EITF35: Introduction to Structured VLSI Design

Part 1.2.1: Finite State Machines

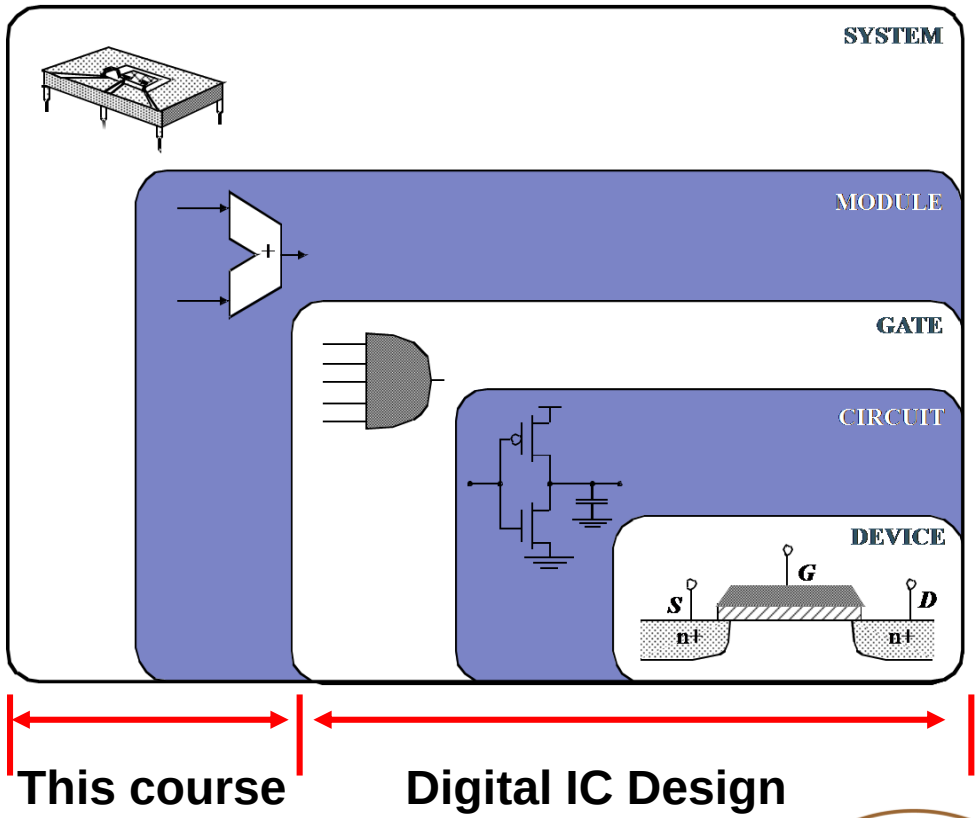
Liang Liu
liang.liu@eit.lth.se

Outline

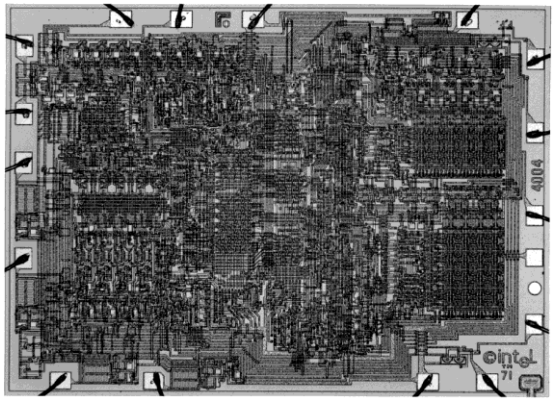
- Why Digital?
  - Advantages
  - Some applications
- History & Roadmap
- Device Technology & Platforms
- **System Representation**
- Design Flow
- RTL Basics

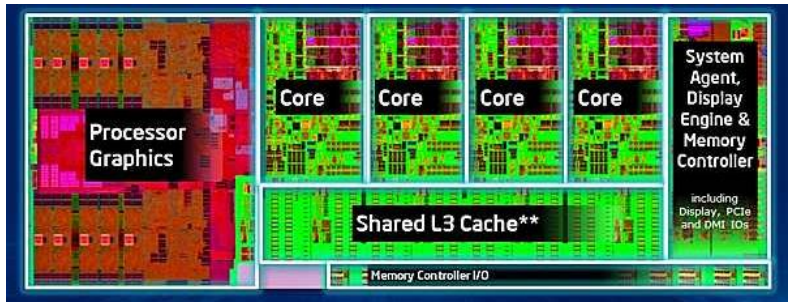
System Representation

- System
  - SoC: a CPU chip …
- Module
  - Macro cell in a chip：ALU…
- Gate
  - Basic logic block：xor, nor…
- Circuit
  - Transistors
- Device
  - Gate, source, drain



View a Design in a Proper Way





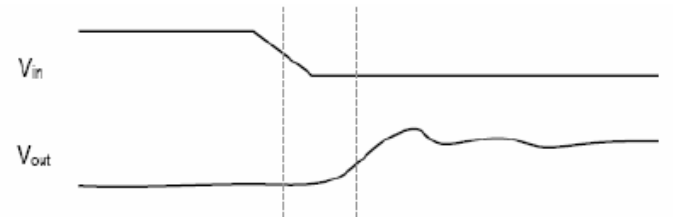
Intel 4004 (2.3K transistors)

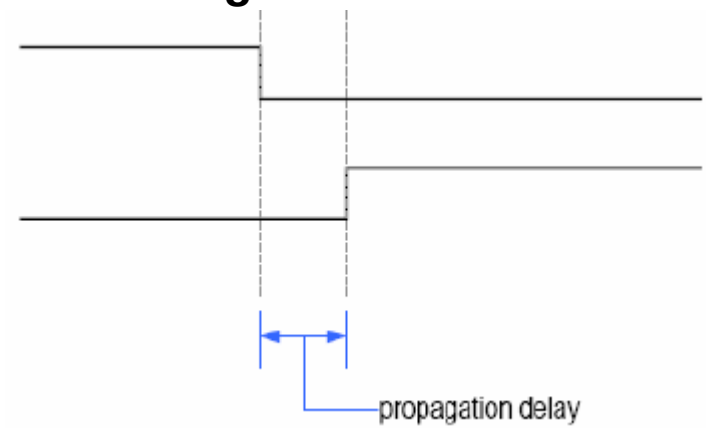
Full-custom

Intel Haswell (1.4B transistors)

?

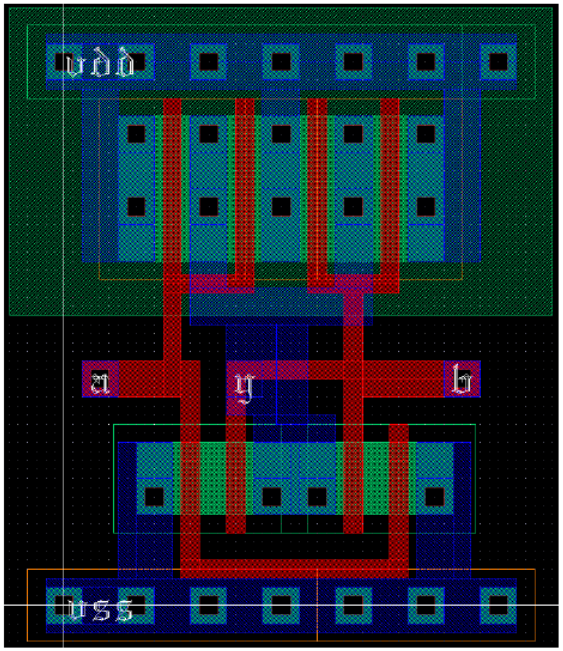
- **Abstraction: simplified model of a system**
  - **Show the selected features accurate enough**
  - **Ignore the others**





