UG936

Lab9

JTAG-to-AXI

경북대학교 2022222447 박주동

Lab9

- ILA Core를 이용하여 JTAG-AXI Transactions를 디버깅해보기!

-> ILA Core를 JTAG-AXI Master IP에 넣고 advanced trigger와 capture 기능 사용하기!

- 이번 LAB 순서
- Step1: JTAG-to-AXI와 System ILA가 있는 IP design 만들기
- Step2: 타겟보드에 비트스트림을 업로드 및 JTAG-to-AXI Master 코어 사용하기
- Step3: ILA Advanced trigger 기능을 사용해서 AXI가 Read할 수 있도록 하기

JTAG-AXI Master 코어란?

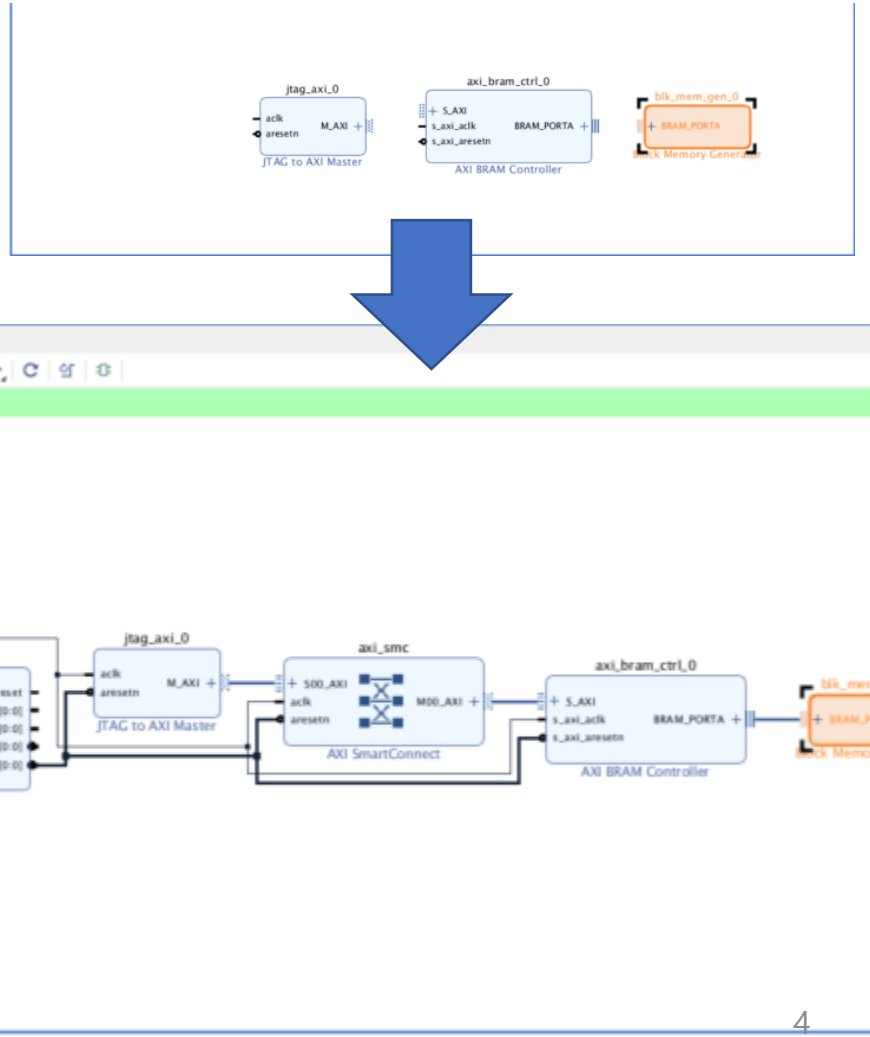
- AXI transactions(read, write)을 만듦 -> 그에 필요한 AXI 신호들을 control함
- AXI4-Stream을 제외한 AXI 인터페이스, AXI lite 프로토콜을 지원
- AXI data의 width는 변경가능 (32 or 64)

- Lab9: JTAG-AXI Master 코어는 **master**역할로서, AXI4-lite 혹은 BRAM 같은 Memory(**slave**)에 연결되어 사용됨

Step1: JTAG-to-AXI와 System ILA가 있는 IP design 만들기

- 프로젝트 열기!
- Block design 생성하기
- 다음 IP 추가하기
  - JTAG-to-AXI Master
  - AXI BRAM Controller
    - Number of BRAM interfaces: 1
  - Block Memory Generator
    - Enable Safety Circuit: 체크해제

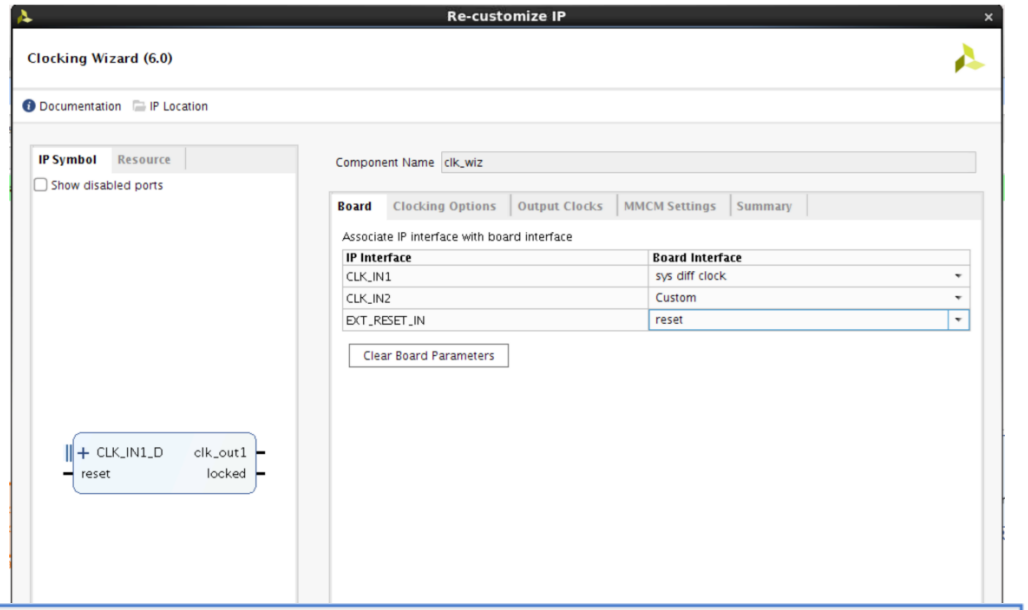
- Run Connection Automation 클릭
  - 모든 요소를 체크하고 Connection 진행
  - 다음 블록들이 자동으로 생성
    - Clocking Wizard
    - Processor System Reset
    - AXI SmartConnect

