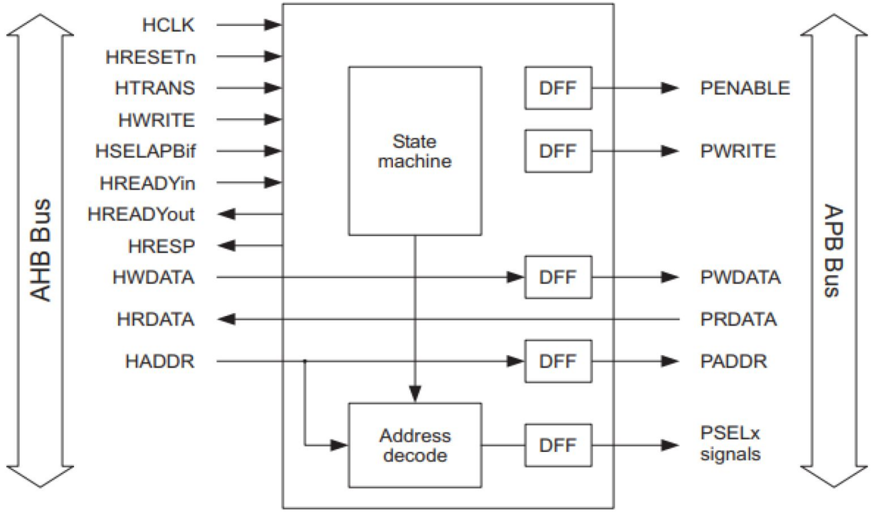
Progress Report Presentation

11/27/2023
Mohamed Ghonim, Hayden Galante, Alex Maso, Phanindra Vemireddy

Project Proposal/Overview

We're verifying the AHB to APB bridge using the AMBA specifications. We are basically writing an assertion IP (AIP) based on the AMBA ARM specs and testing/applying it on an RTL implementation from Github.



Bus Bridge Description

● AHB to APB bridge provides an interface between the high speed AHB and the low power APB.

● Read and write transfers on the AHB are converted into equivalent transfers on the APB.

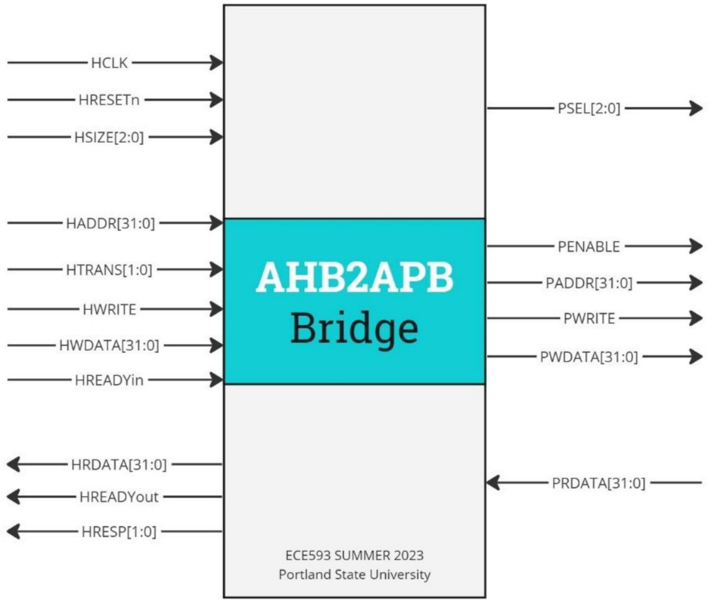
● It has AHB slave bus interface, APB transfer state machine, which is independent of the device memory map, and APB output signal generation.

● It buffers address, controls, and data from the AHB, drives the APB peripherals and returns data along with a response signal to the AHB.

● APB data bus is divided into read (PRDATA), where data travels from the peripherals to the bridge and write (PWDATA), where data travels from the bridge to the peripherals.

● Data transfers can have 3 different sizes (HSIZE) Byte, halfword or full word.

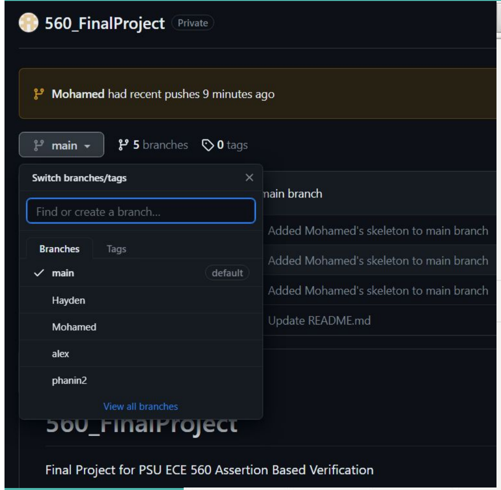
● Data transfers are either sent as a single transfer or can be sent in a burst (HBURST). Bursts can be incremental or wrapping bursts with 4 or 8-beats.



