

SoC 를 위한 *Peripheral* 설계

Reference : MicroBlaze.v16 [IHIL]

2024-06-20

 Kim.S.W



대한상공회의소
인력개발원 @IHIL

digilent.com

 XILINX

1/80

Logic Interface Implementation *[Review and Next Flow]*

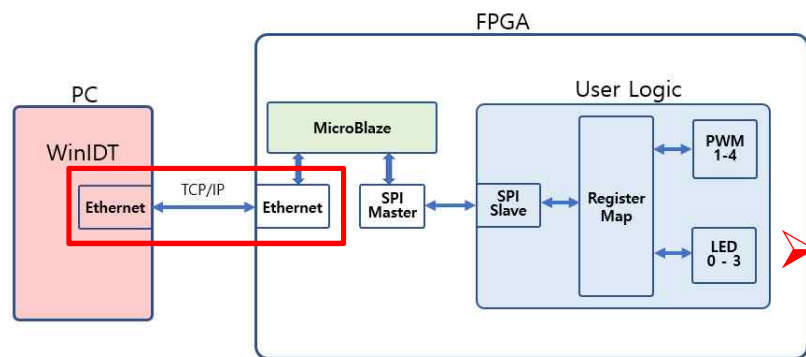
- *User Logic Interface Implementation*
- *Utilize LightWeight IP (lwIP)*
- *Block Memory Interface Implementation*
- *W5500 Interface Implementation*

Logic Interface Implementation

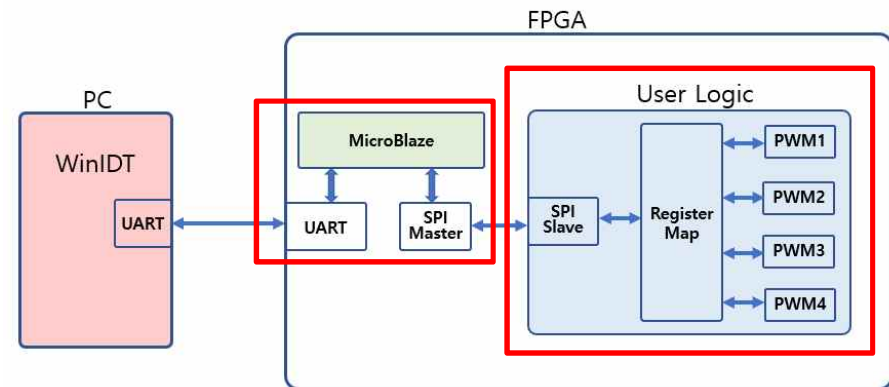
blaze_w5500_Exam *.xpr

[Review and Next Flow]

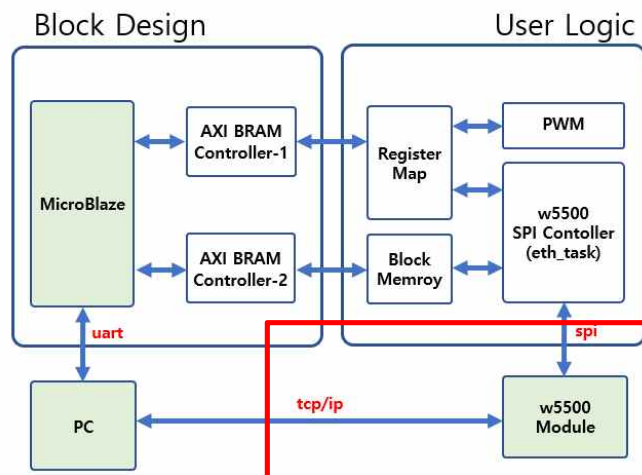
➤ User Logic Interface Implementation



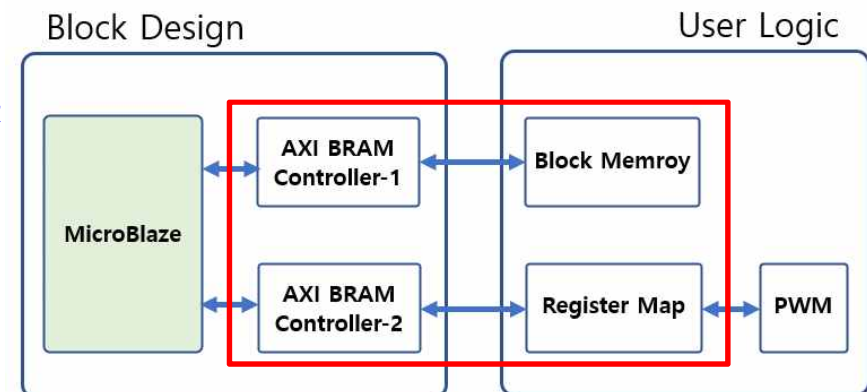
Utilize LightWeight IP (lwIP)



➤ Block Memory Interface Implementation



W5500 Interface Implementation



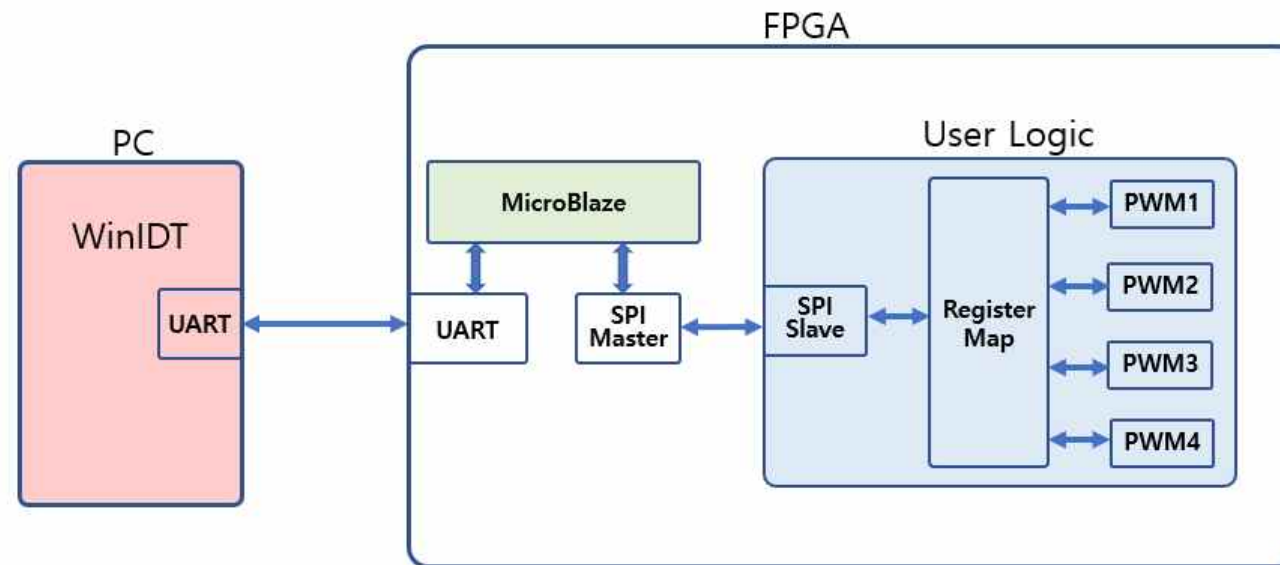
User Logic Interface Implementation

User Logic Interface Implementation

blaze_w5500_Exam *.xpr

➤ User Logic Interface Implementation 실습

- 사용자 Logic 을 추가하고, MicroBlaze와의 인터페이스를 구현
- 인터페이스를 구현하기 위한 여러가지 방법 ➔ UART, I2C, Parallel, SPI 등
- SPI 인터페이스를 사용 ➔ SPI를 사용하는 이유는 연결되는 선이 4개로 적고, I2C에 비해서 데이터를 주고받는 속도가 빠르고 구현하는데 어렵지 않다는 것
- 사용자 Logic은 간단하게 구성할 수 있는 PWM Controller를 구현



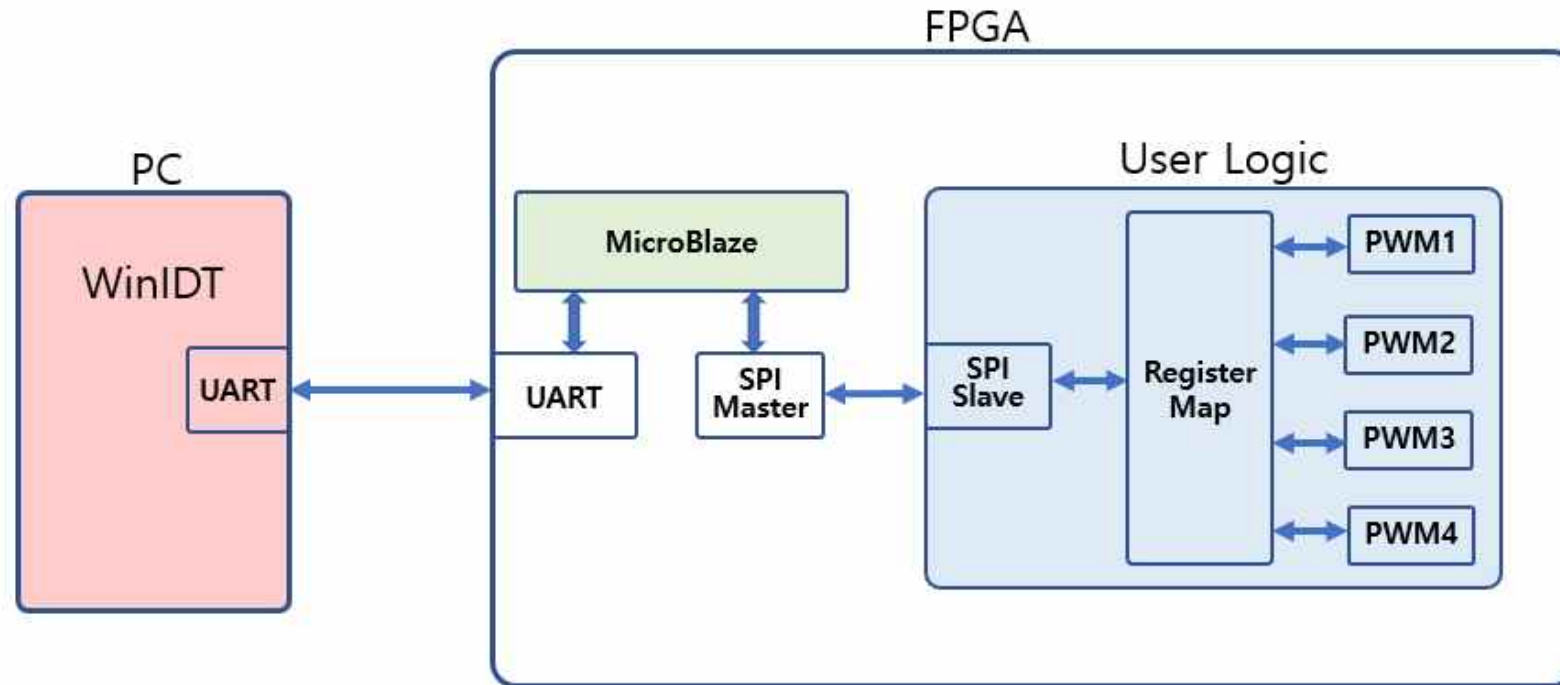
[그림] System Block

User Logic Interface Implementation

blaze_w5500_Exam *.xpr

➤ User Logic Interface Implementation 실습

- PC의 Application Program(ComPortMaster, PuTTY , WinIDT 등)에서 User Logic (PWM)을 제어하기 위하여 UART로 데이터를 전송
- MicroBlaze는 명령어를 수신해서 SPI Master로 Register Read/Write 명령어를 전송
- User Logic의 SPI Slave에서 Register 값을 Read/Write ➔ 이러한 기본 구조를 가지고 있으면 언제든지 User Logic에 필요한 부분들을 추가하여 사용 가능

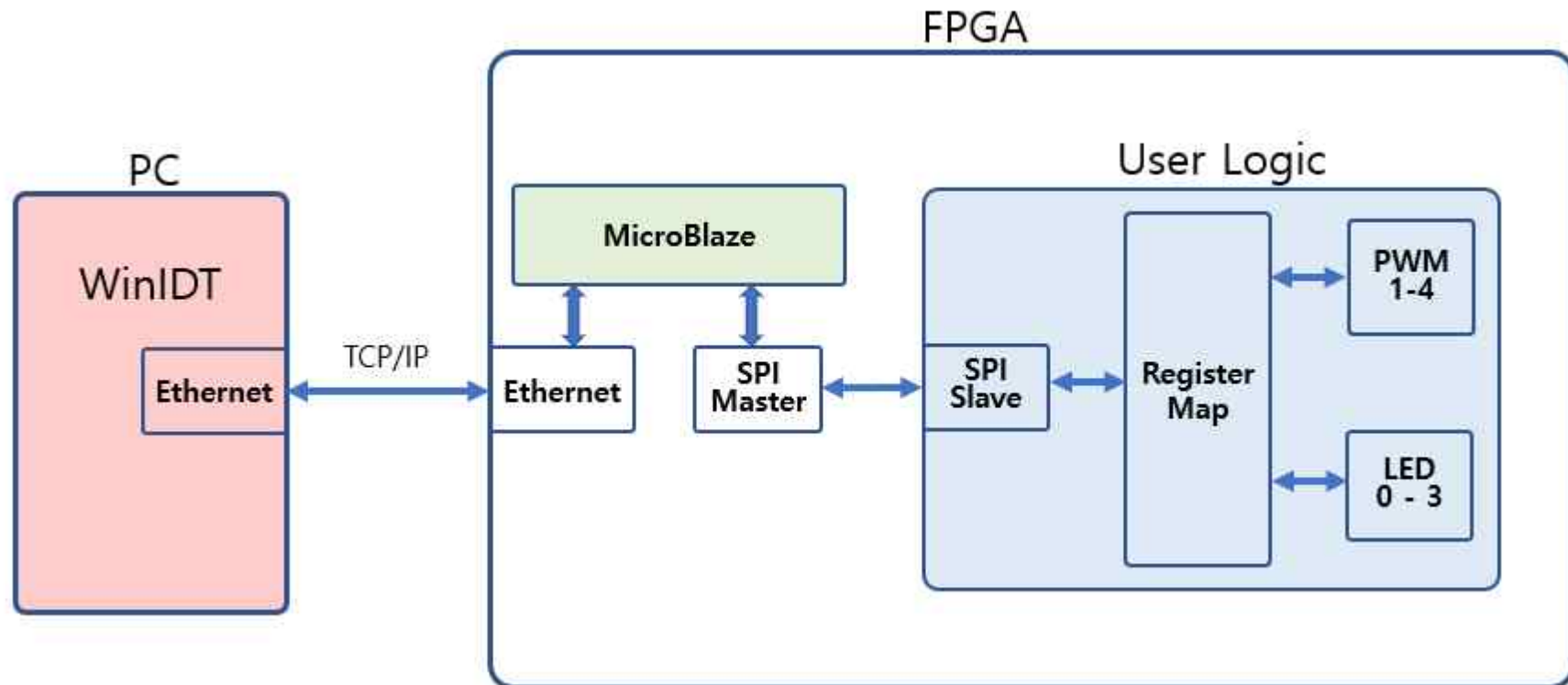


[그림] System Block

Utilize LightWeight IP (lwIP)

➤ LwIP 활용

- 앞에서 구현한 lwIP를 이용하여 PC와 TCP/IP 통신을 통하여 보드의 LED를 제어하는 것을 구현함
- 앞에서 UART 통신을 통하여서 PC에서 PWM을 제어하는 것을 구현하였는데, 이번엔은 TCP/IP 통신을 통하여 PC에서 보드의 LED를 제어하는 것을 구현함
- 아래는 System Block을 보여줌

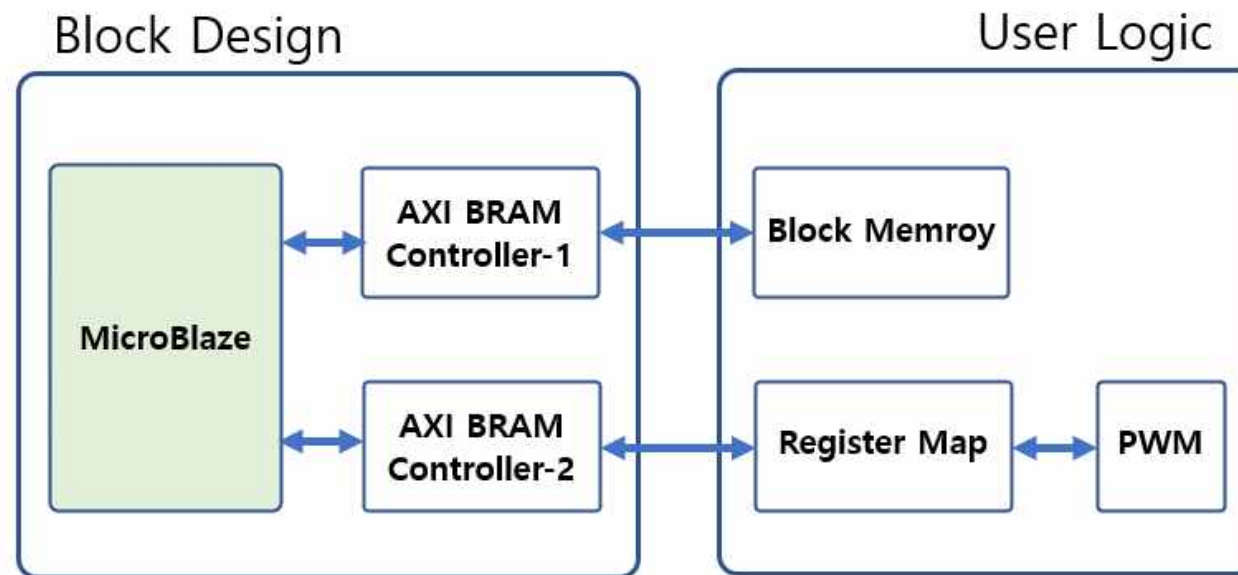


Block Memory Interface Implementation

Block Memory Interface Implementation blaze_w5500_Exam *.xpr

➤ Block Memory Interface Implementation 실습

- Block Memory Interface를 이용한 User Logic Register Map을 구현함. User Logic Interface Implementation에서 SPI Interface (Master / Slave)를 통하여 User Logic Register Map을 구현하였지만, 속도나 효율면에서는 이번장에서 구현하는 방법이 다소 유리
- 이번장에서는 Block Memory Controller는 Block Design에서 구현하고, Block Memory Generator와 User Logic Register map은 사용자가 직접 구현
- 아래는 Block Diagram을 보여줌. 응용 SW는 Microblaze에서 AXI BRAM Controller-2에 Memory를 Access 하듯이 Register Map에 Access하고, 결과적으로 PWM을 제어할 수 있음

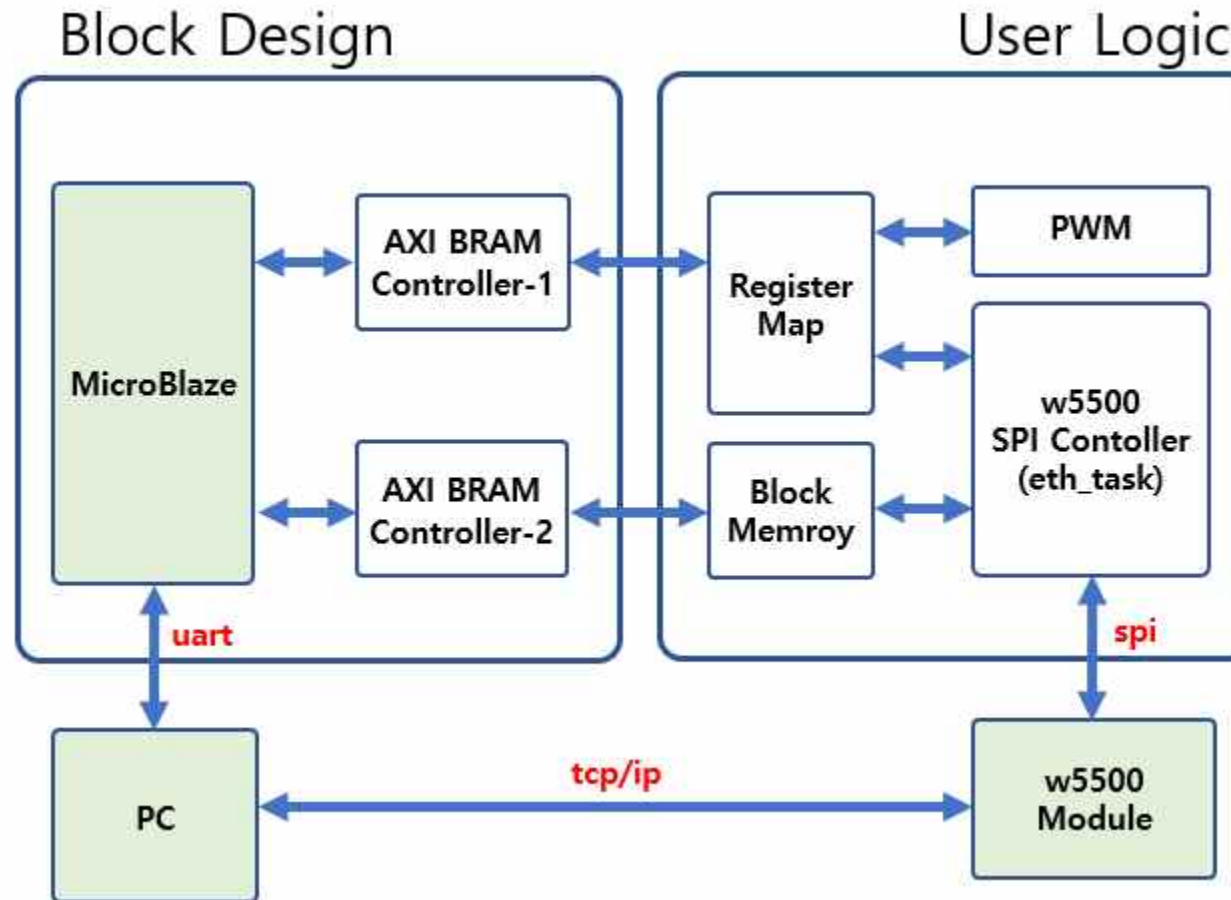


W5500 Interface Implementation

W5500 Interface Implementation

blaze_w5500_Exam *.xpr

➤ System Block



- **Block Design**에서 **AXI BRAM Controller** 2개를 사용함
- 1개는 **User Register Map**을 구현하는데 사용되고, 나머지 하나는 **w5500 Interface**를 위한 **SPI Controller**의 **Data Buffer(Block Memory)**로 사용됨
- Data Buffer는 Microblaze에서도 Access가 가능하고, w5500 SPI Controller에서도 Access가 가능

