CS3350B
Computer Architecture
Winter 2015

Lecture 5.1: Introduction to Synchronous Digital Systems: Switches, Transistors, Gates

Marc Moreno Maza

www.csd.uwo.ca/Courses/CS3350b

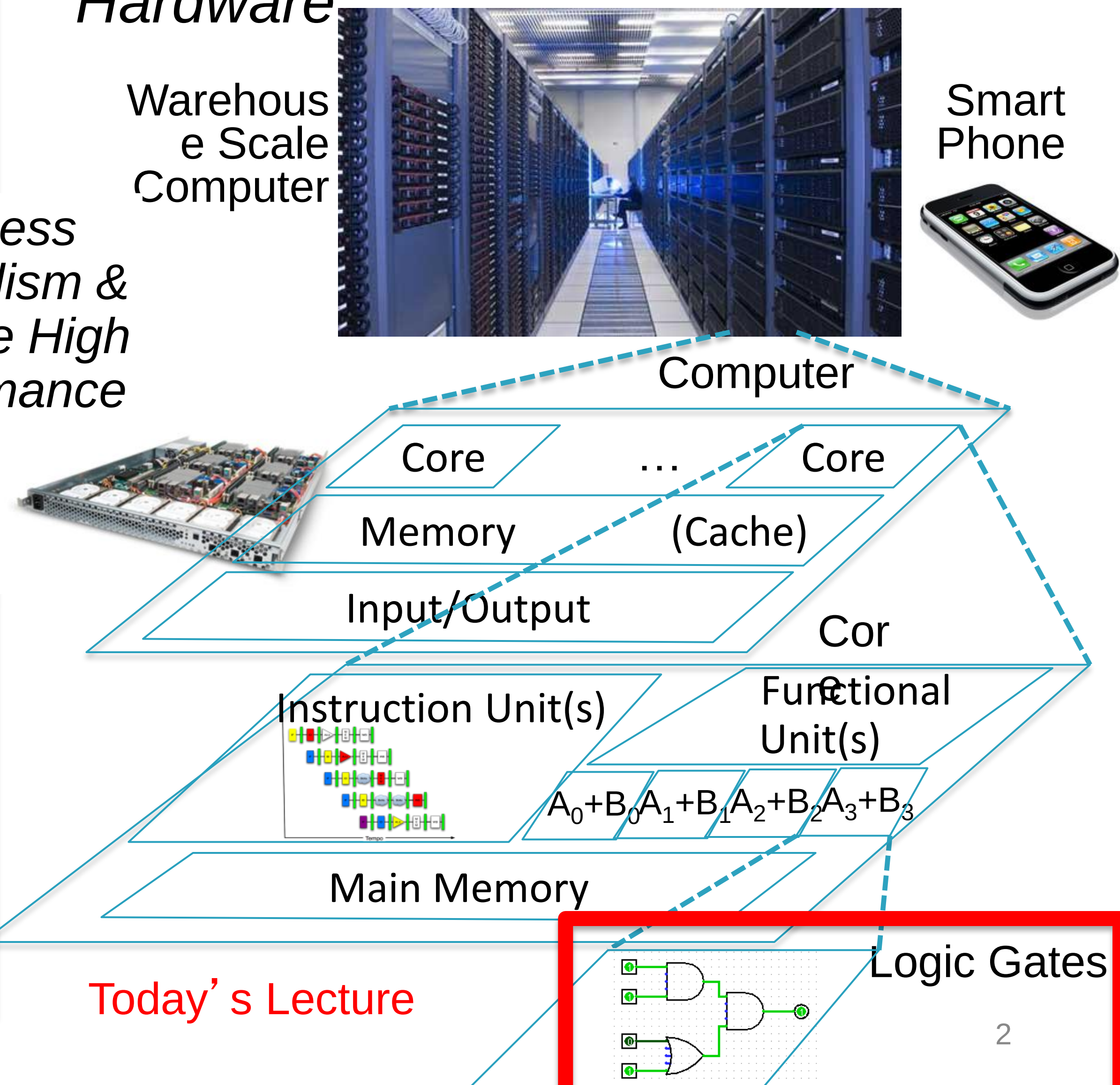
[Adapted from lectures on
***Computer Organization and Design*,**
Patterson & Hennessy, 5th edition, 2013]

New-School Machine Structures (It's a bit more complicated!)

