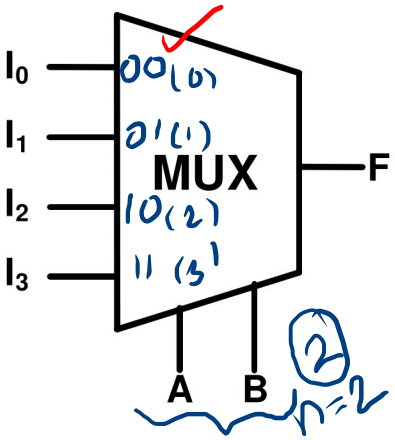
Figure 2-9 4-to-1 Multiplexer




```
F <= (not A and not B and I0) or
     (not A and B and I1) or
     (A and not B and I2) or
     (A and B and I3);
```

MUX model using a *conditional signal assignment statement*:

```
F <= I0 when Sel = 0
     else I1 when Sel = 1
     else I2 when Sel = 2
     else I3;
```

In the above concurrent statement, Sel represents the integer equivalent of a 2-bit binary number with bits A and B.

General form of conditional signal assignment statement:

```
signal_name <= expression1 when condition1
     else expression2 when condition2
     ...
     [else expressionN];
```

Multiplexer Example From Page 55

If a MUX model is used inside a process, a concurrent statement cannot be used. As an alternative, the MUX can be modeled using a **case statement**:

```
case Sel is
      when 0 => F <= I0;
      when 1 => F <= I1;
      when 2 => F <= I2;
      when 3 => F <= I3;
end case;
```

The case statement has the general form:

```
case expression is
      when choice1 => sequential statements1
      when choice2 => sequential statements2
      . . .
      [when others => sequential statements]
end case;
```

