UF Herbert Wertheim College of Engineering
UNIVERSITY of FLORIDA

Digital Logic And Computing Systems

Lecture 06 – Automata – Controlpath - Cont

Dr. Christophe Bobda
EEL3701C Fall 2025

Agenda

- Introduction and Motivation
- Sequential Circuits
- Mealy- and Moore Automata
- Design of Sequential Circuits
- Case studies

Controllers Design

- So far, we have discussed combinational circuits

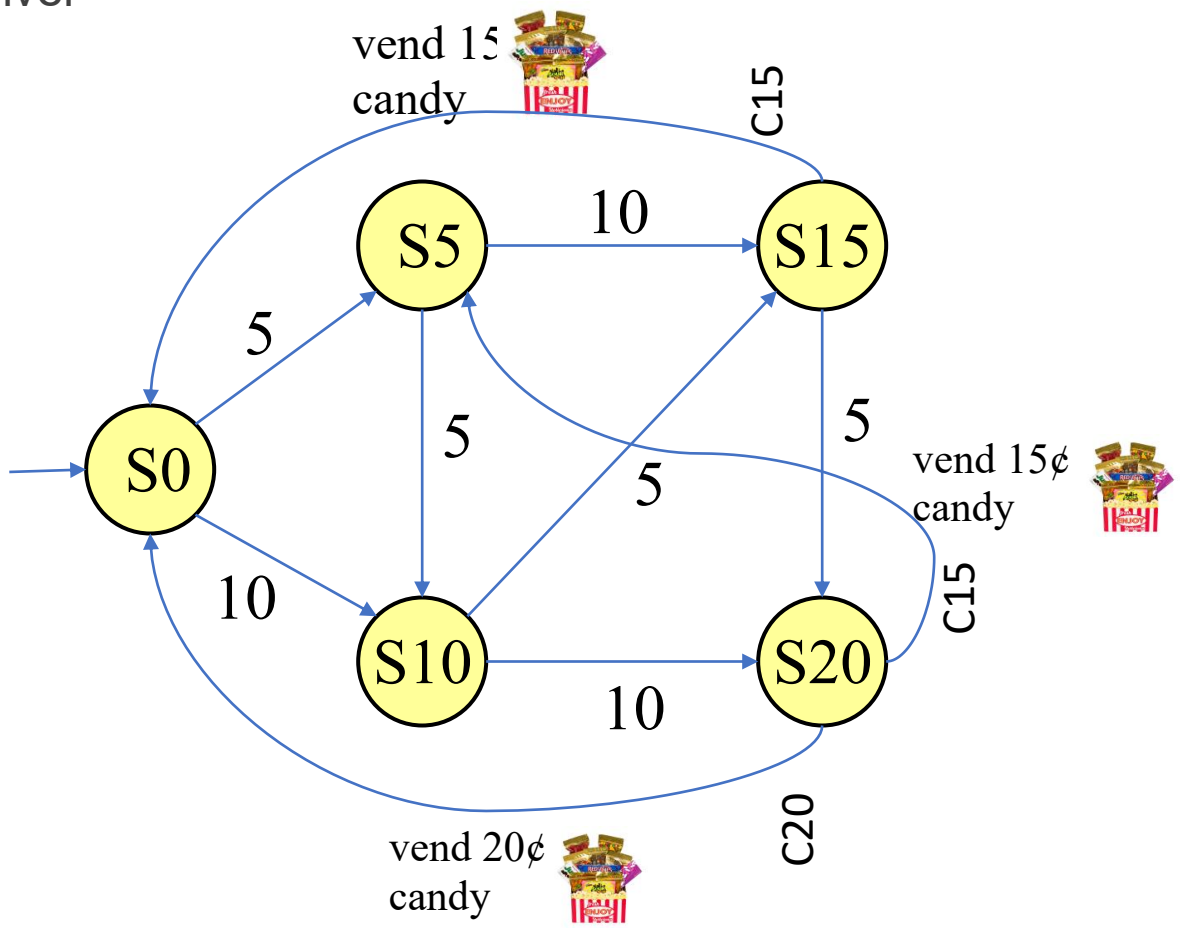
- Ex. Coffee dispenser
  - Select a coffee type (expresso, cappuccino, regular, etc…) → 3 buttons
  - Select size (small, medium, large) → 3 buttons
  - Select sugar option → 1 – 2 buttons
  - Select milk option → 1-2 buttons
  - Pay (cash or card) → ?? 4 – 6?

- A combinational circuit would require
  - All inputs at once → not usable
  - Too many buttons for all possible options → too expensive

- Solution: Sequential Circuits
  - Multipurpose inputs → less expensive
  - One-at-a-time entry → convenient, user friendly

Sequential Circuits - Modeling with Automata

- Design a controller for a vending machine that can deliver
  - 15-Ct candy
  - 20-Ct candy
  - The vending machine accepts
    - 5 ct
    - 10 ct
  - And delivers the 15-ct or 20-ct candy, based On the user's choice

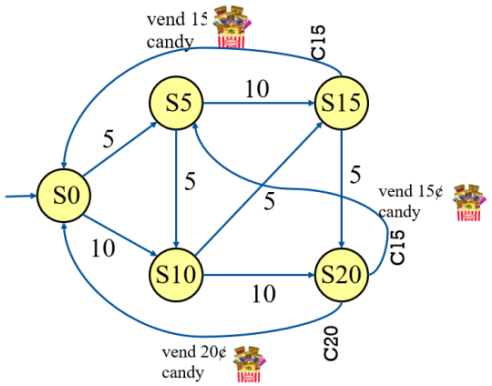


Sequential Circuits

- 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).

Mealy-Automata



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

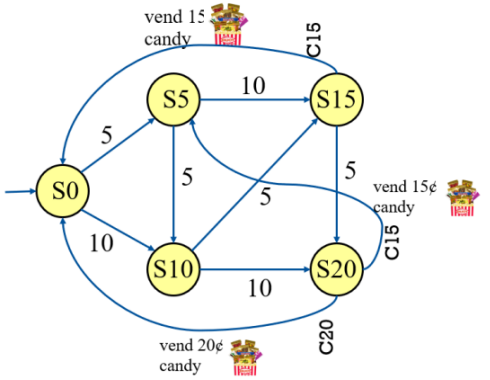
X: finite, non-empty set of input symbols
Y: finite, non-empty set of output symbols
S: finite, non-empty set of states

δ: $S \times X \rightarrow S$ State transition function
λ: $S \times X \rightarrow Y$ Output function
s_I: Initial state

State set finite → 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



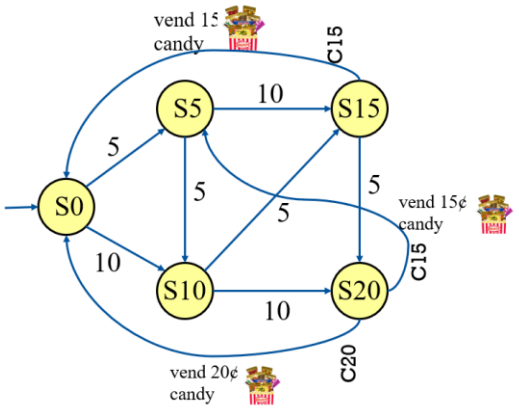
- A state transition table is a table in which the rows represent states and columns represent input symbols.
  - An entry ($\delta(s,x)$ / $\lambda(s,x)$) captures the transition from state $s \in S$ with input symbol $x \in X$ to state $\delta(s,x)$ with output $\lambda(s,x)$