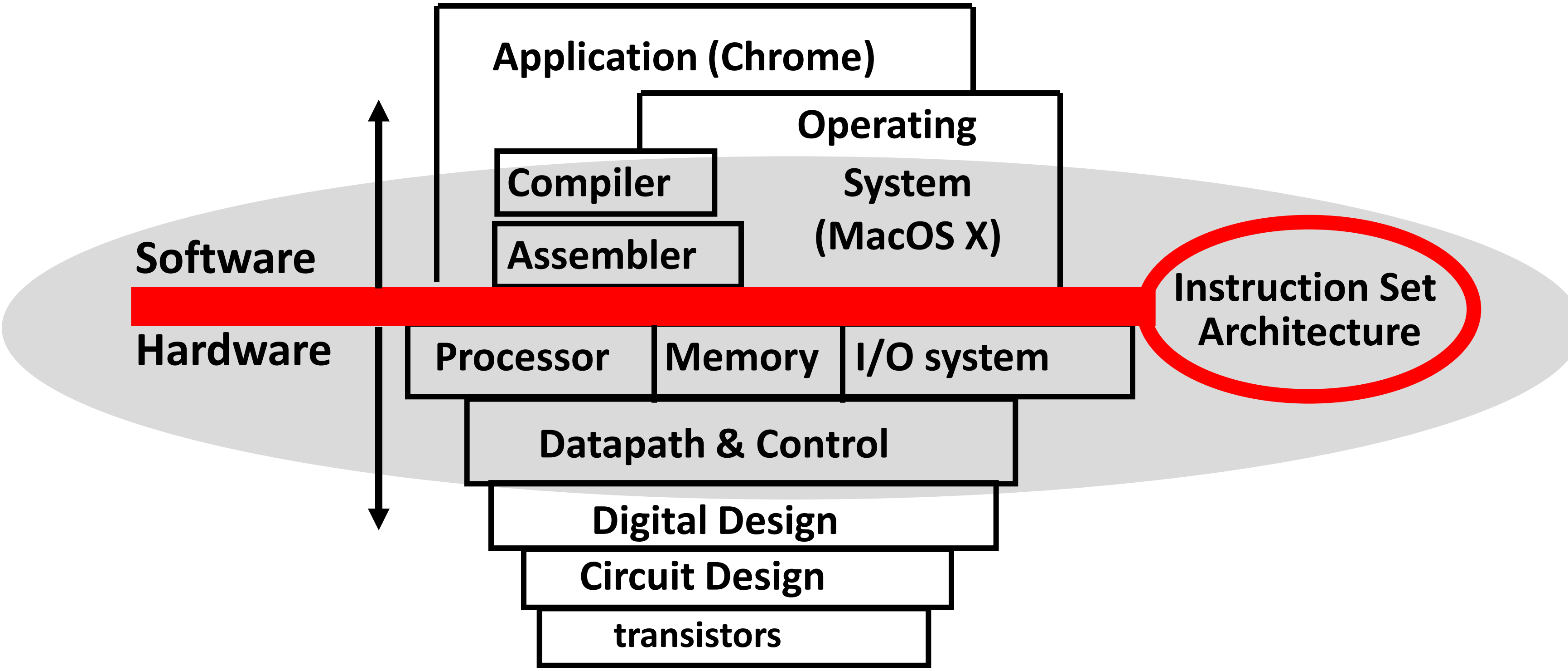
Software | *Hardware*

- **Parallel Requests**
  Assigned to computer
  e.g., Search "Garcia"
- **Parallel Threads**
  Assigned to core
  e.g., Lookup, Ads
- **Parallel Instructions**
  >1 instruction @ one time
  e.g., 5 pipelined instructions
- **Parallel Data**
  >1 data item @ one time
  e.g., Add of 4 pairs of words
- **Hardware descriptions**
  All gates @ one time

Harness Parallelism & Achieve High Performance



What is Machine Structures?



Coordination of many *levels of abstraction*

ISA is an important abstraction level: contract between HW & SW

Levels of Representation/Interpretation

High Level Language Program (e.g., C)

Compiler

Assembly Language Program (e.g., MIPS)

Assembler

Machine Language Program (MIPS)

Machine Interpretation

Hardware Architecture Description (e.g., block diagrams)

Architecture Implementation

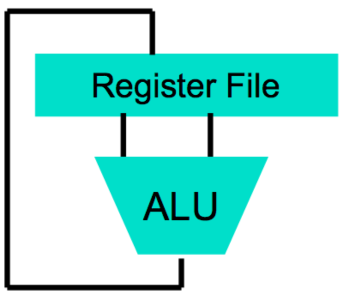
Logic Circuit Description (Circuit Schematic Diagrams)

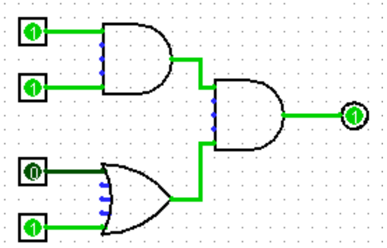
```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw    $t0, 0($2)
lw    $t1, 4($2)
sw    $t1, 0($2)
sw    $t0, 4($2)
```

Anything can be represented as a *number*, i.e., data or instructions

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```





Synchronous Digital Systems

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

***Synchronous*:**

- All operations coordinated by a central clock
  - "Heartbeat" of the system!

