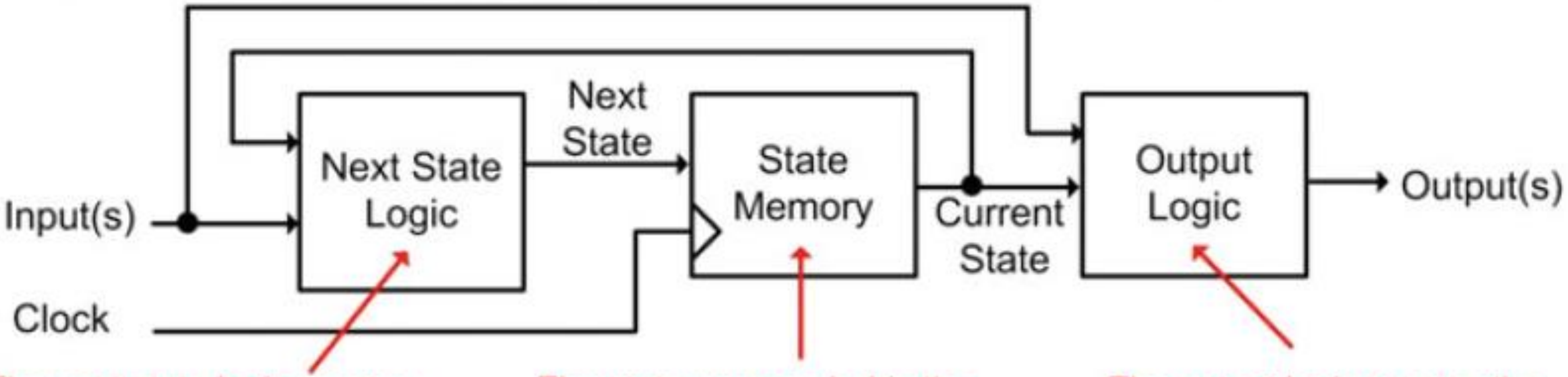
Slide acknowledgments: Textbook by Brock J. LaMeres

https://www.youtube.com/watch?v=_I8CLQazom0&list=PLTd6ceoshprfg23JMtwGysCm4tlc0I1ou&index=1

https://www.youtube.com/watch?v=TGcjn8zMhfM&list=PLTd6ceoshprfg23JMtwGysCm4tlc0I1ou&index=5

Main Components of a Finite State Machine

Mealy Machine – The output(s) depend on both the current state and system input(s).



Input(s)

Clock

Next State Logic

Next State

State Memory

Current State

Output Logic

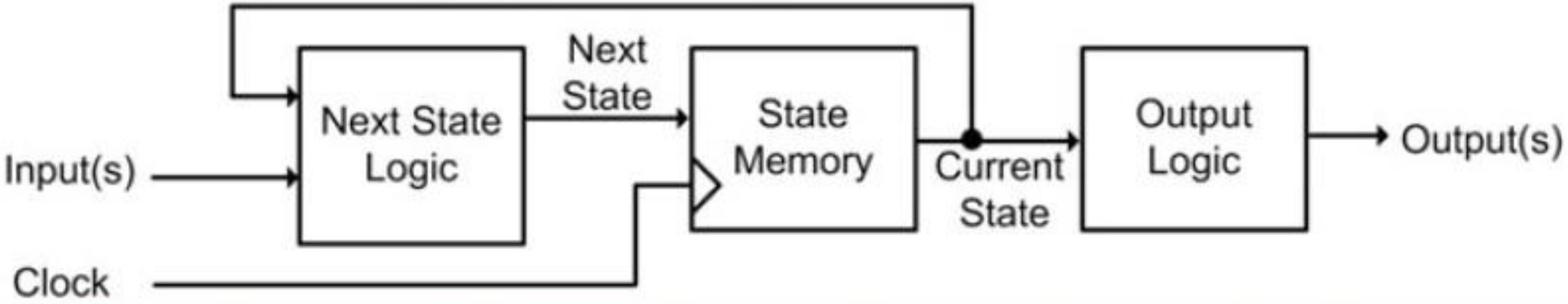
Output(s)

The next state logic creates the signal "next state" based on the current state and any system inputs. This block is implemented with combinational logic.

The state memory holds the current state. The current state is updated with "next state" on the rising edge of the clock. This block is implemented with D-Flip-Flops.

The output logic creates the system outputs. The output logic always depends on the current state of the machine and optionally (Meally vs. Moore) the inputs of the system.

Moore Machine – The output(s) depends only on the current state.



Input(s)

Clock

Next State Logic

Next State

State Memory

Current State

Output Logic

Output(s)

Creating an n-bit Gray Code Pattern

A gray code sequence begins with the known 2-bit pattern of 00, 01, 11, 10.

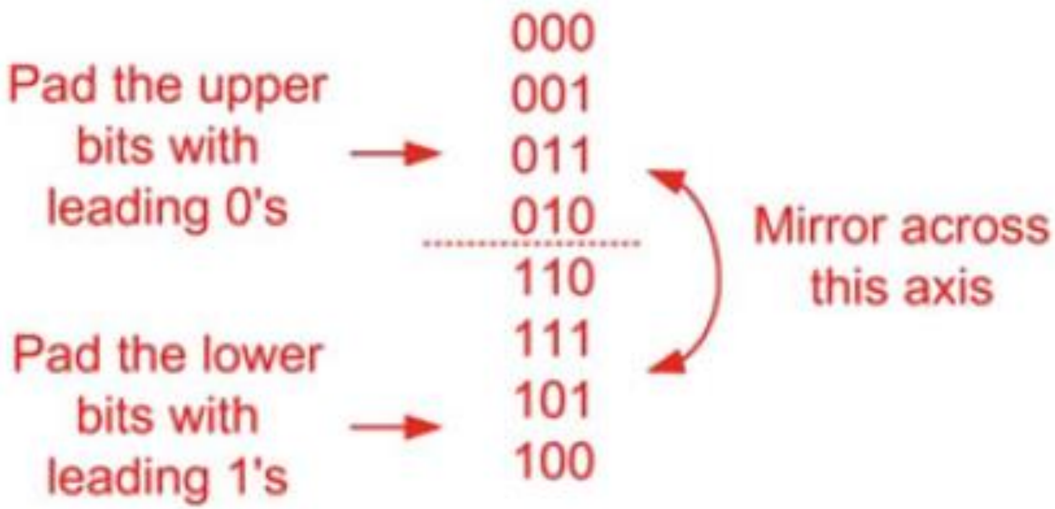
In order to increase the number of bits, the existing pattern is mirrored across an imaginary horizontal axis below the existing pattern. The bits above the axis are padded with leading 0's, and the bits below the axis are padded with leading 1's. This turns a 2-bit gray code pattern into a 3-bit pattern preserving the characteristic that each code only differs by its neighbor by one bit.

This process is repeated to create a 4-bit gray code pattern.

