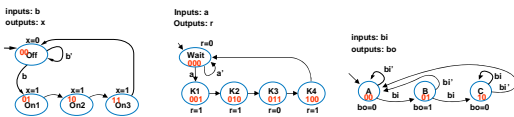


Lecture 12 Sequential Logic Optimization



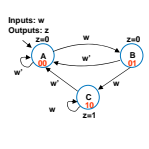
State Encoding

- Assigned each state an encoding
 - Unique bit representation to each state
 - Different encodings may optimize size, or tradeoff size and performance
- Currently using minimum bit-width binary encoding
 - Use smallest number of bits to ensure unique representation



Original Minimum Bit-width Encoding Consecutive 1's FSM

- Consider our original minimum bit-binary encoding assignment



	s1	s0	w	n1	n0	z
A	0	0	0	0	0	0
B	0	1	0	0	0	0
C	1	0	0	0	0	1
Unused	1	1	0	d	d	d
	1	1	1	d	d	d

s1	s0w	00	01	11	10
0	0	0	0	0	0
1	0	1	1	d	d

$z = s1$

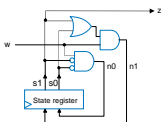
s1	s0w	00	01	11	10
0	0	0	0	1	0
1	0	1	1	d	d

$n1 = w(s1+s0)$

s1	s0w	00	01	11	10
0	0	0	1	0	0
1	0	0	d	d	d

$n0 = s1's0'w$

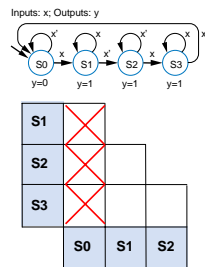
Cost = gate inputs * 2 = 14
Delay = 2 gate-delays



State Reduction

Example 1

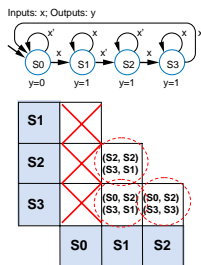
- Given FSM on the right
 - Step 1:** Mark state pairs having different outputs as nonequivalent



State Reduction

Example 1

- Given FSM on the right
 - Step 1:** Mark state pairs having different outputs as nonequivalent
 - Step 2:** For each unmarked state pair, write the next state pairs for the same input values



For pair (S2, S1) For pair (S3, S1) For pair (S3, S2)

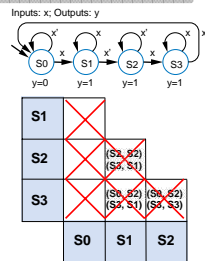
X = 0 X = 0 X = 0
 S2 goes to S2 S3 goes to S0 S3 goes to S0
 S1 goes to S2 S1 goes to S2 S2 goes to S2

X = 1 X = 1 X = 1
 S2 goes to S3 S3 goes to S3 S3 goes to S3
 S1 goes to S1 S1 goes to S1 S2 goes to S3

State Reduction

Example 1

- Given FSM on the right
 - Step 1:** Mark state pairs having different outputs as nonequivalent
 - Step 2:** For each unmarked state pair, write the next state pairs for the same input values
 - Step 3:** For each unmarked state pair, mark state pairs having nonequivalent next state pairs as nonequivalent.
 - Repeat this step until no change occurs, or until all states are marked.



S2 = S2? Yes Repeat
 S3 = S1? Unsure S2 = S2? Yes
 S0 = S2? NO! S3 = S1? NO!
 S0 = S2? NO!

Partitioning Method

- Second test
 - For all possible sequences of inputs, the FSM outputs will be the same starting from either state

$P2 = (ABD)(CEFG)$

When $w = 0$

C goes to F
E goes to F
F goes to E
G goes to F

F, F, E, and F are in same partition

When $w = 1$

C goes to E
E goes to C
F goes to D
G goes to G

D is in different partition than E, C, G!

Move F into it's own partition

ECE 474a/575a
Susan Lysdecky

Current State	Inputs		Outputs	
	w	z	Next State	z
A	0	B	1	
A	1	C	1	
B	0	D	1	
B	1	F	1	
C	0	F	0	
C	1	E	0	
D	0	B	1	
D	1	G	1	
E	0	F	0	
E	1	C	0	
F	0	E	0	
F	1	D	0	
G	0	F	0	
G	1	G	0	

*Use state names instead of state encoding to simply table

37 of 43

Partitioning Method

- Second test
 - For all possible sequences of inputs, the FSM outputs will be the same starting from either state

$P3 = (ABD)(CEG)(F)$ *New partitions – start over!*

When $w = 0$

A goes to B
B goes to D
D goes to B

B, D, and B are in same partition

When $w = 1$

A goes to C
B goes to F
D goes to G

F is in different partition than C and G!

Move B to it's own partition.

ECE 474a/575a
Susan Lysdecky

Current State	Inputs		Outputs	
	w	z	Next State	z
A	0	B	1	
A	1	C	1	
B	0	D	1	
B	1	F	1	
C	0	F	0	
C	1	E	0	
D	0	B	1	
D	1	G	1	
E	0	F	0	
E	1	C	0	
F	0	E	0	
F	1	D	0	
G	0	F	0	
G	1	G	0	

*Use state names instead of state encoding to simply table

38 of 43

Partitioning Method

- Second test
 - For all possible sequences of inputs, the FSM outputs will be the same starting from either state

$P4 = (AD)(B)(CEG)(F)$ *New partitions – start over!*

When $w = 0$

A goes to B
D goes to B

B and B in same partition

When $w = 1$

A goes to C
D goes to G

C and G in same partition

ECE 474a/575a
Susan Lysdecky

Current State	Inputs		Outputs	
	w	z	Next State	z
A	0	B	1	
A	1	C	1	
B	0	D	1	
B	1	F	1	
C	0	F	0	
C	1	E	0	
D	0	B	1	
D	1	G	1	
E	0	F	0	
E	1	C	0	
F	0	E	0	
F	1	D	0	
G	0	F	0	
G	1	G	0	

*Use state names instead of state encoding to simply table

39 of 43

Need for Automation

- What happens when FSM get even bigger?
 - Table for large FSM too big for humans to work with
 - 100 states would have table with $100 \times 100 = 100,000$ state pairs cells
- Automation needed
 - State reduction typically automated
 - Often using heuristics to reduce compute time