***Digital*:**

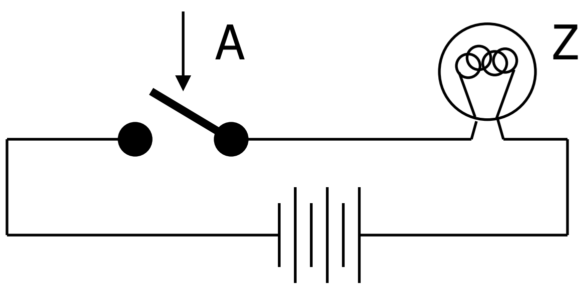
- All values represented by discrete values
- Electrical signals are treated as 1s and 0s; grouped together to form words

Logic Design

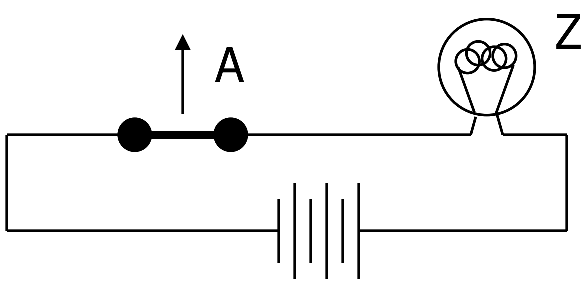
- Next several weeks: we will study how a modern processor is built; starting with basic elements as building blocks
- **Why study hardware design?**
  - Understand capabilities and limitations of hw in general and processors in particular
  - What processors can do fast and what they can't do fast (avoid slow things if you want your code to run fast!)
  - Background for more in depth hw studies for your interest
  - There is just so much you can do with standard processors: you may need to design own custom hw for extra performance
- **Logism,** an educational tool for designing and simulating digital logic circuits
  - http://www.cburch.com/logisim/

Switches: Basic Element of Physical Implementations

- **Implementing a simple circuit (arrow shows action if wire changes to "1"):**



Close switch (if *A* is "1" or asserted) and turn on light bulb (Z)



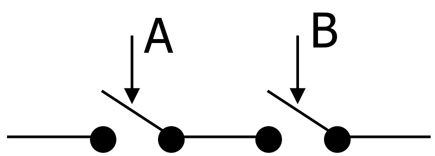
Open switch (if *A* is "0" or unasserted) and turn off light bulb (Z)

$$Z \equiv A$$

Switches (cont’d)

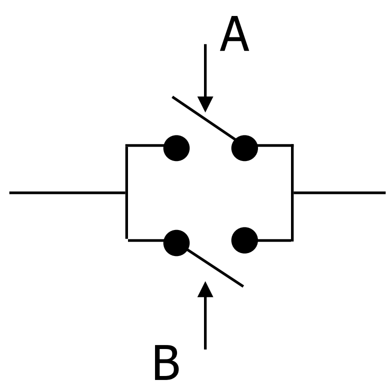
- **Compose switches into more complex ones (Boolean functions):**

AND



Z ≡ A and B

OR



Z ≡ A or B

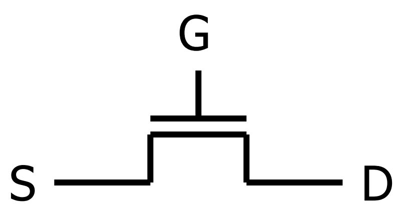
Transistor Networks

- **Modern digital systems designed in CMOS**
  - MOS: Metal-Oxide on Semiconductor
  - C for complementary: normally-open and normally-closed switches

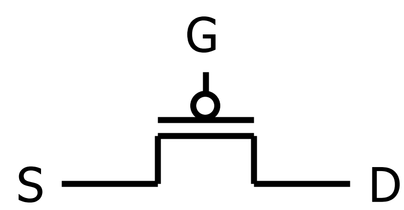
- **MOS transistors act as voltage-controlled switches** http://youtu.be/ZaBLiciesOU

MOS Transistors

- **Three terminals: drain, gate, and source**
  - Switch action:
    if voltage on gate terminal is (some amount) higher/lower than source terminal then conducting path established between drain and source terminals



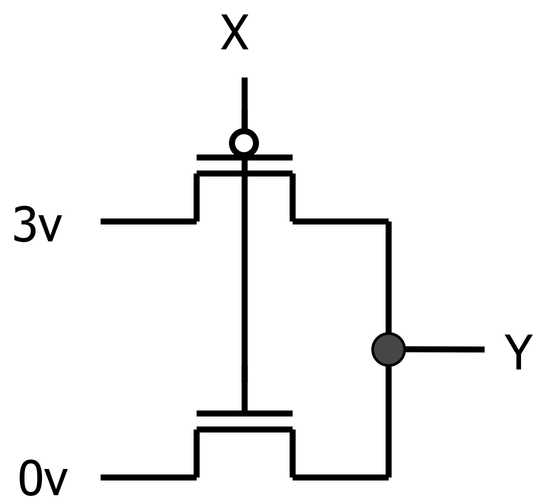
n-channel
open when voltage at G is low
closes when:
voltage(G) > voltage (S) + ε



p-channel
closed when voltage at G is low
opens when:
voltage(G) < voltage (S) – ε

