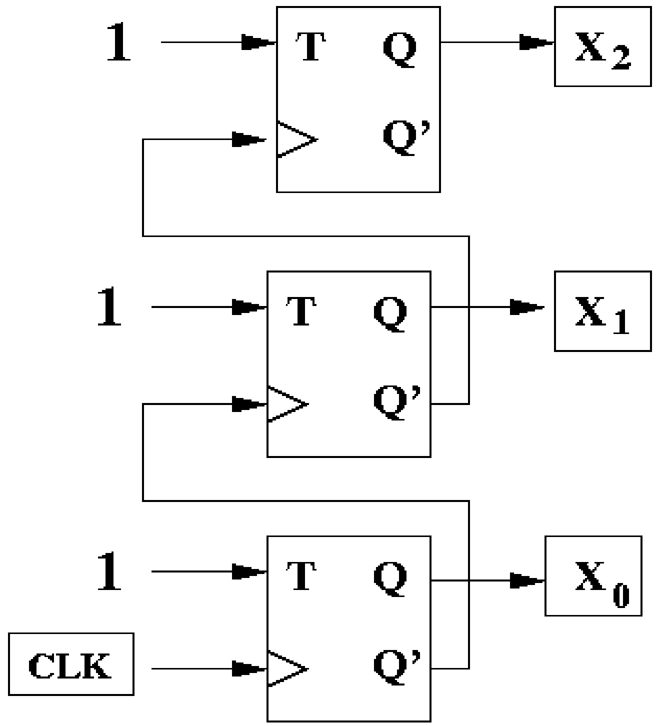
Finite state machines: counter

Asynchronous counter: flip-flops driven by different clocks
Clock period of each successive flip-flop is 2 times previous one
power of 2 times the first clock

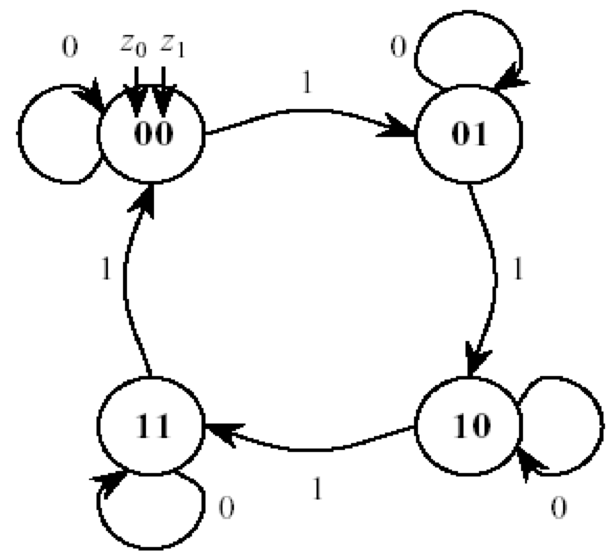


Finite state machines: counter

Use FSM to implement a synchronous counter

2-bit (mod 4) counter
- **starts at 00**
- **counts up to 11**
- **resets to 00 after 11**

Finite state machine
- **state (q): 2 bits, initially 00**
- **output (z): same as state**
- **input**
  - **x = 0: same state**
  - **x = 1: increment**

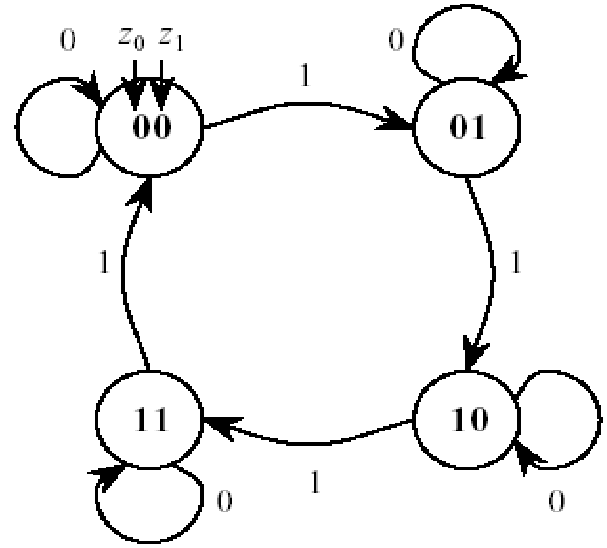


Principles of Computer Architecture by M. Murdocca and V. Heuring

Usage
- **Keeping track of number of bits sent**
- **Program counter (PC)**
  - **Increments each clock cycle to point to next instruction**

