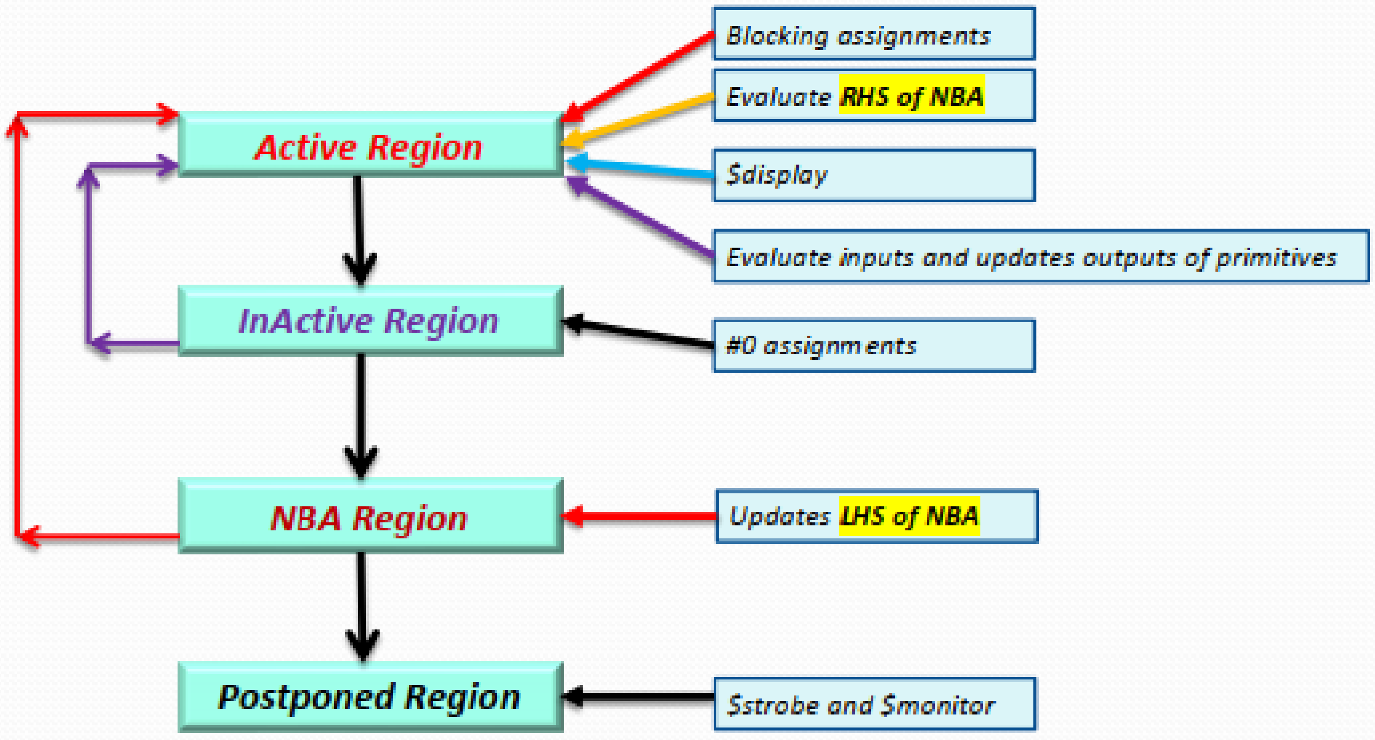
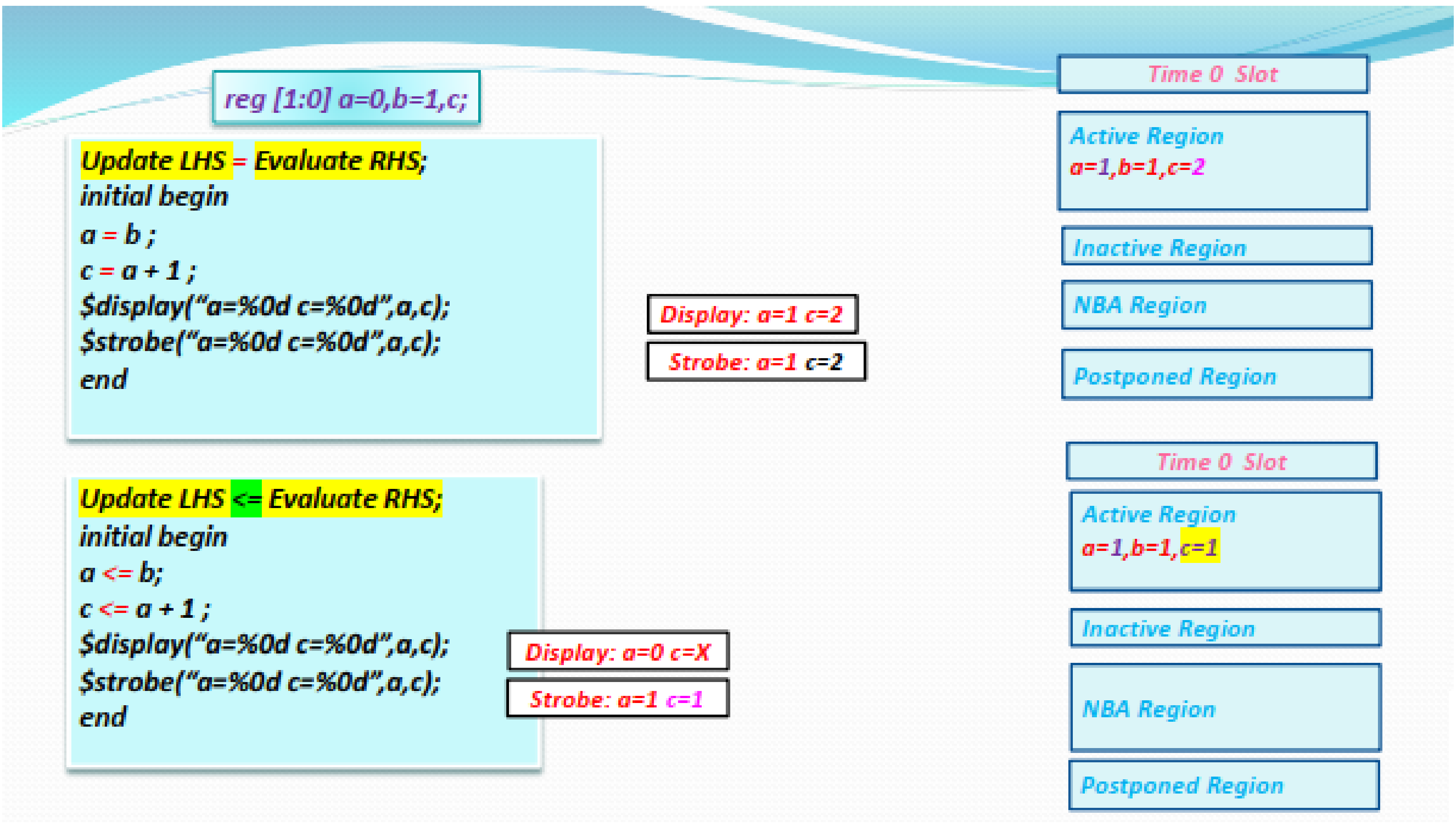
Verilog Stratified Event Queue

Stratified Event Queue





reg [1:0] a=0,b=1,c;

```
Update LHS = Evaluate RHS;
initial begin
a = b ;
c = a + 1 ;
$display("a=%0d c=%0d",a,c);
$strobe("a=%0d c=%0d",a,c);
end
```

Display: a=1 c=2

Strobe: a=1 c=2

Time 0 Slot

Active Region
a=1,b=1,c=2

Inactive Region

NBA Region

Postponed Region

```
Update LHS <= Evaluate RHS;
initial begin
a <= b;
c <= a + 1 ;
$display("a=%0d c=%0d",a,c);
$strobe("a=%0d c=%0d",a,c);
end
```

