

# Code for HALF ADDER

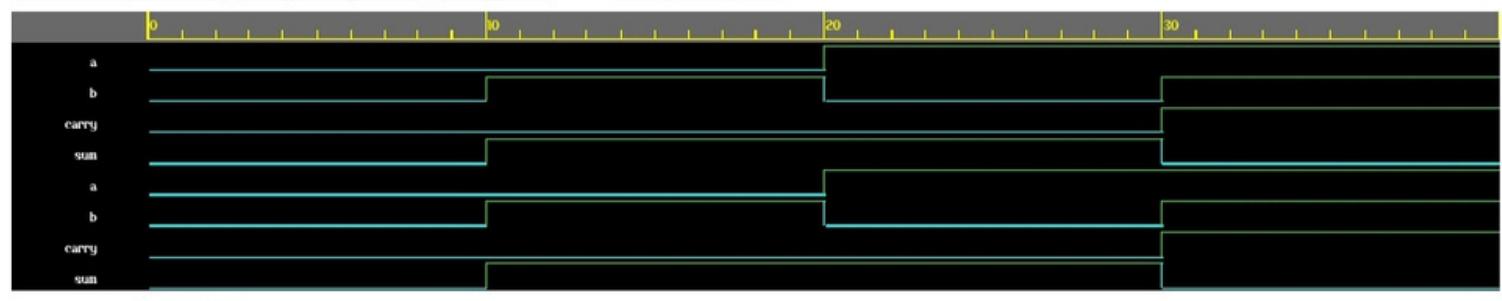
```
1 // Code your testbench here
2 // or browse Examples
3 module tb_half_adder;
4 reg a,b;
5 wire sum, carry;
6
7 half_adder
8 uut(.a(a),.b(b),.sum(sum),.carry(carry));
9
10 initial begin
11 $dumpfile("half_adder.vcd");
12 $dumpvars(0,tb_half_adder);
13 $monitor("Time=%t | A=%b B=%b => SUM=%b
14 carry=%b", $time,a,b,sum,carry);
15
16 a=0;b=0;
17 #10;a=0;b=1;
18 #10;a=1;b=0;
19 #10;a=1;b=1;
20
21 #10;
22 $finish;
23 end
24 endmodule
25
```

```
1 // Code your design here
2 module half_adder (
3   input a, b,
4   output sum,carry
5 );
6   assign sum = a^b;
7   assign carry = a&b;
8 endmodule
```

## Output:

Time=0 | A=0 B=0 => SUM=0 CARRY=0  
Time=10 | A=0 B=1 => SUM=1 CARRY=0  
Time=20 | A=1 B=0 => SUM=1 CARRY=0  
Time=30 | A=1 B=1 => SUM=0 CARRY=1

