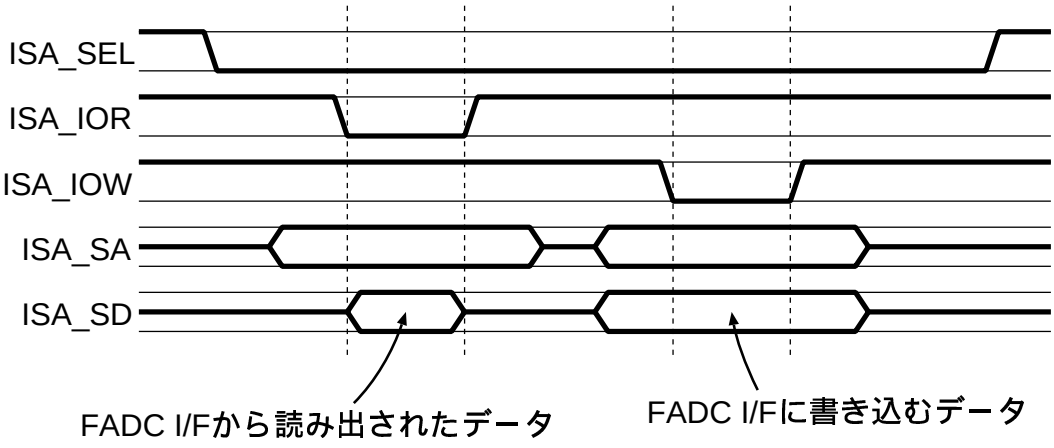


図 3.14: ISA バスの入出力

Low に設定する。

ISA_IOR 1 bit の入力。ISA バスを使ってデータを FADC I/F から読み出すとき、ISA_SA の値を適切に設定してからこの信号の値を Low に設定し、ISA_SD からデータを読み出す。

ISA_IOW 1 bit の入力。ISA バスを使ってデータを FADC I/F へ書き込むとき、ISA_SA と ISA_SD の値を適切に設定してからこの信号の値を Low に設定すると、データが FADC I/F へ書き込まれる。

ISA_SA 4 bit の入力。ISA アドレスを入力する。

ISA_SD 16 bit の入出力。ISA_IOR が Low のときはここからデータが出力され、ISA_IOW が Low のときはここへデータを入力する。

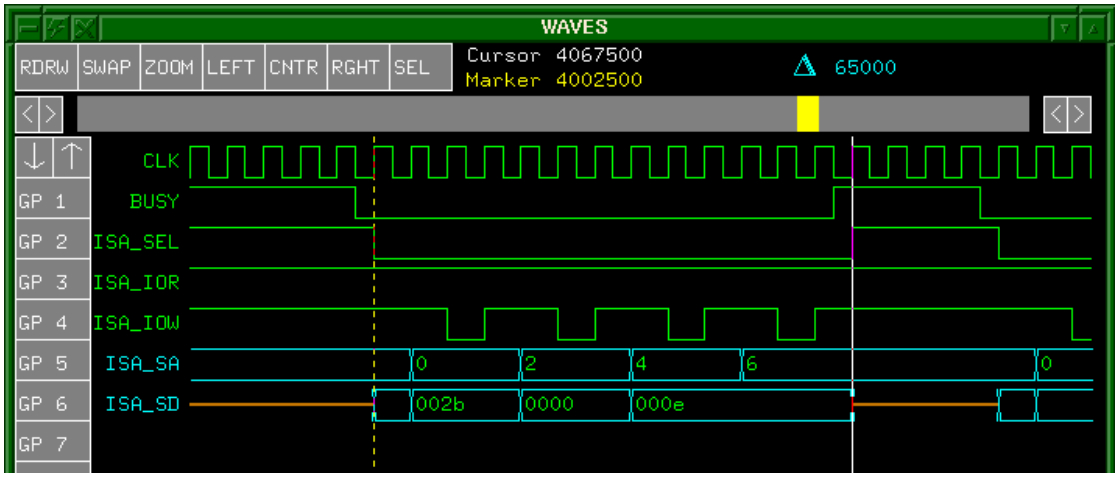
ISA バスから FADC I/F へ命令を送る方法を説明する (図 3.15 参照)。命令の送信は ISA アドレスを指定して命令番号・オプション 1・オプション 2 をデータとして書き込むことで行う。

1. ISA アドレス (ISA_SA) に 0 を設定して 16 bit のデータ (ISA_SD) の下位 8 bit に命令番号 (付録 C 参照) を書き込む。データの上位 8 bit は使わないので 0 を書き込んでおく。
2. ISA アドレス (ISA_SA) に 2 を設定して 16 bit のデータ (ISA_SD) にオプション 1 を書き込む。
3. ISA アドレス (ISA_SA) に 4 を設定して 16 bit のデータ (ISA_SD) にオプション 2 を書き込む。
4. 命令番号・オプション 1・オプション 2 を FADC I/F に書き込んだ後、ISA アドレス (ISA_SA) に 6 を設定して 16 bit のデータ (ISA_SD) に 0 を書き込むと、FADC I/F 上の FPGA が書き込まれた命令の実行を開始する。FPGA が命令を実行している間は、FPGA の BUSY ピンの出力が High になる。

次に ISA バスからデータを読み出す方法を説明する。ISA バスから読み出せるデータは 3 種類あり、それぞれ ISA アドレスの値によりデータの種類を選択することができる。ISA バスから読み出すことのできるデータを以下に説明する。

INMOS リンク出力用データ FPGA の内部には、次の 3.4.5.2 節で説明する INMOS リンクに出力するデータを記憶する 8 bit の記憶領域がある。ISA アドレスに 0 を設定して ISA バスの読み出し動作を行うと、ISA バスから INMOS リンク出力用データを読み出すことができる。ISA バスに出力されるデータは 16 bit の幅があるが、その内の下位 8 bit に INMOS リンクのデータが出力される。

データ出力用 FIFO のデータ データ出力用 FIFO に出力されたデータはイベントビルダーが自動的に読み出すことになっているが、イベントビルダーを接続せずに FADC I/F を単体で動作させる場合のことを考慮して、ISA バスから



CLK 20 MHz のクロック。

BUSY この信号に High が出力されているとき、FPGA が動作中であることを表している。

ISA_SEL ISA バスのセレクト信号が出力されている。ISA バスを使用するときはこの信号を Low に設定する。

ISA_IOR ISA バスの読み出し信号が出力されている。ISA バスから FPGA のデータを読み出すときは Low に設定する。このシミュレーションでは命令を書き込むだけなので、使用されず High のままである。

ISA_IOW ISA バスの書き込み信号が出力されている。ISA バスから FPGA へデータを書き込むときは Low に設定する。このシミュレーションでは一連の命令とオプションを書き込むので、連続して 4 回 Low に設定されている。

ISA_SA データの種類を指定する ISA アドレスが出力されている。

ISA_SD ISA バスと FPGA の間で遣り取りされるデータが出力されている。

ISA バスを使って FPGA に命令を送る様子を、シミュレーションで確認している。図を垂直に横切っている緑の点線から白線までの間が、命令を送るのに必要な 1 サイクルである。まず 1 つ目の ISA_IOW の信号で、命令番号 2b を FPGA に送っている。この 2b は 16 進数なので 10 進数に直すと 43 のことで、命令番号 43 は付録 C によれば CMD_BOARD_ORDER 命令であり、この命令で FADC モジュールを読み出す順番を設定しようとしていることが分かる。2 つ目の ISA_IOW で書き込まれているのはオプション 1 で、CMD_BOARD_ORDER 命令では FADC モジュールを読み出す順番をここで指定する。ここで FPGA に書き込まれているデータは 0 で、これは 0 番目、つまり一番最初に読み出す FADC モジュールを指定しようとしていることが分かる。3 つ目の ISA_IOW で書き込まれているのはオプション 2 で、CMD_BOARD_ORDER 命令では FADC モジュールのボード番号をここで指定する。ここで FPGA に書き込まれているデータは e で、これは 16 進数なので 10 進数に直すと 14 のことであり、つまりこの命令では一番最初に読み出す FADC モジュールのボード番号を 14 に設定している、ということが分かる。最後に ISA アドレスを 6 に設定してデータに 0 を書き込むと、FPGA は命令の実行を開始して、命令実行中は BUSY 信号が High になる。

図 3.15: ISA バスで FPGA に命令を送るシミュレーションの様子

データ出力用 FIFO のデータを読み出せるようになっている。ISA アドレスに 4 を設定して ISA バスの読み出し動作を行うと、ISA バスからデータ出力用 FIFO のデータを読み出すことができる。

データの有無を示すフラグ 先に説明した 2 つの記憶領域には、それぞれデータの有無を示す 1 bit のフラグがある。ISA アドレスに 2 を設定して ISA バスの読み出し動作を行うと、これらの 2 つのフラグを読み出すことができる。ISA バスに出力される 16 bit のデータの内下位の 2 bit の領域にこれらのフラグの値が出力され、2 bit の内の上位 1 bit が INMOS リンク出力データの有無を表すフラグであり、下位 1 bit がデータ出力用 FIFO のデータの有無を表すフラグである。2 つのフラグはいずれも、データが有るときは 1 が出力され、データが無いときは 0 が出力される。

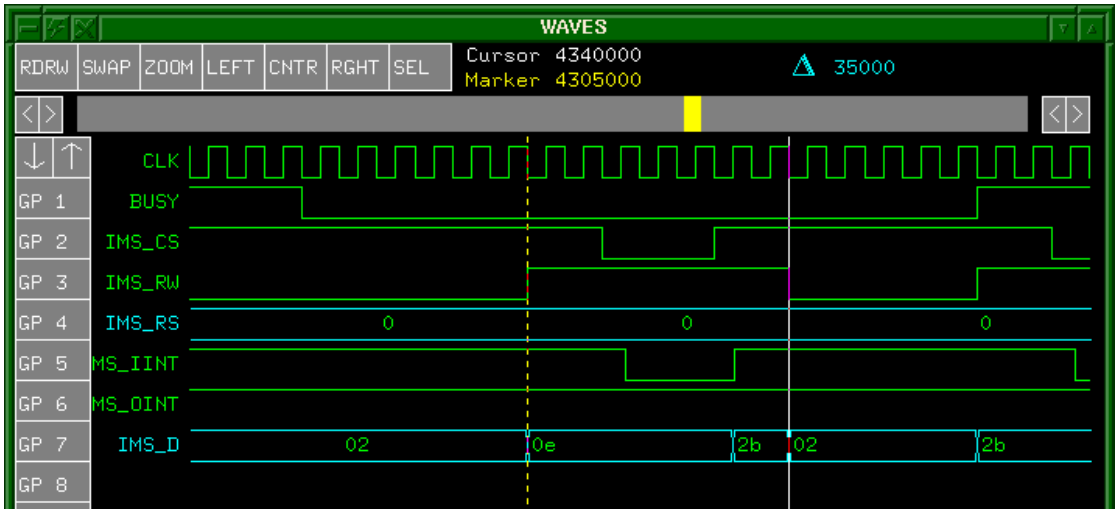
ISA バスからのデータの読み出しは、FPGA が命令を実行している最中でも実行することが可能である。例えば FPGA が FADC データの連続読み出しを実行中にデータ出力用 FIFO に出力されるデータを、連続読み出しを実行中にフラグを読み出してデータの有無を確認してからデータ出力用 FIFO に出力されたデータを読み出す、といった動作も可能である。

3.4.5.2 INMOS リンク

INMOS リンクはトランスピュータという INMOS 社の開発した並列コンピュータを接続するための I/F である。BESS のデータ収集系はトランスピュータにより管理されているので、BESS に組み込まれた状態で FADC I/F に命令を送るためには、INMOS リンクを使う必要がある。トランスピュータは現在開発・販売は中止されているが、BESS では予備のトランスピュータを確保している。

INMOS リンクは 1 bit 毎に同期して信号を送るシリアル方式の通信である。具体的な通信手順は IMS C012[10] という IC が行うので、FPGA はこの IMS C012 に対してデータの読み書きを行えば良い。IMS C012 は入出力を 8 bit 単位で行うので、FADC I/F に命令を送るときも 8 bit 単位になる。INMOS リンクを使って FADC I/F に命令を送る手順を説明する (図 3.16 参照)。

1. 8 bit の命令番号を FADC I/F に送る。
2. 16 bit のオプション 1 を、上位 8 bit と下位 8 bit の 2 つに分けて FADC I/F に送る。まず先に上位 8 bit のデータを送り、次に下位 8 bit のデータを送るようにする。
3. 16 bit のオプション 2 を、上位 8 bit と下位 8 bit の 2 つに分けて FADC I/F に送る。まず先に上位 8 bit のデータを送り、次に下位 8 bit のデータを送るようにする。
4. 8 bit の命令番号・16 bit のオプション 1・16 bit の option 2 を受け取り終えると、FADC I/F 上の FPGA は自動的に命令の実行を開始する。FPGA が命令を実行している間は、FPGA の BUSY ピンに High が出力される。



CLK 20 MHz のクロック。

BUSY この信号に High が出力されているとき、FPGA が動作中であることを表している。

IMS_CS IMS C012 のセレクト信号。INMOS リンクのデータを読み書きするとき、この信号を Low に設定する。

IMS_RW IMS C012 の Read/Write を切り替える信号。Read のときは High に設定し、Write のときは Low に設定する。このシミュレーションでは FPGA が INMOS リンクからデータを読み出しているので、High に設定されている。

IMS_RS IMS C012 へ読み書きするデータの種類を設定している。0 は INMOS リンクのデータを読み書きすることを表している。この他に IMS C012 の内部状態に関するデータを指定することができる。

IMS_IINT IMS C012 に INMOS リンクから送られて来たデータが有るとき、High になる。

IMS_OINT IMS C012 が INMOS リンクへデータを送ることが可能なとき、High になる。

INMOS リンクを使って FPGA に命令を送る様子を、シミュレーションで確認している。図を垂直に横切っている緑の点線から白線までの間が、一つのデータを送るのに必要な 1 サイクルである。INMOS リンクは ISA バスと違って 8 bit 単位でデータを送るため少し時間がかかるので、この図では一つの命令全体を送る様子を表示し切れていない。

図 3.16: INMOS リンクで FPGA に命令を送るシミュレーションの様子

ISA バスのときと同様に、FADC モジュールから読み出したデータを INMOS リンクで取り出すこともできる。この場合は FADC の連続読み出しを行う命令を送るときに、オプションでデータをデータ出力用 FIFO ではなく INMOS リンクに出力するように設定する。オプションの設定は付録 C で説明している。FADC データは 16 bit 単位で読み出されるが、INMOS リンクは 8 bit 単位でデータの入出力を行うので、FPGA は 16 bit の FADC データを 8 bit ずつ 2 つに分けて INMOS リンクに出力する。FPGA はまず先に FADC データの上位 16 bit を送り、次に FADC データの下位 16 bit を送るので、データを取り出すトランスピュータの側は、データの順番に注意して取り出さなければならない。

第4章　まとめ

4.1　将来の課題

FADC I/F は現在、FPGA の内部回路の設計とテスト用の基板の実装が修了して、動作テストを行っている最中である。FADC I/F に接続する FADC モジュールは、入力された信号をデジタル情報に変換するアナログサブモジュールがまだ動作テストを行っている最中で完全ではないが、メインボード上のテストデータ用 FIFO を使用することによりデータ圧縮 IC と FADC I/F の動作テストを行うことは可能である。

FADC I/F で最も重要なのは全 FADC チャンネルのデータを読み出すのに必要な時間で、この時間は直接、トリガーのデッドタイムとして現れてくる。現在のデータ圧縮モジュールがデータの収集・圧縮に必要とする時間は約 200 μ second であり、デッドタイムは約 10 %である。参考までに 1997 年に BESS のデータ収集系に改良が加えられて現在のシステムになる以前の、データ収集の時間とデッドタイムを述べておくと、データ収集時間は 1.2 m second でありデッドタイムは 40 %であった。

3 章の 3.4.3.1 節で説明したように FADC チャンネルのデータの読み出しに必要な遅延が予想以上に大きかったため、新しい FADC システムのための FADC I/F では残念ながら、全 FADC チャンネルのデータ読み出しに必要な時間は現行のデータ圧縮モジュールよりも増加してしまう見込みである。FADC I/F はまだ FADC データ連続読み出しの動作テストには至っていないが、BESS が動作中に FADC チャンネルが出力する典型的なデータの量は、1 チャンネルにつき平均 0.5 クラスタなので、次のようにして計算できる。

1 クラスタのデータ量 BESS で主に使用されるのは 2 章の 2.4.1.1 節で説明した SUM モードなので、8 byte である。

FADC チャンネルから一回のアクセスで読み出せるデータの量 2 byte 単位で読み出される。

FADC チャンネルへ一回アクセスするのに必要な時間 データ圧縮 IC へ送る STRB 信号の幅は 2 クロックだが、Low に下げた STRB 信号を High に戻さなければならないので、合計で 2 倍の 4 クロックが必要である。1 クロックは 20 MHz だから 50 nano second なので、したがって $50 \times 4 = 200$ nano second の時間が必要になる。

FADC チャンネルの切り替えに必要な時間 3 章の 3.4.3.3 節で説明したように、FADC チャンネルを切り替えた直後は、通常の遅延に加えて 1 クロック余分に待つ

必要がある。ただしここではemptyフラグを見るだけなのでSTRB信号を送る必要は無く、したがってデータの遅延である2クロックに1クロックを加えて合計で3クロック待てばよい。したがって $50 \times 3 = 150$ nano secondの時間が必要になる。

全FADCチャンネルの数 新しいFADCシステムでは最大1024チャンネルが接続可能であが、現行のFADCシステムでは最大512チャンネルなので比較のため512で計算する。

これらの情報を総合すると以下のような結果になる。

cl. :クラスタ

ch. :FADC チャンネル

ns :nano second

$$\left(0.5 \text{ cl./ch.} \times \frac{8 \text{ byte/cl.}}{2 \text{ byte/read}} \times 200 \text{ ns/read} + 150 \text{ ns/ch.}\right) \times 512 \text{ ch.}$$
$$= 281.6 \times 10^3 \text{ ns} \simeq 280 \ \mu \text{second}$$

つまり新しいFADCシステムではデータの読み出し時間が現在の200 μ secondより80 μ second増えてしまう見込みである。さらに将来ODCが増設されてFADCチャンネルの最大数である1024近くまでチャンネル数が増えることを考えると、データの読み出し時間は倍の560 nano second必要になり、現在よりも360 nano secondも増えてしまう。

