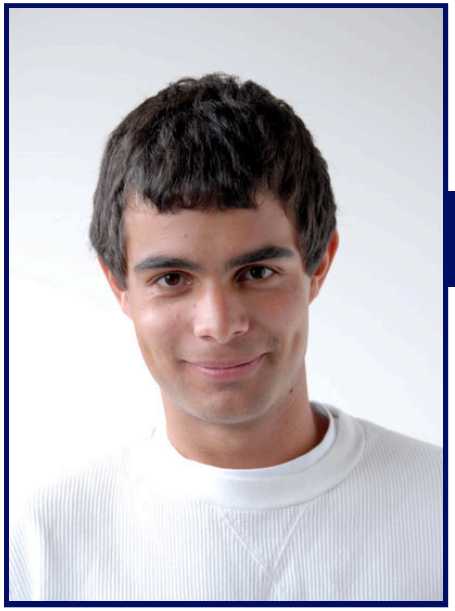
inst.eecs.berkeley.edu/~cs61c

CS61C : Machine Structures

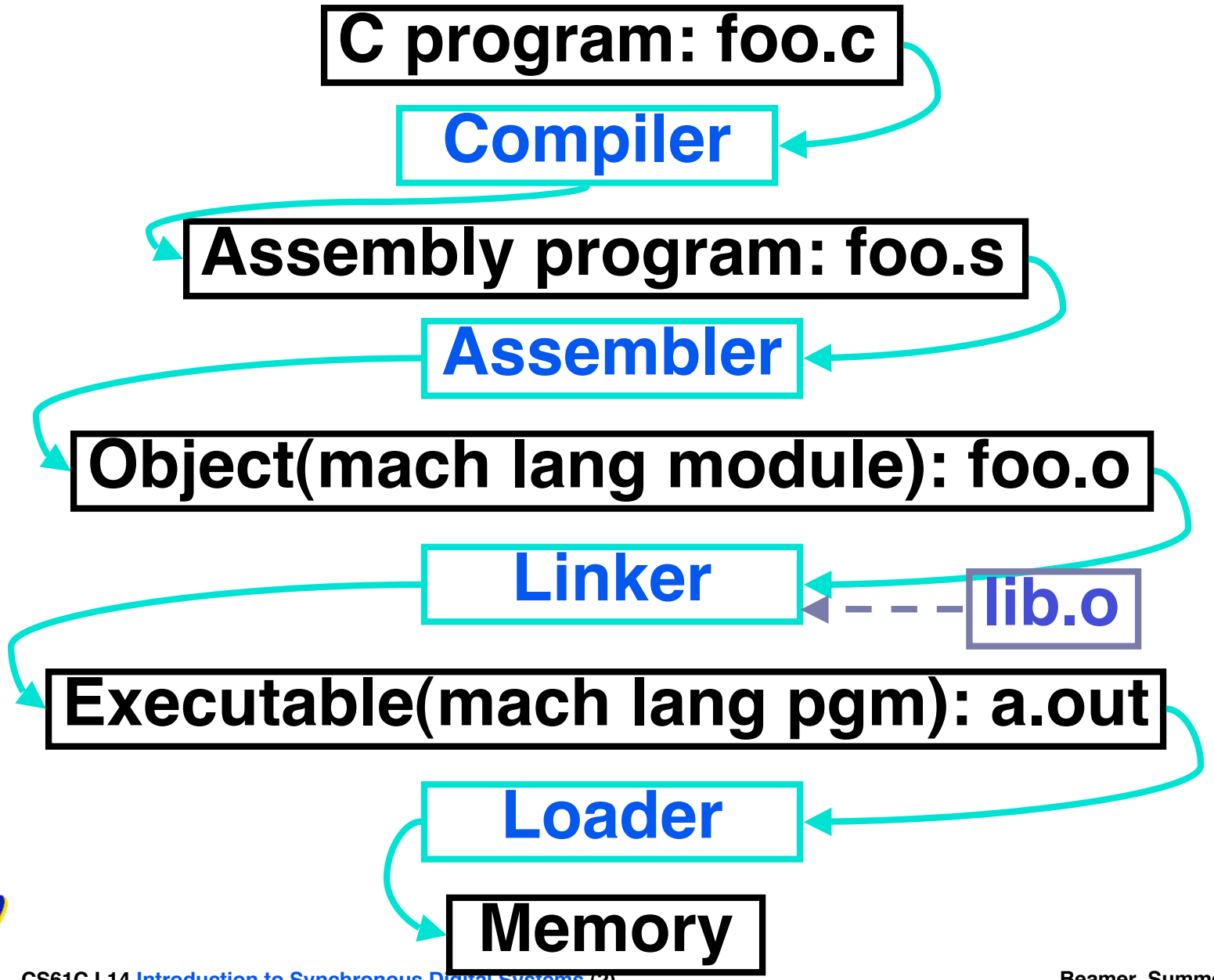
Lecture #14
Introduction to Synchronous Digital Systems

2007-7-18

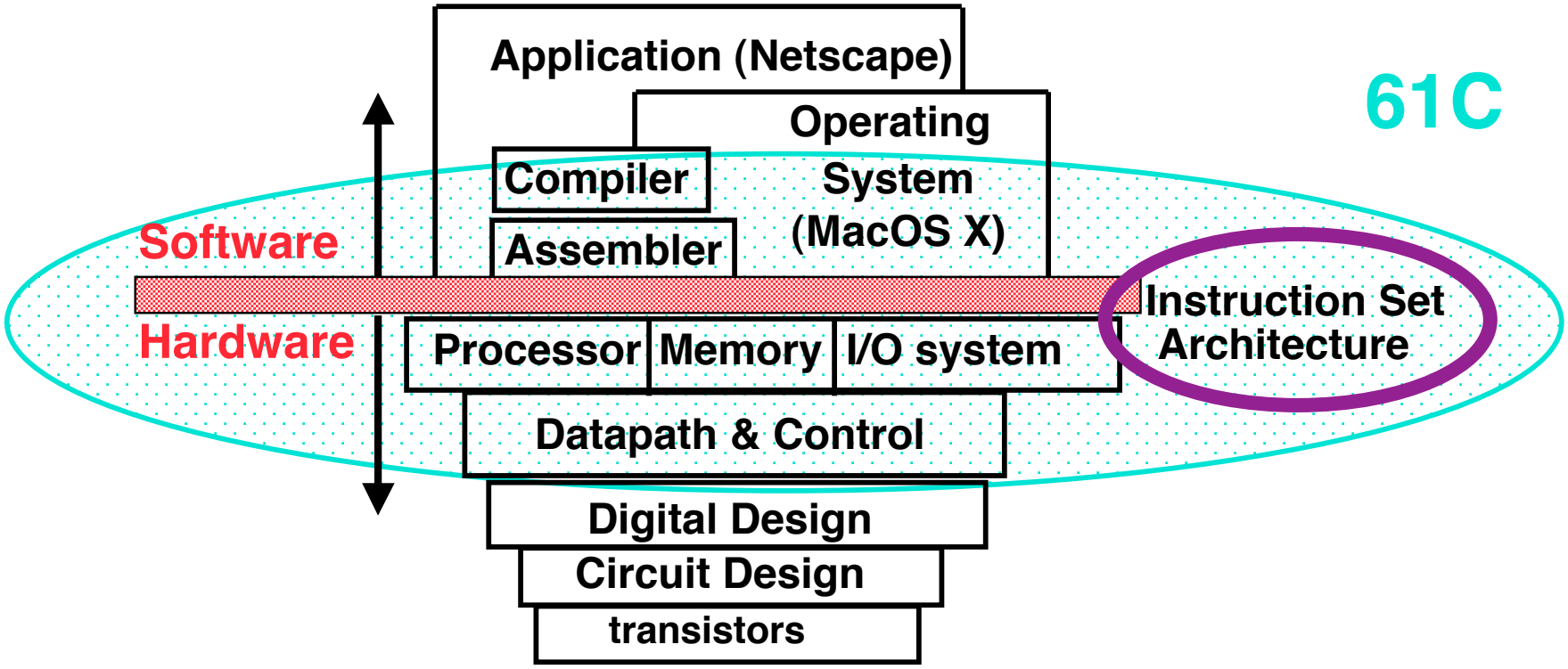


Scott Beamer, Instructor

Review



What are "Machine Structures"?



