ECE 448
Lecture 7

Algorithmic State Machine (ASM) Charts: VHDL Code & Timing Diagrams
Required reading

- P. Chu, *FPGA Prototyping by VHDL Examples*
  
  *Chapter 5, FSM*
Recommended reading

• S. Brown and Z. Vranesci, *Fundamentals of Digital Logic with VHDL Design*
  Chapter 8, *Synchronous Sequential Circuits*
  Sections 8.1-8.5
  Section 8.10, *Algorithmic State Machine (ASM) Charts*
Finite State Machines in VHDL Style 1
Moore FSM

process(clock, reset)

concurrent statements

Moore FSM

process(clock, reset)
Mealy FSM

process(clock, reset)

Next State function

Inputs

 clk
 reset

 Present State Register

Outputs

 concurrent statements

Next State

 Outputs

Present State

Inputs

 clk
 reset

 Outputs

 concurrent statements

Inputs

 clk
 reset

 Outputs

 concurrent statements

Inputs

 clk
 reset

 Outputs

 concurrent statements
ASM Chart of Moore Machine
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY FSM_Moore_1 IS
  PORT ( clk : IN STD_LOGIC ;
         reset : IN STD_LOGIC ;
         input : IN STD_LOGIC ;
         output : OUT STD_LOGIC ) ;
END FSM_Moore_1 ;
ARCHITECTURE behavioral of FSM_Moore_1 IS
TYPE state IS (S0, S1, S2);
SIGNAL Present_State: state;

BEGIN
U_Moore: PROCESS (clk, reset)
BEGIN
  IF(reset = '1') THEN
    Present_State <= S0;
  ELSIF rising_edge(clk) THEN
    CASE Present_State IS
    WHEN S0 =>
      IF input = '1' THEN
        Present_State <= S1;
      ELSE
        Present_State <= S0;
    END IF;
  ELSE
    Present_State <= S0;
  END IF;
END IF;
END U_Moore;
END;

Moore FSM in VHDL (3)

WHEN S1 =>
  IF input = '0' THEN
    Present_State <= S2;
  ELSE
    Present_State <= S1;
  END IF;
WHEN S2 =>
  IF input = '1' THEN
    Present_State <= S1;
  ELSE
    Present_State <= S0;
  END IF;
END CASE;
END IF;
END PROCESS;

Output <= '1' WHEN Present_State = S2 ELSE '0';

END behavioral;
ASM Chart of Mealy Machine
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY FSM_Mealy_1 IS
  PORT ( clk : IN STD_LOGIC ;
         reset : IN STD_LOGIC ;
         input : IN STD_LOGIC ;
         output : OUT STD_LOGIC ) ;
END FSM_Mealy_1 ;
ARCHITECTURE behavioral of FSM_Mealy_1 IS
TYPE state IS (S0, S1);
SIGNAL Present_State: state;

BEGIN
U_Mealy: PROCESS(clk, reset)
BEGIN
   IF(reset = '1') THEN
      Present_State <= S0;
   ELSIF rising_edge(clk) THEN
      CASE Present_State IS
         WHEN S0 =>
            IF input = '1' THEN
                Present_State <= S1;
            ELSE
                Present_State <= S0;
            END IF;
         ELSE
            Present_State <= S0;
        END CASE;
   END IF;
END PROCESS U_Mealy;
END behavioral;
Mealy FSM in VHDL (3)

WHEN S1 =>
    IF input = '0' THEN
        Present_State <= S0;
    ELSE
        Present_State <= S1;
    END IF;
END CASE;
END IF;
END PROCESS;

Output <= '1' WHEN (Present_State = S1 AND input = '0') ELSE '0';

END behavioral;
Finite State Machines in VHDL Style 2
Alternative Coding Style

Process(Present State, Input)

Process(clk, reset)