一方、本来予定していた通りデータの読み出しに必要な遅延が1クロックだった場合は、FADCチャンネルのデータに一回アクセスするのに必要な時間は $1 \times 2 = 2$ クロックの100 nano secondになり、FADCチャンネルの切り替えに必要な時間は $1 + 1 = 2$ クロックの100 nano secondになるので、全データの読み出しに必要な時間は次のようになる。

$$\left(0.5 \text{ cl./ch.} \times \frac{8 \text{ byte/cl.}}{2 \text{ byte/read}} \times 100 \text{ ns/read} + 100 \text{ ns/ch.}\right) \times 512 \text{ ch.}$$
$$= 153.6 \times 10^3 \text{ ns} \simeq 150 \ \mu \text{second}$$

したがって全データの読み出しには150 nano second必要であり、これは現行のFADCシステムより50 nano second速いことになる。またFADCチャンネルの数が最大数の1024まで増えても、現行のFADCシステムより100 nano second遅れるだけで済む。

現在テスト用に製作している新しいFADCモジュールとそのためのFADC I/Fでは、FADCチャンネルデータ読み出しの遅延が1クロックだと間に合わない可能性があるため、データを確実に読み出せるようにするため遅延時間を2クロック以上にしている。しかしBESSに搭載するためのFADCモジュールとFADC I/Fでは、FADCモジュール側のチャンネル選択回路により高速なICを使用し、FADC I/Fに使用するFPGAもより高性能なものを使用することで、遅延時間を1クロックに改善できる可能性がある。したがってFPGA内部の回路は1クロックの遅延でも

FADC チャンネルのデータを読み出せるようにしておくことが望ましいが、現在の読み出し回路は 3 章の 3.4.3.1 節で説明したように遅延が 2 クロック以上の場合にしか対応できていない。したがって 1 クロックの遅延でデータ読み出しを可能にしておくことは、将来の課題である。FPGA 内部の回路はモジュール化されており、データ圧縮 IC の RAM を読み出す部分回路は他の回路に影響を与えずに交換することが可能である。

4.2 まとめ

FPGA を使用することで、大規模で複雑な動作を行う回路を約一年という開発期間で設計することができた。設計を行っている間や設計を終えて動作テストを行っている現在でも、何度か仕様変更の必要が生じているが、FPGA は内部回路の修正が容易なため柔軟に対処することができた。FPGA の外部入出力の割当を固定したまま内部回路を変更すると、ゲートの配置配線の最適化が不十分になるというペナルティがあるが、高性能な FPGA を使用して性能にはある程度の余裕を持たせてあるので、小規模な変更なら問題はない。最後に FADC I/F のために設計した FPGA 内部回路の仕様を表 4.1 に示す。

使用した FPGA	
製品名	Xilinx XC4028XL PG299
CLB(セル) 数	1024
ゲート 数	18000 から 50000 に対応
外部入出力の数	299

FPGA 内部回路の規模	
Verilog-HDL ソース	13645 行
Module の数	98
使用する CLB(セル) の数	925
最大ロジックレベル	6 レベル
外部入出力の数	150

動作速度		
最大遅延時間	(ゲート のみ)	18.466 ns (54.154 MHz)
	(配線含む)	39.140 ns (25.549 MHz)
予定動作速度		50 ns (20 MHz)

ns: nano second

表 4.1: FPGA 内部回路の仕様

謝辞

本研究開発を進めるにあたって直接御指導下さった野崎光昭先生、東京大学の折戸周治先生に深く感謝致します。また同じ FADC システムの開発担当として松井長隆氏と安楽和明氏には非常にお世話になりました。東京大学の井森正敏先生には、多くの助言をして戴きました。

武田先生、川越先生をはじめとする神戸大学の皆様、特に竹内彰氏と塚原知宏氏には研究以外のことについて非常にお世話になりました。心より感謝致します。吉村氏をはじめとする BESS グループの皆様や ICEPP の皆様、その他ここには挙げられなかった皆様、本当にありがとうございました。

参考文献

[1] ANRAKU K., IMORI M., SAEKI T., UEDA I., *et al.*
A Flash ADC System with Fast Data Compression for a Balloon-borne Experiment
IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 39, NO. 4
1992 年

[2] 元木 正和
修士論文
宇宙起源反粒子探索実験に用いるドリフトチェンバーの開発と性能評価
神戸大学大学院理学研究科物理学専攻課程
1993 年

[3] 松井 長隆
修士論文
FADCデータ高速収集装置の製作
東京大学大学院理学系研究科物理学専攻折戸研究室
1997 年

[4] 前野 忠嗣
修士論文
飛跡判別機能を備えた高速データ収集システムの開発
東京大学大学院理学系研究科物理学専攻折戸研究室
1997 年

[5] 桜井 至
HDL によるデジタル設計の基礎
テクノプレス
1997 年

[6] 高速、ビデオ差動アンプ
AD830
アナログ・デバイセズ株式会社

[7] 300 MHz, 1 mA Current Feedback Amplifier
AD8011
ANALOG DEVICES

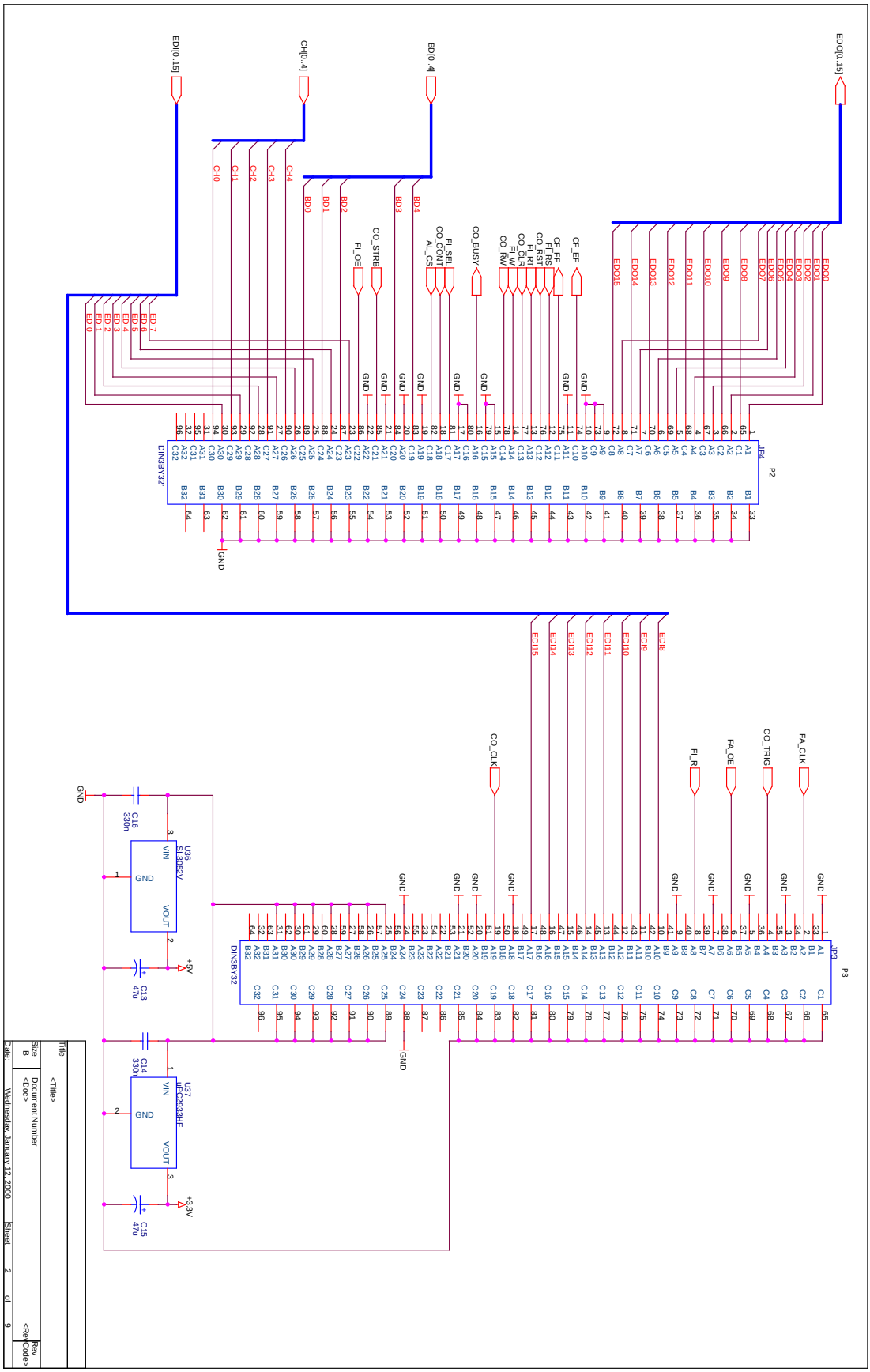
[8] 8-Bit 20 MSPS, 60 mW Sampling A/D Converter
AD775
ANALOG DEVICES

[9] IMS C011 link adaptor
inmos
1987 年

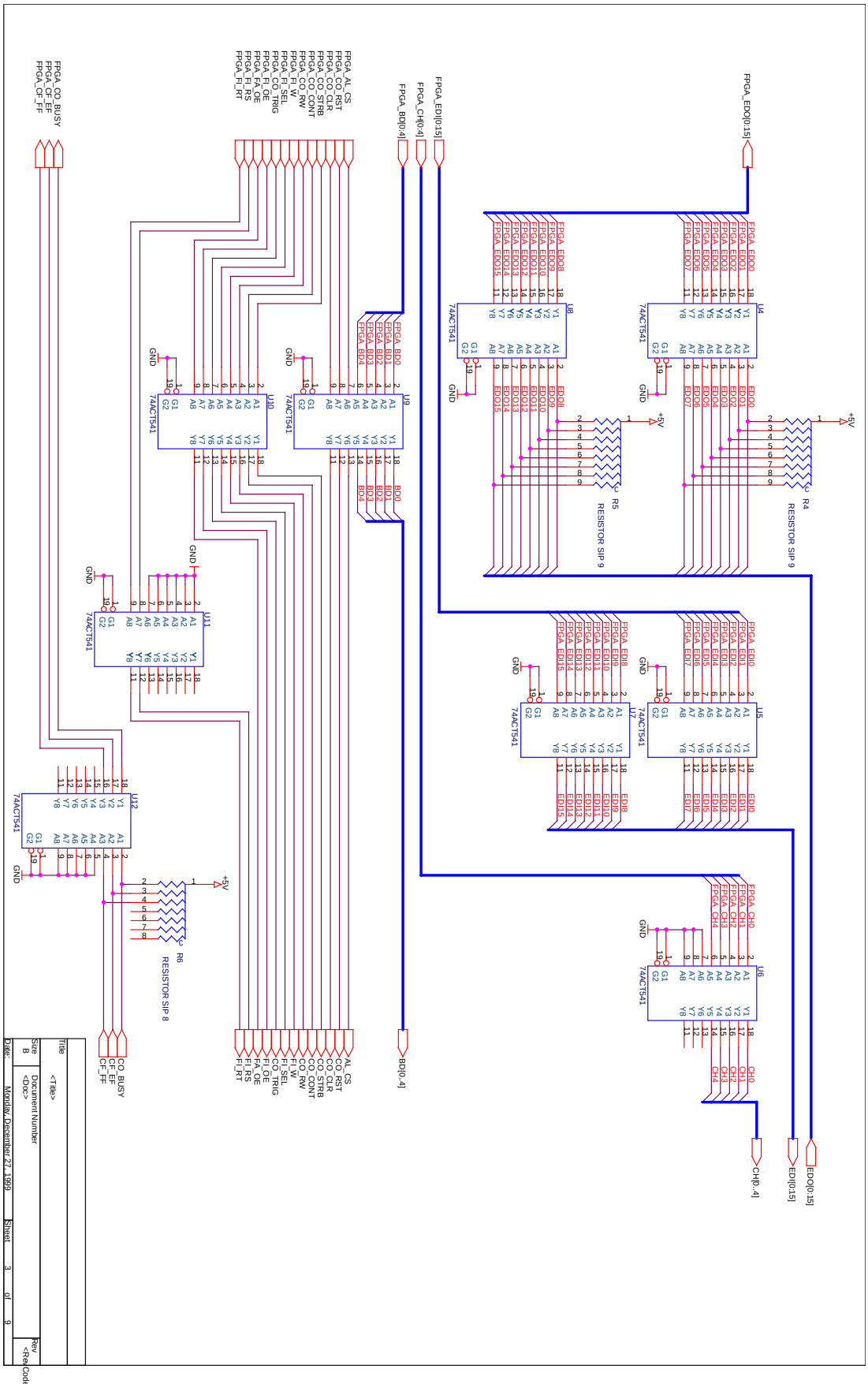
[10] IMS C012 link adaptor
inmos
1987 年

付 録A　FADC クレートコントローラの回路図

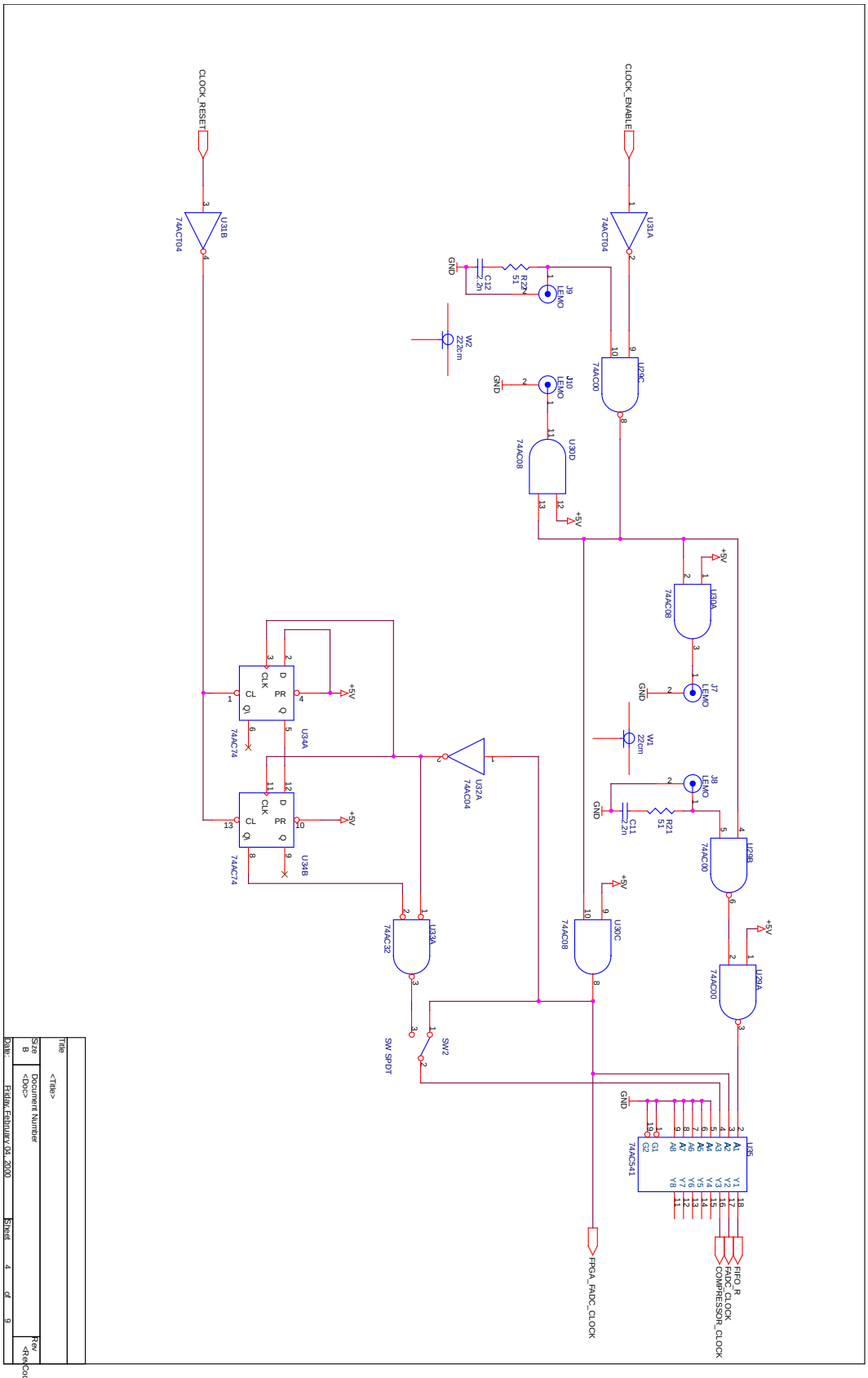
A.1　バックプレーン (FADC モジュール側)



