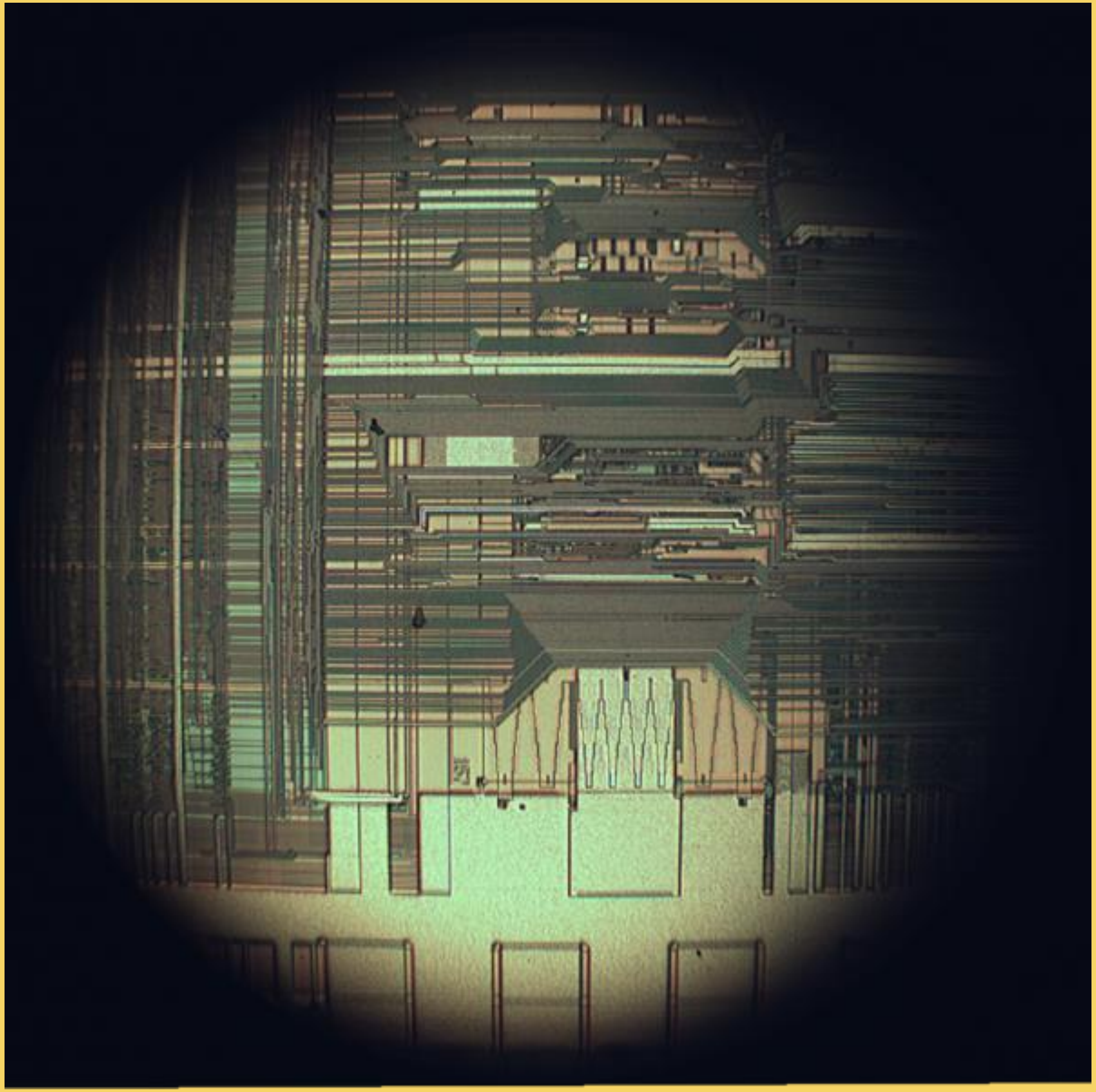
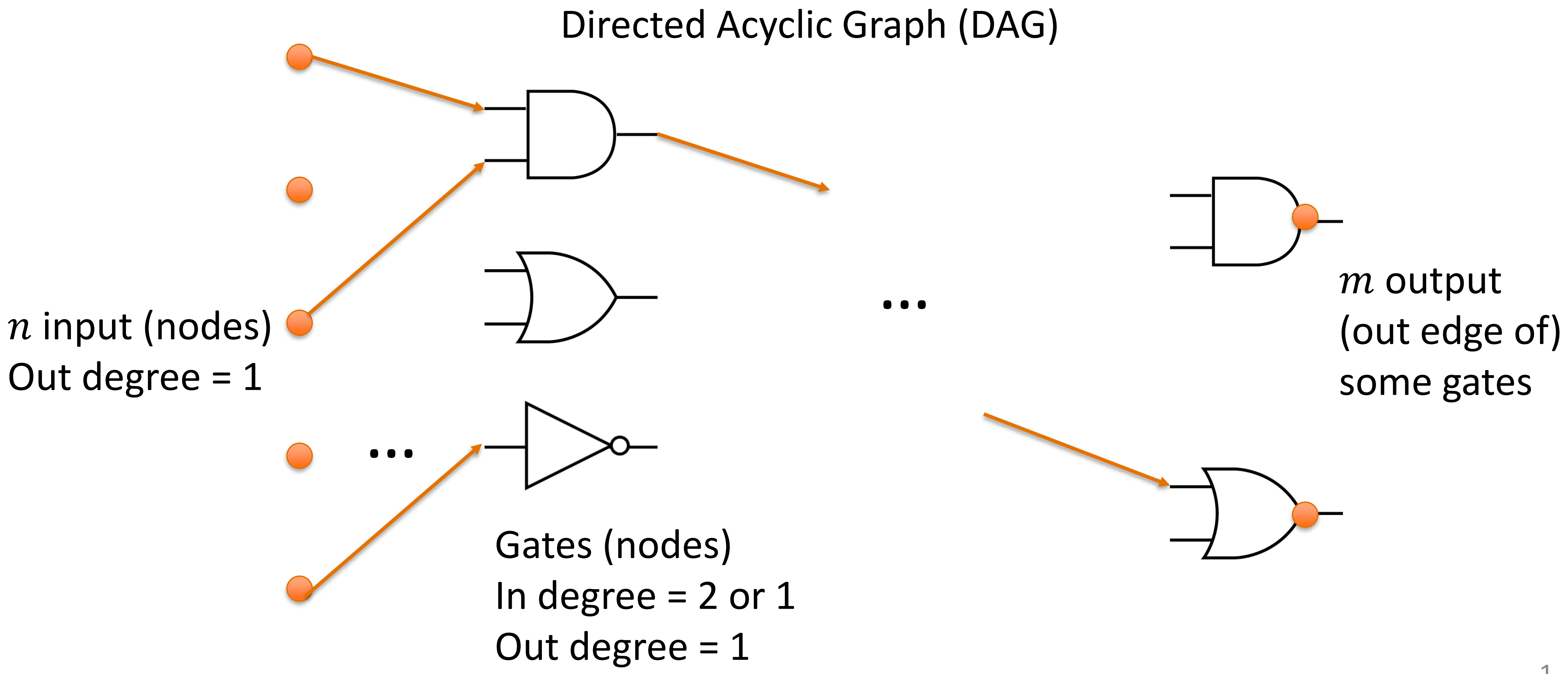