Based on RTL Hardware Design by P. Chu
ASM Chart of Moore Machine
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY FSM_Moore_2 IS
    PORT ( clk : IN    STD_LOGIC ;
          reset : IN    STD_LOGIC ;
          input : IN    STD_LOGIC ;
          output : OUT   STD_LOGIC ) ;
END FSM_Moore_2 ;
ARCHITECTURE behavioral of FSM_Moore_2 IS
TYPE state IS (S0, S1, S2);
SIGNAL Present_State, Next_State: state;

BEGIN

U_Moore: PROCESS (clk, reset)
BEGIN
  IF(reset = '1') THEN
    Present_State <= S0;
    ELSIF rising_edge(clk) THEN
      Present_State <= Next_State;
  END IF;
END PROCESS;

Moore FSM in VHDL (2)

Next_State_Output:
PROCESS (Present_State, input)
BEGIN
    Next_State <= Present_State;
    output <= '0';
    CASE Present_State IS
        WHEN S0 =>
            IF input = '1' THEN
                Next_State <= S1;
            ELSE
                Next_State <= S0;
            END IF;
    END CASE;
END PROCESS;
WHEN S1 =>
    IF input = '0' THEN
        Next_State <= S2;
    ELSE
        Next_State <= S1;
    END IF;
WHEN S2 =>
    output <= '1';
    IF input = '1' THEN
        Next_State <= S1;
    ELSE
        Next_State <= S0;
    END IF;
END CASE;
END PROCESS;

END behavioral;
ASM Chart of Mealy Machine
Mealy FSM in VHDL (1)

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY FSM_Mealy_2 IS
  PORT ( clk : IN STD_LOGIC ;
         reset : IN STD_LOGIC ;
         input : IN STD_LOGIC ;
         output : OUT STD_LOGIC ) ;
END FSM_Mealy_2 ;
ARCHITECTURE behavioral of FSM_Mealy_2 IS
TYPE state IS (S0, S1);
SIGNAL Present_State, Next_State: state;
BEGIN
U_Mealy: PROCESS(clk, reset)
BEGIN
   IF(reset = '1') THEN
      Present_State <= S0;
   ELSIF rising_edge(clk) THEN
      Present_State <= Next_State;
   END IF;
END PROCESS;

Mealy FSM in VHDL (2)

Next_State_Output:
PROCESS (Present_State, input)
BEGIN
    Next_State <= Present_State;
    output <= '0';
    CASE Present_State IS
    WHEN S0 =>
        IF input = '1' THEN
            Next_State <= S1;
        ELSE
            Next_State <= S0;
        END IF;
    END CASE;
END PROCESS;

WHEN S1 =>
  IF input = '0' THEN
    Next_State <= S0;
    Output <= '1' ;
  ELSE
    Next_State <= S1;
  END IF;
END CASE;
END PROCESS;

END behavioral;
Control Unit Example: Arbiter (1)
ASM Chart for Control Unit - Example 4

Reset

Idle

\( r_1 \)

\( r_2 \)

\( r_3 \)

\( gnt_1 \)

\( g_1 \)

\( gnt_2 \)

\( g_2 \)

\( gnt_3 \)

\( g_3 \)

\( r_1 \)

\( r_2 \)

\( r_3 \)
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY arbiter IS
  PORT ( Clk, Reset : IN STD_LOGIC ;
         r       : IN STD_LOGIC_VECTOR(1 TO 3) ;
         g       : OUT STD_LOGIC_VECTOR(1 TO 3) ) ;
END arbiter ;

