2023 年度　卒業論文

sPHENIX 実験中間飛跡検出器 INTT のための データ処理 FPGA の改良

奈良女子大学　理学部
物理科学科　高エネルギー物理学研究室

加納麻衣

2023 年 3 月 31 日

現在、奈良女子大学高エネルギー物理学研究室は、アメリカブルックヘブン国立研究所にて 2023 年より稼働予定の sPHENIX 実験に用いられる中間飛跡検出器 (INTT) の開発を行っている。INTT グループには奈良女子大学のほか、理化学研究所、立教大学、BNL、国立中央大学 (National Central University)、国立台湾大学 (National Taiwan University) 等が参加している。INTT には、Readbacker と呼ばれる設定値読み出しシステムがある。PC から読み出しチップに送られた設定値が正しく送られているか確かめるのに必要である。先行研究で、1.2m あるデータケーブル Bus-Extender を接続した時に、読み出した値が設定値と異なり、Readbacker が正常に動作しないことが確認されていた。原因は読み出しシステム内にあるデータ処理 FPGA にあり、問題解決の為 FPGA の改良を行う必要がある。本研究では、まず FPGA を改良する為の開発環境のテストを行い、その後 FPGA コードの改良を行った。

目次

図目次

第 1 章

序論

1.1 素粒子物理学

1.1.1 素粒子の標準理論

素粒子とは、物質を構成する最小単位の粒子であり、全ての物質は素粒子からできていると考えられる。素粒子には、スピン $\frac{1}{2}$ をもつフェルミ粒子と、整数スピンをもつボース粒子が存在する。さらにフェルミ粒子はレプトンとクォークに分類される。レプトンは、電子 (e)、ミュー粒子 (μ)、タウ粒子 (τ) の 3 種類と、これらに対応する電子ニュートリノ (νe)、ミューニュートリノ ($\nu\mu$)、タウニュートリノ ($\nu\tau$) を合わせた、計 6 種類ある。クォークは、u(up)、d(down)、s(strange)、c(charm)、b(bottom)、t(top) の 6 種類ある。ボース粒子は、スピン 1 のゲージ粒子と、スピン 0 のヒッグス粒子に分類される。ゲージ粒子には、強い相互作用を伝えるグルーオン、電磁相互作用を伝える光子、弱い相互作用を伝える W 粒子と Z 粒子が存在する。グルーオンにより、クォーク同士がハドロン内部において強い力で結合している。この強い力を記述する量子色力学がある。量子色力学とは強い相互作用を色と呼ばれる量子状態の間に働く力として扱う理論であり、クオークが遠距離にあるときは強く、接近する時は弱いという特徴を持つ。また標準理論には電磁気＋弱い力を記述する電弱理論 (Weinberg+Salam) もある。　図 1.1はこれらの素粒子の一覧を示している。

1.1.2 クォーク・グルーオン・プラズマ（QGP)

通常の温度・密度では、ハドロン内部でクォークとグルーオンは強い相互作用をしているため、それらを単体で取り出すことはできない。これは、クォークの閉じ込めと呼ばれている。しかし、超高温・超高密度下では、この閉じ込めが破れることで、クォークとグルーオンがばらばらになり、プラズマ状態が観測される。この状態を、クォーク・グルーオンプラズマ (Quark-Gluon Plasma: QGP) という。QGP は、ビッグバン後の数 10 μ 秒間の初期宇宙で実現していたと考えられている。

図1.1　素粒子標準理論に登場する素粒子の一覧 (大強度用紙加速器施設 J-PARC 素粒子・原子核研究より)

1.2　研究背景

1.2.1　Relativistic Heavy Ion Collider(RHIC)

米国ブルックヘブン国立研究所 (BNL) には、RHIC という世界初の重イオン衝突型加速器があり、QGP を生成し、その性質を解明することが目的である。2 つの加速器リングからなり、その周長は約 3.8km である。これらのリングが交差する衝突点は 6 カ所設けられている。金・金原子核衝突、陽子・陽子衝突などが行われており、最大重心系エネルギーはそれぞれ 200GeV、510GeV である。図 1.2は RHIC の全体写真である。

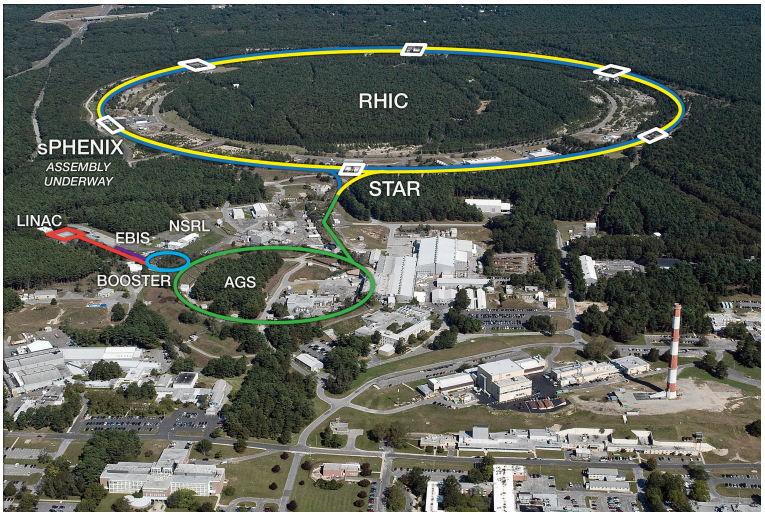

図1.2　RHIC の全体

1.2.2　sPHENIX 実験

sPHENIX(the super Pioneering High Energy Nuclear Interaction eXperiment) 実験は、BNL の RHIC を用いて行われる、2023 年稼働予定の実験である。この実験は、2000 年から 2016 年まで行われていた PHENIX 実験を高度化したものである。QGP における輸送係数の温度依存性や色電荷のデバイ遮蔽長を決定することを目的に、ハドロンジェットやウプシロン粒子の観測が計画されている。衝突から発生した荷電粒子の飛跡を検出する飛跡検出器は、内側から MVTX(MAPS based VerTeX Detector)、INTT(INTermediate Tracking detector)、TPC(Time Projection Chamber) の 3 つで構成されている。図 1.3は sPHENIX 実験の飛跡検出器の全体である。

