TASK 1

3.1.1

And Gate

	inputs	expected output	S-A output
x -S-A- 0	(x1,y1)	(z1)	(z0)
x -S-A- 1	(x0,y1)	(z0)	(z1)
y -S-A- 0	(x1,y1)	(z1)	(z0)
y -S-A- 1	(x1,y0)	(z0)	(z1)
z -S-A- 0	(x1,y1)	(z1)	(z0)
z -S-A- 1	(x0,y0),(x1,y0),(x0,y1)	**(z0)**	(z1)

And Gate truth table

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

equivalent faults	dominant faults	collapsed fault set
x1,y1, z1	x0,y1,z0	x1,y1, z1
	x1, y0, z0	x0,y1,z0
		x1, y0, z0

x	-S-A-	0
x	-S-A-	1
y	-S-A-	0
y	-S-A-	1
z	-S-A-	0
z	-S-A-	1

Or Gate

	inputs	expected output	S-A output
x -S-A- 0	(x1,y0)	(z0)	(z0)
x -S-A- 1	(x0,y0)	(z0)	(z1)
y -S-A- 0	(x0,y1)	(z1)	(z0)
y -S-A- 1	(x0,y0)	(z0)	(z1)
z -S-A- 0	(x0,y1),(x1,y0),(x1,y1)	**(z1)**	(z0)
z -S-A- 1	(x0,y0)	(z0)	(z1)

Or Gate truth table

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

equivalent faults	dominant faults	collapsed fault set
x0,y0,z0	x1,y0,z0	x0,y0,z0
	x0,y1,z1	x1,y0,z0
		x0,y1,z1

x	-S-A-	0
x	-S-A-	1
y	-S-A-	0
y	-S-A-	1
z	-S-A-	0
z	-S-A-	1

Nor Gate

	inputs	check output	S-A output
x -S-A- 0	(x1,y0)	(z0)	(z1)
x -S-A- 1	(x0,y0)	(z1)	(z0)
y -S-A- 0	(x0,y1)	(z0)	(z1)
y -S-A- 1	(x0,y0)	(z1)	(z0)
z -S-A- 0	(x0,y0)	(z1)	(z0)
z -S-A- 1	(x0,y1),(x1,y0),(x1,y1)	(z0)	(z1)

Nor Gate truth table

x	y	z
0	0	1
0	1	0
1	0	0
1	1	0

equivalent faults	dominant faults	collapsed fault set
x0,y0,z1	x1,y0,z0	x0,y0,z1
	x0,y1,z0	x1,y0,z0
		x0,y1,z0

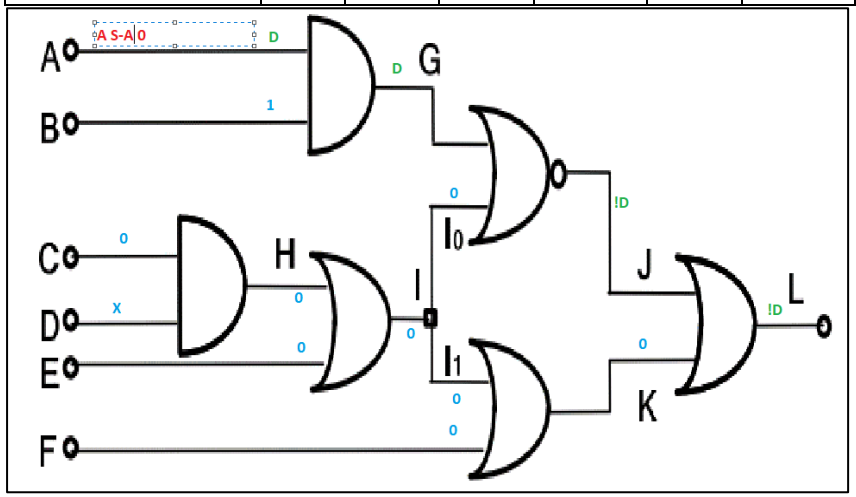
x	-S-A-	0
x	-S-A-	1
y	-S-A-	0
y	-S-A-	1
z	-S-A-	0
z	-S-A-	1

3.1.2

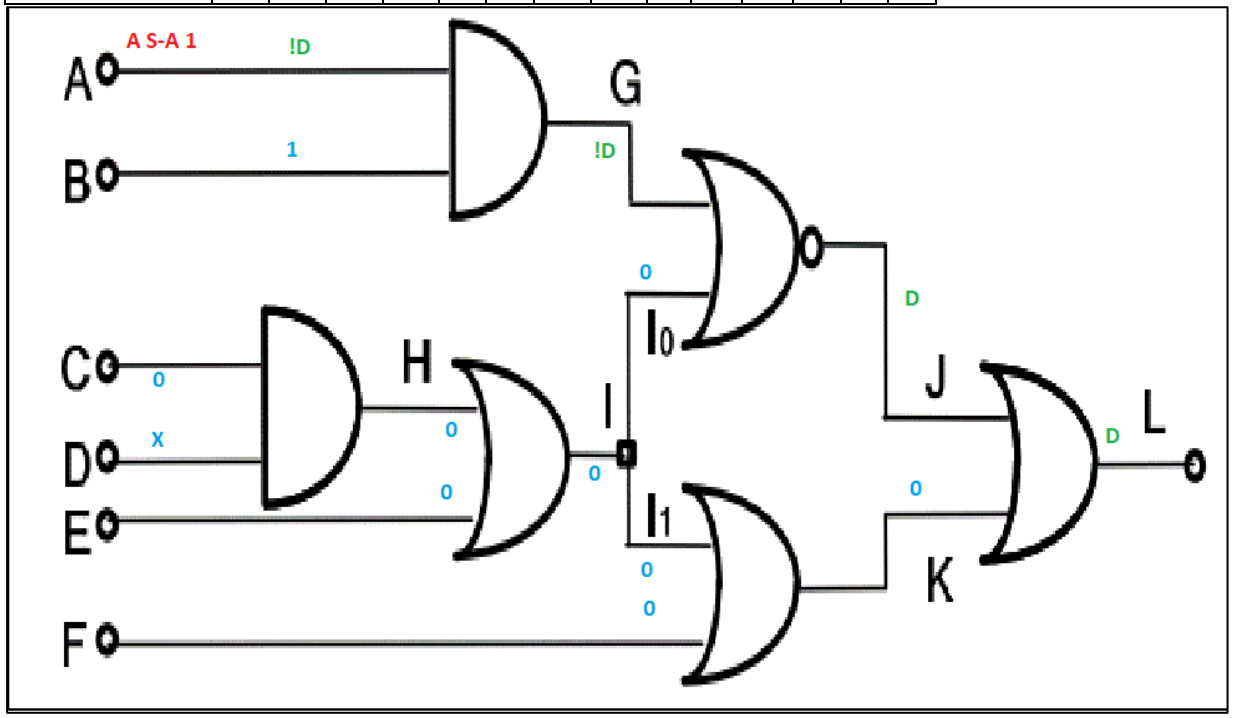
A	-S-A-	0			
A	-S-A-	1			
B	-S-A-	0		Equivalent to A-s-a 0	
B	-S-A-	1			
C	-S-A-	0			
C	-S-A-	1			
D	-S-A-	0		Equivalent to C s-a- 0	
D	-S-A-	1			
E	-S-A-	0			
E	-S-A-	1		Equivalent to H s-a- 1	
F	-S-A-	0			
F	-S-A-	1		Equivalent to I1 s-a- 1	
G	-S-A-	0		Equivalent to A-s-a 0	
G	-S-A-	1		Dominated by A s-a- 1	Dominated by B s-a- 1
H	-S-A-	0		Equivalent to C s-a- 0	
H	-S-A-	1		Dominated by C s-a- 1	Dominated by D s-a- 1
I	-S-A-	0		Dominated by H s-a- 0	Dominated by E s-a- 0
I	-S-A-	1		Equivalent to H s-a- 1	
Io	-S-A-	0			
Io	-S-A-	1		Equivalent to G s-a- 1	
I1	-S-A-	0			
I1	-S-A-	1			
J	-S-A-	0		Equivalent to G s-a- 1	
J	-S-A-	1		Dominated by G s-a- 0	Dominated by I0 s-a- 0
K	-S-A-	0		Dominated by I1 s-a 0	Dominated by F s-a 0
K	-S-A-	1		Equivalent to J s-a-1	Equivalent to I1 s-a- 1
L	-S-A-	0		Dominated by J s-a 0	Dominated by K s-a 0
L	-S-A-	1		Equivalent to J s-a-1	

3.1.3

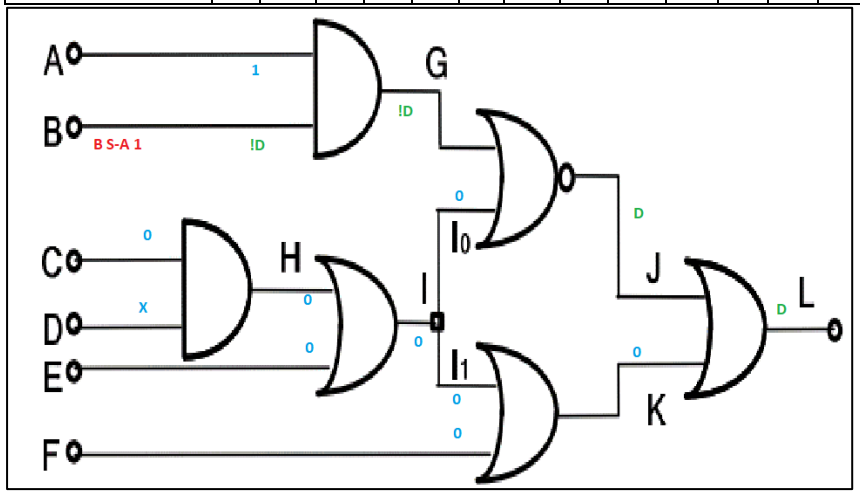
Node A s-a-0	A	B	C	D	E	F	G	H	I	I0	I1	J	K	L
STEP 1	D													
STEP 2	D	1					D							
STEP 3	D	1					D			0		!D		
STEP 4	D	1					D			0		!D	0	!D
STEP 5	D	1				0	D			0	0	!D	0	!D
STEP 6	D	1			0	0	D	0	0	0	0	!D	0	!D
STEP 7	D	1			0	0	D	0	0	0	0	!D	0	!D
STEP 8	D	1	0	X	0	0	D	0	0	0	0	!D	0	!D



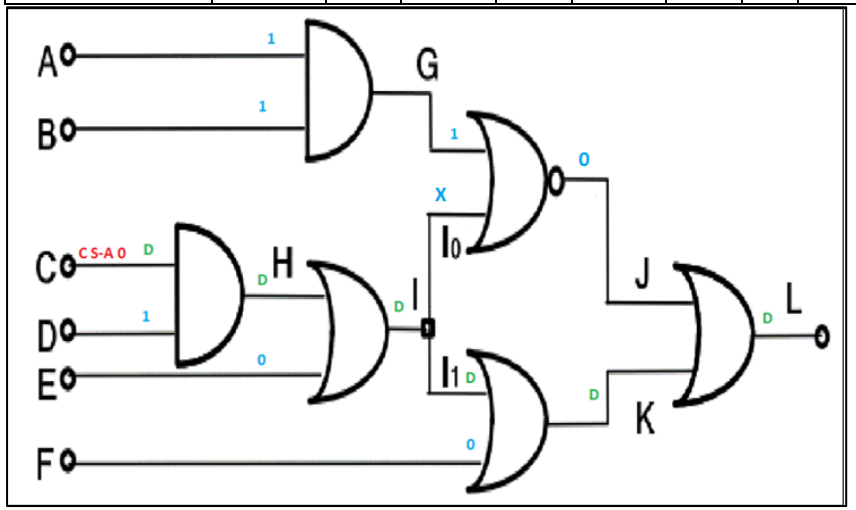
Node A s-a-1	A	B	C	D	E	F	G	H	I	I0	I1	J	K	L
STEP 1	!D													
STEP 2	!D	1					!D							
STEP 3	!D	1					!D			0		D		
STEP 4	!D	1					!D			0		D	0	D
STEP 5	!D	1				0	!D			0	0	D	0	D
STEP 6	!D	1			0	0	!D	0	0	0	0	D	0	D
STEP 7	!D	1	0	X	0	0	!D	0	0	0	0	D	0	D



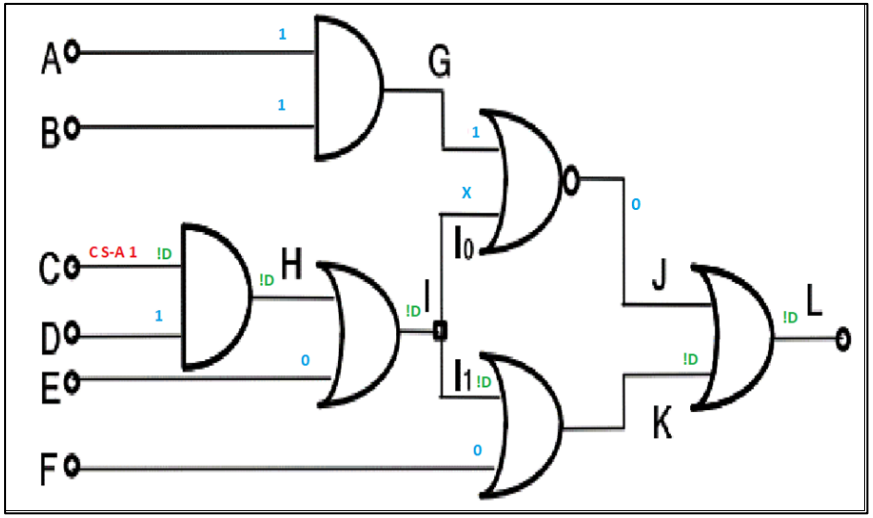
Node B s-a-1	A	B	C	D	E	F	G	H	I	I0	I1	J	K	L
STEP 1		!D												
STEP 2	1	!D					!D							
STEP 3	1	!D					!D			0		D		
STEP 4	1	!D					!D			0		D	0	D
STEP 5	1	!D				0	!D			0	0	D	0	D
STEP 6	1	!D			0	0	!D	0	0	0	0	D	0	D
STEP 7	1	!D	0	X	0	0	!D	0	0	0	0	D	0	D



