Lab 4 - A

3.1.1. For each kind of logic gate in the circuit (AND, OR, NOR), specify equivalent and dominated/dominant faults. Show your derivation.

AND Gate

Without faults, for the output to be0, either of the inputs needs to be0. Therefore, testing for the output being stuck at0 is equivalent to testing for either of the inputs being stuck at0.

In other words, input 1 being stuck at0, input 1 being stuck at0 and the output being stuck at0 are all equivalent faults.

Without faults, for the output to be 1, both inputs need to be 1. Therefore, if both the inputs being stuck at 1 have been tested there is no need to test for the outputs being stuck at 1.

In other words, the output being stuck at 1 is a dominated fault and the input 1 being stuck at 1 and input 1 being stuck at 1 are the dominant faults.

OR Gate

Without faults, for the output to be0, both inputs need to be0. Therefore, if both the inputs being stuck at0 have been tested there is no need to test for the outputs being stuck at0.

In other words, the output being stuck at0 is a dominated fault and the input 1 being stuck at0 and input 1 being stuck at0 are the dominant faults.

Without faults, for the output to be 1, either of the inputs needs to be 1. Therefore, testing for the output being stuck at 1 is equivalent to testing for either of the inputs being stuck at 1.

In other words, input 1 being stuck at 1, input 1 being stuck at 1 and the output being stuck at 1 are all equivalent faults.

NOR Gate

Without faults, for the output to be0, either of the inputs needs to be 1. Therefore, testing for the output being stuck at 1 is equivalent to testing for either of the inputs being stuck at 1.

In other words, input 1 being stuck at 1, input 1 being stuck at 1 and the output being stuck at0 are all equivalent faults.

Without faults, for the output to be 1, both inputs need to be0. Therefore, if both the inputs being stuck at0 have been tested there is no need to test for the outputs being stuck at0.

In other words, the output being stuck at 1 is a dominated fault and the input 1 being stuck at0 and input 1 being stuck at0 are the dominant faults.

This could also just have been calculated by looking at the or gate and switching the dominated output with the equivalent one as a NOR gate is just an inverted or gate.

3.1.2. List all non-equivalent and non-dominated faults for the circuit of Figure 1 (i.e. starting from the full list above, eliminate all equivalent and dominated faults). Show your work by identifying the reason for the eliminations (e.g., X s-a-0 is equivalent to Y s-a-1).

All Faults (28 in total)

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		

Eliminating Faults by Gate

$G = A \cdot B$

Gate	Input 1	Input 2	Output	Dominant Faults	Dominated Faults	Equivalent Faults
AND	A	B	G	Node A s-a-1	Node G s-a-1	Node A s-a-0
				Node B s-a-1		Node B s-a-0
						Node G s-a-0

$H = C \cdot D$

Gate	Input 1	Input 2	Output	Dominant Faults	Dominated Faults	Equivalent Faults
AND	C	D	H	Node C s-a-1	Node H s-a-1	Node C s-a-0
				Node D s-a-1		Node D s-a-0
						Node H s-a-0

$I = H + E$

Gate	Input 1	Input 2	Output	Dominant Faults	Dominated Faults	Equivalent Faults
OR	H	E	I	Node H s-a-0	Node I s-a-0	Node H s-a-1
				Node E s-a-0		Node E s-a-1
						Node I s-a-1

$K = I_1 + F$

Gate	Input 1	Input 2	Output	Dominant Faults	Dominated Faults	Equivalent Faults
OR	I_1	F	K	Node I_1 s-a-0	Node K s-a-0	Node I_1 s-a-1
				Node F s-a-0		Node F s-a-1
						Node K s-a-1

$L = J + K$

Gate	Input 1	Input 2	Output	Dominant Faults	Dominated Faults	Equivalent Faults
OR	J	K	L	Node J s-a-0	Node L s-a-0	Node J s-a-1
				Node K s-a-0		Node K s-a-1
						Node L s-a-1

$J = \overline{G + I_0}$

Gate	Input 1	Input 2	Output	Dominant Faults	Dominated Faults	Equivalent Faults
NOR	G	I_0	J	Node G s-a-0	Node J s-a-1	Node G s-a-1
				Node I_0 s-a-0		Node I_0 s-a-1
						Node J s-a-0

Sets of equivalent faults

Equivalent fault sets		
Node A s-a-0	Node B s-a-0	Node G s-a-0
Node C s-a-0	Node D s-a-0	Node H s-a-0
Node H s-a-1	Node E s-a-1	Node I s-a-1

Node I_1 s-a-1	Node F s-a-1	Node K s-a-1
Node J s-a-1	Node K s-a-1	Node L s-a-1
Node G s-a-1	Node I_0 s-a-1	Node J s-a-0

Merged sets of equivalent faults

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

Dominant and dominated faults

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

All sets of equivalent faults including faults which aren’t equivalent to anything (16 in total)

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

All Sets of equivalent faults including faults which aren’t equivalent to anything with dominated faults removed (10 in total)

All equivalent fault sets				
Node A s-a-0	Node B s-a-0	Node G s-a-0		
Node C s-a-0	Node D s-a-0	Node H s-a-0		
Node A s-a-1				
Node B s-a-1				
Node C s-a-1				
Node D s-a-1				

Node E s-a-0				
Node F s-a-0				
Node I_0 s-a-0				
Node I_1 s-a-0				

We will go with Node A s-a-0 and Node C s-a-0 from the sets o equivalent faults as those are closer to the input and therefore easier to calculate.

All non-equivalent non-dominated faults

Node A s-a-0	Node A s-a-1	Node B s-a-1	Node C s-a-0	Node C s-a-1
Node D s-a-1	Node E s-a-0	Node F s-a-0	Node I_0 s-a-0	Node I_1 s-a-0

3.1.3: List the test patterns (A-F inputs and expected output L, in this order) required to detect all detectable faults in the circuit and how they were derived using the D algorithm. Indicate any undetectable faults separately and show how you determined that they cannot be detected. At this stage, "don't care" input values should be labelled as 'X'.

