CS 110
Computer Architecture
Lecture 10:
Finite State Machines, Functional Units

Instructors:
Sören Schwertfeger & Chundong Wang

https://robotics.shanghaitech.edu.cn/courses/ca/20s/

School of Information Science and Technology SIST

ShanghaiTech University

Slides based on UC Berkley's CS61C

Levels of Representation/Interpretation

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

Compiler

Assembly Language Program (e.g., RISC-V)

Machine Language Program (RISC-V)

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

```
lw    xt0, 0(x2)
lw    xt1, 4(x2)
sw    xt1, 0(x2)
sw    xt0, 4(x2)
```

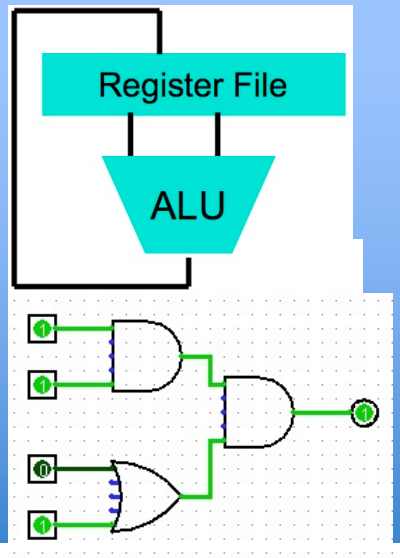
```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

Machine Interpretation

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

Architecture Implementation

Logic Circuit Description (Circuit Schematic Diagrams)

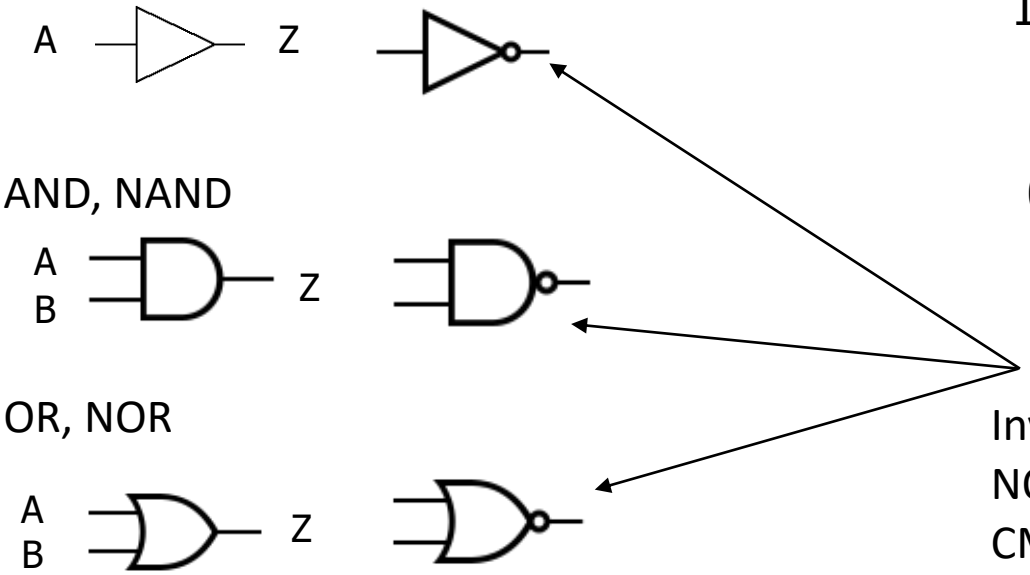


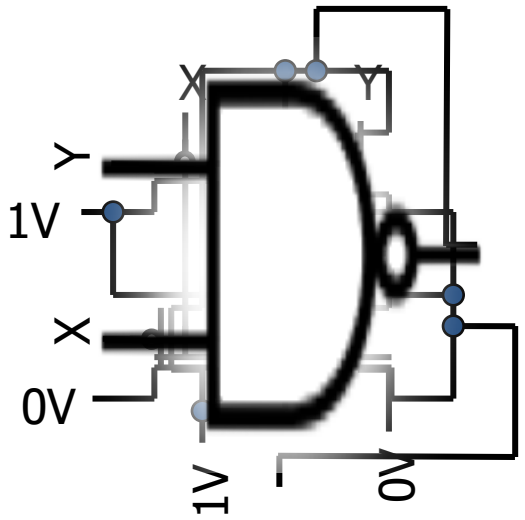
Outline

- Timing and Critical Path
- Finite State Machine
- Admin
- Functional Units
  - Multiplexer
  - ALU
  - Adder/ Subtractor

Combinational Logic Symbols

- Common combinational logic systems have standard symbols called logic gates
  - Buffer, NOT
  - AND, NAND
  - OR, NOR



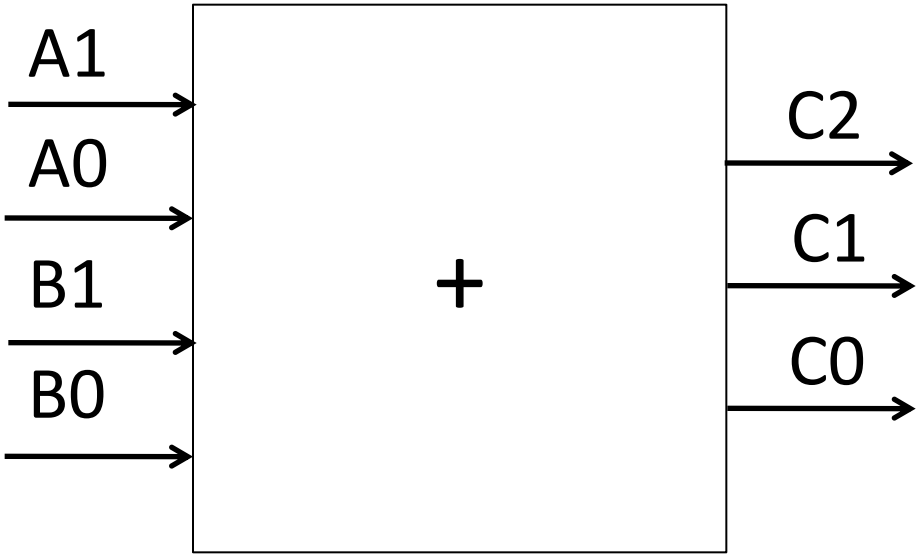


Inverting versions (NOT, NAND, NOR) easiest to implement with CMOS transistors (the switches we have available and use most)

Truth Table Example #2: 2-bit Adder

A	B	C
a_1a_0	b_1b_0	$c_2c_1c_0$

How Many Rows?

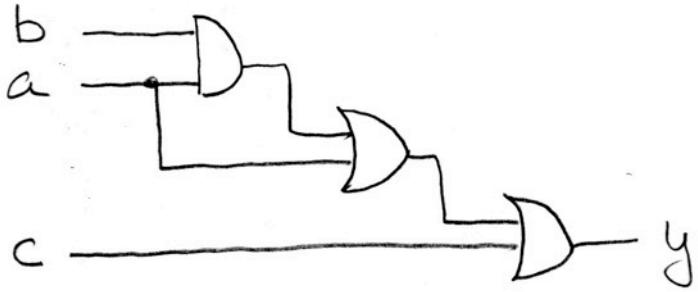


Truth Table Example #3: 32-bit Unsigned Adder

A	B	C
000 ... 0	000 ... 0	000 ... 00
000 ... 0	000 ... 1	000 ... 01
.	.	.
.	.	.
.	.	.
111 ... 1	111 ... 1	111 ... 10

How Many Rows?

Boolean Algebra: Circuit & Algebraic Simplification



original circuit

$$\downarrow$$

$$y = ((ab) + a) + c$$

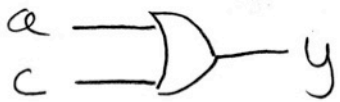
equation derived from original circuit

$$\downarrow$$

$$= ab + a + c$$
$$= a(b + 1) + c$$
$$= a(1) + c$$
$$= a + c$$

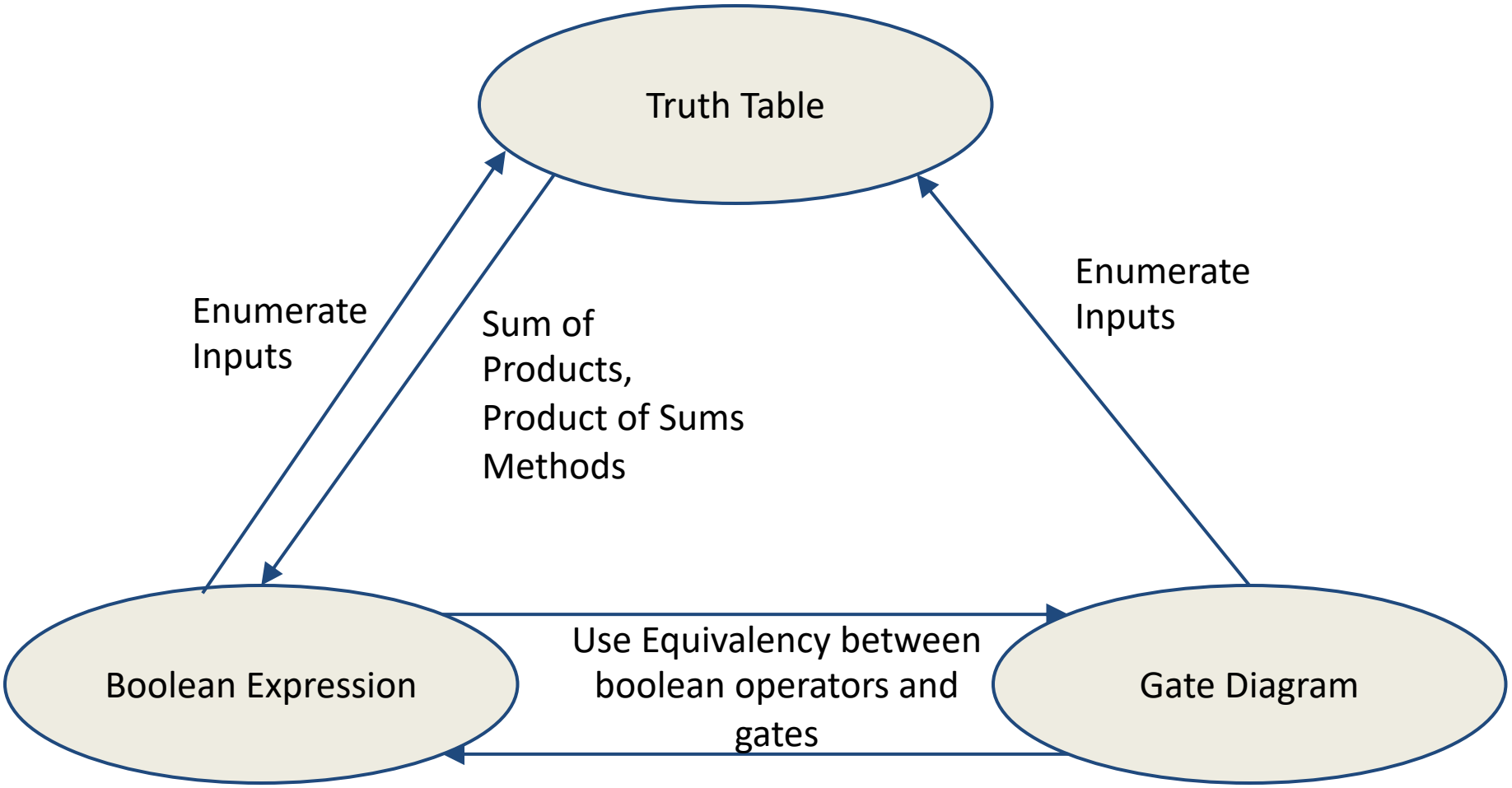
algebraic simplification

$$\downarrow$$



simplified circuit

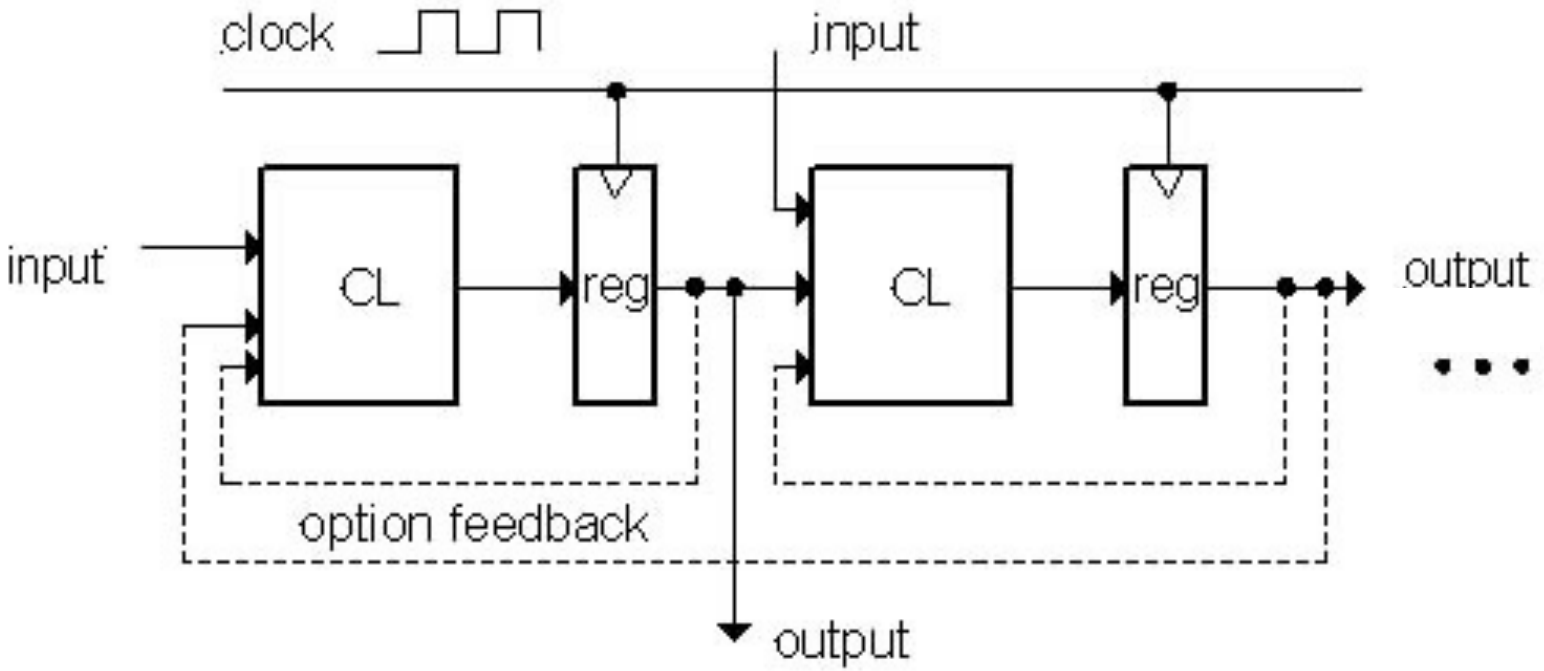
Representations of Combinational Logic (groups of logic gates)



Type of Circuits

- *Synchronous Digital Systems* consist of two basic types of circuits:
  - Combinational Logic (CL) circuits
    - Output is a function of the inputs only, not the history of its execution
    - E.g., circuits to add A, B (ALUs)
  - Sequential Logic (SL)
    - Circuits that "remember" or store information
    - aka "State Elements"
    - E.g., memories and registers (Registers)

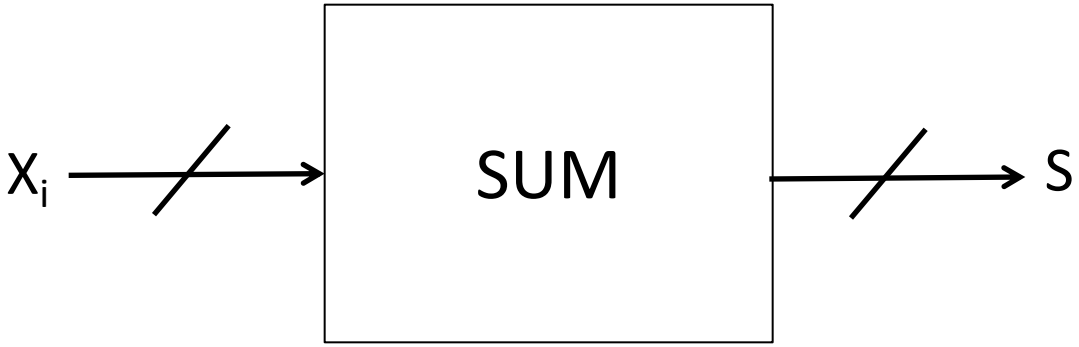
Model for Synchronous Systems



- Collection of Combinational Logic blocks separated by registers
- Feedback is optional
- Clock signal(s) connects only to clock input of registers
- Clock (CLK): steady square wave that synchronizes the system
- Register: several bits of state that samples on rising edge of CLK (positive edge-triggered) or falling edge (negative edge-triggered)

Accumulator Example

Why do we need to control the flow of information?



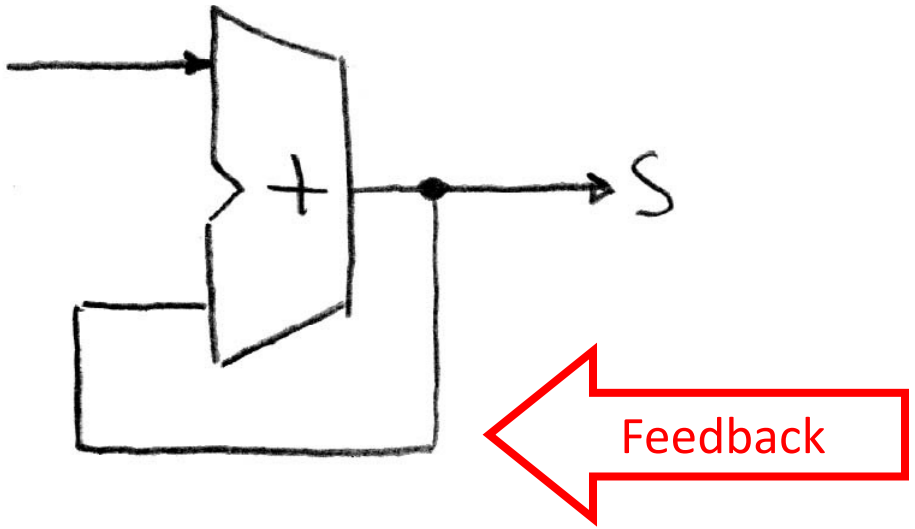
Want:

```
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?