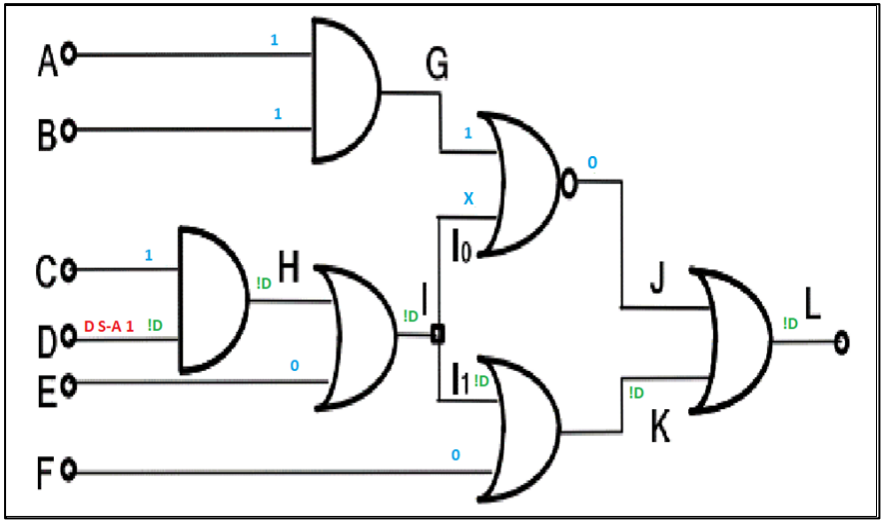
Node C s-a-0	A	B	C	D	E	F	G	H	I	I0	I1	J	K	L
STEP 1			D											
STEP 2			D	1				D						
STEP 3			D	1	0			D	D					
STEP 4			D	1	0			D	D	D	D			
STEP 5			D	1	0	0		D	D	D	D		D	
STEP 6			D	1	0	0		D	D	D	D	!D	D	D
STEP 7	1	1	D	1	0	0	1	D	D	X	D	0	D	D



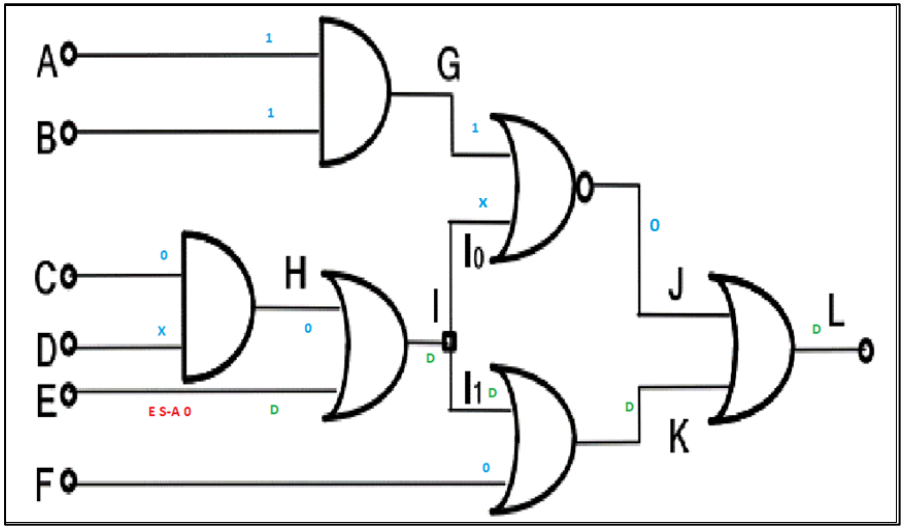
Node C s-a-1	A	B	C	D	E	F	G	H	I	I0	I1	J	K	L
STEP 1			!D											
STEP 2			!D	1				!D						
STEP 3			!D	1	0			!D	!D	!D	!D			
STEP 4			!D	1	0	0		!D	!D	!D	!D		!D	!D
STEP 5			!D	1	0	0	1	!D	!D	X	!D	0	!D	!D
STEP 6	1	1	!D	1	0	0	1	!D	!D	X	!D	0	!D	!D



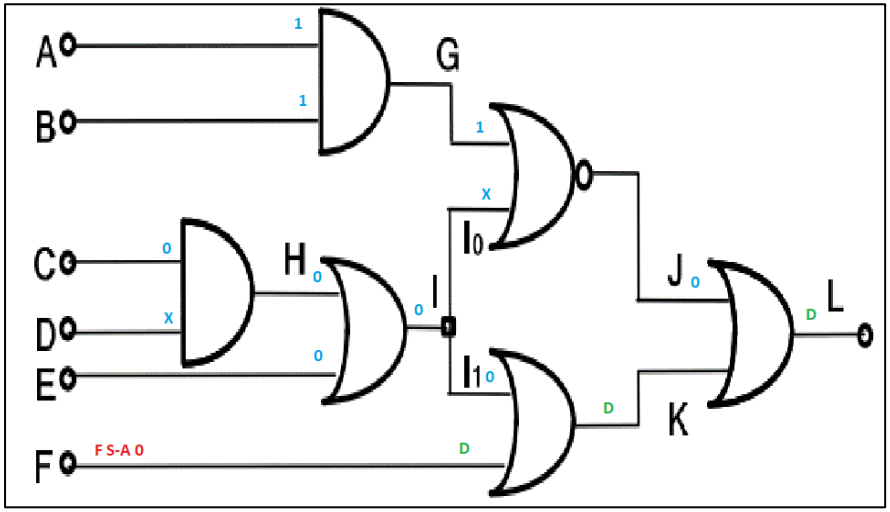
Node D s-a-1	A	B	C	D	E	F	G	H	I	I0	I1	J	K	L
STEP 1				!D										
STEP 2			1	!D				!D						
STEP 3			1	!D	0			!D	!D	!D	!D			
STEP 4			1	!D	0	0		!D	!D	!D	!D		!D	!D
STEP 5			1	!D	0	0	1	!D	!D	X	!D	0	!D	!D
STEP 6	1	1	1	!D	0	0	1	!D	!D	X	!D	0	!D	!D



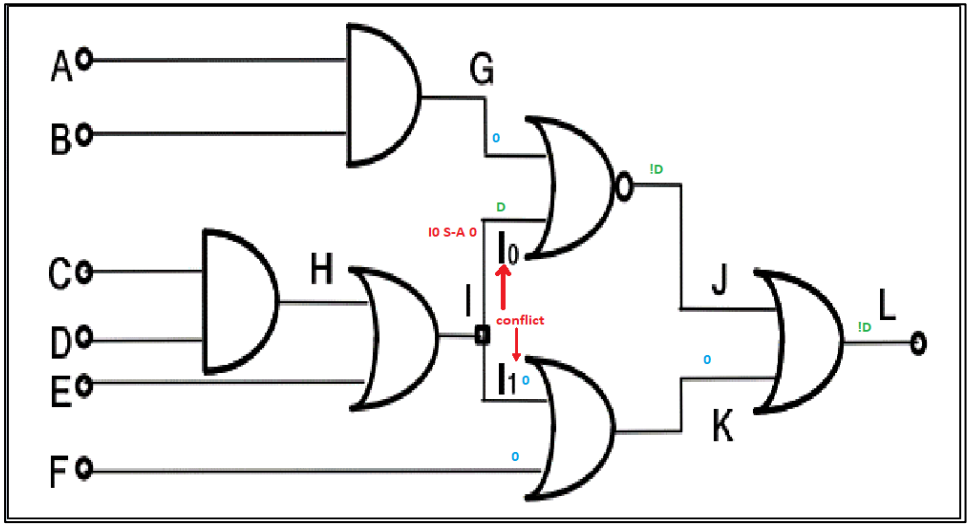
Node E s-a-0	A	B	C	D	E	F	G	H	I	I0	I1	J	K	L
STEP 1					D									
STEP 2					D			0	D					
STEP 3			0	X	D			0	D					
STEP 4			0	X	D			0	D	D	D			
STEP 5			0	X	D	0		0	D	D	D		D	D
STEP 6			0	X	D	0	1	0	D	X	D	0	D	D
STEP 7	1	1	0	X	D	0	1	0	D	X	D	0	D	D



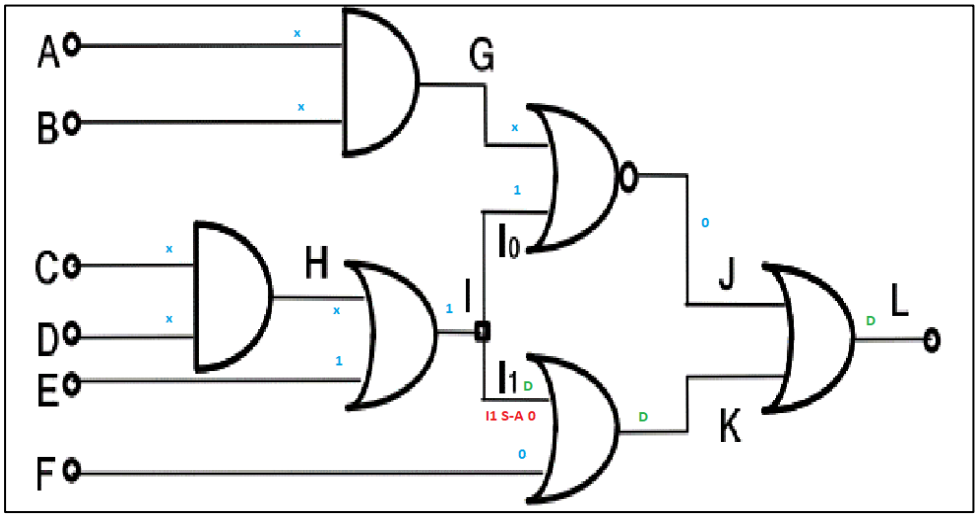
Node F s-a-0	A	B	C	D	E	F	G	H	I	I0	I1	J	K	L
STEP 1						D								
STEP 2						D					0		D	
STEP 3						D					0	0	D	D
STEP 4						D	1			X	0	0	D	D
STEP 5					0	D	1	0	0	X	0	0	D	D
STEP 6			0	X	0	D	1	0	0	X	0	0	D	D
STEP 7	1	1	0	X	0	D	1	0	0	X	0	0	D	D



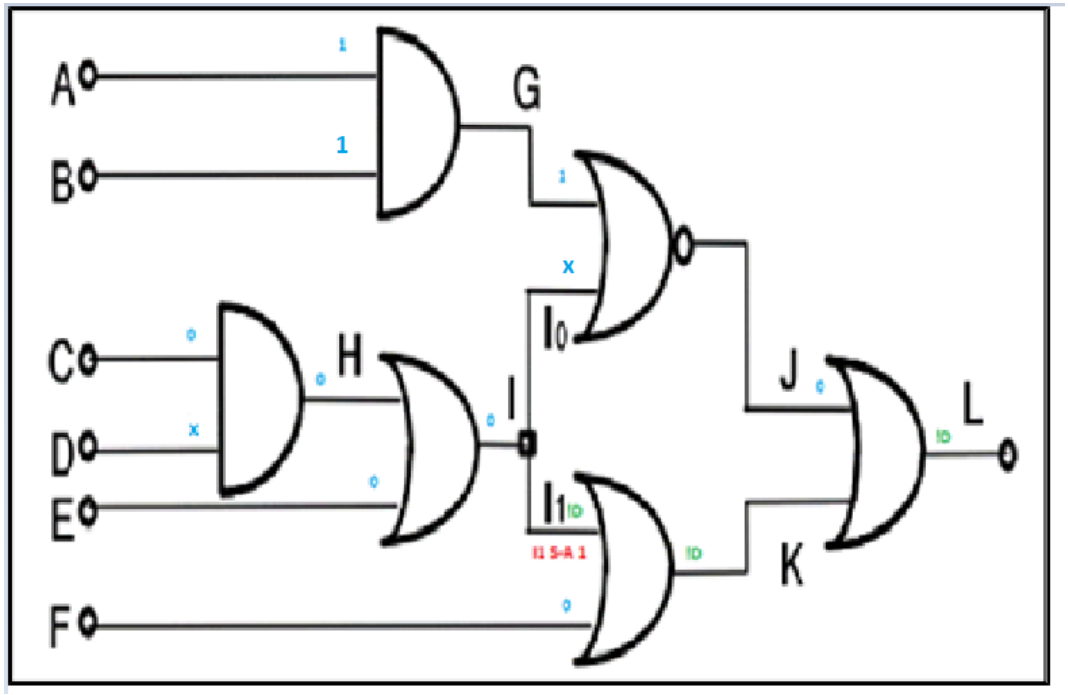
Node I_0 s-a-0	A	B	C	D	E	F	G	H	I	I0	I1	J	K	L
STEP 1														
STEP 2														
STEP 3														
STEP 4														
STEP 5														
STEP 6														
STEP 7														



Node I_1 s-a-0	A	B	C	D	E	F	G	H	I	I0	I1	J	K	L
STEP 1											D			
STEP 2						0					D		D	
STEP 3						0	X		1	1	D	0	D	D
STEP 4	X	X			1	0	X	X	1	1	D	0	D	D
STEP 5	X	X	X	X	1	0	X	X	1	1	D	0	D	D



Node I_1 s-a-1	A	B	C	D	E	F	G	H	I	I0	I1	J	K	L
STEP 1											!D			
STEP 2						0					!D		!D	
STEP 3						0					!D	0	!D	!D
STEP 4						0	1			x	!D	0	!D	!D
STEP 5	1	1				0	1			x	!D	0	!D	!D
STEP 6	1	1	0	X	0	0	1	0	0	x	!D	0	!D	!D



3.1.4

Matching pairs														
Node A s-a-0	D	1	0	X	0	0	D	0	0	0	0	!D	0	!D
Node I_1 s-a-1	1	1	0	X	0	0	1	0	0	x	!D	0	!D	!D

Node I_1 s-a-0	X	X	X	X	1	0	X	X	1	1	D	0	D	D
Node E s-a-0	1	1	0	X	D	0	1	0	D	X	D	0	D	D

3.1.5

Node A s-a-0	1	1	0	0	0	0	1	0	0	0	0	0	0	0
Node A s-a-1	0	1	0	0	0	0	0	0	0	0	0	1	0	1
Node B s-a-1	1	0	0	0	0	0	0	0	0	0	0	1	0	1
Node C s-a-0	1	1	1	1	0	0	1	1	1	0	1	0	1	1
Node C s-a-1	1	1	0	1	0	0	1	0	0	0	0	0	0	0
Node D s-a-1	1	1	1	0	0	0	1	0	0	0	0	0	0	0
Node E s-a-0	1	1	0	0	1	0	1	0	1	0	1	0	1	1
Node F s-a-0	1	1	0	0	0	1	1	0	0	0	0	0	1	1

3.1.6

Node A s-a-0					
Node A s-a-0	Node A s-a-1	Node B s-a-0	Node B s-a-1	Node C s-a-0	Node C s-a-1
Node D s-a-0	Node D s-a-1	Node E s-a-0	Node E s-a-1	Node F s-a-0	Node F s-a-1
Node G s-a-0	Node G s-a-1	Node H s-a-0	Node H s-a-1	Node I s-a-0	Node I s-a-1
Node I_0 s-a-0	Node I_0 s-a-1	Node I_1 s-a-0	Node I_1 s-a-1	Node J s-a-0	Node J s-a-1
Node K s-a-0	Node K s-a-1	Node L s-a-0	Node L s-a-1		

