Clocking Block

Clocking block

- *A clocking block assembles signals that are synchronous to a particular clock and makes their timing explicit.*

clocking clocking_blk_name @(edge specifier);

<clocking block items>

endclocking

```
interface simple_bus (input clk); // Define the interface
logic [31:0] rdata,wdata;
logic [3:0] addr;
logic reset,wr,rd;

clocking cb @(posedge clk);
output addr,wr,rd,wdata; //Directions are w.r.t testbench
input rdata;
endclocking

modport   dut_ports    (input  reset,addr,wr,rd,wdata ,clk,   output rdata);
modport    tb_ports    (clocking cb,  output reset);

endinterface: simple_bus
```

Clocking block

- *An interface block uses a clocking block to specify the timing of synchronous signals relative to the clocks.*
- *Any signal in a clocking block is now driven or sampled synchronously.*

➢ ***Driving** clocking block signal:*

```
initial
    vif.cb.wdata <= 32'hffff;
```

➢ ***Sampling** of clocking block signal:*

```
initial
    local_data = vif.cb.rdata;
```

```
clocking cb @(posedge clk);
output wdata,addr,wr;
input rdata;
endclocking
```

```
program test (dut_if.tb  vif);

initial begin
$display("t1=%0t",$time);
vif.cb.inp1  = 10;
vif.cb.inp2  = 20;

@(vif.cb);
end
endprogram
```

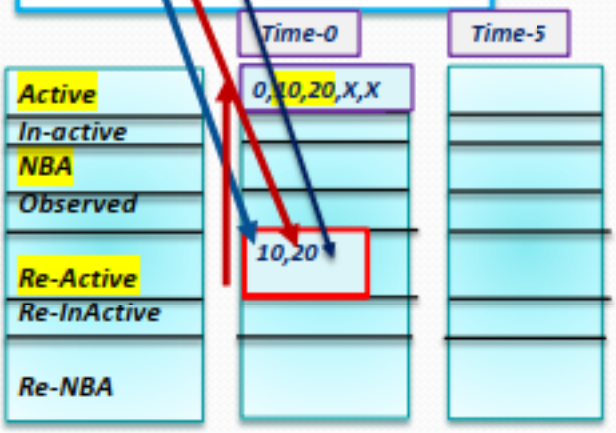
```
module top;
logic clk = 0;

always #5 clk = ~clk;

dut_if    if_inst (clk);

dut  dut_inst (clk,if_inst.inp1, if_inst.inp2. if_inst.q1, if_inst.outp2);


test   test_inst (.vif (if_inst ) );

endmodule
```

```
interface dut_if (input clk);
logic  [7:0] inp1,inp2;
logic  [7:0] q1,outp2;

clocking cb @(posedge clk);
  output inp1,inp2;
  input q1,outp1;
endclocking

modport tb (clocking cb);
endinterface
```

```
module dut (clk,inp1,inp2,q1,outp2);
input clk;
input  [7:0] inp1,inp2;
output [7:0] q1,outp2;

always @(inp2)
outp2 = ++inp2;

always @(posedge clk)
q1<=inp1;

endmodule
```

```
program test (dut_if.tb  vif);

initial begin
$display("t1=%0t",$time);
vif.inp1 = 10;
vif.inp2 = 20;

@(posedge vif.clk);
end
endprogram
```

```
module top;
logic clk = 0;

always #5 clk = ~clk;

dut_if    if_inst (clk);

dut  dut_inst (clk,if_inst.inp1, if_inst.inp2. if_inst.q1, if_inst.outp2);


test   test_inst (.vif (if_inst ) );

endmodule
```

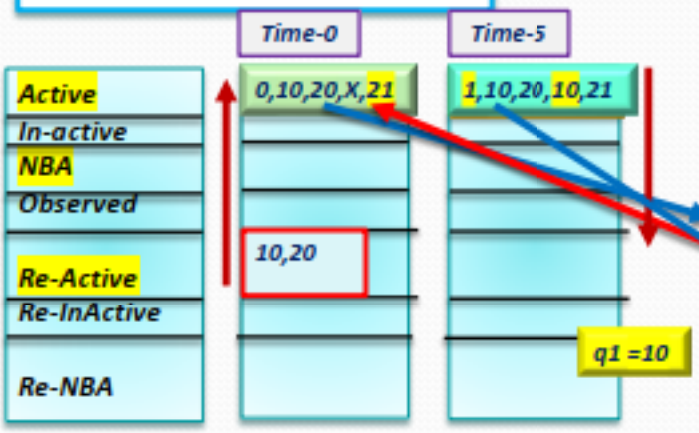



```
module dut (clk,inp1,inp2,q1,outp2);
input clk;
input  [7:0] inp1,inp2;
output [7:0] q1,outp2;

always @(inp2)
outp2 = ++inp2;

always @(posedge clk)
q1<=inp1;

endmodule
```

```
interface dut_if (input clk);
logic  [7:0] inp1,inp2;
logic  [7:0] q1,outp2;

modport tb (output inp1,inp2, input
q1,outp2,clk);
endinterface
```

```
program test (dut_if.tb  vif);

initial begin
$display("t1=%0t",$time);
vif.inp1  = 10;
vif.inp2  = 20;

@(posedge vif.clk);
end
endprogram
```

```
module top;
logic clk = 0;

always #5 clk = ~clk;

dut_if    if_inst (clk);

dut  dut_inst (clk,if_inst.inp1, if_inst.inp2. if_inst.q1, if_inst.outp2);


test   test_inst (.vif (if_inst ) );

endmodule
```



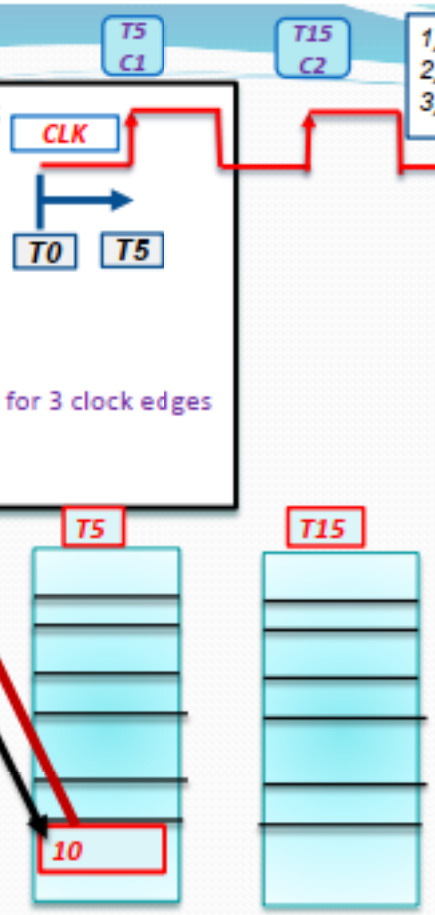

```
module dut (clk,inp1,inp2,q1,outp2);
input clk;
input  [7:0] inp1,inp2;
output [7:0] q1,outp2;

always @(inp2)
outp2 = ++inp2;

always @(posedge clk)
q1<=inp1;

endmodule
```

```
interface dut_if (input clk);
logic  [7:0] inp1,inp2;
logic  [7:0] q1,outp2;

modport tb (output inp1,inp2,
input q1,outp2,clk);
endinterface
```

