VLSI System Design (Graduate Level)

Fall 2022

HOMEWORK II

REPORT

Must do self-checking before submission:

Compress all files described in the problem into one tar

All SystemVerilog files can be compiled under SoC Lab environment

All port declarations comply with I/O port specifications

Organize files according to File Hierarchy Requirement

No any waveform files in deliverables

Student name: 黃冠予 俞杉麒

Student ID: M16111064 P78111519

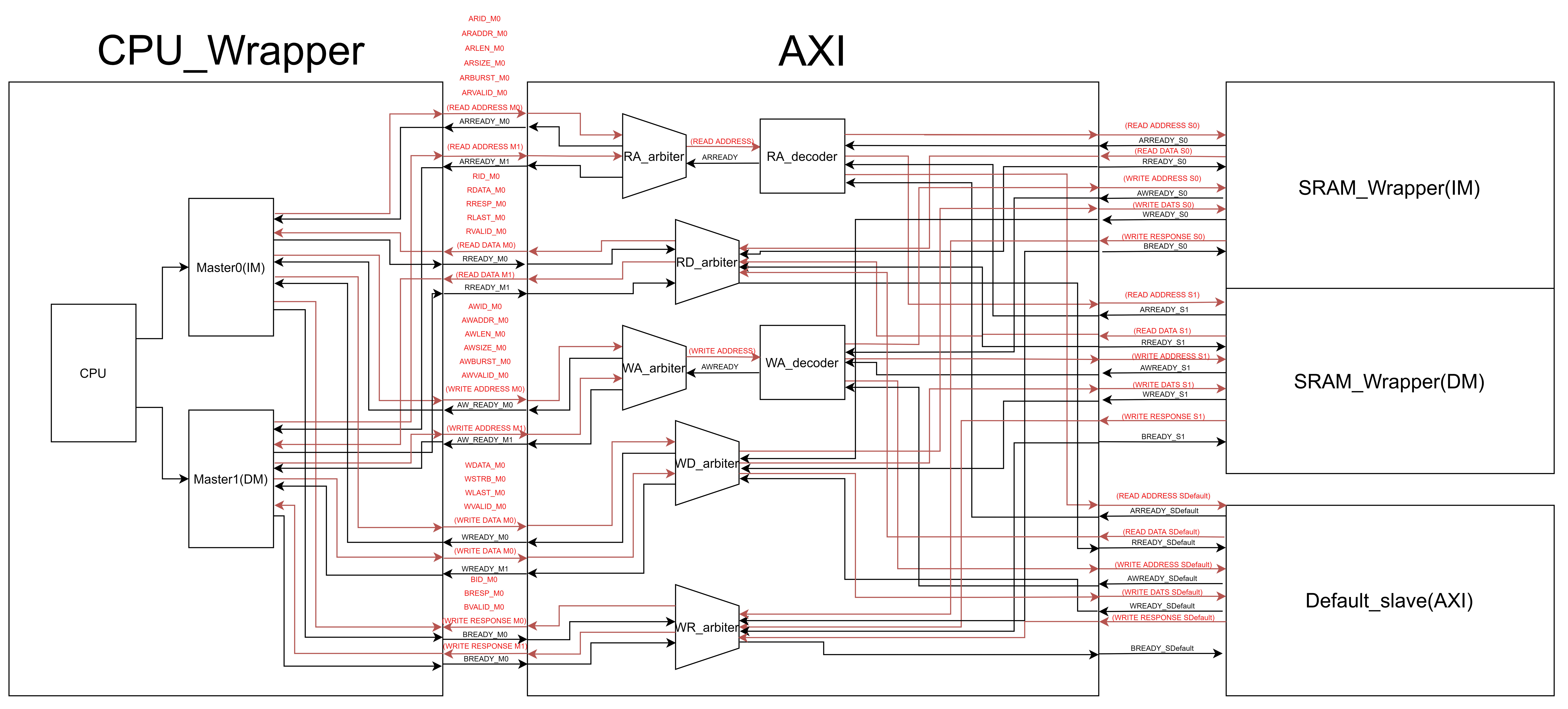
1. Summary

* Implement essential modules for HOMEWORK II.
  + AXI (RA, RD, WA, WD, WR, Arbiter, Decoder)
  + CPU\_wrapper (Master)
  + SRAM\_wrapper
* Add on a stall logic for the CPU implemented in HOMEWORK I, so as to operate the CPU correctly after connecting to AXI bus.

