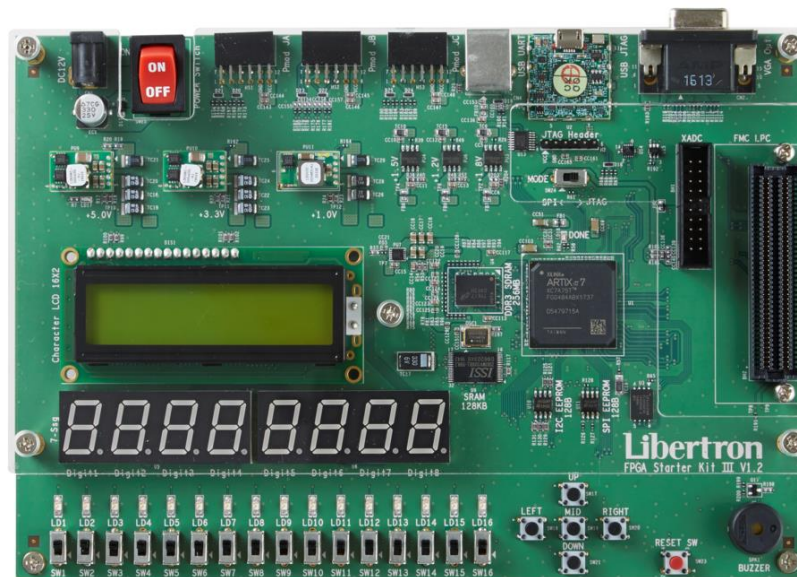


FPGA Starter Kit III

LCD & CAM Module Tutorial (v1.2B)

Authors : 기술지원팀 김민석 팀장



Libertron Co., Ltd

본 설명서를 (주)리버트론의 허락 없이 복제하는 행위는 금지되어 있습니다.

1. 개요

- 본 자료에서는, 리버트론에서 개발한 LCD & CAM Module 을 이용하여 Camera 의 영상을 LCD 로 출력함과 동시에, Touch Pad 기능을 디자인 하여 영상 및 터치패드 제어를 실습 해본다.



2. 상세 설명

2.1 준비 사항 및 테스트 환경

2.1.1 준비사항

- FPGA Starter Kit III (Ver 1.2B)
- LCD & CAM Module
- Power Adapter
- USB B Type Cable (FPGA 다운로드 용)

2.1.2. 테스트 환경

- Windows 10
- Vivado 2020.1 & VITIS 2020.1
- Vivado 2020.2 & VITIS 2020.2

*** Vivado 및 VITIS 버전이 맞지 않는 경우 작업 진행이 안될 수 있음**

- VITIS에서의 SW 컴파일 시, Makefile 생성에 에러가 발생한다.

Tutorial 자료 끝 부분에 Makefile 에러를 해결할 수 있도록 안내

2.2 프로젝트 세부 설명

2.2.1 디자인 동작 방향

- 본 자료에서는 FSK III 와 LCD & CAM Module 을 연동하여 사용하는 디자인을 소개 하였다.
- CAM 으로 받은 영상을 LCD 로 출력함과 동시에 LCD Touch 를 제어하는 디자인 이다.

2.2.2 구성 제품 스펙

● FMC LCD-Camera Module

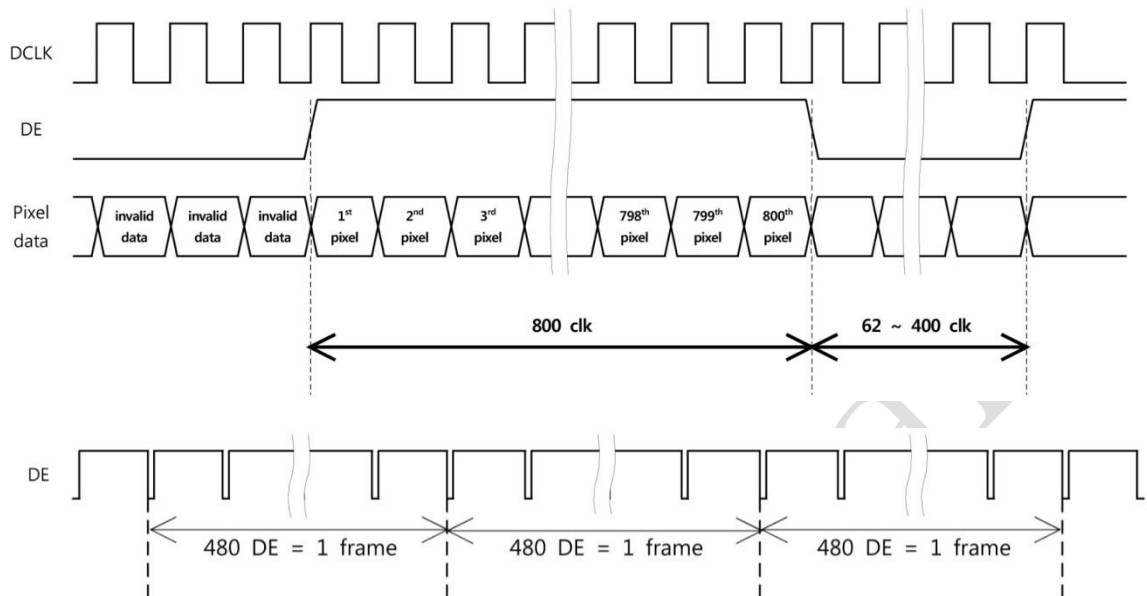
Item	Part	Spec	Note
Display	TFT-LCD	- 800 x 480 Pixel (7인치) - RGB Parallel Interface	
	Touch-Panel	- 7인치 CAP Touch (5-Point) - Controller IC 내장 (I2C)	
Camera	CMOS Image Sensor	- 1080P / 30FPS - Parallel Interface	
Connector	FMC	- Support VITA 57.1 (LPC x 1EA)	
	PMOD	- PMOD Connector x 2EA	
Power	Power Input	- FMC Power Input (+12V, +3.3V, ADJ)	
제품 size	LCD Module	- 205mm x 125mm (W x D) - Height : 40mm ~ 130mm (Adjustable)	
	Camera Module	- 60 x 60 x 17 (W x D x H)	

● LCD Interface 정보

✓ Timing Parameter

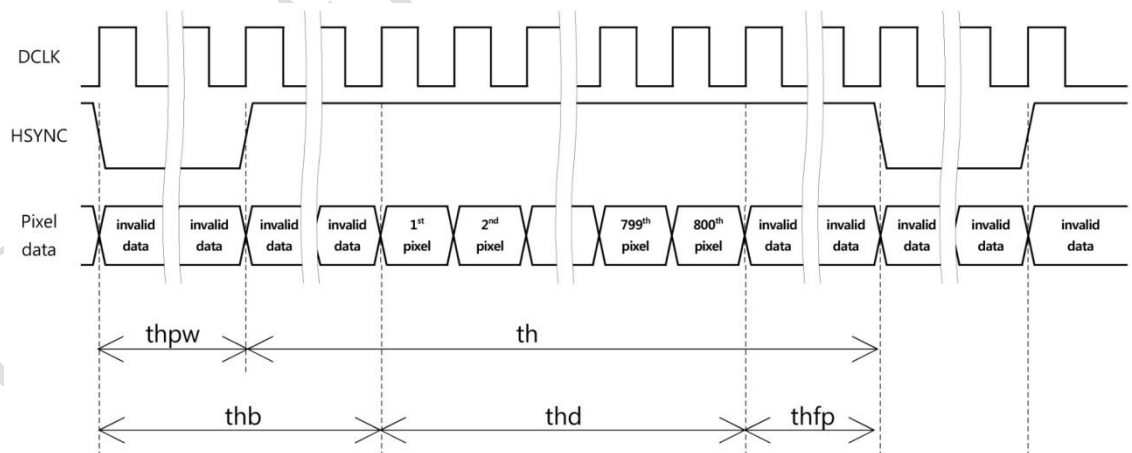
Parameter Name	Symbol	Value			Unit	Remark
		Min	Typical	Max		
DCLK Frequency	fclk	26.4	33.3	46.8	MHz	
Horizontal Display Area	thd	800	800	800	DCLK	
HSYNC Pulse Width	thpw	1	-	40	DCLK	
HSYNC Blanking	thb	46	46	46	DCLK	
HSYNC Front Porch	thfp	16	210	354	DCLK	
One Horizontal Line	th	862	th = thd + thb + thfp	1200	DCLK	
Vertical Display Area	tvd	480	480	480	th	
VSYNC Pulse Width	tpvw	1	-	20	th	
VSYNC Blanking	tvb	23	23	23	th	
VSYNC Front Porch	tvfp	7	22	147	th	
VSYNC Period Time	tv	510	tv = tvd + tvb + tvfp	650	th	

✓ Timing Parameter



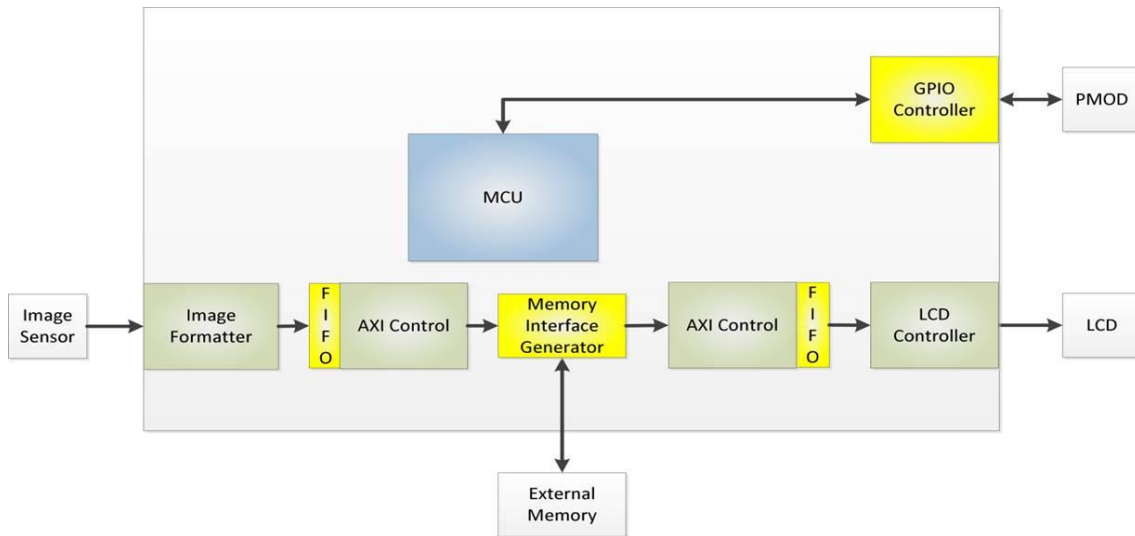
- 상기의 DE Mode 는 HSYNC, VSYNC 를 논리 and 한 신호를 DE 라는 이름으로 생성하여 사용하는 방식이다. 그러므로 DE 신호 하나에 HSYNC 와 VSYNC 신호의 속성을 모두 가지고 있다.

✓ Timing Diagram(HSD/VSD Mode: Mode Pin Low)



- 상기의 HSD, VSD 모드는 HSYNC, VSYNC 를 기반으로 영상을 동작시키는 모드이다. 기본적인 픽셀클럭을 기반으로 HSYNC 를 생성하고 픽셀클럭과 HSYNC 를 기반으로 VSYNC 를 생성하여 영상을 출력 사용한다

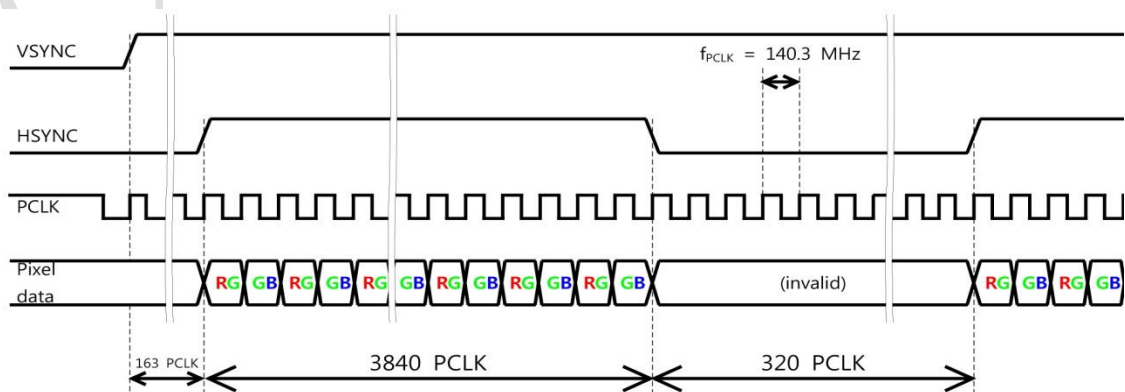
2.2.3 디자인 블록 이해 (Block Diagram)



1) MCU 블록

- 일반적으로 카메라 디자인을 편하게 제어하기 위해 프로세서를 사용하나, FSK III 는 FPGA 기반 플랫폼으로, 디바이스 내에 프로세서가 존재하지 않는다.
- Xilinx 에서는 이를 위해 MicroBlaze 기능을 제공하여 FPGA Logic 에 Softcore의 MCU를 구성하고 프로세서의 기능을 대신 한다.
- MCU 블록은 FPGA 내에서 동작하는 블록이므로 Vivado IPI 에서 MicroBlaze IP 로 환경을 구성하고 차후 SW 영역 개발 환경인 Vitis 에서 C 코드로 동작 디자인을 구성하게 된다.

2) Image Formatter



- 카메라에서는 상기의 그림과 같이 영상데이터가 들어오게 되므로, 일반적인 RGB 순서로 데이터가 출력되지 않고 RG GB 형태로 출력되게 된다. 이러한 RG GB 형태의 데이터를 RGB 형태로 변환 시켜주는 작업을 한다.

3) AXI Control

- 자일링스에서는 많은 IP의 Interface 를 AXI 기반으로 만들어 제공하고 있다. 또한 자일링스는 외부 DDR Memory 인터페이스를 하기 위해 MIG (Memory Interface Generator)를 제공하는데, 이 MIG 또한 User Interface 와 AXI4 Interface 로 사용할 수 있도록 지원한다.
- AXI 에 대한 규격을 맞춰 사용하면 사용하면 User 가 컨트롤 하여 쉽게 데이터를 주고 받을 수 있는데, 본 디자인에서도 외부 메모리 (DDR3) 로 데이터를 보내는 블록과 받는 블록을 MIG IP 로 구성 하였다.
- MIG IP를 AXI Interface 형태로 구성하여 영상 데이터를 DDR3 에 저장하고 다시 불러오는 역할을 한다.

4) Memory Interface Generator

- 상기 내용에 언급했던 내용으로, DDR Memory Controller를 MIG 라고 하는 IP로 제공한다. 본 디자인에서는 MIG 디자인을 IPI (IP Intergrator)환경에서 디자인하여 블록을 구성 하였다.

5) LCD Controller

- 상기의 LCD Interface 정보란과 같이 TFT LCD 는 800 x 480 해상도를 지원한다. 사용하는 픽셀 클록은 33.3MHz를 사용한다.

2.2.4 제공 디자인 블록 이해

★ 리버트론 제공 디자인 ★

구분	목록
HDL 디자인	<ul style="list-style-type: none"> • FMC_LCD_test_top.v • AXI_to_FIFO.v • CAM_IF_to_FIFO.v • CAM_rst_GEN.v • FIFO_to_AXI.v • FIFO_to_PARALLEL.v • pattern_gen_lcd.v
핀 정보	<ul style="list-style-type: none"> • FMC_LCD_test.xdc • mig_7series_0.xdc
IP Block Design	<ul style="list-style-type: none"> • FLC_ctrl_1.0.zip • Touch_Intr_ctrl_1.0.zip • MB_BLOCK.tcl
C 코드	<ul style="list-style-type: none"> • helloworld.c • peri_iic_ctrl.c • peri_iic_ctrl.h • zn220_initialize_list.h

* 프로젝트를 위해 당사에서 제공하는 상기 디자인을 미리 준비한다.
상기 목록에 없는 블록은 Vivado Tool 내의 IP Catalog 에서 직접 구성한다.

1) FMC_LCD_test_top.v 블록

- FSK III 와 FMC LCD Camera 모듈을 동작 시키는 Top 디자인
- Clock, DDR3 Memory 를 연결하고, FMC- LCD CAM Module의 Interface 구성
- LCD CAM Module이 동작하기 위한 모든 인터페이스 연결을 담당한다.

2) CAM_mclk_gen 블록

- Camera 동작 Clock을 생성하는 블록
- 100MHz의 입력 Clock 을 받아, CAM 에서 사용하는 27MHz의 Clock을 생성하는 블록이며, Active Low 로 Reset 초기화 한다.
- Clocking wizard IP를 이용하여 생성

3) SYS_clk_GEN 블록

- 시스템에서 사용하는 Clock 을 생성하는 블록
- MicroBlaze (100MHz), TFT-LCD (33.3MHz), DDR_ref_clk (200MHz) Clock을 생성하며, Active Low 로 Reset 초기화 한다.
- Clocking Wizard IP를 이용하여 생성

4) CAM_IF_to_FIFO.v 블록

- Camera 에서 들어온 1920X1080 영상 데이터의 프레임을 버퍼링 한다.
- 버퍼링을 Camera 의 HSYNC (1920), VSYNC (1080), DE (1920x1080) 신호에 맞게 버퍼링을 한다.

5) FIFO_to_AXI.v 블록

- Camera 에서 나온 데이터를 CAM_IF_to_FIFO.v 블록에서 HSYNC, VSYNC, DE 신호에 맞게 버퍼링하여 전달한 데이터를 받아서 FIFO 에 저장을 한다.
영상데이터를 받아서 저장을 해야 하는데, FPGA 내의 메모리의 용량이 작아, DDR 메모리에 저장 하도록 한다.
- FPGA 내에서 동작하는 Clock 과 DDR Clock 이 다르므로, Read/Write Clock 을 다르게 사용할 수 있는 Async Clock 을 활용 한다.
FIFO Memory 에서 Async Clock 을 사용하여 DDR Memory 에 데이터 전달을 진행 하며, AXI Interface 를 이용하여 FIFO 에 저장된 Memory 를 MB_BLOCK_wrapper.v 블록에 전달 한다.

6) CAM_Pdata_sync_FIFO 블록

- FIFO_to_AXI.v 블록에서 들어오는 데이터를 저장하는 FIFO 블록

7) AXI_to_FIFO.v 블록

- 이 IP 는 Buffer Address / output size, sync signal 값을 받아 이에 맞게 데이터를 읽어온 후 외부 FIFO Memory 에 저장한다.
- 이 IP 는 1개의 double frame buffer 에서 odd_even_write 의 선택에 따라 영상 데이터를 읽어 온다.

8) Pattern_gen_LCD_IOF.v 블록

- TFT LCD 출력을 위한 컨트롤 신호 생성 블록.
- TFT LCD 출력 해상도는 800 x 480 이다.

