DELHI
TECHNOLOGICAL
UNIVERSITY

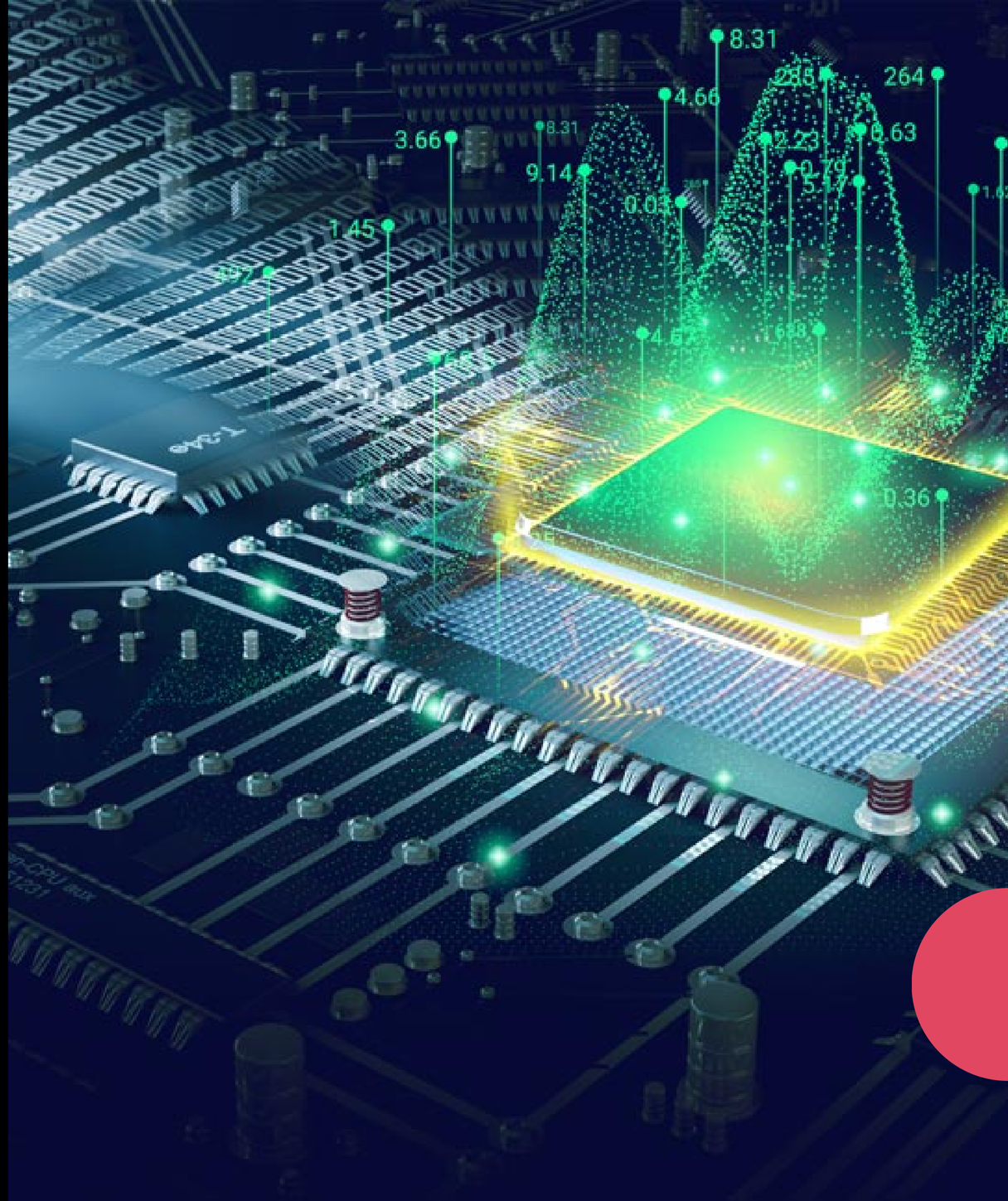
DESIGN OF 4-BIT BOOTH'S MULTIPLIER CIRCUIT WITH BUILT-IN SELF TEST FEATURE

Shlok Garg 2K20/EE/258
Sidakpreet Singh 2K20/EE/265
Vikas Kumar 2K20/EE/300

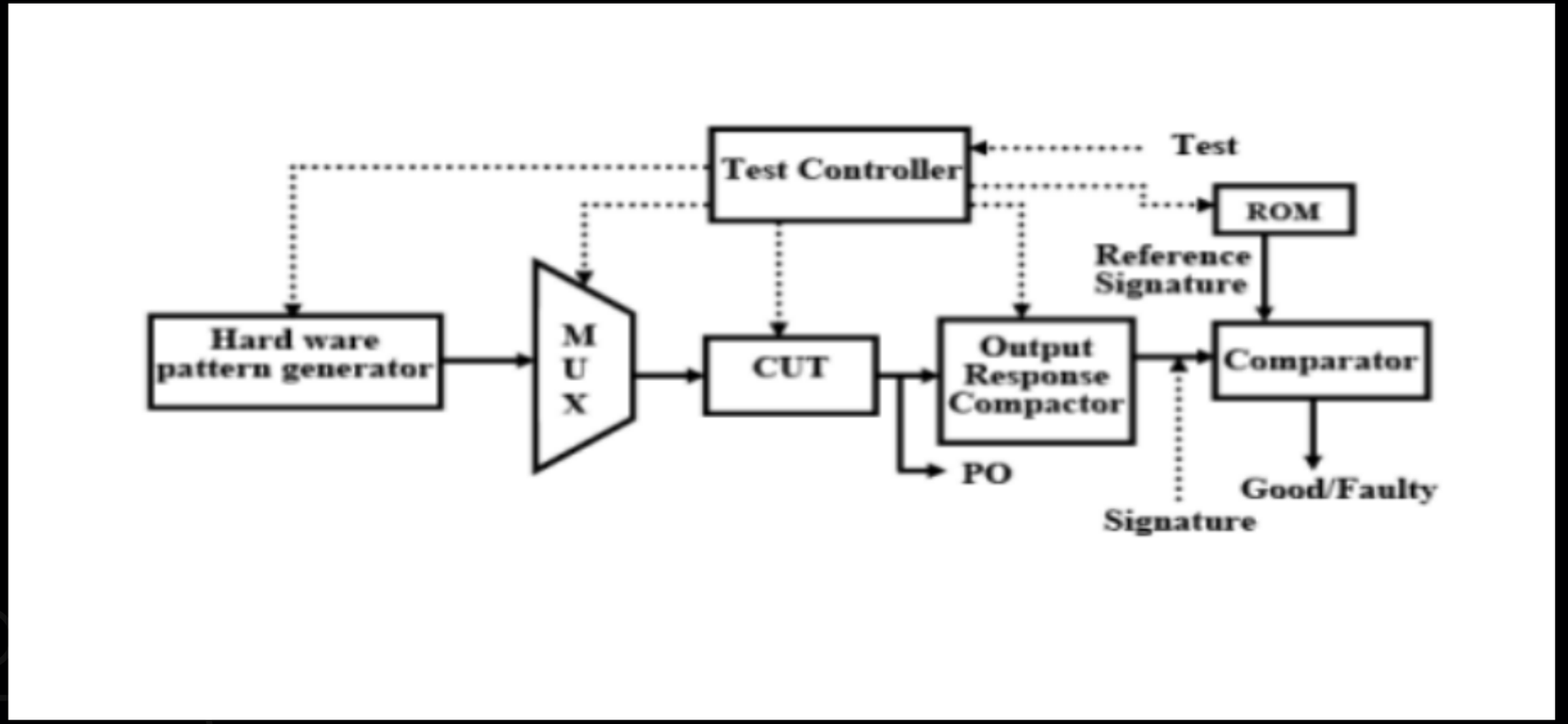
OVERVIEW

Built-in self-test, or BIST, is a DFT methodology involving the insertion of additional hardware and software features into integrated circuits to allow them to perform self-testing, thereby reducing dependence on an external ATE and, thus, reducing testing cost. The BIST concept is applicable to about any kind of circuit.

In BIST, a test pattern generator generates test patterns and a signature analyzer (SA) compares test responses. The entire process is controlled by BIST controller.



BUILT IN SELF TEST



Built-In Self Test (BIST) works by integrating self-testing capabilities into electronic systems. BIST generates test patterns, applies them to the device under test (DUT), and analyzes the response to detect faults or errors.

COMPONENTS OF BIST

01

Test generator

generates test patterns that are applied to the device under test (DUT) to detect faults or errors.

Here in this project, LFSR with primitive polynomial is being used as test pattern generator.

02

Response analyzer

analyzes the response of the DUT to the test patterns and compares it to the expected results. MISR is being used in this project as ORA.

03

BIST controller

controls the overall operation of the BIST system and manages the generation and analysis of test patterns and responses.

BOOTH'S MULTIPLIER

1. Initially the multiplier is in an undefined state. On start = 1. The multiplier resets A, Q_1, and counts to zero.

2. Next state is Idle, now the start signal needs to be zero otherwise, it will again move to reset state and ask for parameters multiplier and multiplicand again.

