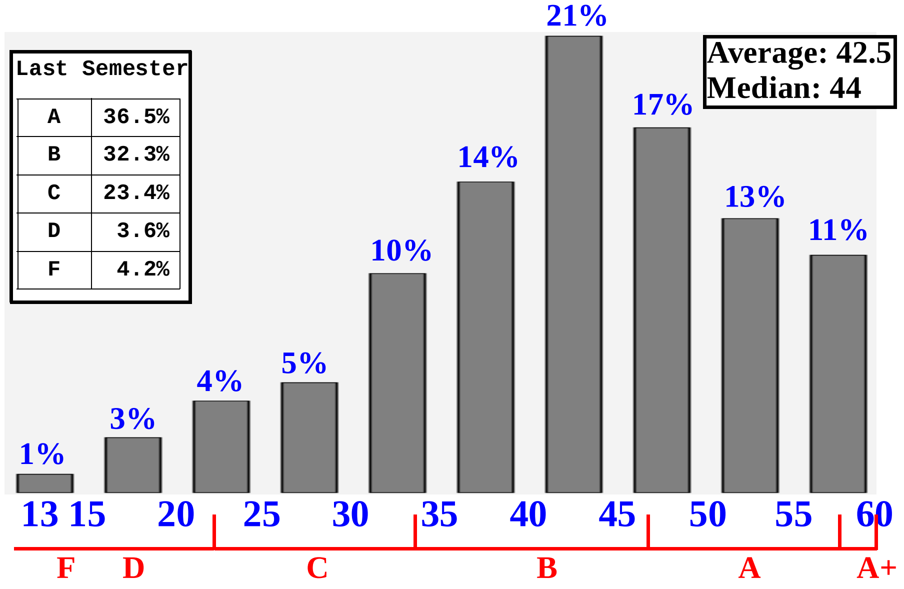
CS 126 Lecture A4: Sequential Circuits

Midterm Statistics

Last Semester	
A	36.5%
B	32.3%
C	23.4%
D	3.6%
F	4.2%

Average: 42.5
Median: 44



Outline

- **Introduction**
- An S-R Flip-flop
- More flip-flops
- Registers and register files
- Counters
- Conclusions

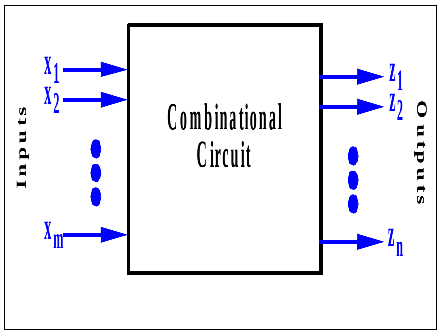
Where We Are At

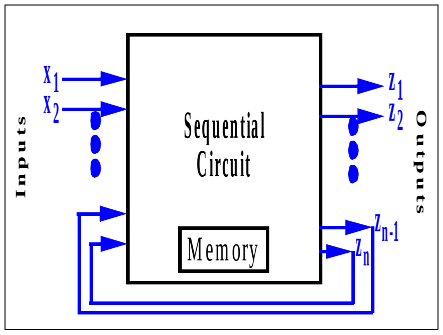
- We have learned the abstract **interface** presented by a machine: the instruction set architecture
- What we are learning: the **implementation** behind the interface:
  - ~~Start with switching devices (such as transistors)~~
  - ~~Build logic gates with transistors~~
  - ~~Build combinational circuit (memory-less) devices using gates~~
  - **Today: build sequential circuit (memory) devices**
  - Thursday: glue these devices into a computer

Memory-less Devices vs. Devices with Memory

- What we we have learned in the last lecture
  - Devices that can carry out one step of operation
- What they can't do
  - "Remember" history of operations
  - Carry out a sequence of operations in which later operations depend on results of previous ones

Combinational vs. Sequential Circuits



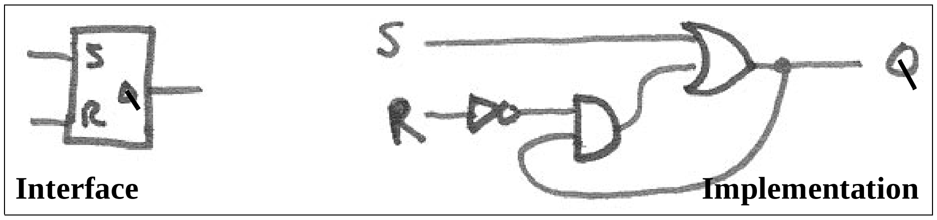


- Combinational circuits
  - Outputs determined solely by inputs
- Sequential Circuits
  - Characterized by feedbacks
  - Outputs determined by inputs and previous outputs

