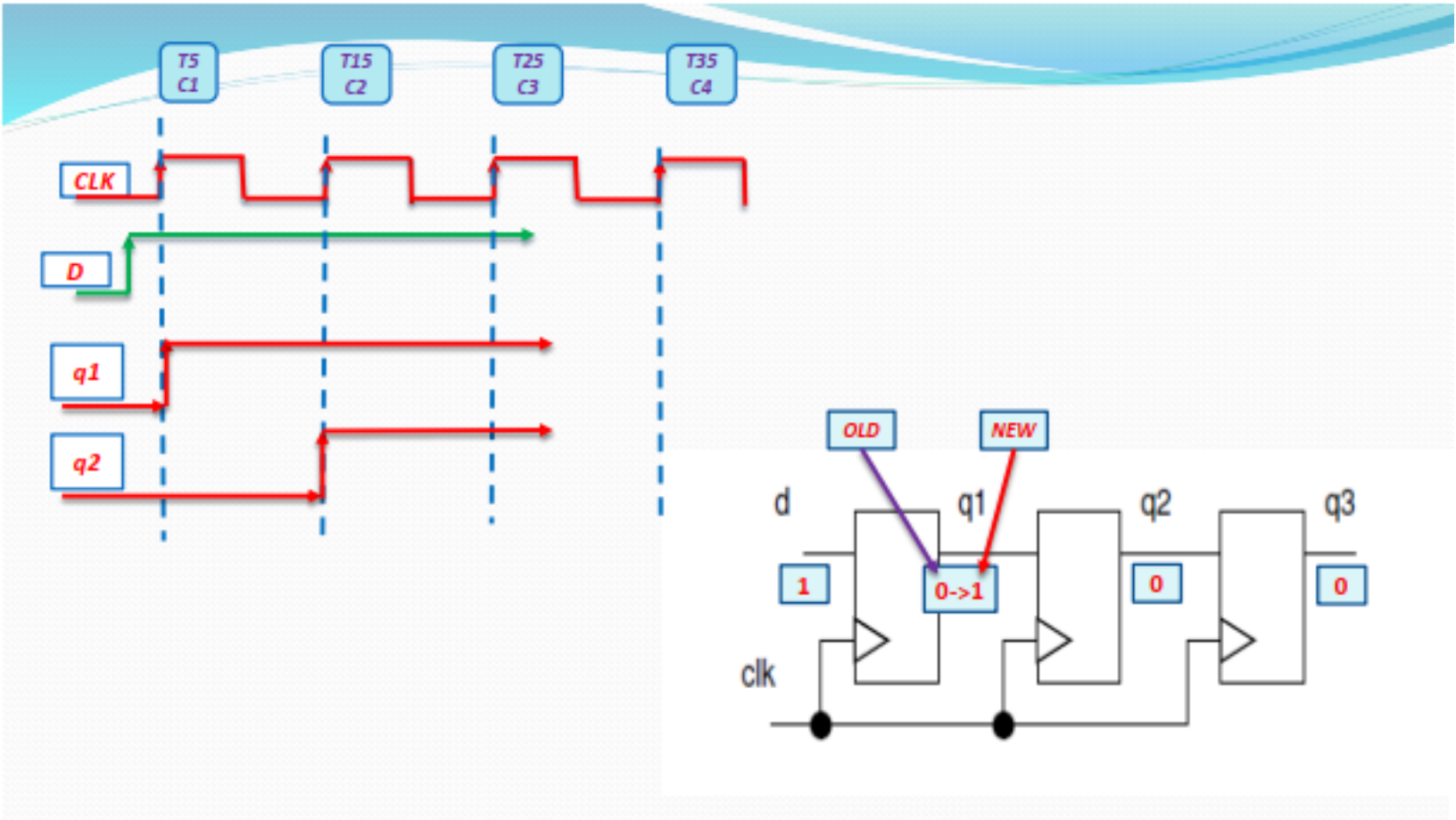
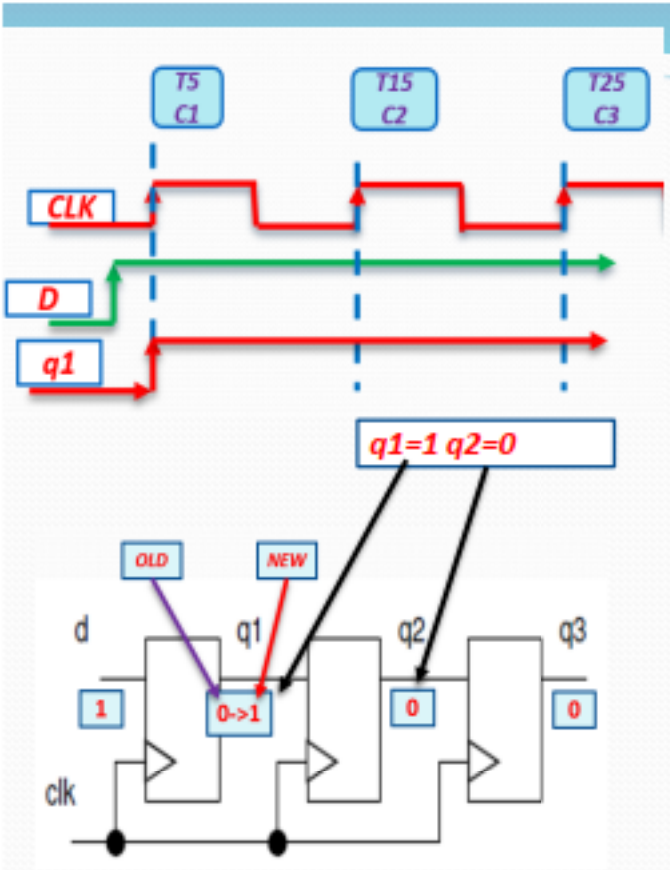
SV PRE Event Regions


