Verilog

Behavioral Description

- **initial:**
  - is executed once at the beginning.
- **always:**
  - is repeated until the end of simulation.

Clock Generation

- Two methods:
  - 15 cycles of clock.
  - clock period = 20 time units.

```
initial
   begin
      clock = 1’b0;
      repeat (30)
      #10 clock = ~clock;
   end
```

```
initial
   begin
      clock = 1’b0;
      #300 $finish;
   end

always
   #10 clock = ~clock;
```

? Description

- Output Q must be declared as **`reg`**.
- Inside **`initial`** and **`always`**, **`if-else`** and **`case`** statements can be used.

```
//HDL Example 5-1
//--------------------------------------
//Description of xxxx (See Fig.5-6)
module xxxx (Q,D,control);
   output Q;
   input D,control;
   reg Q;
   always @ (control or D)
      if (control == 1) Q = D;  //Same as: if (control)
endmodule
```

D-Latch Description

- Output Q must be declared as **`reg`**.
- Inside **`initial`** and **`always`**, **`if-else`** and **`case`** statements can be used.

```
//HDL Example 5-1
//--------------------------------------
//Description of D latch (See Fig.5-6)
module D_latch (Q,D,control);
   output Q;
   input D,control;
   reg Q;
   always @ (control or D)
      if (control == 1) Q = D;  //Same as: if (control)
endmodule
```

D Flip-Flop Description

```
//HDL Example 5-2
//---------------------------
//D flip-flop
module D_FF (Q,D,CLK);
   output Q;
   input D,CLK;
   reg Q;
   always @ (posedge CLK)
     Q = D;
endmodule
```

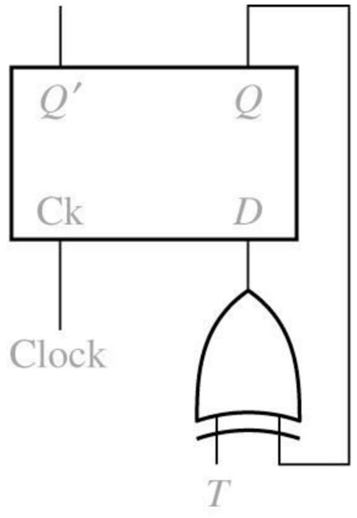
// D Flip-Flop Description

```
//HDL Example 5-2
//-----------------------------
//D flip-flop
module D_FF (Q,D,CLK);
   output Q;
   input D,CLK;
   reg Q;
   always @ (posedge CLK)
     Q = D;
endmodule
```

```
//D flip-flop with              reset.
module DFF (Q,D,CLK,RST);
   output Q;
   input D,CLK,RST;
   reg Q;
   always @(posedge CLK or negedge RST)
     if (RST == 0) Q = 1'b0;    // Same as: if (~RST)
     else Q = D;
endmodule
```

T-FF Description (Structural)

```
//HDL Example 5-3
//-----------------------------------------
//T flip-flop from D flip-flop and gates
module TFF (Q,T,CLK,RST);
   output Q;
   input T,CLK,RST;
   wire DT;
   assign DT = Q ^ T ;
//Instantiate the D flip-flop
   DFF TF1 (Q,DT,CLK,RST);
endmodule
```



(b) Conversion of D to T

JK-FF Description (Structural)

```
//JK flip-flop from D flip-flop and gates
module JKFF (Q,J,K,CLK,RST);
   output Q;
   input J,K,CLK,RST;
   wire JK;
   assign JK = (J & ~Q) | (~K & Q);
//Instantiate D flipflop
   DFF JK1 (Q,JK,CLK,RST);
endmodule
```

JK-FF Description (Behavioral)

- `case` executes one of the statements.

```
//HDL Example 5-4
//----------------------------------
// Functional description of JK flip-flop
module JK_FF (J,K,CLK,Q,Qnot);
   output Q,Qnot;
   input  J,K,CLK;
   reg  Q;
   assign Qnot = ~ Q ;
   always @  (posedge CLK)
           case ({J,K})
             2'b00: Q = Q;
             2'b01: Q = 1'b0;
             2'b10: Q = 1'b1;
             2'b11: Q = ~ Q;
           endcase
endmodule
```

