

2022.05.04

2022 PCFG 보강

**Signal Processing Systems Laboratory
Department of Electronic Engineering
Sogang University, Seoul, Korea**



SOGANG UNIVERSITY



**Signal Processing Systems
Laboratory**

Department of Electronic Engineering, Sogang University

❖ 프로젝트를 위한 Vivado 사용법

- Simulation 돌리는 법
- 에러 발생 시 확인 방법
- RAM 생성 하는 법
- Dat 파일 추가
- Constraints 파일 추가
- Bit 파일 생성하는 법

❖ USB protocol

❖ PC Based Function Generator (PCFG)

- Introduction
- 8254
- Mode 설명
- Test bench
- 시연 및 일정

프로젝트를 위한 Vivado 사용법

❖ 시뮬레이션에서 signal or state 보는 법

Scope x Sources

Name	Design Unit	Block Typ
tb_pcfg	tb_pcfg(behav...	VHDL En.
uut	PCFG_TOP(B...	VHDL En.
clk_gen	TOP_8254(Be...	VHDL En.
Controller	Signal_Contr...	VHDL En.
OPTION	Option_mode...	VHDL En.
PC_LATCH	latch4pc(Beh...	VHDL En.
IN_LATCH	latch(Behavior...	VHDL En.
OUT_LATCH	latch(Behavior...	VHDL En.
DA_LATCH	latch(Behavior...	VHDL En.
AD_LATCH	latch(Behavior...	VHDL En.
ADDR_DECODER	Address_Dec...	VHDL En.
PC_MUX	mux4pc(Beha...	VHDL En.
OUT_MUX	mux4out(Beh...	VHDL En.
TRI_BUF	tristatebuffer(...	VHDL En.

Objects ? _ □ □ X

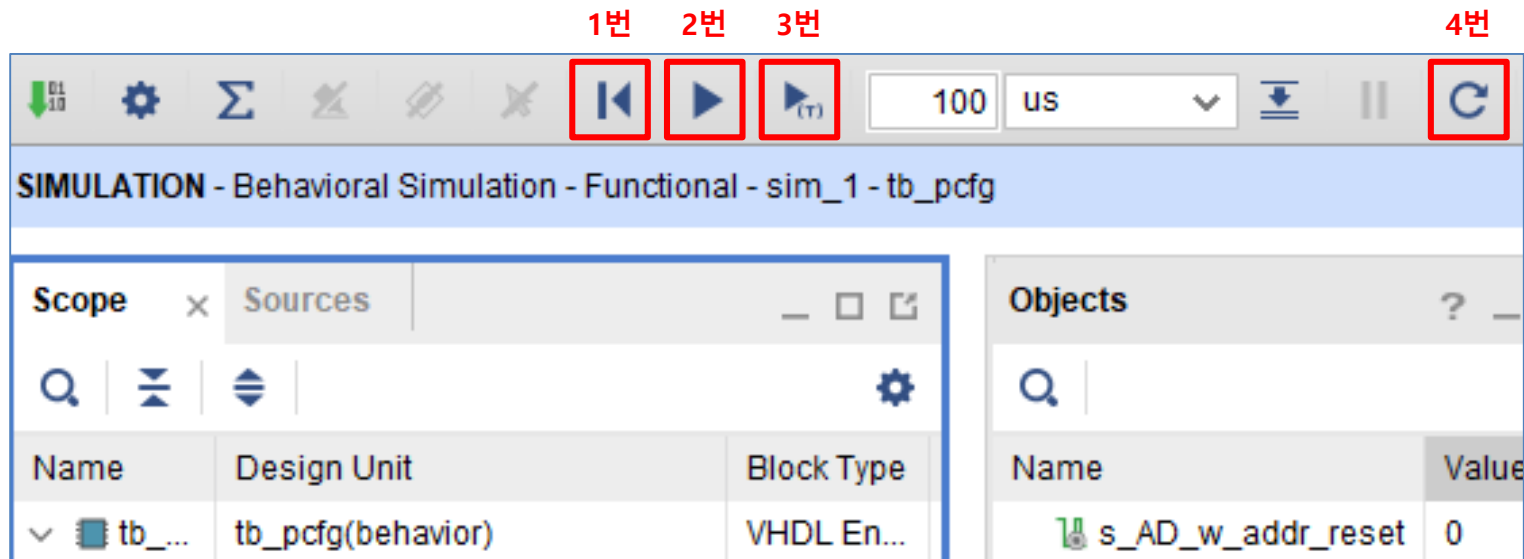
Name
sys_clk
s_reset_b
s_debug_led[6:0]
s_debug_sw[7:0]
s_debug_header[15:2]
s_cmd_data
s_wen
s_ren
s_oe_b
s_address[8:0]
s_inlatch_en
s_inlatch_dout[7:0]
s_outlatch_en
s_outlatch_dout[7:0]

tb_pcfg_behav.wcfg*

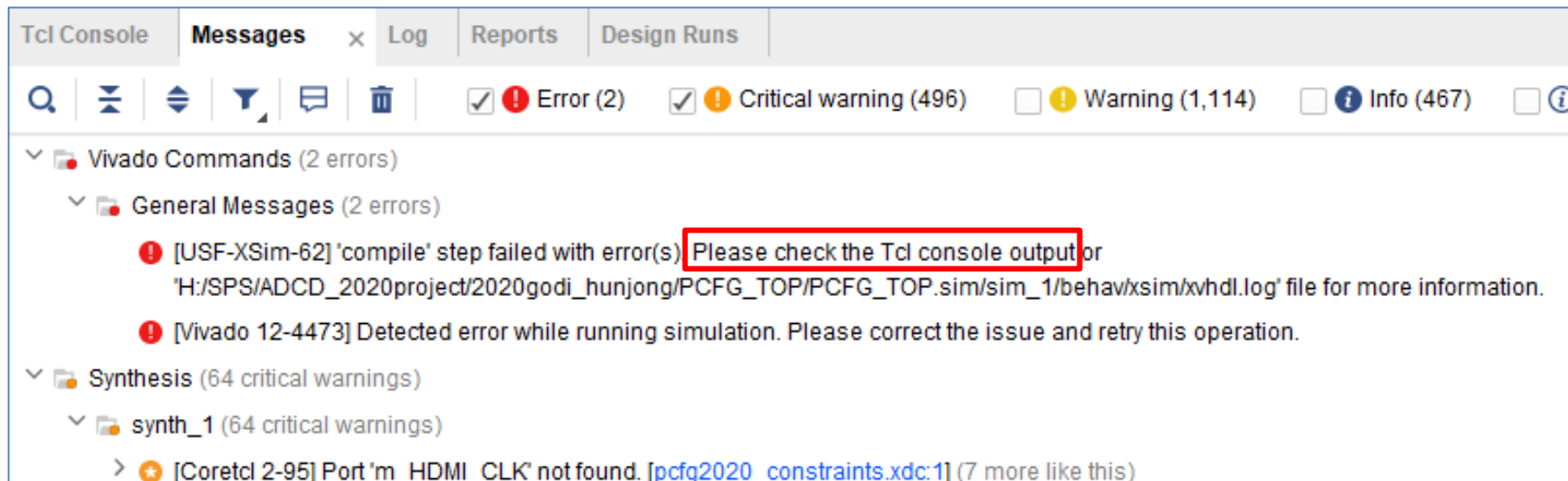
Name
TOP
Controller
OPTION
s_cmd_data
s_wen
s_ren
s_oe_b
s_address[8:0]
s_inlatch_en
s_inlatch_dout[7:0]
s_outlatch_en
s_outlatch_dout[7:0]

❖ 시뮬레이션 시간 설정

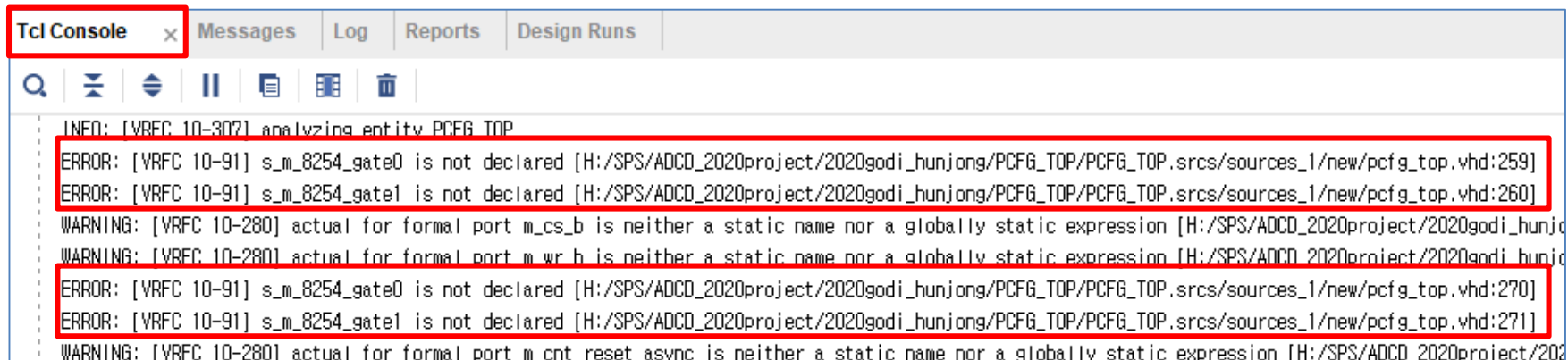
- 1번 : 시뮬레이션을 0초로 reset시킴
- 2번 : 정지 버튼 누를 때 까지 시뮬레이션 진행
- 3번 : 오른쪽의 시간만큼 시뮬레이션 진행
- 4번 : 코드 수정 후 시뮬레이션을 새로 고침 할 때



❖ 아래와 같은 방법으로 해결

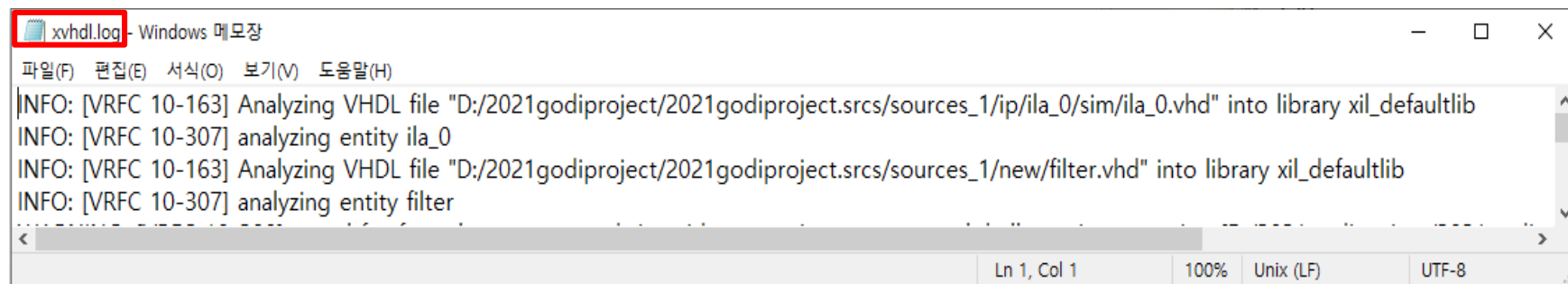
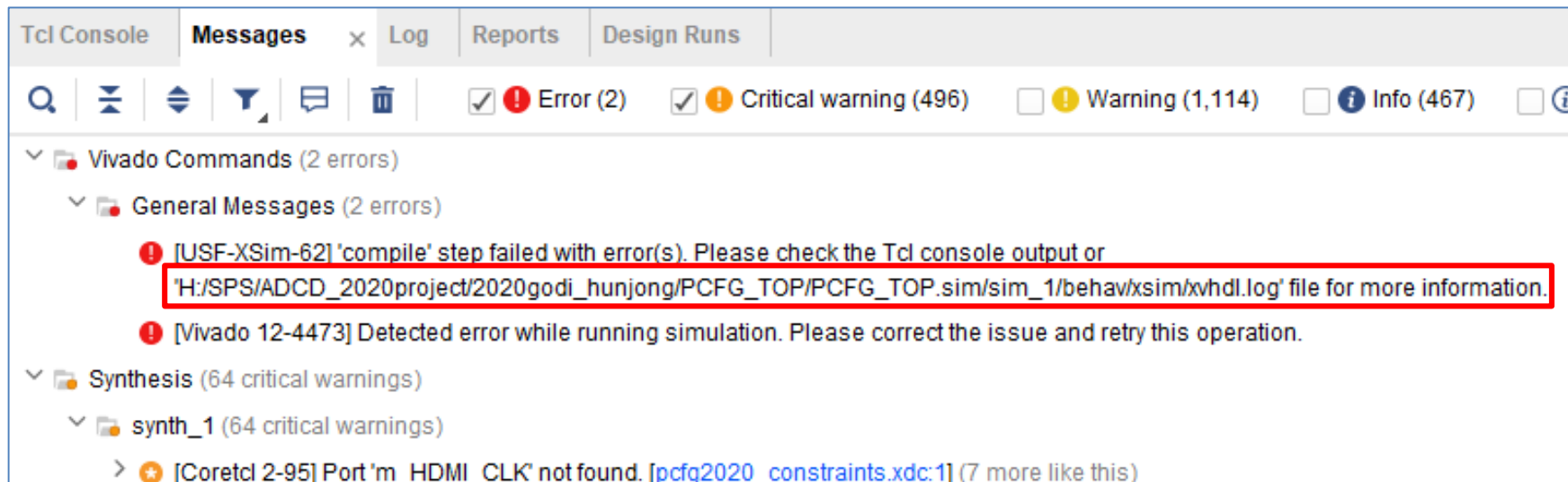


The screenshot shows the 'Messages' window in Vivado. The 'Vivado Commands' section is expanded, showing 'General Messages' with two errors. The first error is highlighted with a red box: '[USF-XSim-62] 'compile' step failed with error(s) Please check the Tcl console output or 'H:/SPS/ADCD_2020project/2020godi_hunjong/PCFG_TOP/PCFG_TOP.sim/sim_1/behav/xsim/xvhdl.log' file for more information.' The second error is '[Vivado 12-4473] Detected error while running simulation. Please correct the issue and retry this operation.'

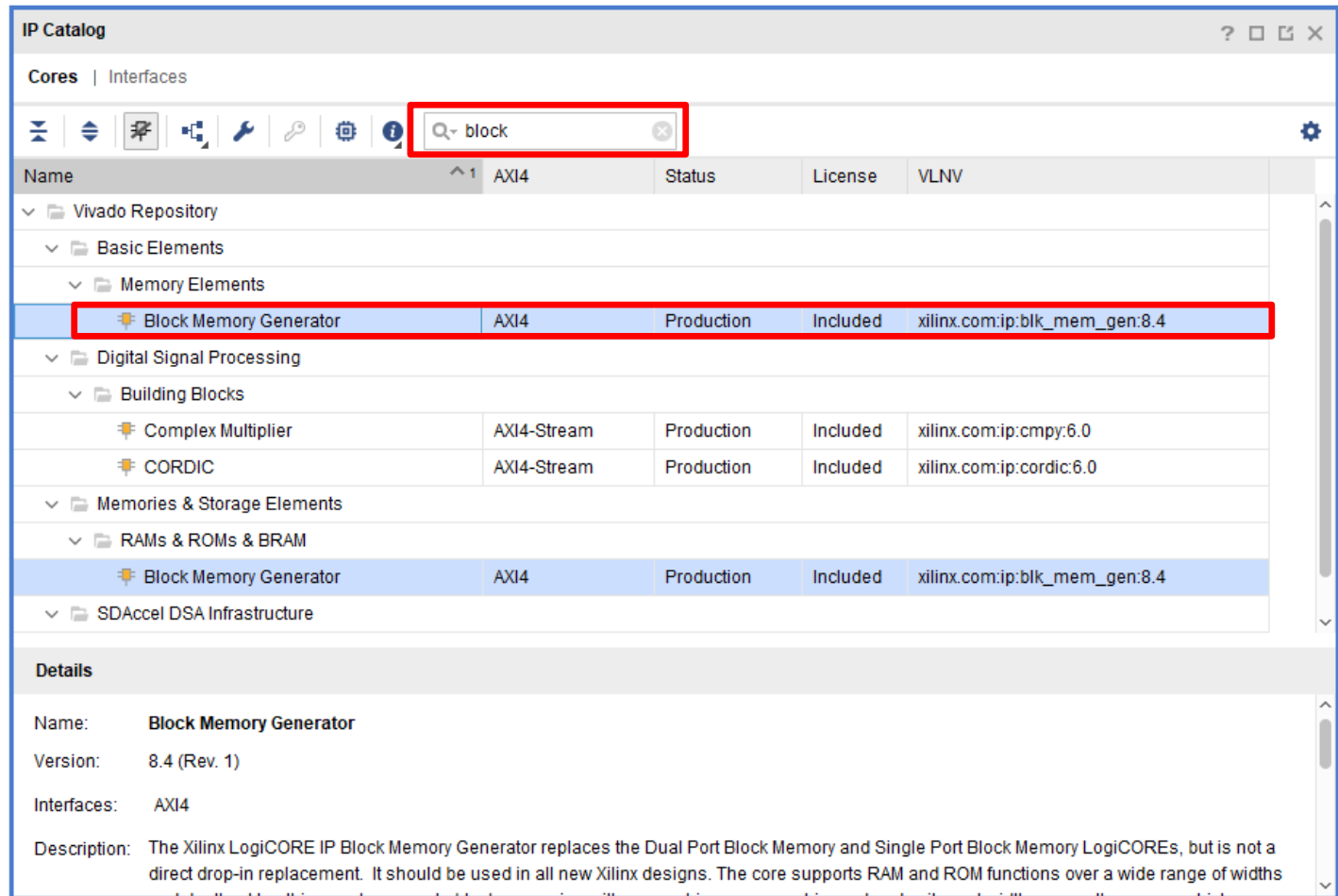


The screenshot shows the 'Tcl Console' window in Vivado. The 'Tcl Console' tab is selected and highlighted with a red box. The console displays several error and warning messages. The first two error messages are highlighted with red boxes: 'ERROR: [VRFC 10-91] s_m_8254_gate0 is not declared [H:/SPS/ADCD_2020project/2020godi_hunjong/PCFG_TOP/PCFG_TOP.srcs/sources_1/new/pcfg_top.vhd:259]' and 'ERROR: [VRFC 10-91] s_m_8254_gate1 is not declared [H:/SPS/ADCD_2020project/2020godi_hunjong/PCFG_TOP/PCFG_TOP.srcs/sources_1/new/pcfg_top.vhd:260]'. The last two error messages are also highlighted with red boxes: 'ERROR: [VRFC 10-91] s_m_8254_gate0 is not declared [H:/SPS/ADCD_2020project/2020godi_hunjong/PCFG_TOP/PCFG_TOP.srcs/sources_1/new/pcfg_top.vhd:270]' and 'ERROR: [VRFC 10-91] s_m_8254_gate1 is not declared [H:/SPS/ADCD_2020project/2020godi_hunjong/PCFG_TOP/PCFG_TOP.srcs/sources_1/new/pcfg_top.vhd:271]'. There are also warning messages about formal ports.

❖ 아래와 같은 방법으로 해결

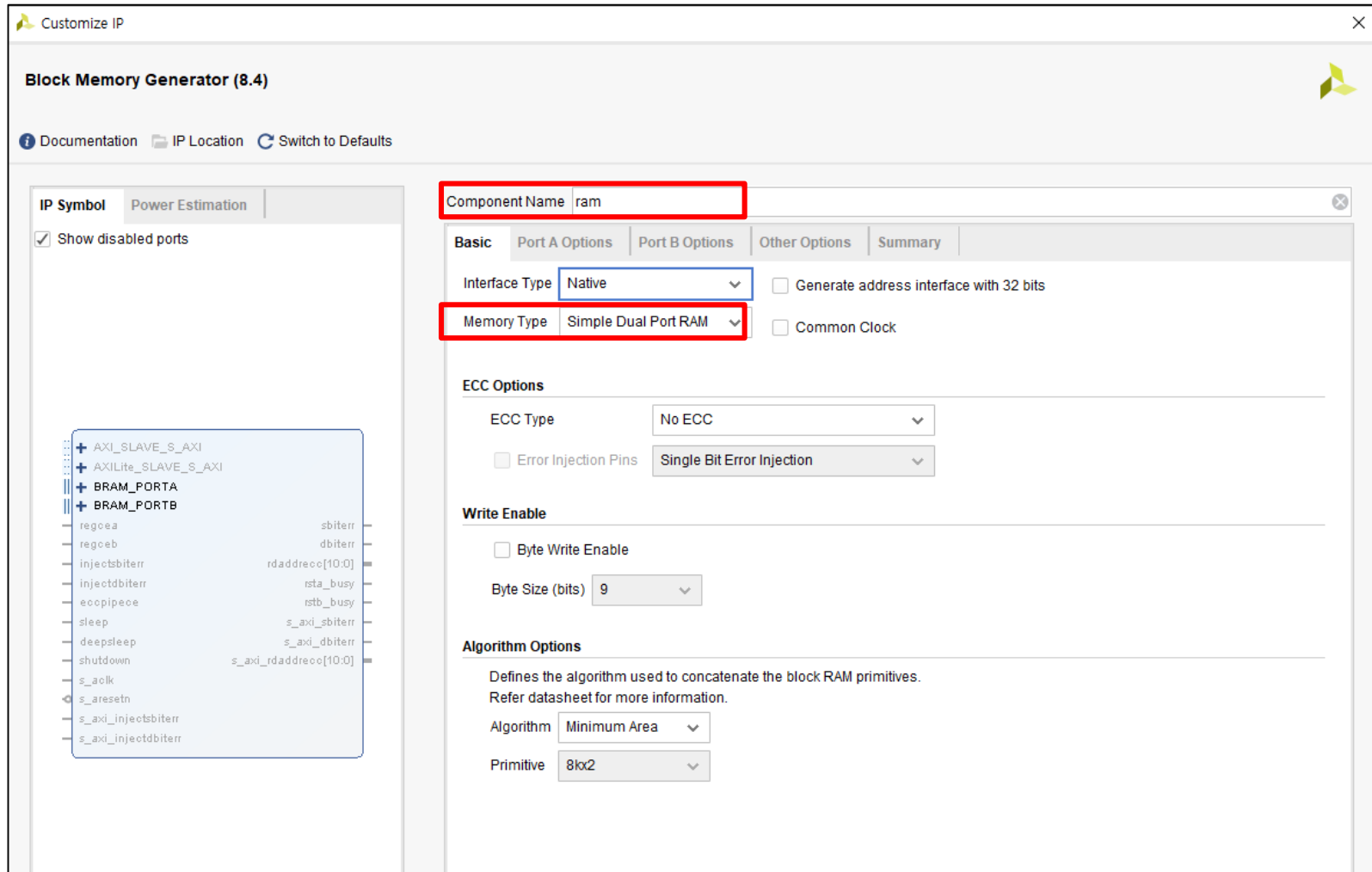


- ❖ 프로젝트 생성 후 'PROJECT MANAGER' -> 'IP Catalog' 클릭
- ❖ 검색창에 'block' 타이핑 후 'Block Memory Generator' 더블 클릭



❖ Component Name 설정(자유롭게) 및 Memory Type 설정

- Memory Type 은 프로젝트에 사용할 'Simple Dual Port RAM' 선택



❖ Memory Size 설정

- 사진과 같이 Port A 설정
 - Width : data의 bit수
 - Depth : 저장할 수 있는 data의 수

Customize IP

Block Memory Generator (8.4)

Documentation IP Location Switch to Defaults

IP Symbol Power Estimation

☒ Show disabled ports

Component Name ram

Port A Options

Memory Size

Port A Width 8 Range: 1 to 4608 (bits)

