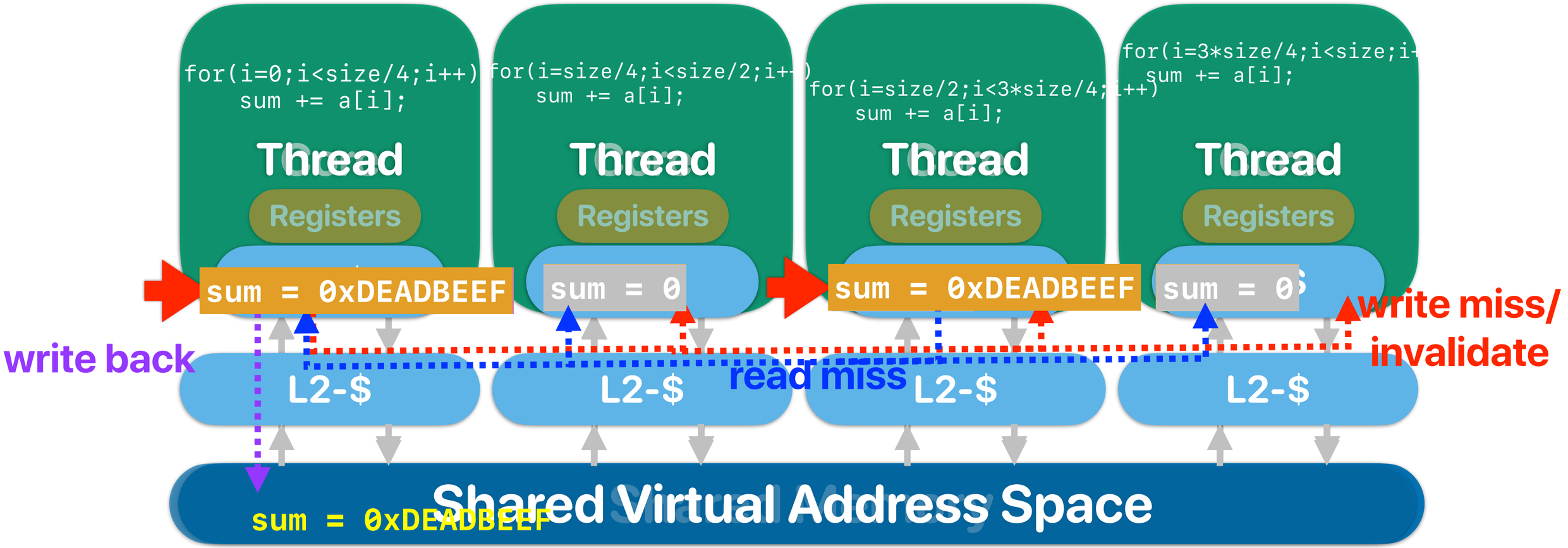
Dusk and dawn: dark silicon and the new golden age of computer architecture (1)

Hung-Wei Tseng

Recap: Modern processors have both CMP/SMT

What happens when we write in coherent caches?



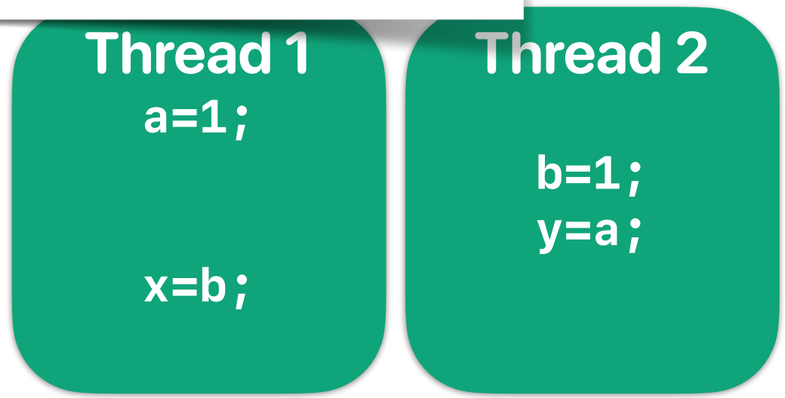
Recap: 4Cs of cache misses

- 3Cs:
  - Compulsory, Conflict, Capacity
- Coherency miss:
  - A “block” invalidated because of the sharing among processors.
- True sharing
  - Processor A modifies X, processor B also want to access X.
- False sharing
  - Processor A modifies X, processor B also want to access Y. However, Y is invalidated because X and Y are in the same block!

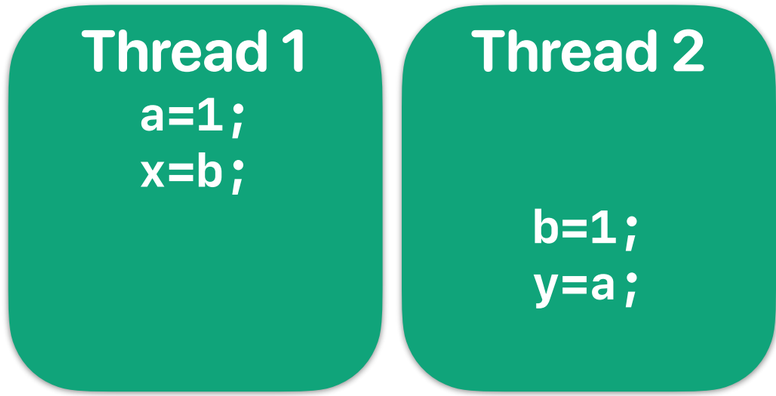
Recap: Possible scenarios

```c
int main() {
  int i;
  pthread_t thread[2];
  pthread_create(&thread[0], NULL, modifya, NULL);
  pthread_create(&thread[1], NULL, modifyb, NULL);
  pthread_join(thread[0], NULL);
  pthread_join(thread[1], NULL);
  fprintf(stderr,"(%d, %d)\n",x,y);
  return 0;
}
```

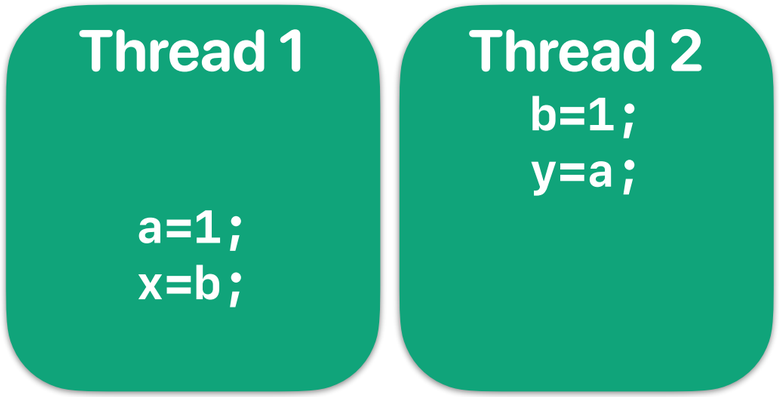
```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>

volatile int a,b;
volatile int x,y;
volatile int f;
void* modifya(void *z) {
  a=1;
  x=b;
  return NULL;
}
void* modifyb(void *z) {
  b=1;
  y=a;
  return NULL;
}
```



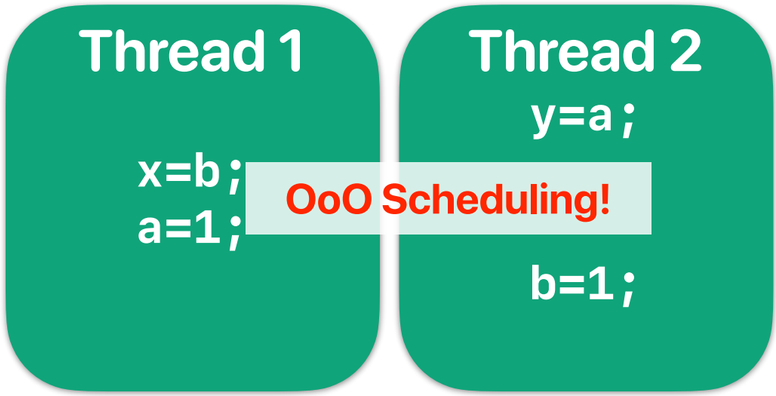
(1,1)



(0,1)



(1,0)



(0,0)

Recap: parallel programming

- Cache coherency only guarantees that everyone would eventually have a coherent view of data, but not when
- Cache coherency may create unexpected cache invalidations/ misses if you do it wrong
- Processor behaviors are non-deterministic
  - You cannot predict which processor is going faster
  - You cannot predict when OS is going to schedule your thread
  - You cannot predict when the processor is going to schedule an instruction
- Cache consistency is hard to support

What kind of processors Google search needs

- If we are designing a processor just for Google search or similar type of applications, how many of the following targets/features would fulfill the demand?
  - ① Can execute many instructions from the same process/thread simultaneously
  - ✓ ② Can execute many processes/threads simultaneously
  - ✓ ③ Can predict branch outcome accurately
  - ④ Have very large cache capacity

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

given how little ILP our application yields, and shorter pipelines would reduce or eliminate branch mispredict penalties. The avail-

For such workloads, a memory system with a relatively modest sized L2 cache, short L2 cache and memory latencies, and longer (perhaps 128 byte) cache lines is likely to be the most effective.

prediction logic. In essence, there isn't that much exploitable instruction-level parallelism (ILP) in the workload. Our measurements suggest that the level of aggressive out-of-order, speculative execution present in modern processors is already beyond the point of diminishing performance returns for such programs.

end ones. Exploiting such abundant thread-level parallelism at the microarchitecture level appears equally promising. Both simultaneous multithreading (SMT) and chip multiprocessor (CMP) architectures target thread-level parallelism and should improve the performance of many of our servers. Some early

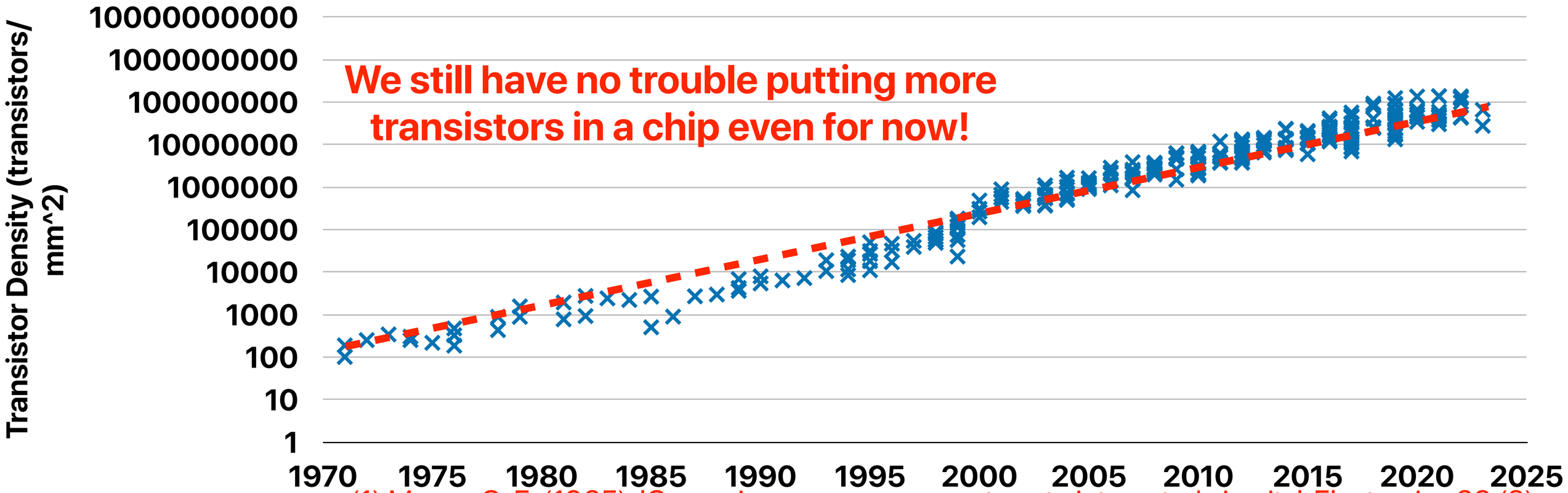
Outline

- The dark silicon problem
- Challenges and state-of-the-art solutions in the dark silicon era
- The new golden age of Computer Architecture

What is and why is dark silicon?

Moore's Law[1]

- The number of transistors we can build in a fixed area of silicon doubles every 12 ~ 24 months.
- Moore's Law "was" the most important driver for historic CPU performance gains



(1) Moore, G. E. (1965), 'Cramming more components onto integrated circuits', Electronics 38 (8) .

