

Hardware Modeling [VU] (191.011) – WS25 – Finite-State Machine Implementation (in VHDL)

Florian Huemer & Sebastian Wiedemann & Dylan Baumann

WS 2025/26

Implementing FSMs in VHDL

HWMod
WS25

FSM Imple-
mentation
Implementation
Steps
Example

- Conversion of FSM model to VHDL code follows simple steps

Implementing FSMs in VHDL

HWMod
WS25

FSM Imple-
mentation
Implementation
Steps
Example

- Conversion of FSM model to VHDL code follows simple steps
 - 1 Create an enumeration type for the FSM state

Implementing FSMs in VHDL

HWMod
WS25

FSM Imple-
mentation
Implementation
Steps
Example

- Conversion of FSM model to VHDL code follows simple steps
 - 1 Create an enumeration type for the FSM state
 - 2 Create a record type for the state register

Implementing FSMs in VHDL

HWMod
WS25

FSM Imple-
mentation
Implementation
Steps
Example

- Conversion of FSM model to VHDL code follows simple steps
 - 1 Create an enumeration type for the FSM state
 - 2 Create a record type for the state register
 - 3 Create signals for the current (`s`) and next (`s_next`) register value

Implementing FSMs in VHDL

HWMod
WS25

FSM Imple-
mentation
Implementation
Steps
Example

- Conversion of FSM model to VHDL code follows simple steps
 - 1 Create an enumeration type for the FSM state
 - 2 Create a record type for the state register
 - 3 Create signals for the current (`s`) and next (`s_next`) register value
 - 4 Implement the state register in a sync. process

Implementing FSMs in VHDL

HWMod
WS25

FSM Imple-
mentation
Implementation
Steps
Example

- Conversion of FSM model to VHDL code follows simple steps
 - 1 Create an enumeration type for the FSM state
 - 2 Create a record type for the state register
 - 3 Create signals for the current (`s`) and next (`s_next`) register value
 - 4 Implement the state register in a sync. process
 - 5 Implement the next-state and output logic in comb. process(es)

Implementing FSMs in VHDL

HWMod
WS25

FSM Imple-
mentation
Implementation
Steps
Example

- Conversion of FSM model to VHDL code follows simple steps
 - 1 Create an enumeration type for the FSM state
 - 2 Create a record type for the state register
 - 3 Create signals for the current (`s`) and next (`s_next`) register value
 - 4 Implement the state register in a sync. process
 - 5 Implement the next-state and output logic in comb. process(es)
- 2- vs. 3-process method
 - Overlap of next-state and output logic $\Rightarrow$ 2-process usually more readable
 - Recommendation: Use the 2-process method

