sPHENIX実験 中間飛跡検出器INTTのためのデータ処理FPGAの改良

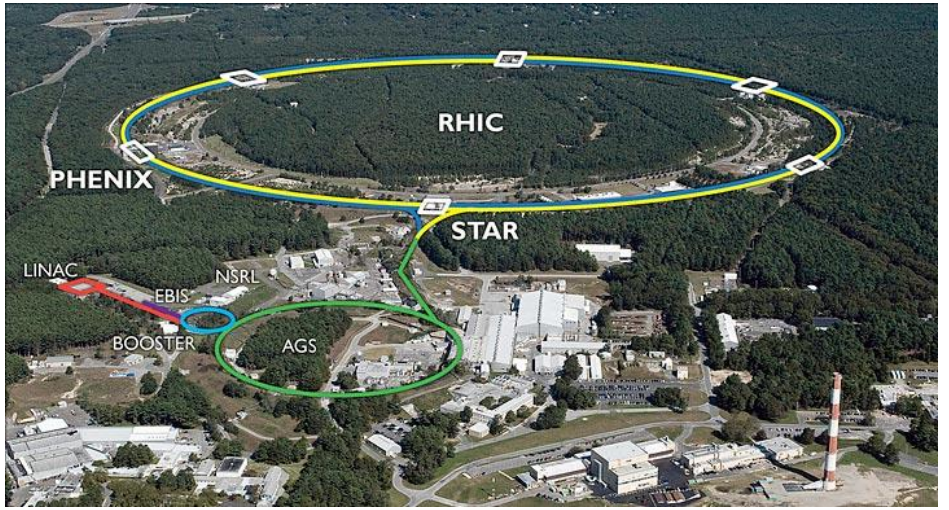
奈良女子大学　理学部数物科学科物理学コース
高エネルギー物理学研究室　4回生
加納麻衣

目次

1. 研究背景
2. 研究目的
3. 開発環境のテスト
4. FPGAコードの改造
5. まとめと今後の課題

研究背景

研究背景



RHIC-sPHENIX実験

アメリカブルックヘブン国立研究所（BNL)

- RHIC（Relativistic Heavy Ion Collider）加速器での実験
- 2000年-2016年まで稼働していたPHENIX実験を高度化、2023年より稼働予定
- 衝突により発生するJet現象やUpsilon粒子を測定し、QGPの性質を決定することが目的
- 金原子核対（200GeV）、陽子対（510GeV)の衝突

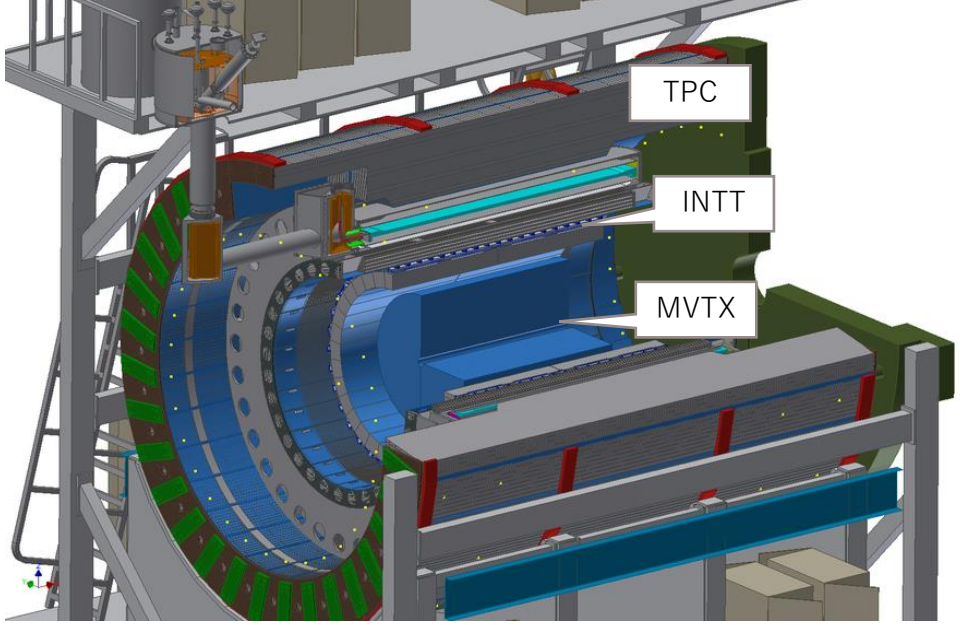
クォーク・グルーオン・プラズマ（QGP)とは

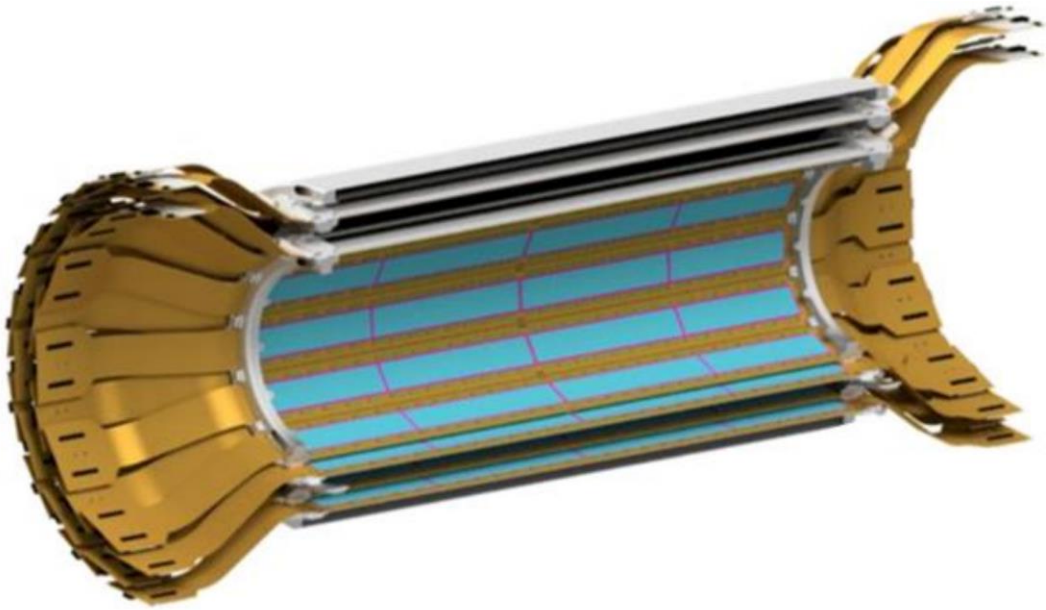
高温高密度状態で、ハドロン内に閉じ込められていたクォークやグルーオンが解放されたプラズマ状態の物質

ビックバンから数10μ秒後に実現していたとされる。

中間飛跡検出器INTT

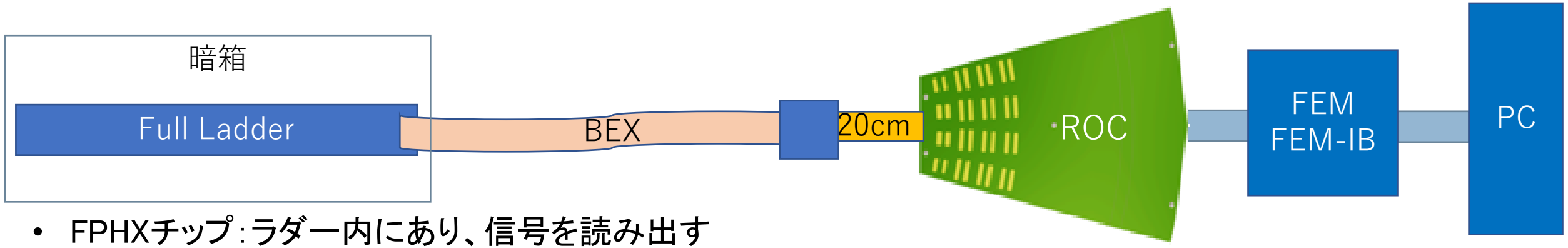
- INTT（INTermadiate Tracker）
- sPHENIX実験で用いられる3つの飛跡検出器のうちの1つ
- ビームパイプから6-12cmに位置する
- バレル状の2層構造のストリップ型シリコン検出器
- 位置分解能と時間分解能が高く、飛跡再構築において重要な役割を果たす





設定値読み出しシステム

INTTには検出器の設定値を読み出すシステムがある（Readbacker）
　設定が正しく行われているか確認するために必要



- FPHXチップ：ラダー内にあり、信号を読み出す
- ROC（Read Out Card）：複数のFPHXチップから送られてきたデータを次に転送する読み出し基板
- FEM（Front End Module）：ROCから送られたデータをまとめ転送する
- FEM-IB：FEM全体を制御する役割
- Bus-Extender（BEX）：シリコンセンサーとROC 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ある

