ECE 274 Digital Logic

Optimization and Tradeoffs
State Encodings, Moore vs. Mealy FSMs
Digital Design 6.3

THE UNIVERSITY OF
ARIZONA.
TUCSON ARIZONA

Digital Design

Chapter 6:
Optimization and Tradeoffs

Sequential Optimizations and Tradeoffs
State Encoding

- **Encoding**: Assigning a unique bit representation to each state
- Different encodings may optimize size, or tradeoff size and performance
- Consider 3-Cycle Laser Timer…
  - Example 3.7's encoding: **15** gate inputs
  - Try alternative encoding
    - x = s1 + s0
    - n1 = s0
   - n0 = s1′b + s1′s0
    - Only **8** gate inputs

Inputs: b; Outputs: x
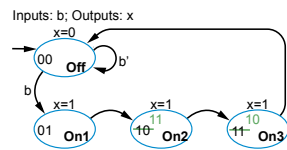
	Inputs			Outputs		
	s1	s0	b	x	n1	n0
Off	0	0	0	0	0	0
	0	0	1	0	0	1
On1	0	1	0	1	1	0
	0	1	1	1	1	0
On2	1	0	0	1	1	1
	1	0	1	1	1	1
On3	1	1	0	1	0	0
	1	1	1	1	0	0

3

Sequential Optimizations and Tradeoffs
State Encoding: One-Hot Encoding

Inputs: none; Outputs: x

- **One-hot encoding**
  - One bit per state – a bit being '1' corresponds to a particular state
  - Alternative to *minimum bit-width encoding* in previous example
  - For A, B, C, D: A: 0001, B: 0010, C: 0100, D: 1000
- Example: FSM that outputs 0, 1, 1, 1
  - Equations if one-hot encoding:
    - n3 = s2; n2 = s1; n1 = s0; x = s3 + s2 + s1
  - Fewer gates and only one level of logic – less delay than two levels, so faster clock frequency

	Inputs		Outputs		
	s1	s0	n1	n0	x
A	0	0	0	1	0
B	0	1	1	0	1
C	1	0	1	1	1
D	1	1	0	0	1
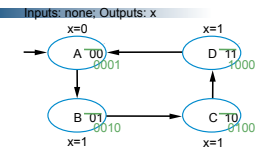
	Inputs				Outputs				
	s3	s2	s1	s0	n3	n2	n1	n0	x
A	0	0	0	1	0	0	1	0	0
B	0	0	1	0	0	1	0	0	1
C	0	1	0	0	1	0	0	0	1
D	1	0	0	0	0	0	1	1	1

4

1
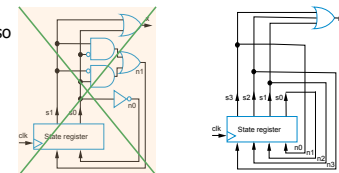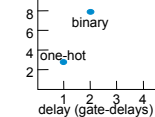
Sequential Optimizations and Tradeoffs
One-Hot Encoding Example: Three-Cycles-High Laser Timer

- o Four states – Use four-bit one-hot encoding
  - □ State table leads to equations:
    - o $x = s3 + s2 + s1$
    - o $n3 = s2$
    - o $n2 = s1$
    - o $n1 = s0*b$
    - o $n0 = s0*b' + s3$
  - □ Smaller
    - o $3+0+0+2+(2+2) =$ **9** gate inputs
    - o Earlier binary encoding (Ch 3): **15** gate inputs
  - □ Faster
    - o Critical path: $n0 = s0*b' + s3$
    - o Previously: $n0 = s1's0'b + s1s0'$
    - o 2-input AND slightly faster than 3-input AND

Inputs: b; Outputs: x

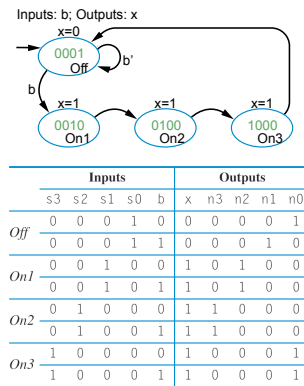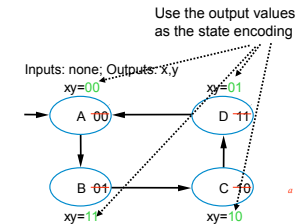|	Inputs					Outputs				
	s3	s2	s1	s0	b	x	n3	n2	n1	n0
Off	0	0	0	1	0	0	0	0	0	1
	0	0	0	1	1	0	0	0	1	0
On1	0	0	1	0	0	1	0	1	0	0
	0	0	1	0	1	1	0	1	0	0
On2	0	1	0	0	0	1	1	0	0	0
	0	1	0	0	1	1	1	0	0	0
On3	1	0	0	0	0	1	0	0	0	1
	1	0	0	0	1	1	0	0	0	1

5

Sequential Optimizations and Tradeoffs
State Encoding: Output Encoding

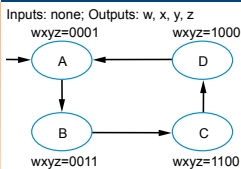- o **Output encoding**: Encoding method where the state encoding is same as the output values
 - □ Possible if enough outputs, all states with unique output values