Port A Depth 2048 Range: 2 to 1048576

The Width and Depth values are used for Write Operations in Port A

Operating Mode No Chan... Enable Port Type Use ENA Pin

Port A Optional Output Registers

☐ Primitives Output Register ☐ Core Output Register

☐ SoftECC Input Register ☐ REGCEA Pin

Port A Output Reset Options

☐ RSTA Pin (set/reset pin) Output Reset Value (Hex) 0

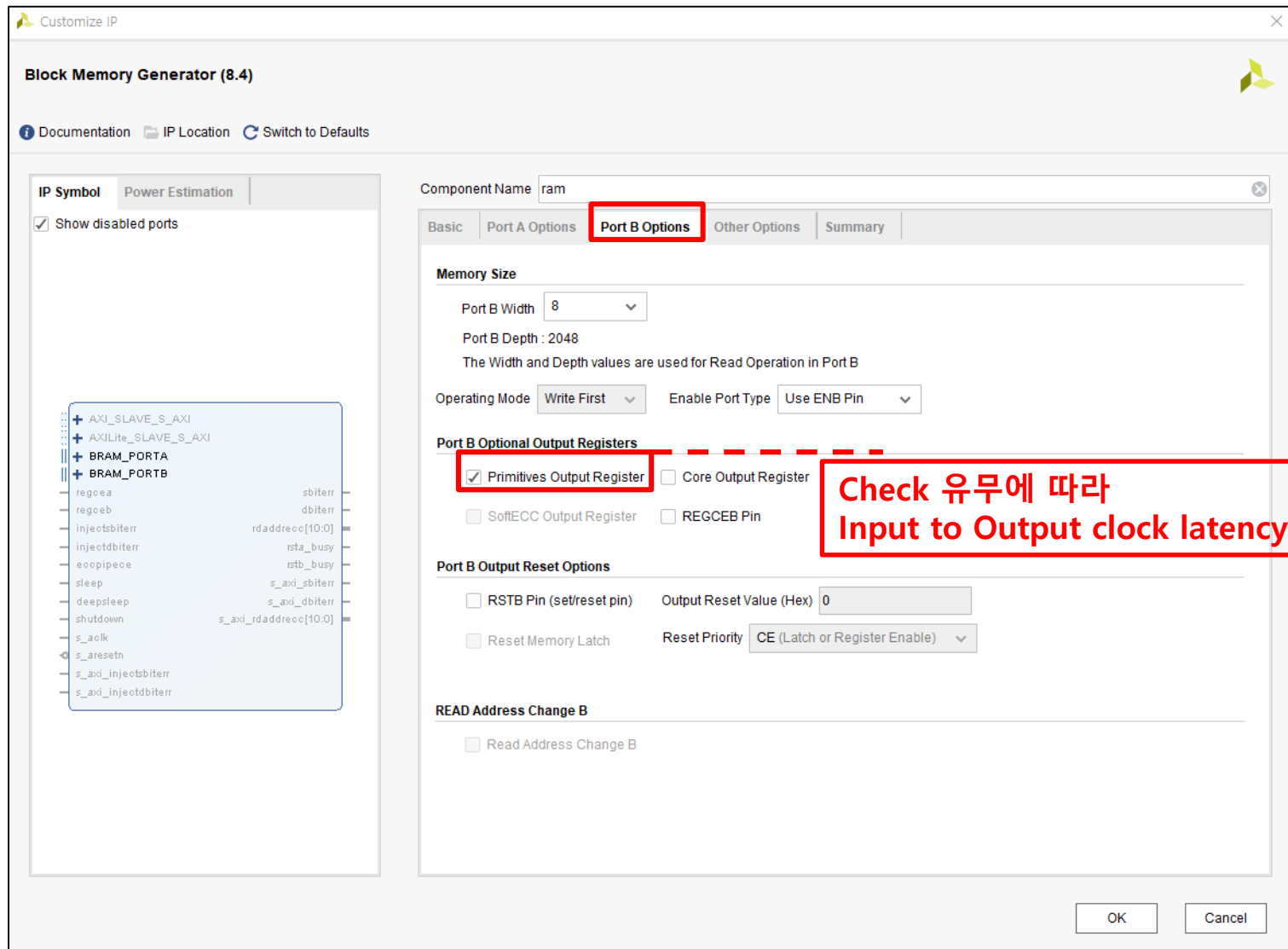
☐ Reset Memory Latch Reset Priority CE (Latch or Register Enable)

READ Address Change A

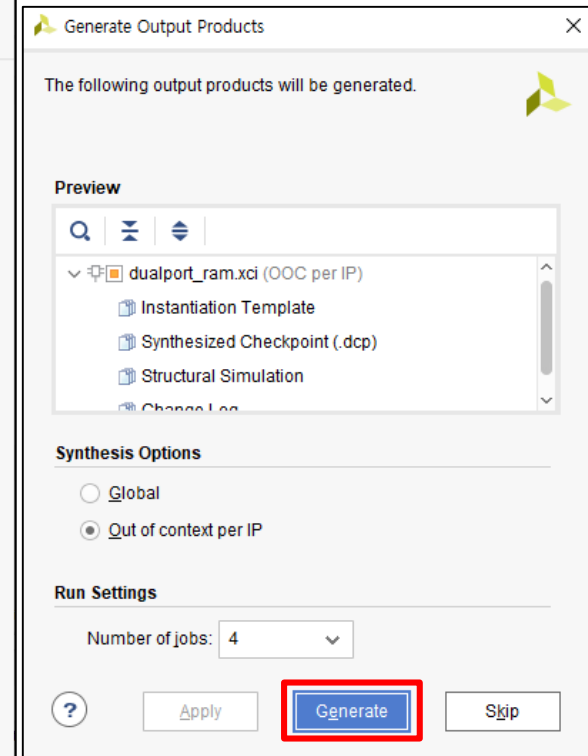
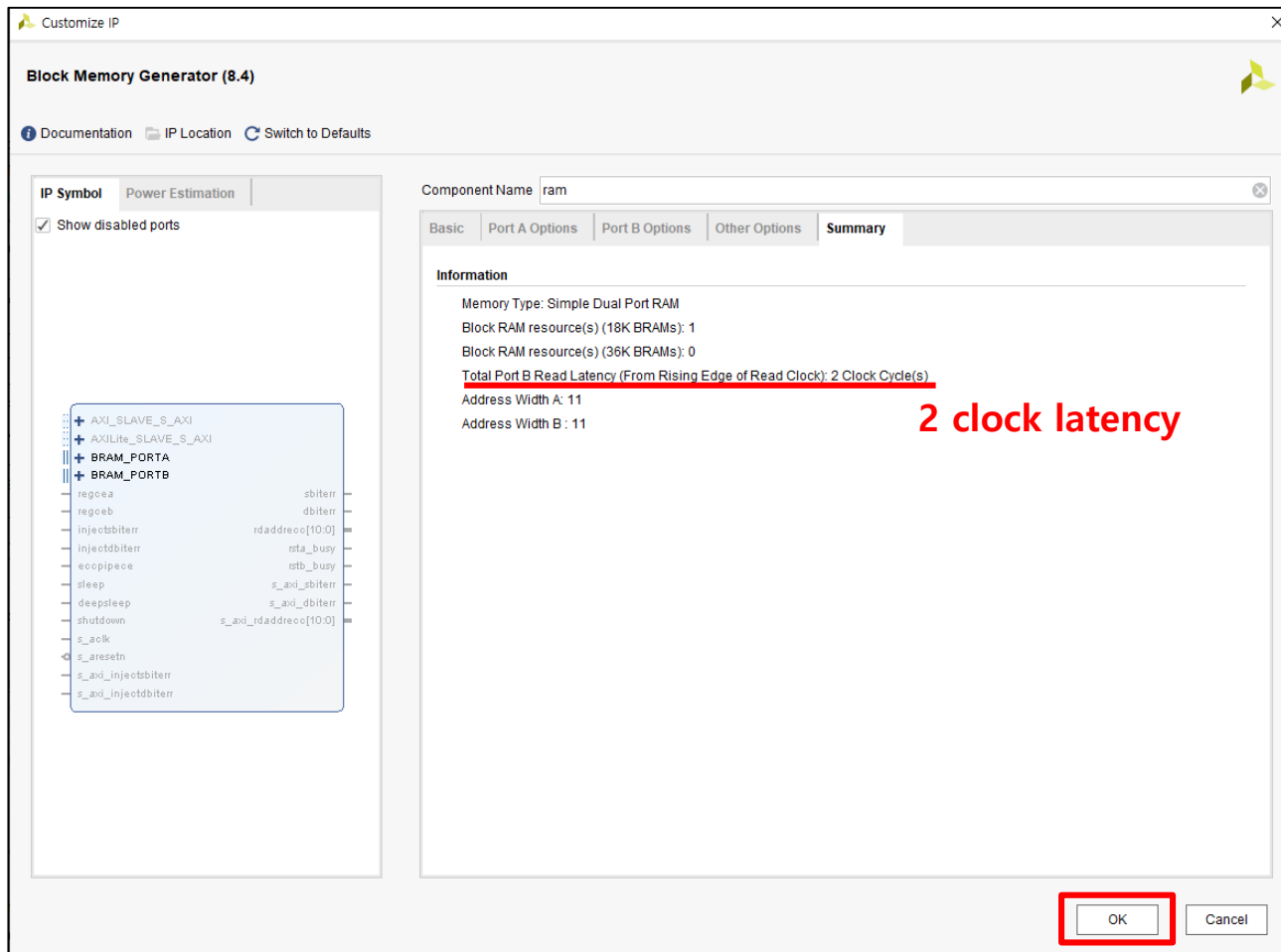
☐ Read Address Change A

AXI_SLAVE_S_AXI
AXILite_SLAVE_S_AXI
BRAM_PORTA
BRAM_PORTB
regcea sbiterr
regceb dbiterr
injectdbiterr rdaddress[10:0]
injectdbiterr rsta_busy
ecopipece rstb_busy
sleep s_axi_sbiterr
deepsleep s_axi_dbiterr
shutdown s_axi_rdaddress[10:0]
s_alk
s_arseln
s_axi_injectdbiterr
s_axi_injectdbiterr

❖ Port B도 마찬가지로 설정

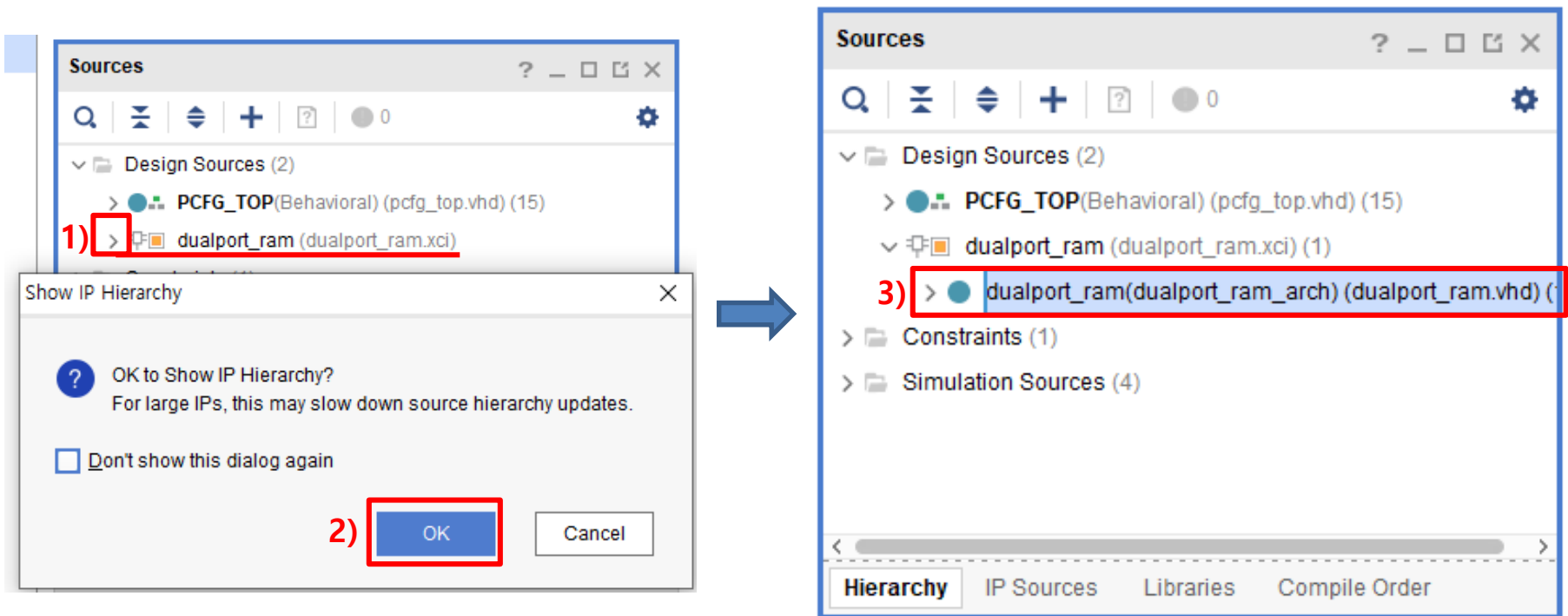


- ❖ 설정한 대로 RAM 생성을 완료하기 위해 'OK' 클릭
- ❖ Generate Output Products 창의 'Generate' 클릭



'Generate' 클릭

- ❖ IP가 생성되면 기본적으로 하위 모듈로 인식되지 않음
- ❖ 생성한 IP 옆 화살표 클릭 -> 'OK' 클릭 -> vhd 파일 클릭

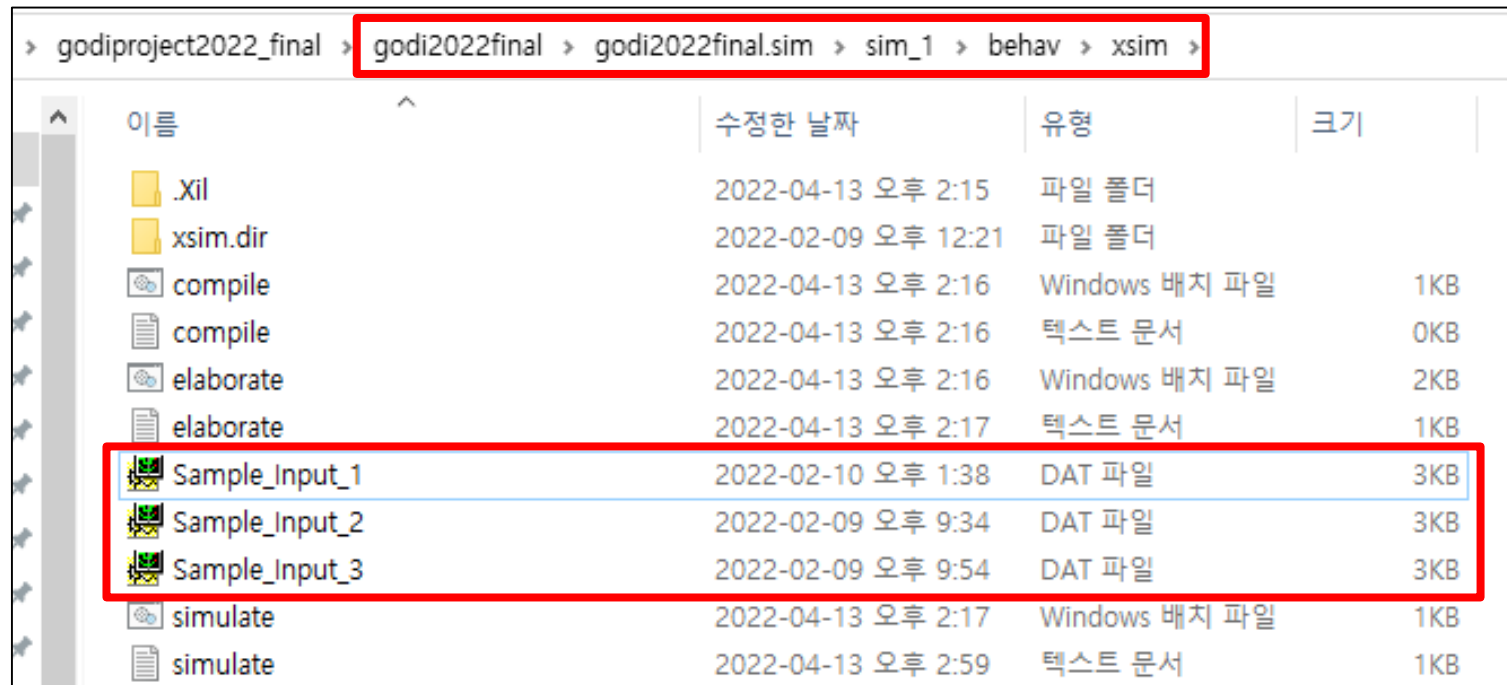


- ❖ vhd 파일을 열어 IP의 input/output port를 확인 후 component & port mapping
 - wea 신호는 1bit 이지만 std_logic_vector(0 downto 0)으로 설정
 - 연결할 signal도 꼭 std_logic_vector(0 downto 0)으로 선언한 뒤 연결
 - 3개의 RAM(PC_RAM, DA_RAM, AD_RAM)필요 : component 선언 후 3번 port mapping

```

dualport_ram.vhd
52  LIBRARY ieee;
53  USE ieee.std_logic_1164.ALL;
54  USE ieee.numeric_std.ALL;
55
56  LIBRARY blk_mem_gen_v8_4_1;
57  USE blk_mem_gen_v8_4_1.blk_mem_gen_v8_4_1;
58
59  ENTITY dualport_ram IS
60  PORT (
61      clka : IN STD_LOGIC;
62      ena : IN STD_LOGIC;
63      wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
64      addra : IN STD_LOGIC_VECTOR(10 DOWNTO 0);
65      dina : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
66      clkb : IN STD_LOGIC;
67      enb : IN STD_LOGIC;
68      addrb : IN STD_LOGIC_VECTOR(10 DOWNTO 0);
69      doutb : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
70  );
71  END dualport_ram;
72
73  ARCHITECTURE dualport_ram_arch OF dualport_ram IS
74      ATTRIBUTE DowngradeIPIdentifiedWarnings : STRING;
75      ATTRIBUTE DowngradeIPIdentifiedWarnings OF dualport_ram_arch: ARCHITECTURE IS "yes";
    
```

- ❖ '\프로젝트 폴더\프로젝트명.sim\sim_1\behav\xsim' 에 제공한 .dat 파일 추가
 - (한글경로x)

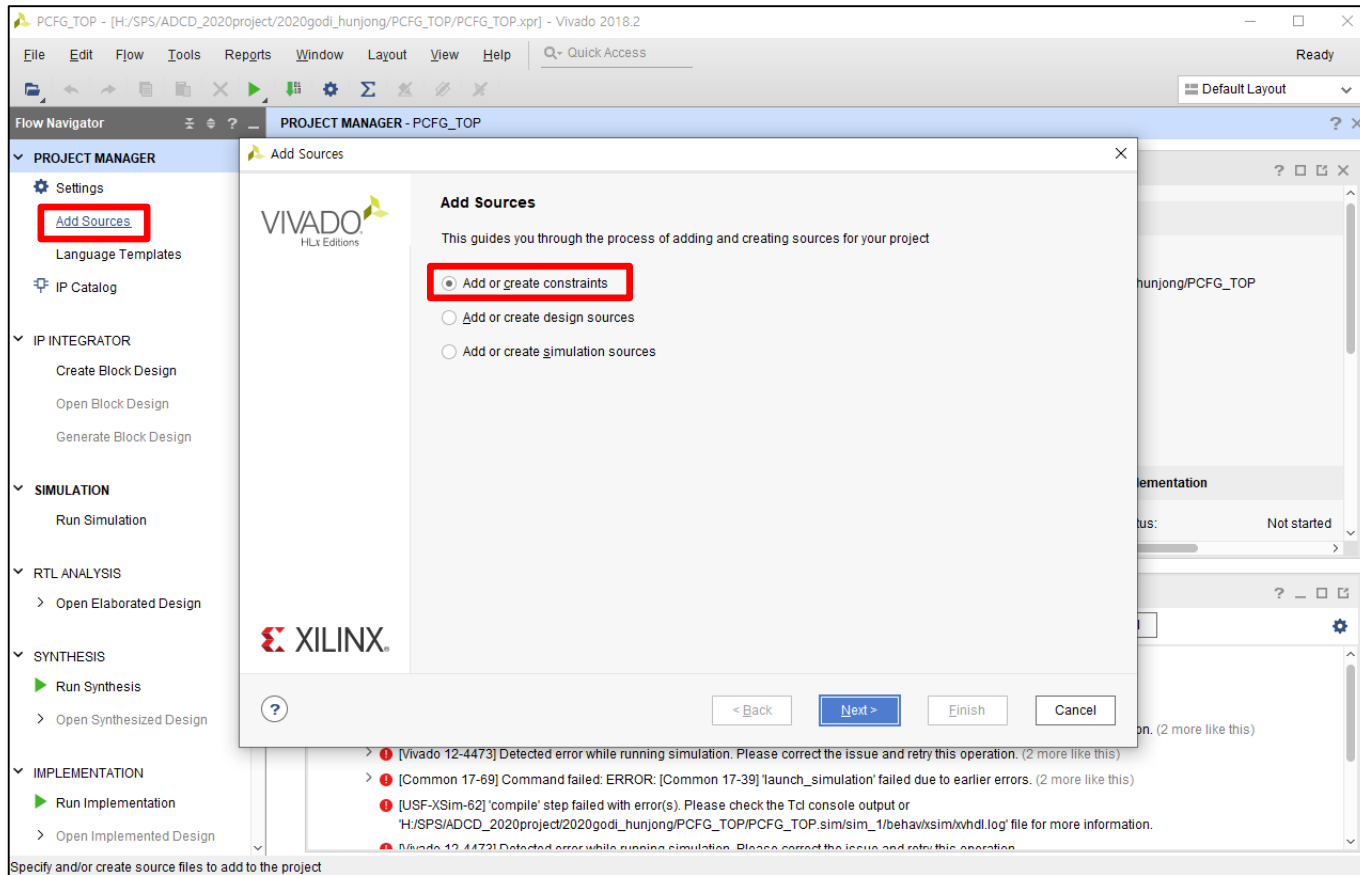


이름	수정한 날짜	유형	크기
.Xil	2022-04-13 오후 2:15	파일 폴더	
xsim.dir	2022-02-09 오후 12:21	파일 폴더	
compile	2022-04-13 오후 2:16	Windows 배치 파일	1KB
compile	2022-04-13 오후 2:16	텍스트 문서	0KB
elaborate	2022-04-13 오후 2:16	Windows 배치 파일	2KB
elaborate	2022-04-13 오후 2:17	텍스트 문서	1KB
Sample_Input_1	2022-02-10 오후 1:38	DAT 파일	3KB
Sample_Input_2	2022-02-09 오후 9:34	DAT 파일	3KB
Sample_Input_3	2022-02-09 오후 9:54	DAT 파일	3KB
simulate	2022-04-13 오후 2:17	Windows 배치 파일	1KB
simulate	2022-04-13 오후 2:59	텍스트 문서	1KB

- .dat 파일 : Option mode Test bench(Option_mode.vhd)의 Input을 넣을 때 사용

❖ 'Add Sources -> Add or create constraints' 에서 제공한 .xdc 파일 추가

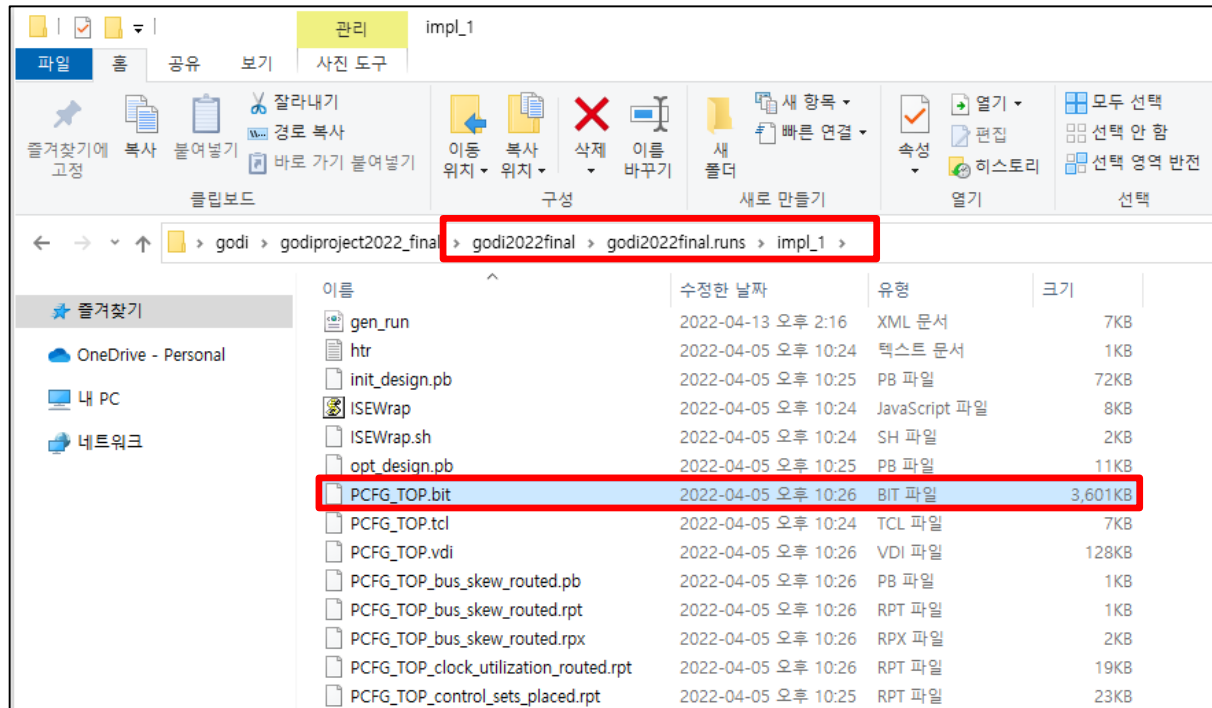
▪ (한글경로x)