Plot based on https://en.wikipedia.org/wiki/Transistor_count by Hung-Wei Tseng

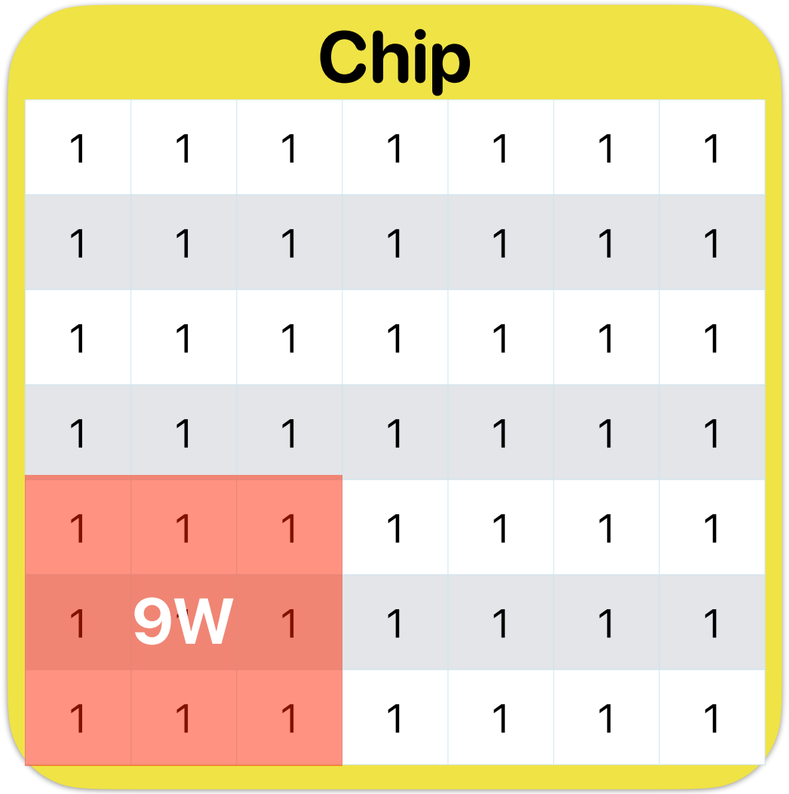
https://www.pollev.com/hungweitseng close in 1:30

What happens if power doesn't scale with process technologies?

- If we are able to cram more transistors within the same chip area (Moore's law continues), but the power consumption per transistor remains the same. Right now, if put more transistors in the same area because the technology allows us to. How many of the following statements are true?
  - ① The power consumption per chip will increase
  - ② The power density of the chip will increase
  - ③ Given the same power budget, we may not able to power on all chip area if we maintain the same clock rate
  - ④ Given the same power budget, we may have to lower the clock rate of circuits to power on all chip area

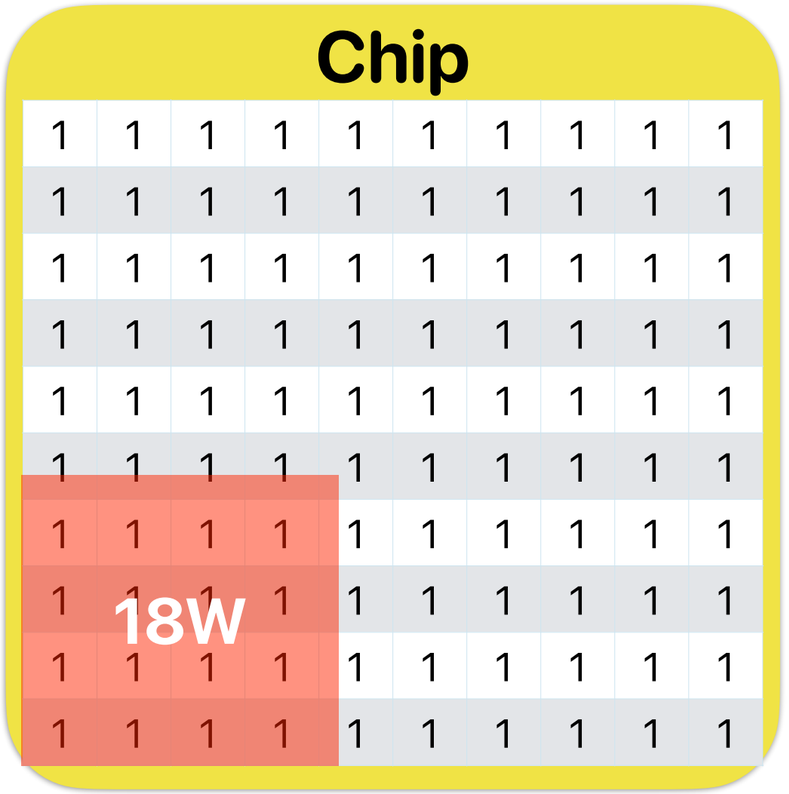
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Power consumption to light on all transistors



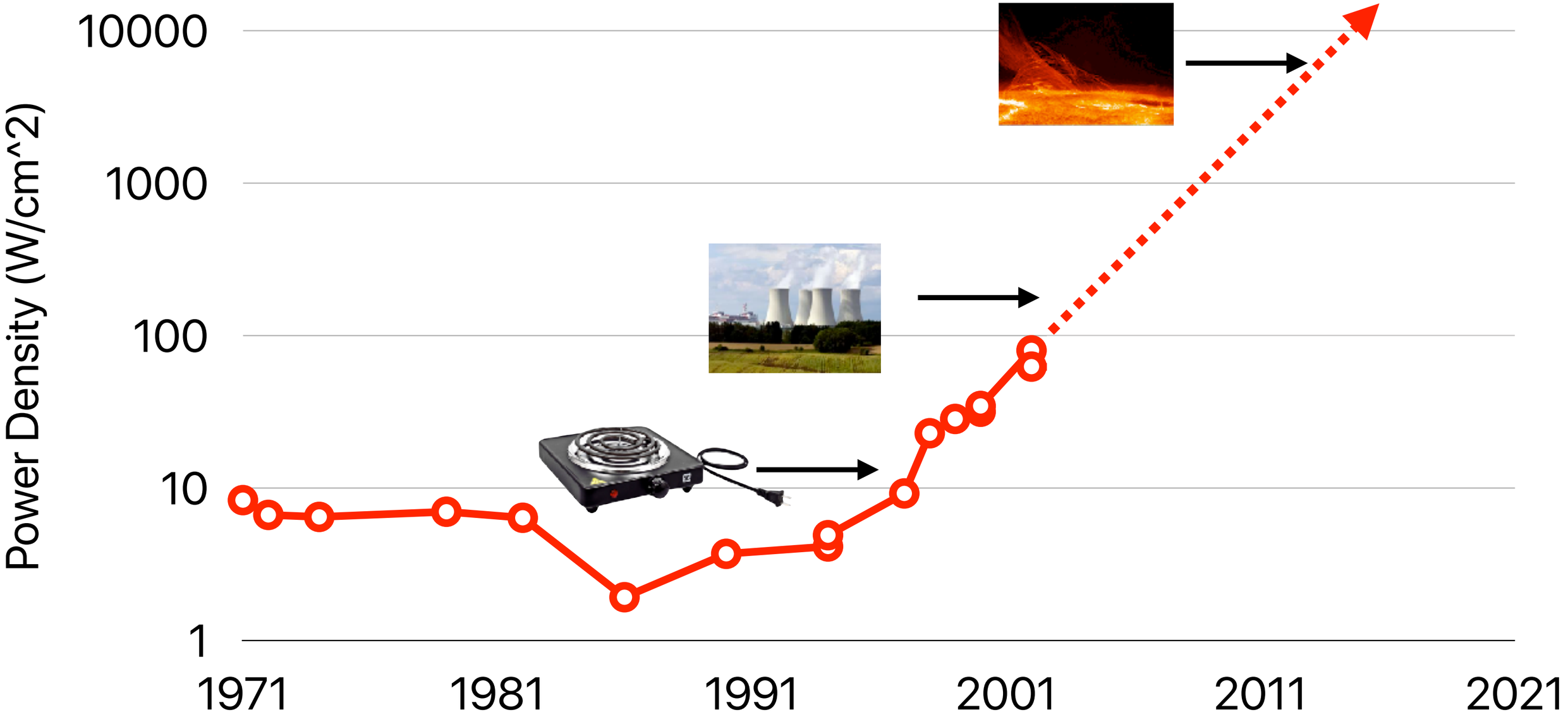
=49W

Dennardian Broken

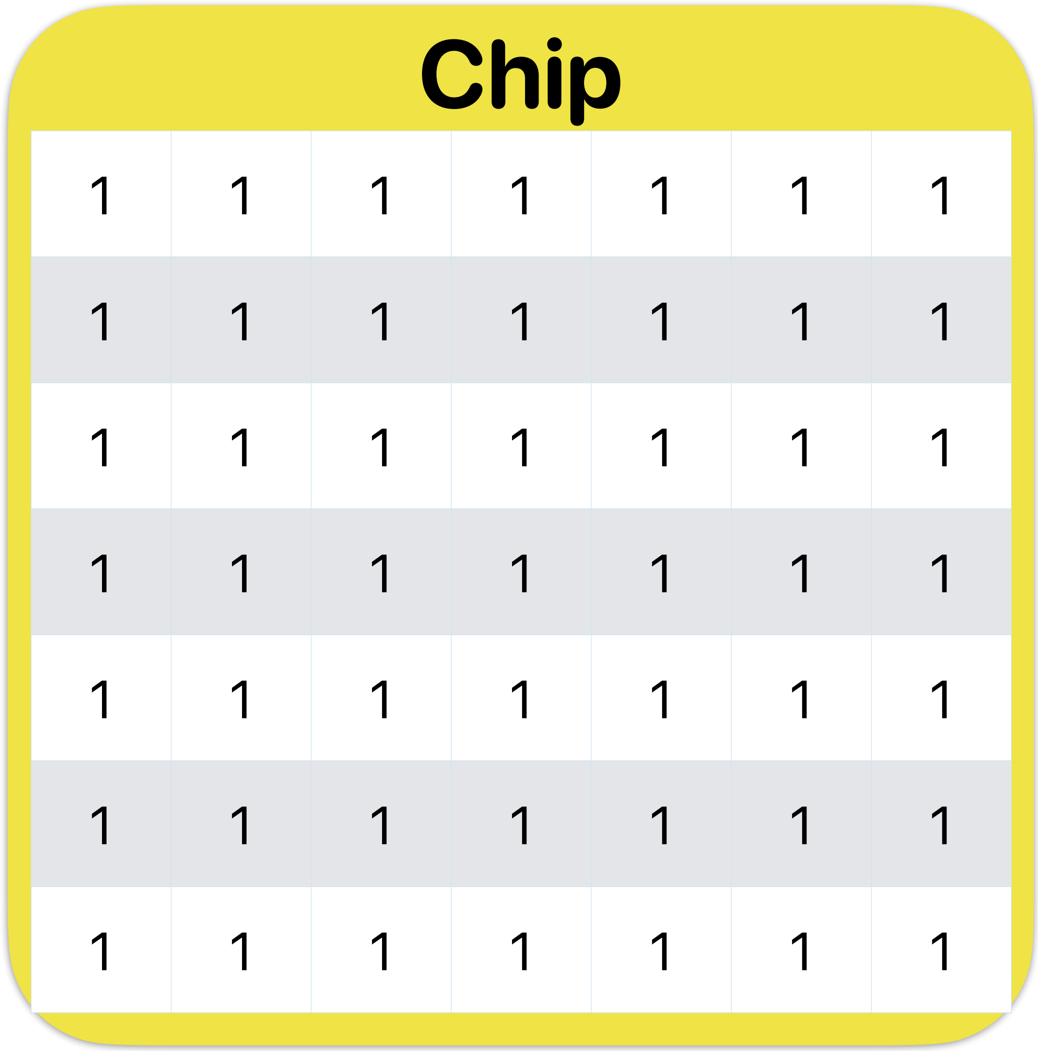


=100W!