Figure 2-10 Compilation, Elaboration, and Simulation of VHDL Code

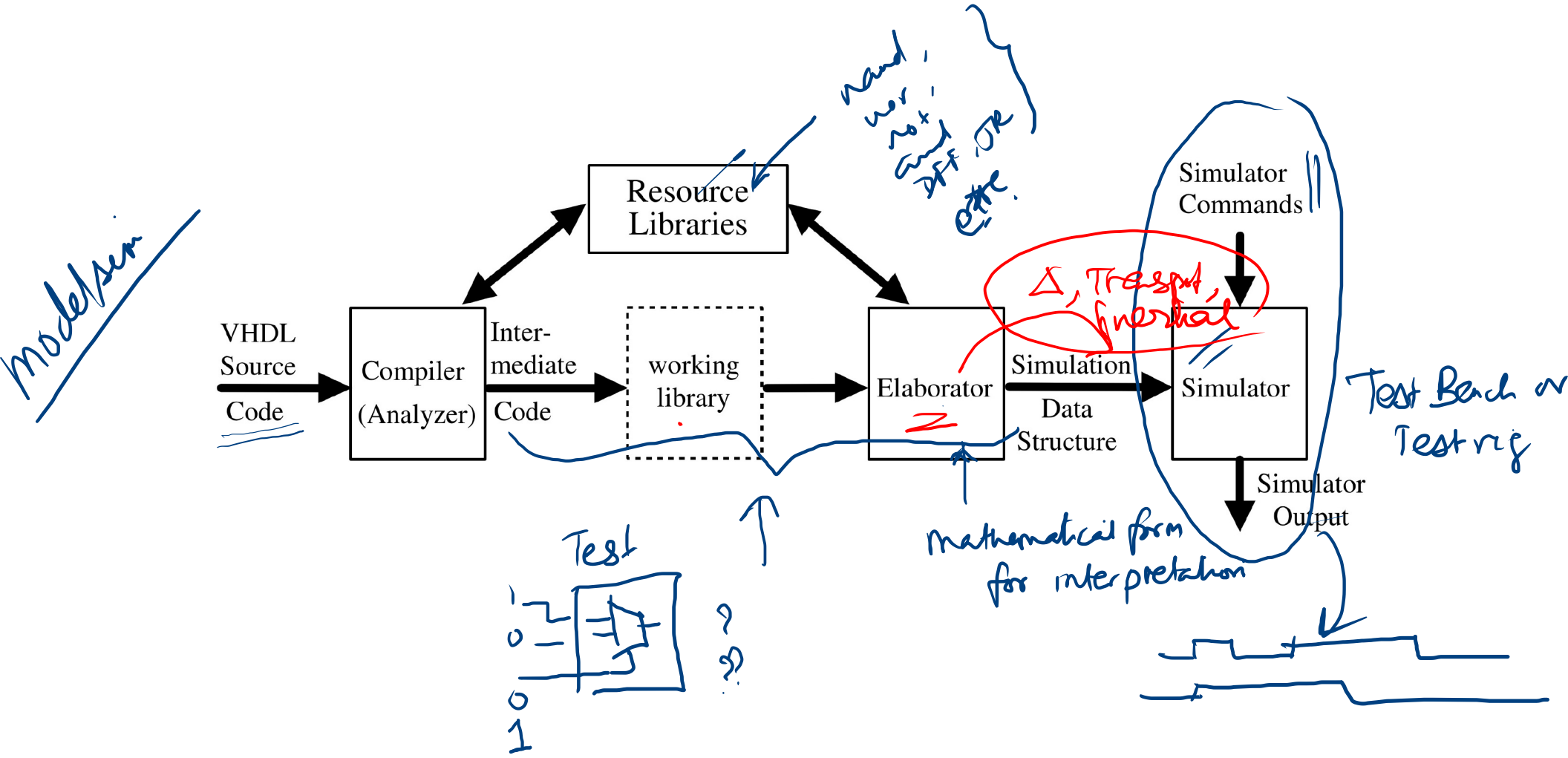


Figure 2-11 VHDL Code for Simulation Example

```
entity simulation_example is
end simulation_example;

architecture test1 of simulation_example is
  signal A,B: bit;
begin
  P1: process(B)
  begin
    A <= '1';
    A <= transport '0' after 5 ns;
  end process P1;

  P2: process(A)
  begin
    if A = '1' then B <= not B after 10 ns; end if;
  end process P2;
end test1;
```

