

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)

Machine Interpretation

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

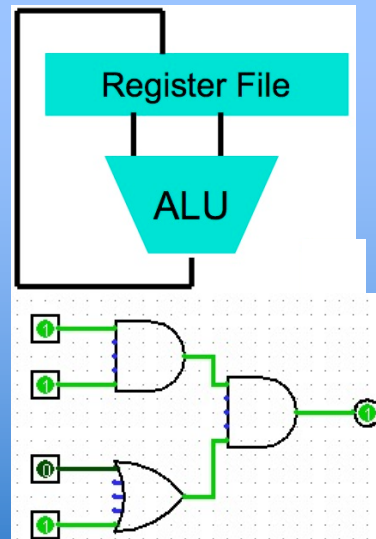
Architecture Implementation

Logic Circuit Description (Circuit Schematic Diagrams)

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

```
lw  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
```



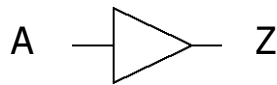
Outline

- 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

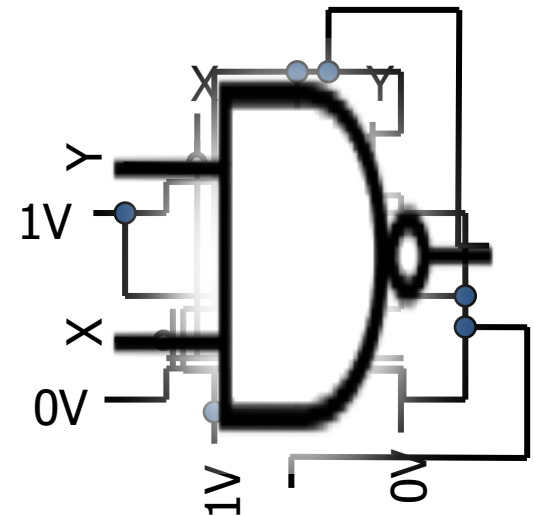
– Buffer, NOT



– AND, NAND

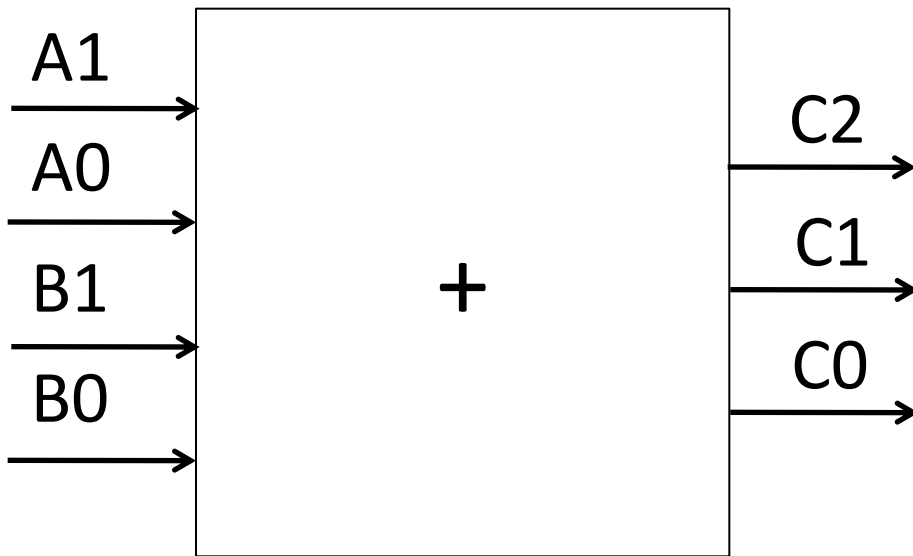


– OR, NOR



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



How
Many
Rows?

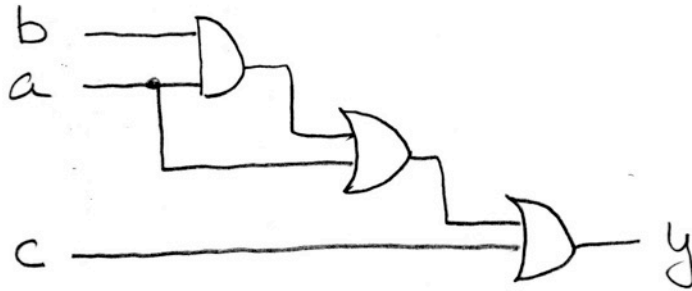
A	B	C
a_1a_0	b_1b_0	$c_2c_1c_0$

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



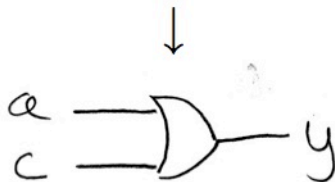
original circuit

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

equation derived from original circuit

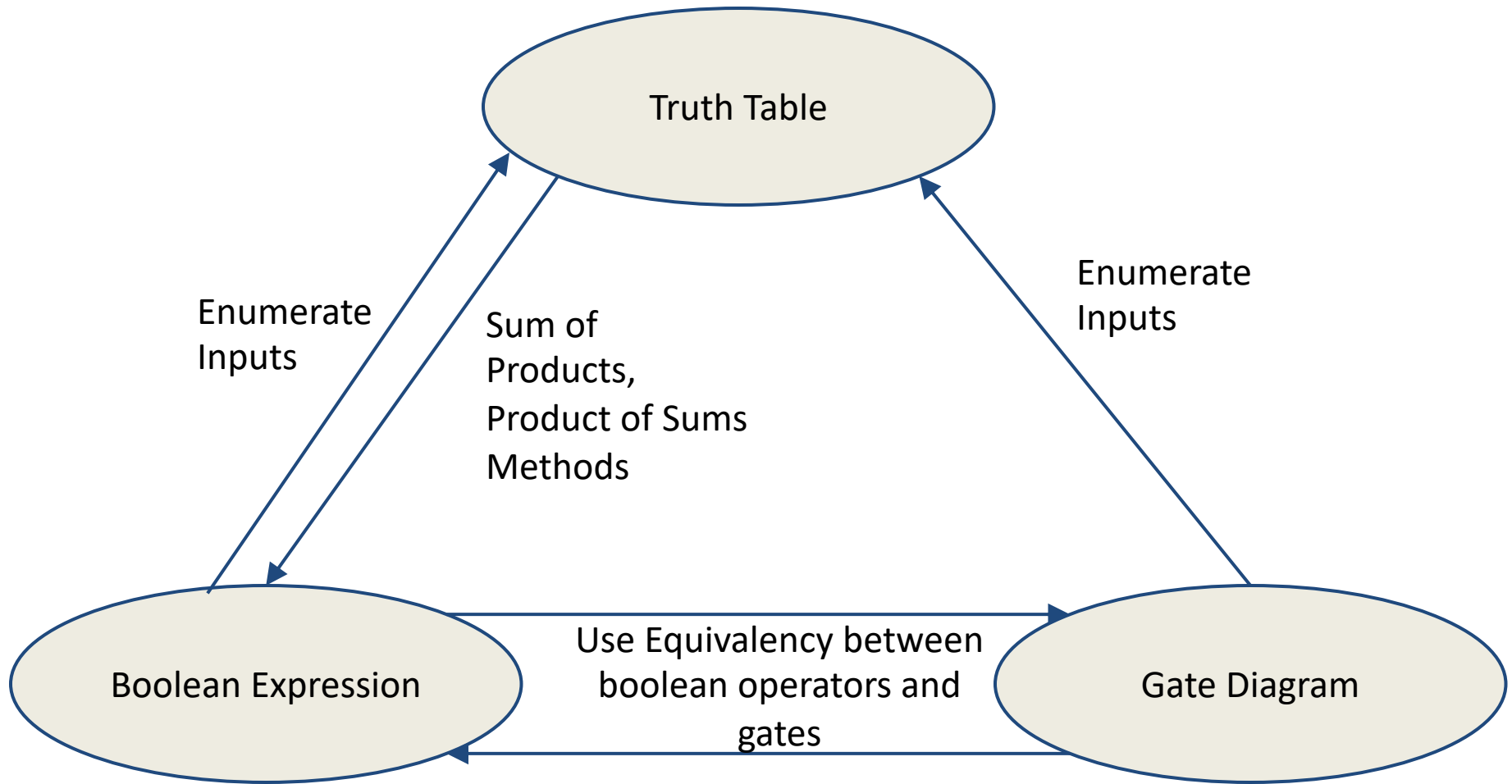
$$\begin{aligned} &\downarrow \\ &= ab + a + c \\ &\downarrow \\ &= a(b + 1) + c \\ &= a(1) + c \\ &= a + c \end{aligned}$$

algebraic simplification



simplified circuit

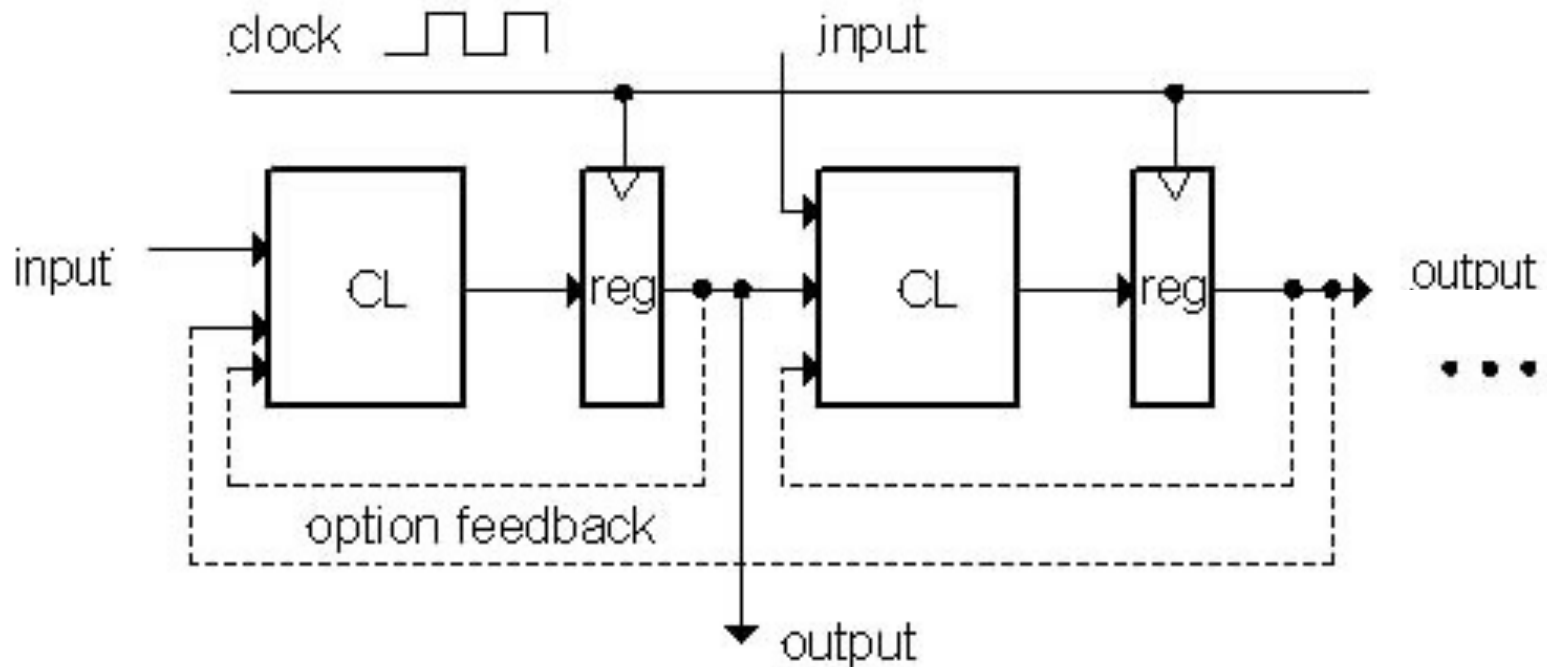
Representations of Combinational Logic (groups of logic gates)



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)

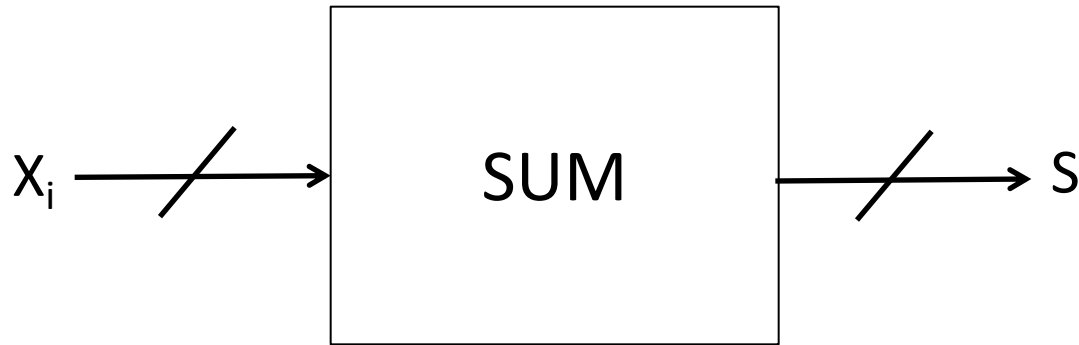
Model for Synchronous Systems



- 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?



