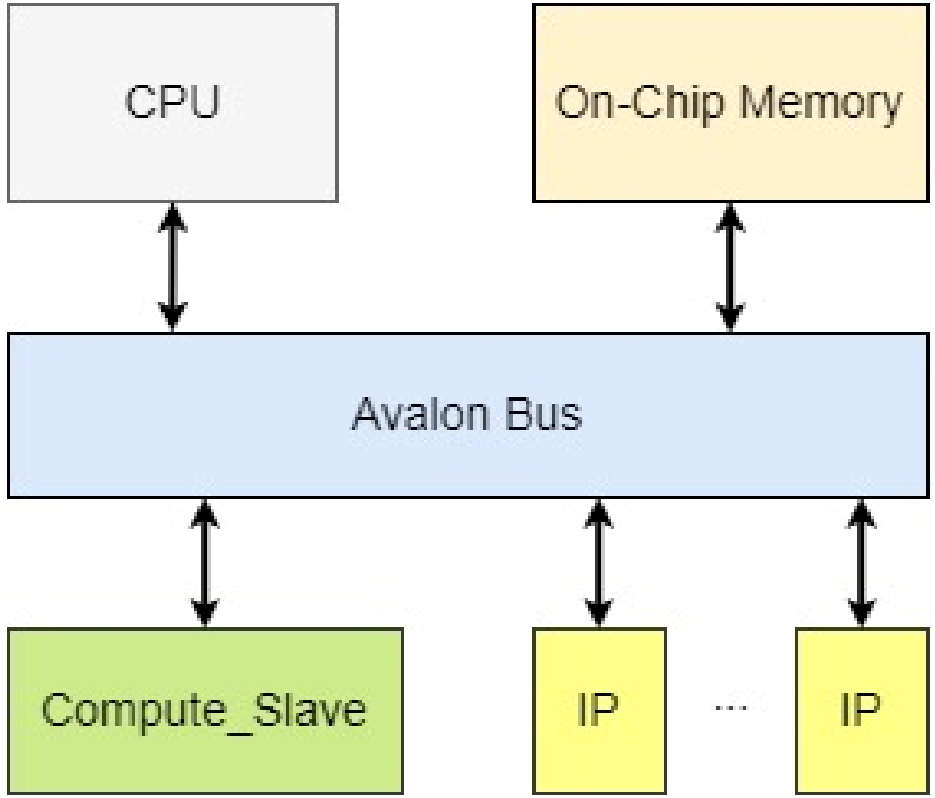
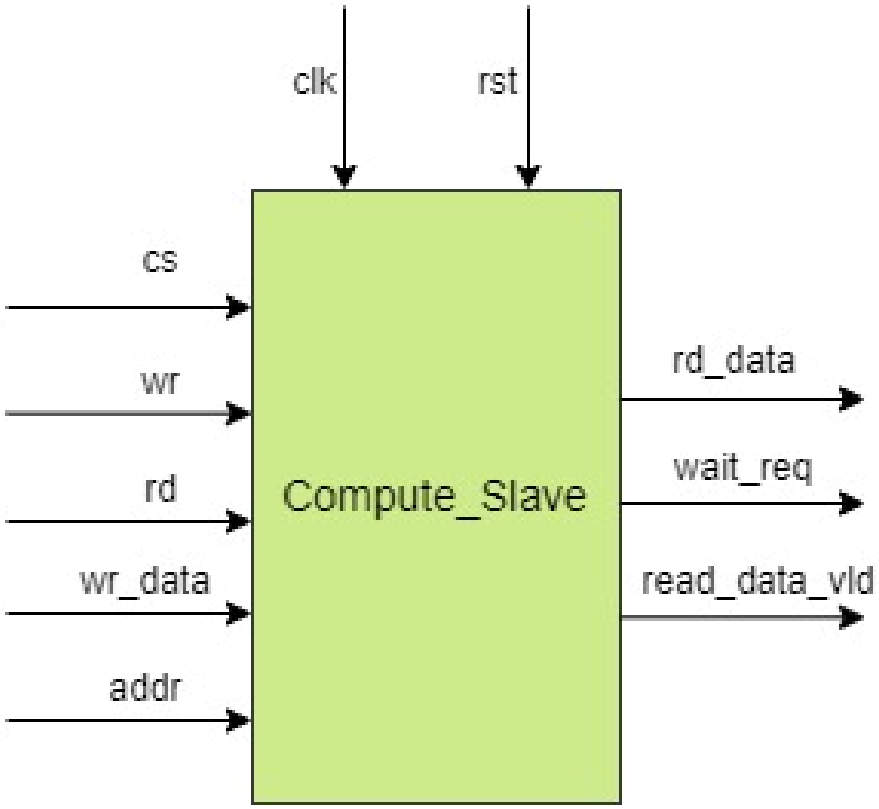
Design Slave with Avalon Interface

Kiet Tran

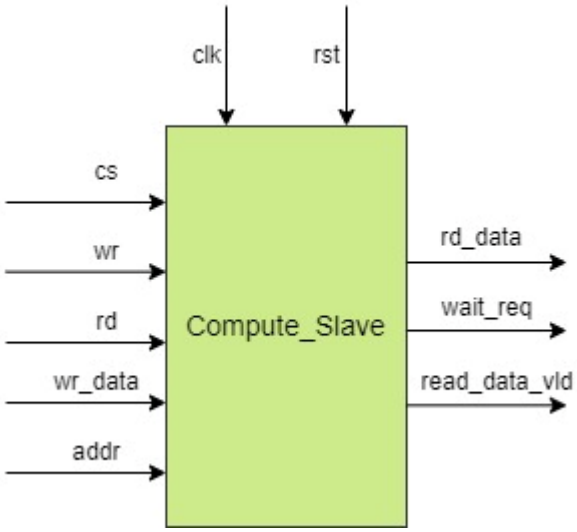
07/2023

System Diagram



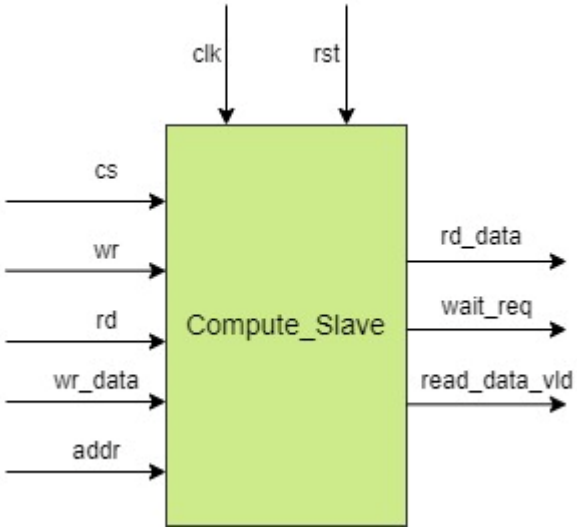


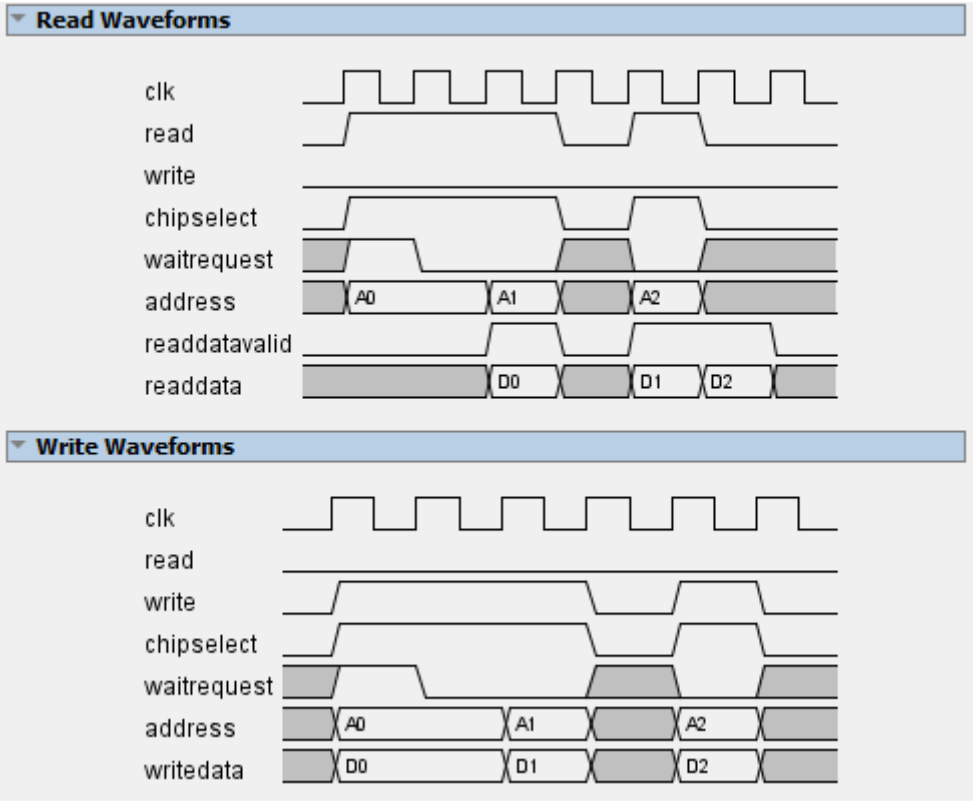
Compute top signal



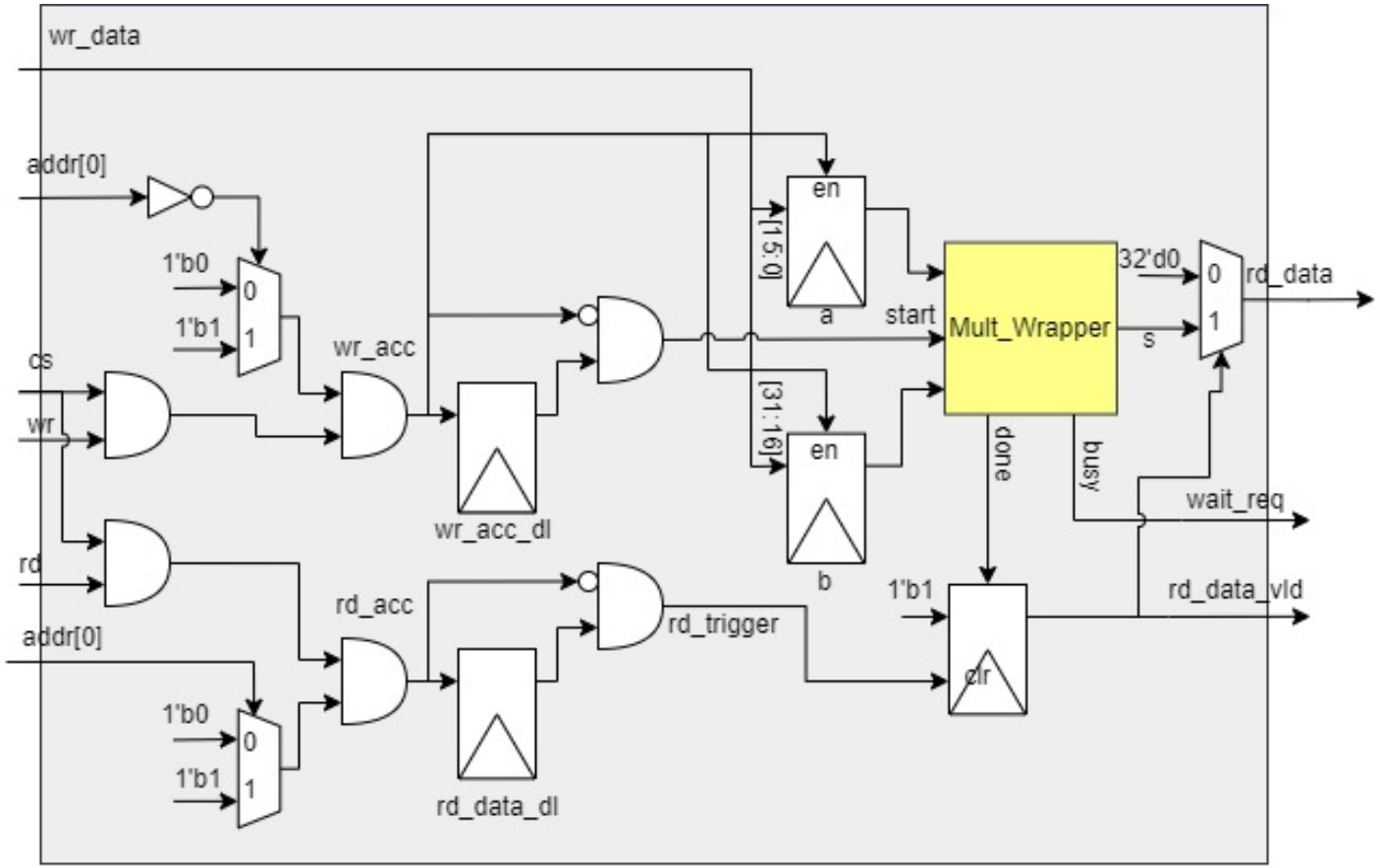
Signal	Width	Direction	Description
clk	1	Input	Synchronous clock signal
Rst	1	Input	Asynchronous reset signal
cs	1	Input	Chip select signal
wr	1	Input	Write active signal
rd	1	Input	Read active signal
addr	2	Input	Address
wr_data	32	Input	Data for write operation
rd_data	32	Output	Data for read operation
wait_req	1	Output	Wait request indicate hold bus
read_data_vld	1	Output	Read data valid indicate data is valid