Example: FSM Types

HWMod
WS25

FSM Implementation

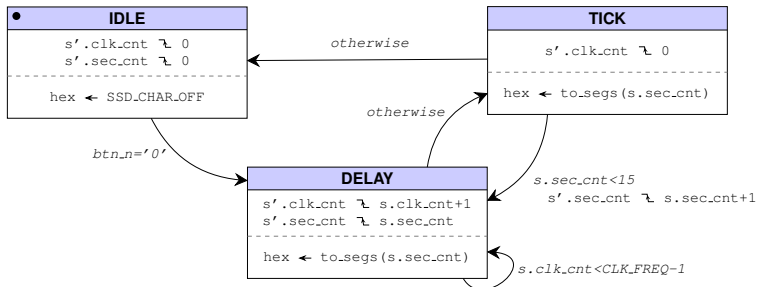
Implementation
Steps

Example

FSM Types

State Register

Comb. Logic

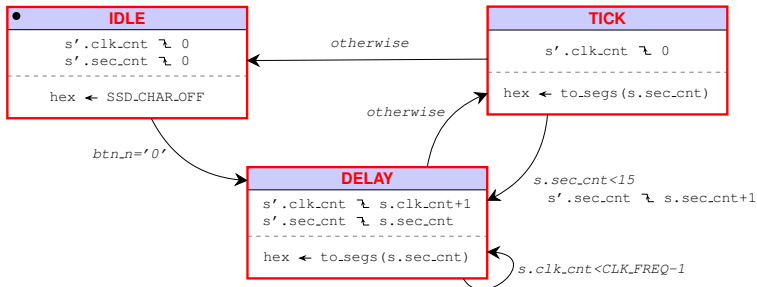


Example: FSM Types

HWMod
WS25

1 Create an enumeration type for the FSM state

```
20  type fsm_state_t is (IDLE, DELAY, TICK);
```



FSM Imple-
mentation

Implementation
Steps

Example

FSM Types

State Register

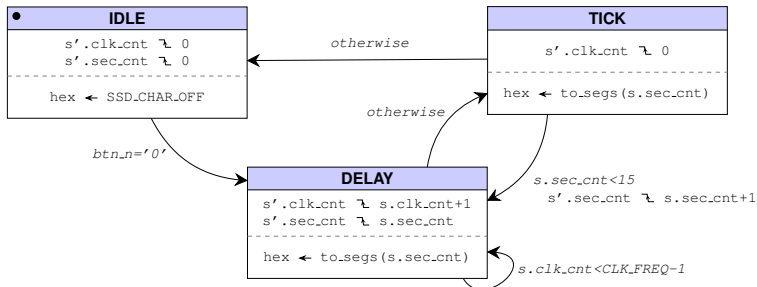
Comb. Logic

Example: FSM Types

HwMod
WS25

2 Create a record type for the state register

```
20 type fsm_state_t is (IDLE, DELAY, TICK);  
21 type state_reg_t is record  
22     state : fsm_state_t;
```



FSM Imple-
mentation

Implementation
Steps

Example

FSM Types

State Register

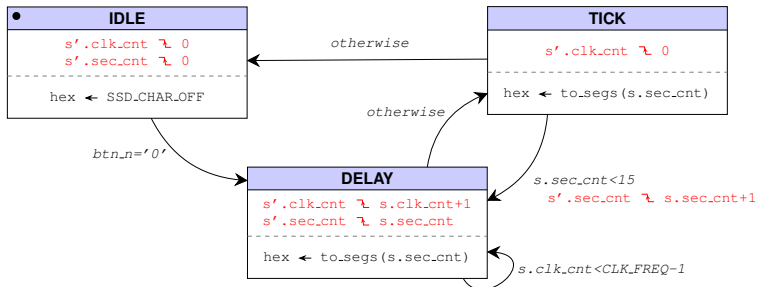
Comb. Logic

Example: FSM Types

HWMod
WS25

2 Create a record type for the state register

```
20 type fsm_state_t is (IDLE, DELAY, TICK);
21 type state_reg_t is record
22     state : fsm_state_t;
23     clk_cnt : unsigned(log2c(CLK_FREQ) downto 0);
24     tick_cnt : unsigned(3 downto 0);
25 end record;
```



FSM Imple-
mentation

Implementation
Steps

Example

FSM Types

State Register

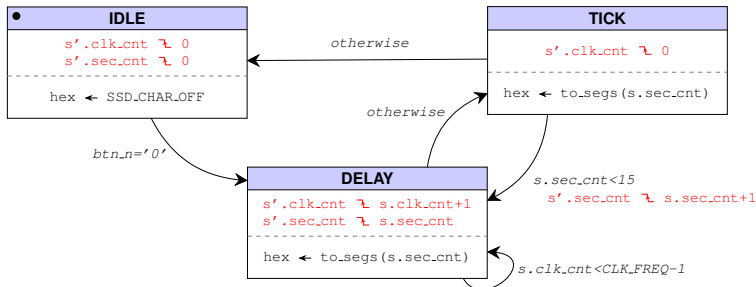
Comb. Logic

Example: FSM Types

HWMod
WS25

2 Create a record type for the state register

```
20 type fsm_state_t is (IDLE, DELAY, TICK);
21 type state_reg_t is record
22     state : fsm_state_t;
23     clk_cnt : unsigned(log2c(CLK_FREQ) downto 0);
24     tick_cnt : unsigned(3 downto 0);
25 end record;
```



FSM Imple-
mentation

Implementation
Steps

Example

FSM Types

State Register

Comb. Logic

Example: State Register

HWMod
WS25

FSM Imple-
mentation

Implementation
Steps

Example

FSM Types

State Register

Comb. Logic

3 Create signals for the current (`s`) and next (`s_next`) register value

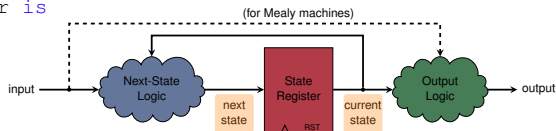
```
18 architecture arch of advanced_timer is
19     -- enum and record declarations
```

