CSSE2010/CSSE7201
Lecture 8

Sequential Circuits 3
State Machines

A

School of Information Technology and Electrical Engineering
The University of Queensland

Outline

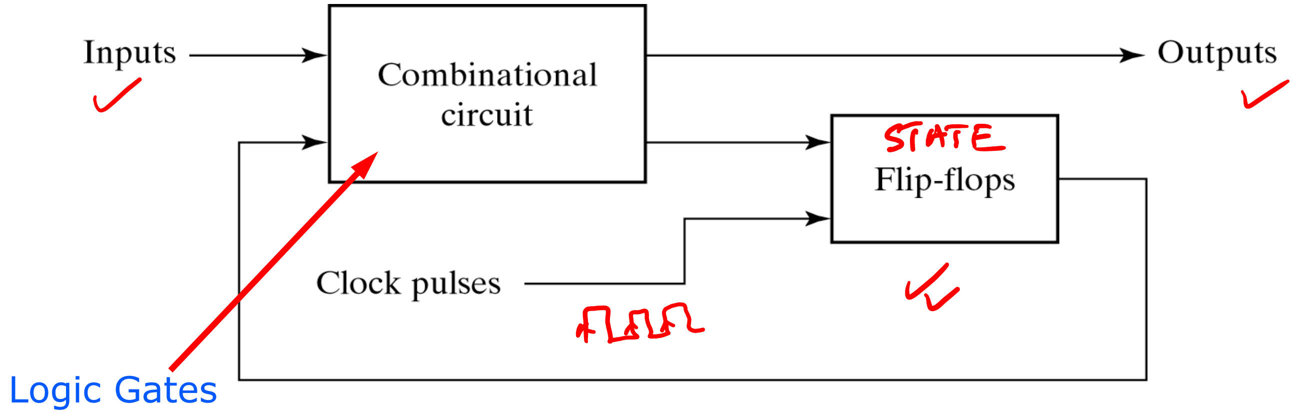
- Admin
- State machines
  - State diagrams
  - State tables
  - State encoding

Admin

- Quiz 3 is due on Friday 4pm this week

Sequential Circuits

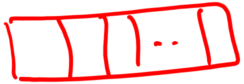
- **State** = value stored in flip-flops
- Output depends on input and state
- Next state depends on inputs and state



[Figure from Mano, "Digital Design", 3rd edition]

State Machines

N-FFs

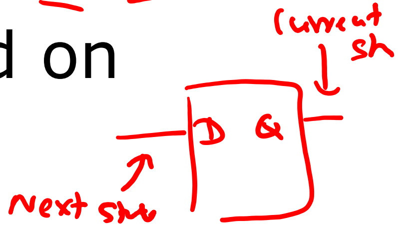


- Sequential circuits can also be called
  - state machines
  - finite state machines (FSMs)
- State machine has
  - Finite number of possible states
  - Only one **current state**
  - Can **transition** to other states based on inputs and current state

N-FFs

2^N possible states

FSM with N-FFs can have $\leq 2^N$ states



State Machines

- The states can be defined based on the problem:

E.g. a vending machine accepting 5 cents and 10 cents coins to dispense a candy when it receives 15 cents in total. What can be the different states?