T5
C1
T15
C2
T25
C3
T35
C4
CLK
D
q1
q2

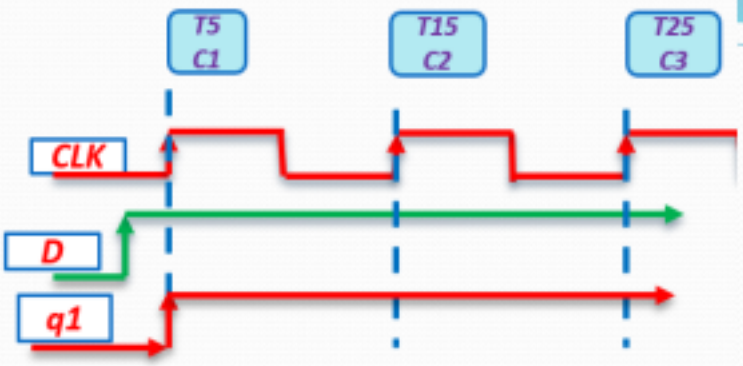
OLD
NEW
d
q1
q2
q3
1
0->1
0
0
clk

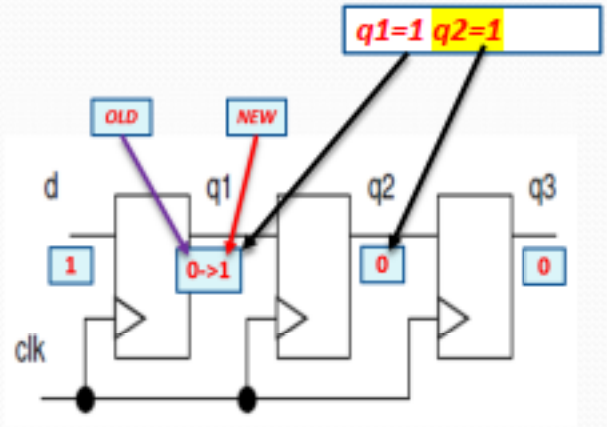


Time 5 slot

Active Region
q1=1 q2=0 q3=0
clk=1 d=1

```
module dut .....

always@(posedge clk)

        q1  =  d;

always@(posedge clk)

        q2 = q1;

always@(posedge clk)
        q3 = q2;

endmodule
```



