Computer Architecture Background

https://web.stanford.edu/class/cs114/

Philip Levis
Stanford University
pal@cs.stanford.edu

Background for Plasticine

Lots of material in this lecture borrowed from
Subhasish Mitra, Bill Dally, and John Hennessy

Computer Systems

Computer systems focus on *abstraction*

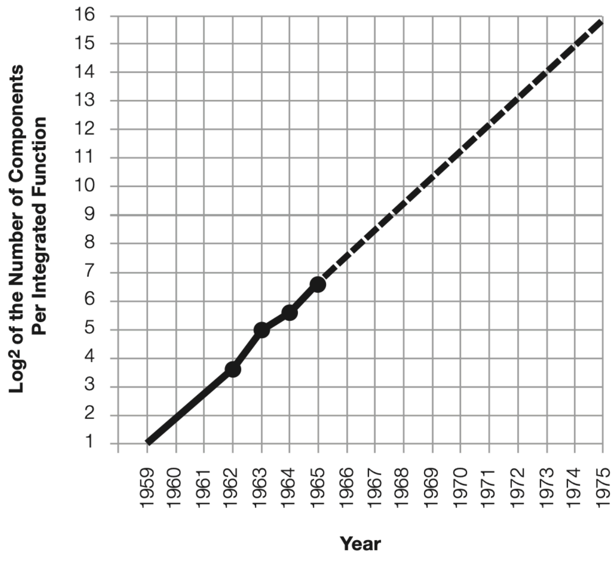
- Software (operating systems): I have some hardware resources, what software API do I provide?
  - File systems: disk blocks become files
  - Spark: network of computers runs thousands of small tasks
- Hardware (architecture): I have digital logic, what mechanisms do I provide to software?
  - An instruction set defines how arithmetic and memory work
  - A bus defines how hardware devices can access each other

A good abstraction is easy to use and efficient while being simple to implement

Architecture

Architecture driven by Moore's Law

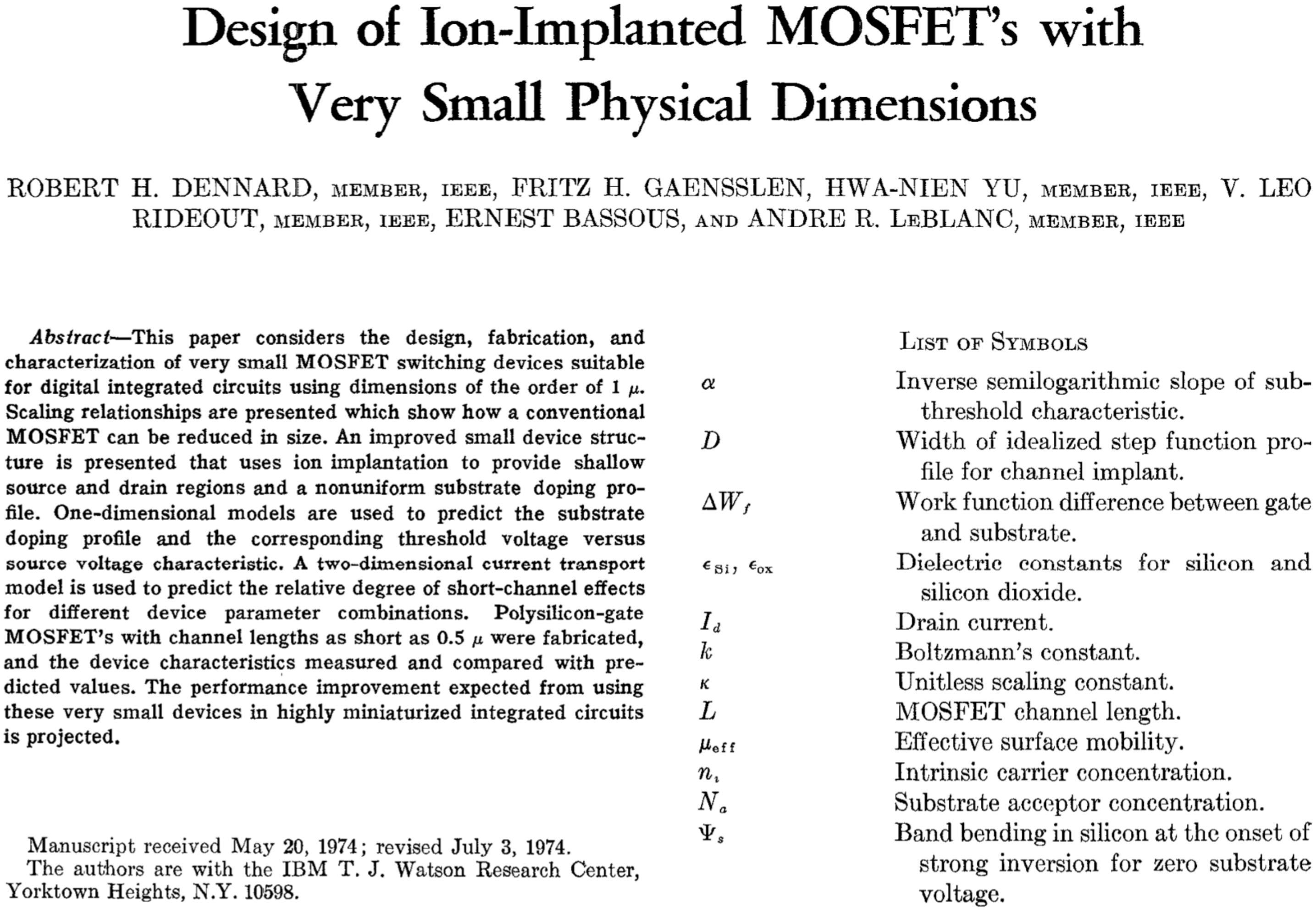
- "The complexity for minimum component costs has increased at a rate of roughly a factor of two per year"



As transistors shrink, many good things happen: Dennard scaling

- Voltage does down
- Delay goes down
- Clock speed goes up
- Power consumption goes down
- Started to break down around 2006

Dennard Scaling



Design of Ion-Implanted MOSFET's with Very Small Physical Dimensions

ROBERT H. DENNARD, MEMBER, IEEE, FRITZ H. GAENSSLEN, HWA-NIEN YU, MEMBER, IEEE, V. LEO RIDEOUT, MEMBER, IEEE, ERNEST BASSOUS, AND ANDRE R. LeBLANC, MEMBER, IEEE

***Abstract*—This paper considers the design, fabrication, and characterization of very small MOSFET switching devices suitable for digital integrated circuits using dimensions of the order of 1 μ. Scaling relationships are presented which show how a conventional MOSFET can be reduced in size. An improved small device structure is presented that uses ion implantation to provide shallow source and drain regions and a nonuniform substrate doping profile. One-dimensional models are used to predict the substrate doping profile and the corresponding threshold voltage versus source voltage characteristic. A two-dimensional current transport model is used to predict the relative degree of short-channel effects for different device parameter combinations. Polysilicon-gate MOSFET's with channel lengths as short as 0.5 μ were fabricated, and the device characteristics measured and compared with predicted values. The performance improvement expected from using these very small devices in highly miniaturized integrated circuits is projected.**

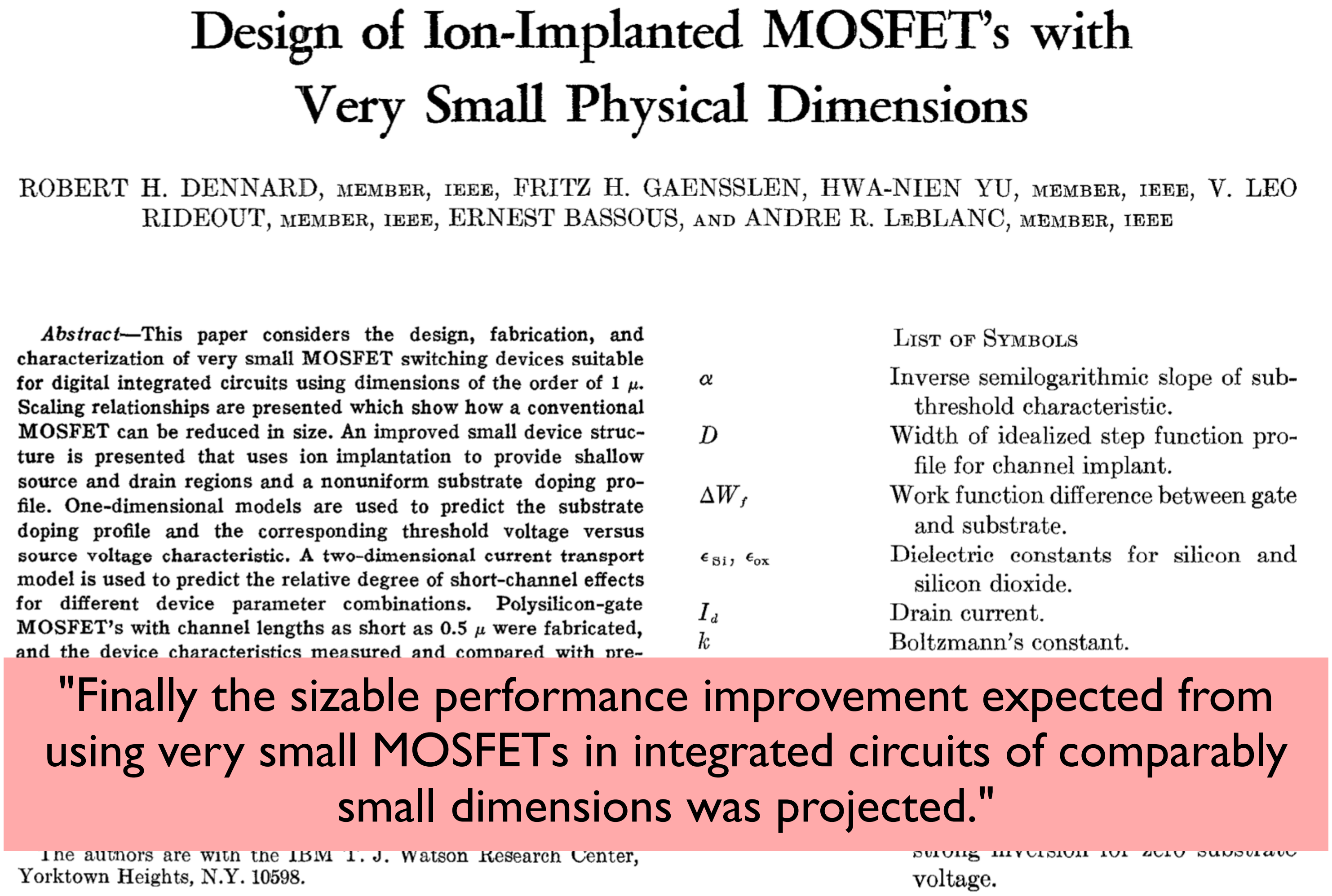
Manuscript received May 20, 1974; revised July 3, 1974.

The authors are with the IBM T. J. Watson Research Center, Yorktown Heights, N.Y. 10598.

List of Symbols

α	Inverse semilogarithmic slope of subthreshold characteristic.
D	Width of idealized step function profile for channel implant.
ΔW_f	Work function difference between gate and substrate.
ϵ_{Si}, ϵ_{ox}	Dielectric constants for silicon and silicon dioxide.
I_d	Drain current.
k	Boltzmann's constant.
κ	Unitless scaling constant.
L	MOSFET channel length.
μ_{eff}	Effective surface mobility.
n_i	Intrinsic carrier concentration.
N_a	Substrate acceptor concentration.
Ψ_s	Band bending in silicon at the onset of strong inversion for zero substrate voltage.

