EECS150 - Digital Design

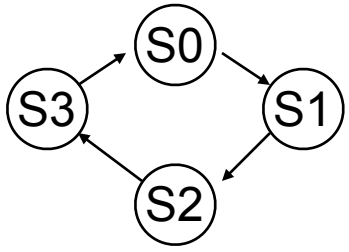
Lecture 18 - Counters

March 14, 2012

John Wawrzynek

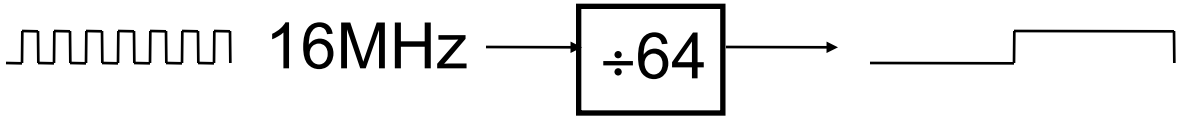
Counters

- Special sequential circuits (FSMs) that repeatedly sequence through a set of outputs.
- Examples:
  - binary counter: 000, 001, 010, 011, 100, 101, 110, 111, 000,
  - gray code counter:
    000, 010, 110, 100, 101, 111, 011, 001, 000, 010, 110, …
  - one-hot counter: 0001, 0010, 0100, 1000, 0001, 0010, …
  - BCD counter: 0000, 0001, 0010, …, 1001, 0000, 0001
  - pseudo-random sequence generators: 10, 01, 00, 11, 10, 01, 00, …
- Moore machines with "ring" structure in State Transition Diagram:



What are they used?

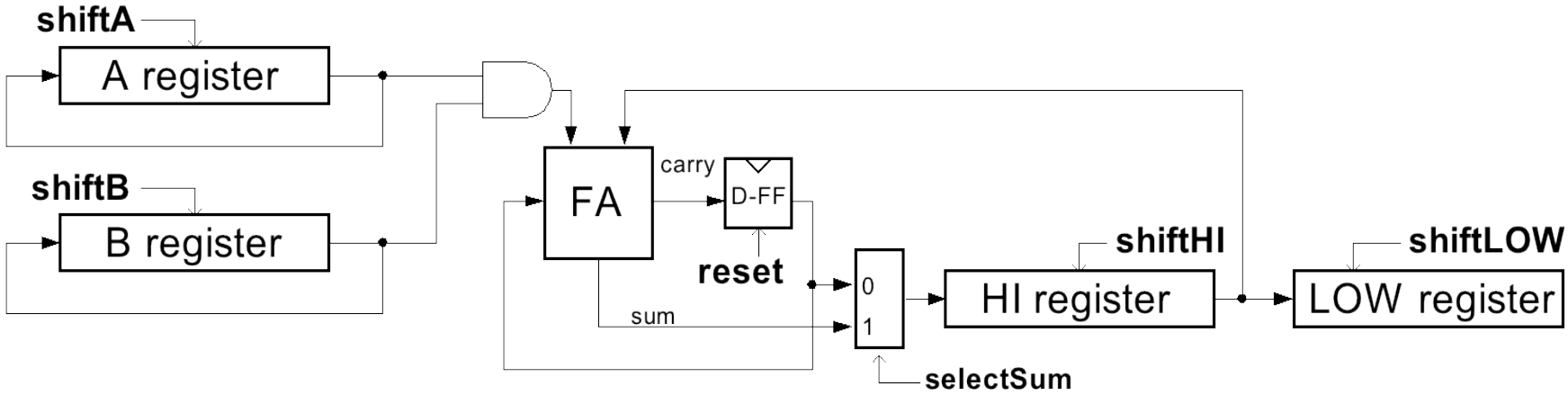
- Counters are commonly used in hardware designs because most (if not all) computations that we put into hardware include iteration (looping). Examples:
  - Shift-and-add multiplication scheme.
  - Bit serial communication circuits (must count one "words worth" of serial bits.
- Other uses for counter:
  - Clock divider circuits



  - Systematic inspection of data-structures
    - Example: Network packet parser/filter control.
- Counters simplify "controller" design by:
  - providing a specific number of cycles of action,
  - sometimes used with a decoder to generate a sequence of timed control signals.
  - Consider using a counter when many FSM states with few branches.