- Example

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

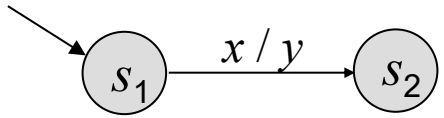
δ / λ	x_1	x_2	x_3
$s_I = s_1$	s_2 / y_1	s_3 / y_2	s_2 / y_1
s_2	s_1 / y_2	s_3 / y_2	s_2 / y_1
s_3	s_3 / y_1	s_2 / y_1	s_4 / y_1
s_4	s_3 / y_2	s_1 / y_2	s_4 / y_2

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

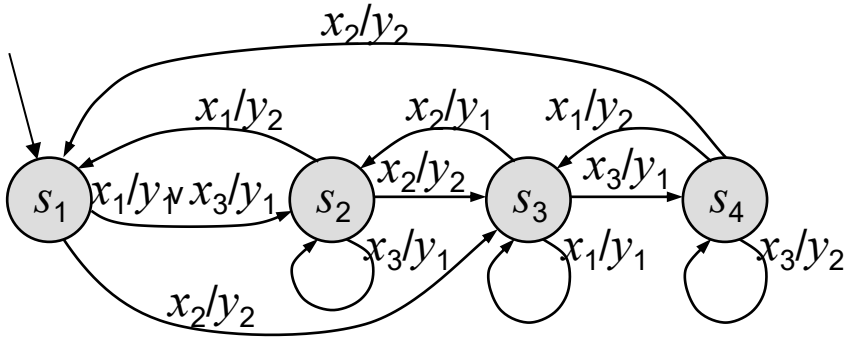
- Example



means $\delta(s_1, x) = s_2, \lambda(s_1, x) = y, \; s_I = s_1$

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

δ / λ	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:

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

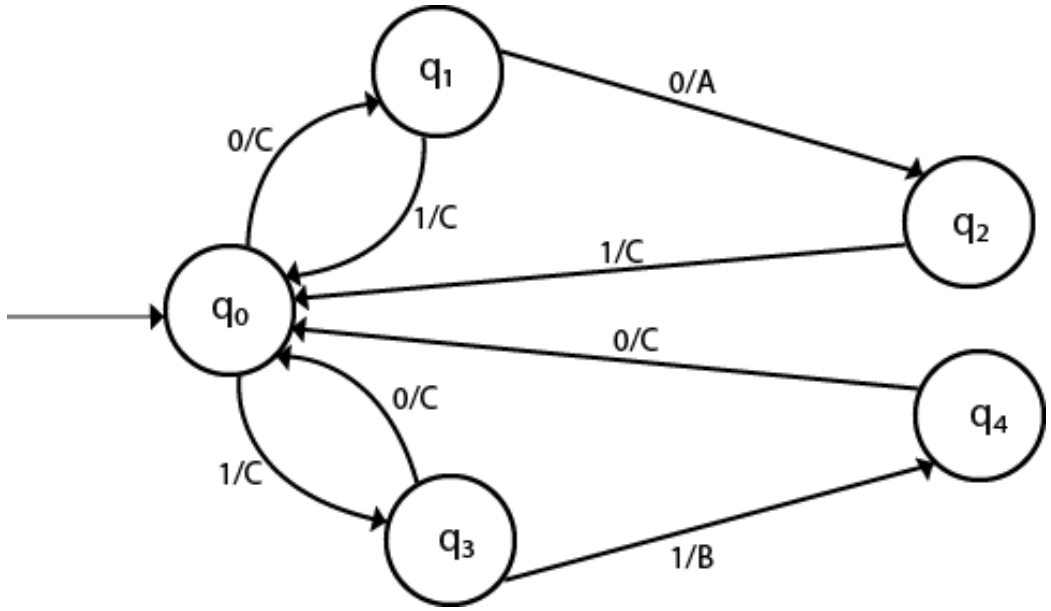
$\rightarrow$ The following is always true for deterministic complete automata:

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

- 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

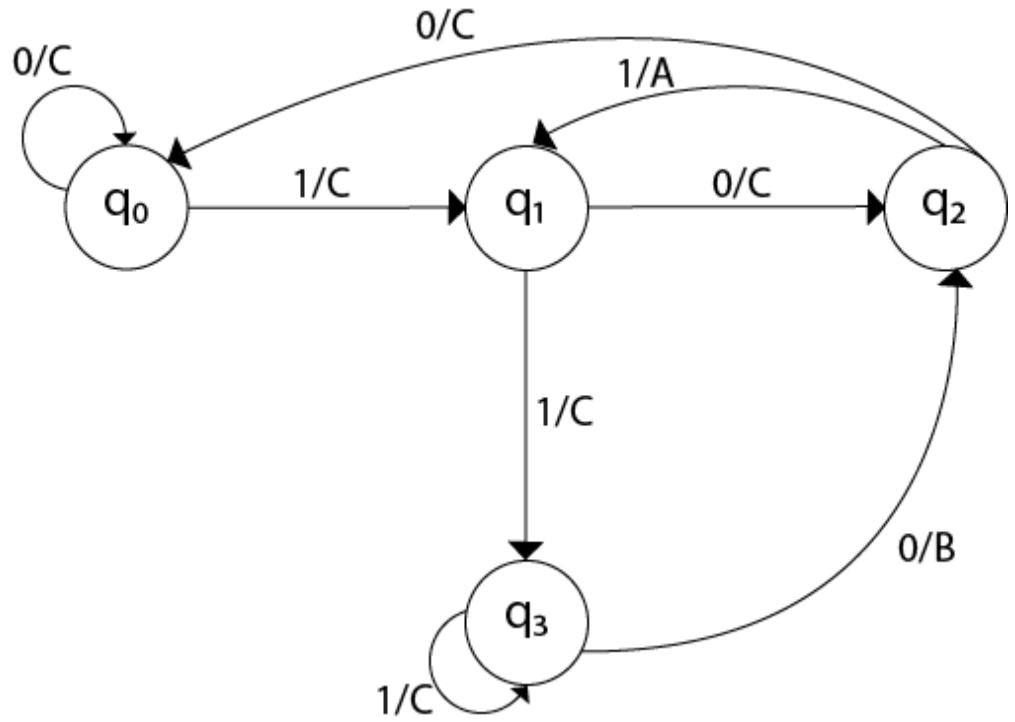
Mealy Automata Example 1

- Design a mealy machine that scans sequence of input of 0 and 1 and generates output 'A' if the input string terminates in 00, output 'B' if the string terminates in 11, and output 'C' otherwise.