Finite state machines: counter

q_1	q_0	x
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

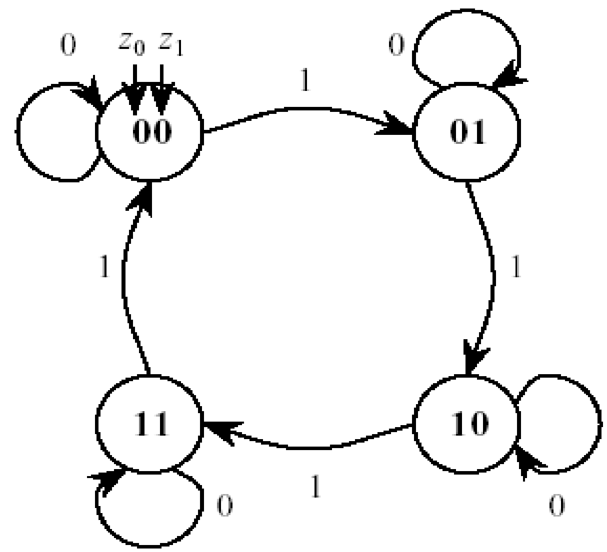


1a. State transition table inputs

Principles of Computer Architecture by M. Murdocca and V. Heuring

Finite state machines: counter

q_1	q_0	x	q_1^+	q_0^+
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0



Principles of Computer Architecture by M. Murdocca and V. Heuring

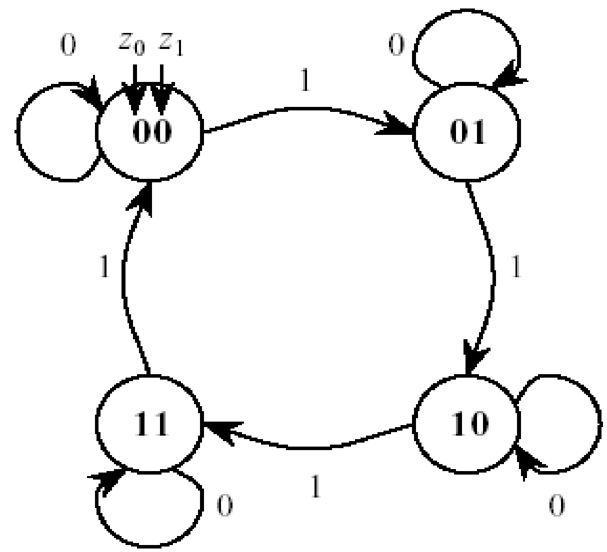
1b. New state

input 0: no change

input 1: increment

Finite state machines: counter

q_1	q_0	x	q_1^+	q_0^+	z_1	z_0
0	0	0	0	0	0	0
0	0	1	0	1	0	0
0	1	0	0	1	0	1
0	1	1	1	0	0	1
1	0	0	1	0	1	0
1	0	1	1	1	1	0
1	1	0	1	1	1	1
1	1	1	0	0	1	1



Principles of Computer Architecture by M. Murdocca and V. Heuring

1c. Output: same as current state label

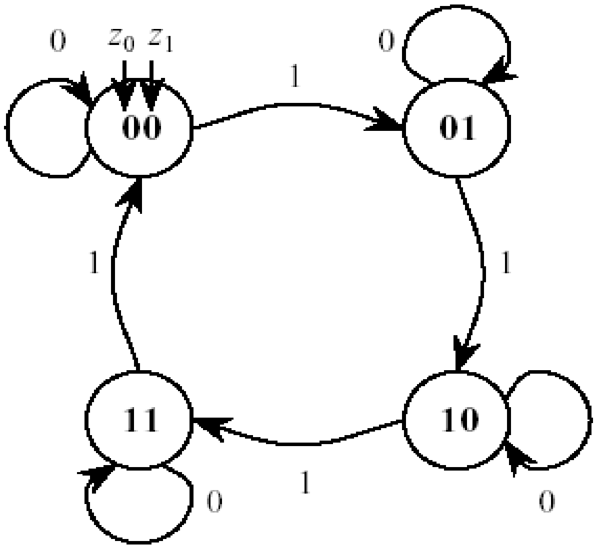
Note that the figure reverses our usual definition of the output bits

Finite state machines: counter

q_1	q_0	x	q_1^+	q_0^+	z_1	z_0	D_1	D_0
0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	1
0	1	0	0	1	0	1	0	1
0	1	1	1	0	0	1	1	0
1	0	0	1	0	1	0	1	0
1	0	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1
1	1	1	0	0	1	1	0	0

