Digital Design with the Verilog HDL

Chapter 6 Finite State Machine

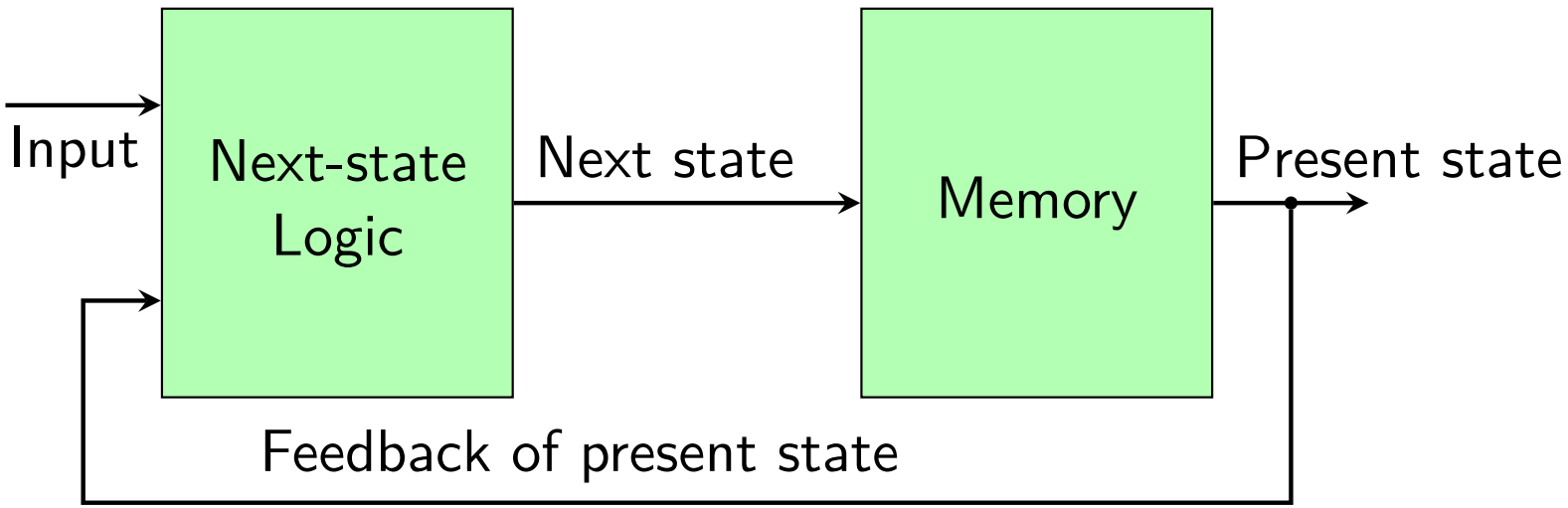
Binh Tran-Thanh

Department of Computer Engineering
Faculty of Computer Science and Engineering
Ho Chi Minh City University of Technology

May 26, 2023

Sequential Machine - Definition

- State of a sequential machine contains current information (t)
- Next state ($t+1$) depends on the current state (t) and inputs
- The number of states in a sequential machine finite $\Rightarrow$ **Finite State Machine - FSM**



Block Diagram of a sequential machine

Synchronous Sequential Machine

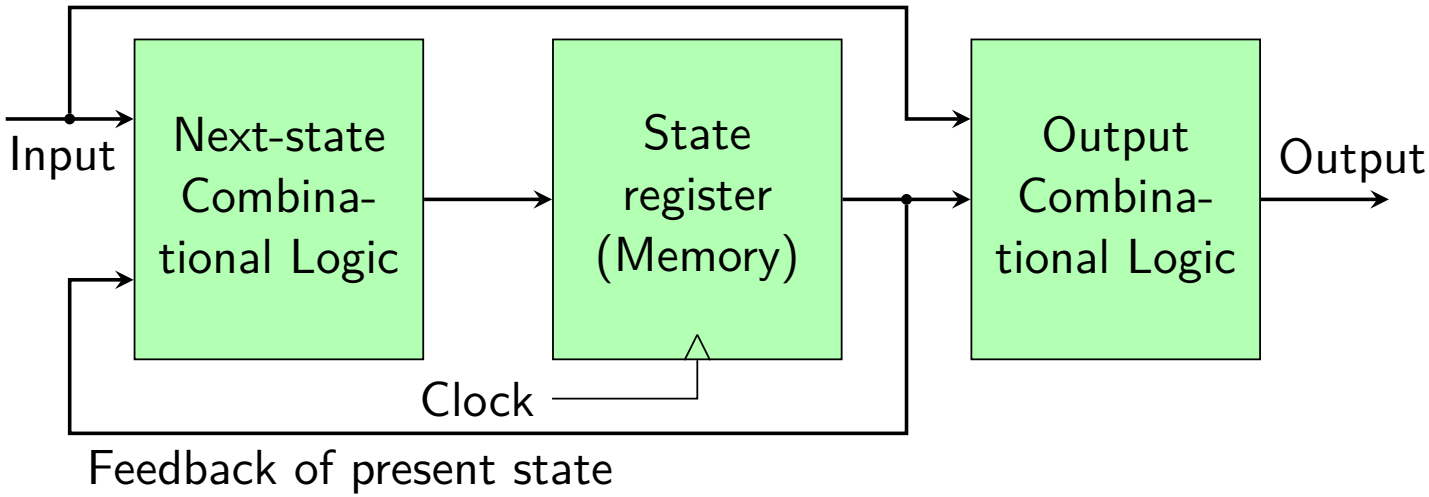
- Synchronous State Machine uses clock to synchronize input states
- Clock is symmetric or asymmetric
- Clock cycle **must be larger than time required for state transaction calculation**
- Synchronous FSMs:
  - Number of states
  - Using clock to control state transaction

FSM Models & Types

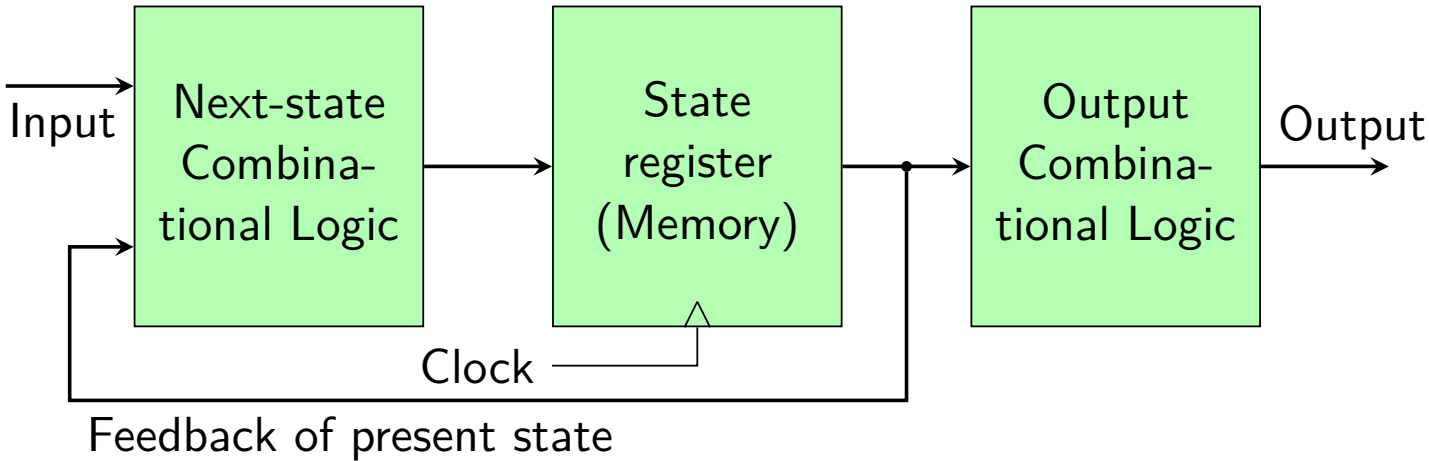
- Explicit
  - Declares a state register that stores the FSM state
  - May not be called “state” –might be a counter!
- Implicit
  - Describes state implicitly by using multiple event controls
- Moore
  - Outputs depend on state only (synchronous)
- Mealy
  - Outputs depend on inputs and state (asynchronous)
  - Outputs can also be registered (synchronous)