研究目的

課題：Readbackerの問題点

- FPHXとROCの間でBEXを繋げない場合、設定値と読み出し値一致する。
  →正常

- BEXを接続し読み出し線路が長くなると、設定値と読み出し値一致しない。
  →正常でない

- ここでの設定値とはDAC値などがある
  DAC値：　FPHXにおいてエネルギー損失の波形整形を行って
  3bitのADCの8つの閾値をそれぞれ8bitのDAC値で設定可能

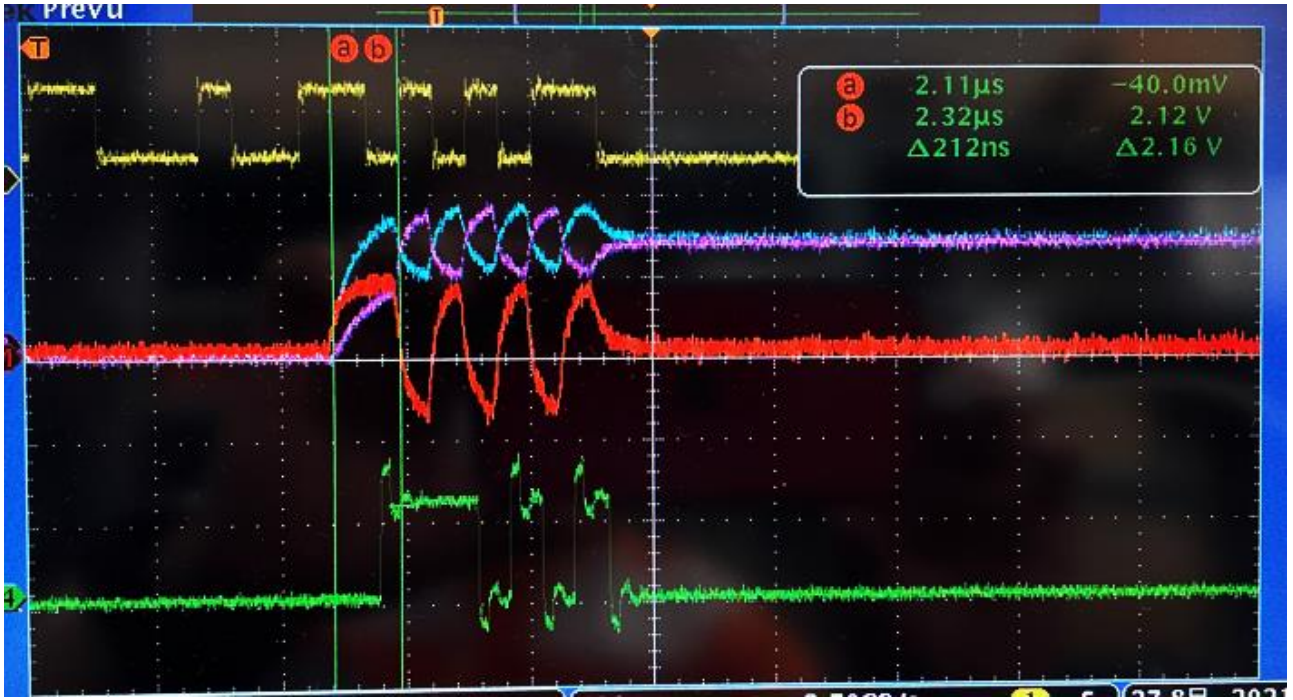
BEX繋げた時のPCの様子

	設定値	読み出し値
DAC0	8	16
DAC1	16	32
DAC2	30	60
DAC3	35	71
DAC4	40	80
DAC5	45	91
DAC6	50	100
DAC7	55	111