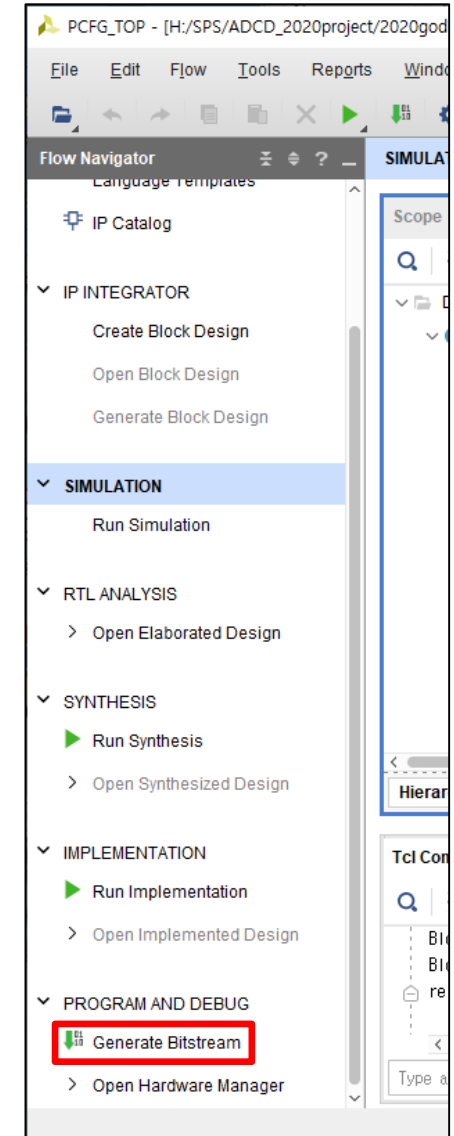
▪ Constraints 파일 : FPGA와 외부 IC chip들과의 pin mapping을 해주는 파일

❖ 왼쪽 하단에 'Generate Bitstream' 클릭 (5~10분 소요)

- Bit 파일은 해당 프로젝트 폴더 -> 프로젝트명.runs -> impl_1 폴더에 생성됨



- Bit 파일 : VHDL로 작성한 소스코드에 따라 FPGA가 동작하도록 만드는 파일

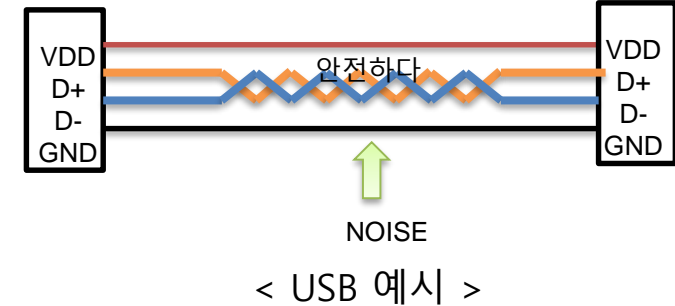


USB Protocol

- ❖ Differential로 되어있어 noise에 강함
- ❖ 약 100m까지 긴 케이블로도 통신 가능

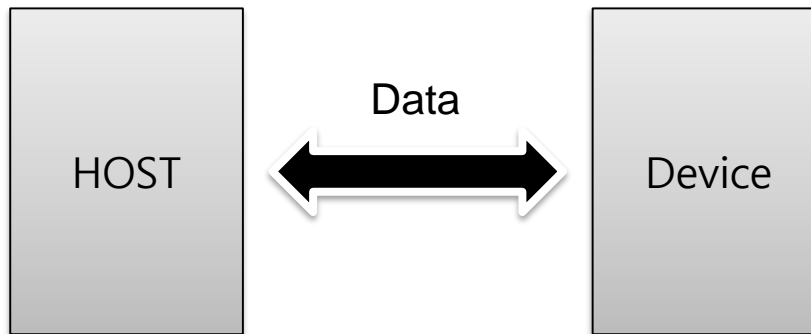


핀	기능
1	V _{BUS} (5V)
2	D-
3	D+
4	GND
Shell	Shield



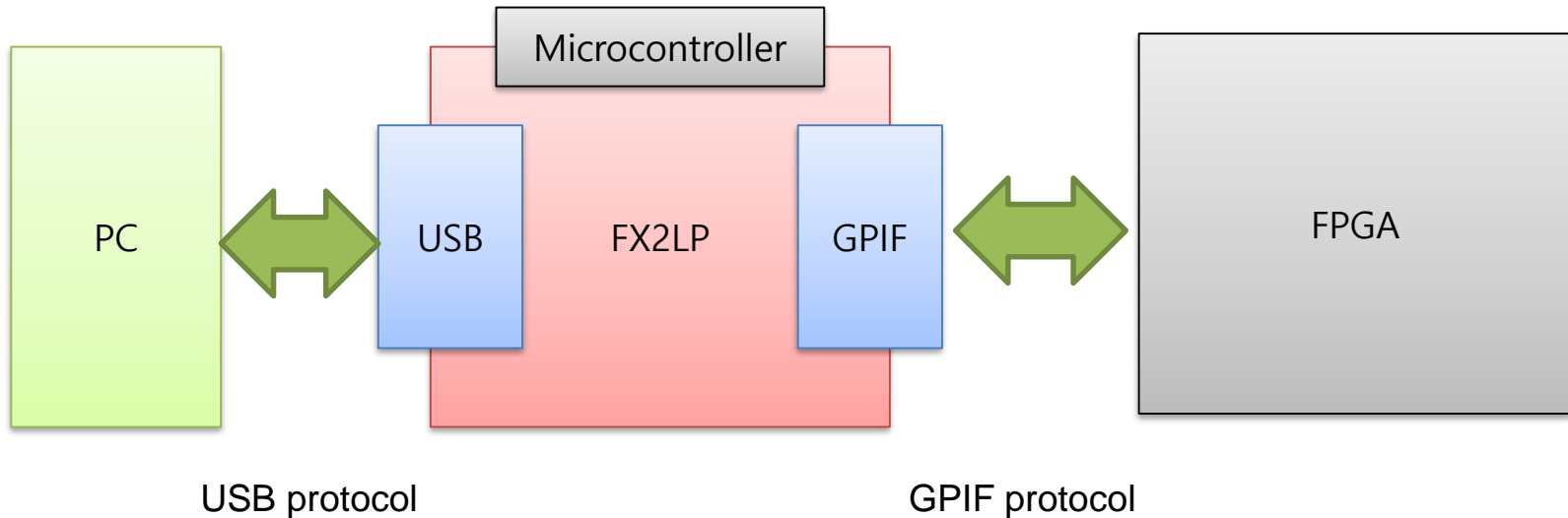
❖ Host와 Device간의 통신 수행

- 둘 중 하나는 반드시 Host
 - Host의 예 : PC 등등 ..
 - Device의 예 : 휴대폰, 고디보드, USB 저장장치 등등 ..



❖ In project

- Host : PC
- Device : Microcontroller



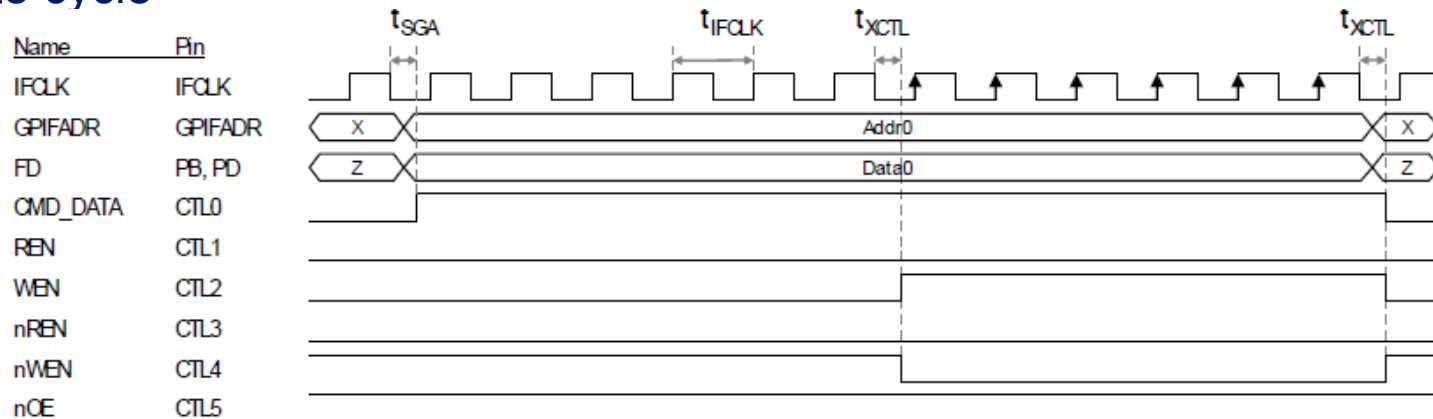
- Write by PC : USB통신으로 PC에서 받은 데이터를 parallel signal로 바꾸어 FPGA로 보냄
- Read by PC : USB통신으로 PC에서 데이터를 요청 받으면 parallel signal로 FPGA의 데이터를 PC로 전송

❖ Quick USB support several protocol for GPIF

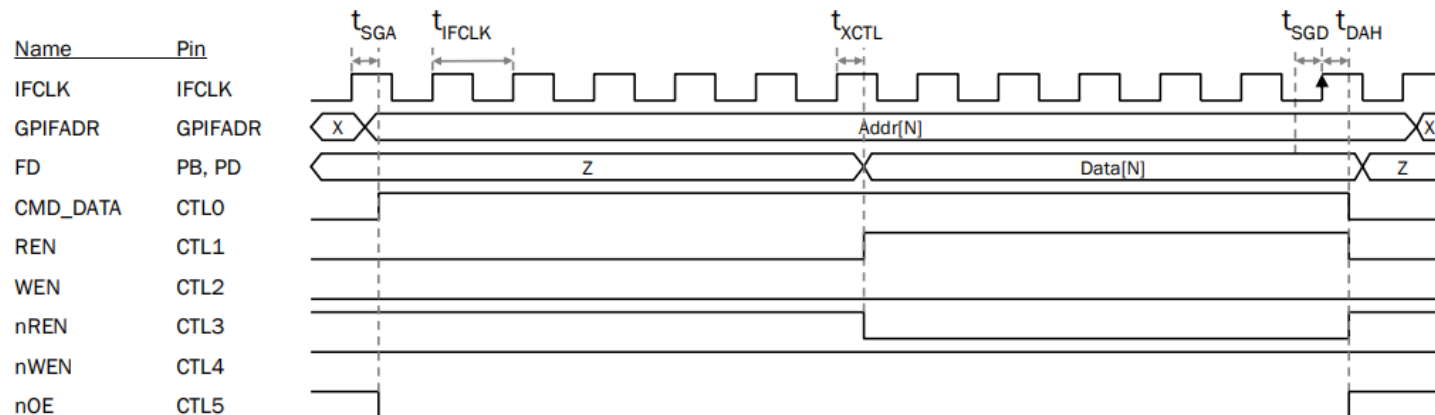
- Simple I/O
- **FIFO Handshake I/O**
- Block Handshake I/O
- Full Handshake I/O
- Slave FIFO I/O
- Slave245 I/O
- Pipeline I/O
- ...

IFCLK = 48MHz

❖ Write cycle



❖ Read cycle

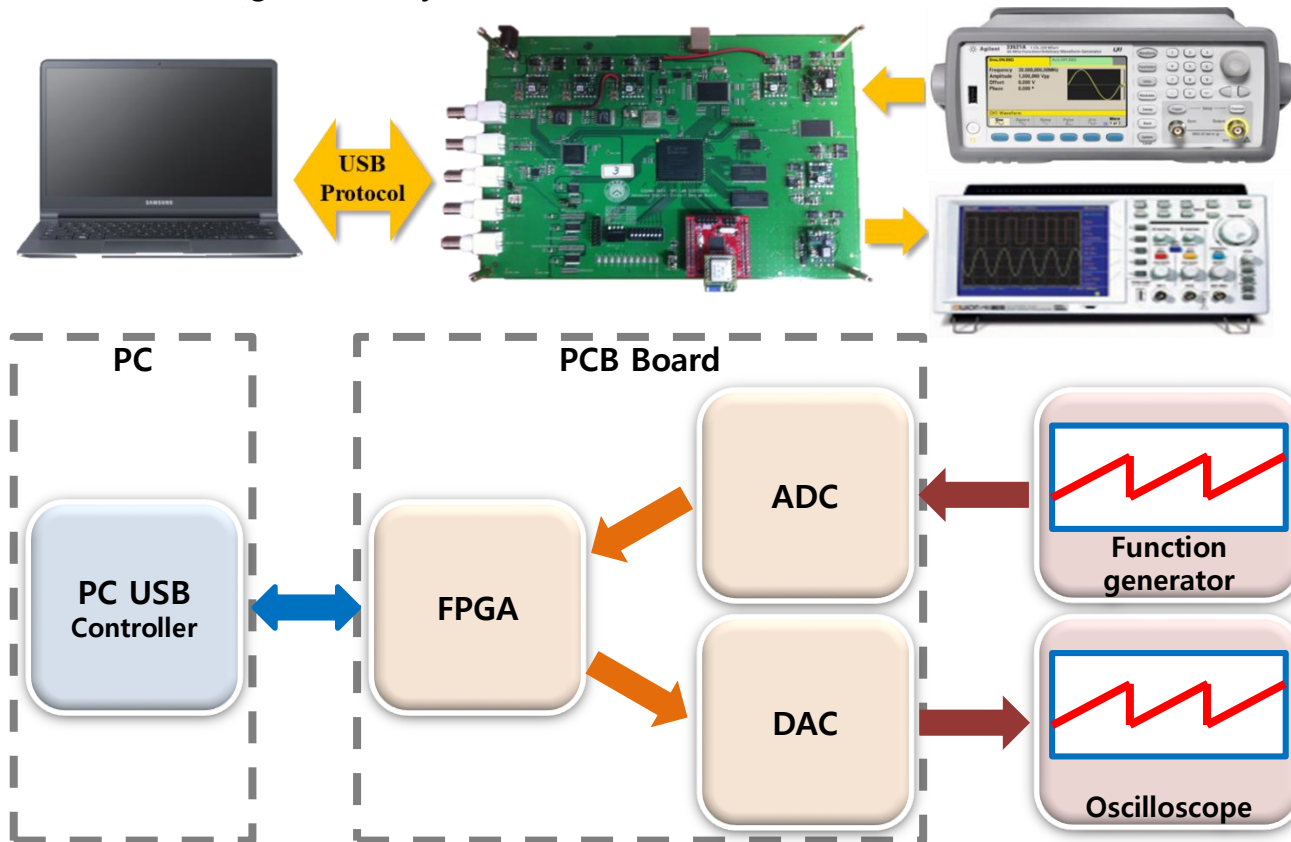


- m_fpga_clk = 40MHz 이므로 10clk 안에 write/read 동작을 수행해야 함

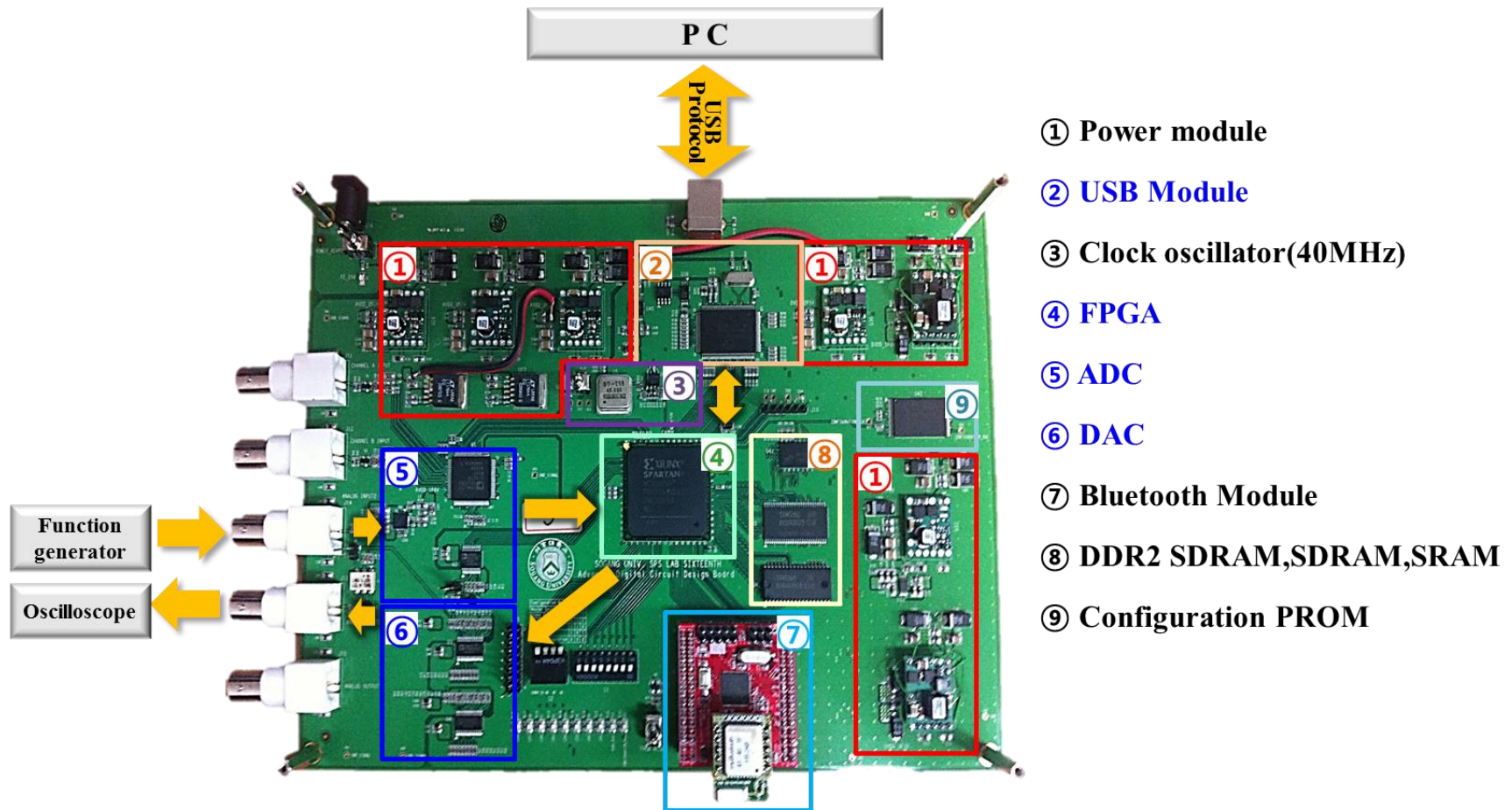
PC Based Function Generator

❖ 목적

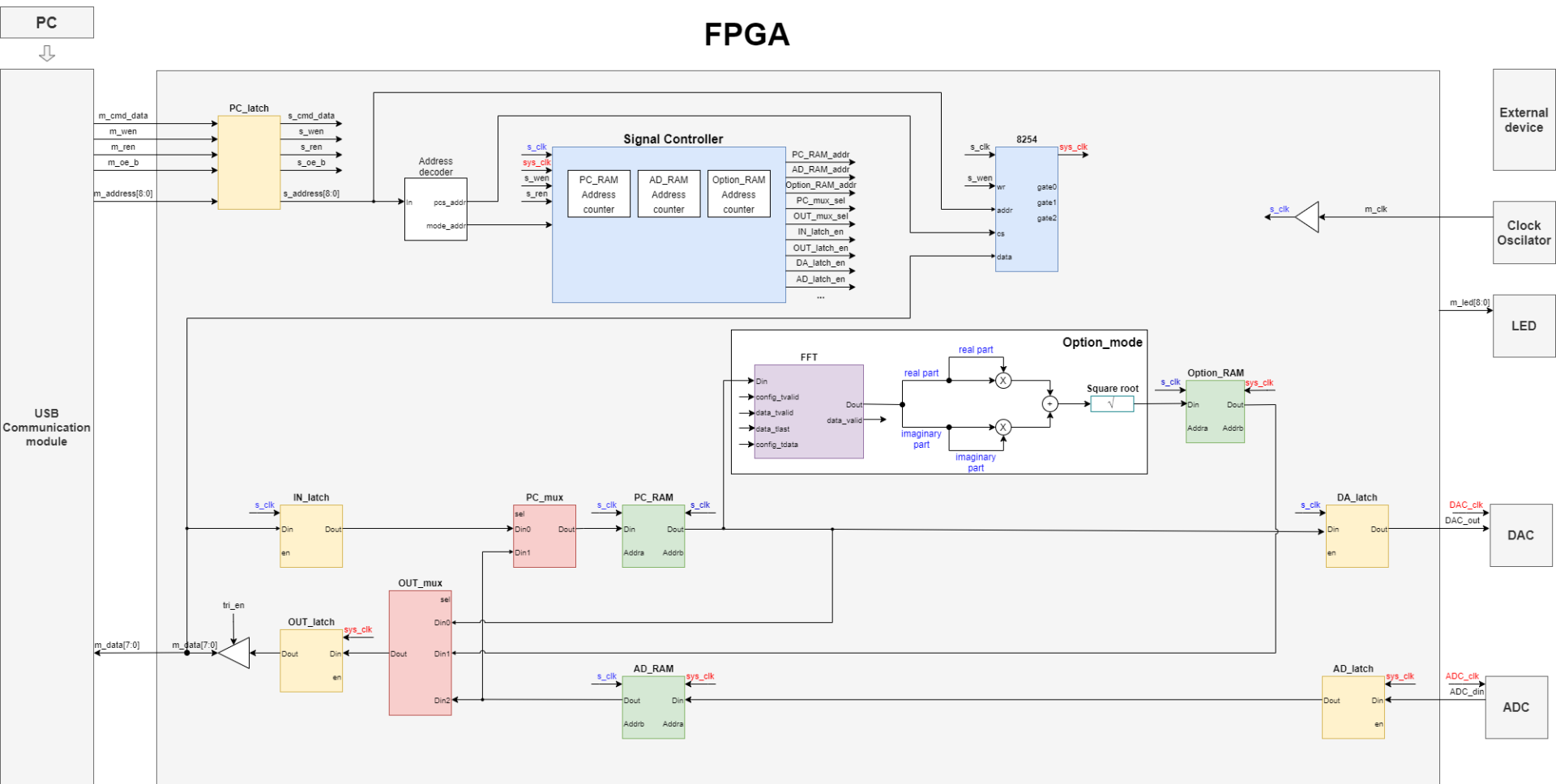
- 내가 원하는 임의의 파형을 생성할 수 있는 PC기반의 Function generator를 구현
 - PC와 FPGA간의 USB통신 방식 이해
 - Bi-directional bus에서의 data control 을 통한 data read/write
 - Address decoding을 통한 system동작 control



