

Department of Electrical & Computer Engineering

# Digital Logic And Computing Systems

## Chapter 04 –Automata, Control Path

Dr. Christophe Bobda

# Agenda

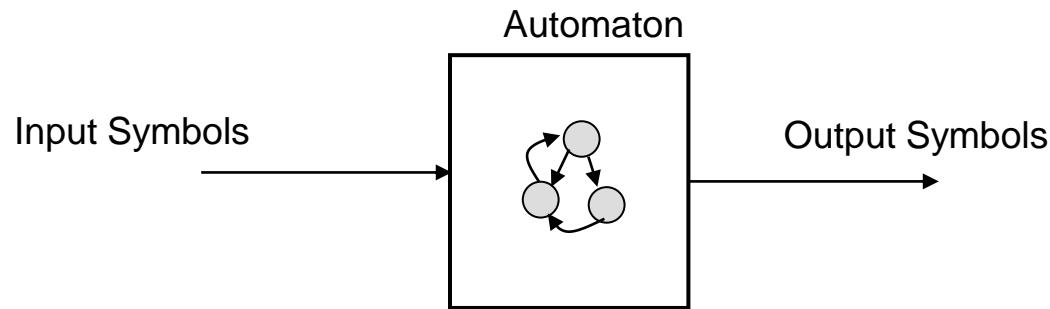
- ❑ Sequential Circuits
- ❑ Mealy- and Moore Automata
- ❑ Design of Sequential Circuits
- ❑ Case studies

# Sequential Circuits

- ❑ Outputs in combinational circuits are functions of the inputs only.
    - Circuits with high number of inputs require large input “alphabet”.
      - Technologically difficult and expensive to realize.
      - Impossible to use.
  - A sequential circuit is a circuit whose outputs depend on the current and previous inputs.
    - Memory is required to store the circuit’s state.
  - We need a viable model, to describe, analyze, optimize and implement a sequence of actions.
- Finite State Automata (Finite State Machine).

# Sequential Circuits- Modeling with Automata

- ❑ We need a formal basis to design, analyze, and optimize sequential circuits → Automata Theory
  - An automaton has inner states, reads input symbols and writes output symbols



- An automaton operates stepwise. In each step, an input symbol is read, an output symbol is produced and internal state possibly changes  
→ this discrete time model can be implemented using synchronous sequential logic
- The transition to the next step is driven by a clock signal

# Mealy-Automata

Definition: A Mealy-Automaton is a tuple  $A=(X, Y, S, \delta, \lambda, s_i)$  where

$X$ : finite, non-empty set of input symbols

$Y$ : finite, non-empty set of output symbols

$S$ : finite, non-empty set of states

$\delta: S \times X \rightarrow S$  State transition function

$\lambda: S \times X \rightarrow Y$  Output function

$s_i$ : Initial state

State set finite  $\rightarrow$  finite state automata (finite state machine, FSM)

- ❑ Mealy-Automata can uniquely be represented through
  - a state transition table or
  - state transition diagram (graph)

# Mealy – State Transition Table

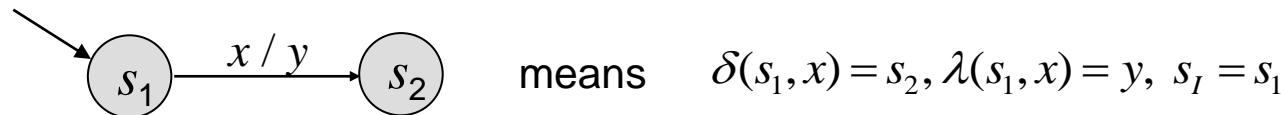
- ❑ A state transition table is a table in which the rows represent states and columns represent input symbols. An entry  $(\delta(s,x) / \lambda(s,x))$  captures the transition from state  $s \in S$  with input symbol  $x \in X$  to state  $\delta(s,x)$  with output  $\lambda(s,x)$
- ❑ Example

$$X = \{x_1, x_2, x_3\}, \quad Y = \{y_1, y_2\}, \quad S = \{s_1, s_2, s_3, s_4\}, \quad s_I = s_1$$

$\delta / \lambda$	$x_1$	$x_2$	$x_3$
$s_I = s_1$	$s_2 / y_1$	$s_3 / y_2$	$s_2 / y_1$
$s_2$	$s_1 / y_2$	$s_3 / y_2$	$s_2 / y_1$
$s_3$	$s_3 / y_1$	$s_2 / y_1$	$s_4 / y_1$
$s_4$	$s_3 / y_2$	$s_1 / y_2$	$s_4 / y_2$

# Mealy – State Transition Graph

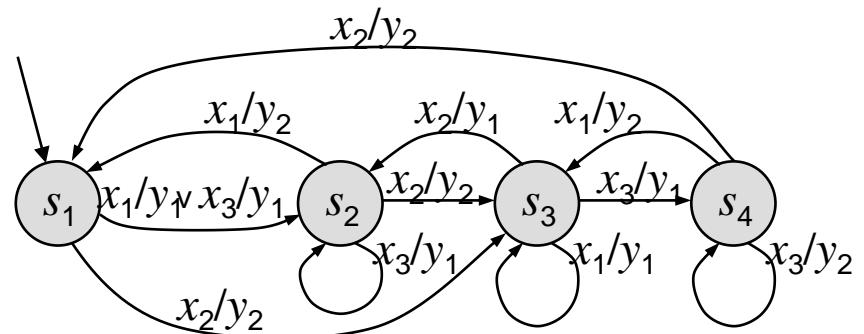
- A state transition graph is a directed graph in which nodes represent states, and edges represent transitions
  - Edges are labeled with the values of the transition function
  - The node without precedence represents the initial state



