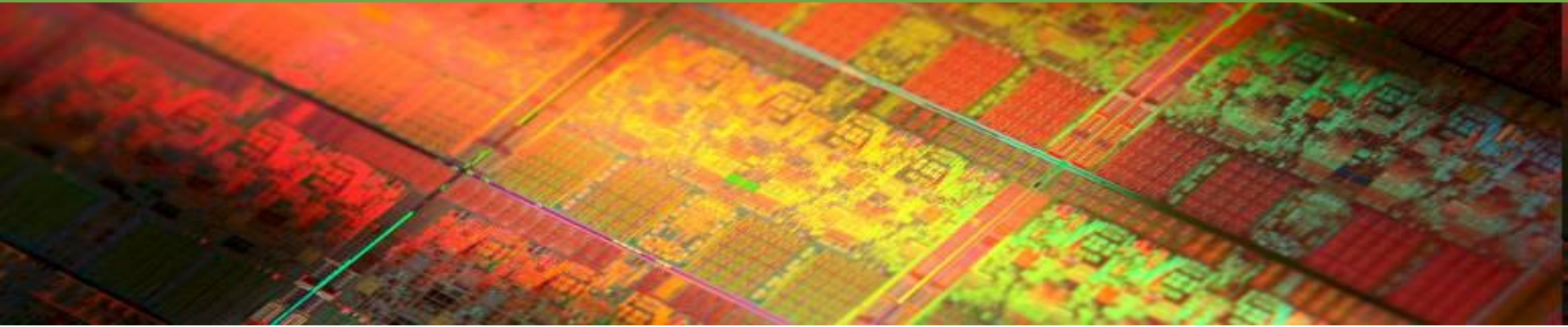


EE 2000

Logic Circuit Design (Fall 2018)



Tutorial 6

Mealy Machine

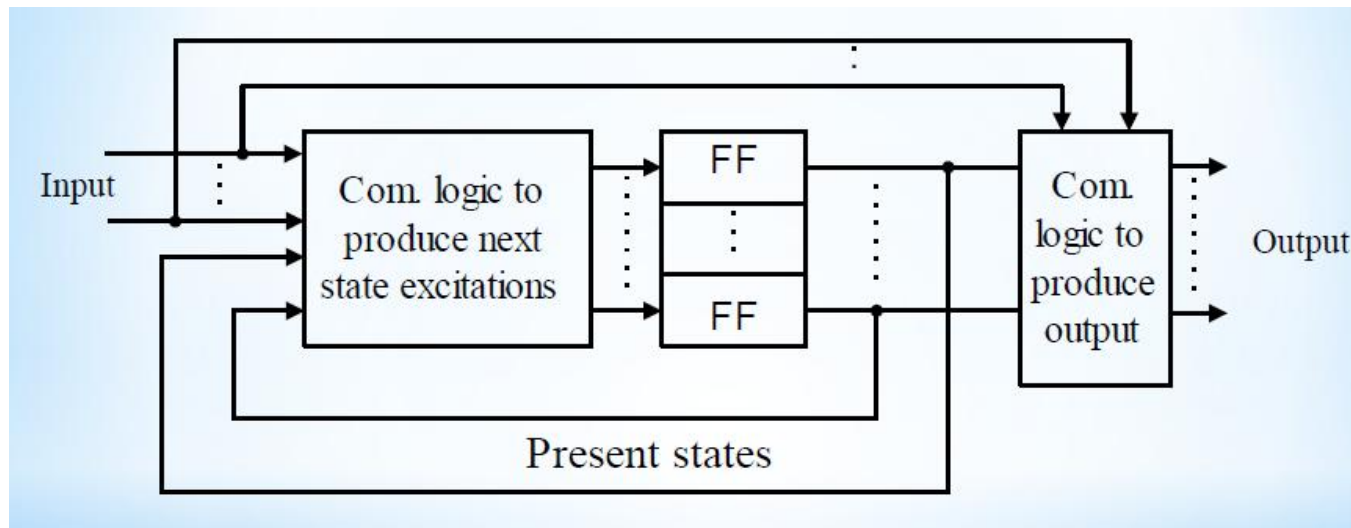
Instructor: Dr. Ray C.C. Cheung

Objectives

- Review Mealy machine and Moore Machine
- Implement a simple Mealy machine
- Learn a good coding style to describe a state machine