State ≡ Money received so far

1) 0 5 10 15

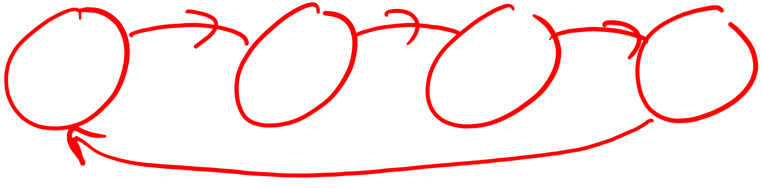
$2^2 = 4$

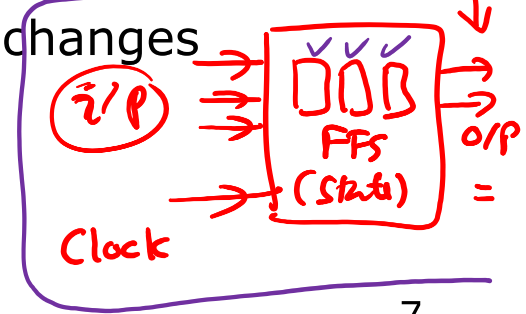
Types of State Machines

FSM

- Two types
  - Mealy machines
    - Outputs depend on current state and inputs
  - Moore machines
    - Outputs depend only on current state (flip-flop values)
      - Outputs can only change when state changes

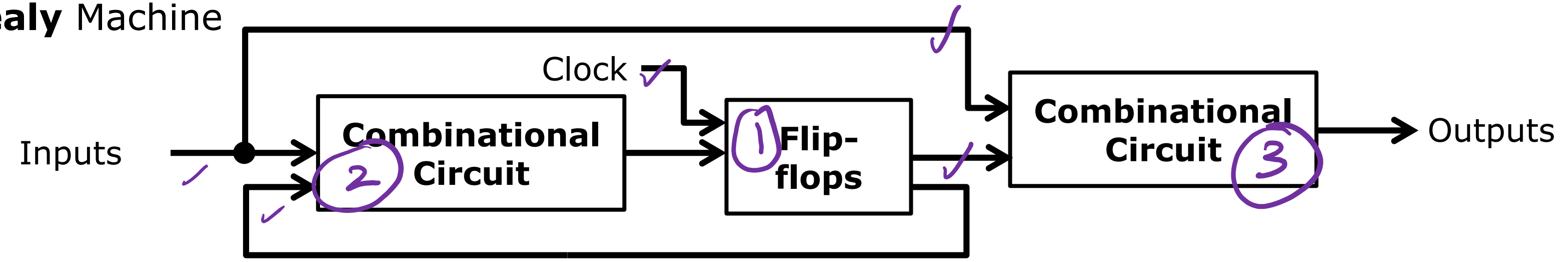
discuss only this



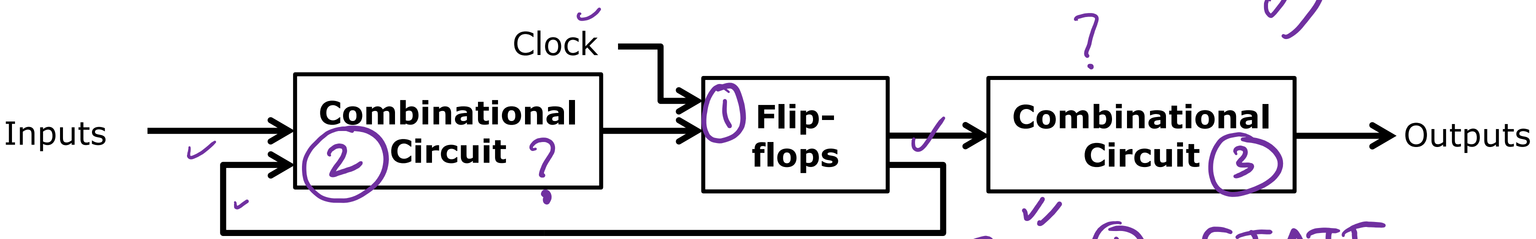


Moore vs Mealy Machine

- **Mealy** Machine



- **Moore** Machine – special case of a Mealy Machine



we only discuss this.

