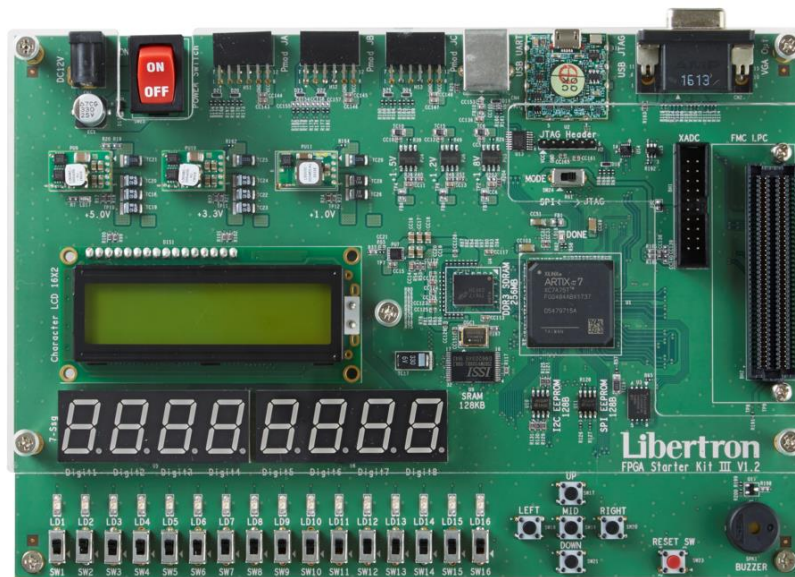


# FPGA Starter Kit III

## UART SDK Tutorial (v1.2B)

Authors : 기술지원팀 김민석 팀장



**Libertron Co., Ltd**

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

## 1. 개요


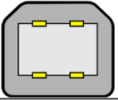





- 본 자료에서는 Xilinx 에서 제공하는 UART 통신의 기본 개념과 Xilinx UART IP 및 Micro Blaze 의 사용방법에 대한 내용을 기술한다.
- 본 디자인은 Vivado Tool 내의 IPI (IP Integrator) 기능을 사용해 FPGA 로직을 구성하고, Tera Term (통신 프로그램) 을 이용하여 UART 동작을 확인 하였다.

## 2. 상세 설명

### 2.1 준비 사항 및 테스트 환경

#### 2.1.1 준비사항

- FPGA Starter Kit III (Ver 1.2B)
- Power Adapter
- USB B Type Connector (FPGA 다운로드 용)
- USB B Type Connector (UART 통신 용)

	Type-A	Type-B	Type-C
Standard			
Mini			
Micro			

#### 2.1.2. 테스트 환경

- Windows 10 / Vivado 2019.1 이하버전

## 2.2 프로젝트 세부 설명

### 2.2.1 디자인 동작 방향

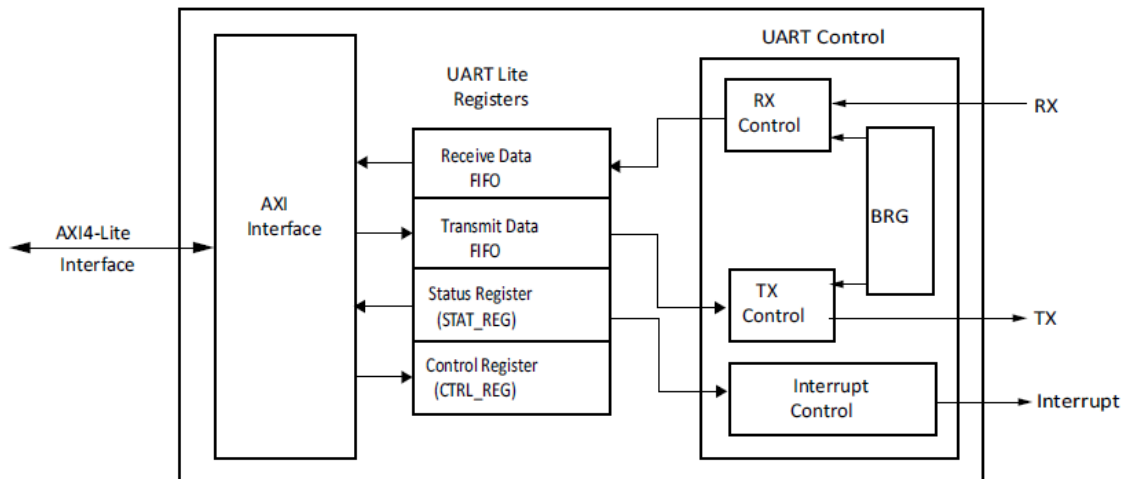
- 본 자료는 Xilinx PG142 Product Guide 의 DataSheet 내용을 기반으로 디자인 하였다. FPGA 에 로직으로 제공되는 UART IP 및 MicroBlaze 라고 하는 S/W 환경을 이용해 UART 를 컨트롤 및 동작 시킨다.
- Vivado Tool 을 이용해 기본 환경을 구성하고, S/W 동작을 위해 **SDK** 또는 **VITIS** 에서 C로 컨트롤 한다.
- **Vivado 2019.1 이하 에서 Vivado 내의 SDK 를 통해 디자인 작업 진행**  
(C 디자인은 Xilinx 에서 제공하는 예제디자인 활용)
- SDK 및 VITIS 에서 제공하는 예제 C 코드에 printf 의 출력메시지를 내보내게 하였으며, PC 에서는 통신 프로그램 (Tera-term) 을 이용하여 메시지 출력 동작을 하도록 한다.

### 2.2.2 UART 및 MicroBlaze 동작 이해

- FSK III 에 장착되어 있는 UART 는 당사에서 제공하는 CD 자료의 회로도 p20 에 나와 있으며, Xilinx PG142 Product Guide 를 참조 하도록 한다.  
(하기 링크에서 다운로드 가능)  
[“https://www.xilinx.com/support/documentation/ip\\_documentation/axi\\_uartlite/v2\\_0/pg142-axi-uartlite.pdf”](https://www.xilinx.com/support/documentation/ip_documentation/axi_uartlite/v2_0/pg142-axi-uartlite.pdf)
- UART (Universal Asynchronous Receiver/Transmitter) 는 개발자들이 간편하게 통신할 수 있는 직렬 방식의 데이터를 전송하는 통신 프로토콜이다.  
기존에는 하기 그림과 같은 포트가 PC 에 나와 있었으나, 현재는 USB to UART 및 Micro 5 Pin 등의 연결 커넥터를 활용한다.

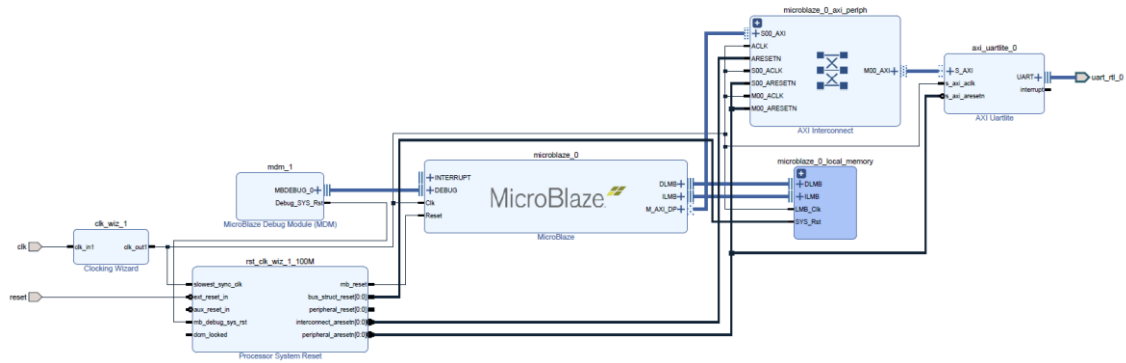


- UART 는 개발자들이 FPGA 보드와 PC를 연결하여 많이 사용하고 있지만, UART 통신을 위해 User 가 로직을 디자인하기에 어려움이 있다.  
이로 인해, Xilinx 에서는 User 를 위해 UART 통신에 관련된 FPGA 로직을 IP Catalog 에서 IP (Intellectual Property) 형태로 무료 제공 한다.
- UART 와 같은 Xilinx IP를 사용하기 위해서 MicroBlaze 라고 하는 Softcore 를 사용해야 하는데, FPGA 로직 환경보다 S/W 환경에서 컨트롤 하는 것이 편하고 쉽기 때문이다. C를 이용한 인터페이스 컨트롤은 사용자의 개발 편의성을 높여, 보다 쉽게 UART를 제어 할 수 있다.
- MicroBlaze 는 SDK (Vivado 2019.1 이하) 및 Vitis (Vivado 2019.2 이상) 에서 C를 이용하여 디자인 할 수 있도록 제공 한다.  
본 자료는 SDK 를 활용하였다. (Vitis 는 UART Tutorial - Vitis 참고)



- 상기 그림은 Xilinx PG142 Product Guide 의 p5에 있는 IP 블록도 이다.  
왼쪽의 AXI4-Lite 로 연결되는 부분은 내부 로직 (여기에서는 MicroBlaze) 과 연결하여 컨트롤 할 수 있도록 제어를 하고, 오른쪽의 RX/TX Port 로 외부 즉, UART Port 로 연결하여 PC 와 통신할 수 있도록 한다.
- AXI Interface 사용에 대해 User 의 어려움이 있을 수 있기에, Xilinx 에서는 Vivado IPI (IP Integrator) 기능을 제공한다.  
수많은 AXI Interface 를 연결해야 하는 포트를 스케메틱 방식으로 다른 IP의 AXI 포트에 연결 하거나, IPI (IP Integrator) 에서 자동 연결하기를 선택하면 그에 맞는 블록을 구성하여 연결 시키므로 어려움 없이 Uart를 다른 로직과 연결하여 사용할 수 있다.

### 2.2.3 디자인 동작 블록도



- 상기 그림과 같이 Uart 를 동작 시키기 위한 블록을 구성하였다.

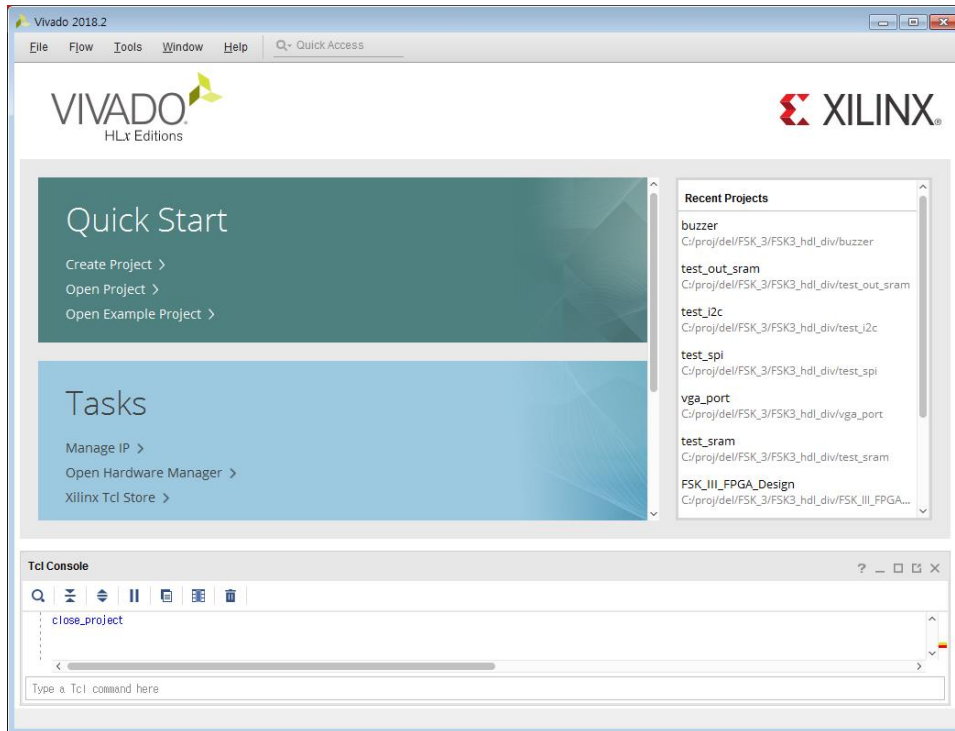
IPI (IP Integrator) 에서 Uart IP 와 MicroBlaze 블록을 가져다 놓고 자동 연결을 실행 하면 상기와 같이 연결되며, 세부 셋팅 사항은 하기 실습 진행 내용에서 언급하도록 한다.