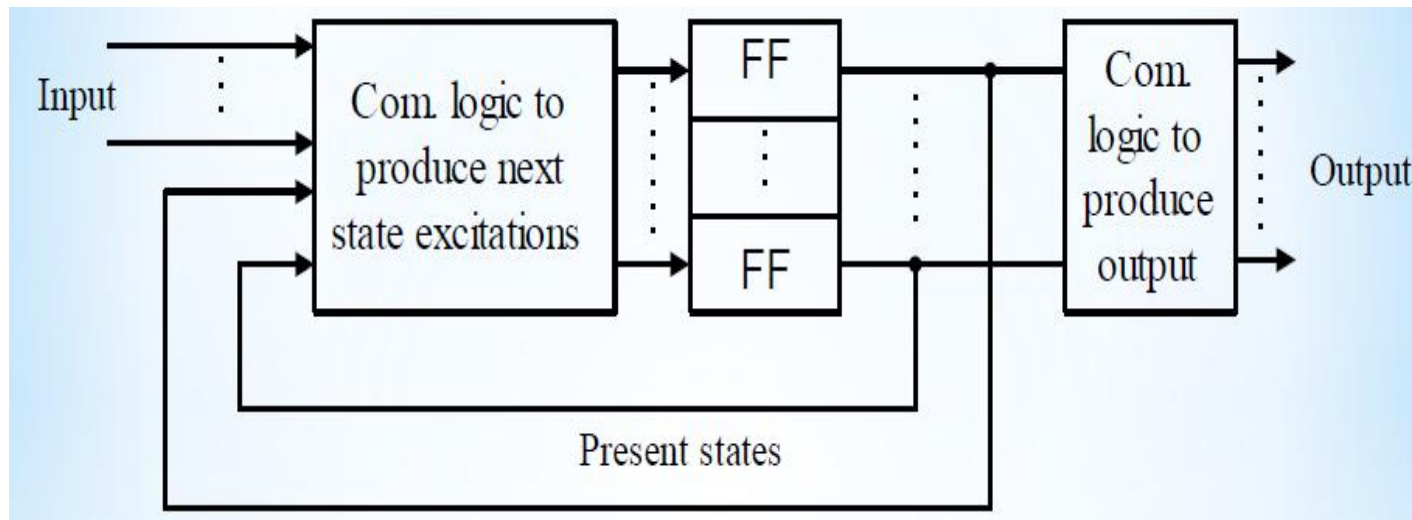
Mealy machine

- The output depends on both present state and input

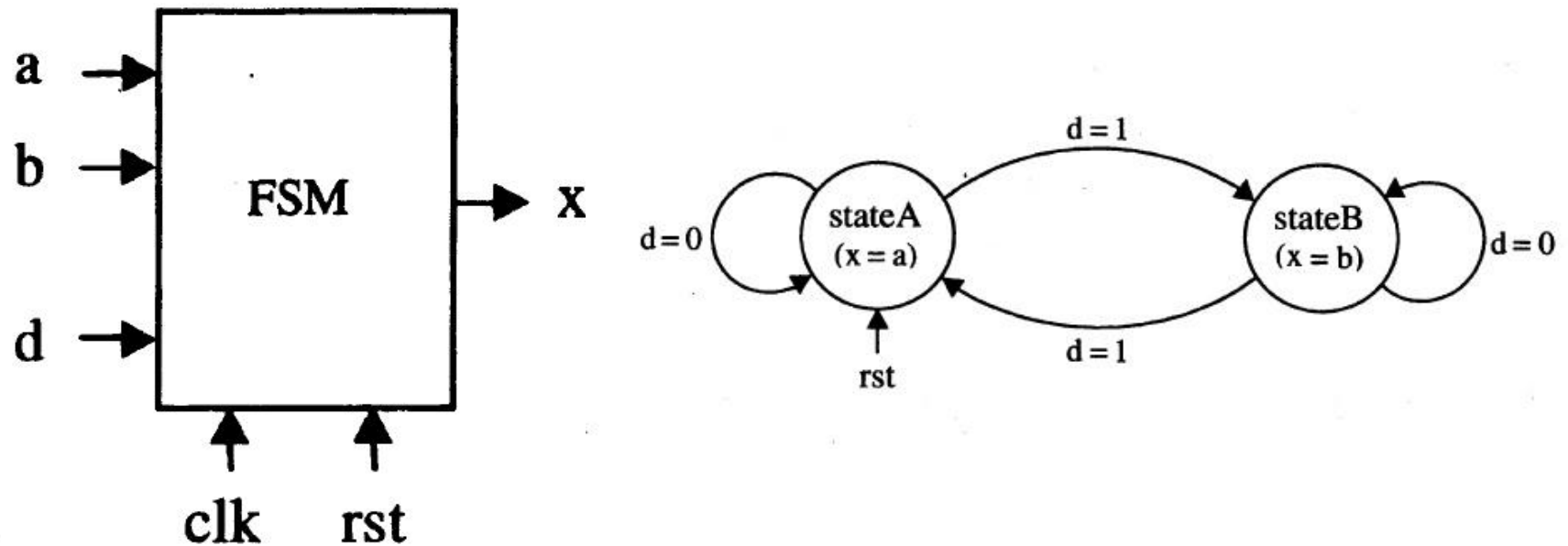


Moore machine

- The output depends on present state only.