Display: a=0 c=X

Strobe: a=1 c=1

Time 0 Slot

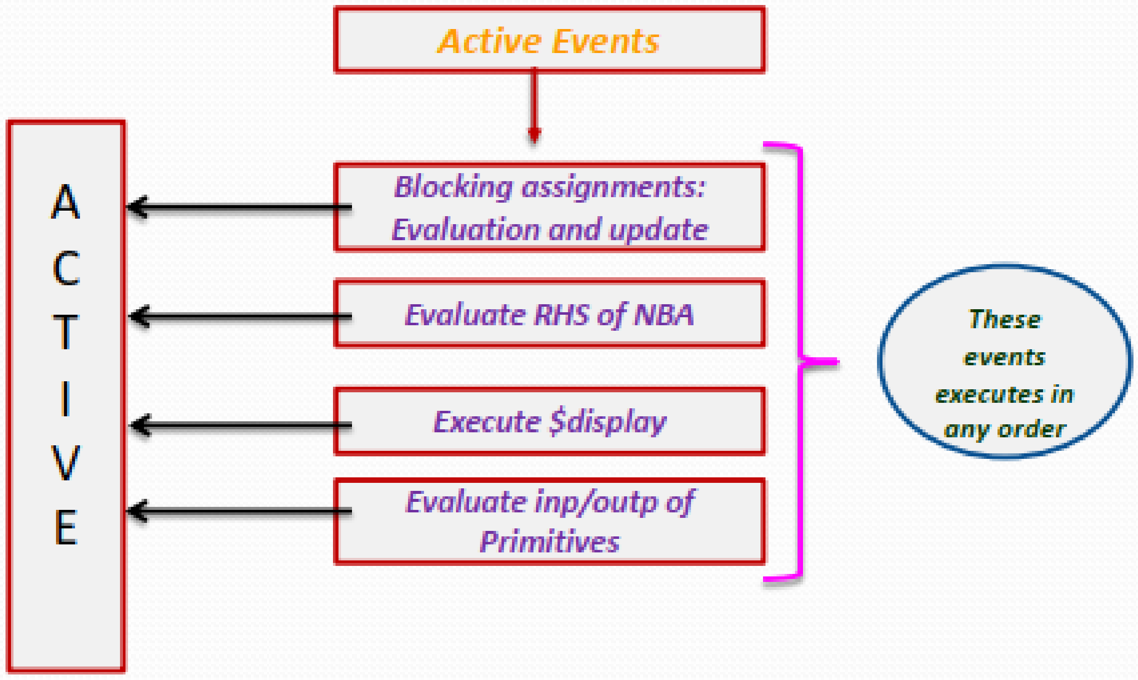
Active Region
a=1,b=1,c=1

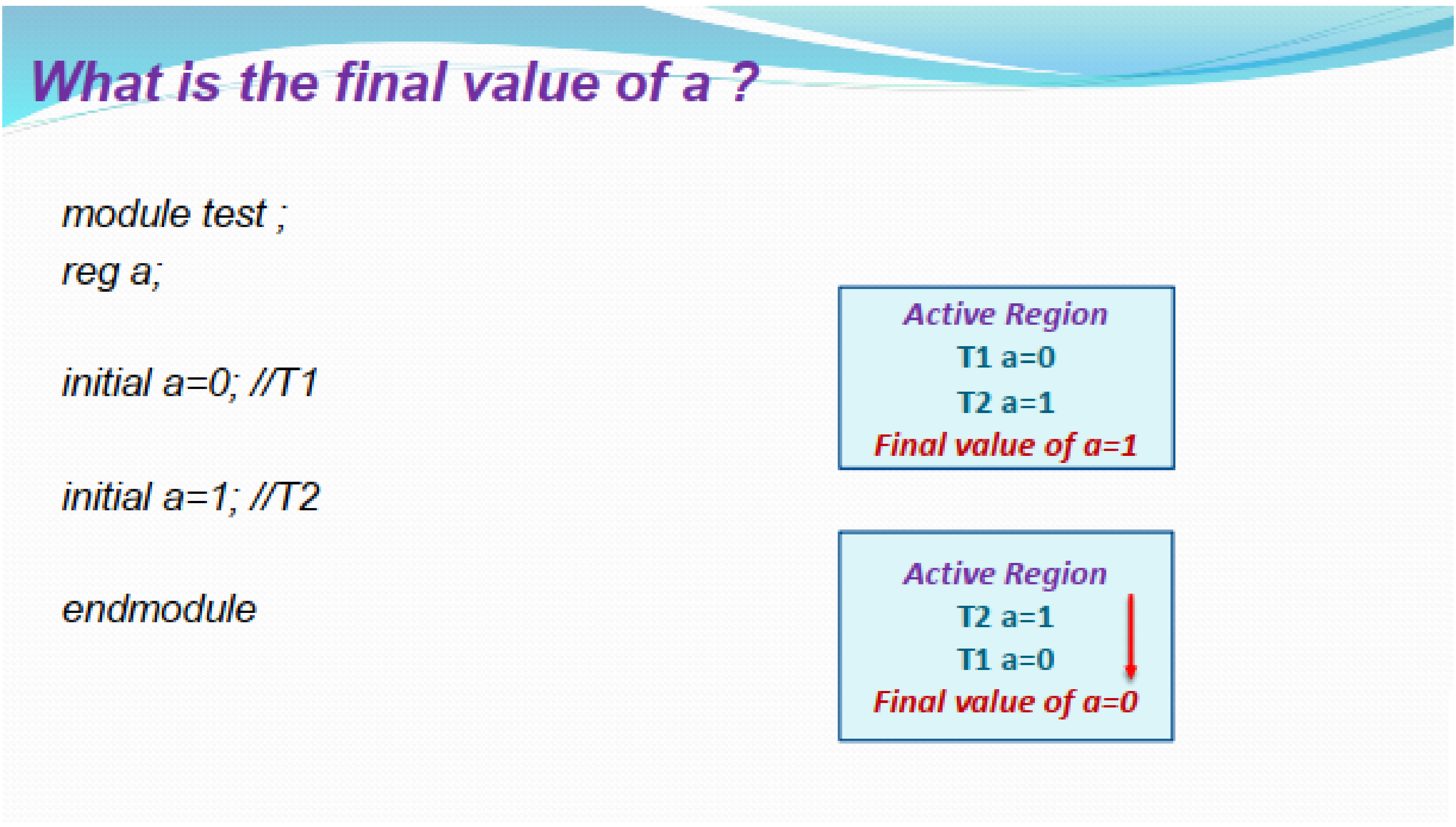
Inactive Region

NBA Region

Postponed Region

Active Event Queue


Active Events
A C T I V E
Blocking assignments: Evaluation and update
Evaluate RHS of NBA
Execute $display
Evaluate inp/outp of Primitives
These events executes in any order