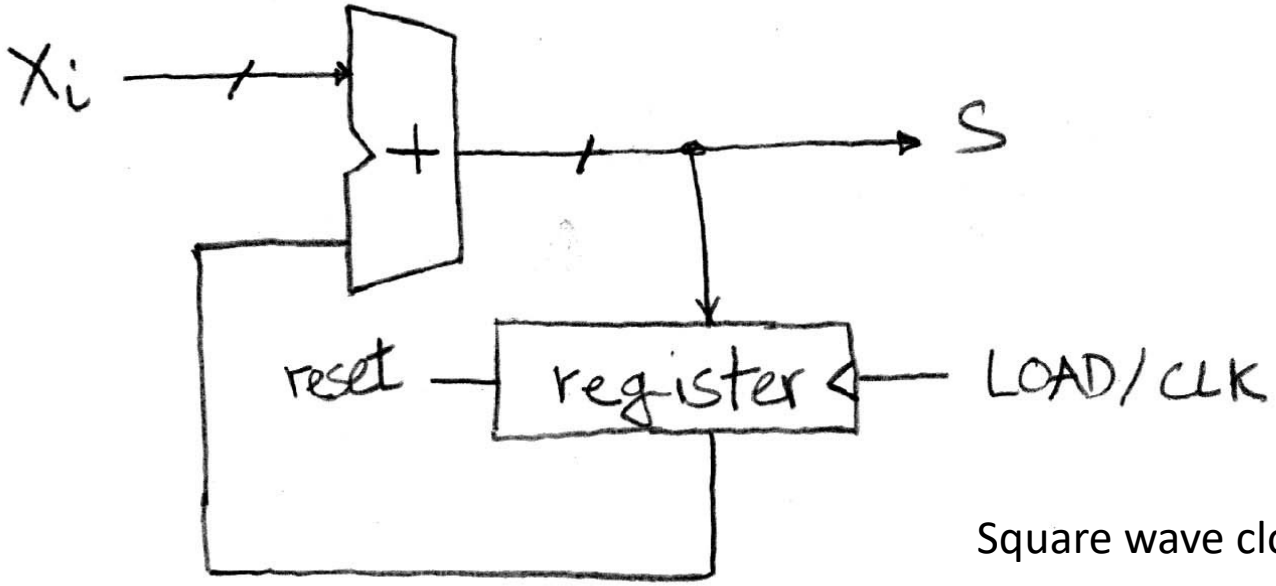


No!

Reason #1: How to control the next iteration of the ‘for’ loop?

Reason #2: How do we say: ‘S=0’?

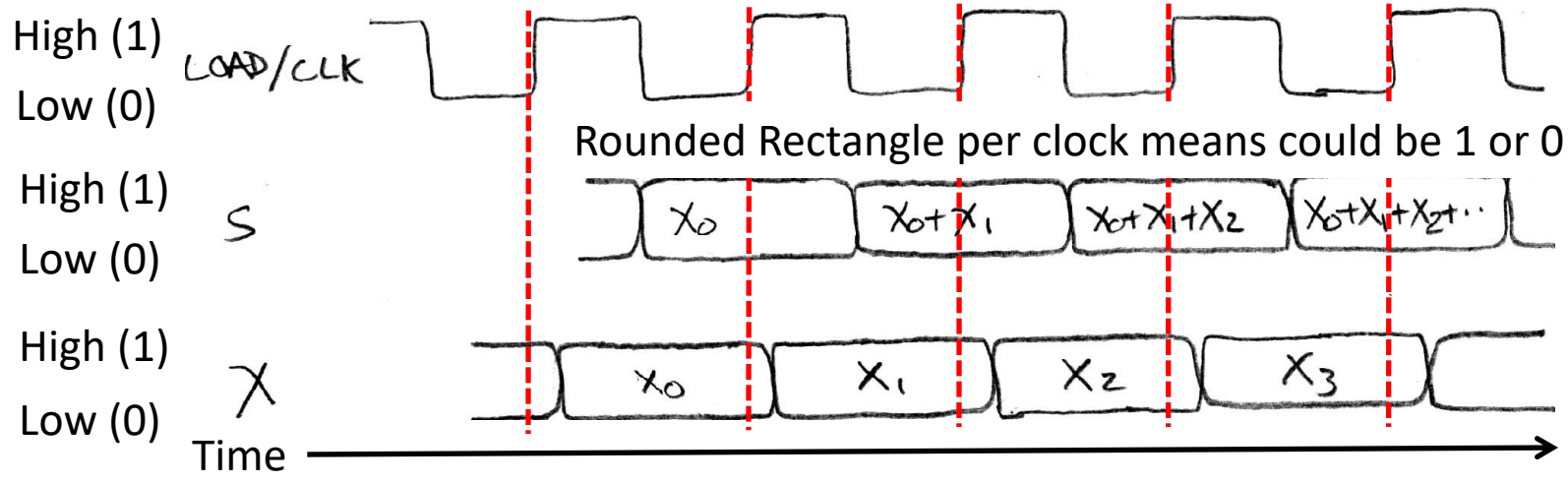
Second Try: How About This?



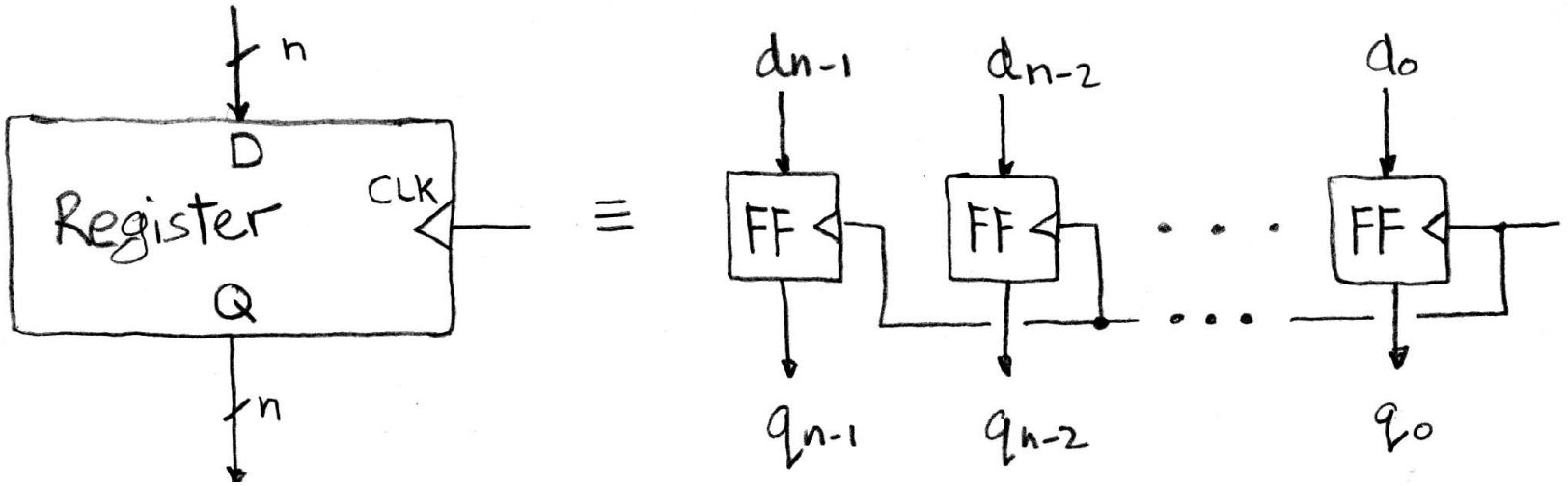
Register is used to hold up the transfer of data to adder

Square wave clock sets when things change

Rough timing …



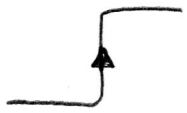
Register Internals



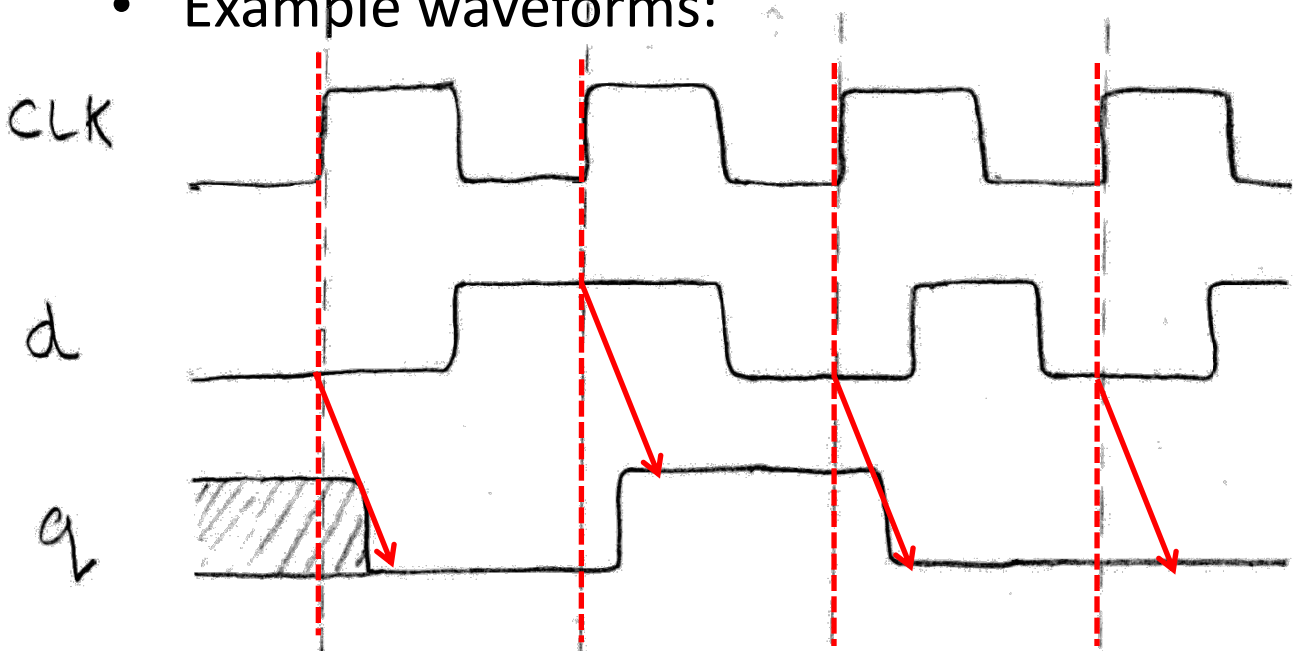
- n instances of a "Flip-Flop"
- Flip-flop name because the output flips and flops between 0 and 1
- D is "data input", Q is "data output"
- Also called "D-type Flip-Flop"

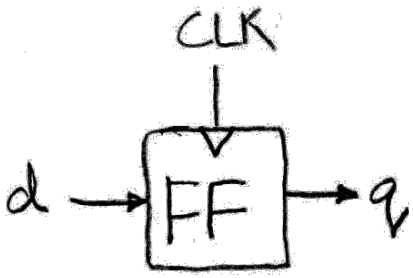
Flip-Flop Operation

- 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:

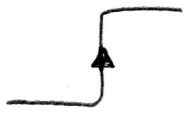




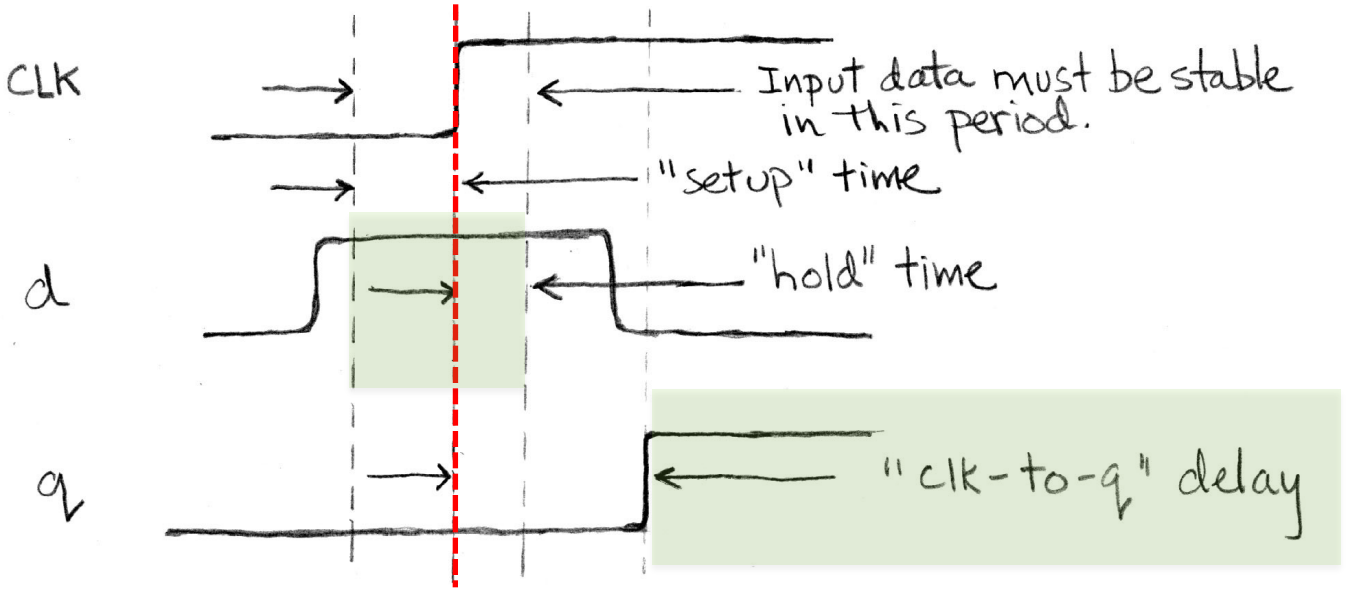
How a flip flop works: https://www.youtube.com/watch?v=-aQH0ybMd3U

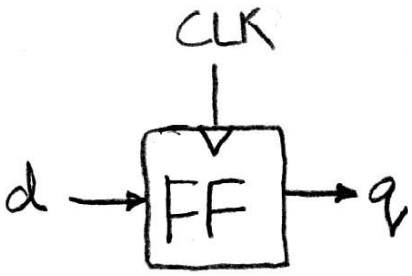
Flip-Flop Timing

- 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 (more detail):