一致しない

先行研究

読み出し値をオシロスコープで測定



ROC前後で読み出し値が変わっている
問題箇所は**ROCと特定**

先行研究より

- 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う。

設定値を変えて測定したデータ

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

ROC前の1bit目と、ROC後の2bit目がいつも同じなことがわかった。

1bit分データが遅れて届いている

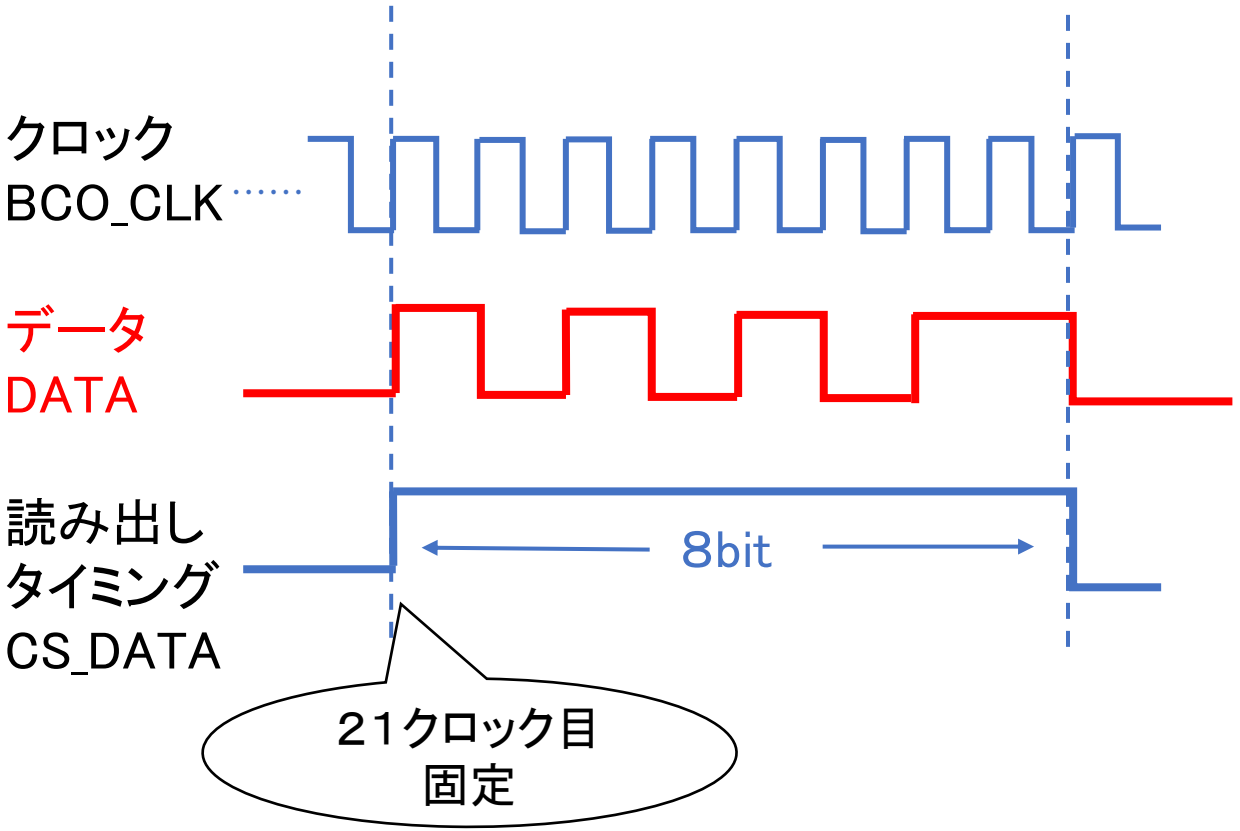
研究目的

設定値読み出しシステムにおいてBus-Extenderを使用した際
Readbackerが正常に動作しない問題の解決

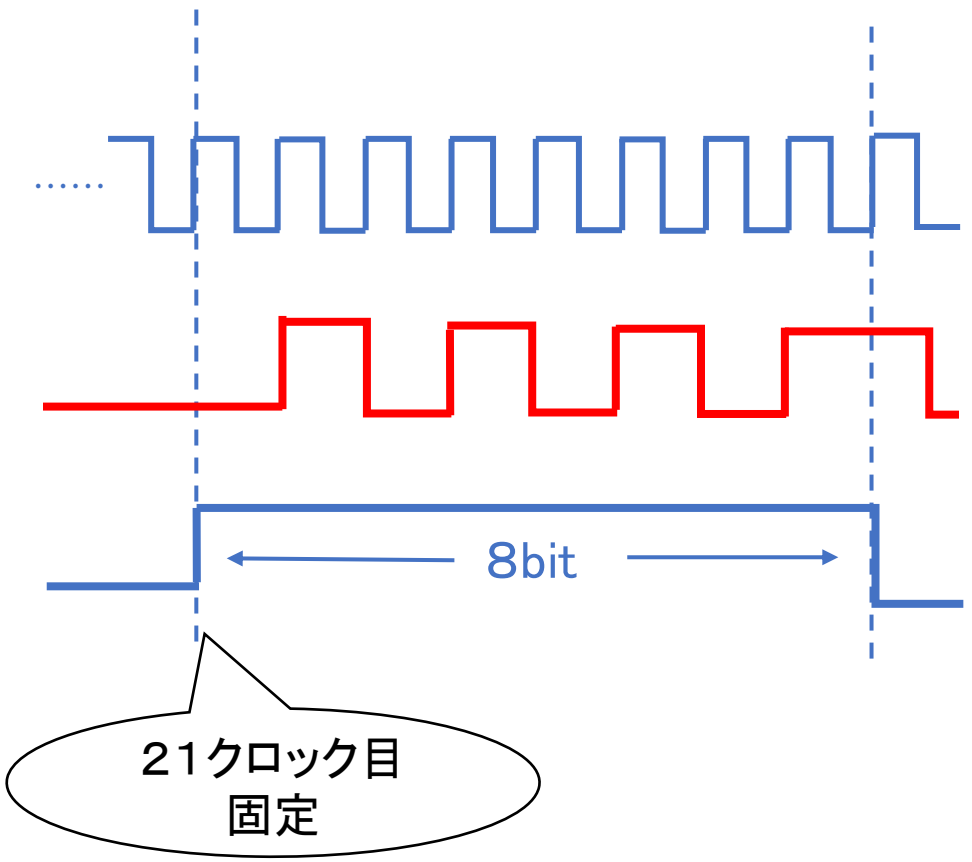
データズレが起きる原因

読み出し値10101011

データが遅れていないとき



データが遅れてくるとき



クロック（BCO_CLK）と読み出しタイミング（CS_CLK)は同期している

→タイミングは固定されている

データが遅れると、データの8bitと読み出しタイミングの8bitがずれる

解決方法

SlowControl　FPGAを改造

・BEXが無いときは、信号の遅れがないのでそのままで良い。

・BEXがあるときは、信号が1bit分遅れてしまうので読み出す
タイミングを1bit分ずらさなければならない。

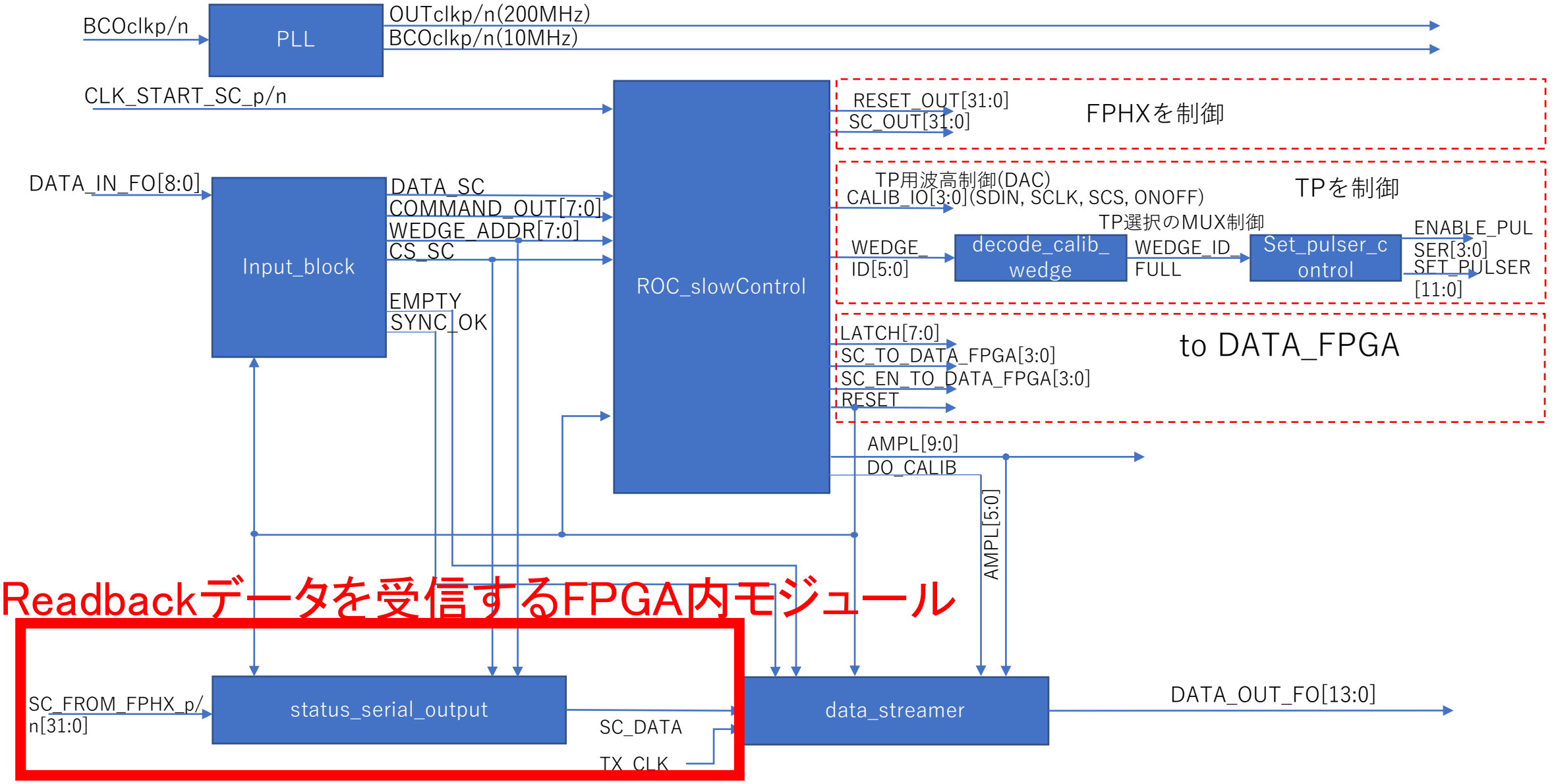
BEXがあってもなくてもデータ8bitを正しく読み出すために
読み出しを8bit→9bitに変更する

ROC-FPGA

- FPGA(field programmable gate array):
  現場でプログラム可能な論理回路配列
  製造後に購入者や設計者が構成を設定できる。
  ハードウェア記述言語(HDL)を用いる。

- 本研究では、ROC内でReadbackerに関係するSlowControl FPGAの改良を行う。
  言語はHDLの1つであるVHDLを使う。

SlowControl FPGAのコードについて



Status_serial_outputの内容

```
    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;
```