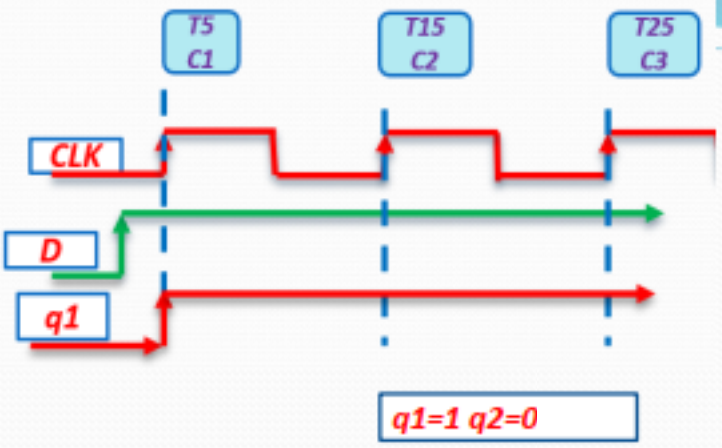


Time 5 slot

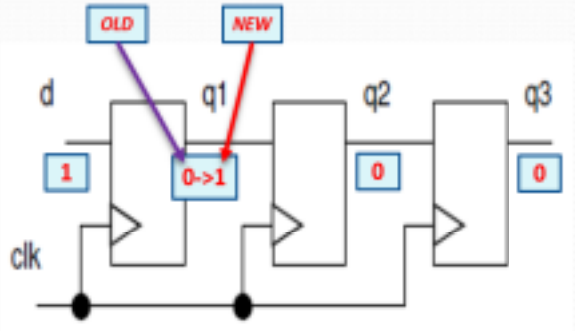
Active Region
q1=1 q2=1 q3=0
clk=1 d=1

always #5 clk = ~clk;

```
module dut .....

always@(posedge clk)

        q1  =  d;

always@(posedge clk)

        q2 = q1;

always@(posedge clk)
        q3 = q2;

endmodule
```



q1=1 q2=0



Time 5 slot

Active Region

q1=0 q2=0 q3=0

clk=1 d=1

always #5 clk = ~clk;

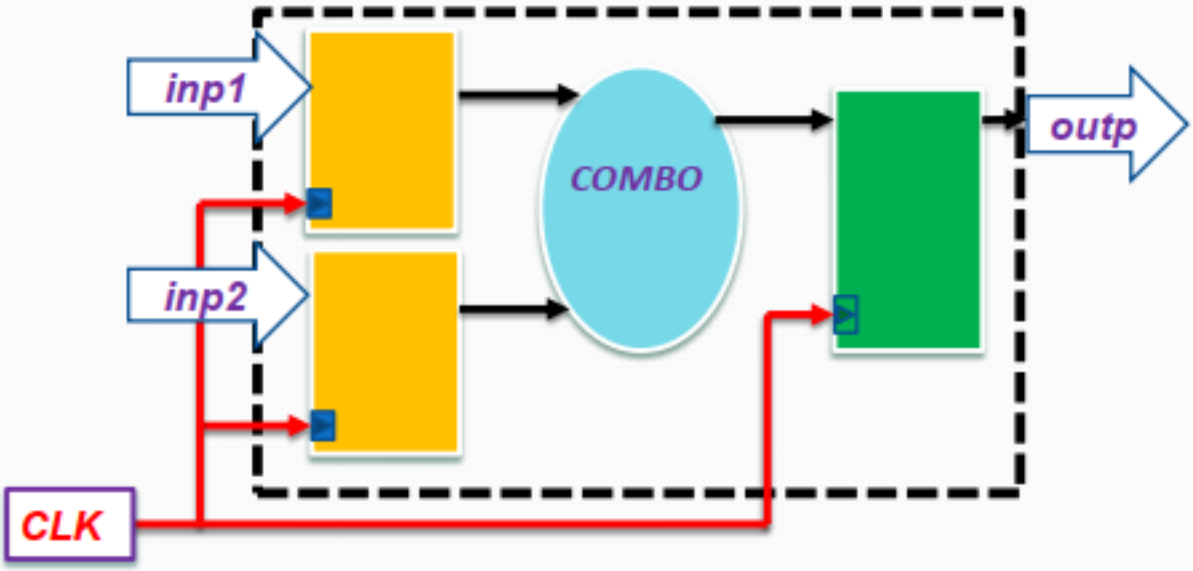
```
module dut ……..

always@(posedge clk)

        q1 <= d;

always@(posedge clk)

        q2 <= q1;

always@(posedge clk)
        q3 <= q2;

endmodule
```