## 2.2.4 Vivado 디자인 구성

### 2.2.4-1 Vivado New Project

#### 1) Vivado New Project 실행

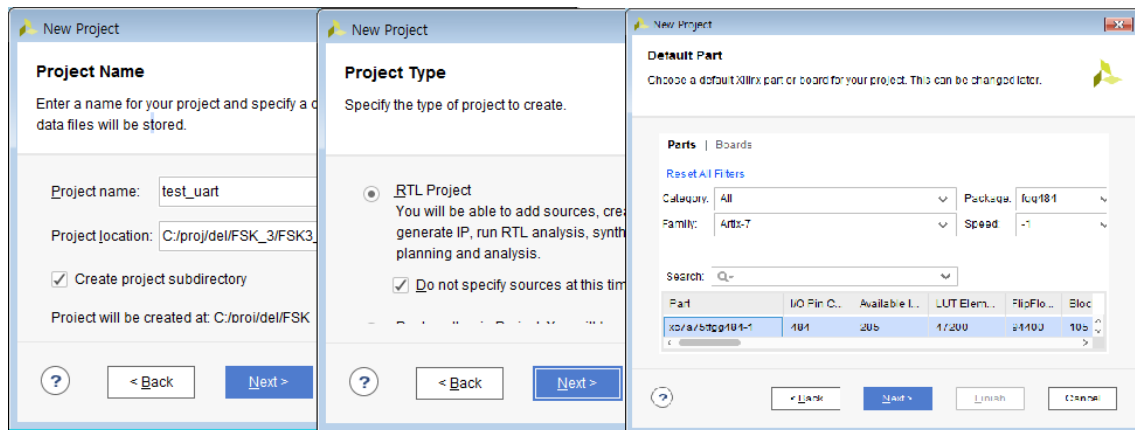
\* Project 작업 경로 및 폴더에 특수문자 및 한글의 인식이 안되므로, 영어만 사용할 것



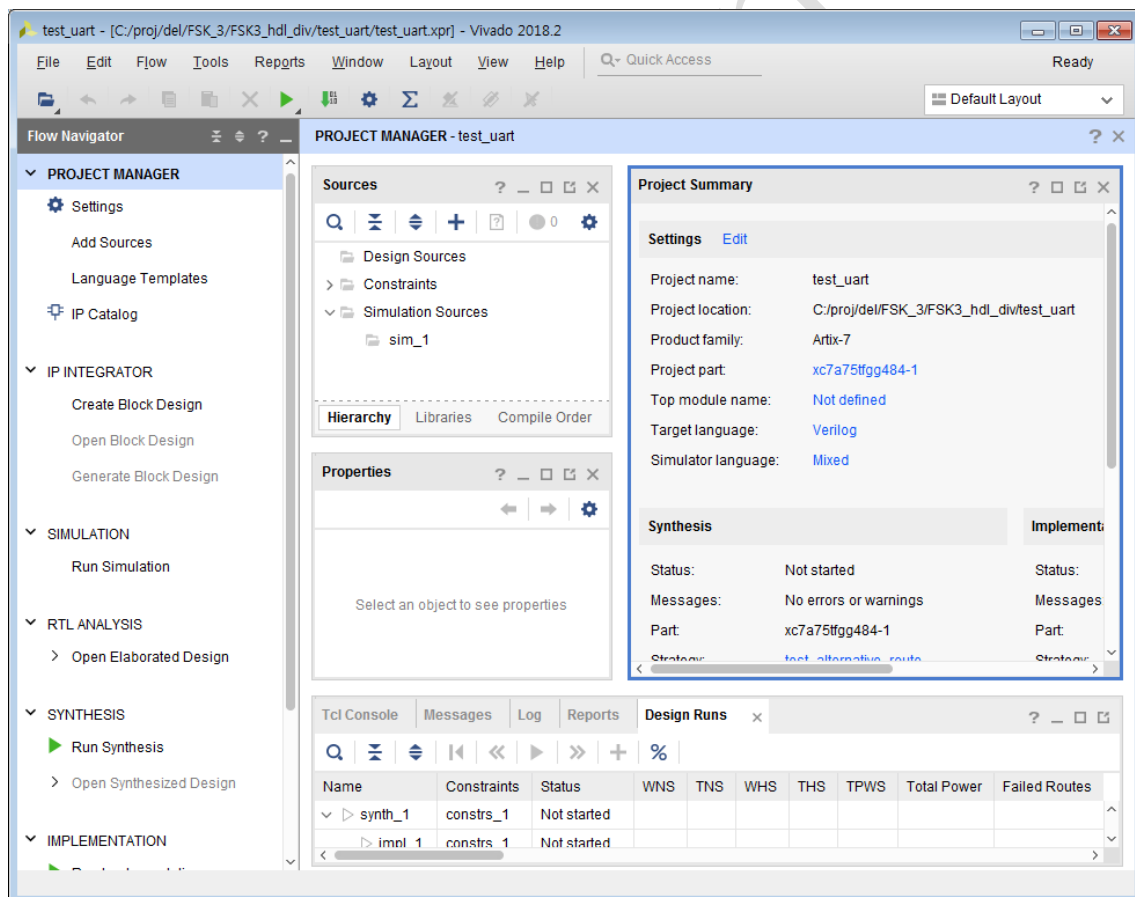
#### 2) 상기 그림에서 다음과 같이 진행 한다.

- Create Project → Create a New Vivado Project 에서 Next 클릭
- Project Name 란에 "test\_uart" 입력 후 Next 클릭
- RTL Project 선택 후 Next 클릭
- Default Part 란에서 하기와 같이 선택 (하기 그림 참조)

<b>Family</b>	Artix-7
<b>Package</b>	Fgg484
<b>Speed</b>	-1
<b>Full Part Name</b>	XC7A75TFGG484-1



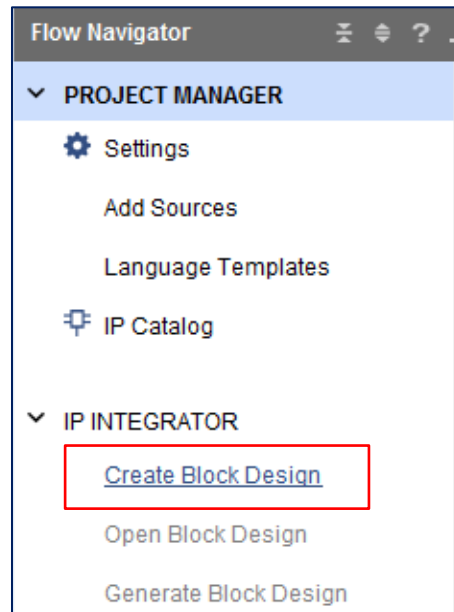
- 3) 상기 작업 후 New Project Summary 창이 나타나면 Finish 클릭  
작업이 완료 되면 하기와 같이 Vivado 초기 프로젝트 화면이 나타난다.



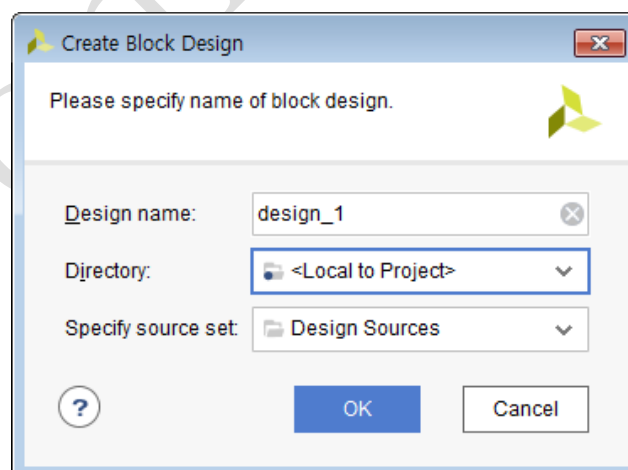


## 2.2.4-2 Vivado New Design Create

- 1) 하기의 그림과 같이 Project Manager 창에 있는 IP Integrator → Create Block Design 을 클릭한다.



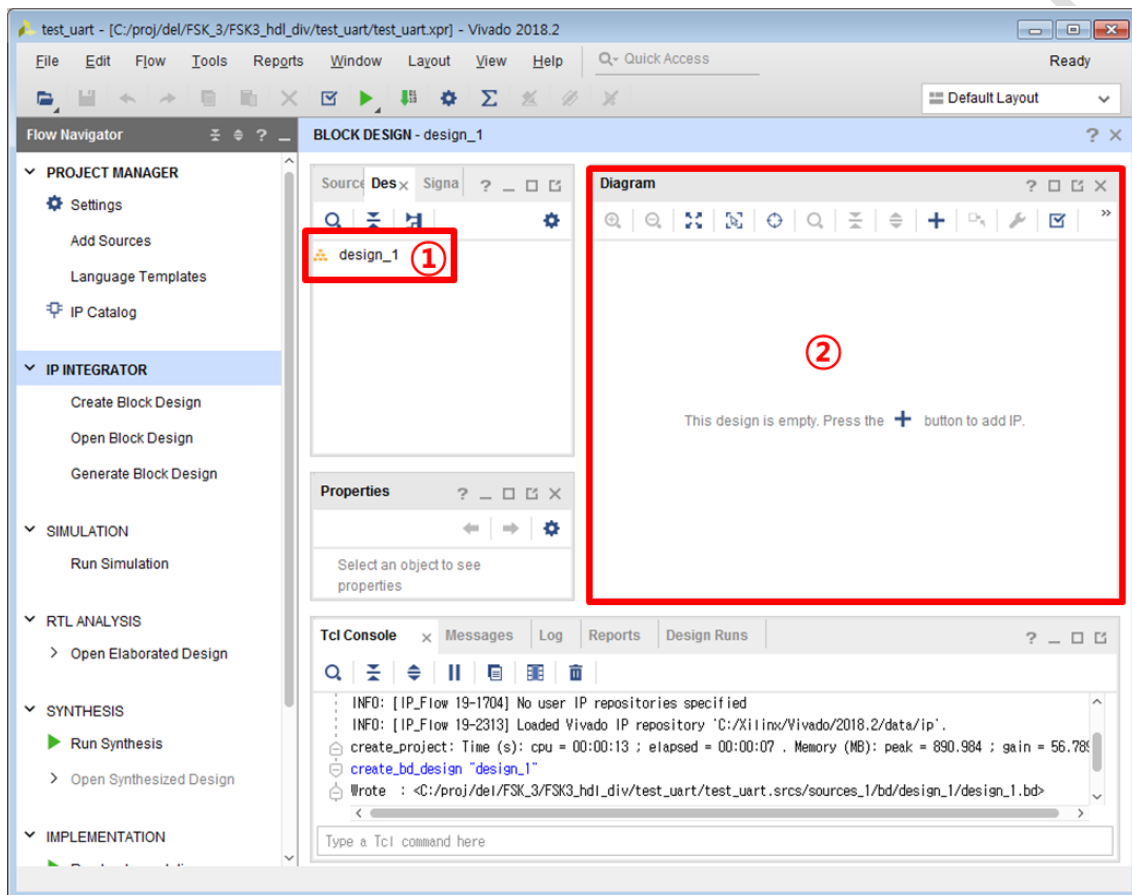
- 2) 하기 창이 뜨면 default 상태에서 OK 를 선택한다.



3) 작업이 완료되면 하기의 그림과 같이 Vivado 창이 나타난다.