Dennard Scaling



Design of Ion-Implanted MOSFET's with Very Small Physical Dimensions

ROBERT H. DENNARD, MEMBER, IEEE, FRITZ H. GAENSSLEN, HWA-NIEN YU, MEMBER, IEEE, V. LEO RIDEOUT, MEMBER, IEEE, ERNEST BASSOUS, AND ANDRE R. LEBLANC, MEMBER, IEEE

Abstract—This paper considers the design, fabrication, and characterization of very small MOSFET switching devices suitable for digital integrated circuits using dimensions of the order of 1 μ. Scaling relationships are presented which show how a conventional MOSFET can be reduced in size. An improved small device structure is presented that uses ion implantation to provide shallow source and drain regions and a nonuniform substrate doping profile. One-dimensional models are used to predict the substrate doping profile and the corresponding threshold voltage versus source voltage characteristic. A two-dimensional current transport model is used to predict the relative degree of short-channel effects for different device parameter combinations. Polysilicon-gate MOSFET's with channel lengths as short as 0.5 μ were fabricated, and the device characteristics measured and compared with pre-

The authors are with the IBM T. J. Watson Research Center, Yorktown Heights, N.Y. 10598.

LIST OF SYMBOLS

α	Inverse semilogarithmic slope of subthreshold characteristic.
D	Width of idealized step function profile for channel implant.
ΔW_f	Work function difference between gate and substrate.
ϵ_{Si}, ϵ_{ox}	Dielectric constants for silicon and silicon dioxide.
I_d	Drain current.
k	Boltzmann's constant.
	strong inversion for zero substrate voltage.

"Finally the sizable performance improvement expected from using very small MOSFETs in integrated circuits of comparably small dimensions was projected."

Historical Impact of Moore's Law

1960-1985: Cheaper, smaller chips

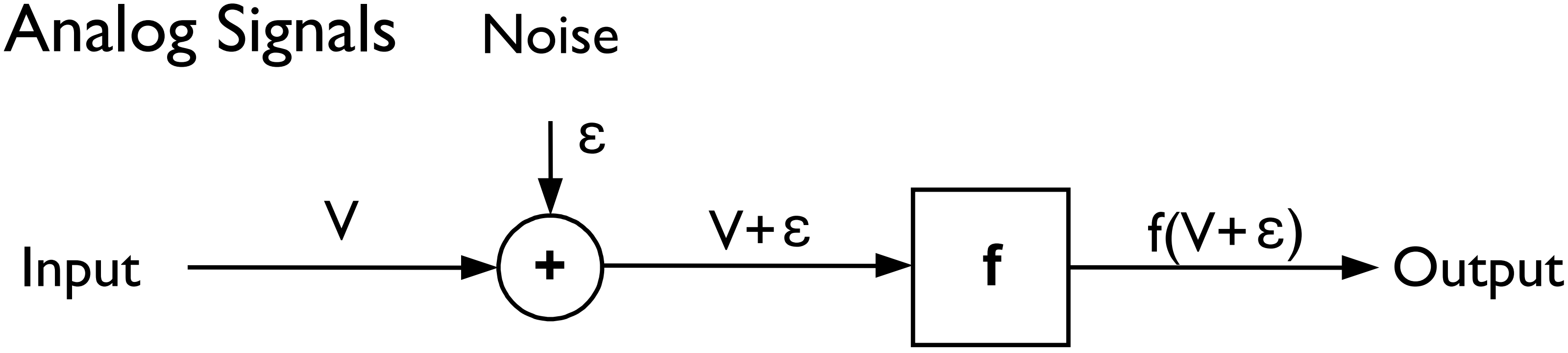
1982-2006: Very Large System Integration (VLSI) and EDA (electronic design automation) takes off; the "golden age" of computer architecture. Exponential growth in computing performance.

2006-Today: Performance mostly plateaus: move to multi-core, parallel architectures (GPUs, TPUs) and accelerators (e.g., AES instruction)

Digital Signals

Effect of Noise on Analog & Digital Signals

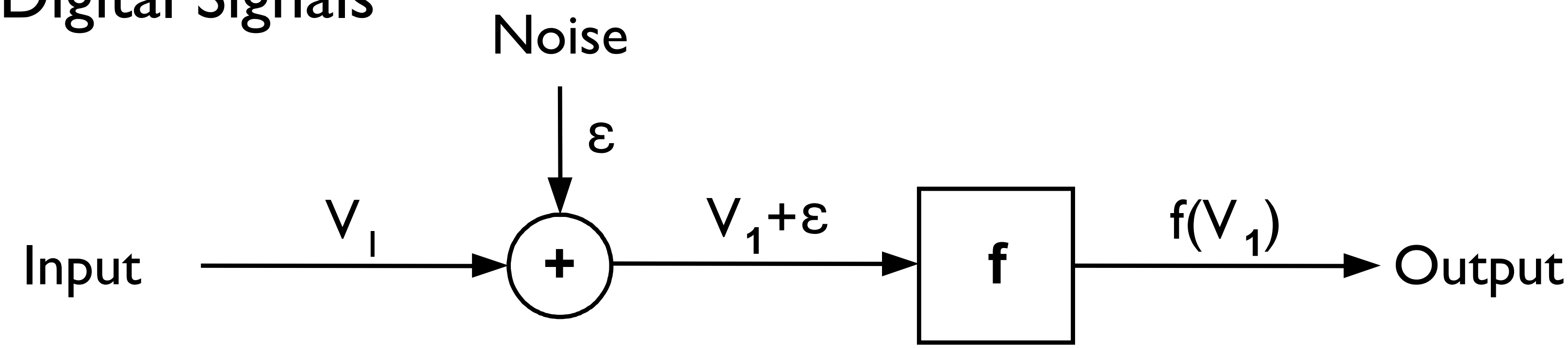
Analog Signals



Noise added to Input

Error at Output

Digital Signals



Noise added to Input < Noise Margin

Correct Value at Output

The Power of the Transfer Function

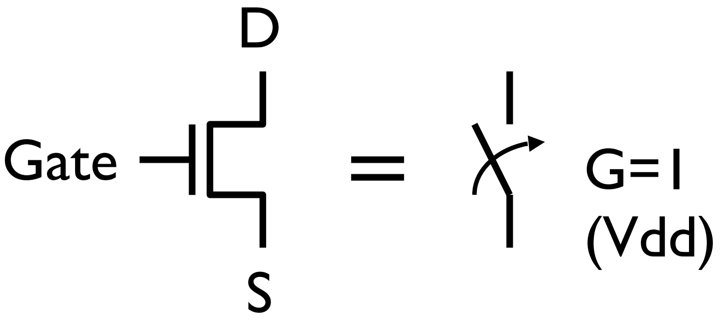
Each transistor restores the signal.

This allows us to build tremendously deep and complex digital circuits out of many transistors.

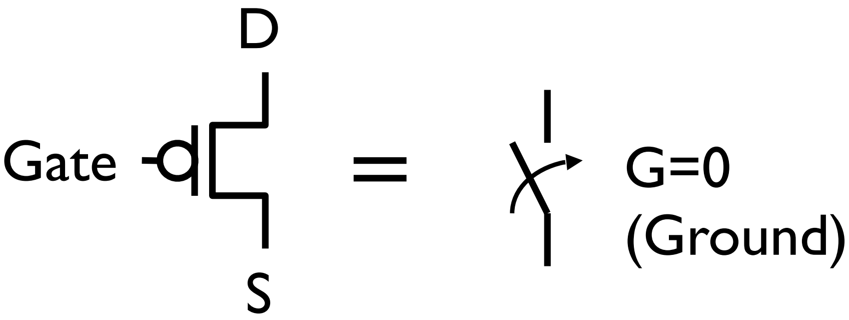
This is in contrast to analog circuits, which need to carefully manage how noise is introduced and propagated (the "black art" of analog design).

Transistors Make Gates

Simplest Model for MOS transistors



NMOS transistors



PMOS transistors

Let value 1 be Vdd (e.g., 1.8V, 2.5V), value 0 be ground

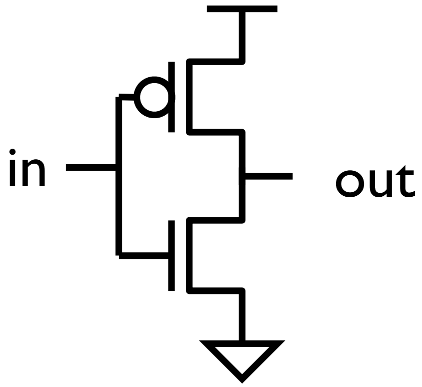
NMOS devices are switches

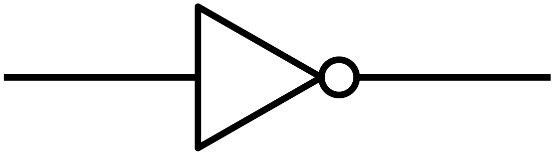
- Gate is 1 -> the drain D and source S **are** connected
- Gate is 0 -> the drain D and source S **are not** connected

PMOS devices are switches

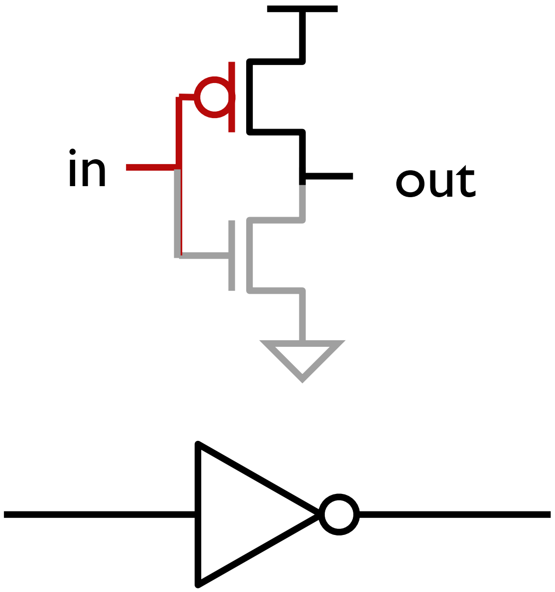
- Gate is 0 -> the drain D and source S **are** connected
- Gate is 1 -> the drain D and source S **are not** connected

Simple Case: Inverter





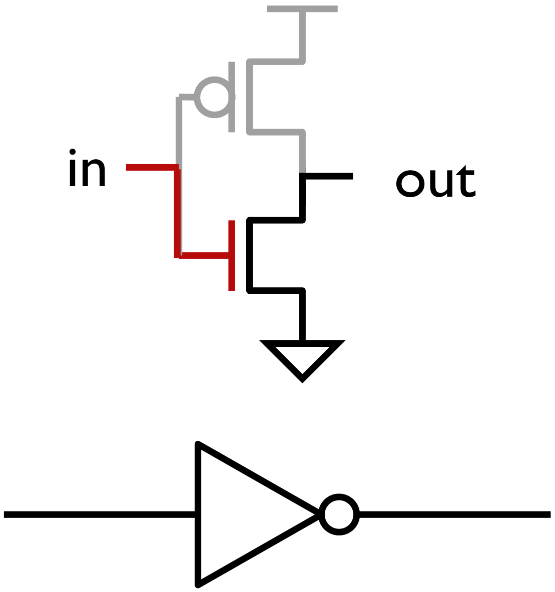
Simple Case: Inverter



If the input signal is 0, PMOS connects output to Vdd (1).

If the input signal is 1, NMOS connects output to ground (0).

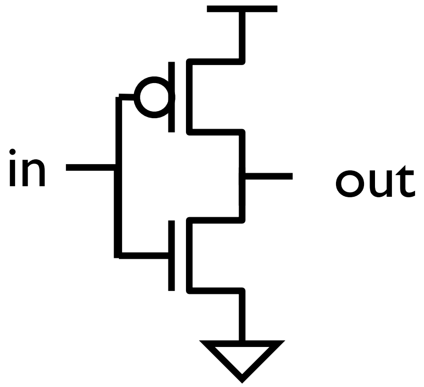
Simple Case: Inverter

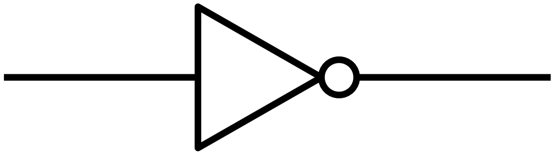


If the input signal is 0, PMOS connects output to Vdd (1).

If the input signal is 1, NMOS connects output to ground (0).

