VHDL
Design with Algorithmic State
Machine (ASM) Charts

EL 310
Erkay Savaş
Sabancı University
Motivation

• Sequential systems are complex and require a formal notation to describe their functionality.
• From this formal notation, a state table and hence Boolean expressions can be derived.
• Algorithmic State Machine (ASM) charts provide a less ambiguous description of a sequential system than state diagrams.
  - State diagrams do not provide explicit timing information.
  - For example, when an output signal is assigned a new value is sometimes not clear.
State Diagram for a Traffic Signal Controller

- Major road
- Minor road
- Sensor

States:
- car/start_timer
- major=G, minor=R
- major=R, minor=G

Transitions:
- From major=G, minor=R to major=R, minor=G
- From major=R, minor=G to major=G, minor=R
- From car/start_timer to major=G, minor=R
- From major=R, minor=G to timed'
- From major=G, minor=R to car'
- From timed to major=R, minor=G

ASM charts resemble flow charts, but contain implicit timing information.

ASM charts represent real hardware.

Hardware cannot suddenly start or stop.

Therefore, all transitions within ASM charts must form closed paths (except a reset signal).
Components of ASM Charts

- state box

The state represented by the state box takes a clock cycle to complete.
- The output signals in the box take the specified values during this clock cycle.
- The notation $X \leftarrow 1$ means that the signal is assigned during the next clock cycle and holds its value until otherwise set elsewhere.
Components of ASM Charts

- Decision box

- Decision box has two or more branches going out.
- Decision is made based on the value of one or more input signals (e.g. signal J)
- Decision box must follow and be associated with a state box.
- Thus, the decision is made in the same clock cycle as the other actions of the state.
- Hence, the input signals must be available and valid at the start of the clock cycle.
Components of ASM Charts

- Conditional output box

- A conditional output box must follow a decision box.
- A conditional output box is attached to a state box through one or more decision boxes.
- Therefore, the output signals in the conditional output box are asserted in the same clock cycle as those in the state box to which it is attached.
- The output signals can change during that state as a result of changes on the inputs.
- The conditional output signals are sometimes referred as Mealy outputs since they depend on the input signals as well.
A State in ASM Charts

- One state, or clock cycle, consists of more than just the state box.
- Decision and conditional output boxes are also a part of the state.

```
major_green = 1
major_green = 0
```

```
minor_green = 0
minor_green = 1
```

```
car
```

```
start_timer
```

```
timed
```

1 clock cycle
State Box vs. Conditional Output Box

(a) (b)
State Box vs. Conditional Output Box

Clock

Z
Y, C=1
Y, C=0
W

Z
Y, C=1
W, C=1
Y, C=0
W, C=0

C is tested here

(a)

(b)
# Synthesis from ASM Charts

## State and output table

<table>
<thead>
<tr>
<th>Present State</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>G,0</td>
<td>G,0</td>
<td>R,1</td>
<td>R,1</td>
</tr>
<tr>
<td>R</td>
<td>R,0</td>
<td>G,0</td>
<td>G,0</td>
<td>R,0</td>
</tr>
</tbody>
</table>

Next State, start_timer

## Transition and output table

<table>
<thead>
<tr>
<th>Q</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0,0</td>
<td>0,0</td>
<td>1,1</td>
<td>1,1</td>
</tr>
<tr>
<td>1</td>
<td>1,0</td>
<td>0,0</td>
<td>0,0</td>
<td>1,0</td>
</tr>
</tbody>
</table>

Q⁺, start_timer
**K-Maps for Traffic Signal Controller**

<table>
<thead>
<tr>
<th>car, timed</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ Q^+ = Q' \text{ car} + Q \text{ timed}' \]

**start_timer = Q' car**
The flip-flop outputs can be used directly to control the traffic signals.
- \( Q=0 \) ➞ the signal for the major road is green and the signal for the minor road is red.
library IEEE;
use IEEE.std_logic_1164.all;

entity traffic_signals is
port(clock, timed, car: in std_ulogic;
    start_timer, major_green, minor_green: out std_ulogic);
end entity traffic_signals;

