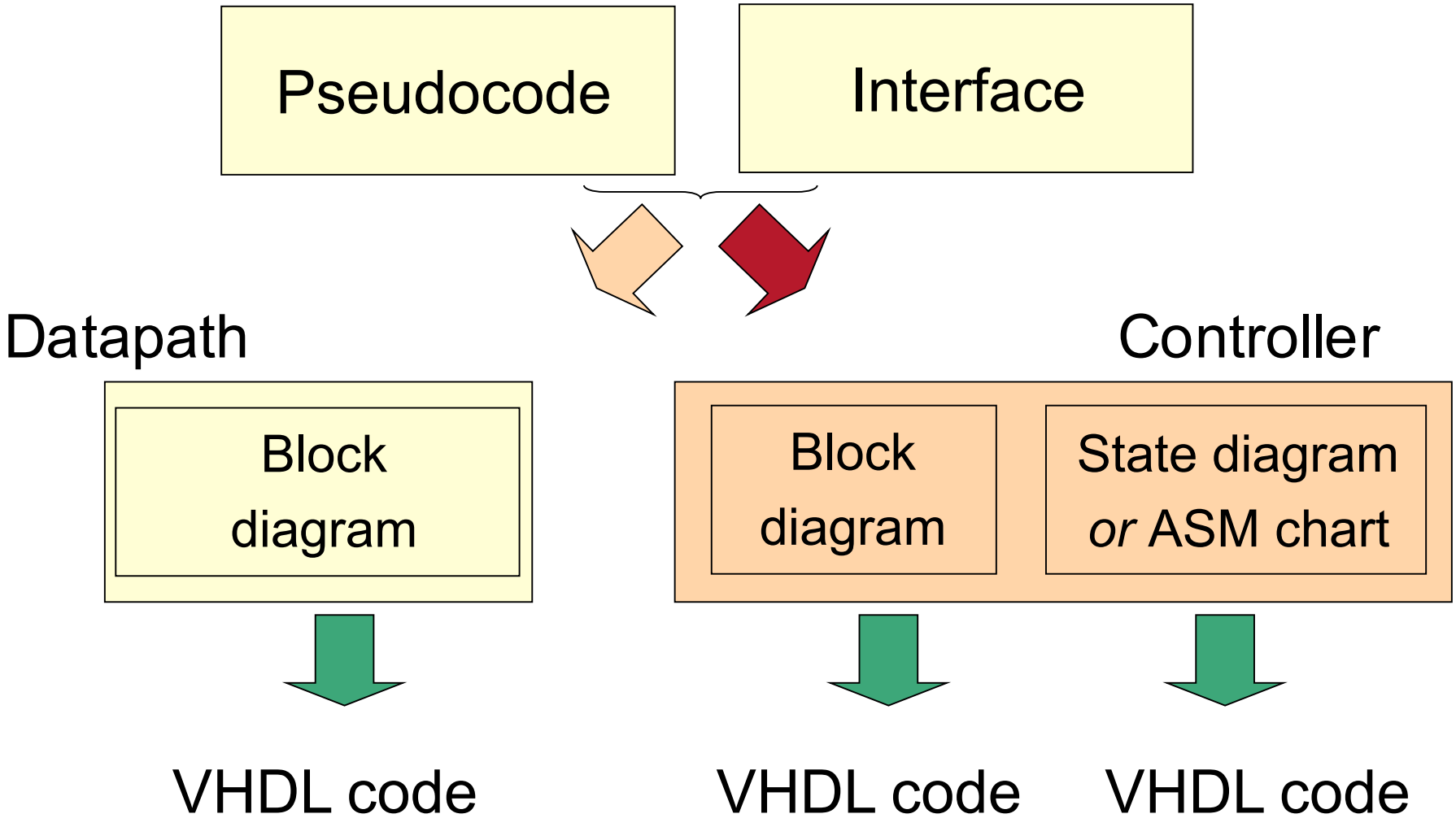
Finite State Machines

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

Hardware Design with RTL VHDL

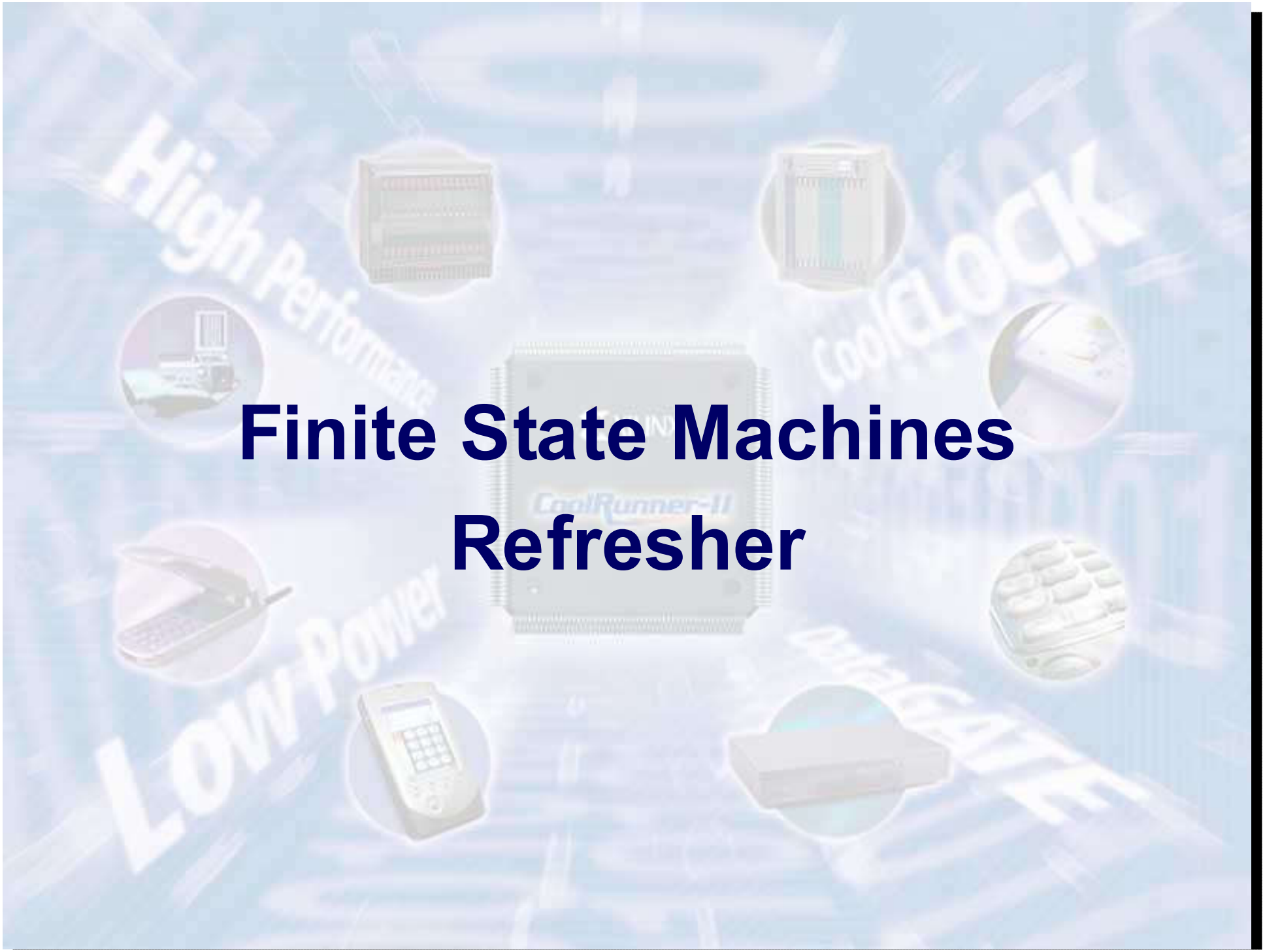


Steps of the Design Process

1. Text description
2. Interface
3. Pseudocode
4. Block diagram of the Datapath
5. Interface with the division into the Datapath and the Controller
6. ASM chart of the Controller
7. RTL VHDL code of the Datapath, the Controller, and the Top Unit
8. Testbench of the Datapath, the Controller, and the Top Unit
9. Functional simulation and debugging
10. Synthesis and post-synthesis simulation
11. Implementation and timing simulation
12. Experimental testing

Steps of the Design Process Practiced in Class Today

1. Text description
2. Interface
3. Pseudocode
4. Block diagram of the Datapath
5. Interface with the division into the Datapath and the Controller
6. **ASM chart of the Controller**
7. **RTL VHDL code of** the Datapath, **the Controller**, and the Top Unit
8. **Testbench of the** Datapath, **the Controller**, and the Top Unit
9. Functional simulation and debugging
10. Synthesis and post-synthesis simulation
11. Implementation and timing simulation
12. Experimental testing