❖ Test board



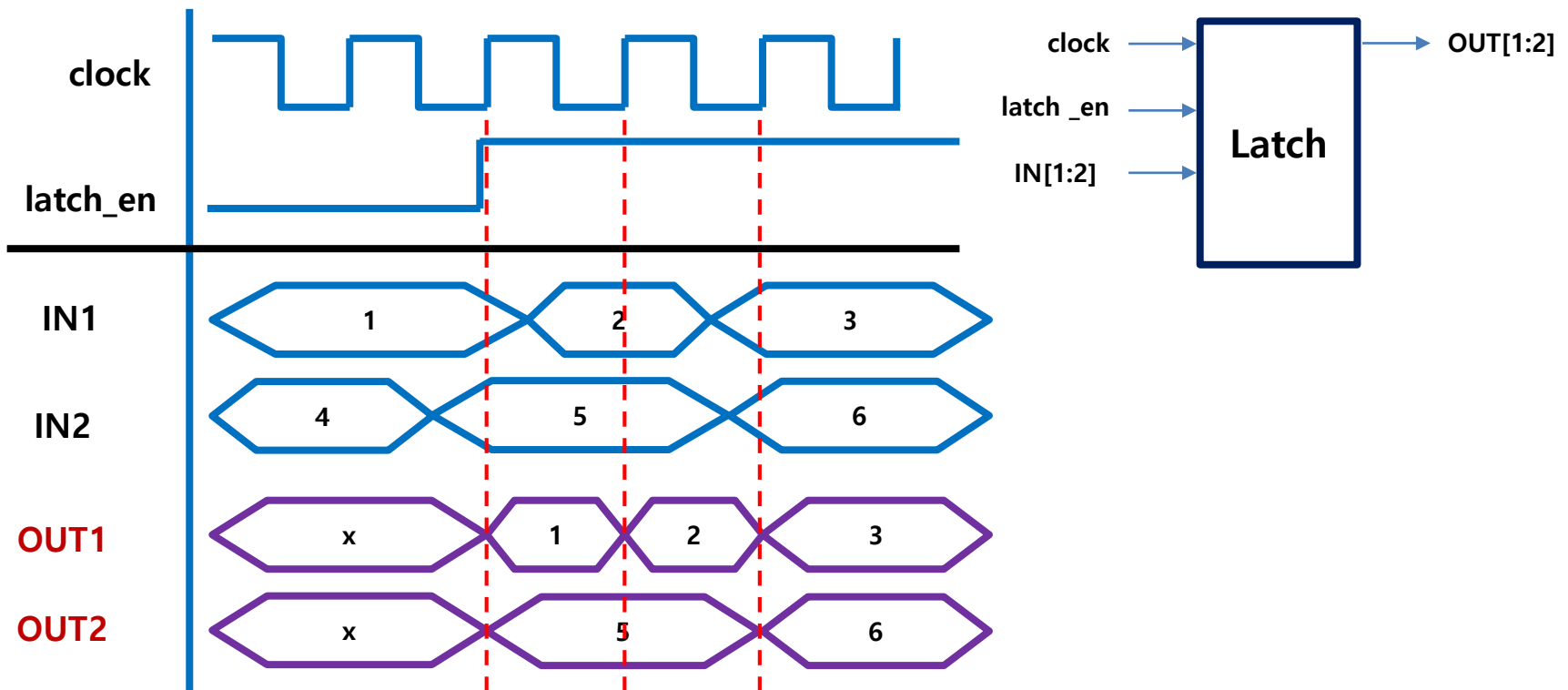
❖ Overall block diagram



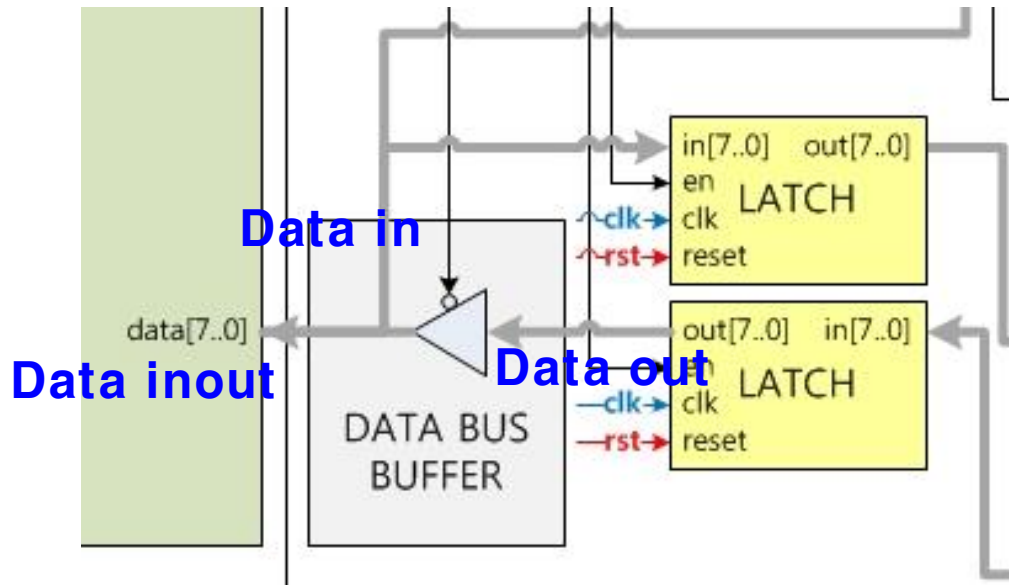
- 본인이 원하는 디자인 구현 가능

❖ Latch 사용 이유

- 각 신호들을 clock에 synchronize 하게 설계하기 위해
- 데이터를 안정적으로 동작하기 위해



❖ Bus control



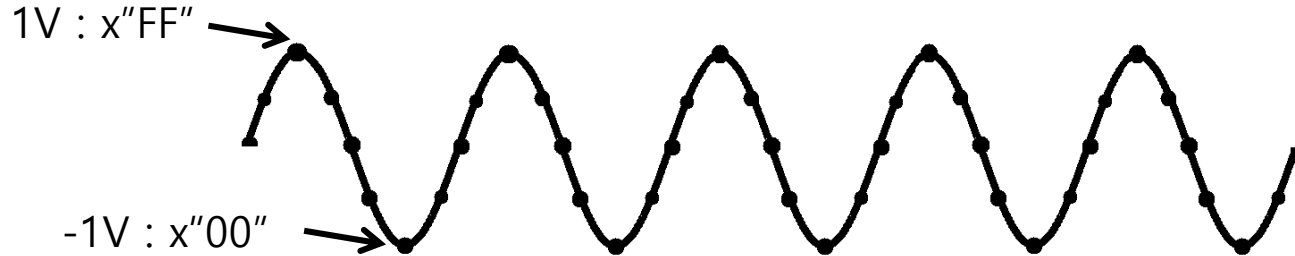
- Tri-buffer control 을 통해 INOUT 신호 제어

Bus control 예시

```
Data_in <= Data_inout;
Data_inout <= Data_out when Buffer_en = '0'
               else x"ZZ";
```

❖ Data format

- ADC 동작 : $-1V \sim +1V \rightarrow x"00" \sim x"FF"$

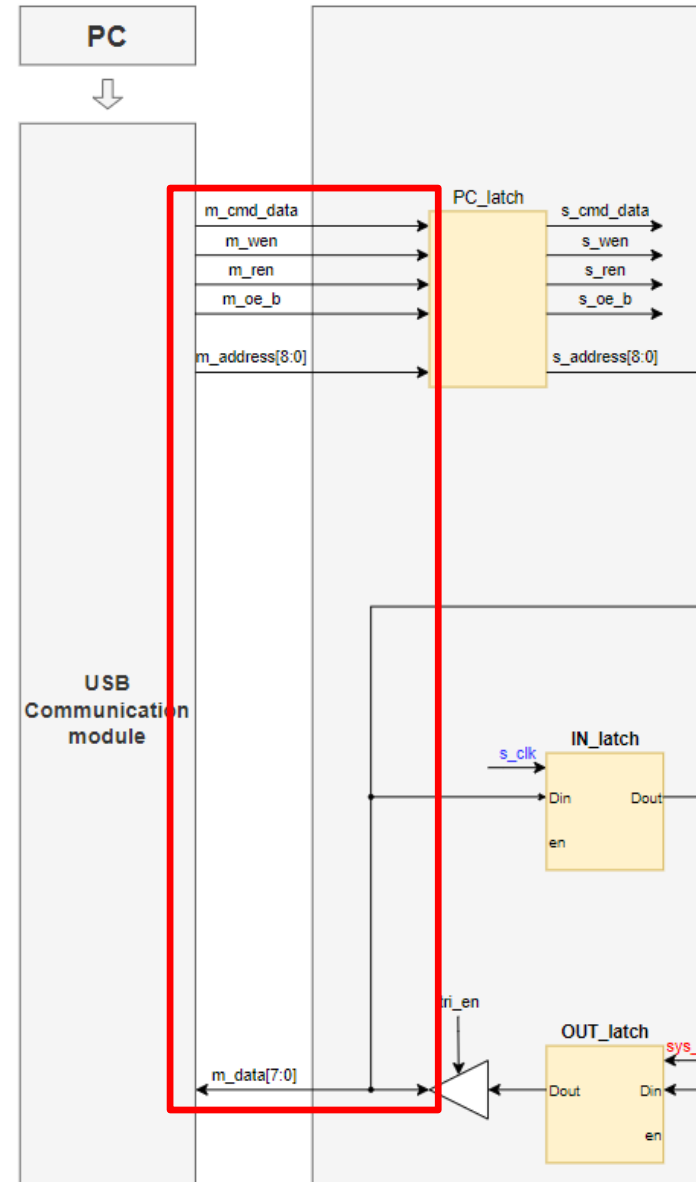


- 제공한 Option_mode.vhd은 Signed operation 으로 설계(아래의 Signed library 반드시 추가)
- 이외의 VHDL code는 IEEE.STD_LOGIC_UNSIGNED.ALL 이용

```
LIBRARY ieee;  
USE ieee.std_logic_1164.ALL;  
use IEEE.STD_LOGIC_ARITH.ALL;  
use IEEE.STD_LOGIC_SIGNED.ALL;
```

❖ PC와 protocol을 위한 신호들

- m_address[8:0]
 - 각 mode를 선택하기 위한 address
- m_cmd_data
 - 유효한 m_address가 들어오고 있다는 신호
- m_wen & m_ren
 - data write/read 를 위한 신호
- m_oe_b
 - data read 를 위한 신호
- m_data[7:0]
 - mode control 또는 write/read 를 위한 신호

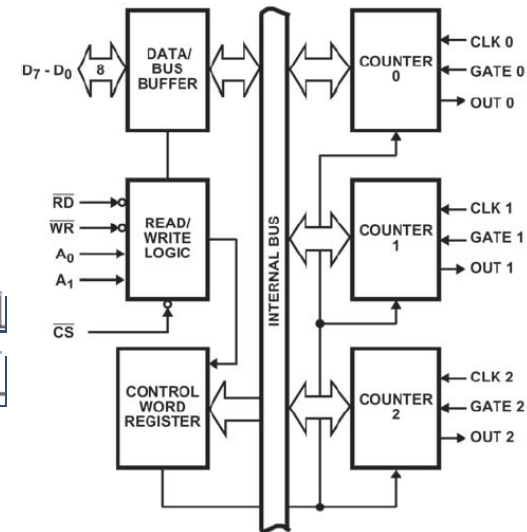
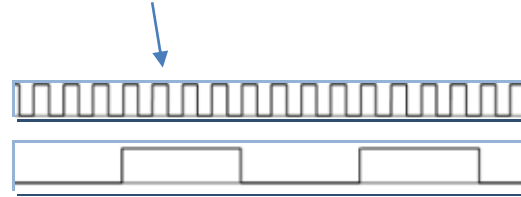


❖ 제공 파일

- 전체 Top file
 - Input, output 정의하는 부분은 절대 건드리지 말 것!
- 8254 file
 - 8254 설명서를 보면서 동작만 시키면 됨
- 시뮬레이션을 위한 test bench file
 - 자유롭게 수정하면서 시뮬레이션 가능 (시뮬레이션 결과는 시연과 관련 x)
- xdc file
 - VHDL code와 FPGA pin과 연결을 위한 파일

❖ 목적

- System 마다 요구하는 clock이 달라 주파수 변화(분주) 필요
- 프로젝트에서는 8254의 여러 기능 중 분주 기능만 사용
- 동작 및 사용법은 강의자료 참고

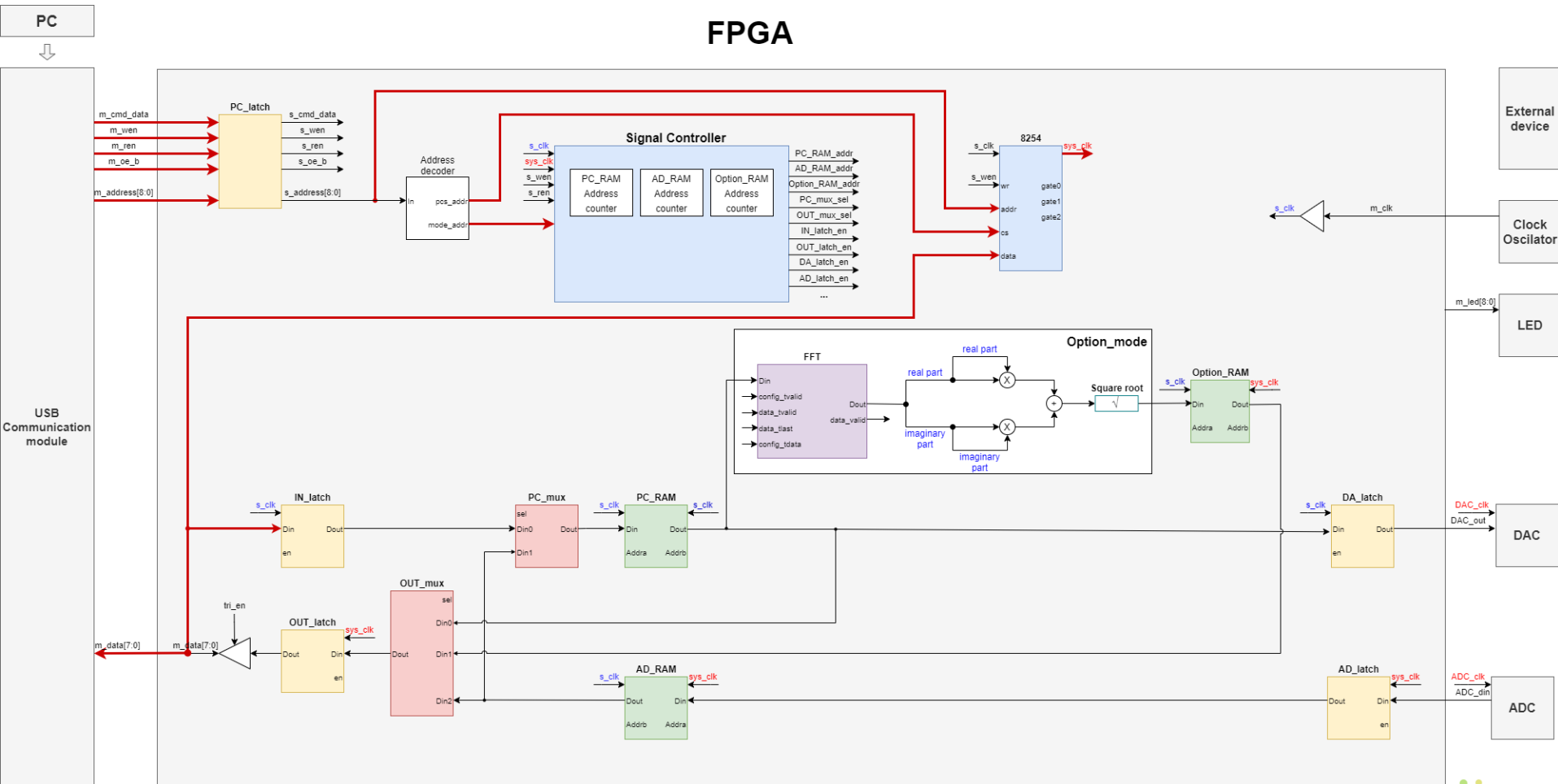


- RAM에서 data를 PC로 read 할 때 8254를 reset 후에 sys_clk를 s_clk와 동일하게 한 뒤 시뮬레이션 진행 (실제 시연도 동일한 과정)

1. System Clock Setting : 110H ~ 113H
2. Software Reset : 120H
3. 8254 Reset : 121H
4. PC Mode : 130H
5. DA Mode(start, stop) : 140H ~ 141H
6. AD Mode : 150H
7. ADR Mode : 151H
8. Option Mode(FFT) : 160H ~ 161H

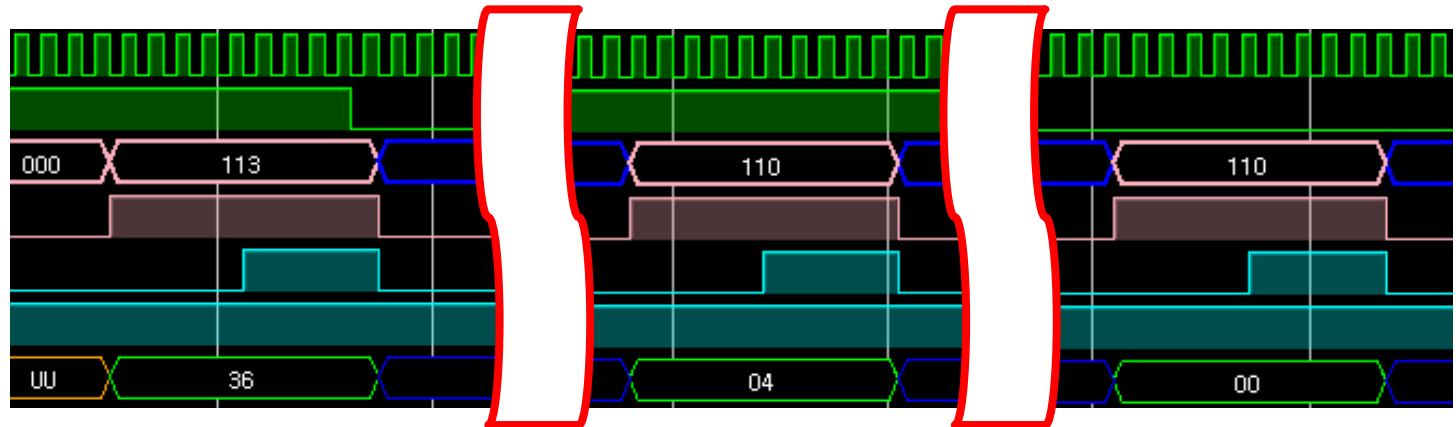
❖ System Clock Setting (110~113H)

- 3번으로 나뉘어져 들어오는 8254 programming data를 통해 s_clk를 분주 시켜 sys_clk를 출력한다. (8254 setting은 s_address 대신 m_address 사용 권장)

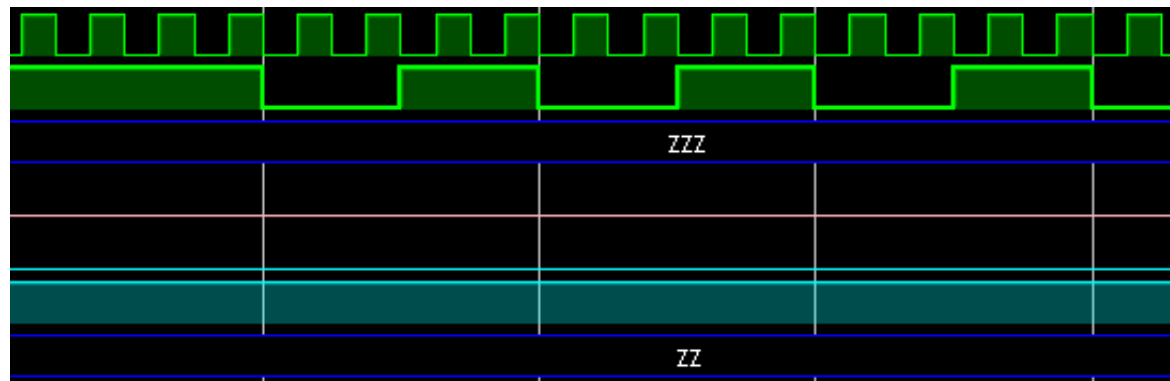


❖ System Clock Setting (simulation)

s_clk
 sys_clk
 m_address[8:0]
 m_cmd_data
 m_wen
 m_oe_b
 m_data[7:0]



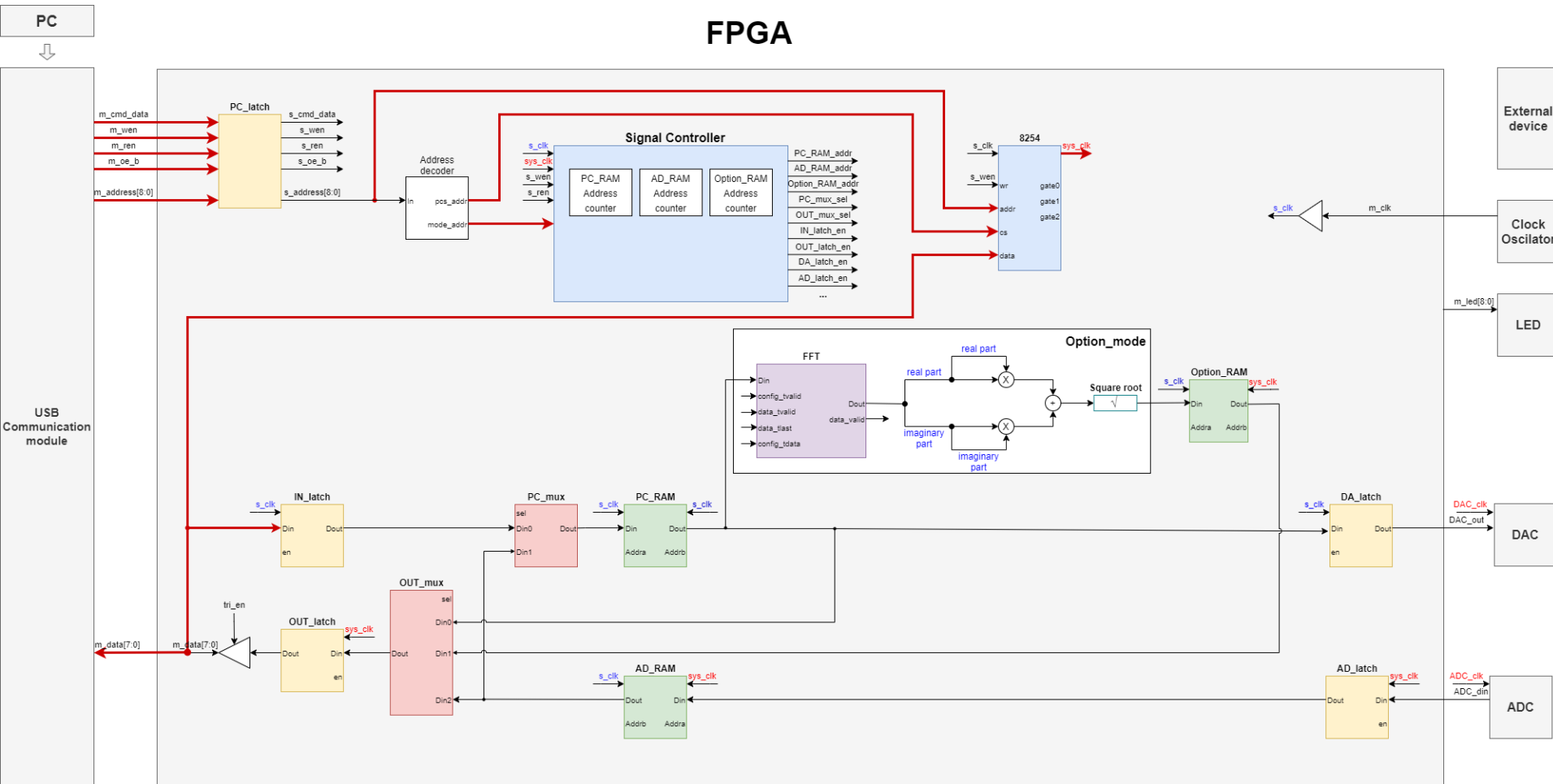
s_clk
 sys_clk
 m_address[8:0]
 m_cmd_data
 m_wen
 m_oe_b
 m_data[7:0]



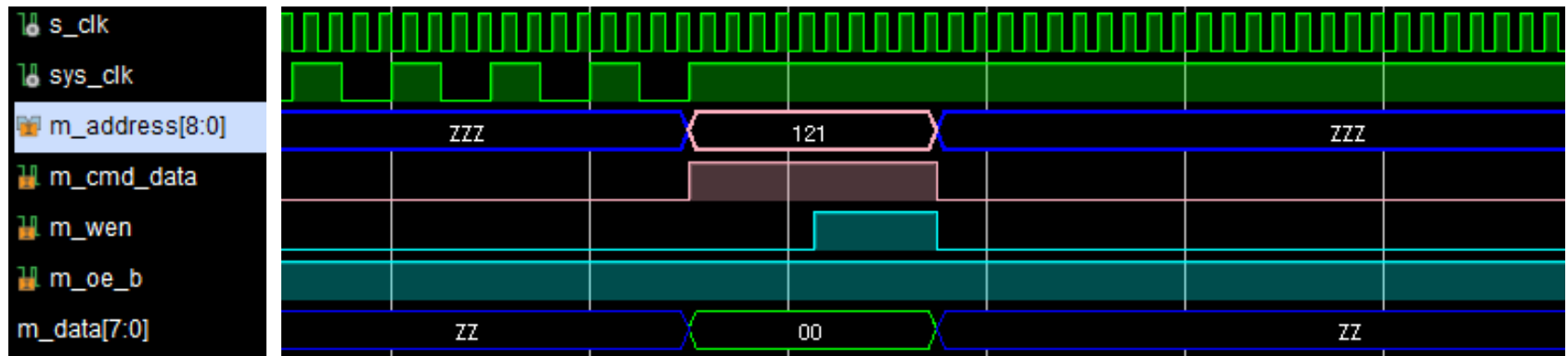
← 4 분주된
sys_clk

❖ 8254 Reset (121H)