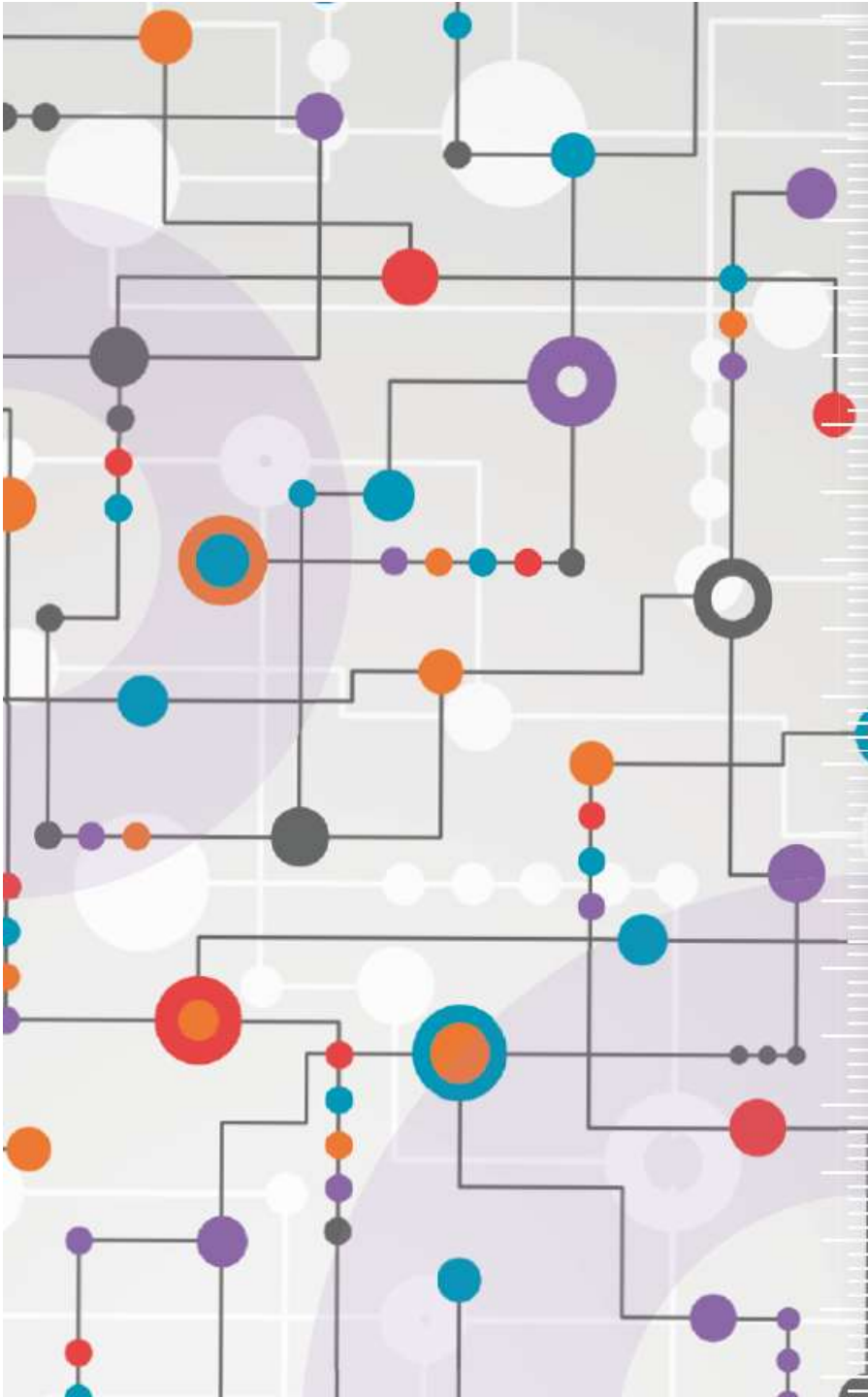
W5500 Interface Implementation

blaze_w5500_Exam *.xpr

➤ System Block

- Microblaze에서 PC로 데이터를 전송하는 동작 시퀀스는 다음과 같음
 - ✓ Data Buffer에 Microblaze가 Access하도록 설정함
 - ✓ 전송할 데이터를 Data Buffer에 Write 함
 - ✓ Data Buffer를 w5500 spi controller가 Access하도록 설정함
 - ✓ w5500 spi controller에 전송할 데이터 사이즈를 설정한 후, Start flag를 Enable 함
 - ✓ w5500 spi controller는 Data Buffer의 데이터를 읽어서 w5500으로 데이터를 전송

- PC에서 전송한 데이터를 Microblaze에서 수신하는 동작 시퀀스는 다음과 같음
 - ✓ w5500의 상태정보를 읽어서 수신할 데이터가 있는지 확인함
 - ✓ 수신할 데이터가 있으면, w5500 spi controller에 Start Flag를 Enable 해서 데이터를 수신함
 - ✓ w5500 spi controller는 w5500에서 수신한 데이터를 Data Buffer에 저장함
 - ✓ Microblaze는 Data Buffer에서 수신한 데이터를 읽음



수고하셨습니다.

SoC 를 위한 *Peripheral* 설계

Reference : MicroBlaze.v15 [IHIL]

2024-06-20

 Kim.S.W

 대한상공회의소
인력개발원 @IHIL

digilent.com

 XILINX®

15/80

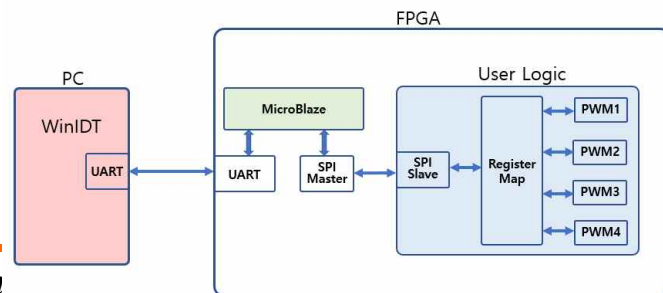
15/75

TCP/IP Implementation Using W5500

Table of Contents

➤ SoC를 위한 Peripheral 설계

1. Xilinx IP
2. Create and Package New IP
3. SPI
 - 1) SPI Master
 - 2) SPI Slave
 - 3) SPI Controller
4. UART
5. AMBA
6. MicroBlaze_Hello World
7. MicroBlaze_LED_Counter
8. MicroBlaze_Peripheral Implementation
9. MicroBlaze_User Logic Interface

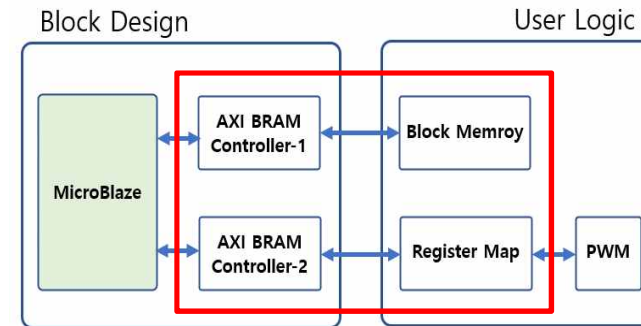


10. SPI_Master_IP(MicroBlaze_User Logic Interface)

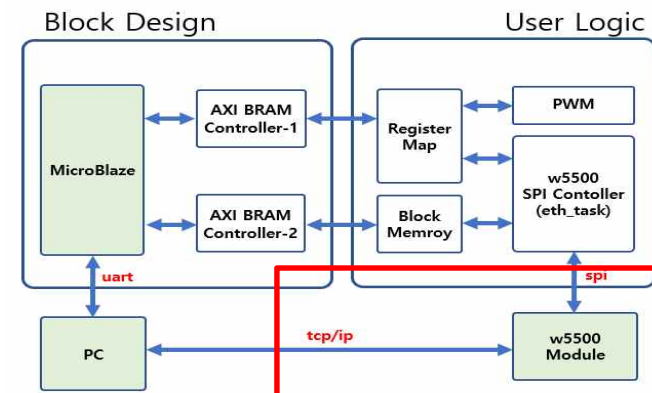
11. TCP_IP Implementation Using W5500

12. MicroBlaze_Block Memory Interface-1

13. MicroBlaze_Block Memory Interface-2



14. w5500 Interface Implementation



➤ SoC Peripheral RTC Design Project

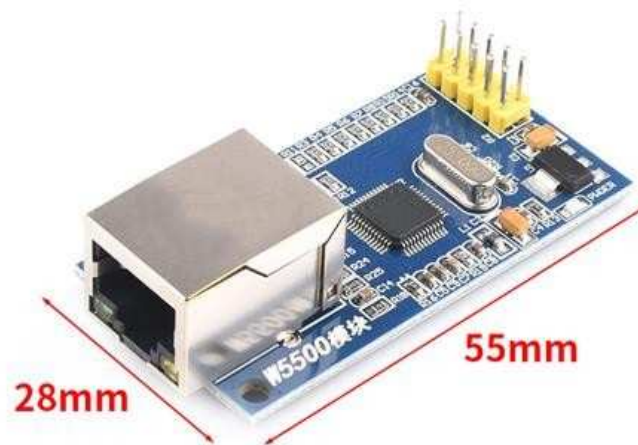
TCP/IP Implementation Using W5500

➤ *TCP/IP Implementation Using W5500* 개요

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

TCP/IP Implementation Using W5500

- wiznet사의 W5500 모듈을 이용한 TCP/IP 를 구현
- W5500 은 wiznet의 Hardware TCP/IP 기술을 이용한 임베디드용 인터넷 솔루션으로 하나의 칩에 TCP/IP 프로토콜 처리부터 10/100 Ethernet PHY와 MAC을 모두 내장 (w5500 datasheet 내용)
- W5500 은 SPI 인터페이스를 지원 : microblaze 와 spi를 이용하여 w5500을 제어 → PC와 Network 으 로 연결해서 데이터를 주고 받는 테스트를 진행 → SW 구현은 wiznet에서 제공하는 API를 사용하여 필요한 부분만 포팅해서 사용
- 아래 그림은 실습에 사용된 모듈 (JK 전자 W5500 TCP-IP 이더넷 모듈 SPI)



위 사진 기준 핀 배열	
3.3V	5V
MISO	GND
MOSI	RST
SCS	INT
SCL	NC

[그림] W5500 TCP-IP Ethernet module SPI

TCP/IP Implementation Using W5500

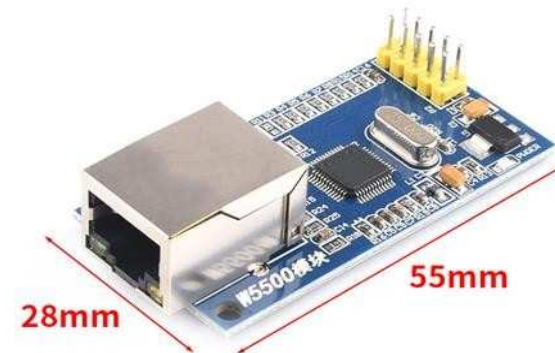
➤ *TCP/IP Implementation Using W5500*

✓ *Pin Mapping*

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Basys3 → Pin Mapping

- w5500 Module Pin Map : 전원은 3.3V 인가

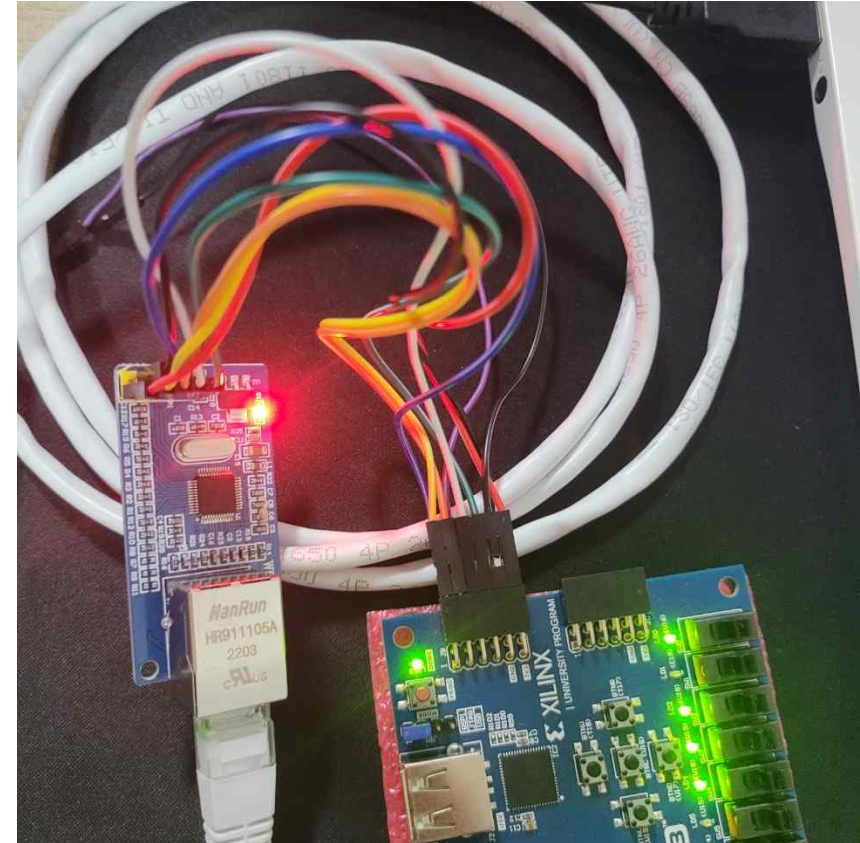
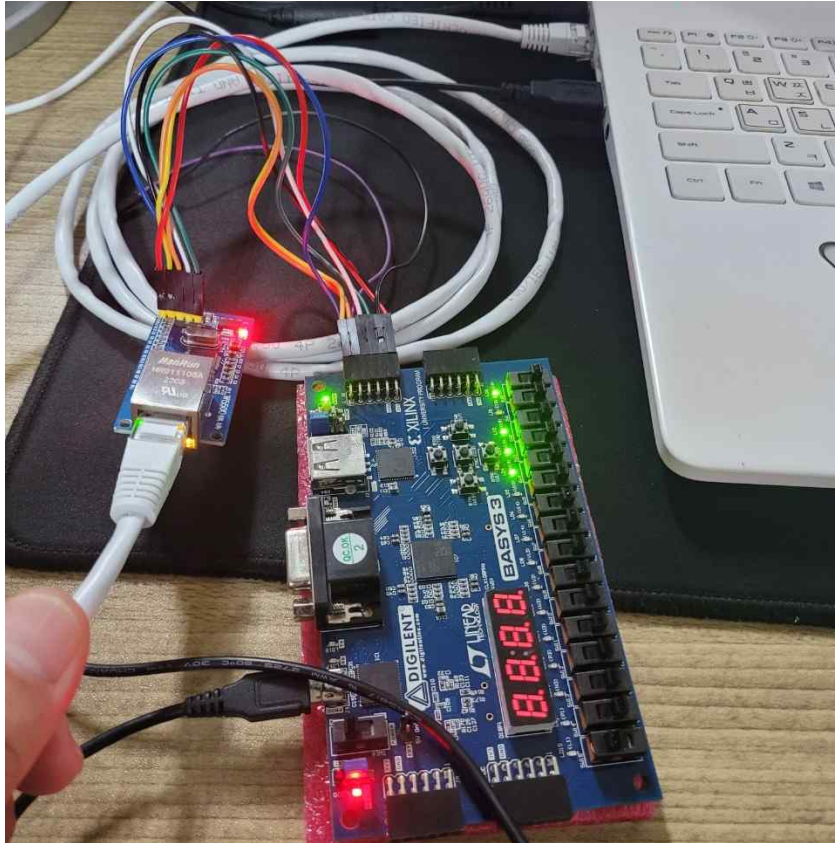


위 사진 기준 핀 배열	
3.3V	5V
MISO	GND
MOSI	RST
SCS	INT
SCL	NC

Basys3 → JB(or JA or JC)	w5500 Module
1 (JB1, ETH_MISO)	(MISO)
2 (JB2, EHT_MOSI)	(MOSI)
3 (JB3, ETH_SCS)	(SCS)
4 (JB4, ETH_SCK)	(SCK)
7 (J7, ETH_RST)	(RST)
	(INT)
5	GND
6	3.3V

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

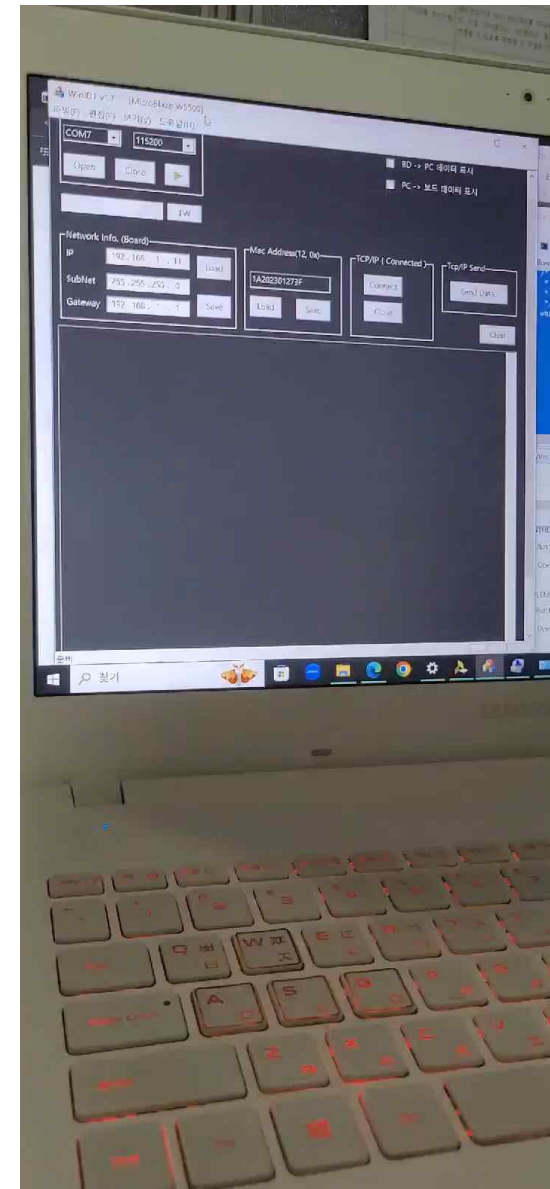
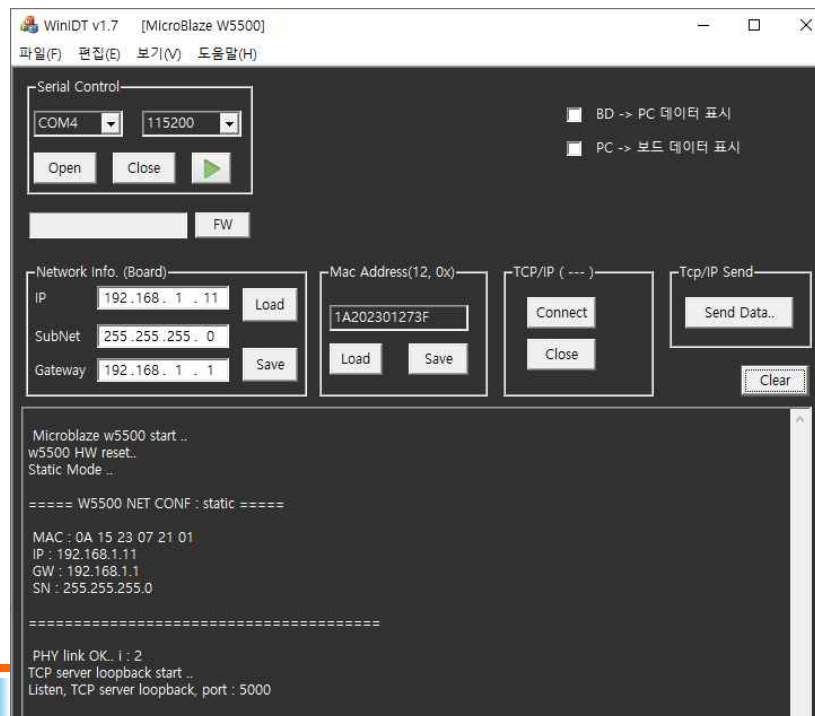
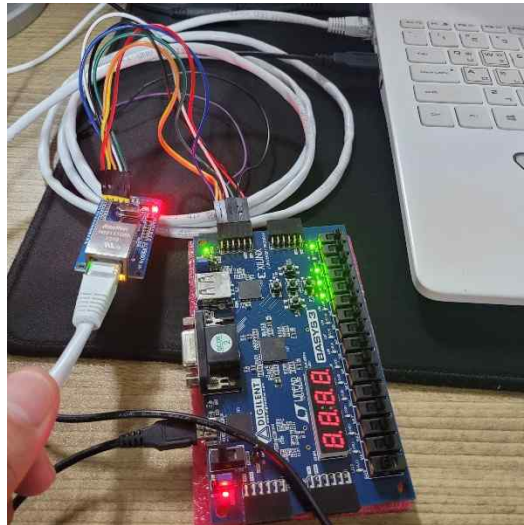
➤ Basys3 → Pin Mapping → Result



- w5500 모듈에 3.3V 전원 공급
- w5500 모듈과 연결 → JB connector
- w5500 모듈
- 보드에서 전원 공급

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Basys3 → Pin Mapping → 구현 결과



TCP/IP Implementation Using W5500

- 프로젝트 생성
- *Block Design*
- *Top Module Implementation*
- *Application SW Implementation*
- 결과 확인

TCP/IP Implementation Using W5500

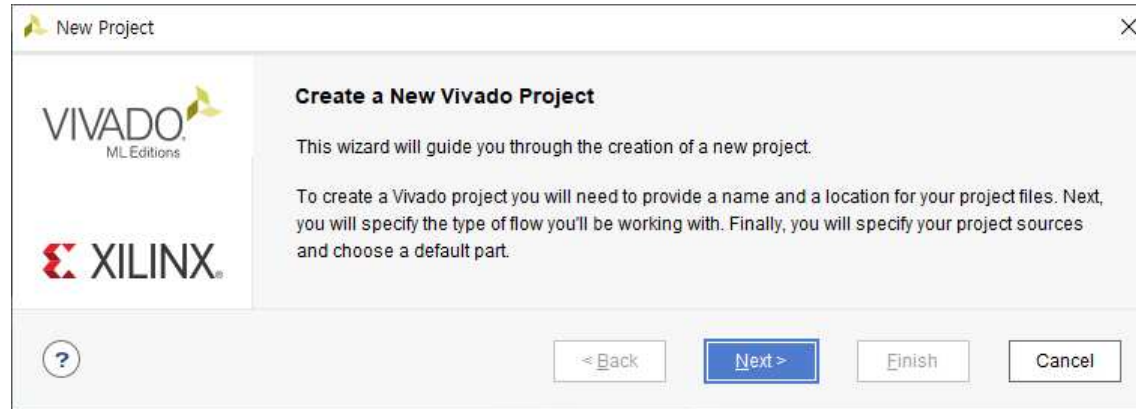
W5500 Exam *.xpr

- **Project Create**
- *Block Design*
- *Top Module Implementation*
- *Application SW Implementation*
- 결과 확인

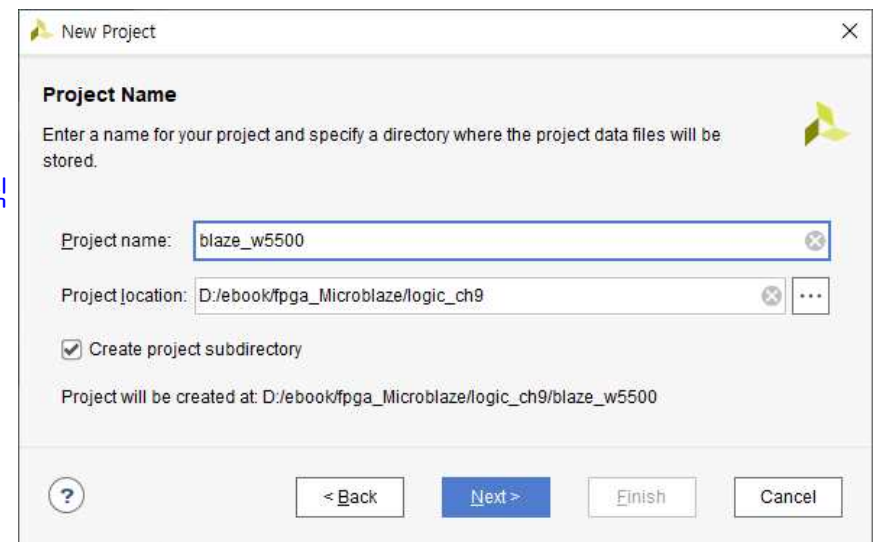
TCP/IP Implementation Using W5500^{blaze_w5500_Exam *.xpr}