Mealy machine vs. Moore machine

Block Diagram of a Mealy sequential machine



Block Diagram of a Moore sequential machine

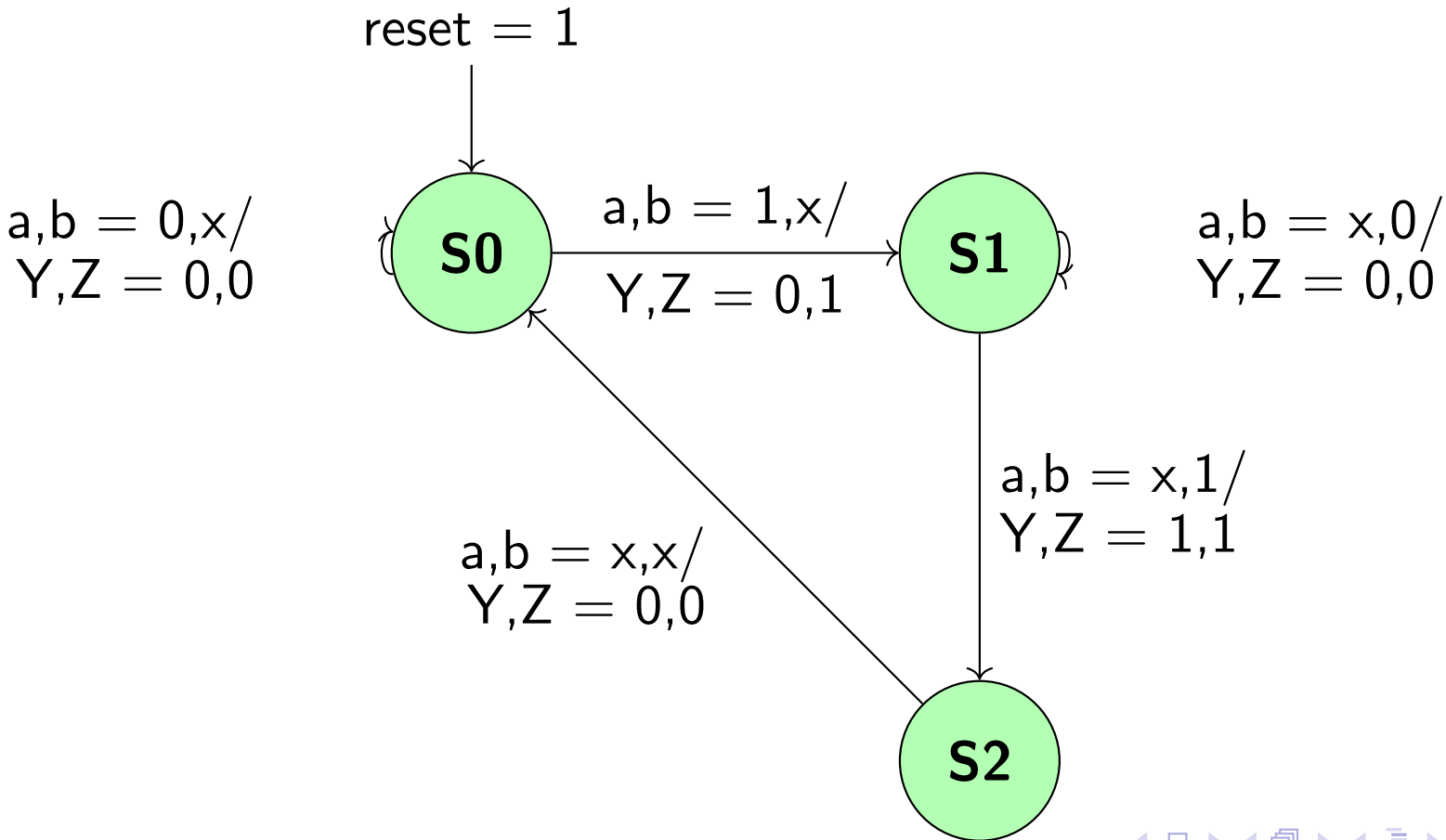


State Transaction Graph

- Finite state machine can be described:
  - State transaction graph, State transaction table
  - Time chart
  - Abstract state machine
- Finite state machine is a directed graph
  - Vertices show states (+outputs if Moore-style machine)
  - Edges show transactions from state to state
- Edges' name
  - Mealy machine: input/output
  - Moore machine: input

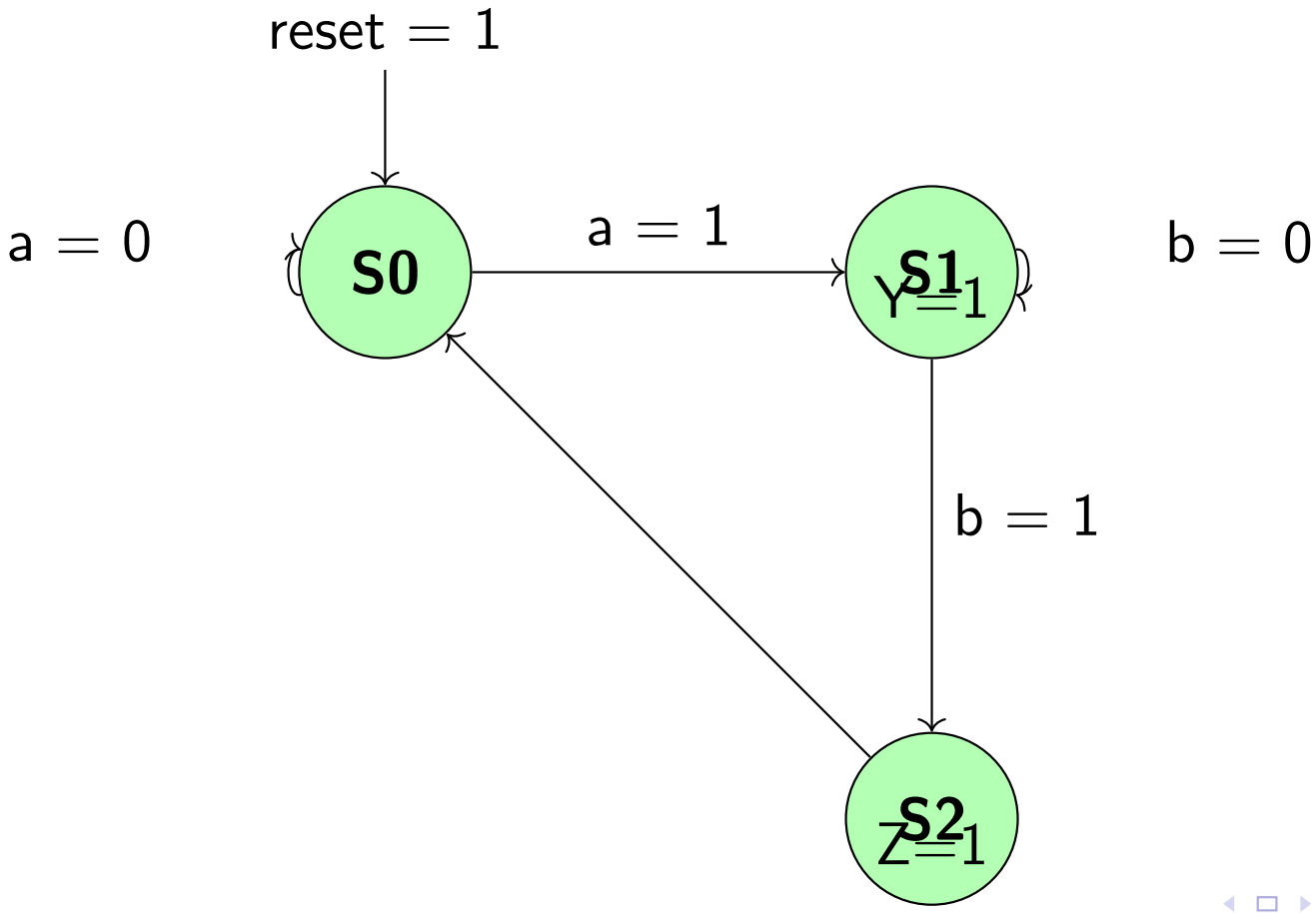
State Diagram: Mealy

- Outputs Y and Z are 0, unless specified otherwise.
- We don't care about the value of b in S0, or the value of a in S1, or either a or b in S2.



