



# Digital Design

## Chapter 4: Datapath Components

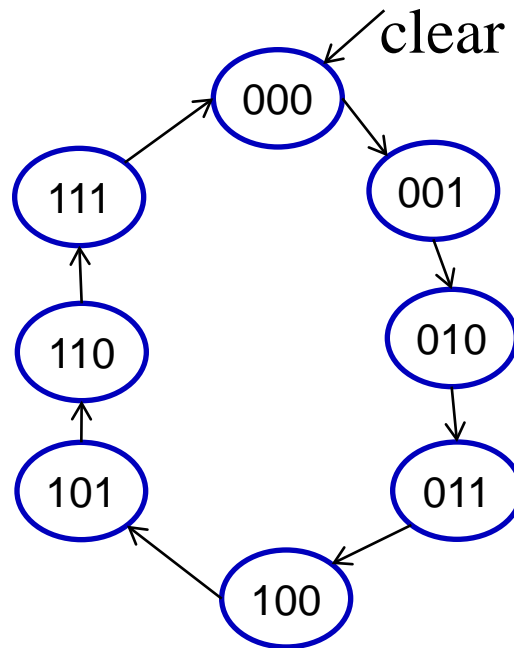
Slides to accompany the textbook *Digital Design*, First Edition,  
by Frank Vahid, John Wiley and Sons Publishers, 2007.  
<http://www.ddvahid.com>

Copyright © 2007 Frank Vahid

Instructors of courses requiring Vahid's *Digital Design* textbook (published by John Wiley and Sons) have permission to modify and use these slides for customary course-related activities, subject to keeping this copyright notice in place and unmodified. These slides may be posted as unanimated pdf versions on publicly-accessible course websites.. PowerPoint source (or pdf with animations) may not be posted to publicly-accessible websites, but may be posted for students on internal protected sites or distributed directly to students by other electronic means. Instructors may make printouts of the slides available to students for a reasonable photocopying charge, without incurring royalties. Any other use requires explicit permission. Instructors may obtain PowerPoint source or obtain special use permissions from Wiley – see <http://www.ddvahid.com> for information.

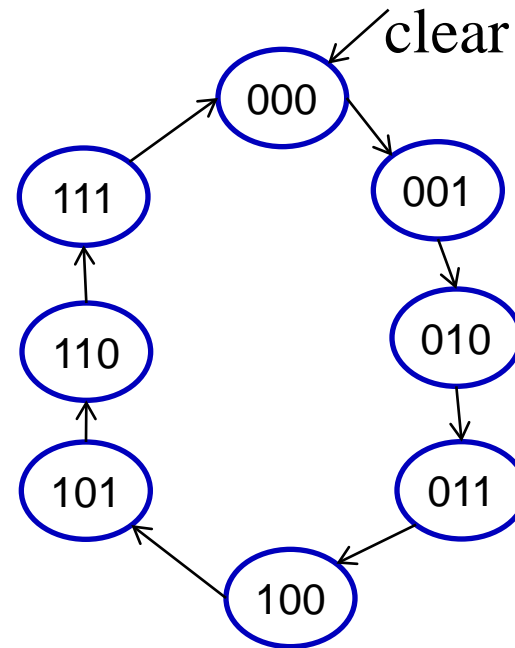
# Counters

- A counter is simply a FSM with the outputs equal to the register outputs.
- The state assignments are chosen to match the desired counting sequence.