3. Check for {Q[0], Q_1}. If {0,1} move to step 4, if {1,0} move to step 5, else skip to step 6.

4. Assign A = A + M. Move to step 6.

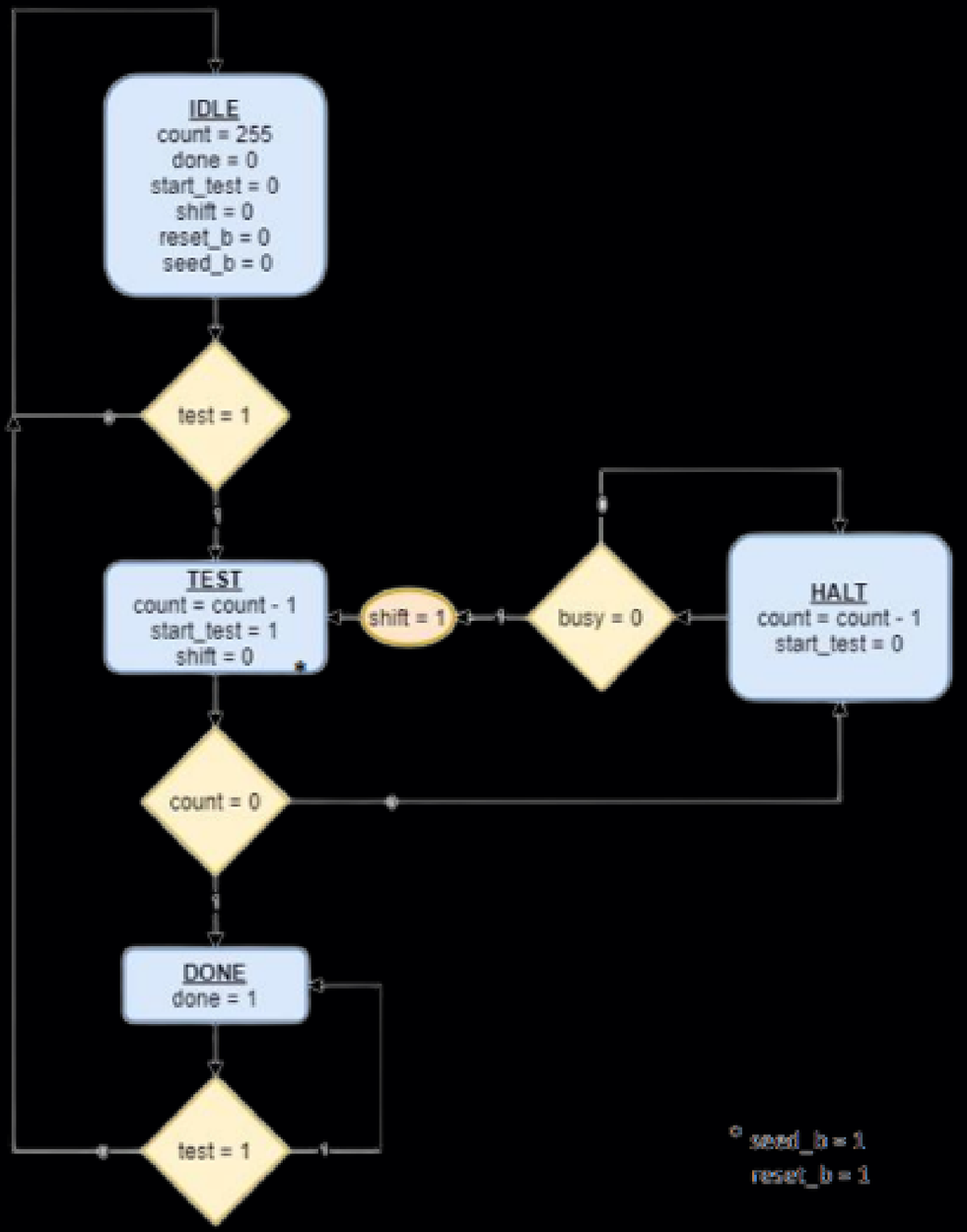
5. Assign A = A - M. Move to step 6.

6. Arithmetic Shift Right, the triplet {A, Q, Q_1}. Decrement count.

7. Check for count < 4, if true set busy=0 otherwise set busy=1. Move to Idle state and continue to step 3

BIST CONTROLLER

1. Reset all the signals to default zero value in Idle State. Reset to initial seed in LFSR and MISR to 1111_1111 and 0000_0000 by applying seed_b = 0 and reset_b = 0 respectively. The LFSR must be seeded to a non-zero value initially to generate the pattern.
2. On test = 1, move to the next state i.e. Test State. Issue start = 1 on Booth's Multiplier. Set shift = 0 to pause the pattern generator LFSR and compactor MISR. Set reset_b and seed_b as 1, as LFSR and MISR are already seeded.
3. If count = 0 jump to step 4, else jump to step 6.
4. This is Halt State, and the state machine remains in this state for the most amount of time. Set the start signal = 0 to begin multiplying.
5. If busy = 0, move to step 2 with shift = 1 as mealy output. This signal resumes the LFSR generator and MISR compactor to shift and accept new values. Else stay in the current state.
6. This is Done State, the testing is completed hence set the done signal as 1.

