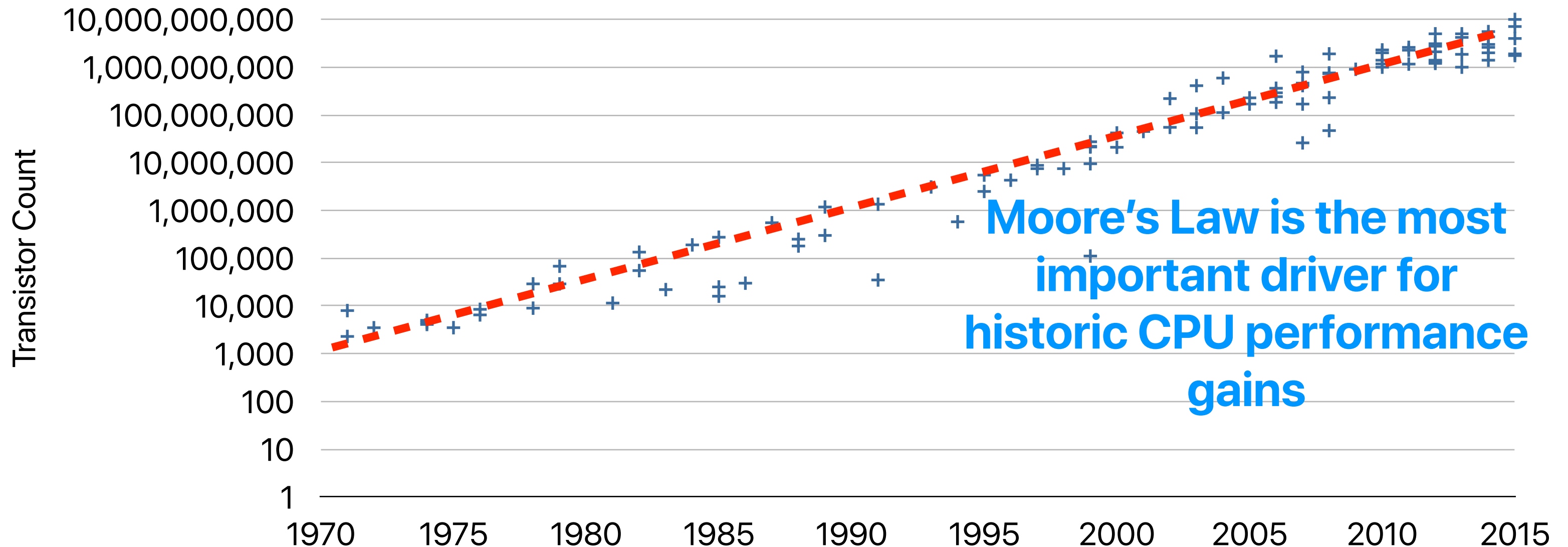


# Clock, Power Consumption and the Future Landscape of Computation

Prof. Usagi

# Recap: Moore's Law<sup>(1)</sup>

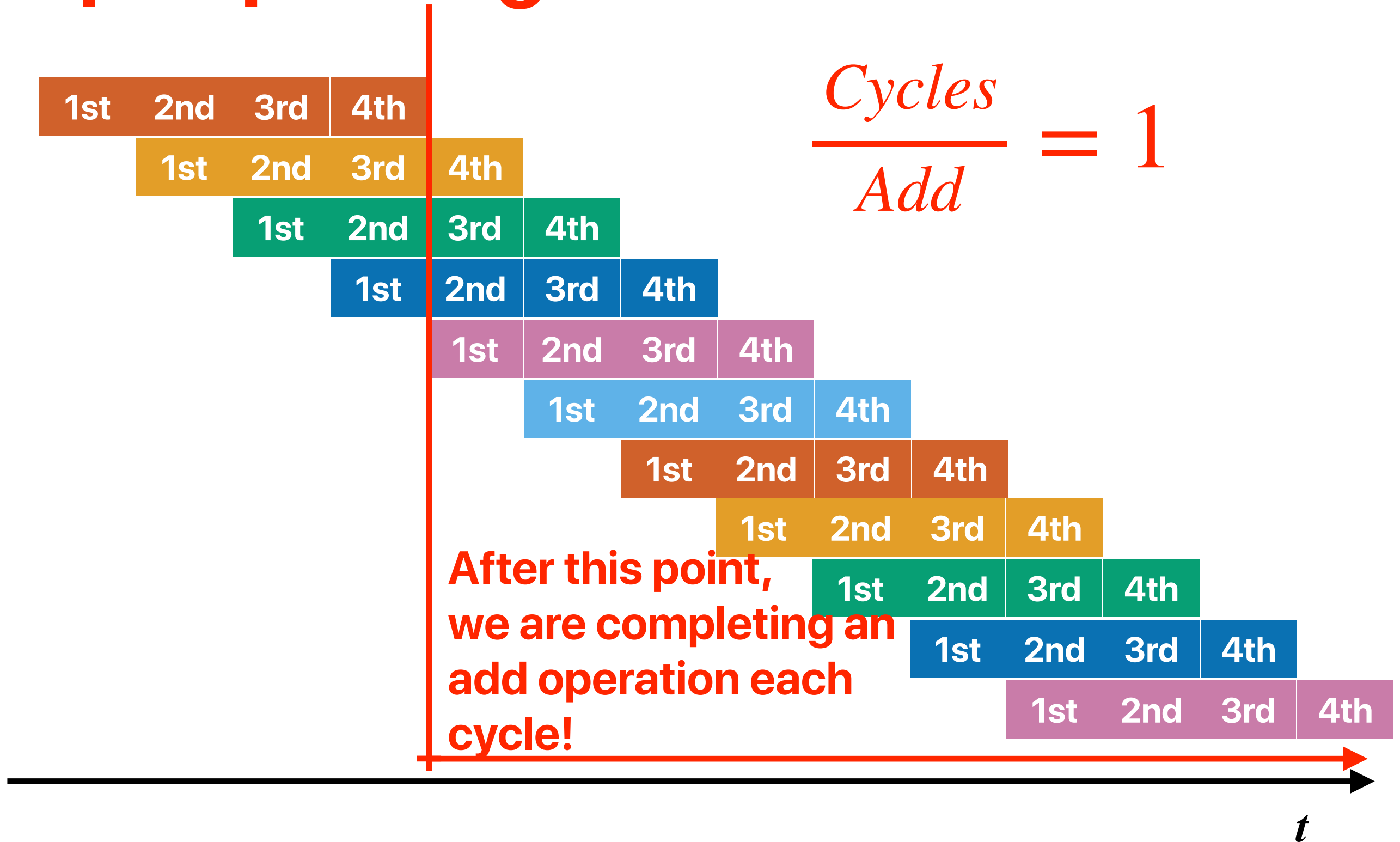
- The number of transistors we can build in a fixed area of silicon doubles every 12 ~ 24 months.



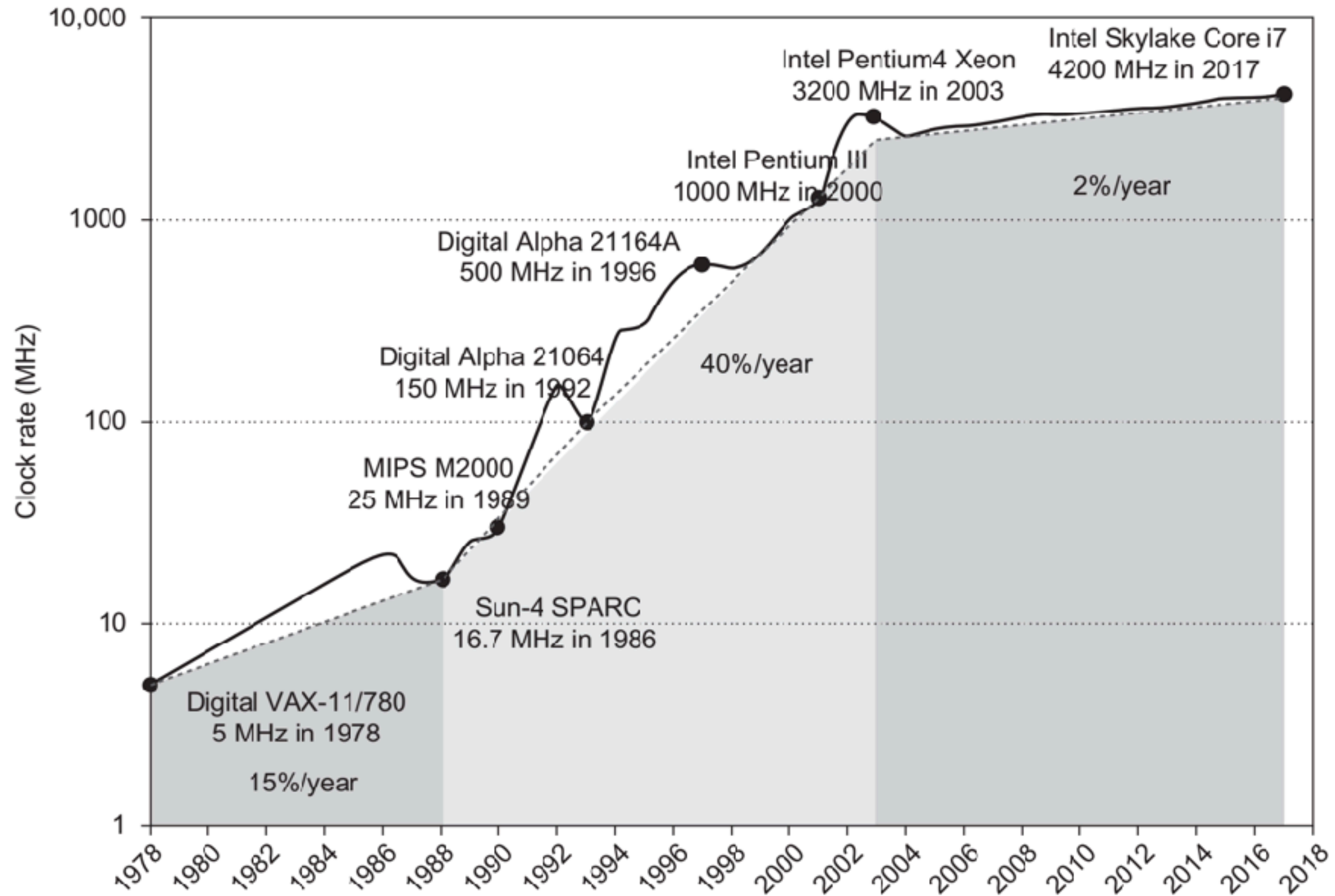
(1) Moore, G. E. (1965), 'Cramming more components onto integrated circuits', Electronics 38 (8) .