Project Collaboration and file Structure

We created a github repo with 4 branches, one for each team member

Initially, we “explored” the design With the AEP and FXP apps


560_FinalProject Private

Mohamed had recent pushes 9 minutes ago

main · 5 branches · 0 tags

Switch branches/tags

Find or create a branch...

Branches · Tags

- ✓ main — default
- Hayden
- Mohamed
- alex
- phanin2

View all branches

main branch

Added Mohamed's skeleton to main branch

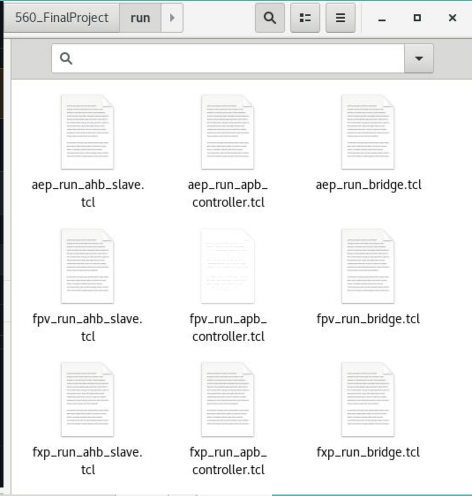
Added Mohamed's skeleton to main branch

Added Mohamed's skeleton to main branch

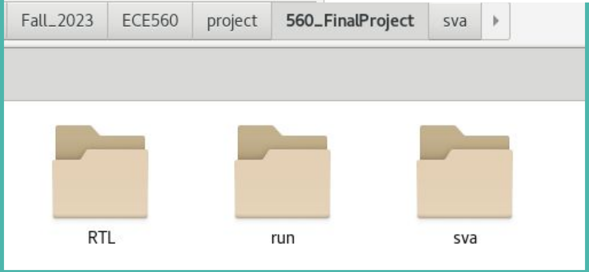
Update README.md

560_FinalProject

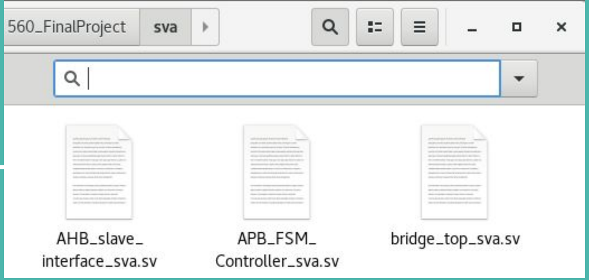
Final Project for PSU ECE 560 Assertion Based Verification


560_FinalProject › run

- aep_run_ahb_slave.tcl
- aep_run_apb_controller.tcl
- aep_run_bridge.tcl
- fpv_run_ahb_slave.tcl
- fpv_run_apb_controller.tcl
- fpv_run_bridge.tcl
- fxp_run_ahb_slave.tcl
- fxp_run_apb_controller.tcl
- fxp_run_bridge.tcl


Fall_2023 › ECE560 › project › 560_FinalProject › sva

- RTL
- run
- sva


560_FinalProject › sva

- AHB_slave_interface_sva.sv
- APB_FSM_Controller_sva.sv
- bridge_top_sva.sv

Input Port Signals

Input Port:

AHB:

- HCLK: Clock signal for the AHB interface.
- HRESETn: Active-low reset signal for the AHB interface.
- HSIZE[2:0]: Size of the transfer on the AHB interface.
- HADDR[31:0]: Address for the AHB transfer.
- HTRANS[1:0]: Transfer type on the AHB interface.
- HWRITE: Write enable signal for the AHB interface.
- HWDATA[31:0]: Write data on the AHB interface.
- HREADYin: Ready signal indicating the availability of the AHB interface.

APB:

- PRDATA[31:0]: Read data on the APB interface.

Output Port Signals

AHB:

- HRDATA[31:0]: Read data from the AHB interface.
- HREADYout: Ready signal indicating the readiness of the AHB interface.
- HRESP[1:0]: Response indicating the status of the AHB transfer.

APB:

- PSEL[2:0]: Slave select signal on the APB interface.
- PENABLE: Enable signal for the APB interface.
- PADDR[31:0]: Address for the APB transfer.
- PWRITE: Write enable signal for the APB interface.
- PWDATA[31:0]: Write data on the APB interface.

