Counters and registers

Lecture 6 — § 2.4-2.6

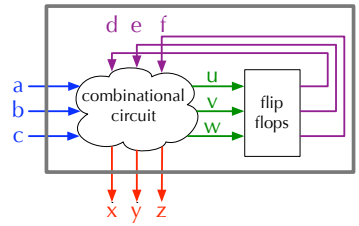
Computer Science 218

Mike Feeley

Key idea

sequential circuit is

- combination circuit plus memory (state)
- flip flops are its memory (one per bit)
- output is function of inputs **+ current state**



describe it using

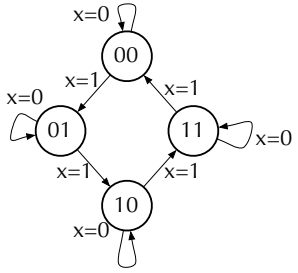
- truth table, boolean functions or logic diagram
  - for combinational part (truth table, boolean function, logic diagram)
  - inputs are external inputs plus flop-flop outputs
  - outputs are external outputs plus flip-flop inputs
  - one boolean function for each output in terms of all inputs
- finite state machine
  - for sequential part
  - finite state machine
  - lists all possible states (memory settings)
  - describes how current input and state determine external output and next state

u = *g*(a,b,c,d,e,f)
v = *h*(a,b,c,d,e,f)
w= *i*(a,b,c,d,e,f)
x = *j*(a,b,c,d,e,f)
y = *k*(a,b,c,d,e,f)
z = *l*(a,b,c,d,e,f)

Designing a sequential circuit

example: a two-bit counter

1. draw a state diagram
2. draw excitation table
   - left hand columns
     - all combinations of current state + input => next state
   - right hand columns
     - one flip flop for each state variable (e.g., A,B)
     - column for each flip-flop input, from excitation table



Q(t)		in	Q(t+1)		flip-flop inputs			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	x	0	x
0	0	1	0	1	0	x	1	x
0	1	0	0	1	0	x	x	0
0	1	1	1	0	1	x	x	1
1	0	0	1	0	x	0	0	x
1	0	1	1	1	x	0	1	x
1	1	0	1	1	x	0	x	0
1	1	1	0	0	x	1	x	1

Q(t)	Q(t+1)	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Designing a sequential circuit (II)

3. get boolean functions

- for
  - flip-flop inputs and external outputs
- in terms of
  - flip-flop outputs and external inputs
- using k-maps

Q(t)		in	Q(t+1)		flip-flop inputs			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	x	0	x
0	0	1	0	1	0	x	1	x
0	1	0	0	1	0	x	x	0
0	1	1	1	0	1	x	x	1
1	0	0	1	0	x	0	0	x
1	0	1	1	1	x	0	1	x
1	1	0	1	1	x	0	x	0
1	1	1	0	0	x	1	x	1

Bx / A	00	01	11	10
0	0	0	1	0
1	x	x	x	x

$J_A = Bx$

Bx / A	00	01	11	10
0	0	1	x	x
1	0	1	x	x

$J_B = x$

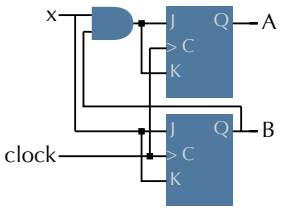
Bx / A	00	01	11	10
0	x	x	x	x
1	0	0	1	0

$K_A = Bx$

Bx / A	00	01	11	10
0	x	x	1	0
1	x	x	1	0

$K_B = x$

4. draw logic diagram



Now you try ...

implement a 2-bit grey-code counter

- an encoding where successive numbers differ by exactly one bit
- here is a three-bit grey code sequence starting at zero

000
001
011
010
110
111
101
100

draw the

- state diagram
- excitation table
- boolean functions
- circuit diagram

here is the excitation table for a JK flop flop

Q(t)	Q(t+1)	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Basic register

stores a multi-bit data word

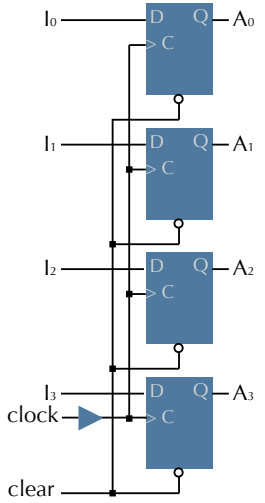
- essentially a group of flip flops
- one flip flop for each bit

on each clock rising/falling edge

- value of I is captured and available as A
- if clear → 0, flip flops reset A to 0

limitations?