Mealy machine - Example



nx_state / output		
pr_state	d=0	d=1
stateA	stateA / a	stateB / a
stateB	stateB / b	stateA / b

VHDL Step by step

Define states



Describe state transition



Describe transition condition (nx_state) and output



Simulation, Synthesis and Implementation

Define states

- Use TYPE to define an enumerated type (just like *enum* in C/C++). Now *state* is a type, the value can be *stateA* or *stateB*.

```
TYPE state IS (stateA, stateB);
```

```
SIGNAL pr_state: state;
```

```
SIGNAL nx_state: state;
```

- Actually, in VHDL, we can use TYPE for user-defined data type. Search for more details.

```
TYPE my_integer IS RANGE -32 TO 32;
```

```
TYPE my_logic IS ('0', '1', 'Z');
```

State transition

- At every rising edge of *clk*, the value of *nx_state* is assigned to *pr_state* (imaging there is a D-FF).
- Different ways to describe sequential logic:

```
process (rst, clk)
begin
    if (rst='1') then
        pr_state <= stateA;
    elsif (clk'event and clk='1') then
        pr_state <= nx_state;
    end if;
end process;
```

State transition

- Different ways to describe sequential logic:

```
process (rst, clk)
begin
    if (rst='1') then
        pr_state <= stateA;
    elsif (rising_edge(clk)) then
        pr_state <= nx_state;
    end if;
end process;
```

State transition

- Different ways to describe sequential logic:

```
process
begin
  wait on rst, clk;
  if (rst='1') then
    pr_state <= stateA;
  elsif (clk'event and clk='1') then
    pr_state <= nx_state;
  end if;
end process;
```

```
process
begin
  wait until rising_edge(clk);
  if (rst='1') then
    pr_state <= stateA;
  else
    pr_state <= nx_state;
  end if;
end process;
```

No sensitivity list if you use WAIT.

nx_state & output

- *nx_state* and output are determined by input and *pr_state*,
- Use combinational logic to describe.
- When you are describing a finite state machine, it is a good habit to divide sequential logic and combinational logic. Put them into different PROCESS block.

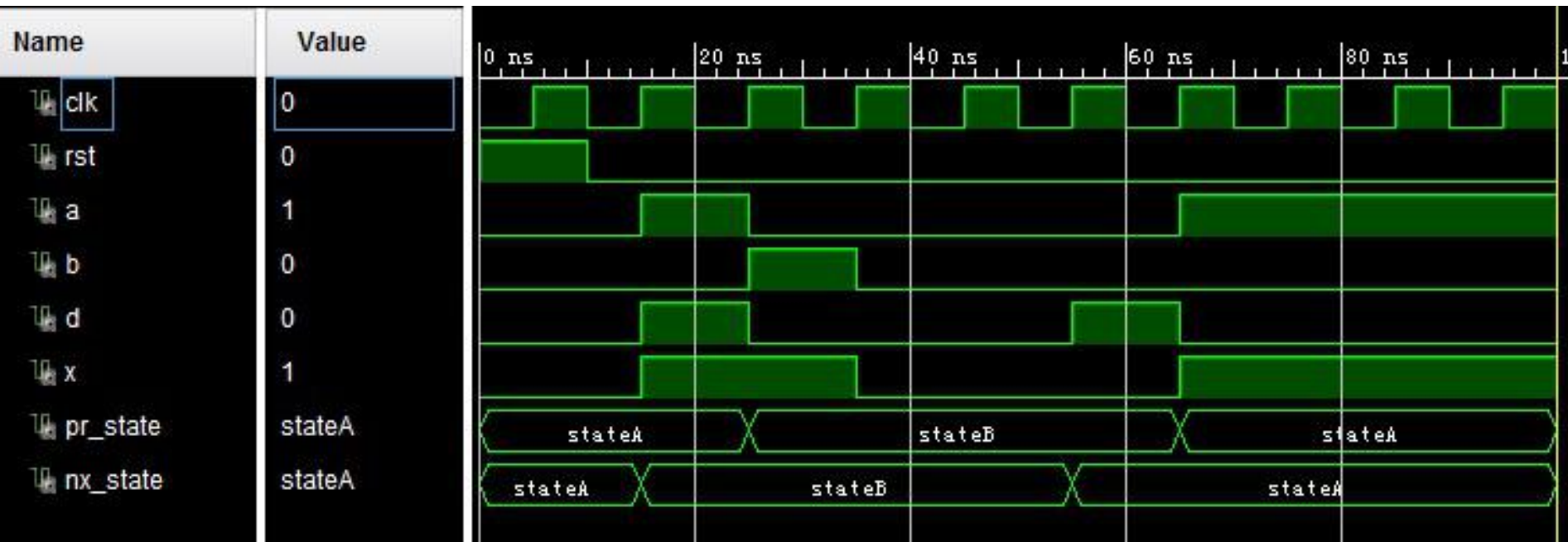
```
case pr_state is
  when stateA =>
    x <= a;
    if (d='1') then
      nx_state <= stateB;
    else
      nx_state <= stateA;
    end if;
  when stateB =>
    x <= b;
    if (d='1') then
      nx_state <= stateA;
    else
      nx_state <= stateB;
    end if;
end case;
```