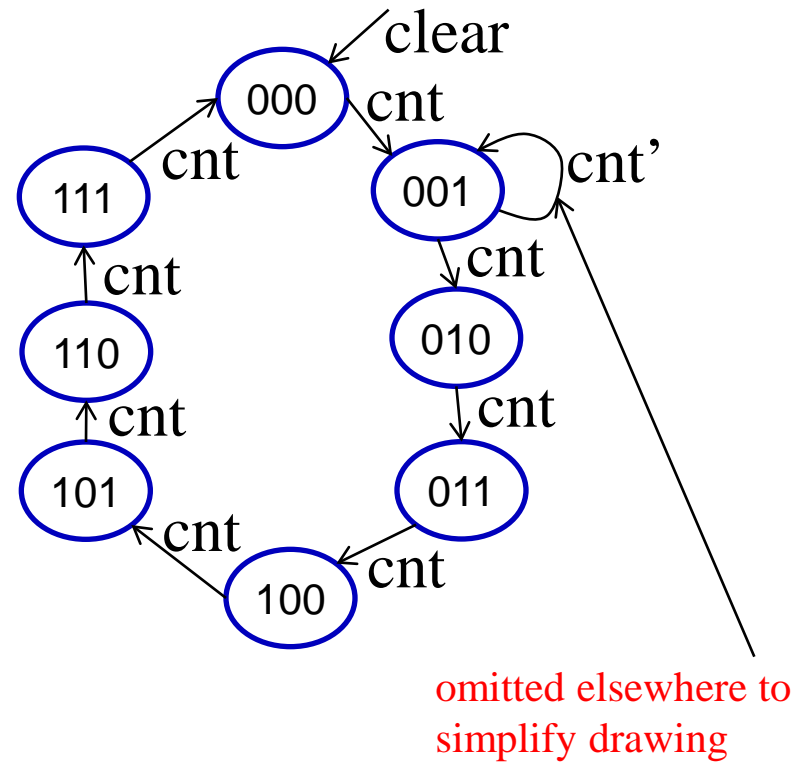
# State Table

present state	next state
$S_2S_1S_0$	$S_2S_1S_0$
000	001
001	010
010	011
011	100
101	110
110	111
111	000



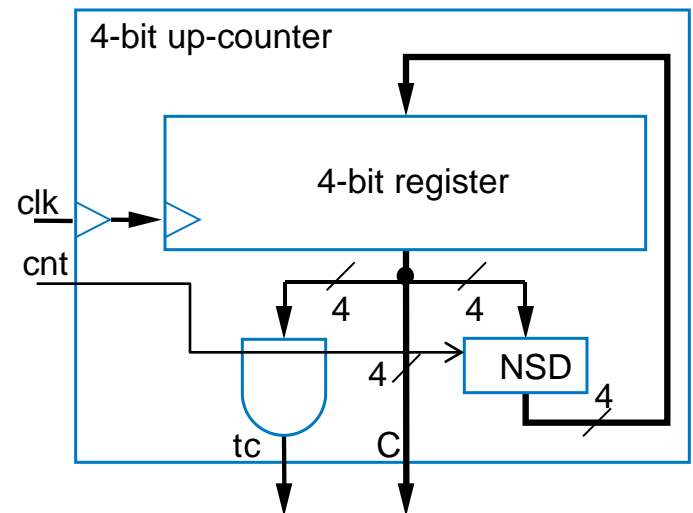
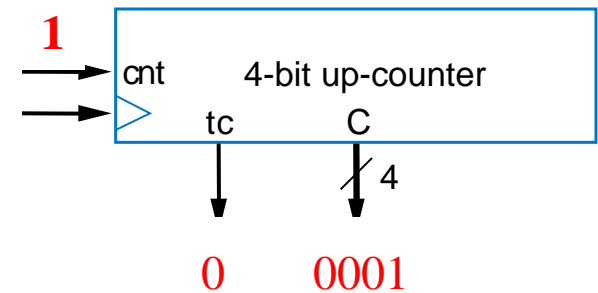
# Counter with Additional Control Input

present state	next state
cnt S <sub>2</sub> S <sub>1</sub> S <sub>0</sub>	S <sub>2</sub> S <sub>1</sub> S <sub>0</sub>
0 000	000
0 001	001
0 010	010
0 011	011
0 101	101
0 110	110
0 111	111
1 000	001
1 001	010
1 010	011
1 011	100
1 101	110
1 110	111
1 111	000



# Counters

- ***N-bit up-counter***: N-bit register that can increment (add 1) to its own value on each clock cycle
  - 0000, 0001, 0010, 0011, ....., 1110, 1111, 0000
  - Note how count “rolls over” from 1111 to 0000
    - Terminal (last) count, tc, equals 1 during value just before rollover
- Internal design
  - Register, next state decoder (NSD), and N-input AND gate to detect terminal count