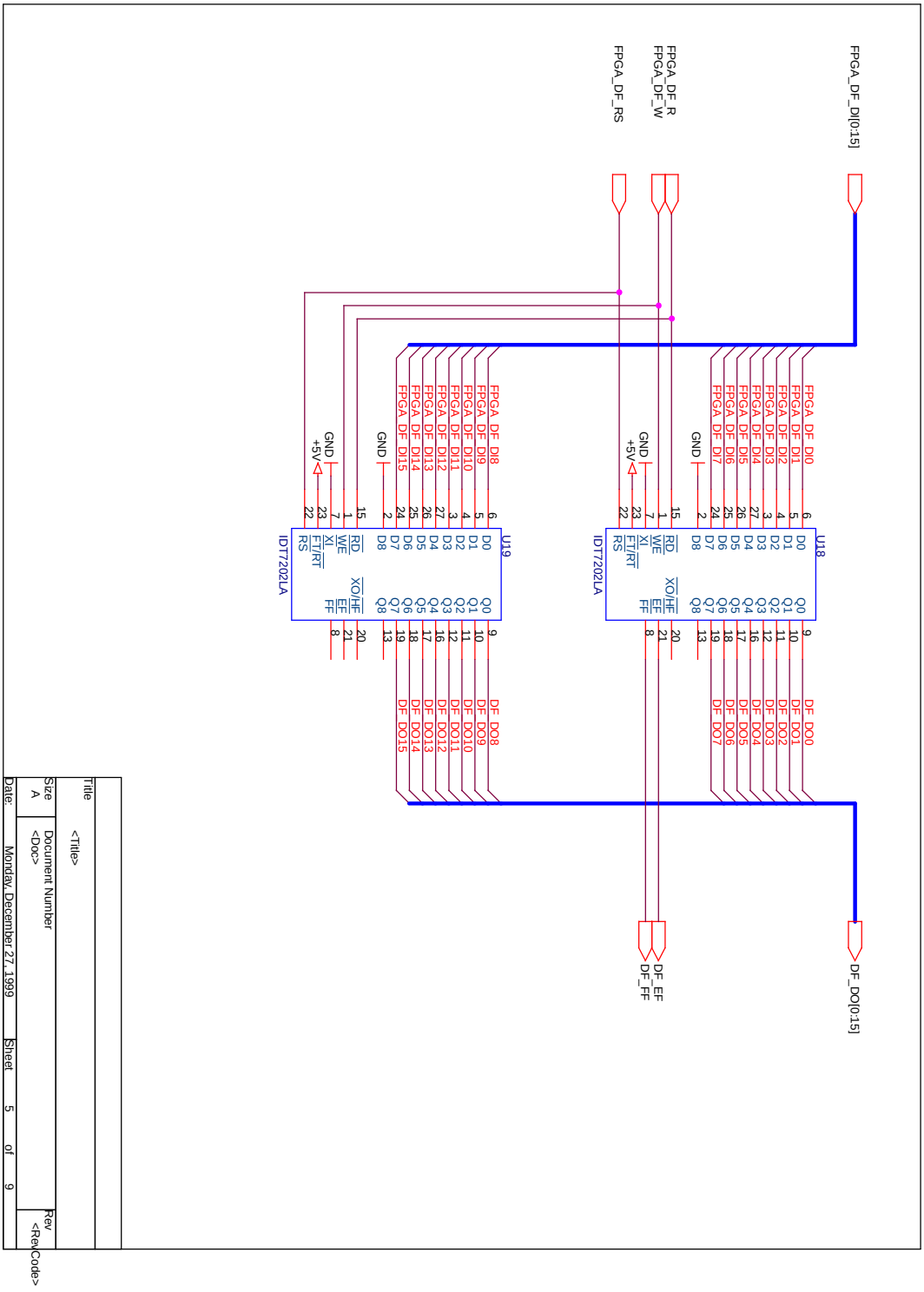
A.2　バックプレーン (FADC I/F 側)



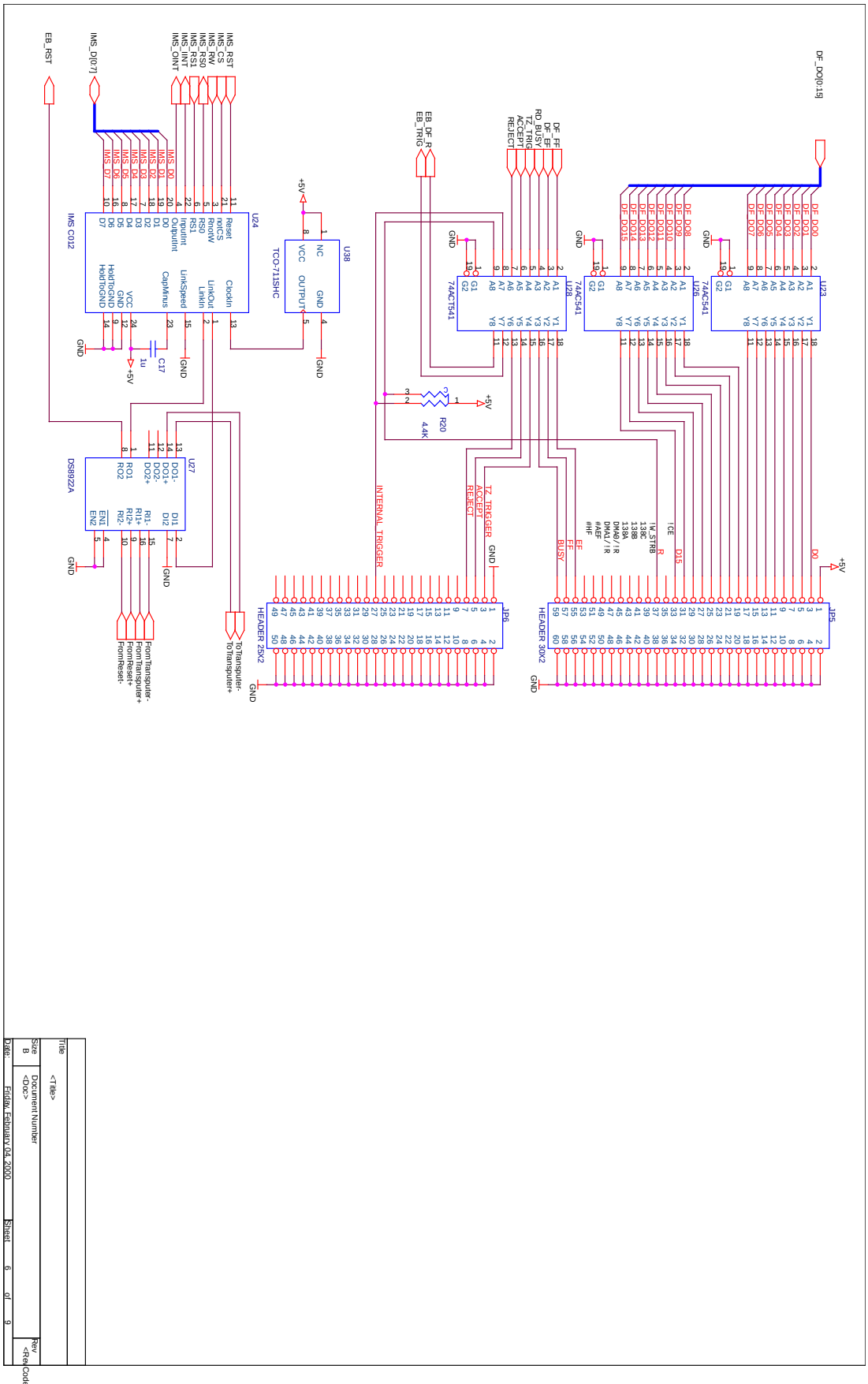
A.3　クロック生成


CLOCK_RESET
CLOCK_ENABLE
U31B
74ACT04
U31A
74ACT04
C12
2.2n
R22
51
J9
LEMO
GND
W2
22cm
J10
LEMO
U29C
74AC00
U30D
74AC08
+5V
U30A
74AC08
J7
LEMO
W1
22cm
J8
LEMO
C11
2.2n
R21
51
U29B
74AC00
U34A
74AC74
U34B
74AC74
U32A
74AC04
U32A
74AC32
U30C
74AC08
U29A
74AC00
SW2
SW SPDT
U35
74AC541
FPGA_FADC_CLOCK
FIFO_R
FADC_CLOCK
COMPRESSOR_CLOCK
Title
<Title>
Size
B
Document Number
<DOC>
Rev
<RevCode>
Date:
Friday, February 04, 2000
Sheet
4
of
9

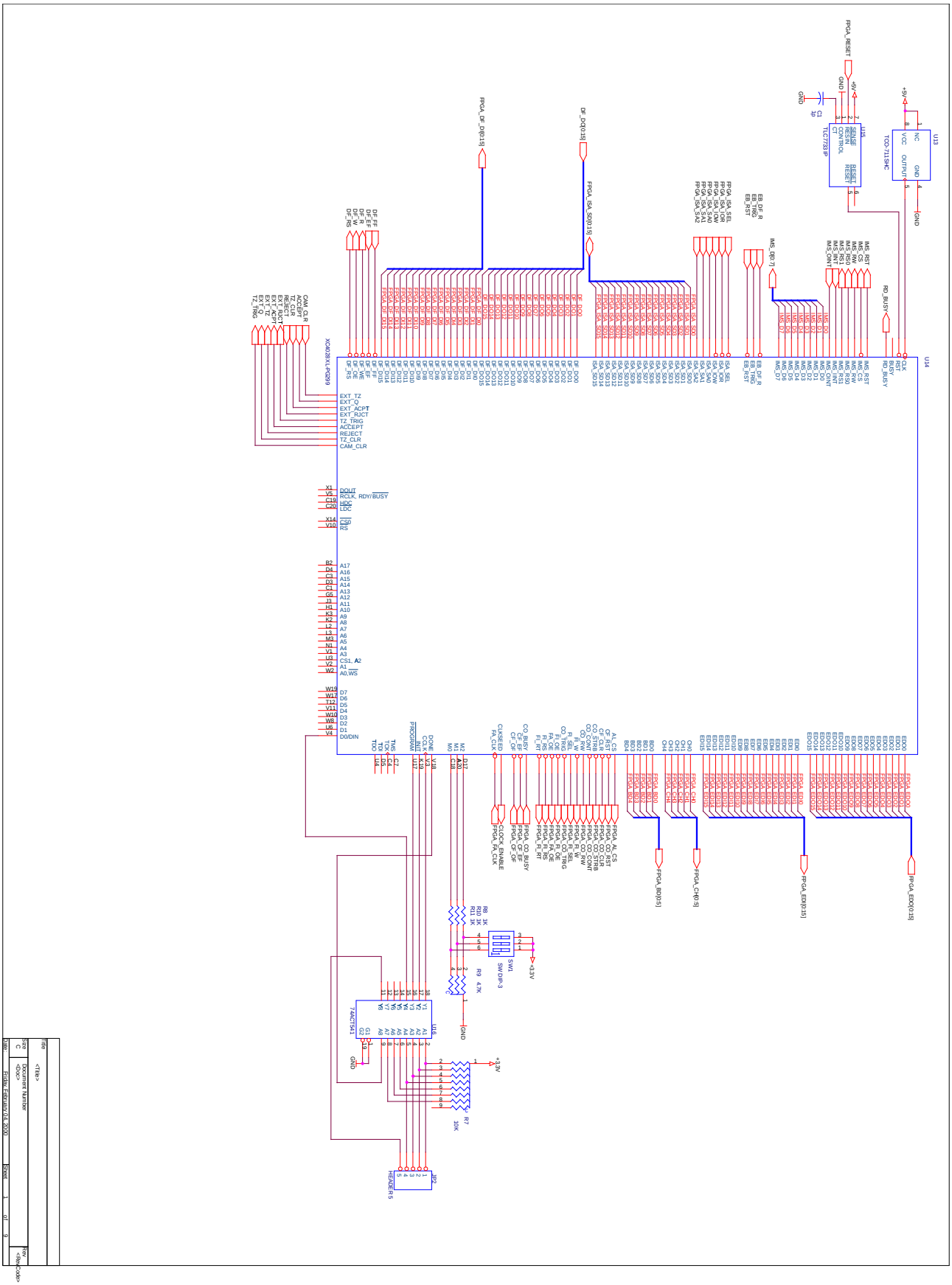
A.4　データ出力用FIFO周辺



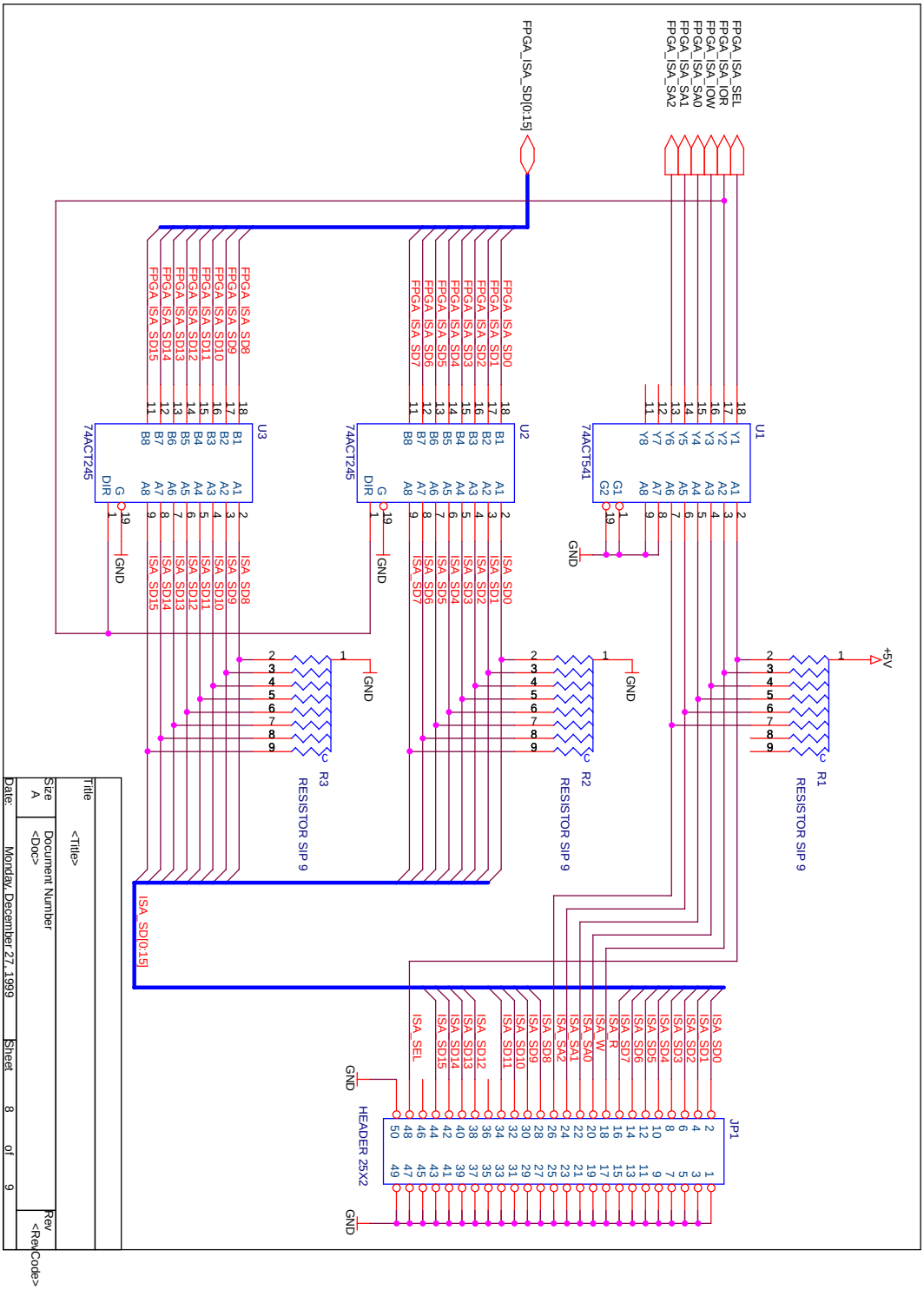
A.5　イベントビルダーとの接続



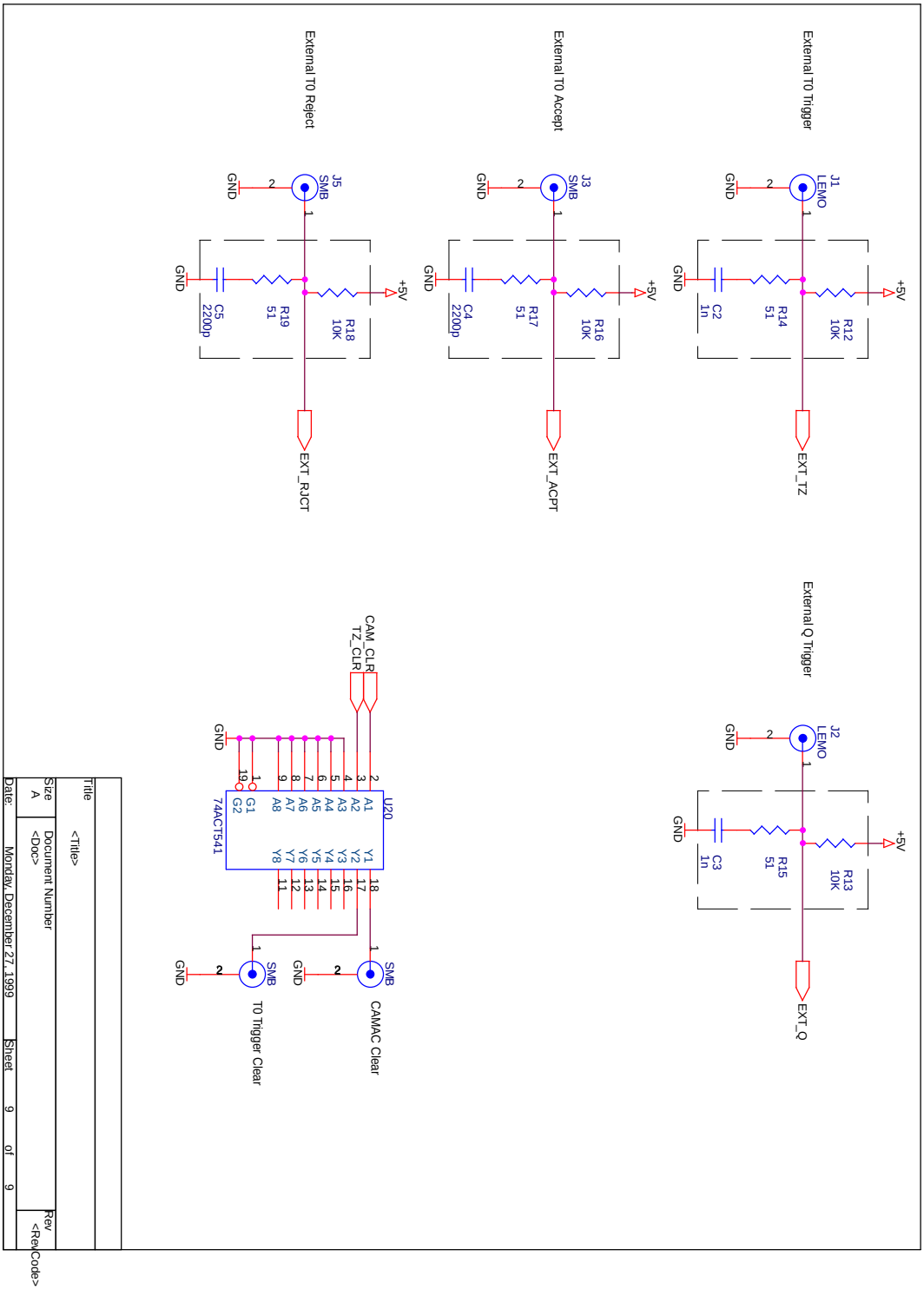
A.6 FPGA 周边



A.7　ISAバスとの接続



A.8　トリガーの入出力


External T0 Trigger
J1
LEMO
GND
C2
1n
R14
51
R12
10K
+5V
EXT_TZ
External T0 Accept
J3
SMB
C4
2200p
R17
R16
EXT_ACPT
External T0 Reject
J5
C5
R19
R18
EXT_RJCT
External Q Trigger
J2
C3
R15
R13
EXT_Q
CAM_CLR
TZ_CLR
U20
74ACT541
T0 Trigger Clear
CAMAC Clear
Title
<Title>
Size
A
Document Number
<Doc>
Rev
<RevCode>
Date:
Monday, December 27, 1999
Sheet
9
of
9

