

01

(001-365) Wk 01

JANUARY

WEDNESDAY

JANUARY - 2020

M	T	W	T	F	S	S	S	M	T	W	T	F	S	S
1	2	3	4	5	6	7	8	9	10	11	12			
13	14	15	16	17	18	19	20	21	22	23	24	25	26	
27	28	29	30	31										

## Modelling

- <sup>structural</sup>  
i) Behavioral      ii) Dataflow      iii) Gate level.      iv) Switch lvl.

### <sup>10</sup> Behavioral

This is the highest lvl of abstraction provided by Verilog.

<sup>11</sup> A module can be written in terms of the desired design algorithm, without concern for the hardware implementation detail.

### <sup>1</sup> Dataflow.

<sup>2</sup> Module is designed by specifying dataflow. The knowledge of Boolean expression & flow of data bet<sup>n</sup> hardware register & gates

### Gate lvl

<sup>4</sup> module is implemented in terms of logic gates & interconnections bet<sup>n</sup> gates.

<sup>5</sup>

### Switch lvl

<sup>6</sup> Lowest lvl of abstraction. Here module is implemented in terms of switches, storage nodes & connect<sup>n</sup> bet<sup>n</sup> them.

ED

There is always a chance to change, but do you take the time off to change.

2020  
S S  
11 12  
25 26

FEBRUARY - 2020  
M T W T F S S M T W T F S S  
1 2 3 4 5 6 7 8 9  
10 11 12 13 14 15 16 17 18 19 20 21 22 23  
24 25 26 27 28 29

MM

~ - not

~ - XOR

(002-364) Wk 01

JANUARY  
THURSDAY

02

## # Dataflow modeling

```
9 module halfsub(d, bw, a, b);  
10    output d, bw;  
11    input a, b;  
12    assign d = a ^ b;  
13    assign bw = (~a) & b;  
14    endmodule
```

```
1 module fullsub(d, bw, a, b, c);  
2    output d, bw;  
3    input a, b, c;  
4    assign d = a ^ b ^ c;  
5    assign bw = (~a & b) | (~a & c) | (b & c);  
6    endmodule
```

FEB

MAR

APR

MAY

- a penny, to speak true and sweet words

03

(003-363) Wk 01

JANUARY

FRIDAY

JANUARY - 2020													
M	T	W	T	F	S	S	S	M	T	W	T	F	S
1	2	3	4	5	6	7	8	9	10	11	12		
13	14	15	16	17	18	19	20	21	22	23	24	25	26
27	28	29	30	31									

## 9 Lexical conventions.

① White space

10 Tab, Blank spaces, NewLine  $\rightarrow$  ignored except in strings.

② Comment

// one line commenting  
/\* multiple lines. \*/

## 1 Unsized Numbers

12345  $\rightarrow$  Base  $\rightarrow$  decimal (default)

32-bit M/C specific.

16'b 0000\_1101\_0010\_0110

underbar is allowed.

## 5 Strings

"Hello world"

## 6 Nets:

7 Wire default 'z'

wand

wor

tri

trireg  $\rightarrow$  default 'x'

triand

trior.

Loneliness comes when one forgets that God is the supreme companion

2020  
S S  
11 12  
25 26

FEBRUARY - 2020

M	T	W	T	F	S	S	M	T	W	T	F	S	S
1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28
29													

(004-362) Wk 01

JANUARY  
SATURDAY

04

<sup>9</sup> Nets represent connections betn hardware elements. They have values continuously driven on them by the o/p's of devices  
<sup>10</sup> they are connected to.

<sup>11</sup> Registers :

They represent data storage elements. They retain a value until another value is driven placed onto them

Default - 'x'

<sup>1</sup> Vectors

<sup>2</sup> wire a;  
<sup>3</sup> wire [7:0] b;  
<sup>3</sup> reg [3:0] c;  
<sup>4</sup> reg [0:15] adder;  
<sup>4</sup> reg [3:0] x;  
<sup>5</sup> reg [0:3] y

<sup>6</sup> x = b [3:0]

<sup>6</sup> y = b [0:3]

<sup>7</sup>, x = b [0:3] // illegal since defined as [3:0].

Variable vector

Sunday 05

Part select

reg [0: 255] data1;  
reg [255: 0] data2;  
reg [7: 0] byte;

byte = data1(24 + :8)

Imagination is like a ferocious lion, do not allow to run it wild

FEB

MAR

APR

MAY

06

(006-360) Wk 02

JANUARY

MONDAY

JANUARY - 2020												
M	T	W	T	F	S	S	M	T	W	T	F	S
1	2	3	4	5	6	7	8	9	10	11	12	
13	14	15	16	17	18	19	20	21	22	23	24	25
26	27	28	29	30	31	-	-	-	-	-	-	-

byte = data[31 - :8]

31-8 places

i.e. = data[31:24]

byter2 = data[24 + :8]

11

Module mux(out, in, s)

output out;

input [3:0] in;

input [1:0] s;

```

2 assign out = in[3] & (~s[0] & ~s[1]) |
3   in[2] & (~s[0] & ~s[1]) |
4   in[1] & (~s[0] & s[1]) |
5   in[0] & (s[0] & s[1])

```

5

Integer, Real &amp; time data types.

```

7 integer i
    begin
        i=2;
        i=-3; etc.
    end

```

```

    real delta
    delta = 2.63;

```

```

    Time sim_time
    initial
    sim_time = $time

```

begin - end for multiple iteration.

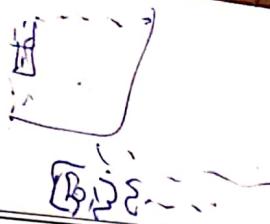
\$time  
predef  
fn for  
finding time.

END

To be child of God means to display his qualities

Y - 2020  
F S S  
10 11 12  
24 25 26

FEBRUARY - 2020  
M T W T F S S M T W T F S S  
1 2 3 4 5 6 7 8 9  
10 11 12 13 14 15 16 17 18 19 20 21 22 23  
24 25 26 27 28 29



(007-359) Wk 02

JANUARY  
TUESDAY

07

9 Array

reg a[0:7]

a => array of 8 elements each elements  
of width 1.