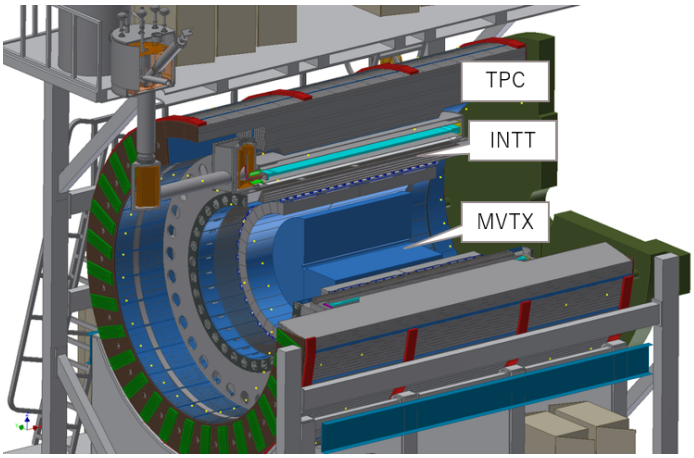

図1.3　sPHENIX 実験の検出器

第 2 章

中間飛跡検出器 INTT

INTT は、飛跡検出器群のうち MVTX と TPC の間で、ビームパイプから 6cm-12cm にあるバレル状の 2 層構造のストリップ型シリコン検出器である。図 2.1は INTT を Z 軸に沿って半分にした断面図を表している。位置分解能と時間分解能が高く、飛跡再構築において重要な役割を果たす。衝突中心からビーム軸方向に ±23cm、方位角方向に対して 2 の範囲を覆っている。INTT には、56 本のラダーが用いられる。INTT で検出した大量のデータは離れた後段の読み出し回路へ伝送・処理される。

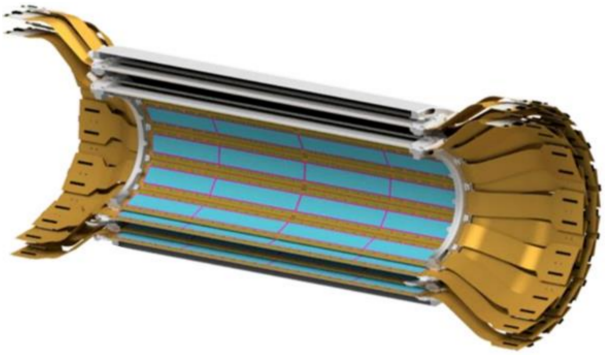

図2.1　INTT を Z 軸に沿って半分にした断面図

2.1　INTT ラダー

センサー部分である INTT ラダーは、シリコンセンサー、FPHX チップ、HDI で構成されている。

2.1.1　シリコンセンサー

INTT 用シリコン検出器では、ストリップ型のシリコンセンサーが使われている。ストリップ長の異なる 16mm×9.984mm(Type-A) と 20mm×9.984mm(Type-B) があり、ラダーは 2 つのハーフラダーでできており、ハーフラダーは A センサー 1 つと B センサー 1 つがある。各セルは 128 個のストリップに分割され、セル 1 個に 1 つの読み出しチップ (FPHX チップ) が接続されている。これ以降、セルを chip、ストリップを channel と呼ぶ。図 2.2はシリコンセンサーの Type-A と Type-B の配置を表している。

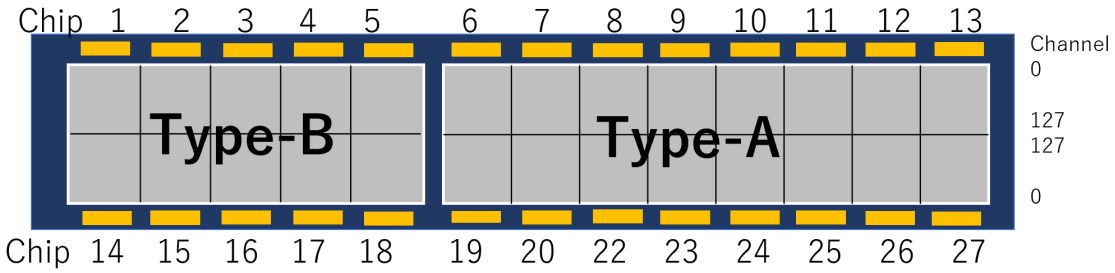

図2.2　シリコンセンサーの Type-A と Type-B の配置図

2.1.2　FPHX チップ

FPHX チップは、シリコンセンサーから読み出したアナログ信号をデジタル値に変換し、後段のモジュールに送り出す役割をもつ。ハーフラダーあたり 26 個の FPHX チップが搭載されている。1FPHX チップあたり 128 個の読み出しチャンネルを持っており、各チャンネルで波形整形を行って 3bit の ADC を出力する。ADC の閾値は 8 bit の DAC 値で設定可能で、このときの電圧変換は $\mathrm{V[mV]} = 210 + 4 \times \mathrm{DAC}$ である。

2.1.3　High Density Interconnect(HDI)

HDI は FPHX チップへの入出力配線と FPHX チップやシリコンセンサーへの電源供給を行う基板である。

2.2　設定値読み出しシステム Readbacker

INTT では Readbacker という設定値読み出しシステムがある。DAC 値などの測定に必要な設定値を PC で設定し、ラダー内にある FPHX チップまで送られる。FPHX チップでの設定値が正しく設定されていることを確認するものであり重要なシステムである。Readbacker は、ROC、Bus-Extender、Conversion Cable、FEM、FEM-IB、PC で構成されており、図 2.3はその構成である。各要素は以下で説明する。

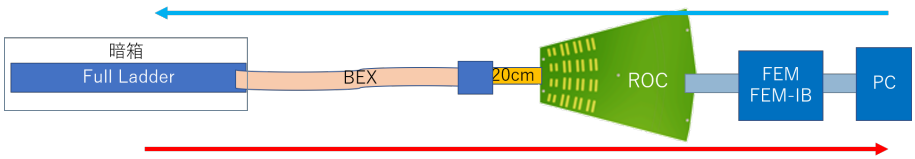

図2.3　Readbacker の構成

図 2.3中の青矢印は設定値を送る、赤矢印は設定値を読み出す順番を表している。

2.2.1　Read Out Card(ROC)

ROC は、複数の FPHX チップから送られてきたデータの同期と結合を行い、次の読み出し回路に転送する読み出し基板である。ROC1 枚あたり 4×4 個のラダーからのデータを取り扱うことができる。ROC にはデ

ータ処理用 FPGA が搭載されており、シリコンラダーから転送されたデータを整形する。また、FPHX チップへの測定条件や閾値といった設定値を指定されたチップへ転送する機能や FPHX チップの読み出し回路の動作確認を行うためのキャリブレーションパルスを発生させる機能、センサーモジュールへの電源供給機能も持つ。

