Introduction to Digital Systems

Part III (Sequential Components)

2022/2023

Design (Synthesis) of Clocked Synchronous Finite State Machines

Arnaldo Oliveira, Augusto Silva, Iouliia Skliarova

Lecture Contents

- FSM design (synthesis) steps
- Modeling formalisms
  - State Diagrams
  - State Tables
- Ilustrative examples
  - Up/down counter with Mealy output flags
  - Moore sequence detector FSM
- Initialization (reset)
- Unused state encodings
- One-hot state encoding

Some figures and content extracted from: John F. Wakerly, "Digital Design – Principles and Practices", 4 ed., Pearson – Prentice Hall, 2006 (chapter 7). Reading chapter 7 (4th ed.) or chapter 10 (5th ed.) is highly recommended.

Introduction

- What is digital system synthesis (in general)?
- What is FSM synthesis?
  - Design steps
- Initial specification in natural language, timing diagrams, etc.
- Modelling formalisms that will be used to translate the initial specification
  - State Diagrams
  - State Tables
- Let's discuss the synthesis flow based on examples…

Design Steps

1.	Construct a state/output table corresponding to the word description or specification, using mnemonic names for the states. (It's also possible to start with a state diagram; this method is discussed in Section 7.5.)	*state/output table*
2.	(Optional) Minimize the number of states in the state/output table.	*state minimization*
3.	Choose a set of state variables and assign state-variable combinations to the named states.	*state assignment*

- Step 1 is when designer "really designs"
- Experienced designers typically skip step 2
- Remaining steps done following a well-defined synthesis procedure

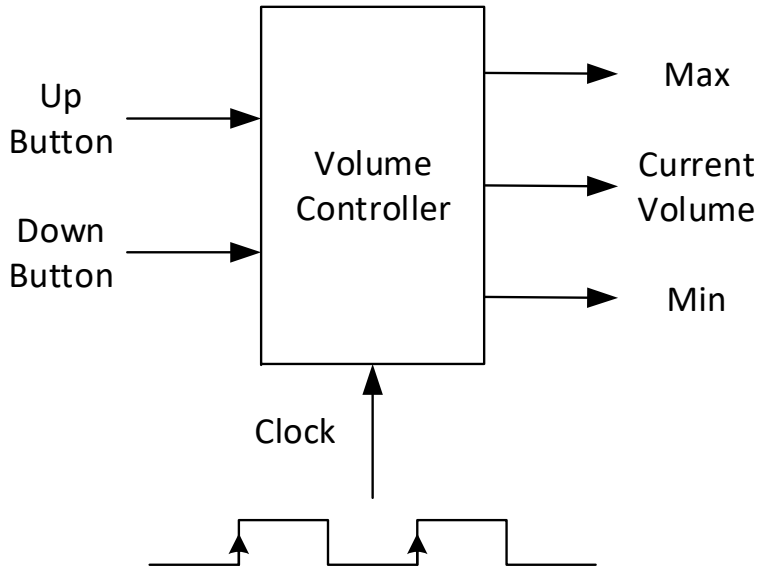
Design Steps

4. Substitute the state-variable combinations into the state/output table to create a transition/output table that shows the desired next state-variable combination and output for each state/input combination. *transition/output table*
5. Choose a flip-flop type (usually, edge-triggered D) for the state memory. In most cases you'll already have a choice in mind at the outset of the design, but this step is your last chance to change your mind.
6. Construct an excitation table that shows the excitation values required to obtain the desired next state for each state/input combination. *excitation table*
7. Derive excitation equations from the excitation table. *excitation equations*
8. Derive output equations from the transition/output table. *output equations*
9. Draw a logic diagram that shows the state-variable storage elements and realizes the required excitation and output equations. (Or realize the equations directly in a programmable logic device.) *logic diagram*

- Steps 4, 6-9 are the most tedious, but typically automated by a compiler/synthesis tool
- Designer must know the procedure to understand / have a critical thinking of the results