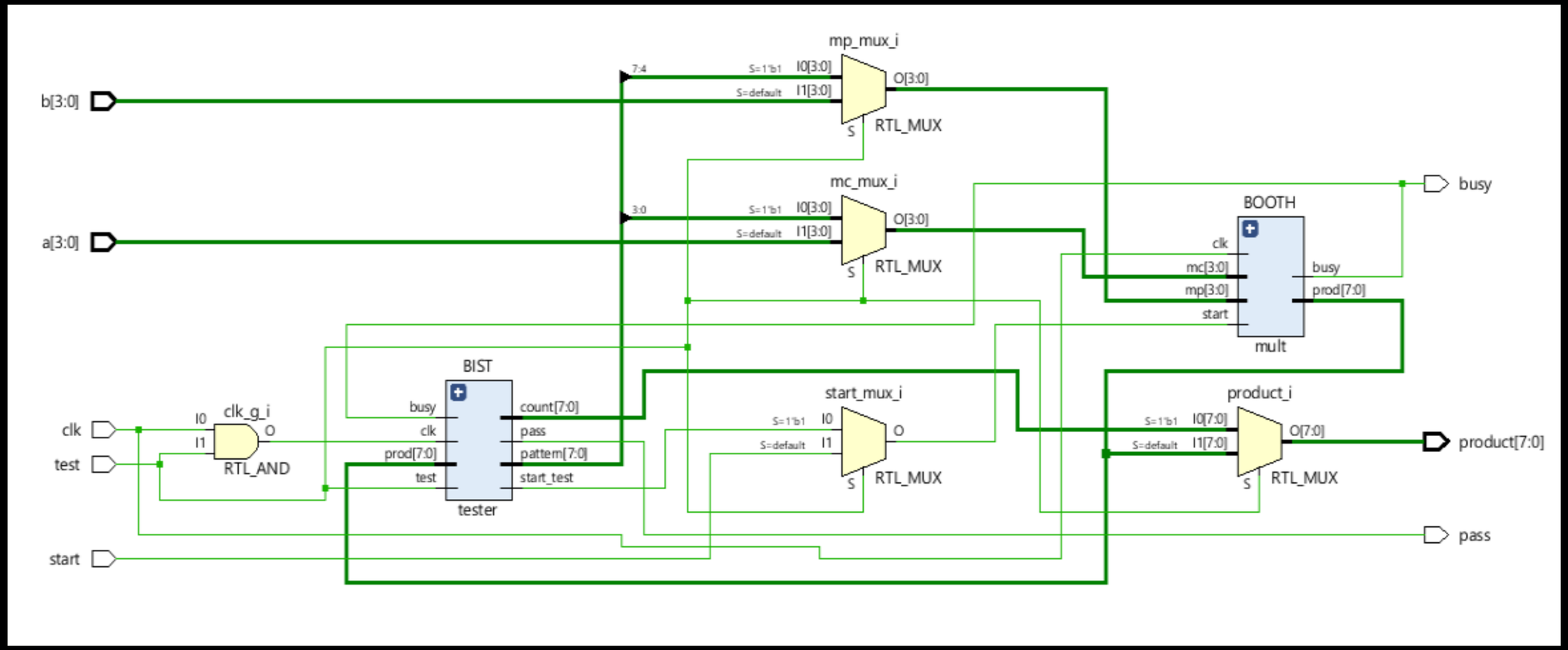


SOFTWARE ENVIRONMENT

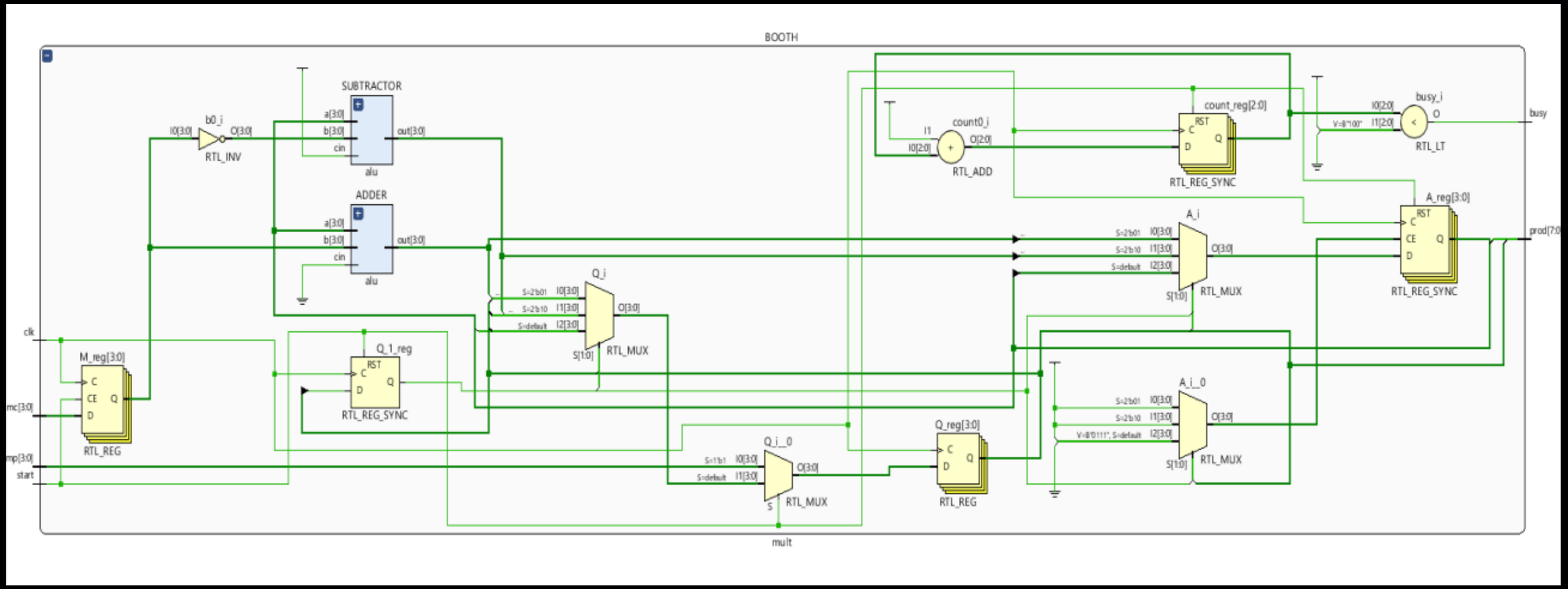
The complete project is done in Xilinx Vivado Software Environment using Verilog as HDL. The various tools used include the text editor, Vivado Simulator, Synthesis and RTL Analysis.