2-bit Gray Code Pattern

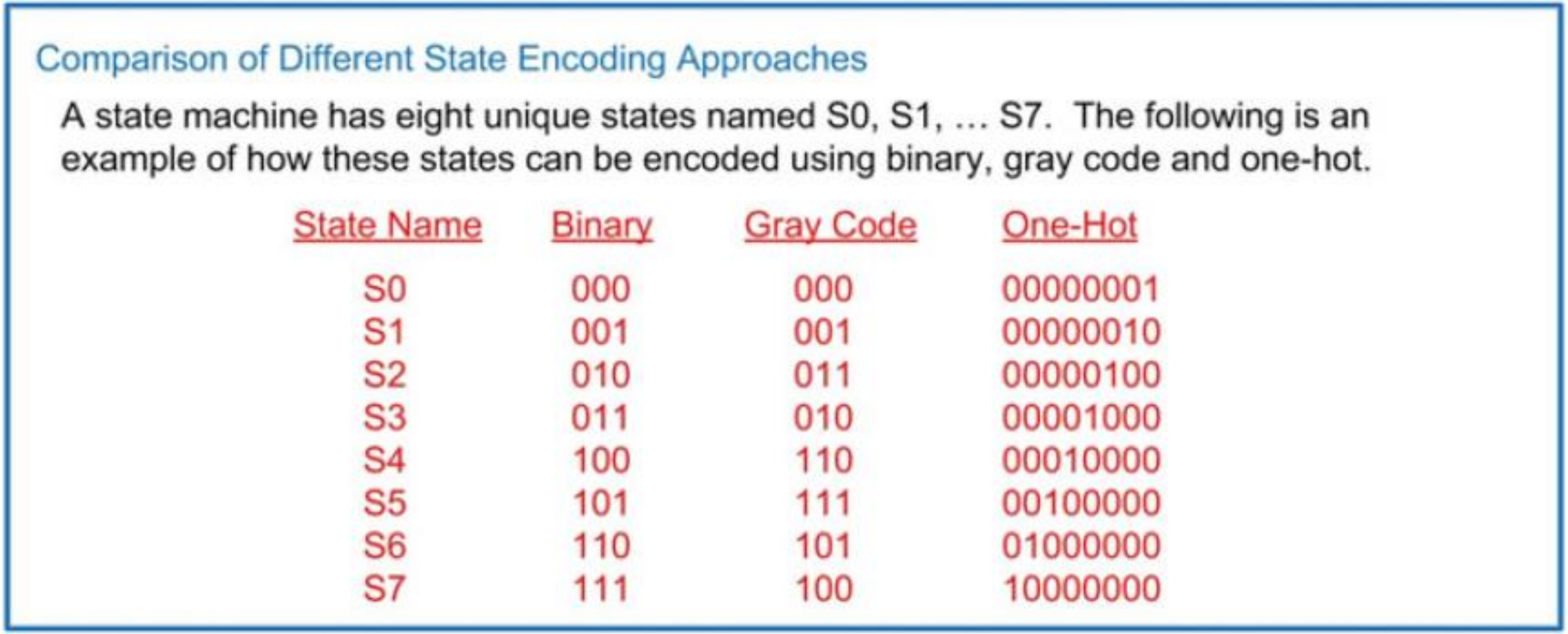
00
01
11
10

3-bit Gray Code Pattern



000
001
011
010
110
111
101
100

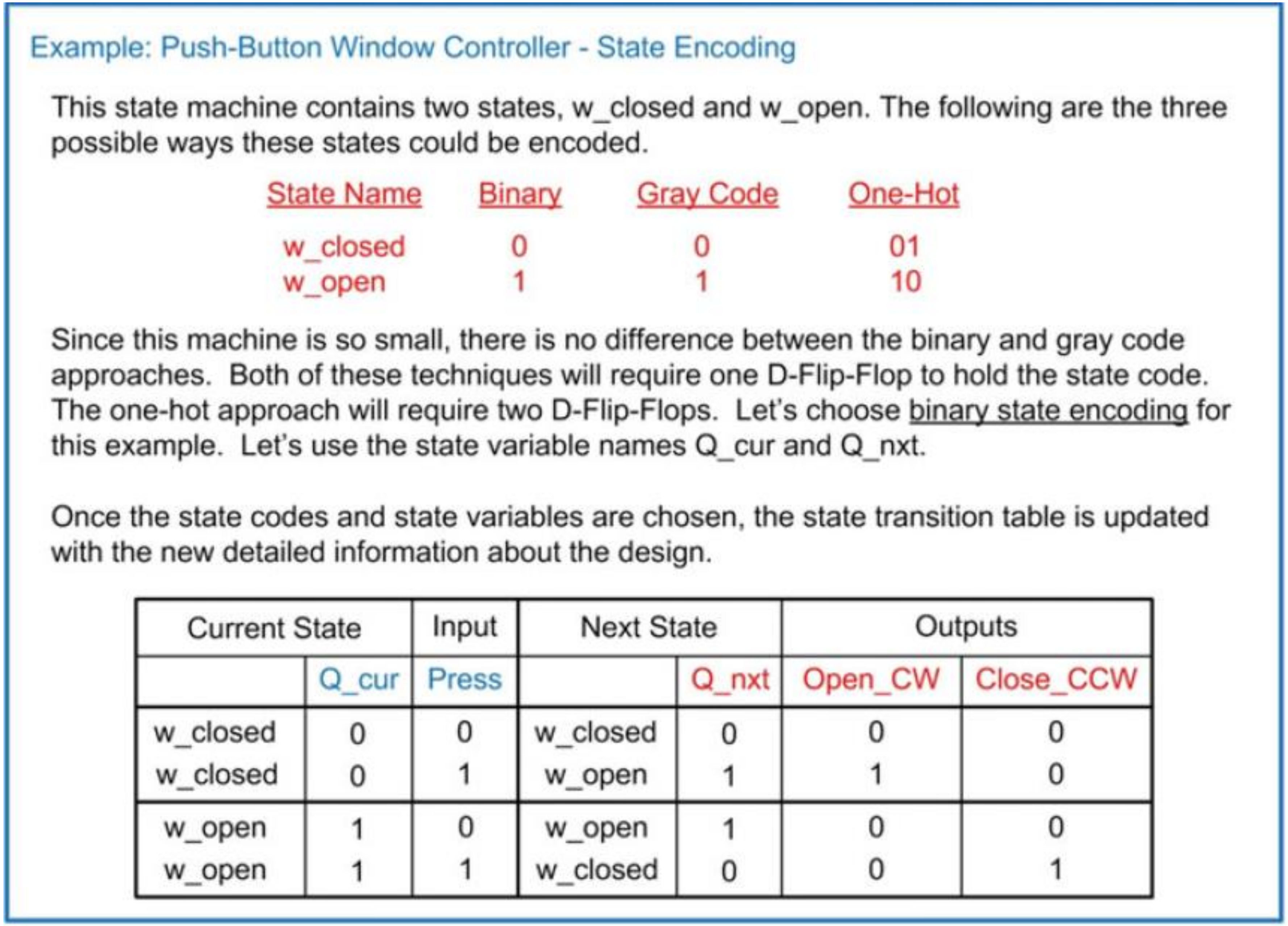
Pad the upper bits with leading 0's
Mirror across this axis
Pad the lower bits with leading 1's



Comparison of Different State Encoding Approaches

A state machine has eight unique states named S0, S1, … S7. The following is an example of how these states can be encoded using binary, gray code and one-hot.

State Name	Binary	Gray Code	One-Hot
S0	000	000	00000001
S1	001	001	00000010
S2	010	011	00000100
S3	011	010	00001000
S4	100	110	00010000
S5	101	111	00100000
S6	110	101	01000000
S7	111	100	10000000



Example: Push-Button Window Controller - State Encoding

This state machine contains two states, w_closed and w_open. The following are the three possible ways these states could be encoded.

State Name	Binary	Gray Code	One-Hot
w_closed	0	0	01
w_open	1	1	10

Since this machine is so small, there is no difference between the binary and gray code approaches. Both of these techniques will require one D-Flip-Flop to hold the state code. The one-hot approach will require two D-Flip-Flops. Let's choose binary state encoding for this example. Let's use the state variable names Q_cur and Q_nxt.

Once the state codes and state variables are chosen, the state transition table is updated with the new detailed information about the design.

Current State		Input	Next State		Outputs	
	Q_cur	Press		Q_nxt	Open_CW	Close_CCW
w_closed	0	0	w_closed	0	0	0
w_closed	0	1	w_open	1	1	0
w_open	1	0	w_open	1	0	0
w_open	1	1	w_closed	0	0	1

Example: Push-Button Window Controller - Next State Logic

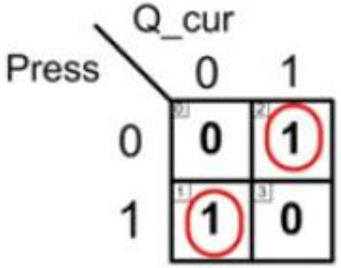
We need to synthesize the combinational logic circuit that will create the next state logic for Q_nxt. The behavior of this combinational logic circuit is described in the state transition table. In order to visualize where this information is within the table, let's pull it out and put it into a traditional truth table format.

Current State		Input	Next State		Outputs	
	Q_cur	Press		Q_nxt	Open_CW	Close_CCW
w_closed	0	0	w_closed	0	0	0
w_closed	0	1	w_open	1	1	0
w_open	1	0	w_open	1	0	0
w_open	1	1	w_closed	0	0	1

These columns are the inputs to the next state logic.

This column is the desired output for the next state logic variable Q_nxt.

Q_cur	Press	Q_nxt
0	0	0
0	1	1
1	0	1
1	1	0



Press \ Q_cur	0	1
0	**0**	**1**
1	**1**	**0**

Q_nxt = (Q_cur' · Press) + (Q_cur · Press')

or

Q_nxt = Q_cur ⊕ Press

Example: Push-Button Window Controller - Output Logic

We need to synthesize the combinational logic circuits that will create the output logic for the signals "Open_CW" and "Close_CCW". The behavior of this combinational logic circuit is described in the state transition table. Again, in order to visualize where this information is within the table, let's pull it out and put it into traditional truth table formats.

Current State		Input	Next State		Outputs	
	Q_cur	Press		Q_nxt	Open_CW	Close_CCW
w_closed	0	0	w_closed	0	0	0
w_closed	0	1	w_open	1	1	0
w_open	1	0	w_open	1	0	0
w_open	1	1	w_closed	0	0	1

↑ ↑ These columns are the inputs to the output logic.

↑ ↑ These columns are the desired behavior of the outputs.

Q_cur	Press	Open_CW
0	0	0
0	1	1
1	0	0
1	1	0

→

Press \ Q_cur	0	1
0	0	0
1	1	0

→ Open_CW = Q_cur' · Press

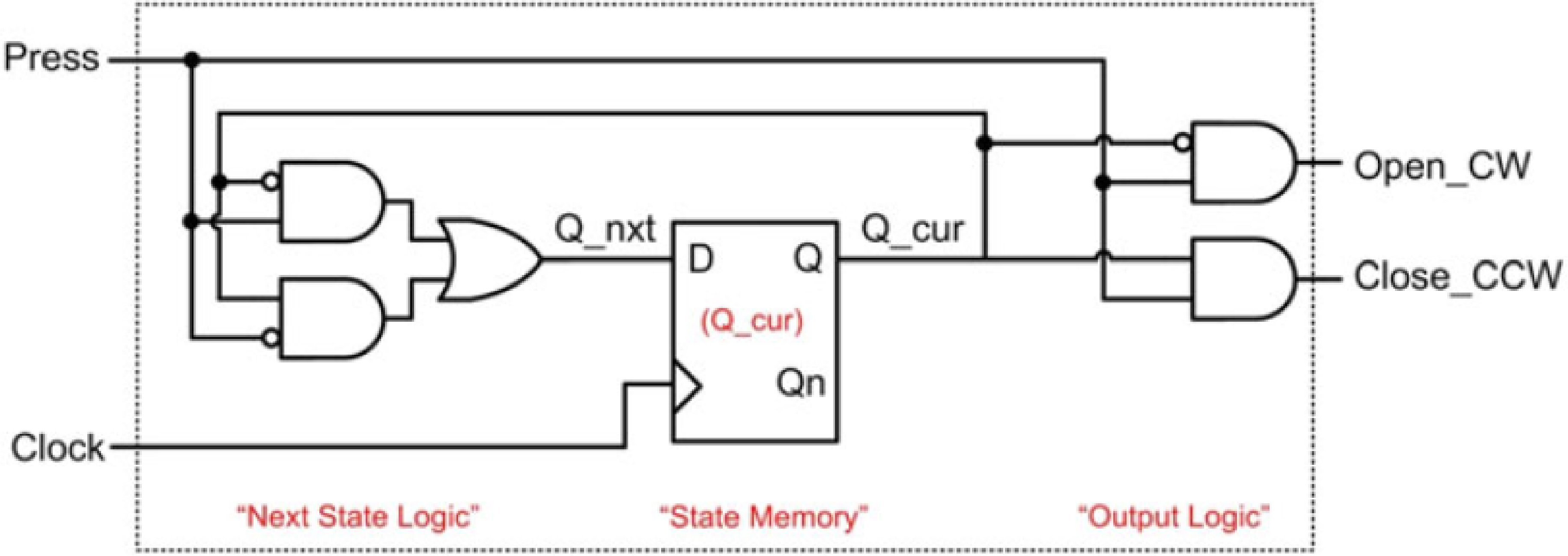
Q_cur	Press	Close_CCW
0	0	0
0	1	0
1	0	0
1	1	1

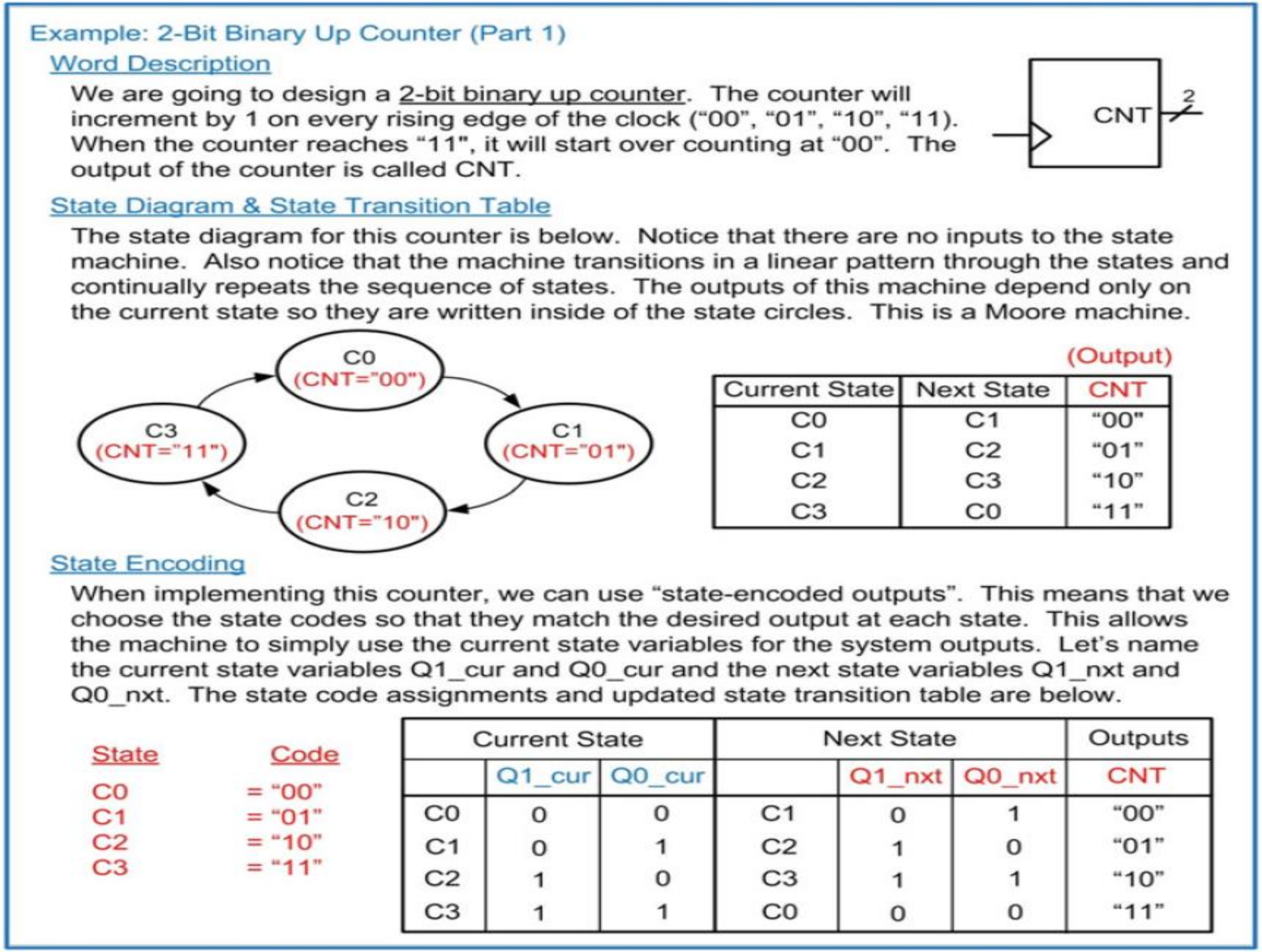
→

Press \ Q_cur	0	1
0	0	0
1	0	1

→ Close_CCW = Q_cur · Press

Example: Push-Button Window Controller - Logic Diagram





Example: 2-Bit Binary Up Counter (Part 1)

Word Description

We are going to design a 2-bit binary up counter. The counter will increment by 1 on every rising edge of the clock ("00", "01", "10", "11). When the counter reaches "11", it will start over counting at "00". The output of the counter is called CNT.