Driving CB

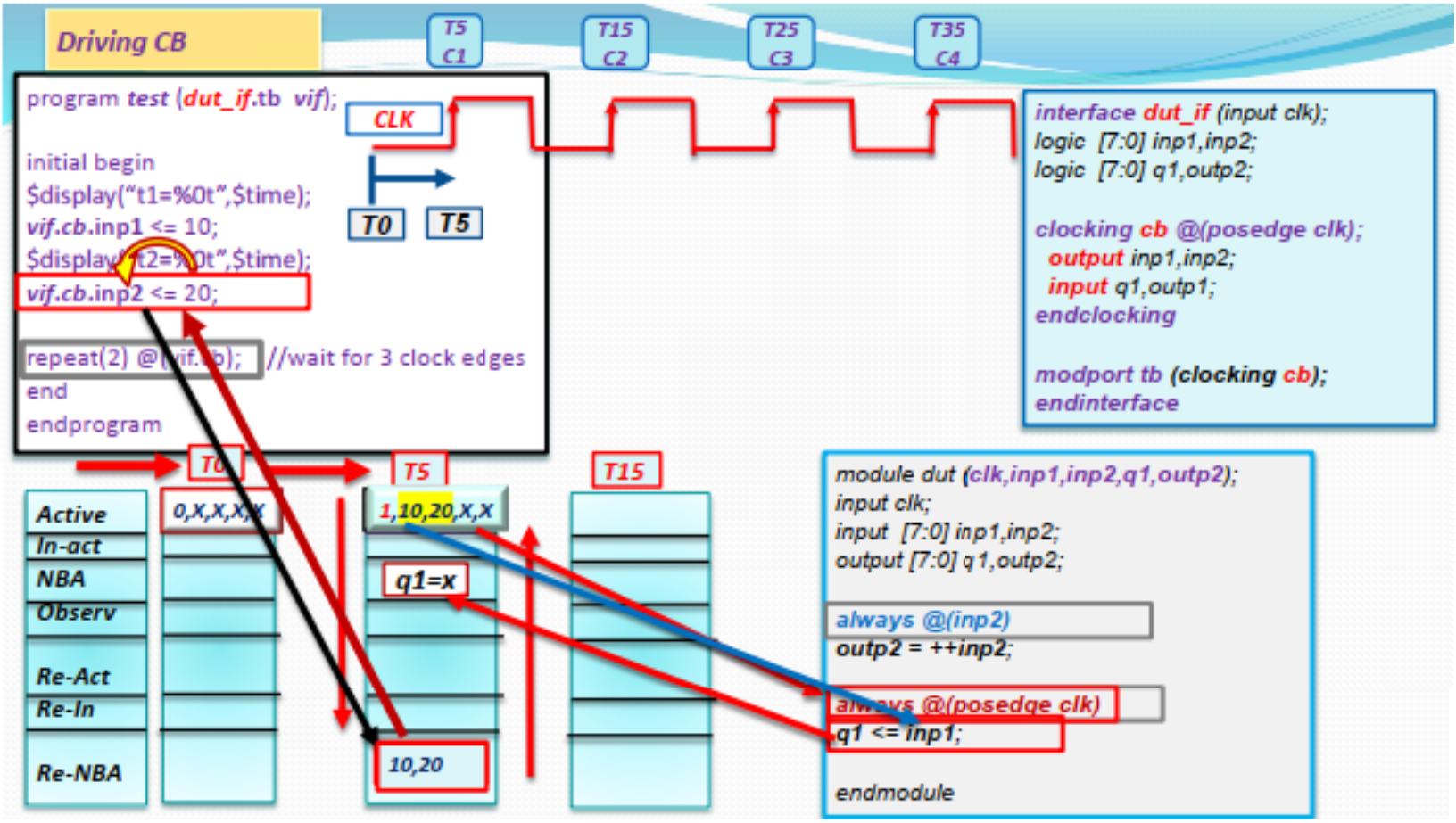
1) What is the current simulation time?
2) Is there a posedge clk available at this time?
3) If(NO) then When the next posedge clk is going to be available?

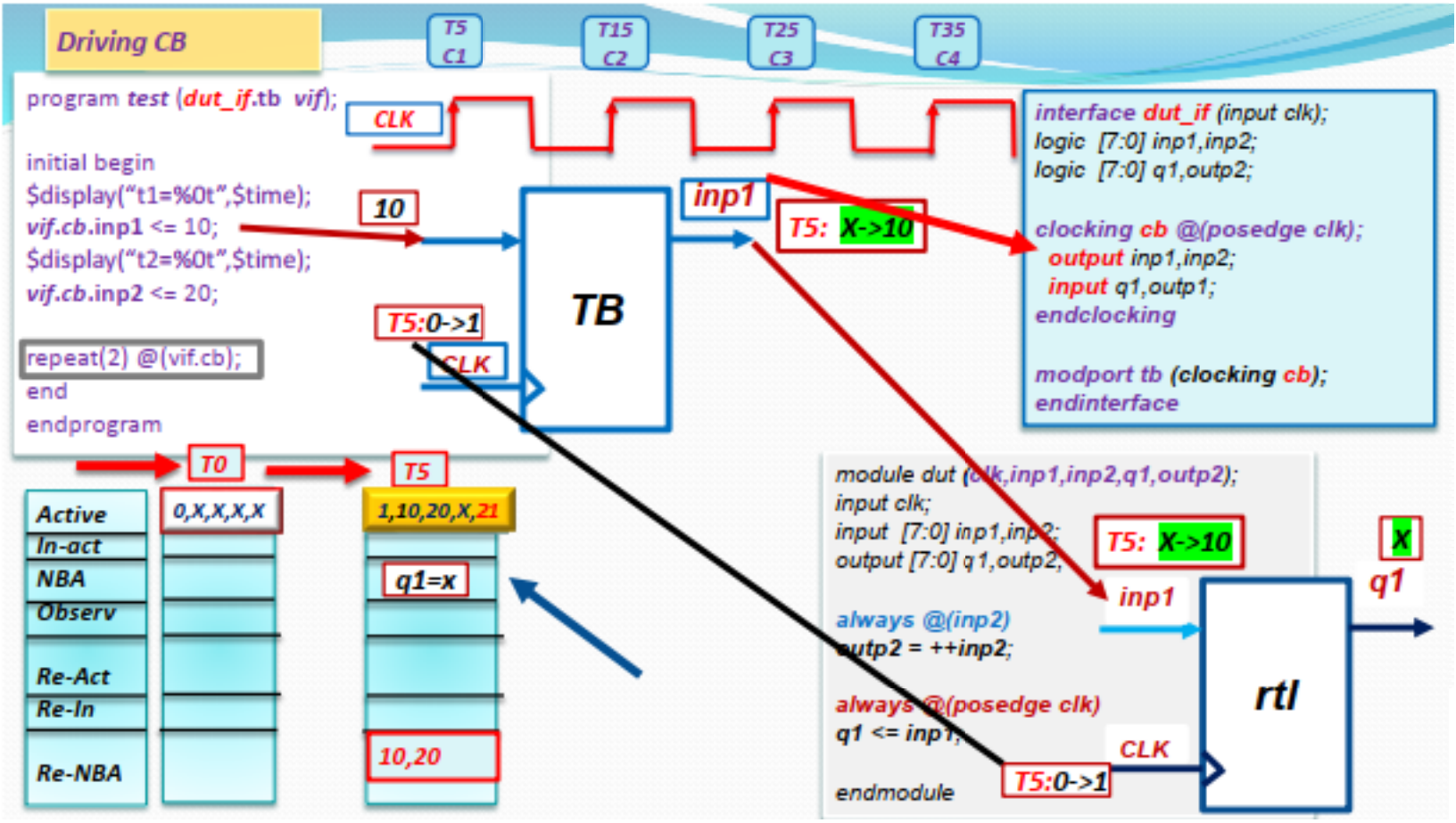
```
program test (dut_if.tb  vif);

initial begin
$display("1=%0t",$time);
vif.cb.inp1 <= 10;
$display("2=%0t",$time);
vif.cb.inp2 <= 10;

repeat(2) @(vif.cb);    //wait for 3 clock edges
end
endprogram
```