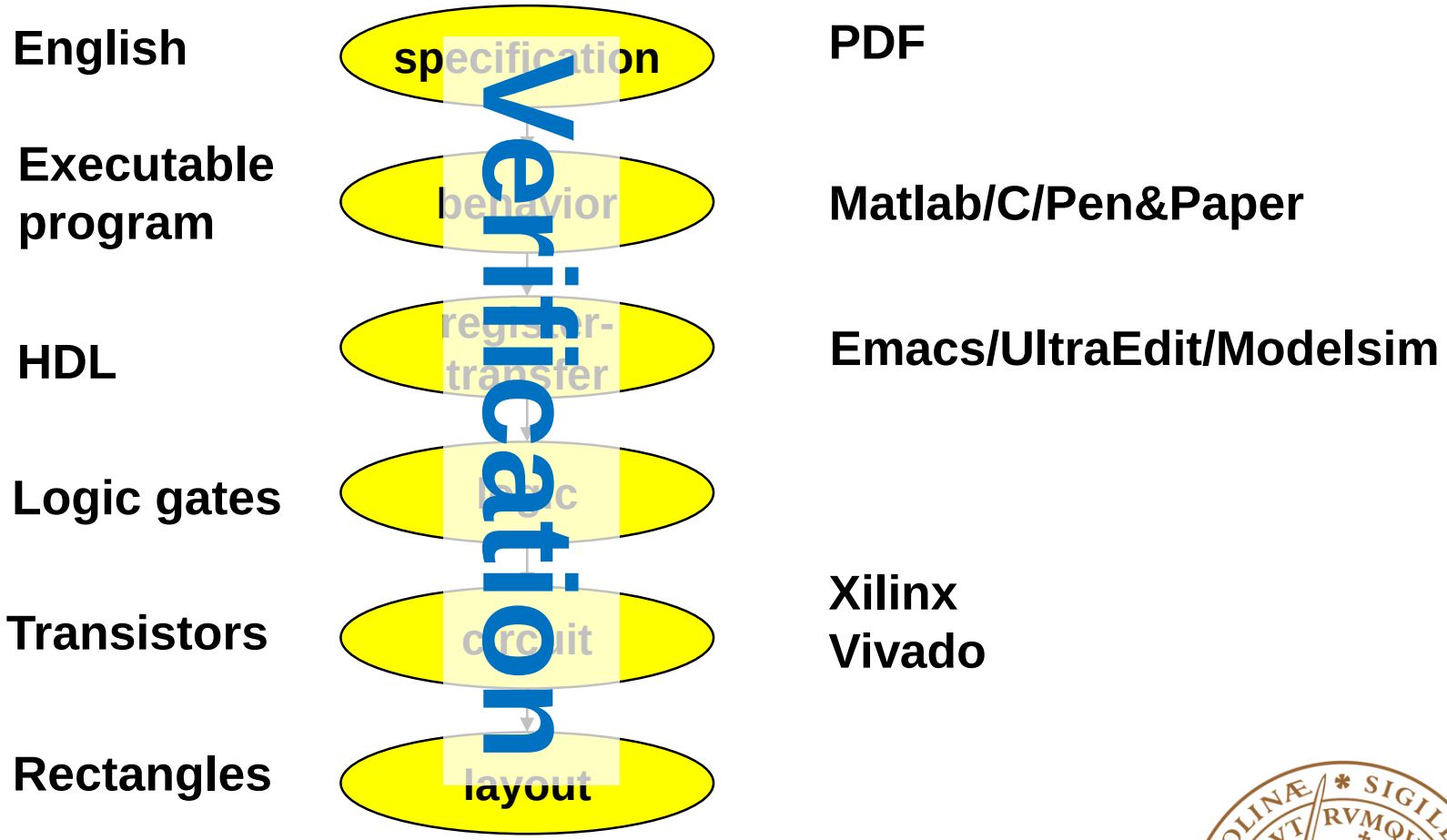
VLSI Design Flow

- **Evolution of circuit design (Design Hierarchy)**
  - Full-custom⇒Design-automation
    - ***Based on library cells and IPs***
    - ***Top-down methodology***
  - Design abstraction⇒“Black box” or “Model”
    - ***Parameter simplification***
    - ***Accurate enough to meet the requirement***



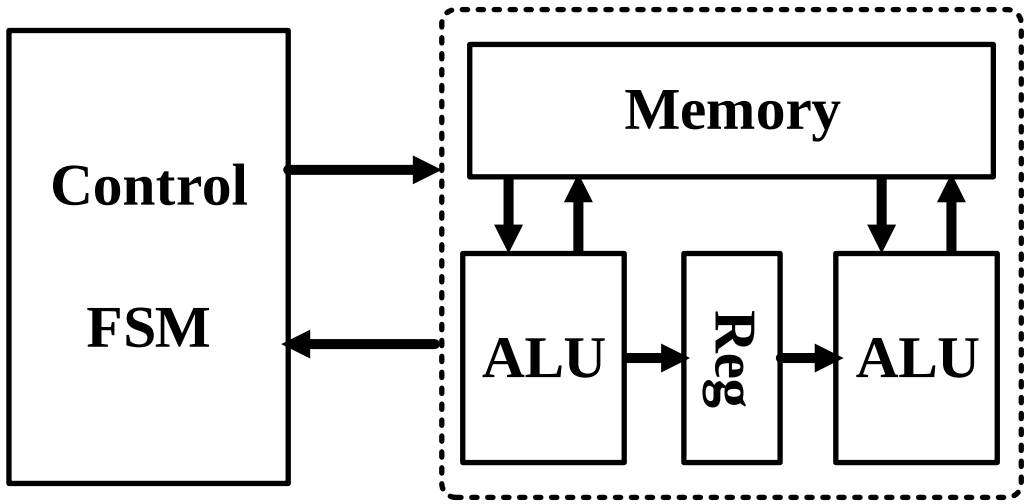

```
module HS65_GH_NAND2AX14 (Z, A, B);
   output Z;
   input A,B;
   not    U1 (INTERNAL1, B) ;
   or   #1 U2 (Z, A, INTERNAL1) ;
   specify
     (A +=> Z) = (0.1,0.1);
     (B -=> Z) = (0.1,0.1);
   endspecify
endmodule // HS65_GH_NAND2AX14
```

VLSI Design Flow (This course): Summary



Digital VLSI in 5 min

Overall VLSI Structure



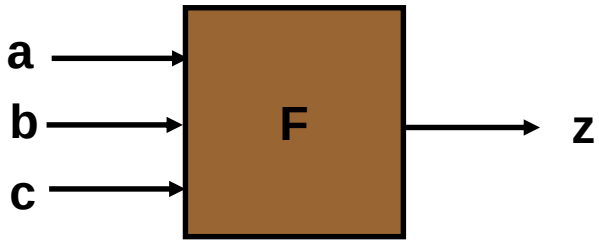

```
IF (a>10)
  b = c + d;
ELSE
  b = c – d;
```

- **Scheduling / ordering / sequencing of operations**
- **Mapping / allocation:**
  - Variables -> {Reg1, ... ,RegN}
  - Operations -> {MUL, ADD, ALU, ... ,}

We will implement something similar in this course

Two Basic Digital Components (What)

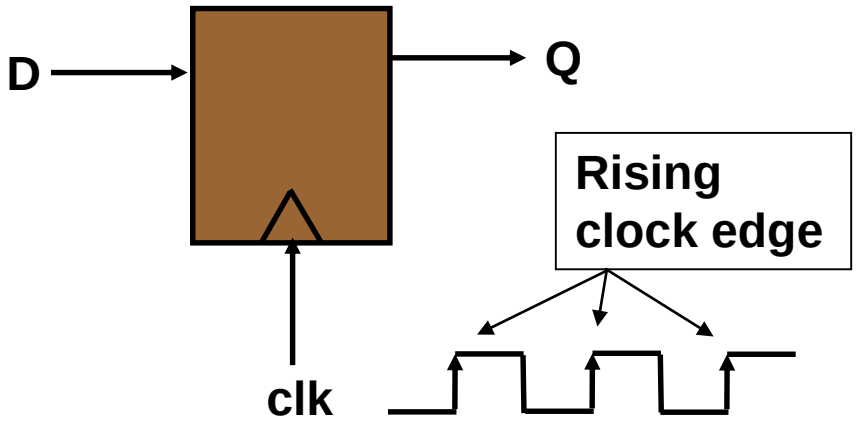
Combinational Logic



Always:
z <= F(a, b, c);

i.e. a function that is always evaluated when an input changes. Can be expressed by a truth table.

Register



if clk' event and clk= '1' then
Q <= D;

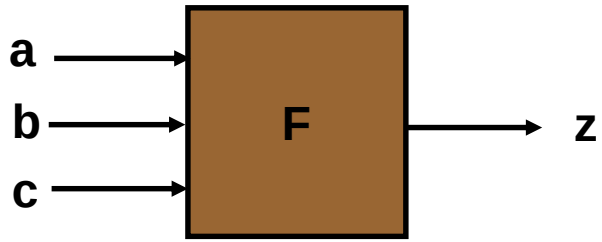
i.e. a stored variable, Edge triggered D Flip-Flop with enable.

