11/03/2019

EE446 LABORATORY

EXPERIMENT 1

PRELIMINARY REPORT

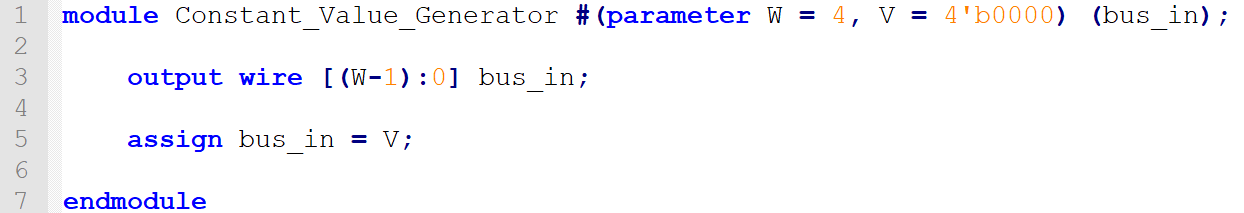
Muttalip Caner TOL

2031466

Tuesday Afternoon

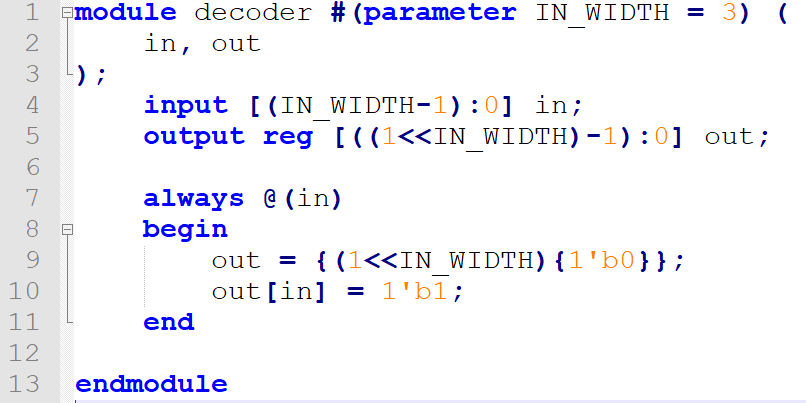
**1.2. Module Design with Verilog HDL**

**1.2.1. Constant Value Generator**

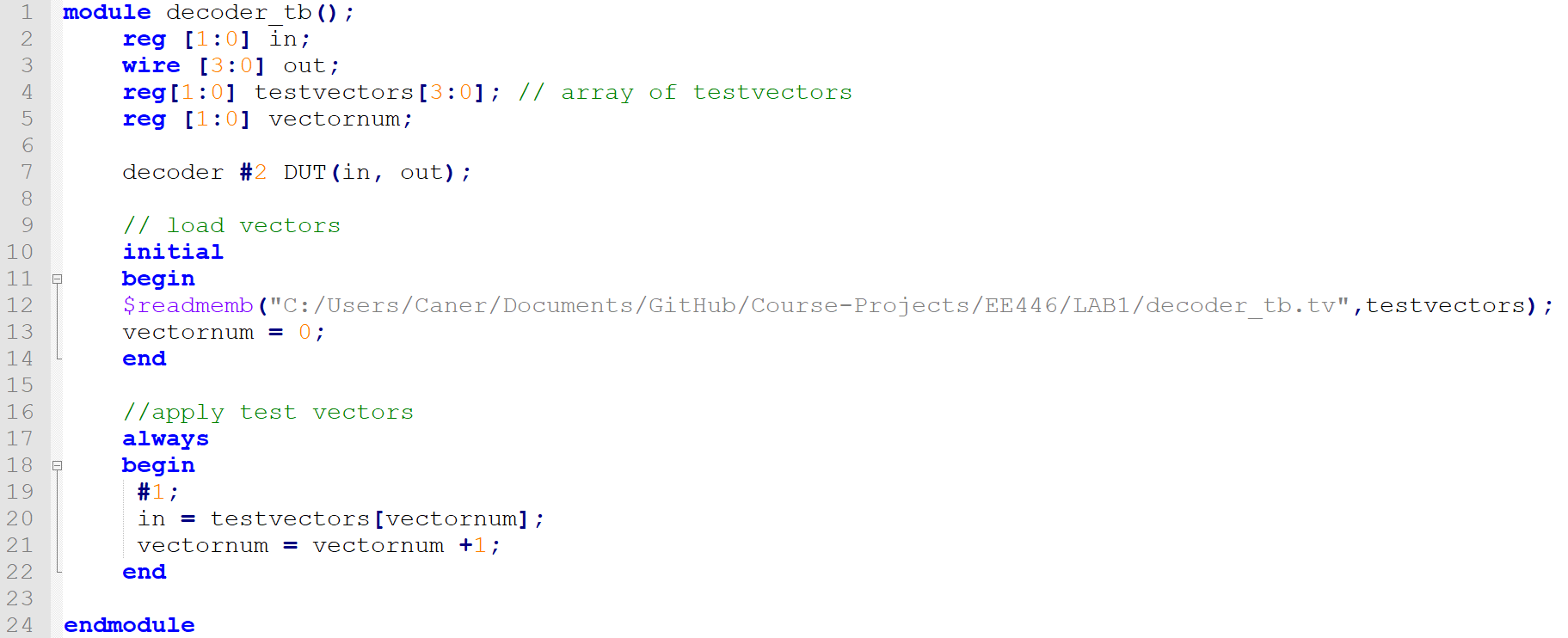


**1.2.2. Decoder**

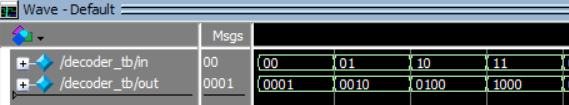
1. Implementation



1. Test bench module

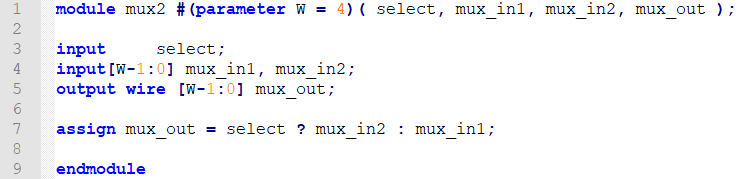


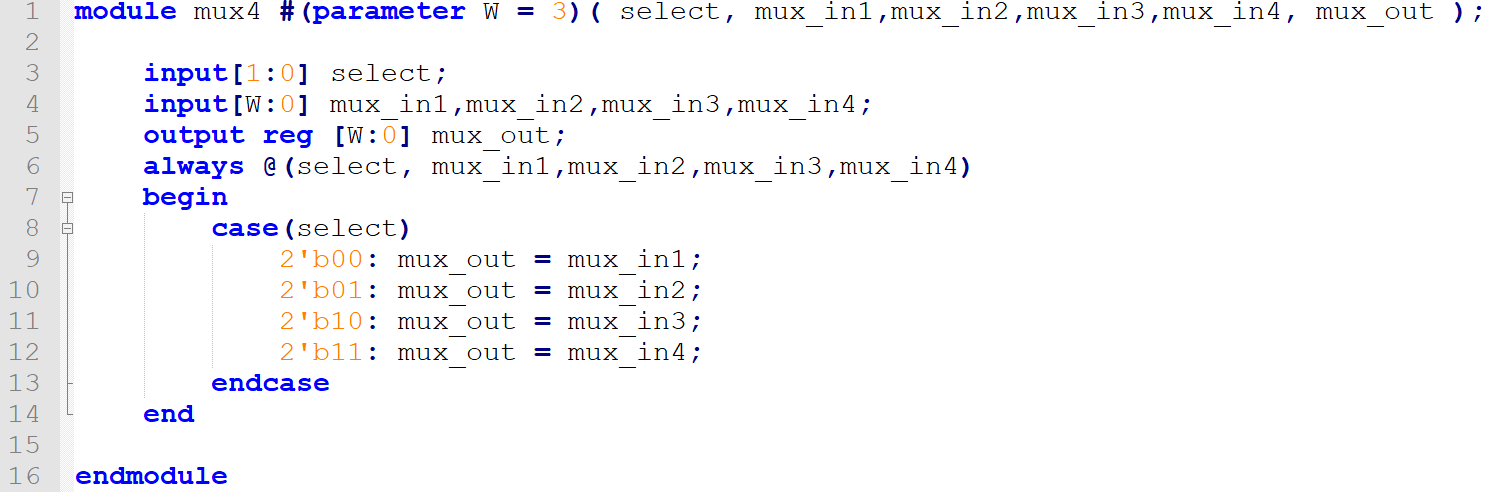