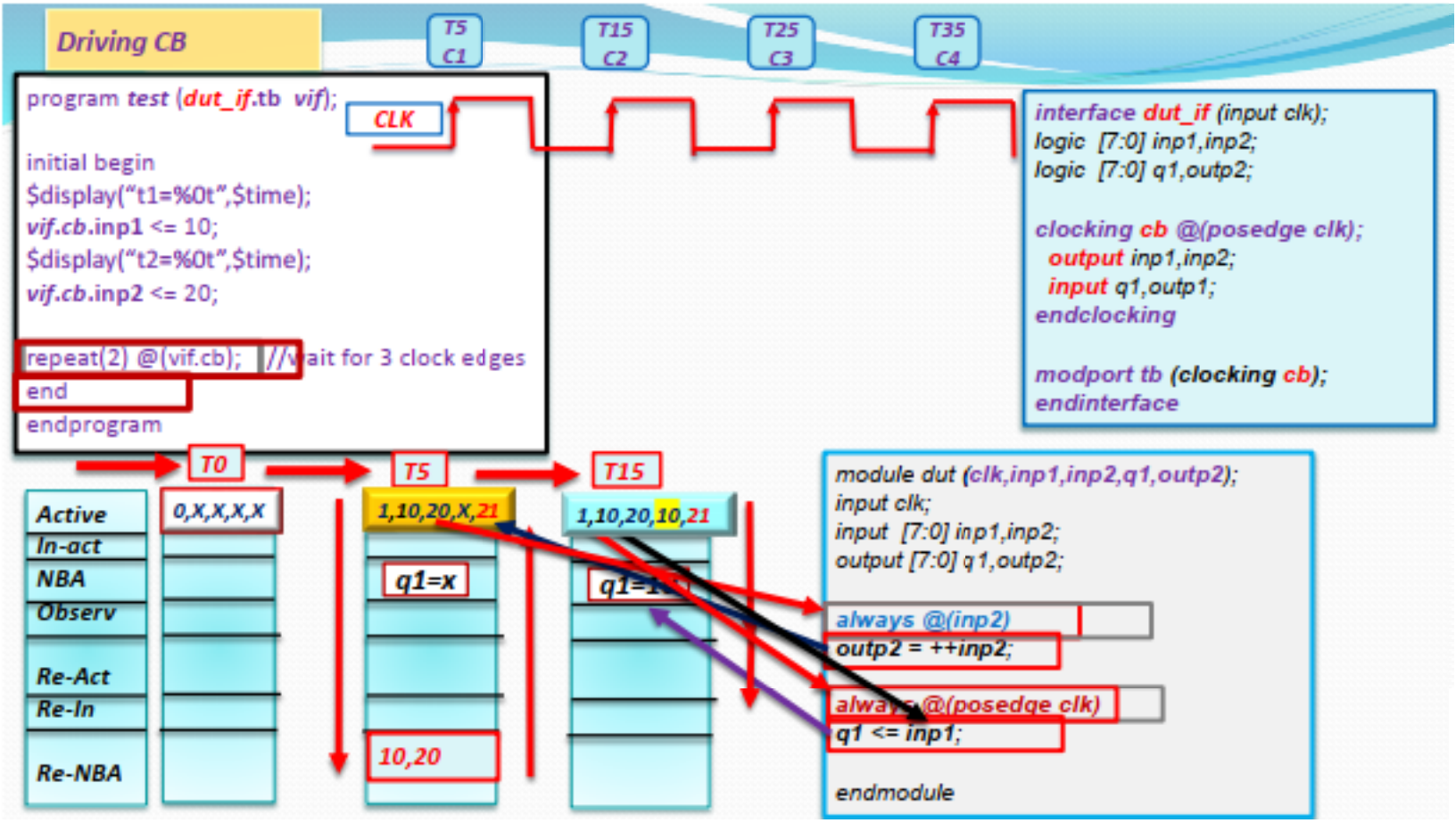



```
logic  [7:0] inp1,inp2;
logic  [7:0] q1,outp2;

clocking cb @(posedge clk);
  output inp1,inp2;
  input q1,outp1;
endclocking

modport tb (clocking cb);
endinterface
```

```
module dut (clk,inp1,inp2,q1,outp2);
input clk;
input  [7:0] inp1,inp2;
output [7:0] q1,outp2;

always @(inp2)
outp2 = ++inp2;

always @(posedge clk)
q1 <= inp1;

endmodule
```


Driving CB
T5 C1
T15 C2
T25 C3
T35 C4
program test (dut_if.tb vif);
initial begin
$display("t1=%0t",$time);
vif.cb.inp1 <= 10;
$display("t2=%0t",$time);
vif.cb.inp2 <= 20;
repeat(2) @(vif.cb); //wait for 3 clock edges
end
endprogram
CLK
T0
T5
interface dut_if (input clk);
logic [7:0] inp1,inp2;
logic [7:0] q1,outp2;
clocking cb @(posedge clk);
output inp1,inp2;
input q1,outp1;
endclocking
modport tb (clocking cb);
endinterface
T0
T5
T15
Active
In-act
NBA
Observ
Re-Act
Re-In
Re-NBA
0,X,X,X,X
1,10,20,X,X
q1=x
10,20
module dut (clk,inp1,inp2,q1,outp2);
input clk;
input [7:0] inp1,inp2;
output [7:0] q1,outp2;
always @(inp2)
outp2 = ++inp2;
always @(posedge clk)
q1 <= inp1;
endmodule


Driving CB
T5 C1
T15 C2
T25 C3
T35 C4
program test (dut_if.tb vif);
initial begin
$display("t1=%0t",$time);
vif.cb.inp1 <= 10;
$display("t2=%0t",$time);
vif.cb.inp2 <= 20;
repeat(2) @(vif.cb);
end
endprogram
CLK
10
inp1
T5: X->10
TB
T5:0->1
CLK
interface dut_if (input clk);
logic [7:0] inp1,inp2;
logic [7:0] q1,outp2;
clocking cb @(posedge clk);
output inp1,inp2;
input q1,outp1;
endclocking
modport tb (clocking cb);
endinterface
T0
T5
Active
In-act
NBA
Observ
Re-Act
Re-In
Re-NBA
0,X,X,X,X
1,10,20,X,21
q1=x
10,20
module dut (clk,inp1,inp2,q1,outp2);
input clk;
input [7:0] inp1,inp2;
output [7:0] q1,outp2;
always @(inp2)
outp2 = ++inp2;
always @(posedge clk)
q1 <= inp1;
endmodule
T5: X->10
inp1
X
q1
rtl
CLK
T5:0->1


Driving CB
T5 C1
T15 C2
T25 C3
T35 C4
CLK
program test (dut_if.tb vif);
initial begin
$display("t1=%0t",$time);
vif.cb.inp1 <= 10;
$display("t2=%0t",$time);
vif.cb.inp2 <= 20;
repeat(2) @(vif.cb); //wait for 3 clock edges
end
endprogram
interface dut_if (input clk);
logic [7:0] inp1,inp2;
logic [7:0] q1,outp2;
clocking cb @(posedge clk);
output inp1,inp2;
input q1,outp1;
endclocking
modport tb (clocking cb);
endinterface
T0
T5
T15
Active
In-act
NBA
Observ
Re-Act
Re-In
Re-NBA
0,X,X,X,X
1,10,20,X,21
q1=x
10,20
1,10,20,10,21
module dut (clk,inp1,inp2,q1,outp2);
input clk;
input [7:0] inp1,inp2;
output [7:0] q1,outp2;
always @(inp2)
outp2 = ++inp2;
always @(posedge clk)
q1 <= inp1;
endmodule

Driving CB

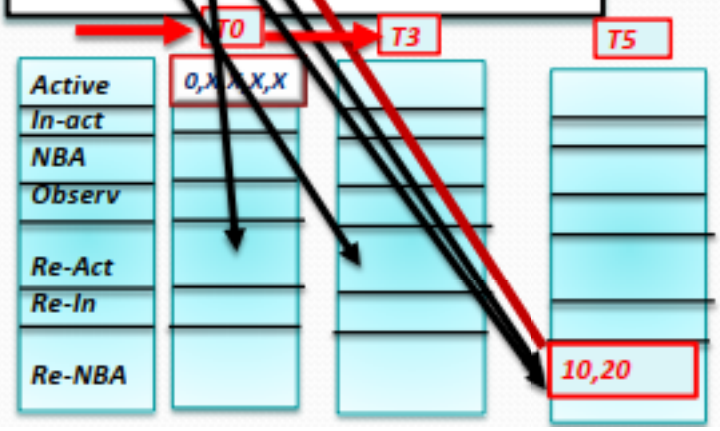
T5 C1 | T15 C2 | T25 C3 | T35 C4

CLK

```
program test (dut_if.tb  vif);

initial begin
$display("T1=%0t",$time);
#3  vif.cb.inp1 <= 10;
$display("T2=%0t",$time);
#1  vif.cb.inp2 <= 20;

repeat(2) @(vif.cb);   //wait for 3 clock edges
end
endprogram
```