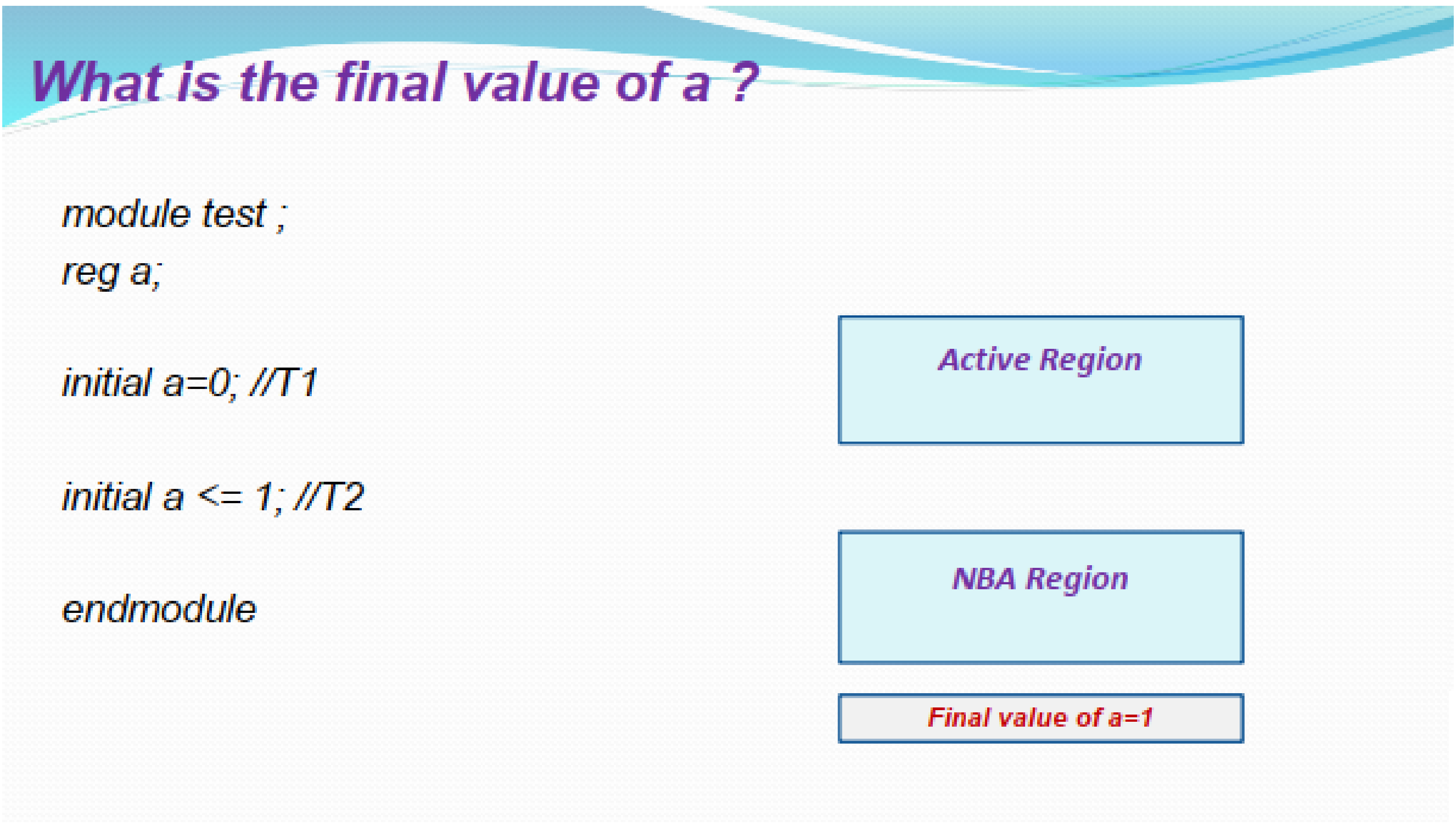


What is the final value of a ?

```
module test ;
reg a;

initial a=0; //T1

initial a=1; //T2

endmodule
```

Active Region
T1 a=0
T2 a=1
Final value of a=1

Active Region
T2 a=1
T1 a=0
Final value of a=0



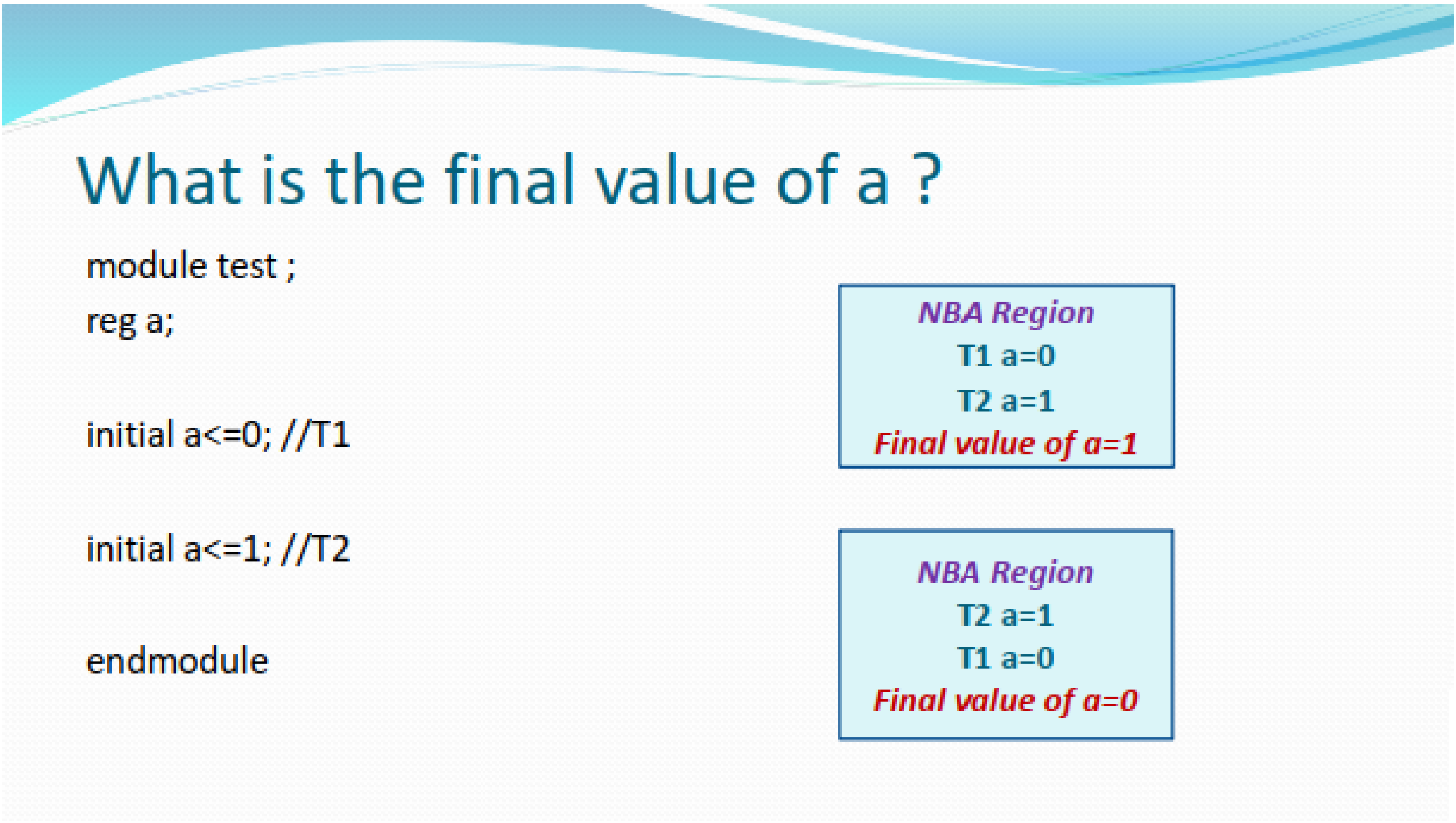
What is the final value of a ?

```
module test ;
reg a;

initial a=0; //T1

initial a <= 1; //T2

endmodule
```

