실험 4. HLS(High Level Synthesis) 실습

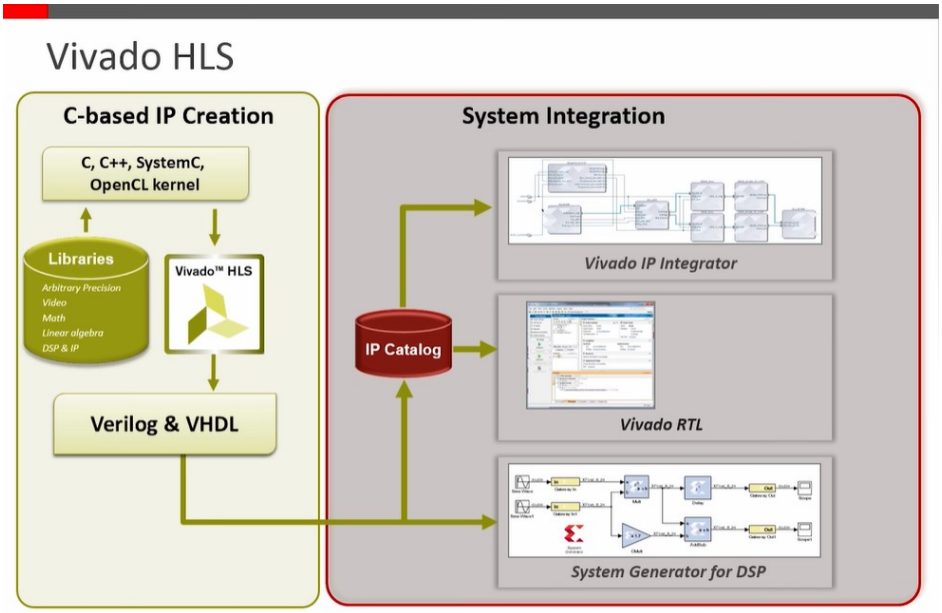
작성자: 박천명

1. 실험 목적

이 실험에서는 Xilinx의 HLS (High Level Synthesis) tool 을 이용하여 matrix multiplication 을 수행하는 HW를 설계하고, 설계한 HW을 Xilinx UltraScale+ MPSoC ZCU104 보드에 올려 multiplication을 올바르게 수행하는 지를 확인한다.

2. 실험 전에 준비해야할 내용

HLS(High level Synthesis)는 C / C++ / System C 와 같은 High level 의 language 를 사용하여, RTL coding 및 verification을 수행하는 것을 의미한다. High level language 로 HW design을 기술하면 HLS tool이 이를 RTL language인 Verilog로 generation을 해준다. Xilinx 사의 HLS tool인 Vivado HLS는 다양한 C library를 지원하여 C 언어를 이용한 coding 및 검증이 용이하도록 하였고, high performance RTL design 을 위해 DSP 및 IP library 또한 지원하고 있다.



Vivado HLS 에 대한 더욱 자세한 내용은 아래 링크에서 확인할 수 있다.

https://www.xilinx.com/video/hardware/vivado-high-level-synthesis.html

3. 실습 실험을 진행하기 위한 환경 세팅

a) Xilinx Vivado 설치 (ver. 2019.2)

아래 링크에서 다운받을 수 있다 (설치가 오래 걸리니, 실험 전 설치 요망)