1. Contribution

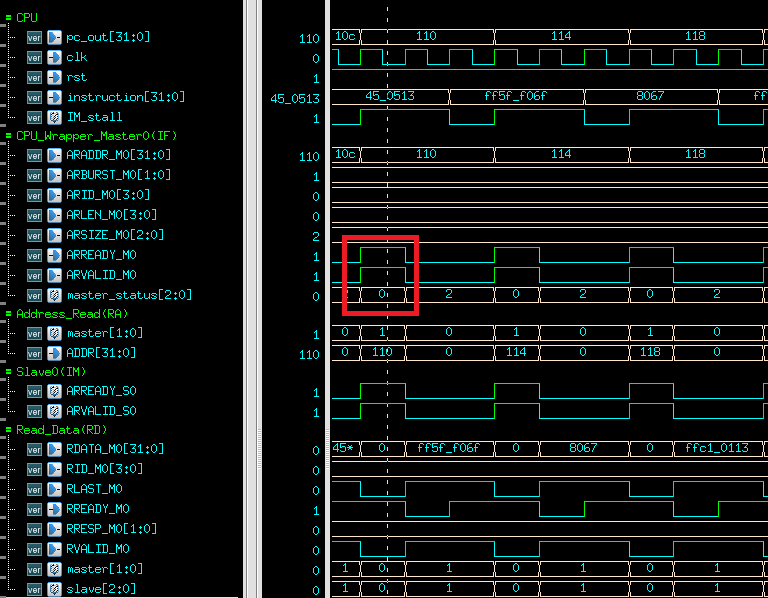
|  |  |
| --- | --- |
| 黃冠予 | 俞杉麒 |
| M16111064 | P78111519 |
| 50% | 50% |

1. Design Structure



1. Simulation Waveform
2. Instruction Fetch

a.

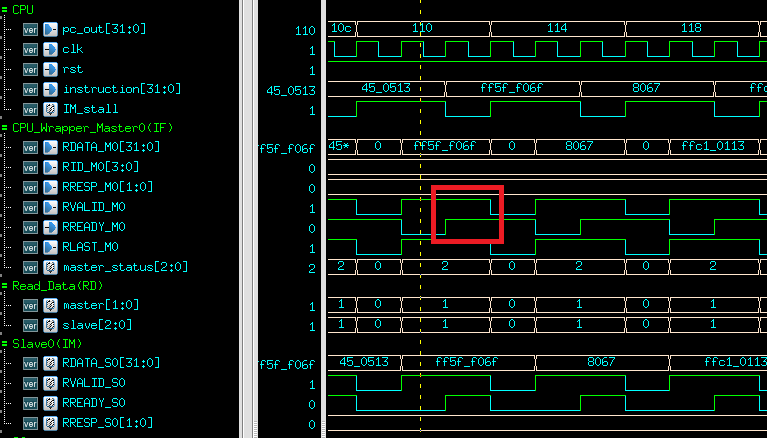


In IF stage, CPU initials a data read request to Master0 in CPU\_Wrapper. ARVALID is then set to 1’b1, and IM\_stall is set to 1’b1 to stall the CPU.

Arbiter in RA chooses the Master0, and ARADDR\_M0 == 0x110 indicates the data is transferred to Slave0(IM), since Slave0 is responsible for the address between 0x0000\_0000 and 0x0000\_FFFF.

ARREADY is then set to 1’b1 to complete the handshake process as the red block shown above.

b.



