

EECS150 - Digital Design
Lecture 15 - Sequential Circuits II
(Finite State Machines revisited)

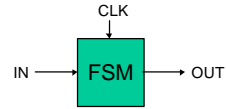
March 14, 2002
John Wawrzynek

Finite State Machines

• Example: Edge Detector

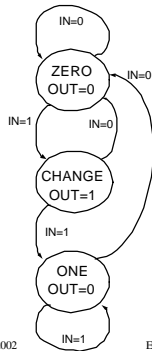
Bit are received one at a time (one per cycle),
such as: 000111010 → time

Design a circuit that asserts
its output for one cycle when
the input bit stream changes
from 0 to 1.



Try two different solutions.

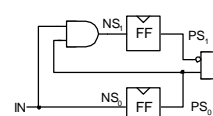
State Transition Diagram Solution A



	IN	PS	NS	OUT
ZERO	0	00	00	0
	1	00	01	0
CHANGE	0	01	00	1
	1	01	11	1
ONE	0	11	00	0
	1	11	11	0

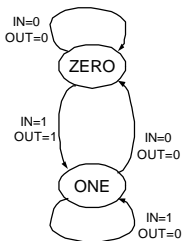
Solution A, circuit derivation

	IN	PS	NS	OUT
ZERO	0	00	00	0
	1	00	01	0
CHANGE	0	01	00	1
	1	01	11	1
ONE	0	11	00	0
	1	11	11	0



Solution B

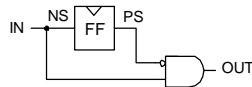
Output depends non only on PS but also on input, IN



Let ZERO=0, ONE=1

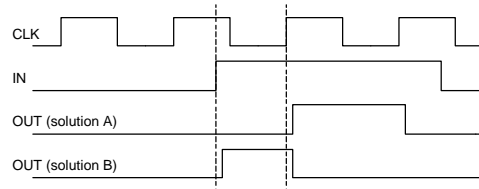
IN	PS	NS	OUT
0	0	0	0
0	1	0	0
1	0	1	1
1	1	1	0

NS = IN, OUT = IN PS'



What's the intuition about this solution?

Edge detector timing diagrams



- Solution A: output follows the clock
- Solution B: output changes with input rising edge and is asynchronous wrt the clock.

FSM Comparison

Solution A

Moore Machine

- output function only of PS
- maybe more state
- synchronous outputs
 - no glitching
 - one cycle "delay"
 - full cycle of stable output

Solution B

Mealy Machine

- output function of both PS & input
- maybe fewer states
- asynchronous outputs
 - if input glitches, so does output
 - output immediately available
 - output may not be stable long enough to be useful.

Spring 2002EECS150 - Lec15-seq2Page 7

FSM Recap

Moore Machine

Mealy Machine

Both machine types allow one-hot implementations.

Spring 2002EECS150 - Lec15-seq2Page 8

FSM Optimization

- State Reduction:**
 - Motivation: lower cost
 - fewer flip-flops in one-hot implementations
 - possibly fewer flip-flops in encoded implementations
 - more don't cares in NS logic
 - fewer gates in NS logic
 - Simpler to design with extra states then reduce later.
- Example: Odd parity checker. Two machines - identical behavior.

Spring 2002EECS150 - Lec15-seq2Page 9

State Reduction

- State Reduction is based on:
 - Two states are 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.
- If two states are equivalent, one can be eliminated without affecting the behavior of the FSM.
- Several algorithms exist:
 - Row matching method.
 - Implication table method.
- "Row Matching" is based on the state-transition table:
 - If two states have the same output, and both transition to the same next state, or both transition to each other, or both self-loop, then they are equivalent.
 - Combine the equivalent states into a new renamed state.
 - Repeat until no more states are combined.
 - Note: This algorithm is slightly different than the book.

Spring 2002EECS150 - Lec15-seq2Page 10

Row Matching Example

State Transition Table

PS	NS		output	
	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

Spring 2002EECS150 - Lec15-seq2Page 11

Row Matching Example

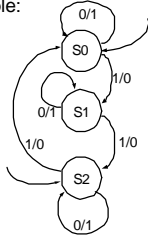
PS	NS		output	
	x=0	x=1	x=0	x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

Reduced State Transition Diagram

Spring 2002EECS150 - Lec15-seq2Page 12

State Reduction

- The "row matching" method is not guaranteed to result in the optimal solution in all cases, because it only looks at pairs of states.
- For example:



- Another (more complicated) method guarantees the optimal solution:
- "Implication table" method. See Mano, chapter 9.