1. Vector table  
   
2. Verification



**1.2.3. Multiplexers**

1. Implementations

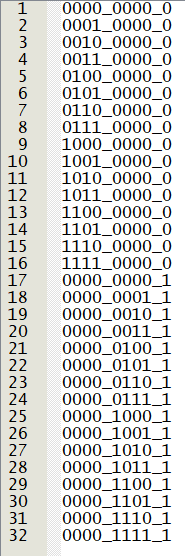
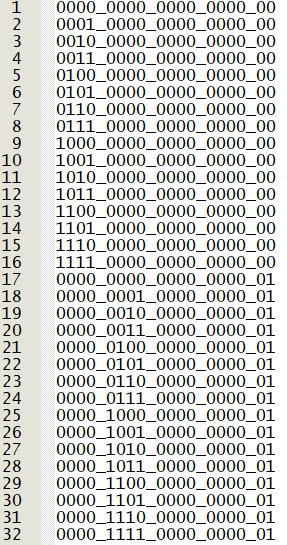
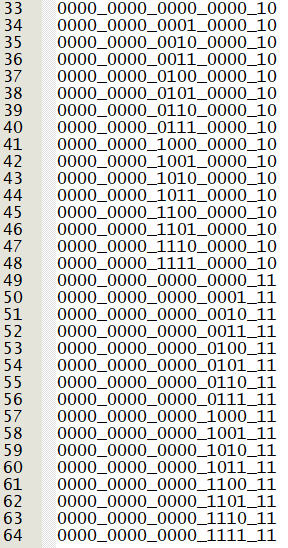




1. Test bench modules

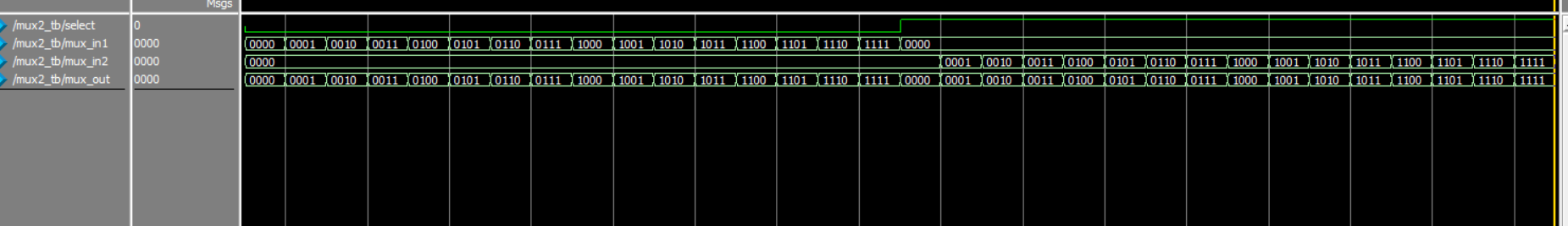


1. Vector tables  
   Test vectors of mux2 and mux4:

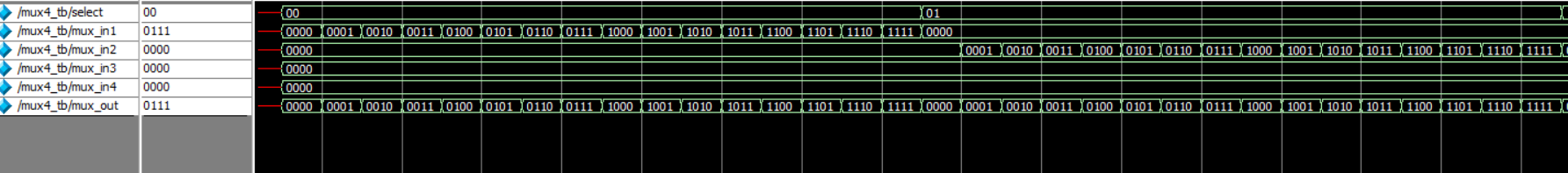
  

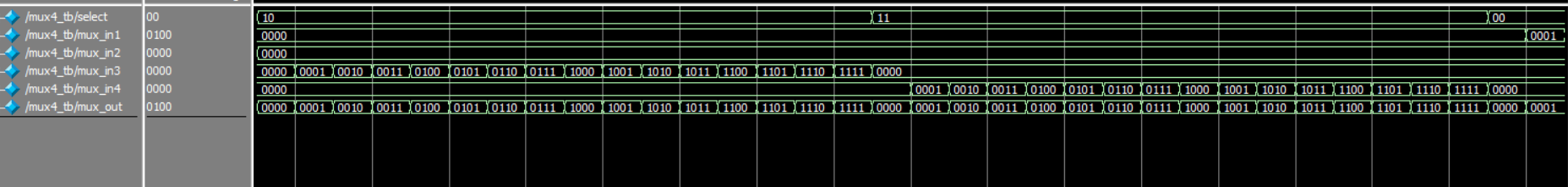
1. Verification

Simulation results for mux2:



Simulation results for mux4:





**1.2.4. Arithmetic Logic Unit (ALU)**

1. Implementation



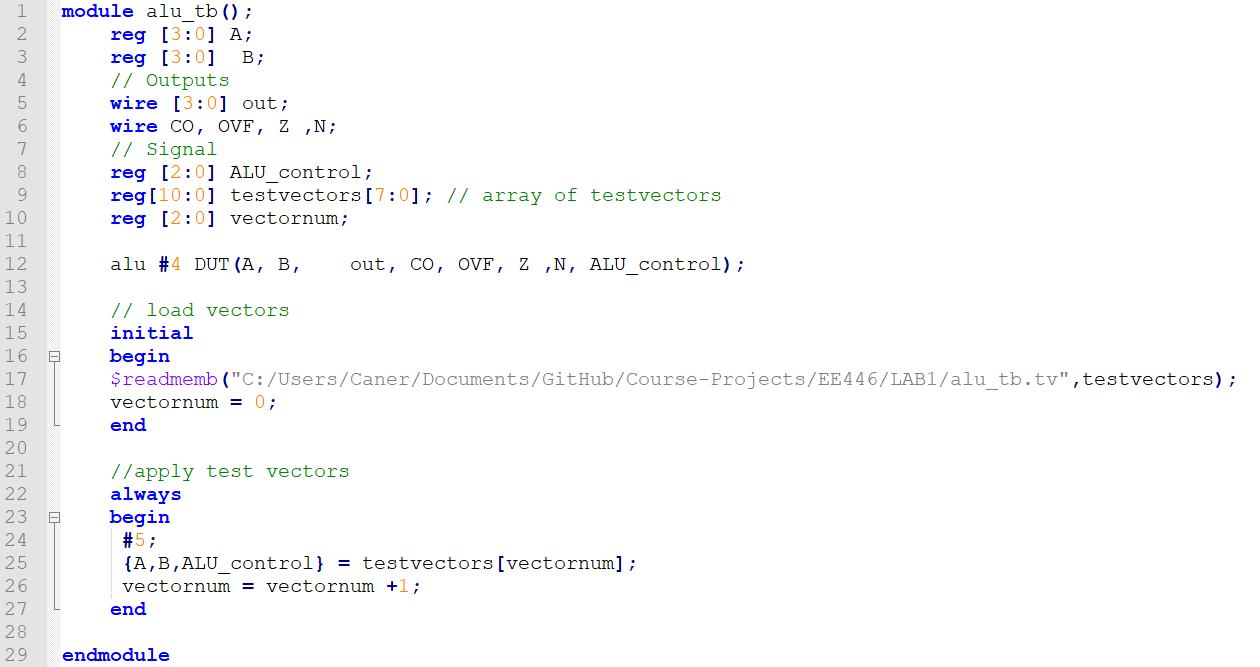


1. Overflow detection

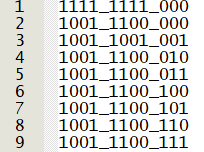
I have put checkpoints on the arithmetic operations. Overflow flag is set when:

* + adding two positives yields a negative
  + or, adding two negatives gives a positive
  + or, subtract a negative from a positive gives a negative
  + or, subtract a positive from a negative gives a positive