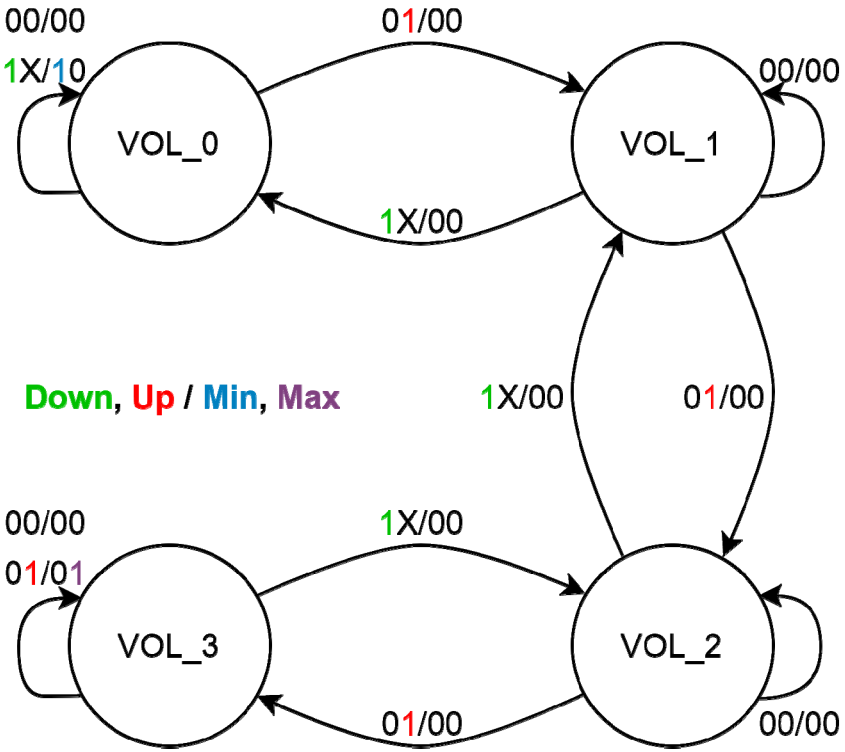
Synthesis Example 1 – Specification

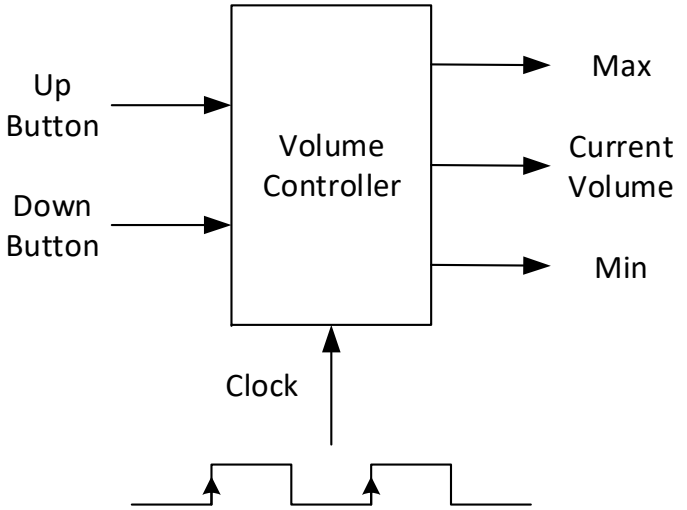
- Whenever “Up” button is pressed, volume must be incremented at the clock tick
- Whenever “Down” button is pressed, volume must be decremented at the clock tick
- To simplify (decrease complexity) assume volume has only 4 levels (possible values)
- “Max” output must be activated if volume is already at maximum level and “Up” is pressed
- “Min” output must be activated if volume is already at minimum and “Down” is pressed
- Current volume level must also be provided at the output of the circuit

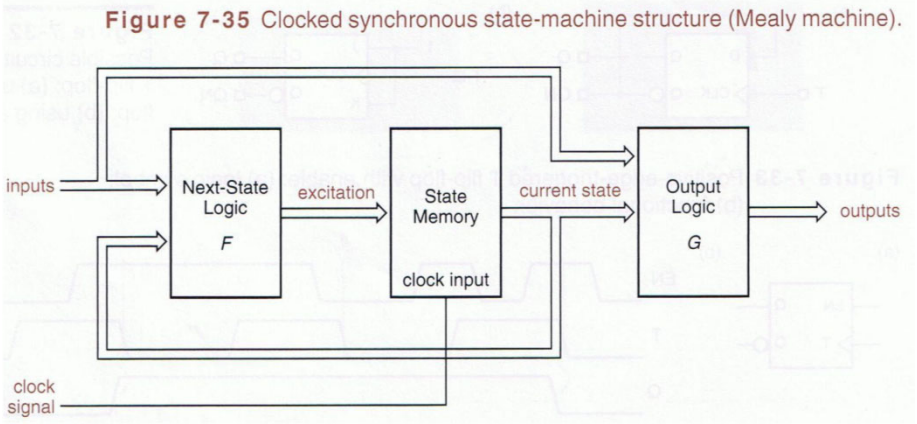


State Diagram

Mealy machine! Why?





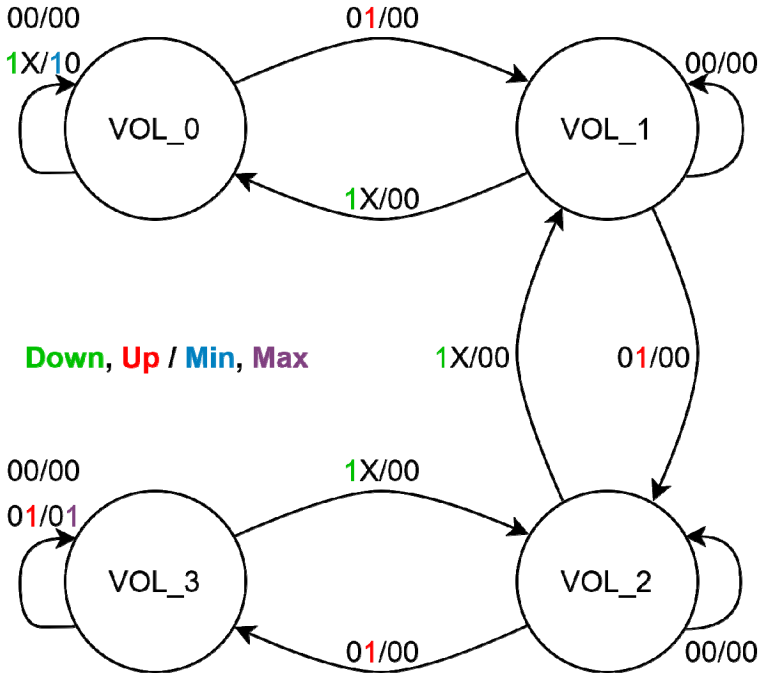


State, Transition/Outputs and Excitation Tables

State / Outputs Table

Down (Dw), Up (Up)

Current State		00		01		11		10	
Meaning	S	S*	Min, Max	S*	Min, Max	S*	Min, Max	S*	Min, Max
Min vol.	Vol_00	Vol_00	00	Vol_01	00	Vol_00	10	Vol_00	10
	Vol_01	Vol_01	00	Vol_02	00	Vol_00	00	Vol_00	00
	Vol_02	Vol_02	00	Vol_03	00	Vol_01	00	Vol_01	00
Max vol.	Vol_03	Vol_03	00	Vol_03	01	Vol_02	00	Vol_02	00



State, Transition/Outputs and Excitation Tables

State / Outputs Table

Down (Dw), Up (Up)

Current State		00		01		11		10	
Meaning	S	S*	Min, Max	S*	Min, Max	S*	Min, Max	S*	Min, Max
Min vol.	Vol_00	Vol_00	00	Vol_01	00	Vol_00	10	Vol_00	10
	Vol_01	Vol_01	00	Vol_02	00	Vol_00	00	Vol_00	00
	Vol_02	Vol_02	00	Vol_03	00	Vol_01	00	Vol_01	00
Max vol.	Vol_03	Vol_03	00	Vol_03	01	Vol_02	00	Vol_02	00

Binary State Encoding		
Meaning	S	Q1 Q0
Min vol.	Vol_00	00
	Vol_01	01
	Vol_02	10
Max vol.	Vol_03	11

Transition / Outputs Table

Down (Dw), Up (Up)

Current State		00		01		11		10	
Meaning	Q1 Q0	Q1* Q0*	Min, Max	Q1* Q0*	Min, Max	Q1* Q0*	Min, Max	Q1* Q0*	Min, Max
Min vol.	00	00	00	01	00	00	10	00	10
	01	01	00	10	00	00	00	00	00
	10	10	00	11	00	01	00	01	00
Max vol.	11	11	00	11	01	10	00	10	00

Excitation / Outputs Table

Down (Dw), Up (Up)