- Example

$$X = \{x_1, x_2, x_3\}, \quad Y = \{y_1, y_2\}, \quad S = \{s_1, s_2, s_3, s_4\}, \quad s_I = s_1$$

$\delta / \lambda$	$x_1$	$x_2$	$x_3$
$s_I = s_1$	$s_2 / y_1$	$s_3 / y_2$	$s_2 / y_1$
$s_2$	$s_1 / y_2$	$s_3 / y_2$	$s_2 / y_1$
$s_3$	$s_3 / y_1$	$s_2 / y_1$	$s_4 / y_1$
$s_4$	$s_3 / y_2$	$s_1 / y_2$	$s_4 / y_2$



# Mealy Automata

Definition: An automaton is deterministic, if :

$$\forall x \in X, \forall s \in S : |\delta(s, x)| \leq 1$$

Definition: An automaton is complete, if:

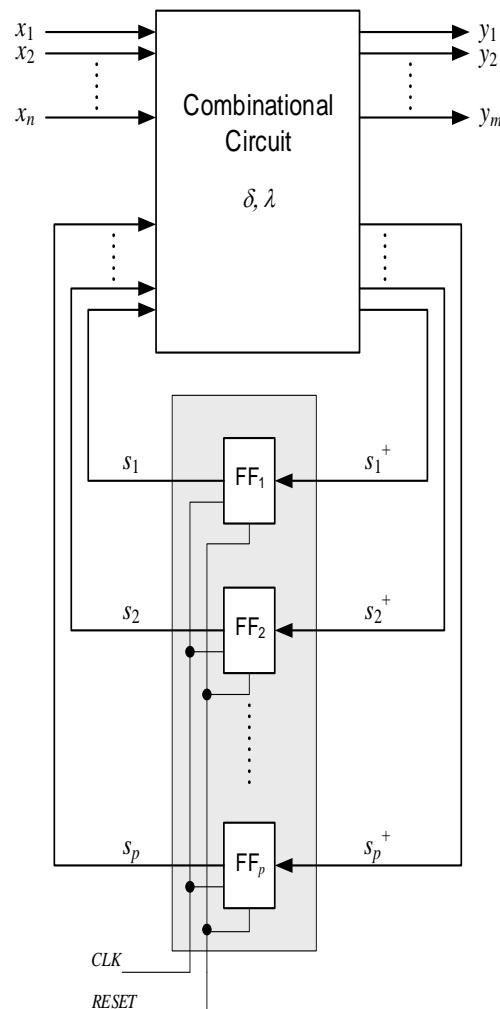
$$\forall x \in X, \forall s \in S : |\delta(s, x)| \geq 1$$

→ The following is always true for deterministic complete automata:

$$\forall x \in X: \forall s \in S : |\delta(s, x)| = 1$$

- ❑ We limit ourselves in this class to deterministic and complete automata
  - Non-complete automata model incomplete specifications
  - Non-deterministic automata are used in analysis and verification of sequential circuits

# Mealy Automata – Circuit Structure



- Finite State Machine = Technical realization of automata
- The structure is known as Huffman Normal Form
- Symbols ( $X, Y, S$ ) used for modeling must first be binary coded for a logic implementation:
  - $x_1 \dots x_n$  binary input variables
  - $y_1 \dots y_m$  binary output variables
  - $s_1 \dots s_p$  binary state variables  $s_1^+ \dots s_p^+$  next state

# Moore Automata

Definition: A Moore automaton is a tuple  $A=(X, Y, S, \delta, \mu, s_I)$  where

$X$ : finite , non-empty set of input symbols

$Y$ : finite, non-empty set of output symbols

$S$ : finite, non-empty set of state symbols

$\delta: S \times X \rightarrow S$  state transition function

$\mu: S \rightarrow Y$  output function

$s_I$ : initial state

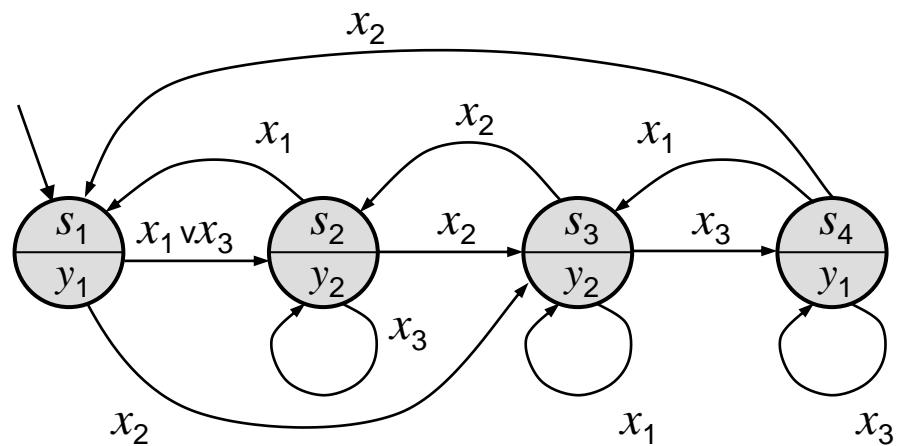
## ■ Difference between Mealy and Moore automata