Controller using Counters

- Example, Bit-serial multiplier (n^2 cycles, one bit of result per n cycles):



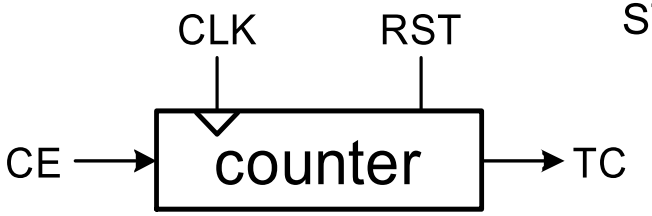
- Control Algorithm:

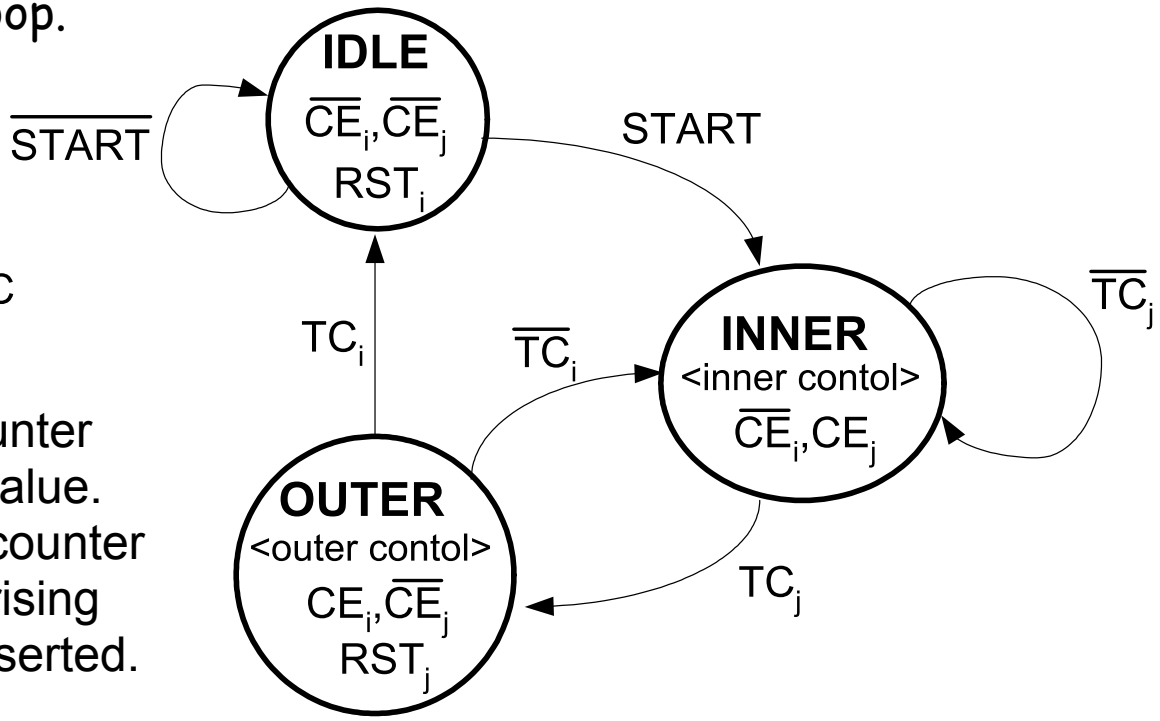
```
repeat n cycles {   // outer (i) loop
        repeat n cycles{    // inner (j) loop
                shiftA, selectSum, shiftHI
        }
        shiftB, shiftHI, shiftLOW, reset
}
```

Note: The occurrence of a control signal x means x=1. The absence of x means x=0.

Controller using Counters

- **State Transition Diagram:**
  - Assume presence of two binary counters. An "i" counter for the outer loop and "j" counter for inner loop.