Simple Case: Inverter





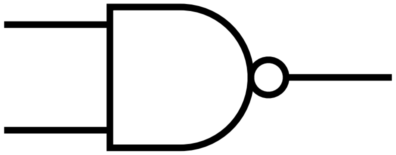
If the input signal is 0, PMOS connects output to Vdd (1).

If the input signal is 1, NMOS connects output to ground (0).

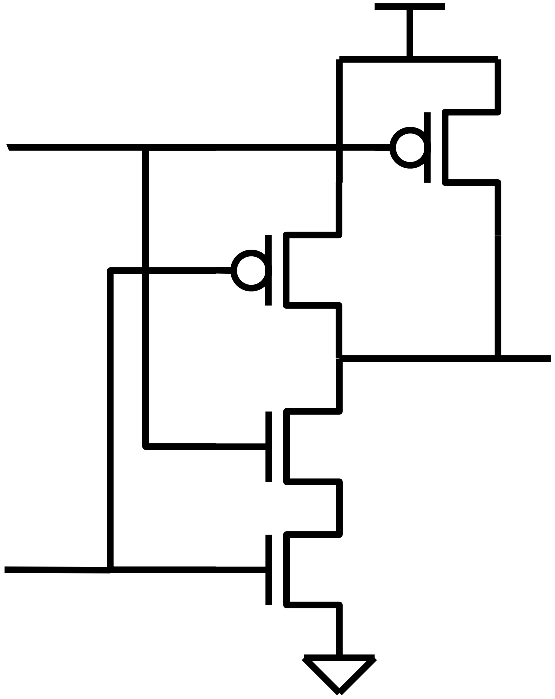
Meets the rules

- Output is always driven (Either pMOS or nMOS is always active)
- Vdd and Gnd are never shorted (directly connected)
    - At least with valid inputs

NAND Gate (universal!)



A	**B**	**Out**
0	0	1
0	1	1
1	0	1
1	1	0



nMOS in *series* pulling to Gnd

- If *both* A and B are 1, connect out to ground

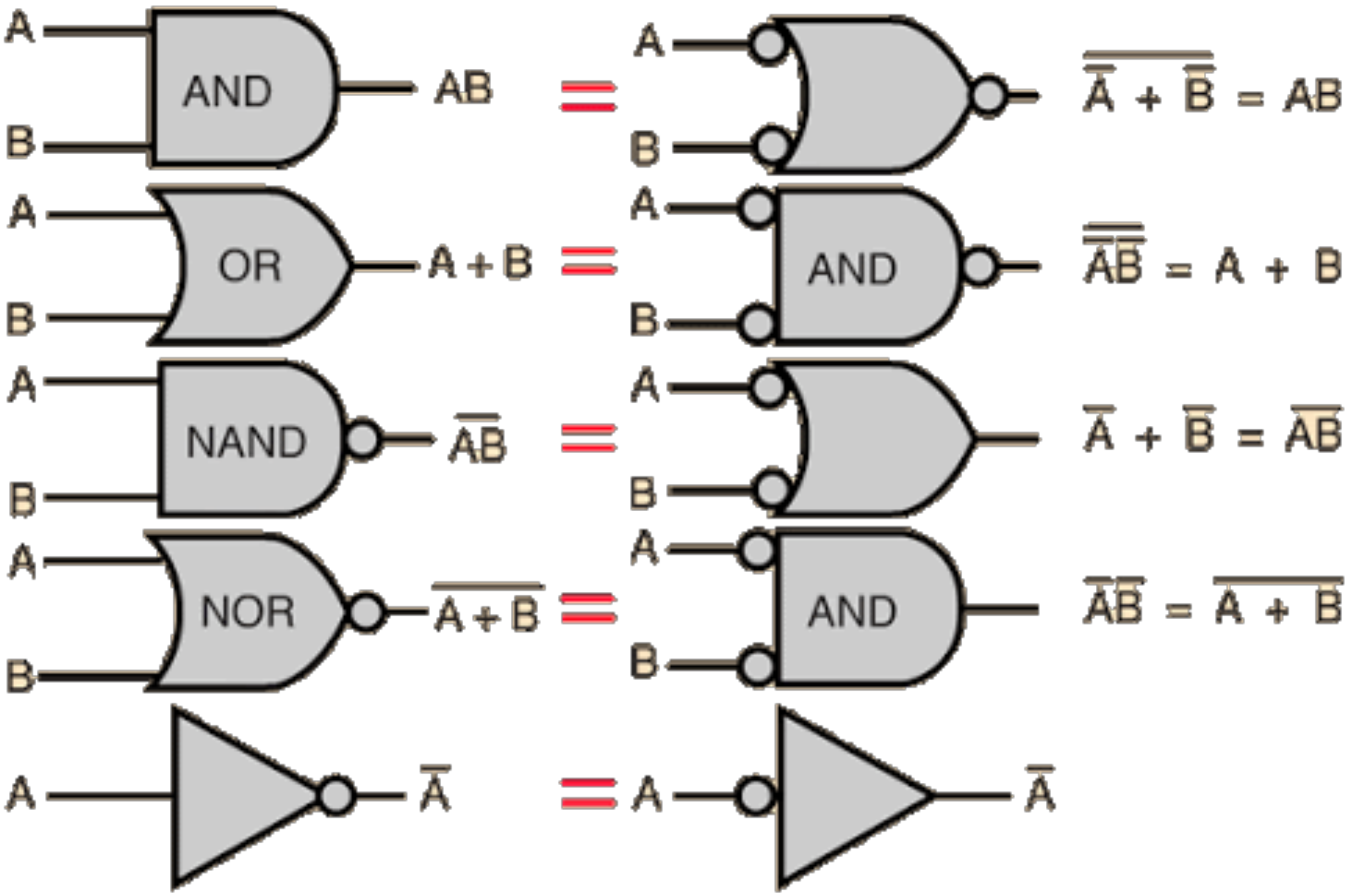
pMOS in *parallel* pulling to Vdd

- If *either* A or B is 0, connect out to Vdd

Out always driven by pMOS's or nMOS's but never both

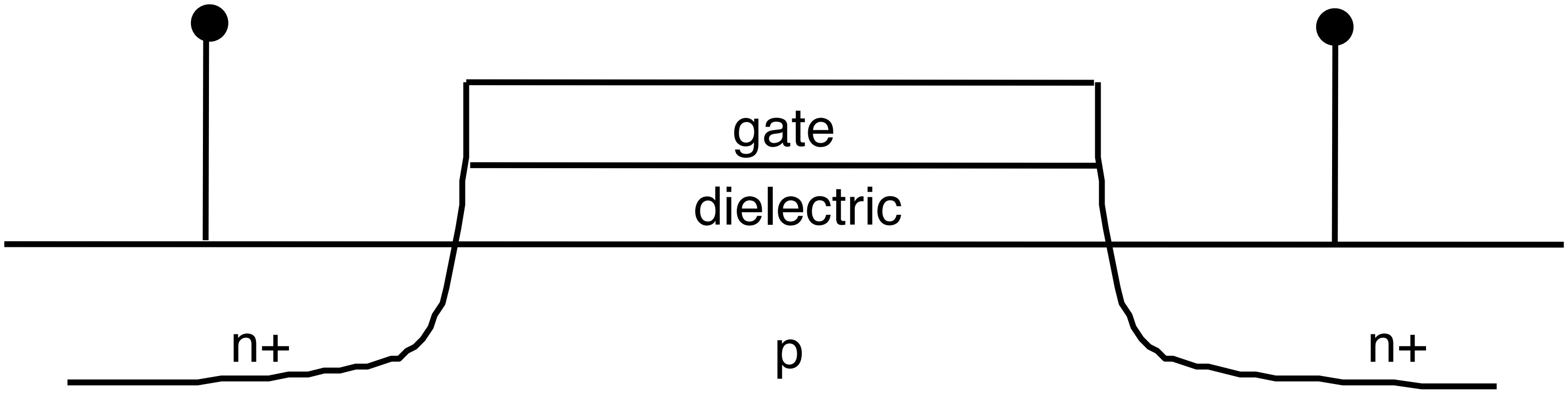
Number of inputs ("fan-in") can be increased

Gates


A
B
AND
AB
=
A
B
$\overline{\overline{A} + \overline{B}} = AB$
A
B
OR
A + B
=
A
B
AND
$\overline{\overline{A}\,\overline{B}} = A + B$
A
B
NAND
$\overline{AB}$
=
A
B
$\overline{A} + \overline{B} = \overline{AB}$
A
B
NOR
$\overline{A + B}$
=
A
B
AND
$\overline{A}\,\overline{B} = \overline{A + B}$
A
$\overline{A}$
=
A
$\overline{A}$

Transistor Performance

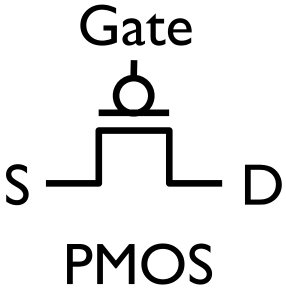
Transistors are Physical Devices

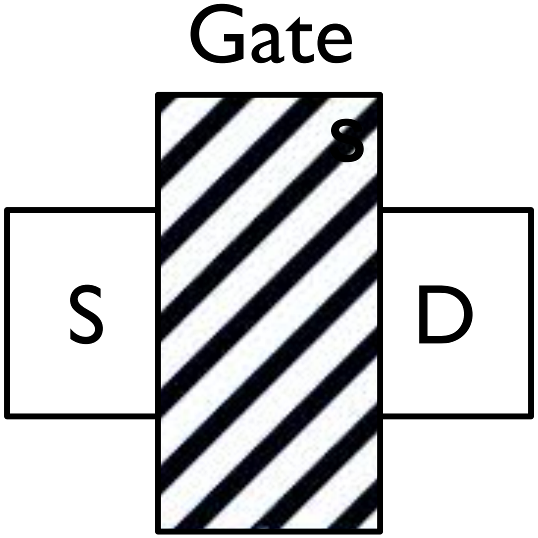


Gates used to be polysilicon (high purity poly-crystalline Si), now metal gates with high-k dielectric (like HfO_2)

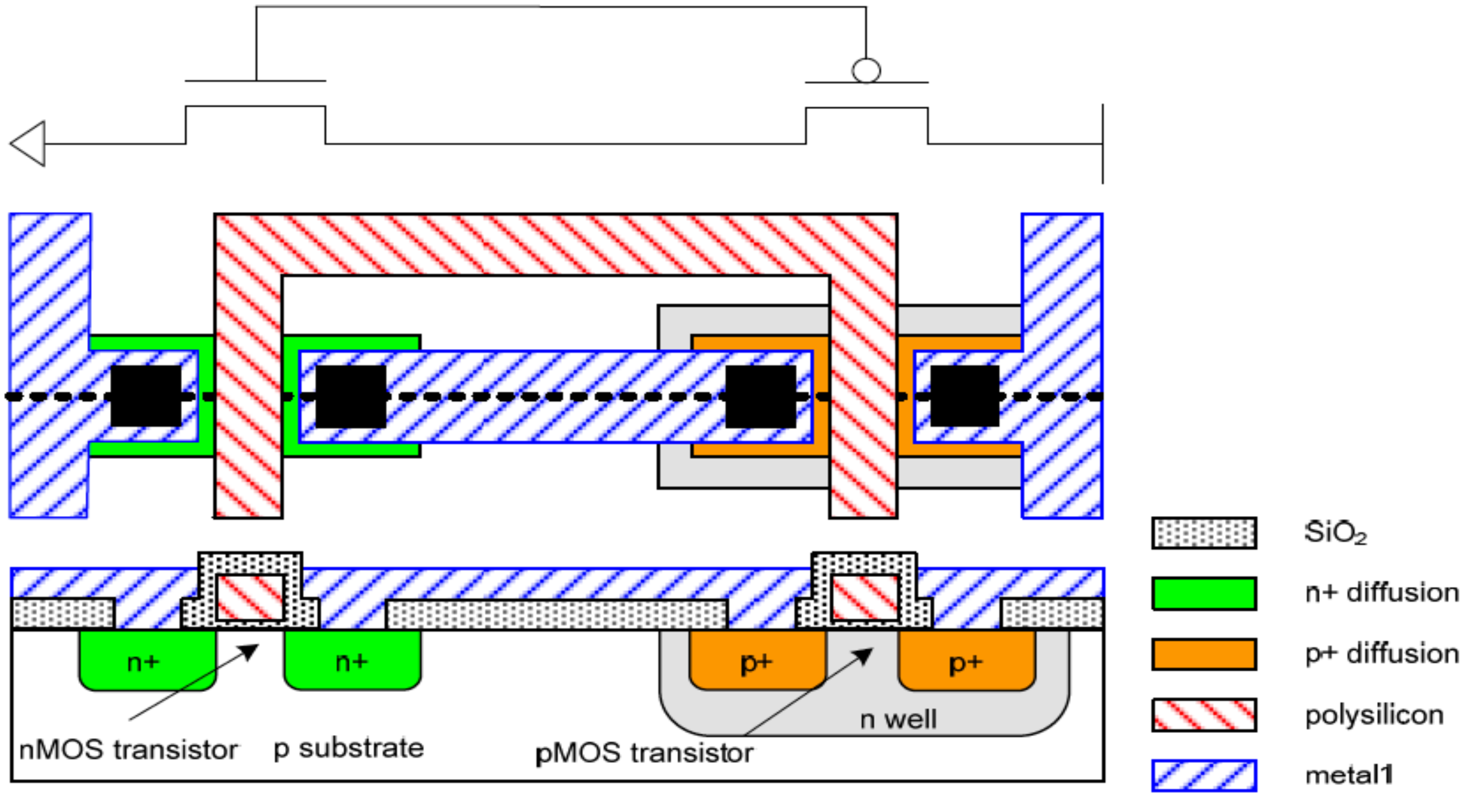
Advanced transistors are 3D (e.g., FinFETs) we'll just talk about planar for simplicity.

