Designing Languages for Designing Hardware

Adrian Sampson
Cornell

How do we harness the power of computing?

HCI

What are the secrets of human intelligence?

ML

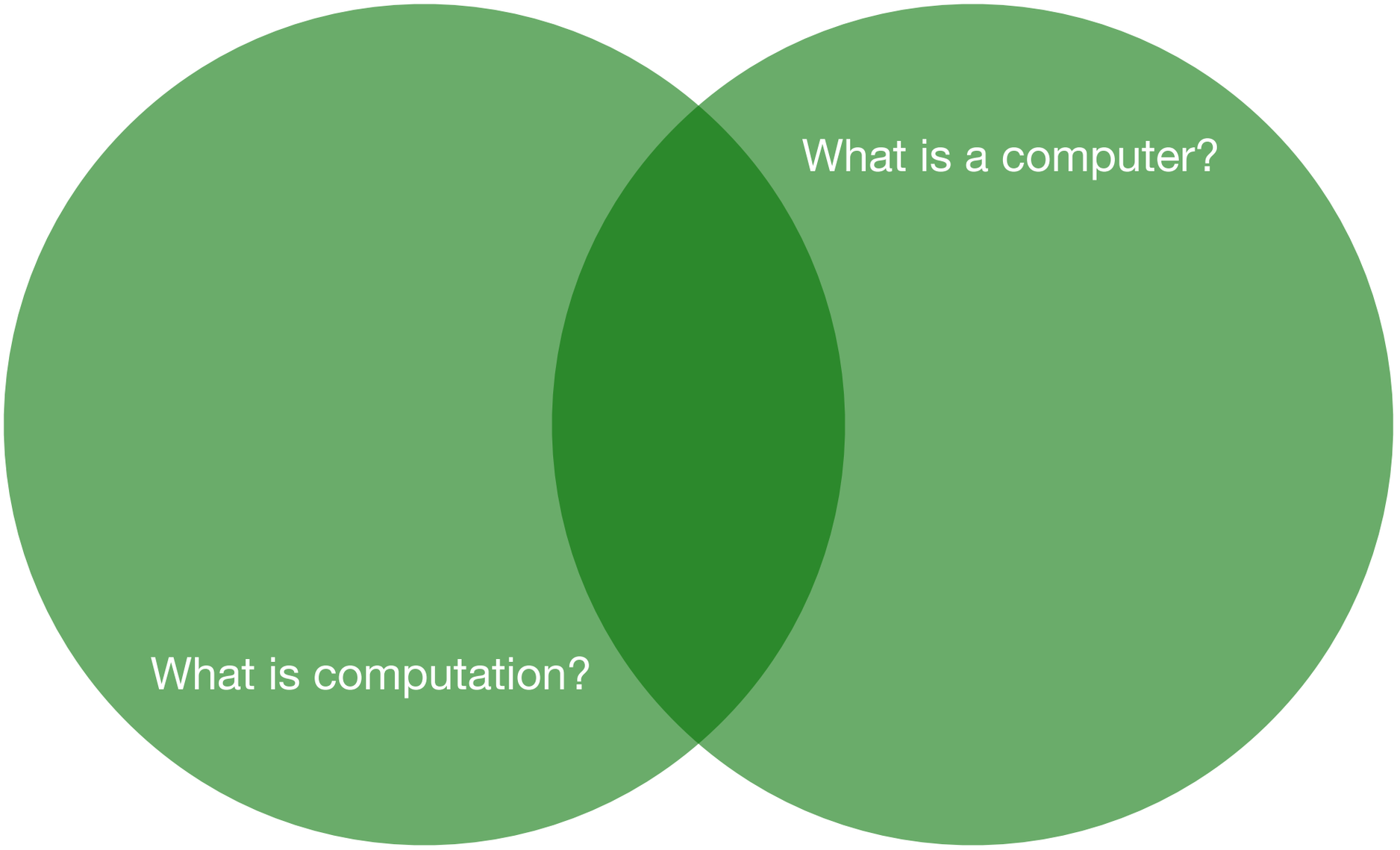
How do comp wh the

What is computation, really?

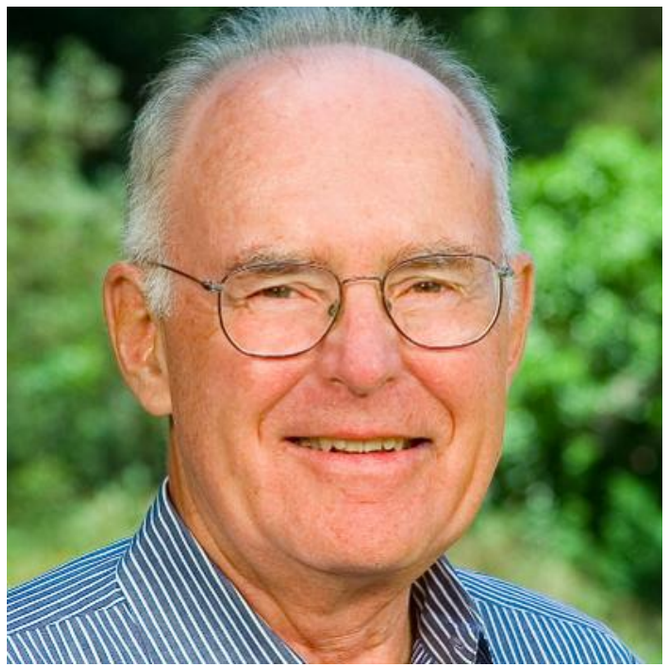
PL

What is a computer?

Arch

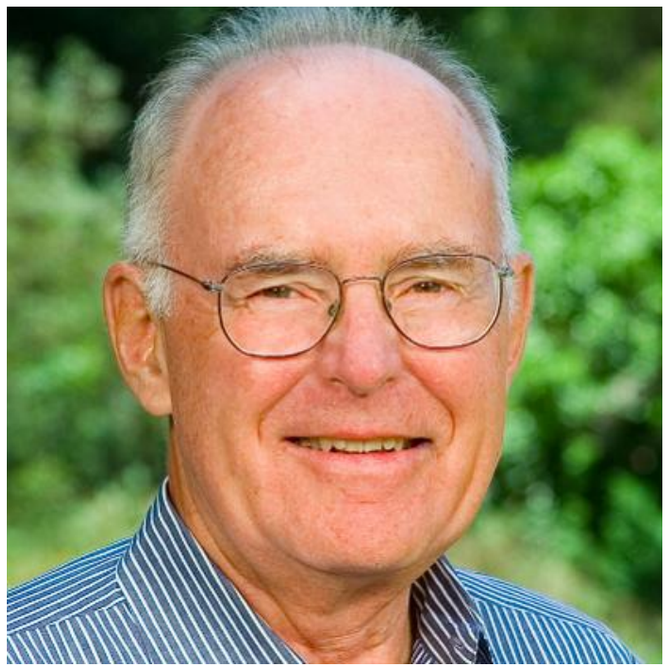

What is a computer?
What is computation?

“The complexity for minimum component costs has increased at a rate of roughly a factor of two per year.”



Gordon Moore
co-founder of Intel
1965

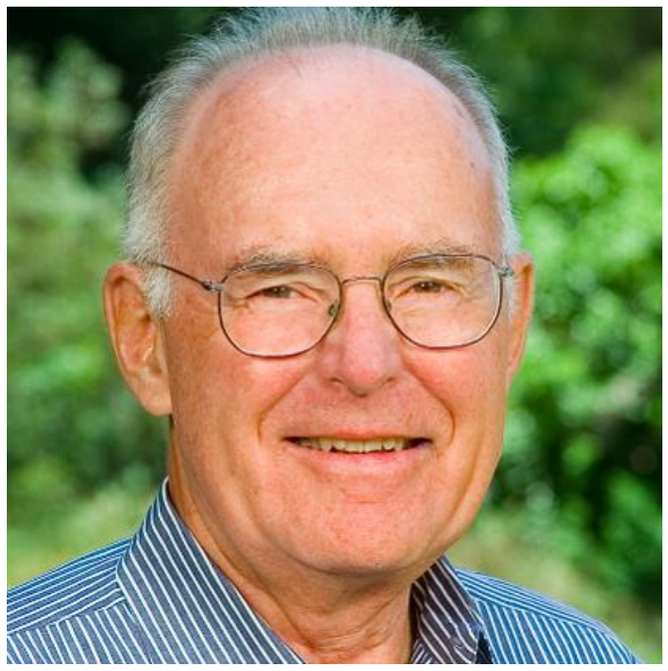
The size of a single transistor decreases by half every 18 months.