By Fault

A s-a-0

A s-a-0	**A**	**B**	**C**	**D**	**E**	**F**	G	H	I	I_0	I_1	J	K	**L**
Step 1	D													
Step 2	D	1					D							
Step 3	D	1					D		0	0	0	D'		
Step 4	D	1					D		0	0	0	D'	0	
Step 5	D	1				0	D		0	0	0	D'	0	
Step 6	D	1				0	D		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	X/0	0	0	D	0	0	0	0	D'	0	D'

Test Pattern:

Test Pattern	**A**	**B**	**C**	**D**	**E**	**F**	**Correct L**
Test Pattern 1	1	1	0	X	0	0	0
Test Pattern 2	1	1	X	0	0	0	0

A s-a-1

A s-a-1	**A**	**B**	**C**	**D**	**E**	**F**	G	H	I	I_0	I_1	J	K	**L**
Step 1	D'													
Step 2	D'	1												
Step 3	D'	1					D'							
Step 4	D'	1					D'		0	0	0	D		
Step 5	D'	1					D'		0	0	0	D	0	D
Step 6	D'	1				0	D'		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	X/0	0	0	D'	0	0	0	0	D	0	D

Test Pattern:

Test Pattern	**A**	**B**	**C**	**D**	**E**	**F**	**Correct L**
Test Pattern 1	0	1	0	X	0	0	1
Test Pattern 2	0	1	X	0	0	0	1

B s-a-1

B s-a-1	**A**	**B**	**C**	**D**	**E**	**F**	G	H	I	I_0	I_1	J	K	**L**
Step 1		D'												
Step 2	1	D'												
Step 3	1	D'					D'							
Step 4	1	D'					D'		0	0	0	D		
Step 5	1	D'					D'		0	0	0	D	0	D
Step 6	1	D'				0	D'		0	0	0	D	0	D
Step 7	1	D'			0	0	D'	0	0	0	0	D	0	D
Step 8	1	D'	0/X	X/0	0	0	D'	0	0	0	0	D	0	D

Test Pattern:

Test Pattern	**A**	**B**	**C**	**D**	**E**	**F**	**Correct L**
Test Pattern 1	1	0	0	X	0	0	1
Test Pattern 2	1	0	X	0	0	0	1

C s-a-0

C s-a-0	**A**	**B**	**C**	**D**	**E**	**F**	G	H	I	I_0	I_1	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	
Step 5			D	1	0	0		D	D	D	D	0	D	D
Step 6			D	1	0	0	1	D	D	D	D	0	D	D
Step 7	1	1	D	1	0	0	1	D	D	D	D	0	D	D

Test Pattern:

Test Pattern	**A**	**B**	**C**	**D**	**E**	**F**	**Correct L**
Test Pattern 1	1	1	1	1	0	0	1

C s-a-1

C s-a-1	**A**	**B**	**C**	**D**	**E**	**F**	G	H	I	I_0	I_1	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		1	D'	D'	D'	D'	D		
Step 5			D'	1	0		1	D'	D'	D'	D'	D	1	D
Step 6			D'	1	0	1	1	D'	D'	D'	D'	D	1	D
Step 7	1	1	D'	1	0	1	1	D'	D'	D'	D'	D	1	D

Test Pattern:

Test Pattern	**A**	**B**	**C**	**D**	**E**	**F**	**Correct L**
Test Pattern 1	1	1	0	1	0	1	1

D s-a-1

D s-a-1	**A**	**B**	**C**	**D**	**E**	**F**	G	H	I	I_0	I_1	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		1	D'	D'	D'	D'	D		

Step 5			1	D'	0		1	D'	D'	D'	D'	D	1	D
Step 6			1	D'	0	1	1	D'	D'	D'	D'	D	1	D
Step 7	1	1	1	D'	0	1	1	D'	D'	D'	D'	D	1	D

Test Pattern:

Test Pattern	**A**	**B**	**C**	**D**	**E**	**F**	**Correct L**
Test Pattern 1	1	1	1	0	0	1	1

E s-a-0

E s-a-0	**A**	**B**	**C**	**D**	**E**	**F**	G	H	I	I_0	I_1	J	K	**L**
Step 1					D									
Step 2					D			0	D	D	D			
Step 3					D		1	0	D	D	D	D'		
Step 4					D		1	0	D	D	D	D'	0	D'

This is impossible because K cannot be 0 when I_1 is D. So we try the other option

E s-a-0	**A**	**B**	**C**	**D**	**E**	**F**	G	H	I	I_0	I_1	J	K	**L**
Step 1					D									
Step 2					D			0	D	D	D			
Step 3					D	0		0	D	D	D		D	
Step 4					D	0		0	D	D	D	0	D	D
Step 5					D	0	1	0	D	D	D	0	D	D
Step 6	1	1			D	0	1	0	D	D	D	0	D	D
Step 7	1	1	0/X	X/0	D	0	1	0	D	D	D	0	D	D

Test Pattern:

Test Pattern	**A**	**B**	**C**	**D**	**E**	**F**	**Correct L**
Test Pattern 1	1	1	0	X	1	0	1
Test Pattern 2	1	1	X	0	1	0	1

F s-a-0

F s-a-0	**A**	**B**	**C**	**D**	**E**	**F**	G	H	I	I_0	I_1	J	K	**L**
Step 1						D								
Step 2						D			0	0	0		D	
Step 3						D			0	0	0	0	D	D
Step 4						D	1		0	0	0	0	D	D
Step 5	1	1				D	1		0	0	0	0	D	D
Step 6	1	1			1	D	1	X	0	0	0	0	D	D
Step 7	1	1	X	X	1	D	1	X	0	0	0	0	D	D

Test Pattern:

Test Pattern	**A**	**B**	**C**	**D**	**E**	**F**	**Correct L**
Test Pattern 1	1	1	X	X	1	1	1

I_0 s-a-0

I_0 s-a-0	**A**	**B**	**C**	**D**	**E**	**F**	G	H	I	I_0	I_1	J	K	**L**
Step 1									1	D	1			
Step 2							0		1	D	1	D'		
Step 3							0		1	D	1	D'	0	D'

This is impossible because K cannot be 0 when I_1 is 1. Additionally, there are no other approaches to try so I_0 s-a-0 is untestable.

I1 s-a-0

I_1 s-a-0	**A**	**B**	**C**	**D**	**E**	**F**	G	H	I	I_0	I_1	J	K	**L**
Step 1									1	1	D			
Step 2							X		1	1	D	0		
Step 3						0	X		1	1	D	0	D	
Step 4						0	X		1	1	D	0	D	D
Step 5					1	0	X	X	1	1	D	0	D	D
Step 6			X/1	X/1	1/X	0	X	X	1	1	D	0	D	D
Step 7	X	X	X/1	X/1	1/X	0	X	X	1	1	D	0	D	D

Test Pattern:

Test Pattern	**A**	**B**	**C**	**D**	**E**	**F**	**Correct L**
Test Pattern 1	X	X	X	X	1	X	1
Test Pattern 2	X	X	1	1	X	0	1

All Test Patterns

Fault	**Test Pattern**	**A**	**B**	**C**	**D**	**E**	**F**	**Correct L**
Node A s-a-0	Test Pattern 1	1	1	0	X	0	0	0
	Test Pattern 2	1	1	X	0	0	0	0
Node A s-a-1	Test Pattern 1	0	1	0	X	0	0	1
	Test Pattern 2	0	1	X	0	0	0	1
Node B s-a-1	Test Pattern 1	1	0	0	X	0	0	1
	Test Pattern 2	1	0	X	0	0	0	1
Node C s-a-0	Test Pattern 1	1	1	1	1	0	0	1
Node C s-a-1	Test Pattern 1	1	1	0	1	0	1	1
Node D s-a-1	Test Pattern 1	1	1	1	0	0	1	1
Node E s-a-0	Test Pattern 1	1	1	0	X	1	0	1
	Test Pattern 2	1	1	X	0	1	0	1
Node F s-a-0	Test Pattern 1	1	1	X	X	1	1	1
Node I_0 s-a-0	No valid test patterns							
Node I_1 s-a-0	Test Pattern 1	X	X	X	X	1	X	1
	Test Pattern 2	X	X	1	1	X	0	1

3.1.4: Indicate which test patterns in the list above can be merged (if any).