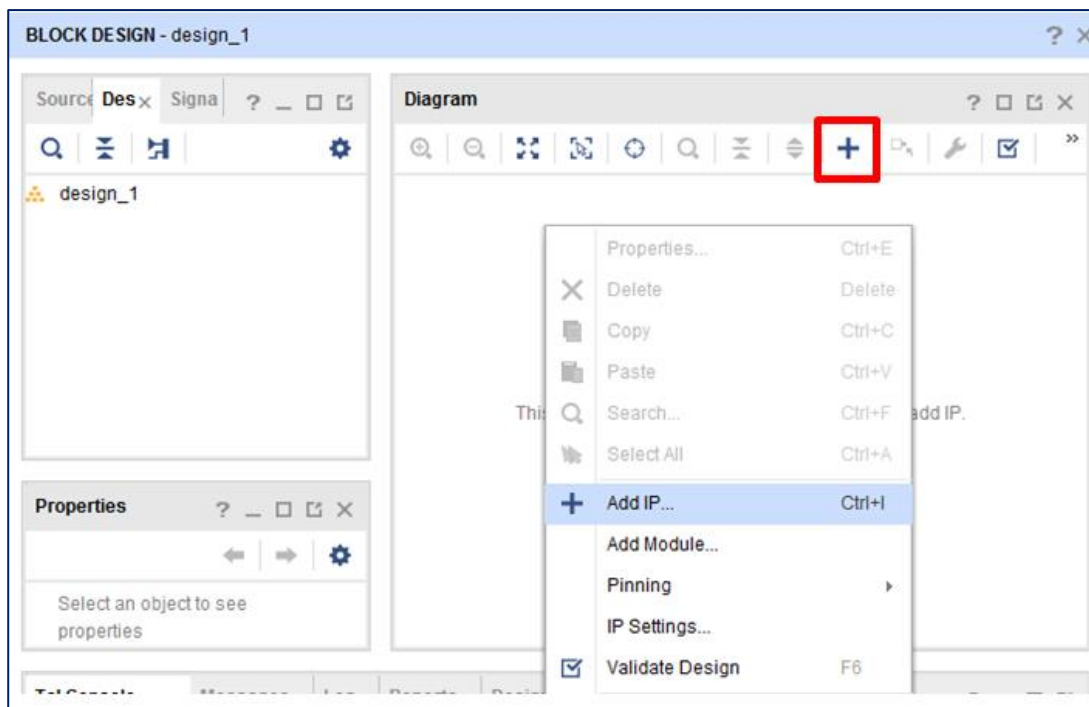
하기 ① 은 이전 작업을 통해 생성한 design\_1.bd 형태의 디자인 파일이며,  
② 는 ① 의 파일을 오픈한 창으로 사용자가 스케메틱과 같은 방식으로 디자인을 할 수 있는 환경이다.

② 에서는 위에서 언급한 MicroBlaze 블록과 Uart 블록을 사용해 디자인을 구성 한다.

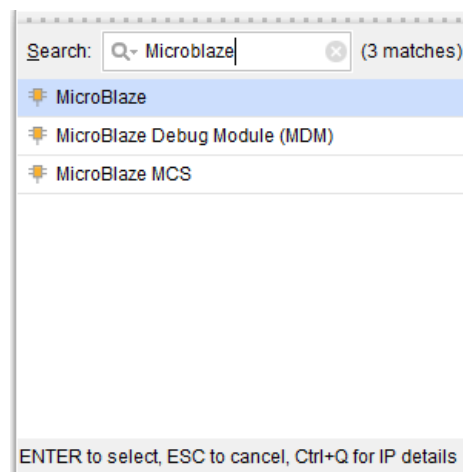
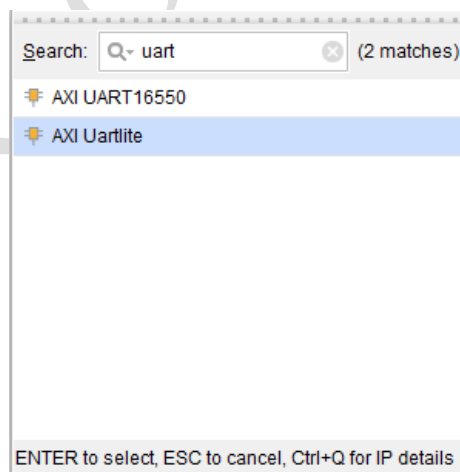


### 2.2.4-3 IPI (IP Integrator) 디자인 진행 (design\_1.bd)

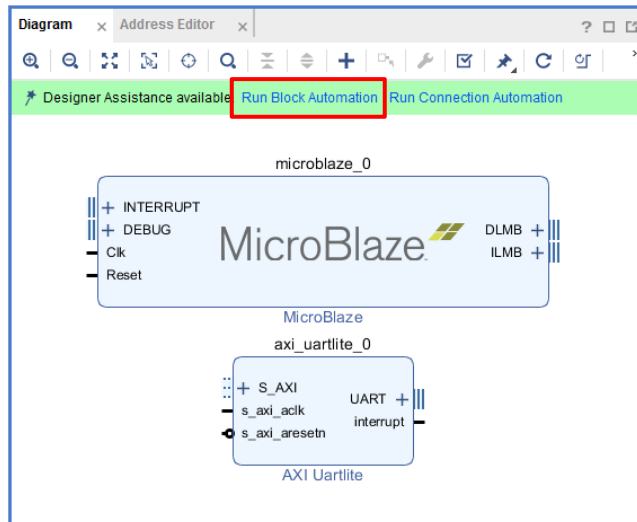
- 1) 하기의 그림과 같이 IPI (IP Integrator) 의 Diagram 창에서 우클릭 or 상단의 메뉴에서 **+** 버튼을 눌러 Add IP 를 선택 한다.



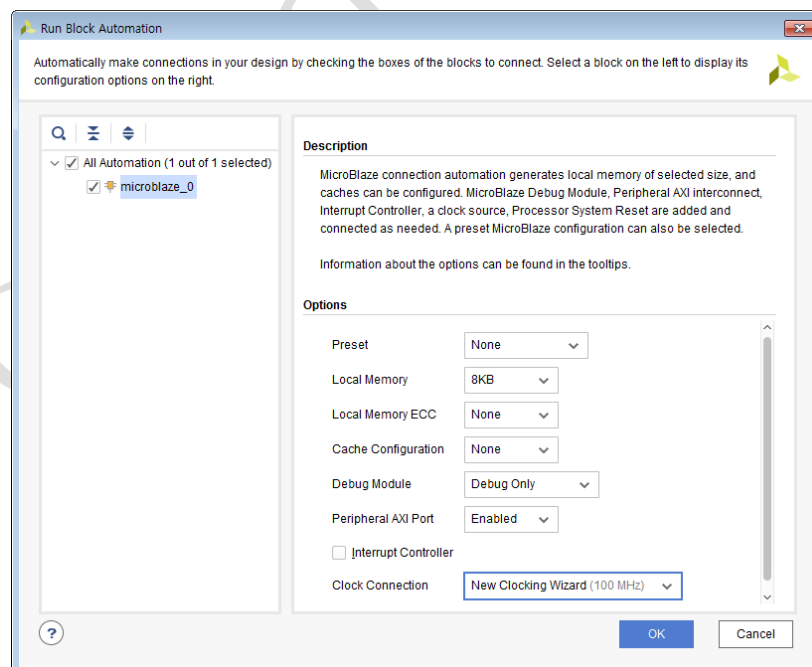
- 2) 상기 작업으로 하기의 창이 나타나면 "UART" 입력 → AXI Uartlite 선택  
→ 추가로 "MicroBlaze" 입력 → MicroBlaze IP 선택



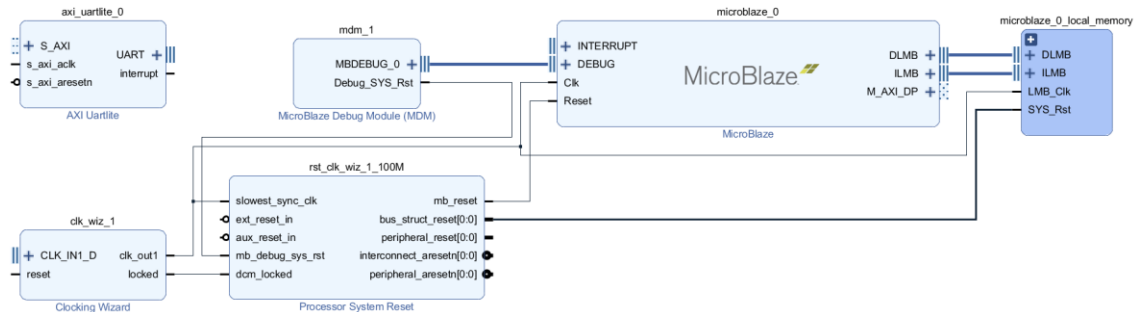
- 3) IPI 의 Diagram 창에 하기의 그림과 같이 MicroBlaze 의 블록이 생성된다.



Run Block Automation 을 클릭하면 Micro Blaze\_0 블록의 셋팅 및 연결을 설정할 수 있는 창이 나타난다. 하기와 같이 설정 후 OK 선택



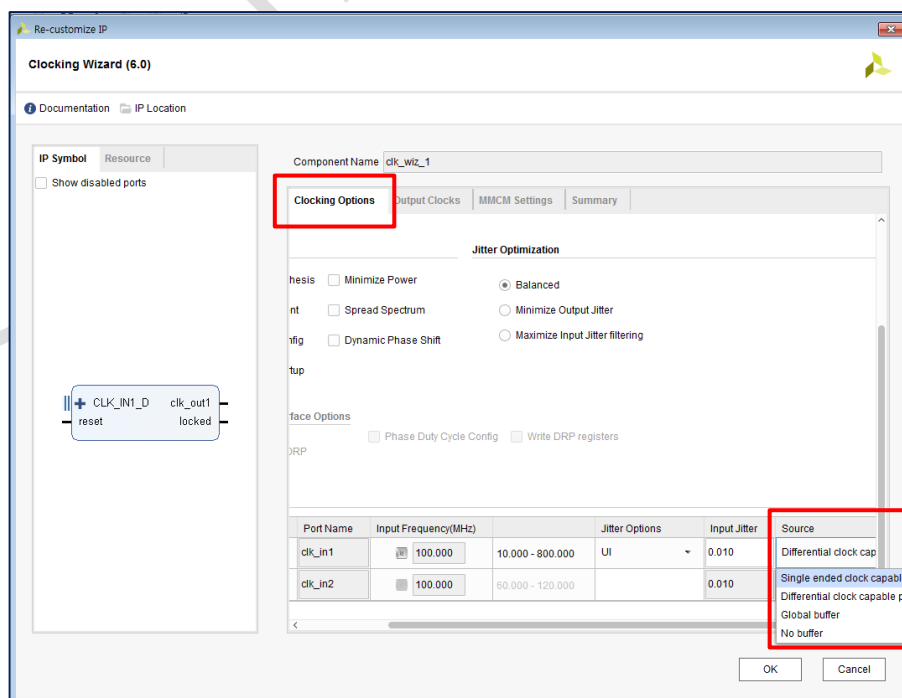
- 4) 작업이 완료되면 하기의 그림과 같이 나타난다.



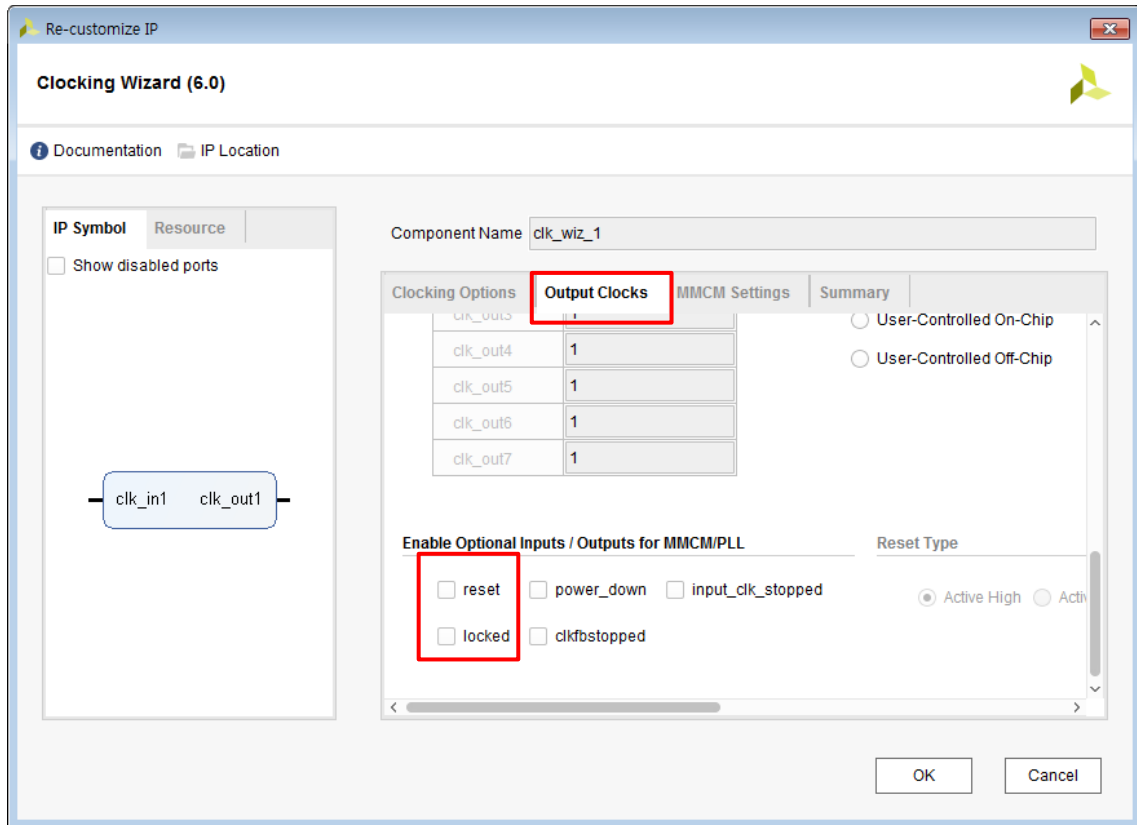
- 그림에서 clk\_wiz\_1 블록에서의 CLK\_IN1\_D 로 들어오는 Clock 은 우리가 사용하는 FSK III 의 Clock 과 맞지 않다.  
FSK III 에서 들어오는 100MHz Clock은 Single ended 방식으로 들어오는데, clk\_wiz\_1 블록은 differential Clock 이 들어오는 것으로 되어 있다.

