Finite State Machines

Contents

State Machine Design

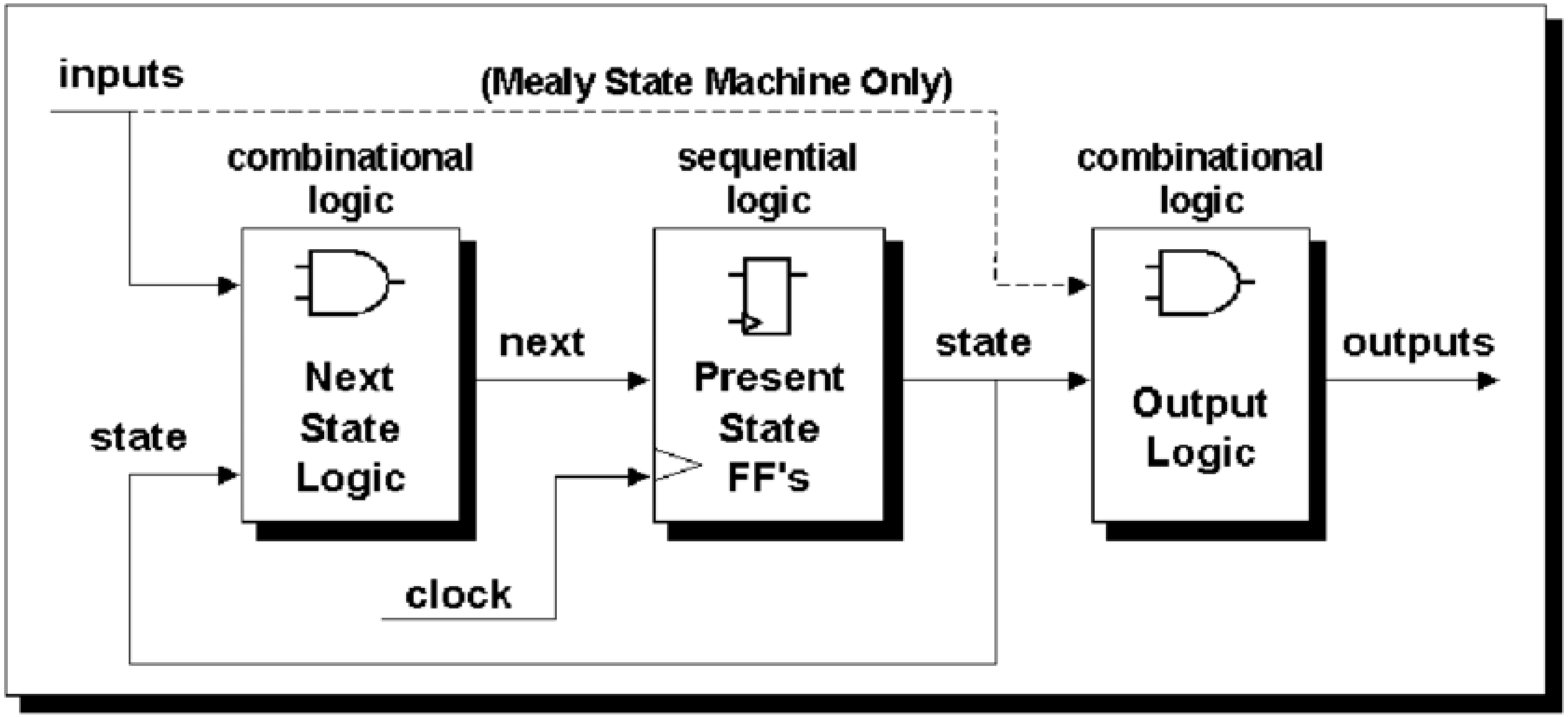
Sequential logic, circuits, or machines:

1. Have internal memory
2. Types:
    - *Synchronous* (clocked) – memory elements controlled by an external signal – can change only at specific times
    - *Asynchronous* – less frequently used but more interesting – memory elements change state whenever 1 or more inputs change – no clock

