inst.eecs.berkeley.edu/~cs61c

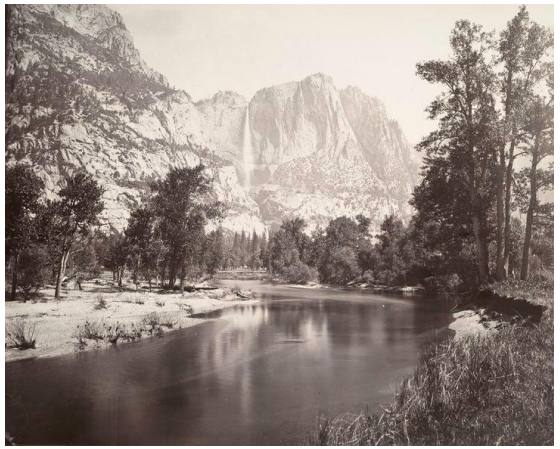
CS61C : Machine Structures

Lecture 20 – Introduction to Synchronous Digital Systems

Lecturer PSOE Dan Garcia

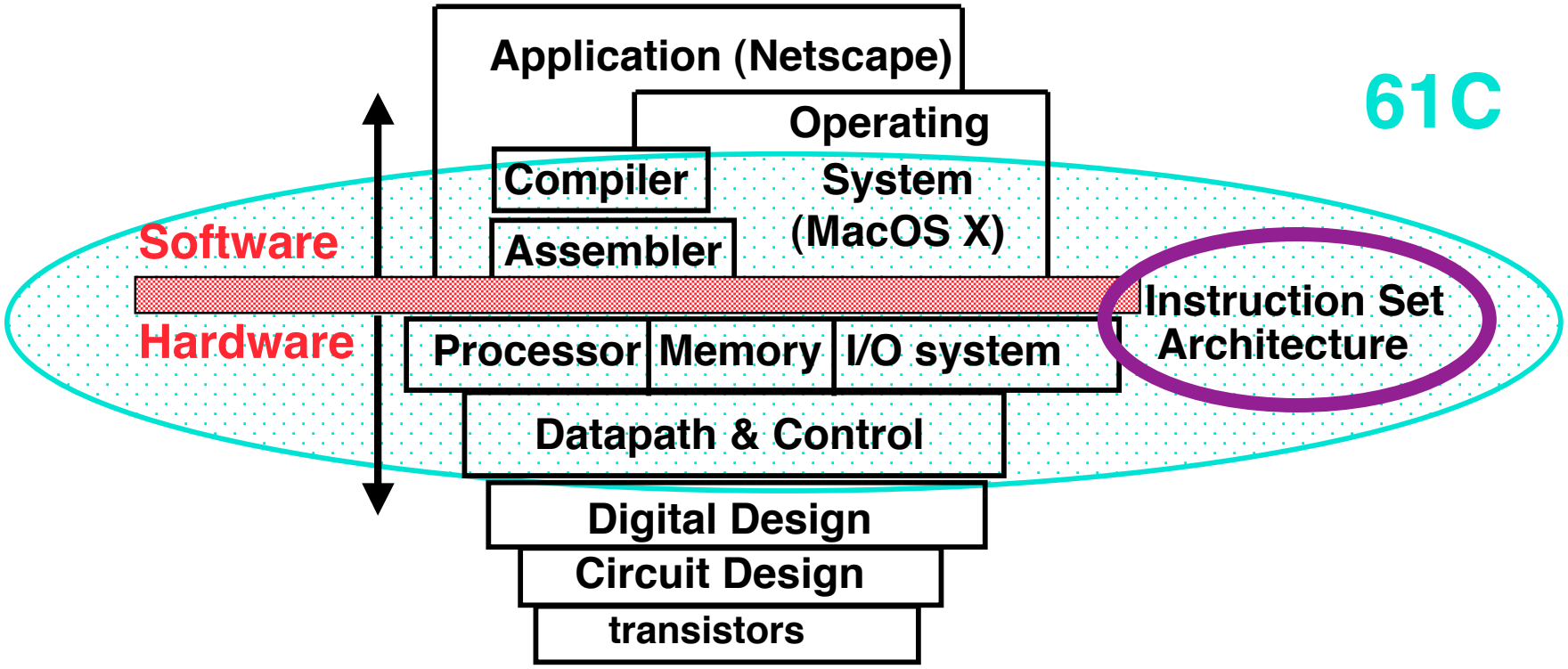
www.cs.berkeley.edu/~ddgarcia

NY Public Library ⇒ They've digitized their collection and put 275,000 images online! Manuscripts, historical maps, vintage posters, rare prints & photos, illustrated books & more!



www.nypl.org

What are "Machine Structures"?



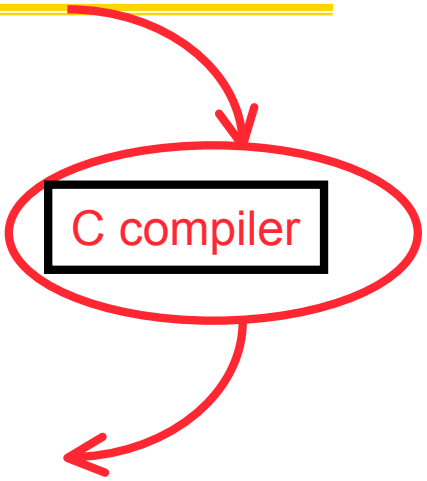
Coordination of many *levels of abstraction*

We'll investigate lower abstraction layers! (contract between HW & SW)