RTL SCHEMATIC

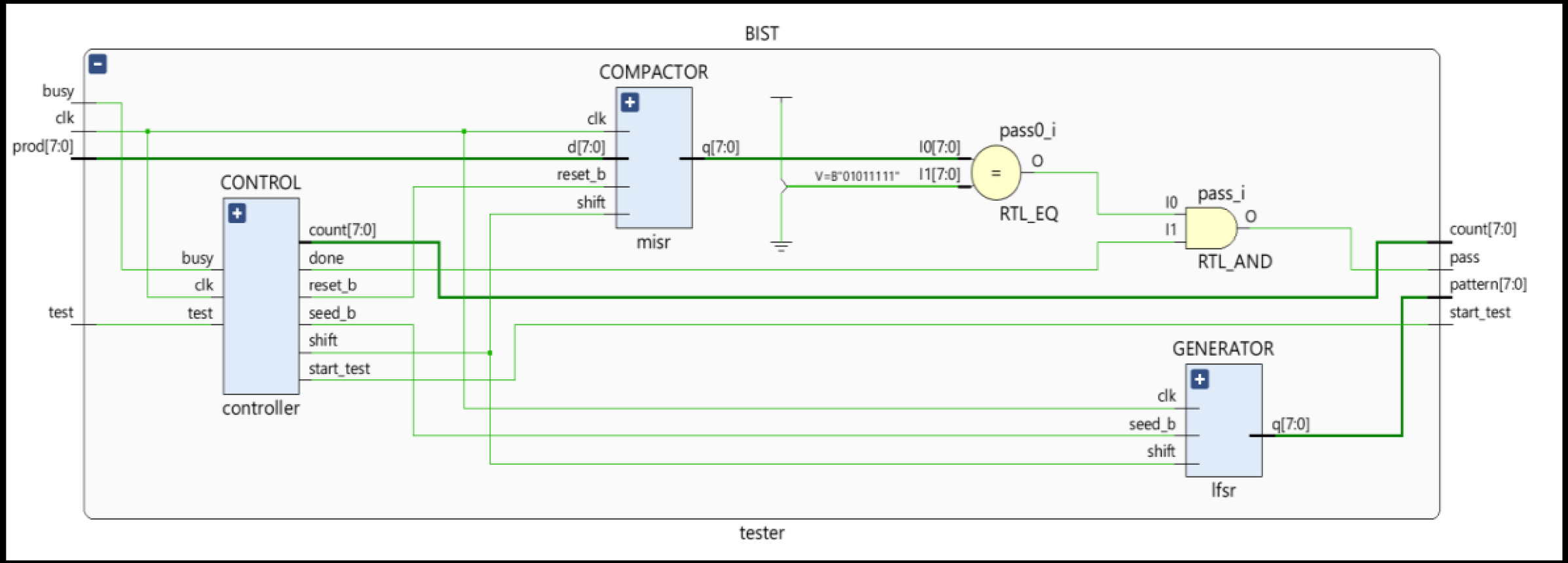
Top Module



Booth's Multiplier

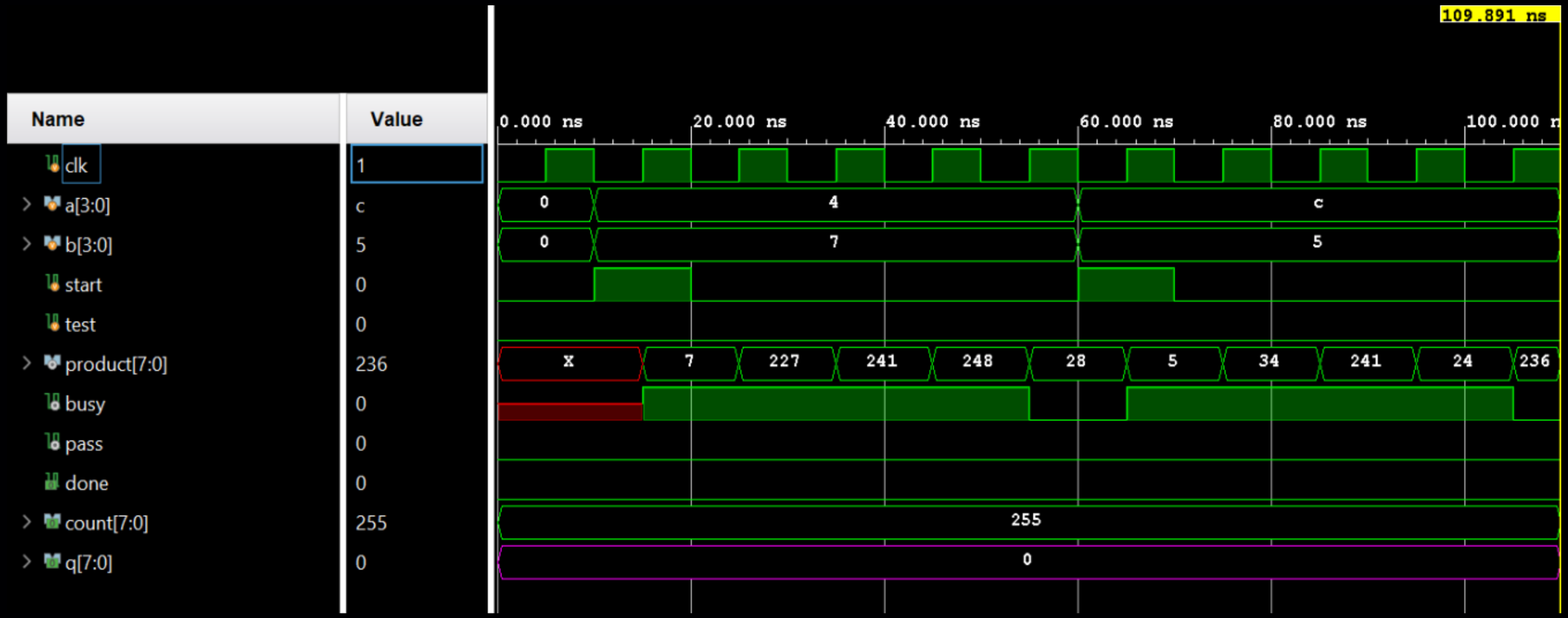


BIST Controller

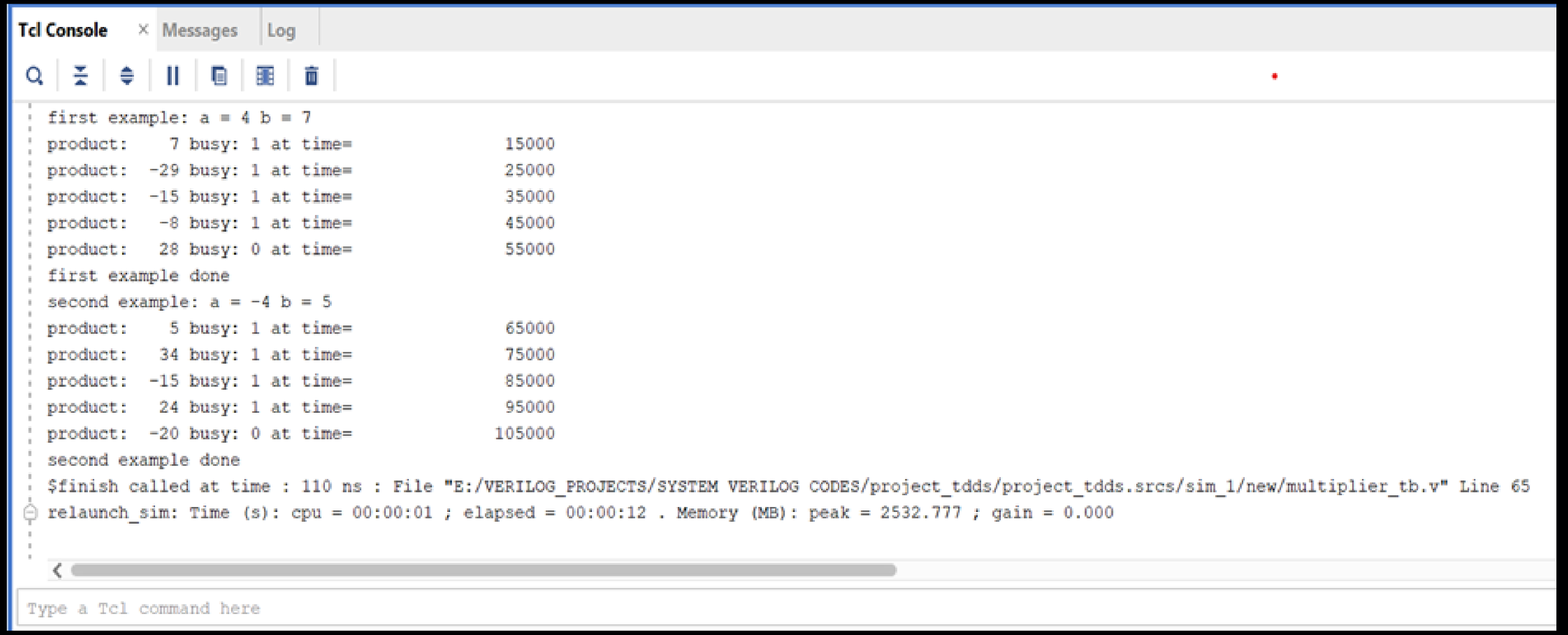


WAVEFORMS

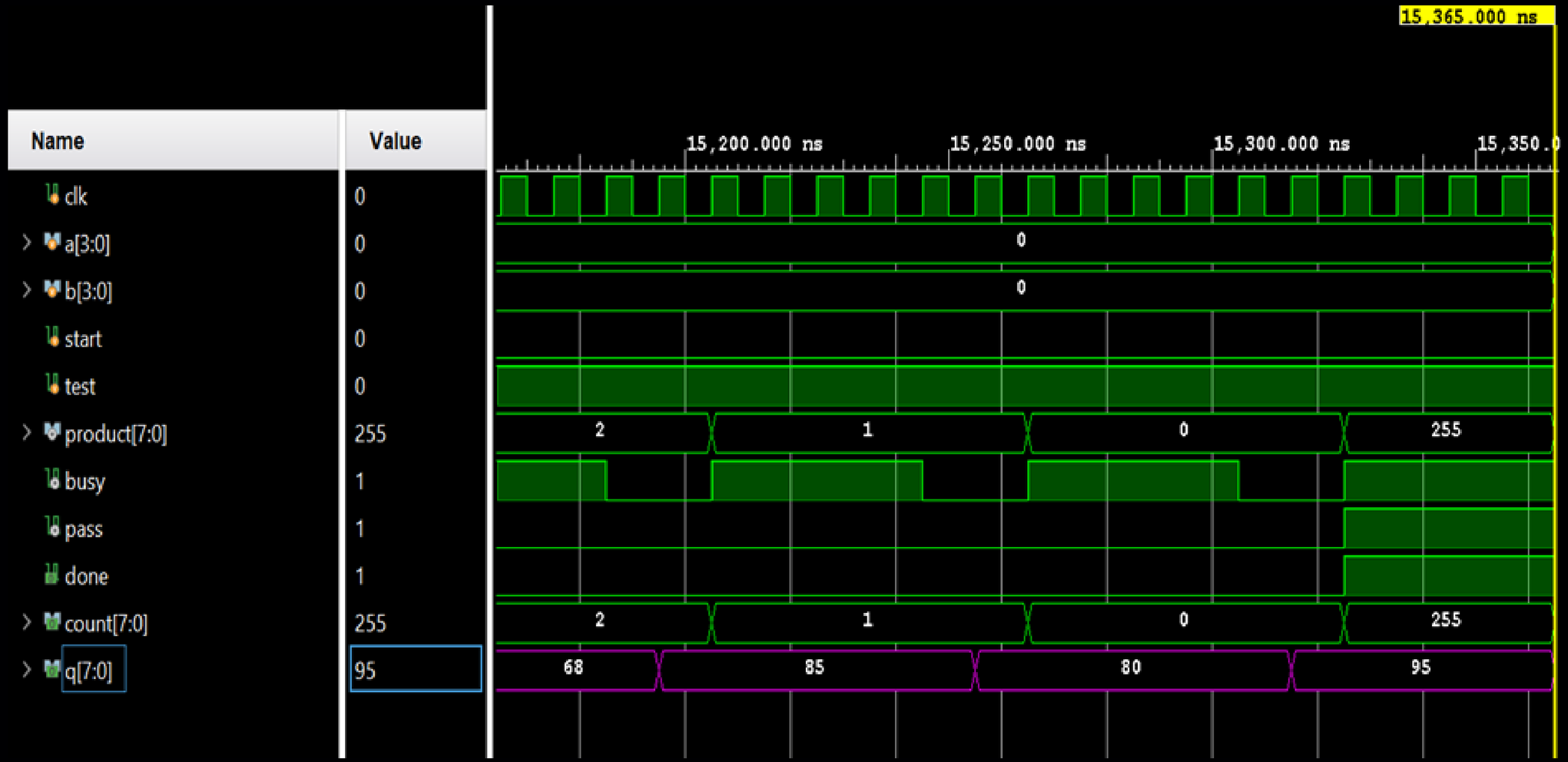
BOOTH'S MULTIPLIER



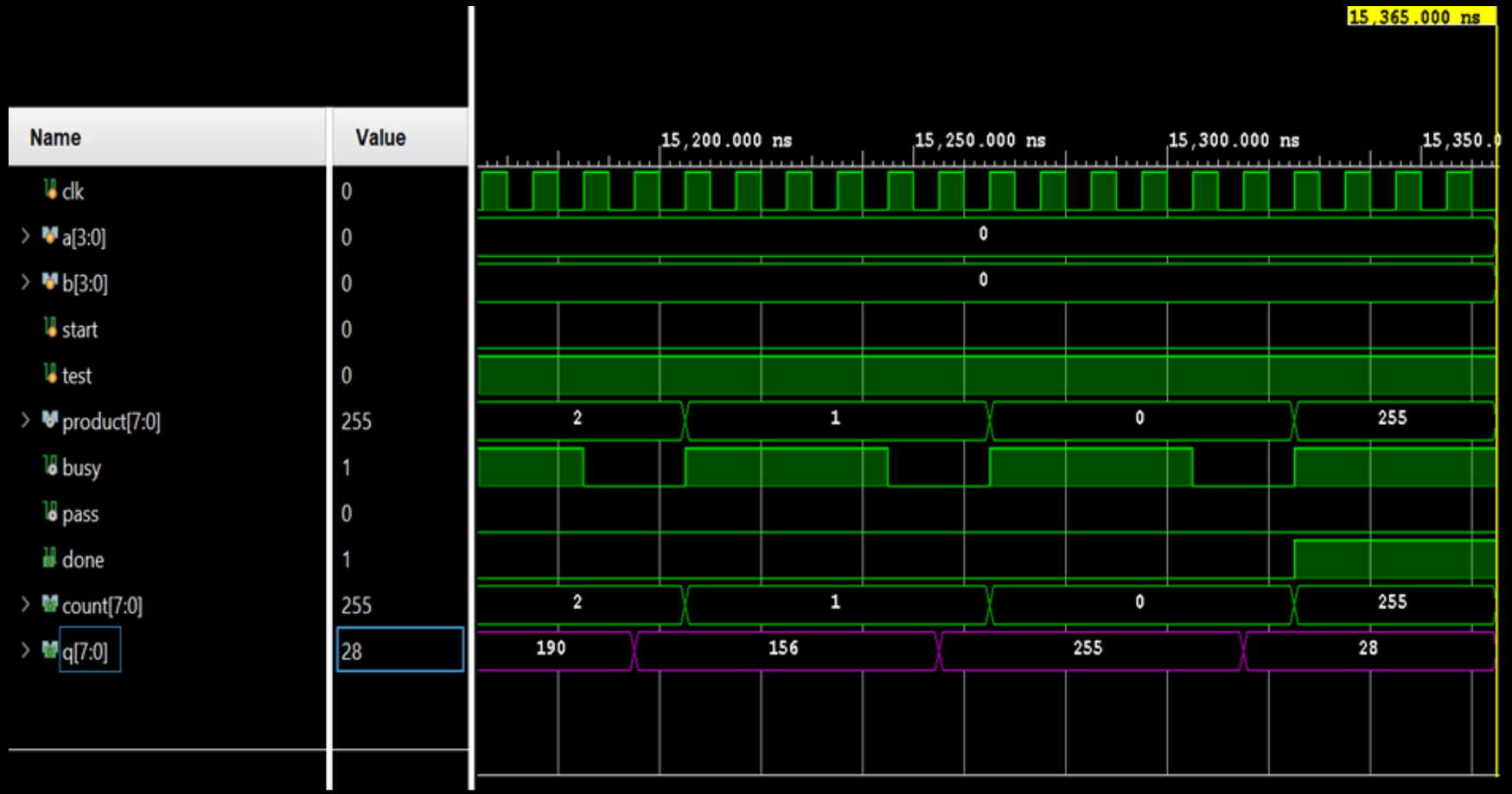
TCL CONSOLE


Tcl Console
Messages
Log