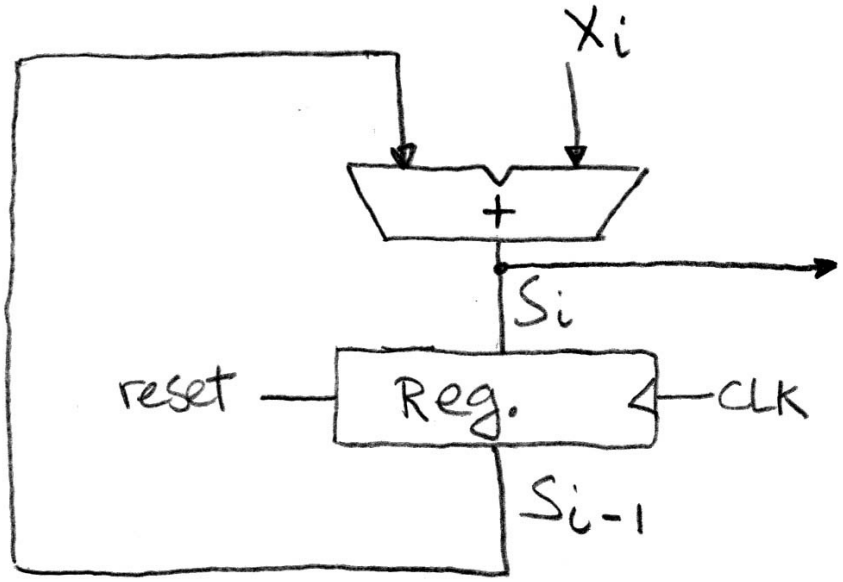




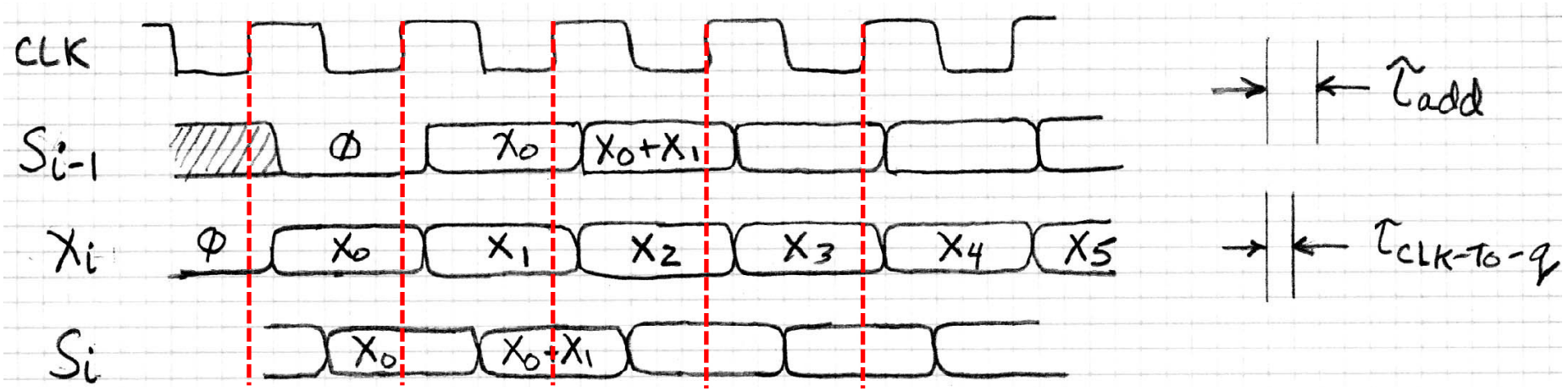
Hardware Timing Terms

- Setup Time: when the input must be stable *before* the edge of the CLK
- Hold Time: when the input must be stable *after* the edge of the CLK
- "CLK-to-Q" Delay: how long it takes the output to change, measured from the edge of the CLK

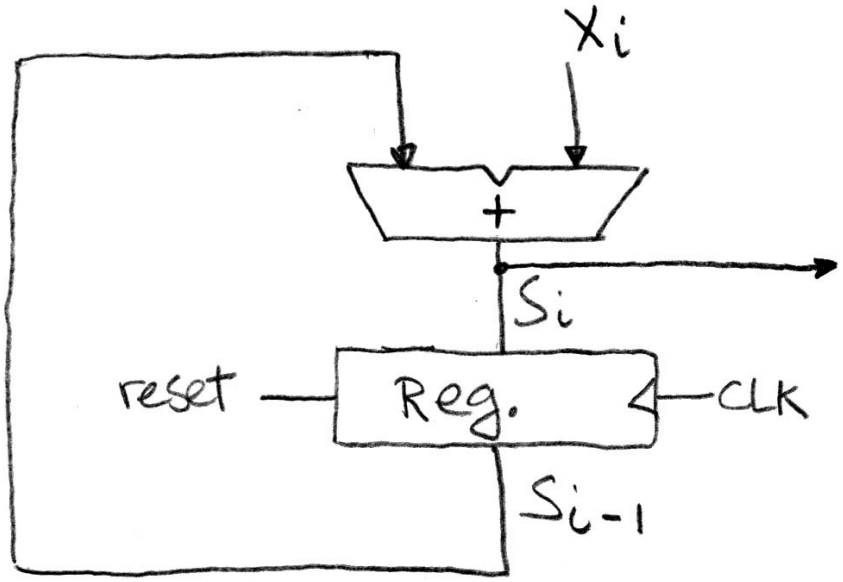
Accumulator Timing 1/2



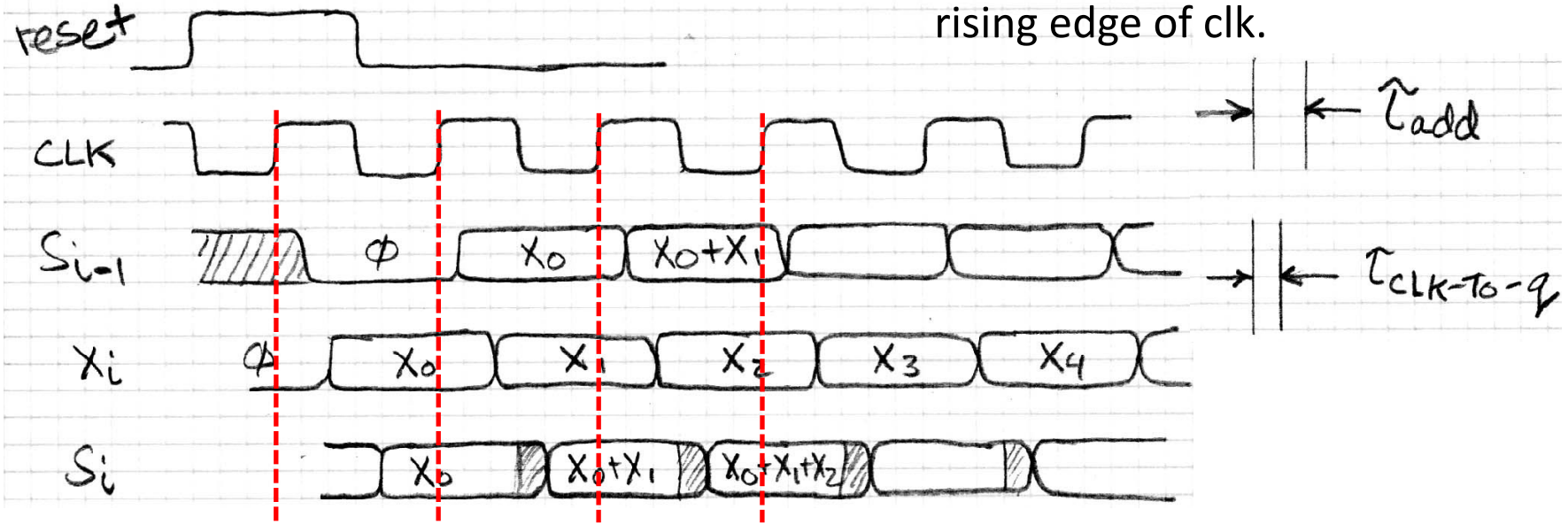
- Reset input to register is used to force it to all zeros (takes priority over D input).
- S_{i-1} holds the result of the i^{th}-1 iteration.
- Analyze circuit timing starting at the output of the register.



Accumulator Timing 2/2

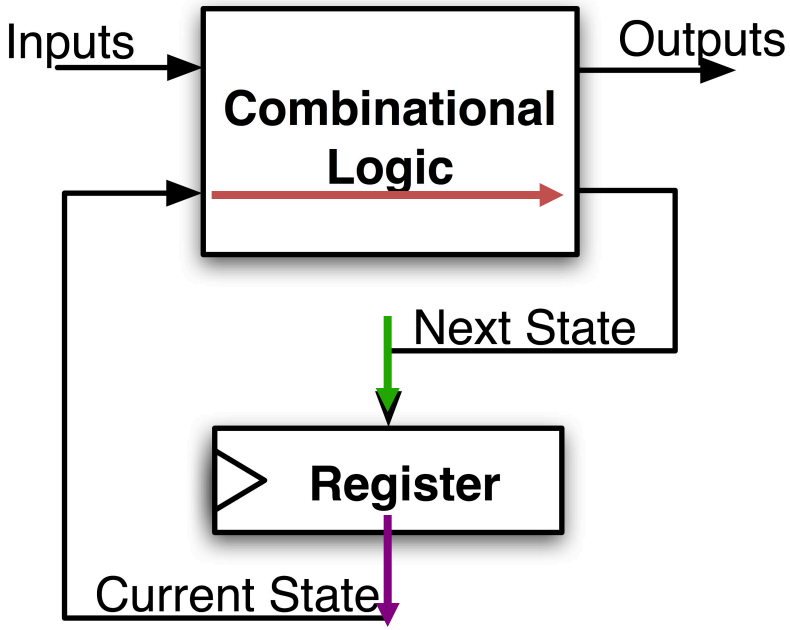


- reset signal shown.
- Also, in practice X might not arrive to the adder at the same time as S_{i-1}
- S_i temporarily is wrong, but register always captures correct value.
- In good circuits, instability never happens around rising edge of clk.



Maximum Clock Frequency

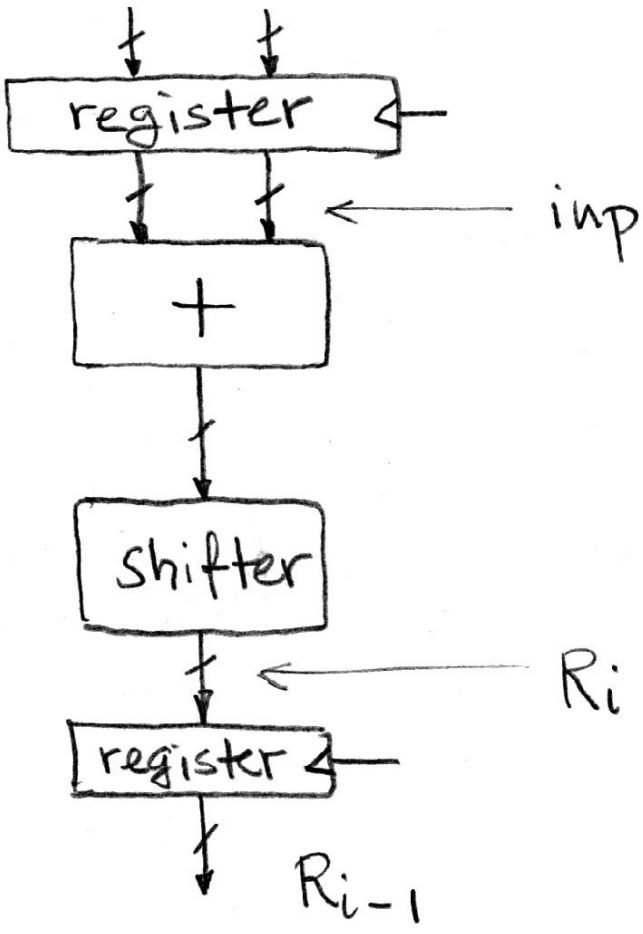
- What is the maximum frequency of this circuit?