- 8254를 reset한다

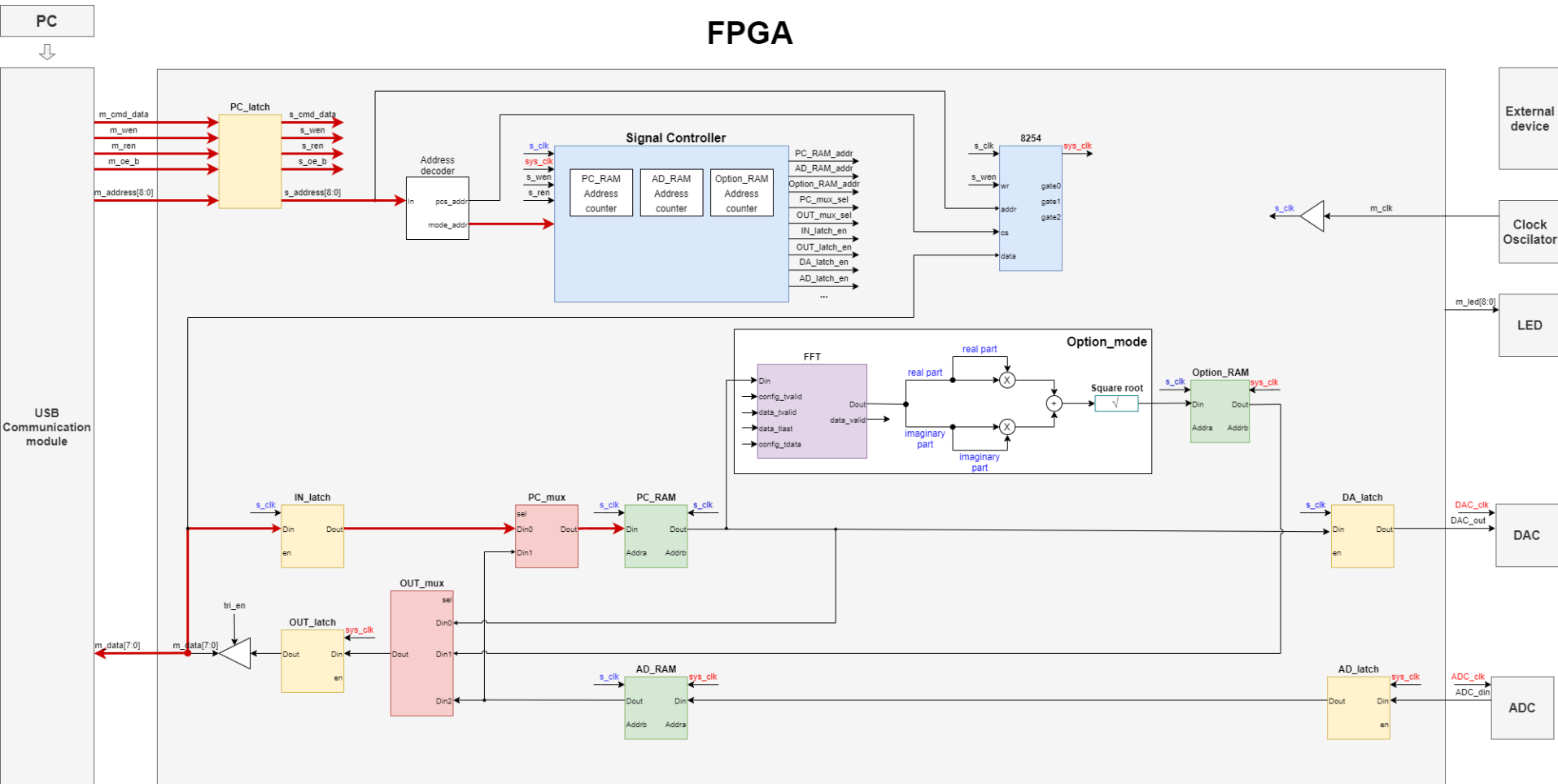


❖ 8254 Reset (simulation)



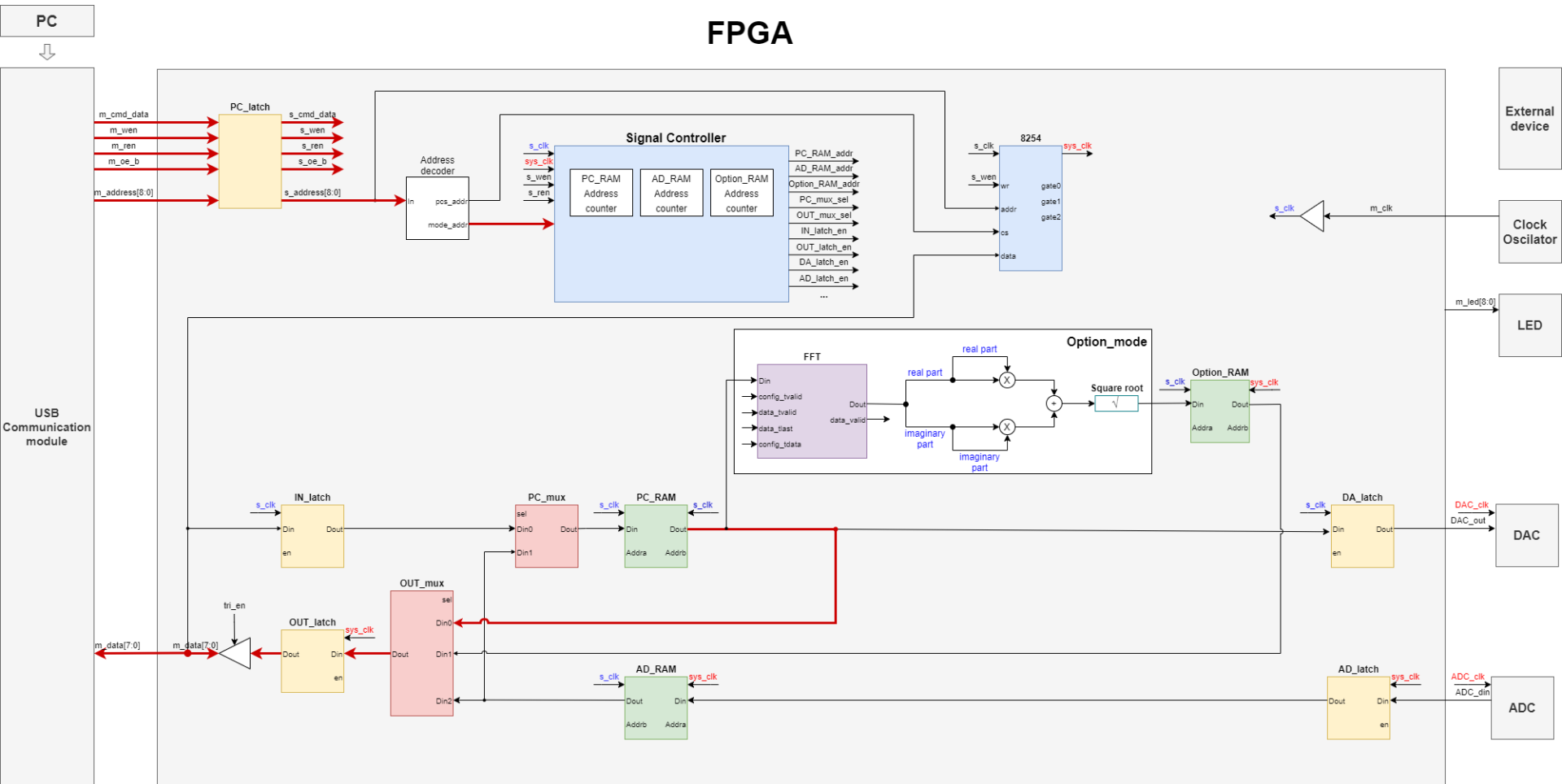
❖ PC Mode (130H)

- Write : PC_RAM에 데이터를 저장한다



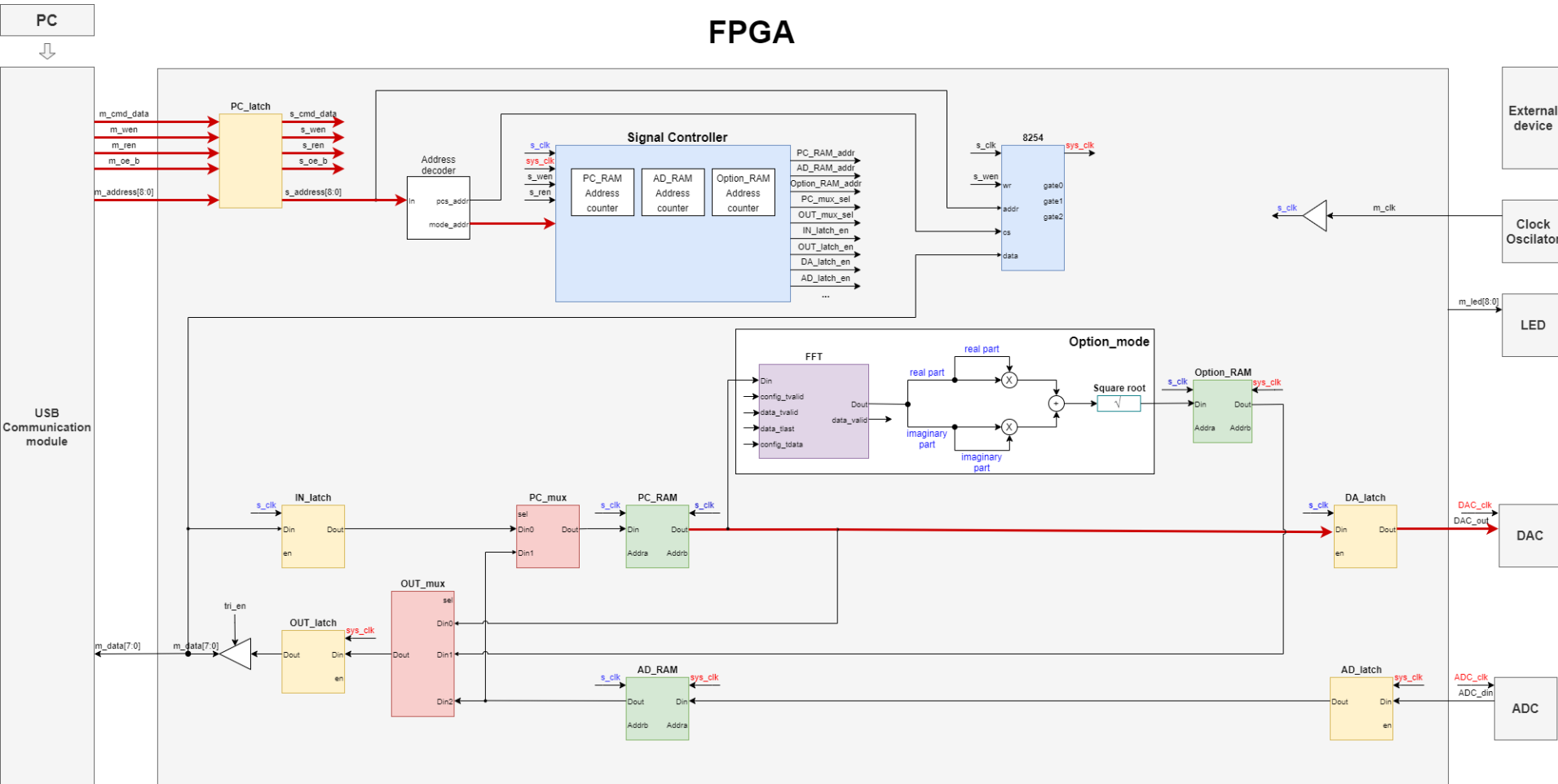
❖ PC Mode (130H)

- Read : PC_RAM의 데이터를 읽는다



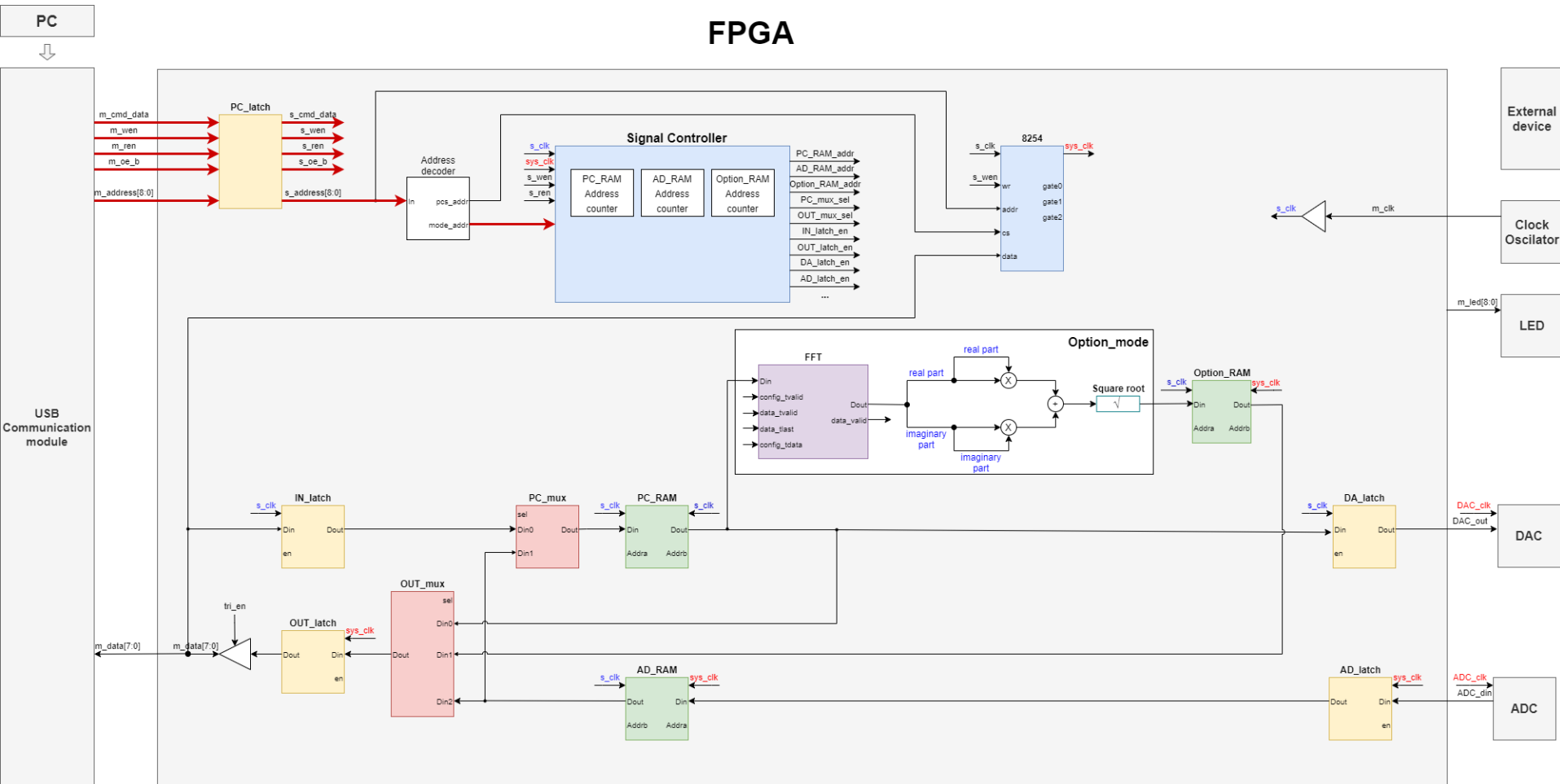
❖ DA start Mode (140H)

- Start : PC_RAM을 read해서 DAC로 내보냄
- PC_RAM에 써진 data 개수만큼 무한반복해서 읽어내야 함

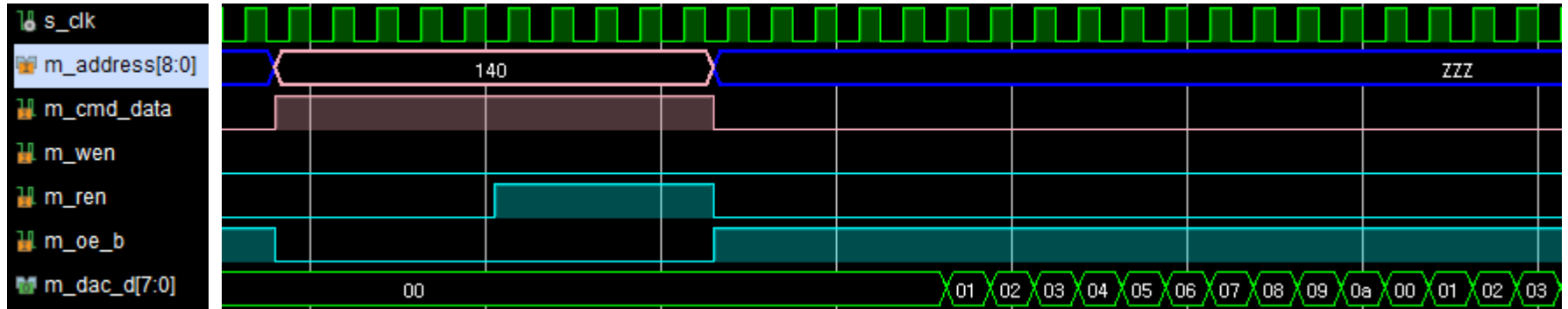


❖ DA stop Mode (141H)

- Stop : PC_RAM의 출력을 멈춤



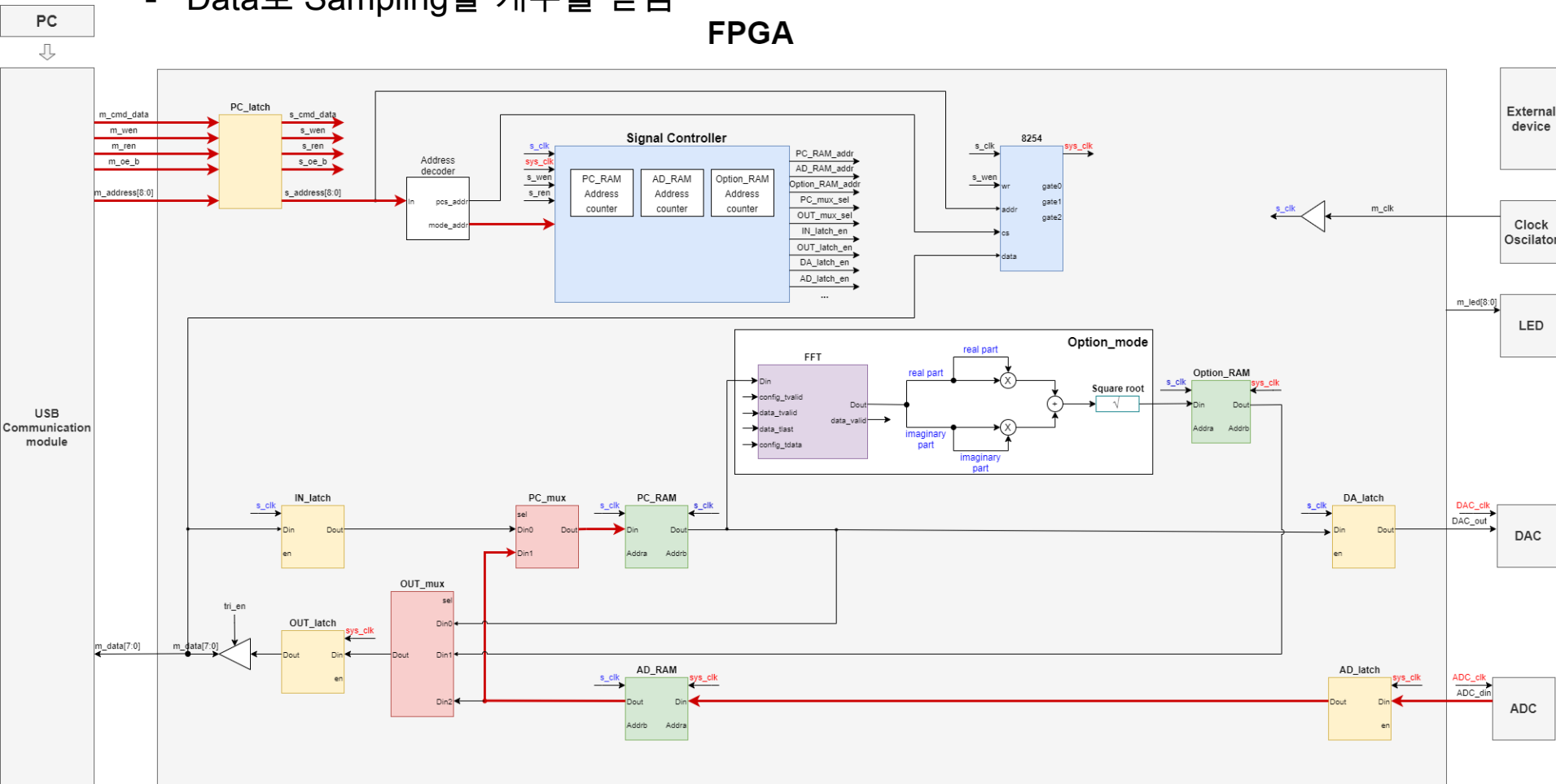
❖ DA Mode (simulation)



❖ AD Mode (150H)

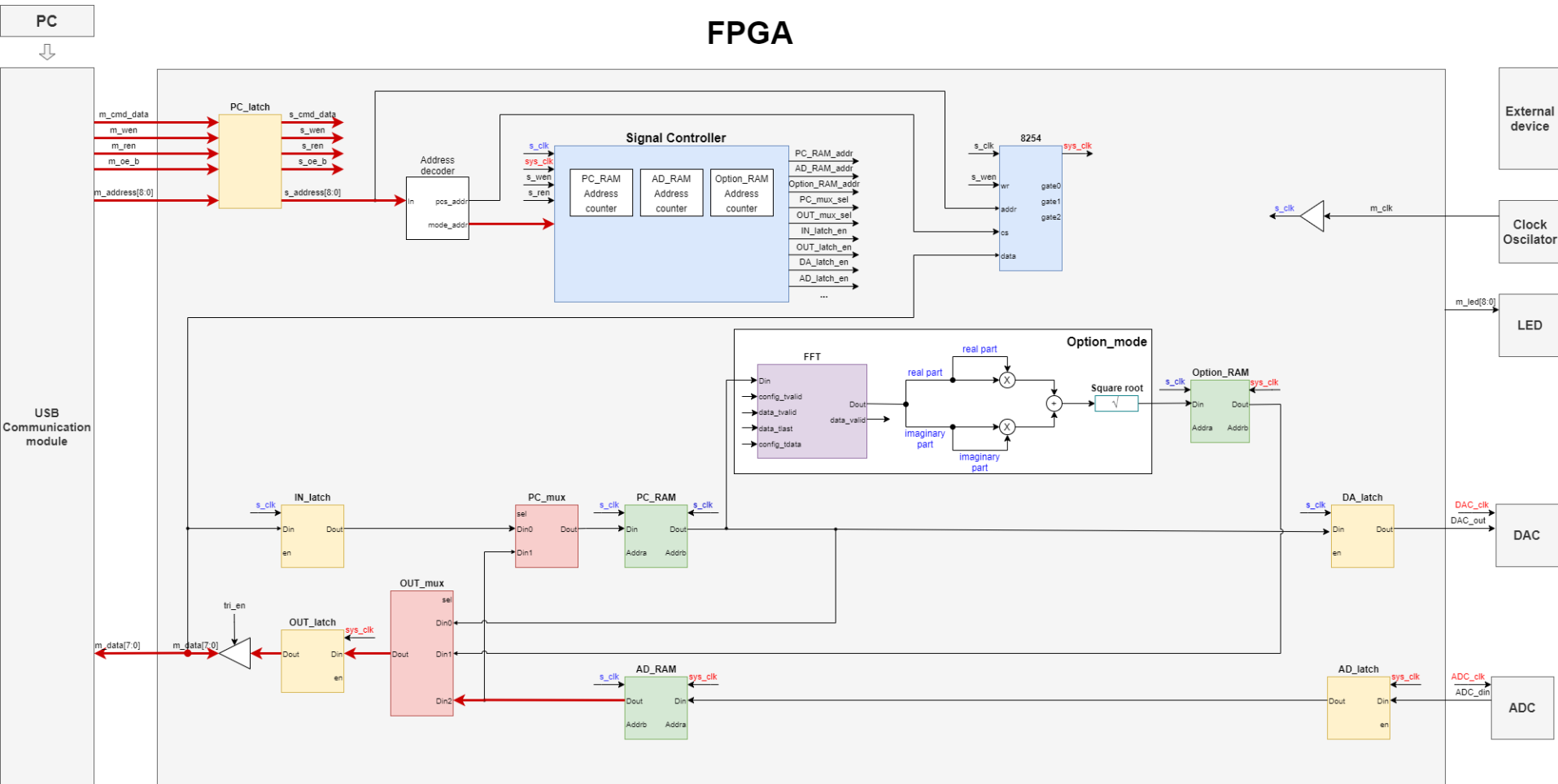
- ADC의 데이터를 AD_RAM을 거쳐 PC_RAM에 저장
- Data로 Sampling할 개수를 받음