https://www.xilinx.com/support/download/index.html/

b) Board Setup

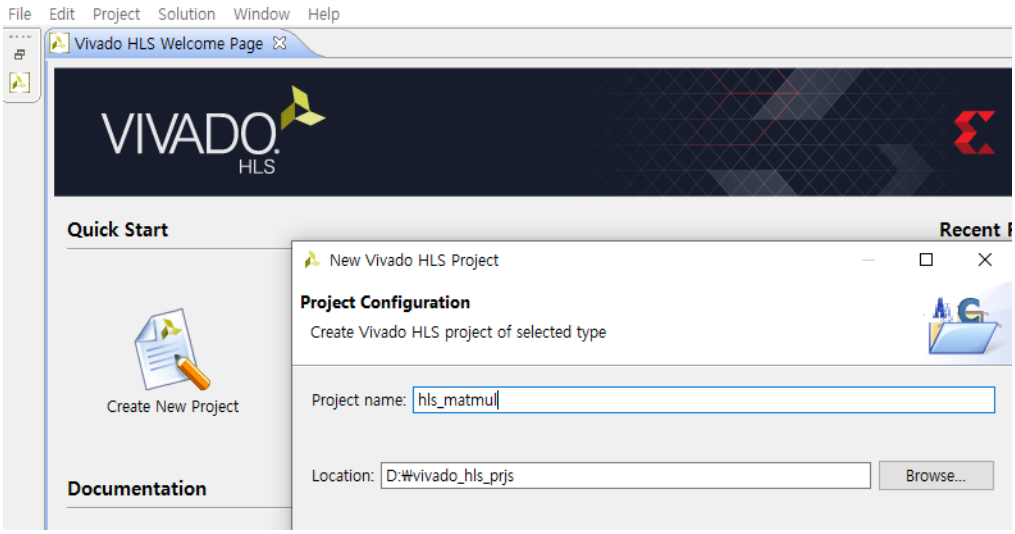
실험 3 과 동일하게 set up

c) Xilinix ZCU104 보드에 wifi dongle를 연결하여 인터넷 접속

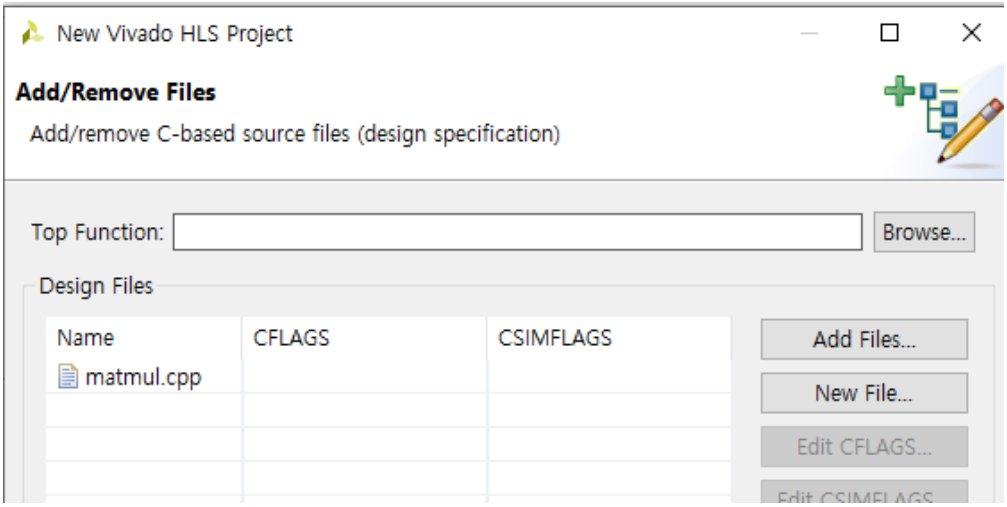
실험 3 과 동일하게 set up

4. HLS(High Level Synthesis) 실습 실험

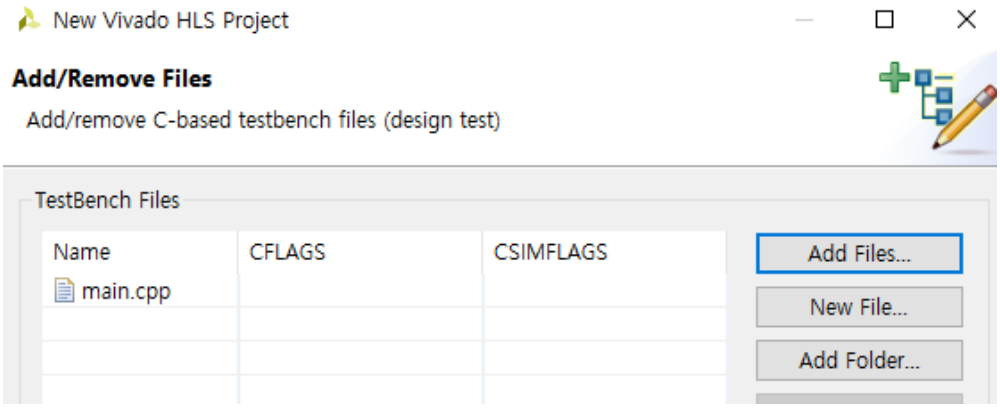
a) Create New Project on Vivado HLS



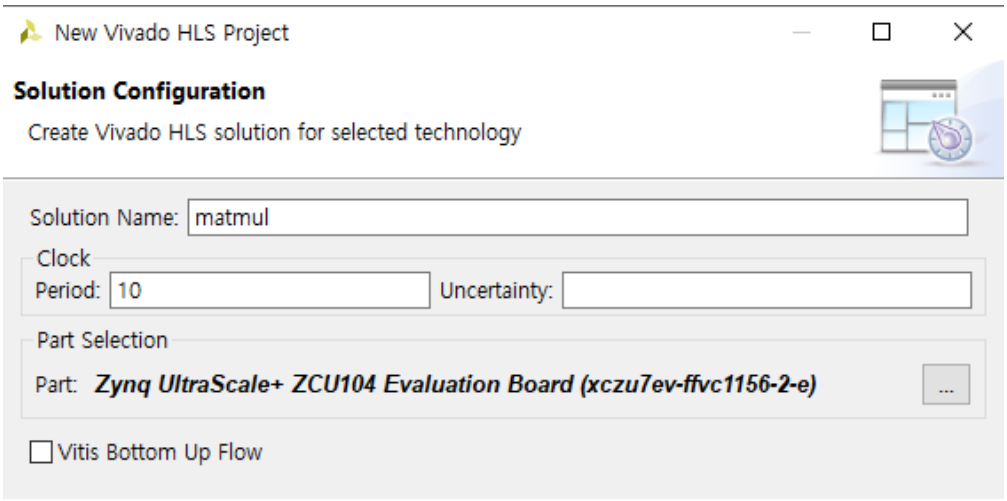
b) Add source file (matmul.cpp)



c) Add testbench file (main.cpp)