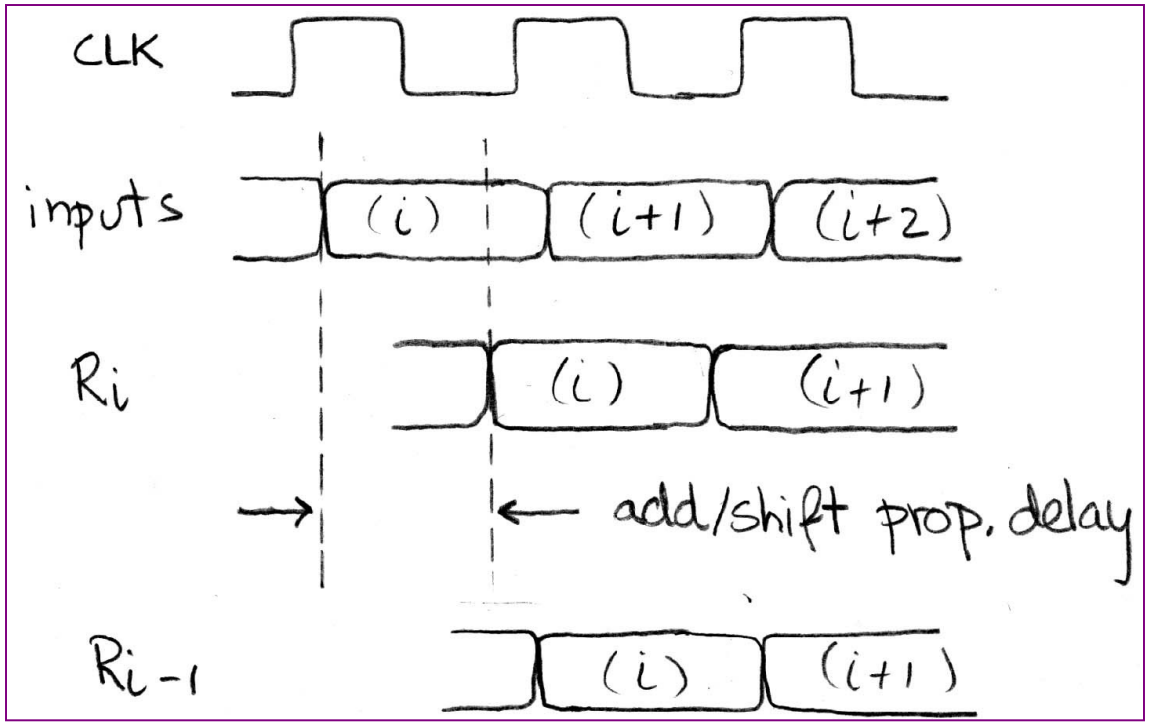
Hint:
Frequency = 1/Period

Max Delay = CLK-to-Q Delay + CL Delay + Setup Time

Critical Paths



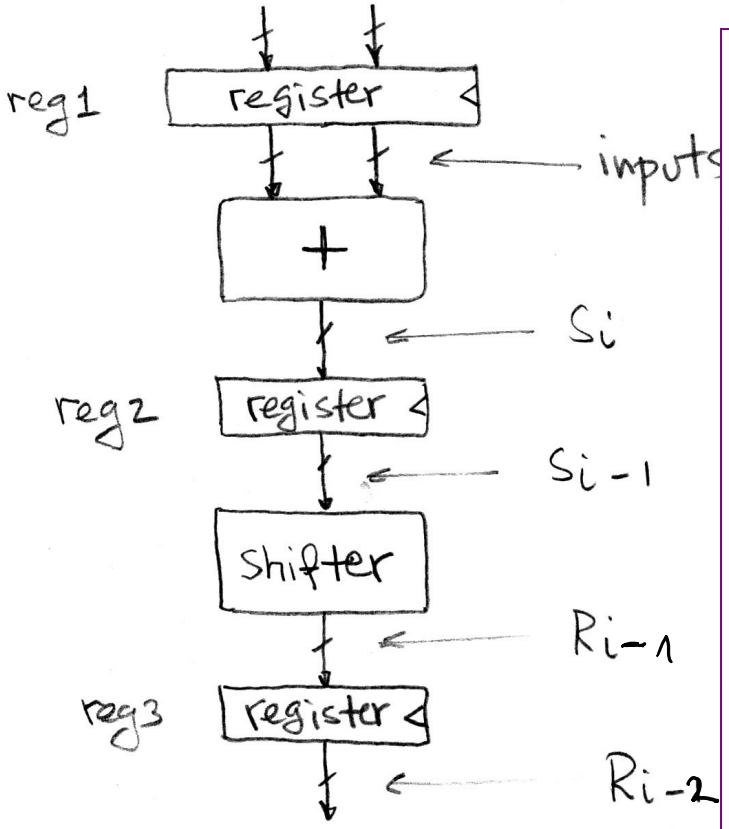
Timing…

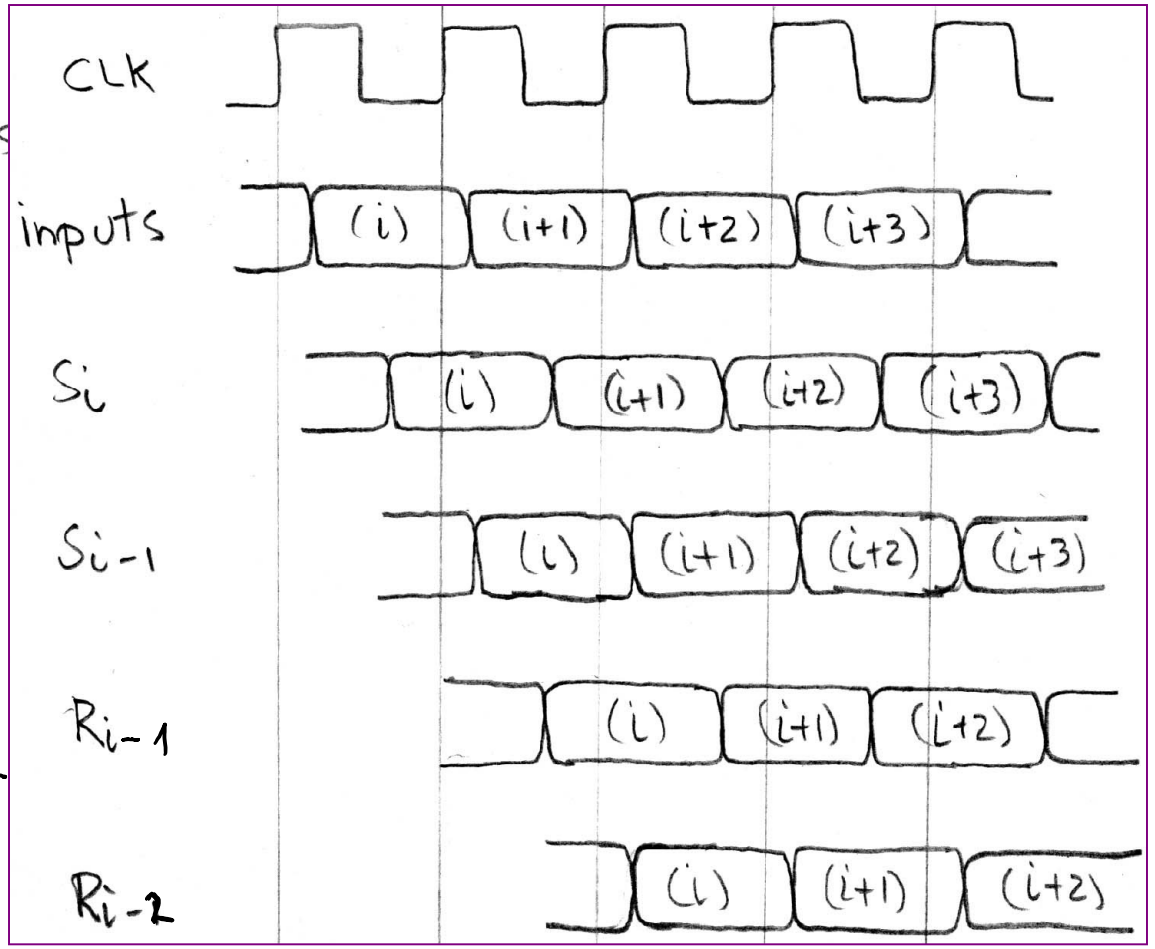


Note: delay of 1 clock cycle from input to output.
Clock period limited by propagation delay of adder/shifter.

Pipelining to improve performance

Timing…





- Insertion of register allows higher clock frequency.
- More outputs per second (higher bandwidth)
- But each individual result takes longer (greater latency)

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 of the CLK

- Flip-flop - one bit of state that samples every rising edge of the CLK (positive edge-triggered)
- Register - several bits of state that samples on rising edge of CLK or on LOAD (positive edge-triggered)

Problems with Clocking

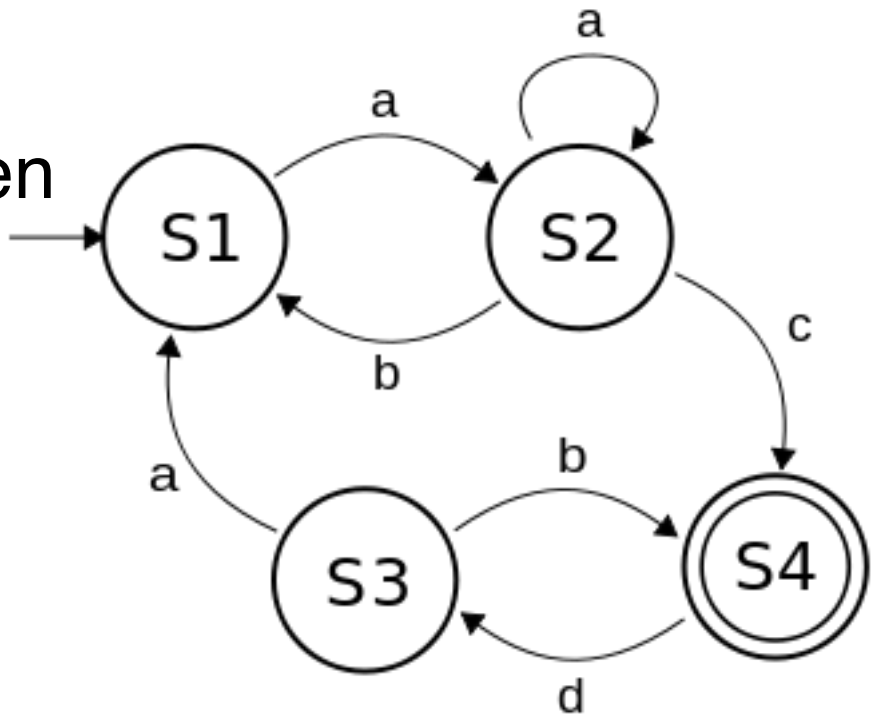
- The clock period ***must be*** longer than the critical path
  - Otherwise, you will get the wrong answers
  - But it can be even longer than that
- Critical path:
  - clk->q time
  - Necessary to get the output of the registers
  - ***worst case*** combinational logic delay
  - ***Setup time*** for the next register
- Must meet all of these to be correct

Hold-Time Violations...

- An alternate problem can occur...
  - Clk->Q + ***best case*** combinational delay < Hold time...
- What happens?
  - Clk->Q + data propagates...
  - And now you don't hold the input to the flip flop long enough
- Solution:
  - ***Add*** delay on the best-case path (e.g. two inverters)

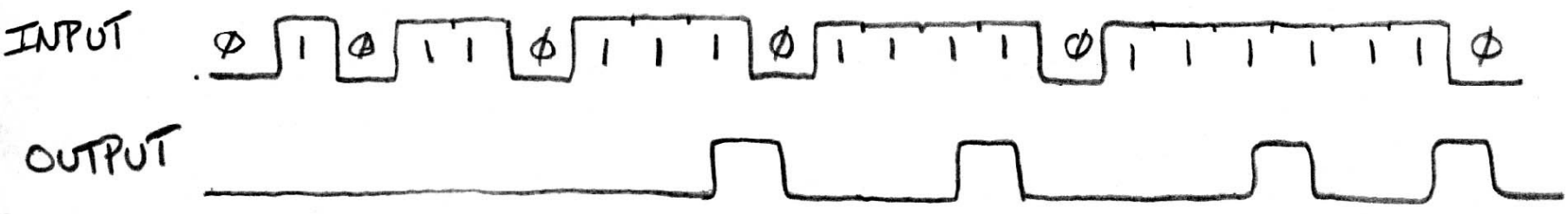
Finite State Machines (FSM) Intro

- A convenient way to conceptualize computation over time
- We start at a state and given an input, we follow some edge to another (or the same) state
- The function can be represented with a “state transition diagram”.
- With combinational logic and registers, any FSM can be implemented in hardware.

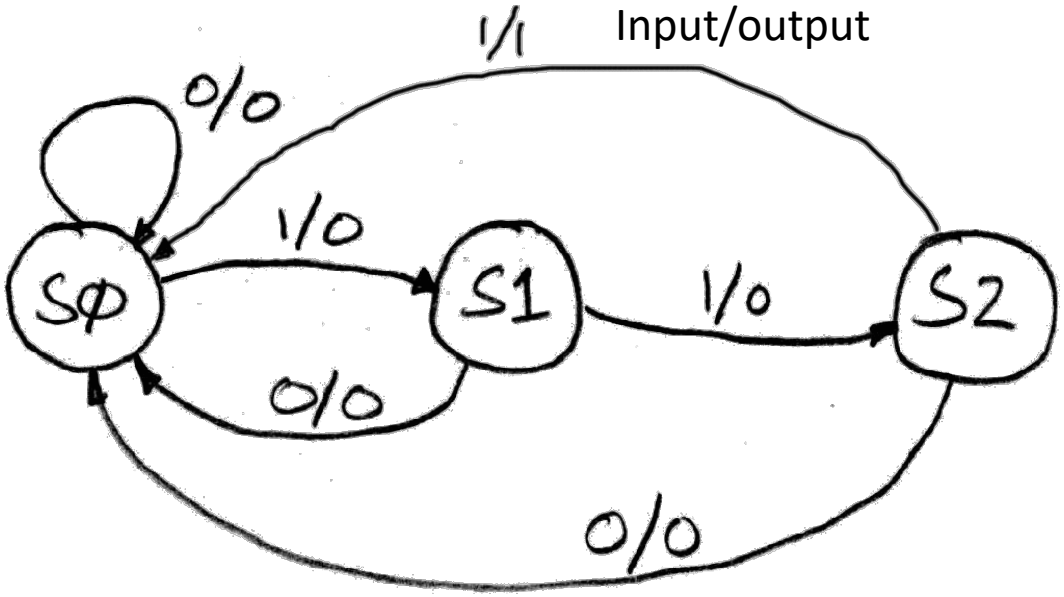


FSM Example: 3 ones…

FSM to detect the occurrence of 3 consecutive 1's in the input.



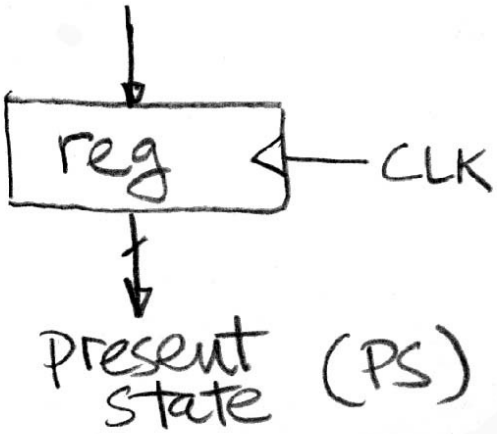
Draw the FSM…



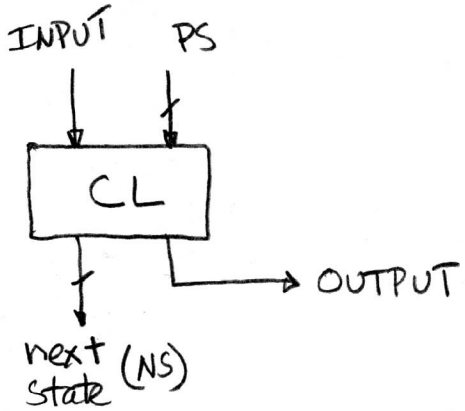
Assume state transitions are controlled by the clock: on each clock cycle the machine checks the inputs and moves to a new state and produces a new output…

Hardware Implementation of FSM

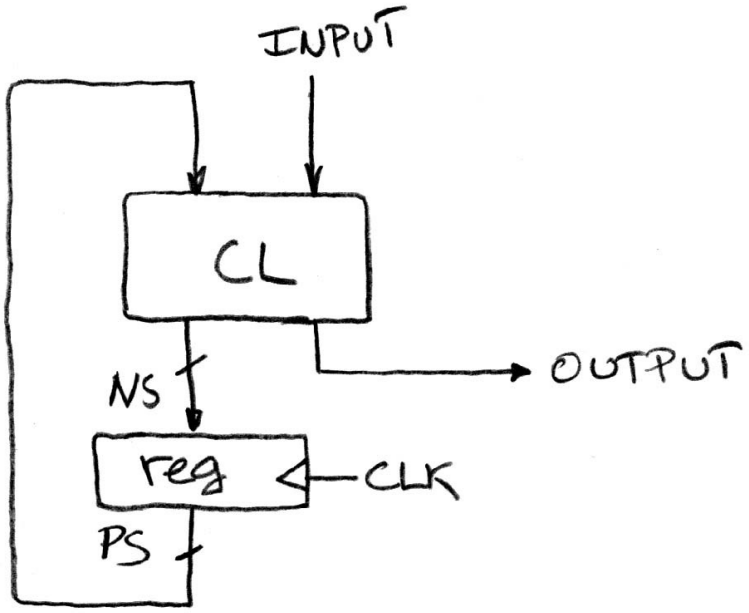
… Therefore a register is needed to hold the a representation of which state the machine is in. Use a unique bit pattern for each state.



+



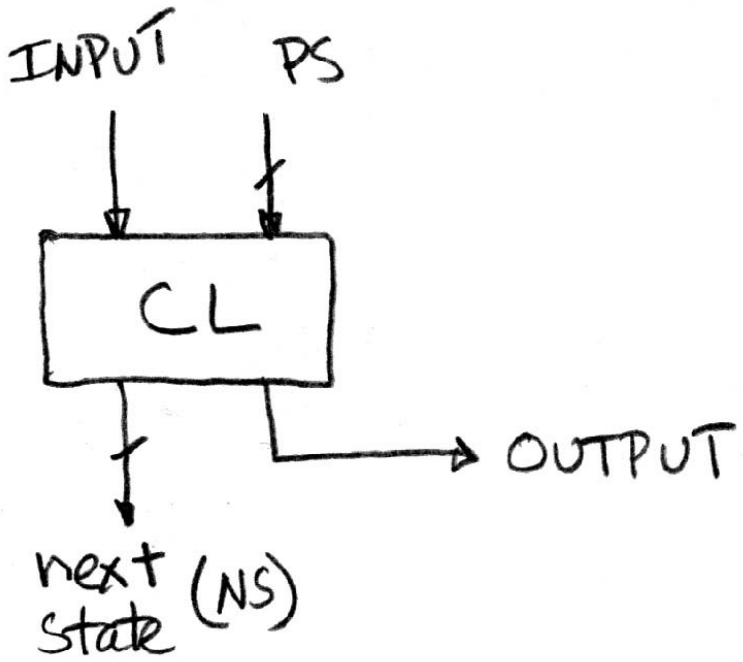
=



Combinational logic circuit is used to implement a function that maps from *present state and input* to *next state and output.*