Node A s-a-1					
Node A s-a-0	Node A s-a-1	Node B s-a-0	Node B s-a-1	Node C s-a-0	Node C s-a-1
Node D s-a-0	Node D s-a-1	Node E s-a-0	Node E s-a-1	Node F s-a-0	Node F s-a-1
Node G s-a-0	Node G s-a-1	Node H s-a-0	Node H s-a-1	Node I s-a-0	Node I s-a-1
Node I_0 s-a-0	Node I_0 s-a-1	Node I_1 s-a-0	Node I_1 s-a-1	Node J s-a-0	Node J s-a-1
Node K s-a-0	Node K s-a-1	Node L s-a-0	Node L s-a-1		

Node B s-a-1					
Node A s-a-0	Node A s-a-1	Node B s-a-0	Node B s-a-1	Node C s-a-0	Node C s-a-1
Node D s-a-0	Node D s-a-1	Node E s-a-0	Node E s-a-1	Node F s-a-0	Node F s-a-1
Node G s-a-0	Node G s-a-1	Node H s-a-0	Node H s-a-1	Node I s-a-0	Node I s-a-1
Node I_0 s-a-0	Node I_0 s-a-1	Node I_1 s-a-0	Node I_1 s-a-1	Node J s-a-0	Node J s-a-1
Node K s-a-0	Node K s-a-1	Node L s-a-0	Node L s-a-1		

Node C s-a-0					
Node A s-a-0	Node A s-a-1	Node B s-a-0	Node B s-a-1	Node C s-a-0	Node C s-a-1
Node D s-a-0	Node D s-a-1	Node E s-a-0	Node E s-a-1	Node F s-a-0	Node F s-a-1
Node G s-a-0	Node G s-a-1	Node H s-a-0	Node H s-a-1	Node I s-a-0	Node I s-a-1
Node I_0 s-a-0	Node I_0 s-a-1	Node I_1 s-a-0	Node I_1 s-a-1	Node J s-a-0	Node J s-a-1
Node K s-a-0	Node K s-a-1	Node L s-a-0	Node L s-a-1		

Node C s-a-1					
Node A s-a-0	Node A s-a-1	Node B s-a-0	Node B s-a-1	Node C s-a-0	Node C s-a-1
Node D s-a-0	Node D s-a-1	Node E s-a-0	Node E s-a-1	Node F s-a-0	Node F s-a-1
Node G s-a-0	Node G s-a-1	Node H s-a-0	Node H s-a-1	Node I s-a-0	Node I s-a-1
Node I_0 s-a-0	Node I_0 s-a-1	Node I_1 s-a-0	Node I_1 s-a-1	Node J s-a-0	Node J s-a-1
Node K s-a-0	Node K s-a-1	Node L s-a-0	Node L s-a-1		

Node D s-a-1					
Node A s-a-0	Node A s-a-1	Node B s-a-0	Node B s-a-1	Node C s-a-0	Node C s-a-1
Node D s-a-0	Node D s-a-1	Node E s-a-0	Node E s-a-1	Node F s-a-0	Node F s-a-1
Node G s-a-0	Node G s-a-1	Node H s-a-0	Node H s-a-1	Node I s-a-0	Node I s-a-1
Node I_0 s-a-0	Node I_0 s-a-1	Node I_1 s-a-0	Node I_1 s-a-1	Node J s-a-0	Node J s-a-1
Node K s-a-0	Node K s-a-1	Node L s-a-0	Node L s-a-1		

Node E s-a-0					
Node A s-a-0	Node A s-a-1	Node B s-a-0	Node B s-a-1	Node C s-a-0	Node C s-a-1
Node D s-a-0	Node D s-a-1	Node E s-a-0	Node E s-a-1	Node F s-a-0	Node F s-a-1
Node G s-a-0	Node G s-a-1	Node H s-a-0	Node H s-a-1	Node I s-a-0	Node I s-a-1
Node I_0 s-a-0	Node I_0 s-a-1	Node I_1 s-a-0	Node I_1 s-a-1	Node J s-a-0	Node J s-a-1
Node K s-a-0	Node K s-a-1	Node L s-a-0	Node L s-a-1		

Node F s-a-0					
Node A s-a-0	Node A s-a-1	Node B s-a-0	Node B s-a-1	Node C s-a-0	Node C s-a-1
Node D s-a-0	Node D s-a-1	Node E s-a-0	Node E s-a-1	Node F s-a-0	Node F s-a-1
Node G s-a-0	Node G s-a-1	Node H s-a-0	Node H s-a-1	Node I s-a-0	Node I s-a-1
Node I_0 s-a-0	Node I_0 s-a-1	Node I_1 s-a-0	Node I_1 s-a-1	Node J s-a-0	Node J s-a-1
Node K s-a-0	Node K s-a-1	Node L s-a-0	Node L s-a-1		

TASK 2

3.2.1

```
memory_initialization_radix=2;
memory_initialization_vector=
1100000,
0100001,
1000001,
1111001,
1101000,
1110000,
1100101,
1100011;
```

3.2.2

Can you explain why the minimum depth of a Xilinx distributed RAM is 16?

Distributed ram is made up of Look Up Tables (LUT) their minimum size in 16 bits.

3.2.3

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity top_level_TB is
end top_level_TB;

-- Tests the combinational circuit element of the ciruit
-- when circuit is in normal mode. The input INPUTS values are
-- tested through the combinational circuit giving
-- the resulting output value L_OUT. To set the circuit to normal mode
-- the B_RST reset button must be active(1) for a min of 3 clk cycles
-- for the debouncer to take effect, the circuit will then be in
-- normal mode.
-- The circuit can be set to automated test mode Via the B_TEST
--(test button) being active(1) for a min of 3 clk cycle. This will
-- test the inputs(MSB downto bit 1) to the expected output LSB of the
-- values within the RAM file. Each of the inputs is inserted into the
--combinational logic and the result compared to the expected output
value
-- for that addresses data.
-- For the automated mode 4 different faults can be inserted into the
circuit
-- via B_F 1:0,
-- 00 = no fault inserted
-- 01 =  Signal E stuck-at-1 fault inserted
-- 10 =  Signal H stuck-at-0 fault inserted
-- 11 =  Signal F stuck-at-0 fault inserted
-- each of these faults is tested against each value in the ram
addresses then
-- the next fault is inserted and tested agains the ram contents then
the next etc.
-- If/when the circuit detects the error from the inputs the circuit
stalls
-- and the output L_ERR is active(1) as error has been detected, the
output L_ID
-- will be stuck at the value of the ram address that the test inputs
originate from
-- the output value of each test from the inputs is still able to be
read from L_OUT
-- The only way to reset the circuit is making the B_RST active(1) min
3clk cycles.
-- this resets the circuit allowing for the circuit to be set to the
next testing
-- sequence/setup.

-- order of testing:
-- test inputs and output values NO AUTOMATION
-- test values from RAM with no fault injected Test Button Pressed
-- reset system
-- test values from RAM with E s-a-1  fault injected Test Button Pressed
-- reset system
-- test values from RAM with H s-a-0  fault injected Test Button Pressed
-- reset system
-- test values from RAM with F s-a-0 fault injected Test Button Pressed
-- reset system
architecture Behavioral of top_level_TB is

-- Constants
constant clk_period  : time := 10ns;
constant wait_period : time := 100ns;
```

```
--input signals
signal GCLK   : STD_LOGIC;
signal B_RST  : STD_LOGIC;
signal B_TEST : STD_LOGIC;
signal INPUTS : STD_LOGIC_VECTOR (5 downto 0);
signal B_F    : STD_LOGIC_VECTOR (1 downto 0);

--output signals
signal L_OUT : STD_LOGIC;
signal L_ERR : STD_LOGIC;
signal L_ID : STD_LOGIC_VECTOR (3 downto 0);

begin

UUT: entity work.top_level

    PORT MAP (GCLK     => GCLK ,
              B_RST    => B_RST,
              B_TEST   => B_TEST,
              INPUTS => INPUTS,
              B_F     => B_F,
              L_OUT     => L_OUT,
              L_ERR    => L_ERR,
              L_ID     => L_ID);

-- Clock process
clk_process :process
begin
 GCLK <= '0';
 wait for clk_period/2;
 GCLK <= '1';
 wait for clk_period/2;
end process;


--testing process
TEST_INPUTS : process
begin
-- wait 100 ns for global reset to finish set by Xilinx
wait for wait_period;

-- waits until falling edge of clock cycle
wait until falling_edge(GCLK);

-- clear registers in B_RST debouncer
-- clear registers in B_TEST debouncer
B_RST   <= '1';
B_TEST  <= '1';
wait for clk_period*4;
B_RST   <= '0';
B_TEST  <= '0';
wait for clk_period;


-- give INPUTS start value
INPUTS <= "000000";
```

```
- resets entire system
B_RST   <= '1';
wait for clk_period*4;
-- stops reseting system
B_RST   <= '0';
wait for clk_period;

--#############################################
--start testing
--#############################################
--test input values to output results
INPUTS <= "110000";
wait for clk_period;
INPUTS <= "010000";
wait for clk_period;
INPUTS <= "100000";
wait for clk_period;
INPUTS <= "111100";
wait for clk_period;
INPUTS <= "110100";
wait for clk_period;
INPUTS <= "111000";
wait for clk_period;
INPUTS <= "110010";
wait for clk_period;
INPUTS <= "110001";
wait for clk_period*3;


--reset system
B_RST   <= '1';
wait for clk_period*3;
-- stops reseting system
B_RST   <= '0';
wait for clk_period;
INPUTS <= "000000";



--###################################
--testing RAM IN/OUTPUTS
--###################################
--Fault inserted: no fault
B_F    <= "00";
--input test values 1
--INPUTS <= "110000";
-- start testing
B_TEST  <= '1';
wait for clk_period*3;
B_TEST  <= '0';
wait for clk_period*5;
--reset system
B_RST   <= '1';
wait for clk_period*3;
-- stops reseting system
B_RST   <= '0';
wait for clk_period;
```

```
--#############################################################
--testing RAM IN/OUTPUTS
--Fault inserted: E s-a-1
B_F    <= "01";
--input test values 1
--INPUTS <= "110000";
-- start testing
B_TEST  <= '1';
wait for clk_period*3;

B_TEST  <= '0';
wait for clk_period*5;
--reset system
B_RST   <= '1';
wait for clk_period*3;
-- stops reseting system
B_RST   <= '0';
wait for clk_period*5;
----#############################################
--testing RAM IN/OUTPUTS
--Fault inserted: H s-a-0
B_F    <= "10";
--input test values 1
--INPUTS <= "110000";
-- start testing
B_TEST  <= '1';
wait for clk_period*3;

B_TEST  <= '0';
wait for clk_period*5;
--reset system
B_RST   <= '1';
wait for clk_period*3;
-- stops reseting system
B_RST   <= '0';
wait for clk_period*5;
--#############################################
--testing RAM IN/OUTPUTS
--Fault inserted: F s-a-0
B_F    <= "11";
--input test values 1
--INPUTS <= "110000";
-- start testing
B_TEST  <= '1';
wait for clk_period*3;

B_TEST  <= '0';
wait for clk_period*5;
--reset system
B_RST   <= '1';
wait for clk_period*3;
-- stops reseting system
B_RST   <= '0';
wait for clk_period*5;