State Diagram & State Transition Table

The state diagram for this counter is below. Notice that there are no inputs to the state machine. Also notice that the machine transitions in a linear pattern through the states and continually repeats the sequence of states. The outputs of this machine depend only on the current state so they are written inside of the state circles. This is a Moore machine.

(Output)

Current State	Next State	CNT
C0	C1	"00"
C1	C2	"01"
C2	C3	"10"
C3	C0	"11"

State Encoding

When implementing this counter, we can use "state-encoded outputs". This means that we choose the state codes so that they match the desired output at each state. This allows the machine to simply use the current state variables for the system outputs. Let's name the current state variables Q1_cur and Q0_cur and the next state variables Q1_nxt and Q0_nxt. The state code assignments and updated state transition table are below.

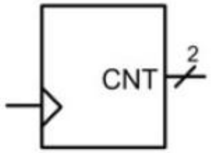
State	Code
C0	= "00"
C1	= "01"
C2	= "10"
C3	= "11"

Current State			Next State			Outputs
	Q1_cur	Q0_cur		Q1_nxt	Q0_nxt	CNT
C0	0	0	C1	0	1	"00"
C1	0	1	C2	1	0	"01"
C2	1	0	C3	1	1	"10"
C3	1	1	C0	0	0	"11"

Example: 2-Bit Binary Up Counter (Part 1)

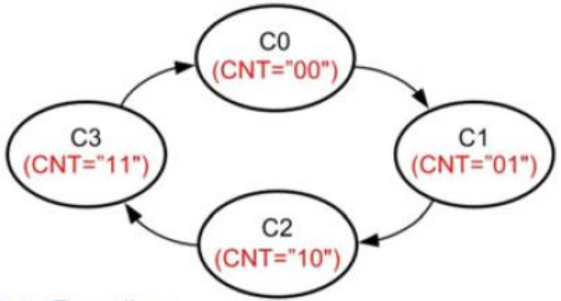
Word Description

We are going to design a 2-bit binary up counter. The counter will increment by 1 on every rising edge of the clock ("00", "01", "10", "11). When the counter reaches "11", it will start over counting at "00". The output of the counter is called CNT.



State Diagram & State Transition Table

The state diagram for this counter is below. Notice that there are no inputs to the state machine. Also notice that the machine transitions in a linear pattern through the states and continually repeats the sequence of states. The outputs of this machine depend only on the current state so they are written inside of the state circles. This is a Moore machine.



(Output)

Current State	Next State	CNT
C0	C1	"00"
C1	C2	"01"
C2	C3	"10"
C3	C0	"11"

State Encoding

When implementing this counter, we can use "state-encoded outputs". This means that we choose the state codes so that they match the desired output at each state. This allows the machine to simply use the current state variables for the system outputs. Let's name the current state variables Q1_cur and Q0_cur and the next state variables Q1_nxt and Q0_nxt. The state code assignments and updated state transition table are below.

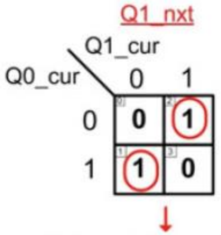
State	Code
C0	= "00"
C1	= "01"
C2	= "10"
C3	= "11"

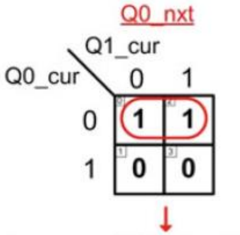
Current State			Next State			Outputs
	Q1_cur	Q0_cur		Q1_nxt	Q0_nxt	CNT
C0	0	0	C1	0	1	"00"
C1	0	1	C2	1	0	"01"
C2	1	0	C3	1	1	"10"
C3	1	1	C0	0	0	"11"

Example: 2-Bit Binary Up Counter (Part 2)

Next State Logic

The next state logic for this counter only depends on the current state variables since there are no inputs to the system.





Q1_nxt = (Q1_cur' · Q0_cur) + (Q1_cur · Q0_cur')

or

Q1_nxt = Q1_cur ⊕ Q0_cur

Q0_nxt = Q0_cur'

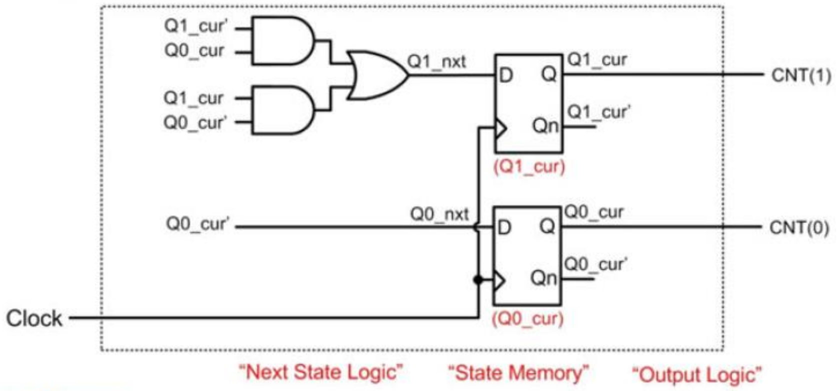
Output Logic

Since we are using state-encoded outputs, the outputs of the system will simply be the current state variables.

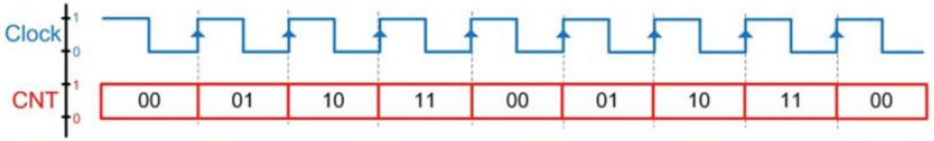
CNT(1) = Q1_cur

CNT(0) = Q0_cur

Logic Diagram



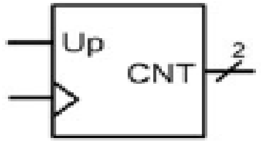
Timing Diagram



Example: 2-Bit Binary Up/Down Counter (Part 1)

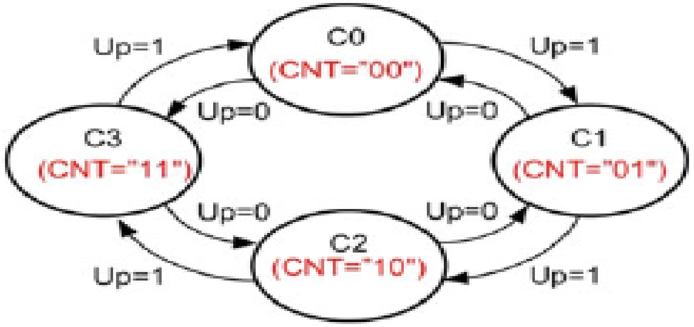
Word Description

We are going to design a 2-bit binary up/down counter. When the system input "Up" is asserted, the counter will increment by 1 on every rising edge of the clock. When Up=0, the counter will decrement by 1 on every rising edge of the clock. The output of the counter is called CNT.



State Diagram & State Transition Table

The state diagram for this counter is below. In this diagram, if the input Up=**1**, the machine will traverse the states in order to create an incrementing count. If the input Up=0, the machine will traverse the states in the opposite order. The outputs of this machine again only depend on the current state so they are written inside of the state circles. This is a Moore machine.



	(Input)		(Output)
Current State	Up	Next State	CNT
C0	0	C3	"00"
	1	C1	
C1	0	C0	"01"
	1	C2	
C2	0	C1	"10"
	1	C3	
C3	0	C2	"11"
	1	C0	

State Encoding

Again, this counter will use "state-encoded outputs". Let's name the current state variables Q1_cur and Q0_cur and the next state variables Q1_nxt and Q0_nxt. The state code assignments and updated state transition table are below.

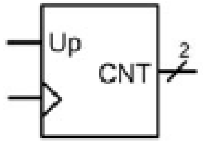
State	Code
C0	= "00"
C1	= "01"
C2	= "10"
C3	= "11"

Current State			Input	Next State			Outputs
	Q1_cur	Q0_cur	Up		Q1_nxt	Q0_nxt	CNT
C0	0	0	0	C3	1	1	"00"
C0	0	0	1	C1	0	1	"00"
C1	0	1	0	C0	0	0	"01"
C1	0	1	1	C2	1	0	"01"
C2	1	0	0	C1	0	1	"10"
C2	1	0	1	C3	1	1	"10"
C3	1	1	0	C2	1	0	"11"
C3	1	1	1	C0	0	0	"11"

Example: 2-Bit Binary Up/Down Counter (Part 1)

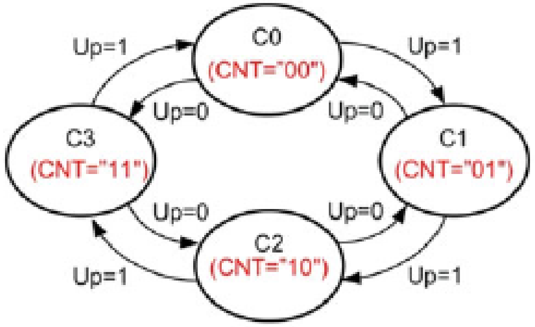
Word Description

We are going to design a 2-bit binary up/down counter. When the system input "Up" is asserted, the counter will increment by 1 on every rising edge of the clock. When Up=0, the counter will decrement by 1 on every rising edge of the clock. The output of the counter is called CNT.



State Diagram & State Transition Table

The state diagram for this counter is below. In this diagram, if the input Up=1, the machine will traverse the states in order to create an incrementing count. If the input Up=0, the machine will traverse the states in the opposite order. The outputs of this machine again only depend on the current state so they are written inside of the state circles. This is a Moore machine.



(Input) (Output)

Current State	Up	Next State	CNT
C0	0	C3	"00"
	1	C1	
C1	0	C0	"01"
	1	C2	
C2	0	C1	"10"
	1	C3	
C3	0	C2	"11"
	1	C0	

State Encoding

Again, this counter will use "state-encoded outputs". Let's name the current state variables Q1_cur and Q0_cur and the next state variables Q1_nxt and Q0_nxt. The state code assignments and updated state transition table are below.