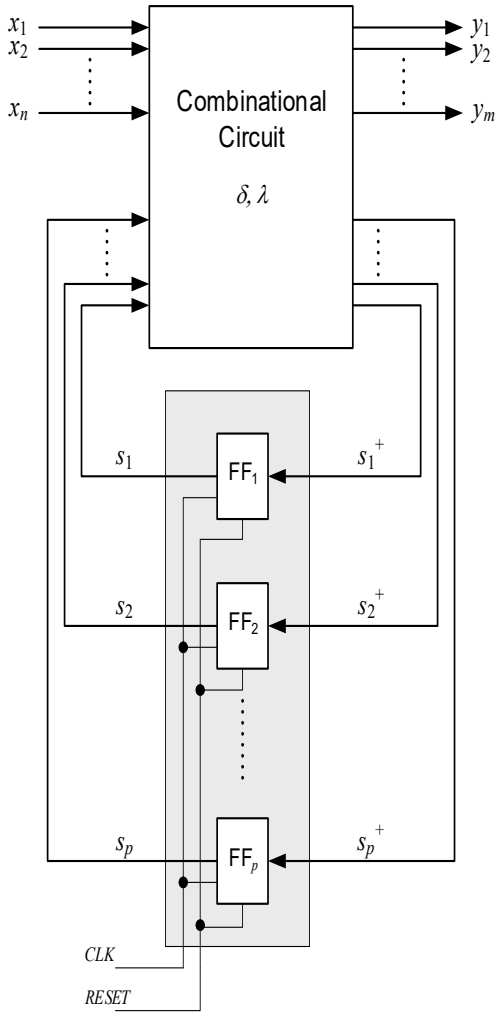


Mealy Automata Example 2

- Design a Mealy machine for a binary input sequence such that if it has a substring 101, the machine output A, if the input has substring 110, it outputs B otherwise it outputs C.



Mealy Automata – Finite State Machines



- Finite Sate 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:
  - $x_1 \ldots x_n$ binary input variables
  - $y_1 \ldots y_m$ binary output variables
  - $s_1 \ldots s_p$ binary state variables
    $s_1^+ \ldots s_p^+$ next state

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 δ and λ resp.
     - μ 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

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

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

Ouputs $Y = \{GC, G7, M, \varnothing\}$

- GC ... Give Coke
- $G7$ ... Give 7up
- M ... Return 1$
- $\varnothing$ ... 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 δ and λ
- Initial state $s_I = I$

δ / λ	$	C	7	R
I	$H / \varnothing$	$I / \varnothing$	$I / \varnothing$	$I / \varnothing$
H	$I / \varnothing$	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
    $Y = \{GC,\ G7,\ M,\ \varnothing\} \rightarrow (y_1,y_0) = \{00,\ 01,\ 10,\ 11\}$
  - 2 states can be represented with just 1 Bit (s)
    $S = \{I,\ H\} \rightarrow (s)\ = \{0,\ 1\}$

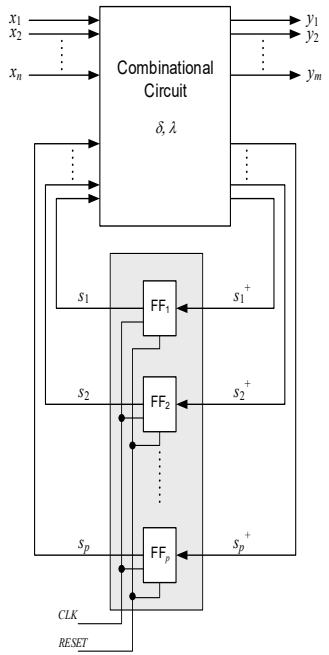
A "mean" Vending Machine

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

δ / λ	$	C	7	R
I	H / ∅	I / ∅	I / ∅	I / ∅
H	I / ∅	I / GC	I / G7	I / M

State transition table with binary codes

δ / λ	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_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

A "mean" Vending Machine

- Compute the combinational functions for δ and λ

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

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

x_1 \ x_0	00	01	11	10
s = 0	0	0	0	0
s = 1	0	0	0	1

$$y_1 = \bar{s} + x_1x_0 + \bar{x}_1\bar{x}_0$$

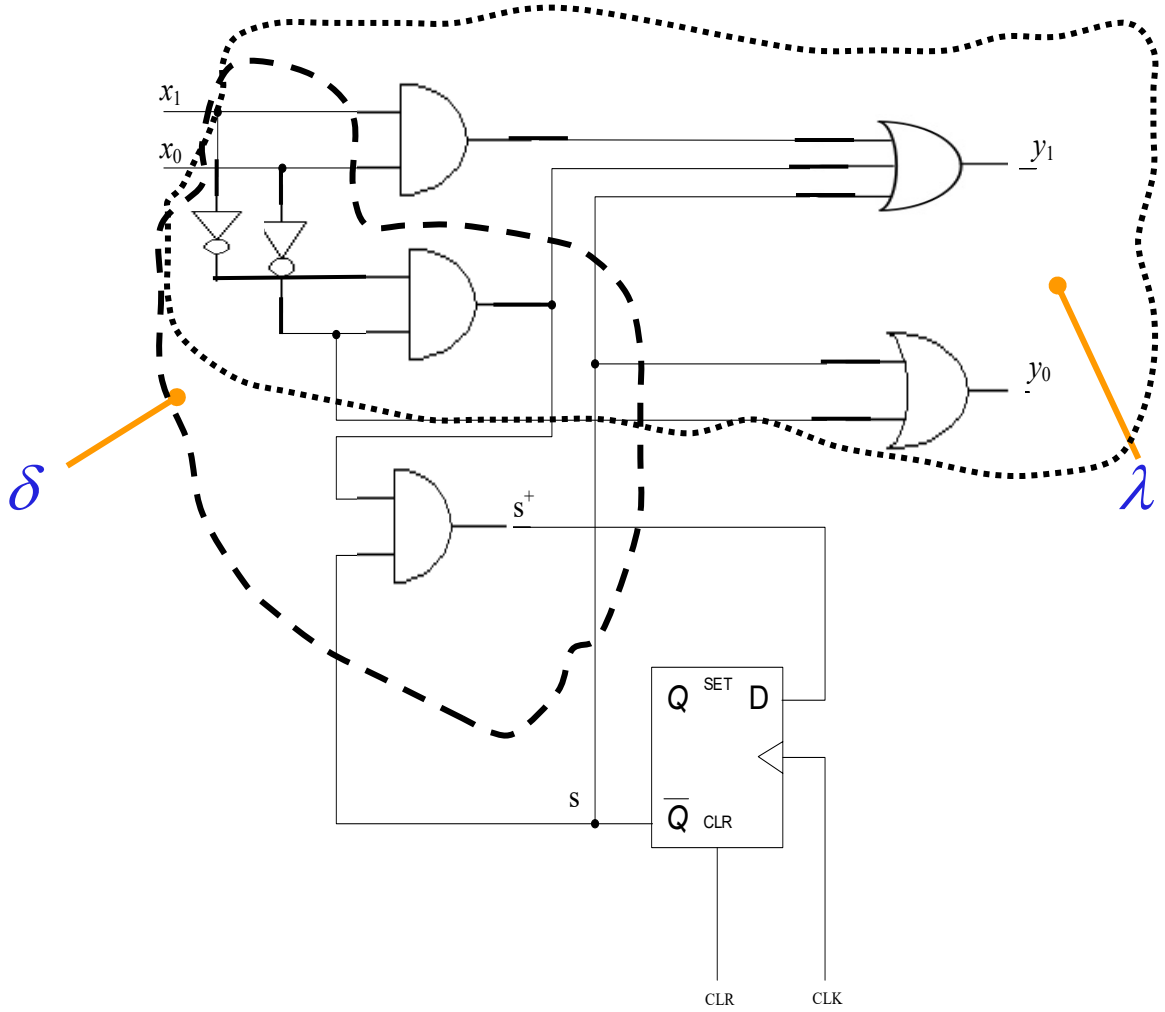
x_1 \ x_0	00	01	11	10
s = 0	1	0	1	1
s = 1	1	1	1	1