読み出し値の受信タイミングについての記述箇所

研究の流れ

FPGA開発ソフトウェアのテスト

改造に用いるコードの内容確認

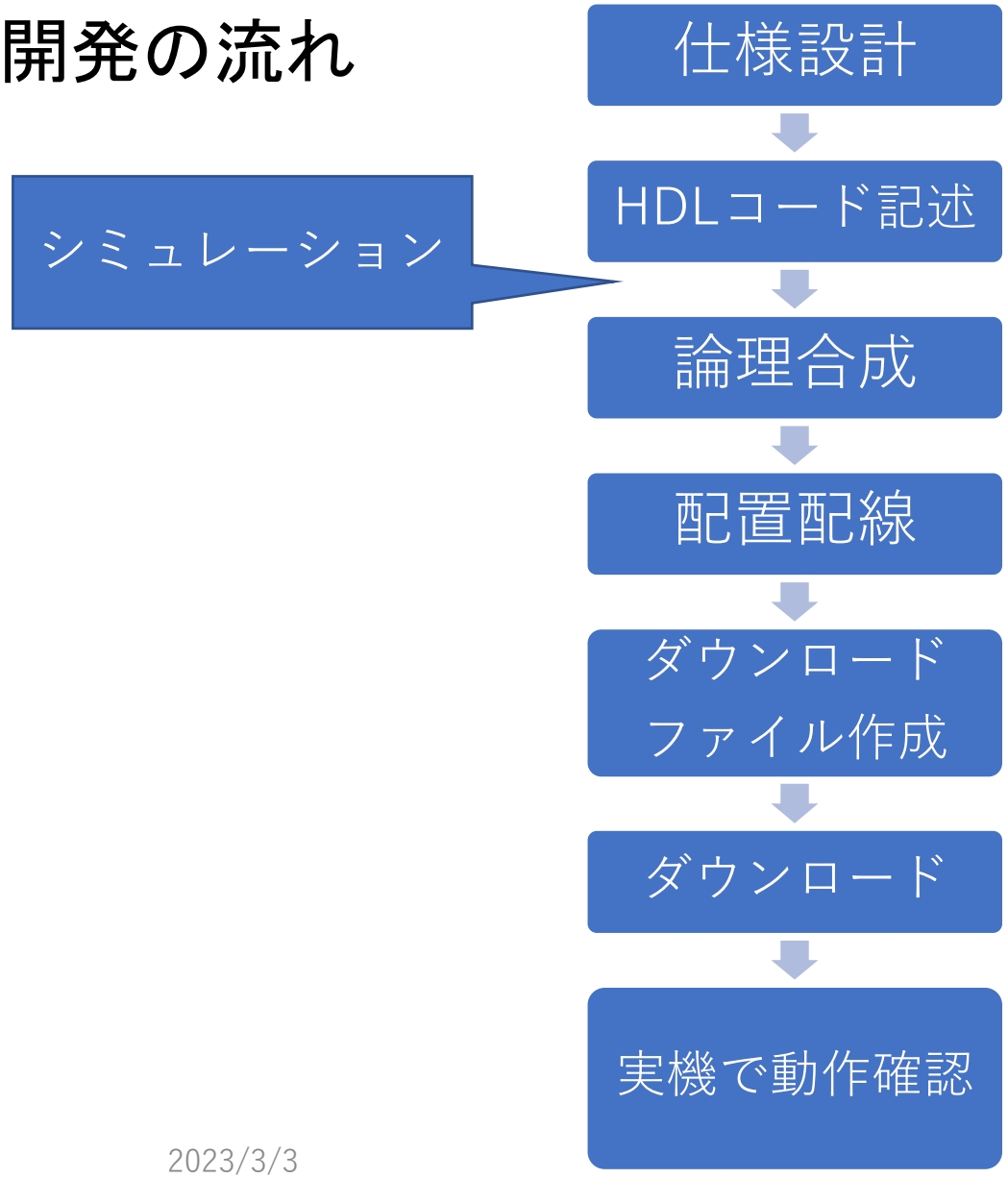
FPGA書き換えのテスト

FPGAコードの改造

改造したコードの動作確認

FPGA開発ソフトウェアについて

開発の流れ



SlowControl FPGAに対応するものは

Libero SoC　v11.9

（一部作業には別のツールも併用）

コード改造前に使用できるかテストする

開発環境のテスト

FPGA開発ソフトウェアのテスト

改造に用いるコードの内容確認

FPGA書き換えのテスト

FPGAコードの改造

改造したコードの動作確認

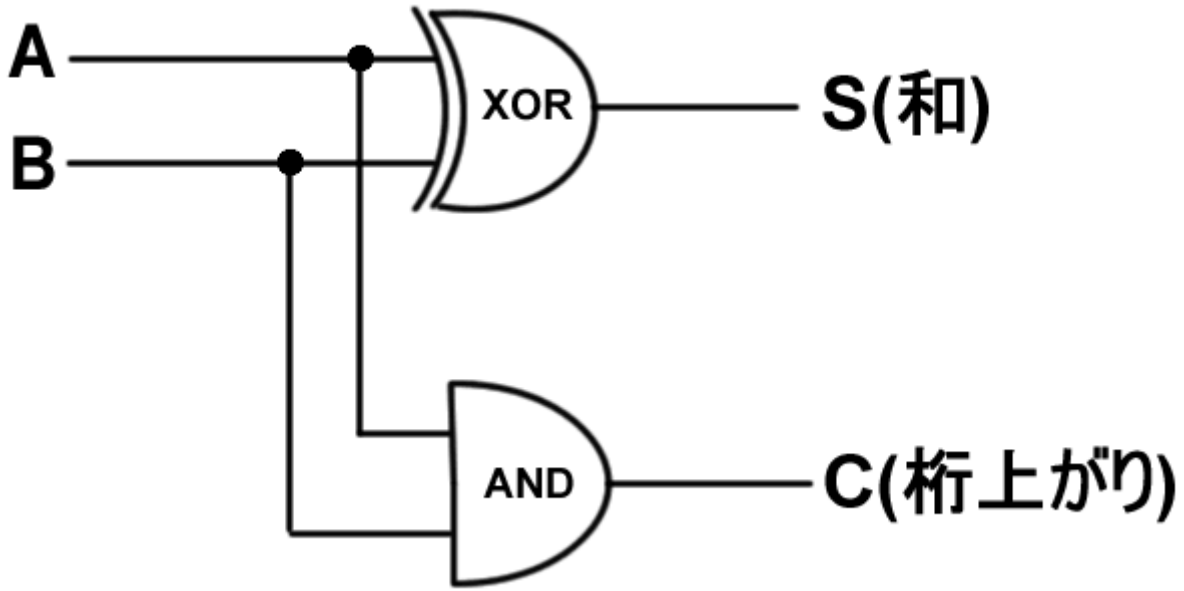
HDLコード記述で回路設計

テストで、回路を一つVHDLコードで設計を行った

右の回路を設計したVHDLコード

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

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;
```

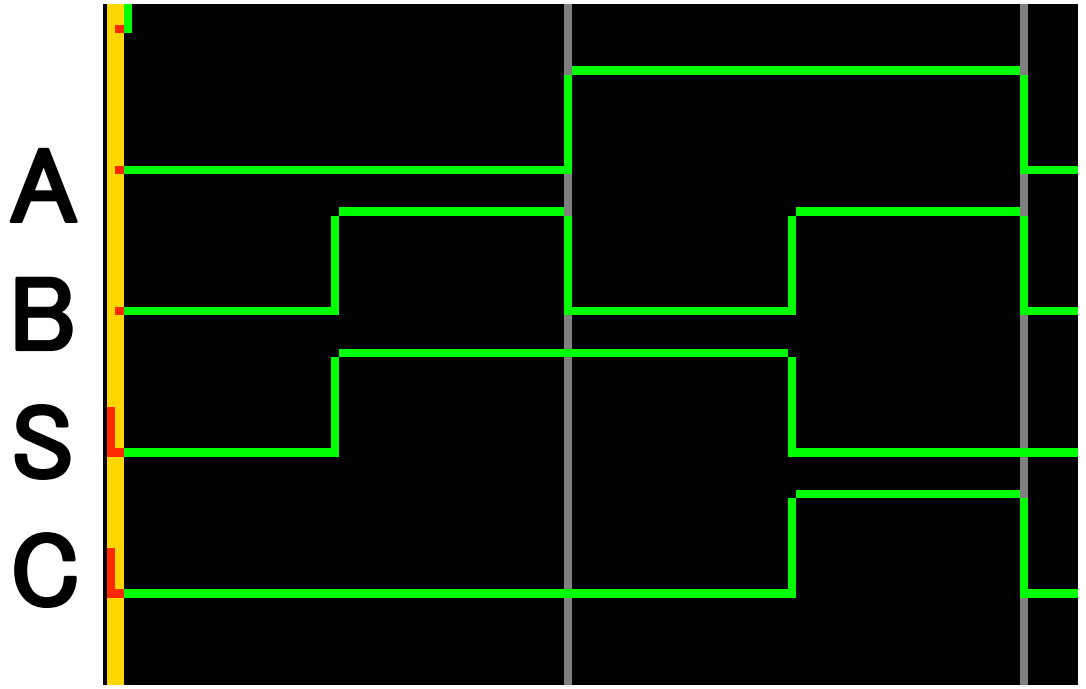
設計した回路



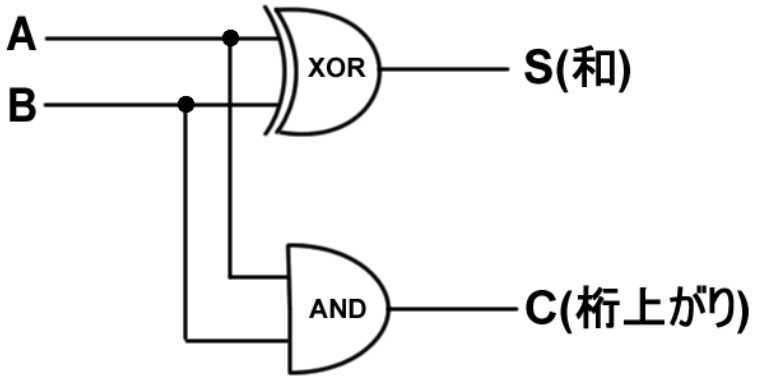
シミュレーションのテスト

設計した回路が正しくできているかのシミュレーションを行う。

任意の入力信号を設定し、動作を表示する。



設計した回路図



設計した回路の真理値表

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

開発ソフトウェアにおいて
VHDLコードで回路を設計し正しくシミュレーションできた

開発環境のテスト

FPGA開発ソフトウェアのテスト

改造に用いるコードの内容確認

FPGA書き換えのテスト

FPGAコードの改造

改造したコードの動作確認

FPGAコードについて

- 以前ROCが用いられていた検出器に関するWebページ上の
  SlowControl FPGAコードを使用。

- 最新のFPGAコード（12-Sep-12,fast ファイル名ROC_slow_control.zip）で
  HDLコードの記述の確認～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