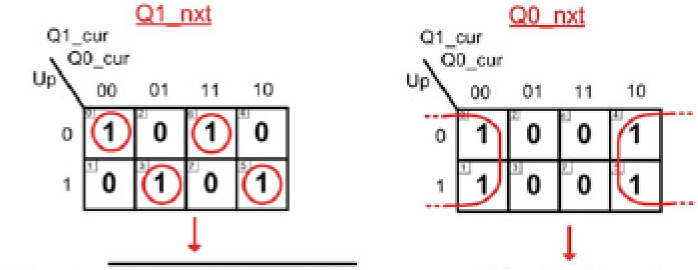
State	Code
C0	= "00"
C1	= "01"
C2	= "10"
C3	= "11"

Current State			Input	Next State			Outputs
	Q1_cur	Q0_cur	Up		Q1_nxt	Q0_nxt	CNT
C0	0	0	0	C3	1	1	"00"
C0	0	0	1	C1	0	1	"00"
C1	0	1	0	C0	0	0	"01"
C1	0	1	1	C2	1	0	"01"
C2	1	0	0	C1	0	1	"10"
C2	1	0	1	C3	1	1	"10"
C3	1	1	0	C2	1	0	"11"
C3	1	1	1	C0	0	0	"11"

Example: 2-Bit Binary Up/Down Counter (Part 2)

Next State Logic

The next state logic for this counter depends on both the current state variables and the input Up.



Q1_nxt

Up \ Q1_cur Q0_cur	00	01	11	10
0	1	0	1	0
1	0	1	0	1

Q0_nxt

Up \ Q1_cur Q0_cur	00	01	11	10
0	1	0	0	1
1	1	0	0	1

$$Q1\_nxt = \overline{Q1\_cur \oplus Q0\_cur \oplus Up}$$

$$Q0\_nxt = Q0\_cur'$$

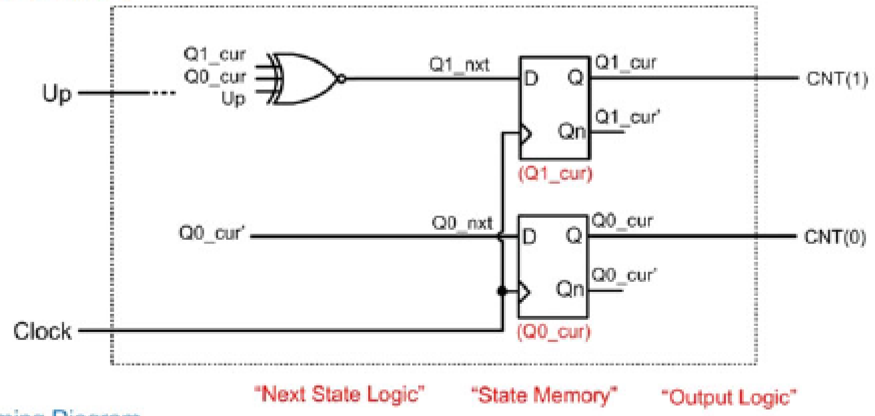
Output Logic

Since we are using state-encoded outputs, the outputs of the system will simply be the current state variables.

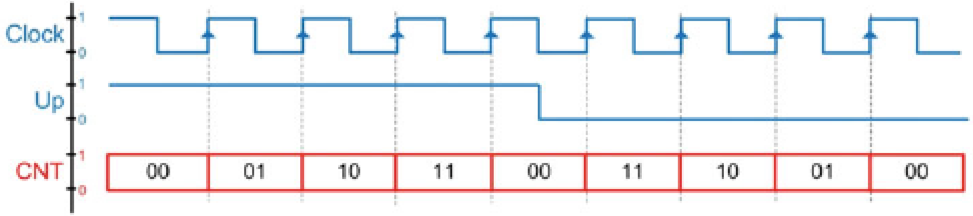
CNT(1) = Q1_cur

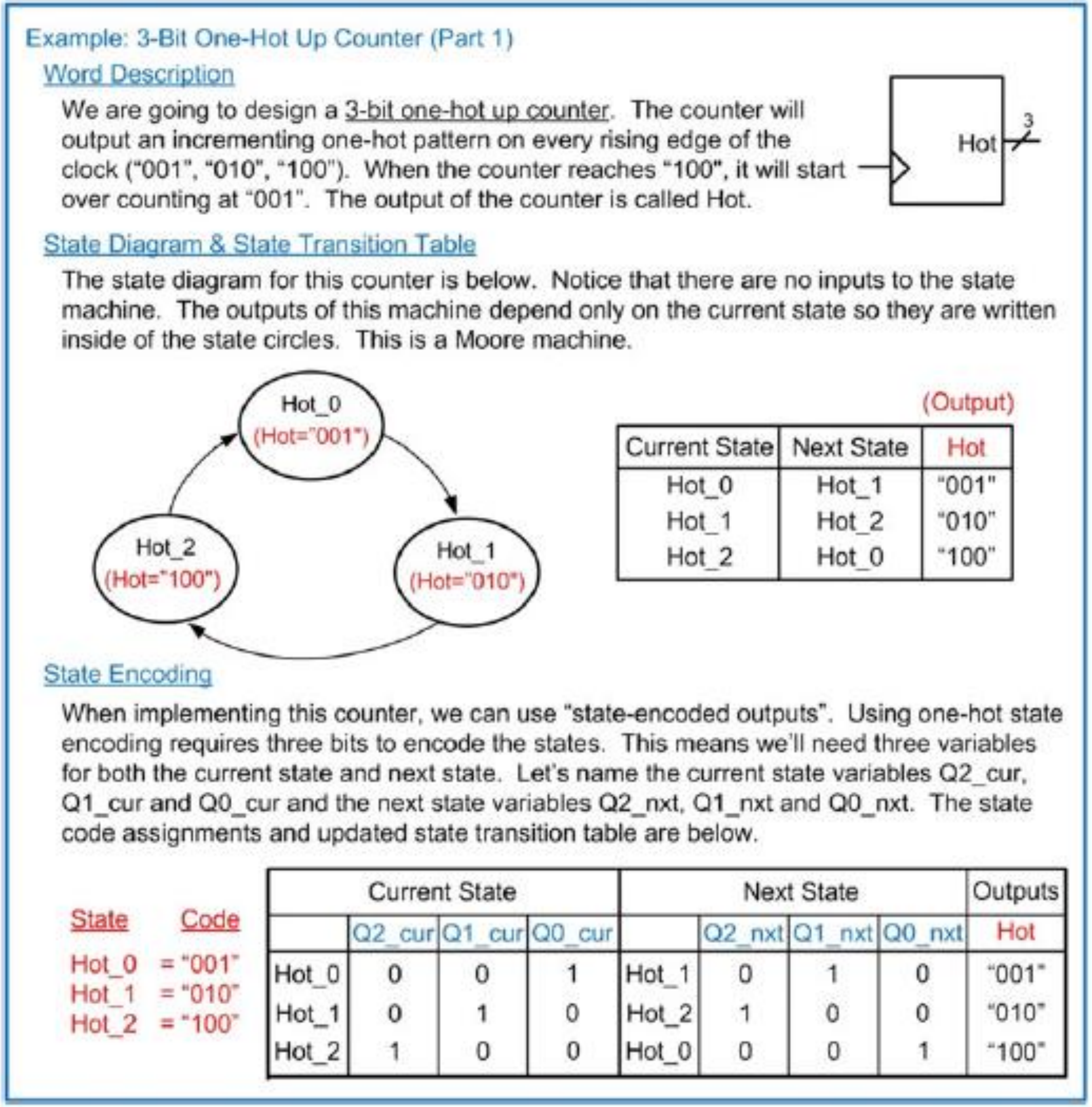
CNT(0) = Q0_cur

Logic Diagram



Timing Diagram





Example: 3-Bit One-Hot Up Counter (Part 1)

Word Description

We are going to design a 3-bit one-hot up counter. The counter will output an incrementing one-hot pattern on every rising edge of the clock ("001", "010", "100"). When the counter reaches "100", it will start over counting at "001". The output of the counter is called Hot.

State Diagram & State Transition Table

The state diagram for this counter is below. Notice that there are no inputs to the state machine. The outputs of this machine depend only on the current state so they are written inside of the state circles. This is a Moore machine.

(Output)

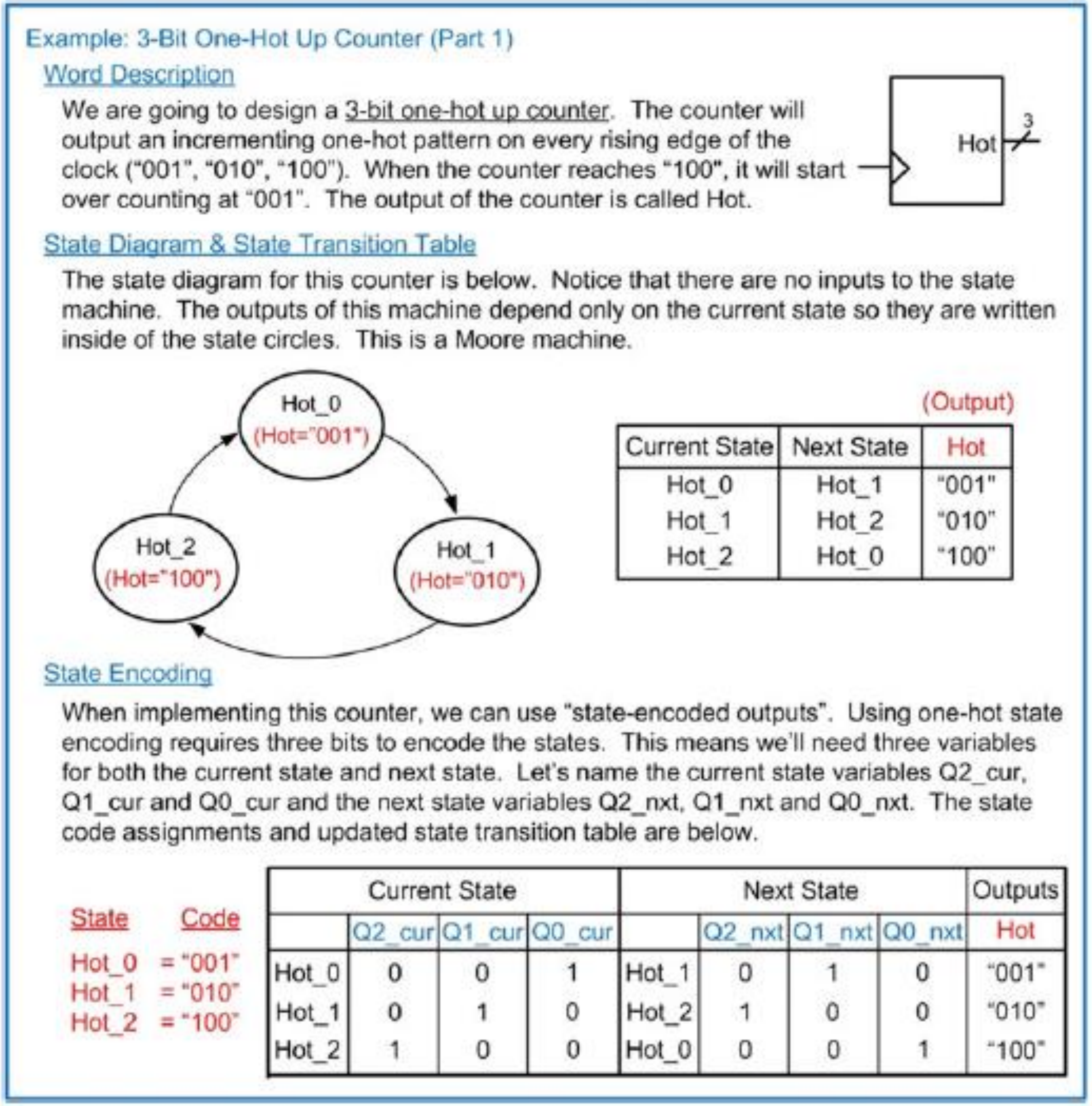
Current State	Next State	Hot
Hot_0	Hot_1	"001"
Hot_1	Hot_2	"010"
Hot_2	Hot_0	"100"

State Encoding

When implementing this counter, we can use "state-encoded outputs". Using one-hot state encoding requires three bits to encode the states. This means we'll need three variables for both the current state and next state. Let's name the current state variables Q2_cur, Q1_cur and Q0_cur and the next state variables Q2_nxt, Q1_nxt and Q0_nxt. The state code assignments and updated state transition table are below.

