TRƯỜNG ĐẠI HỌC CÔNG NGHỆ
KHOA ĐIỆN TỬ VIỄN THÔNG

Digital Design & Microprocessors

Combinational Circuit Design 2

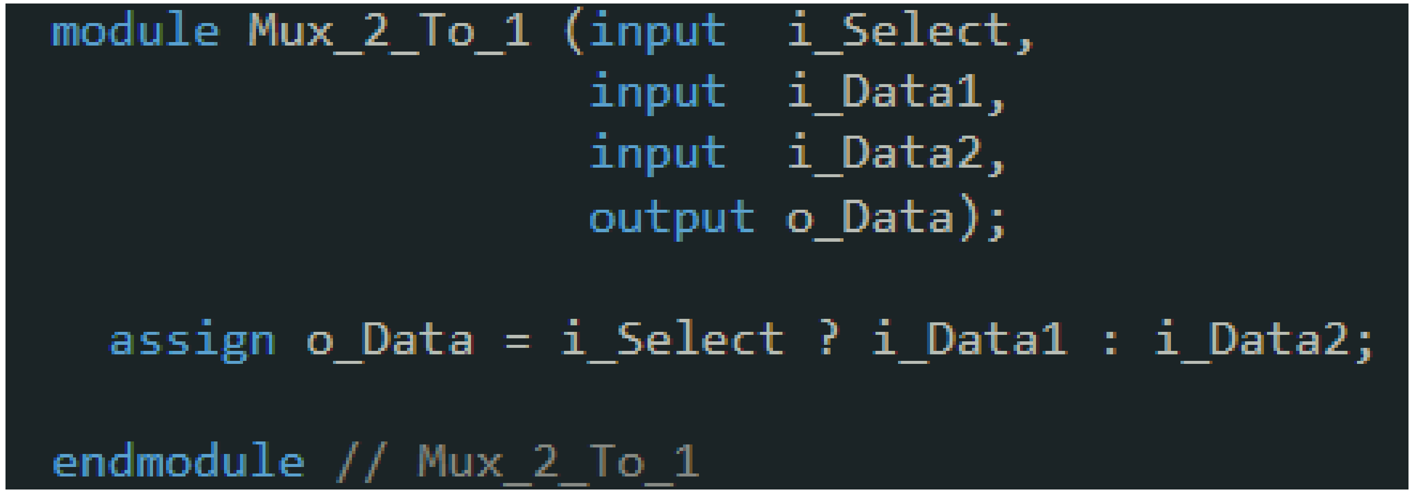
Assignments

Conditional assignment: select the output from among alternatives based on an input called the condition
Using assign statement and conditional operator ?:

Syntax:

assign y = <condition> ?: <value if condition is true> : <value if condition is false>

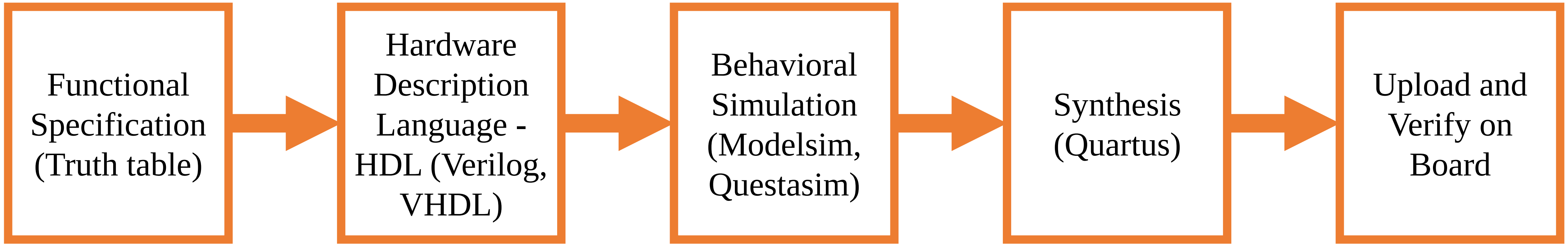
Based on value of condition to assign value for y



```
module Mux_2_To_1 (input  i_Select,
                   input  i_Data1,
                   input  i_Data2,
                   output o_Data);

  assign o_Data = i_Select ? i_Data1 : i_Data2;

endmodule // Mux_2_To_1
```

