CSEE 4823 Advanced Logic Design
Handout: Lecture #6
9/22/16

Prof. Steven M. Nowick
nowick@cs.columbia.edu

Department of Computer Science (and Elect. Eng.)
Columbia University
New York, NY, USA

Iterative Circuits:
Example #2
(Mealy-machine based)

Pattern Detector: Verbal Description

General Problem Statement: given a string X of input bits, which are expected to follow the repeated pattern "0011" (i.e. 001100110011…), produce a corresponding string Z of output bits which flag any error locations with a '1' bit (and all correct locations with a '0' bit).

More Detailed Requirements: flagging errors

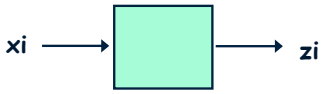
- the machine first expects a '0' input (if it receives any '1' inputs, stay in initial state and flag these erroneous inputs)
- after it receives its first '0' bit, it outputs a '0' (no error), and waits for a second bit
  - if second bit is '0', no error (output '0')
  - if second bit is '1', has error (output '1')
  - … in each case, absorb the second input, and prepare for the next pair of '11' bits
- if the machine receives any more '0' inputs, stay in the current state and flag these erroneous inputs
- after it receives its first '1' bit, it outputs a '0' (no error), and waits for a second bit
  - if second bit is '1', no error (output '0')
  - if second bit is '0', has error (output '1')
  - … in each case, absorb the second input, and prepare for the next pair of '00' bits

… go to top and repeat

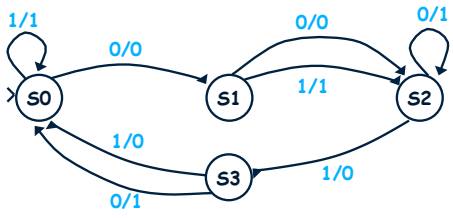
#3

Iterative Circuit Synthesis: Mealy-based

FSM Block Diagram: *bit-serial processing*



Mealy State Diagram: *bit-serial processing*



#4

Iterative Circuit Synthesis: Mealy-based

Symbolic State Table:

PS	input x_i	NS	output z_i
S0	0	S1	0
S0	1	S0	1
S1	0	S2	0
S1	1	S2	1
S2	0	S2	1
S2	1	S3	0
S3	0	S0	1
S3	1	S0	0

State Assignment (i.e. State Encoding):

State	Assigned Code a_i b_i
S0	0 0
S1	0 1
S2	1 1
S3	1 0

#5

Iterative Circuit Synthesis: Mealy-based

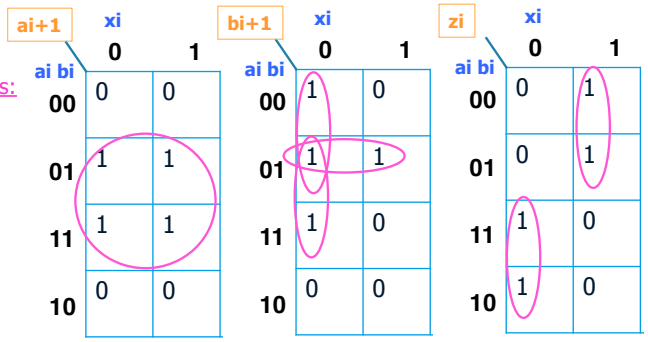
Encoded State Table:

PS a_i b_i	input x_i	NS a_{i+1} b_{i+1}	output z_i
00	0	01	0
00	1	00	1
01	0	11	0
01	1	11	1
11	0	11	1
11	1	10	0
10	0	00	1
10	1	00	0

#6

Iterative Circuit Synthesis: Mealy-based

Karnaugh Maps:



Final 2-Level (Sum-of-Products) Minimized Equations -- with some logic optimization added

ai+1 = bi

bi+1 = ai' xi' + xi' bi + ai' bi

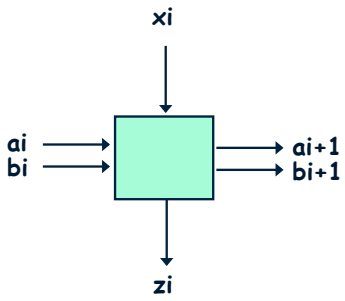
zi = xi ai' + xi' ai = xi ⊕ ai

#7

Iterative Circuit Synthesis: Mealy-based

Iterative Cell: Gate-Level Implementation

regular instance (i.e. typical cell): use equations on previous page

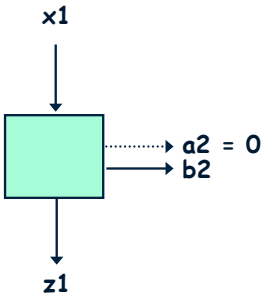


#8

Iterative Circuit Synthesis: Mealy-based

Iterative Cell: Gate-Level Implementation

Optimized leftmost cell (= Cell #1): simplify earlier equations



Special condition for cell #1: a1 b1 = 00!

$a2 = b1 = 0$

$b2 = a1' x1' + x1' b1 + a1' b1 = x1'$

$z1 = x1 a1' + x1' a1 = x1$

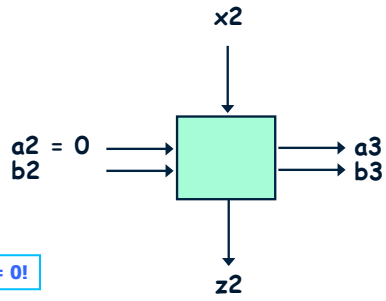
Note: can eliminate a2 output, and simply have a '0' a2 input to next cell (Cell #2)

#9

Iterative Circuit Synthesis: Mealy-based

Iterative Cell: Gate-Level Implementation

Optimized second-to-leftmost cell (= Cell #2): propagate simplification



Special condition for cell #2: a2 = 0!

$a3 = b2$

$b3 = a2' x2' + x2' b2 + a2' b2 = x2' + x2' b2 + b2 = x2' + b2$

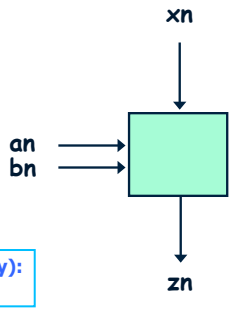
$z2 = x2 a2' + x2' a2 = x2 \oplus a2 = x2$

#10

Iterative Circuit Synthesis: Mealy-based

Iterative Cell: Gate-Level Implementation

Optimized rightmost cell (= Cell #N): no need for next-state logic -- DELETE!



Special condition for cell #N (Mealy only):
no next-state logic

$zn = xn\ an' + xn'\ an = xn \oplus an$

#11