 wait; --waits forever
 end process;
```

3.2.4

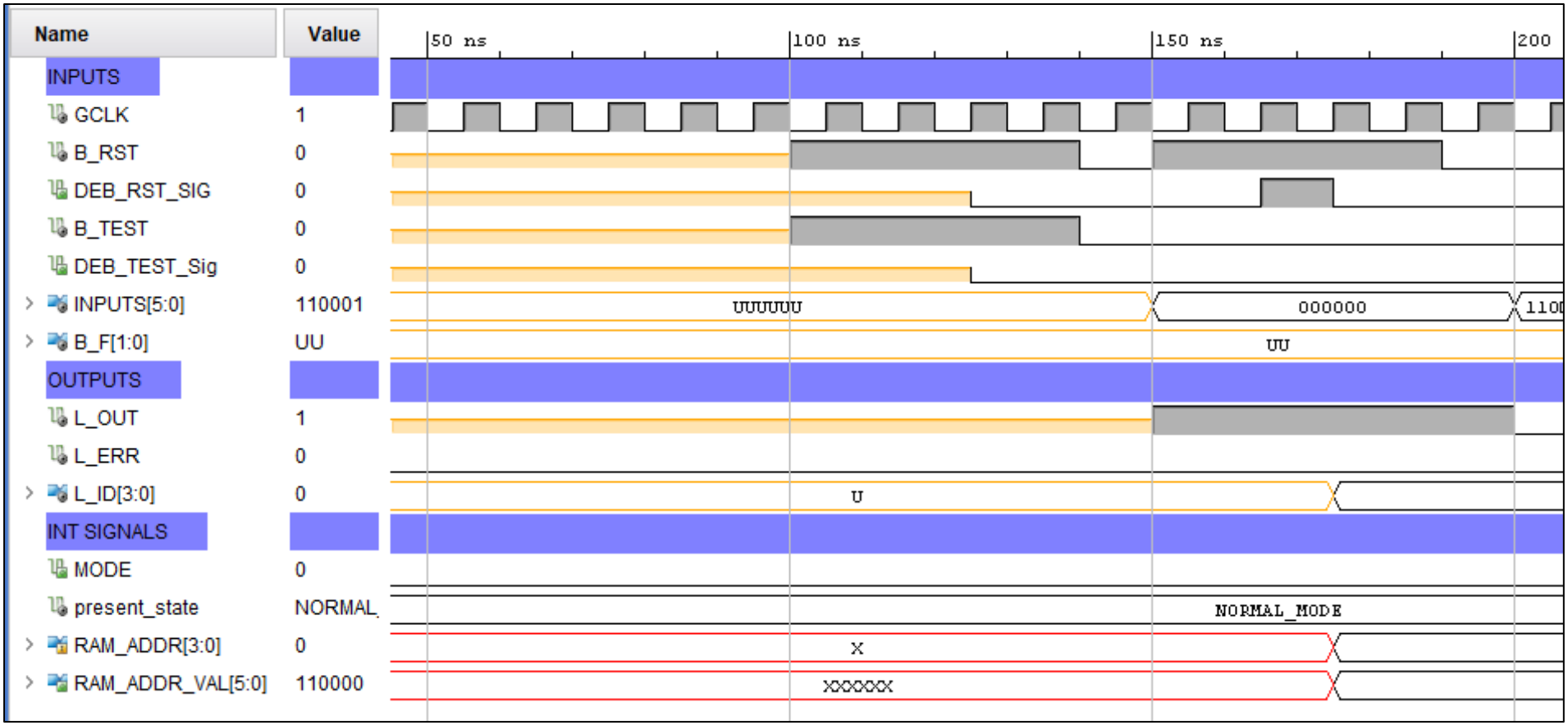


Figure 1 circuit initialisation, reset and debouncer registers given known values

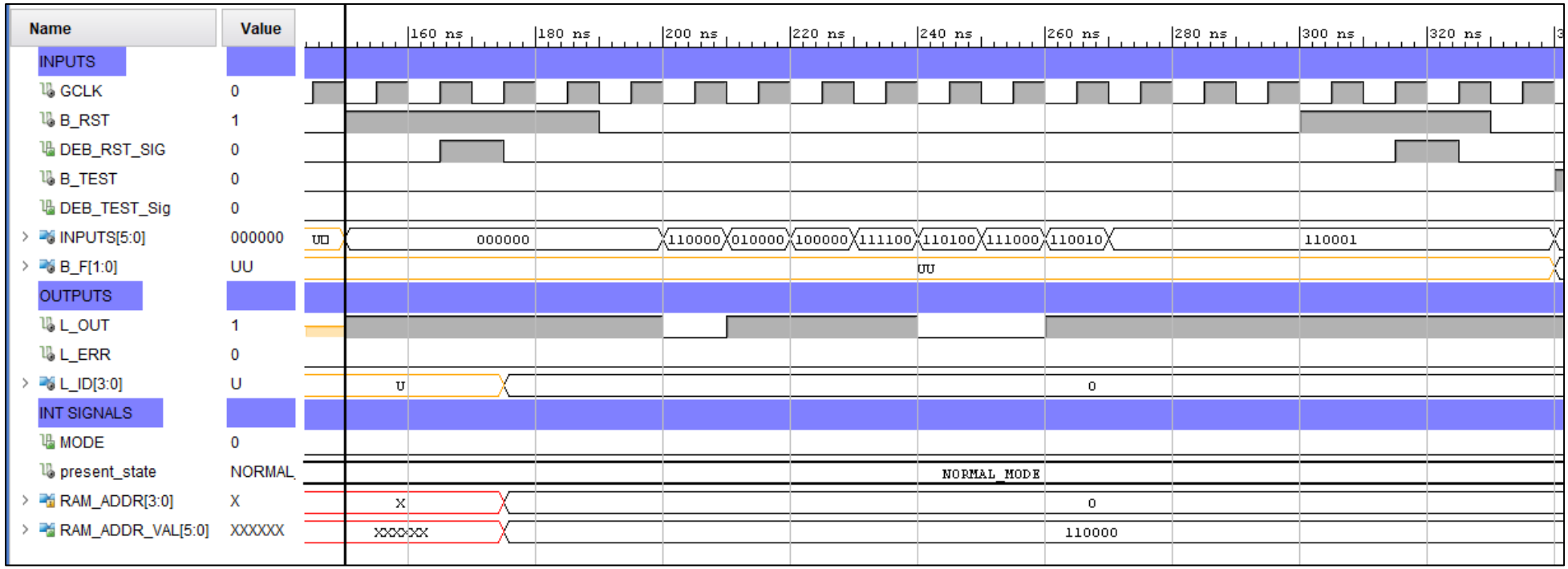


Figure 2 Operation of a Fault Free Circuit

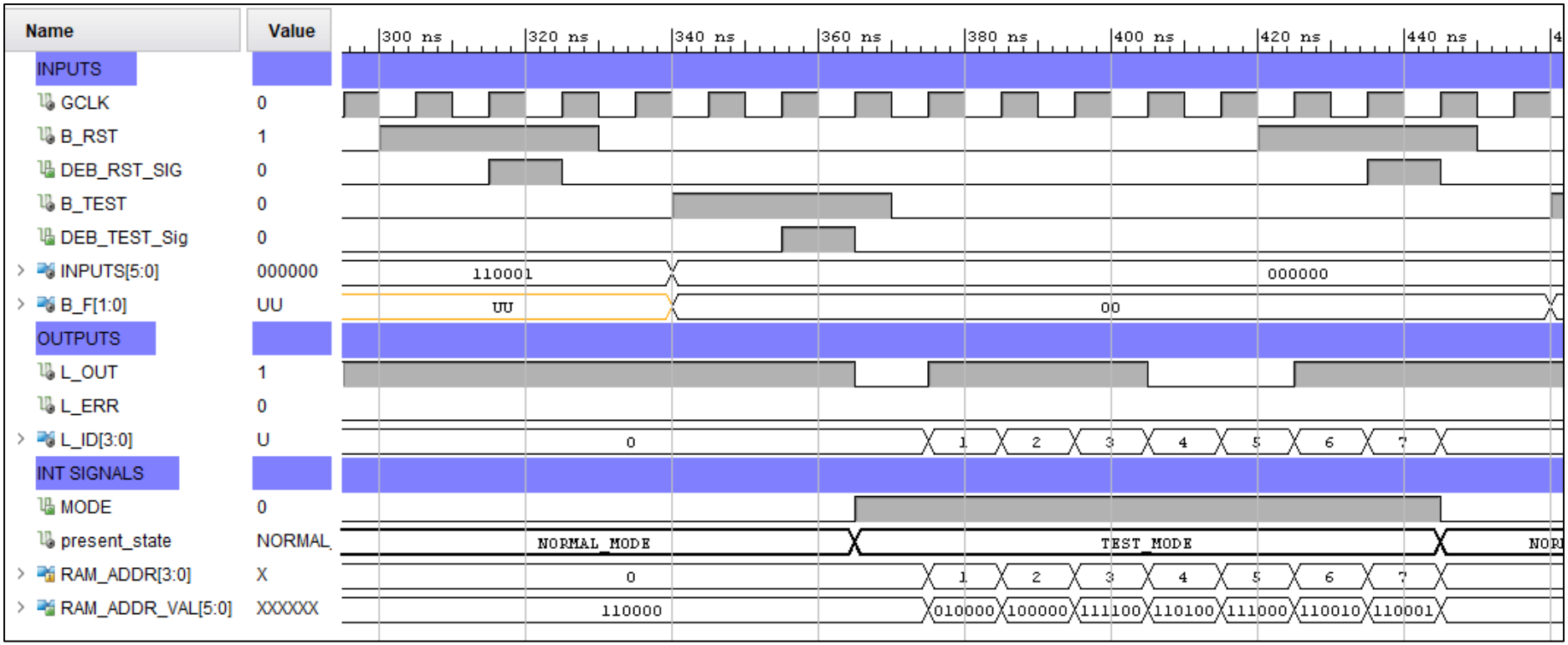


Figure 3 Test cycle no fault injected into system

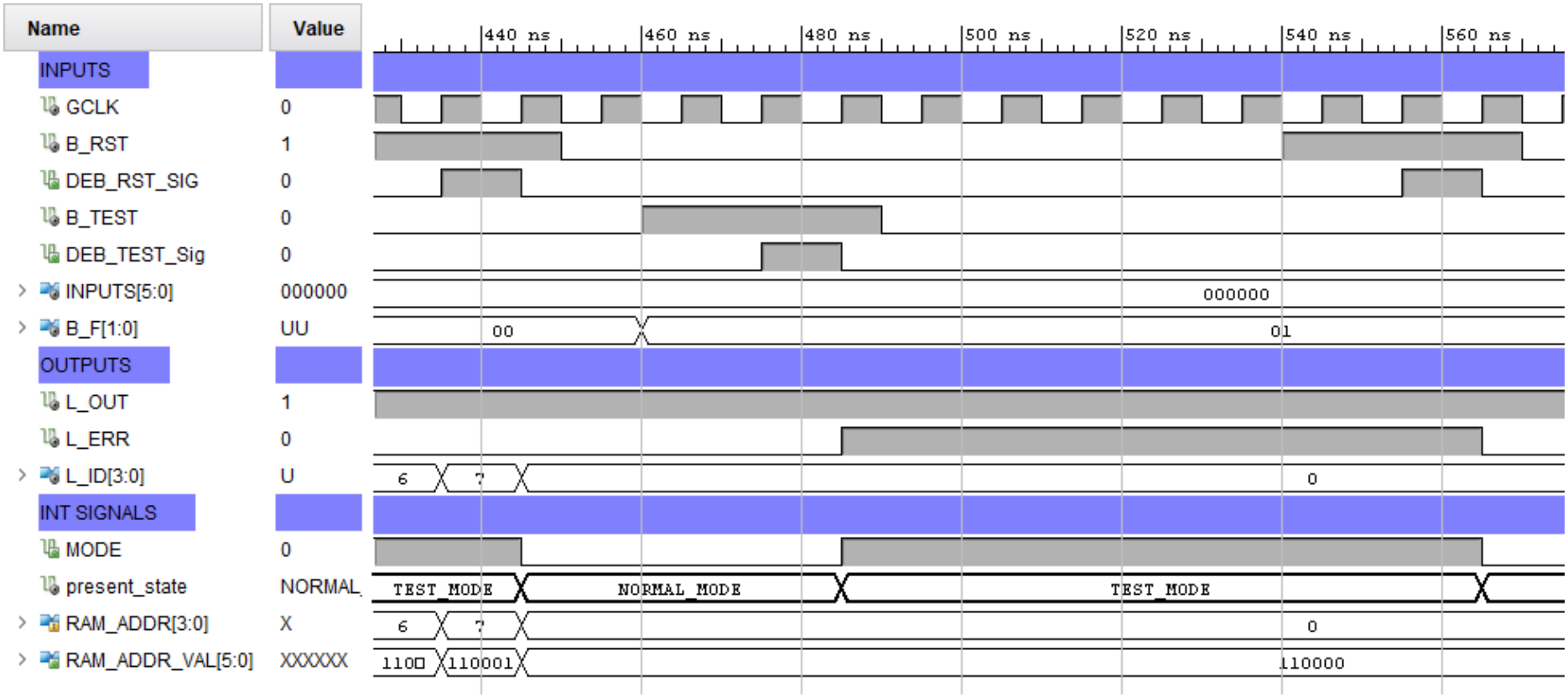


Figure 4 Test cycle E s-a-1 fault injected into system

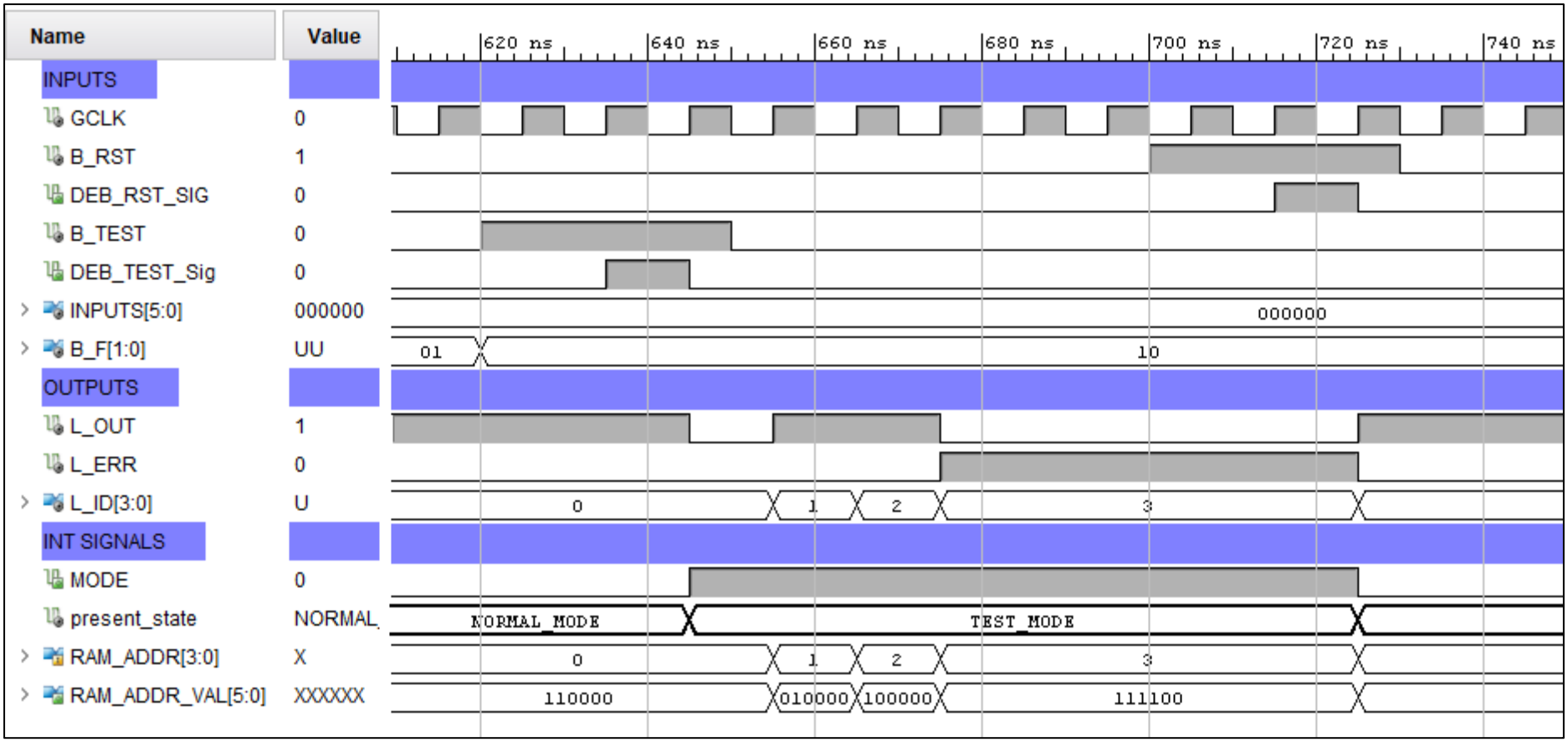


Figure 5 Test cycle H s-a-0 fault injected into system

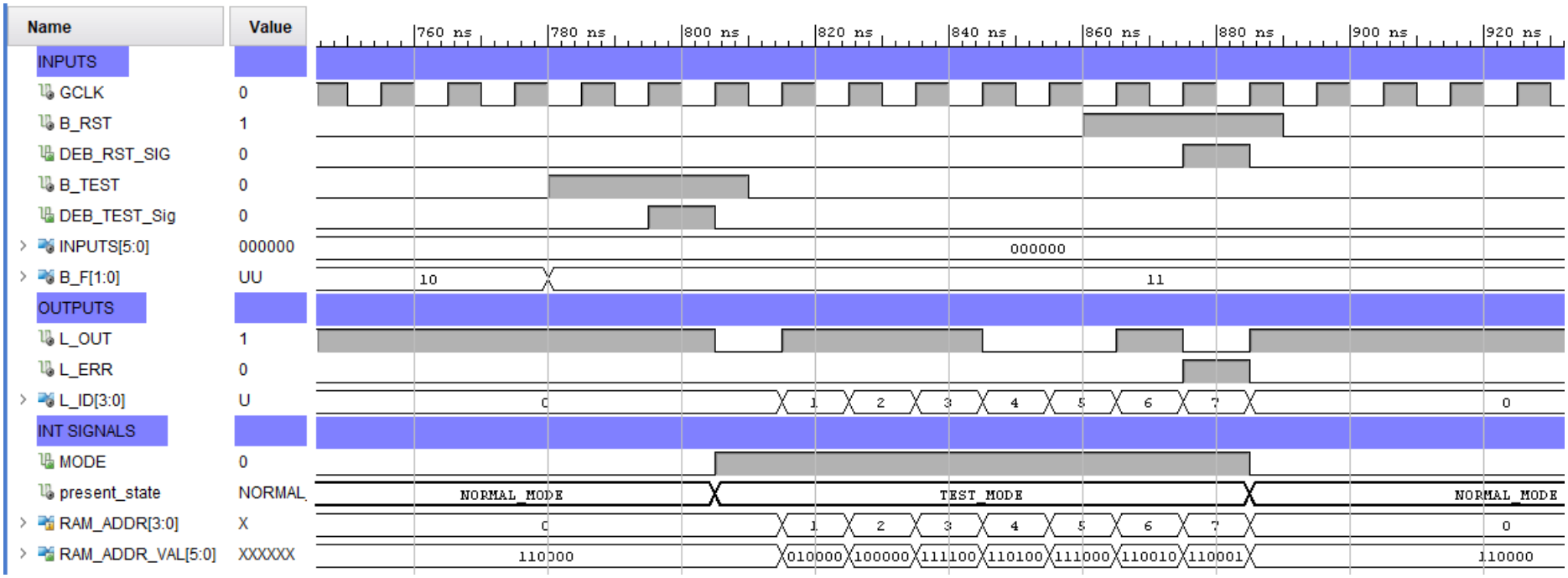


Figure 6 Test cycle F s-a-0 fault injected into system

Task 3

3.3.1

Duty Cycle:

$$Duty\ cycle\ = \frac{Time\ High}{Period} \times 100$$

It is the ratio/percentage of the signal being on to off within the clock period

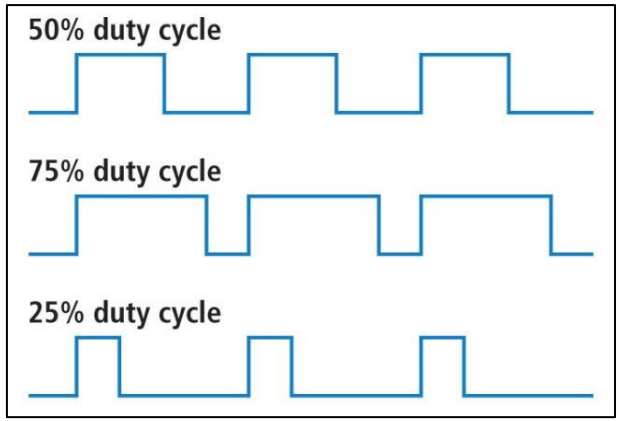


Figure 7 Duty Cycle Example (SparkFun Electronics , 2020)

Phase shift:

Phase shift occurs when there is a time delay between two signals that have identical frequency or period. The phase shift how far one of the signals is shifted horizontally along the X (time) axis compared to the other this value can be negative or positive and is normally measured in degrees or radians.

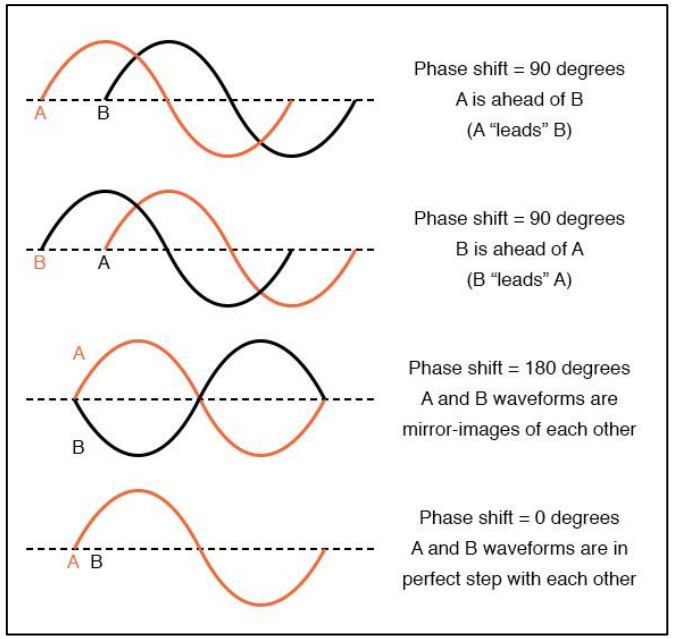
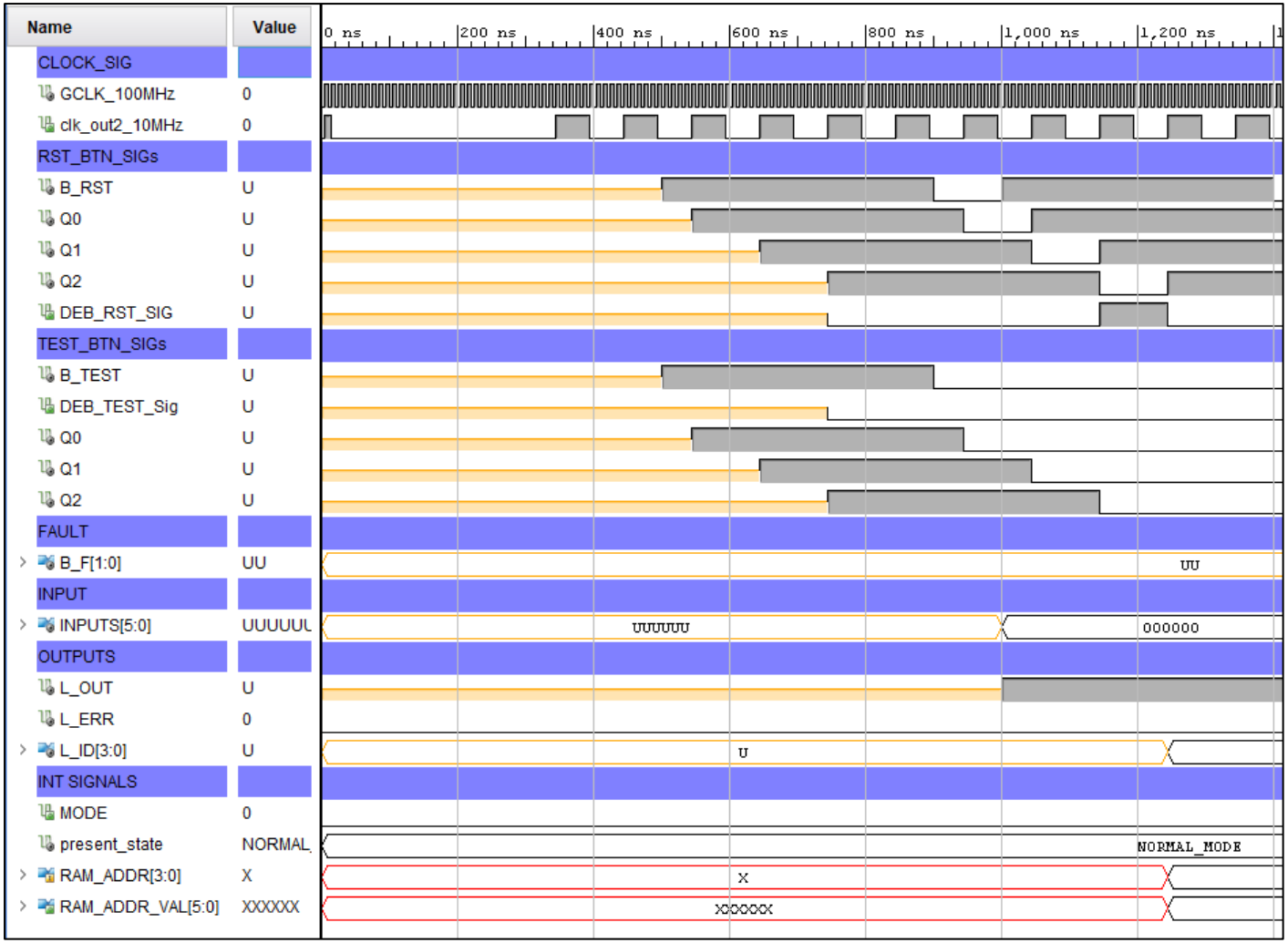
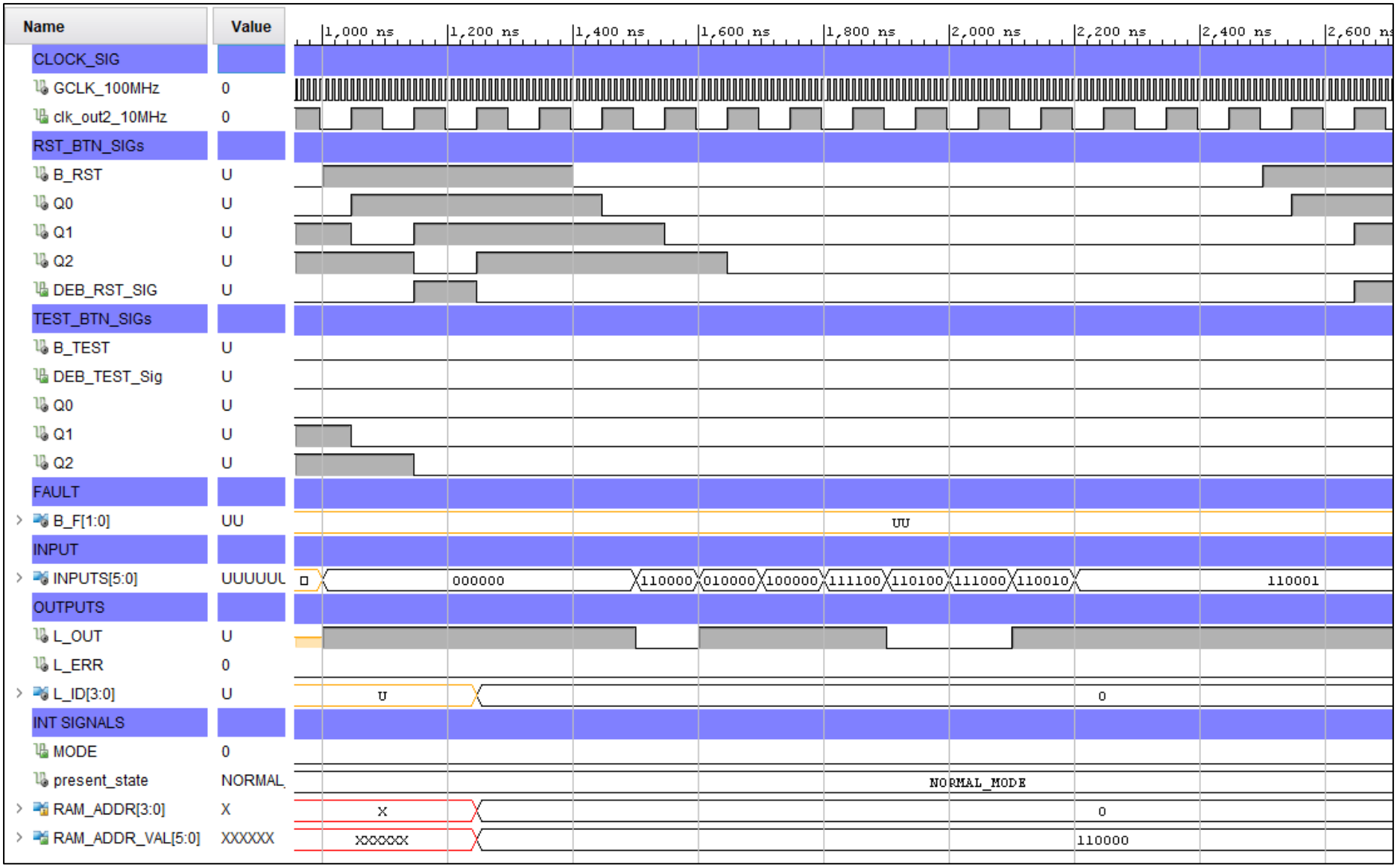
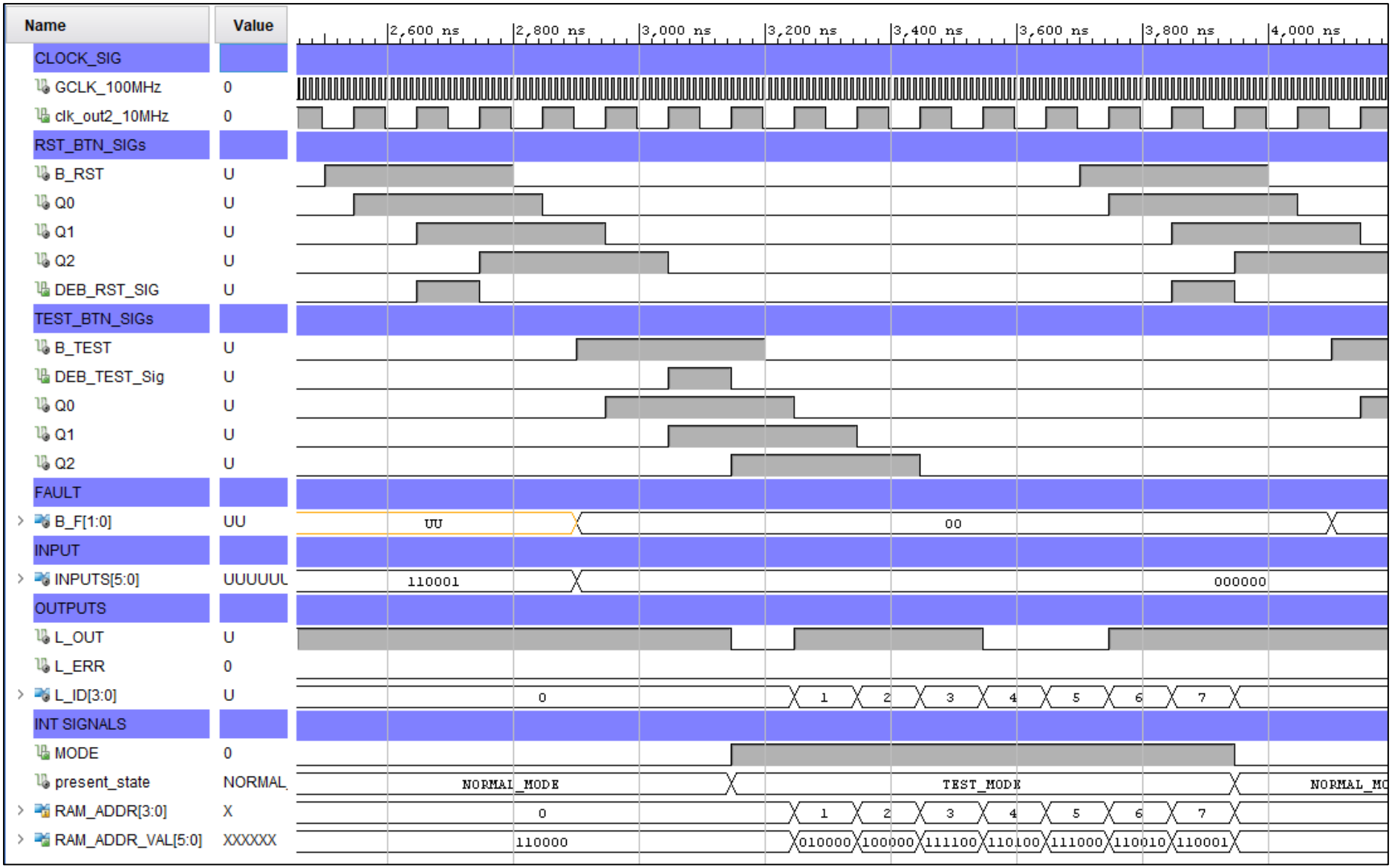


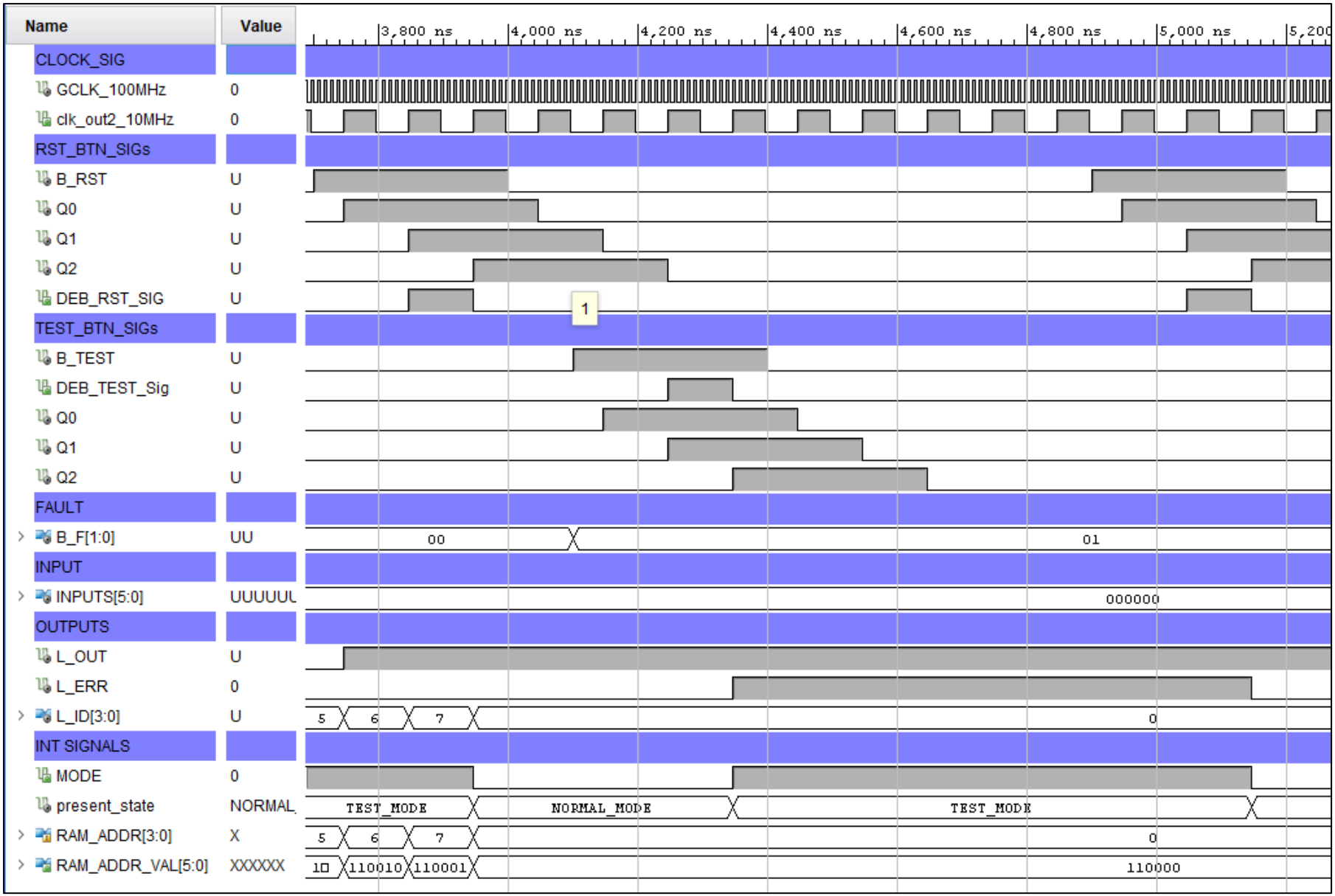
Figure 8 Phase Shift image (EETech Media,LLC, 2020)

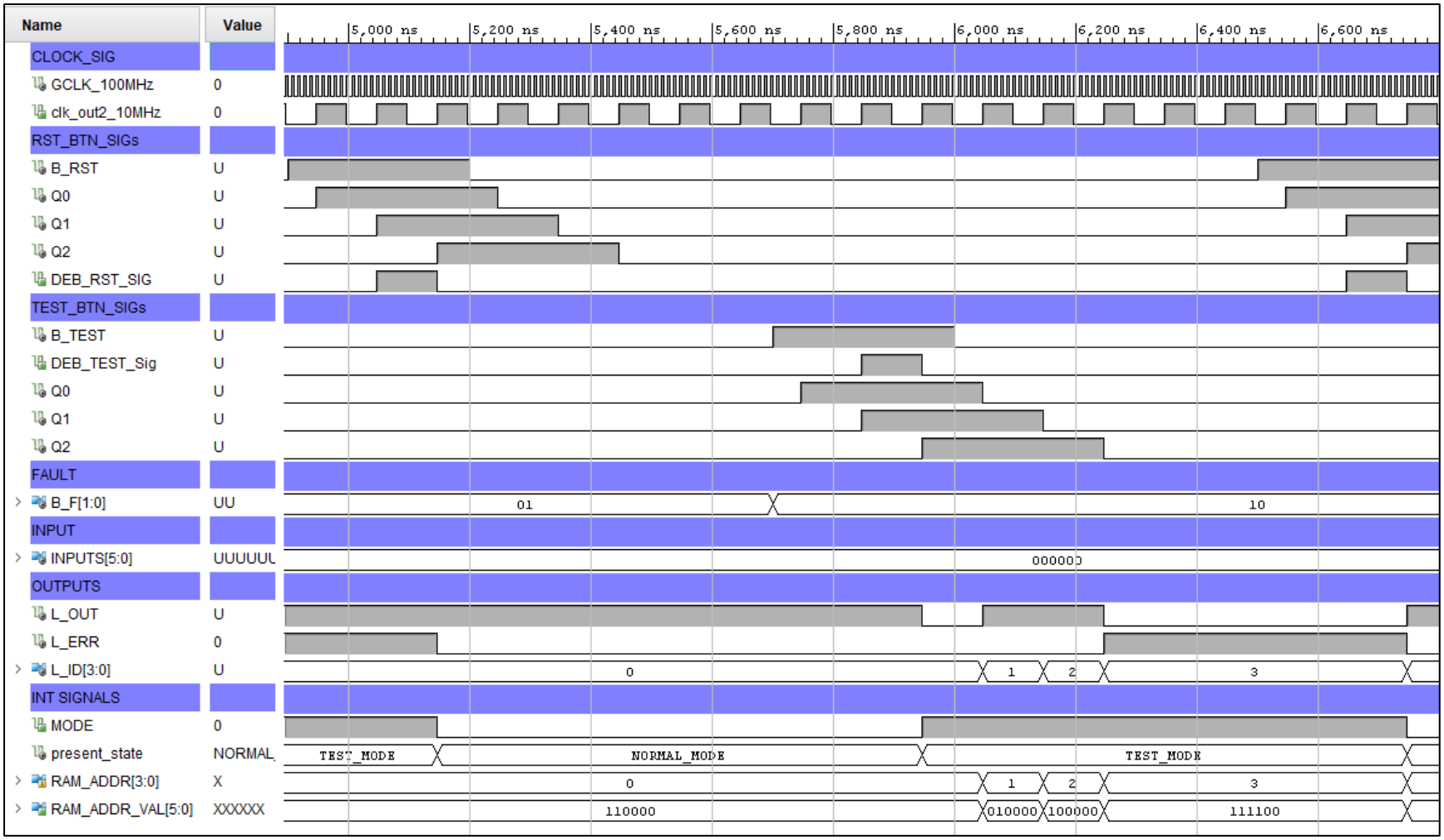
3.3.2

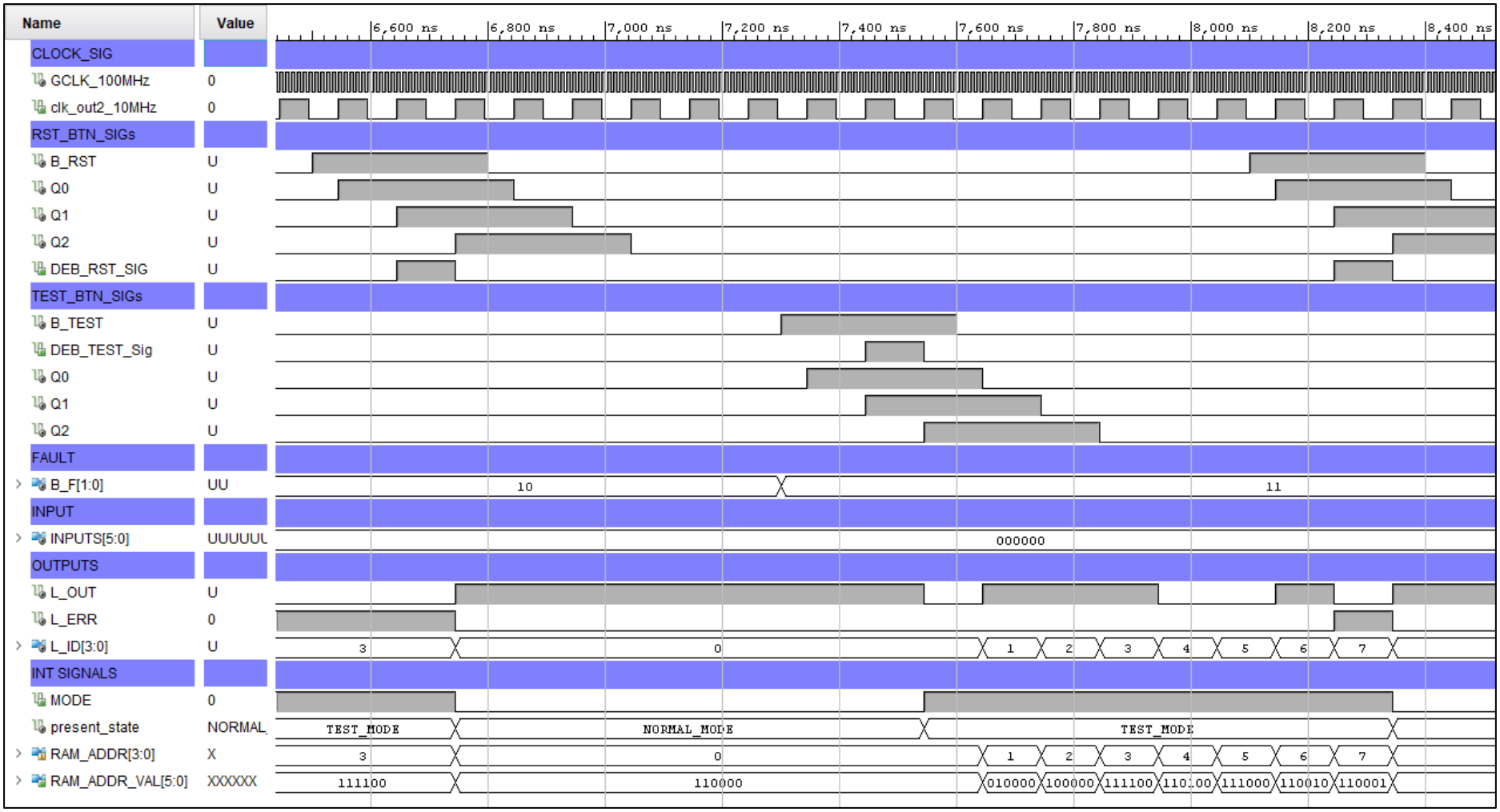

Name
Value
0 ns
200 ns
400 ns
600 ns
800 ns
1,000 ns
1,200 ns
CLOCK_SIG
GCLK_100MHz 0
clk_out2_10MHz 0
RST_BTN_SIGs
B_RST U
Q0 U
Q1 U
Q2 U
DEB_RST_SIG U
TEST_BTN_SIGs
B_TEST U
DEB_TEST_Sig U
Q0 U
Q1 U
Q2 U
FAULT
B_F[1:0] UU
UU
INPUT
INPUTS[5:0] UUUUUU
UUUUUU
000000
OUTPUTS
L_OUT U
L_ERR 0
L_ID[3:0] U
U
INT SIGNALS
MODE 0
present_state NORMAL
NORMAL_MODE
RAM_ADDR[3:0] X
X
RAM_ADDR_VAL[5:0] XXXXXX
XXXXXX

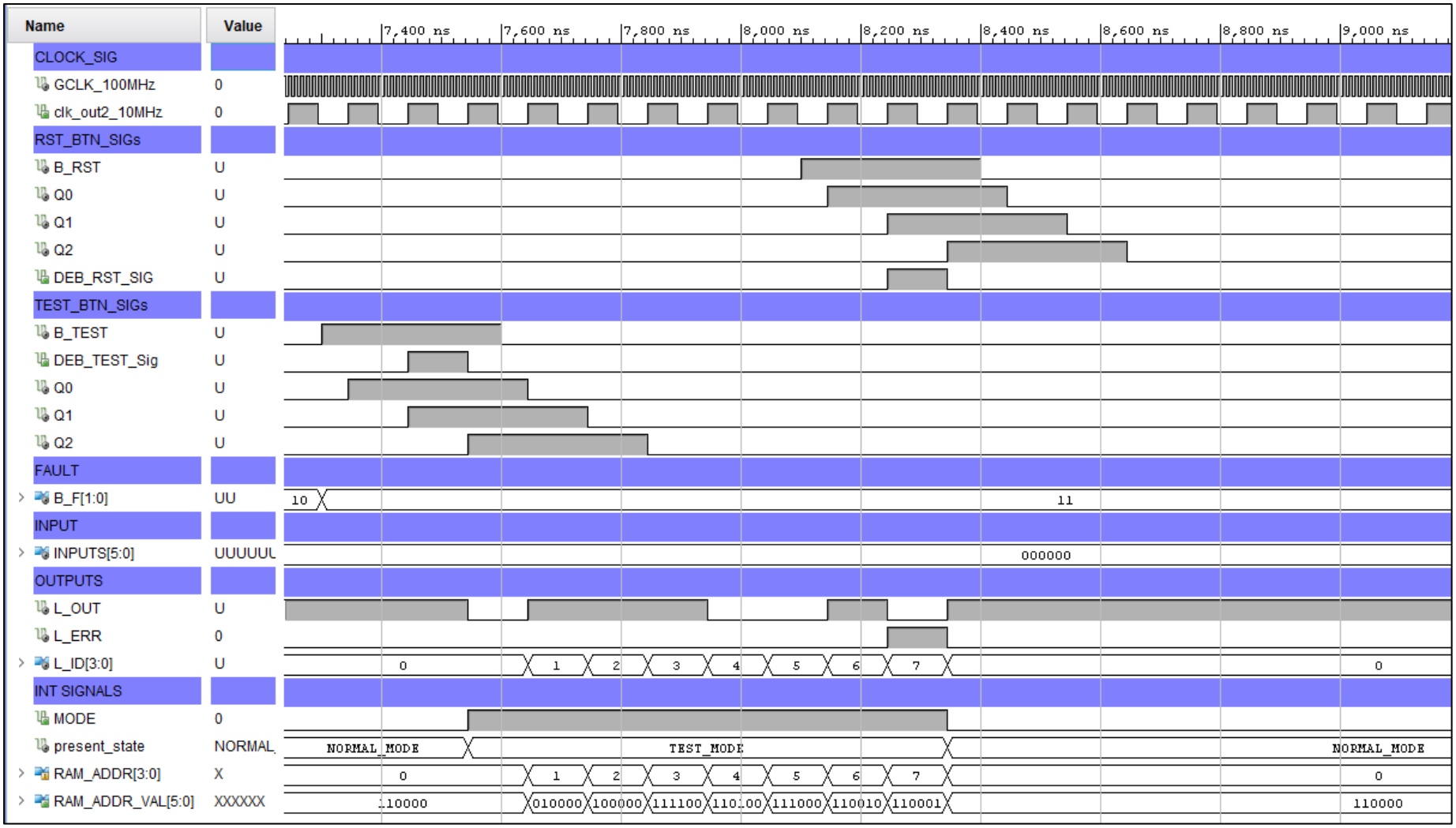

Name
Value
1,000 ns
1,200 ns
1,400 ns
1,600 ns
1,800 ns
2,000 ns
2,200 ns
2,400 ns
2,600 ns
CLOCK_SIG
GCLK_100MHz
0
clk_out2_10MHz
0
RST_BTN_SIGs
B_RST
U
Q0
U
Q1
U
Q2
U
DEB_RST_SIG
U
TEST_BTN_SIGs
B_TEST
U
DEB_TEST_Sig
U
Q0
U
Q1
U
Q2
U
FAULT
B_F[1:0]
UU
UU
INPUT
INPUTS[5:0]
UUUUUU
000000
110000
010000
100000
111100
110100
111000
110010
110001
OUTPUTS
L_OUT
U
L_ERR
0
L_ID[3:0]
U
U
0
INT SIGNALS
MODE
0
present_state
NORMAL
NORMAL_MODE
RAM_ADDR[3:0]
X
X
0
RAM_ADDR_VAL[5:0]
XXXXXX
XXXXXX
110000


Name
Value
2,600 ns
2,800 ns
3,000 ns
3,200 ns
3,400 ns
3,600 ns
3,800 ns
4,000 ns
CLOCK_SIG
GCLK_100MHz
0
clk_out2_10MHz
0
RST_BTN_SIGs
B_RST
U
Q0
U
Q1
U
Q2
U
DEB_RST_SIG
U
TEST_BTN_SIGs
B_TEST
U
DEB_TEST_Sig
U
Q0
U
Q1
U
Q2
U
FAULT
B_F[1:0]
UU
UU
00
INPUT
INPUTS[5:0]
UUUUUU
110001
000000
OUTPUTS
L_OUT
U
L_ERR
0
L_ID[3:0]
U
0
1
2
3
4
5
6
7
INT SIGNALS
MODE
0
present_state
NORMAL
NORMAL_MODE
TEST_MODE
NORMAL_MO
RAM_ADDR[3:0]
X
0
1
2
3
4
5
6
7
RAM_ADDR_VAL[5:0]
XXXXXX
110000
010000
100000
111100
110100
111000
110010
110001


Name
Value
3,800 ns
4,000 ns
4,200 ns
4,400 ns
4,600 ns
4,800 ns
5,000 ns
CLOCK_SIG
GCLK_100MHz
0
clk_out2_10MHz
0
RST_BTN_SIGs
B_RST
U
Q0
U
Q1
U
Q2
U
DEB_RST_SIG
U
1
TEST_BTN_SIGs
B_TEST
U
DEB_TEST_Sig
U
Q0
U
Q1
U
Q2
U
FAULT
B_F[1:0]
UU
00
01
INPUT
INPUTS[5:0]
UUUUUU
000000
OUTPUTS
L_OUT
U
L_ERR
0
L_ID[3:0]
U
5
6
7
0
INT SIGNALS
MODE
0
present_state
NORMAL
TEST_MODE
NORMAL_MODE
TEST_MODE
RAM_ADDR[3:0]
X
5
6
7
0
RAM_ADDR_VAL[5:0]
XXXXXX
110010
110001
110000


Name
Value
5,000 ns
5,200 ns
5,400 ns
5,600 ns
5,800 ns
6,000 ns
6,200 ns
6,400 ns
6,600 ns
CLOCK_SIG
GCLK_100MHz 0
clk_out2_10MHz 0
RST_BTN_SIGs
B_RST U
Q0 U
Q1 U
Q2 U
DEB_RST_SIG U
TEST_BTN_SIGs
B_TEST U
DEB_TEST_Sig U
Q0 U
Q1 U
Q2 U
FAULT
B_F[1:0] UU
01
10
INPUT
INPUTS[5:0] UUUUUU
000000
OUTPUTS
L_OUT U
L_ERR 0
L_ID[3:0] U
0
1
2
3
INT SIGNALS
MODE 0
present_state NORMAL
TEST_MODE
NORMAL_MODE
TEST_MODE
RAM_ADDR[3:0] X
0
1
2
3
RAM_ADDR_VAL[5:0] XXXXXX
110000
010000
100000
111100


Name
Value
6,600 ns
6,800 ns
7,000 ns
7,200 ns
7,400 ns
7,600 ns
7,800 ns
8,000 ns
8,200 ns
8,400 ns
CLOCK_SIG
GCLK_100MHz 0
clk_out2_10MHz 0
RST_BTN_SIGs
B_RST U
Q0 U
Q1 U
Q2 U
DEB_RST_SIG U
TEST_BTN_SIGs
B_TEST U
DEB_TEST_Sig U
Q0 U
Q1 U
Q2 U
FAULT
B_F[1:0] UU
10
11
INPUT
INPUTS[5:0] UUUUUU
000000
OUTPUTS
L_OUT U
L_ERR 0
L_ID[3:0] U
3
0
1
2
3
4
5
6
7
INT SIGNALS
MODE 0
present_state NORMAL
TEST_MODE
NORMAL_MODE
TEST_MODE
RAM_ADDR[3:0] X
3
0
1
2
3
4
5
6
7
RAM_ADDR_VAL[5:0] XXXXXX
111100
110000
010000
100000
111100
111000
111000
110010
110001


Name
Value
7,400 ns
7,600 ns
7,800 ns
8,000 ns
8,200 ns
8,400 ns
8,600 ns
8,800 ns
9,000 ns
CLOCK_SIG
GCLK_100MHz 0
clk_out2_10MHz 0
RST_BTN_SIGs
B_RST U
Q0 U
Q1 U
Q2 U
DEB_RST_SIG U
TEST_BTN_SIGs
B_TEST U
DEB_TEST_Sig U
Q0 U
Q1 U
Q2 U
FAULT
B_F[1:0] UU
10
11
INPUT
INPUTS[5:0] UUUUUU
000000
OUTPUTS
L_OUT U
L_ERR 0
L_ID[3:0] U
0 1 2 3 4 5 6 7 0
INT SIGNALS
MODE 0
present_state NORMAL
NORMAL_MODE
TEST_MODE
NORMAL_MODE
RAM_ADDR[3:0] X
0 1 2 3 4 5 6 7 0
RAM_ADDR_VAL[5:0] XXXXXX
110000
010000
100000
111100
110100
111000
110010
110001
110000

3.3.3

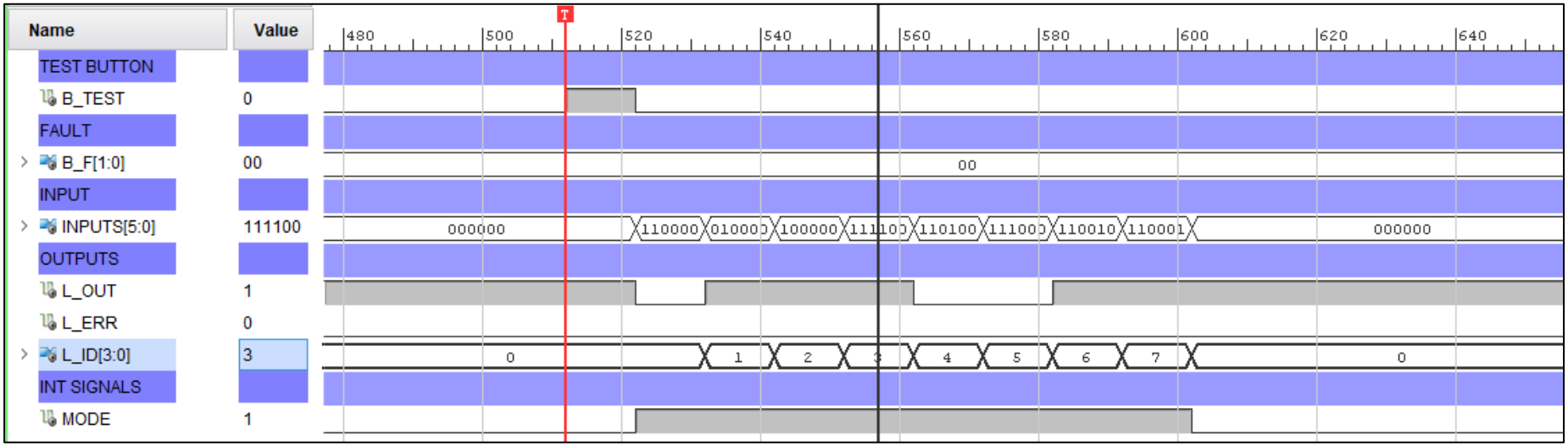
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

-- Top level entity for the BIST system. The circuit implements a
--   combinational UUT with 5 input and 1 output STD_LOGIC signals.
--   The unit includes fault injection logic for three stuck-at faults.
-- Until the B_TST button is pressed, the UUT operates normally on
--   the inputs set by the user using board switches.
-- When the use presses the B_TST button, the circuit de-gates the
--   switch inputs and performs a self-test by sending a set of test
--   patterns to the UUT (one per clock cycle) and comparing the output
--   of the UUT to the known-good response.
-- If no fault is detected, the unit exits test mode and resumes
--   normal operation.
-- If a fault is detected, the circuit freezes, sets the L_ERR output
--   to '1' and outputs the address of the test pattern that detected
--   the fault on the L_ID lines.
-- Note: to prevent collisions, the selected fault is "locked" during
--   the test process.

-- #UPDATE for assignment 17-02-2020#
-- input clock reduced from 100MHz down to 10MHz
-- UUT still uses 100MHz clk
-- debouncers, BIST and SET_FLT register uses 10MHzclk
-- logic analyser
-- analyses the signals below within the physical circuit
--within the physical chip
-- B_TEST_DB  - start test button also the trigger signal
-- VECTOR     - input test values
-- UUT_OUT    - output of uni under test
-- B_F_SET    - fault to be injected selection
-- L_ERR_INT  - internal error detected
-- L_ID_INT   - ID within ram of input values
-- MODE_INT   - current mode normal or testing state
-- #END OF UPDATE#

entity top_level is
Port ( GCLK : in  STD_LOGIC; -- 100MHz global clock
       -- Active-high global reset (BTNL button)
       B_RST : in  STD_LOGIC;
       -- User inputs to the UUT (switches 5:0)
       INPUTS : in  STD_LOGIC_VECTOR (5 downto 0);
       -- Active-high input that starts BIST (BTNC button)
       B_TEST : in  STD_LOGIC;
       -- Select fault to be injected (switches 7:6)
       --   00 = no fault
       --   01 = E s-a-1
       --   10 = H s-a-0
       --   11 = F s-a-0
       B_F : in  STD_LOGIC_VECTOR (1 downto 0);
       L_OUT : out  STD_LOGIC; -- Output of the UUT (LED0)
       -- Signal goes high when the BIST detects an error (LED2)
       L_ERR : out  STD_LOGIC;
       -- Outputs ID of the error, i.e. the address of the test
       --   pattern that has detected the error (LED7:4)
       L_ID : out  STD_LOGIC_VECTOR (3 downto 0));
end top_level;
```