Compute example transfer





Specification



Offset	Register	Access	Descrition
0	Data write	W	a[15:0] b[31:16]
1	Data read	R	s[31:0]

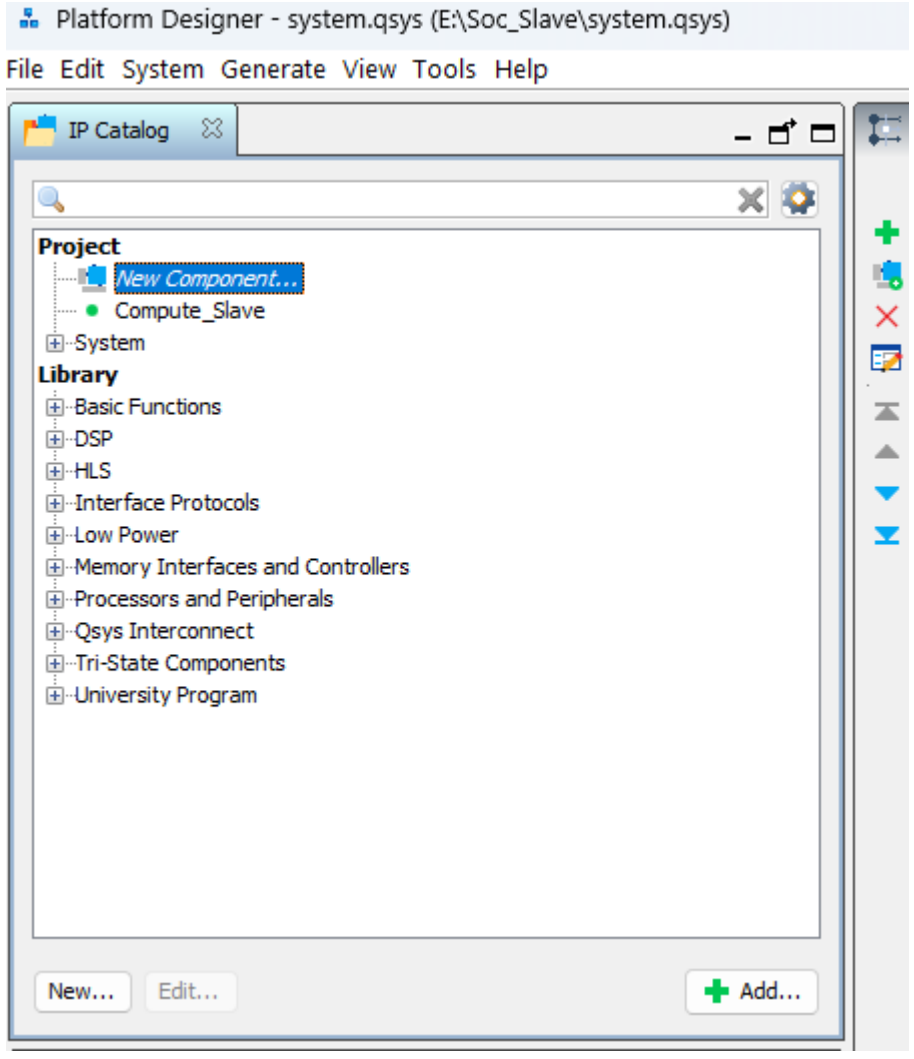
- Mult_Wrapper: Multiplier 16 bit and takes 4 cycles to generate results ***s***.
- ***done*** -> Compute complete, ***busy*** -> computing
- When write data into “Data write” register will trigger Mult Wrapper to start computing.
- When Mult_Wrapper is operating, ***wait_req*** is assert.
- When compute complete, ***read_data_vld*** is assert and clear when read done form master.

Lab: Source and Create Quartus project

- Download Source code of Compute_Slave: https://1drv.ms/f/s!Am4cUdrkdaQhgaADrb6vQXvLo5L47Q?e=AbbiQs
- Create project in Quartus with device Cyclone V with device is: 5CSXFC6D6F31C6

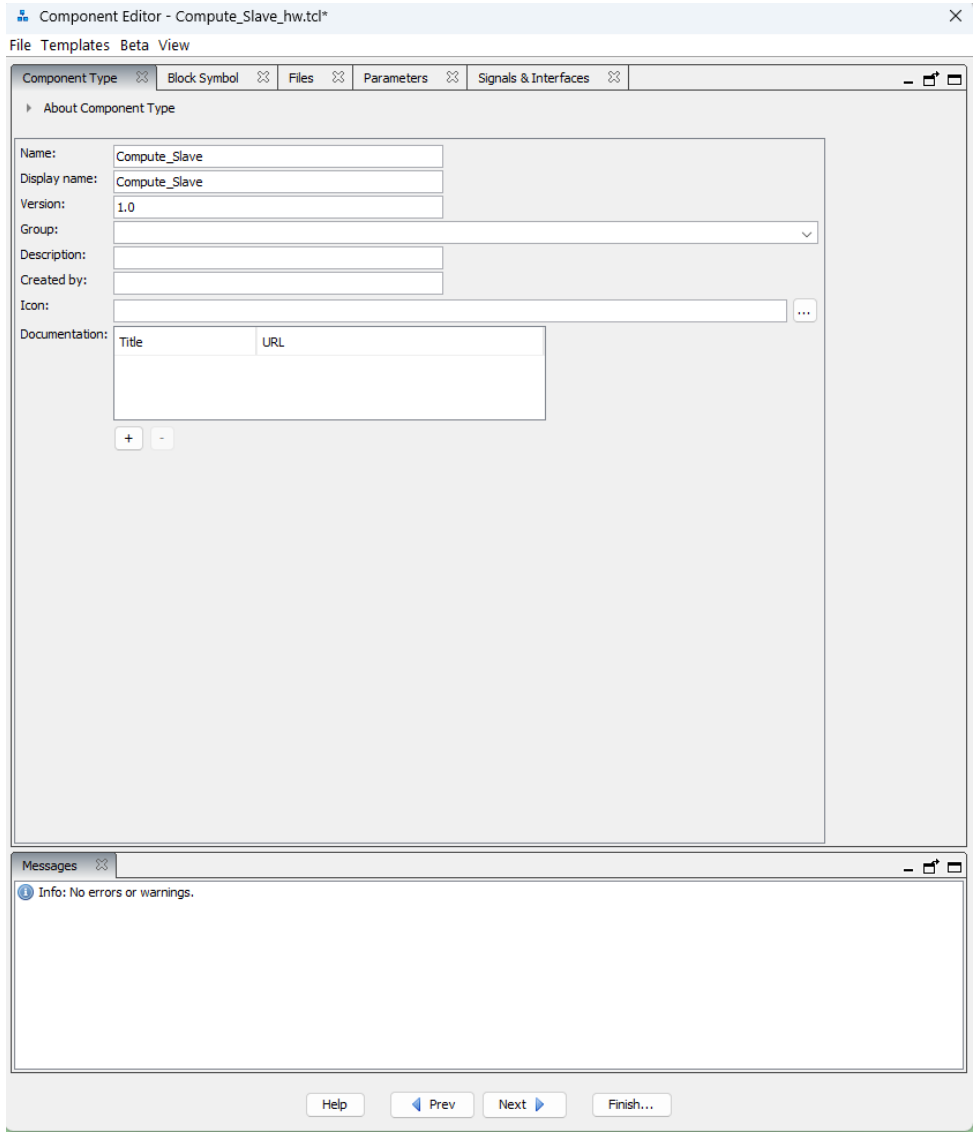
Lab: Create new component in Flatform Designer

- Open Flatform Designer, add new component



Lab: Create new component in Flatform Designer

- Add name: Computer_Slave



Lab: Create new component in Flatform Designer

- Add all file:

Files for synthesis

Files for simulation

Component Editor - Compute_Slave_hw.tcl*

File Templates Beta View

Component Type | Block Symbol | Files | Parameters | Signals & Interfaces

About Files

Synthesis Files

These files describe this component's implementation, and will be created when a Quartus synthesis model is generated.

The parameters and signals found in the top-level module will be used for this component's parameters and signals.

Output Path	Source File	Type	Attributes
Compute_Slave.sv	Compute_Slave.sv	**System Verilog HDL**	**Top-level File**
Mult.v	Mult.v	**Verilog HDL**	*no attributes*
Mult_Wrapper.sv	Mult_Wrapper.sv	**System Verilog HDL**	*no attributes*

Add File... | Remove File | Analyze Synthesis Files | Create Synthesis File from Signals

Top-level Module: (Analyze files to select module)

Verilog Simulation Files

These files will be produced when a Verilog simulation model is generated.

Output Path	Source File	Type	Attributes
Compute_Slave.sv	Compute_Slave.sv	**System Verilog HDL**	
Mult.v	Mult.v	**Verilog HDL**	*no attributes*
Mult_Wrapper.sv	Mult_Wrapper.sv	**System Verilog HDL**	*no attributes*

Add File... | Remove File | Copy from Synthesis Files

VHDL Simulation Files

These files will be produced when a VHDL simulation model is generated.

Output Path	Source File	Type	Attributes
(No files)			

Add File... | Remove File | Copy from Synthesis Files

Messages

Info: No errors or warnings.

Lab: Create new component in Flatform Designer

- Config signals and interfaces

Component Editor - Compute_Slave_hw.tcl*

File Templates Beta View

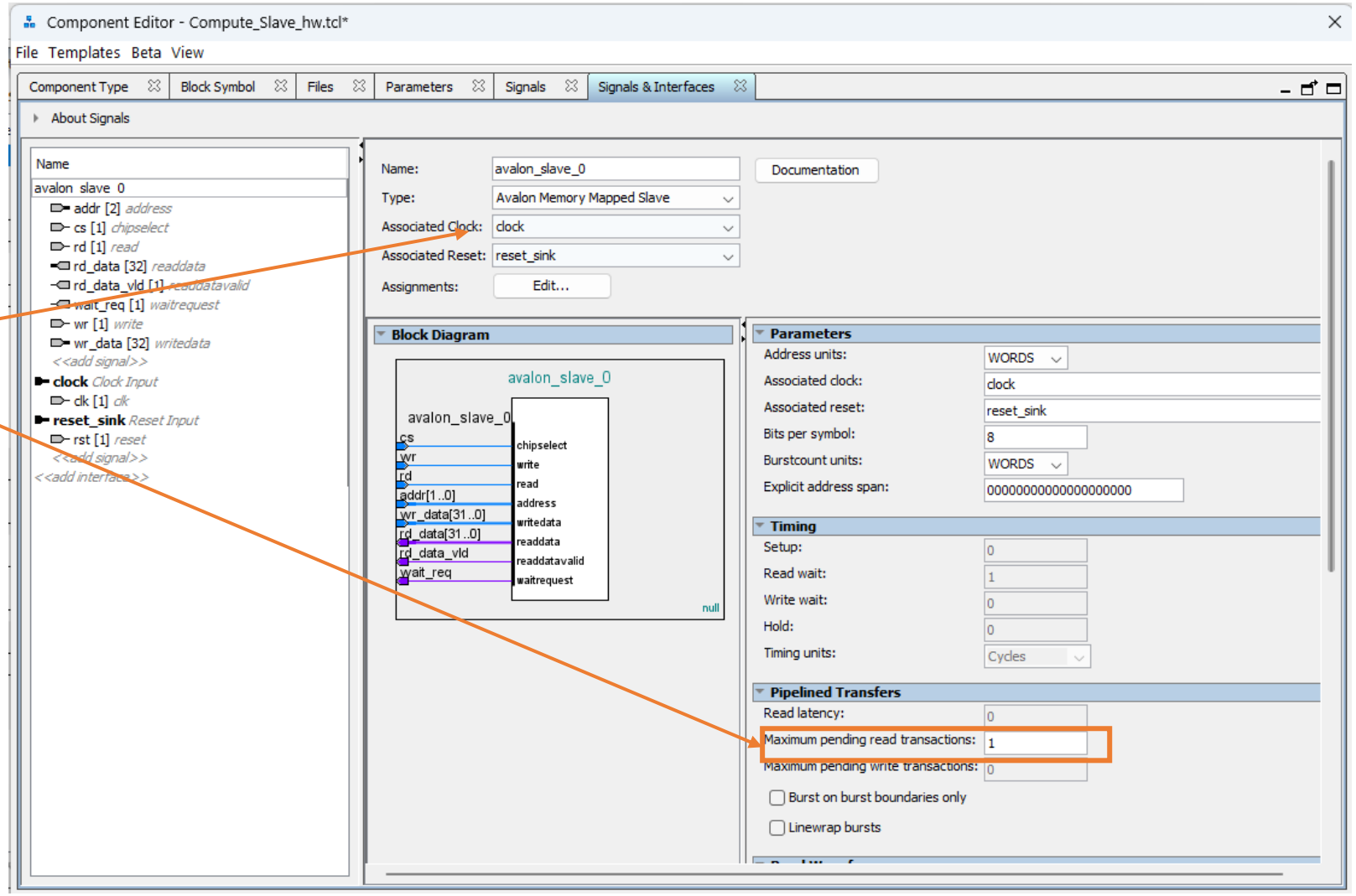
Component Type | Block Symbol | Files | Parameters | Signals | Signals & Interfaces