# State Table

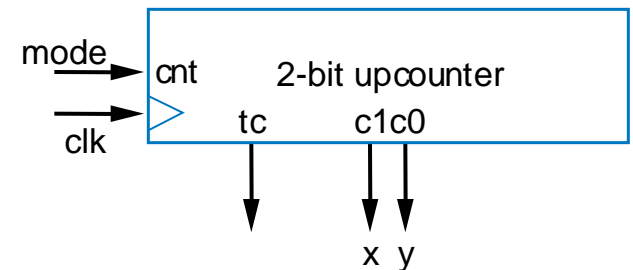
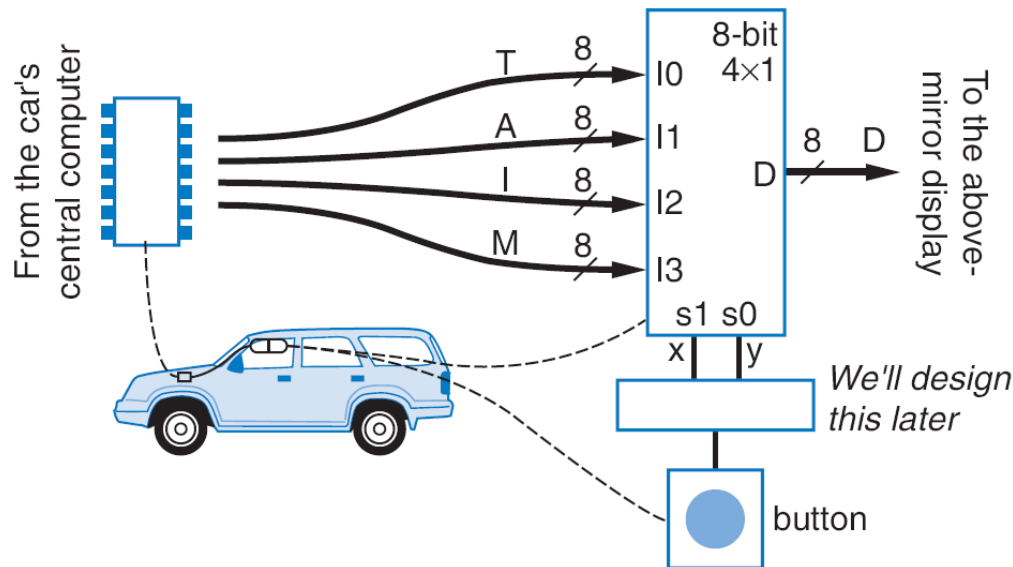
- Can build faster incrementer using combinational logic design process
  - Capture truth table
  - Derive equation for each output
    - $c0 = a3a2a1a0$
    - ...
    - $s0 = a0'$
  - Results in small and fast circuit
  - Note: works for small N -- larger N leads to exponential growth, like for N-bit adder
  - $c0$  can be used as ripple carry to next counter

Present State					Next State			
s3	s2	s1	s0	c0	s3	s2	s1	s0
0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	1	0
0	0	1	0	0	0	0	1	1
0	0	1	1	0	0	1	0	0
0	1	0	0	0	0	1	0	1
0	1	0	1	0	0	1	1	0
0	1	1	0	0	0	1	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	0	1	0	0	1
1	0	0	1	0	1	0	1	0
1	0	1	0	0	1	0	1	1
1	0	1	1	0	1	1	0	0
1	1	0	0	0	1	1	0	1
1	1	0	1	0	1	1	1	0
1	1	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	0



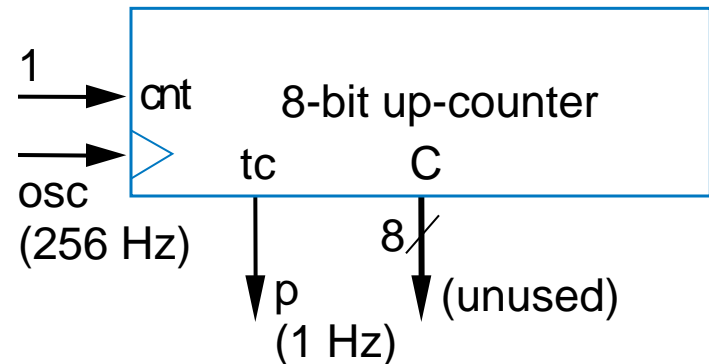
# Counter Example: Mode in Above-Mirror Display

- Recall above-mirror display example from Chapter 2
  - Assumed component that incremented xy input each time button pressed: 00, 01, 10, 11, 00, 01, 10, 11, 00, ...
  - Can use 2-bit up-counter
    - Assumes mode=1 for just one clock cycle during each button press
      - Recall “Button press synchronizer” example from Chapter 3