Active Region

NBA Region

Final value of a=1

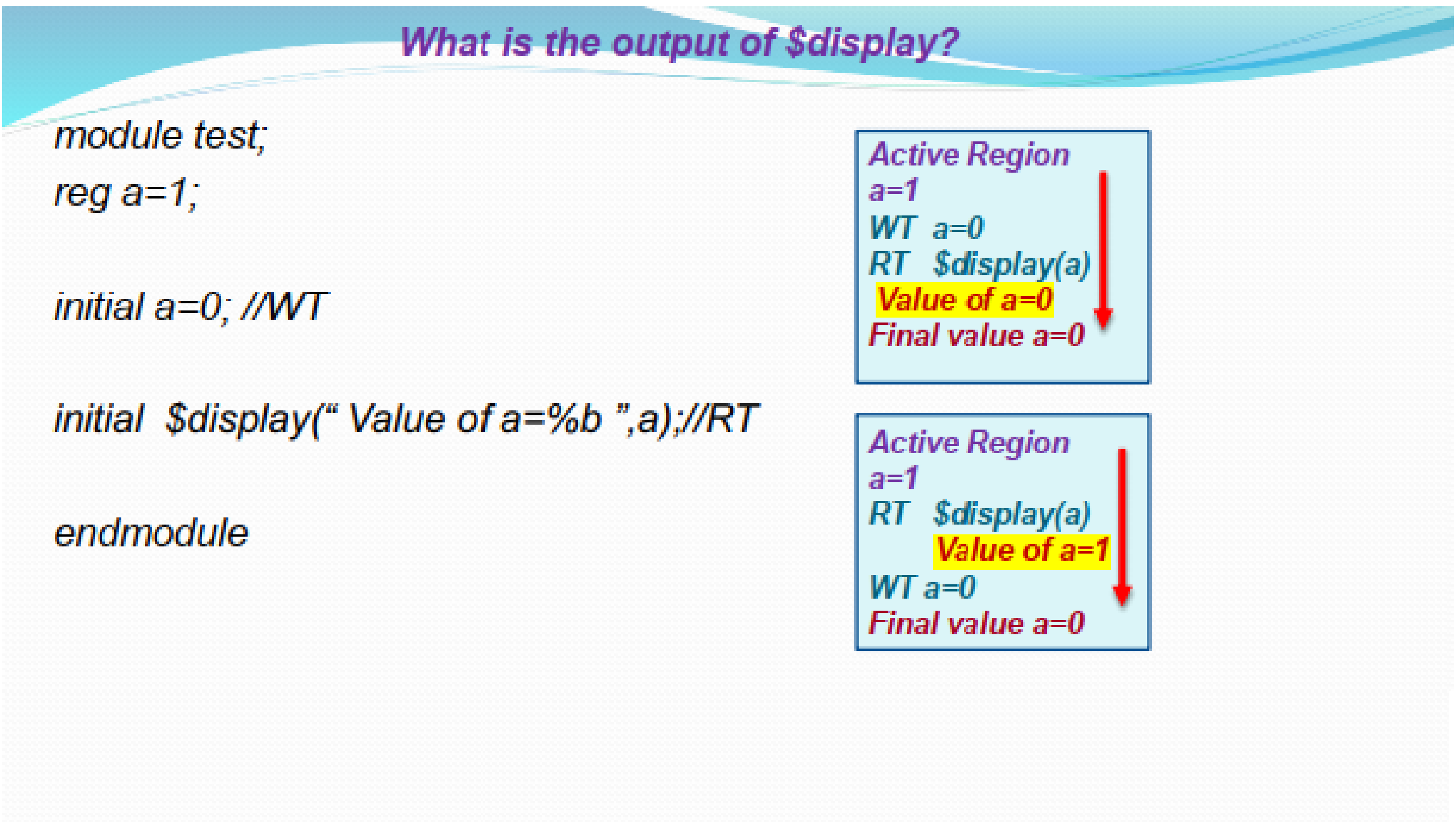


What is the final value of a ?

```
module test ;
reg a;

initial a<=0; //T1

initial a<=1; //T2

endmodule
```

NBA Region
T1 a=0
T2 a=1
Final value of a=1

NBA Region
T2 a=1
T1 a=0
Final value of a=0



What is the output of $display?

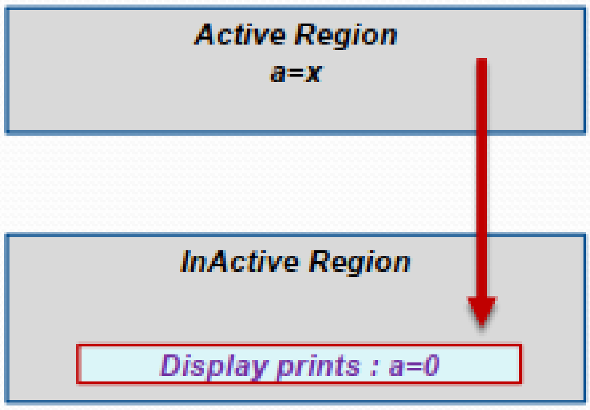
```
module test;
reg a=1;

initial a=0; //WT

initial  $display(" Value of a=%b ",a);//RT

endmodule
```

```
Active Region
a=1
WT  a=0
RT   $display(a)
 Value of a=0
Final value a=0
```

```
Active Region
a=1
RT   $display(a)
          Value of a=1
WT a=0
Final value a=0
```

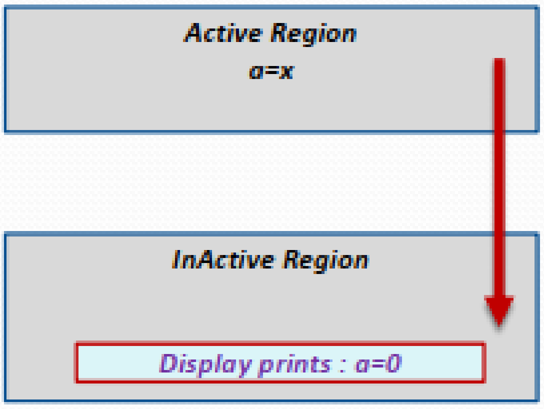
What is the output ?

```
module test;

reg a;

initial a=0;

initial  begin
#0 $display(" a=%b ",a);
end

endmodule
```



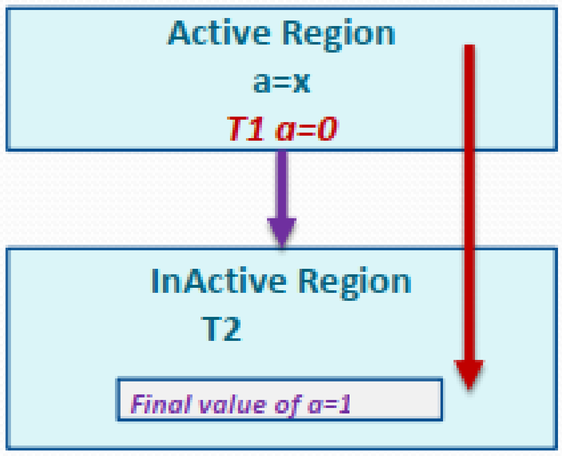
What is the output ?

```
module test;

reg a;

initial a=0;

initial  begin
#0 $display(" a=%b ",a);
end

endmodule
```



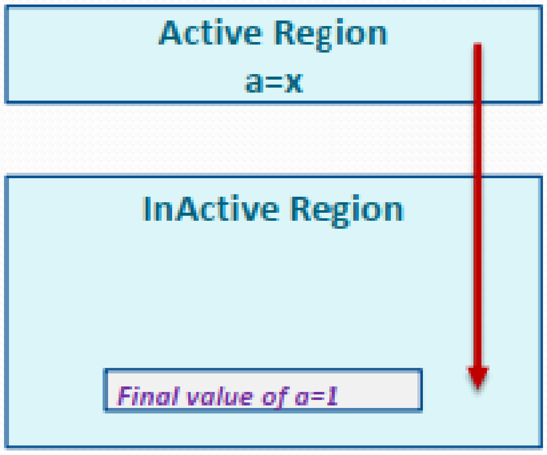
What is the final value of a ?

```
module test ;
reg a;

initial a=0; //T1

initial #0 a=1; //T2

endmodule
```



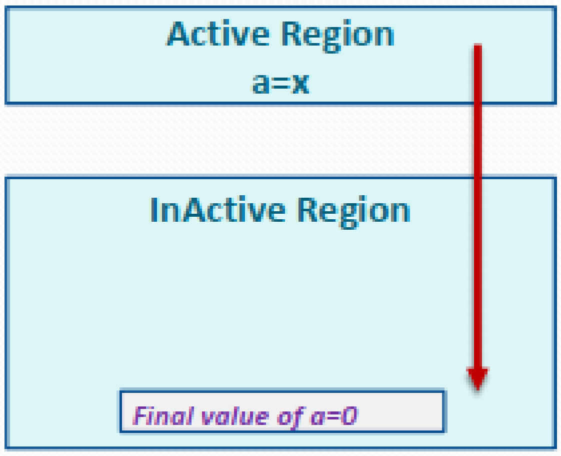
What is the final value of a ?

```
module test ;
reg a;

initial #0 a=0; //T1

initial #0 a=1; //T2

endmodule
```