Problem Set 3 is due
This Friday, Feb 7 (10pm)

Class 7: *Universality and Impossibilities*

University of Virginia
cs3120: DMT2
Wei-Kai Lin

Recap: AND-OR-NOT Circuits



Recap: Universality (or, why care about AON)

Definition 3.20 (General straight-line programs)

Let $\mathcal{F} = \{f_0, \ldots, f_{t-1}\}$ be a finite collection of Boolean functions, such that $f_i : \{0,1\}^{k_i} \to \{0,1\}$ for some $k_i \in \mathbb{N}$. An $\mathcal{F}$ *program* is a sequence of lines, each of which assigns to some variable the result of applying some $f_i \in \mathcal{F}$ to k_i other variables. As above, we use `X[` i `]` and `Y[` j `]` to denote the input and output variables.

We say that $\mathcal{F}$ is a *universal set of operations* (also known as a universal gate set) if there exists a $\mathcal{F}$ program to compute the function $NAND$.

(Informal) Definition. We say a computation model is ***universal*** if for **any** finite function $f: \{0,1\}^n \to \{0,1\}^m$, there is an "instance" of the model that computes f.

Goal: Compute $f: \{0,1\}^n \to \{0,1\}^m$

- For any $n, m \geq 1$
- Compute 1-bit output for some $f_i: \{0,1\}^n \to \{0,1\}$, and repeat m times
- Represent f as a string s_f:

$$s_f = b_0 b_1 \dots b_{2^n - 1}, \qquad b_i = f(i)$$

- Want circuit: LOOKUP(s, i) outputs s[i]

Array: LOOKUP$_k$(s, i)

- k: 1,2,3,...
- s: 2^k-bit string, s = $b_0 b_1 \ldots b_{2^k-1}$
- i: k-bit string, representing $0, 1, \ldots, 2^k - 1$
- LOOKUP$_k$(s, i) outputs s[i] = b_i