Outline

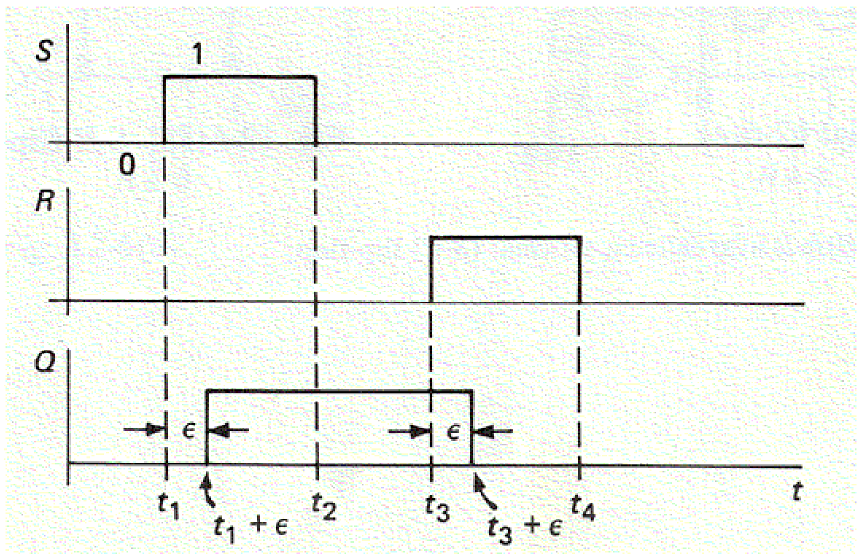
- ~~Introduction~~
- **An S-R Flip-flop**
- More flip-flops
- Registers and register files
- Counters
- Conclusions

Set-Reset Flip-flop



- A flip-flop
  - A smallest sequential circuit
  - Can "remember" a bit of information
- An S-R flip-flop
  - Pulse on Set (S) line turns flip-flop on
  - Pulse on Reset (R) line turns flip-flop off
  - If S=R=0, nothing happens
  - S=R=1 not allowed

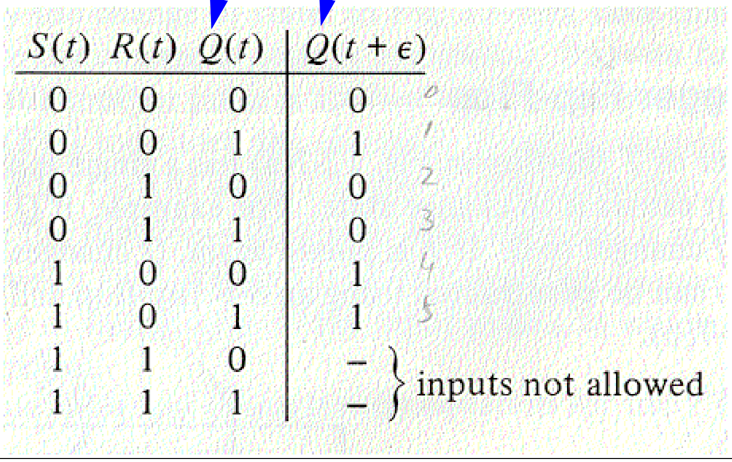
Timing Diagram (for S-R Flip-flop)



- Because sequential circuits are functions of time, a timing diagram is one of the ways of describing them

Truth Table (for S-R Flip-flop)

previous state next state



$S(t)$	$R(t)$	$Q(t)$	$Q(t+\epsilon)$	
0	0	0	0	0
0	0	1	1	1
0	1	0	0	2
0	1	1	0	3
1	0	0	1	4
1	0	1	1	5
1	1	0	–	inputs not allowed
1	1	1	–	

- Previous states become “input variables” in truth table

Characteristic Equation (for S-R Flip-flop)