Test patterns to be merged

Fault 1	**Test Pattern 1**	**Fault 2**	**Test Pattern 2**
Node I_1 s-a-0	Test Pattern 1	Node E s-a-0	Test Pattern 1

3.1.5: List the reduced test patterns required to exhaustively test the circuit. At this stage, all “don’t care” values should be resolved to 0s (technically, 1s would work too, but 0s are slightly better) and D/D’ to 1s and0s. Hint: at this stage, you should have reduced the set to 7 or 8 patterns.

Test Patterns After Merging

A	**B**	**C**	**D**	**E**	**F**	**Correct L**
1	1	0	0	0	0	0
0	1	0	0	0	0	1
1	0	0	0	0	0	1
1	1	1	1	0	0	1

1	1	0	1	0	1	1
1	1	1	0	0	1	1
1	1	0	0	1	0	1
1	1	0	0	1	1	1

3.1.6: For each test pattern, perform fault simulation to determine all faults that the pattern is able to detect. In other words, determine which of the faults on the original list (step2) are detected by each of the vectors you found at 3.1.5. Use a table and list the faults for each vector alphabetically.

[todo]

Lab 4 - B

3.2.1: Print out the memory content file.

```
-- (c) Copyright 1995-2022 Xilinx, Inc. All rights reserved.
--
-- This file contains confidential and proprietary information
-- of Xilinx, Inc. and is protected under U.S. and
-- international copyright and other intellectual property
-- laws.
--
-- DISCLAIMER
-- This disclaimer is not a license and does not grant any
-- rights to the materials distributed herewith. Except as
-- otherwise provided in a valid license issued to you by
-- Xilinx, and to the maximum extent permitted by applicable
-- law: (1) THESE MATERIALS ARE MADE AVAILABLE "AS IS" AND
-- WITH ALL FAULTS, AND XILINX HEREBY DISCLAIMS ALL WARRANTIES
-- AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING
-- BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-
-- INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and
-- (2) Xilinx shall not be liable (whether in contract or tort,
-- including negligence, or under any other theory of
-- liability) for any loss or damage of any kind or nature
-- related to, arising under or in connection with these
-- materials, including for any direct, or any indirect,
-- special, incidental, or consequential loss or damage
-- (including loss of data, profits, goodwill, or any type of
-- loss or damage suffered as a result of any action brought
-- by a third party) even if such damage or loss was
-- reasonably foreseeable or Xilinx had been advised of the
-- possibility of the same.
--
-- CRITICAL APPLICATIONS
-- Xilinx products are not designed or intended to be fail-
-- safe, or for use in any application requiring fail-safe
-- performance, such as life-support or safety devices or
-- systems, Class III medical devices, nuclear facilities,
-- applications related to the deployment of airbags, or any
-- other applications that could lead to death, personal
-- injury, or severe property or environmental damage
-- (individually and collectively, "Critical
-- Applications"). Customer assumes the sole risk and
-- liability of any use of Xilinx products in Critical
-- Applications, subject only to applicable laws and
-- regulations governing limitations on product liability.
--
-- THIS COPYRIGHT NOTICE AND DISCLAIMER MUST BE RETAINED AS
-- PART OF THIS FILE AT ALL TIMES.
--
-- DO NOT MODIFY THIS FILE.

-- IP VLNV: xilinx.com:ip:dist_mem_gen:8.0
-- IP Revision: 12

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
```

```vhdl
LIBRARY dist_mem_gen_v8_0_12;
USE dist_mem_gen_v8_0_12.dist_mem_gen_v8_0_12;

ENTITY Test_Pattern_RAM IS
  PORT (
    a : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
    d : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
    clk : IN STD_LOGIC;
    we : IN STD_LOGIC;
    spo : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
  );
END Test_Pattern_RAM;

ARCHITECTURE Test_Pattern_RAM_arch OF Test_Pattern_RAM IS
  ATTRIBUTE DowngradeIPIdentifiedWarnings : STRING;
  ATTRIBUTE DowngradeIPIdentifiedWarnings OF Test_Pattern_RAM_arch: ARCHITECTURE IS "yes";
  COMPONENT dist_mem_gen_v8_0_12 IS
    GENERIC (
      C_FAMILY : STRING;
      C_ADDR_WIDTH : INTEGER;
      C_DEFAULT_DATA : STRING;
      C_DEPTH : INTEGER;
      C_HAS_CLK : INTEGER;
      C_HAS_D : INTEGER;
      C_HAS_DPO : INTEGER;
      C_HAS_DPRA : INTEGER;
      C_HAS_I_CE : INTEGER;
      C_HAS_QDPO : INTEGER;
      C_HAS_QDPO_CE : INTEGER;
      C_HAS_QDPO_CLK : INTEGER;
      C_HAS_QDPO_RST : INTEGER;
      C_HAS_QDPO_SRST : INTEGER;
      C_HAS_QSPO : INTEGER;
      C_HAS_QSPO_CE : INTEGER;
      C_HAS_QSPO_RST : INTEGER;
      C_HAS_QSPO_SRST : INTEGER;
      C_HAS_SPO : INTEGER;
      C_HAS_WE : INTEGER;
      C_MEM_INIT_FILE : STRING;
      C_ELABORATION_DIR : STRING;
      C_MEM_TYPE : INTEGER;
      C_PIPELINE_STAGES : INTEGER;
      C_QCE_JOINED : INTEGER;
      C_QUALIFY_WE : INTEGER;
      C_READ_MIF : INTEGER;
      C_REG_A_D_INPUTS : INTEGER;
      C_REG_DPRA_INPUT : INTEGER;
      C_SYNC_ENABLE : INTEGER;
      C_WIDTH : INTEGER;
      C_PARSER_TYPE : INTEGER
    );
    PORT (
      a : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
      d : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
      dpra : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
      clk : IN STD_LOGIC;
      we : IN STD_LOGIC;
      i_ce : IN STD_LOGIC;
      qspo_ce : IN STD_LOGIC;
      qdpo_ce : IN STD_LOGIC;
      qdpo_clk : IN STD_LOGIC;
      qspo_rst : IN STD_LOGIC;
      qdpo_rst : IN STD_LOGIC;
      qspo_srst : IN STD_LOGIC;
      qdpo_srst : IN STD_LOGIC;
      spo : OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
      dpo : OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
      qspo : OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
      qdpo : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
    );
  END COMPONENT dist_mem_gen_v8_0_12;
  ATTRIBUTE X_CORE_INFO : STRING;
  ATTRIBUTE X_CORE_INFO OF Test_Pattern_RAM_arch: ARCHITECTURE IS
"dist_mem_gen_v8_0_12,Vivado 2017.4";
  ATTRIBUTE CHECK_LICENSE_TYPE : STRING;
  ATTRIBUTE CHECK_LICENSE_TYPE OF Test_Pattern_RAM_arch : ARCHITECTURE IS
"Test_Pattern_RAM,dist_mem_gen_v8_0_12,{}";
  ATTRIBUTE CORE_GENERATION_INFO : STRING;
  ATTRIBUTE CORE_GENERATION_INFO OF Test_Pattern_RAM_arch: ARCHITECTURE IS
"Test_Pattern_RAM,dist_mem_gen_v8_0_12,{x_ipProduct=Vivado
2017.4,x_ipVendor=xilinx.com,x_ipLibrary=ip,x_ipName=dist_mem_gen,x_ipVersion=8.0,x_ipCoreR
evision=12,x_ipLanguage=VHDL,x_ipSimLanguage=MIXED,C_FAMILY=zynq,C_ADDR_WIDTH=4,C_DEFAULT_D
ATA=0,C_DEPTH=16,C_HAS_CLK=1,C_HAS_D=1,C_HAS_DPO=0,C_HAS_DPRA=0,C_HAS_I_CE=0,C_HAS_QDPO=0,C
_HAS_QDPO_CE=0,C_HAS_QDPO_CLK=0,C_HAS_QDPO_RST=0,C_HAS_QDPO_SRST=0,C_HAS_QSPO=0,C_HAS_QSPO_
CE=0,C_HAS_QSPO_RST=0,C_HAS_QSPO_SRST=0,C_HAS_SPO=1,C_HAS_WE=1,C_MEM_INIT_FILE" &
"=Test_Pattern_RAM.mif,C_ELABORATION_DIR=./,C_MEM_TYPE=1,C_PIPELINE_STAGES=0,C_QCE_JOINED=0
,C_QUALIFY_WE=0,C_READ_MIF=1,C_REG_A_D_INPUTS=0,C_REG_DPRA_INPUT=0,C_SYNC_ENABLE=1,C_WIDTH=
16,C_PARSER_TYPE=1}";
BEGIN
  U0 : dist_mem_gen_v8_0_12
    GENERIC MAP (
      C_FAMILY => "zynq",
      C_ADDR_WIDTH => 4,
      C_DEFAULT_DATA => "0",
      C_DEPTH => 16,
      C_HAS_CLK => 1,
      C_HAS_D => 1,
      C_HAS_DPO => 0,
      C_HAS_DPRA => 0,
      C_HAS_I_CE => 0,
      C_HAS_QDPO => 0,
      C_HAS_QDPO_CE => 0,
      C_HAS_QDPO_CLK => 0,
      C_HAS_QDPO_RST => 0,
      C_HAS_QDPO_SRST => 0,
      C_HAS_QSPO => 0,
      C_HAS_QSPO_CE => 0,
      C_HAS_QSPO_RST => 0,
      C_HAS_QSPO_SRST => 0,
      C_HAS_SPO => 1,
      C_HAS_WE => 1,
      C_MEM_INIT_FILE => "Test_Pattern_RAM.mif",
      C_ELABORATION_DIR => "./",
      C_MEM_TYPE => 1,
      C_PIPELINE_STAGES => 0,
      C_QCE_JOINED => 0,
      C_QUALIFY_WE => 0,
      C_READ_MIF => 1,
      C_REG_A_D_INPUTS => 0,
      C_REG_DPRA_INPUT => 0,
      C_SYNC_ENABLE => 1,
      C_WIDTH => 16,
      C_PARSER_TYPE => 1
    )
    PORT MAP (
      a => a,
      d => d,
      dpra => STD_LOGIC_VECTOR(TO_UNSIGNED(0, 4)),
      clk => clk,
      we => we,
      i_ce => '1',
      qspo_ce => '1',
      qdpo_ce => '1',
      qdpo_clk => '0',
      qspo_rst => '0',
      qdpo_rst => '0',
      qspo_srst => '0',
      qdpo_srst => '0',
      spo => spo
    );
END Test_Pattern_RAM_arch;
```

