

Assignment-2 Submission

Indian Institute of Technology, Delhi

COL216: Computer Architecture

1 Introduction

The overall assignment is designing hardware for implementing a processor that can execute a subset of ARM instructions. This is the stage 1 of assignment where I have designed and tested basic modules including ALU, Register Files, Program Memory and Data Memory.

2 Stage 1

2.1 Objective

The objective of this stage is to design 4 components of the hardware, ALU, Register Files, Program Memory and Data Memory.

ALU is a a combinational circuit that takes two operands, does one of the 16 operations and produces a result. Operands and results are 32-bit `std_logic_vectors`. Further, there is a carry input and a carry output. The operation to be done is specified by another input (opcode) that is a 4-bit `std_logic_vector`.

Register File contains an array of 16 `std_logic_vectors` of 32-bits each. Its inputs include two read addresses, one write address, one data input, one write enable and a clock. There are two data outputs on which contents of the array elements selected by read addresses are continuously available. If write enable is active, at clock edge the input data gets written in the array element selected by write address.

The program and data memory contains an array of 64 `std_logic_vectors` of 32-bits. Contents of Program Memory are initialized in declaration itself as all zeroes. Data Memory has one read port and one write port, whereas Program Memory has only a read port. Read/write operations are modelled in same way as Register File, that is, write is clocked whereas read is unlocked (like a combinational circuit). There are 4 write enable signals to provide byte level write operation.

2.2 Assumptions

I have used Edaplayground to write and test my code.

I have assumed that the ALU arithmetic operations are all signed. Another assumption I have made is that no flags are set inside the ALU. The process of setting flags will be taken care of in subsequent stages.

I have also assumed that the vectors and signals are all of the type `std_logic`.

One significant assumption is that the address input to program memory and data memory are 8 bits long, even though there are only 64 vectors. This is done to align the memory with bytes. The first 6 bits are taken for indexing the array.

2.3 Logistics

There are a total of 8 files. The 4 main files are `ALU.vhd`, `program_memory.vhd`, `data_memory.vhd` and `register_file.vhd`. Each file is accompanied by a test bench that was used to test the code.

Other than this, the zip file may contain some screenshots of the EPWaves generated and the synthesis report as was the output on edaplaygrounds.

The testbench for ALU unit was written by using a C++ code (also witten by me).

2.4 Contributions

Help was taken from already written examples on edaplaygrounds. Only those examples were picked up which were also used in the class. The run.do file was taken from one of the examples already on the website, the permission for doing so was taken in the class itself. No other help of any form has been taken.

2.5 Test Cases

The program was tested on many different test cases.

The test cases were designed to exhaustively cover all the cases.