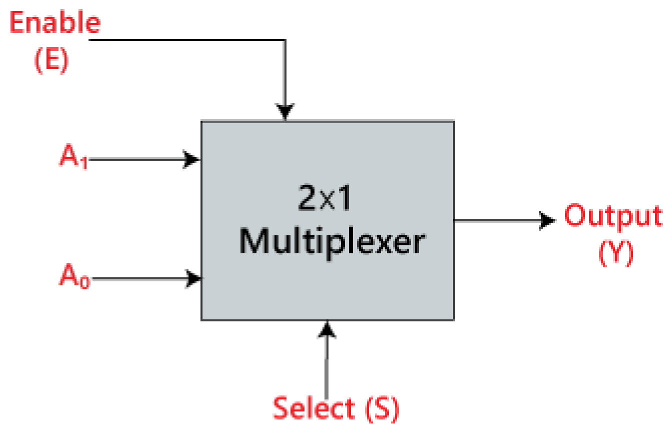
Example

Design flow

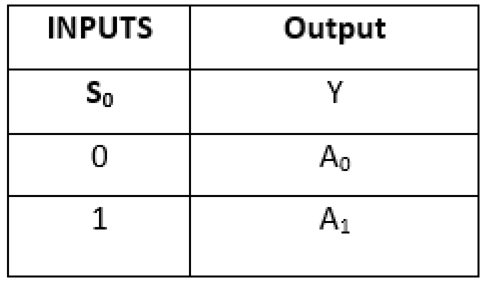

Functional Specification (Truth table)
Hardware Description Language - HDL (Verilog, VHDL)
Behavioral Simulation (Modelsim, Questasim)
Synthesis (Quartus)
Upload and Verify on Board

Multiplexer 2:1 (mux 2:1)

Construct output Boolean function: Base on truth table construct output Boolean functions and simplify them (Boolean algebra, K map).



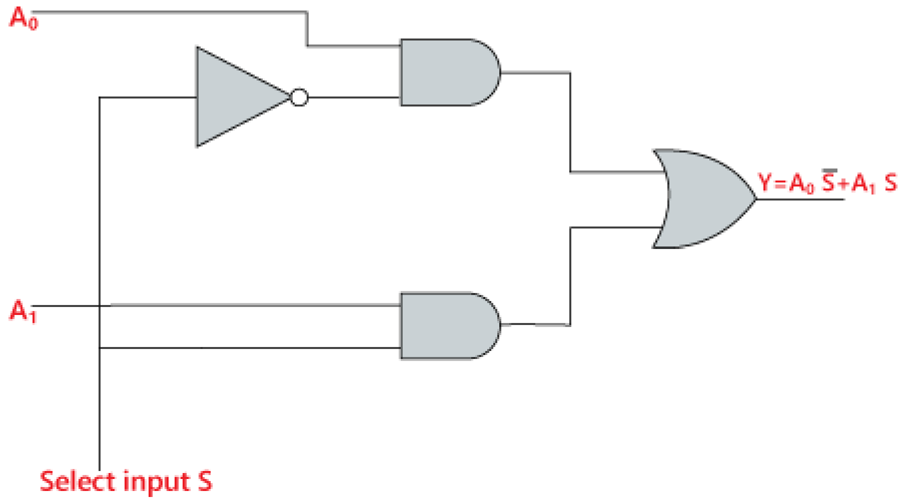
Block Diagram



INPUTS	Output
S_0	Y
0	A_0
1	A_1

Truth table

$Y=S_0'.A_0+S_0.A_1$



Circuit

Multiplexer 2:1 (mux 2:1)

Describe the circuit using Verilog

```
module Mux_2_to_1 (
                input  i_Select,
                input  i_Data1,
                input  i_Data2,
                output o_Data
);

  assign o_Data = i_Select ? i_Data1 : i_Data2;

endmodule
```