➤ Project Create

- vivado 2023.1 을 실행 → Create Project를 클릭 → Project를 생성 → Next 클릭



- 프로젝트 이름은 “blaze_w5500”
 - Project Type : RTL Project 선택 → Next를 클릭
 - Add Sources, Add Constraints 윈도우에서 Next 클릭 → 나중에 소스와 xdc 파일을 추가
 - Part : xc7a35tcsg326-1 을 선택하고 Next 클릭
 - Finish 클릭 → 프로젝트 생성



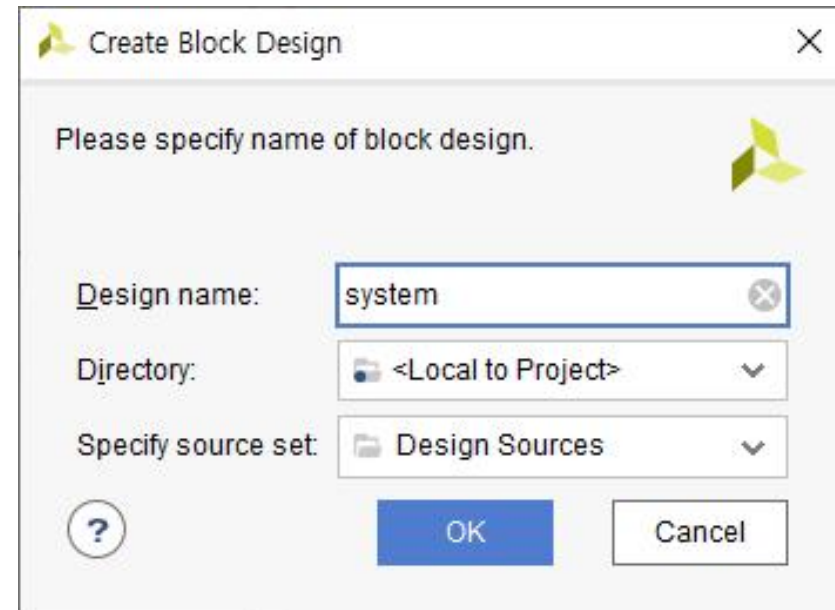
TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Block Design

- 새로운 Block을 생성 위해 PROJECT MANAGER에서 **IP INTEGRATOR** → **Create Block Design**을 클릭



- Design Name : system** 으로 하고 OK를 클릭



- Diagram** 원도가 생성 → Add IP (+ 아이콘)을 클릭하여 4개의 IP 추가 → **Run Block Automation, Run**

Connection Automation은 4개를 모두 추가한 후에 진행

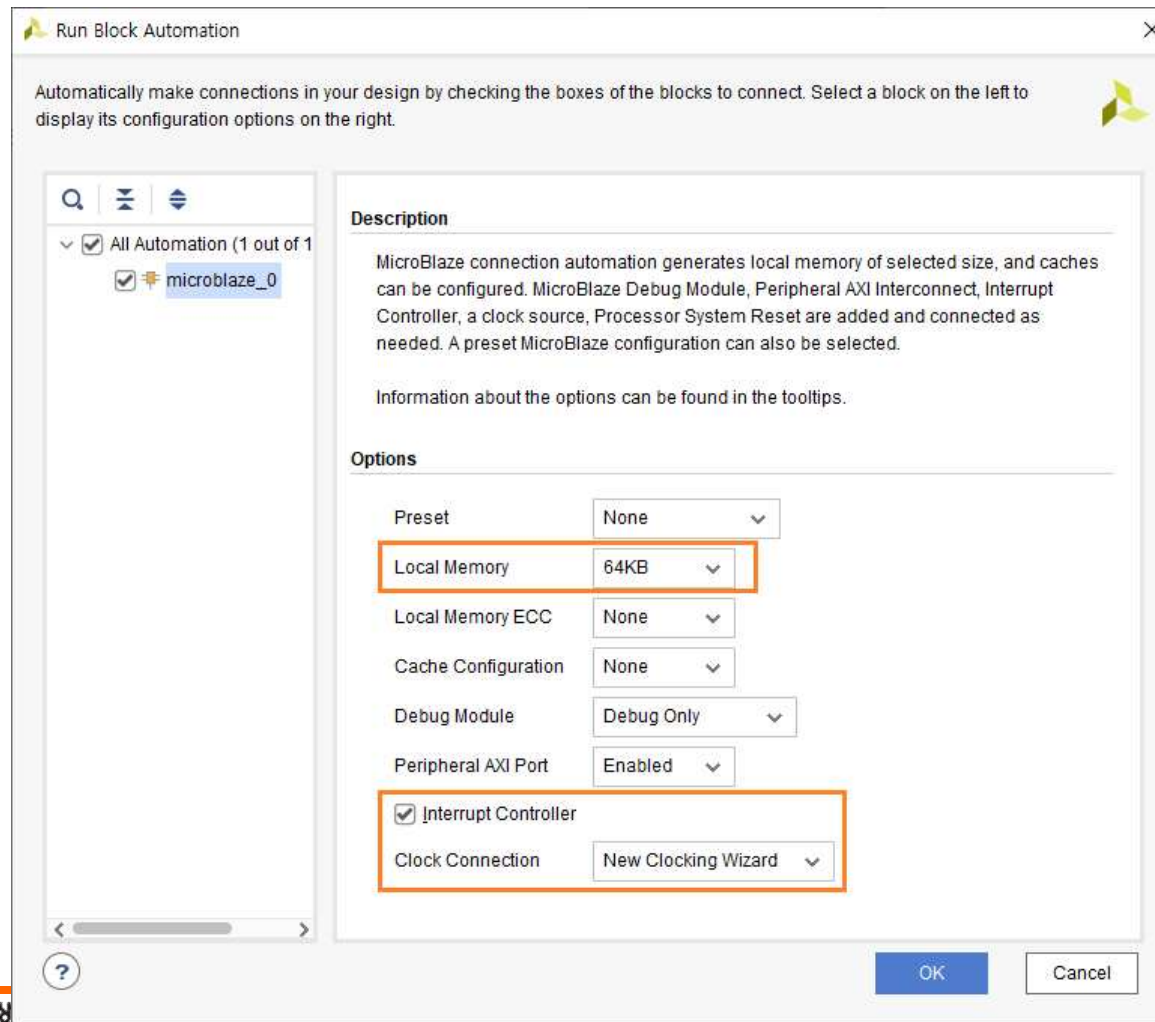
- ✓ **MicroBlaze**
- ✓ **AXI Quad SPI**
- ✓ **AXI Uartlite**
- ✓ **AXI GPIO**

- 4개의 IP를 추가한 후에 **Run Block Automation**을 클릭

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ **Block Design** → 첫번째 **Microblaze**의 속성을 설정

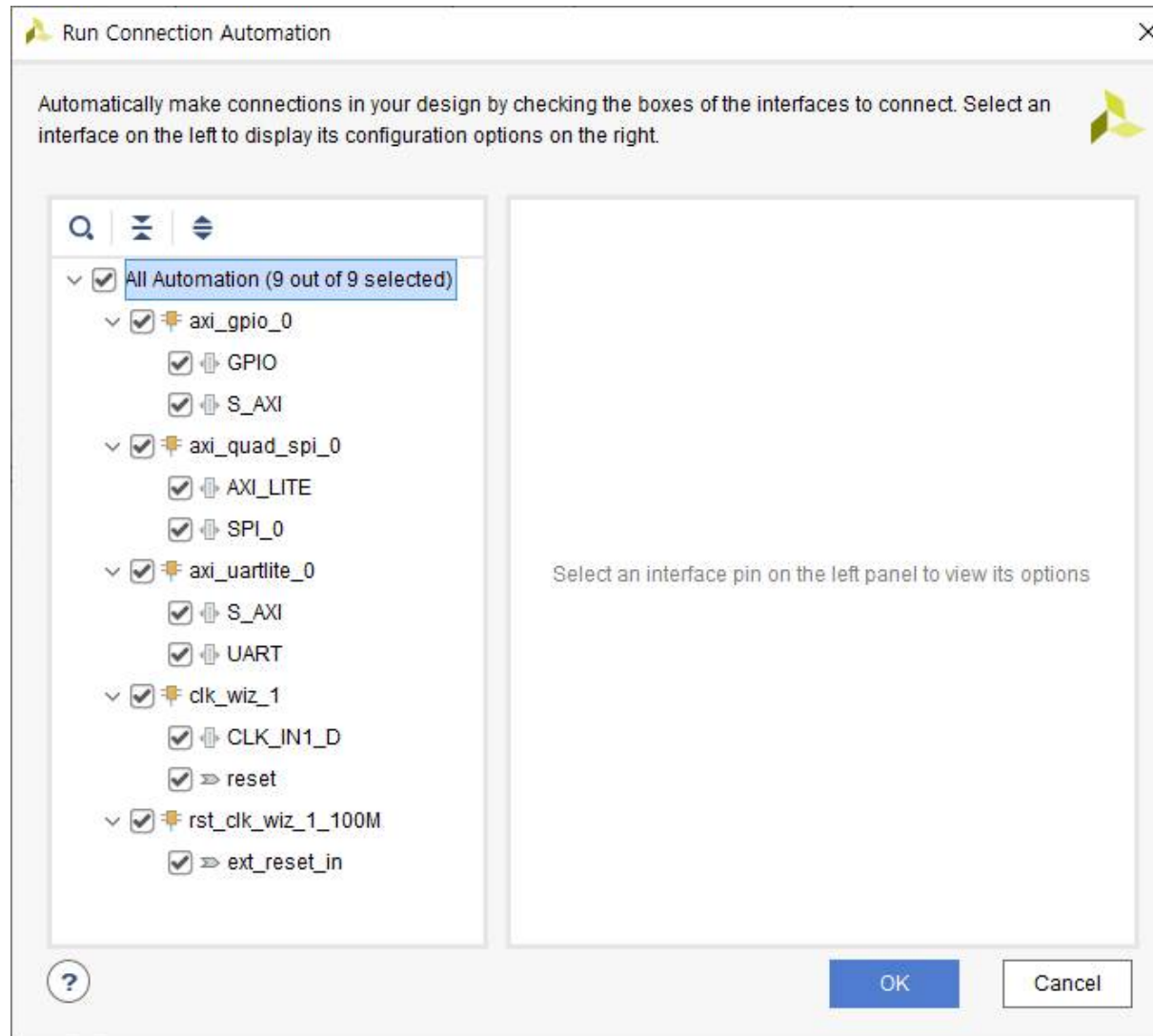
- **Local Memory : 64KB** (32KB도 가능하지만, 64KB로 설정)
- **Interrupt Controller : check**
- **Clock Connection : New Clocking Wizard**



TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ **Block Design** → 첫번째 Microblaze의 속성을 설정

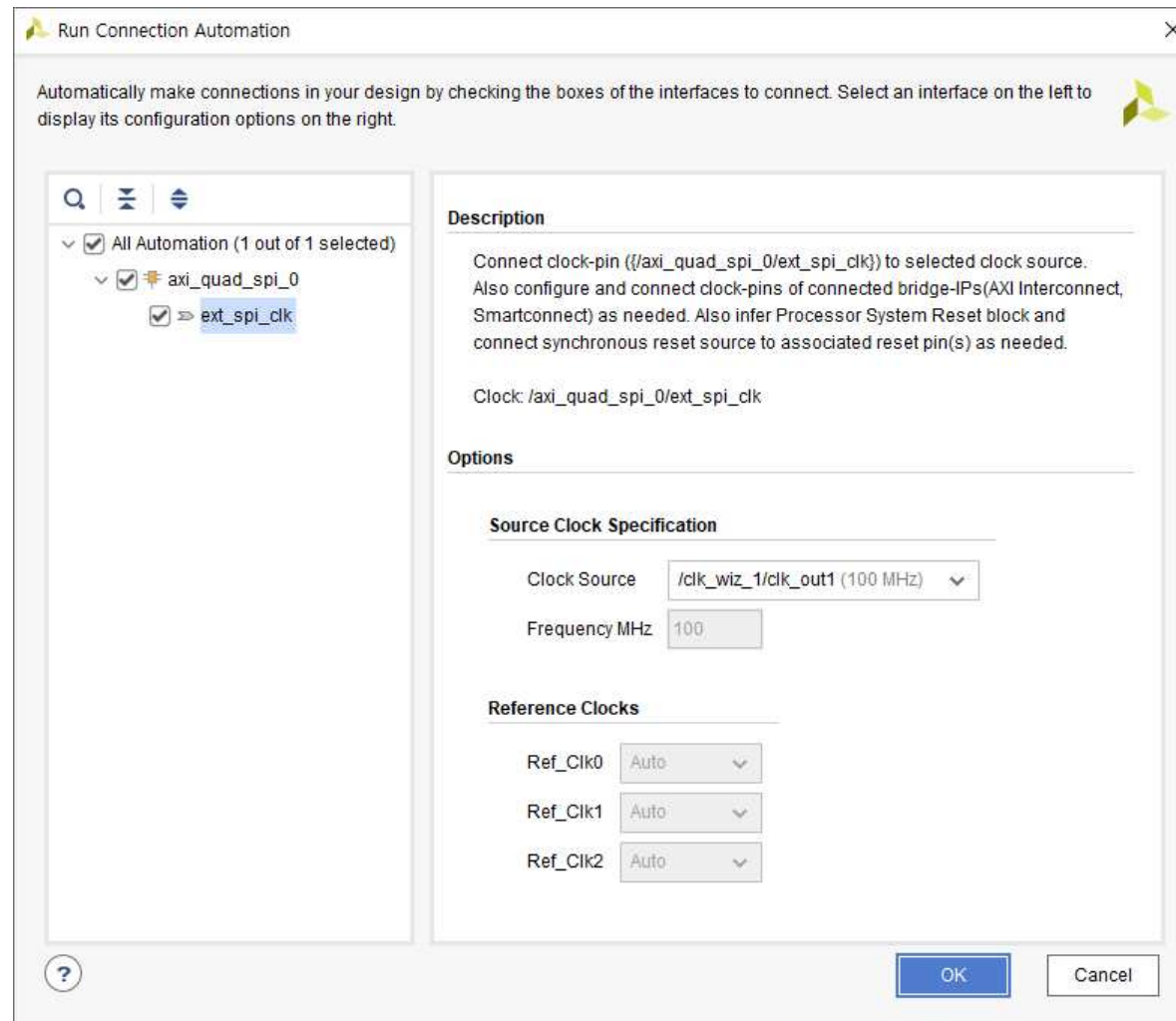
- Run Connection Automation을 클릭 → All Automation을 클릭 → 전체 모듈을 선택 → OK를 클릭



TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Block Design → 첫번째 Microblaze의 속성을 설정

- Run Connection Automation을 클릭 → All Automation을 클릭 → 전체 모듈을 선택 → OK를 클릭



TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Block Design ➔ 첫번째 Microblaze의 속성을 설정

- Validate Design, Regenerate Layout을 차례로 클릭합니다. Error가 없음을 확인

❖ IP의 속성을 설정

1) Clocking Wizard (clk_wiz_1)을 더블 클릭해서 속성을 변경

- Clocking Options 탭 ➔ A. Input Clock (clk_in1) : Source를 Single ended clock capable pin

Input Clock	Port Name	Input Frequency(MHz)	Jitter	Input Jitter	Source
<input checked="" type="checkbox"/> Primary	clk_in1	AUTO 100.000	10.000 - 100.000 UI	0.010	Single ended clock capable pin
<input type="checkbox"/> Secondary	clk_in2	AUTO 100.000	60.000 - 120.000 UI	0.010	Single ended clock capable pin

- Output Clocks 탭 ➔ A. Reset Type : Active Low ➔ OK

Reset Type

☐ Active High ☒ Active Low

- Run Connection Automation이 다시 활성화 ➔ reset_rtl_0 핀과 diff_clock_rtl_0 핀과 clk_wiz_1 을 서로 연결하기 위하여 Run Connection Automation을 클릭 (수동 연결 가능) ➔ All Automation을 클릭 ➔ OK를 클릭

Run Connection Automation

Automatically make connections in your design by checking the boxes of the interfaces on the right.

☒ All Automation (2 out of 2 selected)

☒ clk_wiz_1

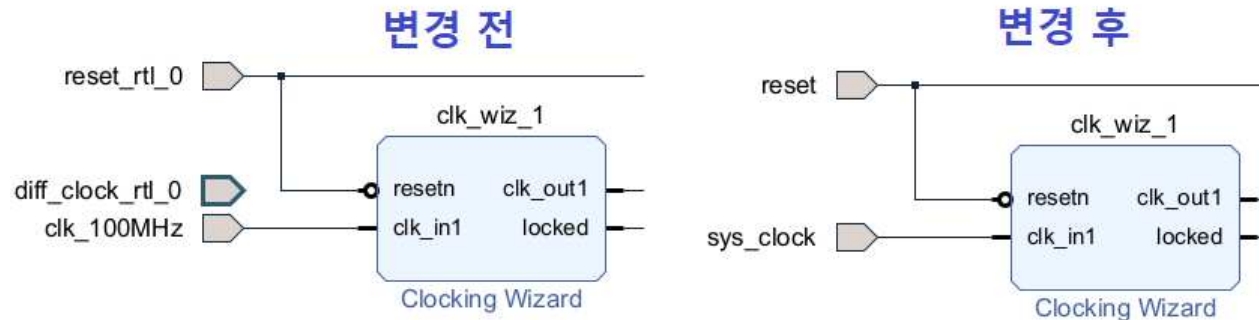
☒ clk_in1

☒ resetn

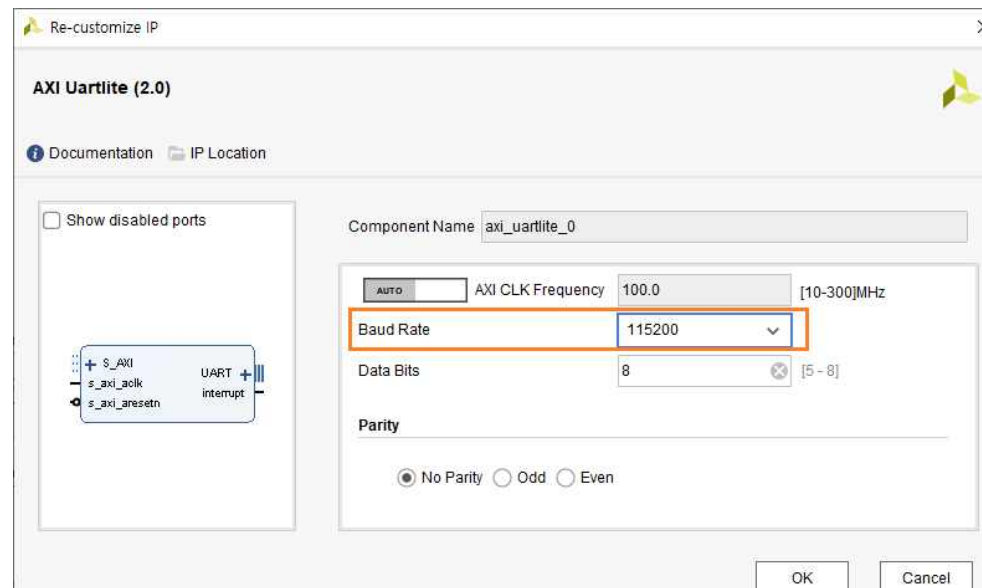
TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Block Design → **Clocking Wizard**의 속성을 설정

- **diff_clock_rtl_0**를 삭제 **reset_rtl_0**를 **reset**으로, **clk_100MHz**를 **sys_clock**으로 변경 → 포트 이름 변경은 포트를 선택하면 왼쪽에 Port Properties 윈도 → Name을 변경



- **axi_uartlite_0**을 더블 클릭해서 속성을 변경 → **Baud Rate : 115200**으로 변경하고 OK를 클릭



TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Block Design → **axi_quad_spi_0** 속성을 설정

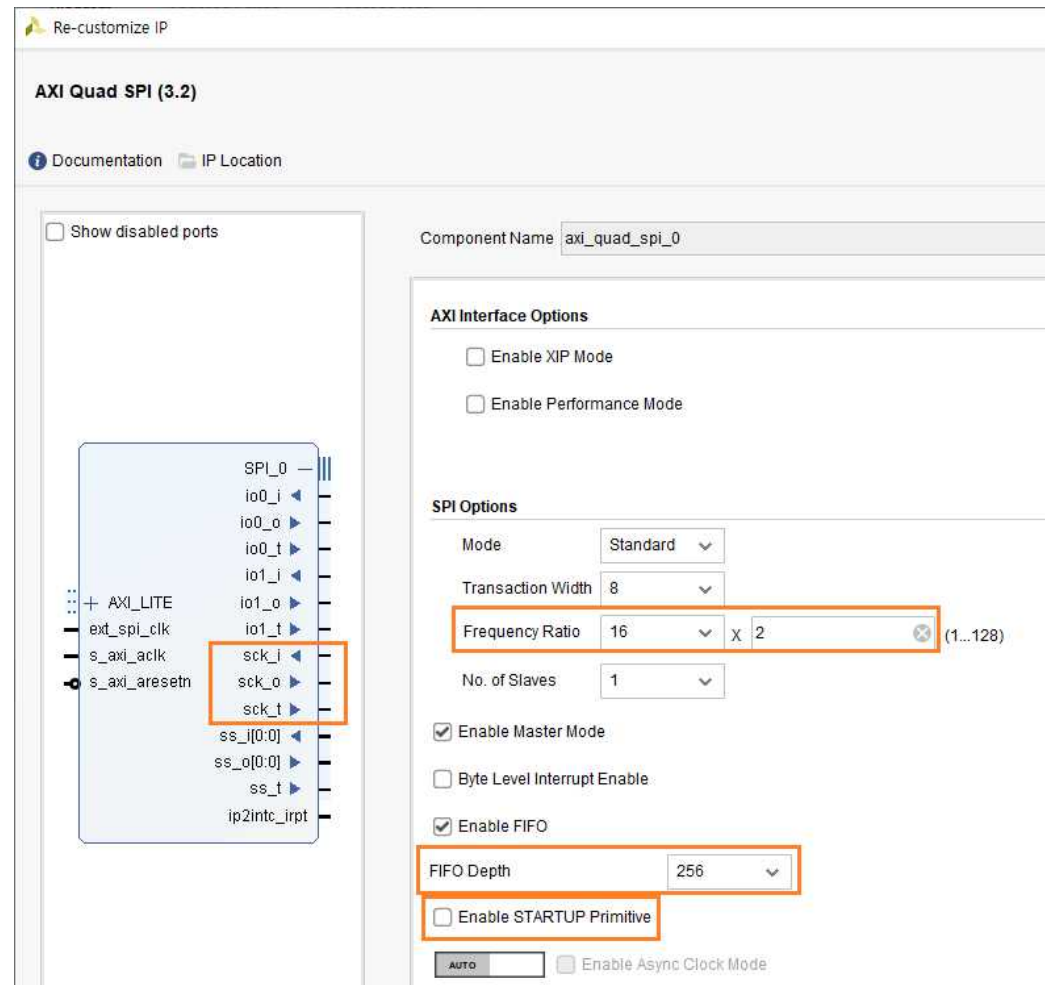
❖ **axi_quad_spi_0**를 더블 클릭해서 속성을 변경합니다. **spi** 속성은 매우 중요

➔ **Frequency Ratio : 16x2, 100Mhz / 32 = 3.125Mhz** (보드상에 점퍼로 연결해서 더 높이 설정하면 통신 에러 발생)

➔ **FIFO Depth : 16 or 256** 지원함, **256 선택** (한번에 최대 전송할 수 있는 사이즈)

➔ **Enable STARTUP Primitive :** 반드시 **uncheck** 해야 우측의 **sck** 핀들이 나타남

➔ OK를 클릭



TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Block Design → **axi_gpio_0**의 속성을 설정

➔ **axi_gpio_0**를 더블 클릭해서 속성을 변경

➔ gpio 핀은 총 5핀을 사용 ([0] : w5500_reset, [1] ~ [4] : led)

- *All Outputs : check*
- *GPIO Width : 5* ➔ OK를 클릭합니다.

