Introduction to VHDL V

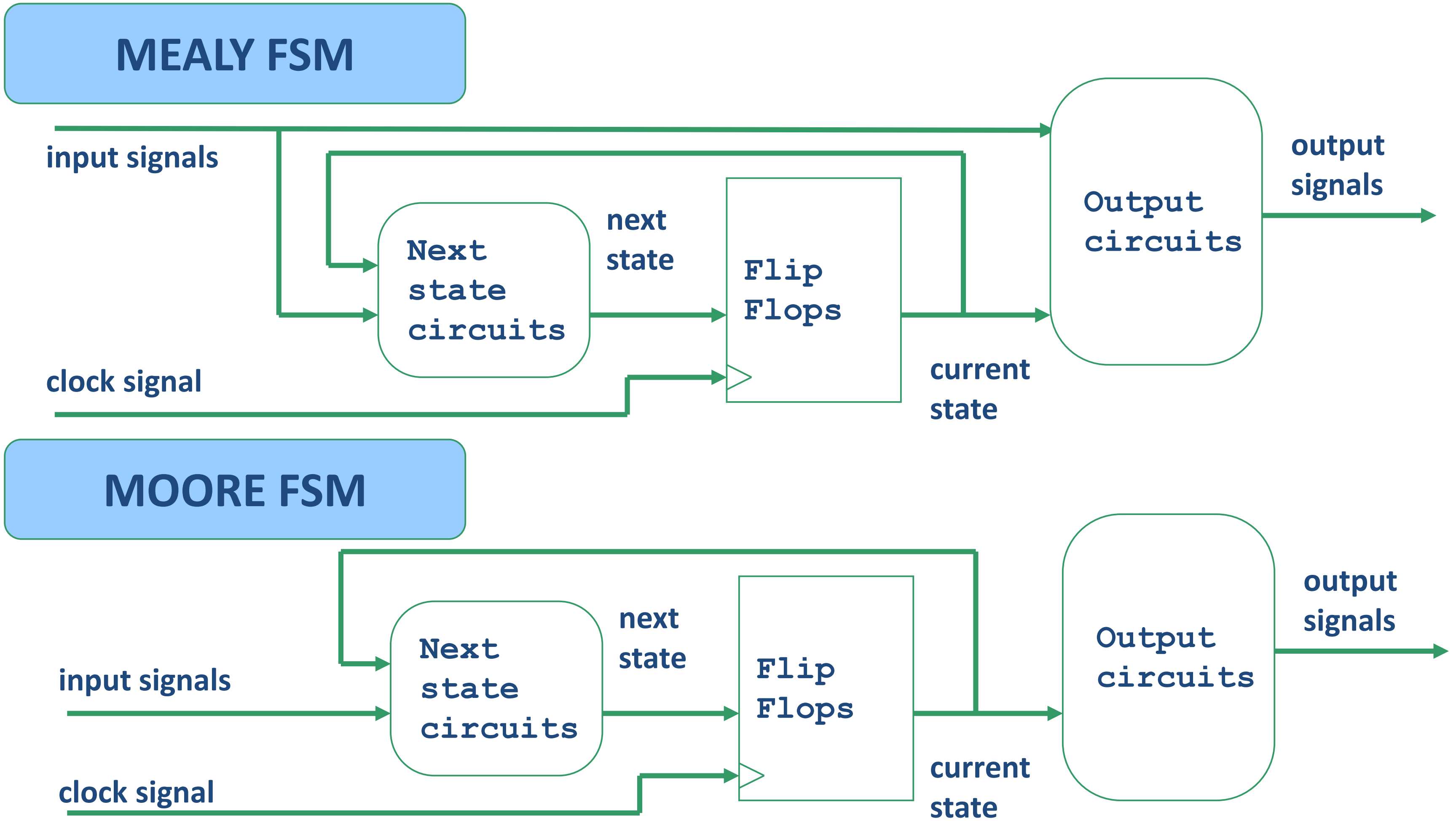
Finite State Machines - Examples

Roland Höller
Email: hoeller@technikum-wien.at

FSM – Finite State Machine

- Why care about FSMs?
  - They are important building blocks of circuits
  - Their proper operation is most often crucial for the design
  - FSM may be optimized very efficiently by special EDA tools
  - FSMs tend to get quite complex