# Counter Example: 1 Hz Pulse Generator Using 256 Hz Oscillator

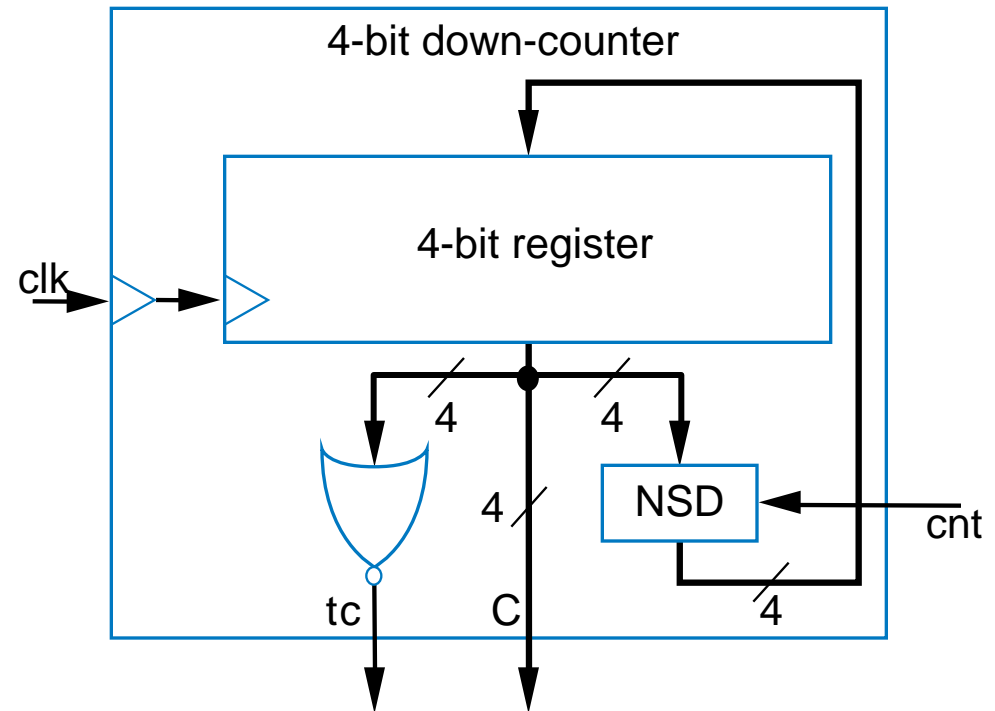
- Suppose we have 256 Hz oscillator, but want 1 Hz pulse
  - 1 Hz is 1 pulse per second -- useful for keeping time
  - Design using 8-bit up-counter, use tc output as pulse
    - Counts from 0 to 255 (256 counts), so pulses tc every 256 cycles





