



University of
Massachusetts
Amherst

Engin112 – Lectures 33-34

Registers and Counters

Maciej Ciesielski
Department of Electrical and Computer Engineering
11/23-28/2011

Recap from last lecture

- What has been covered last
 - Finite state machines
 - » Design steps
 - » Analysis
 - » Simulation

- This lecture
 - Registers
 - » Parallel load register
 - » Shift register
 - » Universal shift register
 - Serial adder
 - Counters

Registers

- Computer systems use n -bit numbers
 - Too complex to worry about individual bits
 - Data is stored in “registers”
- Example: addition of registers
 - Designer-friendly abstraction: Register transfer level (RTL)
- What functions should a register support?
 - Store data (and states)
 - Getting data in
 - » In parallel
 - » In serial

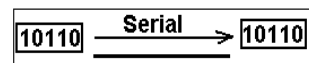
11/23-28/2011

Engin 112 - Intro to ECE

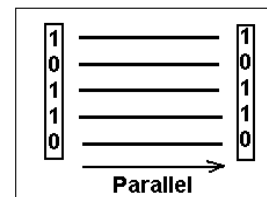
3

Parallel vs Serial Transfer

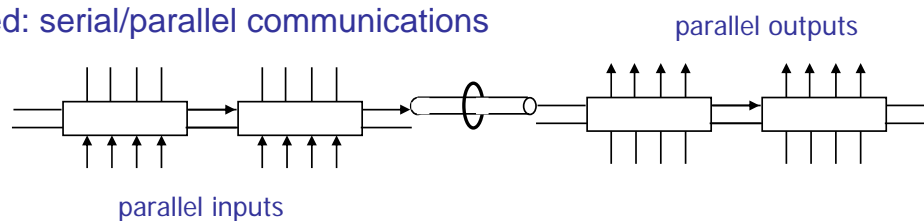
- Serial communications
 - Provides a binary number as a sequence of binary digits, one after another, through one data line.



- Parallel communications
 - Provides a binary number through multiple data lines at the same time.



- Mixed: serial/parallel communications



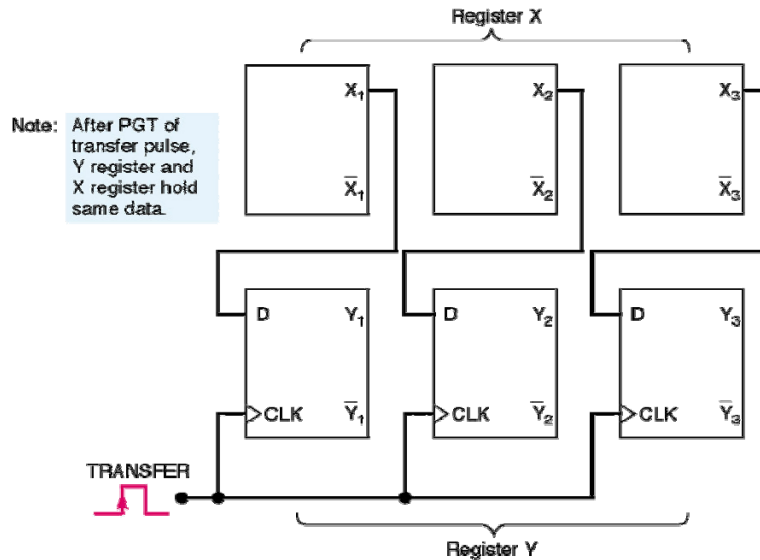
11/23-28/2011

Engin 112 - Intro to ECE

4

Parallel Transfer

- Parallel transfer from register X to register Y:



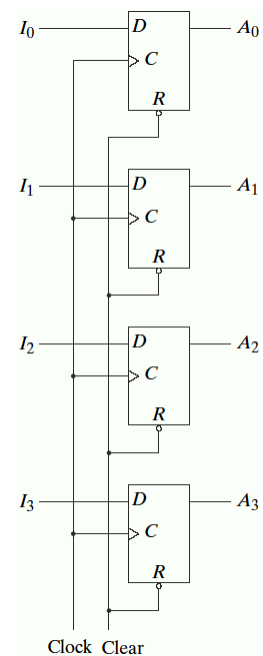
11/23-28/2011

Engin 112 - Intro to ECE

5

Simple Parallel Load Register

- Parallel load:
 - All n bits can be stored at the same time
- Straightforward circuit:
 - n D flip-flops
 - Shared clock
 - Shared reset
- What is the problem with this register?
 - How long is data stored?
- D flip-flops are set on each clock cycle
 - Undesired if we want to store data
 - Undesired if input requires time to stabilize
- Suggested improvement?



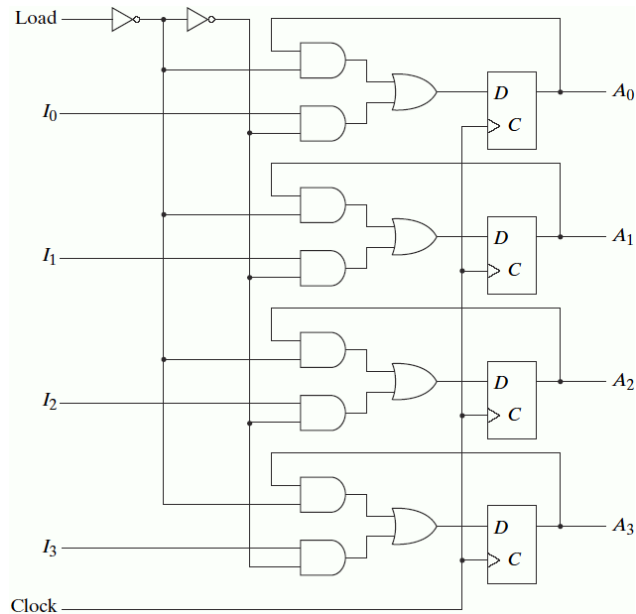
11/23-28/2011

Engin 112 - Intro to ECE

6

Parallel Load Register

- Register should only load when specified
- Design choices:
 - Intercept clock signal
 - » Not desirable, because of added clock delay
 - Feedback flip-flop value
- Circuit:
 - When loading, input is fed to flip-flop
 - When not loading, current state is “recycled”

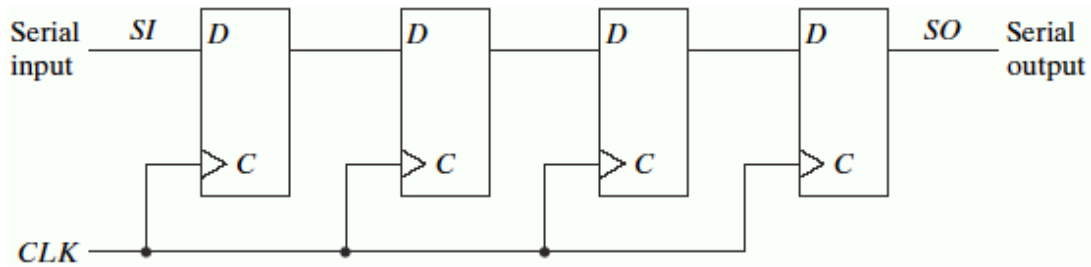


Serial Data Transfer

- What are the limitations of a parallel load register?
 - Requires n data lines
 - Can be expensive for 32-bit or 64-bit registers
- Serial data transfer common in digital systems
 - Transfer of information one bit at a time
- Examples:
 - Serial port on PC
 - USB (Universal Serial Bus)
 - Ethernet (and many other) network devices
 - Programmable and reconfigurable devices (FPGA, PLD)
 - Testing (scan chain), etc.
- Tradeoffs between serial vs. parallel?
 - Serial requires only single wire
 - Serial requires higher data rate to achieve same throughput

Shift Registers

- Serial transfer: load bits individually
 - How should we specify which bit is being loaded?
- Serial loading by shifting is simplest
 - Does not require any control bits
 - One bit per clock tick



