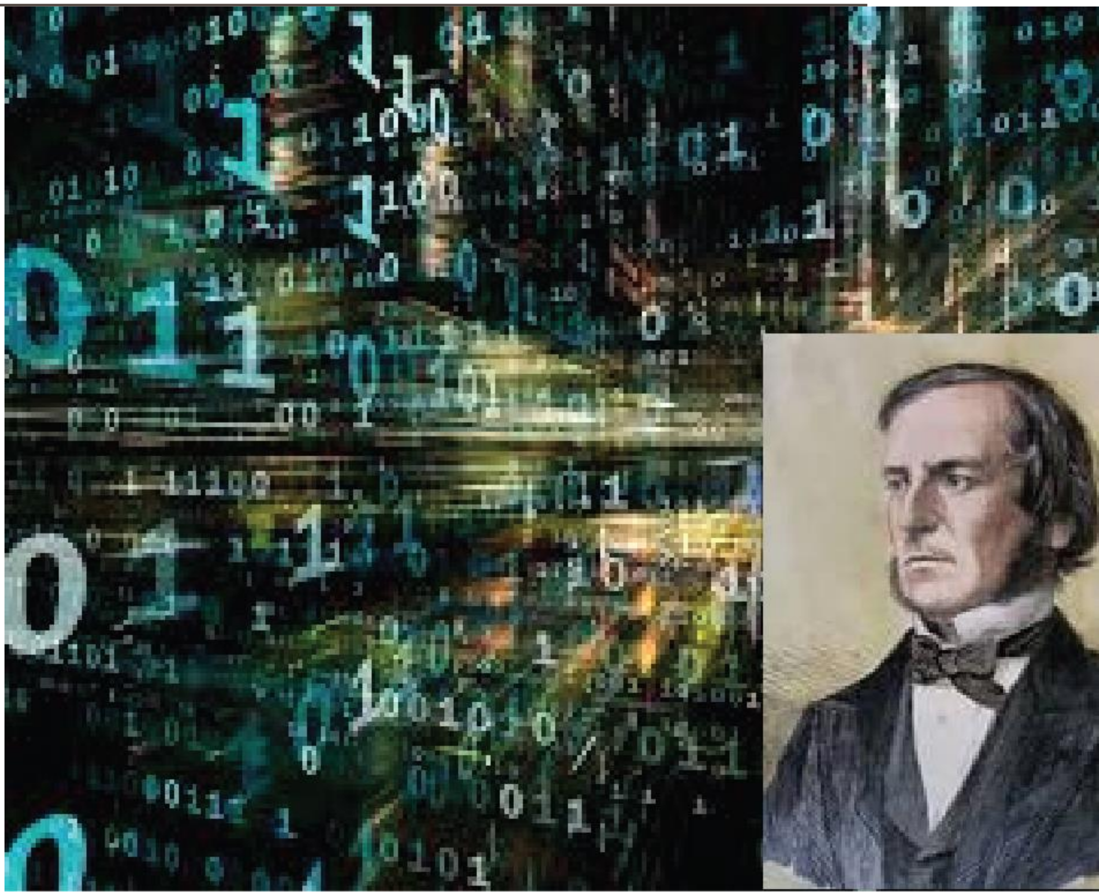
DIGITAL CIRCUITS

Week-13, Lecture-2
Sequential Circuits

Sneh Saurabh
9th November, 2018



Digital Circuits: Announcements/Revision

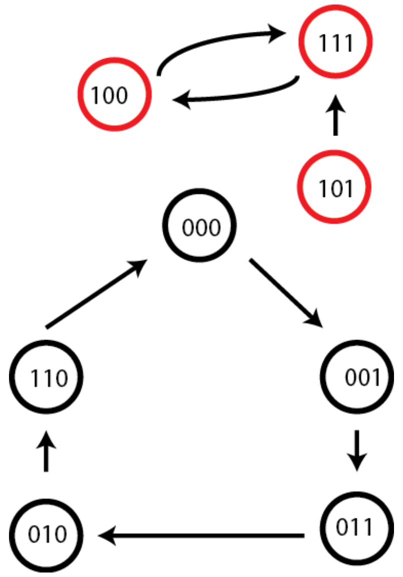
Sequential Circuits

Design

Sequential Circuit Design: Optimized Implementation

- With the following implementation, what would be the next state for the invalid states:
  - $T_A = P_C' P_B$
  - $T_B = P_A + P_B' P_C$
  - $T_C = P_B P_C + P_B' P_C'$