Transistor Geometry





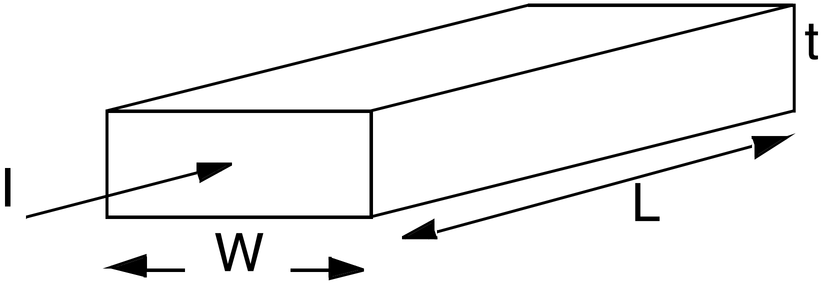
Example Layout of an Inverter



Resistance of a Conductor

Resistance of a conductor

- Resistivity ρ * Length/Area
- Designer does not control ρ, t
- Generally deal with ρ /t
- *Can* control W, L



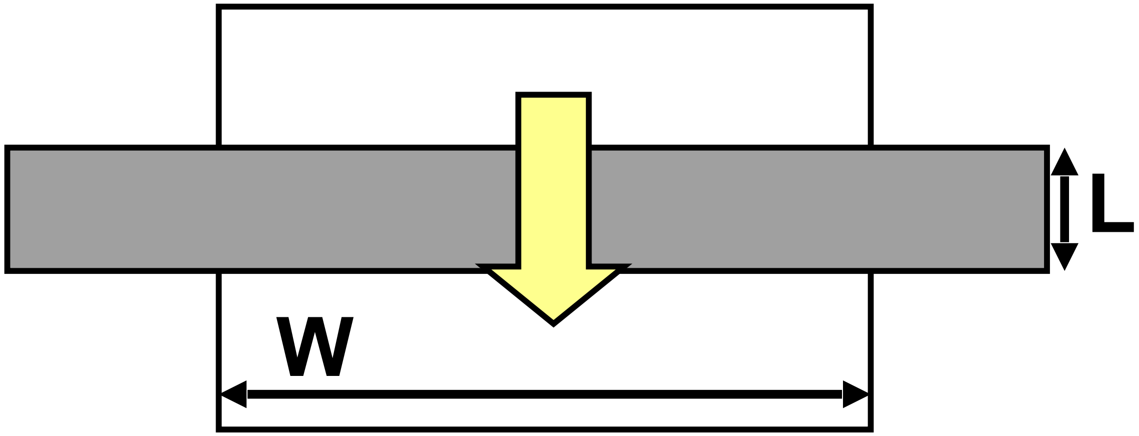
$$R = \frac{\rho L}{tW} = \frac{\rho}{t}\frac{L}{W}$$

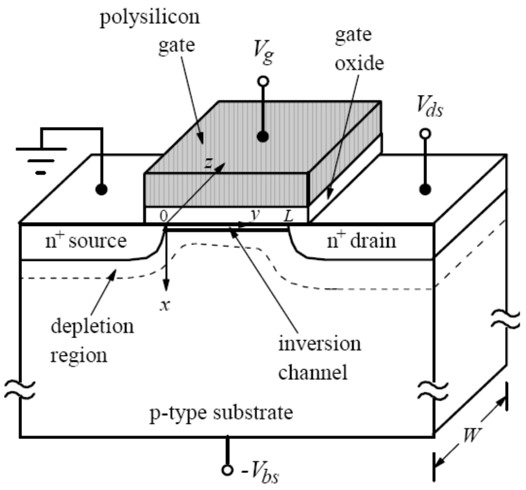
That's why LONG, NARROW (and THIN) wires have higher resistances

Resistance

For transistors

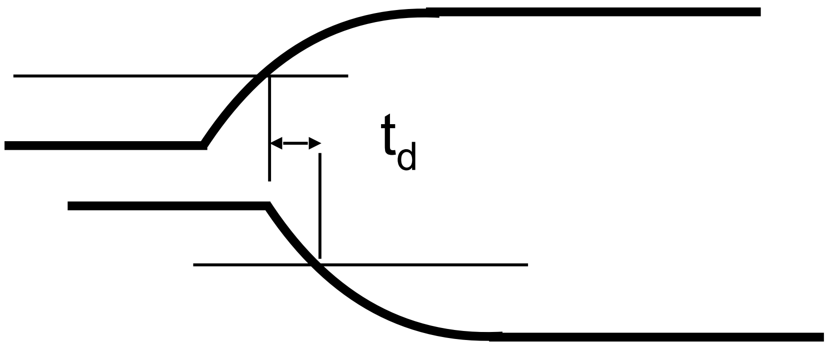
- Designer chooses
  - W and L
- Wider transistor
  - Lower resistance
  - More current
  - *Longer to switch*





Circuit Timing

Delay and Rise/Fall Time



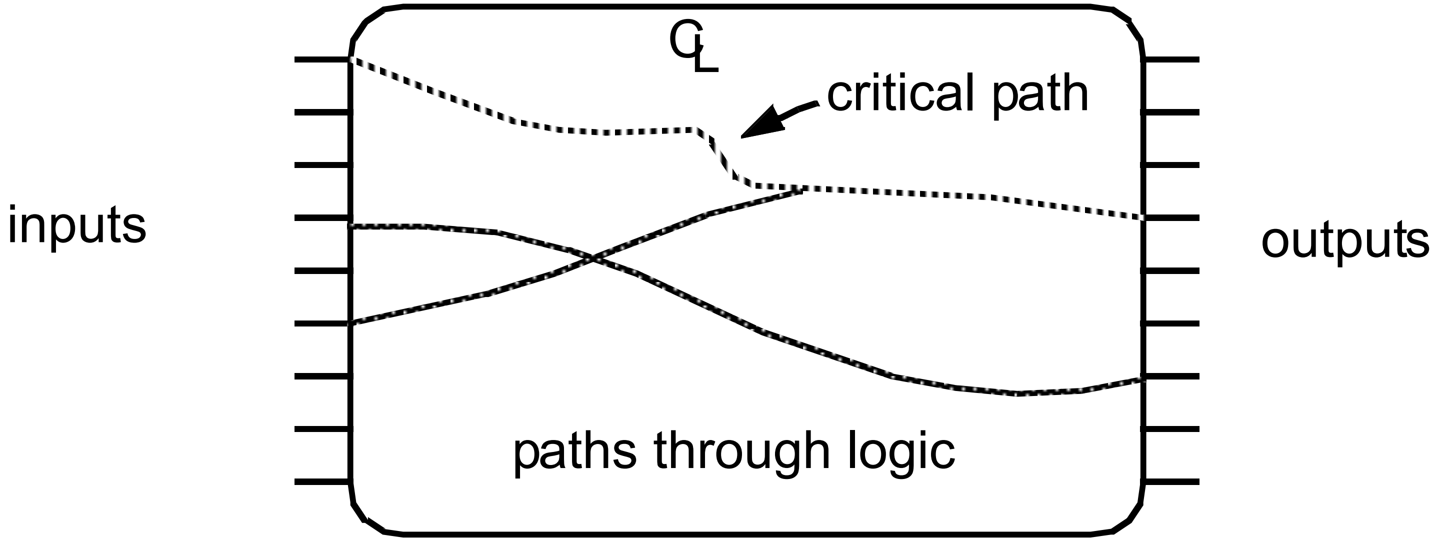
Delay, t_d, is measured from 50% point to 50% point.

Curve depends on resistance of circuit *to* transistor gate and capacitance *of* transistor gate.

- Lower resistance lowers t_d
- Smaller capacitance lowers t_d
- Tradeoff: smaller capacitance has higher resistance for next stage

Critical Paths

There are many signal paths through logic

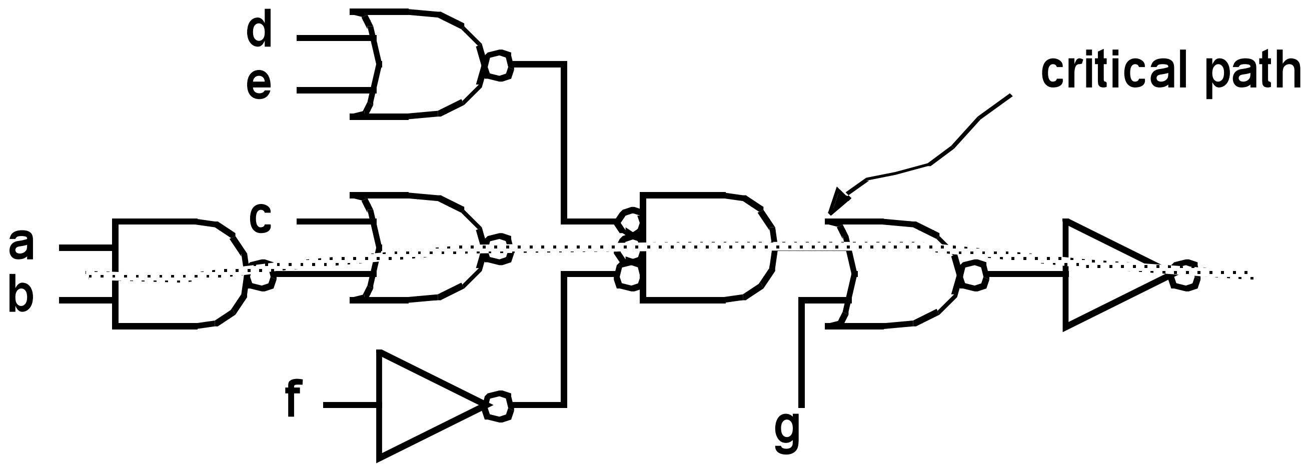


Not all the paths have the same delay

- Path from input to latest output is called the critical path
- It is this path that specifies the maximum rate the circuit can generate results (inverse of path delay)

Critical Path Example

This circuit computes the function $(\neg(a \wedge b) \vee c) \wedge (d \vee e) \wedge f) \vee g$



The *critical path* traverses a 2-NAND, 2 2-NORs, a 3-NAND, and an inverter. The delay of this path determines the maximum delay of a change in the inputs, e.g, $ab\bar{c}def\bar{g}$ becomes $\bar{a}b\bar{c}def\bar{g}$.

The length of wires also matters. Circuits in far away parts of chips take longer to communicate.