Power Density of Processors

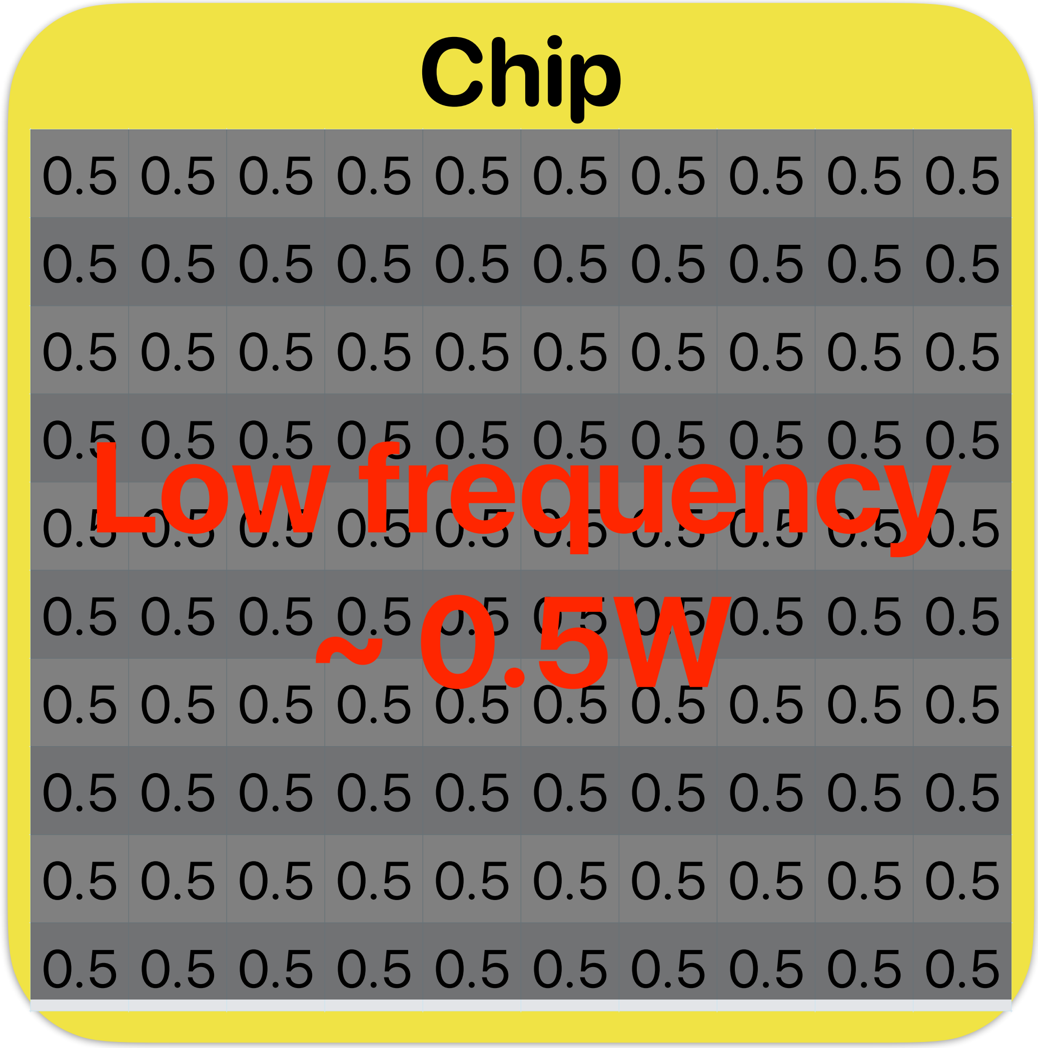


If we have a fixed power budget



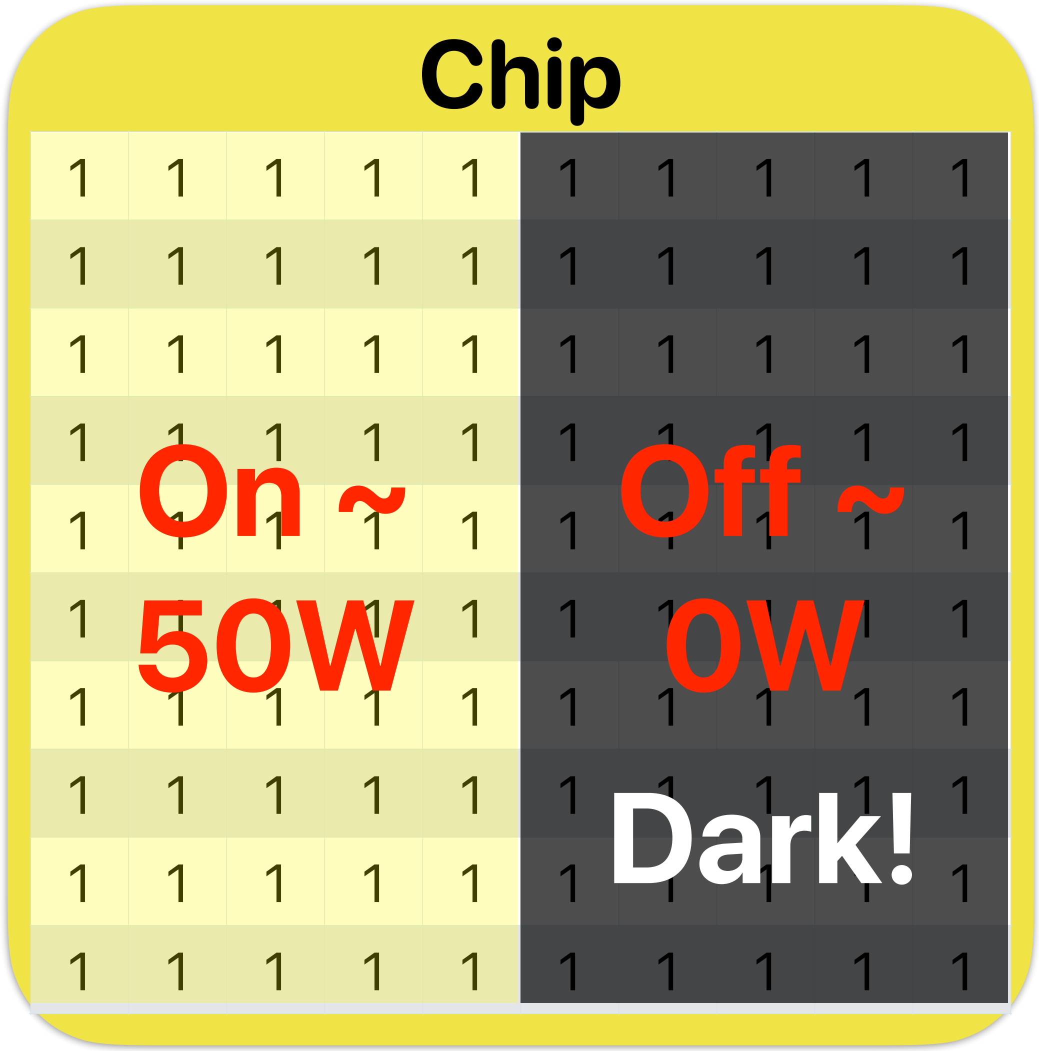
=49W

Slowdown all of them



=50W!

Dark silicon!



=50W!

Power consumption & power density

- $P_{dynamic} \sim \alpha \times C \times V^2 \times f \times N$
  - α: average switches per cycle
  - C: capacitance
  - V: voltage
  - f: frequency, usually linear with V
  - N: the number of transistors
- $P_{leakage} \sim N \times V \times e^{-V}$
  - N: number of transistors
  - V: voltage
  - V_t: threshold voltage where transistor conducts (begins to switch)
- Power density:

$$P_{density} = \frac{P}{area}$$

Dennard scaling discontinued — we cannot make voltage lower

Moore's Law allows higher frequencies as transistors are smaller

Moore's Law makes this smaller

What happens if power doesn't scale with process technologies?

- If we are able to cram more transistors within the same chip area (Moore's law continues), but the power consumption per transistor remains the same. Right now, if put more transistors in the same area because the technology allows us to. How many of the following statements are true?
  - ① The power consumption per chip will increase
  - ② The power density of the chip will increase
  - ③ Given the same power budget, we may not able to power on all chip area if we maintain the same clock rate
  - ④ Given the same power budget, we may have to lower the clock rate of circuits to power on all chip area

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

What happens if power doesn't scale with process technologies?

- If we are able to cram more transistors within the same chip area (Moore's law continues), but the power consumption per transistor remains the same. Right now, if put more transistors in the same area because the technology allows us to. How many of the following statements are true?
  - ① The power consumption per chip will increase
  - ② The power density of the chip will increase
  - ③ Given the same power budget, we may not able to power on all chip area if we maintain the same clock rate — **even if we put more cores, we cannot have all of them on!**
  - ④ Given the same power budget, we may have to lower the clock rate of circuits to power on all chip area — **or we have to operate all of them at a slower speed**

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

as $P_{dynamic} \sim \alpha \times C \times V^2 \times f \times N$

Recap — Take-aways: parallel programming

- Cache coherency only guarantees that everyone would eventually have a coherent view of data, but not when
- Cache coherency may create unexpected cache invalidations/misses if you do it wrong
- Processor behaviors are non-deterministic
  - You cannot predict which processor is going faster
  - You cannot predict when OS is going to schedule your thread
  - You cannot predict when the processor is going to schedule an instruction
- Cache consistency is hard to support
- **Even if we can address all programming challenges, multi-core performance has stopped to scale due to the Dark Silicon problem**

Dark silicon problem

Challenges and state-of-the-art solutions in the dark silicon era

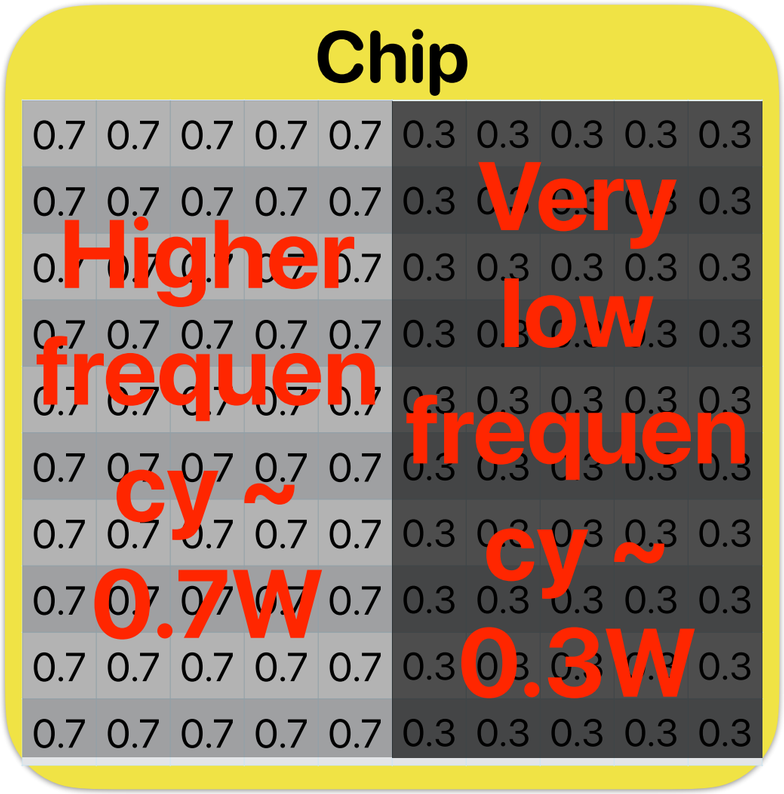
Dark silicon problem

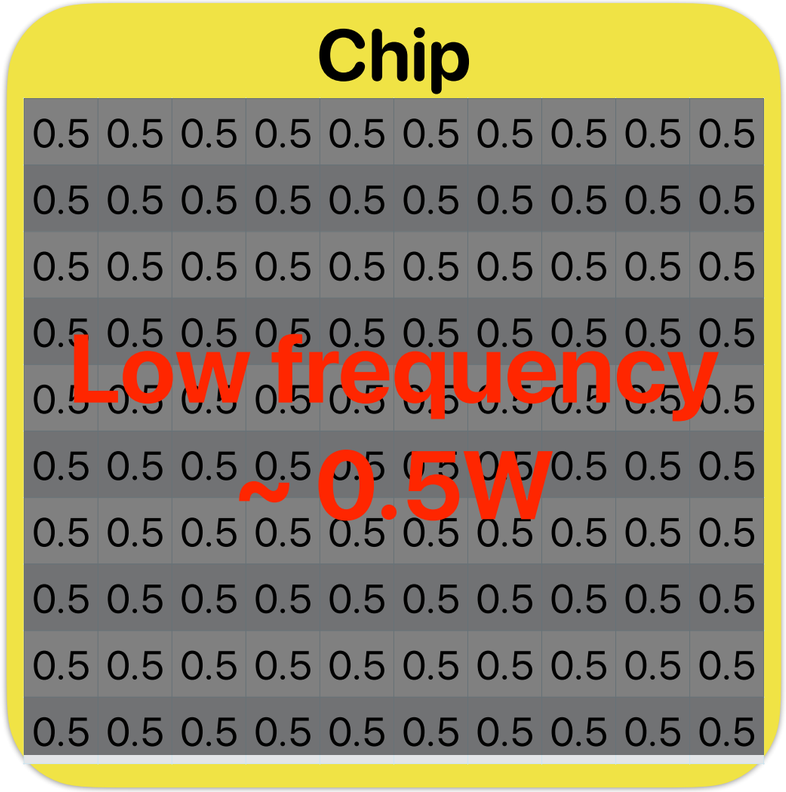
- We are given the same area, twice amount of transistors
- We are given the same power budget
- We are given a certain applications
- How can we maximize the performance for these applications within the given constraints?