付 録B　FPGAの外部入出力ポートの一覧

注意

- in/outは双方向の入出力を表している。
- IMSは IMS C012 に接続することを示している。
- ISAは ISA バスに接続することを示している。
- EBはイベントビルダーに接続することを示している。
- CLKは外部クロック生成回路に接続することを示している。
- DFはデータ出力用 FIFO に接続することを示している。
- BPはバックプレーンに接続することを示している。

名前	入出力	ACTIVE	説明
CLK	input	↓	20 MHz のクロックを入力する。
RST	input	Low	非同期リセット。
BUSY	output	High	FPGA が命令実行中であることを示す。
RD_BUSY	output	High	FPGA が FADC データを読み出し中であることを示す。
IMS_RST	output	High	IMS: IMS C012 のリセット。
IMS_CS	output	Low	IMS: IMS C012 の Chip Select。
IMS_RW	output	High/Low	IMS: IMS C012 の Read/Write。
IMS_RS0	output		IMS: 以下の 0 – 1 は、IMS C012 の Register Select。
IMS_RS1	output		
IMS_IINT	input	High	IMS: IMS C012 の Input Interrupt。
IMS_OINT	input	High	IMS: IMS C012 の Output Interrupt。
IMS_D0	in/out		IMS: 以下の 0 – 7 は、IMS C012 の Data。
IMS_D1	in/out		
IMS_D2	in/out		
IMS_D3	in/out		

名前	入出力	ACTIVE	説明
IMS_D4	in/out		
IMS_D5	in/out		
IMS_D6	in/out		
IMS_D7	in/out		
EB_DF_R	input		EB: Data Fifo Read
EB_TRIG	input		EB: Trigger
EB_RST	input		EB: Reset
ISA_SA0	input		ISA: 以下の 0－4 は、ISA バスのアドレス。
ISA_SA1	input		
ISA_SA2	input		
ISA_SD0	in/out		ISA: 以下の 0－15 は、ISA バスのデータ。
ISA_SD1	in/out		
ISA_SD2	in/out		
ISA_SD3	in/out		
ISA_SD4	in/out		
ISA_SD5	in/out		
ISA_SD6	in/out		
ISA_SD7	in/out		
ISA_SD8	in/out		
ISA_SD9	in/out		
ISA_SD10	in/out		
ISA_SD11	in/out		
ISA_SD12	in/out		
ISA_SD13	in/out		
ISA_SD14	in/out		
ISA_SD15	in/out		
ISA_SEL	input	Low	ISA: ISA バスの select 信号。
ISA_IOR	input	Low	ISA: ISA バスの read 信号 (FPGA outut)。
ISA_IOW	input	Low	ISA: ISA バスの write 信号 (FPGA input)。
DF_DO0	input		DF: 以下の 0－15 は、データ出力用 FIFO の出力 (FPGA input)。
DF_DO1	input		
DF_DO2	input		
DF_DO3	input		
DF_DO4	input		
DF_DO5	input		
DF_DO6	input		
DF_DO7	input		
DF_DO8	input		

名前	入出力	ACTIVE	説明
DF_DO9	input		
DF_DO10	input		
DF_DO11	input		
DF_DO12	input		
DF_DO13	input		
DF_DO14	input		
DF_DO15	input		
DF_DI0	output		DF: 以下の 0 – 15 は、データ出力用 FIFO の入力 (FPGA output)。
DF_DI1	output		
DF_DI2	output		
DF_DI3	output		
DF_DI4	output		
DF_DI5	output		
DF_DI6	output		
DF_DI7	output		
DF_DI8	output		
DF_DI9	output		
DF_DI10	output		
DF_DI11	output		
DF_DI12	output		
DF_DI13	output		
DF_DI14	output		
DF_DI15	output		
DF_FF	input	Low	DF: データ出力用 FIFO の full フラグ。
DF_EF	input	Low	DF: データ出力用 FIFO の empty フラグ。
DF_WE	output	Low	DF: データ出力用 FIFO の Write Enable。
DF_OE	output	Low	DF: データ出力用 FIFO の Read Enable。
DF_RS	output	Low	DF: データ出力用 FIFO のリセット。
EXT_TZ	input	↑	External T0 Trigger
EXT_Q	input	↑	External Q Trigger
EXT_ACPT	input	↑	External Accept
EXT_RJCT	input	↑	External Reject
TZ_TRIG	output	↑	T0 Trigger
ACCEPT	output	High	Accept Trigger
REJECT	output	High	Reject Trigger
TZ_CLR	output	Low	T0 Trigger Clear
CAM_CLR	output	Low	CAMAC Clear
CLKSEED	output	Low	CLK: 外部クロック生成回路の Enable。

名前	入出力	ACTIVE	説明
FA_CLK	input	↓	BP: FADC クロック。
CH0	output		BP: 以下の 0–5 は、チャンネル番号。
CH1	output		
CH2	output		
CH3	output		
CH4	output		
BD0	output		BP: 以下の 0–5 は、ボード番号。
BD1	output		
BD2	output		
BD3	output		
BD4	output		
EDO0	input		BP: 以下の 0–16 は、データ圧縮 IC のデータ出力 (FPGA input)。
EDO1	input		
EDO2	input		
EDO3	input		
EDO4	input		
EDO5	input		
EDO6	input		
EDO7	input		
EDO8	input		
EDO9	input		
EDO10	input		
EDO11	input		
EDO12	input		
EDO13	input		
EDO14	input		
EDO15	input		
EDI0	output		BP: 以下の 0–16 は、データ圧縮 IC のデータ入力 (FPGA output)。
EDI1	output		
EDI2	output		
EDI3	output		
EDI4	output		
EDI5	output		
EDI6	output		
EDI7	output		
EDI8	output		
EDI9	output		

名前	入出力	ACTIVE	説明
EDI10	output		
EDI11	output		
EDI12	output		
EDI13	output		
EDI14	output		
EDI15	output		
AL_CS	output	High	BP: 全 FADC チャンネルの Select。
CO_RST	output	Low	BP: データ圧縮 IC のリセット。
CO_CLR	output	Low	BP: データ圧縮 IC のクリア。
CO_STRB	output	Low	BP: データ圧縮 IC の非同期 Read/Write 信号 (Strobe)。
CO_CONT	output	Low	BP: データ圧縮 IC の Control Mode。
CO_RW	output	High/Low	BP: データ圧縮 IC の Read/Write Select。
FI_W	output	Low	BP: テストデータ用 FIFO の Write。
FI_SEL	output	High	BP: テストデータ用 FIFO の Select。
CO_TRIG	output	Low	BP: データ圧縮 IC の Trigger。
FI_OE	output	Low	BP: テストデータ用 FIFO の Output Enable。
FA_OE	output	Low	BP: FADC の Output Enable。
FI_RS	output	Low	BP: テストデータ用 FIFO のリセット。
FI_RT	output	Low	BP: テストデータ用 FIFO の Retransmit。
CO_BUSY	input	High	BP: データ圧縮 IC が稼働中であることを示す。
CF_EF	input	Low	BP: データ圧縮 IC かまたはテストデータ用 FIFO の empty フラグ。
CF_OF	input	Low	BP: データ圧縮 IC のオーバーフローフラグかまたはテストデータ用 FIFO の full フラグ。
RSV_I0	input		BP: 以下の 0－1 は、将来の拡張に備えて予約されている。
RSV_I1	input		BP:
RSV_O0	output		BP: 以下の 0－1 は、将来の拡張に備えて予約されている。
RSV_O1	output		BP:

付 録C　FADC I/F命令の一覧

注意

- オプションは常に 16 bit の固定長である。
- オプションのパラメータ全体の大きさが 16 bit に満たないときは、パラメータを下位 bit に詰めて、余った上位 bit は 0 で埋める。
- オプションで複数のパラメータを設定するときは、パラメータの bit の中で先に書いてある順に上位 bit から詰めて行く。

名前	番号	オプション 1	オプション 2	戻り値	説明
CMD_NULL	0				無動作。
CMD_AL_CS	1	High/Low			AL_CS ピンの出力を High/Low に設定する。
CMD_CF_RST	2	High/Low			CF_RST ピンの出力を High/Low に設定する。
CMD_CF_CLR	3	High/Low			CF_CLR ピンの出力を High/Low に設定する。
CMD_CO_STRB	4	High/Low			CO_STRB ピンの出力を High/Low に設定する。
CMD_CO_CONT	5	High/Low			CO_CONT の出力を High/Low に設定する。
CMD_CO_RW	6	High/Low			CO_RW の出力を High/Low に設定する。
CMD_FI_W	7	High/Low			FI_W ピンの出力を High/Low に設定する。
CMD_FI_SEL	8	High/Low			FI_SEL ピンの出力を High/Low に設定する。
CMD_BD_CH	9	ボード番号 (5 bit), チャンネル番号 (5 bit): 10 bit			BD ピンとCH ピンの出力の値を設定する。
CMD_EDI	10	data: 16 bit			EDI[0－15] ピンの出力の data の値に設定する。
CMD_DUMMY11	11				番号埋めのダミー命令。

名前	番号	オプション 1	オプション 2	戻り値	説明
CMD_DUMMY12	12				番号埋めのダミー命令。
CMD_DUMMY13	13				番号埋めのダミー命令。
CMD_DUMMY14	14				番号埋めのダミー命令。
CMD_DEFAULT	15				バックプレーンに出力しているピンの値をデフォルト値に戻す。
CMD_CF_RST_P	16	width: 4 bit			CF_RST ピンの出力を width 個のクロックの間 Low に下げる。
CMD_CF_CLR_P	17	width: 4 bit			CF_CLR ピンの出力を width 個のクロックの間 Low に下げる。
CMD_CO_STRB_P	18	width: 4 bit			CO_STRB ピンの出力を width 個のクロックの間 Low に下げる。
CMD_FI_W_P	19	width: 4 bit			FI_W ピンの出力を width 個のクロックの間 Low に下げる。
CMD_RESET	20				CF_RST ピンで FADC モジュールをリセットする。
CMD_CLEAR	21				CF_CLR ピンで FADC モジュールをクリアする。
CMD_SET_THRESHOLD	22	threshold: 8 bit	all (1 bit), ボード番号 (5 bit), チャンネル番号 (5 bit): 11 bit		全 FADC(all) またはボード・チャンネル番号の FADC の THRESHOLD レジスタに threshold の値を設定する。
CMD_SET_CHANNEL	23	channel: 8 bit	all (1 bit), ボード番号 (5 bit), チャンネル番号 (5 bit): 11 bit		全 FADC(all) またはボード・チャンネル番号の FADC の CHANNEL レジスタに channel の値を設定する。
CMD_SET_MODE	24	mode: 8 bit	all (1 bit), ボード番号 (5 bit), チャンネル番号 (5 bit): 11 bit		全 FADC(all) またはボード・チャンネル番号の FADC の MODE レジスタに mode の値を設定する。
CMD_WRITE_COMPRESSOR	25	data: 16 bit	all (1 bit), ボード番号 (5 bit), チャンネル番号 (5 bit): 11 bit		全 FADC(all) またはボード・チャンネル番号の FADC の RAM に data の値を書き込む。

名前	番号	オプション 1	オプション 2	戻り値	説明
CMD_READ_COMPRES-SOR	26		ボード番号 (5 bit), チャンネル番号 (5 bit): 10 bit	data: 16 bit	ボード・チャンネル番号の FADC の RAM からデータを 1 つ取り出す。
CMD_READ_FLAG	27		ボード番号 (5 bit), チャンネル番号 (5 bit): 10 bit	flag: 8 bit	ボード･チャンネル番号の FADC から flag を取り出す。
CMD_READ_DATA	28	Q threshold (1 bit), ボード毎の圧縮モードを使用 (1 bit)[1], 全ボードの圧縮モード (2 bit)[2], INMOS リンクにデータを出力 (1 bit) [3], 指定したチャンネルだけを読み出す (1 bit)[4]: 6 bit	ボード番号 (5 bit), チャンネル番号 (5 bit)[5]: 10 bit		全 FADC のデータを連続して読み出す。
CMD_WRITE_TFIFO	29	data: 8 bit			テスト用 FIFO に data の値を書き込む。
CMD_ENABLE_TFIFO	30	High/Low			テスト用 FIFO を有効 (High)/無効 (Low) に設定する。[6]
CMD_ENABLE_TRIGGER	31	High/Low			外部トリガーの入力を有効 (High)/無効 (Low) に設定する。
CMD_TRIG-GER	32	Q: 1 bit			内部 (Q) トリガーを発生させる。
CMD_CLOCK	33	High/Low			FA_CLK ピンからの FADC 駆動クロックの強制的な発生を有効 (High)/無効 (Low) に設定する。
CMD_CLOCK_COUNT	34	count: 9 bit			トリガー時に FA_CLK ピンから自動的に出力される FADC 駆動クロックの数を count 個に設定する。

[1] このフラグを High に設定すると、FPGA 内部に記憶されたボード毎の FADC データ圧縮モードを参照して FADC のデータを加工する。

[2] ボード毎の圧縮モードを使用するフラグが High のときは無視され、Low のときは全ボードの FADC データ圧縮モードとして使われる。

[3] このフラグが Low のとき読み出したデータがイベントビルダーに接続された FIFO へ送られ、High のときは INMOS リンクを通じてトランスピュータへ送られる。

[4] このフラグが Low のとき全 FADC チャンネルを連続して読み出し、Low のときはオプション 2 で指定した 1 つの FADC チャンネルだけ読み出す。

[5] オプション 1 の指定したチャンネルだけを読み出すフラグに Low を設定したときは無視される。

[6] テスト用 FIFO が有効 (High) なときは FADC の RAM が無効になり、逆にテスト用 FIFO が無効 (Low) なときは FADC の RAM が有効になる。

名前	番号	オプション 1	オプション 2	戻り値	説明
CMD_ENABLE_FAST_CLEAR	35	High/Low			外部 fast clear の入力を有効 (High)/無効 (Low) に設定する。
CMD_FAST_CLEAR	36				内部 fast clear を発生させる。
CMD_ENABLE_AUTO	37	High/Low			トリガー後の CMD_READ_DATA 命令の自動実行を有効 (High)/無効 (Low) に設定する。
CMD_AUTO_OPTION	38	CMD_READ_DATA 命令のオプション 1 と同じ。			CMD_READ_DATA 命令を自動実行するときに使われるオプション 1 の値を設定する。
CMD_SET_BOARD_NUMBER	39	number: 5 bit			CMD_READ_DATA 命令で連続して読み出す FADC ボードの枚数を number 枚に設定する。
CMD_SET_QTHRESHOLD	40	threshold: 4 bit			CMD_READ_DATA 命令で Q threshold が有効なときに使われる値を threshold に設定する。
CMD_KILL_CHANNEL	41	High/Low	all (1 bit), ボード番号 (5 bit), チャンネル番号 (5 bit): 11 bit		CMD_READ_DATA 命令で特定のボード・チャンネル番号の FADC を読み跳ばすフラグを設定する。有効 (High) ならば読み跳ばされ、無効 (Low) ならデータを読み出される。
CMD_READ_KILL	42		ボード番号 (5 bit), チャンネル番号 (5 bit): 10 bit	High/Low	ボード・チャンネル番号の CMD_READ_DATA 命令実行時の読み跳ばしフラグの値を取り出す。
CMD_BOARD_ORDER	43	order: 5 bit	board: 5 bit		CMD_READ_DATA 命令で order 番目に読み出すボード番号を board の値に設定する。

名前	番号	オプション 1	オプション 2	戻り値	説明
CMD_READ_ ORDER	44	order: 5 bit		board: 5 bit	CMD_READ_ DATA 命令で order 番目に読み出すボード番号を取り出す。
CMD_BOARD_ PATCH	45	mode: 2 bit	all (1 bit), ボード番号 (5 bit): 6 bit		CMD_READ_ DATA 命令でボード毎の FADC データ圧縮モードを参照するときに、ボード番号の圧縮モードとして mode の値を設定する。
CMD_READ_ BPATCH	46		board: 5 bit	mode: 2 bit	CMD_READ_ DATA 命令でボード毎の FADC データ圧縮モードを参照するときの、ボード番号の圧縮モードを取り出す。
CMD_FPGA_ CLEAR	47				FPGA 内部を clear する。

付 録D　CVSレポジトリのタグ一覧

ここでは FADC I/F の Verilog-HDL ソースコードの変更履歴を保存した CVS repository に付けられた tag について説明する。CVS についての詳細は、3 章の 3.3.3 節で説明している。

bess CVS システムに予約されている vendor tag。

start 初期状態のソースに付けられた tag。

toki111 FADCデータをデータ圧縮 IC の圧縮モードに応じて適切に加工する回路を実装して、実際にうまく動作するか試してみるために本流から分岐した branch に付けられた tag。現在ではこの branch の機能は本流のソースに merge されている。

toki111-nodelayed_empty “toki111” の branch に 3 章の 3.4.3.3 節で説明した外部信号高速読み込み回路を実装するときに、それ以前のソースを参照できるように目印として付けた tag。

snapshot-19990726 次の “snapshot-19990726-pipeline” の branch を分岐するときに、目印として本流のソースに付けた tag。

snapshot-19990726-pipeline FADCデータの連続読み出しの動作を高速化するために、パイプラインのステージを細かく分割した回路を実装してうまく動作するか試してみるための、実験的な branch の tag。FPGA 内部の RAM の読み出しをパイプライン化してみたが、使用するゲート数が多過ぎて配置配線できなくなってしまった失敗作。

付 録E　Verilog-HDLソースコードの説明

E.1　ファイルの構成

AutoExecute.v トリガーの発生の後に、FADCモジュールへ512サンプルのクロックを送る時間 (FPGA自身のクロック数) を数えて、連続読み出しモジュールを起動する。

B_ANY.v バックプレーン信号を変化させるモジュールは複数あるが、その中から適切なモジュールの出力をセレクトして、実際のバックプレーン信号として出力する。BD, CH, EDI以外のバックプレーン信号用。