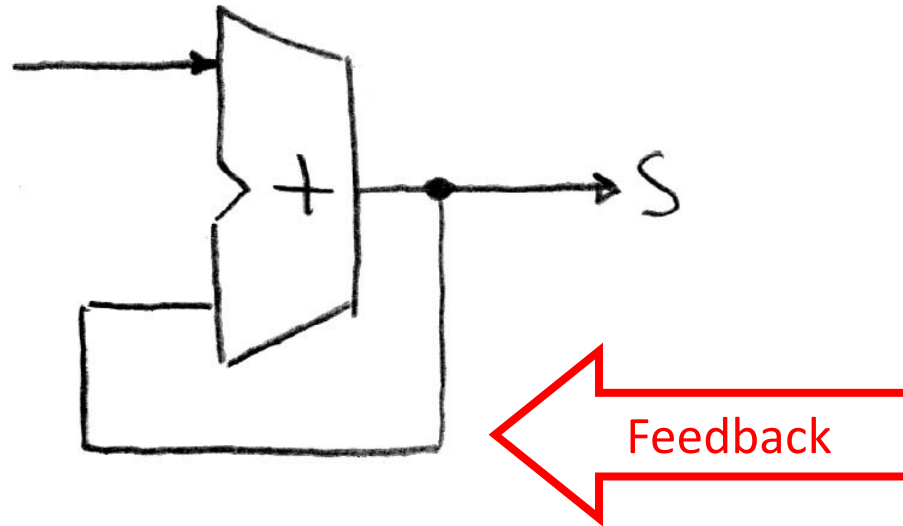
Want:

```
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?

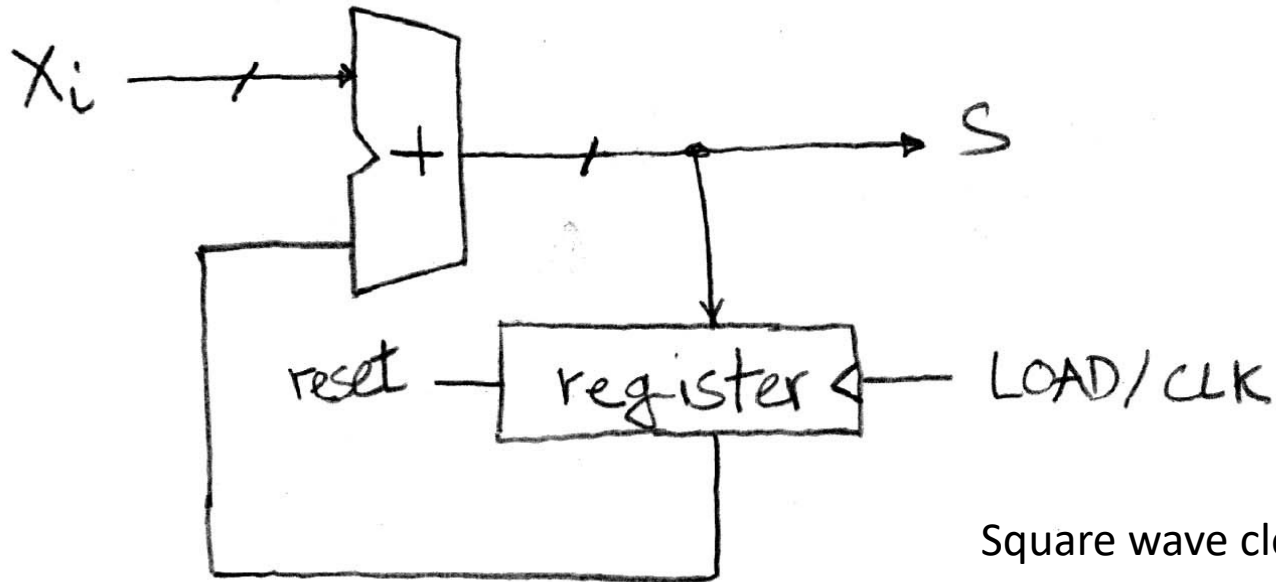


No!

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

Reason #2: How do we say: 'S=0'?

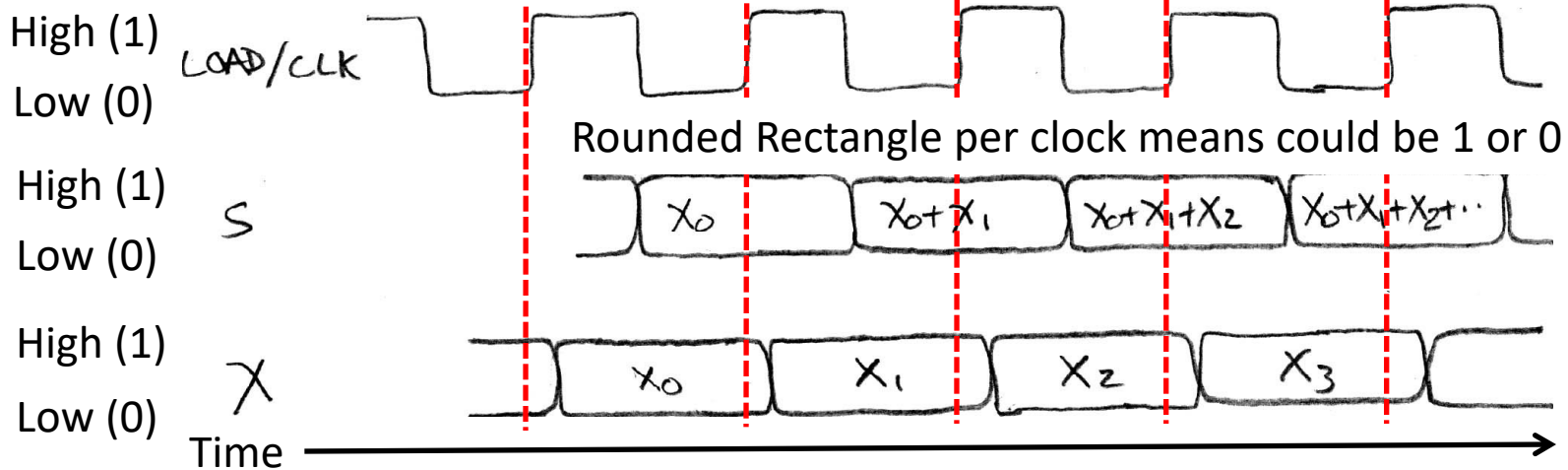
Second Try: How About This?



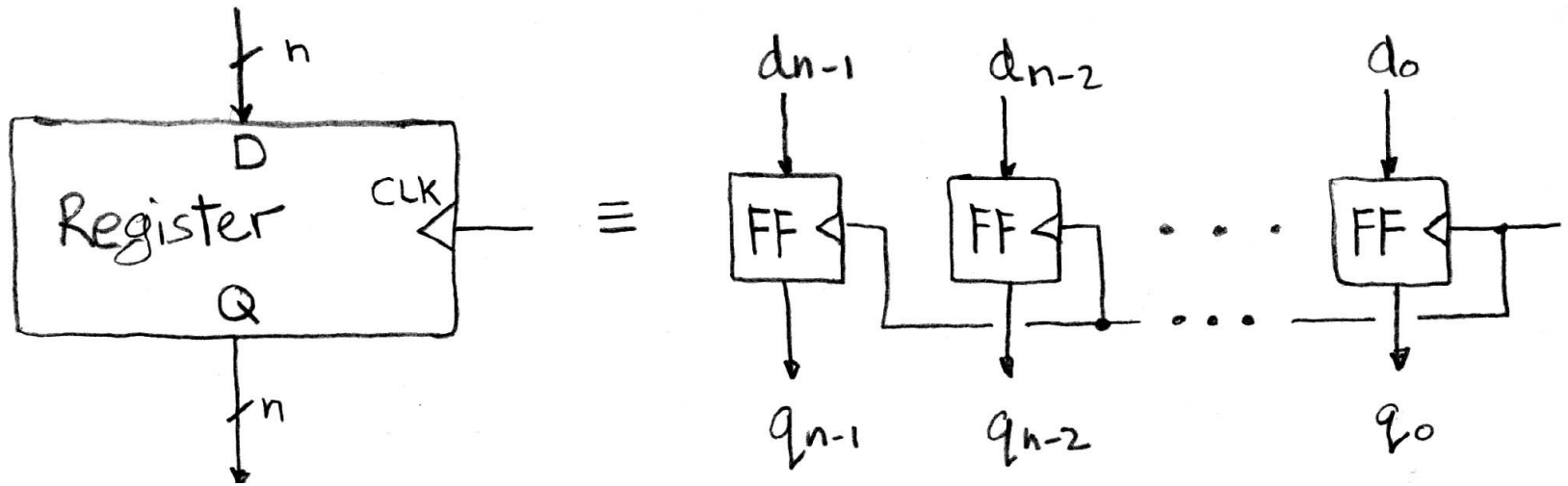
Register is used to hold up the transfer of data to adder

Square wave clock sets when things change

Rough timing ...



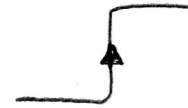
Register Internals



- 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”

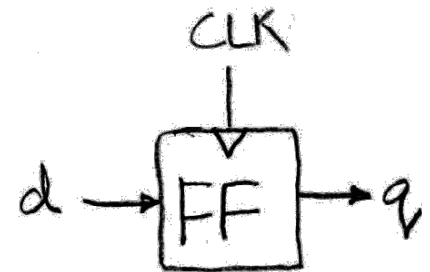
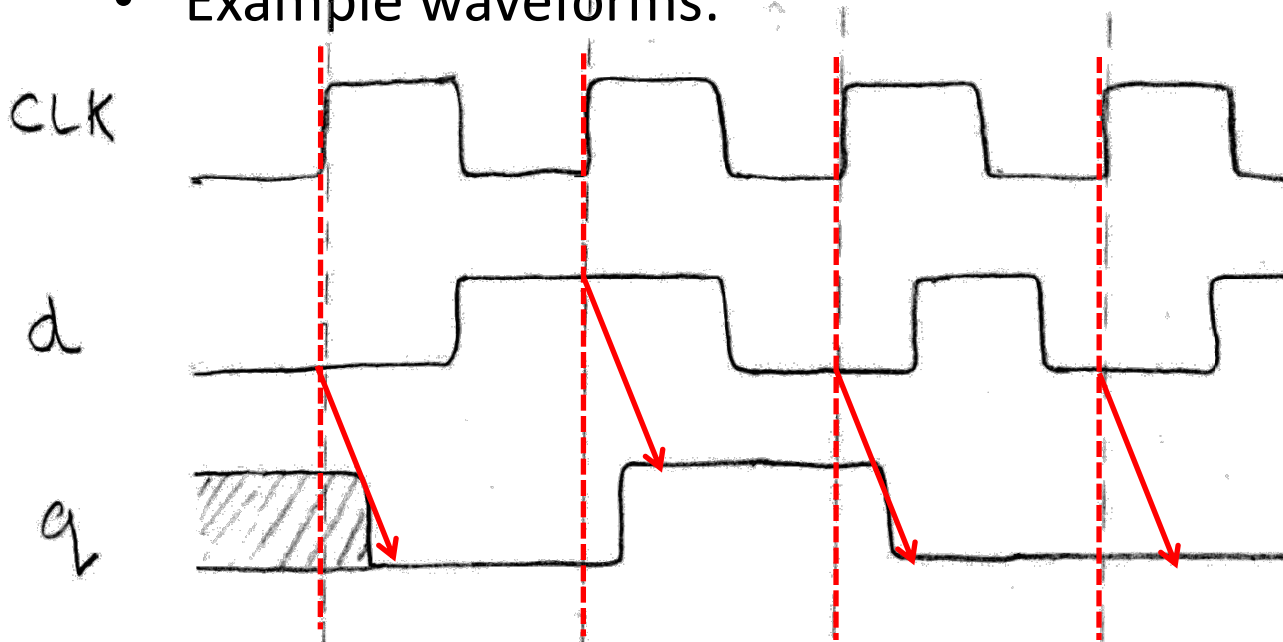
Flip-Flop Operation

- Edge-triggered d-type flip-flop
 - This one is “positive edge-triggered”



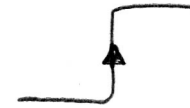
- “On the rising edge of the clock, the input d is sampled and transferred to the output. At all other times, the input d is ignored.”

- Example waveforms:



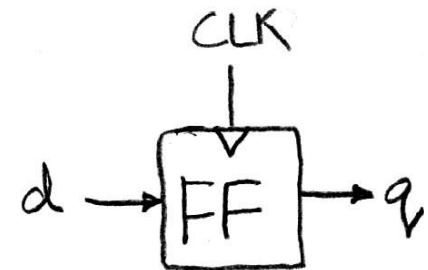
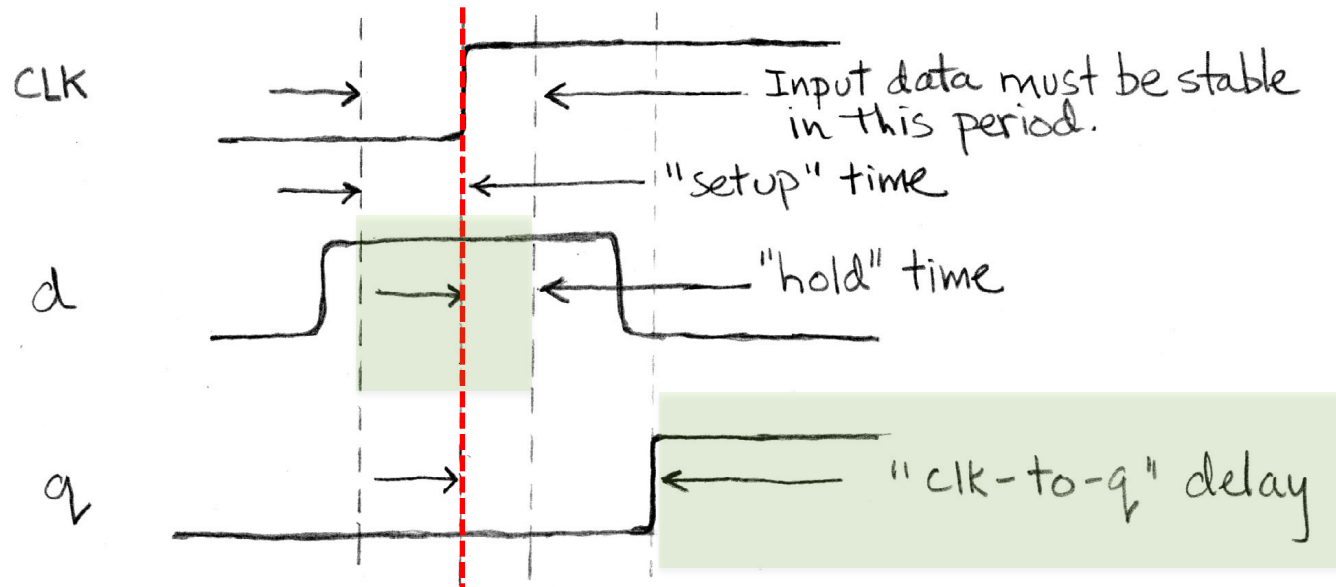
Flip-Flop Timing

- 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.”

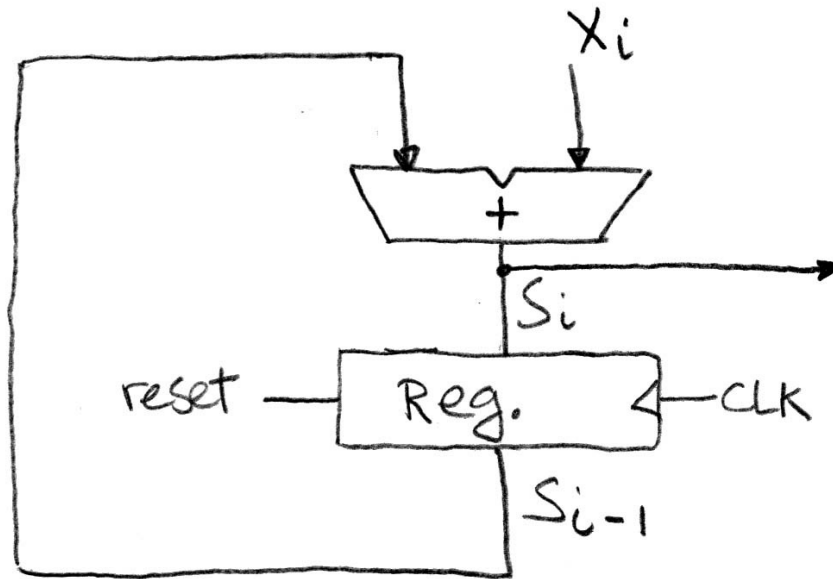
- Example waveforms (more detail):



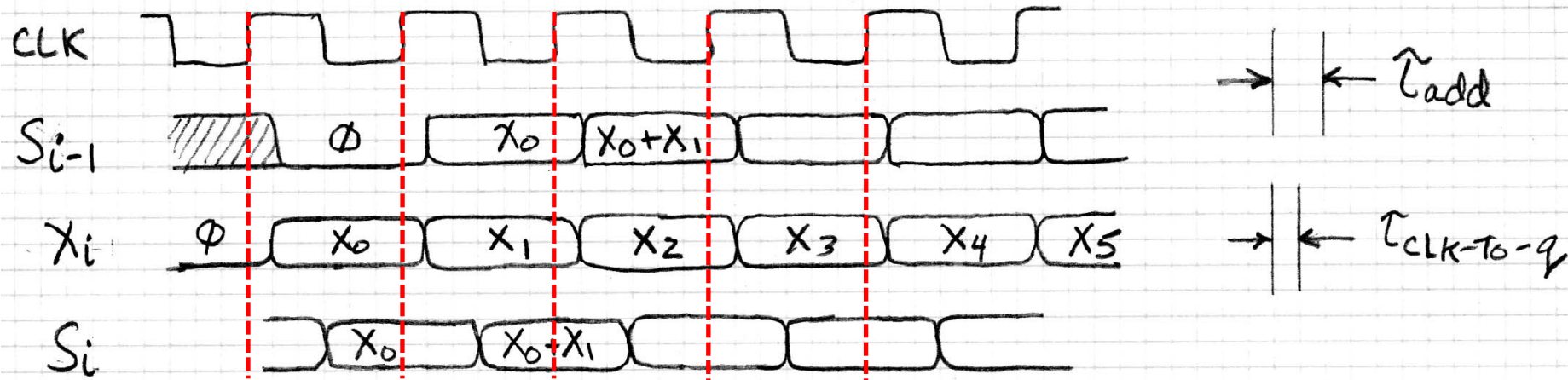
Hardware Timing Terms

- **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

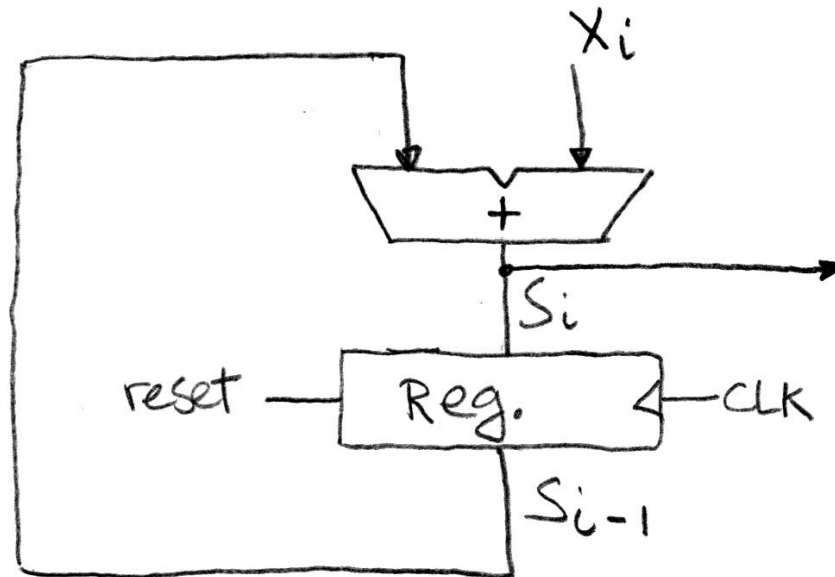
Accumulator Timing 1/2



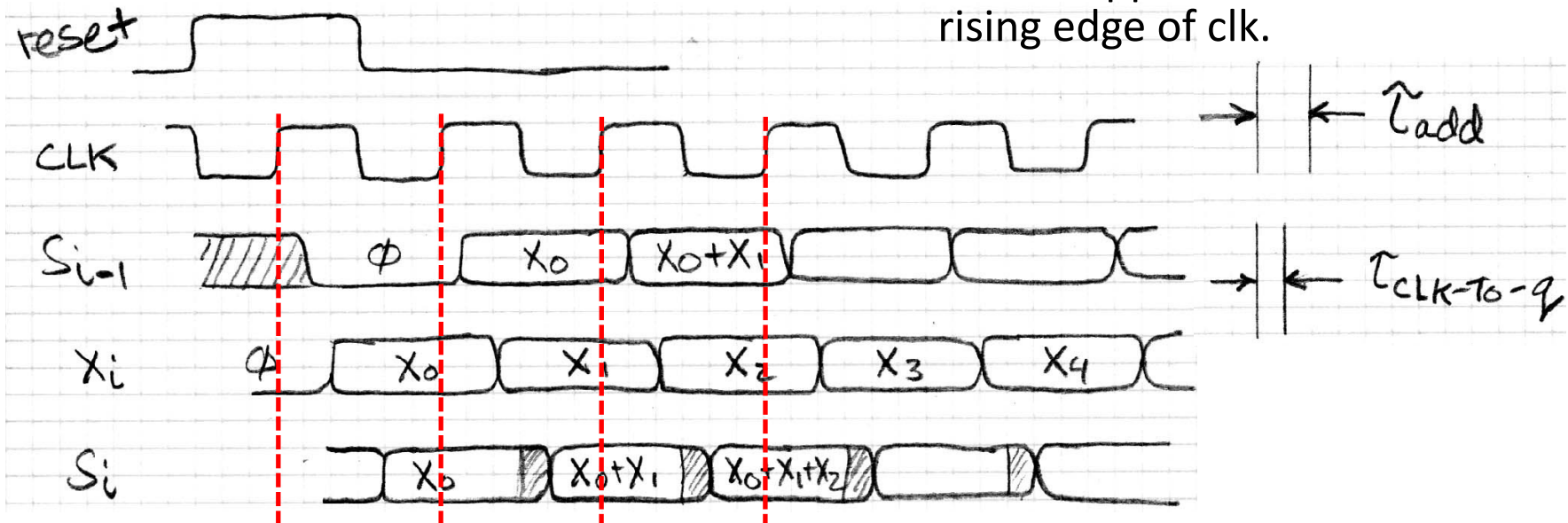
- Reset input to register is used to force it to all zeros (takes priority over D input).
- S_{i-1} holds the result of the $i^{\text{th}}-1$ iteration.
- Analyze circuit timing starting at the output of the register.



Accumulator Timing 2/2

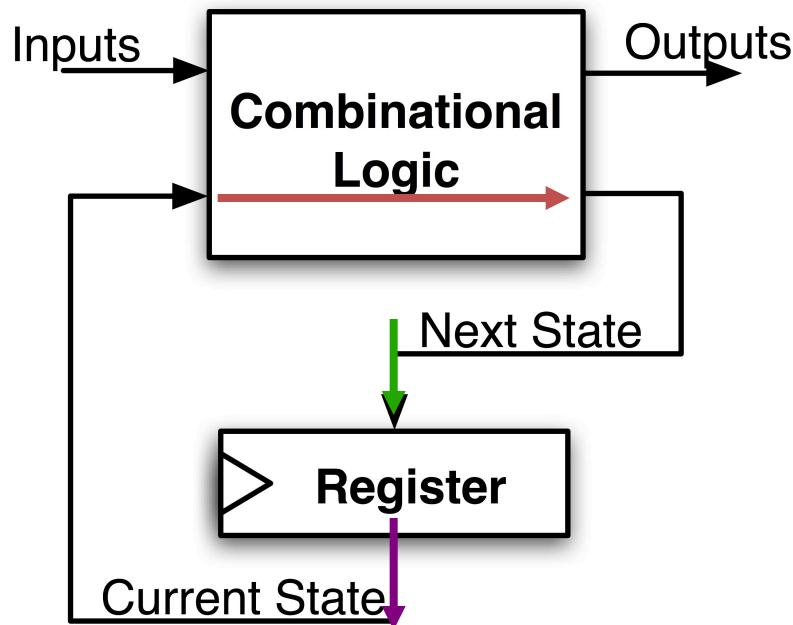


- reset signal shown.
- Also, in practice X might not arrive to the adder at the same time as S_{i-1}
- S_i temporarily is wrong, but register always captures correct value.
- In good circuits, instability never happens around rising edge of clk.



Maximum Clock Frequency

- What is the maximum frequency of this circuit?

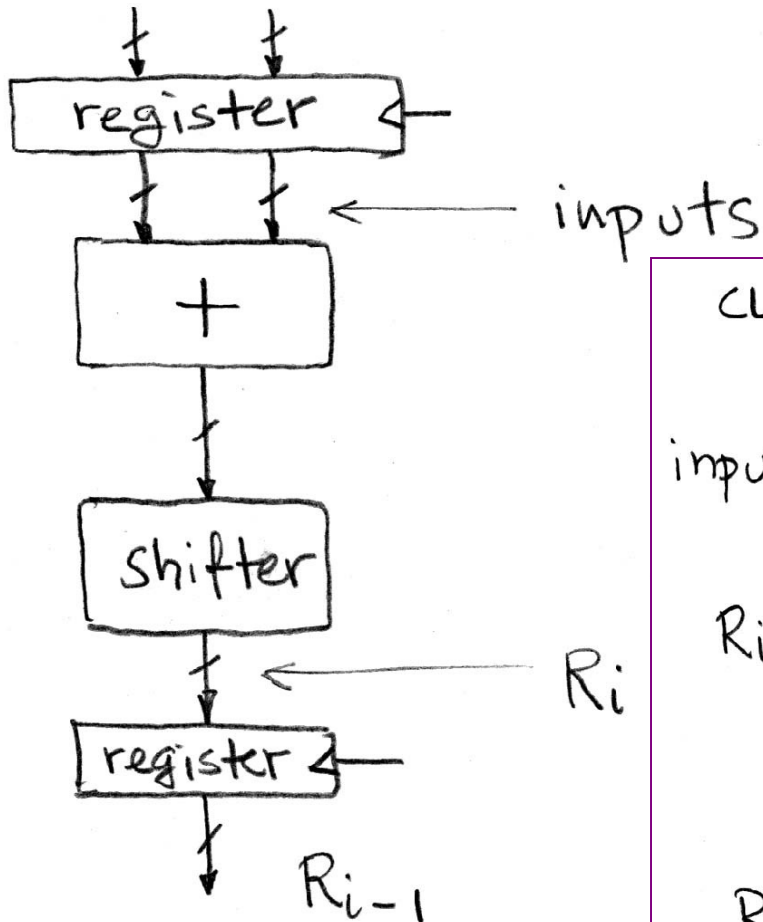


Hint:

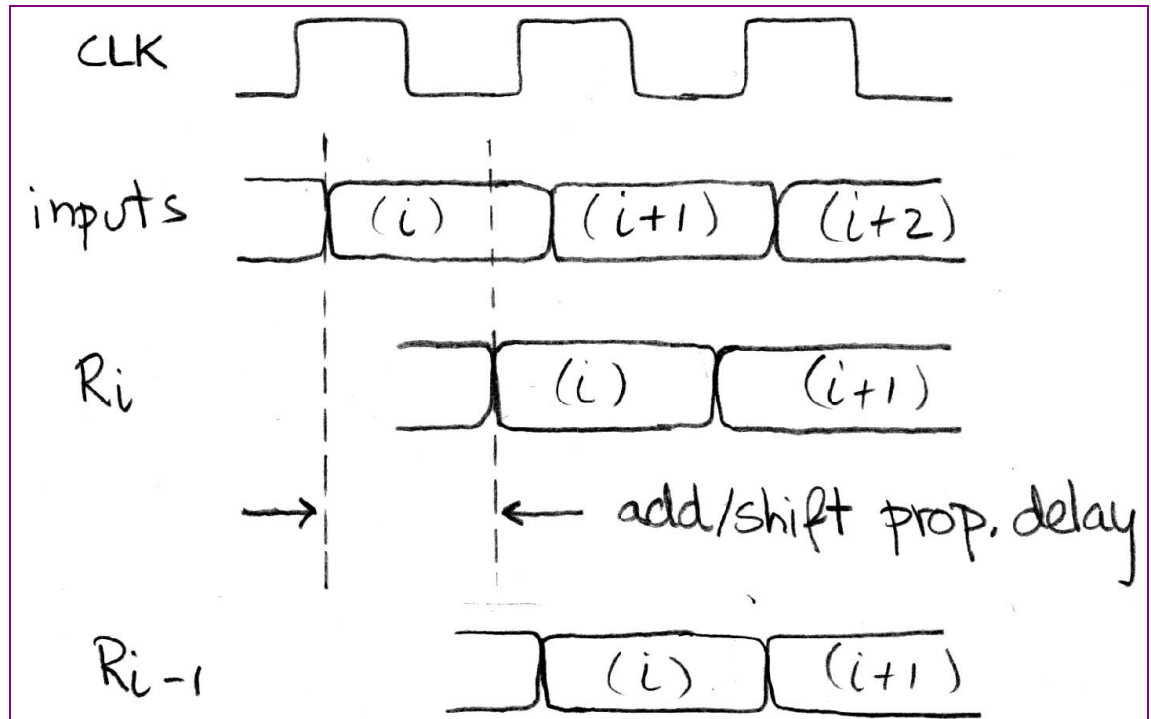
Frequency = $1/\text{Period}$

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

Critical Paths



Timing...

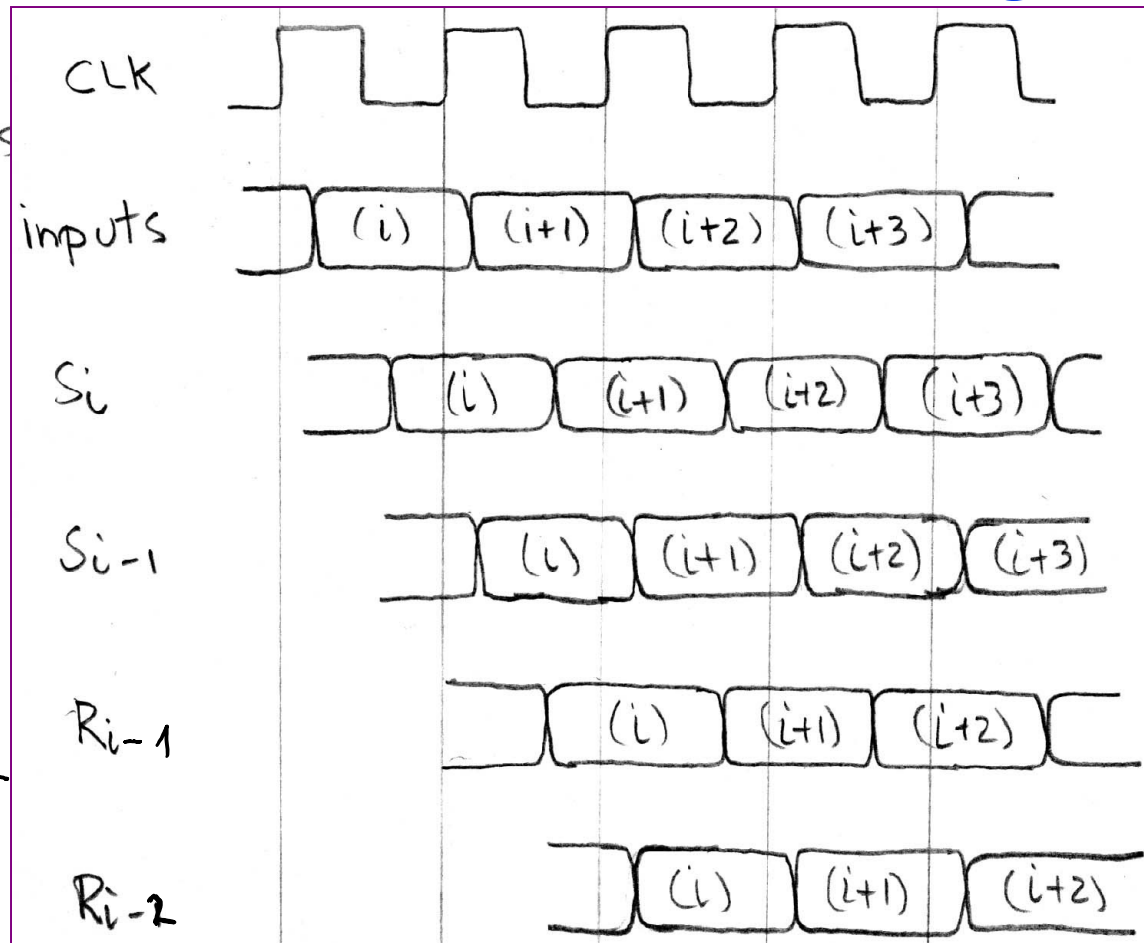
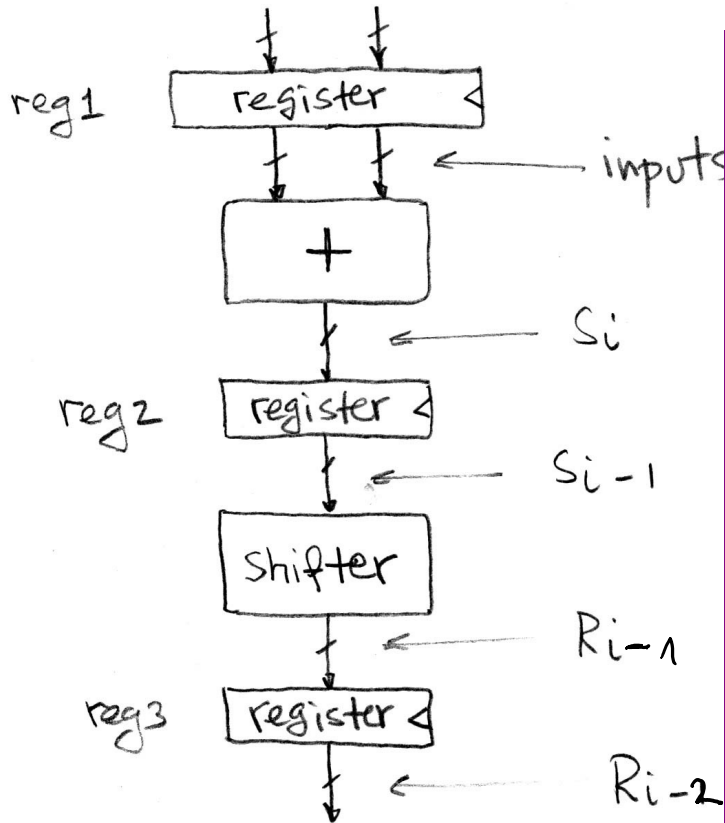


Note: delay of 1 clock cycle from input to output.

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

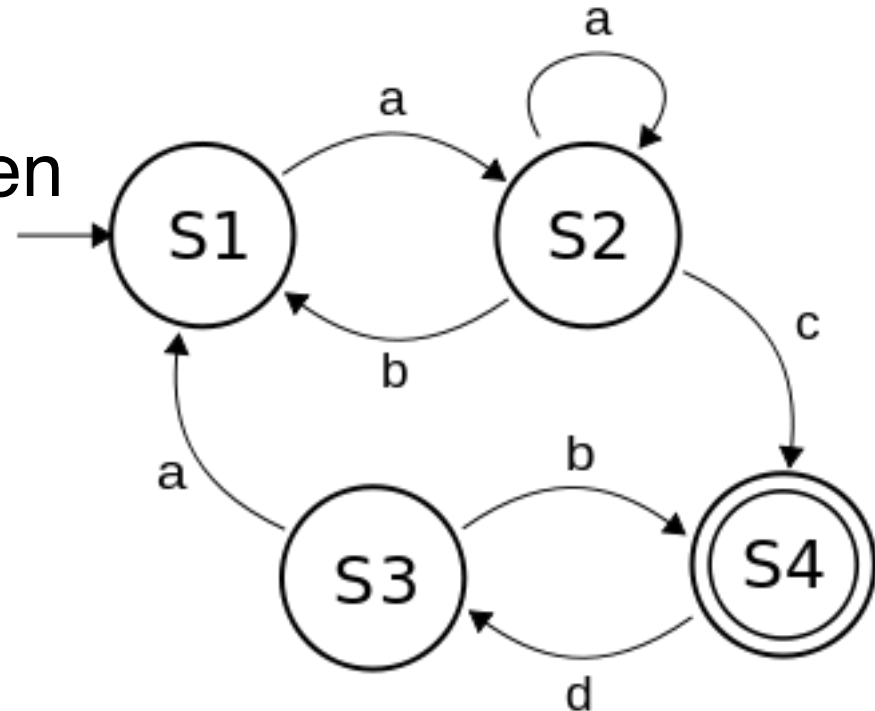
- The clock period ***must be*** longer than the critical path
 - Otherwise, you will get the wrong answers
 - But it can be even longer than that
- Critical path:
 - clk->q time
 - Necessary to get the output of the registers
 - ***worst case*** combinational logic delay
 - ***Setup time*** for the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

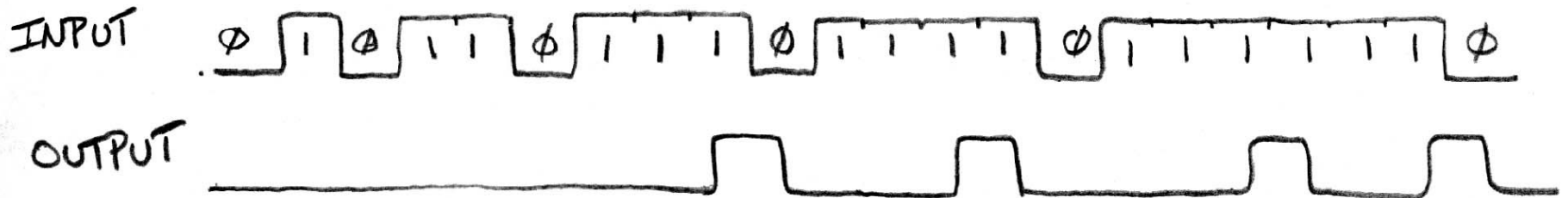
Finite State Machines (FSM) Intro

- 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.

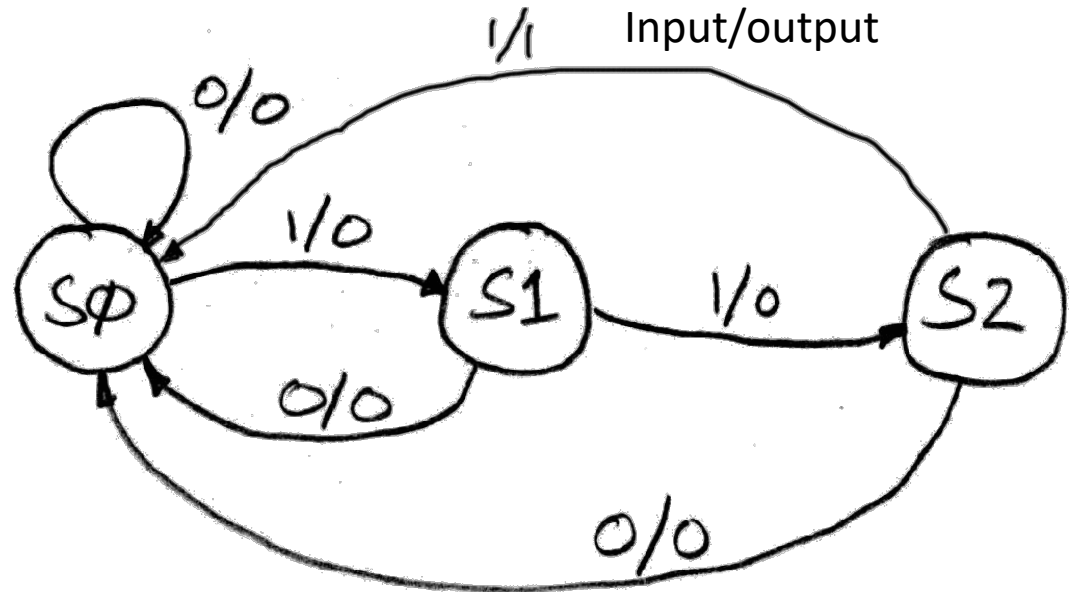


FSM Example: 3 ones...

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



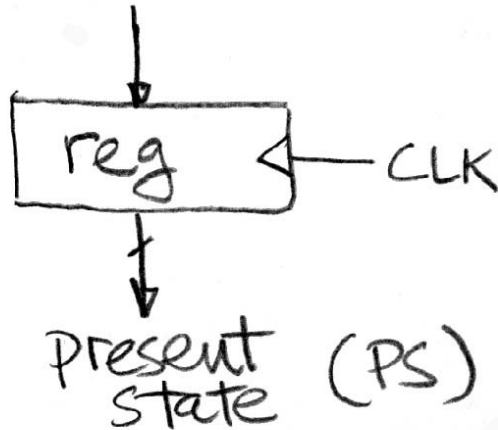
Draw the FSM...



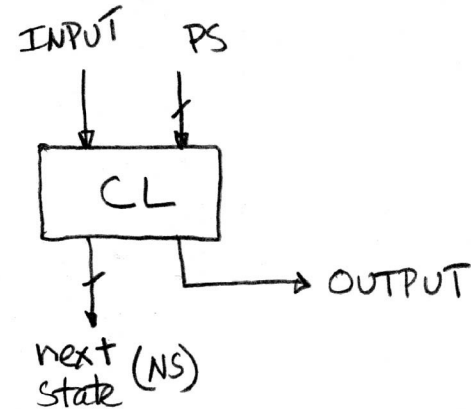
Assume state transitions are controlled by the clock:
on each clock cycle the machine checks the inputs and
moves to a new state and produces a new output...

Hardware Implementation of FSM

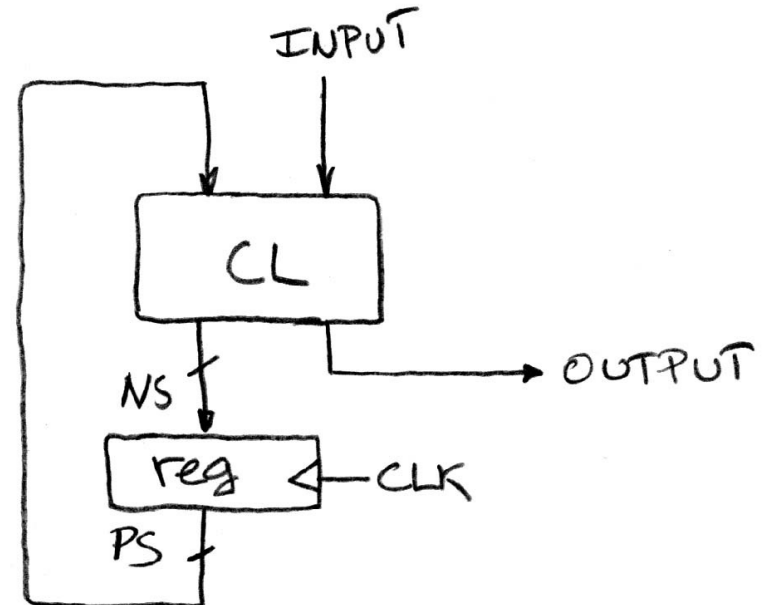
... Therefore a register is needed to hold the a representation of which state the machine is in. Use a unique bit pattern for each state.



+



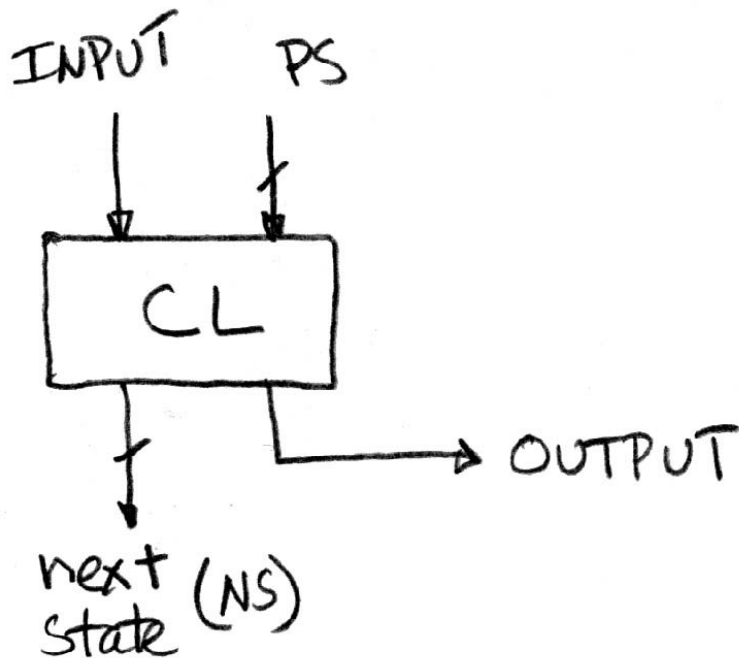
=



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

FSM Combinational Logic

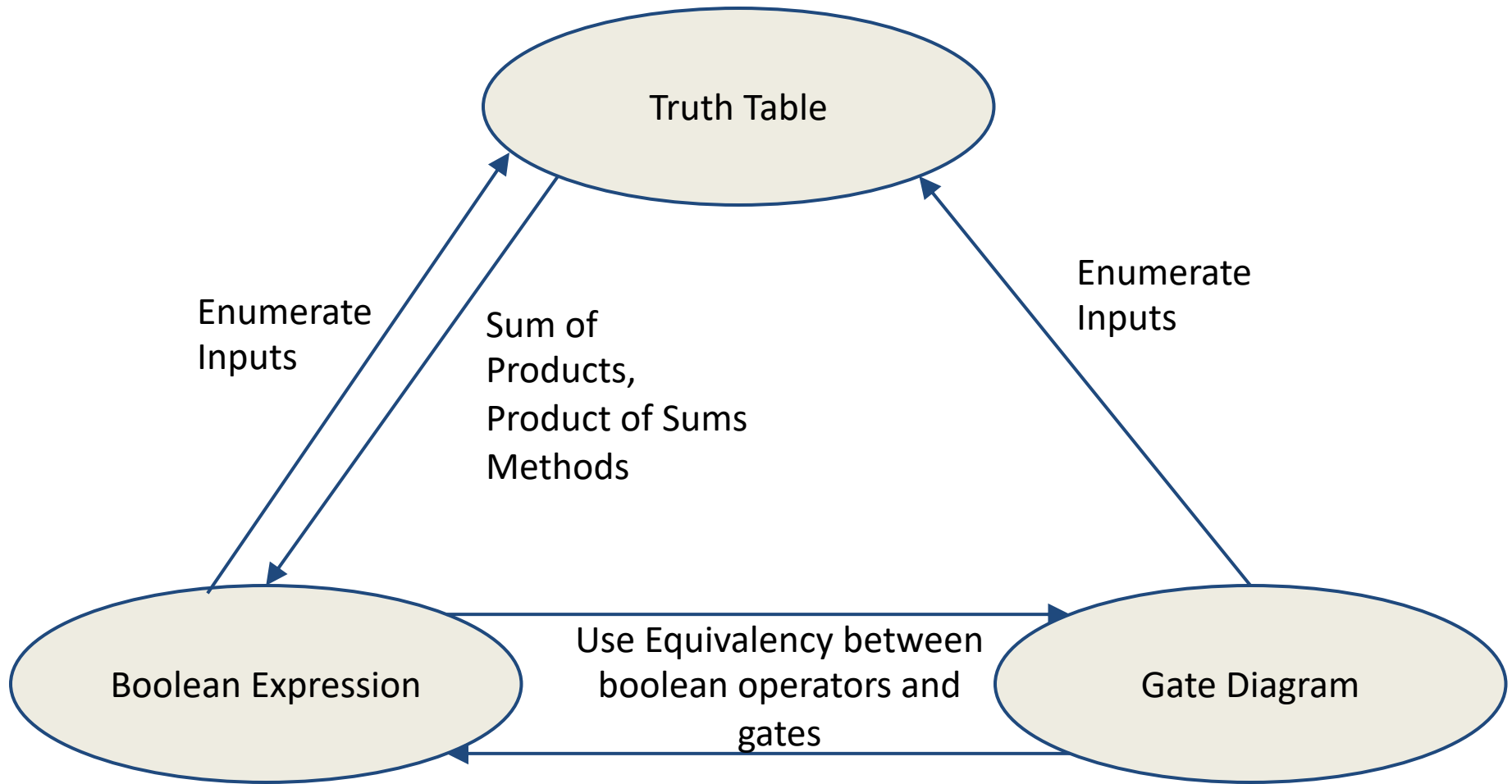
Specify CL using a truth table



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!







