Introduction to VHDL

Introduction to HDL

- In electronics, HDL is a class of computer language for the formal description of the electronic circuit.
- It is used to model the circuit at different levels.
- (Top level to gate level).
- A synthesizable HDL code results to an equivalent hardware.
- HDL can be used for verification of the circuits.

Classification of HDL

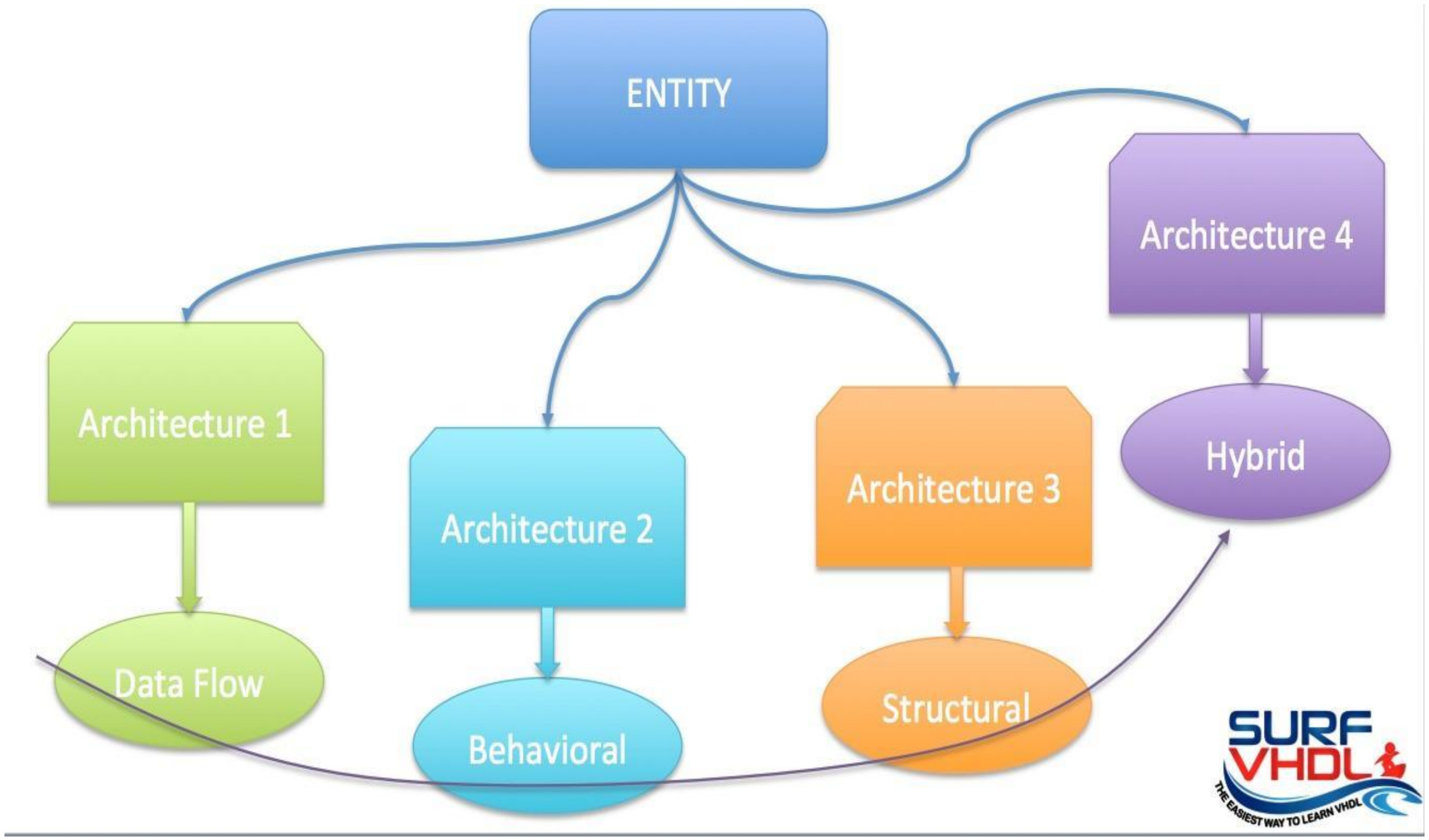
VHDL (VHSIC-HDL)	Verilog
It is strongly typed	It is weakly typed
High verbosity	Low verbosity
Case insensitive	Case sensitive
Non C like	More C like
Upto gate level implementation only	Transistor level implementation is possible