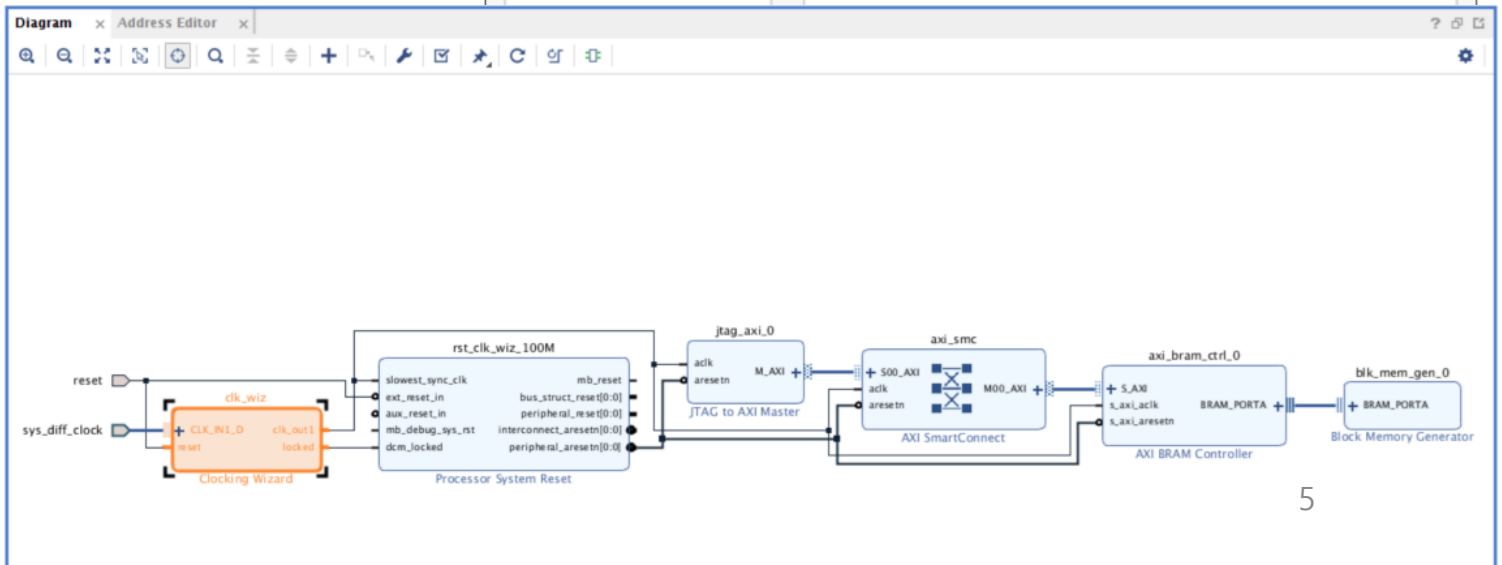


Step1: JTAG-to-AXI와 System ILA가 있는 IP design 만들기

- Clocking Wizard 블록 커스터마이징하기(Optional)
  - Clocking Wizard 블록 더블클릭
  - CLK_IN1: Custom -> sys_diff_clock
  - EXT_RESET_IN: Custom -> reset
  - Ok 클릭

- Run Connection Automation 클릭
  - 모든 요소를 체크하고 Connection 진행
  - sys_diff_clock, reset이 external ports와 연결된 것을 확인 가능
  - sys_diff_clock