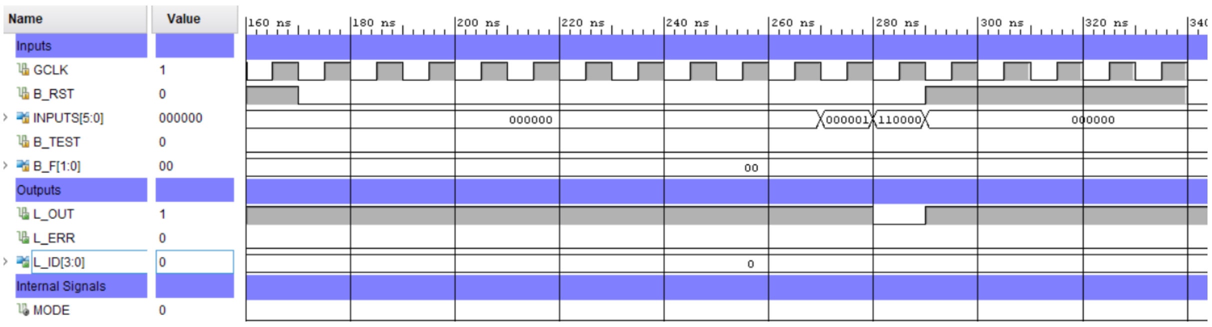
3.2.2: Print out the VHDL testbench.

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

entity top_level_tb is
end top_level_tb;
```

```vhdl
architecture Behavioral of top_level_tb is
    -- Length of clock period
    constant clk_period: time := 10ns;
    -- Length of pause at the start
    constant initial_wait_period: time := clk_period * 10;
    -- Length of button press (as the buttons are debounced)
    constant button_press_period: time := clk_period * 5;
    -- Pause after button press (to let the circuit reset)
    constant button_press_wait_period: time := clk_period * 10;
    -- How long it takes to run a full test set
    constant full_run_period: time := clk_period * 11;

    -- Internal reperesentations of the UUT inputs
    signal GCLK: STD_LOGIC;
    signal B_RST: STD_LOGIC;
    signal INPUTS: STD_LOGIC_VECTOR (5 downto 0);
    signal B_TEST: STD_LOGIC;

    -- Internal reperesentation of the UUT outputs
    signal B_F: STD_LOGIC_VECTOR (1 downto 0);
    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,       -- in: clock
        B_RST => B_RST,     -- in: reset
        INPUTS => INPUTS, -- in: uut input for normal node
        B_TEST => B_TEST, -- in: whether test mode is active
        B_F => B_F,         -- in: which fake faults to activate
        L_OUT => L_OUT,     -- out: uut output for both modes
        L_ERR => L_ERR,     -- out: whether there is an error for normal mode
        L_ID => L_ID        -- out: current test are we running for normal mode
    );

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

    -- Tests
    test: process
    begin
        -- Testing Strategy
        --  Since this has a built-in self-test, we will not need to manually
        --   test many things. We will not need to use asserts since the built-in
        --   self-test does that for us. There are two stages to the testing:
        --  - Firstly we need to test basic operation. For this we just choose
        --     two arbitrary sets of values and set them as inputs. Then we can
        --     look at the simulation to see that the output changes.
        --  - Then we need to run the full build-in self-test test set for each
        --     possible fault as well as for no faults. This can be done by
        --     setting B_TEST to true and just letting it run for 11 clock
        --     cycles.