## Waveform:



# Code for FULL ADDER

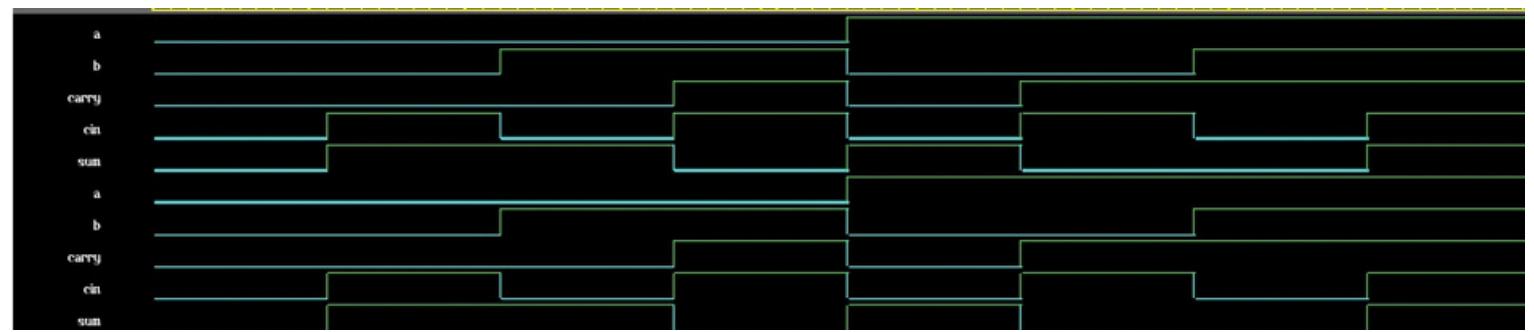
```
1 // Code your testbench here
2 // or browse Examples
3 module tb_full_adder;
4 reg a,b,cin;
5 wire sum, carry;
6
7 full_adder
8 uut(.a(a),.b(b),.cin(cin),.sum(sum),.carry
9 (carry));
10
11 initial begin
12   $dumpfile("full_adder.vcd");
13   $dumpvars(0,tb_full_adder);
14
15   $monitor("Time=%t | A=%b B=%b cin=%b =>
16   SUM=%b carry=%b",
17   $time,a,b,cin,sum,carry);
18
19   a=0;b=0;cin=0;
20   #10;a=0;b=0;cin=1;
21   #10;a=0;b=1;cin=0;
22   #10;a=0;b=1;cin=1;
23   #10;a=1;b=0;cin=0;
24   #10;a=1;b=0;cin=1;
25   #10;a=1;b=1;cin=0;
26   #10;a=1;b=1;cin=1;
27   #10;
28
29   $finish;
30   end
31 endmodule
```

```
1 // Code your design here
2 module full_adder (
3   input a, b,cin,
4   output sum,carry
5 );
6   assign sum = a^b^cin;
7   assign carry = (a&b)|(b&cin)|(a&cin);
8 endmodule
```

## Output:

Time=0 | A=0 B=0 Cin=0 => SUM=0 CARRY=0  
Time=10 | A=0 B=0 Cin=1 => SUM=1 CARRY=0  
Time=20 | A=0 B=1 Cin=0 => SUM=1 CARRY=0  
Time=30 | A=0 B=1 Cin=1 => SUM=0 CARRY=1  
Time=40 | A=1 B=0 Cin=0 => SUM=1 CARRY=0  
Time=50 | A=1 B=0 Cin=1 => SUM=0 CARRY=1  
Time=60 | A=1 B=1 Cin=0 => SUM=0 CARRY=1  
Time=70 | A=1 B=1 Cin=1 => SUM=1 CARRY=1

# Waveform:



# Code for HALF SUBTRACTER

```
1 // Code your testbench here
2 // or browse Examples
3 module tb_half_sub;
4 reg a,b;
5 wire diff,borrow;
6
7 half_sub
8 uut(.a(a),.b(b),.diff(diff),.borrow(borrow));
9
10 initial begin
11 $dumpfile("half_sub.vcd");
12 $dumpvars(0,tb_half_sub);
13
14 $monitor("Time=%t | A=%b B=%b => DIFF=%b
15 BORROW=%b", $time,a,b,diff,borrow);
16
17 a=0;b=0;
18 #10;a=0;b=1;
19 #10;a=1;b=0;
20 #10;a=1;b=1;
21 #10;
22
23 $finish;
24 end
25 endmodule
```

```
1 // Code your design here
2 module half_sub(
3   input a, b,
4   output diff,borrow
5 );
6 assign diff= a^b;
7 assign borrow = ~a&b;
8 endmodule
```

## Output:

Time=0 | A=0 B=0 => DIFF=0 BORROW=0  
Time=10 | A=0 B=1 => DIFF=1 BORROW=1  
Time=20 | A=1 B=0 => DIFF=1 BORROW=0  
Time=30 | A=1 B=1 => DIFF=0 BORROW=0