2.2.2 Bus-extender

Bus-extender(BEX) は、ROC と HDI を接続するためのデータ送信ケーブルであり、その全長は 1.2 mである。

2.2.3 Conversion Cable

Conversion Cable は、ROC と HDI で異なるコネクターを整合するためのデータ送信ケーブルで、全長は 20cm と 40cm の 2 種類がある。sPHENIX 実験では、20cm の Conversion Cable と Bus-extender を併用する。

2.2.4 Front End Module（FEM)

FEM とは、VME 規格の読み出し基板である。ROC から受信したデータをまとめ、sPHENIX 共通のフォーマットに変換し、他の検出器の情報と結合を行うモジュールへと転送する。

2.2.5 FEM-Interface Board(FEM-IB)

FEM-IB とは、FEM と同様に VME 規格の基板で、FEM 全体を制御する役割を持つ。主に、検出器全体を統括するクロック信号やトリガー信号、FEM 制御信号を受け取る。

2.3 Readbacker の課題

2.3.1 課題

ラダーと ROC の間で Bus-Extender を繋げない場合、設定値と読み出し値は一致し Readbacker は正常に動作する。しかし Bus-Extender を接続した場合、設定値と読み出し値が一致せず Readbacker が正常に動作しない問題がある。実際の様子は図 2.4のようになっている。これは PC で DAC 値を設定し、読み出し値を確認する画面である。左は設定値、右は読み出し値である。Readbacker が正常に動作すると、左右は同じ値が表示される。しかし Readbacker が正常でないときは図 2.4のように右に表示される値は左と一致しない。これでは正しく FPHX チップに設定値を送ることが出来ているか分からない。

2.3.2 原因

先行研究では Bus-Extender を接続した場合に正常に動作しない原因を調べた。Bus-Extender を接続している状態で FPHX と ROC の間の読み出し値と、FEM で受信した読み出し値をオシロスコープで測定した。その結果が以下の図 2.5となる。

DAC0	8	16
DAC1	16	32
DAC2	30	60
DAC3	35	71
DAC4	40	80
DAC5	45	91
DAC6	50	100
DAC7	55	111

図2.4　Readbacker が正常に動作しない様子

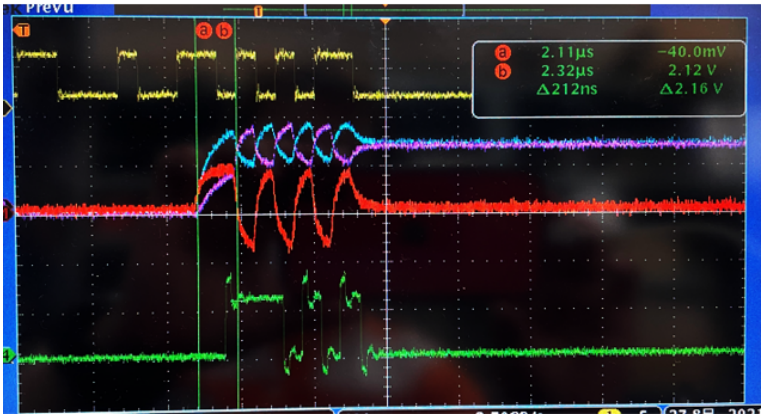

図2.5　オシロスコープで測定した結果

一番上の波形は PC から FPHX に送った設定値 (10101011)、水色とピンクと赤の波形は FPHX と ROC 間の読み出し値 (11010101) を表示している。緑の波形は FEM で受信した読み出し値 (11101010) となっている。ここで読み出し値は設定値に対して順番が反転して出てくる。なので設定値が 10101011 の場合、Readbacker が正常だと読み出し値は 11010101 となる。この測定では、FPHX と ROC 間の読み出し値は設定値と反転しているので、この時点では読み出し値は正しく送られている。しかし FEM で受信した時点では読み出し値は異なっており、これより ROC 前後で読み出し値が変わってしまっていることが分かった。
さらに PC から送る設定値を変更して、同様に測定を行った結果が以下のようになった。

この結果より、ROC 前の 1bit 目と ROC 後の 2bit 目がいつも同じになることが分かった。これより読み出し距離が長いためデータが遅れ、受信タイミングがずれてしまっており、1bit 分データが遅れて届ており、ROC でこの読み出し値の送受信を行っているのが FPGA で、そこに原因があると考えられた。

設定値	読み出し値ROC前	読み出し値ROC後
10101011	11010101	11101010
10101010	01010101	00101010
10101101	10110101	11011010

図2.6　先行研究の結果

2.4　FPGA

FPGA(field programmable gate array) とは、現場でプログラム可能な論理回路配列である。ハードウェア記述言語 (HDI) を用いてソースコードを作成し、論理合成や配置配線などを行い、FPGA に書き込み論理回路を設計する。

2.4.1　Slowcontrol FPGA

ROC 内にある複数の FPGA で、Readback データの送受信しているのは、SlowControlFPGA である。図 2.7は ROC の写真であり、赤枠で囲っているのが SlowControlFPGA である。SlowControl とは FPHX を制御する機能を持ち、FPHX の DAC 値などを設定したり、データ収集を開始・停止したり、また設定した値を読み出すことなどを行うことが出来る。Readbacker は SlowControl の一部である。SlowControl で FPHX を制御するコマンドが図 2.8の 32bit 長データである。

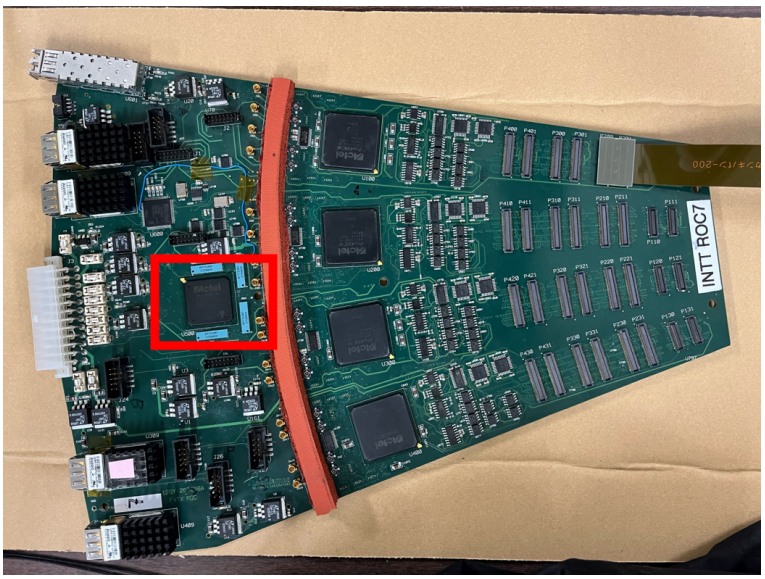

図2.7　ROC

信号の内容は先頭である Header の 7bit(1100111) と後ろの Trailer の 4bit(0000) は固定であり、Header の次から Chip-ID(5bit)、RegAddr(5bit) は DAC 値などどこの設定値なのか、Instruction(3bit) は動作 (write など)、Data(8bit) と構成されている。この中でも Data が FPHX チップに送られる設定値であり、これを受信する。