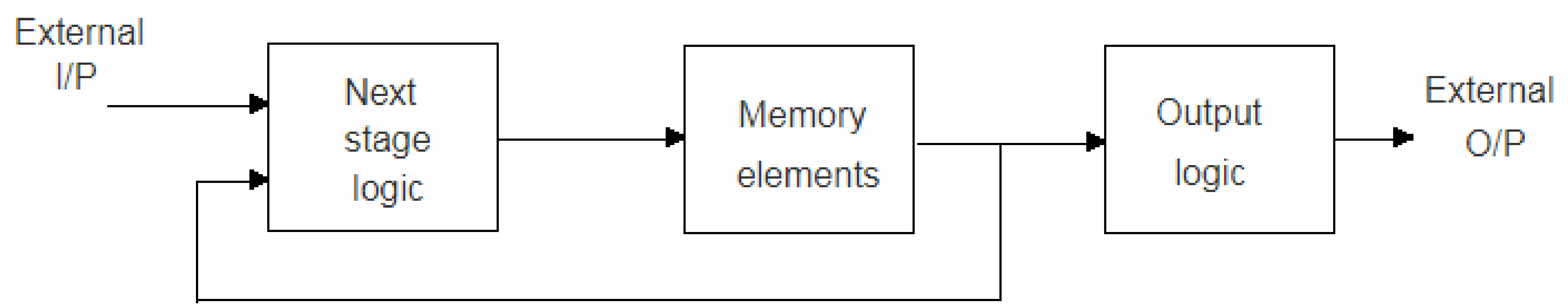
State Machine Design

3. VERY IMPORTANT: Control conditions under which state changes
    - Otherwise single input change causes many state changes, due to relative logic delays
4. *Asynchronous Logic*:
    - Faster than synchronous for small circuits
    - Slower than synchronous for large circuits
        - REASON: Vastly more logic is required due to absence of CLOCK

