SM Charts and Microprogramming

ELEC 418

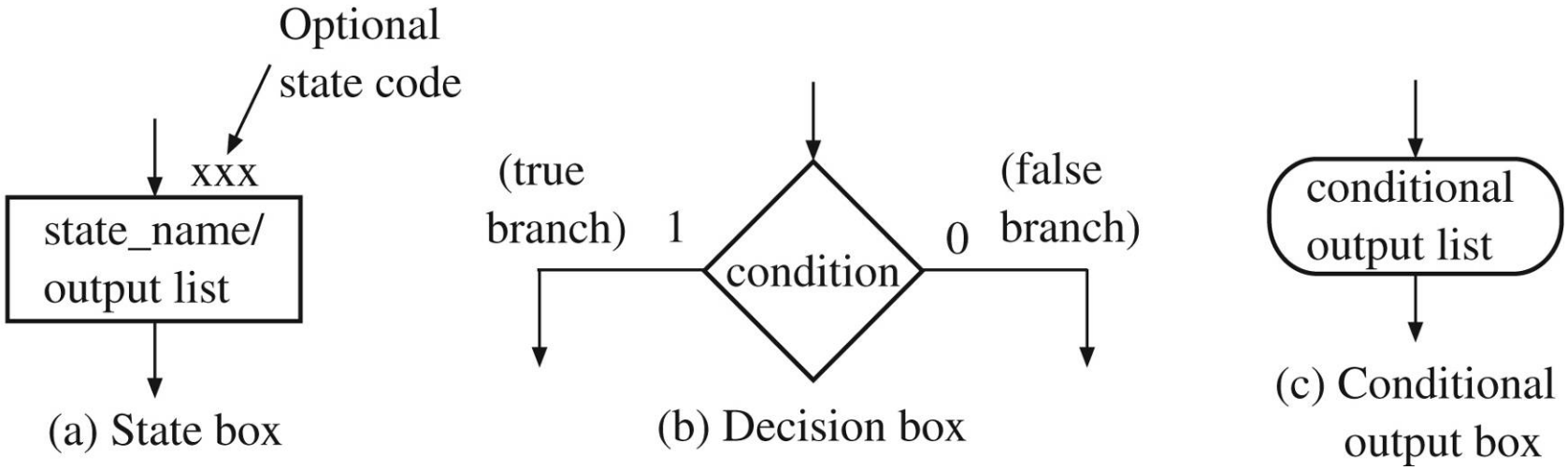
Advanced Digital Systems

Dr. Ron Hayne

Images Courtesy of Thomson Engineering

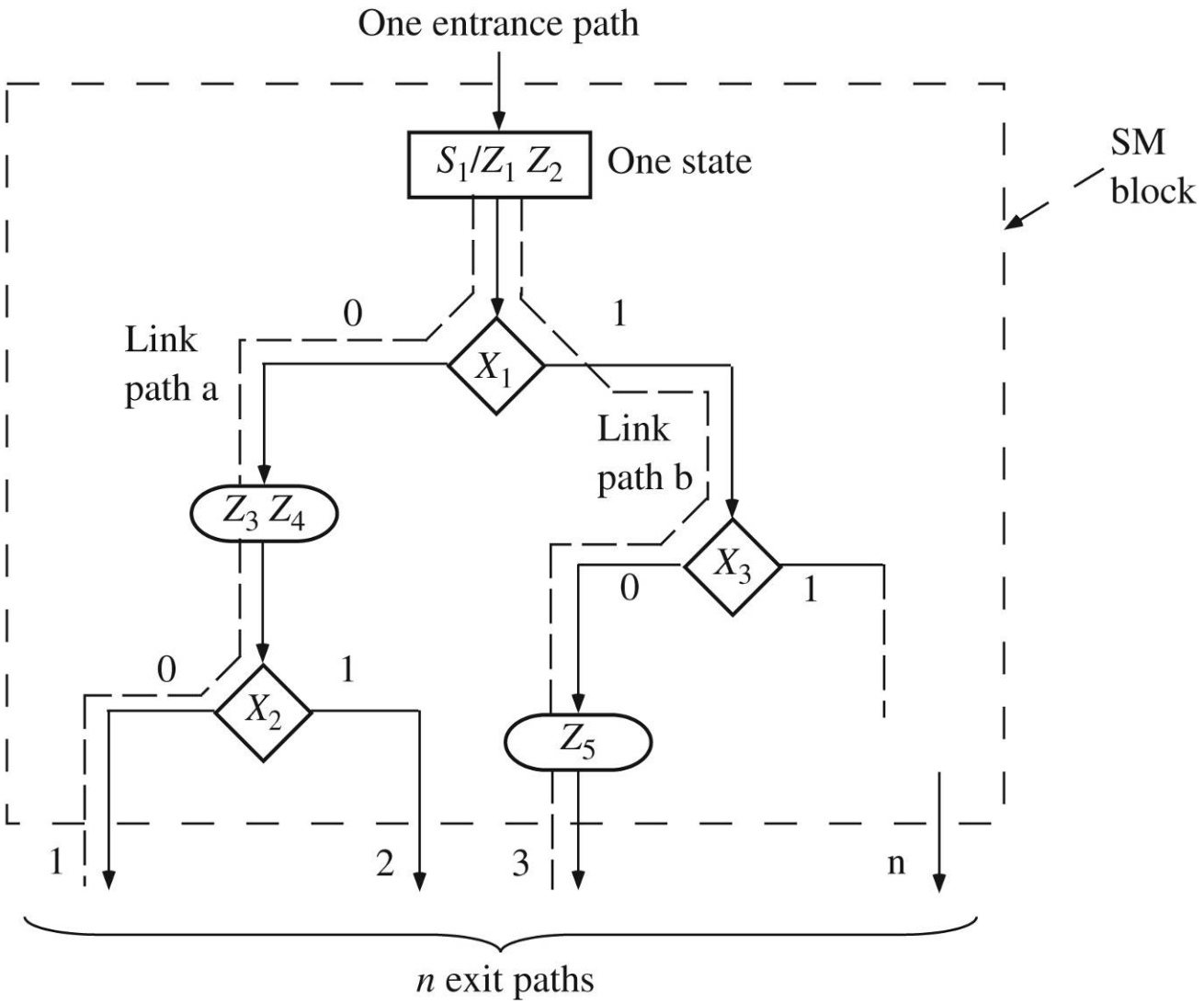
THE CITADEL
DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

State Machine Charts



- **Equivalent State Graph**
  - Exactly One Transition True at Any Time
  - Next State Uniquely Defined
- **SM Block**
  - One Entrance Path
  - One or More Exit Paths
  - No Internal Feedback

Example SM Block



State Graph to SM Chart

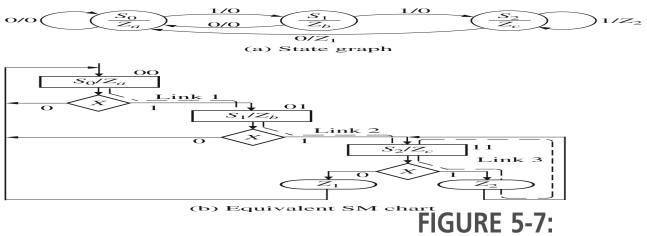
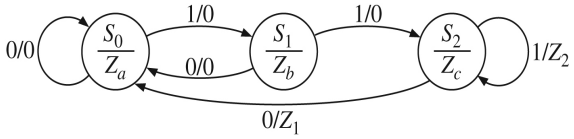
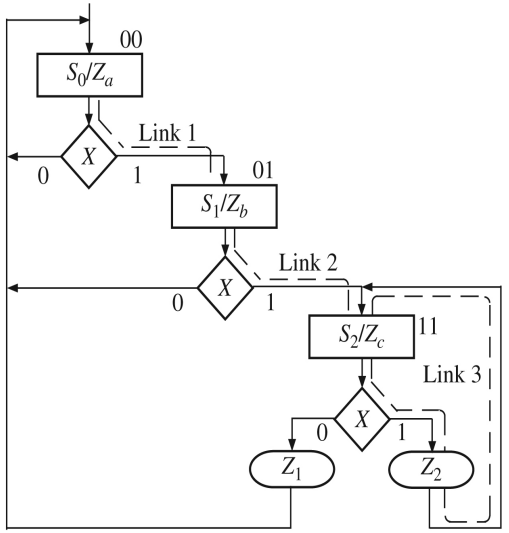