10

reg [3:0] b[0:7]

b => array of 8 elements each element  
of width 4.

#not synthesizable

12 integer array1[7:0](3:0);

; 1 column 8 rows

1

Wire [1:0] c[15:0];

→ 16 elements of 2 bits each

2

Memories

3 reg membit[0:1023]

4 Parameters

Const. can be defined in a module by the keyword parameter.  
5 they cannot be used as variable

6 parameter size = 1023

parameter width = 7

7 reg membit [0:size]

reg [width:0] mem [0:size]

localparam c = 15;

Your specialty influences others, so use it the best way you can

FEB

MAR

APR

MAY

08

(008-358) Wk 02

JANUARY

WEDNESDAY

JANUARY - 2020													
M	T	W	T	F	S	S	S	M	T	W	T	F	S
1	2	3	4	5	6	7	8	9	10	11	12		
13	14	15	16	17	18	19	20	21	22	23	24	25	26
27	28	29	30	31									

9 module parity(~~in~~ p, op, data)  
10 output p, op;  
11 input [7:0] data;

12 assign p = d[7] ^ d[6] ^ d[5] ^ d[4] ^ d[3] ^ d[2] ^ d[1] ^ d[0];  
op = ~(~~p~~ d[7] ^ d[6] ^ d[5] . . . . . ^ d[0]);

1 Strings

2 Strings can be stored in reg

3 reg [8\*18:1] string\_a;  
4 string\_a = "Hello Every Body";

5 Display

\$display("Hello")  
6 \$display(\$time)

7  
8 reg [0:15]addr;  
\$display("At time %d the addr is %h", \$time, addr);

decimal - d

Binary - b

ASCII - c

%m - module name.

%e ⇒ real no. in scientific 4e2.

%f ⇒ ~~real~~ you are special no one can play your role better than you.  
decimal

FEBRUARY - 2020													
M	T	W	T	F	S	S	S	M	T	W	T	F	S
10	11	12	13	14	15								
24	25	26	27	28	29								

9 reg  
initial  
begin  
10 \$monitor  
end

11 0

12 5

1  
2 #10  
#9

3 Ex. 3.5

4 a) write

5 b) 16'h xxx

c) 4 bit

d) unsized 'h

e)

JANUARY - 2020  
W T F S S S  
8 9 10 11 12  
22 23 24 25 26

FEBRUARY - 2020  
M T W T F S S M T W T F S S  
1 2 3 4 5 6 7 8 9  
10 11 12 13 14 15 16 17 18 19 20 21 22 23  
24 25 26 27 28 29

123

LL (009-357) Wk 02

JANUARY

THURSDAY

0111 1011

09

reg a;

initial

begin

10 \$monitor (\$time, "value of a is %b", a);  
end

11

0 value of a is 0

12 5 value of a is 5

1 value of a is 1

2 #100 \$stop

= wait for 100ns

#900 \$finish

Ex. 3.5

4 a) write 123 in binary.

8' b0111\_1011

5

b)

6 16'h xxxx

c) 4 bit -ve 2.

-4'd2

d) unsized hex no. 1234

'h 1234

Vision without action is a daydream. Action without vision is nightmare.

10

(010-356) Wk 02

JANUARY

FRIDAY

11

0/00%

JANUARY - 2020

M	T	W	T	F	S	S	M	T	W	T	F	S
1	2	3	4	5	6	7	8	9	10	11	12	
13	14	15	16	17	18	19	20	21	22	23	24	25
26	27	28	29	30	31							

2. Are following Right or not

9 a) "This is a string displaying the % sign". Wrong  
0%0%

10 b) "out = in1 + in2" Correct

11 c) "Please give a bell \007" Correct  
12 → \007 is octal code for character. [ \007 - • ]

1 d) "This is a backslash \ character \n". Wrong  
→ " " " " " "

2 "This is a \ character.

3 "This is a \*

4 c) "This is a asterick \* character" Correct

5 f) " " " " hash \# " " " Normal

6 4. Declare foll. variables.

a) 8-bit int called a-in

wire [7:0] a-in;

b) 32-bit reg. called address, & set to 32 bit decimal 3.

reg [31:0] address = 32'd3;

c) integer called count

integer count;

Be advised that all flatterers live at the expense of those who listen to them.

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12		
13	14	15	16	17	18	19
20	21	22	23	24	25	26

FEBRUARY - 2020

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

(011-355) Wk 02

JANUARY  
SATURDAY

11

FEB MAR APR MAY

d) time variable snap\_shot  
→ time snap\_shot

10 e) array called delays . Array contains 20 elements of type integer.  
→ integer array [19:0] delays

f) A memory MEM containing 256 words of 64 bits each  
12 reg [63:0] MEM [255:0]

1 g) parameter cache\_size = 512.

5.a) latch = 4'd12 // latch should integer.  
3

4 The current value of latch = ana 1100

5 b) In register value = 2

6 2:4 decoder.

```
7 module decoder (out, in);
  input [1:0] in;
  output [3:0] out;
  assign out[0] = (~in[0]) & (~in[1]);
  out[1] = (~in[0]) & (in[1]);
  out[2] = (in[0]) & (~in[1]);
  out[3] = (in[0]) & (in[1]);
```

Sunday 12

He who has an imagination without learning, has wings but no feet.

# 13

(013-353) Wk 03

JANUARY  
MONDAY

JANUARY - 2020											
M	T	W	T	F	S	S	S	M	T	W	T
				1	2	3	4	5	6	7	8
								9	10	11	12
								13	14	15	16
								17	18	19	20
								21	22	23	24
								25	26		
								27	28	29	30
								31			

9 module encoder(out, in)  
10 input [3:0] in;  
11 output [1:0] out;  
12 assign

1

2

### 3 Modules

4 module name  
5 port list, port declarations, parameters

6 dataflow  
7 → assign  
8 Behavioral  
9 → initial  
10 → always

11 Gate level  
12 → instantiation  
13 Decln of  
14 wire reg.

15 Tasks & Functions

16 end-module

A person's true wealth is the good he or she does in the world.