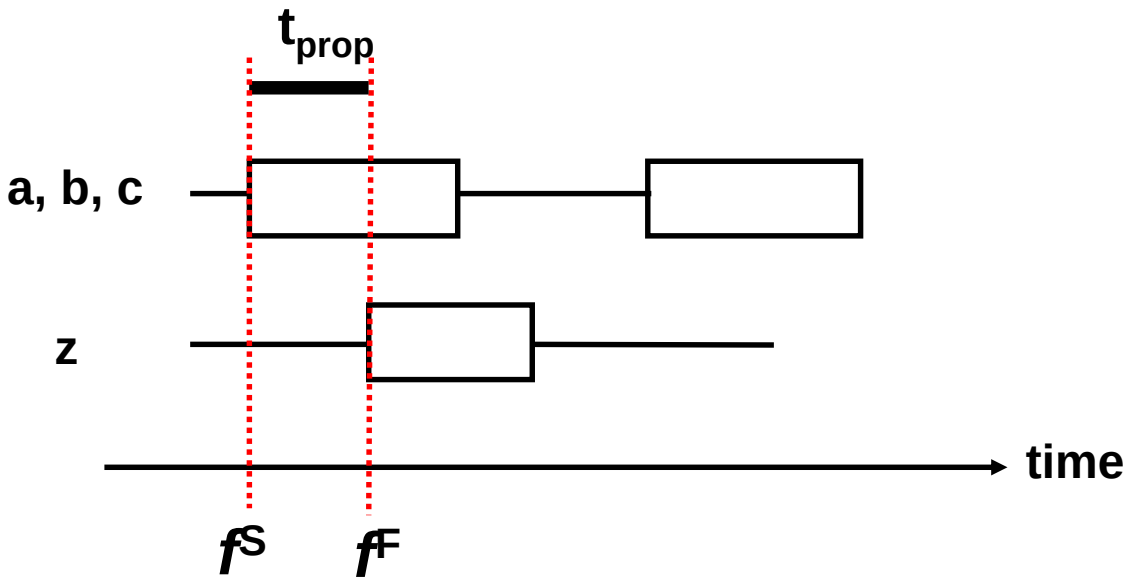
Timing (When)

Only if we guarantee to meet the timing requirements

... do the components guarantee to behave as intended.

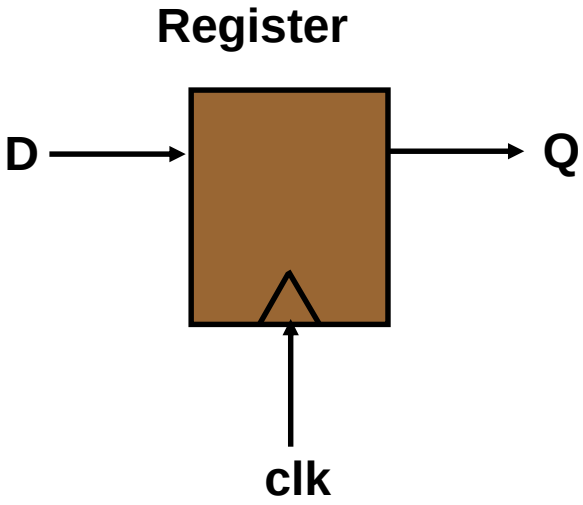
Combinational Logic Timing

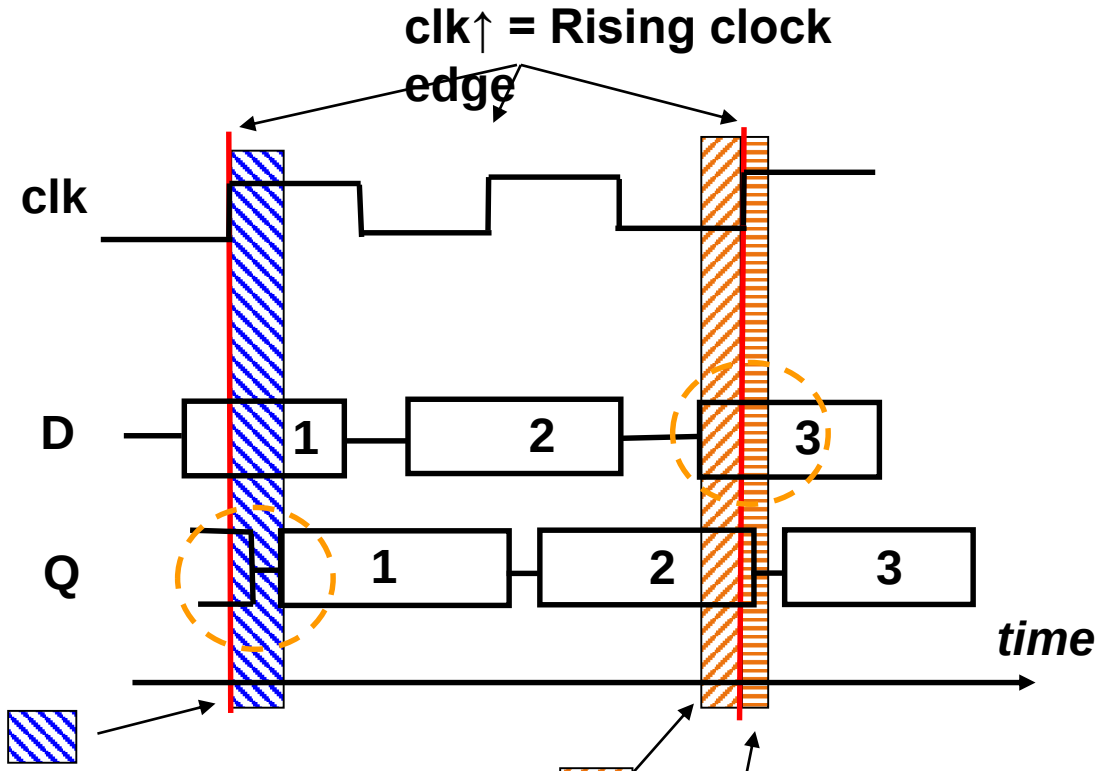




- **Propagation delay:**
  After presenting new inputs
  Worst case delay before producing correct output

Register timing





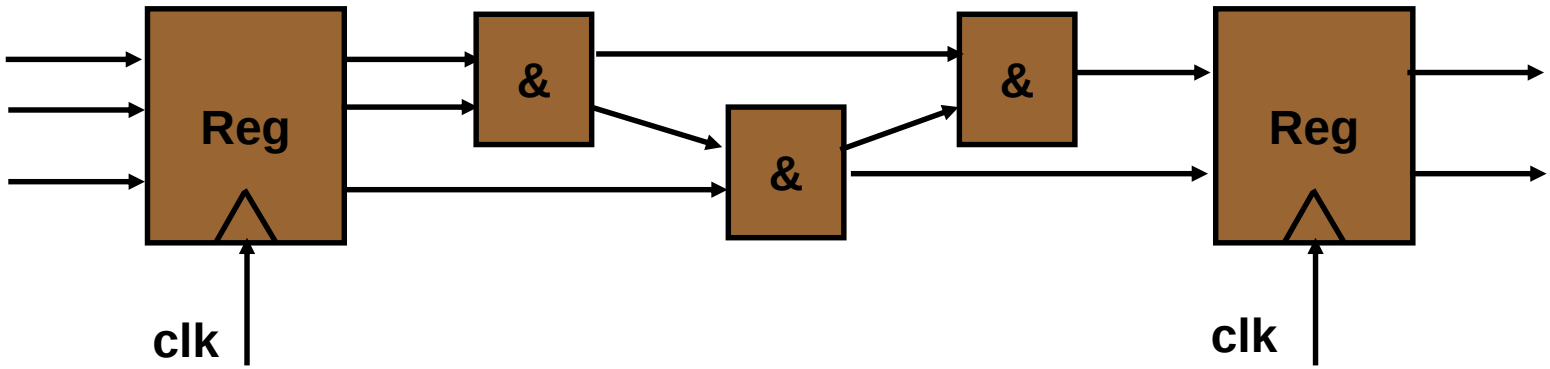
Propagation delay (clk_to_Q):
Worst case (maximum) delay after clk↑ before new output data is valid on Q.

Setup time:
Minimum time input must be stable before clk↑

Hold time:
Minimum time input must be stable after clk↑

Clock Frequency (RTL)

- What is the maximum clock frequency?