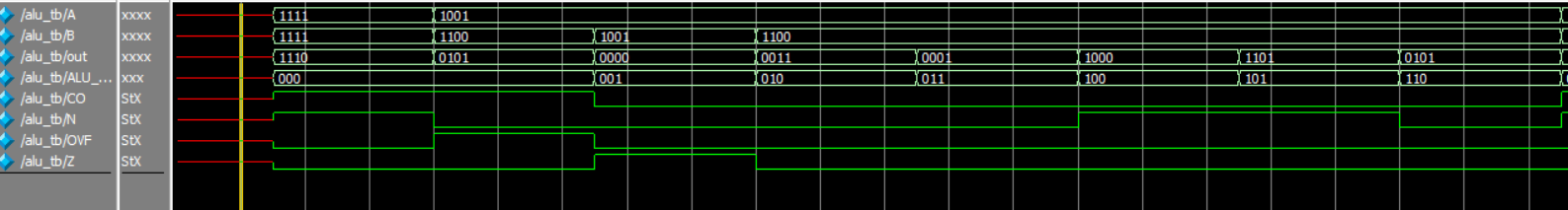
1. Test bench module



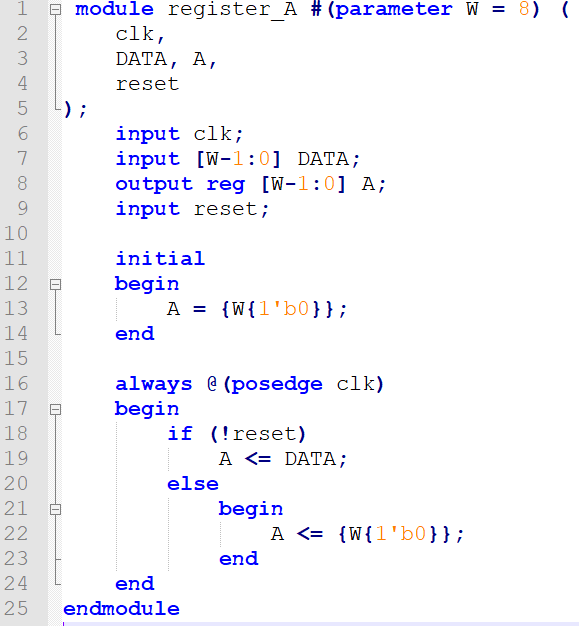
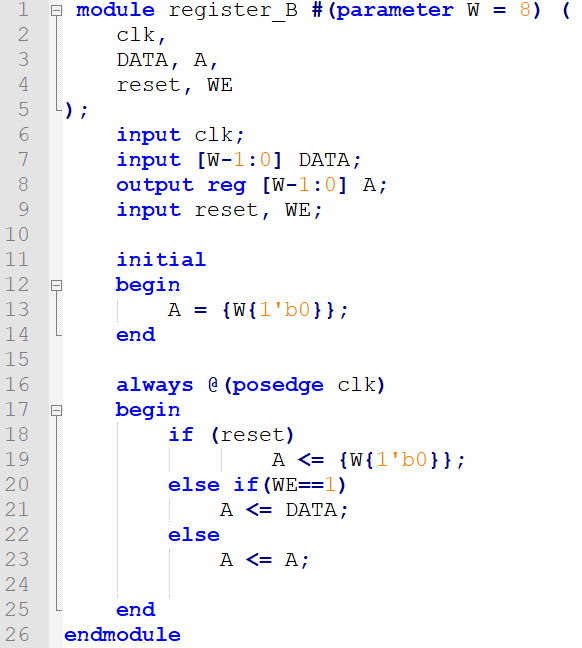
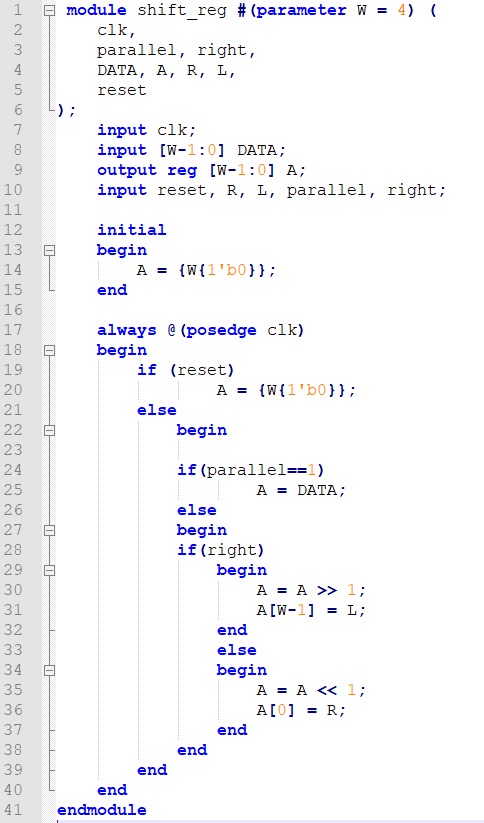
1. Vector table