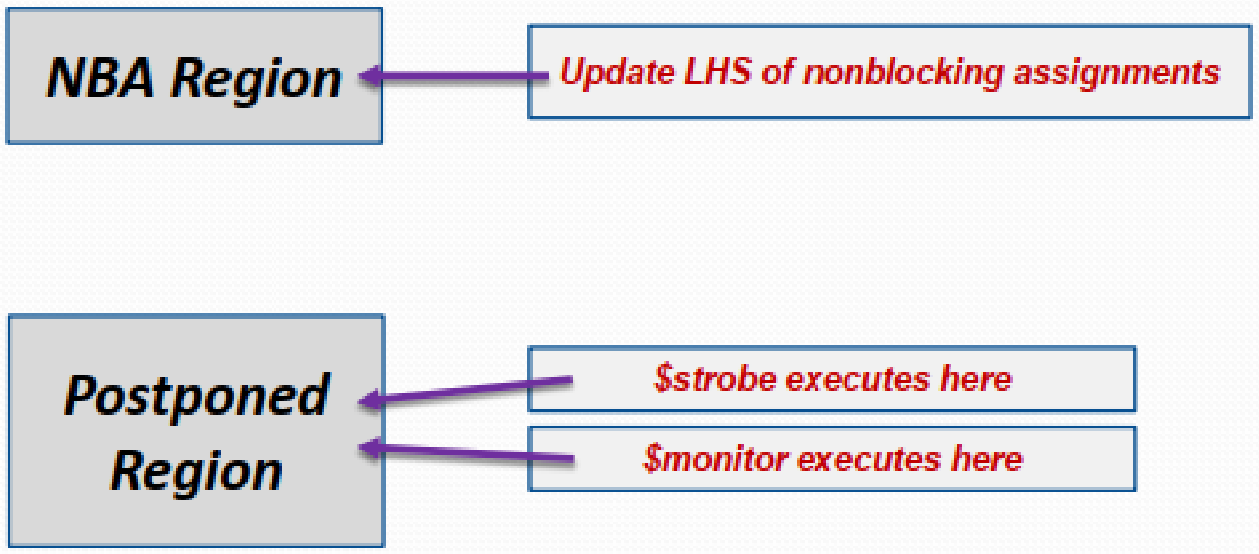


What is the final value of a ?

```
module test ;
reg a;

initial #0 a=0; //T1

initial #0 a=1; //T2

endmodule
```



NBA and Postponed regions



What is the output ?

```
module test;
reg a,b,c=0;


initial  begin
   a=1;
   b <= c;
   a<=b;
$display("a=%0d  b=%0d ",a,b);
end
endmodule
```

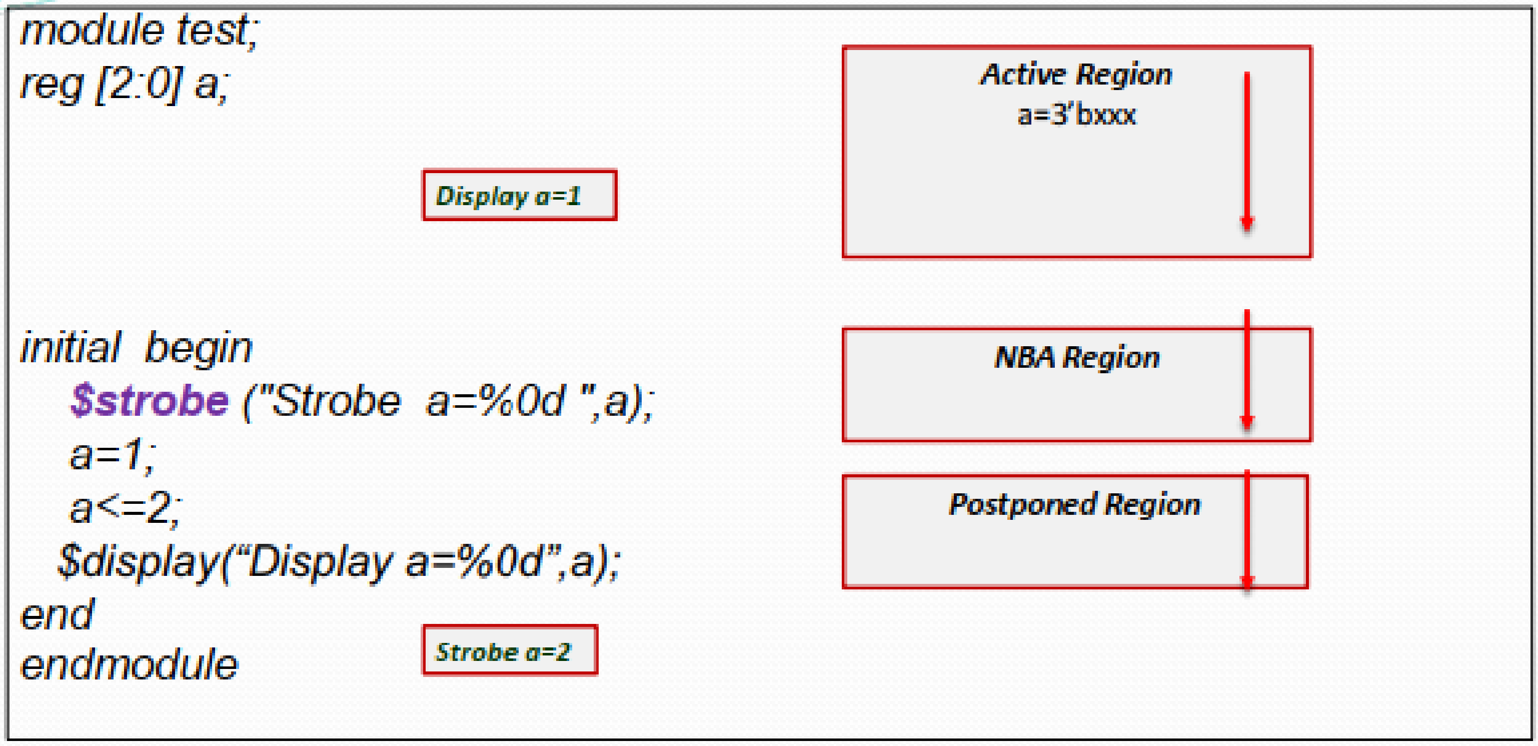
Output of $display is:
a=1 b=x;