Clock Speed

Sequential Logic

So far, we have only looked at combinational circuits: input and outputs, no state

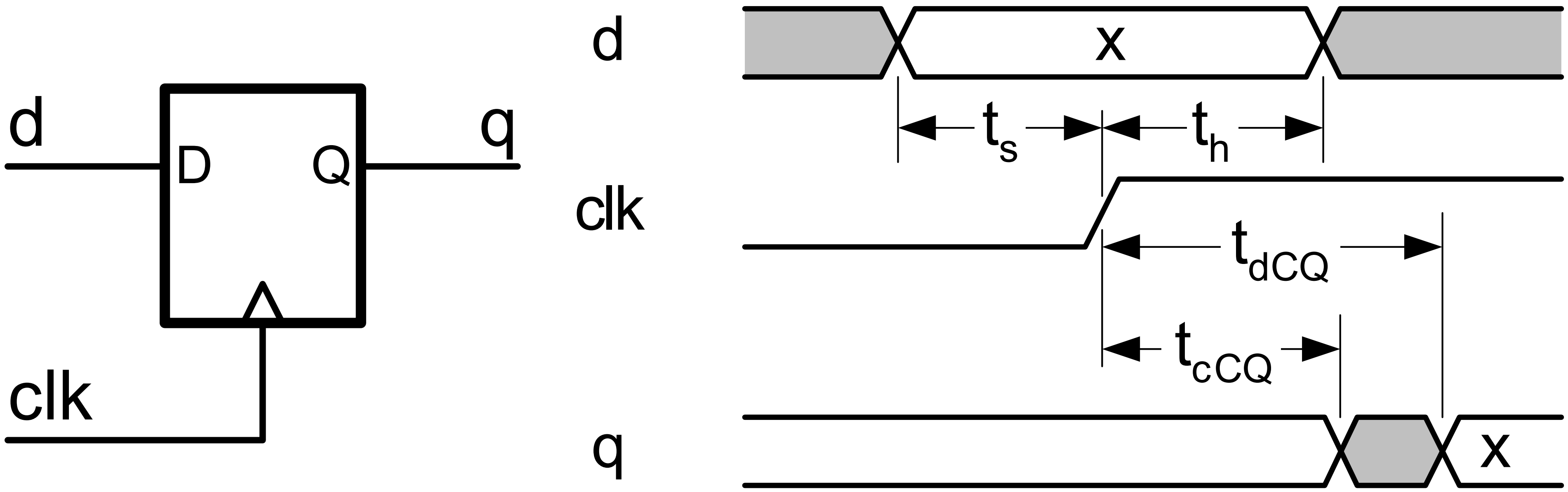
Digital circuits keep state

- Configuration
- Registers (data)

State is stored in flip-flops

- Keep a steady output until clock ticks

Our friend the D flip-flop



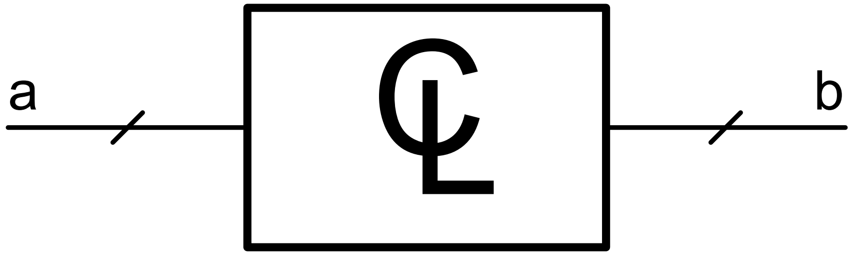
State is stored in flip-flops

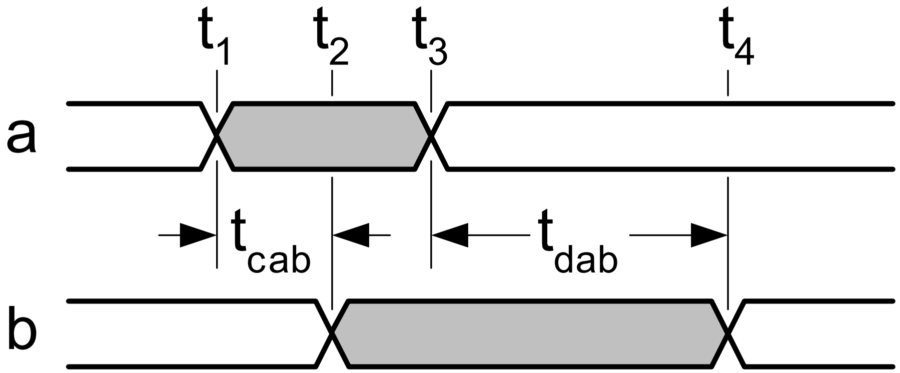
- Keep a steady output until clock ticks
- Output changes to input on clock tick

Propagation Delay and Contamination Delay

Propagation Delay – Time from last input change until last output change. (Input at steady state to output at steady state.)

Contamination Delay – Time from first input change until first output change. (Input contaminated to output contaminated)





Making Circuits Correct

Our clock speed can't be faster than the longest propagation delay of our system

- Otherwise, output might not be stable when we clock the flip-flop and store the results
- Faster transistors have higher clock speeds
- Shallower circuits have higher clock speeds

Contamination delay can't be faster than the time it takes for the flip-flop to store the state

- Otherwise, the state changes on the flip-flop before it's able to store it.

The Essence of Hardware Performance

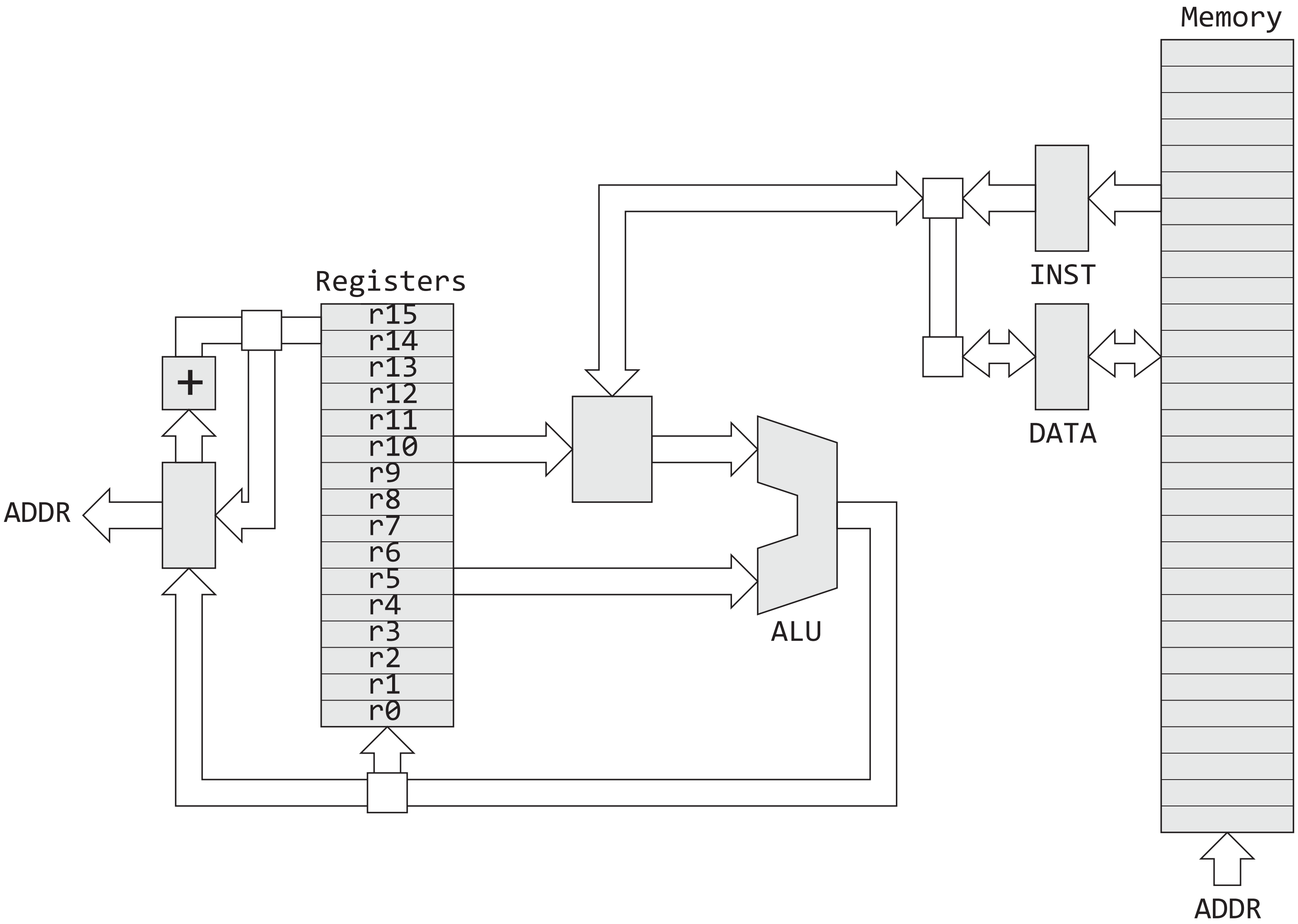
Software likes to think of hardware as just 1s and 0s

Hardware is built out of physical objects that rely on storing charge to build electrical fields to conduct.

Smaller transistors (narrower gates) are faster; shallower circuits are faster; speed is governed by slowest path.

These physical properties govern performance and so influence *how you design a high-performance system.*

Architecture (ARMv6)



Key Fact: Everything is organized into 32-bit words

Instructions (ARMv6)