開発環境のテスト

FPGA開発ソフトウェアのテスト

改造に用いるコードの内容確認

FPGA書き換えのテスト

FPGAコードの改造

改造したコードの動作確認

FPGA書き込みツール

- Flash Pro ：FPGAと接続してファイルを書き込む（プログラミング）作業を行うツール
- Flash 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

Verify

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

書き込みを行う前に、他の測定で使用中のROC-FPGAと
　Webページ上のFPGAコードをそれぞれVerifyを行った。 →全てエラー

	Program	Port	Programmer	Program
	FlashPro5	usbS2001	RUN FAILED	☑

Error: programmer 'S2001L82UX' : Executing action VERIFY FAILED, EXIT 11, refer to FlashPro online help for details.

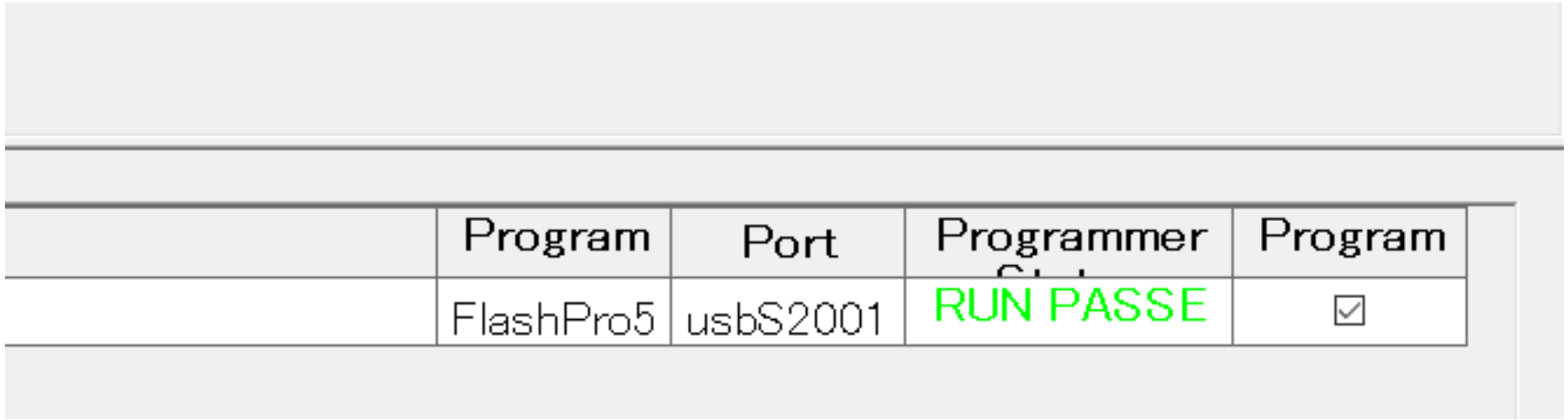
エラーより、Webページ載っているコード全て
現在使われているROC-FPGAの内容と一致していないことが分かった。

FPGAの書き換えテスト

- 先ほどのダウンロードファイルの作成まで成功したコードのProgram成功。

- 次にそのコードでのVerifyも成功。

→FPGAの書き換えすることが出来た。



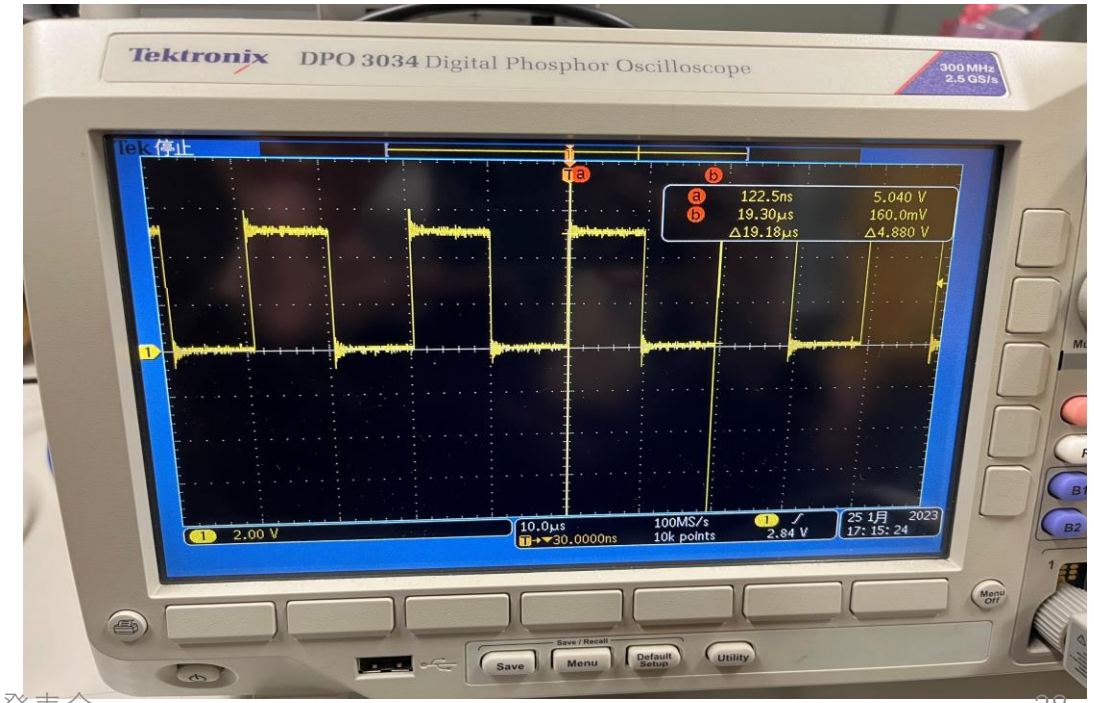
FPGAの書き換えテスト

改造したコードで書き換えて反映できるか、オシロスコープで確認した。

- テストでコードを変更しFPGAに書き込んだ。
  →特定のピンにH/Lを出力することが出来た。

変更した内容

```
 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);
```

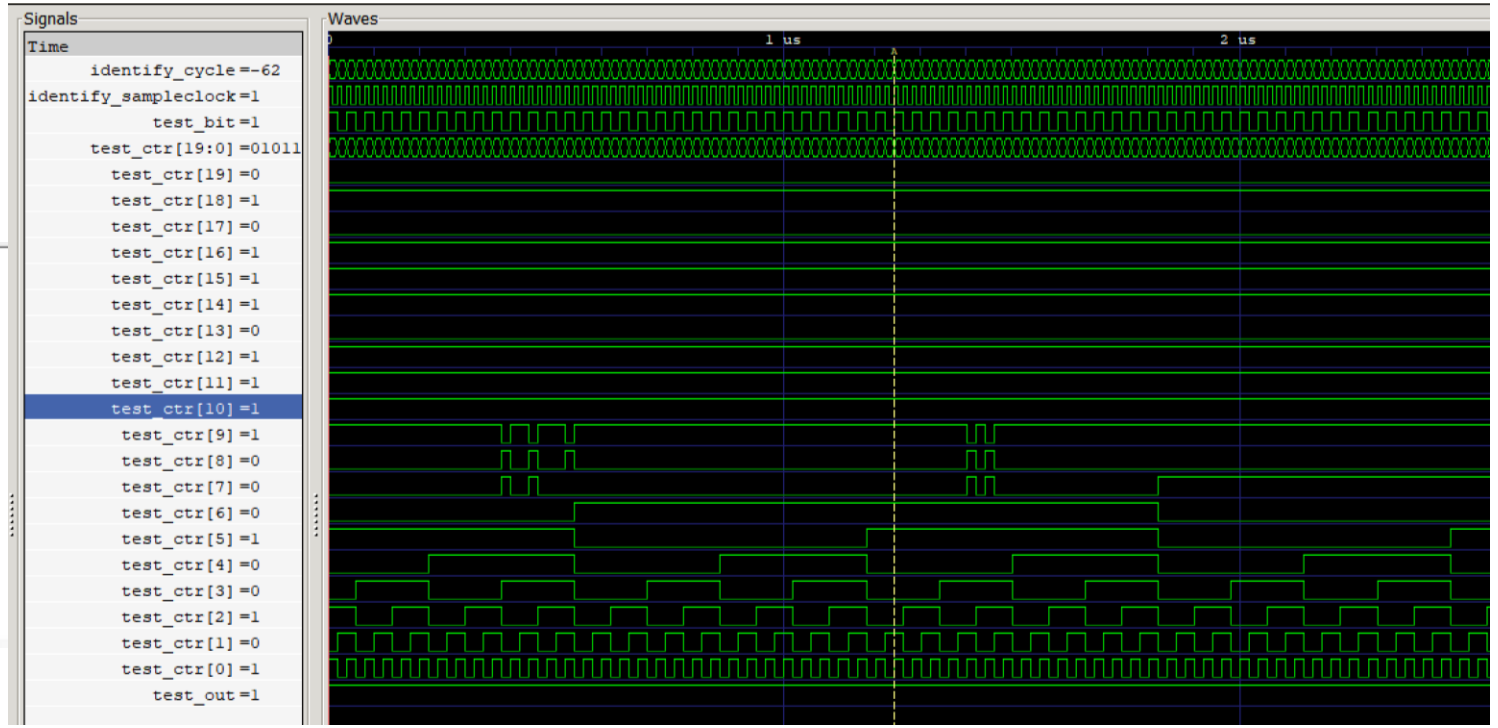


FPGAの書き換えテスト

デジタル回路上の多数信号を表示するツール（Identify）で
書き込んだFPGA内でどのように動作しているかの確認も行った。

→変更したコードの通りに動作した。