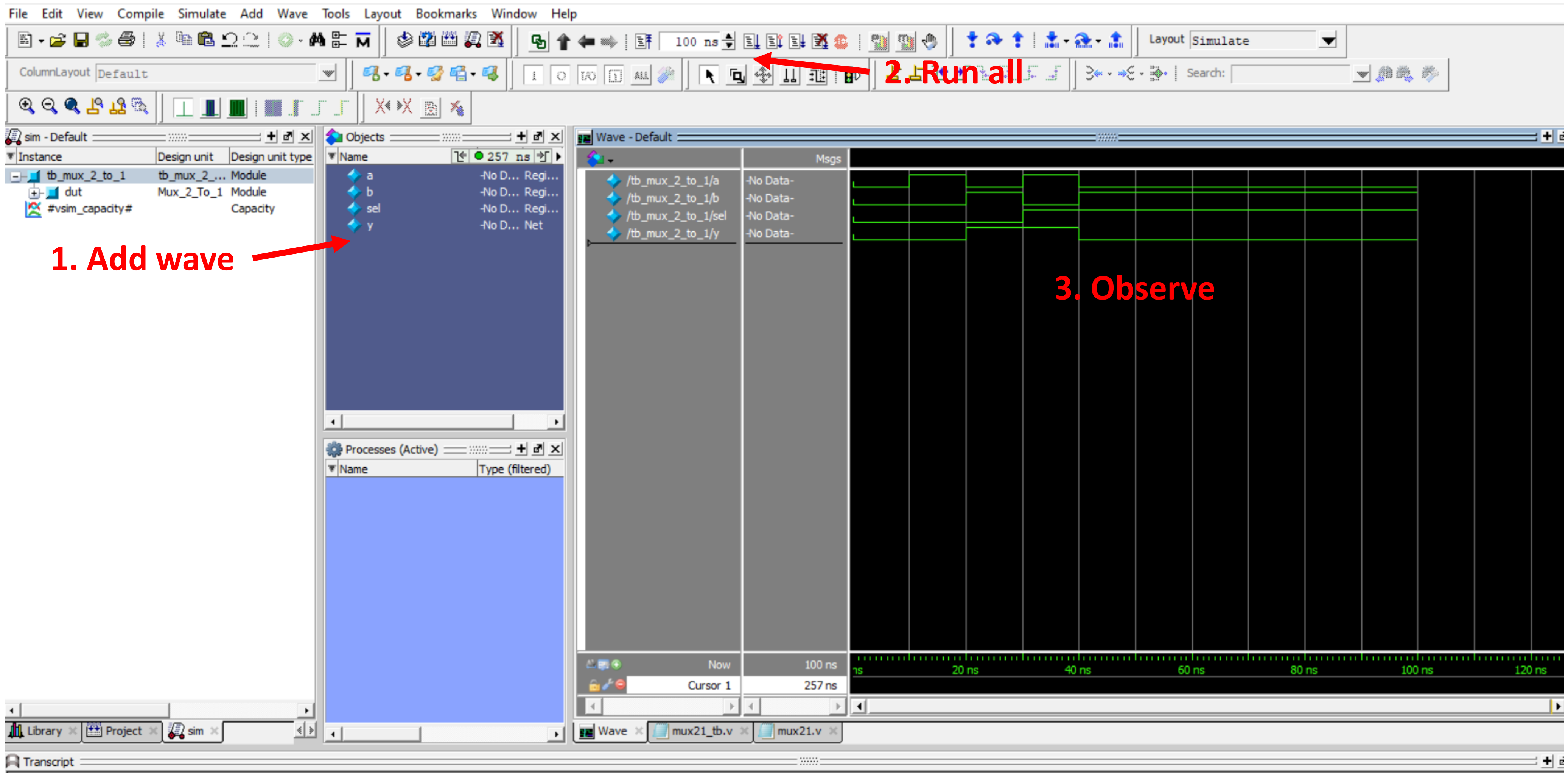
Multiplexer 2:1 (mux 2:1)

Write testbench for the module. DUT stand for *'device under test'*

```
`timescale 1ns/1ns // <time unit> / <time precision>

module tb_mux_2_to_1();
   // Testbench signals
   reg a;
   reg b;
   reg sel;
   wire y;

   // Instantiate the MUX
   Mux_2_to_1 dut (
      .a(a),
      .b(b),
      .sel(sel),
      .y(y)
   );

initial
begin
      // Initialize the inputs
      a = 0;
      b = 0;
      sel = 0;

      // Test case 1: sel = 0, should pass 'a' to 'y'
      #10 a = 1; b = 0; sel = 0;
      #10 a = 0; b = 1; sel = 0;
      // Test case 2: sel = 1, should pass 'b' to 'y'
      #10 a = 1; b = 0; sel = 1;
      #10 a = 0; b = 1; sel = 1;

      // End simulation
      #10 $finish;
   end
endmodule
```

Multiplexer 2:1 (mux 2:1)

Run simulation to check if testbench outputs match the expected outputs (truth table). Add all signal to the wave window then run all to observe and check.



Instance	Design unit	Design unit type
tb_mux_2_to_1	tb_mux_2_...	Module
dut	Mux_2_To_1	Module
#vsim_capacity#		Capacity

Name	257 ns	
a	-No D...	Regi...
b	-No D...	Regi...
sel	-No D...	Regi...
y	-No D...	Net