① STATE
② State transition logic
③ output logic

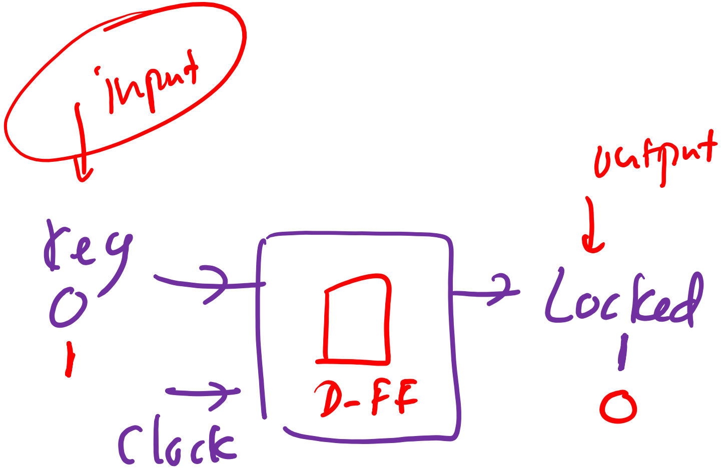
- We'll stick to Moore machines in this course

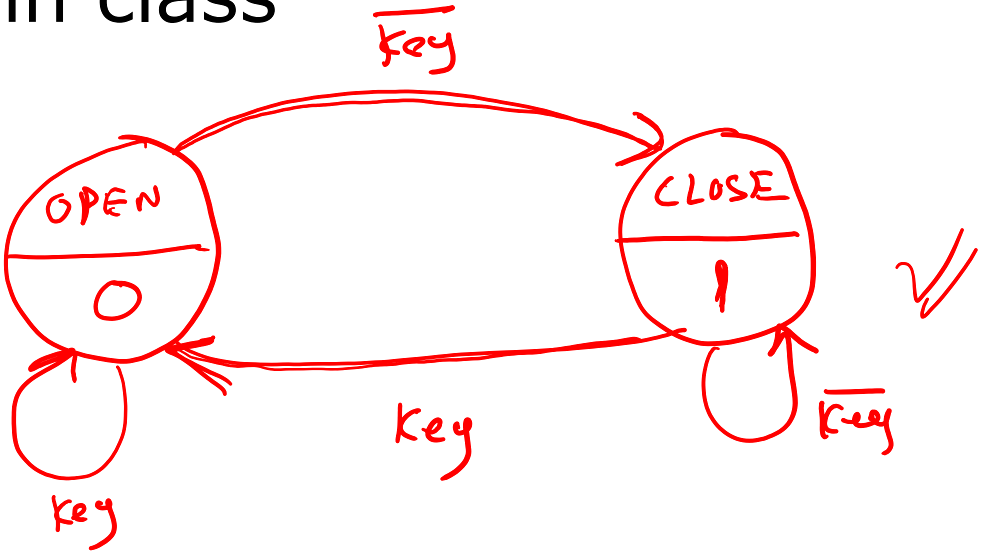
State Diagram

2

- To be illustrated in class

2 State lock

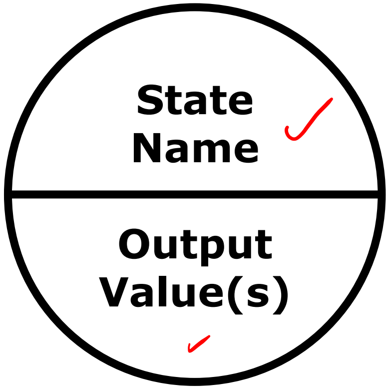




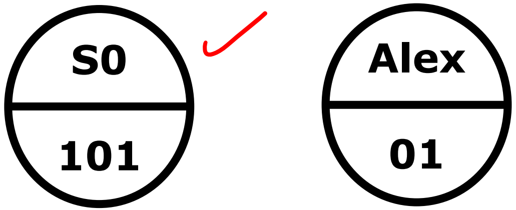
O/P inside the state → "Moore"

State Diagram (cont.)