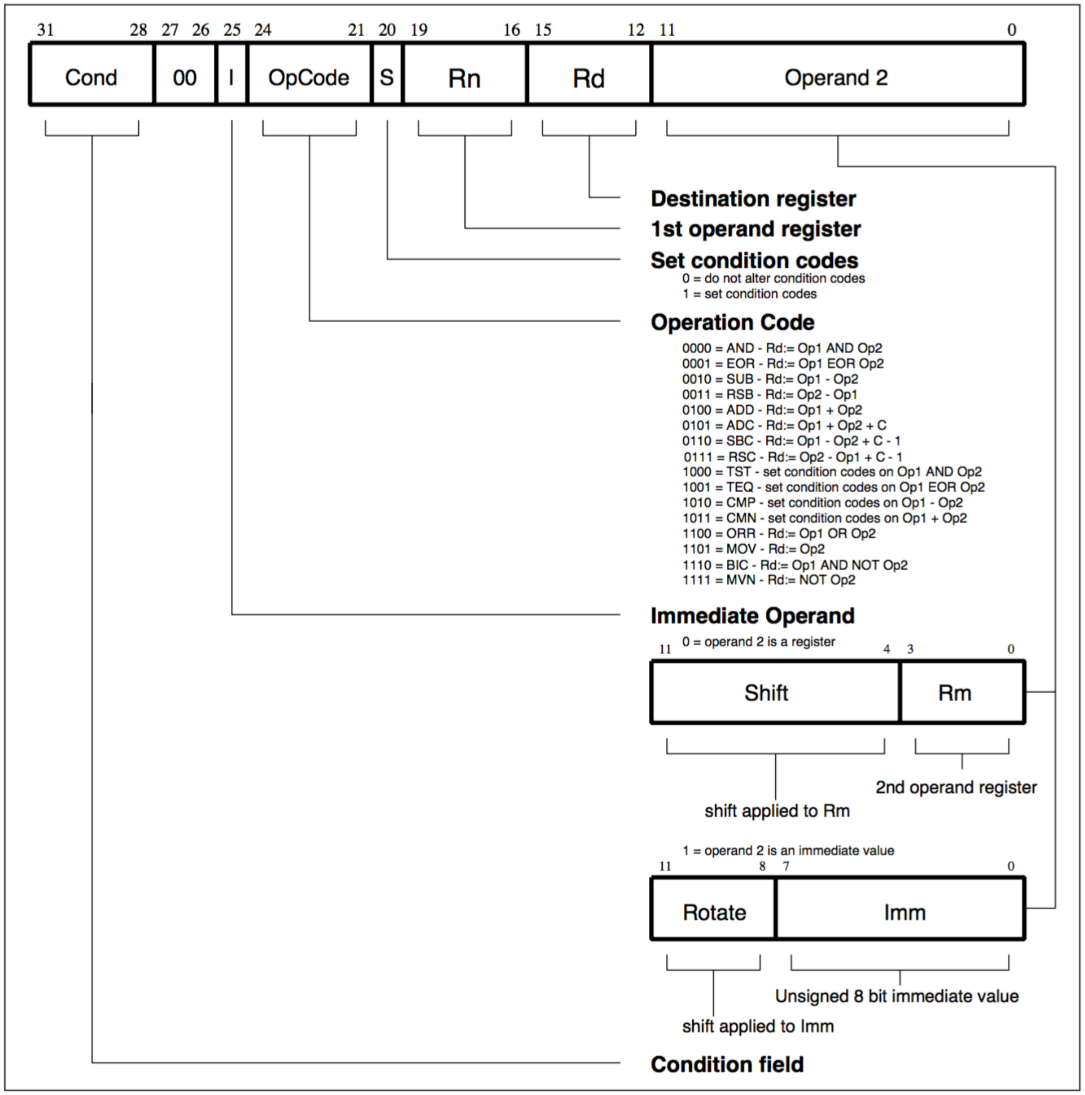

31
28
27
26
25
24
21
20
19
16
15
12
11
0
Cond
00
I
OpCode
S
Rn
Rd
Operand 2
Destination register
1st operand register
Set condition codes
0 = do not alter condition codes
1 = set condition codes
Operation Code
0000 = AND - Rd:= Op1 AND Op2
0001 = EOR - Rd:= Op1 EOR Op2
0010 = SUB - Rd:= Op1 - Op2
0011 = RSB - Rd:= Op2 - Op1
0100 = ADD - Rd:= Op1 + Op2
0101 = ADC - Rd:= Op1 + Op2 + C
0110 = SBC - Rd:= Op1 - Op2 + C - 1
0111 = RSC - Rd:= Op2 - Op1 + C - 1
1000 = TST - set condition codes on Op1 AND Op2
1001 = TEQ - set condition codes on Op1 EOR Op2
1010 = CMP - set condition codes on Op1 - Op2
1011 = CMN - set condition codes on Op1 + Op2
1100 = ORR - Rd:= Op1 OR Op2
1101 = MOV - Rd:= Op2
1110 = BIC - Rd:= Op1 AND NOT Op2
1111 = MVN - Rd:= NOT Op2
Immediate Operand
0 = operand 2 is a register
11
4
3
0
Shift
Rm
shift applied to Rm
2nd operand register
1 = operand 2 is an immediate value
11
8
7
0
Rotate
Imm
shift applied to Imm
Unsigned 8 bit immediate value
Condition field

Figure 4-4: Data processing instructions

```
add r0, r0, r1              r0=r0+r1

ldr r0, [r1]                r0=mem[r1]

add r0, r1, #1<<4           r0=r1+(1<<4)

add r0, r1, #1
```

```
              add     r1    r0                  #1
1110  00  1  0100  0  0001  0000  0000  0000  0001

1110  0010  1000  0001  0000  0000  0000  0001
   E     2     8     1     0     0     0     1
```

Pipelining:
Throughput and Latency

No pipelining

Clock cycle

Instruction

	1	2	3	4	5	6	7	8	9
1	Instruction 1								
2						Instruction 2			
3									
4									
5									

Instruction Pipelining

Clock cycle

Instruction

	1	2	3	4	5	6	7	8	9
1	Fetch	Decode	Execute	Memory	Write				
2									
3									
4									
5									

Instruction Pipelining

Clock cycle

Instruction

	1	2	3	4	5	6	7	8	9
1	Fetch	Decode	Execute	Memory	Write				
2		Fetch	Decode	Execute	Memory	Write			
3									
4									
5									

Instruction Pipelining

Clock cycle

Instruction

	1	2	3	4	5	6	7	8	9
1	Fetch	Decode	Execute	Memory	Write				
2		Fetch	Decode	Execute	Memory	Write			
3			Fetch	Decode	Execute	Memory	Write		
4				Fetch	Decode	Execute	Memory	Write	
5					Fetch	Decode	Execute	Memory	Write

Instruction Pipelining

Clock cycle

Instruction

	1	2	3	4	5	6	7	8	9
1	Fetch	Decode	Execute	Memory	Write				
2		Fetch	Decode	Execute	Memory	Write			
3			Fetch	Decode	Execute	Memory	Write		
4				Fetch	Decode	Execute	Memory	Write	
5					Fetch	Decode	Execute	Memory	Write

Throughput vs. latency

Pipeline stages increase throughput: you can execute one instruction/cycle

Pipeline stages increase latency somewhat: each clock cycle is the *maximum* of all of the stages

You will see this tradeoff often in architecture. Memory access takes a long time, so you pipeline or buffer it to mask that latency: you execute on other data while waiting

Data Movement

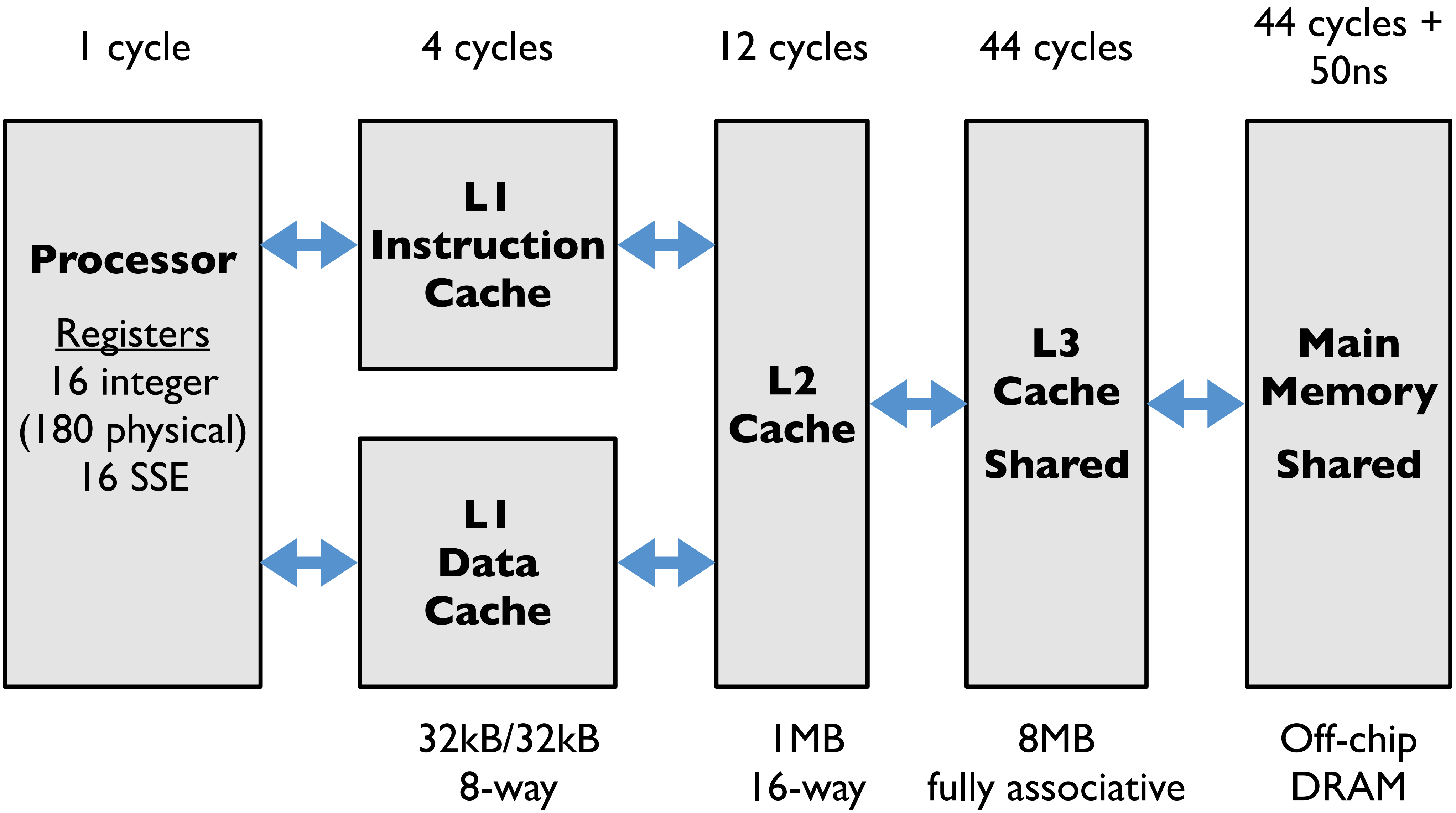
Generally speaking, arithmetic is inexpensive: GPUs have thousands of tiny, lightweight cores to operate on pixels

The major performance bottleneck and energy cost is usually moving data

Bigger memories are slower and more expensive

Hierarchy of memories: keep used data close and fast

Memory Hierarchy (Sky Lake 4GHz)



1 cycle

4 cycles

12 cycles

44 cycles

44 cycles + 50ns

Processor

<u>Registers</u>
16 integer
(180 physical)
16 SSE

L1 Instruction Cache

L1 Data Cache

L2 Cache

L3 Cache

Shared

Main Memory

Shared

32kB/32kB
8-way

1MB
16-way

8MB
fully associative

Off-chip
DRAM

Memory Ports

There are wires to a memory that allow reads and writes (address lines and data lines), called a port

If one instruction is using the port, another instruction has to wait for the port to be available, even if it is accessing a different block of the memory

Multiport memories allow multiple concurrent accesses (at a cost of complexity)

Parallelism

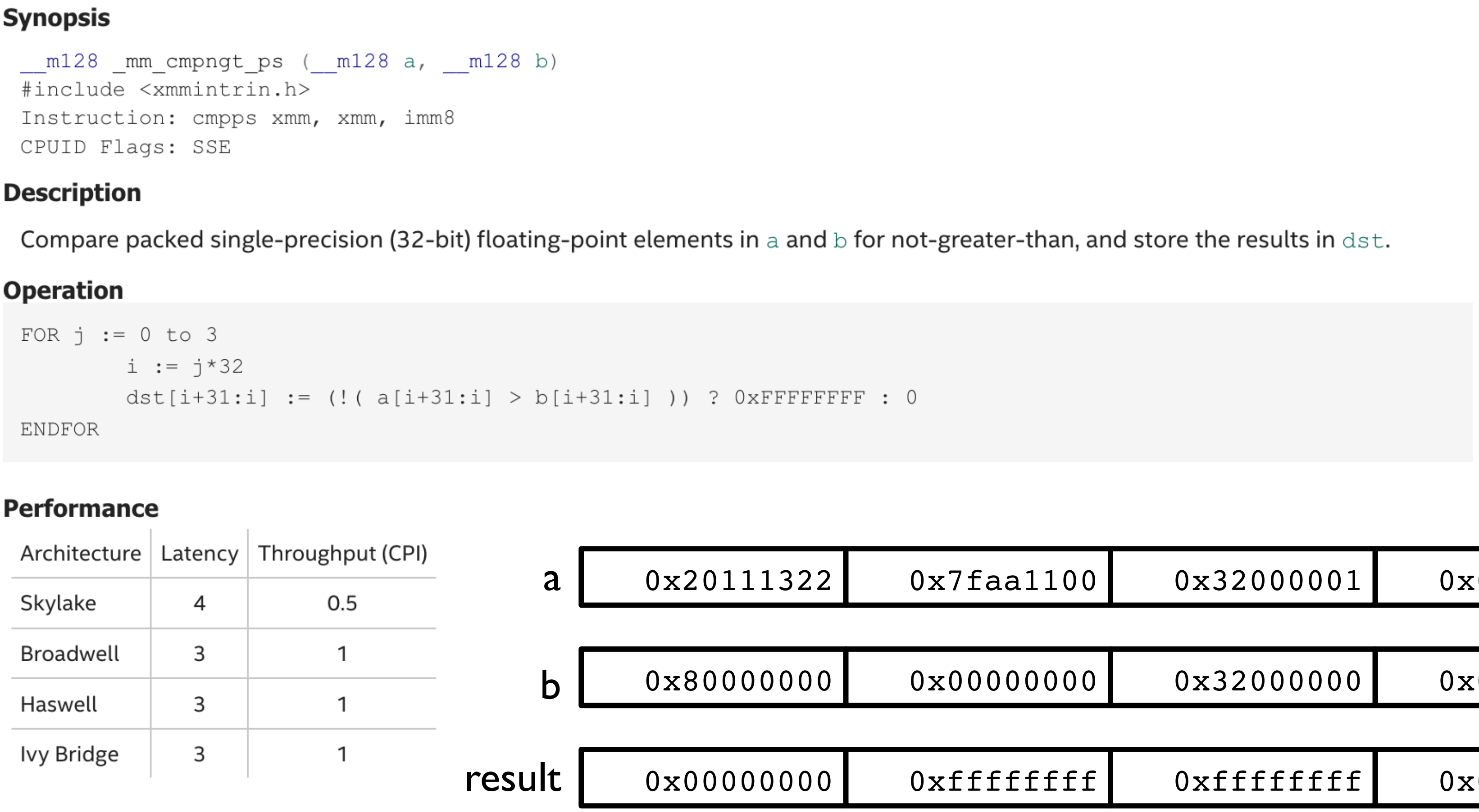
Hardware is inherently parallel: all circuits and gates can operate at once (with power limitations)