또한, reset, Locked 포트를 사용하지 않기에 disable 작업을 해준다.  
이 작업을 위해 clk\_wiz\_1 블록을 더블클릭 한다.

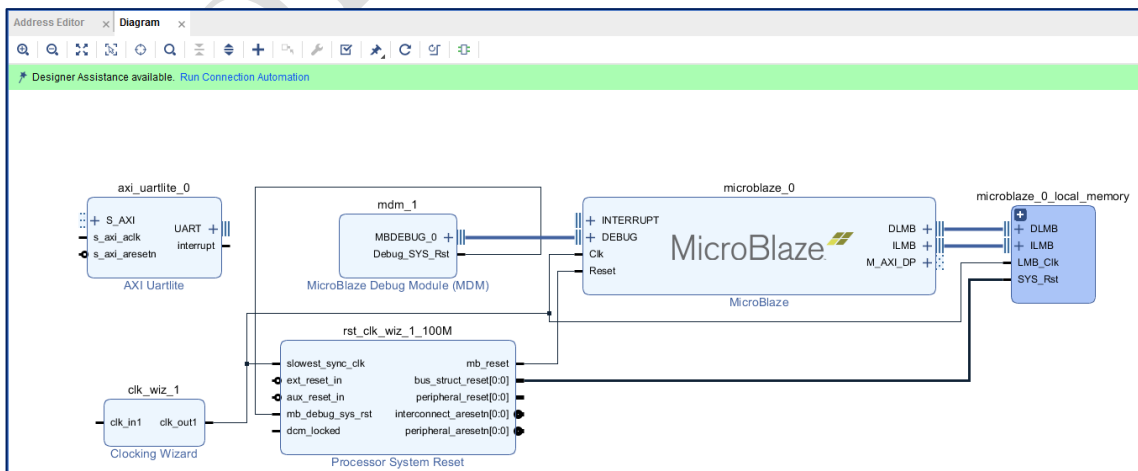
- 5) 하기의 창이 나타나면 clk\_in1 에 있는 Source 의 Differential Clock capable pin 을 클릭하여 Single ended clock capable pin 을 변경한다.



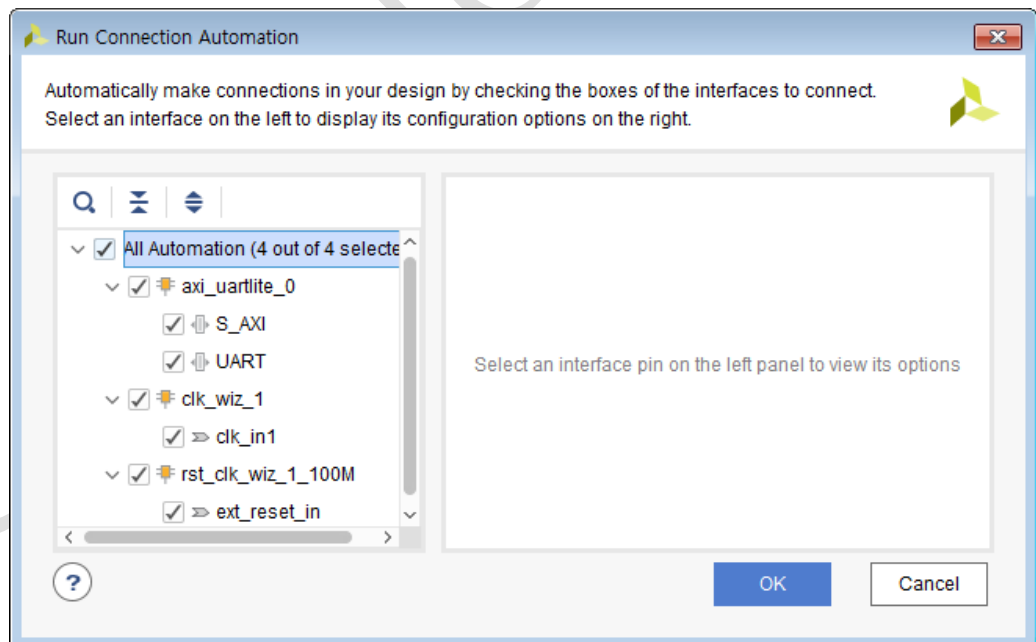
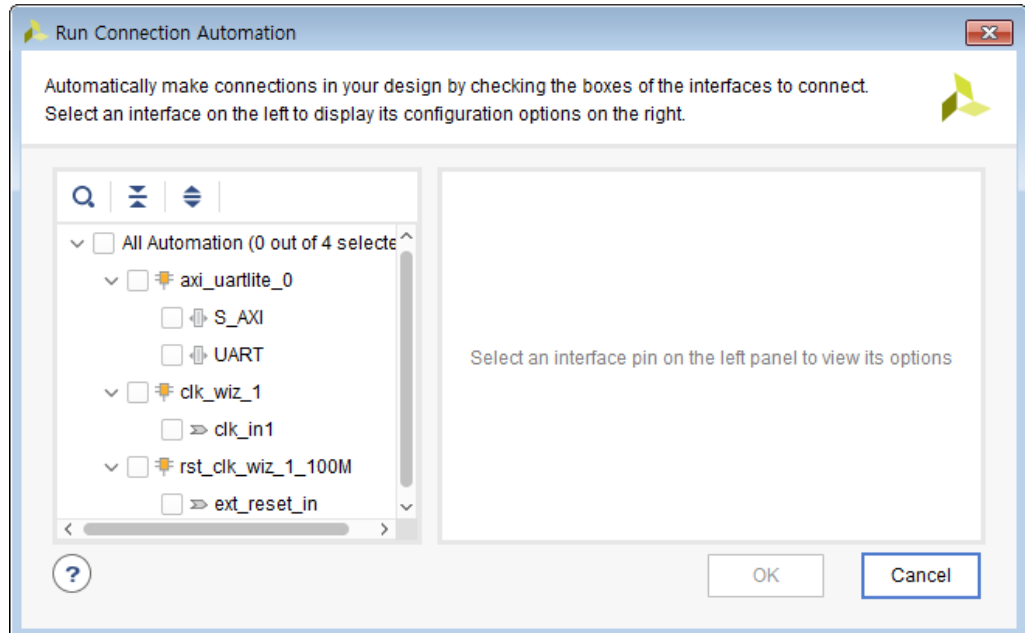
- 6) 상기 작업 후, Output Clocks 탭을 선택하여 하단의 reset, locked 옵션을 disable 시키고 OK 를 클릭한다.



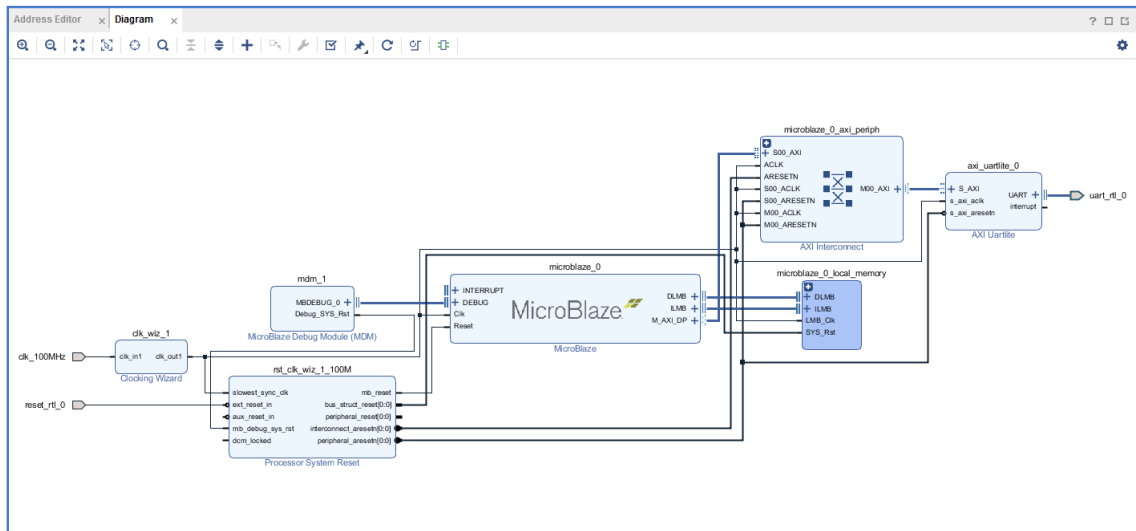
작업이 완료되면 하기의 그림과 같이 clk\_wiz\_1 블록이 셋팅된 것을 확인할 수 있다.



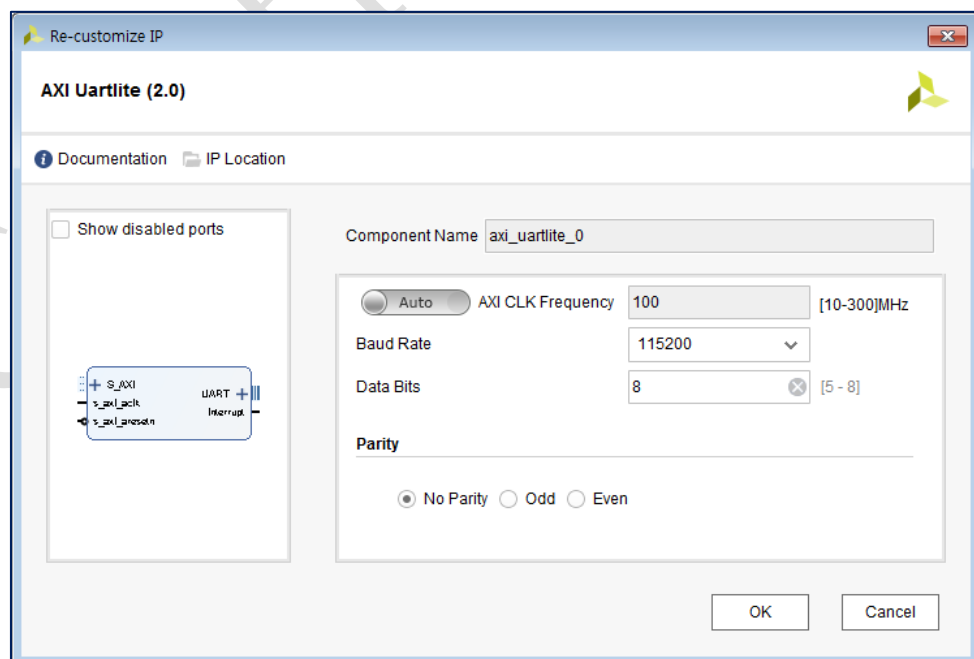
- 7) 다음은 상기 Diagram 창의 **Run Connection Automation** (Run Connection Automation) 을 선택 한 후, 모든 항목에 체크 한 뒤 OK 선택



- 8) 상기 작업이 완료되면 IPI 의 Diagram 창에 하기 그림과 같은 디자인이 구성된다.



- 디자인의 axi\_uartlite\_0 을 더블 클릭하면 하기 그림이 나타나며, UART 의 인터페이스 속도를 셋팅 한다.  
Baud Rate 를 115200 으로 셋팅 후 OK 를 선택.