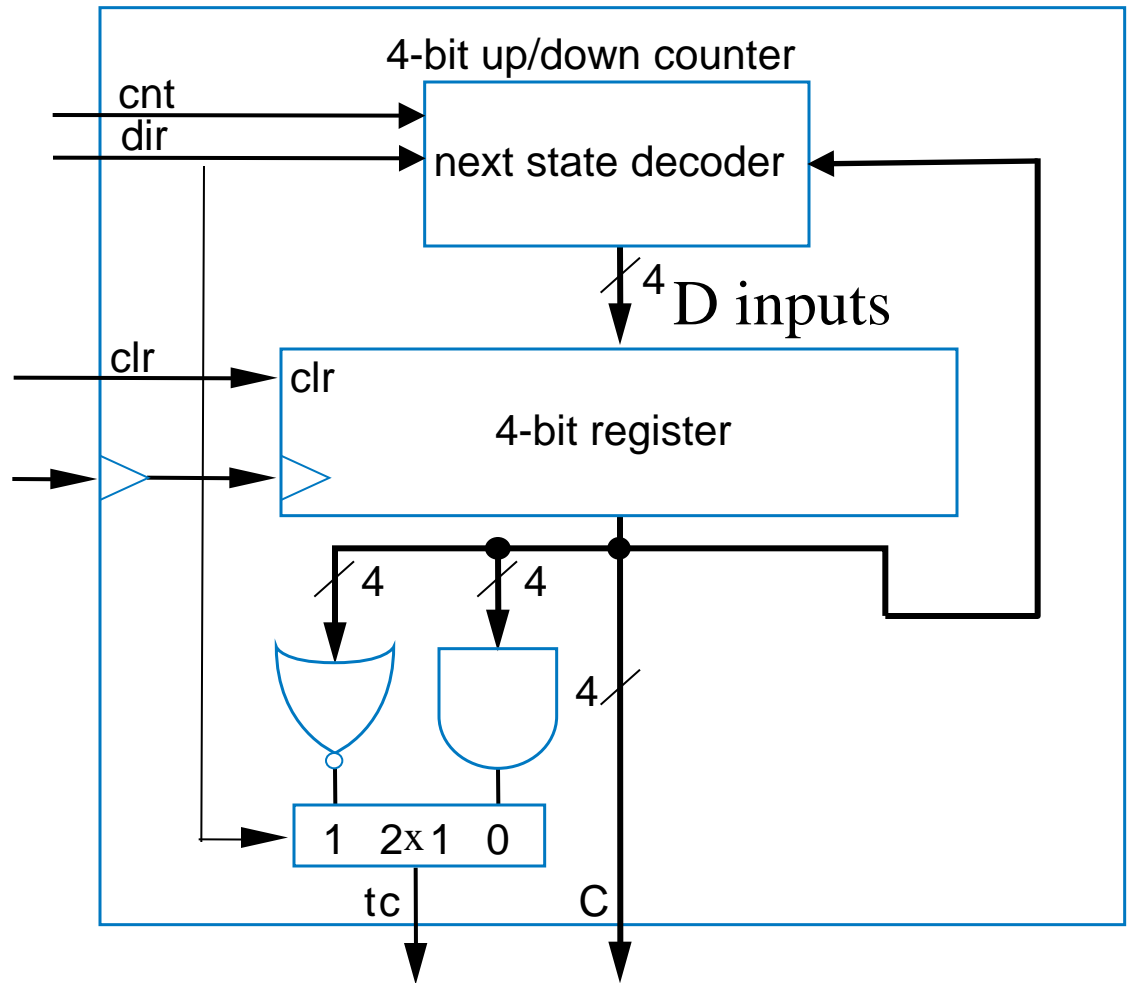
# Down-Counter

- 4-bit down-counter
  - 1111, 1110, 1101, 1100, ..., 0011, 0010, 0001, 0000, 1111, ...
  - Terminal count is 0000
    - Use NOR gate to detect