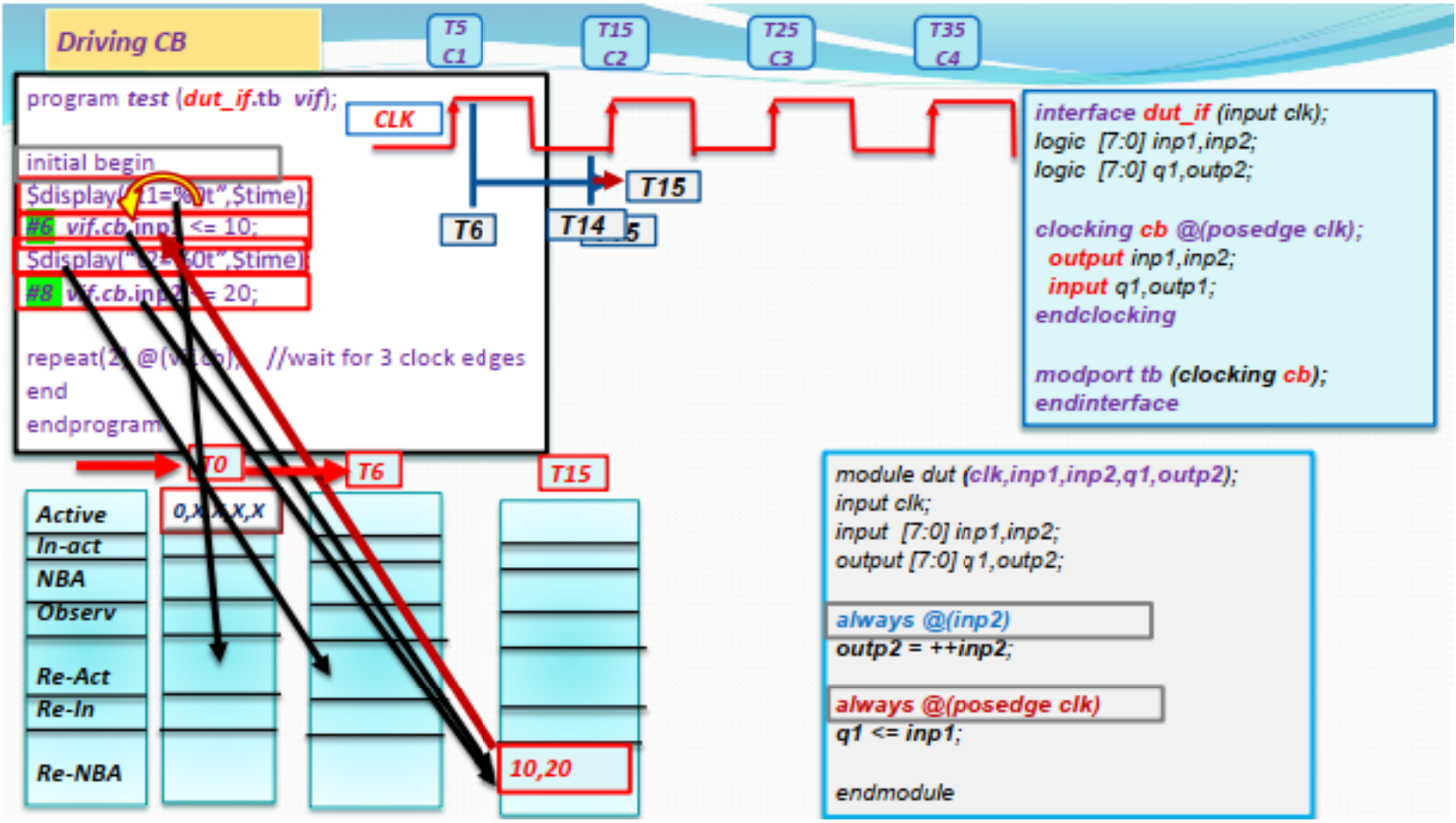
T5 T4 T5

```
interface dut_if (input clk);
logic  [7:0] inp1,inp2;
logic  [7:0] q1,outp2;

clocking cb @(posedge clk);
  output inp1,inp2;
  input q1,outp1;
endclocking

modport tb (clocking cb);
endinterface
```




```
module dut (clk,inp1,inp2,q1,outp2);
input clk;
input  [7:0] inp1,inp2;
output [7:0] q1,outp2;

always @(inp2)
outp2 = ++inp2;

always @(posedge clk)
q1 <= inp1;

endmodule
```

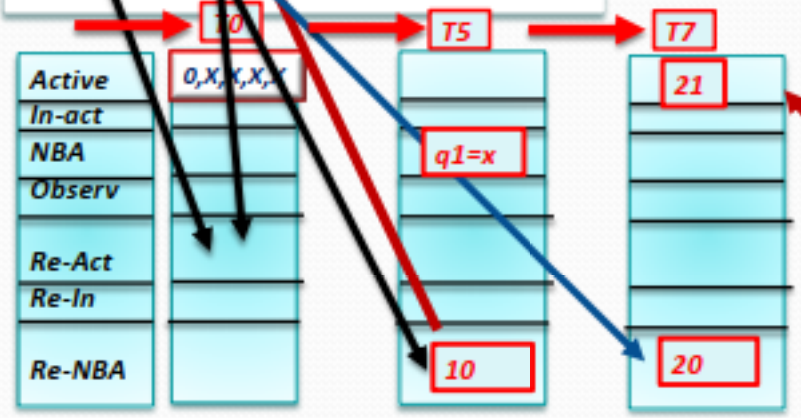
Driving CB


T5
C1
T15
C2
T25
C3
T35
C4
CLK
T6
T14
T15
program test (dut_if.tb vif);
initial begin
$display("1=%0t",$time);
#6 vif.cb.inp1 <= 10;
$display("2=%0t",$time);
#8 vif.cb.inp2 <= 20;
repeat(2) @(vif.cb); //wait for 3 clock edges
end
endprogram
interface dut_if (input clk);
logic [7:0] inp1,inp2;
logic [7:0] q1,outp2;
clocking cb @(posedge clk);
output inp1,inp2;
input q1,outp1;
endclocking
modport tb (clocking cb);
endinterface
T0
T6
T15
Active
In-act
NBA
Observ
Re-Act
Re-In
Re-NBA
0,X,X,X,X
10,20
module dut (clk,inp1,inp2,q1,outp2);
input clk;
input [7:0] inp1,inp2;
output [7:0] q1,outp2;
always @(inp2)
outp2 = ++inp2;
always @(posedge clk)
q1 <= inp1;
endmodule