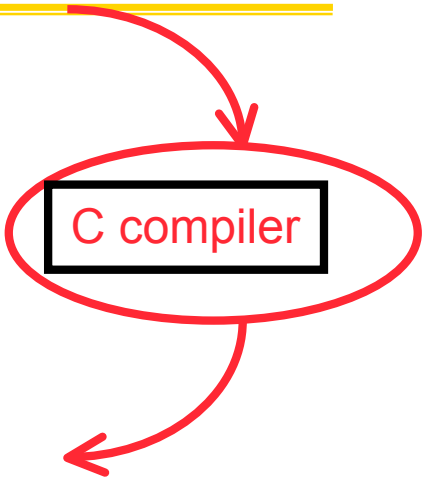
Coordination of many *levels of abstraction*

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

Below the Program

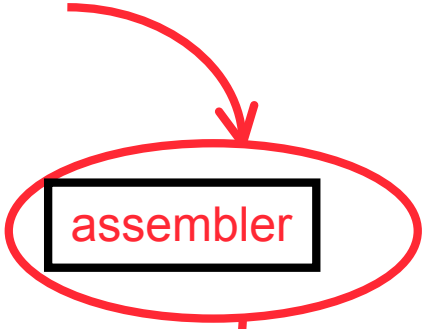
- **High-level language program (in C)**

```
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 . . .
```

Synchronous Digital Systems

The hardware of a processor, such as the MIPS, is an example of a Synchronous Digital System

Synchronous:

- **Means all operations are coordinated by a central clock.**
  - **It keeps the "heartbeat" of the system!**

Digital:

- **Mean all values are represented by discrete values**
- **Electrical signals are treated as 1's and 0's and grouped together to form words.**

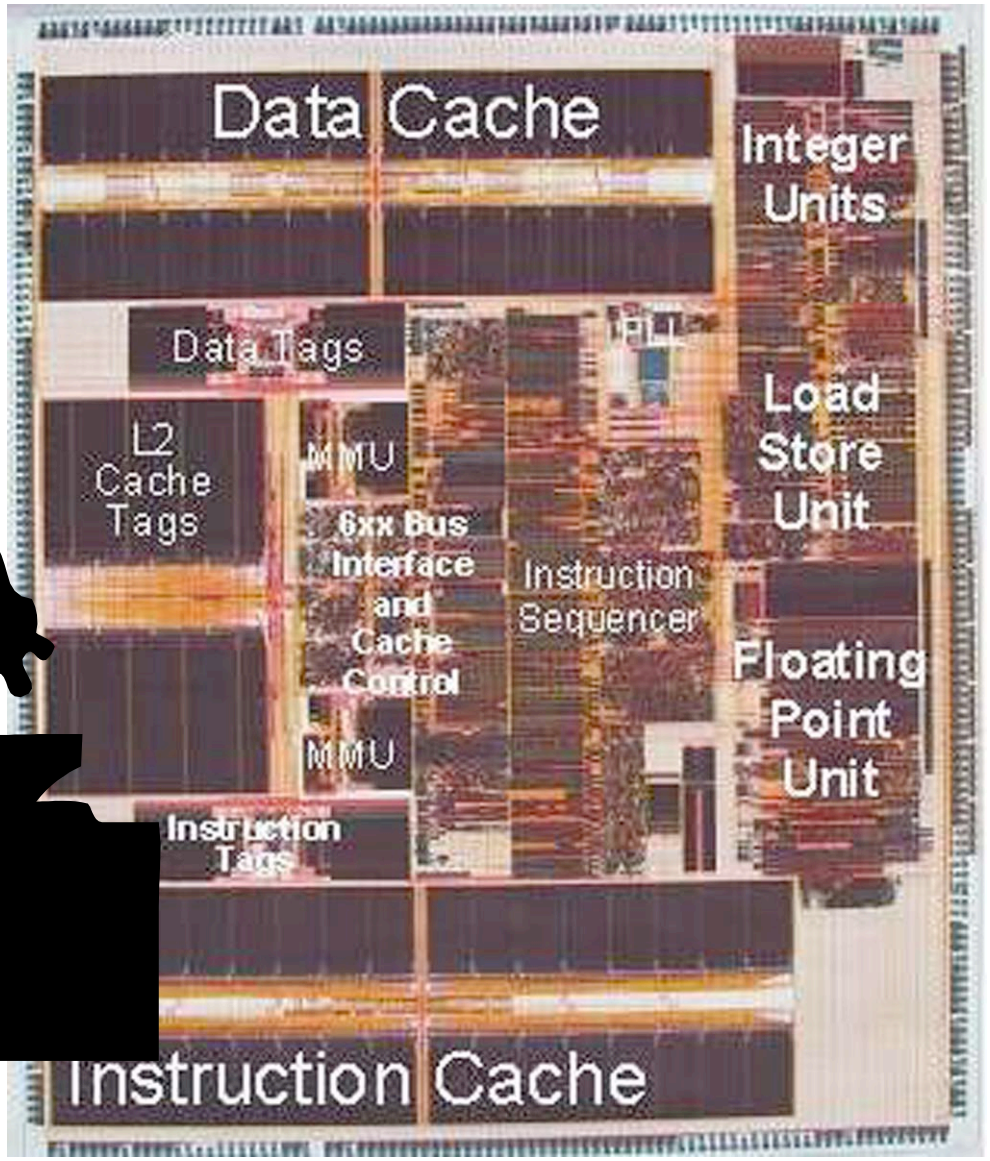
Logic Design

- Next 2 weeks: we'll study how a modern processor is built; starting with basic elements as building blocks.
- Why study hardware design?
  - Understand capabilities and limitations of hardware in general and processors in particular.
  - 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)
  - There is just so much you can do with processors. At some point you may need to design your own custom hardware.

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 some truth tables and Boolean algebra from a math or circuits course.
  - Section B.2 in the textbook has a review

Physical Hardware

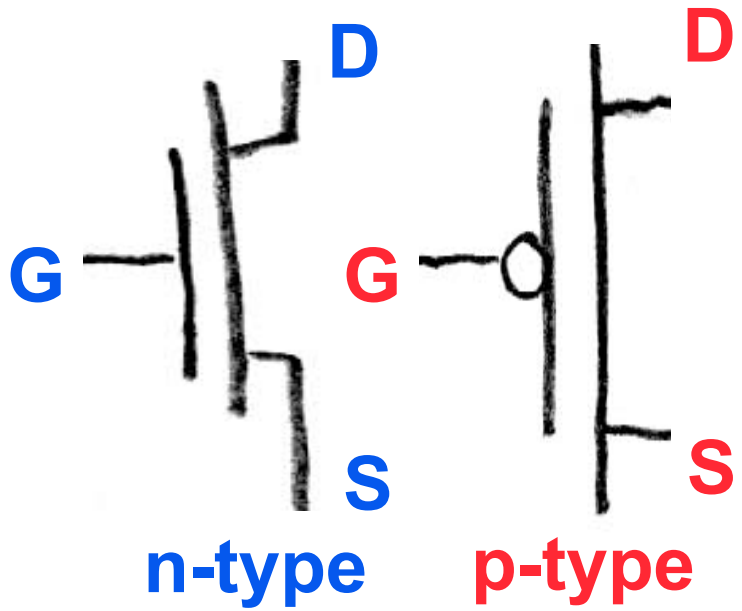


Let's look closer…

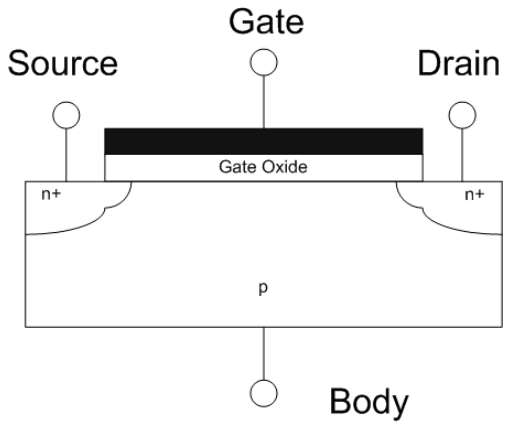
PowerPC

Transistors 101

- MOSFET
  - Metal-Oxide-Semiconductor Field-Effect Transistor
  - Come in two types:
    - n-type NMOSFET
    - p-type PMOSFET



- For n-type (p-type opposite)
  - If voltage not enough between G & S, transistor turns "off" (cut-off) and Drain-Source NOT connected
  - If the G & S voltage is high enough, transistor turns "on" (saturation) and Drain-Source ARE connected