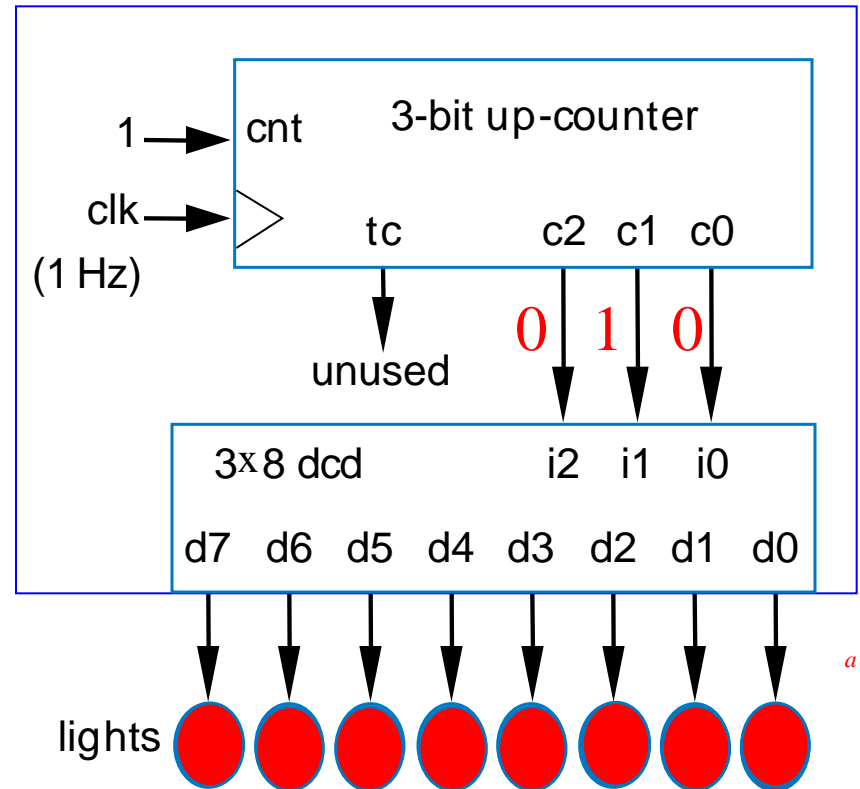
# Up/Down-Counter

- Can count either up or down
  - Use dir input to select, using 2x1: dir=0 means up
  - Likewise, dir selects appropriate terminal count value



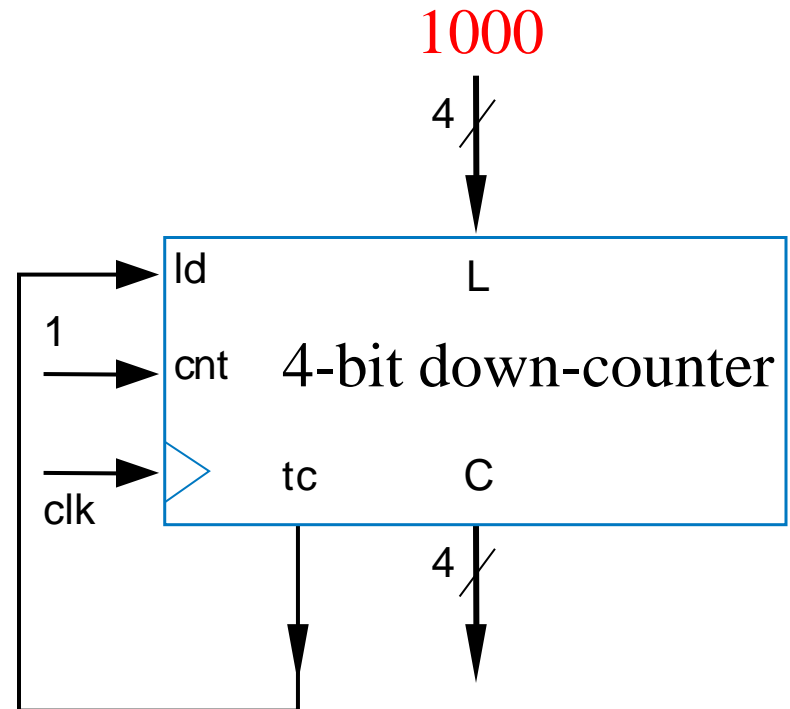
# Counter Example: Light Sequencer

- Illuminate 8 lights from right to left, one at a time, one per second
- Use 3-bit up-counter to counter from 0 to 7
- Use 3x8 decoder to illuminate appropriate light
- Note: Used **3-bit** counter with 3x8 decoder
  - *NOT* an 8-bit counter – why not?



