ECE 448
Lecture 10

Finite State Machines

State Diagrams, Algorithmic State Machine (ASM) Charts, and VHDL Code

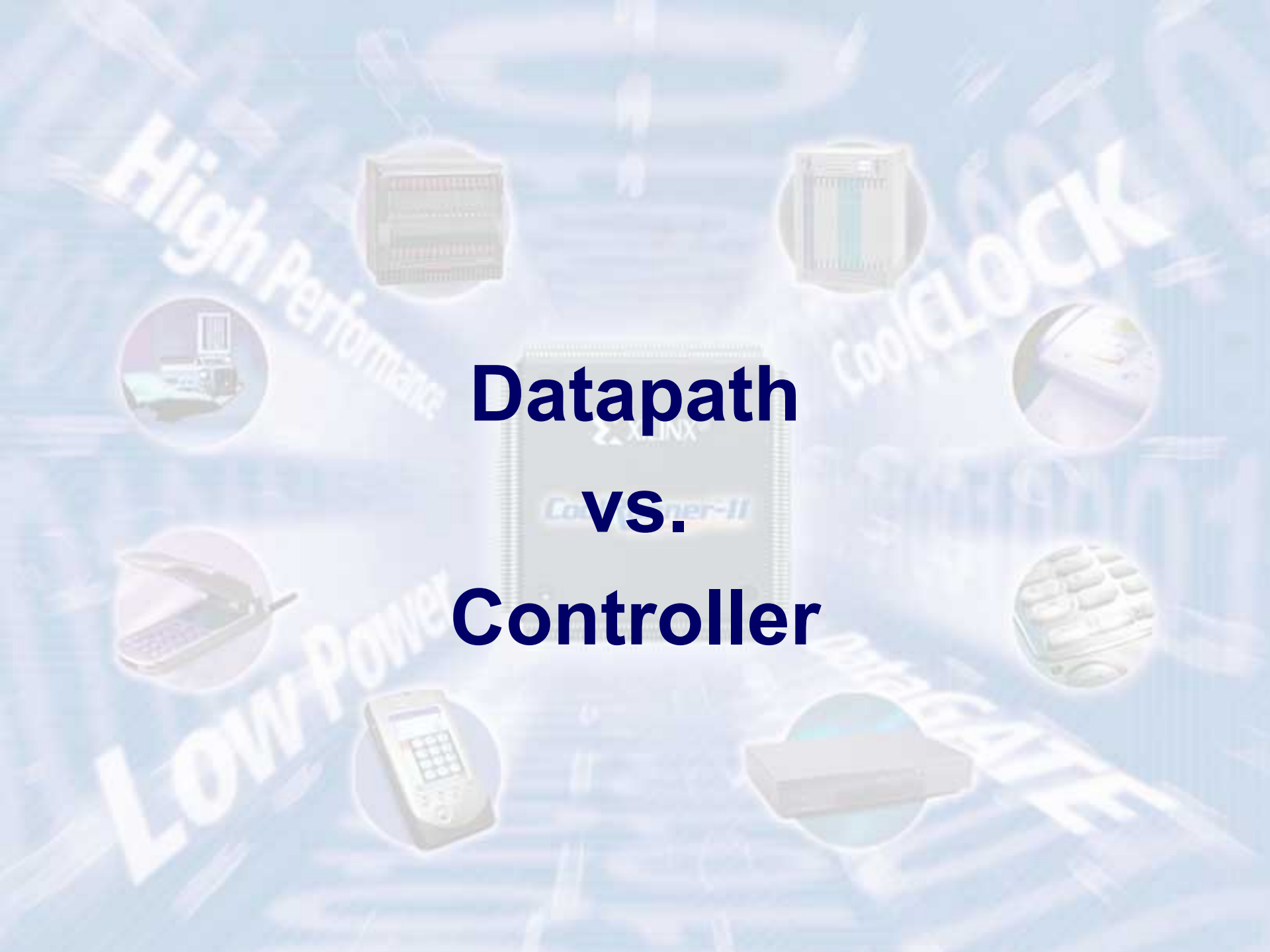
Required reading

- P. Chu, *FPGA Prototyping by VHDL Examples*

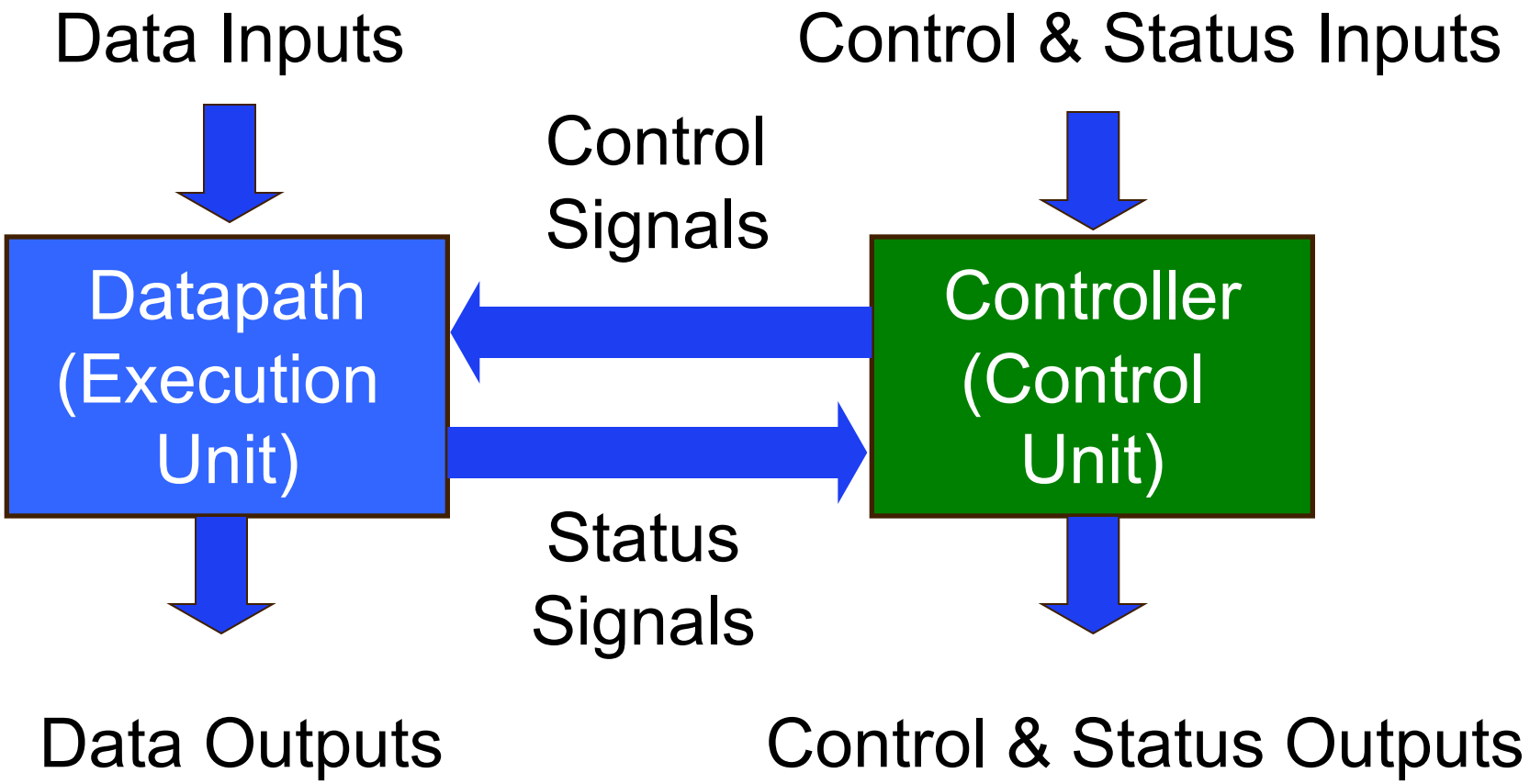
  Chapter 5, FSM

Recommended reading

- S. Brown and Z. Vranesic,
  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


Datapath
vs.
Controller

Structure of a Typical Digital System



Controller (Control Unit)

- Controls data movement in the Datapath by switching multiplexers and enabling or disabling resources
  - enable signals for registers
  - load signals for counters and shift registers
  - select signals for muxes
- Provides signals to activate various processing tasks in the Datapath
- Determines the sequence of operations performed by the Datapath
- Follows Some ‘Program’ or Schedule

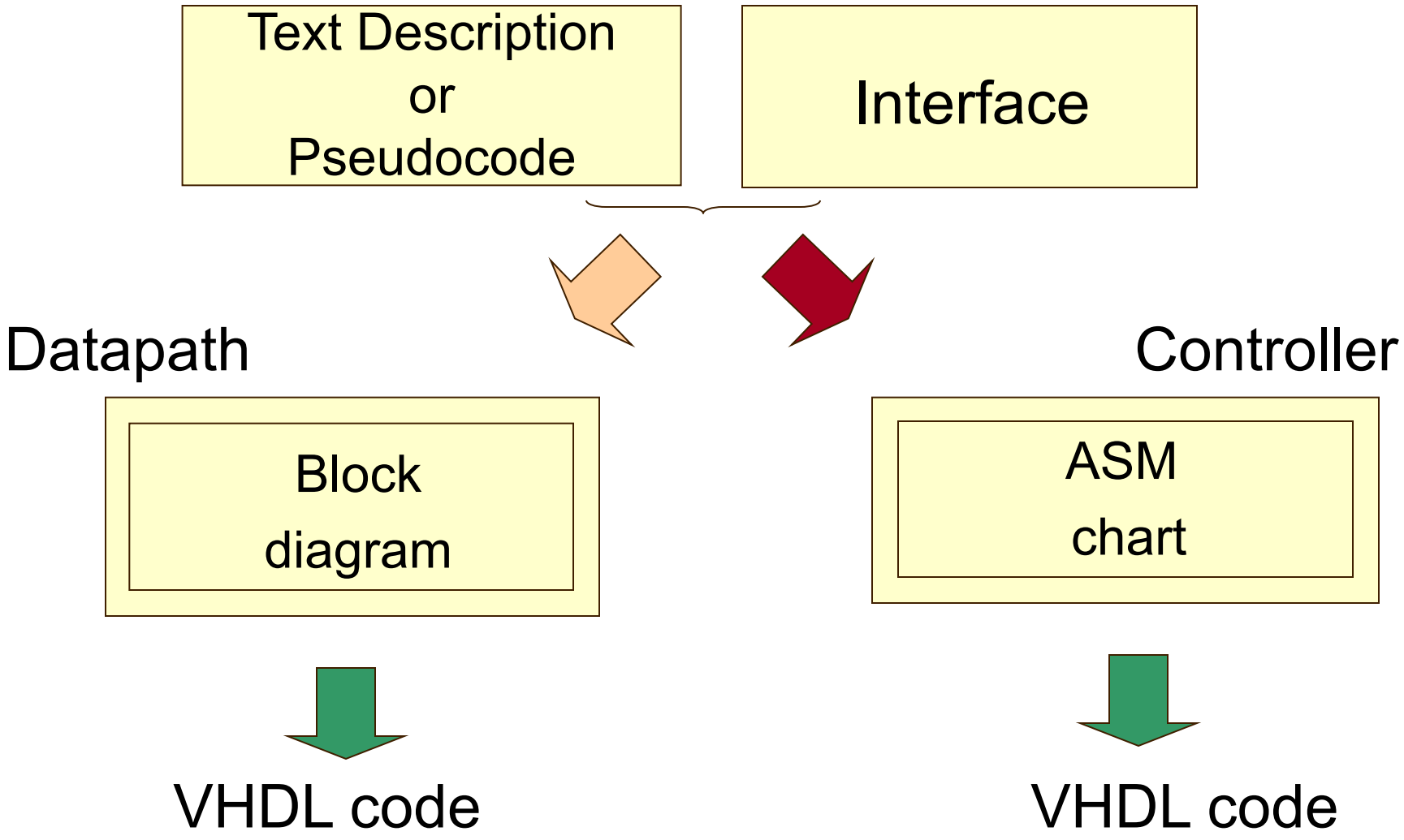
Programmable vs. Non-Programmable Controller

- Controller can be programmable or non-programmable
- Programmable
  - Has a program counter which points to the next instruction
  - Instructions are held in a RAM or ROM
  - Microprocessor is an example of a programmable controller
- Non-Programmable
  - Once designed, implements the same functionality
  - Another term is a “hardwired state machine,” or “hardwired FSM,” or “hardwired instructions”
  - **In this course we will be focusing on non-programmable controllers.**

Finite State Machines

- Controllers can be described as Finite State Machines (FSMs)
- Finite State Machines can be represented using
  - **State Diagrams and State Tables** - suitable for simple controllers with a relatively few inputs and outputs
  - **Algorithmic State Machine (ASM) Charts** - suitable for complex controllers with a large number of inputs and outputs
- All of these descriptions can be easily translated to the corresponding synthesizable VHDL code

Hardware Design with RTL VHDL



Steps of the Design Process Introduced in Class Today

1. Text description
2. Interface
3. Pseudocode (optional)
4. Block diagram of the Datapath
5. Interface divided into the Datapath and Controller
6. **State diagram or ASM chart of the Controller**
7. **RTL VHDL code of the** Datapath, **Controller**, and Top-level Unit
8. Testbench for the Datapath, Controller, and Top-Level Unit
9. Functional simulation and debugging
10. Synthesis and post-synthesis simulation
11. Implementation and timing simulation
12. Experimental testing using FPGA board