Example: State Register

HWMod
WS25

3 Create signals for the current (s) and next (s_nxt) register value

```
18 architecture arch of advanced_timer is
19   -- enum and record declarations
```



FSM Imple-
mentation

Implementation
Steps

Example

FSM Types

State Register

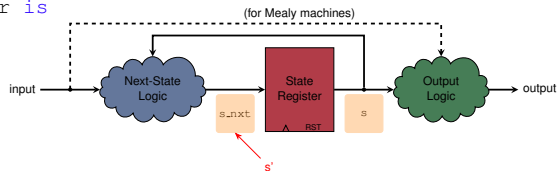
Comb. Logic

Example: State Register

HWMod
WS25

3 Create signals for the current (s) and next (s_next) register value

```
18 architecture arch of advanced_timer is
19   -- enum and record declarations
26   signal s, s_next : state_reg_t;
27 begin
```



FSM Imple-
mentation

Implementation
Steps

Example

FSM Types

State Register

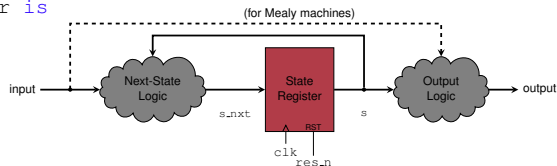
Comb. Logic

Example: State Register

HWMod
WS25

4 Implement the state register in a sync. process

```
18 architecture arch of advanced_timer is
19   -- enum and record declarations
26   signal s, s_nxt : state_reg_t;
27 begin
```



FSM Imple-
mentation

Implementation
Steps

Example

FSM Types

State Register

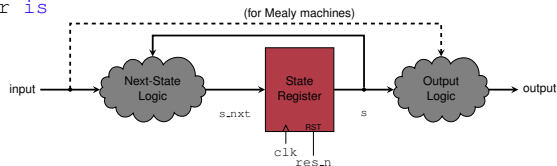
Comb. Logic

Example: State Register

HWMod
WS25

4 Implement the state register in a sync. process

```
18 architecture arch of advanced_timer is
19   -- enum and record declarations
26   signal s, s_nxt : state_reg_t;
27 begin
28   state_reg : process(clk, res_n)
29   begin
30     if res_n = '0' then
31       s <= (state => IDLE, others => (others => '0'));
32     elsif rising_edge(clk) then
33       s <= s_nxt;
34     end if;
35   end process;
```



FSM Imple-
mentation

Implementation
Steps

Example

FSM Types

State Register

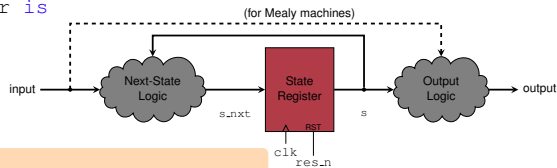
Comb. Logic

Example: State Register

HWMod
WS25

4 Implement the state register in a sync. process

```
18 architecture arch of advanced_timer is
19   -- enum and record declarations
26   signal s, s_nxt : state_reg_t;
27 begin
28   state_reg : process(clk, res_n)
29   begin
30     if res_n = '0' then
31       s <= (state => IDLE, others => (others => '0'));
32     elsif rising_edge(clk) then
33       s <= s_nxt;
34     end if;
35   end process;
```



FSM Imple-
mentation

Implementation
Steps

Example

FSM Types

State Register

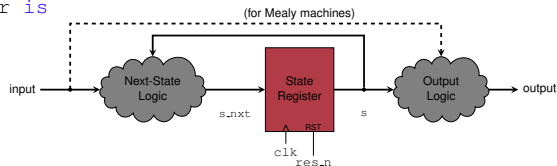
Comb. Logic

Example: State Register

HWMod
WS25

4 Implement the state register in a sync. process

```
18 architecture arch of advanced_timer is
19   -- enum and record declarations
26   signal s, s_nxt : state_reg_t;
27 begin
28   state_reg : process(clk, res_n)
29   begin
30     if res_n = '0' then
31       s <= (state => IDLE, others => (others => '0'));
32     elsif rising_edge(clk) then
33       s <= s_nxt;
34     end if;
35   end process;
```