11/23-28/2011

Engin 112 - Intro to ECE

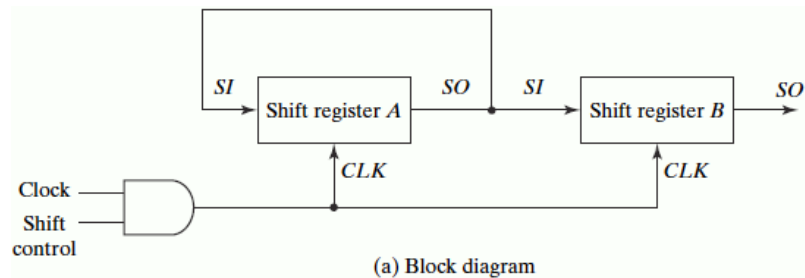
9

Serial Transfer - Example

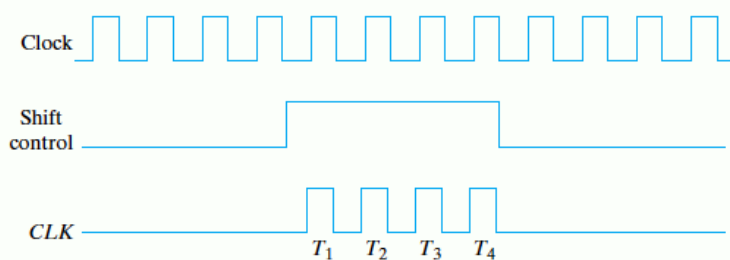
- Serial transfer from register A to register B:

- Example:

- Assume $A = 1011$ and $B = 0011$
- What are the register values at $T_1 - T_4$?



Time	Reg A	Reg B
T_0	1011	0011
T_1	1101	1001
T_2	1110	1100
T_3	0111	0110
T_4	1011	1011



11/23-28/2011

Engin 112 - Intro to ECE

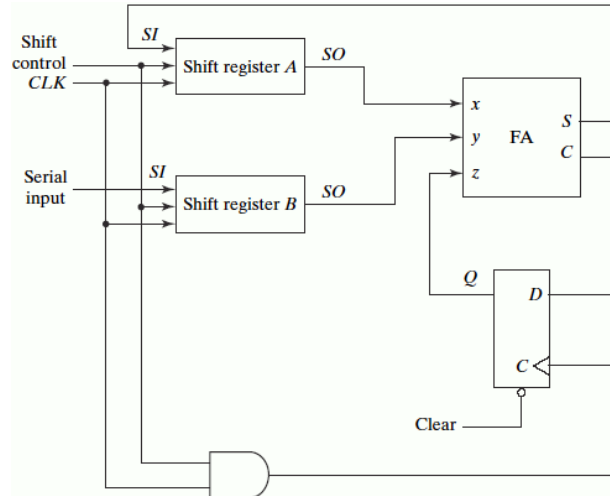
10

Serial Addition

- Addition is a “serial” process
 - Need to determine carry before next significant bit is added
 - We can use shift registers to add bits one-by-one
 - Only one-bit adder needed

» *Serial adder*

- Initial state:
 - Augend and addend in *A* and *B*
- Shift control
 - Controls (stops) addition
- How can we handle carry?
 - Feedback via D flip-flop
 - Carry is saved in DFF *Q*
- Where is the result?
 - Register *A* after *n* shifts