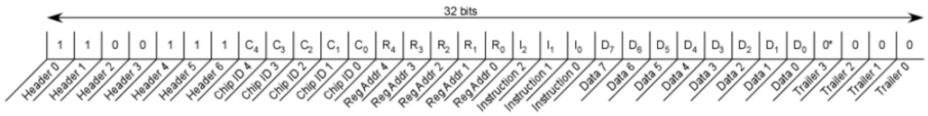

Figure 7 - The Slow Control Word

図2.8　Slowcontrol の内容

2.4.2　SlowControl FPGA コード

SlowControlFPGA のコードは、HDL の種類の 1 つである VHDL という言語で書かれている。その中で Readback データを受信するモジュールは、「Status serial output」というものであり、その中でも読み出し値の受信タイミングについて記述されている箇所を図 2.9で示す。

```
    elsif falling_edge(BCO_CLK) then

--Create an FPHX send_command trigger, and upon this trigger start a counter which will
--tell us when to extract the status word from the FPHX SC line:
SEND_COMMAND_BUF <= SEND_COMMAND;
            SEND_COMMAND_2BUF <= SEND_COMMAND_BUF;
            SEND_COMMAND_TRIG <= SEND_COMMAND_BUF and (not SEND_COMMAND_2BUF);

            if SEND_COMMAND_TRIG = '1' then
                  CE <= '1';
            end if;

--After 24 clocks, read the 8-bit FPHX status word:
            if ADDR = "010010" then
                  READ_DATA_FPHX <= '1';
            end if;
            if ADDR = "011010" then
                  READ_DATA_FPHX <= '0';
                  CE <= '0';
            end if;

            if (READ_DATA_FPHX = '1') then
               STATUS_DATA <= SC_FROM_FPHX_int;
               CS_DATA <= '1';
            else
               STATUS_DATA <= '0';
               CS_DATA <= '0';
end if;
```

図2.9　Status serial output の内容

2.4.3　データ受信の仕組み

図 2.9より、受信タイミングはクロック信号と同期しており、受信開始と終了が固定されている。受信の長さも 8bit となっているので、データが遅れることなく送られてくると、データの 8bit と受信タイミングが合う設計になっている。図 2.10は正しくデータが読み出せた時のクロックとデータのタイミングを示している。しかし図 2.11のように、タイミングは動作を開始してから必ず 21 クロック目に受信開始をすることとなっているので、データが遅れてくると、受信がずれてしまう。

2.4.4　解決方法

Bus-Extender を接続していない場合は、データの受信タイミングはズレない。Bus-Extender を接続するとデータが遅れ 1bit 分受信タイミングがズレてしまう。どちらの場合でもデータ 8bit 分を正しく読み出すために、受信の長さを 8bit から 9bit に変更する。具体的には図 2.9のコードの「ADDR」が受信タイミングを決めているので、終了タイミングについて記述している「ADDR ＝”011010”」を「ADDR=” 011011”」に変更する。

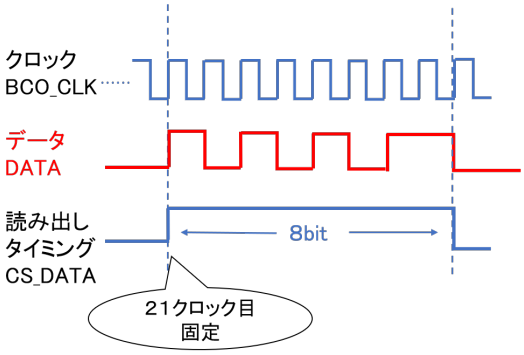

図2.10　データが遅れていないとき

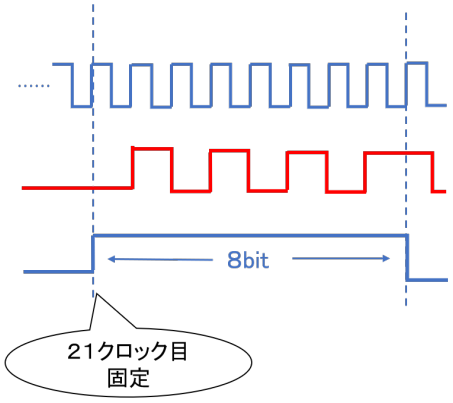

図2.11　データが遅れているとき

2.5　研究の目的

本研究の目的は、上記で説明した INTT で Bus-Extender ををを接続した際に設定値読み出しシステムが正常に動作しない問題を解決するため、原因となっている ROC 内にある SlowControlFPGA の改良である。

第 3 章

開発環境ツールのテスト

FPGA を改良する前に、FPGA 開発を行うツールを操作方法の確認とテストを行った。

3.1　FPGA 開発

まず FPGA 開発の流れは図 3.1となる。まず FPGA に実装する内容を仕様設計を行う。次に HDL を使い論理回路の動作を記述する。論理合成では HDL で記述されたデザインを具体的な回路に変換を行う作業である。配置配線とは FPGA の構造に応じて内部の配置やピンの位置、そして配線などを行う。またここまでの中で設計した論理回路の動作の確認検証を行うシミュレーションも行うことが出来る。そしてダウンロードファイルの作成を行い、基板上の FPGA にパソコンからダウンロードを行う。この動作をプログラミングともいう。そして実機での動作確認を行う。