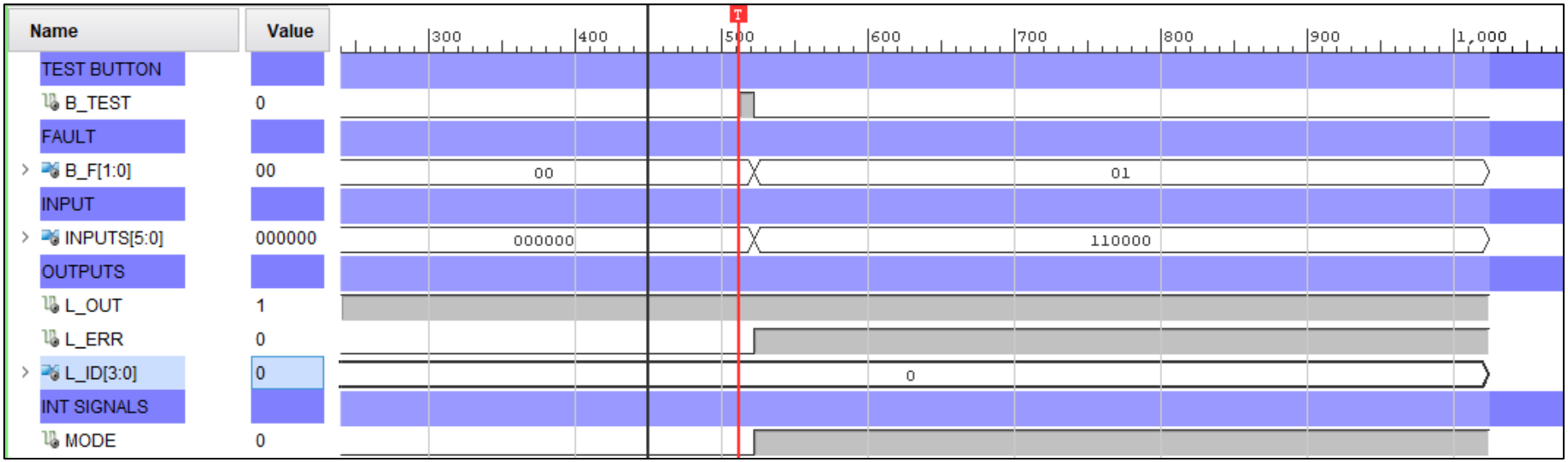
```
architecture Behavioral of top_level is

    -- Test patterns generated by the BIST unit
      signal TEST : STD_LOGIC_VECTOR (5 downto 0);
      -- Inputs to the UUT: user inputs during normal operation
      --   or test patterns during test
      signal VECTOR : STD_LOGIC_VECTOR (5 downto 0);
      -- Fault selection (locked during test)
      signal B_F_SET : STD_LOGIC_VECTOR (1 downto 0);
      -- Flag for the operating mode of the circuit:
      --   '0' during normal operation
      --   '1' during test
      signal MODE: STD_LOGIC;
      -- Debounced control inputs
      signal B_RST_DB, B_TEST_DB: STD_LOGIC;
      -- Output of the UUT
      signal UUT_OUT: STD_LOGIC;
    -- 10MHz clock
    signal CLK_10MHz: STD_LOGIC;

    --start #logic analyser ouput internal signals#
    -- internal error detected signal
    signal L_ERR_INT : STD_LOGIC_VECTOR(0 DOWNTO 0);
    -- internal address in ram of error detected
    signal L_ID_INT  : STD_LOGIC_VECTOR(3 DOWNTO 0);
    -- internal mode of circuit signal
    signal MODE_INT : STD_LOGIC_VECTOR(0 DOWNTO 0);
    --END #logic analyser ouput internal signals#

    --tell the tools not to touch those signals that we want to observe
    --(normally, they would be removed or transformed during the
synthesis process).
    attribute keep : string;
    attribute keep of B_TEST_DB : signal is "true";
    attribute keep of VECTOR : signal is "true";
    attribute keep of UUT_OUT : signal is "true";
    attribute keep of B_F_SET : signal is "true";
    attribute keep of L_ERR_INT : signal is "true";
    attribute keep of L_ID_INT : signal is "true";
    attribute keep of MODE : signal is "true";


begin

-- Debouncers for the reset and test buttons.
Inst_Debouncer_RST: entity work.Debouncer PORT MAP(
  CLK => CLK_10MHz,
  Sig => B_RST,
  Deb_Sig => B_RST_DB
);

Inst_Debouncer_TEST: entity work.Debouncer PORT MAP(
  CLK => CLK_10MHz,
  Sig => B_TEST,
  Deb_Sig => B_TEST_DB
);
```