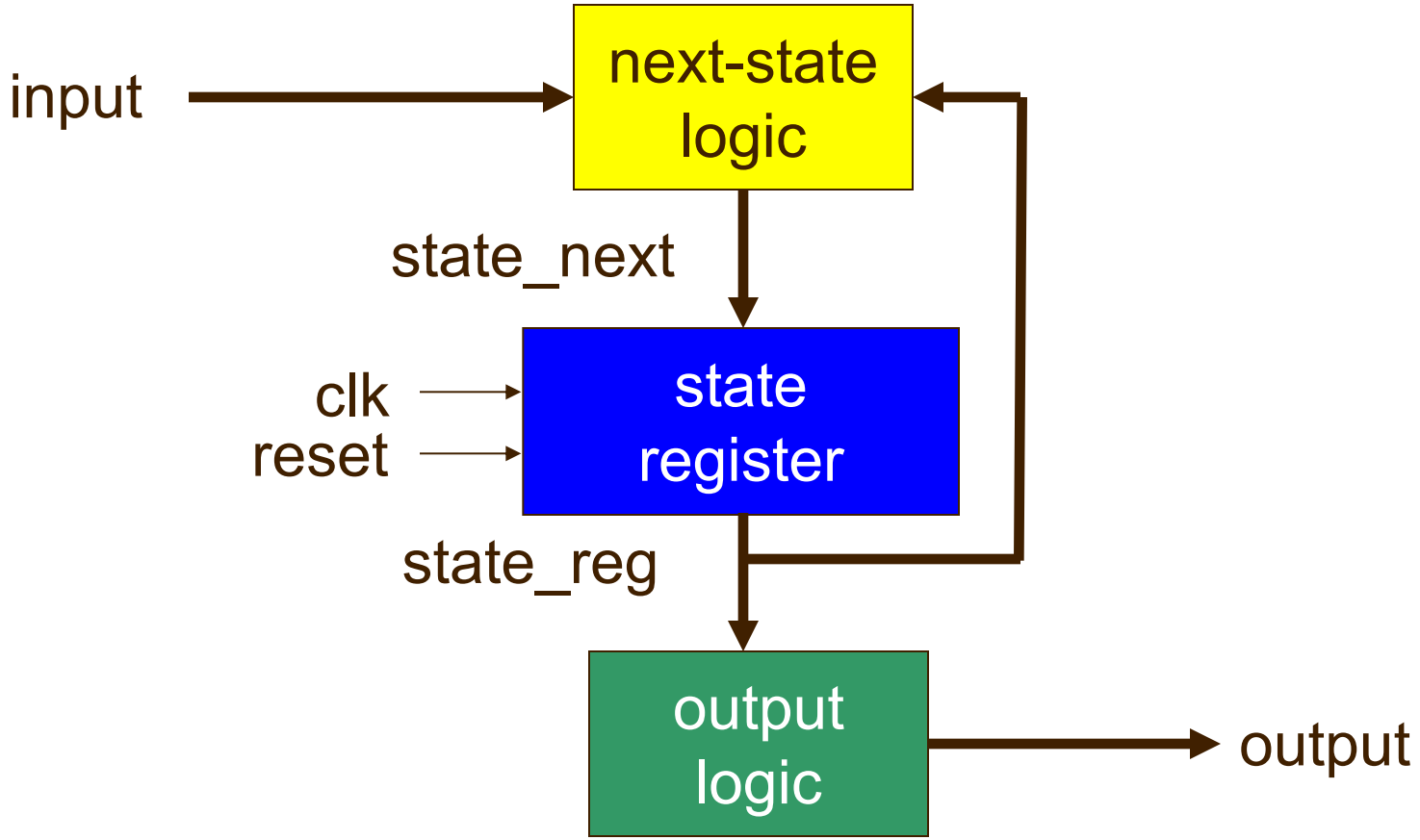
Finite State Machines Refresher

Finite State Machines (FSMs)

- An FSM is used to model a system that transits among a finite number of internal states. The transitions depend on the current state and external input.
- The main application of an FSM is to act as the controller of a medium to large digital system
- Design of FSMs involves
  - Defining states
  - Defining the next-state and output functions
  - Optimization / minimization
- Manual optimization/minimization is practical for small FSMs only

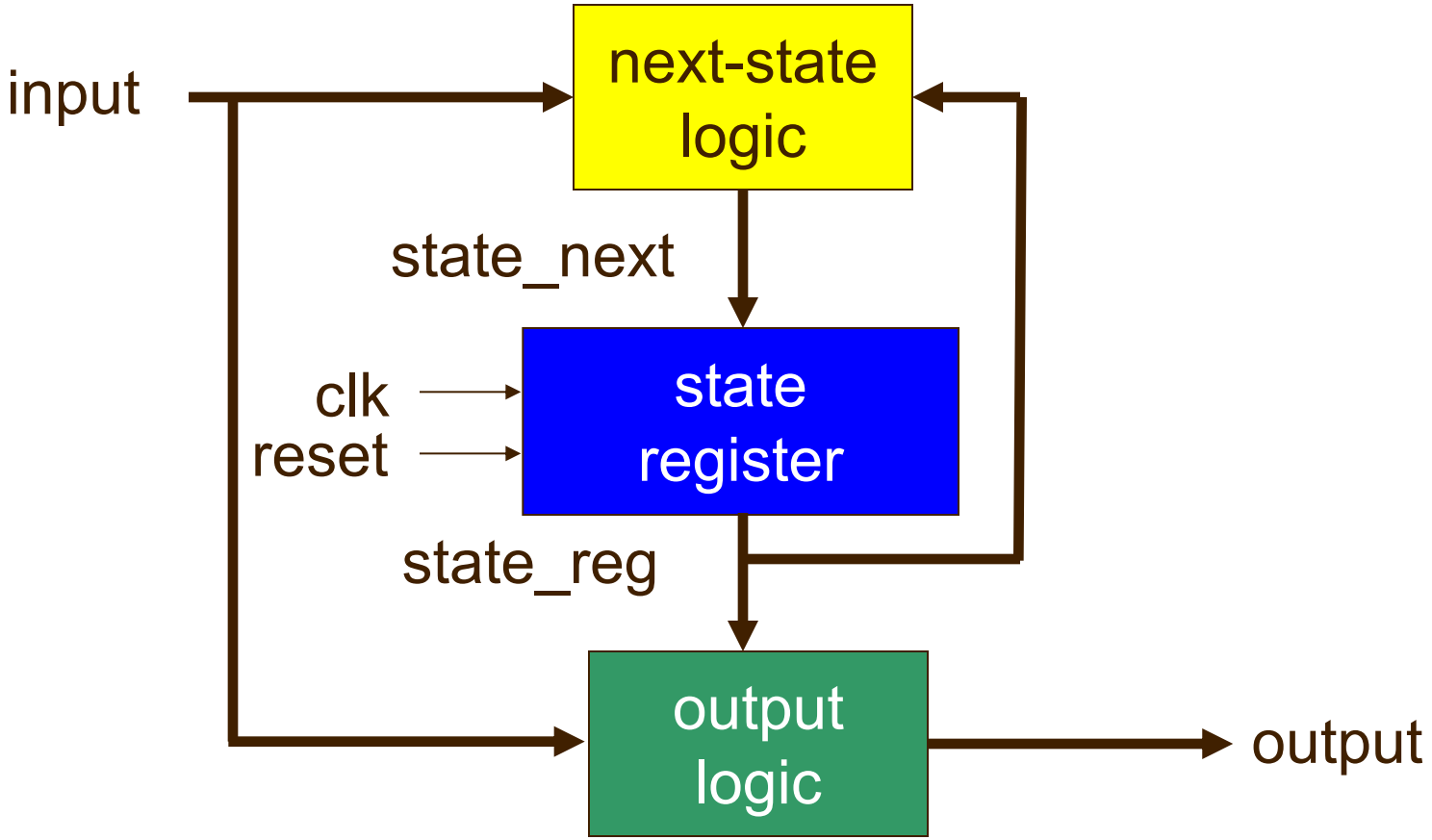
Moore FSM

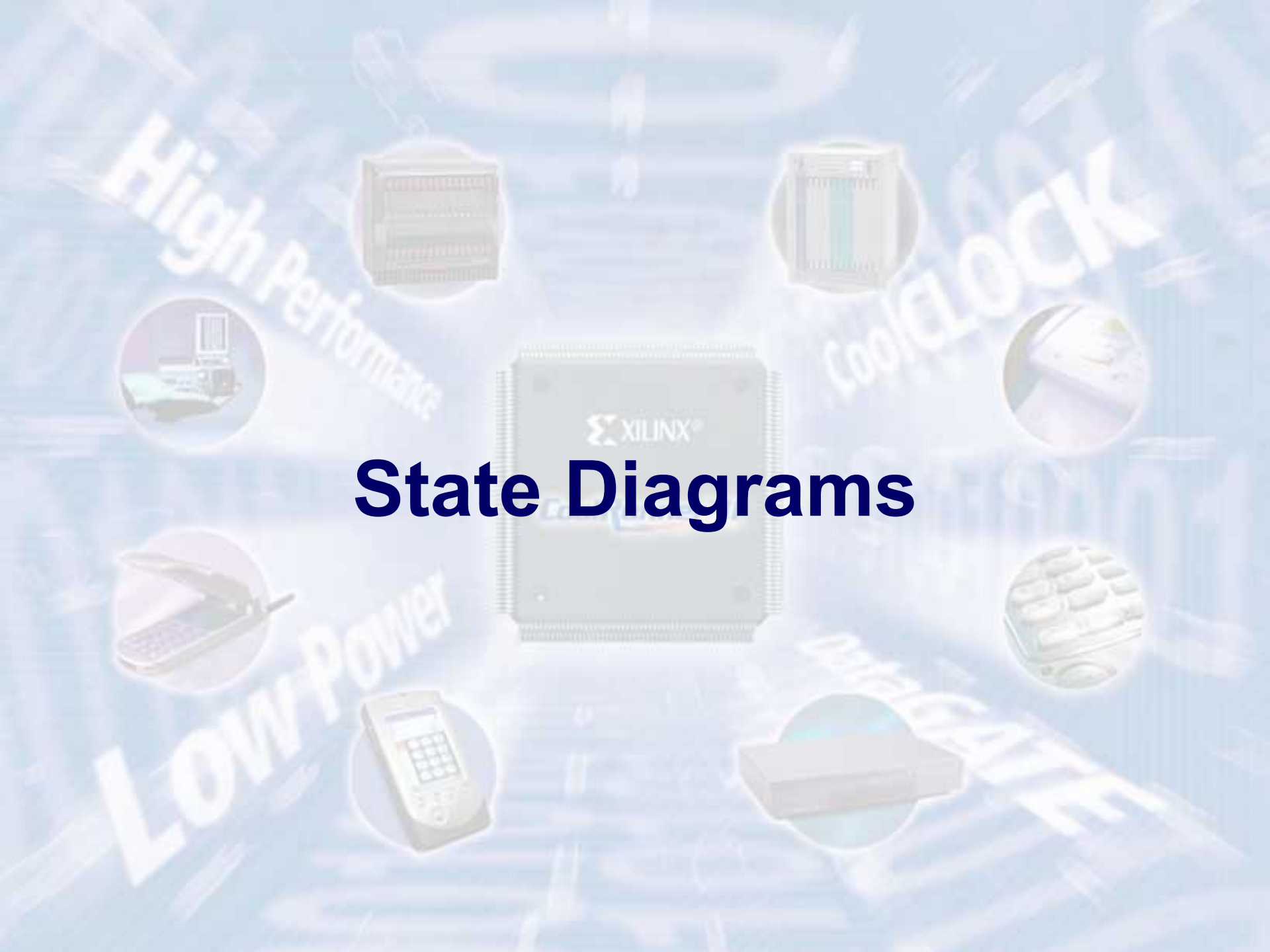
- output is a function of the state only



Mealy FSM

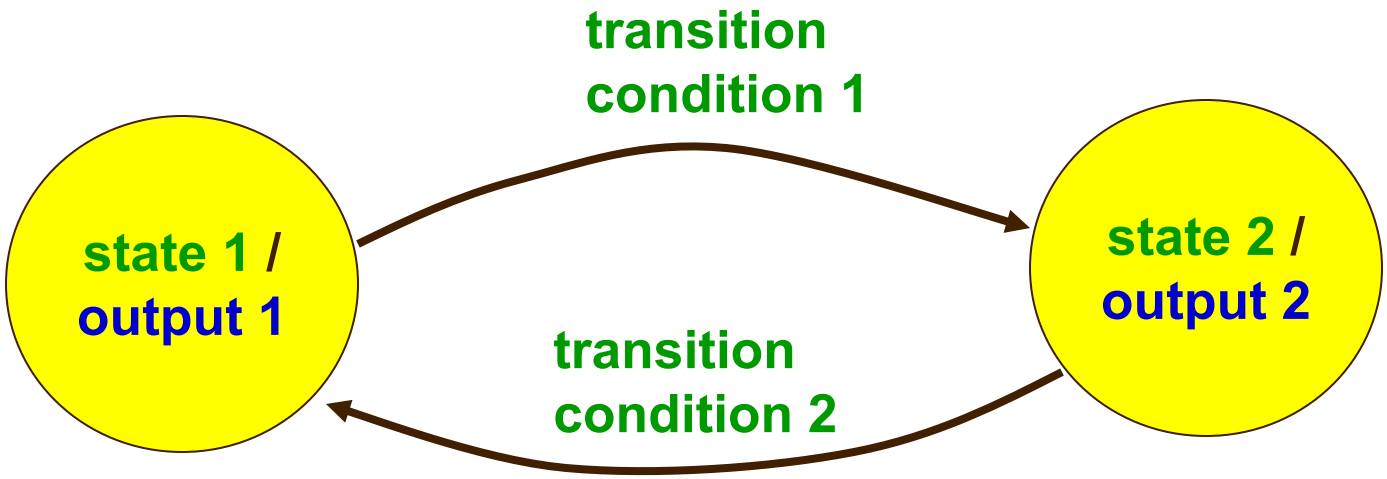
- output is a function of the state and input signals