## Waveform:



# Code for FULL SUBTRACTER

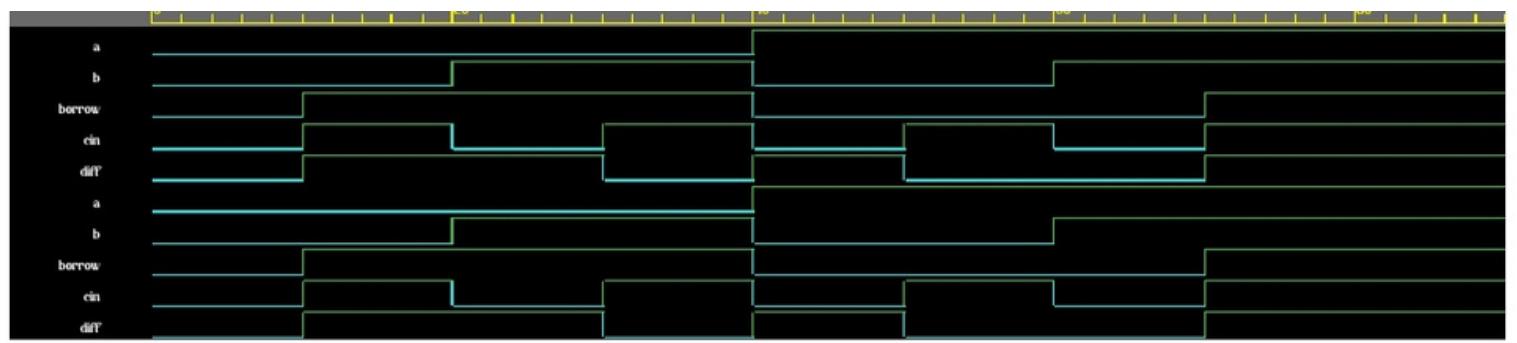
```
1 // Code your testbench here
2 // or browse Examples
3
4 module full_subtractor_tb;
5   reg a, b, cin;
6   wire diff, borrow;
7
8   full_subtractor uut (.a(a),
9   .b(b),.cin(cin),.diff(diff),.borrow(borrow)
10 );
11
12   initial begin
13     $dumpfile("full_subtractor.vcd");
14     $dumpvars(0, full_subtractor_tb);
15
16     $monitor("Time=%0t | A=%b B=%b
17     Cin=%b => DIFF=%b BORROW=%b", $time, a, b,
18     cin, diff, borrow);
19
20     a=0; b=0; cin=0; #10;
21     a=0; b=0; cin=1; #10;
22     a=0; b=1; cin=0; #10;
23     a=0; b=1; cin=1; #10;
24     a=1; b=0; cin=0; #10;
25     a=1; b=0; cin=1; #10;
26     a=1; b=1; cin=0; #10;
27     a=1; b=1; cin=1; #10;
28
29     #10;
30     $finish;
31   end
32 endmodule
```

```
1 // Code your design here
2 module full_subtractor (
3   input a, b, cin,
4   output diff, borrow
5 );
6   assign diff = a ^b ^cin;
7   assign borrow = (~a & b) | (~a ^ b)
8   & cin);
9 endmodule
```

## Output:

Time=0 | A=0 B=0 Cin=0 => DIFF=0 BORROW=0  
Time=10 | A=0 B=0 Cin=1 => DIFF=1 BORROW=1  
Time=20 | A=0 B=1 Cin=0 => DIFF=1 BORROW=1  
Time=30 | A=0 B=1 Cin=1 => DIFF=0 BORROW=1  
Time=40 | A=1 B=0 Cin=0 => DIFF=1 BORROW=0  
Time=50 | A=1 B=0 Cin=1 => DIFF=0 BORROW=0  
Time=60 | A=1 B=1 Cin=0 => DIFF=0 BORROW=1

# Waveform:



# Code for 2:1 multiplexer

```
1 // Code your testbench here
2 // or browse Examples
3 module tb_mux2_1;
4 reg a,b,sel;
5 wire y;
6
7 mux2_1 uut(.a(a),.b(b),.sel(sel),.y(y));
8
9 initial begin
10 $dumpfile("mux2_1.vcd");
11 $dumpvars(0,tb_mux2_1);
12
13 $monitor("Time=%0t | A=%b B=%b sel=%b =>
14 y=%b",$time,a,b,sel,y);
15
16 a = 0; b = 0; sel = 0;
17 #10; a = 0; b = 1; sel = 0;
18 #10; a = 0; b = 1; sel = 1;
19 #10; a = 1; b = 0; sel = 1;
20 #10; a = 1; b = 1; sel = 0;
21 #10; a = 1; b = 1; sel = 1;
22 #10;
23
24 $finish;
25 end
26 endmodule
```

```
1 // Code your design here
2 module mux2_1(
3   input a,b,sel,
4   output y );
5
6 assign y= sel?b:a;
7 endmodule
8
```