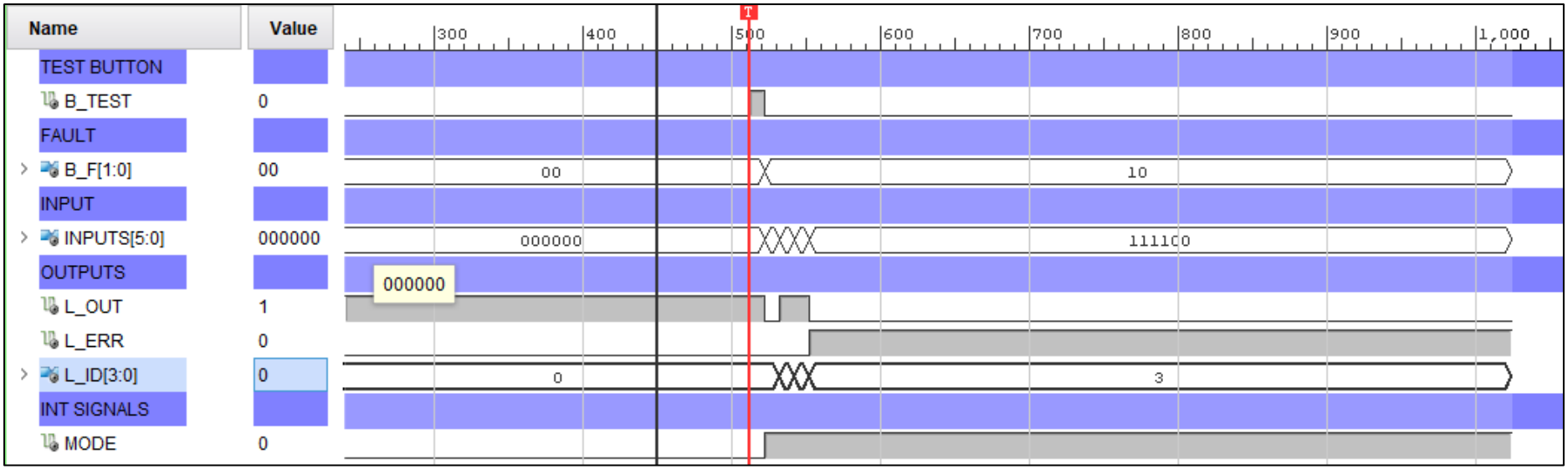
```
-- This register "locks" the fault selected by the B_F switches when
--  the B_TEST button is pressed and does not allow any changes until
--  the test is complete.
-- Note that if the BIST detects a fault, this will lock "mode" to '1'
--  and the register will only be cleared by the global reset.
SET_FLT: process (CLK_10MHz) is
begin
  if rising_edge(CLK_10MHz) then
    if B_RST_DB = '1' then
      B_F_SET <= (others => '0');
    elsif B_TEST_DB = '1' and MODE = '0' then
      B_F_SET <= B_F;
    end if;
  end if;
end process SET_FLT;

-- Circuit that contains the UUT, including fault injection logic.
Inst_UUT: entity work.UUT PORT MAP(
  FLT => B_F_SET,
  INPUTS => VECTOR,
  OUTPUT => UUT_OUT
);

-- This entity implements the BIST subsystem.
Inst_BIST: entity work.BIST_Ctrl PORT MAP(
  CLK => CLK_10MHz,
  B_RST => B_RST_DB,
  B_TEST => B_TEST_DB,
  UUT_OUT => UUT_OUT,
  TEST => TEST,
  MODE => MODE_INT(0),
  L_ERR => L_ERR_INT(0),
  L_ID => L_ID_INT
);

-- Mux used to de-gate the inputs when the circuit is in test mode.
-- User inputs will be replaced by the test patterns output by the
--   BIST unit.
VECTOR <= TEST WHEN MODE = '1' ELSE INPUTS;

-- Maps the output of the UUT to one of the LEDs.
L_OUT <= UUT_OUT;

-- clock manger reduces 100MHz clk signal downto 10MHz
-- 10 MHz signal used on debouncers, SET_FLT register
-- and the BIST subsystem as clock signal
dcm_mgr : entity work.clock_manager
port map
(
-- clock in ports
CLK_IN1 => GCLK,
-- clock ports
-- CLK_OUT1 =>
CLK_OUT2 => CLK_10MHz);
```