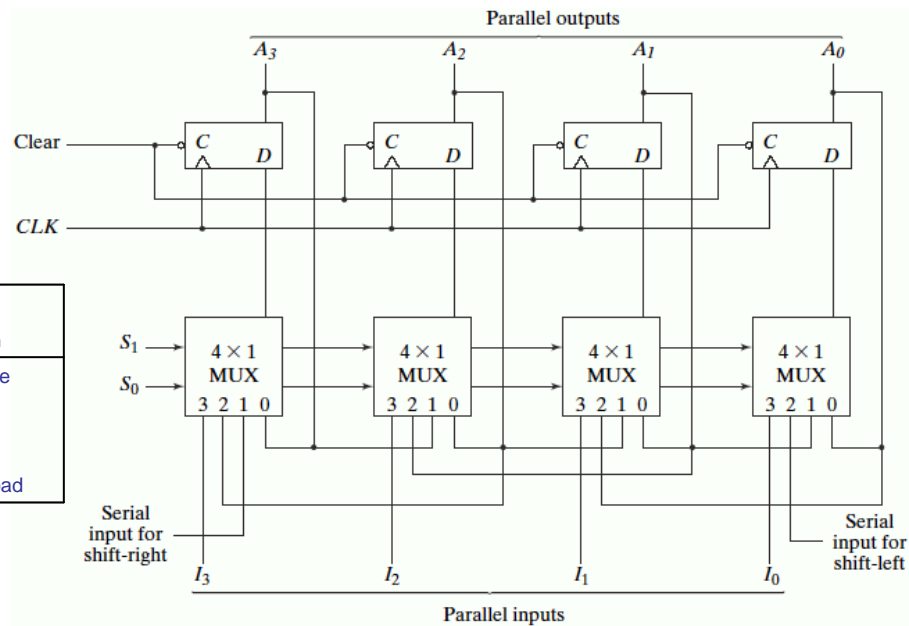


Universal Shift Register

- Shift registers are useful
 - We want to use it as a basic building block
- Universal Shift Register
 - Control functions
 - » CLEAR: reset register to 0
 - » CLOCK: synchronizes operation
 - » SHIFT-RIGHT: shifts right (incl. serial I/O)
 - » SHIFT-LEFT: shifts left (incl. serial I/O)
 - » PARALLEL-LOAD: parallel transfer of register value
 - » PARALLEL OUTPUT: *n* data lines
 - » NO OPERATION: state remains unchanged

Universal Shift Register

▪ Circuit:



▪ Control:

Mode		Control	Register
S1	S0		Operation
0	0		No change
0	1		Shift right
1	0		Shift left
1	1		Parallel load

Recap from last lecture

▪ Last lecture

- Registers
 - » Parallel load register
 - » Shift register
 - » Serial adder
 - How is it different from parallel adder?
 - Combinational or sequential?
 - » Universal shift register

▪ This lecture

- Binary counters
 - » Ripple counters (asynchronous)
 - » Synchronous counters (driven by common clock)
- Special counters
 - » Ring counters

Other Types of Flip-Flops

Other types of flip-flops used in certain designs (counters)

- JK flip-flop
 - » Similar to SR latch but with no undetermined state (all input combinations allowed)
- T flip-flop
 - » Output toggles
 - » can be built from D flip-flop or from JK flip-flop

11/23-28/2011

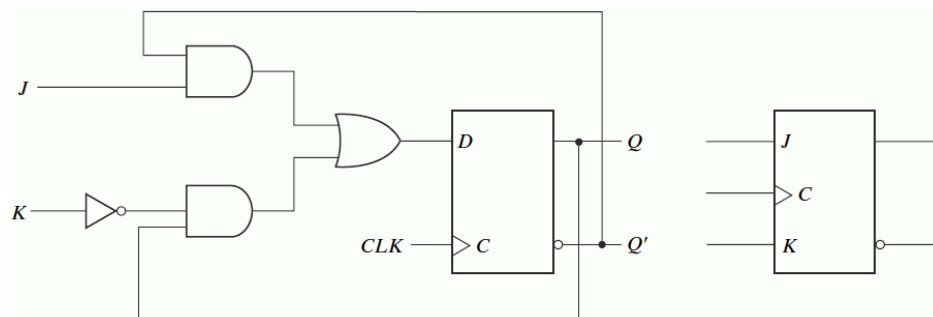
Engin 112 - Intro to ECE

15

JK Flip-Flop

- JK flip-flop:
 - $J=1$ sets Q to 1
 - $K=1$ sets Q to 0
 - $J = K=1$ complement Q
 - $J = K=0$ maintains Q
- JK flip-flop can be built from D flip-flop
 - $D = JQ' + K'Q$