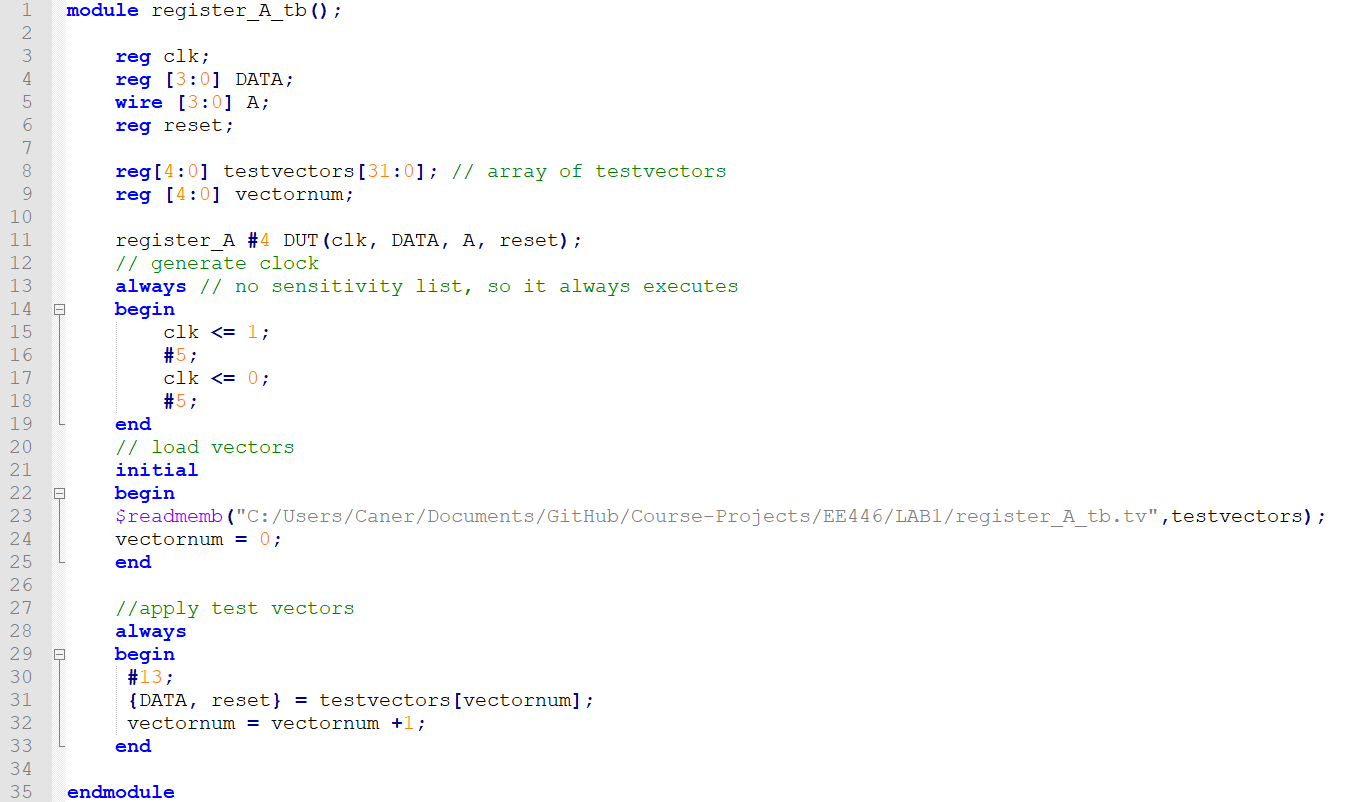
1. Verification



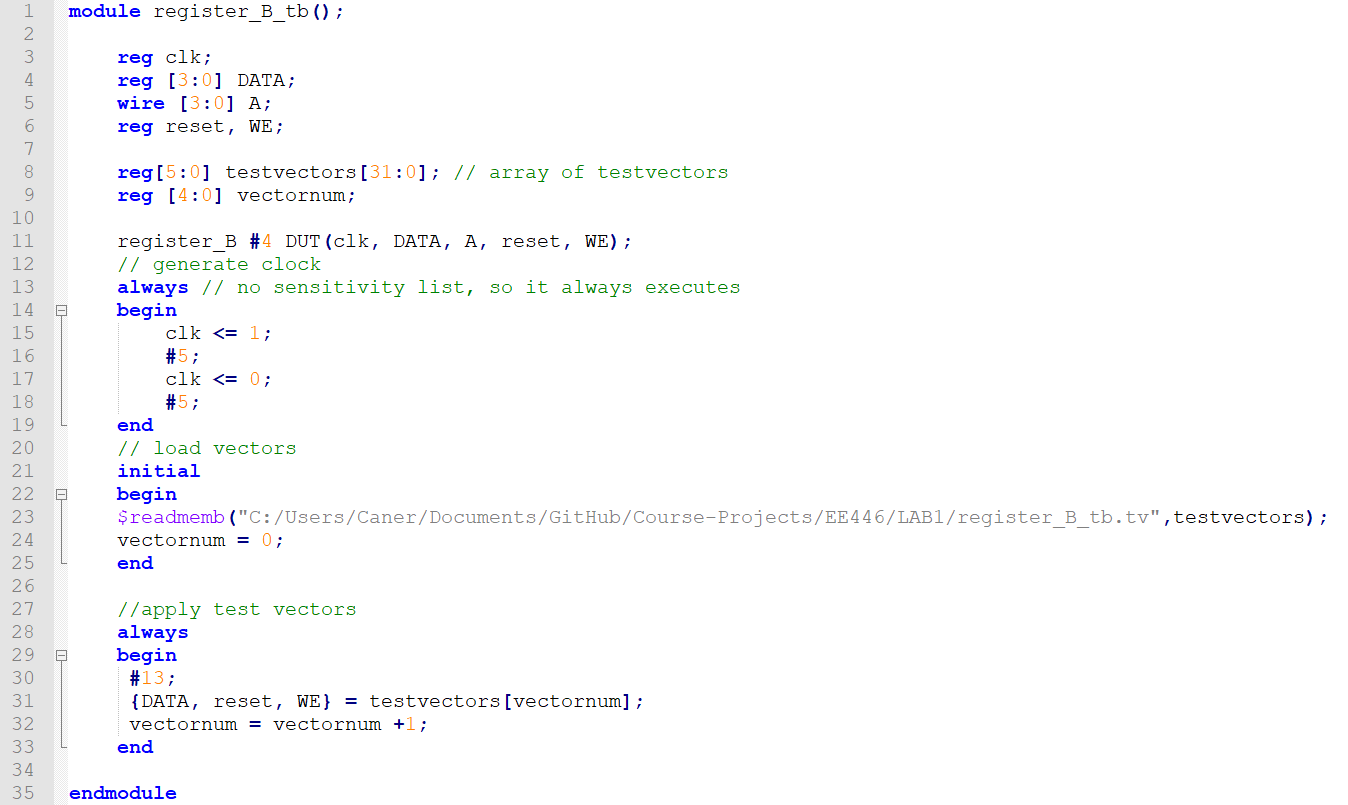
**1.2.5. Registers**

1. Simple register with synchronous reset  
   
2. Register with synchronous reset and write enable  
   
3. Shift register with parallel and serial load  
   
4. Test bench modules

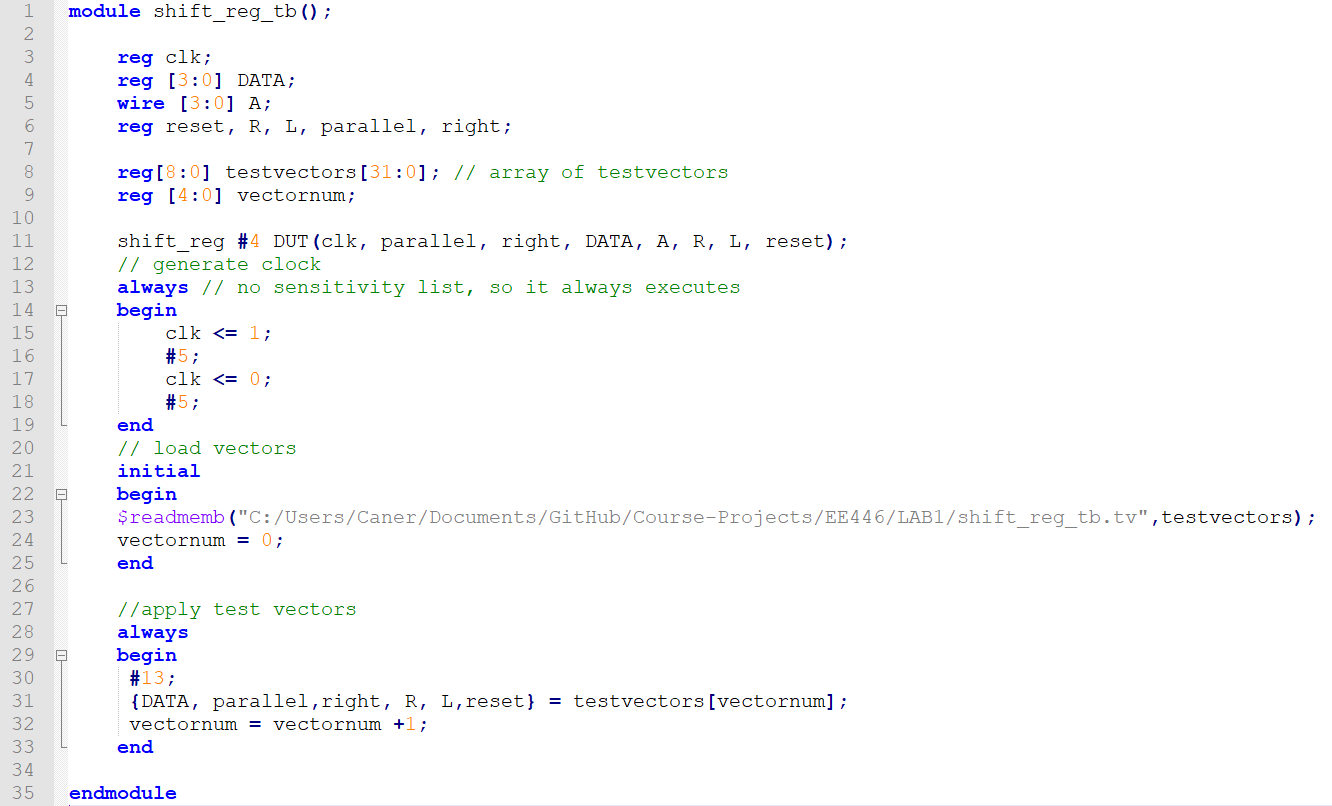
For Simple Register with synchronous reset:



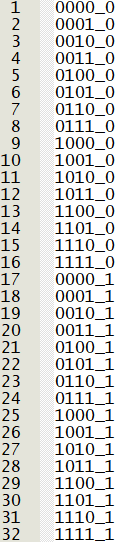
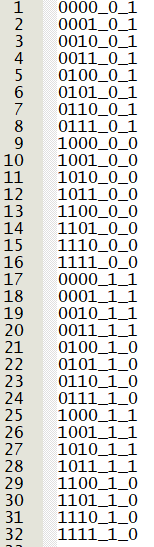
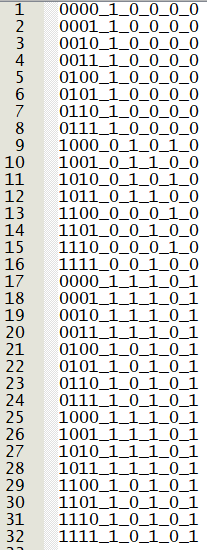
For Register with synchronous reset and write enable:



For Shift register with parallel and serial load:

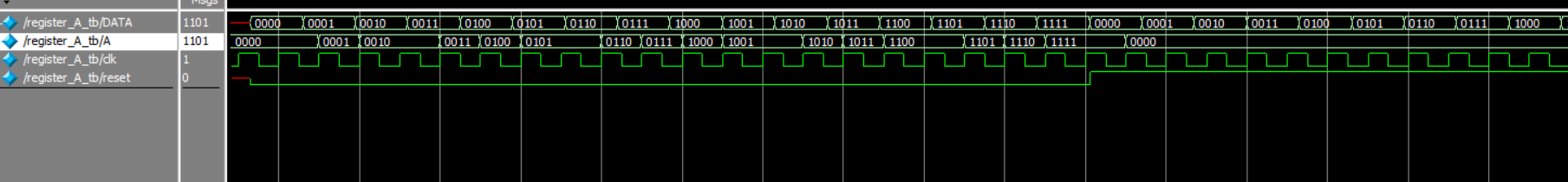


1. Vector tables

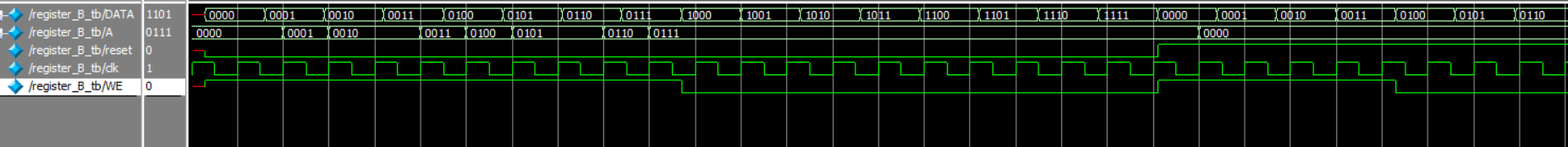
  

1. Verification

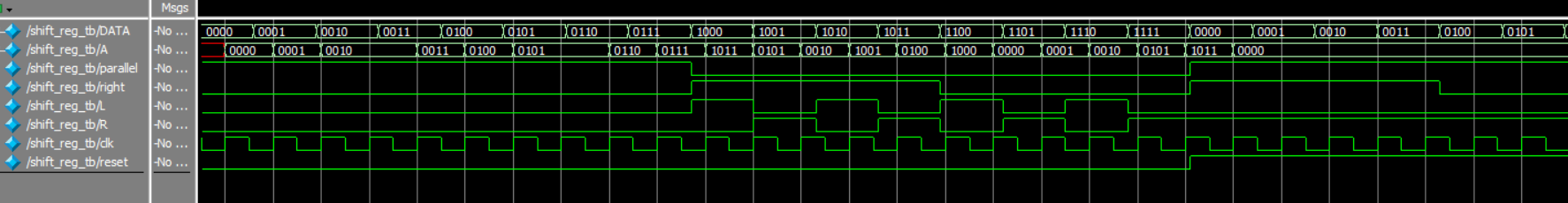
Simulation of Simple register with synchronous reset