About Signals

Name	Interface	Signal Type	Width	Direction
clk	clock	clk	1	input
cs	avalon_slave_0	chipselect	1	input
wr	avalon_slave_0	write	1	input
rd	avalon_slave_0	read	1	input
addr	avalon_slave_0	address	2	input
wr_data	avalon_slave_0	writedata	32	input
rd_data	avalon_slave_0	readdata	32	output
rd_data_vld	avalon_slave_0	readdatavalid	1	output
wait_req	avalon_slave_0	waitrequest	1	output
rst	reset_sink	reset	1	input

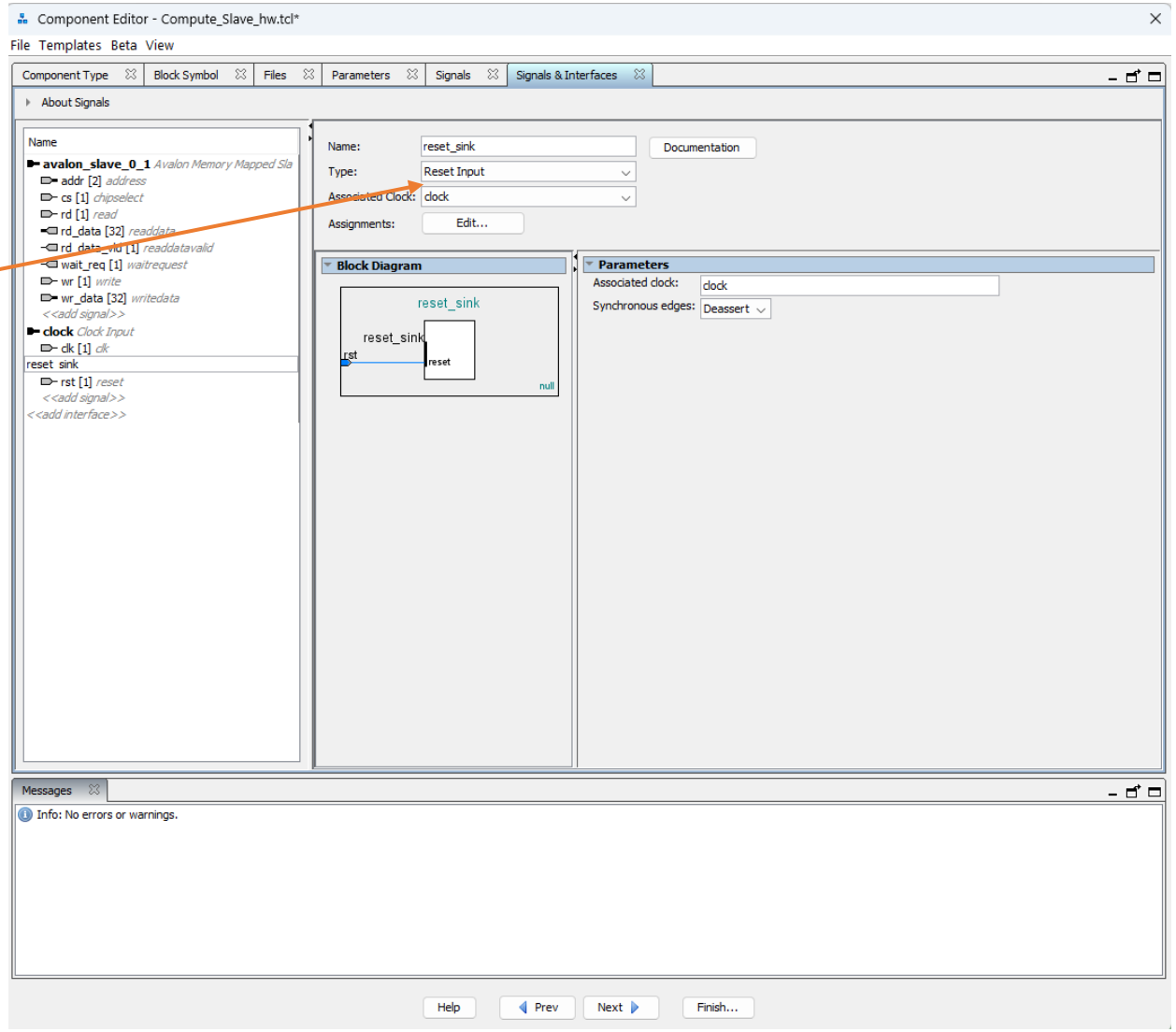
Lab: Create new component in Flatform Designer

- Config clock, number of pipeline transfer



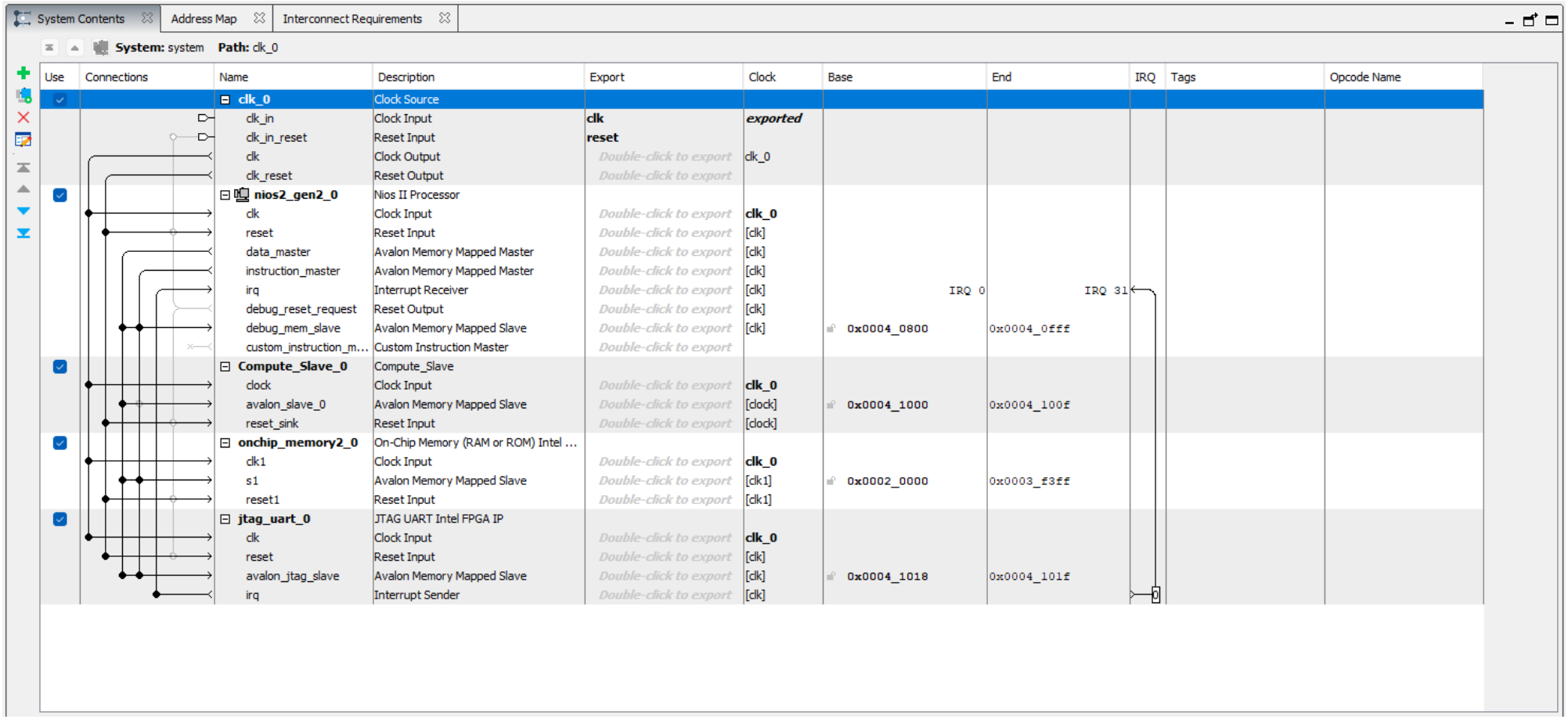
Lab: Create new component in Flatform Designer