J	K	D
0	0	Q
0	1	0
1	0	1
1	1	Q'



11/23-28/2011

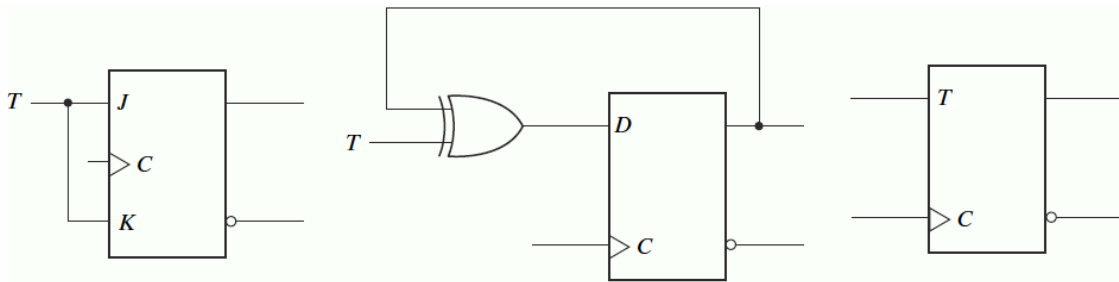
Engin 112 - Intro to ECE

16

T Flip-Flop

- Single input T
 - “Toggles” (complements) Q
- Can be built from D flip-flop:
 - $D = T \oplus Q = TQ' + T'Q$
- Can be built from JK flip-flop:
 - $J = K = T$

T	$Q(t+1)$
0	$Q(t)$
1	$Q'(t)$



11/23-28/2011

Engin 112 - Intro to ECE

17

Characteristic Tables and Equations

- Functionality of flip-flops must be specified
- Characteristic table
 - $Q(t)$ is current state/output at time t
 - $Q(t+1)$ is state/output at next clock period
- Characteristic equation: equation describing FF functionality
 - JK flip-flop
 - $Q(t+1) = JQ'(t) + K'Q(t)$
 - D flip-flop
 - $Q(t+1) = D$
 - T flip-flop
 - $Q(t+1) = Q'(t)$

J	K	$Q(t+1)$
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$Q'(t)$

D	$Q(t+1)$
0	0
1	1

T	$Q(t+1)$
0	$Q(t)$
1	$Q'(t)$

11/23-28/2011

Engin 112 - Intro to ECE

18

Counters

- A counter is a register that goes through a *predetermined* sequence of states upon application of input pulses
 - How is it then different from a generic FSM?
- Example: n -bit binary counter
 - Counts from 0 through 2^n-1
- Two flavors:
 - Ripple counters (asynchronous)
 - Synchronous counters
 - Difference:
 - » How flip-flops are triggered when propagating updates
 - » Who drives the *Clk* input to FF
- Counters can count *up* or *down*.

Binary Counter

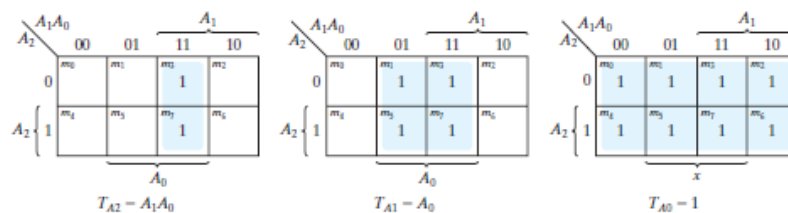
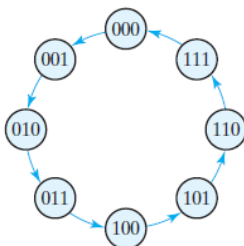
- An n -bit binary counter
 - Counts from 0 to 2^n-1
 - Has 2^n states (n flip-flops)
- State diagram
 - No input except Clk
 - Outputs derived from state FFs

Table 5.14
State Table for Three-Bit Counter

Present State			Next State			Flip-Flop Inputs		
A_2	A_1	A_0	A_2	A_1	A_0	T_{A2}	T_{A1}	T_{A0}
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	1	1
1	1	1	0	0	0	1	1	1