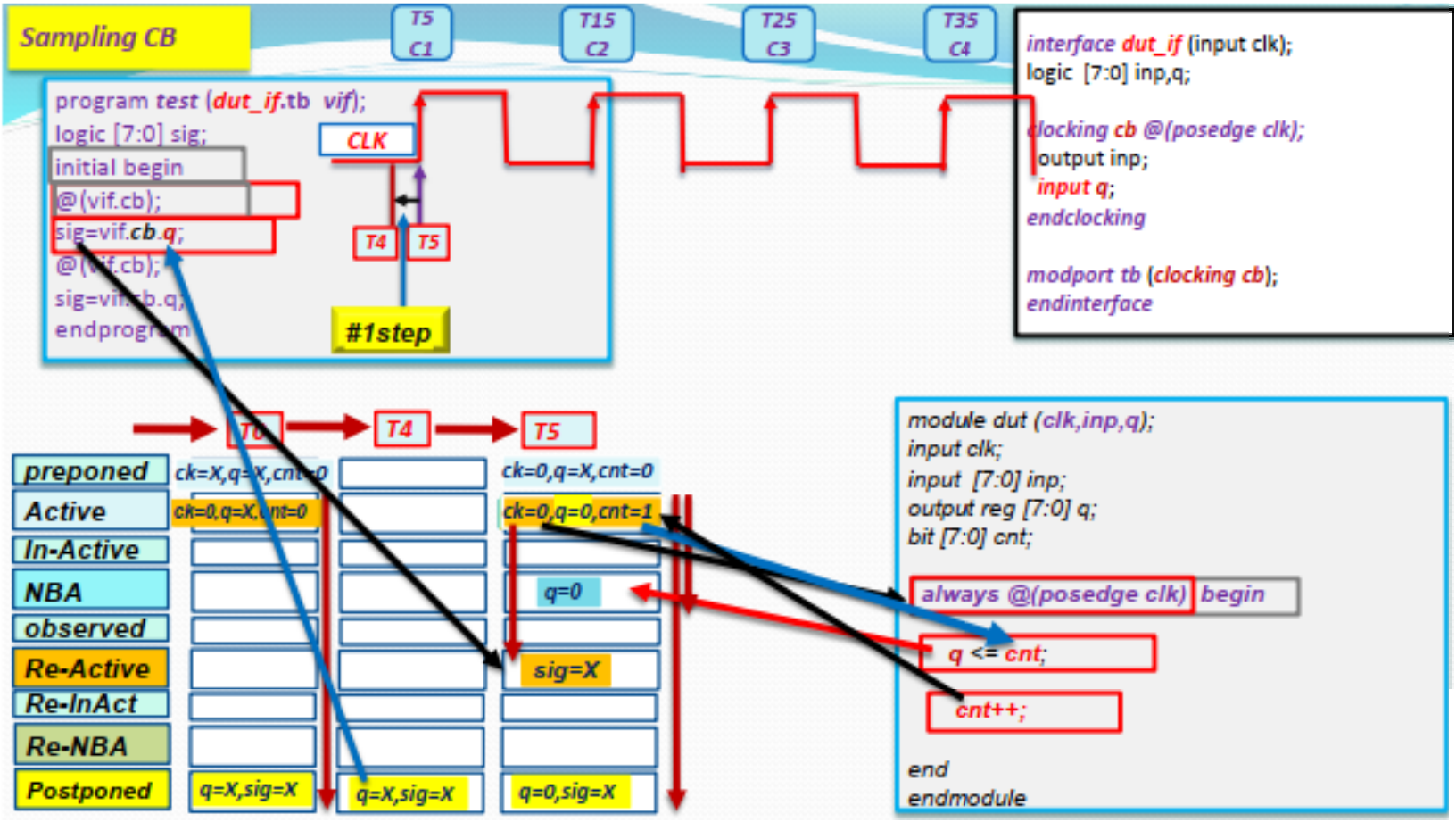
Driving CB with output #skew

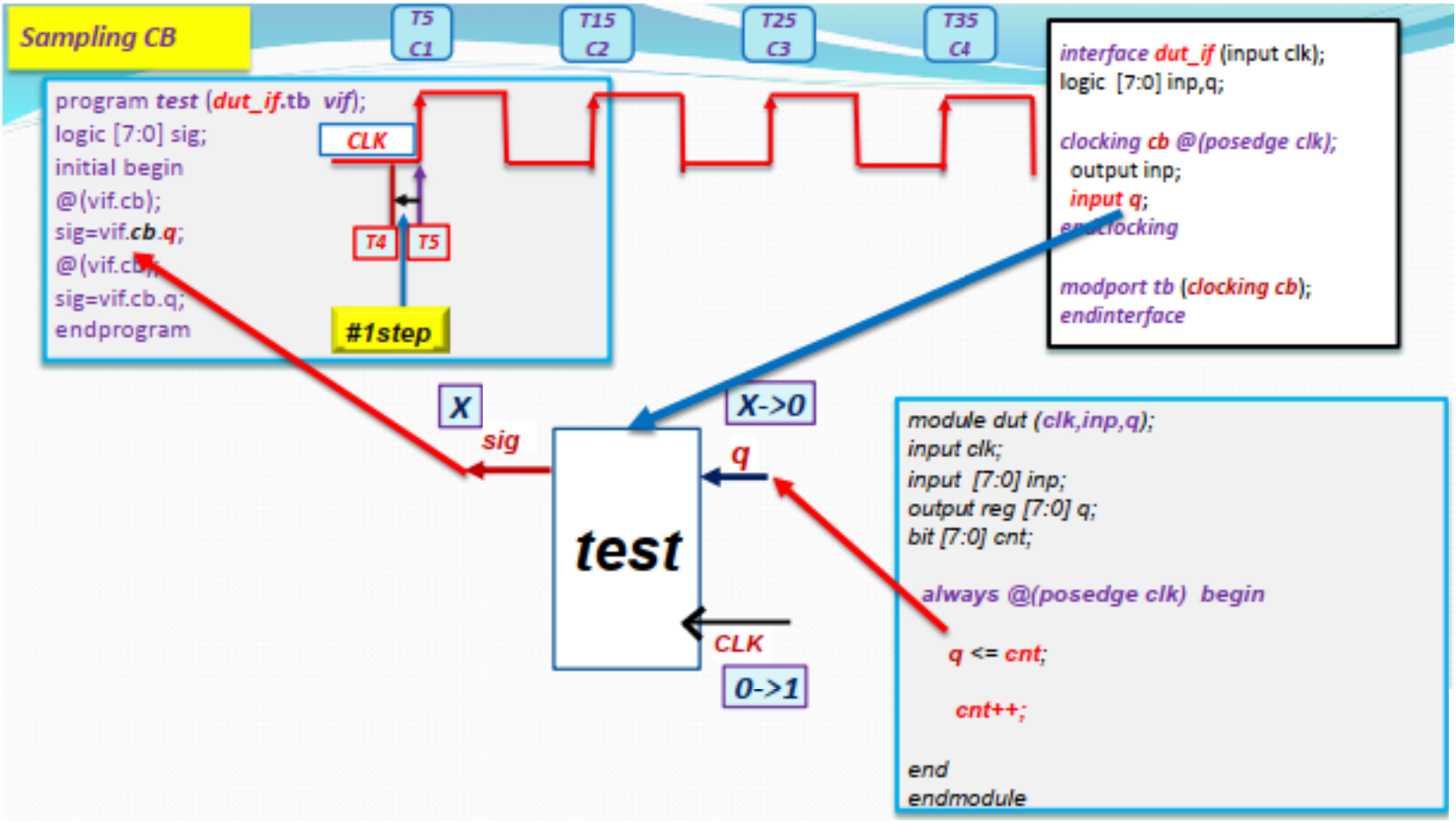
```
program test (dut_if.tb  vif);

initial begin
$display("t1=%0t",$time);
vif.cb.inp1 <= 10;
$display("t2=%0t",$time);
vif.cb.inp2 <= 20;

repeat(2) @(vif.cb);    //wait for 3 clock edges
end
endprogram
```

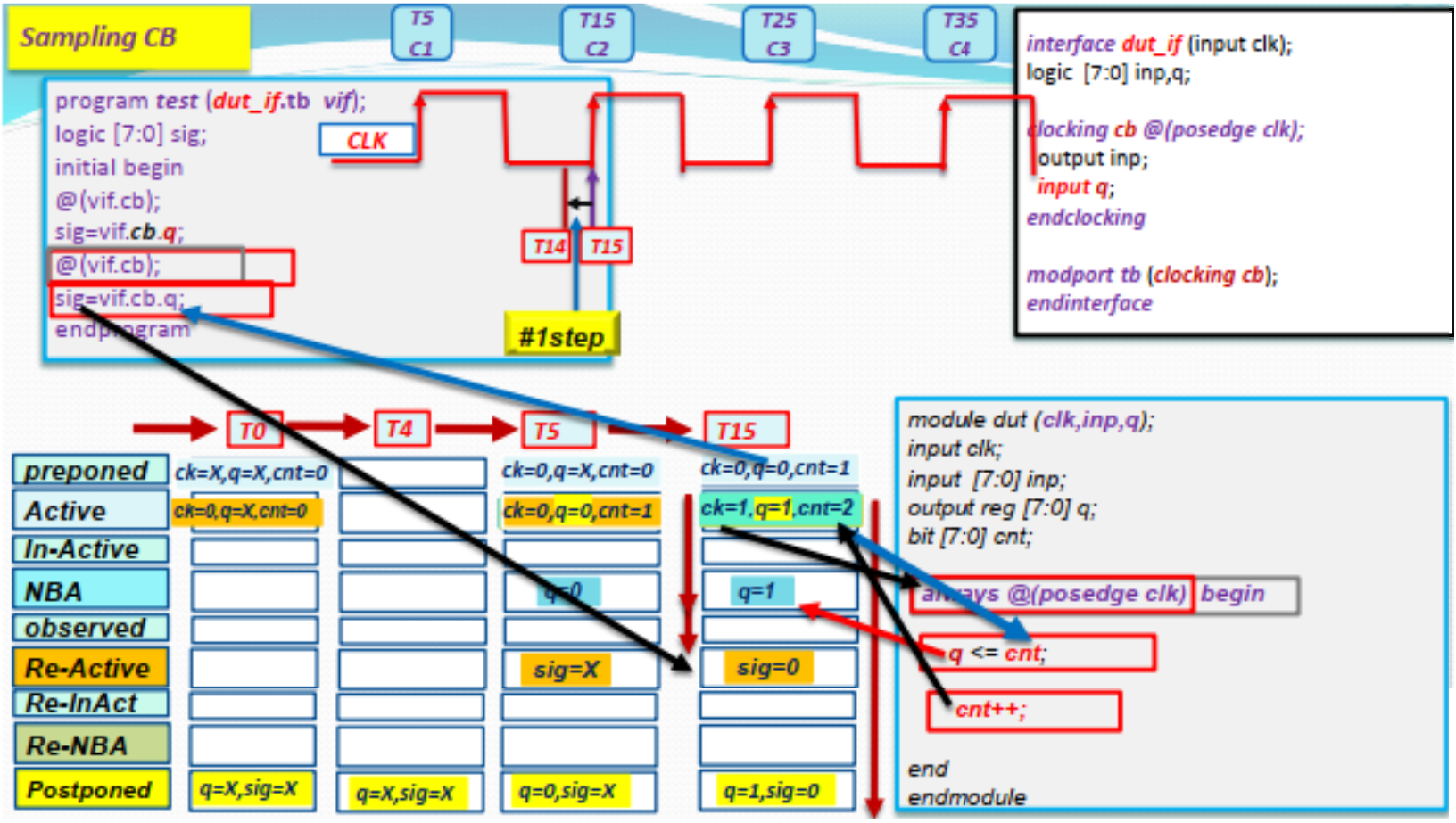

```
interface dut_if (input clk);
logic  [7:0] inp1,inp2;
logic  [7:0] q1,outp2;

clocking cb @(posedge clk);
  output #2 inp2,
  output  inp1;
  input q1,outp1;
endclocking

modport tb (clocking cb);
endinterface
```

