Sequential Logic Design Principles.Clocked Synchronous State-Machine Analysis and Synthesis

Doru Todinca

Department of Computers
Politehnica University of Timisoara

Outline

Clocked Synchronous State-Machine Analysis
- State Machine Structure
- Output Logic
- Characteristic Equations
- Analysis of State Machines with D Flip-Flops

Clocked Synchronous State Machine Design
- State-Table Design Example

Outline

Clocked Synchronous State-Machine Analysis
- State Machine Structure
- Output Logic
- Characteristic Equations
- Analysis of State Machines with D Flip-Flops

Clocked Synchronous State Machine Design
- State-Table Design Example

Introduction

- *State machines* is a generic name for sequential circuits (i.e., circuits with states)
- *clocked* means that the storage elements of these state machines (i.e., the flip-flops or latches) have a *clock* input
- *synchronous* means that all flip-flops use the same clock signal $\Rightarrow$ these state machines change state only at the "tick" of the clock signal ("tick" = rising or falling edge of the clock, the pulse of the clock, etc)
- We will consider next that the storage elements of the state machines are positive edge-triggered flip-flops
- Nowadays mostly D flip-flops are used, but in the past the J-K flip-flops have been also popular for state machine implementation

Outline

Clocked Synchronous State-Machine Analysis
- State Machine Structure
- Output Logic
- Characteristic Equations
- Analysis of State Machines with D Flip-Flops

Clocked Synchronous State Machine Design
- State-Table Design Example

General Structure

- Figure 1 presents the general structure of a clocked synchronous state machine
- The *state memory* is a set of n flip-flops that store the current state of the machine; it has 2^n states
- All flip-flops are connected to a common clock signal and they change state in the same time (at the positive edge of the clock)
- The *next state* of the machine is determined by the *next-state logic F*, being a function of the current states and current inputs
- The *output logic G* determines the output of the state machine as a function of the current state and inputs
- Both the *next state logic F* and *output logic G* are pure combinational circuits
- We can write:
  - Next state = F(current state, input)
  - Output = G(current state, input)

Outline

Clocked Synchronous State-Machine Analysis
- State Machine Structure
- Output Logic
- Characteristic Equations
- Analysis of State Machines with D Flip-Flops

Clocked Synchronous State Machine Design
- State-Table Design Example

Output Logic: Mealy and Moore Machines

Definition

A sequential circuit whose output depends on both state and input is called a *Mealy machine*. Figure 1 shows a Mealy machine.

Definition

A sequential circuit whose output depends on the state alone is called a *Moore machine*. Figure 2 shows the general structure of a Moore machine.
For the Moore machines we can write:
Output = G(current state)

In practice there are variations of Mealy and Moore type machines.

Output Logic: Mealy Machine

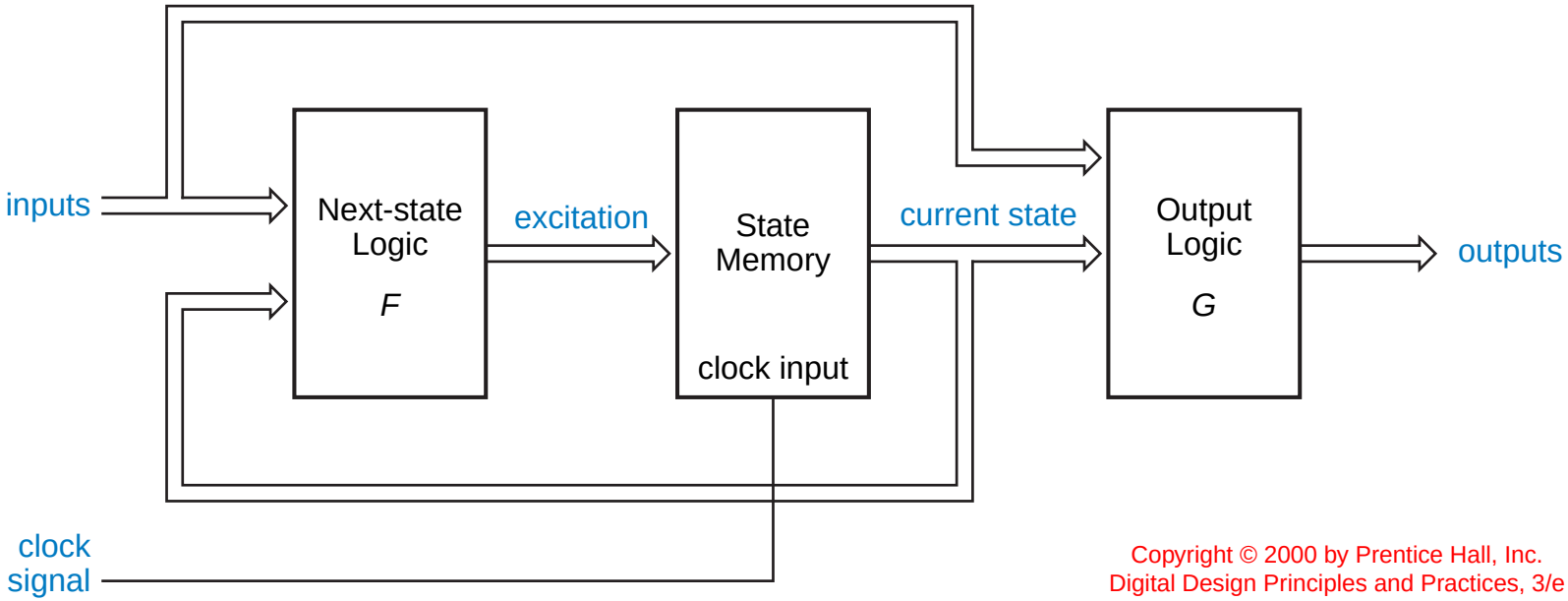

Figure 1 : Clocked synchronous state-machine structure (Mealy machine)

