# Dark Silicon – a currency we do not control

*Holger Pirk <hlgr@ic.ac.uk>*

# Context

I am talking about CPUs and GPUs

I am talking about CPUs and GPUs, not FPGAs

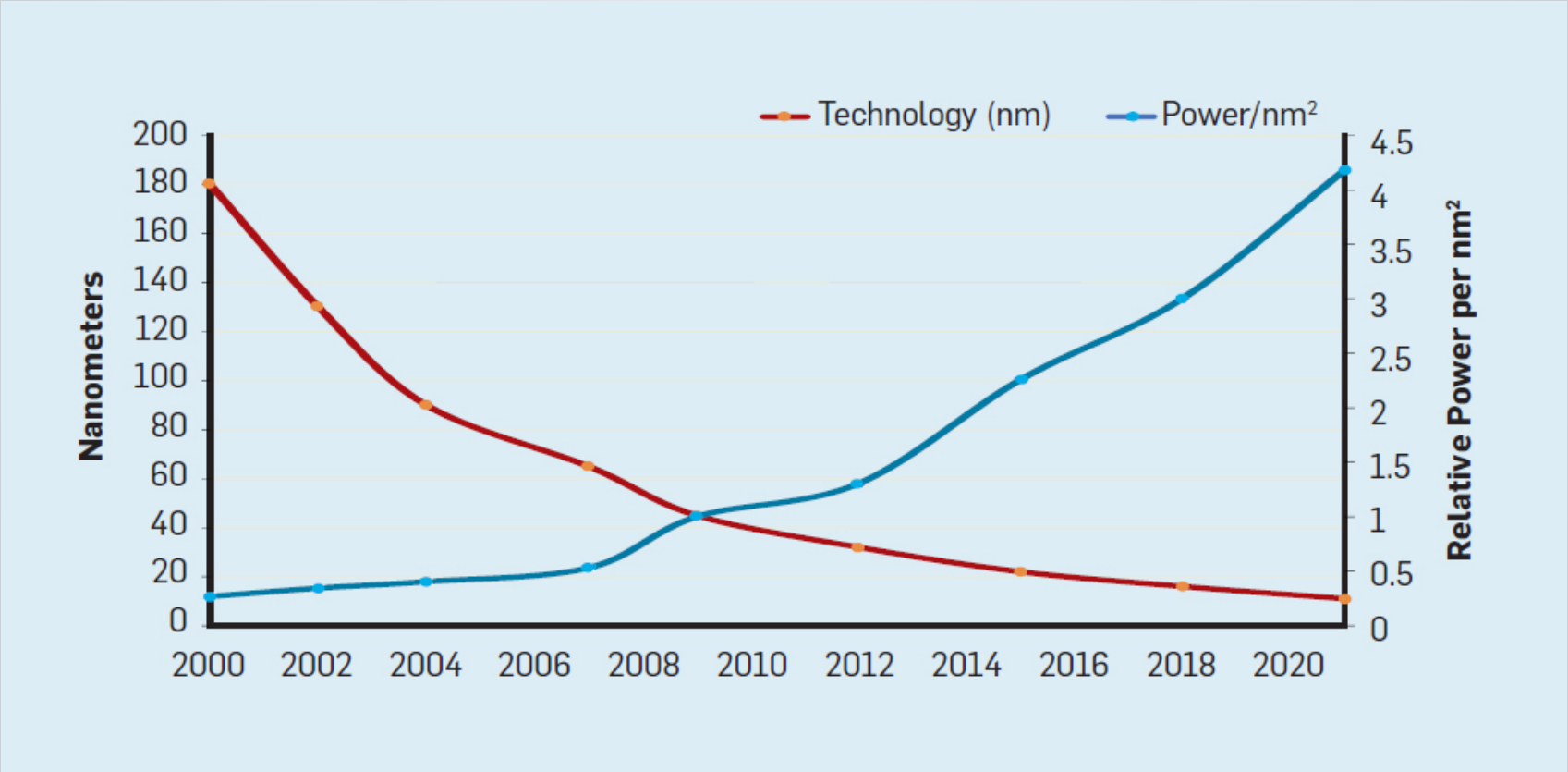
# Premise

# Transistors are getting smaller!

Good! Dennard scaling means that smaller transistors use **proportionally** less energy