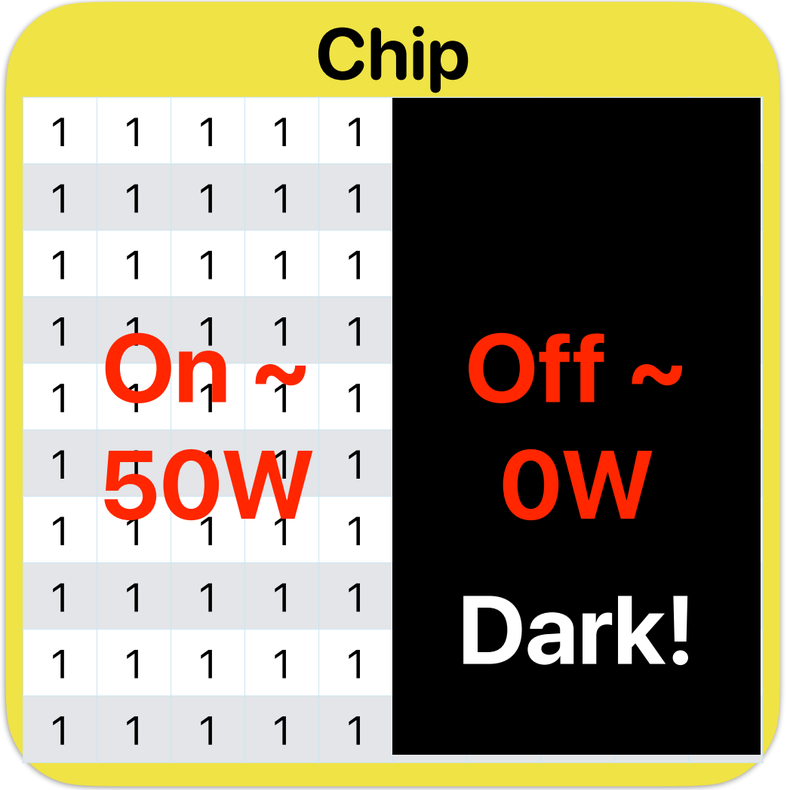
Given the same power budget, maximize the efficiency per chip

Some faster,
some slower

Some at top speed,
Slowdown all of themsome are not functioning







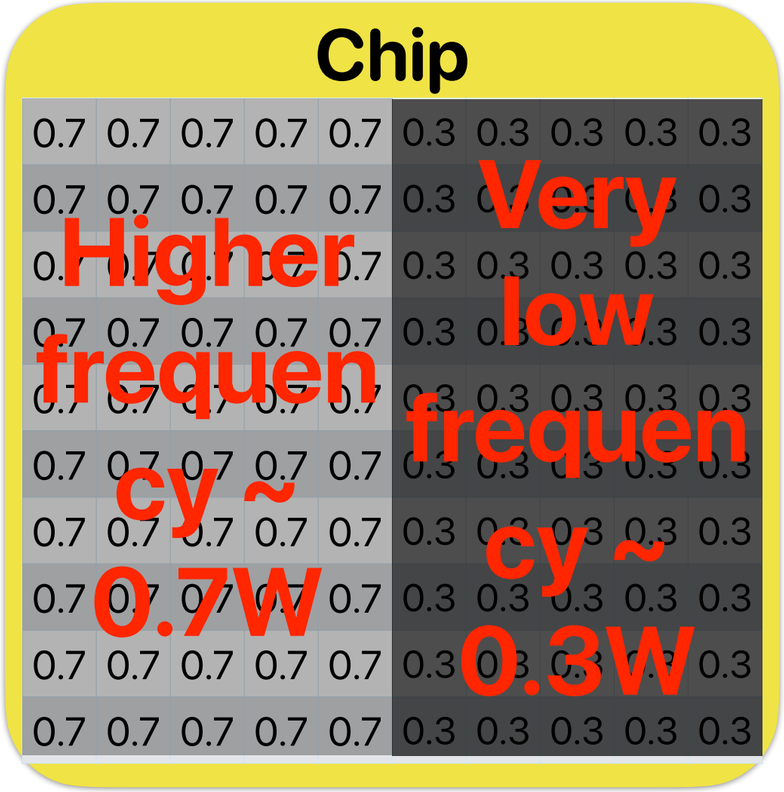
=50W!

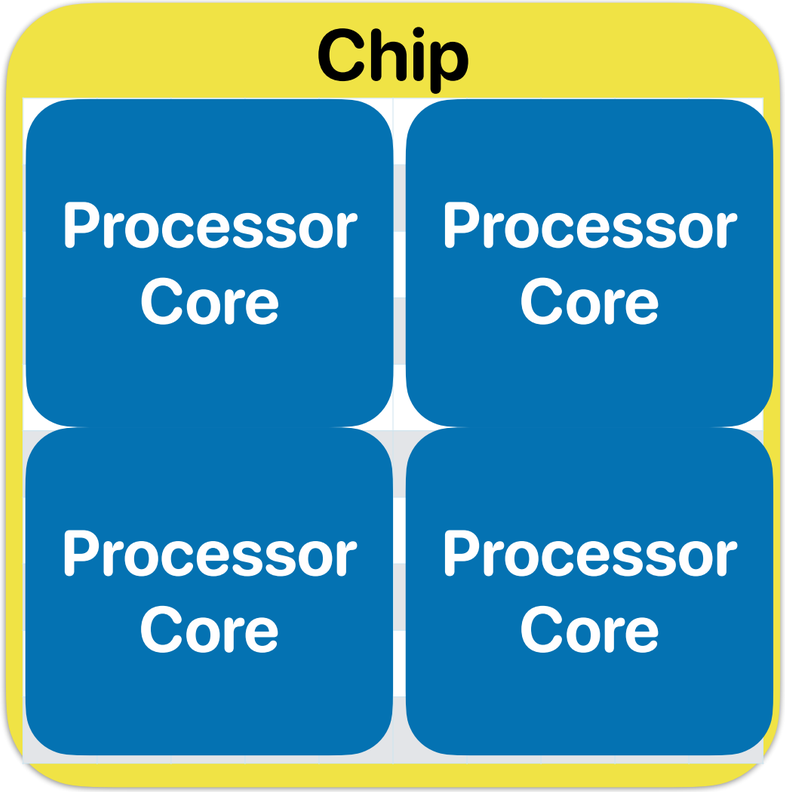
=50W!

=50W!

Given the same power budget, maximize the efficiency per chip

Some faster, some slower





Aggressively adjust the frequency of processor cores — If only one program is compute-intensive, having it running at full speed, others at lower frequency or turning off

Modern processor's frequency

```
NUMA node(s):                    1
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           151
Model name:                      12th Gen Intel(R) Core(TM) i7-12700KF
Stepping:                        2
CPU MHz:                         1226.409
CPU max MHz:                     5000.0000
CPU min MHz:                     800.0000
BogoMIPS:                        7219.20
```

	i7-12700K
	Intel 7
	$450.00 - $460.00
	PC/Client/Tablet, Workstation

CPU Specifications

Total Cores	12
# of Performance-cores	8
# of Efficient-cores	4
Total Threads	20
Max Turbo Frequency	5.00 GHz
Intel® Turbo Boost Max Technology 3.0 Frequency ‡	5.00 GHz
Performance-core Max Turbo Frequency	4.90 GHz
Efficient-core Max Turbo Frequency	3.80 GHz
Performance-core Base Frequency	3.60 GHz
Efficient-core Base Frequency	2.70 GHz
Cache	25 MB Intel® Smart Cache

https://www.intel.com/content/www/us/en/products/sku/134594/intel-core-i712700k-processor-25m-cache-up-to-5-00-ghz/specifications.html

Modern processor's frequency

```
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   48 bits physical, 48 bits virtual
CPU(s):                          12
On-line CPU(s) list:             0-11
Thread(s) per core:              2
Core(s) per socket:              6
Socket(s):                       1
NUMA node(s):                    1
Vendor ID:                       AuthenticAMD
CPU family:                      25
Model:                           80
Model name:                      AMD Ryzen 5 5500
Stepping:                        0
Frequency boost:                 enabled
CPU MHz:                         3600.000
CPU max MHz:                     3600.0000
CPU min MHz:                     1400.0000
```

Take-aways: Challenges and SOTA solutions in the dark silicon era

- **Even if we can address all programming challenges, multi-core performance has stopped to scale due to the Dark Silicon problem**
- Aggressive dynamic frequency/voltage scaling on CMP to accommodate the demand of **latency**-sensitive, parallelism-limited applications, but the area-efficiency of the slower cores is not great

More cores per chip, slower per core

Export comparison	Intel® Xeon® Platinum 849...	Intel® Xeon® Platinum 846...	Intel® Xeon® Gold 6448H ...	Intel® Xeon® Platinum 844...	Intel® Xeon® Gold 6434H ...
Total Cores	60	48	32	16	8
Total Threads	120	96	64	32	16
Max Turbo Frequency	3.50 GHz	3.80 GHz	4.10 GHz	4.00 GHz	4.10 GHz
Processor Base Frequency	1.90 GHz	2.10 GHz	2.40 GHz	2.90 GHz	3.70 GHz
Cache	112.5 MB	105 MB	60 MB	45 MB	22.5 MB
Intel® UPI Speed	16 GT/s	16 GT/s	16 GT/s	16 GT/s	16 GT/s
Max # of UPI Links	4	4	3	4	3
TDP	350 W	330 W	250 W	270 W	195 W

https://ark.intel.com/content/www/us/en/ark/products/series/228622/4th-generation-intel-xeon-scalable-processors.html

Xeon Phi

Essentials

Product Collection	Intel® Xeon Phi™ 72x5 Processor Family
Code Name	Products formerly Knights Mill
Vertical Segment	Server
Processor Number	7295
Off Roadmap	No
Status	Launched
Launch Date	Q4'17
Lithography	14 nm

Performance

# of Cores	72
# of Threads	72
Processor Base Frequency	1.50 GHz
Max Turbo Frequency	1.50 GHz
Cache	36 MB L2 Cache
TDP	320 W

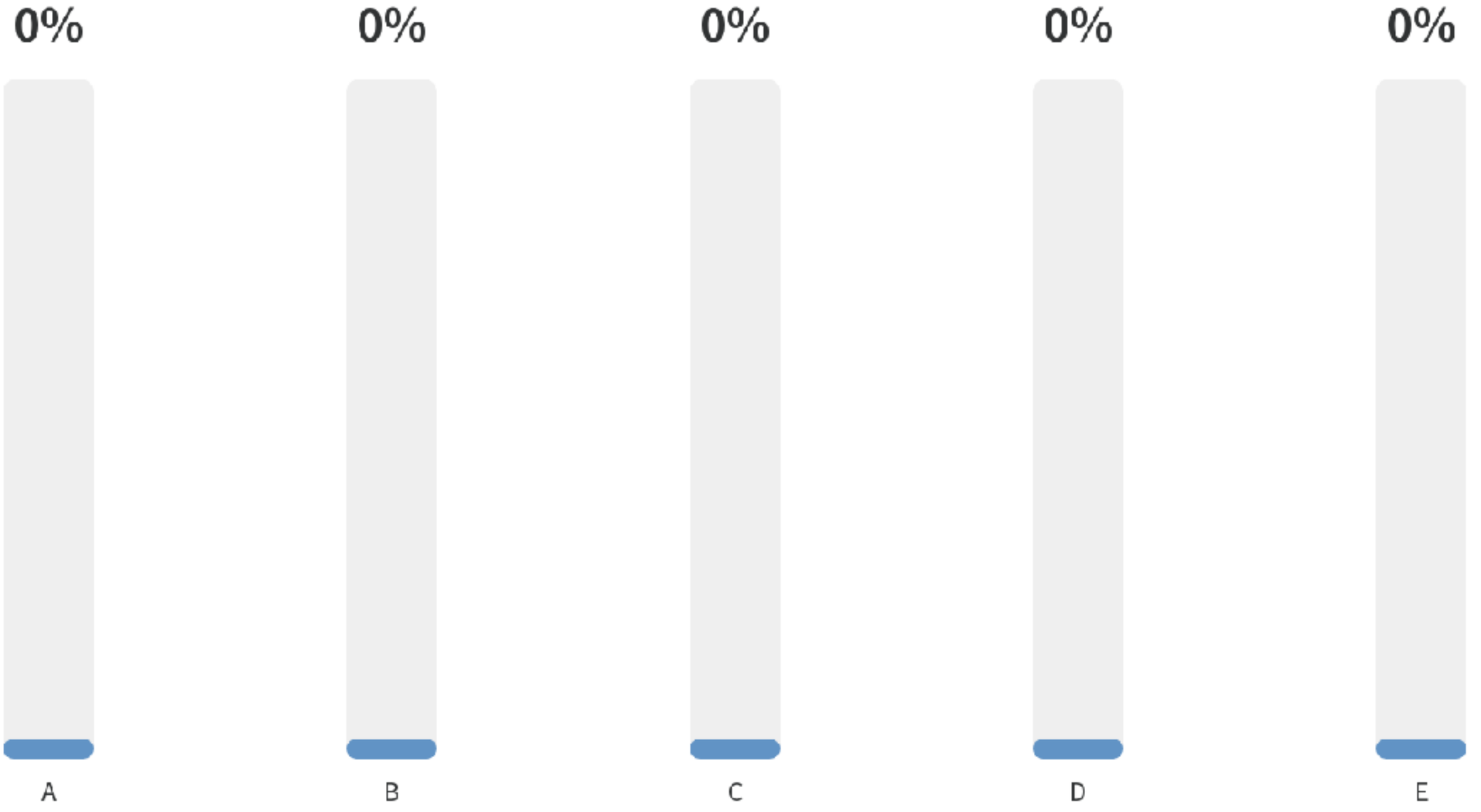
https://www.pollev.com/hungweitseng close in 1:30

Lower frequency

- Regarding lower the operating frequency when running applications, how many of the following statements are correct?
  - ① Lowering the frequency helps the same battery capacity to carry out more tasks
  - ② Lowering the frequency helps reducing the heat generation
  - ③ Lowering the frequency helps reducing the electricity bill/total energy consumption of running the same amount of tasks
  - ④ A CPU operating at 30% of the peak frequency can still consume more than 50% of the peak power