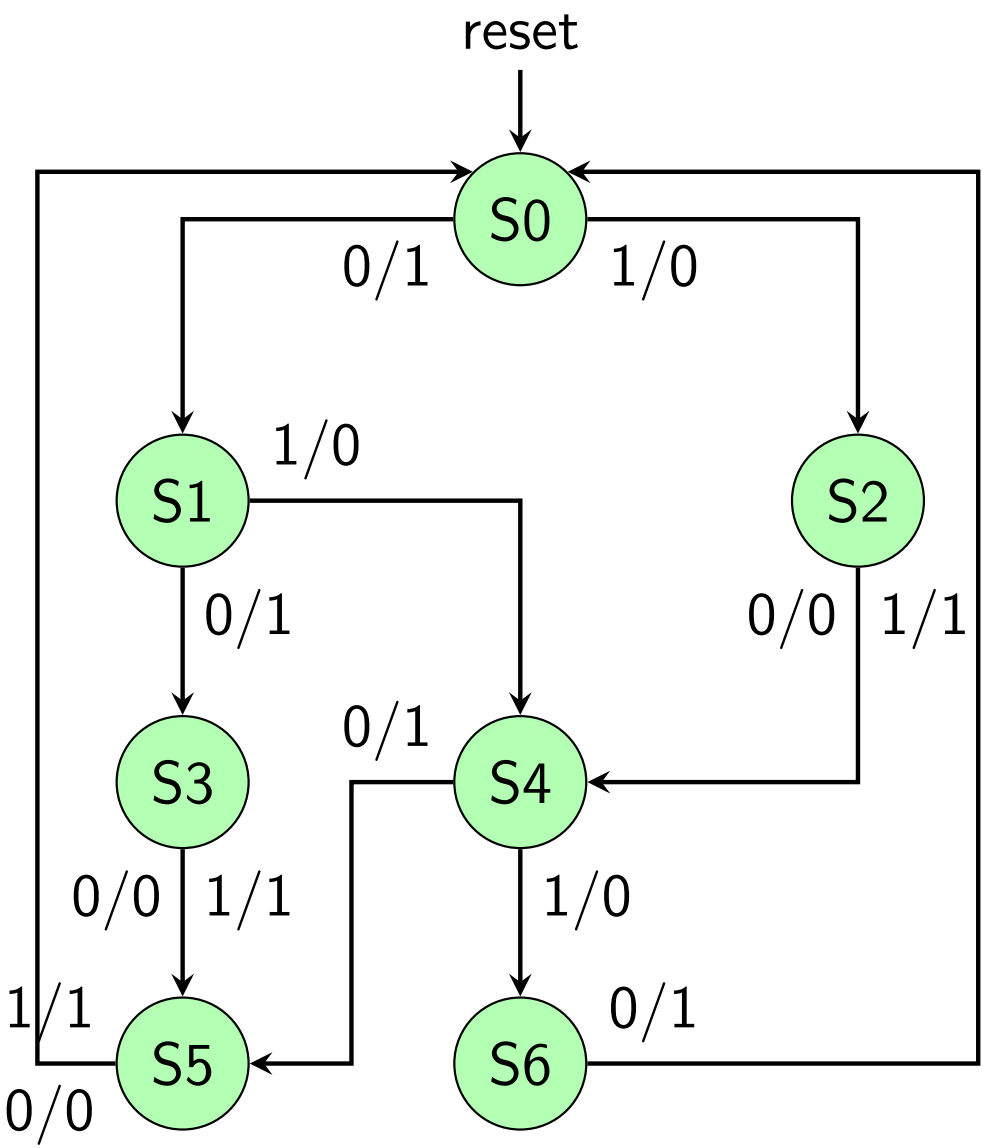
State Diagram: Moore

- Outputs Y and Z are 0, unless specified otherwise.
- If an input isn't listed for a transition, we don't careabout its value for that transition



Example - Mealy

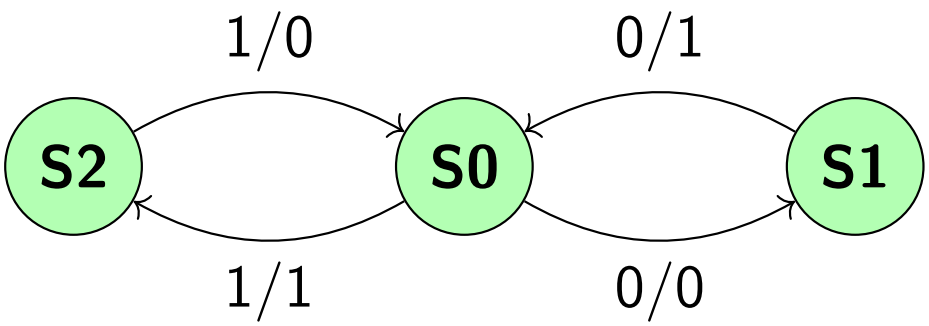
State transition graph



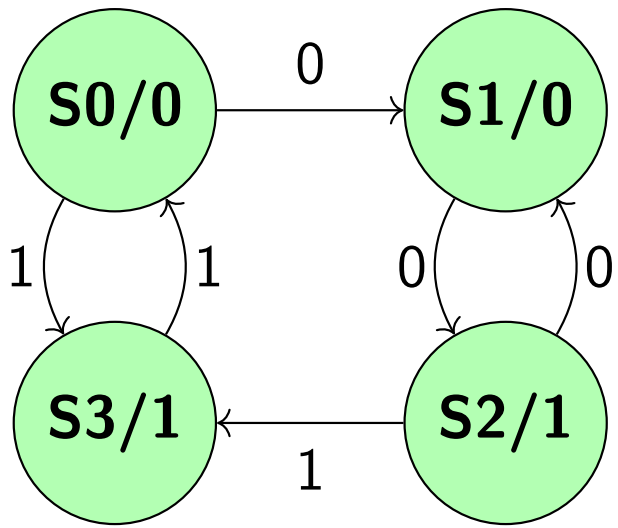
State transition table

State	Next state/ output	
	input	
	0	**1**
S0	S1/1	S2/0
S1	S3/1	S4/0
S2	S4/0	S4/1
S3	S5/0	S5/1
S4	S5/1	S6/0
S5	S0/0	S0/1
S6	S0/1	-/-

Example



State	Next state/ output	
	input	
	0	**1**
S0	S1/0	S2/1
S1	S0/1	-
S2	-	S0/0



State	Next state/ output	
	input	
	0	**1**
S0	S1/0	S3/1
S1	S2/1	-
S2	-	S0/1
S3	S1/0	S3/0

Constraints

- Each vertex describes only one state
- Each edge describes exactly one transaction from current state to the next state
- Each vertex has all out-going edges
- At one edge, there is only one out-going edge at one time