FIGURE 5-7:
Conversion of a State Graph to an SM Chart



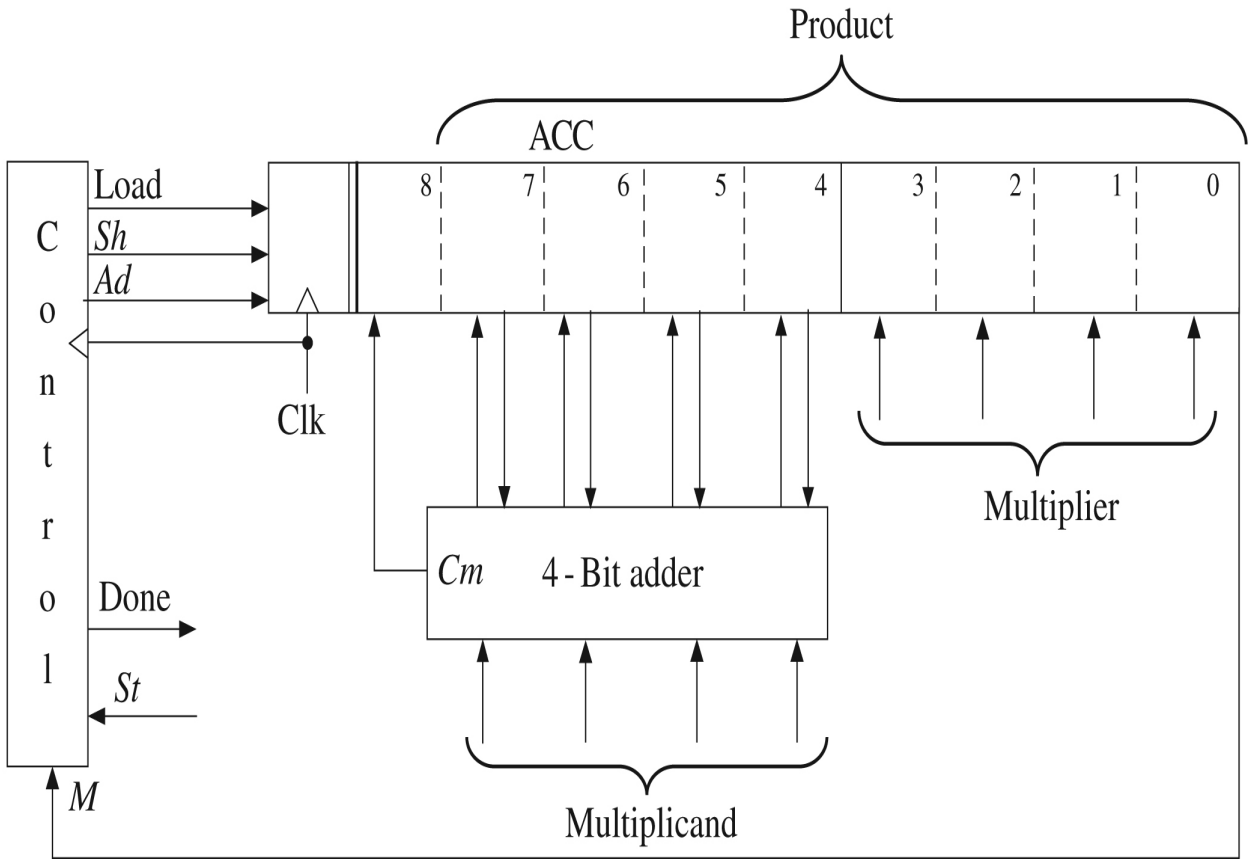
(a) State graph



(b) Equivalent SM chart

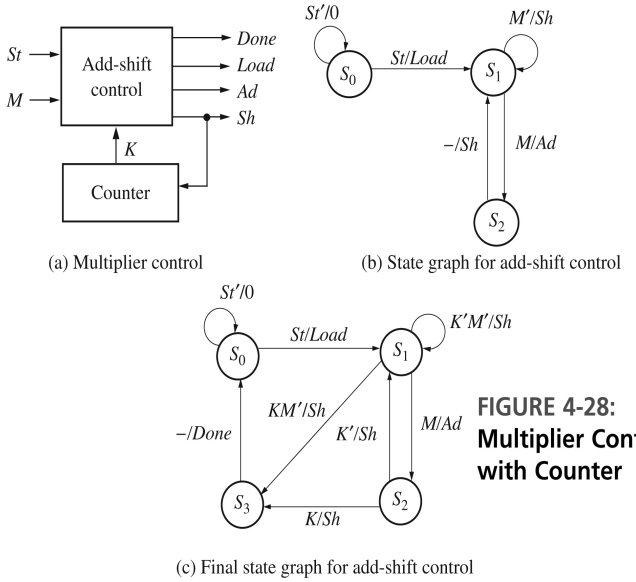
Add-and-Shift Multiplier

FIGURE 4-25: Block Diagram for Binary Multiplier



Multiplier Control

FIGURE 4-28:
Multiplier Control with Counter

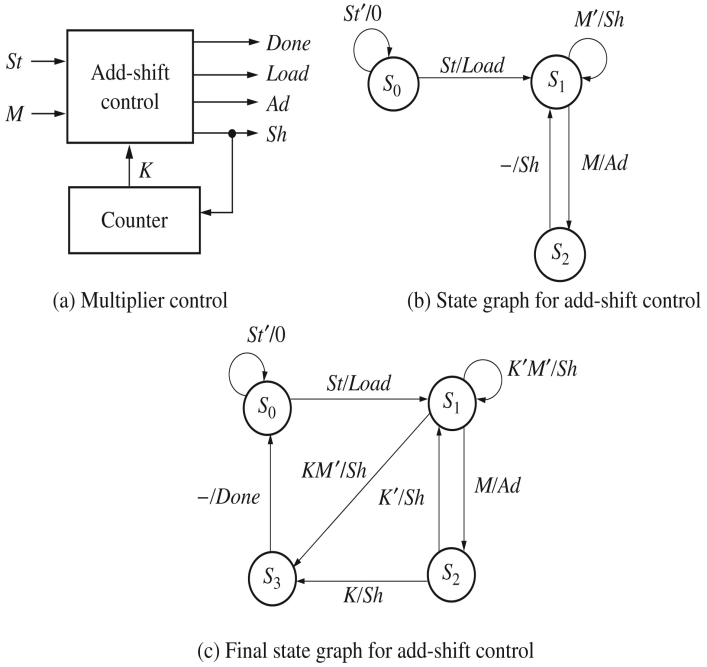


(a) Multiplier control

(b) State graph for add-shift control

(c) Final state graph for add-shift control

FIGURE 4-28:
Multiplier Control with Counter



(a) Multiplier control

(b) State graph for add-shift control

(c) Final state graph for add-shift control

Multiplier Control

```
entity Mult is
  port(CLK, St, K, M: in std_logic;
       Load, Sh, Ad, Done: out std_logic);
end Mult;

architecture SMbehave of Mult is
signal State, Nextstate: integer range 0 to 3;
begin
  process(St, K, M, State)
  begin
    Load <= '0'; Sh <= '0'; Ad <= '0'; Done <= '0';
```