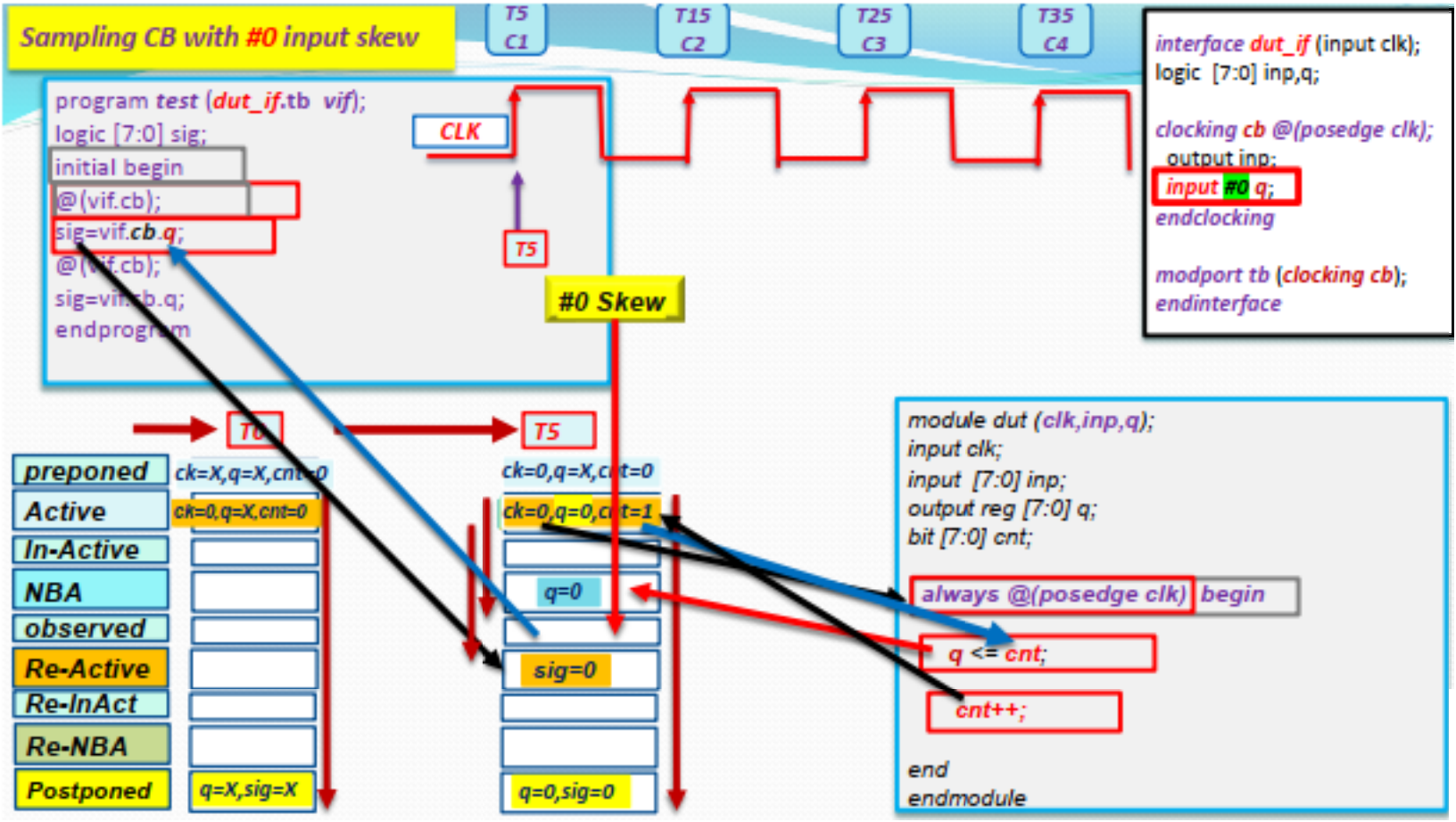


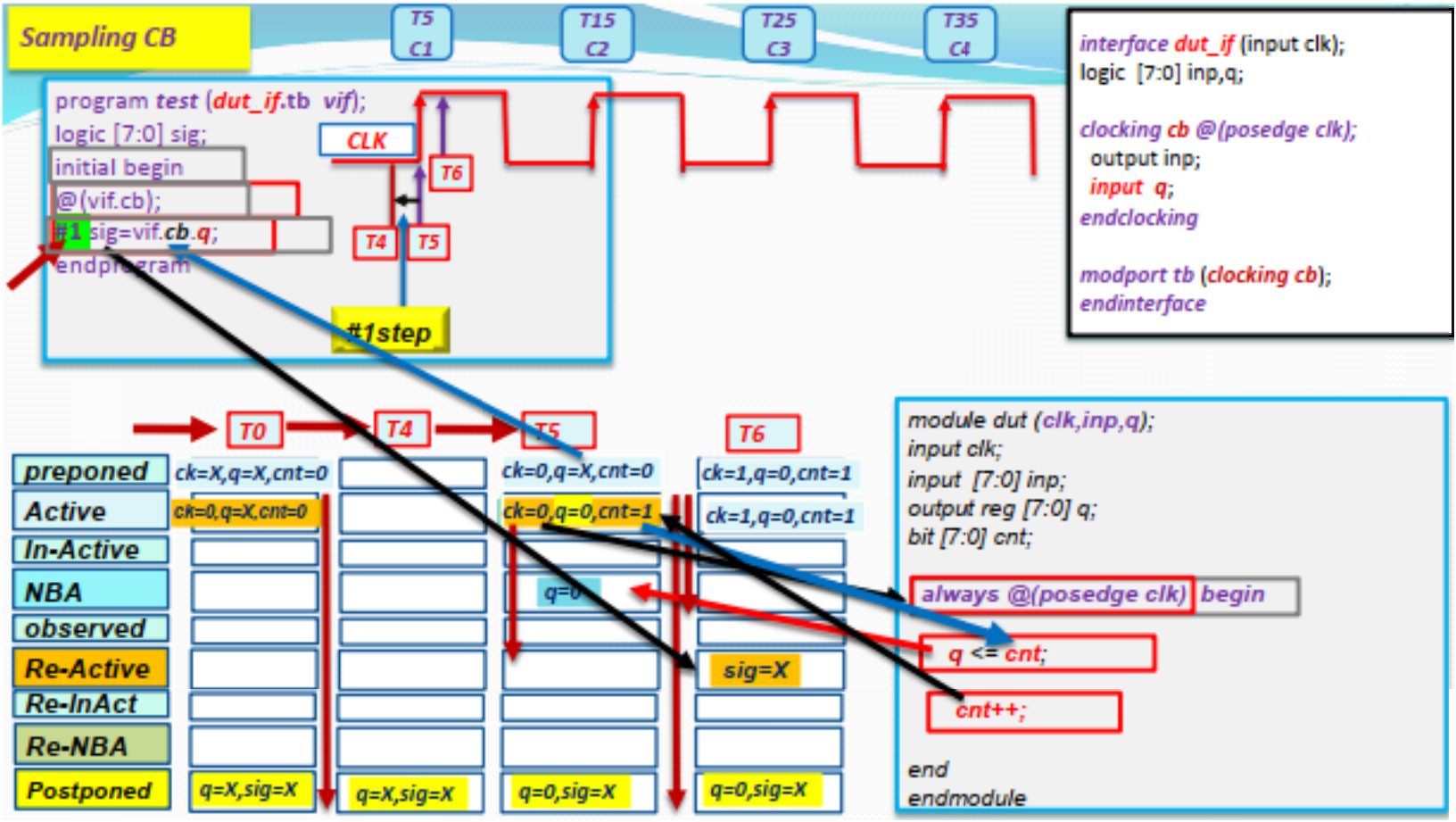

```
module dut (clk,inp1,inp2,q1,outp2);
input clk;
input  [7:0] inp1,inp2;
output [7:0] q1,outp2;

always @(inp2)
outp2 = ++inp2;

always @(posedge clk)
q1 <= inp1;

endmodule
```