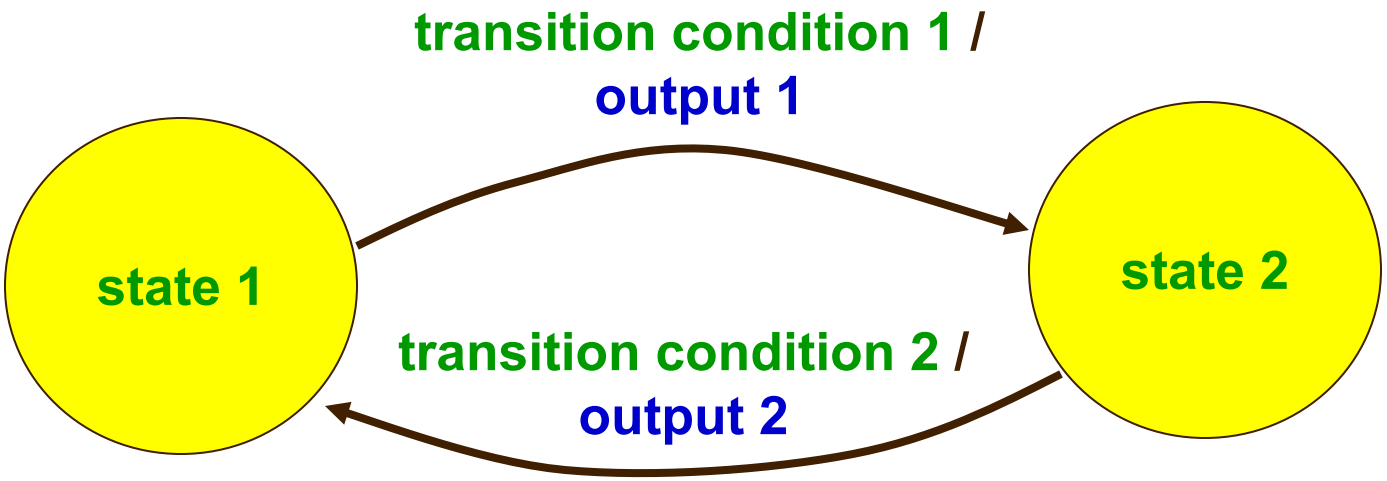


State Diagrams

Moore Machine

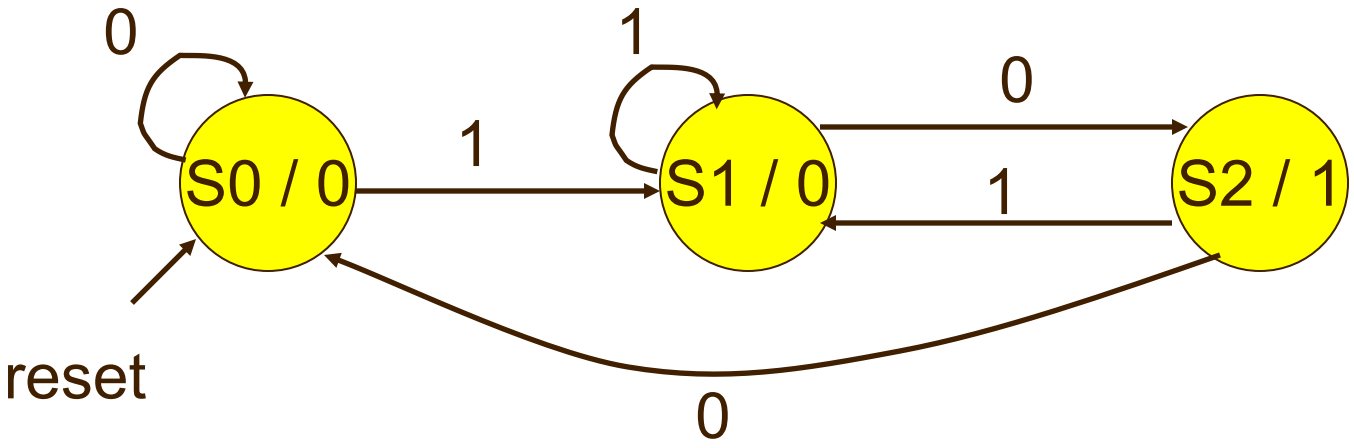


Mealy Machine



Moore FSM - Example 1

- Moore FSM that Recognizes Sequence “10”



Meaning of states:

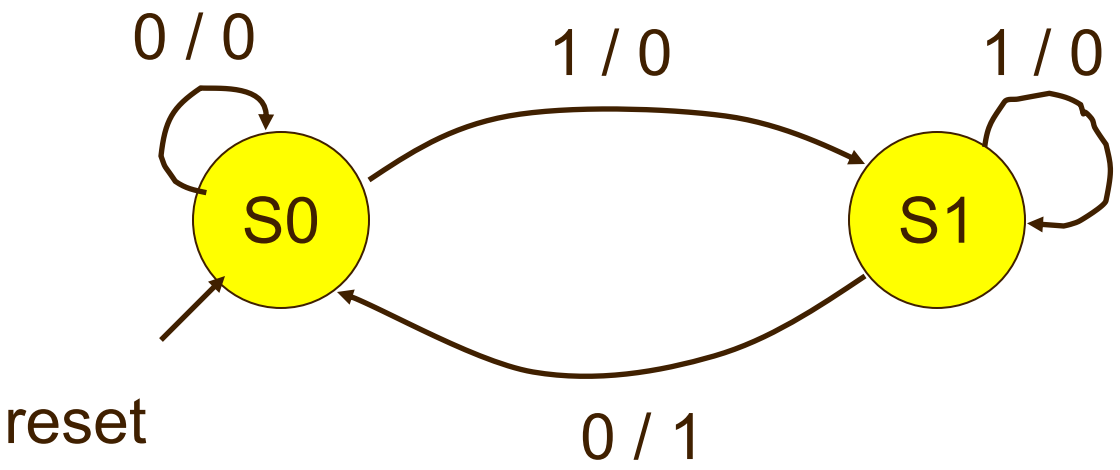
S0: No elements of the sequence observed

S1: “1” observed

S2: “10” observed

Mealy FSM - Example 1

- Mealy FSM that Recognizes Sequence "10"



Meaning of states:

S0: No elements of the sequence observed

S1: "1" observed

Algorithmic State Machine (ASM) Charts

Algorithmic State Machine

Algorithmic State Machine –

representation of a Finite State Machine

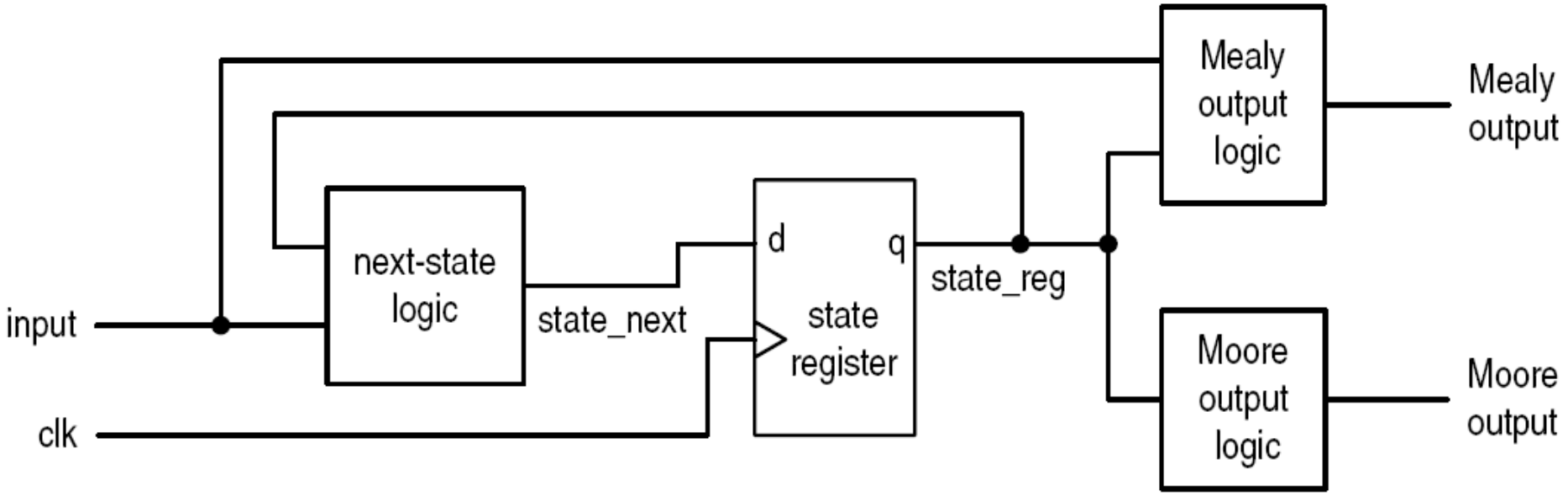
suitable for FSMs with a larger number of inputs and outputs compared to FSMs expressed using state diagrams and state tables.

ASM Chart

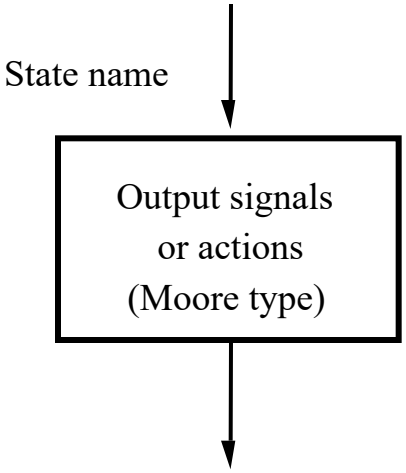
- Flowchart-like diagram
- Provides the same info as a state diagram
- More descriptive, better for complex digital systems

ASM describing generalized FSM

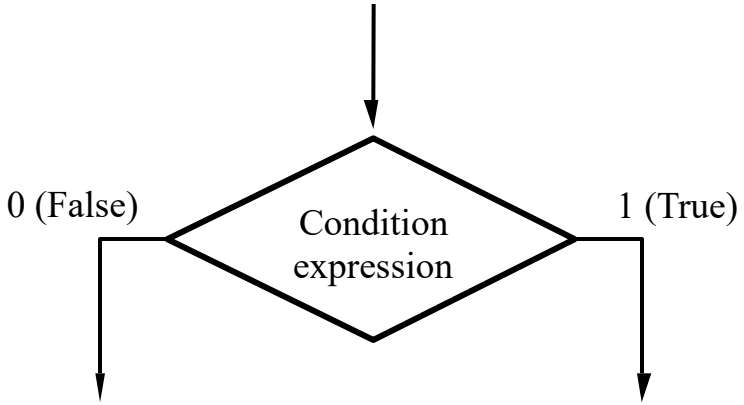
- Algorithmic state machines can model both Mealy and Moore Finite State Machines
- They can also model generalized machines that are of the mixed type



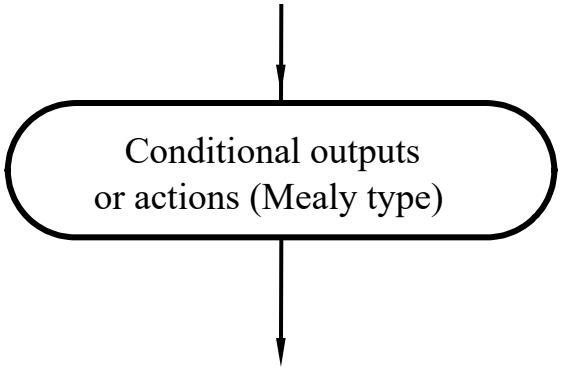
Elements used in ASM charts (1)



(a) State box



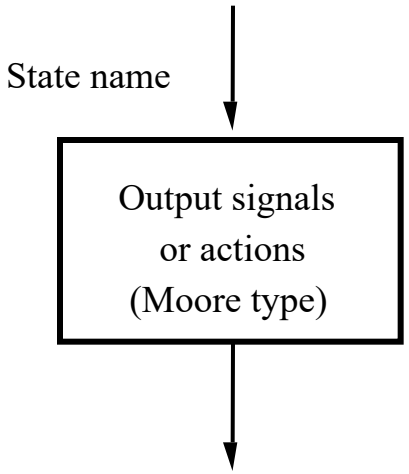
(b) Decision box



(c) Conditional output box

State Box

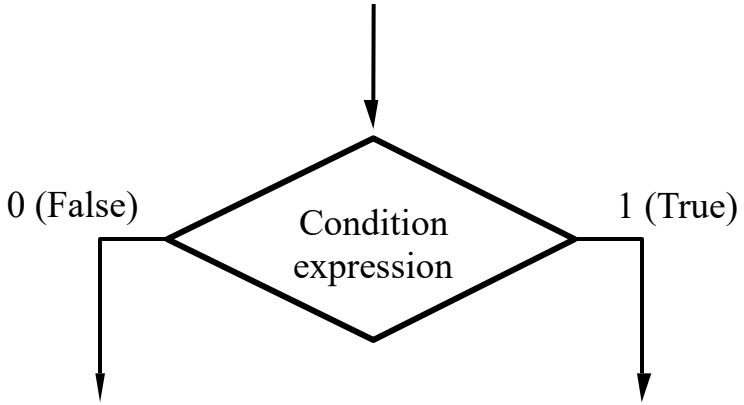
- **State box** – represents a state.
- Equivalent to a node in a state diagram or a row in a state table.
- Contains register transfer actions or output signals
- **Moore-type outputs are listed inside of the box.**
- It is customary to write only the name of the signal that has to be asserted in the given state, e.g., z instead of z<=1.
- Also, it might be useful to write an action to be taken, e.g., count <= count + 1, and only later translate it to asserting a control signal that causes a given action to take place (e.g., enable signal of a counter).