Current State		00		01		11		10	
Meaning	Q1 Q0	D1 D0	Min, Max	D1 D0	Min, Max	D1 D0	Min, Max	D1 D0	Min, Max
Min vol.	00	00	00	01	00	00	10	00	10
	01	01	00	10	00	00	00	00	00
	10	10	00	11	00	01	00	01	00
Max vol.	11	11	00	11	01	10	00	10	00

Using D flip-flops

Q*=D

Minimization of Excitation/Outputs

Excitation / Outputs Table

Down (Dw), Up (Up)

Current State		00		01		11		10	
Meaning	Q1 Q0	D1 D0	Min, Max	D1 D0	Min, Max	D1 D0	Min, Max	D1 D0	Min, Max
Min vol.	00	00	00	01	00	00	10	00	10
	01	01	00	10	00	00	00	00	00
	10	10	00	11	00	01	00	01	00
Max vol.	11	11	00	11	01	10	00	10	00

D0

Q1 \ Dw Up	00	01	11	10
00		1		
01	1			
11	1	1		
10		1	1	1

Q0

D1

Q1 \ Dw Up	00	01	11	10
00				
01		1		
11	1	1	1	1
10	1	1		

Q0

Excitation

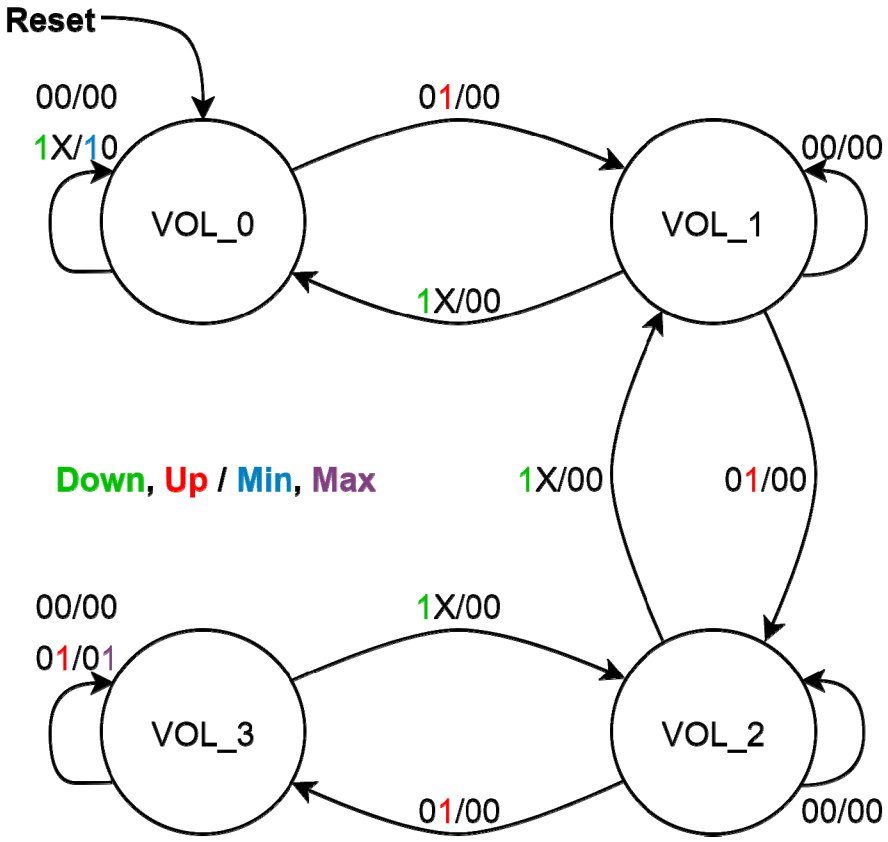
$$D_0 = Q_0.\overline{Dw}.\overline{Up} + \overline{Q_0}.\overline{Dw}.Up + Q_1.Q_0.\overline{Dw} + Q_1.\overline{Q_0}.Dw$$

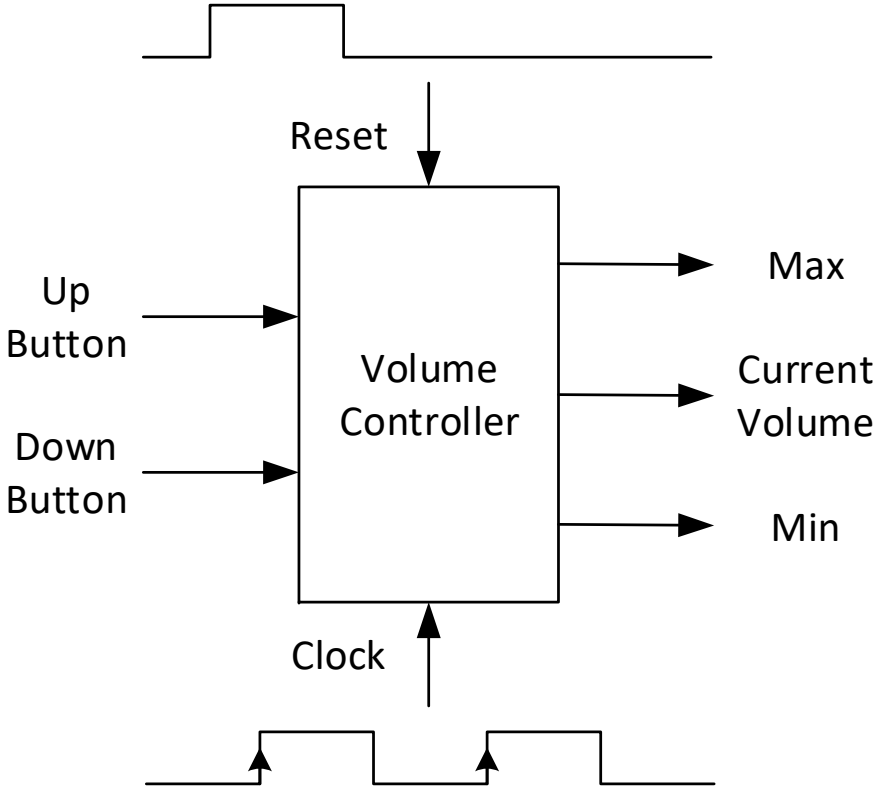
$$D_1 = Q_1.Q_0 + Q_1.\overline{Dw} + Q_0.\overline{Dw}.Up$$