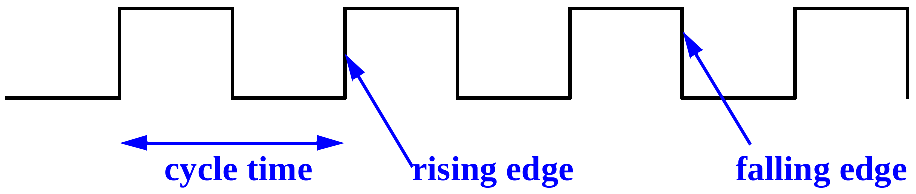
$$Q^{+} = S + R'Q \qquad (SR=0)$$

- An equation that expresses the next state of a flip-flop in terms of its present state and inputs (also called next state equations)
- Timing diagrams, truth tables, and next-state equations are important tools for understanding and constructing more sophisticated sequential circuits as well

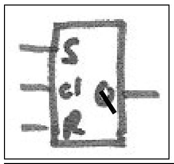
Outline

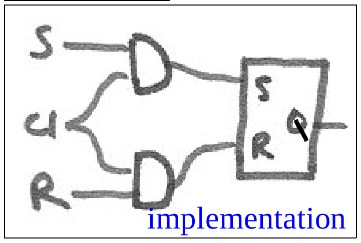
- ~~Introduction~~
- ~~An S-R Flip-flop~~
- **More flip-flops**
- Registers and memory
- Counters
- Conclusions

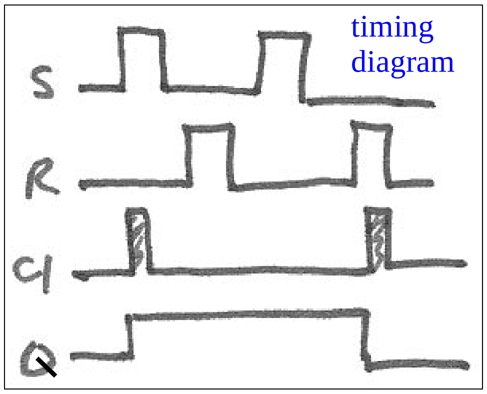
The Clock



A Clocked S-R Flip-flop

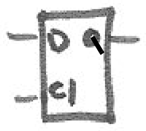




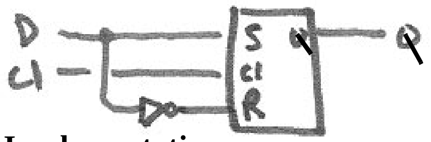


- In large sequential networks, there are many flip-flops
- Need to synchronize operations of different flip-flops
- Synchronization provided by a a common clock (pulse)

A D Flip-flop



Interface



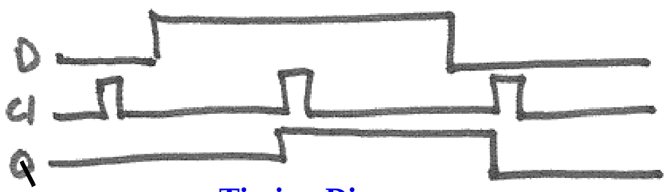
Implementation

On clock pulse

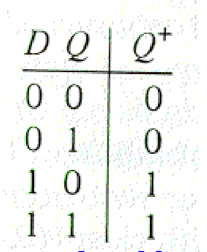
D = 1 set

D = 0 reset

Behavior of D Flip-flop



Timing Diagram



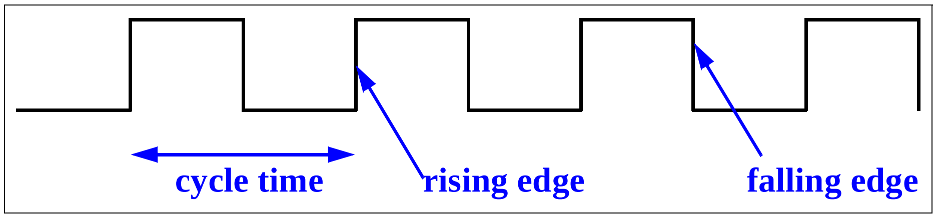
D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