        -- Sync with falling edge
        wait until falling_edge(GCLK);

        -- Initial wait
        wait for initial_wait_period;
        -- Initialise values
        B_RST <= '0';
        INPUTS <= "000000";
        B_TEST <= '0';
        B_F <= "00";
        wait for clk_period;

        -- Reset the system
        B_RST <= '1';
        wait for button_press_period;
        B_RST <= '0';
        wait for button_press_wait_period;

        -- Try out a pattern without faults
        INPUTS <= "000001";
        wait for clk_period;

        -- Try out a second pattern without faults
        INPUTS <= "110000";
        wait for clk_period;

        -- Reset input
        INPUTS <= "000000";

        -- Full runs with various faults
        for I in 0 to 3 loop
            B_F <= std_logic_vector(to_unsigned(I, B_F'length));

            -- Reset the system
            B_RST <= '1';
            wait for button_press_period;
            B_RST <= '0';
            wait for button_press_wait_period;

            -- Test full set without faults
            B_Test <= '1';

            -- Wait for full set to run
            wait for full_run_period;

            -- Turn test mode off
            B_Test <= '0';
        end loop;
        wait;
    end process;
end Behavioral;
```

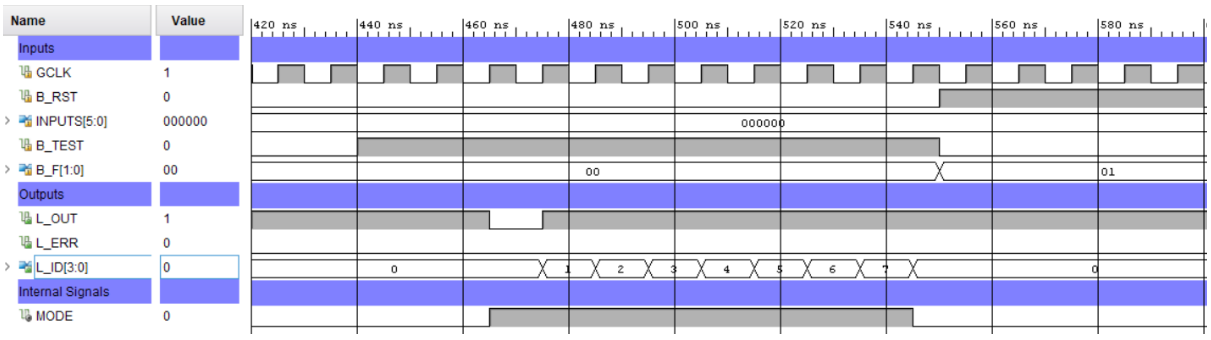