- Config clock, number of pipeline transfer.
- No error -> Finish

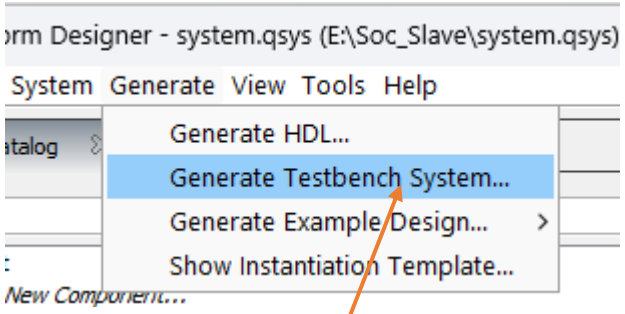


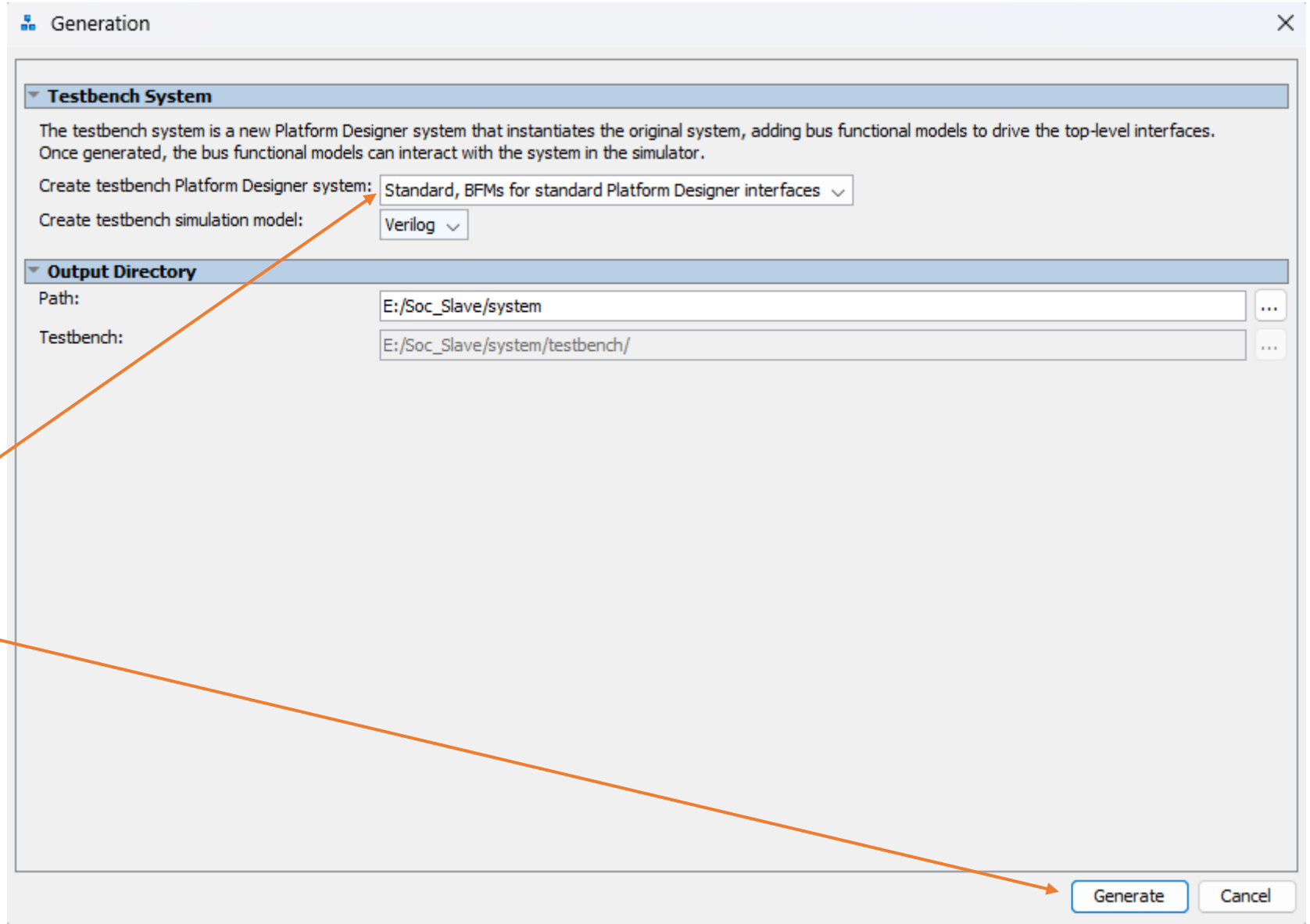
Lab: System

- Make a full system -> Generate



Lab: Generate testbench

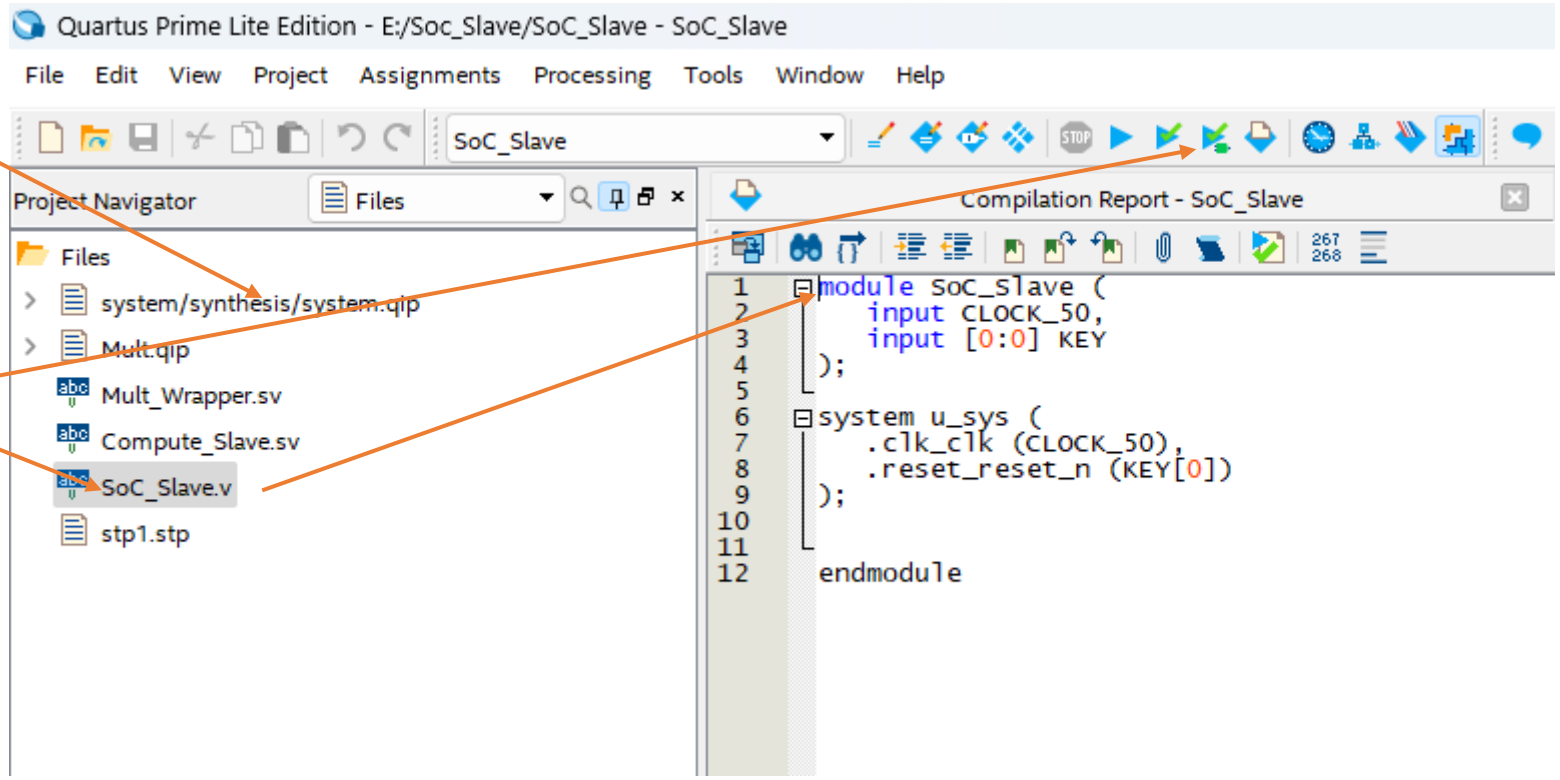




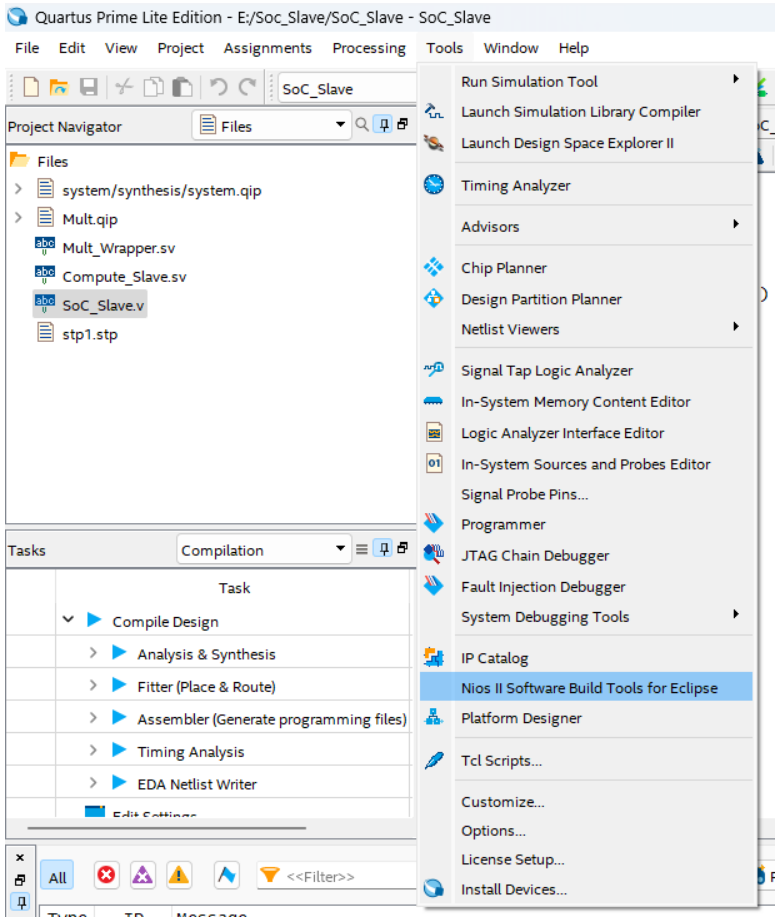
Chose for generate testbench system

Lab: Integrate

- Add sub-system top file.
- Top file
- Pre-compile

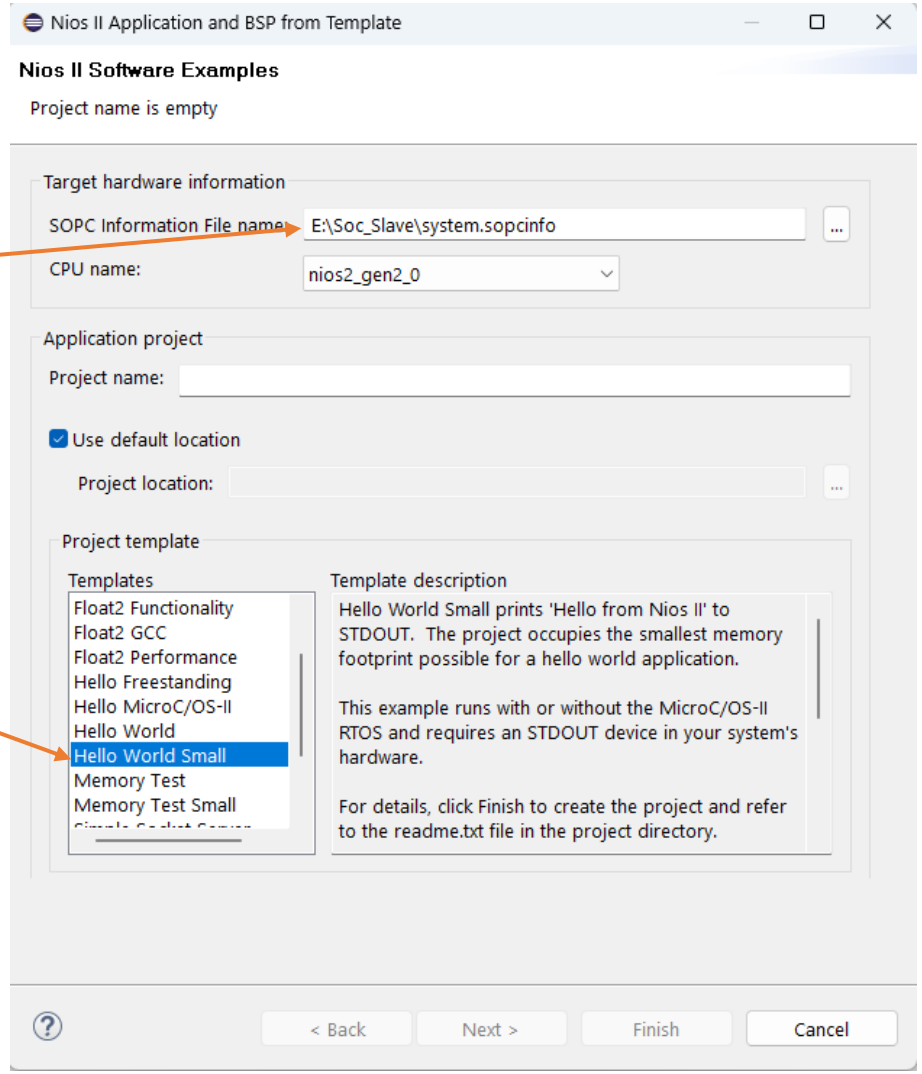


Lab: Create software



Lab: Create software

- Add project with hardawares information
- Example project



Lab: Create software

Nios II - Eclipse

File Edit Source Refactor Navigate Search Project Run Nios II Window Help

Project Explorer

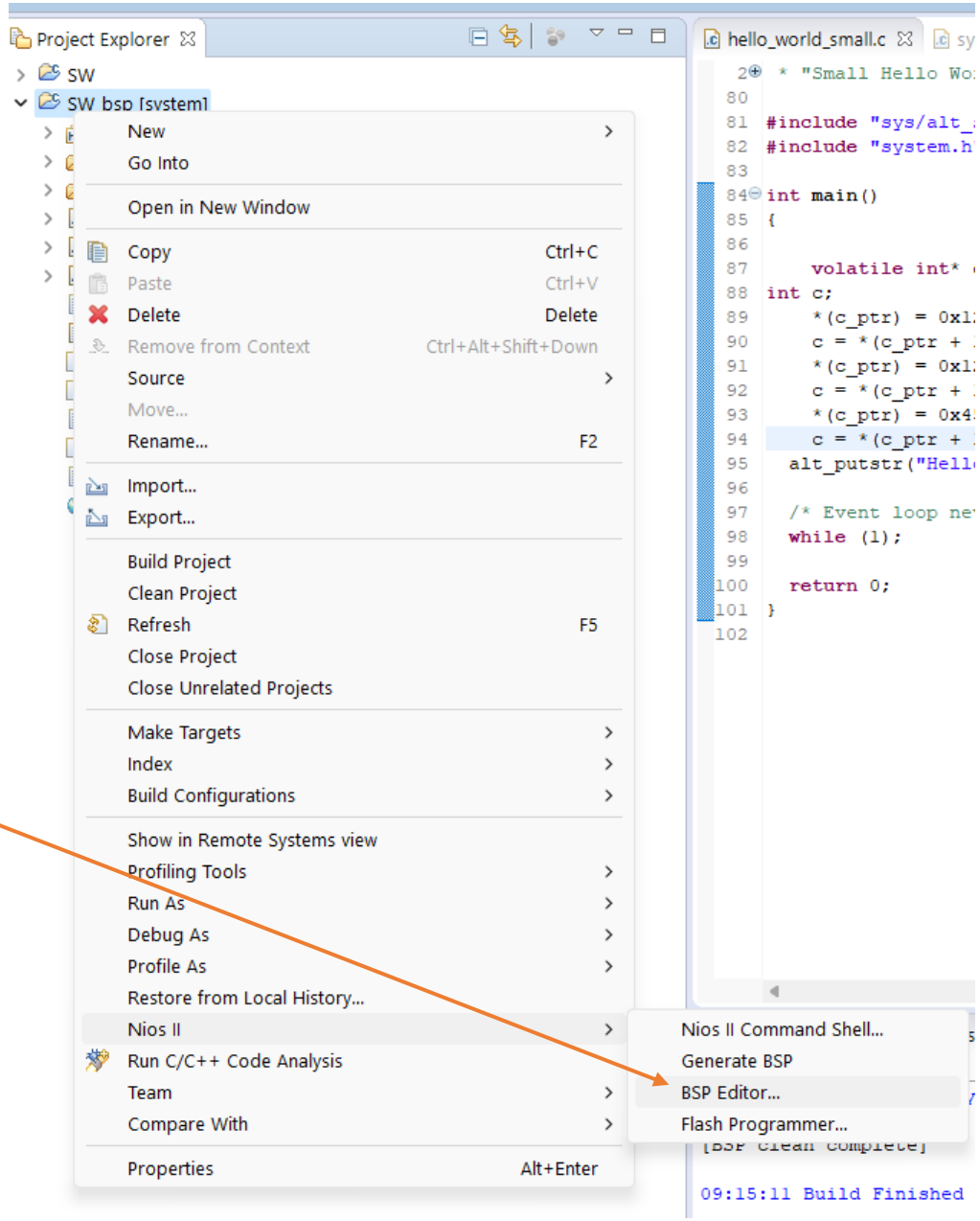
- SW
  - Binaries
  - Includes
  - obj
  - system
  - hello_world_small.c
  - SW.elf - [alteranios2/le]
  - create-this-app
  - Makefile
  - readme.txt
  - SW.map
  - SW.objdump
- SW_bsp [system]

hello_world_small.c | system.h

```c
 * "Small Hello World" example.

#include "sys/alt_stdio.h"
#include "system.h"

int main()
{

    volatile int* c_ptr = (int*)COMPUTE_SLAVE_0_BASE;
int c;
    *(c_ptr) = 0x1245678;
    c = *(c_ptr + 1);
    *(c_ptr) = 0x1241234;
    c = *(c_ptr + 1);
    *(c_ptr) = 0x4567678;
    c = *(c_ptr + 1);
  alt_putstr("Hello from Nios II!\n");

  /* Event loop never exits. */
  while (1);

  return 0;
}
```

Lab: Edit BSP

Open BSP editor to configure the reduce sim



Lab: Configure the BSP

BSP Editor - settings.bsp

File Edit Tools Help

Main | Software Packages | Drivers | Linker Script | Enable File Generation | Target BSP Directory