Truth Table

$Q^+ = D$

Characteristic Equation

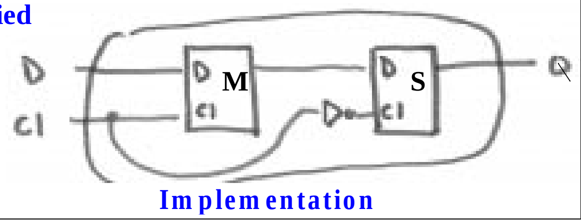
Rising vs. Falling Edge

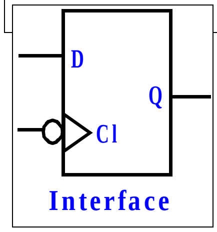


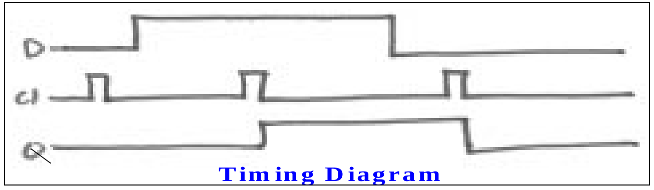
- So far, all the clocked flip-flops "flip-flop" on the rising edge of a clock signal
- When we cram a lot of actions into a single cycle, we sometimes need them to change state on the falling edge

Master-Slave D Flip-flop

On rising edge, input copied into master;
On falling edge, master copies data into slave.





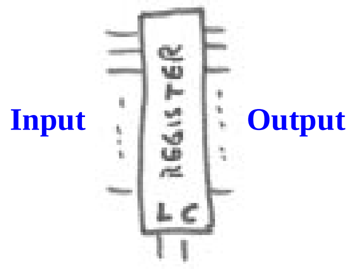


- Input sampled on rising edge, and must remain stable during the pulse, output changes on falling edge
- Question: why don't we just invert the clock using a NOT?
- Another type: "edge-triggered", allows input change during clock pulse

Outline

- ~~Introduction~~
- ~~An S-R Flip-flop~~
- ~~More flip-flops~~
- **Registers and register files**
- Counters
- Conclusions

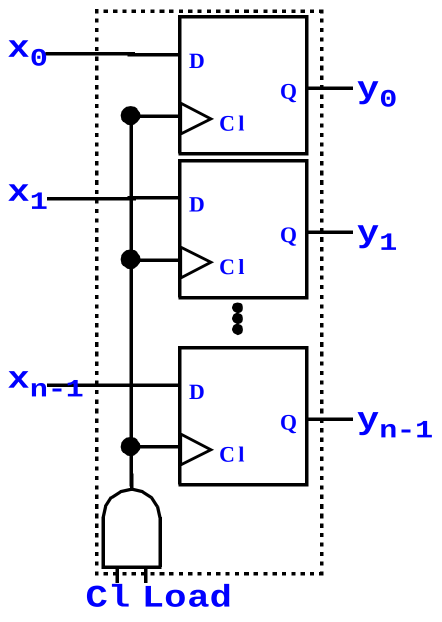
Stand-alone Register Interface



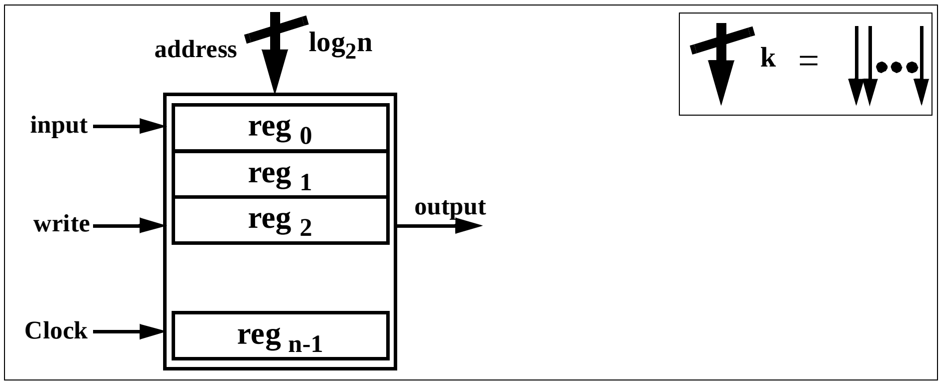
LOAD line (and clock)
controls when new values are loaded