Outputs

$$Min = \overline{Q_1}.\overline{Q_0}.Dw \qquad Max = Q_1.Q_0.\overline{Dw}.Up$$

Initialization (Reset)





Positive-edge-triggered D Flip-flop with Asynchronous Preset and Clear (Reset)

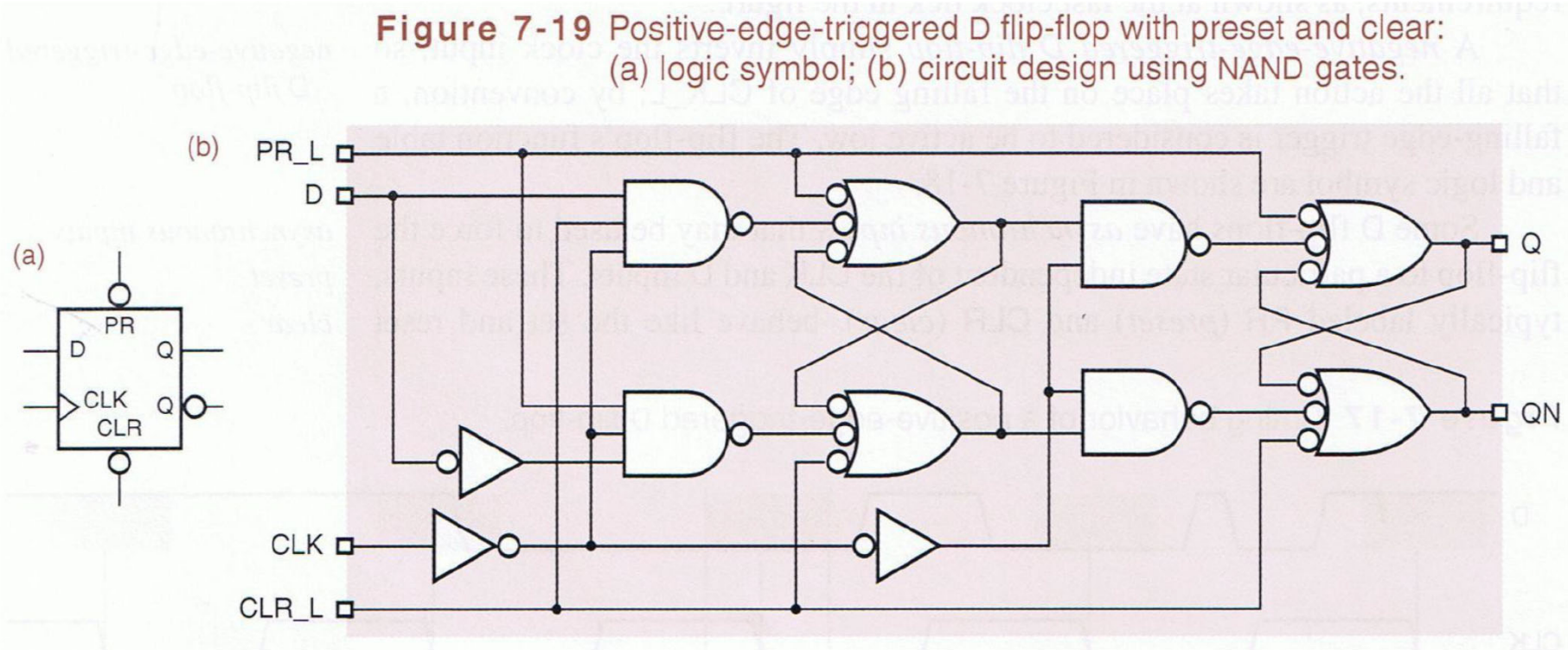
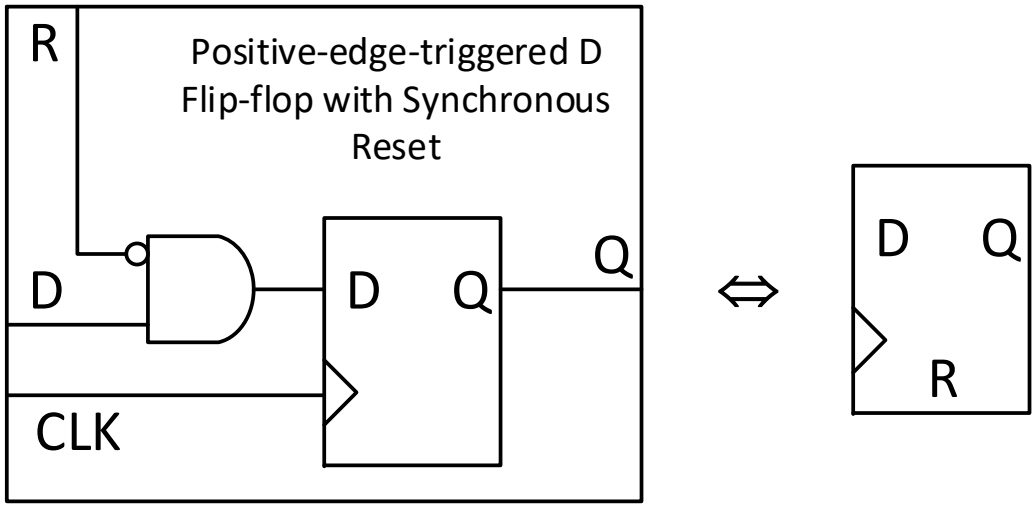


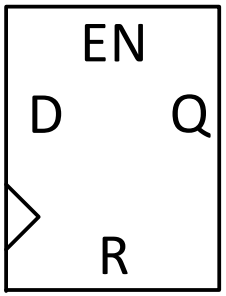
Figure 7-19 Positive-edge-triggered D flip-flop with preset and clear: (a) logic symbol; (b) circuit design using NAND gates.

Positive-edge-triggered D Flip-flop with Synchronous Clear (Reset)

- Positive-edge-triggered D Flip-flop with “gated” data input



- How to add an enable input?