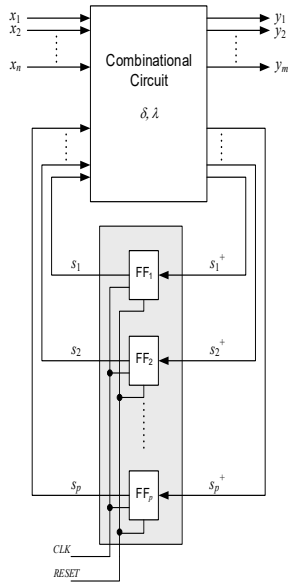
$$y_0 = \bar{s} + \bar{x}_0$$

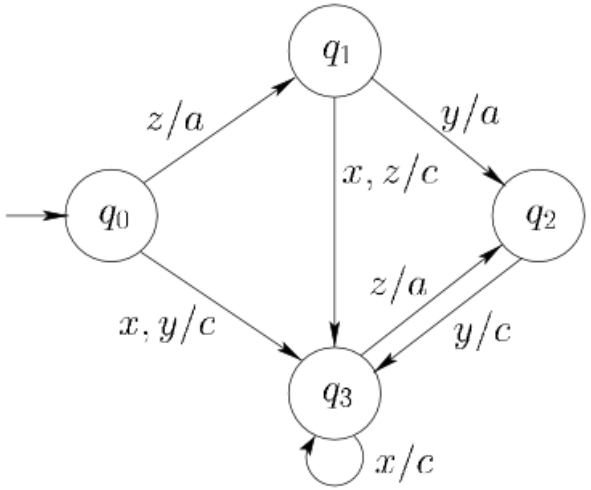
x_1 \ x_0	00	01	11	10
s = 0	1	0	0	1
s = 1	1	1	1	1

A „mean" Vending Machine

- FSM








```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY FSA_MEALY IS

PORT( clk, reset: IN std_logic;
              x,yz: IN std_logic;
                a,c: OUT std_logic);
END FSA_MEALY F;

ARCHITECTURE specific OF FSA_MEALY
TYPE state_type IS (Q0, Q1, Q2, Q3) ;
SIGNAL current_state, next_state: state_type ;

Comb : PROCESS ( current_state )
BEGIN
   CASE current_state IS
      WHEN Q0 =>
         IF ( z = '1' ) THEN
               next_state <= Q1 ; a <= ‚1‘;
         END IF
         IF ( x = '1‘ or y = '1‘) THEN
               next_state <= Q3 ; c <= ‚1‘;
         END IF

      WHEN Q1 =>
         IF ( y = '1' ) THEN
               next_state <= Q2 ; a <= ‚1‘;
         END IF
         IF ( x = '1‘ or y = '1‘) THEN
               next_state <= Q3 ; c <= ‚1‘;
         END IF
        ...
   END CASE;
END PROCESS Comb;
```

Transitions

Outputs in transitions

```
...
Synch : PROCESS ( CLK, RESET )
BEGIN

   IF ( RESET = '1' ) THEN -- asynchronous reset
               CURRENT_STATE <= Q0;
   ELSIF ( CLK'EVENT and CLK = '1' ) THEN
               CURRENT_STATE <= NEXT_STATE;
   END IF;

END PROCESS Synch;
```

Moore Automata

Definition: A Moore automaton is a tuple $A=(X, Y, S, \delta, \mu, s_1)$ 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

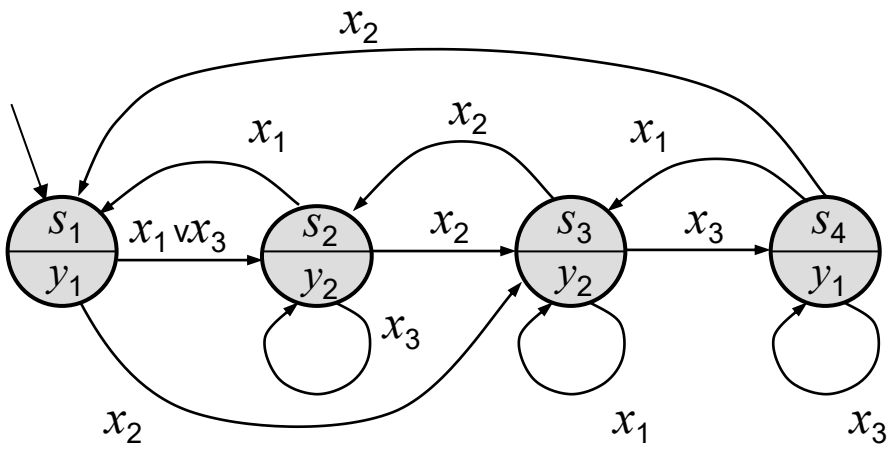
δ: $S \times X \rightarrow S$ state transition function
μ: $S \rightarrow Y$ output function
s_1: initial state