B_BD_CH.v B_ANY.vのバックプレーン信号BD[4:0], CH[4:0]用。

B_EDI.v B_ANY.vのバックプレーン信号EDI[15:0]用。

C_*.v コマンド‘*’の実行モジュール。

ChannelControl.v Board order情報やchannel kill情報に関係する処理を行うfadc_ch.vのラッパー。他モジュールで使用している信号名と整合をとる。

ClockControl.v 外部クロックジェネレータを起動するモジュールは複数あるが、それらの起動信号の中から適切なモジュールの信号をセレクトして、出力する。

ClockGenerate.v 外部クロックジェネレータの起動や停止を制御する。設定された数のクロックを生成すると停止したり、クロック出力状態を保ったりする。

Connect.v モジュールの論理階層的には、トップモジュールであるFADCIF.vの下にあり、ここで大部分のモジュール間の接続を定義している。

Execute.v コマンドのデコードと実行を、外部からのコマンド実行要求とbusyフラグに従って処理する。

FADCIF.v モジュールの論理階層的なトップモジュールであり、外部I/O信号を定義している。

FastClear.v Fast Clearの実行要求に従って、CAMACモジュールとFADCモジュールをクリアした後に、FPGA自体をクリアする。

ImsLink.v INMOS リンクに関する処理を行う`ims_link_c012.v`のラッパー。他モジュールで使用している信号名と整合をとる。

ImsReadFifo.v INMOS リンクを通して送られてきたデータのための FIFO。4 word×8 bit。

ImsWriteFifo.v INMOS リンクを通して送るデータのための FIFO。1 word×8 bit。

Isa.v ISA リンクの示すアドレスに従って、書き込みの場合は、コマンド実行用レジスタの設定と、コマンド実行を行う。読み出しの場合は、`ImsWriteFifo.v`の内容、外部データ FIFO の内容、FADC モジュールのフラグを出力する。

Parameter.v 各モジュールで使用する定数の定義。

SerialLink.v INMOS リンクを通してデータをやり取りするモジュールは複数あるが、その中から適切なモジュールの信号をセレクトして、`ImsLink.v`と接続する。

TriggerGenerate.v 外部トリガー (TZ_TRIG, Q_TRIG, EB_TRIG) 発生あるいはコマンドによる内部トリガー発生から、FADC モジュールへのトリガー信号を生成する。

fadc_params.v 各モジュールで使われる定数のパラメータを定義している。

fadc_ch.v FPGA 内部の RAM とその読み込み・書き込みのためのモジュールを定義している。

fadc_read.v FPGA 内部の RAM の情報を参照しながら最大 1024 チャンネルの FADC を連続して読み出すモジュールを定義している。

fadc_patch.v 連続読み出しで読み出された FADC データを加工するモジュールを定義している。

ims_link_c012.v トランスピュータと INMOS リンクで接続する IC である IMS C012 に読み込み・書き込みを行うモジュールを定義している。

logiblox_ram.v Xilinx の FPGA に組み込まれた RAM モジュールのラッパーとなるモジュールを定義している。コンパイル指示子でシミュレーション用のソースと論理合成用のソースを自動的に切り換えるようになっている。

E.2 ソースコードの説明

この節では FPGA のソースコードの内、FADC データの連続読み出しに関連した部分について解説する。連続読み出しに関連した部分とは、具体的には

- FPGA 内部の RAM
- FADC データの連続読み出し

- FADCデータの加工

のことである。これらの機能は、以下のファイル中に記述されている。

- fadc_params.v
- fadc_patch.v
- logiblox_ram.v
- fadc_ch.v
- ims_link.v
- fadc_read.v
- ims_link_c012.v

E.2.1 fadc_params.vファイル

E.2.1.1 ファイルの説明

fadc_params.vは各モジュールで使われる重要なパラメータを定義している。このファイルは各モジュール内でコンパイラ指示子‘includeを使って取り込まれている。定義されているパラメータは次の通りである。

E.2.1.2 fadc_ch.vで使用されるパラメータ

MAX_CH_ADRS チャンネルアドレスの取り得る最大値。変更してはならない。

MAX_BRD_ADRS ボードアドレスの取り得る最大値。変更してはならない。

MAX_ODR_ADRS Board orderアドレスの取り得る最大値。変更してはならない。

MAX_ADRS FADCアドレス(ボード 5 bit, チャンネル 5 bit, 合計 10 bit)の取り得る最大値。変更してはならない。

MAX_KILL_ADRS channel kill RAMで内部的に使われるRAMモジュールのアドレスの取り得る最大値。変更してはならない。

E.2.1.3 fadc_read.vで使用されるパラメータ

COMPR_EMPTY データ圧縮ICのemptyフラグの空の状態を定義している。データ圧縮ICのemptyフラグは負論理なので、0に設定されている。変更してはならない。

DEFAULT_COMPR_STRB データ圧縮ICのSTRB信号のデフォルト(非選択)状態を定義している。データ圧縮ICのSTRB信号は負論理なので、1に設定されている。変更してはならない。

READ_COUNT_WIDTH 連続読み出し時にチャンネルのアドレスを数えるカウンタのbit幅を定義している。変更してはならない。

STRB_INTVL_COUNT_WIDTH 連続読み出し中にデータ圧縮ICへ送るSTRB信号の幅を数える、カウンタのbit幅を定義している。STRB_INTVL_COUNT_MAXを収められるだけの十分な大きさが必要である。

STRB_INTVL_COUNT_MAX 連続読み出し中にデータ圧縮ICへ送るSTRB信号の幅のクロック数を定義している。0から数え始めるので、実際のクロック数はSTRB_INTVL_COUNT_MAXに1を加えた数になる。ロジックの都合上0に設定することはできない。

ADRS_INTVL_COUNT_WIDTH 連続読み出し中にFADCチャンネルが切り替わるのを待つ間にクロックを数えるカウンタのbit幅を定義している。ADRS_INTVL_COUNT_MAXを収められるだけの十分な大きさが必要である。

ADRS_INTVL_COUNT_MAX 連続読み出し中にFADCチャンネルが切り替わるのを待つクロック数を定義している。0から数え始めるので、実際のクロック数はADRS_INTVL_COUNT_MAXに1を加えた数になる。ロジックの都合上0に設定することはできない。

E.2.1.4 fadc_patch.v で使用されるパラメータ

SUM データ圧縮ICの圧縮モードがSUMモードのときに、fadc_patch_mode_ramモジュールのRAMが記憶する値。変更してはならない。

CRAW データ圧縮ICの圧縮モードがCRAWモードのときに、fadc_patch_mode_ramモジュールのRAMが記憶する値。変更してはならない。

HIST データ圧縮ICの圧縮モードがHISTモードのときに、fadc_patch_mode_ramモジュールのRAMが記憶する値。変更してはならない。

RAW データ圧縮ICの圧縮モードがRAWモードのときに、fadc_patch_mode_ramモジュールのRAMが記憶する値。変更してはならない。

E.2.1.5 ims_link.v で使用されるパラメータ

トランスピュータの INMOS リンクの I/O を制御する IC である IMS C011 を制御するモジュールのパラメータ。IMS C011 は入手不可能であることが判明したので、`ims_link.v` は現在は使われていない。

IMS_LINK_RESET_WIDTH

IMS_LINK_RESET_COUNT

IMS_LINK_TIMEOUT_WIDTH

IMS_LINK_TIMEOUT_COUNT

E.2.1.6 ims_link_c012.v で使用されるパラメータ

DEFAULT_IMS_notCS IMS C012 のnotCS 信号の非アクティブ時の値。notCS 信号は負論理なので、1 に設定されている。変更してはならない。

DEFAULT_IMS_RS IMS C012 の 2 bit のRS 信号の非アクティブ時の値。特に制限は無いので 0 に設定されている。

DEFAULT_IMS_RnotW IMS C012 のRnotW 信号の非アクティブ時の値。RnotW 信号は正論理で、この信号が真のとき IMS C012 は INMOS リンクからデータを読み込み、偽のときデータを読み出す。特に制限は無いのいので 0 に設定されている。

E.2.2 fadc_ch.v ファイル

E.2.2.1 ファイルの説明

このファイルでは連続読み出し時に FADC データを読み出すのに必要なデータを記憶し、それらのデータに読み出し・書き込みを行うモジュールを定義している。

E.2.2.2 fadc_kill_channel_ram モジュール

モジュールの説明

FADC チャンネルを読み飛ばすかどうかを決める Channel kill データの記憶と、その読み出し・書き込みを行う。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

drive_read 1 bit の入力ポート。この信号が 1 のとき`fadc_kill_channel_ram` モジュールは連続読み出しモードで動作する。連続読み出しモードでは、`board` ポート, `address` ポートに入力されたアドレスの kill 値が 2 クロック後に`kill_out` ポートへ出力され、`board`, `address` ポートのアドレスを変化させることで連続して kill 値を読み出すことができる。

read_enable 1 bit の入力ポート。`busy` ポートの出力が 0 のときこのポートに 1 が入力されると、`fadc_kill_channel_ram` モジュールは`board`, `address` ポートのアドレスを読んで、そのアドレスの kill 値を`kill_out` ポートに出力する。動作中は`busy` ポートに 1 が出力される。

write_enable 1 bit の入力ポート。`busy` ポートの出力が 0 のときこのポートに 1 が入力されると、`fadc_kill_channel_ram` モジュールは`board` ポート, `address` ポートのアドレスと`kill_in` ポートの kill 値を読んで、そのアドレスに、入力された KILL 値を設定する。動作中は`busy` ポートに 1 が出力される。

fill 1 bit の入力ポート。`busy` ポートの出力が 0 のときこのポートに 1 が入力されると、`fadc_kill_channel_ram` モジュールは`kill_in` ポートの kill 値を読んで、全てのアドレスの kill 値をその値に設定する。動作中は`busy` ポートに 1 が出力される。

board 5 bit の入力ポート。FADC モジュールのアドレスを入力する。

channel 5 bit の入力ポート。チャンネルのアドレスを入力する。

pipeline_board_out 5 bit の出力ポート。連続読み出しモードのボードアドレス値が 1 クロック遅れて出力される。

pipeline_channel_out 5 bit の出力ポート。連続読み出しモードのチャンネルアドレス値が 1 クロック遅れて出力される。

kill_in 1 bit の入力ポート。kill 値を入力する。

kill_out 1 bit の出力ポート。kill 値が出力される。

E.2.2.3 fadc_board_order_ram モジュール

モジュールの説明

FADC モジュールを読み出す順番 (board order 情報) の記憶と、その読み出し・書き込みを行う。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

drive_read 1 bit の入力ポート。この信号が 1 のとき fadc_board_order_ram モジュールは連続読み出しモードで動作する。連続読み出しモードでは、order ポートに入力された board order アドレスのボードアドレス値が 1 クロック後に board_out ポートへ出力され、order ポートのアドレスを変化させることで連続してボードアドレス値を読み出すことができる。

read_enable 1 bit の入力ポート。busy ポートの出力が 0 のときこのポートに 1 が入力されると、fadc_board_order_ram は order ポートの board order アドレスを読んで、そのアドレスのボードアドレス値を board_out ポートに出力する。動作中は busy ポートに 1 が出力される。

write_enable 1 bit の入力ポート。busy ポートの出力が 0 のときこのポートに 1 が入力されると、fadc_board_order_ram は order ポートの board order アドレスと board_in のボードアドレス値を読んで、そのアドレスに、入力されたボードアドレス値を設定する。動作中は busy ポートに 1 が出力される。

order 5 bit の入力ポート。board order アドレスを設定する。

board_in 5 bit の入力ポート。ボードアドレス値を入力する。

board_out 5 bit の出力ポート。ボードアドレス値が出力される。

E.2.2.4 fadc_patch_mode_ram モジュール

モジュールの説明

FADC モジュール毎の圧縮モードの記憶と、その読み出し・書き込みを行う。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

drive_read 1 bit の入力ポート。この信号が 1 のとき fadc_patch_mode_ram モジュールは連続読み出しモードで動作する。連続読み出しモードでは、board ポートに入力されたアドレスのデータ圧縮 IC の圧縮モード値が 1 クロック後に mode_out ポートへ出力され、board のアドレスを変化させることで連続してデータ圧縮 IC の圧縮モード値を読み出すことができる。

read_enable 1 bit の入力ポート。busy ポートの出力が 0 のときこのポートに 1 が入力されると、fadc_patch_mode_ram モジュールは board ポートのボードアドレスを読んで、そのアドレスのデータ圧縮 IC の圧縮モード値を mode_out ポートに出力する。動作中は busy ポートに 1 が出力される。

write_enable 1 bit の入力ポート。busy ポートの出力が 0 のときこのポートに 1 が入力されると、fadc_patch_mode_ram モジュールは board ポートのボードアドレスと mode_in ポートのデータ圧縮 IC の圧縮モード値を読んで、そのアドレスに、入力されたデータ圧縮 IC の圧縮モード値を設定する。動作中は busy ポートに 1 が出力される。

fill 1 bit の入力ポート。busy ポートの出力が 0 のときこのポートに 1 が入力されると、fadc_patch_mode_ram モジュールは mode_in ポートのデータ圧縮 IC の圧縮モード値を読んで、全てのアドレスのデータ圧縮 IC の圧縮モード値をその値に設定する。動作中は busy ポートに 1 が出力される。

board 5 bit の入力ポート。ボードアドレスを入力する。

mode_in 2 bit の入力ポート。データ圧縮 IC の圧縮モード値を入力する。

mode_out 2 bit の出力ポート。データ圧縮 IC の圧縮モード値が出力される。

E.2.2.5 fadc_channel_counter モジュール

モジュールの説明

fadc_channel_seq_read で連続読み出し時にアドレスを数えるのに使われる。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

drive 1 bit の入力ポート。このポートに 1 が入力されると動作し、0 が入力されると停止する。

board_num 5 bit の入力ポート。ボード数を入力する。

increase 1 bit の入力ポート。drive ポートが 1 のとき、この信号に 1 を入力するとカウンタの数が一つ増える。

overflow 1 bit の出力ポート。カウンタの数がボード数を越えてオーバーフローしたときに 1 が出力される。

count 10 bit の出力。今迄にカウンタが数えた数が出力される。

E.2.2.6 fadc_channel_data_pipeline モジュール

モジュールの説明

fadc_channel_seq_read で使われ、連続読み出し時に各モジュールから読み出したデータを同期させる。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

drive 1 bit の入力ポート。このポートに 1 が入力されると動作し、0 が入力されると停止する。

kill_in 1 bit の入力ポート。fadc_kill_channel_ram モジュールの kill_out ポートの出力を接続する。

patch_mode_in 2 bit の入力ポート。fadc_patch_mode_ram モジュールの mode_out ポートの出力を接続する。

board_in 5 bit の入力ポート。fadc_kill_channel_ram モジュールの pipeline_board_out ポートの出力を接続する。

channel_in 5 bit の入力ポート。fadc_kill_channel_ram モジュールの pipeline_channel_out ポートの出力を接続する。

kill_out 1 bit の出力ポート。kill_in ポートに入力された信号が、他の信号と同期するようにタイミングを調節されて出力される。

patch_mode_out 2 bit の出力ポート。patch_mode_in ポートに入力された信号が、他の信号と同期するようにタイミングを調節されて出力される。

board_out 5 bit の出力ポート。board_in ポートに入力された信号が、他の信号と同期するようにタイミングを調節されて出力される。

channel_out 5 bit の出力ポート。channel_in ポートに入力された信号が、他の信号と同期するようにタイミングを調節されて出力される。

E.2.2.7 fadc_channel_seq_read モジュール

モジュールの説明

連続読み出し時に各 RAM の情報を連続して読み出す。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。fadc_channel_seq_read モジュールにおいては、連続読み出しモードの初期化時と終了時の処理中に 1 を出力する。

mode 1 bit の出力ポート。連続読み出しモードで動作中に 1 を出力する。

setup 1 bit の入力ポート。待機中に 1 を入力すると、パイプラインを初期化して連続読み出しモードを開始する。

pop 1 bit の入力ポート。連続読み出し中に 1 を入力すると、次のデータを読み出す。

board_num 5 bit の入力ポート。ボード数を入力する。

drive_kill_ram_read 1 bit の出力ポート。fadc_kill_channel_ram モジュールの drive_read ポートの入力に接続する。

drive_order_ram_read 1 bit の出力ポート。fadc_board_order_ram モジュールの drive_read ポートの入力に接続する。

drive_patch_mode_ram_read 1 bit の出力ポート。fadc_patch_mode_ram モジュールの drive_read ポートの入力に接続する。

order_address 5 bit の出力ポート。連続読み出し時の board order アドレスを出力する。fadc_board_order_ram モジュールの order ポートの入力に接続する。

board_address 5 bit の出力ポート。連続読み出し時のボードアドレスを出力する。fadc_kill_channel_ram モジュールの board ポートの入力に接続する。

channel_address 5 bit の出力ポート。連続読み出し時のチャンネルアドレスを出力する。fadc_kill_channel_ram モジュールの channel ポートの入力に接続する。

pipeline_board_in 5 bit の入力ポート。連続読み出し時のボードアドレスが入力される。fadc_kill_channel_ram モジュールの pipeline_board_out ポートの出力を接続する。

pipeline_channel_in 5 bit の入力ポート。連続読み出し時のチャンネルアドレスが入力される。fadc_kill_channel_ram モジュールの pipeline_channel_out ポートの出力を接続する。

kill_in 1 bit の入力ポート。連続読み出し時の channel kill データが入力される。fadc_kill_channel_ram モジュールの kill_out ポートの出力を接続する。

board_in 5 bit の入力ポート。連続読み出し時のボードアドレスが入力される。fadc_board_order_ram モジュールの board_out ポートの出力を接続する。

patch_mode_in 2 bit の入力ポート。連続読み出し時に FADC モジュール毎のデータ圧縮 IC の圧縮モードが入力される。fadc_patch_mode_ram モジュールの mode_out ポートの出力を接続する。

kill_out 1 bit の出力ポート。連続読み出し時の channel kill データが出力される。

patch_mode_out 2 bit の出力ポート。連続読み出し時の FADC モジュール毎のデータ圧縮 IC の圧縮モードが出力される。

board_out 5 bit の出力ポート。連続読み出し時のボードアドレスが出力される。

channel_out 5 bit の出力ポート。連続読み出し時のチャンネルアドレスが出力される。

E.2.2.8 fadc_channel モジュール

モジュールの説明

fadc_ch.v ファイルで定義された各モジュールを配線する、最上位モジュール。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