# Recap: Pipelining a 4-bit serial adder

add a, b  
add c, d  
add e, f  
add g, h  
add i, j  
add k, l  
add m, n  
add o, p  
add q, r  
add s, t  
add u, v



# Recap: The growth of clock rate is slowing down

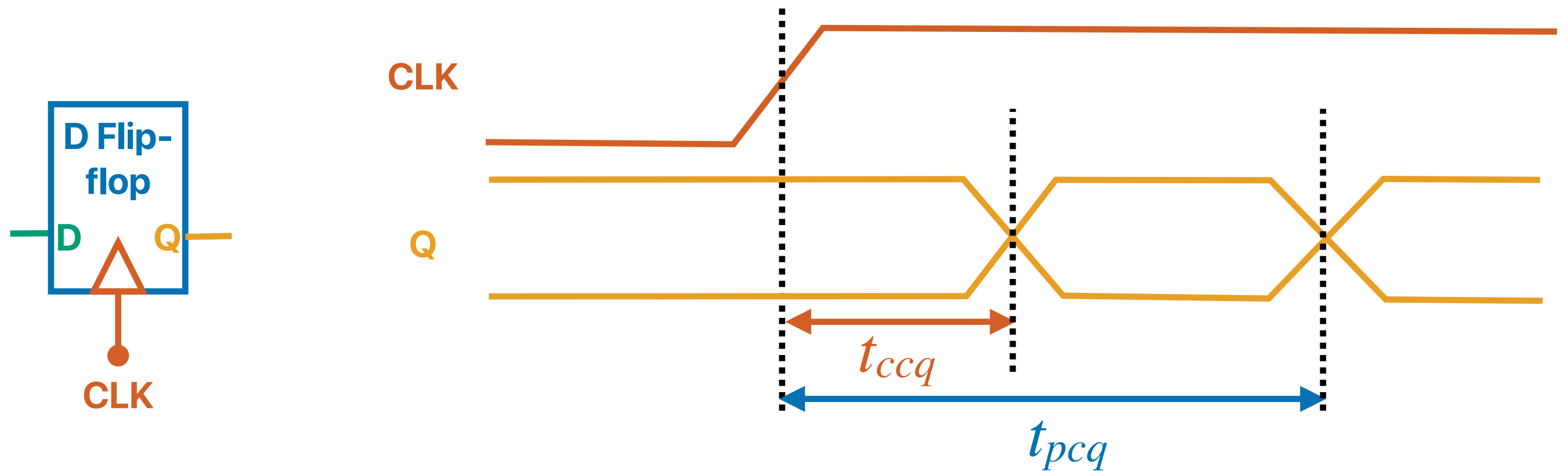


# Outline

- What are the basic limits of clock frequency?
- New challenges: Power consumption
- Opportunities and the future

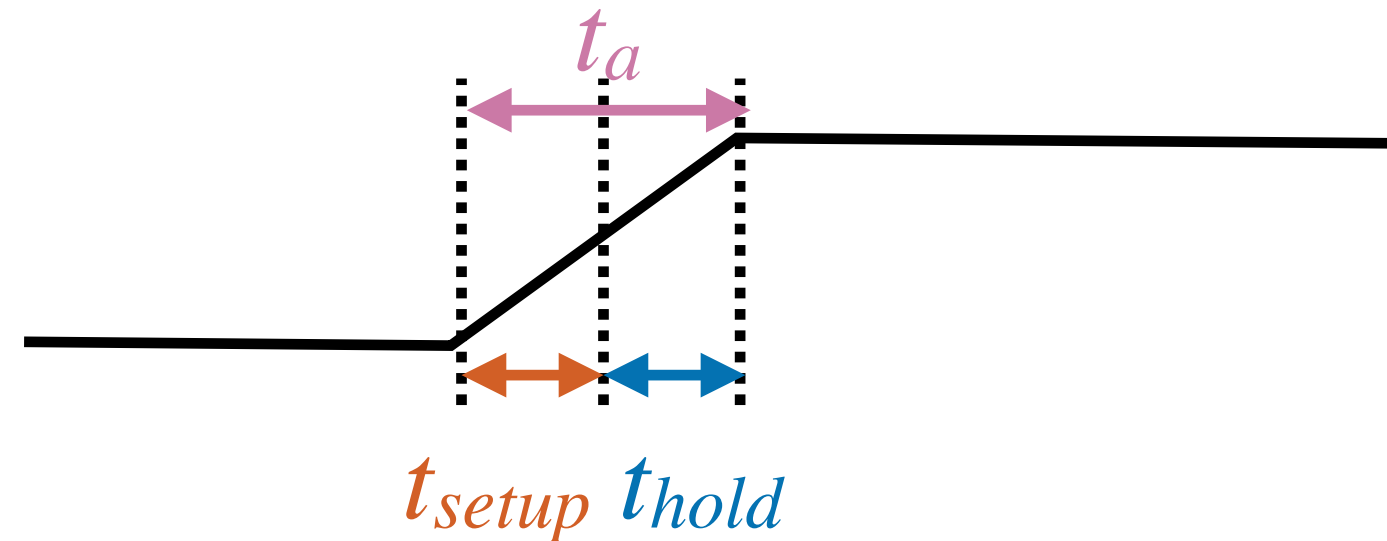
# Timing constraints

# Output Timing Constraints



- Min delay of FF, also called contamination delay or min CLK to Q delay:  $t_{ccq}$ 
  - Time after clock edge that Q might be unstable (i.e., starts changing)
- Max delay of FF, also called propagation delay or maximum CLK to Q delay:  $t_{pcq}$ 
  - Time after clock edge that the output Q is guaranteed to be stable (i.e. stops changing)

# Setup and hold times for a flip-flop



- Setup time:  $t_{setup}$ 
  - Time before the clock edge that data must be stable (i.e. not change)
- Hold time:  $t_{hold}$ 
  - Time after the clock edge that data must be stable
- Aperture time:  $t_a$ 
  - Time around clock edge that data must be stable ( $t_a = t_{setup} + t_{hold}$ )