Use the output values as the state encoding

Inputs: none; Outputs: x, y

6

Sequential Optimizations and Tradeoffs
Output Encoding Example: Sequence Generator

Inputs: none; Outputs: w, x, y, z

	Inputs				Outputs			
	s3	s2	s1	s0	n3	n2	n1	n0
A	0	0	0	1	0	0	1	1
B	0	0	1	1	1	1	0	0
C	1	1	0	0	1	0	0	0
D	1	0	0	0	0	0	0	1

- o Generate sequence 0001, 0011, 1110, 1000, repeat
  - □ FSM shown
- o Use output values as state encoding
- o Create state table
- o Derive equations for next state
  - □ $n3 = s1 + s2; n2 = s1; n1 = s1's0; n0 = s1's0 + s3s2'$
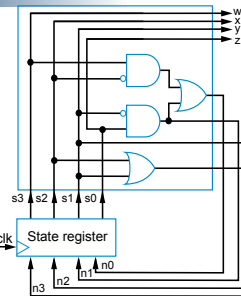
7

Sequential Optimizations and Tradeoffs
Moore vs. Mealy FSMs

Moore (a)   Mealy (b)
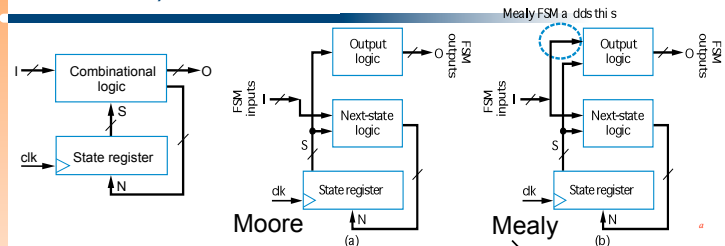
- o FSM implementation architecture
  - □ State register and logic
  - □ Next state logic – function of present state and FSM inputs
  - □ Output logic
    - o If function of present state only – **Moore FSM**
    - o If function of present state and FSM inputs – **Mealy FSM**

Inputs: b; Outputs: x

Graphically: show outputs with arcs, not with states

8

2
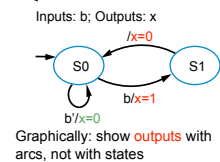
Sequential Optimizations and Tradeoffs
Moore vs. Mealy FSMs: Mealy FSMs May Have Fewer States
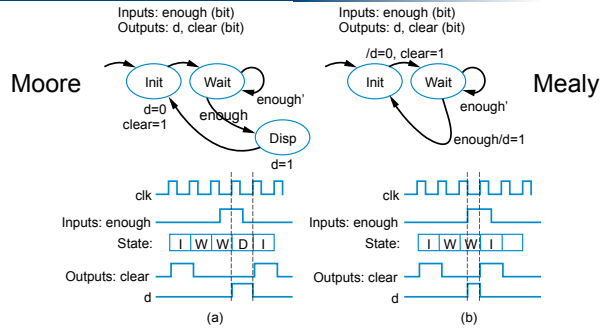
Moore

Inputs: enough (bit)
Outputs: d, clear (bit)

Init → Wait → Disp
d=0 clear=1
enough' (Wait self-loop)
enough
d=1

Mealy

Inputs: enough (bit)
Outputs: d, clear (bit)

/d=0, clear=1
Init → Wait
enough'
enough/d=1

clk
Inputs: enough
State: I W W D I
Outputs: clear
d
(a)

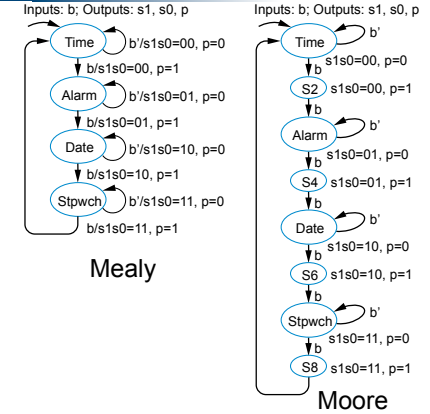clk
Inputs: enough
State: I W W I
Outputs: clear
d
(b)

- Soda dispenser example: Initialize, wait until enough, dispense
  - Moore: 3 states; Mealy: 2 states

Sequential Optimizations and Tradeoffs
Moore vs. Mealy FSMs

- Q: Which is Moore, and which is Mealy?
- A: Mealy on left, Moore on right
  - Mealy outputs on arcs, meaning outputs are function of state AND INPUTS
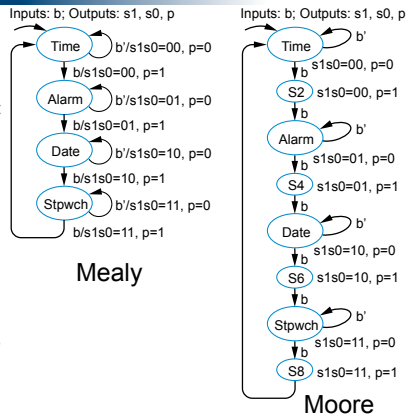 - Moore outputs in states, meaning outputs are function of state only