Active Region
a=x , b=x,c=0

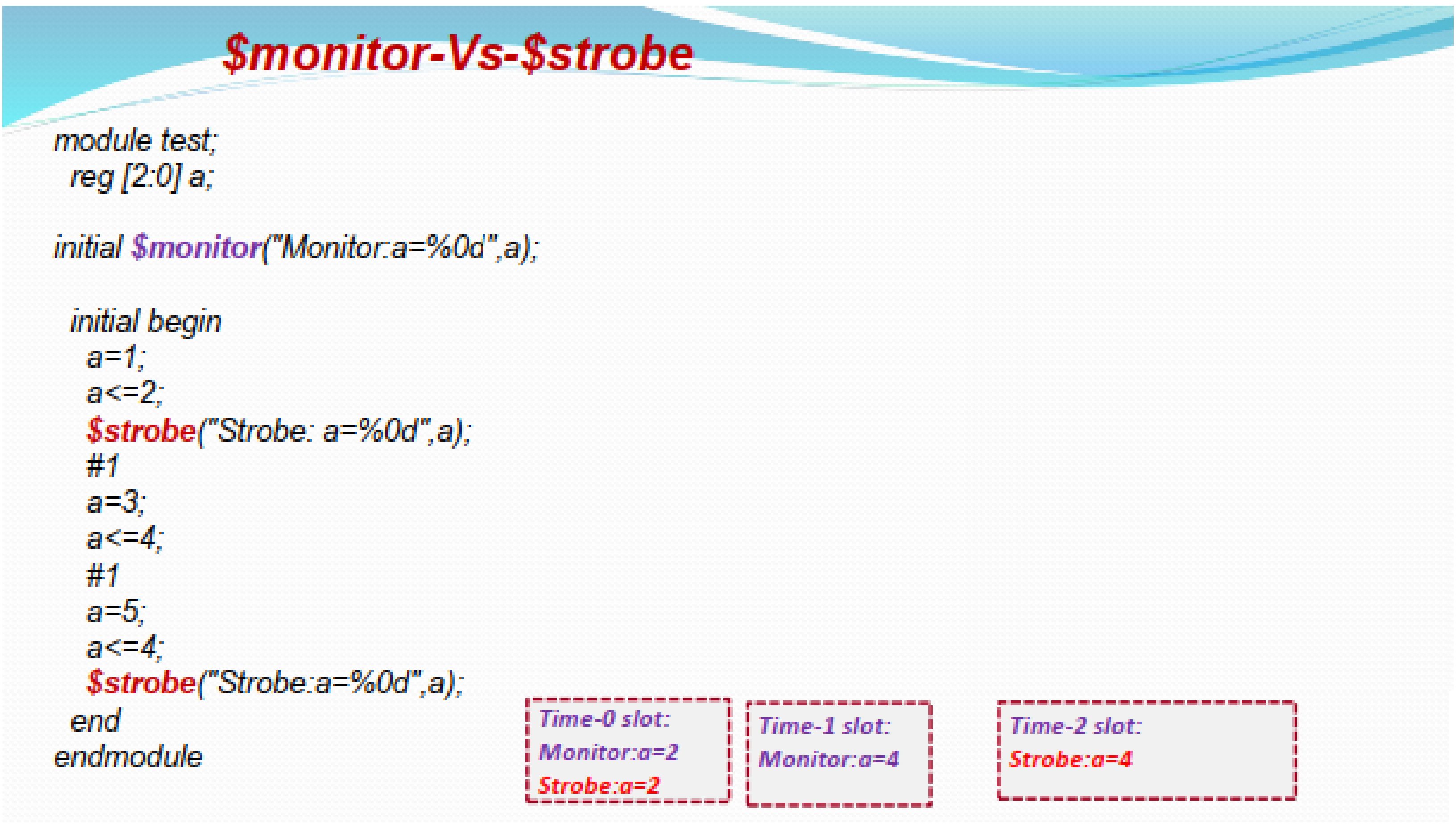
NBA Region

Final values of a,b,c:
a=x , b=0, c=0

$strove-Vs-$display



```
module test;
reg [2:0] a;

initial  begin
   $strobe ("Strobe  a=%0d ",a);
   a=1;
   a<=2;
  $display("Display a=%0d",a);
end
endmodule
```



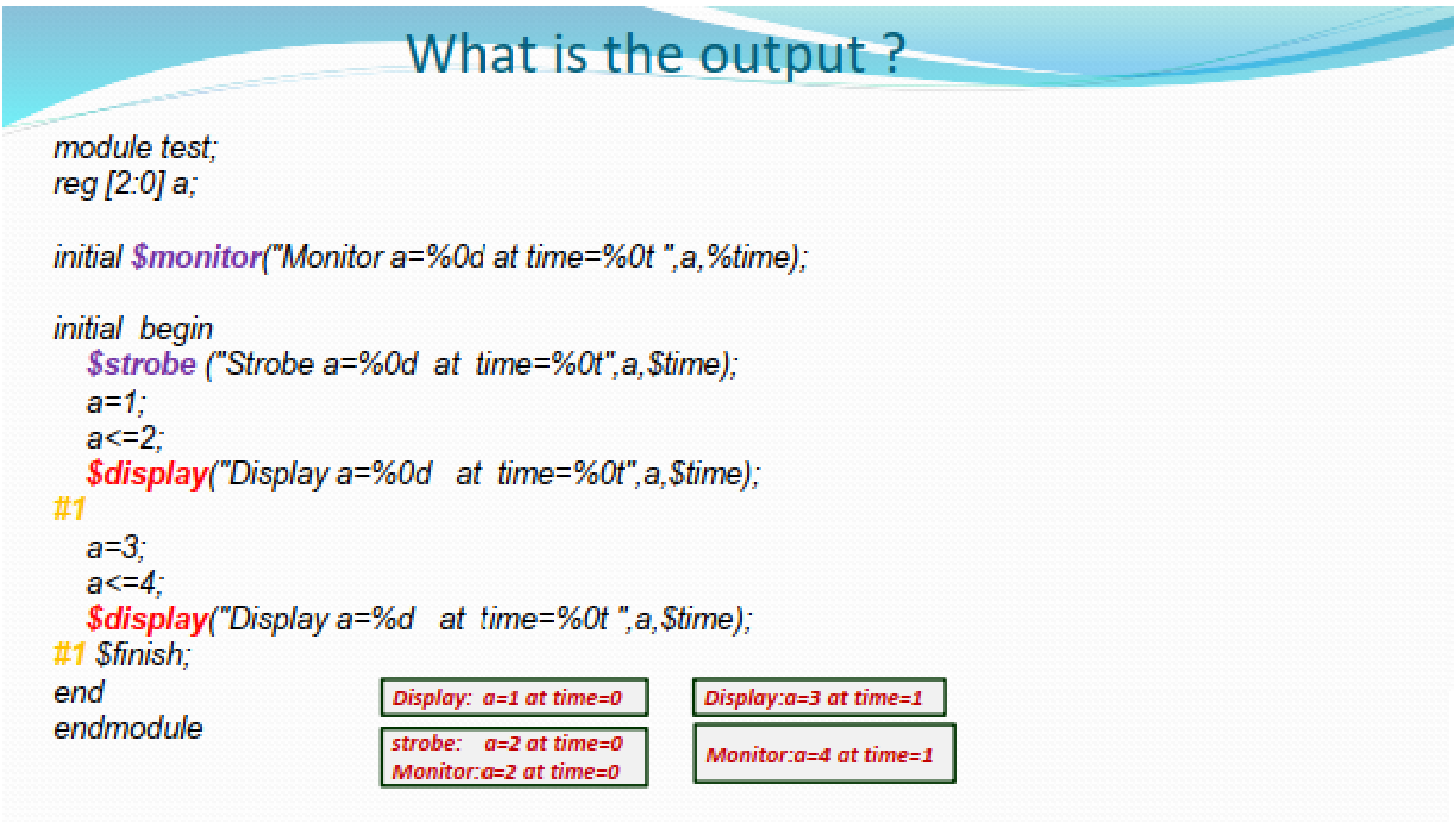
$monitor-Vs-$strobe

```
module test;
  reg [2:0] a;

initial $monitor("Monitor:a=%0d",a);

  initial begin
    a=1;
    a<=2;
    $strobe("Strobe: a=%0d",a);
    #1
    a=3;
    a<=4;
    #1
    a=5;
    a<=4;
    $strobe("Strobe:a=%0d",a);
  end
endmodule
```

```
Time-0 slot:
Monitor:a=2
Strobe:a=2
```

```
Time-1 slot:
Monitor:a=4
```

```
Time-2 slot:
Strobe:a=4
```



What is the output ?