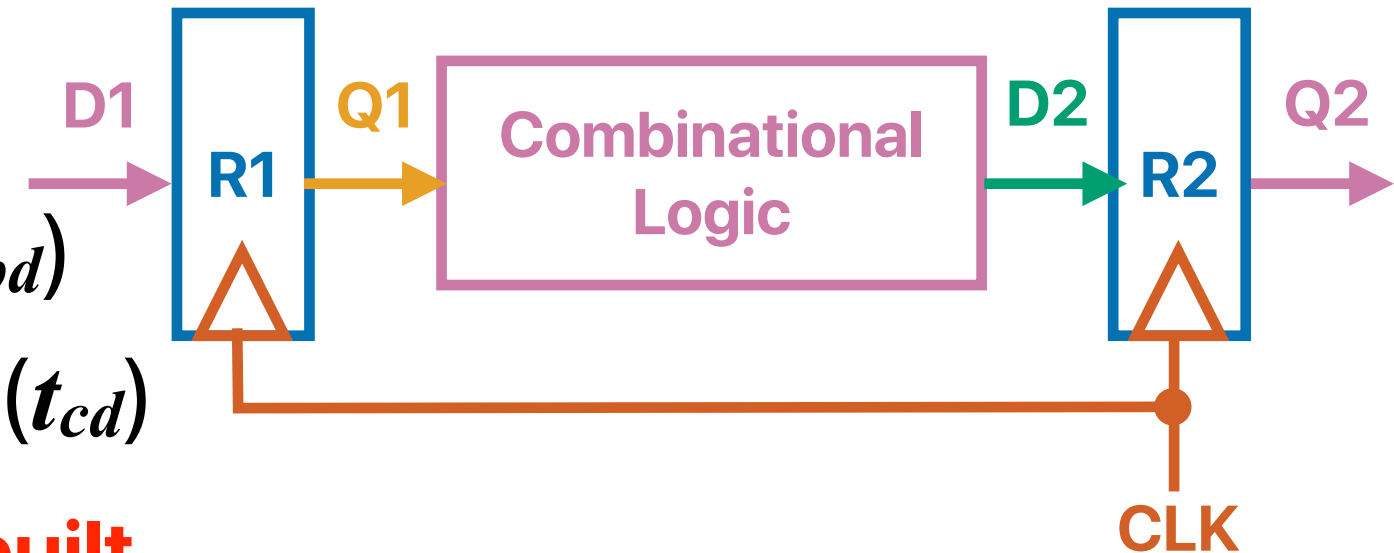


# Summary on timing constraints

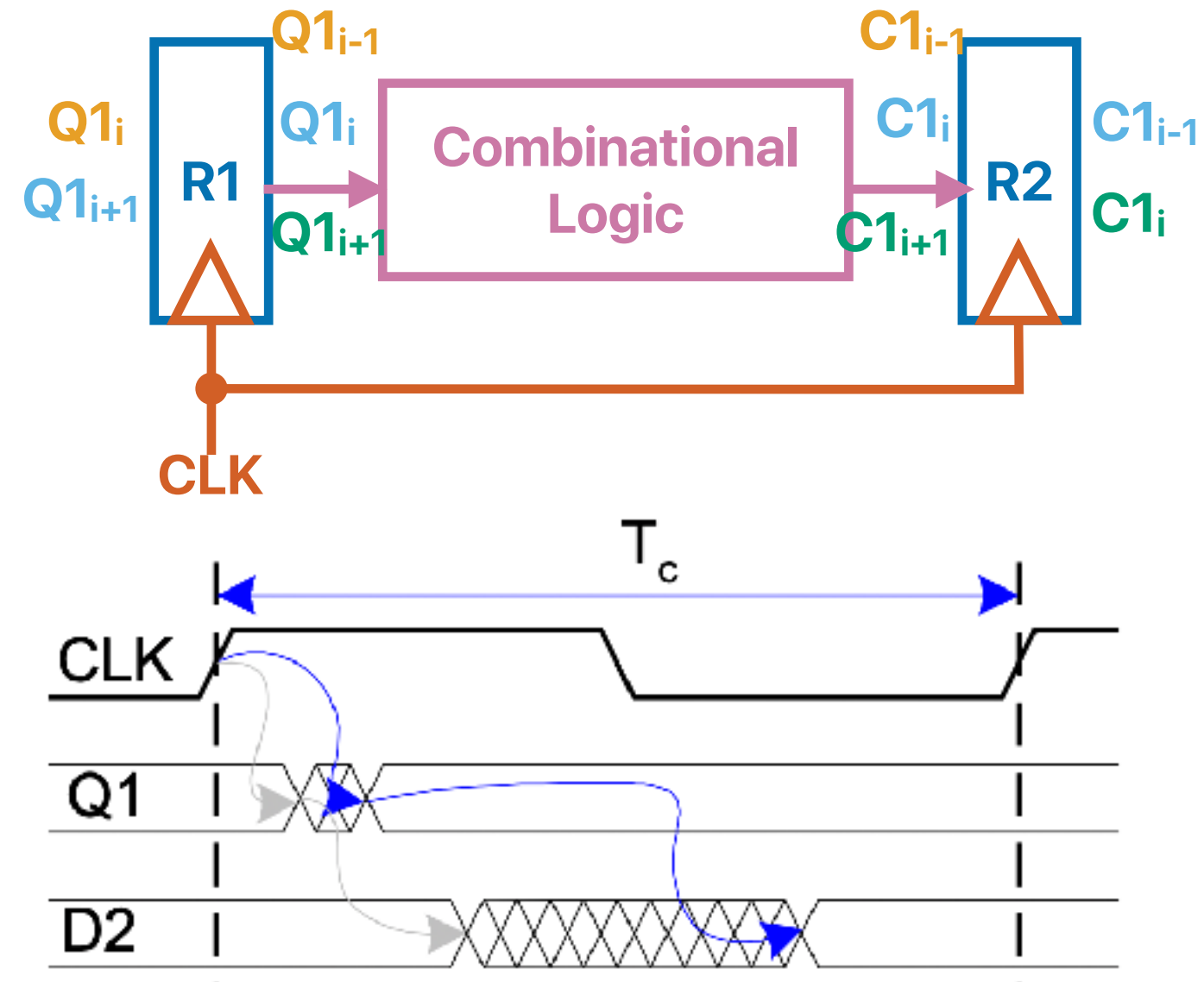
- Combinational:
  - Maximum delay = **P**ropagation **d**elay ( $t_{pd}$ )
  - Minimum delay = **C**ontamination **d**elay ( $t_{cd}$ )

- Flip Flops:
    - Input
- Once the logic/FFs are built, these timing characteristics are fixed properties

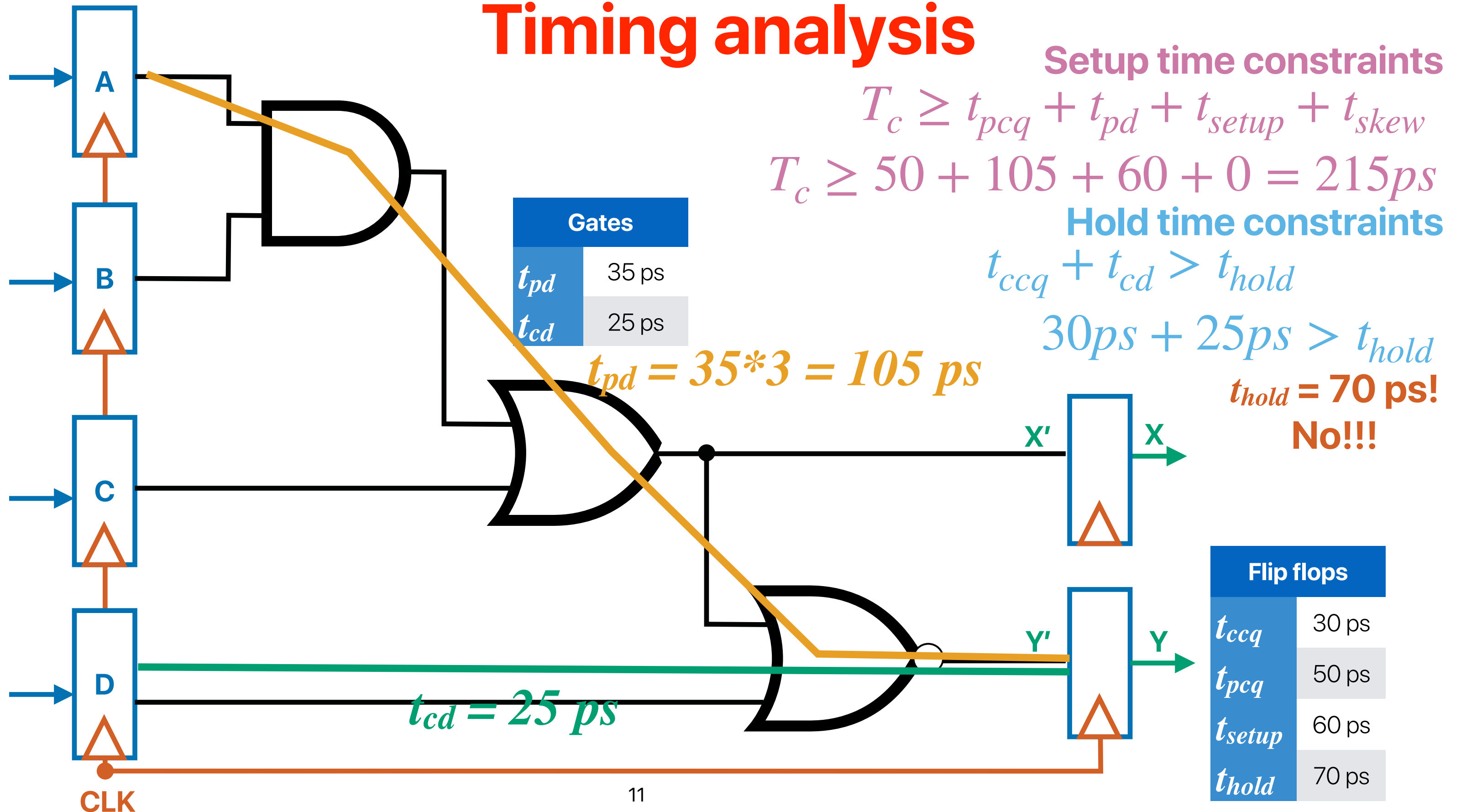
- **S**etup time ( $t_{setup}$ )
  - **H**old time ( $t_{hold}$ )
- Output
  - **P**ropagation **c**lock-to-**Q** time ( $t_{pcq}$ )
  - **C**ontamination **c**lock-to-**Q** time ( $t_{ccq}$ )