FSM Combinational Logic

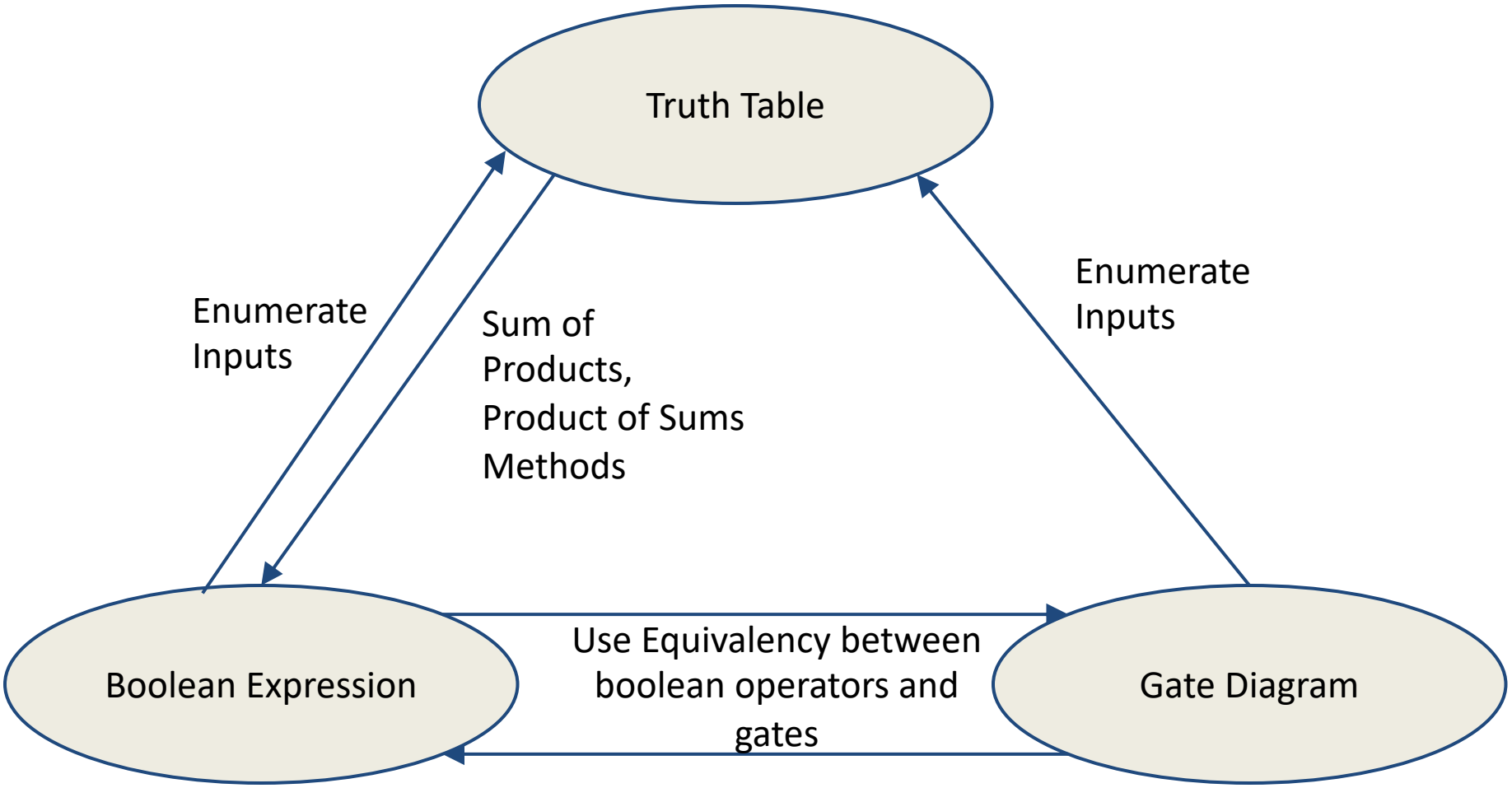
Specify CL using a truth table



Truth table…

PS	Input	NS	Output
00	0	00	0
00	1	01	0
01	0	00	0
01	1	10	0
10	0	00	0
10	1	00	1

Representations of Combinational Logic (groups of logic gates)



Admin

- P1.1 due tomorrow
- P1.2 will be published this week

- HW3 is out

- Midterm I
  - April 6 during lecture hours
- Contents:
  - Everything till (including) Datapath (next lecture)

Midterm I

- Switch cell phones **off**!
  (not silent mode – off!)
  - Put them in your bags.
- Bags in the front. On the table: nothing but:
  pen, 1 drink, 1 snack, your student ID card and your cheat sheet!
- The RISC V green card will be provided
- No other electronic devices are allowed!
  - No ear plugs, music, smartwatch…
- Anybody touching any electronic device will **FAIL** the course!
- Anybody found cheating (copy your neighbors answers, additional material, ...) will **FAIL** the course!


CASIO
JF-120TM
TWO WAY POWER
12 DIGITS
F CUT 5/4
4 3 2 1 0 A
MRC
TAX−
TAX RATE
TAX+
RATE SET
%
GT
ON
AC
C
+/−
7
8
9
4
5
6
1
2
3
0
00
.
×
−
=
+


08:08
Thu, Sep 1
HUAWEI


Tue
9


COMPUTER
ORGANIZATION
AND DESIGN
THE HA WARE/SOFTWA INTERFACE
RISC-V EDITION
MK
MORGAN KAUFMANN
DAVID A. PATTERSON
JOHN L. HENNESSY

Cheat Sheet

- 1 A4 Cheat Sheet allowed (double sided)
  - Midterm II: 2 pages
  - Final: 3 pages
- Rules:
  - *Hand-written* – **not printed!**
  - Your **name** in pinyin on the top!
  - Cheat Sheets not complying to this rule will be **confiscated**!

FUNCTIONS OF SEVERAL VARIABLES

LEVEL CURVES

PARTIAL DERIVATIVES

SECOND PARTIAL DERIVATIVES

THE CHAIN RULE

IMPLICIT DIFFERENTIATION

TANGENT PLANE TO A LEVEL SURFACE

NORMAL LINE TO A LEVEL SURFACE

MAXIMUM AND MINIMUM VALUES

FINDING ABSOLUTE MAX. AND MINS.

MAXIMIZING AND MINIMIZING

FUNCTION OF n VARIABLES

CLAIRAUT'S THEOREM

PARTIAL DIFF. EQS

TOTAL DIFFERENTIAL

THEOREM

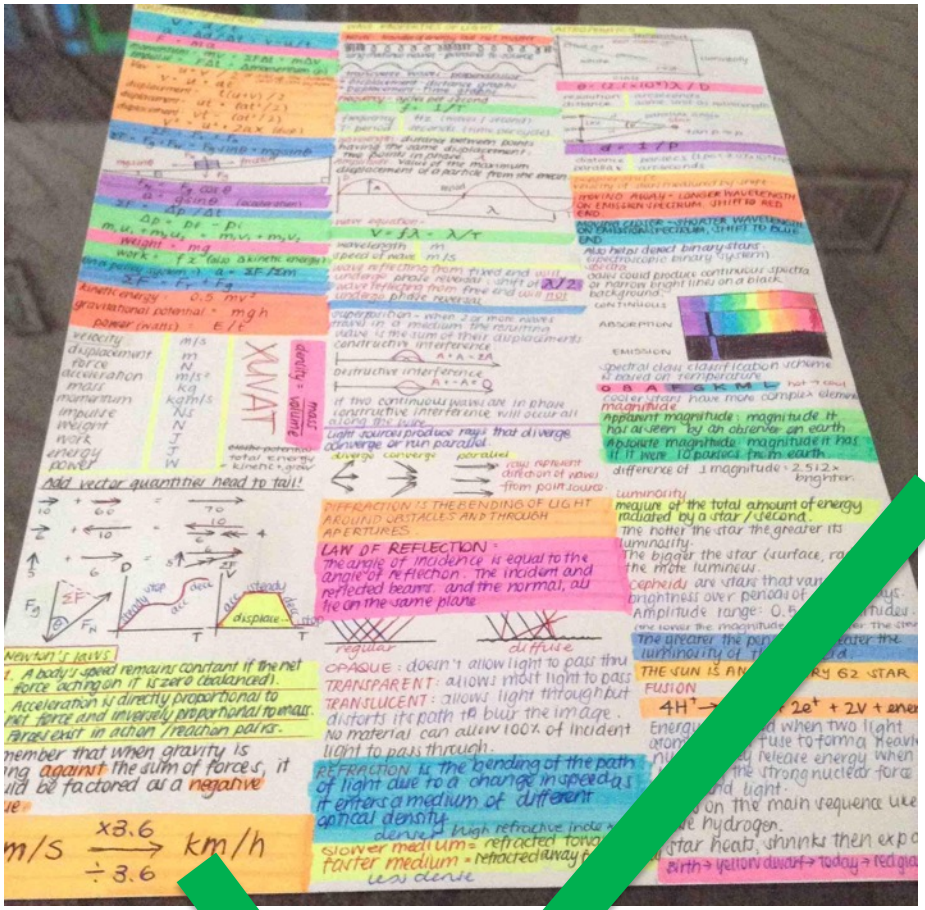
DEPENDENCY DIAGRAMS

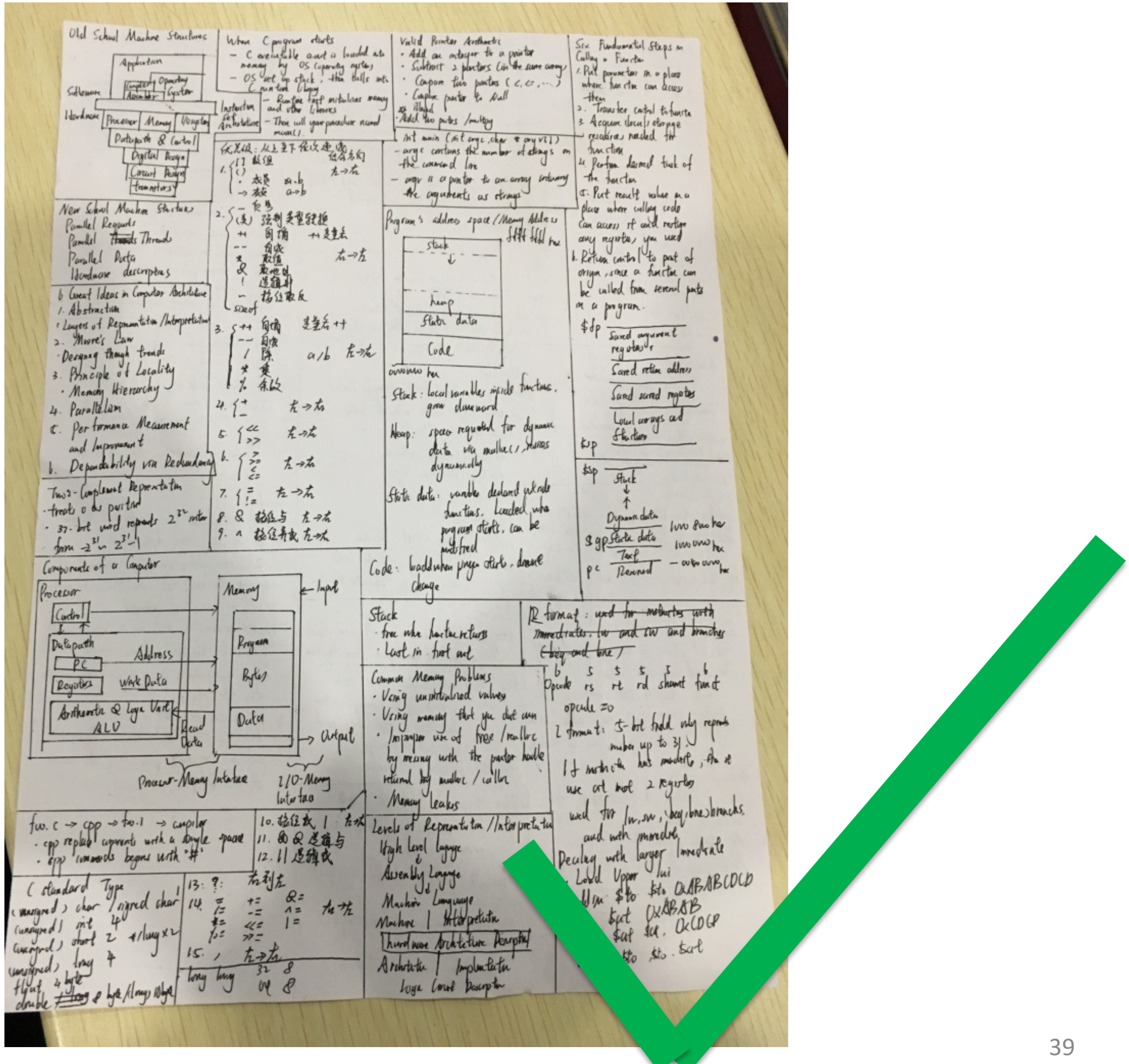
THE GRADIENT VECTOR

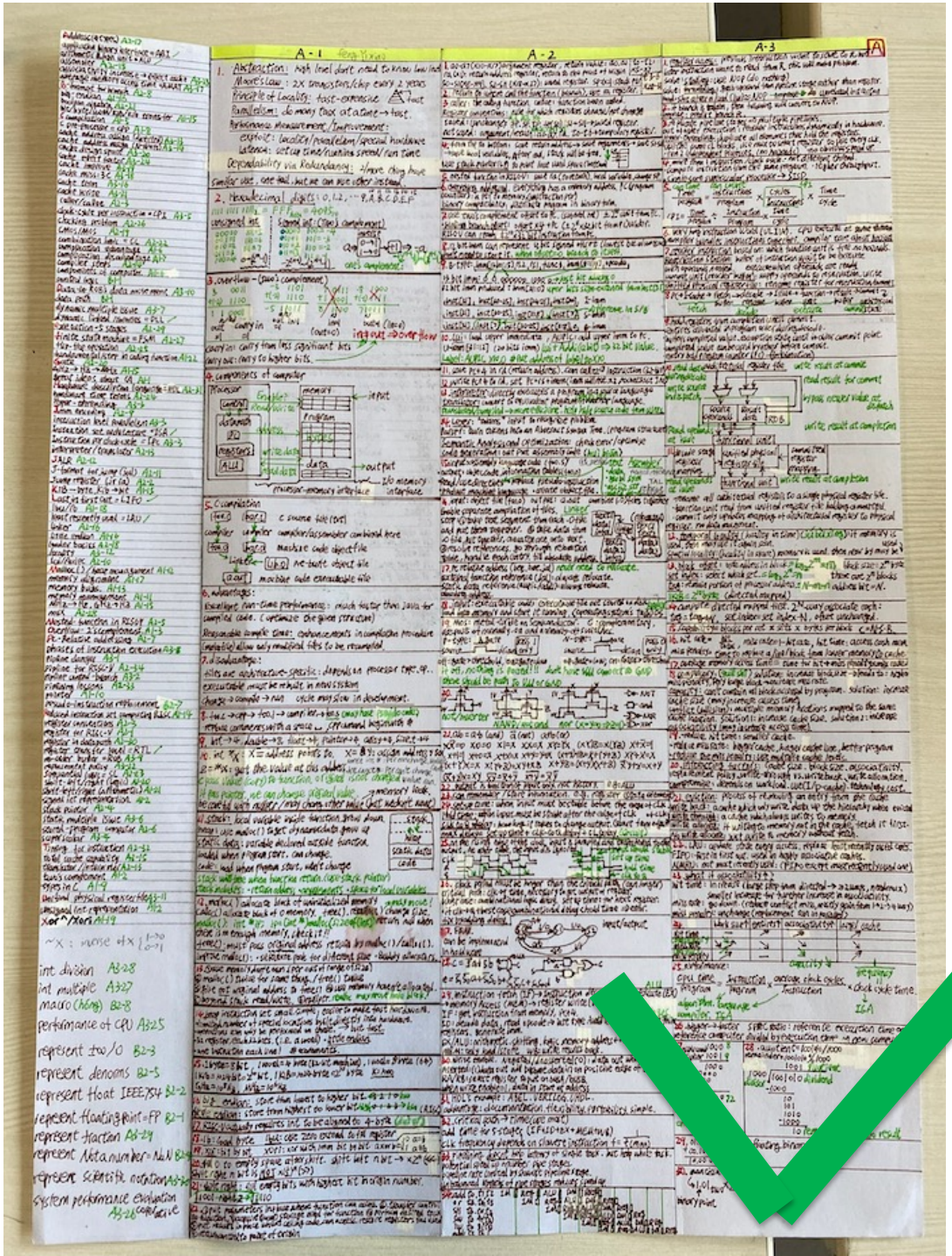
DIRECTIONAL DERIVATIVES

EQUATIONS OF TANGENT PLANES TO SURFACES

ε-δ DEFINITION OF CONTINUITY




Old School Machine Structures
When C program starts
Valid Pointer Arithmetic
Six Fundamental Steps in Calling a Function
New School Machine Structures
6 Great Ideas in Computer Architecture
Two's Complement Representation
Components of a Computer
Program's address space
Common Memory Problems
Levels of Representation / Interpretation
Processor
Memory
Control
Datapath
Input
Output



SIFT REFERENCE GUIDE (V.1.1) – CREATING TIMELINES WITH THE SIFT WORKSTATION

SANS COMPUTER FORENSICS and INCIDENT RESPONSE

THE PURPOSE OF THIS REFERENCE GUIDE IS TO WALK THROUGH THE PROCESS OF BOOTING THE SIFT WORKSTATION, CREATING A TIMELINE ("SUPER" OR "MICRO") AND REVIEWING IT.