Multiplier Control

```
case State is
  when 0 =>
    if St = '1' then
      Load <= '1';
      Nextstate <= 1;
    else
      Nextstate <= 0;
    end if;
```

Multiplier Control

```
when 1 =>
  if M = '1' then
    Ad <= '1';
    Nextstate <= 2;
  else
    Sh <= '1';
    if K = '1' then
      Nextstate <= 3;
    else
      Nextstate <= 1;
    end if;
  end if;
```

Multiplier Control

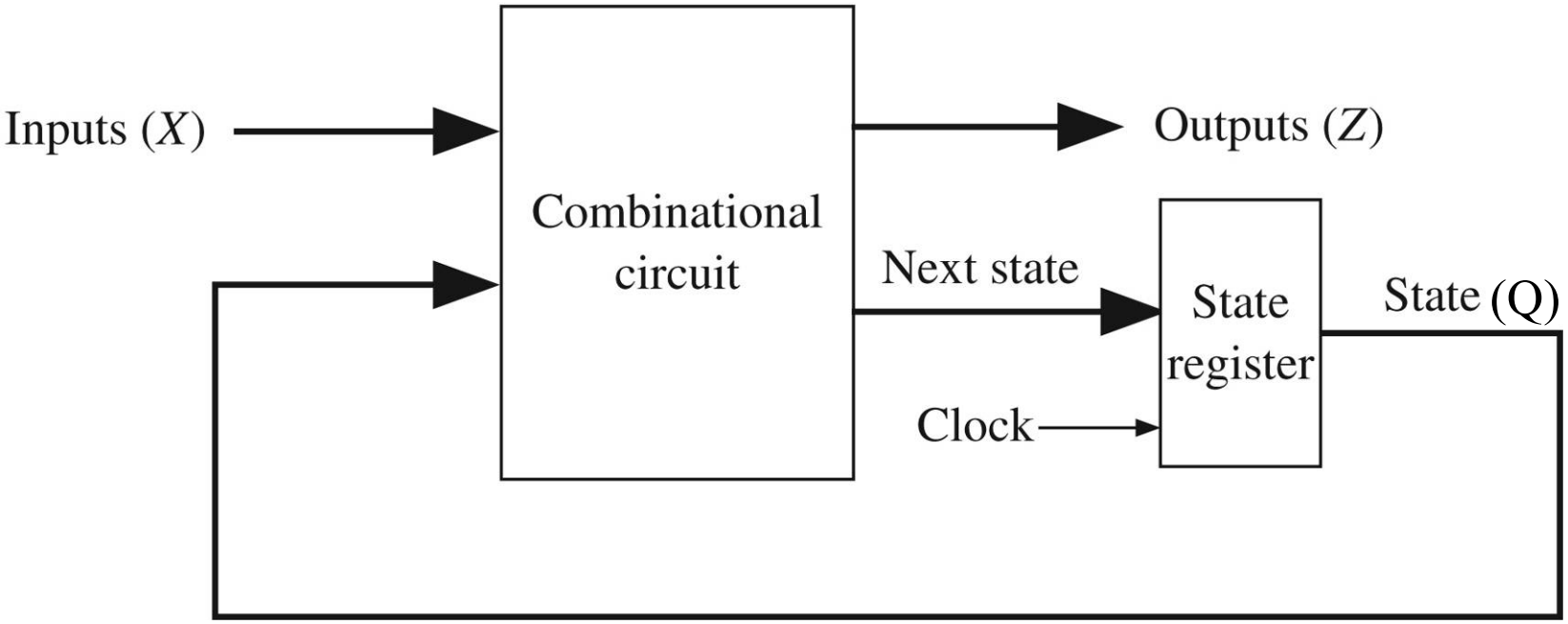
```
    when 2 =>
      Sh <= '1';
      if K = '1' then
        Nextstate <= 3;
      else
        Nextstate <= 1;
      end if;
    when 3 =>
      Done <= '1';
      Nextstate <= 0;
  end case;
end process;
```

Multiplier Control

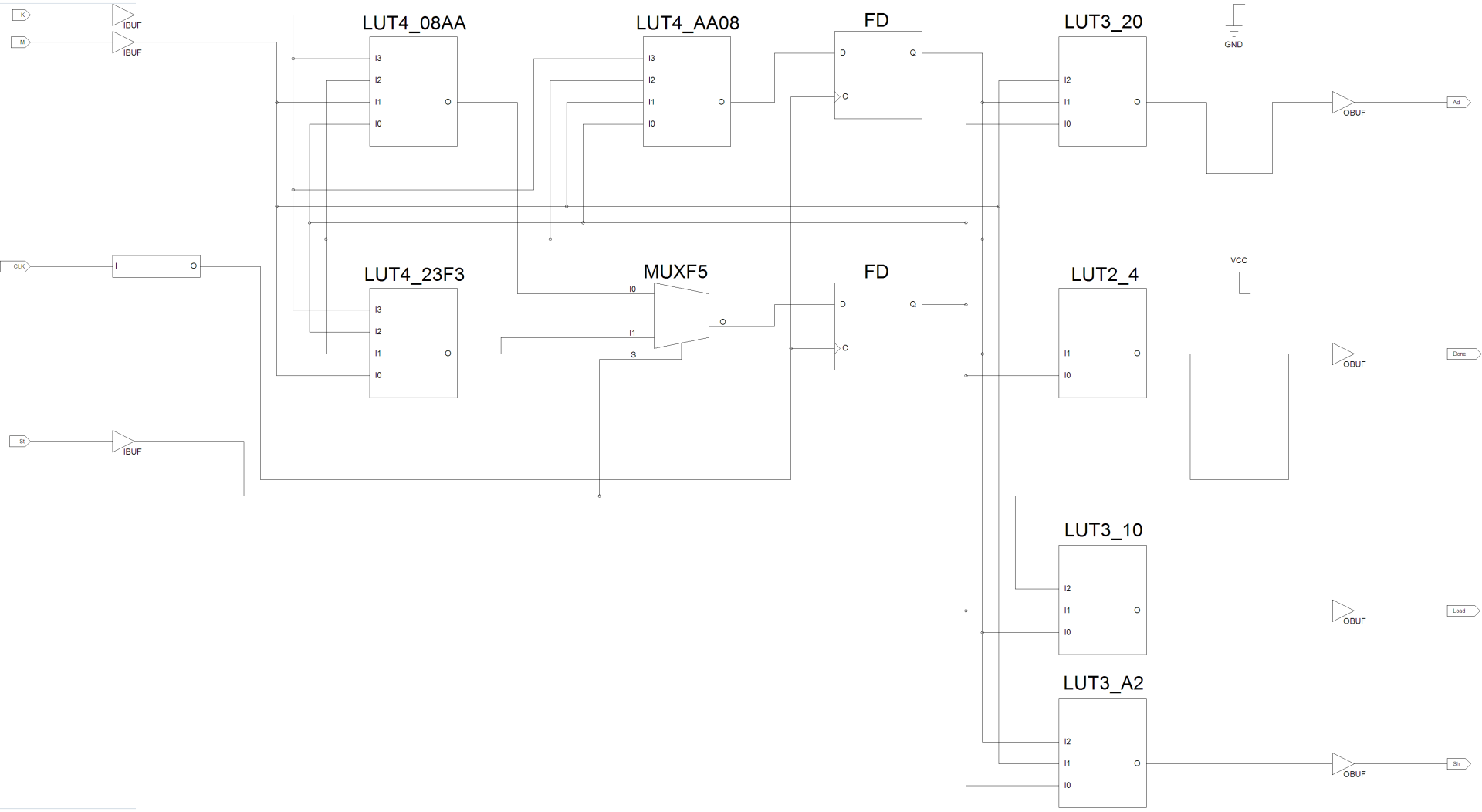
```
  process(CLK)
  begin
    if rising_edge(CLK) then
      State <= Nextstate;
    end if;
  end process;
end SMbehave;
```