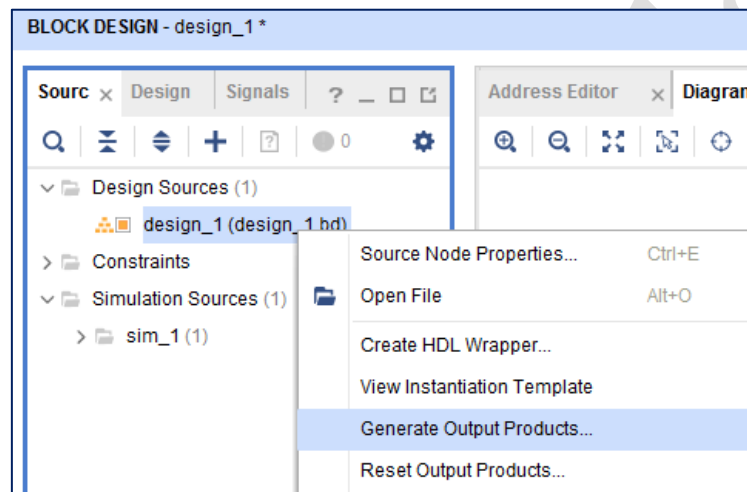




#### 2.2.4-4 IPI 디자인 Generate Output Product 진행하기

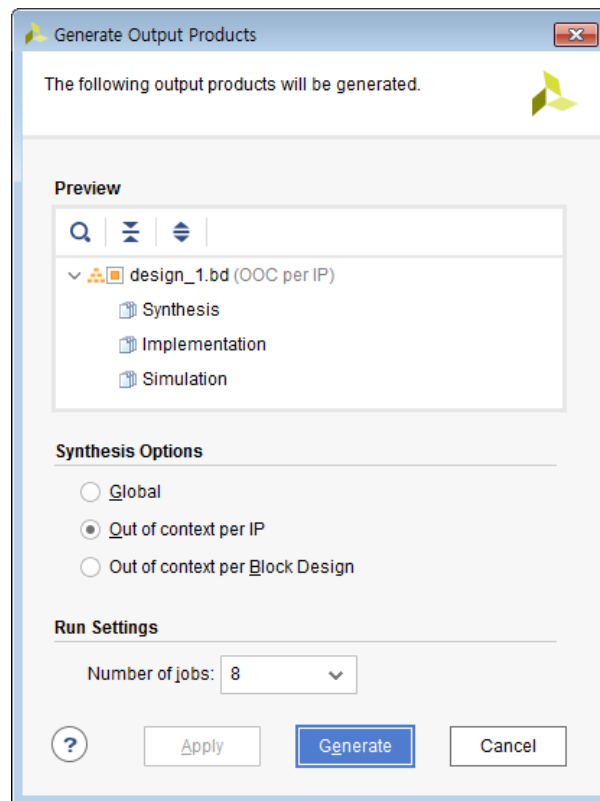
- 1) 하기 그림과 같이 Design\_1.bd 를 우클릭해 Generate Output Product 를 선택한다. 이 작업은 하드웨어 정보를 만드는 작업으로, 차후 SDK 및 VITIS 를 사용할 때 꼭 필요하다.

하기 두 번째 그림과 같이 MicroBlaze 에 연결된 블록들의 Address 가 지정되어 있는데, 디자인 수정이 이루어질 때 마다 이 작업을 해야 MicroBlaze 에 연결되어 있는 디바이스들의 Address 를 업데이트 하여 SDK 및 VITIS 에서 디자인 작업을 할 수 있도록 지원한다.



Cell	Slave Interface	Base Name	Offset Address	Range	High Address
microblaze_0					
Data (32 address bits : 4G)					
microblaze_0_local_memory/dlmb_bram_if_cntlr	SLMB	Mem	0x0000_0000	8K	0x0000_1FFF
axi_uartlite_0	S_AXI	Reg	0x4060_0000	64K	0x4060_FFFF
Instruction (32 address bits : 4G)					
microblaze_0_local_memory/ilmb_bram_if_cntlr	SLMB	Mem	0x0000_0000	8K	0x0000_1FFF

- 2) Generate Output Product 진행 시 하기와 같은 창이 나타난다.  
하기 그림에서 Generate 를 클릭하여 작업을 진행한다.



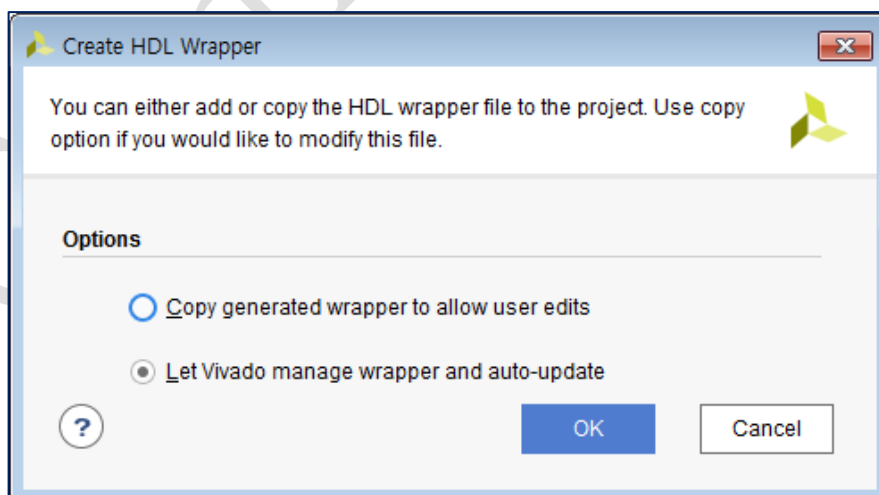
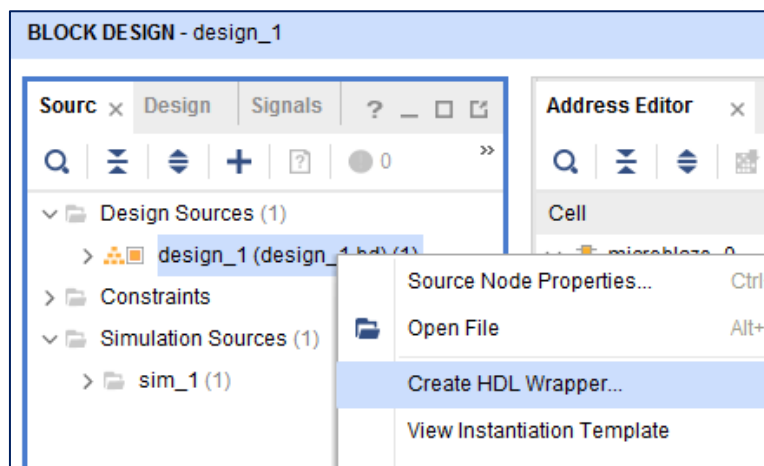
Global	디자인 전체를 하나의 Netlist 로 생성
Out of context per IP	IP 마다 컴파일 된 정보를 생성하여 저장
Out of context per Block Design	Block Design 마다 컴파일 정보를 생성하여 저장
Number of jobs	컴파일 할 때 사용할 CPU 개수를 지정

#### 2.2.4-4 Create HDL Wrapper 작업

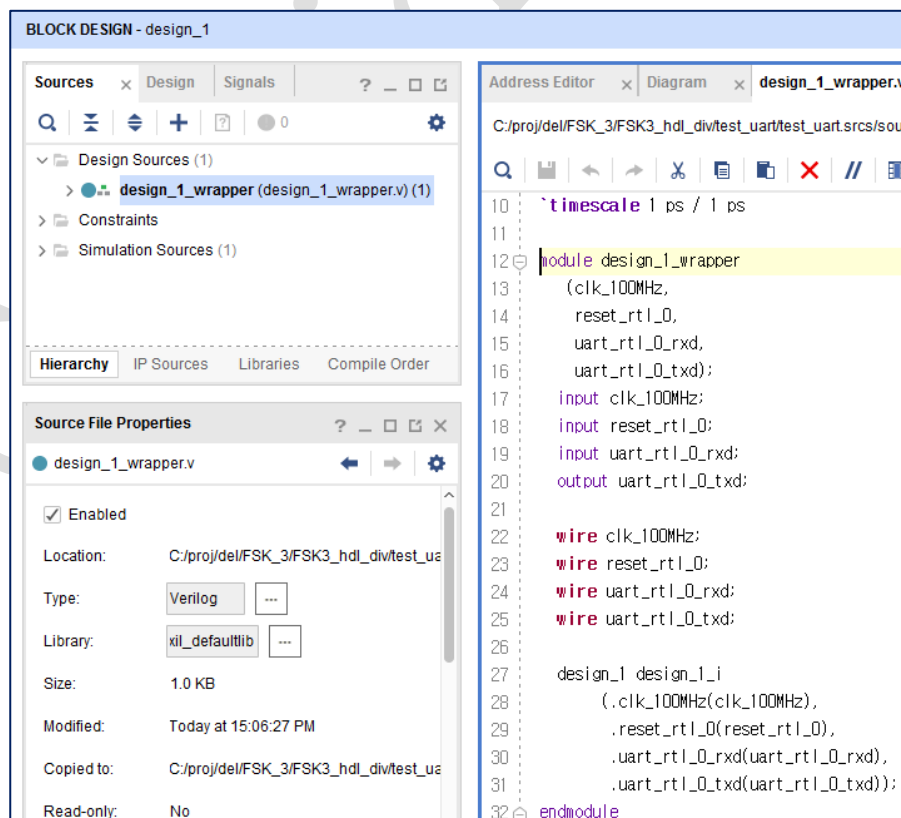
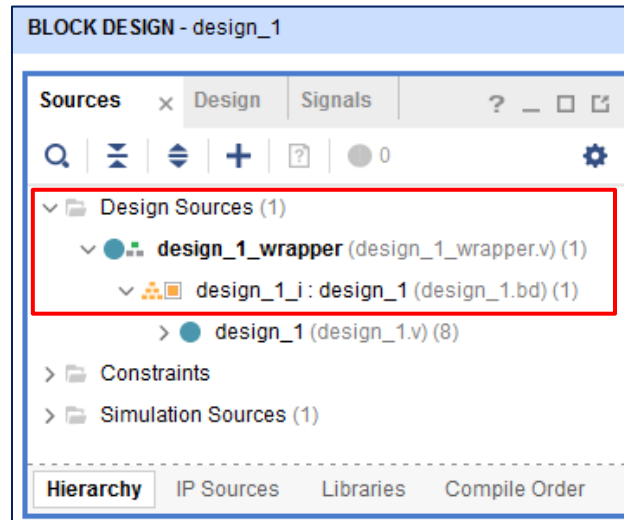
- 1) 이전까지의 작업이 완료되면, 기본 디자인 작업이 완료된 상황이다.  
Vivado 에서는 Top Design 을 HDL로 받아들이게 되어 있으므로,  
Design\_1.bd 파일 위에 Wrapper 를 만들어 줘야 한다.

Vivado 에서는 HDL Wrapper 를 디자인에 맞게 자동으로 구성해주므로,  
하기의 작업을 진행 한다.

- 2) Block Design → Source → Design\_1 우클릭 → Create HDL Wrapper 선택 →  
하기 두번째 창이 나타나면 OK 클릭

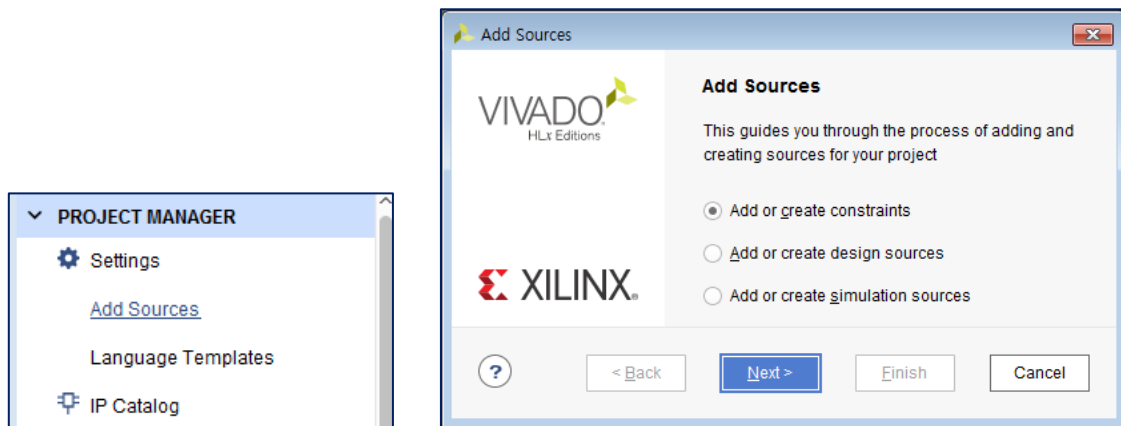


- 3) 상기 작업이 완료되면 Source 창이 하기의 그림과 같이 나타나게 된다.  
우리가 작업한 Block Design (아이콘) 에 Design\_1\_wrapper 라고 하는 HDL Wrapper 가 생성된 것을 확인 할 수 있다.  
HDL 파일을 열어보면 하기 두 번째 그림과 같다.