Finite State Machines



State Vector Representation

- There are different solutions how to specify the state vector and describe the FSM
- By structure:
  - Single sequential process
  - Two processes (sequential and combinatorial)
- By state vector type:
  - enumerated type
  - std_logic_vector (binary, one-hot, gray, …)
  - integer

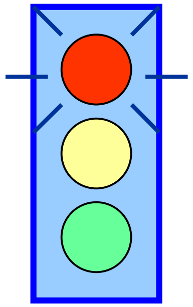
State Vector Representation

- For the purpose of modeling circuits in VHDL there is actually no big difference

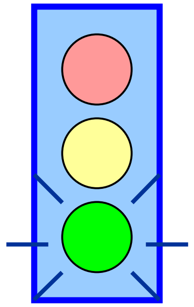
- But there is a difference due to the interpretation of the tools during the design implementation

Example: Traffic Light

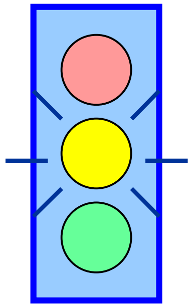
1.



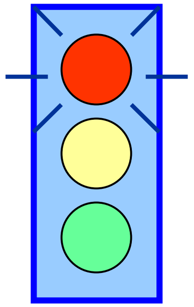
2.
(button pressed)



3.



4.



FSM – One Process

```
   type  t_state is (red, yellow, green);
   signal s_state : t_state;

begin
p_fsm : process (clk_i,reset_i)
begin
   if (reset_i = '1') then
     s_state <= red;
   elsif (clk_i'event and clk_i = '1') then
     case s_state is
       when yellow => s_state <= red;
       when green  => s_state <= yellow;
```

FSM – One Process

```
        when others =>
          if (s_button_i = '1') then
            s_state <= green;
          end if;
      end case;
    end if;
  end process p_fsm;
```

FSM – One Process

- Advantages
  - Enhanced readability (RTL only)
  - Eases code debugging
  - Enables technology specific optimization during synthesis

- Disadvantages
  - No control of implementation
  - After synthesis the enumerated type will be converted to a std_logic_vector type

FSM – Two Processes

```
  type t_state is (red,yellow,green);
  signal s_present_state : t_state;
  signal s_next_state    : t_state;

begin
p_fsm_seq : process (clk_i,reset_i)
begin
  if (reset_i = '1') then
    s_present_state <= red;
  elsif (clk_i'event and clk_i = '1') then
    s_present_state <= s_next_state;
  end if;
end process p_fsm_seq;
```

FSM – Two Processes

```
p_fsm_comb : process(s_present_state,s_button_i)
begin
case s_present_state is
  when yellow =>
    s_next_state <= red;
  when green =>
    s_next_state <= yellow;
  when others =>
    if (s_button_i = '1') then
      s_next_state <= green;
    else                       -- required to avoid
      s_next_state <= red; -- latch insertion !!
    end if;
end case;
end process p_fsm_comb;
```

FSM – Two Processes

- Advantages
  - Most general description of an FSM
  - Both Mealy and Moore type FSMs may be implemented

- Disadvantages
  - More code to write
  - Required complete sensitivity list
  - Sometimes less readable

FSM – Integer Subtype

```
subtype t_state is integer range 0 to 2;
signal  s_state : t_state;

begin
p_fsm : process (clk_i,reset_i)
begin
  if (reset_i = '1') then
    s_state <= 0;
  elsif (clk_i'event and clk_i = '1') then
    case s_state is
      when 1 => s_state <= 0;
      when 2 => s_state <= 1;
```

FSM – Integer Subtype

```
        when others =>
          if (s_button_i = '1') then
            s_state <= 2;
          end if;
      end case;
    end if;
  end process p_fsm;
```

FSM – Integer Subtype

- Advantages
  - Somewhat easier to define
  - Some control of implementation

- Disadvantages
  - Code is less readable
  - Disables technology specific optimization during synthesis
  - After synthesis the enumerated type will be converted to a std_logic_vector type