9) LCD Pdata FIFO 블록 (CAM_frame_buffer, OVR_frame_buffer)

- Camera 의 영상 이미지를 버퍼링 한다.
- TFT LCD 에 출력하기 위한 영상 이미지를 버퍼링 한다.

10) FIFO_to_PARALLEL.v 블록

- 이 블록은 RGB565 데이터를 입력 받아서 각 8bit RED, Green, Blue 총 24bit 로 출력시키는 블록이다.
- TFT LCD 컨트롤 신호는 pattern_gen_LCD_IF.v 블록에서 받아서 사용한다.

11) CAM_rst_GEN.v

- Camera 동작 초기화를 위한 블록이다.
- 동작을 위해서 Clocking wizard 에서 생성한 lock 신호를 이용한다.

12) MB_BLOCK_wrapper.v 블록

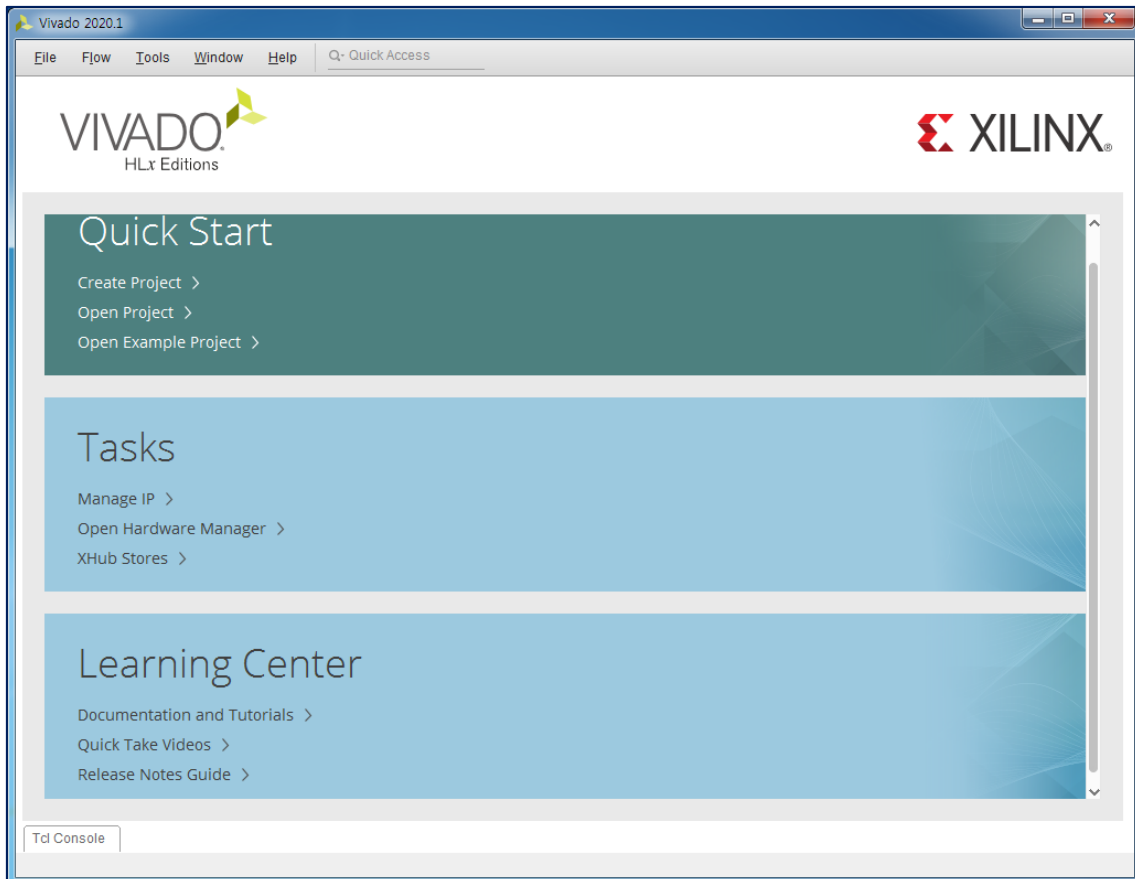
- MB_MCB.bd (IPI 디자인) 를 포함 한다.
MB_MCB.bd 디자인에는 MicroBlaze 블록, MIG (Memory Interface Generator) 블록, Camera IIC 인터페이스 블록, Uart 통신 블록 등이 있다.
- 이 블록은 시스템 컨트롤 및 DDR Memory 컨트롤 등을 위해 IPI (IP Integrator) 에서 MicroBlaze 를 이용하여 디자인 구성을 하였다.
MicroBlaze 로 구성된 블록이므로, 차후 VITIS 에서 SW 디자인을 구성하여 사용한다.
- 블록 디자인 구성은 당사에서 제공하는 MB_BLOCK.tcl 파일을 이용한다.
(MB_BLOCK.tcl 파일은 Vivado 버전이 맞지 않으면 에러가 발생하므로, 버전을 맞춰 사용한다.)

2.3 Vivado를 활용한 디자인 실습

2.3.1 Viavdo 실행

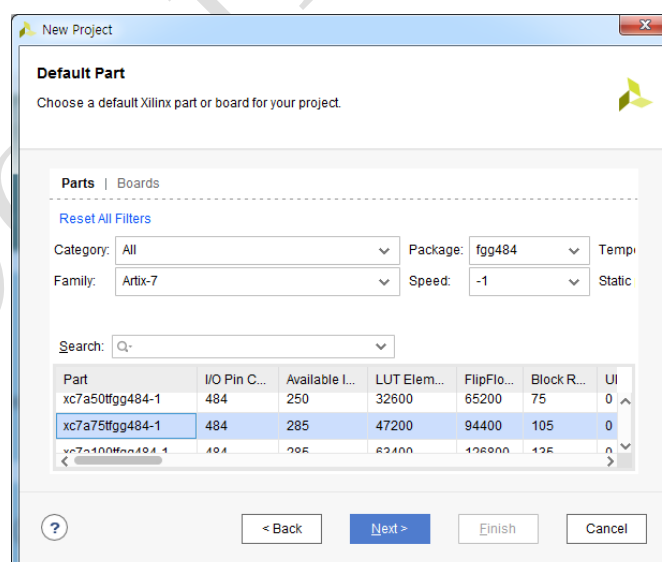
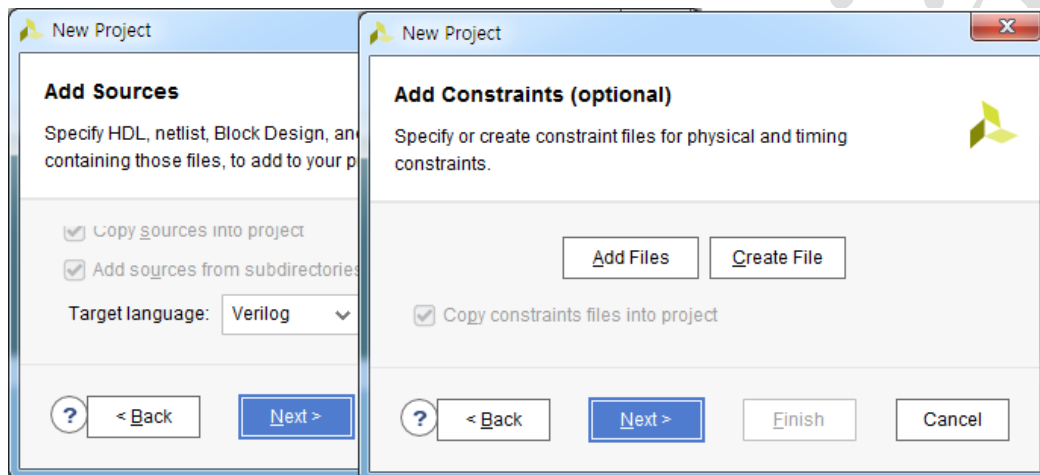
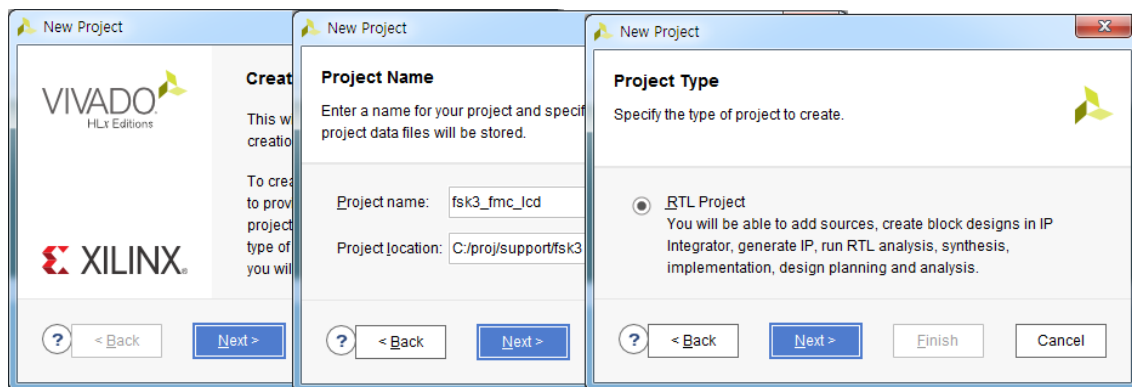
1) Vivado 실행 (Vivado 2020.1 / 2020.2)

* Project 작업 경로 및 폴더에 특수문자 및 한글의 인식이 안되므로, 영어만 사용할 것



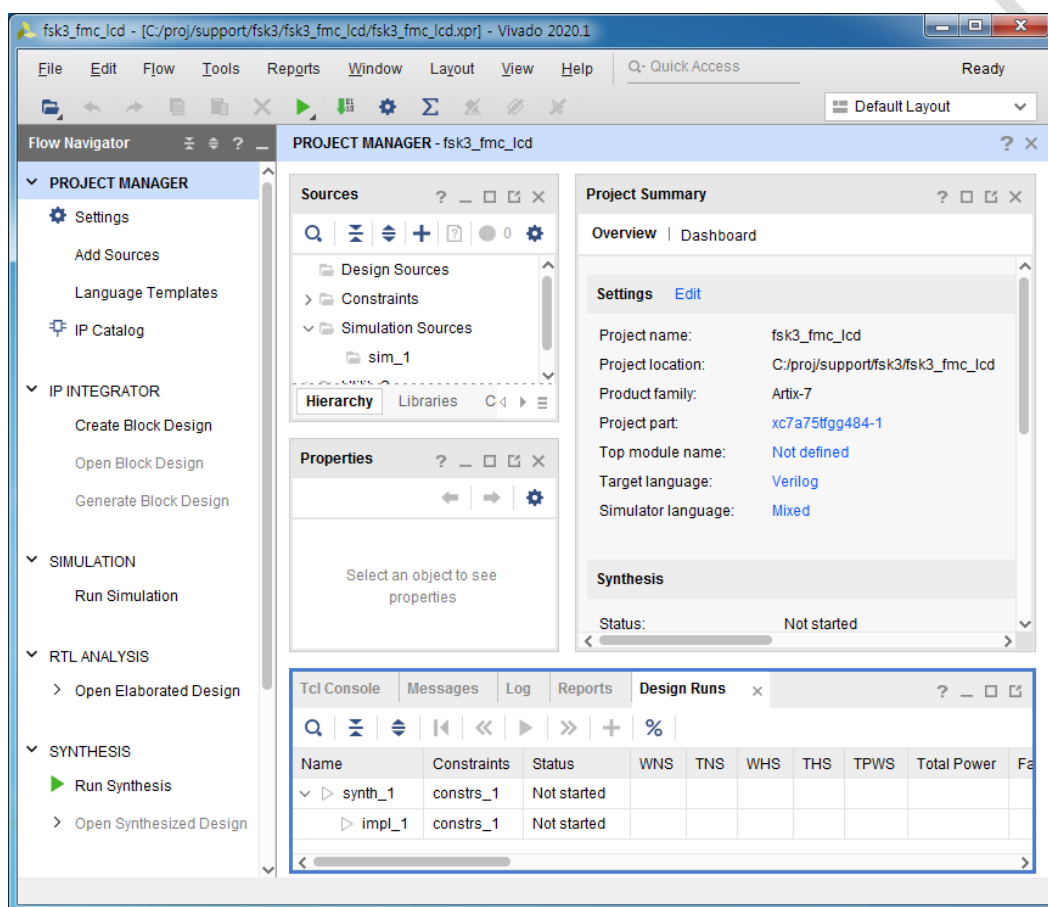
2) 상기 그림에서 다음과 같이 진행 한다.

- Create Project → Create a New Vivado Project 에서 Next 클릭
- Project Name 란에 "fsk3_fmc_lcd" 입력 후 Next 클릭
- RTL Project 선택 후 Next 클릭
- Target Language 란에 Verilog 확인 후 Next 클릭
- Default Part 란에서 하기와 같이 선택 (하기 그림 참조)



Family	Artix-7
Package	Fgg484
Speed	-1
Full Part Name	XC7A75TFGG484-1

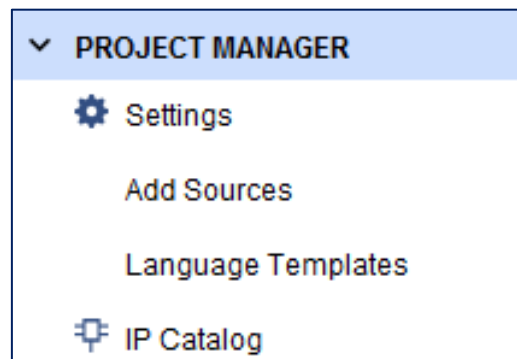
- 3) 상기 작업 후 New Project Summary 창이 나타나면 Finish 클릭
작업이 완료 되면 하기와 같이 Vivado 초기 프로젝트 화면이 나타난다.



2.3.2 HDL Design Import

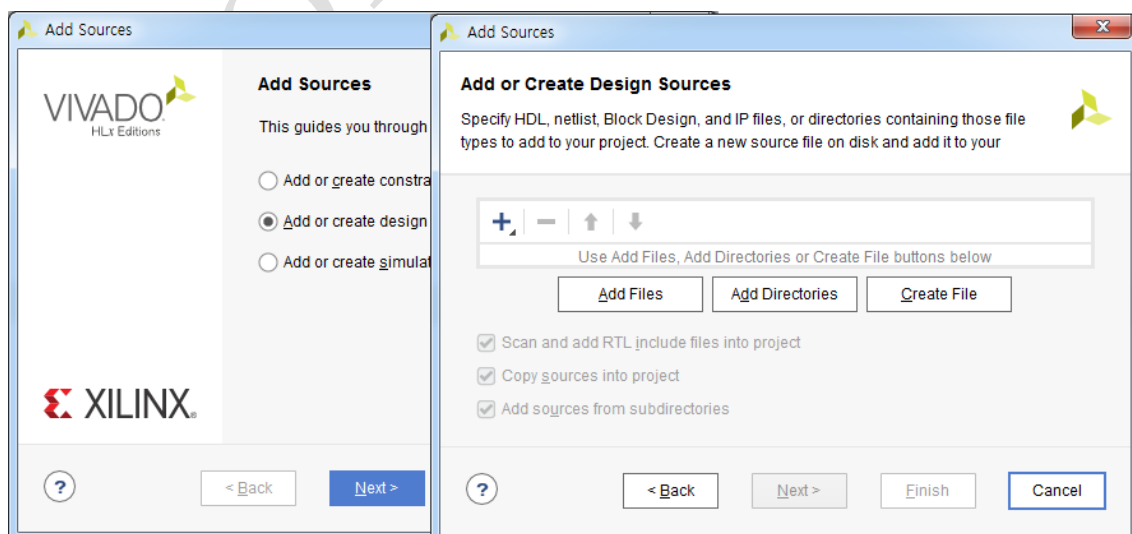
- 1) 하기 그림과 같이 Add sources 를 선택하여 당사에서 제공한 디자인 소스를 받아 들인다.

* 당사에서 제공하는 HDL 소스파일을 Project 폴더에 넣어둔다.

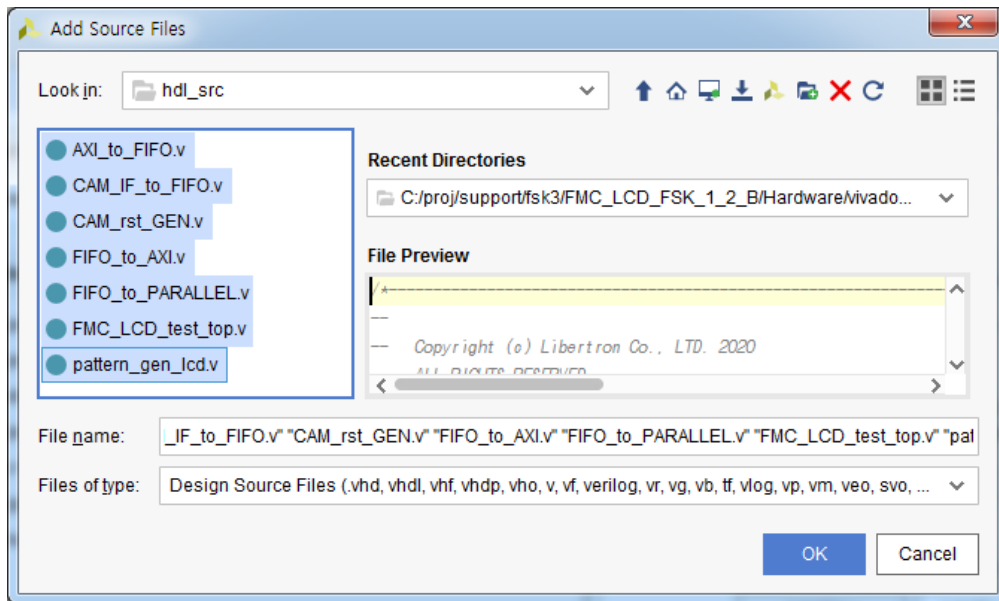


★ 당사 제공 hdl 디자인★

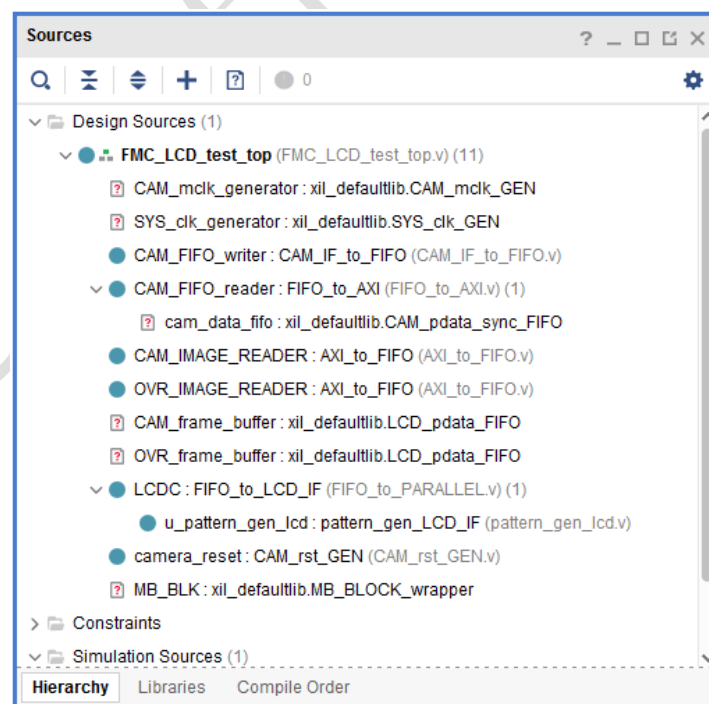
HDL 디자인	FMC_LCD_test_top.v AXI_to_FIFO.v CAM_IF_to_FIFO.v CAM_rst_GEN.v FIFO_to_AXI.v FIFO_to_PARALLEL.v pattern_gen_lcd.v
---------	--



- 2) 상기 좌측 그림에서 Add or Create Design Sources 를 선택하고
우측 그림이 나타나면 Add Files 를 클릭한다.
하기 그림과 같이 당사에서 제공하는 HDL 디자인 선택
(당사 제공 Design_src 파일의 hdl_src 폴더) → OK → Finish

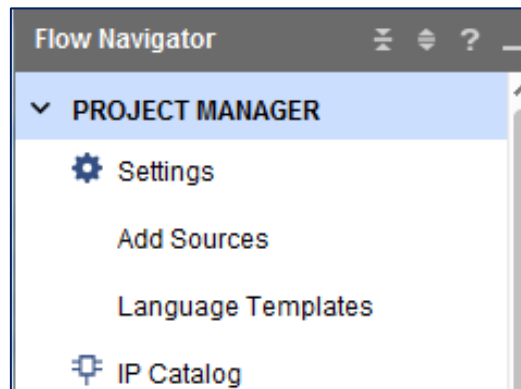



하기 그림과 같이 받아들인 HDL 디자인이 sources 창에 나타난 것을 확인할 수 있다.