- Moore Automata: The output  $Y$  depends only on the current state, not the input
- The output function is also known as state marking function

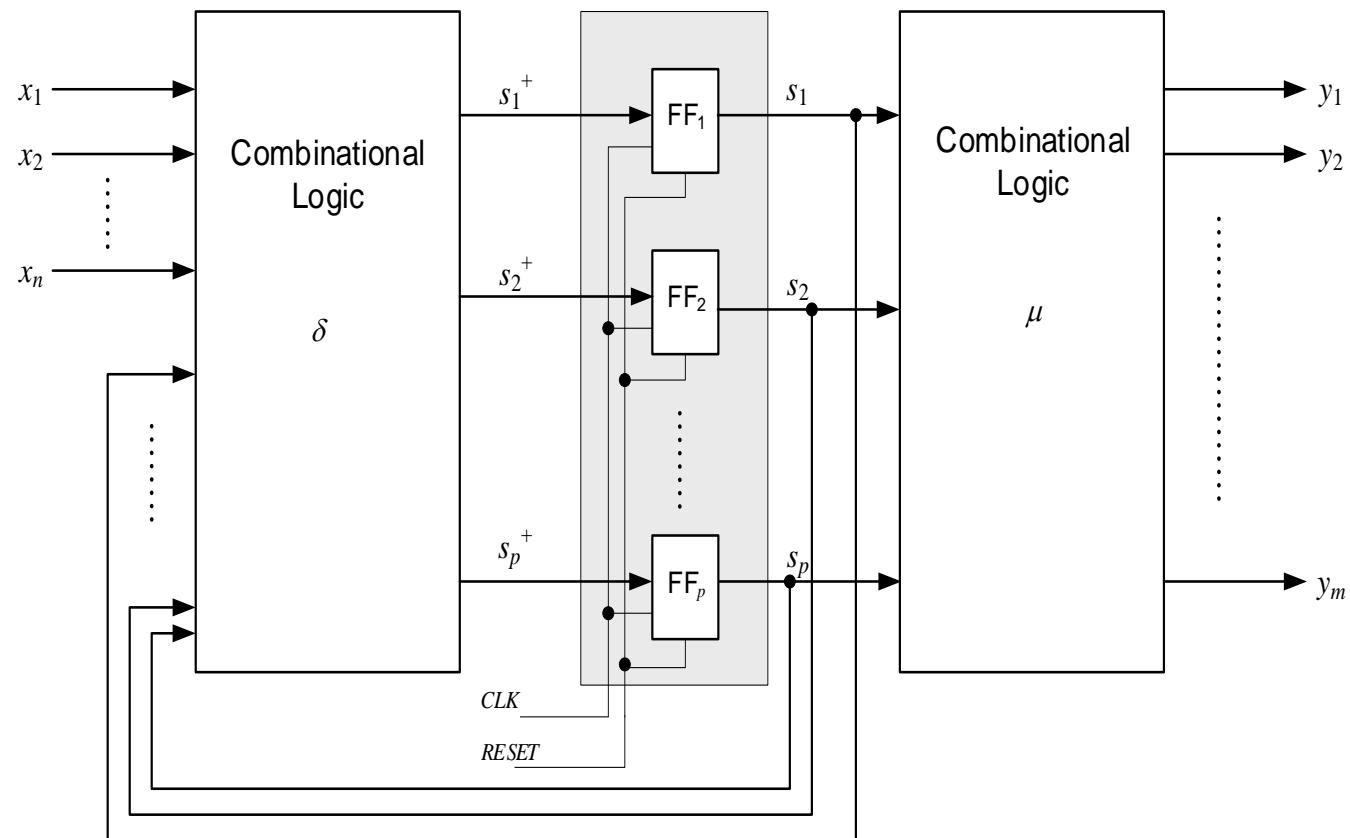
# Moore – State Transition Table/Graph

$$X = \{x_1, x_2, x_3\}, \quad Y = \{y_1, y_2\}, \quad S = \{s_1, s_2, s_3, s_4\}, \quad s_I = s_1$$

	$\mu$	$\delta$		
		$x_1$	$x_2$	$x_3$
$s_I = s_1$	$y_1$	$s_2$	$s_3$	$s_2$
$s_2$	$y_2$	$s_1$	$s_3$	$s_2$
$s_3$	$y_2$	$s_3$	$s_2$	$s_4$
$s_4$	$y_1$	$s_3$	$s_1$	$s_4$



# Moore Automata - Structure



# Design of Sequential Circuits

❑ The design of synchronous sequential circuits is done in the following steps:

1. Specify the automaton using state transition table/graph
  - If needed, minimize the number of states (more on this later)
2. Compute binary coding of the inputs and outputs  
Compute the number of **memory cells** needed to store the states
3. Specify the combinational circuits for the functions  $\delta$  and  $\lambda$  resp.  $\mu$  from the state transition table:  $S^+ = \delta(X, S)$  and  $Y = \lambda(X, S)$  resp.  $Y = \mu(S)$
4. Design of the combination functions as 2-level or multi-level logic

❑ The minimization of gates and flip flops is a complex problem

- In the next, we will learn
  - Automata equivalency and state minimization
  - Coding