FSM Imple-
mentation

Implementation
Steps

Example

FSM Types

State Register

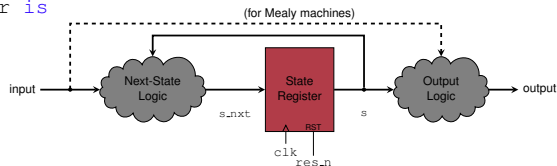
Comb. Logic

Example: State Register

HWMod
WS25

4 Implement the state register in a sync. process

```
18 architecture arch of advanced_timer is
19   -- enum and record declarations
26   signal s, s_nxt : state_reg_t;
27 begin
28   state_reg : process(clk, res_n)
29   begin
30     if res_n = '0' then
31       s <= (state => IDLE, others => (others => '0'));
32     elsif rising_edge(clk) then
33       s <= s_nxt;
34     end if;
35   end process;
```



FSM Imple-
mentation

Implementation
Steps

Example

FSM Types

State Register

Comb. Logic

Example: Next-State and Output Logic

HWMod
WS25

FSM Imple-
mentation

Implementation
Steps

Example

FSM Types

State Register

Comb. Logic

5 Implement the next-state and output logic in comb. process

```
36 comb : process(all)
37 begin
38     s_nxt <= s;
39     hex <= to_segs(std_ulogic_vector(s.tick_cnt));
```

Example: Next-State and Output Logic

HWMod
WS25

FSM Imple-
mentation

Implementation
Steps

Example

FSM Types

State Register

Comb. Logic

- 5 Implement the next-state and output logic in comb. process
 - **Recall:** Path without assignment through comb. process $\Rightarrow$ latch!

```
36 comb : process(all)
37 begin
38     s_nxt <= s;
39     hex <= to_segs(std_ulogic_vector(s.tick_cnt));
```

Example: Next-State and Output Logic

HWMod
WS25

FSM Imple-
mentation

Implementation
Steps

Example

FSM Types

State Register

Comb. Logic

- 5 Implement the next-state and output logic in comb. process
 - **Recall:** Path without assignment through comb. process $\Rightarrow$ latch!
 - $\Rightarrow$ Recommendation: Use default assignments

```
36 comb : process(all)
37 begin
38     s_nxt <= s;
39     hex <= to_segs(std_ulogic_vector(s.tick_cnt));
```

Example: Next-State and Output Logic

HWMod
WS25

FSM Imple-
mentation

Implementation
Steps

Example

FSM Types

State Register

Comb. Logic

- 5 Implement the next-state and output logic in comb. process
 - **Recall:** Path without assignment through comb. process $\Rightarrow$ latch!
 - $\Rightarrow$ Recommendation: Use default assignments
 - Define next-state and output logic per state

```
36 comb : process(all)
37 begin
38     s_nxt <= s;
39     hex <= to_segs(std_ulogic_vector(s.tick_cnt));
40
41     case s.state is
```

Example: Next-State and Output Logic

HWMod
WS25

FSM Imple-
mentation

Implementation
Steps

Example

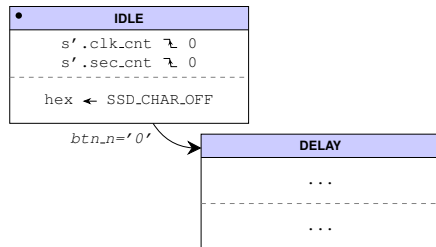
FSM Types

State Register

Comb. Logic

- 5 Implement the next-state and output logic in comb. process
- **Recall:** Path without assignment through comb. process $\Rightarrow$ latch!
- $\Rightarrow$ Recommendation: Use default assignments
- Define next-state and output logic per state

```
36 comb : process(all)
37 begin
38     s_nxt <= s;
39     hex <= to_segs(std_ulogic_vector(s.tick_cnt));
40
41     case s.state is
42         when IDLE =>
```