Use registers for
- PC
- memory addresses
- memory buffer
- arithmetic

Stand-alone Register Implementation

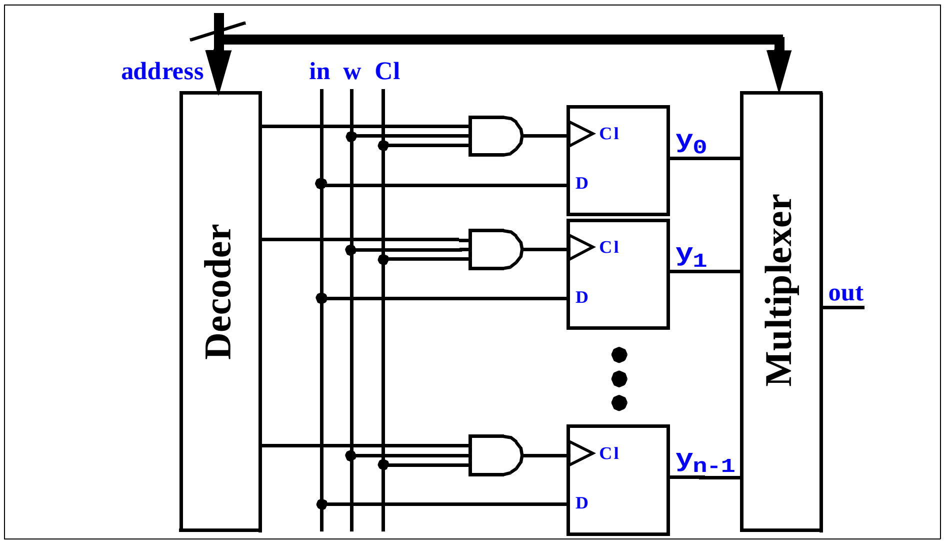


Register File Interface (Bits)



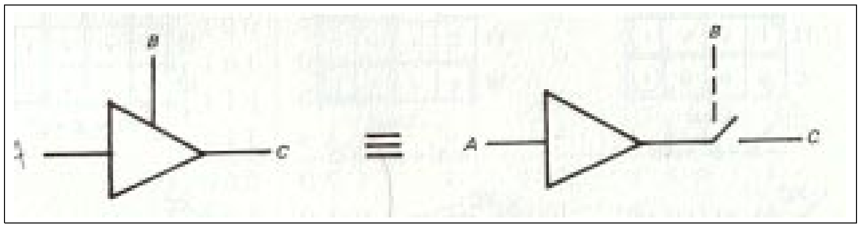
- bunch of bits to choose from
- “address” specifies which bit
- if “write” is 1, “input” gets copied into the chosen bit on clock pulse
- if “write” is 0, chosen bit appears on “output”

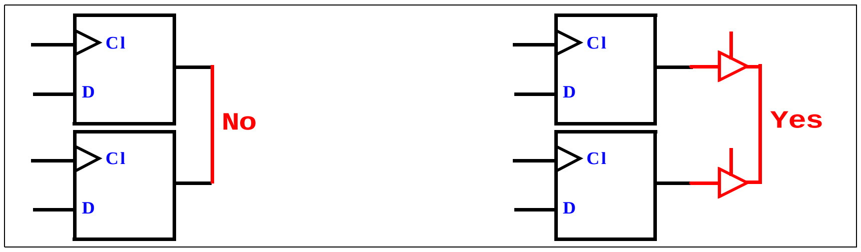
Register File Implementation (Bits)



- Decoder chooses exactly one bit to write into
- Multiplexer chooses exactly one bit to copy out