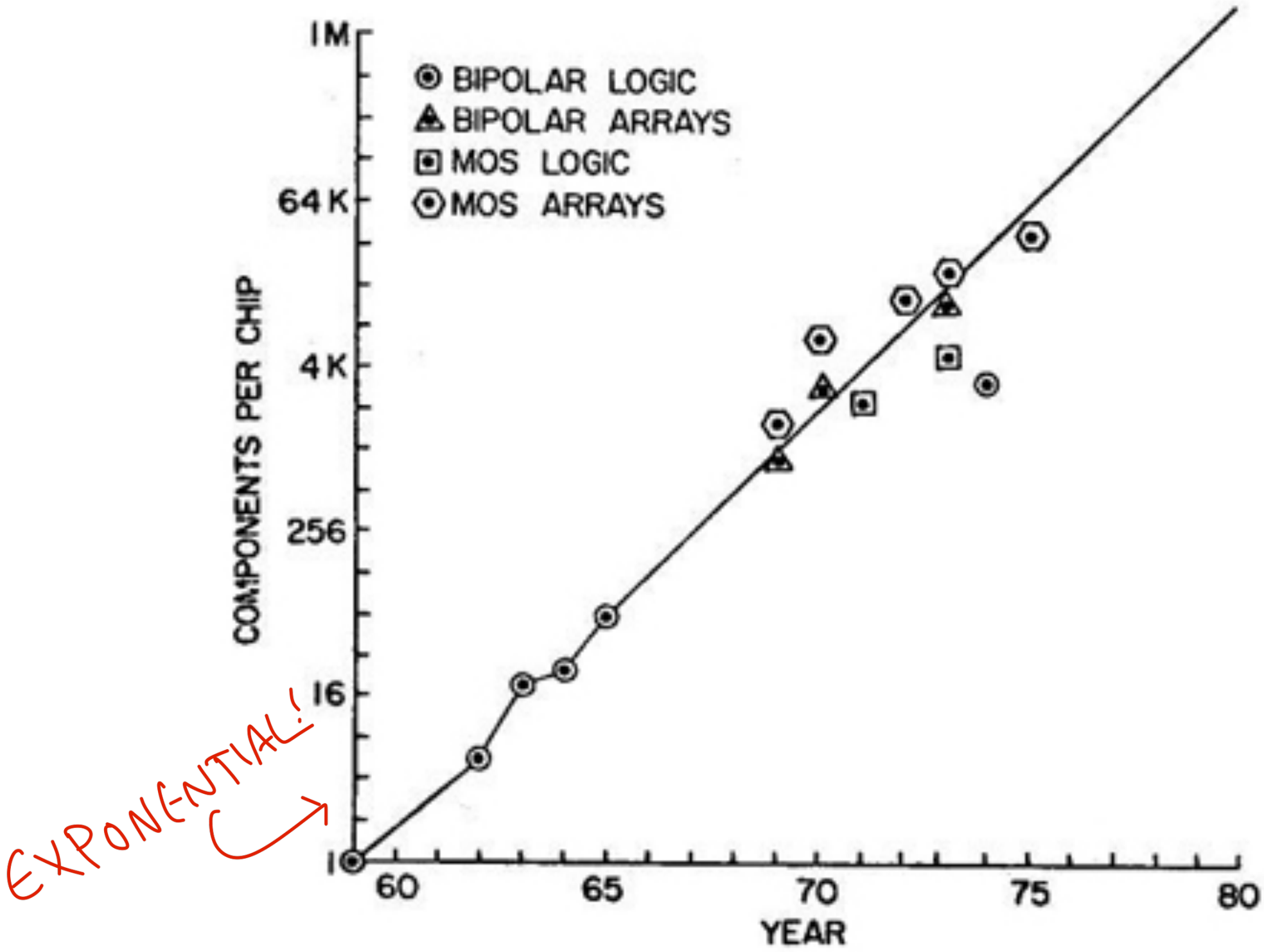
Gordon Moore
co-founder of Intel
1965

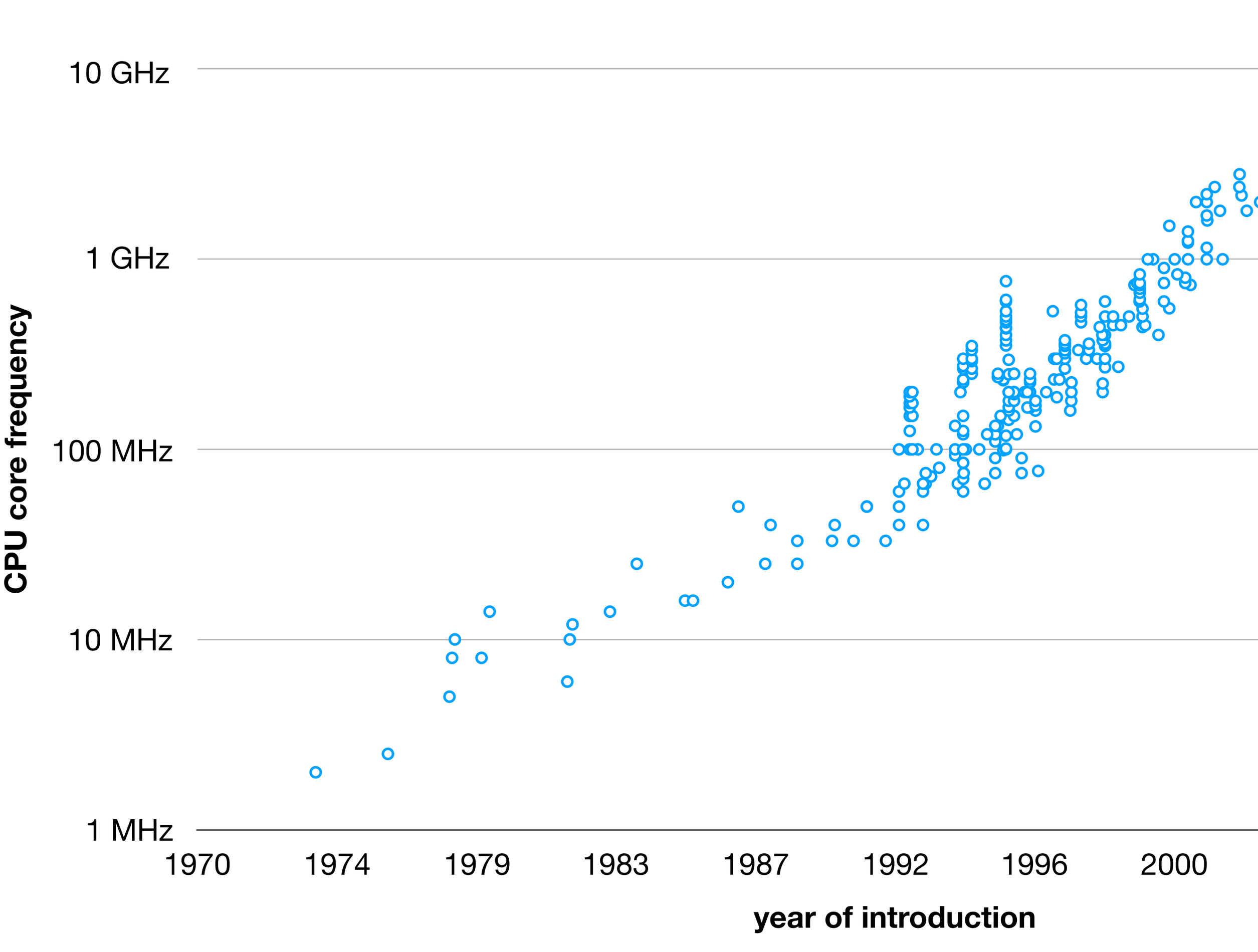
and cost, and power…

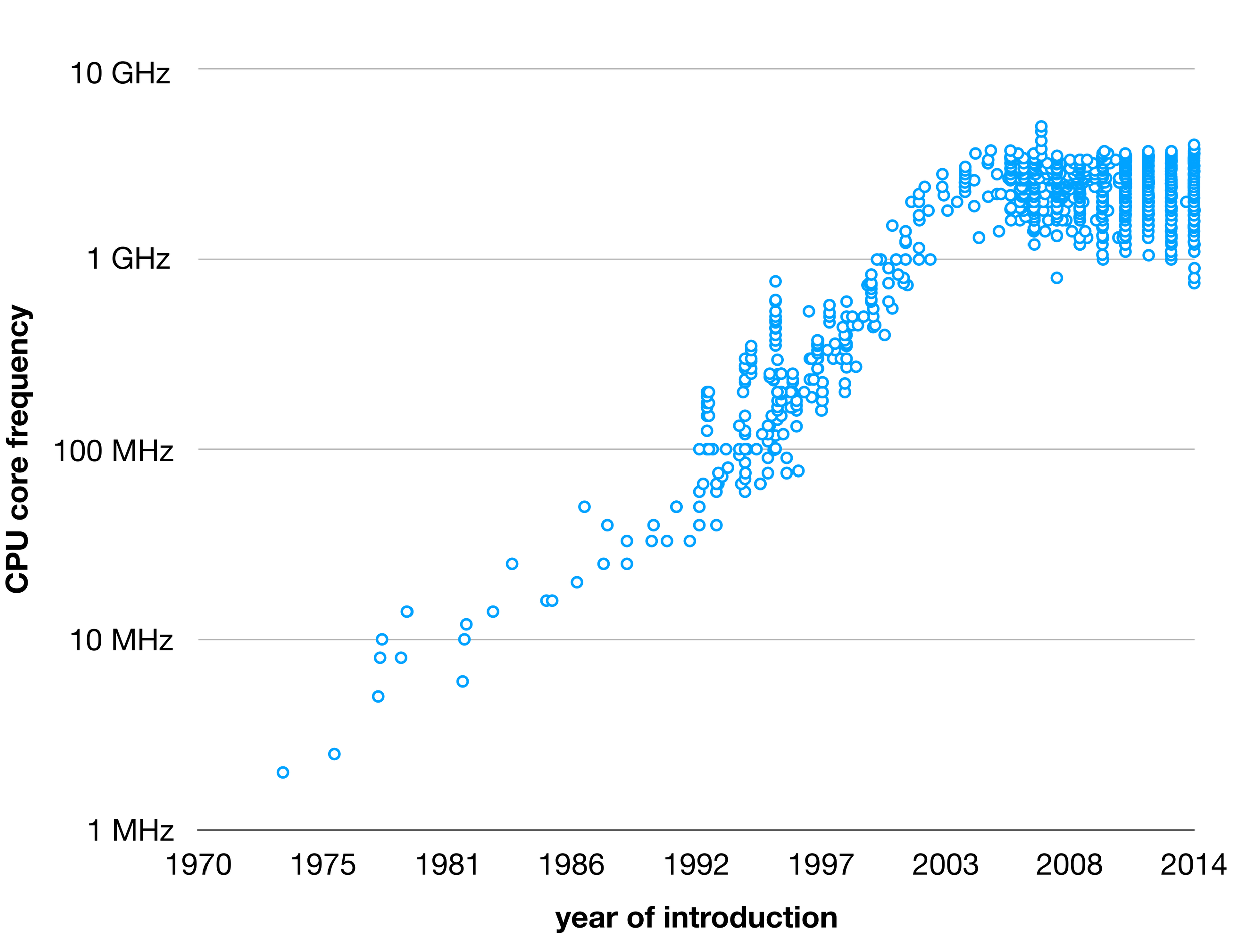
The size^ of a single transistor decreases by half every 18 months.



Gordon Moore
co-founder of Intel
1965


1M
64K
4K
256
16
1
COMPONENTS PER CHIP
BIPOLAR LOGIC
BIPOLAR ARRAYS
MOS LOGIC
MOS ARRAYS
60
65
70
75
80
YEAR
EXPONENTIAL!


10 GHz
1 GHz
100 MHz
10 MHz
1 MHz
CPU core frequency
1970
1974
1979
1983
1987
1992
1996
2000
year of introduction


10 GHz
1 GHz
100 MHz
10 MHz
1 MHz
CPU core frequency
1970
1975
1981
1986
1992
1997
2003
2008
2014
year of introduction

free lunch

time immemorial

2005

2015

exponential single-threaded performance scaling!

(not to scale)