Output Logic: Moore Machine

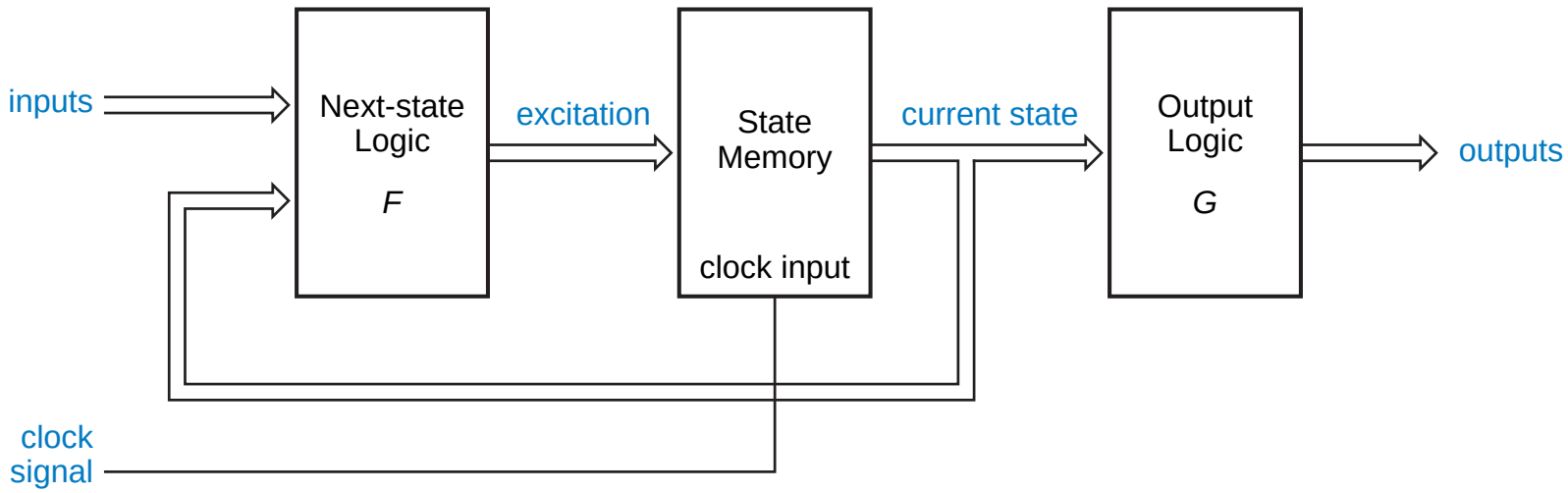

Figure 2 : Clocked synchronous state-machine structure (Moore machine)

Output Logic: Mealy and Moore Machines

- For high-speed circuits it is important to make the outputs available as early as possible.
- A possibility to do this is to encode the state so that the state variables themselves are outputs
- This method is called output-coded state assignment
- The resulting machine is a Moore machine where the output logic G of figure 2 is only wires (no gates)
- Another approach is to design the state machine so that the outputs during one clock period depend on the stare and inputs during the *previous* clock period
- This is called state machine with *pipelined outputs* (represented in figure 3 for a Mealy machine)
- The outputs are obtained by attaching another state of memory (i.e., flip-flops) to a machine's outputs.
- The concept of *pipeline* is very important in computer architectures and in computer engineering in general.
- *Different types of machines models can be transformed into each other.*

Mealy Machine with Pipelined Outputs

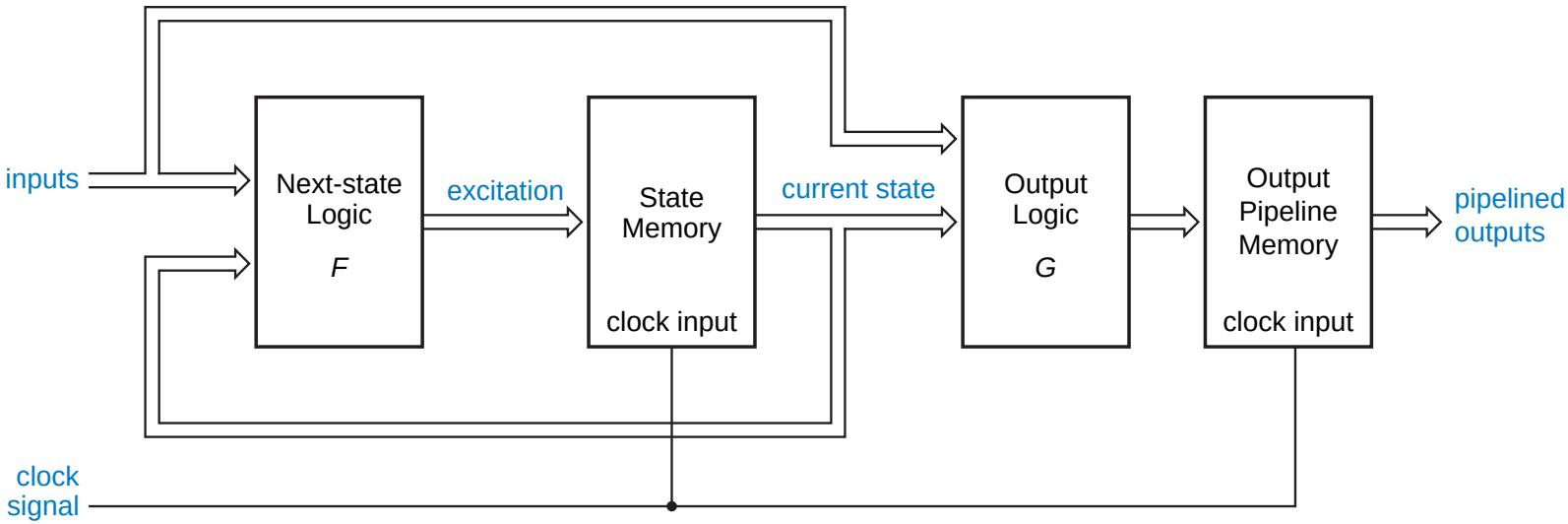

Figure 3 : Mealy machine with pipelined outputs

Outline

Clocked Synchronous State-Machine Analysis
- State Machine Structure
- Output Logic
- Characteristic Equations
- Analysis of State Machines with D Flip-Flops

Clocked Synchronous State Machine Design
- State-Table Design Example

Latch and Flip-Flop Characteristic Equations

Definition

The *characteristic equation* of a latch or flip-flop specifies its next state as a function of its current state and inputs.

- The characteristic equation describes formally the *functional* behavior of a latch or flip-flop (the functional response to the control inputs)
- The characteristic equation does not describe the detailed timing behavior of the device (latch or flip-flop)
- Characteristic equation is useful in the analysis of state machines
- *Notation:* by convention, the suffix * in the expression Q* means "the next value of Q"
- Table 7-1 (from figure 4) contains the characteristic equations for the studied latches and flip-flops.

Latch and Flip-Flop Characteristic Equations

Device Type	*Characteristic Equation*
S-R latch	$Q* = S + R' \cdot Q$
D latch	$Q* = D$
Edge-triggered D flip-flop	$Q* = D$
D flip-flop with enable	$Q* = EN \cdot D + EN' \cdot Q$
Master/slave S-R flip-flop	$Q* = S + R' \cdot Q$
Master/slave J-K flip-flop	$Q* = J \cdot Q' + K' \cdot Q$
Edge-triggered J-K flip-flop	$Q* = J \cdot Q' + K' \cdot Q$
T flip-flop	$Q* = Q'$
T flip-flop with enable	$Q* = EN \cdot Q' + EN' \cdot Q$

Table 7-1
Latch and flip-flop characteristic equations.

Figure 4 : Latch and flip-flop characteristic equations

Outline

Clocked Synchronous State-Machine Analysis
- State Machine Structure
- Output Logic
- Characteristic Equations
- Analysis of State Machines with D Flip-Flops

Clocked Synchronous State Machine Design
- State-Table Design Example

Analysis of State Machines: What We Need to Know ?