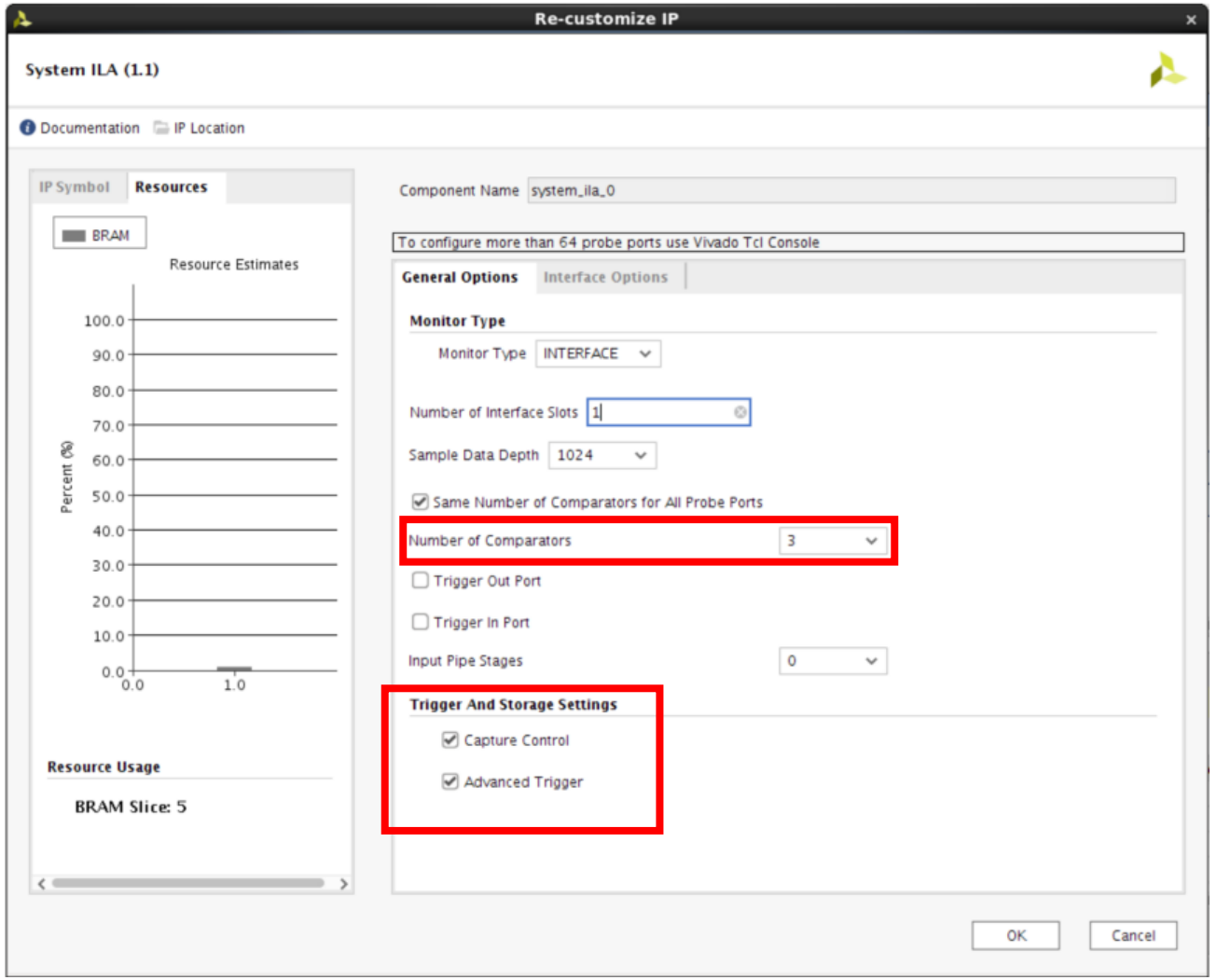




Step1: JTAG-to-AXI와 System ILA가 있는 IP design 만들기

- System ILA 코어 추가하기
  - Block diagram에서, “+” 클릭
  - System ILA 검색 및 추가

- ILA 설정 변경하기
  - ILA 더블클릭
  - Capture Control, Advanced Trigger: 체크
  - Number of Comparators: 3
  - Ok 클릭