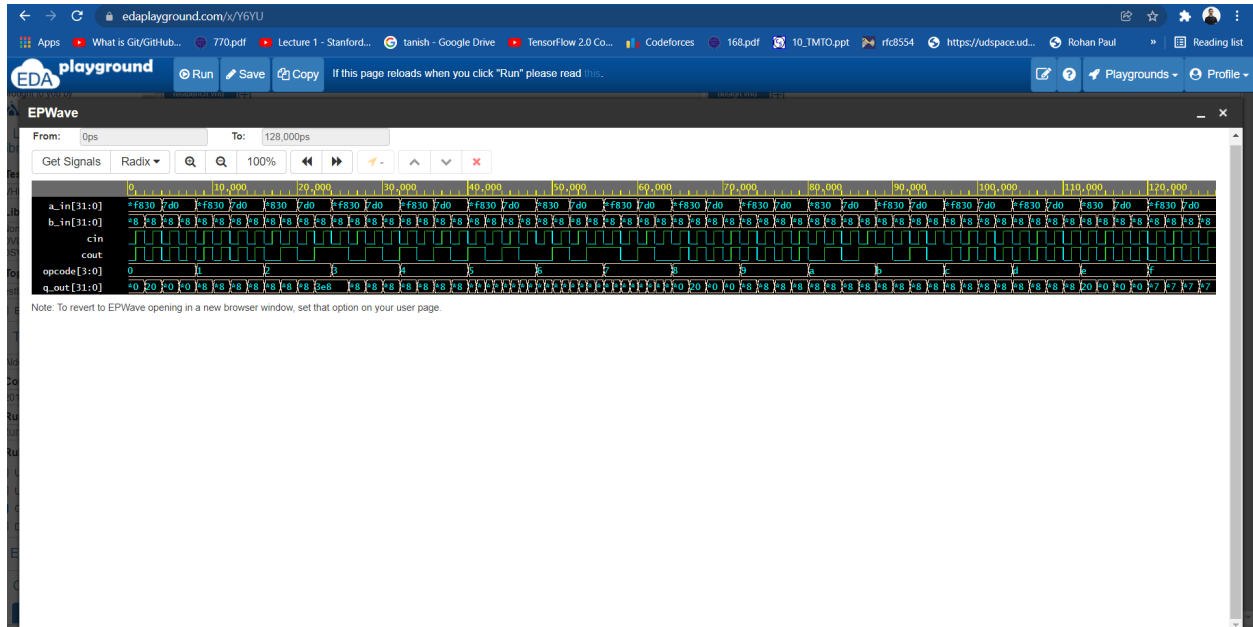


Figure 1: ALU EPWave

```

# (C) 1999-2020 Aldec, Inc. All rights reserved.
# ELBREAD: Elaboration process.
# ELBREAD: Elaboration time 0.0 [s].
# KERNEL: Main thread initiated.
# KERNEL: Kernel process initialization phase.
# ELAB2: Elaboration final pass...
# ELAB2: Create instances ...
# KERNEL: Time resolution set to 1ps.
# ELAB2: Create instances complete.
# SLP: Started
# SLP: Elaboration phase ...
# SLP: Elaboration phase ... skipped, nothing to simulate in SLP mode : 0.0 [s]
# SLP: Finished : 0.0 [s]
# ELAB2: You do not have a license to run VHDL performance optimized simulation. Contact Aldec for ordering information - sales@aldec.com.
# ELAB2: Elaboration final pass complete - time: 0.0 [s].
# KERNEL: Warning: You are using the Riviera-PRO EDU Edition. The performance of simulation is reduced.
# KERNEL: Warning: Contact Aldec for available upgrade options - sales@aldec.com.
# KERNEL: Kernel process initialization done.
# Allocation: Simulator allocated 5616 kb (elbread=427 elab2=5046 kernel=143 sdf=0)
# KERNEL: ASDB file was created in location /home/runner/dataset.asdb
# KERNEL: PLI/VHPI kernel's engine initialization done.
# PLI: Loading library '/usr/share/Riviera-PRO/bin/libsysf.so'
# EXECUTION:: NOTE : Test done.
# EXECUTION:: Time: 128 ns, Iteration: 0, Instance: /testbench, Process: line_32.
# KERNEL: Simulation has finished. There are no more test vectors to simulate.
# VSIM: Simulation has finished.
Finding VCD file...
./dump.vcd
[2022-02-09 07:36:36 EST] Opening EPWave...
Done

```

Figure 2: ALU Logs

Resource Utilization Table:

Resource	Used	Avail	Utilization
Info: I/Os	102	210	48.57%
Info: Global Buffers	0	32	0.00%
Info: LUTs	165	63400	0.26%
Info: CLB Slices	41	15850	0.26%
Info: DFFs or Latches	0	126800	0.00%
Info: Block RAMs	0	135	0.00%
Info: DSP48E1s	0	240	0.00%

Synthesis Statistics:

- Info: Library: work Cell: ALU View: rtl
- Info: Number of ports : 102
- Info: Number of nets : 403
- Info: Number of instances : 302
- Info: Number of references to this view : 0
- Info: Total accumulated area :
- Info: Number of LUTs : 165
- Info: Number of Primitive LUTs : 166
- Info: Number of LUTs with LUTNM/HLUTNM : 2
- Info: Number of MUX CABRYS : 64
- Info: Number of accumulated instances : 399
- Info: IO Register Mapping Report