MOS Networks

"1" (voltage source)



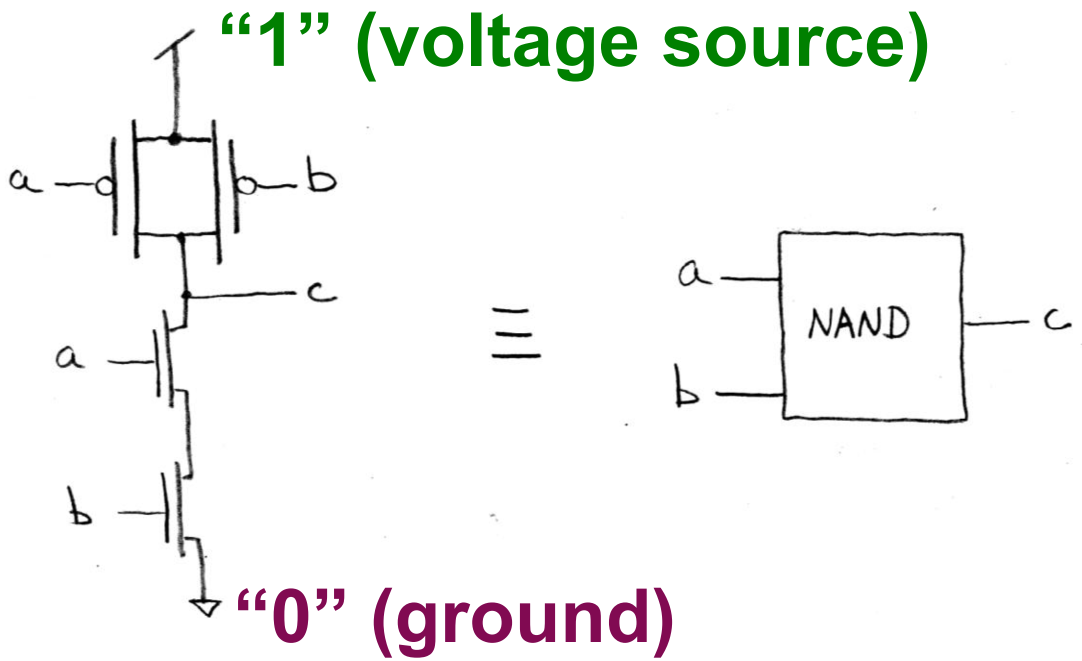
"0" (ground)

what is the relationship between x and y?