Many levels of parallelism possible

- Data-level: single instruction multiple data (SIMD)
- Instruction-level: run multiple instructions at once (multiple issue)
- Thread-level: multiple cores run separate streams of instructions in parallel
- Request level: multiple programs run on loosely coupled processors

SIMD

Example: Intel SSE cmpps instruction



Synopsis

```
__m128 _mm_cmpngt_ps (__m128 a, __m128 b)
#include <xmmintrin.h>
Instruction: cmpps xmm, xmm, imm8
CPUID Flags: SSE
```

Description

Compare packed single-precision (32-bit) floating-point elements in a and b for not-greater-than, and store the results in dst.

Operation

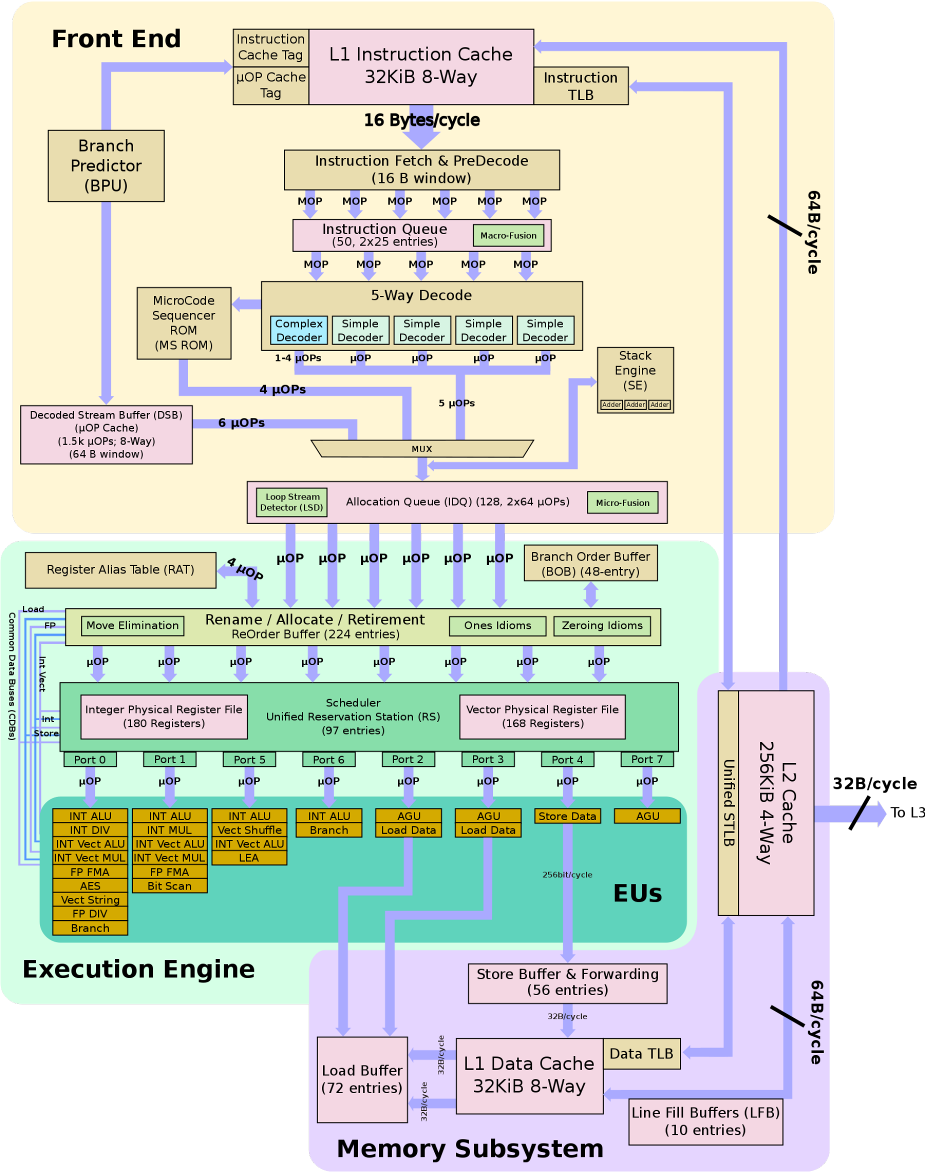
```
FOR j := 0 to 3
        i := j*32
        dst[i+31:i] := (!( a[i+31:i] > b[i+31:i] )) ? 0xFFFFFFFF : 0
ENDFOR
```

Performance

Architecture	Latency	Throughput (CPI)
Skylake	4	0.5
Broadwell	3	1
Haswell	3	1
Ivy Bridge	3	1

a	0x20111322	0x7faa1100	0x32000001	0x00000000
b	0x80000000	0x00000000	0x32000000	0x00000000
result	0x00000000	0xffffffff	0xffffffff	0x00000000

Skylake (retired 2019) pipeline



Can execute up to 16 bytes of instructions/clock

Instructions decoded into simpler "micro-ops": can execute 5 µops/cycle

Instructions reordered to improve parallelism (mask delays)

Limits of Parallelism

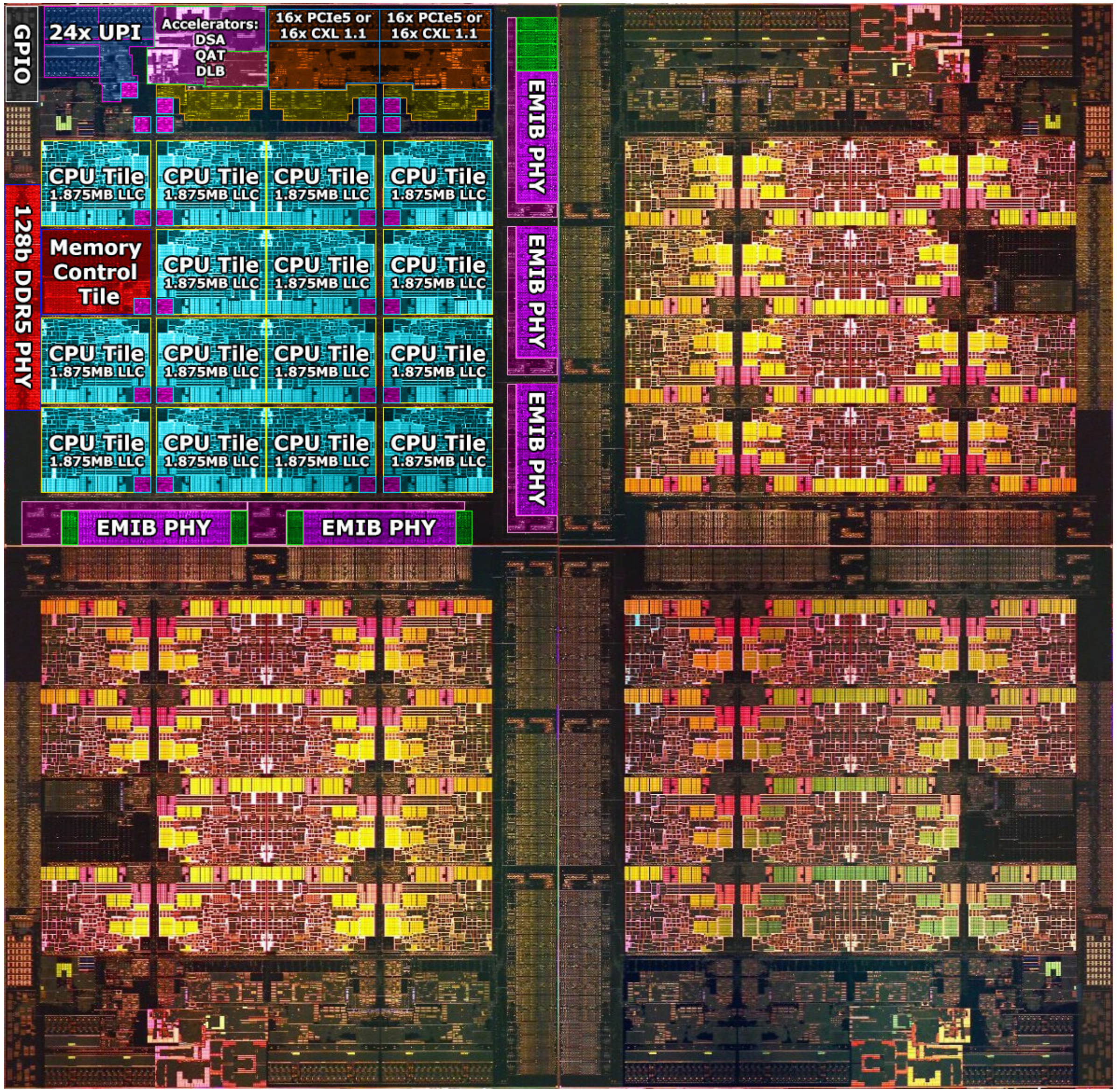
```
add r2, r0, r1          x = x + y;
mul r5, r2, #6          x = x * 6;
ldr r1, [r5]            y = *((int*)x);
```

Data dependencies

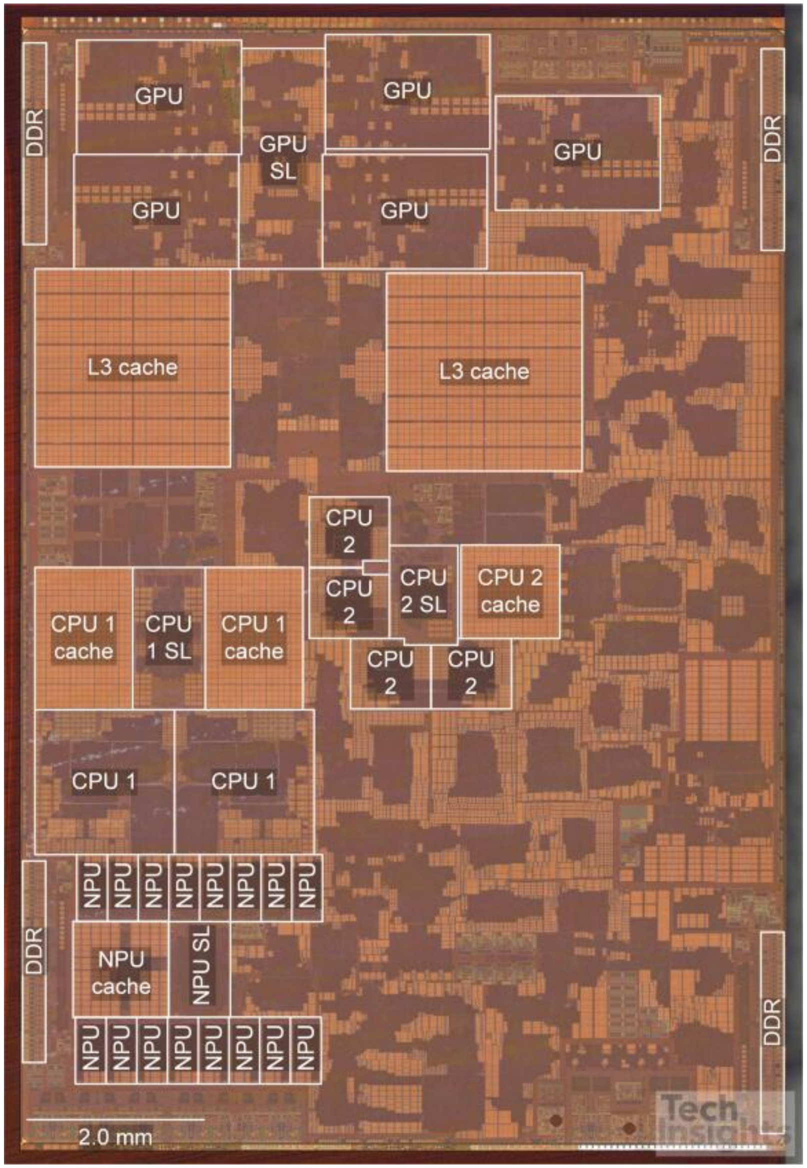
- Can't multiply until addition completes (r2 is input)
- Can't load until multiply completes (r5 is input)