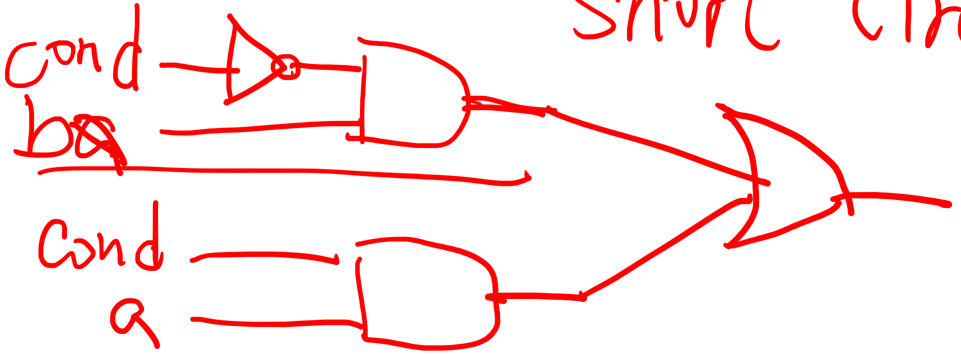
k determines len(s) and len(i)
No need comma as input

Tool: IF(cond, a, b)

ternary operater

- Output a if cond = 1
- Output b if cond = 0

short circuit



- AON circuit implements IF

cond	a	b	IF(cond, a, b)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Circuit: LOOKUP_1(s, i)

2bit 1-bit

k: 1,2,3,…

s = $b_0 b_1 \ldots b_{2^k-1}$

i represent $0, 1, \ldots, 2^k - 1$

outputs s[i] = b_i

- LOOKUP_1(s, i) = $\text{LOOKUP}_1(b_0 b_1$, i)

  return IF(i, b_1, b_0)

Circuit: $\text{LOOKUP}_2(\underbrace{s}_{2^2 = 4\text{bit}}, \underbrace{i}_{2\text{bit}})$

k: 1,2,3,…

s = $b_0 b_1 \ldots b_{2^k - 1}$

i represent $0, 1, \ldots, 2^k - 1$

outputs s[i] = b_i

- $\text{LOOKUP}_2(s, i) = \text{LOOKUP}_2(\underline{b_0 b_1 b_2 b_3}, \underbrace{i}_{})$

$$= \text{IF}\Big(i[0],\ \underline{\text{IF}(i[1], b_3, b_2)}^{\,2\text{bit}},\ \underline{\text{IF}(i[1], b_1, b_0)}\Big)$$

$$\text{LOOKUP}_1(b_2 b_3, i[1]),\ \text{LOOKUP}_1(b_0 b_1, i[1])$$

Circuit: $\text{LOOKUP}_k(s, i)$

k: 1,2,3,…

s = $b_0 b_1 \ldots b_{2^k-1}$

i represent $0, 1, \ldots, 2^k - 1$

outputs s[i] = b_i

- $\text{LOOKUP}_k(s, i) = \text{LOOKUP}_k(b_0 b_1 \ldots b_{2^k-1}, i)$
- Recurse!

$= \text{IF}(i[0], \text{LOOKUP}_{k-1}(b_{2^{k-1}} \ldots b_{2^k-1}, i[1 \ldots k-1]),$

$\text{LOOKUP}_{k-1}(b_0 \ldots b_{2^{k-1}-1}, i[1 \ldots k-1]))$

Circuit: LOOKUP_k(s, i)

k: 1,2,3,…

s = $b_0 b_1 \ldots b_{2^k-1}$

i represent $0, 1, \ldots, 2^k - 1$

outputs s[i] = b_i

LOOKUP_k(s, i):

first_half = $\text{LOOKUP}_{k\text{-}1}$(s[0:$2^{k-1}$], i[1:k])

second_half = $\text{LOOKUP}_{k\text{-}1}$(s[$2^{k-1}$:$2^k$], i[1:k])

return IF(i[0], second_half, first_half)

Theorem:
AON circuit is universal.