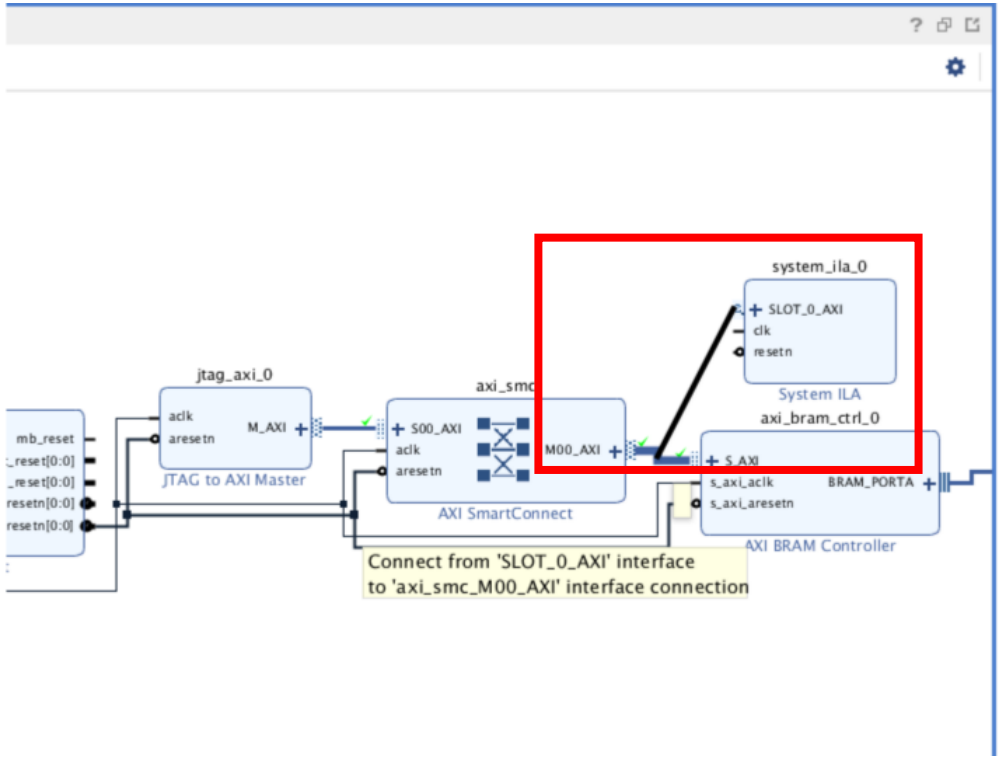


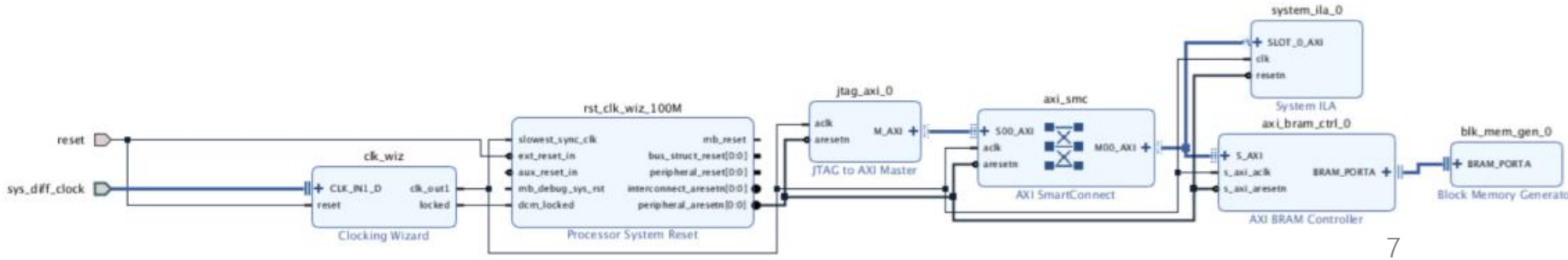
Step1: JTAG-to-AXI와 System ILA가 있는 IP design 만들기

- 기존 ILA 코어로 디버깅하고 싶은 Net들을 코어에 연결 하듯이,
  - system ILA의 SLOT_0_AXI를 AXI BRAM Controller와 AXI Smart Connection간의 path에 연결

- Run Connection Automation 클릭
  - 모든 요소를 체크하고 Connection 진행
  - System ILA의 나머지 부분도 연결

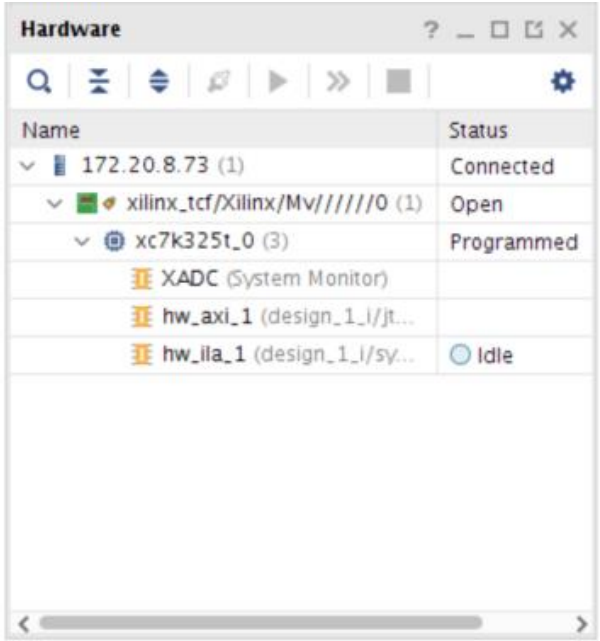
- File – Save Block Design
  - Block Design 닫기
- Sources 창에서, 만든 블록 디자인을 Verilog로 변환
  - Create HDL Wrapper 클릭
- 비트스트림 생성



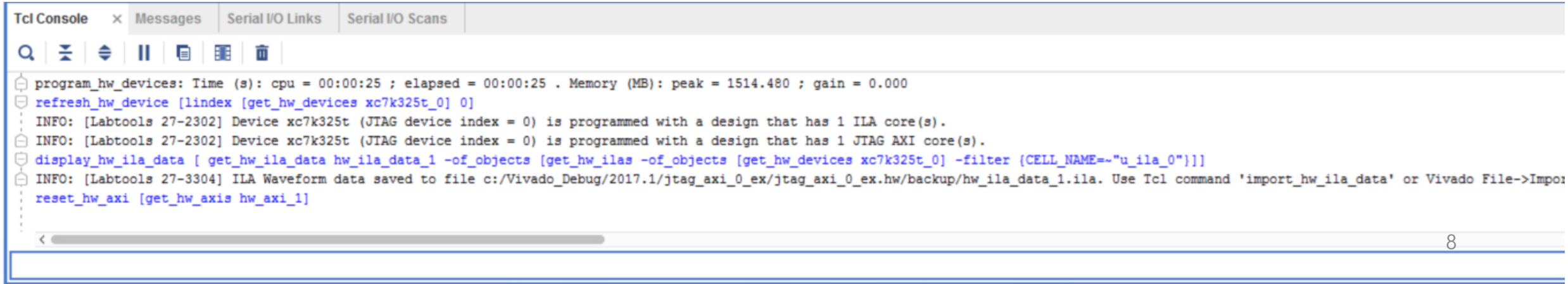


Step2: JTAG-to-AXI Master 코어 사용하기

- 타겟보드를 비바도에 연결
  - Hardware Manager – Auto Connet
- 생성한 비트스트림을 업로드
  - hw_axi_1: JTAG to AXI Master
  - hw_ila_1: ILA



- JTAG to AXI Master 코어는 Tcl 명령어로만 실행됨
- Transaction하기 전에, JTAG to AXI Master 코어를 reset시킴(중요)
- Reset_hw_axi [get_hw_axis hw_axi_1]



Step2: JTAG-to-AXI Master 코어 사용하기

- Write 명령어 생성: 4 Words단위로 AXI burst transaction을 설정 -> trigger시, BRAM의 처음 4개의 위치에 write
- **명령어:**
- **Set wt [create_hw_axi_txn write_txn [get_hw_axis hw_axi_1] –type WRITE – address C0000000 –len 128 –data {44444444_33333333_22222222_11111111}]**
- Write_txn: transaction 이름
- [get_hw_axis hw_axi_1]: write 신호를 줄 axi 코어
- -address C0000000: 시작 주소
- -len 128: AXI burst 길이가 128 word라는 의미
- -data {44444444_33333333_22222222_11111111}: 쓰여질 데이터