FPGA



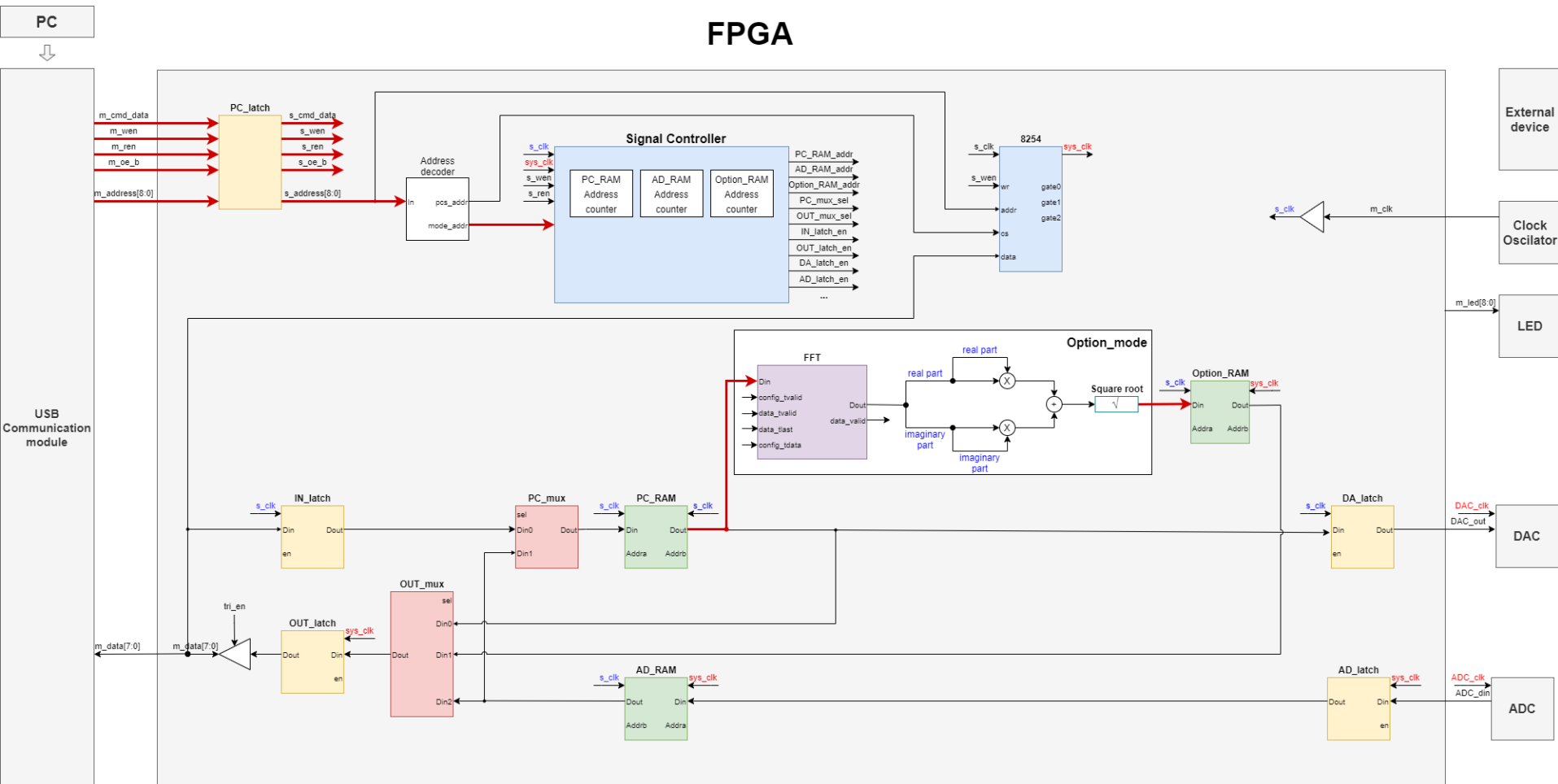
❖ ADR Mode (151H)

- AD mode에서 AD_RAM에 저장된 데이터를 read



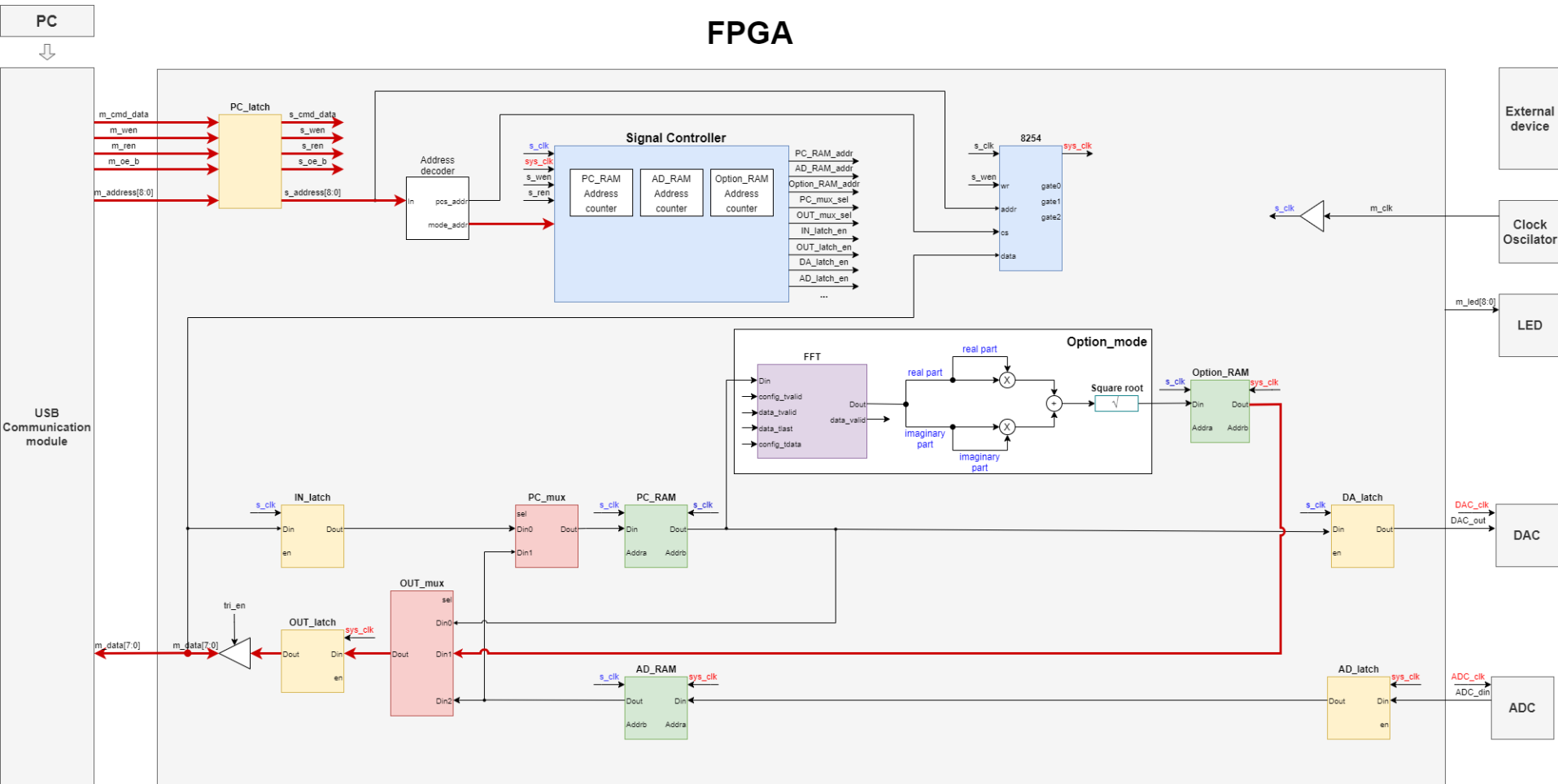
❖ Option Write Mode (160H)

- PC_RAM의 데이터 중 512 sample에 FFT & ABS 적용해 Option_RAM에 저장



❖ Option Read Mode (161H)

- Option_RAM의 데이터를 read



Option Mode

❖ Fast Fourier Transform (FFT)

▪ FFT란?

- Discrete Fourier Transform(DFT) 계산 시 연산 횟수를 줄일 수 있도록 고안된 알고리즘

N-point DFT

$$X(k) = \sum_{n=0}^{N-1} x(n) \underline{W_N^{kn}} \quad k = 0, 1, \dots, N-1$$

where $\underline{W_N = \exp(-j2\pi / N)}$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn} \quad n = 0, 1, \dots, N-1$$

Computational Advantage of FFT

	# of Multiplications	# of Additions
N-point DFT	N^2	$N(N-1)$
N-point FFT	$(N/2) \log_2 N$	$N \log_2 N$

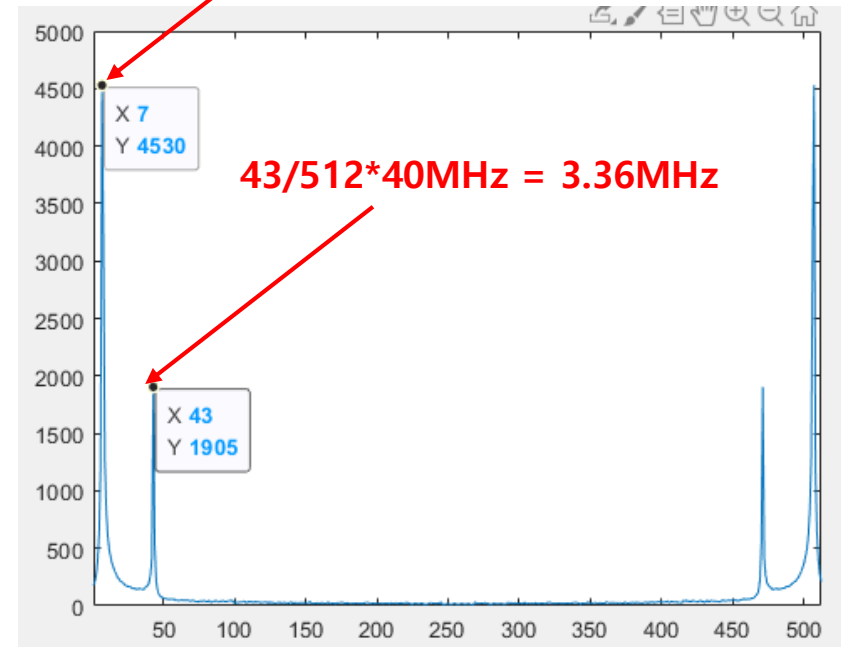
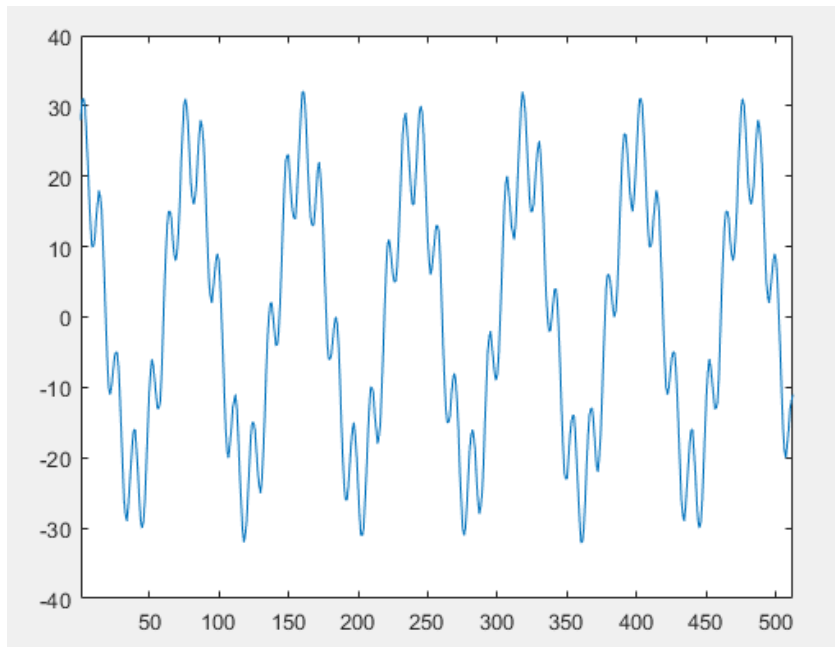
Note) Multiplications and additions are complex operations here.

Ex) $N = 1024$

→ 1M mul
→ 5K mul $\times \frac{1}{200}$

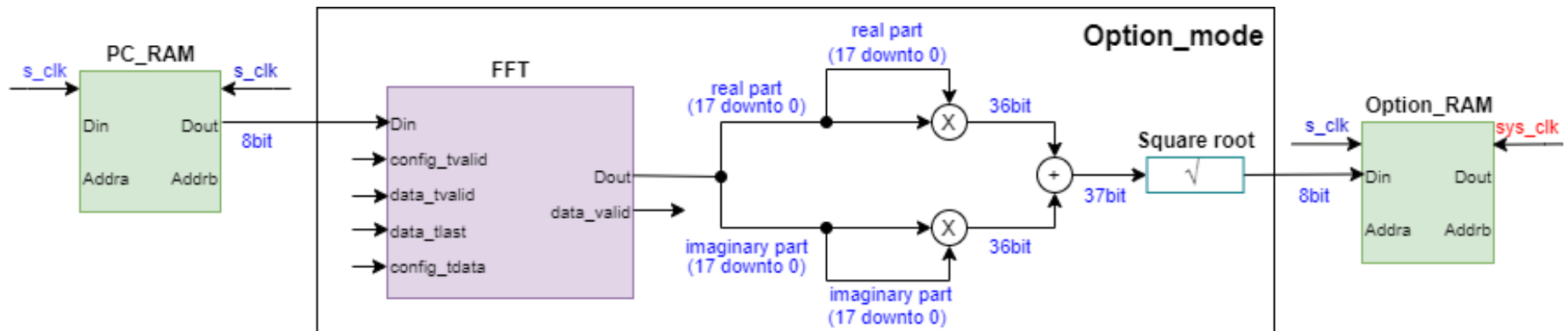
❖ FFT & abs

- 신호의 주파수 스펙트럼 관찰
- FFT 취한 후, $\sqrt{(\text{real part})^2 + (\text{imaginary part})^2}$

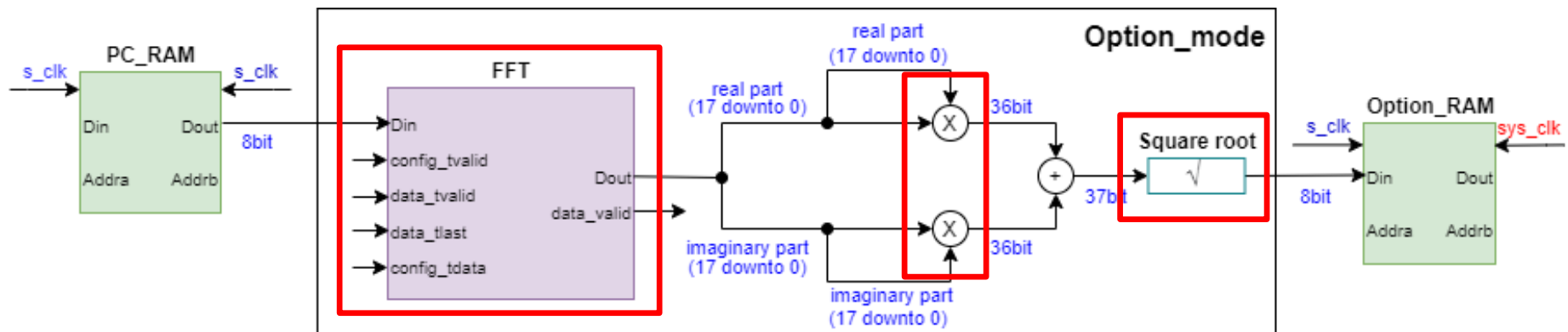


❖ FFT & abs

- 아래와 같은 구조로 연산 진행 (모든 값들은 signed)

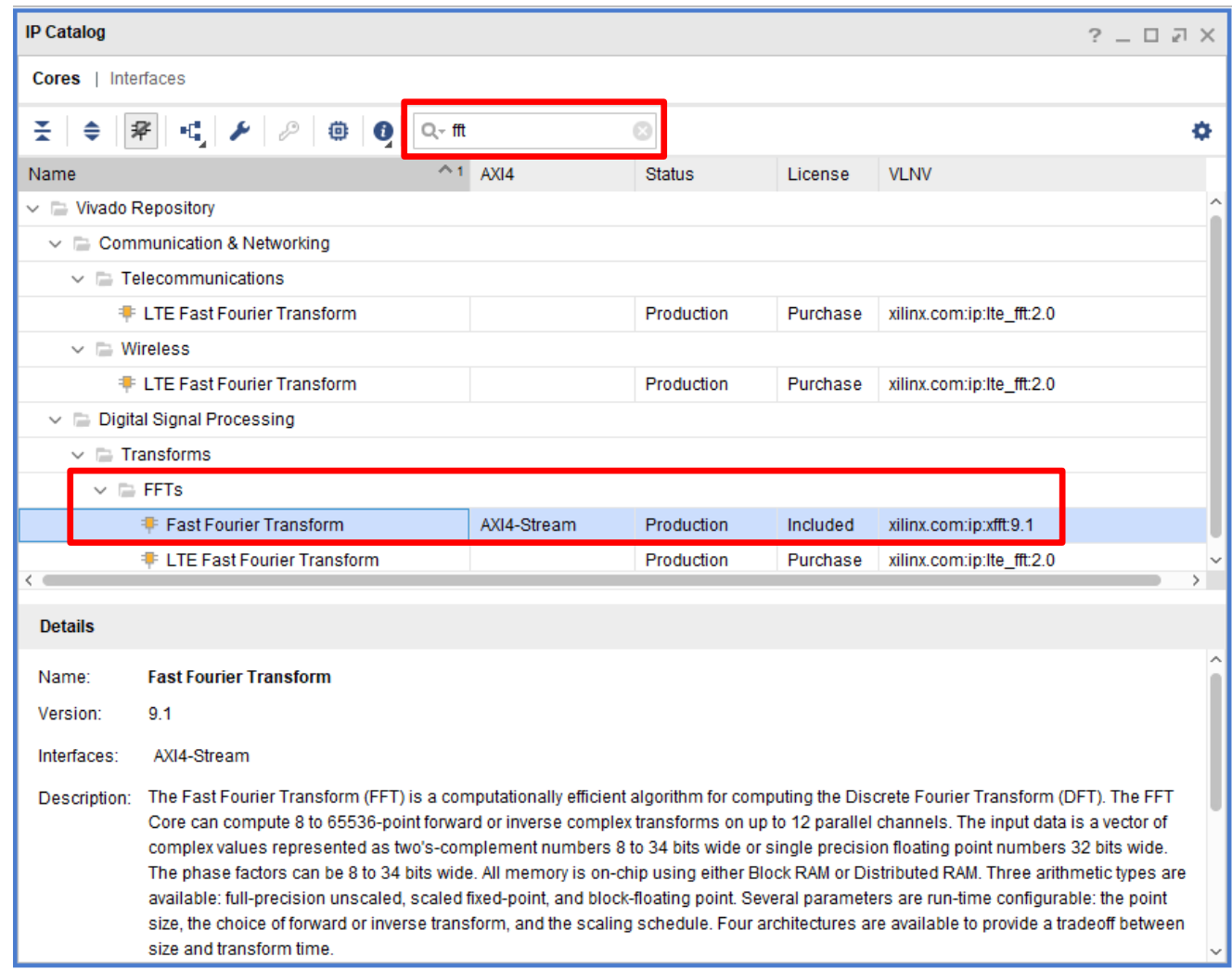


❖ FFT & Multiplier & Cordic(Square root)



❖ FFT

- 'IP Catalog' 클릭 -> 검색창에 'fft' 타이핑 후 'Fast Fourier Transform' 더블 클릭



❖ FFT setting

Customize IP

Fast Fourier Transform (9.1)

Documentation IP Location Switch to Defaults

IP Symbol Implementation Details Latency

☐ Show disabled ports

Component Name FFT

Configuration Implementation Detailed Implementation

Number of Channels 1

Transform Length 512

Architecture Configuration

Target Clock Frequency (MHz) 40 [1 - 1000]

Target Data Throughput (MSPS) 50 [1 - 1000]

Architecture Choice

☐ Automatically Select

☐ Pipelined, Streaming I/O

☐ Radix-4, Burst I/O

☒ Radix-2, Burst I/O

☐ Radix-2 Lite, Burst I/O

☐ Run Time Configurable Transform Length

OK Cancel

Ports:

- S_AXIS_DATA
- S_AXIS_CONFIG
- acclk
- M_AXIS_DATA
- event_frame_started
- event_tlast_unexpected
- event_tlast_missing
- event_data_in_channel_halt

❖ FFT setting

Customize IP

Fast Fourier Transform (9.1)

Documentation IP Location Switch to Defaults

IP Symbol Implementation Details Latency

☐ Show disabled ports

Component Name FFT

Configuration **Implementation** Detailed Implementation

Data Format Fixed Point

Scaling Options Unscaled

Rounding Modes Truncation

Precision Options

Input Data Width 8 Phase Factor Width 8

Control Signals

☐ ACLKEN ☐ ARESETn (active low)

ARESETn must be asserted for a minimum of 2 cycles

Output Ordering Options

Output Ordering Natural Order

☐ Cyclic Prefix Insertion

Optional Output Fields Throttle Scheme

☐ XK_INDEX ☐ OVFO ☐ Non Real Time ☒ Real Time

OK Cancel

IP Symbol Implementation Details Latency

☐ Show disabled ports

M_AXIS_DATA +

+ S_AXIS_DATA event_frame_started

+ S_AXIS_CONFIG event_last_unexpected

- aclk event_last_missing

event_data_in_channel_halt

❖ FFT setting

Customize IP

Fast Fourier Transform (9.1)

Documentation IP Location Switch to Defaults

IP Symbol Implementation Details Latency

☐ Show disabled ports

Component Name FFT

Configuration Implementation **Detailed Implementation**

Memory Options

Data	Phase Factors
<input checked="" type="radio"/> Block RAM	<input checked="" type="radio"/> Block RAM
<input type="radio"/> Distributed RAM	<input type="radio"/> Distributed RAM

Number of stages using Block RAM for Data and Phase Factors 0

Reorder Buffer

☒ Block RAM ☐ Distributed RAM

☐ Optimize Block RAM Count Using Hybrid Memories

Optimize Options

Complex Multipliers Use 3-multiplier structure (resource optimization)

Butterfly Arithmetic Use CLB Logic

OK Cancel

Ports:

- S_AXIS_DATA
- S_AXIS_CONFIG
- ack
- M_AXIS_DATA
- event_frame_started
- event_tlast_unexpected
- event_tlast_missing
- event_data_in_channel_halt

❖ FFT pin configuration (Documentation 참고)

Customize IP

Fast Fourier Transform (9.1)

[Documentation](#) [IP Location](#) [Switch to Defaults](#)

IP Symbol **Implementation Details** Latency

AXI4 Stream Port Structure

S_AXIS_DATA - TDATA

Transaction	Field	Type
0	CHAN_0_XN_IM_0(15:8)	fix8_7
	CHAN_0_XN_RE_0(7:0)	fix8_7
1	CHAN_0_XN_IM_1(15:8)	fix8_7
	CHAN_0_XN_RE_1(7:0)	fix8_7
2	CHAN_0_XN_IM_2(15:8)	fix8_7
	CHAN_0_XN_RE_2(7:0)	fix8_7
3	CHAN_0_XN_IM_3(15:8)	fix8_7
	CHAN_0_XN_RE_3(7:0)	fix8_7
...		
511	CHAN_0_XN_IM_511(15:8)	fix8_7
	CHAN_0_XN_RE_511(7:0)	fix8_7

S_AXIS_CONFIG - TDATA

Transaction	Field	Type
0	FWD_INV_0(0:0)	bit1

M_AXIS_DATA - TDATA

Transaction	Field	Type
0	CHAN_0_XN_IM_0(15:8)	fix18

Component Name: FFT

Configuration | Implementation | Detailed Implementation

Number of Channels: 1

Transform Length: 512

Architecture Configuration

Target Clock Frequency (MHz): 40 [1 - 1000]

Target Data Throughput (MSPS): 50 [1 - 1000]

Architecture Choice

☐ Automatically Select
☐ Pipelined, Streaming I/O
☐ Radix-4, Burst I/O
☒ Radix-2, Burst I/O
☐ Radix-2 Lite, Burst I/O

☐ Run Time Configurable Transform Length

OK Cancel

❖ FFT pin configuration

S_AXIS_DATA - TDATA

Transaction	Field	Type
0	CHAN_0_XN_IM_0(15:8)	fix8_7
	CHAN_0_XN_RE_0(7:0)	fix8_7
1	CHAN_0_XN_IM_1(15:8)	fix8_7
	CHAN_0_XN_RE_1(7:0)	fix8_7
2	CHAN_0_XN_IM_2(15:8)	fix8_7
	CHAN_0_XN_RE_2(7:0)	fix8_7
3	CHAN_0_XN_IM_3(15:8)	fix8_7
	CHAN_0_XN_RE_3(7:0)	fix8_7
...		
511	CHAN_0_XN_IM_511(15:8)	fix8_7
	CHAN_0_XN_RE_511(7:0)	fix8_7

S_AXIS_CONFIG - TDATA

Transaction	Field	Type
0	FWD_INV_0(0:0)	bit1

M_AXIS_DATA - TDATA

Transaction	Field	Type
0	CHAN_0_XN_IM_0(41:24)	fix18_7
	CHAN_0_XN_RE_0(17:0)	fix18_7
1	CHAN_0_XN_IM_1(41:24)	fix18_7
	CHAN_0_XN_RE_1(17:0)	fix18_7
2	CHAN_0_XN_IM_2(41:24)	fix18_7
	CHAN_0_XN_RE_2(17:0)	fix18_7
3	CHAN_0_XN_IM_3(41:24)	fix18_7
	CHAN_0_XN_RE_3(17:0)	fix18_7
...		
511	CHAN_0_XN_IM_511(41:24)	fix18_7
	CHAN_0_XN_RE_511(17:0)	fix18_7

```

ENTITY xfft_0 IS
  PORT (
    aclk : IN STD_LOGIC;
    s_axis_config_tdata : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    s_axis_config_tvalid : IN STD_LOGIC;
    s_axis_config_tready : OUT STD_LOGIC;
    s_axis_data_tdata : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
    s_axis_data_tvalid : IN STD_LOGIC;
    s_axis_data_tready : OUT STD_LOGIC;
    s_axis_data_tlast : IN STD_LOGIC;
    m_axis_data_tdata : OUT STD_LOGIC_VECTOR(47 DOWNTO 0);
    m_axis_data_tvalid : OUT STD_LOGIC;
    m_axis_data_tlast : OUT STD_LOGIC;
    event_frame_started : OUT STD_LOGIC;
    event_tlast_unexpected : OUT STD_LOGIC;
    event_tlast_missing : OUT STD_LOGIC;
    event_data_in_channel_halt : OUT STD_LOGIC
  );
END xfft_0;