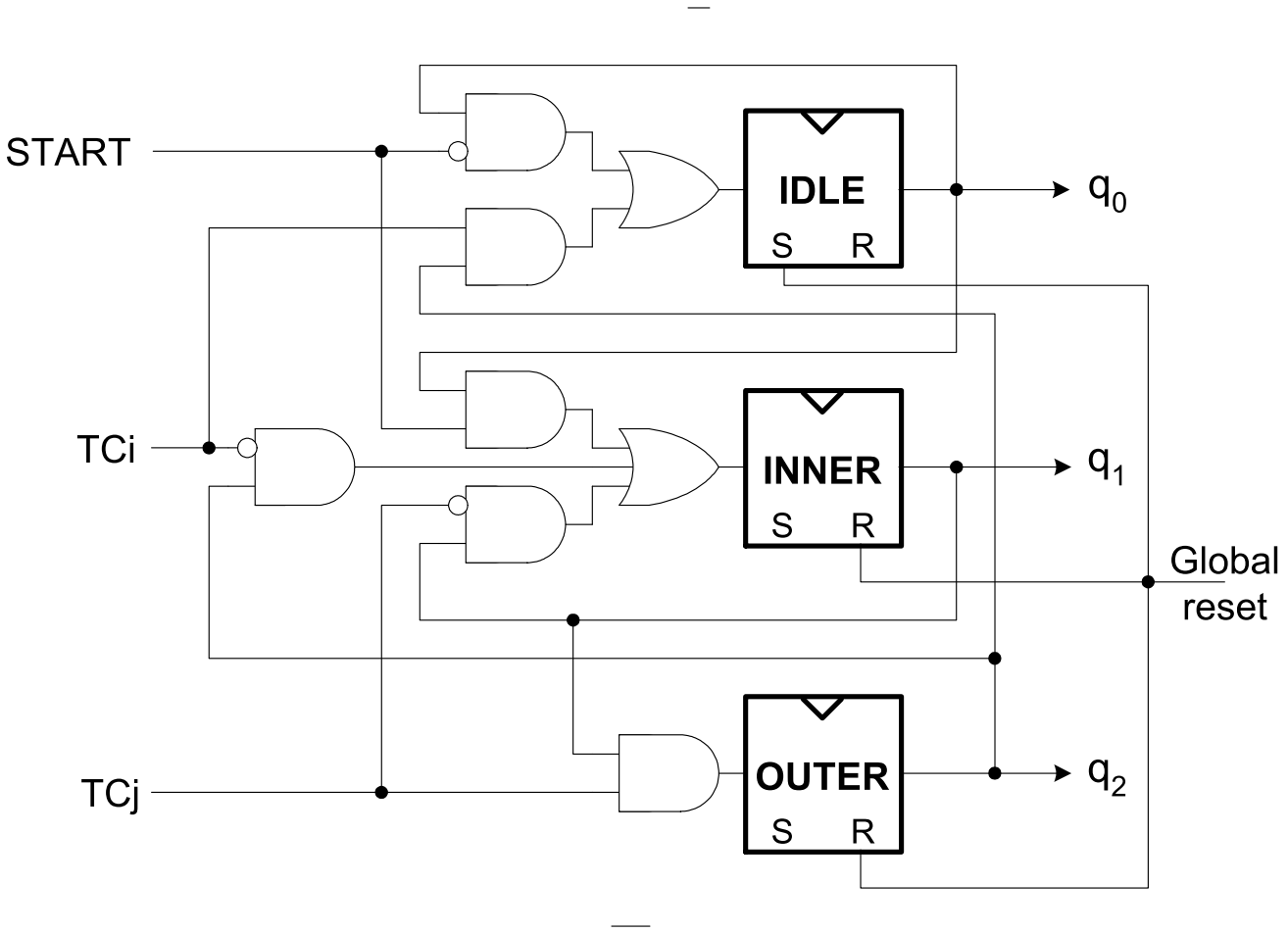




TC is asserted when the counter reaches it maximum count value. **CE** is "count enable". The counter increments its value on the rising edge of the clock if CE is asserted.

Controller using Counters

- **Controller circuit implementation:**



- Outputs:

$CE_i = q_2$

$CE_j = q_1$

$RST_i = q_0$

$RST_j = q_2$

$shiftA = q_1$

$shiftB = q_2$

$shiftLOW = q_2$

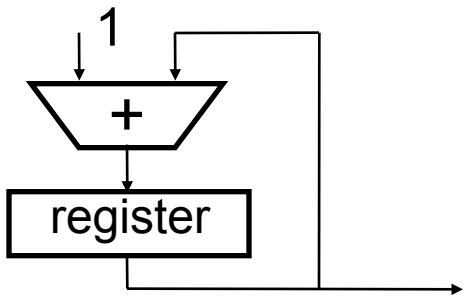
$shiftHI = q_1 + q_2$

$reset = q_2$

$selectSUM = q_1$

How do we design counters?

- For binary counters (most common case) incrementer circuit would work:



- In Verilog, a counter is specified as: x = x+1;
  - This does *not* imply an adder
  - An incrementer is simpler than an adder
  - And a counter might be simpler yet.

- In general, the best way to understand counter design is to think of them as FSMs, and follow general procedure, however some special cases can be optimized.

Synchronous Counters

All outputs change with clock edge.