COMPUTER ORGANIZATION AND DESIGN

THE HARDWARE/SOFTWARE INTERFACE

 RISC-V EDITION



MK
MORGAN KAUFMANN

DAVID A. PATTERSON
JOHN L. HENNESSY

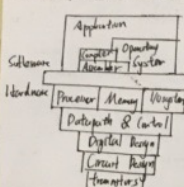
Cheat Sheet

- 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	$z = f(x, y)$ $w = f(x, y, z)$	DOMAIN: Allowed (x, y) , (x, y, z) RANGES: z, w 's
LEVEL CURVES 2-Dim $z = f(x, y) = k = \text{const.}$ CONTOUR MAPS (2-D) $w = f(x, y, z) = k = \text{const.}$ SURFACE LAYERS (3-D)	FUNCTION OF n VARIABLES $z = f(x_1, x_2, \dots, x_n)$ $f: \mathbb{R}^n \rightarrow \mathbb{C}$ 1. As a function of n real variables x_1, x_2, \dots, x_n 2. As a function of a single variable x_i 3. As a function of a single vector var. $\vec{x} = (x_1, \dots, x_n)$	ϵ-δ DEFINITION OF CONTINUITY LET f BE A FUNCTION OF 2 VARIABLES DEFINED ON A DISK w CENTER (a, b) , EXCEPT POSSIBLY (a, b) . THEN $\lim_{(x, y) \rightarrow (a, b)} f(x, y) = L$ IF FOR EVERY $\epsilon > 0$, THERE IS A CORRESPONDING $\delta > 0$ ST. IF $(x, y) \in w$ whenever $0 < \sqrt{(x-a)^2 + (y-b)^2} < \delta$ IF THE LIMIT AS A FUNCTION APPROACHES A POINT (a, b) ALONG TWO DIFFERENT PATHS IS NOT THE SAME, THE LIMIT DOES NOT EXIST @ (a, b) . $f(x, y)$ IS CONTINUOUS AT (a, b) IF THE LIMIT OF (x, y) AS $(x, y) \rightarrow (a, b)$ EXISTS.
PARTIAL DERIVATIVES $z = f(x, y)$ $f_x(x, y) = \frac{\partial z}{\partial x}$ $f_y(x, y) = \frac{\partial z}{\partial y}$	Derivatives w/ respect to one variable, holding holding the other variables constant. SAME METHODS FOR FUNCTIONS OF MORE THAN TWO VARIABLES	COMPOSITE FUNCTIONS OF CONTINUOUS FUNCTIONS ARE CONTINUOUS, AS ARE SUMS AND PRODUCTS
SECOND PARTIAL DERIVATIVES $f_{xx} = \frac{\partial^2 z}{\partial x^2}$ $f_{xy} = \frac{\partial^2 z}{\partial x \partial y}$ $f_{yx} = \frac{\partial^2 z}{\partial y \partial x}$ $f_{yy} = \frac{\partial^2 z}{\partial y^2}$	CLAIRAUT'S THEOREM IF f_{xy} AND f_{yx} ARE BOTH CONTINUOUS $f_{xy}(a, b) = f_{yx}(a, b)$ PARTIAL DIFF. EQS LAPLACE'S EQUATION $\Delta u = 0$ etc... THE WAVE EQUATION $\Delta u = \frac{\partial^2 u}{\partial t^2}$	EQUATIONS OF TANGENT PLANES TO SURFACES $z = f(x, y)$ @ (x_0, y_0, z_0) EVALUATED AT A POINT $z - z_0 = f_x(x_0, y_0)(x - x_0) + f_y(x_0, y_0)(y - y_0)$ TOTAL DIFFERENTIAL ($dz = f'(x, y) \cdot dx$ SINGLE VARIABLE) $dz = f_x(x, y)dx + f_y(x, y)dy = \frac{\partial z}{\partial x} dx + \frac{\partial z}{\partial y} dy$ INCREMENTS dx, dy, dz DIFFERENTIALS, dx, dy, dz OF R SMALL BOX, BY $dx = dx, dy = dy$ (ie change in height of surface (Δz) change in height of the tangent plane (dz))
THE CHAIN RULE SINGLE VARIABLE $y = f(x)$, $w = g(y)$, $z = f(g(x))$ $\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx}$ CASE 2 $z = f(x, y)$, $x = g(t)$, $y = h(t)$ $z = f(g(t), h(t))$ $\frac{dz}{dt} = \frac{\partial z}{\partial x} \frac{dx}{dt} + \frac{\partial z}{\partial y} \frac{dy}{dt}$ OR $w = z = f(x, y)$ $\frac{dw}{dt} = \frac{\partial w}{\partial x} \frac{dx}{dt} + \frac{\partial w}{\partial y} \frac{dy}{dt} + \frac{\partial w}{\partial z} \frac{dz}{dt}$ (if $z = f(x, y)$) CHAIN RULE: GENERAL VERSION $u = f(x_1, \dots, x_n)$ $x_j = g_j(t)$ $1 \leq j \leq n$ $\frac{du}{dt} = \frac{\partial u}{\partial x_1} \frac{dx_1}{dt} + \dots + \frac{\partial u}{\partial x_n} \frac{dx_n}{dt}$ for each $i = 1, 2, \dots, n$	IF f_x AND f_y ARE CONTINUOUS Δz $\approx dz$ (ie change in height of surface (Δz) change in height of the tangent plane (dz)) THEOREM $\Delta z = f_x(a, b)\Delta x + f_y(a, b)\Delta y - f(a, b)$ $\Delta z = f_x(a, b)\Delta x + f_y(a, b)\Delta y + \epsilon_1 \Delta x + \epsilon_2 \Delta y$ where ϵ_1 and ϵ_2 are functions of Δx and Δy that approach 0 as $(\Delta x, \Delta y) \rightarrow (0, 0)$ DEF. f IS DIFFERENTIABLE @ (a, b) FOR $z = f(x, y)$ DEPENDENCY DIAGRAMS (CASE 1 & 2) YOU CAN FIND DERIVATIVES FOR ALL THE TEMPERATURES SUBSTITUTED VARIABLE	VELOCITY m/s DISPLACEMENT m FORCE $N = kg \cdot m/s^2$ MASS kg MOMENTUM $kg \cdot m/s$ IMPULSE $N \cdot s$ WORK J ENERGY J POWER W GRAVITATIONAL POTENTIAL mgh POWER (WATT) E/t KINETIC ENERGY $0.5mv^2$ GRAVITATIONAL POTENTIAL mgh POWER (WATT) E/t GRAVITATIONAL POTENTIAL mgh POWER (WATT) E/t KINETIC ENERGY $0.5mv^2$ GRAVITATIONAL POTENTIAL mgh POWER (WATT) E/t
