#100 DAYS OF RTL

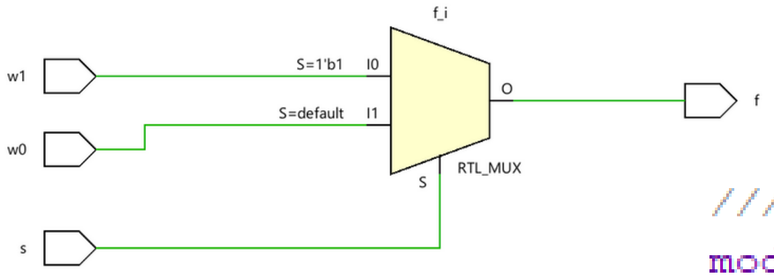
DAY 2/100 Insights

MULTIPLEXERS

Explored and Implemented different methods of implementing 2x1 MUX

Using assign statement

```
module mux2to1 (w0, w1, s, f);
input w0, w1, s;
output f;
assign f = s ? w1 : w0;
endmodule
```



Using if-else

```
module mux2to1 (w0, w1, s, f);
input w0, w1, s;
output reg f;
always @(w0, w1, s)
if (s == 0)
f = w0;
else
f = w1;
endmodule
```

using always block

s	f
0	w_0
1	w_1

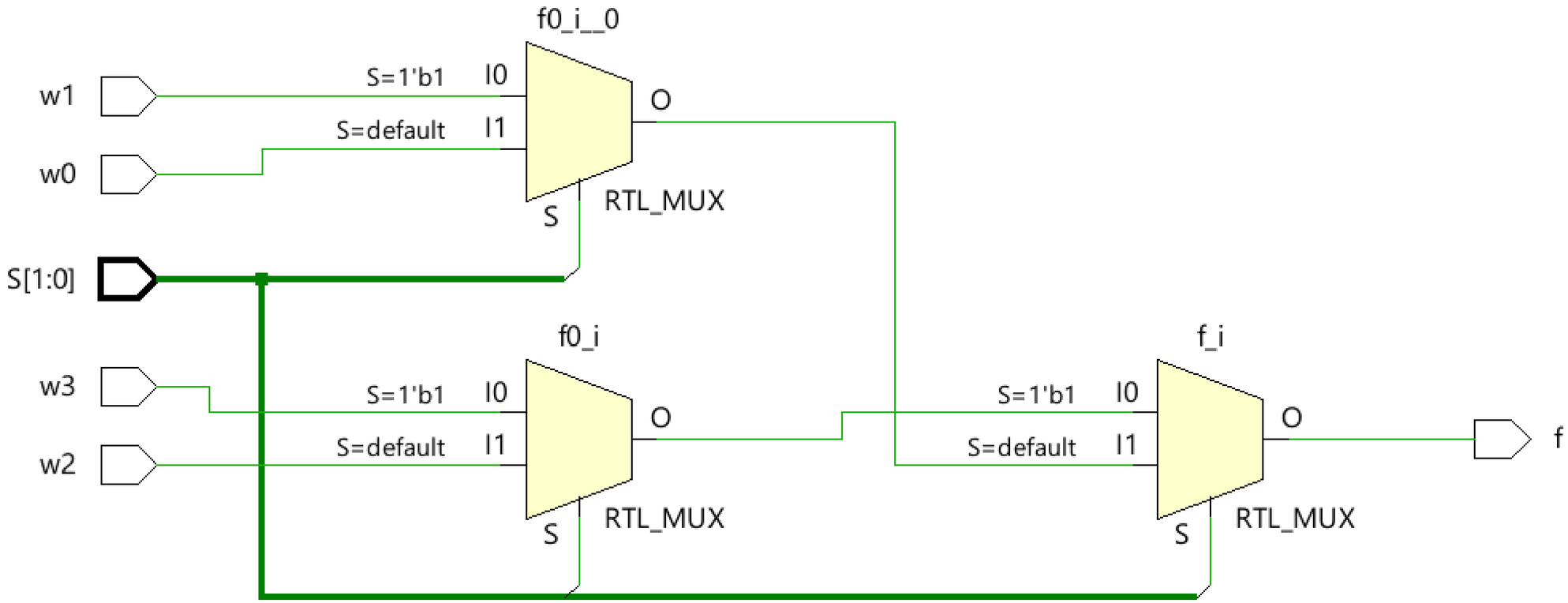
What is different in this than assign statement

<u>It is not continously being evaluated</u>

<u>only gets evaluated when always statement gets activated</u>

```
module mux2to1 (w0, w1, s, f);
input w0, w1, s;
output reg f;
always @(w0, w1, s)
f = s ? w1 : w0;
endmodule
```