A Sequential Circuit: Schematic Diagram

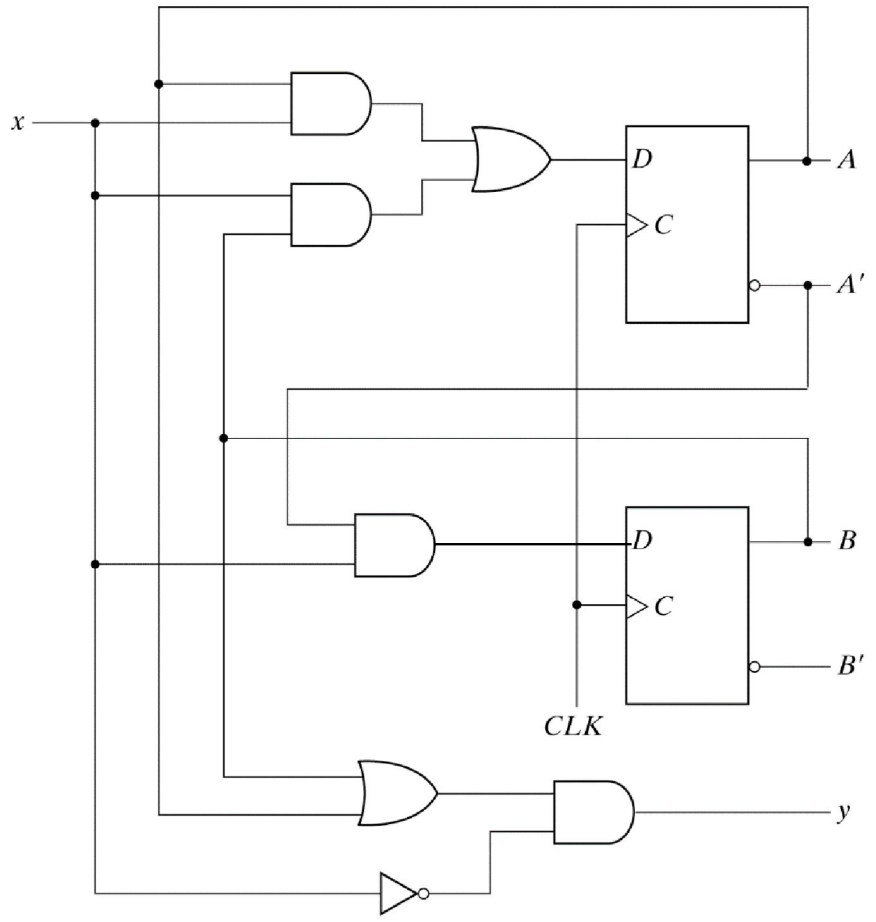


Fig. 5-15 Example of Sequential Circuit

State Diagram (Behavioral)