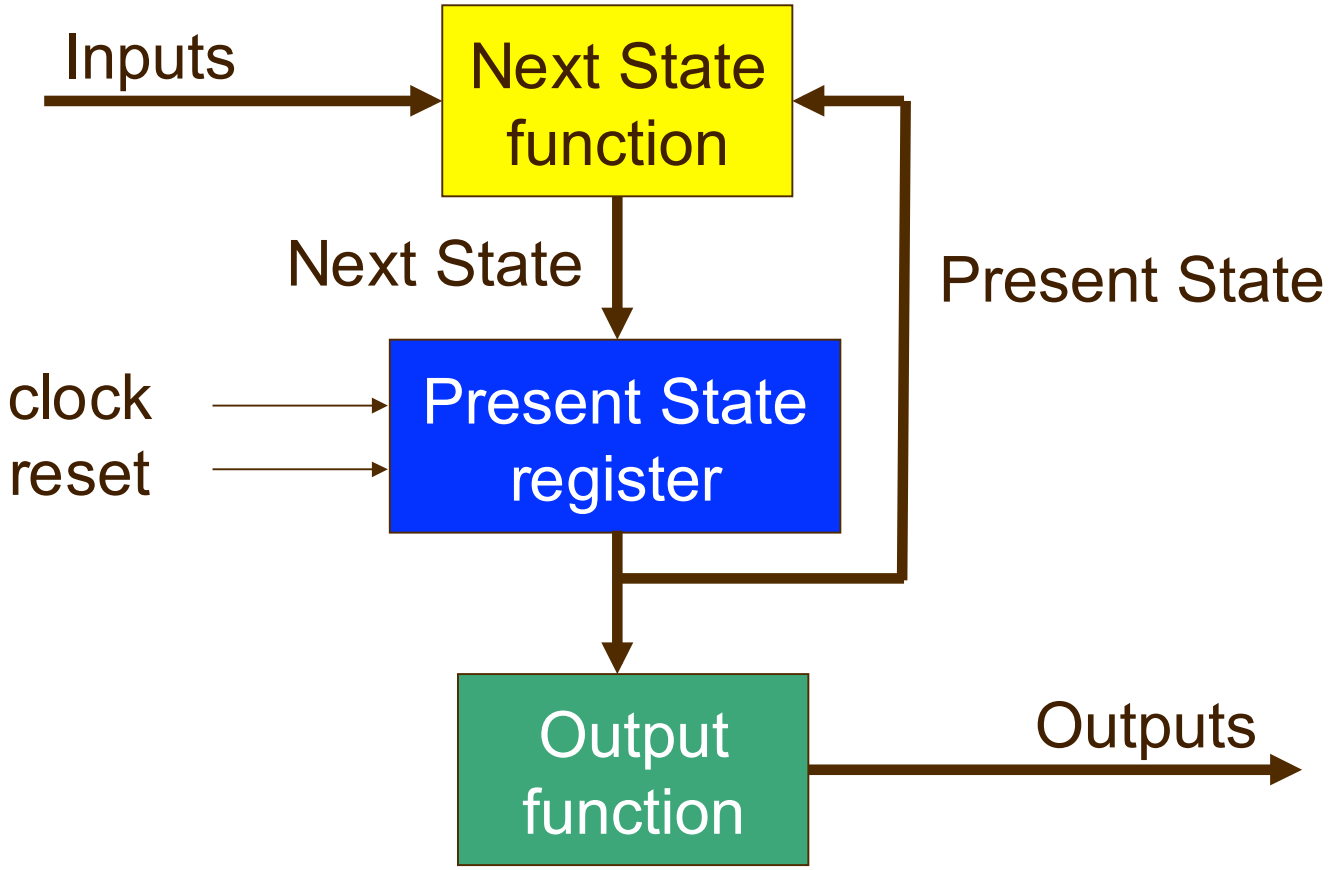
Finite State Machines Refresher

Finite State Machines (FSMs)

- Any Circuit with Memory Is a Finite State Machine
  - Even computers can be viewed as huge FSMs
- Design of FSMs Involves
  - Defining states
  - Defining transitions between states
  - Optimization / minimization
- Manual Optimization/Minimization Is Practical for Small FSMs Only