1. VISIT: http://computer-forensics11.sans.org/community/downloads

- **oad:** SIFT Workstation VM Appliance
- **Do** FT Workstation Installation

2. BOOT SIFT VM

Login: *sansforensics*
Password: *forensics*

3. ELEVATE PRIVS

$ sudo su

4. CONNECT IMAGE TO SIFT

Plug hard drive to physical host and attach to SIFT VM

5. HARD DRIVE MOUNTING *(if you are using log2timeline-sift and Single DD you can skip to 7-A)*

SINGLE OR SPLIT IMAGE (2 options):

EWF/

- **nt_ewf.py** image.E01 /mnt/ewf
- # **i**mage.E01 /mnt/ewf/

Not Needed For 7-A

# mount -t ntfs -o ro,loop,show_sys_files,streams_interface=wind
offset=#### /mnt/ewf/<image> /mnt/windows_mount/

MOUNT TO MOUNT POINT

DD

SINGLE IM

mount -t ntfs -o ... w_sys_files,streams_interface=windows,offset=#### image.dd /m... ows_mount/

SPLIT IMAGE (2 step proc

affuse image.001 /mnt/aff
mount –t ntfs-3g –o loop,ro,show_... /mnt/aff/<image> /mnt/windows_

HOW TO CALCULATE THE OFFSET FOR MOUNTING

1. Run mmls to query partition layout
 # mmls image.E01
2. Identify partition and byte offset
3. (Partition byte offset) x (bytes per sector) = **offset #### to use!**
Example: 63 X 512 = 32256

Note: If needed, repeat for each partition. Make new mount point:
***# mkdir** /mnt/windows_mount2/*

6. log2timeline default timezone is set to examiner local host. To change use -z [TIMEZONE] option. To list all available timezones:
 # log2timeline -z list

7-A & 7-B

7-A: AUTOMATED SUPER TIMELINE CREATION

log2timeline-sift -o –z [TIMEZONE] -p [PARTITION #] -i [IMAGE FILE]

DISK IMAGE (prompt for partition, mount, and run):

XP # log2timeline-sift –z EST5EDT -i image

WIN7 # log2timeline-sift -win7 -z EST5EDT -i image

FOR PARTITION (mount and run using all applicable pl

XP # log2timeline-sift –z EST5EDT -p 0 -i parti

WIN7 # log2timeline-sift -win7 -z EST5EDT ...tion

OTHER USAGE EXAMPLES:

Display list of available plu
 # log2timeline -f list
Run log2timeline use... use only specific plugins:
 # log2timeline-sift ... prefetch –z EST5EDT -i image.dd
Help (man page...
 # log2timeli...

ANUAL "MICRO" TIMELINE CREATION

...PTIONS] [-f FORMAT] [-z TIMEZONE] [-o OUTPUT MODULE] [-w ...G_FILE/LOG_DIR |--|] [FORMAT FILE OPTIONS]

...EM METADATA (using log2timeline or fls)

...stem data w/log2timeline from mounted file system:
 # log... ...f mft -o mactime –r -z EST5EDT -w
mft.bo... ...olume/
OR Extract M... ...age using Sleuthkit:
 # fls -m "" -o offs... ...dd > fls.body
Convert body file for... ...format w/ mactime:
 # mactime –b fls.body –d

ARTIFACTS (run l2l on mounted file s... ...h plugins recursively)

Extract artifacts w/ log2timeline and ru... ...d file system:
 # log2timeline -f firefox3,chrome -o mactim... ...EDT -w
web.body /mnt/volume/
Convert body file format to CSV format w/ mactime:
 # mactime –b log2timeline.body –d > log2timeline.csv

HELP? OPTIONS? USAGE?

log2timeline -help
Log2timeline-sift -help
L2t_process -help

OTHER log2timeline OUTPUT FORMATS

Note: CSV is Default Output

-BeeDocs - Mac OS X visualization tool
-CEF - Common Event Format - ArcSight
-CFTL - XML file- CyberForensics TimeLab visualization tool
-CSV - comma separated value file
-Mactime - Both older and newer version of the format supported for use by TSK's mactime
-SIMILE - XML file - SIMILE timeline visualization widget
-SQLite - SQLite database
-TLN - Tab Delimited File
-TLN - Format used by some of H Carvey tools, expressed as a ASCII output
-TLNX - Format used by some of H Carvey tools, expressed as a XML document

8. CSV F... ...PUT *(/cases/timeline-output-folder)*

-dat... ...e event, in the format of MM/DD/YYYY
-t... ...of day, expressed in a 24h format, HH:MM:SS
-z: the timezone that was used to call the tool with.
-m: MACB meaning of the fields, comp w/ mactime format.
-source: Source short name (i.e. registry entries are REG)
-sourcetype: Desc of the source ("Internet Explorer" instead of WEBHIST)
-type: Timestamp type (i.e. "Last Accessed", "Last Written")
-user: Username associated with the entry, if one is available.
-host: Hostname associated with the entry, if one is available.
-short: Contains less text than the full description field.
-desc: where majority info is stored, the actual parsed desc of the entry.
-version: Version number of the timestamp object.
-filename: Filename with the full path that contained the entry
-inode: inode number of the file being parsed.
-notes: Some input modules insert additional information in the form of a note, which comes here. Or it can be used during the review.
-format: Input module name used to parse the file.
-extra: Additional information parsed is joined together and put here.

9. FILTER TIMELINE

Filter timeline with date range to include only:
l2t_process -b timeline.csv MM-DD-YYYY..MM-DD-YYYY > filtered.csv
Filter timeline with keyword list (one term per line in keywords.txt):
l2t_process -b timeline.csv -k keywords.txt > filtered.csv
What sources are in your timeline?
awk–F , '{print $6;}' timeline.csv| grep–v sourcetype|sort | uniq
Find all LNK files that reference E Drive
grep"Shortcut LNK" timeline.csv| grep"E:"
FiindMountPoints2 entries that reference E Drive
grep"MountPoints2 key" timeline.csv} | grep"E drive"
grepUSB timeline.csv| grep"SetupAPILog"

File System	M	A	C	B
Ext2/3	Modified	Accessed	Changed	N/A
FAT	Written	Accessed	N/A	Created
NTFS	File Modified	Accessed	MFT Modified	Created
UFS	Modified	Accessed	Changed	N/A

10. CONNECT TO SIFT

✓ ... > SETTINGS -> OPTIONS -> Shared Fo...ers -> Always Enabled (Check)
✓ 2. SIFT Desktop > VMware-Shared-Drive
✓ Access from a Win Machine \\SIFTWORKSTATION

11. REVIEW TIMELINE

Review timelines using:
- Open, Soft, Filter with Excel
- Import into SPLUNK
- SIMILE
- Tapestry

log2timeline PARSING PLUGINS

apache2_error - Apache2 error log file
chrome - Chrome history file
encase_dirlisting - CSV file that is exported from encase
evt - Windows 2k/XP/2k3 Event Log
evtx - Windows Event Log File (EVTX)
exif - Metadata information from files using ExifTool
ff_bookmark - Firefox bookmark file
firefox2 - Firefox 2 browser history
firefox3 - Firefox 3 history file
ftk_dirlisting - CSV file that is exported from FTK Imager (dirlisting)
generic_linux - Generic Linux logs that start with MMM DD HH:MM:SS
iehistory - index.dat file containg IE history
iis - IIS W3C log file
isatxt - ISA text export log file
jp_ntfs_change - CSV output file from JP (NTFS Change log)
mactime - Body file in the mactime format
mcafee - Log file
mft - NTFS MFT file
mssql_errlog - ERRORLOG file produced by MS SQL server
ntuser - NTUSER.DAT registry file
opera - Opera's global history file
oxml - OpenXML document pcap
pcap - PCAP file
pdf - Available PDF document metadata
prefetch - Prefetch directory
recycler - Recycle bin directory
restore 0.9 - Restore point directory
safari - Safari History.plist file
sam - SAM registry file
security - SECURITY registry file
setupapi - SetupAPI log file in Windows XP
skype_sql - Skype database
software - SOFTWARE registry file
sol - .sol (LSO) or a Flash cookie file
squid - Squid access log (http_emulate off)
syslog - Linux Syslog log file
system - SYSTEM registry file
tln - Body file in the TLN format
volatility - Volatility output files (psscan2, sockscan2, ...)
win_link - Windows shortcut file (or a link file)
wmiprov - wmiprov log file
xpfirewall - XP Firewall log

List plugins # log2timeline -f list
...HELP EXPAND THIS LIST. BUILD PLUGINS!!!

BY DAVID NIDES (12/16/2011)
TWITTER: @DAVNADS
BLOG: DAVNADS.BLOGSPOT.COM
EMAIL: DNIDES@KPMG.COM

CREDITS TO: ED GOINGS, ROB LE... KRISTINN GUDJONSSON, KPMG &...!
QUESTIONS/FEEDBACK–CONTACT US!

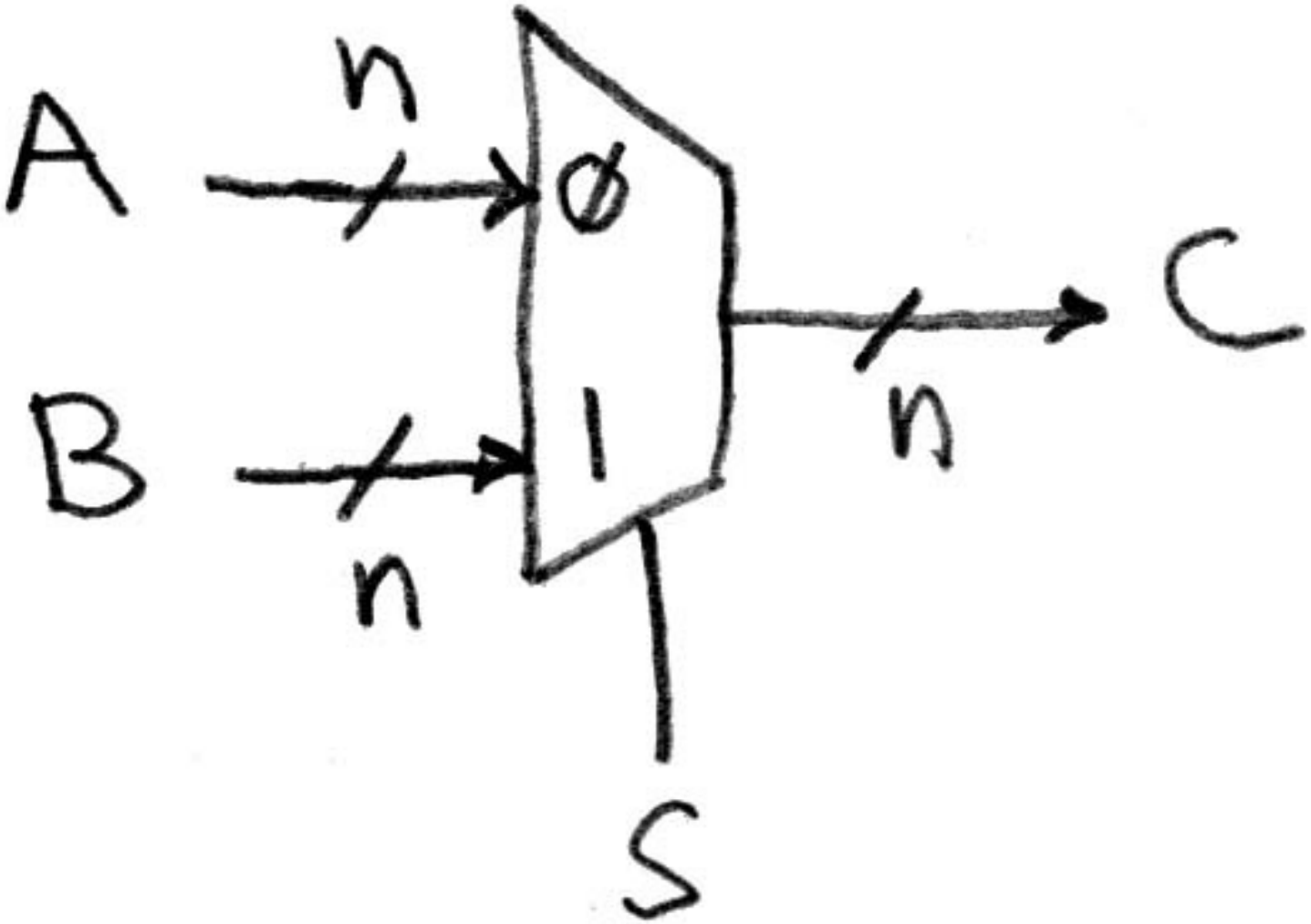
KEY

Red text – image/source
Blue text – mount point
Purple text – output file
Green text – log2timeline plugins
Brown text - TimeZone

Building Standard Functional Units

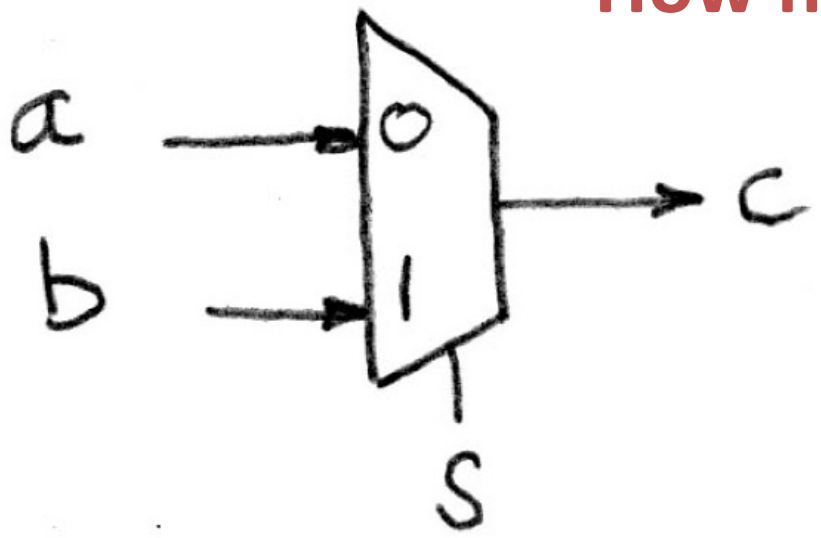
- Data multiplexers
- Arithmetic and Logic Unit
- Adder/ Subtractor

Data Multiplexer ("Mux")
(here 2-to-1, n-bit-wide)



N instances of 1-bit-wide mux

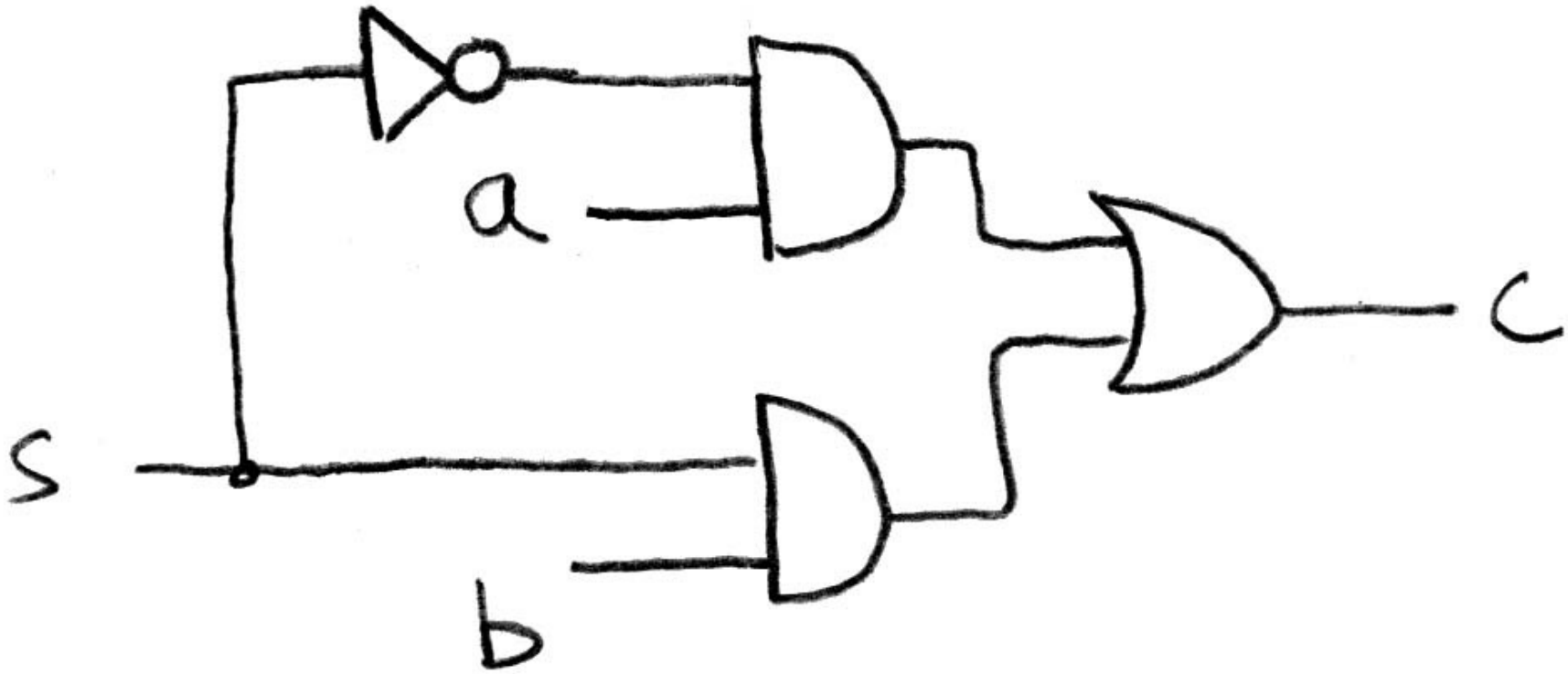
How many rows in TT?