Different Methods of implementing 4x1 MUX

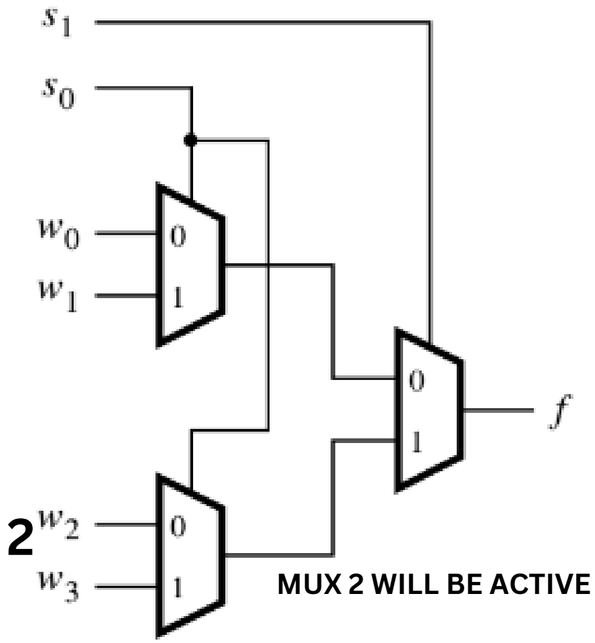


Method 1 conditional Operator

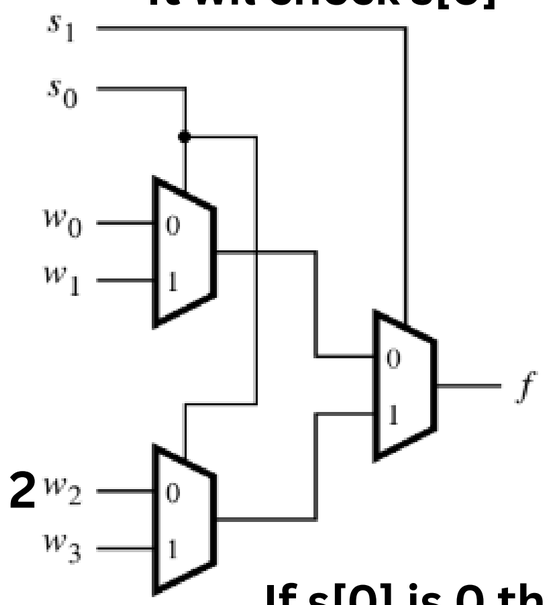
```
module mux4to1 (w0, w1, w2, w3, S, f);
input w0, w1, w2, w3;
input [1:0] S;
output f;
assign f = S[1] ? (S[0] ? w3 : w2) : (S[0] ? w1 : w0);
endmodule
```

How it works?