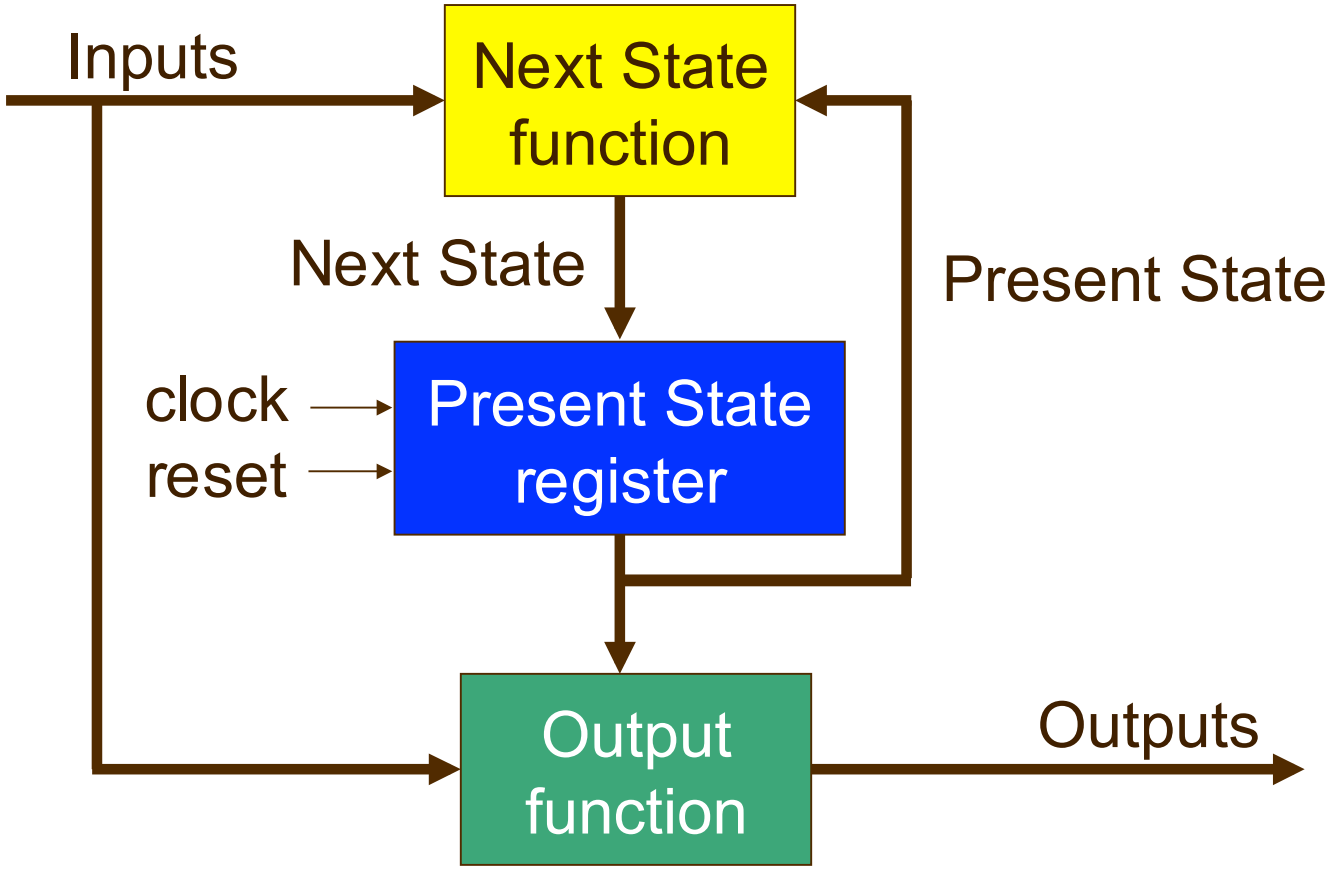
Moore FSM

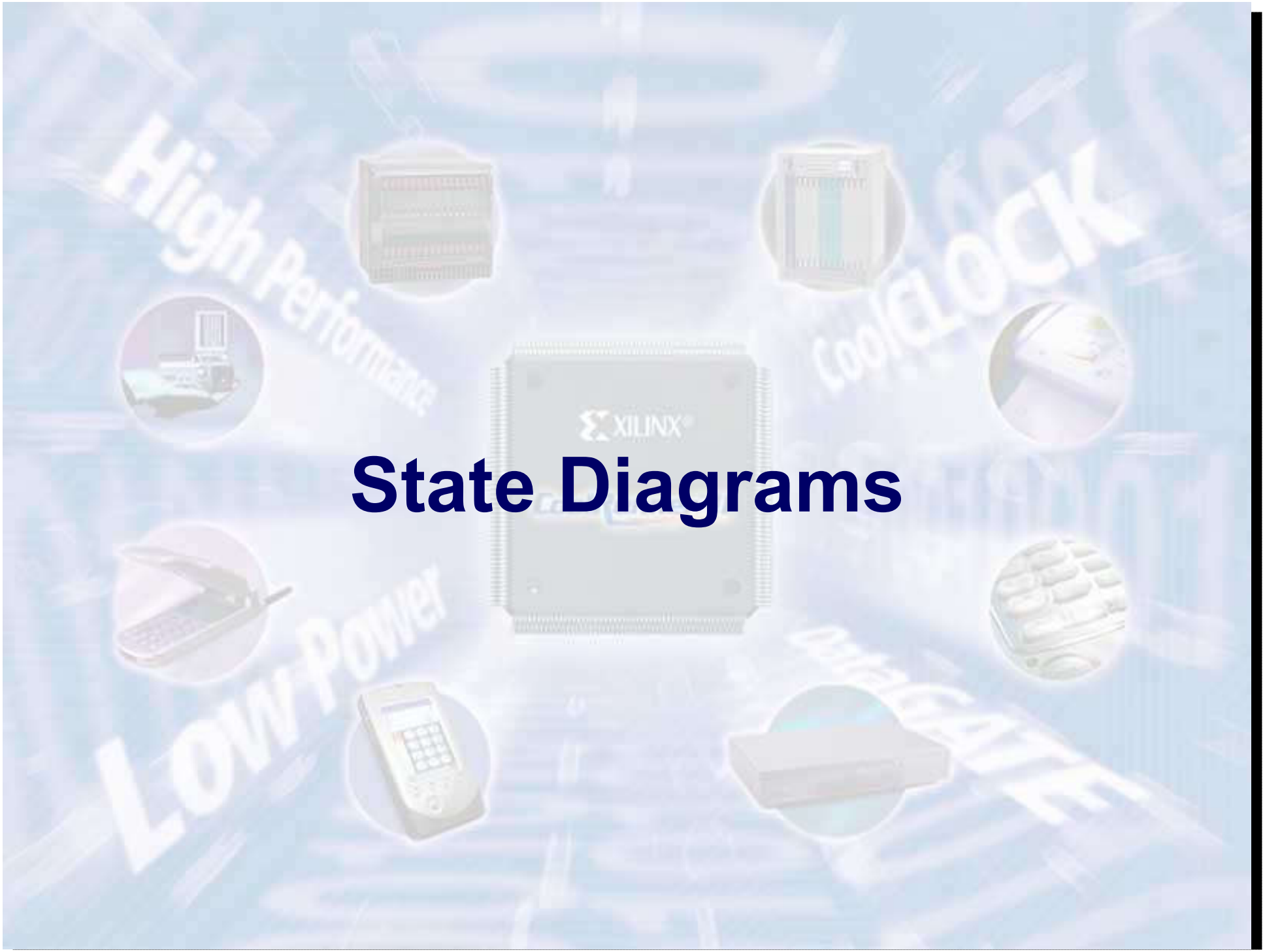
- Output Is a Function of a Present State Only