State	Code
Hot_0	= "001"
Hot_1	= "010"
Hot_2	= "100"

Current State				Next State				Outputs
	Q2_cur	Q1_cur	Q0_cur		Q2_nxt	Q1_nxt	Q0_nxt	Hot
Hot_0	0	0	1	Hot_1	0	1	0	"001"
Hot_1	0	1	0	Hot_2	1	0	0	"010"
Hot_2	1	0	0	Hot_0	0	0	1	"100"



Example: 3-Bit One-Hot Up Counter (Part 1)

Word Description

We are going to design a 3-bit one-hot up counter. The counter will output an incrementing one-hot pattern on every rising edge of the clock ("001", "010", "100"). When the counter reaches "100", it will start over counting at "001". The output of the counter is called Hot.

State Diagram & State Transition Table

The state diagram for this counter is below. Notice that there are no inputs to the state machine. The outputs of this machine depend only on the current state so they are written inside of the state circles. This is a Moore machine.

(Output)

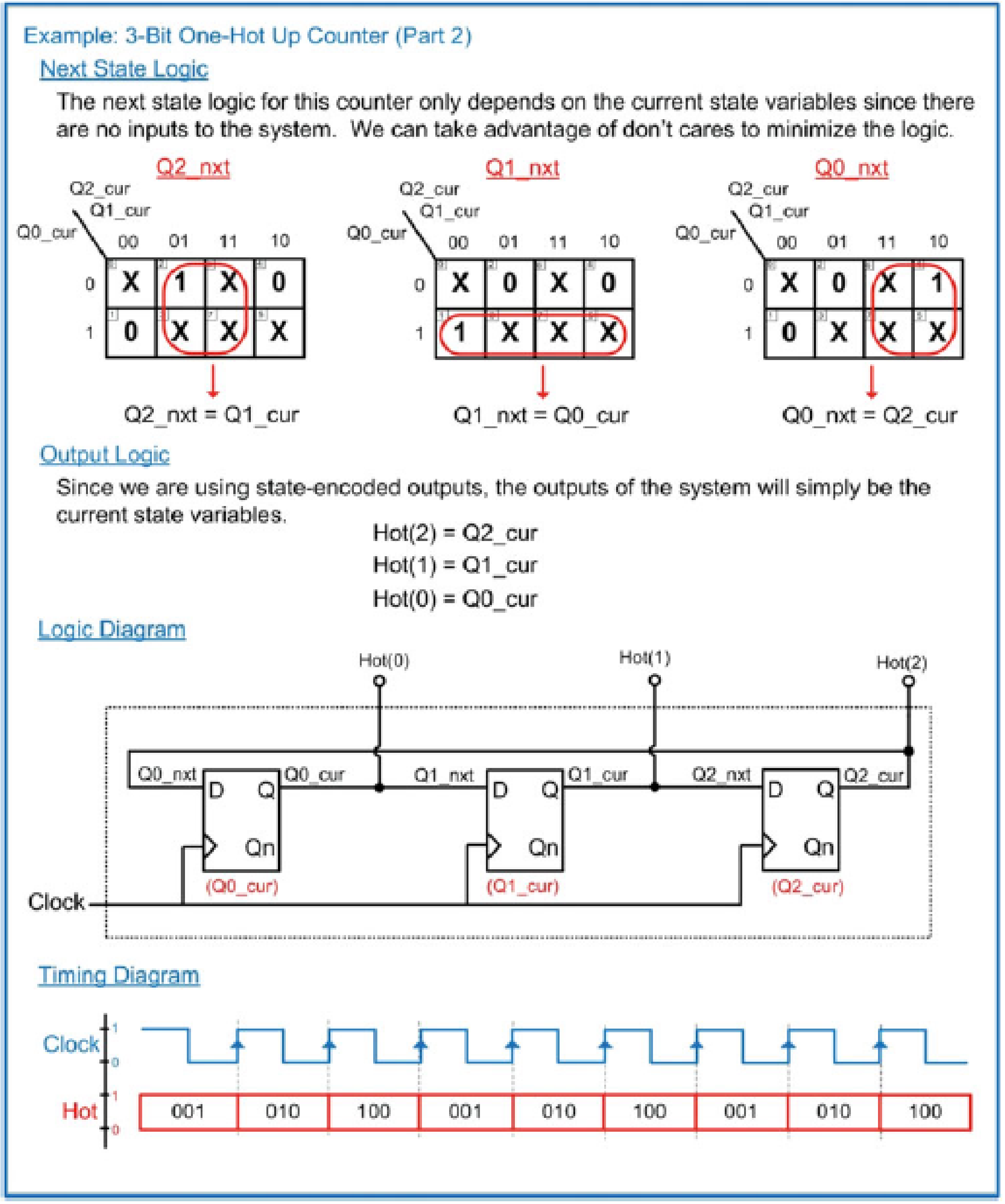
Current State	Next State	Hot
Hot_0	Hot_1	"001"
Hot_1	Hot_2	"010"
Hot_2	Hot_0	"100"

State Encoding

When implementing this counter, we can use "state-encoded outputs". Using one-hot state encoding requires three bits to encode the states. This means we'll need three variables for both the current state and next state. Let's name the current state variables Q2_cur, Q1_cur and Q0_cur and the next state variables Q2_nxt, Q1_nxt and Q0_nxt. The state code assignments and updated state transition table are below.

State	Code
Hot_0	= "001"
Hot_1	= "010"
Hot_2	= "100"

Current State				Next State				Outputs
	Q2_cur	Q1_cur	Q0_cur		Q2_nxt	Q1_nxt	Q0_nxt	Hot
Hot_0	0	0	1	Hot_1	0	1	0	"001"
Hot_1	0	1	0	Hot_2	1	0	0	"010"
Hot_2	1	0	0	Hot_0	0	0	1	"100"



Example: 3-Bit One-Hot Up Counter (Part 2)

Next State Logic

The next state logic for this counter only depends on the current state variables since there are no inputs to the system. We can take advantage of don't cares to minimize the logic.

Q2_nxt

Q0_cur \ Q2_cur Q1_cur	00	01	11	10
0	X	1	X	0
1	0	X	X	X

Q2_nxt = Q1_cur

Q1_nxt

Q0_cur \ Q2_cur Q1_cur	00	01	11	10
0	X	0	X	0
1	1	X	X	X

Q1_nxt = Q0_cur

Q0_nxt

Q0_cur \ Q2_cur Q1_cur	00	01	11	10
0	X	0	X	1
1	0	X	X	X

Q0_nxt = Q2_cur

Output Logic

Since we are using state-encoded outputs, the outputs of the system will simply be the current state variables.

Hot(2) = Q2_cur

Hot(1) = Q1_cur

Hot(0) = Q0_cur

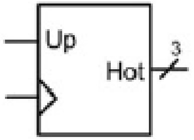
Logic Diagram

Timing Diagram

Example: 3-Bit One-Hot Up/Down Counter (Part 1)

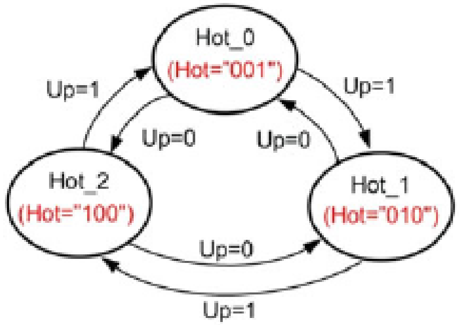
Word Description

We are going to design a 3-bit one-hot up/down counter. When the system input "Up" is asserted, the counter will output an incrementing one-hot pattern on every rising edge of the clock ("001", "010", "100"). When the input Up=0, the counter will output a decrementing one-hot pattern ("100", "010", "001"). The output of the counter is called Hot.



State Diagram & State Transition Table

The state diagram and state transition table for this counter are below.



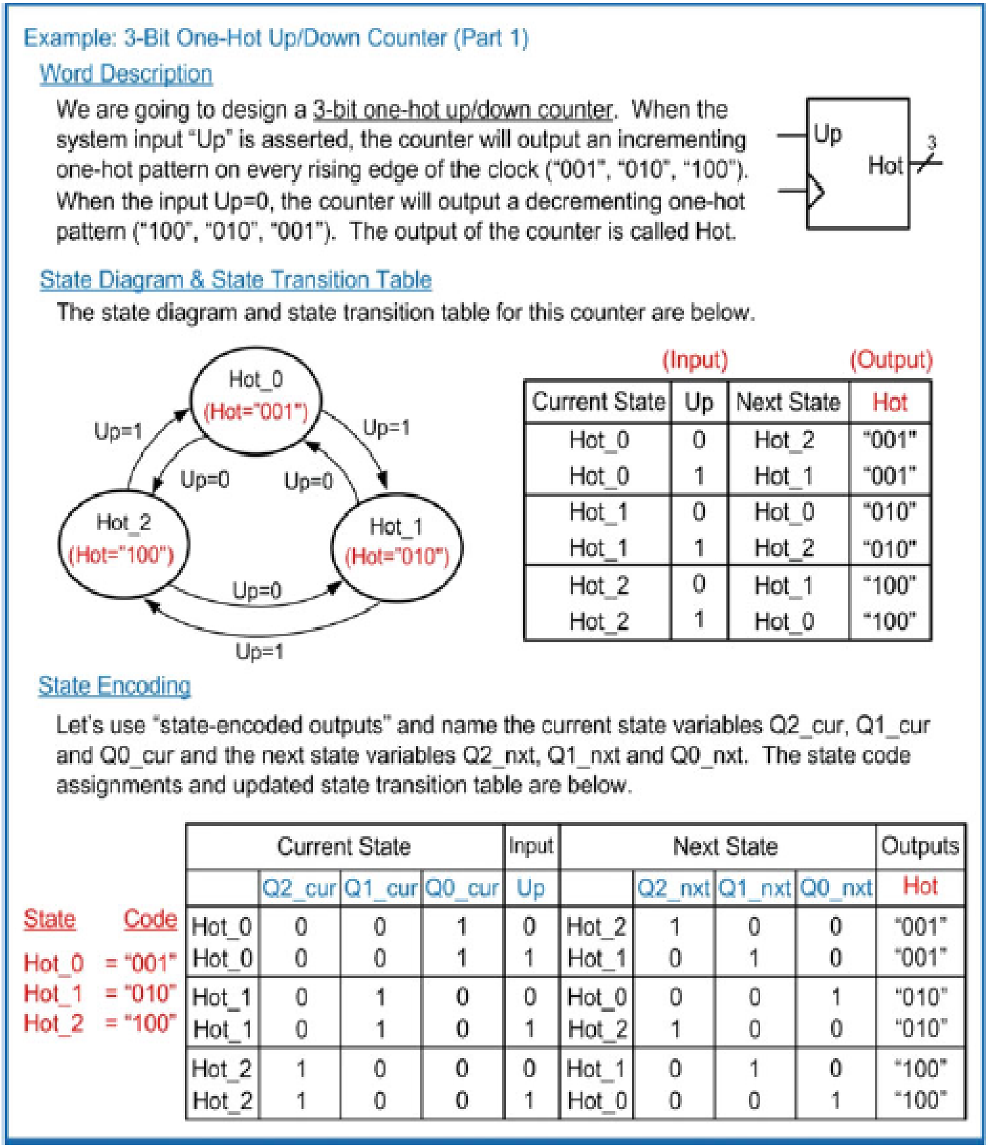
Current State	Up (Input)	Next State	Hot (Output)
Hot_0	0	Hot_2	"001"
Hot_0	1	Hot_1	"001"
Hot_1	0	Hot_0	"010"
Hot_1	1	Hot_2	"010"
Hot_2	0	Hot_1	"100"
Hot_2	1	Hot_0	"100"

State Encoding

Let's use "state-encoded outputs" and name the current state variables Q2_cur, Q1_cur and Q0_cur and the next state variables Q2_nxt, Q1_nxt and Q0_nxt. The state code assignments and updated state transition table are below.