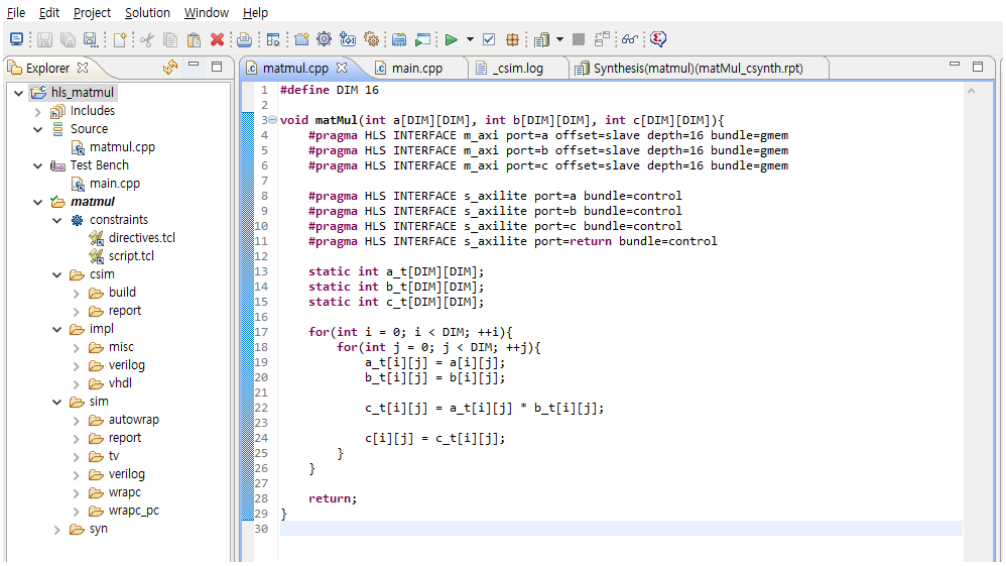


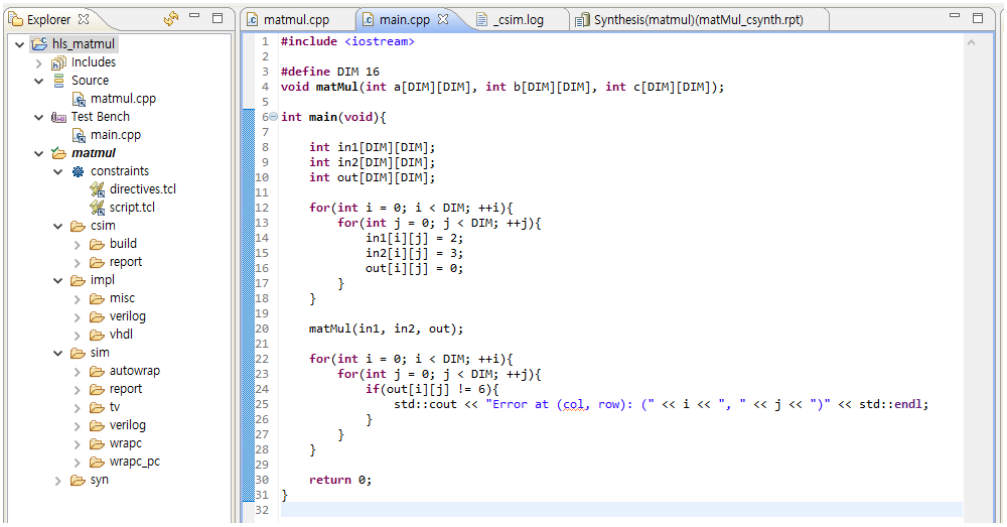
d) Select ZCU104 for part selection



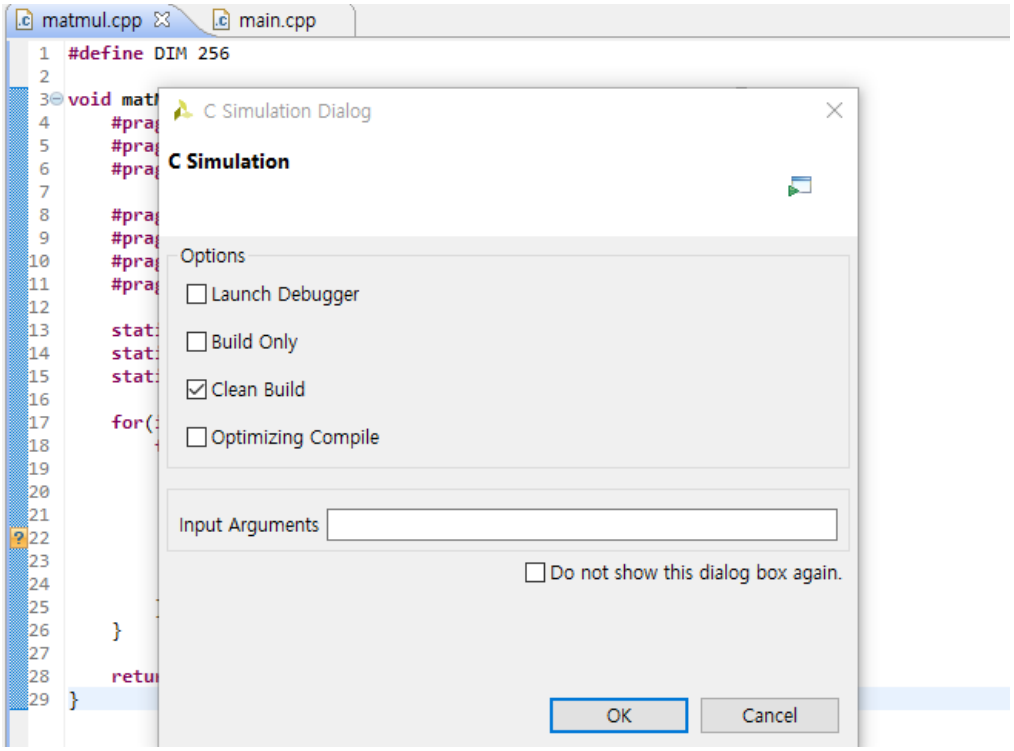
e) HLS project 생성을 완료하면, 아래와 같이 source code로 추가된 matmul.cpp와

testbench code로 추가된 main.cpp를 확인할 수 있다.

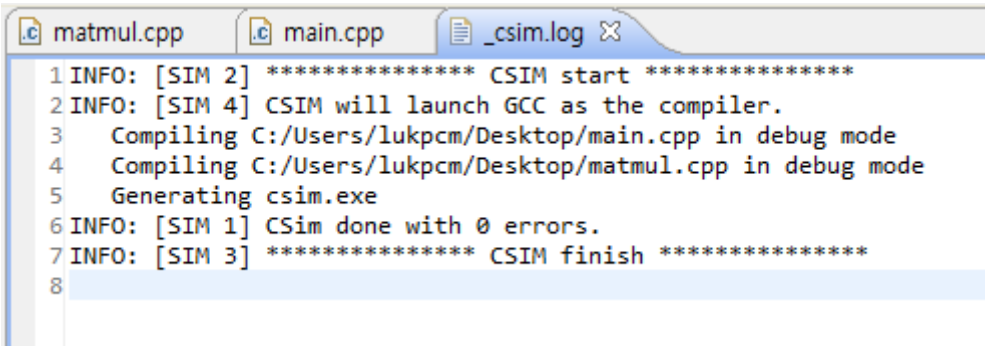




f) 상단 아이콘을 클릭하여, C simulation을 수행한다.



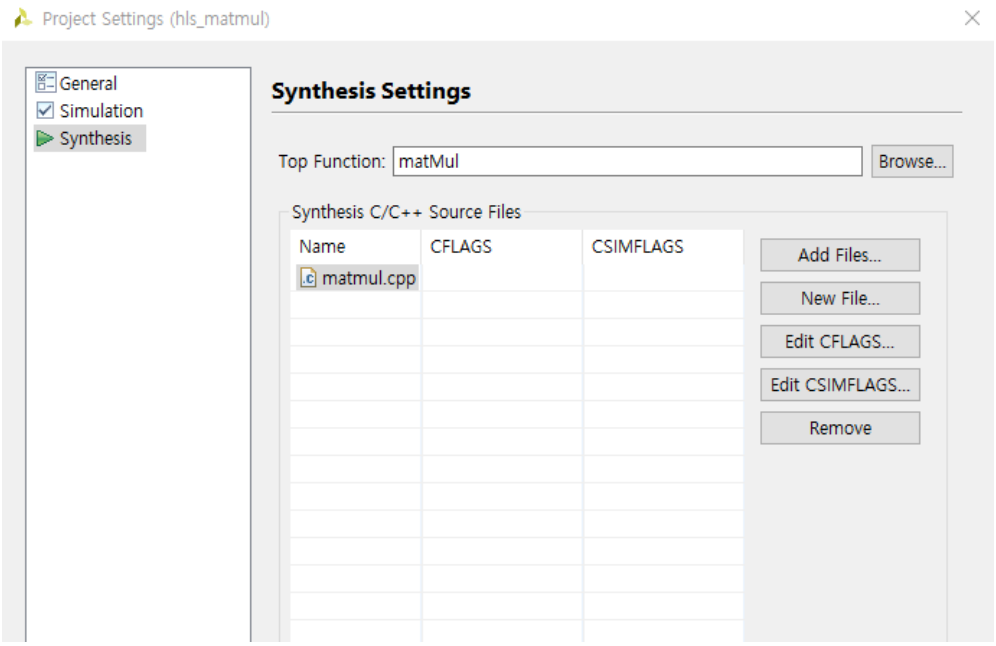
g) C simulation 이 정상적으로 완료되면 아래와 같은 log 파일을 확인할 수 있다.



h) Set top function

다음 단계인 C synthesis로 넘어가기 위해서는 top function을 설정해야 한다.

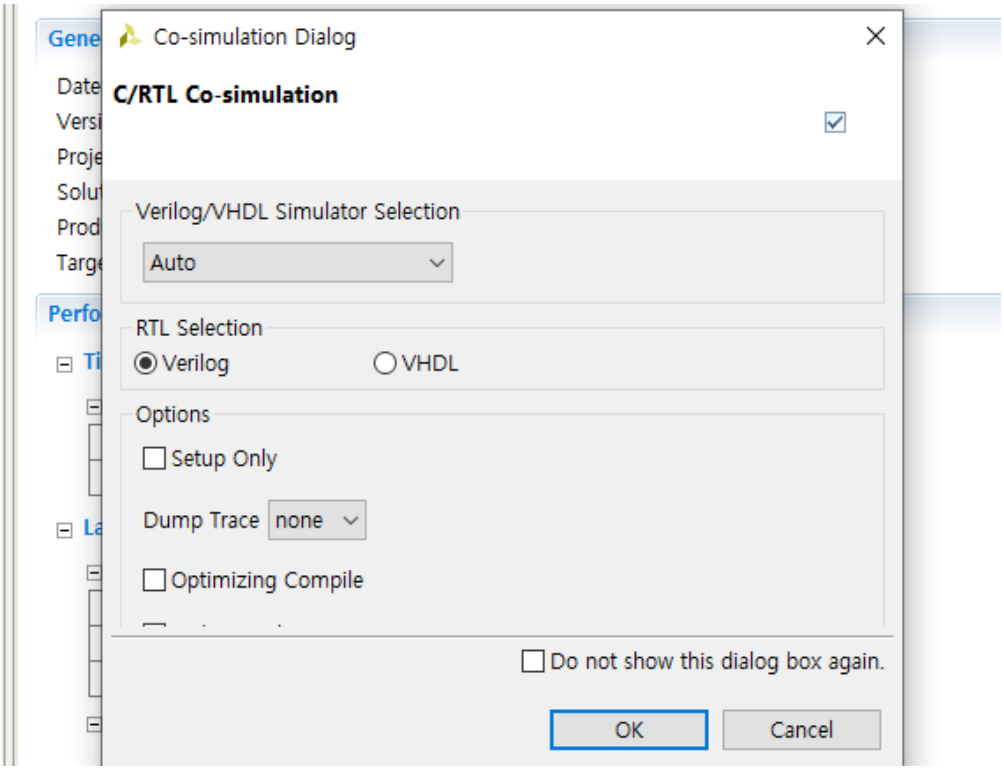
Project setting에서 matmul.cpp 파일을 top function으로 설정한다.