Mealy FSM

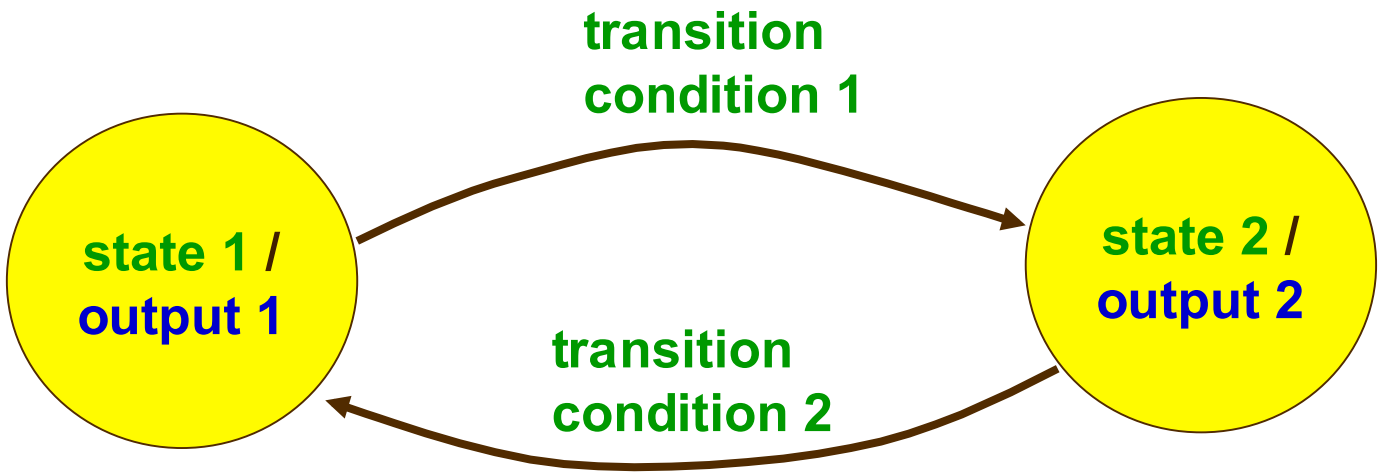
- Output Is a Function of a Present State and Inputs