```

s_axis_config_tdata : bit 0이 0이면 inverse FFT mode, 1이면 FFT mode

s_axis_config_tvalid : 1->0으로 떨어지면 configuration 입력

s_axis_data_tdata : (15 downto 8) -> imaginary part (0으로 채우기),
(7 downto 0) -> real part (pc ram data) 512 samples 넣어주기

s_axis_data_tvalid : 유효한 입력이 들어갈 때 1

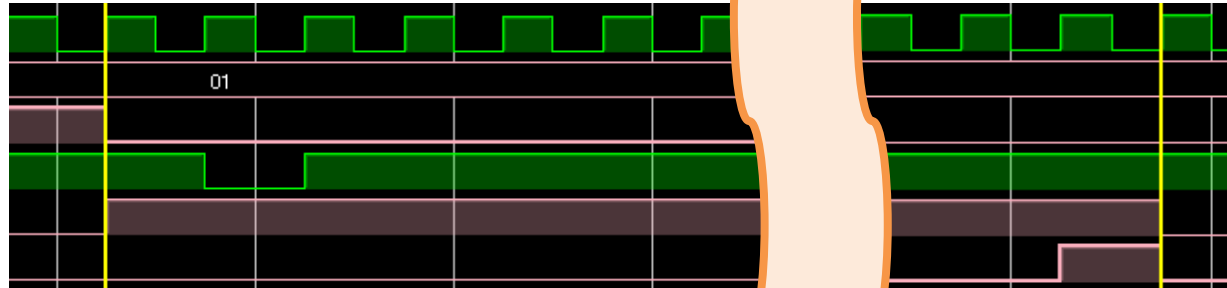
s_axis_data_tlast : 마지막 sample일 때 1

m_axis_data_tdata : (41 downto 24) -> imaginary part, (17 downto 0) -> real part

m_axis_data_tvalid : 유효한 output이 나올 때 1

❖ FFT simulation example

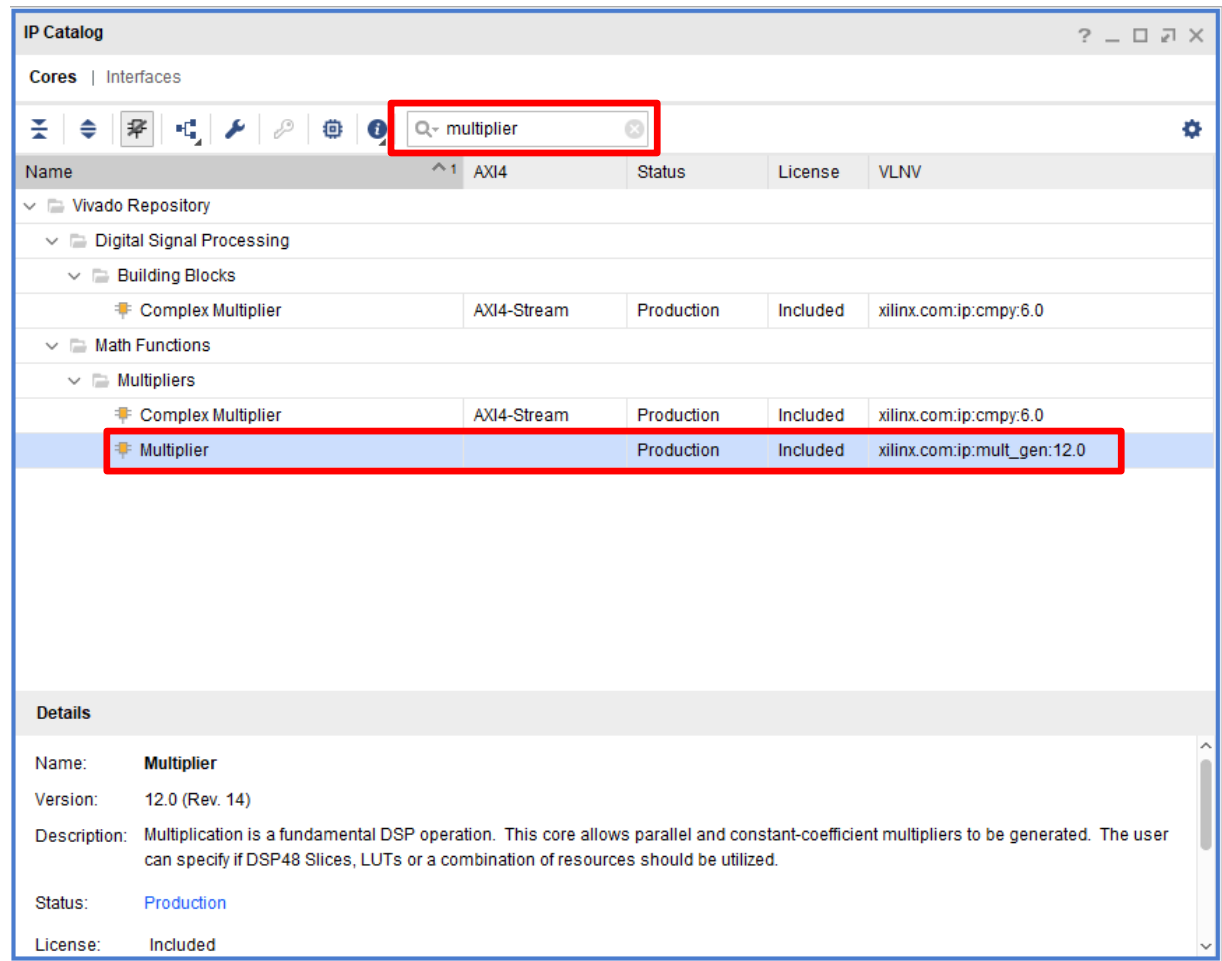
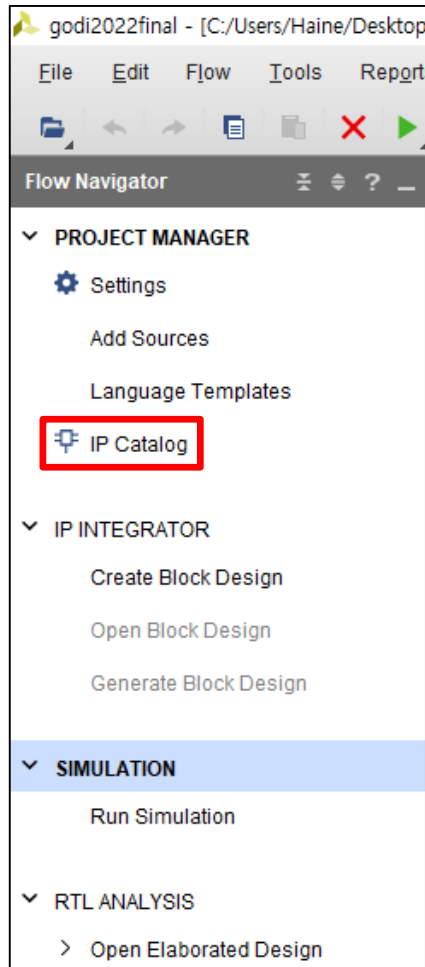
```
🔌 aclk  
> 🔌 s_axis_config_tdata[7:0]  
🔌 s_axis_config_tvalid  
🔌 s_axis_data_tready  
🔌 s_axis_data_tvalid  
🔌 s_axis_data_tlast  
> 🔌 s_axis_data_tdata[15:0]
```



← 512 sample input

❖ Multiplier

- 'PROJECT MANAGER'의 'IP Catalog' 클릭
- 검색창에 'multiplier' 타이핑 후 'Multiplier' 더블 클릭



❖ Multiplier

Re-customize IP

Multiplier (12.0)

Documentation IP Location Switch to Defaults

IP Symbol Information

☐ Show disabled ports

Component Name: mult_gen_0

Basic Output and Control

Multiplier Type

☒ Parallel Multiplier ☐ Constant Coefficient Multiplier

Input Options

$P = A * B$

Data Type: Signed Signed

Width: 18 18

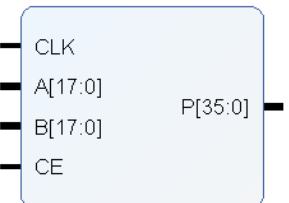
Range: 2..64 Range: 2..64

Multiplier Construction: Use Mults

Optimization Options: Speed Optimized

Area: Optimizes the multiplier for DSP48 slice resources by splitting the multiplication between DSP48 slices and slice logic
Speed: Optimizes the multiplier for performance using as many DSP48 slices as necessary

OK Cancel



The diagram shows a multiplier block with inputs CLK, A[17:0], B[17:0], and CE. The output is P[35:0].

❖ Multiplier : OK & Generate 클릭 후 생성

Re-customize IP

Multiplier (12.0)

Documentation IP Location Switch to Defaults

IP Symbol Information

☐ Show disabled ports

CLK
A[17:0]
B[17:0]
CE

P[35:0]

Component Name: mult_gen_0

Basic **Output and Control**

Output Product Range

☐ Use Custom Output Width

Output MSB: 35 [0 - 127]
Output LSB: 0 [0 - 35]

Output product width (max, min) = (35,0)

☐ Use Symmetric Rounding

Pipelining and Control Signals

Pipeline Stages: 1 Optimum pipeline stages: 3

☒ Clock Enable ☐ Synchronous Clear

Synchronous Controls and Clock Enable(CE) Priority: SCLR Overrides CE

OK Cancel

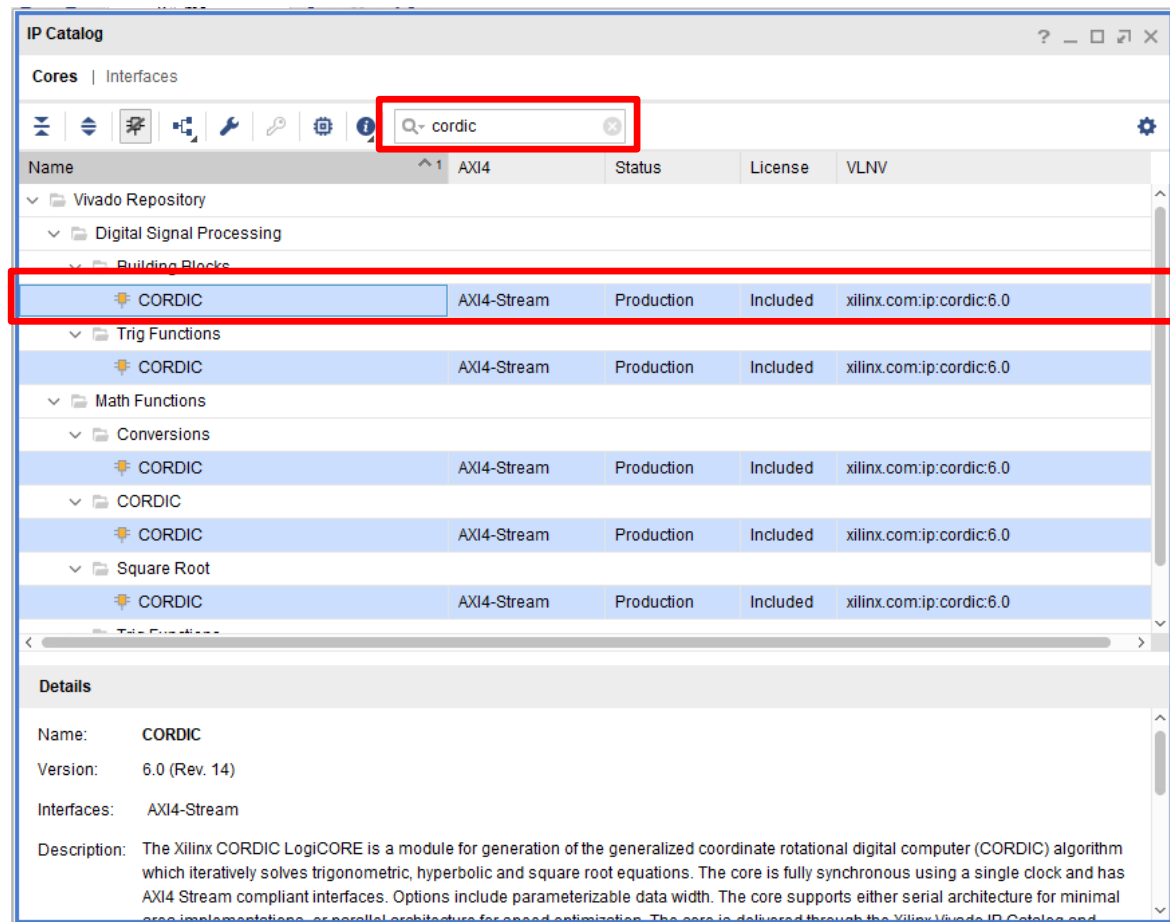
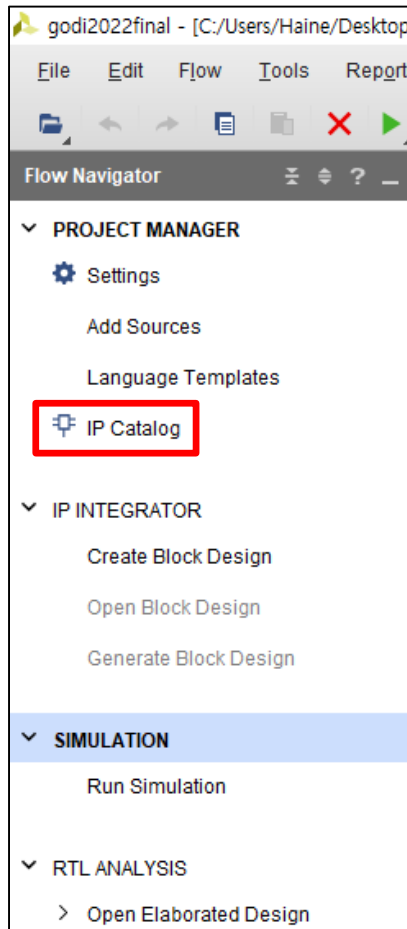
❖ Multiplier

- FFT 결과로 나온 imaginary part와 real part를 각각 제공해야 함 : multiplier 2개 필요
- Multiplier component 선언 후 port mapping 2번

```
ENTITY mult_gen_0 IS
  PORT (
    CLK : IN STD_LOGIC;
    A : IN STD_LOGIC_VECTOR(17 DOWNT0 0);
    B : IN STD_LOGIC_VECTOR(17 DOWNT0 0);
    CE : IN STD_LOGIC;
    P : OUT STD_LOGIC_VECTOR(35 DOWNT0 0)
  );
END mult_gen_0;
```

❖ CORDIC – Square Root

- 'PROJECT MANAGER'의 'IP Catalog' 클릭
- 검색창에 'cordic' 타이핑 후 'CORDIC' 더블 클릭



❖ CORDIC – Square Root

Re-customize IP

CORDIC (6.0)

Documentation IP Location Switch to Defaults

IP Symbol Implementation Details

☐ Show disabled ports

Component Name: cordic_0

Configuration Options AXI4 Stream Options

Configuration Parameters

Functional Selection	Square Root
Architectural Configuration	Parallel
Pipelining Mode	Optimal
Data Format	UnsignedInteger
Phase Format	Radians

Input/Output Options

Input Width	37	[8 - 48]
Output Width	19	[5 - 48]
Round Mode	Truncate	

Advanced Configuration Parameters

Iterations	0	[0 - 48]
Precision	0	[0 - 48]
<input type="checkbox"/> Coarse Rotation		
Compensation Scaling	No Scale Compensation	

OK Cancel

IP Symbol: S_AXIS_CARTESIAN, M_AXIS_DOUT, clk

❖ CORDIC – Square Root : AXI4 Stream Options에서 따로 건드릴 건 없음

Re-customize IP

CORDIC (6.0)

Documentation IP Location Switch to Defaults

IP Symbol Implementation Details

Implementation Details

Latency	10
BRAM	N/A
XtremeDSP	N/A

AXI4-Stream Port Structure

S_AXIS_CARTESIAN - TDATA

Transaction	Field	Type
0	REAL(36:0)	uint37

M_AXIS_DOUT - TDATA

Transaction	Field	Type
0	REAL(18:0)	uint19

Component Name: cordic_0

Configuration Options: AXI4 Stream Options

Cartesian Channel Options

☐ Has TLAST

☐ Has TUSER

TUSER Width: 1 [1 - 256]

Phase Channel Options

☐ Has TLAST

☐ Has TUSER

TUSER Width: 1 [1 - 256]

Flow Control: NonBlocking

Optimize Goal: Performance

☐ Output has TREADY

Output TLAST Behavior: Null

Optional Pins

☐ ACLKEN

☐ ARESETN

OK Cancel

❖ Cordic – Square Root : Implementation Details에서 Latency 확인

Re-customize IP

CORDIC (6.0)

Documentation IP Location Switch to Defaults

IP Symbol **Implementation Details**

Implementation Details

Latency	10	10 clock latency
BRAM	N/A	
XtremeDSP	N/A	

AXI4-Stream Port Structure

S_AXIS_CARTESIAN - TDATA

Transaction	Field	Type
0	REAL(36:0)	uint37

M_AXIS_DOUT - TDATA

Transaction	Field	Type
0	REAL(18:0)	uint19

s_axis_cartesian_tvalid : 유효한 input이 들어갈 때 1

s_axis_cartesian_tdata : $\text{input } (real\ part)^2 + (imaginary\ part)^2$

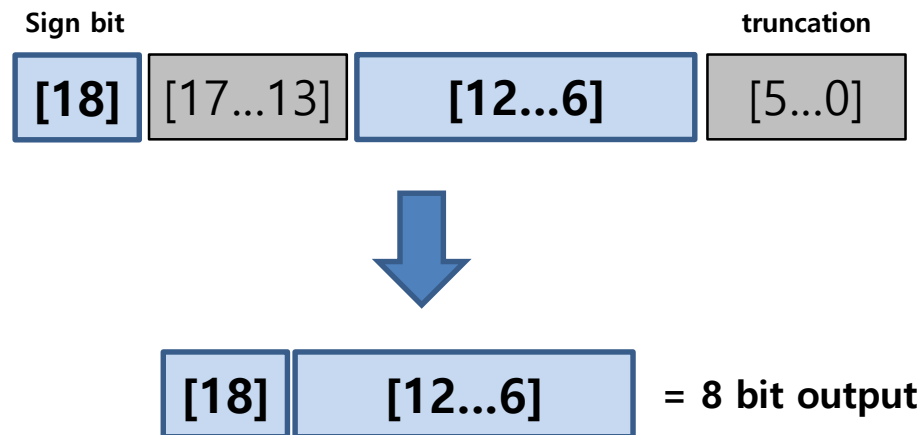
m_axis_dout_tvalid : 유효한 output이 나올 때 1

m_axis_dout_tdata : $\text{output } \sqrt{(real\ part)^2 + (imaginary\ part)^2}$

```
ENTITY cordic_0 IS
PORT (
    aclk : IN STD_LOGIC;
    s_axis_cartesian_tvalid : IN STD_LOGIC;
    s_axis_cartesian_tdata : IN STD_LOGIC_VECTOR(39 DOWNTO 0);
    m_axis_dout_tvalid : OUT STD_LOGIC;
    m_axis_dout_tdata : OUT STD_LOGIC_VECTOR(23 DOWNTO 0)
);
END cordic_0;
```

❖ Bit Truncation

- FFT, Multiplier, Cordic(Root)를 거쳐 나온 출력은 총 19bit (18 downto 0)
- USB로 통신하는 m_data는 8bit (7 downto 0)
- 파형을 헤치지 않는 선에서 **bit truncation** 필요



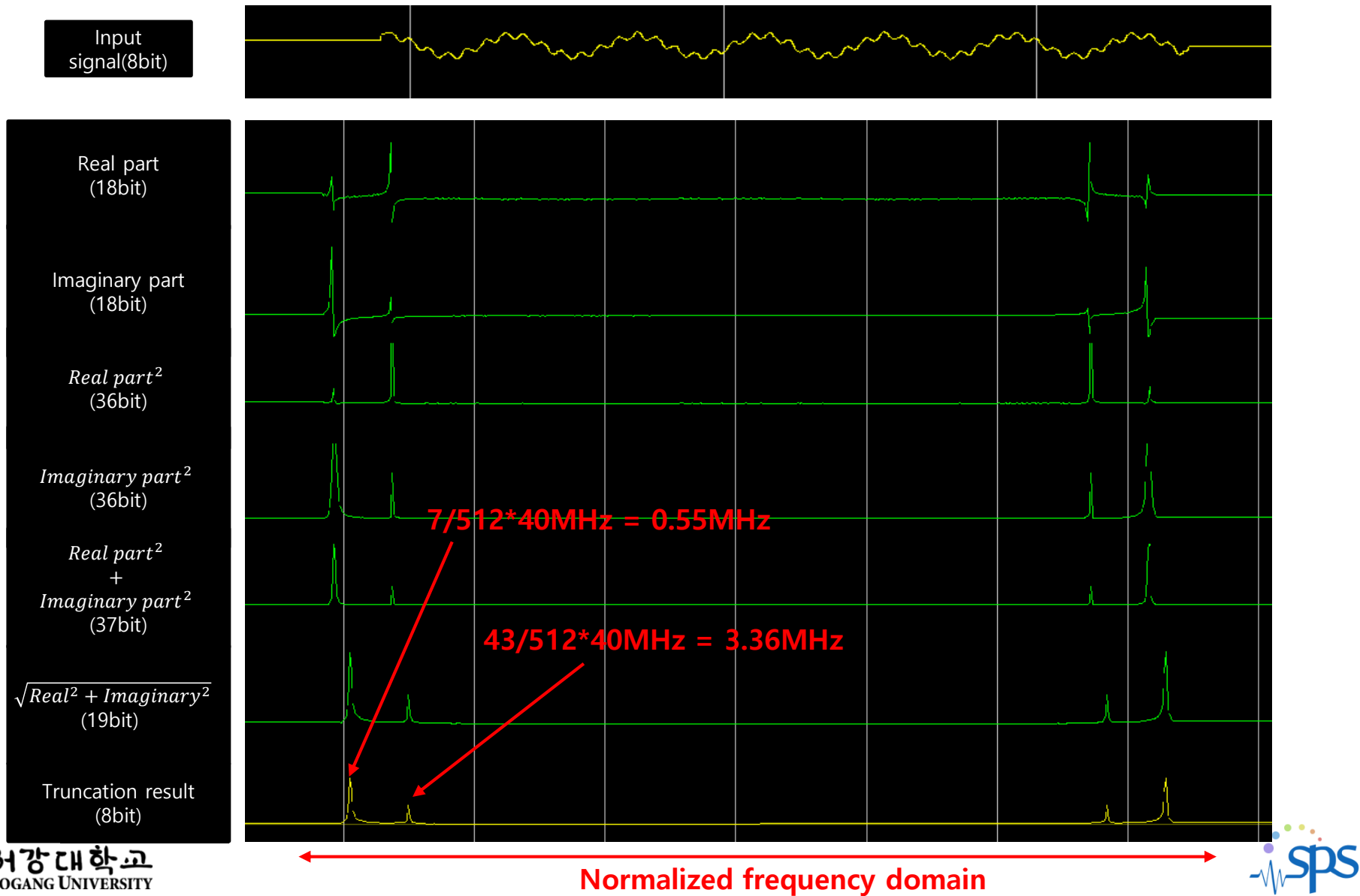
Sign extension

```

        Sign bit                                Data
m_yout <= root_result(18) & root_result(12 downto 6);