free lunch

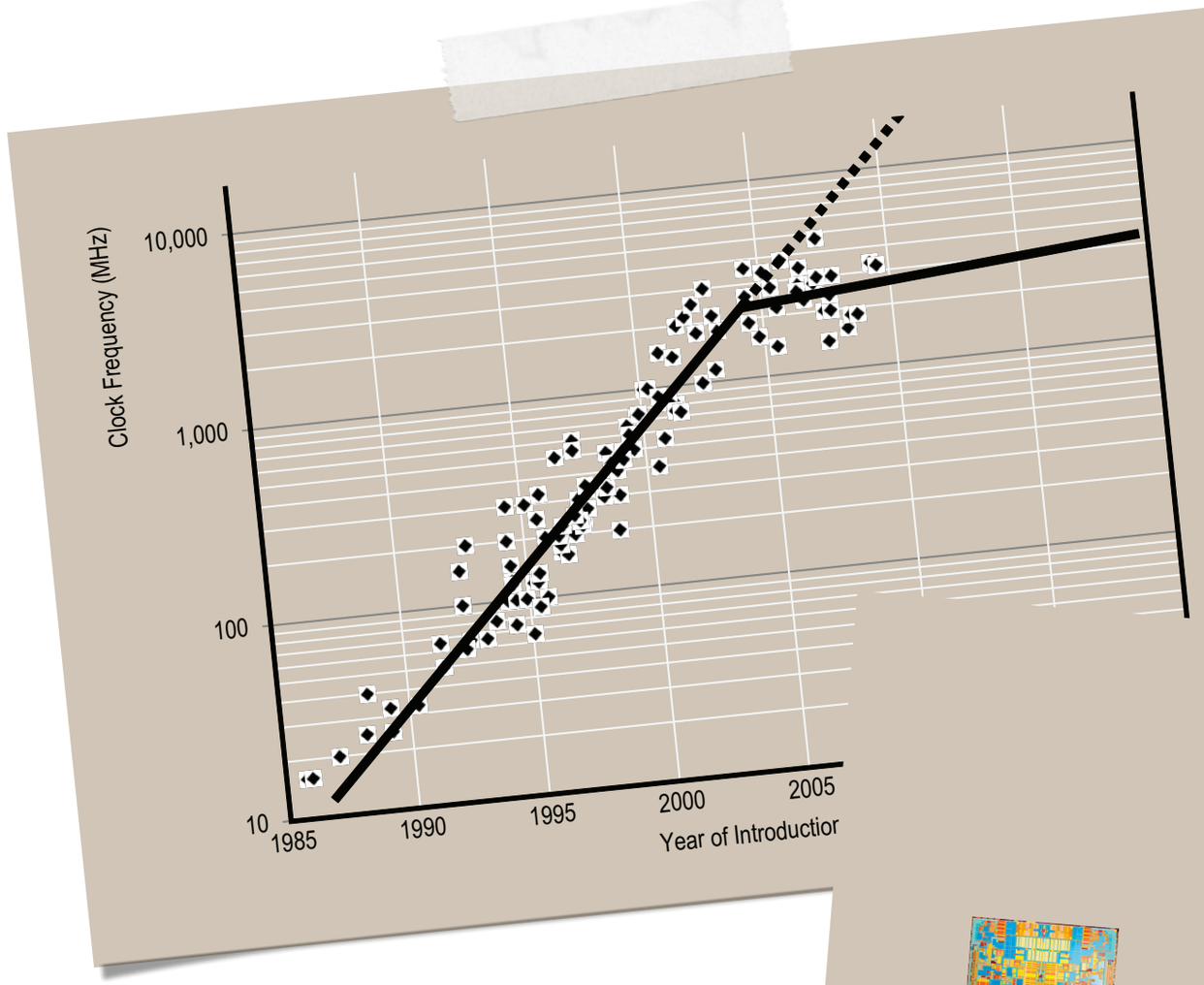
multicore era

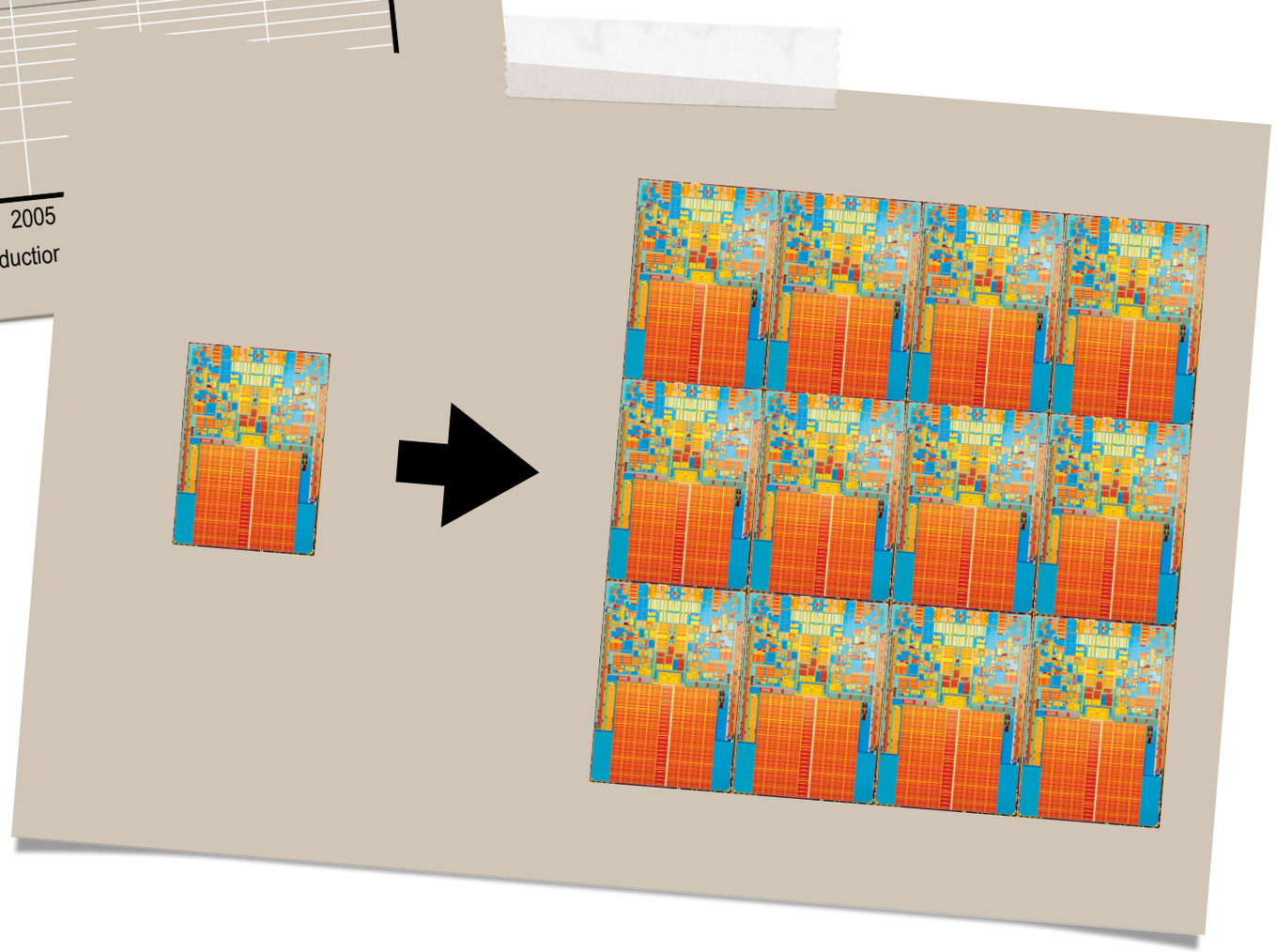
time immemorial

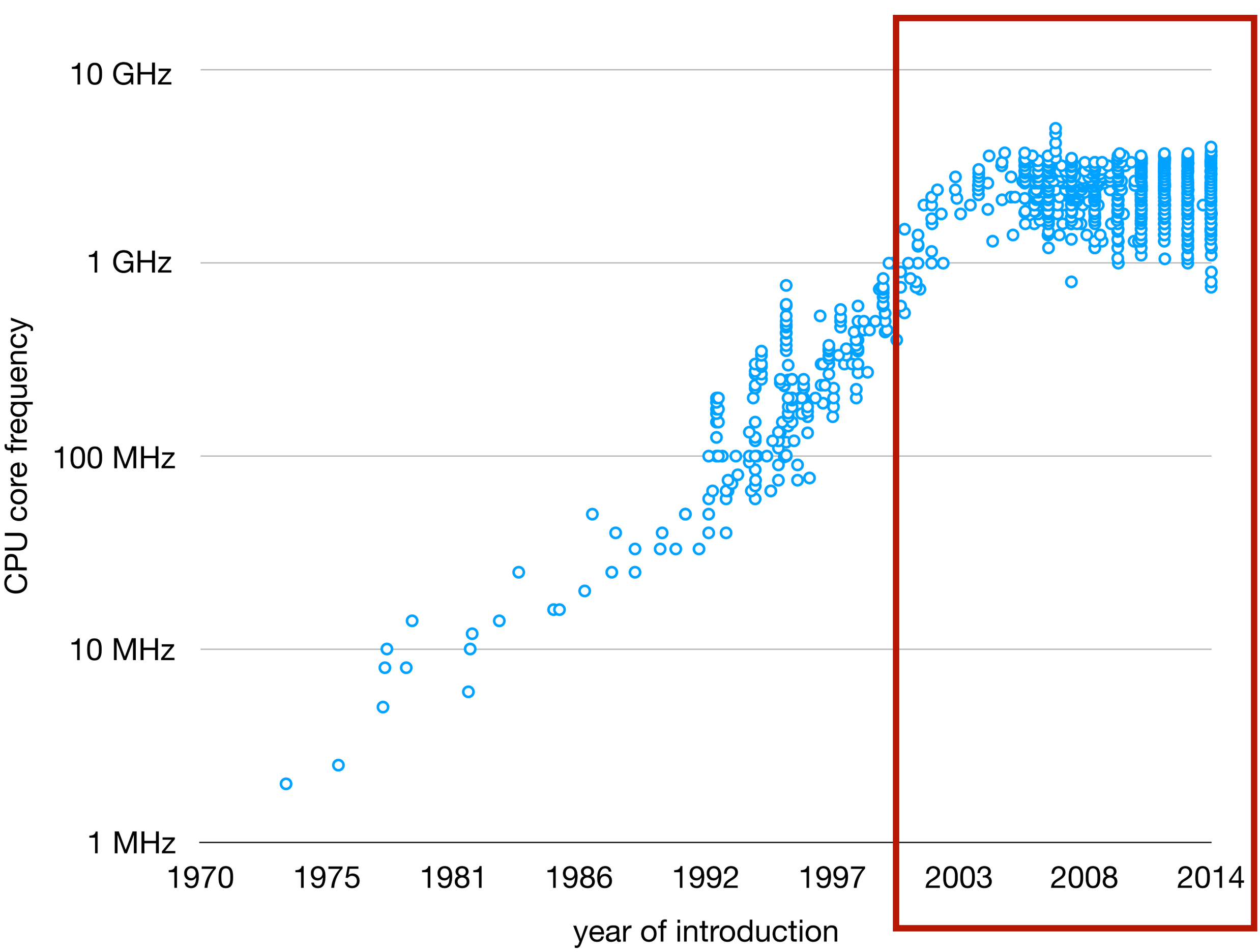
2005

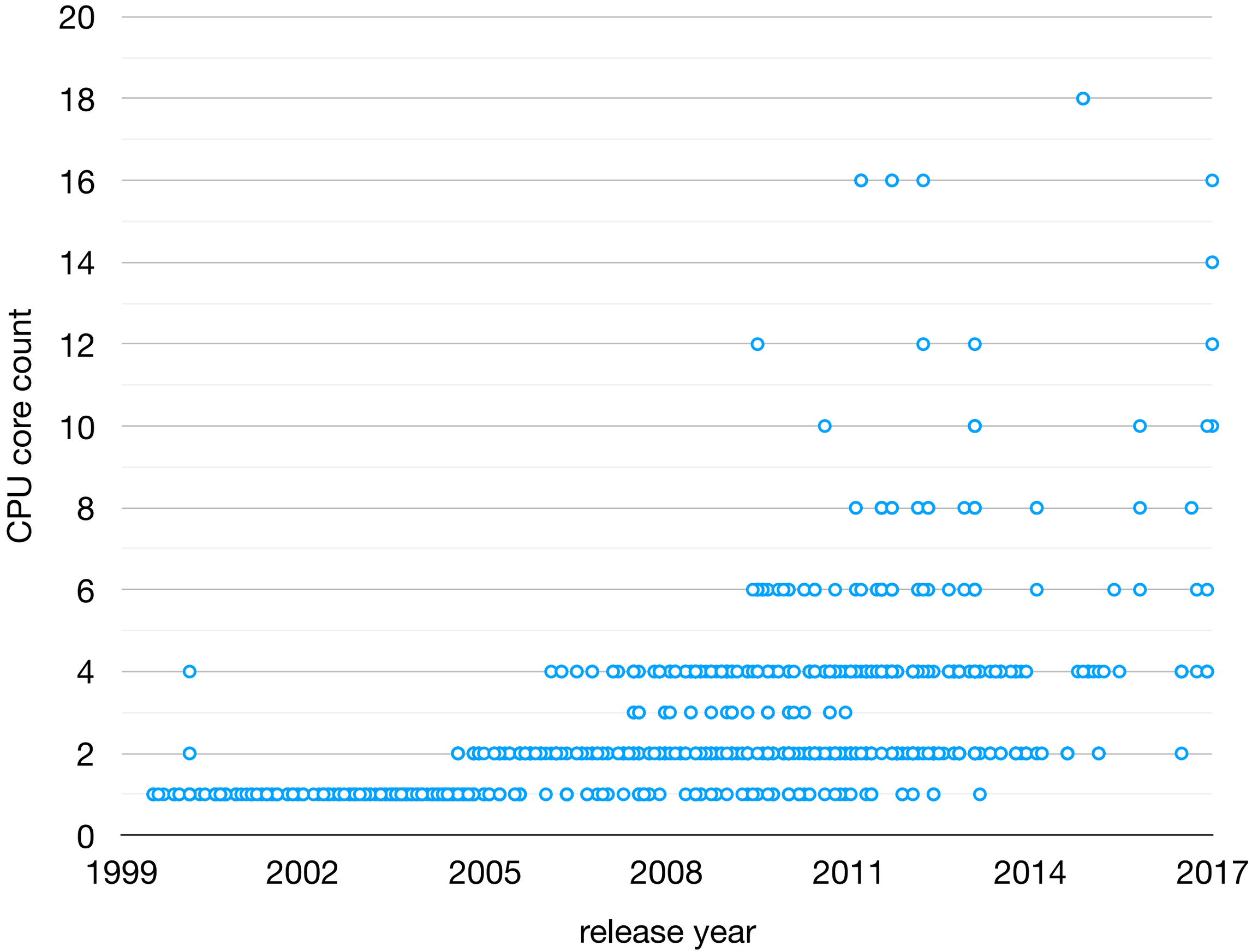
2015

we'll scale the number of cores instead


Clock Frequency (MHz)
10,000
1,000
100
10
1985
1990
1995
2000
2005
Year of Introductior




10 GHz
1 GHz
100 MHz
10 MHz
1 MHz
CPU core frequency
1970
1975
1981
1986
1992
1997
2003
2008
2014
year of introduction


CPU core count
20
18
16
14
12
10
8
6
4
2
0
1999
2002
2005
2008
2011
2014
2017
release year

free lunch **multicore era** **who knows?**

time immemorial

2005

2015

?

?

?

?

?

The performance returns from Moore's Law ended in 2015!

The only way forward is to trade off generality for efficiency!

A New Golden Age f

Domain Specific Ha

A New Golden Age for Computer Architecture:
Domain-Specific Hardware, Compiler-Enabled Architecture Design,
Enhanced Security, Open Instruction Sets, and
Agile Chip Development

John L. Hennessy and David A. Patterson


CATAPULT
DOUG BURGER
(MICROSOFT, FORMERLY UT)

A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services

Andrew Putnam Adrian M. Caulfield Eric S. Chung Derek Chiou[1]
Kypros Constantinides[2] John Demme[3] Hadi Esmaeilzadeh[4] Jeremy Fowers
Gopi Prashanth Gopal Jan Gray Michael Haselman Scott Hauck[5] Stephen Heil
Amir Hormati[6] Joo-Young Kim Sitaram Lanka James Larus[7] Eric Peterson
Simon Pope Aaron Smith Jason Thong Phillip Yi Xiao Doug Burger

Microsoft

23 AUTHORS!

Abstract

Datacenter workloads demand high computational capabilities, flexibility, power efficiency, and low cost. It is challenging to improve all of these factors simultaneously. To advance datacenter capabilities beyond what commodity server designs can provide, we have designed and built a composable, reconfigurable fabric to accelerate portions of large-scale software services. Each instantiation of the fabric consists of a 6x8 2-D torus of high-end Stratix V FPGAs embedded into a half-rack of 48 machines. One FPGA is placed into each server, accessible through PCIe, and wired directly to other FPGAs with pairs of 10 Gb SAS cables.