3.2.3: Print out screenshots of the behavioural simulation for the system. The screenshot(s) should show, in readable format, all inputs and outputs of the circuit as well as the internal signal MODE and the address and output data of all internal memories, for:

The operation of the fault-free circuit for at least two different combinations of inputs

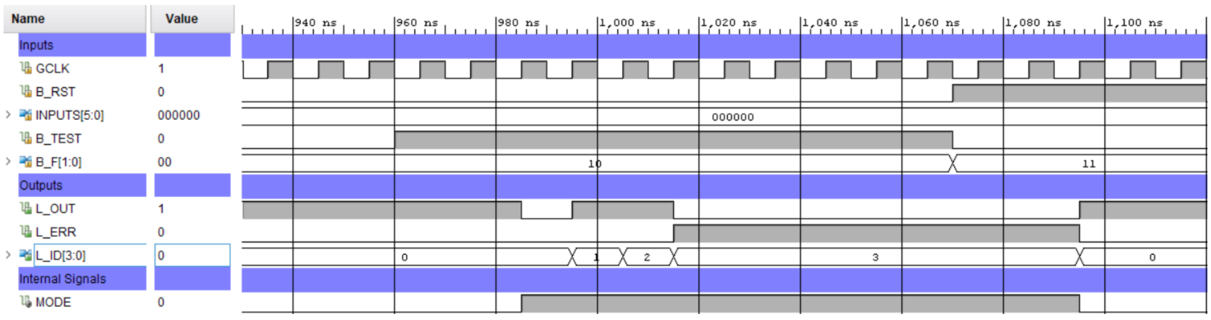


Name	Value
Inputs	
GCLK	1
B_RST	0
INPUTS[5:0]	000000
B_TEST	0
B_F[1:0]	00
Outputs	
L_OUT	1
L_ERR	0
L_ID[3:0]	0
Internal Signals	
MODE	0

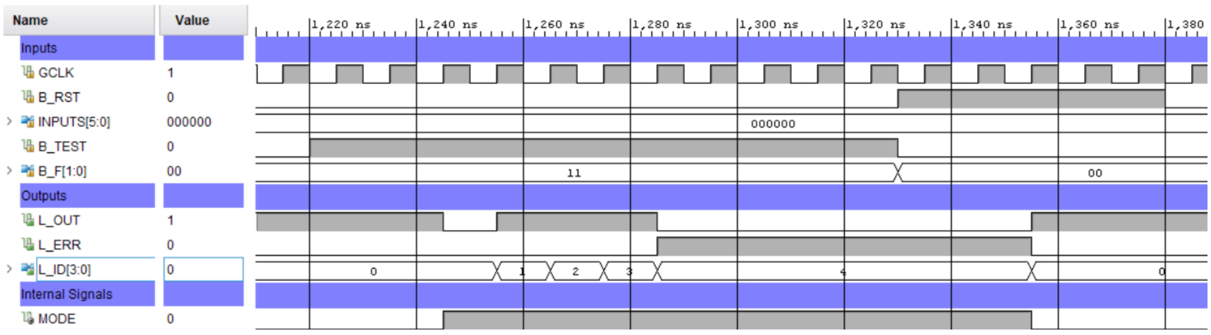
The complete test cycle when no fault is present



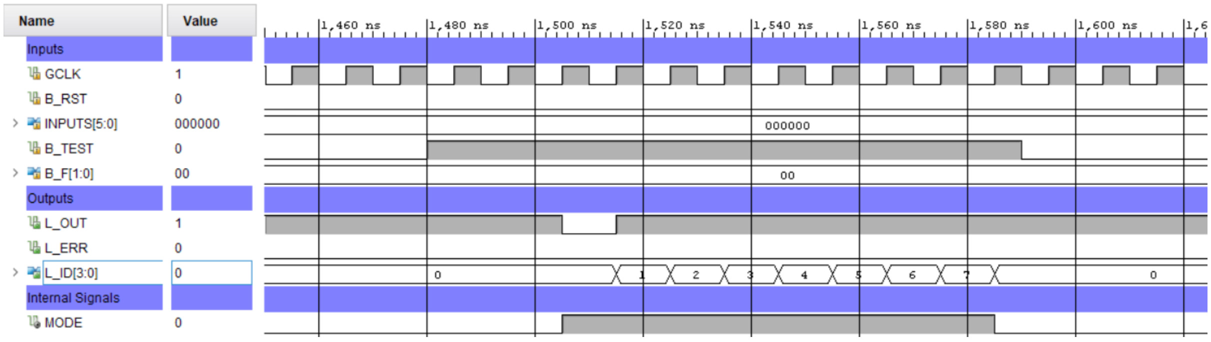
The complete test cycle when each of the three faults is present (Node E s-a-1)



The complete test cycle when each of the three faults is present (Node H s-a-0)



The complete test cycle when each of the three faults is present (Node F s-a-0)

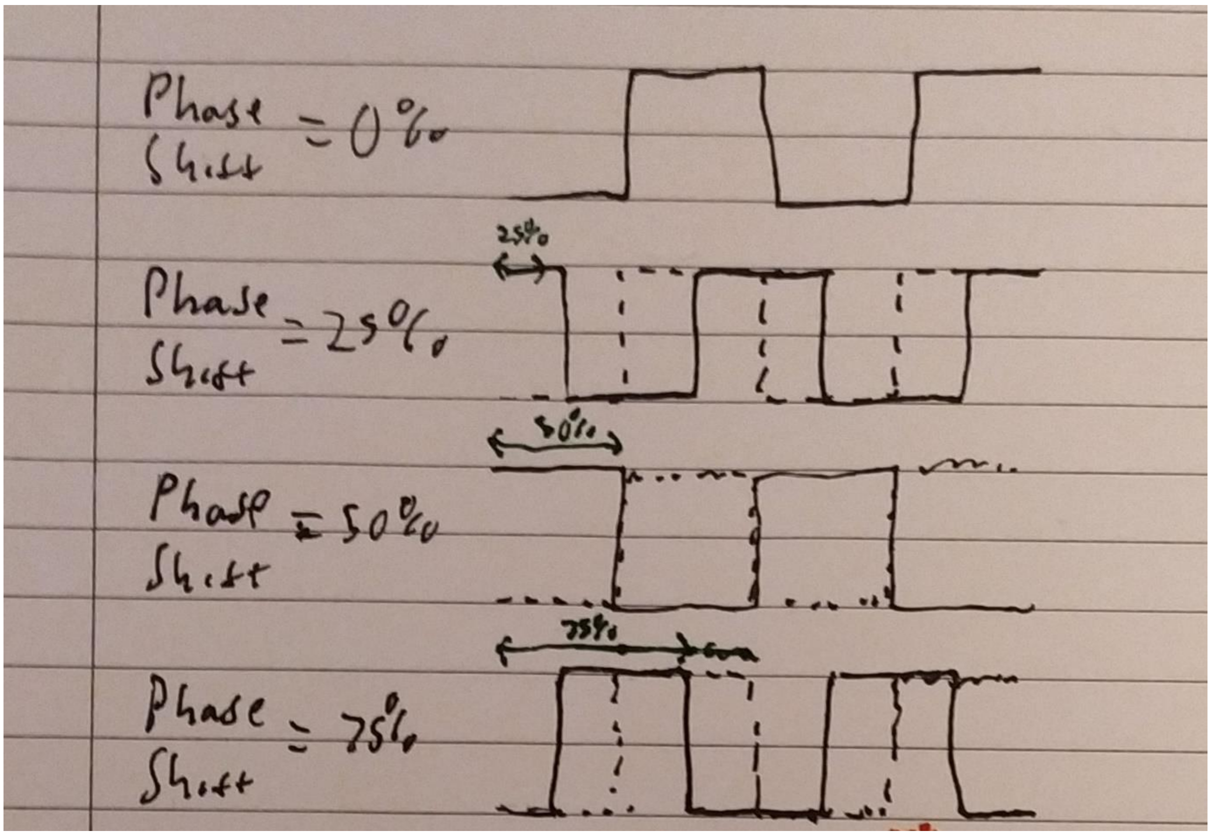


Lab 4 – C

3.3.1: Explain, using a few words and graphs, the concepts of phase shift and duty cycle in a clock signal.

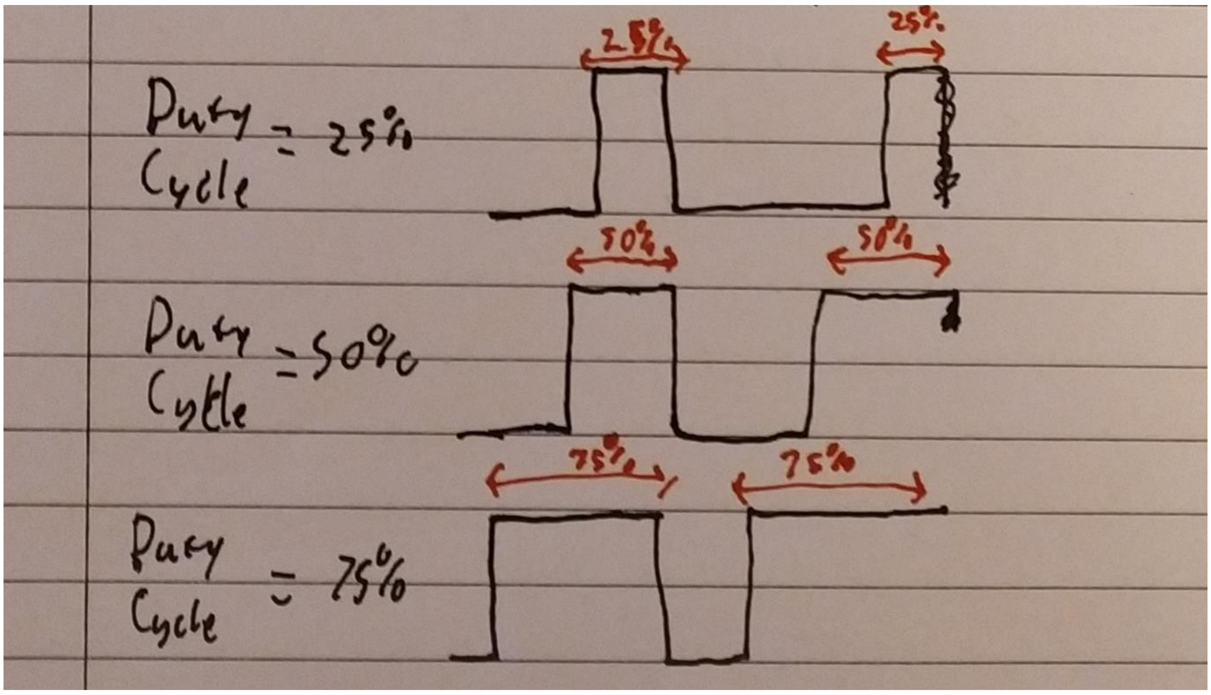
Phase Shift

Phase shift is the offset of a signal in the time domain compared to the default signal. A signal with a phase shift of 0% will rise at the same time as the default signal whereas a signal with a phase shift of 50% will rise half a phase after the default signal.