First find T_A, T_B and T_C and then N_A, N_B and N_C



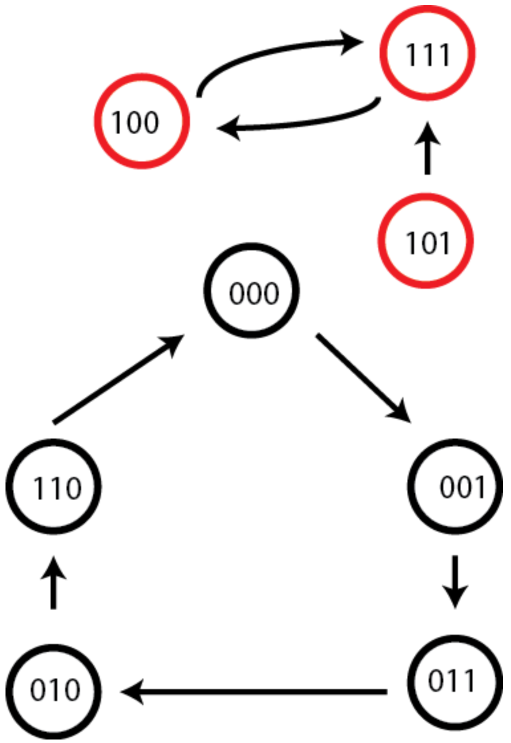
If the counter enters an invalid state, it remains in the invalid state for ever!

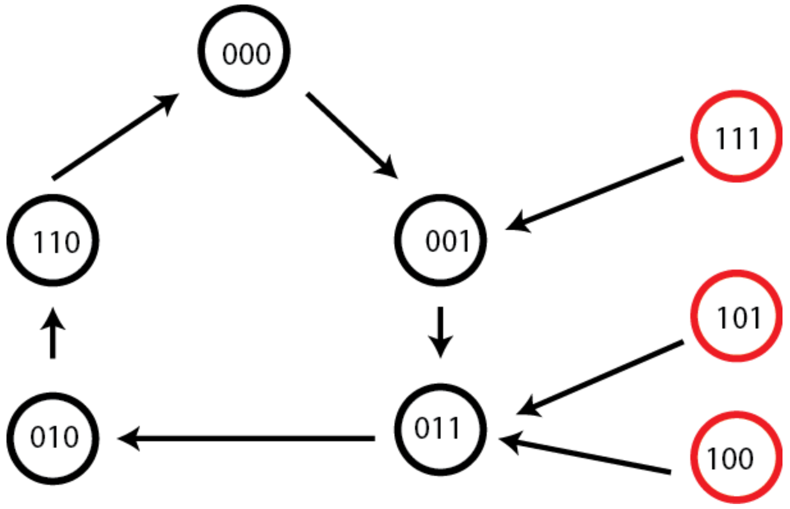
How to avoid this?

Present State			Next State			Flip-flop inputs		
P_A	P_B	P_C	N_A	N_B	N_C	T_A	T_B	T_C
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	0	1	1	0	1	0	0
0	1	1	0	1	0	0	0	1
1	**0**	**0**	**1**	**1**	**1**	**0**	**1**	**1**
1	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**0**
1	1	0	0	0	0	1	1	0
1	**1**	**1**	**1**	**0**	**0**	**0**	**1**	**1**

Sequential Circuit Design: Avoiding being stuck

- Just use reset
- Design such that FSM eventually transitions to a valid state
  - ➢ May limit exploiting don't care states





Sequential Circuit Design: State Reduction

- Reduction in the number of flip-flops

- Involves reduction in the number of states in the state table without changing the input/output behaviour

- Resource required in realizing next state logic and output logic: effect of reducing the number of flip-flops unpredictable

- Since m flip-flops realize 2^m states, reduction in the number of states may not lead to reduction in flip-flops

- Example: number of states reduce from 14 to 9, still 4 flip-flops will be required

- Example: number of states reduce from 10 to 6, number of flip-flops reduced from 4 to 3

Sequential Circuit Design: State Reduction Algorithm

- **Algorithm**: Two states are said to be equivalent if, for each member of the set of inputs, they give exactly the same output and send the circuit either to the same state or to an equivalent state.