- Remember that a "state" embodies all we need to know about the past history of a sequential circuit
- From the equation *Next state = F(current state, input)* results that, if we want to know the evolution of the circuit, it is sufficient if we know the current state and input of the circuit
- The equation *Output = G(current state, input)* tells us that the output can be determined from the current state and input
- Hence, if we know the next state and output function of a sequential circuit, we can predict its future behavior and determine its output, from circuit's current state and input
- *The goal of sequential circuit analysis is to determine the next state and output functions*

Steps of Clocked Synchronous State Machine Analysis

1. Determine the next state and output functions F and G
2. Use F and G to construct a *state/output table* that completely specifies the next state and output of the circuit for every possible combination of current state and input
3. (Optional) Draw a *state diagram*, that presents the information from the previous step in graphical form

Clocked Synchronous State Machine Example

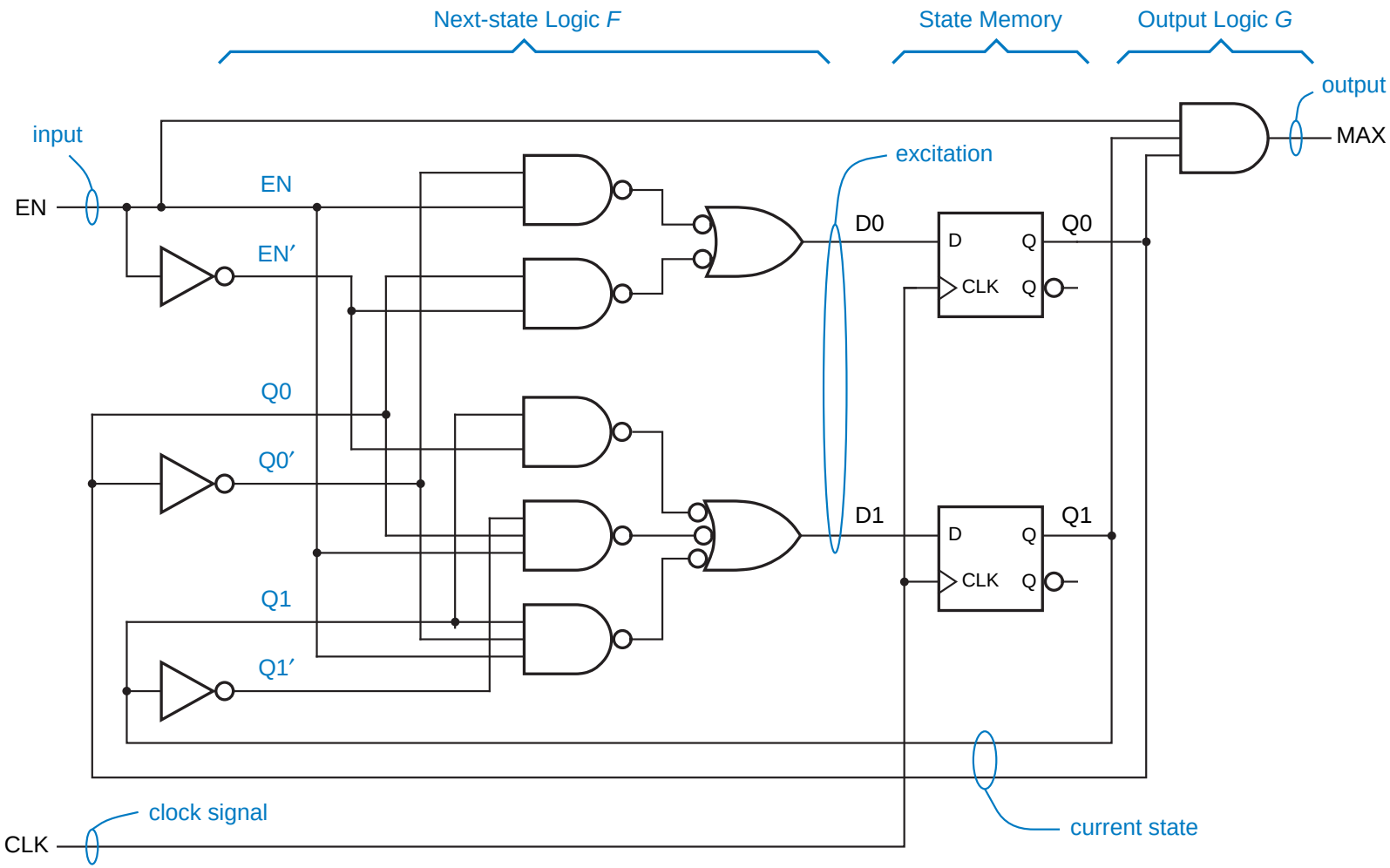

Figure 5 : Clocked synchronous state machine using positive-edge-triggered D flip-flops

Example of Analysis of a Clocked Synchronous State Machine

- Figure 5 shows a simple state machine, with only two flip-flops (positive-edge-triggered D flip-flops)
- At each rising edge of the clock, each flip-flop works according to the characteristic equation: $Q* = D$
- Hence, to determine next value of Q, we must first determine the current value of D
- The signals at the outputs of the two flip-flops from fig 5 are named Q0 and Q1
- Q0 and Q1 are the state variables, their value (i.e., Q1 Q0) is the current state of the machine
- The input signals of the two flip-flops are named D1 and D0
- These signals provide an *excitation* for the D flip-flop at each clock tick
- Definition: logic equations that express the excitation signals as functions of the current state and input are called *excitation equations*

Example of Analysis of a Clocked Synchronous State Machine

For our example, the excitation equations are:

$$D0 = Q0 \cdot EN' + Q0' \cdot EN$$
$$D1 = Q1 \cdot EN' + Q1' \cdot Q0 \cdot EN + Q1 \cdot Q0' \cdot EN$$

The next state function of the example machine with equations for the next value of the state variables is:

$$Q0* = D0$$
$$Q1* = D1$$

Substituting the excitation equations for D0 and D1, we have:

$$Q0* = Q0 \cdot EN' + Q0' \cdot EN$$
$$Q1* = Q1 \cdot EN' + Q1' \cdot Q0 \cdot EN + Q1 \cdot Q0' \cdot EN$$

Transition Equations

Definition

These equations, which express the next value of the state variables as a function of current state and input, are called *transition equations*.

- For each combination of current states and input variables, the transition equations predict the next state.
- In this example, each state is described by two bits, the current values of Q1 and Q0
- For each state, there are possible only two input values: $EN = 0$ or $EN = 1$
- It means that there are a total of 8 state/input combinations
- In general, a machine with s state bits and i inputs has 2^{s+i} state/input combinations

Transition Tables

- Table 7-2 (a) shows the *transition table* that is created by evaluating the transition equations for every possible state/input combination
- A transition table lists the states along the left and the input combinations along the top of the table, like in example
- From the transition table we can infer that the function of the example machine is a 2-bit binary counter with an enable input named EN
- When EN=0, the state remains unchanged, when EN=1, the count advances by 1 at each clock tick
- When the counter value is 11 (its maximum), at the next clock tick, it will roll over to 00

State Tables

- We may assign alphanumeric *state names* to each state
- The simples naming is $00 = A$, $01 = B$, $10 = C$, $11 = D$
- Replacing the alphanumeric names in the transition table from table 7-2 (a), we obtain the *state table* in 7-2 (b)
- In the state table, S denotes the current state, and $S*$ the next state of the state machine
- A state table is in general easier to understand than a transition table, because in complex machines we can use states that have a meaning
- But a state table contains less information than the transition table because it does not indicate the *state encoding* (the binary values assigned to state variables in each state)

State/Output Tables

- Now, after producing the state table we have to analyze only the output logic of the machine
- In the example, there is only one output, of Mealy type
- The *output equation* is $MAX = Q1 \cdot Q0 \cdot EN$
- If we combine the output behavior with the next state information, we will obtain the *state/output table* (shown in Table 7-2 (c))
- For a Moore machine, the state/output tables are simpler, because in each state can be only one output value
- If we modify our example and remove the EN signal from the output AND gate that produces the MAX output, we obtain a Moore machine producing the MAXS output with the equation: $MAXS = Q1 \cdot Q0$
- The state/output table for the Moore machine is shown in Table 7-3 (figure 7)

Transition, State and State/Output Tables

(a)

	EN	
Q1 Q0	**0**	**1**
00	00	01
01	01	10
10	10	11
11	11	00
	Q1* Q0*	

(b)

	EN	
S	**0**	**1**
A	A	B
B	B	C
C	C	D
D	D	A
	S*	

(c)

	EN	
S	**0**	**1**
A	A, 0	B, 0
B	B, 0	C, 0
C	C, 0	D, 0
D	D, 0	A, 1
	S*, MAX	

Table 7-2
Transition, state, and state/output tables for the state machine in Figure 7-38.