Figure 3: ALU_{synthesis}

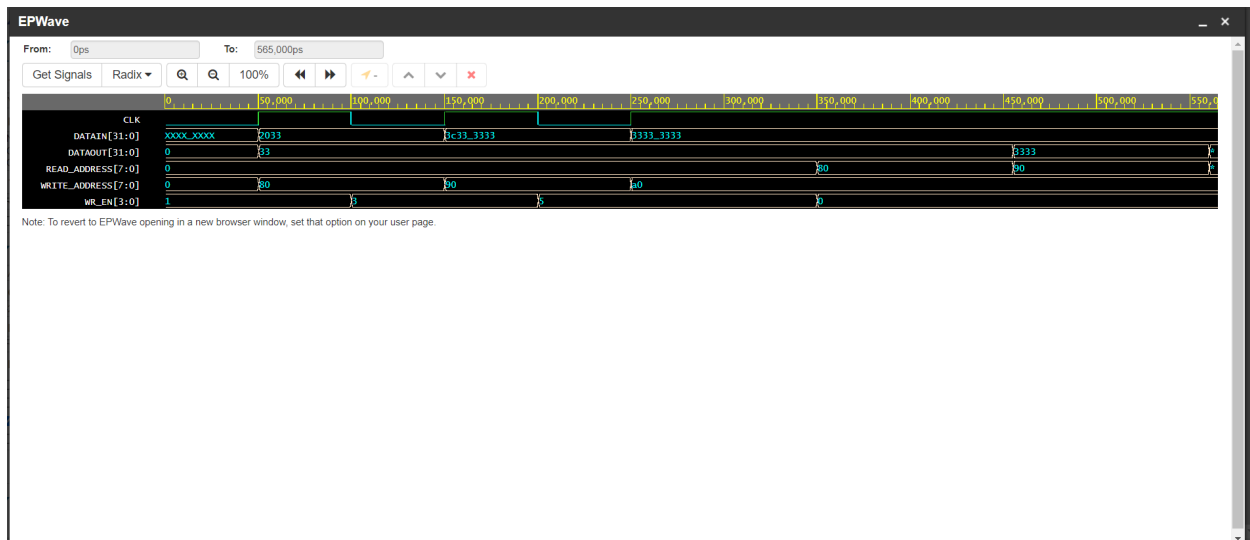


Figure 4: Data Memory EPWave

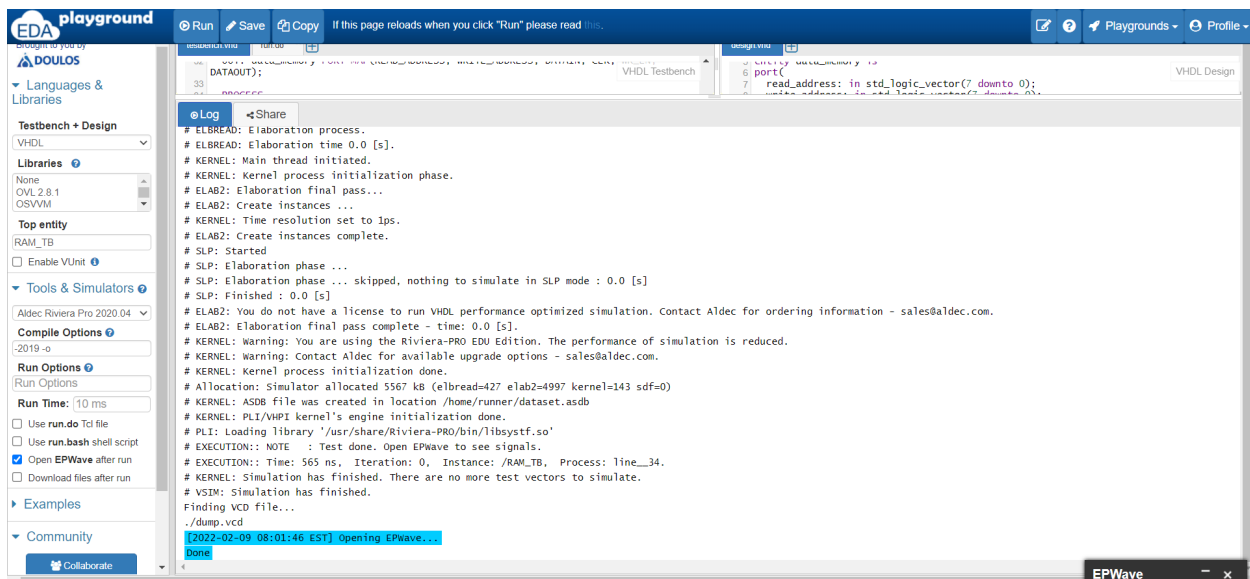


Figure 5: Data Memory Log

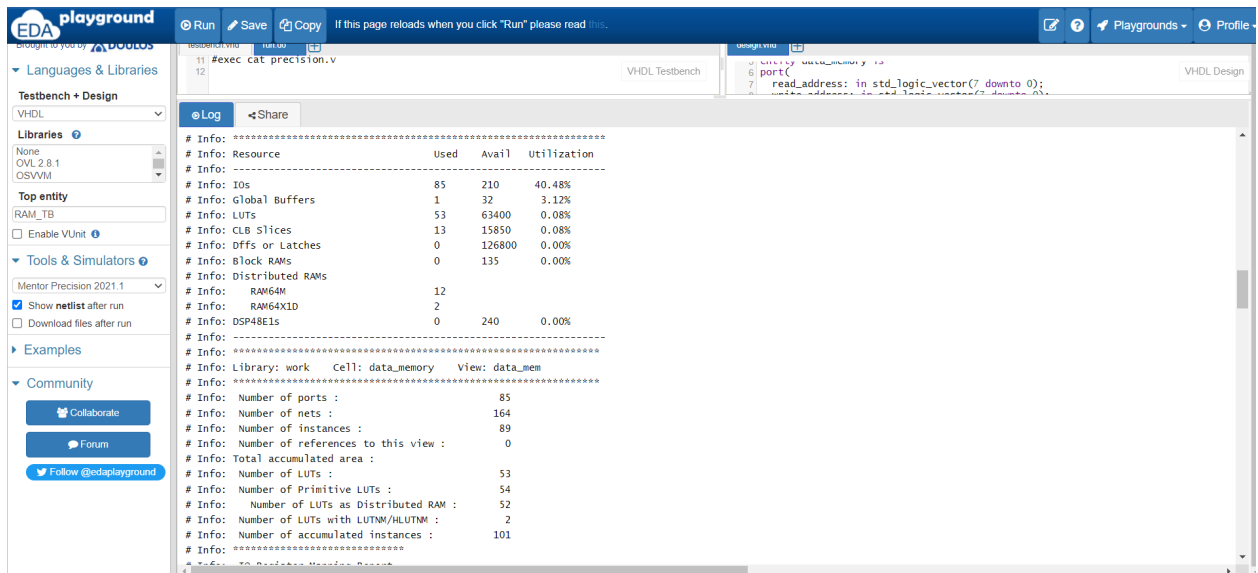


Figure 6: Data Memory Synthesis

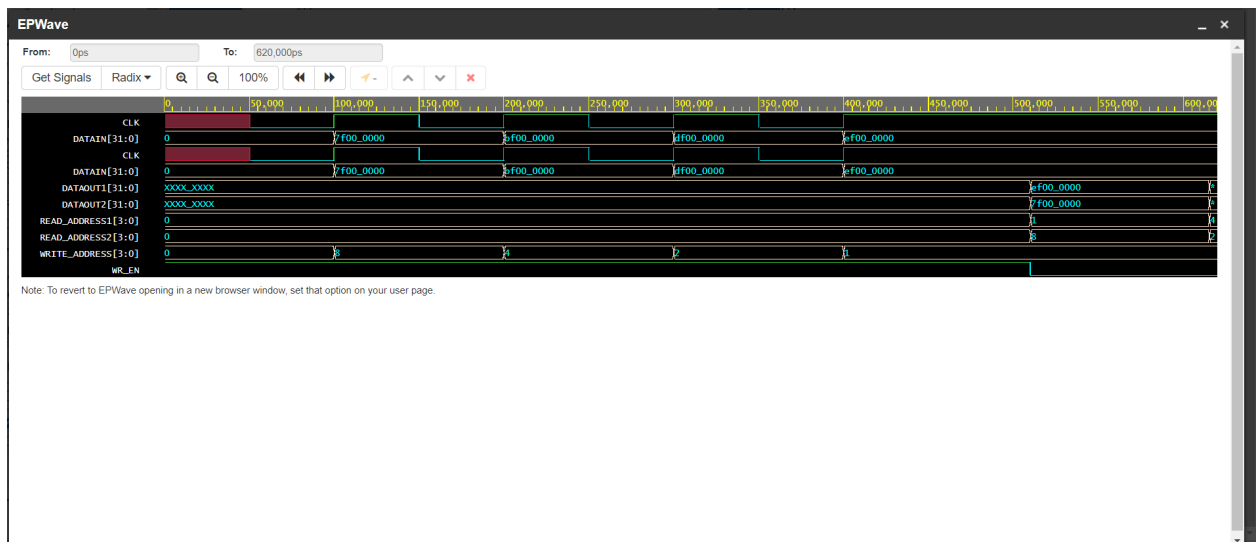


Figure 7: Register File EPWave

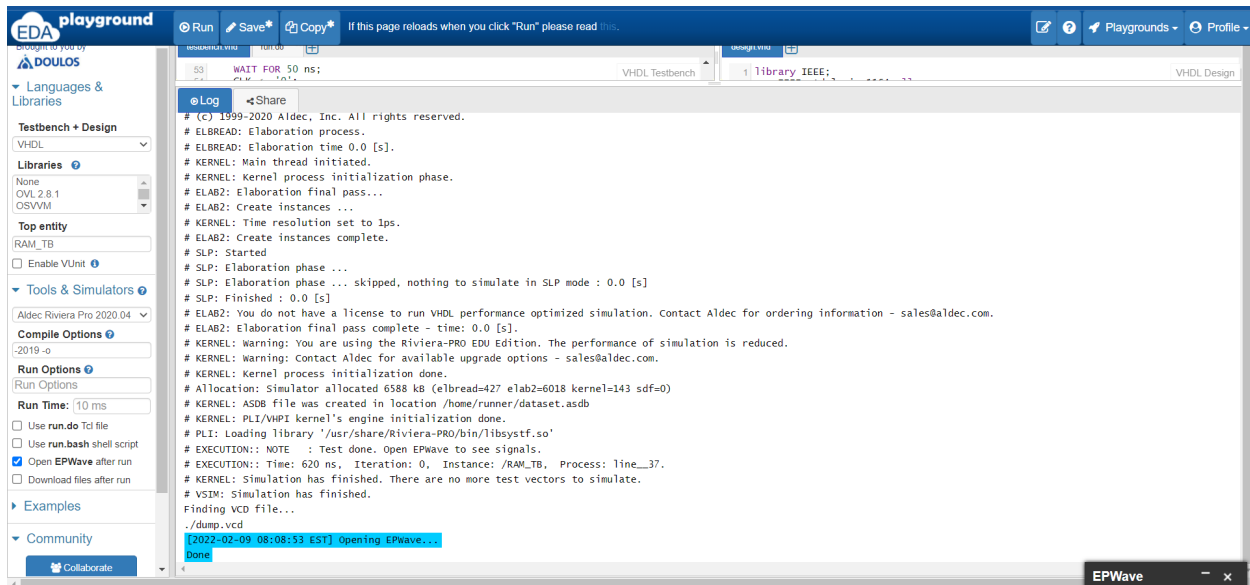


Figure 8: Register File Log

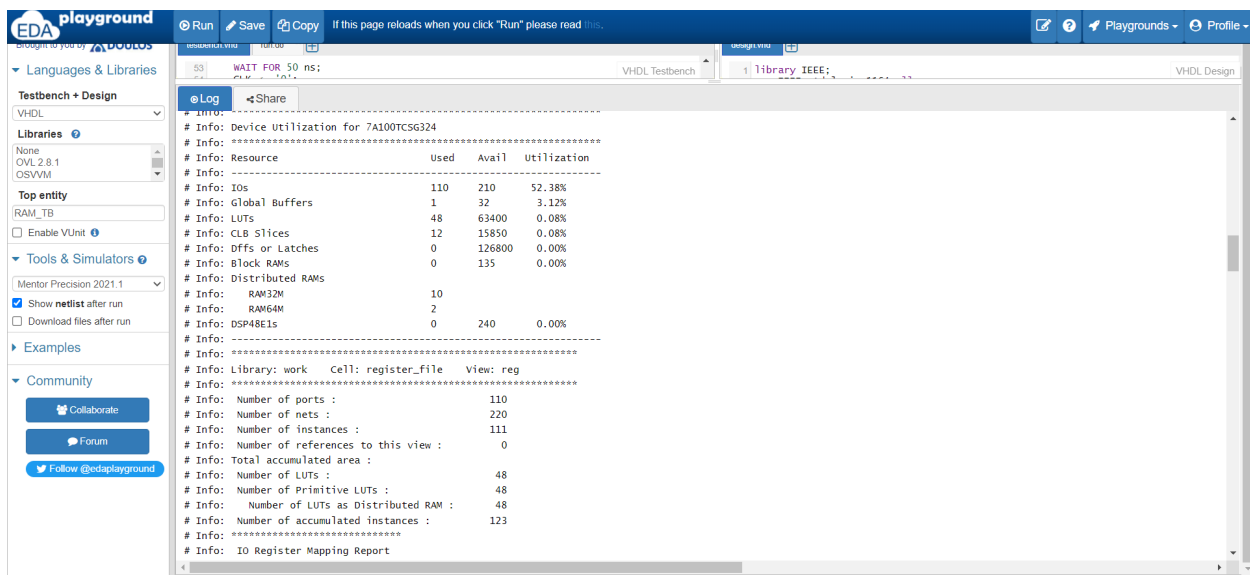


Figure 9: Register File Synthesis