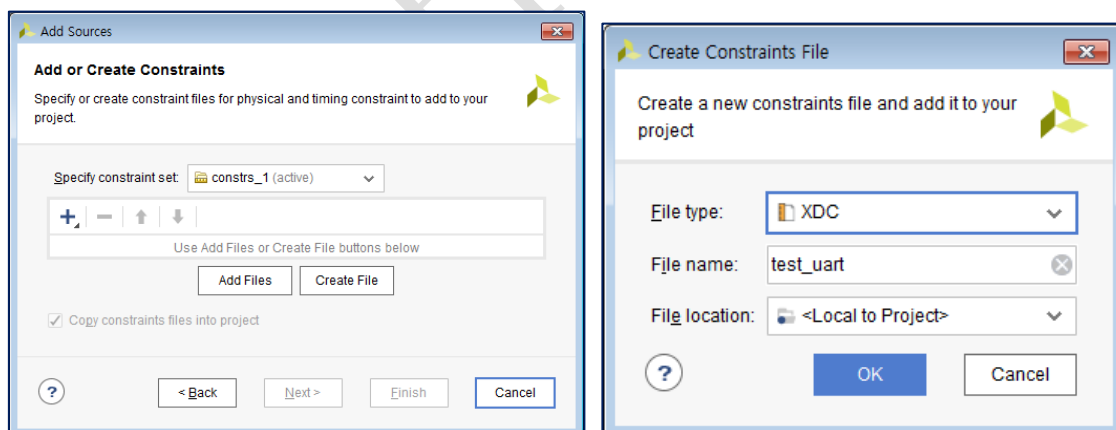


## 2.2.4-5 Vivado 디자인 Pin 정보 (XDC) 입력 작업

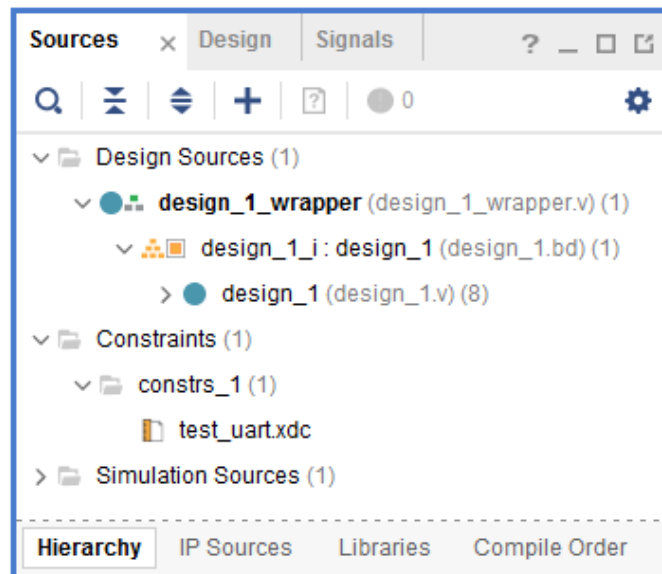
- 1) 하기의 그림과 같이 PROJECT MANAGER 창에서 Add Sources 클릭  
→ Add or create constraints 선택 후 Next 클릭



- 2) Add or Create Constraints 창에서 Create File 클릭  
→ File name 란에 "test\_uart" 라고 입력한 후 OK 클릭  
→ 그리고 다시 Add or Create Constraints 창이 나타나면 Finish 클릭



- 3) 하기의 그림과 같이 Sources 창에서 test\_uart.xdc 가 나타난 것을 확인 할 수 있으며, test\_uart.xdc 파일을 더블 클릭하면 Vivado 우측창에 XDC 를 편집할 수 있는 에디터 창이 나타난다.  
이곳에 하기의 핀 정보를 입력 한다.



```
set_property -dict {IOSTANDARD LVCMOS33 PACKAGE_PIN R4} [get_ports clk_100MHz]

set_property -dict {IOSTANDARD LVCMOS33 PACKAGE_PIN U7} [get_ports reset_rtl_0]

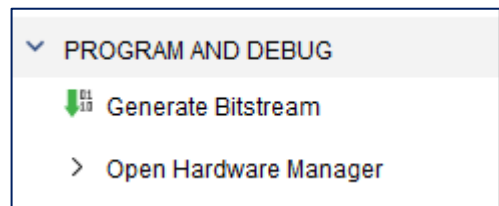
set_property -dict {IOSTANDARD LVCMOS33 PACKAGE_PIN N13} [get_ports uart_rtl_0_rxd]

set_property -dict {IOSTANDARD LVCMOS33 PACKAGE_PIN N14} [get_ports uart_rtl_0_txd]
```

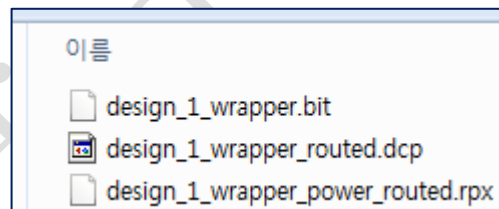
#### 2.2.4-6 Vivado 디자인 다운로드 파일(bit) 생성

- 1) 지금까지의 작업으로 디자인 소스와 핀 정보(XDC) 파일 생성이 완료되었으며, FSK III 에 Design Download 를 위해 bit 파일을 생성한다.

하기 그림과 같이 PROJECT MANAGER 창에 있는 Generate Bitstream 클릭 후 다른 창이 나타나면 OK를 눌러 진행한다.

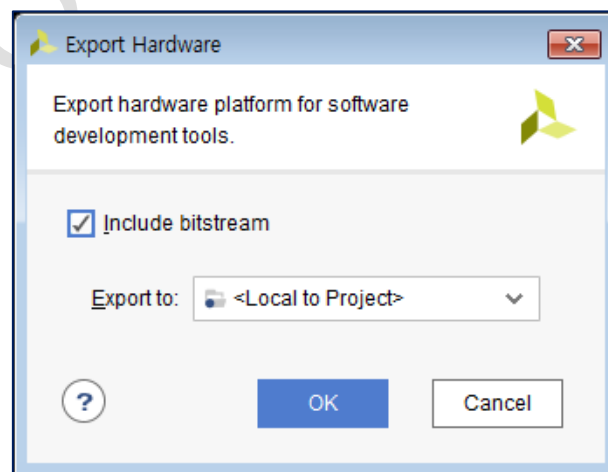
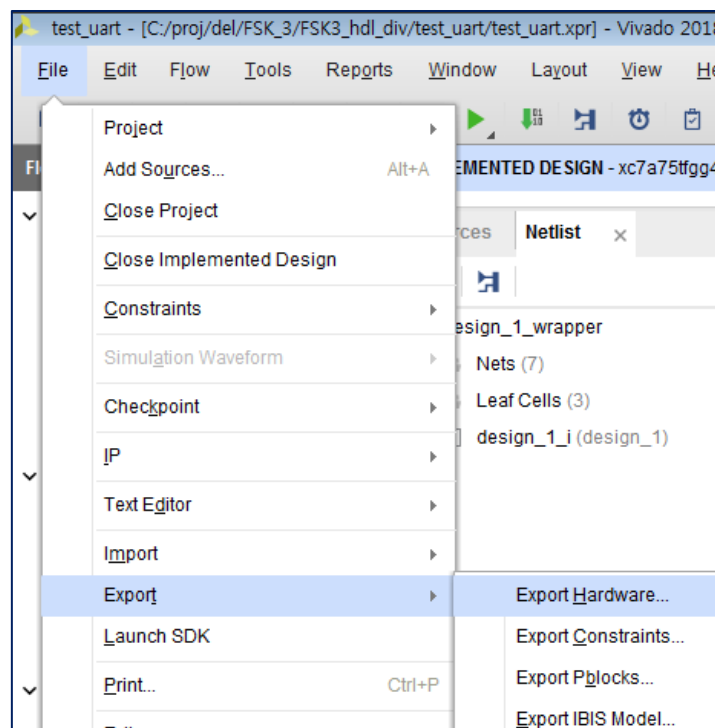


- 2) 작업이 완료 되면 프로젝트 폴더에서 Bit 파일이 생성됨을 확인할 수 있다.  
(프로젝트 폴더 → test\_uart.runs 폴더 → impl\_1 폴더 → design\_1\_wrapper.bit)



### 2.2.4-7 Vivado Hardware 정보 파일 생성

- 1) 현재까지 작업으로 하드웨어 다운로드 파일이 생성 되었으며,  
SW 적인 작업을 진행 하기 위해 SDK 에서의 디자인 작업을 진행 한다.  
SDK 는 하드웨어 정보를 기반으로 디자인을 진행 하므로, Vivado 에서  
하드웨어 정보를 입력 한다.
- 2) Viavdo 메인 화면 → File → Export → Export Hardware 선택  
→ 하기 두 번째 창이 나타나면 Include bitstream 을 체크한 후 OK 선택





- 3) 작업이 완료 되면 하기 그림처럼 test\_uart.sdk 폴더가 생성되고 design\_1\_wrapper.hdf 파일이 생성 된다.

프로세서는 외부 인터페이스를 모두 메모리로 규정하기에, 메모리 맵을 구성한 뒤 디자인이 구성되는데, 이 hdf 파일이 하드웨어를 구성한 메모리맵 정보를 가지고 있다. 때문에 이 파일을 가지고 SDK 가 디자인을 할 수 있다.

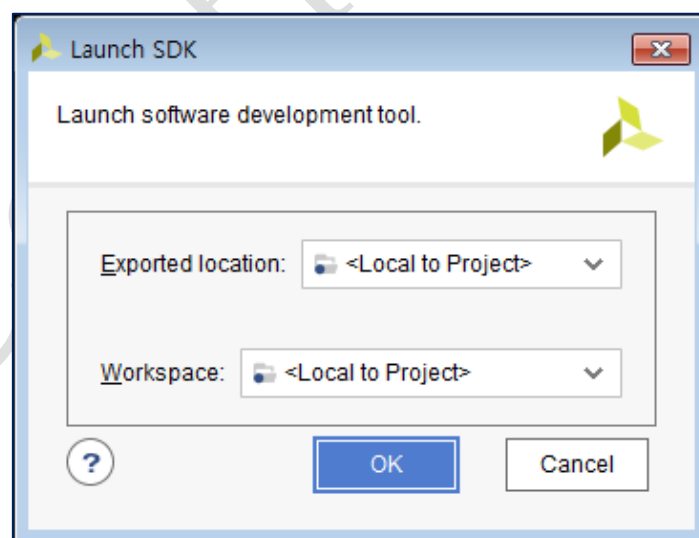
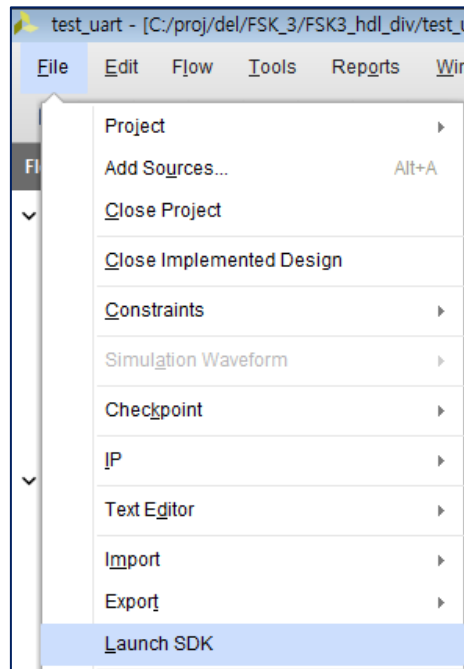
이름	수정한 날짜	유형
test_uart.cache	2020-05-07 오후...	파일 폴더
test_uart.hw	2020-05-07 오전...	파일 폴더
test_uart.ip_user_files	2020-05-07 오후...	파일 폴더
test_uart.runs	2020-05-07 오후...	파일 폴더
test_uart.sdk	2020-05-07 오후...	파일 폴더
test_uart.sim	2020-05-07 오전...	파일 폴더
test_uart.srds	2020-05-07 오후...	파일 폴더
test_uart.xpr	2020-05-07 오후...	Vivado Project File