Features of VHDL

- Supports concurrent and sequential execution of statements.
- It possesses IEEE standards.
- Like C, VHDL also supports different data types, conditional, relational, logical, arithmetic operators.
- Supports 3 different modelling styles

  1. Dataflow
  2. Behavioural
  3. Structural

Different Modelling Style



https://surf-vhdl.com/wp/wp-content/uploads/2015/07/CodingStyle1.jpg

Operators

LOGICAL :

- AND
- OR
- NOR
- NAND
- NOT
- XOR
- XNOR

ARITHMETIC:

use ieee.NUMERIC_STD.all;

This header file must be included for all arithmetic computations.

- + : addition
- - : subtraction
- * : multiplication
- / : division
- ABS : absolute value
- MOD : modulus
- REM : remainder
- ** : exponent

Comparison Operators:

- = equal to
- /= not equal to
- < less than
- > greater than
- <= less than or equal to
- >= greater than or equal to

Branching Statements:

Always used in Behavioural modelling style

```
if condition_1 then
   sequential statements
elsif condition2 then
   sequential statements
else
   sequential statements
end if;
```

Branching Statements:

Always used in Behavioural modelling style

```
case expression is
   when choice =>
       sequential statements
   when choice =>
        sequential statements
end case;
```

Looping Statements

```
optional_label: for parameter in range loop
      sequential statements
end loop label;
```

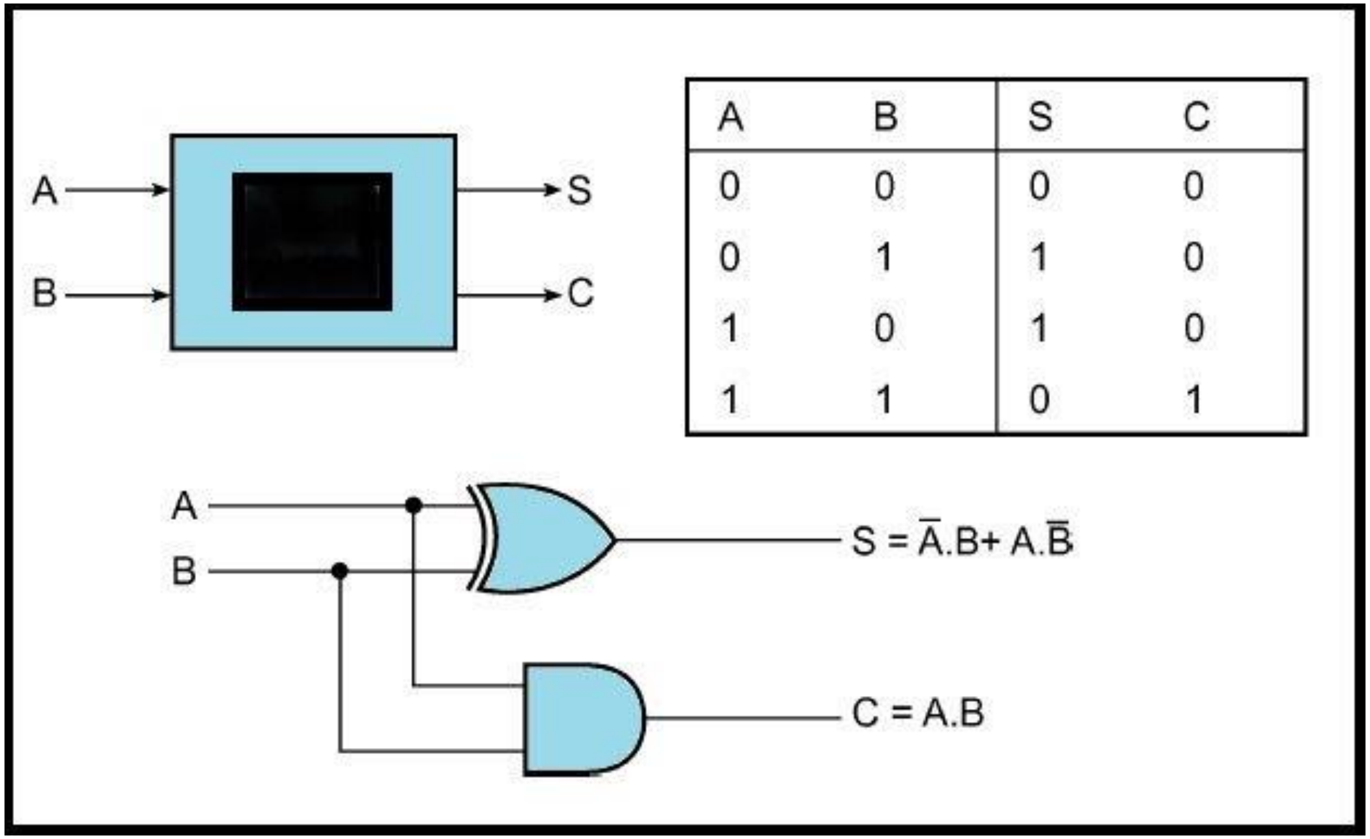
Ex:

```
for I in 1 to 10 loop
      if (REPEAT = '1') then
            I := I-1;  -- Illegal
      end if;
end loop;
```