Below the Program

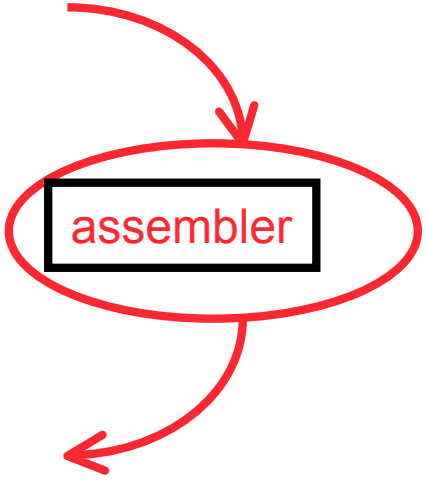
- **High-level language program (in C)**

```
swap   int v[], int k){
       int temp;
       temp = v[k];
       v[k] = v[k+1];
       v[k+1] = temp;
}
```



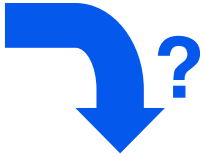
- **Assembly language program (for MIPS)**

```
swap: sll   $2, $5, 2
      add   $2, $4,$2
      lw    $15, 0($2)
      lw    $16, 4($2)
      sw    $16, 0($2)
      sw    $15, 4($2)
      jr    $31
```



- **Machine (object) code (for MIPS)**

```
000000 00000 00101 0001000010000000
000000 00100 00010 0001000000100000 . . .
```