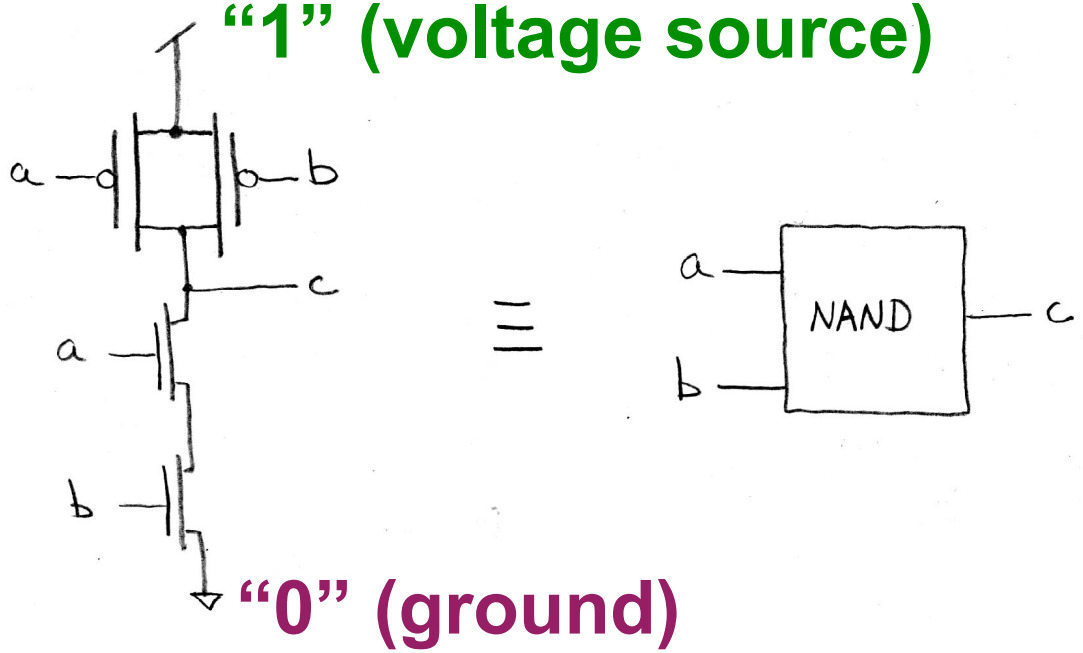


Side view

www.wikipedia.org/wiki/Mosfet

Transistor Circuit Rep. vs. Block diagram

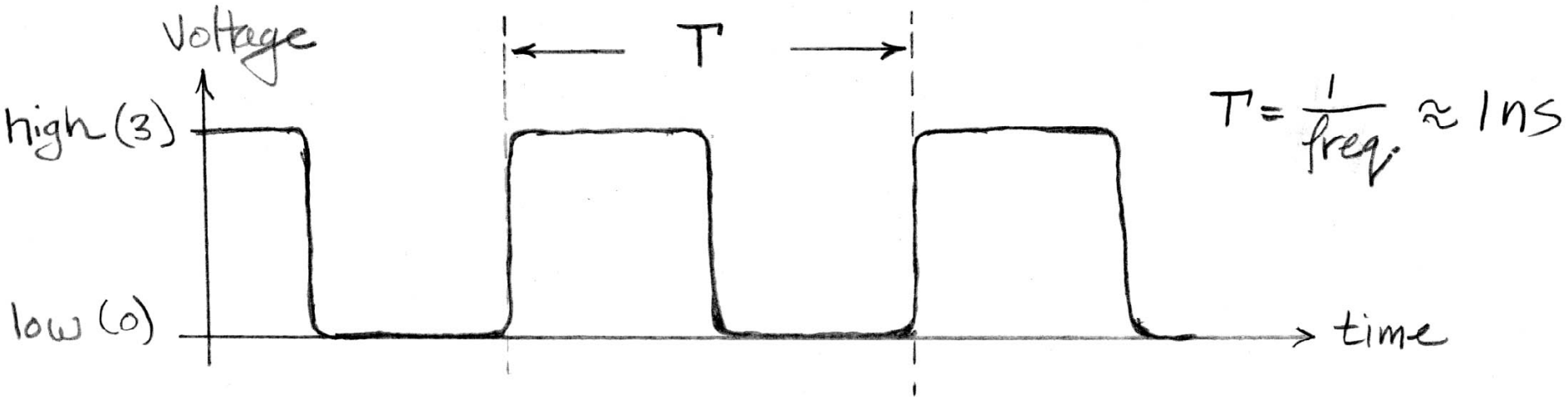
- Chips is composed of nothing but transistors and wires.
- Small groups of transistors form useful building blocks.



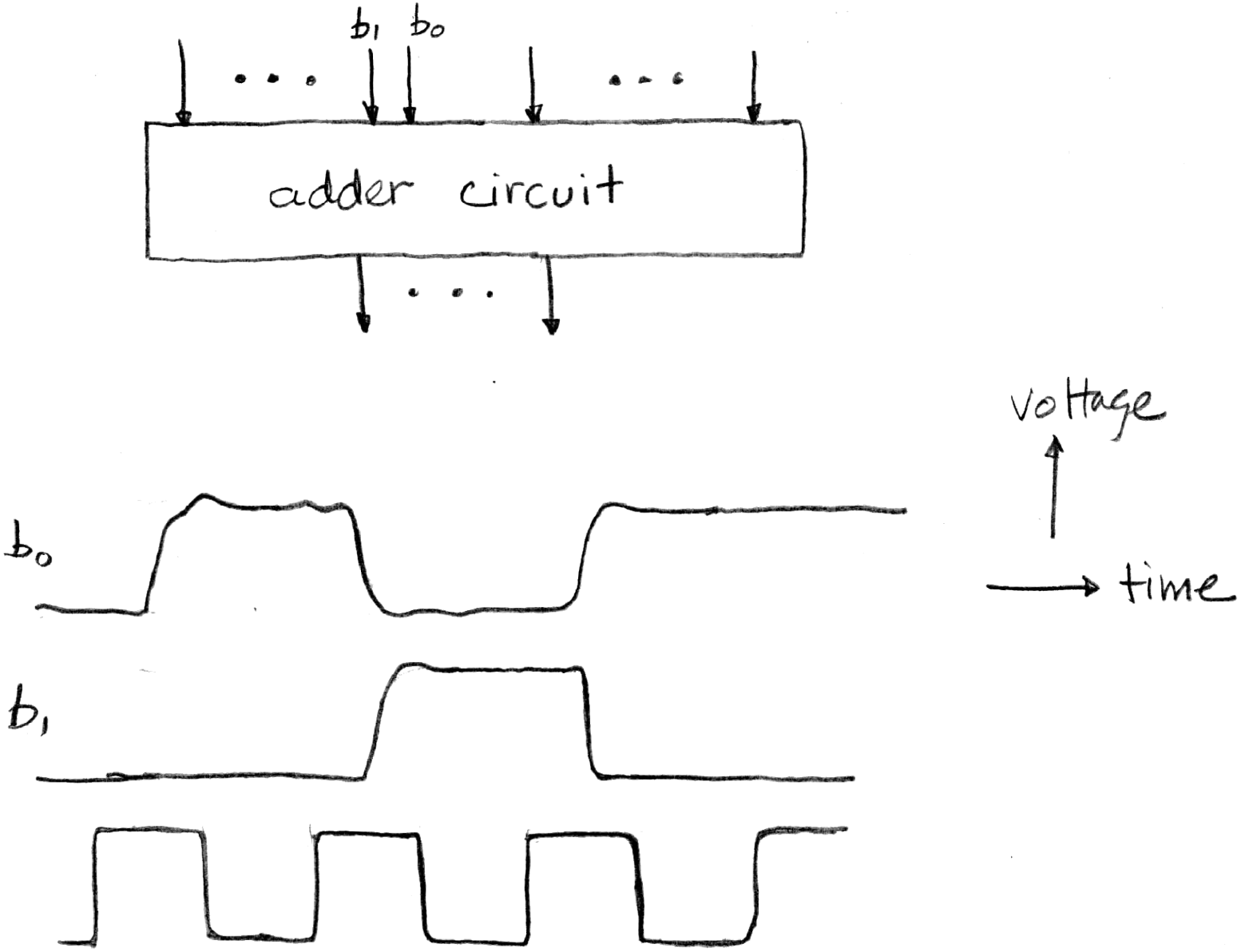
a	b	c
0	0	1
0	1	1
1	0	1
1	1	0

- Block are organized in a hierarchy to build higher-level blocks: ex: adders.