이름	수정한 날짜	유형
design_1_wrapper.hdf	2020-05-07 오후...	HDF 파일

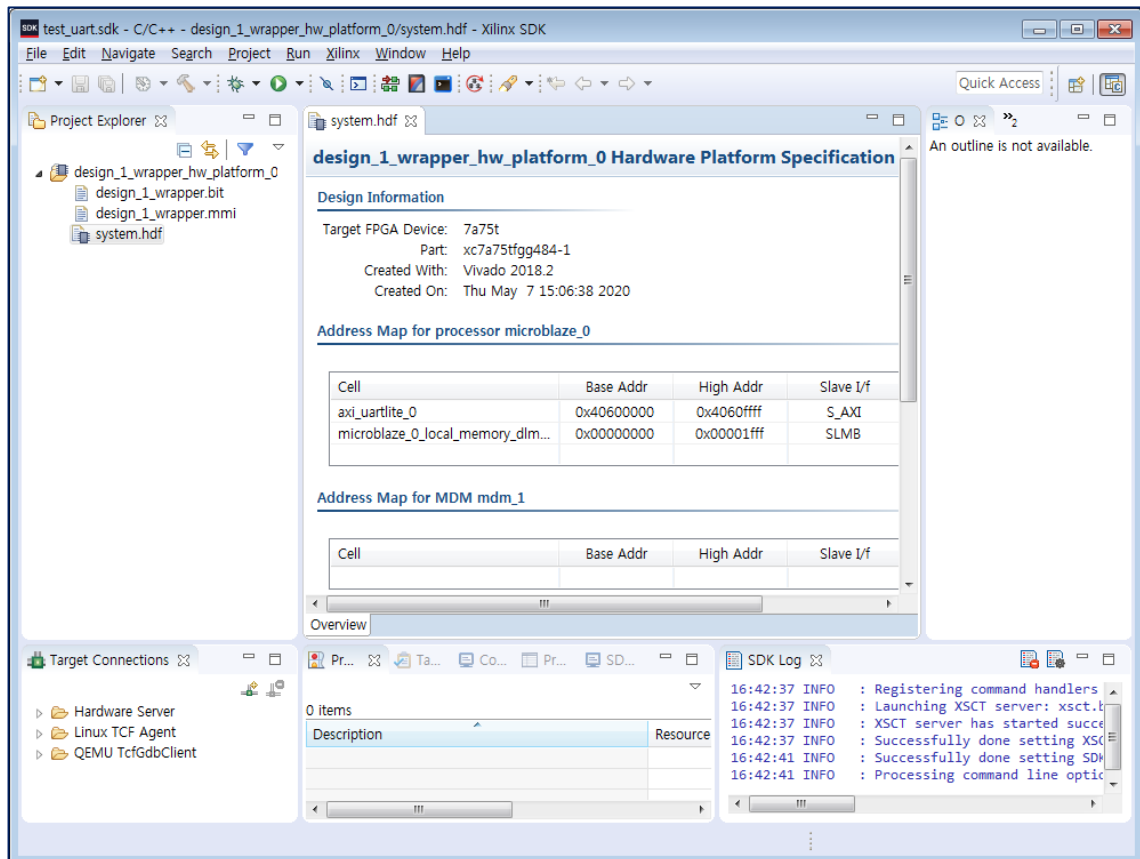
## 2.2.5 SDK 디자인 구성

### 2.2.5-1 SDK Project 실행

- 1) SDK 를 실행하기 위해 하기의 그림과 같이 Vivado 창에서 File → Launch SDK 를 선택하고 , 하기 두 번째 창이 나타나면 default 상태로 OK 클릭

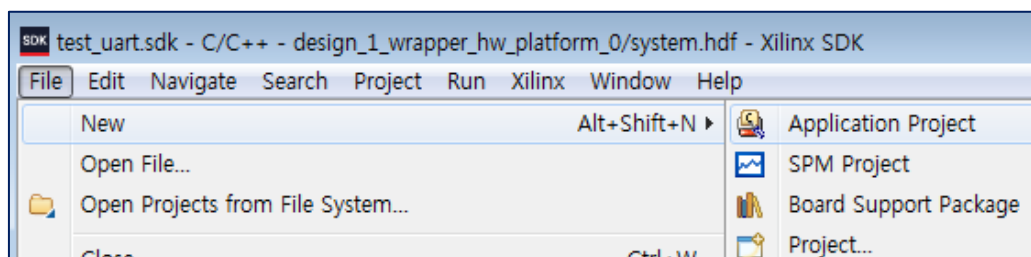


- 2) SDK 가 실행되면 하기 그림과 같이 SDK 창에 system.hdf 파일이 자동으로 열리며, axi\_uartlite\_0 과 microblaze\_0\_local\_memory\_dlmbram\_if\_cntl 에 대한 Address 가 나타나는 것을 확인할 수 있다.

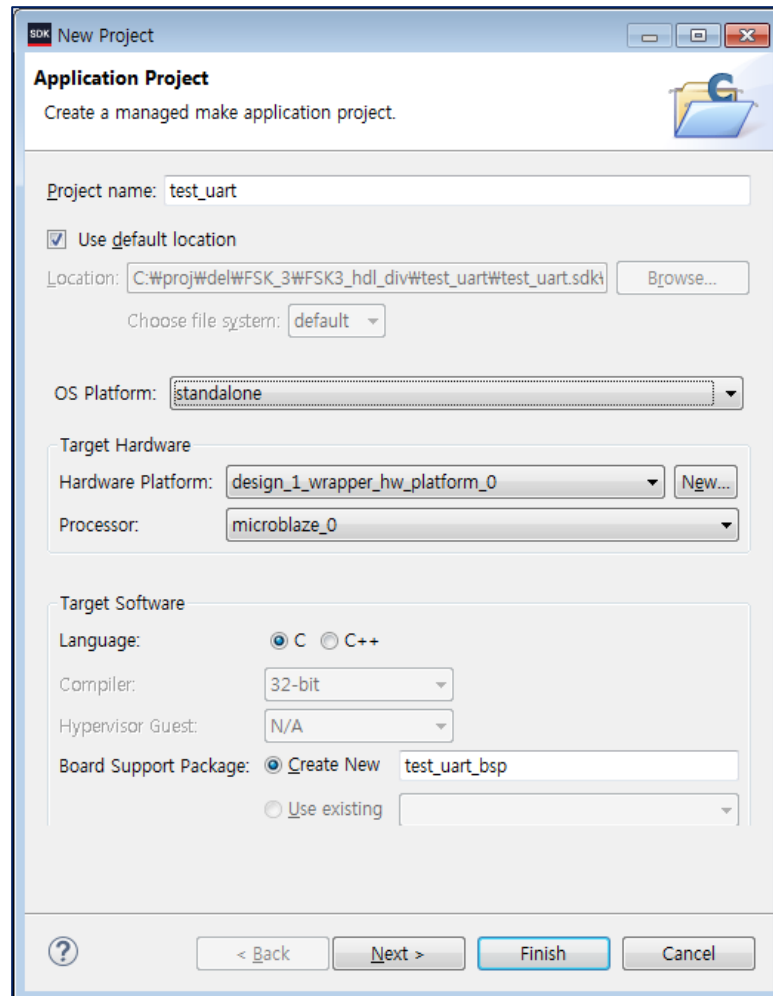


## 2.2.5-2 SDK Application Project 진행하기

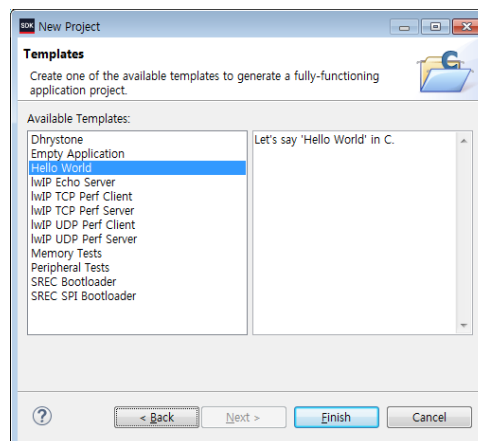
- 1) 하기 그림과 같이 SDK 창에서 File → New → Application Project 를 클릭



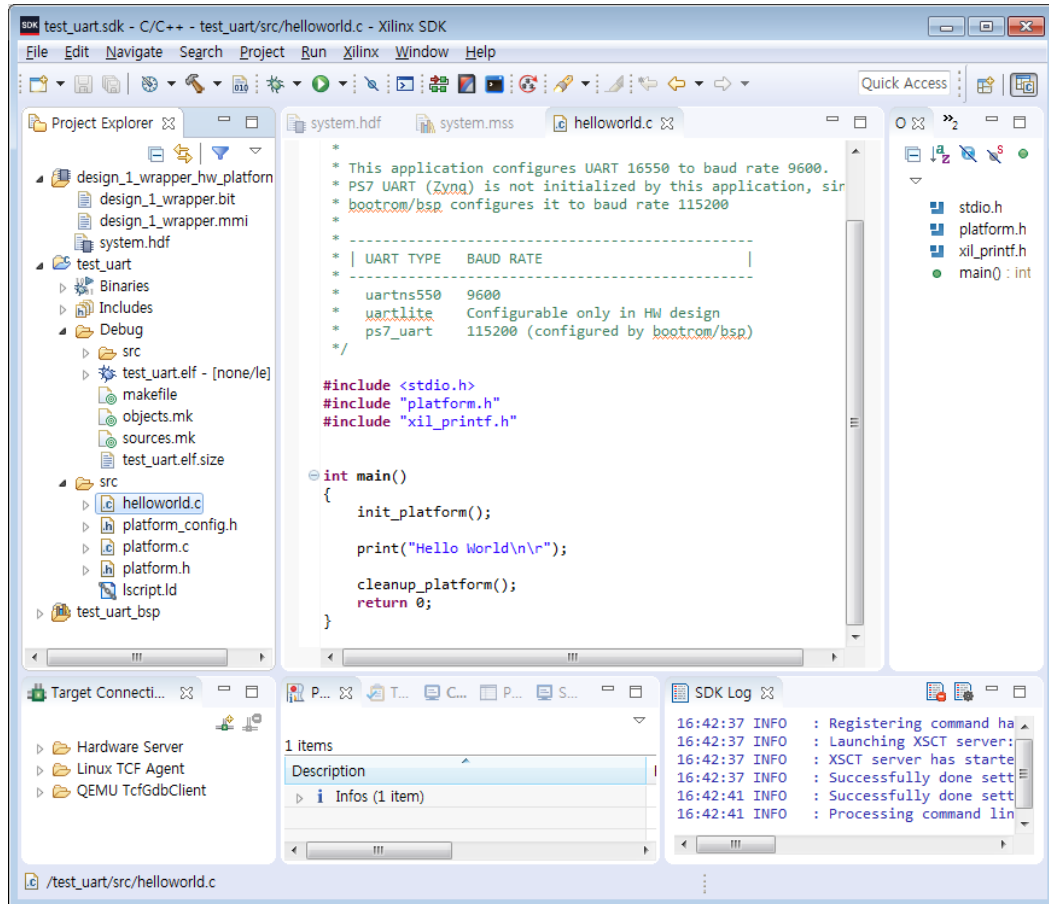
- 2) 하기 창이 나타나면 Project name 란에 test\_uart 라고 입력하고 Next 를 클릭한다.



- 3) 하기 창에서 Hello World 를 선택한 후 Finish 클릭



- 4) SDK 창의 Project Explorer 탭에서 test\_uart → Debug → src → Helloworld.c  
를 더블클릭하면 하기의 그림과 같이 나타난다.