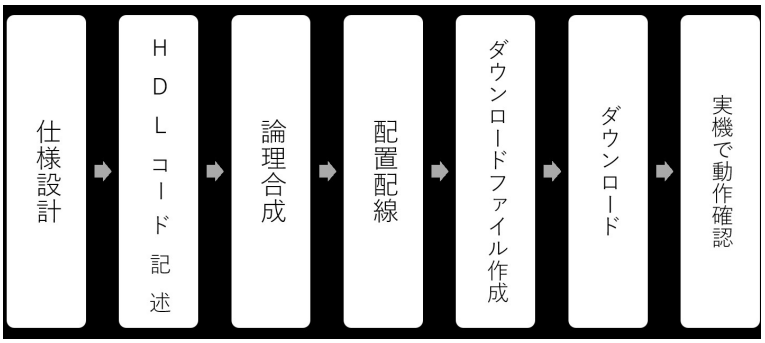

図3.1　FPGA 開発の流れ

3.2　FPGA 開発ツール

本研究で改良を行う FPGA は Microchip 社の ProAsic3E である。この FPGA の開発には、同社が提供する設計ツール LiberoSOC を用いる。

3.2.1　Libero SoC

図 3.1において、HDL コード～ダウンロードファイル作成までを行うことが出来る FPGA 向けの統合開発ツールである。バージョンは V11.9 は ProAsic3E 用の最終版であり、以前 ROC が用いられていた検出器 FVTX の開発には v9.1 が使われていた。本研究では v11.9 を用いた。セットアップにはダウンロードの他にライセンスの入手とライセンスサーバーの設定が必要である。

3.2.2　FrashPro

FPGA 開発において、プログラミングを行うことができるソフトウェア。Libero SoC から立ち上げることも可能である。プログラミング以外に、Verify という FPGA に書き込まれている内容と PC 上にある FPGA プログラミングファイルの内容が一致するかどうかの確認する動作も出来る。

3.2.3　Frash Pro5

基板上に実装された FPGA を PC と接続しプログラミングするハードウェア。FPGA の JTAG ポート経由でプログラム可能である。

図3.2　FlashPro5

3.3　回路設計テスト

まず、Libero SoC を用いて FPGA の開発する手法を習得するために、簡単な例題として半加算器を FPGA 中に構築し、論理シミュレーションを行った。ここで設計した回路は図 3.3である。そしてこの回路の真理値表は図 3.4となる。

この回路を設計した HDL コードは図 3.5、ここでは実際に改良する FPGA に合わせて VHDL で記述した。

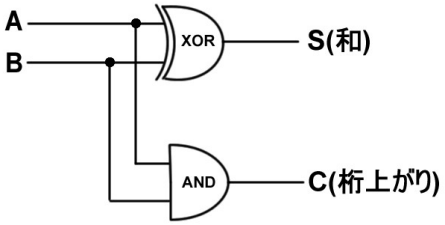

図3.3　設計した回路

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

図3.4　真理値表

```
------------------------------------------------------------

library IEEE;

use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity sample1 is
port (
    A:in std_logic;
    B:in std_logic;
    S:out std_logic;
    C: out std_logic);
end sample1;

architecture architecture_sample1 of sample1 is

begin
S<=A xor B;
C<=A and B;
   -- architecture body
end architecture_sample1;
```

図3.5　VHDL 記述内容

この設計したコードでシミュレーションを行った結果が図 3.6 である。上から順に A,B,S,C の信号を示しており、真理値表と一致することが確認できる。このことから Libero SoC で VHDL コードで回路を設計し正しくシミュレーションできることが確認できた。

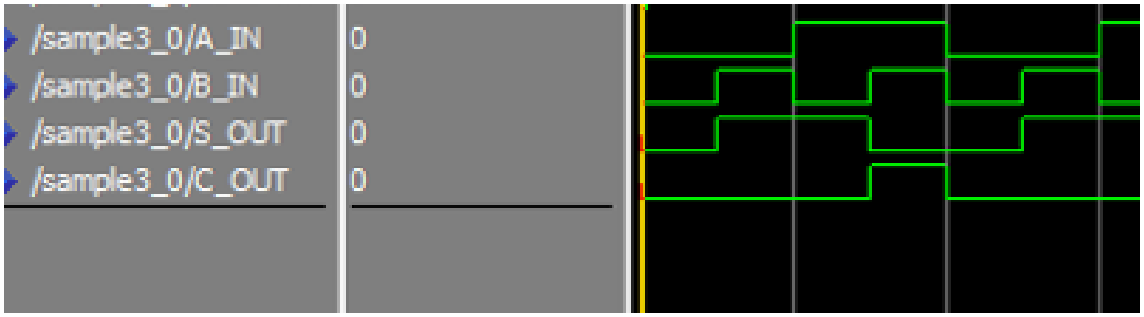

図3.6　シミュレーションした結果

3.4　FPGA コード

ここからは SlowControlFPGA コードを用いてテストを行った。コードは ROC が以前用いられていた検出器 FVTX に関する Web ページ上にあるものを使用した。まず FPGA コードの内容を確認する作業を行った。

3.4.1　Verify

他の測定で使用中の ROC を用いて、FPGA の書き換えを行う前に FPGA 内で設計されている内容と、コードの内容が一致しているか確認する作業である Verify を行った。結果は図 3.8のようなエラーが出た。FVTX の Web ページにあるその他のコードを全て試したが、一部のコードは異なるエラーが、それ以外は全て同じエラーが出た。Verify を行ったコードは図 3.7に示す。エラーの内容より、Web ページに載っているコードは全て現在使われている ROC の FPGA の内容と一致していないことが分かった。

3.4.2　コードのコンパイル

FVTX の Web ページにある最新の FPGA コード（12-Sep-12,fast ファイル名 ROCslowcontrol.zip) で HDL コード記述の文法の確認と修正、論理合成、配置配線、ダウンロードファイルの作成までを行った。このファイルを FPGA 書き換えテストに利用した。

Webページの表示	ファイル名
12-Sep-12 fast	ROC_slow_control.zip
12-Sep-12 slow	ROC_slow_control_slow_5Feb13.zip
5-Sep-12 fast	ROC_slow_control_NoReadClkDelay_94Mhz.zip
29-May-12 slow	ROC_slow_control_slow_change_clock_edge.zip
21-May-12 slow	ROC_slow_control_slow_21May12.zip
21-May-12 fast	ROC_slow_control_fast_21May12.zip
1-Feb-12 slow	ROC_slow_control_slow.zip
1-Feb-12 fast	ROC_slow_control_fast.zip
31-Jan-12 fast	ROC_slow_control_fast_sc_enable_reset.zip
31-Jan-12 slow	ROC_slow_control_slow_sc_enable_reset.zip
26-Jan-12 slow	ROC_slow_control_nodelay_nosyncreset_slow.zip
24-Jan-12 fast	ROC_slow_control_fast.zip
24-Jan-12 slow	ROC_slow_control_slow.zip
23-Jan-12 fast	ROC_slow_control.zip
23-Jan-12 slow	ROC_slow_control_slow.zip
9-Jan-12	ROC_slow_control.zip
7-Dec-11	ROC_slow_control.zip
2-Dec-11	ROC_slow_control.zip
21-Nov-11	ROC_slow_control.zip
10-Nov-11	ROC_slow_control.zip
8-Nov-11	ROC_slow_control.zip
4-Nov-11	ROC_slow_control.zip