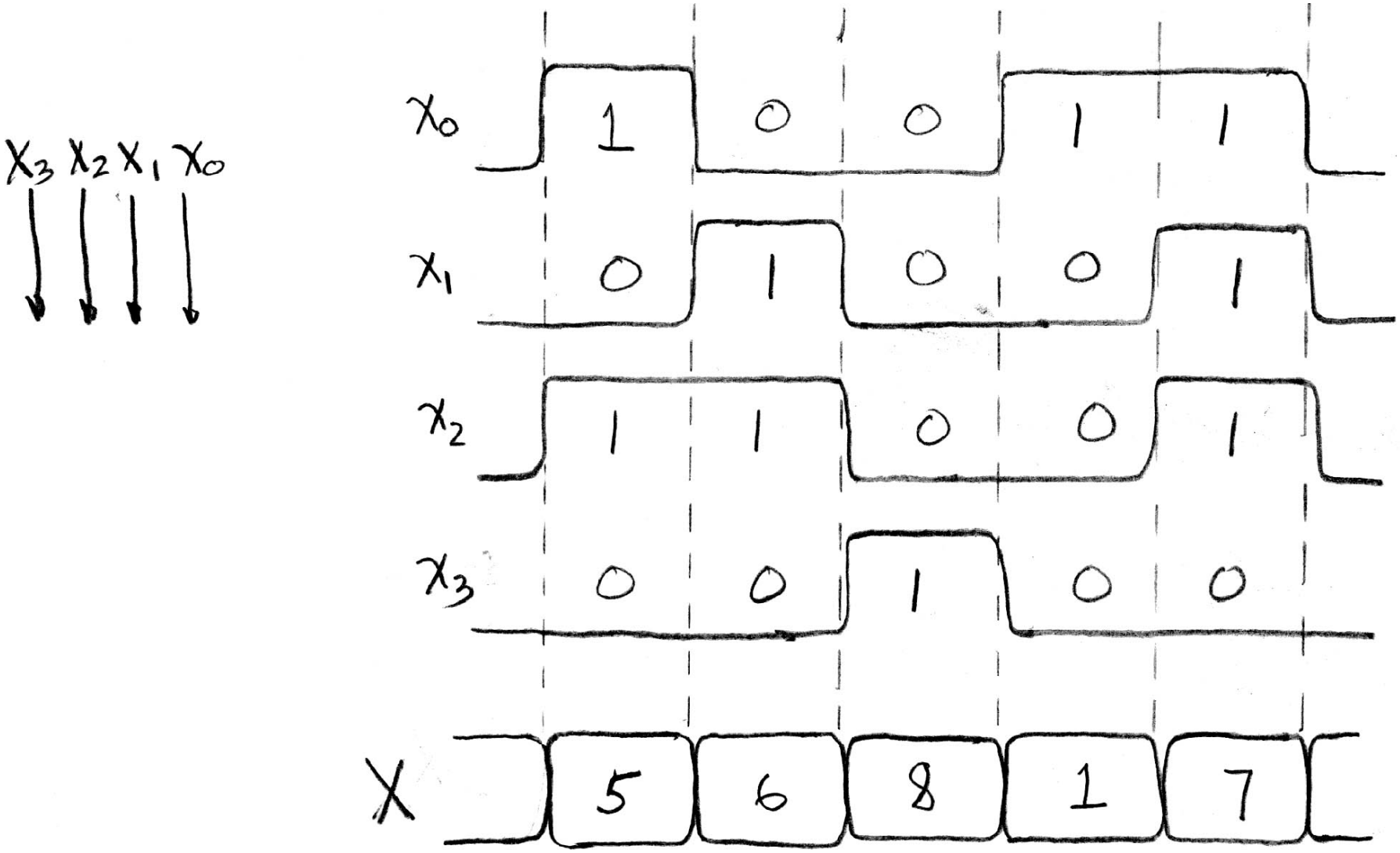
Signals and Waveforms: Clocks



Signals and Waveforms: Adders

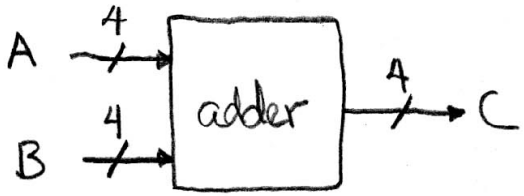


Signals and Waveforms: Grouping



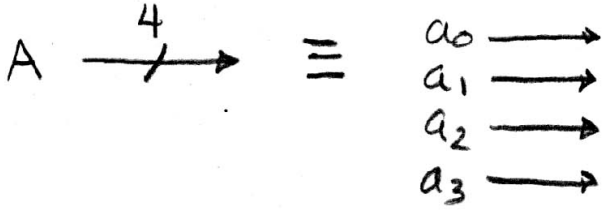
Bus - more than one signal treated as a unit