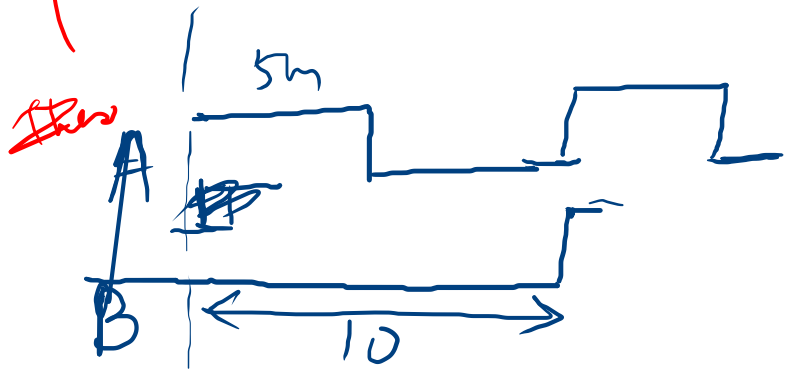


Figure 2-12 Signal Drivers for Simulation Example

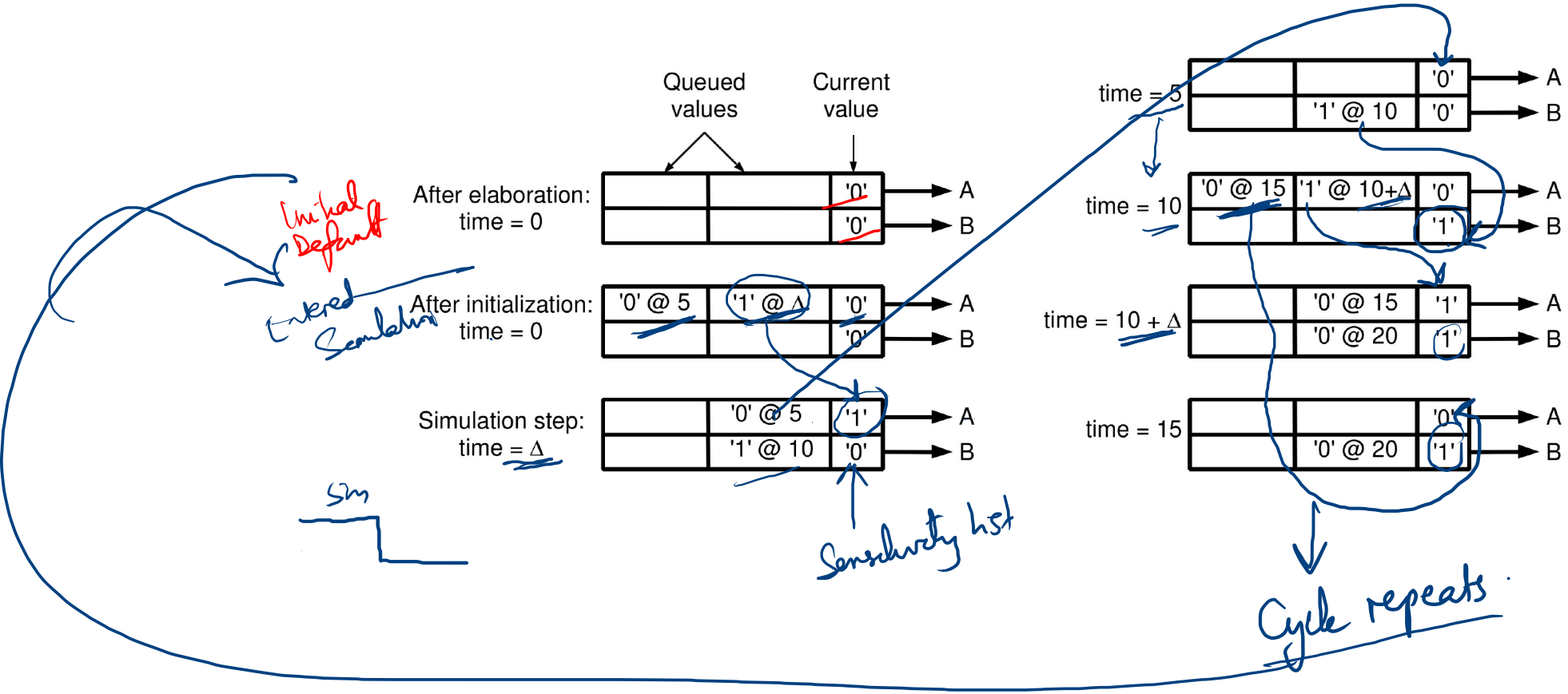


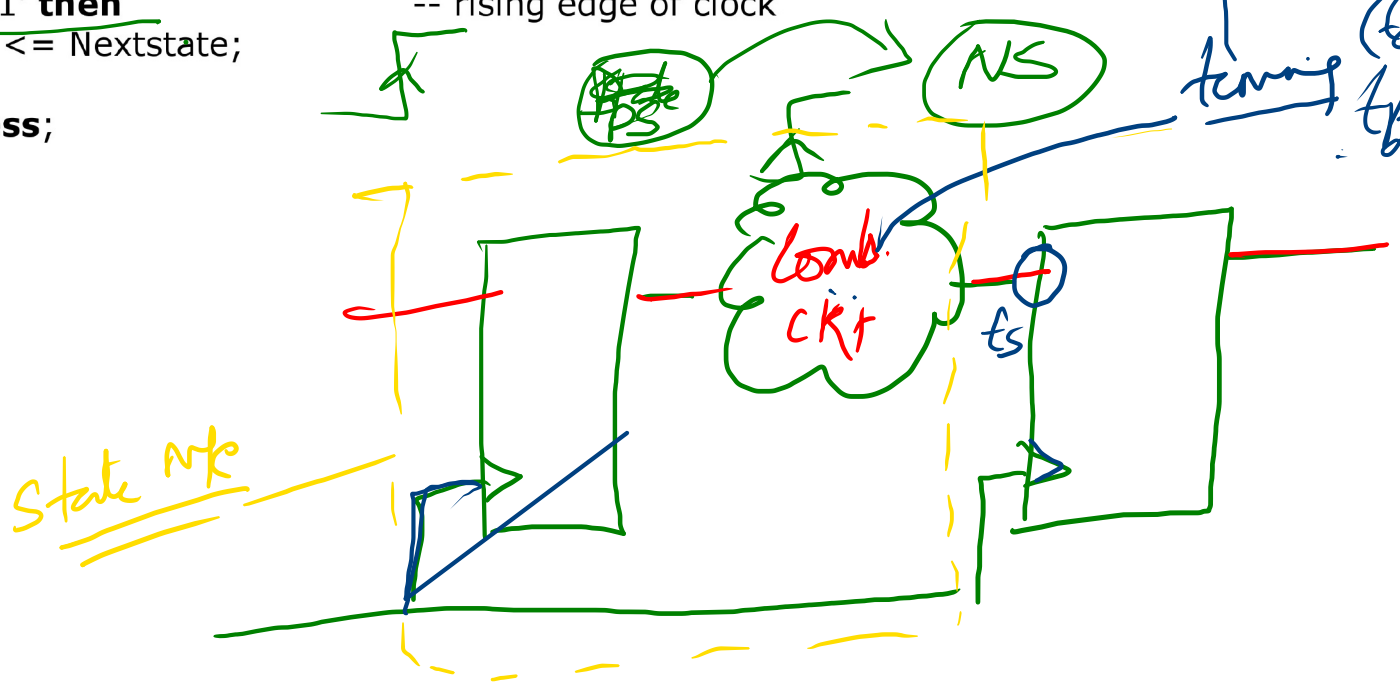
Figure 2-13(a) Behavioral Model for Figure 1-23

```
entity SM1_2 is
   port(X, CLK: in bit;  Z: out bit);
end SM1_2;

architecture Table of SM1_2 is
   signal State, Nextstate: integer := 0;
begin
   process(State,X)                    --Combinational Network
   begin
   case State is
      when 0 =>
         if X='0' then Z<='1'; Nextstate<=1; end if;
         if X='1' then Z<='0'; Nextstate<=2; end if;
      when 1 =>
         if X='0' then Z<='1'; Nextstate<=3; end if;
         if X='1' then Z<='0'; Nextstate<=4; end if;
      when 2 =>
         if X='0' then Z<='0'; Nextstate<=4; end if;
         if X='1' then Z<='1'; Nextstate<=4; end if;
      when 3 =>
         if X='0' then Z<='0'; Nextstate<=5; end if;
         if X='1' then Z<='1'; Nextstate<=5; end if;
      when 4 =>
         if X='0' then Z<='1'; Nextstate<=5; end if;
         if X='1' then Z<='0'; Nextstate<=6; end if;
```

PS	NS X = 0	NS X = 1	Z X = 0	Z X = 1
S0	S1	S2	1	0
S1	S3	S4	1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	–	1	–

Figure 2-13(b) Behavioral model for Figure 1-17

```
        when 5 =>
            if X='0' then Z<='0'; Nextstate<=0; end if;
            if X='1' then Z<='1'; Nextstate<=0; end if;
        when 6 =>
            if X='0' then Z<='1'; Nextstate<=0; end if;
        when others => null;                -- should not occur
    end case;
end process;

    process(CLK)                            -- State Register
        begin
            if CLK='1' then                 -- rising edge of clock
                State <= Nextstate;
            end if;
        end process;
end Table;
```



A simulator command file that can be used to test Figure 2-13 is as follows:

```
wave CLK X State NextState Z
force CLK 0 0, 1 100 -repeat 200
force X 0 0, 1 350, 0 550, 1 750, 0 950, 1 1350
run 1600
```

Execution of the preceding command file produces the waveforms shown in Figure 2-14.