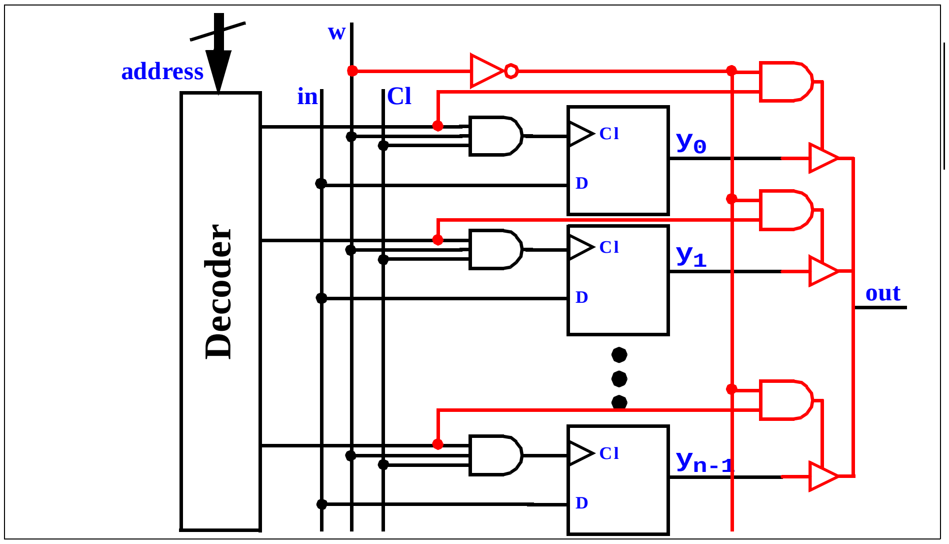
3-State Logic





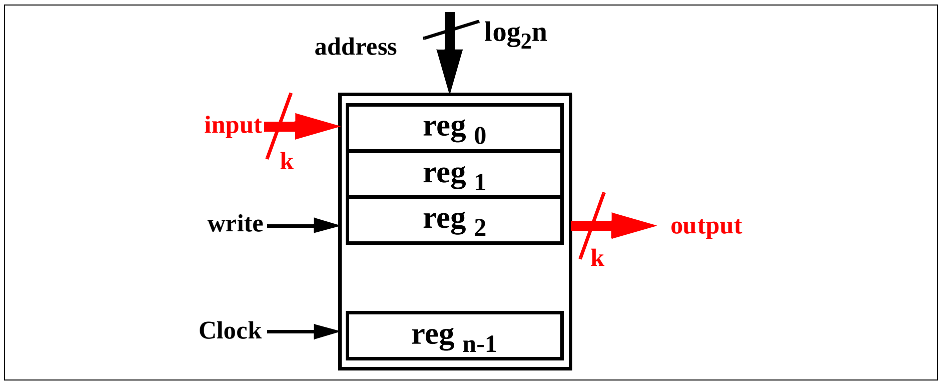
- Can’t connect outputs together (even if they are zero)
- Must use multiplexer (or its equivalent: **[3-state logic]**)

Register File Implementation 2 (Bits)



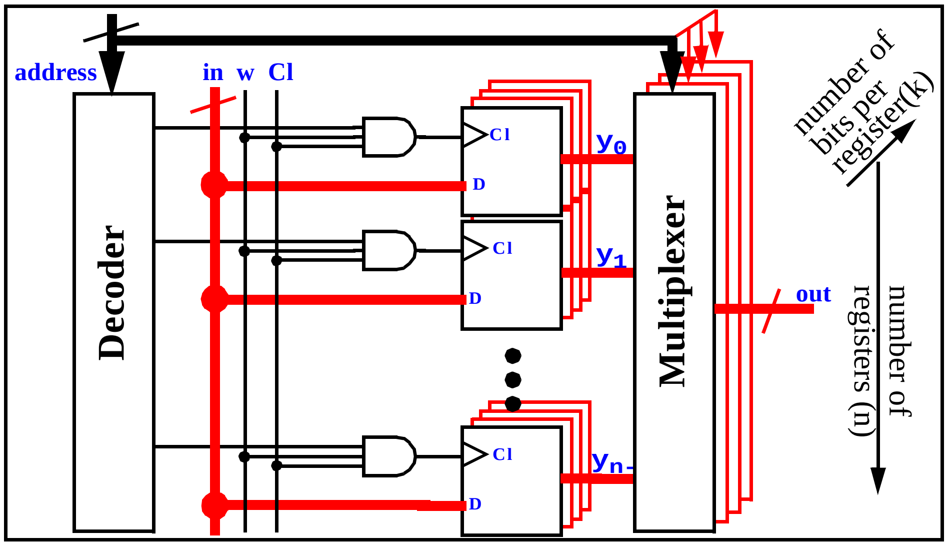
- Red things are new: replace MUX with 3-state logic
- Less Complex than MUX version

Register File Interface (Words)



- Register file of k-bit words
- red things show the differences between word case and bit case

Register File Implementation (Words)