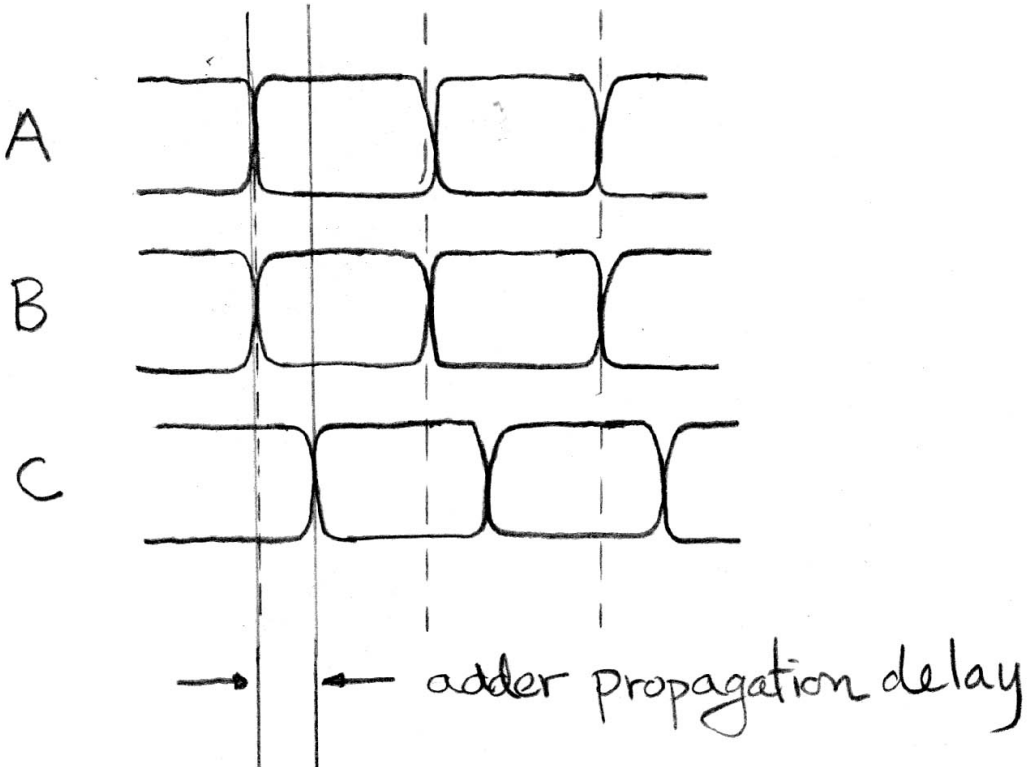
Signals and Waveforms: Circuit Delay



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

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

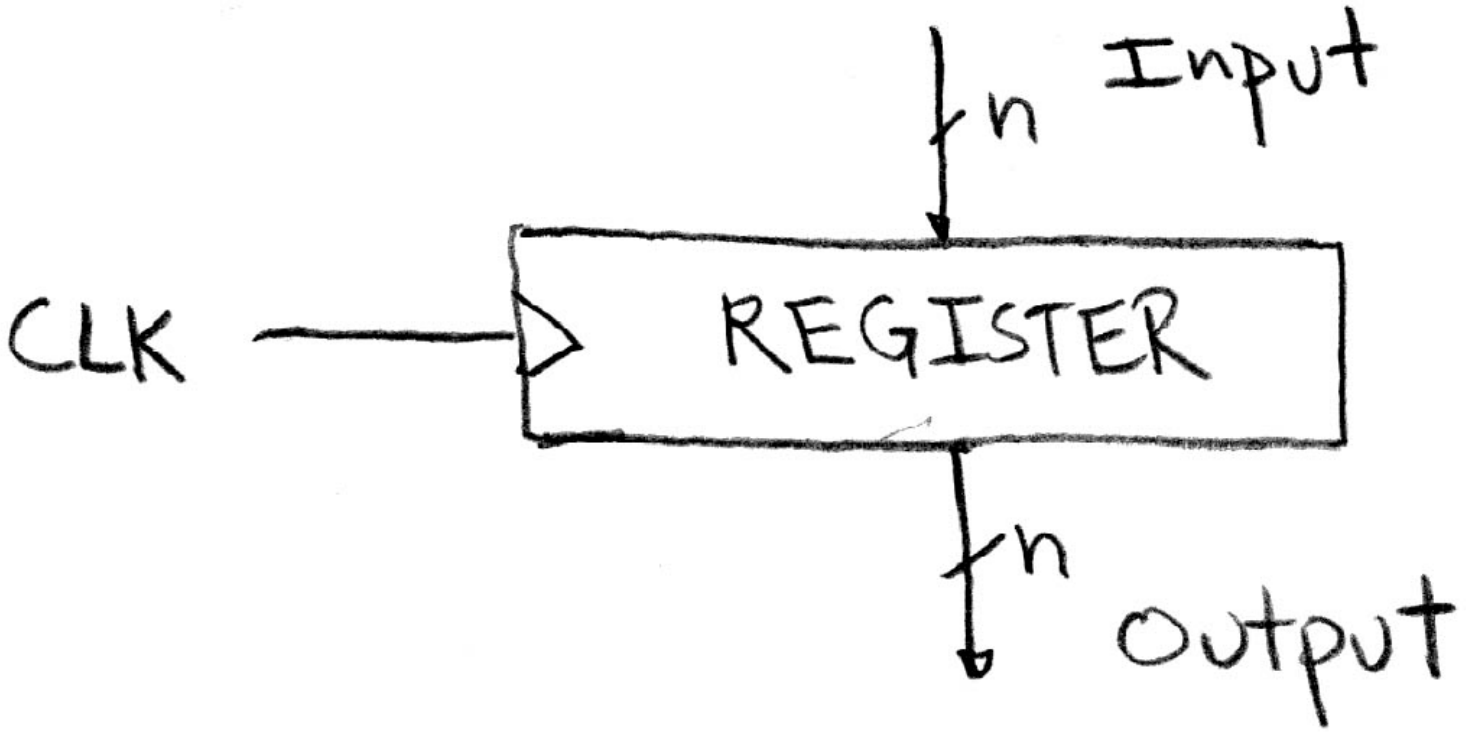




Type of Circuits

- Synchronous Digital Systems are made up of two basic types of circuits:
- Combinational Logic (CL) circuits
  - Our previous adder circuit is an example.
  - Output is a function of the inputs only.
  - Similar to a pure function in mathematics, y = f(x). (No way to store information from one invocation to the next. No side effects)
- State Elements: circuits that store information.

Circuits with STATE (e.g., register)



Peer Instruction

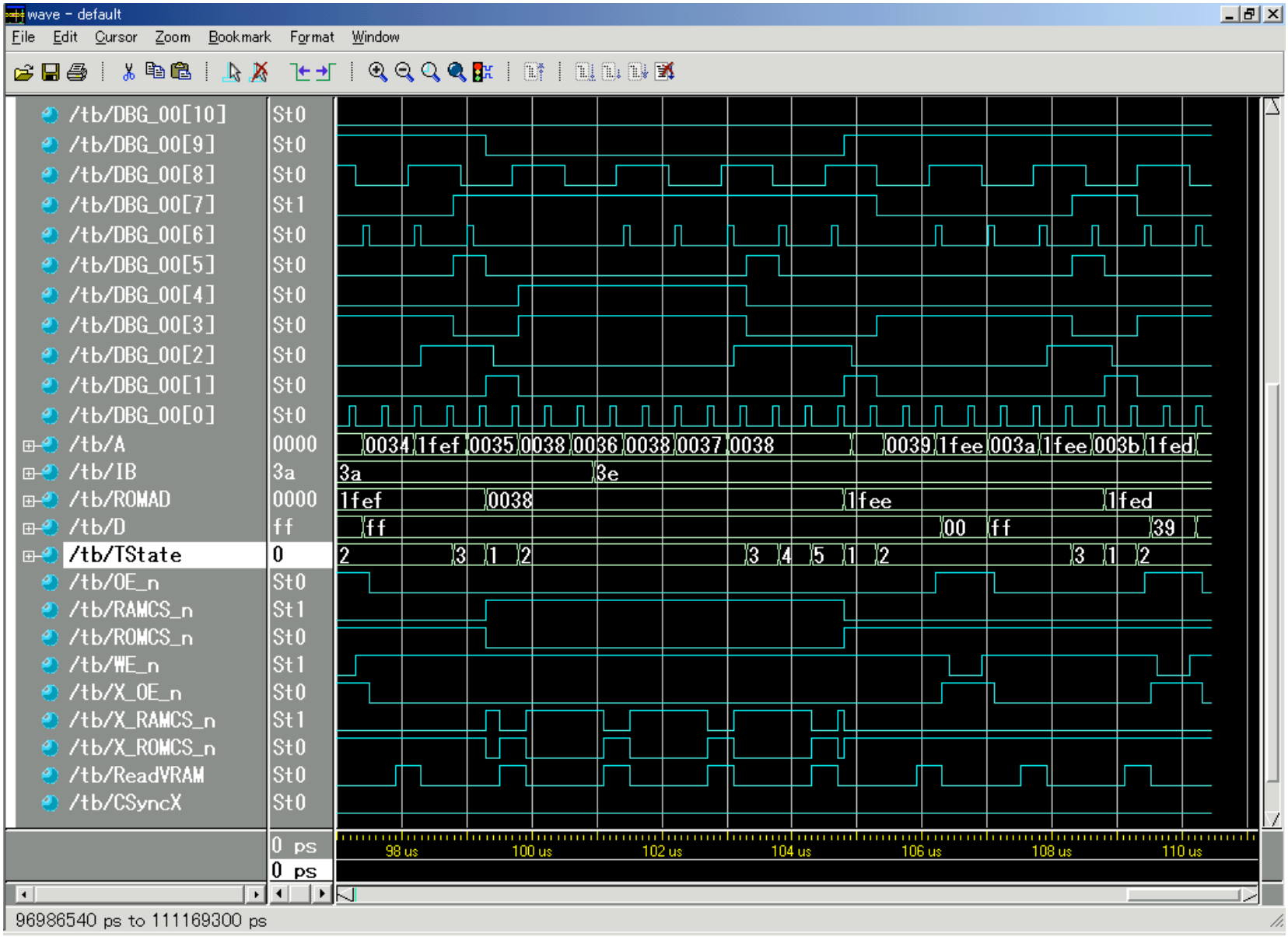
A. SW can peek at HW (past ISA abstraction boundary) for optimizations

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

Sample Debugging Waveform



And in semi conclusion…

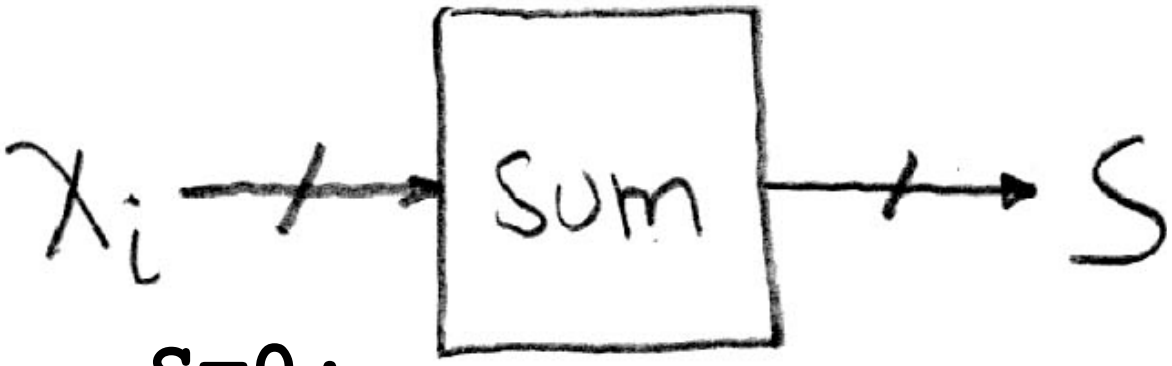
- ISA is very important abstraction layer
  - Contract between HW and SW
- Basic building blocks are logic *gates*
- Clocks control pulse of our circuits
- Voltages are analog, quantized to 0/1
- Circuit delays are fact of life
- Two types
  - Stateless Combinational Logic (&,|,~)
  - State circuits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

Administrivia

- Proj2 due Friday
- Midterm 7-10p on Monday in 10 Evans
- Midterm Review 11-2 on Friday, probably in 10 or 60 Evans
- Scott is not holding OH on Monday, but is holding extra OH on Friday 3-5

Accumulator Example

Why do we need to control the flow of information?