# Timing in a circuit



# Timing analysis



# Timing analysis

## Setup time constraints

$$T_c \geq t_{pcq} + t_{pd} + t_{setup}$$

$$T_c \geq 50 + 105 + 60 + 0 = 215ps$$

## Hold time constraints

$$t_{ccq} + t_{cd} > t_{hold}$$

$$30ps + 25ps + 25ps > t_{hold}$$

$$t_{pd} = 35 * 3 = 105 \text{ ps}$$

# Buffers

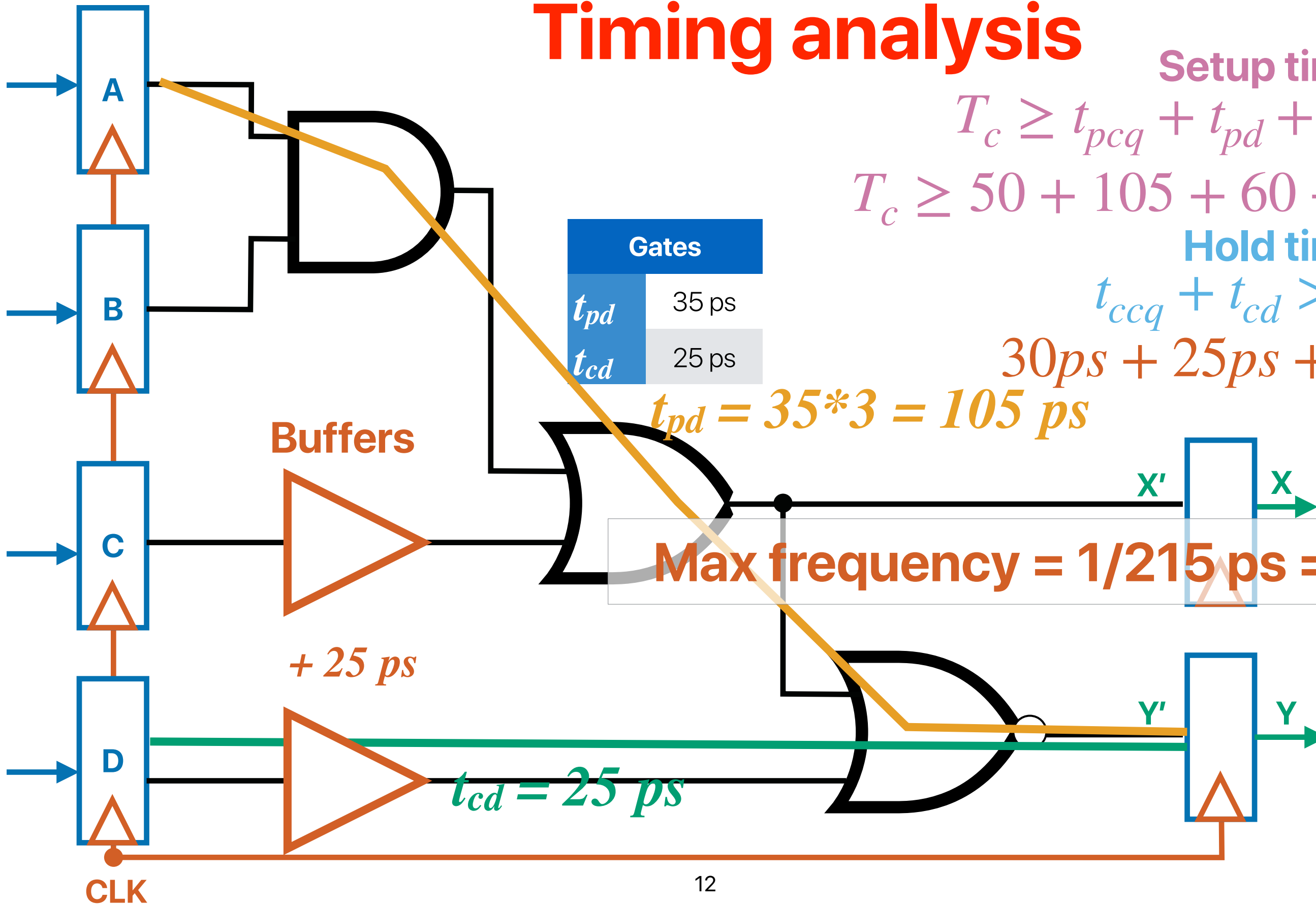
*+ 25 ps*

 ~~$t_{cd} = 25 \text{ ps}$~~ 

**Max frequency =  $1/215 \text{ ps} = 4.65 \text{ GHz}$ !**

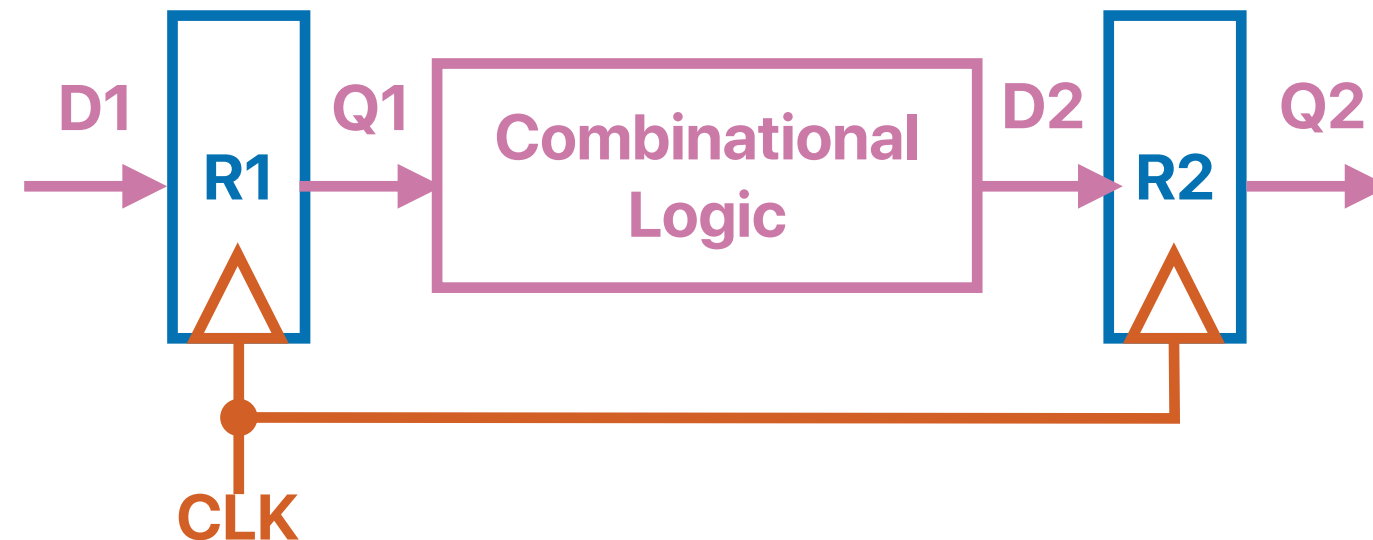
Gates	
$t_{pd}$	35 ps
$t_{cd}$	25 ps

Flip flops	
$t_{ccq}$	30 ps
$t_{pcq}$	50 ps
$t_{setup}$	60 ps
$t_{hold}$	70 ps



# FF Timing Parameters

- Once a flip flop has been built, its timing characteristics stay fixed:  $t_{setup}$  ,  $t_{hold}$  ,  $t_{ccq}$  ,  $t_{pcq}$
- What about the clock? Does the clock edge arrive at the same time to all the D-FFs on the chip?





Zoom

poll close in 0:48

### Minimum number of SOP terms

• Minimum number of SOP terms to cover the following function?

Input			Output
A	B	C	
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Participants (16): Hung-Wei Tseng (主持人, 我), Hung-Wei... (聯席主持人), Abhi Madduri, Alex Castillo, Alex Nguyen, Alp Aloglu, Anthony Muresan, Anthony Salinas, Apurav Dhesi, Chapman Hui, CHENGYUE LIN

EECS120A Lectures - YouTube

poll close in 1:13

### Minimum number of SOP terms

• Minimum number of SOP terms to cover the following function?

Input			Output
A	B	C	
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

zoom

poll close in 0:44

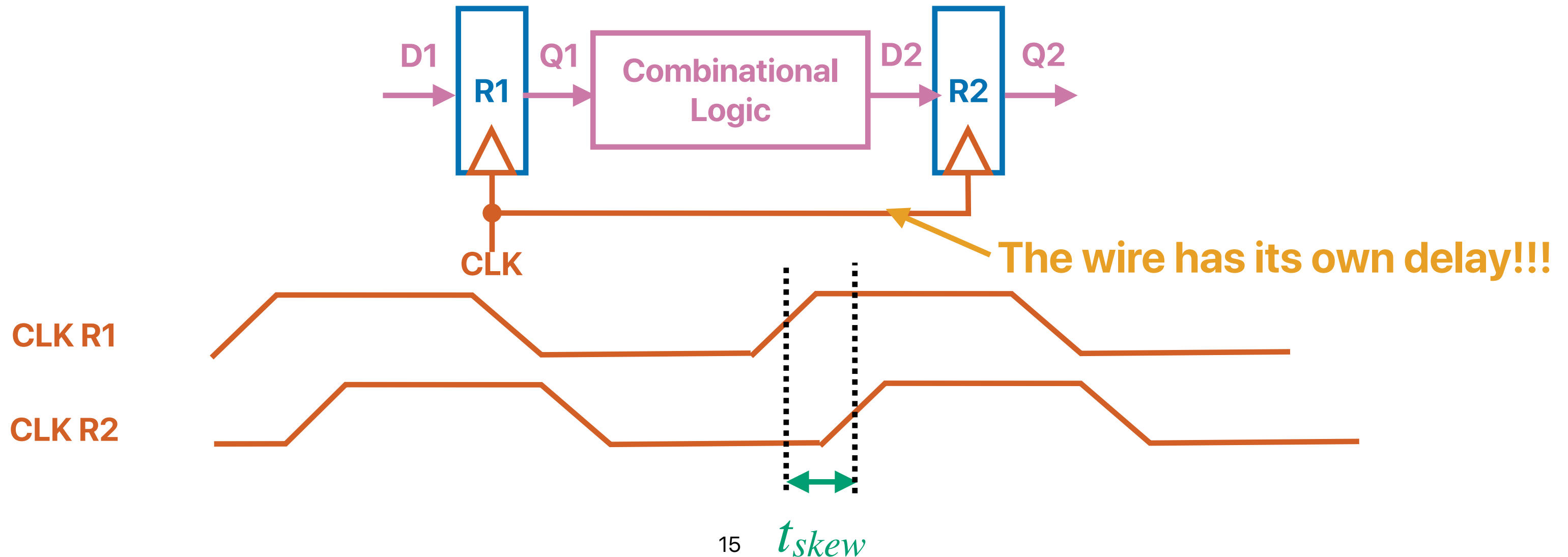
### Minimum number of SOP terms

• Minimum number of SOP terms to cover the following function?

Input			Output
A	B	C	
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

# Clock Skew

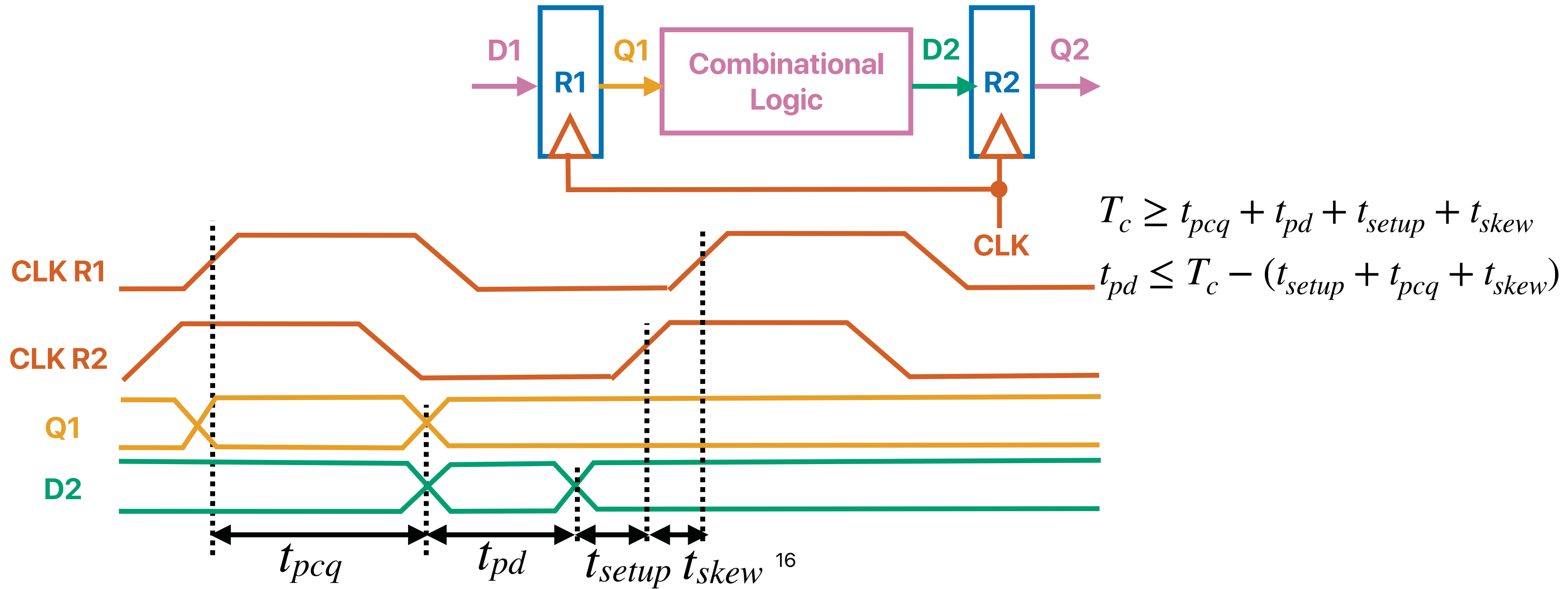
- The clock doesn't arrive at all registers at the same time
- **Skew**: difference between the two clock edges
- Perform the **worst case analysis**