SOPC Information file: E:\SoC_Slave\system.sopcinfo
CPU name: nios2_gen2_0
Operating system: Altera HAL
Version: default
BSP target directory: E:\Soc_Slave\software\SW_bsp

- Settings
  - Common
    - hal
      - sys_clk_timer
      - timestamp_timer
      - stdin
      - stdout
      - stderr
      - enable_small_c_library
      - enable_gprof
      - enable_reduced_device_drivers
      - enable_sim_optimize
      - linker
        - enable_exception_stack
        - exception_stack_size
        - exception_stack_memory_region_r
        - enable_interrupt_stack
        - interrupt_stack_size
        - interrupt_stack_memory_region_na

hal

sys_clk_timer: none
timestamp_timer: none
stdin: jtag_uart_0
stdout: jtag_uart_0
stderr: jtag_uart_0
[x] enable_small_c_library
[ ] enable_gprof
[x] enable_reduced_device_drivers
[x] enable_sim_optimize

hal.linker

[ ] enable_exception_stack

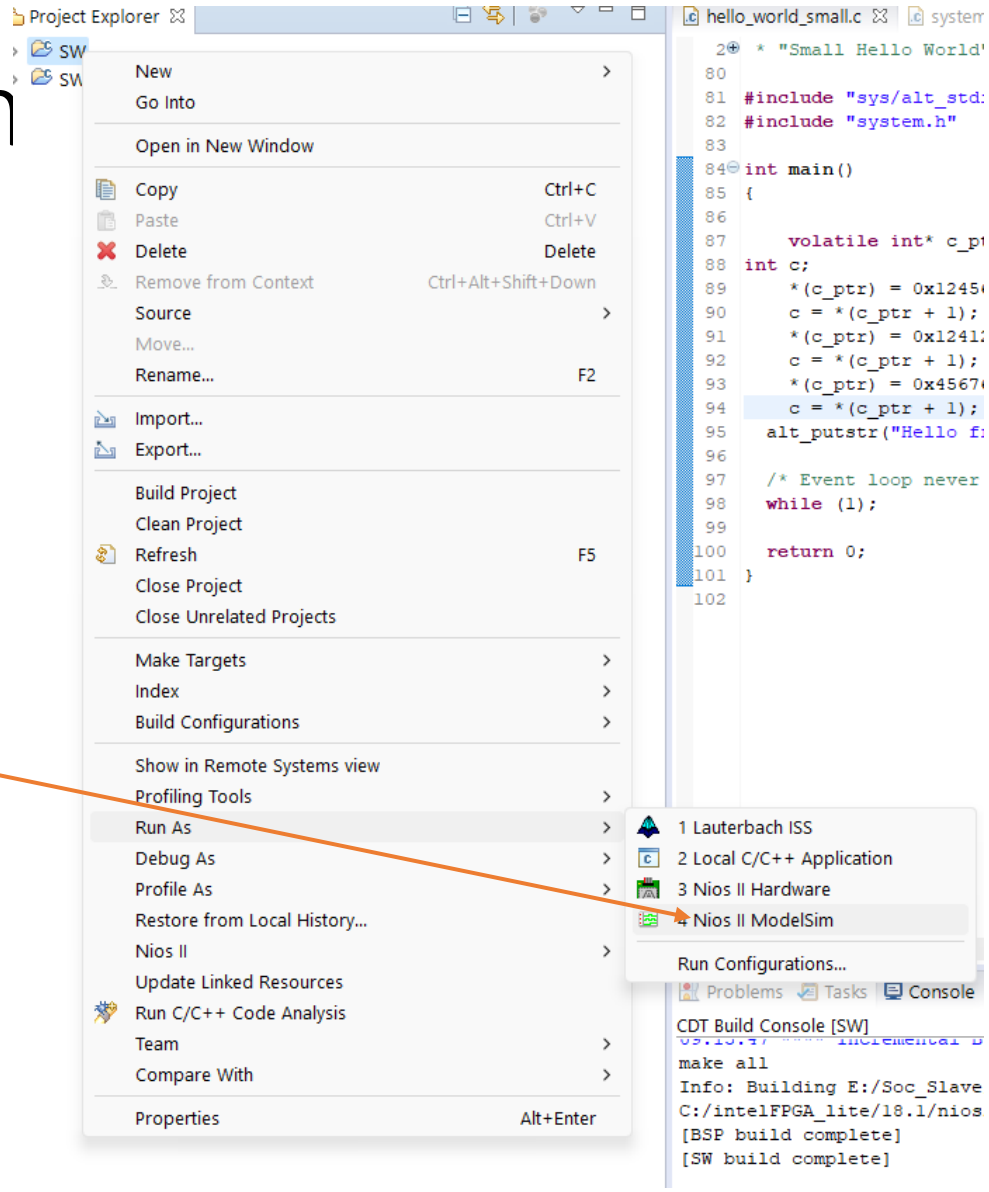
Information | Problems | Processing

- Finished loading BSP section mappings from settings file.
- Setting "hal.linker.interrupt_stack_memory_region_name" set to "onchip_memory2_0".
- Setting "hal.linker.exception_stack_memory_region_name" set to "onchip_memory2_0".
- Loading drivers from ensemble report.
- Mapped module: "nios2_gen2_0" to use the default driver version.
- Mapped module: "jtag_uart_0" to use the default driver version.
- Finished loading drivers from ensemble report.
- Loading BSP settings from settings file.
- Finished loading SOPC Builder system info file "E:\SoC_Slave\system.sopcinfo"