Example: Next-State and Output Logic

HWMod
WS25

FSM Imple-
mentation

Implementation
Steps

Example

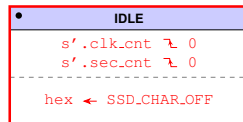
FSM Types

State Register

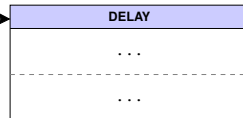
Comb. Logic

- 5 Implement the next-state and output logic in comb. process
- **Recall:** Path without assignment through comb. process $\Rightarrow$ latch!
- $\Rightarrow$ Recommendation: Use default assignments
- Define next-state and output logic per state

```
36 comb : process(all)
37 begin
38     s_nxt <= s;
39     hex <= to_segs(std_ulogic_vector(s.tick_cnt));
40
41     case s.state is
42         when IDLE =>
43             s_nxt.clk_cnt <= (others => '0');
44             s_nxt.tick_cnt <= (others => '0');
45             hex <= SSD_CHAR_OFF;
```



$btn_n = '0'$



Example: Next-State and Output Logic

HWMod
WS25

FSM Imple-
mentation

Implementation
Steps

Example

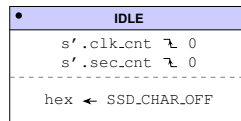
FSM Types

State Register

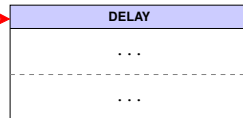
Comb. Logic

- 5 Implement the next-state and output logic in comb. process
 - **Recall:** Path without assignment through comb. process $\Rightarrow$ latch!
 - $\Rightarrow$ Recommendation: Use default assignments
 - Define next-state and output logic per state

```
36 comb : process(all)
37 begin
38     s_nxt <= s;
39     hex <= to_segs(std_ulogic_vector(s.tick_cnt));
40
41     case s.state is
42         when IDLE =>
43             s_nxt.clk_cnt <= (others => '0');
44             s_nxt.tick_cnt <= (others => '0');
45             hex <= SSD_CHAR_OFF;
46             if btn_n = '0' then
47                 s_nxt.state <= DELAY;
48             end if;
```



btn_n='0'



Example: Next-State and Output Logic

HWMod
WS25

FSM Imple-
mentation

Implementation
Steps

Example

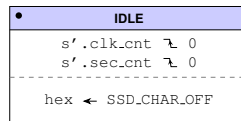
FSM Types

State Register

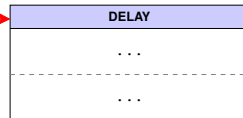
Comb. Logic

- 5 Implement the next-state and output logic in comb. process
 - **Recall:** Path without assignment through comb. process $\Rightarrow$ latch!
 - $\Rightarrow$ Recommendation: Use default assignments
 - Define next-state and output logic per state

```
36 comb : process(all)
37 begin
38     s_nxt <= s;
39     hex <= to_segs(std_ulogic_vector(s.tick_cnt));
40
41     case s.state is
42         when IDLE =>
43             s_nxt.clk_cnt <= (others => '0');
44             s_nxt.tick_cnt <= (others => '0');
45             hex <= SSD_CHAR_OFF;
46             if btn_n = '0' then
47                 s_nxt.state <= DELAY;
48             end if;
```



btn_n='0'



Example: Next-State and Output Logic (cont'd)

HWMoD
WS25

FSM Imple-
mentation

Implementation
Steps

Example

FSM Types

State Register

Comb. Logic

```
36 comb : process (all)
37 begin
38     s_nxt <= s;
39     hex <= to_segs(std_ulogic_vector(s.tick_cnt));
40
41     case s.state is
42         when IDLE =>
43             s_nxt.clk_cnt <= (others => '0');
44             s_nxt.tick_cnt <= (others => '0');
45             hex <= SSD_CHAR_OFF;
46             if btn_n = '0' then
47                 s_nxt.state <= DELAY;
48             end if;
49
50         when DELAY =>
51             if s.clk_cnt < CLK_FREQ-1 then
52                 s_nxt.clk_cnt <= s.clk_cnt + 1;
53             else
54                 s_nxt.state <= TICK;
55             end if;
56         when TICK =>
57             s_nxt.clk_cnt <= (others => '0');
58             s_nxt.state <= DELAY;
59             if s.tick_cnt < 15 then
60                 s_nxt.tick_cnt <= s.tick_cnt + 1;
61             else
62                 s_nxt.state <= IDLE;
63             end if;
```

Example: Next-State and Output Logic (cont'd)