# Setup Time Constraint with Skew

- In the worst case, CLK2 is earlier than CLK1
- $t_{pcq}$  is max delay through FF,  $t_{pd}$  is max delay through logic





# Power consumption

# Power v.s. Energy

- Power is the direct contributor of "heat"
  - Packaging of the chip
  - Heat dissipation cost
  - $\text{Power} = P_{\text{Dynamic}} + P_{\text{static}}$
- $\text{Energy} = P * ET$ 
  - 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

# Dynamic/Active Power

- The power consumption due to the switching of transistor states

- Dynamic power per transistor

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

- $\alpha$ : average switches per cycle
- $C$ : capacitance
- $V$ : voltage
- $f$ : frequency, usually linear with  $V$
- $N$ : the number of transistors

# Static/Leakage Power

- The power consumption due to leakage — transistors do not turn all the way off during no operation
- Becomes the **dominant** factor in the most advanced process technologies.

$$P_{leakage} \sim N \times V \times e^{-V_t}$$

- $N$ : number of transistors
- $V$ : voltage
- $V_t$ : threshold voltage where transistor conducts (begins to switch)

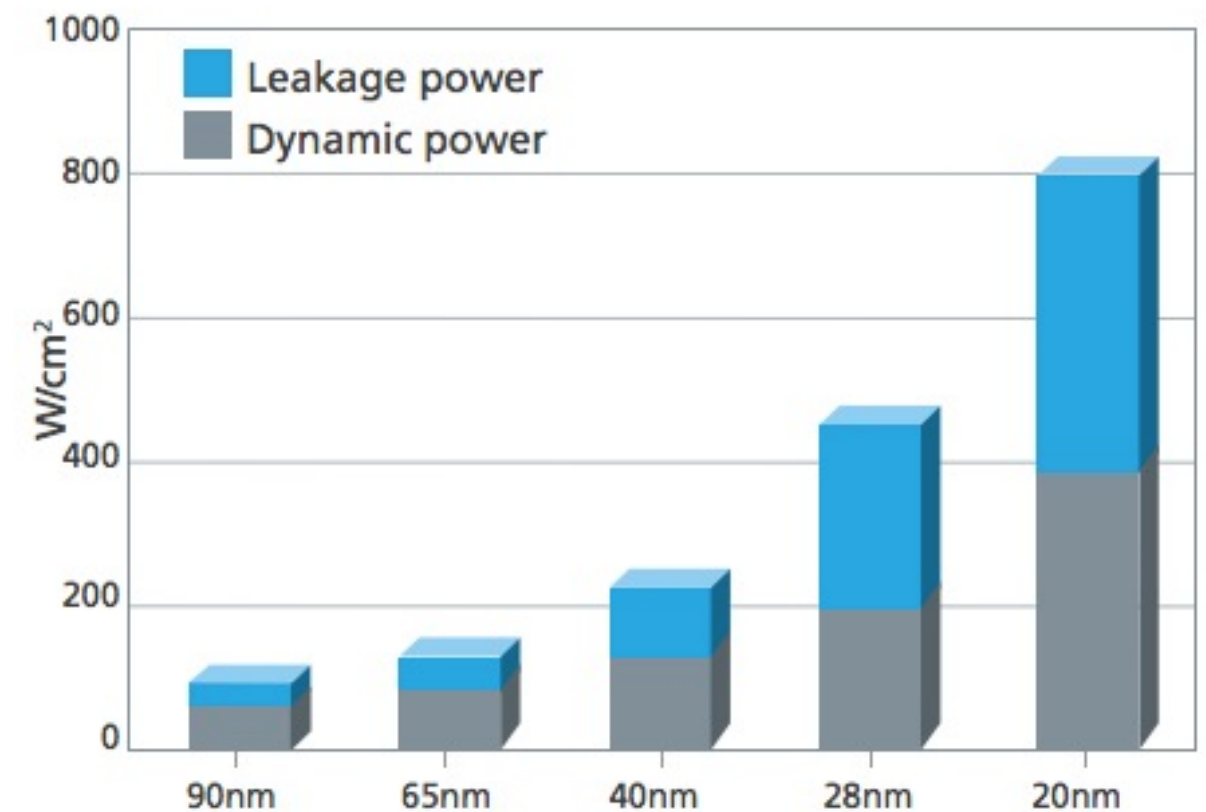


Figure 1: Leakage power becomes a growing problem as demands for more performance and functionality drive chipmakers to nanometer-scale process nodes (Source: IBS).

# Dynamic/Active Power

- The power consumption due to the switching of transistor states

- Dynamic power per transistor

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

- $\alpha$ : average switches per cycle

- $C$ : capacitance

- $V$ : voltage

- $f$ : frequency, usually linear with  $V$

- $N$ : the number of transistors

# Dennardian Broken

- Given a scaling factor  $S$

Parameter	Relation	Classical Scaling	Leakage Limited
Power Budget		1	1
Chip Size		1	1
Vdd (Supply Voltage)		$1/S$	1
Vt (Threshold Voltage)	$1/S$	$1/S$	1
tex (oxide thickness)		$1/S$	$1/S$
W, L (transistor dimensions)		$1/S$	$1/S$
Cgate (gate capacitance)	$WL/tox$	$1/S$	$1/S$
Isat (saturation current)	$WVdd/tox$	$1/S$	1
F (device frequency)	$Isat/(CgateVdd)$	$S$	$S$
D (Device/Area)	$1/(WL)$	$S^2$	$S^2$
p (device power)	$IsatVdd$	$1/S^2$	1
P (chip power)	$Dp$	1	$S^2$
U (utilization)	$1/P$	1	$1/S^2$

# Power consumption

## Dennardian Scaling

## Dennardian Broken

### Chip

1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

**=49W**

### Chip

0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

**=50W**

### Chip

1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

**=100W!**

# Power density

## Dennardian Scaling

## Dennardian Broken

Chip

1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

$$= \frac{49W}{\text{Chip Area}}$$

Chip

0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

$$= \frac{50W}{\text{Chip Area}}$$

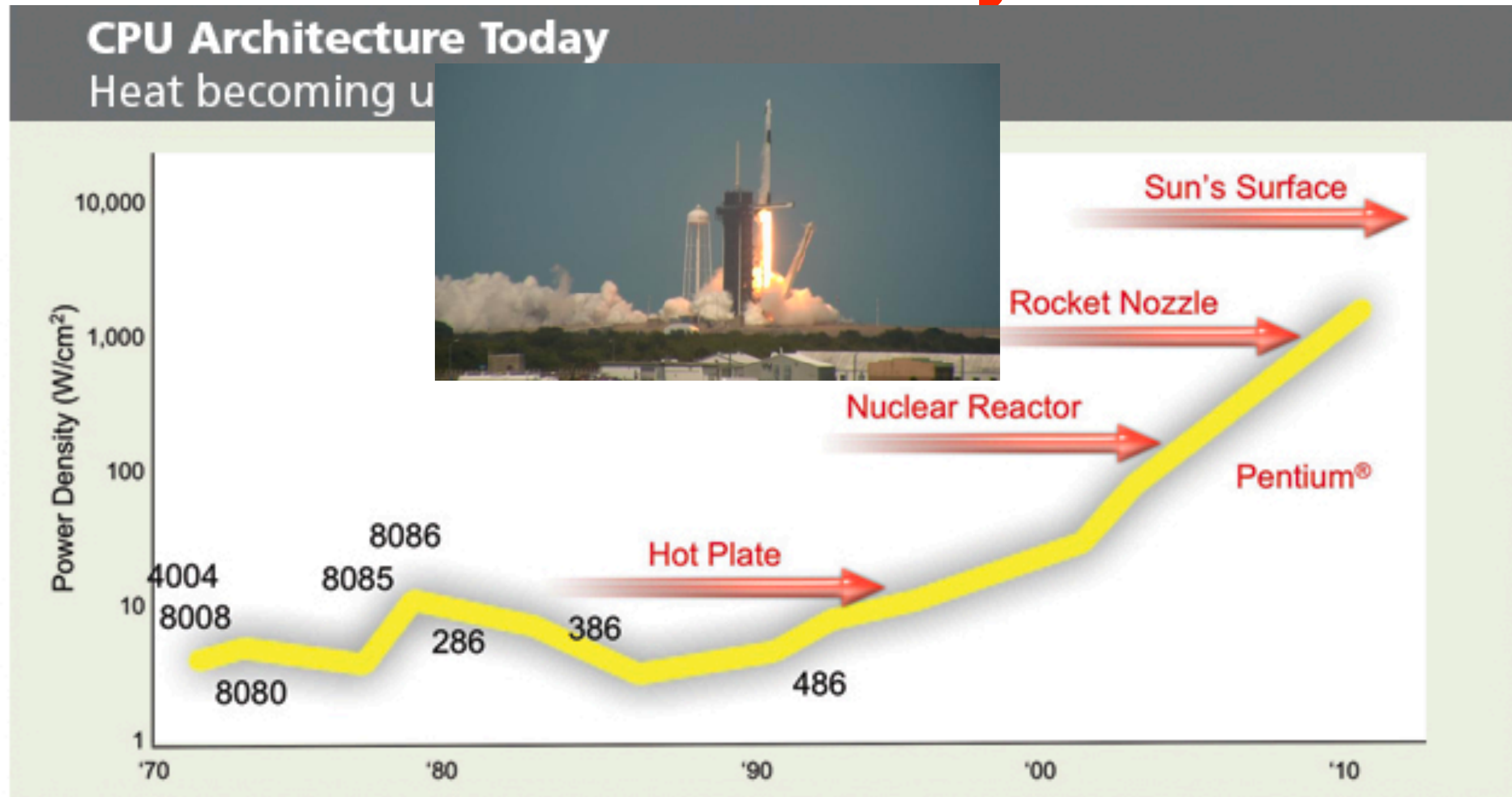
Chip

1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

$$= \frac{100W}{\text{Chip Area}}$$



# Power density



**Figure 1.** In CPU architecture today, heat is becoming an unmanageable problem.  
(Courtesy of Pat Gelsinger, Intel Developer Forum, Spring 2004)  
<https://www.cadalyt.com/hardware/workstation-performance-tomorrow039s-possibilities-viewpoint-column-6351>

# Power consumption to light on all transistors

If we can only cool down 50W in the same area —

**Dennardian Scaling**

**Dennardian Broken**

**Chip**

1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

**=49W**

**Chip**

0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

**=50W**

**Chip**

1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

**On ~  
50W**

**Off ~  
0W**

**Dark!**

**=100W!**

# Dark silicon

- Your power consumption goes up as the number of transistors goes up
- Even Moore's Law allows us to put more transistors within the same area—we cannot use them all simultaneously!
- We have no choice to not activate all transistors at the same time!