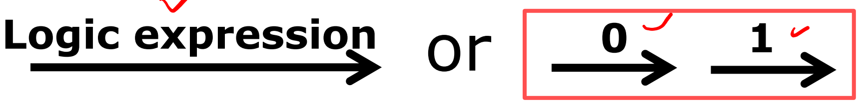
- Notation
  - State



Examples:

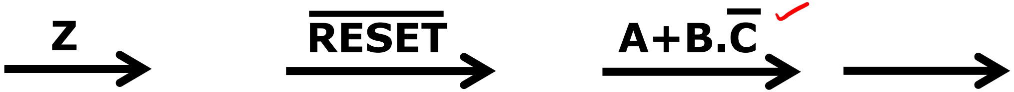


  - Transition



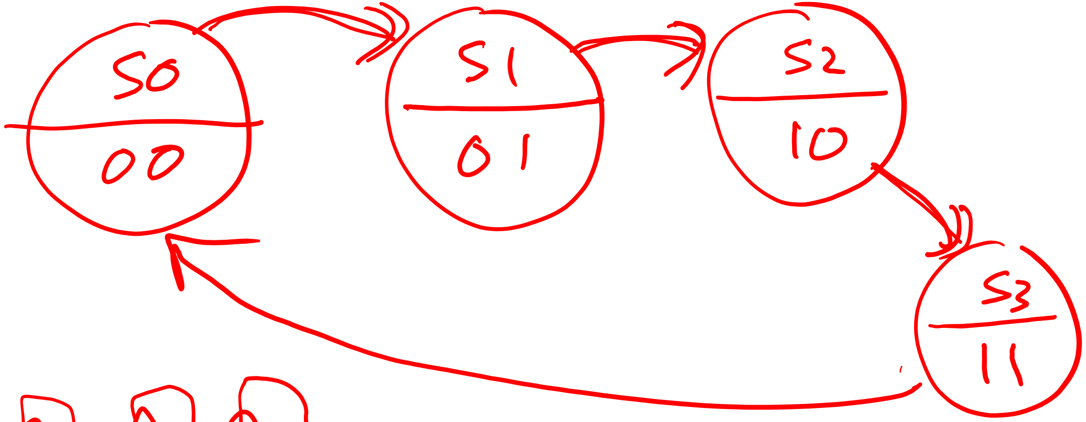
Special case for only one input

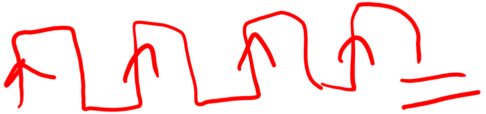
    - If expression is true, state transition is made
    - No label means "true" – i.e. always transition
    - Examples