Figure 2-14 Waveforms for Figure 2-13

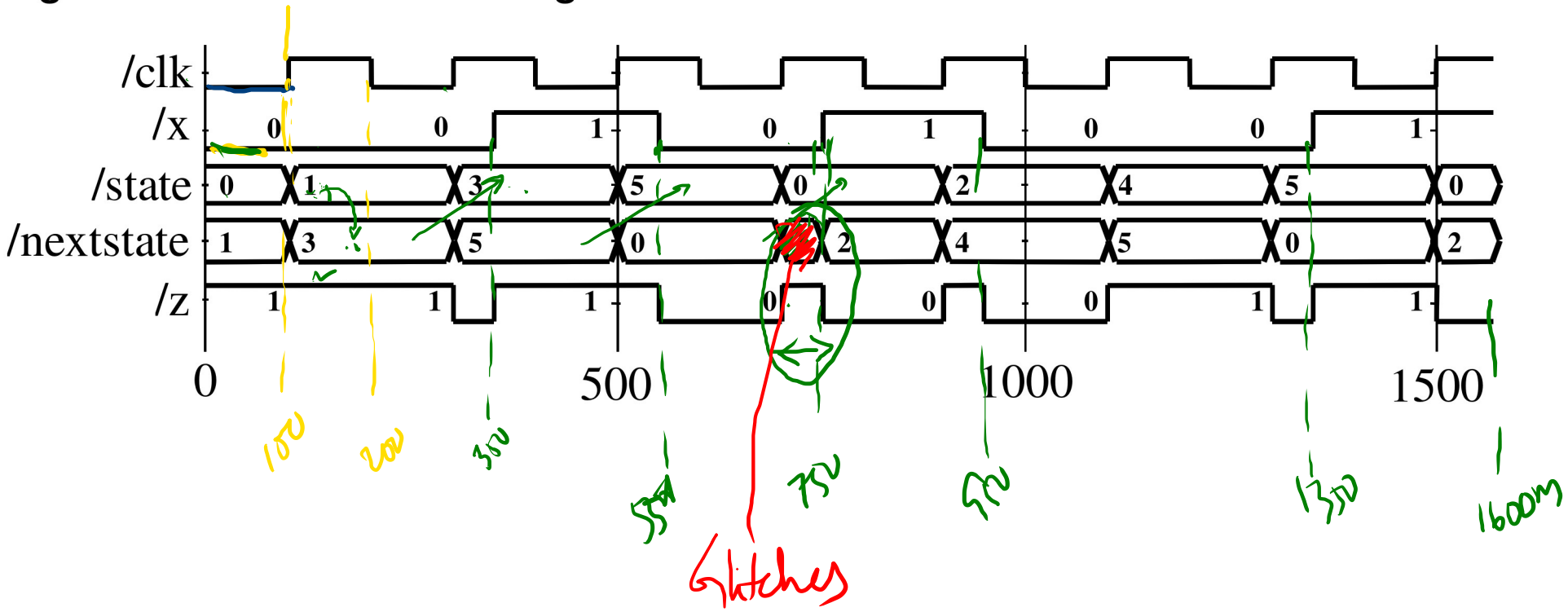


Figure 2-15 Sequential Machine Model Using Equations

```
-- The following is a description of the sequential machine of
-- Figure 1-17 in terms of its next state equations.
-- The following state assignment was used:
-- S0-->0; S1-->4; S2-->5; S3-->7; S4-->6; S5-->3; S6-->2

entity SM1_2 is
   port(X,CLK: in bit;
      Z: out bit);
end SM1_2;

architecture Equations1_4 of SM1_2 is
   signal Q1,Q2,Q3: bit;
begin
   process(CLK)
   begin
      if CLK='1' then              -- rising edge of clock
         Q1<=not Q2 after 10 ns;
         Q2<=Q1 after 10 ns;
         Q3<=(Q1 and Q2 and Q3) or (not X and Q1 and not Q3) or
            (X and not Q1 and not Q2) after 10 ns;
      end if;
   end process;
   Z<=(not X and not Q3) or (X and Q3) after 20 ns;
end Equations1_4;
```