- When two states are equivalent, one of them can be removed without altering the input–output relationships.

Problem 1:

Reduce the number of states in the state table shown alongside

- Find two present states that go to the same next state and have the same output for both input combinations

- e and g are equivalent states

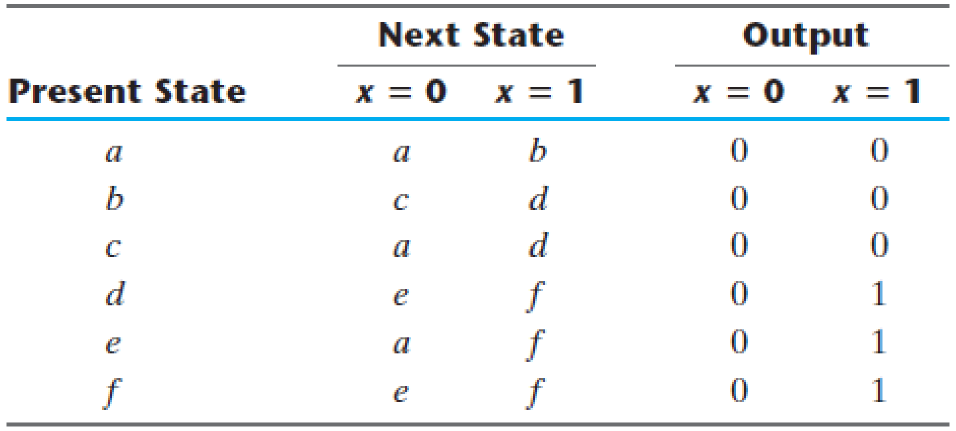
	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

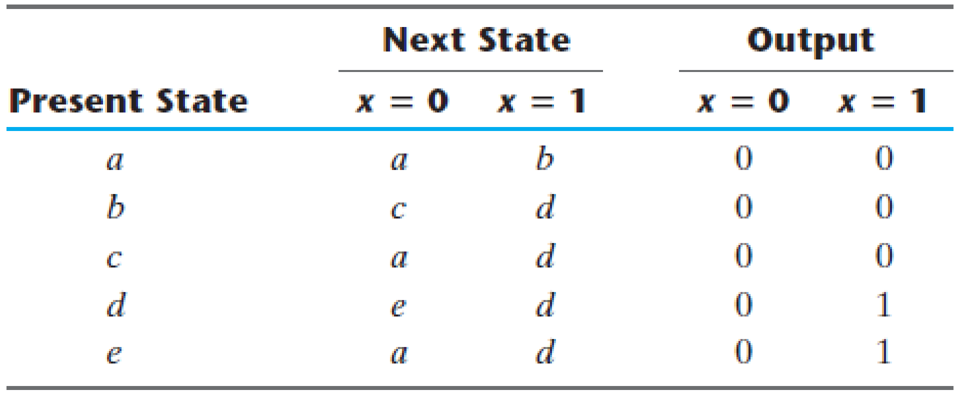
Sequential Circuit Design: State Reduction Algorithm

- Find two present states that go to the same next state and have the same output for both input combinations
- d and f are equivalent states

- Number of states reduced from 7 to 5

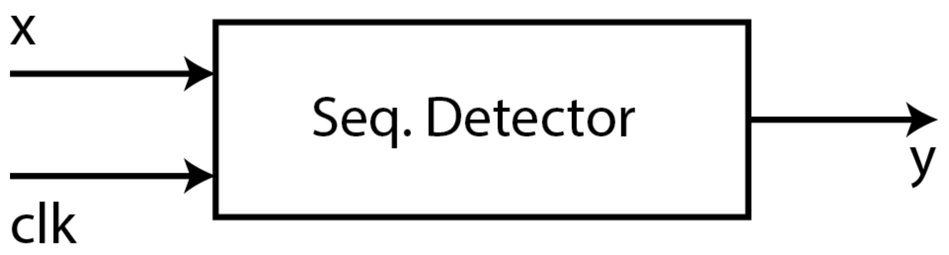


	Next State		Output	
Present State	**$x = 0$**	**$x = 1$**	**$x = 0$**	**$x = 1$**
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1