Mealy Machines

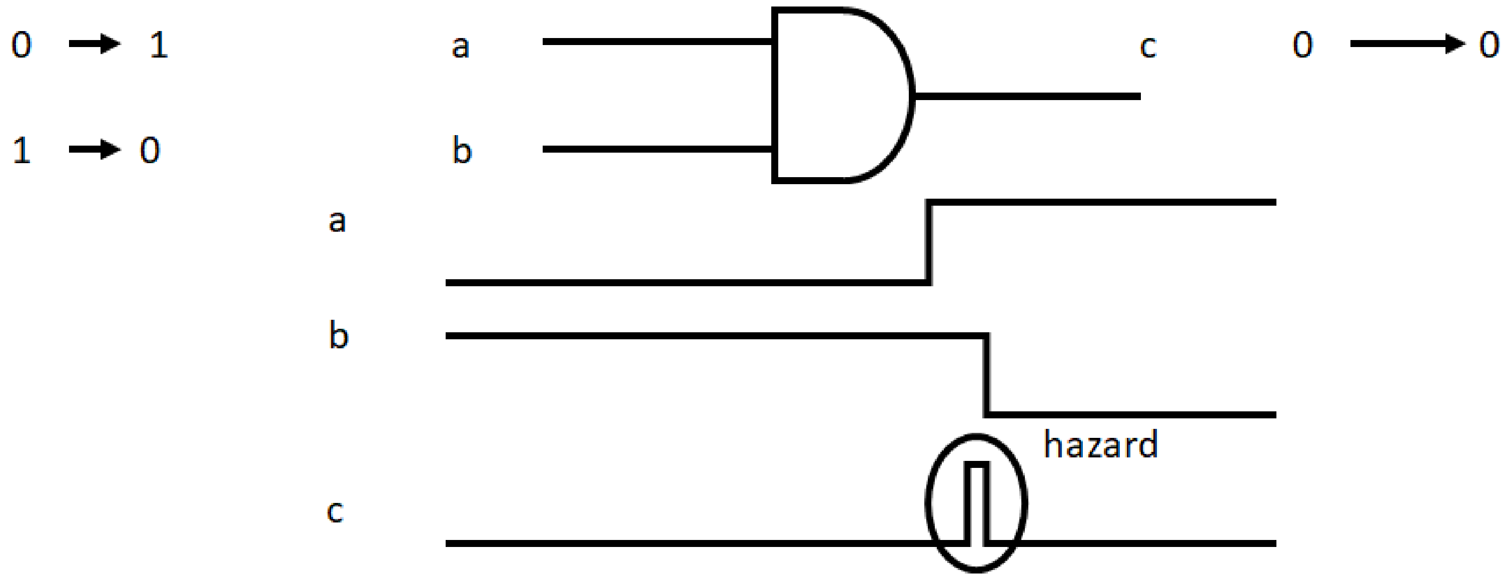


Moore Machines



Mealy Machines

- Nasty to design reliably and debug
- WHY?
  - Real circuits have hazards:
  - Undesirable: You expect c to be 0, and run it as input to a flip-flop which catches the short logic 1 pulse on c (called *one's catching*)
  - Flip-flop gets set, but you expected it to be cleared



Hazards

- Unavoidable
- Different signals have different propagation delays
  - Different paths through circuit
  - Different logic gates have different delay times – determined by:
    1. Gate type
    2. Number of inputs
- Mealy machines do not filter out hazards, from inputs to outputs
  - WHY? Output decoder is a function of inputs as well as of state

Moore Machine

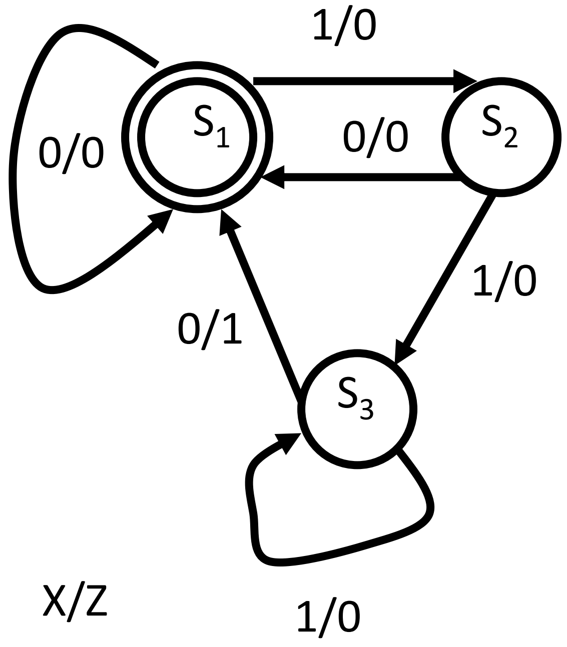
- Output is stable:
  - Filters out hazards in primary outputs, since they cannot propagate from inputs to outputs
- Rule: Never design a Mealy Machine unless you really have to
  - Unfortunately, you often have to do it to satisfy the circuit functional specification

State Machine Design Process

1. Identify State Variables S
2. Identify Output Decoder & Next State Decoder
3. Build State Transition Diagram
4. Minimize States
5. Choose appropriate type of flip-flops
6. Choose *State Assignment*
    - Assignment of binary codes to machine states
7. Design next state decoder & output decoder – use combinational logic structured design methods – K-maps, Variable-Entered Map, Verilog