17 end  
18 end m

JANUARY - 2020  
 W T F S S  
 8 9 10 11 12  
 22 23 24 25 26

FEBRUARY - 2020  
 M T W T F S S M T W T F S S  
 1 2 3 4 5 6 7 8 9  
 10 11 12 13 14 15 16 17 18 19 20 21 22 23  
 24 25 26 27 28 29

(014-352) Wk 03

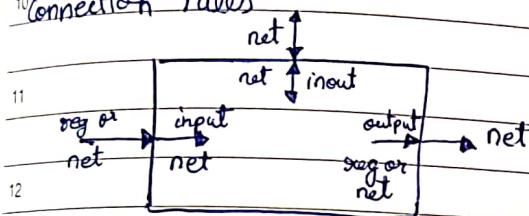
JANUARY  
 TUESDAY

14

## PORTS

input / output / inout

connection rules



Inside module:  
 inputs  $\Rightarrow$  wire  
 outputs  $\Rightarrow$  reg

In testbench:  
 inputs  $\stackrel{\text{from model}}{\Rightarrow}$  reg  
 output  $\Rightarrow$  wire

## Testbench for Full Adder

```

module testadd ;
//a, b, c, sum, carry
reg a, b, c;
wire sum, carry;
full adder f1(sum, carry, a, b, c);
initial
begin
    a <= 1'b0; b <= 1'b0; c <= 1'b0;
    #5 a <= 1'b0; b <= 1'b0; c <= 1'b1;
    :
    #5 a <= 1'b1; b <= 1'b1; c <= 1'b1;
end
end module

```

That which he selfishly keeps, he loses.

FEB  
 MAR  
 APR  
 MAY

15

(015-351) Wk 03

JANUARY

WEDNESDAY

JANUARY - 2020											
M	T	W	T	F	S	S	M	T	W	T	F
1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31					

9 testbench for mux

module tmux

10 reg [3:0] in;

reg [1:0] s;

11 wire out;

mux M1(out, in, s)

12 initial

begin

1 in[3] <= 1'b0 ; in[2] <= 1'b1 ; in[1] <= 1'b0 ; in[0] <= 1'b1;

2 ~~#\$~~ s[1] <= 1'b0 ; s[0] <= 1'b0 ;

#\$ s[1] <= 1'b0 ; s[0] <= 1'b1 ;

#\$ s[1] <= 1'b1 ; s[0] <= 1'b0 ;

#\$ s[1] <= 1'b1 ; s[0] <= 1'b1 ;

end

4 end module

5 Gate level modelling

6 Verilog supports basic logic gates as predefined primitives.

These primitives can be instantiated like modules. There are

7 two classes of basic gates. One is and, or & Buf/Not.

AND/OR

AND

: NOR

NAND

NOR

XOR

XNOR

8 out and ai(out, i1, i2);

We cannot forever hide the truth about overselves, from ourselves.

ARY - 2020  
T F S S  
9 10 11 12  
23 24 25 26

FEBRUARY - 2020  
M T W T F S S M T W T F S S  
1 2 3 4 5 6 7 8 9  
10 11 12 13 14 15 16 17 18 19 20 21 22 23  
24 25 26 27 28 29

(016-350) Wk 03

JANUARY  
THURSDAY

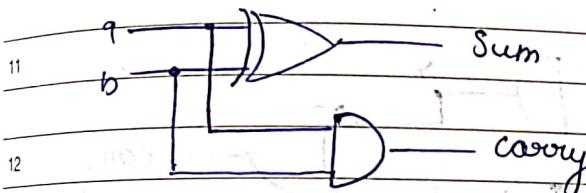
16

i1 → out  
i2 →

nor n1 (out, i1, i2);

no. of 1/p's can  
be extended

Code for Half adder



1 module halfadd (sum, carry, a, b);  
2     output sum, carry;  
3     input a, b;  
4     reg sum, carry;  
5     wire a, b;  
6     xor x1 (sum, a, b);  
7     and a1 (carry, a, b);  
8     end module

same testbench.

FEB  
MAR  
APR  
MAY

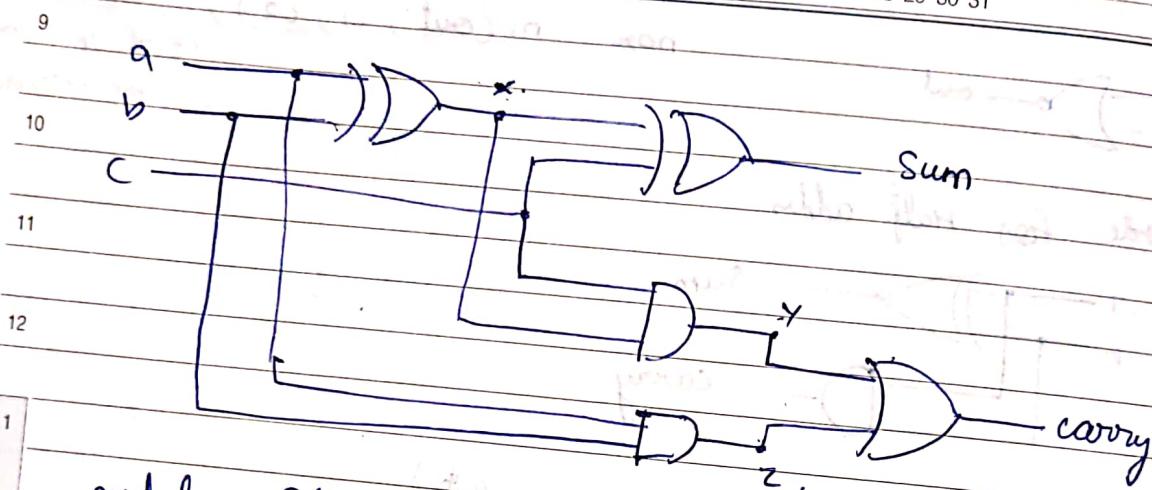
I believe that the first test of a truly great man is his humility.