Decision Box

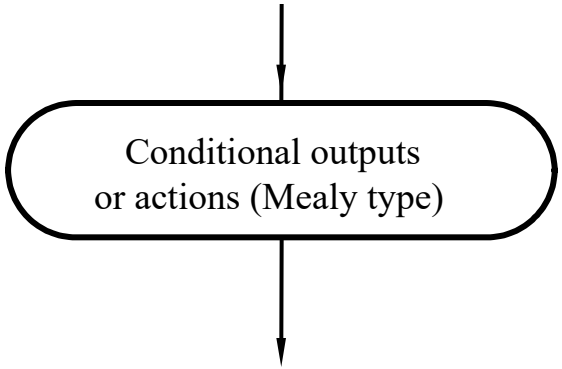
- **Decision box** – indicates that a given condition is to be tested and the exit path is to be chosen accordingly.

  The condition expression may include one or more inputs to the FSM.



Conditional Output Box

- **Conditional output box**
- Denotes output signals that are of the Mealy type.
- The condition that determines whether such outputs are generated is specified in the preceding decision box.



Simple Example

- BTND – reset controller
- BTNC – start counting from 0 to k-1

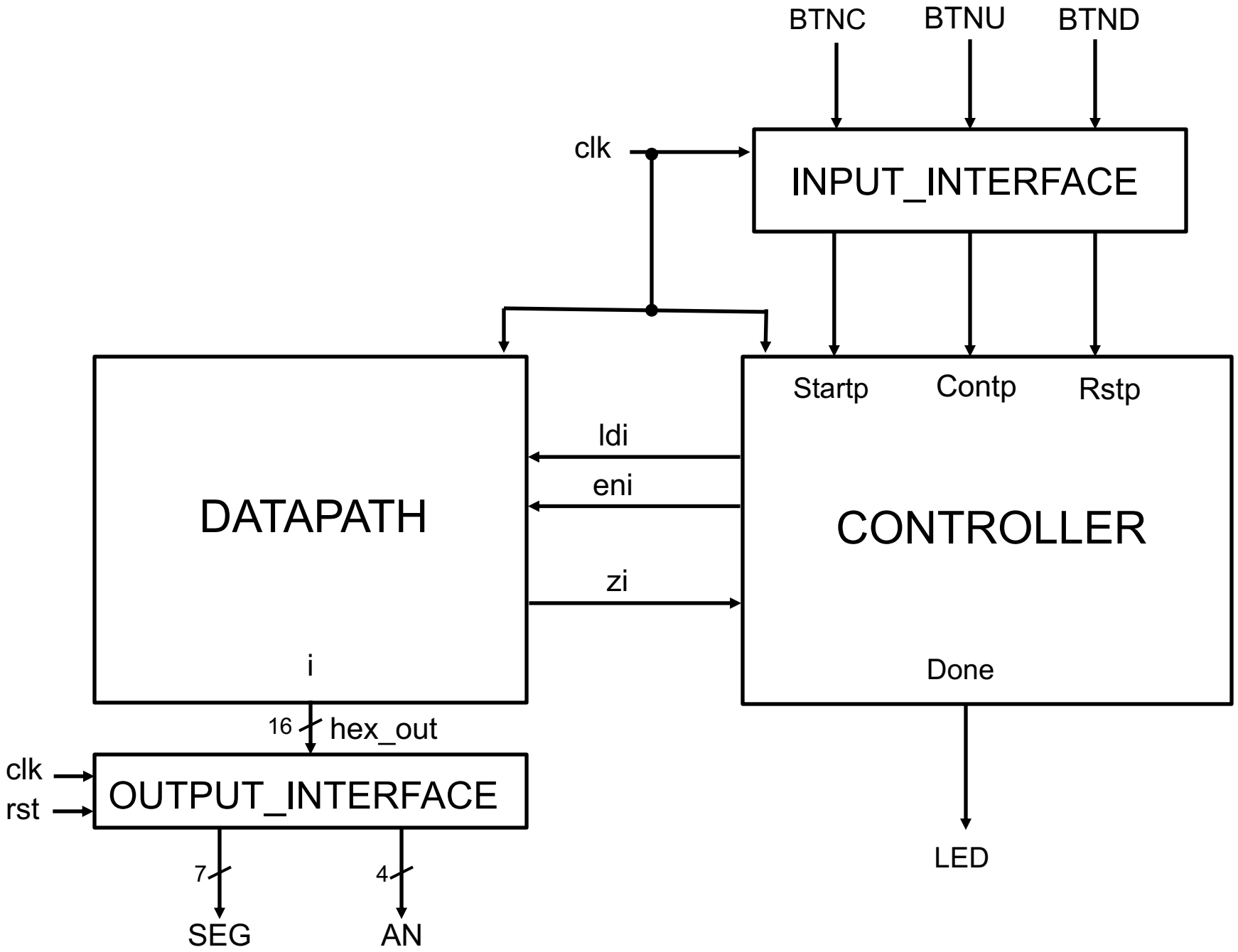
After the Counter reaches k-1, Done=1, LED LD0 on

- BTNU – resume the operation

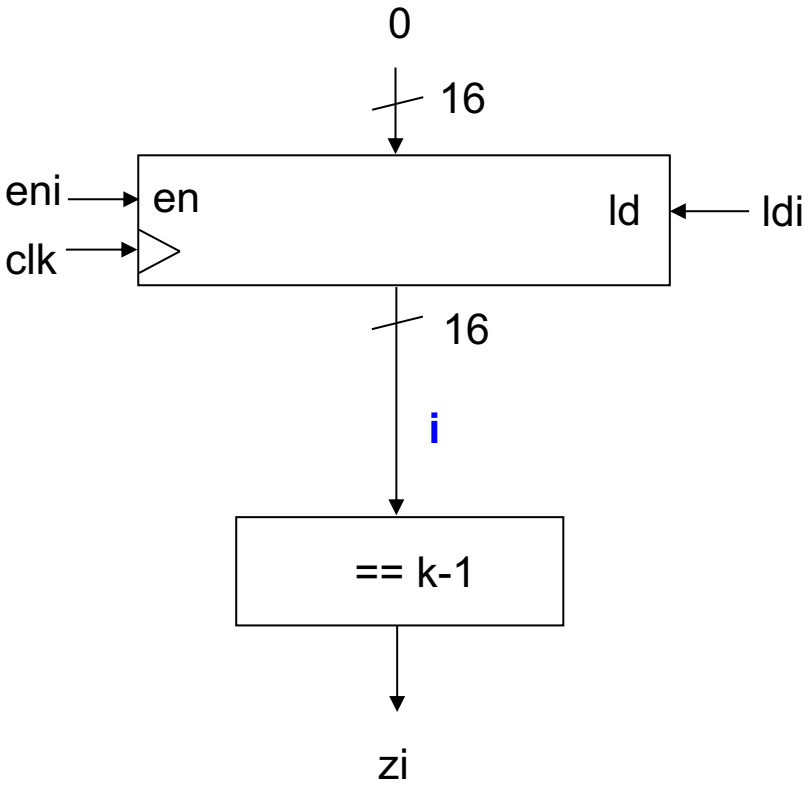
Done=0, LED LD0 off

The current value of the counter should be displayed in the hexadecimal notation using four seven-segment displays available on the board.

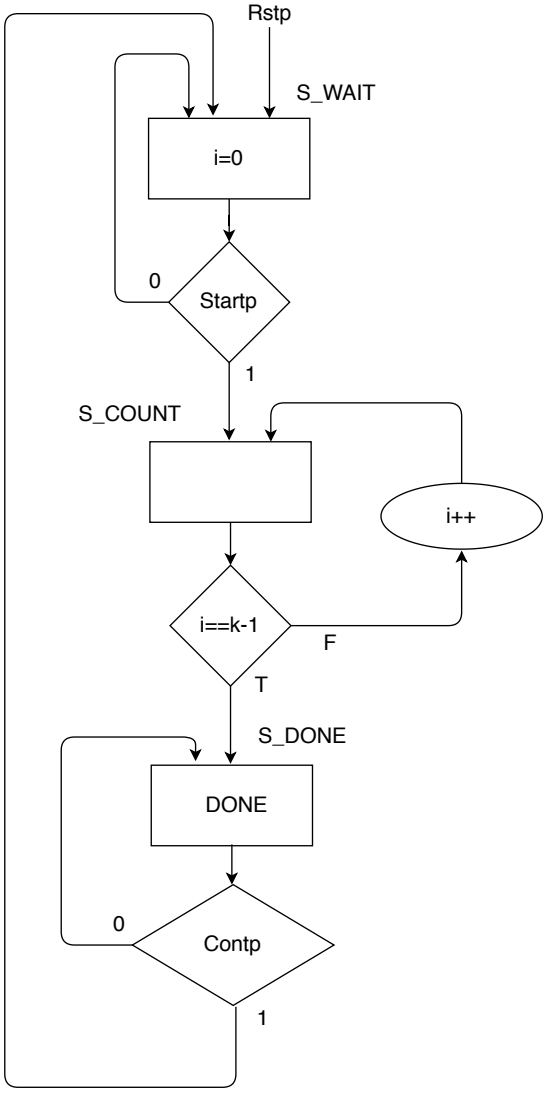
After reset: Counter = 0x0000

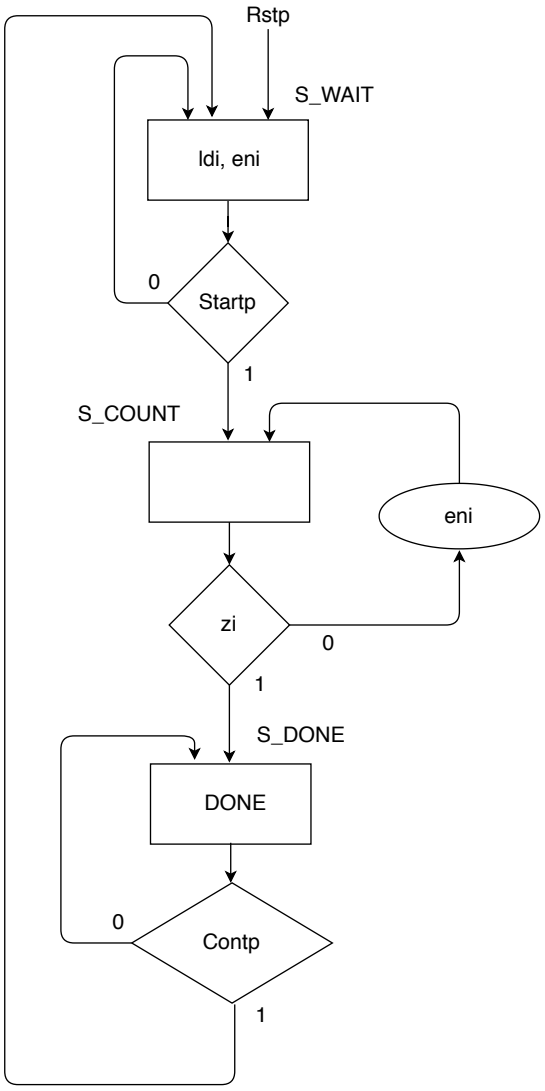

BTNC
BTNU
BTND
clk
INPUT_INTERFACE
Startp
Contp
Rstp
ldi
eni
zi
DATAPATH
CONTROLLER
i
Done
16
hex_out
clk
rst
OUTPUT_INTERFACE
LED
7
4
SEG
AN