図3.7　Verify をしたコード一覧

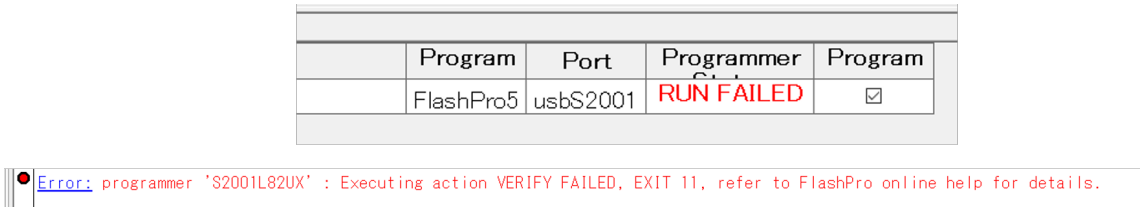

図3.8　Verify でエラーがでた様子

3.5　FPGA 書き換えテスト

次に FlashPro を用いて FPGA に書き込むことが出来るのか、また正しく書き換えを行えるかどうかのテストを行った。

3.5.1　書き換え

ここからは ROC は測定に使えない古い ROC 図 3.9を使用した。なぜなら FPGA を書き換えてしまうと、もともと設計されていた内容のコードを持っていなければ元の状態に戻せないので、まず書き換えのテストを行うにあたって、他の測定に用いられている ROC を使えない。書き換えテストのセットアップは図 3.11で、

ROC に電源を流し、PC と FPGA は FlashPro5 を用いて接続している。PC の画面は FlashPro のソフトウェアである。

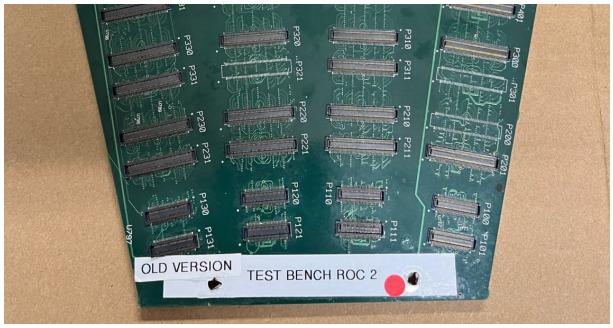

図3.9　古い ROC

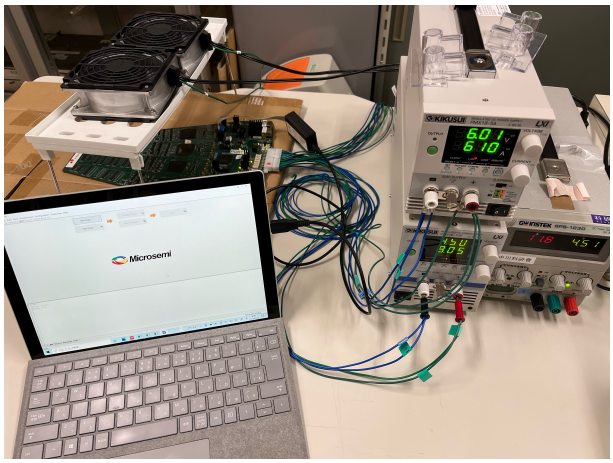

図3.10　書き換えテストのセットアップ

先ほどのダウンロードファイル作成まで行った SlowControlFPGA コードを古い ROC の FPGA に書き込むと、図のように表示されプログラミングに成功した。またそのまま同じコードで Verify を行うとこれも成功した。ここで Libero SoC を用いて FPGA コードを開発し、それを FPGA に書き込んで、正しく書き込めたことを検証するまでの書き込み課程のテストが出来た。

	Program	Port	Programmer	Program
	FlashPro5	usbS2001	RUN PASSE	☑

図3.11　書き込みに成功した様子

3.5.2　動作確認

次にコードの内容を変更したものを書き込み、それが反映されるかのテストを行った。ピンの一カ所を図3.12のように変更した。

```
 test
process (CLK_TX)
begin


  if rising_edge(CLK_TX) then
    TEST_CTR <= TEST_CTR + "00000000000000000001";
    TEST_BIT <= not TEST_BIT;
  end if;
end process;
TEST_OUT <= TEST_CTR(10);
```

図3.12　変更したコード内容

それをオシロスコープで変更したピンを測定した結果、図 3.13のように H/L を出力することが出来た。

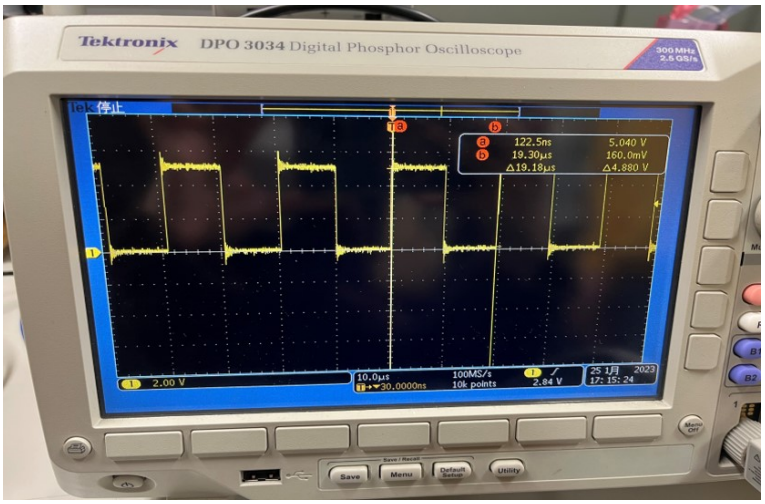

図3.13　書き換えテストでのオシロスコープの様子

またデジタル回路上の多数信号を表示するツールである Identify を用いて、同じように書き込んだ FPGA 内でどのように動作しているかを確認した。ここでは PC と FPGA は FlashPro5 で接続されている。Identify の結果は図 3.14となり、コードを変更した通りに動作していることが確認できた。