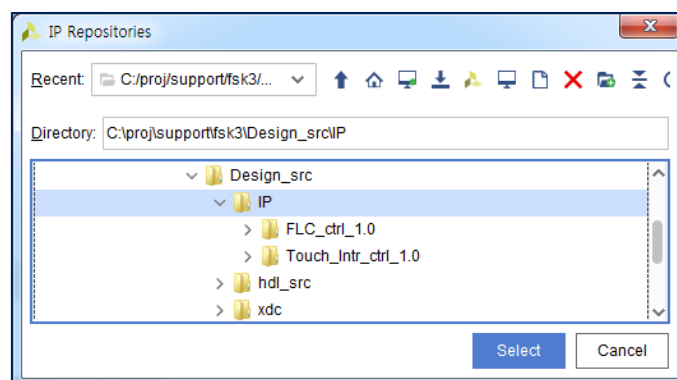
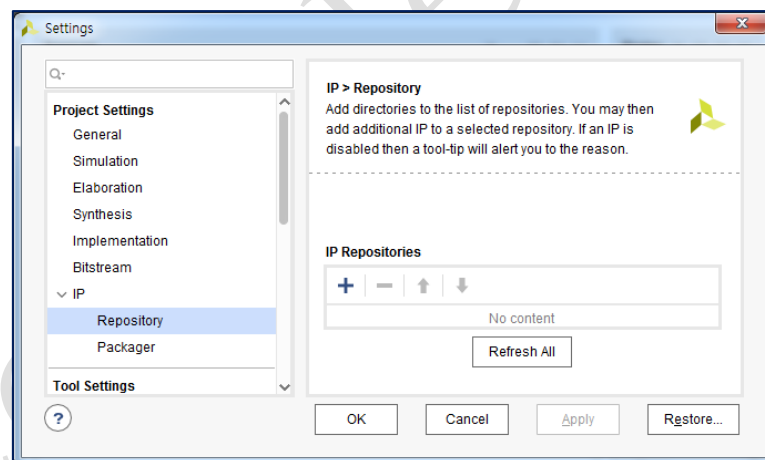


2.3.3 HDL Design Import

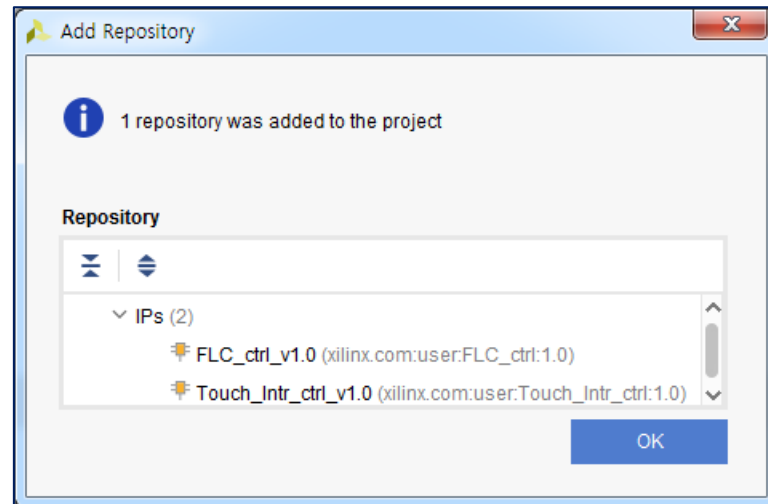
- 1) 당사에서 제공하는 FLC_ctrl_1.0.zip, Touch_intr_ctrl_1.0.zip 파일을 영어로된 폴더에 넣고 압축을 푼다. (당사 제공 Design_src 파일의 IP 폴더)
하기 그림과 같이 Flow Navigator → PROJECT Manager → Settings 선택



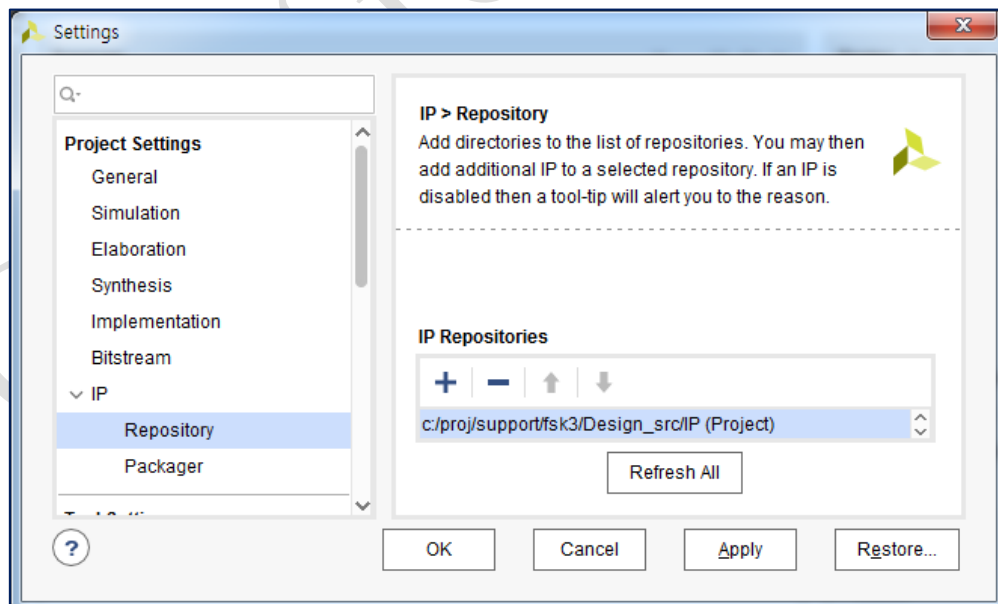
- 2) 하기 창이 나타나면 화면 좌측의 IP → Repository 를 선택하고 오른쪽 화면에 있는  를 클릭하여 , 당사에서 제공한 IP 가 있는 폴더를 선택한다.



- 3) 상기 작업을 완료하면 하기와 같이 받아들인 IP 리스트가 나타난다.
하기 창에서 OK 를 선택한다.
(IP List : FLC_ctrl_1.0 , Touch_Intr_ctrl_1.0)

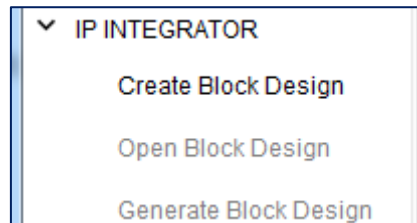


- 4) 하기와 같이 IP Repositories 탭에 받아들인 IP 폴더가 나타난다.
확인 후 OK 를 선택 한다.

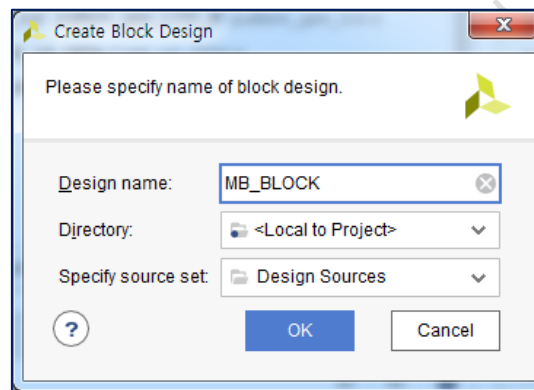


2.2.4 IPI (IP Integrator) 를 이용한 MicroBlaze 디자인 구성

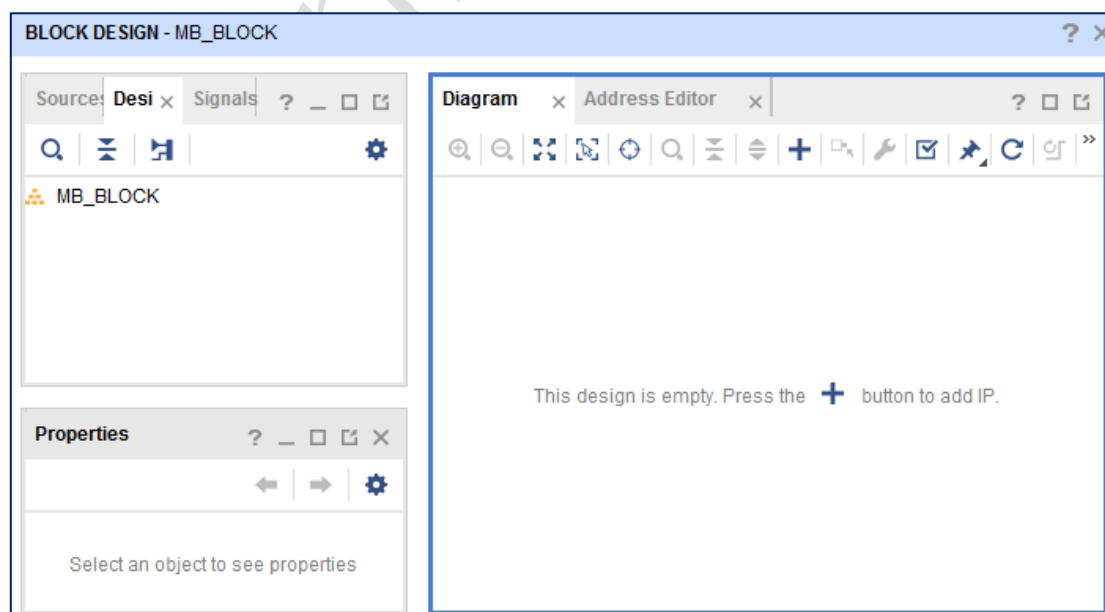
- 1) 하기와 같이 Flow Navigator 창에 IP INTERGRATOR 탭에 있는 Create BlockDesign 을 클릭한다.



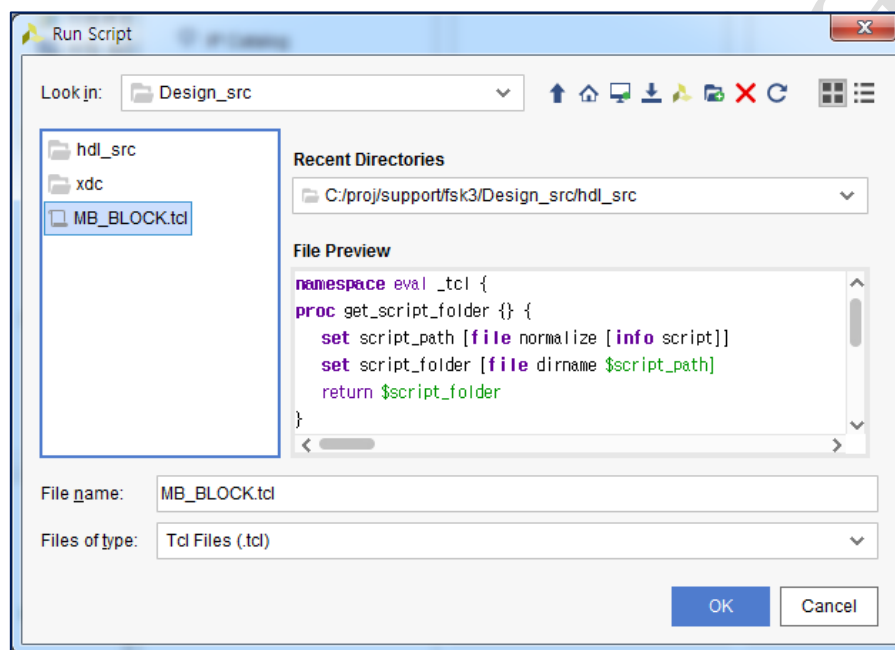
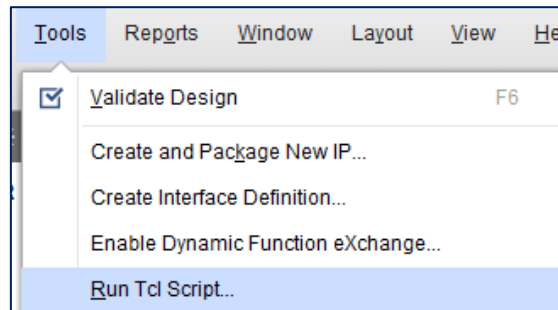
하기 창에서 Design Name 란에 MB_BLOCK을 입력하고 OK 선택



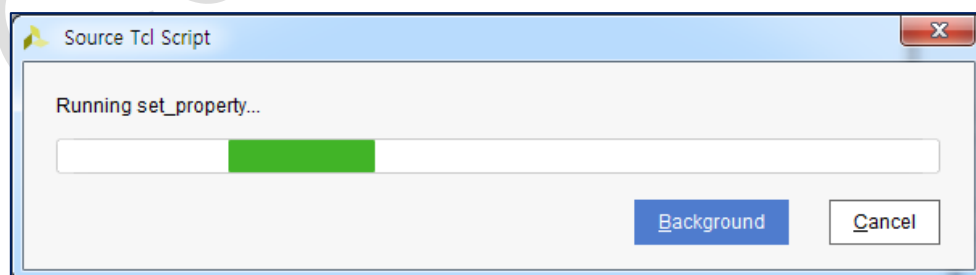
하기와 같이 IPI Block Design 창이 나타난다.



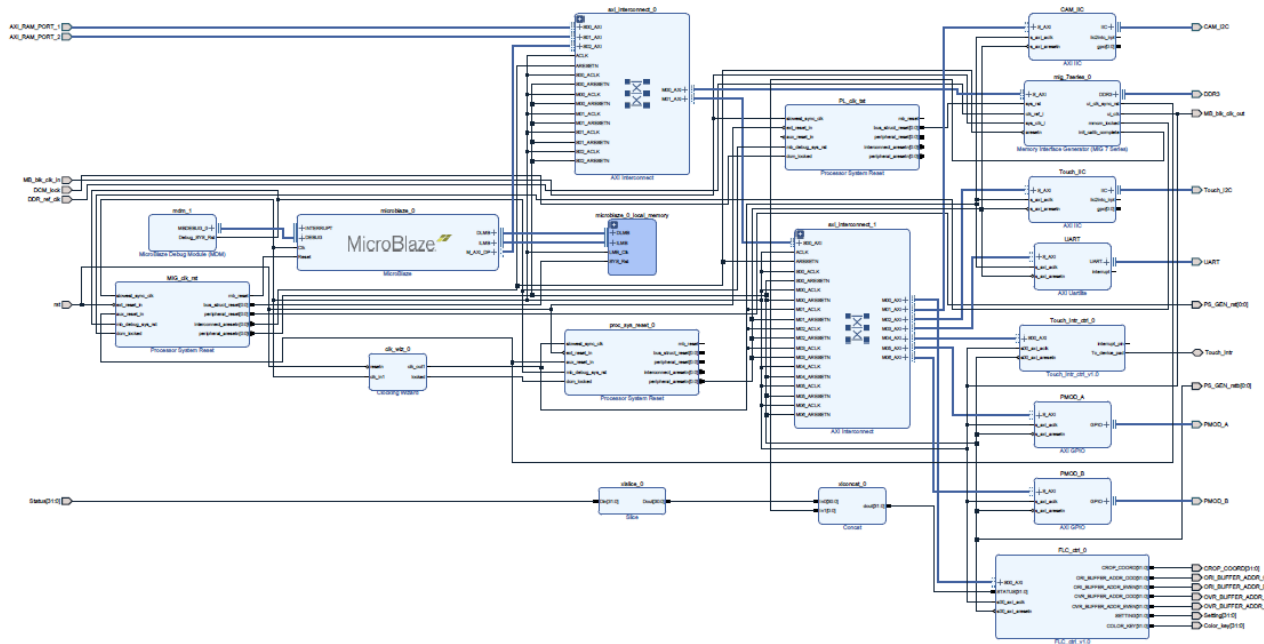
- 2) Vivado 창에서 Tools → Run Tcl Script... → Vivado 버전에 맞는 MB_BLOCK.tcl (당사 제공 Design_src 파일) 선택 → OK



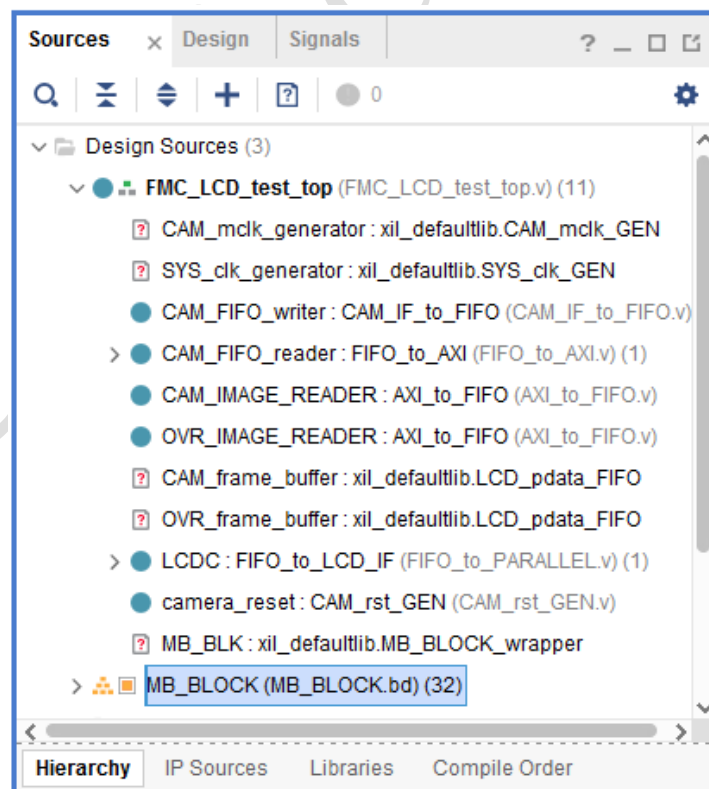
하기와 같이 Tcl Script 가 실행 된다.