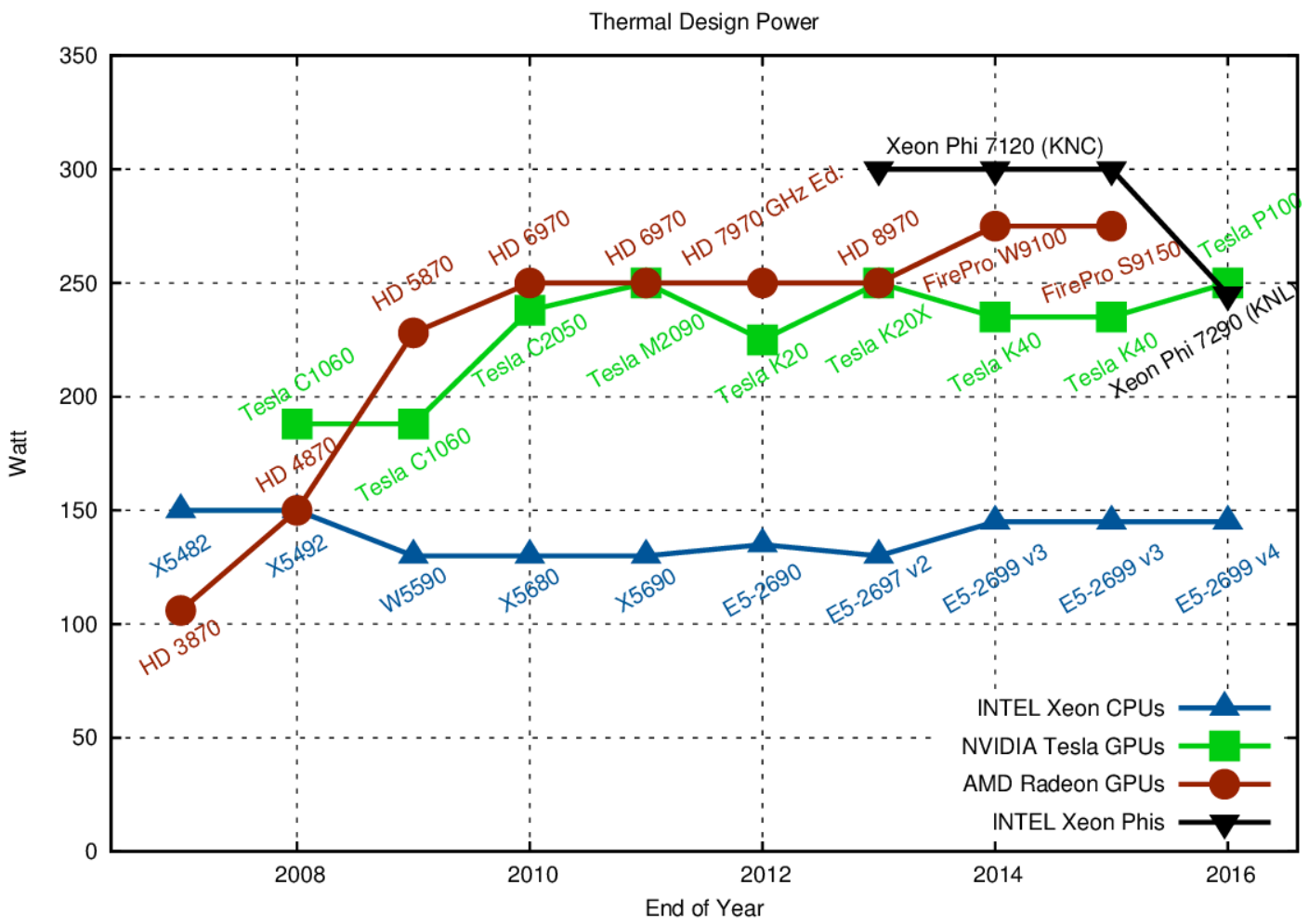
Well, that was in the good old days

# Today: More transistors per $nm^2$, more power consumption per $nm^2$



Okay, so they are using more power !?!