Looping Statements

```
while condition loop
   sequential statements
end loop;
```

Circuit 1



A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

VHDL Dataflow Modelling Style

```
library ieee;
use ieee.std_logic_1164.all;

entity circuit_1 is
  port (A, B: in std_logic;
    S, C: out std_logic);
  end circuit_1;

architecture dataflow of
circuit_1 is
  Begin
  S <= A xor B;
  C <= a and b;
end dataflow;
```

https://buzztech.in/vhdl-modelling-styles-behavioral-dataflow-structural/

VHDL Behavioral Modelling Style

```
library ieee;
use ieee.std_logic_1164.all;

entity circuit_1 is
  port (A, B: in std_logic;
    S, C: out std_logic);
  end circuit_1;

architecture behavior of circuit_1 is
  Begin
    c1: process (A,B)
    Begin
      if A = '1' then
        S <= not B;
        C <= B;
      Else
        S <= B;
        C <= '0';
      end if;
    end process c1;
end behavior;
```

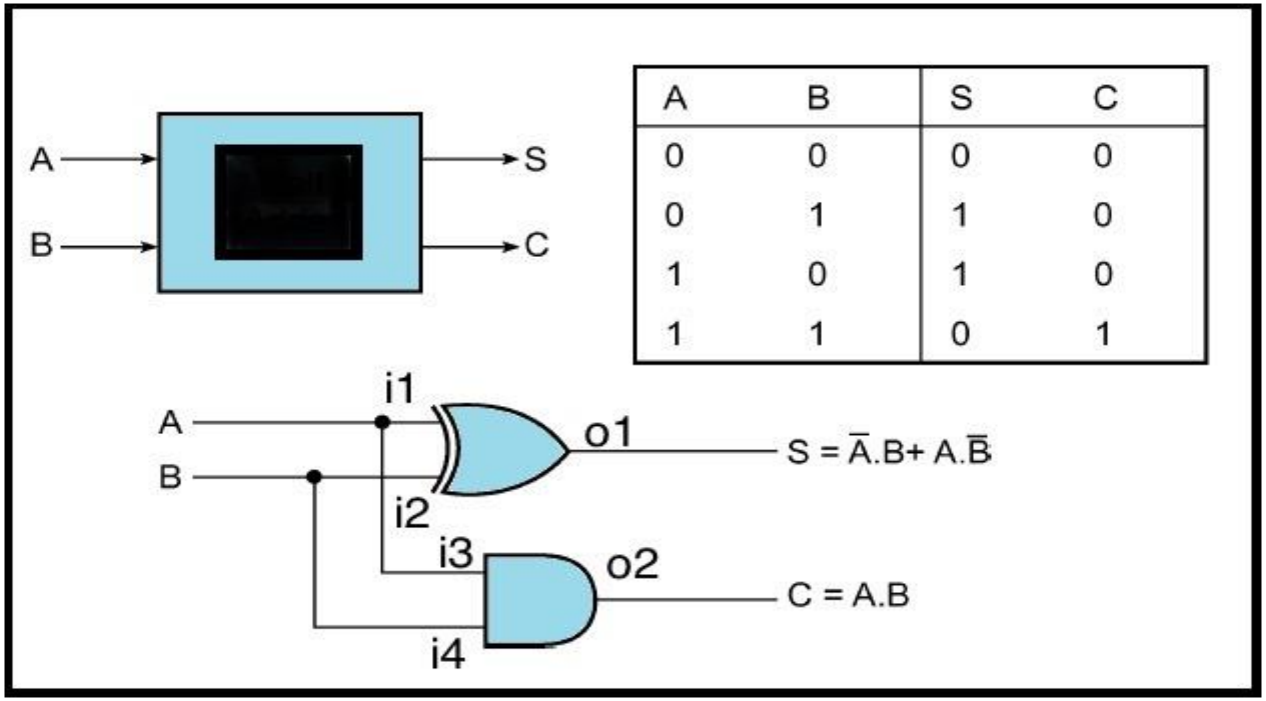
https://buzztech.in/vhdl-modelling-styles-behavioral-dataflow-structural/

VHDL Structural Modelling Style

```
library ieee;
use ieee.std_logic_1164.all;

entity circuit_1 is                          -- Entity declaration for circuit_1
port (A, B: in std_logic;
    S, C: out std_logic);
end circuit_1;

architecture structure of circuit_1 is  -- Architecture body for circuit_1
component xor_gate                               -- xor component declaration
    port (i1, i2: in std_logic;
      o1: out std_logic);
  end component;

  component and_gate                             -- and component declaration
    port (i3, i4: in std_logic;
      o2: out std_logic);
  end component;
```

VHDL Structural Modelling Style

```
begin
   u1: xor_gate port map (i1 => A, i2 => B, o1 => S);
   u2: and_gate port map (i3 => a, i4 => b, o2 => C);
-- We can also use Positional Association
--       => u1: xor_gate port map (a, b, sum);
--       => u2: and_gate port map (a, b, carry_out);
end structure;
```



A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

--Functionality for XOR gate in data flow modelling style

```
library ieee;
use ieee.std_logic_1164.all;

entity xor_gate is
  port (i1, i2: in std_logic;
   o1: out std_logic);
  end circuit_1;

architecture dataflow of xor_gate is
  Begin
  o1 <= i1 xor i2;
 end dataflow;
```

The **std_logic_vecto**r type can be used for creating signal buses in VHDL. The **std_logic** is the most commonly used type in VHDL, and the **std_logic_vecto**r is the array version of it.

Syntax used:

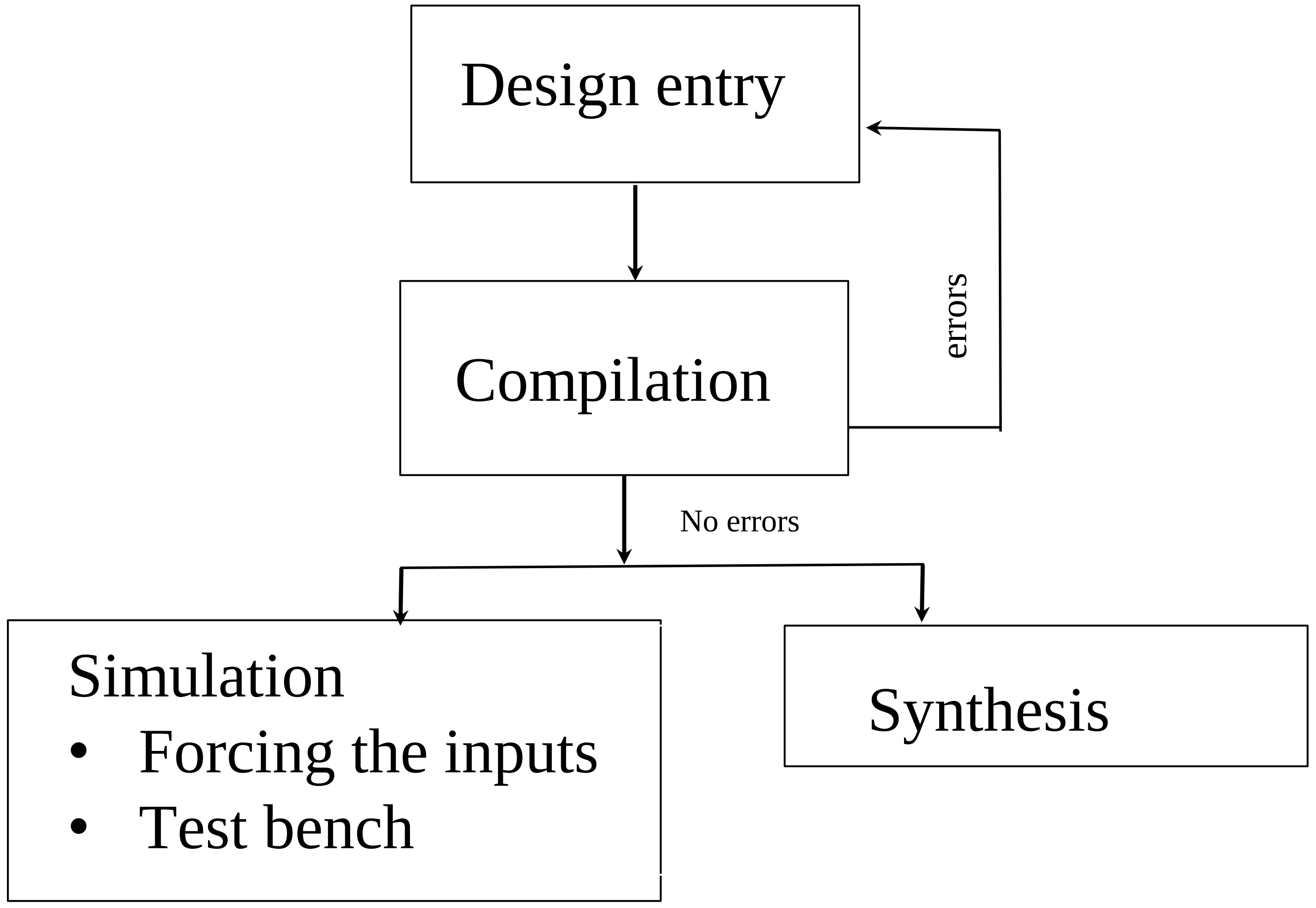
signal <name> : std_logic_vector(<lsb> to <msb>) := <initial_value>;

or

signal <name> : std_logic_vector(<msb> downto <lsb>) := <initial_value>;

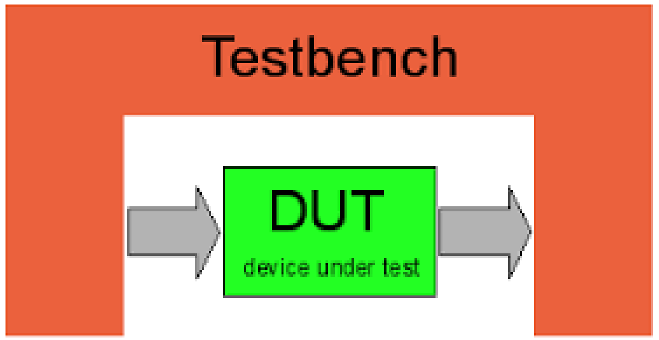
--Functionality for AND gate in data flow modelling style

```
library ieee;
use ieee.std_logic_1164.all;

entity and_gate is
  port (i3, i4: in std_logic;
   o2: out std_logic);
  end circuit_1;

architecture dataflow of and_gate is
  Begin
  o2 <= i3 and i4;
 end dataflow;
```