	Next State		Output	
Present State	**$x = 0$**	**$x = 1$**	**$x = 0$**	**$x = 1$**
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

Sequential Circuit Design: Sequence Detector



- x is input bits coming in sequence
- $x = 10111000111010111 \ldots$
- y is output indicating whether a given (target) sequence was detected
- $y = 00000000010001 \ldots$
- y bit is 1 when a given (target) sequence is detected

Sequence Detector: Overlapping vs. Non-overlapping

Detection of target sequence can be defined in two ways:

1. **Overlapping sequence**: the final bits of one sequence can be the start of another sequence
2. **Non-overlapping sequence**: two matched sequences have all bits different

Example: Given sequence is 11011

Assume x is 11011011011.

What would be the output for a) overlapping sequence b) non-overlapping sequence?

Answer:

a) If it is overlapping: y is 00001001001

b) If it is non-overlapping: 00001000001

Sequence detector can be implemented as either Moore or Mealy FSM

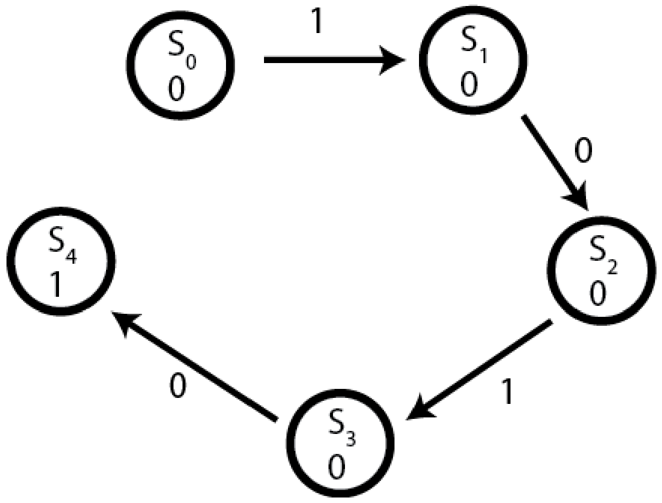
Sequence Detector: Non-overlapping

Problem 2: Draw the state diagram for an FSM that detects a non-overlapping sequence **1010**

a) First draw the transitions that correspond to correct sequence

b) Draw the transitions that break the sequence

c) Draw transition from the final state



Define states based on bits that have matched (take the maximal length):