- 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

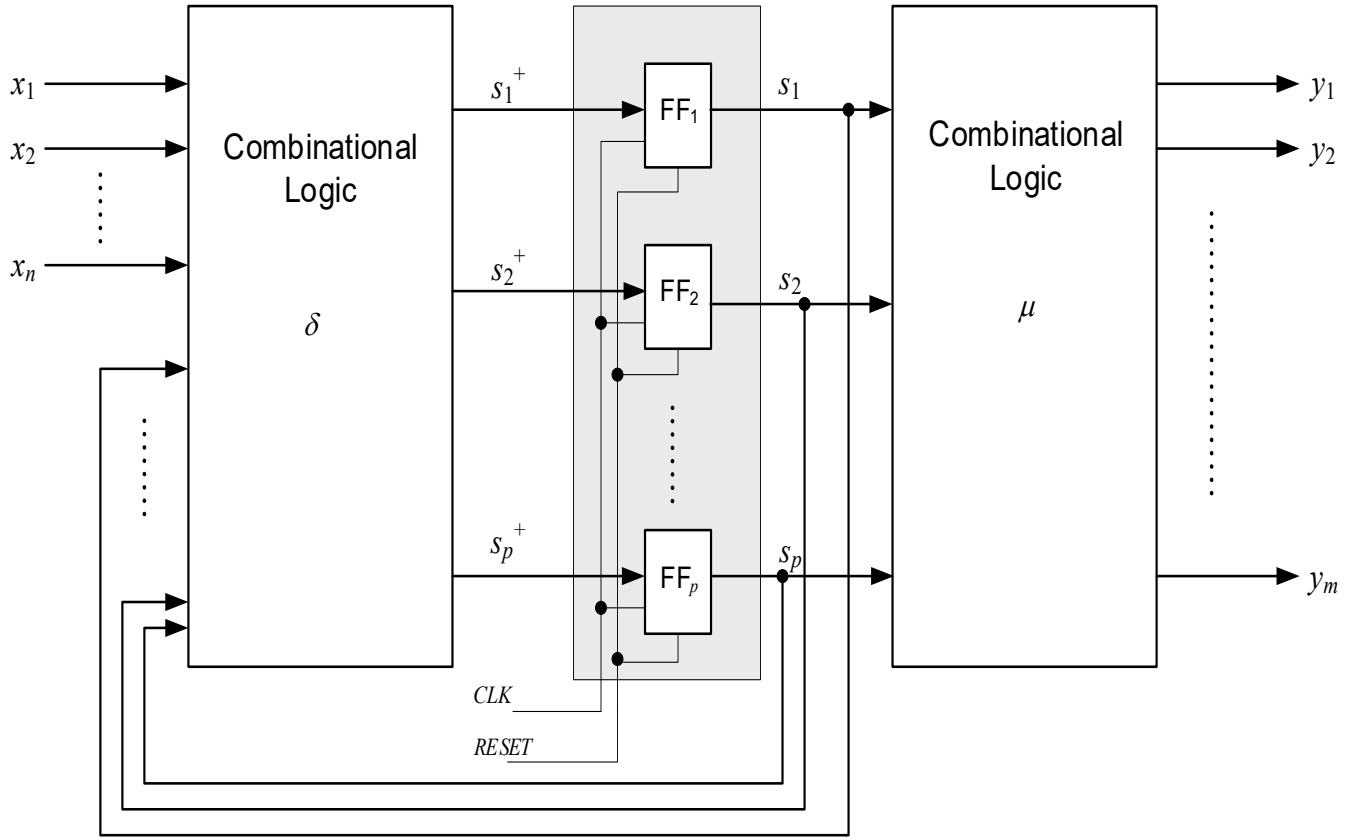
Moore – State Transition Table/Graph

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

	μ	δ		
		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

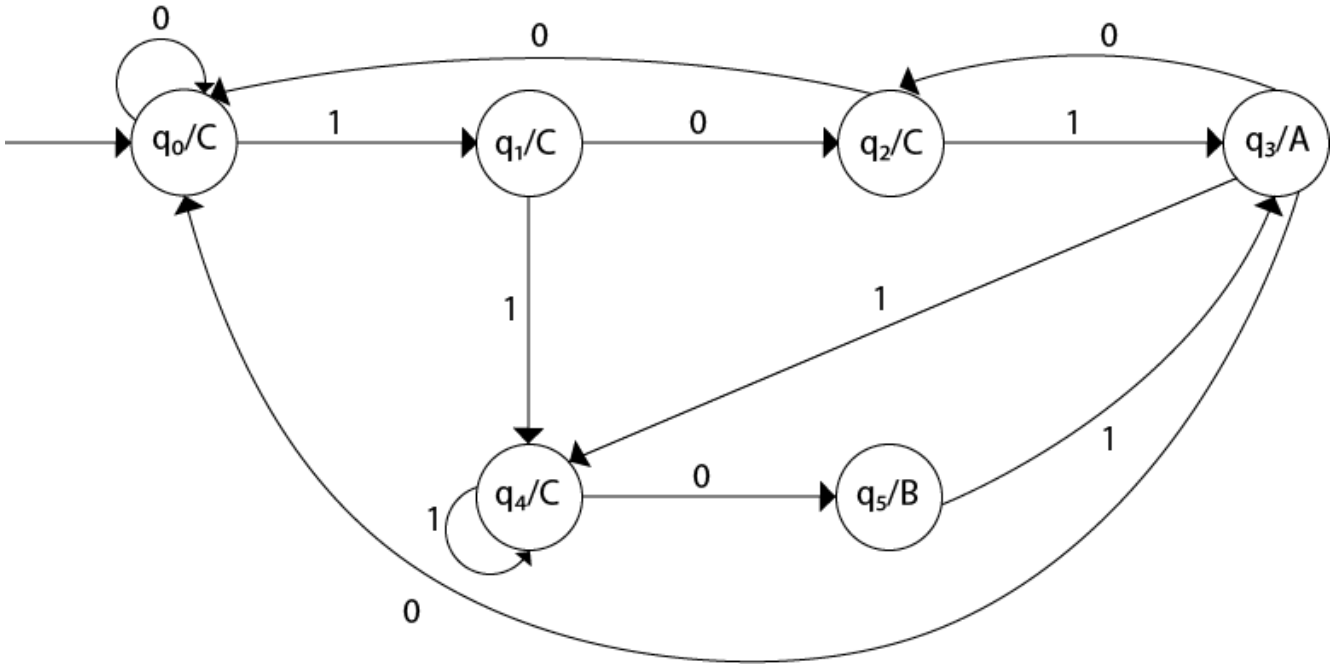


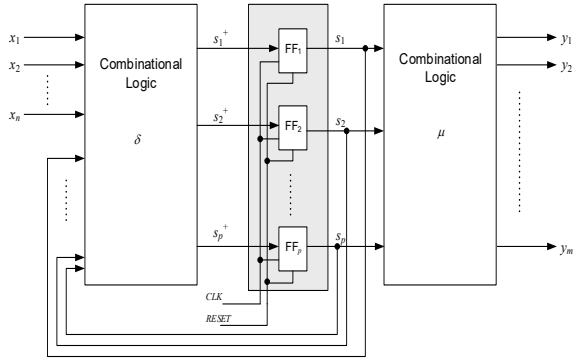
Moore Automata - Structure



Moore Automata Example

- Design a Moore machine for a binary input sequence such that if it has a substring 101, the machine output A, if the input has substring 110, it outputs B otherwise it outputs C.

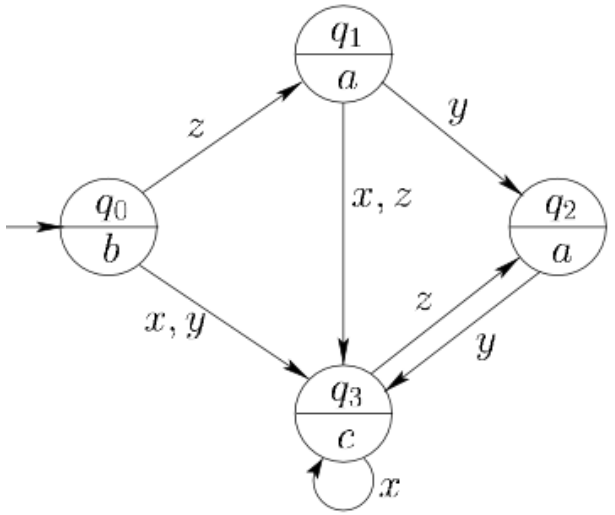





```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY FSA_MEALY IS

PORT( clk, reset: IN std_logic;
              x,yz: IN std_logic;
                a,c: OUT std_logic);
END FSA_MEALY F;

ARCHITECTURE specific OF FSA_MEALY
TYPE state_type IS (Q0, Q1, Q2, Q3) ;
SIGNAL current_state, next_state: state_type ;

Comb : PROCESS ( current_state )
BEGIN
   CASE current_state IS
      WHEN Q0 =>
         a <= ,1‘;
         IF ( z = '1' ) THEN
               next_state <= Q1 ;
         END IF
         IF ( x = '1‘ or y = '1‘) THEN
               next_state <= Q3 ;
         END IF

      WHEN Q1 =>
         a <= ,1‘;
         IF ( y = '1' ) THEN
               next_state <= Q2 ;
         END IF
         IF ( x = '1‘ or z = '1‘) THEN
               next_state <= Q3 ;
         END IF
      ...
```