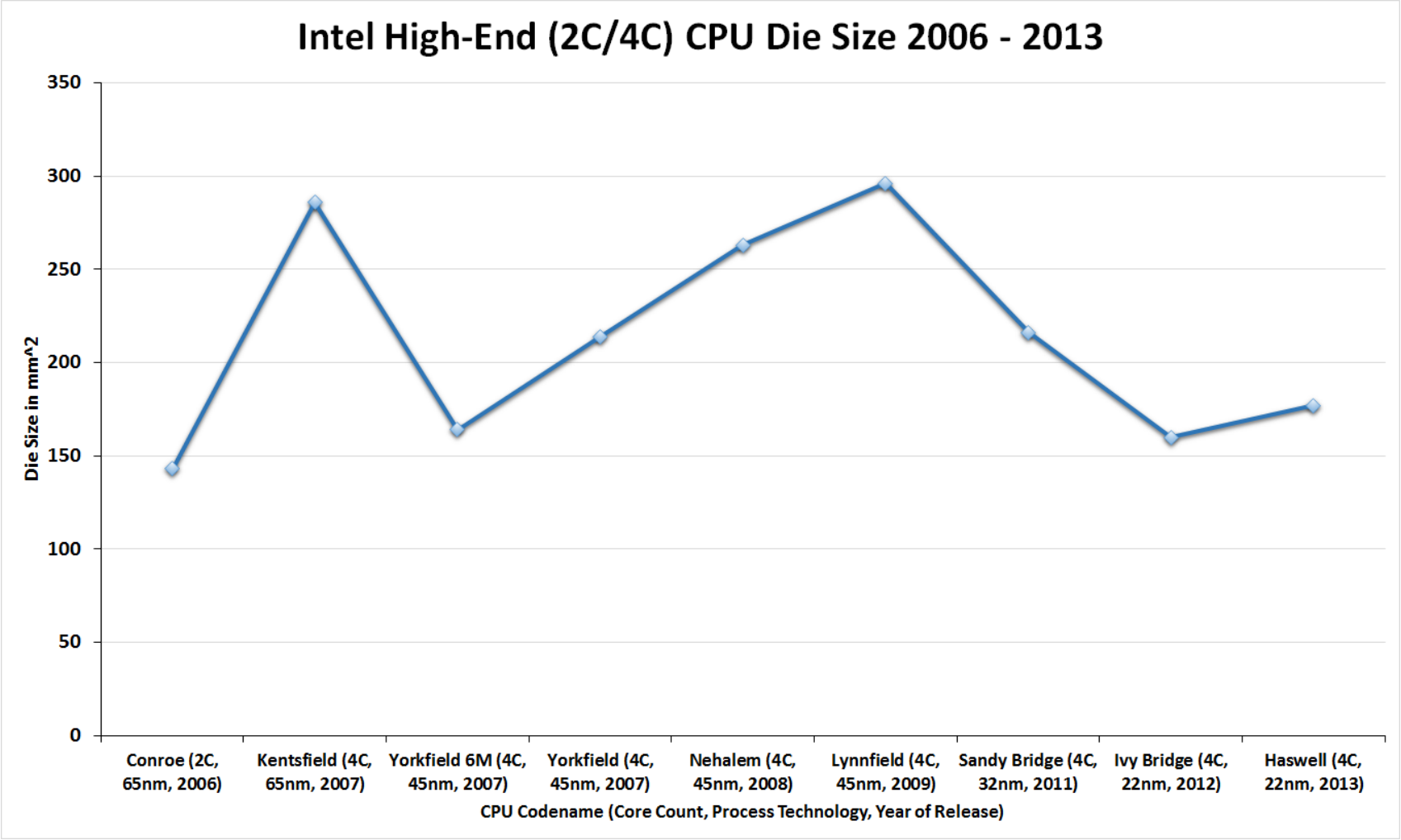
# But how does the TDP of chips stays constant?




[Karl Rupp, CPU, GPU and MIC Hardware Characteristics over Time]

Chips must be getting smaller!

# Dies stay roughly the same size

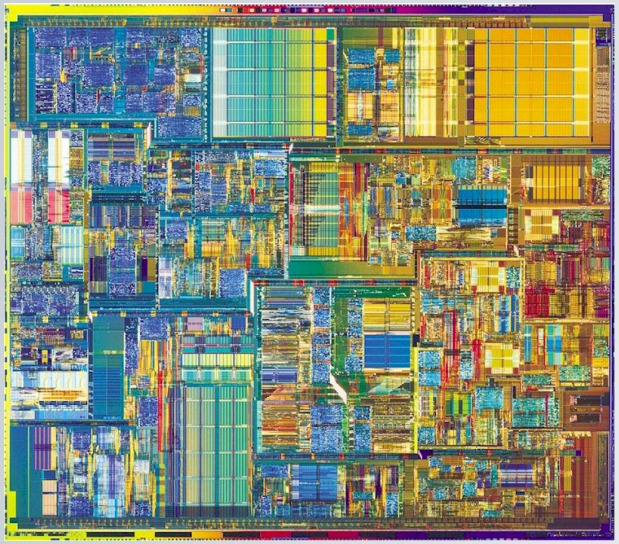


Okay, something has to give...