17

(017-349) Wk 03

JANUARY  
FRIDAY

JANUARY - 2020						
M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12		
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		



```
1 module full add (sum, carry, a, b, c);
2   output sum, carry;
3   input a, b, c;
4   xreg z sum, carry;
5   wire a, b, c, x, y, z;
6   XOR x1 (x, a, b);
7   XOR x2 (sum, x, c);
8   AND A1 (y, x, c);
9   AND A2 (z, a, b);
10  OR O1 (carry, y, z);
11 end module.
```

JANUARY - 2020						
M	T	W	T	F	S	S
6	7	8	9	10	11	12
20	21	22	23	24	25	26

FEBRUARY - 2020						
M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

(018-348) Wk 03

JANUARY  
SATURDAY

18

## Gate Delays : Gate Level Modeling.

① Rise

② Fall

③ Turn-off Delay

q/p delay goes to Z from any value.

1 If one delay is specified then this value is used for all 3 transitions.  $\Sigma$

2 If two values are specified then they correspond to rise & fall. Minimum of two is considered for

turn off delay.

3 case 1

and  $\#(5)$  a1(out, in1, in2);

rise  
fall } 5

turn-off } 5

6 case 2

and  $\#(3, 4)$  a1(out, in1, in2); rise = 3

fall = 4

turn-off = 3.

Sunday 19

Case 3

and  $\#(3, 4, 5)$  a1(out, in1, in2);

rise = 3

fall = 4

turn off = 5

Toleration is the greatest gift of the mind



20

(020-346) Wk 04

JANUARY  
MONDAY

JANUARY - 2020												
M	T	W	T	F	S	S	S	M	T	W	F	S
1	2	3	4	5	6	7	8	9	10	11	12	
13	14	15	16	17	18	19	20	21	22	23	24	25
26	27	28	29	30	31							

- 9 +min delays
- at given time
- 10 designer can specify the min value of the delay that the design +min delays.
- 11
- 12

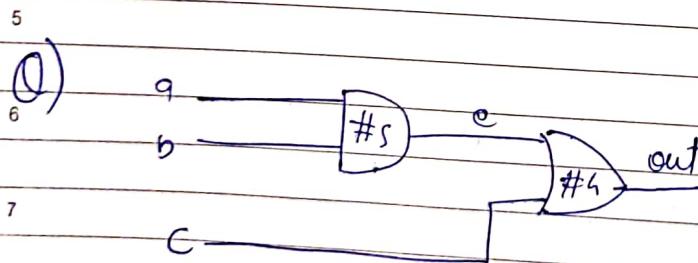
+type delays  
typical value can be specified  
+typdelays

+max delays.  
Maximum value can be specified as +max delay

1	rise time	fall time	turn-off Delay
2	min = 3	min = 4	min = 5
3	typ = 4	typ = 5	typ = 6
4	max = 5	max = 6	max = 7

min = 5  
typ = 6  
max = 7

and #( 3:4:5, 4:5:6, 5:6:7 ) a1 (out, in1, in2);



module D(out, a, b, c)

output D;

input a, b, c;

end

wire wire e;  
or d = (out) a, b, c;

One man gives freely, yet gains even more;

FEBRUARY  
M T W T  
10 11 12 13  
24 25 26 27

9  
10  
m  
11  
c  
12  
ir  
1  
b  
2 #10  
#10  
3 end m

4 q o  
5 b o  
6 c o  
7 e XXX  
out XXX

JANUARY - 2020  
T F S S  
9 10 11 12  
23 24 25 26

FEBRUARY - 2020  
M T W T F S S M T W T F S S  
1 2 3 4 5 6 7 8 9  
10 11 12 13 14 15 16 17 18 19 20 21 22 23  
24 25 26 27 28 29

(021-345) Wk 04

JANUARY

TUESDAY

21

AND ~~x1(e, a, b); # (5)~~ x1(e, a, b);  
OR ~~x1(out, e, c); # (4)~~ y1(out, e, c);  
endmodule;

10 module test

11 were out;

12 reg a, b, c;

13 o d1(out, a, b, c);

initial

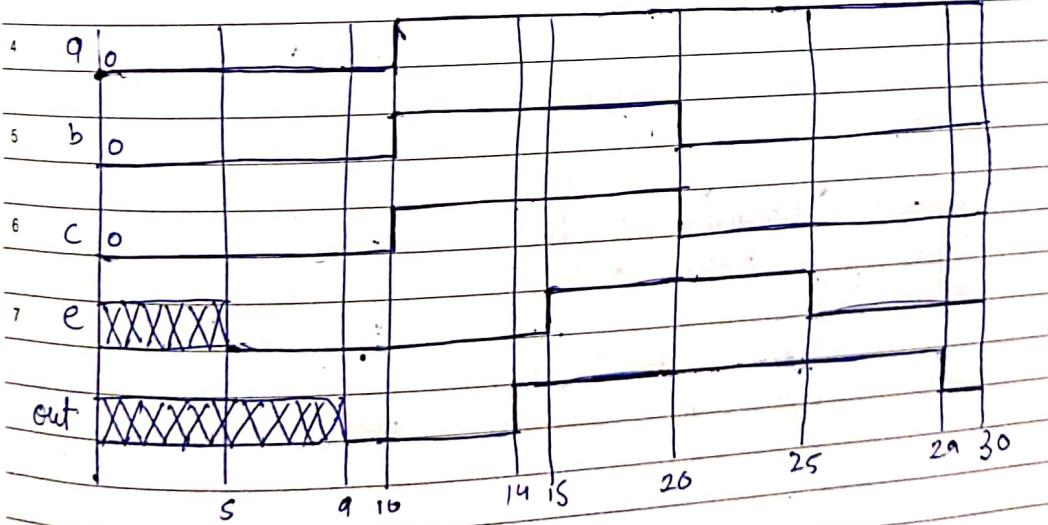
begin

14 a <= 1'b0 ; b <= 1'b0 ; c <= 1'b0 ;

15 #10 a <= 1'b1 ; b <= 1'b1 ; c <= 1'b1 ;

16 #10 a <= 1'b1 ; b <= 1'b0 ; c <= 1'b0 ;

17 endmodule



If your ship doesn't come in, swim out to it.