Figure 6 : (a) Transition table, (b) state table, and (c) state/output table for the state machine of figure 5

State/Output Table for a Moore Machine

Table 7-3
State/output table for a Moore machine.

	EN		
S	**0**	**1**	**MAXS**
A	A	B	0
B	B	C	0
C	C	D	0
D	D	A	1
	S*		

Figure 7 : State/output tables for a Moore machine

State Diagram for the State Machine of Table 7.2

- A *state diagram* presents the informations from the state/output table in a graphical form
- It has one *node* (circle) for each state and a *directed arc* (an arrow) for each transition (see figure 8 for the state diagram for our example state machine)
- The letter inside each node is the state name
- Each arrow leaving a given state points to the next state for a given input combination
- The arrow also shows the output value produced in the given state for that input combination
- Sometimes the notation on the arrow is different: input/output
- For our example: (EN=0)/(MAX=0) or even 0/0, without specifying the names for input and output

State Diagram for the State Machine of Table 7.2

- For a Mealy machine, the state-diagram notation is somehow misleading, because the output is produced continuously, when the machine is in the indicated state and has the indicated input, not just during the transition to the next state
- For a Moore machine the state diagram is simpler, the output values being shown inside each node (circle), since the outputs are functions of state only
- Figure 9 shows the state diagram of the Moore machine from our example

State Diagram for the State Machine of Table 7.2

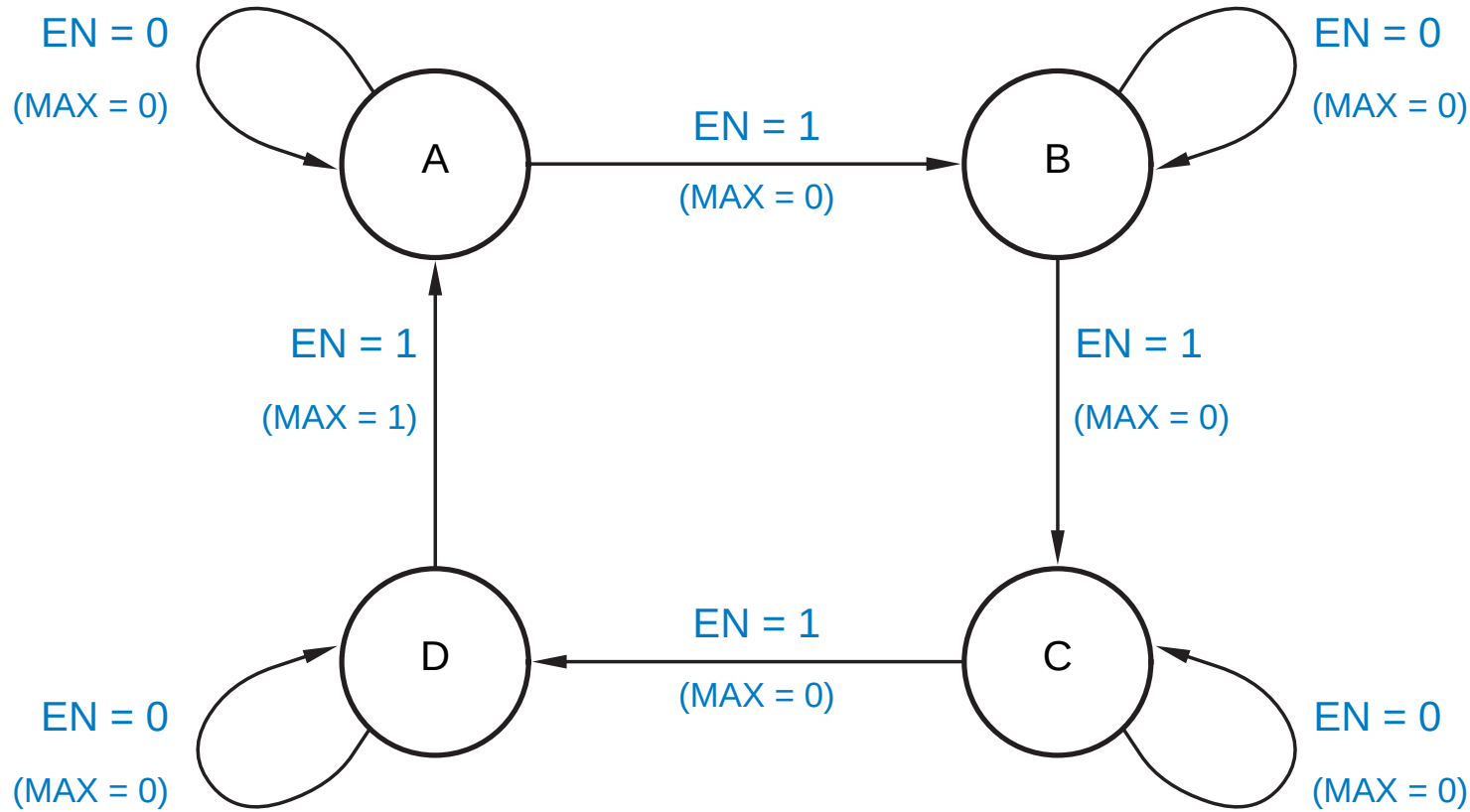

Figure 8 : State diagram corresponding to the state machine of Table 7.2 (from figure 6)

