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 unclocked (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.

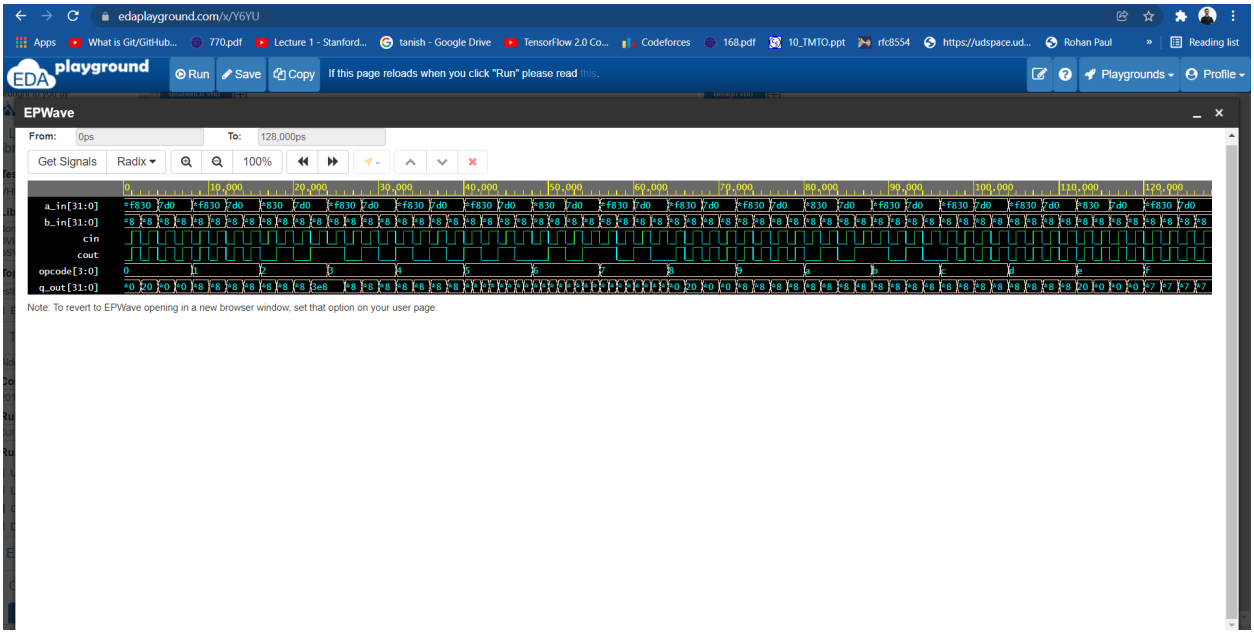

Figure 1: ALU EPWave

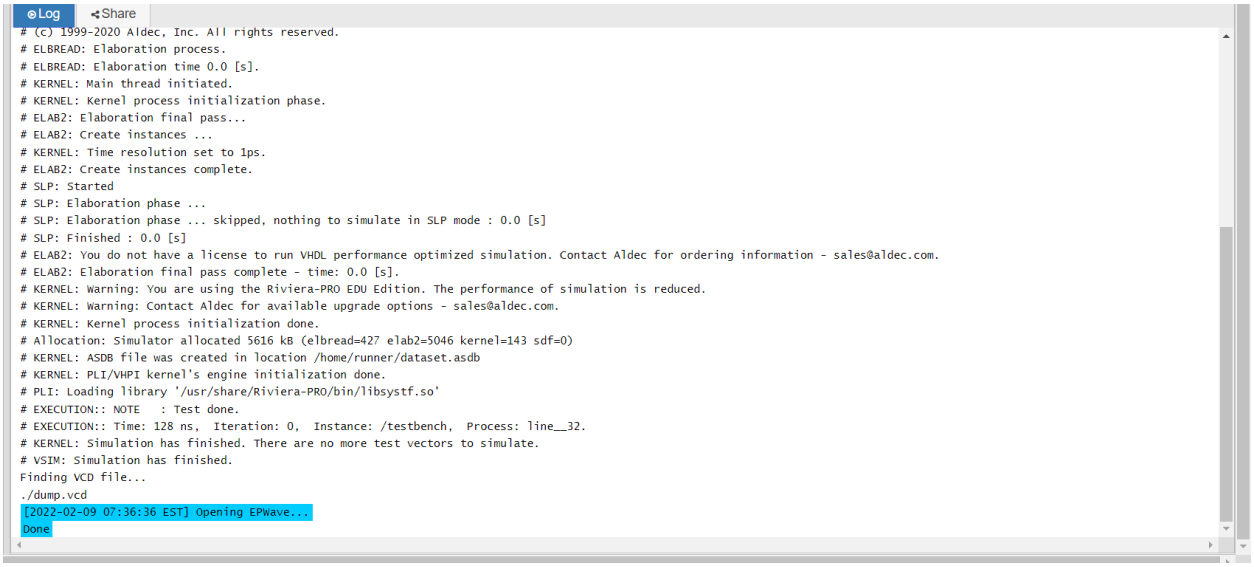

Figure 2: ALU Logs

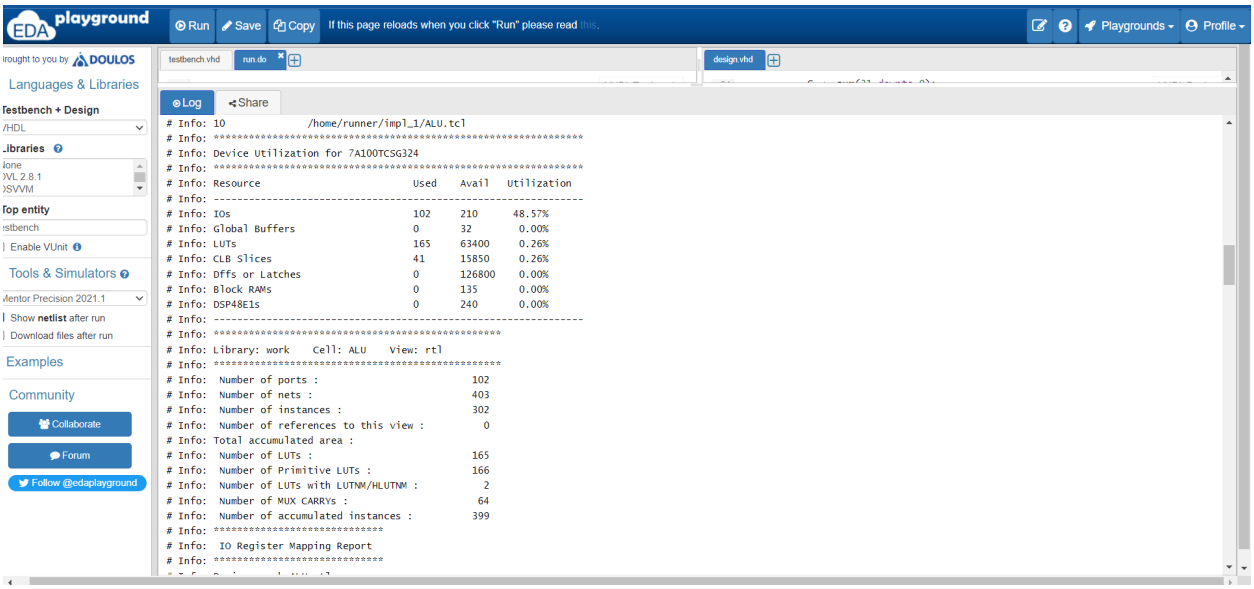