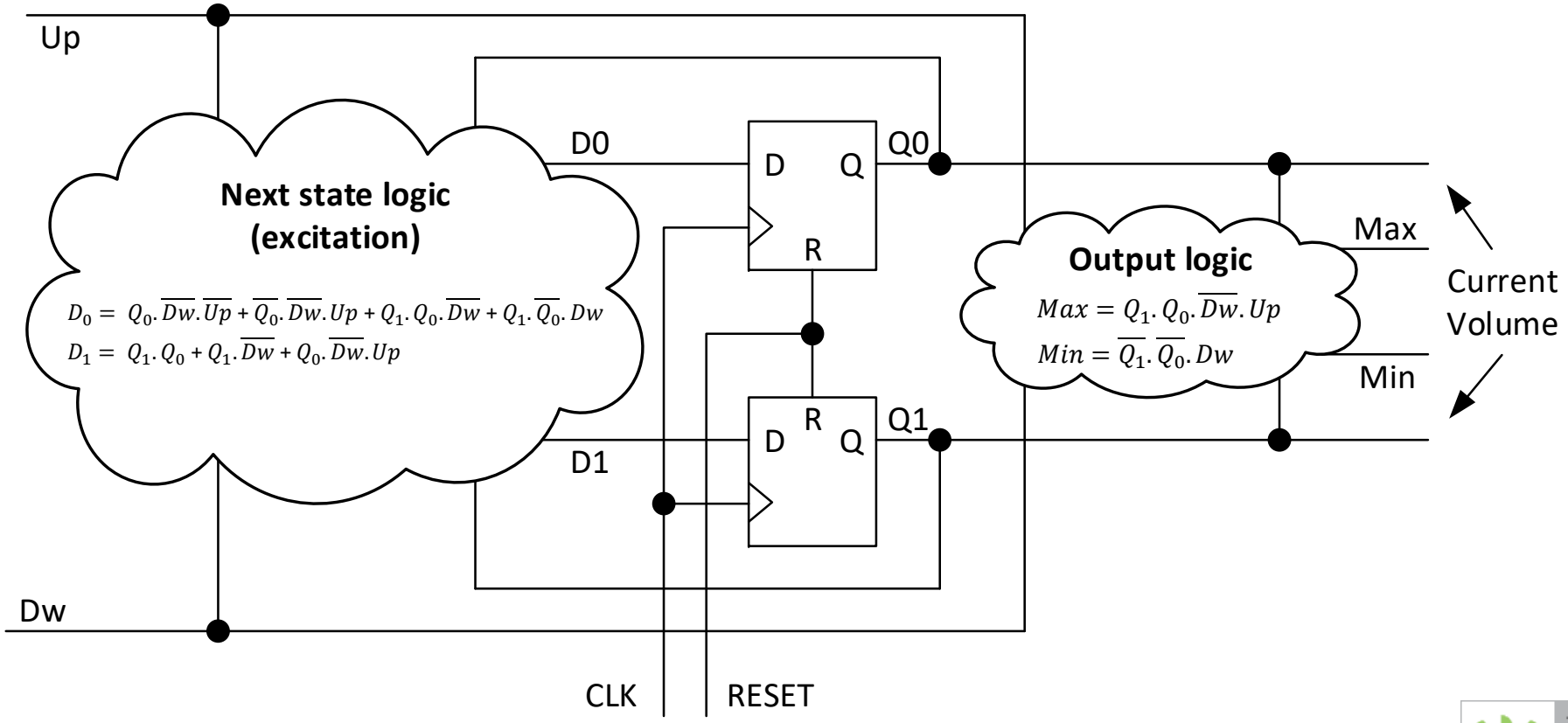
Simplified Logic Diagram

$$D_0 = Q_0.\overline{Dw}.\overline{Up} + \overline{Q_0}.\overline{Dw}.Up + Q_1.Q_0.\overline{Dw} + Q_1.\overline{Q_0}.Dw$$

$$D_1 = Q_1.Q_0 + Q_1.\overline{Dw} + Q_0.\overline{Dw}.Up$$

$$Max = Q_1.Q_0.\overline{Dw}.Up$$

$$Min = \overline{Q_1}.\overline{Q_0}.Dw$$



Gray State Encoding

Excitation / Outputs Table

Down (Dw), Up (Up)

Current State		00		01		11		10	
Meaning	Q1 Q0	D1 D0	Min, Max	D1 D0	Min, Max	D1 D0	Min, Max	D1 D0	Min, Max
Min vol.	00	00	00	01	00	00	10	00	10
	01	01	00	11	00	00	00	00	00
	11	11	00	10	00	01	00	01	00
Max vol.	10	10	00	10	01	11	00	11	00

D0

Q1 \ Dw, Up	00	01	11	10
00		1		
01	1	1		
11	1		1	1
10			1	1

Q0

D1

Q1 \ Dw, Up	00	01	11	10
00				
01		1		
11	1	1		
10	1	1	1	1

Q0

Excitation

$$D_0 = Q_0.\overline{Dw}.\overline{Up} + \overline{Q_1}.\overline{Dw}.Up + Q_1.\overline{Dw}$$

$$D_1 = Q_1.\overline{Dw} + Q_1.\overline{Q_0} + Q_0.\overline{Dw}.Up$$

Gray State Encoding		
Meaning	S	Q1 Q0
Min vol.	Vol_min	00
	Vol_low	01
	Vol_med	11
Max vol.	Vol_max	10

Outputs

$$Min = \overline{Q_1}.\overline{Q_0}.Dw \qquad Max = Q_1.\overline{Q_0}.\overline{Dw}.Up$$

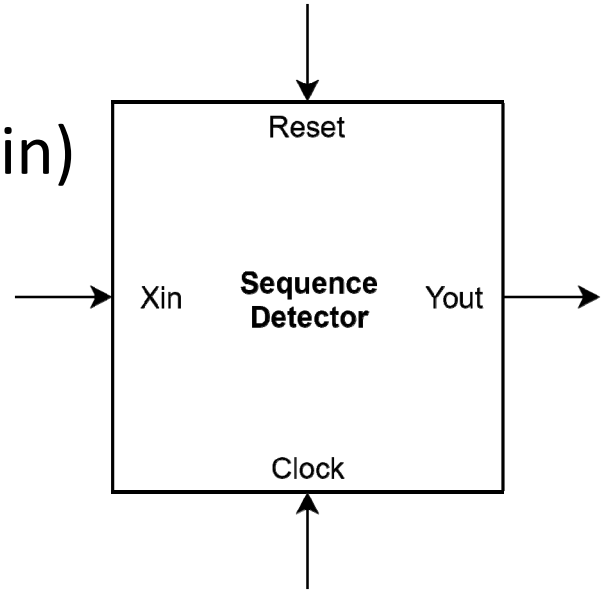
State Encoding/Assignment Guidelines/Rules