Step2: JTAG-to-AXI Master 코어 사용하기

- Read 명령어 생성: 128 Words단위로 AXI burst transaction을 설정 -> trigger시, BRAM의 처음 4개의 위치를 Read
- **명령어:**
- **set rt [create_hw_axi_txn read_txn [get_hw_axis hw_axi_1] -type READ - address C0000000 -len 128]**
- read_txn: read시 사용되는 transaction 이름
- [get_hw_axis hw_axi_1] :read 신호를 줄 axi 코어
- -address C0000000: 시작주소
- -len 128: AXI burst 길이가 4 words라는 의미

Step2: JTAG-to-AXI Master 코어 사용하기

- Transaction을 다 만든 뒤에, 만든 명령어를 이용해서 axi로 bram에 read/write할 수 있음

Write

- axi가 write하겠다고 신호를 보냄 -> write transaction이 일어남 -> bram의 처음 4개의 주소에 데이터가 write됨
- 명령어: run_hw_axi $wt

```
INFO: [Labtools 27-147] : WRITE DATA is :
44444444333333332222222211111111…
```

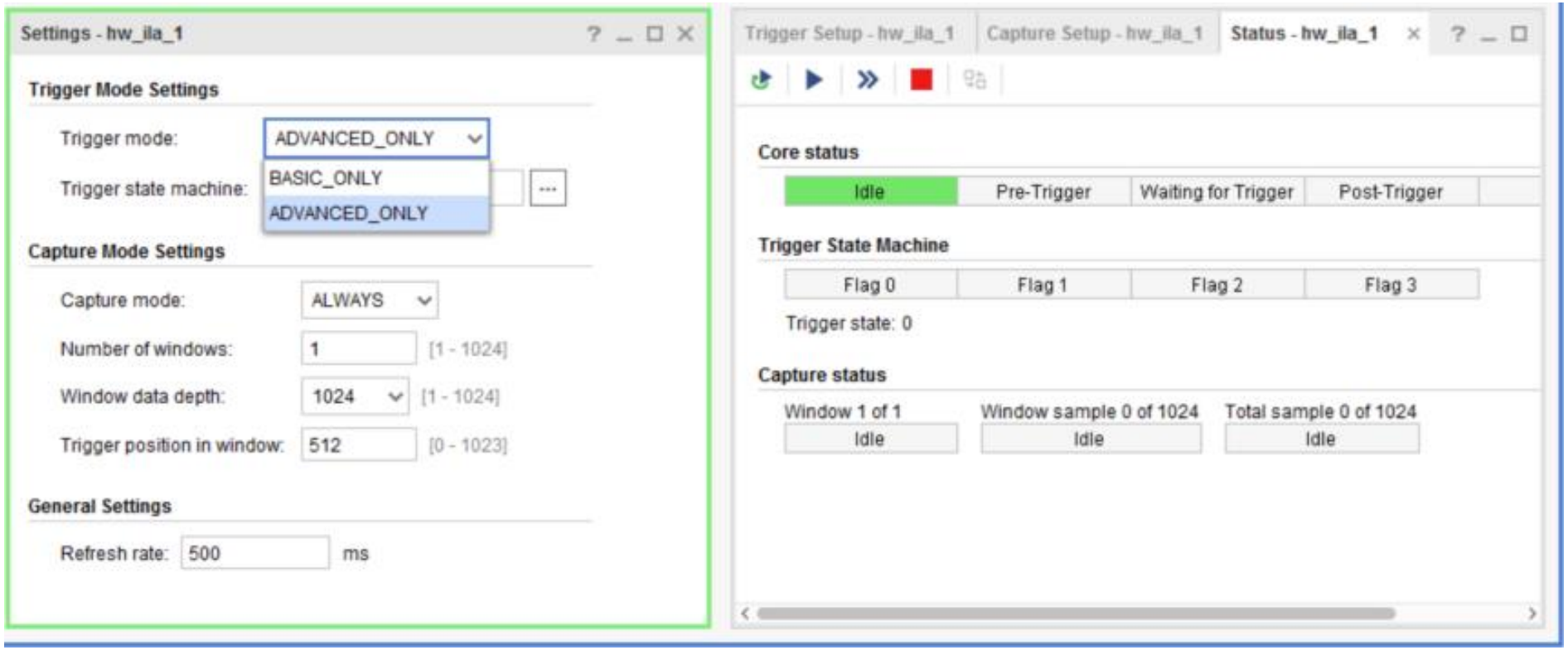
Read

- Axi가 read하겠다고 신호를 보냄 -> read transactio이 일어남 -> bram의 처음 4개의 주소의 데이터가 read됨
- 명령어: run_hw_axi $rt

```
INFO: [Labtools 27-147] : READ DATA is :
44444444333333332222222211111111…
```