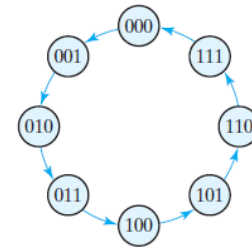
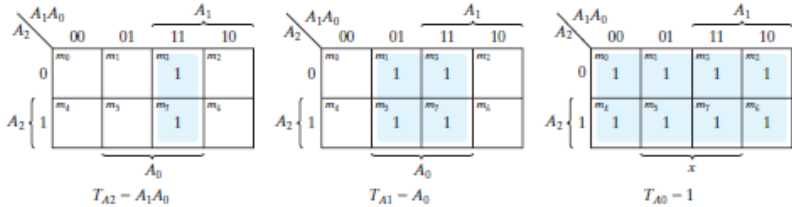
State table for implementation with T flip-flops

3-bit binary counter ($n=3$)

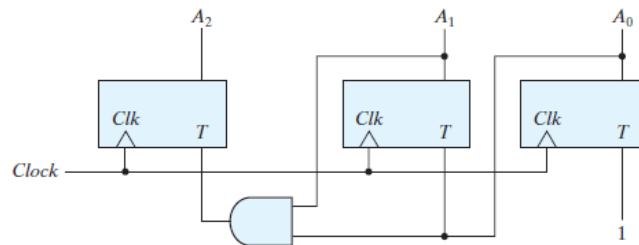


Binary Counter

- An n -bit binary counter
 - Implementation with T flip-flops

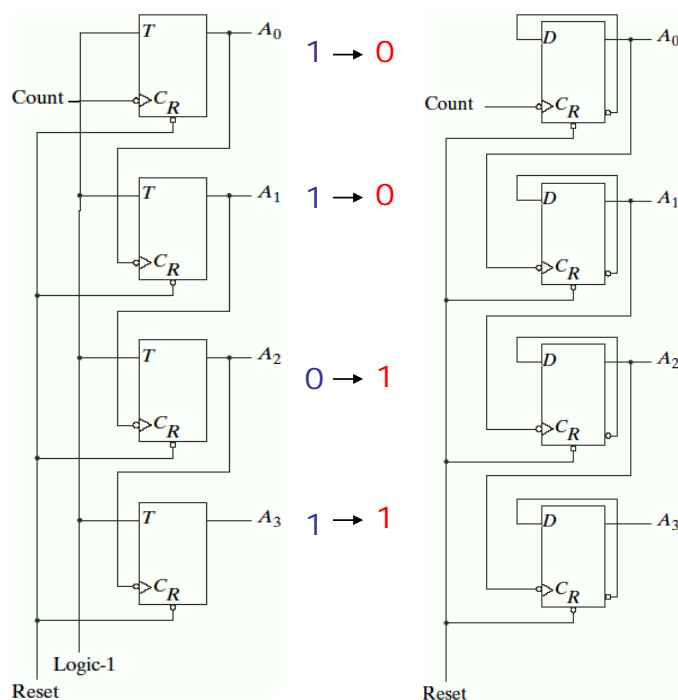


3-bit binary counter ($n=3$)



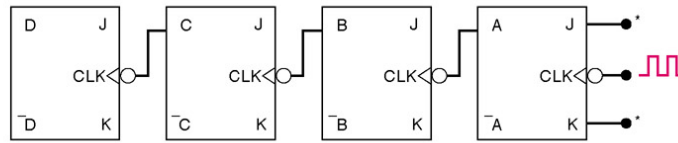
Ripple Counter

- Clock input driven by *Count* signal or by other flip-flops
- Circuit with T flip-flops
 - Count signal toggles A_0
 - Low-order flip flop provides trigger for adjacent flip flop
 - Are flip-flops positive or negative edge triggered?
 - Up-counting for negative edge
 - Down-counting ??
 - Not all fops change value simultaneously
 - Lower-order fops change first
- Example:
 - Current state: 1011
 - Count toggles A_0 , falling edge
 - Falling Edge A_0 toggles A_1
 - Falling Edge A_1 toggles A_2
 - Rising edge A_2 – no change on A_3
 - New state: 1100
- Similar with D flip-flops