22

(022-344) Wk 04

JANUARY

WEDNESDAY

## Ripple Carry.

JANUARY - 2020													
M	T	W	T	F	S	S	S	M	T	W	T	F	S
	1	2	3	4	5	6	7	8	9	10	11	12	
	13	14	15	16	17	18	19	20	21	22	23	24	25
	26	27	28	29	30	31							

assume fulladder is written.  
fulladd fa1 (sum, carry, a, b, c);  
module RCA (s, cout, A, B, Cin);  
input [3:0] A, B;  
input Cin;  
output [3:0] s;  
output cout;  
wire c1, c2, c3;  
fulladd fa0 (s[0], c1, a[0], b[0], Cin);  
fulladd fa1 (s[1], c2, a[1], b[1], c1);  
fulladd fa2 (s[2], c3, a[2], b[2], c2);  
fulladd fa3 (s[3], cout, a[3], b[3], c3);  
endmodule;

assume JKFF is written.  
JKFF J1 C

module Counter (Q, CLK);  
input [3:0] Q;  
input CLK;  
output [3:0] Q;  
wire J1, K1, J2, K2, J3, K3, J0, K0;  
wire Q1bar, Q2bar, Q3bar, Q4bar;

If there is no struggle, there is no progress.

F	S	S	M	T	W	T	F	S
3	4	5	6	7	8	9	10	11
17	18	19	20	21	22	23	24	25
26	1	2	3	4	5	6	7	8
11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28
29								

FEBRUARY - 2020								
M	T	W	T	F	S	S	M	T
1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18
19	20	21	22	23	24	25	26	27
28	29							

(023-343) Wk 04  
25eason

JANUARY  
THURSDAY

23

FEB  
MAR  
APR  
MAY

```

9 module Counter (@, clk);
10   input clk;
11   output [3:0] @;
12   wire [1:0]JK;
13   assign JK = 1'b1;
14   wire Qobar, Q1bar, Q2bar, Q3bar;
15   JKFF J0 (@[0], Qobar, JK, JK, clk);
16   JKFF J1 (@[1], Q1bar, JK, JK, clk);
17   JKFF J2 (@[2], Q2bar, JK, JK, clk);
18   JKFF J3 (@[3], Q3bar, JK, JK, clk);
19 endmodule.

```

```

3 module JKFF (@, Qbar, JK, clk);
4   output @ Qbar;
5   input [1:0] JK;
6   input clk;
7   always @ (negedge clk)
8     begin
9       case (JK)
10         2'b00 : temp <= temp;
11         2'b01 : temp <= 1'b0;
12         2'b10 : temp <= 1'b1;
13         2'b11 : temp <= ~temp;
14     end
15 endmodule;

```

Be content with your lot; one cannot be first in everything.

24

(024-342) Wk 04

JANUARY

FRIDAY

JANUARY - 2020												
M	T	W	T	F	S	S	S	M	T	W	F	S
1	2	3	4	5	6	7	8	9	10	11	12	
13	14	15	16	17	18	19	20	21	22	23	24	25
26	27	28	29	30	31							

9 Dataflow modeling.

10 Delays.

① Regular Assignment Delay

11 assign #10 out = i1 & i2;



BCD to

Gray conv. in Gate lvl.

Ex1. A=2 ; B=0;

A & B

evaluates to 0

A || B

evaluates to 1.

Ex2. A = 2'bx

A & B

B = 2'b10

evaluates to x.

A wise man knows how little he knows.

JANUARY - 2020  
 W T F S S  
 8 9 10 11 12  
 22 23 24 25 26

FEBRUARY - 2020												
M	T	W	T	F	S	S	M	T	W	T	F	S
1	2	3	4	5	6	7	8	9	10	11	12	13
14	15	16	17	18	19	20	21	22	23	24	25	26
27	28	29										

(025-341) Wk 04

JANUARY  
SATURDAY

25

9 ex. 3.  $a = 2$   
 $b = 3$

10  $(a = 2) \& \& (b = 3)$

evaluating to .

11

③ Related.

12  $>$   
 $<$   
 }  
 1  $\neq$   
 $\leq$   
 }  
 2

equally

3  $= =$  can equally  
 $\neq = \neq$

4  $A = 4$        $B = 3$

5  $X = 4'1b_1010$        $y = 4'1b1101$   
 $Z = 4'1b1xxz$ ,  $M = 4'1b1xxz$

$N = 4'1b1xx$

- 6 A  
 1)  $A == B$       '0'  
 2)  $X1 = Y$       '-'  
 3)  $X == Z$       'X'  
 4)  $Z == M$       '1'  
 5)  $Z == N$       '0'  
 6)  $M == N$       '1'

Sunday 26

Life is shaped by the people you meet every day.



27

(027-339) Wk 05

JANUARY

MONDAY

JANUARY - 2020											
M	T	W	T	F	S	S	S	M	T	W	F
1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31					

9 Bitwise.

& and 2

10 1 or 2

11 ^ xor 2

~ not 1

12 ~^ xor 2

1 Reduction

&

~ &

! or

3 ~!

^

4 ~^.

$$5 a = 4' b001$$

$$b = 4' b1111$$

$$6 a \& b = 0011$$

$$\& a = 0$$

$$\& b = 1$$

and all digits in aa.

JANUARY - 2020						
M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

FEBRUARY - 2020						
M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

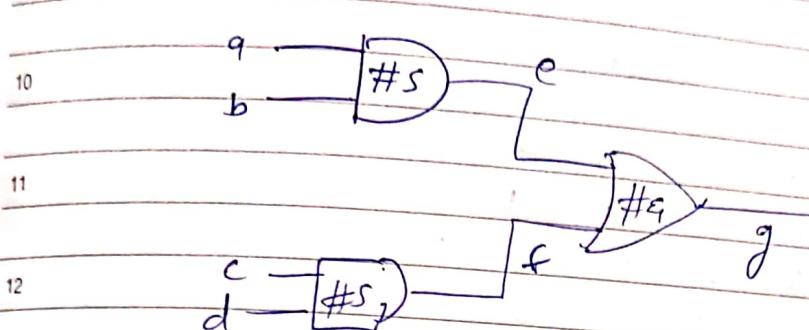
(028-338) Wk 05

JANUARY

TUESDAY

28

Using implicit assignment write code.



```

1 regular
2 module ckt(g,a,b,c,d)
3   input a,b,c,d;
4   output g;
5   wire e,f;
6   assign #5 e = a & b;
7   assign #5 f = c & d;
8   assign #4 g = e | f;
9 endmodule.