Design entry
errors
Compilation
No errors
Simulation
• Forcing the inputs
• Test bench
Synthesis

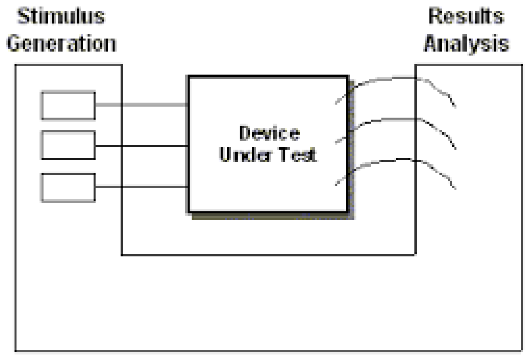
Test Bench

- Used for verification of the design
- HDL code with no port list and non synthesizable constructs.
- The main objectives of Test Bench is to:
- 1.Instantiate the design under test (DUT)
- 2.Generate stimulus waveforms for DUT
- 3.Generate reference outputs and compare them with the outputs of DUT
- 4.Automatically provide a pass or fail indication



VHDL Test Bench for Circuit 1

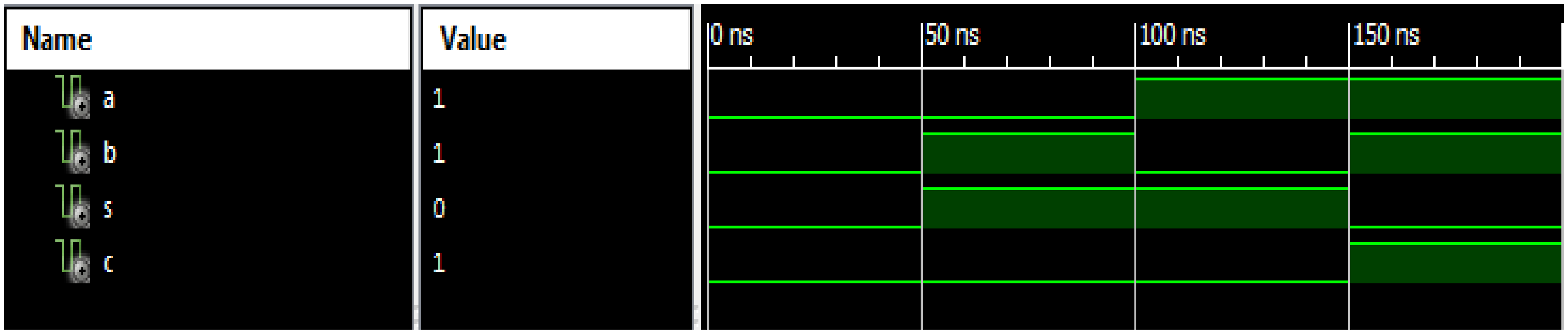
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY circuit_1_tb IS
END circuit_1_tb;

ARCHITECTURE circuit_1_tb OF circuit_1_tb IS

COMPONENT circuit_1
    PORT(
         A : IN  std_logic;
         B : IN  std_logic;
         S : OUT  std_logic;
         C : OUT  std_logic
        );
    END COMPONENT;
```