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

0



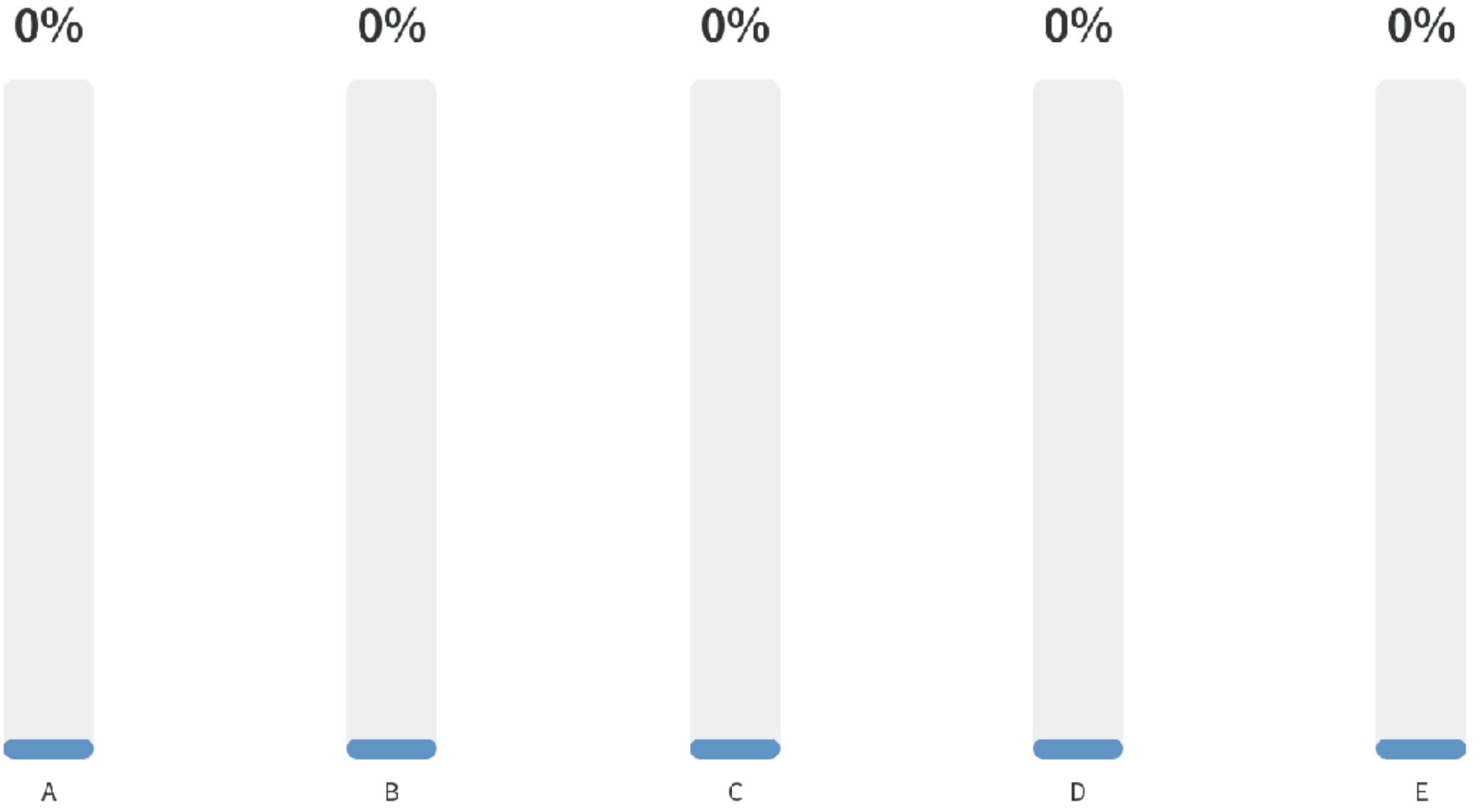
https://www.pollev.com/hungweitseng close in 1:30

Lower frequency

- Regarding lower the operating frequency when running applications, how many of the following statements are correct?
  - ① Lowering the frequency helps the same battery capacity to carry out more tasks
  - ② Lowering the frequency helps reducing the heat generation
  - ③ Lowering the frequency helps reducing the electricity bill/total energy consumption of running the same amount of tasks
  - ④ A CPU operating at 30% of the peak frequency can still consume more than 50% of the peak power

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

0

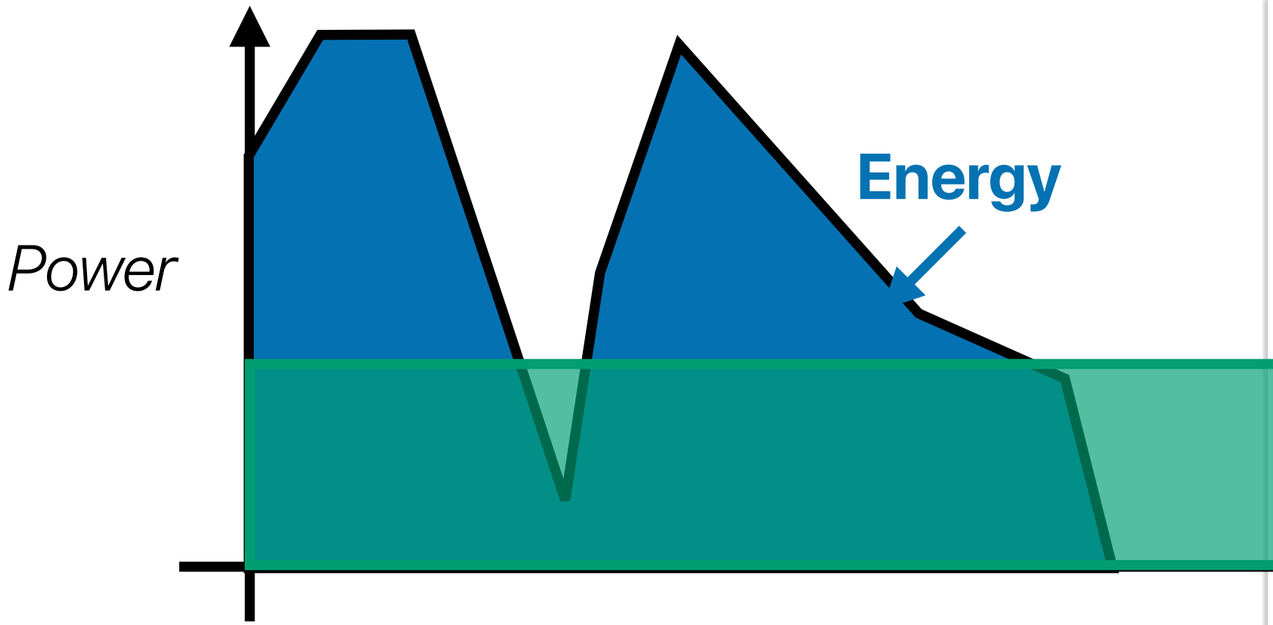


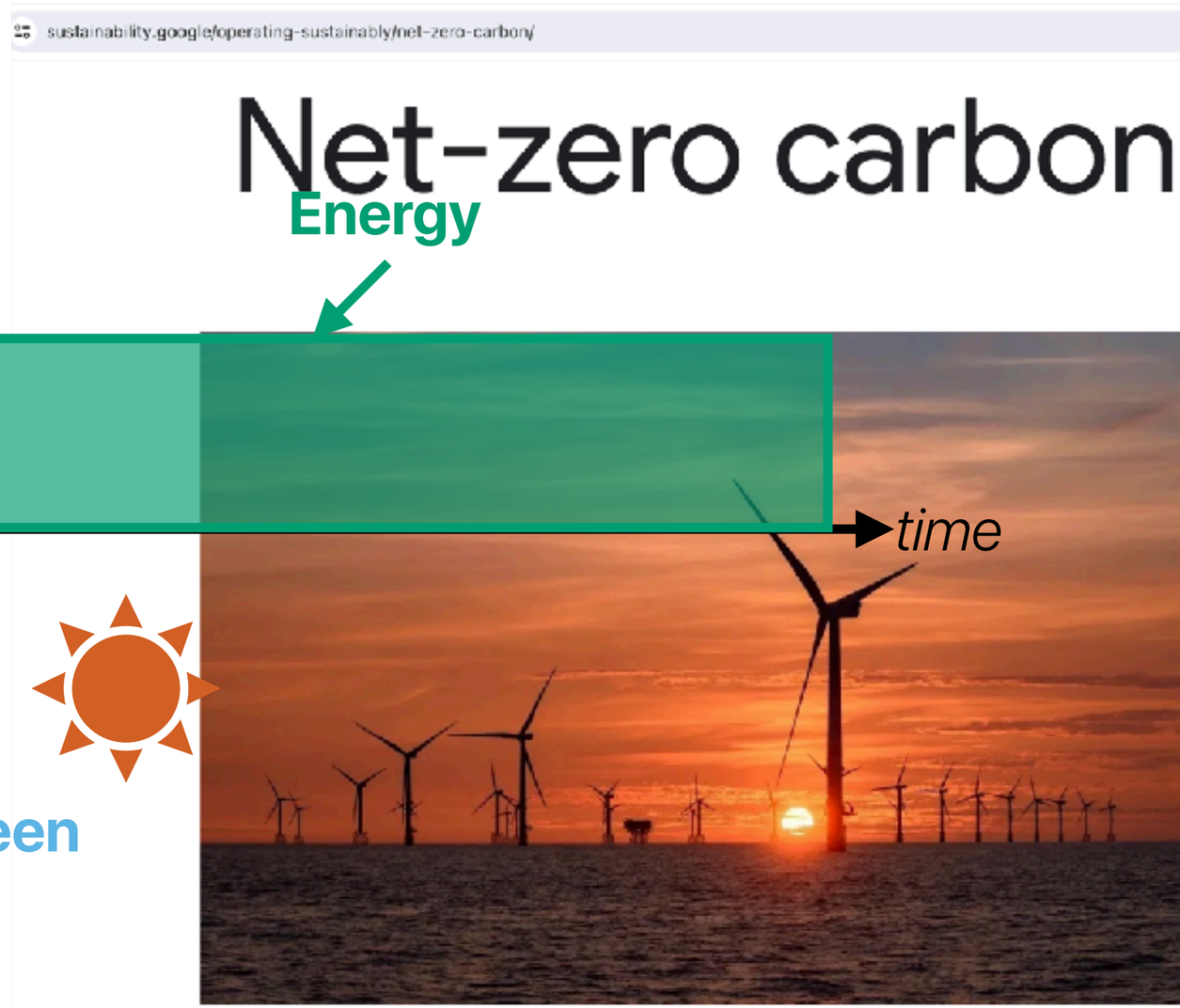
Power v.s. Energy

- Power is the direct contributor of “heat”
  - Packaging of the chip
  - Heat dissipation cost
  - Dynamic power
  - Leakage power
- Energy = $Power \times Execution\_Time$
  - The electricity bill and battery life is related to energy!
  - Lower power does not necessary means better battery life if the processor slow down the application too much

Power/Energy/Carbon footprint

The Green can be more if power is not low enough





If we run the task when there is no green energy— more carbon footprint!

Recap: Demo — changing the max frequency and performance

- Change the maximum frequency of the intel processor — you learned how to do this when we discuss programmer's impact on performance
- LIKWID a profiling tool providing power/energy information
  - likwid-perfctr -g ENERGY [command_line]
  - Let's try blockmm and popcount and see what's happening!

Power consumption & power density

- $P_{dynamic} \sim \alpha \times C \times V^2 \times f \times N$
  - α: average switches per cycle
  - C: capacitance
  - V: voltage
  - f: frequency, usually linear with V
  - N: the number of transistors
- $P_{leakage} \sim N \times V \times e^{-V_t}$
  - N: number of transistors
  - V: voltage
  - V_t: threshold voltage where transistor conducts (begins to switch)
- Power density:

$$P_{density} = \frac{P}{area}$$

Voltage is a key factor of power consumption

Frequency is depending on the supply voltage

Power consumption & power density

- $P_{dynamic} \sim \alpha \times C \times V^2 \times f \times N$
  - α: average switches per cycle
  - C: capacitance
  - V: voltage
  - f: frequency, usually linear with V
  - N: the number of transistors
- $P_{leakage} \sim N \times V \times e^{-V_t}$
  - N: number of transistors
  - V: voltage
  - V_t: threshold voltage where transistor conducts (begins to switch)
- Power density:

$$P_{density} = \frac{P}{area}$$

Lower the frequency can only reduce active power

Moore's Law allows higher frequencies as transistors are smaller

Moore's Law makes this smaller

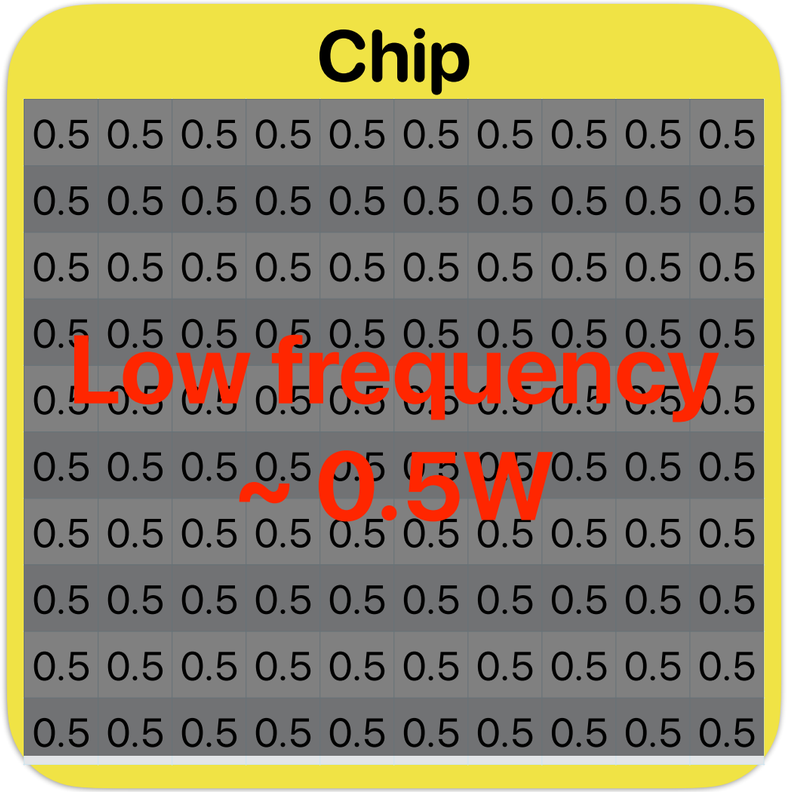
Lower frequency

- Regarding lower the operating frequency when running applications, how many of the following statements are correct?
  - ① Lowering the frequency helps the same battery capacity to carry out more tasks
  - ② Lowering the frequency helps reducing the heat generation
  - ③ Lowering the frequency helps reducing the electricity bill/total energy consumption of running the same amount of tasks
  - ④ A CPU operating at 30% of the peak frequency can still consume more than 50% of the peak power