# Counter with Parallel Load

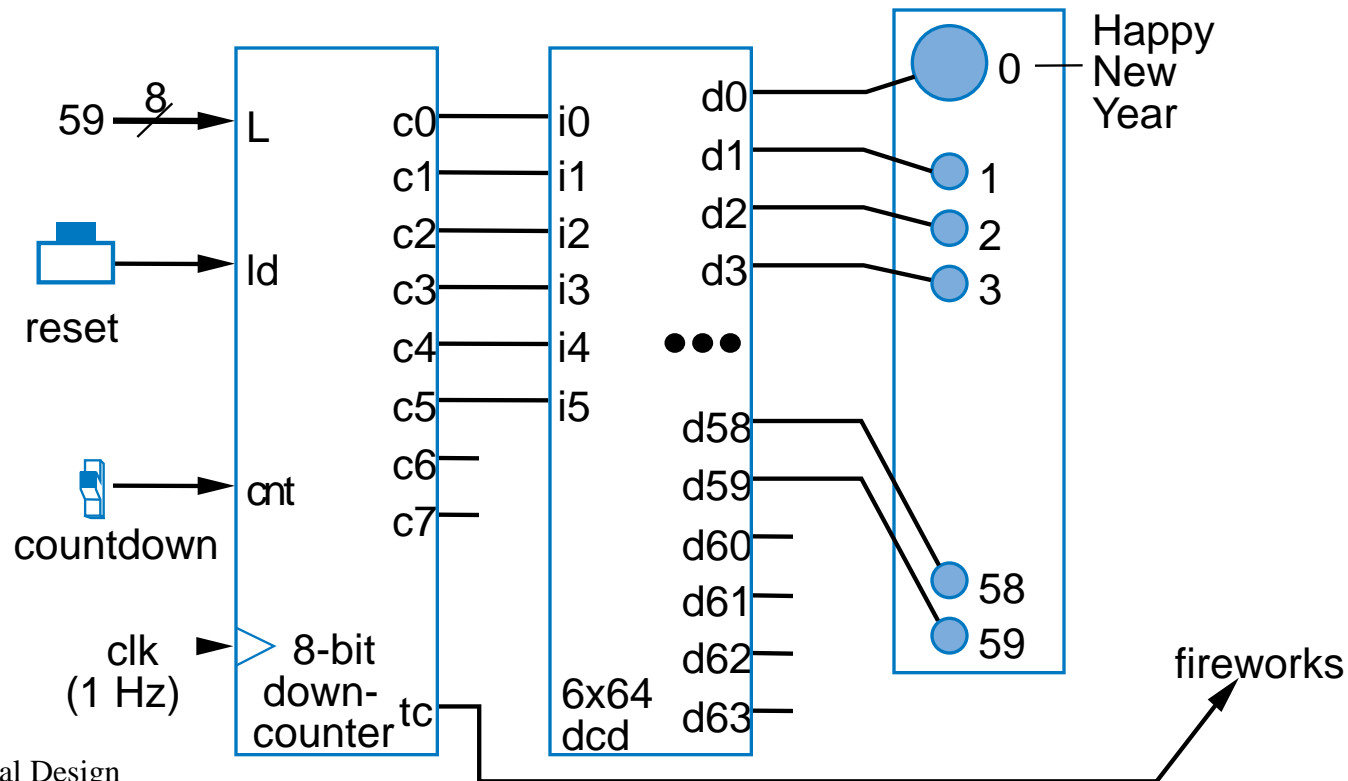
- Useful to create pulses at specific multiples of clock
  - Not just at N-bit counter's natural wrap-around of  $2^N$
- Example: Pulse every 9 clock cycles
  - Use 4-bit down-counter with parallel load
  - Set parallel load input to 8 (1000)
  - Use terminal count to reload
    - When count reaches 0, next cycle loads 8.
  - Why load 8 and not 9? Because 0 is included in count sequence:
    - 8, 7, 6, 5, 4, 3, 2, 1, 0 → 9 counts



# Counter Example:

## New Year's Eve Countdown Display

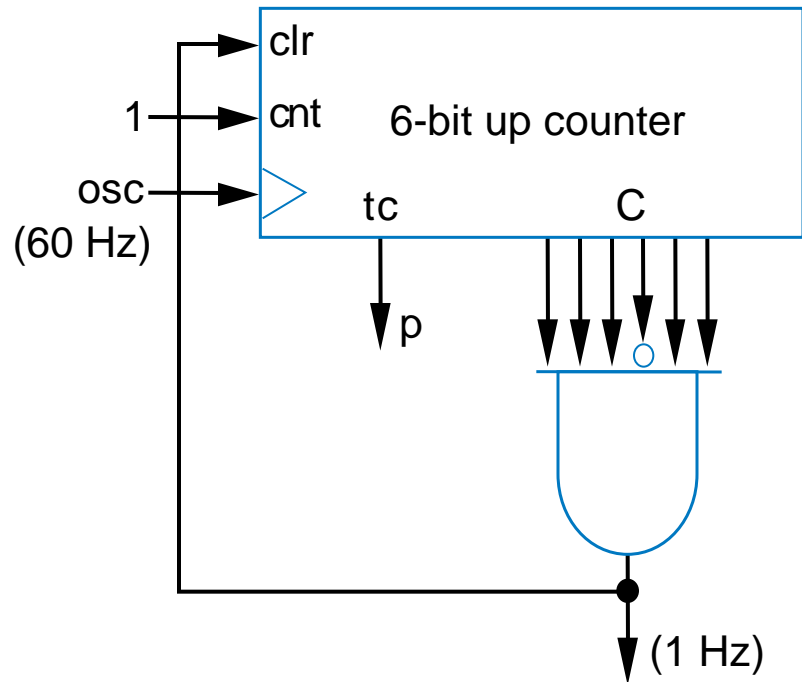
- Chapter 2 example previously used microprocessor to count from 59 down to 0 in binary
- Can use 8-bit (or 7- or 6-bit) down-counter instead, initially loaded with 59