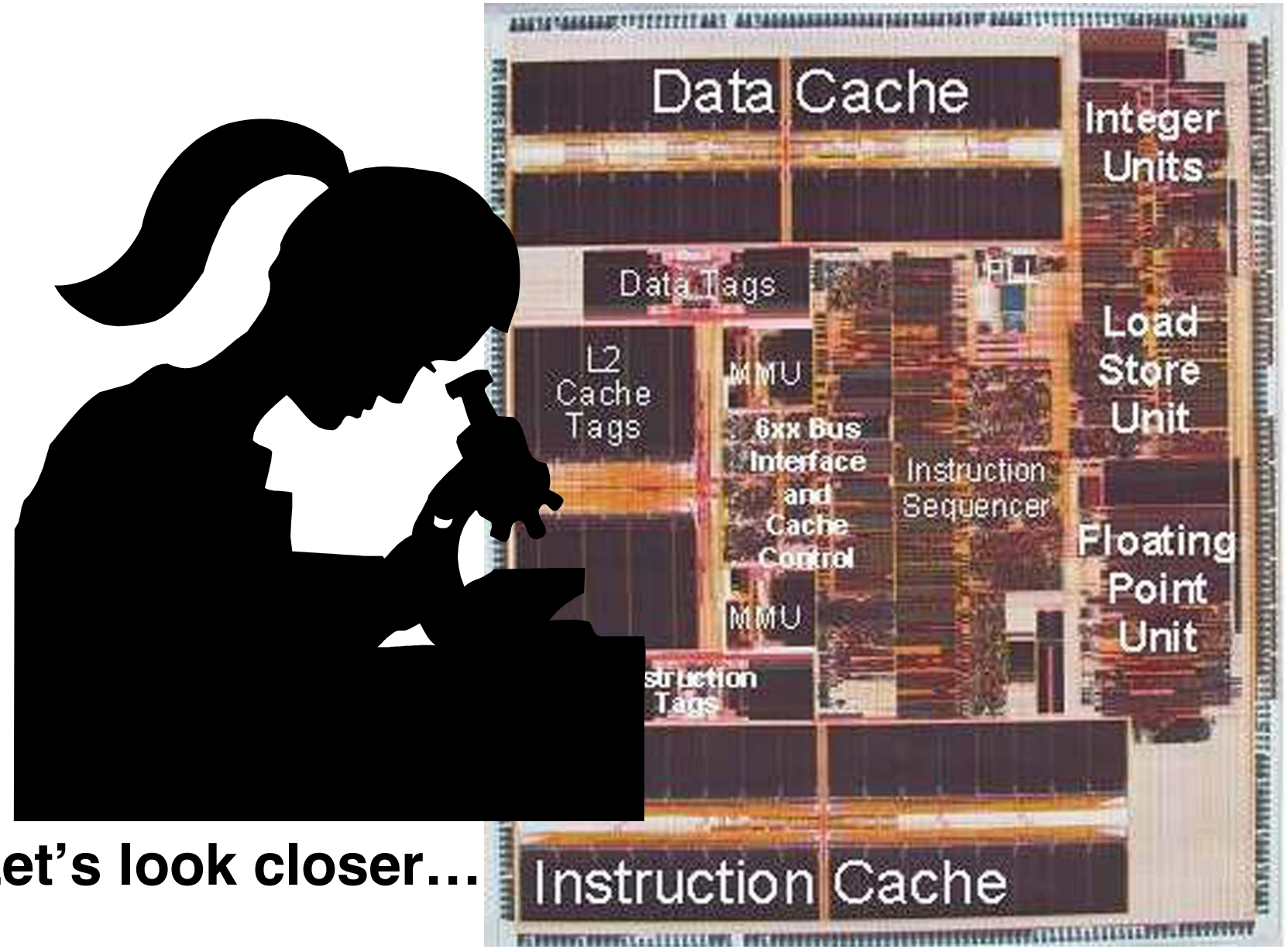
Logic Design

- Next 2 weeks: we'll study how a modern processor is built starting with basic logic elements as building blocks.
- Why study logic design?
  - Understand what processors can do fast and what they can't do fast (avoid slow things if you want your code to run fast!)
  - Background for more detailed hardware courses (CS 150, CS 152)