Outputs in the states independent of transitions

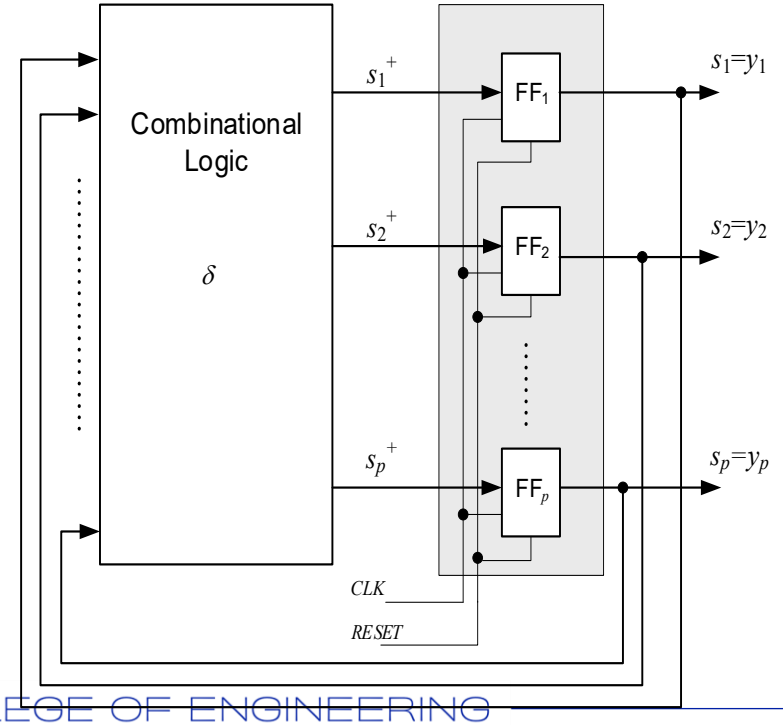
Transitions




```
...
Synch : PROCESS ( CLK, RESET )
BEGIN

   IF ( RESET = '1' ) THEN -- asynchronous reset
            CURRENT_STATE <= S0;
   ELSIF ( CLK'EVENT and CLK = '1' ) THEN
            CURRENT_STATE <= NEXT_STATE;
   END IF;

END PROCESS Synch;
```

Case Study: Synchronous Counter

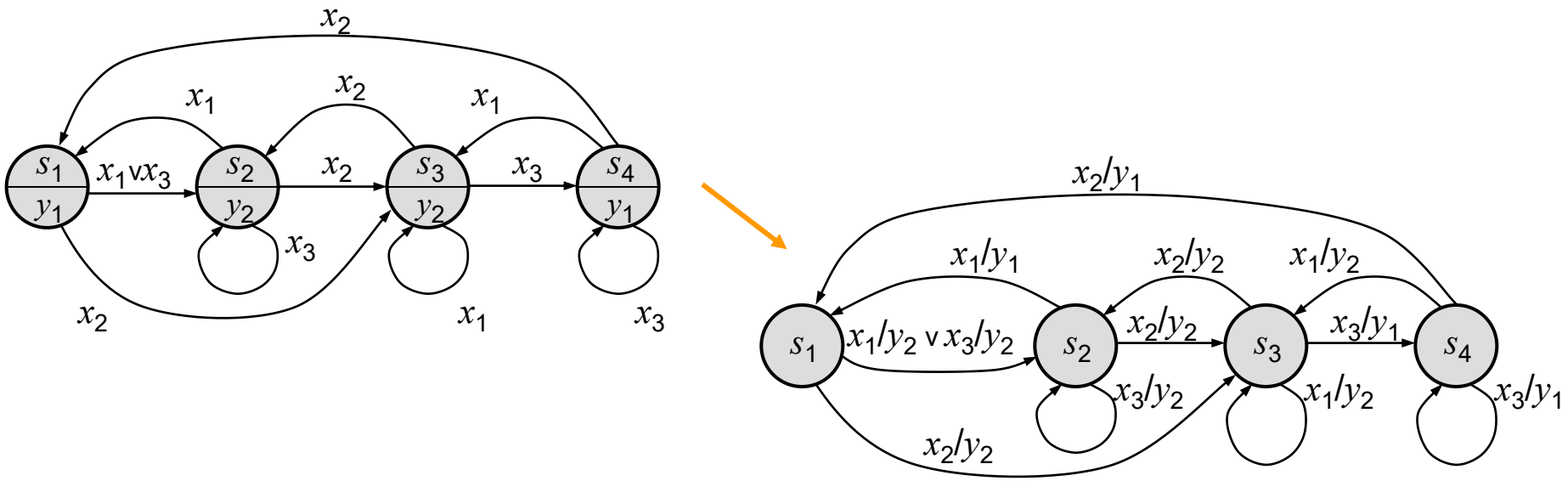
- Synchronous counters can be built using Moore automata
  - No input, i.e. $X=\{\ \}$
  - No output function μ, i.e. the binary codes of the states are the identical to the outputs



Conversion - Moore to Mealy

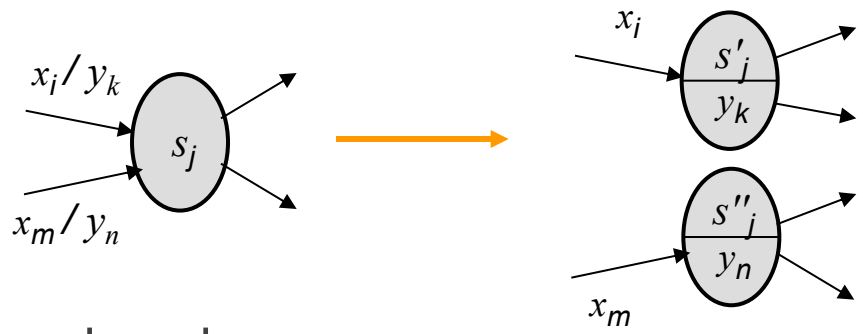
- A Moore-automaton is converted in an equivalent Mealy-automaton, by marking transitions with the output symbol of the target state