Read transfer to AHB

- HWRITE - Low for Read
- HREADY - AHB asserted with HWRITE then strobed once
- PSEL - Asserted for two cycles along with PWRITE starting in T3 - Enter SETUP
- PENABLE is asserted in T4, the cycle after PSEL is asserted - Enter ENABLE
- The following cycle, T5, PSEL and PENABLE deasserted. - Single Read

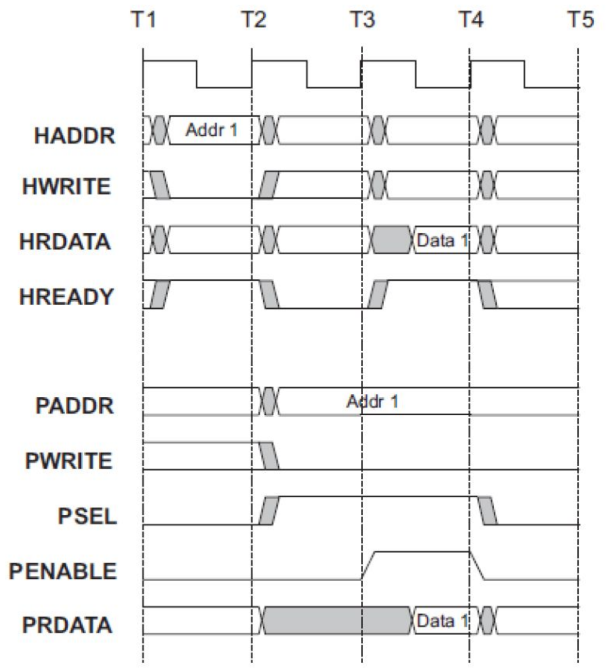


Figure 5-9 Read transfer to AHB

Burst of Read Transfers

- HWRITE - Held low for Seq Read
- HREADY - AHB asserted with HWRITE then strobed during burst
- PSEL - Asserted along with PWRITE in T3 - Enter SETUP
- PENABLE is asserted in T4, the cycle after PSEL is asserted - Enter ENABLE
- The following cycle, T5, PENABLE deasserted. Strobed for burst
- PSEL held high for burst

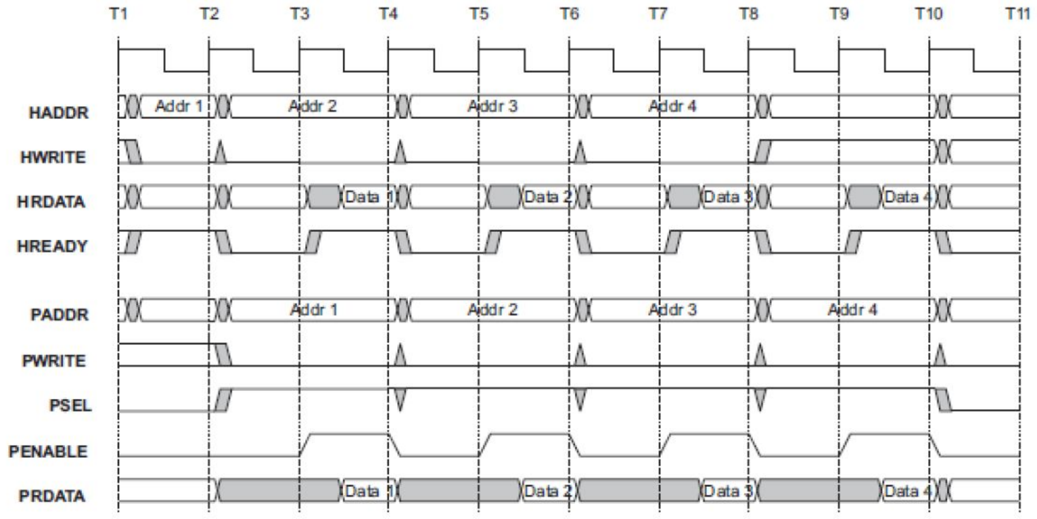


Figure 5-10 Burst of read transfers

Write Transfer from AHB

- HWRITE - High for Read
- HREADY - AHB asserted for two cycles
- PSEL - Asserted for two cycles along with PWRITE starting in T4 - Enter SETUP
- PENABLE is asserted in T5, the cycle after PSEL is asserted Enter ENABLE
- The following cycle, T6, PSEL and PENABLE deasserted. - Single Read