read_kill 1 bit の入力ポート。待機状態でこの信号が 1 になると channel kill データの読み出しを行う。

write_kill 1 bit の入力ポート。待機状態でこの信号が 1 になると channel kill データの書き込みを行う。

fill_kill 1 bit の入力ポート。待機状態でこの信号が 1 になると channel kill データの初期化を行う。

read_order 1 bit の入力ポート。待機状態でこの信号が 1 になると board order データの読み出しを行う。

write_order 1 bit の入力ポート。待機状態でこの信号が 1 になると board order データの書き込みを行う。

read_patch_mode 1 bit の入力ポート。待機状態でこの信号が 1 になるとデータ圧縮 IC の圧縮モードデータの読み出しを行う。

write_patch_mode 1 bit の入力ポート。待機状態でこの信号が 1 になるとデータ圧縮 IC の圧縮モードデータの書き込みを行う。

fill_patch_mode 1 bit の入力ポート。待機状態でこの信号が 1 になるとデータ圧縮 IC の圧縮モードデータの初期化を行う。

order 5 bit の入力ポート。board order アドレスを入力する。read_kill ポート, write_kill ポートの命令で使われる。

board 5 bit の入力ポート。ボードアドレスを入力する。read_kill ポート, write_kill ポート, read_patch_mode ポート, write_patch_mode ポートの命令で使われる。

channel 5 bit の入力ポート。チャンネルアドレスを入力する。read_kill ポート, write_kill ポートの命令で使われる。

kill_in 1 bit の入力ポート。channel kill データを入力する。write_kill ポート, fill_kill ポートの命令で使われる。

kill_out 1 bit の出力ポート。Channel kill データが出力される。read_kill ポートの命令で使われる。

board_in 5 bit の入力ポート。ボードデータを入力する。write_order ポートの命令で使われる。

board_out 5 bit の出力ポート。ボードアドレスが出力される。read_order ポートの命令で使われる。

patch_mode_in 2 bit の入力ポート。データ圧縮 IC の圧縮モードデータが入力される。write_patch_mode ポート, fill_patch_mode ポートの命令で使われる。

patch_mode_out 2 bit の出力ポート。データ圧縮 IC の圧縮モードデータが出力される。read_patch_mode ポートの命令で使われる。

seq_read_mode 1 bit の出力ポート。連続読み出しモードで動作中に 1 が出力される。

seq_read_setup 1 bit の入力ポート。この信号に 1 を入力すると連続読み出しモードを開始する。

seq_read_pop 1 bit の入力ポート。連続読み出しモードでこの信号に 1 を入力すると、次の kill されていないチャンネルのデータを読み出す。Kill されたチャンネルを読み飛ばしている間は、busy に 1 が出力される。

seq_read_board_num 5 bit の入力ポート。連続読み出しモードで読み出す FADC モジュールの数を入力する。

seq_read_patch_mode_out 2 bit の出力ポート。連続読み出しモードで読み出されたデータ圧縮 IC の圧縮モードのデータが出力される。

seq_read_board_out 5 bit の出力ポート。連続読み出しモードで読み出されたボードアドレスが出力される。

seq_read_channel_out 5 bit の出力ポート。連続読み出しモードで読み出されたチャンネルアドレスが出力される。

E.2.3 fadc_read.v ファイル

このファイルでは連続読み出し時に FADC データを読み出すモジュールを定義している。

E.2.3.1 fadc_read_empty モジュール

モジュールの説明

データ圧縮 IC の empty フラグを高速に読み込む。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

drive 1 bit の入力ポート。このポートに 1 が入力されたとき、データ圧縮 IC から読み込んだ empty フラグを empty ポートに出力する。

empty 1 bit の出力ポート。データ圧縮 IC から読み込んだ FADC の empty フラグの値が出力される。

compr_empty 1 bit の入力ポート。データ圧縮 IC の empty フラグの出力を接続する。

E.2.3.2 fadc_read_data モジュール

モジュールの説明

FADC データを高速に読み込む。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

drive 1 bit の入力ポート。このポートに 1 が入力されたとき、データ圧縮 IC から読み込んだデータを `data` ポートに出力する。

data 16 bit の出力ポート。データ圧縮 IC から読み込まれたデータが出力される。

compr_data 16 bit の入力ポート。データ圧縮 IC のデータ出力を接続する。

E.2.3.3 fadc_read_overflow モジュール

モジュールの説明

データ圧縮 IC のオーバーフローフラグを高速に読み込む。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

drive 1 bit の入力ポート。この信号の入力が 1 のとき、データ圧縮 IC から読み込んだオーバーフローフラグを `overflow` ポートに出力する。

overflow 1 bit の出力ポート。データ圧縮 IC から読み込んだオーバーフローフラグが出力される。

compr_overflow 1 bit の入力ポート。データ圧縮 IC のオーバーフローフラグの出力を接続する。

E.2.3.4 fadc_read モジュール

モジュールの説明

FADC データの連続読み出しを行う。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

drive 1 bit の入力ポート。このポートに 1 が入力されているとき連続読み出しを行い、0 が入力されると連続読み出しを一時停止する。

read 1 bit の入力ポート。このポートに 1 を入力すると連続読み出しを開始する。

all_mode 1 bit の入力ポート。`read` ポートに 1 を入力して連続読み出しを開始するとき、このポートに 1 が入力されているとデータ圧縮 IC の empty フラグを無視して 256 個の全データを読み出し、0 が入力されているとデータ圧縮 IC の empty フラグが立つまでデータを読み出す。

seq_mode 1 bit の入力ポート。`read` ポートに 1 を入力して連続読み出しを開始するとき、このポートに 1 が入力されていると全チャンネルの FADC データの読み出しを行い、0 が入力されていると指定した 1 つのチャンネルの FADC データだけを読み出す。

board 5 bit の入力ポート。`seq_mode` ポートに 0 を入力したときに使われるボードアドレスを入力する。

channel 5 bit の入力ポート。`seq_mode` ポートに 0 を入力したときに使われるチャンネルアドレスを入力する。

seq_read_busy 1 bit の入力ポート。`fadc_channel` モジュールの `busy` ポートの出力を接続する。

seq_read_mode 1 bit の入力ポート。fadc_channel モジュールの seq_read_mode ポートの出力を接続する。

seq_read_pop 1 bit の出力ポート。fadc_channel モジュールの seq_read_pop ポートの入力に接続する。

seq_read_patch_mode_in 2 bit の入力ポート。fadc_channel モジュールの seq_read_patch_mode_out ポートの出力を接続する。

seq_read_board_in 5 bit の入力ポート。fadc_channel モジュールの seq_read_board_out ポートの出力を接続する。

seq_read_channel_in 5 bit の入力ポート。fadc_channel モジュールの seq_read_channel_out ポートの出力を接続する。

E.2.3.5 fadc_read_mode モジュール

モジュールの説明

FADC I/F に送られて来るコマンドのオプションから、fadc_read モジュールの圧縮モードを決定する。

入出力ポートの説明

one_not_all_channel 1 bit の入力ポート。このポートの信号が 1 のとき、FADC チャンネルを 1 つだけ読み出し、0 のとき全 FADC チャンネルを連続して読み出すように fadc_read モジュールを設定する。

patch_each_board 1 bit の入力ポート。このポートの信号が 1 のとき FADC モジュール毎にデータ圧縮 IC の圧縮モードが変更されており、0 のとき全 FADC モジュールのデータ圧縮 IC の圧縮モードが同じであると仮定してデータ圧縮 IC を読み出すように fadc_read モジュールを設定する。

patch_mode 2 bit の入力ポート。全 FADC モジュールで共通のデータ圧縮 IC の圧縮モードを入力する。

data_board_patch_mode 2 bit の入力ポート。各 FADC モジュール毎のデータ圧縮 IC の圧縮モードを入力する。fadc_channel モジュールの seq_read_patch_mode_out ポートの出力を接続する。

all_mode 1 bit の出力ポート。fadc_read モジュールの all_mode ポートの入力に接続する。

seq_mode 1 bit の出力ポート。fadc_read モジュールの seq_mode ポートの入力に接続する。

E.2.3.6 fadc_read_latch_drive モジュール

モジュールの説明

各パイプライン間でデータを記憶するラッチの動作信号をコントロールする。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

drive_in 1 bit の入力ポート。前段のパイプラインの動作信号を入力する。

drive_out 1 bit の出力ポート。後段のパイプラインの動作信号として、drive_in の出力が 1 クロック遅れて出力される。

E.2.3.7 fadc_read_data_latch モジュール

モジュールの説明

各パイプラインの間でデータを記憶するラッチ。このモジュールがデータを記憶しているので、FIFO から full フラグが出力されてパイプラインを一時停止させるとき、FIFO に近いパイプラインから順番に停止させることができる。

このモジュールがパイプラインの間に無いと、全パイプラインを一斉に停止させなければならないので、遅延のために FADC I/F が必要な速度で動かなくなる (3 章の 3.4.3.2 節参照)。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

drive_in 1 bit の入力ポート。fadc_read_latch_drive モジュールのdrive_inポートへ出力するのと同じ信号を接続する。

drive_out 1 bit の入力ポート。fadc_read_latch_drive モジュールのdrive_outポートから出力される信号を接続する。

data_in 1 bit の入力ポート。前段のパイプラインが出力する信号を接続する。

data_out 1 bit の出力ポート。後段のパイプラインへ入力する信号が出力される。

E.2.3.8 fadc_read_data2_latch モジュール

モジュールの説明

fadc_read_data_latch モジュールと同じだが、2 bit のデータを記憶する。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

drive_in 1 bit の入力ポート。fadc_read_latch_drive モジュールのdrive_inポートへ出力するのと同じ信号を接続する。

drive_out 1 bit の入力ポート。fadc_read_latch_drive モジュールのdrive_outポートから出力される信号を接続する。

data_in 2 bit の入力ポート。前段のパイプラインが出力する信号を接続する。

data_out 2 bit の出力ポート。後段のパイプラインへ入力する信号が出力される。

E.2.3.9 fadc_read_data4_latch モジュール

モジュールの説明

fadc_read_data_latch モジュールと同じだが、4 bit のデータを記憶する。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

drive_in 1 bit の入力ポート。fadc_read_latch_drive モジュールのdrive_inポートへ出力するのと同じ信号を接続する。

drive_out 1 bit の入力ポート。fadc_read_latch_drive モジュールのdrive_outポートから出力される信号を接続する。

data_in 4 bit の入力ポート。前段のパイプラインが出力する信号を接続する。

data_out 4 bit の出力ポート。後段のパイプラインへ入力する信号が出力される。

E.2.3.10 fadc_read_data5_latch モジュール

fadc_read_data_latch モジュールと同じだが、5 bit のデータを記憶する。

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

drive_in 1 bit の入力ポート。fadc_read_latch_drive モジュールのdrive_inポートへ出力するのと同じ信号を接続する。

drive_out 1 bit の入力ポート。fadc_read_latch_drive モジュールのdrive_outポートから出力される信号を接続する。

data_in 5 bit の入力ポート。前段のパイプラインが出力する信号を接続する。

data_out 5 bit の出力ポート。後段のパイプラインへ入力する信号が出力される。

E.2.3.11 fadc_read_data8_latch モジュール

モジュールの説明

fadc_read_data_latch モジュールと同じだが、8 bit のデータを記憶する。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

drive_in 1 bit の入力ポート。fadc_read_latch_drive モジュールの drive_in ポートへ出力するのと同じ信号を接続する。

drive_out 1 bit の入力ポート。fadc_read_latch_drive モジュールの drive_out ポートから出力される信号を接続する。

data_in 8 bit の入力ポート。前段のパイプラインが出力する信号を接続する。

data_out 8 bit の出力ポート。後段のパイプラインへ入力する信号が出力される。

E.2.3.12 fadc_read_data16_latch モジュール

モジュールの説明

fadc_read_data_latch モジュールと同じだが、16 bit のデータを記憶する。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

drive_in 1 bit の入力ポート。fadc_read_latch_drive モジュールの drive_in ポートへ出力するのと同じ信号を接続する。

drive_out 1 bit の入力ポート。fadc_read_latch_drive モジュールの drive_out ポートから出力される信号を接続する。

data_in 16 bit の入力ポート。前段のパイプラインが出力する信号を接続する。

data_out 16 bit の出力ポート。後段のパイプラインへ入力する信号が出力される。

E.2.3.13 fadc_read_control モジュール

モジュールの説明

fadc_read.v ファイルと fadc_patch.v ファイルで定義された各モジュールを配線する、最上位モジュール。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

read 1 bit の入力ポート。このポートに 1 が入力されると、FADC データの連続読み出しを開始する。

threshold 4 bit の入力ポート。Threshold を 2 の冪乗で入力する。

option1 16 bit の入力ポート。FADC I/F に送られて来るコマンドの 1 番目のオプションを入力する。

option2 16 bit の入力ポート。FADC I/F に送られて来るコマンドの 2 番目のオプションを入力する。

seq_read_busy 1 bit の入力ポート。fadc_channel モジュールの busy ポートの出力を接続する。

seq_read_mode 1 bit の入力ポート。fadc_channel モジュールの seq_read_mode ポートの出力を接続する。

seq_read_setup 1 bit の出力ポート。fadc_channel モジュールの seq_read_setup ポートの入力に接続する。

seq_read_pop 1 bit の出力ポート。fadc_channel モジュールのseq_read_pop ポートの入力に接続する。

seq_read_patch_mode_in 2 bit の入力ポート。fadc_channelモジュールのseq_read_patch_mode_out ポートの出力を接続する。

seq_read_board_in 5 bit の入力ポート。fadc_channel モジュールのseq_read_board_out ポートの出力を接続する。

seq_read_channel_in 5 bit の入力ポート。fadc_channelモジュールのseq_read_channel_out ポートの出力を接続する。

E.2.4 fadc_patch.v ファイル

E.2.4.1 ファイルの説明

データ圧縮 IC の各圧縮モードの出力データを加工するモジュールを定義している。

E.2.4.2 fadc_last_address_latch モジュール

モジュールの説明

最後に有効だったボード・チャンネルアドレスを記憶する。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

drive 1 bit の入力ポート。このポートに 1 が入力されると動作し、0 が入力されると停止する。

data_board_in 5 bit の入力ポート。ボードアドレスを入力する。

data_channel_in 5 bit の入力ポート。チャンネルアドレスを入力する。

data_channel_begin 1 bit の入力ポート。このポートに 1 が入力されたチャンネルは有効であると見なされ、ボード・チャンネルアドレスを記憶する。このポートに 0 が入力されたチャンネルは無効であると見なされ、ボード・チャンネルアドレスは記憶されない。

data_channel_end 1 bit の入力ポート。data_channel_begin ポートに 1 が入力された場合、このポートに 1 が入力されると、そのクロックでボード・チャンネルアドレスが記憶される。

last_board_out 5 bit の出力ポート。最後に有効だったボードアドレスが出力される。

last_channel_out 5 bit の出力ポート。最後に有効だったチャンネルアドレスが出力される。

E.2.4.3 fadc_convert_16to8 モジュール

モジュールの説明

16 bit 幅の 1 つのデータを、8 bit 幅の 2 つのデータに分ける。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

drive 1 bit の入力ポート。このポートに 1 が入力されると動作し、0 が入力されると停止する。

data16_in_busy 1 bit の入力ポート。16 bit 幅のデータを出力するモジュールのbusy ポートの出力を接続する。

data16_in_drive 1 bit の出力ポート。16 bit 幅のデータを出力するモジュールのdrive ポートの入力を接続する。

data16_in_valid 1 bit の入力ポート。16 bit 幅のデータを出力するモジュールのvalid ポートの出力を接続する。

data16_in 16 bit の入力ポート。16 bit 幅のデータを入力する。

data8_out_valid 1 bit の出力ポート。data8_out ポートの出力が有効なとき 1 が出力され、無効なとき 0 が出力される。

data8_out 8 bit の出力ポート。16 bit 幅のデータを 2 つに分けた 8 bit 幅のデータが、上位の bit から順番に出力される。

E.2.4.4 fadc_sum_patch_without_threshold モジュール

モジュールの説明

データ圧縮 IC の SUM 圧縮モードのデータを加工する。Threshold はチェックしない。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

drive 1 bit の入力ポート。このポートに 1 が入力されると動作し、0 が入力されると停止する。

patch 1 bit の入力ポート。このポートに 1 が入力されると、動作を開始する。

fadc_read_busy 1 bit の入力ポート。fadc_read モジュールの busy ポートの出力を接続する。

fadc_read_drive 1 bit の出力ポート。fadc_read モジュールの drive ポートの入力を接続する。

data_in_valid 1 bit の入力ポート。fadc_read モジュールの data_out_valid ポートの出力を接続する。

data_in 16 bit の入力ポート。fadc_read モジュールの data_out ポートの出力を接続する。

data_overflow_in 1 bit の入力ポート。fadc_read モジュールの data_overflow_out ポートの出力を接続する。

data_board_in 5 bit の入力ポート。fadc_read モジュールの data_board_out ポートの出力を接続する。

data_channel_in 5 bit の入力ポート。fadc_read モジュールの data_channel_out ポートの出力を接続する。

data_channel_begin 1 bit の入力ポート。fadc_read モジュールの data_channel_begin ポートの出力を接続する。

data_channel_end 1 bit の入力ポート。fadc_read モジュールの data_channel_end ポートの出力を接続する。

last_address_drive 1 bit の出力ポート。fadc_last_address_latch モジュールの drive ポートの入力に接続する。

last_board_in 1 bit の入力ポート。fadc_last_address_latch モジュールの last_board_out ポートの出力を接続する。

last_channel_in 1 bit の入力ポート。fadc_last_address_latch モジュールの last_channel_out ポートの出力を接続する。

data_out_valid 1 bit の出力ポート。data_out ポートの出力が有効なとき 1 が出力され、無効なとき 0 が出力される。

data_out_count 2 bit の出力ポート。データ圧縮 IC の SUM 圧縮モードのデータは 16 bit × 4 個で一つのクラスタを形成し、このポートの出力は、data_out ポートが出力しているデータがその内の何番目のデータであるかを示している。

data_out 16 bit の出力ポート。加工されたデータが出力される。

E.2.4.5 fadc_sum_patch_threshold モジュール

モジュールの説明

データ圧縮 IC の SUM 圧縮モードのデータの threshold をチェックする。

入出力ポートの説明

sum 16 bit の入力ポート。データ圧縮 IC の SUM 圧縮モードのデータを入力する。

threshold 4 bit の入力ポート。Threshold の値を 2 の冪乗で入力する。

over_threshold 1 bit の出力ポート。入力されたデータが threshold を越えているとき 1 が出力され、そうでなければ 0 が出力される。