A simpler proof of universality of {AND,OR,NOT}

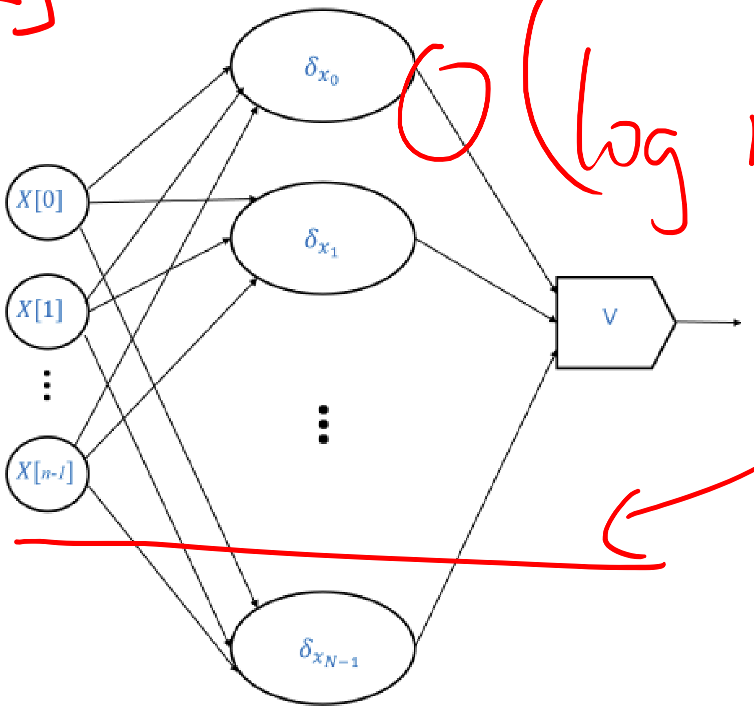
- For $\alpha \in \{0,1\}^n$, the delta function $\delta_\alpha(X) = 1$ iff $x = \alpha$
- Function $\delta_\alpha(X)$ can be computed by a circuit of size $O(n)$

AND(α[0]=X[0], α[1]=X[1]....)

O(log n)

- To implement function f,
  we use the following approach:

$$f(X) = \bigvee_{\alpha \in \{0,1\}^n,\, f(\alpha)=1}$$



Why do we want **Universal** Models?

(Informal) Definition. We say a computation model is ***universal*** if for **any** finite function $f: \{0,1\}^n \to \{0,1\}^m$, there is an "instance" of the model that computes f.

Definition: We say that a gate set $\mathcal{G}$ is a *universal set of operations* (also known as a universal gate set) if there exists a $\mathcal{G}$ program to compute the function NAND.

(Textbook, Definition 3.20)

$\mathcal{G}$ comp NAND $\Leftrightarrow$ $\mathcal{G}$ comp AON

$\Leftrightarrow$ $\mathcal{G}$ comp LOOKUP_k

$\Leftrightarrow$ $\mathcal{G}$ comp any finite func

Proving Universality

A gate set $\mathcal{G}$ is a *universal set of operations* (also known as a universal gate set) if there exists a $\mathcal{G}$ program to compute the function NAND.

How can we prove some gate set G is universal?

show $\mathcal{G}$-circuit comp NAND.

- {CNOT}

Proving **Non**-Universality

A gate set $\mathcal{G}$ is a *universal set of operations* (also known as a universal gate set) if there exists a $\mathcal{G}$ program to compute the function NAND.

*How can we prove some gate set G is **not** universal?*

{OR}

Exists f s.t. $\mathcal{G}$ ~~cannot~~ cannot comp.

$\mathcal{G}$ comp ~~any~~ every finite func

Non-Universality of $\{OR\}$

A gate set $\mathcal{G}$ is a *universal set of operations* (also known as a universal gate set) if there exists a $\mathcal{G}$ program to compute the function NAND.

OR comp NOT?