```
585     -- test
586     process (CLK_TX)
587     begin
588
589
590       if rising_edge(CLK_TX) then
591         TEST_CTR01011101111001100011 <= TEST_CTR01011101111001100011 + "00000000000000000001";
592         TEST_BIT'1' <= not TEST_BIT'1';
593       end if;
594     end process;
595     TEST_OUT'1' <= TEST_CTR01011101111001100011(10);
596
597
598
599
600 end RTL;
```



開発ツールの使用テスト完了した
→実際にFPGAコードの改造を行う

FPGAコード改造

FPGA開発ソフトウェアのテスト

改造に用いるコードの内容確認

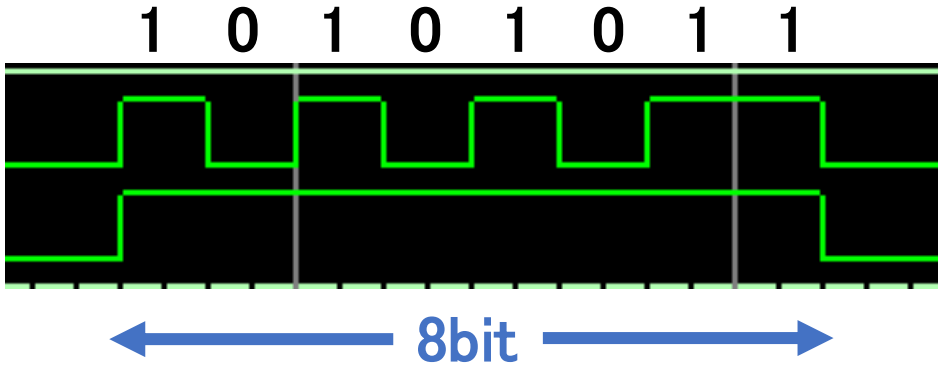
FPGA書き換えのテスト

FPGAコードの改造

改造したコードの動作確認

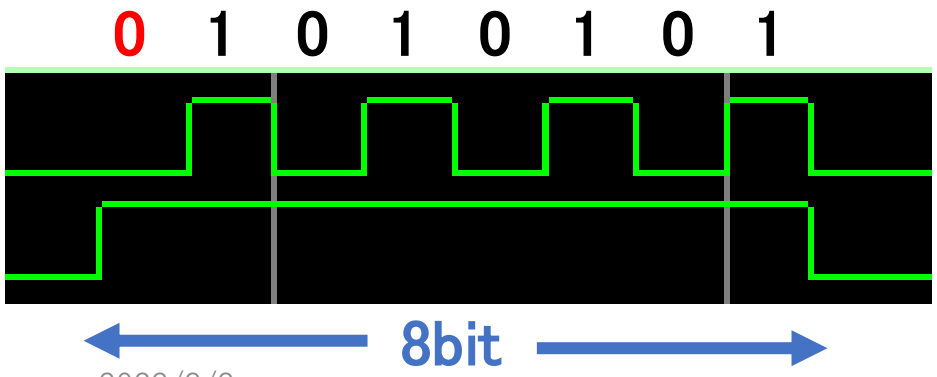
FPGAコードの改造

改造前のシミュレーション
（読み出しが正常にできる状態に設定）



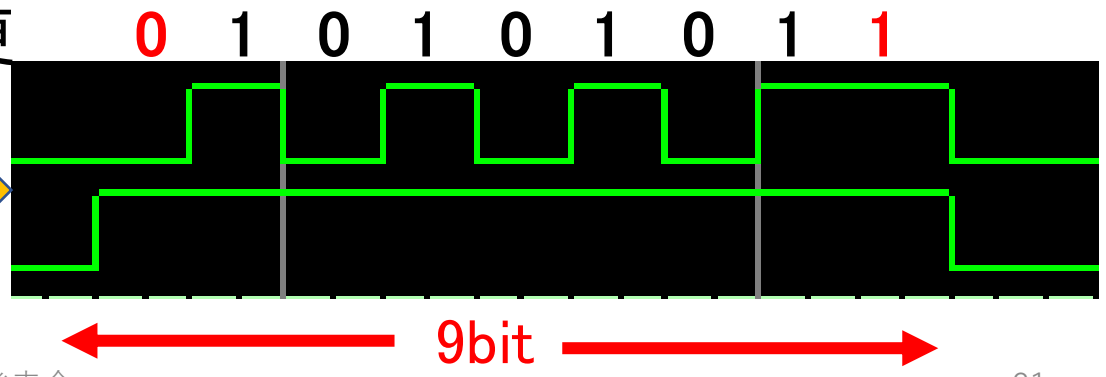
設定値　11010101
→正しい読み出し値　10101011

改造前のシミュレーション
（1bit信号が遅れてくる状態に設定）



9bitに変更

改造後のシミュレーション



変更したコード