Register

Propagation delay:	Tckl-Q	250ps
Setup time:	Tsu	200ps
Hold time:	Th	100ps

AND-gate

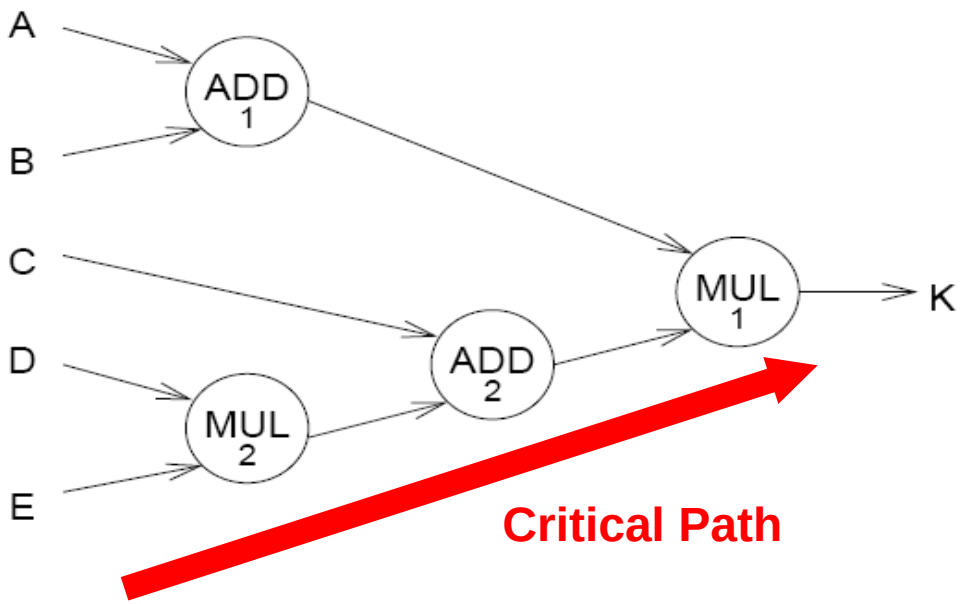
Propagation delay:	Tprop	250ps

250+250×3+200=1.2ns

***f*=833MHz**

Critical path

- …begin to explore the construction of digital systems with complex behavior
  - Example: K = (A $+_1$ B) $*_1$ (C $+_2$ D $*_2$ E)
- Combinational circuit:

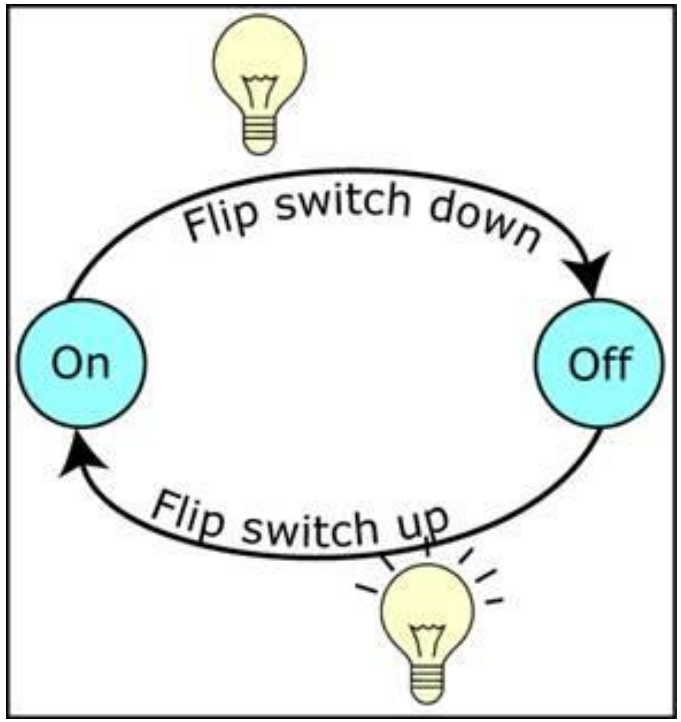


Outline

- FSM Overview
- FSM Representation
  - examples
- Moore vs. Mealy Machine
  - from circuits perspective

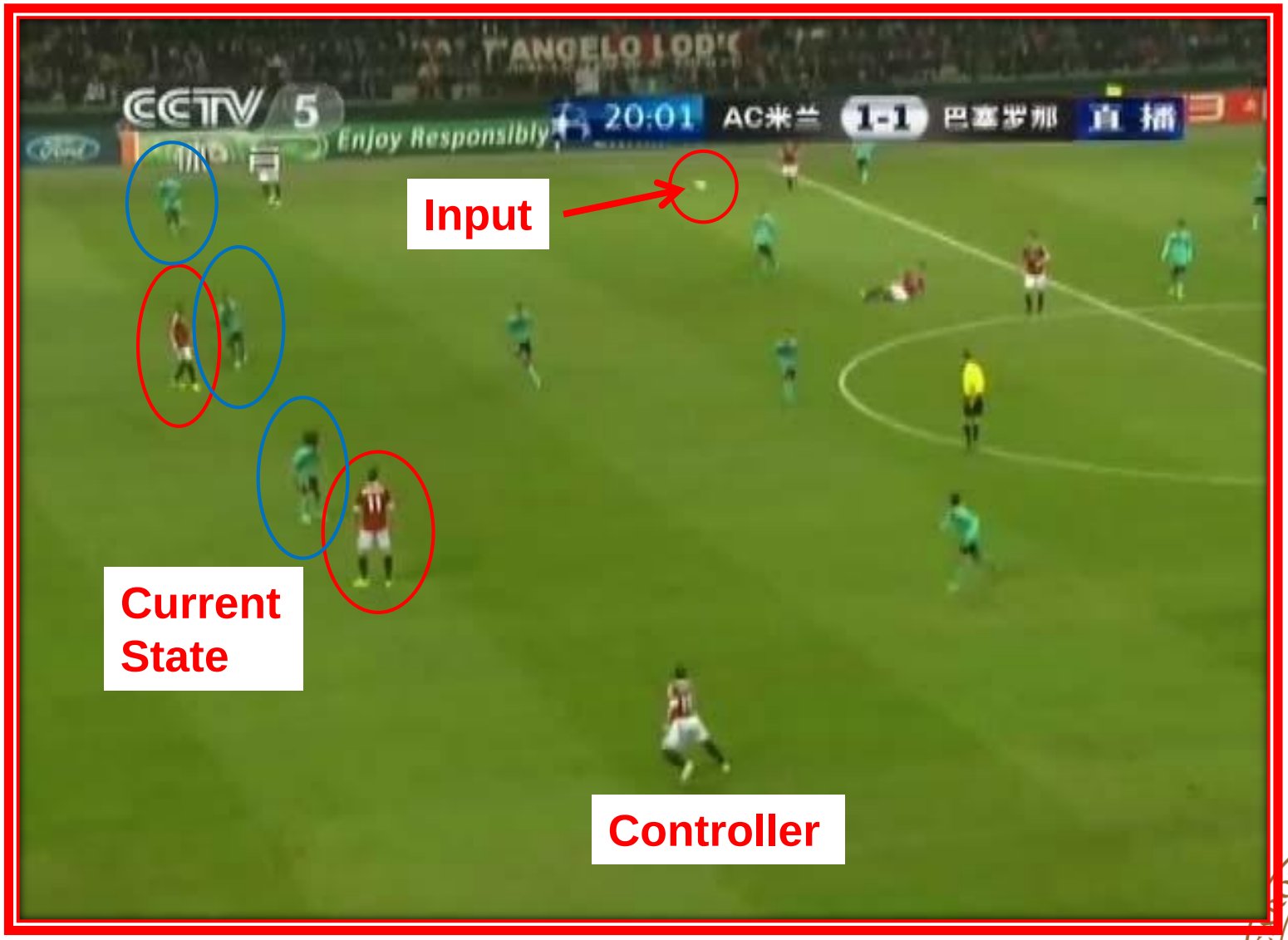
FSM Overview

- Models for representing sequential circuits
- Used mainly as a controller in a large system