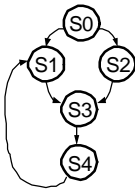
State Assignment (from Katz)

- In encoded (non-one-hot) FSMs, the choice of binary encodings for the states has an influence on the number of logic gates (or LUTs) needed to compute the next state and outputs.
- For n states, at least s bits are needed for a binary encoding.

$$\text{Where } s = \lceil \log_2 n \rceil$$

- $2^s!$ different encodings exist.
- We will look at several "by-hand" heuristic methods for choosing good assignments.
- Some CAD tools will make assignments automatically.

State Maps



Assignment

State	q2	q1	q0
S0	0	0	0
S1	1	0	1
S2	1	1	1
S3	0	1	0
S4	0	1	1

Assignment

State	q2	q1	q0
S0	0	0	0
S1	0	0	1
S2	0	1	0
S3	0	1	1
S4	1	1	1

q2	q1		q0	
0	00	01	11	10
0	S0	S4	S3	S2
1	S1	S2		

q2	q1		q0	
0	00	01	11	10
0	S0	S1	S3	S2
1			S4	

- "K-maps" are used to help visualize good encodings.
- Adjacent states in the STD should be made adjacent in the map.

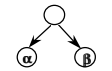
State Assignment

Alternative heuristics based on input and output behavior as well as transitions:



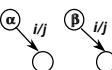
Adjacent assignments to:
states that share a common next state
(group 1's in next state map)

Highest Priority



states that share a common ancestor state
(group 1's in next state map)

Medium Priority

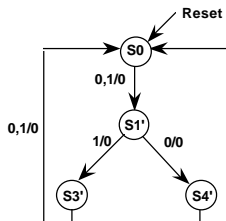


states that have common output behavior
(group 1's in output map)

Lowest Priority

Example

Example: 3-bit Sequence Detector



Highest Priority: (S3', S4')

Medium Priority: (S3', S4')

Lowest Priority:

0/0: (S0, S1', S3')

1/0: (S0, S1', S3', S4')

Example

Paper and Pencil Methods

Q0	Q1	
0	0	1
0	S0	S1'
1	S3'	S4'

Reset State = 00

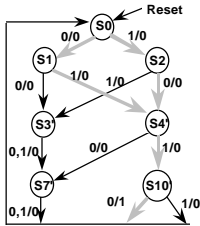
Highest Priority Adjacency

Q0	Q1	
0	0	1
0	S0	S3
1	S1'	S4'

Not much difference in these two assignments

State Assignment Example

Another Example: 4 bit String Recognizer



Highest Priority: $2x(S1, S2)$
 $(S3', S4'), (S7', S10')$

Medium Priority:
 $(S1, S2), 2x(S3', S4'), (S7', S10')$

Lowest Priority:
0/0: $(S0, S1, S2, S3', S4', S7')$
1/0: $(S0, S1, S2, S3', S4', S7')$

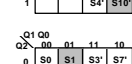
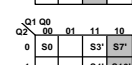
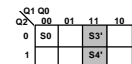
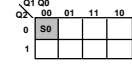
Spring 2002

EECS150 - Lec15-seq2

Page 19

State Assignment

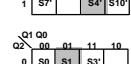
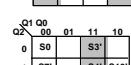
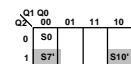
State Map



(a)

00 = Reset = S0

$(S1, S2), (S3', S4'), (S7', S10')$
placed adjacently



(b)

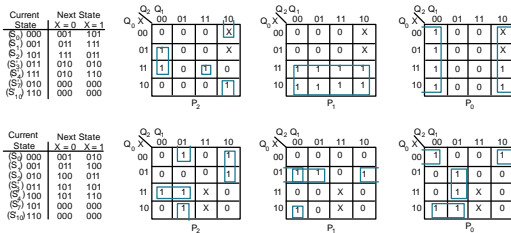
Spring 2002

EECS150 - Lec15-seq2

Page 20

State Assignment

Effect of Adjacencies on Next State Map



First encoding exhibits a better clustering of 1's in the next state map

Spring 2002

EECS150 - Lec15-seq2

Page 21