State	Code
Hot_0	= "001"
Hot_1	= "010"
Hot_2	= "100"

Current State				Input	Next State				Outputs
	Q2_cur	Q1_cur	Q0_cur	Up		Q2_nxt	Q1_nxt	Q0_nxt	Hot
Hot_0	0	0	1	0	Hot_2	1	0	0	"001"
Hot_0	0	0	1	1	Hot_1	0	1	0	"001"
Hot_1	0	1	0	0	Hot_0	0	0	1	"010"
Hot_1	0	1	0	1	Hot_2	1	0	0	"010"
Hot_2	1	0	0	0	Hot_1	0	1	0	"100"
Hot_2	1	0	0	1	Hot_0	0	0	1	"100"



Example: 3-Bit One-Hot Up/Down Counter (Part 1)

Word Description

We are going to design a 3-bit one-hot up/down counter. When the system input "Up" is asserted, the counter will output an incrementing one-hot pattern on every rising edge of the clock ("001", "010", "100"). When the input Up=0, the counter will output a decrementing one-hot pattern ("100", "010", "001"). The output of the counter is called Hot.

State Diagram & State Transition Table

The state diagram and state transition table for this counter are below.

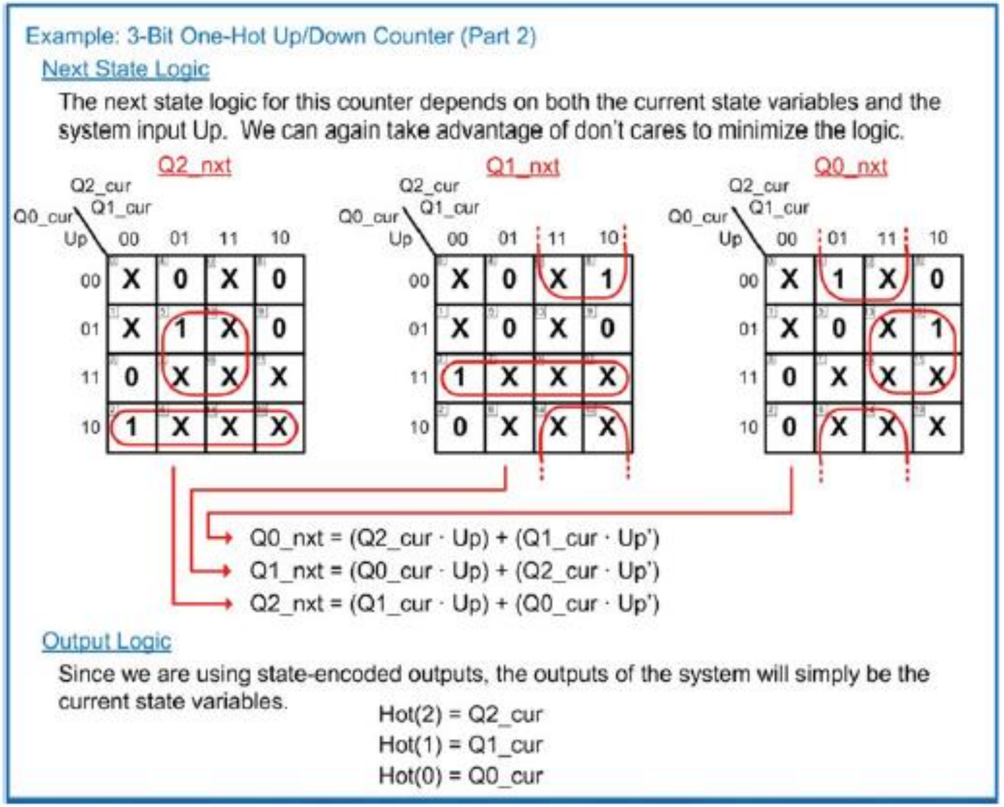
Current State	(Input) Up	Next State	(Output) Hot
Hot_0	0	Hot_2	"001"
Hot_0	1	Hot_1	"001"
Hot_1	0	Hot_0	"010"
Hot_1	1	Hot_2	"010"
Hot_2	0	Hot_1	"100"
Hot_2	1	Hot_0	"100"

State Encoding

Let's use "state-encoded outputs" and name the current state variables Q2_cur, Q1_cur and Q0_cur and the next state variables Q2_nxt, Q1_nxt and Q0_nxt. The state code assignments and updated state transition table are below.

State	Code
Hot_0	= "001"
Hot_1	= "010"
Hot_2	= "100"

Current State				Input	Next State				Outputs
	Q2_cur	Q1_cur	Q0_cur	Up		Q2_nxt	Q1_nxt	Q0_nxt	Hot
Hot_0	0	0	1	0	Hot_2	1	0	0	"001"
Hot_0	0	0	1	1	Hot_1	0	1	0	"001"
Hot_1	0	1	0	0	Hot_0	0	0	1	"010"
Hot_1	0	1	0	1	Hot_2	1	0	0	"010"
Hot_2	1	0	0	0	Hot_1	0	1	0	"100"
Hot_2	1	0	0	1	Hot_0	0	0	1	"100"



Example: 3-Bit One-Hot Up/Down Counter (Part 2)

Next State Logic

The next state logic for this counter depends on both the current state variables and the system input Up. We can again take advantage of don't cares to minimize the logic.

Q2_nxt

Q0_cur Up \ Q2_cur Q1_cur	00	01	11	10
00	X	0	X	0
01	X	1	X	0
11	0	X	X	X
10	1	X	X	X

Q1_nxt

Q0_cur Up \ Q2_cur Q1_cur	00	01	11	10
00	X	0	X	1
01	X	0	X	0
11	1	X	X	X
10	0	X	X	X

Q0_nxt

Q0_cur Up \ Q2_cur Q1_cur	00	01	11	10
00	X	1	X	0
01	X	0	X	1
11	0	X	X	X
10	0	X	X	X

Q0_nxt = (Q2_cur · Up) + (Q1_cur · Up')

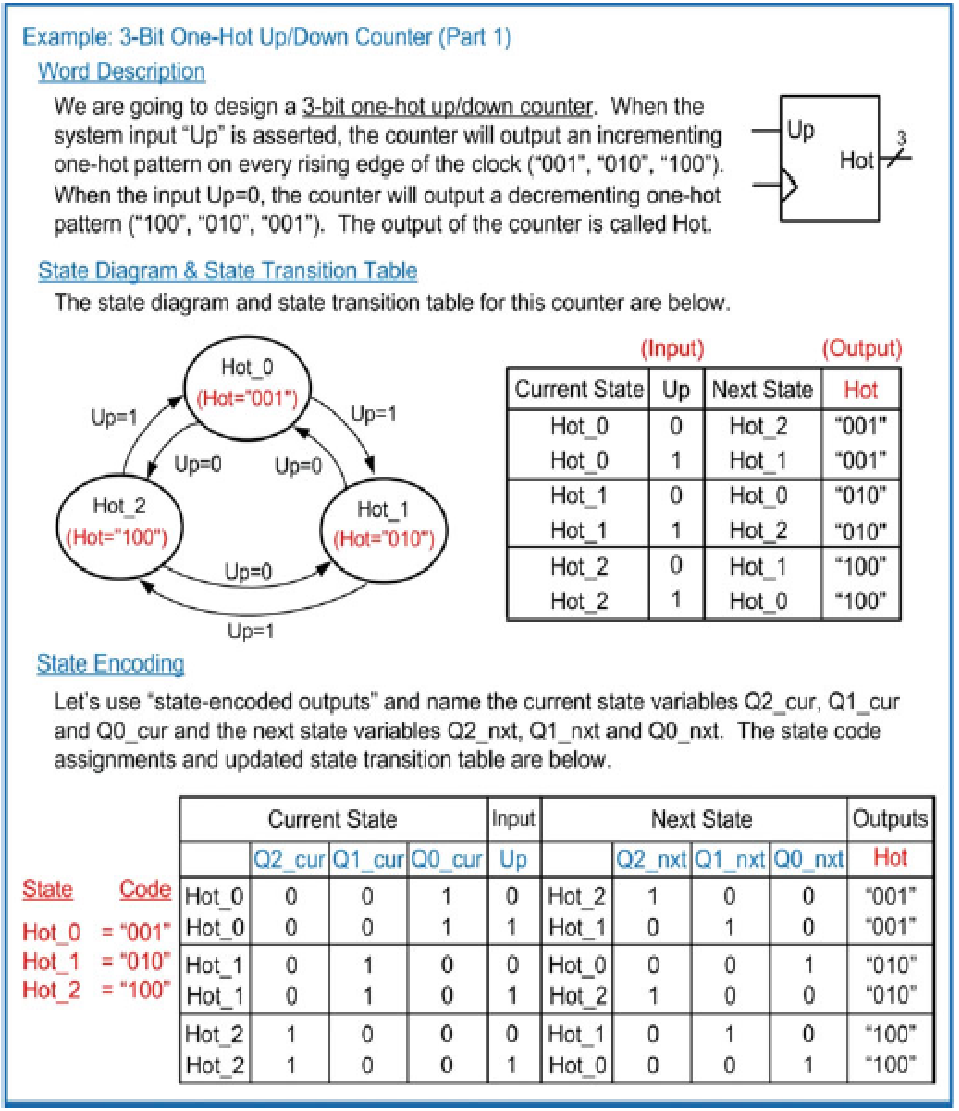
Q1_nxt = (Q0_cur · Up) + (Q2_cur · Up')

Q2_nxt = (Q1_cur · Up) + (Q0_cur · Up')

Output Logic

Since we are using state-encoded outputs, the outputs of the system will simply be the current state variables.

Hot(2) = Q2_cur
Hot(1) = Q1_cur
Hot(0) = Q0_cur



Example: 3-Bit One-Hot Up/Down Counter (Part 1)

Word Description

We are going to design a 3-bit one-hot up/down counter. When the system input "Up" is asserted, the counter will output an incrementing one-hot pattern on every rising edge of the clock ("001", "010", "100"). When the input Up=0, the counter will output a decrementing one-hot pattern ("100", "010", "001"). The output of the counter is called Hot.

State Diagram & State Transition Table

The state diagram and state transition table for this counter are below.