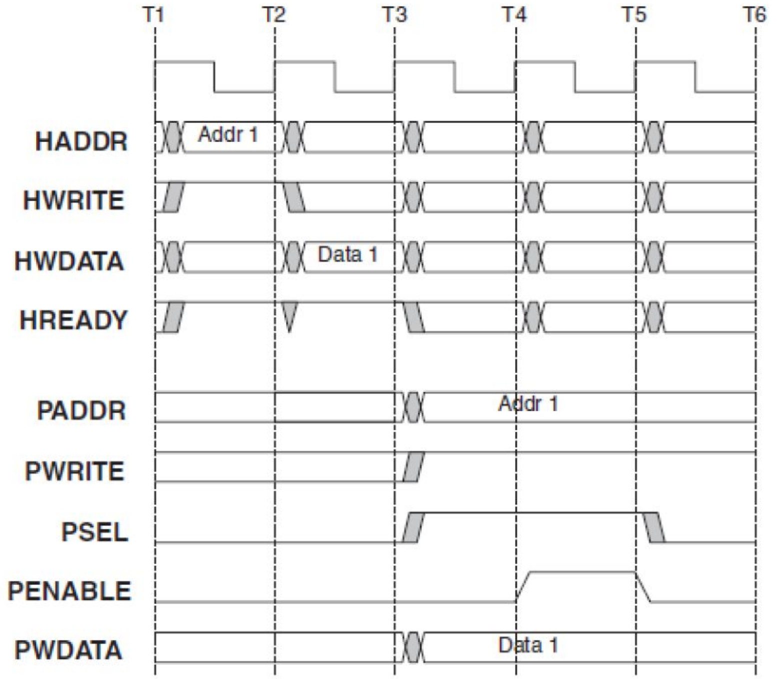


Figure 5-11 Write transfer from AHB

Burst of Write Transfers

- HWRITE - Held high for Seq Read
- HREADY - AHB asserted with HWRITE then strobed during burst
- PWRITE - Low in T2, after AHB asserts HREADY
- PSEL - Asserted along with PWRITE - Enter SETUP
- PENABLE is asserted in T3, the cycle after PSEL is asserted - Enter ENABLE
- The following cycle, T4, PENABLE deasserted. Cycles high and low for burst
- PSEL held high for burst

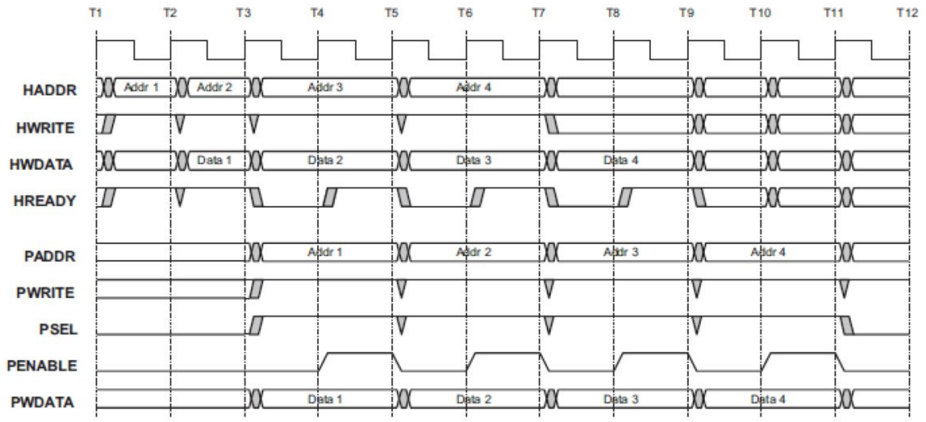
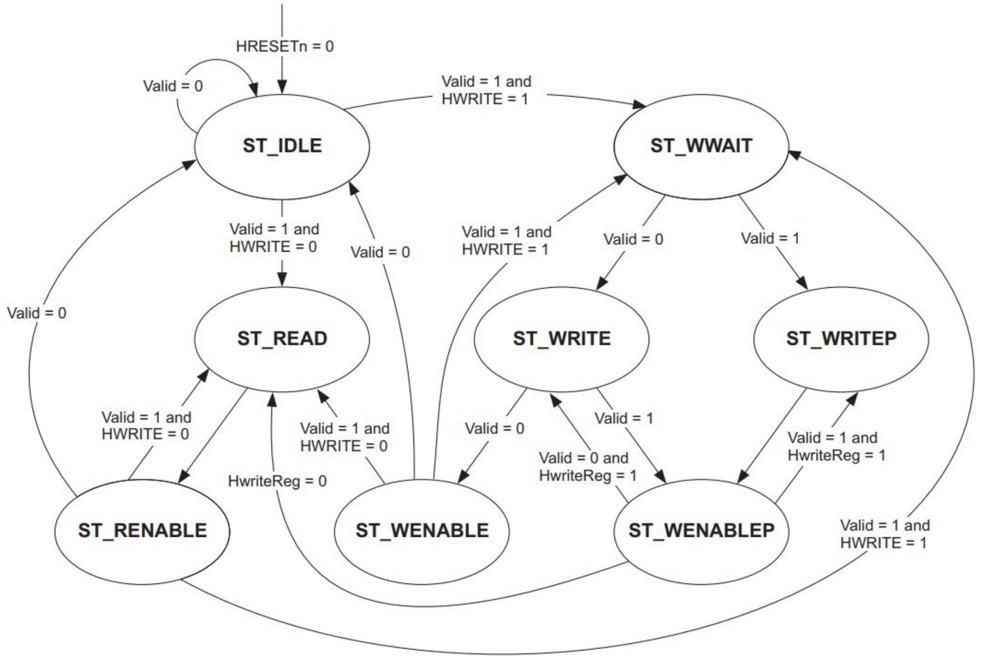
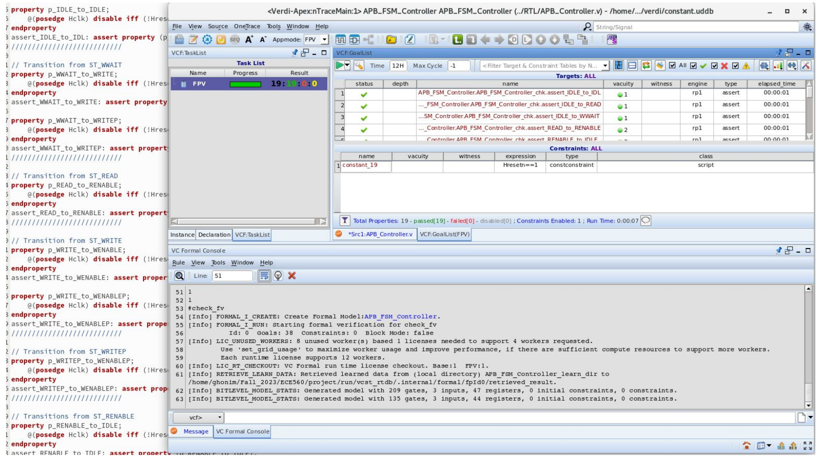