x	y
0 volts	
3 volts	

Transistor Circuit Rep. vs. Block diagram

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

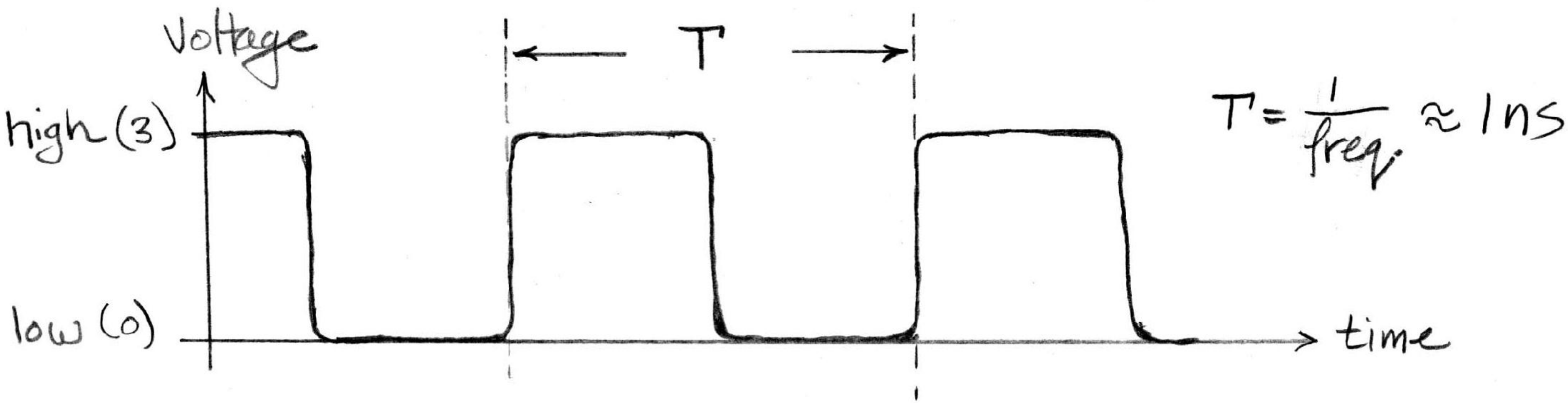


a	b	c
0	0	1
0	1	1
1	0	1
1	1	0

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

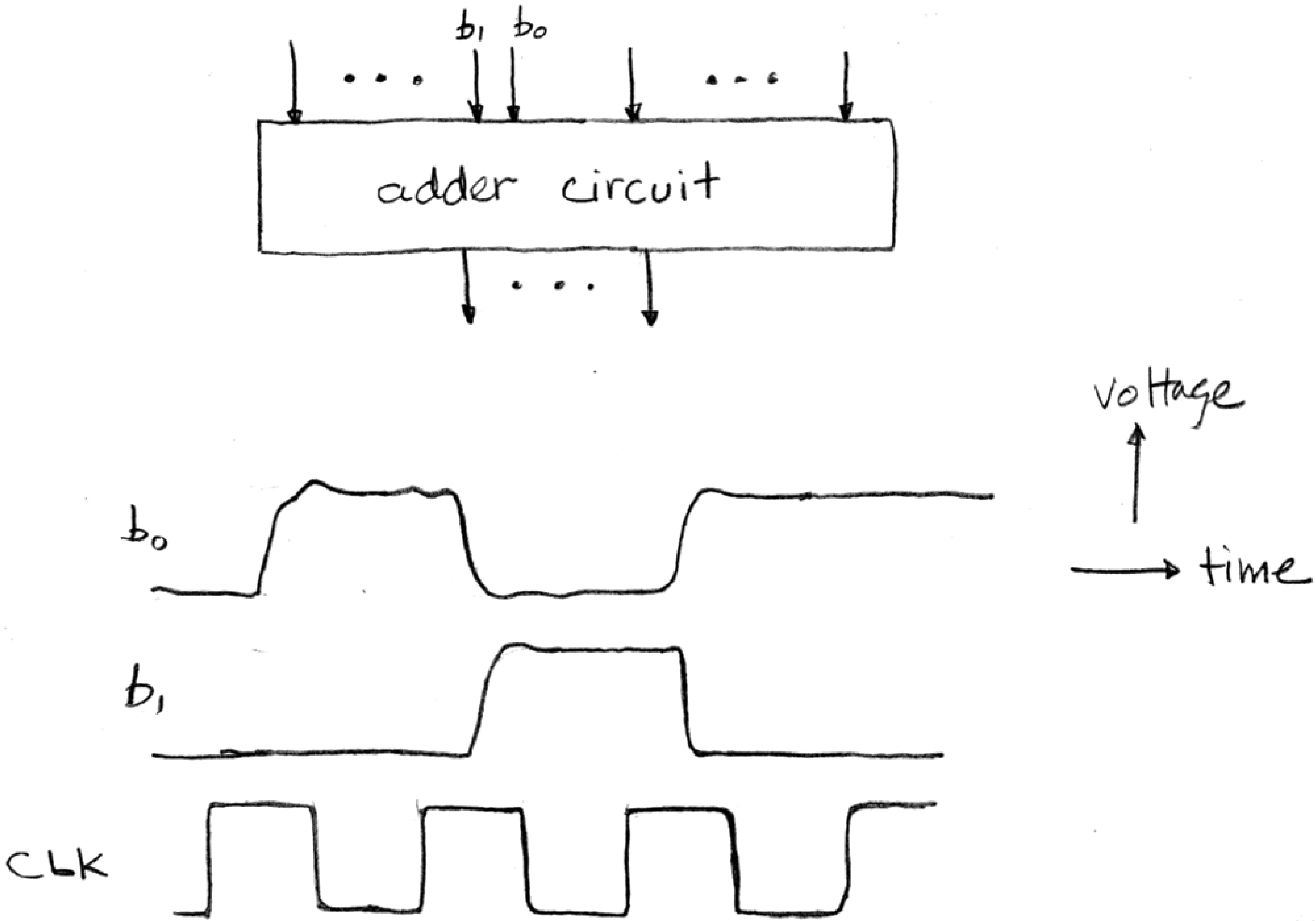
(You can build AND, OR, NOT out of NAND!)