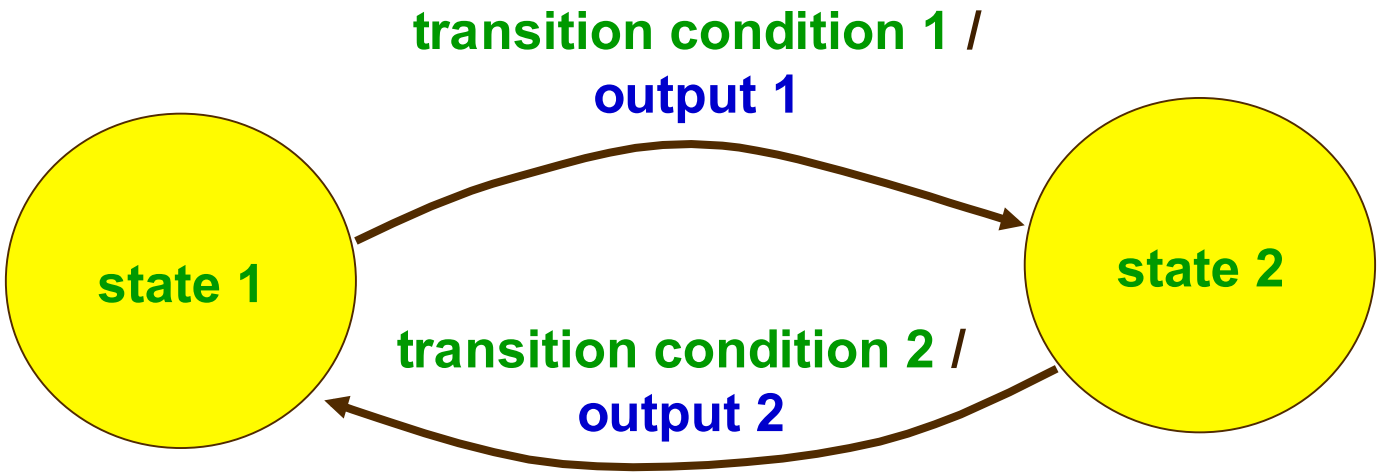


State Diagrams

Moore Machine



Mealy Machine



Moore vs. Mealy FSM (1)

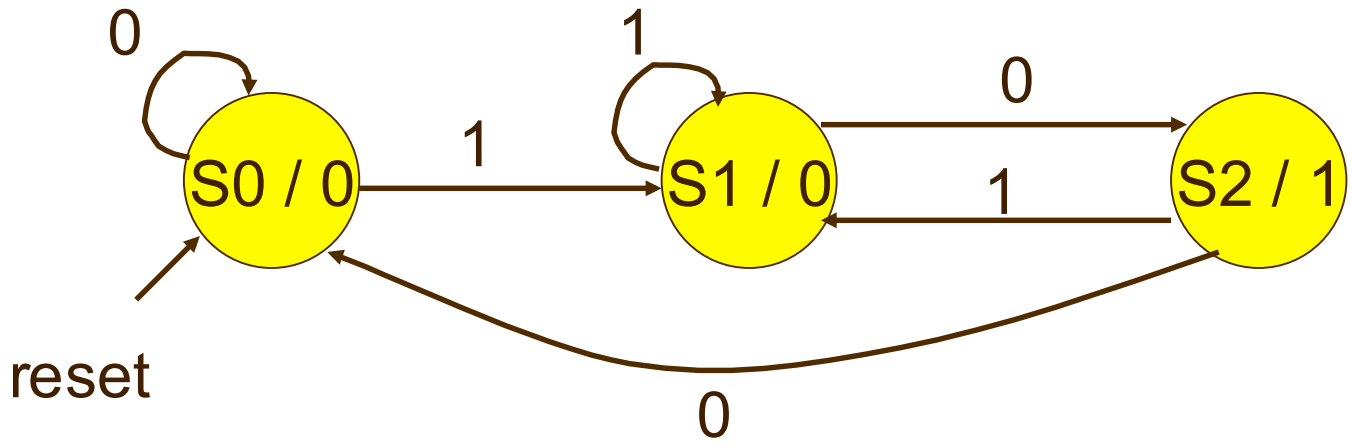
- Moore and Mealy FSMs Can Be Functionally Equivalent
  - Equivalent Mealy FSM can be derived from Moore FSM and vice versa
- Mealy FSM Has Richer Description and Usually Requires Smaller Number of States
  - Smaller circuit area

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 - Example 1

- Moore FSM that Recognizes Sequence “10”



Meaning of states:

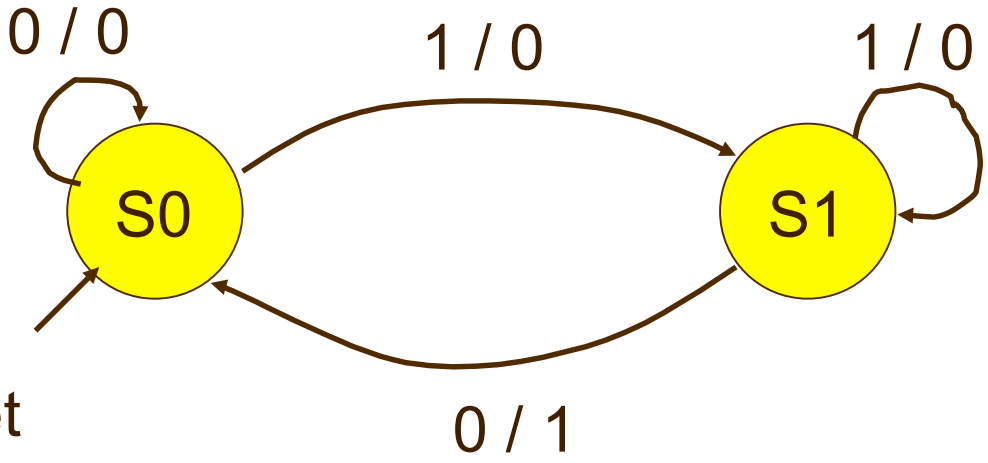
S0: No elements of the sequence observed

S1: “1” observed

S2: “10” observed

Mealy FSM - Example 1

- Mealy FSM that Recognizes Sequence “10”