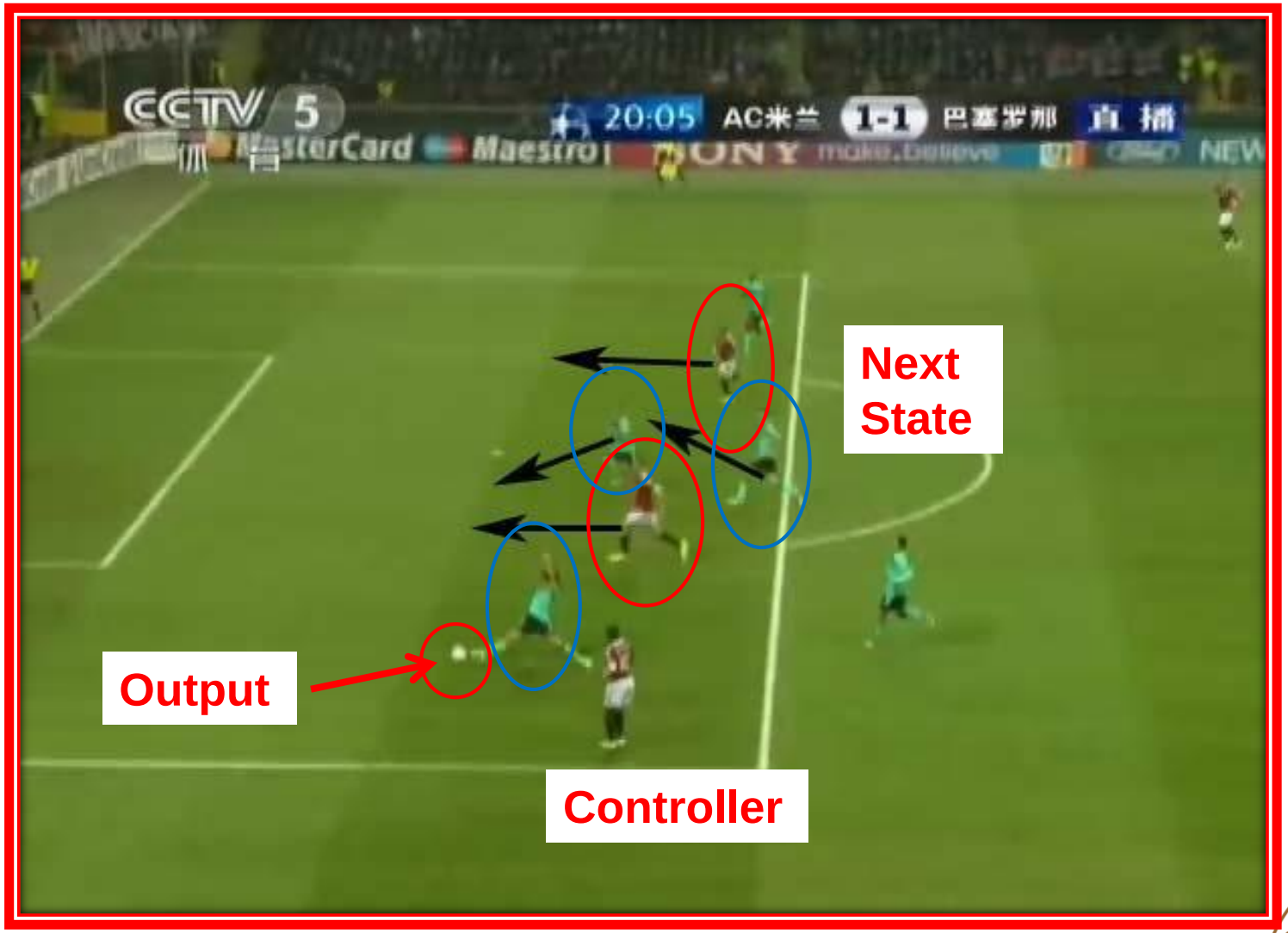


How does a controller work in a system?