In this paper, we describe a medium-scale deployment of this fabric on a bed of 1,632 servers, and measure its efficacy in accelerating the Bing web search engine. We describe the requirements and architecture of the system, detail the

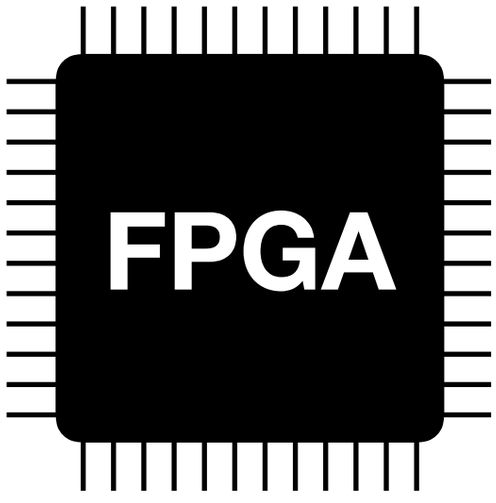
desirable to reduce management issues and to provide a consistent platform that applications can rely on. Second, datacenter services evolve extremely rapidly, making non-programmable hardware features impractical. Thus, datacenter providers are faced with a conundrum: they need continued improvements in performance and efficiency, but cannot obtain those improvements from general-purpose systems.

Reconfigurable chips, such as Field Programmable Gate Arrays (FPGAs), offer the potential for flexible acceleration of many workloads. However, as of this writing, FPGAs have not been widely deployed as compute accelerators in either datacenter infrastructure or in client devices. One challenge traditionally associated with FPGAs is the need to fit the accelerated function into the available reconfigurable area. One could virtualize the FPGA by reconfiguring it at run-time to support more functions than could fit into a single device. However, current reconfiguration times for standard FPGAs

RTL

Verilog VHDL Bluespec Chisel

register-transfer level



C

RTL

register-transfer level

FPGA

C

Assembly

CPU

C

High-Level Synthesis

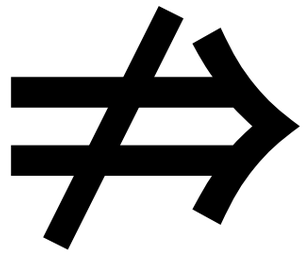
RTL

register-transfer level

image and video processing, financial analytics, bioinformatics, and scientific computing applications. Since RTL programming in VHDL or Verilog is unacceptable to most application software developers, it is essential to provide a highly automated compilation/synthesis flow from C/C++ to FPGAs.