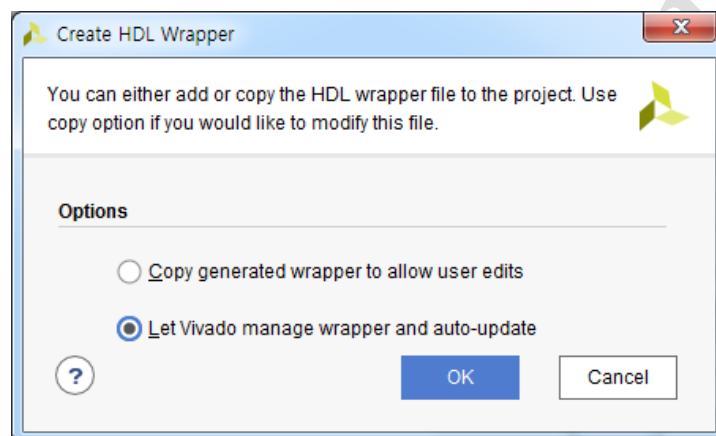
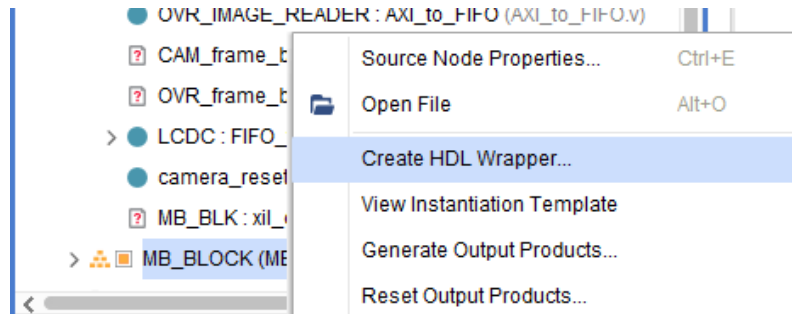
3) 상기 작업이 완료 되면 하기와 같이 IPI 창에 디자인 구성이 완료 된다.



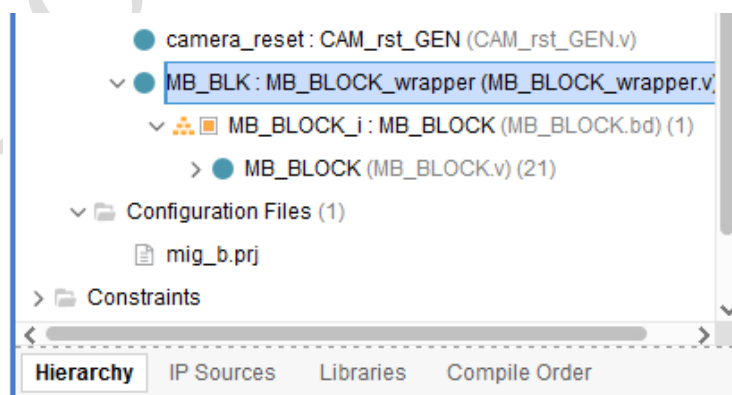
작업이 완료된 후 Source 창을 보면 하기와 같이 MB_BLOCK.bd 파일이 올라와 있는 것을 확인할 수 있다.



- 4) 하기의 그림과 같이 MB_BLOCK.bd 를 우 클릭 → Create HDL Wrapper... → OK 선택



하기와 같이 MB_BLOCK.bd 블록에 HDL Wrapper 가 붙어 Top 디자인과 같이 구성된 모습을 확인할 수 있다.



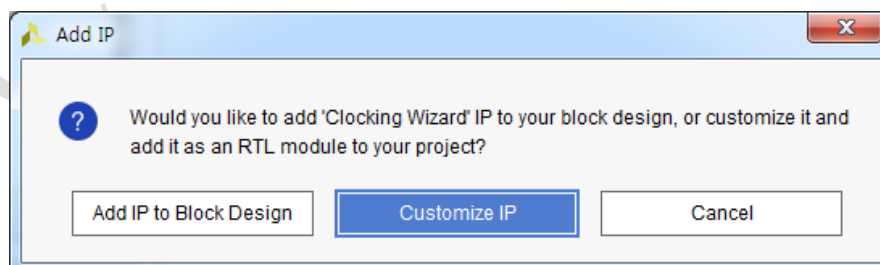
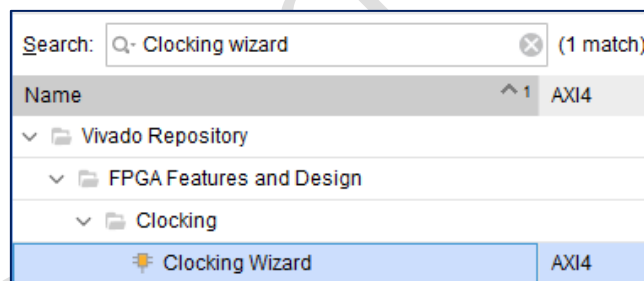
2.2.5 IP Catalog 이용 → IP 구성하기

- 이전 HDL 을 받는 작업 중 하기 이름의 디자인이 빠져 있는 것을 확인할 수 있으며, 다음의 작업을 통해 각 IP 들을 구성한다.

구성해야 할 IP 리스트	1) CAM_mclk_GEN 2) SYS_clk_GEN 3) LCD_pdata_FIFO 4) CAM_data_fifo
---------------	--

1) CAM_mclk_GEN IP 구성 작업

- Vivado Flow Navigator 창에 있는 IP Catalog 를 클릭 → IP Catalog 창에서 Clocking Wizard 를 검색 및 더블클릭 → Customize IP 선택



- Clocking Wizard 창이 나타나면 하기의 내용을 입력한다.

옵션 리스트		입력내용
Component Name	-	CAM_mclk_GEN
Output Clocks 탭	Port Name	CAM_mclk
	Requested	27
	Reset Type	Active Low

Component Name: CAM_mclk_GEN

Tab: Output Clocks

The phase is calculated relative to the active input clock.

Output Clock	Port Name	Output Freq (MHz)	
		Requested	Actual
<input checked="" type="checkbox"/> clk_out1	CAM_mclk	27	27.00000

Reset Type

☐ Active High ☒ Active Low

- 하기 창에서 Generate 를 클릭한다. 하기 작업은 상기에서 작업한 내용을 기반으로 IP 를 구성하는 과정 이다.

Generate Output Products

The following output products will be generated.

Preview

- CAM_mclk_GEN.xci (OOC per IP)
 - Instantiation Template
 - Synthesized Checkpoint (.dcp)
 - Structural Simulation

Synthesis Options

☐ Global ☒ Out of context per IP

Run Settings

Number of jobs: 8

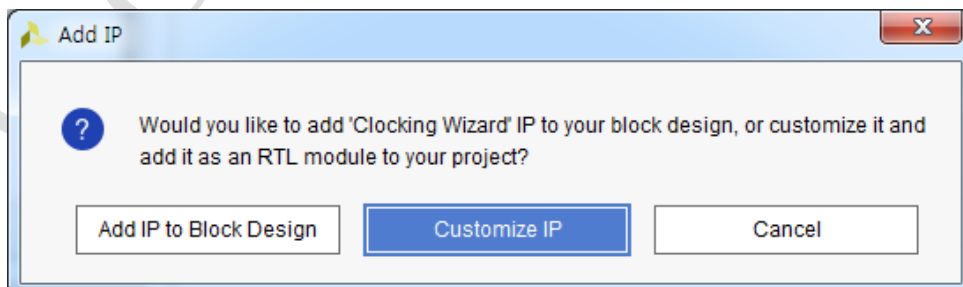
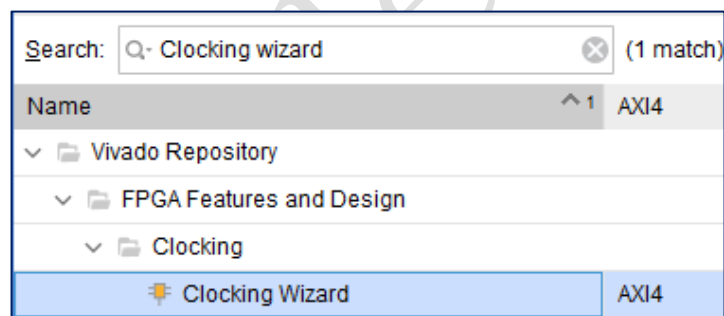
Buttons: ? Apply Generate Skip

작업이 완료되면 CAM_mclk_GEN 블록이 나타난 것을 확인 할 수 있다.



2) SYS_clk_GEN 블록 구성

- Vivado Flow Navigator 창에 있는 IP Catalog 를 클릭 →
- IP Catalog 창에서 Clocking Wizard 를 검색 및 더블클릭
- Customize IP 선택



- Clocking Wizard 창이 나타나면 하기의 내용을 입력한다.

옵션 리스트		입력내용
Component Name	-	SYS_clk_GEN
Output Clock 탭	Port Name	Requested
Clk_out1	MB_blk_clk	100.000
Clk_out2	LCD_clk	33.3333
Clk_out3	DDR_ref_clk	200.000
Reset Type	-	Active Low

Component Name **SYS_clk_GEN**

Clocking Options | **Output Clocks** | Port Renaming | MMCM Settings

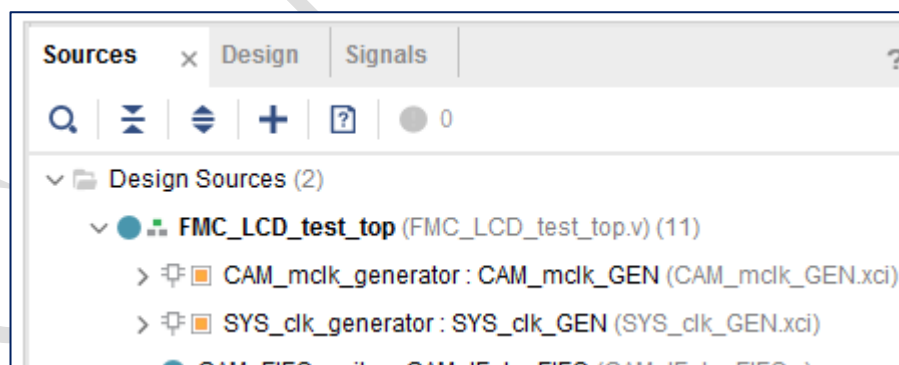
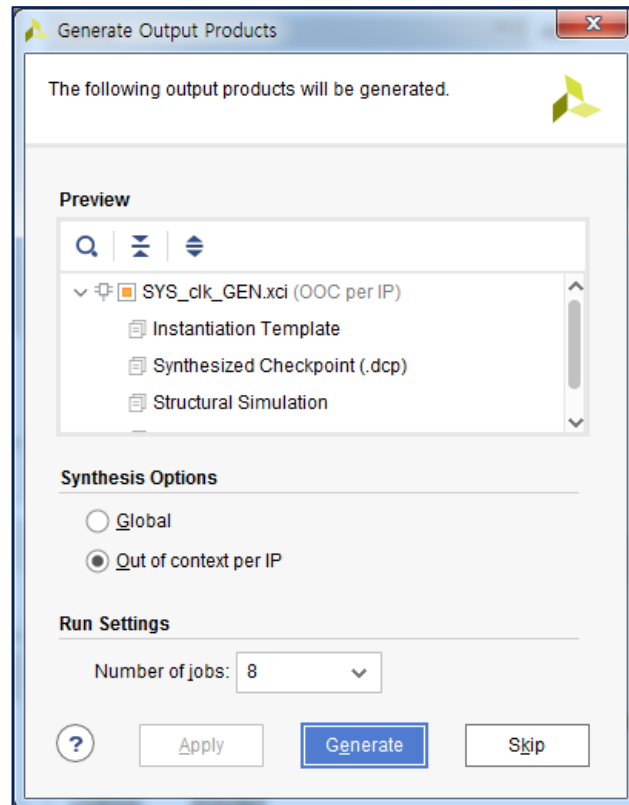
The phase is calculated relative to the active input clock.

Output Clock	Port Name	Output Freq (MHz)	
		Requested	Actual
<input checked="" type="checkbox"/> clk_out1	MB_blk_clk	100.000	100.00000
<input checked="" type="checkbox"/> clk_out2	LCD_clk	33.3333	33.33333
<input checked="" type="checkbox"/> clk_out3	DDR_ref_clk	200.000	200.00000

Reset Type

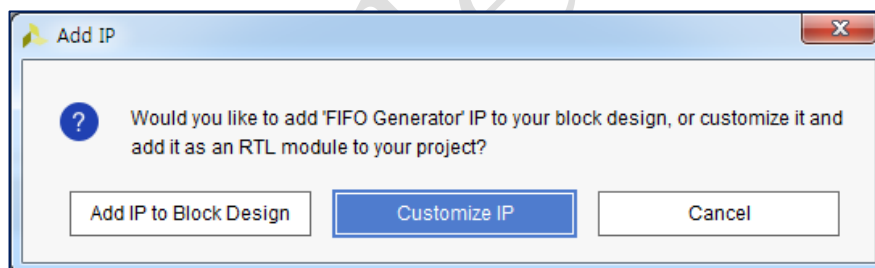
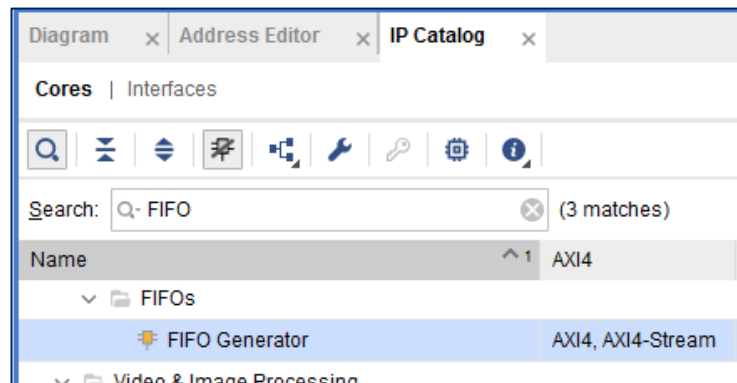
☐ Active High ☒ Active Low

- 하기 창에서 Generate 를 진행하면 다음과 같이 SYS_clk_GEN 블록이 나타나는 것을 확인할 수 있다.



3) LCD_pdata_FIFO IP 구성

- Vivado Flow Navigator 창에 있는 IP Catalog 를 클릭 →
- IP Catalog 창에서 "FIFO" 를 검색 및 더블클릭
- Customize IP 선택



- Clocking Wizard 창이 나타나면 하기의 내용을 입력한다.

옵션 리스트		입력내용(붉은색)
Basic	Interface Type	Native
	Fifo Implementation	Independent Clocks Block RAM
	Synchronization Stages	2
Native Ports	Component Name	LCD_pdata_FIFO
	Read Mode	Standard FIFO
	Data Port Parameters	Write Width: 32 Write Depth: 1024 Read Width: 16
	Initialization	Reset Pin: v Reset Type: Asynchronous Reset Enable Safety Circuit: v Full Flags Reset Value: 1 Dout Reset Value: 0
Status Flags	Programmable Flags	Programmable Full Type: Single Programmable Full Threshold Constant Full Threshold Asset Value: 910 Programmable Empty Type: Single Programmable Empty Threshold Constant Empty Threshold Asset Value: 1800
Data Counts	Write Data Count Width	v (enable) 10

Component Name: LCD_pdata_FIFO

Basic Native Ports Status Flags Data Counts Summary

Interface Type

☒ Native ☐ AXI Memory Mapped ☐ AXI Stream

Fifo Implementation: Independent Clocks Block RAM

Synchronization Stages: 2

FIFO Implementation Options

Supported Features

	Memory Type	(1)	(2)	(3)	(4)	(5)
Common Clock (CLK)	Block RAM	✓	✓		✓	✓
Common Clock (CLK)	Distributed RAM		✓			
Common Clock (CLK)	Shift Register					
Common Clock (CLK)	Built-in FIFO		✓	✓	✓	✓
Independent Clocks (RD_CLK, WR_CLK)	Block RAM	✓	✓		✓	✓

LCD_pdata_FIFO의 Basic

Component Name: LCD_pdata_FIFO

Basic Native Ports Status Flags Data Counts Summary

Read Mode

☒ Standard FIFO ☐ First Word Fall Through

Data Port Parameters

Write Width: 32 (1,2,3,...,1024)

Write Depth: 1024 (Actual Write Depth: 1023)

Read Width: 16

Read Depth: 2048 (Actual Read Depth: 2046)

ECC, Output Register and Power Gating Options

☐ ECC (Hard ECC) ☐ Single Bit Error Injection ☐ Double Bit Error Injection

☐ ECC Pipeline Reg ☐ Dynamic Power Gating

☐ Output Registers (Embedded Registers)

Initialization

☒ Reset Pin ☒ Enable Reset Synchronization ☒ Enable Safety Circuit

Reset Type: Asynchronous Reset

Full Flags Reset Value: 1

☒ Dout Reset Value: 0 (Hex)

Read Latency: 1

LCD_pdata_FIFO의 Native Port

Component Name: LCD_pdata_FIFO

Basic | Native Ports | Status Flags | Data Counts | Summary

Optional Flags

☐ Almost Full Flag ☐ Almost Empty Flag

Handshaking Options

Write Port Handshaking

☐ Write Acknowledge Active High ☐ Overflow Active High

Read Port Handshaking

☐ Valid Flag Active High ☐ Underflow Flag Active High

Programmable Flags

Programmable Full Type: Single Programmable Full Threshold Constant

Full Threshold Assert Value: 910 [3 - 1021]

Full Threshold Negate Value: 909 [2 - 909]

Programmable Empty Type: Single Programmable Empty Threshold Constant

Empty Threshold Assert Value: 1800 [2 - 2043]

Empty Threshold Negate Value: 1801 [1801 - 2044]

LCD_pdata_FIFO의 Status Flag

Component Name: LCD_pdata_FIFO

Basic | Native Ports | Status Flags | Data Counts | Summary

Data Count Options

☒ More Accurate Data Counts

☐ Data Count

Data Count Width: 10 [1 - 10]

☒ Write Data Count (Synchronized with Write Clk)