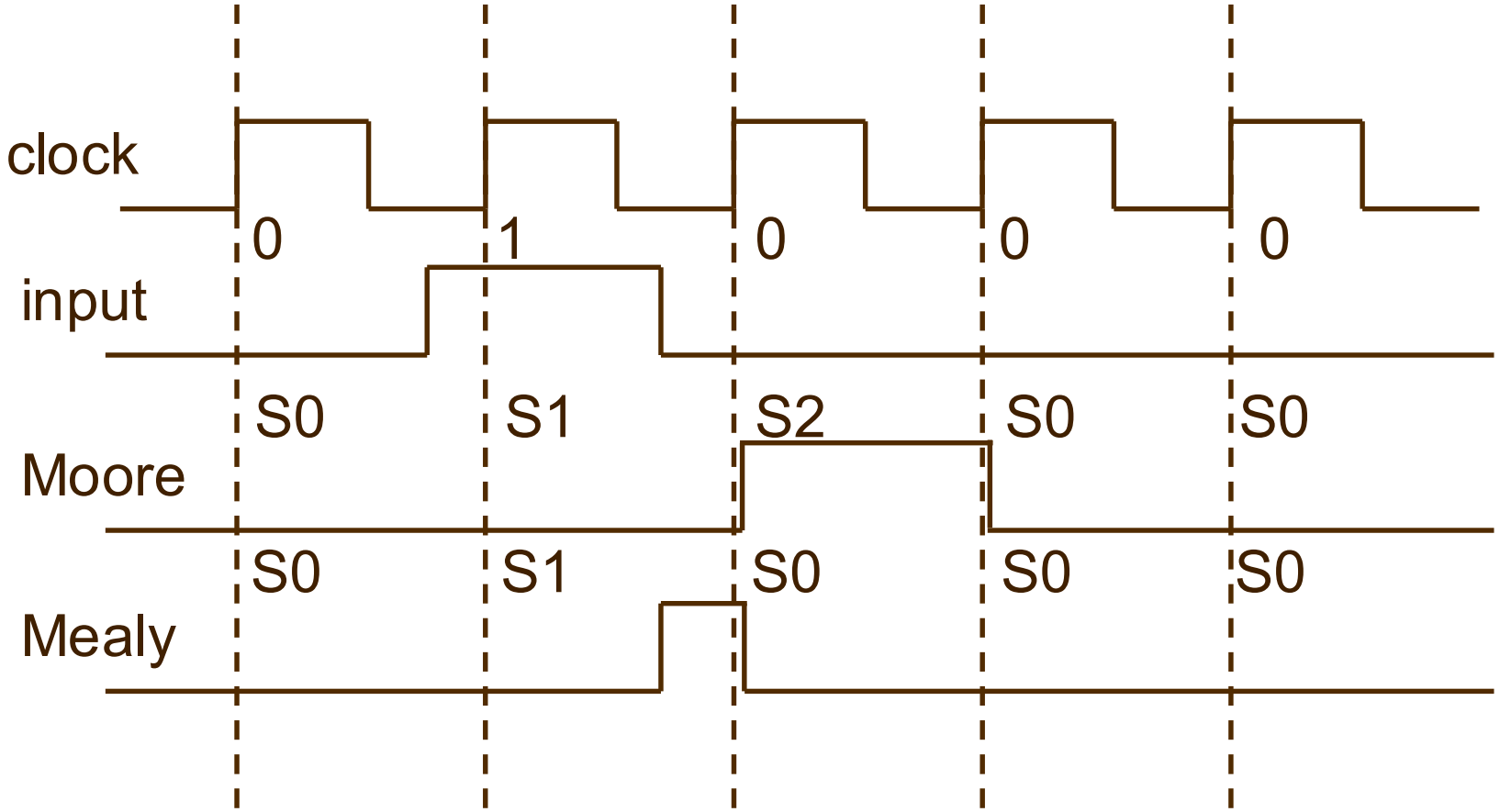


Meaning of states:

S0: No elements of the sequence observed

S1: “1” observed

Moore & Mealy FSMs – Example 1



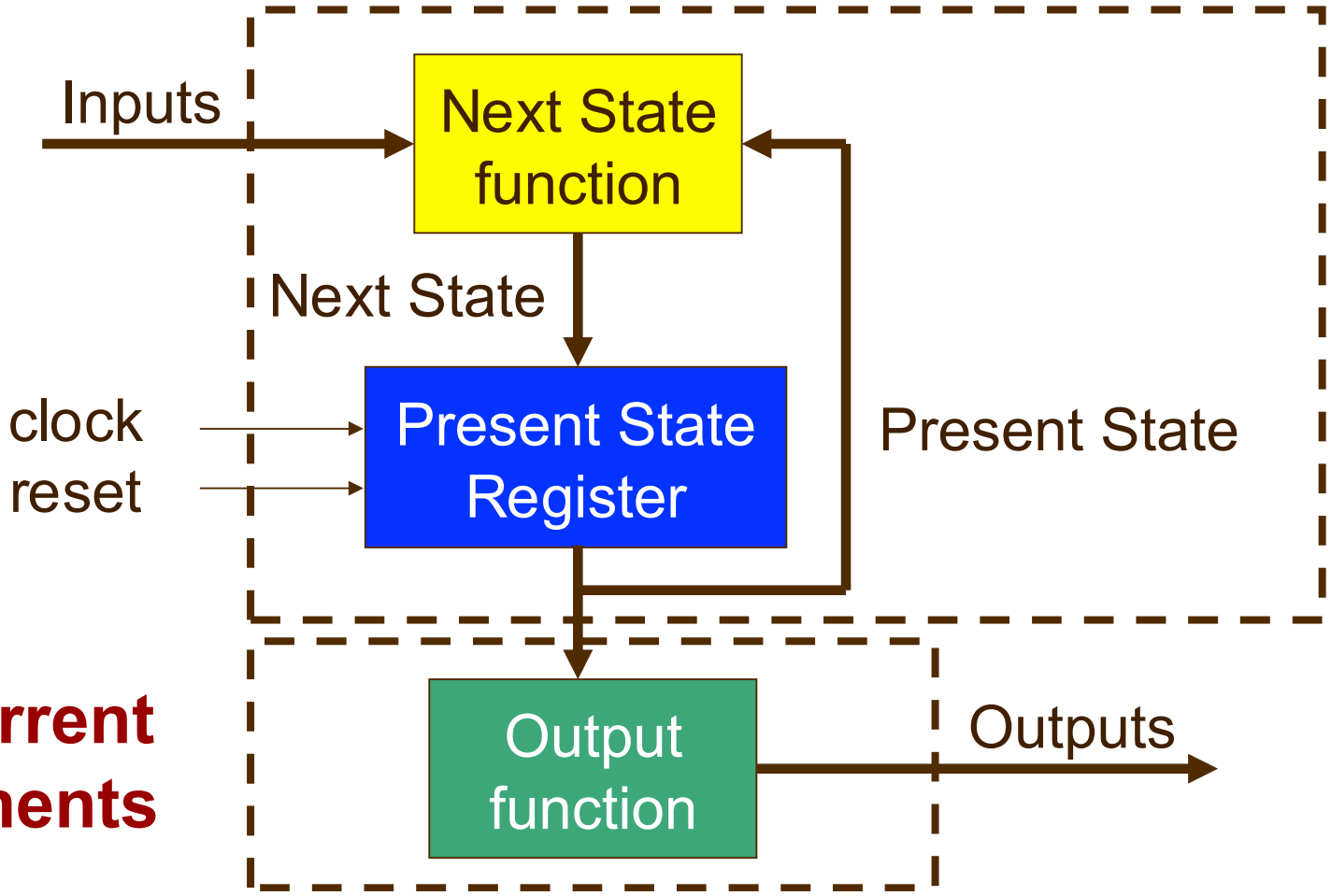
Finite State Machines in VHDL

FSMs in VHDL

- Finite State Machines Can Be Easily Described With Processes
- Synthesis Tools Understand FSM Description if Certain Rules Are Followed
  - State transitions should be described in a process sensitive to *clock* and *asynchronous reset* signals only
  - Output function described using rules for combinational logic, i.e. as concurrent statements or a process with all inputs in the sensitivity list

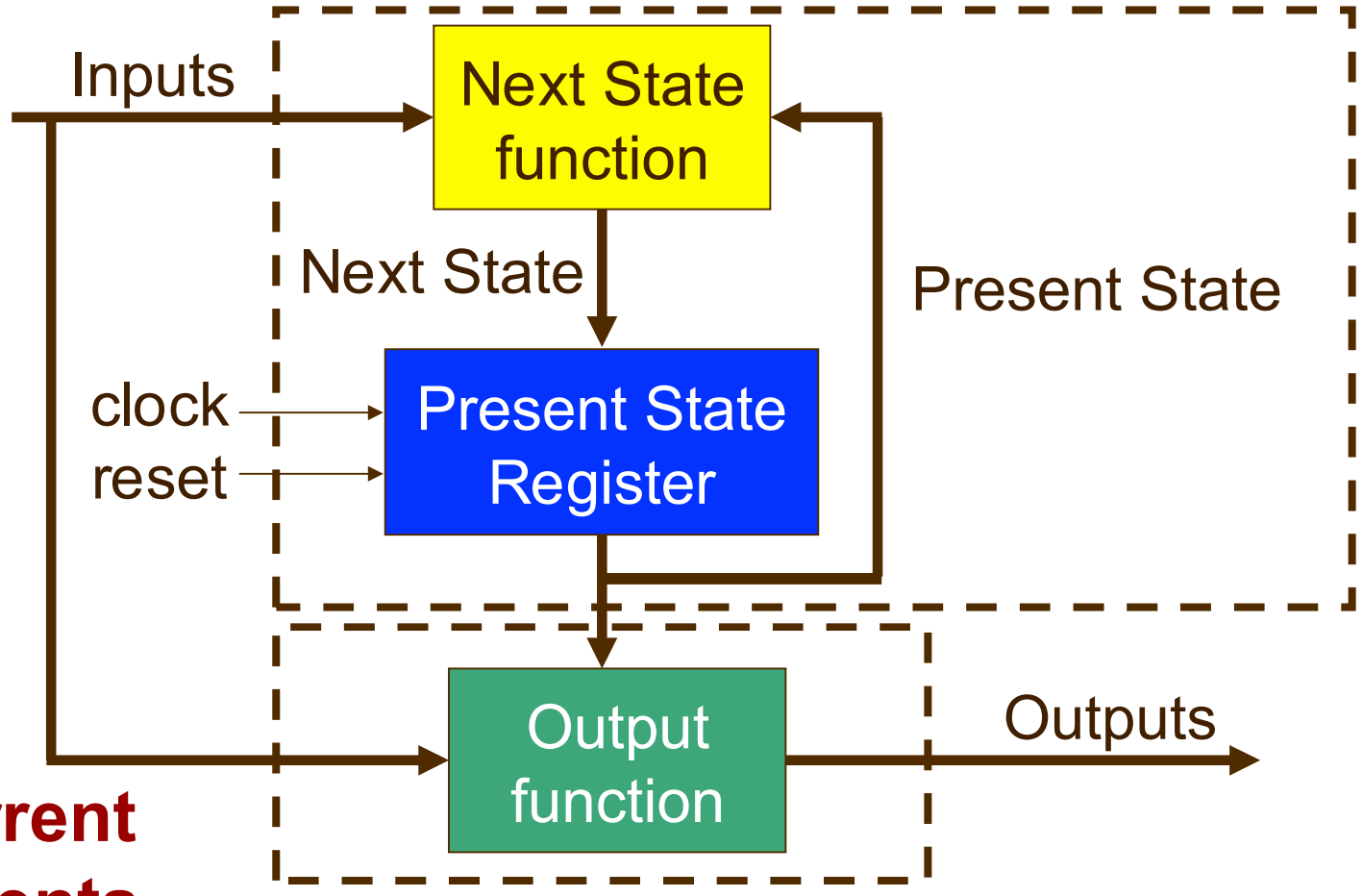
Moore FSM

process(clock, reset)