- Binary Counter Design:

  Start with 3-bit version and generalize:

c	b	a	c^+	b^+	a^+
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

$a^+ = a'$

$b^+ = a \oplus b$

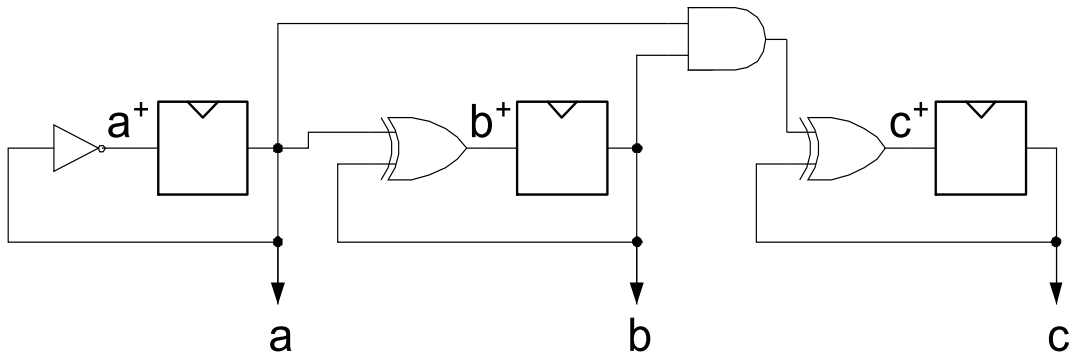
$c^+ = abc' + a'b'c + ab'c + a'bc$

$= a'c + abc' + b'c$

$= c(a'+b') + c'(ab)$

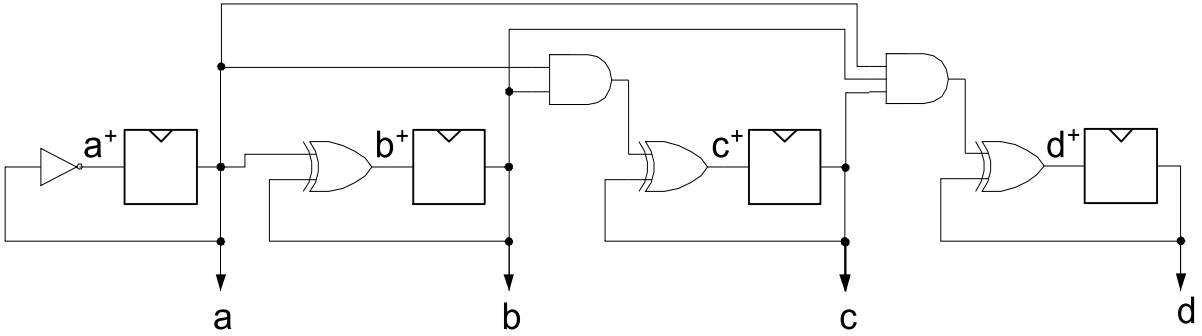
$= c(ab)' + c'(ab)$

$= c \oplus ab$

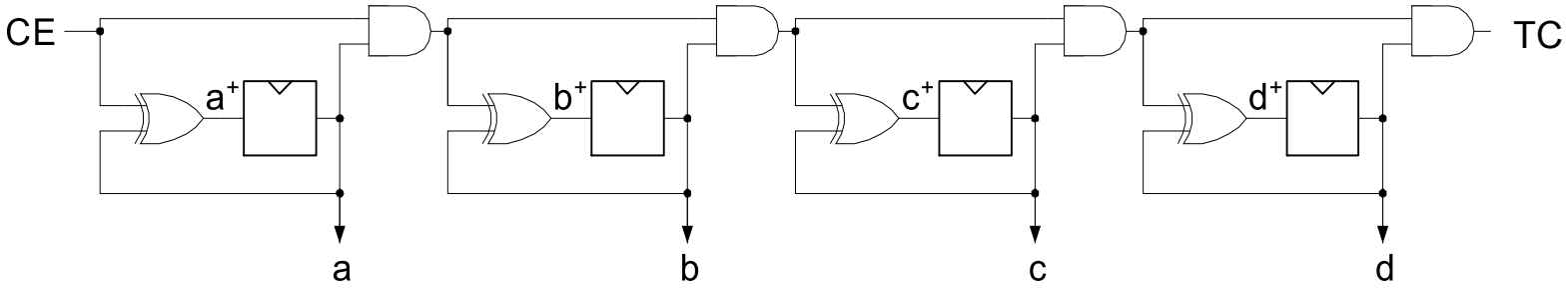


Synchronous Counters

- How do we extend to n-bits?
- Extrapolate c^+: $d^+ = d \oplus abc$, $e^+ = e \oplus abcd$