$$
\begin{aligned}
c &= \overline{s}a\overline{b} + \overline{s}ab + s\overline{a}b + sab \\
&= \overline{s}(a\overline{b} + ab) + s(\overline{a}b + ab) \\
&= \overline{s}(a(\overline{b} + b)) + s((\overline{a} + a)b) \\
&= \overline{s}(a(1) + s((1)b) \\
&= \overline{s}a + sb
\end{aligned}
$$

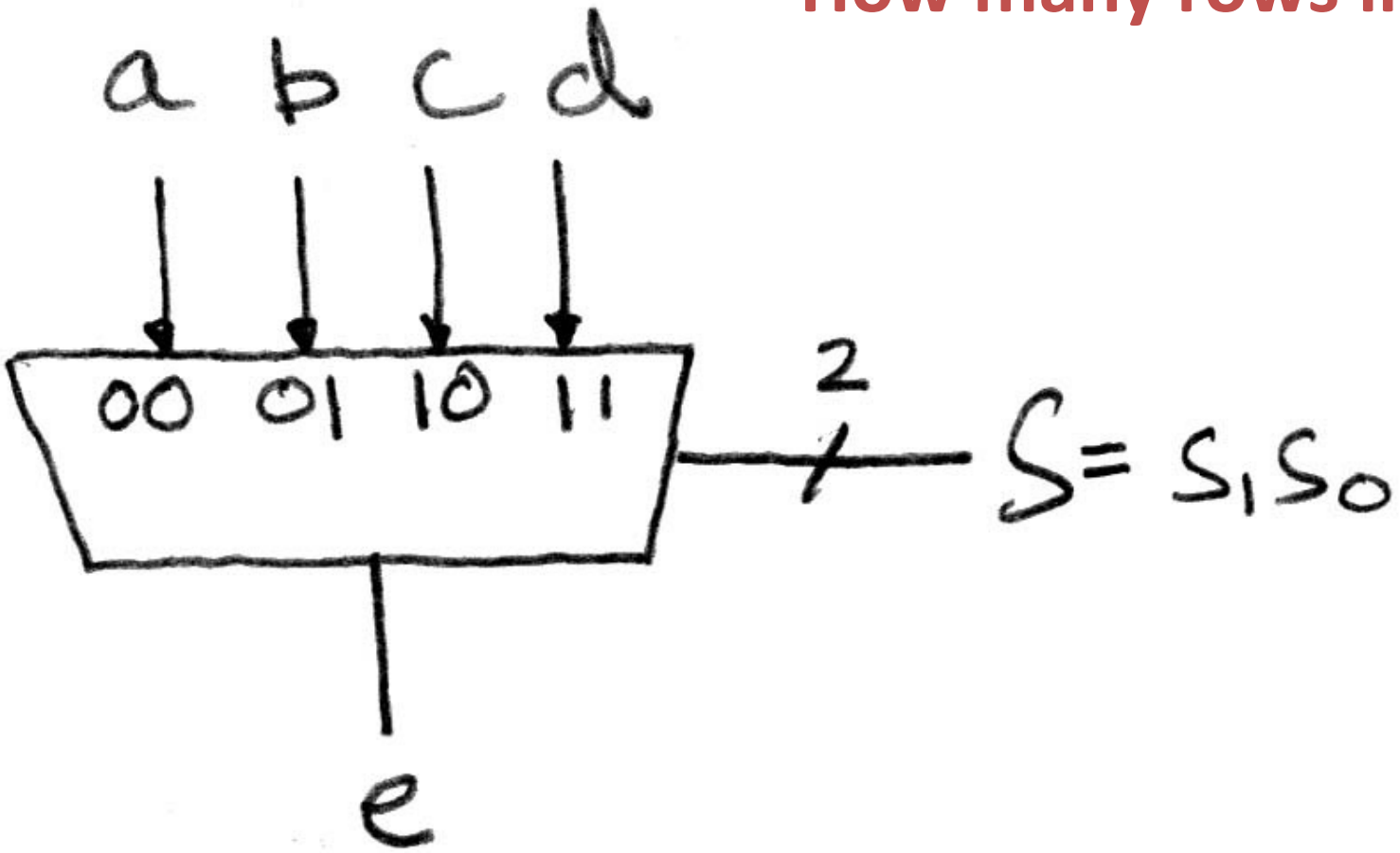
How do we build a 1-bit-wide mux?

$$\overline{s}a + sb$$



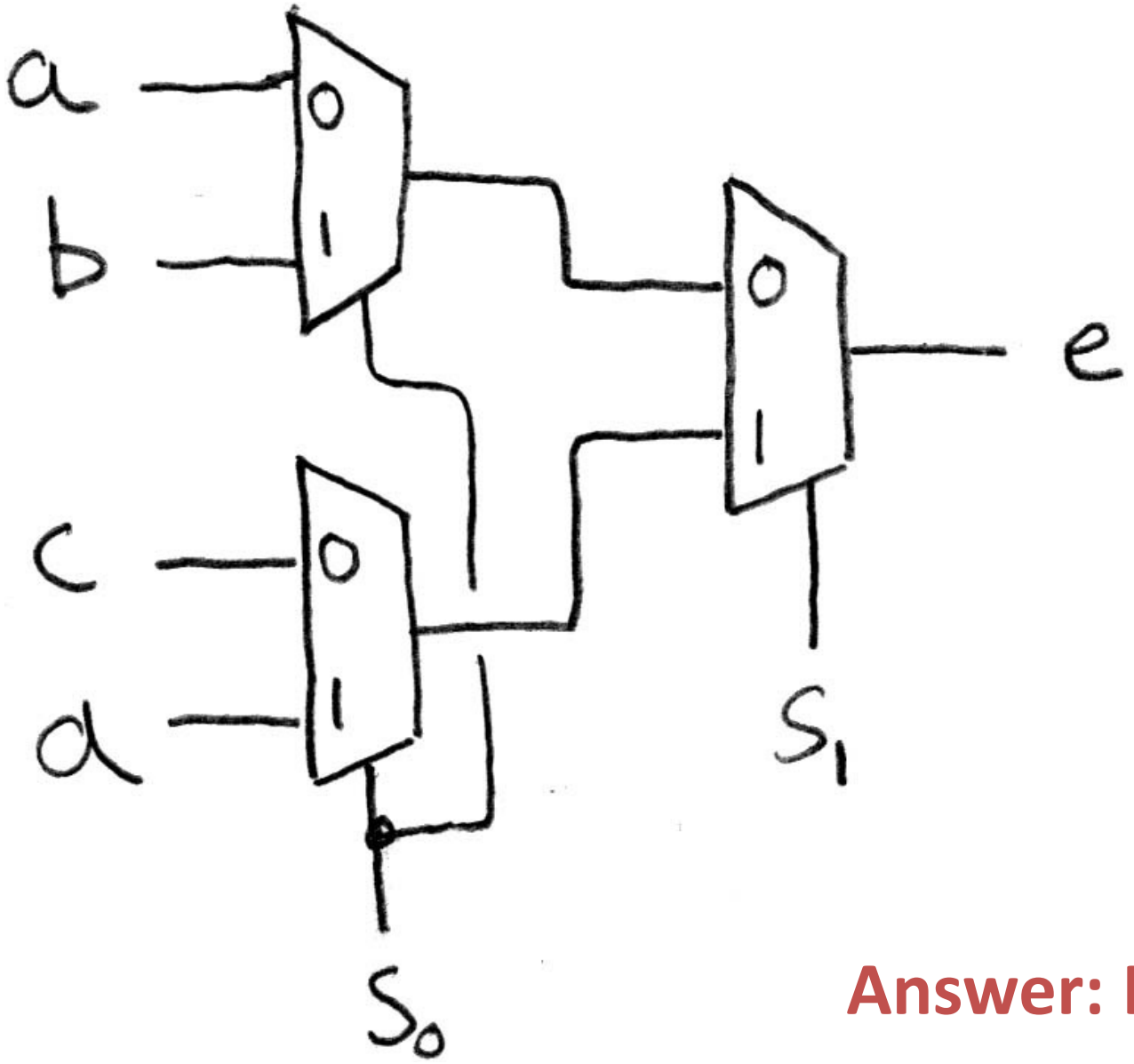
4-to-1 multiplexer?

How many rows in TT?



$$e = \overline{s_1}\,\overline{s_0}a + \overline{s_1}s_0b + s_1\overline{s_0}c + s_1s_0d$$

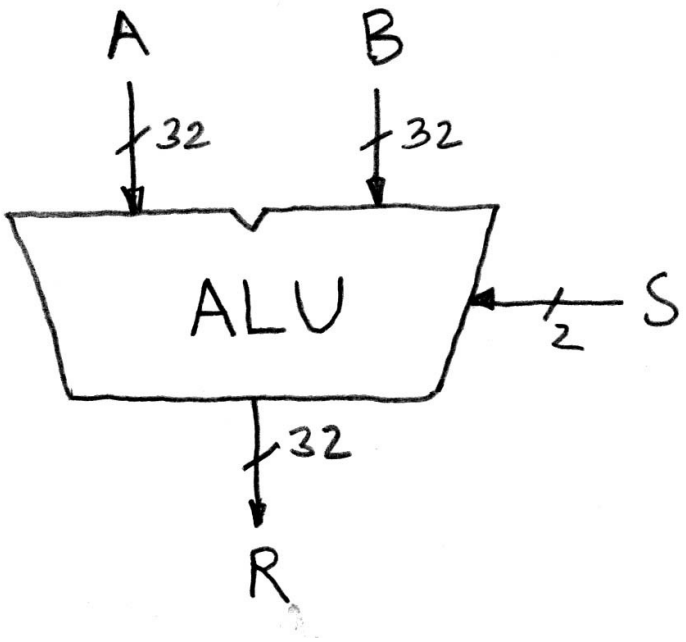
Another way to build 4-1 mux?



Answer: Hierarchically!

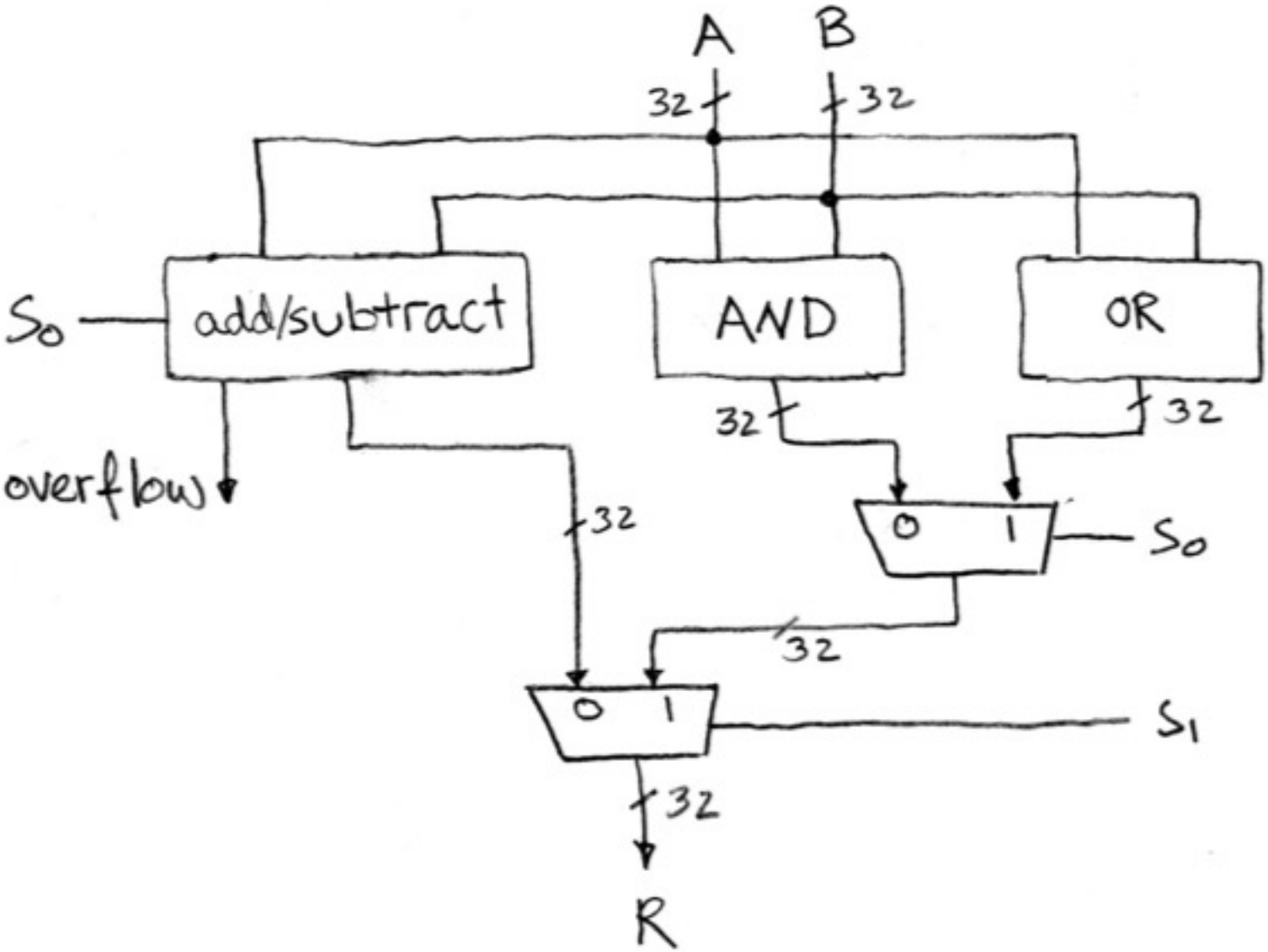
Arithmetic and Logic Unit

- Most processors contain a special logic block called the "Arithmetic and Logic Unit" (ALU)
- We'll show you an easy one that does ADD, SUB, bitwise AND, bitwise OR



when S=00, R=A+B
when S=01, R=A-B
when S=10, R=A AND B
when S=11, R=A OR B

Our simple ALU



Question

Convert the truth table to a boolean expression (no need to simplify):

A: F = xy + x(~y)

B: F = xy + (~x)y + (~x)(~y)

C: F = (~x)y + x(~y)

D: F = xy + (~x)y

E: F = (x+y)(~x+~y)

x	y	F(x,y)
0	0	0
0	1	1
1	0	0
1	1	1

How to design Adder/Subtractor?