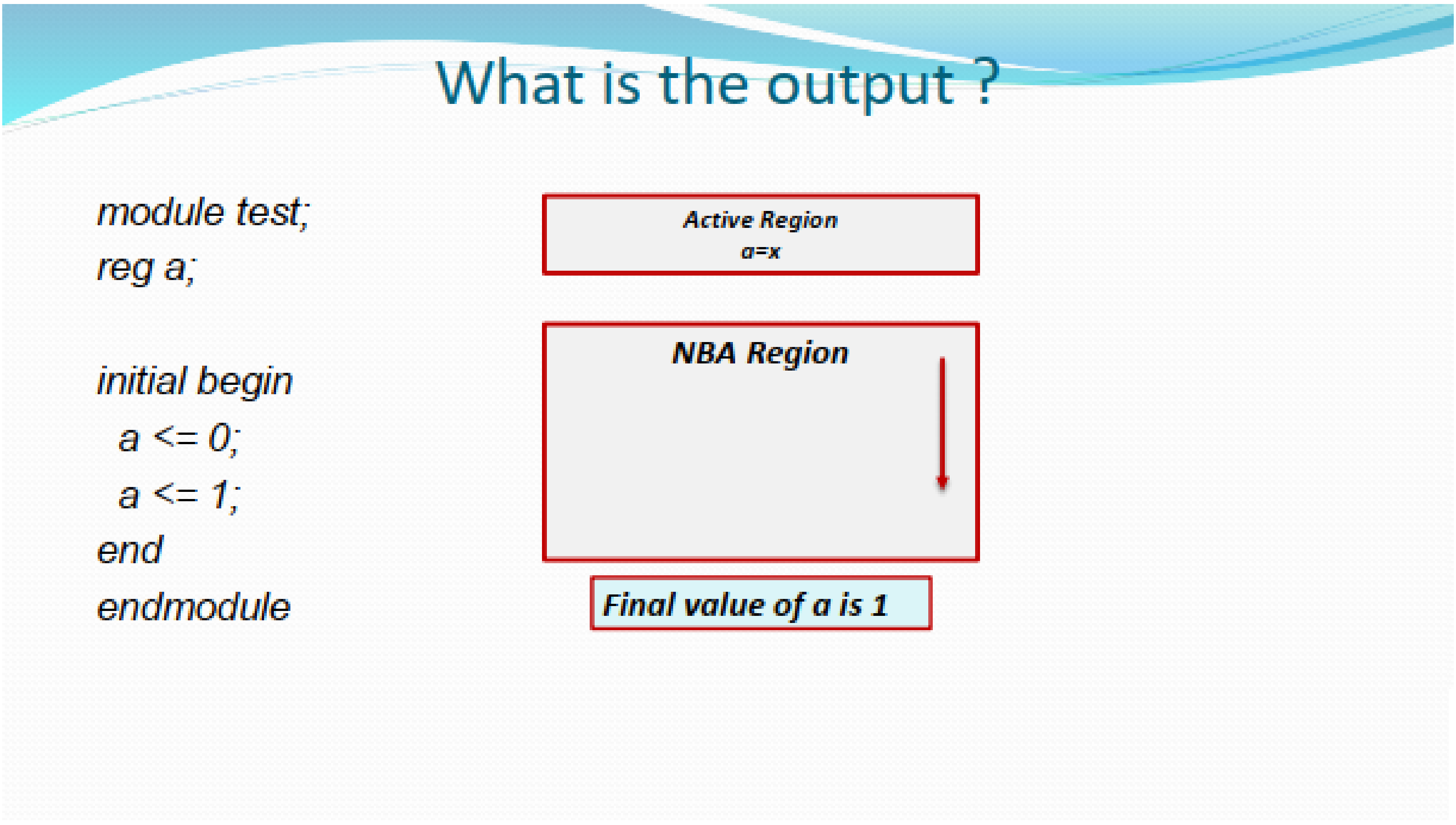
```
module test;
reg [2:0] a;

initial $monitor("Monitor a=%0d at time=%0t ",a,%time);

initial  begin
   $strobe ("Strobe a=%0d  at  time=%0t",a,$time);
   a=1;
   a<=2;
   $display("Display a=%0d   at  time=%0t",a,$time);
#1
   a=3;
   a<=4;
   $display("Display a=%d   at  time=%0t ",a,$time);
#1 $finish;
end
endmodule
```

```
Display:  a=1 at time=0
```

```
strobe:    a=2 at time=0
Monitor:a=2 at time=0
```

```
Display:a=3 at time=1
```

```
Monitor:a=4 at time=1
```



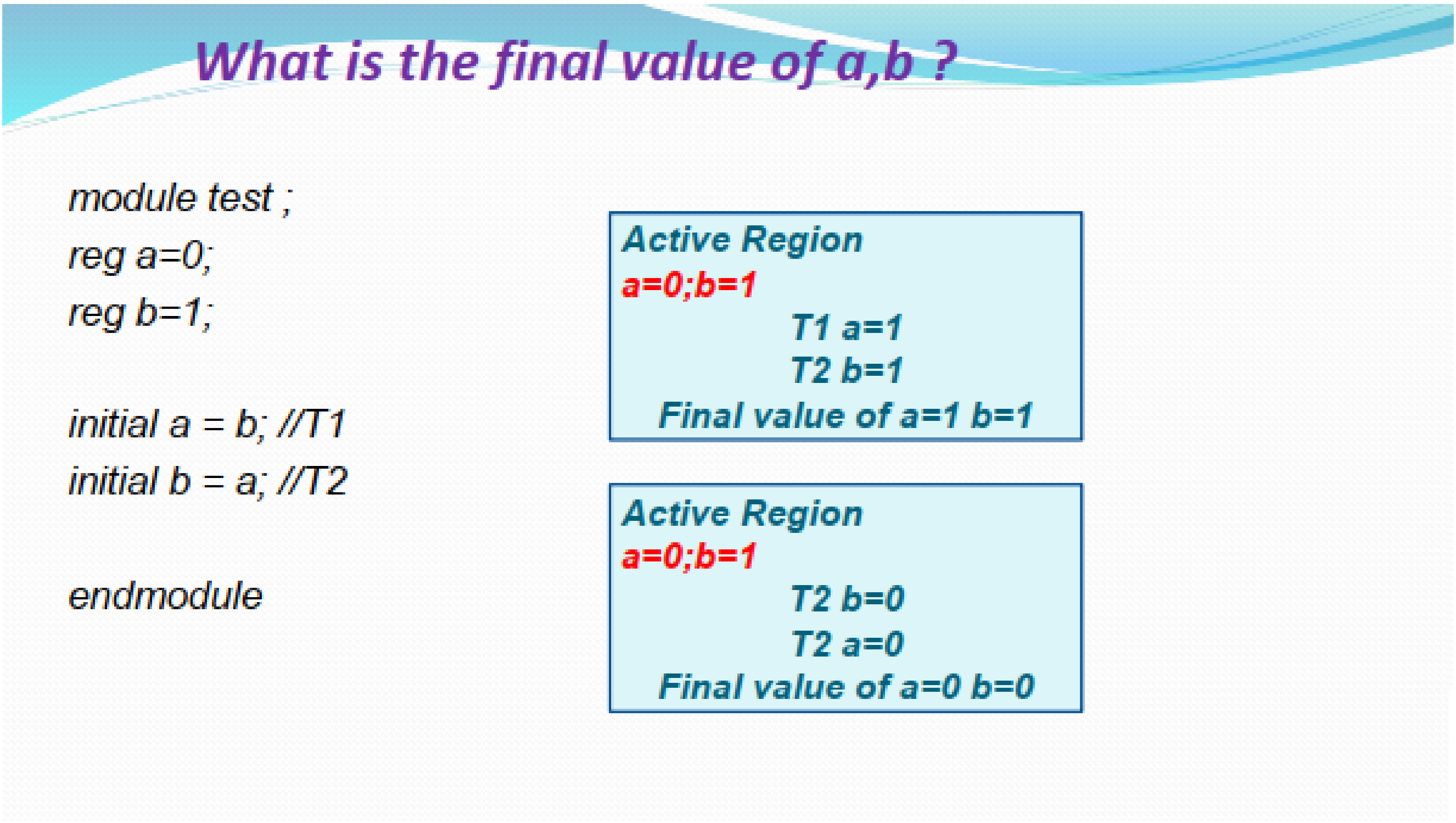
What is the output ?

```
module test;
reg a;

initial begin
  a <= 0;
  a <= 1;
end
endmodule
```

Active Region
a=x

NBA Region

Final value of a is 1



What is the final value of a,b ?