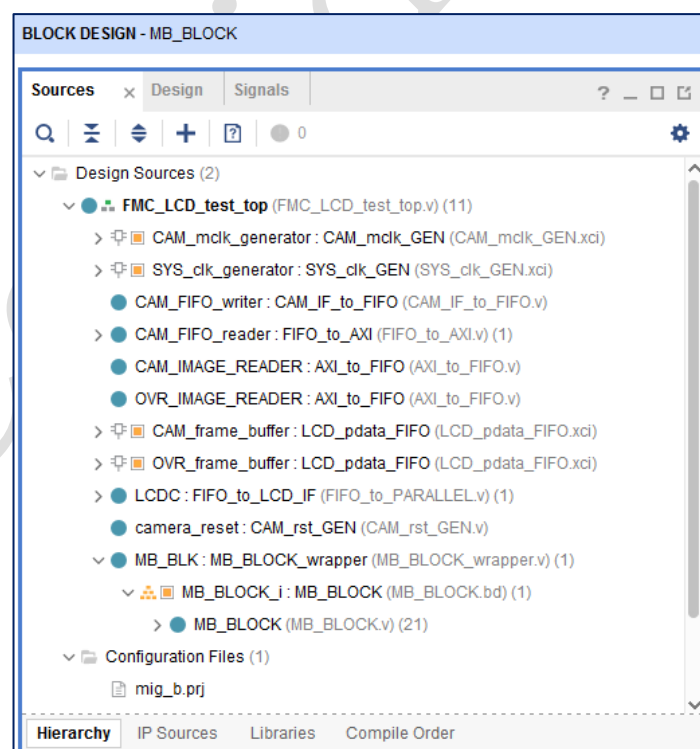
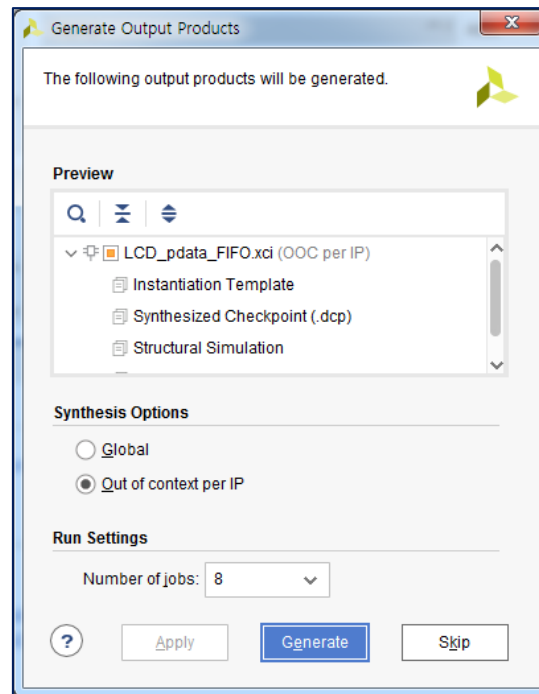
Write Data Count Width: 10 [1 - 11]

☐ Read Data Count (Synchronized with Read Clk)

Read Data Count Width: 12 [1 - 12]

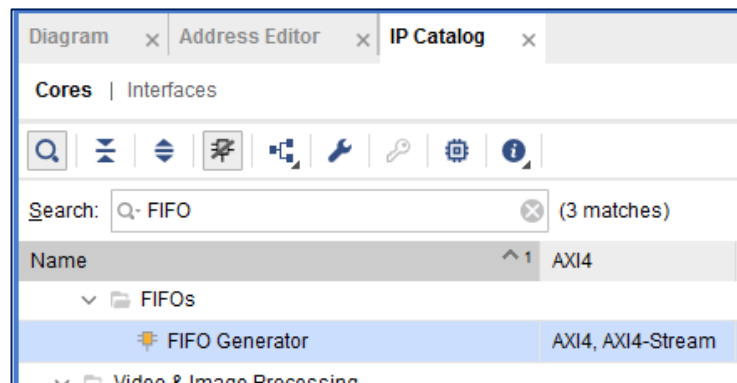
LCD_pdata_FIFO의 Data Counts

- 상기의 내용입력이 완료되면 OK 버튼을 누른 후 하기와 같이 Generate 한다
Generate가 완료되면 Sources 창에서 IP 생성이 완료된 것을 확인할 수 있다.



4) CAM_Pdata_sync_FIFO

- Vivado Flow Navigator 창에 있는 IP Catalog 를 클릭 →
- IP Catalog 창에서 "FIFO" 를 검색 및 더블클릭
- Customize IP 선택



- Clocking Wizard 창이 나타나면 하기의 내용을 입력한다.

옵션 리스트		입력내용(붉은색)
Basic	Interface Type	Native
	Fifo Implementation	Independent Clocks Block RAM
	Synchronization Stages	4
Native Ports	Component Name	CAM_pdata_sync_FIFO
	Read Mode	First Word Fall Through
	Data Port Parameters	Write Width: 16 Write Depth: 1024 Read Width: 32
	ECC, Output Register and Power Gating Options	v (Output Registers) Embedded Registers
	Initialization	Reset Pin: v Enable Reset Synchronization: v Enable Safety Circuit: v Reset Type: Asynchronous Reset Full Flags Reset Value: 1 Dout Reset Value: v, 0
Status Flags	Read Port Handshaking	v (Valid Flag) : Active High
	Programmable Flags	Programmable Full Type: Multiple Programmable Full Threshold Constants Full Threshold Asset Value: 800 Full Threshold Negative Value: 150 Programmable Empty Type: Single Programmable Empty Threshold Constant Empty Threshold Asset Value: 64
Data Counts	Write Data Count Width	v (enable) 10
	Read Data Count Width	v (enable) 9

Component Name CAM_pdata_sync_FIFO

Basic Native Ports Status Flags Data Counts Summary

Interface Type

☒ Native ☐ AXI Memory Mapped ☐ AXI Stream

Fifo Implementation Independent Clocks Block RAM

Synchronization Stages 4

FIFO Implementation Options

Supported Features

	Memory Type	(1)	(2)	(3)	(4)	(5)
Common Clock (CLK)	Block RAM	✓	✓		✓	✓
Common Clock (CLK)	Distributed RAM		✓			
Common Clock (CLK)	Shift Register					
Common Clock (CLK)	Built-in FIFO		✓	✓	✓	✓
Independent Clocks (RD_CLK, WR_CLK)	Block RAM	✓	✓		✓	✓
Independent Clocks (RD_CLK, WR_CLK)	Distributed RAM		✓			

CAM_pdata_sync_FIFO의 Basic

Component Name CAM_pdata_sync_FIFO

Basic Native Ports Status Flags Data Counts Summary

Read Mode

☐ Standard FIFO ☒ First Word Fall Through

Data Port Parameters

Write Width 16 1,2,3,...1024

Write Depth 1024 Actual Write Depth: 1027

Read Width 32

Read Depth 512 Actual Read Depth: 513

ECC, Output Register and Power Gating Options

☐ ECC Hard ECC ☐ Single Bit Error Injection ☐ Double Bit Error Injection

☐ ECC Pipeline Reg ☐ Dynamic Power Gating

☒ Output Registers Embedded Registers

Initialization

☒ Reset Pin ☒ Enable Reset Synchronization ☒ Enable Safety Circuit

Reset Type Asynchronous Reset

Full Flags Reset Value 1

☒ Dout Reset Value 0 (Hex)

Read Latency : 0

CAM_pdata_sync_FIFO의 Native Port

Component Name CAM_pdata_sync_FIFO

Basic Native Ports Status Flags Data Counts Summary

Optional Flags

☐ Almost Full Flag ☐ Almost Empty Flag

Handshaking Options

Write Port Handshaking

☐ Write Acknowledge Active High ☐ Overflow Active High

Read Port Handshaking

☒ Valid Flag Active High ☐ Underflow Flag Active High

Programmable Flags

Programmable Full Type Multiple Programmable Full Threshold Constants

Full Threshold Assert Value 800 [15 - 1023]

Full Threshold Negate Value 150 [14 - 799]

Programmable Empty Type Single Programmable Empty Threshold Constant

Empty Threshold Assert Value 64 [4 - 510]

Empty Threshold Negate Value 65 [65 - 511]

CAM_pdata_sync_FIFO의 Status Flags

Component Name CAM_pdata_sync_FIFO

Basic Native Ports Status Flags Data Counts Summary

Data Count Options

☐ More Accurate Data Counts

☐ Data Count

Data Count Width 10 [1 - 10]

☒ Write Data Count (Synchronized with Write Clk)

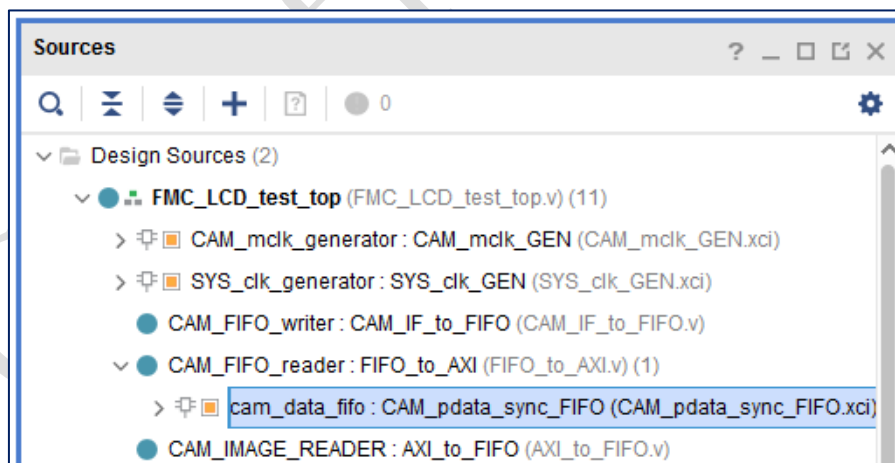
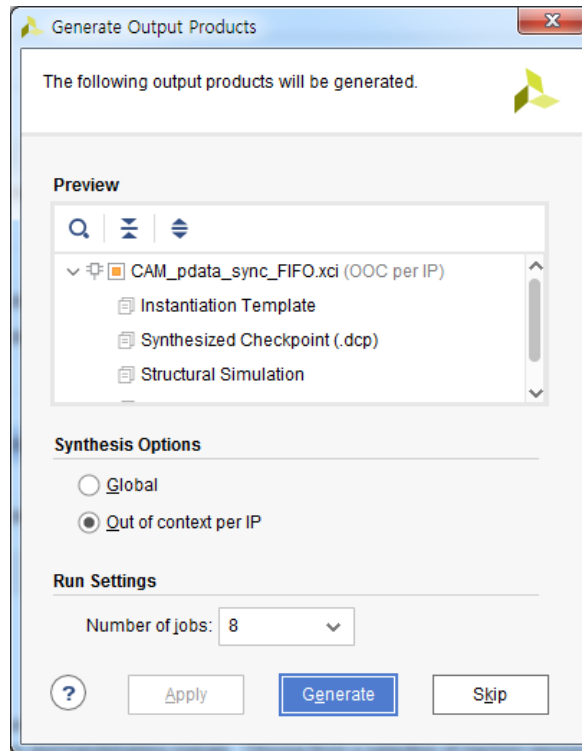
Write Data Count Width 10 [1 - 10]

☒ Read Data Count (Synchronized with Read Clk)

Read Data Count Width 9 [1 - 9]

CAM_pdata_sync_FIFO의 Data Counts

- 상기의 내용입력이 완료되면 OK 버튼을 누른 후 하기와 같이 Generate 한다
Generate가 완료되면 Sources 창에서 IP 생성이 완료된 것을 확인할 수 있다

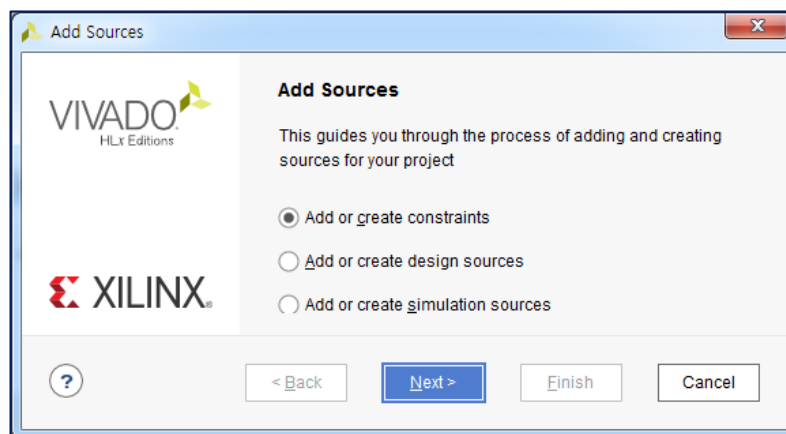


2.2.6 핀 정보 셋팅 (XDC 파일 Import)

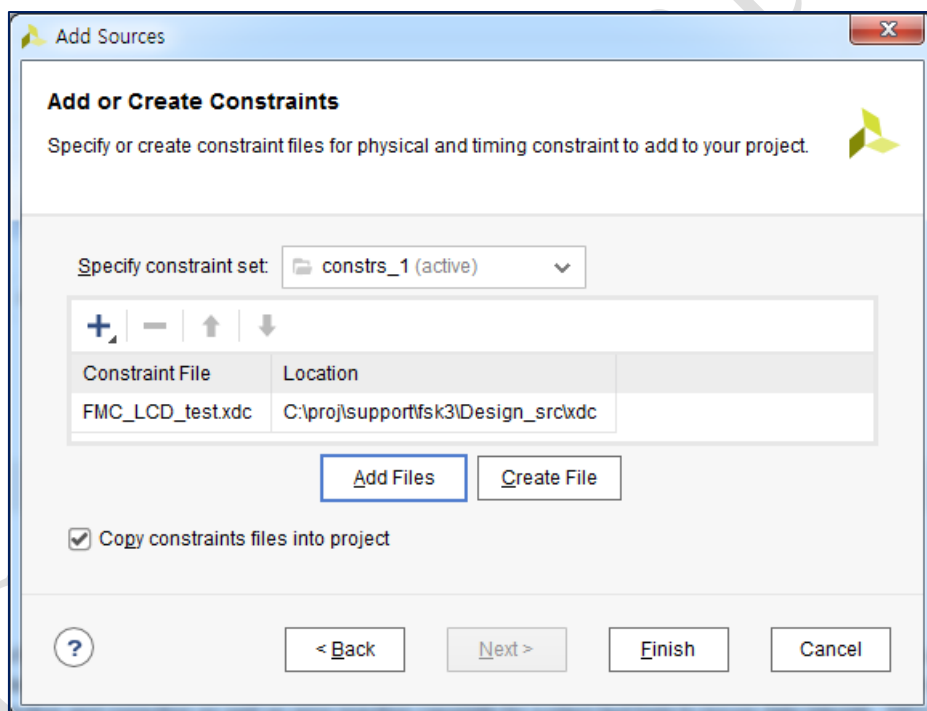
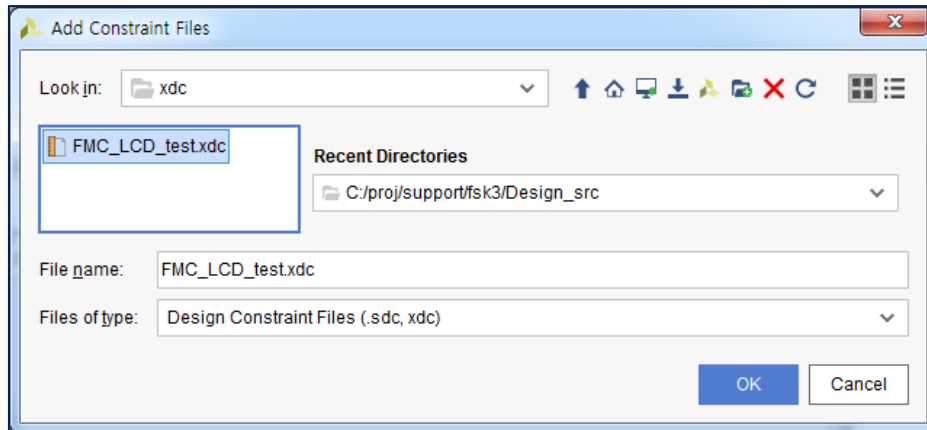
1) XDC 파일 Import 작업

- 당사에서 제공하는 FMC_LCD_test.xdc 파일을 받아들인다.

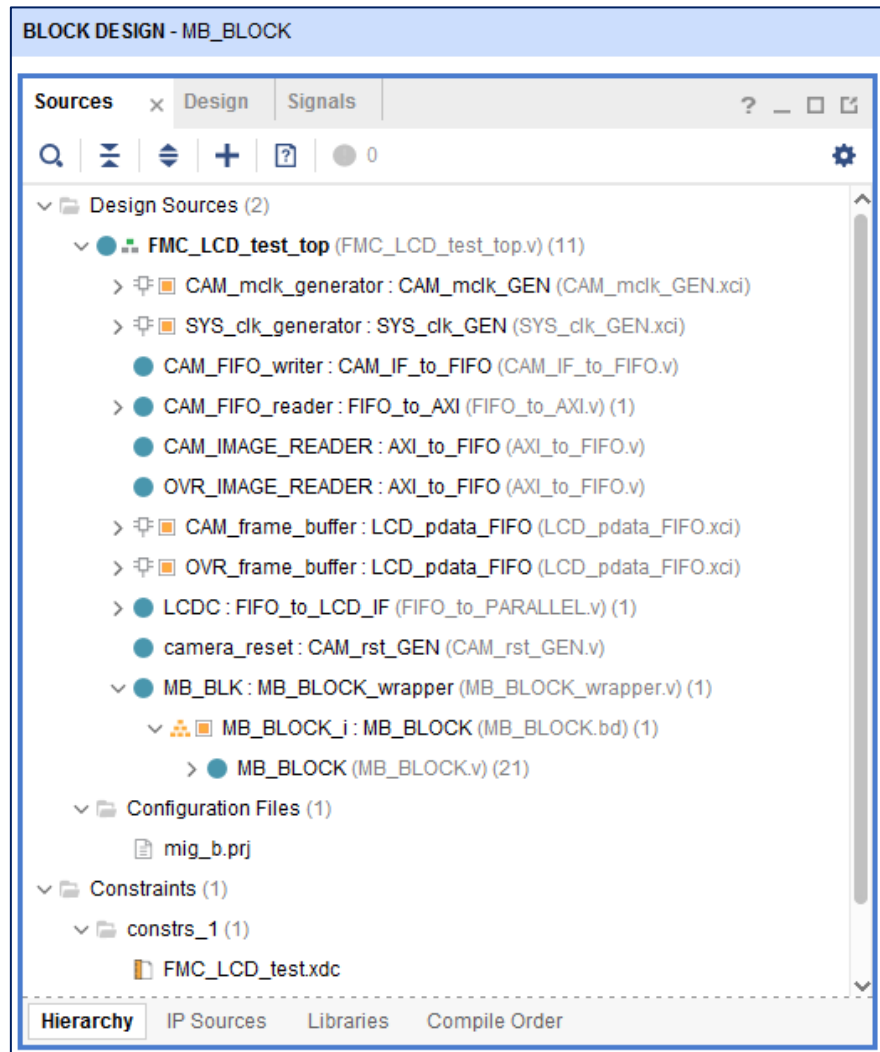
하기 그림에 있는 Add Sources 선택 → Add or create constraints → Next
→ Add Files 선택



- 하기와 같이 당사에서 제공하는 FMC_LCD_test.xdc 파일 선택 후 OK →
하기 두번째 그림에서 Finish 선택



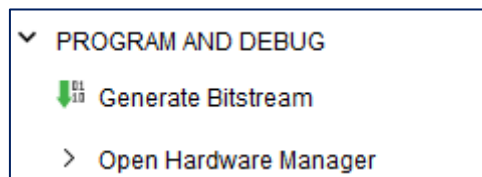
- 상기의 작업이 완료되면 하기의 그림과 같이 Constraints 탭에 FMC_LCD_test.xdc 파일이 나타난다.