➔ 포트 이름을 변경

- **gpio_rtl_0** → **gpio**
- **spi_rtl_0** → **spi**
- **uart_rtl_0** → **uart**

➔ 인터럽트 핀들을 연결

- microBlaze_0_xlconcat의 In0[0:0] 핀과 axi_quad_spi_0의 ip2intc_irpt 핀
- microBlaze_0_xlconcat의 In1[0:0] 핀과 axi_uartlite_0의 interrupt 핀

Component Name: axi_gpio_0

GPIO

- ☐ All Inputs
- ☒ All Outputs
- GPIO Width: 5 [1 - 32]
- Default Output Value: 0x00000000 [0x00000000, 0xFFFFFFFF]
- Default Tri State Value: 0xFFFFFFFF [0x00000000, 0xFFFFFFFF]
- ☐ Enable Dual Channel

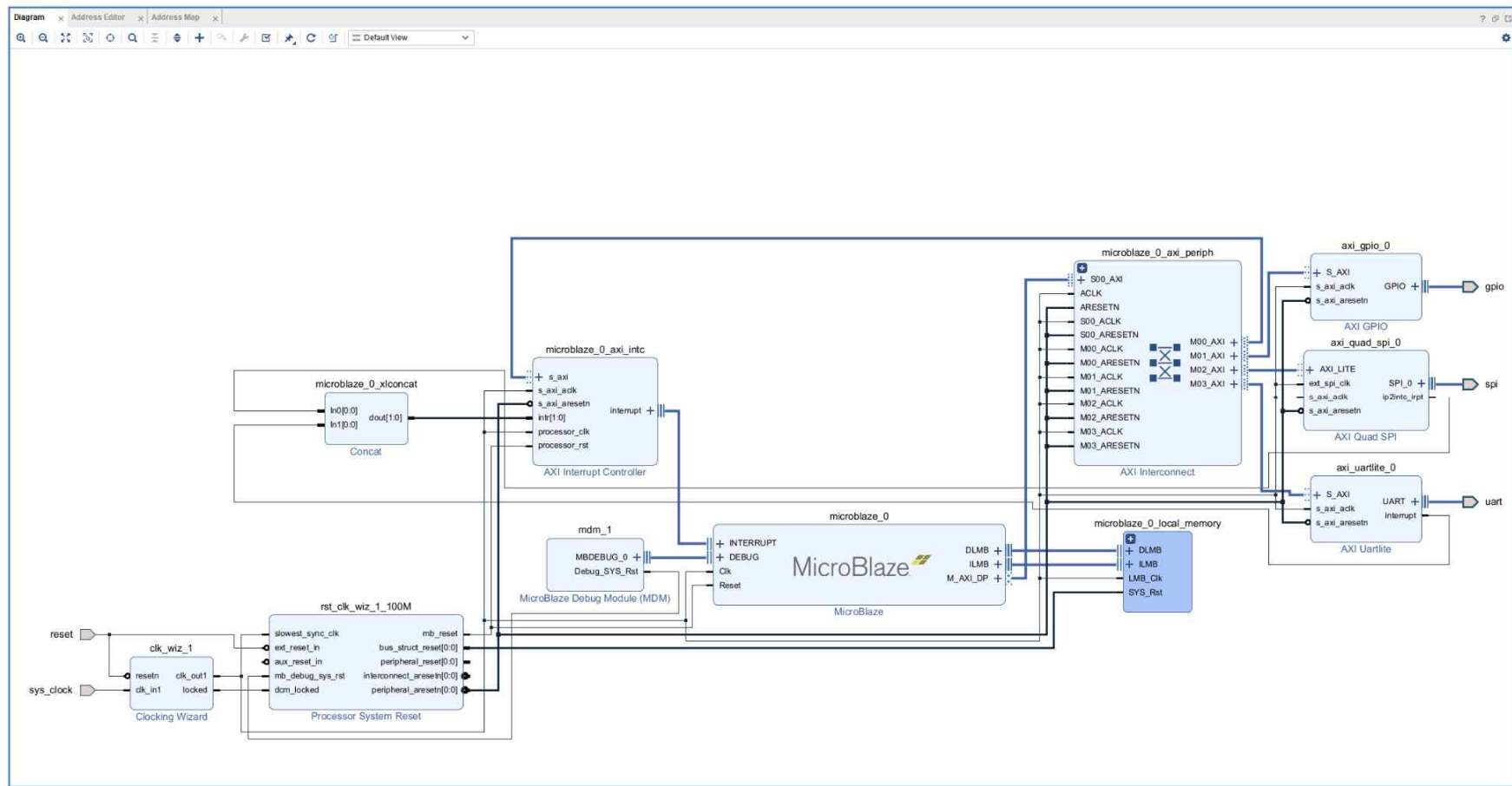
GPIO 2

- ☐ All Inputs
- ☐ All Outputs
- GPIO Width: 32 [1 - 32]
- Default Output Value: 0x00000000 [0x00000000, 0xFFFFFFFF]
- Default Tri State Value: 0xFFFFFFFF [0x00000000, 0xFFFFFFFF]
- ☐ Enable Interrupt

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Block Design ➔ Result

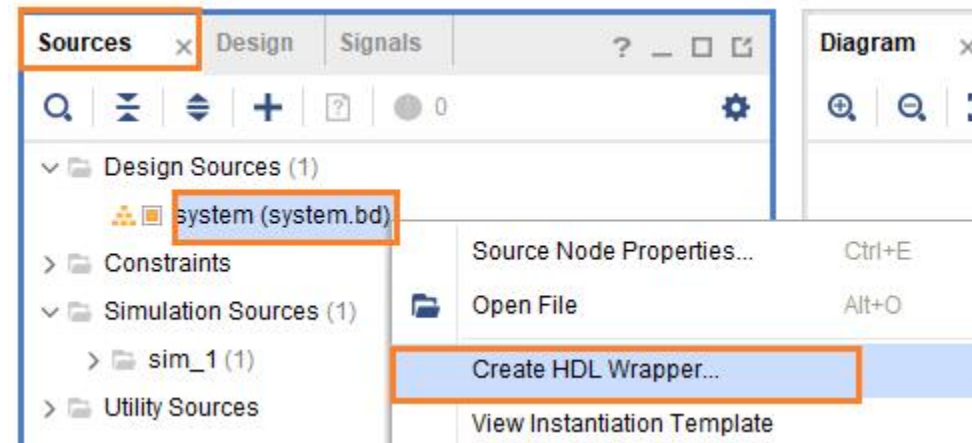
- ❖ *Validate Design, Regenerate layout* 아이콘을 차례대로 클릭 ➔ *Validation successful* 메시지 창을 확인
- ❖ 아래 그림은 지금까지 구현된 *Diagram*



TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

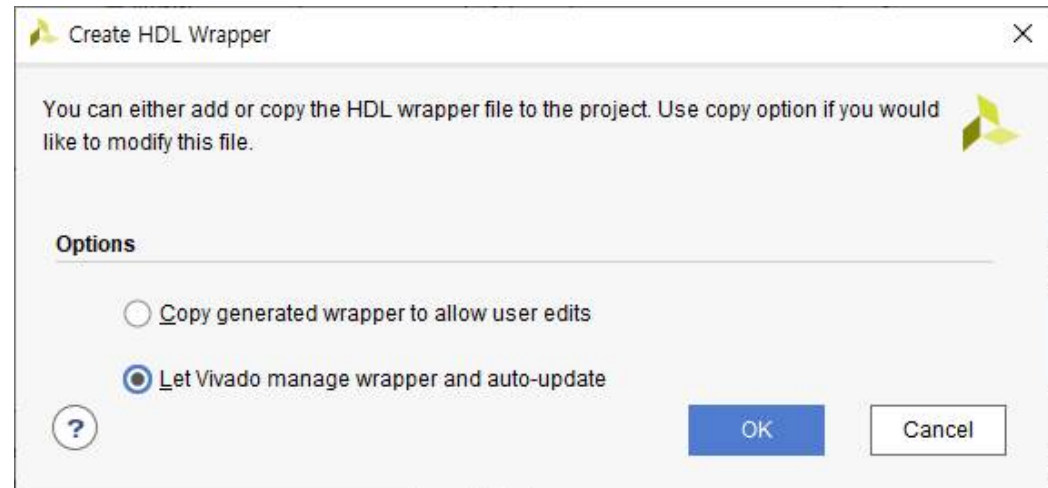
➤ HDL Wrapper

- ❖ Sources 탭에서 Design Sources → system
→ 우 클릭 → Create HDL Wrapper... 클릭



- ❖ OK를 클릭
- ❖ HDL Wrapper 파일이 생성되었습니다.
(system_wrapper.v)

➤ [Next] Top Module 구현



TCP/IP Implementation Using W5500^{blaze_w5500_Exam *.xpr}

➤ Top Module Implementation

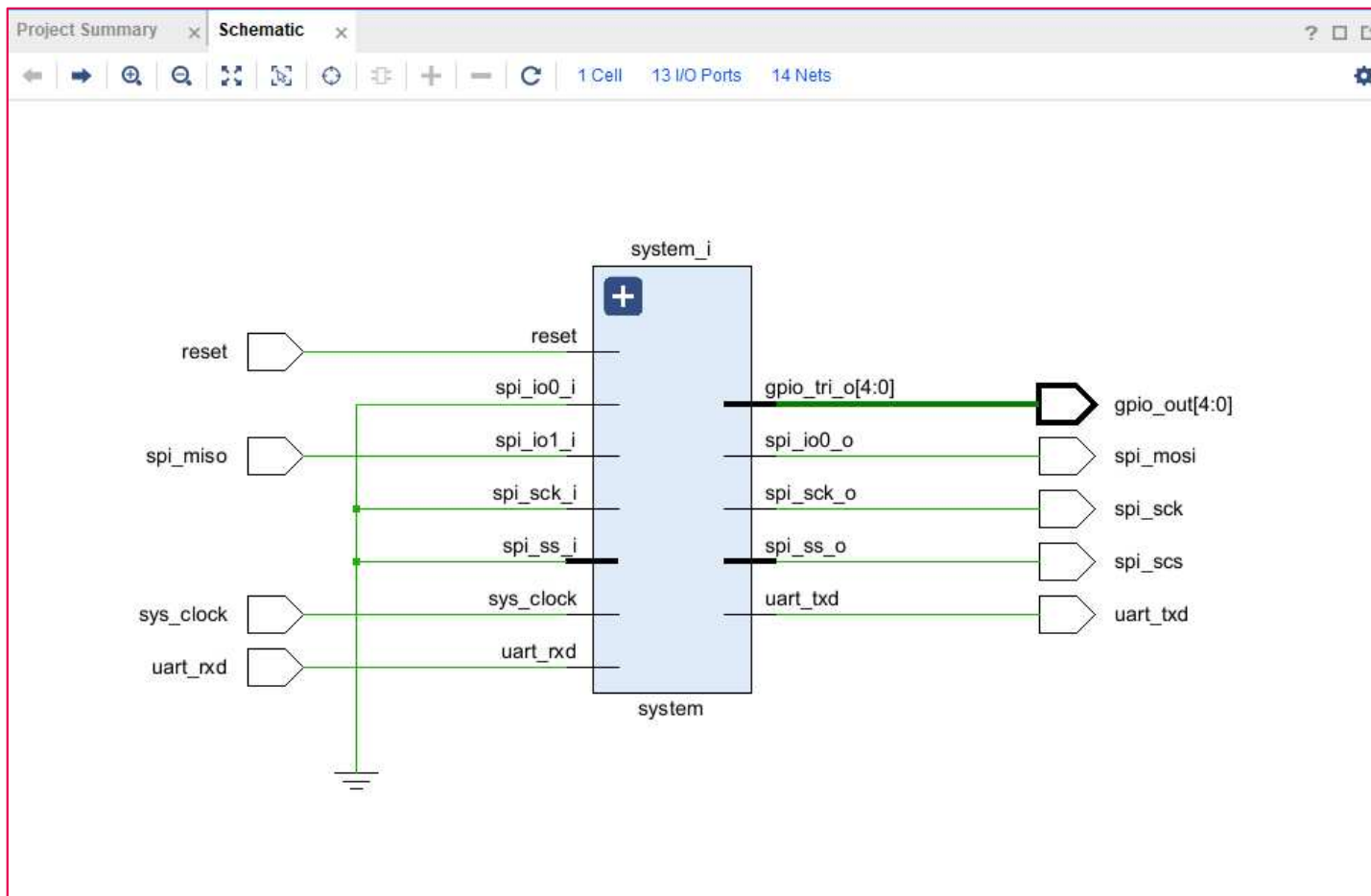
- ❖ 생성한 **system_wrapper.v** 를 이용하여 **Top Module**을 생성하고 **xdc** 파일 생성
- ❖ Design Sources ➔ 우클릭 ➔ Add Sources... 클릭해서
Top Module을 추가 : 모듈 이름 ➔ **"blaze_w5500Top"**
- ❖ Add Sources 윈도우에서 Create File 버튼을 클릭해서
"blaze_w5500Top" 입력 ➔ OK를 클릭 ➔ 모듈이 추가
되면 Finish 버튼 클릭
- ❖ 코드를 **blaze_w5500Top.v** 에 추가
 - 23 – 42 : 모듈 선언, in/out port 선언 (uart, spi, gpio 가 추가되었음)
 - 44 – 67 : system 모듈 추가

```
23 module blaze_w5500Top(  
24     reset  
25     sys_clock  
26     uart_rxd  
27     uart_txd  
28     gpio_out  
29     spi_scs  
30     spi_sck  
31     spi_mosi  
32     spi_miso  
33 );  
34 input      reset  
35 input      sys_clock  
36 input      uart_rxd  
37 output     uart_txd  
38 output [4:0] gpio_out  
39 output     spi_scs  
40 output     spi_sck  
41 output     spi_mosi  
42 input      spi_miso  
43  
44 wire       uart_txd  
45 wire [4:0] gpio_out  
46 wire       spi_scs  
47 wire       spi_sck  
48 wire       spi_mosi  
49 system     system_i (  
50     .reset      (reset      ),  
51     .sys_clock   (sys_clock  ),  
52     .uart_rxd    (uart_rxd   ),  
53     .uart_txd    (uart_txd   ),  
54     .gpio_tri_o  (gpio_out   ),  
55     .spi_io0_i   (1'b0      ),  
56     .spi_io0_o   (spi_mosi   ),  
57     .spi_io0_t   (            ),  
58     .spi_iol_i   (spi_miso   ),  
59     .spi_iol_o   (            ),  
60     .spi_iol_t   (            ),  
61     .spi_sck_i   (1'b0      ),  
62     .spi_sck_o   (spi_sck    ),  
63     .spi_sck_t   (            ),  
64     .spi_ss_i    (1'b0      ),  
65     .spi_ss_o    (spi_scs    ),  
66     .spi_ss_t    (            ),  
67 );  
68  
69 endmodule
```

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Top Module Implementation

❖ RTL ANALYSIS → Open Elaborated Design → Schematic



TCP/IP Implementation Using W5500^{blaze_w5500_Exam *.xpr}

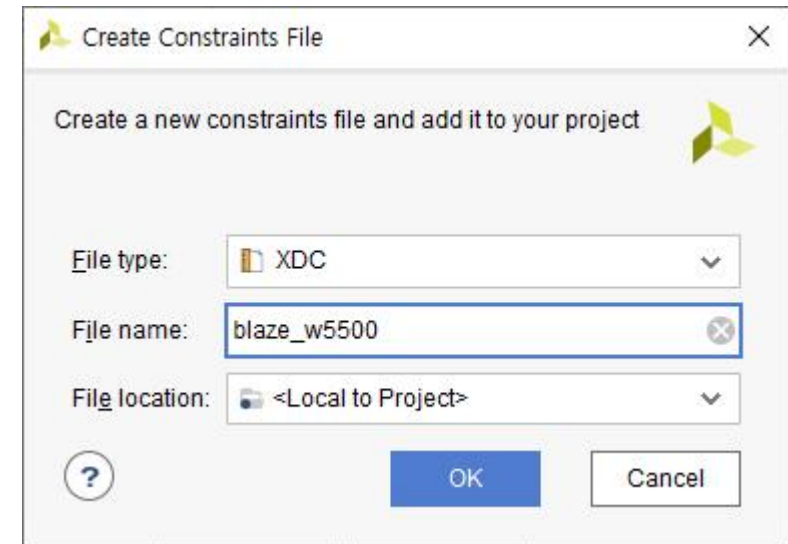
➤ Top Module Implementation

❖ xdc 파일을 생성 : Constraints → 우클릭 → Add

Sources... 를 클릭 → Add Sources 윈도우에서 Create File

을 클릭 → “blaze_w5500”을 입력 → OK를 클릭

❖ Finish 클릭 → 파일이 생성



TCP/IP Implementation Using W5500^{blaze_w5500_Exam *.xpr}

➤ Constraints(XDC) Implementation

❖ **blaze_w5500.xdc** 파일에 코드 추가 → **gpio_out[0]**를 **w5500_rst** 핀으로 사용한 것에 유의

Clock signal

```
set_property -dict { PACKAGE_PIN W5 IOSTANDARD LVCMOS33 } [get_ports sys_clock]
```

```
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports sys_clock]
```

Switches

```
set_property -dict { PACKAGE_PIN R2 IOSTANDARD LVCMOS33 } [get_ports {reset}]
```

LEDs

```
#set_property -dict { PACKAGE_PIN U16 IOSTANDARD LVCMOS33 } [get_ports {gpio_out[0]}]
```

```
set_property -dict { PACKAGE_PIN E19 IOSTANDARD LVCMOS33 } [get_ports {gpio_out[1] }]; #LED4
```

```
set_property -dict { PACKAGE_PIN U19 IOSTANDARD LVCMOS33 } [get_ports {gpio_out[2] }]; #LED5
```

```
set_property -dict { PACKAGE_PIN V19 IOSTANDARD LVCMOS33 } [get_ports {gpio_out[3] }]; #LED6
```

```
set_property -dict { PACKAGE_PIN W18 IOSTANDARD LVCMOS33 } [get_ports {gpio_out[4] }]; #LED7
```

##Pmod Header JB

```
set_property -dict { PACKAGE_PIN A14 IOSTANDARD LVCMOS33 } [get_ports {spi_miso}];#Sch name = JB1 , w5500_miso
```

```
set_property -dict { PACKAGE_PIN A16 IOSTANDARD LVCMOS33 } [get_ports {spi_mosi}];#Sch name = JB2 , w5500_mosi
```

```
set_property -dict { PACKAGE_PIN B15 IOSTANDARD LVCMOS33 } [get_ports {spi_scs}];#Sch name = JB3 , w5500_scs
```

```
set_property -dict { PACKAGE_PIN B16 IOSTANDARD LVCMOS33 } [get_ports {spi_sck}];#Sch name = JB4 , w5500_sck
```

```
set_property -dict { PACKAGE_PIN A15 IOSTANDARD LVCMOS33 } [get_ports {gpio_out[0]}];#Sch name = JB7 , w5500_rst
```

```
#set_property -dict { PACKAGE_PIN A17 IOSTANDARD LVCMOS33 } [get_ports {JB[5]}];#Sch name = JB8
```

```
#set_property -dict { PACKAGE_PIN C15 IOSTANDARD LVCMOS33 } [get_ports {JB[6]}];#Sch name = JB9
```

```
#set_property -dict { PACKAGE_PIN C16 IOSTANDARD LVCMOS33 } [get_ports {JB[7]}];#Sch name = JB10
```

##USB-RS232 Interface

```
set_property -dict { PACKAGE_PIN B18 IOSTANDARD LVCMOS33 } [get_ports uart_rxd]
```

```
set_property -dict { PACKAGE_PIN A18 IOSTANDARD LVCMOS33 } [get_ports uart_txd]
```

Configuration options, can be used for all designs

```
set_property CONFIG_VOLTAGE 3.3 [current_design]
```

```
set_property CFGBVS VCCO [current_design]
```

SPI configuration mode options for QSPI boot, can be used for all designs

```
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
```

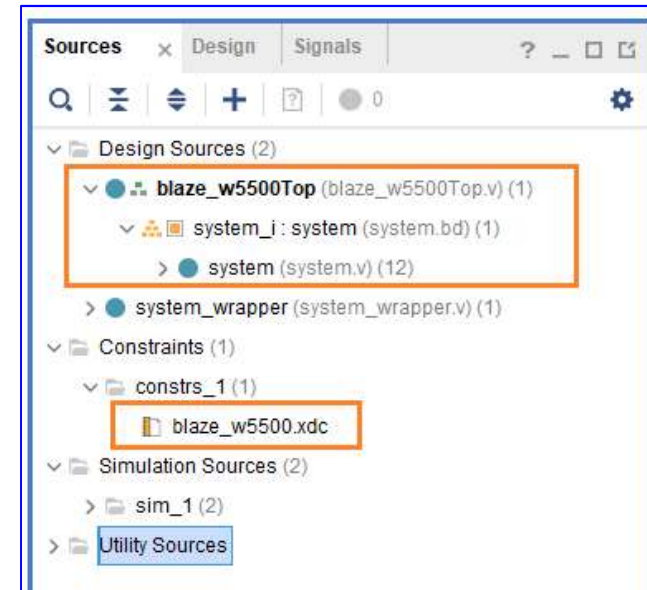
```
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
```

```
set_property CONFIG_MODE SPIx4 [current_design]
```


TCP/IP Implementation Using W5500^{blaze_w5500_Exam *.xpr}

➤ Top Module Implementation & Bitstream

- ❖ blaze_w5500Top 모듈을 Top 모듈로 지정

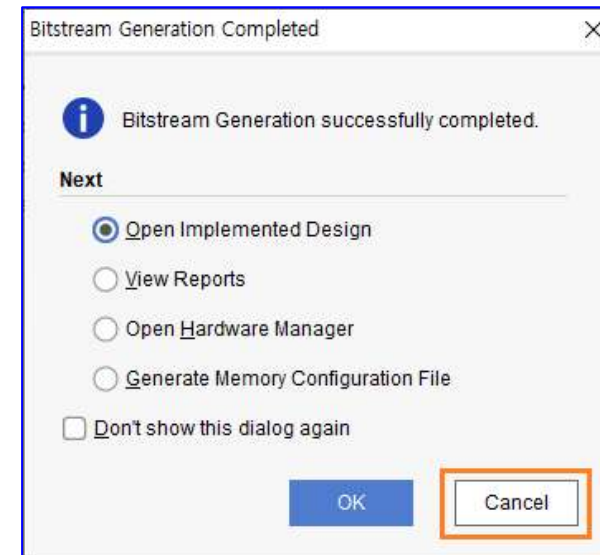


- ❖ 모든 파일이 준비 : **PROGRAM AND DEBUG** →

Generate Bitstream을 클릭 → **Bitstream**을 생성

- ❖ 각 IP들을 생성 → **Synthesis, Implementation** 까지

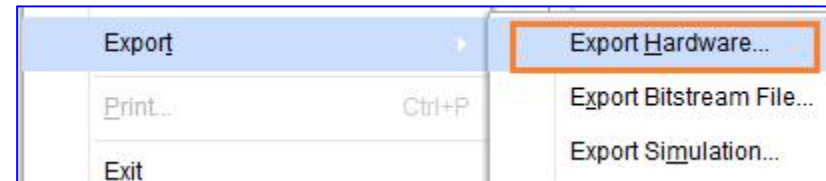
진행해서 **Bitstream**을 생성



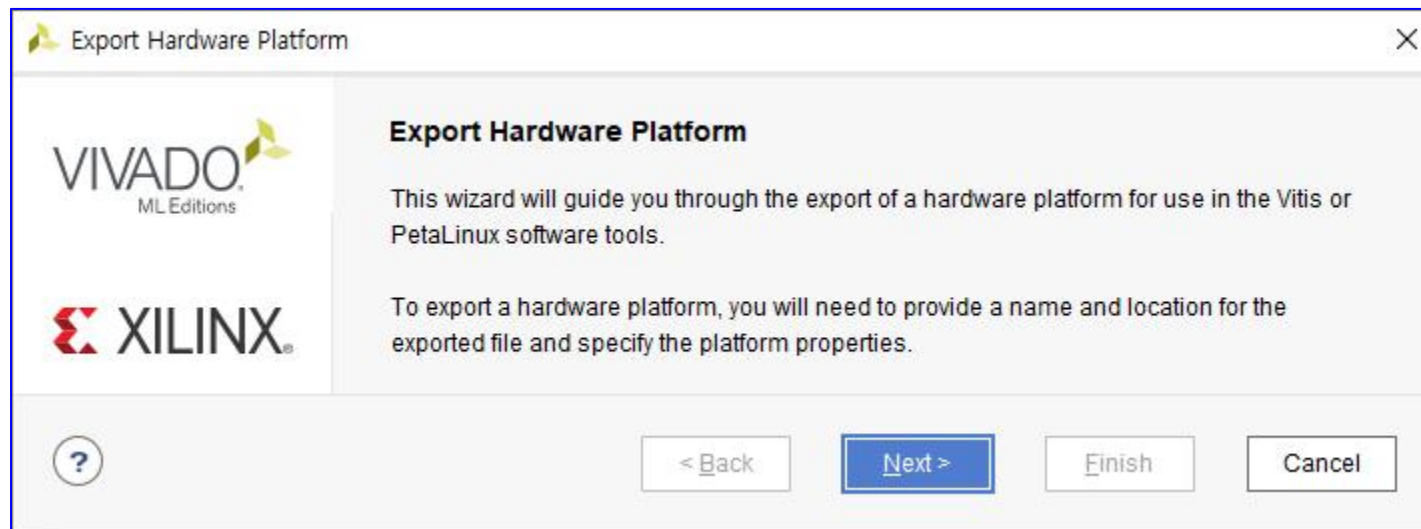
TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Export Hardware Platform