S_0 = \`0′

S_1 = \`1′

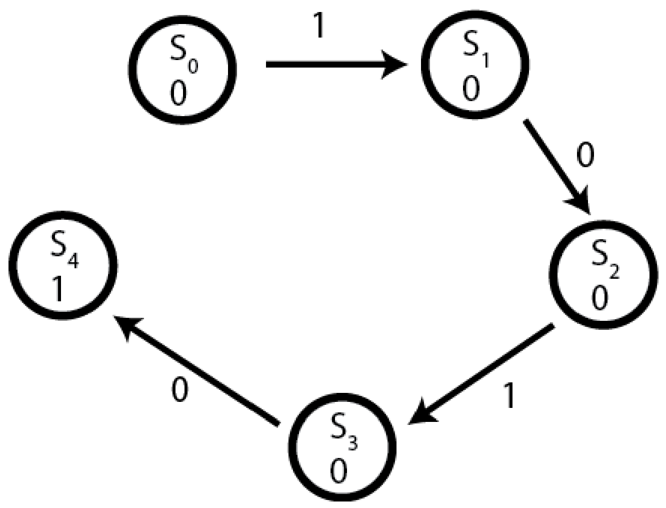
S_2 = \`10′

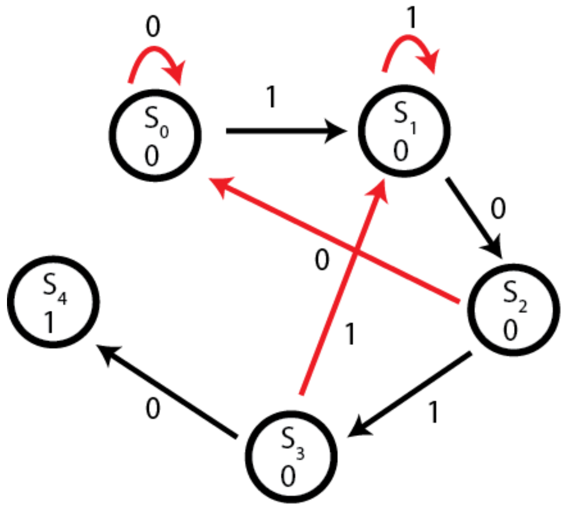
S_3 = \`101′

S_4 = \`1010′

Sequence Detector: Non-overlapping

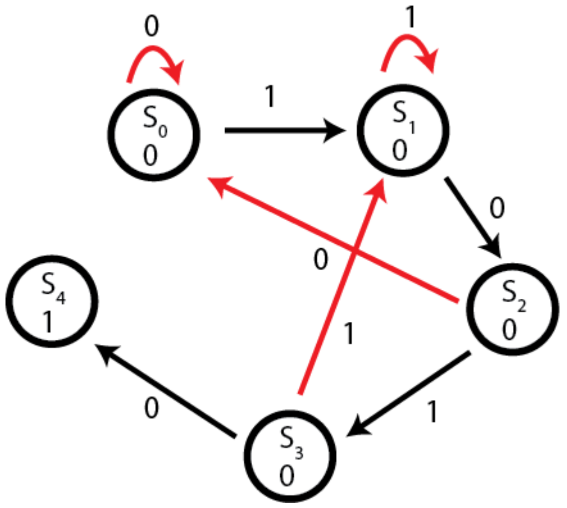
b) Draw the transitions that break the sequence