Figure 3: ALU$_{synthesis}$

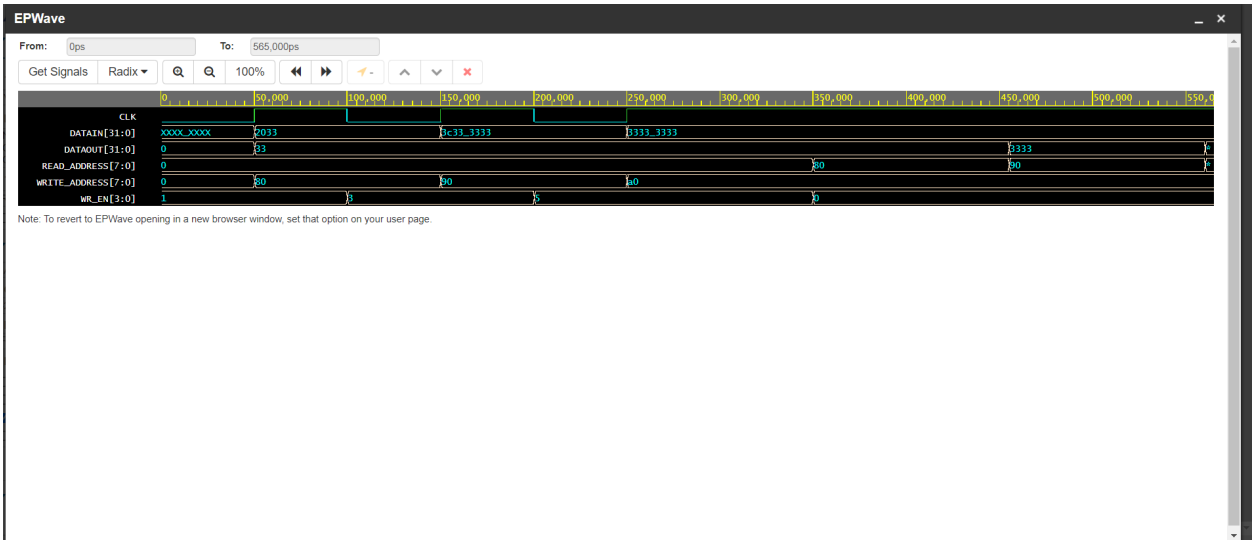

Figure 4: Data Memory EPWave

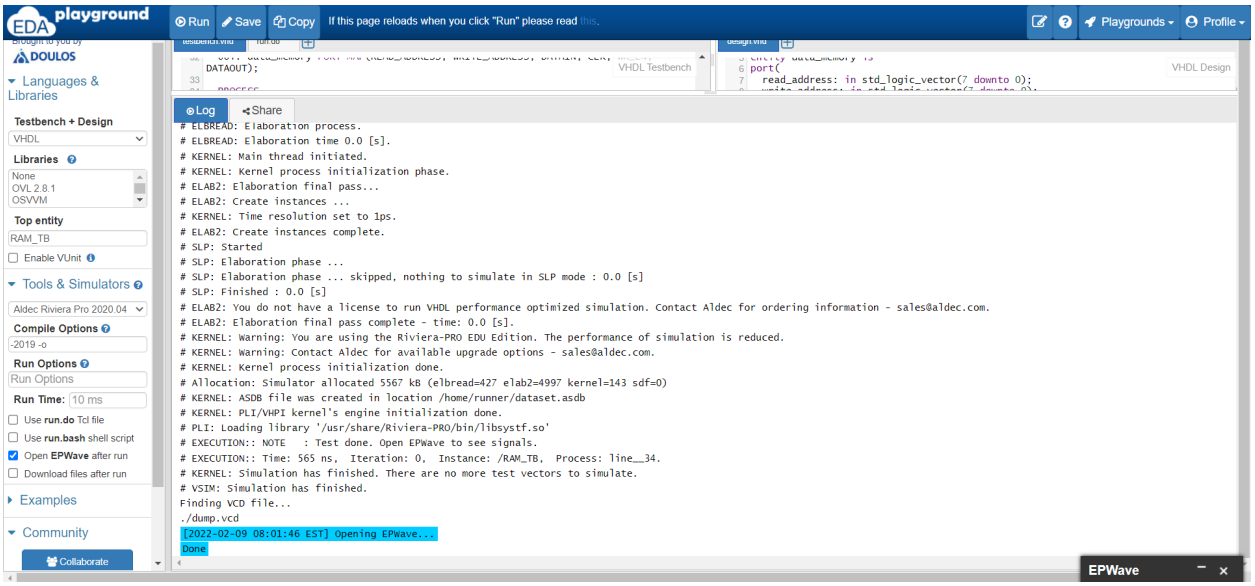