first example: a = 4 b = 7
product: 7 busy: 1 at time= 15000
product: -29 busy: 1 at time= 25000
product: -15 busy: 1 at time= 35000
product: -8 busy: 1 at time= 45000
product: 28 busy: 0 at time= 55000
first example done
second example: a = -4 b = 5
product: 5 busy: 1 at time= 65000
product: 34 busy: 1 at time= 75000
product: -15 busy: 1 at time= 85000
product: 24 busy: 1 at time= 95000
product: -20 busy: 0 at time= 105000
second example done
$finish called at time : 110 ns : File "E:/VERILOG_PROJECTS/SYSTEM VERILOG CODES/project_tdds/project_tdds.srcs/sim_1/new/multiplier_tb.v" Line 65
relaunch_sim: Time (s): cpu = 00:00:01 ; elapsed = 00:00:12 . Memory (MB): peak = 2532.777 ; gain = 0.000
Type a Tcl command here

BIST TESTER CIRCUIT

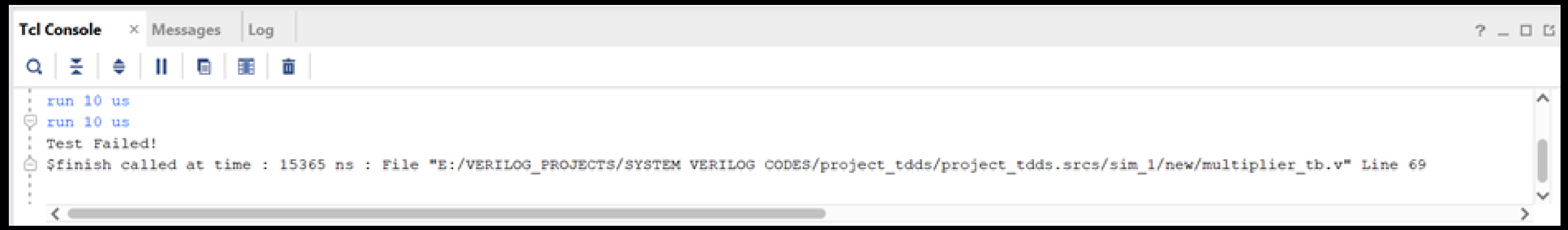


BIST TESTER CIRCUIT(WITH FAULT)



CONCLUSION

TCL CONSOLE FOR FAULT DETECTION



Built in self test for Booth's algorithm is implemented and its signature 95 for particular seed has been stored and now compared with MISR outputs and if they dont match BIST gives output as 'TEST FAILED.'


THANK YOU