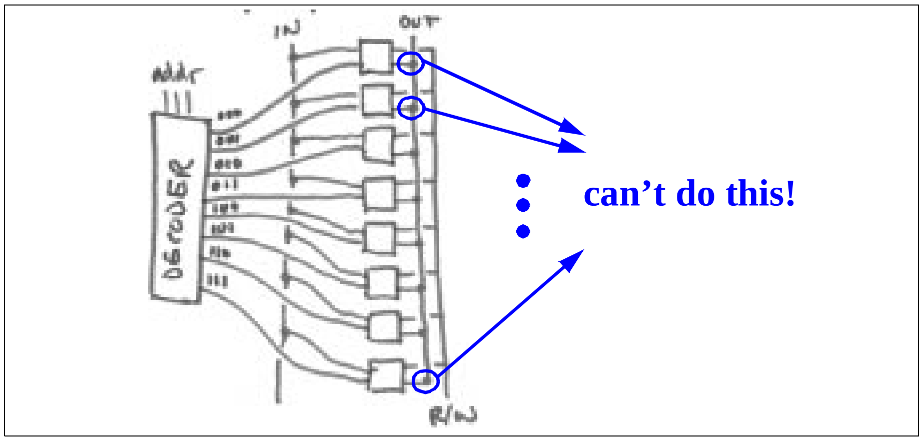


- red things show the differences between word case and bit case
- Multiply the number of flip-flops and MUXes by bits per register (k)
- May replace MUXes with 3-state logic (see previous slides)

Correting Lecture Notes in Your Course Reader

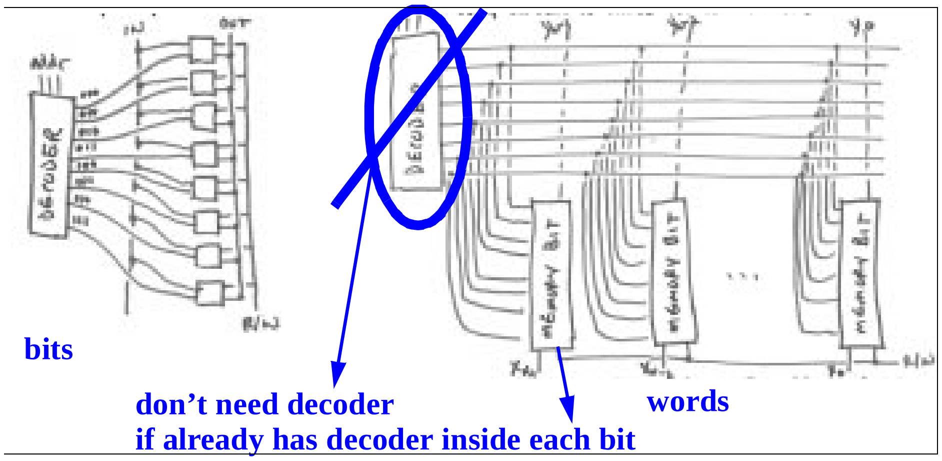
- Memory vs. register files
  - Lecture notes use the term “memory”
  - Meant to say register files (or SRAM)
  - DRAM made differently--no flip-flops
  - DRAM: one transistor per bit!
  - Much higher density than flip-flops, but slower

Correting Lecture Notes in Your Course Reader (cont.)



- Can't connect outputs together (even if they are zero)
- Must use multiplexer (or its equivalent: **[3-state logic]**)

Correting Lecture Notes in Your Course Reader (cont.)

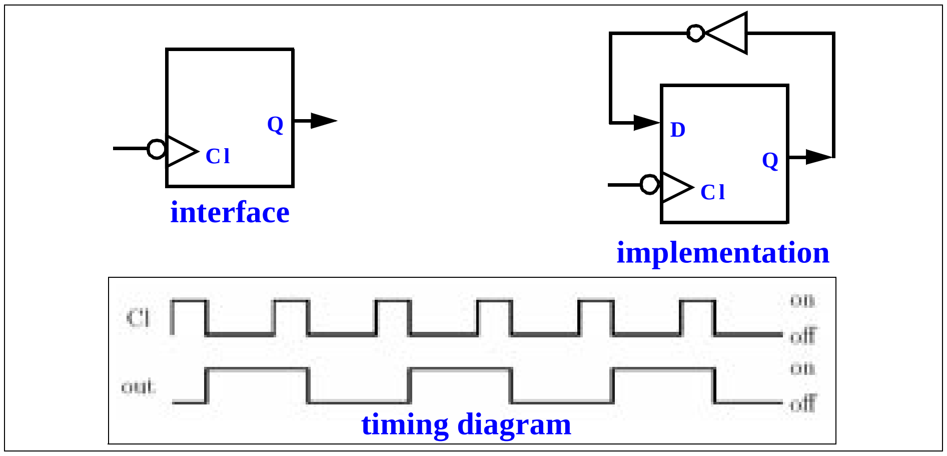


- Don't need decoder
- But even if you remove it, still not quite right for TOY register file: no need to replicate decoders for each bit