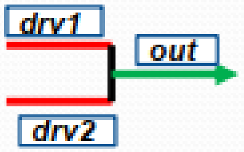
```
module test ;
reg a=0;
reg b=1;

initial a = b; //T1
initial b = a; //T2

endmodule
```

Active Region
a=0;b=1
T1 a=1
T2 b=1
Final value of a=1 b=1

Active Region
a=0;b=1
T2 b=0
T2 a=0
Final value of a=0 b=0

Multiple drivers on wire (net)

```
module test;
wire [3:0] out;
reg [3:0] drv1,drv2;

assign out = drv1;
assign out = drv2;

initial begin
#1 drv1=0; drv2=0;
#1 drv1=1; drv2=1;
#1 drv1=0; drv2=1;
#1 drv1=1; drv2=0;
#1 drv1=1; drv2='z;
#1 drv1='z; drv2='0;
#1 $finish;
end
endmodule
```

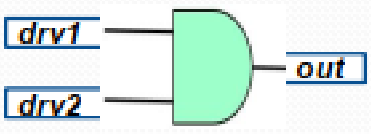


drv1	drv2	out
0	0	0
1	1	1
0	1	X
1	0	X
1	Z	1
Z	0	0

A net can be written by one or more continuous assignments, by primitive outputs, or through module ports.
The resultant value of multiple drivers is determined by the resolution function of the net type

Multiple drivers on wand (net)

```
module test;
 wand   [3:0]  out;
reg [3:0] drv1,drv2;

assign out = drv1;
assign out = drv2;

initial begin
#1 drv1=0; drv2=0;
#1 drv1=1; drv2=1;
#1 drv1=0; drv2=1;
#1 drv1=1; drv2=0;
#1 drv1=1; drv2='z;
#1 drv1='z; drv2='0;
#1 $finish;
end
endmodule
```

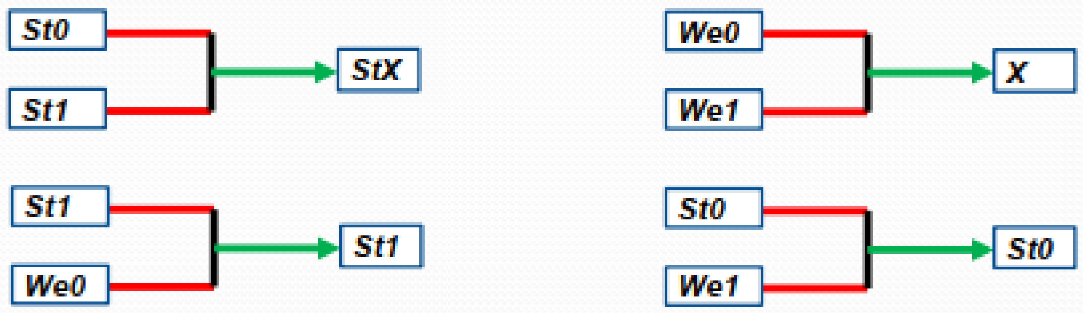


drv1	drv2	out
0	0	0
1	1	1
0	1	0
1	0	0
1	Z	1
Z	0	0

Net type **wand:**
The resultant value of wand net is the logic AND operation of the list of drivers

net types

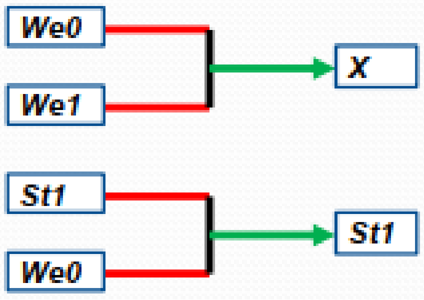
- *Wired nets are of type* ***wor, wand, trior, and triand*** *and are used to model wired logic configurations.*
- *The uwire net is an unresolved or unidriver wire and is used to model nets that* ***allow only a single driver***
- ***Strength resolution:***



Strength resolution:

```
module test;
wire [3:0] out;
reg [3:0] drv1,drv2;

assign (strong1,weak0)  out = drv1;
assign (weak1,weak0)    out = drv2;

initial begin
#1 drv1=0; drv2=0;
#1 drv1=1; drv2=1;
#1 drv1=0; drv2=1;
#1 drv1=1; drv2=0;
#1 drv1=1; drv2='z;
#1 drv1='z; drv2='0;
#1 $finish;
end
endmodule
```

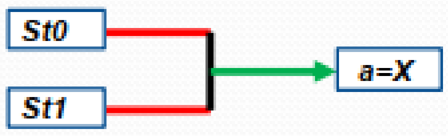
drv1	drv2	out
We0	We0	We0
St1	We1	St1
We0	We1	WeX
St1	We0	St1
St1	Z	St1
Z	We0	We0



What is the output ?

```
module test;
wire a;

assign a = 0;
assign a = 1;

endmodule
```

Final value of a is X



Verilog RTL coding guidelines

- When modeling **sequential logic**, use **nonblocking assignments**.
- When **modeling latches**, use **nonblocking assignments.**
- When **modeling combinational logic** with an always block, **use blocking assignments**.
- When **modeling both sequential and combinational logic** within the same always block, **use nonblocking assignments.**
- Do not mix blocking and nonblocking assignments in the same always block.
- Do not make assignments to the same variable from more than one always block.
- Use $strobe to display values that have been assigned using nonblocking assignments.
- Do not make assignments using #0 delays in RTL.

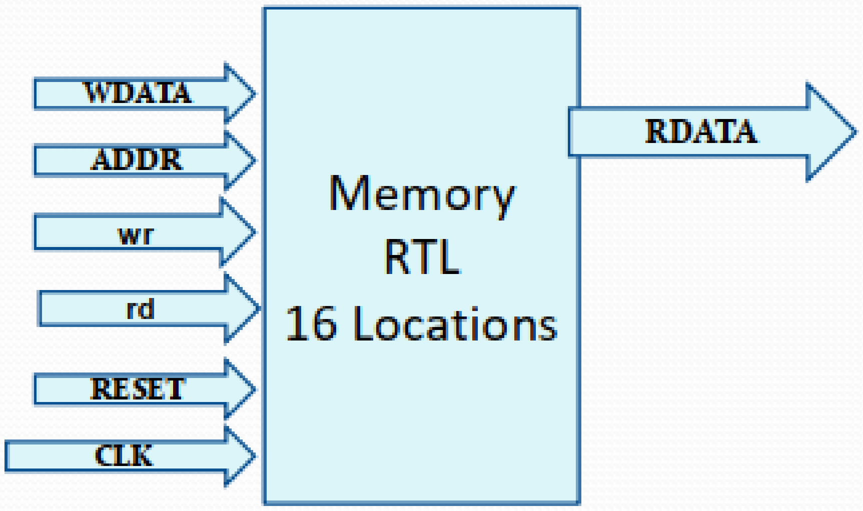
Write verilog DUT

Module rtl_memory

input clk,reset;
input wr,rd;
input [3:0] addr;
input [31:0] wdata;

output [31:0] rdata;

wr = 1 //Write
rd = 1 //Read



Synchronous WRITE and Synchronous READ (Posedge clk)
Asynchronous reset (reset high)

- *Write Verilog RTL memory model (DUT).*
- *Write Verilog Testbench .*

  Write a task to apply reset to DUT.

  Write a task to Write to all 16 locations

  Write a task to Read from all 16 locations.

  Write comparison function to compare write data and read data.

  Write result function to print Test PASS or FAIL.

For example :

WRITE : wr=1, addr=3, wdata=50

READ : rd=1 , addr=3 then you should see value 50 on rdata;