# Case Study

- A "mean" vending machine
  - The machine expects in its initial state a \$1
  - The user can then choose between Coke and 7up, or press the R-button to get his money back
  - If the user inserts \$2 (1 additional \$) the automat returns to its initial state



# A "mean" Vending Machine

## □ Specification with automata: input , output

Input:  $X = \{\$, C, 7, R\}$

- \$ ... Money (\$1) inserted
- C ... Coke selected
- 7 ... 7up selected
- R ... „Refund“ selected

Output  $Y = \{GC, G7, M, \emptyset\}$

- GC ... Give Coke
- G7 ... Give 7up
- M ... Return 1\$
- $\emptyset$  ... Do nothing

# A "mean" Vending Machine

- Specification as Mealy automaton: states, state transition table

States  $S = \{I, H\}$

- $I$  ... Initial state
- $H$  ... Automaton has \$1

State transition table

- Specifies  $\delta$  and  $\lambda$
- Initial state  $s_I = I$

$\delta / \lambda$	\$	C	7	R
$I$	$H / \emptyset$	$I / \emptyset$	$I / \emptyset$	$I / \emptyset$
$H$	$I / \emptyset$	$I / GC$	$I / G7$	$I / M$

# A "mean" Vending Machine

- Binary coding of inputs, outputs and states
  - 4 input symbols can be represented with 2 bits  $(x_1, x_0)$  code, for example
$$X = \{\$, C, 7, R\} \rightarrow (x_1, x_0) = \{00, 01, 10, 11\}$$
  - 4 input symbols can be represented with 2 bits  $(y_1, y_0)$  code, for example
$$Y = \{GC, G7, M, \emptyset\} \rightarrow (y_1, y_0) = \{00, 01, 10, 11\}$$
  - 2 states can be represented with just 1 Bit ( $S$ )
$$S = \{I, H\} \rightarrow (s) = \{0, 1\}$$

# A "mean" Vending Machine

- Specification of  $\delta$  and  $\lambda$  using state transition table
  - Devise truth table of  $S^+ = \delta(X, S)$  and  $Y = \lambda(X, S)$  from state transition table

State transition table with binary codes

$\delta / \lambda$	00	01	10	11
0	1 / 11	0 / 11	0 / 11	0 / 11
1	0 / 11	0 / 00	0 / 01	0 / 10

Truth table

$s$	$x_I$	$x_0$	$s^+$	$y_I$	$y_0$
0	0	0	1	1	1
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	0	1	1
1	0	0	0	1	1
1	0	1	0	0	0
1	1	0	0	0	1
1	1	1	0	1	0

# A "mean" Vending Machine

- Compute the combinational functions for  $\delta$  and  $\lambda$

$s$	$x_1$	$x_0$	$s+$	$y_1$	$y_0$
0	0	0	1	1	1
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	0	1	1
1	0	0	0	1	1
1	0	1	0	0	0
1	1	0	0	0	1
1	1	1	0	1	0