State Diagram for the State Machine of Table 7.3

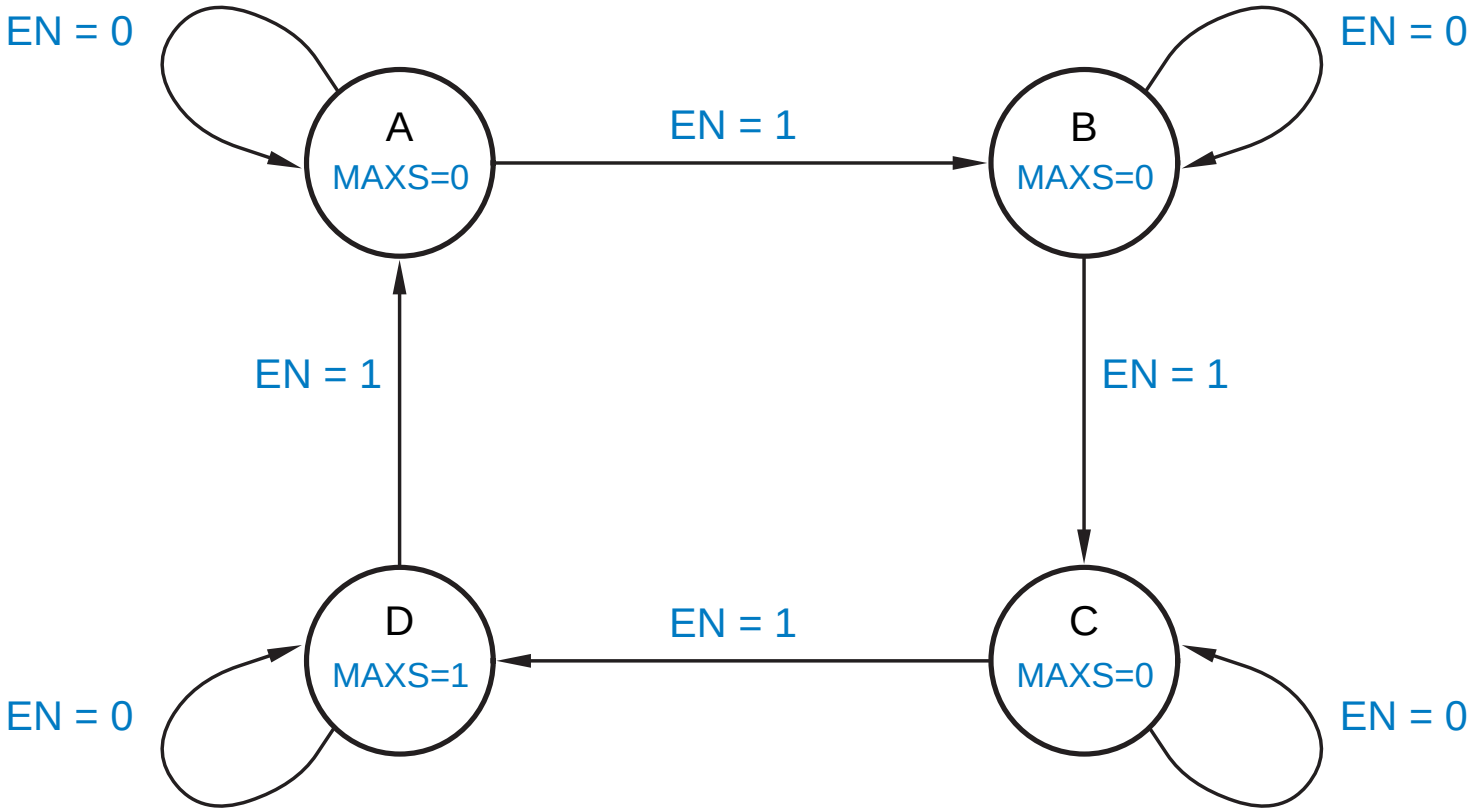

Figure 9 : State diagram corresponding to the state machine of Table 7.3 (from figure 7)

- The logic diagram for the state machine in our example was laid out to match the conceptual model of a Mealy machine
- However, the information about the state machine can be extracted even if the logic diagram looks differently, without grouping in this way next-state logic, state memory and output logic
- For example, figure 10 shows another logic diagram for the same state machine
- The only circuit difference is that in the new diagram we have used flip-flop's QN outputs in order to save a couple of inverters
- We can also construct a timing diagram for our state machine example, if we consider a starting state and imagine a sequence of inputs
- Figure 11 shows such a timing diagram