Mealy FSM

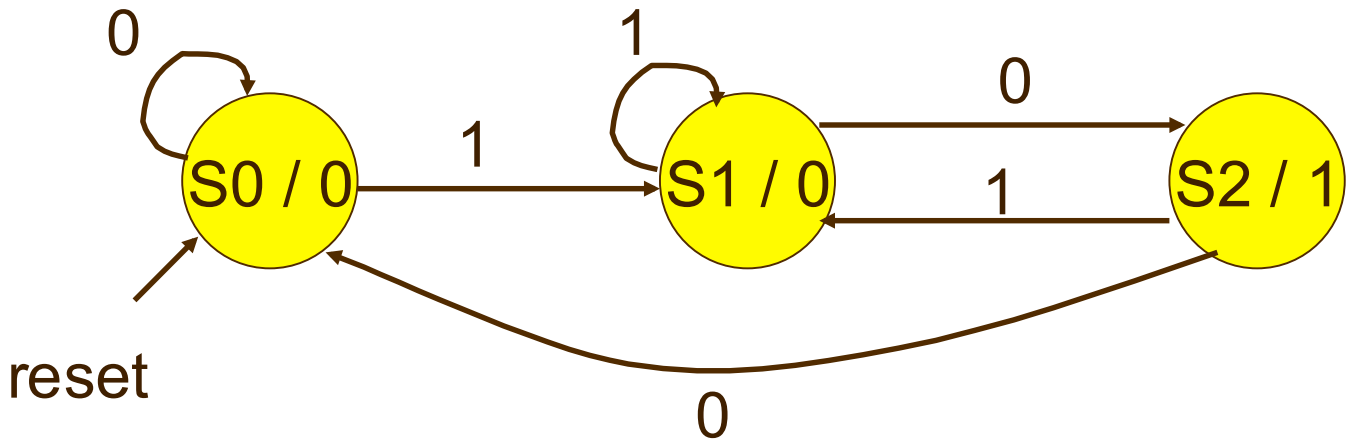
process(clock, reset)



concurrent statements

Moore FSM - Example 1

- Moore FSM that Recognizes Sequence “10”



Moore FSM in VHDL (1)

```
TYPE state IS (S0, S1, S2);
SIGNAL Moore_state: state;

U_Moore: PROCESS (clock, reset)
BEGIN
      IF(reset = '1') THEN
         Moore_state <= S0;
      ELSIF (clock = '1' AND clock'event) THEN
         CASE Moore_state IS
            WHEN S0 =>
             IF input = '1' THEN
                Moore_state <= S1;
             ELSE
               Moore_state <= S0;
             END IF;
```

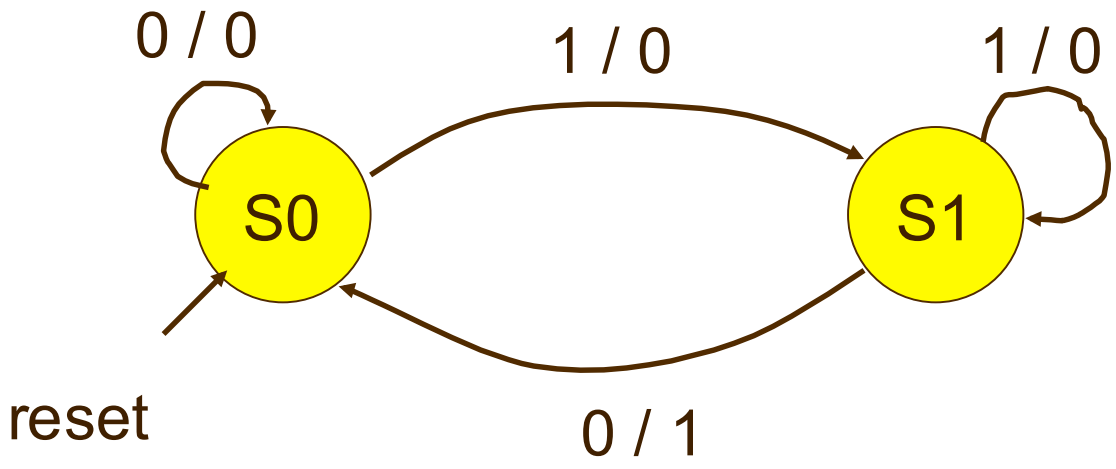
Moore FSM in VHDL (2)

```
            WHEN S1 =>
                IF input = '0' THEN
                    Moore_state <= S2;
                ELSE
                    Moore_state <= S1;
                END IF;
            WHEN S2 =>
                IF input = '0' THEN
                    Moore_state <= S0;
                ELSE
                    Moore_state <= S1;
                END IF;
        END CASE;
    END IF;
END PROCESS;

Output <= '1' WHEN Moore_state = S2 ELSE '0';
```

Mealy FSM - Example 1

- Mealy FSM that Recognizes Sequence “10”



Mealy FSM in VHDL (1)

```
TYPE state IS (S0, S1);
SIGNAL Mealy_state: state;

U_Mealy: PROCESS(clock, reset)
BEGIN
      IF(reset = '1') THEN
         Mealy_state <= S0;
      ELSIF (clock = '1' AND clock'event) THEN
         CASE Mealy_state IS
               WHEN S0 =>
                  IF input = '1' THEN
                       Mealy_state <= S1;
                  ELSE
                       Mealy_state <= S0;
                  END IF;
```

Mealy FSM in VHDL (2)

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

Output <= '1' WHEN (Mealy_state = S1 AND input = '0') ELSE '0';
```