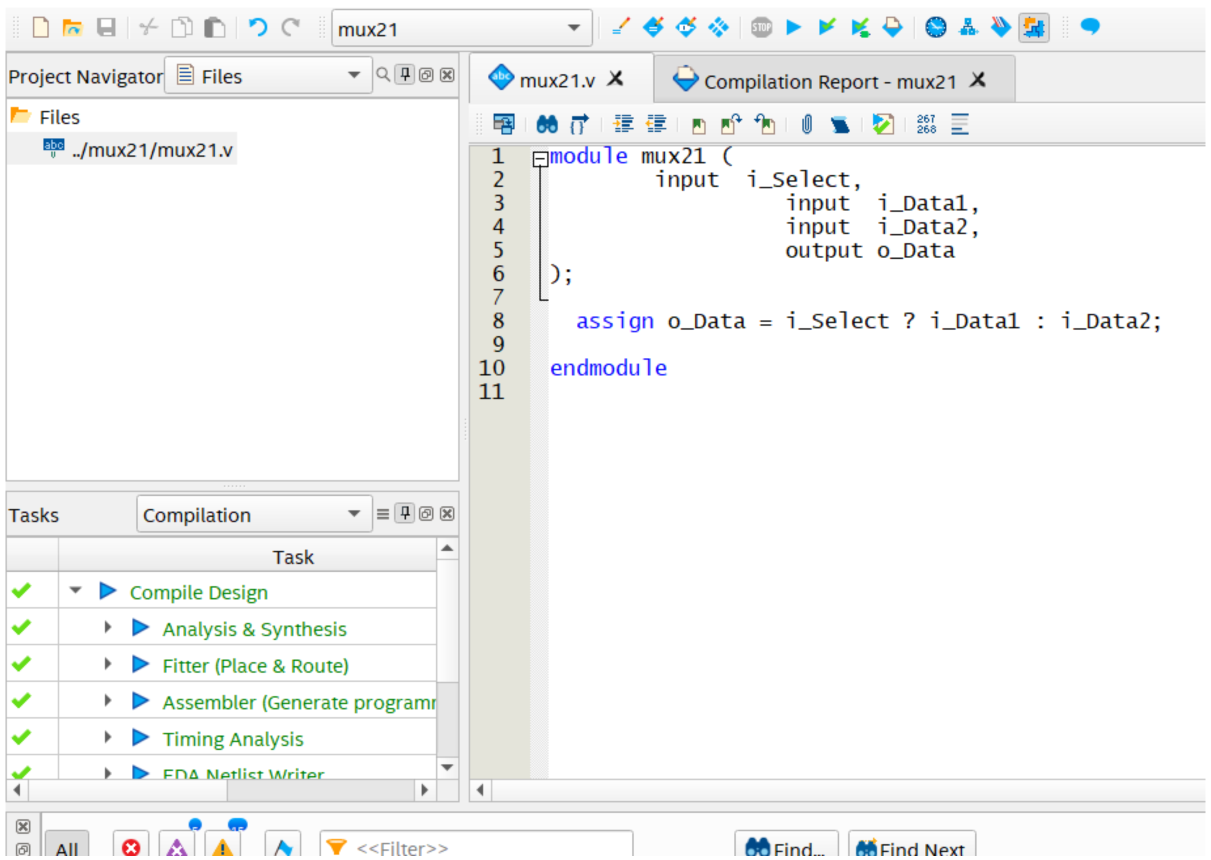
Name	Type (filtered)

	Msgs
/tb_mux_2_to_1/a	-No Data-
/tb_mux_2_to_1/b	-No Data-
/tb_mux_2_to_1/sel	-No Data-
/tb_mux_2_to_1/y	-No Data-
Now	100 ns
Cursor 1	257 ns

ns 20 ns 40 ns 60 ns 80 ns 100 ns 120 ns

Multiplexer 2:1 (mux 2:1)

Upload and verify on Board: Click on **Pin Planner** or press (Ctrl + Shift + N) then base on the device manual to assign signal to specific pin.

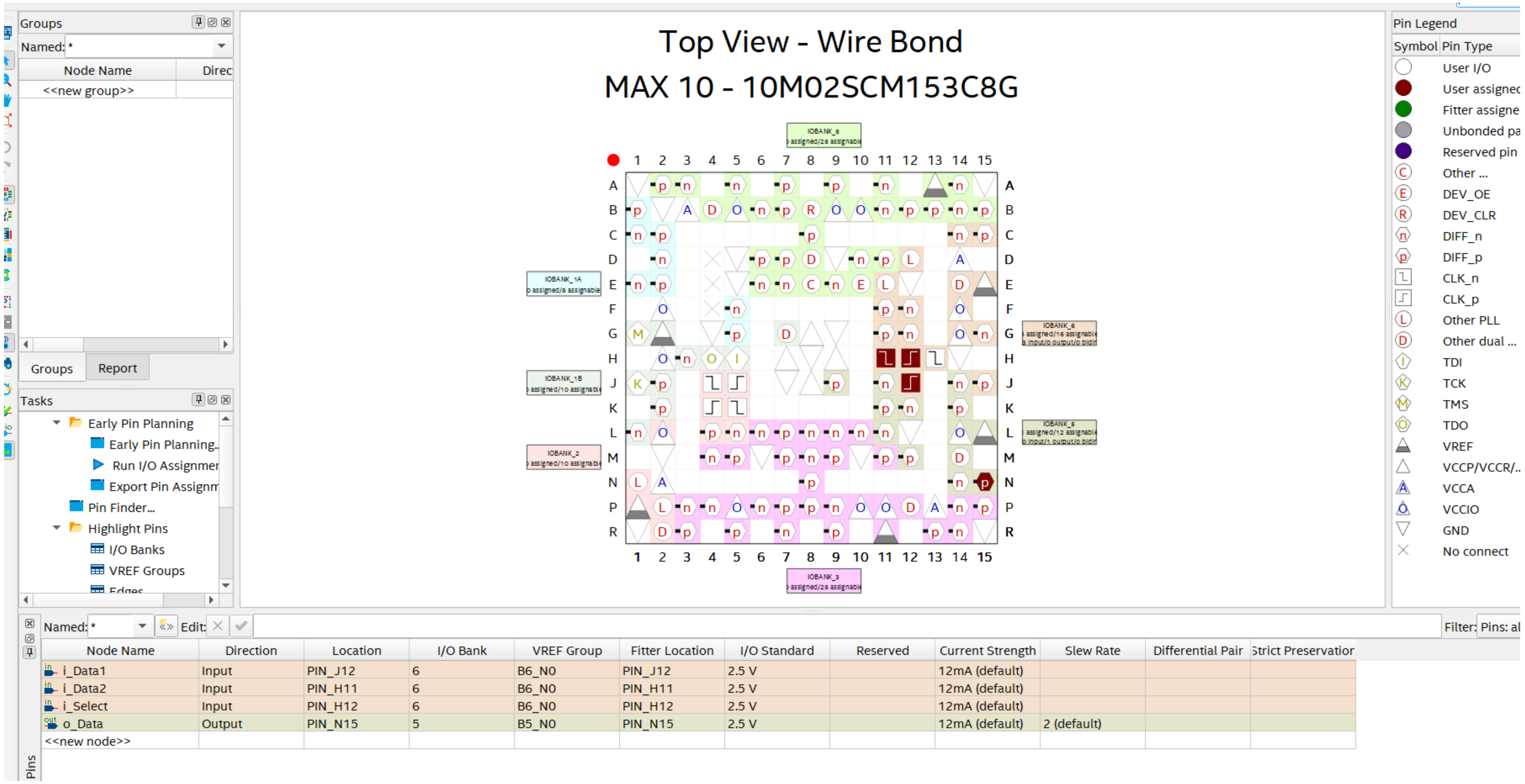


```verilog
module mux21 (
            input   i_Select,
                    input   i_Data1,
                    input   i_Data2,
                    output o_Data
);

  assign o_Data = i_Select ? i_Data1 : i_Data2;

endmodule
```