Redrawn Logic Diagram for a Clocked Synchronous State Machine

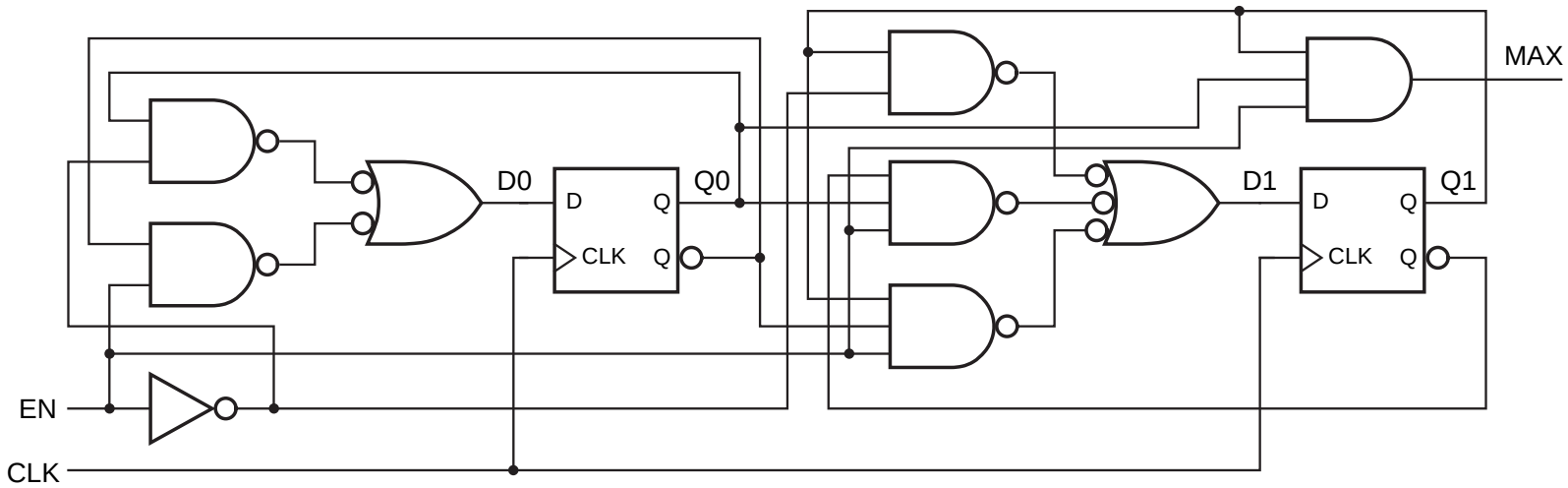

Figure 10 : Redrawn logic diagram for a clocked synchronous state machine

Timing Diagram for Example State Machine

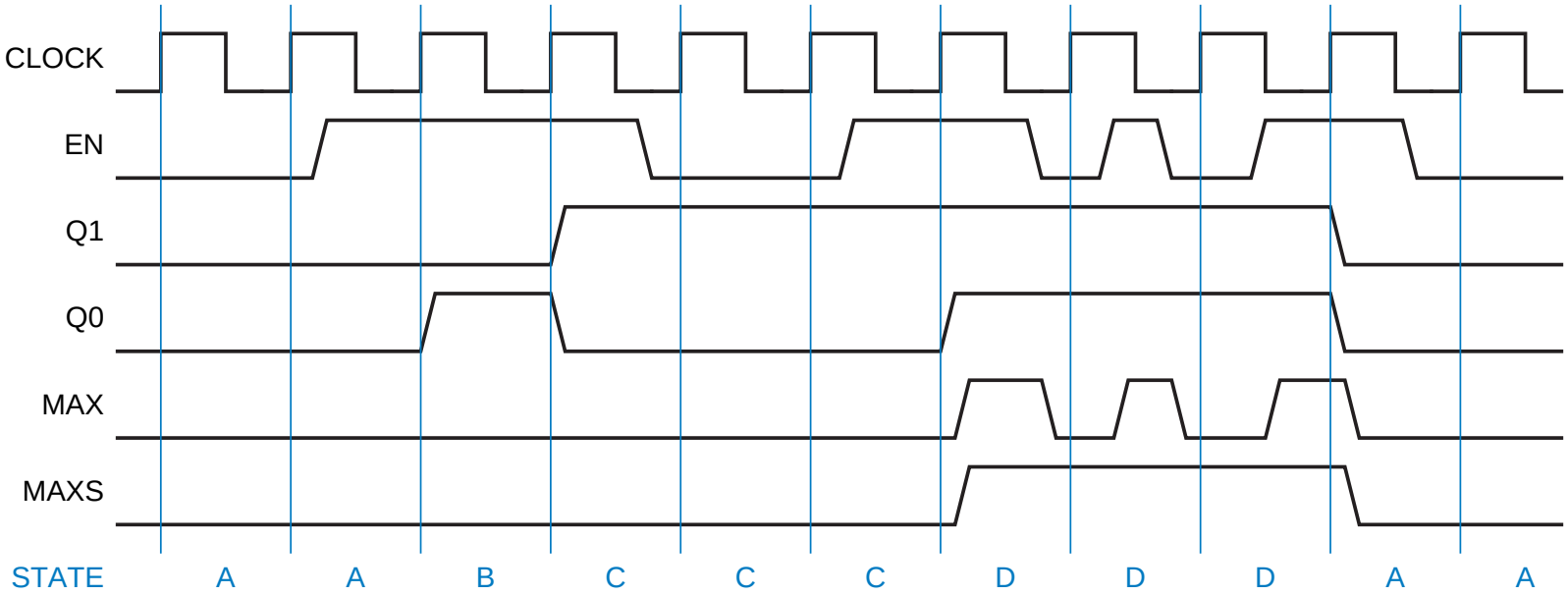

Figure 11 : Timing diagram for example clocked synchronous state machine

Detailed Steps for Analyzing of a Clock Synchronous State Machine

1. Determine the excitation equations for the flip-flop control inputs
2. Substitute the excitation equations into the flip-flop characteristic equations to obtain transition equations
3. Use the transition equations to construct a transition table
4. Determine the output equations
5. Add output values to the transition table for each state (Moore) or state/input combination (Mealy) to create a transition/output table
6. Name the states and substitute state names for state-variable combinations in the transition/output table to obtain a state/output table
7. (Optional) Draw a state diagram corresponding to the state/output table

Another Example of Clocked Synchronous State Machine

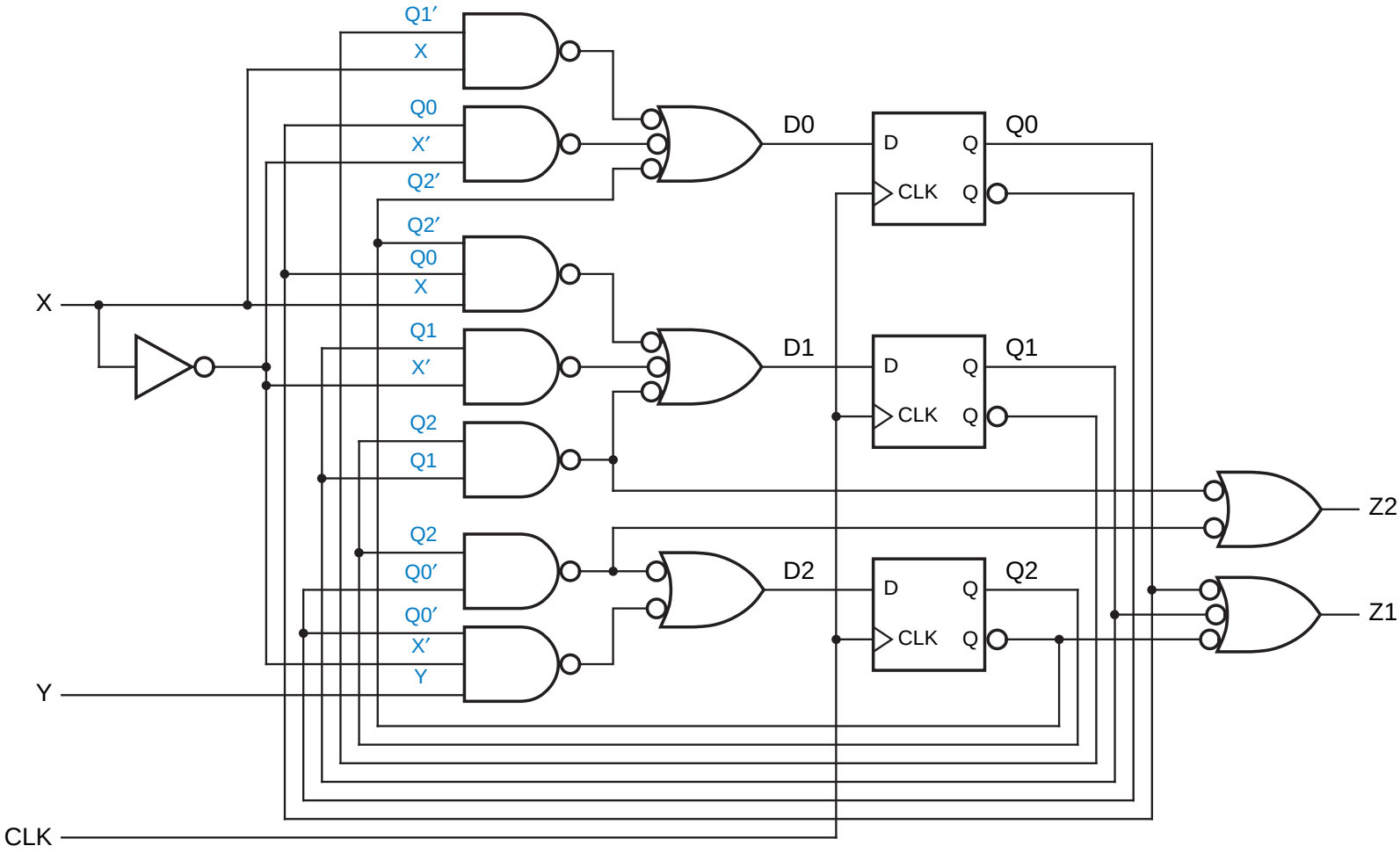

Figure 12 : Clocked synchronous state machine with three flip-flops and eight states

Another Example of Clocked Synchronous State Machine

We apply the previous steps to analyze the state machine from figure 12
From the logic diagram, we find the excitation equations:

$$D0 = Q1' \cdot X + Q0 \cdot X' + Q2$$
$$D1 = Q2' \cdot Q0 \cdot X + Q1 \cdot X' + Q2 \cdot Q1$$
$$D2 = Q2 \cdot Q0' + Q0' \cdot X' \cdot Y$$

Substituting into the characteristic equations for D flip-flops, we obtain the transition equations:

$$Q0* = Q1' \cdot X + Q0 \cdot X' + Q2$$
$$Q1* = Q2' \cdot Q0 \cdot X + Q1 \cdot X' + Q2 \cdot Q1$$
$$Q2* = Q2 \cdot Q0' + Q0' \cdot X' \cdot Y$$

Another Example of Clocked Synchronous State Machine

From the logic diagram we can write the two output equations:

$Z1 = Q2 + Q1' \cdot Q0$
$Z2 = Q2 \cdot Q1 + Q2 \cdot Q0'$

From the transition equations and output equations we obtain the transition/output table from Table 7-4 (a) (figure 13)

Assigning state names from A to H, we obtain the state/output table from Table 7-4 (b) (figure 13)

Transition/Output and State/Output Table for the Eight State State Machine

Table 7-4 Transition/output and state/output tables for the state machine in Figure 7-43.