# Counter Example:

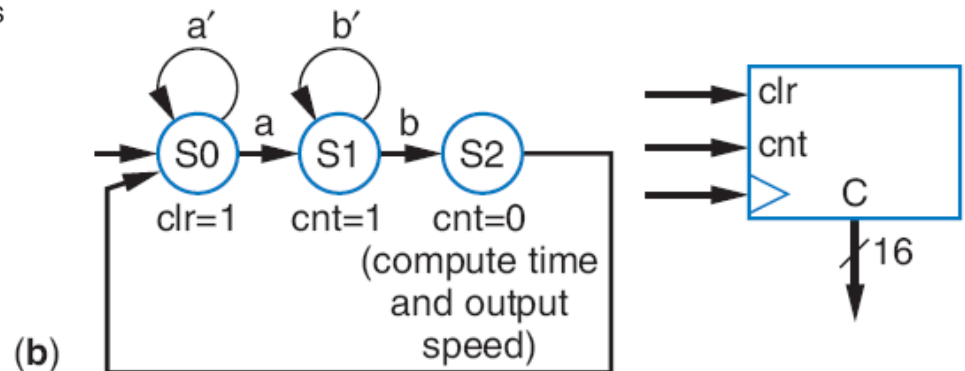
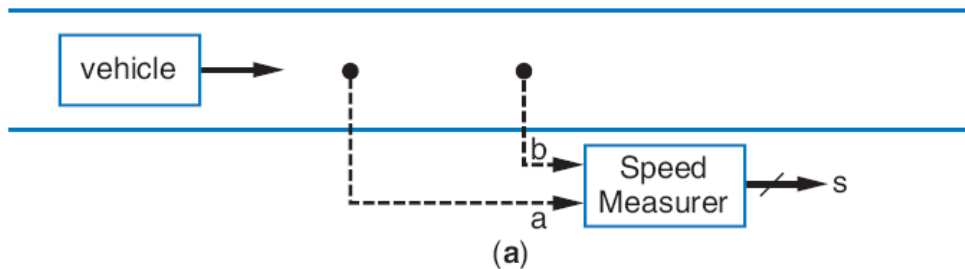
## 1 Hz Pulse Generator from 60 Hz Clock

- U.S. electricity standard uses 60 Hz signal
  - Device may convert that to 1 Hz signal to count seconds
- Use 6-bit up-counter
  - Can count from 0 to 63
  - Create simple logic to detect 59 (for 60 counts)
    - Use to clear the counter back to 0 (or to load 0)
    - Do not use asynchronous clear



# Timer

- A type of counter used to measure time
  - If we know the counter's clock frequency and the count, we know the time that's been counted
- Example: Compute car's speed using two sensors
  - First sensor (a) clears and starts timer
  - Second sensor (b) stops timer
  - Assuming clock of 1kHz, timer output represents time to travel between sensors. Knowing the distance, we can compute speed



# Lab #7

- Design a 3-bit counter using logic gates and flip-flops that functions according to the table shown below. Demonstrate your simulation in lab.
- Due week of March 16 during lab.

ld	cnt	dir	mode
0	0	0	HOLD
0	0	1	HOLD
0	1	0	COUNT DOWN
0	1	1	COUNT UP
1	0	0	PARALLEL LOAD
1	0	1	PARALLEL LOAD
1	1	0	PARALLEL LOAD
1	1	1	PARALLEL LOAD

ld	cnt	dir	mode
0	0	x	HOLD
0	1	0	COUNT DOWN
0	1	1	COUNT UP
1	x	x	PARALLEL LOAD



# Comments on Lab Assignment

- Draw the state diagram
  - The state assignments follow the count sequence
- Create the present/next state table
- Create the Karnaugh maps for each flip-flop D input
- Create a separate logic circuit for each state flip-flop
- Write a test bench for each state flip-flop and verify correct functionality
- Combine the state flip-flops into a next state decoder (NSD)
- Simulate the counter and verify correct functionality