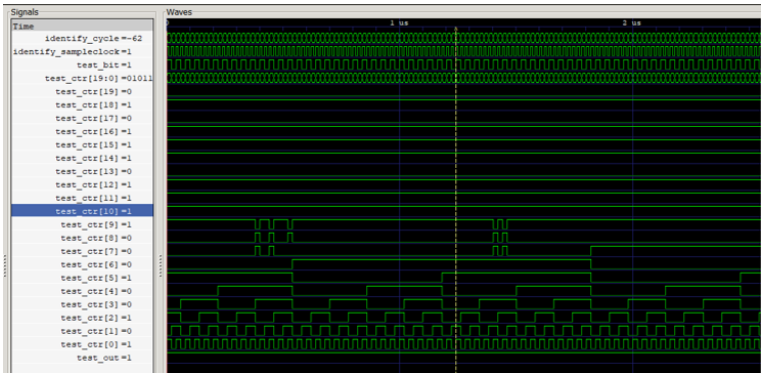

図3.14　Identify の様子

ここまでで、正常にコードの内容を変更したものを FPGA に書き込むことが出来ることが確認できたので、開発ツールのテストは完了した。次に研究目的である FPGA の改良を行っていく。

第 4 章

FPGA の改良

4.1　FPGA コードの改良

4.1.1　変更点

2.4.4 で示したとおり、データ受信のタイミングの長さを 8bit から 9bit に変更を行った。ここで用いたコードは、先程第 3 章で変更した内容は削除している。8bit から 9bit に変更した内容は

```
--Create an FPHX send_command trigger, and upon this trigger start a counter which will
--tell us when to extract the status word from the FPHX SC line:
SEND_COMMAND_BUF <= SEND_COMMAND;
  SEND_COMMAND_2BUF <= SEND_COMMAND_BUF;
  SEND_COMMAND_TRIG <= SEND_COMMAND_BUF and (not SEND_COMMAND_2BUF);

  if SEND_COMMAND_TRIG = '1'  then
      CE <= '1';
  end if;

--After 24 clocks, read the 8-bit FPHX status word:
  if ADDR = "010010" then
      READ_DATA_FPHX <= '1';
  end if;
-     if ADDR = "011010" then
      if ADDR = "011011" then -- TH 2023.2.8 send 9 bit length including 8bit word
          READ_DATA_FPHX <= '0';
-         CE <= '0';
      end if;

      if ADDR = "011111" then -- add by TH 2023.2.8 to wait 32 clocks correspoding to SC command length
          CE <= '0';
      end if;
```

図4.1　変更したコード

4.1.2　シミュレーション

変更したコードで、読み出し値を正しく受信できるかシミュレーションを行った。その前にコードの変更前、受信の長さが 8bit のままでシミュレーションを行った。設定値は 11010101 としたので、正しく読み出せていれば読み出し値は 10101011 となる。シミュレーションの結果の図は上段が受信したデータであり、下段は受信タイミングを表示している。変更前のシミュレーションで、読み出しが正常に出来た状態は図 4.2となり、読み出し値は 10101011 となり正しい。データが遅れてきた状態は図 4.3であり、01010101 となっており、確かにデータ受信タイミングがずれることを確認できた。

そしてコードを変更した状態でシミュレーションを行った結果が図 4.4である。読み出し値は 010101011 となっており、9bit 読み出し、信号が 1bit 遅れてきても正常に読み出せることを確認できた。

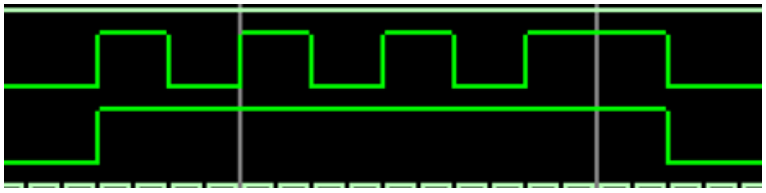

図4.2　変更前: 読み出し正常

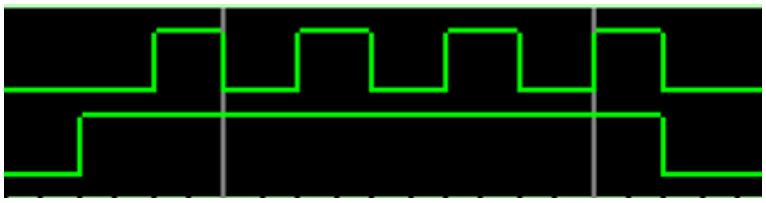

図4.3　変更前: データ遅れ

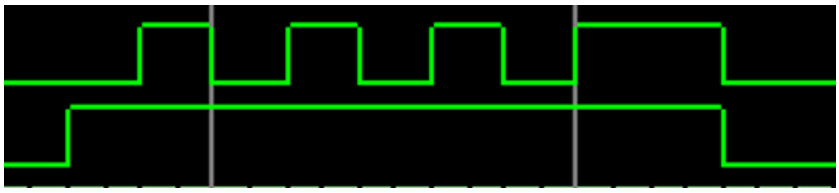

図4.4　変更後

4.2　FPGA 書き換えと動作確認

実際に FPGA に変更したコードを書き込み、実機での動作確認を行った。確認にはキャリブレーションテストを行った。キャリブレーションテストとは、テストベンチ全体の動作確認を行うためのテストである。ROC で生成されたテストパルスは、FPHX チップでデジタル信号に変換され、ROC,FEM を通り PC でデータとして出力される。ROC はキャリブレーションテストができる ROC7 を用いて、ラダーと電源と Conversion Cable、FEM、FEM-IB を接続し、変更したコードを FPGA に書き込みキャリブレーションテストを行った。しかしキャリブレーションデータが来ず、何度行ってもデータが一切来ないか、数回はノイズのみ来る状態で、キャリブレーションテストが出来なくなった。(図 4.5) また他に異常があるか確認したところ、本来キャリブレーションテストの時 FEM でランプが点灯しない箇所で点灯していることが確認できた。(図 4.6、図 4.7)

ここで内容を変更する前のコードを複数同様に FPGA に書き込みキャリブレーションテストを行った。一つは今まで書き換えテストや変更に使っていたコード。他は FVTX の Web ページにある別のコードである、12-Sep-12,slow version ファイル名 ROCslowcontrolslow5Feb13.zip 5-Sep-12,fast ファイル名 ROCslowcontrolNoReadClkRelay94Mhz.zip この二つを用いた。しかしいずれも結果は同じくキャリブレーションテストは正常に動作しなかった。