As a result, a growing number of FPGA designs are

Verilog is unacceptable



we must program FPGAs in C

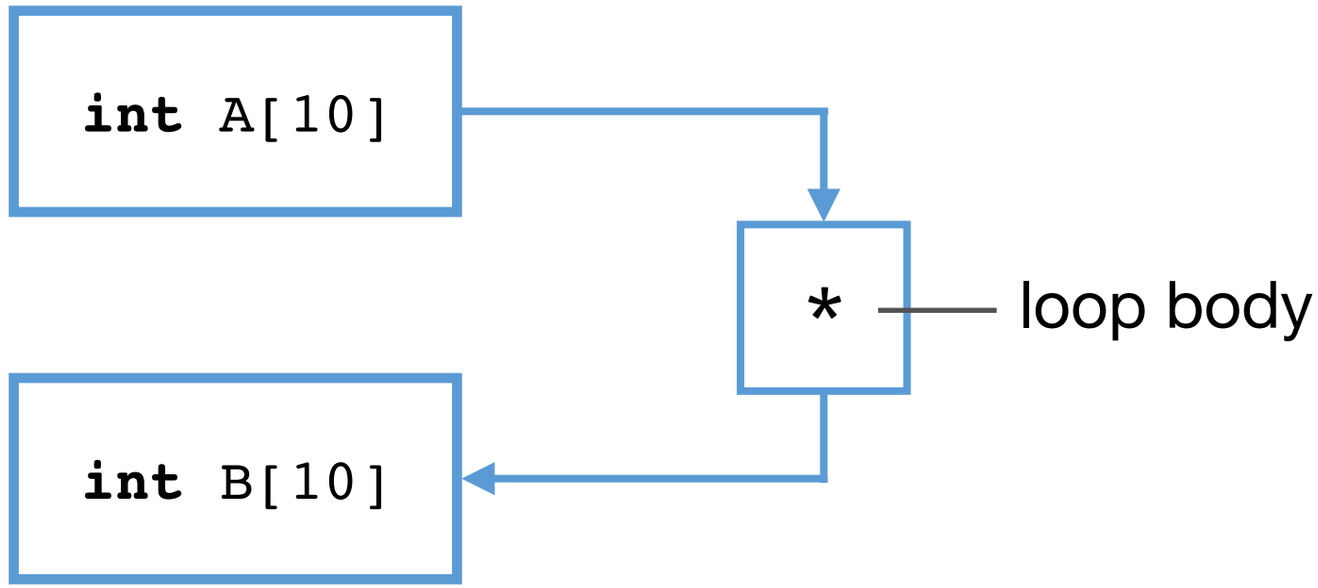
HLS

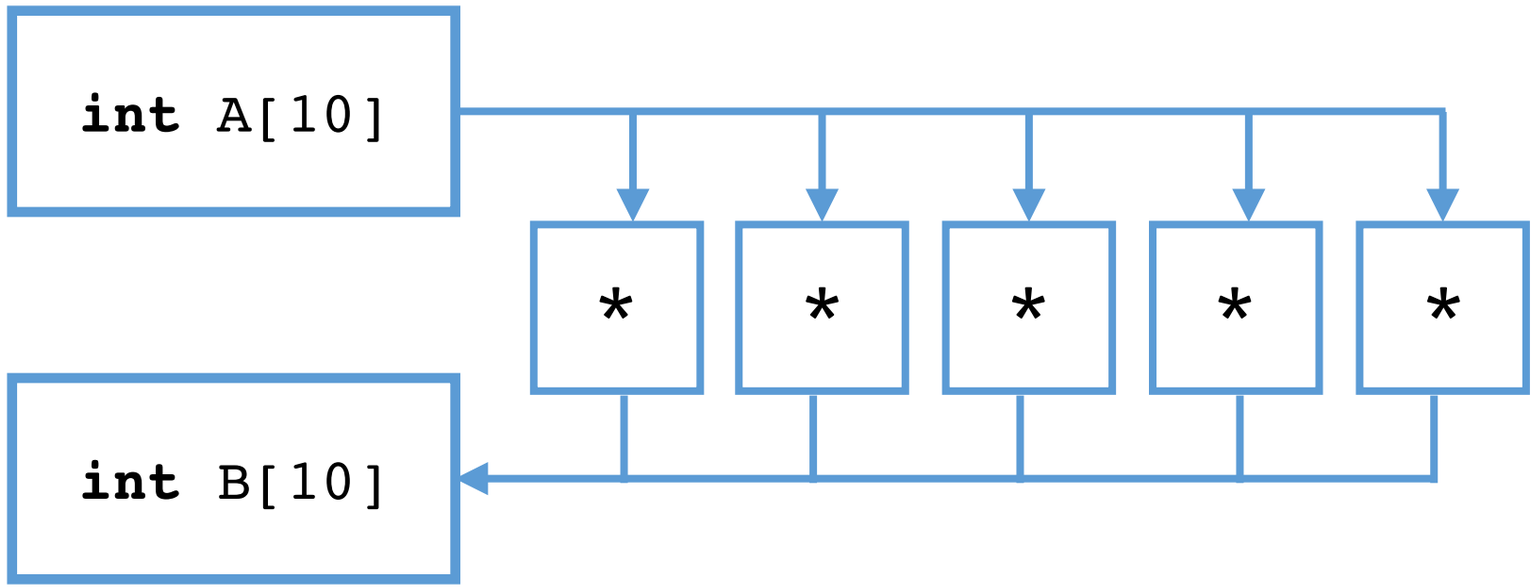
An enormous series of *ad hoc* consistency checks, hacks, and workarounds to compile some C programs to Verilog.

Seashell

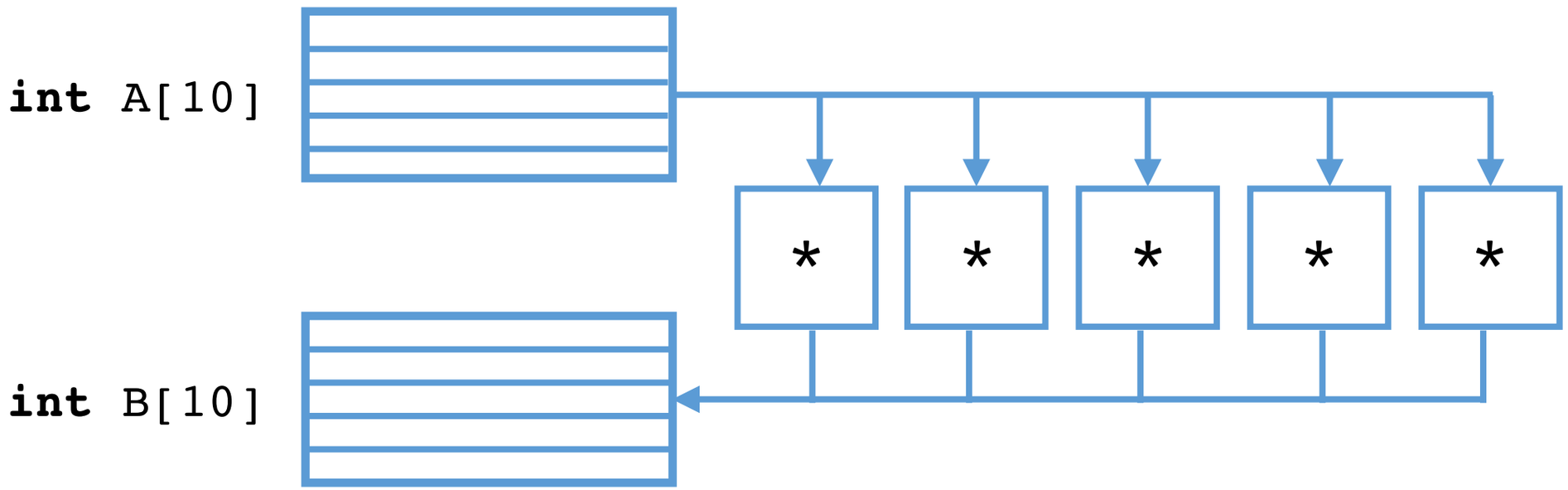
A new language for hardware accelerator design with a **type system** that defines which programs are realizable on FPGAs.

```
int A[10];
int B[10];
for (int i = 0; i < 10; i++) {
    int x = A[i];
    int y = x * 5;
    B[i] = y;
}
```


int A[10]
*
loop body
int B[10]


int A[10]
*
*
*
*
*
int B[10]

```
#pragma HLS ARRAY_PARTITION variable=A factor=5
#pragma HLS ARRAY_PARTITION variable=B factor=5
int A[10];
int B[10];
for (int i = 0; i < 10; i++) {
    #pragma HLS UNROLL factor=5
    int x = A[i];
    int y = x * 5;
    B[i] = y;
}
```


int A[10]
*
*
*
*
*
int B[10]

```
#pragma HLS ARRAY_PARTITION variable=A factor=5
#pragma HLS ARRAY_PARTITION variable=B factor=5
int A[10];
int B[10];
for (int i = 0; i < 10; i++) {
    #pragma HLS UNROLL factor=5
    int x = A[i];
    int y = x * 5;
    B[i] = y;
}
```