Logic Gates

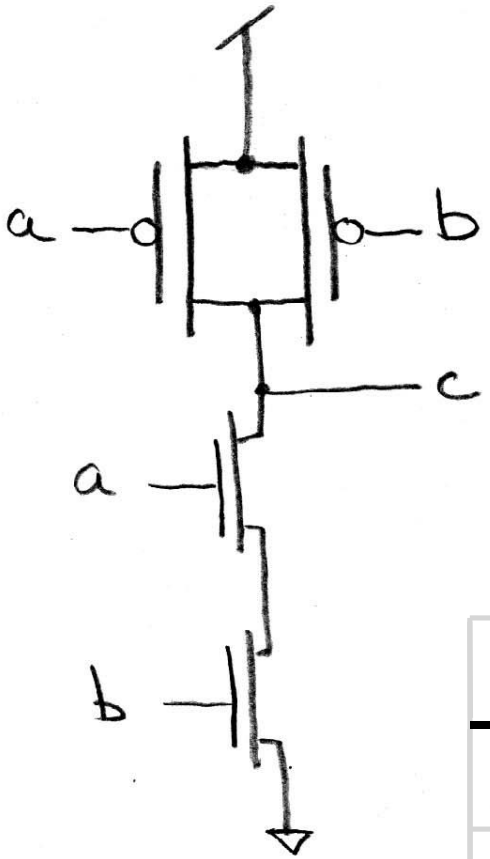
- Basic building blocks are logic *gates*.
  - In the beginning, did ad hoc designs, and then saw patterns repeated, gave names
  - Can build gates with transistors and resistors
- Then found theoretical basis for design
  - Can represent and reason about gates with truth tables and Boolean algebra
  - Assume know truth tables and Boolean algebra from a math or circuits course.
  - Section B.2 in the textbook has a review