Sequential Machine

- **Mealy**
  - $Z = f(X, Q)$
- **Moore**
  - $Z = f(Q)$



FPGA Synthesis



Microprogramming

- **Hardwired Control**
  - Implemented using gates and flip-flops
  - Faster, less flexible, limited complexity
- **Microprogram Control**
  - Control Store
    - Memory storing control signals and next state info
    - Controller sequences through memory
  - Slower, more flexible, greater complexity

Microprogram Controllers

FIGURE 5-29:
Typical Hardware Arrangement for Microprogramming

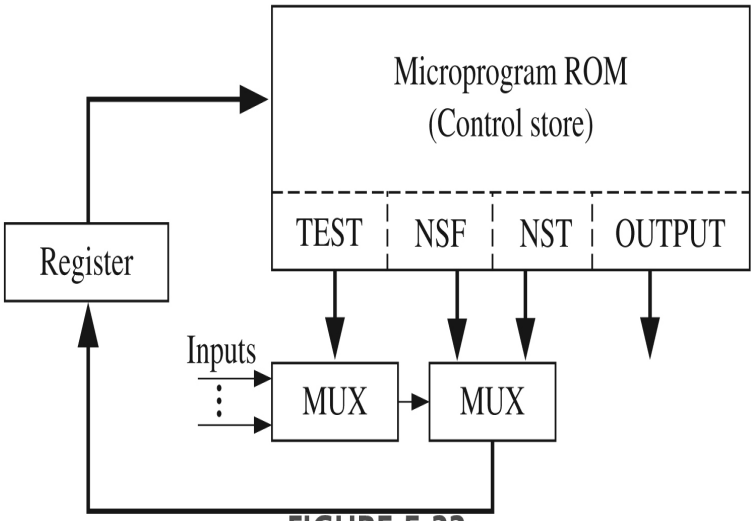
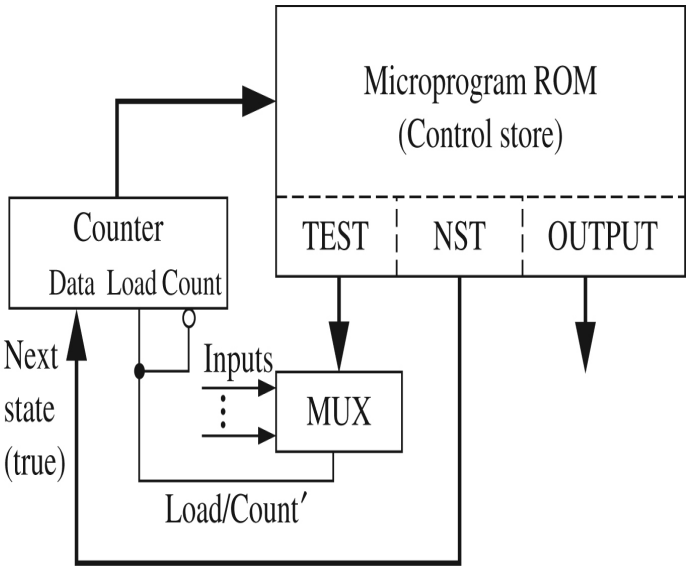


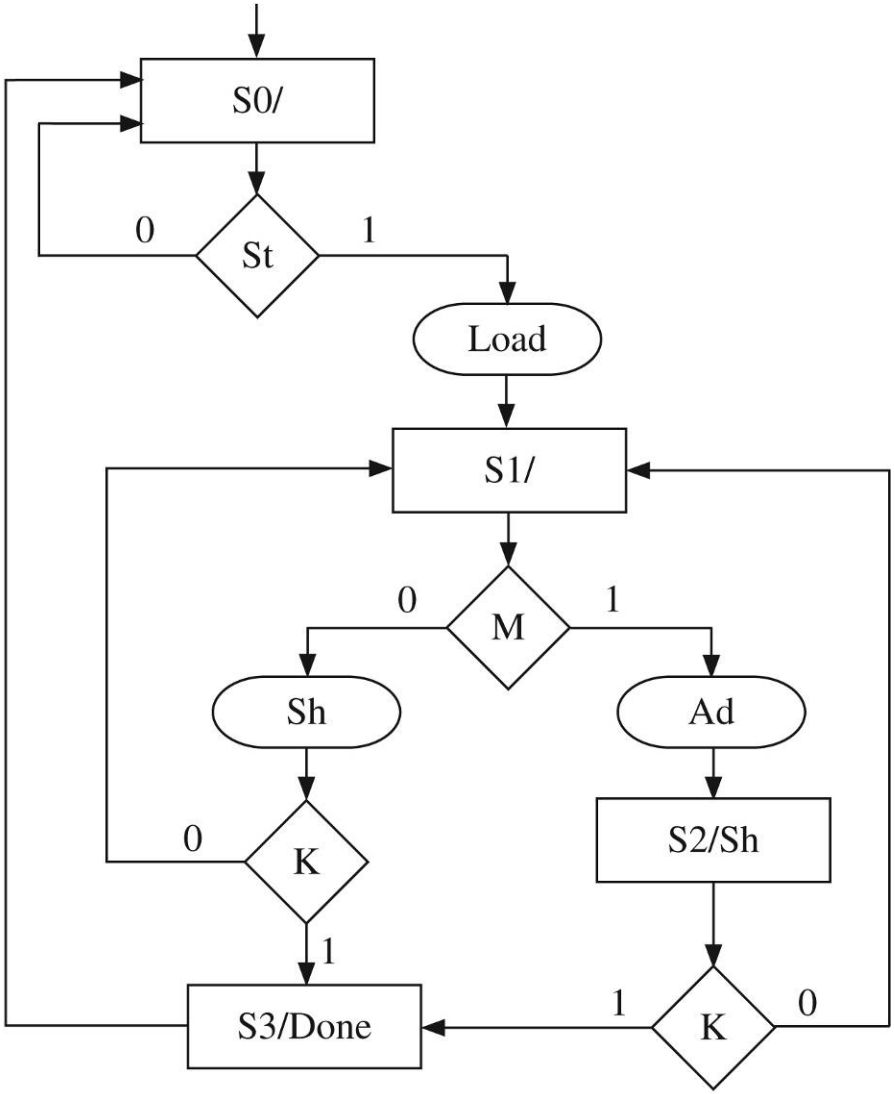
FIGURE 5-33:
Microprogrammed System with Single Address Microcode