Memory types

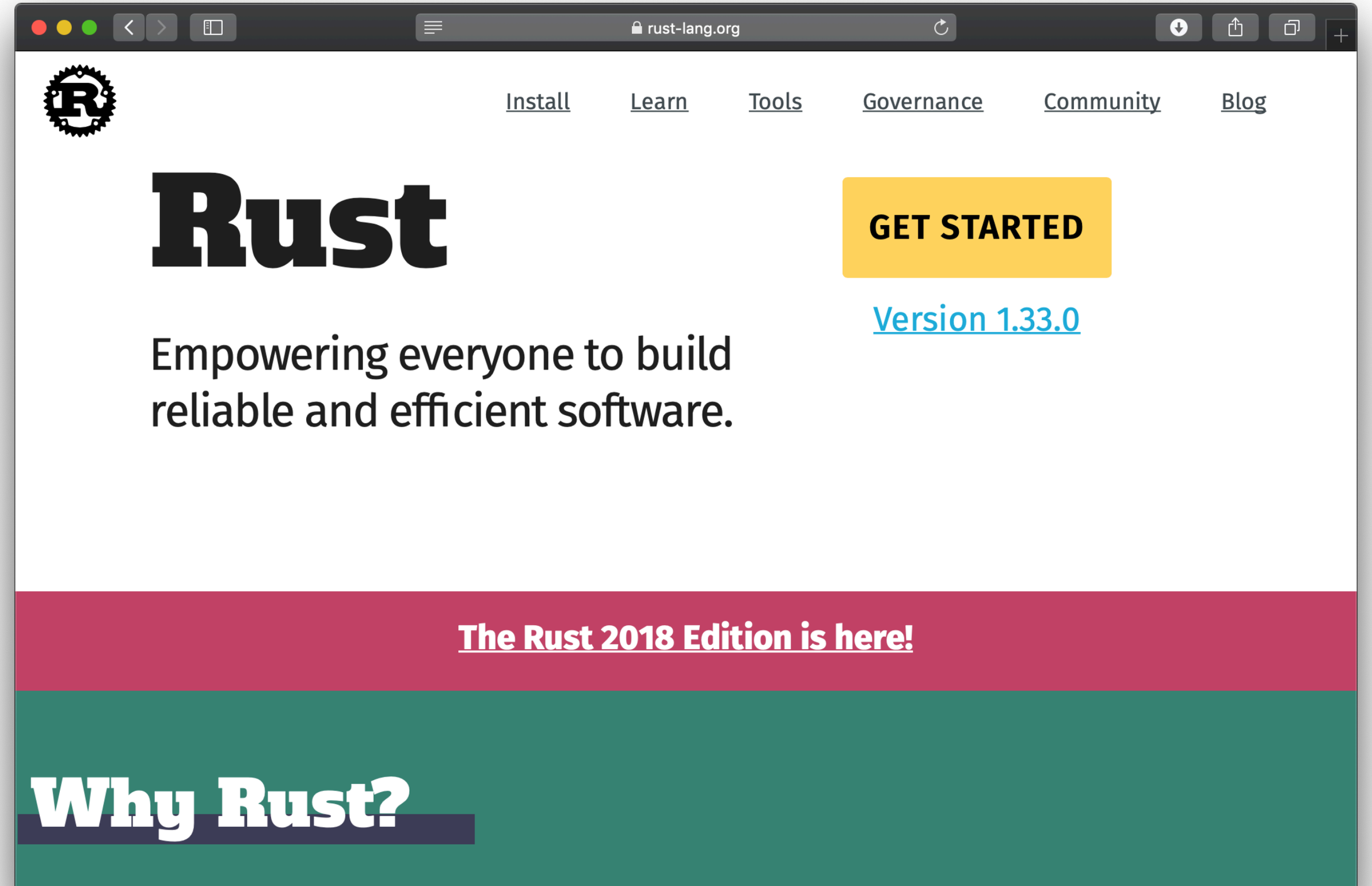
```
memory A : int[10];
for (...) {
    access A[i];
    access A[i+1];
}
```

Affine memory types

```
memory A : int[10];
for (...) {
    access A[i];
    access A[i+1];
}
```

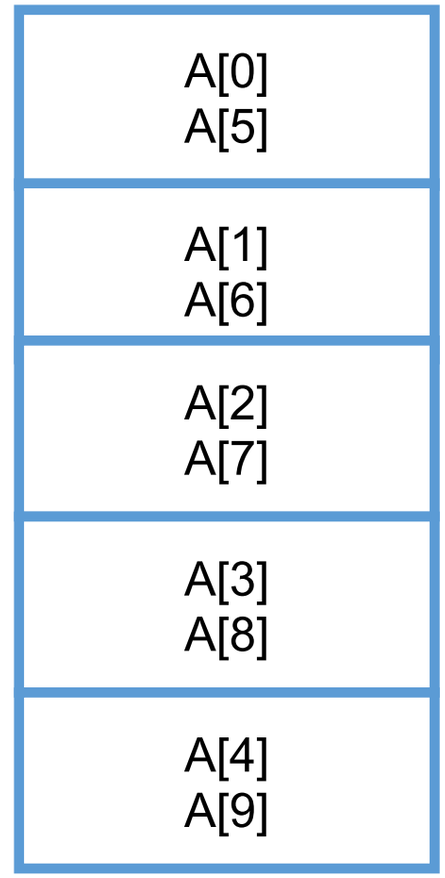
error: A already used in this context

Affine types and **linear types**, as made famous recently by **Rust**.



Banked memory types

```
memory bank(5) A : int[10];
```



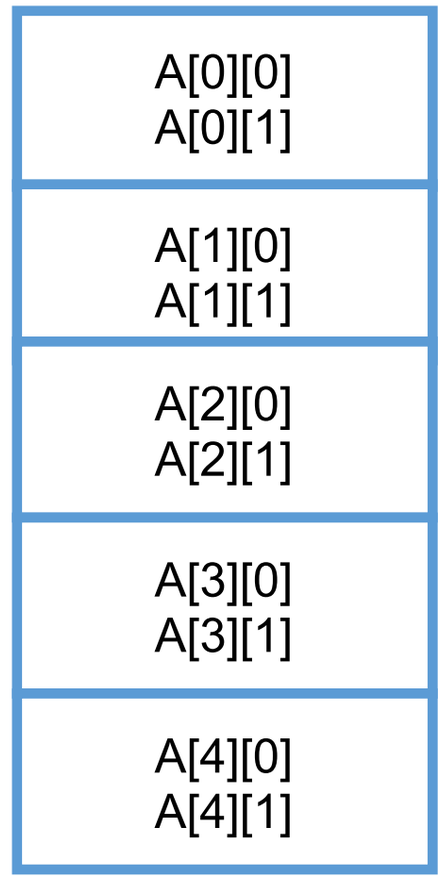
Banked memory types

```
memory bank(5) A : int[10];
for (let i in 0..1) {
    access A[0][i];
    access A[1][i];
    access A[2][i];
    access A[3][i];
    access A[4][i];
}
```

// one access to each A[j] allowed here

STATIC

DYNAMIC



Hybrid indices for unrolling

```
memory bank(5) A : int[10];
for (let i in 0..9) unroll 5 {
    access A[??][??];
}
```

i : idx<0..5, 0..2>

A pair of a **static index** from 0 through 4 and a **dynamic index** that's either 0 or 1.