$$OR: \{0,1\}^2 \to \{0,1\}$$

$$OR(a,b) = \begin{cases} 0, & a = b = 0 \\ 1, & otherwise \end{cases}$$

$NOT(x)$

$OR(x, x) = x$ 1 bit

$OR(x, y)$ monotonic

Def. $f(x_1, x_2 \cdots)$ is monotonic

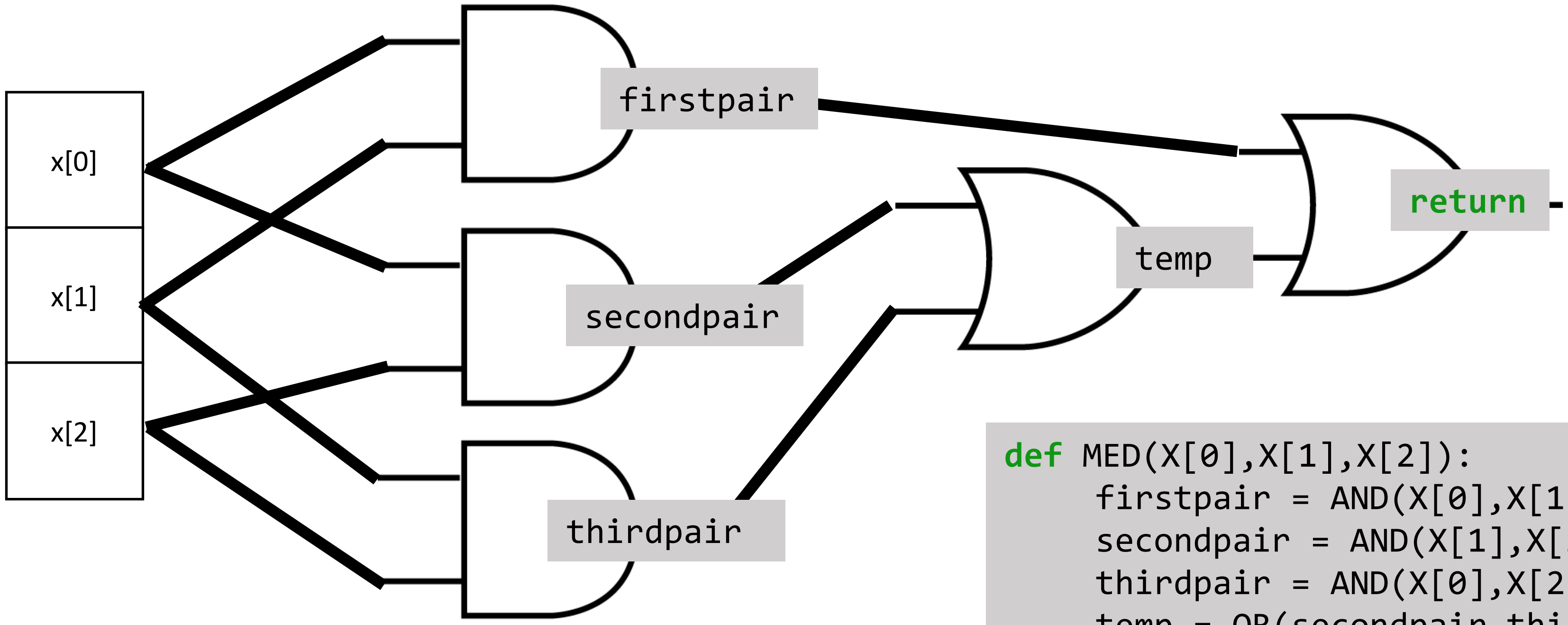
if $\forall i$, $f(x_1 \cdots x_i \cdots x_n) \le f(x_1 \cdots 1 \cdots x_n)$

Circuit Complexity

Complexity: What's the cost?

Sure, we can build the circuit!

But how many *gates* are needed?




```
def MED(X[0],X[1],X[2]):
    firstpair = AND(X[0],X[1])
    secondpair = AND(X[1],X[2])
    thirdpair = AND(X[0],X[2])
    temp = OR(secondpair,thirdpair)
    return OR(firstpair,temp)
```

Recap: The big O and friends $\Omega, \Theta, o, \omega$

- $O\big(g(n)\big)$ is a set of functions:
  By $f(n) = O\big(g(n)\big)$, we mean $f(n) \in O\big(g(n)\big)$
- Similarly for $\Omega, \Theta, o, \omega$

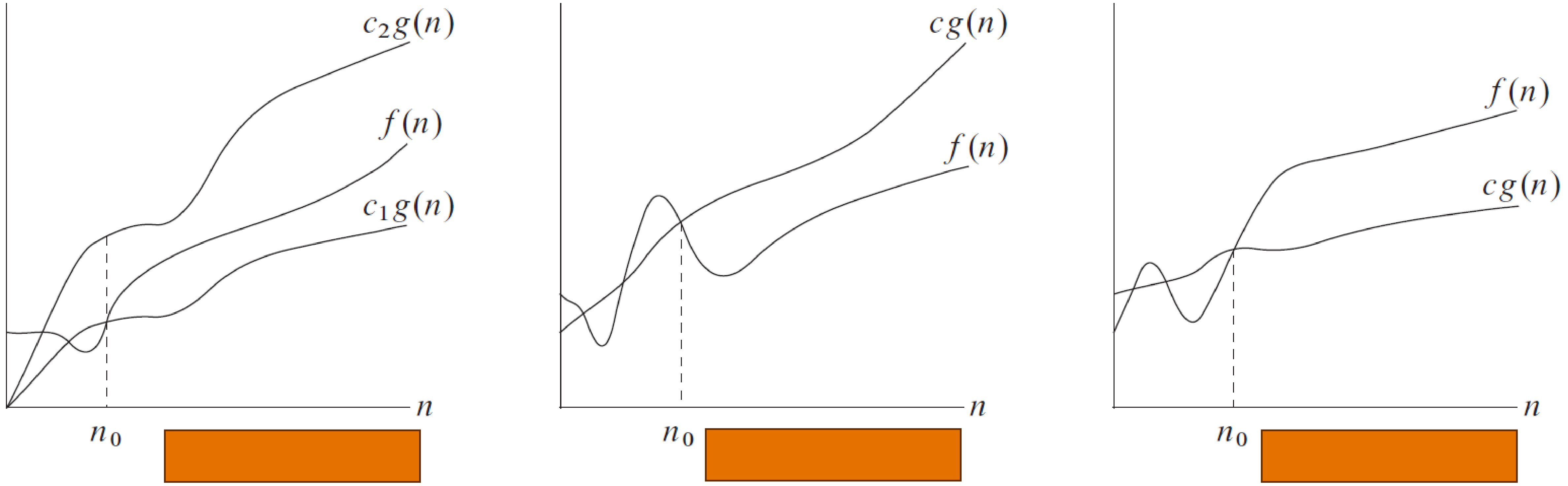
- $f(n) = O\big(g(n)\big)$
  There exist constants c and n_0 such that for all $n \geq n_0$,
  $0 \leq f(n) \leq cg(n)$