Implementing SM Charts

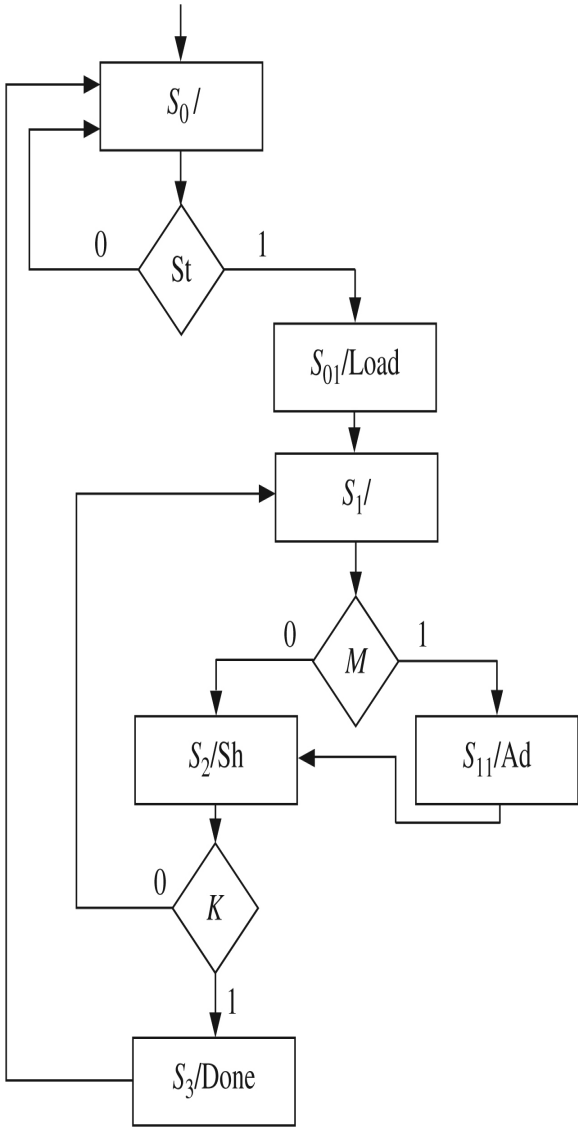
- **`Transformations for Microprogramming`**
  - Eliminate conditional outputs
    - Transform to a Moore machine
  - Test only one input in each state
  - Eliminate redundant states
    - Same output and same next states

Multiplier Control



Modified Multiplier Control

FIGURE 5-31:
Modified Multiplier SM Chart After State Minimization is Applied to Figure 5-30



Two-Address Microcode

FIGURE 5-29:
Typical Hardware Arrangement for Microprogramming

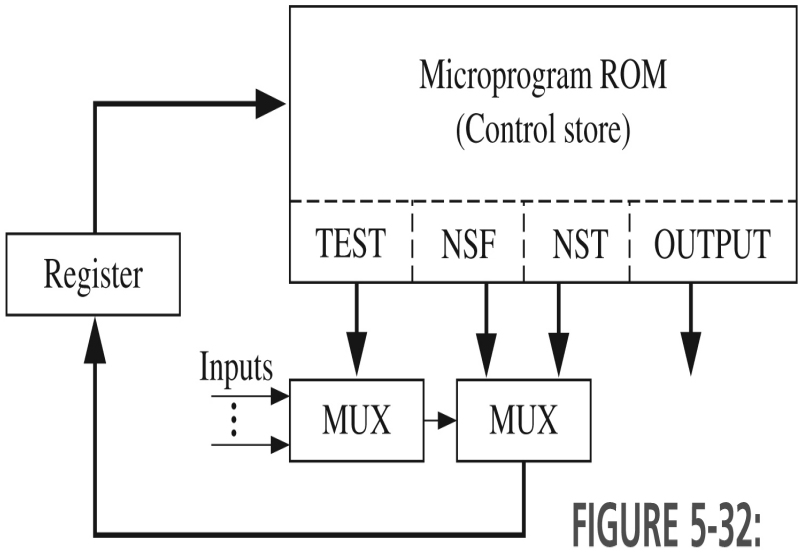
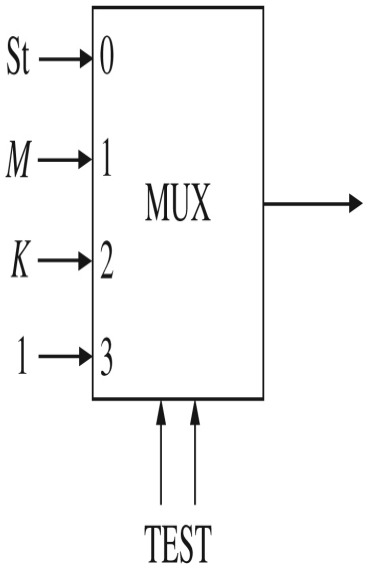


FIGURE 5-32:
4-to-1 MUX for Microprogramming the Multiplier (Two Address Microcode)



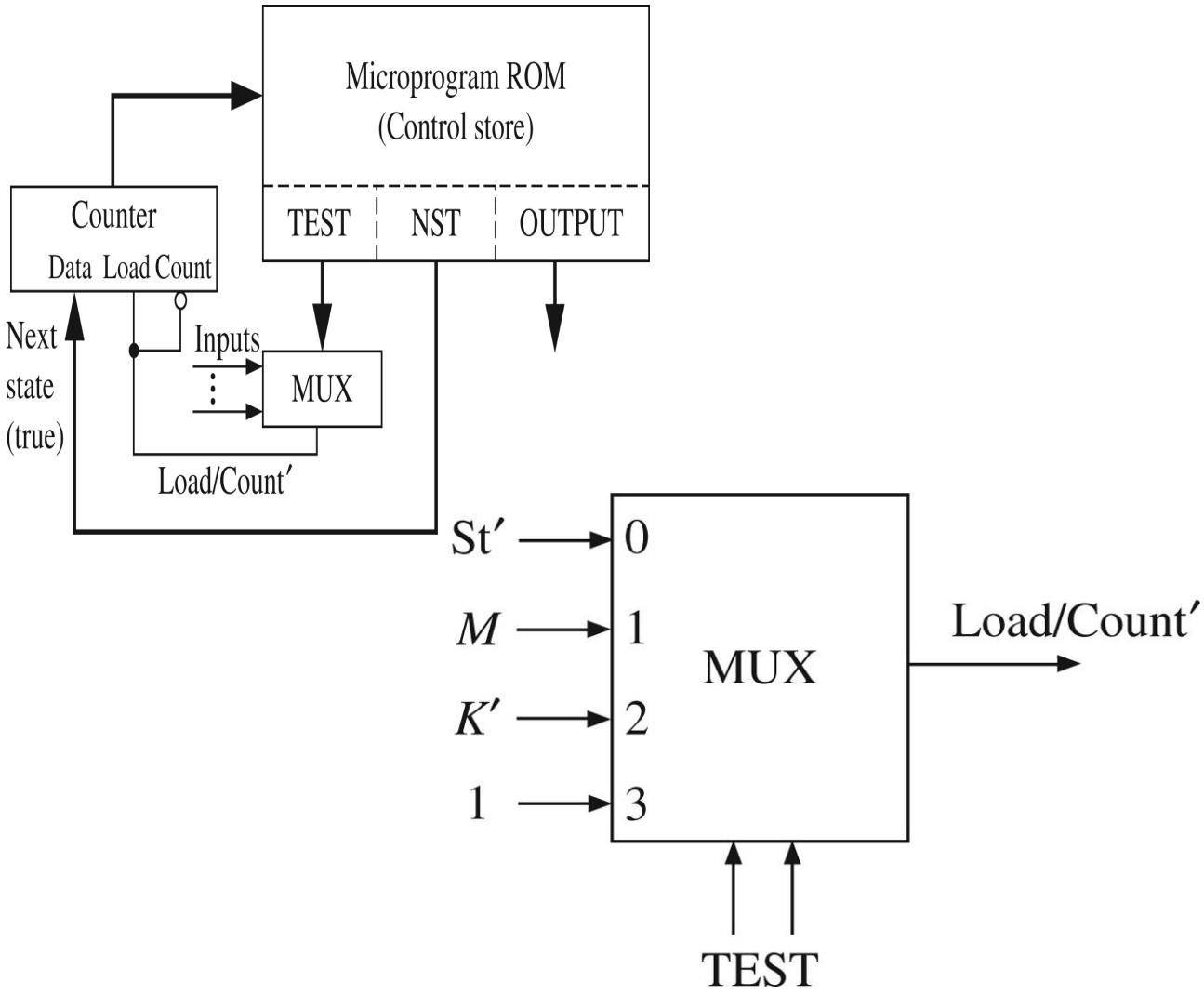
Two-Address Microprogram

TABLE 5-3: Two Address Microprogram for Multiplier. Both NST and NSF Specified (Corresponds to Figure 5-29)