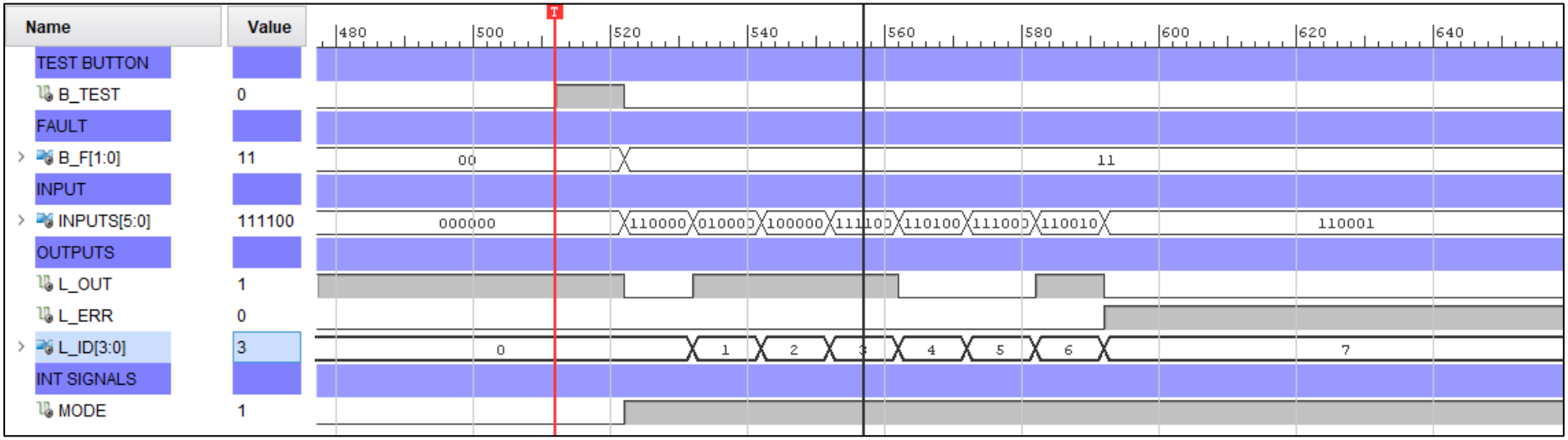
```
-- Logic analyster
-- used to observe signals changeing on board when the
-- hardware has bit stream implmented upon it.
-- setup to use B_TEST as trigger button
-- signal levels are analysed in the physical circuit
-- once the logic analyser is triggered
ILA: ENTITY work.ila_0
PORT MAP (
 clk => gclk,
 probe0(0) => B_TEST_DB,
 probe1 => VECTOR,
 probe2(0) => UUT_OUT,
 probe3 => B_F_SET,
 probe4(0) => L_ERR_INT(0),
 probe5 => L_ID_INT,
 probe6(0) => MODE_INT(0));


 -- Internal signal connections to outputs
  L_ERR <= L_ERR_INT(0);
  L_ID  <= L_ID_INT;
  MODE  <= MODE_INT (0);
end Behavioral;
```