Simple Example: Datapath



Simple Example: ASM Charts





a) with actions and condition expressions

b) with control outputs and status inputs

ASM Block

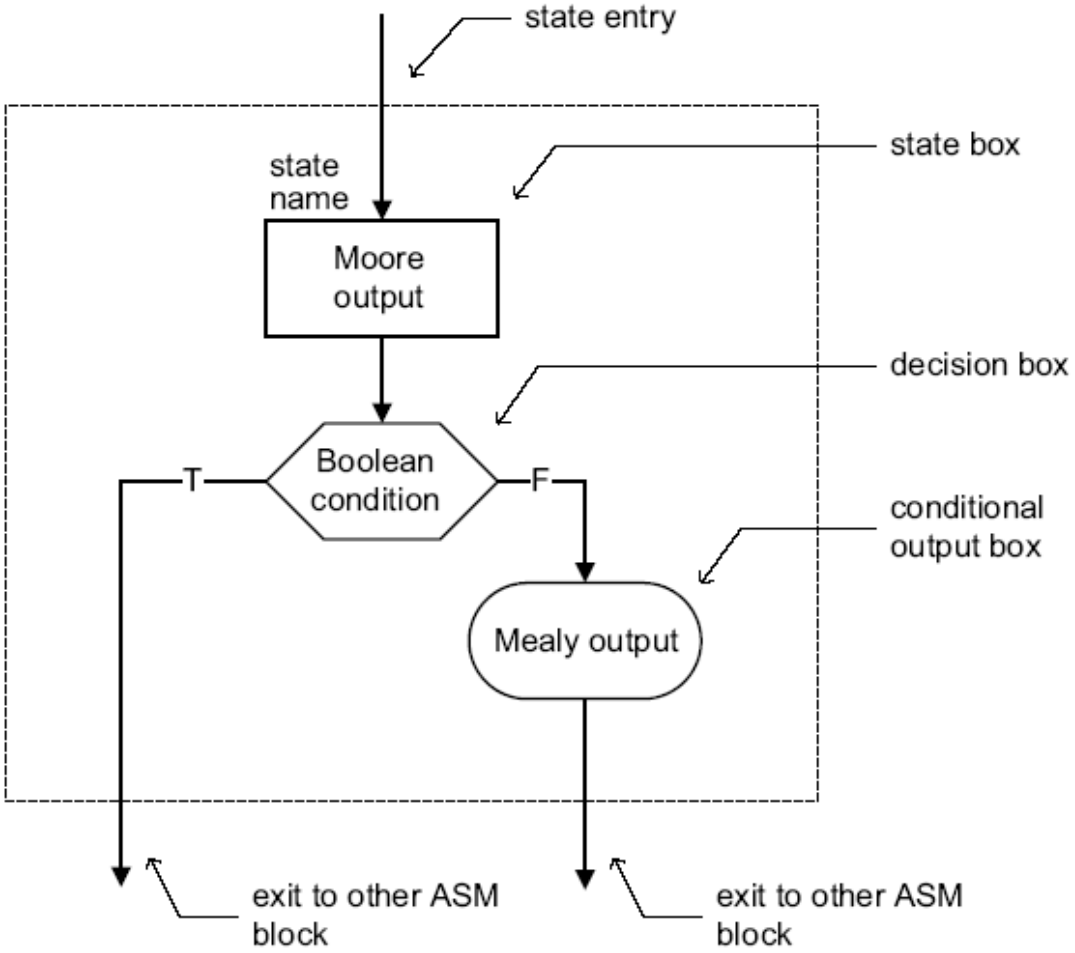


Figure 10.4 ASM block.

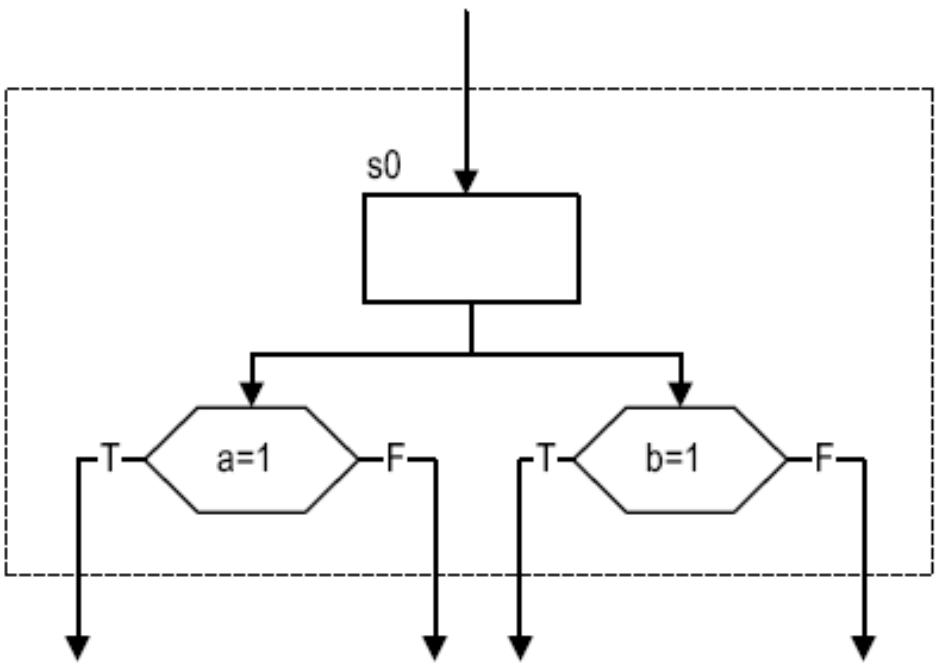
- One state box
- One or more (optional) decision boxes: with T (1) or F (0) exit paths
- One or more (optional) conditional output boxes: for Mealy outputs

ASM Chart Rules

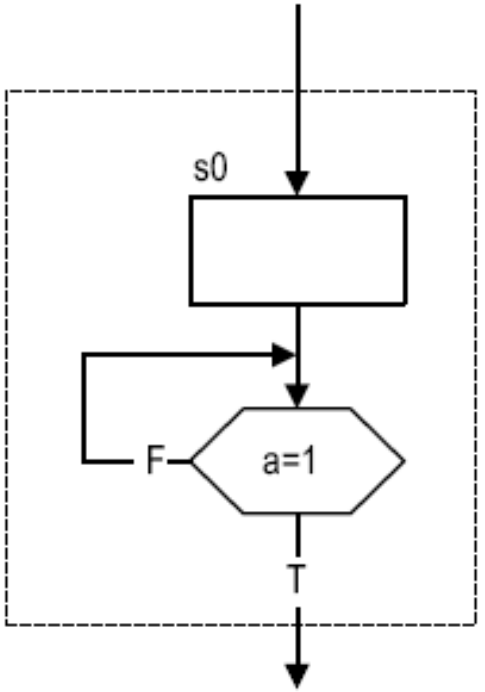
- Difference between a regular flowchart and an ASM chart:
  - Transition governed by clock
  - Transition occurs between ASM blocks
- Basic rules:
  - For a given input combination, there is one unique exit path from the current ASM block
  - Any closed loop in an ASM chart must include a state box

Based on RTL Hardware Design by P. Chu

Incorrect ASM Charts



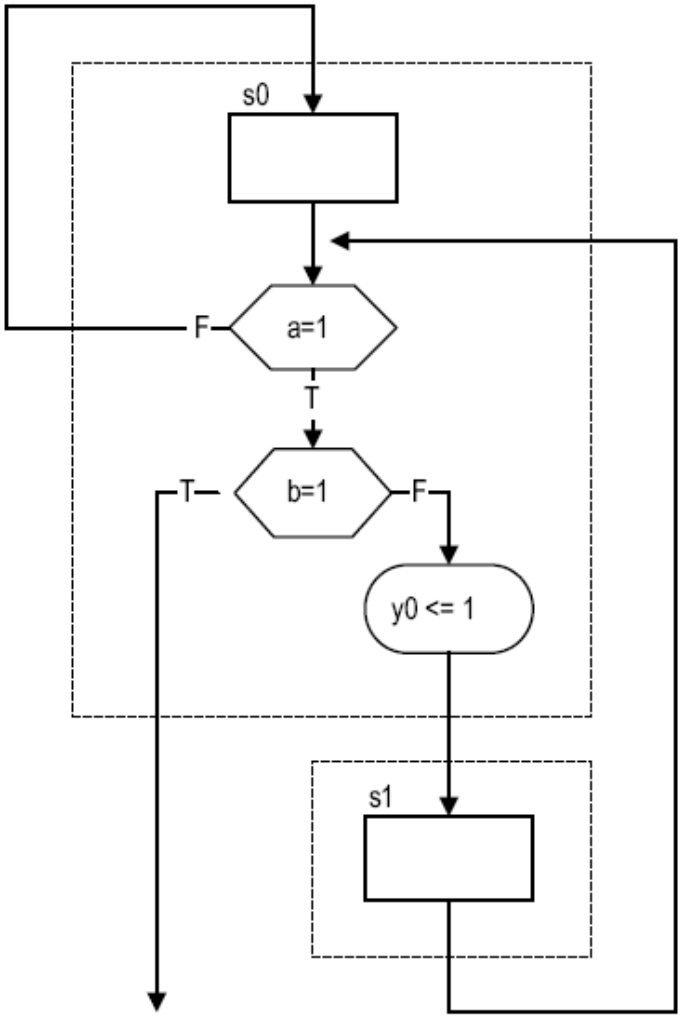
(a)



(b)

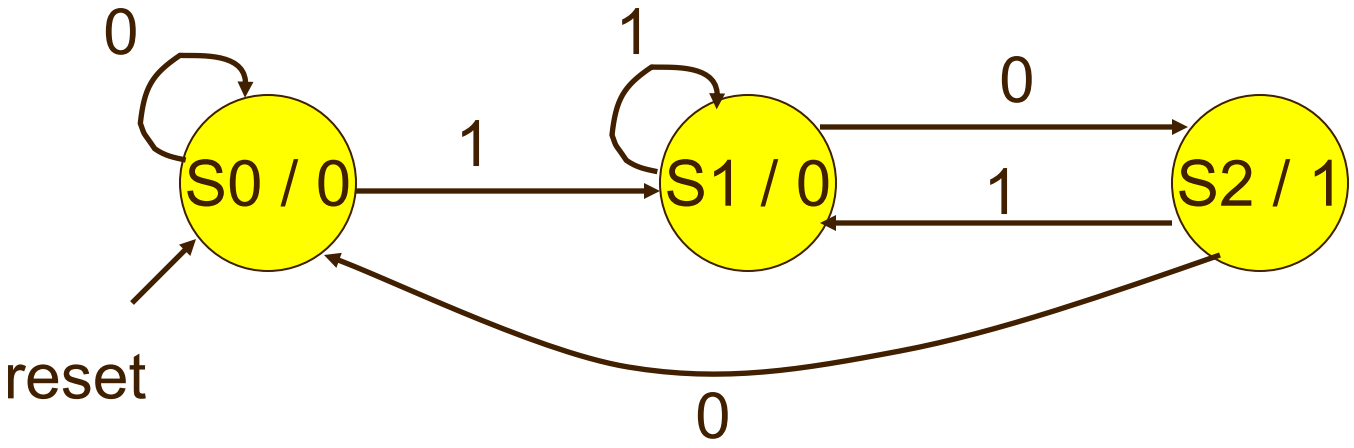
Based on RTL Hardware Design by P. Chu

Correct ASM Chart



State Diagram of Moore FSM

- Moore FSM that Recognizes Sequence “10”



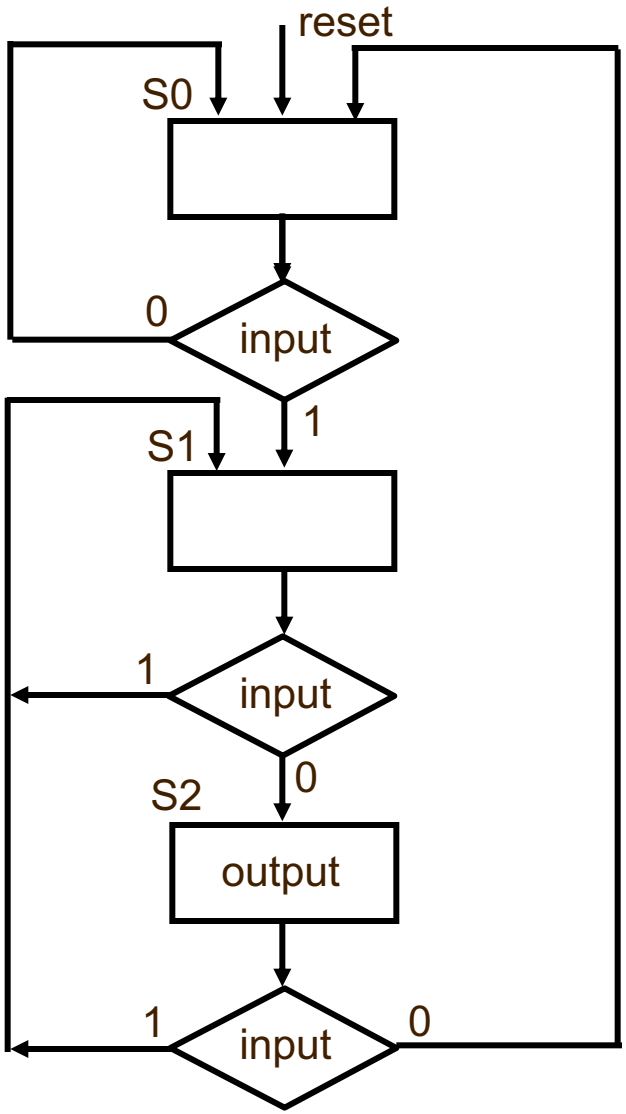
Meaning of states:

S0: No elements of the sequence observed

S1: “1” observed

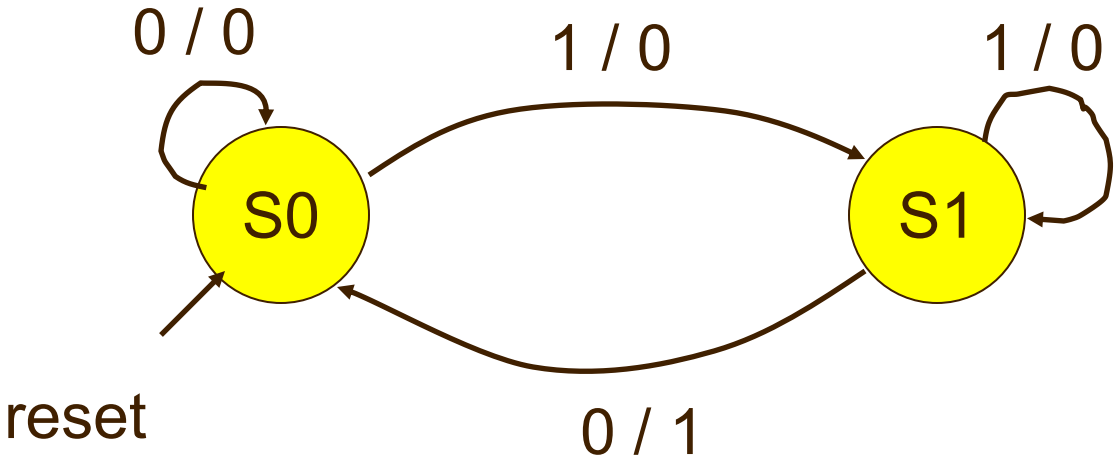
S2: “10” observed

ASM Chart of Moore FSM



State Diagram of Mealy FSM

- Mealy FSM that Recognizes Sequence "10"

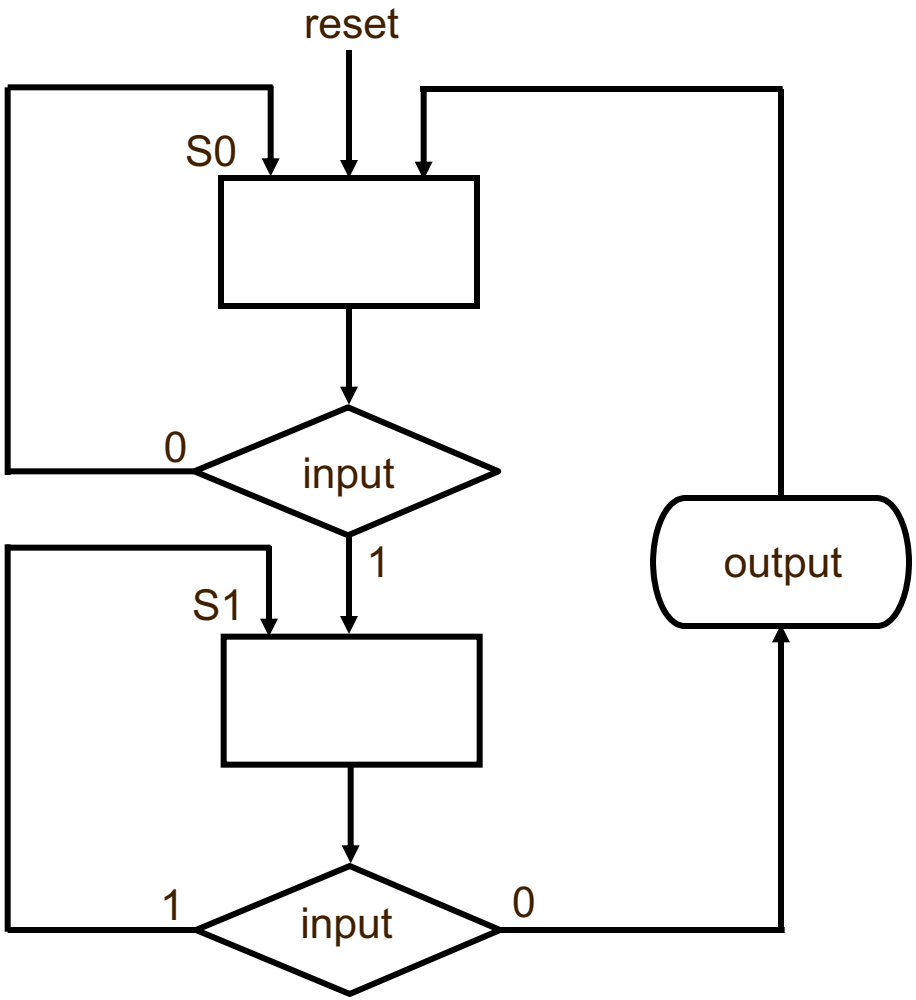


Meaning of states:

S0: No elements of the sequence observed

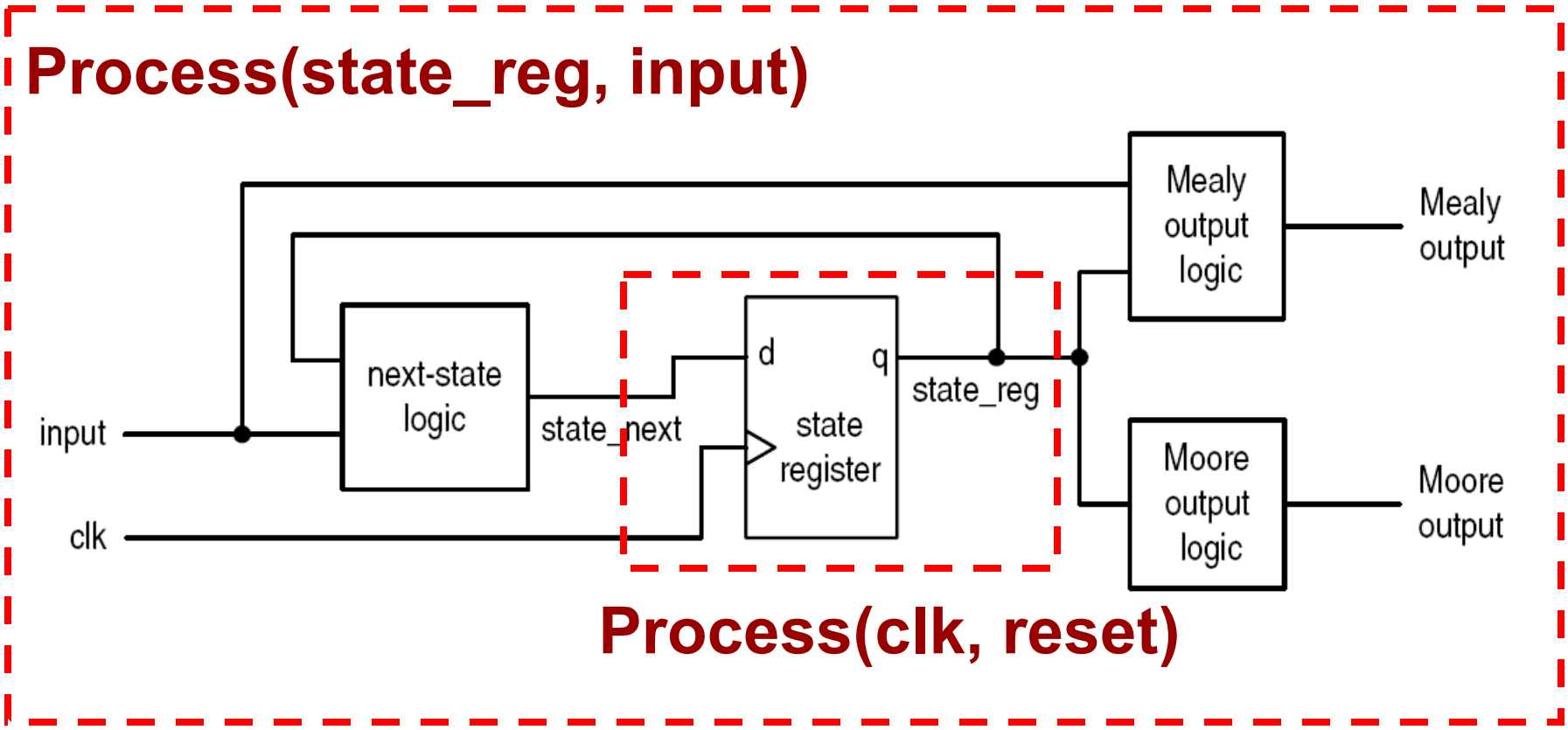
S1: "1" observed

ASM Chart of Mealy Machine



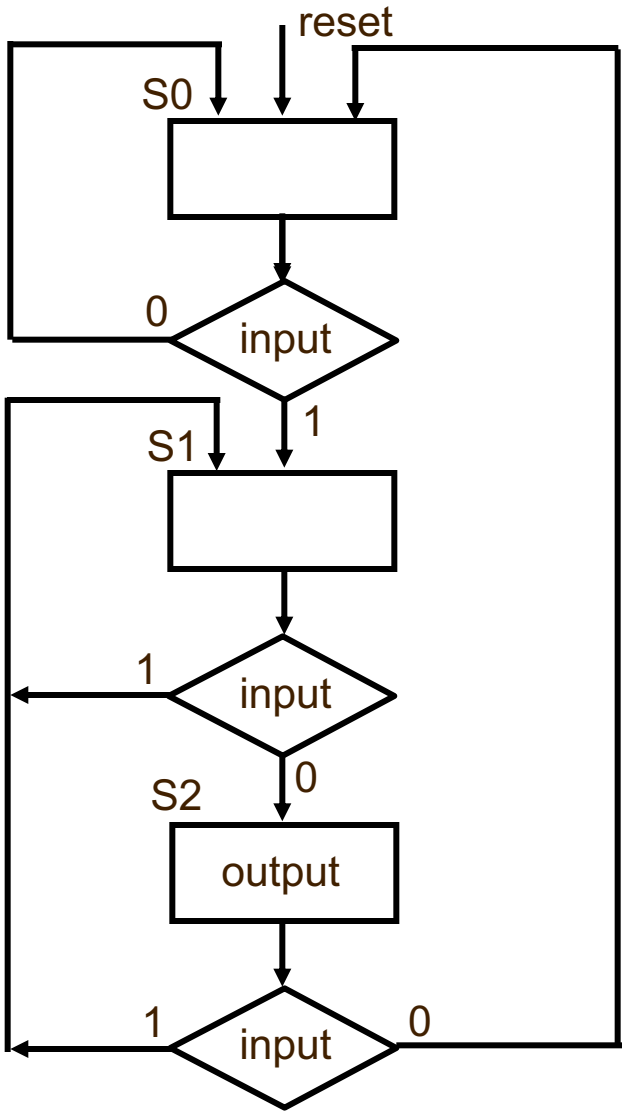
Finite State Machines in VHDL

Recommended FSM Coding Style



Based on RTL Hardware Design by P. Chu

ASM Chart of Moore Machine



Moore FSM in VHDL (1)

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY FSM_Moore IS
      PORT ( clk          : IN     STD_LOGIC ;
             reset        : IN     STD_LOGIC ;
             input        : IN     STD_LOGIC ;
             output       : OUT    STD_LOGIC) ;
END FSM_Moore ;
```

Moore FSM in VHDL (1)

```
ARCHITECTURE behavioral of FSM_Moore IS
TYPE state IS (S0, S1, S2);
SIGNAL state_reg, state_next: state;

BEGIN

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

Moore FSM in VHDL (2)

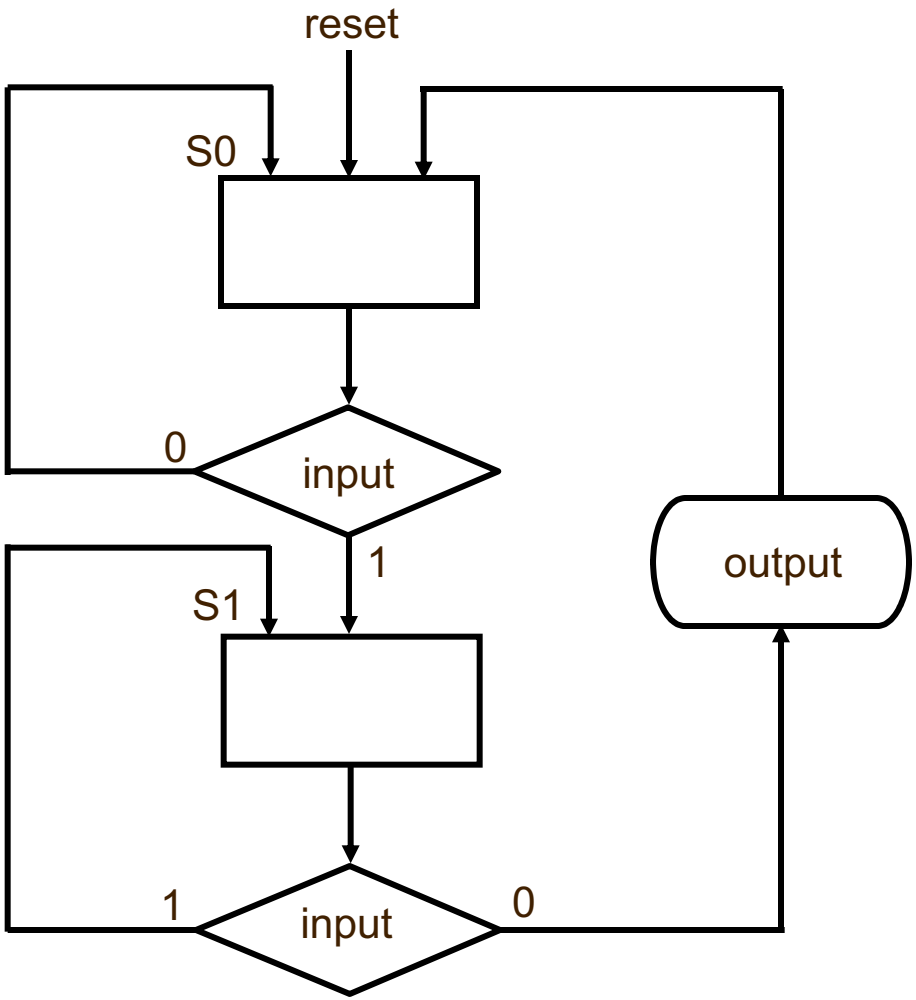
```
Next_State_Output:
PROCESS (state_reg, input)
BEGIN
      state_next <= state_reg;
      output <= '0';
      CASE state_reg IS
            WHEN S0 =>
             IF input = '1' THEN
                state_next <= S1;
             ELSE
                state_next <= S0;
             END IF;
```