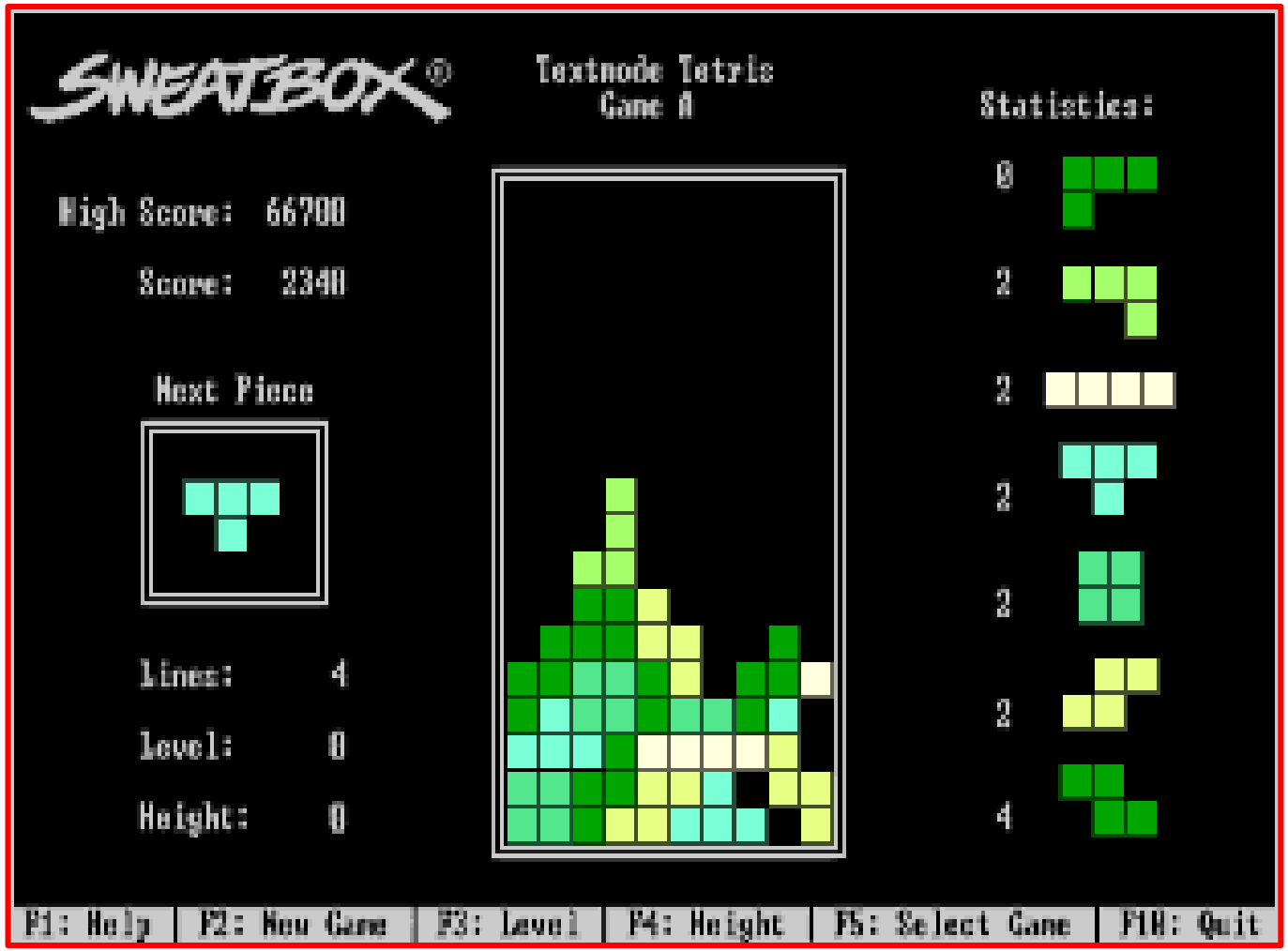
Controller


CCTV 5
Enjoy Responsibly
20:01
AC米兰
1-1
巴塞罗那
直播
Input
Current State
Controller

Controller

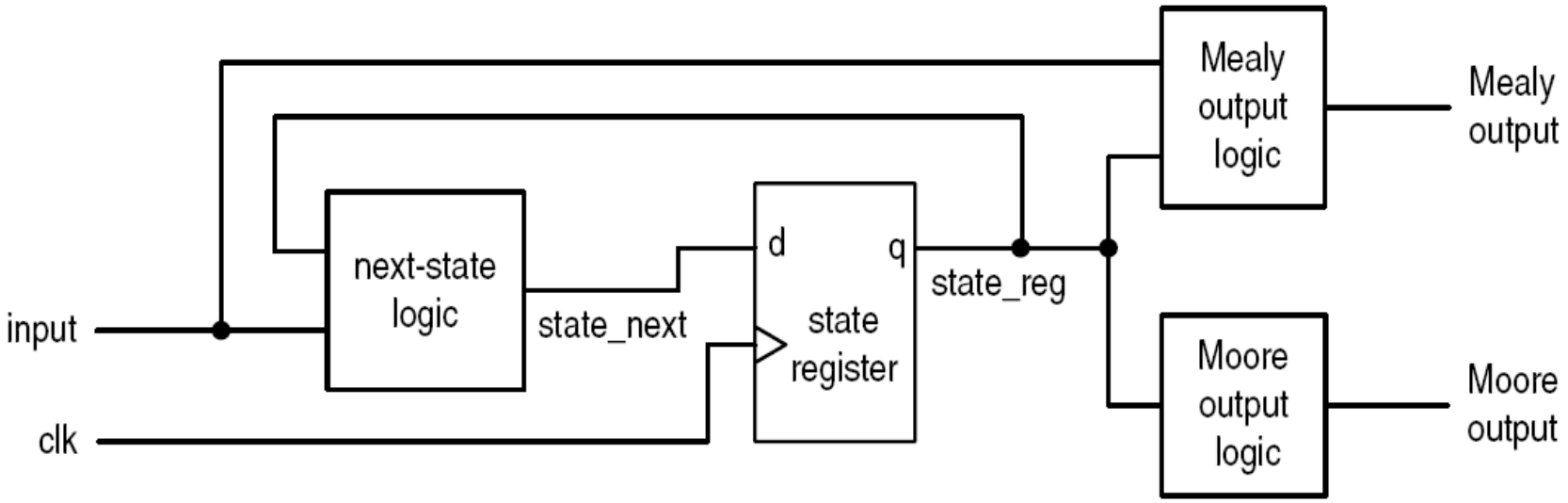


The model can be used in many places



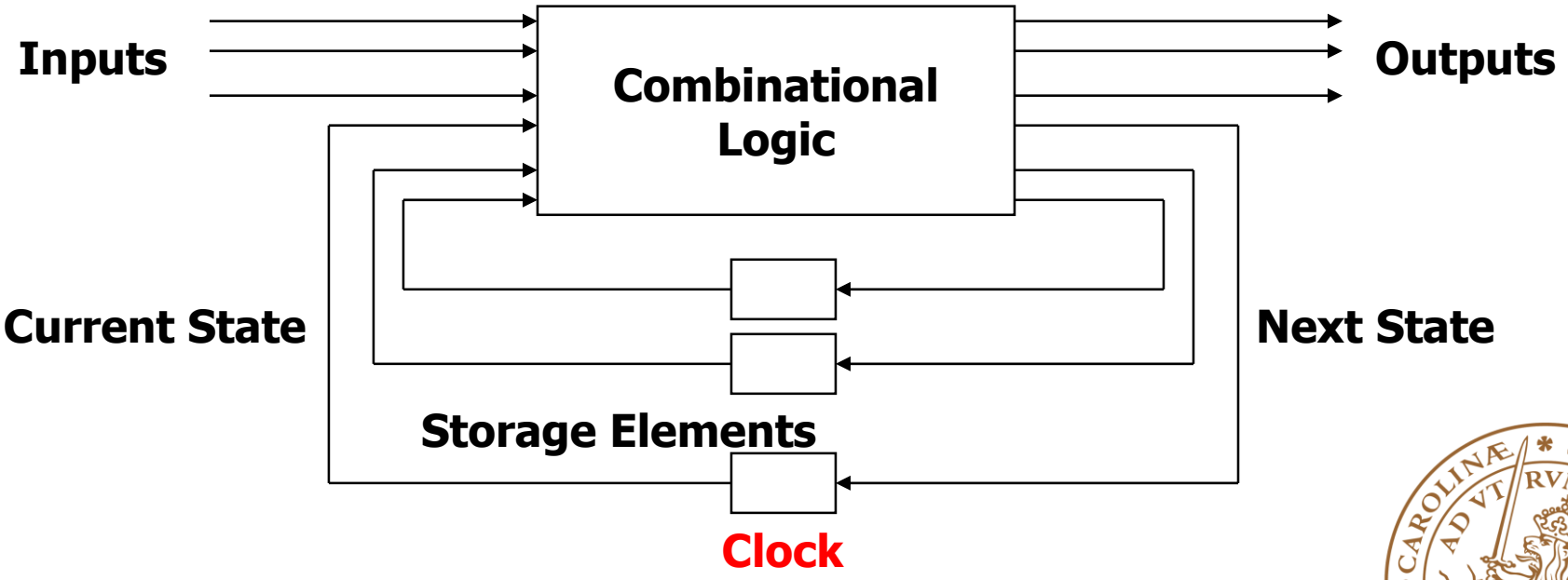
Abstraction of state elements

- A FSM consists of several states. Inputs into the machine are combined with the *current state* of the machine to determine the new state or *next state* of the machine.
- Depending on the state of the machine, outputs are generated based on either the state or the state and inputs of the machine.



Abstraction of state elements

- A FSM consists of several states. Inputs into the machine are combined with the *current state* of the machine to determine the new state or *next state* of the machine.
- Depending on the state of the machine, outputs are generated based on either the state or the state and inputs of the machine.
- Divide circuit into combinational logic and state (registers)



Outline

- FSM Overview
- **FSM Representation**
- Moore vs. Mealy Outputs
- Exercise

FSM Representation

- Can be represented using a state transition table which shows the *current state*, *input*, any *outputs*, and the *next state*.

Input / Current State	$Input_0$	$Input_1$	....	$Input_n$
$State_0$	Next State / Output		....	Next State / Output
$State_1$	....	....		....
....	....	....		....
$State_n$	....	....		....

FSM Representation

- **It can also be represented using a *state diagram* which has the same information as the state transition table.**

- **Mealy Output**

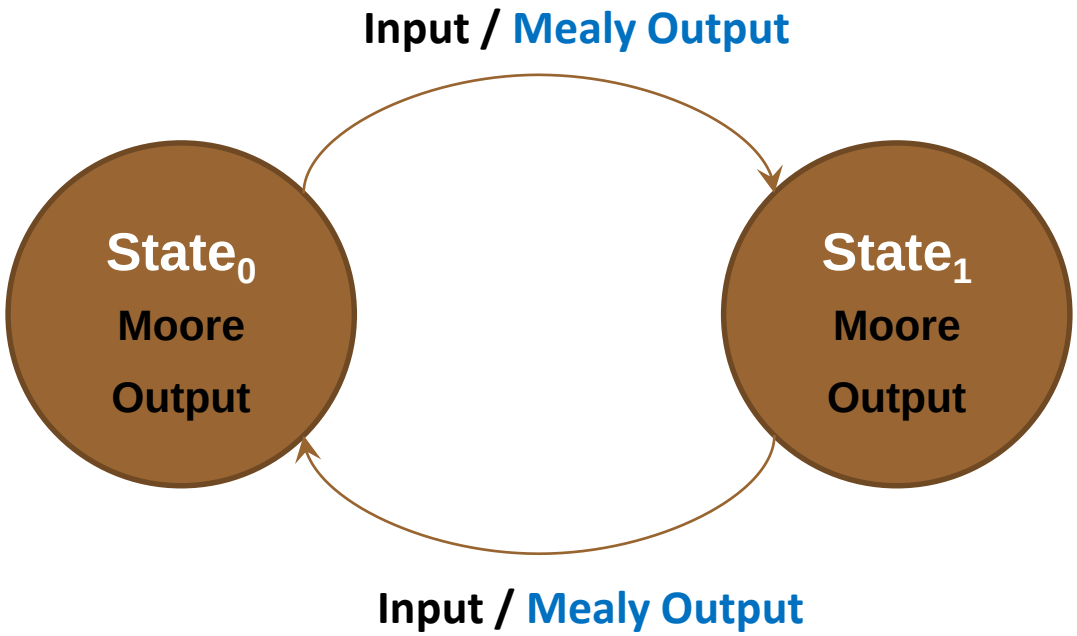
Outputs =F(*Inputs*, *Current state*)

Next state = F(*Inputs*, *Current state*)

- **Moore Output**

Outputs = F(*Current state*)

Next state = F(*Inputs*, *current state*)



Example 1: A mod-4 synchronous counter

- **Function: Counts from 0 to 3 and then repeats; Reset signal reset the counter to 0.**
- **It has a clock (*CLK*) and a *RESET* input.**
- **Outputs appear as a sequence of values of 2 bits (q1 q0)**
- **As the outputs are generated, a new state (s1 s0) is generated which takes on values of 00, 01, 10, and 11.**