```

```

1 Implicit
2 module ckt(g,a,b,c,d);
3   input a,b,c,d;
4   output g;
5   wire #5 e = a & b;
6   wire #5 f = c & d;
7   assign #4 g = e | f;
8 endmodule.

```

Angels fly because they take themselves lightly.

FEB

MAR

APR

MAY

29

(029-337) Wk 05

JANUARY  
WEDNESDAY

JANUARY											
M	T	W	T	F	S	S	M	T	W	T	F
1	2	3	4	5	6	7	8	9	10		
13	14	15	16	17	18	19	20	21	22	23	24
27	28	29	30	31							

9 BCD to Gray.

BCD				Gray				
10	A	B	C	n	P	Q	R	S
0	0	0	0	0	0	0	0	0
11	0	0	0	1	0	0	0	1
12	0	0	1	0	0	0	1	1
13	0	0	1	1	0	0	1	0
14	0	1	0	0	0	1	1	0
15	0	1	0	1	0	1	1	1
16	0	1	1	0	0	1	0	1
17	0	1	1	1	0	1	0	1
18	1	0	0	0	1	1	0	0
19	1	0	0	1	1	0	0	0
20	1	0	1	1	1	1	0	1
21	1	0	1	1	0	1	0	1
22	1	1	0	0	1	0	1	1
23	1	1	0	1	0	1	0	1
24	1	1	1	1	0	1	0	1

4	5	6	7
AB	CD	AB	CD
00	00	X <sup>12</sup>	1
00	00	X <sup>13</sup>	1
00	00	X <sup>14</sup>	X <sup>15</sup>
00	00	X <sup>16</sup>	X <sup>17</sup>

$$P = A\bar{B}\bar{C}$$

7	8	9	10
AB	CD	AB	CD
00	00	01	11
00	01	11	(1)
01	00	11	(1)
01	01	11	1
00	10	10	1
00	11	10	1

Q = B + A\bar{B}\bar{C} . A+B.

Lack on money is no obstacle. Lack of an idea is an obstacle.

FEBRUARY - 2020						
M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

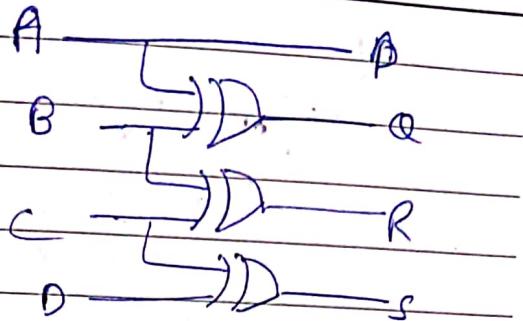
(030-336) Wk 05

JANUARY  
THURSDAY

30

$$S = C \oplus D$$

$$R = B \oplus C$$



module BCtoGray (P, Q, R, S, A, B, C, D)

output P, Q, R, S;

input A, B, C, D;

assign

buf b1(P, A);

or o1(Q, A, B);

xor x1(R, B, C);

xor x2(S, C, D);

endmodule

BCD to XS-3.

A	B	C	D	P	Q	R	S
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

The rule is, jam tomorrow and jam yesterday - but never jam today.

31

(031-335) Wk 05

JANUARY  
FRIDAY

P

	AB		CD			
9	00	00	01	X <sup>12</sup>	11	10
10	01	01	10	X <sup>13</sup>	11	11
11	10	01	11	X <sup>15</sup>	X <sup>11</sup>	
12	10	01	11	X <sup>14</sup>	X <sup>10</sup>	

JANUARY - 2020						
M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12		
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

$$P = A + BD + BC = A + B(D+C)$$

Q =

$$R = CO D$$

$$S = \bar{D}$$

	AB		CD			
1	00	00	01	11	10	
2	01	01	10	X <sup>13</sup>	11	00
3	10	11	01	X <sup>15</sup>	X <sup>11</sup>	X <sup>10</sup>
4	10	11	01	X <sup>14</sup>	X <sup>13</sup>	X <sup>13</sup>

$$\begin{aligned} Q &= \bar{A}D + \bar{A}\bar{B}D + \bar{A}\bar{B}C \\ &\quad + \bar{A}B\bar{C}\bar{D} \end{aligned}$$

$$\begin{aligned} Q &= \bar{A}D + \bar{A}\bar{B}(D+C) \\ &\quad + \bar{A}B\bar{C}\bar{D} \end{aligned}$$

$$\begin{aligned} Q &= \bar{B}D + \bar{A}\bar{B}C + \bar{A}B\bar{C}\bar{D} \\ &= \bar{B}(D+C) + B\bar{C}\bar{D} \\ &= B \oplus (C+D) \end{aligned}$$

The more you chase money, the harder it is to catch it.

01

(032-334) Wk 05

FEBRUARY

SATURDAY

FEBRUARY - 2023											
M	T	W	T	F	S	S	M	T	W	F	S
					1	2	3	4	5	6	7
10	11	12	13	14	15	16	17	18	19	20	21
24	25	26	27	28	29						

9      BCD to xs-3 .

10

11

12

1  
2

3

4

5

module

BCD to XS3 (P, Q, R, S, A, B, C, D)

output

P, Q, R, S;

input

A, B, C, D;

wire

e, f;

or

o1(e, c, d);

and

o1(f, b, e);

02 Sunday

o2 (P, A, f);