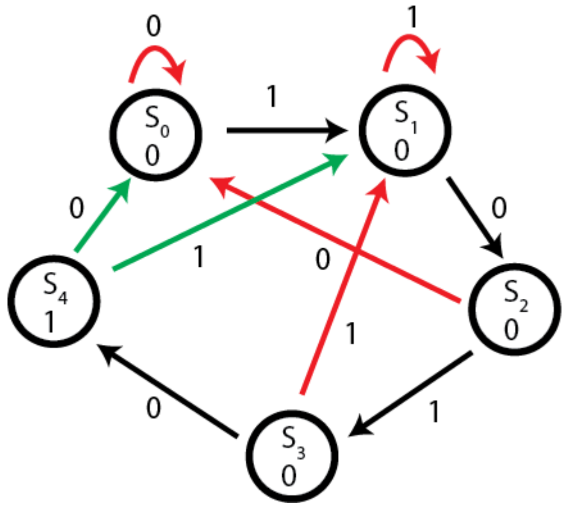




Sequence Detector: Non-overlapping

c) Draw the transition from the final state

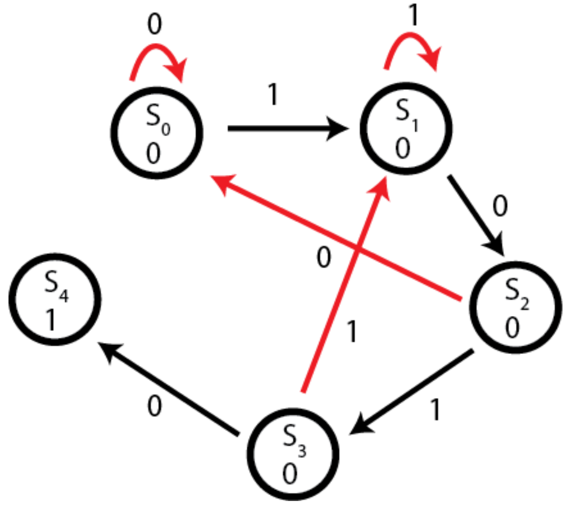


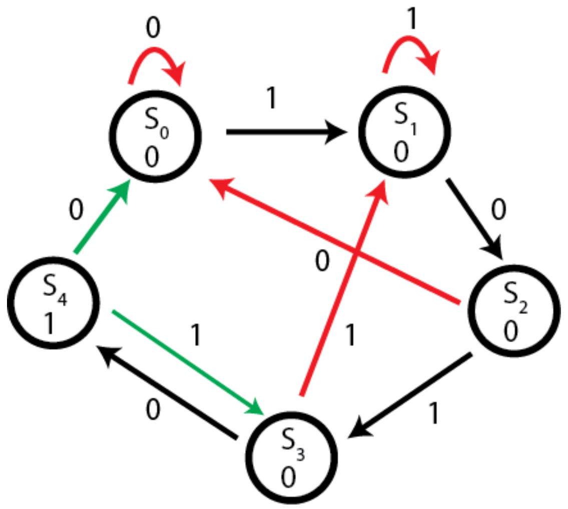


Sequence Detector: Overlapping

Problem 3: Draw the state diagram for an FSM that detects a overlapping sequence **1010**

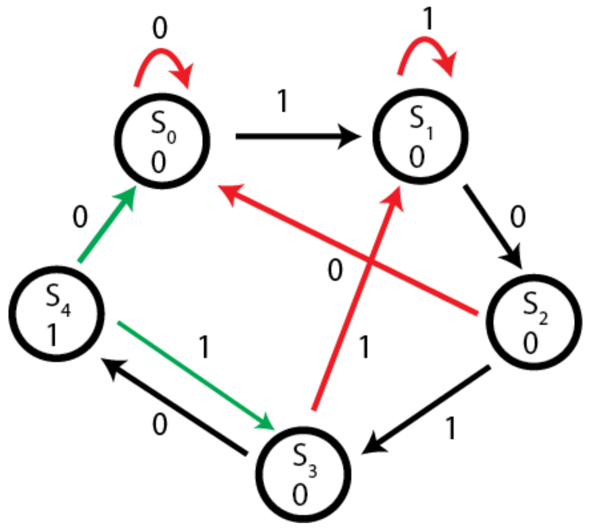
c) Draw the transition from the final state





Sequence Detector: Mealy Machine (1)

Problem 3: Draw the state diagram for an FSM that detects a overlapping sequence **1010.** The machine should be ***Mealy Type***.



State	Inputs		Output
	0	**1**	
S_0	S_0	S_1	0
S_1	S_2	S_1	0
S_2	S_0	S_3	0
S_3	S_4	S_1	0
S_4	S_0	S_3	1

State	Inputs	
	0	**1**
S_0	$S_0/0$	$S_1/0$
S_1	$S_2/0$	$S_1/0$
S_2	$S_0/0$	$S_3/0$
S_3	$S_4/1$	$S_1/0$
S_4	$S_0/0$	$S_3/0$

Sequence Detector: Mealy Machine (2)

State Reduction: Possible in ***Mealy Machine***

State	Inputs	
	0	1
S_0	$S_0/0$	$S_1/0$
S_1	$S_2/0$	$S_1/0$
$\boldsymbol{S_2}$	$\boldsymbol{S_0/0}$	$\boldsymbol{S_3/0}$
S_3	$S_4/1$	$S_1/0$
$\boldsymbol{S_4}$	$\boldsymbol{S_0/0}$	$\boldsymbol{S_3/0}$

State	Inputs	
	0	1
S_0	$S_0/0$	$S_1/0$
S_1	$S_2/0$	$S_1/0$
S_2	$S_0/0$	$S_3/0$
S_3	$S_2/1$	$S_1/0$

- The Mealy Machine requires one less state than the corresponding Moore Machine
- Mealy Machines make use of inputs also when computing the output.

Digital Circuits: Practice Problems

Problems 6.11-6.30

from “Digital Design”– M. Morris Mano & Michael D. Ciletti, Ed-5, Pearson (Prentice-Hall).