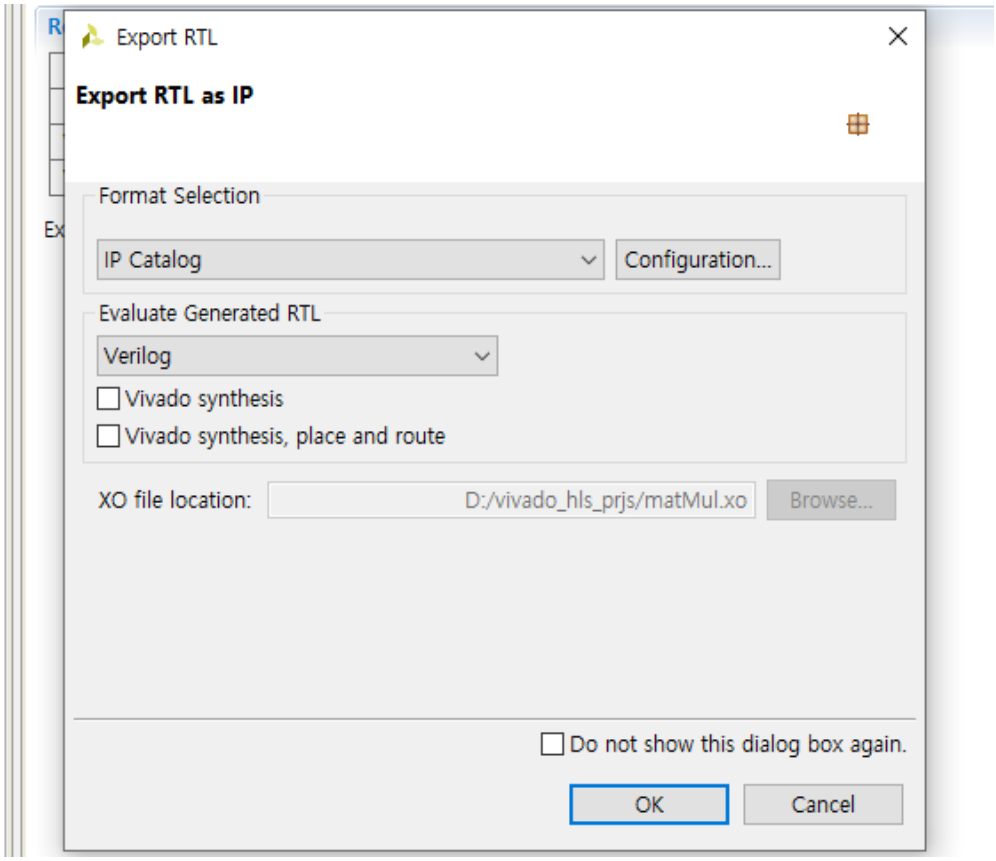
i) Run C synthesis

상단 아이콘을 클릭하여 C synthesis를 수행하면, synthesis report를 확인할 수 있다. Synthesis report에서는 timing, latency, utilization 등을 확인할 수 있다.