- Truth-table, then determine canonical form, then minimize and implement as we've seen before
- Look at breaking the problem down into smaller pieces that we can cascade or hierarchically layer

Adder/Subtractor – One-bit adder LSB…

	a_3	a_2	a_1	a_0
+	b_3	b_2	b_1	b_0
	s_3	s_2	s_1	s_0

a_0	b_0	s_0	c_1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$s_0 =$$

$$c_1 =$$

Adder/Subtractor – One-bit adder (1/2)…

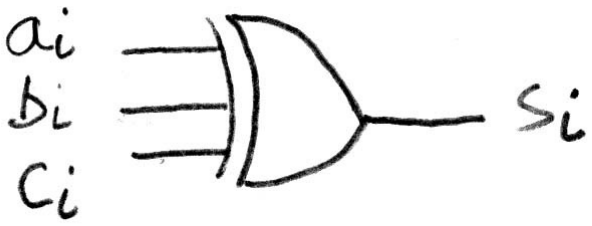
	c_3	c_2	c_1	
	a_3	a_2	a_1	a_0
+	b_3	b_2	b_1	b_0
	s_3	s_2	s_1	s_0

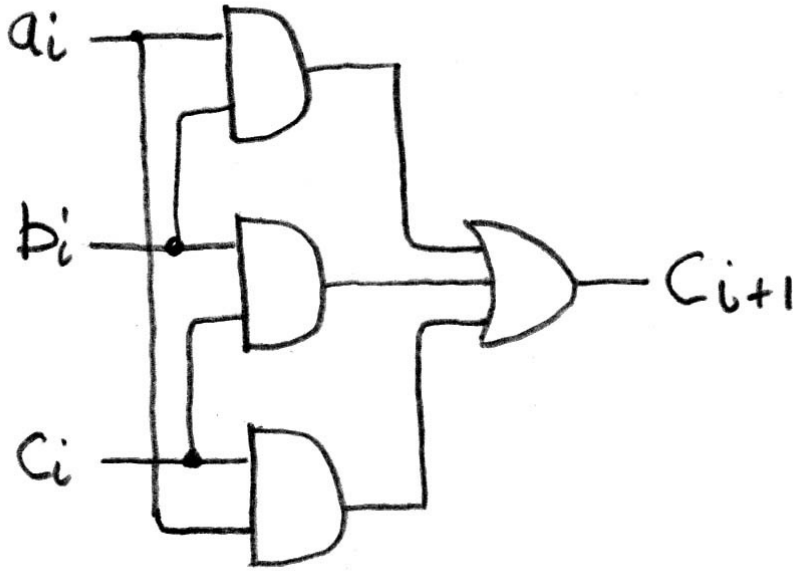
a_i	b_i	c_i	s_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$s_i =$$

$$c_{i+1} =$$

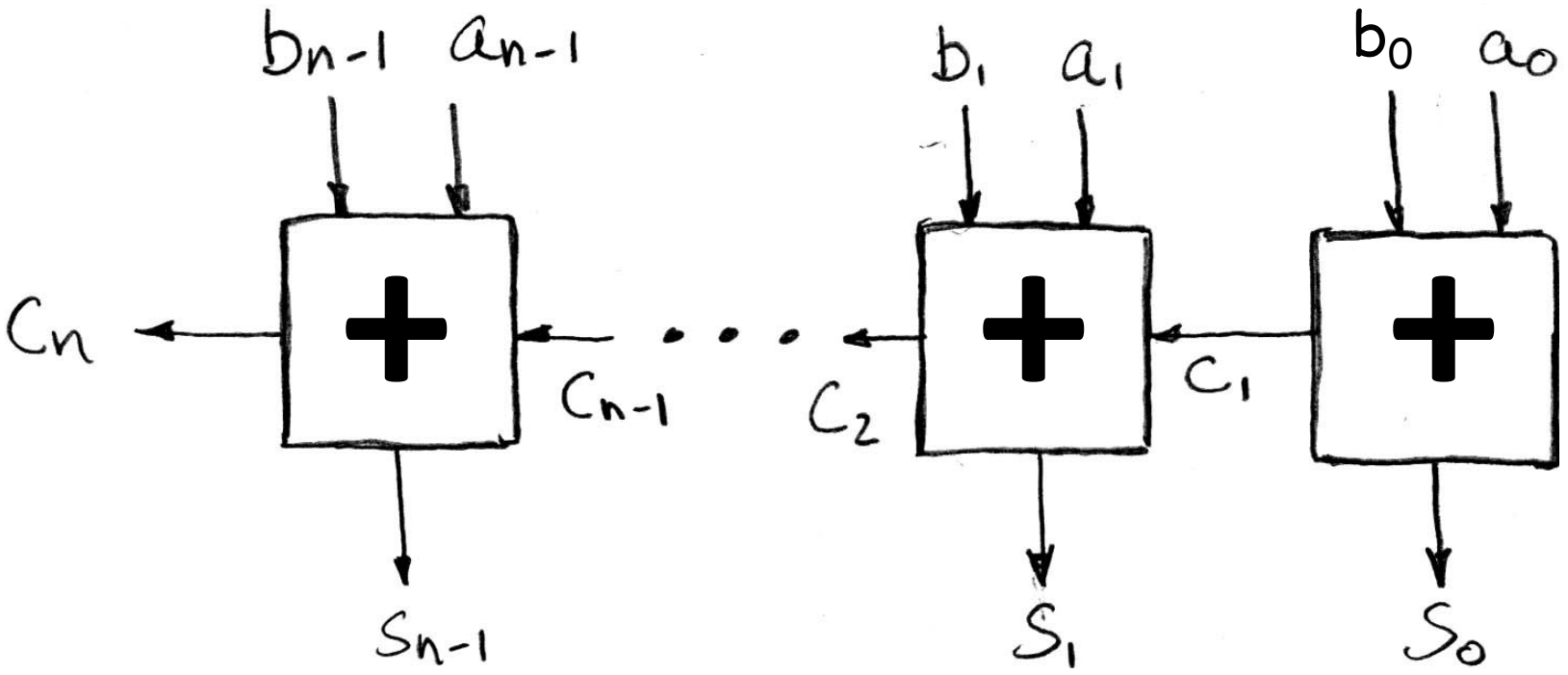
Adder/Subtractor – One-bit adder (2/2)





$$
\begin{aligned}
s_i &= \mathrm{XOR}(a_i, b_i, c_i) \\
c_{i+1} &= \mathrm{MAJ}(a_i, b_i, c_i) = a_i b_i + a_i c_i + b_i c_i
\end{aligned}
$$

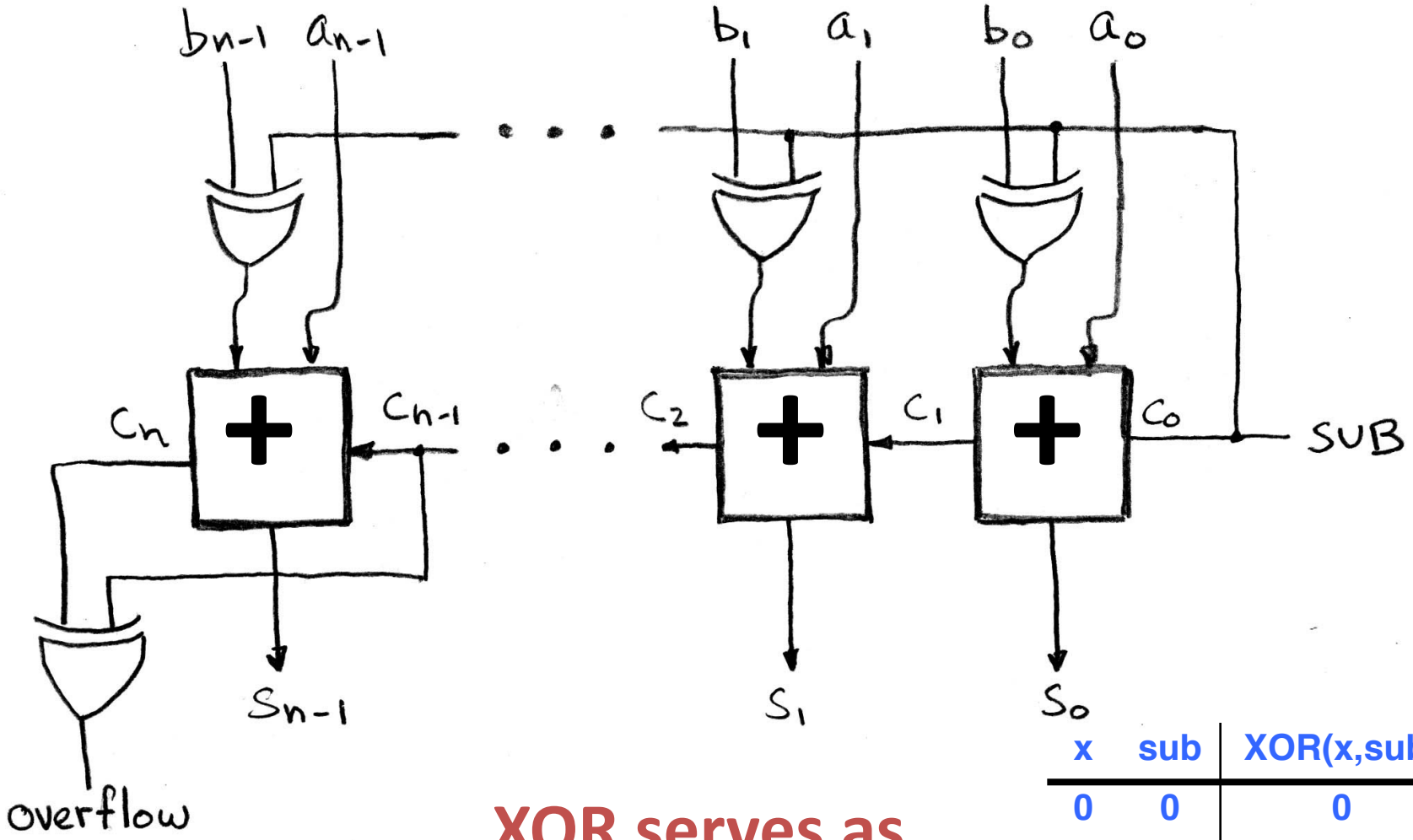
N 1-bit adders => 1 N-bit adder



What about overflow?

Overflow = c_n?

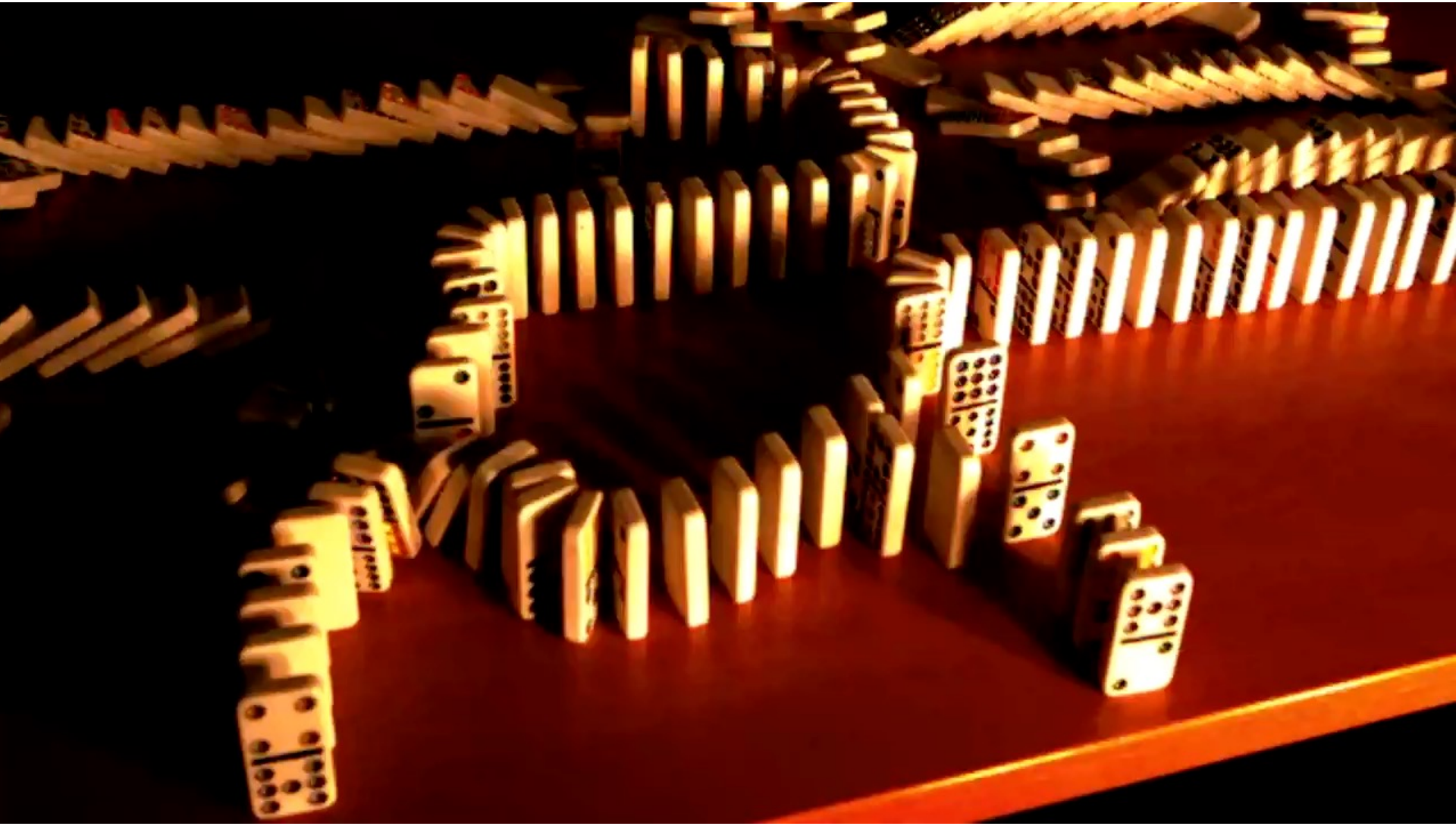
Extremely Clever Subtractor:
s = a + (-b)



x	sub	XOR(x,sub)
0	0	0
0	1	1
1	0	1
1	1	0

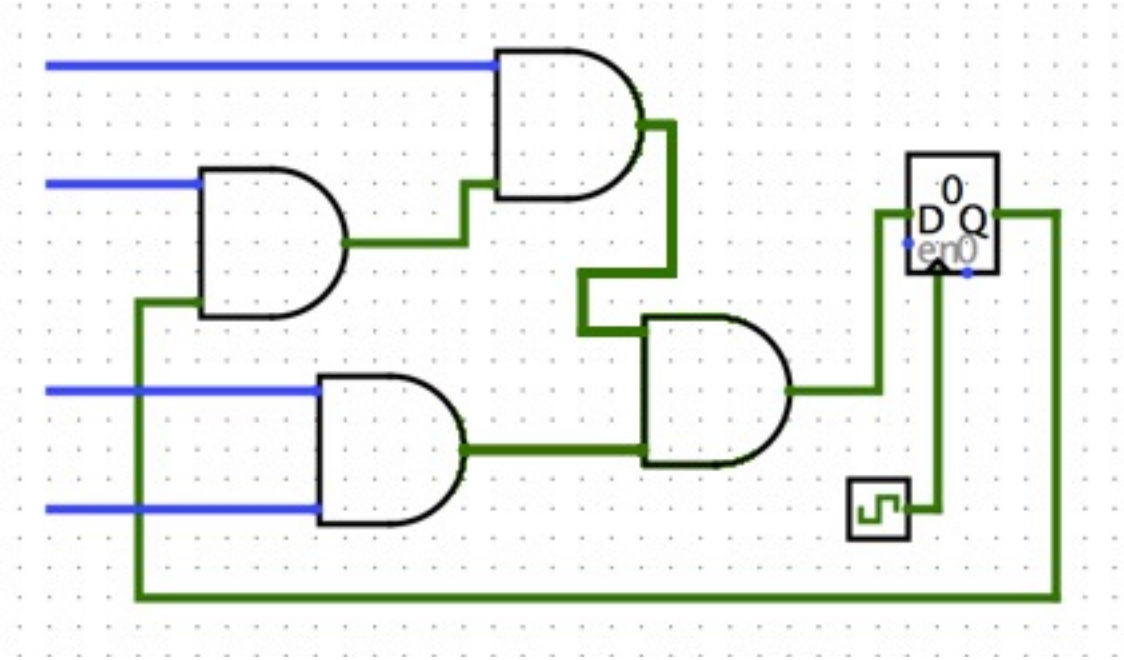
XOR serves as conditional inverter!

Domino Adder



Explanation: https://www.youtube.com/watch?v=lNuPy-r1GuQ
4-bit adder: https://www.youtube.com/watch?v=OpLU__bhu2w&t=0s

Question



Clock->Q 1ns
Setup 1ns
Hold 1ns
AND delay 1ns

What is maximum clock frequency?

- A: 5 GHz
- B: 500 MHz
- C: 200 MHz
- D: 250 MHz
- E: 1/6 GHz

In Conclusion

- Finite State Machines have clocked state elements plus combinational logic to describe transition between states
  - Clocks synchronize D-FF change (Setup and Hold times important!)
- Standard combinational functional unit blocks built hierarchically from subcomponents