NBA Region

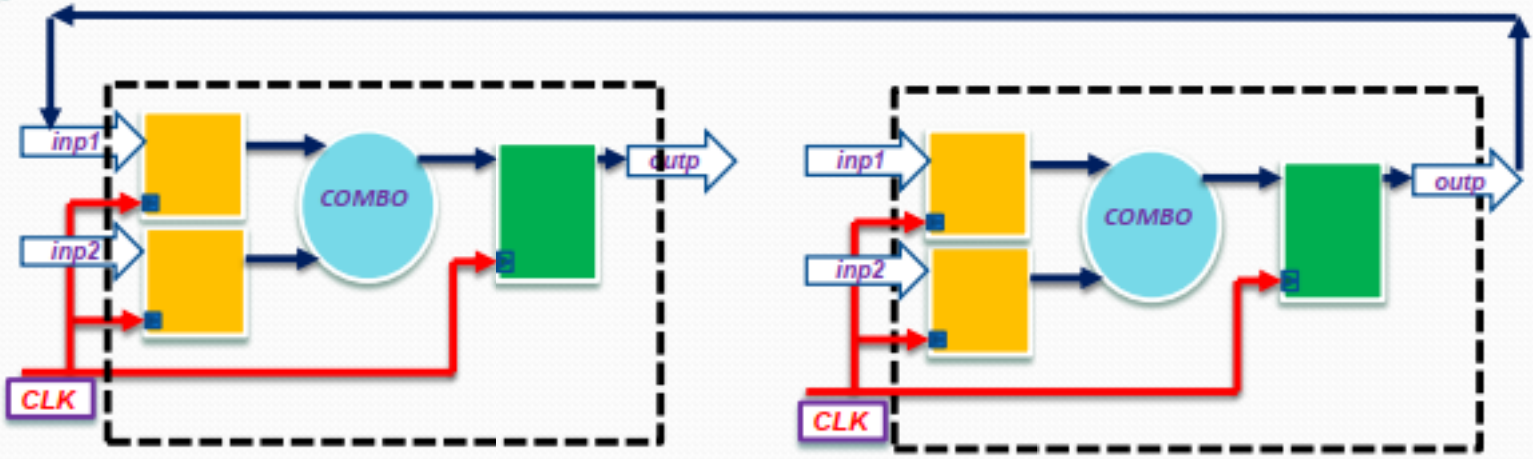
q1=1 q2=0 q3=0

clk=1 d=1

RTL: Register Transfer Level



RTL




```
module testbench;

initial begin

@(posedge clk);
    inp <= 3;

end
endmodule
```

```
module RTL(clk,inp,q.....;

always @(posedge clk);
 q <= inp;


endmodule
```

```
module testbench;
 reg clk,reset;
 reg [3:0] tb_din;
 wire [3:0] tb_dout;

 dff dff_inst (clk,reset,tb_din,tb_dout);

 always #5 clk = ~clk;

 initial begin
   clk=0;reset=0;tb_din=0;
   #1 reset=1;
   #3 reset=0;

   @(posedge clk);

    tb_din=1;

   @(posedge clk);
   tb_din=2;
   @(posedge clk);
#50 $finish;
 end
endmodule
```

```
module dff (clk,reset,din,dout);

 input clk,reset;
 input [3:0] din;
 output [3:0] dout;

 reg [3:0] dout;

always @(posedge clk or posedge reset)
 begin
   if(reset==1'b1)
    dout <= 4'b000;
  else begin
    dout <= din;
  end
 end

endmodule
```


```
module testbench;
 reg clk,reset;
 reg [3:0] tb_din;
 wire [3:0] tb_dout;

 dff dff_inst (clk,reset,tb_din,tb_dout);

 always #5 clk = ~clk;

 initial begin
   clk=0;reset=0;tb_din=0;
   #1 reset=1;
   #3 reset=0;

   @(posedge clk);

    tb_din=1;

   @(posedge clk);
   tb_din=2;
   @(posedge clk);
#50 $finish;
 end
endmodule
```

Time 5 slot

Active Region
clk=1 din=1
reset=0 dout=0

NBA Region
dout=0

din=1 dout=0

RTL:Sampling Stimulus (READ)
TB: Stimulus Gen (Write)

```
odule dff (clk,reset,din,dout);

 input clk,reset;
 input [3:0] din;
 output [3:0] dout;

 reg [3:0] dout;

always @(posedge clk or posedge reset)
 begin
   if(reset==1'b1)
    dout <= 4'b000;
  else begin
    dout <= din;
  end
 end

endmodule
```


```
module testbench;
 reg clk,reset;
 reg [3:0] tb_din;
 wire [3:0] tb_dout;

 dff dff_inst (clk,reset,tb_din,tb_dout);