3.3.4


Name
Value
TEST BUTTON
B_TEST
0
FAULT
B_F[1:0]
00
INPUT
INPUTS[5:0]
111100
OUTPUTS
L_OUT
1
L_ERR
0
L_ID[3:0]
3
INT SIGNALS
MODE
1
480
500
520
540
560
580
600
620
640
T
000000
110000
010000
100000
111100
110100
111000
110010
110001
000000
00
0
1
2
3
4
5
6
7
0

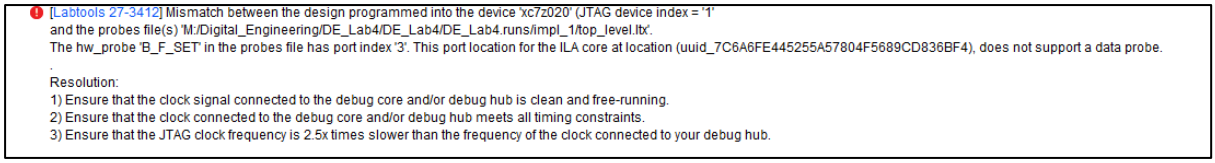

Name
Value
TEST BUTTON
B_TEST
0
FAULT
B_F[1:0]
00
INPUT
INPUTS[5:0]
000000
OUTPUTS
L_OUT
1
L_ERR
0
L_ID[3:0]
0
INT SIGNALS
MODE
0
300
400
500
600
700
800
900
1,000
T
00
01
000000
110000
0


Name
Value
TEST BUTTON
B_TEST
0
FAULT
B_F[1:0]
00
INPUT
INPUTS[5:0]
000000
OUTPUTS
L_OUT
1
L_ERR
0
L_ID[3:0]
0
INT SIGNALS
MODE
0
300
400
500
600
700
800
900
1,000
00
10
000000
111100
000000
0
3


Name
Value
TEST BUTTON
B_TEST
0
FAULT
B_F[1:0]
11
INPUT
INPUTS[5:0]
111100
OUTPUTS
L_OUT
1
L_ERR
0
L_ID[3:0]
3
INT SIGNALS
MODE
1
480
500
520
540
560
580
600
620
640
00
11
000000
110000
010000
100000
111100
110100
111000
110010
110001
0
1
2
3
4
5
6
7

3.3.5

What do you think would have changed if we had used the 10MHz clock to clock the logic analyser instead of the 100MHz clock? Would you still expect to be able to observe the internal data signals, which also switch at 10MHz? Explain your reasoning and feel free to experiment.



You cannot use the logic analyser if the 10MHz clk signal is connected to it instead of 100MHz clk signal. I originally thought you would have not been able to due to a sampling issue but according to the error report it is because “the JTAG clock frequency needs to be 2.5X slower than the clock connected to the debug hub”. I take this to mean that if the reduced clk signal needs to be 40% or lower speed of the clock connected to the debug hub.

Theoretically there should be no issue with running the logic analyser at 10MHz instead of the 100MHz as both have the same clock edge signal so there is no chance of a signal being missed normally Nyquist theorem should be taken into account and the sample clk signal should be at least double the frequency of the signal being sampled.

References

EETech Media,LLC. (2020, February 17th). *All About Circuits*. Retrieved from AC phase: https://www.allaboutcircuits.com/textbook/alternating-current/chpt-1/ac-phase/

SparkFun Electronics . (2020, February 17th). *Sparkfun Start Something*. Retrieved from Pulse Width modulation: https://learn.sparkfun.com/tutorials/pulse-width-modulation/duty-cycle