Inputs: b; Outputs: s1, s0, p

Time b'/s1s0=00, p=0
  b/s1s0=00, p=1
Alarm b'/s1s0=01, p=0
  b/s1s0=01, p=1
Date b'/s1s0=10, p=0
  b/s1s0=10, p=1
Stpwch b'/s1s0=11, p=0
  b/s1s0=11, p=1

Mealy

Inputs: b; Outputs: s1, s0, p

Time b'
  b s1s0=00, p=0
S2 s1s0=00, p=1
  b
Alarm b' s1s0=01, p=0
  b
S4 s1s0=01, p=1
  b
Date b' s1s0=10, p=0
  b
S6 s1s0=10, p=1
  b
Stpwch b' s1s0=11, p=0
  b
S8 s1s0=11, p=1

Moore

Sequential Optimizations and Tradeoffs
Mealy vs. Moore Example: Beeping Wristwatch

- Button b
  - Sequences mux select lines *s1s0* through 00, 01, 10, and 11
    - Each value displays different internal register
  - Each unique button press should cause 1-cycle beep, with *p*=1 being beep
- Must wait for button to be released (*b'*) and pushed again (*b*) before sequencing
  - Note that Moore requires unique state to pulse *p*, while Mealy pulses *p* on arc
  - Tradeoff: Mealy's pulse on *p* may not last one full cycle

Inputs: b; Outputs: s1, s0, p

Time b'/s1s0=00, p=0
  b/s1s0=00, p=1
Alarm b'/s1s0=01, p=0
  b/s1s0=01, p=1
Date b'/s1s0=10, p=0
  b/s1s0=10, p=1
Stpwch b'/s1s0=11, p=0
  b/s1s0=11, p=1

Mealy

Inputs: b; Outputs: s1, s0, p

Time b'
  b s1s0=00, p=0
S2 s1s0=00, p=1
  b
Alarm b' s1s0=01, p=0
  b
S4 s1s0=01, p=1
  b
Date b' s1s0=10, p=0
  b
S6 s1s0=10, p=1
  b
Stpwch b' s1s0=11, p=0
  b
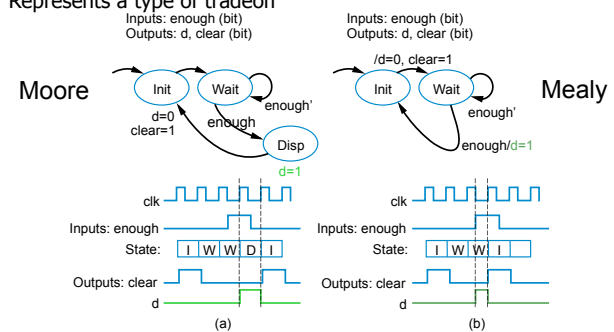S8 s1s0=11, p=1

Moore

Sequential Optimizations and Tradeoffs
Moore vs. Mealy Tradeoff

- Mealy outputs change mid-cycle if input changes
  - Note earlier soda dispenser example
    - Mealy had fewer states, but output *d* not 1 for full cycle
 - Represents a type of tradeoff

Moore

Inputs: enough (bit)
Outputs: d, clear (bit)

Init → Wait → Disp
d=0 clear=1
enough'
enough
d=1

Mealy

Inputs: enough (bit)
Outputs: d, clear (bit)

/d=0, clear=1
Init → Wait
enough'
enough/d=1

clk
Inputs: enough
State: I W W D I
Outputs: clear
d
(a)

clk
Inputs: enough
State: I W W I
Outputs: clear
d
(b)

Sequential Optimizations and Tradeoffs
Implementing a Mealy FSM

- ○ Straightforward
  - ▫ Convert to state table
  - ▫ Derive equations for each output
  - ▫ Key difference from Moore: External outputs (*d, clear*) may have different value in same state, depending on input values
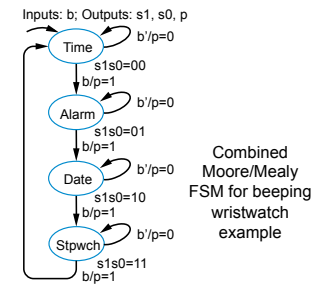
Inputs: enough (bit)
Outputs: d, clear (bit)
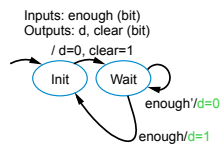
	Inputs		Outputs		
	s0	enough	n0	d	clear
Init	0	0	1	0	1
	0	1	1	0	1
Wait	1	0	1	0	0
	1	1	0	1	0

Sequential Optimizations and Tradeoffs
Mealy and Moore can be combined

- ○ Final note on Mealy/Moore
  - ▫ May be combined in same FSM

Inputs: b; Outputs: s1, s0, p

Combined Moore/Mealy FSM for beeping wristwatch example