STEP PINs	FPGA PINs	STEP PINs	FPGA PINs	Digital Display1	FPGA PINs	12M CLOCK	FPGA PINs
3.3V		VBUS		SEG-A1	E1	PCLK	J5
GPIO0	M4	GPIO35	B4	SEG-B1	D2	LED	FPGA PINs
GPIO1	P3	GPIO34	A5	SEG-C1	K2	LED1	N15
GPIO2	M5	GPIO33	A7	SEG-D1	J2	LED2	N14
GPIO3	R3	GPIO32	B6	SEG-E1	G2	LED3	M14
GPIO4	L6	GPIO31	E7	SEG-F1	F5	LED4	M12
GPIO5	P4	GPIO30	D7	SEG-G1	G5	LED5	L15
GPIO6	L7	GPIO29	B7	SEG-DP1	L1	LED6	K12
GPIO7	R5	GPIO28	C8	SEG-DIG1	E2	LED7	L11
GPIO8	P6	GPIO27	B8	Digital Display2	FPGA PINs	LED8	K11
GPIO9	R7	GPIO26	D10			Switch	FPGA PINs
GPIO10	P7	GPIO25	A9	SEG-A2	A3	SW1	J12
GPIO11	P8	GPIO24	A11	SEG-B2	A2	SW2	H11
GPIO12	P9	GPIO23	A13	SEG-C2	P2	SW3	H12
GPIO13	R9	GPIO22	B11	SEG-D2	P1	SW4	H13
GPIO14	R11	GPIO21	A14	SEG-E2	N1	Button	FPGA PINs
GPIO15	P12	GPIO20	B13	SEG-F2	C1	KEY1	J9
GPIO16	R14	GPIO19	B14	SEG-G2	C2	KEY2	K14
GPIO17	P15	GPIO18	B15	SEG-DP2	R2	KEY3	J11
GND		GND		SEG-DIG2	B1	KEY4	J14
RGB LED1	R	G	B	RGB_LED2	R	G	B
FPGA PINs	G15	E15	E14	FPGA PINs	C15	C14	D12

Multiplexer 2:1 (mux 2:1)

Upload and verify on Board: base on the device manual to assign signal to specific pin. Assign input for **Switch** pins and output for **LED** pins. After finish pin planning close the tab and then click on **Processing->Start Compination** to compile