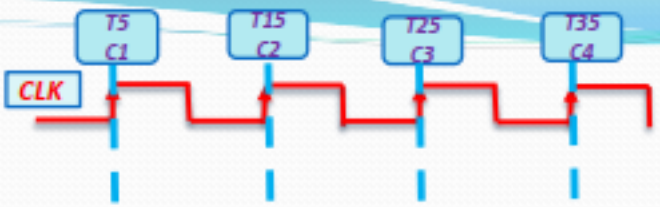

Sampling CB
T5 C1
T15 C2
T25 C3
T35 C4
program test (dut_if.tb vif);
logic [7:0] sig;
initial begin
@(vif.cb);
sig=vif.cb.q;
@(vif.cb);
sig=vif.cb.q;
endprogram
CLK
T4
T5
#1step
interface dut_if (input clk);
logic [7:0] inp,q;
clocking cb @(posedge clk);
output inp;
input q;
endclocking
modport tb (clocking cb);
endinterface
T6
T4
T5
preponed
Active
In-Active
NBA
observed
Re-Active
Re-InAct
Re-NBA
Postponed
ck=X,q=X,cnt=0
ck=0,q=X,cnt=0
q=X,sig=X
q=X,sig=X
ck=0,q=X,cnt=0
ck=0,q=0,cnt=1
q=0
sig=X
q=0,sig=X
module dut (clk,inp,q);
input clk;
input [7:0] inp;
output reg [7:0] q;
bit [7:0] cnt;
always @(posedge clk) begin
q <= cnt;
cnt++;
end
endmodule

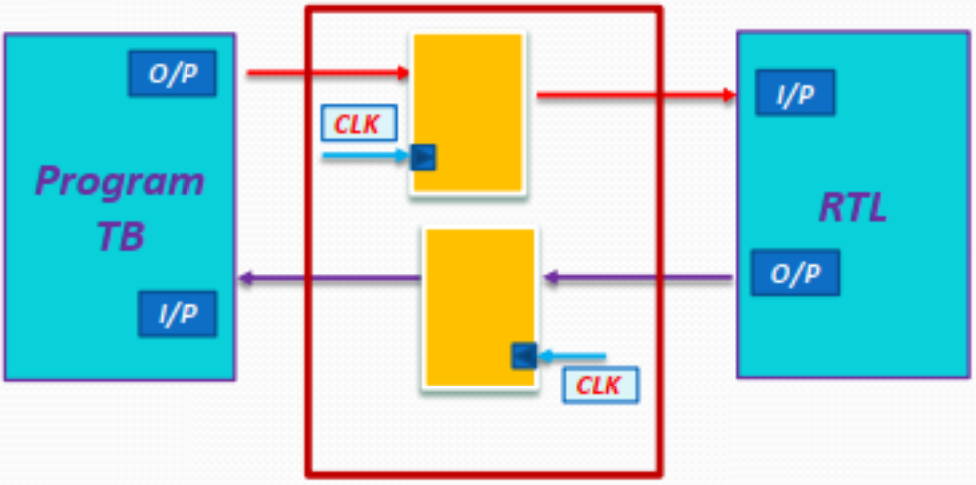

Sampling CB
T5
C1
T15
C2
T25
C3
T35
C4
program test (dut_if.tb vif);
logic [7:0] sig;
initial begin
@(vif.cb);
sig=vif.cb.q;
@(vif.cb);
sig=vif.cb.q;
endprogram
CLK
T4
T5
#1step
interface dut_if (input clk);
logic [7:0] inp,q;
clocking cb @(posedge clk);
output inp;
input q;
endclocking
modport tb (clocking cb);
endinterface
X
sig
X->0
q
test
CLK
0->1
module dut (clk,inp,q);
input clk;
input [7:0] inp;
output reg [7:0] q;
bit [7:0] cnt;
always @(posedge clk) begin
q <= cnt;
cnt++;
end
endmodule


Sampling CB
T5 C1
T15 C2
T25 C3
T35 C4
program test (dut_if.tb vif);
logic [7:0] sig;
initial begin
@(vif.cb);
sig=vif.cb.q;
@(vif.cb);
sig=vif.cb.q;
endprogram
CLK
T14
T15
#1step
interface dut_if (input clk);
logic [7:0] inp,q;
clocking cb @(posedge clk);
output inp;
input q;
endclocking
modport tb (clocking cb);
endinterface
T0
T4
T5
T15
preponed
Active
In-Active
NBA
observed
Re-Active
Re-InAct
Re-NBA
Postponed
ck=X,q=X,cnt=0
ck=0,q=X,cnt=0
ck=0,q=0,cnt=1
ck=1,q=1,cnt=2
q=0
q=1
sig=X
sig=0
q=X,sig=X
q=X,sig=X
q=0,sig=X
q=1,sig=0
module dut (clk,inp,q);
input clk;
input [7:0] inp;
output reg [7:0] q;
bit [7:0] cnt;
always @(posedge clk) begin
q <= cnt;
cnt++;
end
endmodule


Sampling CB with #0 input skew
T5 C1
T15 C2
T25 C3
T35 C4
program test (dut_if.tb vif);
logic [7:0] sig;
initial begin
@(vif.cb);
sig=vif.cb.q;
@(vif.cb);
sig=vif.cb.q;
endprogram
CLK
T5
#0 Skew
interface dut_if (input clk);
logic [7:0] inp,q;
clocking cb @(posedge clk);
output inp;
input #0 q;
endclocking
modport tb (clocking cb);
endinterface
T0
T5
preponed
Active
In-Active
NBA
observed
Re-Active
Re-InAct
Re-NBA
Postponed
ck=X,q=X,cnt=0
ck=0,q=X,cnt=0
q=X,sig=X
ck=0,q=X,cnt=0
ck=0,q=0,cnt=1
q=0
sig=0
q=0,sig=0
module dut (clk,inp,q);
input clk;
input [7:0] inp;
output reg [7:0] q;
bit [7:0] cnt;
always @(posedge clk) begin
q <= cnt;
cnt++;
end
endmodule


Sampling CB
T5 C1
T15 C2
T25 C3
T35 C4
program test (dut_if.tb vif);
logic [7:0] sig;
initial begin
@(vif.cb);
#1 sig=vif.cb.q;
endprogram
CLK
T6
T4
T5
#1step
interface dut_if (input clk);
logic [7:0] inp,q;
clocking cb @(posedge clk);
output inp;
input q;
endclocking
modport tb (clocking cb);
endinterface
T0
T4
T5
T6
preponed
Active
In-Active
NBA
observed
Re-Active
Re-InAct
Re-NBA
Postponed
ck=X,q=X,cnt=0
ck=0,q=X,cnt=0
ck=0,q=X,cnt=0
ck=0,q=0,cnt=1
ck=1,q=0,cnt=1
ck=1,q=0,cnt=1
q=0
sig=X
q=X,sig=X
q=X,sig=X
q=0,sig=X
q=0,sig=X
module dut (clk,inp,q);
input clk;
input [7:0] inp;
output reg [7:0] q;
bit [7:0] cnt;
always @(posedge clk) begin
q <= cnt;
cnt++;
end
endmodule