- $f(n) = O\big(g(n)\big)$:
  There exist constants c and n_0 such that for all $n \geq n_0$, $0 \leq f(n) \leq cg(n)$

- $f(n) = \Omega\big(g(n)\big)$ is equivalent to $g(n) = O(f(n))$

- $f(n) = \Theta\big(g(n)\big)$ is equivalent to $f = O(g)$
  $g = O(f)$

Quick Quiz



(a) (b) (c)

$f = \Theta(g)$ $f = O(g)$ $f = \Omega(g)$

- $f(n) = o\big(g(n)\big)$:
  For any c there exists $n_0 > 0$ such that for all $n \geq n_0$,
  $0 \leq f(n) < cg(n)$

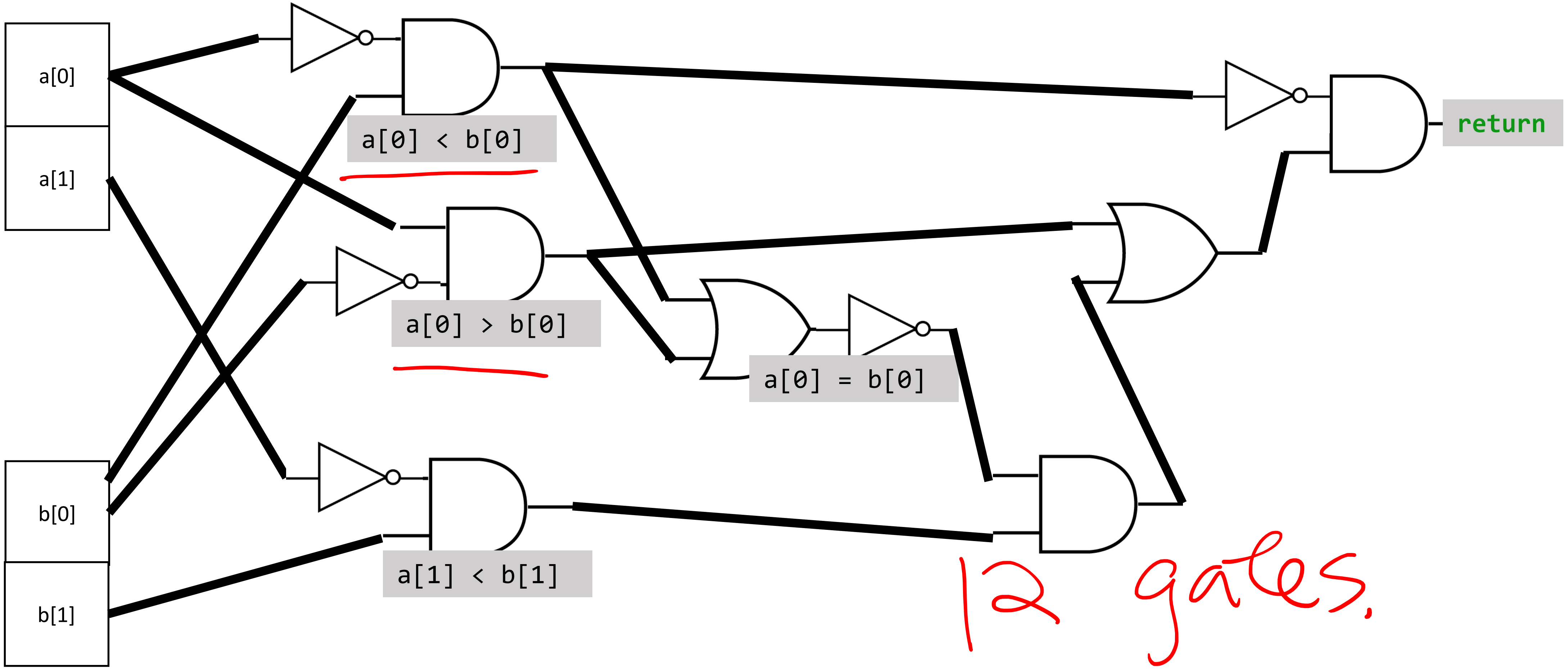
- $f(n) = \omega\big(g(n)\big)$ is equivalent to $g(n) = o\big(f(n)\big)$