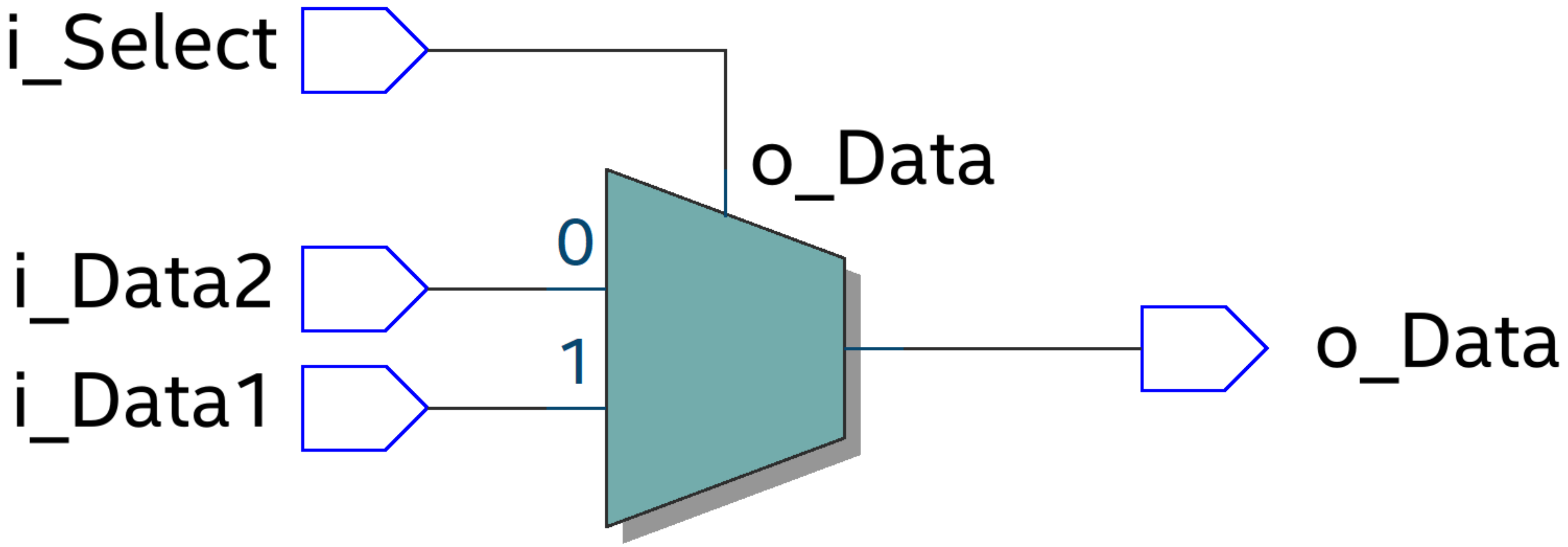


Symbol	Pin Type
○	User I/O
●	User assigned
●	Fitter assigne
●	Unbonded pa
●	Reserved pin
Ⓒ	Other ...
Ⓔ	DEV_OE
Ⓡ	DEV_CLR
ⓝ	DIFF_n
ⓟ	DIFF_p
⎍	CLK_n
⎍	CLK_p
Ⓛ	Other PLL
Ⓓ	Other dual ...
Ⓘ	TDI
Ⓚ	TCK
Ⓜ	TMS
Ⓞ	TDO
▲	VREF
△	VCCP/VCCR/..
Ⓐ	VCCA
Ⓞ	VCCIO
▽	GND
✕	No connect

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservatior
i_Data1	Input	PIN_J12	6	B6_N0	PIN_J12	2.5 V		12mA (default)			
i_Data2	Input	PIN_H11	6	B6_N0	PIN_H11	2.5 V		12mA (default)			
i_Select	Input	PIN_H12	6	B6_N0	PIN_H12	2.5 V		12mA (default)			
o_Data	Output	PIN_N15	5	B5_N0	PIN_N15	2.5 V		12mA (default)	2 (default)		
<<new node>>											