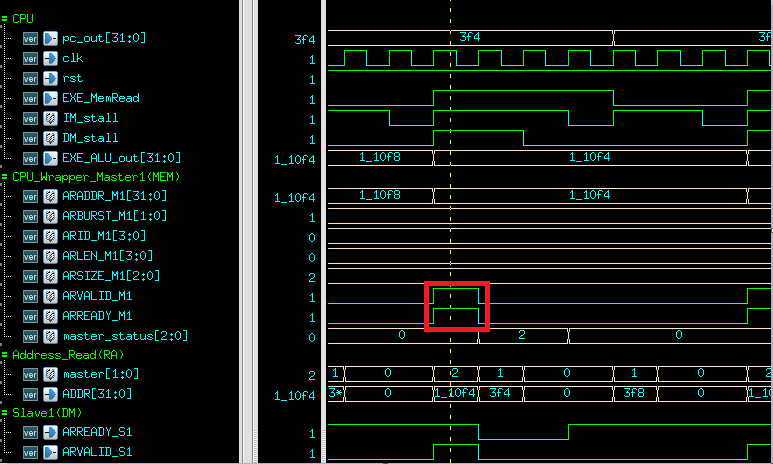
After the handshake process in RA, we enter RD. Slave0(IM) sends the data via RDATA\_S0 to AXI, which eventually transfers the data to RD so as to transfer to Master0. RVALID\_S0 is set to 1’b1 and waits for the RREADY in Master0 to be set as 1’b1. Handshake process hasn’t completed as the yellow line shown above.

After a clock cycle, RREADY in Master0 has set to 1’b1, which indicates the handshake process has completed as the red block shown above.

Data has been correctly sent to CPU, i.e., instruction has been correctly fetched. Stall is then released, and CPU returns executing.

1. LW instruction

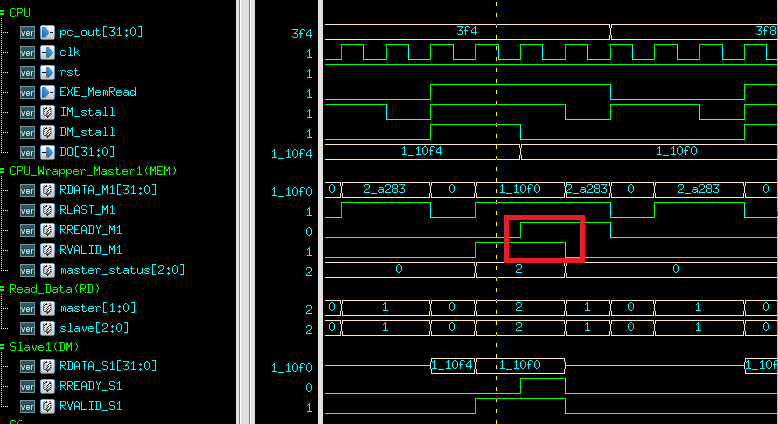
a.



In MEM stage, CPU initials a data read request to Master1 in CPU\_Wrapper. ARVALID is then set to 1’b1, and DM\_stall is set to 1’b1 to stall the CPU.

Arbiter in RA chooses the Master1, and ARADDR\_M1 == 0x1\_10F4 indicates the data is transferred to Slave1(DM), since Slave1 is responsible for the address between 0x0001\_0000 and 0x0001\_FFFF.

ARREADY is then set to 1’b1 to complete the handshake process as the red block shown above.

b. 

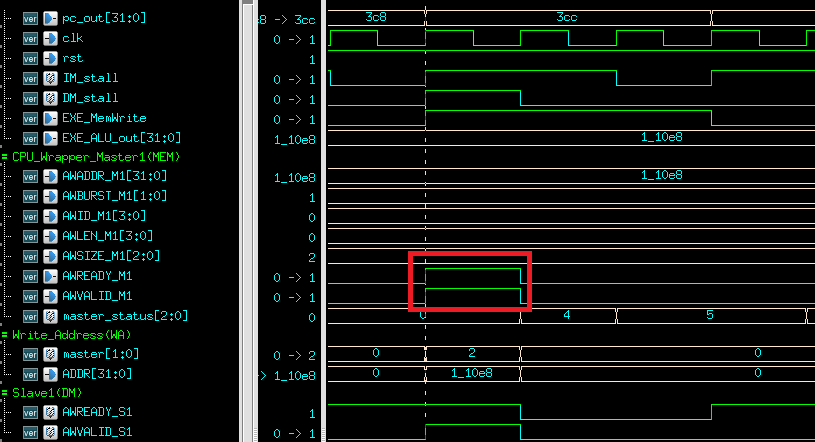
After the handshake process in RA, we enter RD (master\_status == 2). Slave1(DM) sends the data via RDATA\_S1 to AXI, which eventually transfers the data to RD so as to transfer to Master1. RVALID\_S1 is set to 1’b1 and waits for the RREADY in Master1 to be set as 1’b1. Handshake process hasn’t completed as the yellow line shown above.