Entire design

```
34 entity fsm is
35   port (
36     clk, rst, a, b, d: IN STD_LOGIC;
37     x: OUT STD_LOGIC
38   );
39 end fsm;
40
41 architecture Behavioral of fsm is
42   TYPE state IS (stateA, stateB);
43   SIGNAL pr_state: state;
44   SIGNAL nx_state: state;
45 begin
46   process (rst, clk)
47   begin
48     if (rst='1') then
49       pr_state <= stateA;
50     elsif (clk'event and clk='1') then
51       pr_state <= nx_state;
52     end if;
53   end process;
```

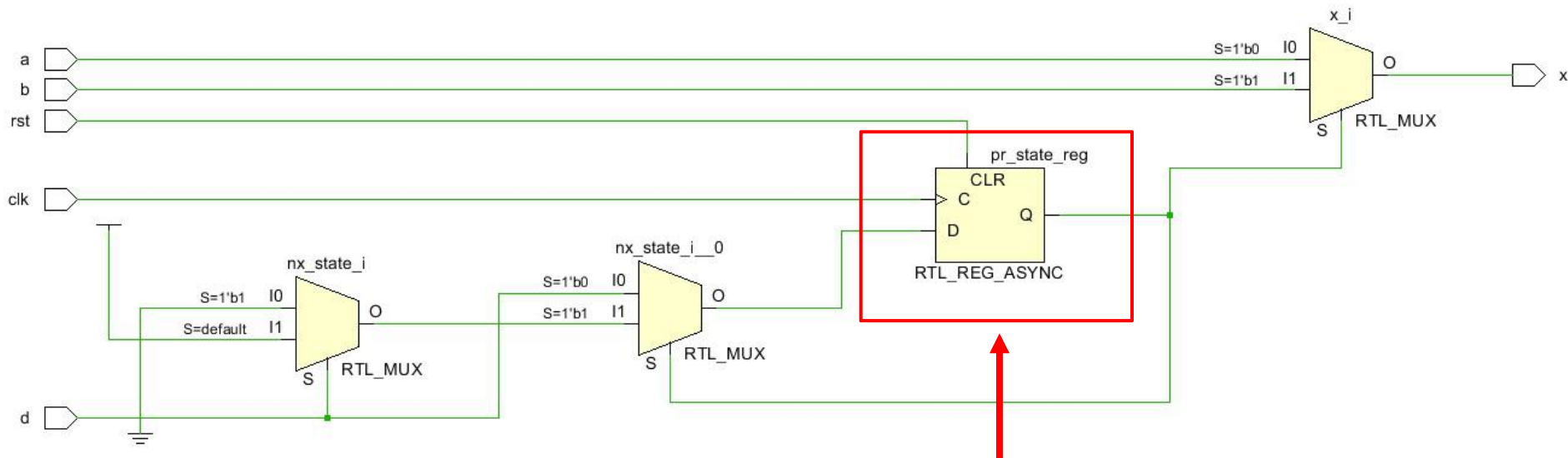
```
55   process (a, b, d, pr_state)
56   begin
57     case pr_state is
58       when stateA =>
59         x <= a;
60         if (d='1') then
61           nx_state <= stateB;
62         else
63           nx_state <= stateA;
64         end if;
65       when stateB =>
66         x <= b;
67         if (d='1') then
68           nx_state <= stateA;
69         else
70           nx_state <= stateB;
71         end if;
72     end case;
73   end process;
74 end Behavioral;
```

Simulation results



- *pr_state* is one clock later than *nx_state*.
- *nx_state* is determined by input *d*.
- Output *x* is determined by both *pr_state* and input *a*, *b*.

RTL analysis



One register (D-FF) for state transition

Reference

- Design of finite state machine
 - <https://www.youtube.com/watch?v=P56oe1ZkDF0&t=4s>
- Xilinx finite state machine lab workbook
 - <https://www.xilinx.com/support/documentation/university/Vivado-Teaching/HDL-Design/2015x/VHDL/docs-pdf/lab10.pdf>

Tutorial - Remarks

- Please raise your hand if you have questions!
 - 9 course tutors + course Instructors
 - If you finish the tutorial quickly, please try different settings in the Vivado tool.
- You can also refer to the course Canvas homepage.
- Get familiar with sequential logic design.
- Learn a good coding style for state machine description.