```
-- input test signals and resulting output signals from DUT to be
                              declared

          signal a_tb : std_logic := '0';
          signal b_tb :std_logic  := '0';
          signal s_tb :std_logic  ;
          signal c_tb :std_logic  ;
```

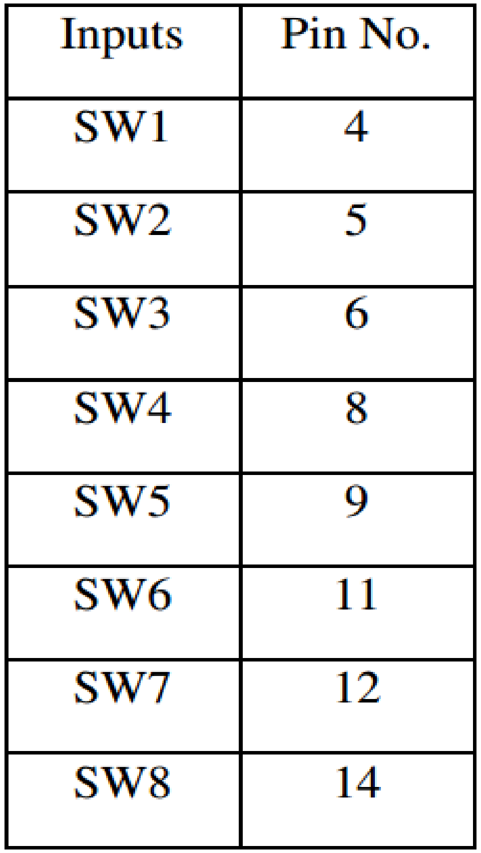
```
BEGIN

uut: circuit_1 PORT MAP
       (
          A => a_tb,
          B => b_tb,           -- instantaite the DUT with
          S => s_tb,           -- structual modelling style
          C => c_tb
        );

Process                         --apply the stimulus with
  Begin                         --behavioural modelling style
  a_tb <= '0';  b_tb <= '0';
  wait for 100 ps;
  a_tb <= '0';  b_tb <= '1';
      wait for 200 ps;
 a_tb<= '1';   b_tb <= '0';
      wait for 500 ps;
 a_tb <= '1';   b_tb <= '1';
      wait;
end process;

END circuit_1_tb ;
```

Simulation Results of Circuit 1

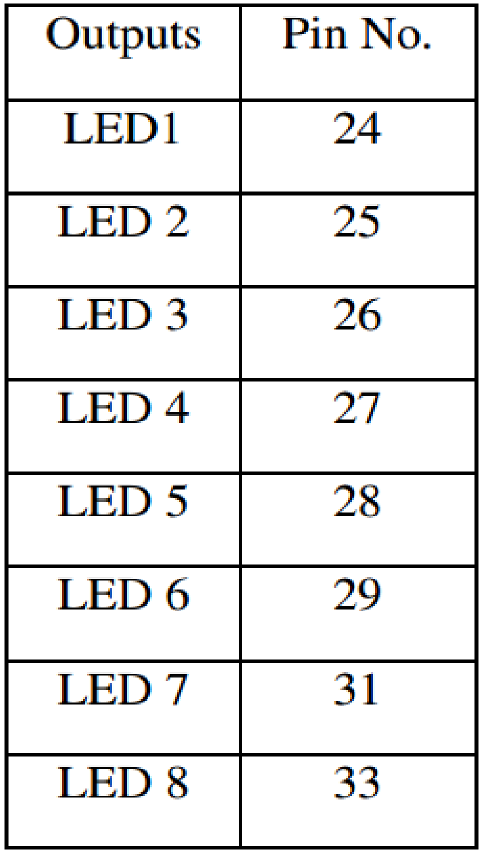


Name	Value
a	1
b	1
s	0
c	1

INPUT AND OUTPUT PINS DETAILS OF MAX 3000A



Inputs	Pin No.
SW1	4
SW2	5
SW3	6
SW4	8
SW5	9
SW6	11
SW7	12
SW8	14



Outputs	Pin No.
LED1	24
LED 2	25
LED 3	26
LED 4	27
LED 5	28
LED 6	29
LED 7	31
LED 8	33

Configuring Helium Board using JTAG

1. Open terminal -> jtag

2. cable ft2232 vid=0x0403 pid=0x6010

3. detect

4.svf “file path.svf”