j) Run C/RTL cosimulation

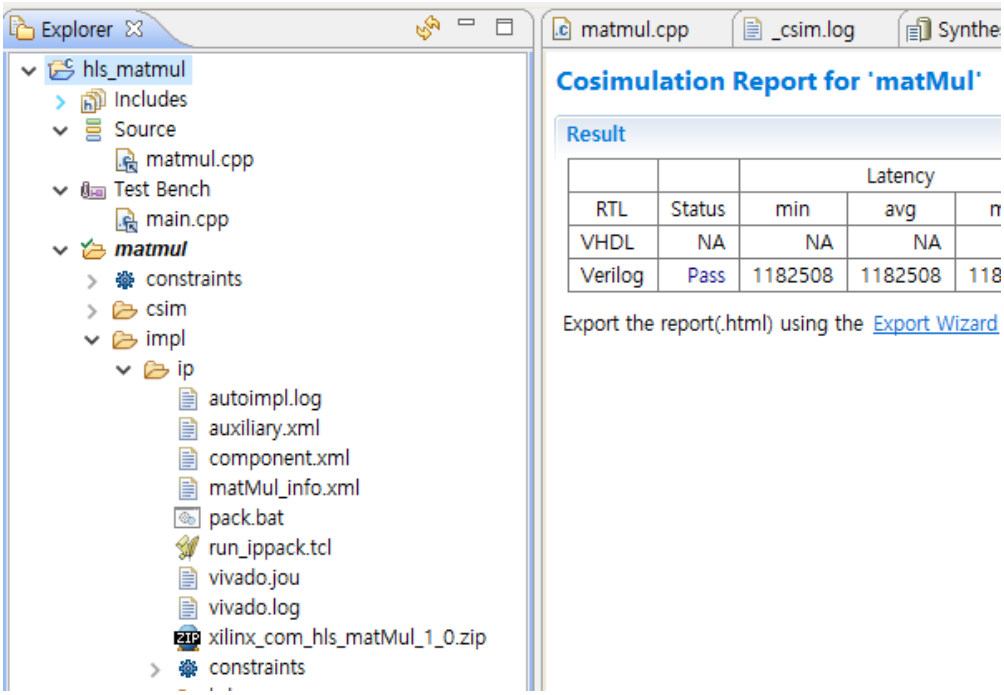


k) Export RTL



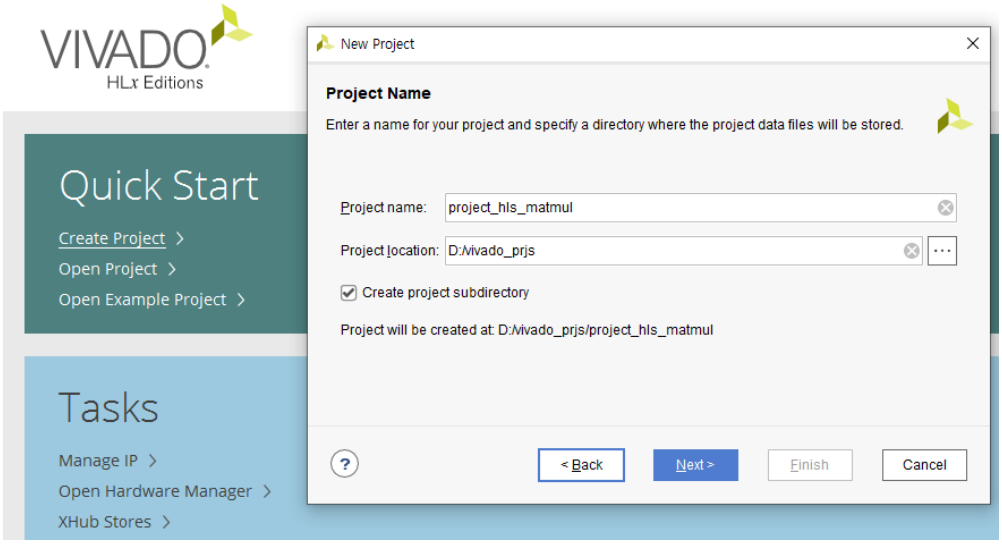
l) Check IP

Export RTL 이 된다면, 아래와 같이 'impl' directory 아래 'ip' directory가 생성되며, ip 가 정상적으로 생성된 것을 확인할 수 있다.



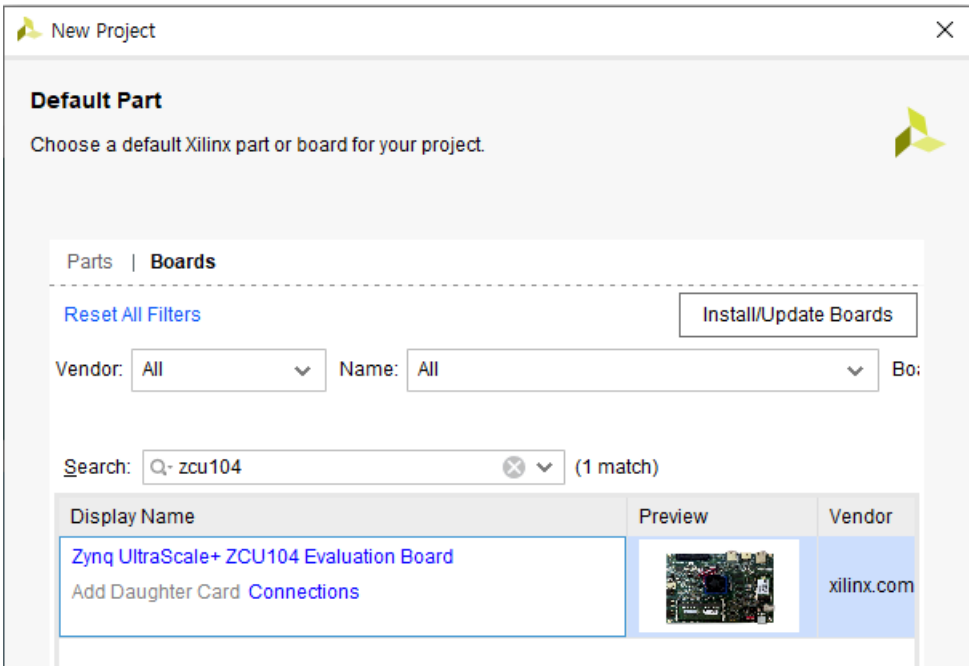
m) Create Vivado project (not HLS)

다음으로 Xilinx Vivado tool을 실행하여 project를 생성한다.



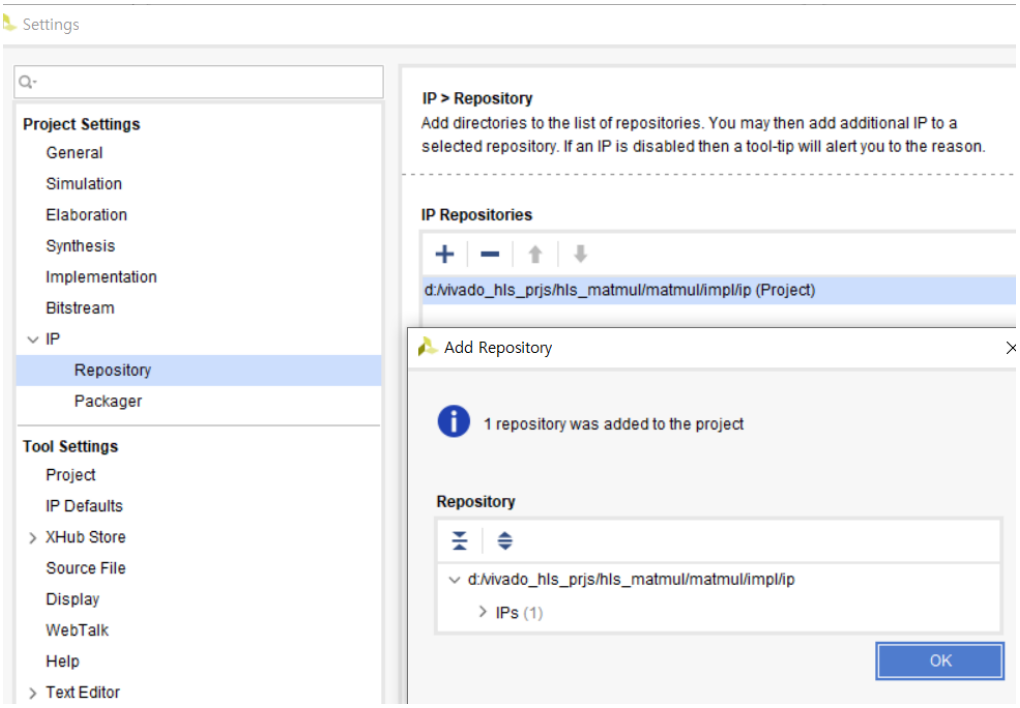
n) Board selection

마찬가지로 board는 ZCU104 보드를 선택한다.



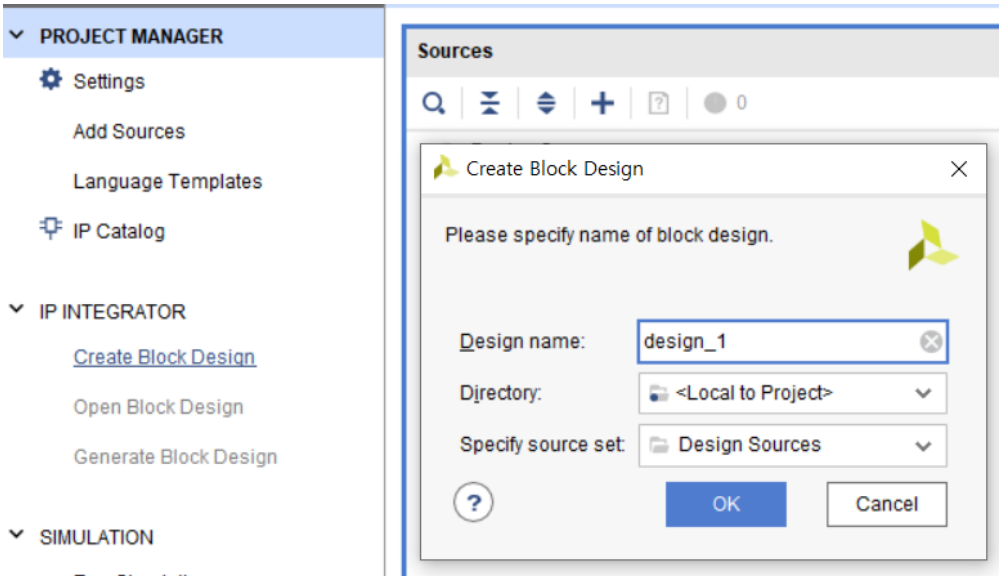
o) Add IP repository

Project setting에 들어가서 IP repository를 추가한다. HLS project에서 생성한 ip의 디렉토리 path를 입력하면 된다.