Figure 5-12 Burst of write transfers

Verifying the State Machine for the AHB to APB interface



FSM from the official AMBA Technical Reference



```
property p_IDLE_to_IDLE;
    @(posedge Hclk) disable iff (!Hres
endproperty
assert_IDLE_to_IDL: assert property (p
/////////////////////////////

// Transition from ST_WWAIT
property p_WWAIT_to_WRITE;
    @(posedge Hclk) disable iff (!Hres
endproperty
assert_WWAIT_to_WRITE: assert property

property p_WWAIT_to_WRITEP;
    @(posedge Hclk) disable iff (!Hres
endproperty
assert_WWAIT_to_WRITEP: assert propert
/////////////////////////////

// Transition from ST_READ
property p_READ_to_RENABLE;
    @(posedge Hclk) disable iff (!Hres
endproperty
assert_READ_to_RENABLE: assert propert
/////////////////////////////

// Transition from ST_WRITE
property p_WRITE_to_WENABLE;
    @(posedge Hclk) disable iff (!Hres
endproperty
assert_WRITE_to_WENABLE: assert proper

property p_WRITE_to_WENABLEP;
    @(posedge Hclk) disable iff (!Hres
endproperty
assert_WRITE_to_WENABLEP: assert prope
/////////////////////////////

// Transition from ST_WRITEP
property p_WRITEP_to_WENABLEP;
    @(posedge Hclk) disable iff (!Hres
endproperty
assert_WRITEP_to_WENABLEP: assert prop
/////////////////////////////

// Transitions from ST_RENABLE
property p_RENABLE_to_IDLE;
    @(posedge Hclk) disable iff (!Hres
endproperty
assert_RENABLE_to_IDLE: assert property
```

<Verdi-Apex:nTraceMain:1> APB_FSM_Controller APB_FSM_Controller (../RTL/APB_Controller.v) - /home/.../verdi/constant.uddb

VCF:TaskList

Name	Progress	Result
FPV		19:0:0

VCF:GoalList

Time 12H Max Cycle -1

Targets: ALL

	status	depth	name	vacuity	witness	engine	type	elapsed_time
1	✓		APB_FSM_Controller.APB_FSM_Controller_chk.assert_IDLE_to_IDL	● 1		rp1	assert	00:00:01
2	✓		..._FSM_Controller.APB_FSM_Controller_chk.assert_IDLE_to_READ	● 1		rp1	assert	00:00:01
3	✓		...SM_Controller.APB_FSM_Controller_chk.assert_IDLE_to_WWAIT	● 1		rp1	assert	00:00:01
4	✓		..._Controller.APB_FSM_Controller_chk.assert_READ_to_RENABLE	● 2		rp1	assert	00:00:01

Constraints: ALL

	name	vacuity	witness	expression	type	class
1	constant_19			Hresetn==1	constconstraint	script