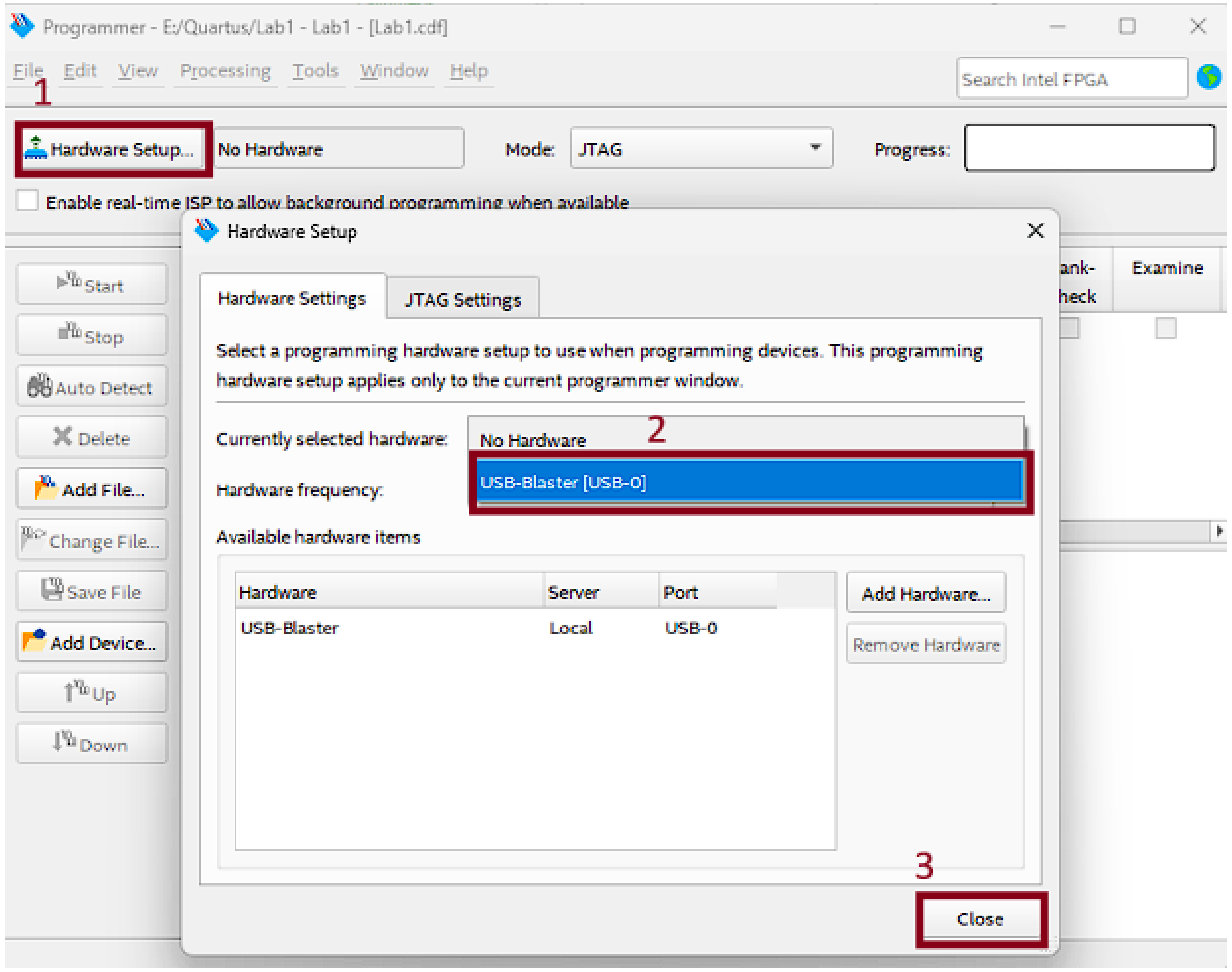
Multiplexer 2:1 (mux 2:1)

Synthesize Design in Quartus: after combile project and then click on **Tools->Netlist Viewers->RLT Viewer**



Multiplexer 2:1 (mux 2:1)

Upload and verify on Board: Click on **Tools->Programmer** then select hardware setup as below and then click on Start to upload on boad. Finally verify your design on board



Enable real-time ISP to allow background programming when available

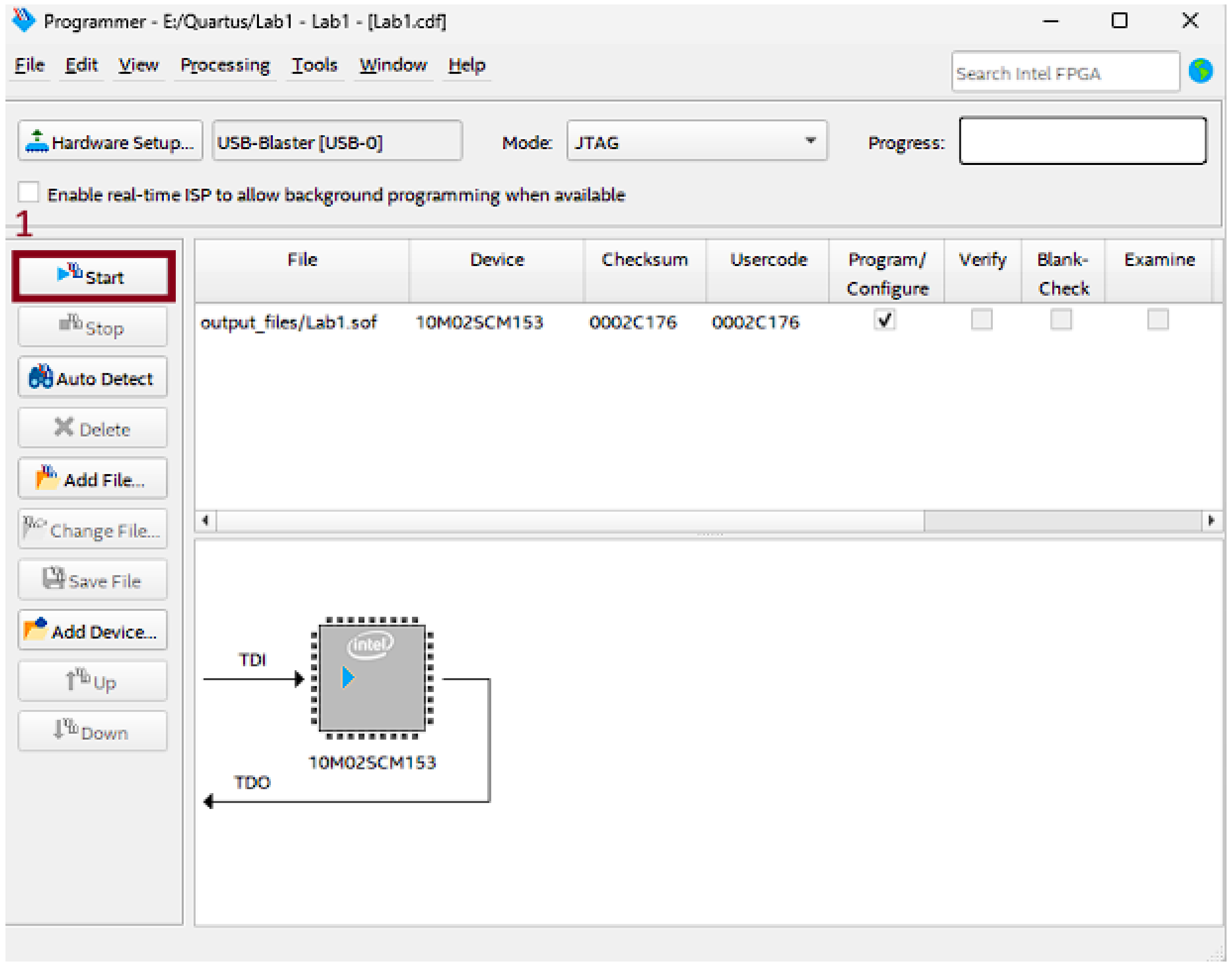
Select a programming hardware setup to use when programming devices. This programming hardware setup applies only to the current programmer window.