(Input) (Output)

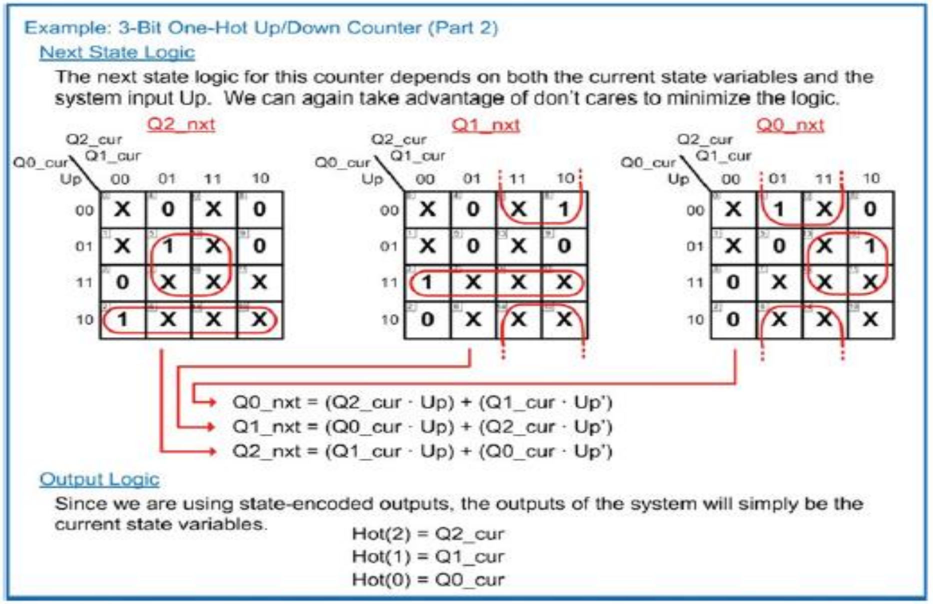
Current State	Up	Next State	Hot
Hot_0	0	Hot_2	"001"
Hot_0	1	Hot_1	"001"
Hot_1	0	Hot_0	"010"
Hot_1	1	Hot_2	"010"
Hot_2	0	Hot_1	"100"
Hot_2	1	Hot_0	"100"

State Encoding

Let's use "state-encoded outputs" and name the current state variables Q2_cur, Q1_cur and Q0_cur and the next state variables Q2_nxt, Q1_nxt and Q0_nxt. The state code assignments and updated state transition table are below.

State	Code
Hot_0	= "001"
Hot_1	= "010"
Hot_2	= "100"

Current State				Input	Next State				Outputs
	Q2_cur	Q1_cur	Q0_cur	Up		Q2_nxt	Q1_nxt	Q0_nxt	Hot
Hot_0	0	0	1	0	Hot_2	1	0	0	"001"
Hot_0	0	0	1	1	Hot_1	0	1	0	"001"
Hot_1	0	1	0	0	Hot_0	0	0	1	"010"
Hot_1	0	1	0	1	Hot_2	1	0	0	"010"
Hot_2	1	0	0	0	Hot_1	0	1	0	"100"
Hot_2	1	0	0	1	Hot_0	0	0	1	"100"



Example: 3-Bit One-Hot Up/Down Counter (Part 2)

Next State Logic

The next state logic for this counter depends on both the current state variables and the system input Up. We can again take advantage of don't cares to minimize the logic.

Q2_nxt

Q0_cur Up \ Q2_cur Q1_cur	00	01	11	10
00	X	0	X	0
01	X	1	X	0
11	0	X	X	X
10	1	X	X	X

Q1_nxt

Q0_cur Up \ Q2_cur Q1_cur	00	01	11	10
00	X	0	X	1
01	X	0	X	0
11	1	X	X	X
10	0	X	X	X

Q0_nxt

Q0_cur Up \ Q2_cur Q1_cur	00	01	11	10
00	X	1	X	0
01	X	0	X	1
11	0	X	X	X
10	0	X	X	X

Q0_nxt = (Q2_cur · Up) + (Q1_cur · Up')

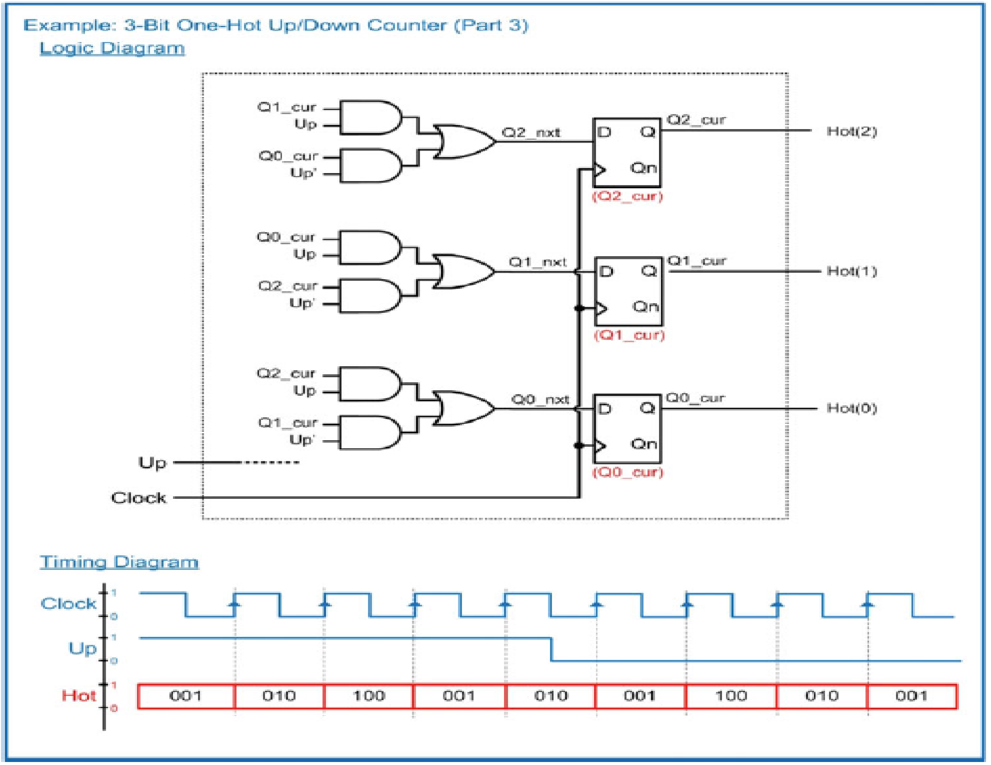
Q1_nxt = (Q0_cur · Up) + (Q2_cur · Up')

Q2_nxt = (Q1_cur · Up) + (Q0_cur · Up')

Output Logic

Since we are using state-encoded outputs, the outputs of the system will simply be the current state variables.

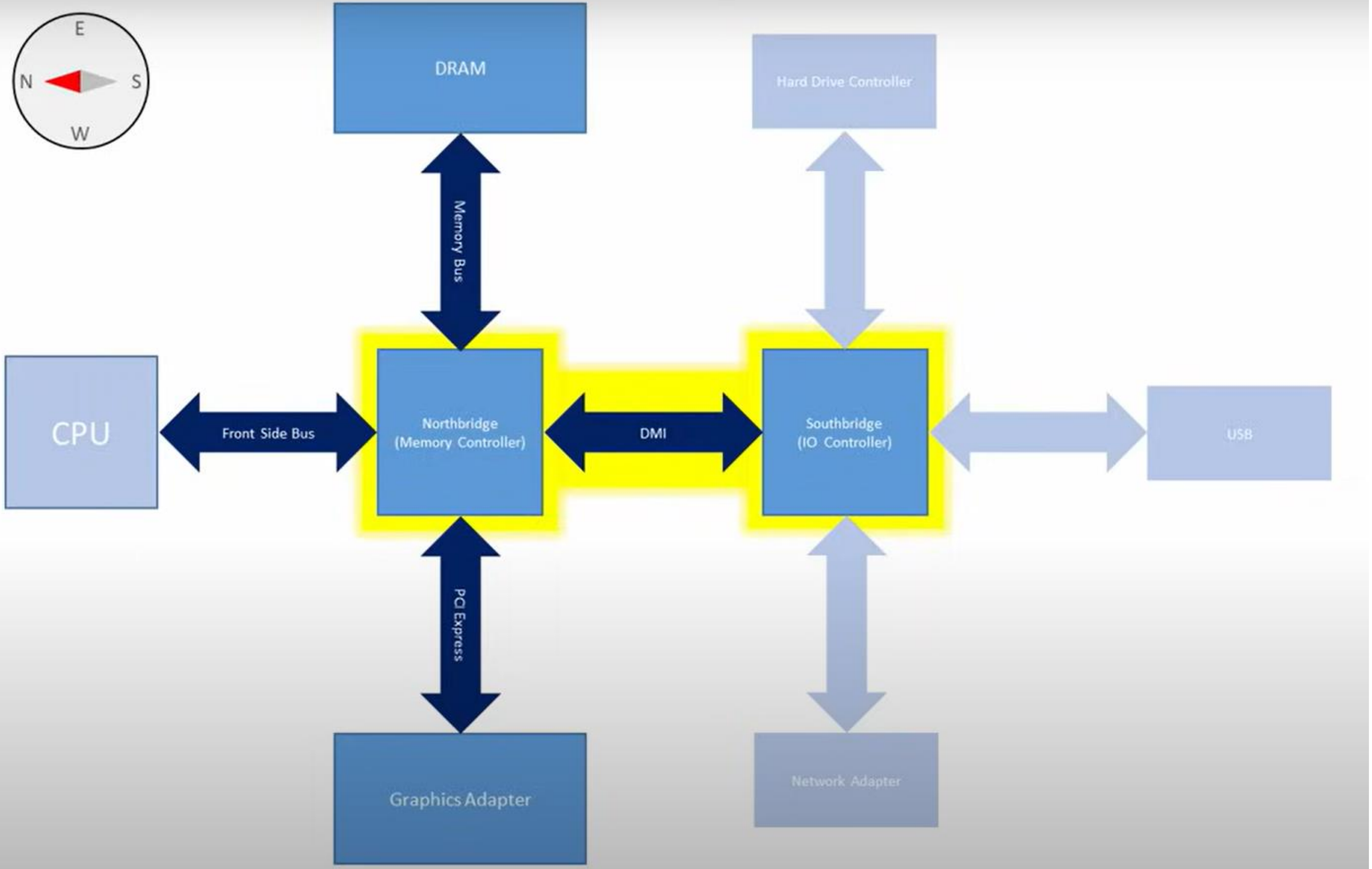
Hot(2) = Q2_cur
Hot(1) = Q1_cur
Hot(0) = Q0_cur