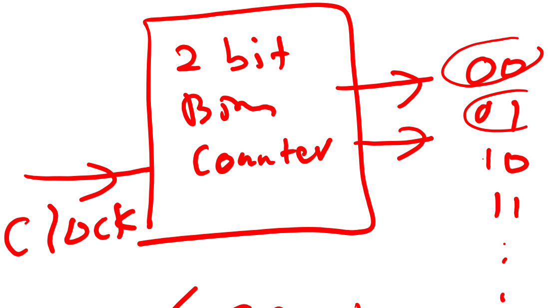


Example – to be worked out in class

- Binary counter with 2-bit output



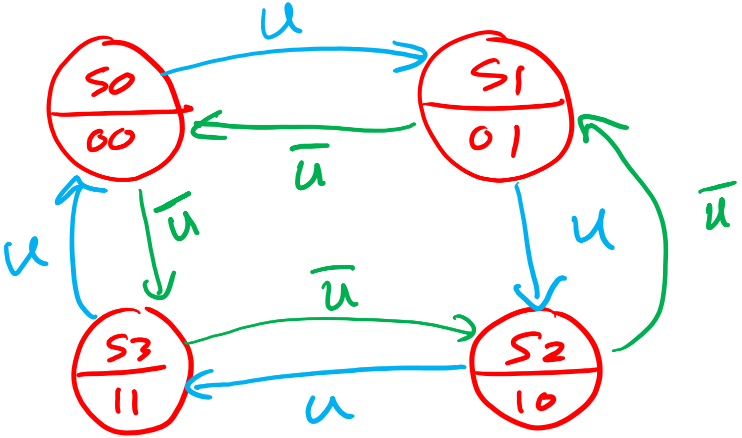




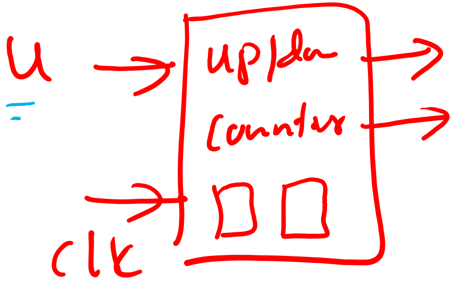
00
01
10
11
00
01
10
11

Example

- Binary up/down counter with 2-bit output
  - One input: U
    - 1 means count up
    - 0 means count down



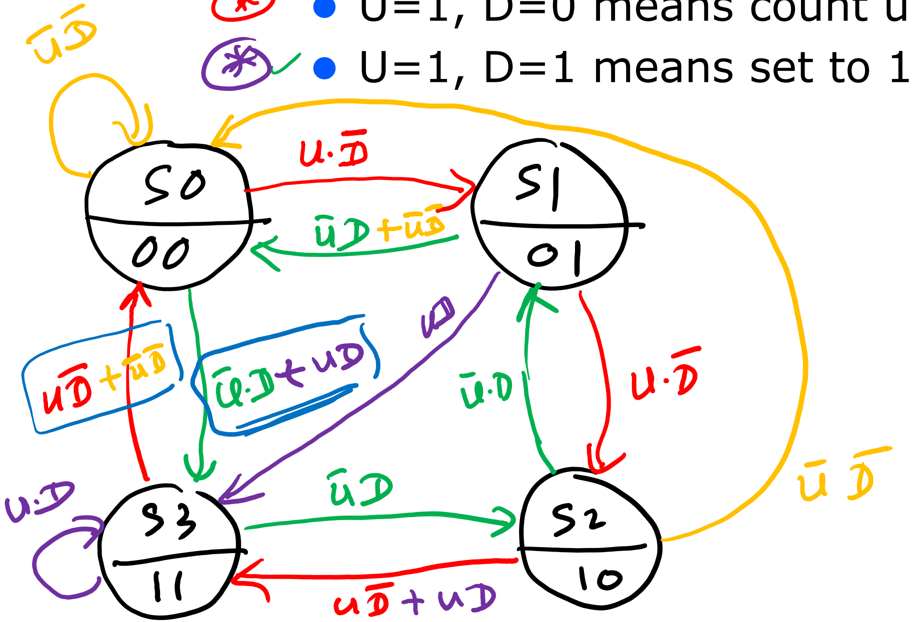
"complete"

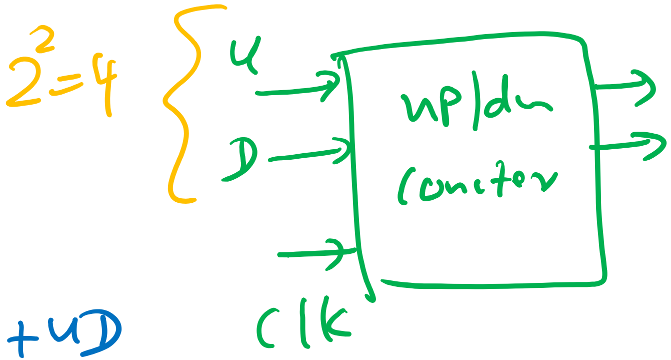


00	11
01	10
10	01
11	00
U=1	U=0

Example

- Binary up/down counter with 2-bit output
  - Two Inputs: U,D
    - U=1, D=0 means count up
    - U=1, D=1 means set to 11
    - U=0, D=1 means count down
    - U=0, D=0 means reset to 00





$\bar{U}D+UD$

$D(U+\bar{U})=D$

- Binary up/down counter with 2-bit output
  - U=1, D=0 means count up
  - U=0, D=1 means count down
  - U=1, D=1 means set to 11
  - U=0, D=0 means reset to 00