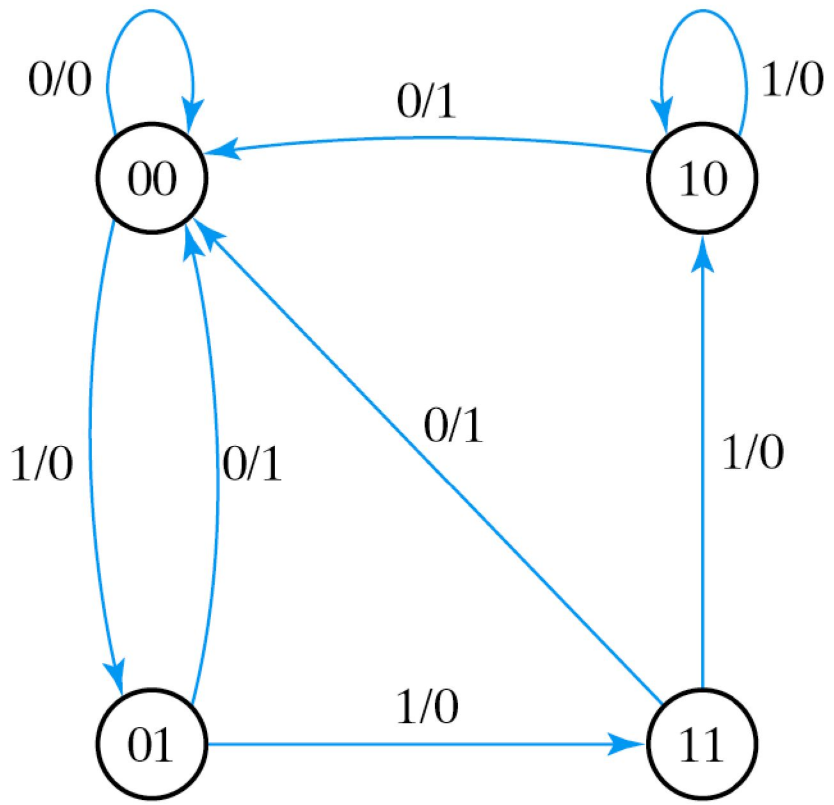


Fig. 5-16 State Diagram of the Circuit of Fig. 5-15

Mealy (Behavioral)

- `parameter` defines constants.

```
module Mealy_mdl (x,y,CLK,RST);
  input x,CLK,RST;
  output y;
  reg y;
  reg [1:0] Prstate, Nxtstate;
  parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;
   always @ (posedge CLK or negedge RST)
      if (~RST) Prstate = S0;  //Initializ
      else Prstate = Nxtstate; //Clock ope
   always @ (Prstate or x)     //Determine
         case (Prstate)
            S0: if (x) Nxtstate = S1;
                   else Nxtstate = S0;
            S1: if (x) Nxtstate = S3;
                   else Nxtstate = S0;
            S2: if (~x)Nxtstate = S0;
                   else Nxtstate = S2;
            S3: if (x) Nxtstate = S2;
                   else Nxtstate = S0;
         endcase
   always @ (Prstate or x)     //Evaluate output
         case (Prstate)
            S0: y = 0;
            S1: if (x) y = 1'b0; else y = 1'b1;
            S2: if (x) y = 1'b0; else y = 1'b1;
            S3: if (x) y = 1'b0; else y = 1'b1;
         endcase
```

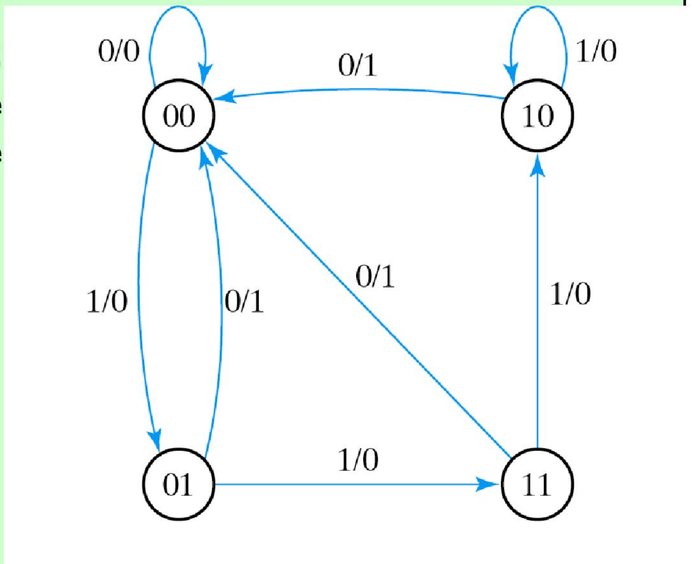
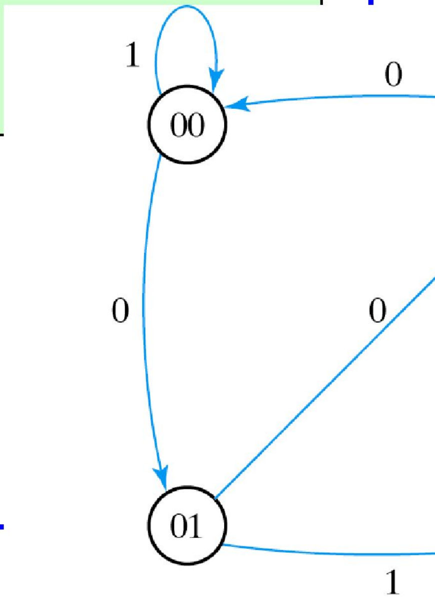