Outline

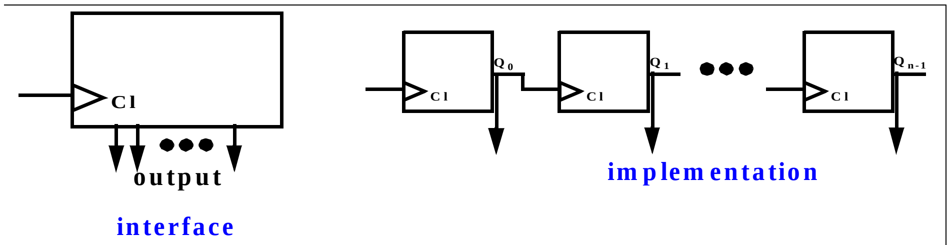
- ~~Introduction~~
- ~~An S-R Flip-flop~~
- ~~More flip-flops~~
- ~~Registers and register files~~
- Counters
- Conclusions

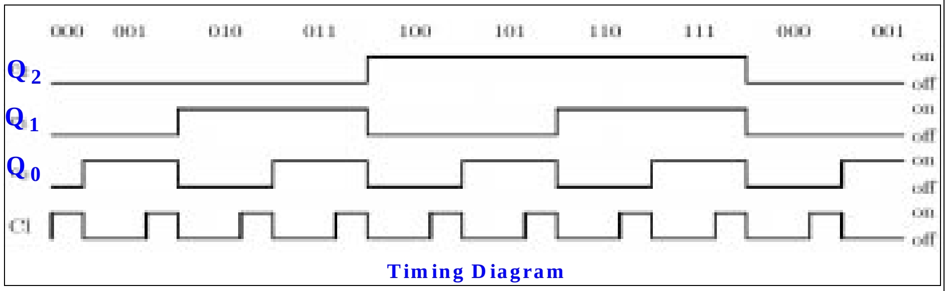
1-Bit Counter



- The behavior of a 1-bit binary counter is a clock whose cycle is twice as long as the input clock

N-bit Counter



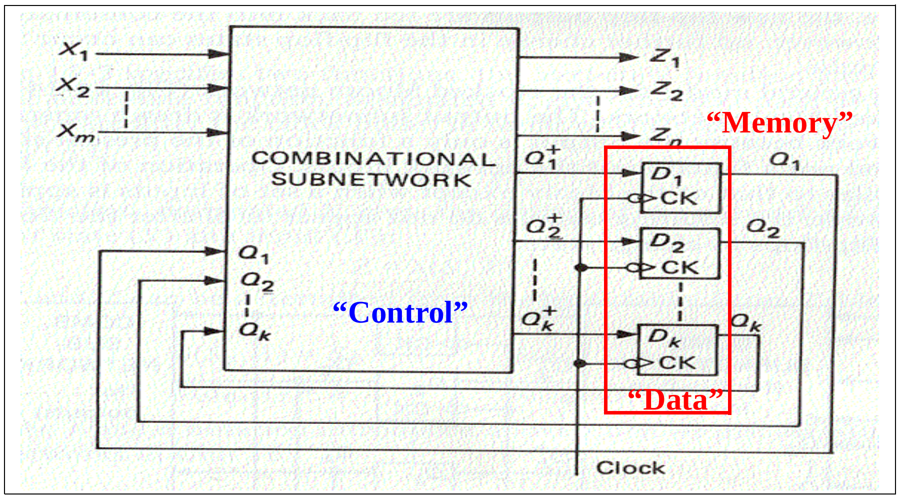


- n-bit counter: chaining n 1-bit counters together
- Recursive! An n-bit counter is made by gluing one extra bit to an (n-1) bit counter

Outline

- ~~Introduction~~
- ~~An S-R Flip-flop~~
- ~~More flip-flops~~
- ~~Registers and register files~~
- ~~Counters~~
- Conclusions

High-Level View of Computer



- Computer: "memory" state with feedback, clocked
- Each clock enables changes in memory state
- Combinational logic (topic of last lecture) employed to specify what changes to make in response to inputs and past history

What We Have Learned Today

- Flip-flops ([S-R, D], [unclocked, clocked, master-slave, edge triggered])
  - Their behavior (timing diagrams, truth tables, characteristic equations)
  - How they are made
- Some sequential devices (registers, register files, counters)
  - Their behavior
  - How they are made