Figure 5: Data Memory Log

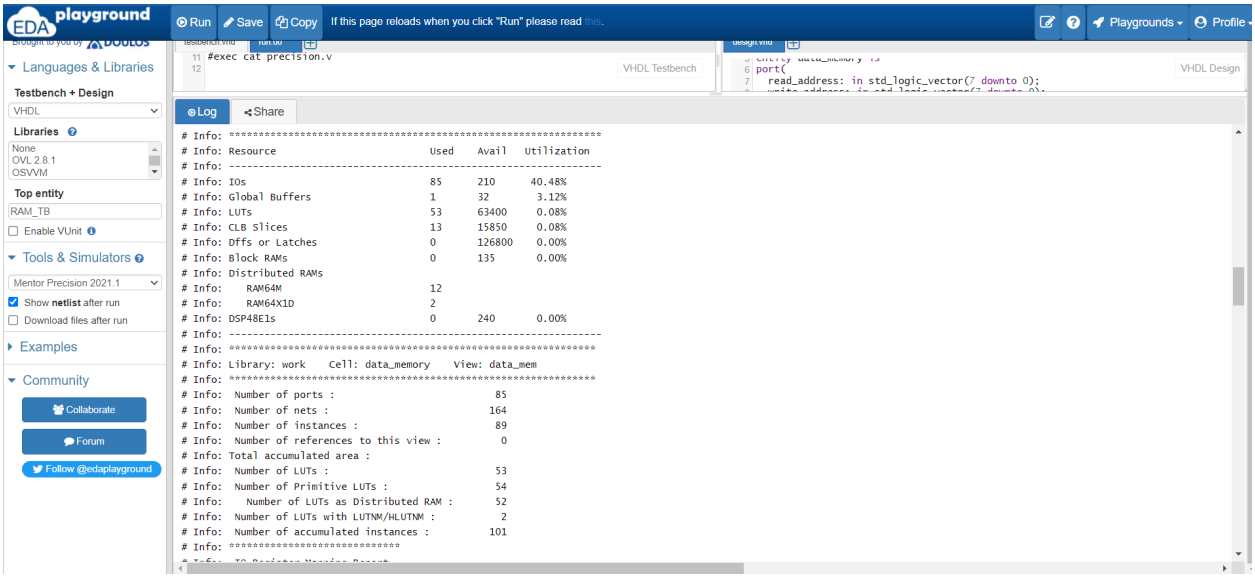

Figure 6: Data Memory Synthesis

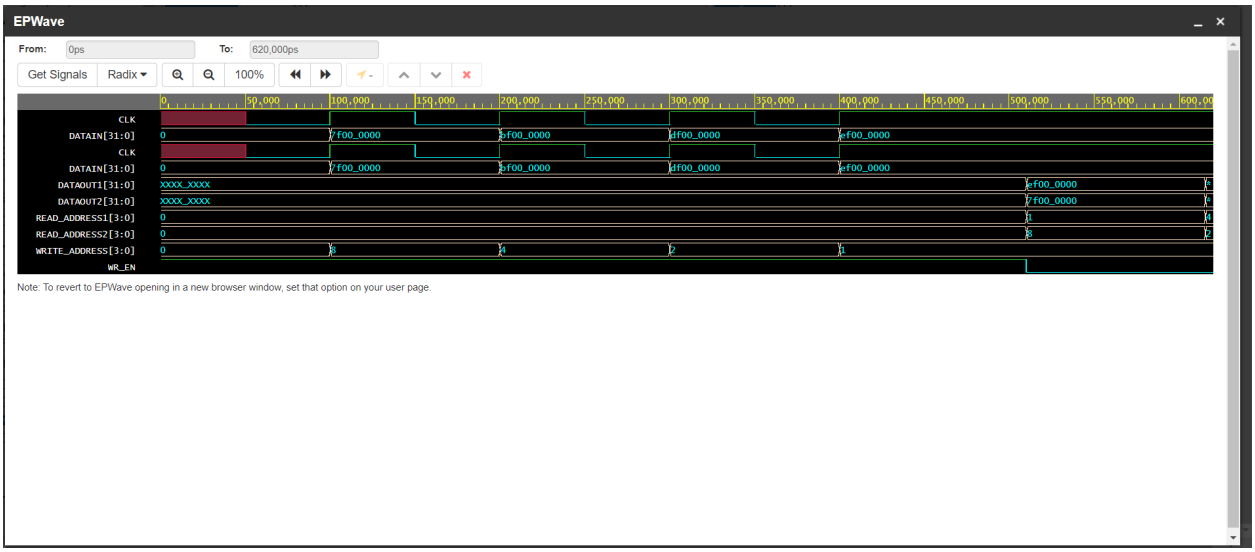