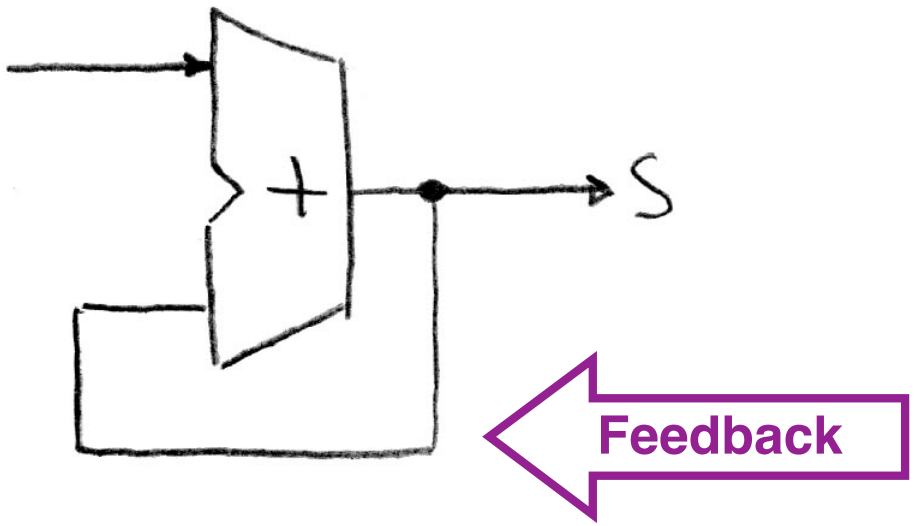
Want:

```
S=0;
for (i=0;i<n;i++)
    S = S + Xi
```

Assume:

- Each X value is applied in succession, one per cycle.
- After n cycles the sum is present on S.

First try…Does this work?

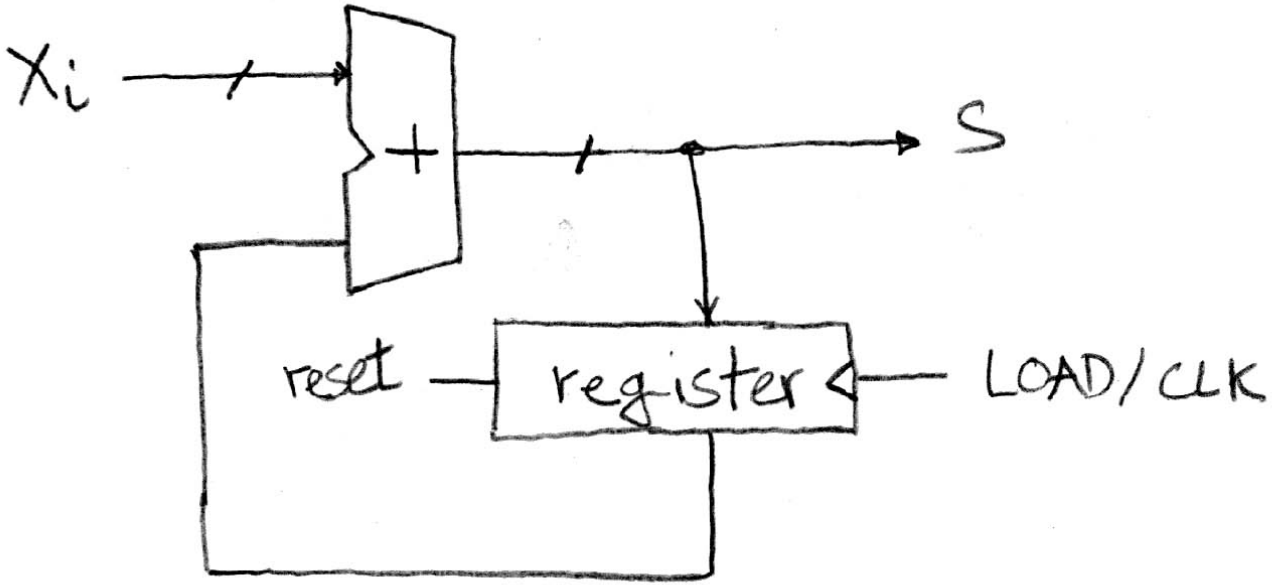


Nope!

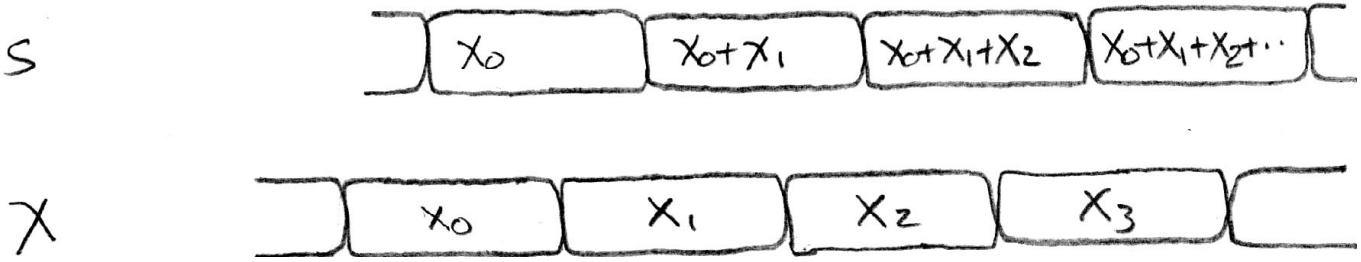
Reason #1… What is there to control the next iteration of the '`for`' loop?

Reason #2… How do we say: '`S=0`'?

Second try…How about this?

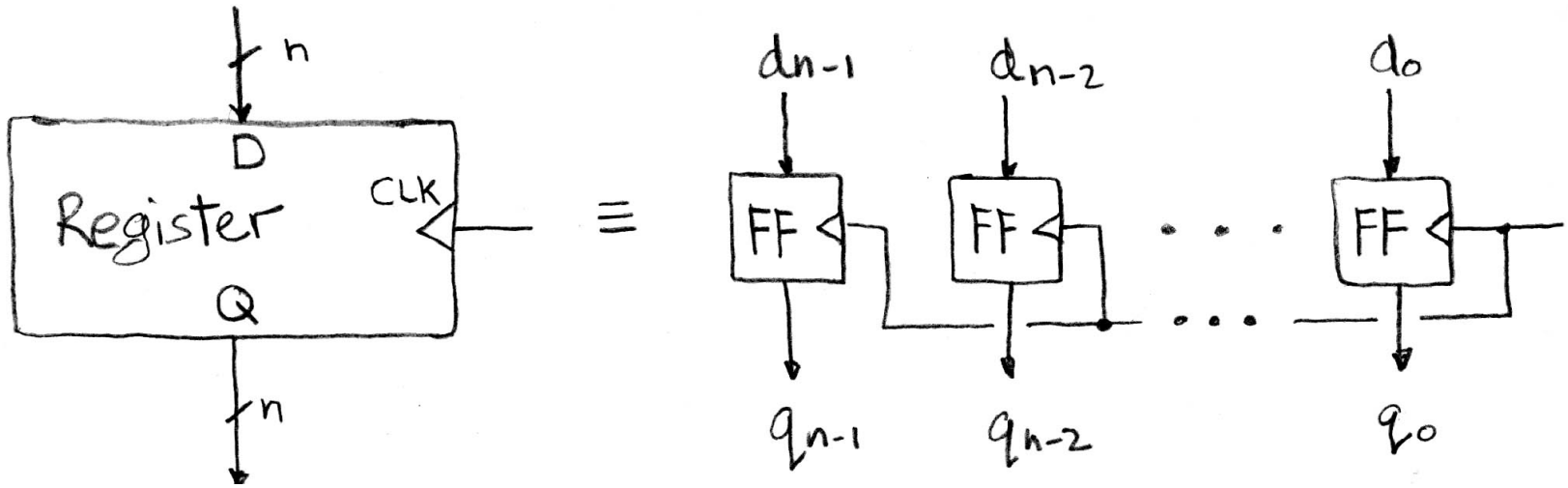


Rough timing…



Register is used to hold up the transfer of data to adder.