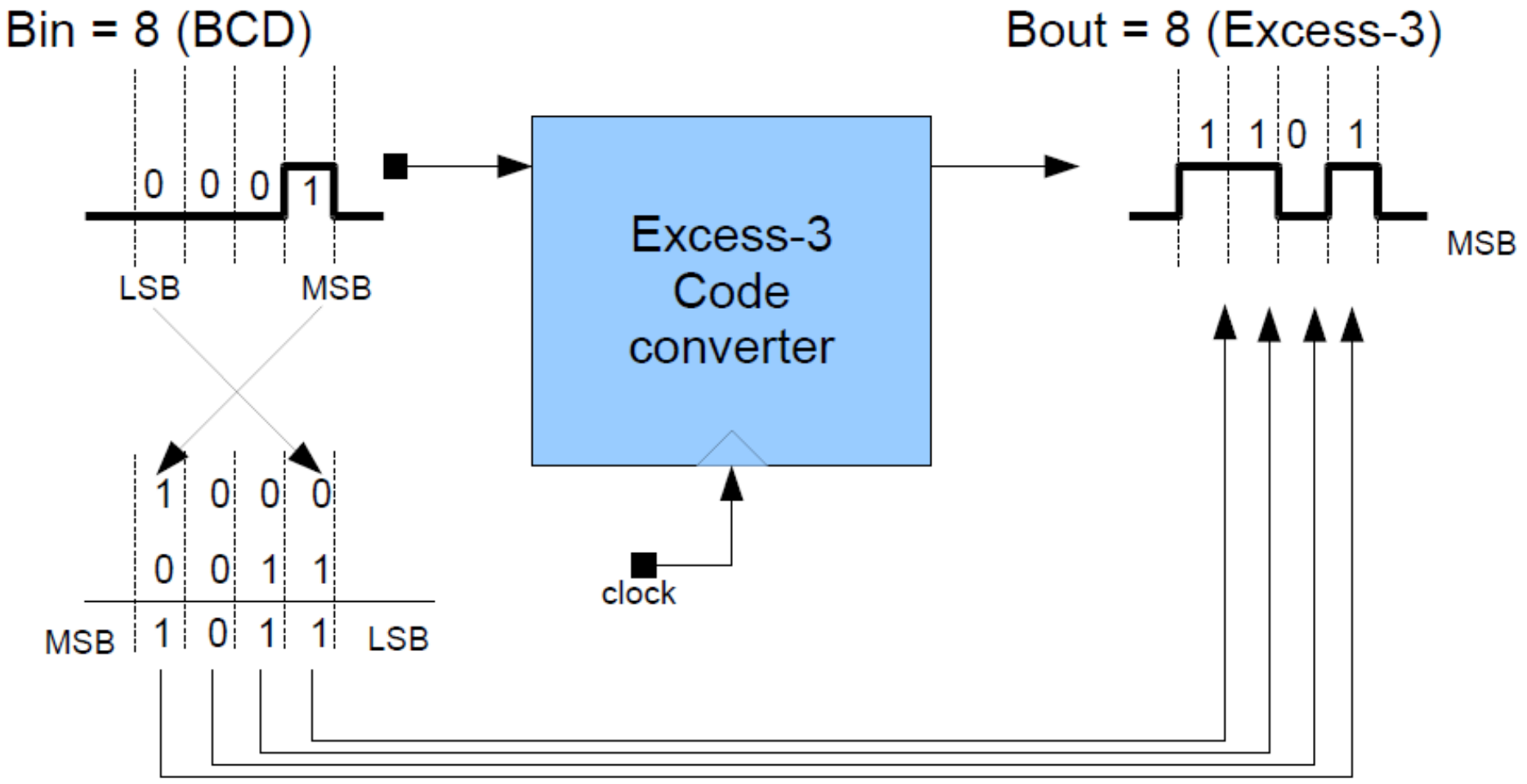
BCD to Excess-3 code Converter

Decimal digit	BCD	Excess-3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Excess-3 is self-complementing

- $6_{10} = 0110_2$
- $6_{excess-3} = 0110_2 + 0011_2 = 1001_2$

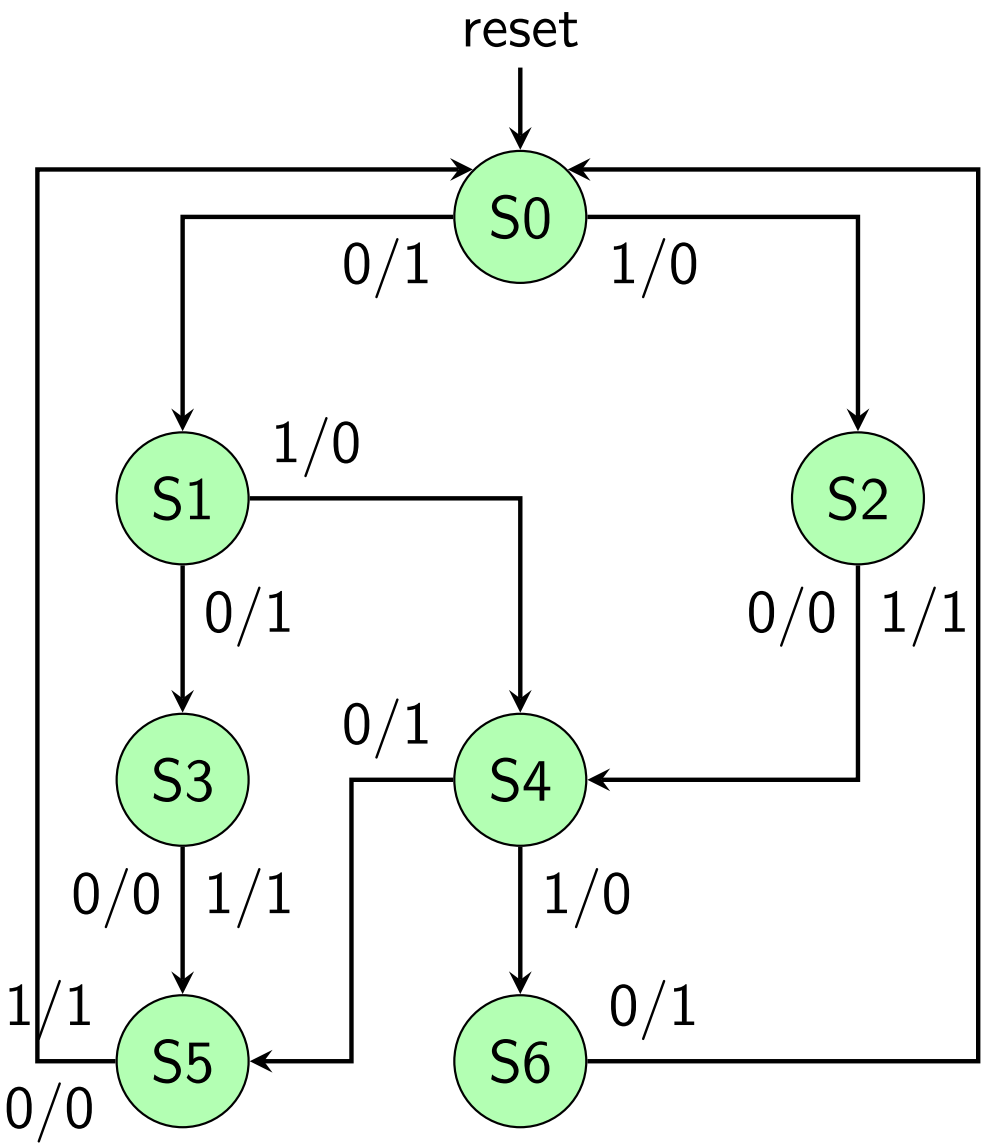
Input/output Relation



Input-output bit stream in a BCD to Excess-3 serial code converter

State Transaction Graph – State Transaction Table

State transition graph (Mealy)



State transition table (Mealy)

State	Next state/ output	
	input	
	0	**1**
S0	S1/1	S2/0
S1	S3/1	S4/0
S2	S4/0	S4/1
S3	S5/0	S5/1
S4	S5/1	S6/0
S5	S0/0	S0/1
S6	S0/1	-/-

State Encoding

- States are stored by FFs
- 7 states, using 3 FFs

State assignment	
$q_2q_1q_0$	State
000	S0
001	S1
010	S6
011	S4
100	-
101	S2
110	S5
111	S3