Total Properties: 19 - passed[19] - failed[0] - disabled[0] ; Constraints Enabled: 1 ; Run Time: 0:00:07

VC Formal Console

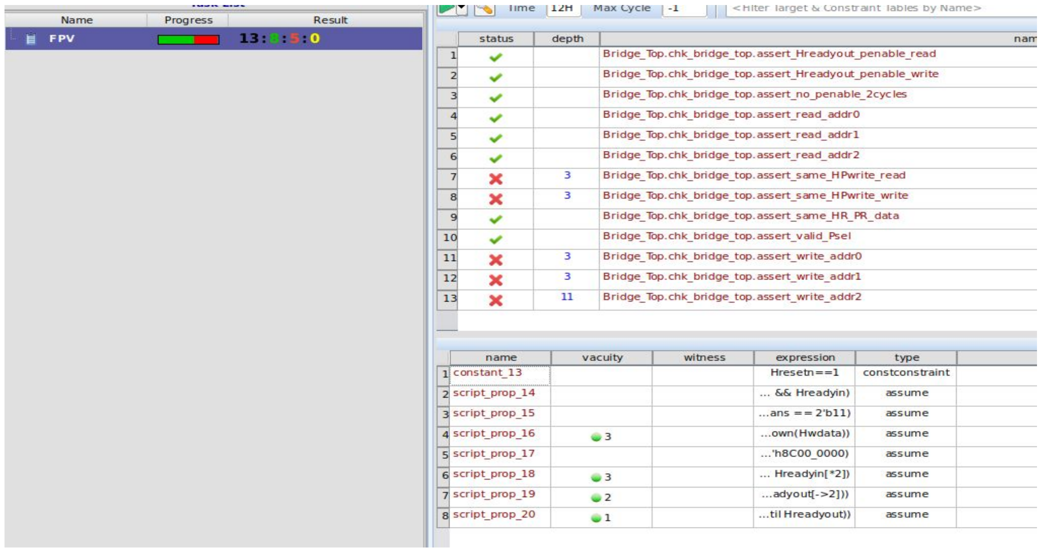
Line: 51

```
1
1
#check_fv
[Info] FORMAL_I_CREATE: Create Formal Model:APB_FSM_Controller.
[Info] FORMAL_I_RUN: Starting formal verification for check_fv
          Id: 0  Goals: 38  Constraints: 0  Block Mode: false
[Info] LIC_UNUSED_WORKERS: 8 unused worker(s) based 1 licenses needed to support 4 workers requested.
        Use 'set_grid_usage' to maximize worker usage and improve performance, if there are sufficient compute resources to support more workers.
        Each runtime license supports 12 workers.
[Info] LIC_RT_CHECKOUT: VC Formal run time license checkout. Base:1  FPV:1.
[Info] RETRIEVE_LEARN_DATA: Retrieved learned data from (local directory) APB_FSM_Controller_learn_dir to
/home/ghonim/Fall_2023/ECE560/project/run/vcst_rtdb/.internal/formal/fpId0/retrieved_result.
[Info] BITLEVEL_MODEL_STATS: Generated model with 209 gates, 3 inputs, 47 registers, 0 initial constraints, 0 constraints.
[Info] BITLEVEL_MODEL_STATS: Generated model with 135 gates, 3 inputs, 44 registers, 0 initial constraints, 0 constraints.
```

Wrote 19 assertions testing the 19 transitions in the FSM. They all passed.

Verifying the Read and Write Transactions

- Wrote multiple assertions to check validity for output signals like, PENABLE, PSEL, PWRITE, HRDATA
- Wrote a new assumption for back to back reads after observing failing assertions for read transfers
- Some of them are failing during the scenario of Read followed by Write
- Need to update assertions for the falling scenario



Name	Progress	Result
FPV	🟩🟥	13:8:5:0

	status	depth	nam
1	✓		Bridge_Top.chk_bridge_top.assert_Hreadyout_penable_read
2	✓		Bridge_Top.chk_bridge_top.assert_Hreadyout_penable_write
3	✓		Bridge_Top.chk_bridge_top.assert_no_penable_2cycles
4	✓		Bridge_Top.chk_bridge_top.assert_read_addr0
5	✓		Bridge_Top.chk_bridge_top.assert_read_addr1
6	✓		Bridge_Top.chk_bridge_top.assert_read_addr2
7	✗	3	Bridge_Top.chk_bridge_top.assert_same_HPwrite_read
8	✗	3	Bridge_Top.chk_bridge_top.assert_same_HPwrite_write
9	✓		Bridge_Top.chk_bridge_top.assert_same_HR_PR_data
10	✓		Bridge_Top.chk_bridge_top.assert_valid_Psel
11	✗	3	Bridge_Top.chk_bridge_top.assert_write_addr0
12	✗	3	Bridge_Top.chk_bridge_top.assert_write_addr1
13	✗	11	Bridge_Top.chk_bridge_top.assert_write_addr2