Signals and Waveforms: Clocks

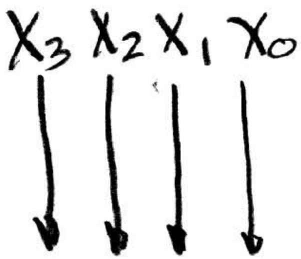


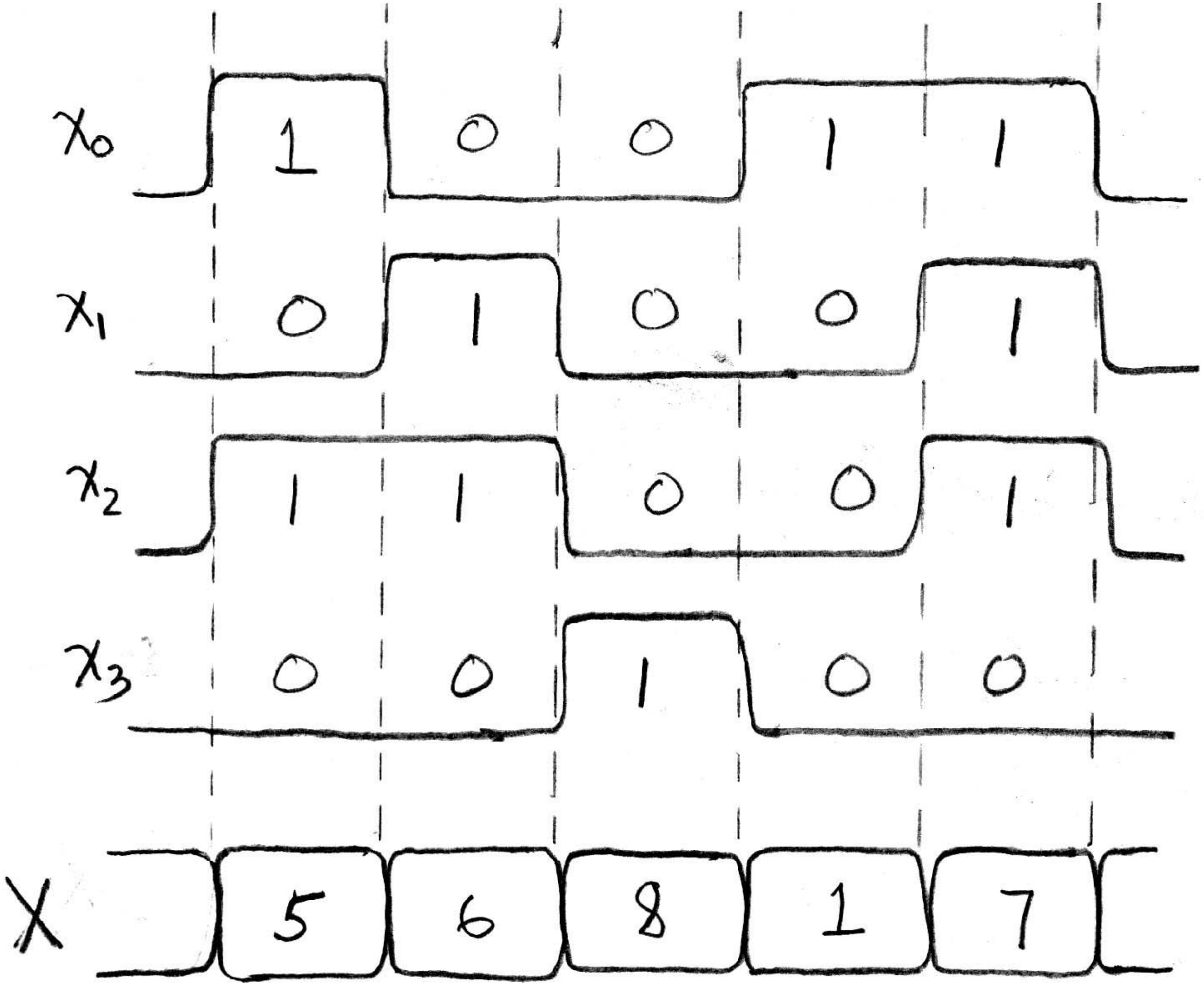
- **Signals**
  - **When digital is only treated as 1 or 0**
  - **Is transmitted over wires continuously**
  - **Transmission is effectively instant**
    - **Implies that any wire only contains 1 value at a time**