State $q_2q_1q_0$	**Next state** $q_2^+q_1^+q_0^+$		**output**	
	Input		**Input**	
	0	1	0	1
000 (S0)	001	101	1	0
001 (S1)	111	011	1	0
101 (S2)	011	011	0	1
111 (S3)	110	110	0	1
011 (S4)	110	010	1	0
110 (S5)	000	000	0	1
010 (S6)	000	-	1	-
100 (-)	-	-	-	-

Simplify State Transaction Function

q_2q_1 \ q_0B_{in}	00	01	11	10
00	1	1	1	1
01	0	x	0	0
11	0	0	0	0
10	x	x	1	1

$$q_0^+ = q_1'$$

q_2q_1 \ q_0B_{in}	00	01	11	10
00	0	0	1	1
01	0	x	1	1
11	0	0	1	1
10	x	x	1	1

$$q_1^+ = q_0$$

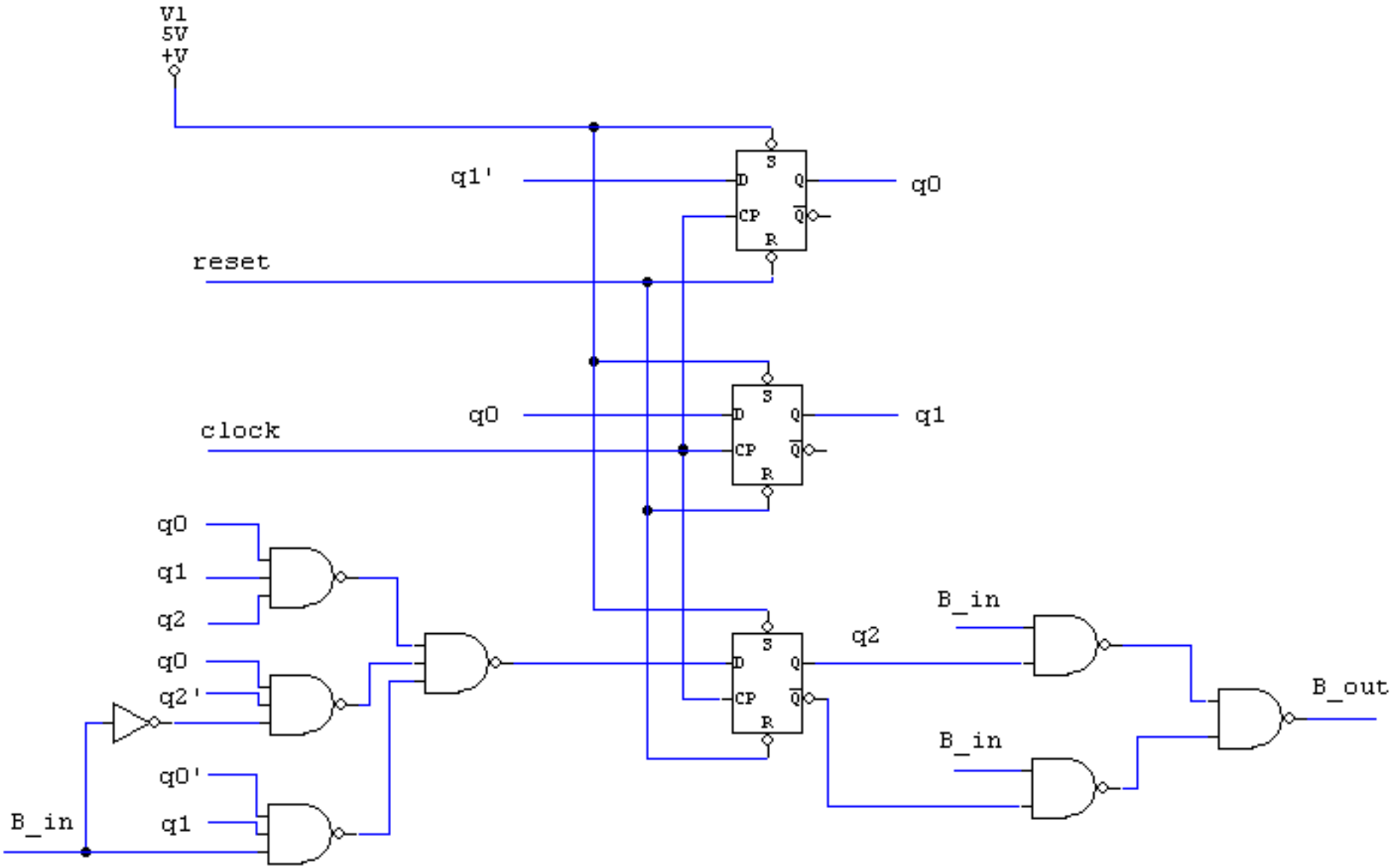
q_2q_1 \ q_0B_{in}	00	01	11	10
00	0	1	0	1
01	0	x	0	1
11	0	0	1	1
10	x	x	0	0