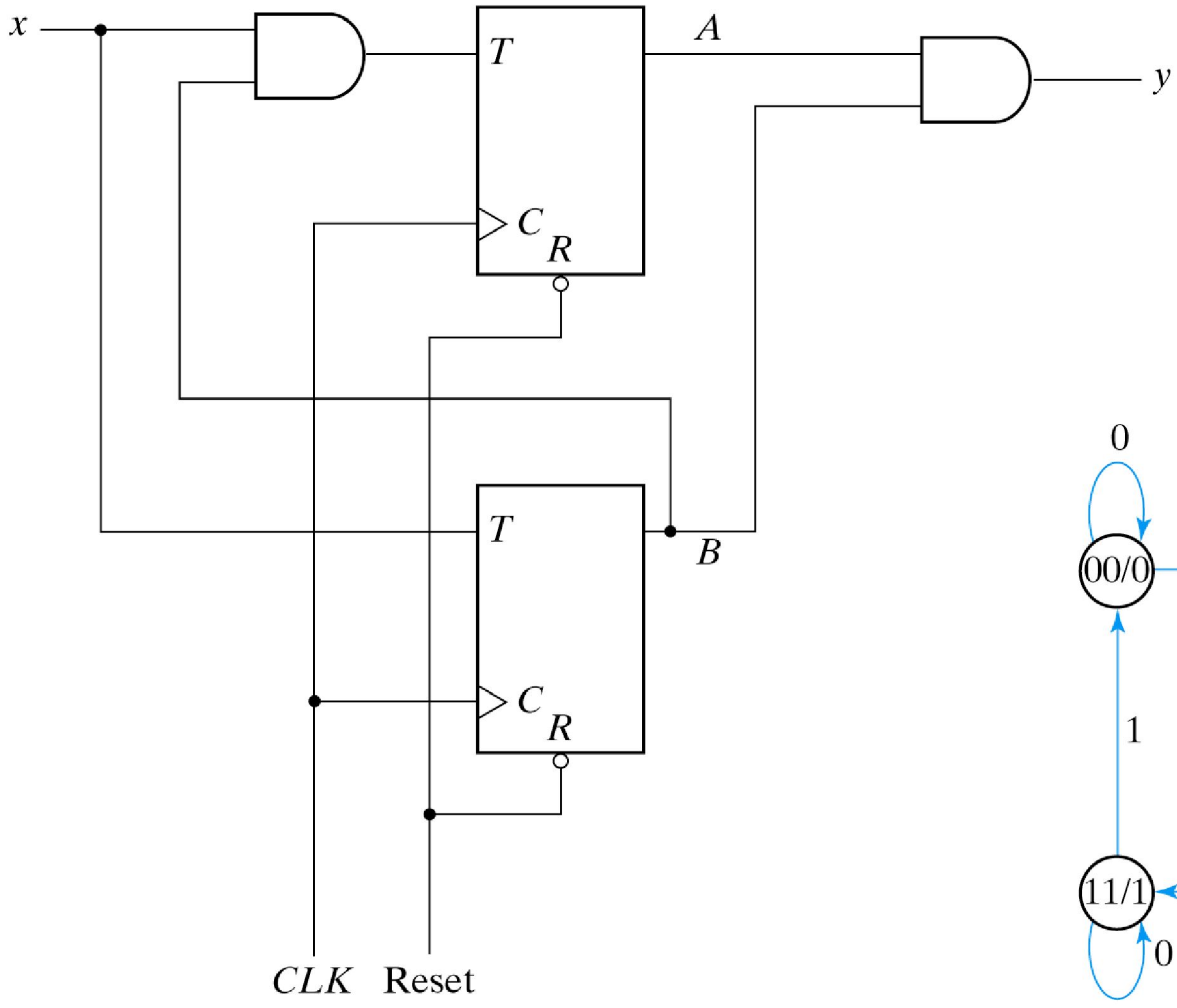
Fig. 5-16 State Diagram of the Circuit of Fig. 5-15