Figure 7: Register File EPWave

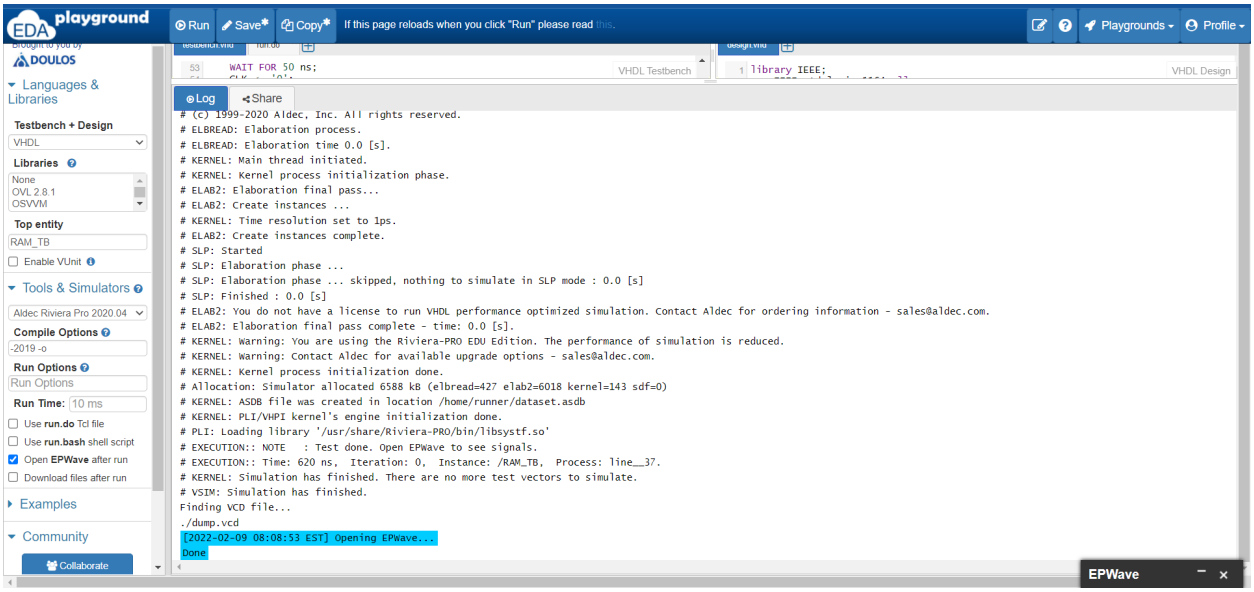

Figure 8: Register File Log

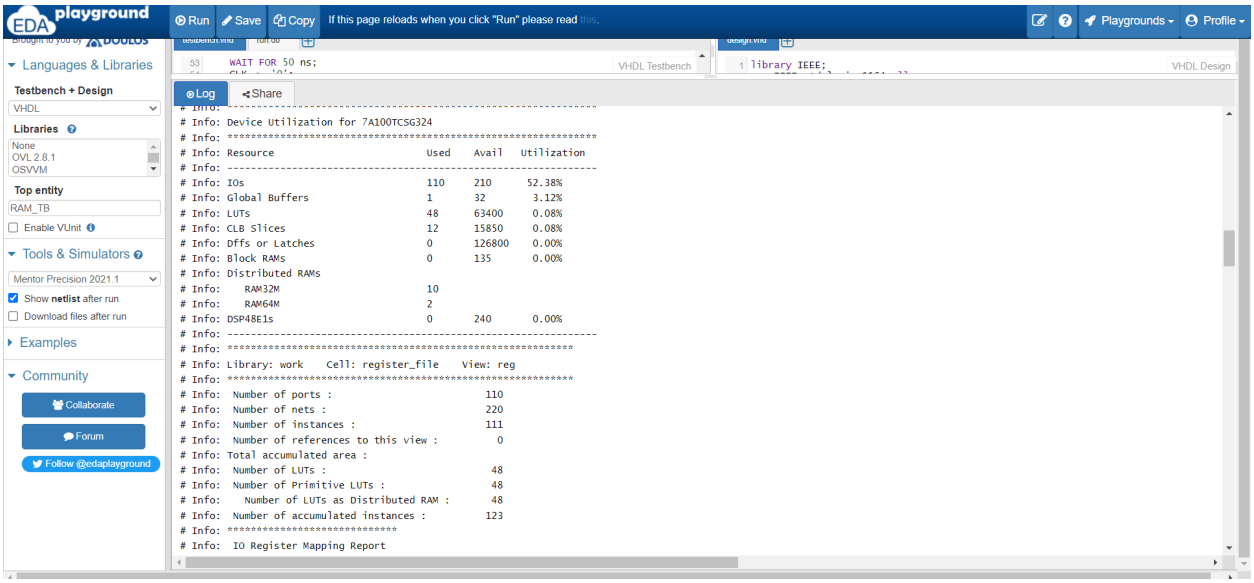