# Dynamic/Active Power

- The power consumption due to the switching of transistor states

- Dynamic power per transistor

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

- $\alpha$ : average switches per cycle

- $C$ : capacitance

- $V$ : voltage

- $f$ : frequency, usually linear with  $V$

- $N$ : the number of transistors

# **Solutions/trends in dark silicon era**

# Trends in the Dark Silicon Era

- Aggressive dynamic voltage/frequency scaling
- Throughout oriented — slower, but more
- Just let it dark — activate part of circuits, but not all
- From general-purpose to domain-specific — ASIC

# **Aggressive dynamic frequency scaling**

# Dynamic/Active Power

- The power consumption due to the switching of transistor states

- Dynamic power per transistor

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

- $\alpha$ : average switches per cycle

- $C$ : capacitance

- $V$ : voltage

- $f$ : frequency, usually linear with  $V$

- $N$ : the number of transistors



# Frequency varies per core

Products Solutions Support				intel			
		Intel® Xeon® Processor E7-8890 v4		Intel® Xeon® Processor E7-8893 v4		Intel® Xeon® Processor E7-8880 v4	
Status		Launched		Launched		Launched	
Launch Date ⓘ		Q2'16		Q2'16		Q2'16	
Lithography ⓘ		14 nm		14 nm		14 nm	
Performance							
# of Cores ⓘ		24		4		22	
# of Threads ⓘ		48		8		44	
Processor Base Frequency ⓘ		2.20 GHz		3.20 GHz		2.20 GHz	
Max Turbo Frequency ⓘ		3.40 GHz		3.50 GHz		3.30 GHz	
Cache ⓘ		60 MB		60 MB		55 MB	
Bus Speed ⓘ		9.6 GT/s		9.6 GT/s		9.6 GT/s	
# of QPI Links ⓘ		3		3		3	
TDP ⓘ		165 W		140 W		150 W	


# Demo

- You may use `cat /proc/cpuinfo` to see all the details of your processors
- You may add `"| grep MHz"` to see the frequencies of your cores
- Only very few of them are on the boosted frequency

**Slower, but more**

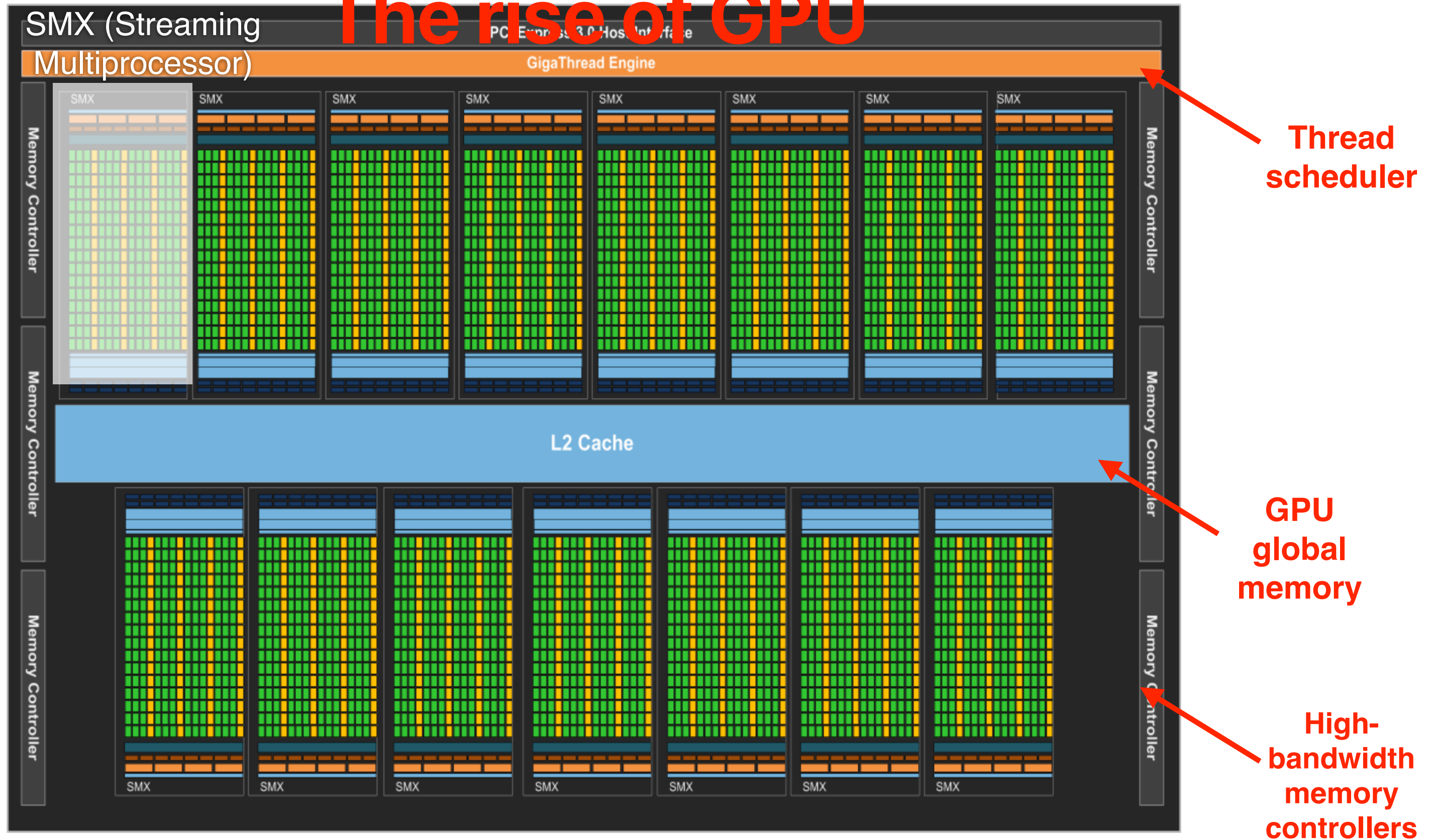
# More cores per chip, slower per core

Products Solutions Support



	Intel® Xeon® Processor E7-8890 v4	Intel® Xeon® Processor E7-8893 v4	Intel® Xeon® Processor E7-8880 v4
Status	Launched	Launched	Launched
Launch Date ⓘ	Q2'16	Q2'16	Q2'16
Lithography ⓘ	14 nm	14 nm	14 nm
Performance			
# of Cores ⓘ	24	4	22
# of Threads ⓘ	48	8	44
Processor Base Frequency ⓘ	2.20 GHz	3.20 GHz	2.20 GHz
Max Turbo Frequency ⓘ	3.40 GHz	3.50 GHz	3.30 GHz
Cache ⓘ	60 MB	60 MB	55 MB
Bus Speed ⓘ	9.6 GT/s	9.6 GT/s	9.6 GT/s
# of QPI Links ⓘ	3	3	3
TDP ⓘ	165 W	140 W	150 W