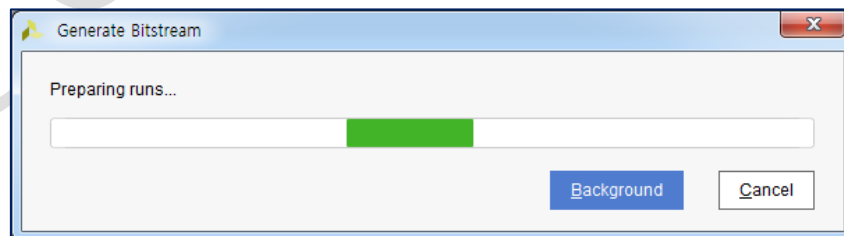
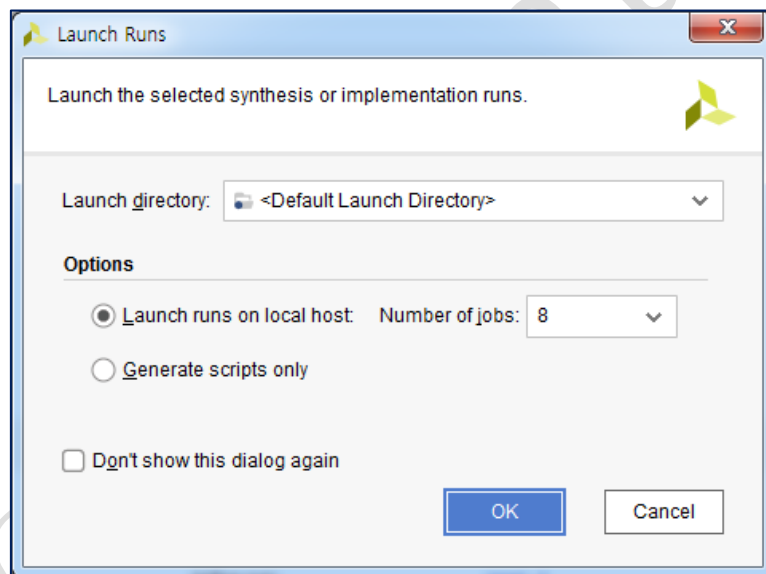
2.2.7 Bit File 생성 및 Hardware 정보 내보내기 (xsa 파일 내보내기)

1) Generate Bitstream 실행

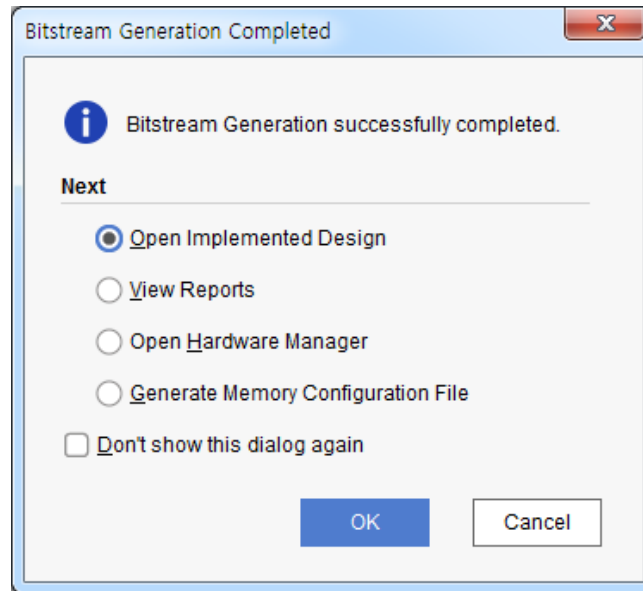
- 지금까지의 작업으로, HW 디자인 관련 작업이 모두 완료 되었다.
이에 FPGA HW 영역에 대한 다운로드 파일인 bit 파일을 생성하기 위해
하기와 Flow Navigator 창 하단에 있는 Generate Bitstream 을 실행한다.



- 하기 창에서 OK 를 클릭하면 Bitstream File 생성을 시작한다.

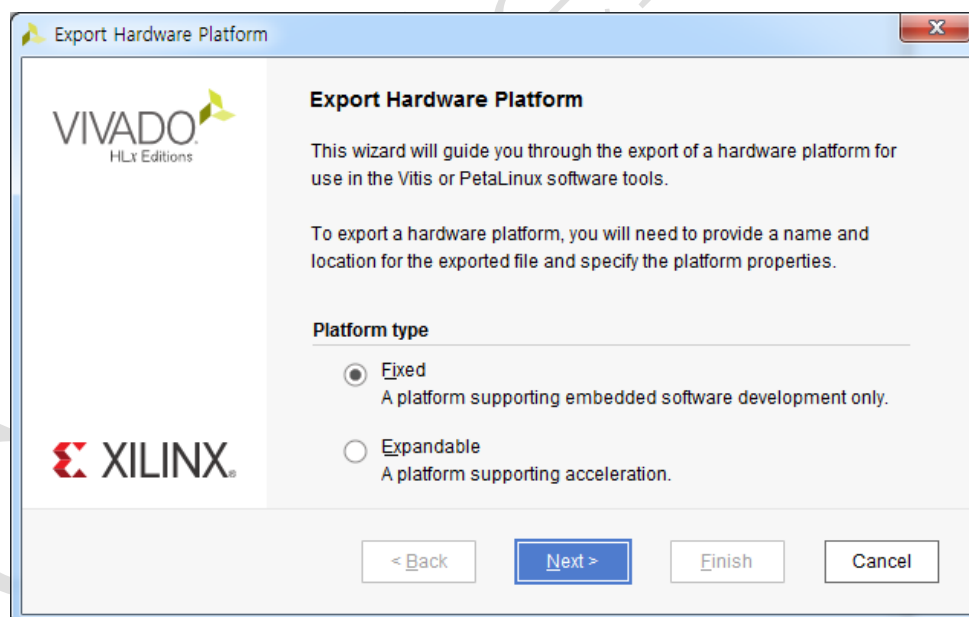
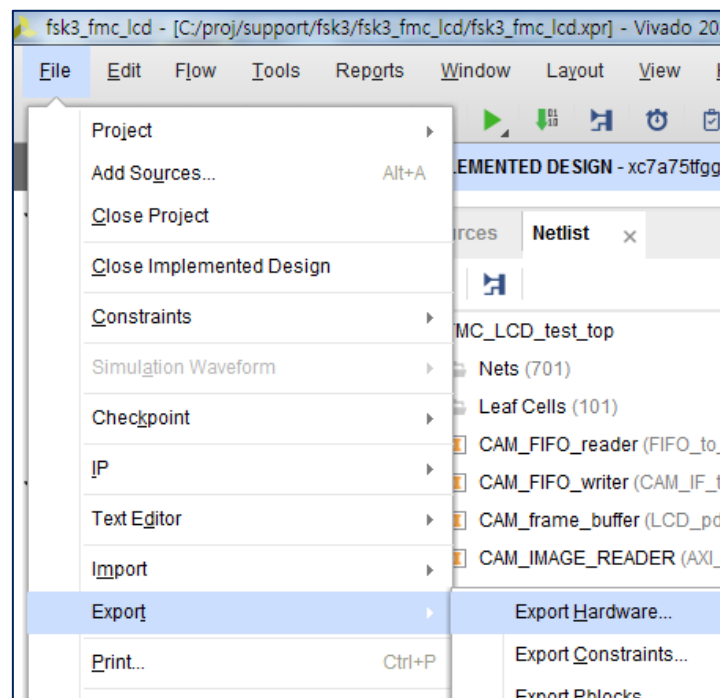


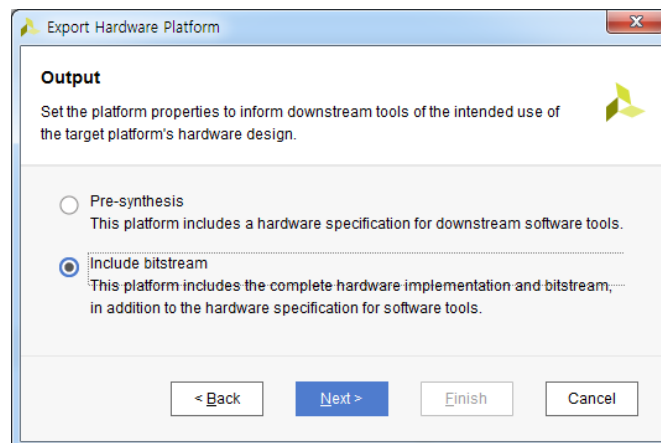
- Bit File 이 생성되고 하기 창이 나타나면 OK를 클릭한다.
- 하기 선택된 옵션의 내용은 Implementation 작업 후에 Implementation 작업에 대한 여러가지 리포트 파일을 생성할 수 있는 환경을 만들어 준다.



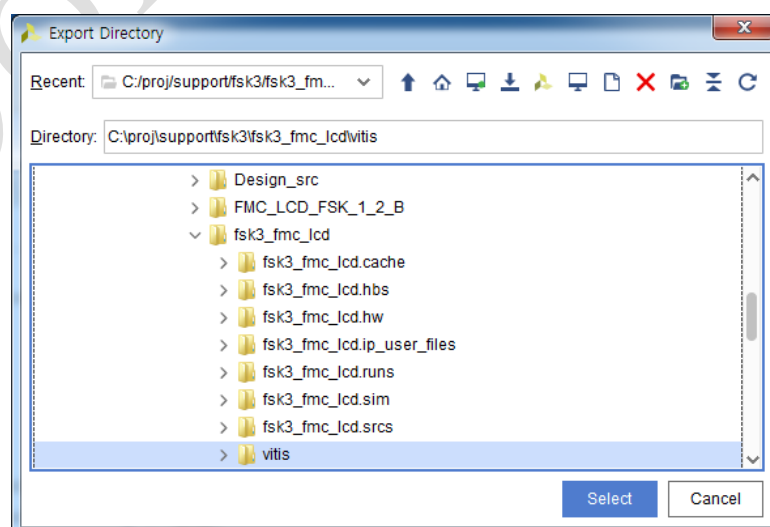
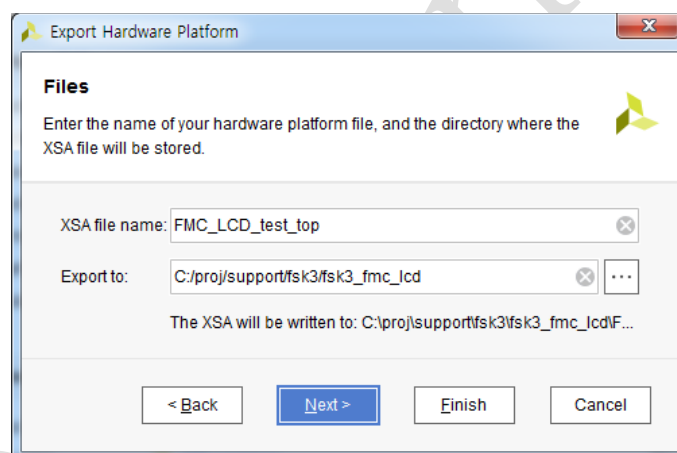
2) Hardware 정보 내보내기 (xsa 파일 내보내기)

- 본 디자인은 Hardware 디자인과 Software 디자인이 섞여 있다. HW의 기본적인 동작을 HDL로 디자인 하였으며, 카메라 컨트롤 등, 전체 동작에 대한 컨트롤을 SW 영역에서 진행 한다. SW영역에서 동작하는 디자인을 위해 HW 구성에 대한 정보가 담긴 .xsa 파일을 내보내게 된다.
- 일반적으로 프로세서들은 외부 Interface 를 Memory로 인식하므로, 여기서 작업하여 생성되는 내용은 Memory Map 형태로 HW 정보를 내보내게 된다.
- Hardware 정보를 내보내기 위한 폴더를 생성한다. 본자료에서는 현재 작업 중인 프로젝트 폴더에 "Vitis" 라는 폴더를 미리 만들어 진행 하도록 한다.
- Vivado 에서 하기와 같이 File → Export → Export Hardware... → Next (Fixed) → Include bitstream → Next

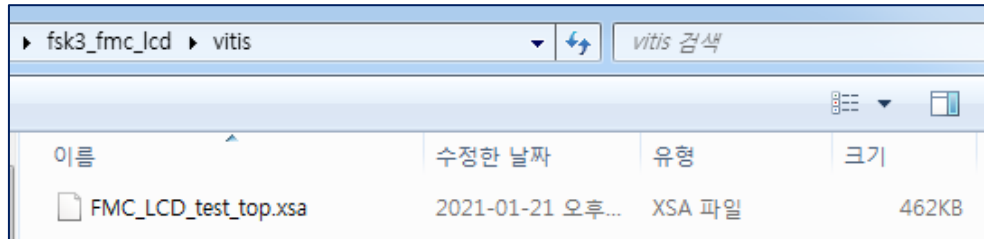




- 하기 창이 나타나면 [...] 를 클릭하여 상기에서 언급한 Vitis 폴더를 지정한 후 , Next → Finish를 클릭



- 상기 작업이 완료되면 하기와 같이 FMC_LCD_test_top.xsa 파일이 Vitis 폴더에 생성된다.

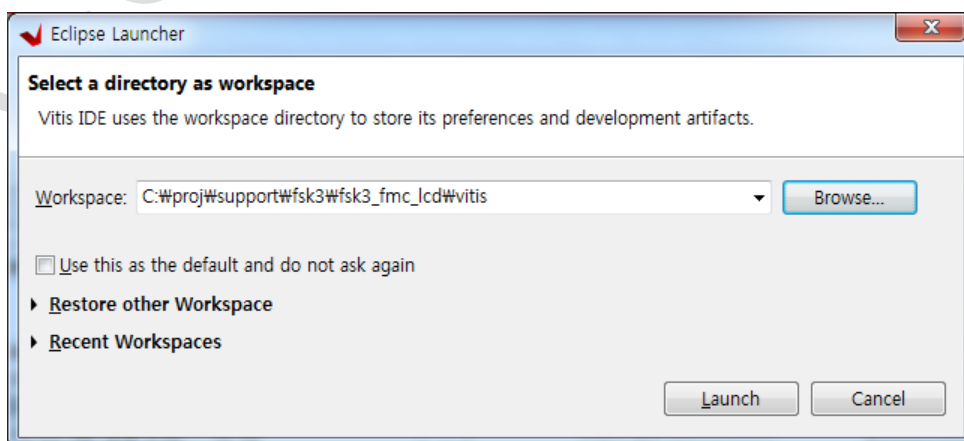


2.3 Vitis 기반의 SW 영역 디자인 실습

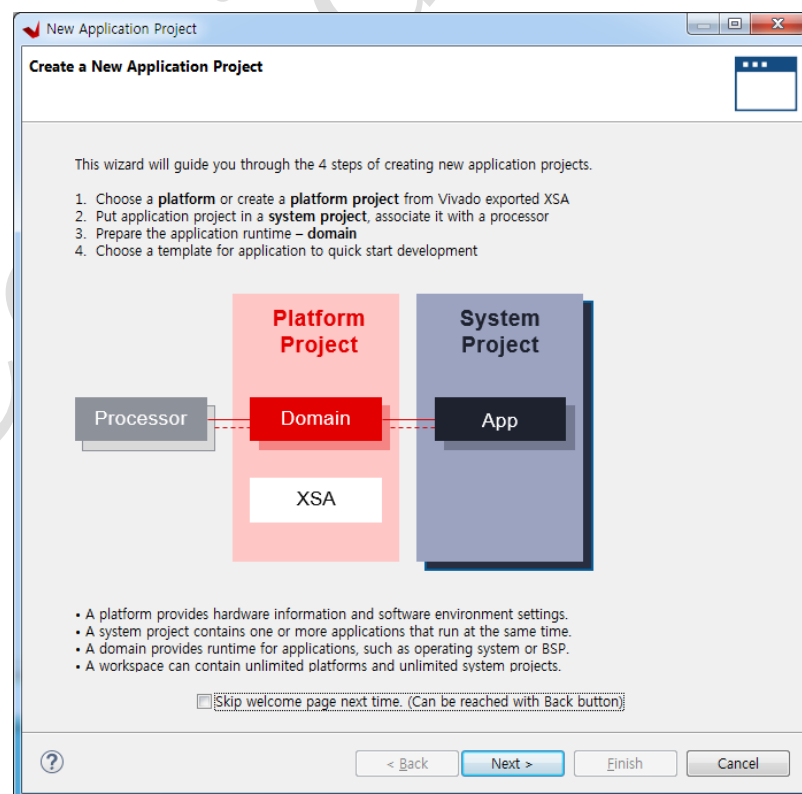
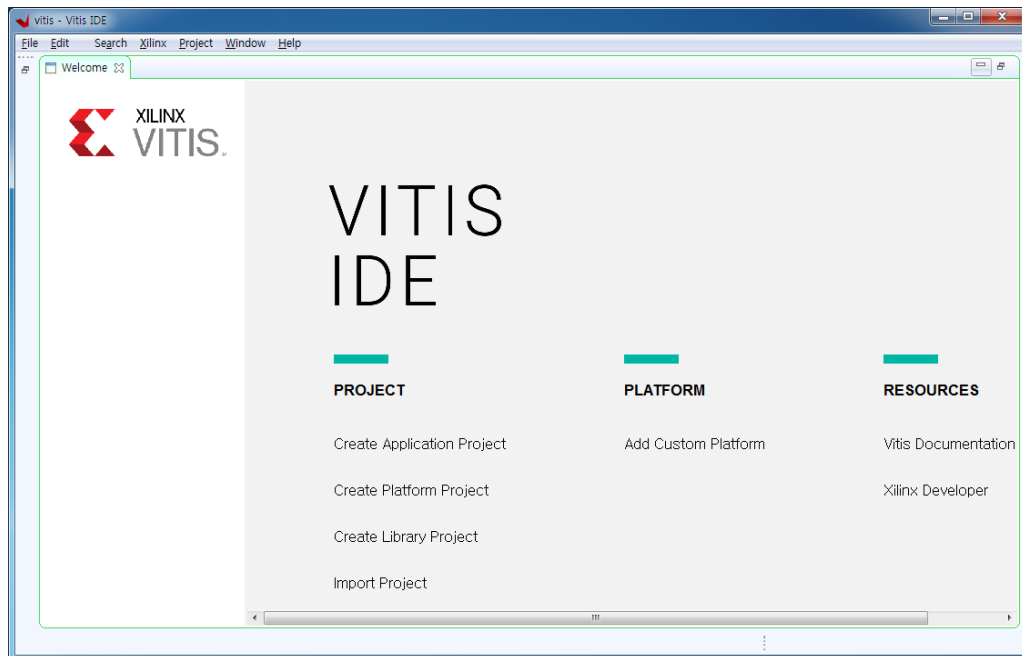
- 자료 도입부에서 언급했듯이, 당사에서 제공하는 하기 C코드를 활용하여 다음 작업을 진행한다.