Figure 9: Register File Synthesis

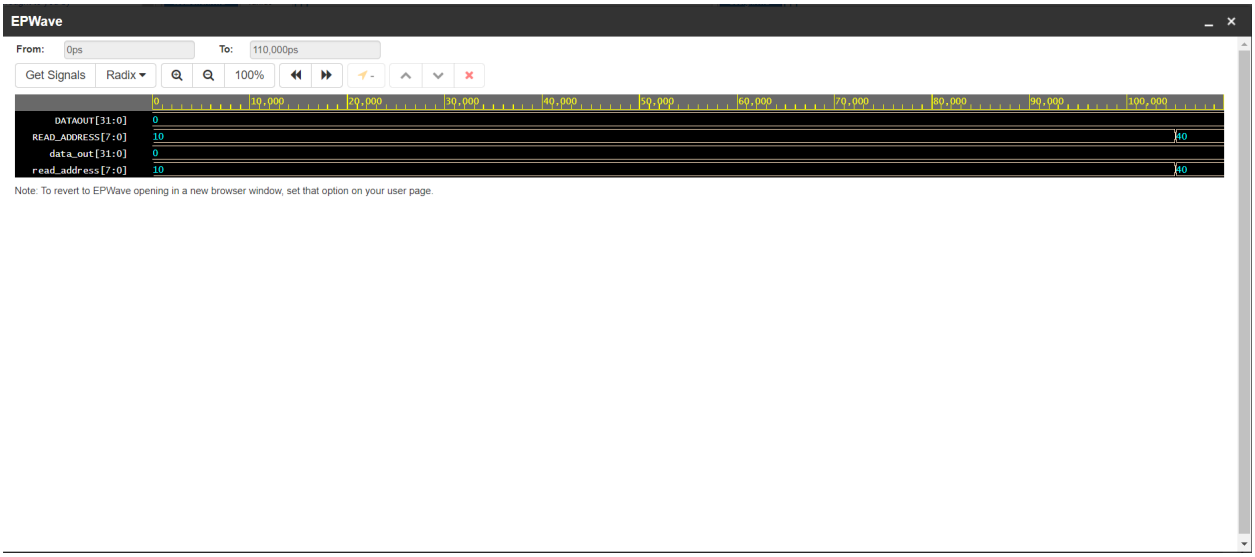

Figure 10: Program Memory EPwave