$$s^+ = \bar{s} \bar{x}_1 \bar{x}_0$$

$$y_1 = \bar{s} + x_1 x_0 + \bar{x}_1 \bar{x}_0$$

$$y_0 = \bar{s} + \bar{x}_0$$

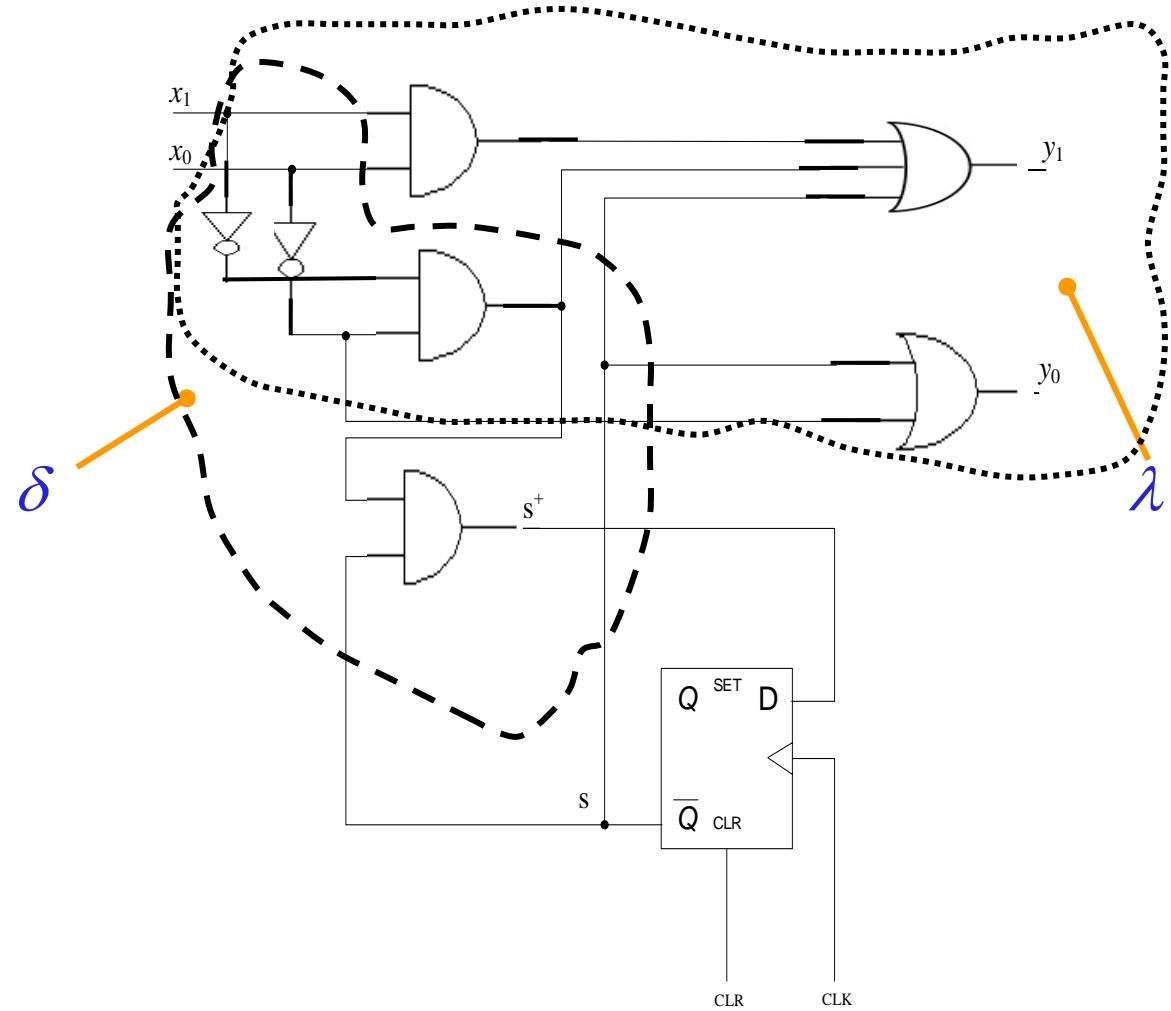
$x_1$				$ _s$
0	0	0	0	
0	0	0	1	

$x_1$				$ _s$
0	1	0	1	
1	1	1	1	

$x_1$				$ _s$
1	0	0	1	
1	1	1	1	

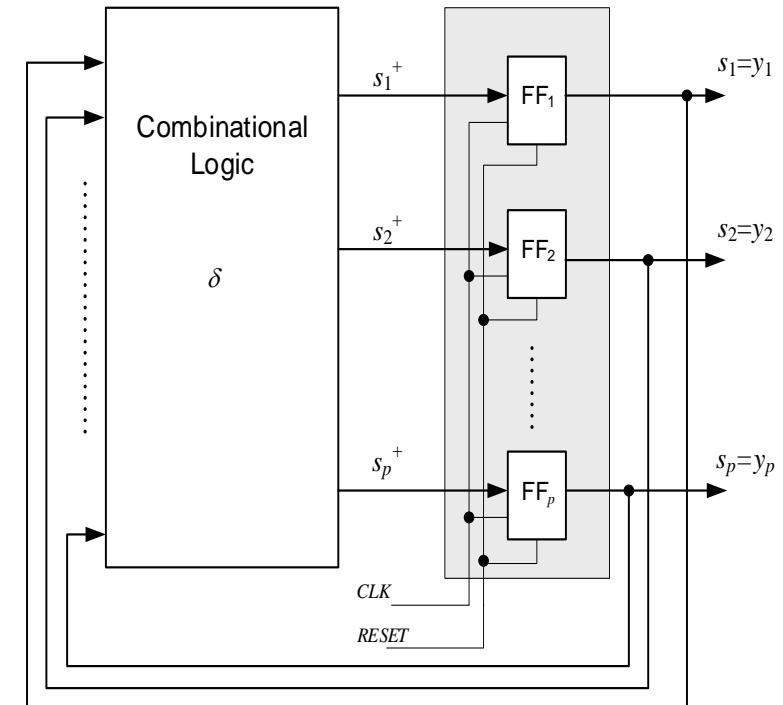
# A „mean" Vending Machine

□ Control Path



# Case Study: Synchronous Counter

- Synchronous counters can be built using Moore automata
  - No input, i.e.  $X=\{\}$
  - No output function  $\mu$ , i.e. the binary codes of the states are the identical to the outputs