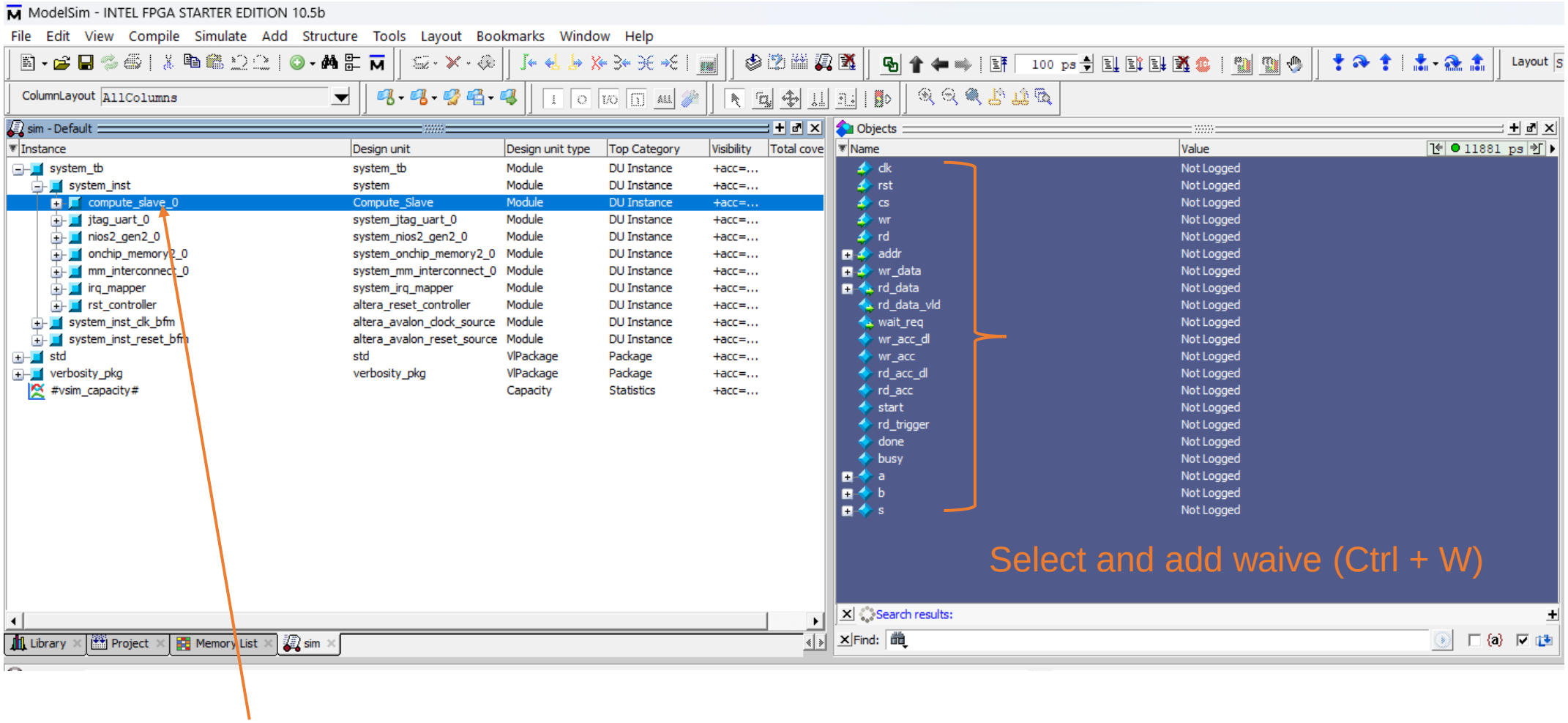
Generate | Exit

Lab: Simulation systen

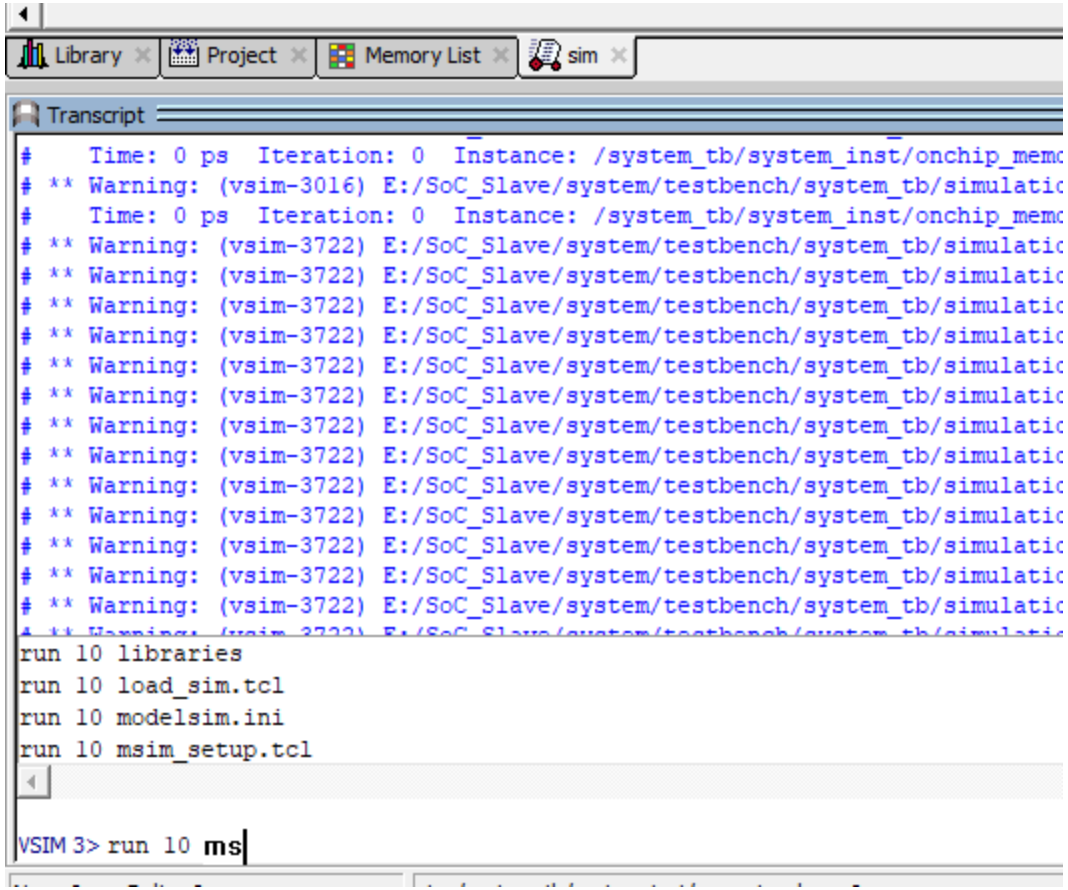
Run as Nios II Modelsim



Lab: Simulation



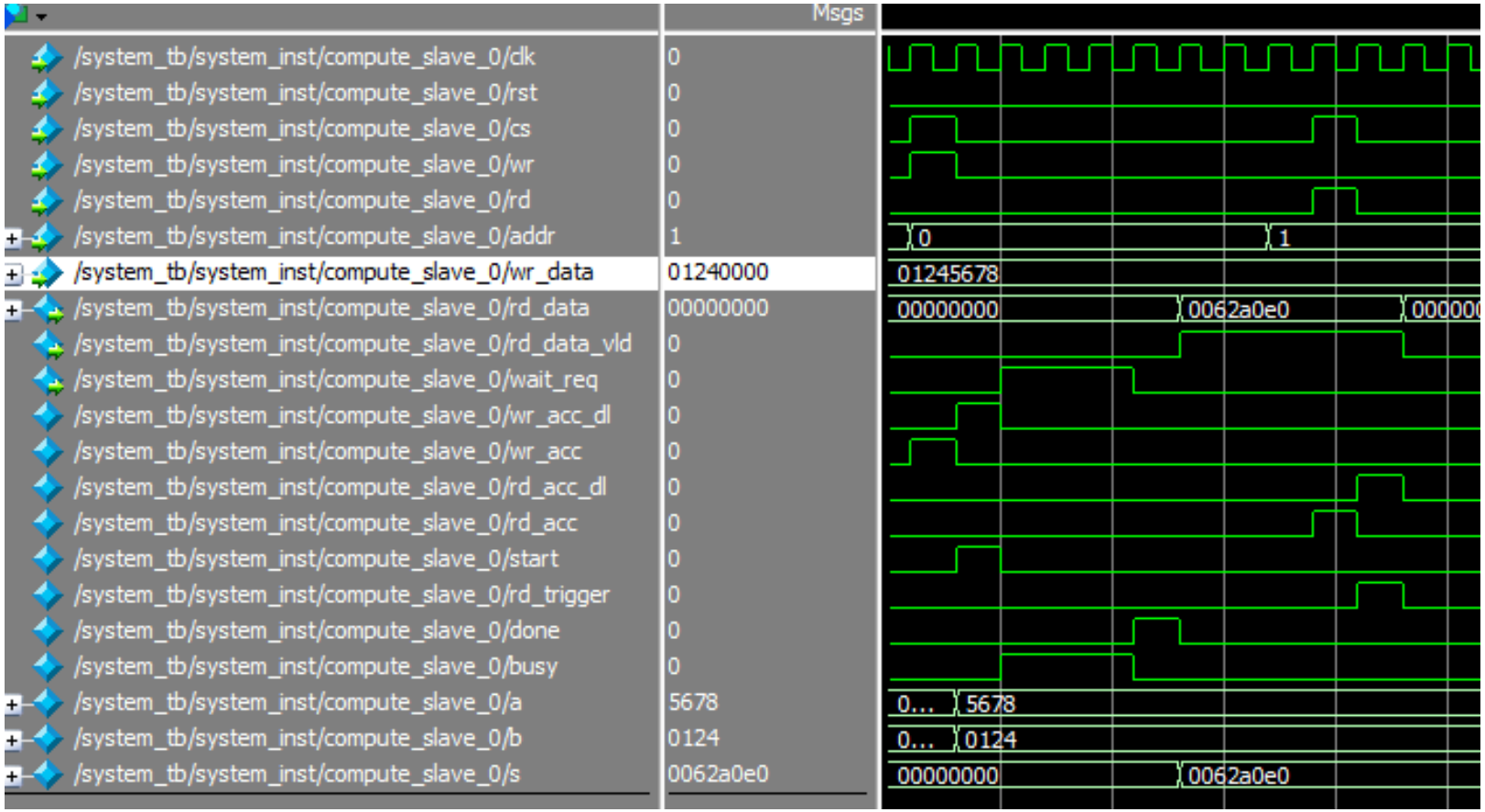
Lab: Simulation



```
#    Time: 0 ps  Iteration: 0  Instance: /system_tb/system_inst/onchip_memo
# ** Warning: (vsim-3016) E:/SoC_Slave/system/testbench/system_tb/simulatio
#    Time: 0 ps  Iteration: 0  Instance: /system_tb/system_inst/onchip_memo
# ** Warning: (vsim-3722) E:/SoC_Slave/system/testbench/system_tb/simulatio
# ** Warning: (vsim-3722) E:/SoC_Slave/system/testbench/system_tb/simulatio
# ** Warning: (vsim-3722) E:/SoC_Slave/system/testbench/system_tb/simulatio
# ** Warning: (vsim-3722) E:/SoC_Slave/system/testbench/system_tb/simulatio
# ** Warning: (vsim-3722) E:/SoC_Slave/system/testbench/system_tb/simulatio
# ** Warning: (vsim-3722) E:/SoC_Slave/system/testbench/system_tb/simulatio
# ** Warning: (vsim-3722) E:/SoC_Slave/system/testbench/system_tb/simulatio
# ** Warning: (vsim-3722) E:/SoC_Slave/system/testbench/system_tb/simulatio
# ** Warning: (vsim-3722) E:/SoC_Slave/system/testbench/system_tb/simulatio
# ** Warning: (vsim-3722) E:/SoC_Slave/system/testbench/system_tb/simulatio
# ** Warning: (vsim-3722) E:/SoC_Slave/system/testbench/system_tb/simulatio
# ** Warning: (vsim-3722) E:/SoC_Slave/system/testbench/system_tb/simulatio
# ** Warning: (vsim-3722) E:/SoC_Slave/system/testbench/system_tb/simulatio
run 10 libraries
run 10 load_sim.tcl
run 10 modelsim.ini
run 10 msim_setup.tcl

VSIM 3> run 10 ms
```