IMPLICIT DIFFERENTIATION $\frac{dz}{dx} = \frac{\partial z}{\partial x} + \frac{\partial z}{\partial y} \frac{dy}{dx}$ $\frac{dz}{dy} = \frac{\partial z}{\partial y} + \frac{\partial z}{\partial x} \frac{dx}{dy}$	You can always solve for y and dy/dx $\frac{\partial z}{\partial x} = \frac{F_x}{F_y} \frac{dy}{dx}$ $\frac{\partial z}{\partial y} = \frac{F_y}{F_x} \frac{dx}{dy}$	DIRECTIONAL DERIVATIVES $\text{Dir}_u f = \nabla f \cdot \vec{u}$ $\text{Dir}_u f(x, y) = f_x(x, y)u_x + f_y(x, y)u_y$ SAME FOR 3 VARIABLES $\text{Dir}_u f(x, y, z) = \nabla f(x, y, z) \cdot \vec{u}$ 3 VARIABLES THE GRADIENT VECTOR $\nabla f(x, y, z) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)$ POINT THE GRADIENT VECTOR IS ORTHOGONAL TO THE LEVEL CURVES OF A SURFACE OF HAS AS MANY COMPONENTS AS f HAS INDEPENDENT VARIABLES. $\nabla \cdot (\vec{f}_x, \vec{f}_y, \vec{f}_z)$ TO FIND THE NORMAL (AND LATER TANGENT) PLANE TO A SURFACE, LET THAT SURFACE BE THE LEVEL SET OF SOME HIGHER DIMENSIONAL FUNCTION. THEN THE GRADIENT OF THE HIGHER D FUNCTION IS \perp TO YOUR SURFACE $\vec{u} \cdot \nabla f = 0$ $\vec{u} = (u_x, u_y, u_z)$ $\vec{u} \cdot \nabla f = u_x f_x + u_y f_y + u_z f_z = 0$ $\vec{u} = (2, 1, 3)$ IS A NORMAL VECTOR TO THE $z = 2 - 3x + 4y + z = 2$
TANGENT PLANE TO A LEVEL SURFACE $\nabla f \perp$ TO $f = \text{const.}$ $F_x(x - x_0) + F_y(y - y_0) + F_z(z - z_0) = 0$	NORMAL LINE TO LEVEL SURFACE $\frac{dx}{F_x} = \frac{dy}{F_y} = \frac{dz}{F_z}$ ALL 0 $\frac{dx}{F_x} = \frac{dy}{F_y} = \frac{dz}{F_z}$ ALL ∞	NEWTON'S RULES 1. A body's speed remains constant if the net force acting on it is zero (balanced). Acceleration is directly proportional to net force and inversely proportional to mass. Forces exist in action/reaction pairs. member that when gravity is acting against the sum of forces, it will be factored as a negative $m/s \times 3.6 \rightarrow km/h$ $\div 3.6$
SPECIAL CASE $z = f(x, y)$ $F_x(x, y, z) = F_y(x, y, z) = 0$ LEVEL SURFACE W/ NO OLD DEFINITION THEN $F_x = -1$, $F_y = (f_x, f_y, -1)$ AND TANGENT PLANE $z = z_0 + f_x(x - x_0) + f_y(y - y_0)$	MAXIMUM AND MINIMUM VALUES $z = f(x, y)$ $f_x(a, b) = 0$ $f_y(a, b) = 0$ $\nabla f(a, b) = 0$ NECESSARY BUT NOT SUFFICIENT TO GUARANTEE A MAX. OR MIN. THEN APPLY THE 2ND DERIVATIVE TEST $\text{Hess } f_x, f_y, f_{xy}$ $D = \begin{bmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{bmatrix} = \text{Hess } f = \text{Hess } f - (f_{xy})$ $\Delta > 0$, $\det D > 0$ LOCAL MIN. $\Delta < 0$, $\det D > 0$ LOCAL MAX. $\Delta < 0$ SADDLEPT. $\det D = 0$ UNDETERM.	DIFFUSION OPACQUE: doesn't allow light to pass thru TRANSPARENT: allows most light to pass TRANSLUCENT: allows light through but distorts its path \rightarrow blur the image. No material can allow 100% of incident light to pass through. REFRACTION is the bending of the path of light due to a change in speed as it enters a medium of different optical density. denser \rightarrow high refractive index slower medium = refracted toward faster medium = refracted away less dense