## Output:

Time=0 | A=0 B=0 SEL=0 => Y=0  
Time=10 | A=0 B=1 SEL=0 => Y=0  
Time=20 | A=0 B=1 SEL=1 => Y=1  
Time=30 | A=1 B=0 SEL=1 => Y=0  
Time=40 | A=1 B=1 SEL=0 => Y=1  
Time=50 | A=1 B=1 SEL=1 => Y=1

## Waveform:



# Code for 4:1 multiplexer

```
1 // Code your testbench here
2 // or browse Examples
3 module tb_mux4_1;
4   reg a,b,c,d;
5   reg [1:0] sel;
6   wire y;
7
8 mux4_1 uut (a,b,c,d,sel, y);
9
10 initial begin
11   $dumpfile("mux4_1.vcd");
12   $dumpvars(0,tb_mux4_1);
13
14   a=0; b=1;c=0;d=1;
15
16   sel=2'b00:#10;
17   sel=2'b01:#10;
18   sel=2'b10:#10;
19   sel=2'b11:#10;
20
21   $finish;
22 end
23
24 initial begin
25   $monitor("Time=%0t | a=%b b=%b c=%b
26 d=%b sel=%b => y=%b", $time,a,b,c,d, sel,
y);
27 end
28 endmodule
```

```
1 // Code your design here
2 module mux4_1(input a,input b,input c,
3   input d,input [1:0] sel, output y);
4   assign y = (sel==2'b00) ? a :
5             (sel==2'b01) ? b :
6             (sel==2'b10) ? c :
7             d;
8 endmodule
```

## Output:

Time=0 | a=0 b=1 c=0 d=1 sel=00 => y=0

Time=10 | a=0 b=1 c=0 d=1 sel=01 => y=1

Time=20 | a=0 b=1 c=0 d=1 sel=10 => y=0

Time=30 | a=0 b=1 c=0 d=1 sel=11 => y=1

# Code for 8:1 multiplexer

```
1 // Code your testbench here
2 // or browse Examples
3 module test_mux8to1;
4   reg [7:0] a;
5   reg [2:0] sel;
6   wire y;
7
8   mux8to1 m1(y,a,sel);
9
10 initial begin
11   $dumpfile("mux8to1.vcd");
12   $dumpvars(0,test_mux8to1);
13
14   a = 8'b10101010;
15
16 initial begin
17   a = 8'b10101010;
18   sel = 3'b000;
19   $monitor("Time=%0t | a=%b sel=%b =>
y=%b", $time, a, sel, y);
20 end
21
22   sel=3'b000; #10;
23   sel=3'b001; #10;
24   sel=3'b010; #10;
25   sel=3'b011; #10;
26   sel=3'b100; #10;
27   sel=3'b101; #10;
28   sel=3'b110; #10;
29   sel=3'b111; #10;
30
31   $finish;
32 end
33 endmodule
```

```
1 // Code your design here
2 module mux8to1(y, a, sel);
3   output y;
4   input [7:0] a;
5   input [2:0] sel;
6   assign y = sel == 3'b000 ? a[0] :
7     sel == 3'b001 ? a[1] :
8     sel == 3'b010 ? a[2] :
9     sel == 3'b011 ? a[3] :
10    sel == 3'b100 ? a[4] :
11    sel == 3'b101 ? a[5] :
12    sel == 3'b110 ? a[6] :
13    a[7];
14 endmodule
```

## Output:

Time=0 | a=10101010 sel=000 => y=0

Time=10 | a=10101010 sel=001 => y=1

Time=20 | a=10101010 sel=010 => y=0

Time=30 | a=10101010 sel=011 => y=1

Time=40 | a=10101010 sel=100 => y=0

Time=50 | a=10101010 sel=101 => y=1

Time=60 | a=10101010 sel=110 => y=0

Time=70 | a=10101010 sel=111 => y=1