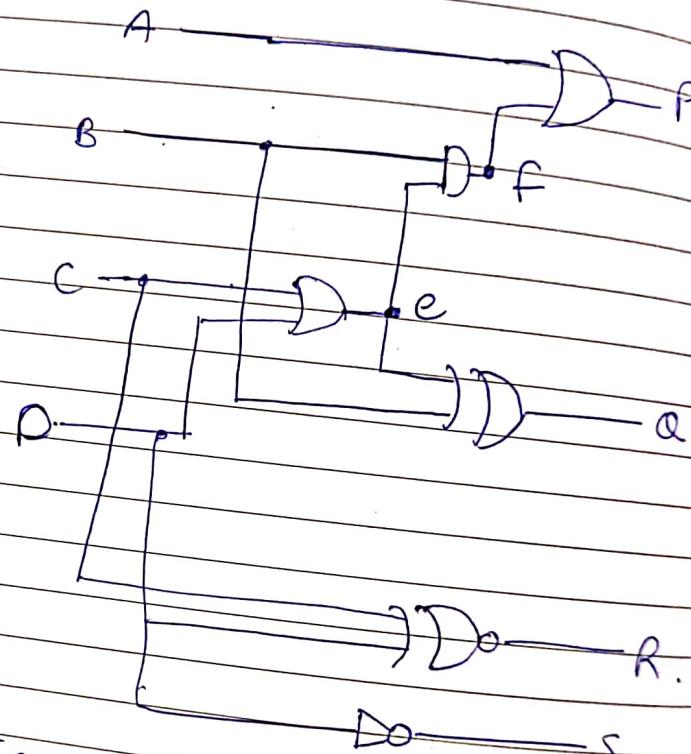
not

n1 (S, D);

xnor

xn1 (R, C, D);

end module.



Peace is dependant upon honesty and simplicity

FEBRUARY - 2020						
T	W	T	F	S	S	S
4	5	6	7	8	9	
8	19	20	21	22	23	

MARCH - 2020						
M	T	W	T	F	S	S
	1	2	3	4	5	6
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

(034-332) Wk 06

FEBRUARY

MONDAY

03

### 9 Concatenation

$$10 \quad \{ \}$$

$$11 \quad A = 1'b1$$

$$B = 1'b0$$

$$12 \quad C = \{A, B\}$$

$$C = 2'b10$$

$$1 \quad A = 1'b1;$$

$$2 \quad B = 1'b0;$$

$$3 \quad C = \{4\{A\}, 2\{B\}\}$$

$$4 \quad C = 6'b111100$$

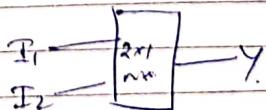
### Conditional operator

5 (condition)? true expr : false expr;

6 ( $a == 1'b1$ )?  $b = a$  :  $b = \sim b$ ;

7 2:1 mux using conditional opr.

assign  $y = s ? I_2 : I_1$ ;



MAR

APR

MAY

Ask not what your country can do for you. Ask what you can do for your country.

04

(035-331) Wk 06

FEBRUARY  
TUESDAY

FEBRUARY - 2020												
M	T	W	T	F	S	S	M	T	W	T	F	S
					1	2	3	4	5	6	7	8
10	11	12	13	14	15	16	17	18	19	20	21	22
24	25	26	27	28	29							

9 4: 1 mix.

10 assign  $y = S1 ? (S0 ? I3 : I2) : (S0 ? I1 : I0);$

11

~~write code for t-FF.~~

12

SR Latch,

1 module SRL(Q, S, R)  
2   inputs Q;  
3   input S, R;  
4   assign S ? (R ? i'bx : i'b1) : (R ? i'b0 : Q);  
5   endmodule.

4

Behavioural Modeling

5 module Example;

6 reg a, b, c, n, y;  
7 initial  
n = i'b0;  
initial  
begin  
a = i'b0  
#5 b = i'b1  
#10 c = i'b1  
end

initial  
begin  
#15 n = i'b1;  
#5 y = i'b1;  
end  
initial  
#30 \$finish  
endmodule

In the practice of tolerance, one's enemy is the best teacher.

FEBRUARY - 2020  
M T W T F S S  
5 6 7 8 9  
9 20 21 22 23

MARCH - 2020  
M T W T F S S  
1 2 3 4 5 6 7 8  
9 10 11 12 13 14 15 16 17 18 19 20 21 22  
23 24 25 26 27 28 29 30 31

(036-330) Wk 06  
FEBRUARY  
WEDNESDAY  
05

module Example

reg clock;

initial

clock = 1'b0;

case II reg clock = 1'b0;

case III module example(a, b, c);

input b, c;

output [7:0] a

begin example(a,

case IV

module example (output reg[7:0] a, input b, c,  
input [3:0] p, output reg x);

initial

x = 1'b0;

endmodule.

always

→ starts at time 0 and executes all statements inside  
the always block continuously in a looping manner.

module clockgen(output reg clock);

initial

clock = 1'b0;

always

#5 always = ~clock

endmodule.

If you're not sure where you're going, you'll probably end up nowhere.

MAR

APR

MAY

06

(037-329) Wk 06

FEBRUARY

THURSDAY

FEBRUARY - 2020											
M	T	W	T	F	S	S	M	T	W	T	F
1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24

```
9 module srff(output, q, qbar, input, clk, s, r);
10 reg q, qbar;
11 reg temp; = 1'b0;
12 always @ (posedge clk)
13 begin
14 case ({s, r})
15 2'b00: temp <- temp;
16 2'b01: temp <- 1'b0;
17 2'b10: temp <- 1'b1;
18 2'b11: temp <- 1'bx;
19 end case
20 q <- temp;
21 qbar <- ~temp;
22 end;
23 endmodule;
```

```
5 module Tff(q, qbar, t, clk)
6   output reg q, qbar;
7   input t, clk;
8   reg temp;
9   initial
10 begin
11   temp <- 1'b0
12   always @ (negedge clk)
13   begin
14     case (t)
15       1'b0: temp <- temp;
16       1'b1: temp <- ~temp;
17     endcase
18   end
19 endmodule
```

Our task now is not to fix the blame for the past, but to fix the course for the future.