ARCHITECTURE Behavior OF arbiter IS
  TYPE State_type IS (Idle, gnt1, gnt2, gnt3) ;
  SIGNAL y : State_type ;
BEGIN
  PROCESS ( Reset, Clk )
  BEGIN
    IF Reset = '1' THEN y <= Idle ;
    ELSIF rising_edge(Clk) THEN
      CASE y IS
        WHEN Idle =>
          IF r(1) = '1' THEN y <= gnt1 ;
          ELSIF r(2) = '1' THEN y <= gnt2 ;
          ELSIF r(3) = '1' THEN y <= gnt3 ;
          ELSE y <= Idle ;
          END IF ;
        WHEN gnt1 =>
          IF r(1) = '0' THEN y <= Idle ;
          ELSE y <= gnt1 ;
          END IF ;
      END CASE ;
  END PROCESS ;
END
WHEN gnt2 =>  
  IF r(2) = '0' THEN y <= Idle ;  
  ELSE y <= gnt2 ;  
  END IF ;  
WHEN gnt3 =>  
  IF r(3) = '0' THEN y <= Idle ;  
  ELSE y <= gnt3 ;  
  END IF ;  
  END CASE ;  
  END IF ;  
END PROCESS ;  
g(1) <= '1' WHEN y = gnt1 ELSE '0' ;  
g(2) <= '1' WHEN y = gnt2 ELSE '0' ;  
g(3) <= '1' WHEN y = gnt3 ELSE '0' ;  
END Behavior ;
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY arbiter IS
  PORT ( Clk, Reset : IN STD_LOGIC ;
         r    : IN STD_LOGIC_VECTOR(1 TO 3) ;
         g    : OUT STD_LOGIC_VECTOR(1 TO 3) ) ;
END arbiter ;

ARCHITECTURE Behavior OF arbiter IS
  TYPE State_type IS (Idle, gnt1, gnt2, gnt3) ;
  SIGNAL y, y_next : State_type ;
BEGIN
  PROCESS ( Reset, Clk )
  BEGIN
    IF Reset = '1' THEN
      y <= Idle ;
    ELSEIF rising_edge(Clk) THEN
      y <= y_next;
    END IF;
  END IF;
END PROCESS;
PROCESS (y, r)
BEGIN
  y_next <= y;
g <= "000";
CASE y IS
  WHEN Idle =>
    IF r(1) = '1' THEN y_next <= gnt1 ;
    ELSIF r(2) = '1' THEN y_next <= gnt2 ;
    ELSIF r(3) = '1' THEN y_next <= gnt3 ;
    ELSE y_next <= Idle ;
    END IF ;
  WHEN gnt1 =>
    g(1) <= '1' ;
    IF r(1) = '0' THEN y_next <= Idle ;
    ELSE y_next <= gnt1 ;
    END IF ;
WHEN gnt2 =>
  g(2) <= '1' ;
  IF r(2) = '0' THEN y_next <= Idle ;
  ELSE y_next <= gnt2 ;
  END IF ;
WHEN gnt3 =>
  g(2) <= '1' ;
  IF r(3) = '0' THEN y_next <= Idle ;
  ELSE y_next <= gnt3 ;
  END IF ;
END CASE ;
END PROCESS ;

END Behavior ;
Problem 1

Assuming ASM chart given on the next slide, supplement timing waveforms given in the answer sheet with the correct values of signals State, g1, g2, g3, in the interval from 0 to 575 ns.
ASM Chart

```
Reset

Idle

r1 1
0

0

r2 1
0

0

r3 1
0

gnt1 1

1

1

0

0

0

0

gnt2 1

1

1

0

0

0

gnt3 1

1

1

0

0

0

0

0

0
```

37
Problem 2

Assuming state diagram given on the next slide, supplement timing waveforms given in the answer sheet with the correct values of signals State and c, in the interval from 0 to 575 ns.
ASM Summary by Prof. Chu

- ASM (algorithmic state machine) chart
  - Flowchart-like diagram
  - Provides the same info as a state diagram
  - More descriptive, better for complex description
  - ASM block
    - One state box
    - One or more optional decision boxes: with 1 (T) or 0 (F) exit path
    - One or more conditional output boxes: for Mealy output
Figure 10.4 ASM block.
ASM Chart Rules

• Difference between a regular flowchart and an ASM chart:
  – Transition governed by clock
  – Transition occurs between ASM blocks

• Basic rules:
  – For a given input combination, there is one unique exit path from the current ASM block
  – Any closed loop in an ASM chart must include a state box
Incorrect ASM Charts

(a) 

(b) 

Based on RTL Hardware Design by P. Chu
Generalized FSM

Based on RTL Hardware Design by P. Chu