C 코드	helloworld.c peri_iic_ctrl.c peri_iic_ctrl.h zn220_initialize_list.h
------	---

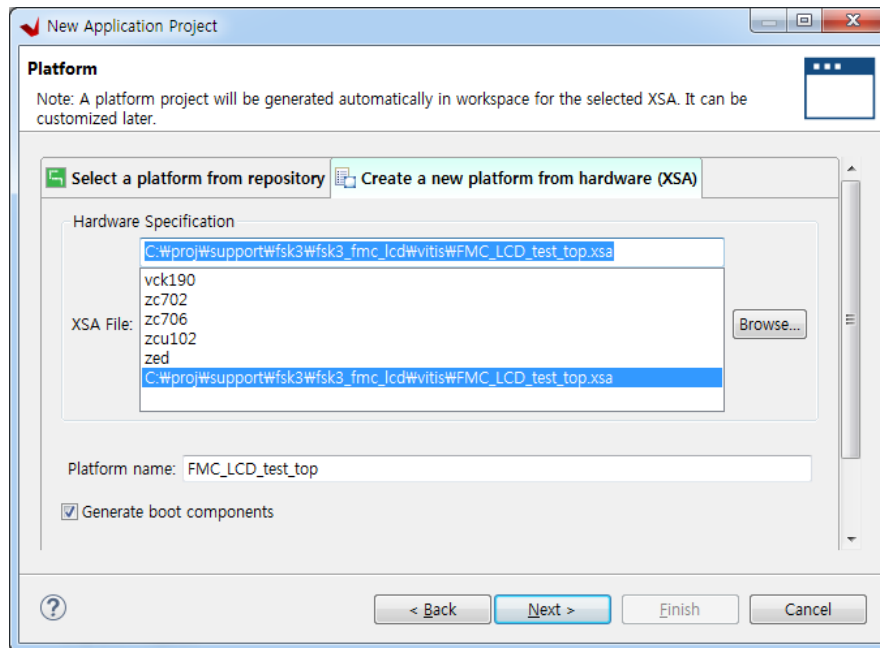
- 1) Vitis Project 실행하기
 - Vitis 2020.1 or 2020.2 버전을 실행한다.
 - 하기 그림이 나타나면 이전 작업에서 FMC_LCD_test_top.xsa 파일이 있는 폴더로 위치를 지정한 후 Launch 를 클릭한다.



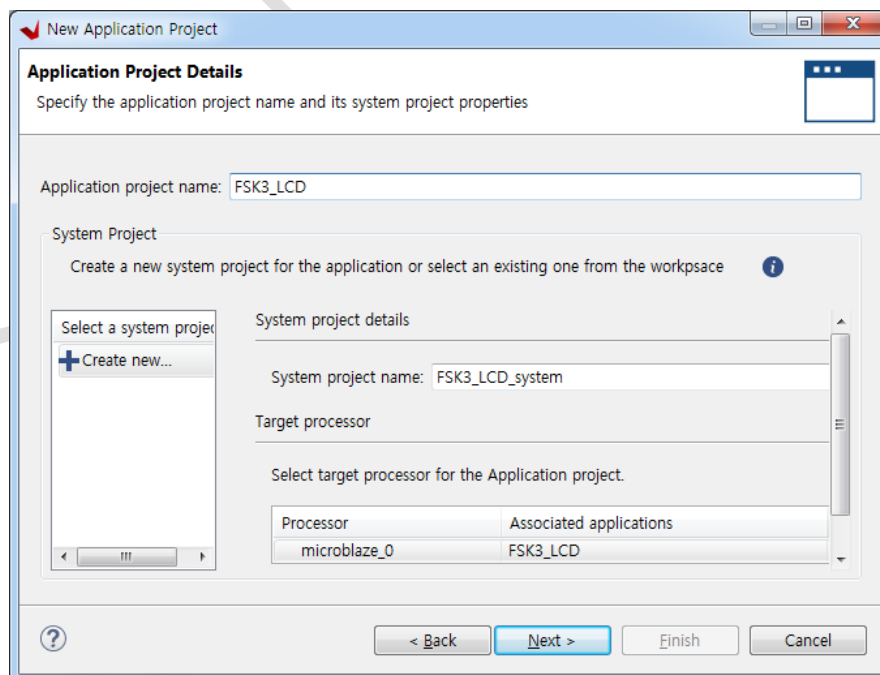
- 하기와 같이 Vitis 초기 화면이 나타나며, Create Application Project 를 선택하고 Next 를 클릭한다.



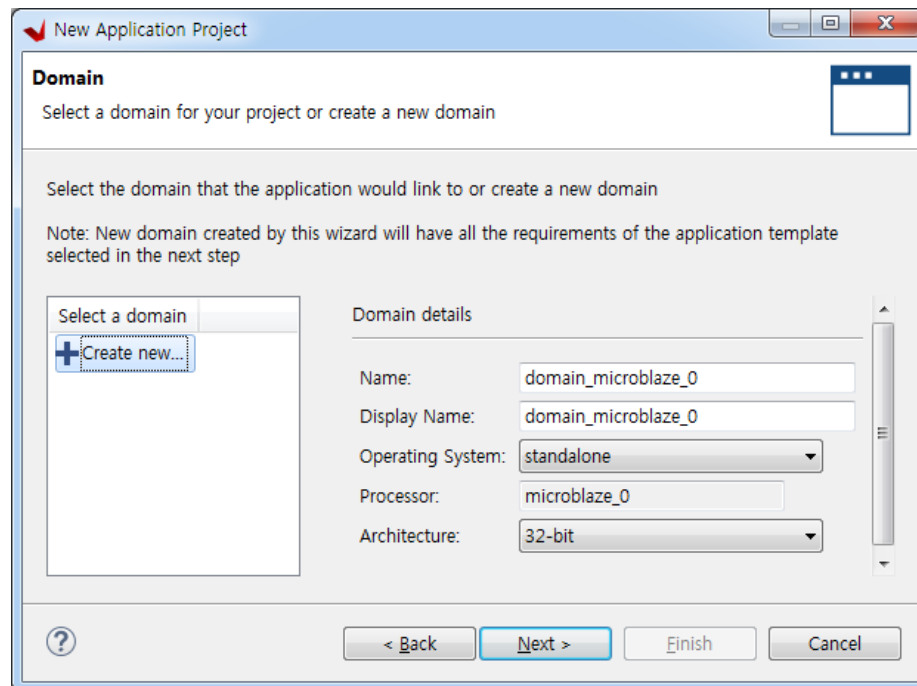
- 하기의 창에서 Browse... 를 선택한 후 이전에 xsa 파일을 저장했던 Vitis 폴더를 지정한 뒤 FMC_LCD_test_top.xsa 파일을 선택 및 Next 클릭



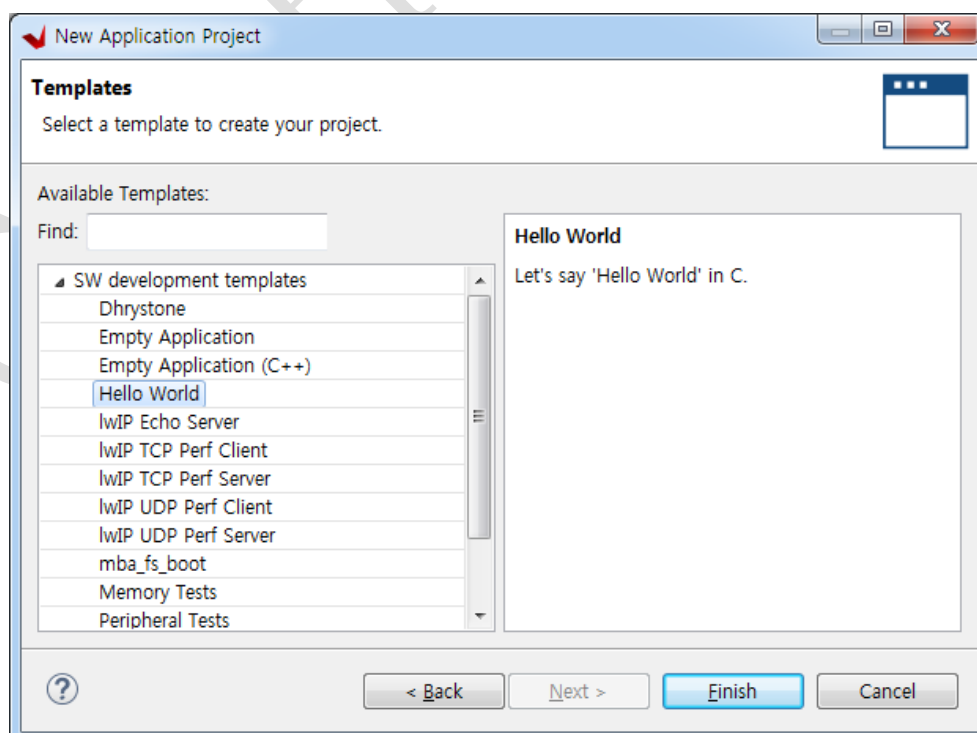
- 하기 창에서 Application Project Name 란에 FSK3_LCD를 입력하고 Next를 클릭한다.



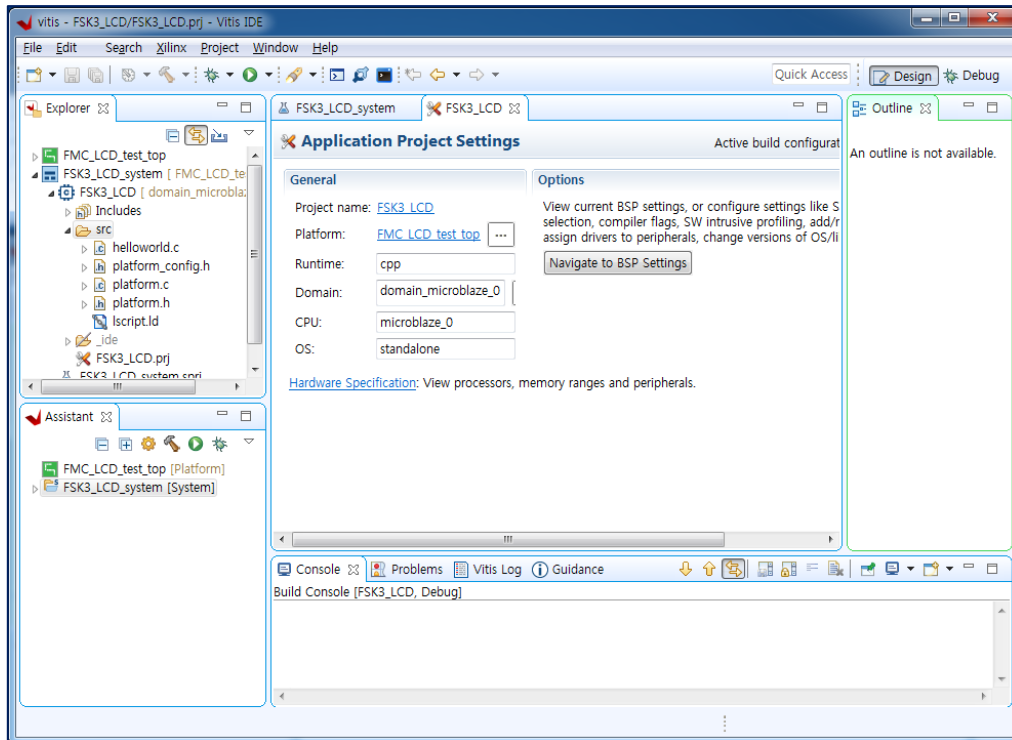
- 하기 창이 나타나면 Next 선택



- 하기 에서 Hello World 를 선택한 후 Finish 를 클릭한다.

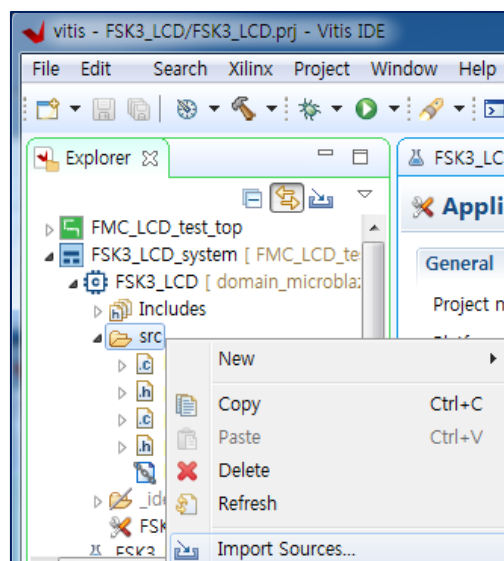


- 작업이 완료되면 하기의 그림과 같이 Vitis 프로젝트 화면이 나타난다.

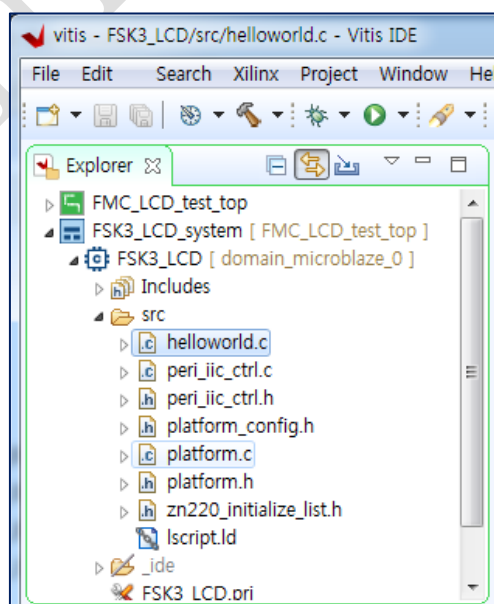
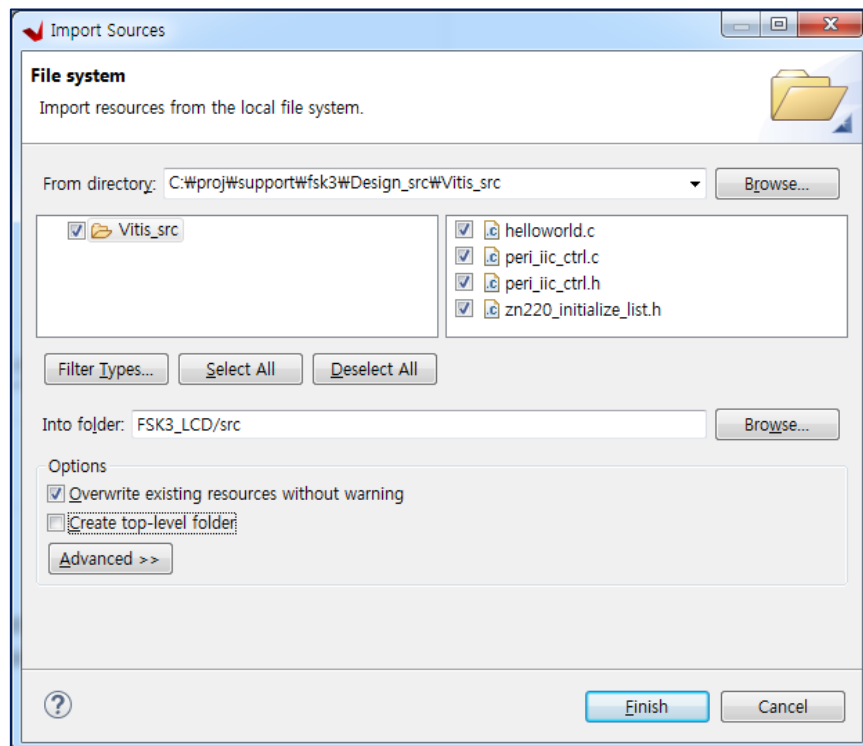


2) Import Source 작업

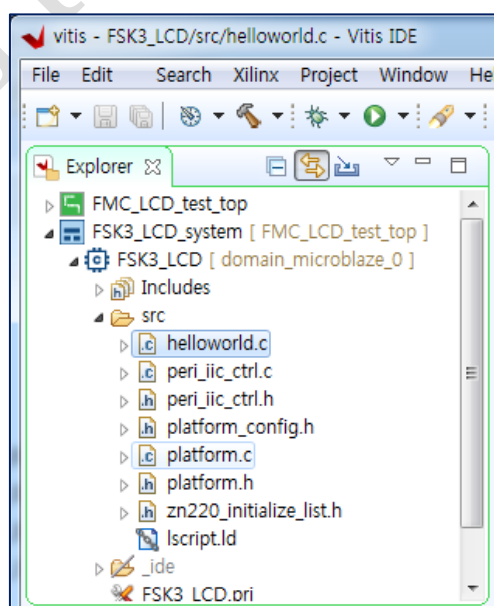
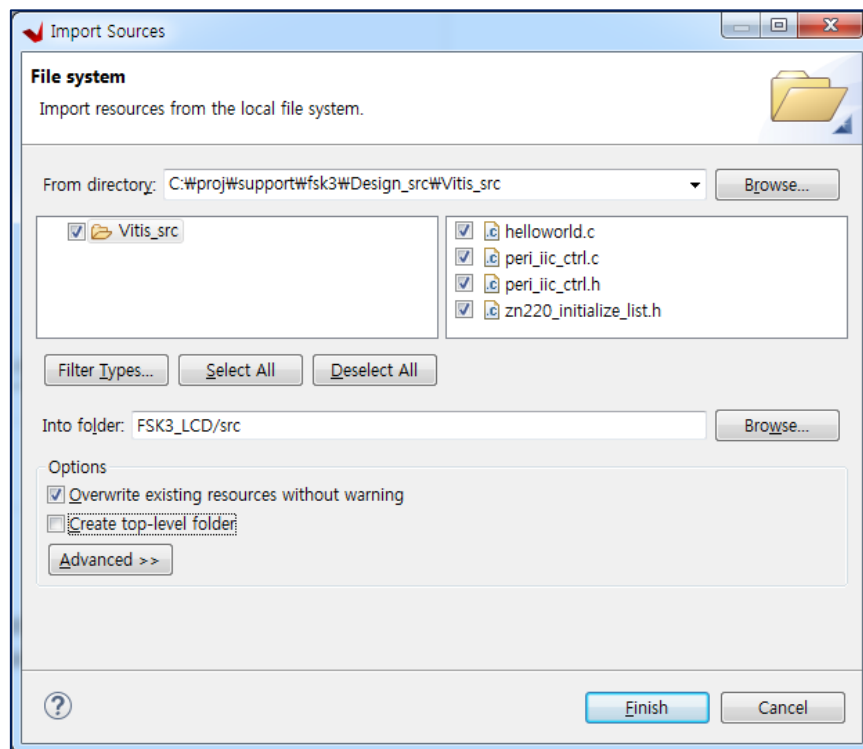
- 하기와 같이 Vitis Explorer 창에서 FSK3_LCD 하위에 있는 src 폴더를 선택 우클릭 후 Import Sources... 를 클릭한다.