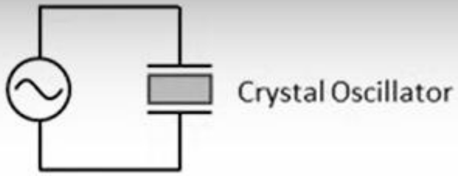


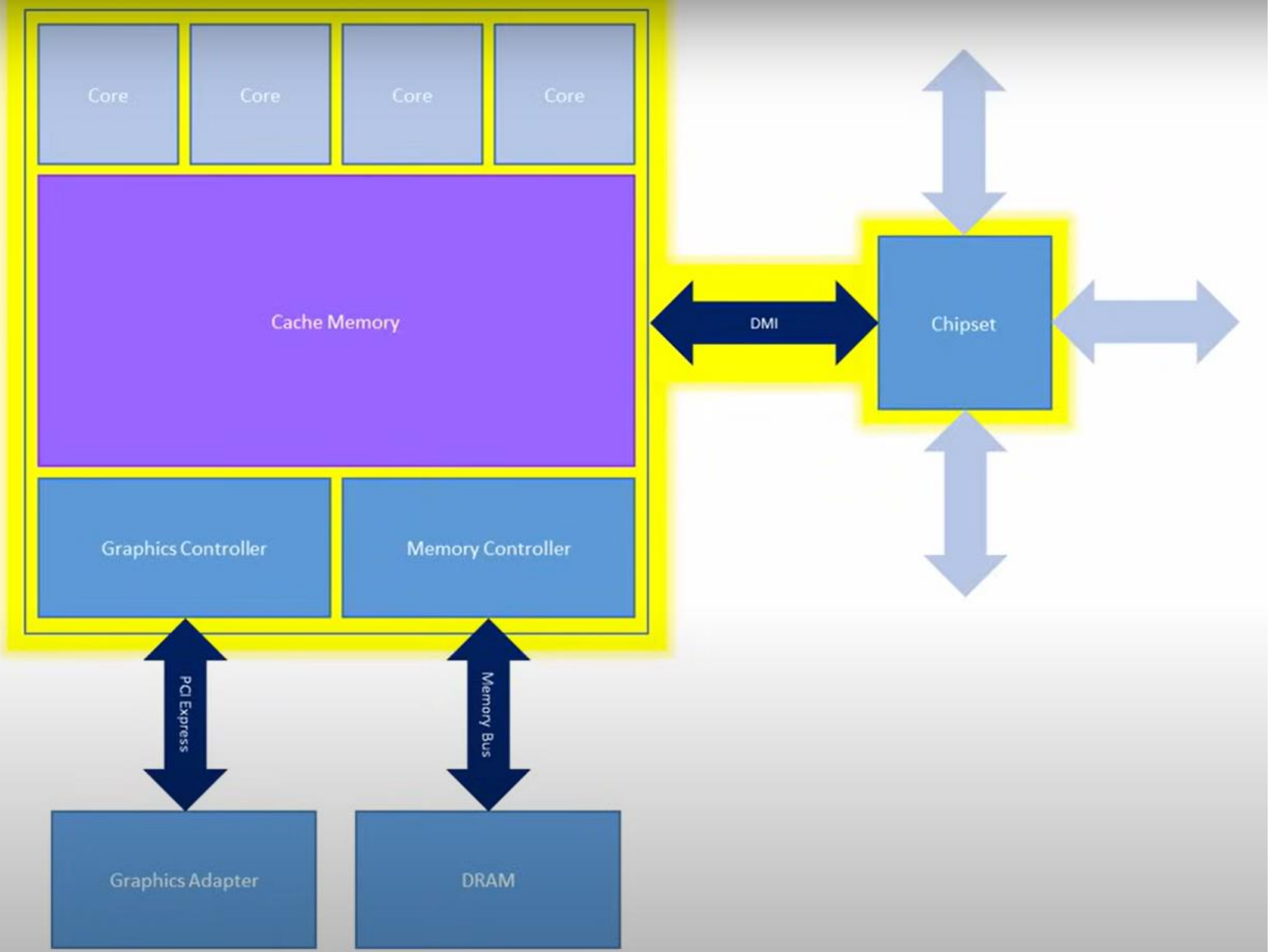
Example: 3-Bit One-Hot Up/Down Counter (Part 3)

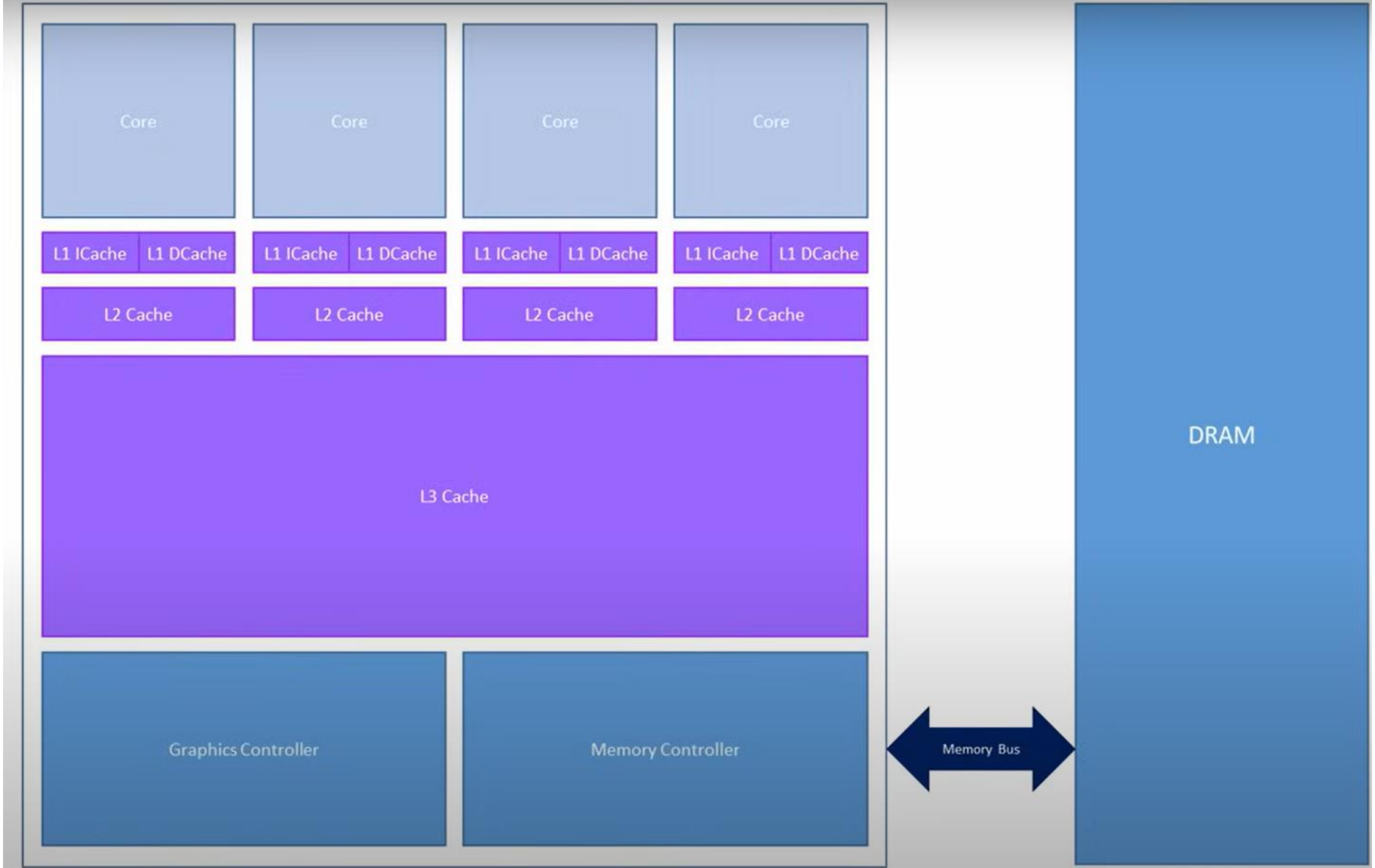
Logic Diagram

Timing Diagram


E
N
S
W
DRAM
Memory Bus
Hard Drive Controller
CPU
Front Side Bus
Northbridge
(Memory Controller)
DMI
Southbridge
(IO Controller)
USB
PCI Express
Graphics Adapter
Network Adapter


Crystal Oscillator


Core
Core
Core
Core
Cache Memory
DMI
Chipset
Graphics Controller
Memory Controller
PCI Express
Memory Bus
Graphics Adapter
DRAM



Core

Core

Core

Core

L1 ICache

L1 DCache

L1 ICache

L1 DCache

L1 ICache

L1 DCache

L1 ICache

L1 DCache

L2 Cache

L2 Cache

L2 Cache

L2 Cache

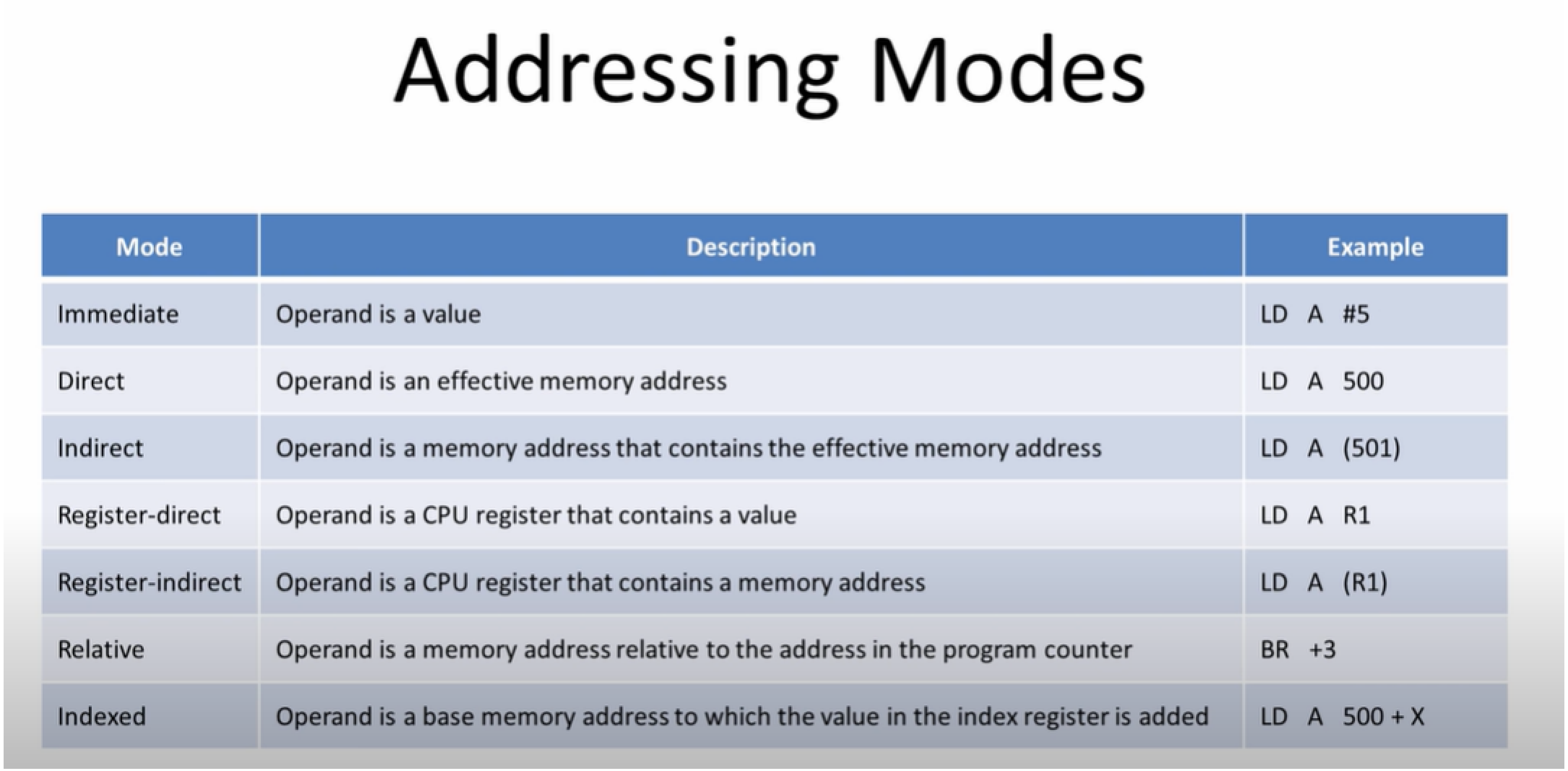
L3 Cache

Graphics Controller

Memory Controller

Memory Bus

DRAM



Addressing Modes

Mode	Description	Example
Immediate	Operand is a value	LD A #5
Direct	Operand is an effective memory address	LD A 500
Indirect	Operand is a memory address that contains the effective memory address	LD A (501)
Register-direct	Operand is a CPU register that contains a value	LD A R1
Register-indirect	Operand is a CPU register that contains a memory address	LD A (R1)
Relative	Operand is a memory address relative to the address in the program counter	BR +3
Indexed	Operand is a base memory address to which the value in the index register is added	LD A 500 + X