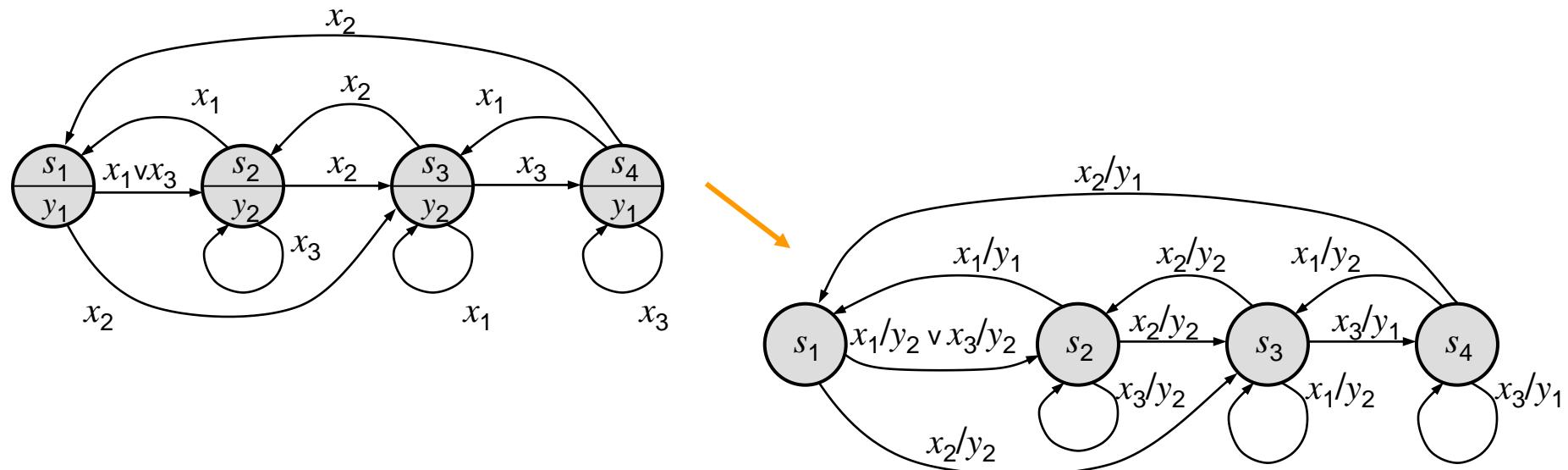


# Conversion - Moore to Mealy

- A Moore-automaton is converted in an equivalent Mealy-automaton, by marking transitions with the output symbol of the target state

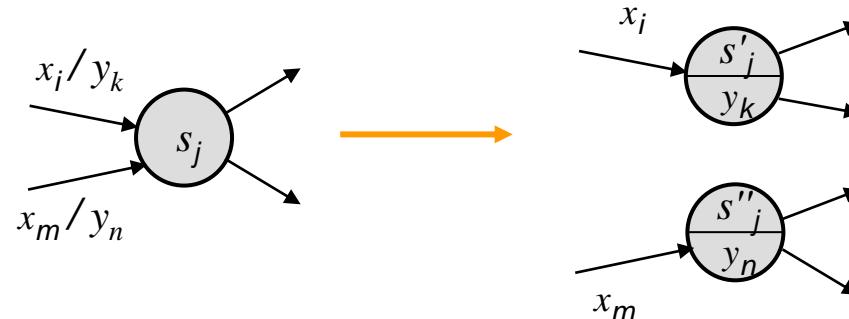


- Example



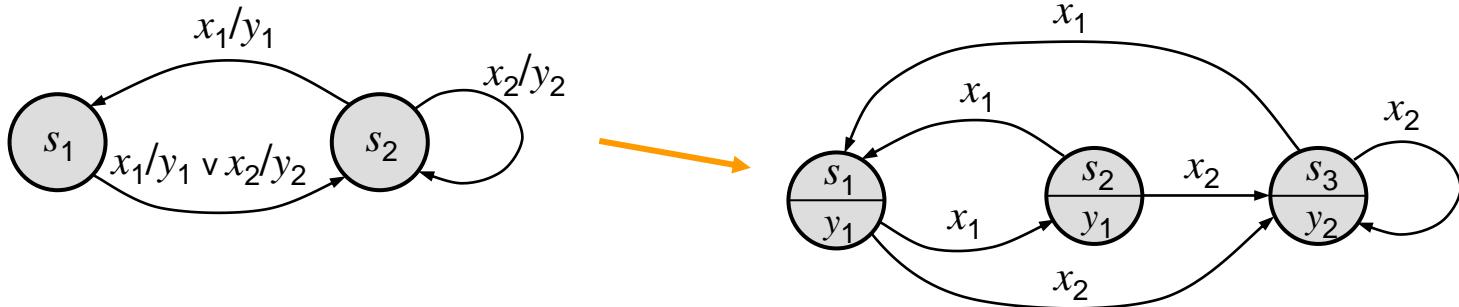
# Conversion - Mealy to Moore

- For each state  $s$  of the Mealy automaton, insert as many states as there are incident edges to  $s$ . Label the new states with the output variables of the corresponding incident transitions



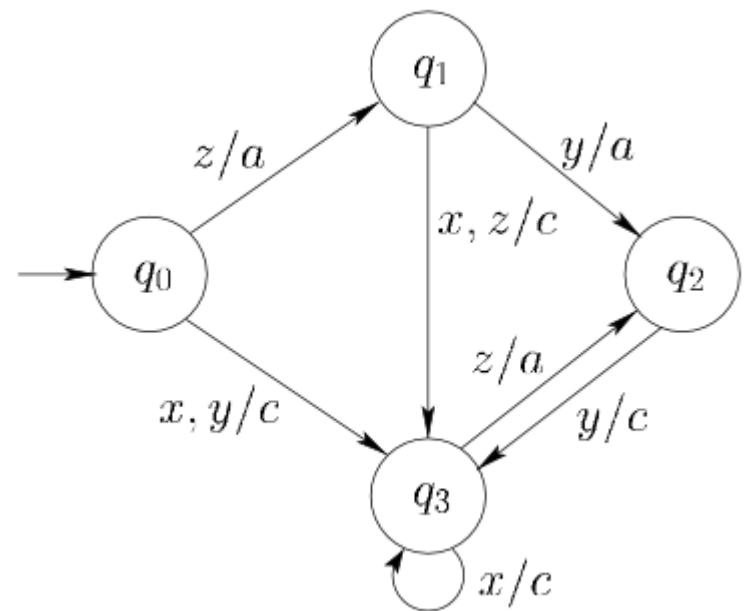
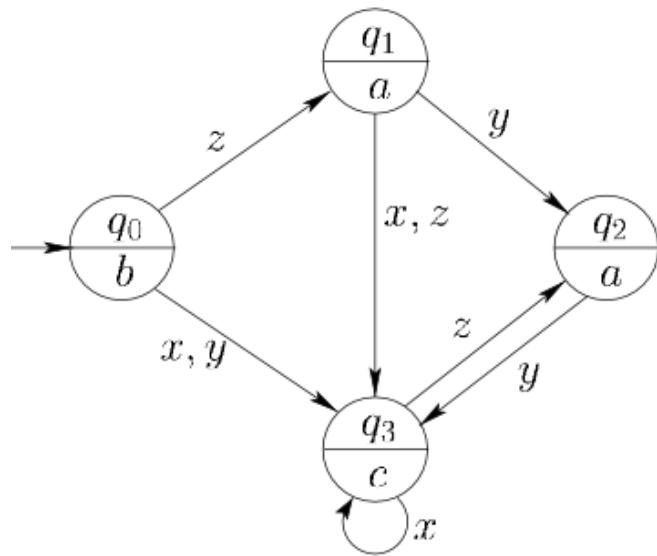
- Resulting automaton can be reduced