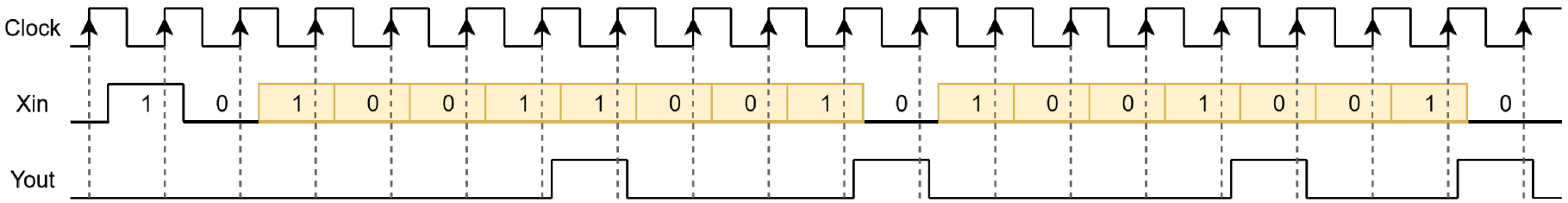
- Choose an initial coded state into which the machine can easily be forced at reset (00. . . 00 or 11. . . 11 in typical circuits).
- Minimize the number of state variables that change on each transition.
- Maximize the number of state variables that don't change in a group of related states (i.e., a group of states in which most transitions stay in the group).
- If there are unused states (i.e., if $s < 2^n$ where $n = \lceil \log_2 s \rceil$), then choose the "best" of the available state-variable combinations to achieve the foregoing goals. That is, don't limit the choice of coded states to the first s n-bit integers.
- Decompose the set of state variables into individual bits or fields, where each bit or field has a well-defined meaning with respect to the input effects or output behavior of the machine.

(subset of the rules provided in the bibliography)

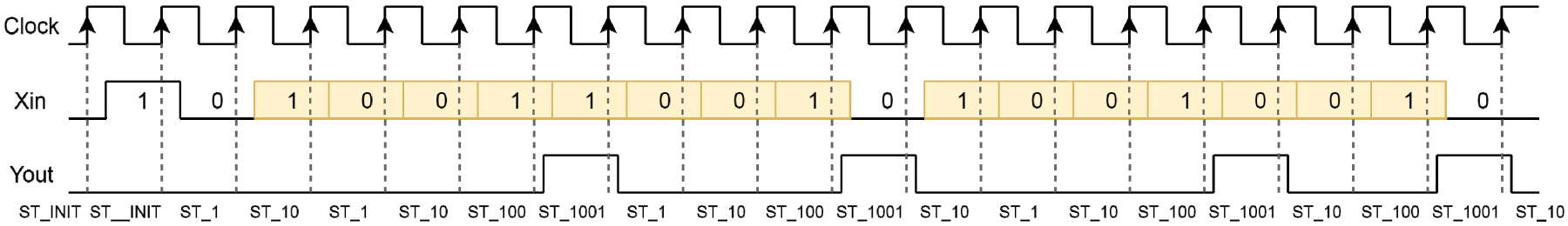
Synthesis Example 2 – Specification

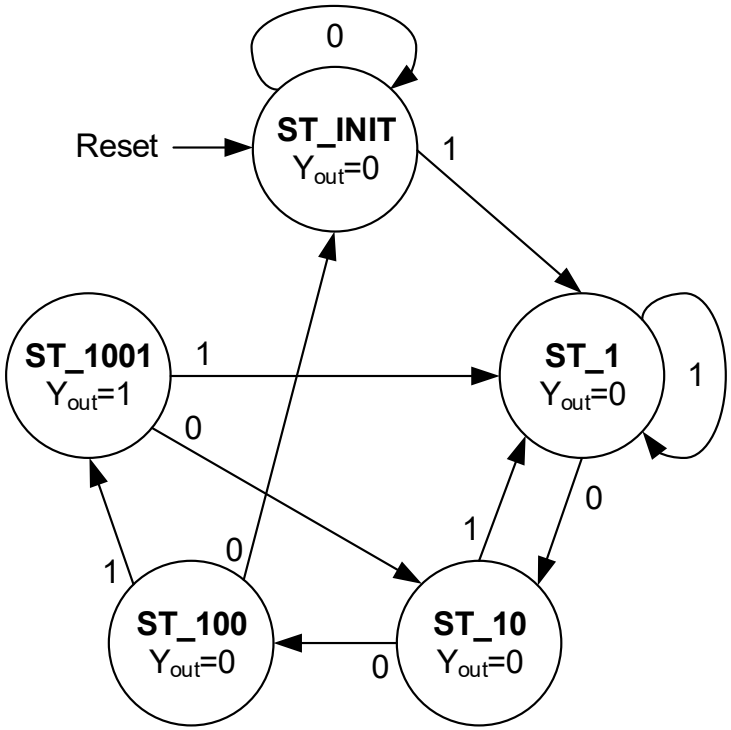
- Design a detector that activates the output (Yout) whenever the 1001 sequence is detected at the input (Xin)
- Input bits are sampled at clock tick rate
- Support overlapped sequences
- Output depends on the previous 4 input bits



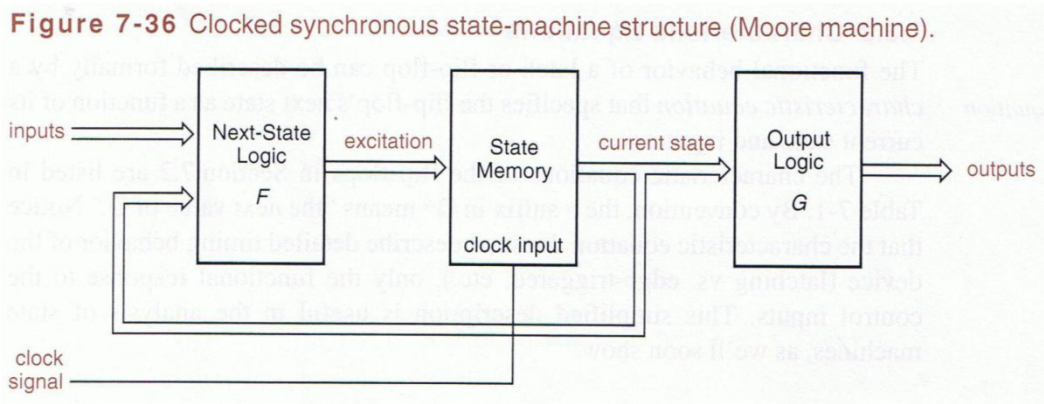


State Diagram





Moore machine! Why?