- ❖ xsa 파일을 생성하기 위하여 **File – Export – Export Hardware...** 를 클릭



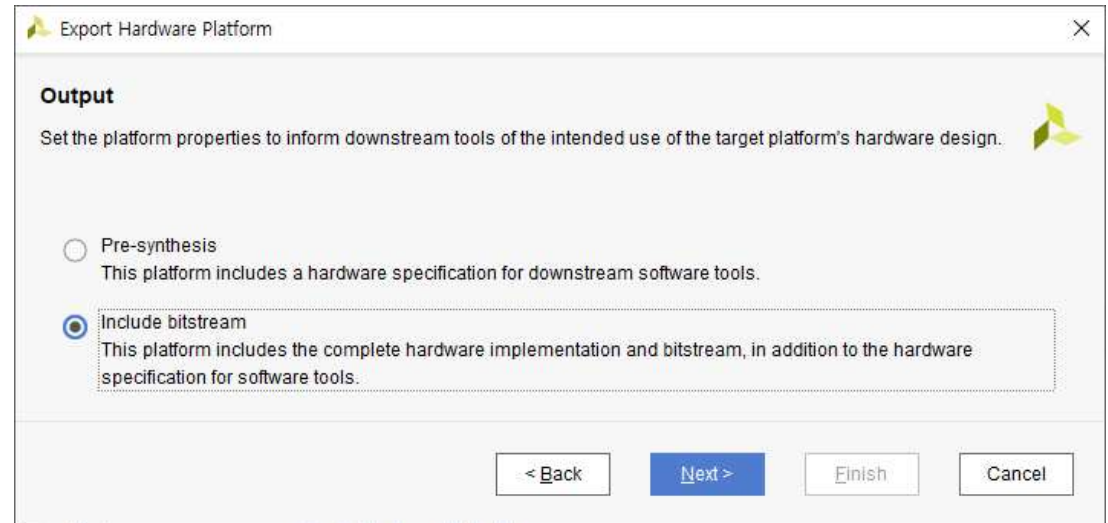
- ❖ Next를 클릭



TCP/IP Implementation Using W5500^{blaze_w5500_Exam *.xpr}

➤ Export Hardware Platform

❖ Include bitstream 을 선택하고 Next



Export Hardware Platform

Output

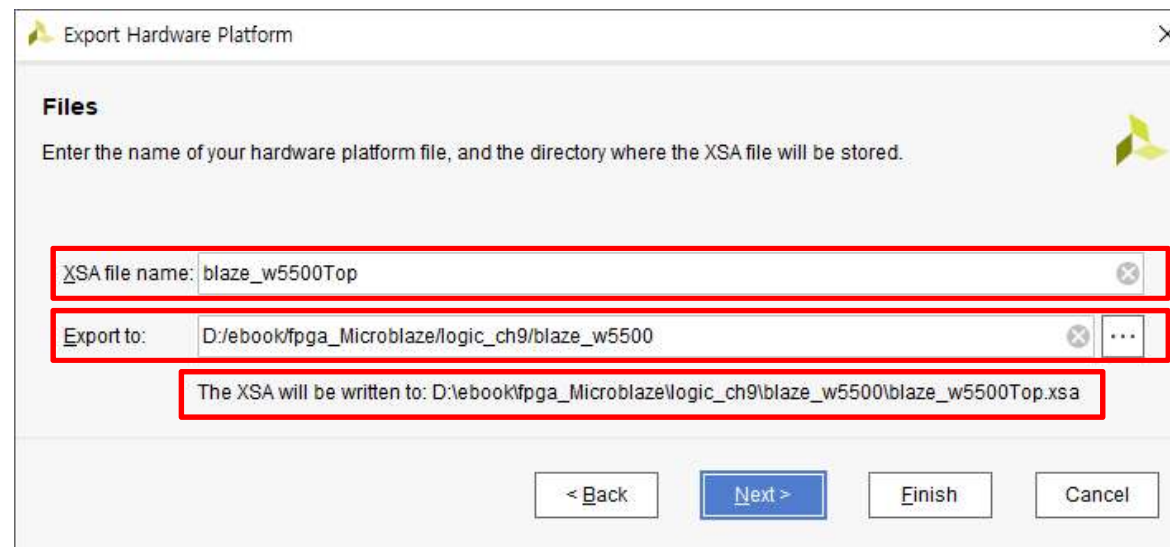
Set the platform properties to inform downstream tools of the intended use of the target platform's hardware design.

☐ Pre-synthesis
This platform includes a hardware specification for downstream software tools.

☒ Include bitstream
This platform includes the complete hardware implementation and bitstream, in addition to the hardware specification for software tools.

< Back Next > Finish Cancel

❖ 파일명, 위치를 확인하고 Next를 클릭



Export Hardware Platform

Files

Enter the name of your hardware platform file, and the directory where the XSA file will be stored.

XSA file name: blaze_w5500Top

Export to: D:\ebook\fpga_Microblaze\logic_ch9\blaze_w5500

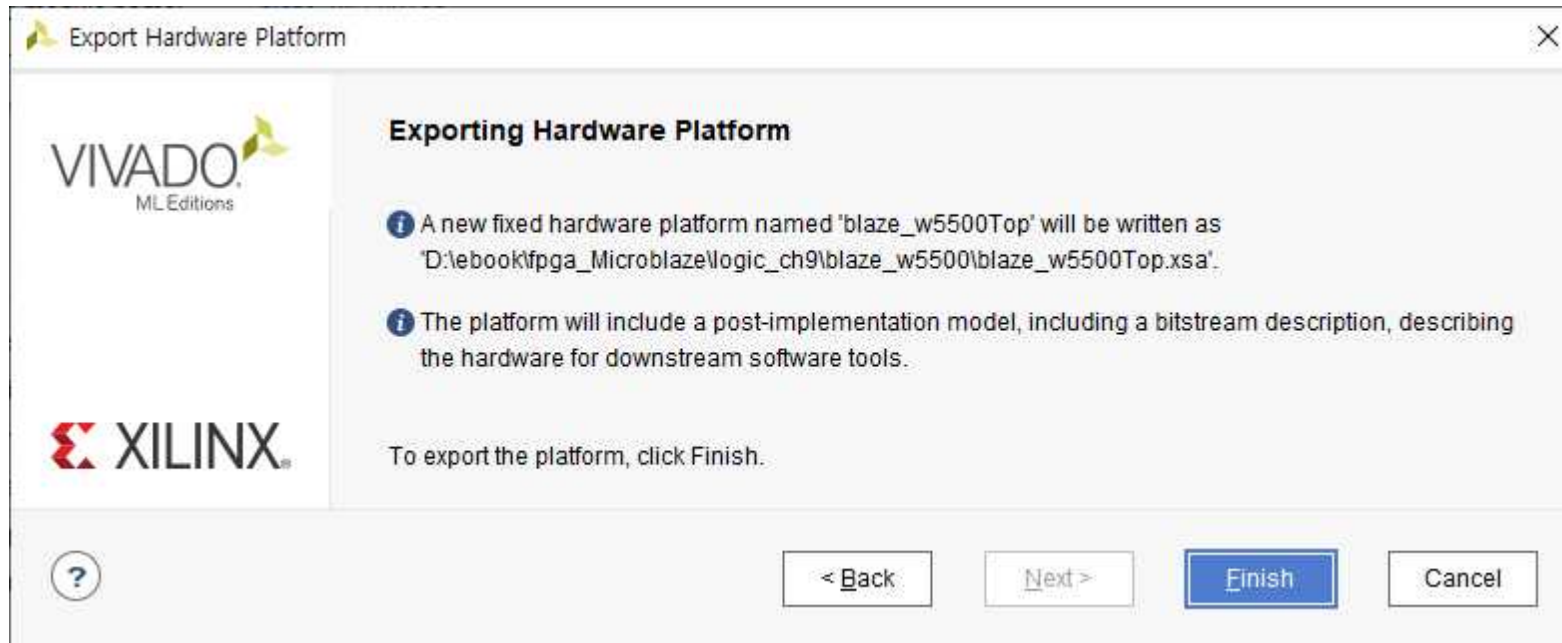
The XSA will be written to: D:\ebook\fpga_Microblaze\logic_ch9\blaze_w5500\blaze_w5500Top.xsa

< Back Next > Finish Cancel

TCP/IP Implementation Using W5500^{blaze_w5500_Exam *.xpr}

➤ Export Hardware Platform

- ❖ 마지막으로 Finish를 클릭하면 xsa 파일이 생성



- ❖ [Next] Vitis 에서 SW를 구현

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Application SW Implementation

❖ Vitis을 이용하여 **Application SW**를 구현 → 먼저 **w5500의 SPI 타이밍** 검토

- w5500에서 지원하는 SPI Mode는 0 과 3
- **Mode 0**는 **Idle** 상태에서 **SCLK**은 **Low**이고, **SCLK Falling Edge** 에서 데이터(**MISO/MOSI**)가 변함
- 데이터를 읽을 때에는 **SCLK Rising Edge**에서 읽음

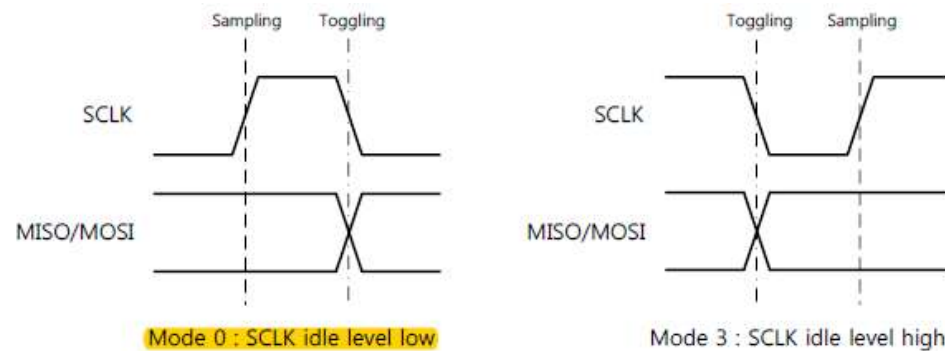


Figure 6. SPI Mode 0 & 3

❖ 아래 그림은 **SPI Frame Format**

- **Address Phase(16bits), Control Phase(8bits), Data Phase (8bits x N)**으로 구성되고 MSB 부터 전송

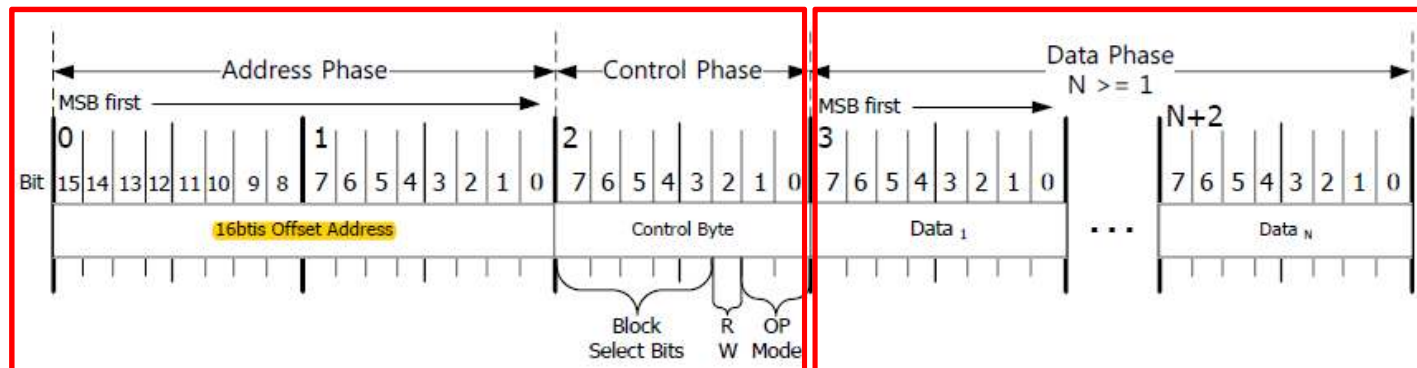
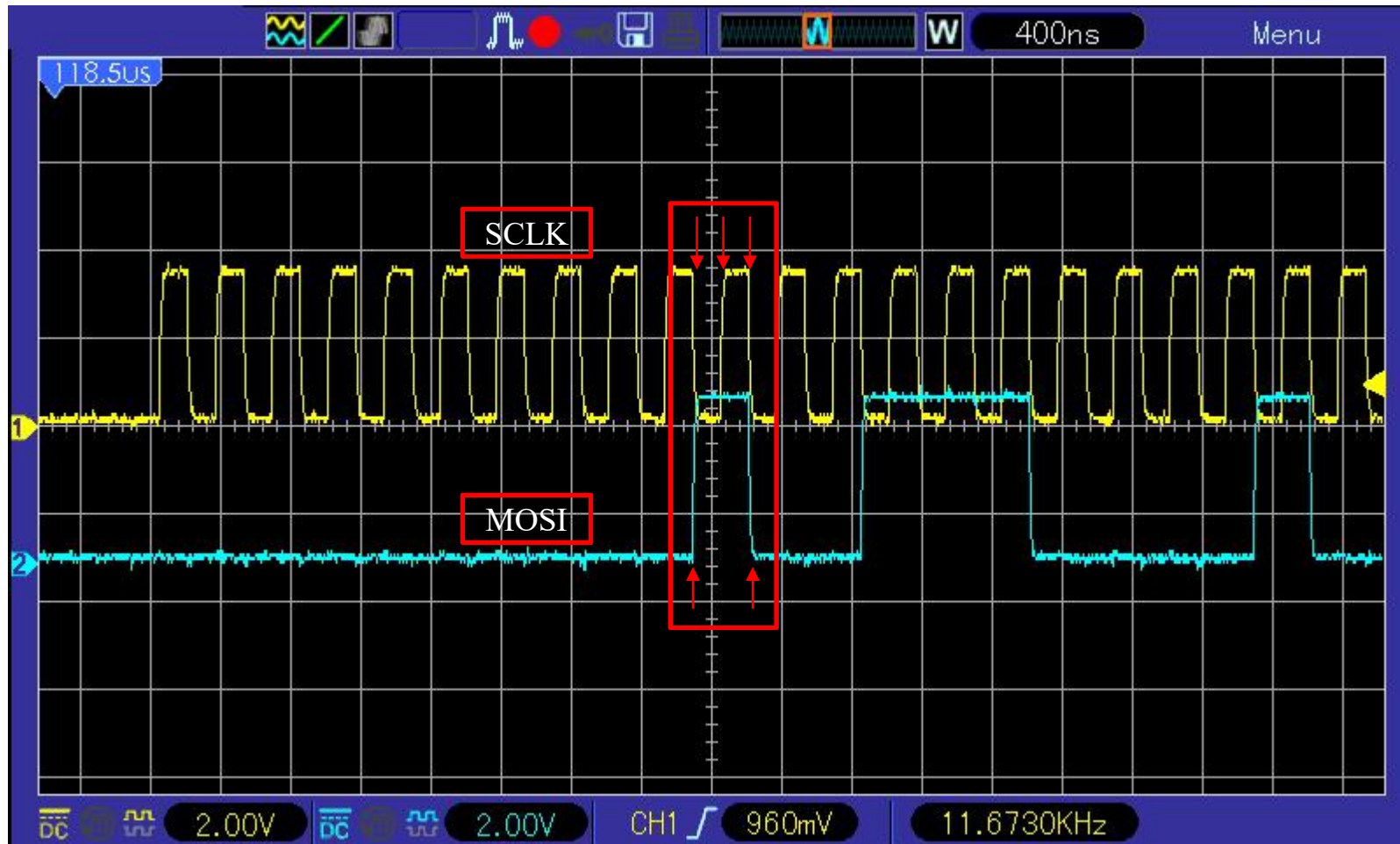


Figure 7. SPI Frame Format

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Application SW Implementation

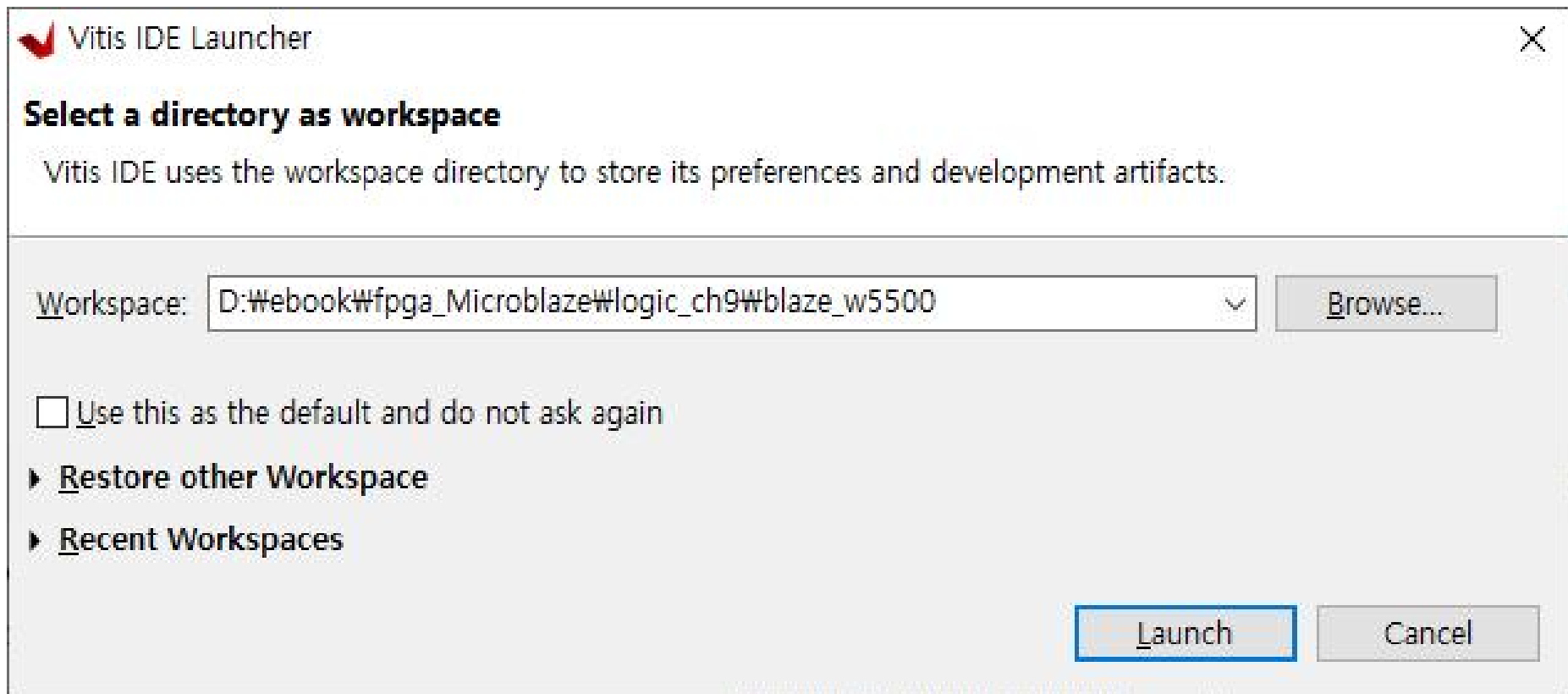
- ❖ AXI SPI 파형 → *Xilinx*사에서 제공하는 *user manual (AXI Quad SPI v3.2)*과 차이가 있음
- ❖ 아래 파형은 SW 구현후에 PC와의 데이터 통신을 하는 파형을 오실로스코프로 측정 → **Falling Edge**에서 데이터가 변하고, **Rising Edge**에서 데이터 Read (채널 1 : SCLK, 채널 2 : MOSI 임)



TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Application SW Implementation

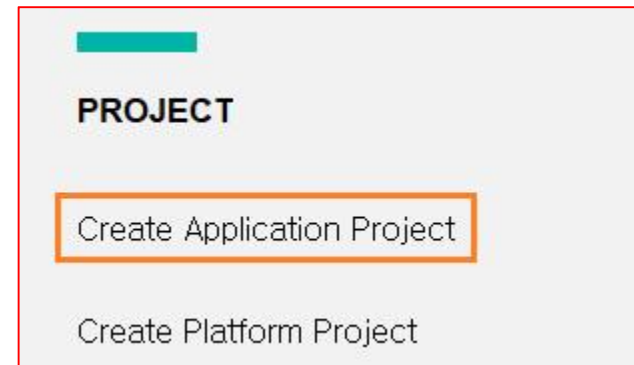
- ❖ Vitis를 실행하기 위하여 **Tools – Launch Vitis IDE** 를 클릭 ➔ 프로젝트 위치를 확인하고 **Launch** 버튼을 클릭하면 Vitis 가 실행



TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

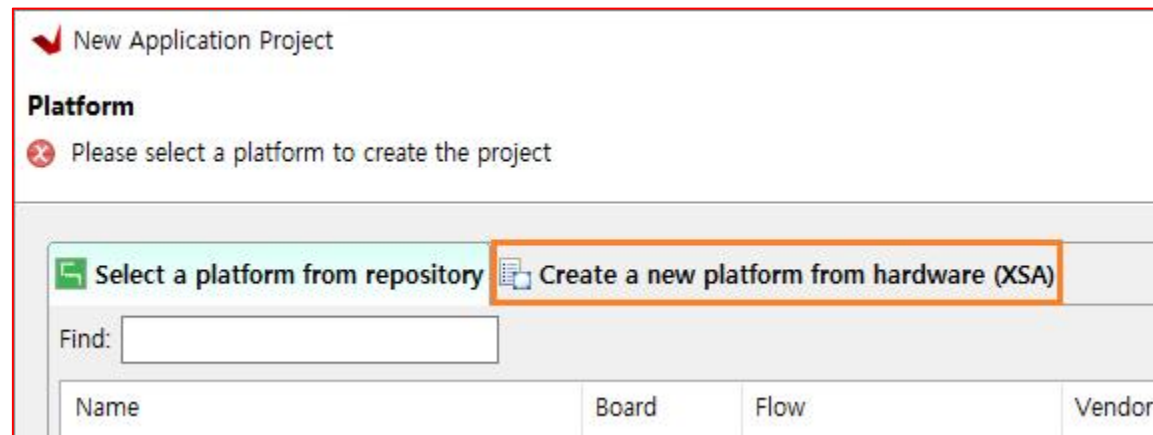
➤ Application SW Implementation

❖ PROJECT – Create Application Project 클릭



❖ Create a New Application Project 윈도우에서 Next를 클릭

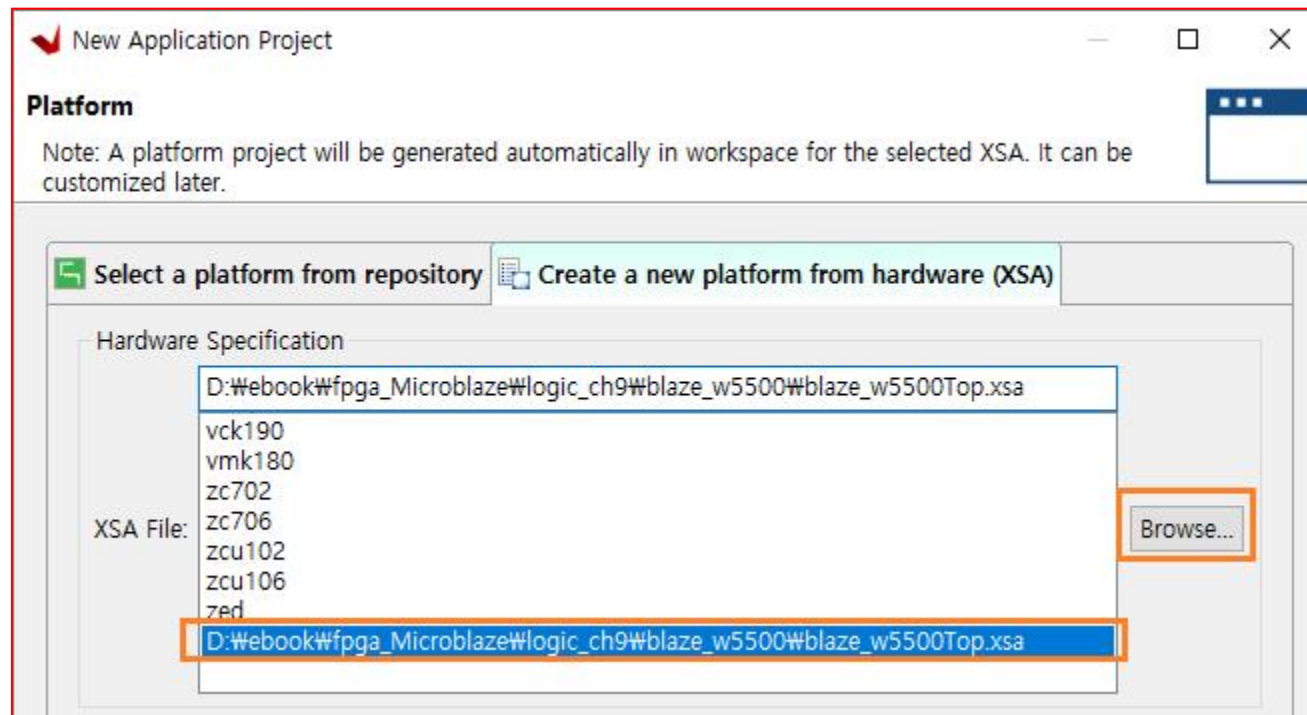
❖ Platform 윈도우에서 Create a new platform from hardware (XSA)를 클릭



TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Application SW Implementation

- ❖ **XSA File** : 우측의 Browse... 를 클릭하여 생성한 blaze_w5500.xsa 파일을 선택 ➔ Next 버튼 클릭



TCP/IP Implementation Using W5500^{blaze_w5500_Exam *.xpr}

➤ Application SW Implementation

❖ Application project name 에 “blaze_w5500App”을 입력 → Next를 클릭

New Application Project

Application Project Details
Specify the application project name and its system project properties

Application project name: blaze_w5500App

System Project
Create a new system project for the application or select an existing one from the workspace

Select a system project
+ Create new...

System project details

System project name: blaze_w5500App_system

Target processor

Select target processor for the Application project.

Processor	Associated applications
microblaze_0	blaze_w5500App

Show all processors in the hardware specification: ☒

< Back Next > Finish Cancel

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Application SW Implementation

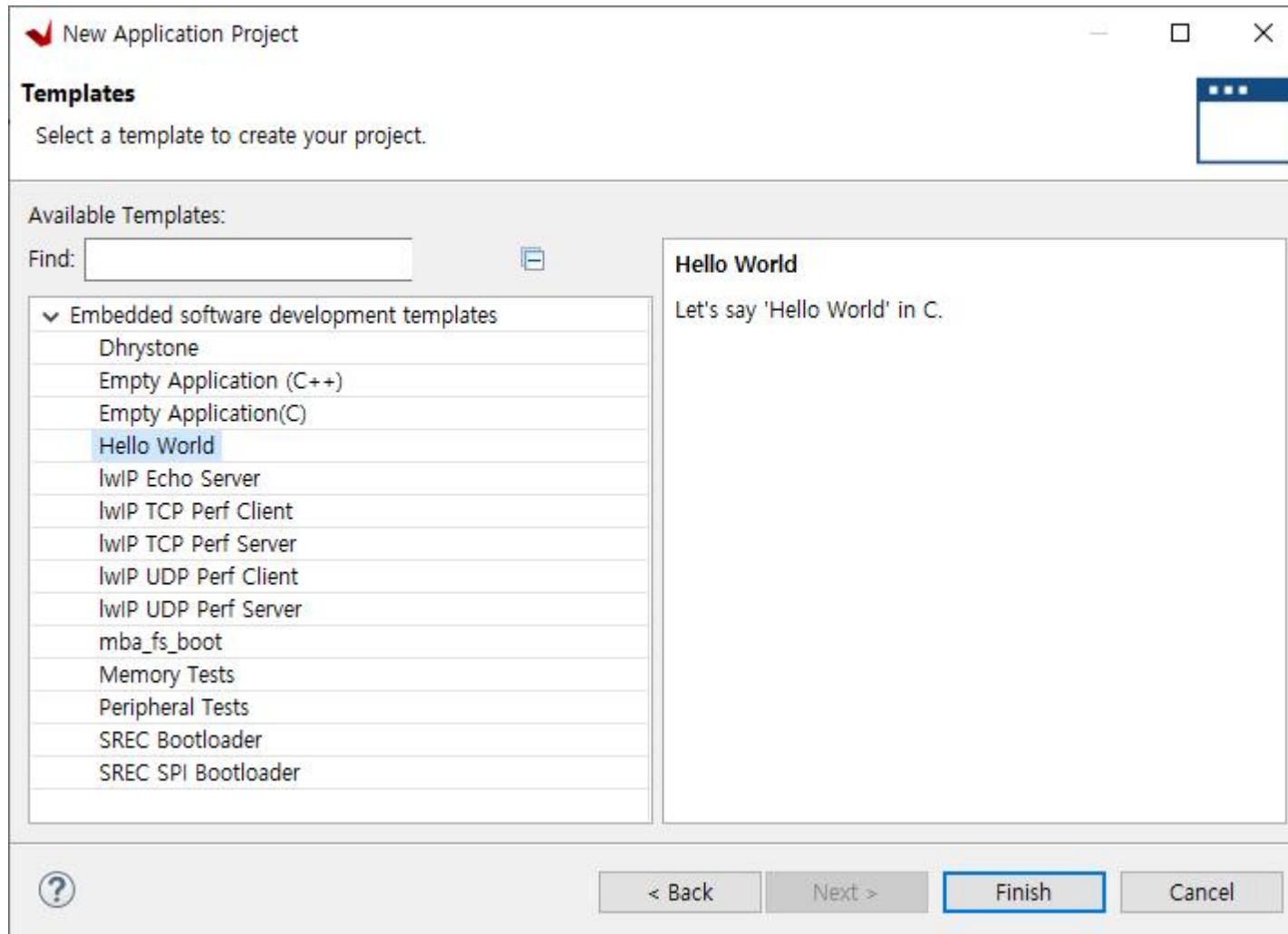
❖ Next 를 클릭

The screenshot shows the 'New Application Project' wizard window, specifically the 'Domain' step. The window title is 'New Application Project'. The 'Domain' section has a subtitle 'Select a domain for your project or create a new domain'. Below this, it says 'Select the domain that the application would link to or create a new domain' and includes a note: 'Note: New domain created by this wizard will have all the requirements of the application template selected in the next step'. On the left, there is a box with 'Select a domain' and a '+ Create new...' button. On the right, the 'Domain details' section shows the following fields: 'Name' (standalone_microblaze_0), 'Display Name' (standalone_microblaze_0), 'Operating System' (standalone), 'Processor' (microblaze_0), and 'Architecture' (32-bit). At the bottom, there are buttons for '< Back', 'Next >', 'Finish', and 'Cancel'. The 'Next >' button is highlighted with a blue border.

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Application SW Implementation

❖ Template 에서 Hello World를 선택 ➔ Finish를 클릭



TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

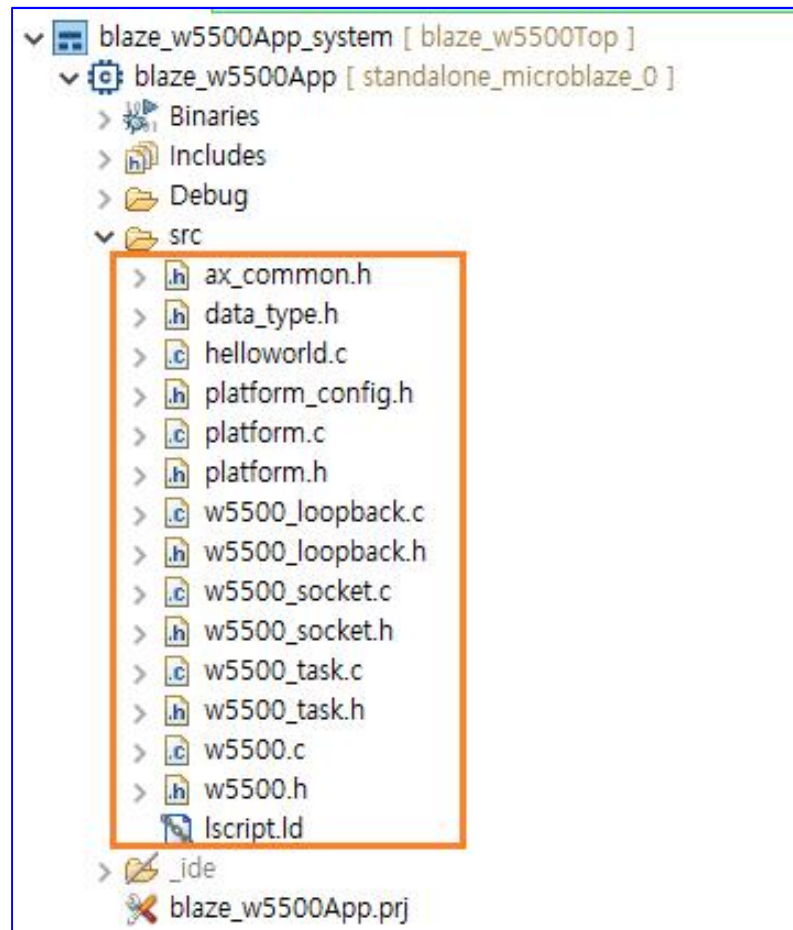
➤ Application SW Implementation

- ❖ 프로젝트 생성 → [Next] wiznet사에서 제공하는 API를 이용하여 w5500 tcp/ip를 구현할 예정
- ❖ API는 “W5500_EVB-master.zip” 파일을 다운로드 [필요한 파일은 아래와 같음]
 - ✓ `loopback.c, loopback.h` : `W5500_EVB-master\ioLibrary\Appmod\Loopback\` 폴더
 - ✓ `socket.c, socket.h` : `W5500_EVB-master\ioLibrary\Ethernet\` 폴더
 - ✓ `wizchip_conf.c, wizchip_conf.h` : `W5500_EVB-master\ioLibrary\Ethernet\` 폴더
 - ✓ `w5500.c, w5500.h` : `W5500_EVB-master\ioLibrary\Ethernet\W5500\` 폴더
- ❖ w5500 API를 포팅하는 것은 인터넷에서 자료를 찾아서 이해에 참고

TCP/IP Implementation Using W5500^{blaze_w5500_Exam *.xpr}

➤ Source File Structure

❖ 파일 구조 보기



➔ *platform_config.h, platform.c, platform.h* 파일은 기본적으로 생성되는 파일임 ➔ 수정 불필요

➔ *helloworld.c* 안에 *main()* 함수가 있음 ➔ 수정 및 구현

TCP/IP Implementation Using W5500^{blaze_w5500_Exam *.xpr}

➤ Application SW Implementation

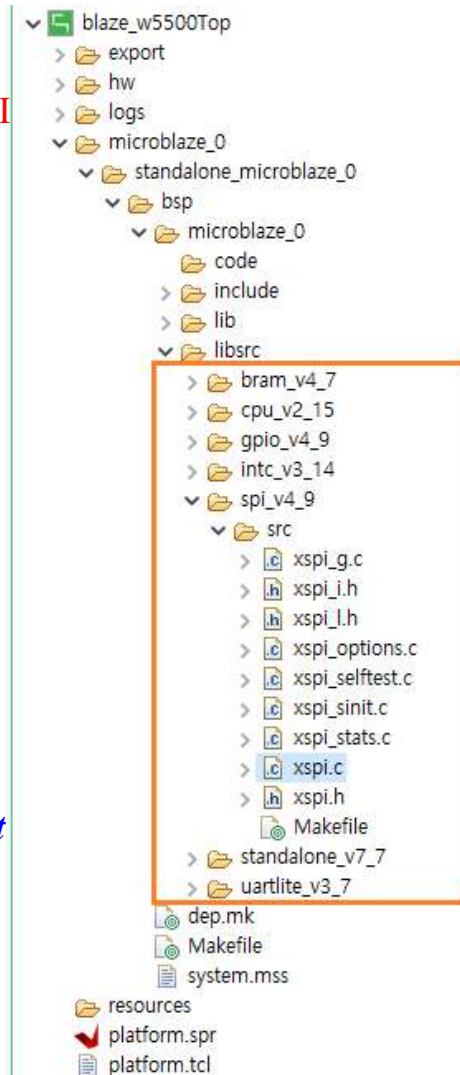
- ❖ w5500의 API를 포팅하기 위해서는 상위 API들은 그대로 활용하고, 하위 API를 사용자의 시스템에 맞게 변경해 주어야함. 특별히 SPI 인터페이스 API를 사용자의 시스템에 맞게 변경해 주면 됨

➔ SPI 관련 API 들은 다음의 4가지가 있음 (w5500.c 파일내에 있음)

- ✓ `uint8_t WIZCHIP_READ(uint32_t AddrSel) : 1-바이트 read`
- ✓ `void WIZCHIP_WRITE(uint32_t AddrSel, uint8_t wb) : 1-바이트 write`
- ✓ `void WIZCHIP_READ_BUF(uint32_t AddrSel, uint8_t* pBuf, uint16_t len) : n-바이트 read`
- ✓ `void WIZCHIP_WRITE_BUF(uint32_t AddrSel, uint8_t* pBuf, uint16_t len) : n-바이트 write`

➔ 위의 4개의 함수들을 우리가 사용하는 “Xspi_Transfer” 함수로 변환해주면 됨

➔ Xspi_Transfer 함수는 xspi.c 안에 정의되어 있음 : 파일 위치는 그림과 같음



TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Application SW Implementation → xspi.c

❖ 아래는 *Xspi_Transfer* 함수를 보여줌

```
525 int Xspi_Transfer(Xspi *InstancePtr, u8 *SendBufPtr,  
526                  u8 *RecvBufPtr, unsigned int ByteCount)  
527 {
```

- ✓ *InstancePtr* : Xspi instance pointer
- ✓ *SendBufPtr* : 전송할 데이터의 시작 주소
- ✓ *RecvBufPtr* : 수신한 데이터를 저장할 변수의 시작 주소, *Write*시에는 *NULL*로 지정
- ✓ *ByteCount* : 전송할 데이터의 바이트 수

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Application SW Implementation ➔ w5500 spi frame

❖ 아래 그림은 w5500 spi frame 구조를 보여줌

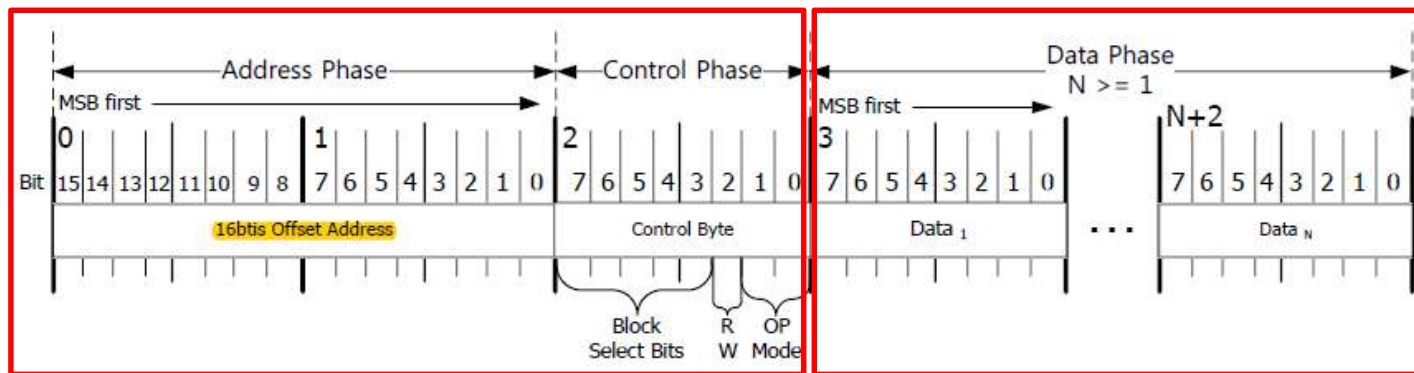


Figure 7. SPI Frame Format

❖ w5500에 1-바이트 Data를 전송하기 위해서는 총 4바이트(Address Phase : 2바이트 + Control Phase : 1바이트 + Data Phase : 1바이트)를 전송 ➔ w5500에 N-바이트 데이터를 전송하기 위해서는 총 N+3 바이트를 전송 ➔ 따라서 write시에 ByteCount 값은 “전송할 데이터 사이즈 + 3”

❖ w5500에서 read 하는 경우는 더욱 복잡

- Xspi_Transfer 함수는 데이터를 전송할 때, miso 핀으로 들어오는 데이터를 처음부터 read
- w5500의 spi read 는 Address Phase, Control Phase를 지나고 Data Phase 에 miso 핀으로 들어오는 데이터를 read data로 인식하는데, Xspi_Transfer 함수는 Address Phase부터 miso 핀으로 들어오는 데이터를 read data로 인식 ➔ 따라서 Xspi_Transfer 함수의 read 부분을 수정할 필요 ➔ 즉 Address Phase, Control Phase의 read data는 버리고, Data Phase에서 수신되는 데이터만 RecvBufPtr에 저장

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Application SW Implementation → xspi.c

❖ Xspi_Transfer 함수를 아래와 같이 수정해야 함

```
525 int Xspi_Transfer(XSpi *InstancePtr, u8 *SendBufPtr,
526                  u8 *RecvBufPtr, unsigned int ByteCount)
527 {
528     u32 ControlReg;
529     u32 GlobalIntrReg;
530     u32 StatusReg;
531     u32 Data = 0;
532     u8 DataWidth;
533     u32 DataLen;
534     u32 Index;
535     u32 rcv_index=0;
536
```

- 535 : 수신된 데이터를 count 하기 위한 변수 생성

```
716 while ((StatusReg & XSP_SR_RX_EMPTY_MASK) == 0) {
717
718     Data = Xspi_ReadReg(InstancePtr->BaseAddr,
719                       XSP_DRR_OFFSET);
720     if (DataWidth == XSP_DATAWIDTH_BYTE) {
721         /*
722          * Data Transfer Width is Byte (8 bit).
723          */
724         if(InstancePtr->RecvBufferPtr != NULL) {
725             rcv_index++;
726             if(rcv_index>3)
727             {
728                 *InstancePtr->RecvBufferPtr++ = (u8)Data;
729             }
730
```

- 725 – 729 : 수신된 데이터의 처음 3-바이트는 버리고 4번째 바이트부터 저장함

TCP/IP Implementation Using W5500^{blaze_w5500_Exam *.xpr}

➤ Application SW Implementation → *xspi.c*

❖ *Xspi_Transfer* 함수 수정 사항

```
599  /*  
600   * Set up buffer pointers.  
601   */  
602   InstancePtr->SendBufferPtr = SendBufPtr;  
603   InstancePtr->RecvBufferPtr = RecvBufPtr;  
604  
605   InstancePtr->RequestedBytes = ByteCount;  
606   InstancePtr->RemainingBytes = ByteCount;
```

- 603 : *InstancePtr* → *RecvBufferPtr* = *RecvBufPtr*, 두 포인터 변수가 동일함

❖ [주의할 점]

- *xspi.c* 는 bsp 폴더 안에 있음 → 즉 Vivado 에서 IP를 생성할 때 자동으로 생성되는 파일 → 따라서 Vivado에서 Design을 수정후에, Vitis에서 업데이트된 내용을 적용하기 위하여 “Update Hardware Specification”을 클릭하면 수정했던 코드(*xspi.c*)가 모두 없어짐 → 따라서, *xspi.c* 파일을 수정후에 는 반드시 다른 폴더에 저장해 둔 후에 다시 복사해 주어야함!!