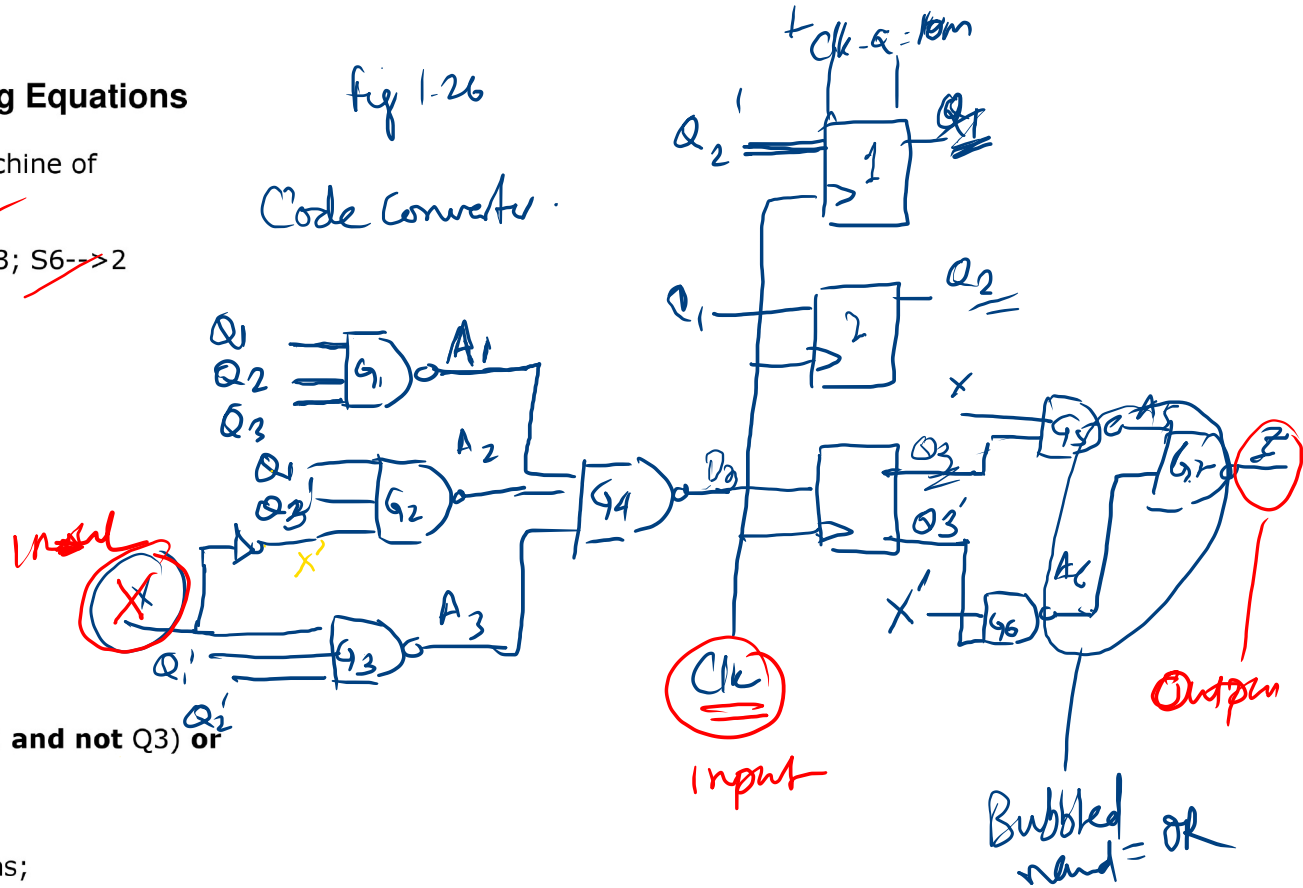
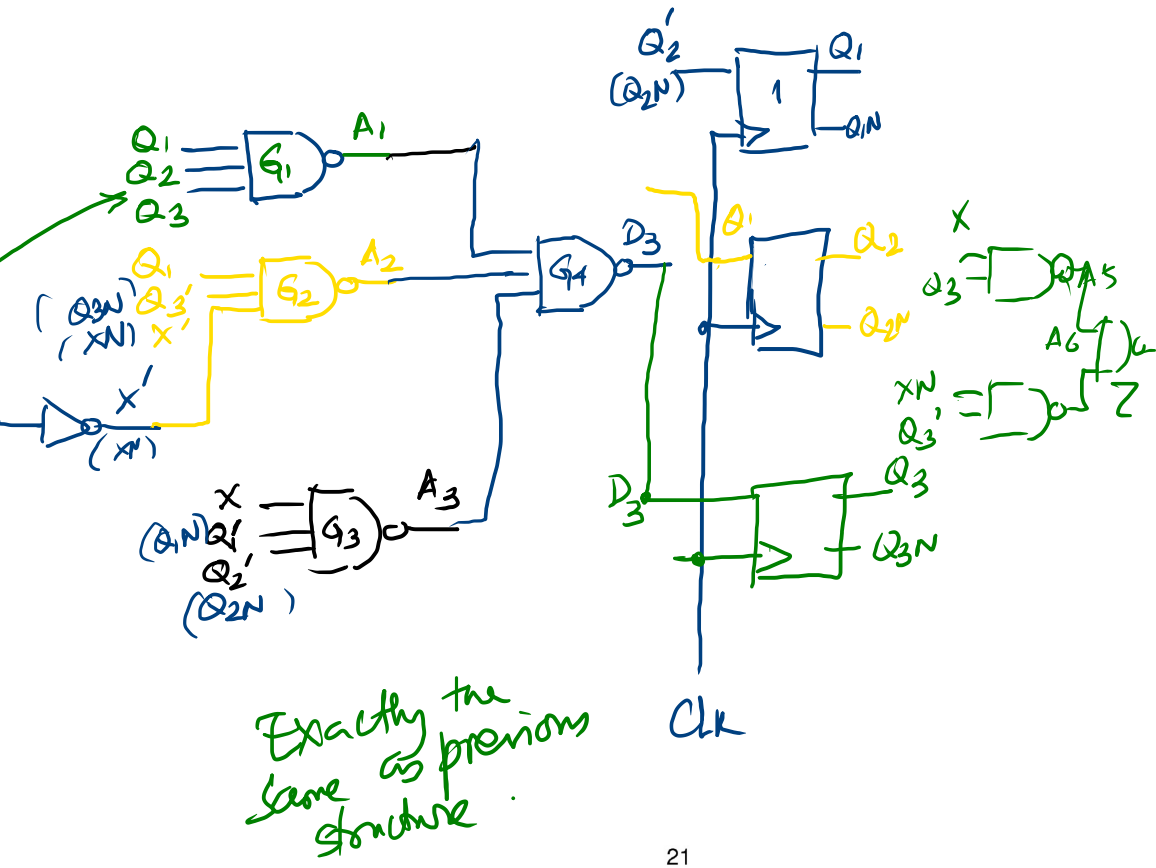