FINDING ABSOLUTE MAX. AND MINS. FOR A CLOSED BOUNDARY 1. Find values of f at the critical points of f in D 2. Find the endpoint values of f on the boundary of D 3. The largest value from 1, 2 is the ABS. MAX, the smallest is the ABS. MIN.	MAXIMIZING AND MINIMIZING Set f or a function of two variables of the form $z = f(x, y)$ and such domain usual constraint	DIFFRACTION is the bending of light around obstacles and through APERTURES. LAW OF REFLECTION : the angle of incidence is equal to the angle of reflection. The incident and reflected beams, and the normal, all lie on the same plane. CONSTRUCTIVE INTERFERENCE will occur all along the path. DESTRUCTIVE INTERFERENCE $A + A = 2A$ $A - A = 0$ DIFFRACTION is the bending of light around obstacles and through APERTURES. LAW OF REFLECTION : the angle of incidence is equal to the angle of reflection. The incident and reflected beams, and the normal, all lie on the same plane. CONSTRUCTIVE INTERFERENCE will occur all along the path. DESTRUCTIVE INTERFERENCE $A + A = 2A$ $A - A = 0$

WAVELENGTH m
SPEED OF WAVE m/s
PERIOD s
AMPLITUDE m
PHASE rad
INTERFERENCE
 Constructive: $A + A = 2A$
 Destructive: $A - A = 0$
DIFFRACTION is the bending of light around obstacles and through APERTURES.
LAW OF REFLECTION: the angle of incidence is equal to the angle of reflection. The incident and reflected beams, and the normal, all lie on the same plane.
CONSTRUCTIVE INTERFERENCE will occur all along the path.
DESTRUCTIVE INTERFERENCE
 $A + A = 2A$
 $A - A = 0$
EMISSION
 Spectral clay clean light spectrum
 cooler stars have more complex elements
OPACITY: doesn't allow light to pass thru
TRANSPARENT: allows most light to pass
TRANSLUCENT: allows light through but distorts its path \rightarrow blur the image.
 No material can allow 100% of incident light to pass through.
REFRACTION is the bending of the path of light due to a change in speed as it enters a medium of different optical density.
 denser \rightarrow high refractive index
 slower medium = refracted toward
 faster medium = refracted away
 less dense
GRAVITY
 member that when gravity is acting against the sum of forces, it will be factored as a negative
 $m/s \times 3.6 \rightarrow km/h$
 $\div 3.6$
GRAVITATIONAL POTENTIAL mgh
POWER (WATT) E/t
KINETIC ENERGY $0.5mv^2$
GRAVITATIONAL POTENTIAL mgh
POWER (WATT) E/t
KINETIC ENERGY $0.5mv^2$
GRAVITATIONAL POTENTIAL mgh
POWER (WATT) E/t

Old School Machine Structure



When C program starts
 - C executable and is loaded into memory by OS (opening system)
 - OS put up stack, then calls into application
 - Run-time info: variables memory and the library
 - This will give processor some manual

Valid Pointer Arithmetic
 - Add an integer to a pointer
 - Subtract 2 pointers for the same array
 - Compare two pointers (C, C++, ...)
 - Capture pointer to null
 - Add two pointers / multiply
 - int main (int argc, char * argv[])
 - argc contains the number of things in the command line
 - argv is a pointer to an array of strings
 - The arguments as strings

Five Fundamental Steps in Calling a Function
 1. Put parameters in a place where function can access them
 2. Transfer control to function
 3. Access local storage
 4. Execute needed code
 5. Return result value in a place where calling code can access it and restore any resources you used
 6. Return control to point of origin, since a function can be called from several parts in a program.

New School Machine Structure

Parallel Requests
 Parallel Threads
 Parallel Data
 Hardware descriptors

6. Guest Ideas in Computer Architecture
 1. Abstraction
 Layers of Representation/Interpretation
 2. Moore's Law
 Designing through trends
 3. Principle of Locality
 Memory Hierarchy
 4. Parallelism
 5. Performance Measurement and Improvement
 6. Dependability via Redundancy

Two-Complement Representation
 - treats 0 as positive
 - 31-bit word represents 2^{32} values from -2^{31} to $2^{31}-1$

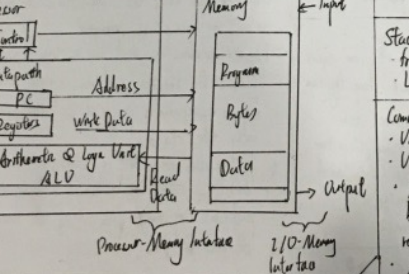
位运算: 从下往上依次进位
 1. 位运算
 2. 强制类型转换
 3. 自增自减
 4. 左移右移
 5. 左移右移
 6. 左移右移
 7. 左移右移
 8. 左移右移
 9. 左移右移

Program's address space / Memory Address
 Stack
 Heap
 Static data
 Code

Stack: local variables inside functions, grow downwards
 Heap: space requested for dynamic data via malloc(), grows dynamically
 Static data: variable declared outside functions, loaded when program starts, can be modified