(a)

	X Y				
Q2 Q1 Q0	00	01	10	11	Z1 Z2
000	000	100	001	001	10
001	001	001	011	011	10
010	010	110	000	000	10
011	011	011	010	010	00
100	101	101	101	101	11
101	001	001	001	001	10
110	111	111	111	111	11
111	011	011	011	011	11
	Q2* Q1* Q0*				

(b)

	X Y				
S	00	01	10	11	Z1 Z2
A	A	E	B	B	10
B	B	B	D	D	10
C	C	G	A	A	10
D	D	D	C	C	00
E	F	F	F	F	11
F	B	B	B	B	10
G	H	H	H	H	11
H	D	D	D	D	11
	S*				

Figure 13 : Transition/output and state/output tables for the state machine in figure 12

State Diagram for the State Machine of Table 7.4

- A state diagram for this example is shown in figure 14
- The output values are written on each state, and each arc is labeled with a *transition expression*
- A transition is taken for input combinations for which the transition expression is 1
- Transitions labeled 1 are always taken
- The transition expressions on arcs leaving a particular state must be mutually exclusive and all-inclusive
- *Mutually exclusive:* no two transition expression can equal 1 for the same input combination, since a machine can't have two next states for one input combination
- *All inclusive:* for every possible input combination, some transition expression must equal 1, so that all next states are defined.

State Diagram for the State Machine of Table 7.4

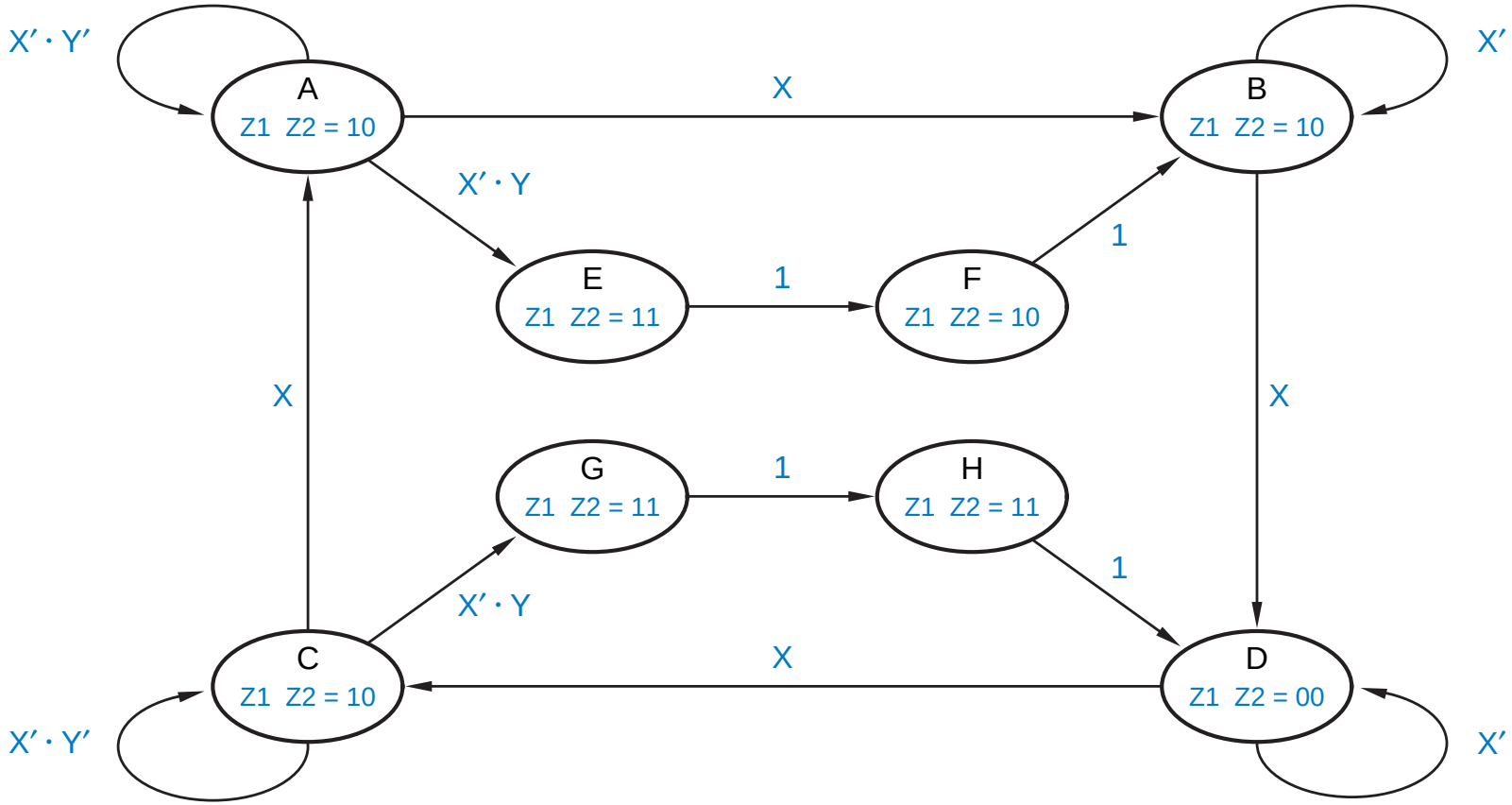

Figure 14 : State diagram corresponding to the state machine of Table 7.4 (from figure 13)

Outline

Clocked Synchronous State-Machine Analysis
- State Machine Structure
- Output Logic
- Characteristic Equations
- Analysis of State Machines with D Flip-Flops

Clocked Synchronous State Machine Design
- State-Table Design Example

The Steps for Designing a Clocked Synchronous State Machine

1. Construct a state/output table corresponding to the word description or specification, using mnemonic names for states
2. (Optional) Minimize the number of states
3. Choose a set of state variables and assign a state-variable combination to the named states
4. Substitute the state-variable combination 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
5. Choose a flip-flop type
6. Construct an excitation table
7. Derive excitation equations from the excitation table
8. Derive output equations from the transition/output table
9. Draw a logic diagram for the circuit

The Steps for Designing a Synchronous State Machine. Comments

- Step 1 is the most important, being the most creative: the designer has to translate a possibly ambiguous word description into a formal tabular description
- Step 2 is hardly ever performed, so we will not discuss it
- Step 3 is important, since not always the most simple state assignment (state coding) does not always lead to the simplest excitation equations, output equations, and resulting logic circuit (will be discussed in the future)
- Step 5: most often we use D flip-flops, but we can use also J-K flip-flops
- Step 6: for D flip-flops, the excitation table and the state table are identical, so we can call it *transition/excitation and output table*
- For J-K flip-flops, excitation tables and transition tables are different

More Comments on State Assignment (Step 3)

- If there are *unused states* (i.e., if the number of states available with n flip-flops exceeds the number of states required s), there are two reasonable approaches:
  1. *Minimal risk:* all unused state-variable are identified and explicit next-state entries are made so that, for any input combination, the unused states go to the "initial", or "idle", or any "safe" state (usually coded $00\ldots00$
  2. *Minimal cost:* the next-state entries for the unused states can be marked "don't cares"
- Minimal risk approach assumes that somehow, maybe due to hardware failure or unexpected input, or design error, the circuit can go into an "illegal" state, and, accordingly, makes the circuit to go to the initial state from that illegal state
- The minimal cost approach assumes that the machine will never enter an unused states. If somehow the machine *does* go to an unused state, its behavior can be weired !