Quick Quiz

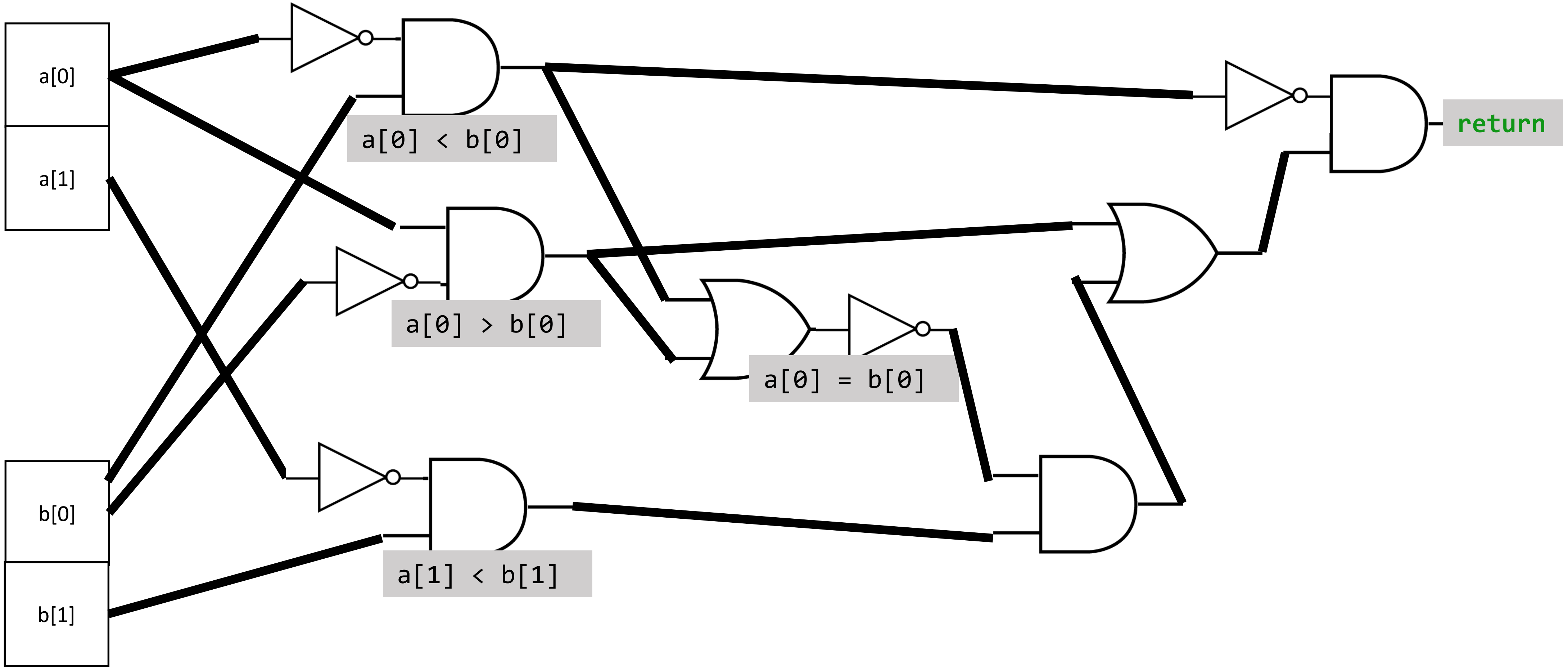
Suppose $f(n) = o\big(g(n)\big)$. Which must hold? Can hold?

1. $g(n) = O(f(n))$ N N
2. $g(n) = \Omega\big(f(n)\big)$ Y Y
3. $g(n) = \Theta(f(n))$ N N
4. $g(n) = o(f(n))$ N N
5. $g(n) = \omega(f(n))$ Y Y