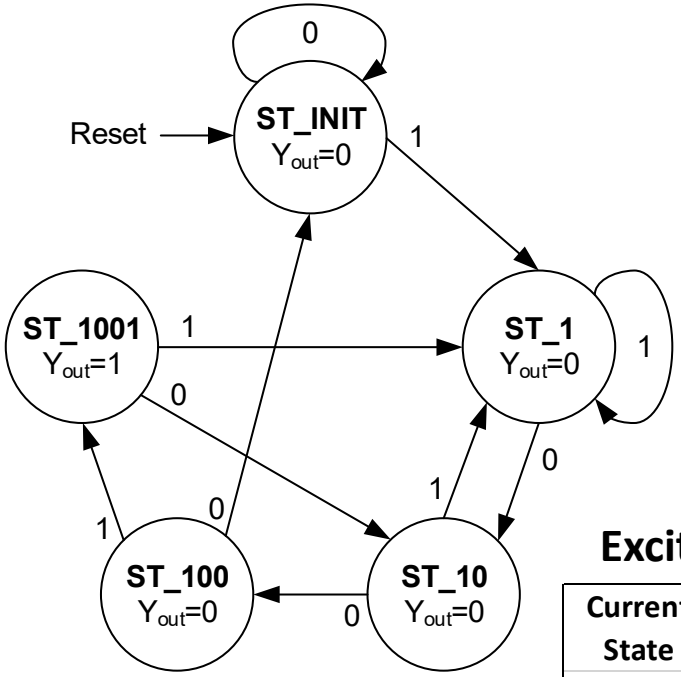
State, Transition/Output and Excitation Tables

State / Output Table

Current State	Xin		Yout
	0	1	
S	S*		
ST_INIT	ST_INIT	ST_1	0
ST_1	ST_10	ST_1	0
ST_10	ST_100	ST_1	0
ST_100	ST_INIT	ST_1001	0
ST_1001	ST_10	ST_1	1

Transition / Output Table

Current State	Xin		Yout
	0	1	
Q2 Q1 Q0	Q2* Q1* Q0*		
000	000	001	0
001	011	001	0
011	010	001	0
010	000	100	0
100	011	001	1



Using D flip-flops
Q*=D

State Encoding	
S	Q2 Q1 Q0
ST_INIT	000
ST_1	001
ST_10	011
ST_100	010
ST_1001	100

Excitation / Output Table

Current State	Xin		Yout
	0	1	
Q2 Q1 Q0	D2 D1 D0		
000	000	001	0
001	011	001	0
011	010	001	0
010	000	100	0
100	011	001	1

Minimization of Excitation/Output

Excitation / Output Table

Current State	Xin		Yout
Q2 Q1 Q0	0	1	
	D1 D1 D0		
000	000	001	0
001	011	001	0
011	010	001	0
010	000	100	0
100	011	001	1

Output

$$Y_{out} = Q_2.\overline{Q_1}.\overline{Q_0}$$

D0

Q2 \ Q1 Q0	00	01	11	10
00		1	1	1
01			1	
11				
10	1	1		

Xin

Excitation

$$D_0 = \overline{Q_2}.\overline{Q_1}.X_{in} + \overline{Q_2}.\overline{Q_1}.Q_0 + \overline{Q_2}.Q_0.X_{in} + Q_2.\overline{Q_1}.\overline{Q_0}$$

D1

Q2 \ Q1 Q0	00	01	11	10
00				1
01				1
11				
10	1			

Xin

$$D_1 = \overline{Q_2}.Q_0.\overline{X_{in}} + Q_2.\overline{Q_1}.\overline{Q_0}.\overline{X_{in}}$$

D2

Q2 \ Q1 Q0	00	01	11	10
00				
01		1		
11				
10				

Xin

$$D_2 = \overline{Q_2}.Q_1.\overline{Q_0}.X_{in}$$

Logic Diagram

$$D_0 = \overline{Q_2}.\overline{Q_1}.X_{in} + \overline{Q_2}.\overline{Q_1}.Q_0 + \overline{Q_2}.Q_0.X_{in} + Q_2.\overline{Q_1}.\overline{Q_0}$$

$$D_2 = \overline{Q_2}.Q_1.\overline{Q_0}.X_{in}$$

$$D_1 = \overline{Q_2}.Q_0.\overline{X_{in}} + Q_2.\overline{Q_1}.\overline{Q_0}.\overline{X_{in}}$$

$$Y_{out} = Q_2.\overline{Q_1}.\overline{Q_0}$$

Homework: draw the complete diagram...

Unused State Encodings
Minimal Risk

Minimal risk. This approach assumes that it is possible for the state machine somehow to get into one of the unused (or "illegal") states, perhaps because of a hardware failure, an unexpected input, or a design error. Therefore, all of the unused state-variable combinations are identified and explicit next-state entries are made so that, for any input combination, the unused states go to the "initial" state, the "idle" state, or some other "safe" state. This is an automatic consequence of some design methodologies if the initial state is coded 00. . . 00.

This is the approach used implicitly in the previous example

Current State	Xin		Yout
	0	1	
Q2 Q1 Q0	D2 D1 D0		
000	000	001	0
001	011	001	0
011	010	001	0
010	000	100	0
100	011	001	1
101	000	000	0
110	000	000	0
111	000	000	0

Unused State Encodings
Minimal Cost

Minimal cost. This approach assumes that the machine will never enter an unused state. Therefore, in the transition and excitation tables, the next-state entries of the unused states can be marked as "don't-cares." In most cases this simplifies the excitation logic. However, the machine's behavior if it ever does enter an unused state may be pretty weird.