After a clock cycle, RREADY in Master1 has set to 1’b1, which indicates the handshake process has completed as the red block shown above.

Data has been correctly sent to CPU, i.e., DO receives the RDATA. DM\_Stall is then released, since there is still a request from Master0(IF), we will wait for the request to be completed then the CPU can return executing.

(3) SW instruction

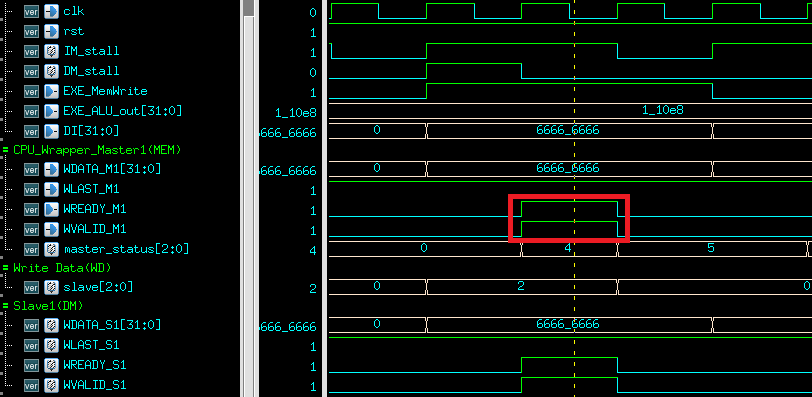
a.



In MEM stage, CPU initials a data write request to Master1 in CPU\_Wrapper. AWVALID is then set to 1’b1, and DM\_stall is set to 1’b1 to stall the CPU.

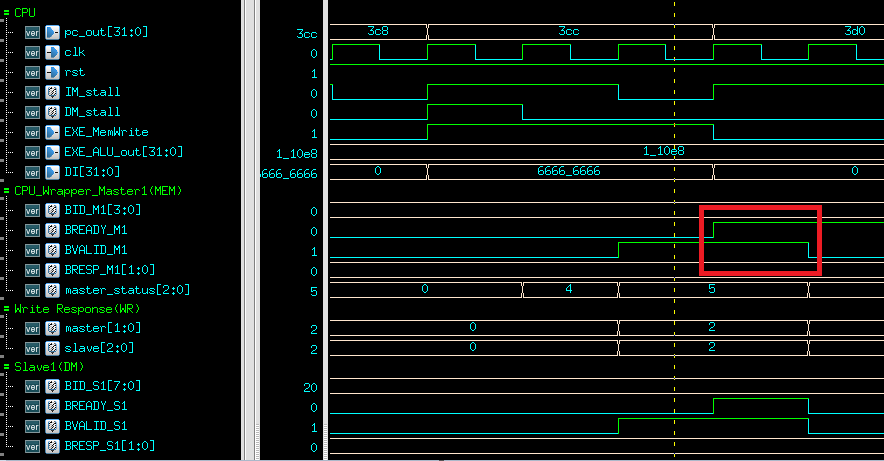
Arbiter in WA chooses the Master1(master == 2), and AWADDR\_M1 == 0x1\_10e8 indicates the data is transferred to Slave1(DM), since Slave1 is responsible for the address between 0x0001\_0000 and 0x0001\_FFFF. AWREADY is then set to 1’b1 to complete the handshake process as the red block shown above.

b.



1. After the handshake process in WA, we enter WD (master\_status == 4). Master1 sends the WDATA to AXI, which eventually transfers the data to WD. WVALID is then set to 1’b1.
2. “slave” in WD implies to transfer the data to DM (slave == 2).
3. Slave1(DM) receives the write data and sets WREADY to 1’b1, which completes the handshake as the red block shown above.

c.