 Code: loaded when program starts, doesn't change

Save argument registers
 Save return address
 Save saved registers
 Local variables and functions
 Stack
 Dynamic data
 Static data
 Heap
 Reserved

Components of a Computer



foo.c → cpp → fo.o → compiler
 - cpp replaces comments with a single space
 - cpp comments begins with #

Standard Type
 (unsigned) char signed char
 (unsigned) int 4
 (unsigned) short 2 + long x 2
 (unsigned) long 4
 float 4 byte
 double 8 byte / long / long long

10. 位运算
 11. 位运算与
 12. 位运算
 13. ? 右到左
 14. += -= * = / = % = << = >> =
 15. / 左到右
 long long 8
 long long 8
 long long 8

Stack
 - free when function returns
 - last in first out
 R format: used for instructions with immediate, lw and sw and branches (every other one)
 Opcode rs rt rd shamt funct
 opcode = 0
 L format: 5-bit field only represents numbers up to 31
 If machine has multiple, then use at most 2 registers
 used for lw, sw, jal, jalr, branches and with immediate
 Delay with larger immediate
 Label Upper lui
 Lower \$t0 \$t1 OADR ABCD
 \$t0 \$t1 OADR AB
 \$t0 \$t1 OADR
 \$t0 \$t1

Common Memory Problems
 - Very uninitialized values
 - Very memory that you don't own
 - Important use of free/malloc by memory with the pointer handle returned by malloc/calloc
 - Memory leaks
 Levels of Representation / Interpretation
 High level language
 Assembly Language
 Machine Language
 Machine Interpretation
 Hardware Architecture Description
 Architecture Interpretation
 Logic Level Description

ADDRESSING MODES AND OPERATIONAL MODES

ADDRESSING MODES:

- BASE REGISTER
- BASE PLUS DISPLACEMENT
- BASE REGISTER PLUS DISPLACEMENT
- BASE PLUS REGISTER
- BASE REGISTER PLUS REGISTER
- BASE PLUS IMMEDIATE
- BASE REGISTER PLUS IMMEDIATE
- BASE PLUS INDEXED
- BASE REGISTER PLUS INDEXED
- BASE PLUS INDEXED PLUS DISPLACEMENT
- BASE REGISTER PLUS INDEXED PLUS DISPLACEMENT
- BASE PLUS INDEXED PLUS REGISTER
- BASE REGISTER PLUS INDEXED PLUS REGISTER
- BASE PLUS INDEXED PLUS REGISTER PLUS DISPLACEMENT
- BASE REGISTER PLUS INDEXED PLUS REGISTER PLUS DISPLACEMENT

OPERATIONAL MODES:

- BASE
- BASE PLUS DISPLACEMENT
- BASE REGISTER PLUS DISPLACEMENT
- BASE PLUS REGISTER
- BASE REGISTER PLUS REGISTER
- BASE PLUS IMMEDIATE
- BASE REGISTER PLUS IMMEDIATE
- BASE PLUS INDEXED
- BASE REGISTER PLUS INDEXED
- BASE PLUS INDEXED PLUS DISPLACEMENT
- BASE REGISTER PLUS INDEXED PLUS DISPLACEMENT
- BASE PLUS INDEXED PLUS REGISTER
- BASE REGISTER PLUS INDEXED PLUS REGISTER
- BASE PLUS INDEXED PLUS REGISTER PLUS DISPLACEMENT
- BASE REGISTER PLUS INDEXED PLUS REGISTER PLUS DISPLACEMENT

int division A328
int multiply A327
macro (hlong) B28
performance of CPU A325

represent i/o B23
represent denoms B25
represent float IEEE754 B22
represent floating point FP B21
represent fraction A324
represent hexadecimal A323
represent scientific notation A322
system performance evaluation A321

A-1 Introduction

1. Introduction: why use data and how to use it

Architecture: AX (addresses) every 2 bytes

Registers: 16 many data registers -> bus

Performance: throughput, instructions

Cache: set of cache (transistor) / on chip

Reliability via Redundancy: more things have same data, one bad, not all bad

2. Numbering: digits 0, 1, 2, ... & R, B, D, E, P

3. Overview - (bus) components

4. Components of computer

5. Computer

6. Advantages

7. Disadvantages

8. Example

9. Application

10. Performance

11. Memory

12. Cache

13. Bus

14. Addressing

15. Instruction

16. Execution

17. Control

18. Logic

19. Hardware

20. Software

A-2 Introduction

1. Introduction: why use data and how to use it

Architecture: AX (addresses) every 2 bytes

Registers: 16 many data registers -> bus

Performance: throughput, instructions

Cache: set of cache (transistor) / on chip

Reliability via Redundancy: more things have same data, one bad, not all bad