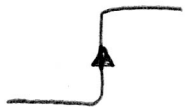
Register Details…What's inside?



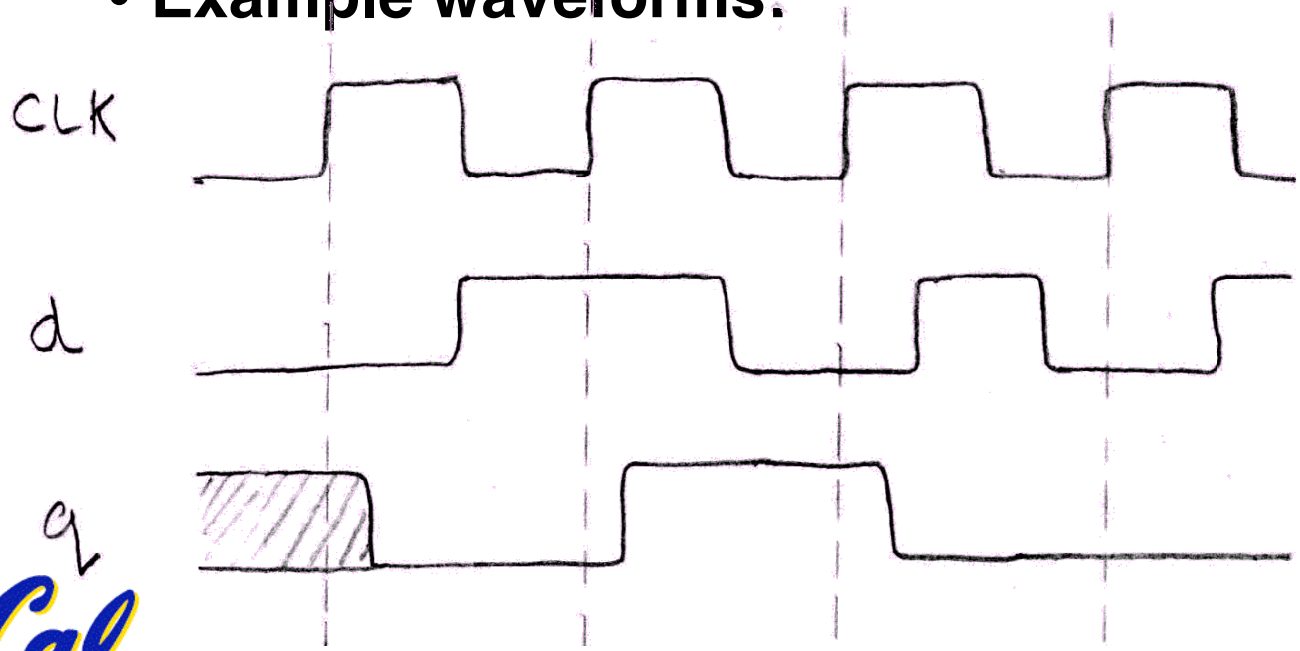
- n instances of a "Flip-Flop"
- Flip-flop name because the output flips and flops between and 0,1
- D is "data", Q is "output"
- Also called "d-type Flip-Flop"

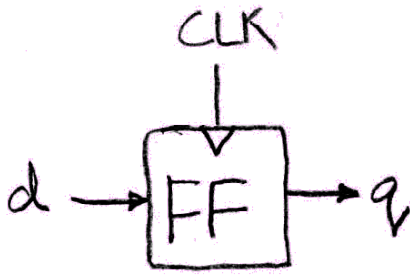
What's the timing of a Flip-flop? (1/2)

- **Edge-triggered d-type flip-flop**
  - **This one is "positive edge-triggered"**

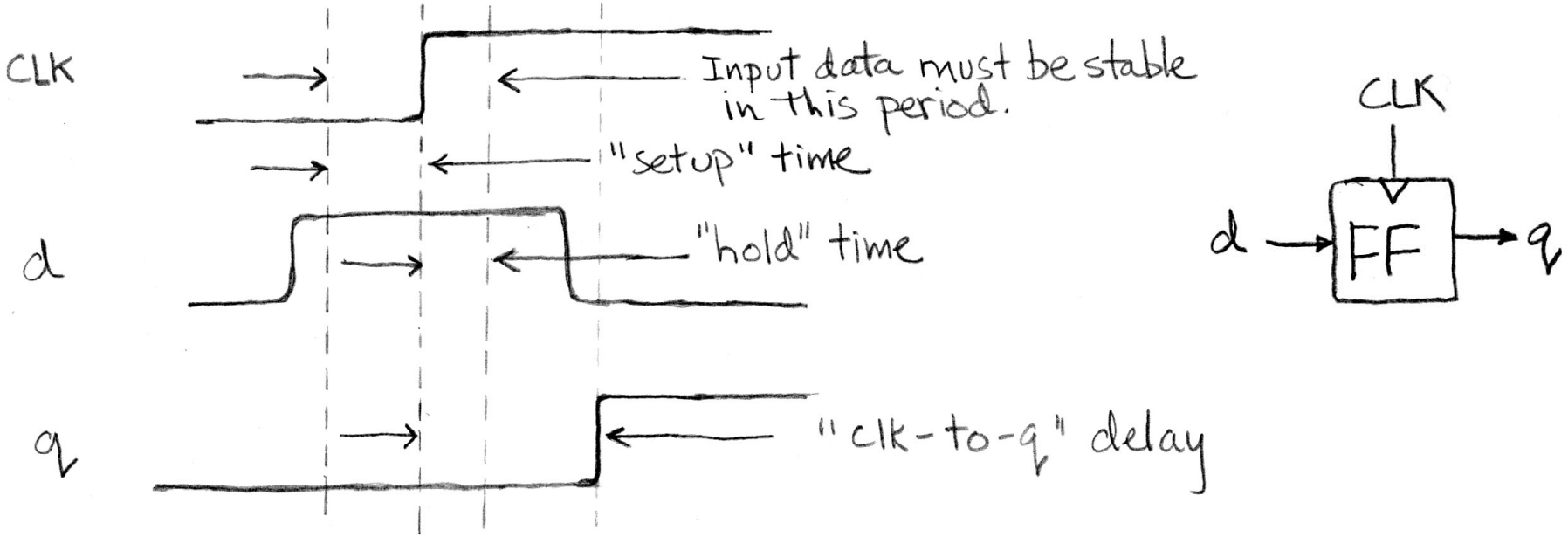


- **"On the rising edge of the clock, the input d is sampled and transferred to the output. At all other times, the input d is ignored."**
- **Example waveforms:**

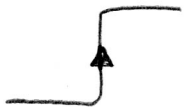




What's the timing of a Flip-flop? (2/2)



- **Edge-triggered d-type flip-flop**
  - **This one is "positive edge-triggered"**

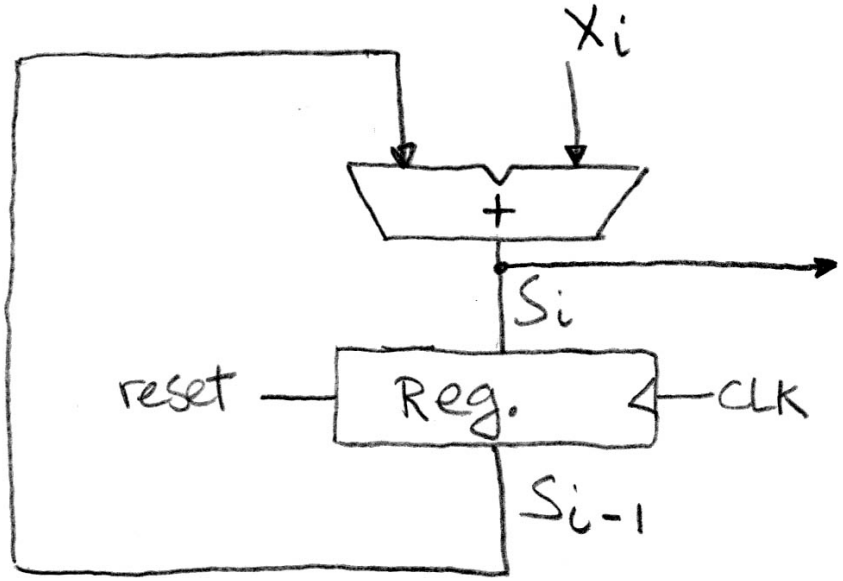


- **"On the rising edge of the clock, the input d is sampled and transferred to the output. At all other times, the input d is ignored."**