$$q_2' = q_1'q_0'B_{in} + q_2'q_0B_{in}' + q_2q_1q_0$$

q_2q_1 \ q_0B_{in}	00	01	11	10
00	1	0	0	1
01	1	x	0	1
11	0	1	1	0
10	x	x	1	0

$$B_{out} = q_2'B_{in}' + q_2B_{in}$$

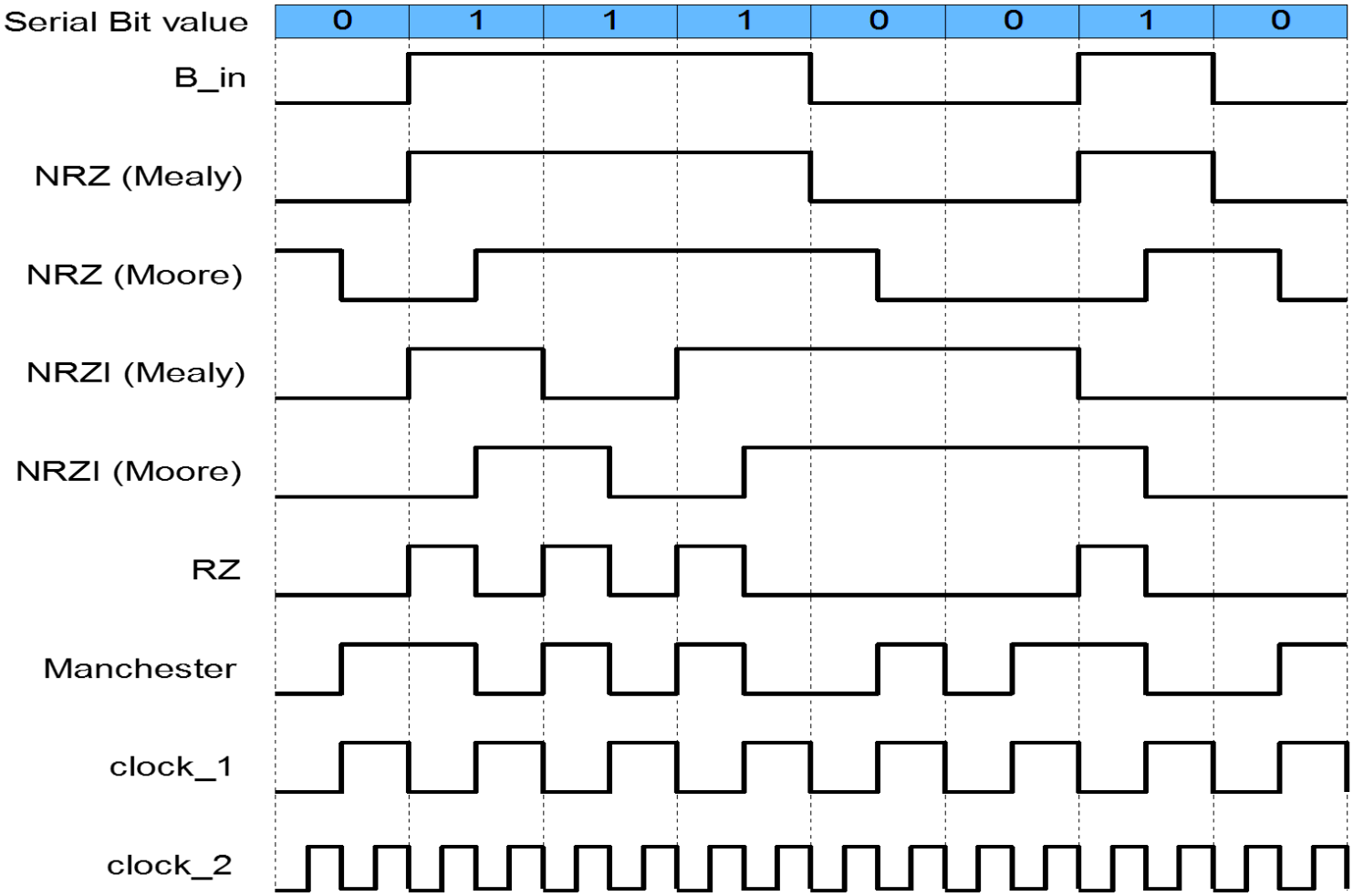
Implementing BCD to Excess-3 Converter



FSM Example: Serial-Line Code Converter

- 3 signals:
  - Clock
  - Handshaking signal
  - Data
- Well-known encoding algorithms:
  - NRZ
  - NRZI: if input is 1, the previous output value is inversed while 0 input keeps output unchanged
  - RZ: if input is 1, output is 1 during the first half cycle and 0 during the second half cycle while 0 input produces 0 output
  - Manchester: if input is 0, output is 0 during the first half cycle and 1 during the second half cycle while 1 input produces 1 output during the first half and 0 output during the second half

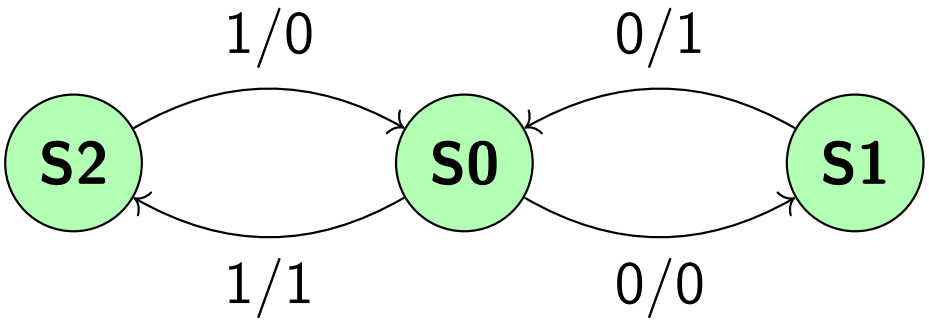
Serial Encoding Examples



Clock_2's frequency is double clock_1's frequency to implement the NRZI, RZ, and Manchester encoding algorithms

Mealy FSM for Serial Encoding

- The Manchester algorithm
  - Waiting state (S0)
  - Just receiving 1 state (S2)
  - Just receiving 0 state (S1)



	Next state/ output	
State	**input**	
	0	**1**
S0	S1/0	S2/1
S1	S0/1	-
S2	-	S0/0

state	$q_1 q_0$
S0	00
S1	01
S2	10

State $q_1 q_0$	**Next state** $q_1^+ q_0^+$		**output**	
	Input		**Input**	
	0	1	0	1
00 (S0)	01	10	0	1
01 (S1)	00	00	1	-
10 (S2)	00	00	-	0

Implementing the Mealy FSM

q_1q_0 \ B_{in}	0	1
00	0	1
01	0	0
11	-	-
10	0	0

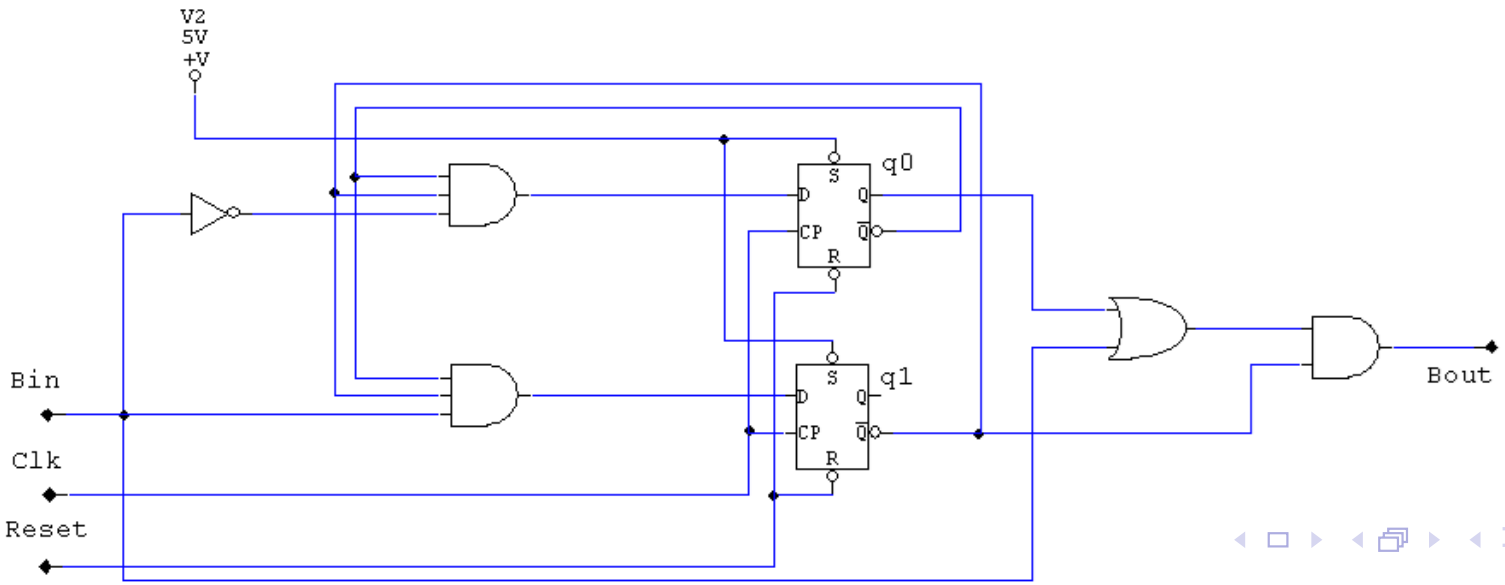
$$q_1^+ = q_1' q_0' B_{in}$$

q_1q_0 \ B_{in}	0	1
00	1	0
01	0	0
11	-	-
10	0	0

$$q_0^+ = q_1' q_0' B_{in}'$$

q_1q_0 \ B_{in}	0	1
00	0	1
01	1	1
11	-	-
10	0	0

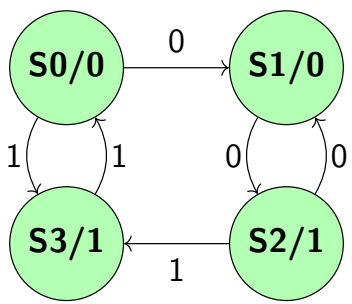
$$B_{out} = q_1' q_0 + q_1' B_{in}$$



Moore FSM for Serial Encoding

The Manchester Algorithm

- S0: starting/second half of the cycle receiving 1, the output is 0
- S1: first half of the cycle receiving 0, the output is 0
- S2: second half of the cycle receiving 0, the output is 1
- S3: first half of the cycle receiving 1, the output is 1



	Next state/ output	
State	**input**	
	0	**1**
S0	S1/0	S3/1
S1	S2/1	-
S2	-	S0/1
S3	S1/0	S3/0

State $q_1 q_0$	**Next state** $q_1^+ q_0^+$		**output**
	Input		
	0	1	
00 (S0)	01	11	0
01 (S1)	10	-	0
11 (S3)	-	00	1
10 (S2)	01	11	1