- 이 과정의 시퀀스

- 1) vivado 에서 design 수정 (bitstream, xsa 파일 생성)
- 2) blaze_w5500Top → 우클릭 → Update Hardware Specification 클릭 → 업데이트 된 xsa 파일 지정
- 3) blaze_w5500Top → 우클릭 → Clean Project
- 4) blaze_w5500Top → 우클릭 → Build Project
- 5) 백업해둔 *xspi.c* 파일 해당 폴더(bsp\microblaze_0\libsrc\spi_v4_9\src)로 복사
- 6) 다시 blaze_w5500Top → 우클릭 → Build Project

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Application SW Implementation

➔ w5500.c

- ❖ w5500_write / w5500_read 함수 구현 함(w5500.c)

```
28 uint8_t tcpRxBuffer[SPI_FIFO_SIZE];
29 uint8_t tcpTxBuffer[SPI_FIFO_SIZE];
30
31 void w5500_write(uint16_t addr16, uint8_t bsb, uint16_t data_len, uint8_t *dat8)
32 {
33     uint16_t i;
34
35     tcpTxBuffer[0] = (uint8_t)(addr16>>8);
36     tcpTxBuffer[1] = (uint8_t)(addr16&0xff);
37     tcpTxBuffer[2] = (bsb<<3) | (1<<2);
38     for(i=0; i<data_len; i++) tcpTxBuffer[i+3] = dat8[i];
39
40     XSpi_Transfer(&SpiInstance, tcpTxBuffer, NULL, data_len+3);
41     //xil_printf("spi wrtie ..\r\n");
42 }
43
44 void w5500_read(uint16_t addr16, uint8_t bsb, uint16_t data_len, uint8_t *rbuf)
45 {
46     tcpTxBuffer[0] = (uint8_t)(addr16>>8);
47     tcpTxBuffer[1] = (uint8_t)(addr16&0xff);
48     tcpTxBuffer[2] = (bsb<<3);
49
50     XSpi_Transfer(&SpiInstance, tcpTxBuffer, rbuf, data_len+3);
51     //xil_printf("spi read ..\r\n");
52 }
```

- 28 – 29 : tx, rx에 사용할 버퍼를 정의 ➔ SPI_FIFO_SIZE는 256 ➔ Vivado 에서 AXI SPI를 설정시 FIFO를 256으로 설정 ➔ 한번에 최대 256바이트까지 전송할 수 있음
- 31 – 42 : w5500_wrtie 함수를 구현 ➔ addr16: Address, bsp: control, data_len: 전송할 메시지의 바이트 수, Xspi_Transfer에서 전송할 바이트 수는 “data_len + 3” 됨, write 시에는 Xspi_Transfer 함수의 3번째 인자는 NULL로 설정함
- 44 – 52 : w5500_read 함수를 구현 ➔ addr16은 Address, bsp는 control, data_len은 read할 데이터의 바이트 수, Xspi_Transfer에서 전송할 바이트 수는 “data_len + 3”이 됨

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Application SW Implementation → w5500.c

❖ 다음으로 이 함수를 이용하여 **WIZCHIP_READ ~ WIZCHIP_WRITE_BUF** 함수를 구현함

```
54 void WIZCHIP_WRITE(uint32_t AddrSel, uint8_t wb )
55 {
56     w5500_write( (uint16_t)((AddrSel>>8)&0xffff), (uint8_t)((AddrSel>>3)&0x1f), 1, &wb);
57 }
58
59 void WIZCHIP_WRITE_BUF(uint32_t AddrSel, uint8_t* pBuf, uint16_t len)
60 {
61     w5500_write( (uint16_t)((AddrSel>>8)&0xffff), (uint8_t)((AddrSel>>3)&0x1f), len, pBuf);
62 }
63
64 uint8_t WIZCHIP_READ(uint32_t AddrSel)
65 {
66     uint8_t rbuf[6];
67
68     w5500_read( (uint16_t)((AddrSel>>8)&0xffff), (uint8_t)((AddrSel>>3)&0x1f), 1, rbuf);
69
70     return(rbuf[0]);
71 }
72
73 void WIZCHIP_READ_BUF (uint32_t AddrSel, uint8_t* pBuf, uint16_t len)
74 {
75     w5500_read( (uint16_t)((AddrSel>>8)&0xffff), (uint8_t)((AddrSel>>3)&0x1f), len, pBuf);
76 }
```

- 54 – 57 : **WIZCHIP_WRITE** 함수를 구현 → 1-바이트를 write
- 59 – 62 : **WIZCHIP_WRITE_BUF** 함수를 구현 → N-바이트를 write
- 64 – 71 : **WIZCHIP_READ** 함수를 구현 → 1-바이트를 read
- 73 – 76 : **WIZCHIP_READ_BUF** 함수를 구현 → N-바이트를 read

❖ **w5500 API**들은 모두 **WIZCHIP_WRITE ~ WIZCHIP_READ_BUF**를 사용해서 구현되었기 때문에 w5500 API를 그대로 사용할 수 있음

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ Memory Size

- ❖ **Linker Script (lscript.ld)** 파일을 클릭하면,
 - ➔ **Stack, Heap Size** 확인할 수 있음
 - ➔ **stack size : 0x400 (1KB) = dec 1024** 로 설정됨
 - ➔ **Heap size : 0x800 (2KB) = dec 2048** 로 설정됨

➔ stack and heap memory 참고: <https://blog.naver.com/PostView.nhn?blogId=techref&logNo=222274484731>

- 전역변수로 사용가능한 **Memory**는 **2KB**임 ,또한 **AXI SPI**는 최대 **256** 바이트의 **FIFO**를 가지고 있음
- AXI SPI로 한 번에 전송할 수 있는 바이트는 256 바이트임 ➔ **그러나 w5500은 data를 전송하기 전에 3바이트(Address 2바이트, Control 1바이트)을 전송하기 때문에 253바이트까지 전송 가능 ➔ 여기서** 한 번에 최대 240바이트를 전송
- **PC와 Network 으로 연결 후, TCP/IP 통신에서는 1200 바이트(240바이트 x 5)를 송수신하는 것을 구현 ➔ 즉 PC에서 보드로 1200바이트를 전송하면, AXI SPI는 240바이트씩 5번 수신 ➔ 보드에서 PC로 1200바이트를 전송할 때에도, AXI SPI는 240바이트씩 5번 전송 ➔ 따라서 전역변수를 아래와 같이 3개를 설정하도록 함**
 - `uint8_t tcpRxBuffer[SPI_FIFO_SIZE];` // AXI SPI 송신에 사용되는 버퍼, SPI_FIFO_SIZE = 256
 - `uint8_t tcpTxBuffer[SPI_FIFO_SIZE];` // AXI SPI 수신에 사용되는 버퍼, SPI_FIFO_SIZE = 256
 - `uint8_t tcpMsgBuffer[TCP_MSG_MAX];` // 수신 메시지를 저장하는 버퍼, TCP_MSG_MAX = 1200
- TCP/IP 송수신에 사용되는 전역변수는 총 1712 바이트: Heap Size 가 2048 사이즈이기 때문에 충분

Linker Script: lscript.ld

A linker script is used to control where different sections of an executable are placed in memory. In this page, you can define new memory regions, and change the assignment of sections to memory regions.

Available Memory Regions

Name	Base Address	Size
microblaze_0_local_memory_ilmb_bram_if_cntlr_Mem_microb...	0x50	0xFFB0

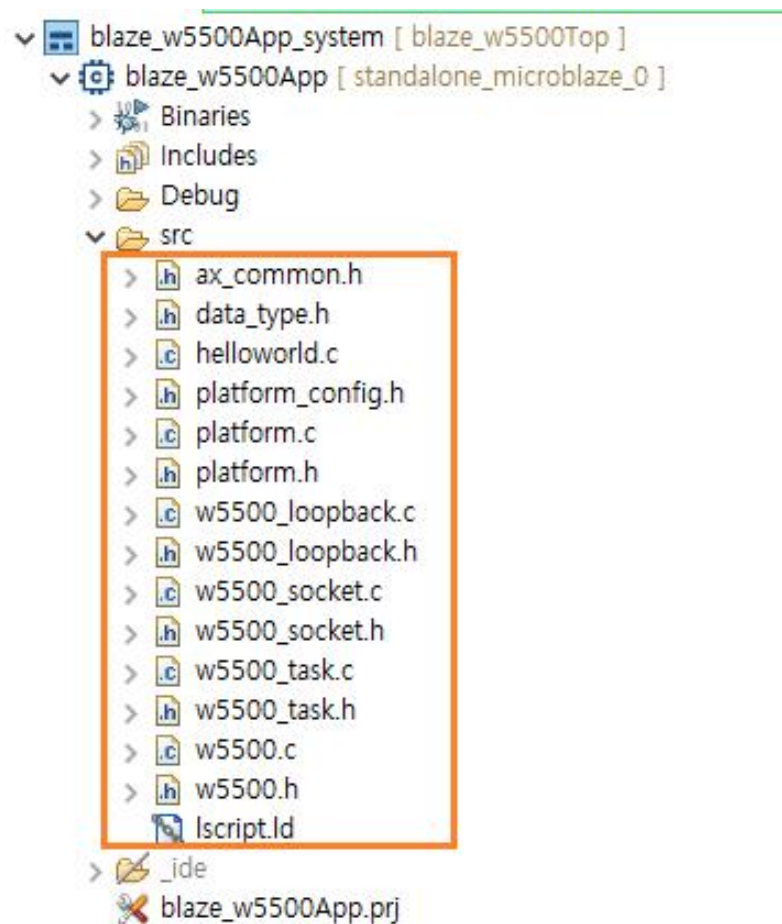
Stack and Heap Sizes

Stack Size	<input type="text" value="0x400"/>
Heap Size	<input type="text" value="0x800"/>

TCP/IP Implementation Using W5500^{blaze_w5500_Exam *.xpr}

➤ Source File Structure

❖ 파일 구조 보기



➔ *platform_config.h, platform.c, platform.h* 파일은 기본적으로 생성되는 파일임 ➔ 수정 불필요

➔ *helloworld.c* 안에 *main()* 함수가 있음 ➔ 수정 및 구현

TCP/IP Implementation Using W5500^{blaze_w5500_Exam *.xpr}

➤ **data_type.h** 파일 구조

❖ 변수 타입 정의되어 있음

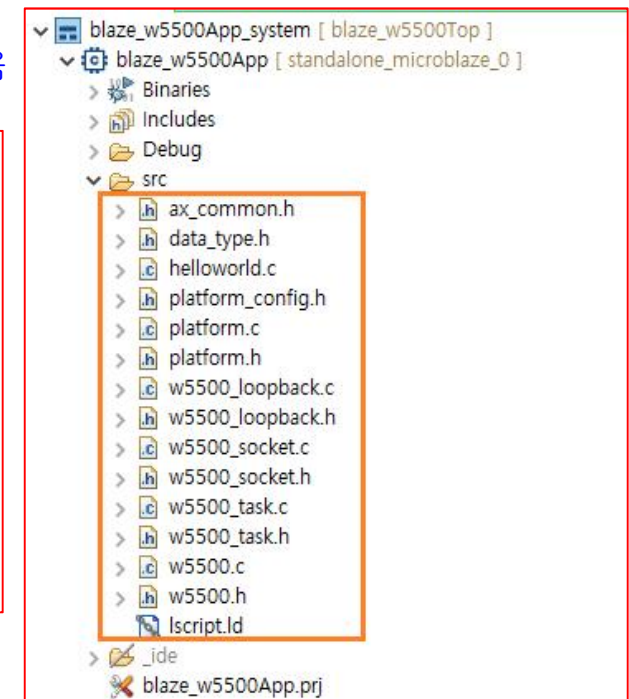
➔ 즐겨 사용하는 **uint8_t, uint16_t, uint32_t** 를 정의함

```
5 typedef signed short int16;
6 typedef signed long long32;
7 typedef unsigned char uchar8;
8 typedef unsigned short uint16;
9 typedef unsigned long ulong32;
10
11 #define uint8_t uchar8
12 #define uint16_t uint16
13 #define uint32_t ulong32
```

➤ **ax_common.h**

❖ 각 **IP들의 ID, TCP/IP 통신에 사용되는 버퍼 사이즈**가 정의되어 있음

```
4 /* definitions for UART */
5 #define GPIO_5BIT_DEVICE_ID XPAR_GPIO_0_DEVICE_ID
6 #define GPIO_5BIT_CHANNEL1 1
7
8 /* definitions for UART */
9 #define UARTLITE_DEVICE_ID XPAR_UARTLITE_0_DEVICE_ID
10 #define UARTLITE_INT_IRQ_ID XPAR_INTC_0_UARTLITE_0_VEC_ID
11
12 /* definitions for SPI */
13 #define SPI_DEVICE_ID XPAR_SPI_0_DEVICE_ID
14
15 #define SPI_FIFO_SIZE 256
16 #define TCP_MSG_SIZE 240
17 #define TCP_MSG_MAX 1200
```



➤ **w5500.c, w5500.h**

❖ wiznet에서 제공하는 w5500.c, w5500.h 파일을 맞게 수정하였음 ➔ 앞에서 설명한 w5500 read/write 함수들이 Xspi_Transfer() 함수로 구현됨

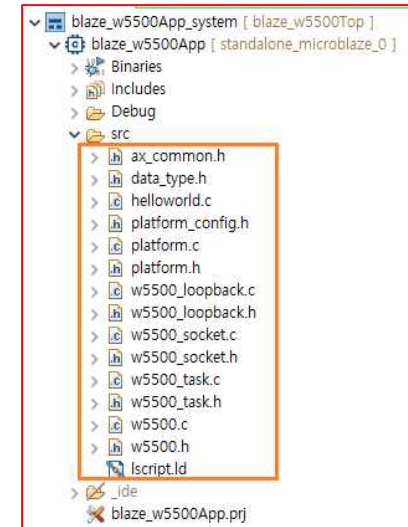
TCP/IP Implementation Using W5500^{blaze_w5500_Exam *.xpr}

➤ w5500_task.c, w5500_task.h

❖ wiznet에서 제공하는 **wizchip_conf.c, wizchip_conf.h** 파일에서 필요한 부분만 가져와서 수정하였음 ➔ 주

로 w5500의 초기화와 관련된 내용들이 포함되어 있음

- ✓ **w5500_reset_HW()** : w5500 **RESET** 핀을 이용한 **reset, gpio[0]** 를 사용
- ✓ **w5500_sw_reset()** : w5500 register들을 초기화 하기 위한 **sw reset**.
- ✓ **w5500_init()** : w5500에는 **TX, RX** 각각 8개의 Socket을 지원 ➔ 각각의 Socket은 내부 메모리를 최대 **32KB**까지 설정 ➔ 각각의 Socket의 메모리를 설정
- ✓ **w5500_getphylink()** : **Network Link**을 Check



➤ w5500_socket.c, w5500_socket.h

❖ wiznet에서 제공하는 **socket.c, socket.h** 파일을 본 예제에 맞게 수정하였음 ➔ 주로 socket에 관련된 API가 들어 있음

- ✓ **socket()** : **socket**을 open ➔ 본 예제에서는 **w5500 보드가 server**가 되고, **PC의 응용 프로그램이 client**가 됨 ➔ 전원이 인가되면, **w5500은 network link를 확인**해서 **network link**가 존재하면 socket을 open 해서 client의 접속을 기다림
- ✓ **close()** : **socket**을 close
- ✓ **listen ()** : **client**의 접속을 기다림
- ✓ **send()** : **client**로 데이터를 전송
- ✓ **recv()** : **client**에서 전송한 데이터를 수신

TCP/IP Implementation Using W5500

➤ w5500_loopback.c, w5500_loopback.h

❖ 본 예제에서는 wiznet에서 제공하는 **loopback Test**를 이용하여 TCP/IP

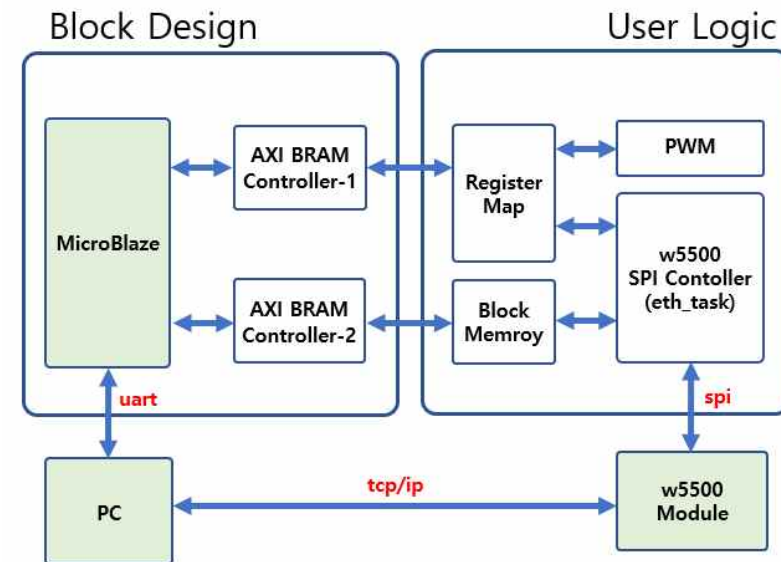
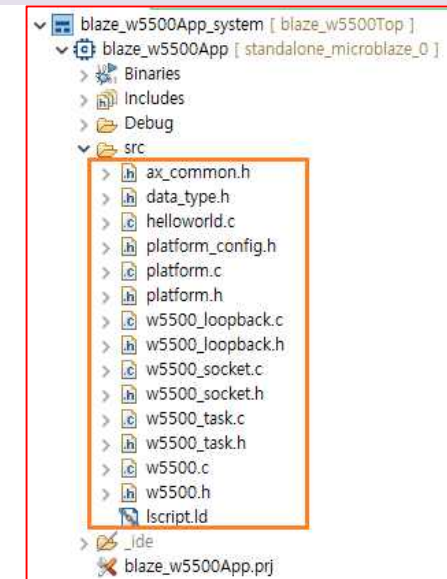
통신을 진행함 → [통신 순서]

1. 먼저 **PC에서 1200 바이트를 전송**
2. **microblaze는 w5500의 수신 상태 레지스터를 계속해서 확인**해서 수신한 데이터가 있으면 **수신한 데이터를 읽음**
3. **1200 바이트를 수신하면**, 수신한 데이터를 **UART로 뿌려서 전송한 데이터가 맞는지 확인**함[WinIDT]
4. 그리고 **PC로 다시 1200바이트를 전송**함

```

19 // TCP & UDP Loopback Test
20 int32_t loopback_tcps(uint8_t sn, uint8_t* buf, uint16_t port)
21 {
22     int32_t ret;
23     uint16_t size = 0;
24     uint8_t destip[4];
25     uint16_t destport;
26     int i, j;
27
28     switch(getSn_SR(sn))
29     {
30         // -----
31         // SOCK_ESTABLISHED
32         case SOCK_ESTABLISHED :
33             if(getSn_IR(sn) & Sn_IR_CON)
34             {
35                 getSn_DIPR(sn, destip);
36                 destport = getSn_DPORT(sn);
37                 xil_printf("\r\n Connected - %d.%d.%d.%d %d \r\n",
38                     destip[0], destip[1], destip[2], destip[3], destport);
39                 setSn_IR(sn, Sn_IR_CON);
40             }

```



▪ 32 – 40 : socket이 연결되었을 때의 상태 & 연결된 client의 정보

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ w5500_loopback.c, w5500_loopback.h

- 41 : 수신된 데이터를 확인하여 수신한 데이터가 있을 때 사이즈를 리턴 함
- 43 – 68 : AXI SPI가 한번에 읽을 수 있는 데이터가 최대 253 (256-3) → 본 예제에서는 240바이트씩 읽도록 구현됨 :
TCP_MSG_SIZE = 240
- 70 – 89 : PC로 부터 1200 바이트를 수신하게 되면, 수신한 데이터를 UART를 통해 화면에 보여줌 → PC에서 송신한 데이터는 “A~X”, “a~x”의 데이터가 반복해서 구성됨 (48바이트씩 25번, 48 x 25 = 1200) PC에서는 1200바이트를 전송함 → 또한 수신한 데이터를 다

시 PC로 전송함 : 240바이트씩 5번 전송함

```
41      if((size = getSn_RX_RSR(sn)) > 0)
42      {
43          xil_printf(" W5500 Received size : %d \r\n", size);
44          tcpRxTotal += size;
45          while(1)
46          {
47              if(size > TCP_MSG_SIZE)
48              {
49                  memset(buf, 0, SPI_FIFO_SIZE);
50                  ret = recv(sn, buf, TCP_MSG_SIZE);
51                  if(ret <= 0) return ret;
52                  size -= TCP_MSG_SIZE;
53                  xil_printf(" BD Received size : %d \r\n", TCP_MSG_SIZE);
54                  memcpy(tcpMsgBuffer+tcpMsgIndex, buf, TCP_MSG_SIZE);
55                  tcpMsgIndex += TCP_MSG_SIZE;
56              }
57              else
58              {
59                  memset(buf, 0, SPI_FIFO_SIZE);
60                  ret = recv(sn, buf, size);
61                  if(ret <= 0) return ret;
62                  xil_printf(" BD Received size : %d \r\n", size);
63                  memcpy(tcpMsgBuffer+tcpMsgIndex, buf, TCP_MSG_SIZE);
64                  tcpMsgIndex += size;
65              }
66              break;
67          }
68      }
69
70      if(tcpRxTotal >= TCP_MSG_MAX)
71      {
72          tcpRxTotal = 0;
73          tcpMsgIndex = 0;
74
75          for(i=0; i<(TCP_MSG_MAX/48); i++)
76          {
77              xil_printf("B : %d ", i);
78              for(j=0; j<48; j++) xil_printf("%c", tcpMsgBuffer[i*48+j]);
79              xil_printf("\r\n");
80          }
81
82          send(SOCK_TCPS, tcpMsgBuffer, TCP_MSG_SIZE);
83          send(SOCK_TCPS, tcpMsgBuffer+TCP_MSG_SIZE, TCP_MSG_SIZE);
84          send(SOCK_TCPS, tcpMsgBuffer+2*TCP_MSG_SIZE, TCP_MSG_SIZE);
85          send(SOCK_TCPS, tcpMsgBuffer+3*TCP_MSG_SIZE, TCP_MSG_SIZE);
86          send(SOCK_TCPS, tcpMsgBuffer+4*TCP_MSG_SIZE, TCP_MSG_SIZE);
87      }
88      break;
89  }
```

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ w5500_loopback.c, w5500_loopback.h

- 92 – 95 : *socket*이 close 될 때, *SOCK_CLOSE_WAIT*, *SOCK_CLOSED* 상태가 발생
- 98 – 101 : *socket*이 생성될 때 발생
- *Socket Status Register*에 대한 자세한 사항은 w5500 데이터 시트의 48 페이지를 참조

```
90      // -----
91      // SOCK_CLOSE_WAIT
92      case SOCK_CLOSE_WAIT :
93          if((ret=disconnect(sn)) != SOCK_OK) return ret;
94          xil_printf("Socket closed .. \r\n");
95          break;
96      // -----
97      // SOCK_INIT
98      case SOCK_INIT :
99          xil_printf("Listen, TCP server loopback, port : %d \r\n", port);
100         if( (ret = listen(sn)) != SOCK_OK) return ret;
101         break;
102     // -----
103     // SOCK_CLOSED
104     case SOCK_CLOSED:
105         xil_printf("TCP server loopback start .. \r\n");
106         if((ret=socket(sn, Sn_MR_TCP, port, 0x00)) != sn)
107             return ret;
108         break;
109
110     default:
111         break;
```


TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ **helloworld.c**

- 62 – 64 : IP Instance 정의
- 69 – 92 : uart, gpio, spi 초기화

[Next]

```
62 XGpio    Gpio_5bits;          /* The instance of the gpio */
63 XUartLite UartLite;           /* The instance of the UartLite Device */
64 XSpi      SpiInstance;        /* The instance of the SPI device */
65
66 void Display_Net_Conf(void);
67 void Net_Conf(void);
68
69 void uart_init(void)
70 {
71     XUartLite_Initialize(&UartLite, UARTLITE_DEVICE_ID);
72     XUartLite_SelfTest(&UartLite);
73 }
74
75 void gpio_init(void)
76 {
77     XGpio_Initialize(&Gpio_5bits, GPIO_5BIT_DEVICE_ID);
78 }
79
80 void spi_init(void)
81 {
82     XSpi_Config *ConfigPtr; /* Pointer to Configuration data */
83
84     ConfigPtr = XSpi_LookupConfig(SPI_DEVICE_ID);
85     XSpi_CfgInitialize(&SpiInstance, ConfigPtr, ConfigPtr->BaseAddress);
86     XSpi_SelfTest(&SpiInstance);
87
88     XSpi_SetOptions(&SpiInstance, XSP_MASTER_OPTION );
89     XSpi_Start(&SpiInstance);          /* Start SPI */
90     XSpi_IntrGlobalDisable(&SpiInstance); /* Disable interrupt */
91     XSpi_SetSlaveSelect(&SpiInstance, 0x35); /* 이 문이 빠지면 동작하지 않음 */
92 }
```

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ **helloworld.c**

- 109 – 113 : w5500 초기화, 아래의 정보로 w5500 이 설정

```

16 wiz_NetInfo gWIZNETINFO = {
17     {0x0A, 0x15, 0x23, 0x07, 0x21, 0x01},
18     {192, 168, 1, 11},
19     {255, 255, 255, 0},
20     {192, 168, 1, 1},
21     {0, 0, 0, 0},
22     NETINFO_STATIC
23 };
    
```