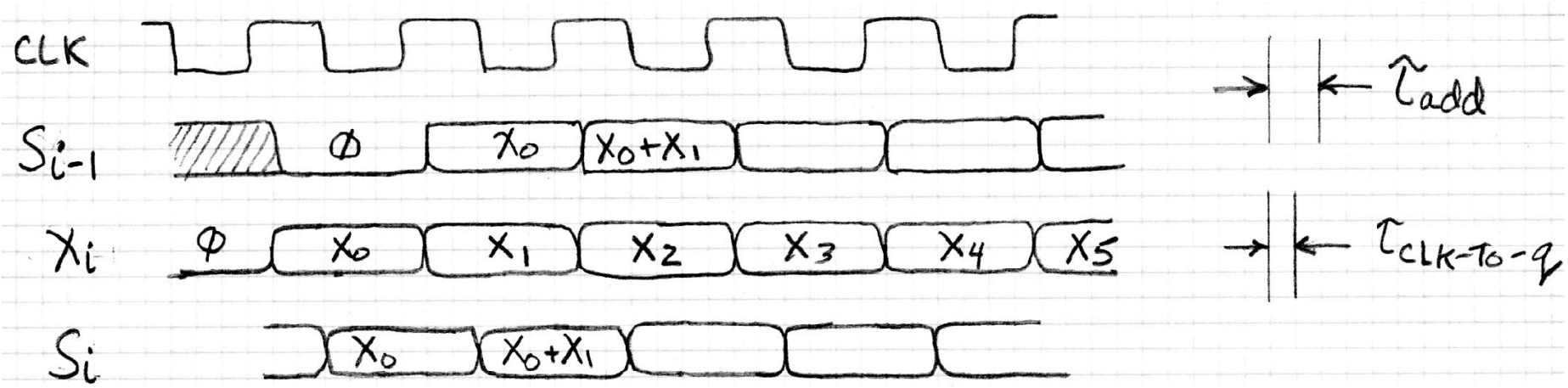
Recap of Timing Terms

- **Clock (CLK)** - steady square wave that synchronizes system
- **Setup Time** - when the input must be stable before the rising edge of the CLK
- **Hold Time** - when the input must be stable after the rising edge of the CLK
- **"CLK-to-Q" Delay** - how long it takes the output to change, measured from the rising edge
- **Flip-flop** - one bit of state that samples every rising edge of the CLK
- **Register** - several bits of state that samples on rising edge of CLK or on LOAD

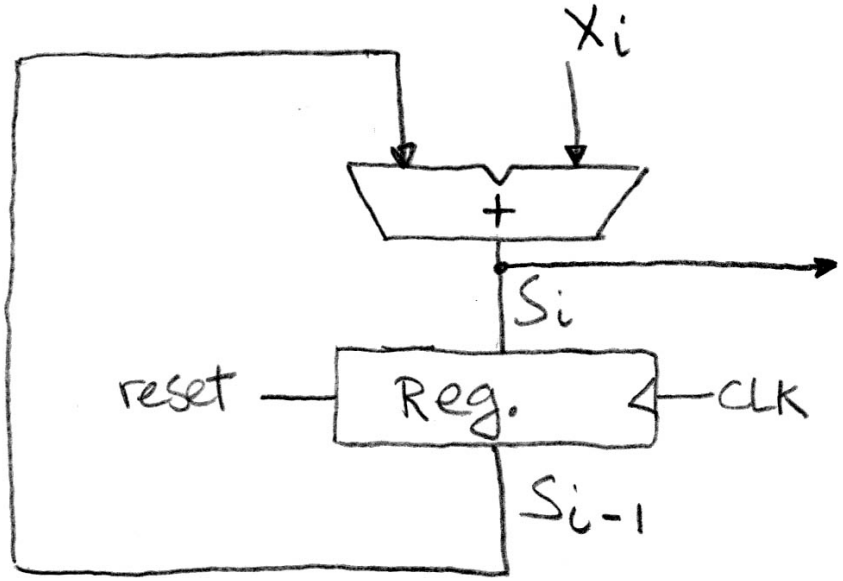
Accumulator Revisited (proper timing 1/2)



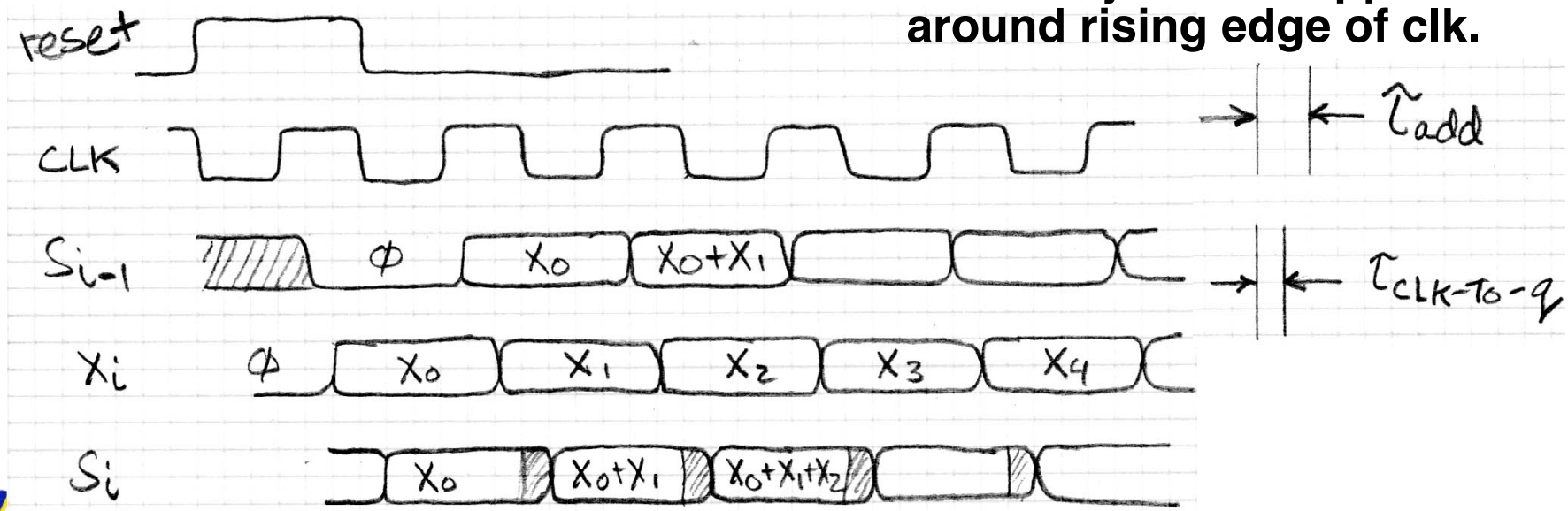
- **Reset input to register is used to force it to all zeros (takes priority over D input).**
- **S_{i-1} holds the result of the i^{th}-1 iteration.**
- **Analyze circuit timing starting at the output of the register.**



Accumulator Revisited (proper timing 2/2)



- reset signal shown.
- Also, in practice X might not arrive to the adder at the same time as S_{i-1}
- S_i temporarily is wrong, but register always captures correct value.
- In good circuits, instability never happens around rising edge of clk.



Peer Instruction

A. CLK-to-Q delays propagate in a synchronized circuit

B. The hold time should be less than the CLK-to-Q delay

C. The minimum period of a usable synchronous circuit is at least the CLK-to-Q delay

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

"And In conclusion…"

- We use feedback to maintain state
- Register files used to build memories
- D Flip-Flops used to build Register files
- Clocks tell us when D Flip-Flops change
  - Setup and Hold times important