Currently selected hardware: No Hardware / USB-Blaster [USB-0]

Hardware frequency:

Available hardware items

Hardware	Server	Port
USB-Blaster	Local	USB-0



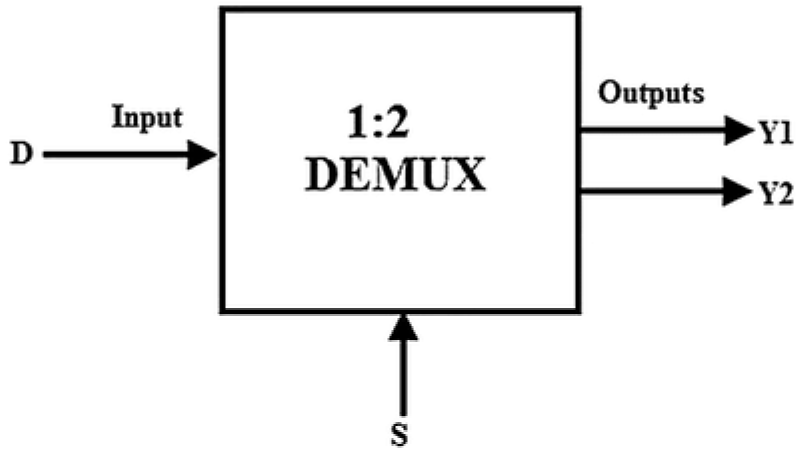
Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine
output_files/Lab1.sof	10M02SCM153	0002C176	0002C176	✓			

1-to-2 Demultiplexer

Select	Input	Outputs	
S	D	Y_2	Y_1
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	0

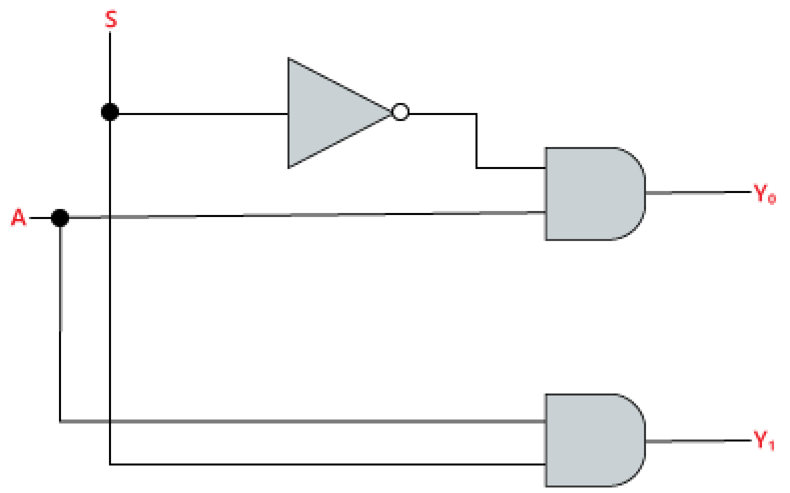
Truth Table



Block Diagram

```
module demux_1_to_2 (
    input wire din,   // Input data
    input wire sel,   // Select signal
    output wire y0,   // Output 0
    output wire y1    // Output 1
);
    // Assign outputs based on select signal
    assign y0 = (sel == 0) ? din : 0;
    assign y1 = (sel == 1) ? din : 0;
endmodule
```

$Y_0=S_0'.A$
$Y_1=S_0.A$



Exercises

Ex1: Design a tri-state buffer using verilog, simulate on ModelSim
Ex2: Design a 4:1 multiplexer using Verilog, simulate on ModelSim.
Ex3: Design a 1:4 De-multiplexer using Verilog, simulate on ModelSim.