Parallel programs minimize these dependencies, restructure programs for high parallelism

Sapphire Rapids (multicore)



A15 (heterogenous cores)



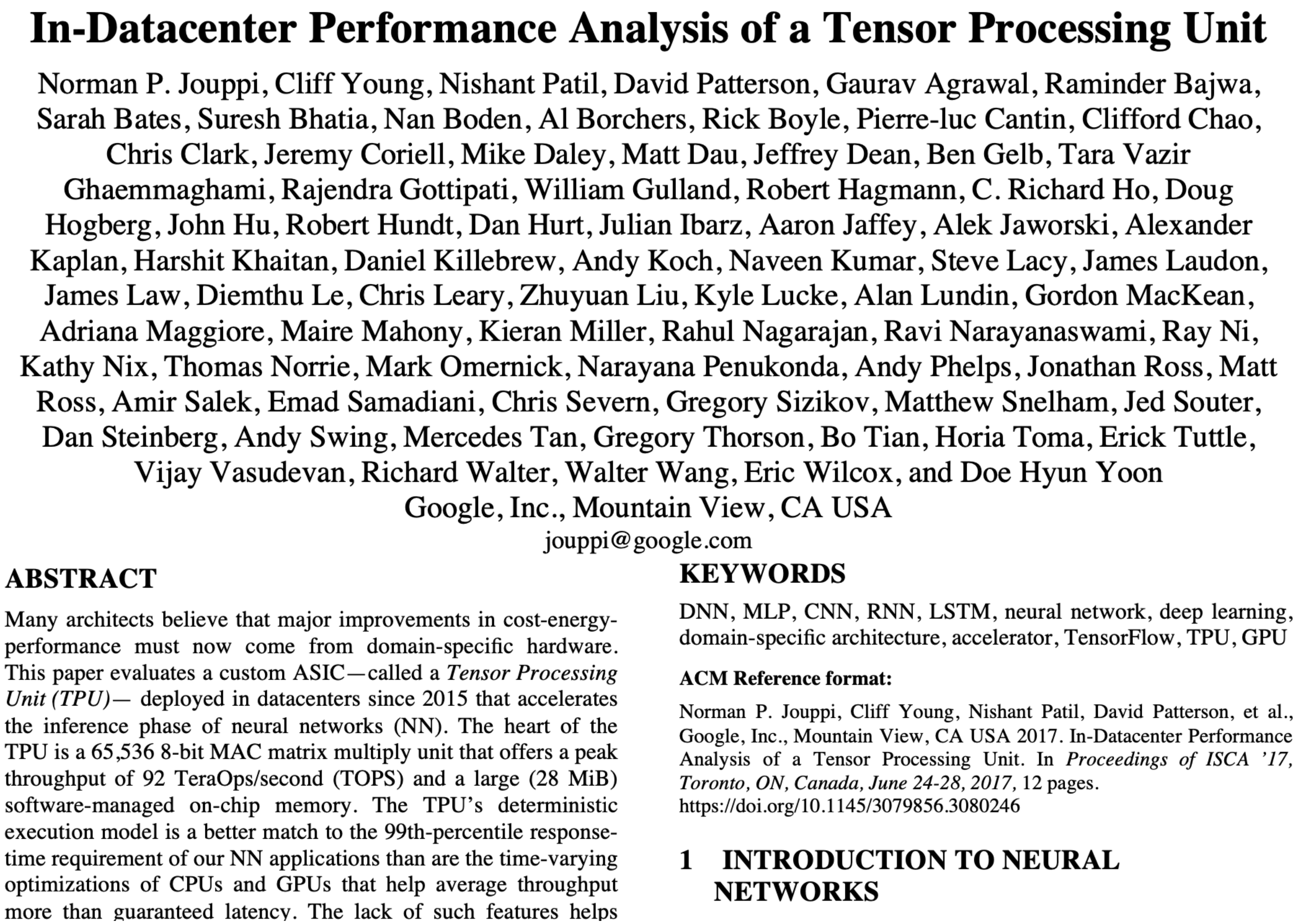
CPU 1: high performance core

CPU 2: efficiency core

NPU: Neural processing unit

GPU: graphics processing unit

CPU < GPU < FPGA < ASIC



In-Datacenter Performance Analysis of a Tensor Processing Unit

Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon
Google, Inc., Mountain View, CA USA
jouppi@google.com

ABSTRACT

Many architects believe that major improvements in cost-energy-performance must now come from domain-specific hardware. This paper evaluates a custom ASIC—called a *Tensor Processing Unit (TPU)*— deployed in datacenters since 2015 that accelerates the inference phase of neural networks (NN). The heart of the TPU is a 65,536 8-bit MAC matrix multiply unit that offers a peak throughput of 92 TeraOps/second (TOPS) and a large (28 MiB) software-managed on-chip memory. The TPU's deterministic execution model is a better match to the 99th-percentile response-time requirement of our NN applications than are the time-varying optimizations of CPUs and GPUs that help average throughput more than guaranteed latency. The lack of such features helps

KEYWORDS

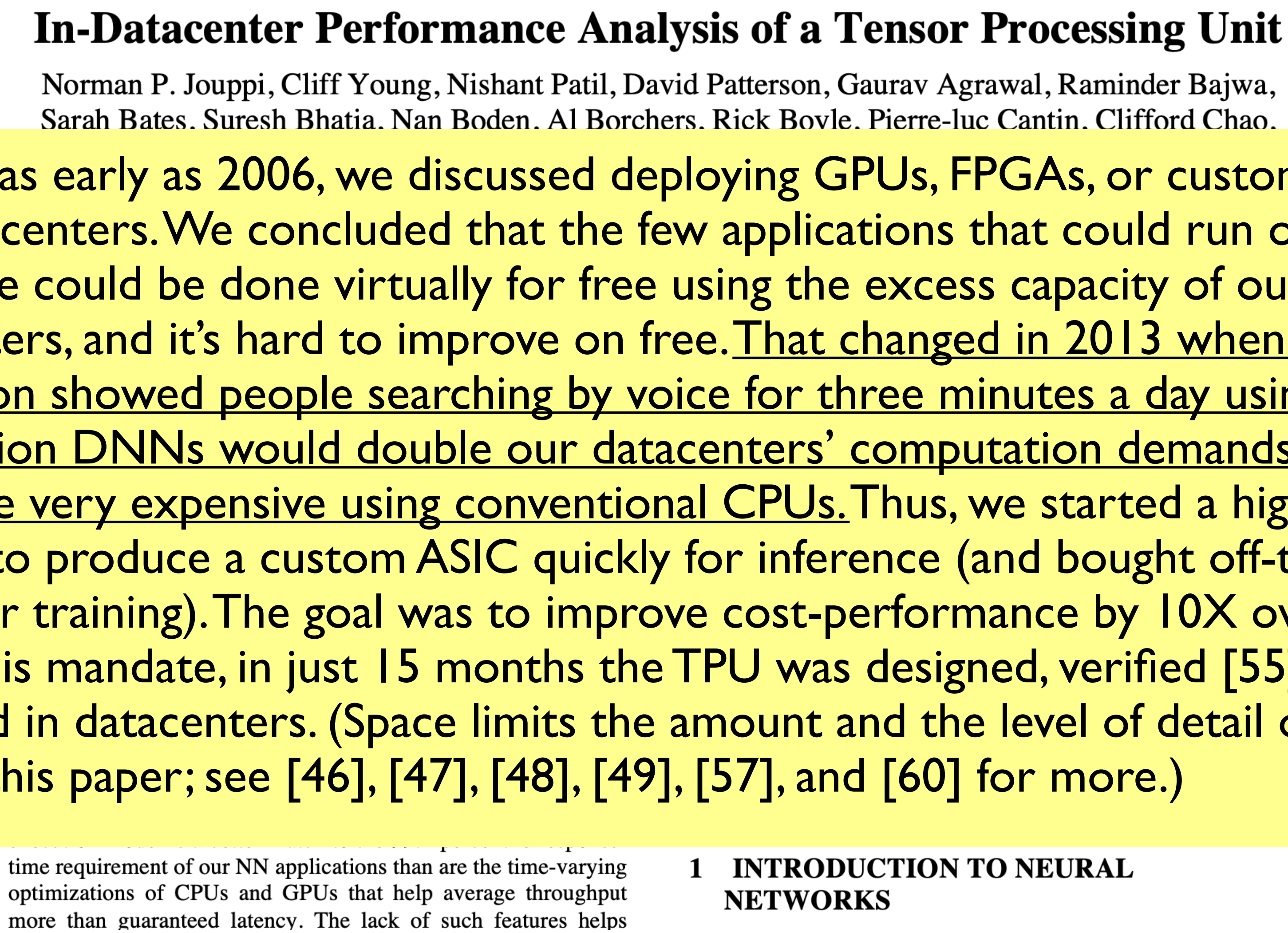
DNN, MLP, CNN, RNN, LSTM, neural network, deep learning, domain-specific architecture, accelerator, TensorFlow, TPU, GPU

ACM Reference format:

Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, et al., Google, Inc., Mountain View, CA USA 2017. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of ISCA '17, Toronto, ON, Canada, June 24-28, 2017,* 12 pages.
https://doi.org/10.1145/3079856.3080246

1 INTRODUCTION TO NEURAL NETWORKS

CPU < GPU < FPGA < ASIC



In-Datacenter Performance Analysis of a Tensor Processing Unit

Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa,
Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao,

time requirement of our NN applications than are the time-varying optimizations of CPUs and GPUs that help average throughput more than guaranteed latency. The lack of such features helps

1 INTRODUCTION TO NEURAL NETWORKS

Starting as early as 2006, we discussed deploying GPUs, FPGAs, or custom ASICs in our datacenters. We concluded that the few applications that could run on special hardware could be done virtually for free using the excess capacity of our large datacenters, and it's hard to improve on free. That changed in 2013 when a projection showed people searching by voice for three minutes a day using speech recognition DNNs would double our datacenters' computation demands, which would be very expensive using conventional CPUs. Thus, we started a high- priority project to produce a custom ASIC quickly for inference (and bought off-the-shelf GPUs for training). The goal was to improve cost-performance by 10X over GPUs. Given this mandate, in just 15 months the TPU was designed, verified [55], built, and deployed in datacenters. (Space limits the amount and the level of detail on the TPU in this paper; see [46], [47], [48], [49], [57], and [60] for more.)

Historical Impact of Moore's Law

1960-1985: Cheaper, smaller chips

1982-2006: Very Large System Integration (VLSI) and EDA (electronic design automation) takes off; the "golden age" of computer architecture. Exponential growth in computing performance.

2006-Today: Performance mostly plateaus: move to multi-core, parallel architectures (GPUs, TPUs) and accelerators (e.g., AES instruction)