State	ABC	TEST	NSF	NST	Load	Ad	Sh	Done
S_0	000	00	000	001	0	0	0	0
S_{01}	001	11	010	010	1	0	0	0
S_1	010	01	100	011	0	0	0	0
S_{11}	011	11	100	100	0	1	0	0
S_2	100	10	010	101	0	0	1	0
S_3	101	11	000	000	0	0	0	1

Single-Address Microcode

FIGURE 5-33:
Microprogrammed System with Single Address Microcode



Single-Address Microprogram

State	ABC	TEST	NST	Load	Ad	Sh	Done
S_0	000	00	000	0	0	0	0
S_{01}	001	11	010	1	0	0	0
S_1	010	01	101	0	0	0	0
S_2	011	10	010	0	0	1	0
S_3	100	11	000	0	0	0	1
S_{11}	101	11	011	0	1	0	0

Summary

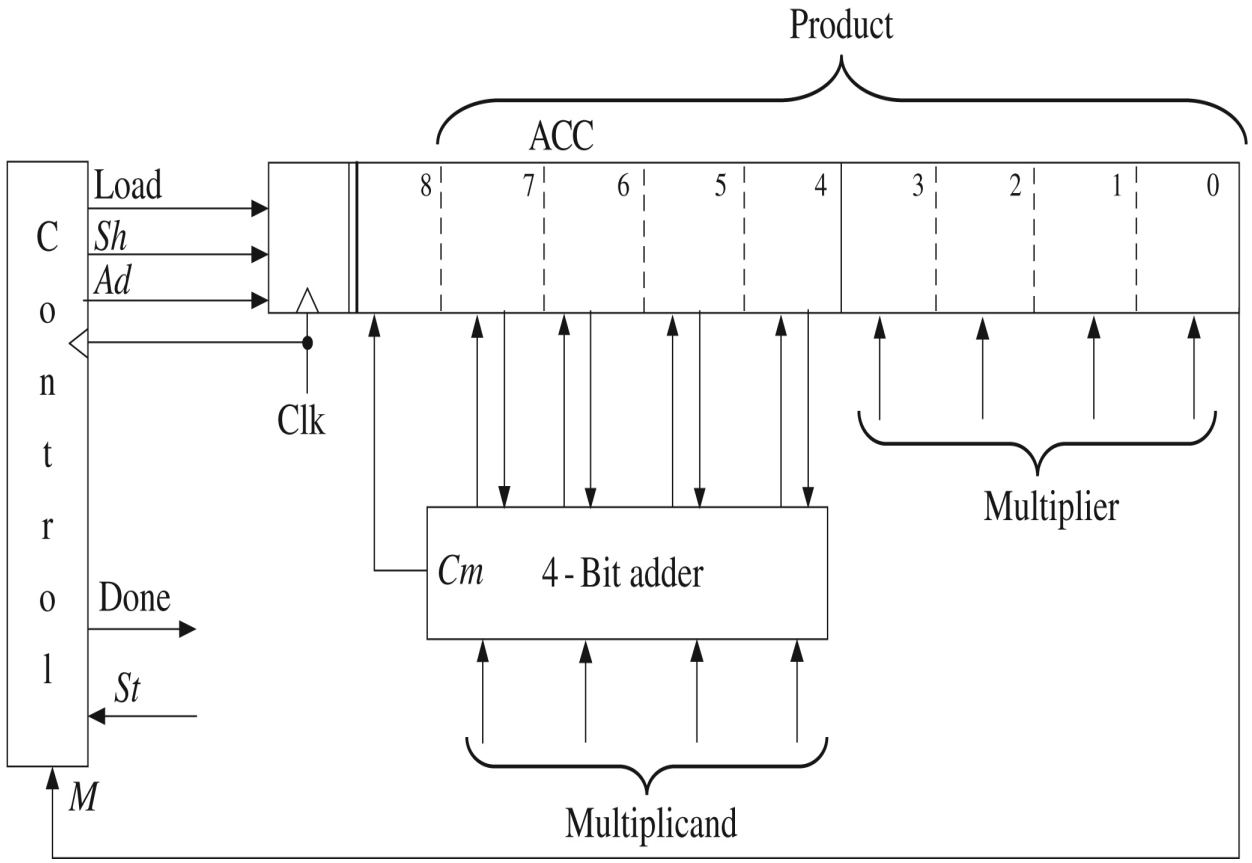
- **SM Charts**
  - Equivalent State Graph
- **Microprogramming**
  - Two-Address Microcode
  - Single-Address Microcode

Putting It All Together

- **Add-and-Shift Multiplier**
- **Multiplier Control**
  - Counter
  - SM Chart
- **Two-Address Microcode**
  - Microprogram ROM
- **ModelSim Simulation**
- **FPGA Implementation**
  - ChipScope Pro