Case 1 s[1]=1

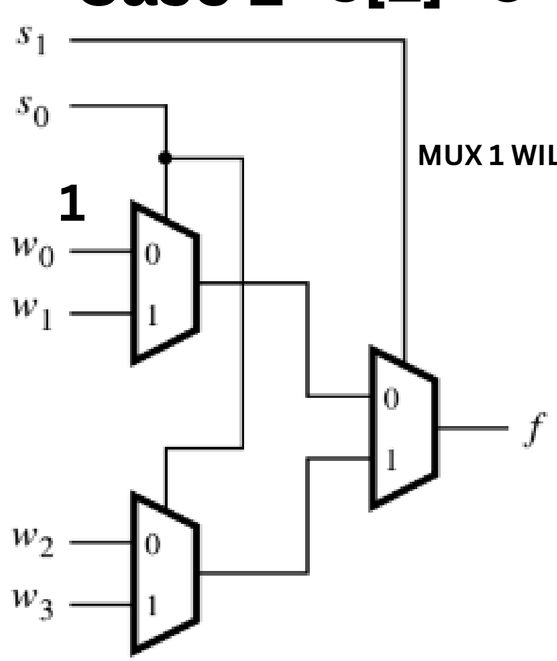


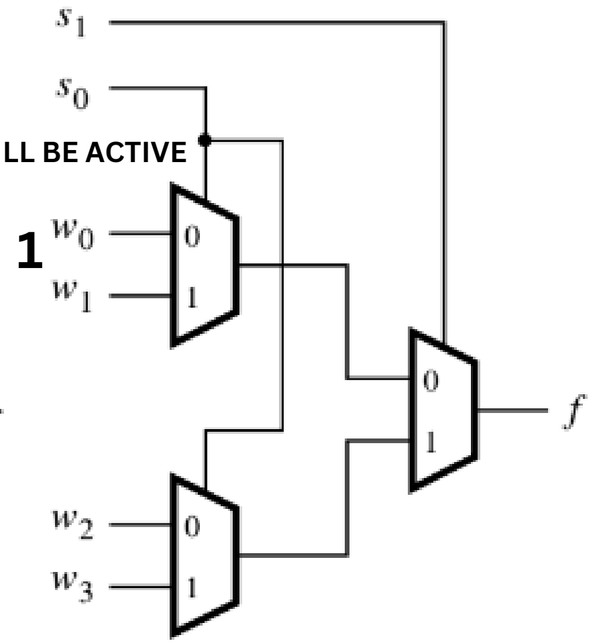
After checking s[1] condition
It wil check s[0]



If s[0] is 0 than f=w2 else w3

Case 2 s[1]=0





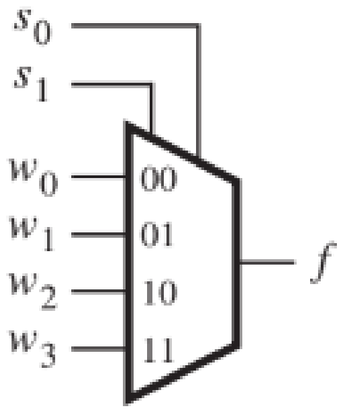
If s[0] is 0 than f=w0 else w1

Method 2 If else

```
module mux4to1 (w0, w1, w2, w3, S, f);
input w0, w1, w2, w3;
input [1:0] S;
output reg f;
always @(*)
if (S == 2'b00)
f = w0;
else if (S == 2'b01)
f = w1;
else if (S == 2'b10)
f = w2;
else
f = w3;
endmodule
```

Method 3 CASE statement

```
module mux4to1 (W, S, f);
input [0:3] W;
input [1:0] S;
output reg f;
always @(W, S)
case (S)
0: f = W[0];
1: f = W[1];
2: f = W[2];
3: f = W[3];
endcase
endmodule
```