State Transition Table of Mod-4 Counter

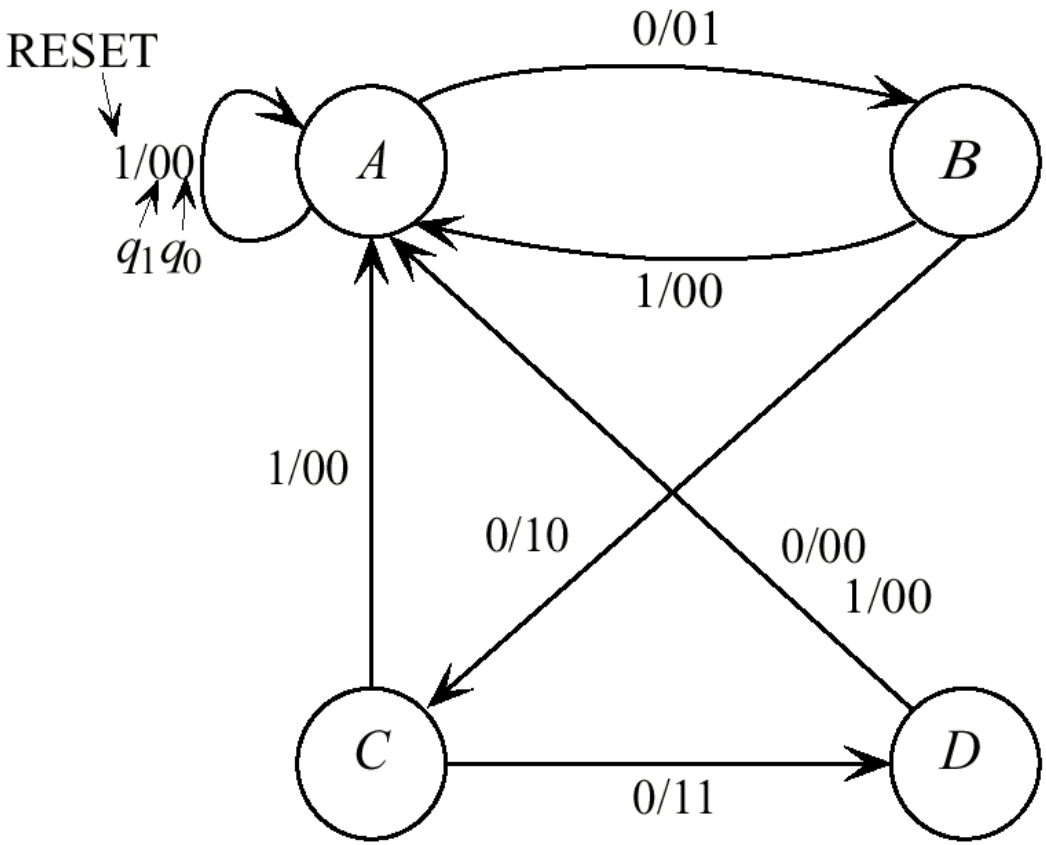
Present state (S_t) \ Input	*RESET* 0	*RESET* 1
A:00	01/01	00/00
B:01	10/10	00/00
C:10	11/11	00/00
D:11	00/00	00/00

State Next

Output

Clock?

State Transition Diagram for the Mod-4 Counter



Use meaningful names for states

Outline

- FSM Overview
- FSM Representation
- **Moore vs. Mealy Outputs**
- Exercise

Mealy and Moore FSM

A Method for Synthesizing Sequential Circuits

By GEORGE H. MEALY

(Manuscript received May 6, 1955)

The theoretical basis of sequential circuit synthesis is developed, with particular reference to the work of D. A. Huffman and E. F. Moore. A new method of synthesis is developed which emphasizes formal procedures rather than the more familiar intuitive ones. Familiarity is assumed with the use of switching algebra in the synthesis of combinational circuits.

CONTENTS

1. INTRODUCTION

GEDANKEN-EXPERIMENTS ON SEQUENTIAL MACHINES

Edward F. Moore

INTRODUCTION

This paper is concerned with finite automata[1] from the experimental point of view. This does not mean that it reports the results of any experimentation on actual physical models, but rather it is concerned with what kinds of conclusions about the internal conditions of a finite machine it is possible to draw from external experiments. To emphasize the conceptual nature of these experiments, the word "gedanken-experiments" has been borrowed from the physicists for the title.

The sequential machines considered have a finite number of states, a finite number of possible input symbols, and a finite number of possible output symbols. The behavior of these machines is strictly deterministic (i.e., no random elements are permitted in the machines) in that the present state of a machine depends only on its previous input and previous state, and the present output depends only on the present state.

The point of view of this paper might also be extended to probabilistic machines (such as the noisy discrete channel of communication theory[2]), but this will not be attempted here.

EXPERIMENTS

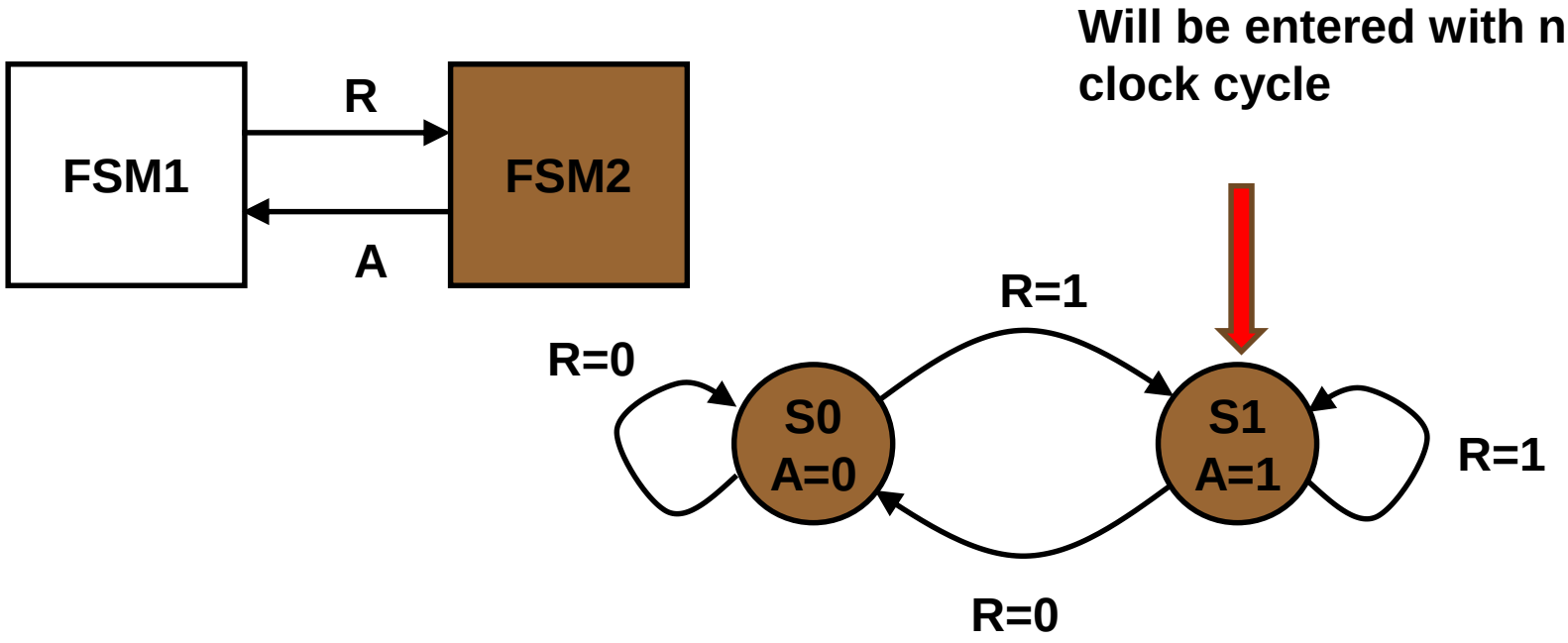
There will be two kinds of experiments considered in this paper. The first of these, called a simple experiment, is depicted in Figure 1.

[1] The term "finite" is used to distinguish these automata from Turing machines [considered in Turing's "On Computable Numbers, with an Application to the Entscheidungsproblem", Proc. Lond. Math. Soc., (1936) Vol. 24, pp. 230-265] which have an infinite tape, permitting them to have more complicated behavior than these automata.

[2] Defined in Shannon's "A Mathematical Theory of Communication", B.S.T.J. Vol. 27, p. 406.