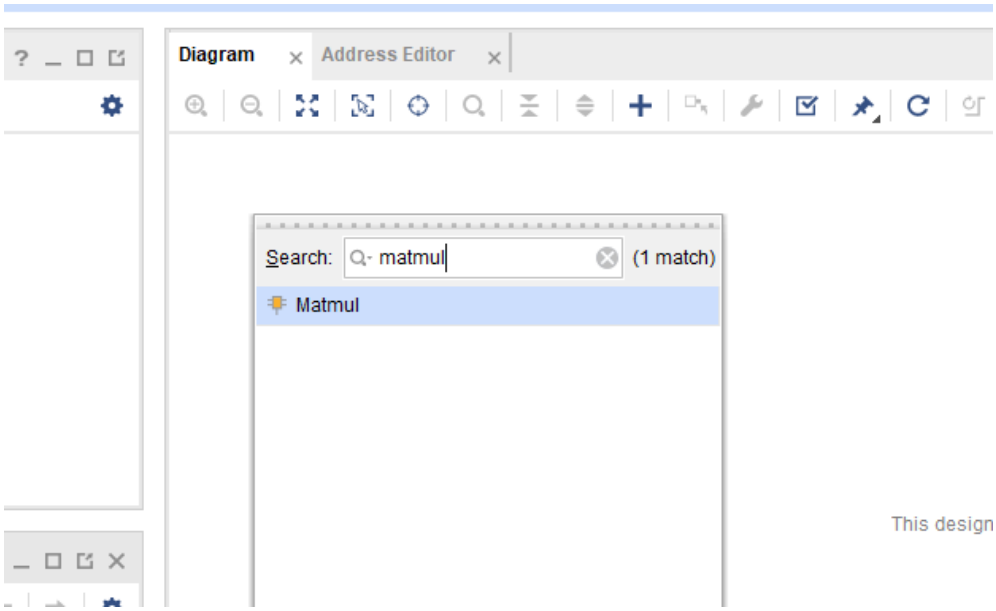
p) Create block design

좌측 네비게이션 바에서 Create block design을 선택하여 block design을 생성한다.



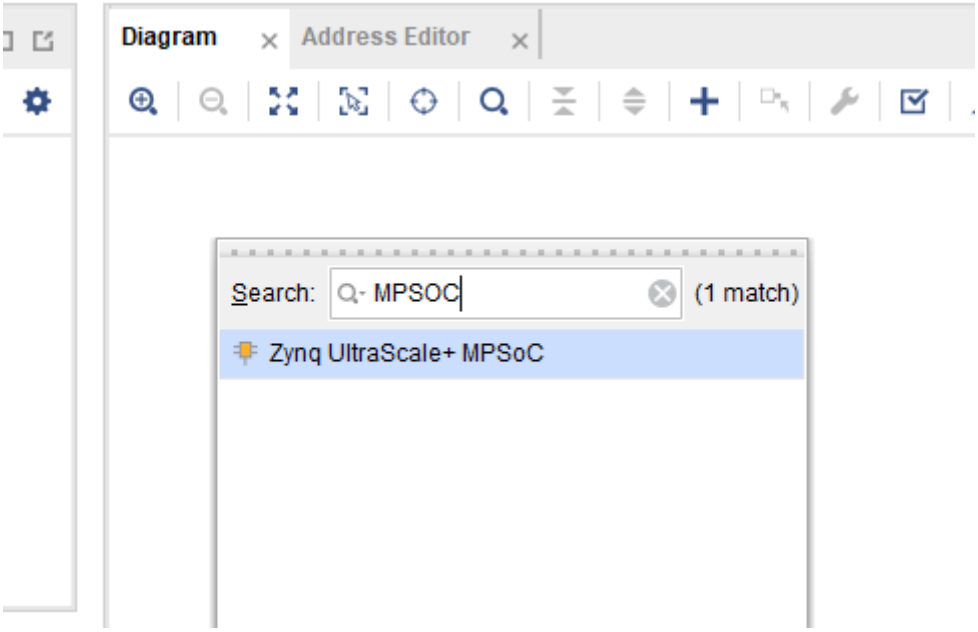
q) Add IP (matmul)

+ 버튼을 눌러 IP를 추가한다. 앞서 project setting에서 IP repository가 정상적으로 추가되었다면, HLS project에서 생성한 IP인 matmul을 아래와 같이 추가할 수 있다.

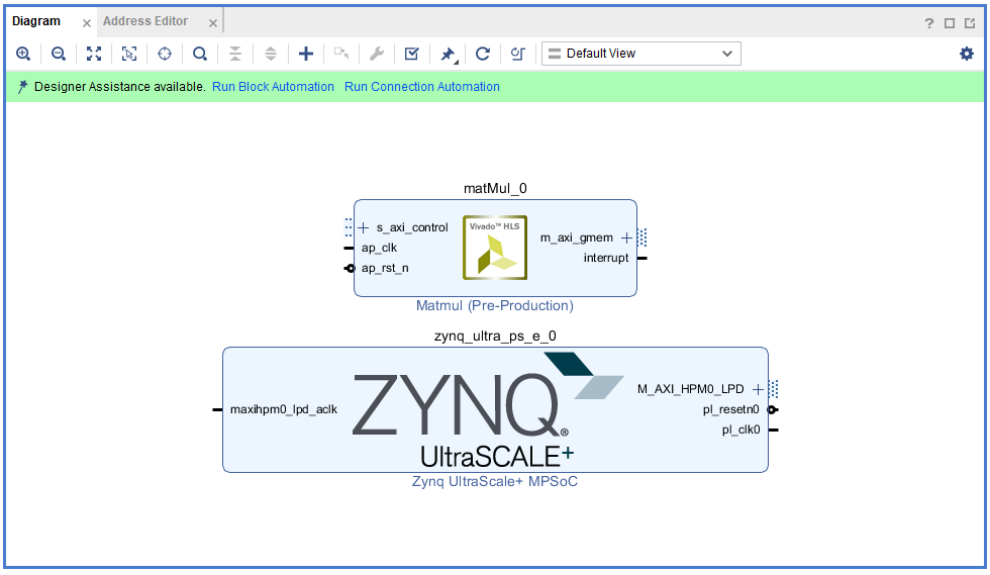


r) Add IP (ZYNQ Ultrascale+ MPSOC)

ZCU104 보드의 processor에 해당하는 ZYNQ Ultrascale+ MPSOC IP도 같은 방법으로 block design에 추가한다.