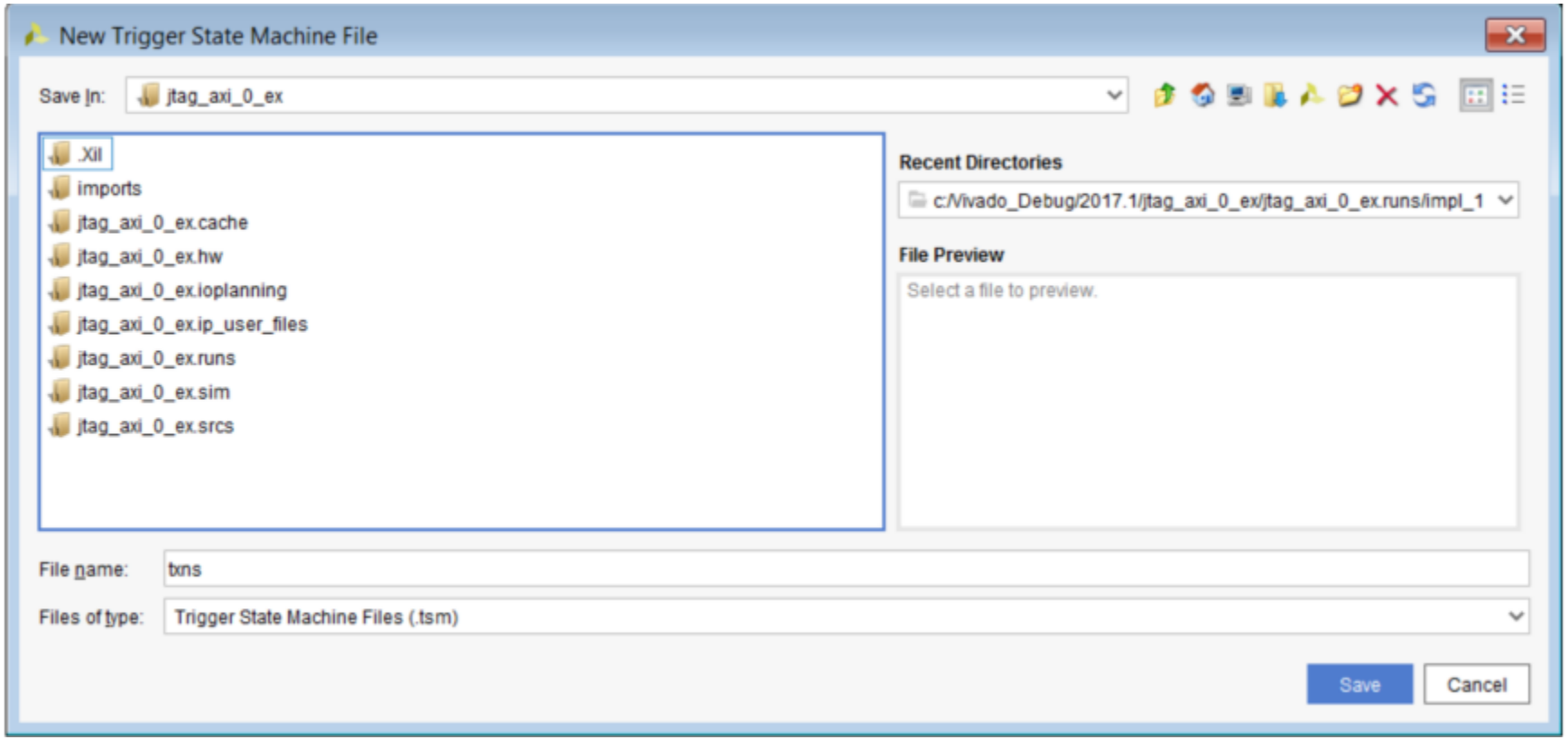
Step3: ILA Advanced trigger 기능을 사용해서 AXI가 Read할 수 있도록 하기

- (ILA) hw_ila_1 더블클릭
  - Trigger Mode Settings - Trigger mode - **ADVANCED_ONLY** 설정
  - Capture Mode Settings - Trigger position: **512**
  - Trigger State Machine - **Create new trigger state machine** 클릭



Step3: ILA Advanced trigger 기능을 사용해서 AXI가 Read할 수 있도록 하기

- Trigger State Machine - **Create new trigger state machine** 클릭
- state machine script 추가(txns.tsm)