Implementing the Moore FSM

q_1q_0 \ B_{in}	0	1
00	0	1
01	1	x
11	-	0
10	0	1

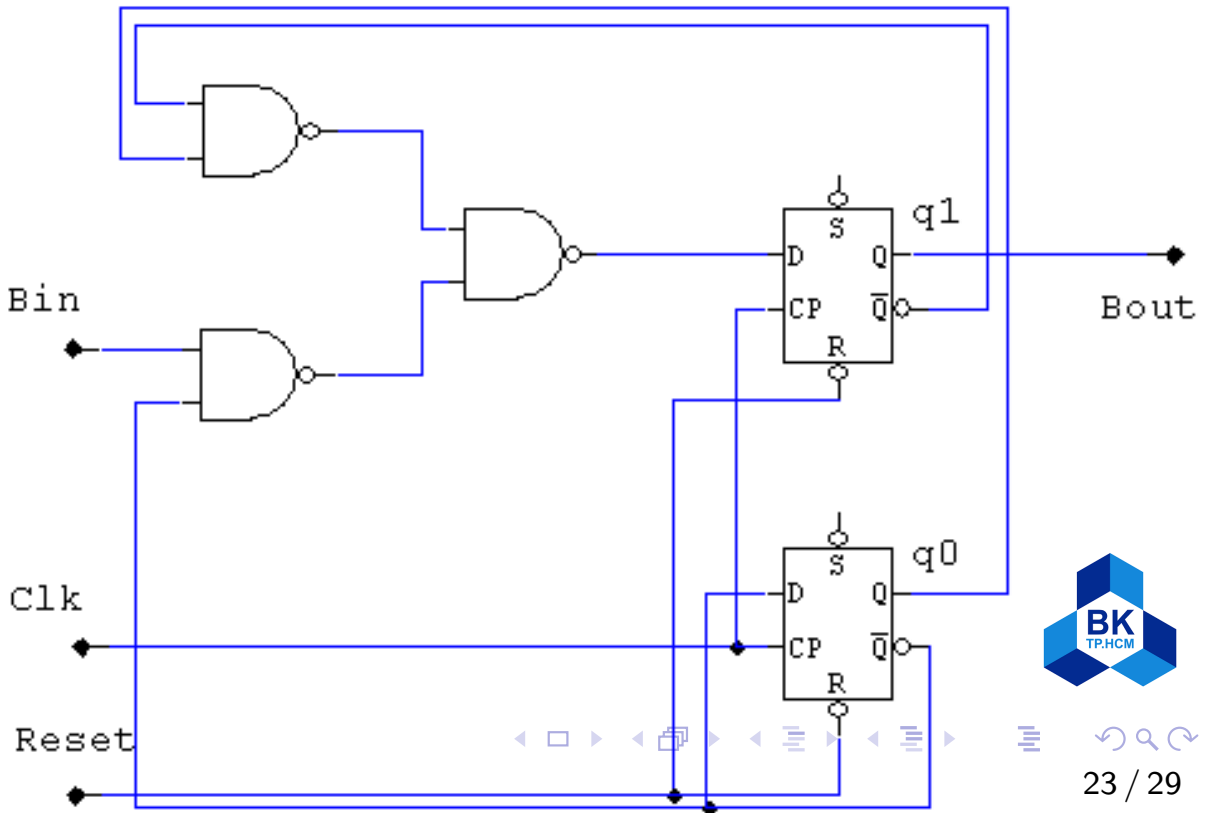
$$q_1^+ = q_0' B_{in} + q_1' q_0$$

q_1q_0 \ B_{in}	0	1
00	1	1
01	0	-
11	-	0
10	1	1

$$q_0^+ = q_0'$$

q_0 \ B_{in}	0	1
0	0	0
1	1	1

$$B_{out} = q_0$$

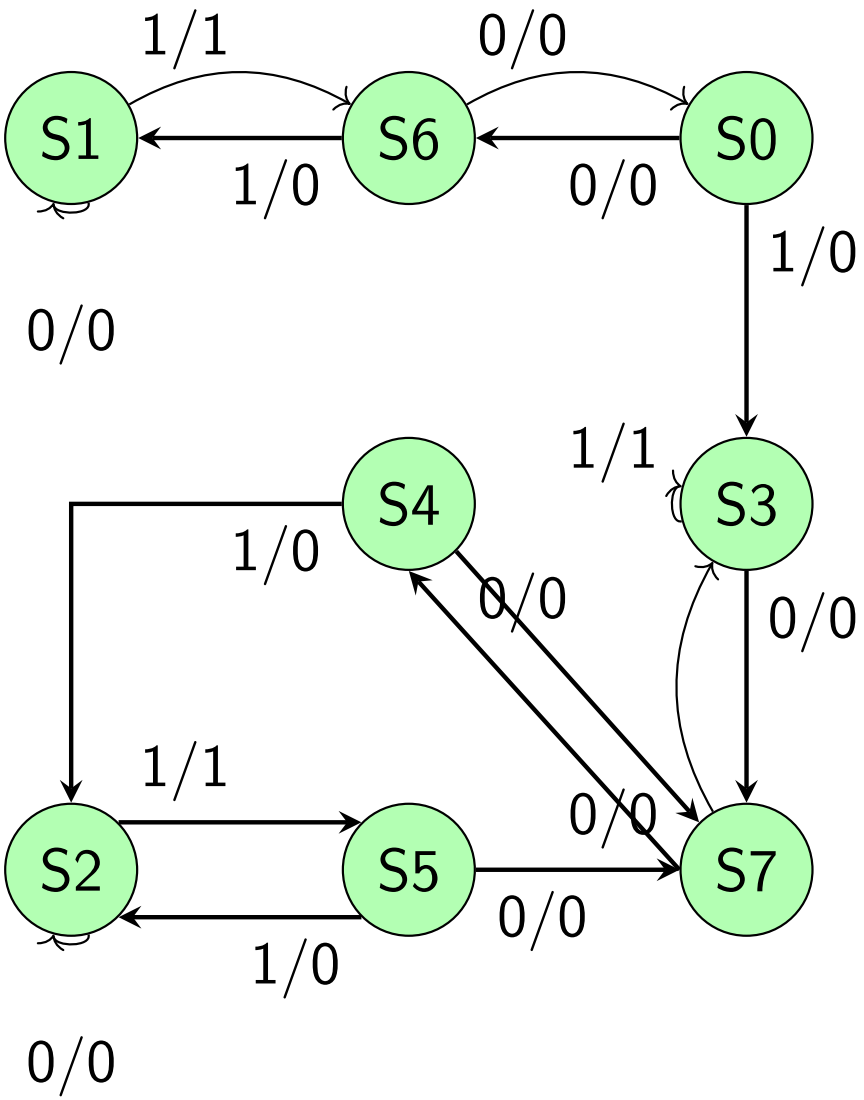


Simplify Equivalent States

- Two states are equivalent:
  - Output and the next states are the same in all inputs (c1)
  - Can be combined together without any changed behavior (c2)
- Reducing two equivalent states reduces hardware cost
- Each FSM has **one and only one simplest equivalent FSM**

State	**Next state Input**		**output Input**	
	0	1	0	1
S0	S6	S3	0	0
S1	S1	S6	0	1
S2	S2	S5	0	1
S3	S7	S3	0	1
S4	S7	S2	0	0
S5	S7	S2	0	0
S6	S0	S1	0	0
S7	S4	S3	0	0