Signals and Waveforms

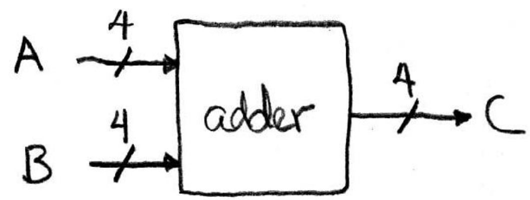


Signals and Waveforms: Grouping



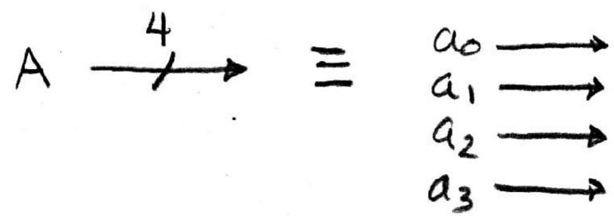


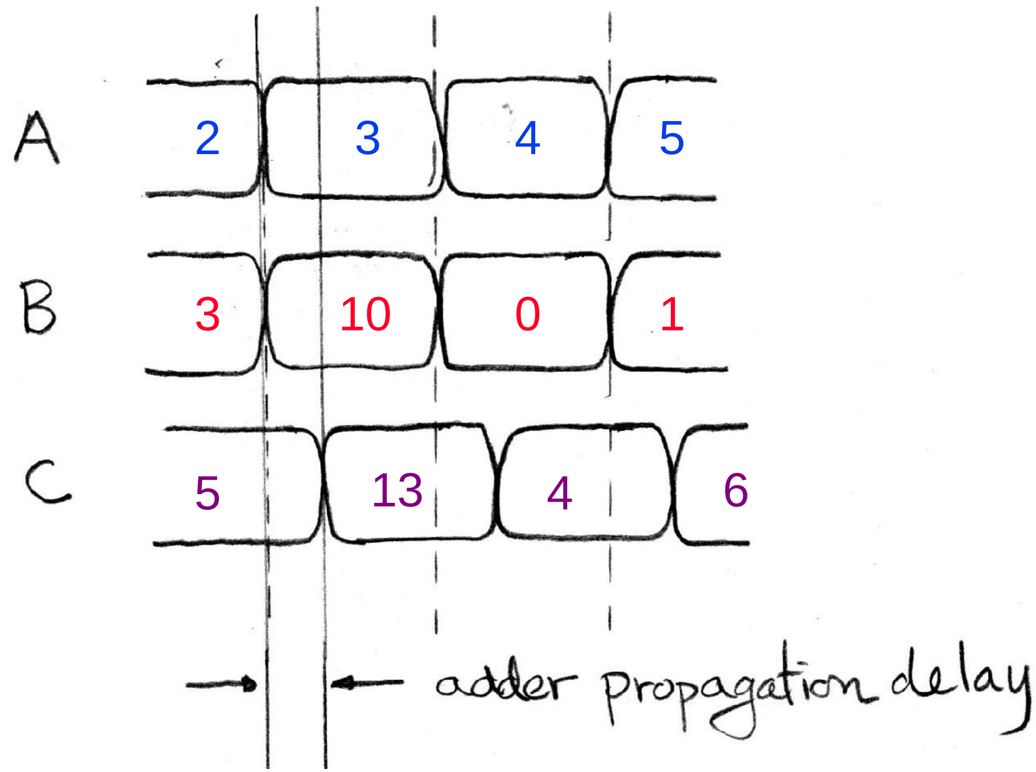
Signals and Waveforms: Circuit Delay



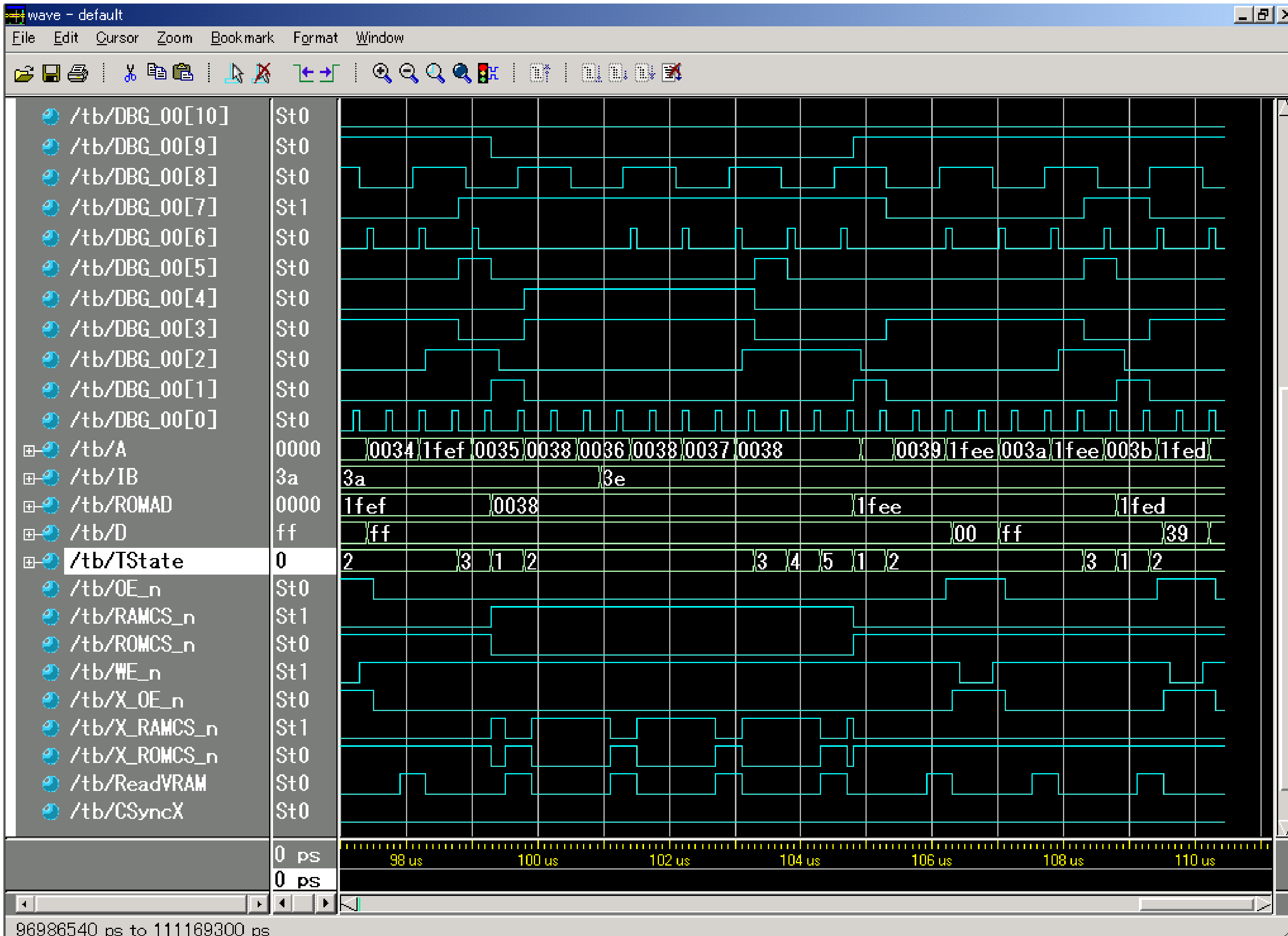