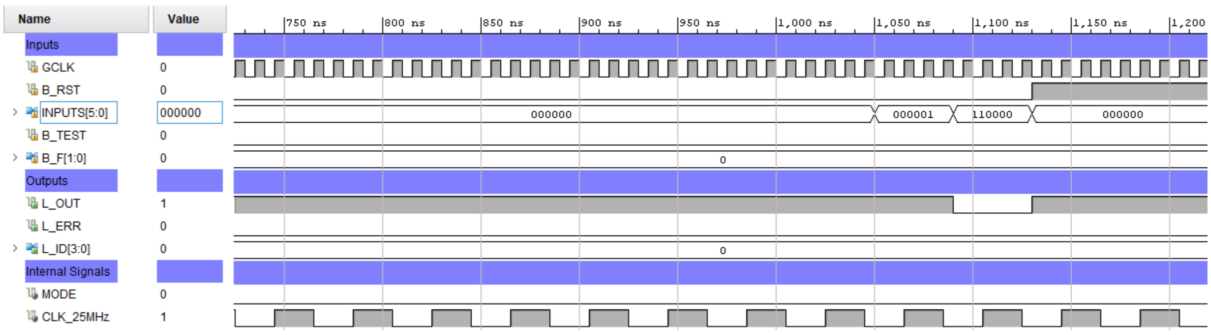
Duty Cycle

The duty cycle of a system is the time in which a signal is active (i.e. has a value of 1) as a proportion of a period. A signal with 25% duty cycle will be active one quarter of the time and a signal with 50% duty cycle will be active half the time.



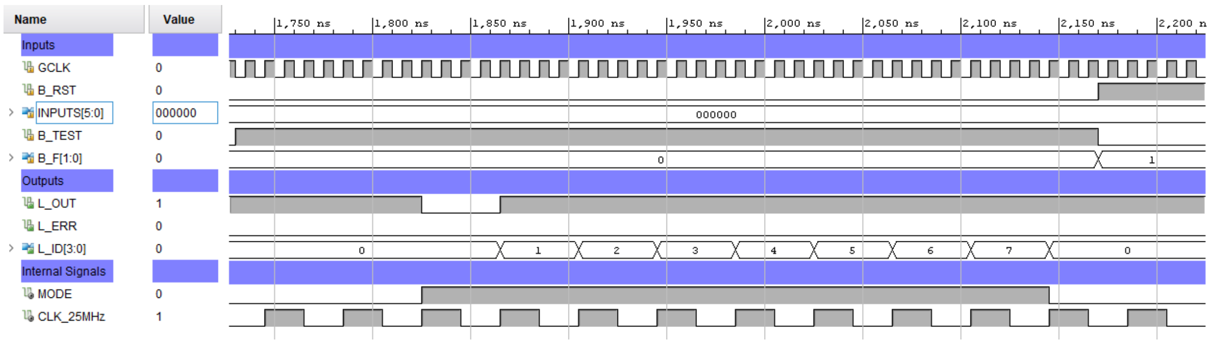
3.3.2: Print out screenshots of the behavioural simulation for the system. The screenshot(s)should show, in readable format, all the events and signals of the previous task (3.2.5) but also clearly display the new internal clock and how it affects the circuit.

The operation of the fault-free circuit for at least two different combinations of inputs



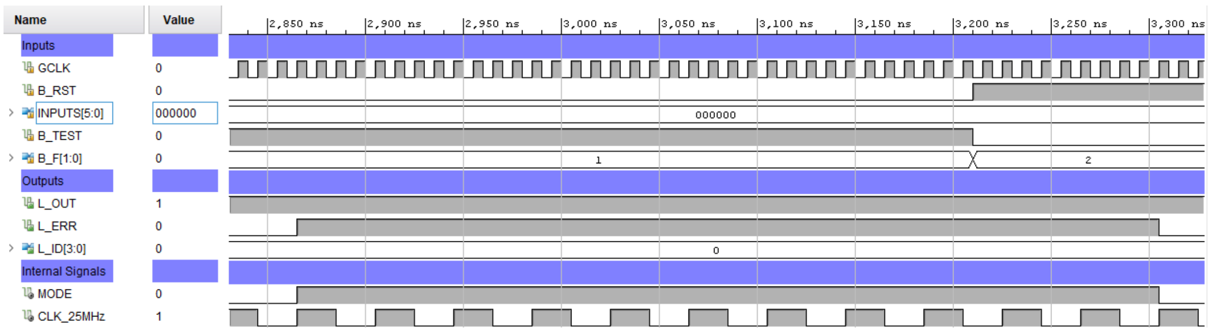
Name	Value
Inputs	
GCLK	0
B_RST	0
INPUTS[5:0]	000000
B_TEST	0
B_F[1:0]	0
Outputs	
L_OUT	1
L_ERR	0
L_ID[3:0]	0
Internal Signals	
MODE	0
CLK_25MHz	1

The complete test cycle when no fault is present

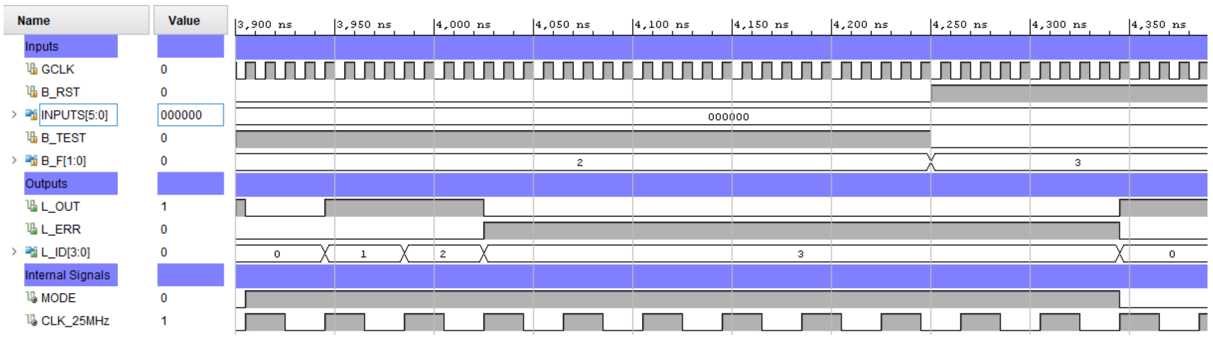


Name	Value
Inputs	
GCLK	0
B_RST	0
INPUTS[5:0]	000000
B_TEST	0
B_F[1:0]	0
Outputs	
L_OUT	1
L_ERR	0
L_ID[3:0]	0
Internal Signals	
MODE	0
CLK_25MHz	1

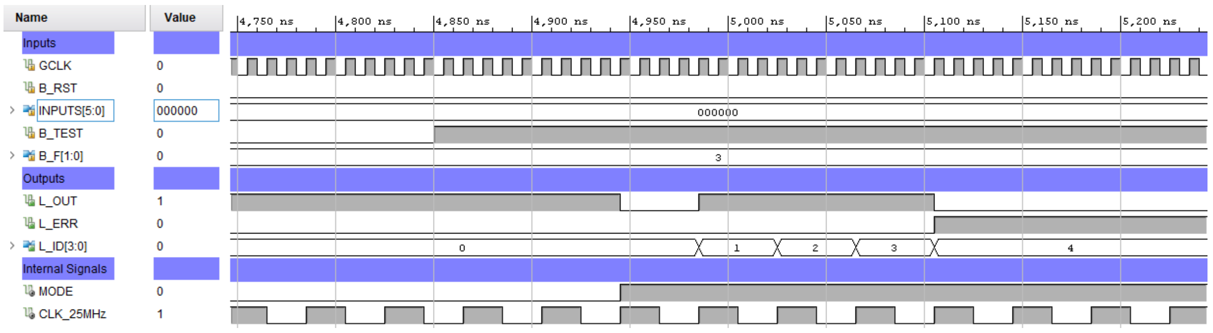
The complete test cycle when each of the three faults is present (Node E s-a-1)



The complete test cycle when each of the three faults is present (Node H s-a-0)