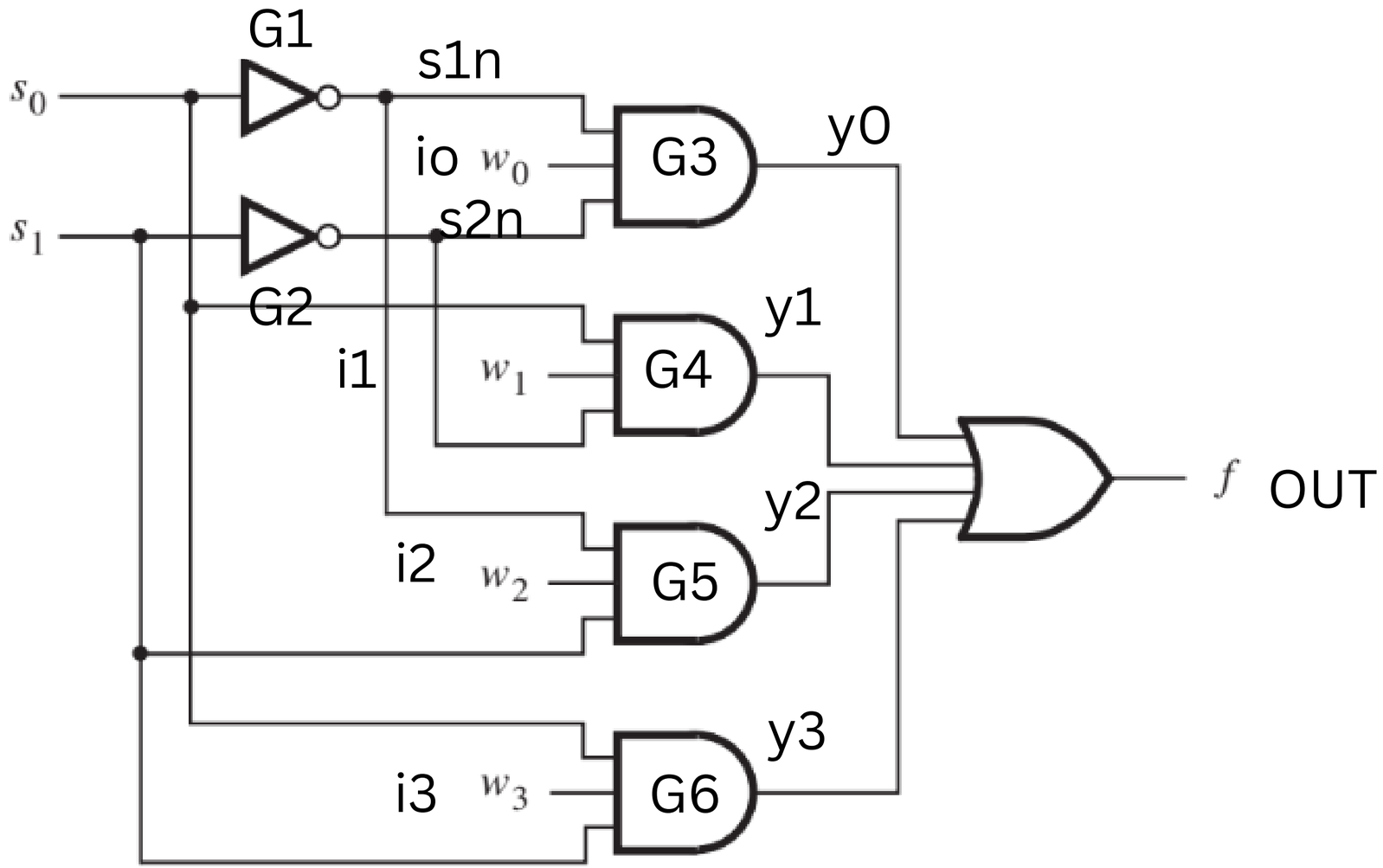


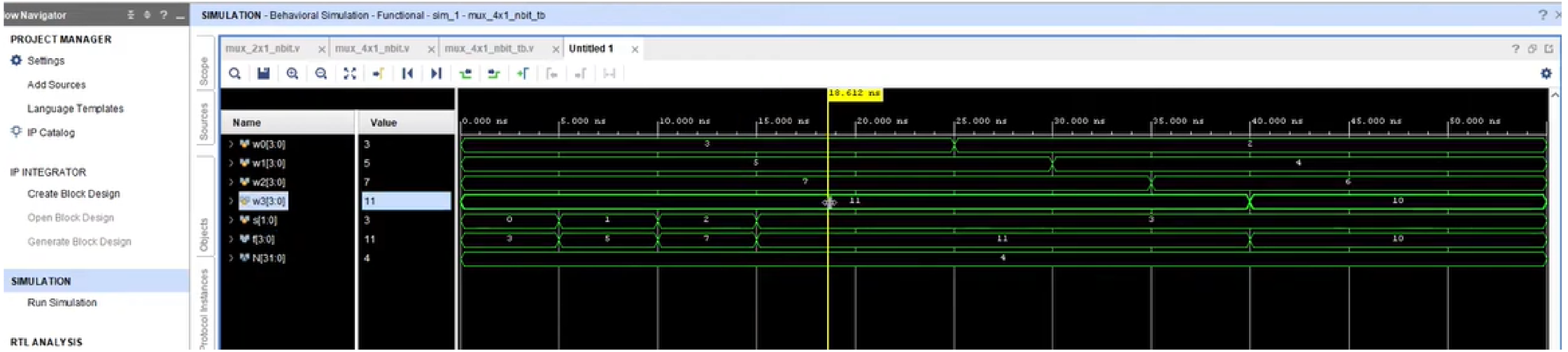
s_1	s_0	f
0	0	w_0
0	1	w_1
1	0	w_2
1	1	w_3

4X1_GATE_LEVEL_MODELING

```
module mux_4_to_1( out ,  i0 , i1 , i2 , i3 , s1 , s0 );
output out;
input i0 , i1 , i2 , i3 ;
input s0 , s1 ;
wire s1n , s0n , y1 , y2 , y3 ;
not G1( s1n, s1) ;
not G2( s0n, s2) ;
and  G3(y0 , i0 , s1n, s0n);
and  G4(y1 , i1 , s1n, s0);
and  G5(y2 , i2 , s1, s0n);
and  G6(y3 , i3 , s1, s0);
or G7(out , y0, y1, y2, y3);
endmodule
```

How it works




ow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
SIMULATION - Behavioral Simulation - Functional - sim_1 - mux_4x1_nbit_tb
mux_2x1_nbit.v
mux_4x1_nbit.v
mux_4x1_nbit_tb.v
Untitled 1
Scope
Sources
Objects
Protocol Instances
Name
Value
w0[3:0]
w1[3:0]
w2[3:0]
w3[3:0]
s[1:0]
f[3:0]
N[31:0]
3
5
7
11
3
11
4
18.612 ns
0.000 ns
5.000 ns
10.000 ns
15.000 ns
20.000 ns
25.000 ns
30.000 ns
35.000 ns
40.000 ns
45.000 ns
50.000 ns

More interesting Methods of implementing MUX On Day 3!