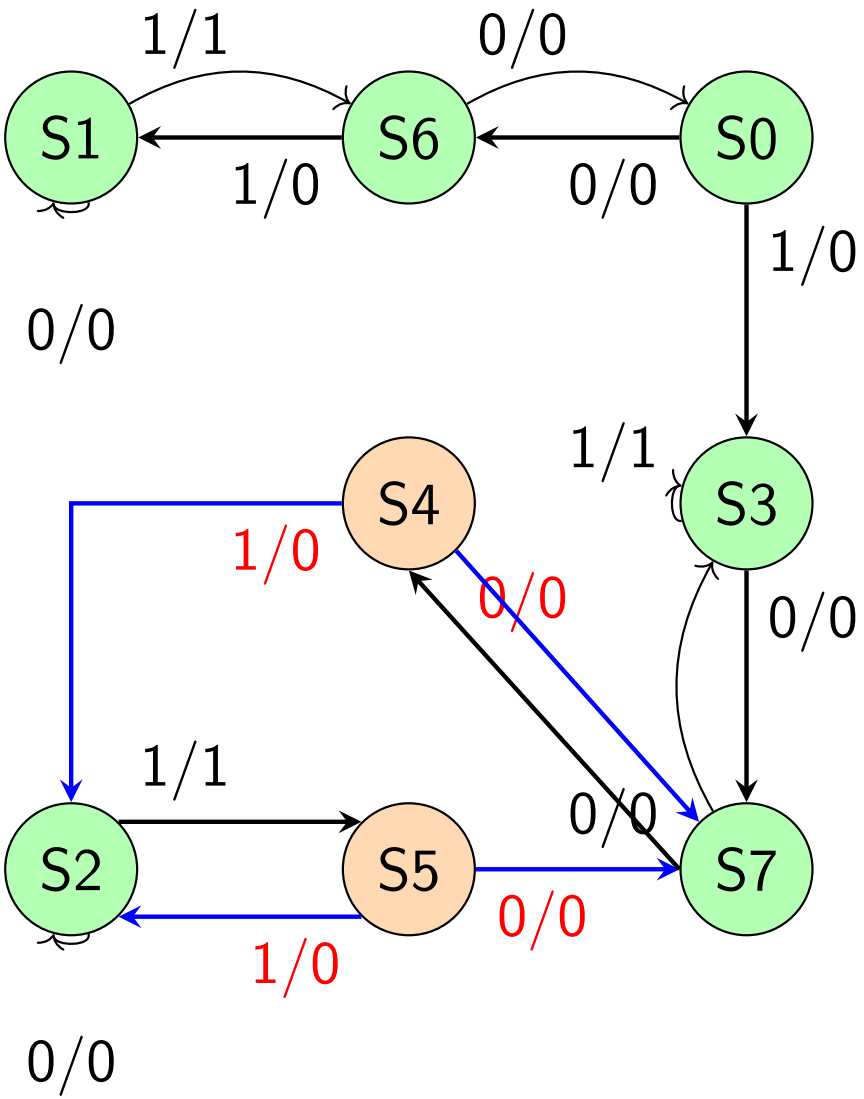
Simplify Equivalent States example

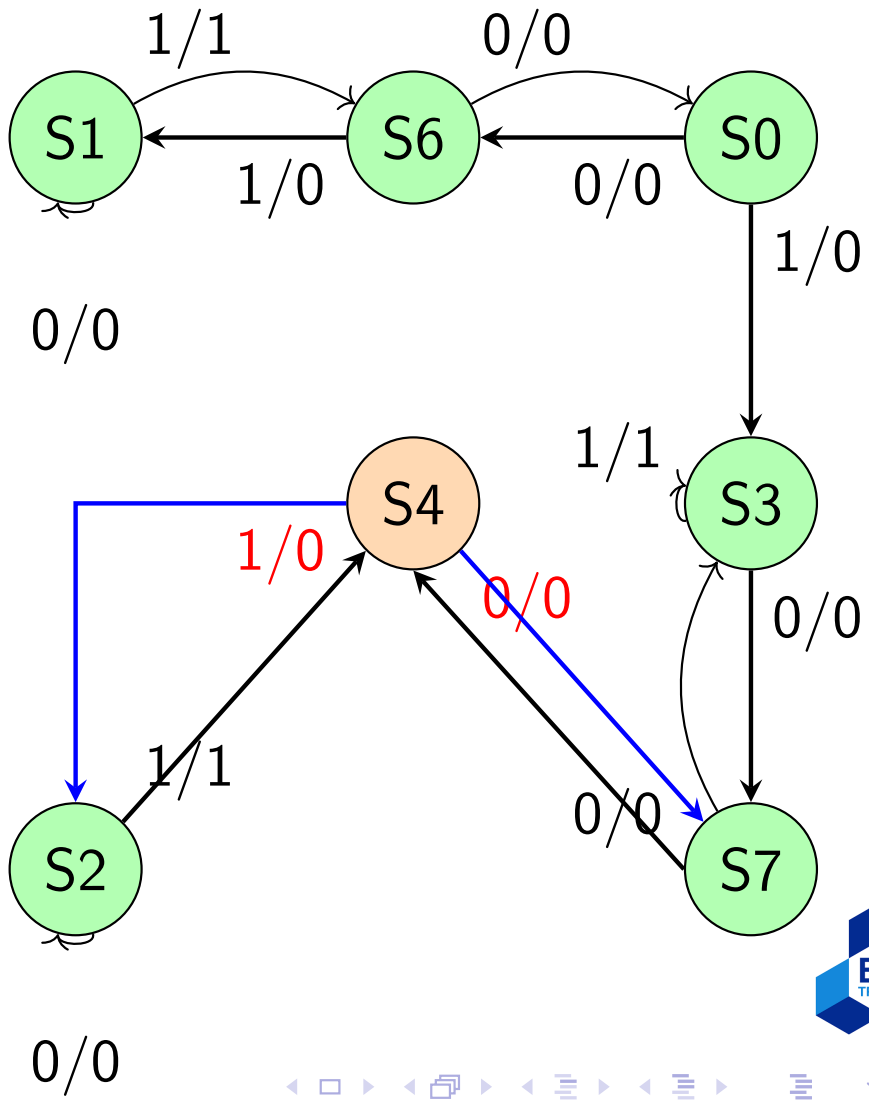


State	Next state Input		output Input	
	0	1	0	1
S0	S6	S3	0	0
S1	S1	S6	0	1
S2	S2	S5	0	1
S3	S7	S3	0	1
S4	S7	S2	0	0
S5	S7	S2	0	0
S6	S0	S1	0	0
S7	S4	S3	0	0

Simplify Equivalent States Algorithm

- Step 1: Find basic equivalent states (c1)





Simplify Equivalent States Algorithm

- Step 2: Create a possible equivalent states table (c2)
  - Let impossible equivalent cells be empty
  - Fill conditions upon which two corresponding states can be equivalent

S_1						
S_2		S_6 S_4				
S_3		S_1 S_7 S_6 S_3	S_2 S_7 S_4 S_3			
S_4	S_6 S_7 S_3 S_2					
S_6	S_3 S_1				S_7 S_0 S_2 S_1	
S_7	S_6 S_4				S_2 S_3	S_0 S_4 S_1 S_3
	S_0	S_1	S_2	S_3	S_4	S_6

S_1 và S_0 không thể tương đương

S_2 và S_1 tương đương khi
S_6 và S_4 tương đương

	Next state		Output	
	Input		Input	
State	0	1	0	1
S_0	S_6	S_3	0	0
S_1	S_1	S_6	0	1
S_2	S_2	S_4	0	1
S_3	S_7	S_3	0	1
S_4	S_7	S_2	0	0
S_6	S_0	S_1	0	0
S_7	S_4	S_3	0	0

Simplify Equivalent States Algorithm

- Step 3: Consider equivalent conditions of any two states, delete corresponding cell if the cell contains any inequivalent couple

S_1					
S_2		S_6 S_4			
S_3		S_1 S_7 / S_6 S_3	S_2 S_7 / S_4 S_3		
S_4	S_6 S_7 / S_3 S_2				
S_6	S_3 S_1				S_7 S_0 / S_2 S_1
S_7	S_6 S_4				S_2 S_3	S_0 S_4 / S_1 S_3
	S_0	S_1	S_2	S_3	S_4	S_6

$S\_0 \equiv S\_7$
$S\_1 \equiv S\_2$
$S\_4 \equiv S\_6$