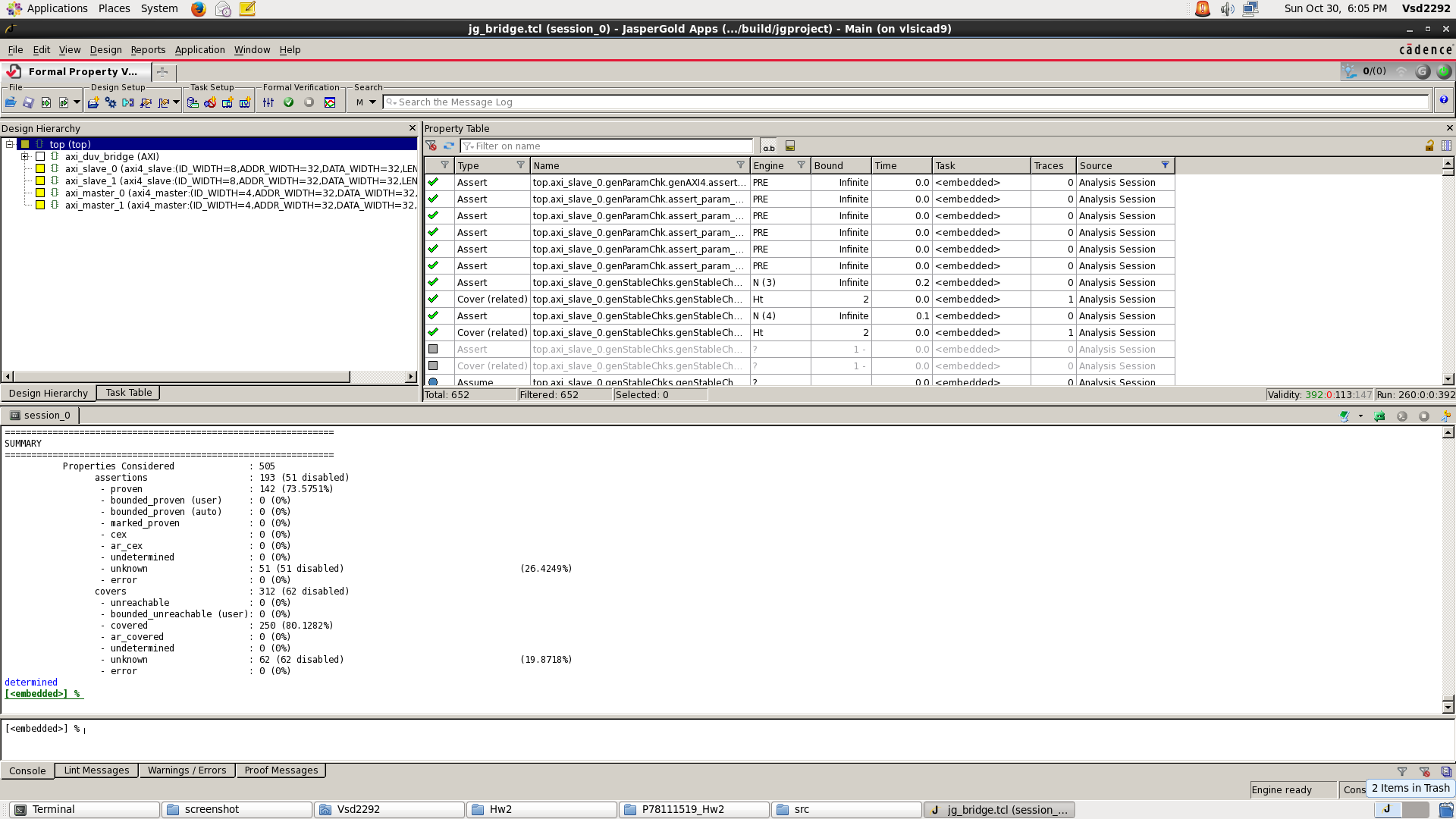


1. After the handshake process in WD, we enter WR (master\_status == 5).
2. Slave1 sends BRESP\_S1 to AXI. BVALID is then set to 1’b1.
3. As the yellow line shown above, WR chooses to connect Master1(master == 2) and Slave1(slave == 2)
4. After a clock cycle, BREADY from Master1 is set as 1’b1, which completes the handshake process as the red block shown above.
5. Lesson Learnt