- Example



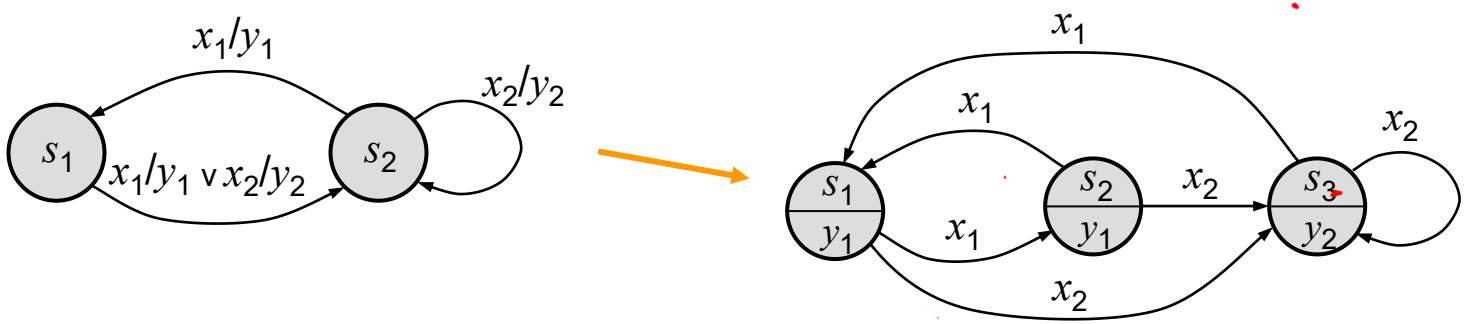
Conversion - Mealy to Moore

- 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



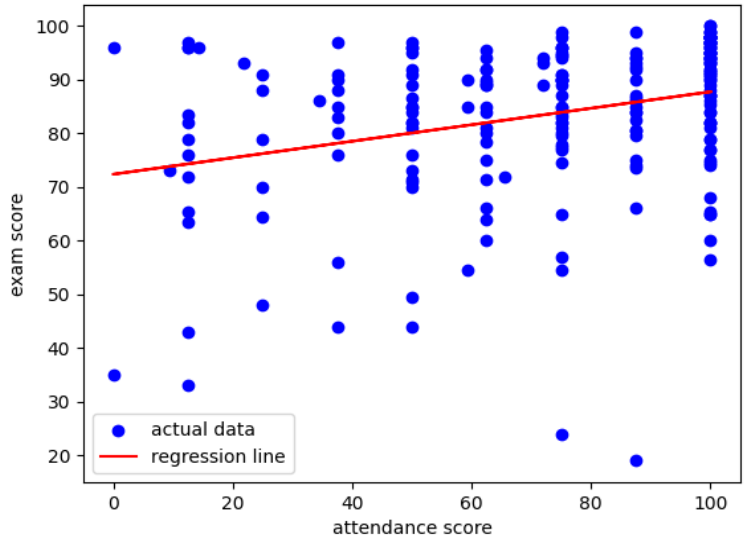
  - Resulting automaton can be reduced

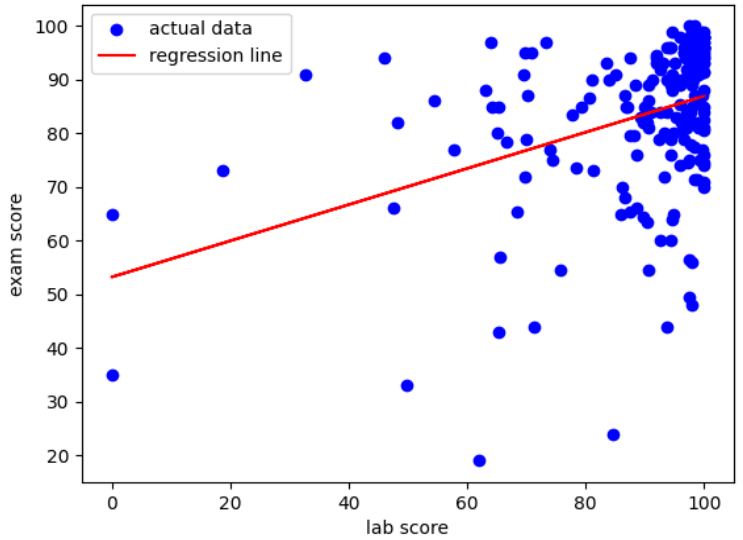
- Example

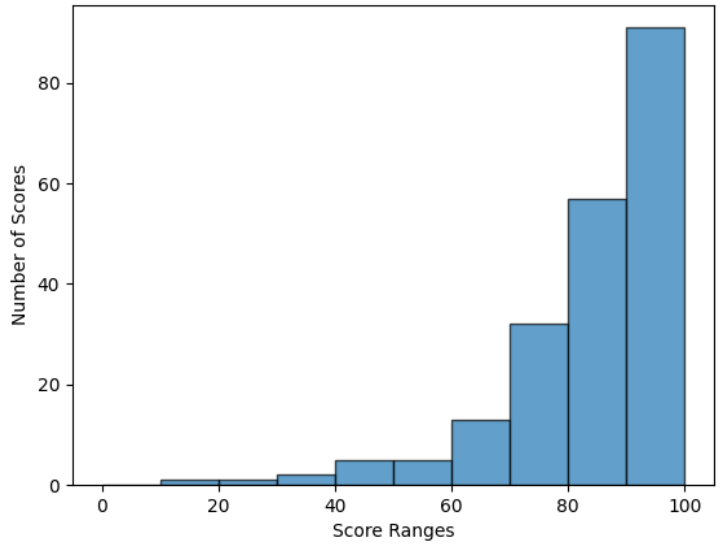


Exam Statistics

- p-values: lab = 2.39e-08, attendance = 1.28e-05. Smaller values mean more statistical significance, 0.05 is a good benchmark and we're way below that.
- Every +3 points to overall lab score = +1 exam point, every +7 attendance points = +1 exam point.
- Mean: 83.4, Median: 87.0, 25th percentile: 78.25, 75th percentile: 94.0.
- 43.9% got a 90 or higher, 71.4% got an 80 or higher, 86.9% got a 70 or higher.
- TLDR: Go to class and focus on labs and you do better on exams.







iClicker Question

What is the difference between Mealy and Moore?

A) Mealy machines are more robust for short impulses of the inputs

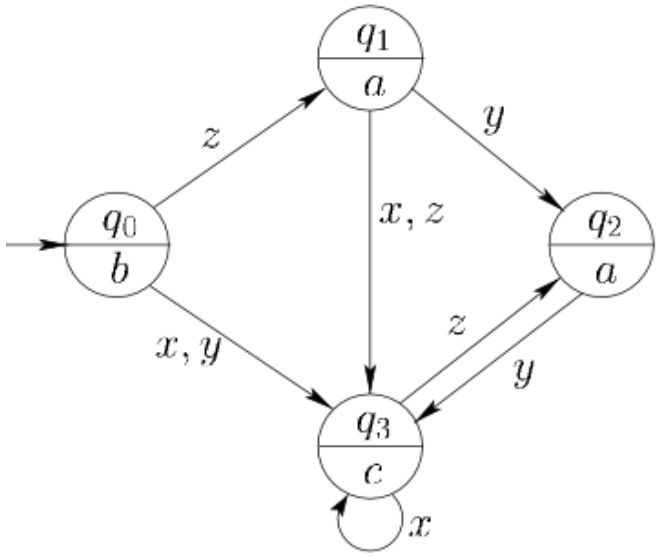
B) Moore machines are more robust for short impulses of the inputs