Simulation of Register with synchronous reset and write enable

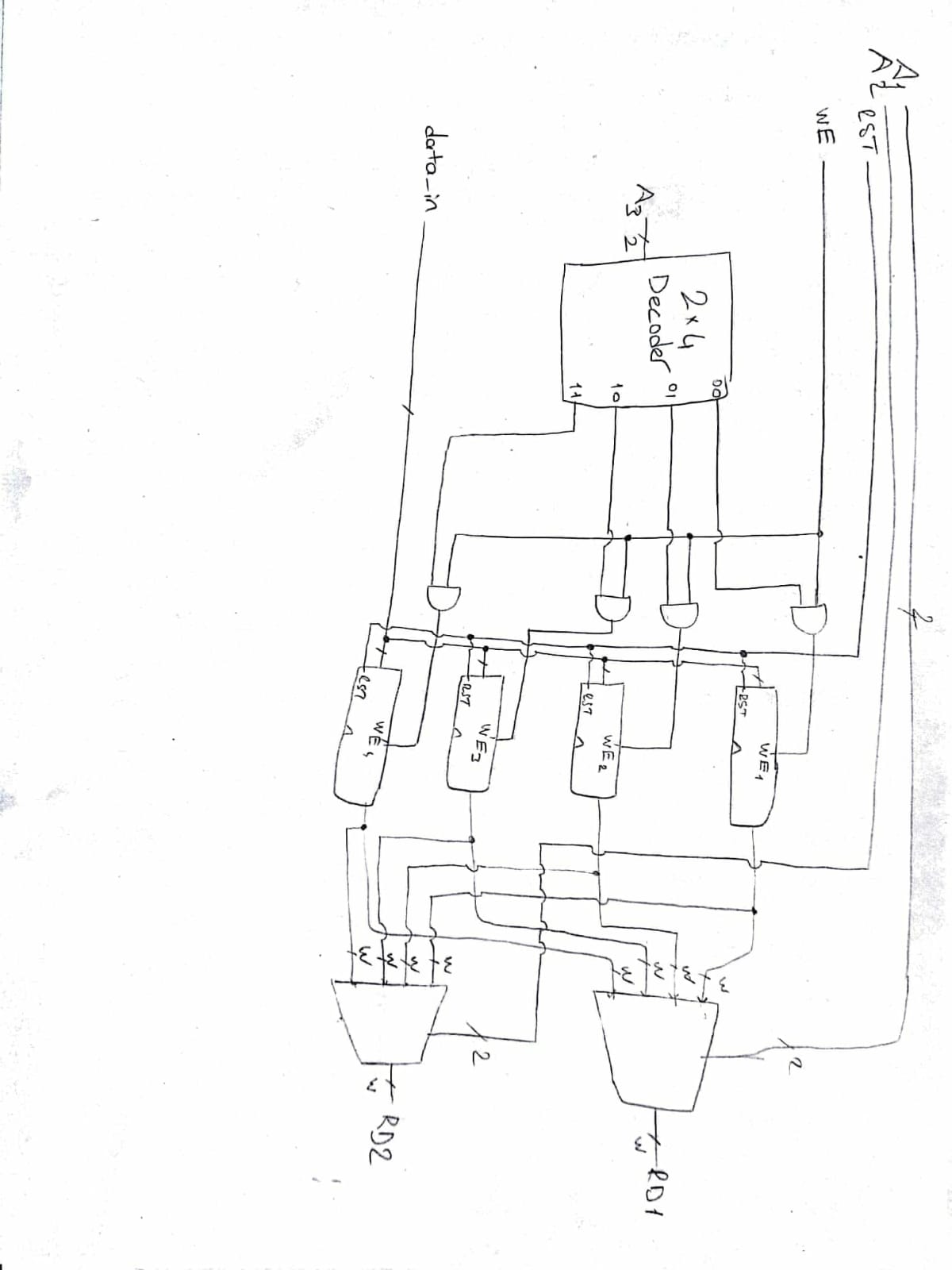


Simulation of Shift register with parallel and serial load

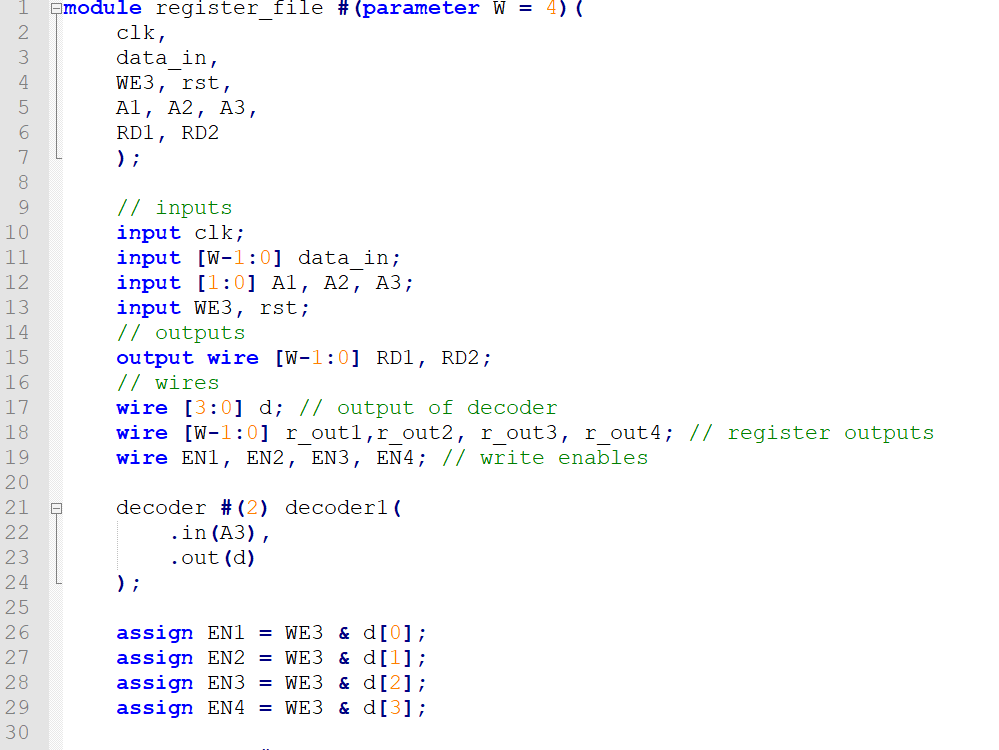


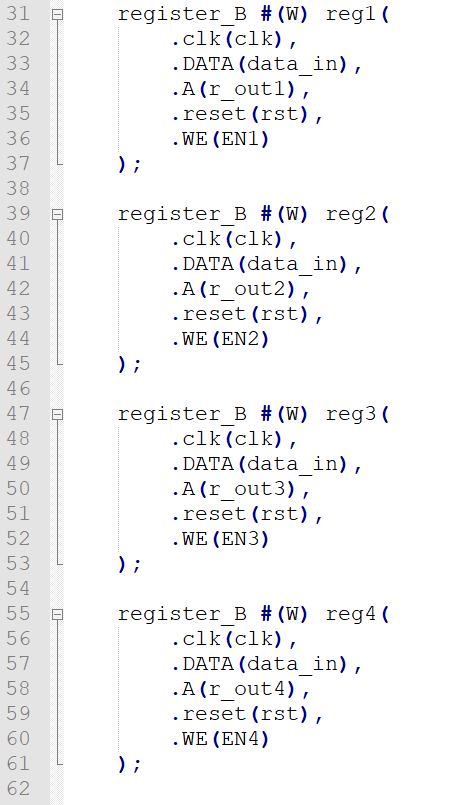
**1.3. Register File**

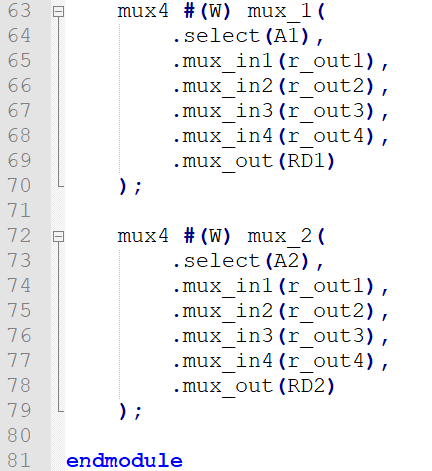
1. Design



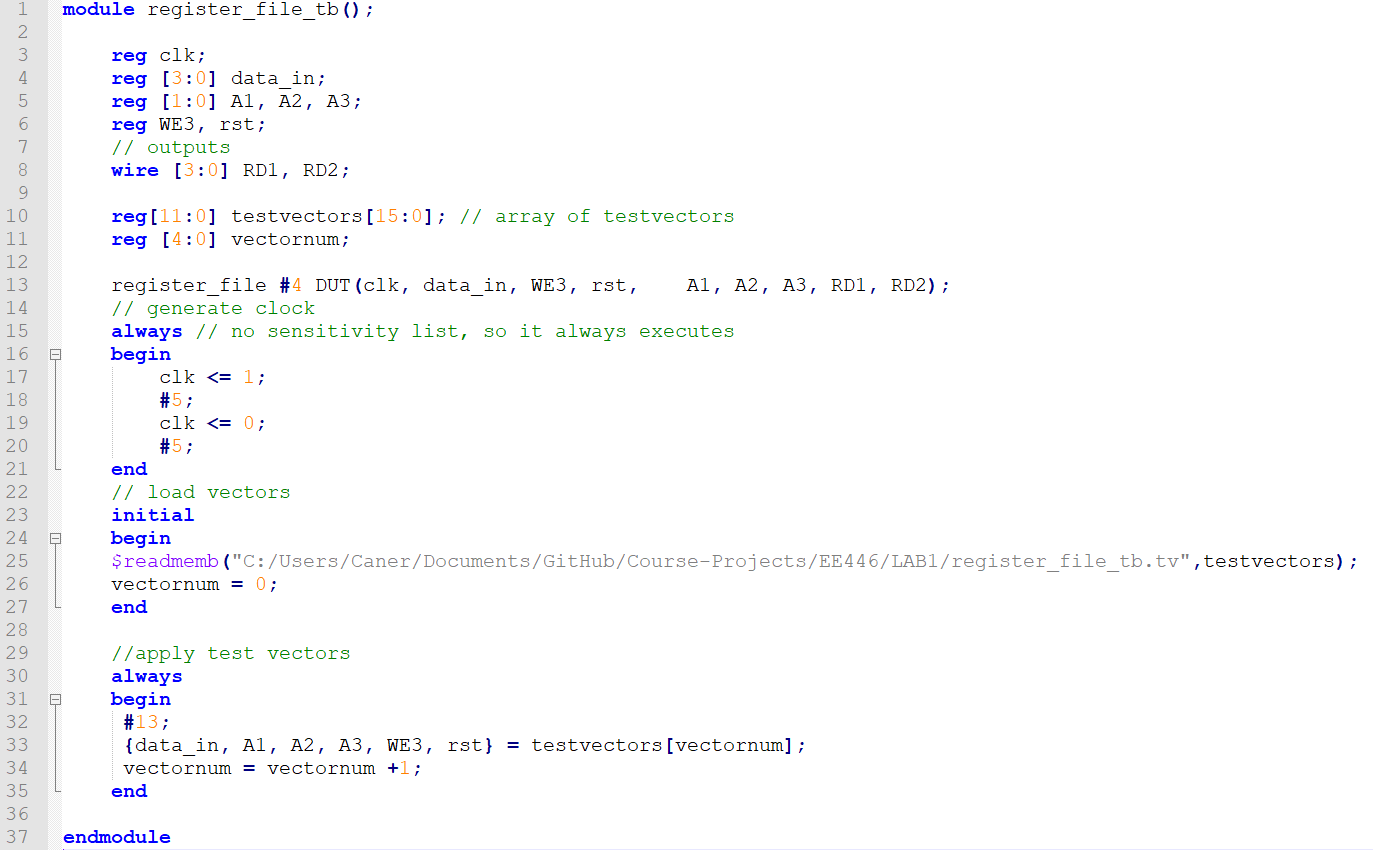
1. Implementation



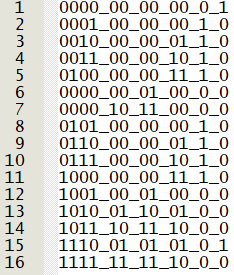




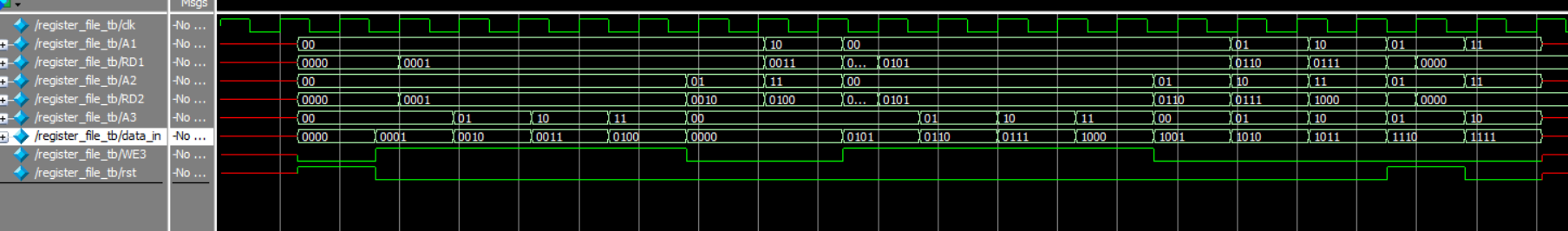
1. Test bench module



1. Vector table



1. Verification



**1.4. Datapath Design for an Architecture**

* How many control pins for the control signals does your architecture have?

R1Src, ASrc, ALUControl[2:0], Acc\_parallel, Acc\_Lsrc[1..0], Q\_right and the reset signals of the four registers.

There are 9 control pins.

* How many different control signals does your architecture use to perform the desired tasks?