Figure 2-16 Structural Model of Sequential Machine

```
-- The following is a STRUCTURAL VHDL description of the network of Figure 1-20.

library BITLIB;
use BITLIB.bit_pack.all;

entity SM1_2 is
   port(X,CLK: in bit;
        Z: out bit);
end SM1_2;

architecture Structure of SM1_2 is
   signal A1,A2,A3,A5,A6,D3: bit:='0';
   signal Q1,Q2,Q3: bit:='0';
   signal Q1N,Q2N,Q3N, XN: bit:= '1';
begin
   I1:  Inverter port map (X,XN);
   G1:  Nand3 port map (Q1,Q2,Q3,A1);
   G2:  Nand3 port map (Q1,Q3N,XN,A2);
   G3:  Nand3 port map (X,Q1N,Q2N,A3);
   G4:  Nand3 port map (A1,A2,A3,D3);
   FF1: DFF port map (Q2N,CLK,Q1,Q1N);
   FF2: DFF port map (Q1,CLK,Q2,Q2N);
   FF3: DFF port map (D3,CLK,Q3,Q3N);
   G5:  Nand2 port map (X,Q3,A5);
   G6:  Nand2 port map (XN,Q3N,A6);
   G7:  Nand2 port map (A5,A6,Z);
end Structure
```

Executing the simulator command file given below produces the waveforms of Figure 2-17.

```
wave  CLK  X  Q1  Q2  Q3  Z
force  CLK  0  0,  1  100  -repeat 200
force  X  0  0,  1  350,  0  550,  1  750,  0  950,  1  1350
run 1600
```

Figure 2-17 Waveforms for Figure 2-16