Moore (Behavioral) (One `always`)

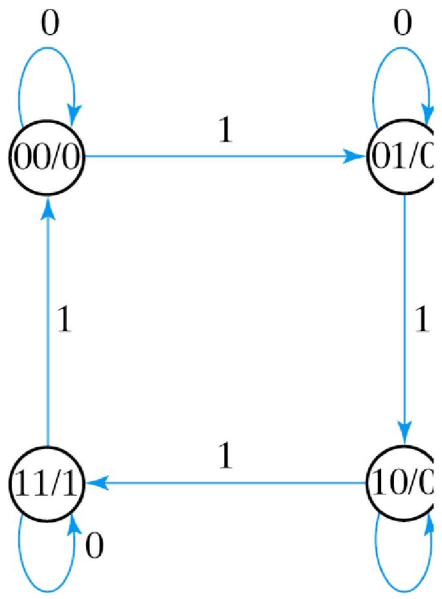
```
module Moore_mdl (x,AB,CLK,RST);
   input x,CLK,RST;
   output [1:0]AB;
   reg [1:0] state;
   parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;
      always @ (posedge CLK or negedge RST)
         if (~RST) state = S0;  //Initialize to state S0
             else
case (state)
           S0: if (~x) state = S1; else state = S0;
           S1: if (x)  state = S2; else state = S3;
           S2: if (~x) state = S3; else state = S2;
           S3: if (~x) state = S0; else state = S3;
         endcase
   assign AB = state;        //Output of flip-flops
endmodule
```



Moore (Structural)



(a) Circuit diagram



(b) State diagram

Moore (Structural)

```
module Tcircuit (x,y,A,B,CLK,RST);
   input x,CLK,RST;
   output y,A,B;
   wire TA,TB;
//Flip-flip input equations
   assign TB = x,
          TA = x & B;
//Output equation
   assign y = A & B;
//Instantiate T flip-flops
   T_FF BF (B,TB,CLK,RST);
   T_FF AF (A,TA,CLK,RST);
endmodule
```

```
//T flip-flop
module T_FF (Q,T,CLK,RST);
   output Q;
   input T,CLK,RST;
   reg Q;
     always @ (posedge CLK or negedge RST)
       if (~RST) Q = 1'b0;
       else Q = Q ^ T;
endmodule
```