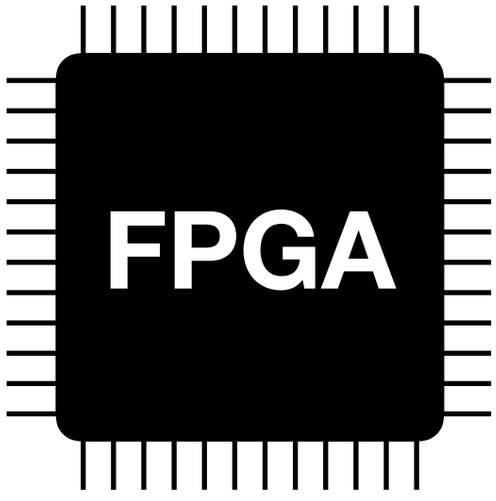
Seashell

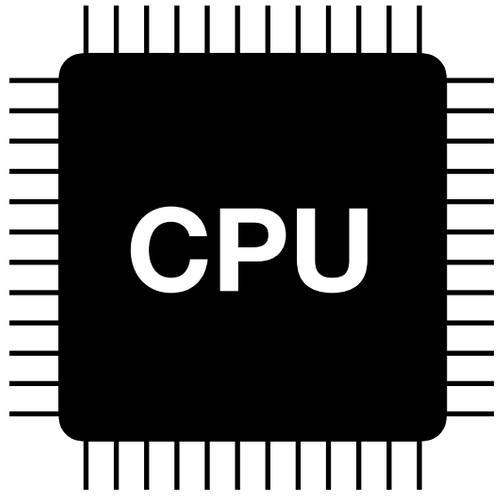
C

RTL

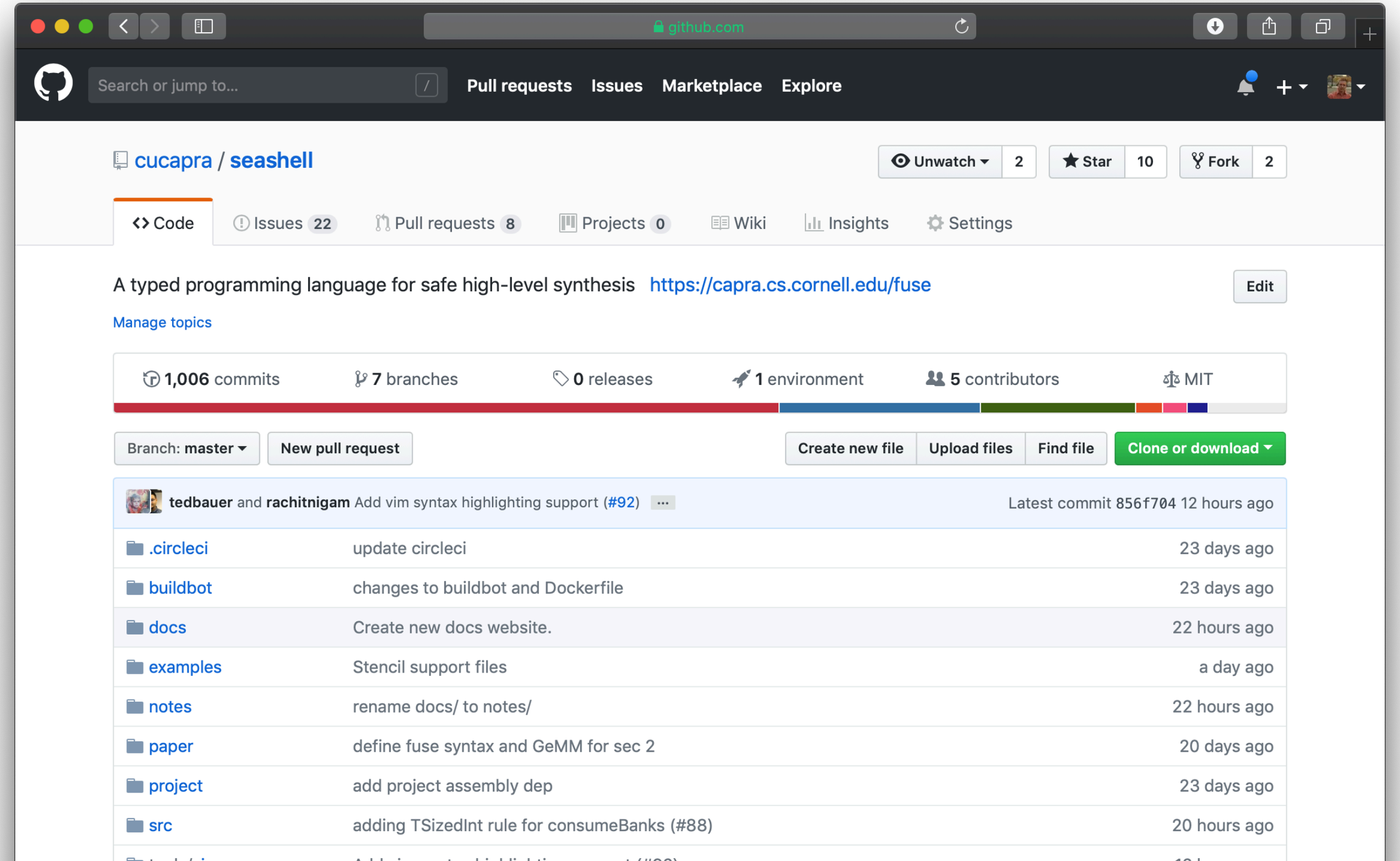
register-transfer level

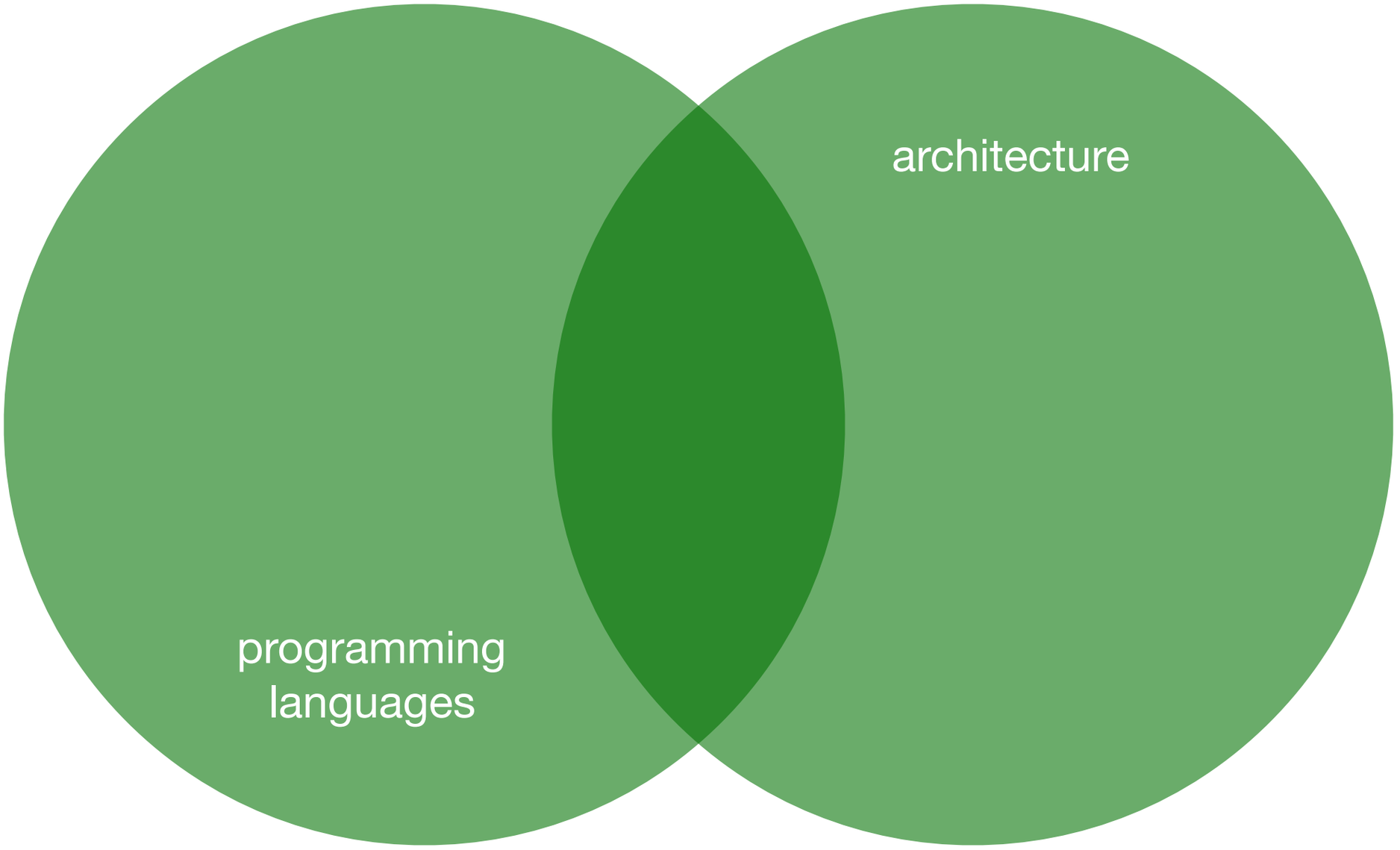
Assembly





github.com/cucapra/seashell




architecture
programming
languages