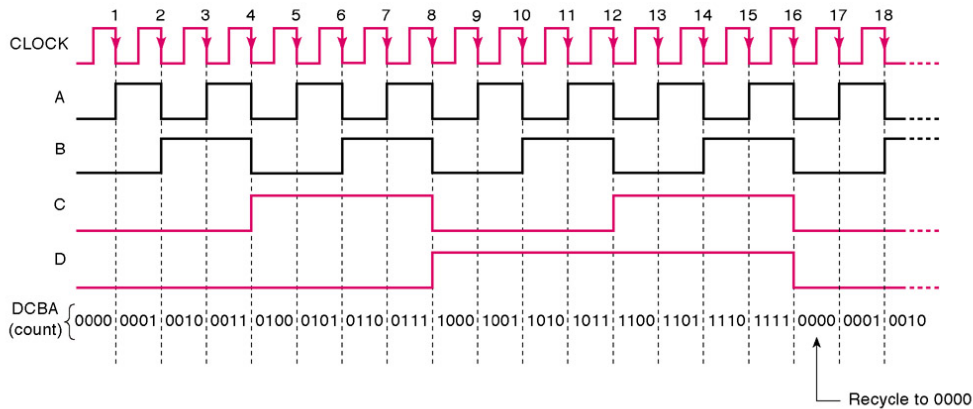
Another Asynchronous Ripple Counter

- Similar to T flop example on previous slide



*All J and K inputs assumed to be 1.

Recall: if $J = K = 1$, flip-flop toggles



Asynchronous (Ripple) Counters

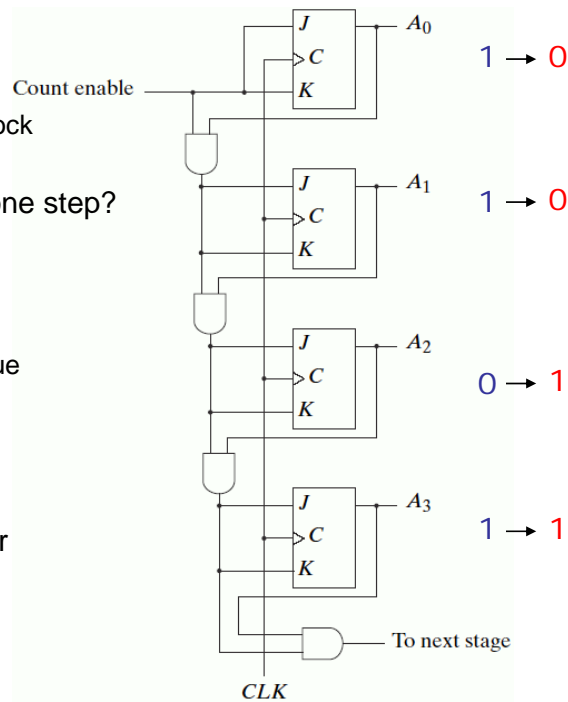
- Ripple counter:**
 - Carry propagates through bits
 - FFs respond one after another in a rippling effect
 - Each FF output drives the CLK input of the next FF.
- FFs do not change states in exact synchronism with the applied clock pulses
 - There is *delay* between the responses of successive FFs
 - Undesirable because of transition states
 - Example:
 - » Desired: 0011 → 0100 (*directly*)
 - » Real: 0011 → 0010 → 0000 → 0100
- Remedy: Synchronous counter

A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	0	1	0
0	0	0	0
0	1	0	0
0	0	0	1

delay ↓

Synchronous Counter

- **Synchronous (parallel) counter**
 - State switches with common clock
 - » All *Clk* input connected to common clock
 - All FFs change at same time
 - What is necessary to switch state in one step?
 - » Need to know which bits to toggle
 - » Very simple “carry look-ahead”
 - **Remember**
 - » If $J = K = 0$, flip-flop maintains old value
 - » If $J = K = 1$, flip-flop toggles
- **Circuit:**
 - AND gates test which bits will roll over
 - Polarity of clock edge is not important
 - » Do you know why ?