HWMod
WS25

FSM Imple-
mentation

Implementation
Steps

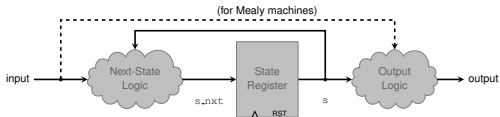
Example

FSM Types

State Register

Comb. Logic

```
36 comb : process (all)
37 begin
38     s_nxt <= s;
39     hex <= to_segs(std_ulogic_vector(s.tick_cnt));
40
41     case s.state is
42         when IDLE =>
43             s_nxt.clk_cnt <= (others => '0');
44             s_nxt.tick_cnt <= (others => '0');
45             hex <= SSD_CHAR_OFF;
46             if btn_n = '0' then
47                 s_nxt.state <= DELAY;
48             end if;
49
50         when DELAY =>
51             if s.clk_cnt < CLK_FREQ-1 then
52                 s_nxt.clk_cnt <= s.clk_cnt + 1;
53             else
54                 s_nxt.state <= TICK;
55             end if;
56         when TICK =>
57             s_nxt.clk_cnt <= (others => '0');
58             s_nxt.state <= DELAY;
59             if s.tick_cnt < 15 then
60                 s_nxt.tick_cnt <= s.tick_cnt + 1;
61             else
62                 s_nxt.state <= IDLE;
63             end if;
```



Example: Next-State and Output Logic (cont'd)

HWMod
WS25

FSM Imple-
mentation

Implementation
Steps

Example

FSM Types

State Register

Comb. Logic

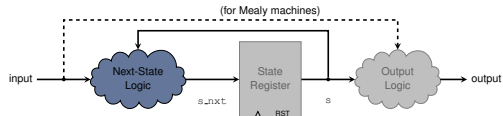
```
36 comb : process (all)
37 begin
38     s_nxt <= s;
39     hex <= to_segs(std_ulogic_vector(s.tick_cnt));
```

```
41 case s.state is
42     when IDLE =>
43         s_nxt.clk_cnt <= (others => '0');
44         s_nxt.tick_cnt <= (others => '0');
45         hex <= SSD_CHAR_OFF;
46         if btn_n = '0' then
47             s_nxt.state <= DELAY;
48         end if;
```

```
49
50 when DELAY =>
51     if s.clk_cnt < CLK_FREQ-1 then
52         s_nxt.clk_cnt <= s.clk_cnt + 1;
53     else
54         s_nxt.state <= TICK;
55     end if;
```

```
55 when TICK =>
```

```
56     s_nxt.clk_cnt <= (others => '0');
57     s_nxt.state <= DELAY;
58     if s.tick_cnt < 15 then
59         s_nxt.tick_cnt <= s.tick_cnt + 1;
60     else
61         s_nxt.state <= IDLE;
62     end if;
```



Example: Next-State and Output Logic (cont'd)

HWMoD
WS25

FSM Imple-
mentation

Implementation
Steps

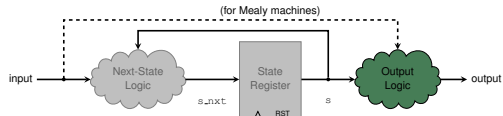
Example

FSM Types

State Register

Comb. Logic

```
36 comb : process(all)
37 begin
38     s_nxt <= s;
39     hex <= to_segs(std_ulogic_vector(s.tick_cnt));
40
41     case s.state is
42         when IDLE =>
43             s_nxt.clk_cnt <= (others => '0');
44             s_nxt.tick_cnt <= (others => '0');
45             hex <= SSD_CHAR_OFF;
46             if btn_n = '0' then
47                 s_nxt.state <= DELAY;
48             end if;
49
50         when DELAY =>
51             if s.clk_cnt < CLK_FREQ-1 then
52                 s_nxt.clk_cnt <= s.clk_cnt + 1;
53             else
54                 s_nxt.state <= TICK;
55             end if;
56         when TICK =>
57             s_nxt.clk_cnt <= (others => '0');
58             s_nxt.state <= DELAY;
59             if s.tick_cnt < 15 then
60                 s_nxt.tick_cnt <= s.tick_cnt + 1;
61             else
62                 s_nxt.state <= IDLE;
63             end if;
```



Lecture Complete!