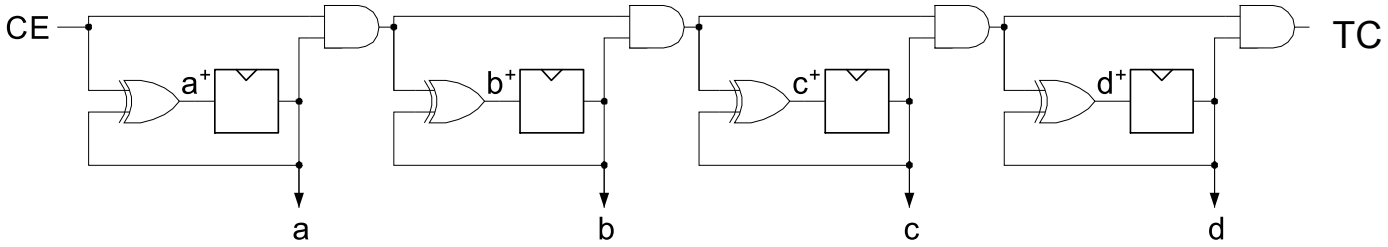


- Has difficulty scaling (AND gate inputs grow with n)



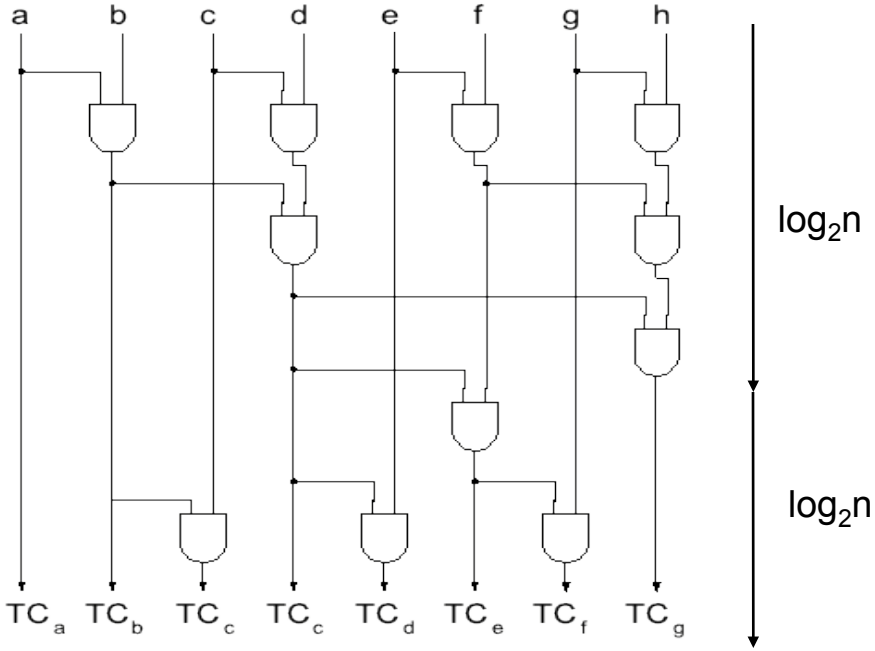
- CE is "count enable", allows external control of counting,
- TC is "terminal count", is asserted on highest value, allows cascading, external sensing of occurrence of max value.