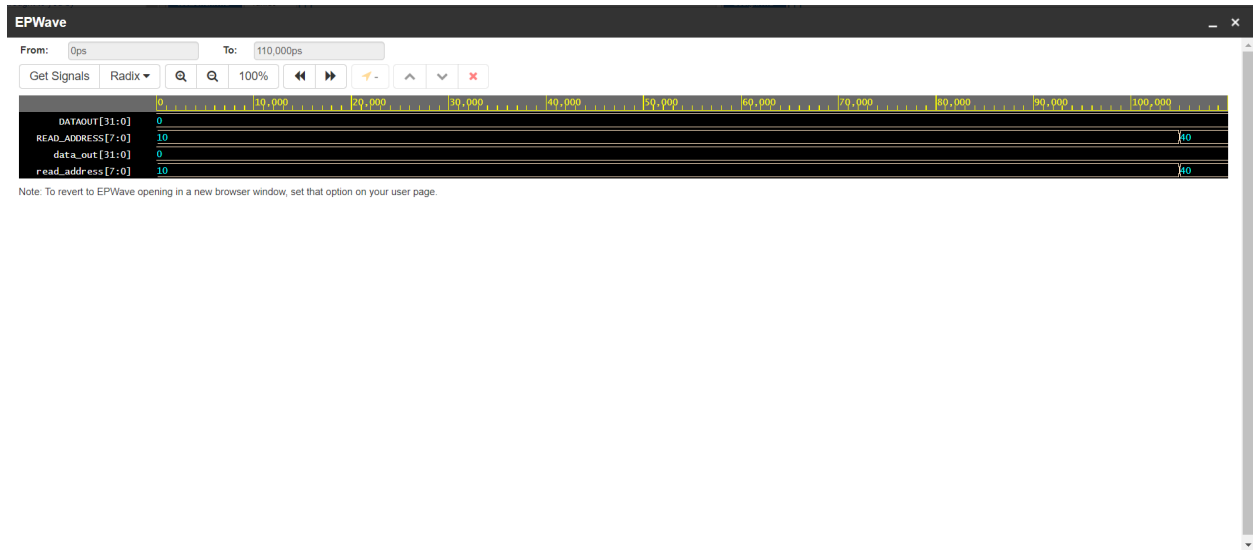


Figure 10: Program Memory EPwave

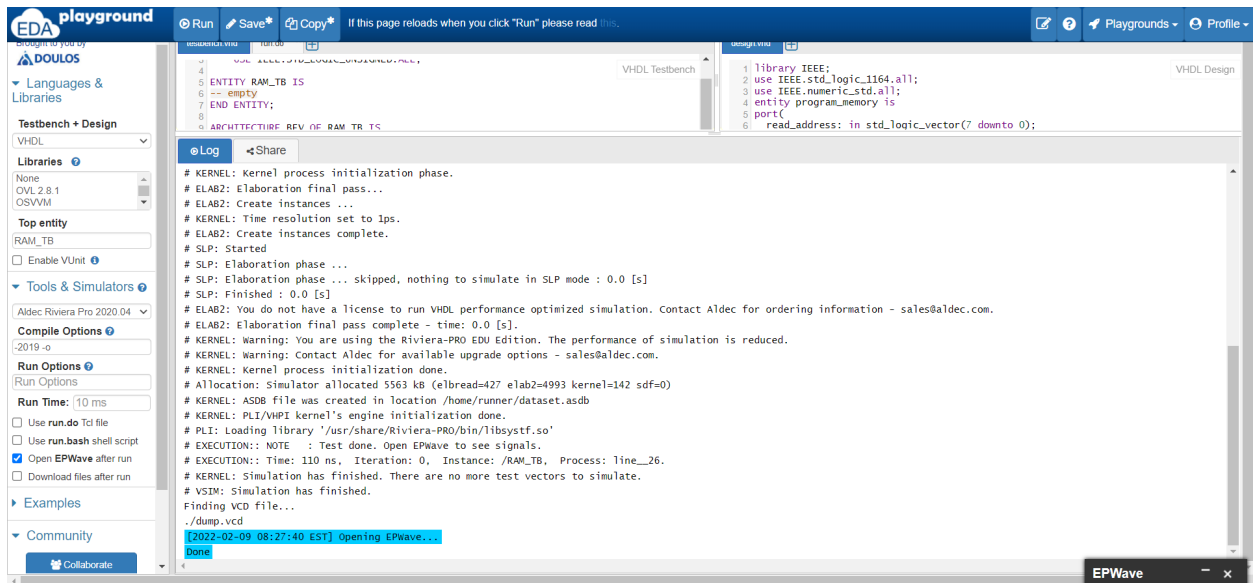


Figure 11: Program Memory Logs

The screenshot displays the EDA Playground web interface. On the left, the 'Languages & Libraries' sidebar is visible, showing 'Testbench + Design' selected, 'VHDL' as the language, and 'RAM_TB' as the top entity. The main workspace is divided into three panes: 'VHDL Testbench' (top left), 'VHDL Design' (top right), and a 'Log' pane (bottom) showing synthesis results.