Step3: ILA Advanced trigger 기능을 사용해서 AXI가 Read할 수 있도록 하기

- txns.tsm
  - AXI read 상태변화의 단계를 관찰하는데 쓰임

Hand-Shake 원리로 동작

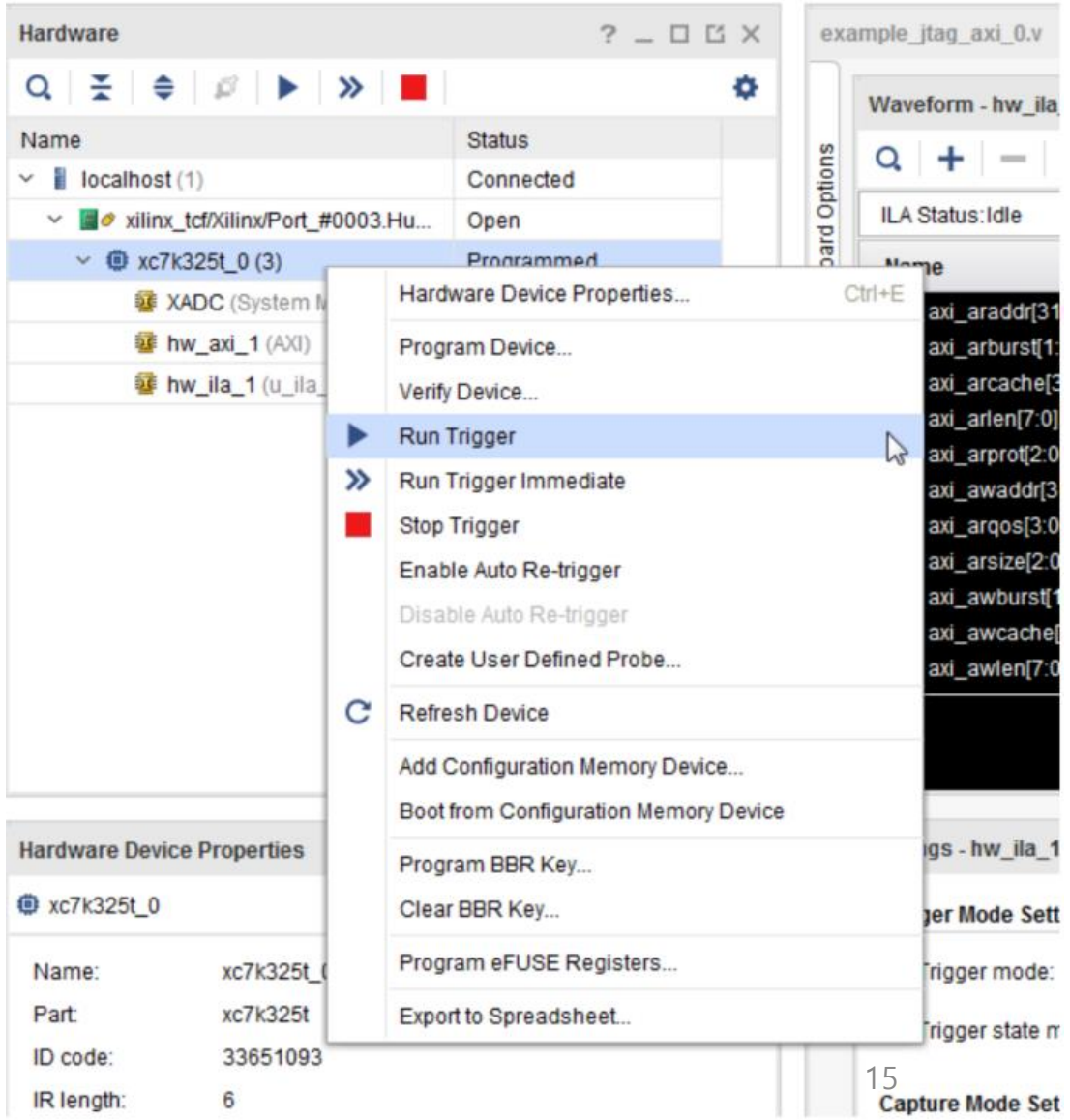
- Read할 주소를 전송하는 단계
- Read할 데이터가 전송 시작되는 단계
- Read할 데이터가 전달 완료되는 단계

```
# The "wait_for_arvalid" state is used to detect the start
# of the read address phase of the AXI transaction which
# is indicated by the axi_arvalid signal equal to '1'
#
state wait_for_arvalid:
    if (design_1_i/system_ila_0/U0/net_slot_0_axi_arvalid == 1'b1) then
      goto wait_for_rready;
    else
      goto wait_for_arvalid;
    endif
#
# The "wait_for_rready" state is used to detect the start
# of the read data phase of the AXI transaction which
# is indicated by the axi_rready signal equal to '1'
#
state wait_for_rready:
  if (design_1_i/system_ila_0/U0/net_slot_0_axi_rready == 1'b1) then
    goto wait_for_rlast;
  else
    goto wait_for_rready;
  endif

#
# The "wait_for_rlast" state is used to detect the end
# of the read data phase of the AXI transaction which
# is indicated by the axi_rlast signal equal to '1'.
# Once the end of the data phase is detected, the ILA core
# will trigger.
#
state wait_for_rlast:
  if (design_1_i/system_ila_0/U0/net_slot_0_axi_rlast == 1'b1) then
    trigger;
  else
    goto wait_for_rlast;
  endif
```

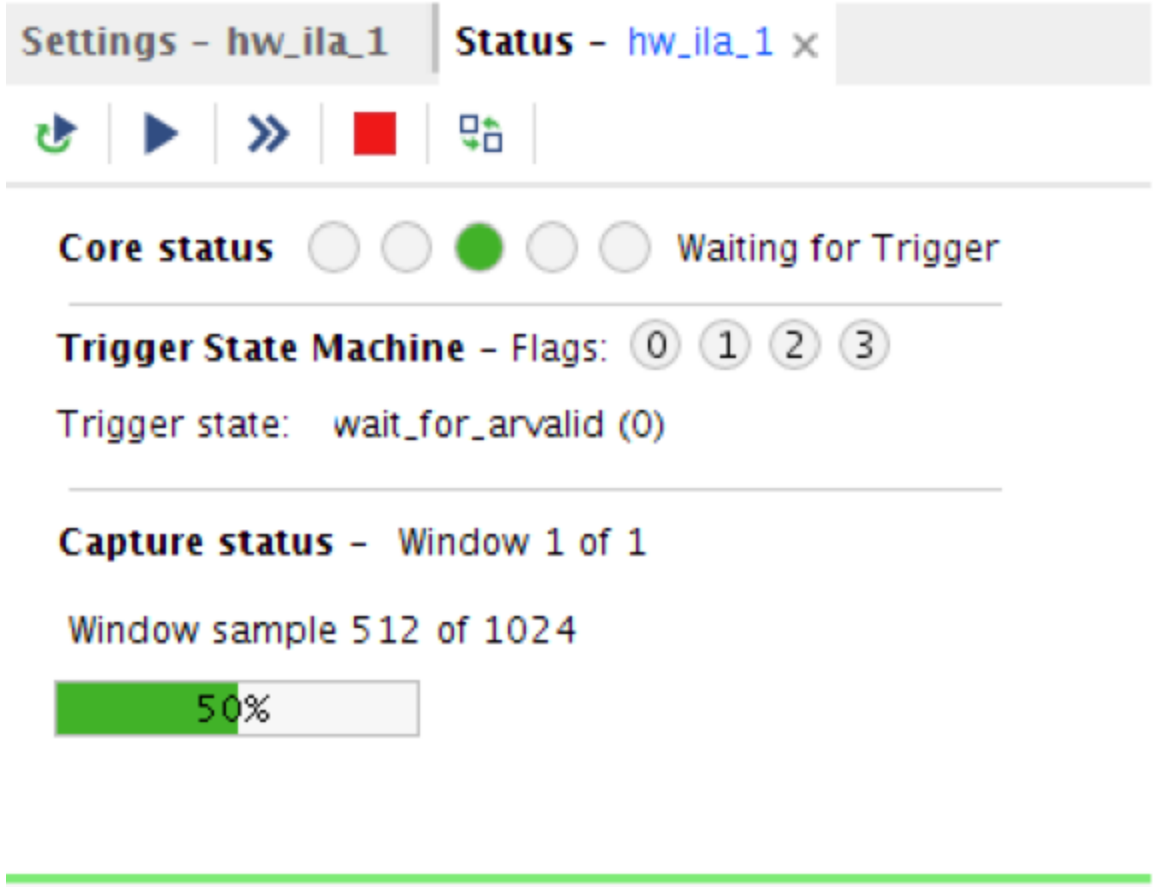
Step3: ILA Advanced trigger 기능을 사용해서 AXI가 Read할 수 있도록 하기