E.2.4.6 fadc_sum_patch_data_pipeline モジュール

モジュールの説明

データ圧縮 IC の SUM 圧縮モードのデータを、threshold をチェックするまで保存する。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

drive 1 bit の入力ポート。このポートに 1 が入力されると動作し、0 が入力されると停止する。

shift 1 bit の入力ポート。

E.2.4.7 fadc_sum_patch_with_threshold モジュール

モジュールの説明

データ圧縮 IC の SUM 圧縮モードのデータを、threshold をチェックしながら加工する。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

drive 1 bit の入力ポート。このポートに 1 が入力されると動作し、0 が入力されると停止する。

patch 1 bit の入力ポート。このポートに 1 が入力されると動作を開始する。

threshold_bit_in 4 bit の入力ポート。Threshold の値を 2 の冪乗で入力する。内部で`fadc_sum_patch_threshold` モジュールの`threshold` ポートの入力に接続されている。

sum_patch_drive 1 bit の入力ポート。`fadc_sum_patch_without_threshold` モジュールの`drive` ポートの出力を接続する。

sum_patch_busy 1 bit の入力ポート。`fadc_sum_patch_without_threshold` モジュールの`busy` ポートの出力を接続する。

sum_patch_data_in_valid 1 bit の入力ポート。`fadc_sum_patch_without_threshold`モジュールの`data_out_valid`ポートの出力を接続する。

sum_patch_data_in_count 2 bit の入力ポート。`fadc_sum_patch_without_threshold`モジュールの`data_out_count`ポートの出力を接続する。

sum_patch_data_in 16 bit の入力ポート。`fadc_sum_patch_without_threshold` モジュールの`data_out` ポートの出力を接続する。

data_out_valid 1 bit の出力ポート。`data_out` ポートの出力が有効なとき 1 が出力され、無効なとき 0 が出力される。

data_out 16 bit の出力ポート。加工されたデータが出力される。

E.2.4.8 fadc_sum_patch モジュール

モジュールの説明

データ圧縮 IC の SUM 圧縮モードのデータを加工する。内部で`fadc_sum_patch_without_threshold` モジュールと`fadc_sum_patch_with_threshold`モジュールを使用している。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

drive 1 bit の入力ポート。このポートに 1 が入力されると動作し、0 が入力されると停止する。

patch 1 bit の入力ポート。このポートに 1 が入力されると動作を開始する。

threshold_mode 1 bit の入力ポート。Threshold をチェックするとき 1 を入力し、チェックしないときは 0 を入力する。

threshold_bit_in 4 bit の入力ポート。Threshold の値を 2 の冪乗で入力する。

fadc_read_busy 1 bit の入力ポート。fadc_read モジュールの busy ポートの出力を接続する。

fadc_read_drive 1 bit の出力ポート。fadc_read モジュールの drive ポートの入力に接続する。

data_in_valid 1 bit の入力ポート。fadc_read モジュールの data_out_valid ポートの出力を接続する。

data_in 16 bit の入力ポート。fadc_read モジュールの data_out ポートの出力を接続する。

data_overflow_in 1 bit の入力ポート。fadc_read モジュールの data_overflow_out ポートの出力を接続する。

data_board_in 5 bit の入力ポート。fadc_read モジュールの data_board_out ポートの出力を接続する。

data_channel_in 5 bit の入力ポート。fadc_read モジュールの data_channel_out ポートの出力を接続する。

data_channel_begin 1 bit の入力ポート。fadc_read モジュールの data_channel_begin ポートの出力を接続する。

data_channel_end 1 bit の入力ポート。fadc_read モジュールの data_channel_end ポートの出力を接続する。

last_address_drive 1 bit の出力ポート。fadc_last_address_latch モジュールの drive ポートの入力に接続する。

last_board_in 1 bit の入力ポート。fadc_last_address_latch モジュールの last_board_out ポートの出力を接続する。

last_channel_in 1 bit の入力ポート。fadc_last_address_latch モジュールの last_channel_out ポートの出力を接続する。

data_out_valid 1 bit の出力ポート。data_out ポートの出力が有効なとき 1 が出力され、無効なとき 0 が出力される。

data_out 16 bit の出力ポート。加工されたデータが出力される。

E.2.4.9 fadc_craw_patch モジュール

モジュールの説明

データ圧縮 IC の CRAW 圧縮モードのデータを加工する。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

drive 1 bit の入力ポート。このポートに 1 が入力されると動作し、0 が入力されると停止する。

patch 1 bit の入力ポート。このポートに 1 が入力されると動作を開始する。

fadc_read_busy 1 bit の入力ポート。fadc_read モジュールの busy ポートの出力を接続する。

fadc_read_drive 1 bit の出力ポート。fadc_read モジュールの drive ポートの入力に接続する。

data_in_valid 1 bit の入力ポート。fadc_read モジュールの data_out_valid ポートの出力を接続する。

data_in 16 bit の入力ポート。fadc_read モジュールの data_out ポートの出力を接続する。

data_overflow_in 1 bit の入力ポート。fadc_read モジュールの data_overflow_out ポートの出力を接続する。

data_board_in 5 bit の入力ポート。fadc_read モジュールの data_board_out ポートの出力を接続する。

data_channel_in 5 bit の入力ポート。fadc_read モジュールの data_channel_out ポートの出力を接続する。

data_channel_begin 1 bit の入力ポート。fadc_read モジュールのdata_channel_begin ポートの出力を接続する。

data_channel_end 1 bit の入力ポート。fadc_read モジュールのdata_channel_end ポートの出力を接続する。

last_address_drive 1 bit の出力ポート。fadc_last_address_latch モジュールのdrive ポートの入力に接続する。

last_board_in 1 bit の入力ポート。fadc_last_address_latch モジュールのlast_board_out ポートの出力を接続する。

last_channel_in 1 bit の入力ポート。fadc_last_address_latch モジュールのlast_channel_out ポートの出力を接続する。

data_out_valid 1 bit の出力ポート。data_out ポートの出力が有効なとき 1 が出力され、無効なとき 0 が出力される。

data_out 16 bit の出力ポート。加工されたデータが出力される。

E.2.4.10 fadc_hist_patch モジュール

モジュールの説明

データ圧縮 IC の HIST 圧縮モードのデータを加工する。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

drive 1 bit の入力ポート。このポートに 1 が入力されると動作し、0 が入力されると停止する。

patch 1 bit の入力ポート。このポートに 1 が入力されると動作を開始する。

fadc_read_busy 1 bit の入力ポート。fadc_read モジュールのbusy ポートの出力を接続する。

fadc_read_drive 1 bit の出力ポート。fadc_read モジュールのdrive ポートの入力に接続する。

data_in_valid 1 bit の入力ポート。fadc_read モジュールのdata_out_valid ポートの出力を接続する。

data_in 16 bit の入力ポート。fadc_read モジュールのdata_out ポートの出力を接続する。

data_overflow_in 1 bit の入力ポート。fadc_read モジュールのdata_overflow_out ポートの出力を接続する。

data_board_in 5 bit の入力ポート。fadc_read モジュールのdata_board_out ポートの出力を接続する。

data_channel_in 5 bit の入力ポート。fadc_read モジュールのdata_channel_out ポートの出力を接続する。

data_channel_begin 1 bit の入力ポート。fadc_read モジュールのdata_channel_begin ポートの出力を接続する。

data_channel_end 1 bit の入力ポート。fadc_read モジュールのdata_channel_end ポートの出力を接続する。

last_address_drive 1 bit の出力ポート。fadc_last_address_latch モジュールのdrive ポートの入力に接続する。

last_board_in 1 bit の入力ポート。fadc_last_address_latch モジュールのlast_board_out ポートの出力を接続する。

last_channel_in 1 bit の入力ポート。fadc_last_address_latch モジュールのlast_channel_out ポートの出力を接続する。

data_out_valid 1 bit の出力ポート。data_out ポートの出力が有効なとき 1 が出力され、無効なとき 0 が出力される。

data_out 16 bit の出力ポート。加工されたデータが出力される。

E.2.4.11 fadc_raw_patch モジュール

モジュールの説明

データ圧縮 IC の RAW 圧縮モードのデータを加工する。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

drive 1 bit の入力ポート。このポートに 1 が入力されると動作し、0 が入力されると停止する。

patch 1 bit の入力ポート。このポートに 1 が入力されると動作を開始する。

fadc_read_busy 1 bit の入力ポート。fadc_read モジュールの busy ポートの出力を接続する。

fadc_read_drive 1 bit の出力ポート。fadc_read モジュールの drive ポートの入力に接続する。

data_in_valid 1 bit の入力ポート。fadc_read モジュールの data_out_valid ポートの出力を接続する。

data_in 16 bit の入力ポート。fadc_read モジュールの data_out ポートの出力を接続する。

data_overflow_in 1 bit の入力ポート。fadc_read モジュールの data_overflow_out ポートの出力を接続する。

data_board_in 5 bit の入力ポート。fadc_read モジュールの data_board_out ポートの出力を接続する。

data_channel_in 5 bit の入力ポート。fadc_read モジュールの data_channel_out ポートの出力を接続する。

data_channel_begin 1 bit の入力ポート。fadc_read モジュールの data_channel_begin ポートの出力を接続する。

data_channel_end 1 bit の入力ポート。fadc_read モジュールの data_channel_end ポートの出力を接続する。

last_address_drive 1 bit の出力ポート。fadc_last_address_latch モジュールの drive ポートの入力に接続する。

last_board_in 1 bit の入力ポート。fadc_last_address_latch モジュールの last_board_out ポートの出力を接続する。

last_channel_in 1 bit の入力ポート。fadc_last_address_latch モジュールの last_channel_out ポートの出力を接続する。

data_out_valid 1 bit の出力ポート。data_out ポートの出力が有効なとき 1 が出力され、無効なとき 0 が出力される。

data_out 16 bit の出力ポート。加工されたデータが出力される。

E.2.4.12 fadc_patch_mode_decoder モジュール

モジュールの説明

データ圧縮 IC の圧縮モードを表わす 2 bit の信号を、各データ加工モジュールのスタート信号に変換する。

入出力ポートの説明

patch_mode 2 bit の入力ポート。データ圧縮 IC の圧縮モードを入力する。

E.2.4.13 fadc_patch_mode_selecter モジュール

モジュールの説明

データ圧縮 IC の圧縮モードにより 4 つの 1 bit の入力信号から 1 つだけ選択する。

入出力ポートの説明

patch_mode 2 bit の入力ポート。データ圧縮 IC の圧縮モードを入力する。

sum_patch_in 1 bit の入力ポート。データ圧縮 IC の圧縮モードが SUM モードのとき使用する信号を入力する。

craw_patch_in 1 bit の入力ポート。データ圧縮 IC の圧縮モードが CRAW モードのとき使用する信号を入力する。

hist_patch_in 1 bit の入力ポート。データ圧縮 IC の圧縮モードが HIST モードのとき使用する信号を入力する。

raw_patch_in 1 bit の入力ポート。データ圧縮 IC の圧縮モードが RAW モードのとき使用する信号を入力する。

signal_out 1 bit の出力ポート。sum_patch_in, craw_patch_in, hist_patch_in, raw_patch_in の 4 つの入力ポートの内、patch_mode ポートに入力したデータ圧縮 IC の圧縮モードによって選択されたポートの信号が出力される。

E.2.4.14 fadc_patch_mode_selecter16 モジュール

モジュールの説明

データ圧縮 IC の圧縮モードにより 4 つの 16 bit の入力信号から 1 つだけ選択する。

入出力ポートの説明

patch_mode 2 bit の入力ポート。データ圧縮 IC の圧縮モードを入力する。

sum_patch_in 16 bit の入力ポート。データ圧縮 IC の圧縮モードが SUM モードのとき使用する信号を入力する。

craw_patch_in 16 bit の入力ポート。データ圧縮 IC の圧縮モードが CRAW モードのとき使用する信号を入力する。

hist_patch_in 16 bit の入力ポート。データ圧縮 IC の圧縮モードが HIST モードのとき使用する信号を入力する。

raw_patch_in 16 bit の入力ポート。データ圧縮 IC の圧縮モードが RAW モードのとき使用する信号を入力する。

signal_out 16 bit の出力ポート。sum_patch_in, craw_patch_in, hist_patch_in, raw_patch_in の 4 つの入力ポートの内、patch_mode ポートに入力したデータ圧縮 IC の圧縮モードによって選択されたポートの信号が出力される。

E.2.4.15 fadc_patch モジュール

モジュールの説明

全 FADC モジュールのデータ圧縮 IC が同一の圧縮モードであると仮定してデータを加工するように、各圧縮モードにおいてデータ圧縮 IC のデータを加工するモジュールの入出力を配線する。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

patch 1 bit の入力ポート。このポートに 1 が入力されると、FADC データの加工を開始する。

patch_mode 2 bit の入力ポート。データ圧縮 IC の圧縮モードを入力する。

fadc_read_busy 1 bit の入力ポート。fadc_read モジュールの busy ポートの出力を接続する。

fadc_read_drive 1 bit の出力ポート。fadc_read モジュールの drive ポートの入力に接続する。

last_address_drive 1 bit の出力ポート。fadc_last_address_latch モジュールの drive ポートの入力に接続する。

sum_patch 1 bit の出力ポート。fadc_sum_patch モジュールの patch ポートの入力に接続する。

sum_patch_busy 1 bit の入力ポート。fadc_sum_patch モジュールの busy ポートの出力を接続する。

sum_patch_drive 1 bit の出力ポート。fadc_sum_patch モジュールの busy ポートの入力に接続する。

sum_patch_read_busy 1 bit の出力ポート。fadc_sum_patch モジュールの fadc_read_busy ポートの入力に接続する。

sum_patch_read_drive 1 bit の入力ポート。fadc_sum_patch モジュールの fadc_read_drive ポートの出力を接続する。

sum_patch_last_address_drive 1 bit の入力ポート。fadc_sum_patch モジュールの last_address_drive ポートの出力を接続する。

sum_patch_data_out_valid 1 bit の入力ポート。fadc_sum_patch モジュールの data_out_valid ポートの出力を接続する。

sum_patch_data_out 16 bit の入力ポート。fadc_sum_patch モジュールの data_out ポートの出力を接続する。

craw_patch 1 bit の出力ポート。fadc_craw_patch モジュールのpatch ポートの入力に接続する。

craw_patch_busy 1 bit の入力ポート。fadc_craw_patch モジュールのbusy ポートの出力を接続する。

craw_patch_drive 1 bit の出力ポート。fadc_craw_patch モジュールのdrive ポートの入力に接続する。

craw_patch_read_busy 1 bit の出力ポート。fadc_craw_patch モジュールのfadc_read_busy ポートの入力に接続する。

craw_patch_read_drive 1 bit の入力ポート。fadc_craw_patch モジュールのfadc_read_drive ポートの出力を接続する。

craw_patch_last_address_drive 1 bit の入力ポート。fadc_craw_patch モジュールのlast_address_drive ポートの出力を接続する。

craw_patch_data_out_valid 1 bit の入力ポート。fadc_craw_patch モジュールのdata_out_valid ポートの出力を接続する。

craw_patch_data_out 16 bit の入力ポート。fadc_craw_patch モジュールのdata_out ポートの出力を接続する。

hist_patch 1 bit の出力ポート。fadc_hist_patch モジュールのpatch ポートの入力に接続する。

hist_patch_busy 1 bit の入力ポート。fadc_hist_patch モジュールのbusy ポートの出力を接続する。

hist_patch_drive 1 bit の出力ポート。fadc_hist_patch モジュールのdrive ポートの入力に接続する。

hist_patch_read_busy 1 bit の出力ポート。fadc_hist_patch モジュールのfadc_read_busy ポートの入力に接続する。

hist_patch_read_drive 1 bit の入力ポート。fadc_hist_patch モジュールのfadc_read_drive ポートの出力を接続する。

hist_patch_last_address_drive 1 bit の入力ポート。fadc_hist_patch モジュールのlast_address_drive ポートの出力を接続する。

hist_patch_data_out_valid 1 bit の入力ポート。fadc_hist_patch モジュールのdata_out_valid ポートの出力を接続する。

hist_patch_data_out 16 bit の入力ポート。fadc_hist_patch モジュールのdata_out ポートの出力を接続する。

raw_patch 1 bit の出力ポート。fadc_raw_patch モジュールのpatch ポートの入力に接続する。

raw_patch_busy 1 bit の入力ポート。fadc_raw_patch モジュールのbusy ポートの出力を接続する。

raw_patch_drive 1 bit の出力ポート。fadc_raw_patch モジュールのdrive の入力に接続する。

raw_patch_read_busy 1 bit の出力ポート。fadc_raw_patch モジュールのfadc_read_busy ポートの入力に接続する。

raw_patch_read_drive 1 bit の入力ポート。fadc_raw_patch モジュールのfadc_read_drive ポートの出力を接続する。

raw_patch_last_address_drive 1 bit の入力ポート。fadc_raw_patch モジュールのlast_address_drive ポートの出力を接続する。

raw_patch_data_out_valid 1 bit の入力ポート。fadc_raw_patch モジュールのdata_out_valid ポートの出力を接続する。

raw_patch_data_out 16 bit の入力ポート。fadc_raw_patch モジュールのdata_out ポートの出力を接続する。

data_out_drive 1 bit の入力ポート。このポートに 1 が入力されると FADC データの加工を行い、0 が入力されると一時停止する。

data_out_valid 1 bit の出力ポート。data_out に加工された FADC データが出力されるときこのポートに 1 が出力され、それ以外のときは 0 が出力される。

data_out 16 bit の出力ポート。このポートに加工された FADC データが出力される。

E.2.4.16 fadc_patch_each_board モジュール

モジュールの説明

各 FADC モジュール毎にデータ圧縮 IC の圧縮モードを見てデータ加工するように、各モジュールの入出力を配線する。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

patch 1 bit の入力ポート。このポートに 1 が入力されると、FADC データの加工を開始する。

fadc_read_busy 1 bit の入力ポート。fadc_read モジュールの busy ポートの出力を接続する。

fadc_read_drive 1 bit の出力ポート。fadc_read モジュールの drive ポートの入力に接続する。

data_board_begin 1 bit の入力ポート。fadc_read モジュールの data_board_begin ポートの出力を接続する。

data_board_patch_mode 2 bit の入力ポート。data_board_patch_mode ポートの出力を接続する。

last_address_drive 1 bit の出力ポート。fadc_last_address_latch モジュールの drive ポートの入力に接続する。