```
    --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;
```

改造後、信号が1bit遅れても正常に読み出せることをシミュレーションで確認できた

FPGAコード改造

FPGA開発ソフトウェアのテスト

改造に用いるコードの内容確認

FPGA書き換えのテスト

FPGAコードの改造

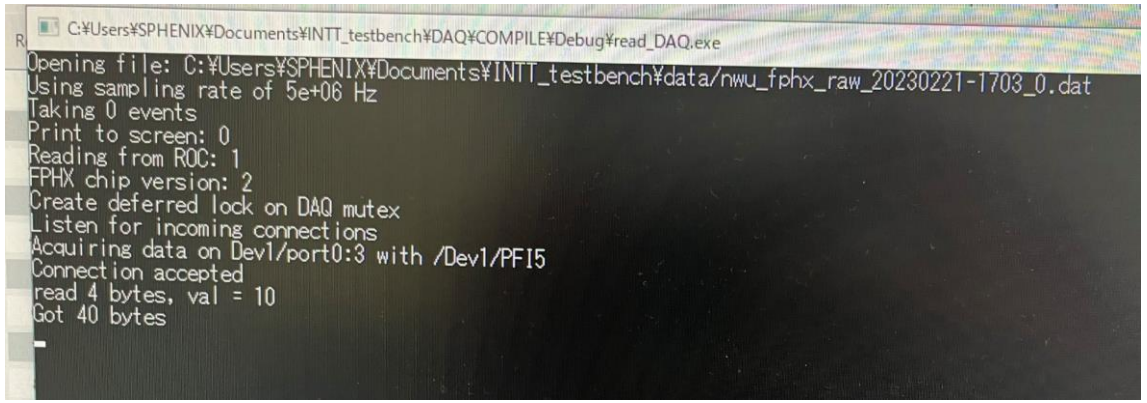
改造したコードの動作確認

改造したコードの動作確認

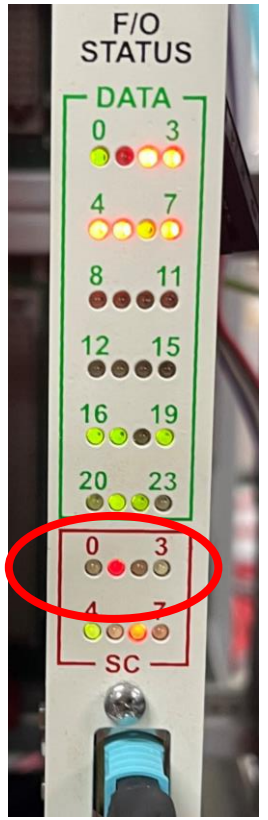
- 改造したコードをROC7のFPGAに書き込み、電源とラダーに繋ぎ全体の動作確認を行うため、キャリブレーションテストを行った。

→キャリブレーションテストが出来なくなった

↓キャリブレーションデータが来ないPCの様子



↓キャリブが正常に動いている時のFEM



FEMに本来ついていない箇所でランプがついている↑

改造していないコードの動作確認

①今まで使っていたコードの改造前のもの

②FVTXのWebページにある別のFPGAコード　（未改造）

- 12-Sep-12,slow version
  ファイル名ROC_slow_control_slow_5Feb13.zip
- 5-Sep-12,fast
  ファイル名ROC_slow_control_NoReadClkRelay_94Mhz.zip

上記をFPGAに書き込み、キャリブレーションテストを行った。
→いずれもデータが取れなかった。

まとめ

- FPGAの書き換えを行うことが出来るようになった。

- SlowControl FPGAコードで読み出しを8bit →9bitに変更できた。
  動作をシミュレーションでも確認できた。

- 改造したコードを書き込んだROCでは、キャリブレーションテストが出来ない。

今後の課題

- 改造したコードを書き込んでる状態で、FPGAがどのように動作しているのかを調べる。

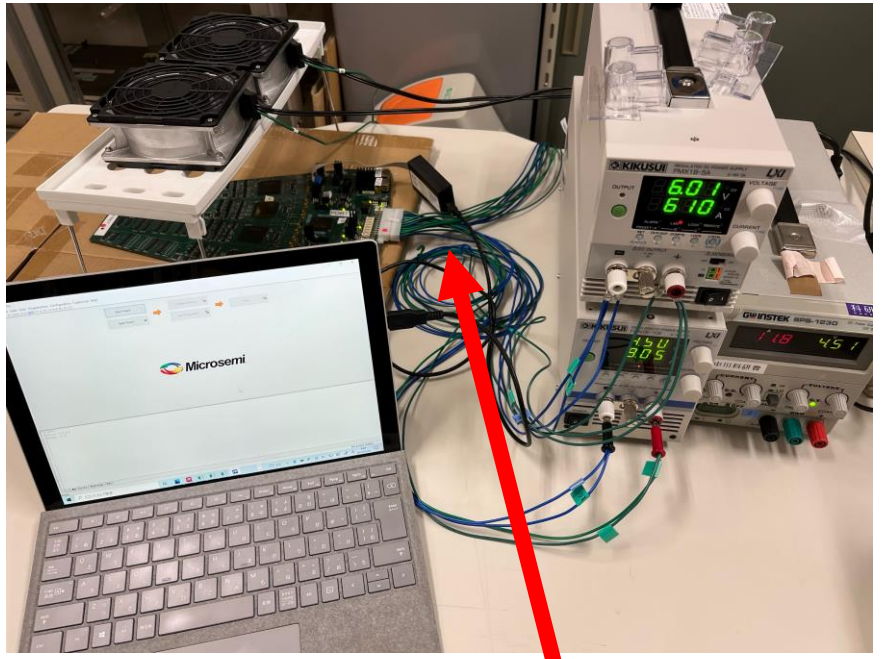
- FEMで正常な場合と異なるランプがついている原因の確認。

Back Up

先行研究のデータまとめ

GUI INPUT	SC_OUT(ICB)	SC_OUT(FEM)	GUI ReadBack	
171 (10101011)	11010101	11101010 (87)	87 (01010111)	
170 (10101010)	01010101	00101010 (84)	84 (01010100)	
172 (10101100)	00110101	00011010 (88)	0	読み出し結果がおかしい
173 (10101101)	10110101	11011010 (91)	91	
174 (10101110)	01110101	00111010 (92)	92	
138 (10001000)	01010001	00101000 (20)	20 (00010100)	
122 (01111010)	01011110	00101111 (244)	244 (11110100)	
20 (00010100)	00101000	00010100 (40)	40 (00101000)	
40 (00101000)	00010100	00001010 (80)	80	
25 (00011001)	10011000	11001100 (51)	51	
30 (00011110)	01111000	00111100 (60)	60	
35 (00100011)	11000100	11100010 (71)	71	

開発環境について



Libero SoC　v11.9（ProAsic3Eで使える）

- VHDL言語を用いる　論理合成＋配線ツール
- V11.9はProAsic3E用の最終版
- FVTX開発にはv9.1が使われている

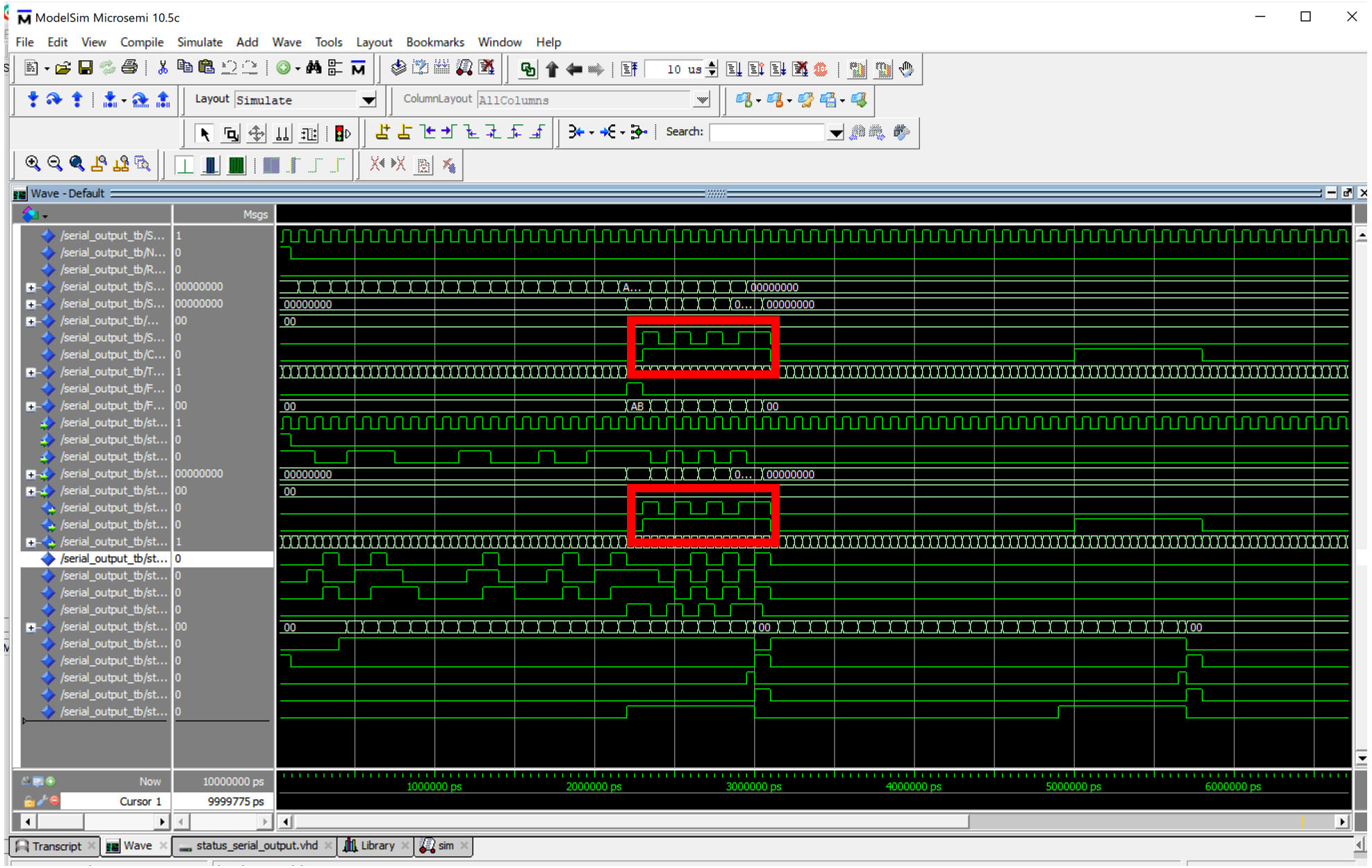
FlashPro（ソフトウェア）

- FPGA書き込み用ソフトウェア
- PCとの接続にFlashPro5を用いる

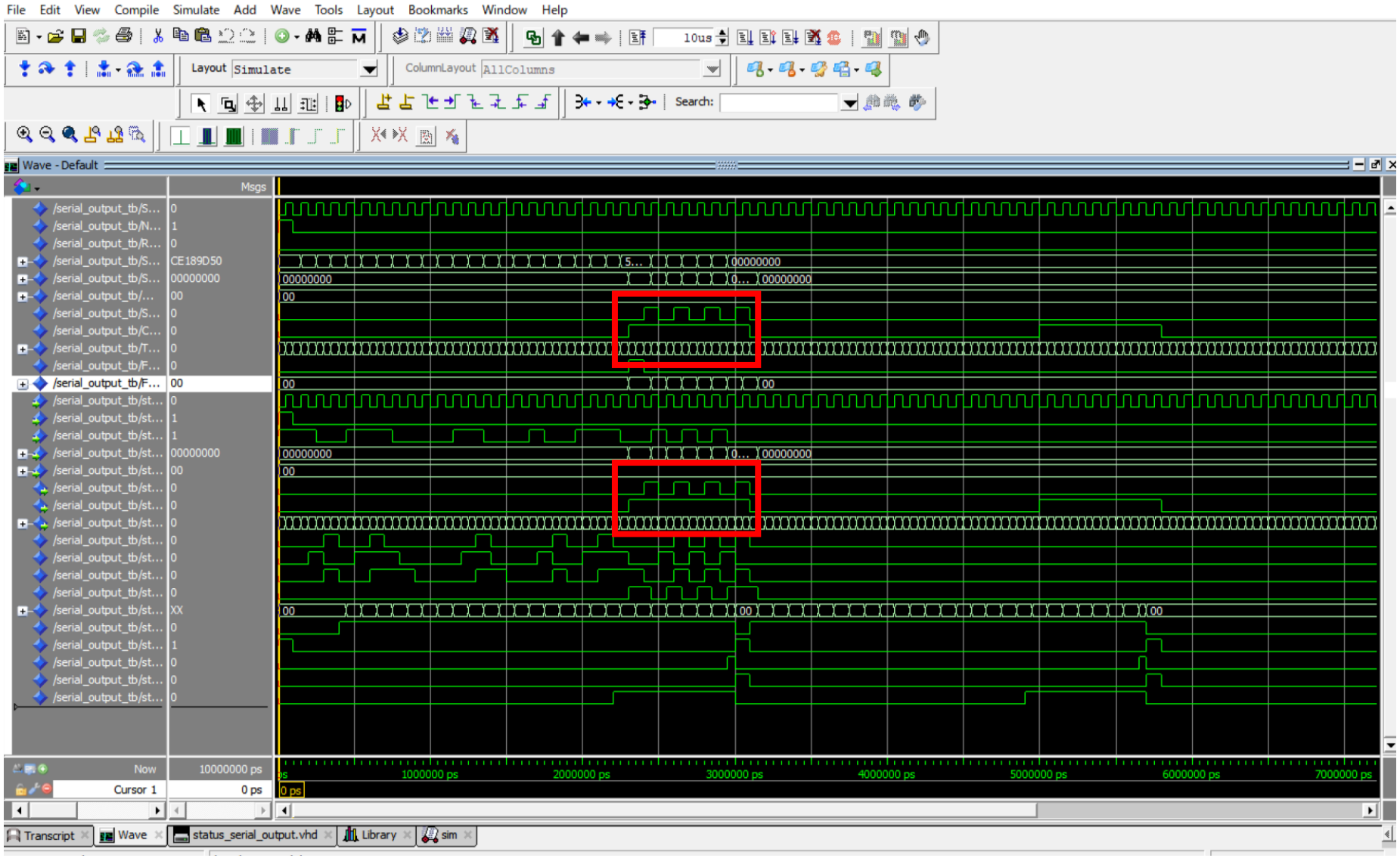
Identify　Synopsys社のハードウェアデバッガ

- ロジックアナライザー（デジタル回路上の多数のデジタル信号を表示するもの）のように動作する
- Identifyで見たい信号などを設定し、波形で見ることができる

改造前　遅れず信号が来た時のシミュレーション

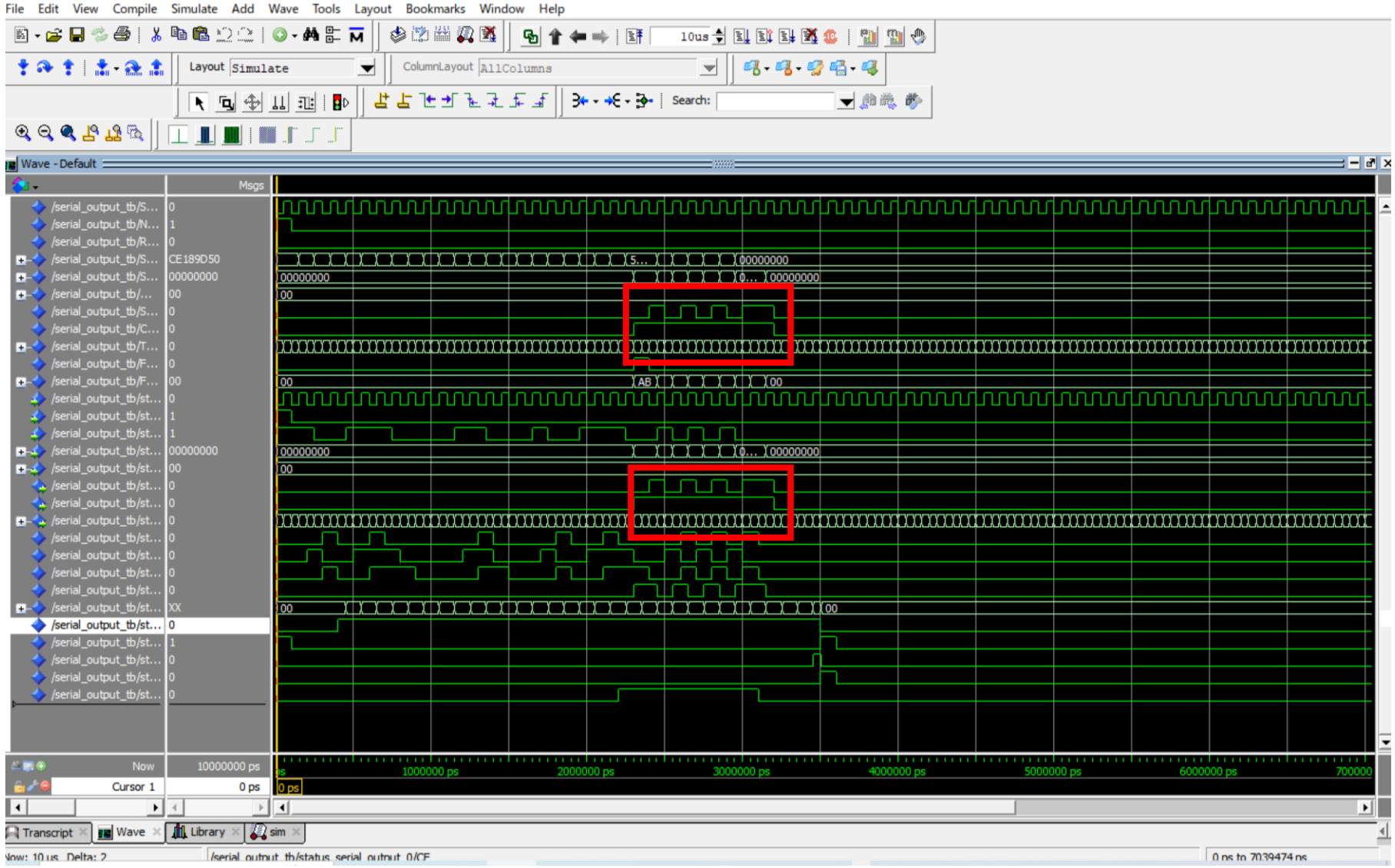


Bus-Extender（BEX）をつないだ時のように読み出し値1bitずれる状態をシミュレーション（読み取り開始クロックを21→22に変更）



読み出し信号が10101011 -> 01010101に変わった。 1つずれた。

読み出し値が1bit遅れる場合に合わせてコードを改造した時のシミュレーション



9bit 読み出せるようにできた。読み出しデータは010101011

シミュレーションの設定（テストベンチ）