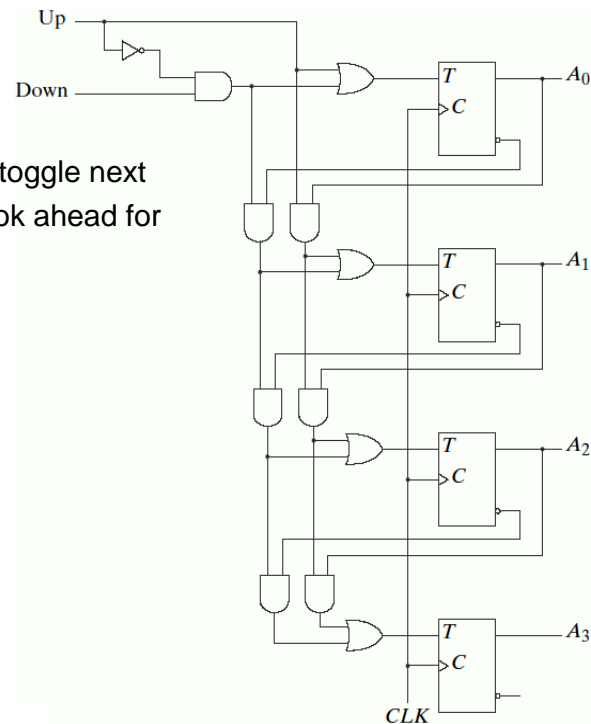
11/23-28/2011

Engin 112 - Intro to ECE

25

Up-down Counter

- **How can we count down?**
 - Least significant bit toggles most frequently
 - When transition from 0 to 1, then toggle next
 - Synchronous counter needs to look ahead for flip-flops that are 0
 - » Connect look-ahead to Q' output
- **Up-down counter circuit:**
 - If $Up = 1$, then increment
 - If $Down = 1$, then decrement
 - What if $Up = Down = 1$?



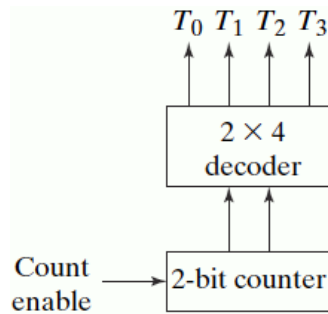
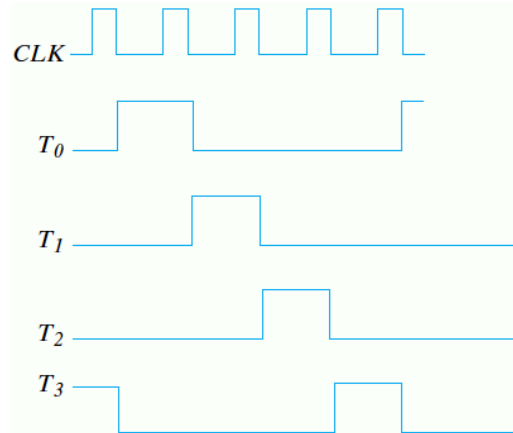
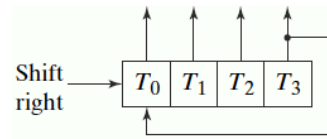
11/23-28/2011

Engin 112 - Intro to ECE

26

Ring Counter

- Shift register with one bit set to 1
 - Value shifts with every count cycle
- Timing diagram:
- Alternative implementation
 - Counter has only n states
 - $\lceil \log_2(n) \rceil$ bits should be enough
- Counter and decoder:



11/23-28/2011

Engin 112 - Intro to ECE

27

Reading Assignment

- Read Mano 6.1 – 6.6 (this lecture)
- Next: Mano 7.1-7.4 (Memory)

11/23-28/2011

Engin 112 - Intro to ECE

28