FSM – Binary Encoding

```
signal s_state : std_logic_vector(1 downto 0);

begin
p_fsm : process (clk_i,reset_i)
begin
  if (reset_i = '1') then
    s_state <= "00";
  elsif (clk_i'event and clk_i = '1') then
    case s_state is
      when "01" => s_state <= "00";
      when "10" => s_state <= "01";
```

FSM – Binary Encoding

```
      when others =>
        if (s_button_i = '1') then
          s_state <= "10";
        end if;
    end case;
  end if;
end process p_fsm;
```

FSM – One-Hot Encoding

```
signal s_state : std_logic_vector(2 downto 0);

begin
p_fsm : process (clk_i,reset_i)
begin
  if (reset_i = '1') then
    s_state <= "000";
  elsif (clk_i'event and clk_i = '1') then
    case s_state is
      when "010" => s_state <= "001";
      when "100" => s_state <= "010";
```

FSM – One-Hot Encoding

```
        when others =>
          if (s_button_i = '1') then
            s_state <= "100";
          end if;
      end case;
    end if;
  end process p_fsm;
```

FSM – Binary / One-Hot Encoding

- Advantages
  - Somewhat easier to define
  - Full control of implementation
  - Same encoding on RTL and post synthesis/layout

- Disadvantages
  - Code is less readable
  - Disables technology specific optimization during synthesis

Summary

- Moore Finite State Machine
- Mealy Finite State Machine
- VHDL Examples

Questions

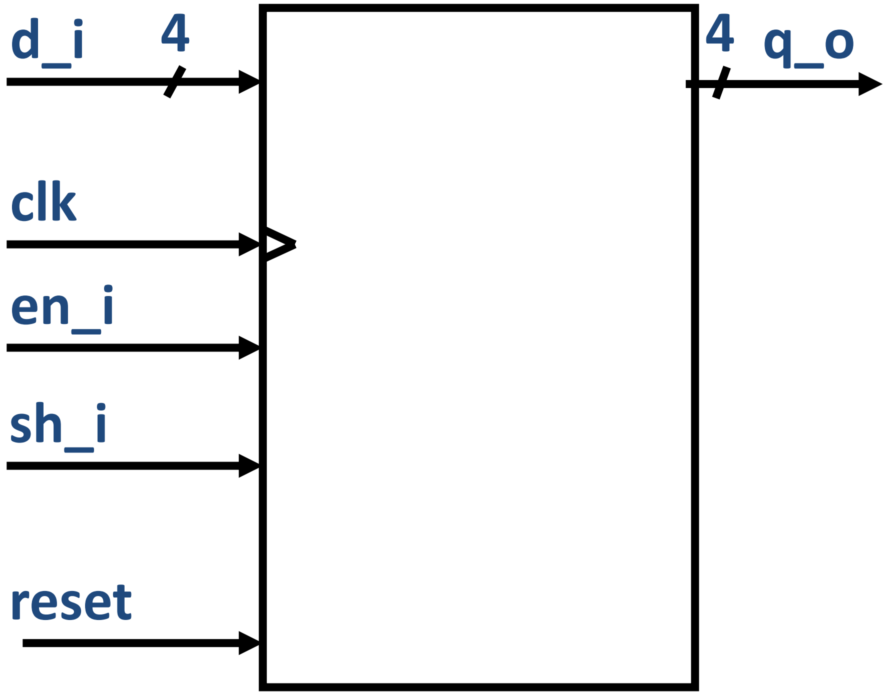
- 1) Which encodings can be used to represent the state vector of an FSM?
  - A binary
  - B gray
  - C one-hot
  - D “0001” -> “0011” -> “0111” -> “1111”-> “0001” -> …
- 2) What is an automaton called, whose outputs are a function of the state vector only?
  - A Moore
  - B Mealy
  - C secure
  - D minimal
- 3) Which VHDL data type is best used for encoding the state vector of a complex FSM?
  - A integer
  - B real
  - C enumeration type
  - D bit_vector

Class Example: 4-bit Shift Register

1) **Describe and simulate a D- Flip Flop in VHDL**

2) **Add the shift capability to the VHDL code**

en_i	sh_i	q_o
0	0	hold
0	1	hold
1	0	d_i -> q_o
1	1	shift left of q_o


d_i
4
4
q_o
clk
en_i
sh_i
reset