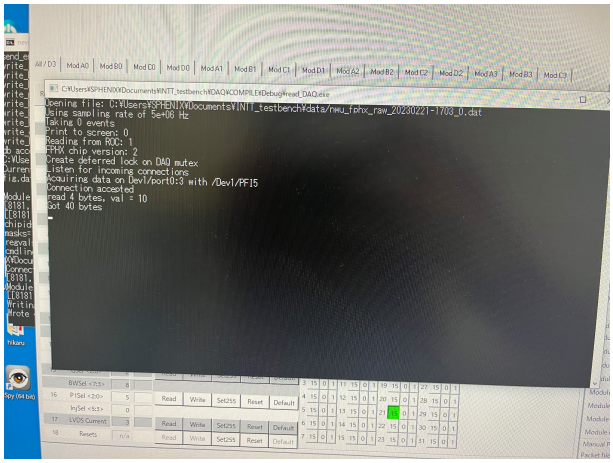

図4.5　キャリブレーションデータが来ない PC の様子

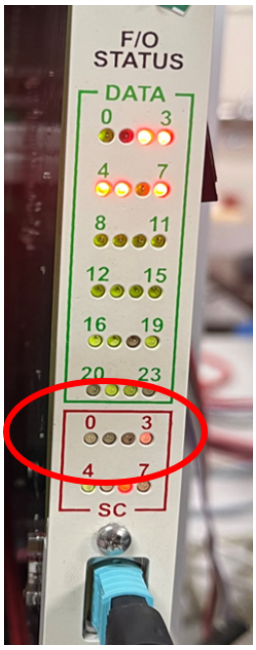

図4.6　キャリブが正常に動作している時の FEM

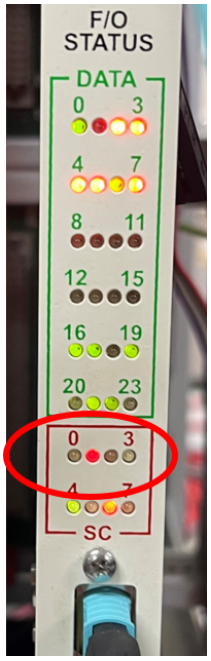

図4.7　キャリブが正常に動作していない時の FEM

4.3　結果

Slowcontrol FPGA コードで読み出し値の受信タイミングの長さを 8bit から 9bit に変更したもので、FPGA を書き換えキャリブレーションテストを行ったが、キャリブレーションデータが取れなくなった。また改造前のコードを書き込みキャリブレーションテストを行ったが、同じく出来なかった。よってコードの内容を変更したことによっての異常ではなく、FVTX の Web ページにあるコードがキャリブレーションテストは行えないことが分かった。

第 5 章

結論と今後の課題

FPGA 開発のツールのテストは完了し、FPGA を正常に書き換え出来ることも確認できた。また FPGA コードで読み出しの長さを 8bit から 9bit に変更し、シミュレーションでデータが 1bit 遅れても正常に読み出し値を受信できるようになった。しかし FVTX の Web ページにある SlowControlFPGA のコードではキャリブレーションテストが出来ず、またキャリブレーションテストの出来る ROC の FPGA の中身と改良に用いたコードの内容一致するかの確認（Verify）を行ったが一致しないというエラーが出た。これらより、FVTX の Web ページにある本研究に用いたコードは、現在他の測定に使用されている ROC の SlowControlFPGA に設計されている内容が記述されているコードと異なることが分かった。今後の課題としては、現在使用されている正常に測定が行える SlowControl FPGA コードが必要である。なぜ本研究のコードではキャリブレーションテストが出来ないのか、FEM の異常の原因を含めて考える必要もある。

謝辞

本研究を進めるにあたって、多くの方々にご指導ご鞭撻頂いた蜂谷先生に心より感謝申し上げます。蜂谷先生には、ハードウェアの取り扱い、検出器や FPGA に関する様々なこと、物理学の知識など本当に多くのことをお教えていただき、助けていただきました。初心者で知識が足りない私に初歩的なことから分かりやすく丁寧に教えてくださり面倒を見てくださったことで、こうしてここまで取り組むことができました。
また本研究へアドバイスを頂いた理化学研究所の秋葉さん、中川さん、糠塚さんにも心より感謝申し上げます。
高エネルギー物理学研究室において、物理学の基本的なことや様々なことをご指導くださいました、宮林先生、下村先生及び研究室の先輩方にも大変お世話になりました。心より感謝申し上げます。
そして、1 年間共に過ごした同回生の水上さん、辻端さん、藤原さん、岡田さん、佐々木さんへ感謝申し上げます。皆様と励ましあいながら頑張ってきたおかげでここまでめげずに卒業研究を進めることができました。
本研究を進めるにあたり、支えてくださった全ての方に御礼申し上げます。

参考文献

[1] 並本ゆみか 2021 「放射線源を用いた sPHENIX 実験-中間飛跡検出器 INTT 用シリコンセンサーの性能評価」, 卒業論文, 奈良女子大学.

[2] 西森早紀子 2021 「sPHENIX 実験における中間飛跡検出器 INTT の宇宙線を用いた検出効率の研究」, 卒業論文, 奈良女子大学.

[3] 杉山由佳　2022 「RHIC-sPHENIX 実験における中間飛跡検出器 INTT 用シリコンセンサーでのエネルギー損失測定の評価」, 卒業論文, 奈良女子大学.

[4] 森田美羽 2022「RHIC-sPHENIX 実験のための INTT 検出器のデータ読み出し性能の評価」, 修士論文, 奈良女子大学.

[5] 大強度用紙加速器施設　J-PARC　素粒子・原子核研究 http://www.j-parc.jp/c/facilities/nuclear-and-particle-physics/index.html

[6] Brookhaven National Laboratory Newsroom Start-up of 22nd Run at the Relativistic Heavy Ion Collider (RHIC) https://www.bnl.gov/newsroom/news.php?a=119262

[7] Macnica FPGA はじめてガイド　ほんとのほんとの導入編 FPGA 開発の流れ https://www.macnica.co.jp/business/semiconductor/articles/intel/110145/

[8] Macnica Microchip 社開発ツールの種類とサポート・デバイス対応 https://www.macnica.co.jp/business/semiconductor/articles/intel/110145/

[9] Macnica Microchip 社 Flash ベース FPGA ポータル Microchip 社プログラミング関連情報 https://www.macnica.co.jp/business/semiconductor/articles/microchip/135131/

[10] 堀桂太郎　2009「図解　VHDL 実習［第 2 版］」, 森北出版.