- Hw_lia_1 오른쪽 마우스 클릭 – Run trigger 클릭



Step3: ILA Advanced trigger 기능을 사용해서 AXI가 Read할 수 있도록 하기

- Trigger Capture Status 창에서,
  - ILA 코어는 트리거가 일어나기를 기다리게 됨(wait for a valid 상태)
  - 아직 vaild한 주소를 보내지 않았다는 뜻

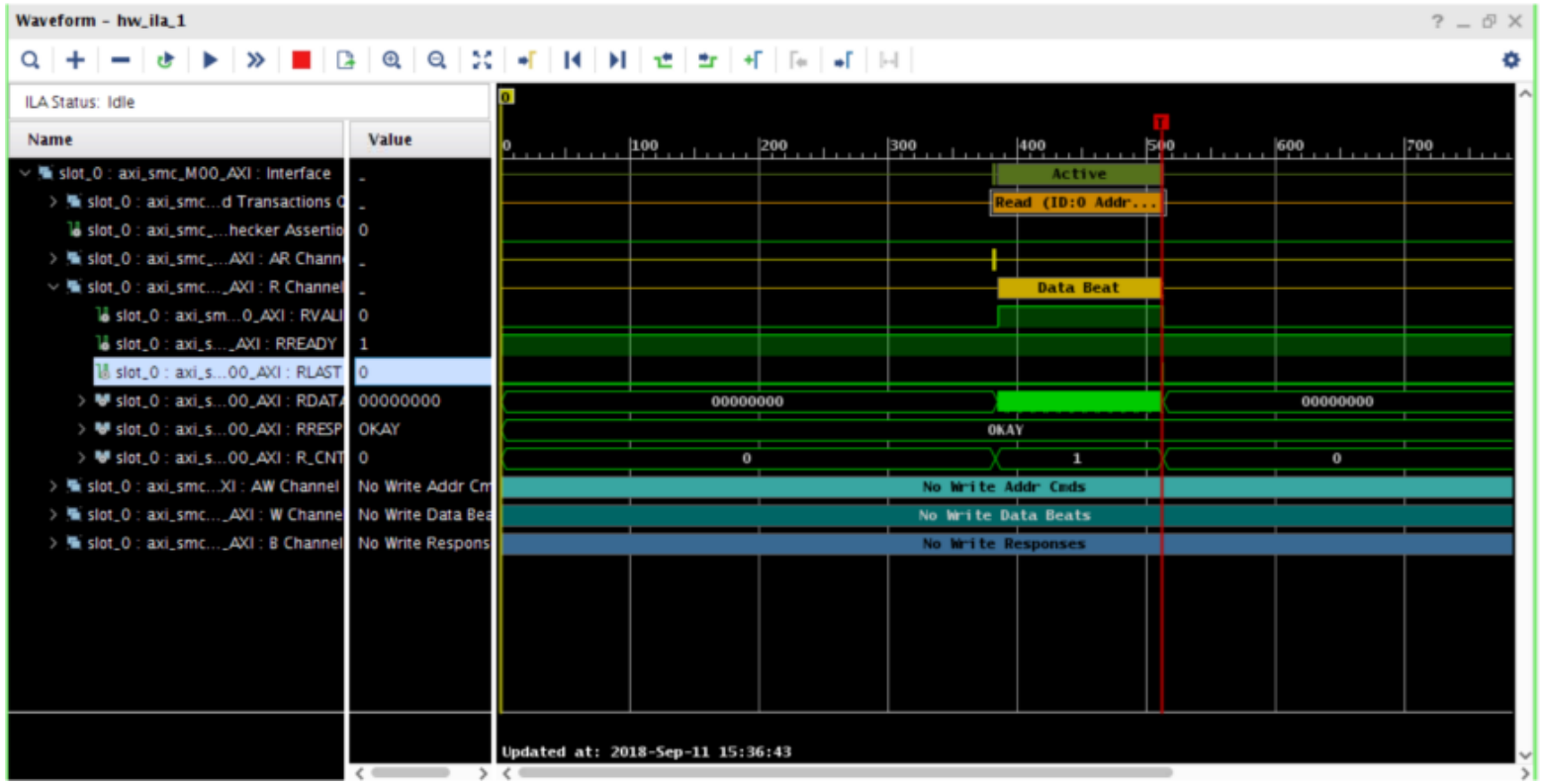


Step3: ILA Advanced trigger 기능을 사용해서 AXI가 Read할 수 있도록 하기

- Tcl 명령어를 통해, read transaction이 일어나게 함

```
run_hw_axi $rt
```

- Waveform을 통해 read 주소와 데이터가 전송되는 것을 관찰
- **데이터 read가 끝남 -> Axi_rlast 신호가 1 -> trigger 발생**



감사합니다.

- Q&A