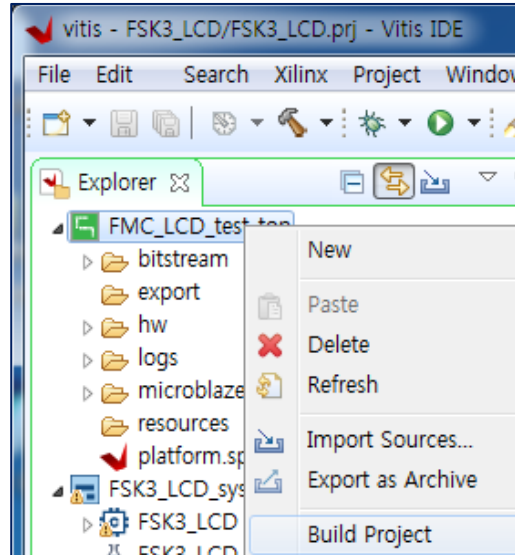
- 하기와 같이 Helloworld.c , peri_iic_ctrl.c , zn220_initialize_list.h , peri_iic_ctrl.h 코드가 나타나면 source의 체크박스를 하기의 그림과 같이 enable 한 후 Finish 를 선택하면 받아들이는 소스들이 src 폴더에 위치해 있는 것을 확인할 수 있다.



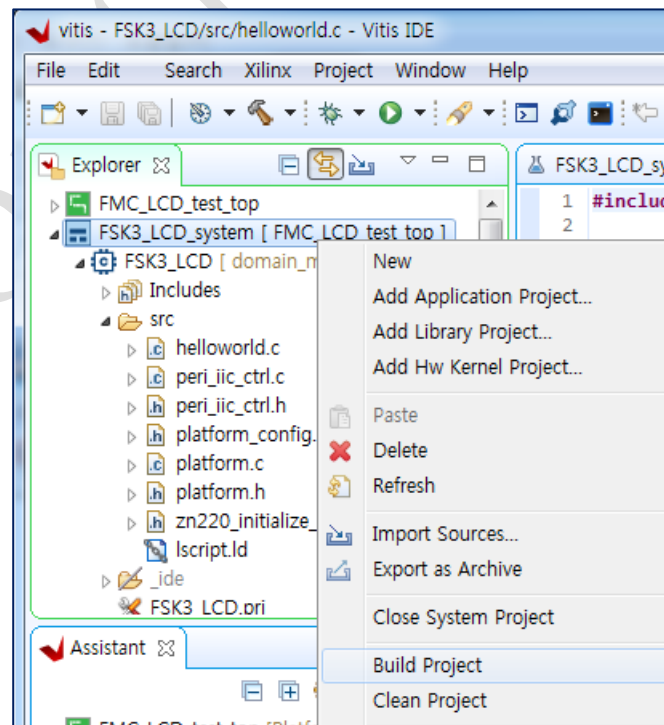
- 하기와 같이 Helloworld.c , peri_iic_ctrl.c , zn220_initialize_list.h , peri_iic_ctrl.h 코드가 나타나면 source의 체크박스를 하기의 그림과 같이 enable 한 후 Finish 를 선택하면 받아들이는 소스들이 src 폴더에 위치해 있는 것을 확인할 수 있다.



- FMC_LCD_test_top 을 우클릭 하여 Build Project를 선택한다.



- 상기 동작이 완료 되면, FSK3_LCD_system 의 Build Project를 진행한다. 이때, Vitis 의 버그이슈로 에러가 나타난다. 그에 대한 해결책을 하기와 같이 명시하였으며, 진행 후 FMC_LCD_camera 가 정상적으로 동작되는 것을 확인할 수 있다.



3) Vitis 컴파일 버그 해결 방법

- Vitis 에서 컴파일 시 에러가 발생하는 이유는 다음과 같다.

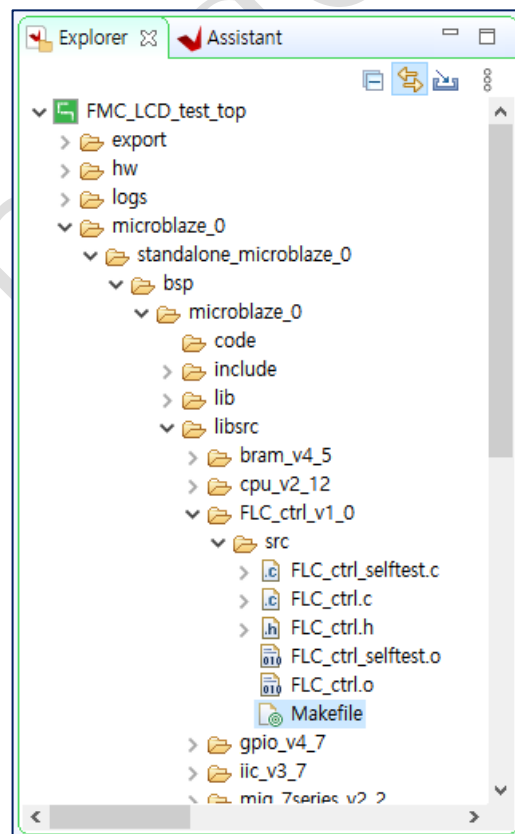
Vivado 에서 Vitis 로 작업하는 과정 중 Makefile 이 생성되는데, Makefile 의 설정 값에 오류가 있어 컴파일 에러가 발생한다.

- 본 디자인에서는 FPGA 로직에서 FLC_ctrl_v1.0 과 Touch_Intr_vtr1_1.0 IP 를 받아 작업했는데, 이 디자인과 관련되어 생성된 Makefile 을 하기의 경로에서 찾아 다음과 같이 수정한다

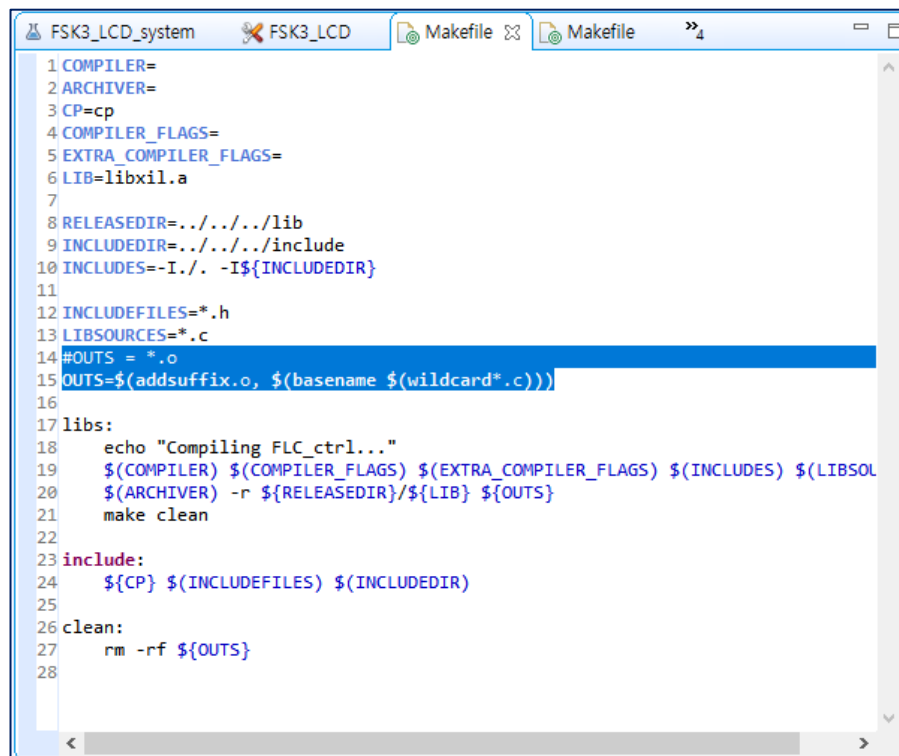
▶ **FMC_LCD_test_top** : microblaze_0 → standalone_microblaze_0 → bsp → microblaze_0 → libsrc → FLC_ctrl_1_0 → Makefile

▶ **FMC_LCD_test_top** : microblaze_0 → standalone_microblaze_0 → bsp → microblaze_0 → libsrc → Tocuh_Intr_ctrl_1_0 → Makefile

* FLC_ctrl_v1_0 의 Makefile



- 하기의 내용과 같이 FLC_ctrl_v1_0 과 Touch_intr_vtrl_v1_0에 있는 Makefile 을 열어서 다음과 같이 수정한다.
- Makefile 14번째 라인에 있는 OUTS = *.o 을 하기 내용으로 대체한다.
`OUTS = $(addsuffix .o, $(basename $(wildcard *.c)))`

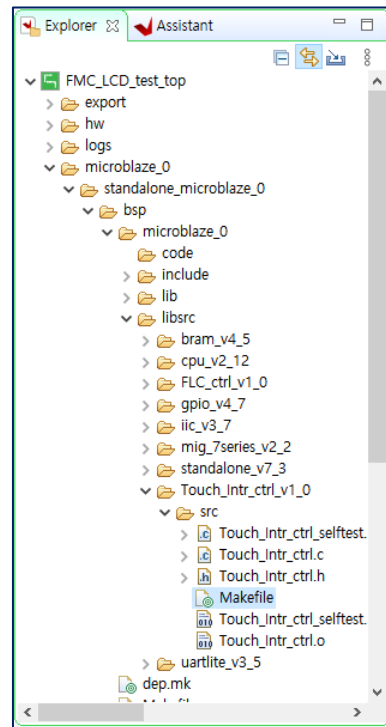


```

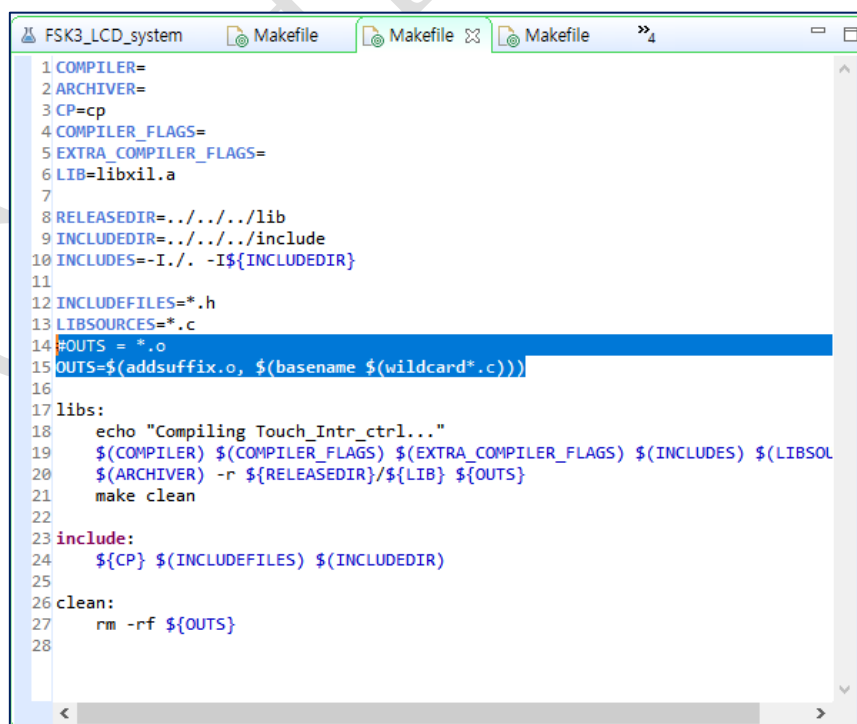
1 COMPILER=
2 ARCHIVER=
3 CP=cp
4 COMPILER_FLAGS=
5 EXTRA_COMPILER_FLAGS=
6 LIB=libxil.a
7
8 RELEASEDIR=../../lib
9 INCLUDEDIR=../../include
10 INCLUDES=-I./ -I${INCLUDEDIR}
11
12 INCLUDEFILES=*.h
13 LIBSOURCES=*.c
14 #OUTS = *.o
15 OUTS=$(addsuffix.o, $(basename $(wildcard*.c)))
16
17 libs:
18     echo "Compiling FLC_ctrl..."
19     $(COMPILER) $(COMPILER_FLAGS) $(EXTRA_COMPILER_FLAGS) $(INCLUDES) $(LIBSOURCES)
20     $(ARCHIVER) -r ${RELEASEDIR}/${LIB} ${OUTS}
21     make clean
22
23 include:
24     ${CP} $(INCLUDEFILES) $(INCLUDEDIR)
25
26 clean:
27     rm -rf ${OUTS}
28

```

* Touch_Intr_ctrl_v1_0 의 Makefile



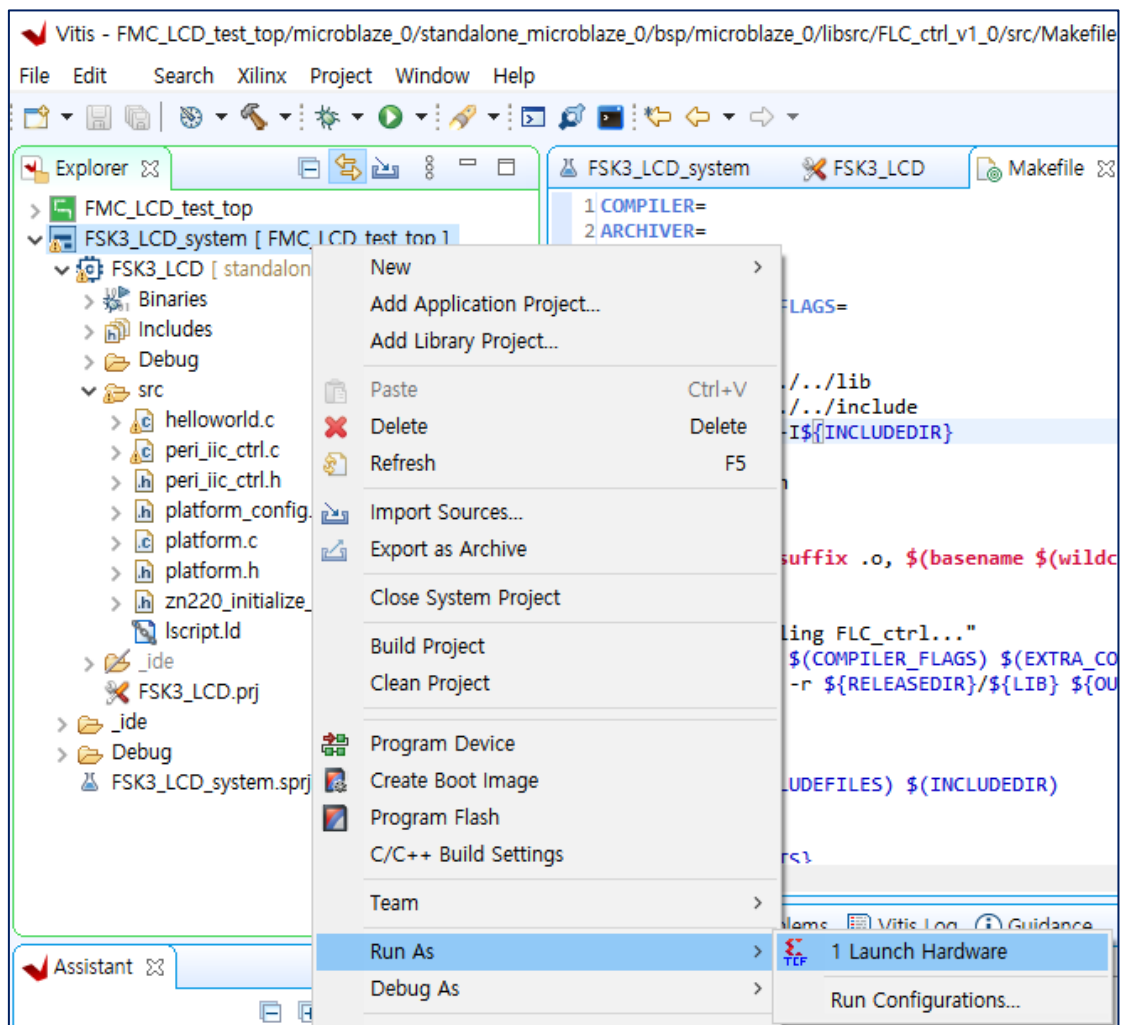
- Makefile 14번째 라인에 있는 OUTS = *.o 을 하기 내용으로 대체한다.
`OUTS = $(addsuffix .o, $(basename $(wildcard *.c)))`



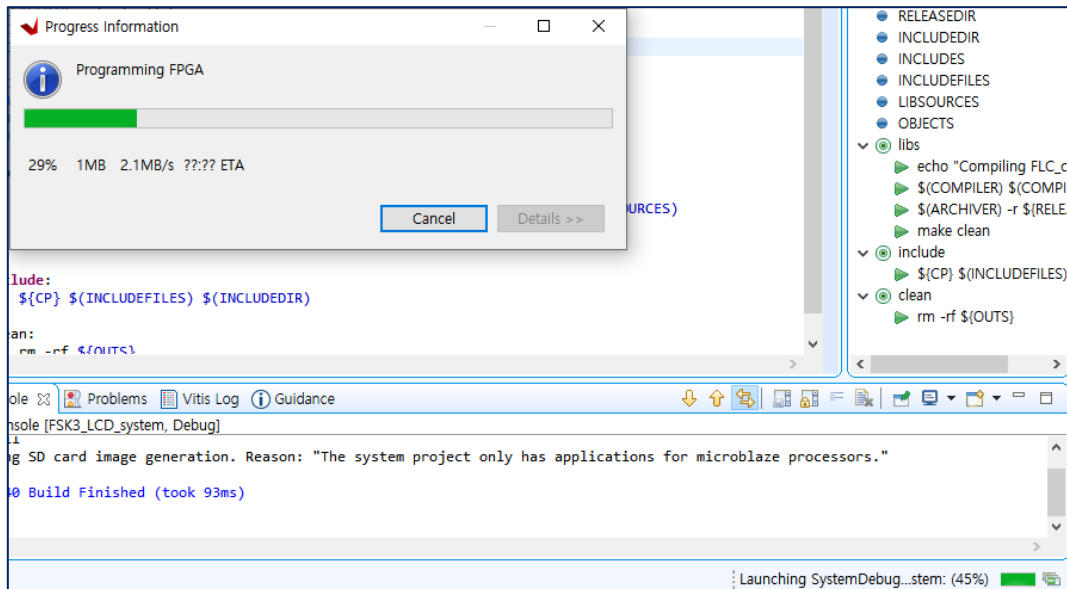
2.4 FPGA에 다운로드 하기

1) FPGA 다운로드

- 지금까지의 작업이 완료되었다면, 하기 그림과 같이 FSK3_LCD_system 을 우클릭 하여 Run As → Launch Hardware 를 클릭하여 FPGA 에 동작 파일을 다운로드 한다.



- 하기와 같이 Vitis 화면에 다운로드가 나타나고, 우측 하단에 전체 동작 진행상황이 표기 된다.



- 카메라 영상이 FMC_LCD_Camera 모듈에 출력 되는 것을 확인할 수 있다.



감사합니다.

Revision History

Ver	Date	Revision
1.0	2021-04-06	Initial Document Release.