Moore FSM in VHDL (3)

```
            WHEN S1 =>
                IF input = '0' THEN
                    state_next <= S2;
                ELSE
                    state_next <= S1;
                END IF;
            WHEN S2 =>
                output <= '1' ;
                IF input = '1' THEN
                    state_next <= S1;
                ELSE
                    state_next <= 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 IS
      PORT ( clk : IN        STD_LOGIC ;
             reset     : IN      STD_LOGIC ;
             input               : IN      STD_LOGIC ;
             output              : OUT     STD_LOGIC) ;
END FSM_Mealy ;
```

Mealy FSM in VHDL (1)

```
ARCHITECTURE behavioral of FSM_Mealy IS
TYPE state IS (S0, S1);
SIGNAL state_reg, state_next: state;

BEGIN
U_Mealy: PROCESS(clk, reset)
BEGIN
      IF(reset = '1') THEN
         state_reg <= S0;
      ELSIF rising_edge(clk) THEN
         state_reg <= state_next;
      END IF;
END PROCESS;
```

Mealy FSM in VHDL (2)

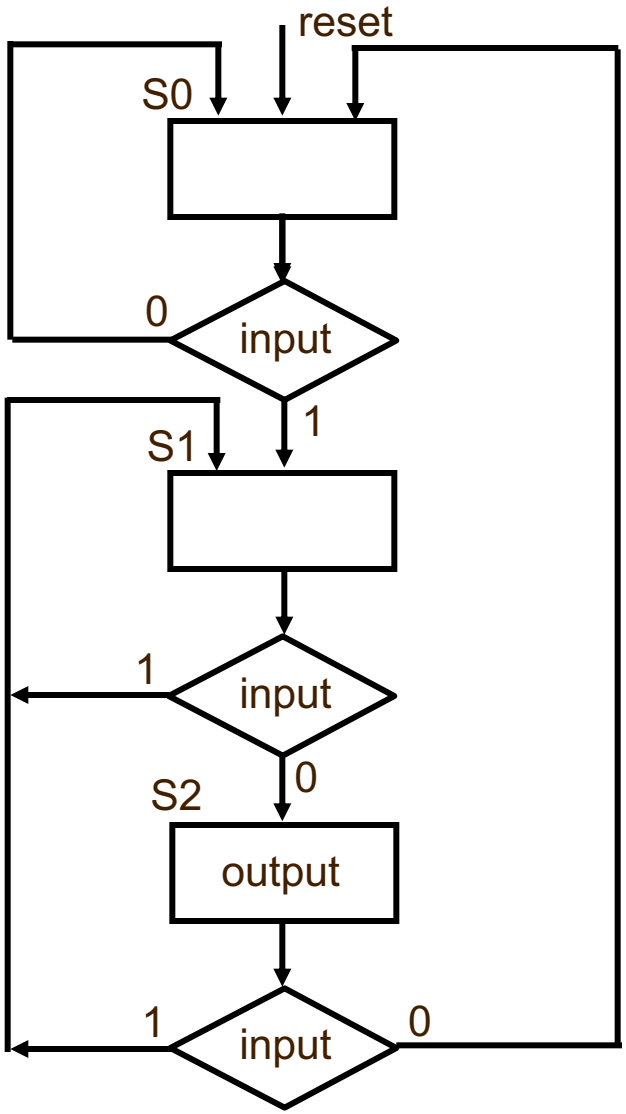
```
Next_State_Output:
PROCESS (state_reg, input)
BEGIN
      state_next <= state_reg;
      output <= '0';
      CASE state_reg IS
            WHEN S0 =>
               IF input = '1' THEN
                    state_next <= S1;
               ELSE
                    state_next <= S0;
              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

Mealy FSM in VHDL (3)

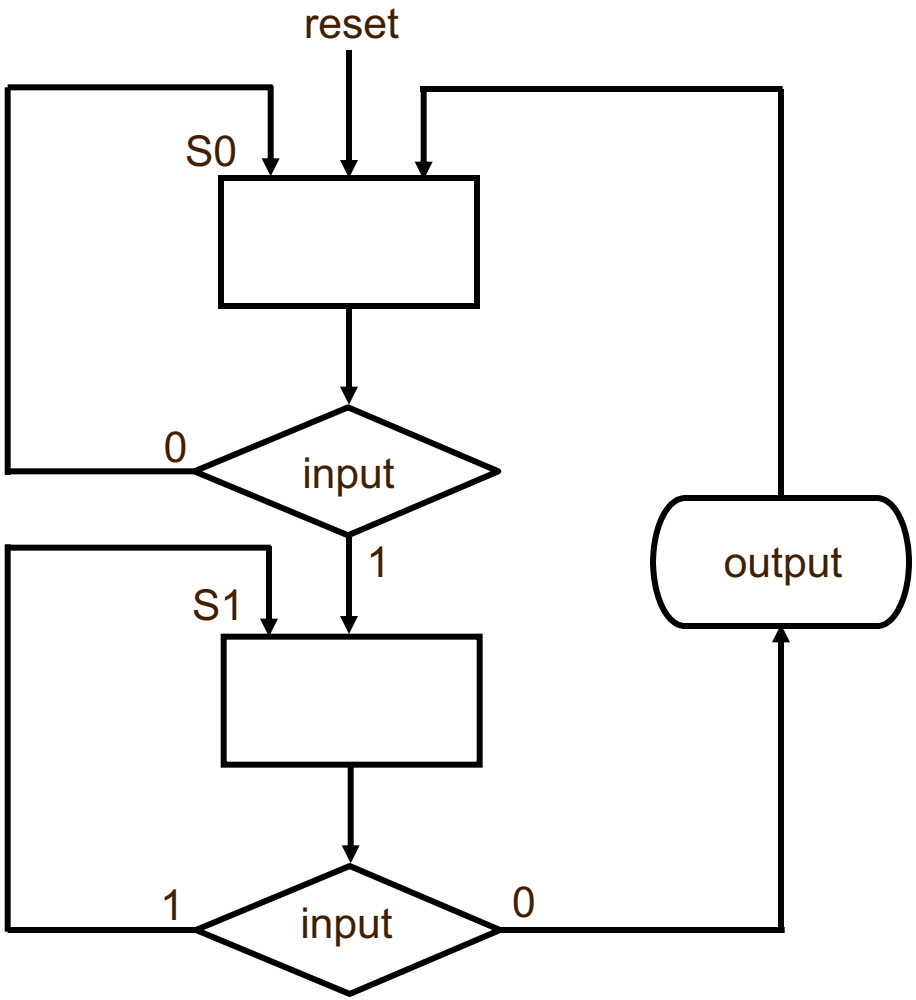
```
                WHEN S1 =>
                  IF input = '0' THEN
                        state_next <= S0;
                        output <= '1' ;
                  ELSE
                        state_next <= S1;
                  END IF;
        END CASE;
END PROCESS;