 always #5 clk = ~clk;

 initial begin
   clk=0;reset=0;tb_din=0;
   #1 reset=1;
   #3 reset=0;

   @(posedge clk);

    tb_din=1;

   @(posedge clk);
   tb_din=2;
   @(posedge clk);
#50 $finish;
 end
endmodule
```

Time 5 slot

Active Region
clk=1 din=1
reset=0 dout=0

NBA Region
dout=1

din=1 dout=1

TB: Stimulus Gen (Write)
RTL:Sampling Stimulus (READ)

```
odule dff (clk,reset,din,dout);

 input clk,reset;
 input [3:0] din;
 output [3:0] dout;

 reg [3:0] dout;

always @(posedge clk or posedge reset)
 begin
   if(reset==1'b1)
    dout <= 4'b000;
  else begin
    dout <= din;
  end
 end

endmodule
```

TB -> RTL races

Active Region

T1 : RTL Sampling the stimulus (Read on inputs):
RTL Thread waiting on posedge clk : always@(posedge clk) q <= d;

T2: TB Stimulus Generation and Driving (Write on inputs):
TB Thread : @(posedge clk) d = 1;

- Mixing Design and Testbench activity in the active region causes race .
- Will your design sample old value or new value ?

Depends on simulator scheduling order


```
module testbench;
 reg clk,reset;
 reg [3:0] tb_din;
 wire [3:0] tb_dout;

 dff dff_inst (clk,reset,tb_din,tb_dout);

 always #5 clk = ~clk;

 initial begin
   clk=0;reset=0;tb_din=0;
   #1 reset=1;
   #3 reset=0;

   @(posedge clk);

    tb_din <= 1;

   @(posedge clk);
   tb_din <= 2;
   @(posedge clk);
#50 $finish;
 end
endmodule
```

Time 5 slot

Active Region
clk=1 din=1
reset=0 dout=0

NBA Region
din=1
dout=0

din=1 dout=0

TB: Stimulus Gen (Write)
RTL:Sampling Stimulus (READ)

```
odule dff (clk,reset,din,dout);

 input clk,reset;
 input [3:0] din;
 output [3:0] dout;

 reg [3:0] dout;

always @(posedge clk or posedge reset)
 begin
   if(reset==1'b1)
    dout <= 4'b000;
  else begin
    dout <= din;
  end
 end

endmodule
```