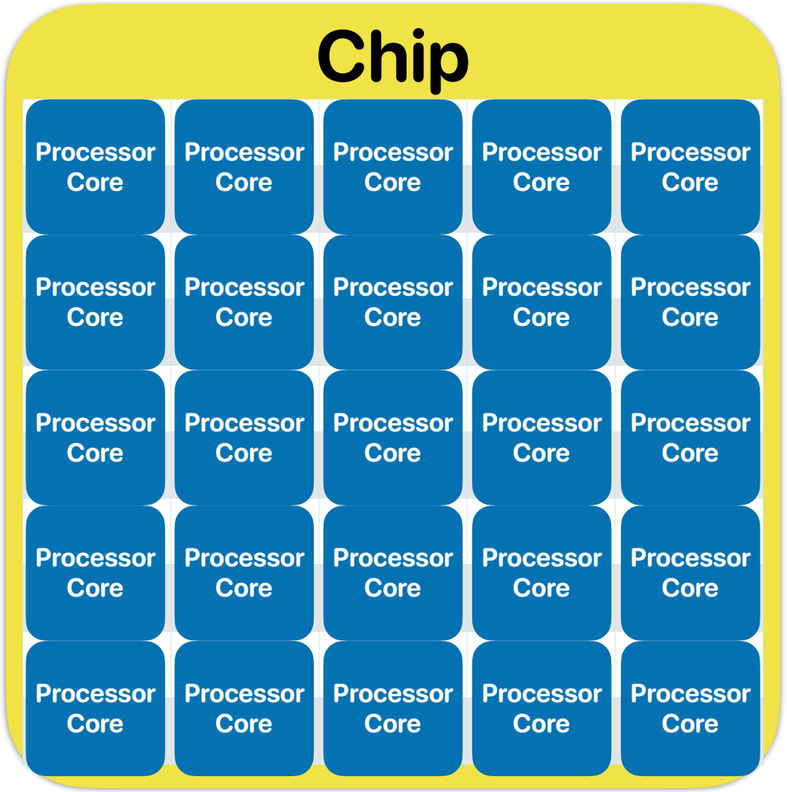
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Given the same power budget, maximize the efficiency per chip

Slowdown all of them



=50W!



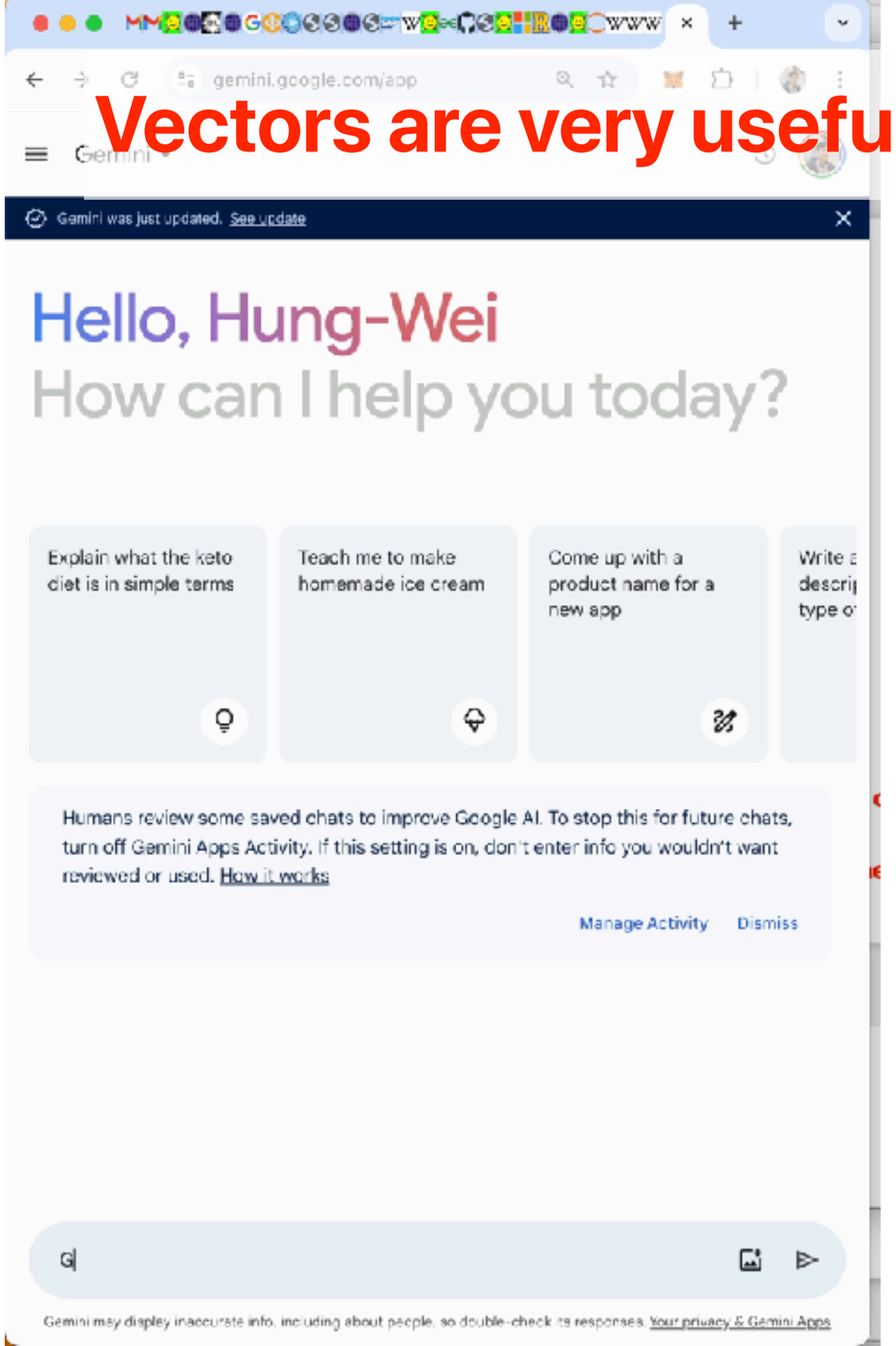
All of them are smaller, simpler, lower-frequency, lower-power cores

New type of parallelism

Parallelism in modern computers

- Instruction-level parallelism — perform various, independent instructions simultaneously
  - Pipeline
  - OoO/Superscalar
- Thread-level parallelism — perform independent computation streams (composed of many instructions or SIMD instructions)
  - Multicore/SMT processors

Vectors are very useful data/mathematical representations

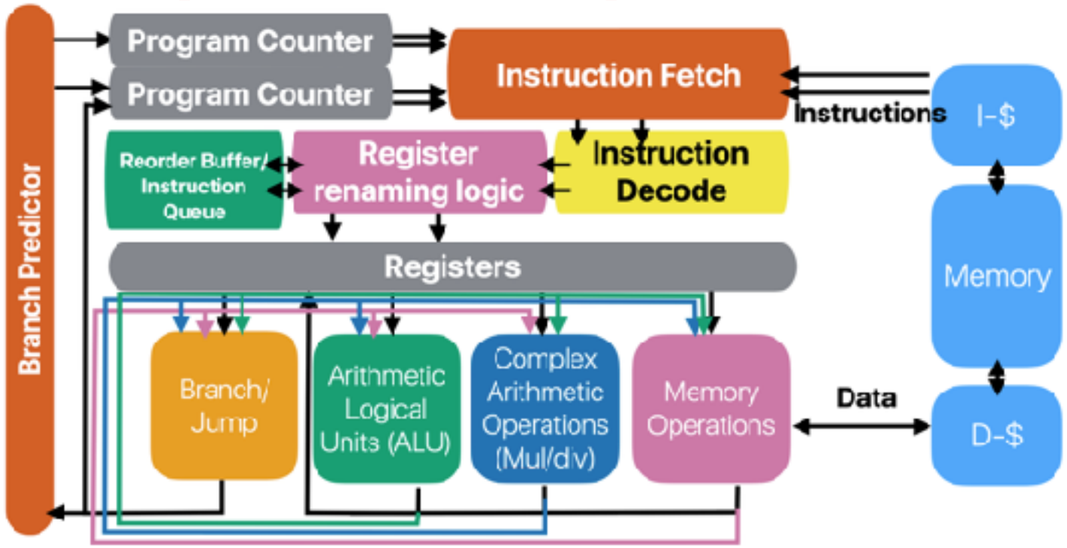


- But how do we process them using "superscalar" processors?

```
for(uint64_t i = 0; i < m; i++) {
    result = 0;
    for(uint64_t j = 0; j < n; j++) {
        result += matrix[i][j]*vector[j];
    }
     output[i] = result;
}
```

How well does vector processing map to super "scalar" processors

```
for(uint64_t i = 0; i < m; i++) {
    result = 0;
    for(uint64_t j = 0; j < n; j++) {
        result += matrix[i][j]*vector[j];
    }
     output[i] = result;
}
```



Assume we have a 5-issue INT, 3-load, 4-store pipeline like Alder Lake

Performance is very limited by both the memory bandwidth and available functional units

```
load vector[0]      load matrix[0][1]          load vector[3]          load matrix[0][4]
load matrix[0][0]   load vector[2]             load matrix[0][3]       load vector[5]
load vector[1]      load matrix[0][2]          load vector[4]          load matrix[0][5]

                    mul matrix[0][0], vector[0] mul matrix[0][1], vector[1] mul matrix[0][3], vector[3]
                                                mul matrix[0][2], vector[2]
```

Characteristics of vector processing

- Their operations are uniform across all pairs of elements

— Can we have just one instruction to control all pair-wise operations?

- There are very limited vector operations with mathematical meanings

— Can we simplify the processing elements design and make space for more PEs?

- They have very good spatial locality

— Can we have fetch them once & put in wide registers?

```
for(uint64_t i = 0; i < m; i++) {
    result = 0;
    for(uint64_t j = 0; j < n; j++) {
          result += matrix[i][j]*vector[j];
    }
     output[i] = result;
}
```

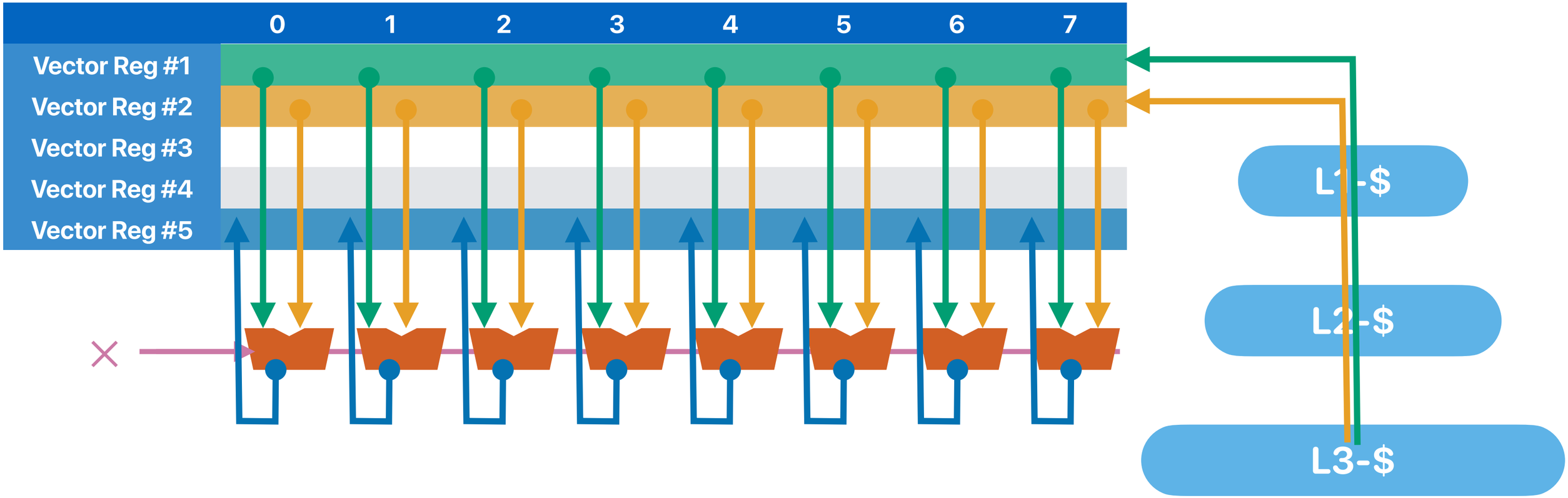
```
load vector[0]
load matrix[0][0]
load vector[1]
```

```
load matrix[0][1]
load vector[2]
load matrix[0][2]

mul matrix[0][0], vector[0]
```