How many gates for 2-bit Compare?
If a < b, output 0; o.w., output 1

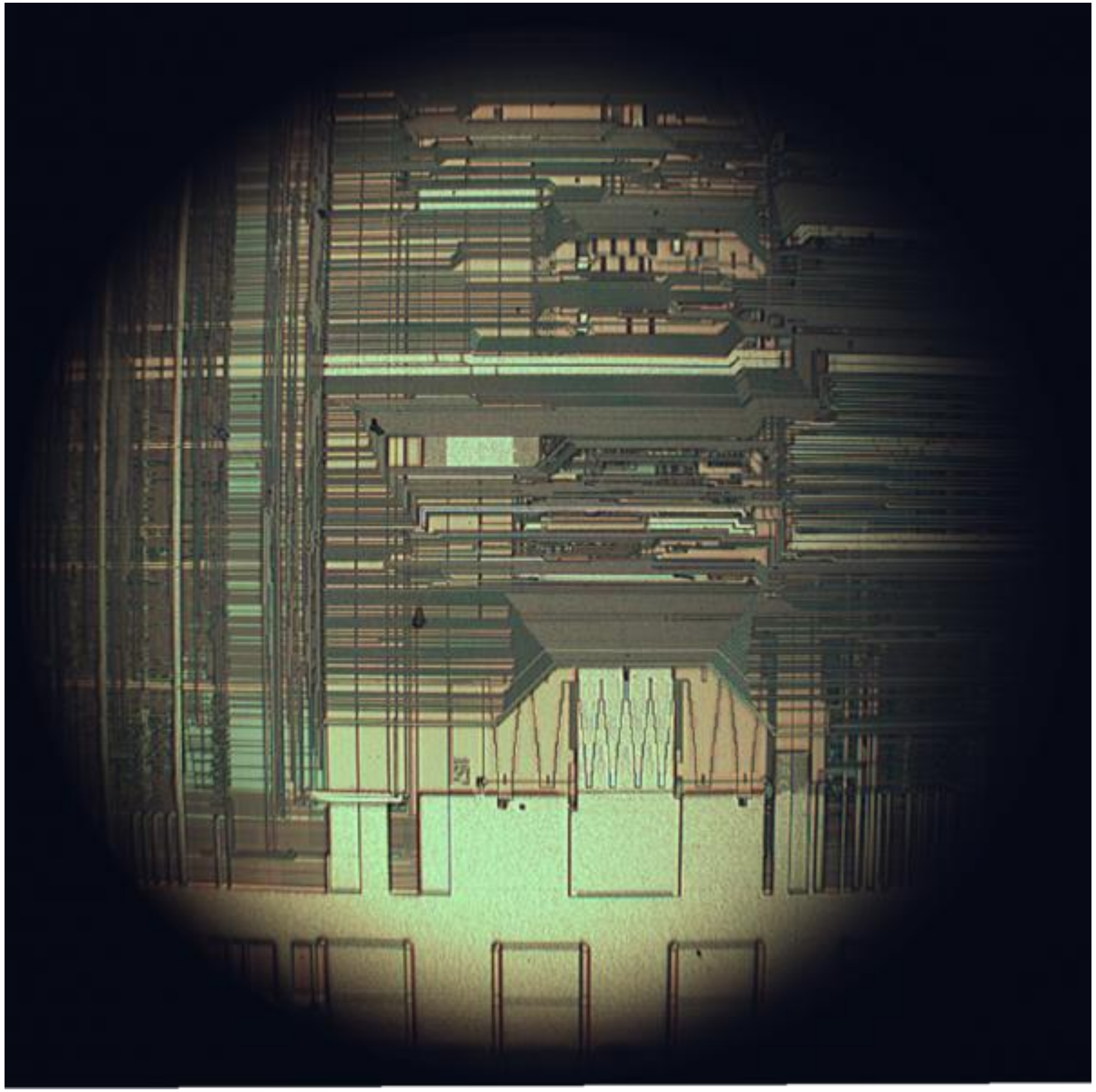


Can we do better = less gates?



Still the main goal!

"The Intel **486**, officially named **i486** and also known as **80486**, is a microprocessor introduced in 1989."
Wikipedia: i486



https://en.wikipedia.org/wiki/Integrated_circuit

How many gates?

k: 1,2,3,...

s = $b_0 b_1 \ldots b_{2^k-1}$

i represent $0, 1, \ldots, 2^k - 1$

outputs s[i] = b_i

S(k)

LOOKUP$_k$(s, i):

    first_half = LOOKUP$_{k-1}$(s[0:2^{k-1}], i[1:k])
    second_half = LOOKUP$_{k-1}$(s[2^{k-1}:2^k], i[1:k])
    return IF(i[0], second_half, first_half)

$S(k) = 2S(k-1) + c$

$\Rightarrow S(k) = c' \cdot 2^k = O(2^k)$

How much time did you spend on PRR3?

- ~120 response

- 47: 15-30 minutes
- 68: longer than 30 minutes

- 1 did not answer

Charge

Universality

Any finite function

Circuit Size

Number of gates

PS3: due this Friday 10:00pm