```

❖ Simulation example



Test Bench

❖ Data read/write 를 위한 CMD_RD, CMD_WR 함수가 있음

```

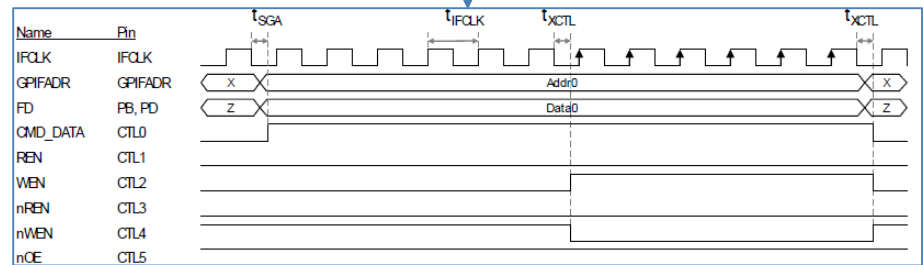
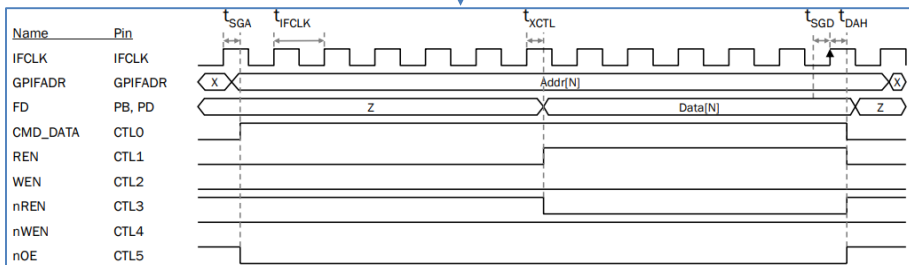
--CMD_RD--
procedure CMD_RD (
    Addr          : in    std_logic_vector(8 downto 0);
    Address_tmp   : out   std_logic_vector(8 downto 0);
    Data_tmp      : inout std_logic_vector(7 downto 0);
    CMD_DATA_tmp  : out   std_logic;
    WEN_tmp       : out   std_logic;
    REN_tmp       : out   std_logic;
    n_OE          : out   std_logic
) is
begin
    Address_tmp      <= Addr;
    CMD_DATA_tmp     <= '1';
    WEN_tmp          <= '0';
    REN_tmp          <= '0';
    n_OE             <= '0';
    Data_tmp         <= "ZZZZZZZZ";
    wait for m_clk_period*5;
    REN_tmp          <= '1';
    wait for m_clk_period*5;
    Address_tmp      <= (others=>'Z');
    CMD_DATA_tmp     <= '0';
    REN_tmp          <= '0';
    n_OE             <= '1';
    wait for 10 ns;
end CMD_RD;
    
```

```

--CMD_WR--
procedure CMD_WR (
    Addr          : in    std_logic_vector(8 downto 0);
    Data_in       : in    std_logic_vector(7 downto 0);
    Address_tmp   : out   std_logic_vector(8 downto 0);
    Data_tmp      : out   std_logic_vector(7 downto 0);
    CMD_DATA_tmp  : out   std_logic;
    WEN_tmp       : out   std_logic;
    REN_tmp       : out   std_logic;
    n_OE          : out   std_logic
) is
begin
    Address_tmp      <= Addr;
    Data_tmp         <= Data_in;
    CMD_DATA_tmp     <= '1';
    WEN_tmp          <= '0';
    REN_tmp          <= '0';
    n_OE             <= '1';
    wait for m_clk_period*5;

    WEN_tmp          <= '1';
    wait for m_clk_period*5;

    Address_tmp      <= (others=>'Z');
    Data_tmp         <= (others=>'Z');
    CMD_DATA_tmp     <= '0';
    WEN_tmp          <= '0';
    wait for 10 ns;
end CMD_WR;
    
```



❖ Process

▪ Clock process

```
-- Clock process definitions
m_clk_process : process
begin
    m_clk <= '0';
    wait for m_clk_period/2;
    m_clk <= '1';
    wait for m_clk_period/2;
end process;

ADC_input : process
begin
    m_ADC_data <= m_ADC_data + x"04";
    wait for m_clk_period*13; -- 자유롭게 변경 가능
end process;
```


❖ Process

- Option mode Input data process
 - Option Block의 m_en = '1'일 때, 순차적으로 data가 나옴
 - 총 512개의 data가 들어있음

```

Option_input_data : process(s_dat_clk)
file               : text is in "Sample_Input_1.dat"; --원하는 dat파일 이름을 적어주세요 fs=40MHz
variable linein1   : line;
variable inputtmp1 : integer;
begin
  if rising_edge(s_dat_clk) then
    if s_dat_en = '1' then
      if not(endfile(filein1)) then
        readline(filein1, linein1);
        read(linein1, inputtmp1);
        option_data <= conv_std_logic_vector(inputtmp1,8);
      else
        assert false
        report "End of File_I!"
        severity note;
      end if;
    else
      option_data <= option_data;
    end if;
  end if;
end process;

```

❖ Process

▪ Simulation process1

```

-- Stimulus process
stim_proc: process
begin
    -- hold reset state for 100 ns.
    wait for 100 ns;
    m_reset_b <= '1';
    wait for 10 us;

    -- 8254 setting (m_clk를 4분주해서 div_clk을 만들기 위한 과정)
    CMD_WR('1' & x"13", "00110110", m_address, m_data, m_cmd_data, m_wen, m_ren, m_OE_b);
    wait for 10 us;
    CMD_WR('1' & x"10", "00000100", m_address, m_data, m_cmd_data, m_wen, m_ren, m_OE_b); -- LSB 04
    wait for 10 us;
    CMD_WR('1' & x"10", "00000000", m_address, m_data, m_cmd_data, m_wen, m_ren, m_OE_b); -- MSB 00
    wait for 10 us;

    ----PC Write
    for i in 0 to 10 loop
        CMD_WR('1' & x"30", conv_std_logic_vector(i, 8), m_address, m_data, m_cmd_data, m_wen, m_ren, m_OE_b);
        wait for 1 us;
    end loop;

    -- PC read mode : 8254reset => 8254 1분주 => PC read mode
    CMD_WR('1' & x"21", "00000000", m_address, m_data, m_cmd_data, m_wen, m_ren, m_OE_b);
    wait for 10 us;
    CMD_WR('1' & x"13", "00110110", m_address, m_data, m_cmd_data, m_wen, m_ren, m_OE_b);
    wait for 10 us;
    CMD_WR('1' & x"10", "00000001", m_address, m_data, m_cmd_data, m_wen, m_ren, m_OE_b); -- LSB 01
    wait for 10 us;
    CMD_WR('1' & x"10", "00000000", m_address, m_data, m_cmd_data, m_wen, m_ren, m_OE_b); -- MSB 00
    wait for 10 us;
    for i in 0 to 10 loop
        CMD_RD('1' & x"30", m_address, m_data, m_cmd_data, m_wen, m_ren, m_OE_b);
        wait for 1 us;
    end loop;
    wait for 10 us;

```

❖ Process

▪ Simulation process2

```

-- DA mode : 8254reset => 8254 n 분주 => DA mode
CMD_WR('1' & x"21","00000000",m_address,m_data,m_cmd_data,m_wen,m_ren,m_OE_b);
wait for 10 us;
CMD_WR('1' & x"13","00110110",m_address,m_data,m_cmd_data,m_wen,m_ren,m_OE_b);
wait for 10 us;
CMD_WR('1' & x"10","00000100",m_address,m_data,m_cmd_data,m_wen,m_ren,m_OE_b); -- LSB 04
wait for 10 us;
CMD_WR('1' & x"10","00000000",m_address,m_data,m_cmd_data,m_wen,m_ren,m_OE_b); -- MSB 00
wait for 10 us;
CMD_RD('1' & x"40",m_address,m_data,m_cmd_data,m_wen,m_ren,m_OE_b);
wait for 10 us;

-- DA stop mode
CMD_RD('1' & x"41",m_address,m_data,m_cmd_data,m_wen,m_ren,m_OE_b);
wait for 10 us;
...
-- AD mode : 8254reset => 8254 n분주 => AD mode
CMD_WR('1' & x"21","00000000",m_address,m_data,m_cmd_data,m_wen,m_ren,m_OE_b);
wait for 10 us;
CMD_WR('1' & x"13","00110110",m_address,m_data,m_cmd_data,m_wen,m_ren,m_OE_b);
wait for 10 us;
CMD_WR('1' & x"10","00000100",m_address,m_data,m_cmd_data,m_wen,m_ren,m_OE_b); -- LSB 04
wait for 10 us;
CMD_WR('1' & x"10","00000000",m_address,m_data,m_cmd_data,m_wen,m_ren,m_OE_b); -- MSB 00
wait for 10 us;
CMD_WR('1' & x"50","00001011",m_address,m_data,m_cmd_data,m_wen,m_ren,m_OE_b);
wait for 10 us;

```

❖ Process

▪ Simulation process3

```

-- PC read mode : 8254reset => 8254 1분주 => PC read mode
CMD_WR('1' & x"21", "00000000", m_address, m_data, m_cmd_data, m_wen, m_ren, m_OE_b);
wait for 10 us;
CMD_WR('1' & x"13", "00110110", m_address, m_data, m_cmd_data, m_wen, m_ren, m_OE_b);
wait for 10 us;
CMD_WR('1' & x"10", "00000001", m_address, m_data, m_cmd_data, m_wen, m_ren, m_OE_b); -- LSB 01
wait for 10 us;
CMD_WR('1' & x"10", "00000000", m_address, m_data, m_cmd_data, m_wen, m_ren, m_OE_b); -- MSB 00
wait for 10 us;
for i in 0 to 10 loop
CMD_RD('1' & x"30", m_address, m_data, m_cmd_data, m_wen, m_ren, m_OE_b); -- PC RAM에 10개 읽기
wait for 1 us;
end loop;
wait for 10 us;

-- ADR mode : 8254reset => 8254 1분주 => ADR mode
CMD_WR('1' & x"21", "00000000", m_address, m_data, m_cmd_data, m_wen, m_ren, m_OE_b);
wait for 10 us;
CMD_WR('1' & x"13", "00110110", m_address, m_data, m_cmd_data, m_wen, m_ren, m_OE_b);
wait for 10 us;
CMD_WR('1' & x"10", "00000001", m_address, m_data, m_cmd_data, m_wen, m_ren, m_OE_b); -- LSB 01
wait for 10 us;
CMD_WR('1' & x"10", "00000000", m_address, m_data, m_cmd_data, m_wen, m_ren, m_OE_b); -- MSB 00
wait for 10 us;
for i in 0 to 10 loop
CMD_RD('1' & x"51", m_address, m_data, m_cmd_data, m_wen, m_ren, m_OE_b);
wait for 1 us;
end loop;
wait for 10 us;

```

❖ Process

▪ Simulation process4

```

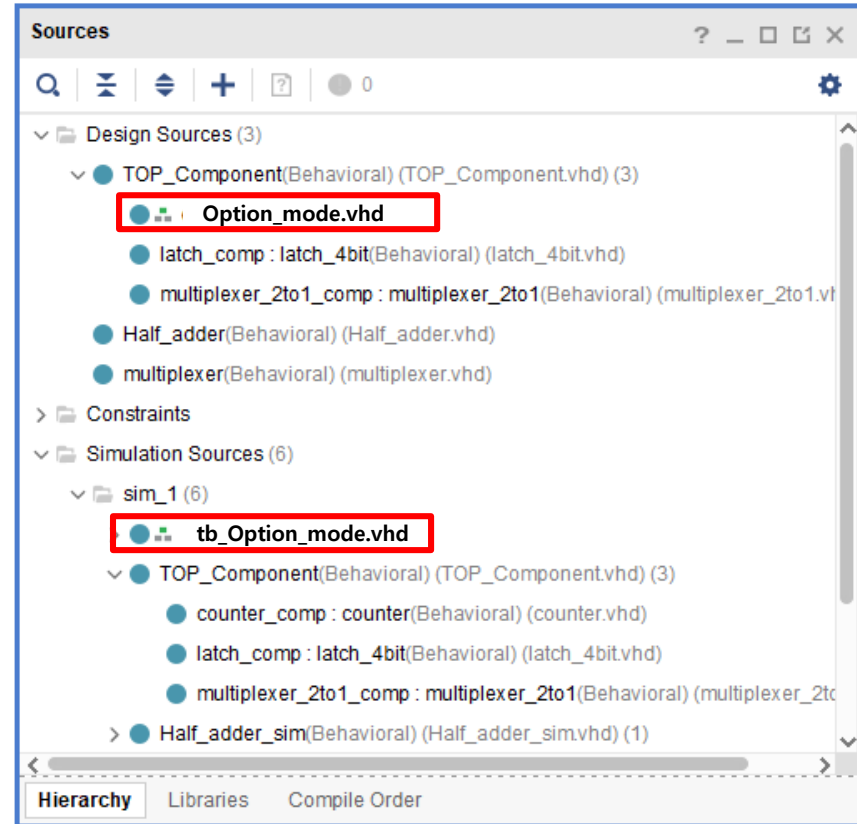
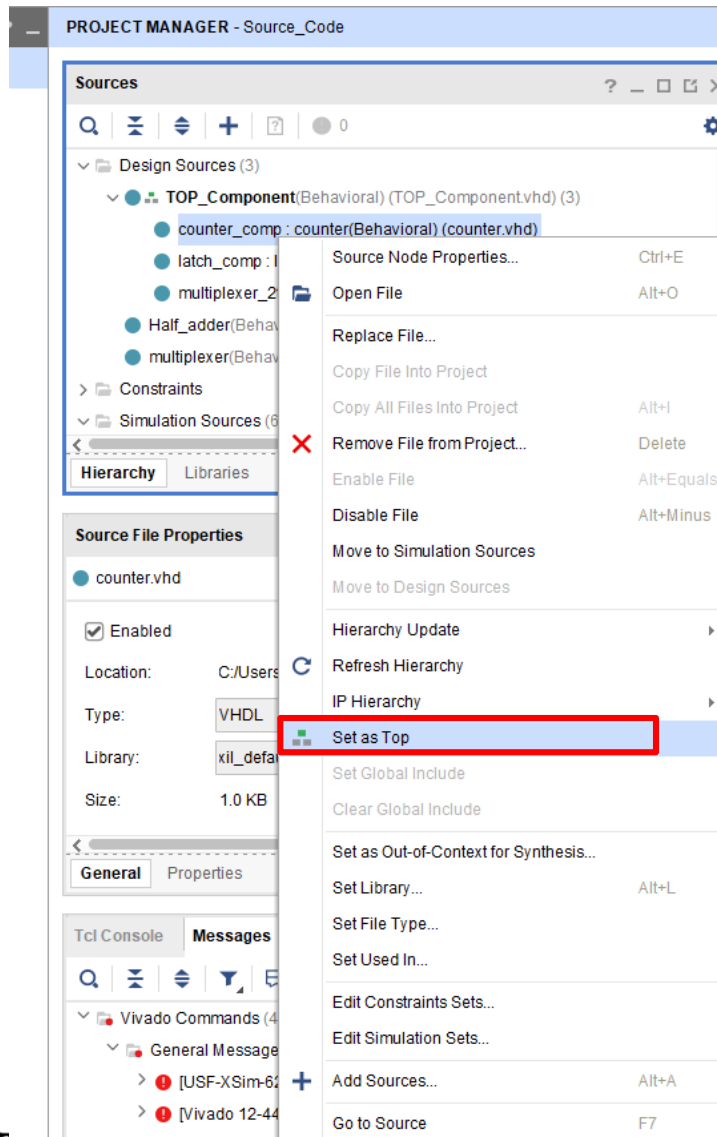
-----Option mode sequence-----
--PC Write mode : PC RAM0 "Sample_Input_1.dat" Write
s_dat_en <= '1';
  for i in 0 to 511 loop
    s_dat_clk <= '1';
    wait for 1 us;
    CMD_WR('1' & x"30",option_data,m_address,m_data,m_cmd_data,m_wen,m_ren,m_OE_b);
    s_dat_clk <= '0';
    wait for 1 us;
  end loop;
wait for 10 us;

--Option mode(step1)
CMD_WR('1' & x"60","00000000",m_address,m_data,m_cmd_data,m_wen,m_ren,m_OE_b);
wait for 300 us;

--Option mode(step2)
for i in 0 to 511 loop
  CMD_RD('1' & x"61",m_address,m_data,m_cmd_data,m_wen,m_ren,m_OE_b); -- OPTION RAM값 읽기
  wait for 1 us;
end loop;

```

❖ tb_Option_mode.vhd (component simulation code)

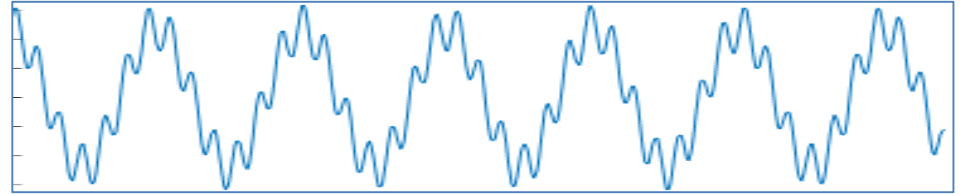


- Design source, Sim에서 Option_mode.vhd , tb_Option_mode.vhd를 각각 Set as Top으로 설정
- Component에 대한 Simulation

❖ Sample_Input.dat file Example

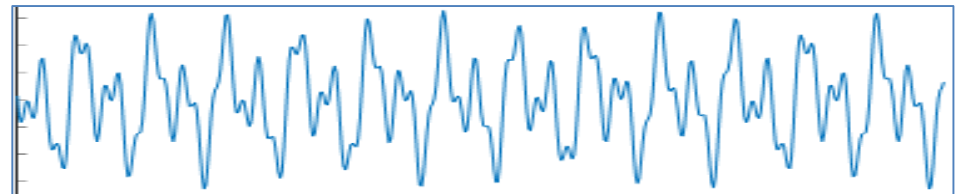
▪ Sample_Input_1.dat

- $3\cos\left(\frac{2\pi f_0}{f_s} * n\right) + \sin\left(\frac{2\pi f_1}{f_s} * n\right)$
- $f_0 = 0.5\text{MHz}$, $f_1 = 3.3\text{MHz}$, $f_s = 40\text{MHz}$
- $n = [0 \ 1 \ 2 \ \dots \ 510 \ 511]$ (총 512개 data)



▪ Sample_Input_2.dat

- $3\cos\left(\frac{2\pi f_2}{f_s} * n\right) - 4\sin\left(\frac{2\pi f_3}{f_s} * n\right) - 2\sin\left(\frac{2\pi f_4}{f_s} * n\right)$
- $f_2 = 1\text{MHz}$, $f_3 = 2\text{MHz}$, $f_4 = 4.7\text{MHz}$
- $n = [0 \ 1 \ 2 \ \dots \ 510 \ 511]$ (총 512개 data)



❖ 조편성

조	학번	이름	학번	이름
1	20181510	나일영	20181482	강원호
2	20181504	김태윤	20181581	홍준화
3	20181521	박지원	20171553	안재석
4	20171560	이경원	20171582	정진원
5	20171517	김준석	20171557	오준석
6	20171497	국승현	20171567	이용
7	20171591	최성우	20171510	김윤수
8	20161469	김태건	20161519	이우열
9	20191509	유민주	20191523	이채원
10	20171548	신민재	20171536	박지호
11	20171540	박창우	20181585	황용하
12	20171556	염하겸	20171564	이세직
13	20171594	추승엽	20171527	노현석
14	20171530	박건희	20171543	백승우
15	20191459	김민주	20171535	박준희
16	20171522	김혁진	20171552	신지철
17	20171549	신승재	20171579	임채진

조	학번	이름	학번	이름	학번	이름
18	20171492	고기윤	20171506	김성영		
19	20171541	박태준	20171585	정현우		
20	20171524	남상윤	20171490	강희범		
21	20191451	강승민	20191455	고제우		
22	20171596	허준석	20171569	이장원		
23	20171544	손재균	20171512	김정현		
24	20171531	박경원	20171519	김진우		
25	20161537	정재영	20161514	이민우		
26	20171547	송현우	20171537	박진형		
27	20162075	김동이	20181501	김준원		
28	20151477	이종민	20161513	이명환		
29	20161997	양주열	20161270	곽지균		
30	20171525	남우재	20161533	장철민		
31	20171518	김지홍	20171561	이계명		
32	20161481	박종호	20181538	유혁준		
33	20171563	이지한	20201470	박민재		
34	20192126	박보람	20161522	이종빈	20182193	심호섭

❖ 각 조 질문 멘토 및 오픈채팅방

- 1~5조 : 조민성
– <https://open.kakao.com/o/gxgw5see>
 - 6~10조 : 이예원
– <https://open.kakao.com/o/g1flbtee>
 - 11~17조 : 김준성
– <https://open.kakao.com/o/gMmu0see>
 - 18~22조 : 김유진
– <https://open.kakao.com/o/gLcY6see>
 - 23조~34조 : 이해은
– <https://open.kakao.com/o/g2rS6see>
- ✓ 각 조는 해당 조교에게만 질문하며, 자신이 해당하는 조의 오픈채팅방에서 질문
 ✓ 추가로, 옵션모드 관련 질문은 저(이해은)에게만 질문


❖ 공지 사항

- 5/25(수) 부터 통과될 때까지 매일 진행 (시간 및 장소 추후 공지)
- 시연은 하루에 1번
- Option Mode를 포함한 모든 모드가 통과해야 통과
- 1팀 당 약 10분의 시간이 소요되므로 각 조의 시연시간 준수
 - 추후 사이버캠퍼스에 공지에정 (선착순 댓글로 시간 신청)
- 각 조에 해당하는 멘토 또는 조교에게 질문
- 준비 사항
 - 시연할 Vivado 프로젝트 파일이 담긴 USB (또는 노트북)
 - Bit파일 생성시 에러나는 경우가 있으므로 미리 Bit파일 생성해보기
 - HDMI signal 관련 Critical error는 무시해도 상관없음
 - 빠른 진행을 위해 적어도 시연시간 10분 전에는 시연 장소에 있기

❖ Software UI

MSI PCFG UI

Graph Data


MSi
 Medical Solution Institute

Clear

USB Open

Configuration

CLOCK

INTERNAL for 8254 setting

EXTERNA general

2022_generation

Please Connect UsbModule

8254 Reset Software Reset

Step1 : Clock Setting

8 start

Step2 : PC mode

Line write read

Step3 : DA mode

da_start

wen ren start

da_stop

wen ren stop

Step4 : AD /ADR mode

ad

wen ren AD mode ADR mode

value 200

Step5 : Envelope Detection

Step1 Step2

ADDRESS DATA

CMD Write Confirm

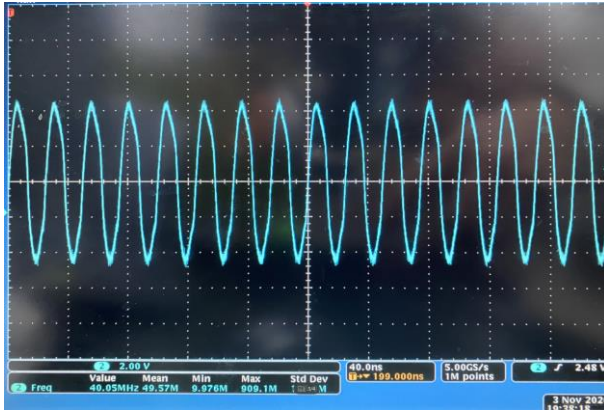
CMD Read Confirm

Burst Write Confirm

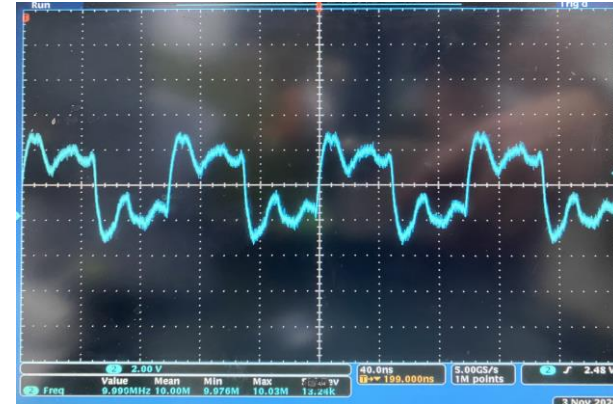
Burst Read Confirm

123H 120H 120H

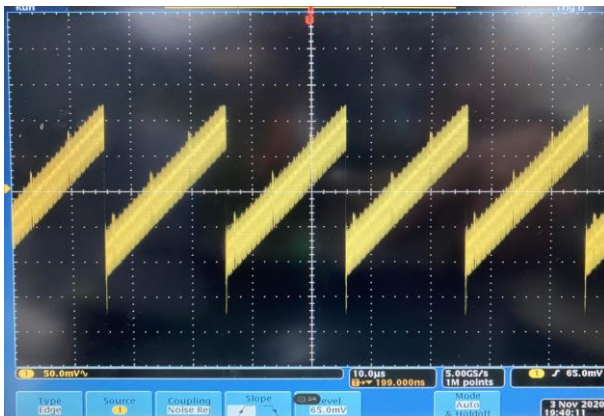
❖ Oscilloscope 파형 (clock & DA mode)



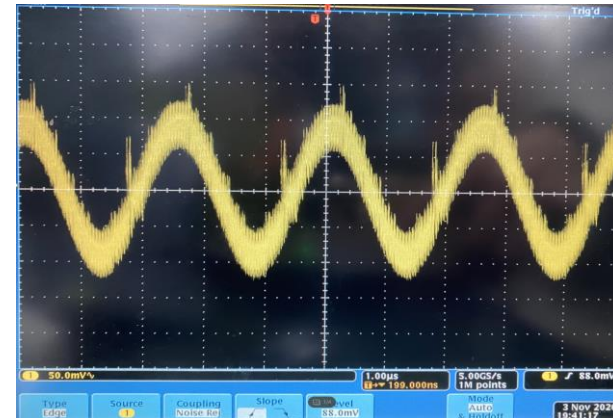
< s_clk >



< sys_clk (8분주) >



< line (DA mode) >



< cos (DA mode) >

Thank you