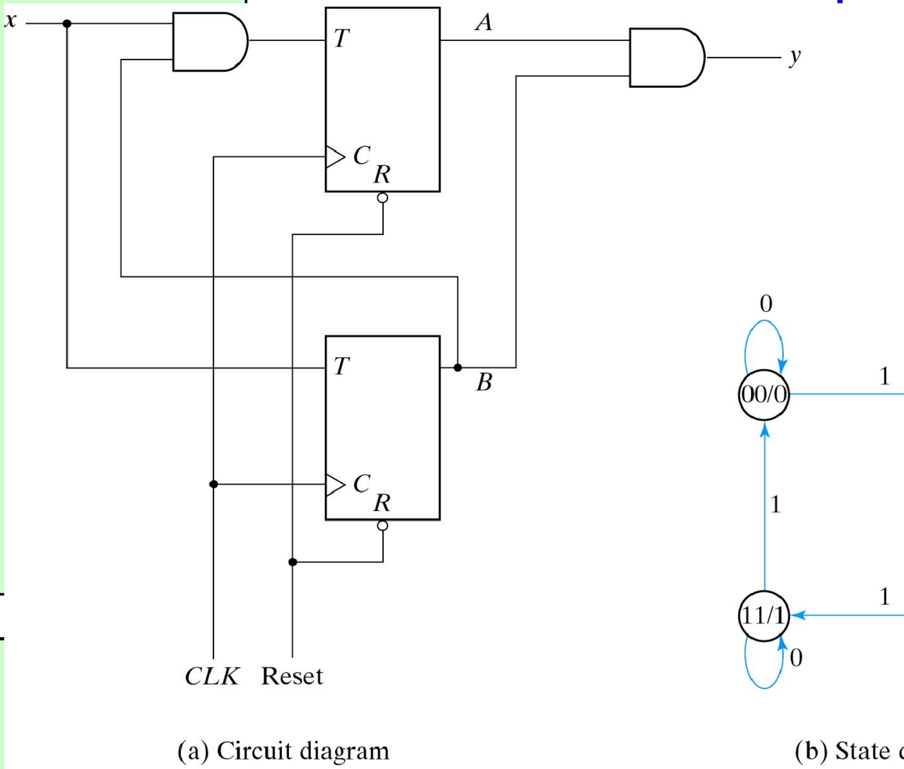


Fig. 5-20 Sequential Circuit with *T* Flip-Flops

Testbench

```
//Stimulus for testing sequential circuit
module testTcircuit;
  reg x,CLK,RST;  //inputs for circuit
  wire y,A,B;    //output from circuit
  Tcircuit TC (x,y,A,B,CLK,RST);  // instantiate circuit
  initial
     begin
         RST = 0;
         CLK = 0;
      #5 RST = 1;
         repeat (16)
      #5 CLK = ~CLK;
     end
  initial
     begin
          x = 0;
      #15 x = 1;
          repeat (8)
      #10 x = ~ x;
     end
endmodule
```

Waveforms

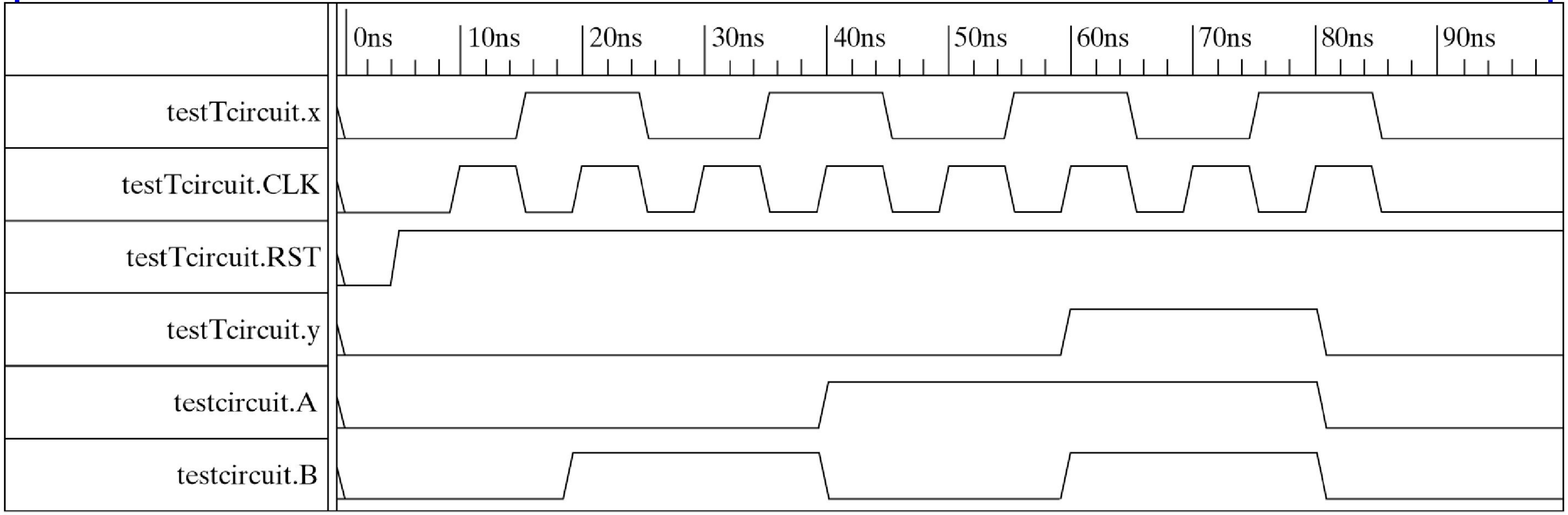


Fig. 5-21 Simulation Output of HDL Example 5-7