	name	vacuity	witness	expression	type
1	constant_13			Hresetn==1	constconstraint
2	script_prop_14			... && Hreadyin)	assume
3	script_prop_15			...ans == 2'b11)	assume
4	script_prop_16	● 3		...own(Hwdata))	assume
5	script_prop_17			...'h8C00_0000)	assume
6	script_prop_18	● 3		... Hreadyin[*2])	assume
7	script_prop_19	● 2		...adyout[->2]))	assume
8	script_prop_20	● 1		...til Hreadyout))	assume

Read after Write transfer

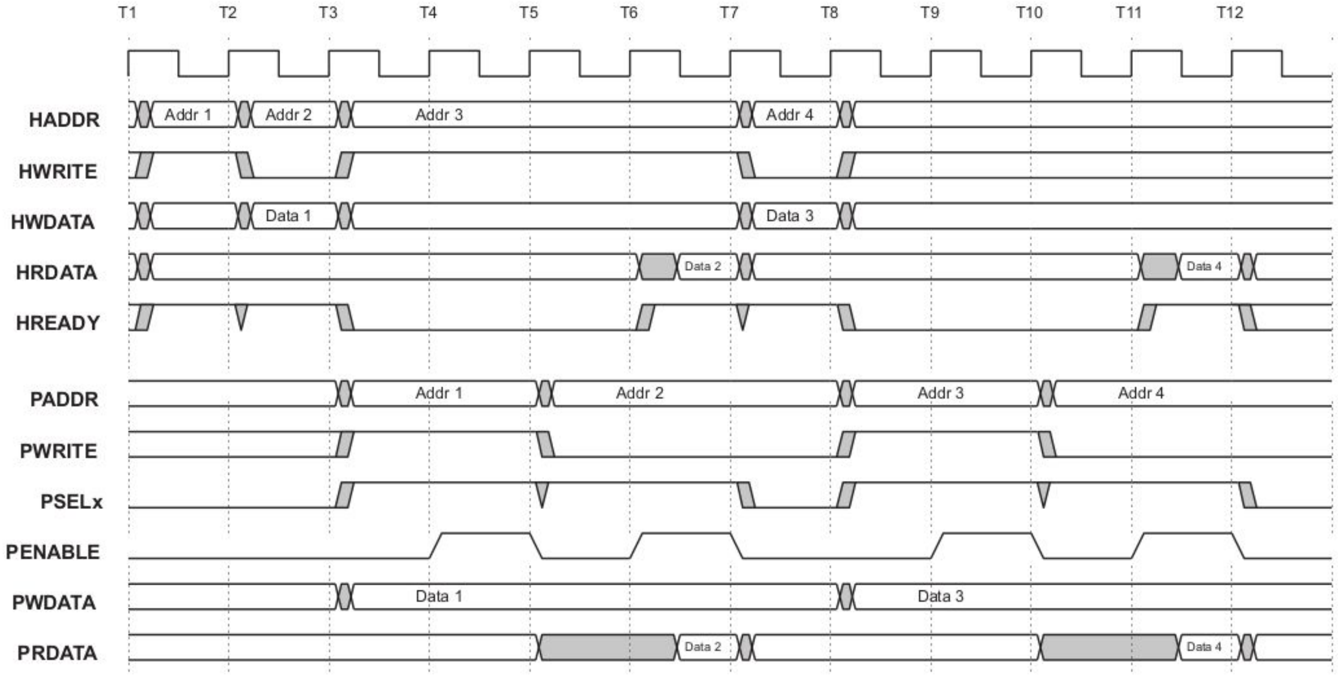
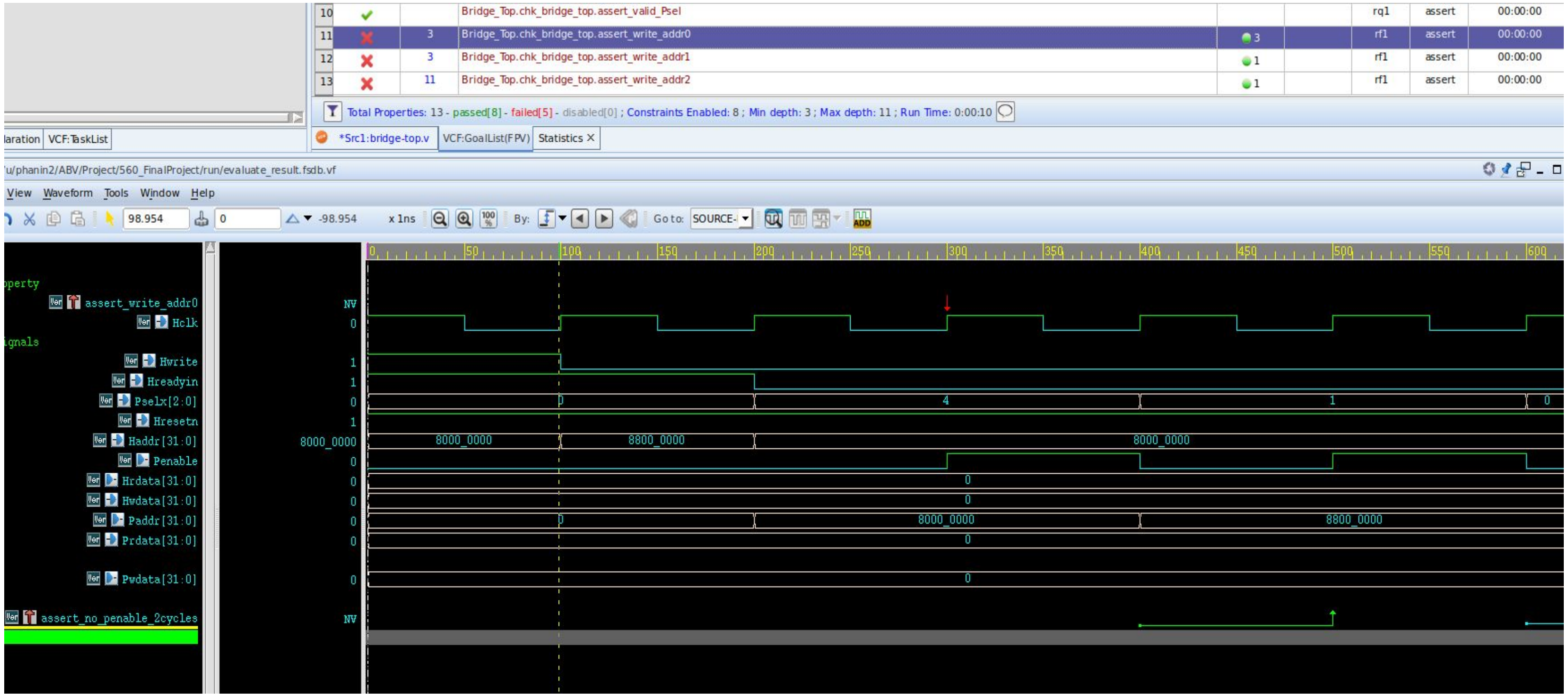