architecture asm of traffic_signals is
begin
    process(clock, timed, car) is
        type state_type is (G, R);
        variable state: state_type;
    begin
        start_timer <= '0';
        if (rising_edge(clock)) then
            case state is
                when G =>
                    major_green <= '1'; minor_green <= '0';
                    if (car = '1') then start_timer = '1'; state := R; end if;
                when R =>
                    major_green <= '0'; minor_green <= '1';
                    if (timed = '1') then state := G; end if;
            end case;
        end if;
    end process;
end architecture asm;
Alternative VHDL Code

```vhdl
architecture asm2 of traffic_signals is
  type state_type is (G, R);
  variable state, next_state: state_type;
begin
  seq: process(clock) is -- for sequential part
  begin
    if (rising_edge(clock)) then state <= next_state; end if;
  end process seq;

  com: process(timed, car, state) is -- for combinational part
  begin
    start_timer <= '0';
    case state is
      when G =>
        major_green <= '1'; minor_green <= '0';
        if(car = '1') then
          start_timer = '1'; next_state <= R;
        else
          next_state <= G;
        end if;
      when R =>
        major_green <= '0'; minor_green <= '1';
        if( timed = '1') then next_state <= G;
        else
          next_state <= R;
        end if;
    end case;
  end process com;
end architecture asm2;
```
The multiplier starts when Start = 1

The counter counts the number of shifts and outputs co_out = 1 just before the last shift occurs.

M0 is the LSB of the multiplier
entity multiplier is
port(clock, Start, co_out, M0: in std_ulogic;
    Load, Shift, Done, Add: out std_ulogic);
end entity multiplier;

architecture asm_beh of multiplier is
    signal state, next_state: integer range 0 to 3;
begin
    process(Start, co_out, M0, state) is
        begin
            Load <= '0'; Shift <= '0'; Add <= '0'; -- conditional outputs
                     -- get default values
            case state is
                when 0 =>
                    if Start = '1' then
                        Load <= '1'; next_state <= 1;
                    else
                        next_state <= 0;
                    end if;
                when 1 =>
                    if M0 = '1' then Add <= '1'; next_state <= 2;
                    else
                        Shift <= '1';
                        if co_out = '1' then next_state <= 3;
                        else
                            next_state <= 1;
                        end if;
                    end if;
                ...
            end case;
    end process;
end architecture asm_beh;
architecture asm_beh of multiplier is
  signal state, next_state: integer range 0 to 3;
beginn
process(Start, co_out, M0, state) is
  begin
    Load <= '0'; Shift <= '0'; Add <= '0'; -- conditional outputs
      -- get default values
    case state is
      ...
        when 2 =>
          Shift <= '1';
          if co_out = '1' then next_state <= 3;
          else next_state <= 1;
            end if;
        when 3 => done <= '1'; next_state <= 0;
          end case;
        end process;
    process (clock)is
      begin
        if (rising_edge(clock)) then state <= next_state; end if;
      end process;
    end architecture asm_beh;
ASM Chart for Sequence Detector

A: \( \text{sum} \equiv 0 \mod 3 \)

\[
\begin{align*}
z &= 1 \\
0 &\quad 1 \\
0 &\quad 1 \\
\end{align*}
\]

B: \( \text{sum} \equiv 1 \mod 3 \)

\[
\begin{align*}
z &= 0 \\
0 &\quad 1 \\
0 &\quad 1 \\
\end{align*}
\]

C: \( \text{sum} \equiv 2 \mod 3 \)

\[
\begin{align*}
z &= 0 \\
0 &\quad 1 \\
0 &\quad 1 \\
\end{align*}
\]
entity seq_detector is
port (clock, x, y: in std_ulogic;
    z: out std_ulogic);
end entity seq_detector;

architecture asm_beh of seq_detector is
    type state_type is (state_A, state_B, state_C);
    signal state, next_state: state_type;
begin
    process (x, y, state) is
        begin
            case state is
                when state_A => z <= '1';
                if x = '0' then
                    if y = '0' then next_state <= state_A;
                    else next_state <= state_B;
                    end if;
                else
                    if y = '0' then next_state <= state_B;
                    else next_state <= state_C;
                    end if;
                end if;
            ...
end process;
end architecture asm_beh;
VHDL Code for Sequence Detector

architecture asm_beh of seq_detector is

process(x, y, state) is

case state is

when state_B => z <= '0';
if x = '0' then
  if y = '0' then next_state <= state_B;
  else next_state <= state_C;
  end if;
else
  if y = '0' then next_state <= state_C;
  else next_state <= state_A;
  end if;
end if;

end process;
end architecture asm_beh;
VHDL Code for Sequence Detector

... 

architecture asm_beh of seq_detector is 
... 

process(x, y, state) is 
... 

case state is 
...

when state_C => z <= '0';
if x = '0' then 
if y = '0' then next_state <= state_C;
else next_state <= state_A;
end if;
else 
if y = '0' then next_state <= state_A;
else next_state <= state_B;
end if;
end if;
end case;
end process;

process (clock) is
begin
if (rising_edge(clock)) then state <= next_state; end if;
end process;
end architecture asm_beh;