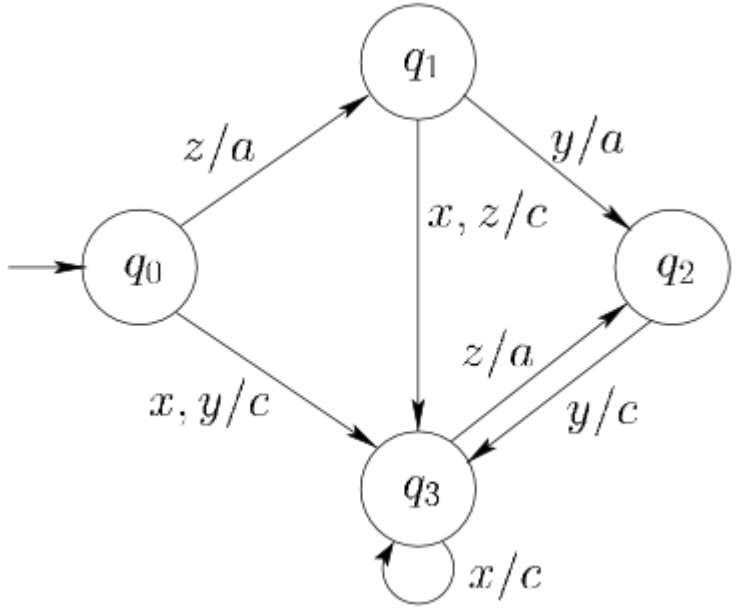
C) Moore machines react faster to changes at the input

D) Mealy machines react faster to changes at the input

Conversion - Moore to Mealy

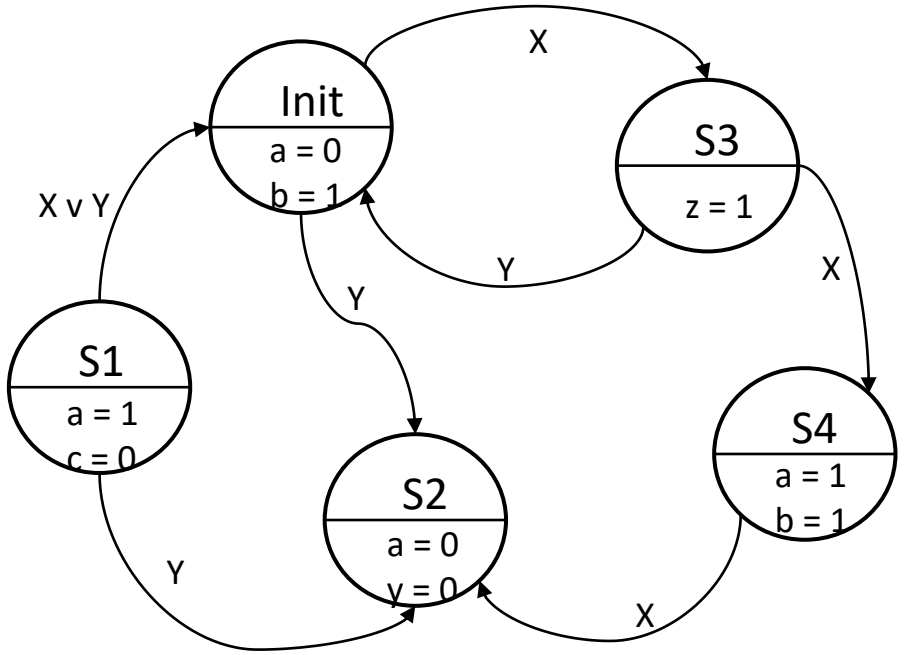
- Convert the following Moore automaton into a Mealy automaton





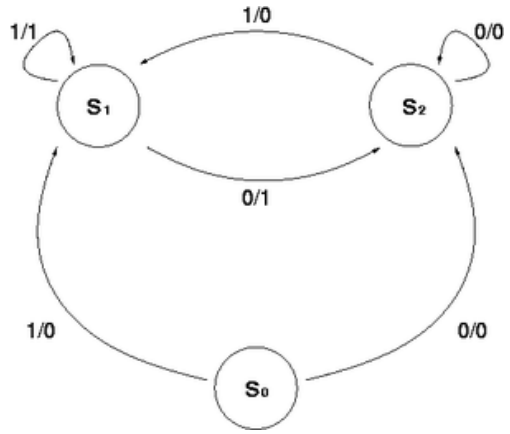
Conversion - Moore to Mealy

- Convert the following Moore automaton into a Mealy automaton



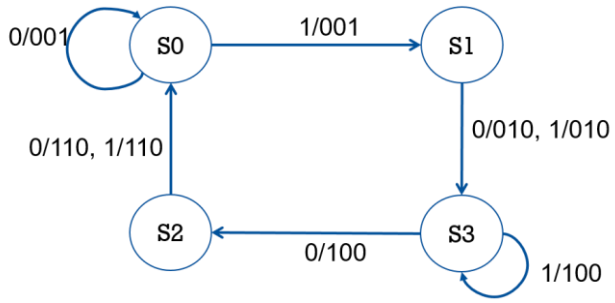
Conversion - Moore to Mealy

- Convert the following Mealy automaton into a Moore automaton



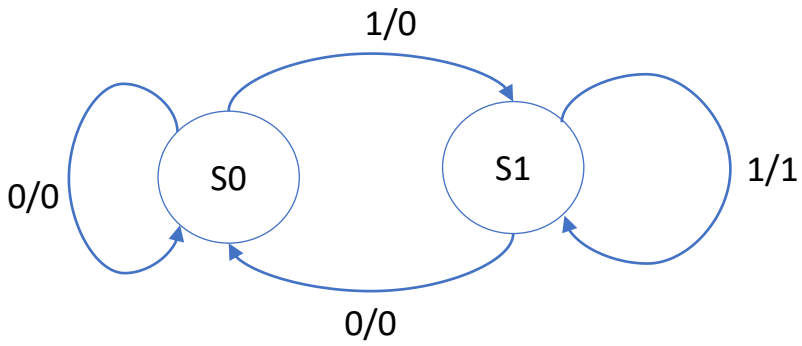
Conversion - Moore to Mealy

- Convert the following Mealy automaton into a Moore automaton



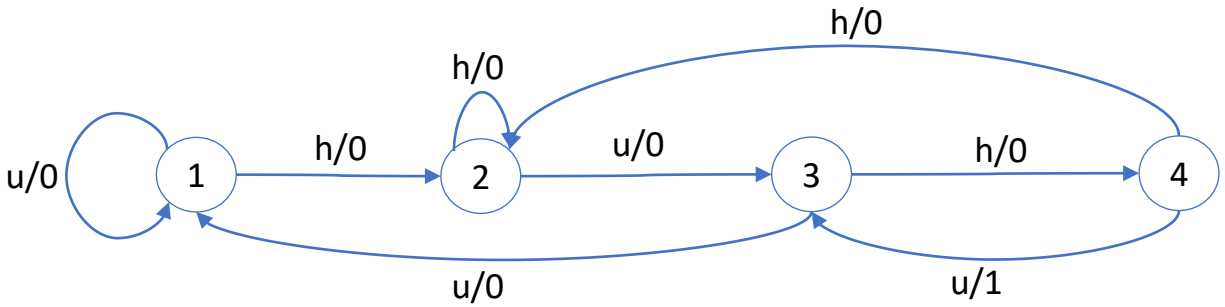
Conversion - Moore to Mealy

- Convert the following Mealy automaton into a Moore automaton



Conversion - Moore to Mealy

- Convert the following Mealy automaton into a Moore automaton




UF
Herbert Wertheim
College of Engineering
UNIVERSITY of FLORIDA
LEADING THE CHARGE, CHARGING AHEAD