# The rise of GPU



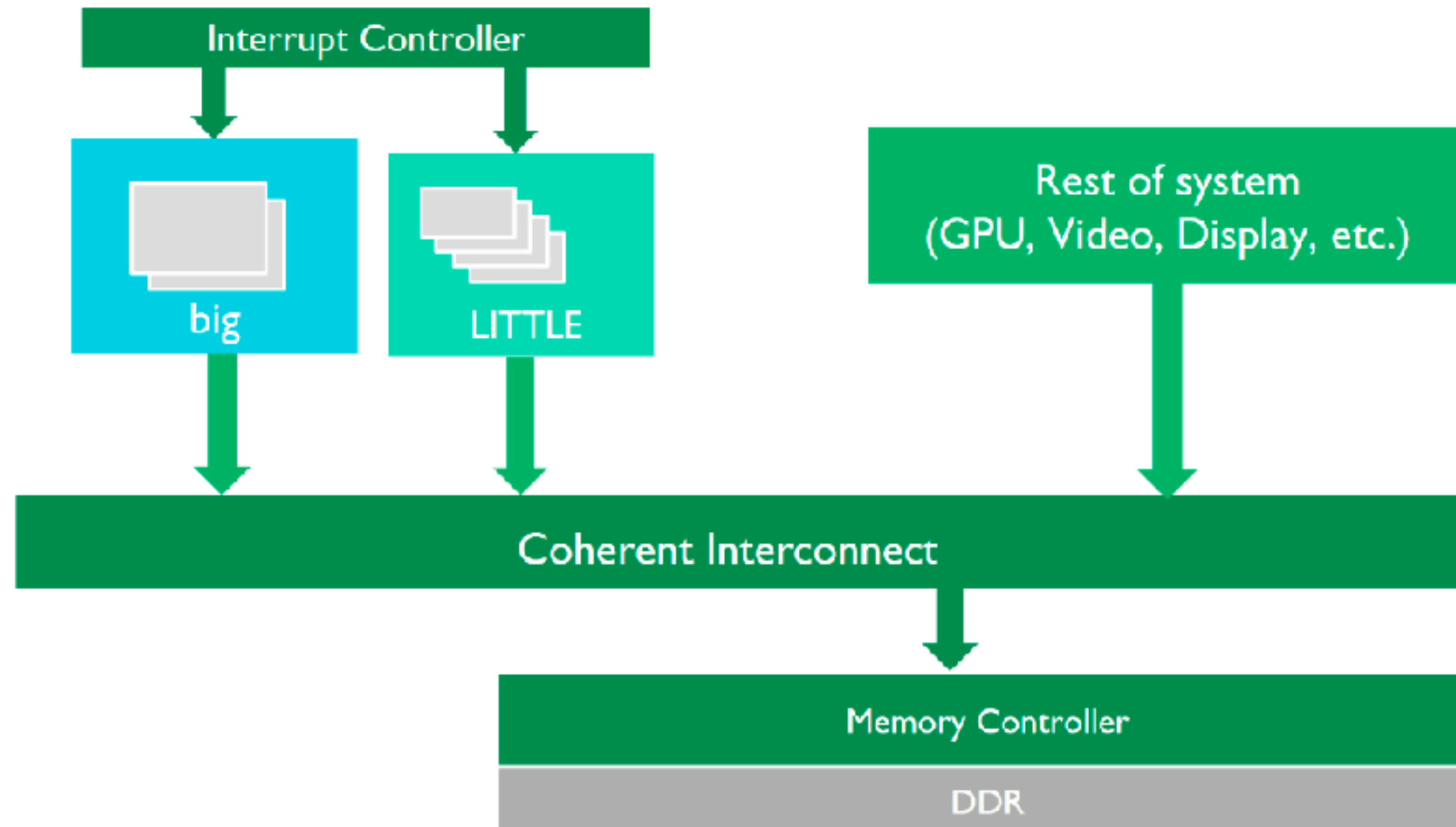


Each of these performs the same operation, but each of these is also a "thread"



# ARM's big.LITTLE architecture

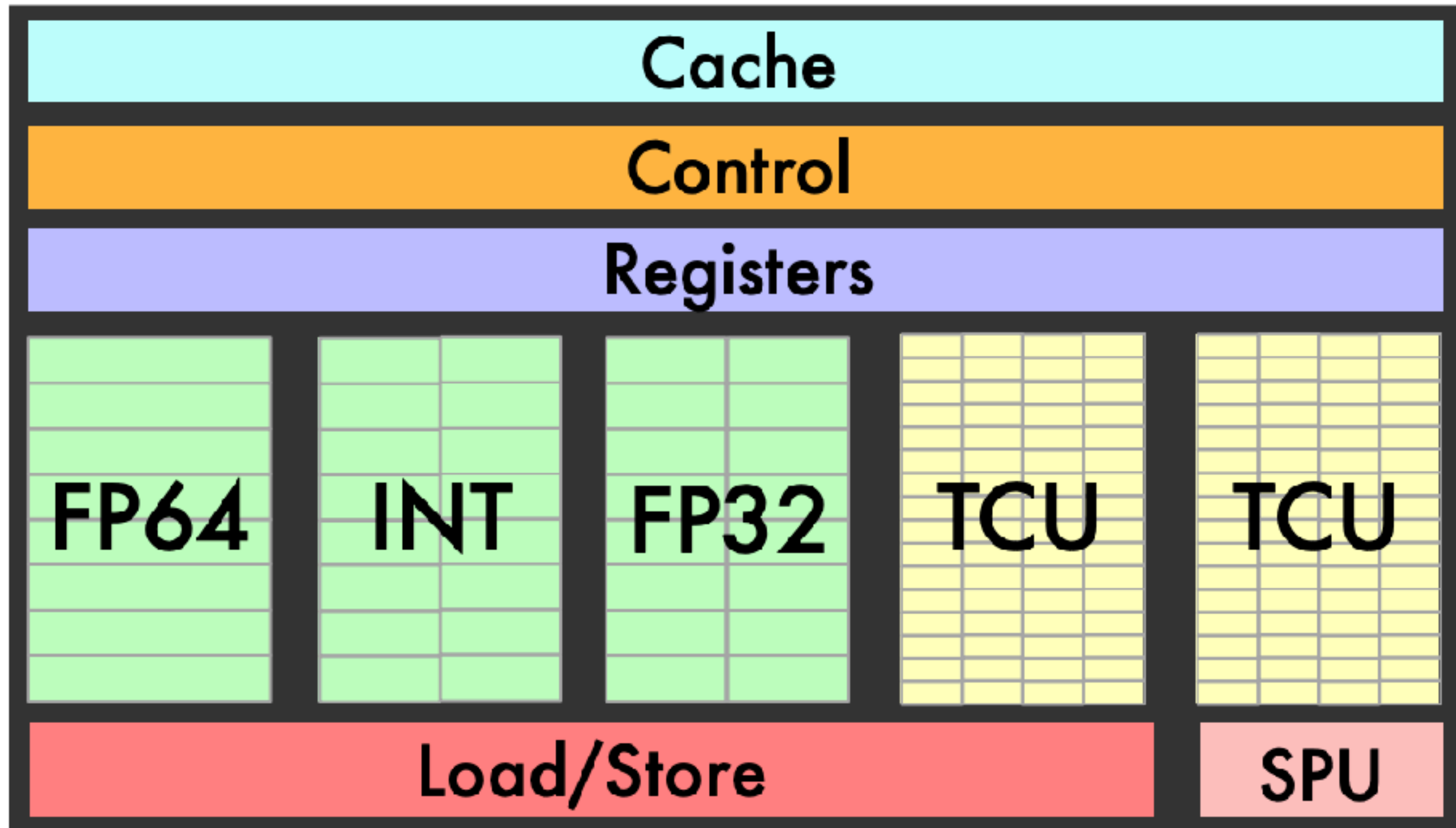
big.LITTLE system



**Just let it dark**



# NVIDIA's Turing Architecture



# Programming in Turing Architecture

Use tensor cores

```
cublasErrCheck(cublasSetMathMode(cublasHandle, CUBLAS_TENSOR_OP_MATH));
```

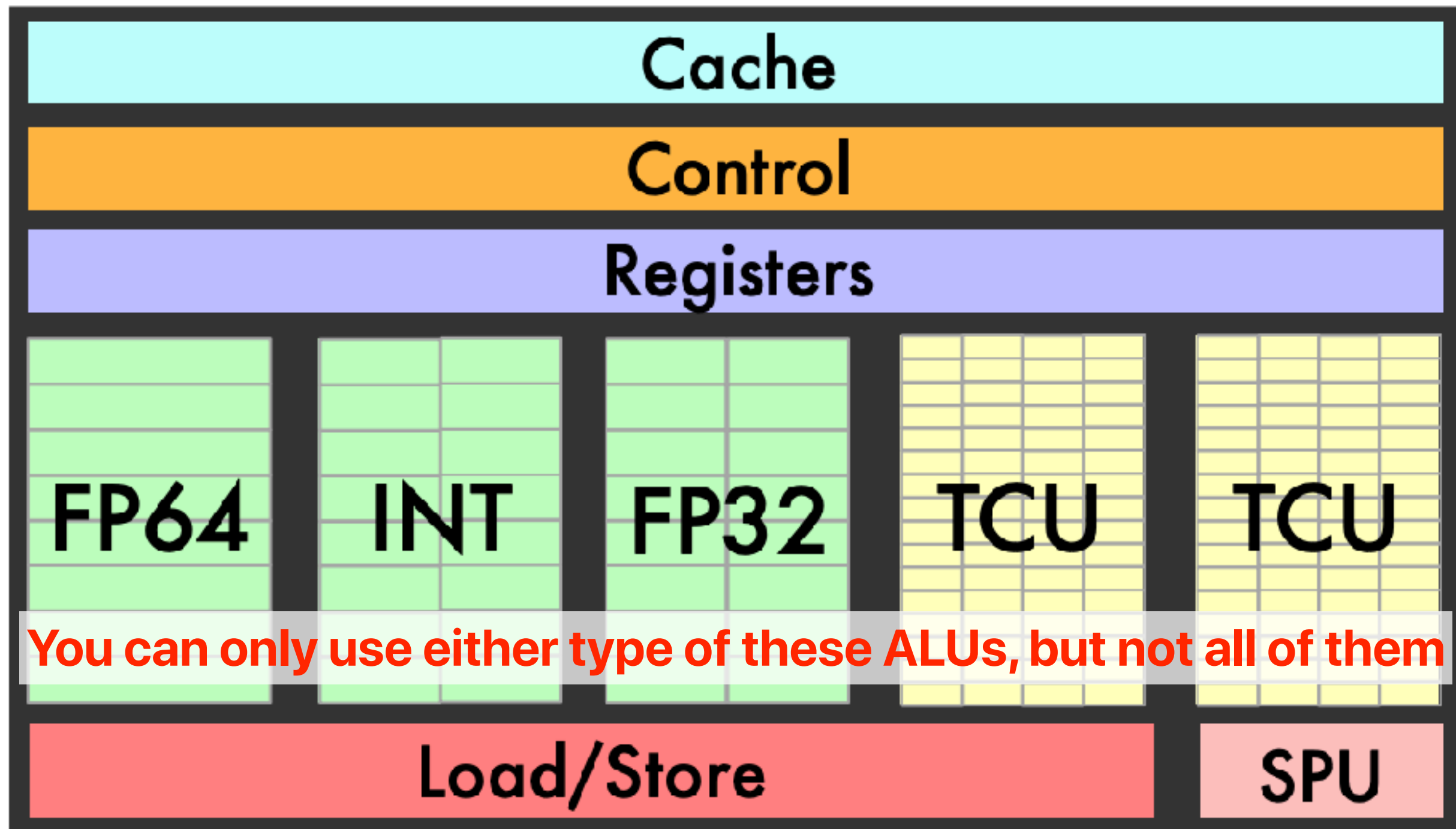
Make them 16-bit

```
convertFp32ToFp16 <<< (MATRIX_M * MATRIX_K + 255) / 256, 256 >>> (a_fp16, a_fp32,  
MATRIX_M * MATRIX_K);  
    convertFp32ToFp16 <<< (MATRIX_K * MATRIX_N + 255) / 256, 256 >>> (b_fp16, b_fp32,  
MATRIX_K * MATRIX_N);
```

```
cublasErrCheck(cublasGemmEx(cublasHandle, CUBLAS_OP_N, CUBLAS_OP_N,  
    MATRIX_M, MATRIX_N, MATRIX_K,  
    &alpha,  
    a_fp16, CUDA_R_16F, MATRIX_M,  
    b_fp16, CUDA_R_16F, MATRIX_K,  
    &beta,  
    c_cublas, CUDA_R_32F, MATRIX_M,  
    CUDA_R_32F, CUBLAS_GEMM_DFALT_TENSOR_OP));
```