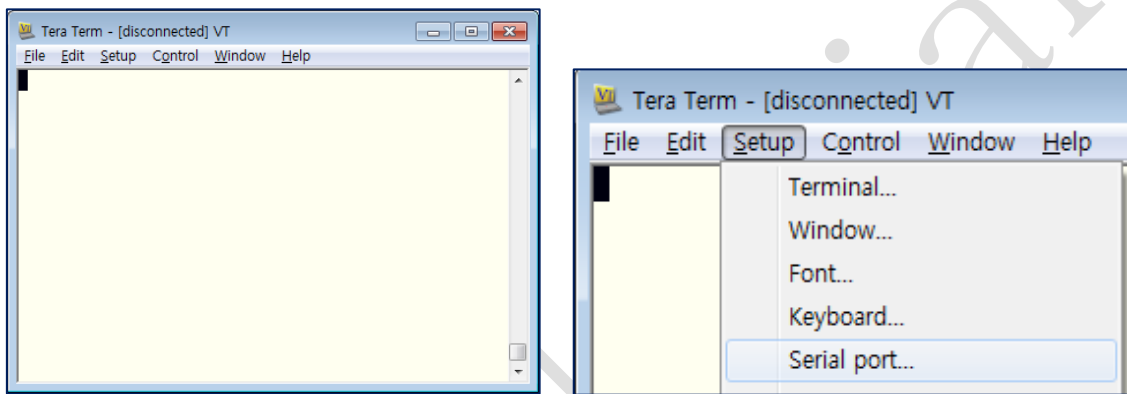
- 5) 상기 디자인을 하기의 내용과 같이 수정하고 저장한다.

```
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
```

```
int main()
{
    init_platform();
    while(1) {
        print("Libertron FSK3 Test\n\r");
    }
    cleanup_platform();
    return 0;
}
```

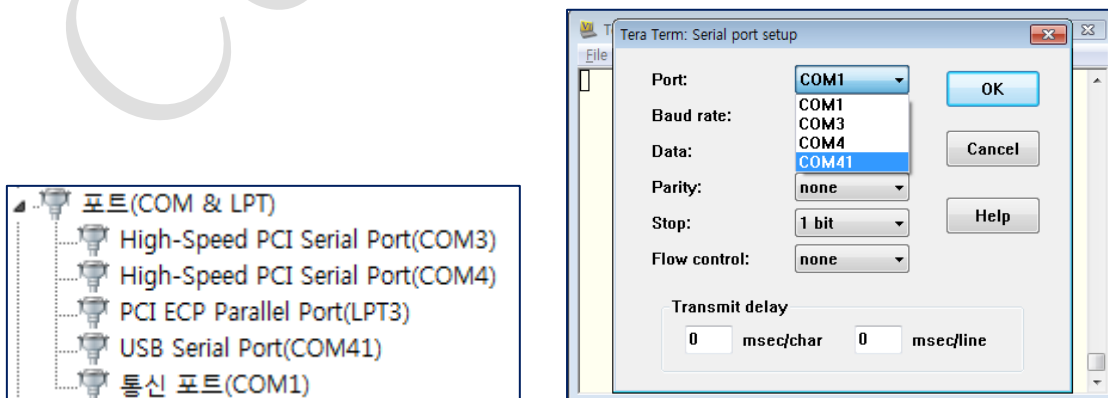
### 2.2.5-3 FSK III 에 UART 디자인 (bit file) 테스트를 위한 준비사항.

- 1) PC 와 FSK III 에 JTAG Cable (USB B Cable) 을 연결한다.
- 2) PC 와 FSK III 의 USB B Cable (UART 통신용) 을 연결하고 전원을 켜다.
- 3) PC 에 Tera-term 과 같은 통신 프로그램을 실행하여 통신 포트를 셋팅한다.



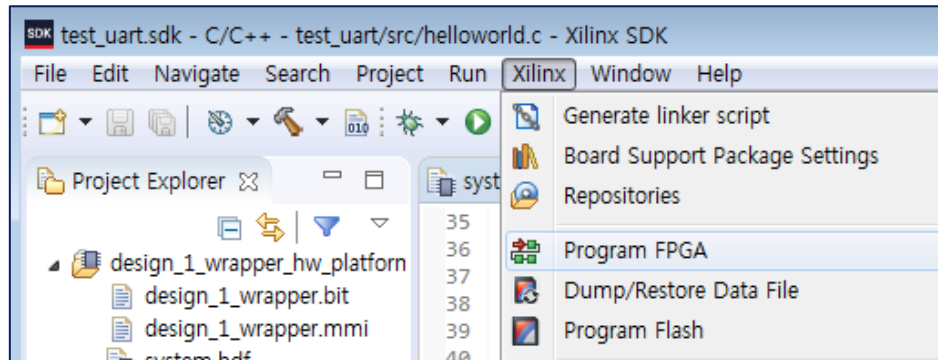
- 4) 하기 좌측 그림과 같이 PC 의 장치 관리자에서 USB Serial Port 가 몇번의 COM 포트에 연결되었는지 확인 하고, Teraterm 과 같은 통신 프로그램에서 COM 포트를 셋팅한다.

(Tera Term → Set up → Serial Port Setup → Baud rate : 115200)

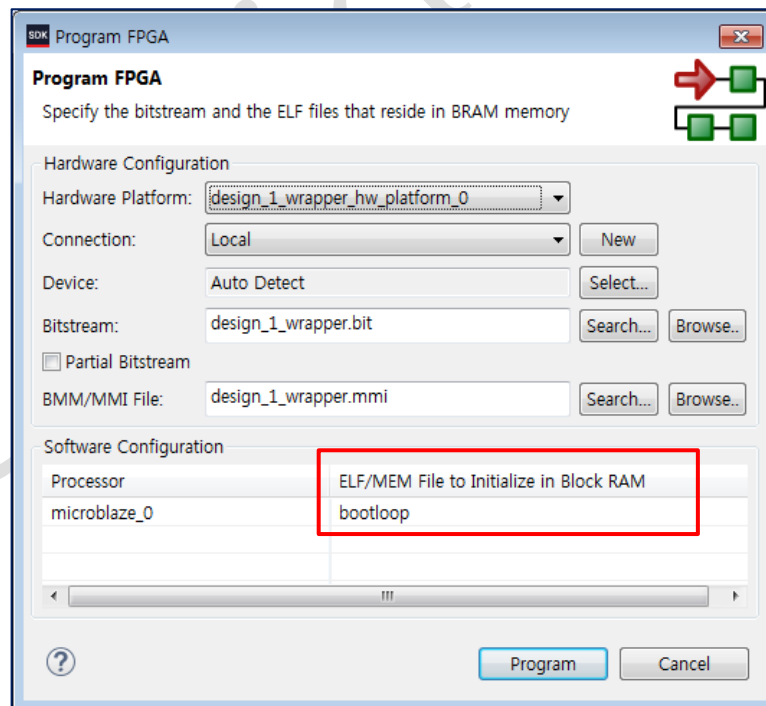


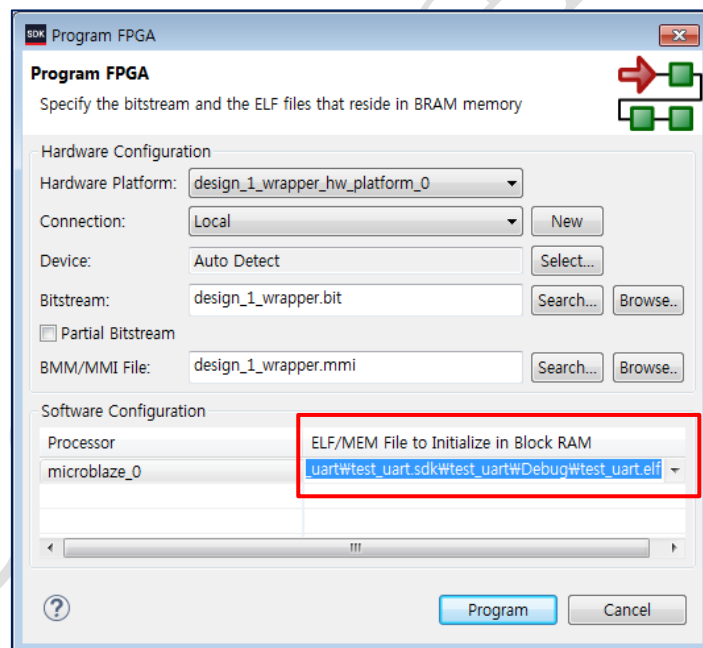
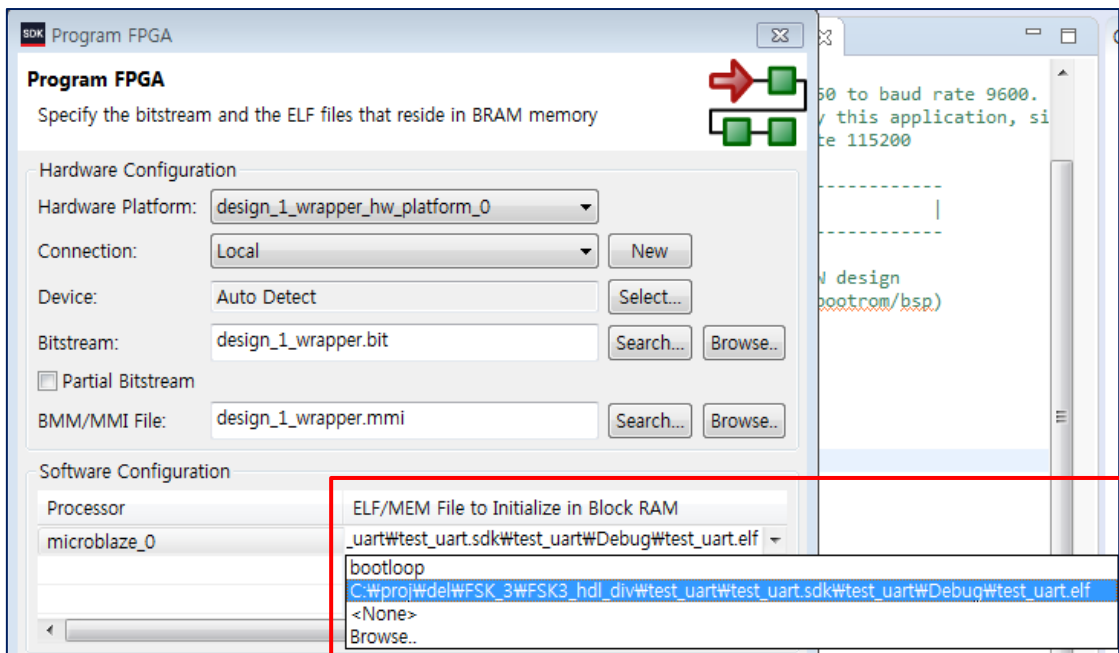
#### 2.2.5-4 FSK III 에 디자인 다운로드 (bit file + elf file)

- 1) 이전 내용으로 동작 준비를 완료한 후, 하기 그림과 같이 SDK 에서  
Xilinx → Program FPGA 를 선택한다.



- 2) Vivado 에서 hdf 파일을 bit file 과 함께 export 하기에, 하기 그림과 같이 Bitstream 란에 design\_1\_wrapper.bit 가 나타나는 것을 확인할 수 있다.  
또한, 하단의 microblaze\_0 란에 bootloop 로 되어 있는 것을 하기 test\_uart.elf 파일로 지정하여 선택한다.





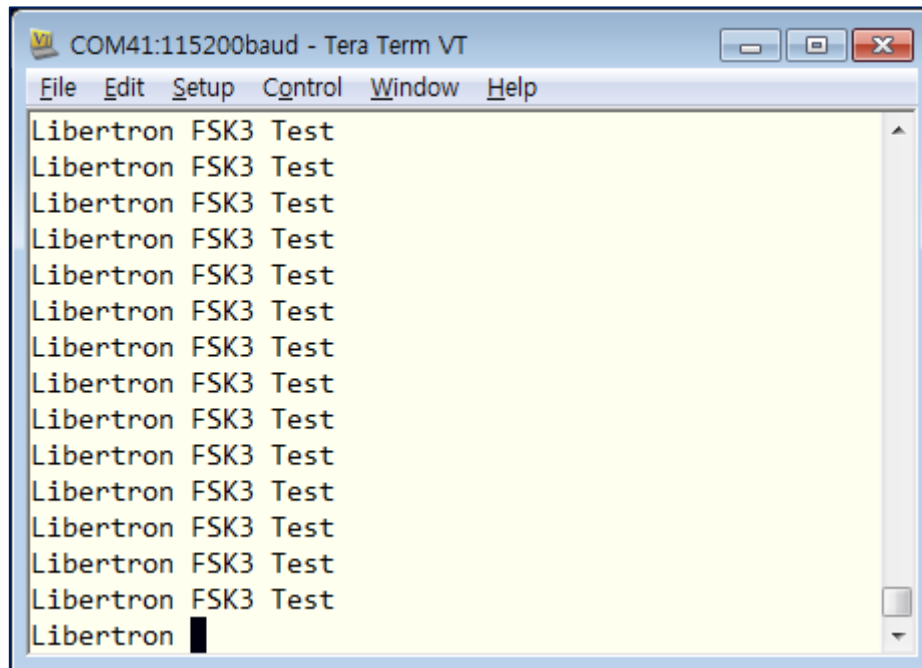
- 3) 상기와 같이 test\_uart.elf 파일로 지정하면, FPGA 에 다운로드 시 FPGA 영역의 bit 파일과 microblaze 영역의 다운로드 파일인 elf 가 순차적으로 같이 다운로드가 된다.

선택을 완료한 후 Program 을 클릭하여 FPGA 에 bit 파일과 elf 파일이 다



운로드 되도록 한다.

- 4) 다운로드가 완료되면 하기 그림과 같이 Teraterm 통신 프로그램에서 FSK III 가 출력하는 Libertron FSK3 Test 라는 문구가 반복적으로 출력되는 것을 확인할 수 있다



감사합니다.

### Revision History

Ver	Date	Revision
1.0	2020-06-02	Initial Document Release.