sum_patch 1 bit の出力ポート。fadc_sum_patch モジュールの patch ポートの入力に接続する。

sum_patch_busy 1 bit の入力ポート。fadc_sum_patch モジュールの busy ポートの出力を接続する。

sum_patch_drive 1 bit の出力ポート。fadc_sum_patch モジュールの busy ポートの入力に接続する。

sum_patch_read_busy 1 bit の出力ポート。fadc_sum_patch モジュールの fadc_read_busy ポートの入力に接続する。

sum_patch_read_drive 1 bit の入力ポート。fadc_sum_patch モジュールの fadc_read_drive ポートの出力を接続する。

sum_patch_last_address_drive 1 bit の入力ポート。fadc_sum_patch モジュールの last_address_drive ポートの出力を接続する。

sum_patch_data_out_valid 1 bit の入力ポート。fadc_sum_patch モジュールの data_out_valid ポートの出力を接続する。

sum_patch_data_out 16 bit の入力ポート。fadc_sum_patch モジュールの data_out ポートの出力を接続する。

craw_patch 1 bit の出力ポート。fadc_craw_patch モジュールの patch ポートの入力に接続する。

craw_patch_busy 1 bit の入力ポート。fadc_craw_patch モジュールの busy ポートの出力を接続する。

craw_patch_drive 1 bit の出力ポート。fadc_craw_patch モジュールの drive ポートの入力に接続する。

craw_patch_read_busy 1 bit の出力ポート。fadc_craw_patch モジュールの fadc_read_busy ポートの入力に接続する。

craw_patch_read_drive 1 bit の入力ポート。fadc_craw_patch モジュールの fadc_read_drive ポートの出力を接続する。

craw_patch_last_address_drive 1 bit の入力ポート。fadc_craw_patch モジュールの last_address_drive ポートの出力を接続する。

craw_patch_data_out_valid 1 bit の入力ポート。fadc_craw_patch モジュールの data_out_valid ポートの出力を接続する。

craw_patch_data_out 16 bit の入力ポート。fadc_craw_patch モジュールの data_out ポートの出力を接続する。

hist_patch 1 bit の出力ポート。fadc_hist_patch モジュールの patch ポートの入力に接続する。

hist_patch_busy 1 bit の入力ポート。fadc_hist_patch モジュールの busy ポートの出力を接続する。

hist_patch_drive 1 bit の出力ポート。fadc_hist_patch モジュールの drive ポートの入力に接続する。

hist_patch_read_busy 1 bit の出力ポート。fadc_hist_patch モジュールの fadc_read_busy ポートの入力に接続する。

hist_patch_read_drive 1 bit の入力ポート。fadc_hist_patch モジュールの fadc_read_drive ポートの出力を接続する。

hist_patch_last_address_drive 1 bit の入力ポート。fadc_hist_patch モジュールの last_address_drive ポートの出力を接続する。

hist_patch_data_out_valid 1 bit の入力ポート。fadc_hist_patch モジュールの data_out_valid ポートの出力を接続する。

hist_patch_data_out 16 bit の入力ポート。fadc_hist_patch モジュールの data_out ポートの出力を接続する。

raw_patch 1 bit の出力ポート。fadc_raw_patch モジュールの patch ポートの入力に接続する。

raw_patch_busy 1 bit の入力ポート。fadc_raw_patch モジュールの busy ポートの出力を接続する。

raw_patch_drive 1 bit の出力ポート。fadc_raw_patch モジュールの drive の入力に接続する。

raw_patch_read_busy 1 bit の出力ポート。fadc_raw_patch モジュールの fadc_read_busy ポートの入力に接続する。

raw_patch_read_drive 1 bit の入力ポート。fadc_raw_patch モジュールの fadc_read_drive ポートの出力を接続する。

raw_patch_last_address_drive 1 bit の入力ポート。fadc_raw_patch モジュールの last_address_drive ポートの出力を接続する。

raw_patch_data_out_valid 1 bit の入力ポート。fadc_raw_patch モジュールの data_out_valid ポートの出力を接続する。

raw_patch_data_out 16 bit の入力ポート。fadc_raw_patch モジュールの data_out ポートの出力を接続する。

data_out_drive 1 bit の入力ポート。このポートに 1 が入力されると FADC データの加工を行い、0 が入力されると一時停止する。

data_out_valid 1 bit の出力ポート。data_out に加工された FADC データが出力されるときこのポートに 1 が出力され、それ以外のときは 0 が出力される。

data_out 16 bit の出力ポート。このポートに加工された FADC データが出力される。

E.2.4.17 fadc_patch_control モジュール

モジュールの説明

FADC データを加工する各モジュールを内部で配置・配線する、最上位モジュール。fadc_read モジュールからはこのモジュールが使われる。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

patch 1 bit の入力ポート。このポートに 1 が入力されると FADC データの加工を開始する。

patch_mode 2 bit の入力ポート。全 FADC モジュールを同一の圧縮モードで処理するとき、このポートにデータ圧縮 IC の圧縮モードを入力する。

each_board_mode 1 bit の入力ポート。FADC モジュール毎にデータ圧縮 IC の圧縮モードを指定するときはこのポートに 1 を入力し、そうではなく全 FADC モジュールを同一の圧縮モードで処理するときは 0 を入力する。

data8_not_16_mode 1 bit の入力ポート。加工したデータを 8 bit 幅で出力するときはこのポートに 1 を入力し、そうではなく 16 bit 幅で出力するときは 1 を入力する。

threshold_mode 1 bit の入力ポート。データ圧縮 IC の SUM 圧縮モードのデータを加工するとき、閾値[1]以下のデータを捨てるときはこのポートに 1 を入力し、そうではなく全てを出力するときは 0 を入力する。

threshold_bit_in 4 bit の入力ポート。threshold_mode ポートに 1 を入力して閾値を使うときは、このポートに閾値を入力する。閾値は 2 の冪乗で指定することに注意する。

fadc_read_busy 1 bit の入力ポート。fadc_read モジュールの busy ポートの出力を接続する。

fadc_read_drive 1 bit の出力ポート。fadc_read モジュールの drive ポートの入力に接続する。

[1]この閾値はデータ圧縮 IC の閾値とは別の FADC I/F 独自のもので、全 FADC チャンネルに対して同一の値を取り、外部からのコマンドで自由に ON/OFF できる。

data_in_valid 1 bit の入力ポート。fadc_read モジュールのdata_out_valid ポートの出力を接続する。

data_in 16 bit の入力ポート。fadc_read モジュールのdata_in ポートの出力を接続する。

data_overflow_in 1 bit の入力ポート。fadc_read モジュールのdata_overflow_out ポートの出力を接続する。

data_board_in 5 bit の入力ポート。fadc_read モジュールのdata_board_out ポートの出力を接続する。

data_board_begin 1 bit の入力ポート。fadc_read モジュールのdata_board_begin ポートの出力を接続する。

data_board_patch_mode 2 bit の入力ポート。fadc_read モジュールのdata_board_patch_mode ポートの出力を接続する。

data_channel_in 5 bit の入力ポート。fadc_read モジュールのdata_channel_out ポートの出力を接続する。

data_channel_begin 1 bit の入力ポート。fadc_read モジュールのdata_channel_begin ポートの出力を接続する。

data_channel_end 1 bit の入力ポート。fadc_read モジュールのdata_channel_end ポートの出力を接続する。

data16_out_drive 1 bit の入力ポート。data8_not_16_mode ポートに 0 を入力して 16 bit 幅で加工されたデータを出力するとき、このポートに 1 が入力されるとデータ加工の処理を行い、0 が入力されると一時停止する。

data16_out_valid 1 bit の出力ポート。data16_out ポートにデータが出力されるときこのポートに 1 が出力され、それ以外のときは 0 が出力される。

data16_out 16 bit の出力ポート。data8_not_16_mode ポートに 0 を入力して 16 bit 幅で加工されたデータを出力するとき、このポートからデータが出力される。

data8_out_drive 1 bit の入力ポート。data8_not_16_mode ポートに 1 を入力して 8 bit 幅で加工されたデータを出力するとき、このポートに 1 が入力されるとデータ加工の処理を行い、0 が入力されると一時停止する。

data8_out_valid 1 bit の出力ポート。data8_out ポートにデータが出力されるときこのポートに 1 が出力され、それ以外のときは 0 が出力される。

data8_out 8 bit の出力ポート。data8_not_16_mode ポートに 1 を入力して 8 bit 幅で加工されたデータを出力するとき、このポートからデータが出力される。

E.2.5 ims_link_c012.v ファイル

E.2.5.1 ims_c012_reset モジュール

モジュールの説明

IMS C012 をリセットするモジュール。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

c012_reset 1 bit の入力ポート。通常は 0 を入力し、IMS C012 をリセットするとき 1 を入力する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

IMS_Reset 1 bit の出力ポート。このポートから IMS C012 のリセット信号が出力される。IMS C012 のReset ピンの入力に接続する。

E.2.5.2 ims_c012_read モジュール

モジュールの説明

IMS C012 からデータを読み出すモジュール。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

read 1 bit の入力ポート。このポートに 1 が入力されると IMS C012 からデータの読み出しを開始する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

select 2 bit の入力ポート。IMS C012 のデータを読み出すレジスタを指定する。

data_out 8 bit の出力ポート。IMS C012 から読み出したデータが出力される。

IMS_notCS 1 bit の出力ポート。IMS C012 のnotCS ピンの入力に接続する。

IMS_RS 2 bit の出力ポート。IMS C012 のRS0, RS1 ピンの入力に接続する。

IMS_RnotW 1 bit の出力ポート。IMS C012 のRnotW ピンの入力に接続する。

IMS_DIN 8 bit の入力ポート。IMS C012 の D0･･･D7 ピンの出力を接続する。

E.2.5.3 ims_c012_write モジュール

モジュールの説明

IMS C012 にデータを書き込むモジュール。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

write 1 bit の入力ポート。このポートに 1 が入力されると IMS C012 へデータの書き込みを開始する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

select 2 bit の入力ポート。IMS C012 のデータを書き込むレジスタを選択する。

data_in 8 bit の入力ポート。IMS C012 に書き込むデータを入力する。

IMS_notCS 1 bit の出力ポート。IMS C012 のnotCS ピンの入力に接続する。

IMS_RS 2 bit の出力ポート。IMS C012 のRS0, RS1 ポートの入力に接続する。

IMS_RnotW 1 bit の出力ポート。IMS C012 のRnotW ピンの入力に接続する。

IMS_DOUT 8 bit の出力ポート。IMS C012 の D0･･･D7 ピンの入力に接続する。

E.2.5.4 ims_link_reset

モジュールの説明

IMS C012 のリセットと初期化を行う。`ims_c012_reset`, `ims_c012_write` モジュールを使用している。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

ims_reset 1 bit の入力ポート。通常は 0 を入力し、IMS C012 をリセットするとき 1 を入力する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

c012_reset 1 bit の出力ポート。`ims_c012_reset` モジュールの`c012_reset` ポートの入力に接続する。

c012_reset_busy 1 bit の入力ポート。`ims_c012_reset` モジュールの`busy` ポートの出力を接続する。

c012_write 1 bit の出力ポート。`ims_c012_write` モジュールの`write` ポートの入力に接続する。

c012_write_busy 1 bit の入力ポート。`ims_c012_write` モジュールの`busy` ポートの出力を接続する。

c012_write_select 2 bit の出力ポート。`ims_c012_write` モジュールの`select` ポートの入力に接続する。

c012_write_data_out 8 bit の出力ポート。`ims_c012_write` モジュールの`data_in` ポートの入力に接続する。

E.2.5.5 ims_link_read モジュール

モジュールの説明

INMOS リンクからデータを読み出す。ims_c012_read モジュールを使用している。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

fifo_full 1 bit の入力ポート。FADC I/F が内部に持つ、命令データ保存用の FIFO の full フラグの値を入力する。

push_data 1 bit の出力ポート。INMOS リンクからデータを読み込んでdata_out ポートに出力されるとき、このポートに 1 が出力される。

data_out 8 bit の出力ポート。このポートに INMOS リンクから読み込んだデータが出力される。

c012_read 1 bit の出力ポート。ims_c012_read モジュールのread ポートの入力に接続する。

c012_read_busy 1 bit の入力ポート。ims_c012_read モジュールのbusy ポートの出力を接続する。

c012_read_select 2 bit の出力ポート。ims_c012_read モジュールのselect ポートの入力に接続する。

c012_read_data_in 8 bit の入力ポート。ims_c012_read モジュールのdata_out ポートの出力を接続する。

IMS_InputInt 1 bit の入力ポート。IMS C012 のInputInt ピンの出力に接続する。

E.2.5.6 ims_link_write モジュール

モジュールの説明

INMOS リンクへデータを書き込む。ims_c012_write モジュールを使用している。

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

busy 1 bit の出力ポート。待機中は 0 を出力し、動作中は 1 を出力する。

timeout 1 bit の出力ポート。未使用。IMS C011 のために用意されたが、IMS C011 が入手困難であったため IMS C012 を使用することになり不要になった。

fifo_empty 1 bit の入力ポート。FADC I/F が内部に持つ、出力データ保存用の FIFO の empty フラグの値を入力する。

pop_data 1 bit の出力ポート。出力データが保存された FIFO からデータを取り出すとき 1 が出力される。

data_in 8 bit の入力ポート。INMOS リンクに出力するデータを入力する。

c012_write 1 bit の出力ポート。ims_c012_write モジュールのwrite ポートの入力に接続する。

c012_write_busy 1 bit の入力ポート。ims_c012_write モジュールのbusy ポートの出力を接続する。

c012_write_select 2 bit の出力ポート。ims_c012_write モジュールのselect ポートの入力に接続する。

c012_write_data_out 8 bit の出力ポート。ims_c012_write モジュールのdata_in ポートの入力に接続する。

IMS_OutputInt 1 bit の入力ポート。IMS C012 のOutputInt ピンの出力を接続する。

E.2.5.7 ims_link_control モジュール

モジュールの説明

入出力ポートの説明

clock 1 bit の入力ポート。クロック信号を入力する。Positive edge で動作する。

reset 1 bit の入力ポート。非同期リセット信号を入力する。Positive edge でリセットが動作する。

clear 1 bit の入力ポート。同期リセット信号を入力する。1 を入力するとクロックの positive edge のタイミングでリセットが動作する。

reset_busy 1 bit の出力ポート。IMS C012 をリセット・初期化している間、1 が出力される。

read_busy 1 bit の出力ポート。INMOS リンクからデータを読み出している間、1 が出力される。

write_busy 1 bit の出力ポート。INMOS リンクへデータを書き込んでいる間、1 が出力される。

write_timeout 1 bit の出力ポート。現在は使用していない。

ims_reset 1 bit の入力ポート。1 を入力すると IMS C012 のリセット・初期化を行う。

fifo_full 1 bit の入力ポート。内部で`ims_link_read` モジュールの`fifo_full` ポートに接続されている。

fifo_empty 1 bit の入力ポート。内部で`ims_link_write` モジュールの`fifo_empty` ポートに接続されている。

push_data 1 bit の出力ポート。内部で`ims_c012_read` モジュールの`push_data` ポートに接続されている。

pop_data 1 bit の出力ポート。内部で`ims_c012_write` モジュールの`pop_data` に接続されている。

data_in 8 bit の入力ポート。内部で`ims_c012_write` モジュールの`data_in` 接続されている。

data_out 8 bit の出力ポート。内部で`ims_c012_read` モジュールの`data_out` ポートに接続されている。

IMS_Reset 1 bit の出力ポート。IMS C012 のReset ピンの入力に接続する。

IMS_notCS 1 bit の出力ポート。IMS C012 のnotCS ピンの入力に接続する。

IMS_RS 2 bit の出力ポート。IMS C012 のRS0, RS1 ピンの入力に接続する。

IMS_RnotW 1 bit の出力ポート。IMS C012 のRnotW ピンの入力に接続する。

IMS_InputInt 1 bit の入力ポート。IMS C012 のInputInt ピンの出力を接続する。

IMS_OutputInt 1 bit の入力ポート。IMS C012 のOutputInt ピンの出力を接続する。

IMS_DIN 8 bit の入力ポート。IMS C012 の D0･･･D7 ピンの出力を接続する。

IMS_DOUT 8 bit の出力ポート。IMS C012 の D0･･･D7 ピンの入力に接続する。

E.2.6 logiblox_ram.v ファイル

E.2.6.1 logiblox_prim_async_ram モジュール

モジュールの説明

aAVerilog-HDL ソースコードでのシミュレーション時に使われる、FPGA 内部の RAM の動作を定義している。論理合成時には FPGA を製作している会社の提供している回路に置き換えられるため、使用されない。Parameter で RAM に記憶するデータの bit 幅と個数を変更できる。

入出力ポートの説明

A RAM のアドレスを入力するポート。`ADDRS_WIDTH` パラメータで bit 幅を変更できる。

DO RAM のデータが出力されるポート。`DATA_WIDTH` パラメータで bit 幅を変更できる。

DI RAM のデータを入力するポート。`DATA_WIDTH` パラメータで bit 幅を変更できる。

WR_EN 1 bit の入力ポート。このポートに入力される信号の positive edge で RAM のA ポートのアドレスに`DI` ポートの値を書き込む。

E.2.6.2 logiblox_async_ram256x1 モジュール

モジュールの説明

Channel kill 情報を記憶する RAM。

入出力ポートの説明

A 10 bit のアドレスを入力するポート。

DO 1 bit のデータを出力するポート。

DI 1 bit のデータを入力するポート。

WR_EN 1 bit の入力ポート。このポートに入力される信号の positive edge で RAM のAポートのアドレスにDIポートの値を書き込む。

E.2.6.3 logiblox_async_ram32x5 モジュール

モジュールの説明

Board order 情報を記憶する RAM。

入出力ポートの説明

A 5 bit のアドレスを入力するポート。

DO 5 bit のデータを出力するポート。

DI 5 bit のデータを入力するポート。

WR_EN 1 bit の入力ポート。このポートに入力される信号の positive edge で RAM のAポートのアドレスにDIポートの値を書き込む。

E.2.6.4 logiblox_async_ram32x2 モジュール

モジュールの説明

データ圧縮 IC の圧縮モードを記憶する RAM。

入出力ポートの説明

A 5 bit のアドレスを入力するポート。

DO 2 bit のデータを出力するポート。

DI 2 bit のデータを入力するポート。

WR_EN 1 bit の入力ポート。このポートに入力される信号の positive edge で RAM のAポートのアドレスにDIポートの値を書き込む。