Current State	Xin		Yout
	0	1	
Q2 Q1 Q0	D2 D1 D0		
000	000	001	0
001	011	001	0
011	010	001	0
010	000	100	0
100	011	001	1
101	XXX	XXX	X
110	XXX	XXX	X
111	XXX	XXX	X

Dn

Yout

Q2 \ Q1Q0	00	01	11	10
00				
01				
11	X	X	X	X
10			X	X

Q0 (Q1) Q2 Xin

Homework: determine the minimized equations of the previous example, assuming this approach (don't cares):

$$D_0 = ?$$

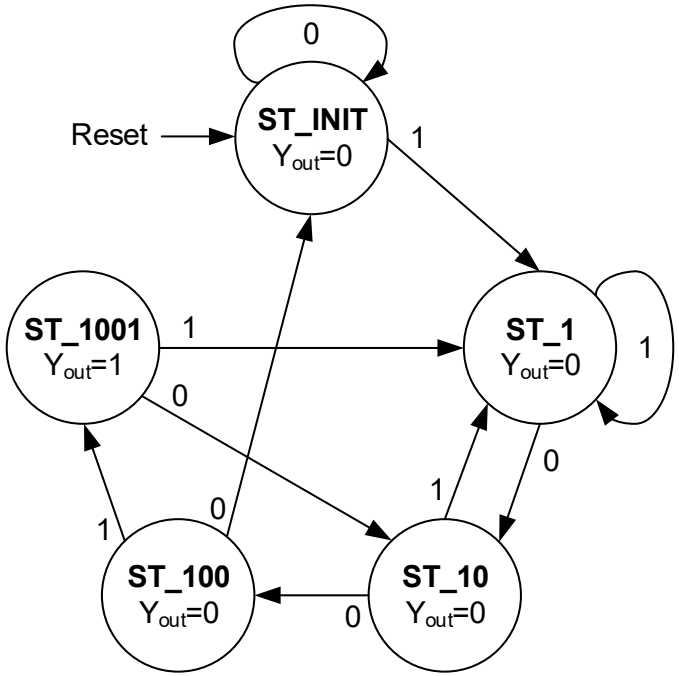
$$D_1 = ?$$

$$D_2 = ?$$

$$Y_{out} = ?$$

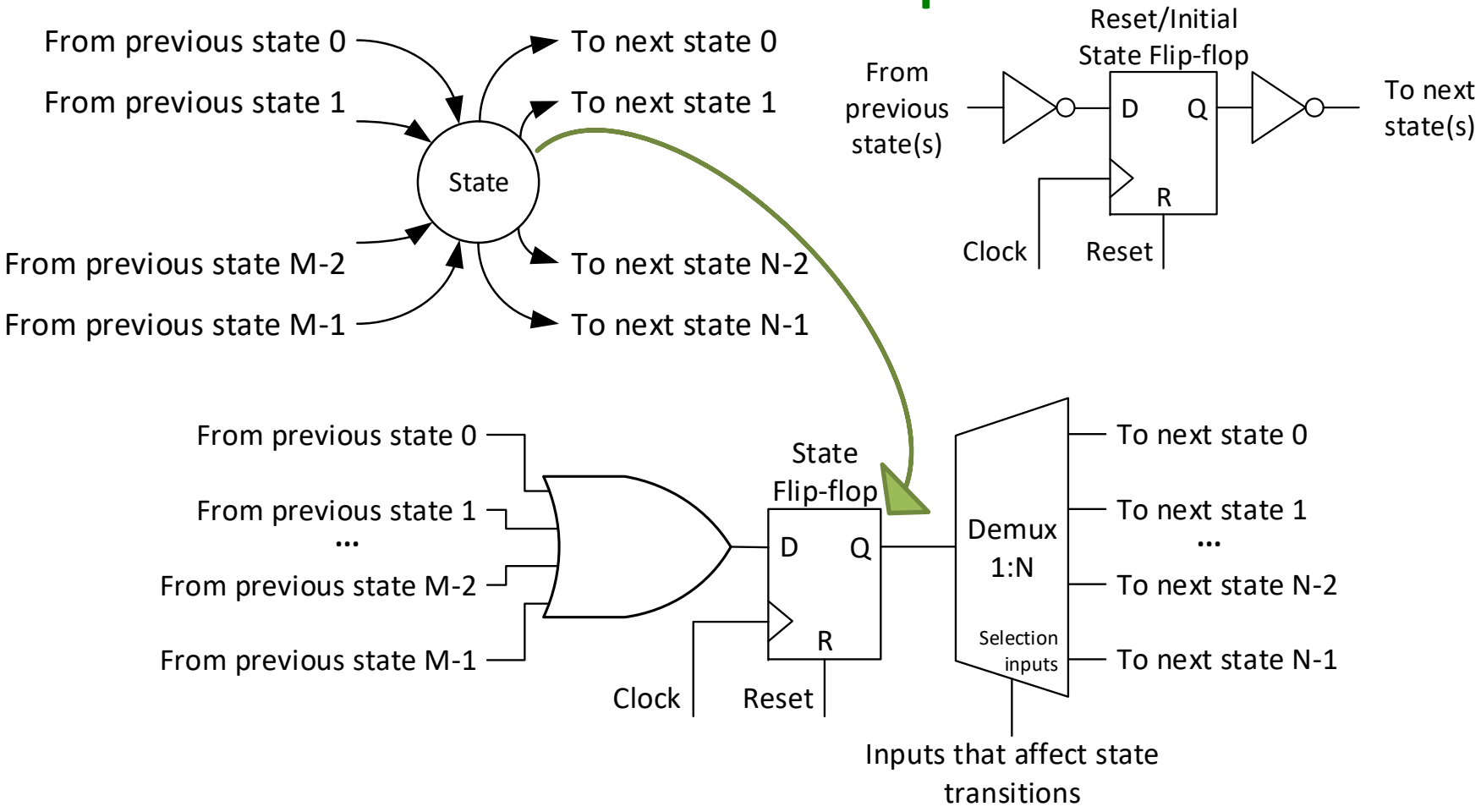
One-hot State Encoding

One flip-flop per state but straight forward synthesis leading to circuits with less and faster combinatorial logic (typically)



One-hot State Encoding	
S	**Q4 Q3 Q2 Q1 Q0**
ST_INIT	00001
ST_1	00010
ST_10	00100
ST_100	01000
ST_1001	10000

One-hot State Encoding Hardware Template



Homework: draw the logic diagram for the "1001" sequence detector based on one-hot state encoding

Conclusion

- At the end of this lecture and corresponding lab, it is fundamental to know how to design sequential circuits described by finite state machines and implemented with D type flip-flops, from state diagrams to hardware, including simulation and timing aspects
- Plan for the next lectures
  - Standard sequential circuits
    - Registers and shift registers
    - Counters
  - Iterative vs. sequential circuits

Reading chapter 7 (4th ed.) or chapter 10 (5th ed.) of *John F. Wakerly, "Digital Design – Principles and Practices", Pearson – Prentice Hall*, is highly recommended.