2. Numbering: digits 0, 1, 2, ... & R, B, D, E, P

3. Overview - (bus) components

4. Components of computer

5. Computer

6. Advantages

7. Disadvantages

8. Example

9. Application

10. Performance

11. Memory

12. Cache

13. Bus

14. Addressing

15. Instruction

16. Execution

17. Control

18. Logic

19. Hardware

20. Software

A-3 Introduction

1. Introduction: why use data and how to use it

Architecture: AX (addresses) every 2 bytes

Registers: 16 many data registers -> bus

Performance: throughput, instructions

Cache: set of cache (transistor) / on chip

Reliability via Redundancy: more things have same data, one bad, not all bad

2. Numbering: digits 0, 1, 2, ... & R, B, D, E, P

3. Overview - (bus) components

4. Components of computer

5. Computer

6. Advantages

7. Disadvantages

8. Example

9. Application

10. Performance

11. Memory

12. Cache

13. Bus

14. Addressing

15. Instruction

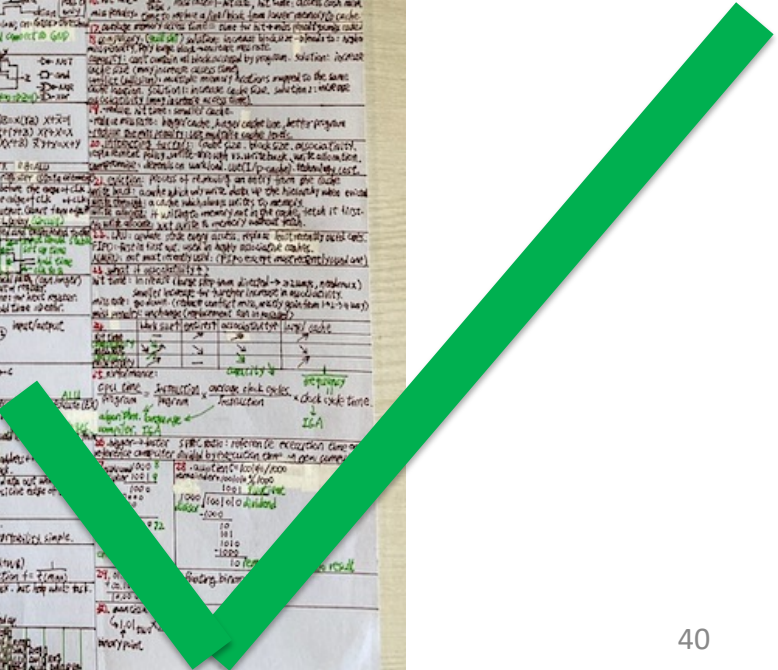
16. Execution

17. Control

18. Logic

19. Hardware

20. Software



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



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

2. BOOT SIFT VM

Load: SIFT Workstation VM Appliance
Download SIFT Workstation Installation

4. CONNECT IMAGE TO SIFT

Plug hard drive to physical host and attach to SIFT VM

3. ELEVATE PRIVS
Login: `sansforensics`
Password: `forensics`

3. ELEVATE PRIVS
`$ sudo su`

- 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 containing 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
- `tlf` - Body file in the TLF format
- `volatility` - Volatility output files (psscan2, socksca2, ...)
- `win_link` - Windows shortcut file (or a link file)
- `wmipro` - WMI 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 LEKSTROM, KRISTINN GUDJONSSON, KPMG & SANS
QUESTIONS/FEEDBACK-CONTACT US!!!

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

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):

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

`# mount -t ntfs -o ro,loop,show_sys_files,streams_interface=windows,offset=#### image.dd /mnt/windows_mount/`

MOUNT TO MOUNT POINT

SINGLE IMAGE

`# mount -t ntfs -o ro,loop,show_sys_files,streams_interface=windows,offset=#### image.dd /mnt/windows_mount/`

SPLIT IMAGE (2 step process)

`# affuse image.001 /mnt/aff`
`# mount -t ntfs-3g -o loop,ro,show_sys_files,streams_interface=windows,offset=#### /mnt/aff/<image> /mnt/windows_mount/`

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

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.

HOW TO CALCULATE THE OFFSET FOR MOUNTING

- Run `mmls` to query partition layout `# mmls image.E01`
- Identify partition and byte offset
- (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/`

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 plugins)

XP `# log2timeline-sift -z EST5EDT -p 0 -i partition`

WIN7 `# log2timeline-sift -win7 -z EST5EDT -i partition`

OTHER USAGE EXAMPLES:

Display list of available plugins: `# log2timeline -f list`
Run log2timeline using only specific plugins: `# log2timeline-sift -p prefetch -z EST5EDT -i image.dd`
Help (man pages): `# log2timeline -h`

ANNUAL "MICRO" TIMELINE CREATION

`log2timeline-sift -o -z [TIMEZONE] [-f FORMAT] [-z TIMEZONE] [-o OUTPUT MODULE] [-w LOG_FILE/LOG_DIR [-] [FORMAT FILE OPTIONS]]`

EXTRACT METADATA (using log2timeline or fls)

Extract metadata w/ log2timeline from mounted file system:
`# log2timeline -f mft -o mactime -r -z EST5EDT -w mft.body /mnt/volume/`
OR Extract metadata using Sleuthkit:
`# fls -m "" -o offset -d dd > fls.body`
Convert body file format to CSV format w/ mactime:
`# mactime -b fls.body -d > log2timeline.csv`

ARTIFACTS (run I2I on mounted file system with plugins recursively)

Extract artifacts w/ log2timeline and run mactime on mounted file system:
`# log2timeline -f firefox3,chrome -o mactime -r -z EST5EDT -w web.body /mnt/volume/`