Completeness

- Each possible combination of inputs should be addressed exactly once for each state
  - i.e. transition arrows from each state must encompass all possibilities (exactly once)
- Example:

UD	
00	$\bar{U}\bar{D}$
01	$\bar{U}D$
10	$U\bar{D}$
11	UD

3 State Table

u=0
u=1

- State diagrams can also be represented in a state table
- Example - binary up/down counter with 2-bit output and single input U (1 means up, 0 means down)
- One dimensional state table:

1D

Current State *	Input U	Next State	Outputs	
			Q1	Q0
S0	0	S3	0	0
S0	1	S1	0	0
S1	0	S0	0	1
S1	1	S2 ~~S1~~	0	1
S2	0	S1	1	0
S2	1	S3	1	0
S3	0	S2	1	1
S3	1	S0	1	1

Every combination of state and inputs

00
01
10
11

CSSE2010

3 State Table (cont.)

- Two-dimensional state table
- Same example *

Every combination of inputs

2D Format

Current State	Next State		Outputs	
	$\overline{U}$ ✓	U ✓	Q1	Q0
S0	S3	S1	0	0
S1	S0	S2	0	1
10 S2	S1	S3	1	0
S3	S2	S0	1	1

11

One row per state

4 State encoding

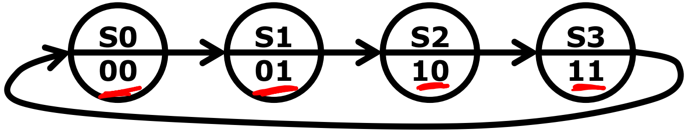
S0
S1

- Must encode each state into flip-flop values
- Choose
  - Number of flip-flops
  - Bit patterns that represent each state
- Ideally
  - Choose state encoding to make combinational logic simple, for both
    - ✓ Output logic ✓
    - ✓ Next state logic ✓ ✓

Binary encoding ✓
Gray code encoding
One-hot encoding

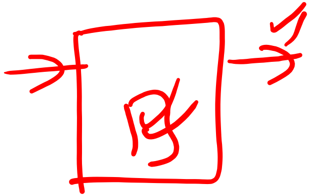
Example

- Binary counter with 2-bit output



State	(encoding) Bit pattern	One hot enc
S0	00	0001
S1	01	0010
S2	10	0100
S3	11	1000

4 FFs
State transit Logic is simple



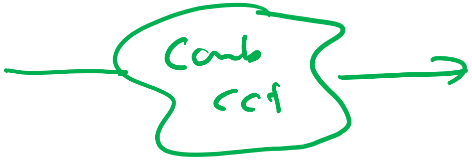
Binary encoding ⇒ output logic is simple.

One-hot coding

- Use one flip-flop per state
- Only one flip-flop has 1 at any time
- Example – binary counter with 2-bit output

State	Q_3	Q_2	Q_1	Q_0
S_0	0	0	0	1
S_1	0	0	1	0
S_2	0	1	0	0
S_3	1	0	0	0

0
1
2
3



Sequence Detector Example

- Design a state machine which detects the pattern 101 in an incoming bit stream and outputs a 1 whenever it detects this pattern
  - Output is 1 for one clock cycle after the third digit is clocked in
- Example:
  - Input: 00<u>101</u>1<u>10</u><u>1</u>0100
  - Output: 0000100010100

1 ✓
2 ✓
3 ✓ → state table
$s_0, s_1, \ldots$
4 ✓
5
6

1 → define the states

2 → Draw the state diagram (Moore)

3 → State table < 1D, 2D } still have state names e.g S_0, S_1.

4 → State encoding → i.e binary, one-hot

5 → State table again with encoding.

6 → get the expressions for State tran & outputs & draw the logic diagram

⇒ watch the example for steps 5 & 6.

A pre-recorded example will be provided on Blackboard