VHDL Testbench Code:

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.numeric_std.all;
4 entity program_memory is
5   port(
6     read_address: in std_logic_vector(7 downto 0);
  
```

VHDL Design Code:

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.numeric_std.all;
4 entity program_memory is
5   port(
6     read_address: in std_logic_vector(7 downto 0);
  
```

Synthesis Log Output:

```

# Info: 9 /home/runner/impl_1/program_memory.xdc
# Info: 10 /home/runner/impl_1/program_memory.tcl
# Info: *****
# Info: Device Utilization for 7A100TCSG324
# Info: *****
# Info: Resource      Used      Avail      Utilization
# Info: -----
# Info: I/Os            40       210      19.05%
# Info: Global Buffers   0        32       0.00%
# Info: LUTs             0      63400     0.00%
# Info: CLB Slices       0     15850     0.00%
# Info: DFFs or Latches   0    126800     0.00%
# Info: Block RAMs       0       135     0.00%
# Info: DSP48E1s        0       240     0.00%
# Info: *****
# Info: *****
# Info: Library: work Cell: program_memory View: prog_mem
# Info: *****
# Info: Number of ports :           40
# Info: Number of nets :           33
# Info: Number of instances :        33
# Info: Number of references to this view :    0
# Info: Total accumulated area : unknown
# Info: *****
# Info: IO Register Mapping Report
# Info: *****
  
```

Figure 12: Program Memory Synthesis