Synchronous Counters

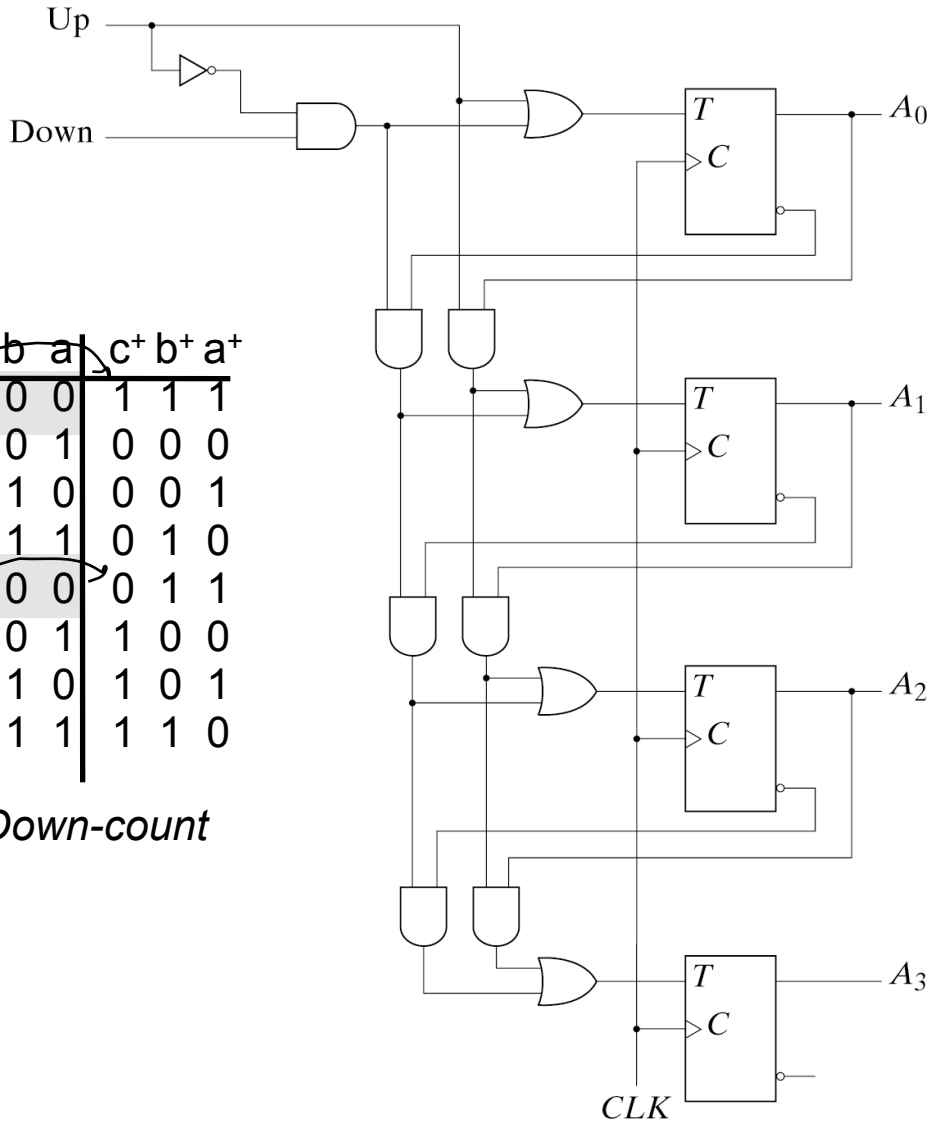


- How does this one scale?
- ☹ Delay grows α n

- Generation of TC signals very similar to generation of carry signals in adder.
- "Parallel Prefix" circuit reduces delay:

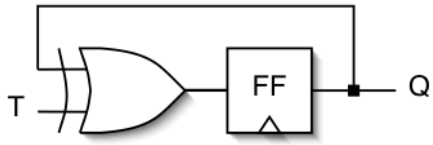


Up-Down Counter



c	b	a	c^+	b^+	a^+
0	0	0	1	1	1
0	0	1	0	0	0
0	1	0	0	0	1
0	1	1	0	1	0
1	0	0	0	1	1
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	1	0

Down-count

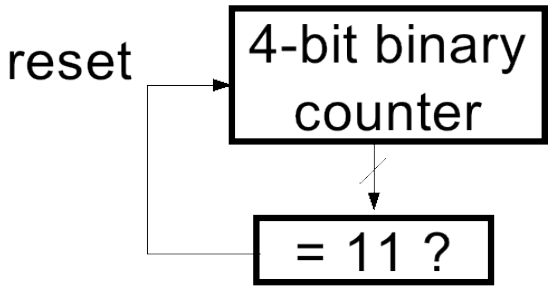


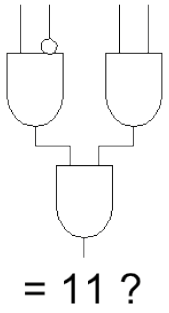
Toggle Flip-Flop

Fig. 6-13 4-Bit Up-Down Binary Counter

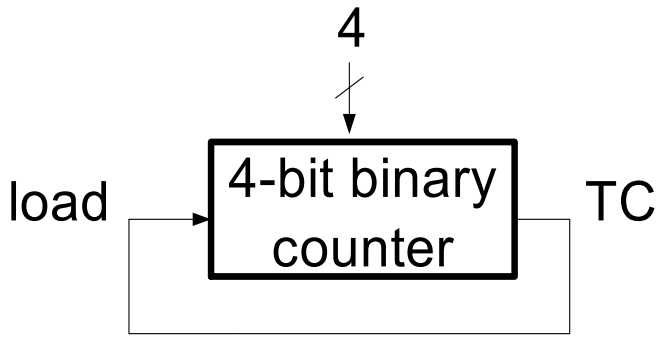
Odd Counts

- Extra combinational logic can be added to terminate count before max value is reached:
- Example: **count to 12**