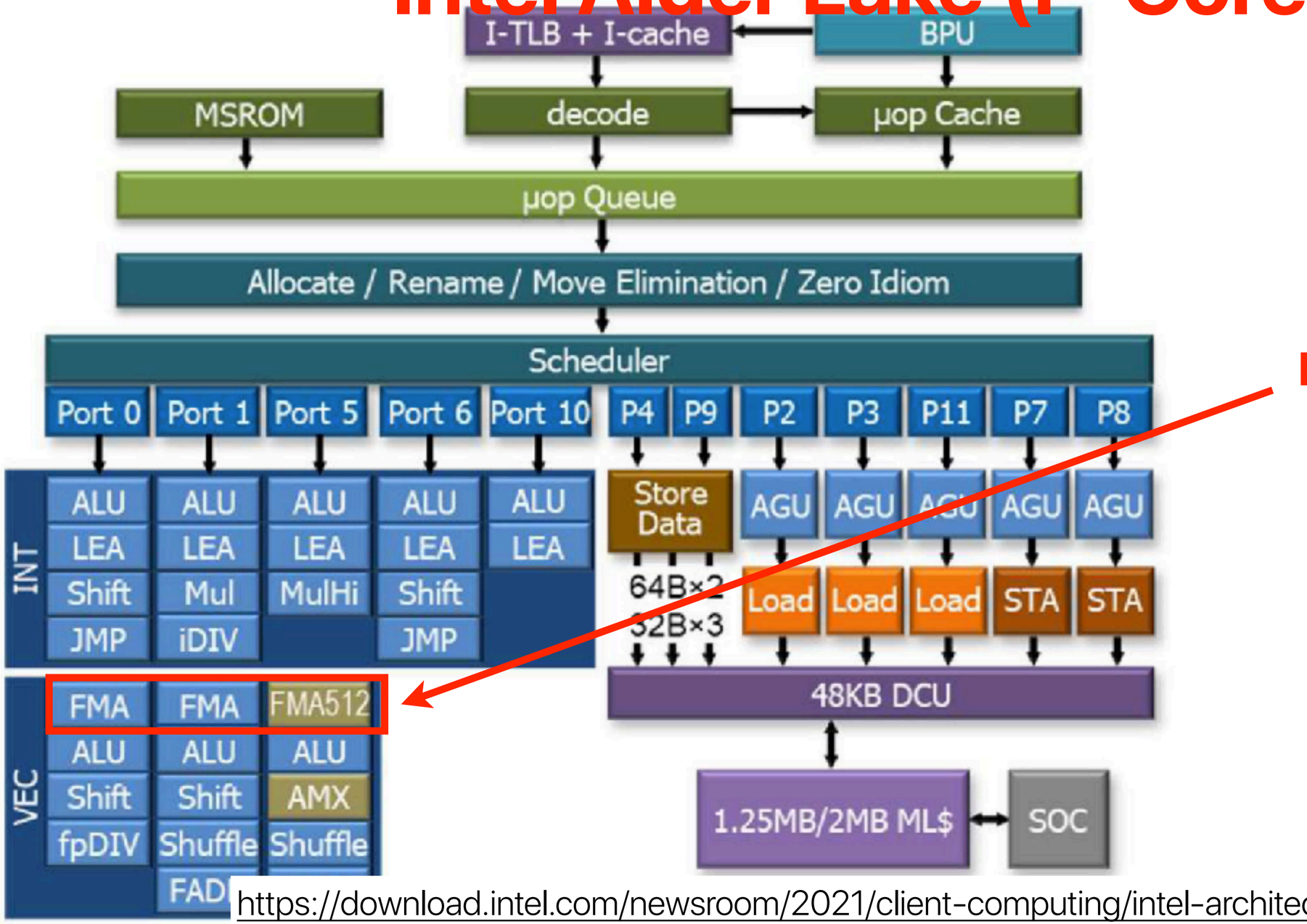
```
load vector[3]
load matrix[0][3]
load vector[4]

mul matrix[0][1], vector[1]
mul matrix[0][2], vector[2]
```

```
load matrix[0][4]
load vector[5]
load matrix[0][5]

mul matrix[0][3], vector[3]
```

Vector processing architecture



Intel Alder Lake (P-Core)



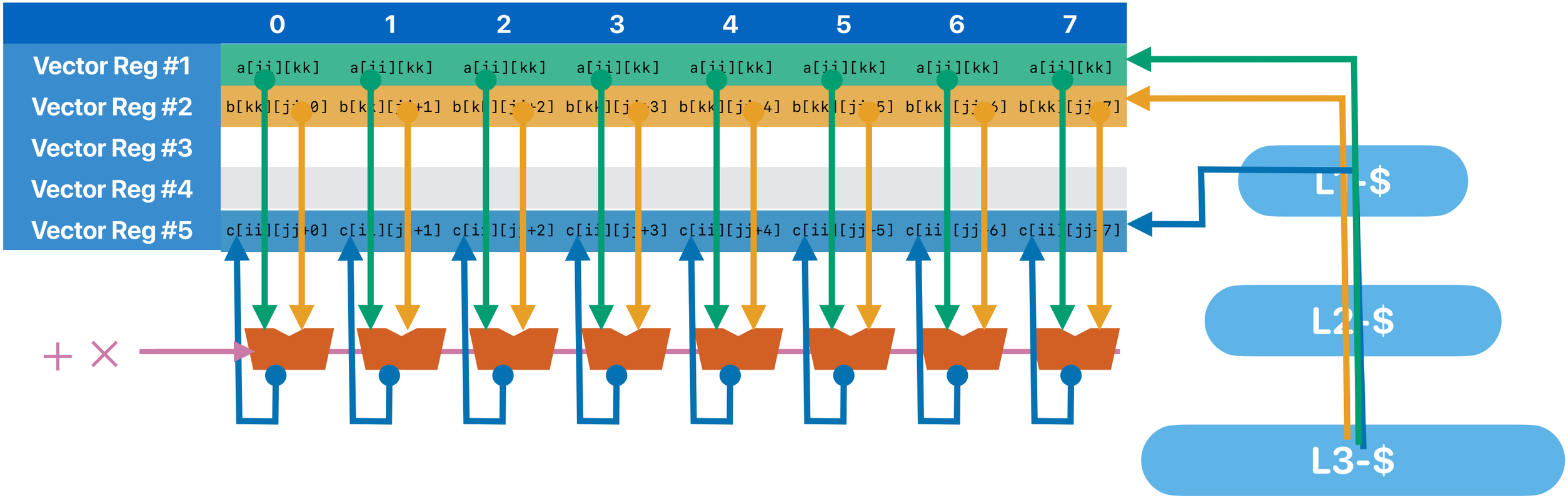
AVX Vector Processing Units

https://download.intel.com/newsroom/2021/client-computing/intel-architecture-day-2021-presentation.pdf

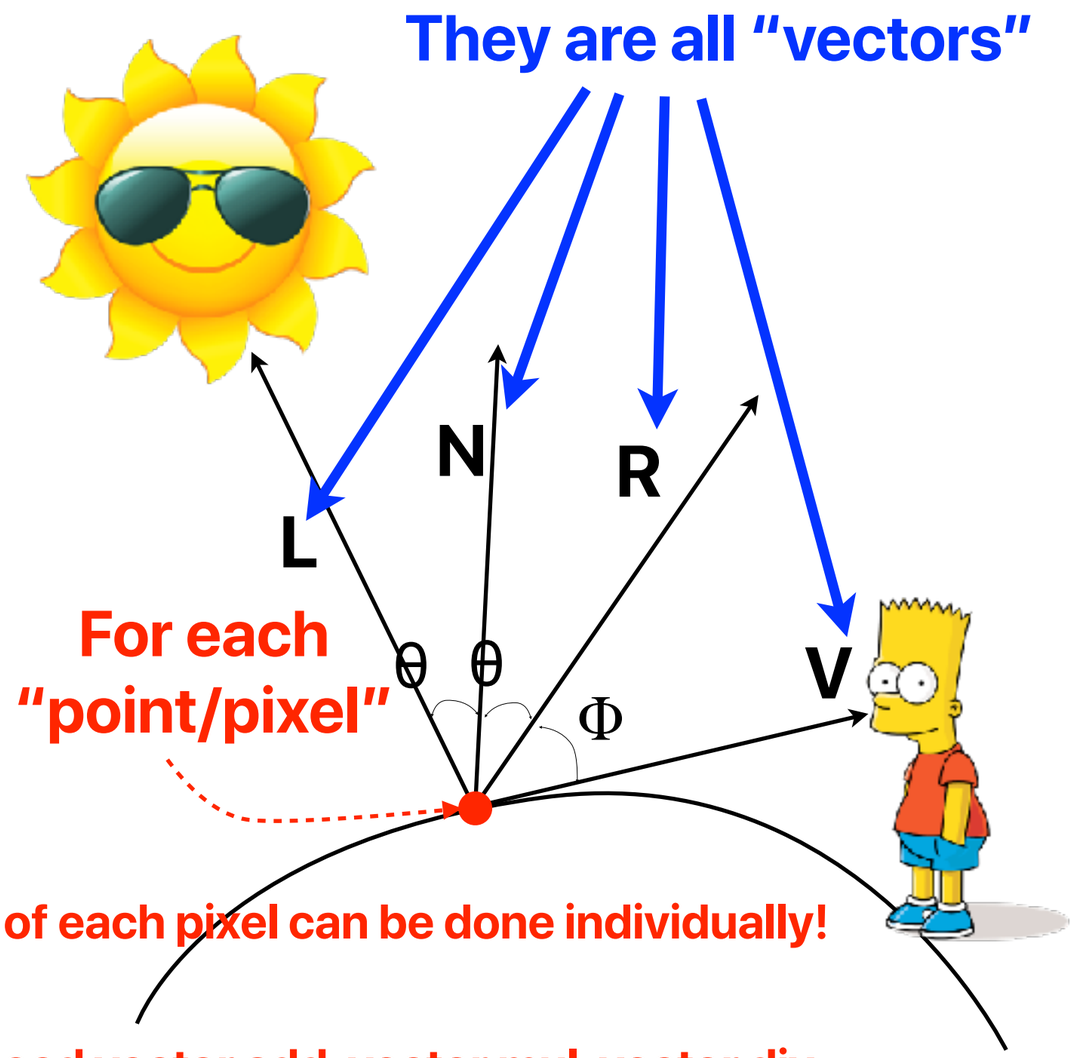
Vectorized matrix multiplications

```
#define VECTOR_WIDTH 8 // VECTOR_WIDTH is 8 as we're using AVX-512
void vector_blockmm(double **a, double **b, double **c, \
uint64_t tile_size) {
  int i,j,k, ii, jj, kk, x;
  __m512d va, vb, vc;
  for(i = 0; i < ARRAY_SIZE; i+=tile_size) {
    for(j = 0; j < ARRAY_SIZE; j+=tile_size) {
      for(k = 0; k < ARRAY_SIZE; k+=tile_size) {
        for(ii = i; ii < i+tile_size; ii++) {
          for(jj = j; jj < j+tile_size; jj+=VECTOR_WIDTH) {
             vc = _mm512_load_pd(&c[ii][jj]); // load c[ii][jj] to c[ii][jj+7] to vc[0-7]
             for(kk = k; kk < k+tile_size; kk++) {
                va = _mm512_broadcastsd_pd(&a[ii][kk]);// load a[ii][kk] to va[0-7]
                vb = _mm512_load_pd(&b[kk][jj]);  // load b[kk][jj] to b[kk][jj+7] to vb[0-7]
                vc = _mm512_add_pd(vc,_mm512_mul_pd(va,vb));    // vc += va * vb
             }
             _mm512_store_pd(&c[ii][jj],vc);      // store vc to c[ii][jj] to c[ii][jj+4]
          }
        }
      }
    }
  }
}
```

Vector processing architecture



Basic concept of shading



$$I_{amb} = K_{amb} \cdot M_{amb}$$

$$I_{diff} = K_{diff} \cdot M_{diff} \cdot (N \cdot L)$$

$$I_{spec} = K_{spec} \cdot M_{spec} \cdot (R \cdot V)^n$$

$$I_{total} = I_{amb} + I_{diff} + I_{spec}$$

```
void main(void)
{
    // normalize vectors after interpolation
    vec3 L = normalize(o_toLight);
    vec3 V = normalize(o_toCamera);
    vec3 N = normalize(o_normal);

    // get Blinn-Phong reflectance components
    float Iamb = ambientLighting();
    float Idif = diffuseLighting(N, L);
    float Ispe = specularLighting(N, L, V);

    // diffuse color of the object from texture
    vec3 diffuseColor = texture(u_diffuseTexture, o_texcoords)

    // combination of all components and diffuse color of the
    resultingColor.xyz = diffuseColor * (Iamb + Idif + Ispe);
    resultingColor.a = 1;
}
```

The work of each pixel can be done individually!