```
            NSYSRESET <= '1';
            wait for (SYSCLK_PERIOD*1);

            NSYSRESET <= '0';
--            wait for ( SYSCLK_PERIOD * 21 );
           wait for ( SYSCLK_PERIOD * 22 ); -- 1clock delayed

            FPHX_IN_DATA_EN <= '1';
            wait for ( SYSCLK_PERIOD);

            FPHX_IN_DATA_EN <= '0';
            wait for ( SYSCLK_PERIOD * 100 );
            wait;

        end if;
    end process;

    -- Clock Driver
    SYSCLK <= not SYSCLK after (SYSCLK_PERIOD / 2.0 );


    process (SYSCLK, NSYSRESET, FPHX_IN_DATA_EN)
    begin
        if ( NSYSRESET = '1' ) then
          SEND_CMD_DATAin <= "1100111" & "00001" & "10001" & "001" & "11010101" & "0000";
          FPHX_IN_DATA <= "00000000";

        elsif ( FPHX_IN_DATA_EN = '1' ) then
          FPHX_IN_DATA <= "10101011";

        elsif ( rising_edge(SYSCLK) ) then

          SEND_CMD_DATAin(31 downto 0) <= SEND_CMD_DATAin(30 downto 0) & '0';


          FPHX_IN_DATA(7 downto 0) <= FPHX_IN_DATA(6 downto 0) & '0';
```

SlowControl FPGA

ROC内で読み出しコマンド（SlowControl コマンド）をラダーに送受信する。

- SlowControl の信号の内容(32bit)

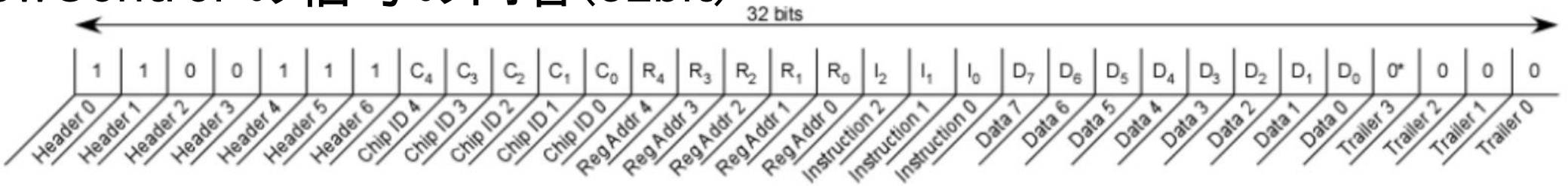


Figure 7 - The Slow Control Word

- Header（7bit): 1100111　　固定
- Chip-ID(5bit): 00001 = chip1　　chip位置
- RegAddr(5bit): 10001 = 17　　何の値か
- Instruction(3bit): 001 = write　　動作
- Data(8bit): 11010101 = 213　　値　　←これを読み出している
- Trailer(4bit) : 0000　　固定

- この時の入力データは　1100111_00001_10001_001_11010101_0000 になる