JANUARY  
W T F S S S  
1 2 3 4 5 6 7 8  
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

MARCH - 2020  
M T W T F S S M T W T F S S  
1 2 3 4 5 6 7 8  
9 10 11 12 13 14 15 16 17 18 19 20 21 22  
23 24 25 26 27 28 29 30 31

(038-328) Wk 06

FEBRUARY

FRIDAY

07

MAR APR MAY

9  $q \leftarrow \text{temp};$   
10  $qbui \leftarrow \sim \text{temp}$   
11  $\text{end}$   
12  $\text{chdmodul}$

## Blocking and Non-blocking Assignments

```
reg [15:0] a, b;  
reg x, y, z;  
reg clock;  
initial .  
begin  
 3  $x = 1'b0;$   $y = 1'b1;$   $z = 1'b1;$   
 4  $a = 16'b0;$   
#10  $a[12] = 1'b1;$  → at time 10  
#15  $b[15:13] = \{x, y, z\};$  → at time 25  
 6  $clock = 1'b1;$  → at time 25  
 7 end
```

= blocks next statement till it is executed, i.e. value is assigned  
→ order matters

← non-blocking : any order  
another way of delay instr.  
 $a[12] = #10 1'b1;$   
 $b[15:13] = #15 \{x, y, z\},$

Luck usually comes to those who are too busy to be looking for it.

08

(039-327) Wk 06

FEBRUARY

SATURDAY

FEBRUARY - 2020											
M	T	W	T	F	S	S	S	M	T	W	F
					1	2	3	4	5	6	7
					8	9	10	11	12	13	14
					15	16	17	18	19	20	21
					22	23	24	25	26	27	28
					29						

9 reg a, b;  
10 always @ (posedge clk);  
10 a = b;  
11 always @ (posedge clk);  
11 b = a;

Race conditions.

Not appropriate

1 reg a, b  
2 always @ (posedge clk);  
2 a <= b;  
3 always @ (posedge clk);  
3 b <= a;

} appropriate

4 Parallel Block & sequential blocks.

5 begin - end block is sequential.

6 reg n, y, z;  
reg [1:0] a;  
initial

7 09 Sunday begin  
n = 1'b0;  
y = 1'b1;  
z = 1'b0;

#10  
#15 a = {x, y};

end

} sequential block assignment.

time	Ex
0	x = 0 y = 1
10.	z = 0
25	a = 01

Discontent is the first step in the progress of man or a nation.

FEBRUARY - 2020  
 W T F S S  
 5 6 7 8 9  
 19 20 21 22 23

MARCH - 2020  
 M T W T F S S M T W T F S S  
 1 2 3 4 5 6 7 8  
 9 10 11 12 13 14 15 16 17 18 19 20 21 22  
 23 24 25 26 27 28 29 30 31

(041-325) Wk 07

FEBRUARY

MONDAY

10

9 neg  $n, y, \tau$   
 neg  $[1:0]\alpha$

10 initial

begin fork

11  $n = 1'b0$ ;

$y = 1'b1$ ;

12  $\tau = 1'b0$ ;

#10  $z = 1'b0$ ;

1 #15  $\alpha = \{n, y\}$ ;

join



parallel block.

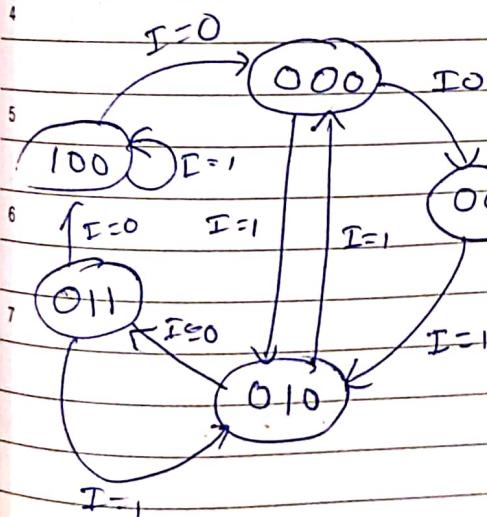
t/m

0  $n=0$   $y=1$

10  $z=0$

15  $\alpha=01$

3 Fork-join is parallel.



State	O/P
000	000
001	111
010	010
011	101
100	100

What we see depends mainly on what we look for.

10

MAR

APR

MAY

11

(042-324) Wk 07

FEBRUARY

TUESDAY

FEBRUARY - 2020											
M	T	W	T	F	S	S	M	T	W	T	F
				1	2	3	4	5	6	7	8
10	11	12	13	14	15	16	17	18	19	20	21
24	25	26	27	28	29						

9 # for state diag. behind

```
10 module FSM(I, clk, Y, rst);
11   output reg [2:0] y;
12   input I, clk, rst;
13   parameter S0 = 3'b000;
14   parameter S1 = 3'b001;
15   parameter S2 = 3'b010;
16   parameter S3 = 3'b011;
17   parameter S4 = 3'b100;
18   reg [2:0] ps, ns;
19   initial ps = S0;
20   always @(posedge clk);
21     begin
22       if (rst == 1'b1)
23         ns = S0;
24       else
25         case(ps)
26           S0: if (I == 1'b0)
27             ns <= S1; else
28             ns <= S2;
29           S1: if (I == 1'b0)
30             ns <= S1; else
31             ns <= S2;
32           S2: if (I == 1'b0)
33             ns <= S3; else
34             ns <= S0;
```

Try not to become a man of success but rather to become a man of value.

FEBRUARY - 2020						
F	S	S	M	T	W	F
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29		30	31			

MARCH - 2020						
M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

(043-323) Wk 07

FEBRUARY  
WEDNESDAY

12

```

9  S3: if (I# == 1'b0)
10   ns <= S4; else
11   ns <= S2;
12   S4: if (I == 1'b0)
13     ns <= S0; else
14     ns <= S4;
15   end case
16   PS <= NS;
17   end case
18
19  always @ (posedge clk)
20    begin
21      if (rst == 1'b1)
22        Y <= 3'b000;
23      else
24        case (PS)
25          S0: Y <= 3'b000;
26          S1: Y <= 3'b111;
27          S2: Y <= 3'b010;
28          S3: Y <= 3'b101;
29          S4: Y <= 3'b100;
30        end case
31    end
  
```

The time to repair the roof is when the sun is shining.

of value.