T5
C1
T15
C2
T25
C3
T35
C4
CLK


O/P
Program
TB
I/P
CLK
CLK
I/P
RTL
O/P

Default clocking block

- One clocking block can be specified as the default for all cycle delay operations.

```
program test(simple_bus.tb intf);
default clocking bus @(posedge intf.clk);
endclocking

initial begin
## 5;
intf.cb.addr <= 10;
intf.cb.rd <= 1;
## 1; //Wait for 1 clock cycle =  @(posedge intf.clk);

end
endprogram
```

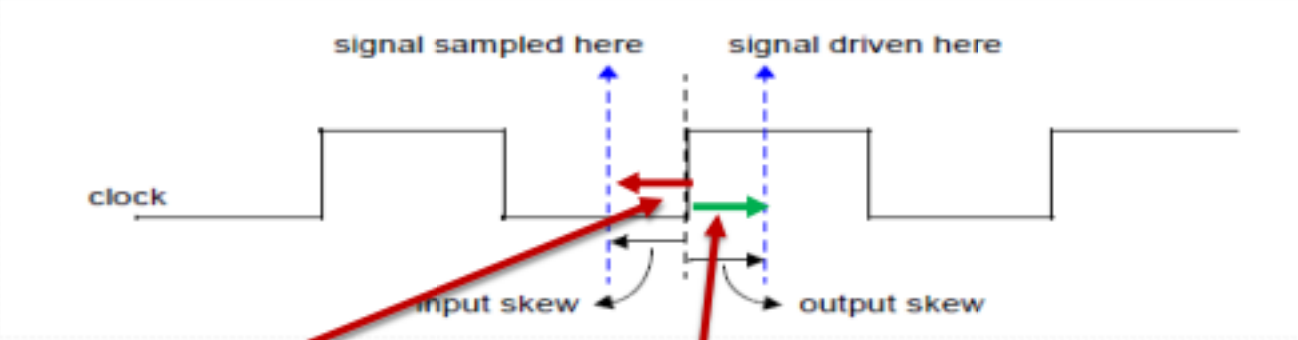
Wait for 5 clock cycles
repeat(5) @(posedge intf.clk);

Synchronous Drives

```
// drive data in Re-NBA region of the current cycle
vif.bus.data[3:0] <= 4'h5;

// wait for 2 default clocking cycles, then drive data
##2;
vif.bus.data <= 2;

// remember the value of r and then drive  data 2 (bus) cycles later
vif.bus.data <= ##2 r;

// Error: regular intra-assignment delay not allowed in synchronous drives
vif.bus.data <= #4 r;
```

```
interface dut_if (input clk);
logic [7:0] data;

default clocking bus @(posedge clk);
output data;
endclocking

endinterface
```

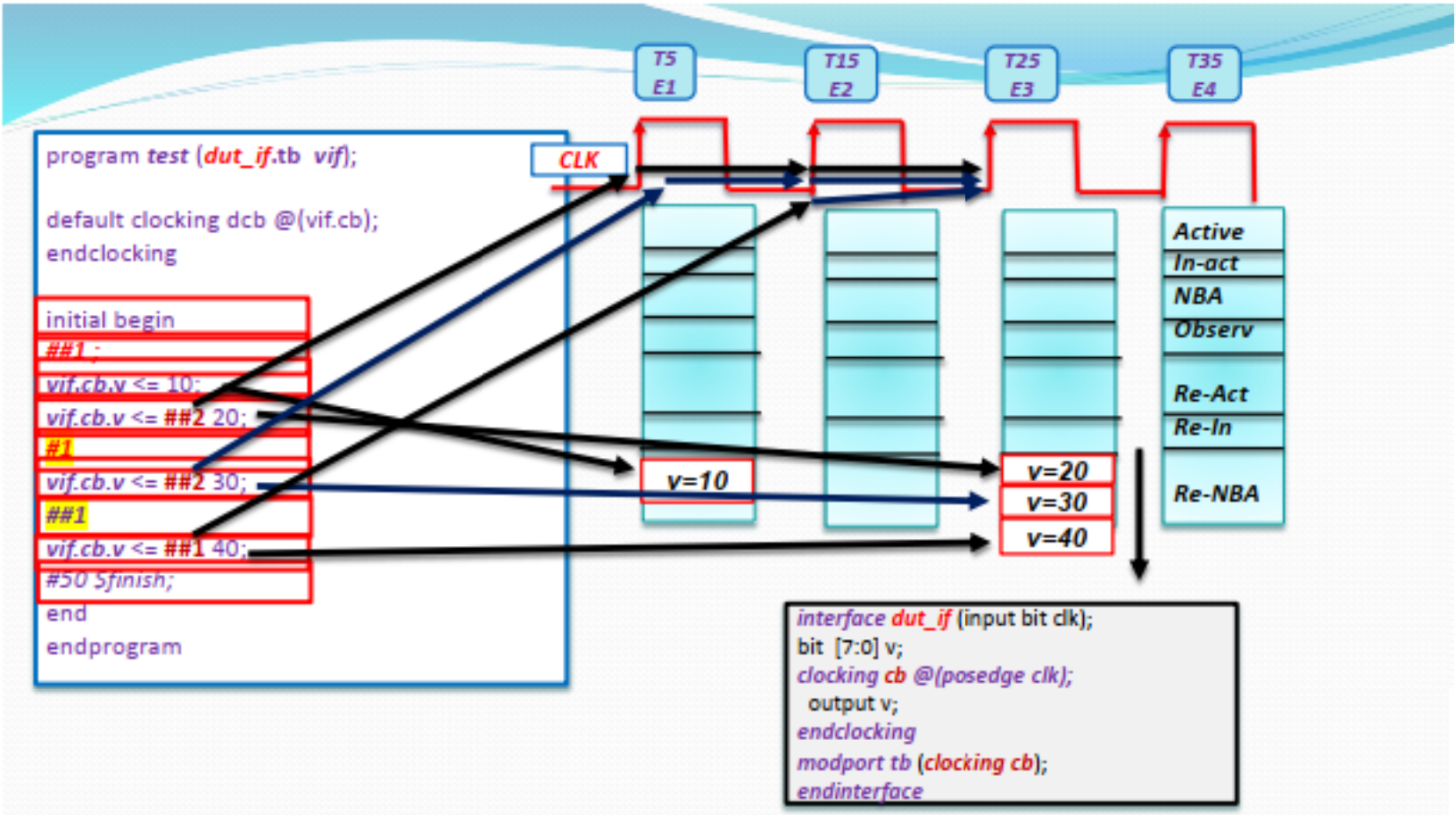
Input and output skews



- *If an input skew is specified, then the signal is sampled at skew time units before the clock event.*
- *If an output skew is specified, then output (or inout) signals are driven skew simulation time units after the corresponding clock event.*
- ***Default input skew is #1step***
- ***Default output skew is #0***

Clocking block skews

```
clocking cb_mem @(posedge clk);
input  #1     rdata;
output #2    wdata;
output       addr,wr;
endclocking

clocking cb_mem @(posedge clk);
default input #1ns  output #2ns
input     rdata;
output  wdata;
output  addr,wr;
endclocking
```


T5
E1
T15
E2
T25
E3
T35
E4
program test (dut_if.tb vif);
default clocking dcb @(vif.cb);
endclocking
initial begin
##1 ;
vif.cb.v <= 10;
vif.cb.v <= ##2 20;
#1
vif.cb.v <= ##2 30;
##1
vif.cb.v <= ##1 40;
#50 $finish;
end
endprogram
CLK
v=10
v=20
v=30
v=40
Active
In-act
NBA
Observ
Re-Act
Re-In
Re-NBA
interface dut_if (input bit clk);
bit [7:0] v;
clocking cb @(posedge clk);
output v;
endclocking
modport tb (clocking cb);
endinterface

Thank You