$$A = [a_3, a_2, a_1, a_0]$$

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





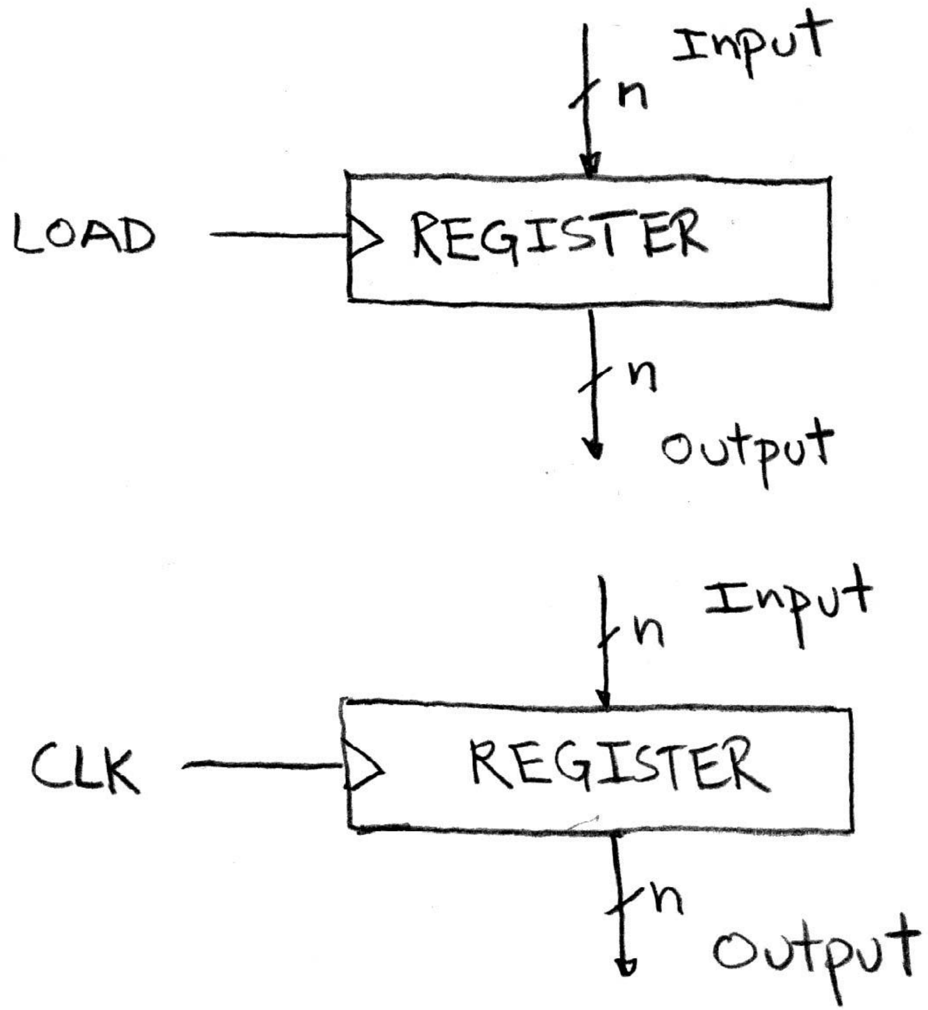
Sample Debugging Waveform



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)



And in conclusion…

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