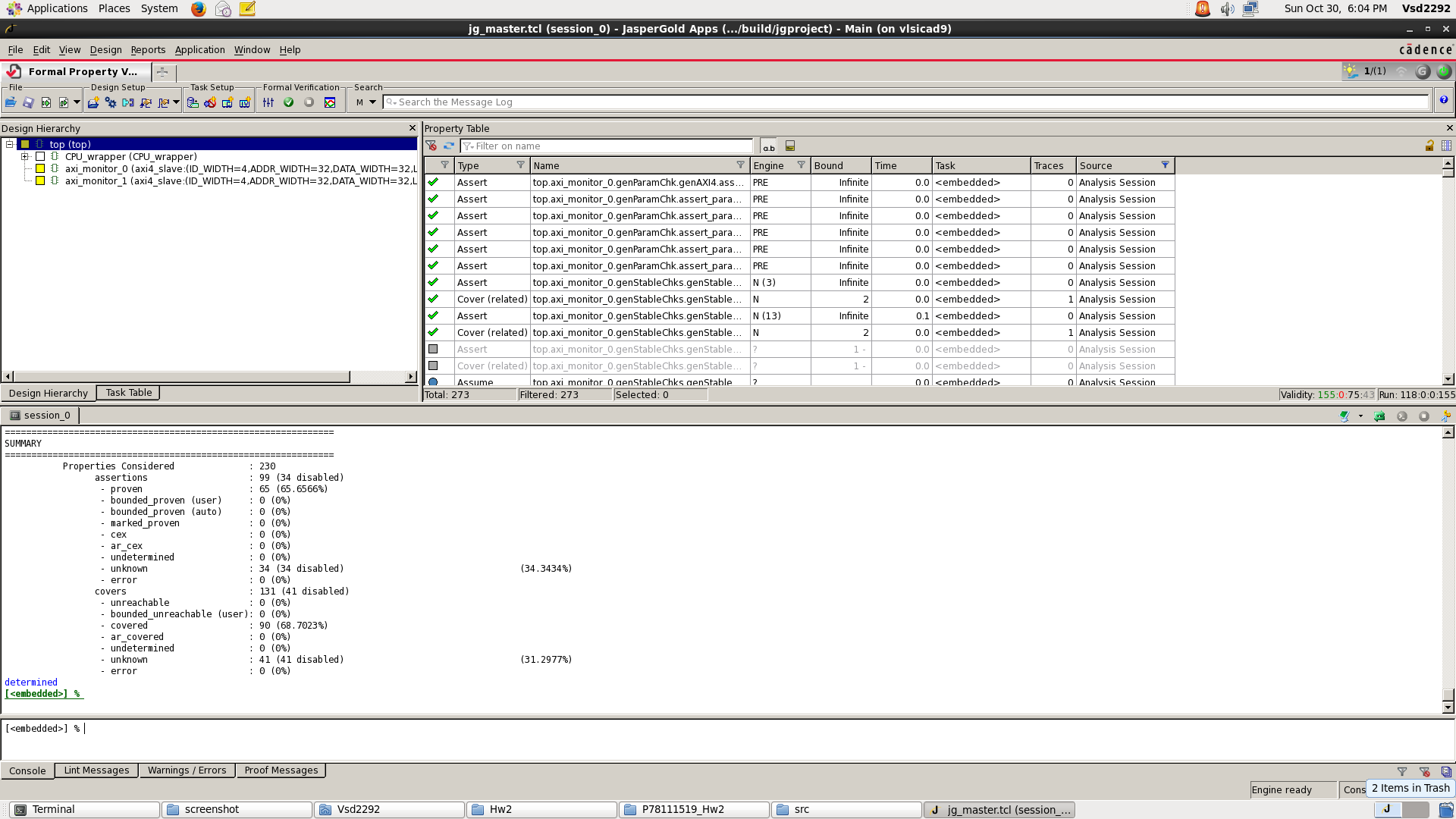
It’s our first time dealing with JasperGold verification. Lots of time are required for verification after implementing the AXI bus since some of the requirements are somewhat implicit to achieve.

There are more to consider in data transferring between multiple masters and slaves, i.e., when 2 masters requiring the same slave. Moreover, “stall” taking turns arising in 2 masters leads to a CPU deadlock, which is eventually solved by adding some restrictions to the data requesting logic for masters.