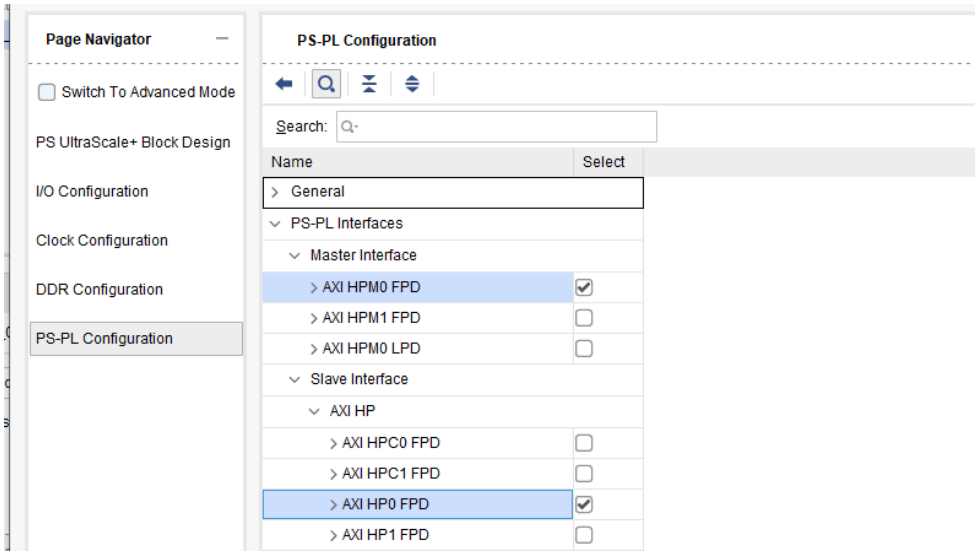
두 IP가 모두 추가되었다면 아래와 같이 diagram 화면에 2개의 모듈을 확인할 수 있다.



s) Re-customize IP

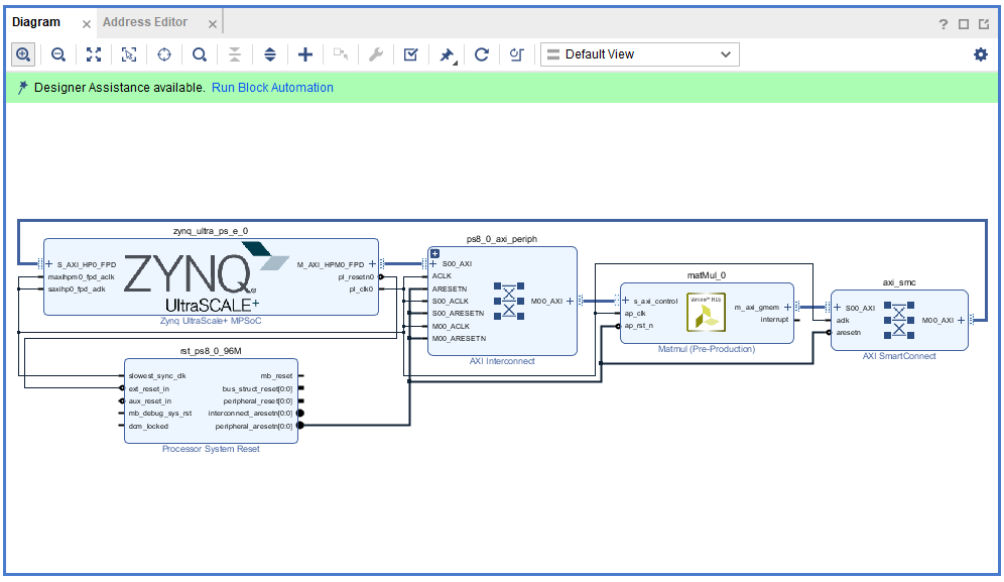
ZYNQ Ultrascale+ MPSOC 모듈을 더블 클릭하여, master interface와 slave

interface 설정을 수정한다.

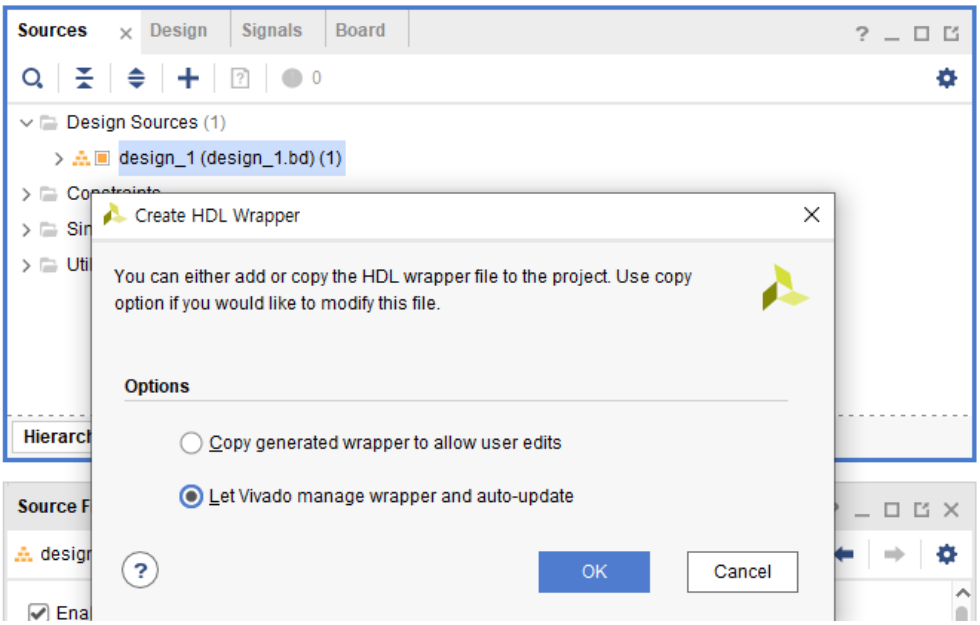


t) Run Connection Automation

초록색으로 highlight 되어 있는 바에서 Run connection Automation을 실행하면, processor 모듈과 matmul 모듈의 연결이 완료된다. (Run block Automation은 건들지 말 것!)