Physical Hardware

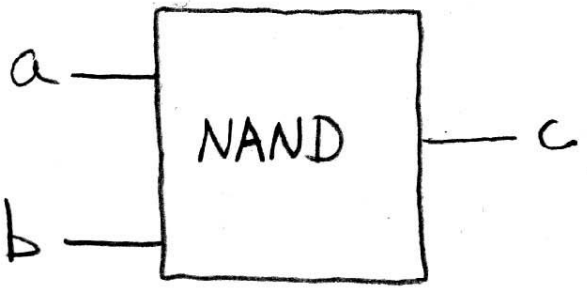


Let's look closer…

Gate-level view vs. Block diagram

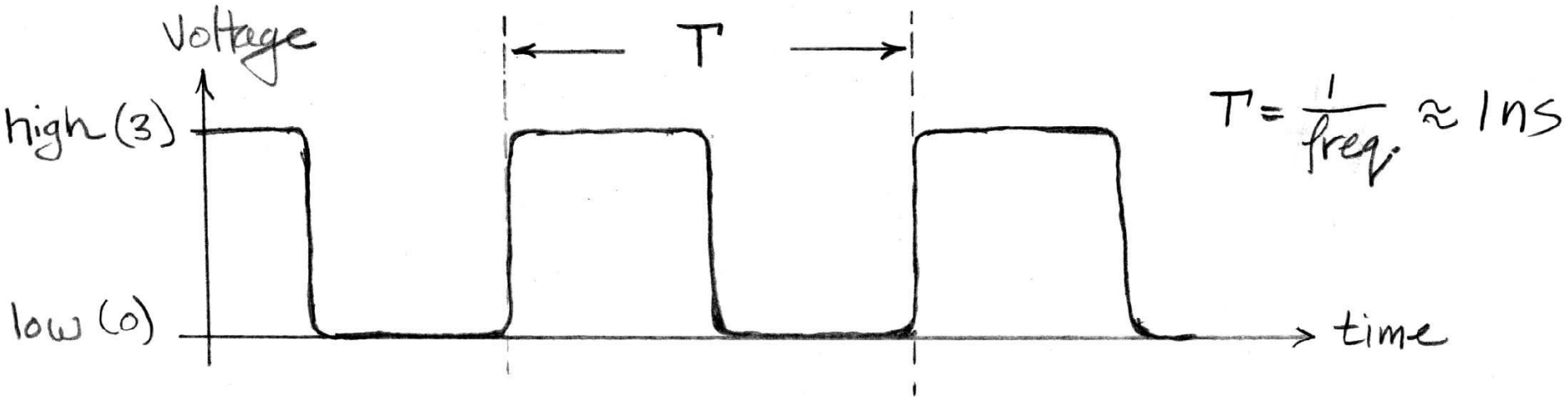


≡

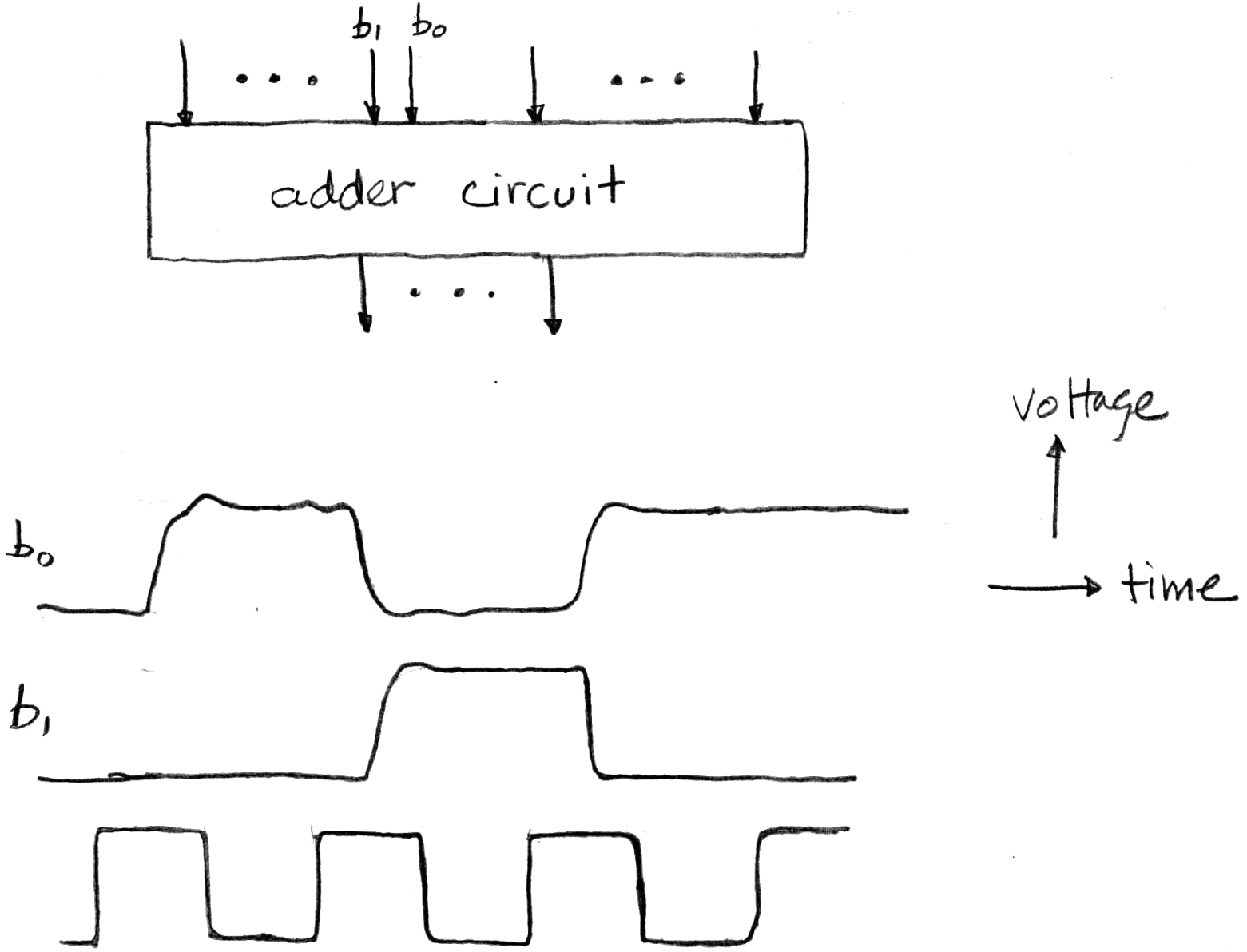


A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

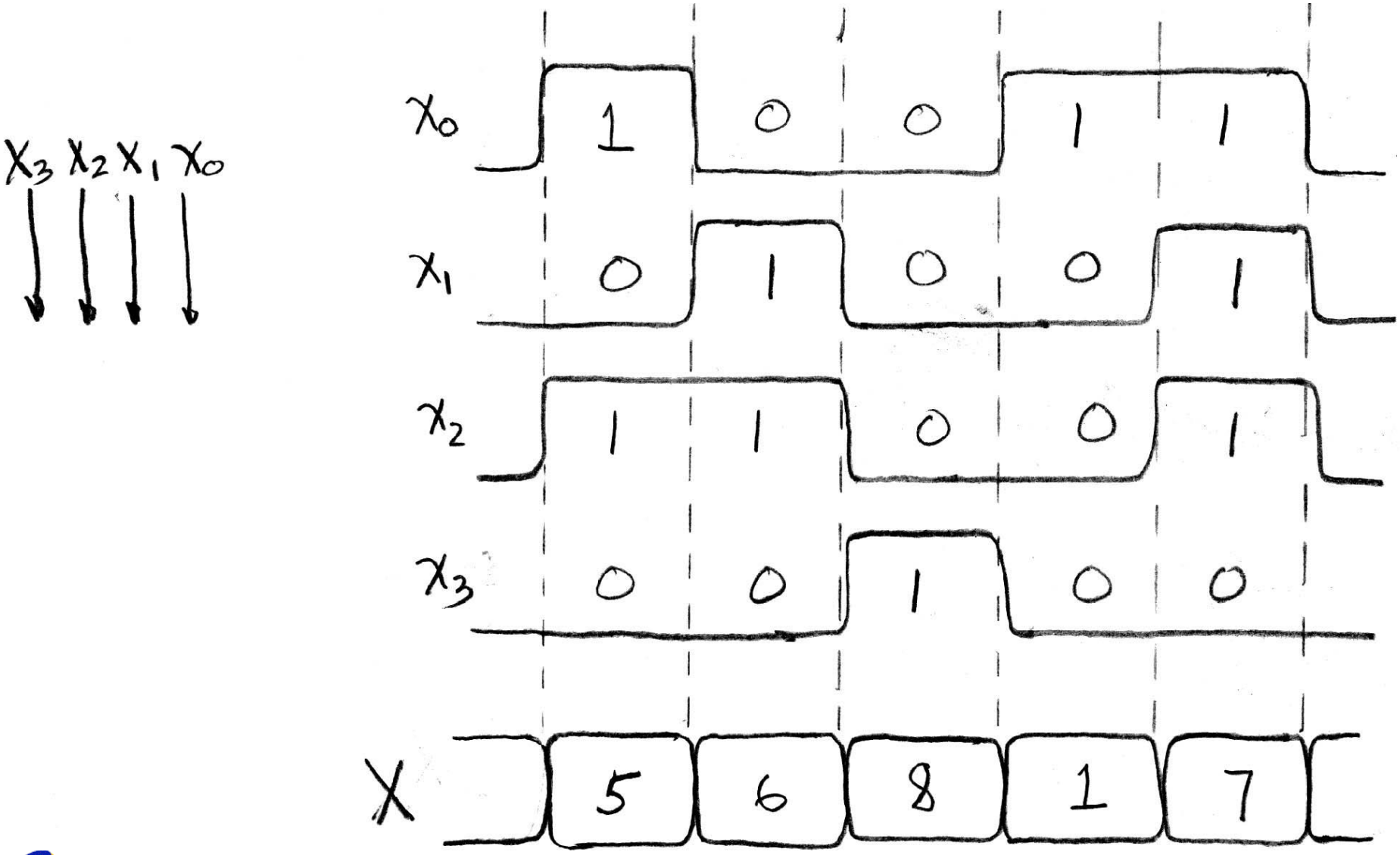
Signals and Waveforms: Clocks



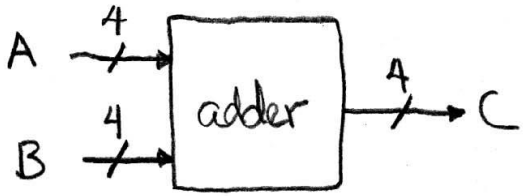
Signals and Waveforms: Adders



Signals and Waveforms: Grouping