2. Pick flip-flops: both D

3. Use excitation tables to get values for D
(copy columns for next state)



Principles of Computer Architecture by M. Murdocca and V. Heuring

Finite state machines: counter

Input			Next		Output				ROM	
q_1	q_0	x	q_1^+	q_0^+	z_1	z_0	D_1	D_0	Address	Data
0	0	0	0	0	0	0	0	0	000	0000
0	0	1	0	1	0	0	0	1	001	0001
0	1	0	0	1	0	1	0	1	010	0101
0	1	1	1	0	0	1	1	0	011	0110
1	0	0	1	0	1	0	1	0	100	1010
1	0	1	1	1	1	0	1	1	101	1011
1	1	0	1	1	1	1	1	1	110	1111
1	1	1	0	0	1	1	0	0	111	1100

4. Draw circuit: ROM

address: q_1q_0x

data: $z_1z_0D_1D_0$

Finite state machines: counter

Input			Next		Output				Minterms
q_1	q_0	x	q_1^+	q_0^+	z_1	z_0	D_1	D_0	z_1
0	0	0	0	0	0	0	0	0	
0	0	1	0	1	0	0	0	1	
0	1	0	0	1	0	1	0	1	
0	1	1	1	0	0	1	1	0	
1	0	0	1	0	1	0	1	0	$q_1\backslash q_0\backslash x$
1	0	1	1	1	1	0	1	1	$q_1\backslash q_0 x$
1	1	0	1	1	1	1	1	1	$q_1 q_0\backslash x$
1	1	1	0	0	1	1	0	0	$q_1 q_0 x$

4. Draw circuit: gates

Minterms

$z_1 = q_1\backslash q_0\backslash x + q_1\backslash q_0 x + q_1 q_0\backslash x + q_1 q_0 x$

etc.

Simplified

$z_1 = q_1$

$z_0 = q_0$

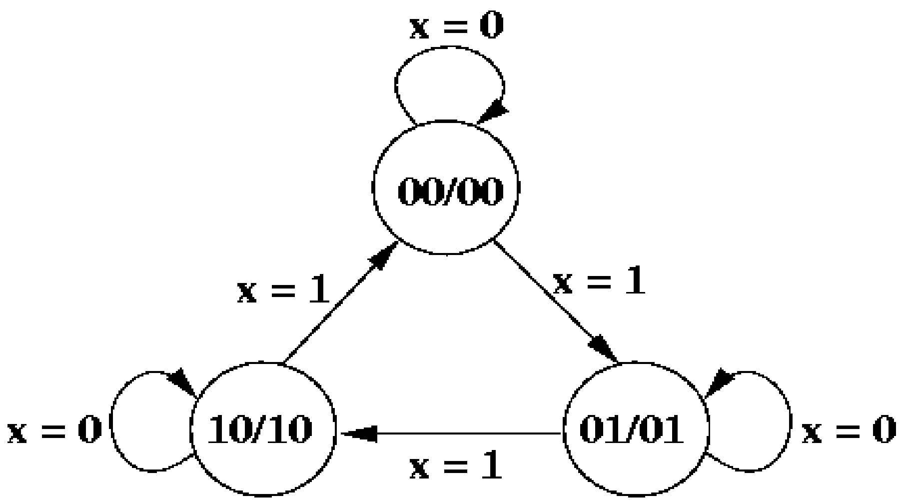
$D_1 = q_1\backslash q_0 + q_0 (\backslash q_1 x + q_1\backslash x)$