Convert body file format to CSV format w/ mactime:
`# mactime -b log2timeline.body -d > log2timeline.csv`

8. CSV FILE OUTPUT (/cases/timeline-output-folder)

`-date`: Time of the event, in the format of MM/DD/YYYY
`-time`: Time of day, expressed in a 24h format, HH:MM:SS
`-tz`: The timezone that was used to call the tool with.
`-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:
`I2t_process -b timeline.csv MM-DD-YYYY..MM-DD-YYYY > filtered.csv`
Filter timeline with keyword list (one term per line in keywords.txt):
`I2t_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:"`
Find MountPoints2 entries that reference E Drive
`grep "MountPoints2 key" timeline.csv | grep "E drive"`
`grep "USB 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

7-A & 7-B

HELP? OPTIONS? USAGE?

`log2timeline -help`
`log2timeline-sift -help`
`I2t_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 an ASCII output
`-TLNX` - Format used by some of H Carvey tools, expressed as a XML document

10. CONNECT TO SIFT

- Settings -> Shared Folders -> Always Enabled (Check)
- SIFT Desktop -> VMware-Shared-Drive
- Access from a Win Machine \\SIFTWORKSTATION

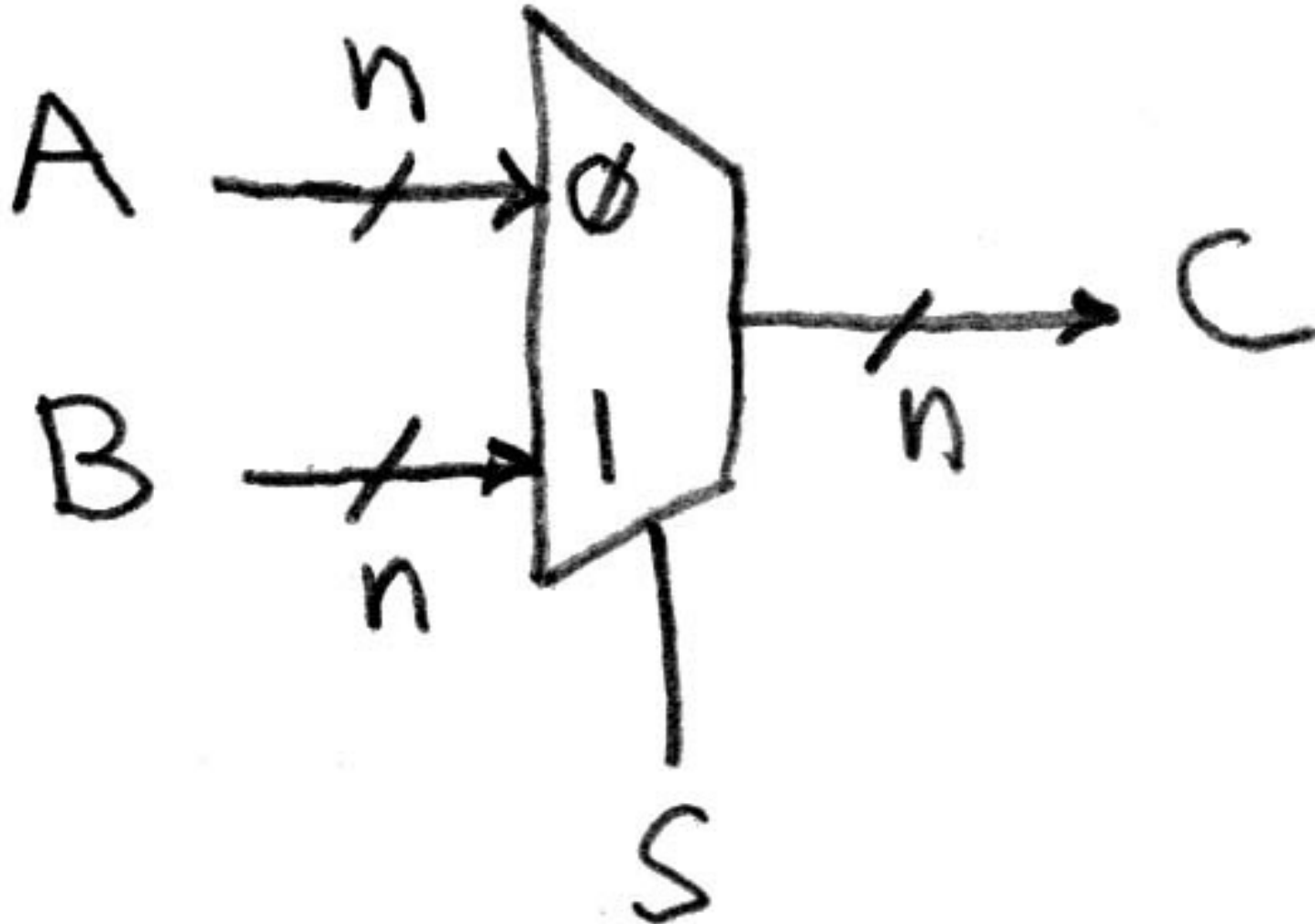
11. REVIEW TIMELINE

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

Building Standard Functional Units

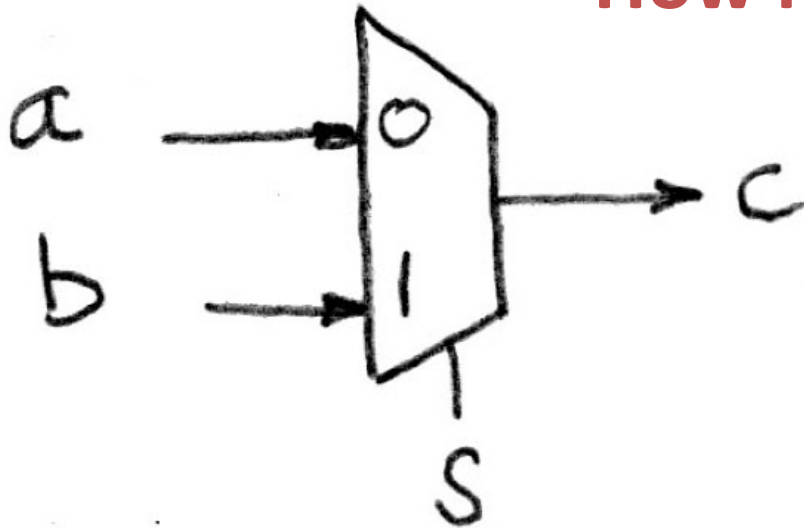
- Data multiplexers
- Arithmetic and Logic Unit
- Adder/ Subtractor

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



N instances of 1-bit-wide mux

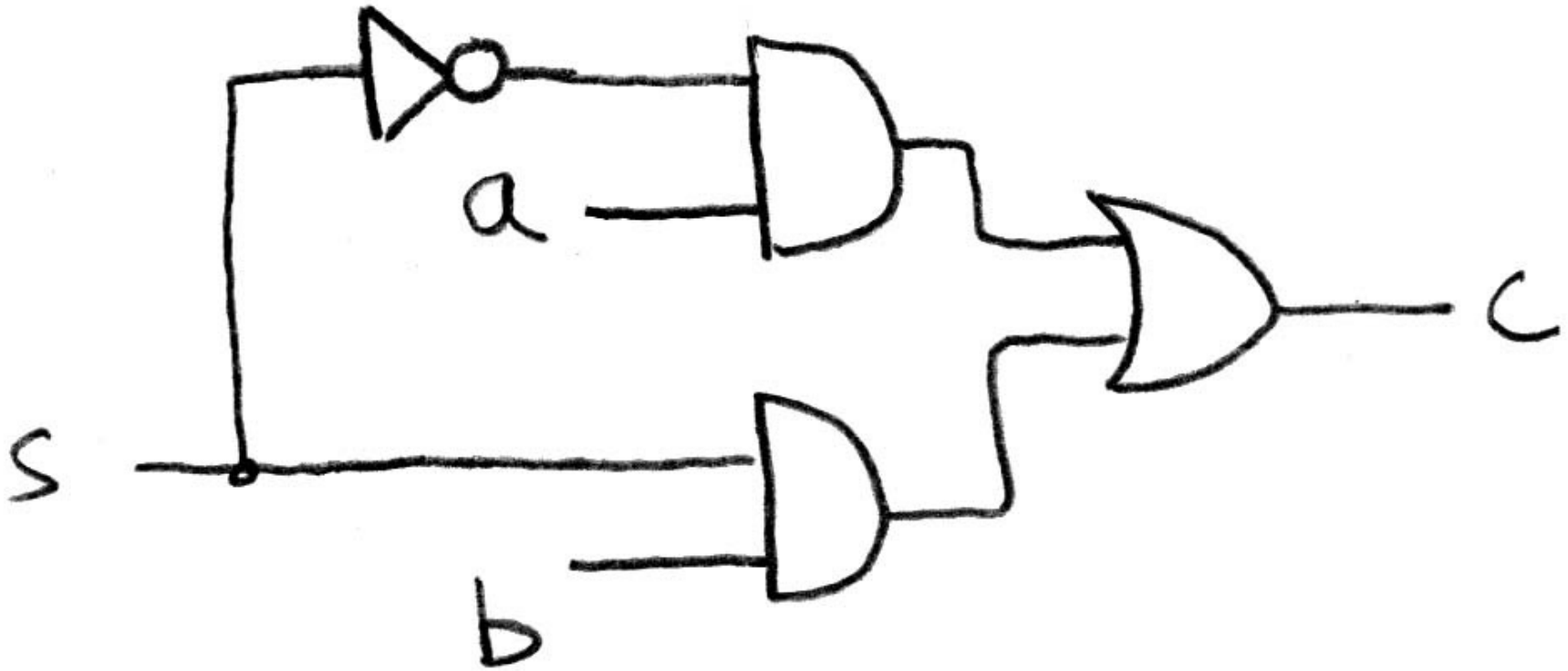
How many rows in TT?



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

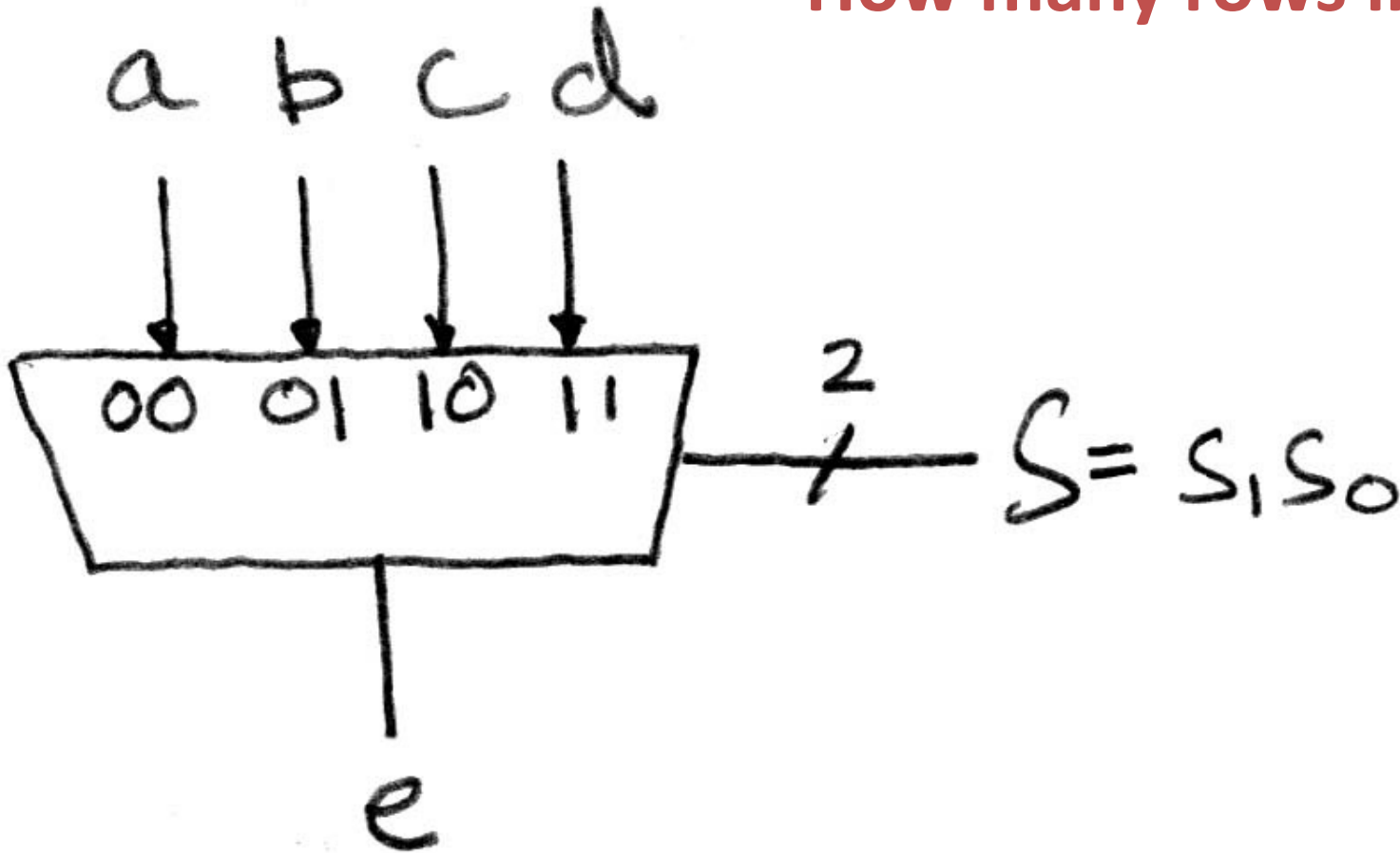
How do we build a 1-bit-wide mux?

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



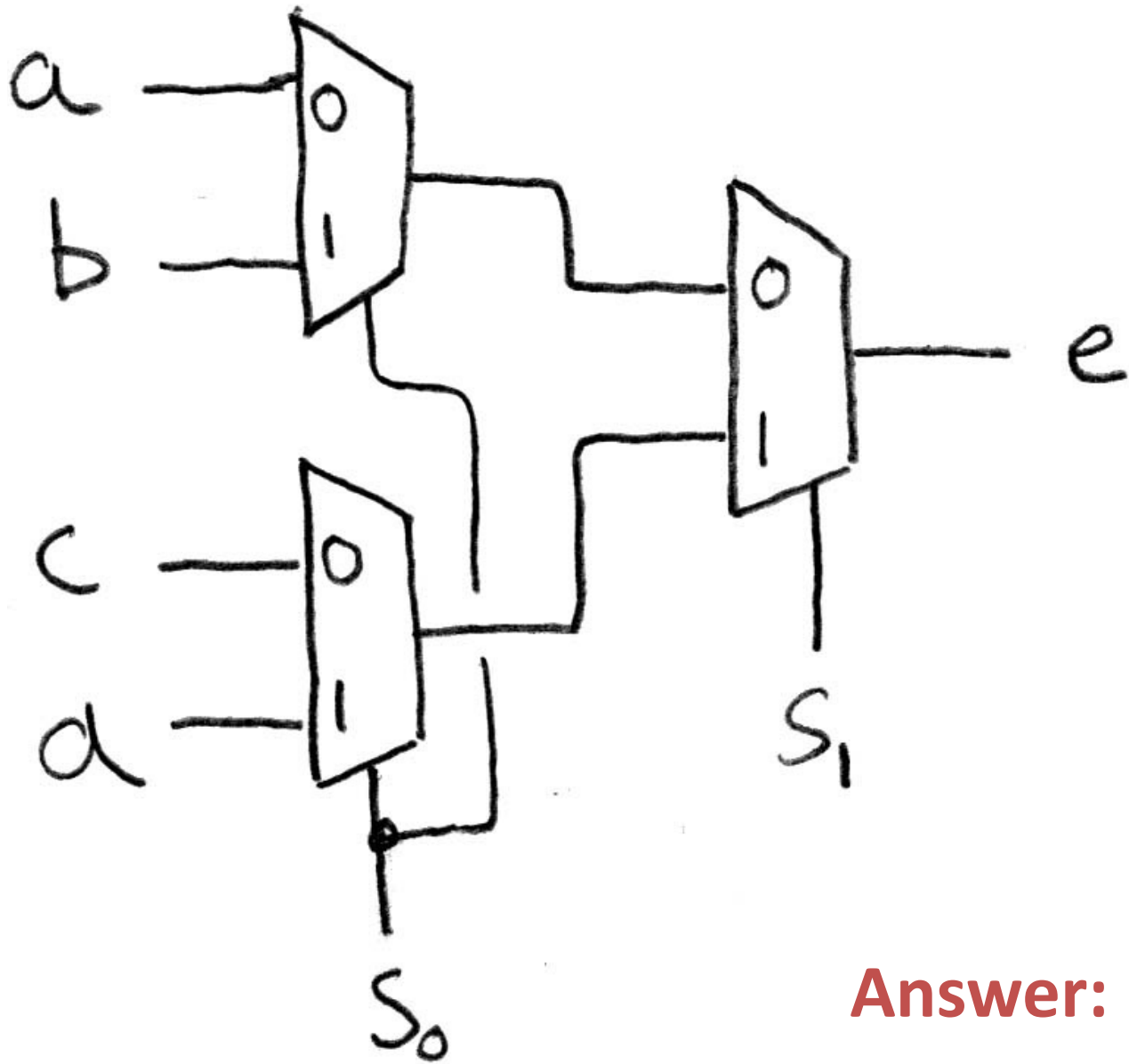
4-to-1 multiplexer?

How many rows in TT?



$$e = \bar{s}_1\bar{s}_0a + \bar{s}_1s_0b + s_1\bar{s}_0c + s_1s_0d$$

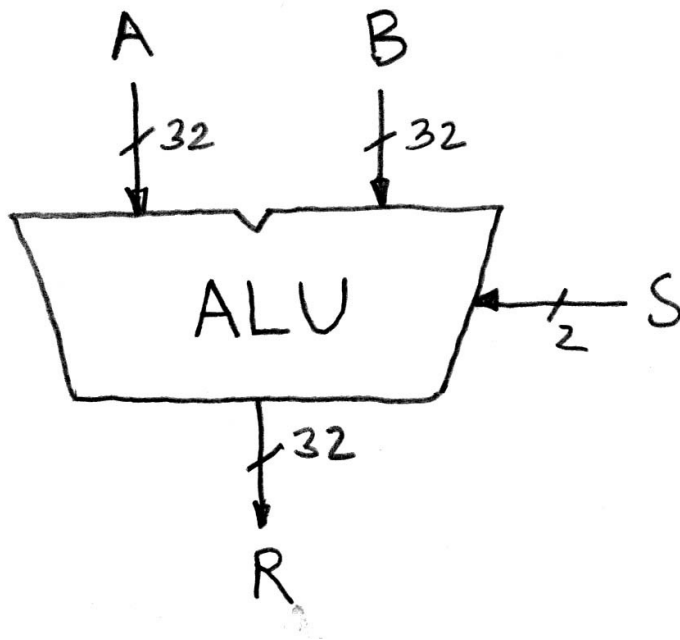
Another way to build 4-1 mux?



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



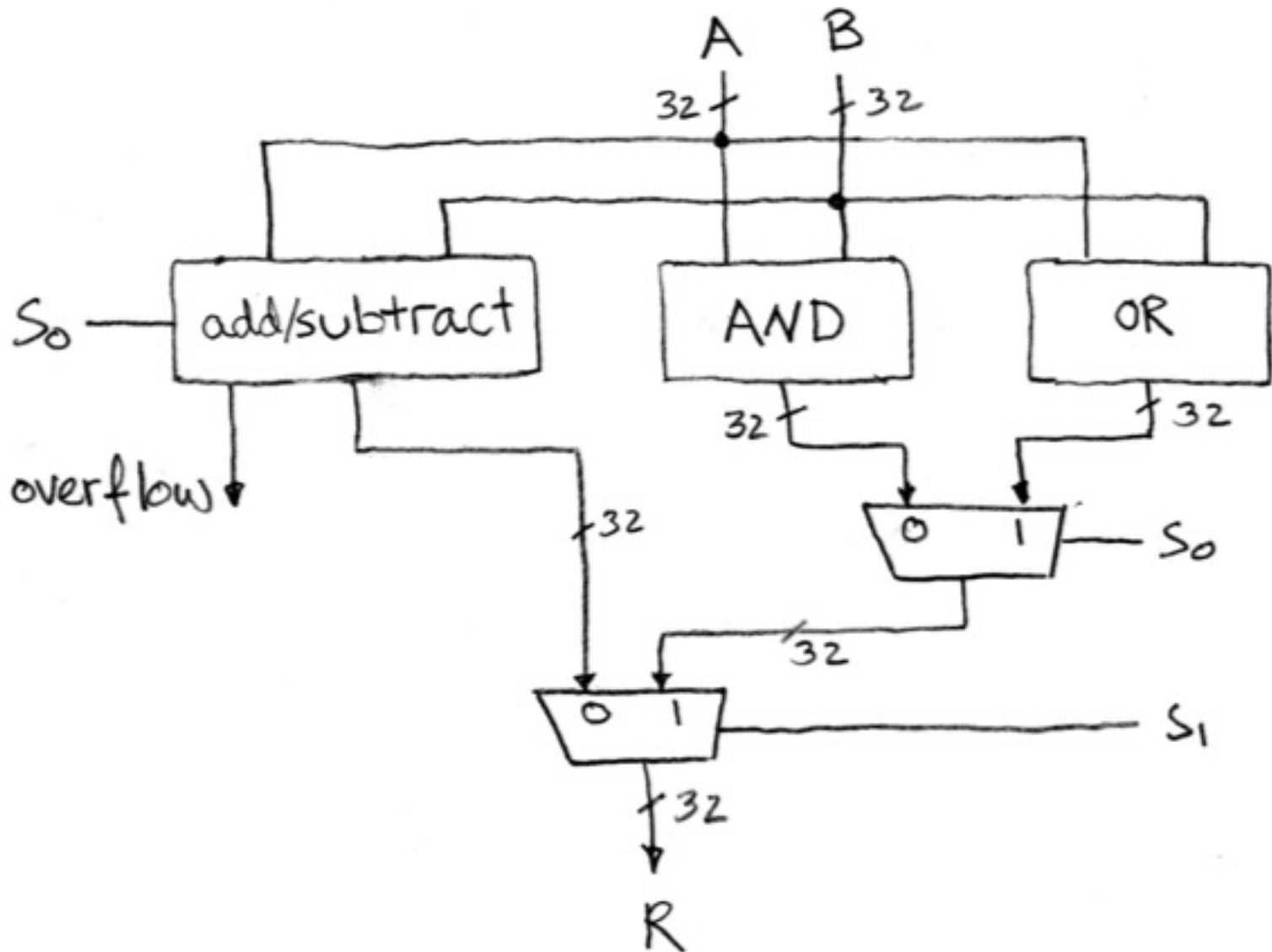
when $S=00$, $R=A+B$

when $S=01$, $R=A-B$

when $S=10$, $R=A \text{ AND } B$

when $S=11$, $R=A \text{ OR } B$

Our simple ALU



Question

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

A: $F = xy + x(\sim y)$

B: $F = xy + (\sim x)y + (\sim x)(\sim y)$

C: $F = (\sim x)y + x(\sim y)$

D: $F = xy + (\sim x)y$

E: $F = (x+y)(\sim x+\sim y)$

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...

$$\begin{array}{rcccc} & \mathbf{a_3} & \mathbf{a_2} & \mathbf{a_1} & \mathbf{a_0} \\ + & \mathbf{b_3} & \mathbf{b_2} & \mathbf{b_1} & \mathbf{b_0} \\ \hline \mathbf{s_3} & \mathbf{s_2} & \mathbf{s_1} & \mathbf{s_0} & \end{array}$$

$\mathbf{a_0}$	$\mathbf{b_0}$	$\mathbf{s_0}$	$\mathbf{c_1}$
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)...

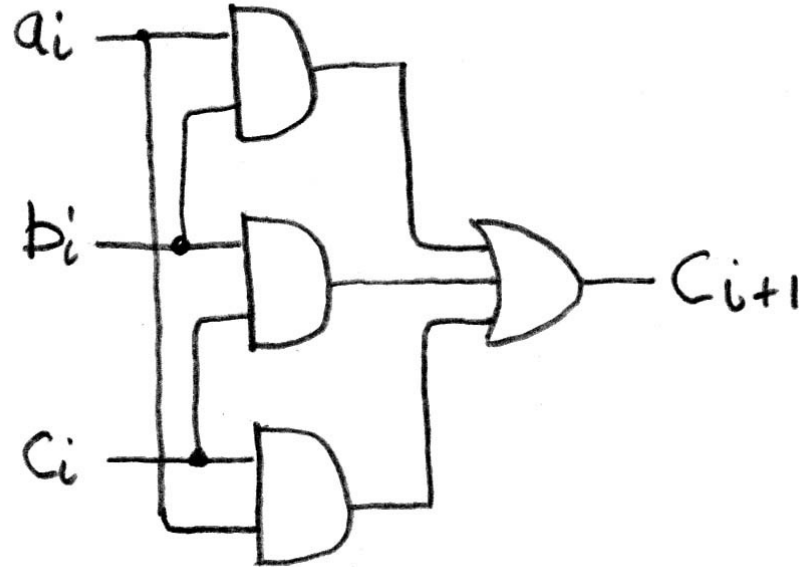
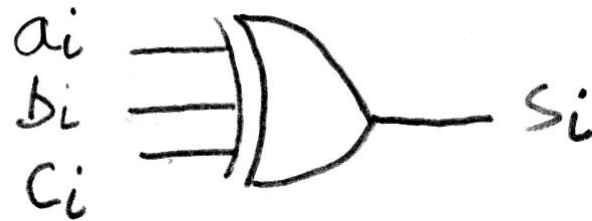
				a_i	b_i	c_i	s_i	c_{i+1}
				0	0	0	0	0
				0	0	1	1	0
				0	1	0	1	0
				0	1	1	0	1
				1	0	0	1	0
				1	0	1	0	1
				1	1	0	0	1
				1	1	1	1	1

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

$$s_i =$$

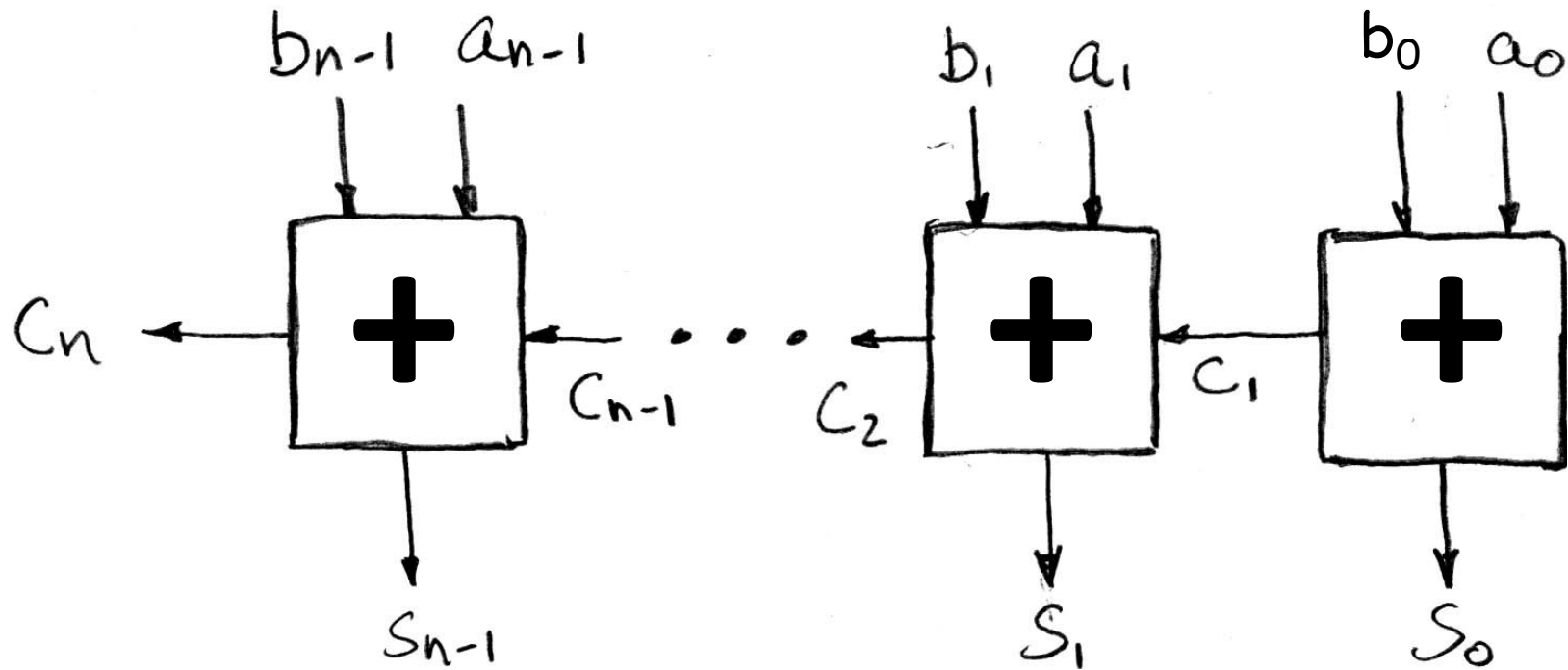
$$c_{i+1} =$$

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



$$S_i = \text{XOR}(a_i, b_i, c_i)$$
$$C_{i+1} = \text{MAJ}(a_i, b_i, c_i) = a_i b_i + a_i c_i + b_i c_i$$

N 1-bit adders \Rightarrow 1 N-bit adder

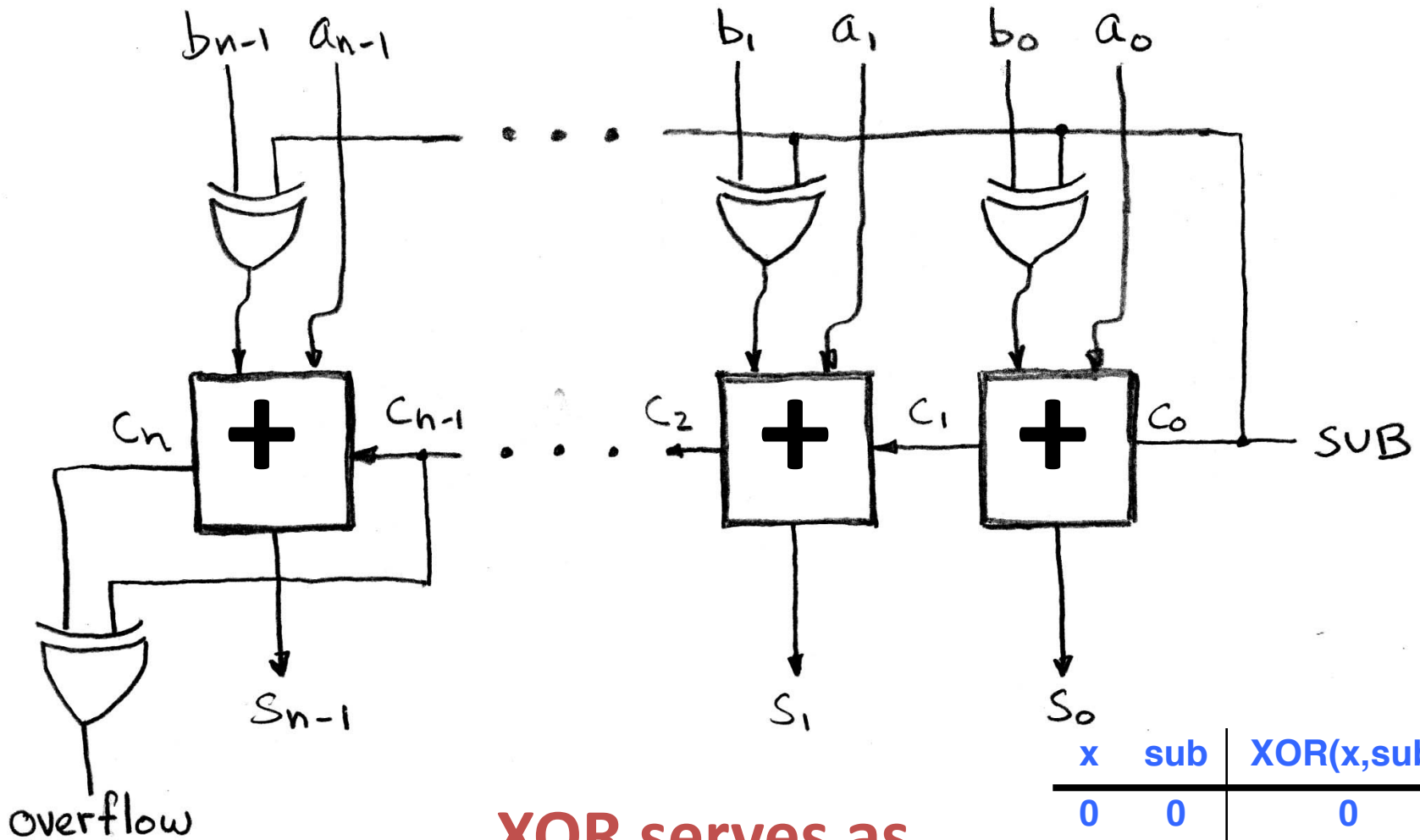


What about overflow?

Overflow = c_n ?

Extremely Clever Subtractor:

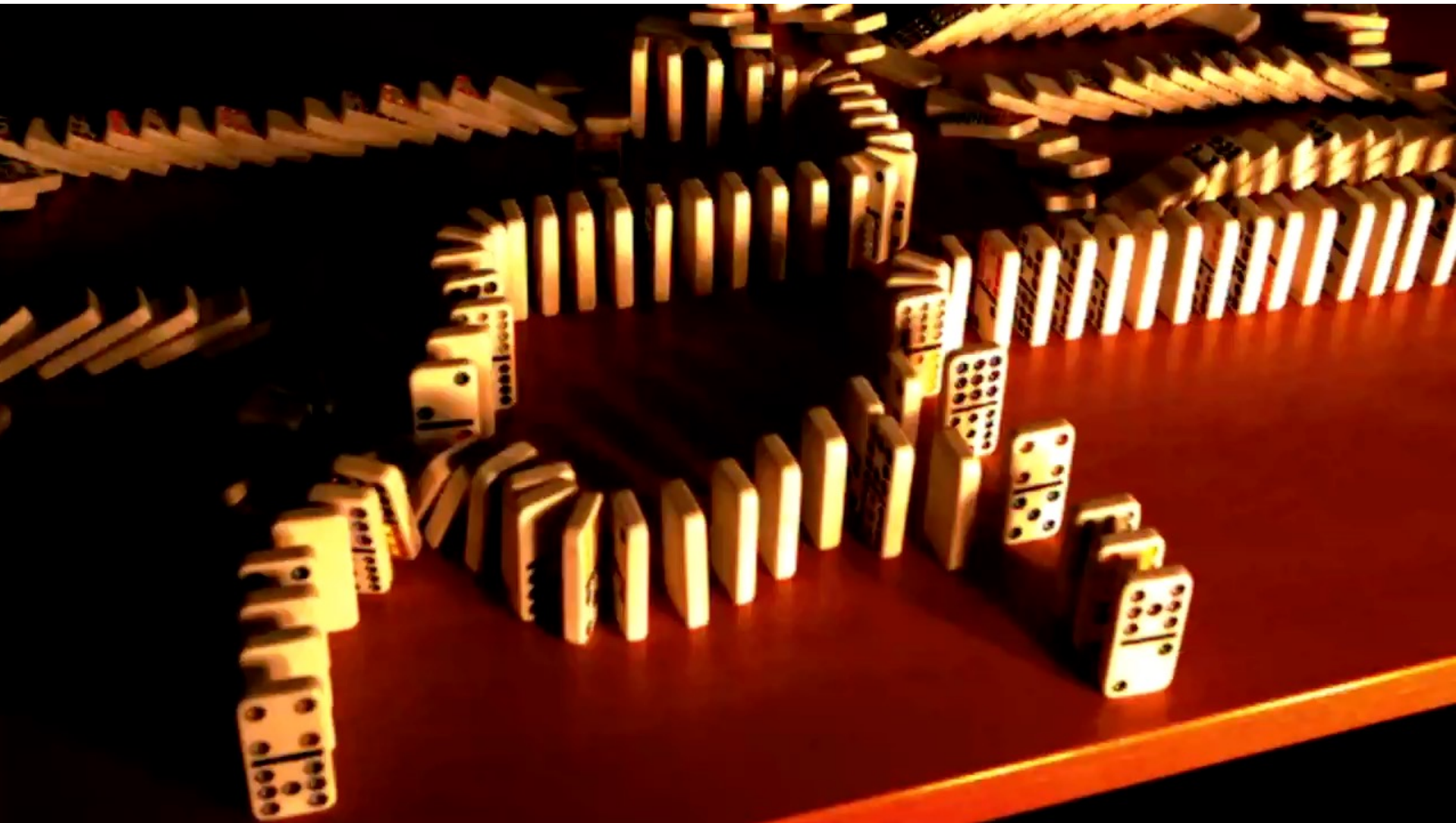
$$s = a + (-b)$$



**XOR serves as
conditional inverter!**

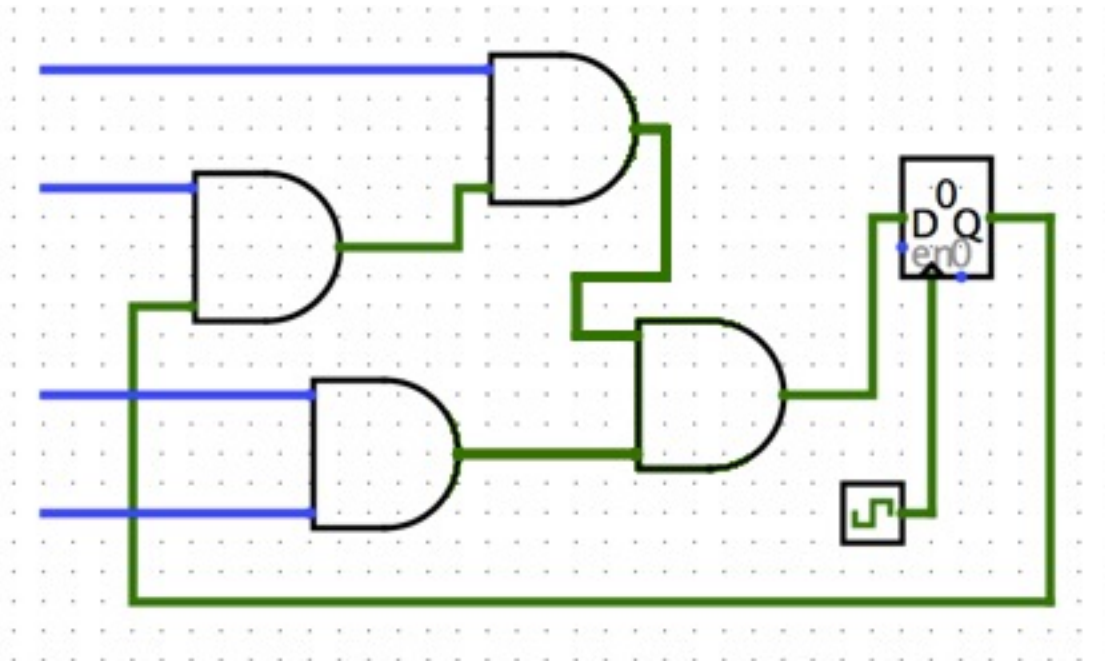
x	sub	XOR(x,sub)
0	0	0
0	1	1
1	0	1
1	1	0

Domino Adder



Explanation: <https://www.youtube.com/watch?v=INuPy-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