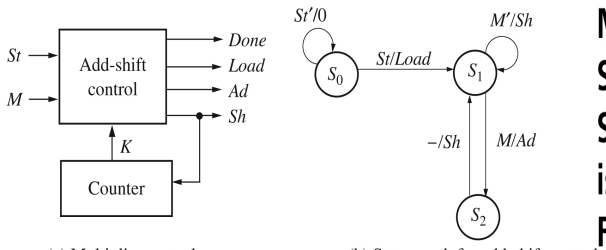
Add-and-Shift Multiplier

FIGURE 4-25: Block Diagram for Binary Multiplier



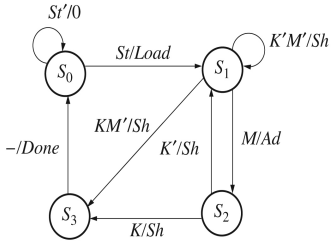
Multiplier Control

FIGURE 4-28:
Multiplier Control with Counter



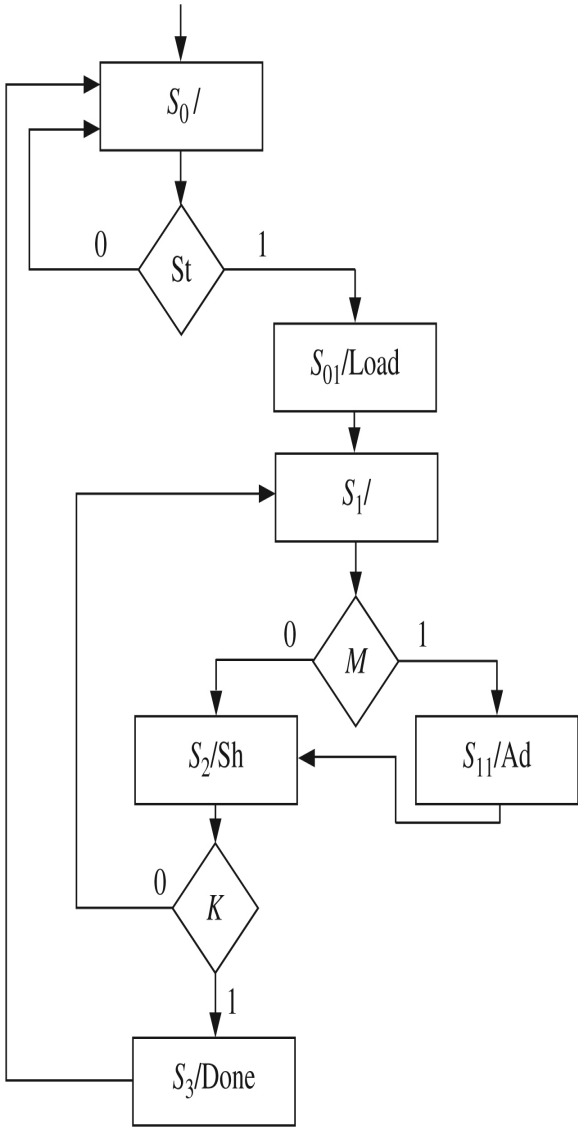
(a) Multiplier control

(b) State graph for add-shift control



(c) Final state graph for add-shift control

FIGURE 5-31:
Modified Multiplier SM Chart After State Minimization is Applied to Figure 5-30



Two-Address Microcode

FIGURE 5-29:
Typical Hardware Arrangement for Microprogramming

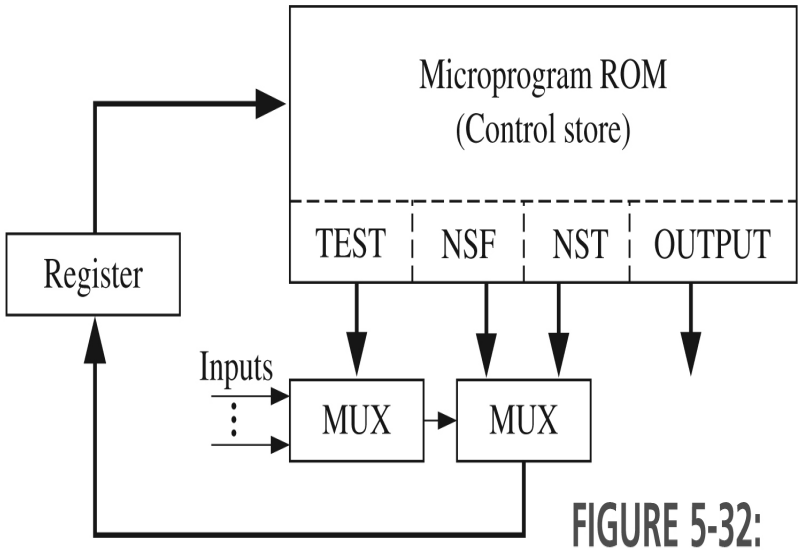
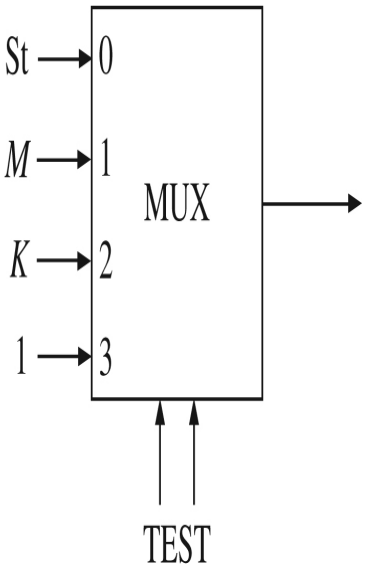


FIGURE 5-32:
4-to-1 MUX for Microprogramming the Multiplier (Two Address Microcode)



Two-Address Microprogram

TABLE 5-3: Two Address Microprogram for Multiplier. Both NST and NSF Specified (Corresponds to Figure 5-29)

State	ABC	TEST	NSF	NST	Load	Ad	Sh	Done
S_0	000	00	000	001	0	0	0	0
S_{01}	001	11	010	010	1	0	0	0
S_1	010	01	100	011	0	0	0	0
S_{11}	011	11	100	100	0	1	0	0
S_2	100	10	010	101	0	0	1	0
S_3	101	11	000	000	0	0	0	1

Look-Up Tables (ROM)

```
architecture Table of Parity_Gen is
  type OutTable is array(0 to 15) of std_logic;
  signal ParityBit: std_logic;
  constant OT: OutTable :=
     ('1','0','0','1','0','1','1','0',
      '0','1','1','0','1','0','0','1');
begin
  ParityBit <= OT(conv_integer(X));
  Y <= X & ParityBit;
end Table;
```

Multiplexers

```
entity MUX4to1 is
  port(I: in std_logic_vector(3 downto 0);
       S: in std_logic_vector(1 downto 0);
       F: out std_logic);
end MUX4to1;
architecture Dataflow of MUX4to1 is
begin
  with S select
    F <= I(0) when "00",
         I(1) when "01",
         I(2) when "10",
         I(3) when "11";
end Dataflow;
```

Multiplier VHDL Model

```
library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity mult4X4_micro is
 port(Clk, St: in std_logic;
      Mplier, Mcand: in std_logic_vector(3 downto 0);
      Product: out std_logic_vector(7 downto 0);
      Done: out std_logic);
end mult4X4_micro;
```

VHDL Model

```
architecture microprogram of mult4X4_micro is

type ROM is array(0 to 5) of
            std_logic_vector(11 downto 0);
constant control_store: ROM :=
  (X"010", X"D28", X"630", X"E44", X"952", X"C01");

signal ACC: std_logic_vector(8 downto 0);
alias M: std_logic is ACC(0);
signal Load, Ad, Sh, K: std_logic;
signal counter: std_logic_vector(1 downto 0) := "00";
```

VHDL Model

```
signal TMUX: std_logic;
signal uAR: std_logic_vector(2 downto 0) := "000";
signal uIR: std_logic_vector(11 downto 0) := X”000”;

alias TEST: std_logic_vector(1 downto 0) is
            uIR(11 downto 10);
alias NSF: std_logic_vector(2 downto 0) is
           uIR(9 downto 7);
alias NST: std_logic_vector(2 downto 0) is
           uIR(6 downto 4);
```