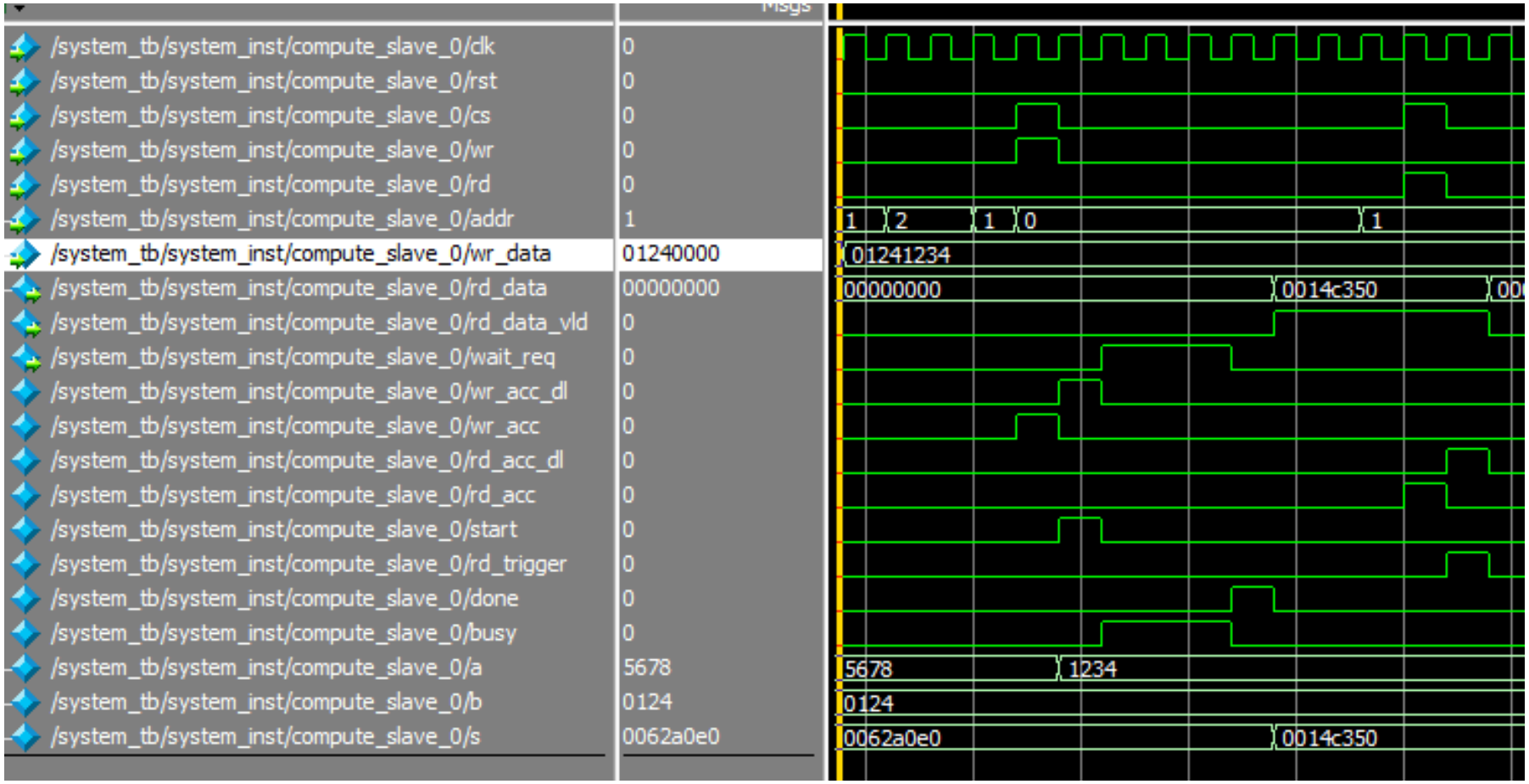
Run 10 ms and check the result

Lab: Sim results



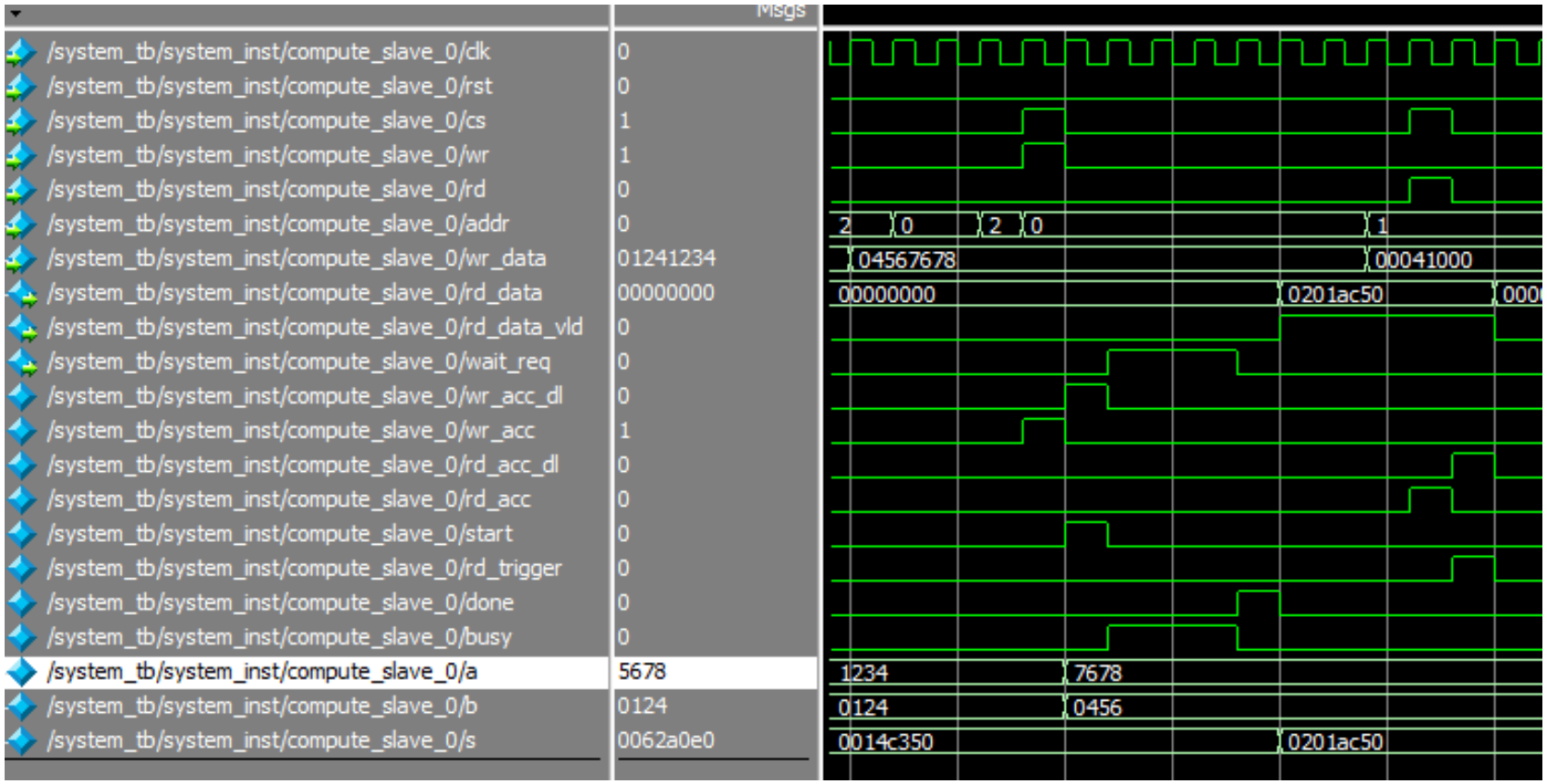
Result 1

Lab: Sim results



Result 2

Lab: Sim results

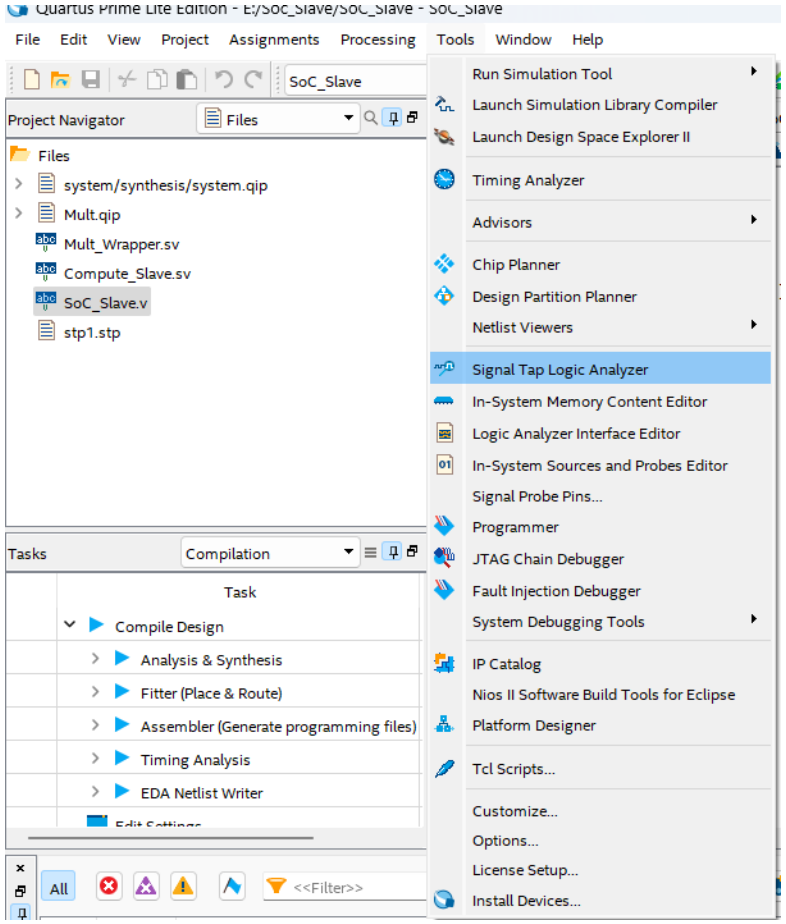


Result 3

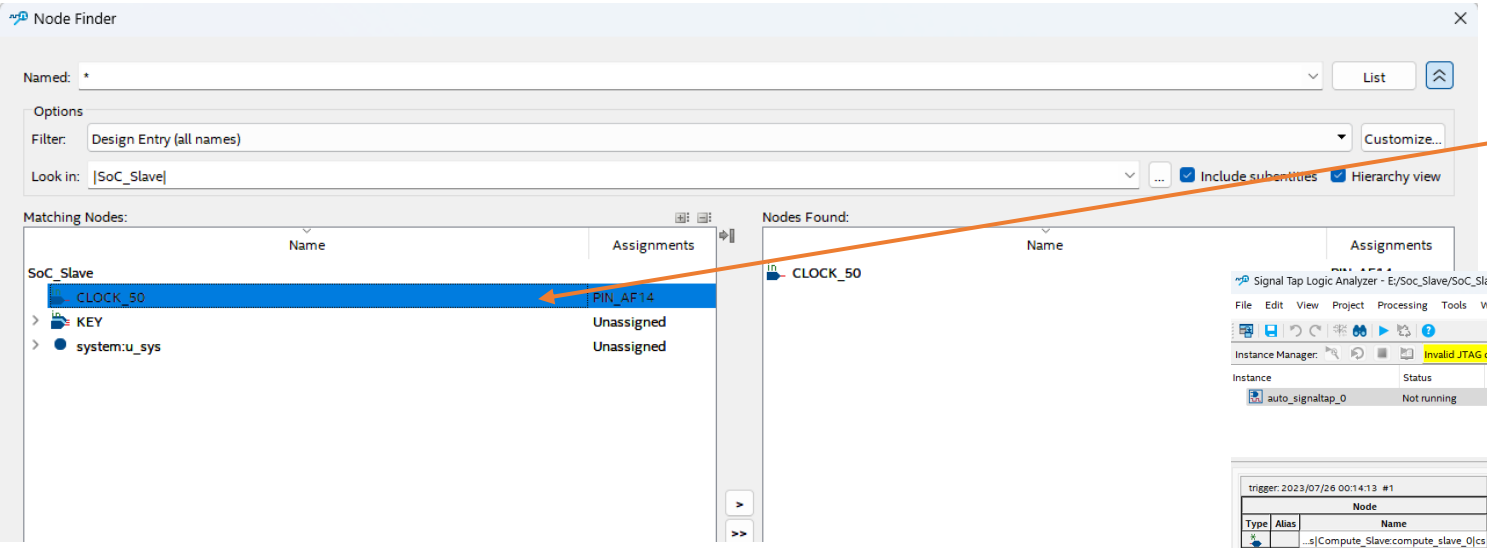
SignalTab

Lab: Signal Tap

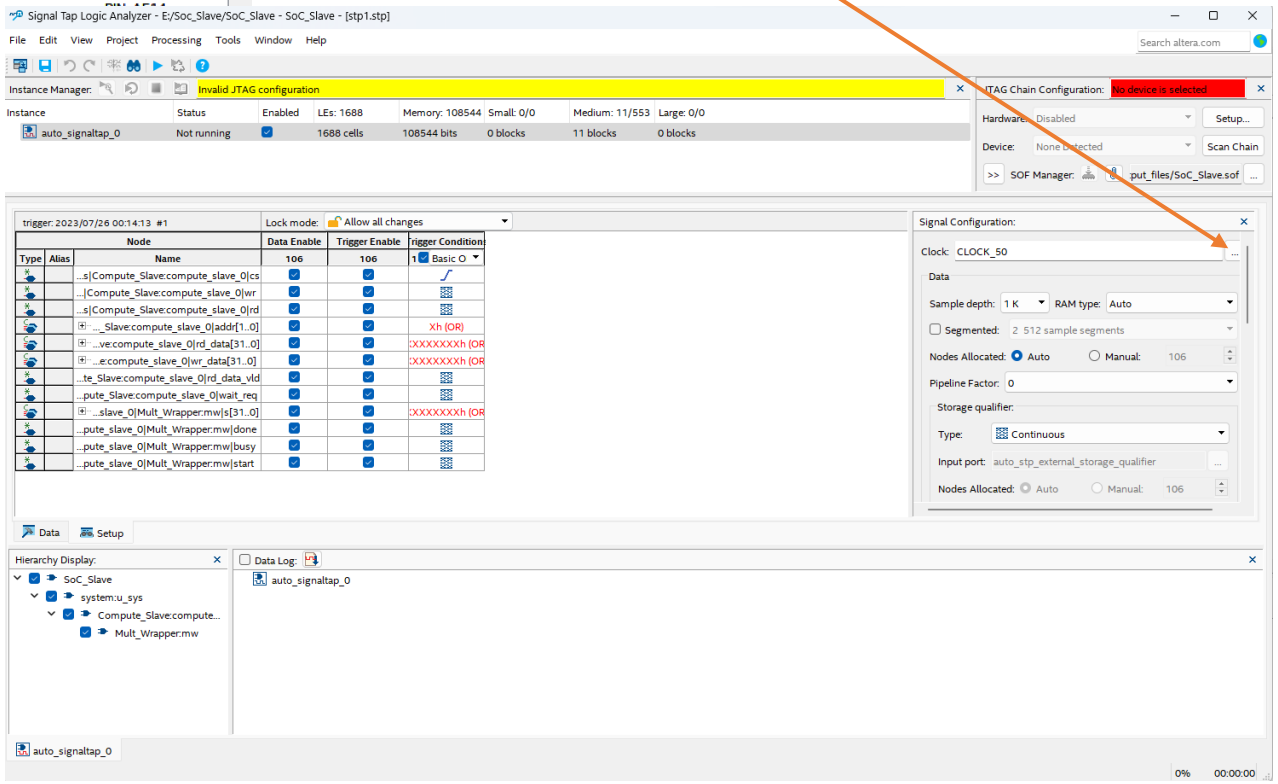
- Open signaltab



Lab: Signal Tap

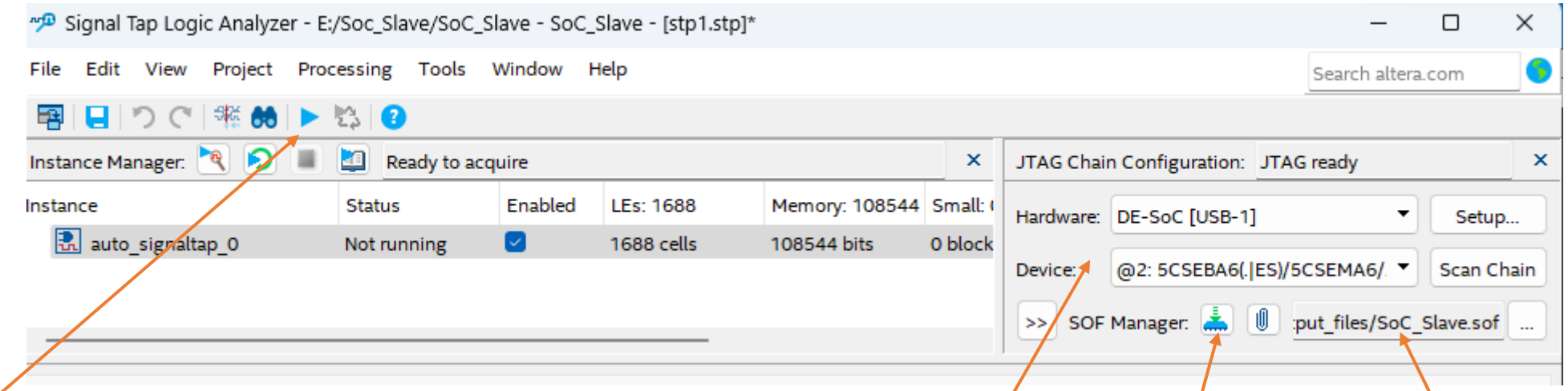


- Add clock



- Add signals

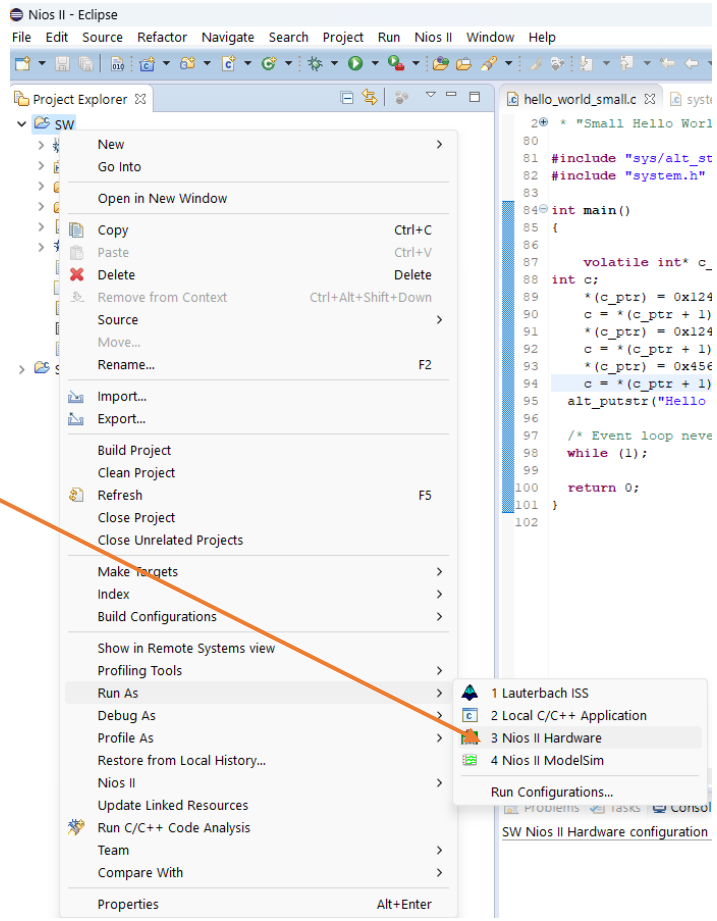
Lab: Compile hardware and program



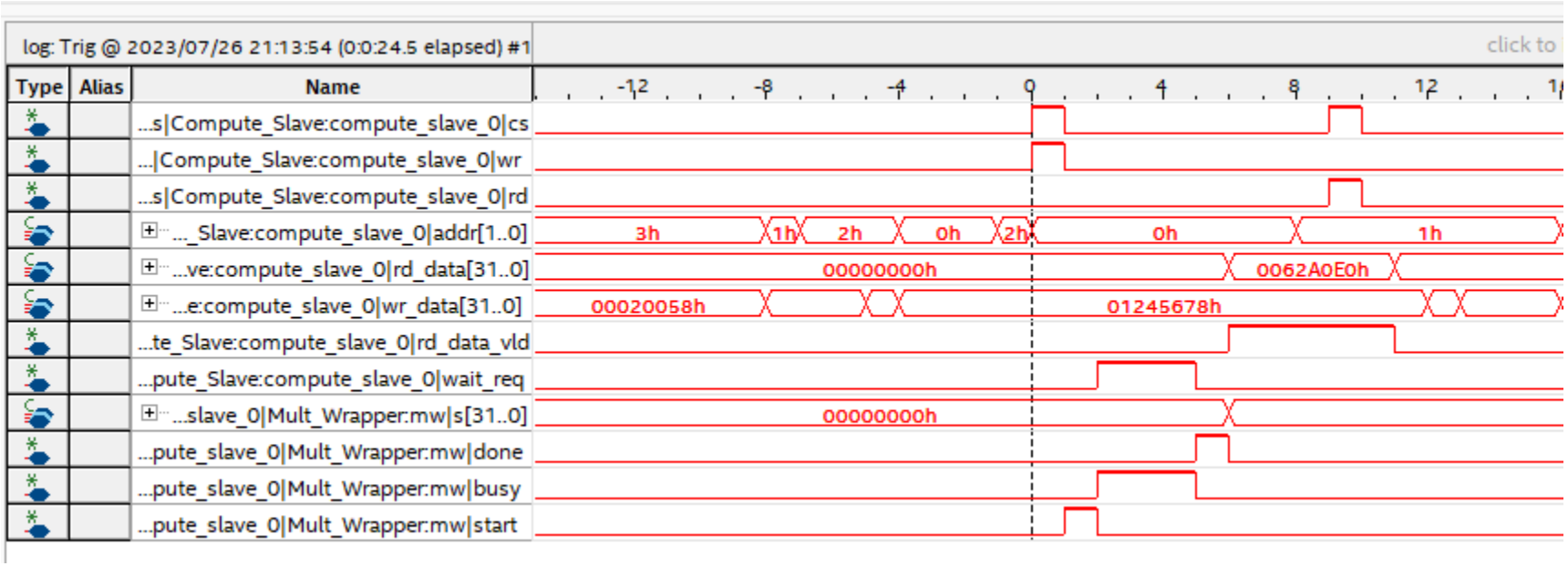
- Compile
- Config Jtag
- Program
- Hardware config stream file

Lab: Run software

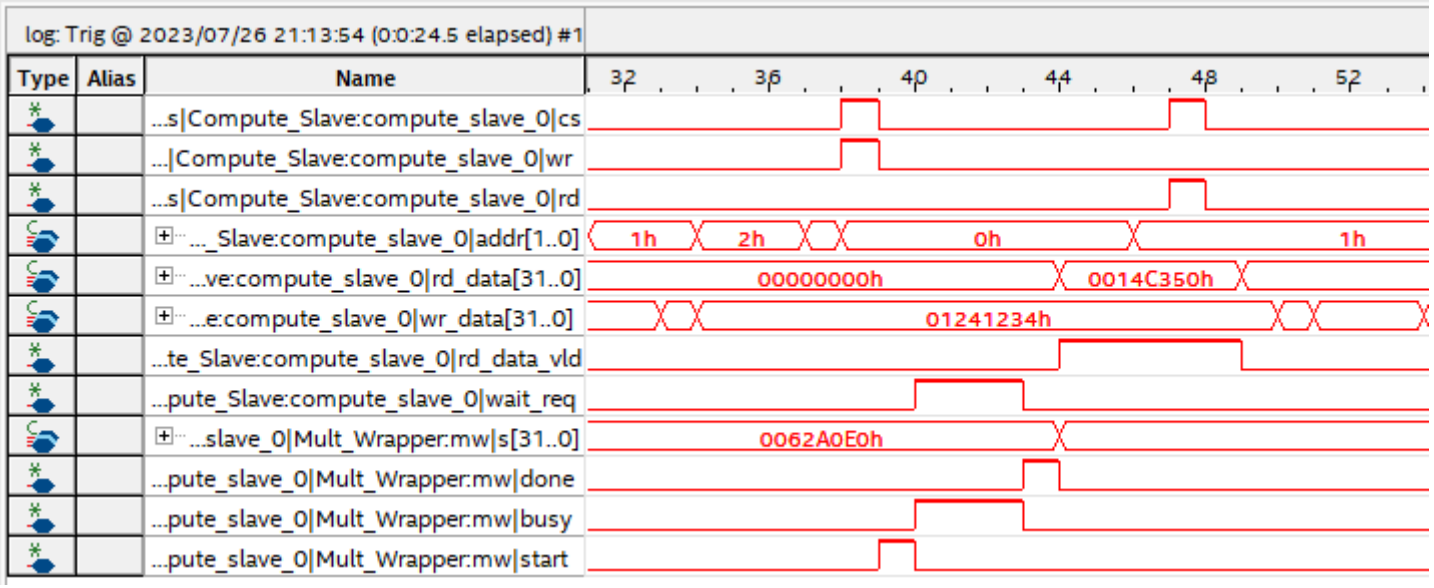
- Run project



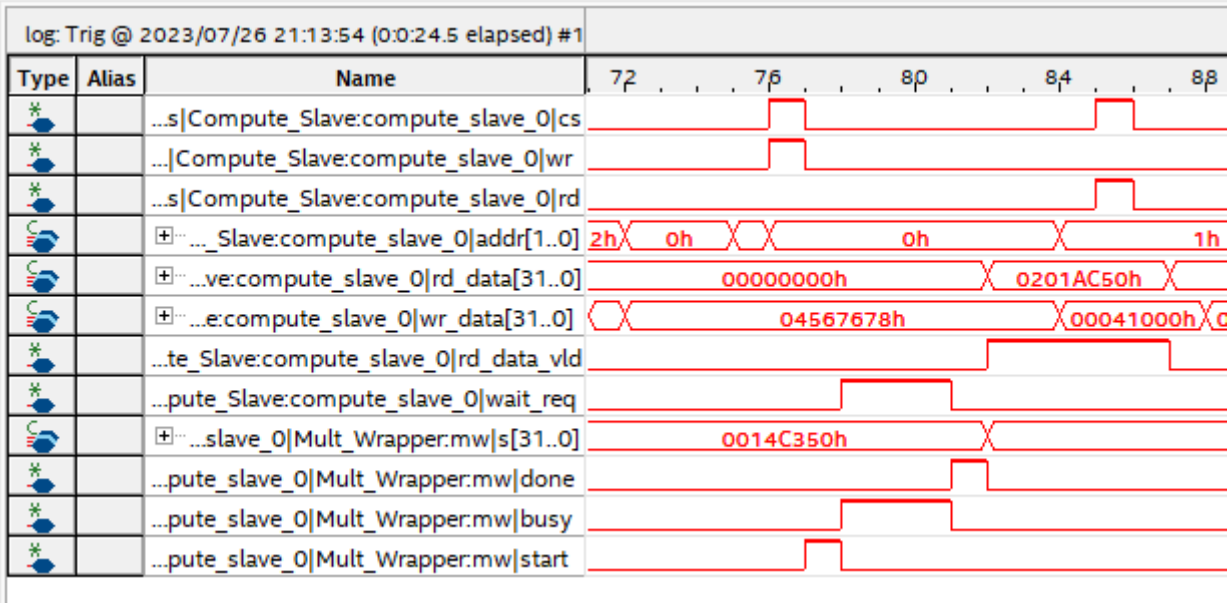
Lab: waveform



Lab: waveform



Lab: waveform



Assignment

TODO

- Use the current SoC system and create new software to make the matrix multiplication 4x4.

Assignment

- Assignment: Designing a Slave Device with Avalon Bus Interface

Objective:

The objective of this assignment is to design a slave device that communicates using the Avalon bus interface. The Avalon bus is a widely used industry-standard interface for system-on-chip (SoC) designs, commonly used in Intel FPGA devices. In this assignment, you will design a simple slave component that can communicate with a master device using the Avalon bus protocol.

Assignment

- Requirements:

1. Avalon Bus Specification: Familiarize yourself with the Avalon bus interface. Understand the various signals and protocols used in the Avalon bus specification. You can refer to the official Intel documentation for this purpose.

2. Slave Device Functionalities: The slave device should have the following functionalities:

a. Read and Write Operations: The slave device should be able to respond to read and write requests from the master device.

b. Address Decoding: The slave device should decode the address provided by the master and respond to the appropriate address space.

c. Data Handling: For write operations, the slave should accept data from the master and store it in an internal data memory. For read operations, it should provide the requested data back to the master.

4. Testbench: Develop a testbench to verify the functionality of your slave device. Use simulation tools (e.g., ModelSim) to run the testbench and verify that the slave operates as expected under different read and write scenarios.

5. Run and check with DE10 board and get waveforms form SignalTab.

6. Documentation: Prepare clear and concise documentation that explains your design, the Avalon bus interface configuration, and how to use the slave device in a larger system.

Assignment

- Submission:

Submit the following deliverables:

1. HDL source code for the slave device.
2. Testbench code and simulation results.
3. Documentation explaining the design, Avalon bus configuration, and usage of the slave device.

- Feel free to ask if you have any questions or need further clarification on any aspect of the assignment. Good luck with your design!