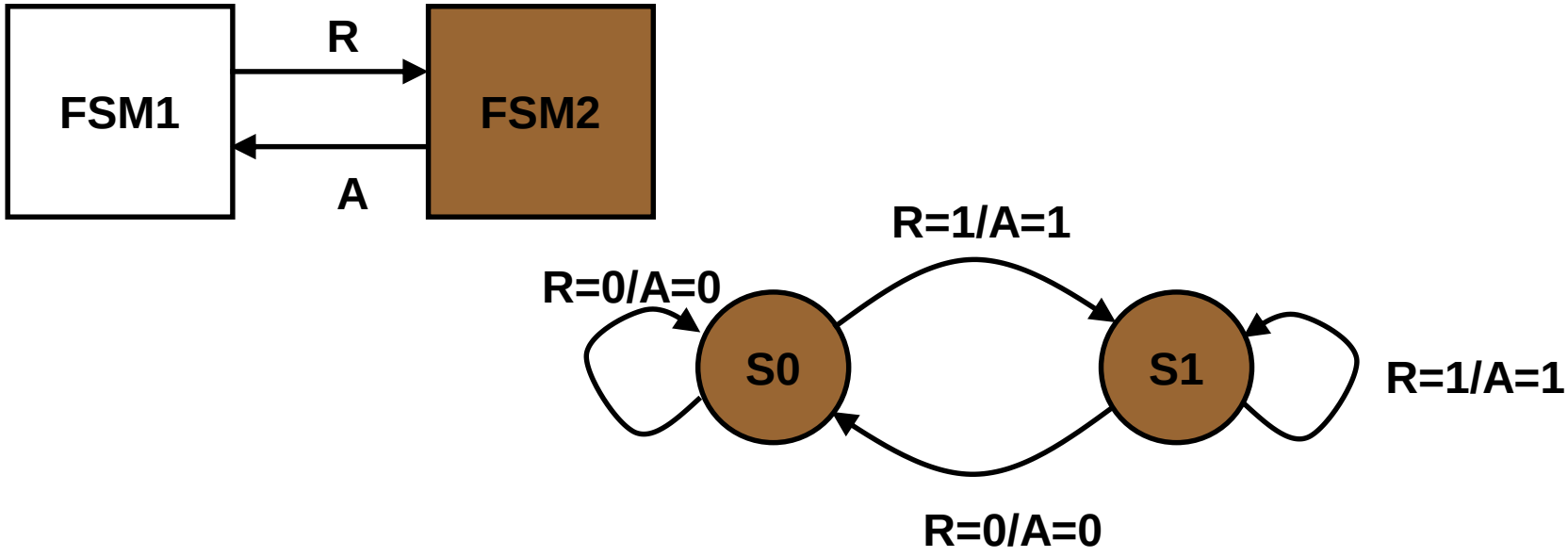
129

Output Timing: Moore



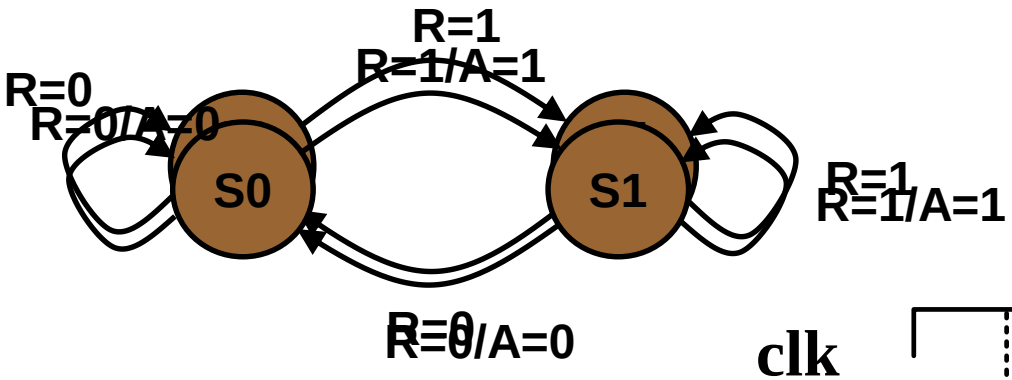
- … a Moore machine is not able to produce A->1 until the next clock when it enters s1

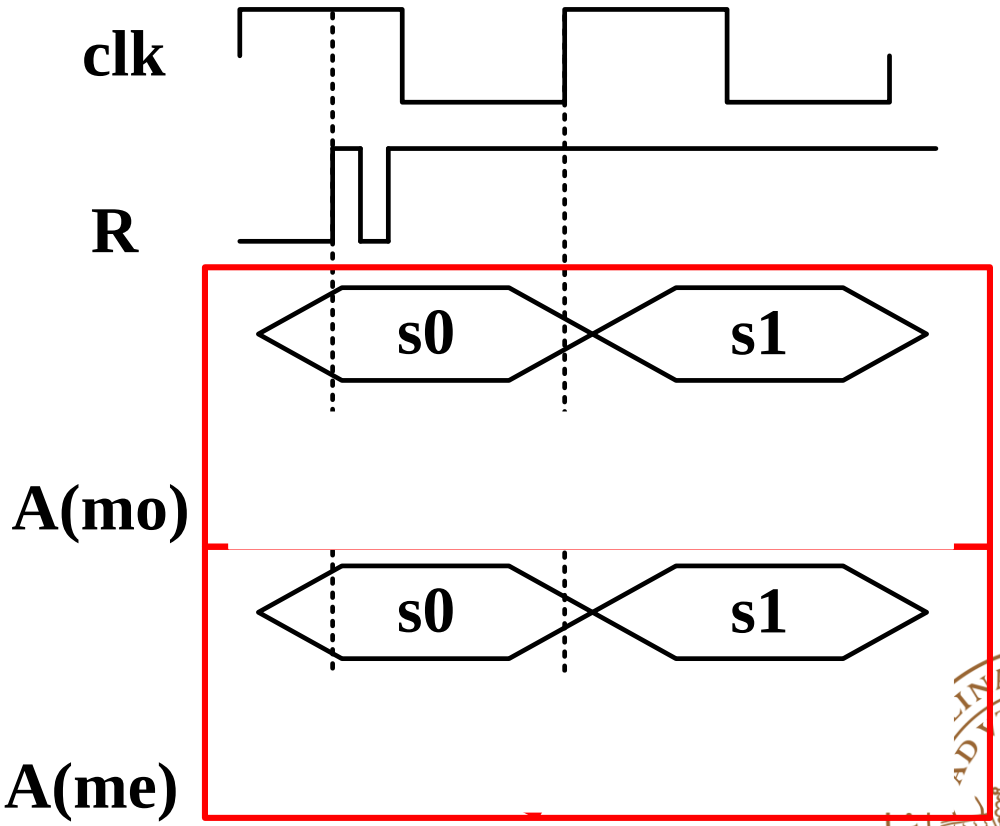
Output Timing: Mealy



- When in s0, a Mealy machine may produce A->1 immediately in response to R->1

Output Timing: Moore and Mealy





Moore vs. Mealy

- Detecting a pair of “1s” or “0s” and output “1”

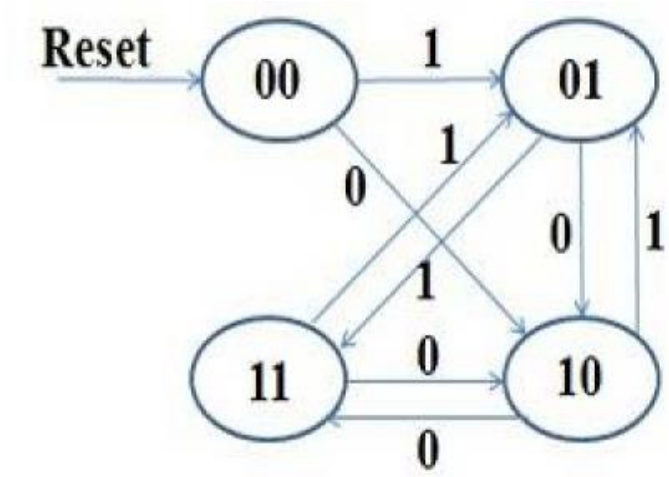


Figure 3State Diagram of Moore Machine

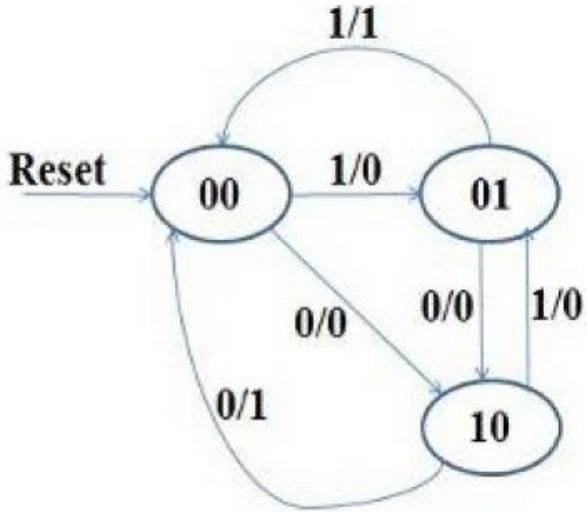


Figure 4State Diagram of Mealy Machine

Moore vs. Mealy (summary)

- **A Moore machine produces glitch free outputs**
  - Output change at the clock edge only
- **A Moore machine produces outputs depending only on states, and this may allow using a higher-frequency clock**
  - Less gate delay for the output combinational logic
- **A Mealy machine can be specified using less states**
  - Because it is capable of producing different outputs in a given state, (nm) possible outputs v.s. (n)
- **A Mealy machine can be faster**
  - Because an output may be produced immediately instead of at the next clock tick

Suggestion: do NOT mix Mealy and Moore in one design (before getting experienced)

?


SIGILL
AD VTRVMQVE
CAROLINÆ
ORVM