- Alternative:

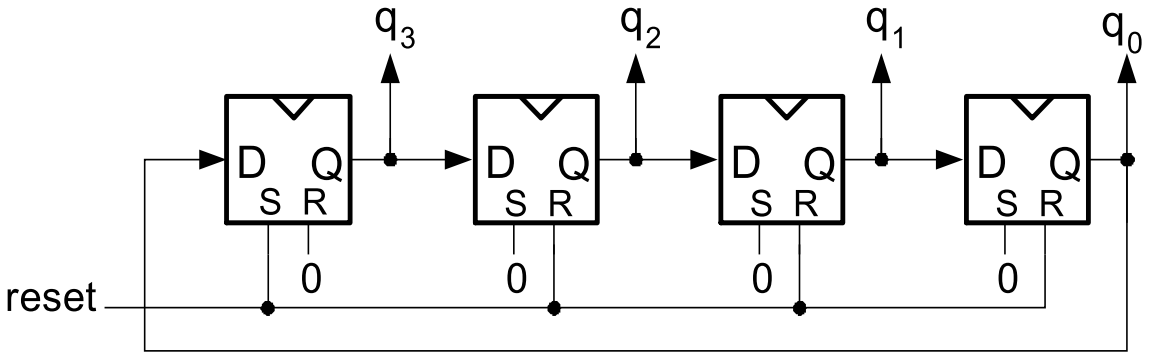


Ring Counters

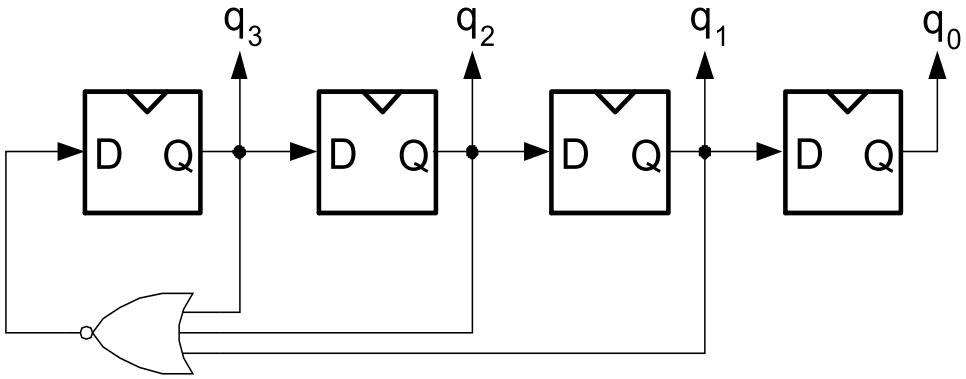
- "one-hot" counters

0001, 0010, 0100, 1000, 0001, …

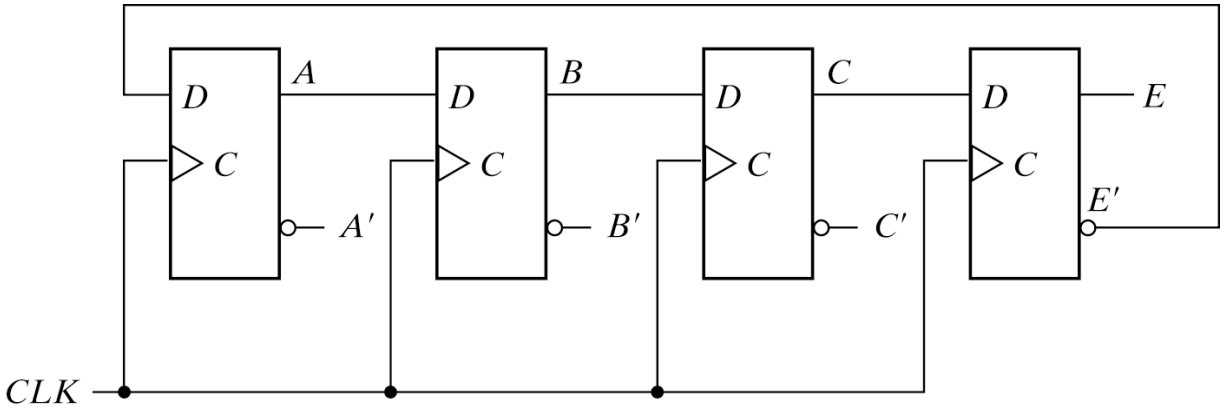
- What are these good for?