Mealy Machine Sequence Detector Recognizing 110_2

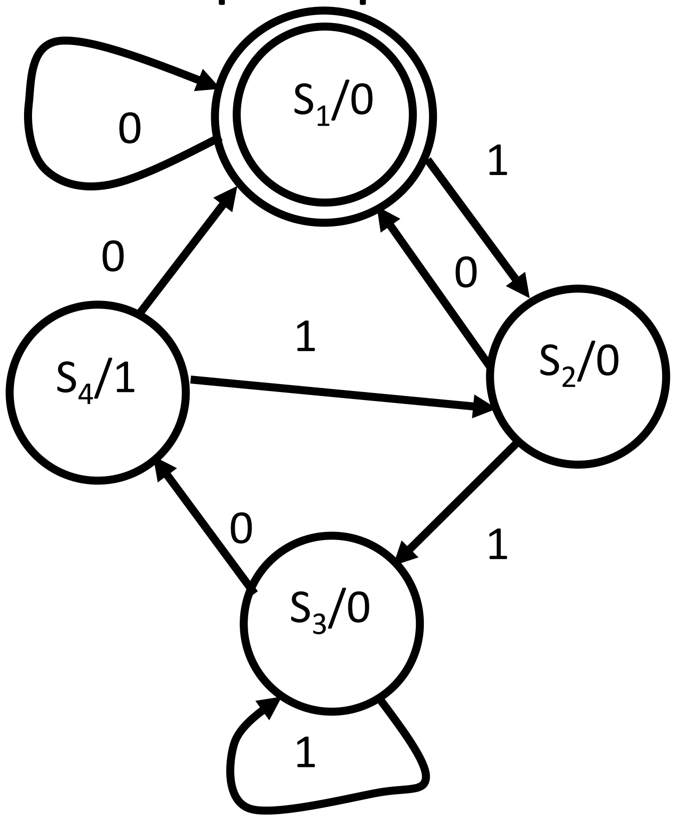
- Double circle shows reset state



Present State	Present Input X/Z	
	0	1
S_1	$S_1/0$	$S_2/0$
S_2	$S_1/0$	$S_3/0$
S_3	$S_1/1$	$S_3/0$

Moore Machine Sequence Detector Recognizing 110_2

- Pay for better behavior of Moore machine with extra flip-flop



	Present Input		Present Output
Present State	0	1	Z
S_1	S_1	S_2	0
S_2	S_1	S_3	0
S_3	S_4	S_3	0
S_4	S_1	S_2	1

Mixed Machine

- Demonstrate combined FSMs
- FSMs for 2 sequences (101 and 1001)

FSM Optimization

- More number of states leads to more area
- If next states for 2 state are same
- If output of 2 states is same

FSM Optimization

- A and B equivalent

PS	NS,Z	
	X=0	X=1
A	C,1	E, 1
B	C, 1	E,1
C	B,0	A,1
D	D,0	E,1
E	D,1	A,0

FSM Optimization

PS	NS,Z	
	X=0	X=1
A	C,1	E,1
C	A,0	A,1
D	D,0	E,1
E	D,1	A,0

FSM Optimization - Implication

Present State	Next State X=0	Next State X=1	Output
A	D	C	0
B	F	H	0
C	E	D	1
D	A	E	0
E	C	A	1
F	F	B	1
G	B	H	0
H	C	G	1

B	X D-F, C-H						
C	X	X					
D	A-D, C-E	C-F, A-H X	X				
E	X	X	C-E, A-D	X			
F	X	X	E-F, B-D X	X	C-F, A-B X		
G	B-D, C-H X	B-F X	X	A-B, E-H X	X	X	
H	X	X	C-E, D-G X	X	A-G X	C-F, B-G X	X
	A	B	C	D	E	F	G

FSM Optimization - Implication

A=D
C=E

Present State	Next State X=0	Next State X=1	Output
A	A	C	0
B	F	H	0
C	C	A	1
~~D~~	~~A~~	~~E~~	~~0~~
~~E~~	~~C~~	~~A~~	~~1~~
F	F	B	1
G	B	H	0
H	C	G	1

Assignment

- Optimize the following using implication method

PS	I1	I2
A	E,0	B,0
B	F,0	A,0
C	E,-	C,0
D	F,1	D,0
E	C,1	C,0
F	D,-	B,0