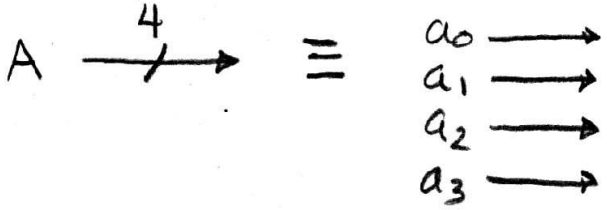


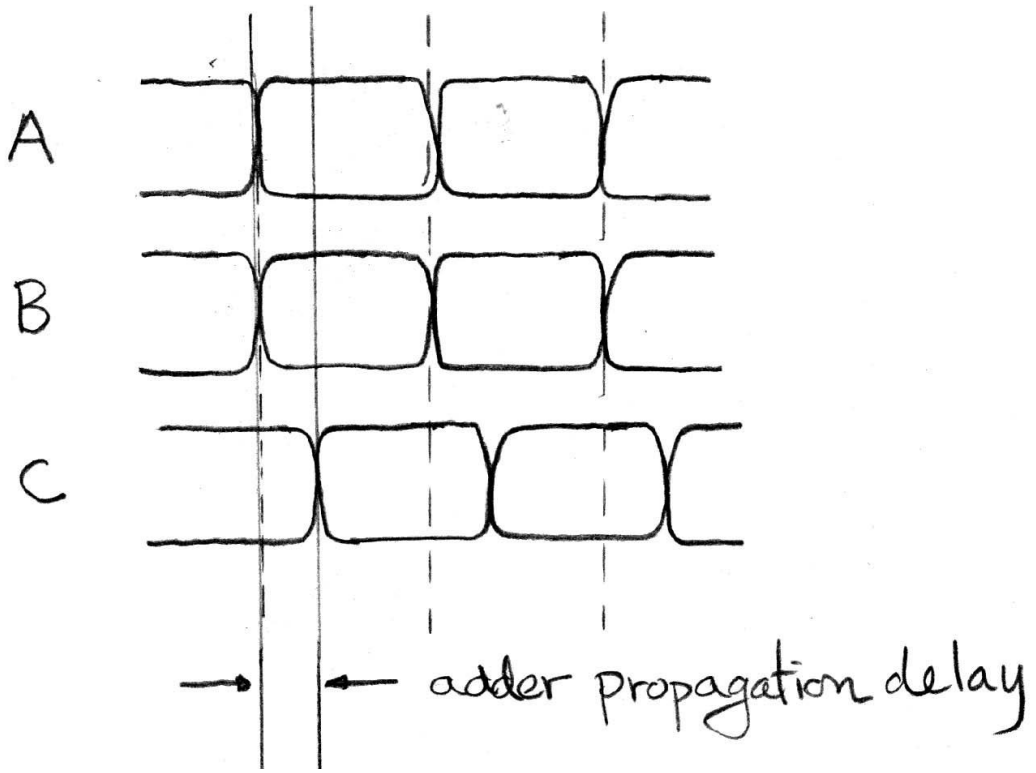
Signals and Waveforms: Circuit Delay



$A = [a_3, a_2, a_1, a_0]$

$B = [b_3, b_2, b_1, b_0]$

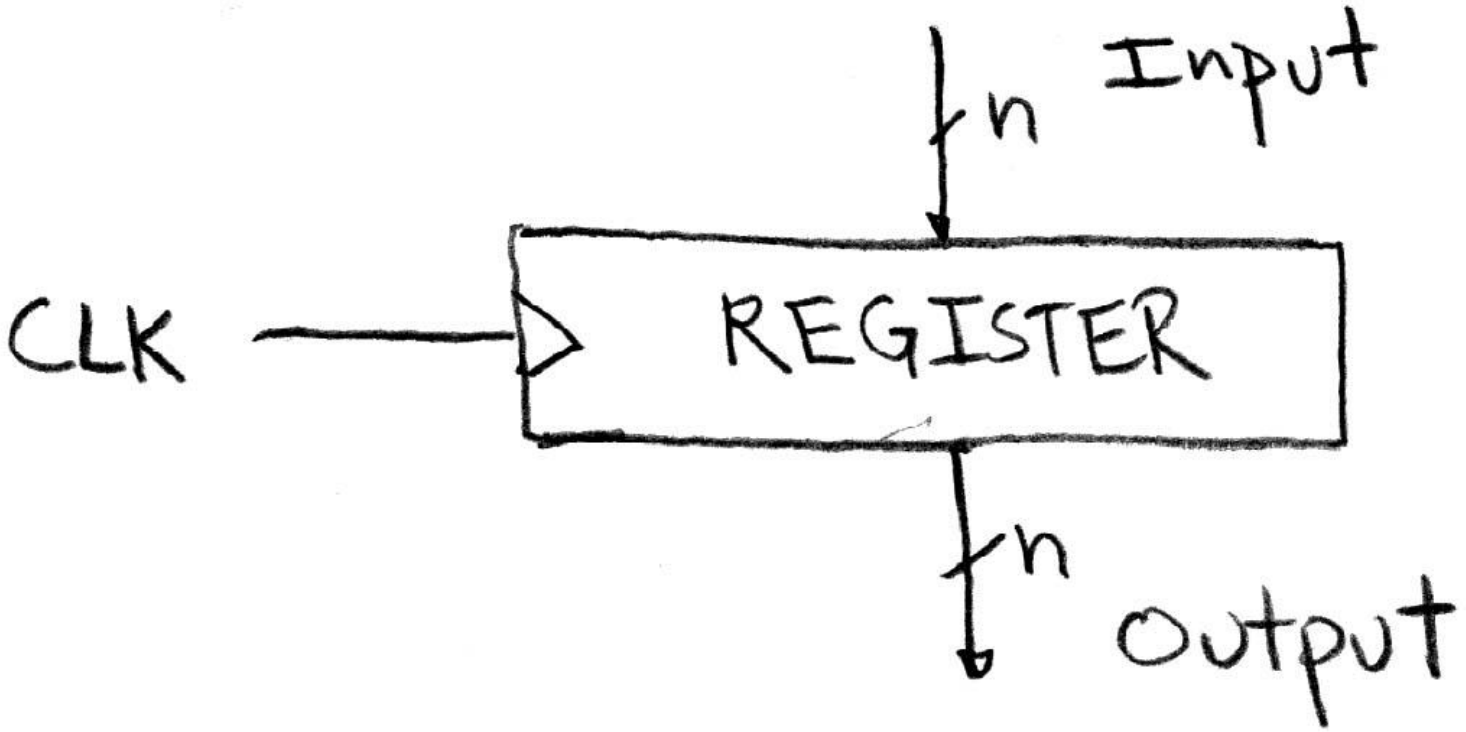




Combinational Logic

- Complex logic blocks are built from basic AND, OR, NOT building blocks we'll see shortly.
- A *combinational* logic block is one in which the output is a function only of its current input.
- Combinational logic cannot have memory (e.g., a register is not a combinational unit).

Circuits with STATE (e.g., register)



Administrivia

- **Midterm tonight @ 7pm in 1 Le Conte. Heard this enough yet?**

Peer Instruction

A. SW can peek at HW (past ISA abstraction boundary) for optimizations

B. SW can depend on particular HW implementation of ISA

C. Timing diagrams serve as a critical debugging tool in the EE toolkit

	ABC
1:	FFF
2:	FFT
3:	FTF
4:	FTT
5:	TFF
6:	TFT
7:	TTF
8:	TTT

And in conclusion…

- ISA is very important abstraction layer
  - Contract between HW and SW
- Basic building blocks are logic *gates*
- Clocks control pulse of our circuits
- Voltages are analog, quantized to 0/1
- Circuit delays are fact of life
- Two types
  - Stateless Combinational Logic (&,|,~)
  - State circuits (e.g., registers)