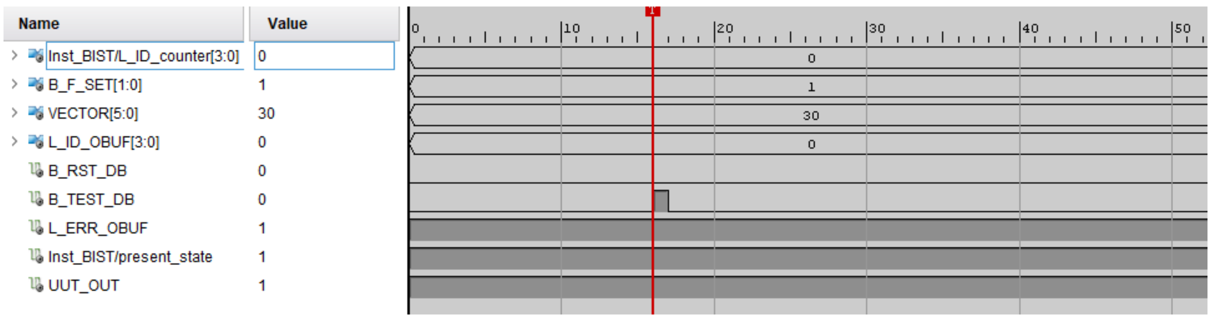


The complete test cycle when each of the three faults is present (Node F s-a-0)

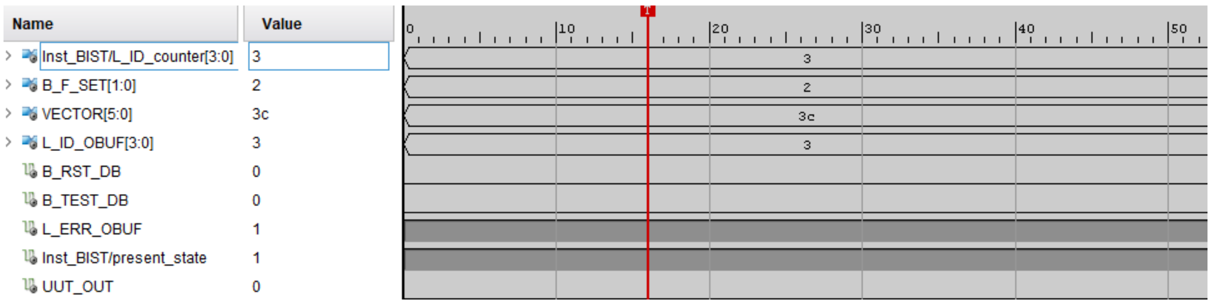


3.3.3: Print out the ILA window showing the values of the internal signals when the analyser is triggered for the fault-free cycle and for each of the three faults. Make sure that the signal names (including vectors) are the same as the original VHDL file and that the output matches your simulations.

E s-a-1

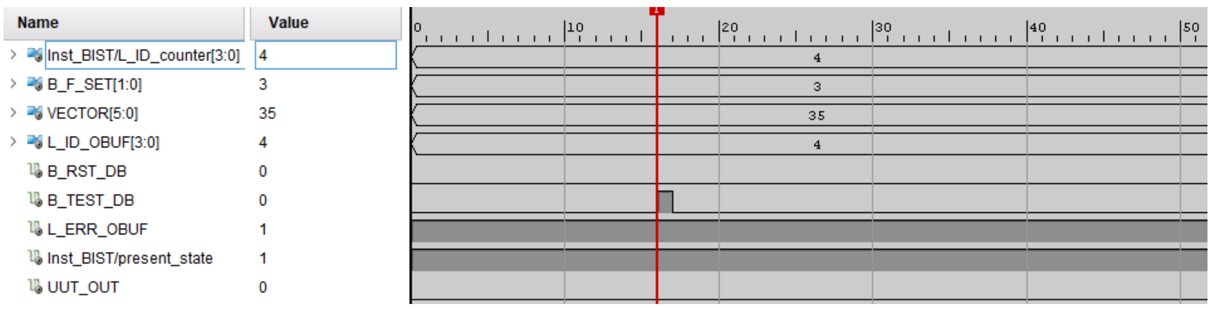


H s-a-0



Name	Value
Inst_BIST/L_ID_counter[3:0]	3
B_F_SET[1:0]	2
VECTOR[5:0]	3c
L_ID_OBUF[3:0]	3
B_RST_DB	0
B_TEST_DB	0
L_ERR_OBUF	1
Inst_BIST/present_state	1
UUT_OUT	0

F s-a-0



Name	Value
Inst_BIST/L_ID_counter[3:0]	4
B_F_SET[1:0]	3
VECTOR[5:0]	35
L_ID_OBUF[3:0]	4
B_RST_DB	0
B_TEST_DB	0
L_ERR_OBUF	1
Inst_BIST/present_state	1
UUT_OUT	0

3.3.4: When the ILA was set up, the tools automatically selected the 25MHz clock for the analyser. In practice, the ILA is working at the same frequency as the circuit it is observing. This appears to be in direct violation of Nyquist's theorem. Can you think of a reason why this works?

The actual parts of the circuit that are being measured are not changing every clock cycle. It is only the clock and some of the internal signals that are changing that fast.