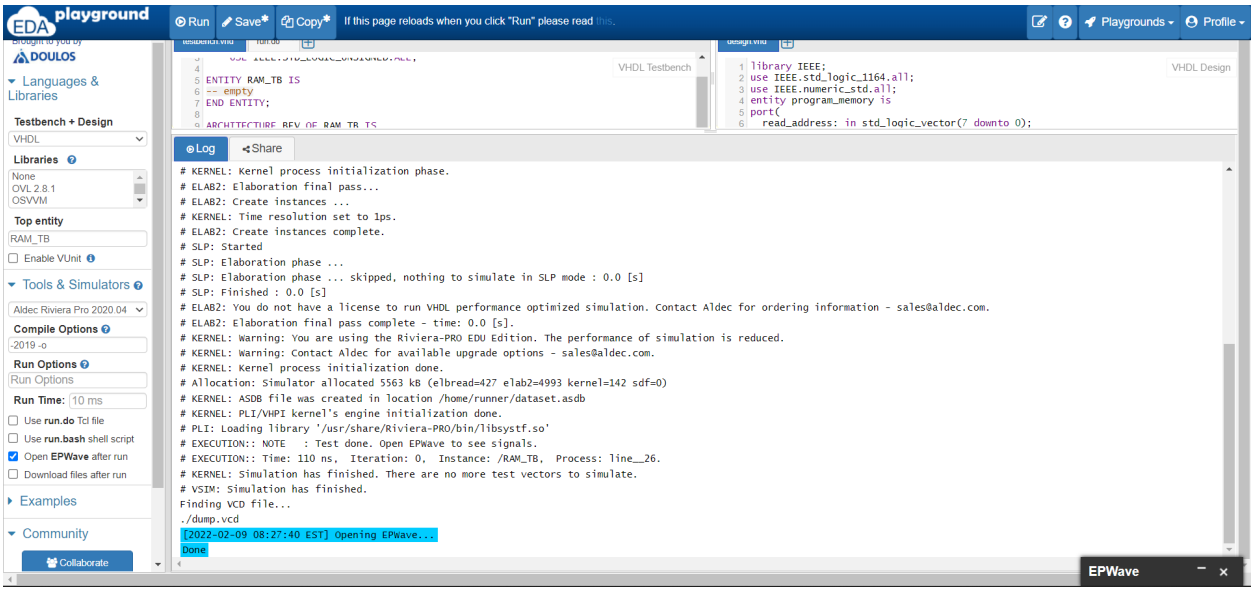

Figure 11: Program Memory Logs

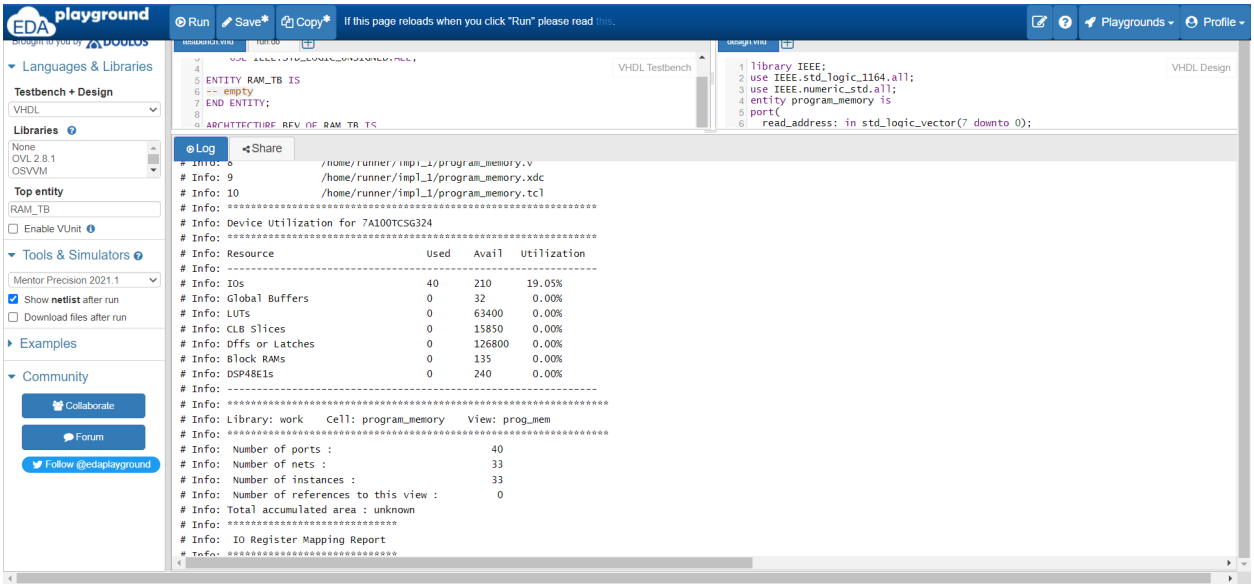

Figure 12: Program Memory Synthesis