← w5500.c

- 111 : Net_Conf(); w5500에 Network 정보 설정
- 113 : Display_Net_conf() : w5500에 설정된 Network 정보 display
- 117 : network link를 확인
- 120 – 127 : network link가 존재하면, w5500의 상태를 계속해서 확인

```

95 int main()
96 {
97     uint8_t phy_link=0;
98     int32_t ret;
99
100     init_platform();
101     gpio_init();
102     uart_init();
103     spi_init();
104     XGpio_DiscreteWrite(&Gpio_5bits, GPIO_5BIT_CHANNEL1, 0xf);
105
106     xil_printf("\r\n Microblaze w5500 start .. \r\n");
107
108     // W5500_Init()
109     w5500_reset_HW();           // 1) HW reset
110     w5500_init();               // 2) tx, rx buffer size set..
111     Net_Conf();
112     xil_printf("Static Mode .. \r\n");
113     Display_Net_Conf();
114
115     // -----
116     // W5500 : Link Check..
117     phy_link = w5500_getphylink();
118     while(1)
119     {
120         if(phy_link==WIZ_YES)
121         {
122             // TCP server loopback test
123             if ((ret = loopback_tcps(SOCK_TCPS, tcpRxBuffer, PORT_TCPS)) < 0)
124             {
125                 xil_printf("SOCKET_TCP ERROR !! \r\n");
126             }
127         }
128     }
129     cleanup_platform();
130     return 0;
131 }
    
```

w5500.h

```

#define WIZ_YES    1
#define WIZ_NO     0
        
```

TCP/IP Implementation Using W5500^{blaze_w5500_Exam *.xpr}

➤ 결과 확인

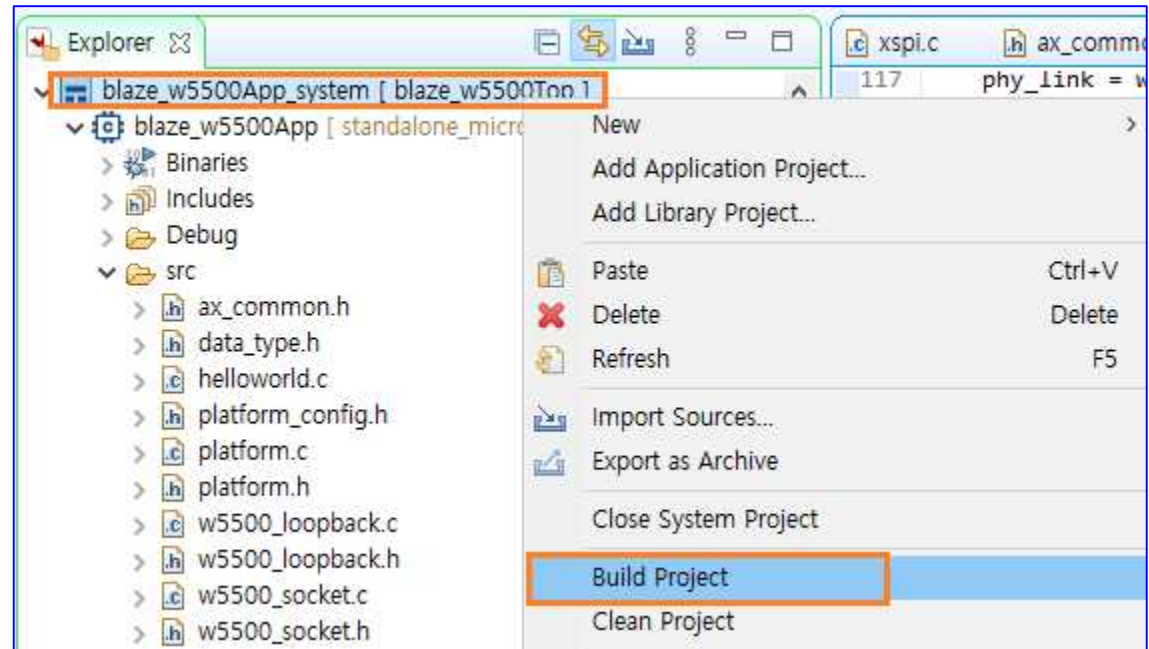
- ❖ 프로그램을 Build하고 보드에 다운

로드 해서 결과를 확인

➤ Build Project

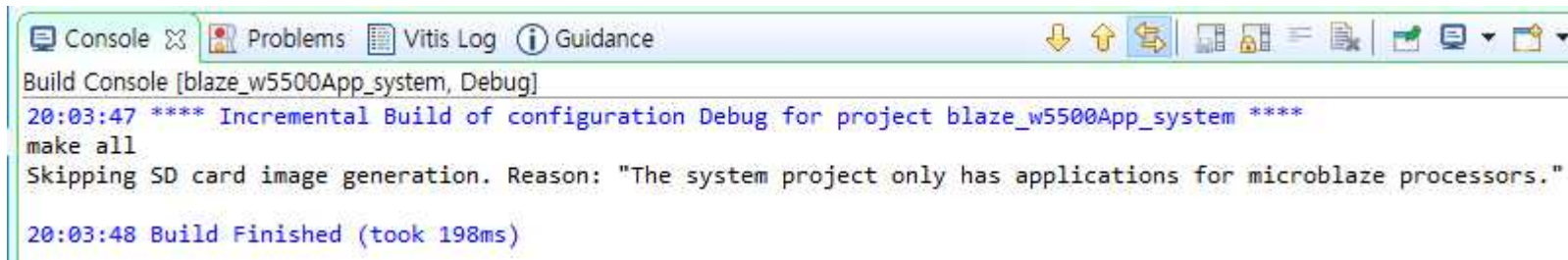
- ❖ vitis 에서 blaze_w5500App.system –

우 클릭 – Build Project를 클릭



- ❖ Console 창에 에러가 없이 Build가

완료되었음을 확인



TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ PC Network 설정

- ❖ 보드와 데이터 송수신을 하기 위하여 PC의 Network 정보를 설정 ➔ PC와 보드를 Network Cable로 바로 연결해도 되고 공유기를 거쳐서 연결해도 됨
- ➔ PC 설정에서 네트워크 및 인터넷 – 고급 네트워크 설정 – 어댑터 옵션 변경 – 이더넷 선택 – 우 클릭 – 속성 – 인터넷 프로토콜 버전 4 (TCP/IPv4) 선택 – 속성 클릭
- ❖ 오른쪽 그림과 같이 설정을 변경한 후 확인을 클릭
- ❖ 보드의 IP가 11번으로 설정 ➔ 따라서 PC는 10번으로 설정

인터넷 프로토콜 버전 4(TCP/IPv4) 속성

일반

네트워크가 IP 자동 설정 기능을 지원하면 IP 설정이 자동으로 할당되도록 할 수 있습니다. 지원하지 않으면, 네트워크 관리자에게 적절한 IP 설정값을 문의해야 합니다.

☐ 자동으로 IP 주소 받기(O)

☒ 다음 IP 주소 사용(S):

IP 주소(I): 192 . 168 . 1 . 10

서브넷 마스크(U): 255 . 255 . 255 . 0

기본 게이트웨이(D): . . .

☐ 자동으로 DNS 서버 주소 받기(B)

☒ 다음 DNS 서버 주소 사용(E):

기본 설정 DNS 서버(P): . . .

보조 DNS 서버(A): . . .

☐ 끝낼 때 설정 유효성 검사(L)

고급(V)...

확인 취소

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ 프로그램 다운로드 및 결과 확인

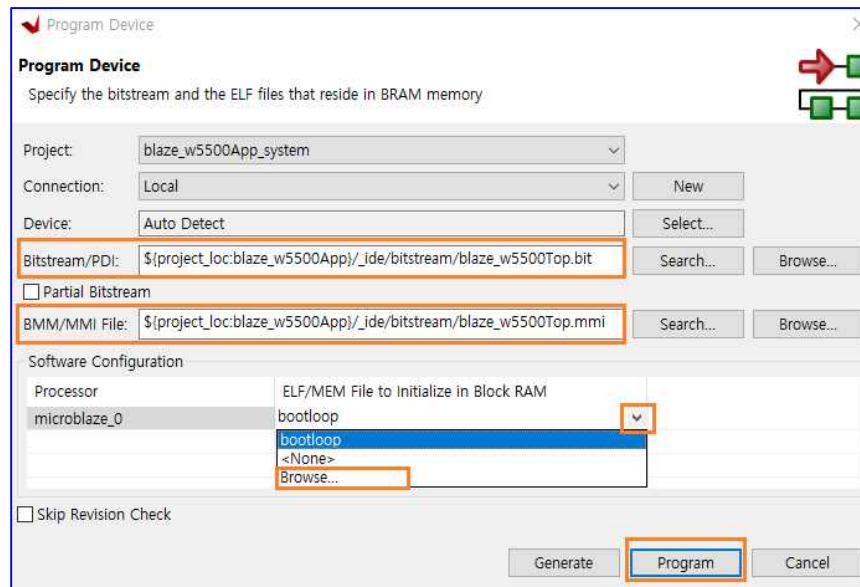
- ❖ 다운로드 받은 WinIDT_v17.zip 파일을 압축을 풀고, Release 폴더안에 있는 “WinIDT.exe”를 실행합니다. PC에 연결된 COM port를 확인하고, 속도는 115200으로 설정하고, Open 버튼을 클릭하여 Com Port를 Open



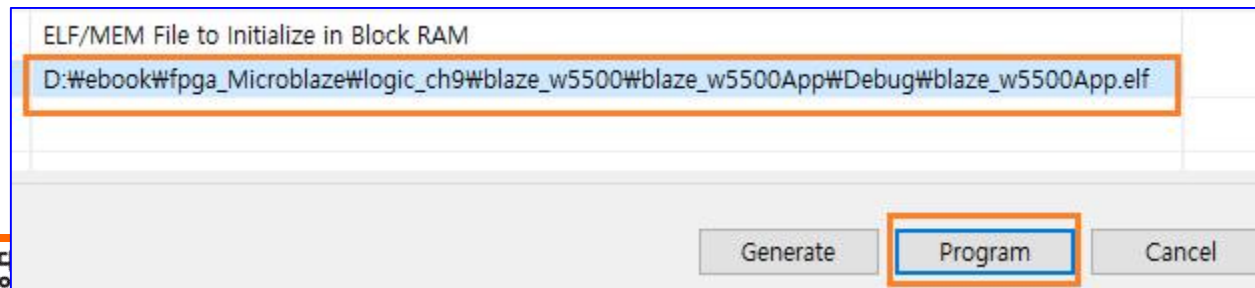
TCP/IP Implementation Using W5500^{blaze_w5500_Exam *.xpr}

➤ 프로그램 다운로드 및 결과 확인

- ❖ 보드의 USB 케이블을 연결하여 전원을 인가하고 프로그램을 다운로드 ➔ vitis 에서 Xilinx – Program Device 를 클릭 ➔ Program Device 윈도우에서 **Bitstream(.bit), MMI file(.mmi)** 의 위치가 현재 프로젝트 폴더인지 확인 ➔ Software Configuration 의 microblaze_0 우측의 하단 화살표를 클릭 ➔ Brows... 를 선택하고 Program 버튼을 클릭



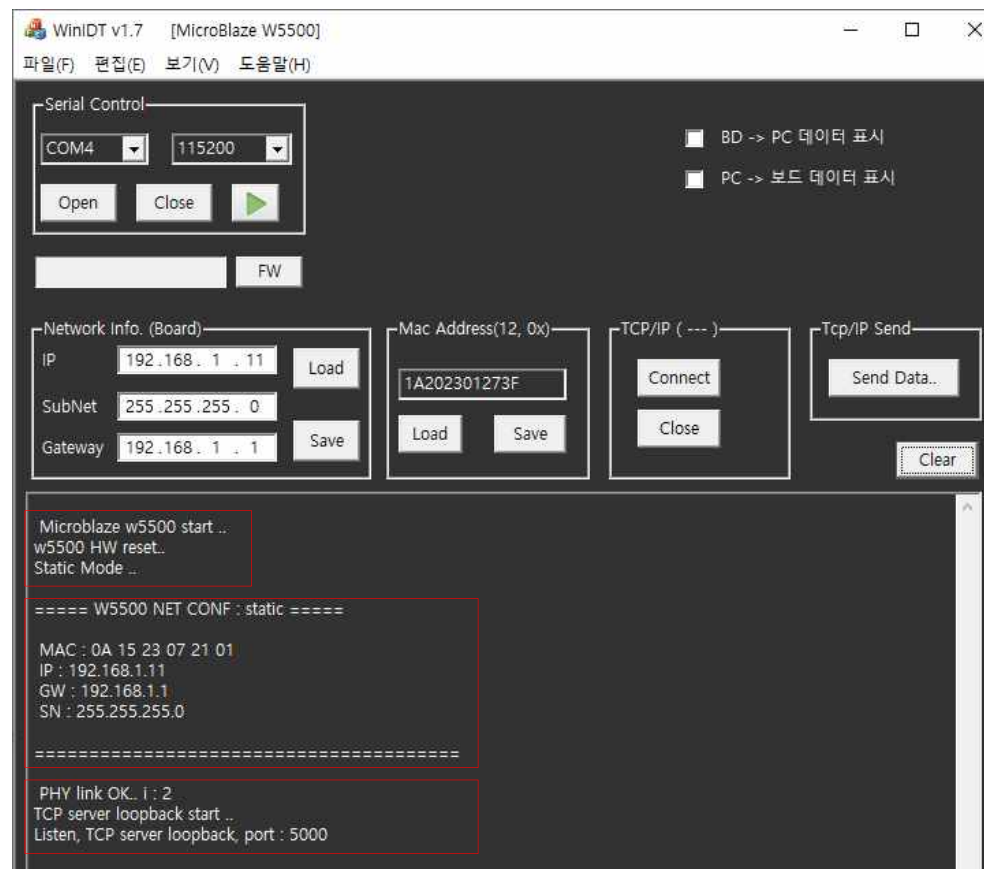
- ❖ blaze_w5500\blaze_w5500App\Debug” 폴더에 있는 blaze_w5500App.elf 를 더블 클릭 ➔ 파일이 선택 ➔ 다시 Program 버튼을 클릭하면 프로그램이 다운로드



TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ 프로그램 다운로드 및 결과 확인

- ❖ 아래 그림은 프로그램이 다운로드 되고 동작된 모습
- ❖ PC의 Network 설정이 제대로 이루어지고, PC와 Network Cable이 연결되었다면, 아래와 같은 메시지
- ❖ w5500을 초기화 하고, w5500에 network 정보를 설정하고, w5500에 설정된 network 정보를 display
- ❖ PC와 Network 이 연결되었다면, PHY link Ok.. 가 되면서 w5500은 TCP Server 로 동작하고, Client의 접속을 기다림



TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ 프로그램 다운로드 및 결과 확인

- ❖ Network Info. 가 제대로 설정되었는지 확인 (보드의 IP : 192.168.1.11) TCP/IP의 Connect 버튼을 클릭하면 연결 ➔ 첫번째 라인은 보드의 IP 이고, 두번째 라인은 PC의 IP

- ❖ TCP/IP Send 의 Send Data.. 버튼을 클릭 ➔ PC에서 1200바이트의 데이터를 전송 ➔ 보드에서는 1200바이트를 수신하고, 디버깅을 위해 화면에 보여주고, 수신한 데이터를 다시 PC로 전송

- ✓ send ok ... : PC에서 1200 바이트 전송
- ✓ W5500 Received size ~ : w5500에서 1200 바이트 수신함, AXI SPI 를 통해 240바이트씩 5회 수신함, 수신한 데이터를 화면에 출력 (Uart-PC)함(48바이트 x 25 라인)
- ✓ PC Received size ~ : 보드에서 수신한 데이터를 다시 PC로 전송, PC에서 수신한 데이터를 나타냄



TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ 프로그램 다운로드 및 결과 확인

- ❖ Send Data ... 버튼을 반복해서 클릭해도 1200 바이트를 정상적으로 송수신 하는 것을 확인

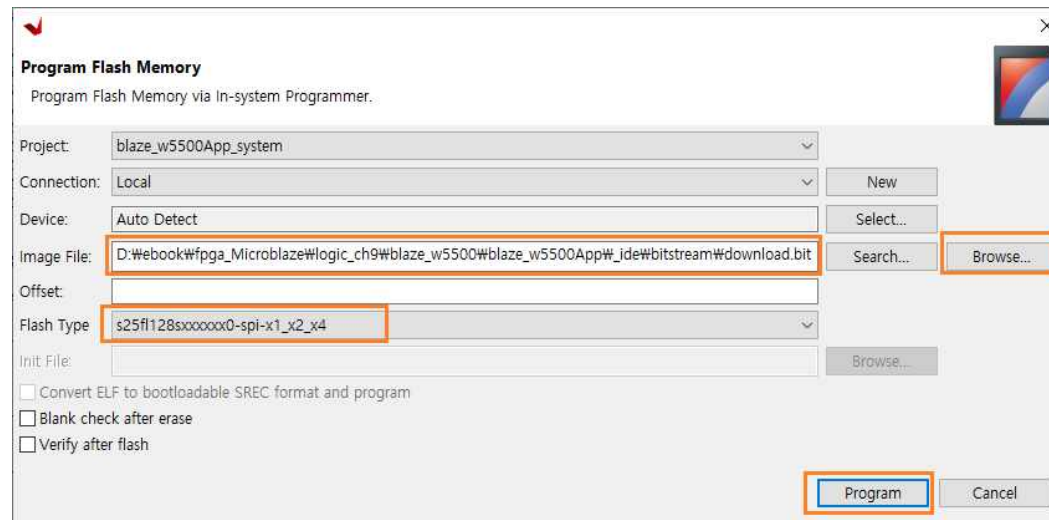
- ❖ BD → PC 데이터 표시를 check 하고 Send Data.. 버튼을 클릭하면, PC쪽 수신 데이터를 확인



TCP/IP Implementation Using W5500^{blaze_w5500_Exam *.xpr}

➤ 외부 Flash에 프로그램 다운로드

- ❖ 지금까지는 내부 SRAM에 프로그램을 다운로드 하였음 ➔ 전원을 Off 하면 데이터가 날라가 버려서 프로그램을 다시 시작할 수 없음
- ❖ Basys3 보드는 외부에 Flash를 가지고 있으므로 외부 Flash에 프로그램을 저장하면 전원을 Off 해도 데이터가 사라지지 않아서 프로그램을 다시 동작할 수 있음
- ❖ vitis 에서 Xilinx – Program Flash 를 클릭합니다. Image File 우측 Browse... 를 클릭해서 download.bit 파일을 선택함 ➔ download.bit 파일은 “blaze_w5500\blaze_w5500App_ide\bitstream” 폴더 안에 있음 ➔ 그리고 Flash Type을 mxs25l3273f-spi-x1_x2_x4를 선택(part : mx23l3273f, Alias : mx25l3233f-spi-x1_x2_x4)하고 Program 버튼을 클릭

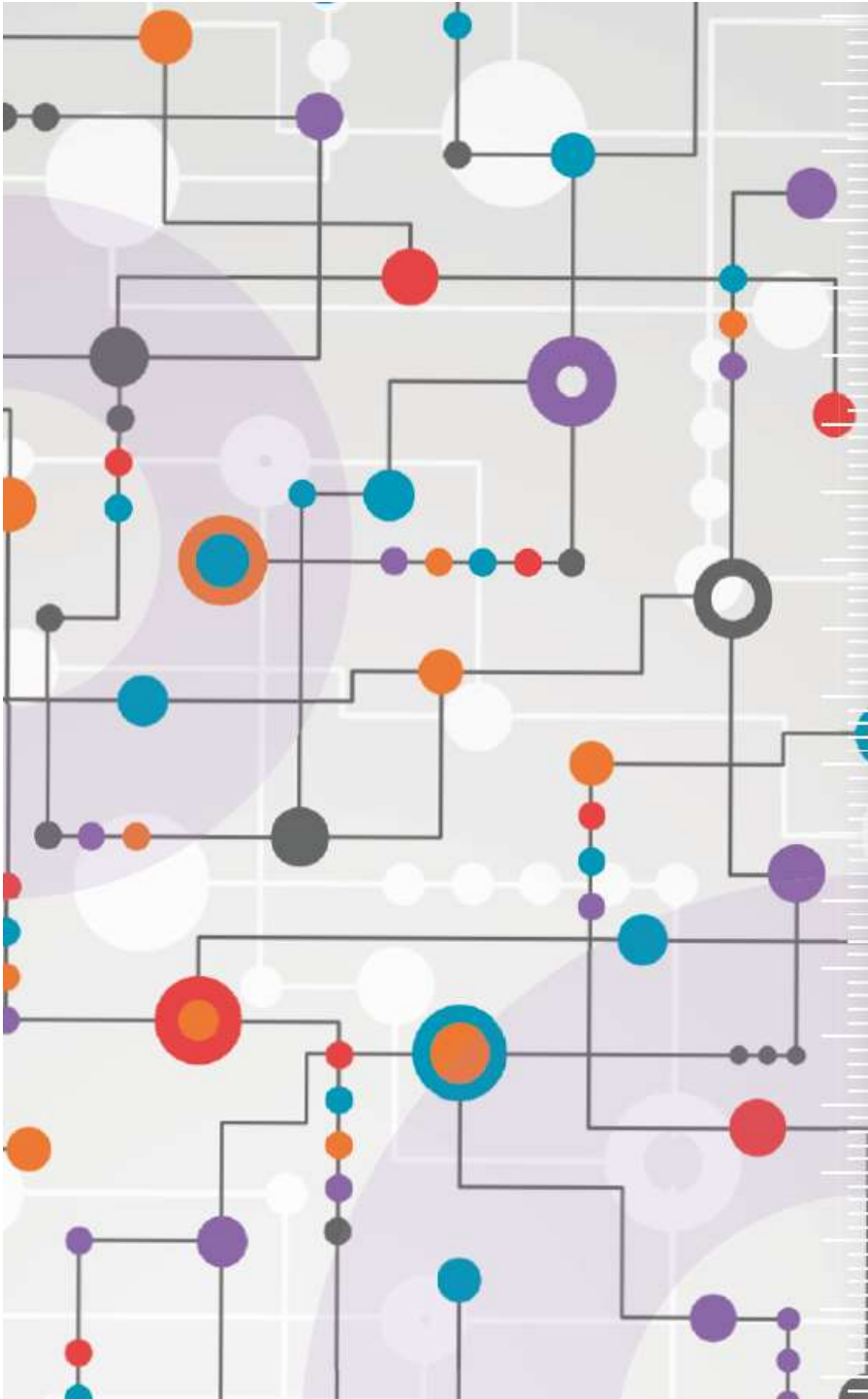


- ❖ 만일 해당 폴더에 download.bit 파일이 보이지 않으면, Xilinx – Program Device를 클릭 후, elf 파일을 불러와서 Generate 버튼을 클릭하면 download.bit 파일이 생성됩니다. 정상적으로 다운로드 되면, 하단의 Console 창에 “Flash Operation Successful” 메시지가 나타남
- ❖ 보드의 전원을 끄고, 다시 인가하면 프로그램이 동작하는 것을 확인할 수 있음

TCP/IP Implementation Using W5500^{laze_w5500_Exam *.xpr}

➤ 결론

- ❖ w5500 모듈을 이용하여 TCP/IP 를 구현
- ❖ Microblaze 와 AXI_QUAD_SPI IP를 사용하여 구현
- ❖ AXI_QUAD_SPI IP의 FIFO 최대값이 256 바이트여서, 용량이 큰 데이터는 240바이트씩 나누어서 송수신해야 하는 단점도 있지만, 그래도 데이터 송수신이 잘 되고 있음을 확인 ➔ 이미지 데이터와 같은 대량의 데이터를 전송하지 않고, 간단한 제어 목적이나 텍스트 데이터를 위한 송수신에는 적용
- ❖ 속도를 높이기 위해서는 spi clock 속도를 높일 수 있음 ➔ 본 예제에서는 w5500 모듈을 점퍼를 이용하여 연결했기 때문에 속도를 높이는데 한계가 있었지만, 24Mhz까지 사용해도 동작 가능



수고하셨습니다.