VHDL Model

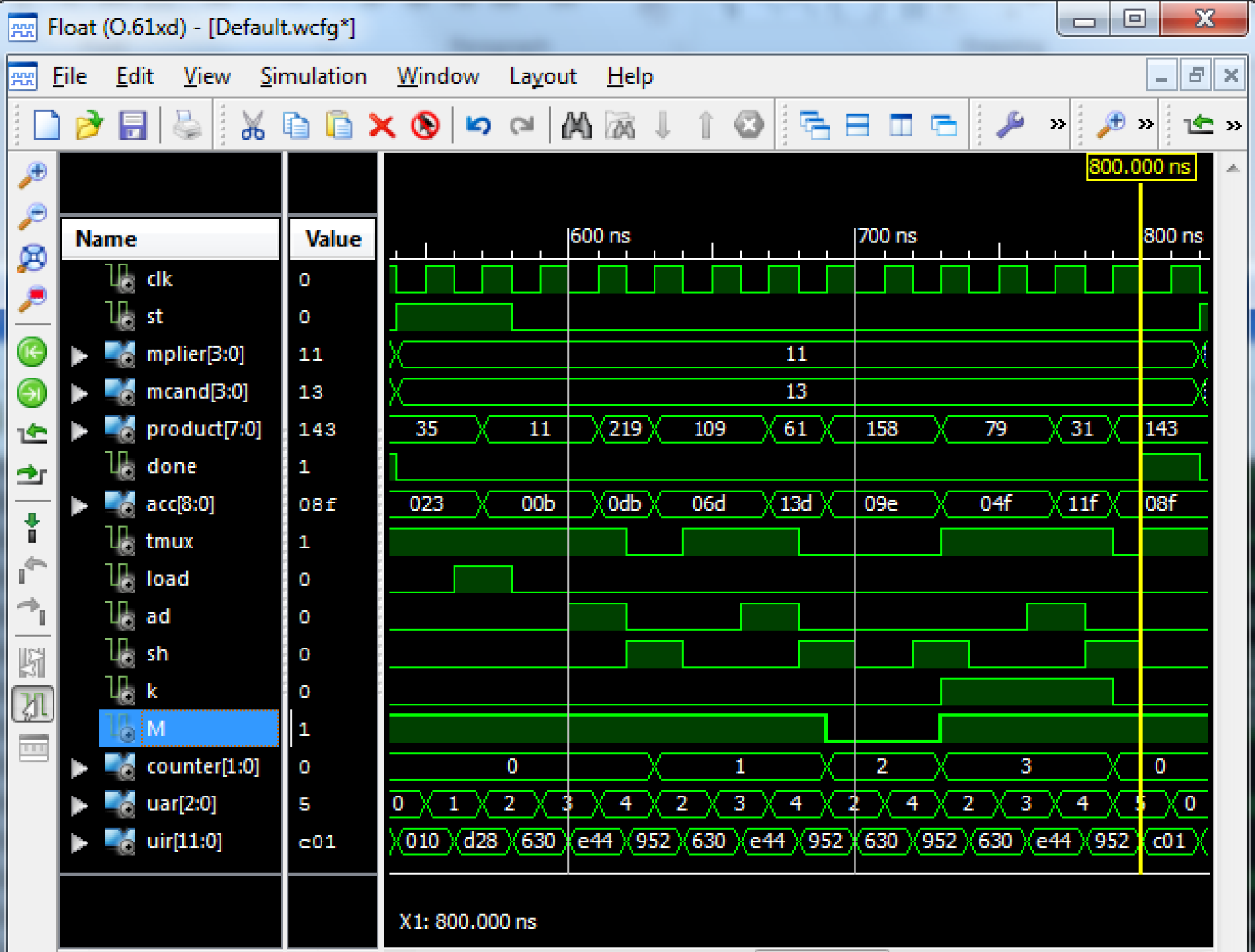
```
begin
  Load <= uIR(3);
  Ad <= uIR(2);
  Sh <= uIR(1);
  Done <= uIR(0);
  Product <= ACC(7 downto 0);
  K <= '1' when counter = "11" else '0';
  with TEST select
    TMUX <= St when "00",
            M  when "01",
            K  when "10",
            '1' when others;
```

VHDL Model

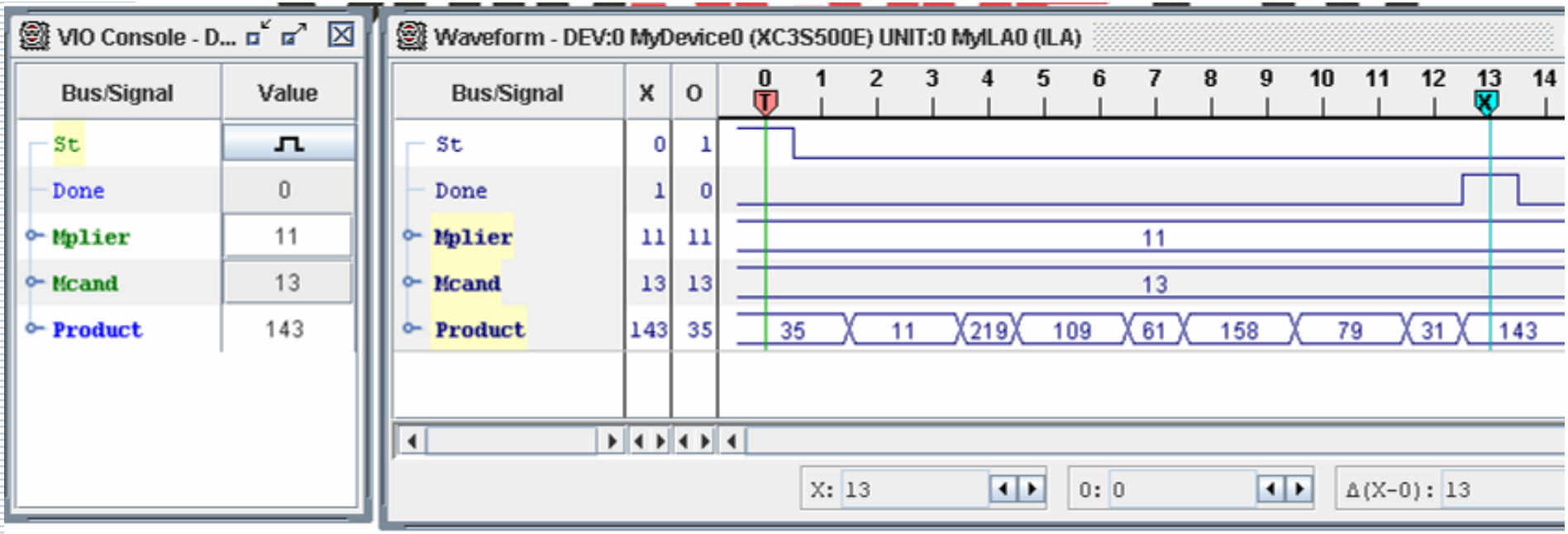
```
controller: process(Clk)
begin
  if falling_edge(Clk) then
    uIR <= control_store(to_integer(uAR));
  end if;
  if rising_edge(Clk) then
    if TMUX = '0' then
      uAR <= NSF;
    else
      uAR <= NST;
    end if;
    if Sh = '1' then
      counter <= counter + 1;
    end if;
  end if; end process;
```

VHDL Model

```
datapath: process(Clk)
begin
  if rising_edge(Clk) then
    if Load = '1' then
      ACC(8 downto 4) <= "00000";
      ACC(3 downto 0) <= Mplier;
    end if;
    if Ad = '1' then
      ACC(8 downto 4) <= '0' & ACC(7 downto 4)
                          + Mcand;
    end if;
    if Sh = '1' then
      ACC <= '0' & ACC(8 downto 1);
    end if;
  end if; end process; end microprogram;
```


Float (O.61xd) - [Default.wcfg*]
File
Edit
View
Simulation
Window
Layout
Help
Name
Value
clk
0
st
0
mplier[3:0]
11
mcand[3:0]
13
product[7:0]
143
done
1
acc[8:0]
08f
tmux
1
load
0
ad
0
sh
0
k
0
M
1
counter[1:0]
0
uar[2:0]
5
uir[11:0]
c01
800.000 ns
600 ns
700 ns
800 ns
11
13
35
11
219
109
61
158
79
31
143
023
00b
0db
06d
13d
09e
04f
11f
08f
0
1
2
3
0
0
1
2
3
4
2
3
4
2
4
2
3
4
5
0
010
d28
630
e44
952
630
e44
952
630
952
630
e44
952
c01
X1: 800.000 ns

FPGA ChipScope Pro



Summary

- **Add-and-Shift Multiplier**
- **Multiplier Control**
  - Counter
  - SM Chart
- **Two-Address Microcode**
  - Microprogram ROM
- **ModelSim Simulation**
- **FPGA Implementation**
  - ChipScope Pro