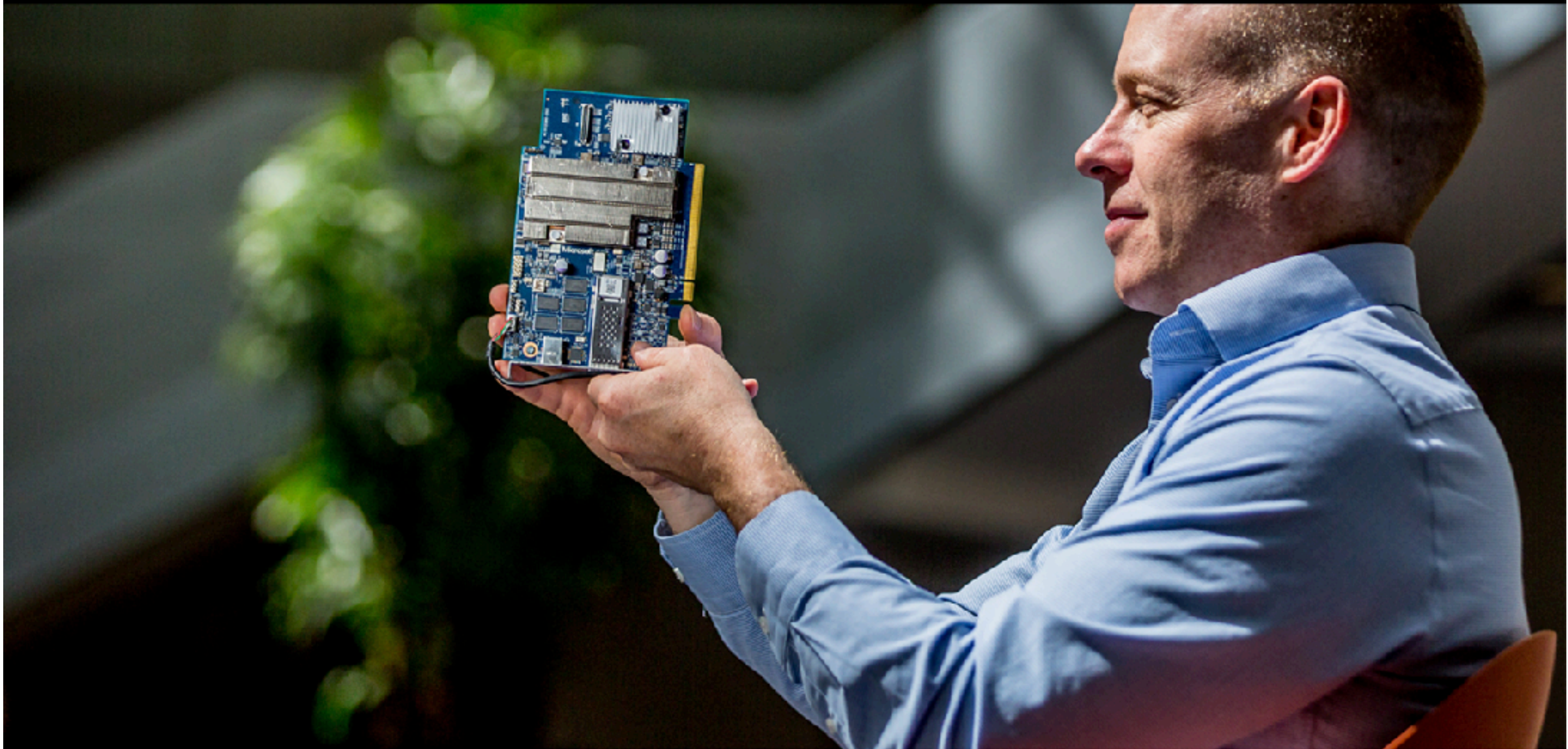
call Gemm

# NVIDIA's Turing Architecture



# The rise of ASICs





Real-time AI: Microsoft announces preview of Project Brainwave



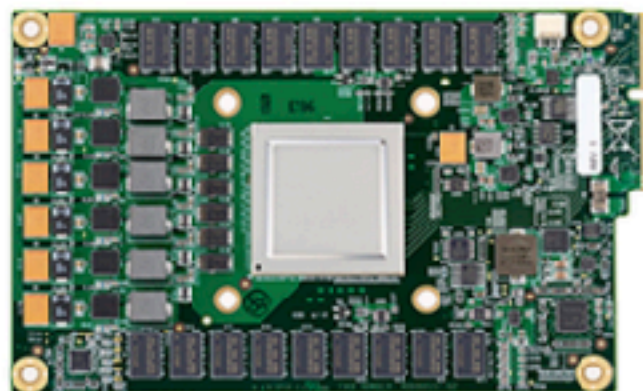
## AI &amp; MACHINE LEARNING

# An in-depth look at Google's first Tensor Processing Unit (TPU)

**Kaz Sato**Staff Developer Advocate,  
Google Cloud**Cliff Young**Software Engineer, Google  
Brain**David Patterson**Distinguished Engineer, Google  
Brain

May 12, 2017

There's a common thread that connects Google services such as Google Search, Street View, Google Photos and Google Translate: they all use Google's Tensor Processing Unit, or TPU, to accelerate their neural network computations behind the scenes.

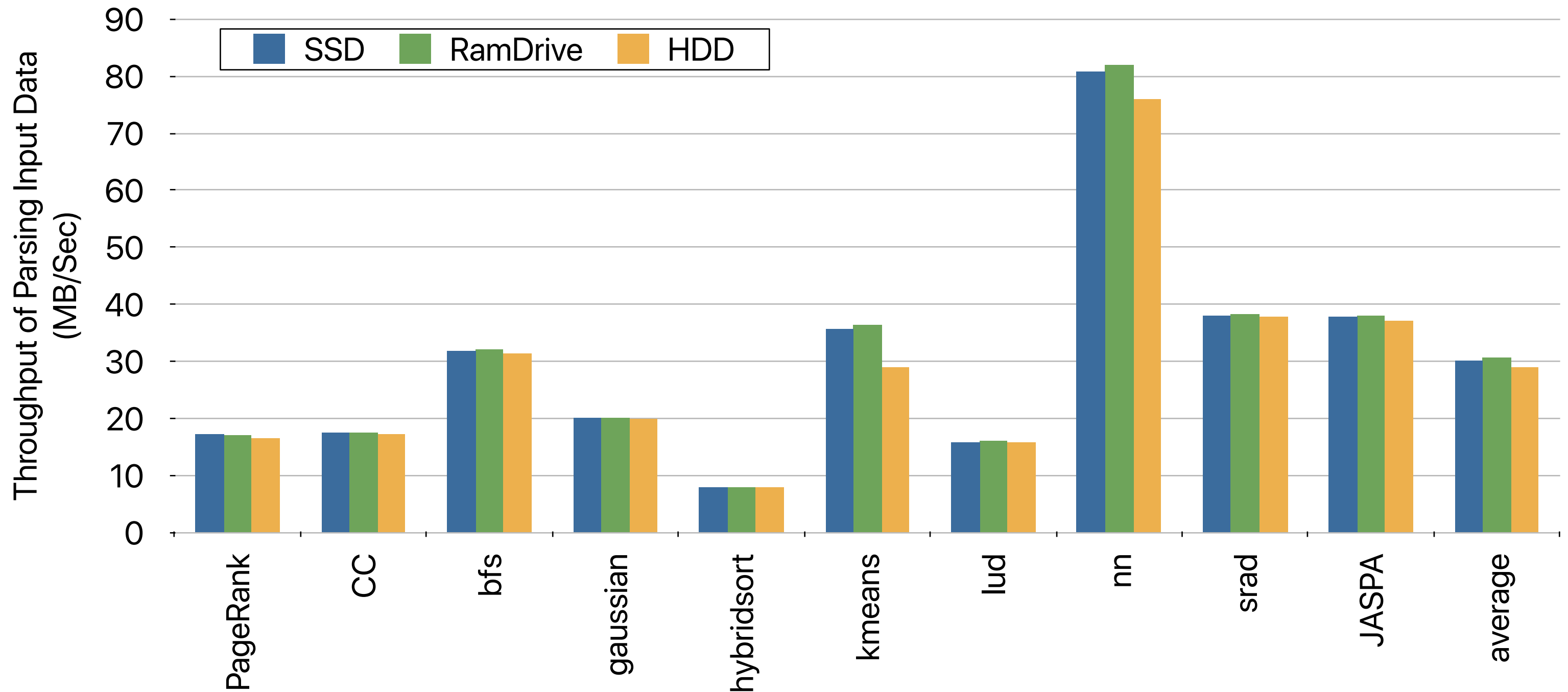


Google's first Tensor Processing Unit (TPU) on a printed circuit board (left); TPUs deployed in a Google datacenter (right)

**Try GCP**Get \$300 free credit to spend  
over 12 months.

# **Data storage matters**

# ASCII to Binary doesn't scale with I/O bandwidth





# Demo

There is no pure “software” or “hardware” design in the dark silicon era.  
Everything needs to be hardware/software co-designed.

*–Prof. Usagi*

# Announcement

- iEval — Capture your screenshot, submit through iLearn and you will receive a full credit assignment
- Lab 6 due tonight
- Assignment 6 due 6/4
- Final exam will be held during the campus scheduled period to avoid conflicts
  - Final review — 6/4 during the lecture, will also release the sample final
  - 6/11 11:30am — 2:59:59pm
  - About the same format as midterm, but longer
  - Will have a final review on 6/6 to help you prepare

# Electrical Computer Science Engineering

# 120A

# つづく