END behavioral;
```

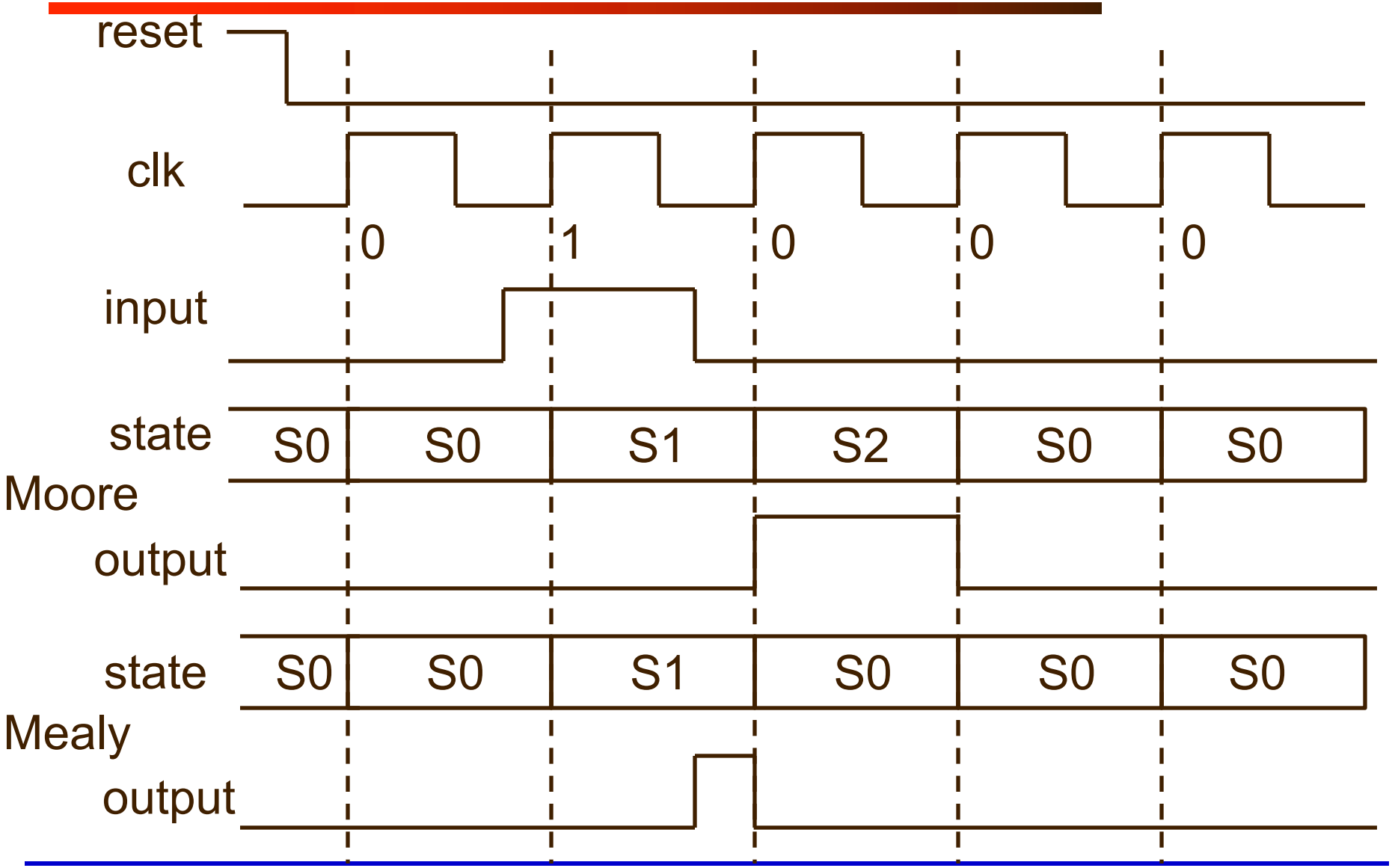
ASM Chart of Moore Machine



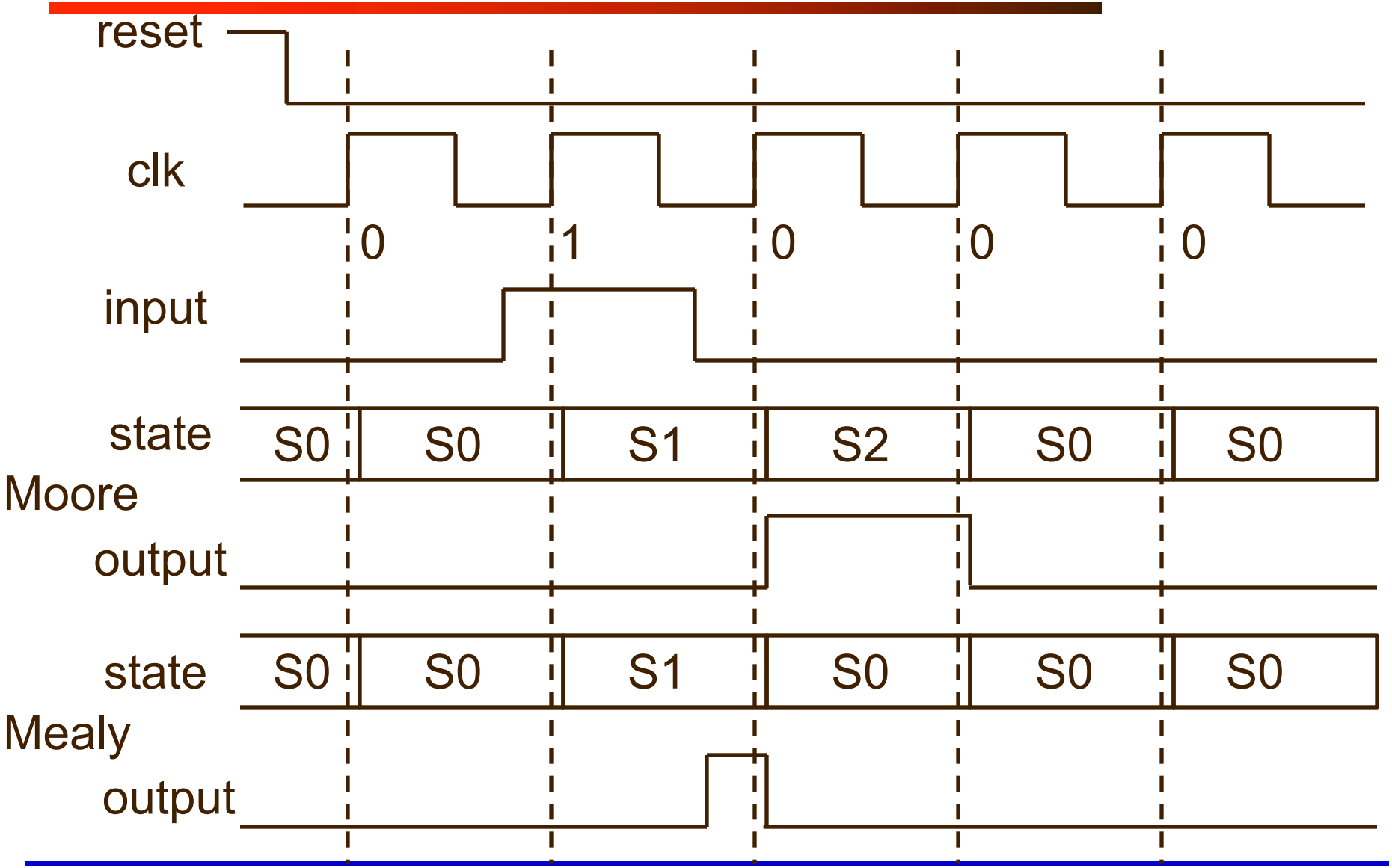
ASM Chart of Mealy Machine



Moore & Mealy FSMs without delays



Moore & Mealy FSMs with delays



Moore vs. Mealy FSM (1)

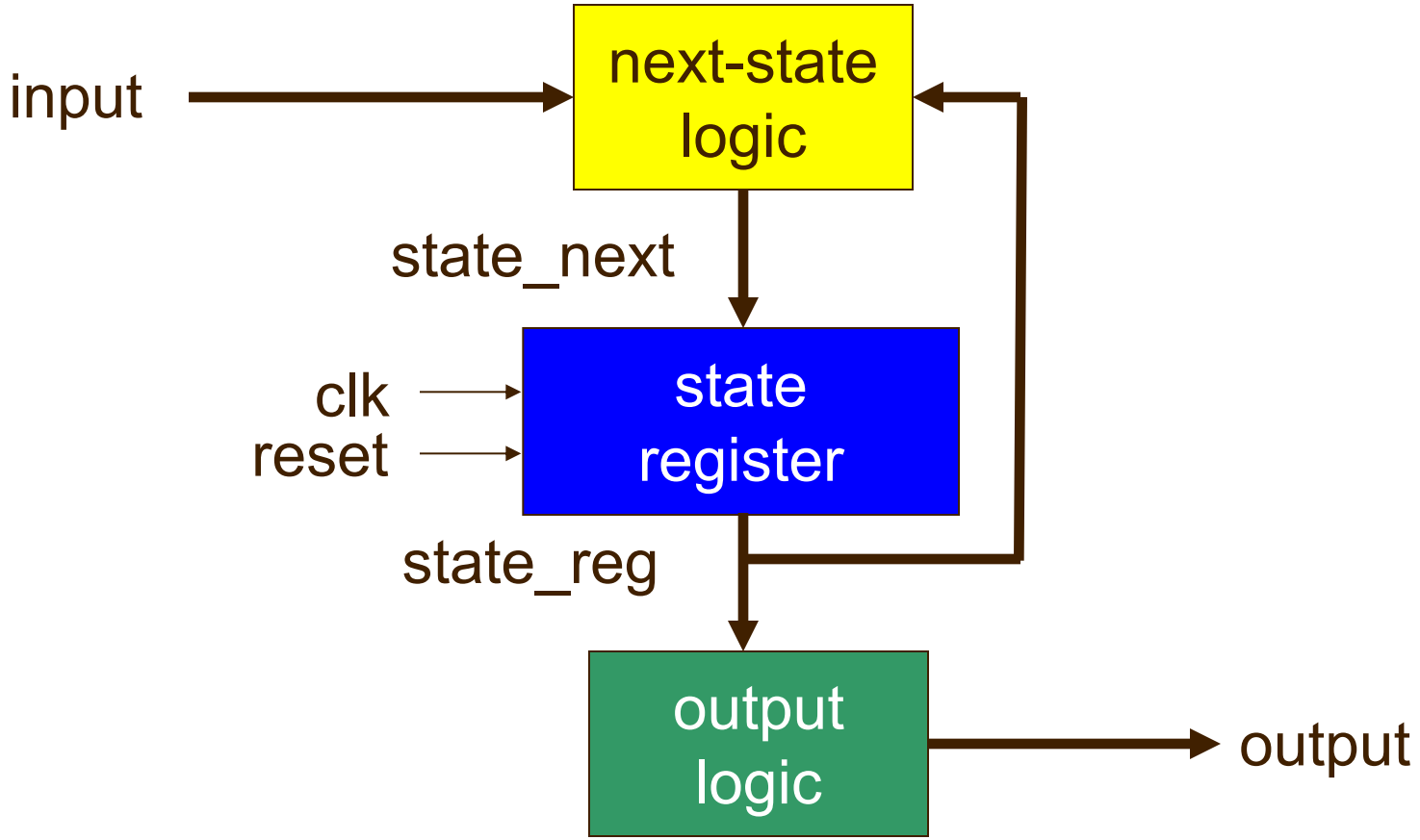
- Moore and Mealy FSMs Can Be Functionally Equivalent
  - Equivalent Mealy FSM can be derived from Moore FSM and vice versa
- Mealy FSM Has Richer Description and Usually Requires Smaller Number of States
  - Smaller circuit area

Moore vs. Mealy FSM (2)

- Mealy FSM Computes Outputs as soon as Inputs Change
  - Mealy FSM responds one clock cycle sooner than equivalent Moore FSM
- Moore FSM Has No Combinational Path Between Inputs and Outputs
  - Moore FSM is less likely to affect the critical path of the entire circuit

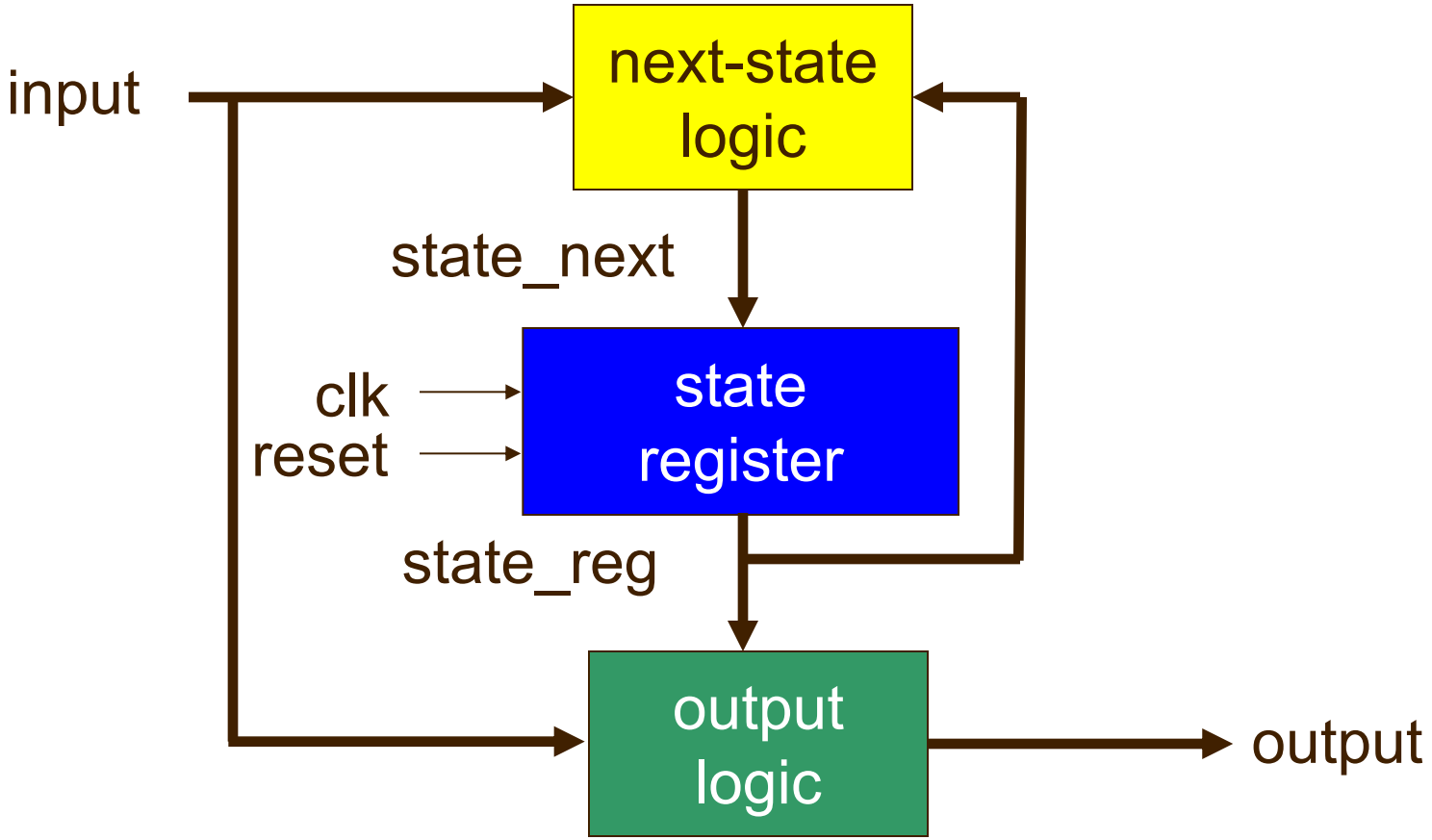
Moore FSM

- output is a function of the state only



Mealy FSM

- output is a function of the state and input signals



Which Way to Go?

Mealy FSM

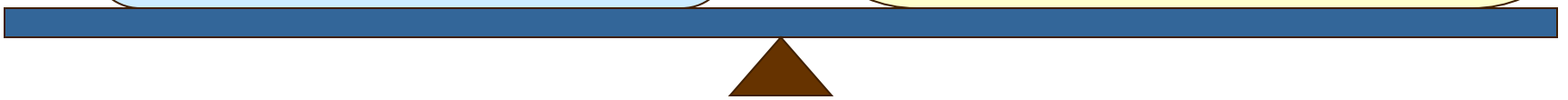
Fewer states

Lower Area

Responds one clock cycle earlier

Moore FSM

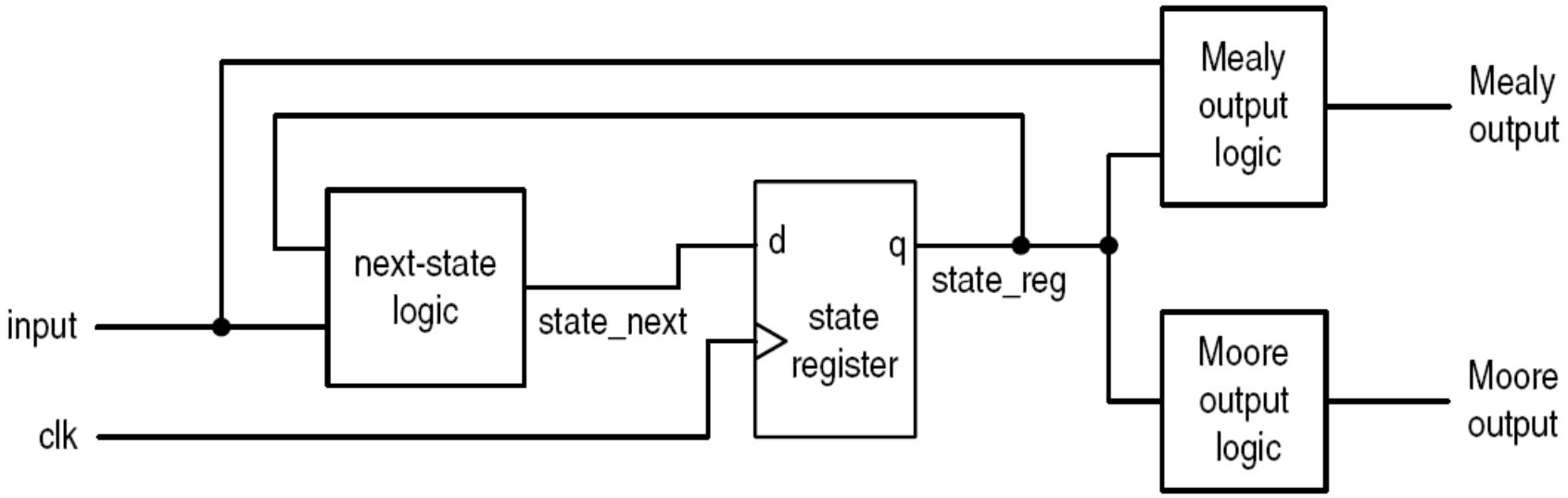
Safer.
Less likely to affect the critical path.



ASMs representing simple FSMs

- Algorithmic state machines can model both Mealy and Moore Finite State Machines
- They can also model machines that are of the mixed type

Generalized FSM



Based on RTL Hardware Design by P. Chu