- Example



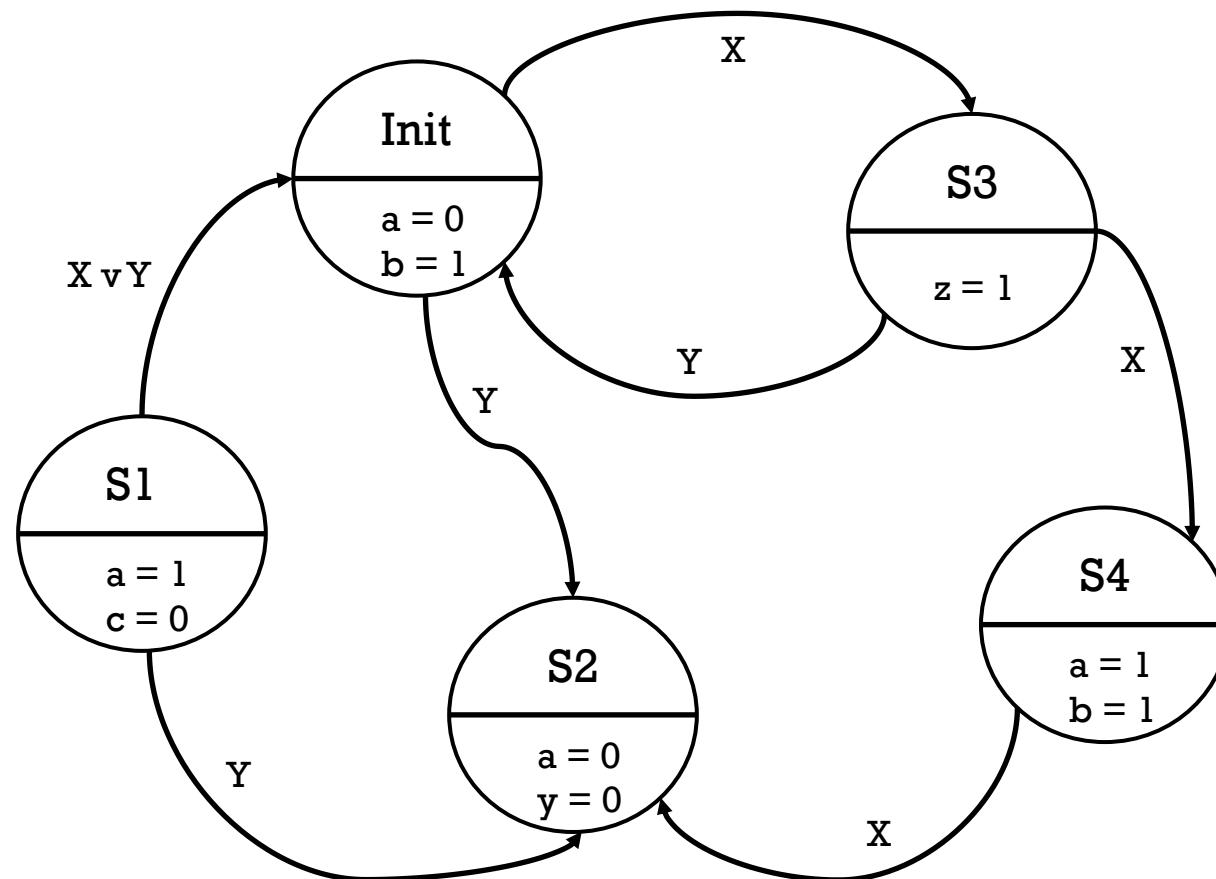
# Conversion - Moore to Mealy

- Convert the following Moore automaton into a Mealy automaton



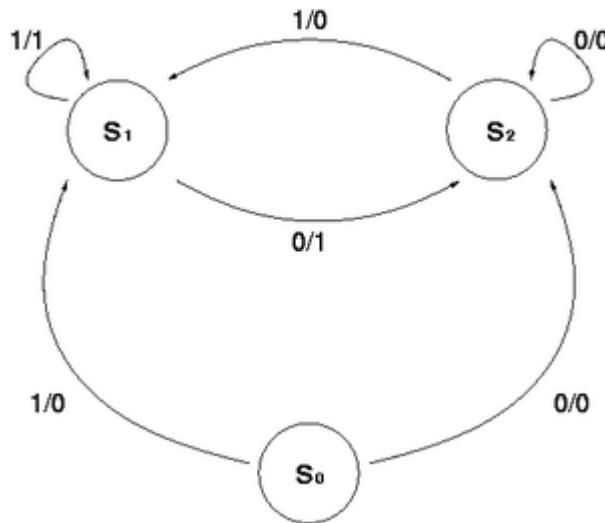
# Conversion - Moore to Mealy

- Convert the following Moore automaton into a Mealy automaton



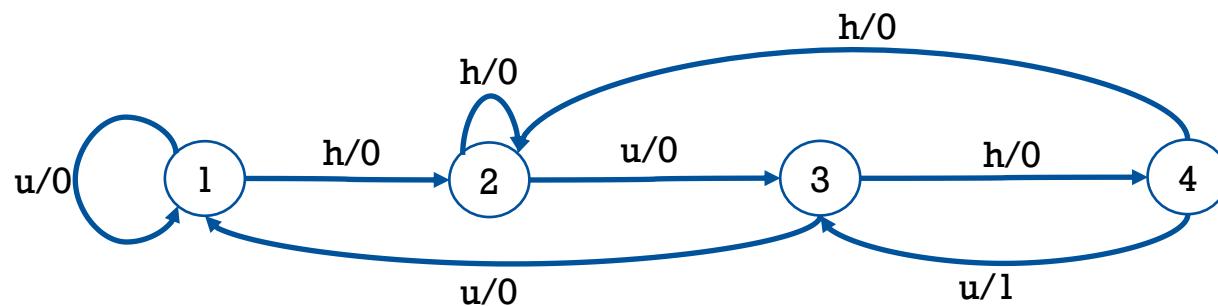
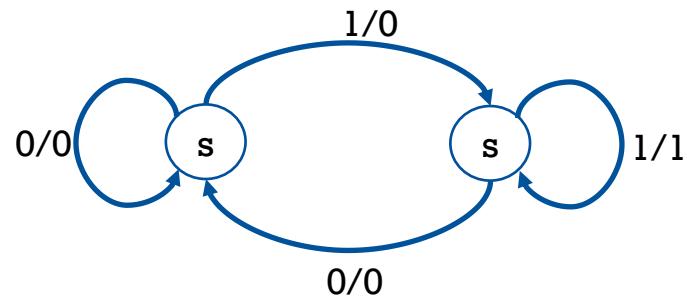
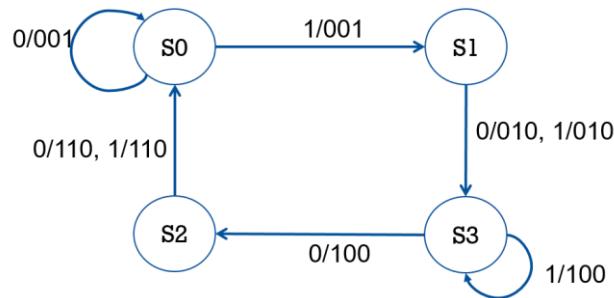
# Conversion - Moore to Mealy

- Convert the following Mealy automaton into a Moore automaton



# Conversion - Moore to Mealy

- Convert the following Mealy automaton into a Moore automaton



# Using SR, JK and T-FFs in FSM

$s$	$x_1$	$x_0$	$s+$	$y_1$	$y_0$	$S$	$R$	$J$	$K$	$T$
0	0	0	1	1	1					
0	0	1	0	1	1					
0	1	0	0	1	1					
0	1	1	0	1	1					
1	0	0	0	1	1					
1	0	1	0	0	0					
1	1	0	0	0	1					
1	1	1	0	1	0					

# Using SR, JK and T-FFs in FSM

$Q_3\ Q_2\ Q_1\ Q_0$	$Q_3^+\ Q_2^+\ Q_1^+\ Q_0^+$	$S$	$R$	$J$	$K$	$T$
0 0 0 0	0 0 0 1					
0 0 0 1	0 0 1 0					
0 0 1 0	0 0 1 1					
0 0 1 1	0 1 0 0					
0 1 0 0	0 1 0 1					
0 1 0 1	0 1 1 0					
0 1 1 0	0 1 1 1					
0 1 1 1	1 0 0 0					
1 0 0 0	1 0 0 1					
1 0 0 1	1 0 1 0					
1 0 1 0	1 0 1 1					
1 0 1 1	1 1 0 0					
1 1 0 0	1 1 0 1					
1 1 0 1	1 1 1 0					
1 1 1 0	1 1 1 1					
1 1 1 1	0 0 0 0					

# Comparison Mealy-Moore Automata

## □ Modelling

- Moore and Mealy are equivalent: have the same modelling power

## □ Implementation

- Moore automata need more states in general than Mealy automata, which leads to higher hardware costs.
- Output of Moore automata change only upon state transition; Output of Mealy-automata change with inputs
  - Moore-automata are more robust to short impulses of the inputs (glitches) and usually preferred to Mealy in hardware design
  - Mealy-automata react faster to changes at the input



Herbert Wertheim  
College of Engineering  
UNIVERSITY *of* FLORIDA