- only holds A for one clock cycle



Parallel-load register

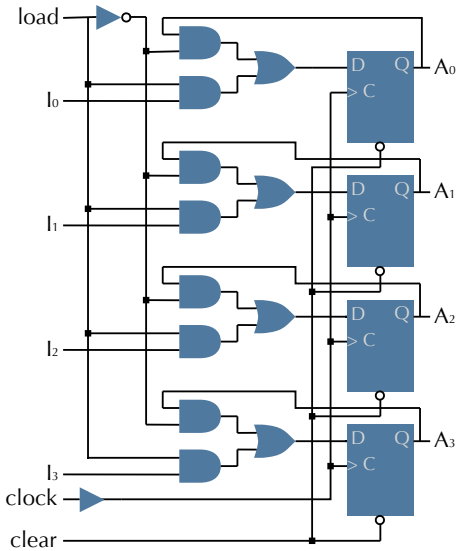
D flip flop makes it tricky

- unlike JK, it doesn't hold value
- its gone on next clock

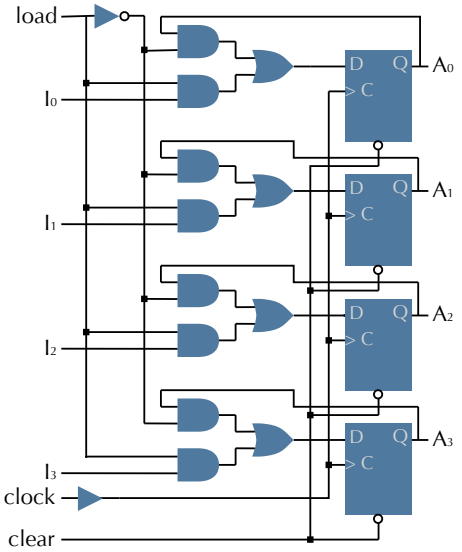
add *load* input

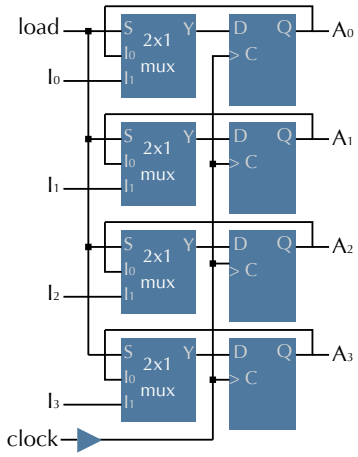
- if 0, A doesn't change
  - feed A back into FF input
- if 1, A=I
  - fin I into FF input

can you make it simpler?



Parallel-load register (II)





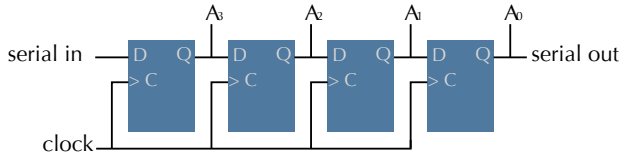
Shift register

register that can shift value to left or right

uses

- serialize data for "long-distance" transmission
  - *n*-bit data transmitted one bit at a time in *n* cycles
- multiply or divide an integer by two

simple unidirectional shift register



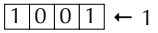
what's missing?

Bidirectional shift register

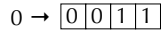
two-bit mode input

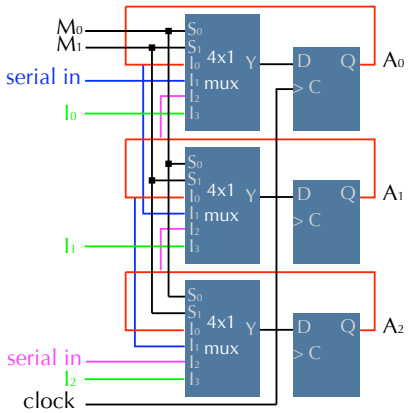
- 00 – no change
- 01 – shift left
- 10 – shift right
- 11 – parallel load (A=I)

shift left



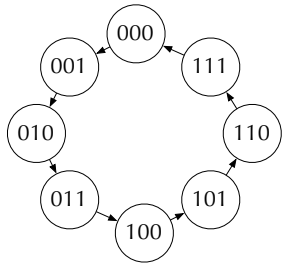
shift right





Binary n-bit counter ...

we could draw state machine any value of *n*



- but this requires new design for each value of *n*
- is there a generic design that makes changing *n* easy/incremental?

design a state machine for each bit

- then compose them
- what is the rule for switching a bit?
  - based on lower-order bits

switch when lower-order bits are all 1

draw the state diagram for one bit ...

One-bit counter

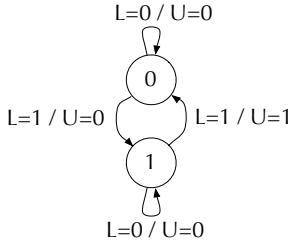
one-bit-couter(L) ⇒ (A,U)

- L = conjunction of all lower-order bits
  - L=1 iff all lower-order bits are one
- U = conjunction of this bit and L
  - input to L of next bit

d flip flop excitation table

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

state diagram, excitation table, function and logic



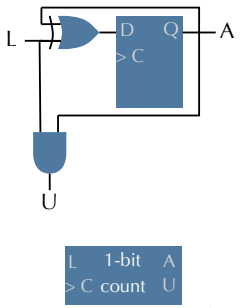
cur state A	in L	next state A	out U	FF in Di
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	0	1	0

Di

A \ L	0	1
0	0	1
1	1	0

Di = A ⊕ L

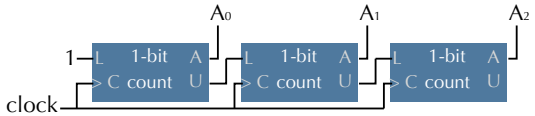
U = AL



Now a 3-bit counter

using

we get

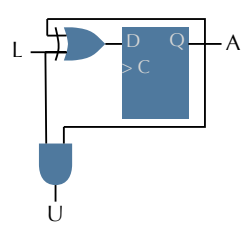


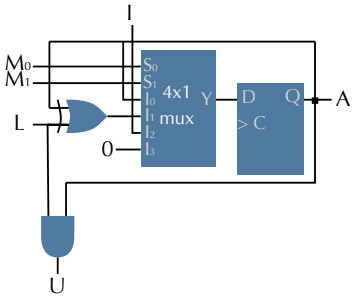
N-bit counter with parallel load

mode input

- 00 – no change
- 01 – increment
- 10 – load
- 11 – clear

add mux to one-bit counter





3-bit counter with parallel load