Outline

Clocked Synchronous State-Machine Analysis
- State Machine Structure
- Output Logic
- Characteristic Equations
- Analysis of State Machines with D Flip-Flops

Clocked Synchronous State Machine Design
- State-Table Design Example

State-Table Design Example: "1s-counting machine"

Design a clocked synchronous state machine with two inputs, X and Y, and one output, Z. The output should be 1 if the number of 1 inputs on X and Y since reset is a multiple of 4, and 0 otherwise.

- Since we count the number of 1s received modulo 4, it means that 4 states are sufficient.
- We name the states $S0 - S3$: $S0$ is the initial state and the total number of 1s received in state Si is i modulo 4.
- The number of state variables is 2, with no unused states. (Explain why !)
- Table 7-12 (figure 15) is the resulted state and output table

State and Output Table for 1s Counting Machine

Table 7-12
State and output table for 1s-counting machine.

		XY				
Meaning	***S***	***00***	***01***	***11***	***10***	***Z***
Got zero 1s (modulo 4)	S0	S0	S1	S2	S1	1
Got one 1 (modulo 4)	S1	S1	S2	S3	S2	0
Got two 1s (modulo 4)	S2	S2	S3	S0	S3	0
Got three 1s (modulo 4)	S3	S3	S0	S1	S0	0
		$S*$				

Figure 15 : State and output tables for the 1s counting machine

State Assignment

- We assign coded states to the named states in Karnaugh-map order (00, 01, 11, 10) for two reasons:
  1. In this state table, it minimizes the number of state variables that change for most transitions, potentially simplifying the excitation equations
  2. It simplifies the mechanical transfer of information to excitation maps
- A transition/excitation table based on this state assignment is shown in Table 7-13 (figure 16)
- The corresponding Karnaugh maps for D1 and D2 are shown in figure 17
- The excitation equations can be read from the maps, and the output equation can be read directly from the transition/excitation table

Transition/Excitation and Output Table for 1s Counting Machine

	XY				
Q1 Q2	**00**	**01**	**11**	**10**	**Z**
00	00	01	11	01	1
01	01	11	10	11	0
11	11	10	00	10	0
10	10	00	01	00	0
	Q1* Q2* or D1 D2				

Table 7-13
Transition/excitation and output table for 1s-counting machine.

Figure 16 : Transition/excitation and output tables for the 1s counting machine

Excitation Maps for 1s Counting Machine

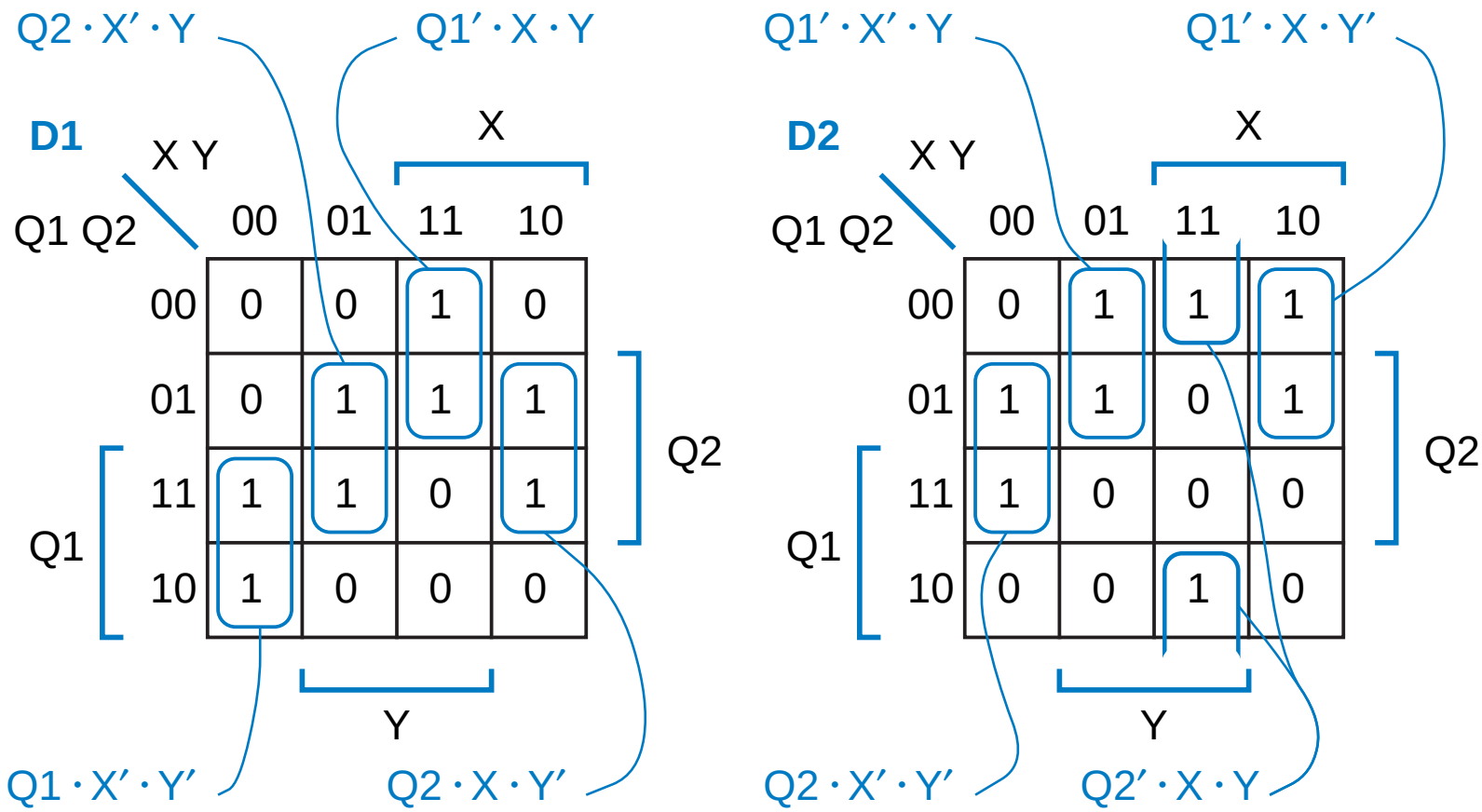

Figure 17 : Excitation maps for D1 and D2 inputs in 1s counting machine

Excitation and Output Equations

$$D1 = Q2 \cdot X' \cdot Y + Q1' \cdot X \cdot Y + Q1 \cdot X' \cdot Y' + Q2 \cdot X \cdot Y'$$
$$D2 = Q1' \cdot X' \cdot Y + Q1' \cdot X \cdot Y' + Q2 \cdot X' \cdot Y' + Q2' \cdot X \cdot Y$$
$$Z = Q1' \cdot Q2'$$

A logic diagram using D flip-flops and AND-OR or NAND-NAND gates can be drawn for these equations.