Figure 5-13 Back to back transfers

Pselx Issue



10	✓		Bridge_Top.chk_bridge_top.assert_valid_Psel			rq1	assert	00:00:00
11	✗	3	Bridge_Top.chk_bridge_top.assert_write_addr0	● 3		rf1	assert	00:00:00
12	✗	3	Bridge_Top.chk_bridge_top.assert_write_addr1	● 1		rf1	assert	00:00:00
13	✗	11	Bridge_Top.chk_bridge_top.assert_write_addr2	● 1		rf1	assert	00:00:00

Total Properties: 13 - passed[8] - failed[5] - disabled[0] ; Constraints Enabled: 8 ; Min depth: 3 ; Max depth: 11 ; Run Time: 0:00:10

'u/phanin2/ABV/Project/560_FinalProject/run/evaluate_result.fsdb.vf

```
operty
  assert_write_addr0     NV
    Hclk                 0
ignals
    Hwrite               1
    Hreadyin             1
    Pselx[2:0]           0    0 | 4 | 1 | 0
    Hresetn              1
    Haddr[31:0]  8000_0000    8000_0000 | 8800_0000 | 8000_0000
    Penable              0
    Hrdata[31:0]         0    0
    Hwdata[31:0]         0    0
    Paddr[31:0]          0    0 | 8000_0000 | 8800_0000
    Prdata[31:0]         0    0
    Pwdata[31:0]         0    0
  assert_no_penable_2cycles  NV
```

Next steps, and what to verify next.

So far, we have written assertions and assumptions verifying the basic functionality of the bridge, and we plan to write more thorough assertions and cover properties between now and the due date.

Issues we ran into:

We couldn’t find an RTL that clearly and completely implements the HBURST mechanism with multiple burst transactions. We decided to still write the (AIP) for those transactions, and if time permits, one or two of our teammates will write the HBURST code and the other one or two will write the assertions to verify it, but overall, we set out goal to “translate” the specs into properties (assertions and assumptions) and test as many of them as we can given the available RTLs, but there’s a chance we’ll have more properties than those implemented in the RTLs available, we’ll use ifdefs to isolate our additional assertions when running VC Formal.