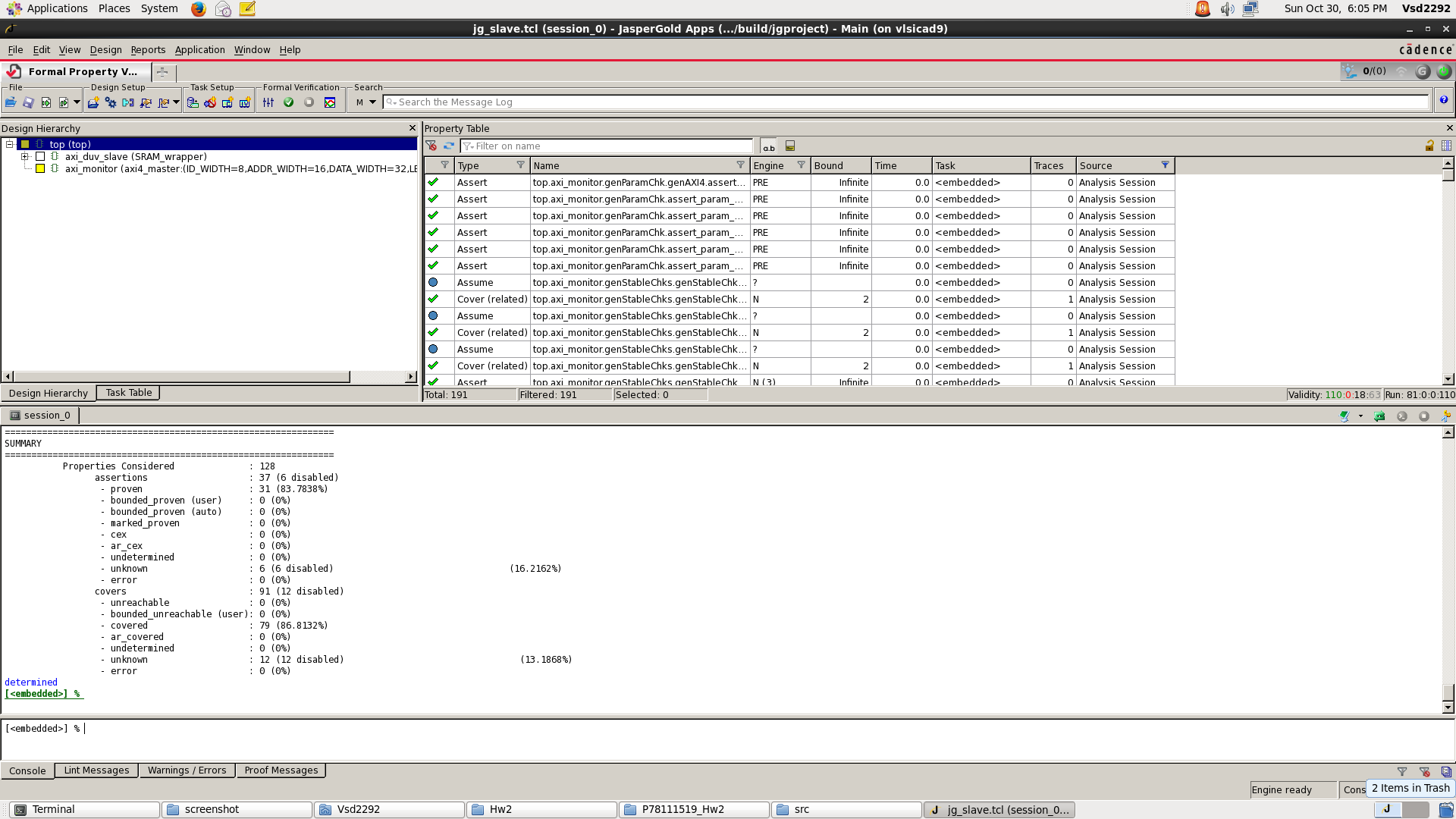
1. JasperGold Verification
2. Bridge



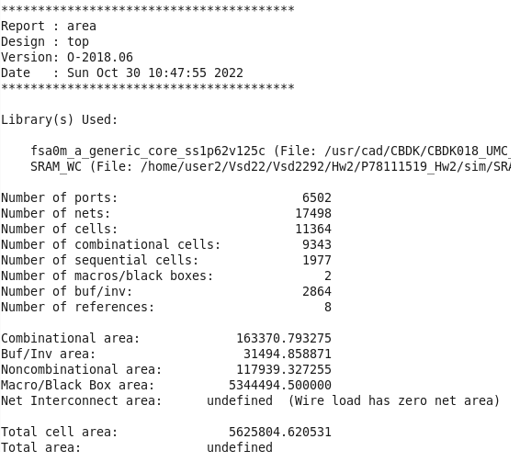
1. CPU\_Wrapper



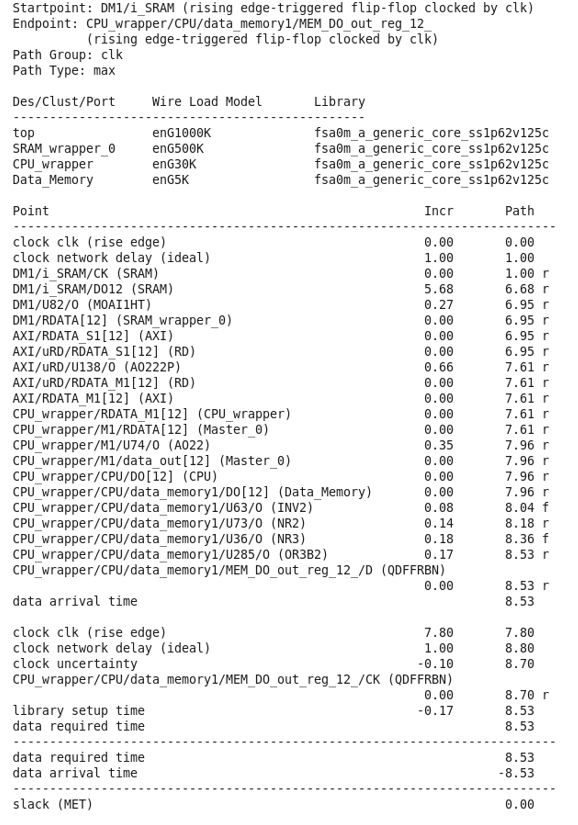
1. SRAM\_Wrapper



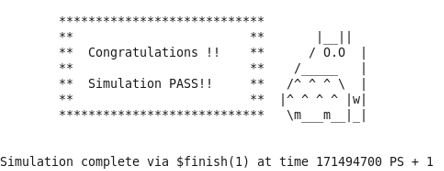
1. Simulation Result
2. Area



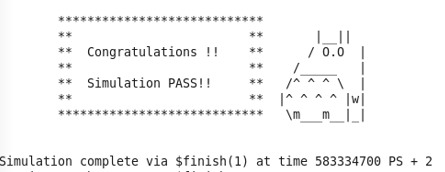
1. Timing



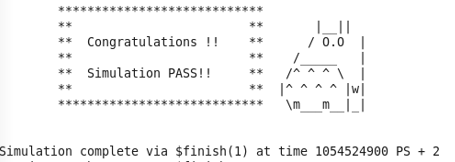
1. Results
2. RTL0



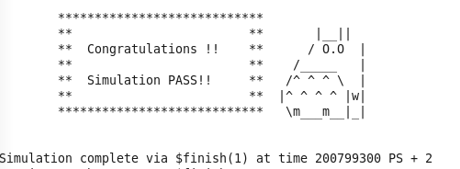
1. RTL1



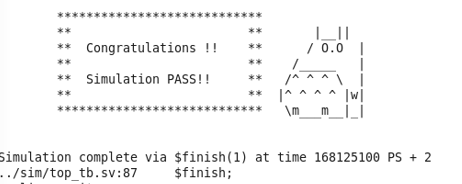
1. RTL2



1. RTL3



1. RTL4



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Prog0 | Prog1 | Prog2 | Prog3 | Prog4 |
| RTL | PASS | PASS | PASS | PASS | PASS |
| SYN | PASS | PASS | PASS | PASS | PASS |

1. Performance

|  |  |
| --- | --- |
| Total Cell Area | 5625804.620531 |
| Clock cycle time | 7.8ns |