We only need vector add, vector mul, vector div.

https://www.pollev.com/hungweitseng close in 1:30

What GPU architectures should look like?

- Based on the original target of GPU design, what do you think would be reasonable design decisions that GPU architectures make?
  - ① The architecture should render pixels as fast as possible
  - ② The architecture does not need a branch unit
  - ③ The architecture should contain an array of powerful ALUs and functional units that can perform a rich set of operations
  - ④ The architecture should offer very large bandwidth of memory accesses

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

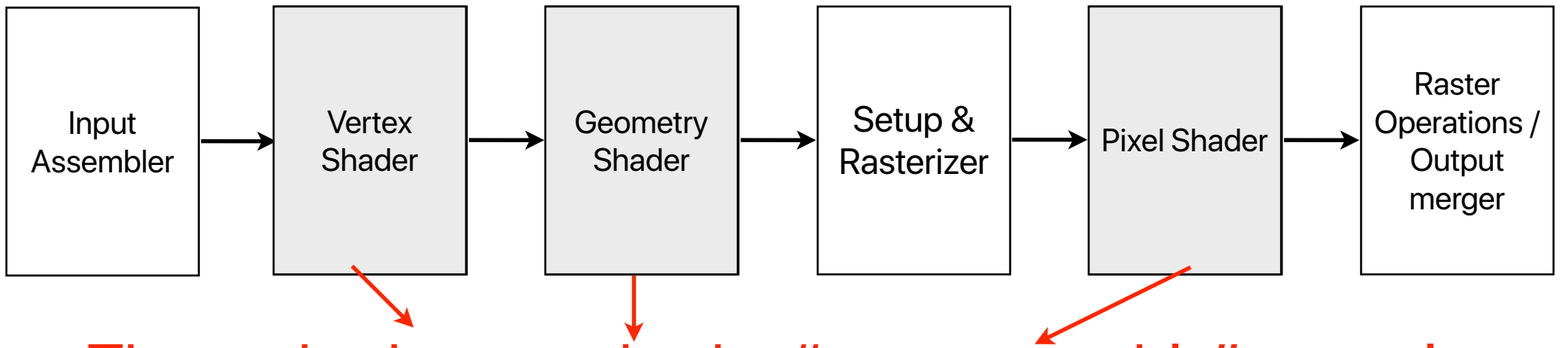
What GPU architectures should look like?

- Based on the original target of GPU design, what do you think would be reasonable design decisions that GPU architectures make?
  - ① The architecture should render pixels as fast as possible
  - ② The architecture does not need a branch unit
  - ③ The architecture should contain an array of powerful ALUs and functional units that can perform a rich set of operations
  - ④ The architecture should offer very large bandwidth of memory accesses
  - A. 0
  - B. 1
  - C. 2
  - D. 3
  - E. 4

GPU (Graphics Processing Unit)

- Originally for displaying images
- HD video: 1920*1080 pixels * 60 frames per second
  - Therefore, GPU is not latency-oriented by design!
  - Even for 120 frames, you still have 8ms latency to get everything done!
- Graphics processing pipeline

1 GHz can give you 8000000 cycles!!!



These shaders need to be "programmable" to apply different rendering effects/algorithms (Phong shading, Gouraud shading, and etc...)

What GPU architectures should look like?

- Based on the original target of GPU design, what do you think would be reasonable design decisions that GPU architectures make?
  - ① The architecture should render pixels as fast as possible
  - ② The architecture does not need a branch unit
  - ③ The architecture should contain an array of powerful ALUs and functional units that can perform a rich set of operations
  - ④ The architecture should offer very large bandwidth of memory accesses

  A. 0

  B. 1

  C. 2

  D. 3

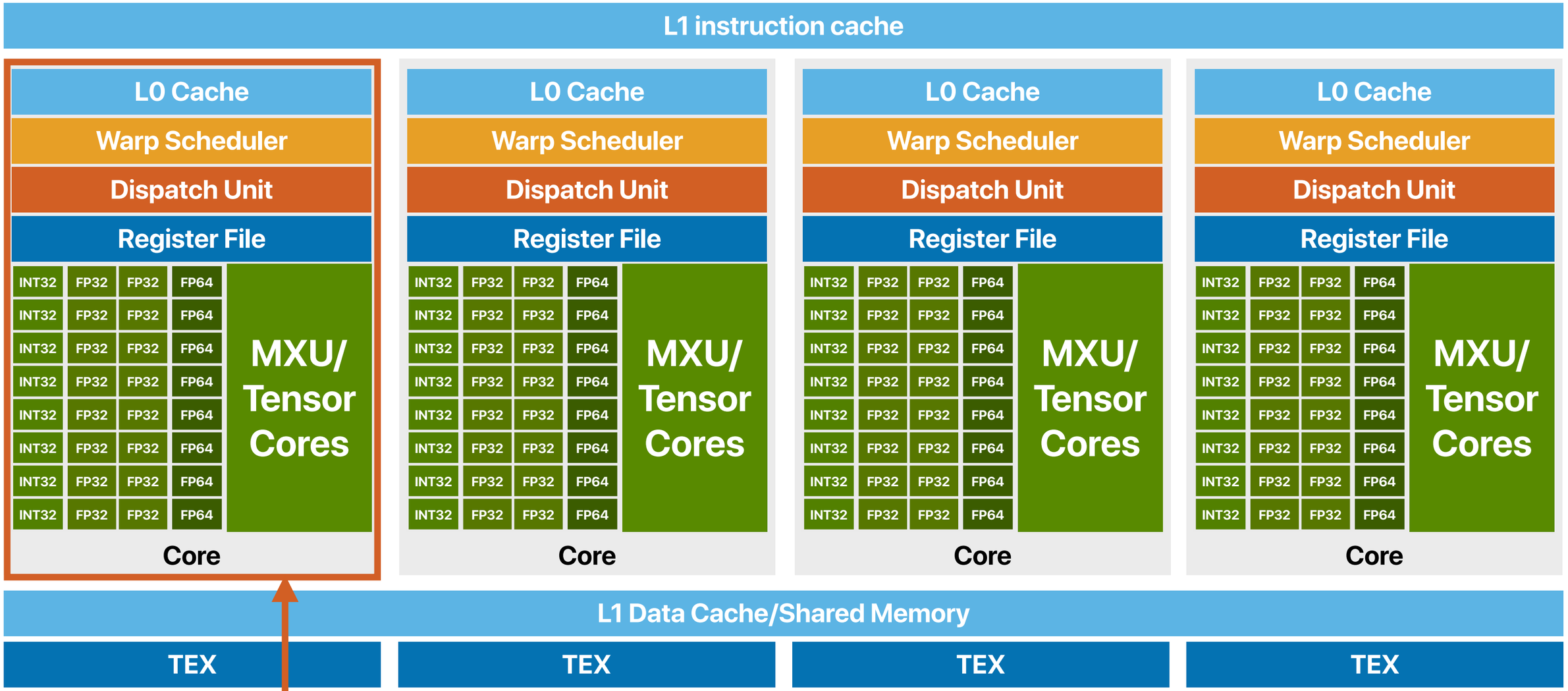
  E. 4

What's the "appropriate" GPU architecture

- Lots of ALUs to process pixels in parallel — 2M pixels in HD resolution, very regular workloads
  - Vector processing model
- Simple operations
  - The ALUs only supports very few instructions
  - Almost no branches
- Deadline driven and throughput-oriented rather than latency oriented
  - High-bandwidth but also "higher-latency" memory
  - ALUs can be slower

GPU also follows the idea of slower, but more!

GPU Architecture



NVIDIA CUDA GPU function

```
__global__ void gpu_matrix_mult(D_TYPE *a,D_TYPE *b,
D_TYPE *c, int size)
{
    int row = blockIdx.y * blockDim.y + threadIdx.y;
    int col = blockIdx.x * blockDim.x + threadIdx.x;
    D_TYPE sum = 0;
    for(int i = 0; i < size; i++) {
            sum += a[row * size + i] * b[i * size + col];
    }
    c[row * size + col] = sum;
}
```

Parallelism in modern computers

- Instruction-level parallelism — perform various, independent instructions simultaneously
  - Pipeline
  - OoO/Superscalar
- Data-level parallelism — perform the same operation on multiple data elements in parallel
  - SIMD instructions
  - Compute within an SM in GPUs
- Thread-level parallelism — perform independent computation streams (composed of many instructions or SIMD instructions)
  - Multicore/SMT processors
  - Compute using multiple SMs in GPUs

GPUs still suffer from broken Dennard Scaling

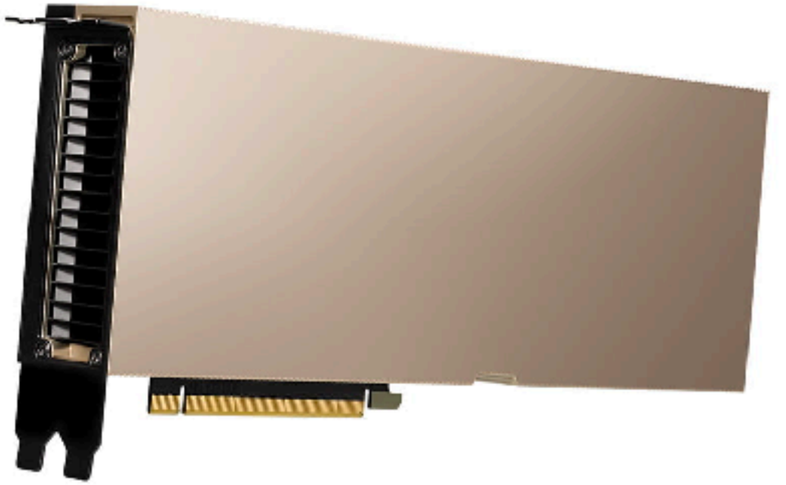
NVIDIA Accelerator Specification Comparison			
	H100	A100 (80GB)	V100
FP32 CUDA Cores	16896	6912	5120
Tensor Cores	528	432	640
GPU	GH100 (814mm2)	GA100 (826mm2)	GV100 (815mm2)
Transistor Count	80B	54.2B	21.1B
TDP	700W	400W	300W/350W
Manufacturing Process	TSMC 4N	TSMC 7N	TSMC 12nm FFN
Interface	SXM5	SXM4	SXM2/SXM3
Architecture	Hopper	Ampere	Volta

1.46x

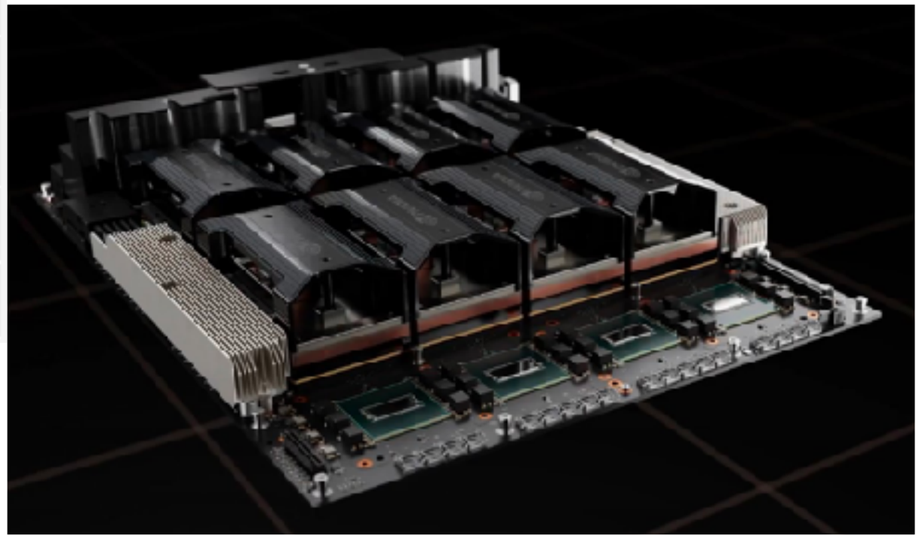
1.75x

8.75 W/ B Transistors

7.38 W/ B Transistors



https://www.workstationspecialist.com/product/nvidia-tesla-a100/



https://www.servethehome.com/wp-content/uploads/2022/03/NVIDIA-GTC-2022-H100-in-HGX-H100.jpg

Take-aways: Challenges and SOTA solutions in the dark silicon era

- **Even if we can address all programming challenges, multi-core performance has stopped to scale due to the Dark Silicon problem**
- Aggressive dynamic frequency/voltage scaling on CMP to accommodate the demand of **latency**-sensitive, parallelism-limited applications, but the area-efficiency of the slower cores is not great
- GPUs/many-core processors improve the **throughput** per-chip through providing massive parallelism where each processing element operates at a lower speed, but not-ideal for latency-sensitive workloads

Announcements

- Last office hour by Hung-Wei — tomorrow
- iEVAL started and ends on 12/05/2024
  - Submit the prove of your participation in course evaluation through Gradescope
  - It can become a full credit reading quiz (it helps to amortize the penalty of another least performing one) — we will drop a total of three now
- **Assignment 5 and programming assignment 3 due this Thursday**
- **Final exam**
  - 12/10 8a-11a
  - Closed book, no cheatsheet — the same rules as the midterm

Computer
Science &
Engineering

203

つづく