$D_0 = \backslash q_0 x + q_0\backslash x$

Finite state machines: 3-state counter

Note that it is not necessary to use all possible states for the counter

3-state counter: reset to 00 after 10

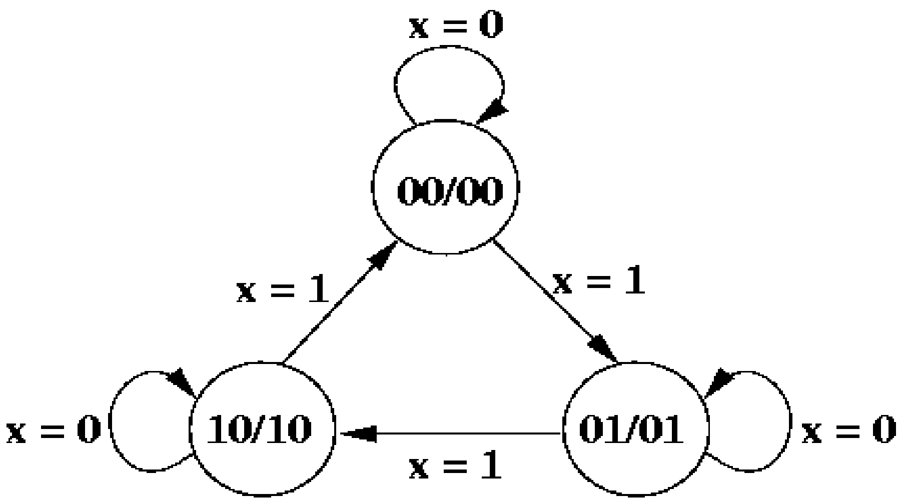


Changes:

Replace entries for state 11 in state transition table with "d"

Next state after state 10 is 00 with input 1

Finite state machines: 3-state counter

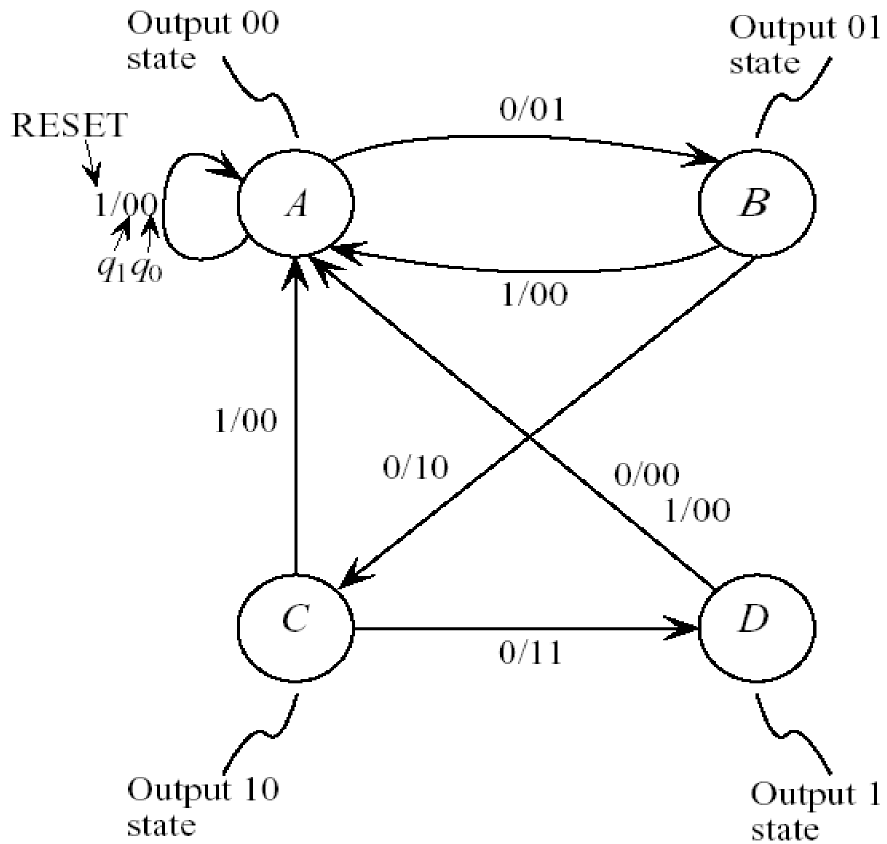


Input			Next		Output			
q_1	q_0	x	q_1^+	q_0^+	z_1	z_0	D_1	D_0
0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	1
0	1	0	0	1	0	1	0	1
0	1	1	1	0	0	1	1	0
1	0	0	1	0	1	0	1	0
1	0	1	0	0	1	0	0	0
1	1	0	d	d	d	d	d	d
1	1	1	d	d	d	d	d	d

Finite state machines: counter

Other possible counter variations

- **Use Mealy machine**
- **Use input to reset**
  - **Input 0: increment**
  - **Input 1: reset**



Principles of Computer Architecture by M. Murdocca and V. Heuring

Finite state machines: counter

Other possible counter variations

- Decrement
  - Input 0: hold
  - Input 1: decrement
- Increment/decrement
  - Input 0: increment
  - Input 1: decrement
- Additional inputs
  - Asynchronous clear: reset value immediately to 00
  - Enable/disable
    - When this input is 0, counter continues to output current value
    - When 1, perform normal operations
- Additional output
  - Counter out
    - Normally 1, but 0 when maximum value is reached
    - What could this be used for?
    - (Think of a connection with enable)