R1Src, ASrc, ALUControl[2:0], Acc\_parallel, Acc\_Lsrc[1..0], Q\_right

6 control signals.

* Can you reduce the number of the control pins? Why not or how?

We can ignore the reset pins of the registers.

* Write down the sequence of the control signals for REVERSED LOAD operation. How many cycle does this operation take?

1st Clock: ASrc = 0, ALUControl = 00

R0 <- DATA

2nd Clock: Acc\_parallel = 1

Acc <- DATA + 0x0

3rd Clock: Acc\_parallel = 0, Acc\_Lsrc[1..0] = 00, Q\_right=1

Acc[3] <- Q[0]

Q[3] <- Acc[0]

4th Clock:

Acc[3] <- Q[0]

Q[3] <- Acc[0]

5th Clock:

Acc[3] <- Q[0]

Q[3] <- Acc[0]

6th Clock:

Acc[3] <- Q[0]

Q[3] <- Acc[0]

7th Clock: Q\_right = 0, Acc\_Lsrc[1..0] = 01

Acc[3] <- Q[3]

Q[0] <- 0

8th Clock:

Acc[3] <- Q[3]

Q[0] <- 0

9th Clock:

Acc[3] <- Q[3]

Q[0] <- 0

10th Clock:

Acc[3] <- Q[3]

Q[0] <- 0

11th Clock: Acc\_parallel = 1, R1Src = 1

R1 <- Acc

11 Clock cycles in total.