u) Create HDL wrapper



v) Run Synthesis → Run Implementation → Generate Bitstream

w) Export bitstream (file → export → export bitstream file)

x) Export 완료 후, bitstream 파일과 hwh 파일 확인

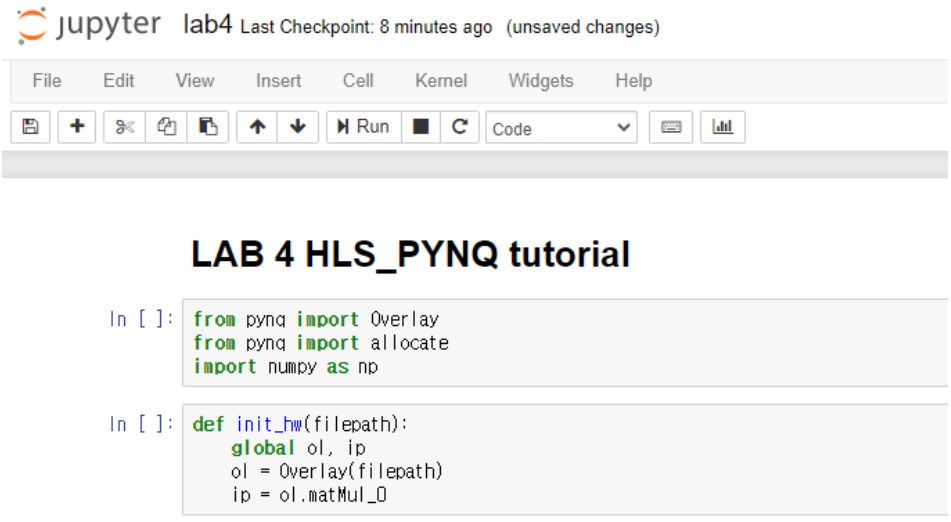
(hwh 파일은 .srcs/sources_1/bd/design_1/hw_handoff 에서 확인 가능)

y) ZCU104 board 연결 상태 및 무선 네트워크 연결 상태 확인

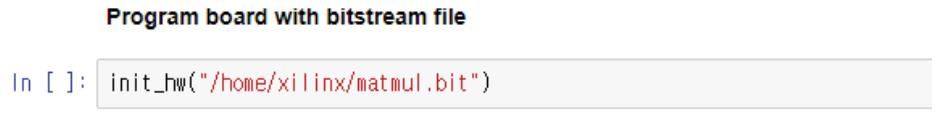
z) 앞단계서 확인한 bitstream file 과 hwh file 그리고 실습에 사용할 lab4.ipynb 파일을 Windows terminal를 이용해 board로 옮긴다.

```
C:₩Users       .>scp Desktop₩matmul.bit xilinx@192.168.50.101:/home/xilinx
xilinx@192.168.50.101's password:
matmul.bit                                                    100%   18MB 650.0KB/s   00:29

C:₩Users,      >scp Desktop₩matmul.hwh xilinx@192.168.50.101:/home/xilinx
xilinx@192.168.50.101's password:
matmul.hwh                                                    100%  253KB 640.7KB/s   00:00

C:₩Users\      .>scp Desktop₩lab4_matmul.ipynb xilinx@192.168.50.101:/home/xilinx
xilinx@192.168.50.101's password:
lab4_matmul.ipynb                                             100% 6051   109.8KB/s   00:00

C:₩Users₩lukpcm>
```

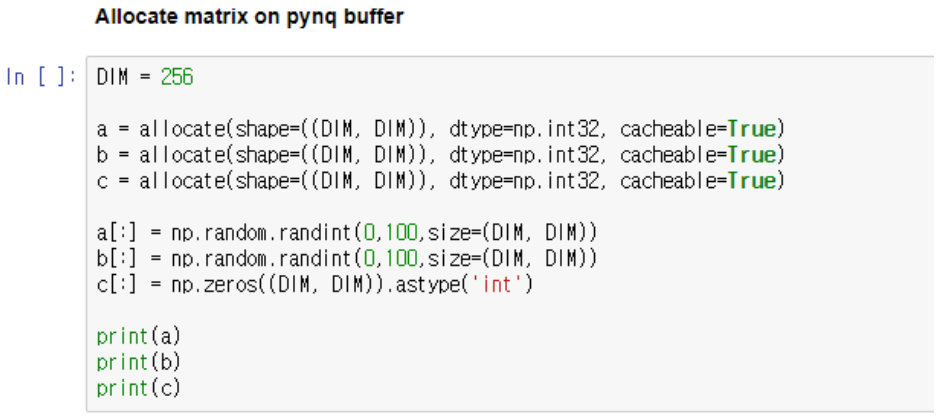
aa) PC 에서 보드IP주소에 접속해서 lab4.ipynb file open.



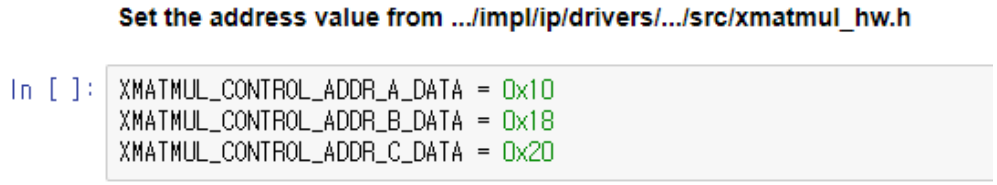
bb) Bitstream 파일 불러와서 program FPGA



cc) Allocate matrix on pynq buffer



dd) Vivado project path에 있는 xmatmul_hw.h 파일 참조하여 address 값 설정



ee) Write data to input port

Write data to input port

In [ ]:
```
a_ptr = a.physical_address
b_ptr = b.physical_address
c_ptr = c.physical_address

ip.write(0x00, 4)
fpga_state = ip.read(0x00)

if fpga_state == 4:
    ip.write(XMATMUL_CONTROL_ADDR_A_DATA, a_ptr)
    ip.write(XMATMUL_CONTROL_ADDR_B_DATA, b_ptr)
    ip.write(XMATMUL_CONTROL_ADDR_C_DATA, c_ptr)
else:
    print("Can't write values, must be in IDLE state")
    raise KeyboardInterrupt
```

ff) Start multiplication

Start multiplication

In [ ]:
```
ip.write(0x00, 1)
fpga_state = ip.read(0x00)

max_try = 1000000
while fpga_state != 6 and fpga_state != 4:
    fpga_state = ip.read(0x00)
    max_try = max_try - 1
    if max_try == 0:
        print("ERROR: Can't go ahead")
        ip.write(0x00, 4)
        raise KeyboardInterrupt

print(c)
```

gg) Verify the output

board에서 계산한 값과 numpy로 계산한 값을 비교하여 검증

Verify the output

In [ ]:
```
if (c == a*b).any():
    print("Correct!")
else:
    print("Wrong!")
```

5. 실험 후 보고서에 포함될 내용

i. Vivado HLS project에서 C Synthesis report에서 latency 와 utilization 결과를 작성하시오

ii. ZCU104 보드의 matrix multiplication 동작을 검증한 결과를 작성하시오. (jupyter notebook의 실행 결과를 캡처)

iii. Matrix의 크기를 16에서 32, 64로 바꾼 뒤(matmul.cpp 의 DIM 변수), synthesis report의 latency와 utilization 을 확인하고, 차이가 있다면 그 이유를 설명하시오

iv. Matmul.cpp에서 사용하고 있는 datatype은 int type으로 32bit format 이다. Data type을 8, 16, 24, 32 bit으로 바꾼 뒤 synthesis report의 latency와 utilization을 비교하고, 그 이유를 설명하시오.
(line15~17의 data type을 "ap_int<bit수>" 로 바꾸면 된다. ap_int.h 에 나와있는 definition 참고)

제출 기한 : 2021.11.12(금) 오후 11:59

제출 양식 : 이름_학번_보고서4.pdf (ex. 홍길동_2021-12345_보고서4.pdf)

*보고서는 pdf로 변환하여 제출