# ITI Summer Training

## LAB  Python Project

```
Enter module name : module
illegal name please enter correct module name : #Adder
illegal name please enter correct module name : $Adder
illegal name please enter correct module name : 2adder
illegal name please enter correct module name : Adder
Enter the number of inputs signals : 3
Enter the name of  input number 1 : clk
Is the input parameterized ? (Y/N) : N
Enter the width of  input number 1 : 1
Enter the name of  input number 2 : rst
Is the input parameterized ? (Y/N) : N
Enter the width of  input number 2 : 1
Enter the name of  input number 3 : a
Is the input parameterized ? (Y/N) : Y
Enter the parameter name of  input number 3 : width
Enter the number of outputs signals : 1
Enter the name of  output number 1 : b
Enter the type of output number 1 (reg=R / wire=W) : R
Is the output parameterized ? (Y/N) : Y
Enter the parameter name of  output number 1 : width
Enter the number of parameters : 1
Enter the name of parameter number 1 : width
Enter the value of parameter number 1 : 5
Do you need reset signal ? (Y/N) : Y
Enter reset signal name : rst
1-Asynchronous
2-Synchronous
Chosse the type of reset : 1
1-posedge
2-negedge
Chosse the edge of reset :  2
```

```
1-Combinational
2-Sequential
3-Mixed
Choose type of your circuit : 3
Enter the clock signal name : clk
Is your clock level or edge triggered (L/E) : E
Enter the edge triggered of clock(P/N) : P
```

---

# The generated files

| | | | |
|---|---|---|---|
| Adder | 7/18/2024 9:31 PM | V File | 1 KB |
| Adder_TB | 7/18/2024 9:31 PM | V File | 1 KB |

```
module Adder (clk , rst , a , b);


************Parameters Declaration***********

parameter width = 5 ;



************Ports Declaration***********

input wire clk ;
input wire rst ;
input wire[width-1:0] a ;
output reg[width-1:0] b ;



************Implementation Code***********

always @(posedge clk,negedge rst) begin




end




always @(*) begin




end




endmodule
```

```
module Adder_TB ();


************Signals Declaration***********

reg clk ;
reg rst ;
reg[width-1:0] a ;


wire[width-1:0] b ;



************Module Instantiation***********

Adder  DUT (.clk(clk),.rst(rst),.a(a),.b(b));


************Clock Genertion***********

initial begin

   forever begin

    end

end



************Driving Stimuli***********

initial begin




end




endmodule
```

---