"Self-starting" version:



Johnson Counter



(a) Four-stage switch-tail ring counter

Sequence number	Flip-flop outputs				AND gate required for output
	A	B	C	E	
1	0	0	0	0	$A'E'$
2	1	0	0	0	AB'
3	1	1	0	0	BC'
4	1	1	1	0	CE'
5	1	1	1	1	AE
6	0	1	1	1	$A'B$
7	0	0	1	1	$B'C$
8	0	0	0	1	$C'E$

(b) Count sequence and required decoding

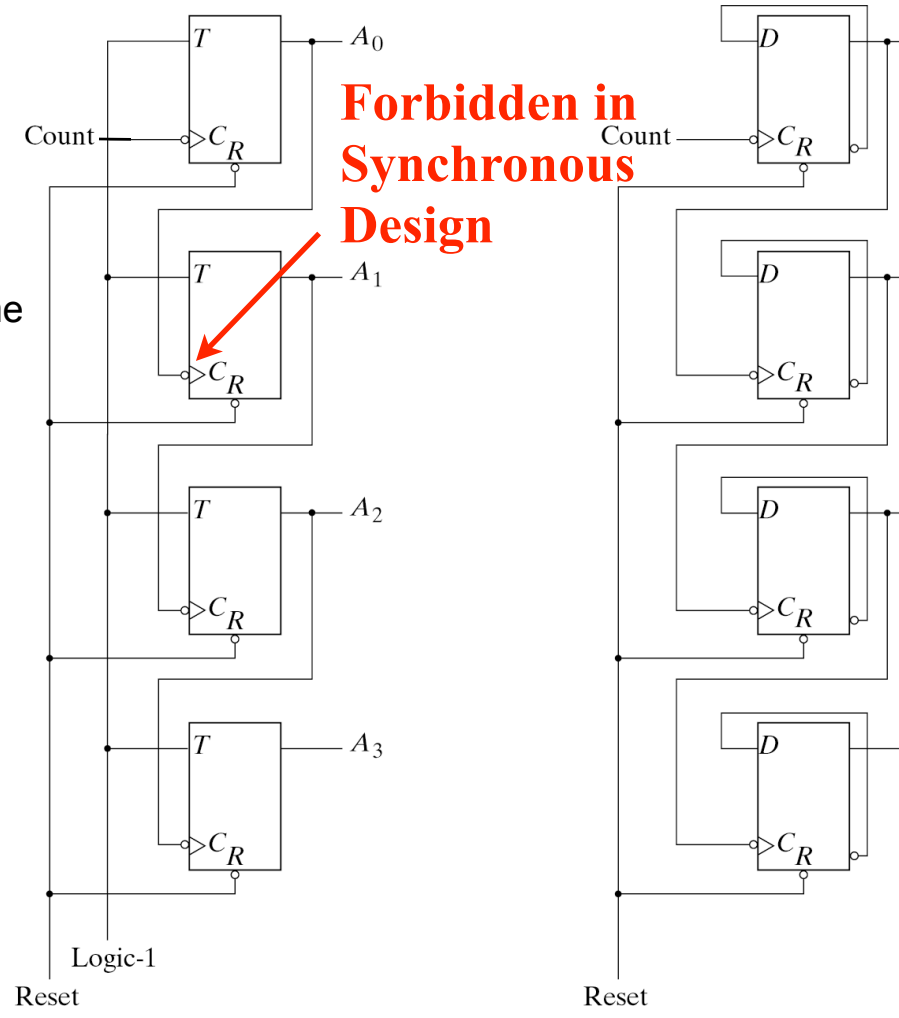
Fig. 6-18 Construction of a Johnson Counter

Asynchronous "Ripple" counters

$A_3\ A_2\ A_1\ A_0$

0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

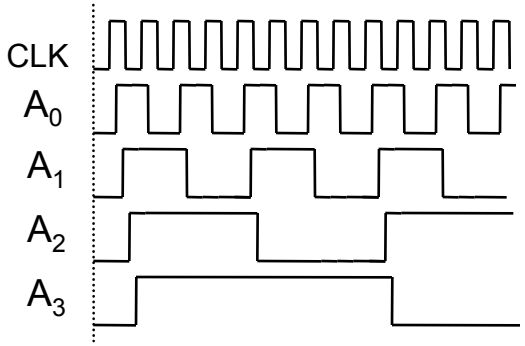
time



(a) With T flip-flops (b) With D flip-flops

Fig. 6-8 4-Bit Binary Ripple Counter

- Each stage is ÷2 of previous.
- Look at output waveforms:



- Often called "asynchronous" counters.
- A "T" flip-flop is a "toggle" flip-flop. Flips it state on cycles when T=1.