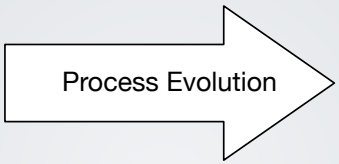
. . . chips are getting hottter!

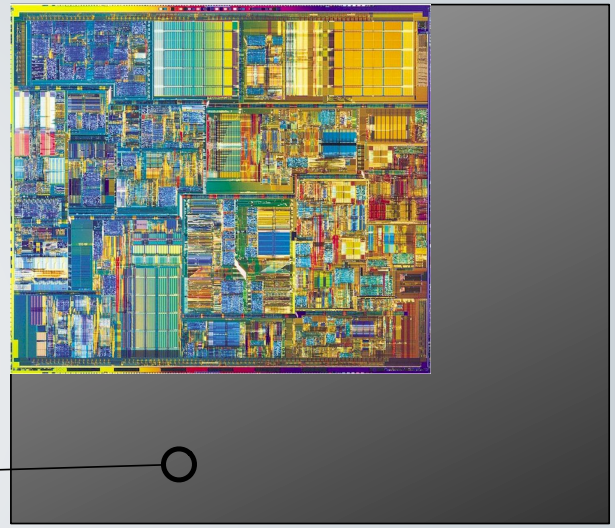
If chips are getting hotter overall. . .

# . . . parts of the chip need to go dim (clocked down or off)!



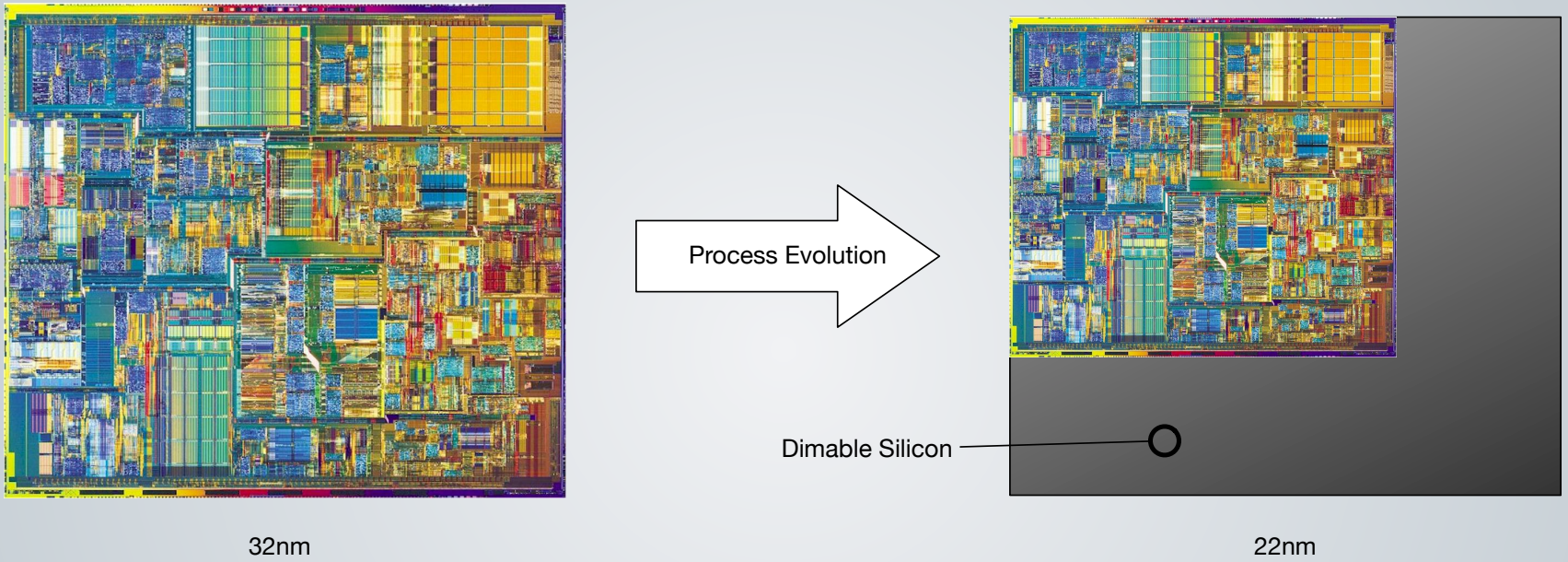
32nm



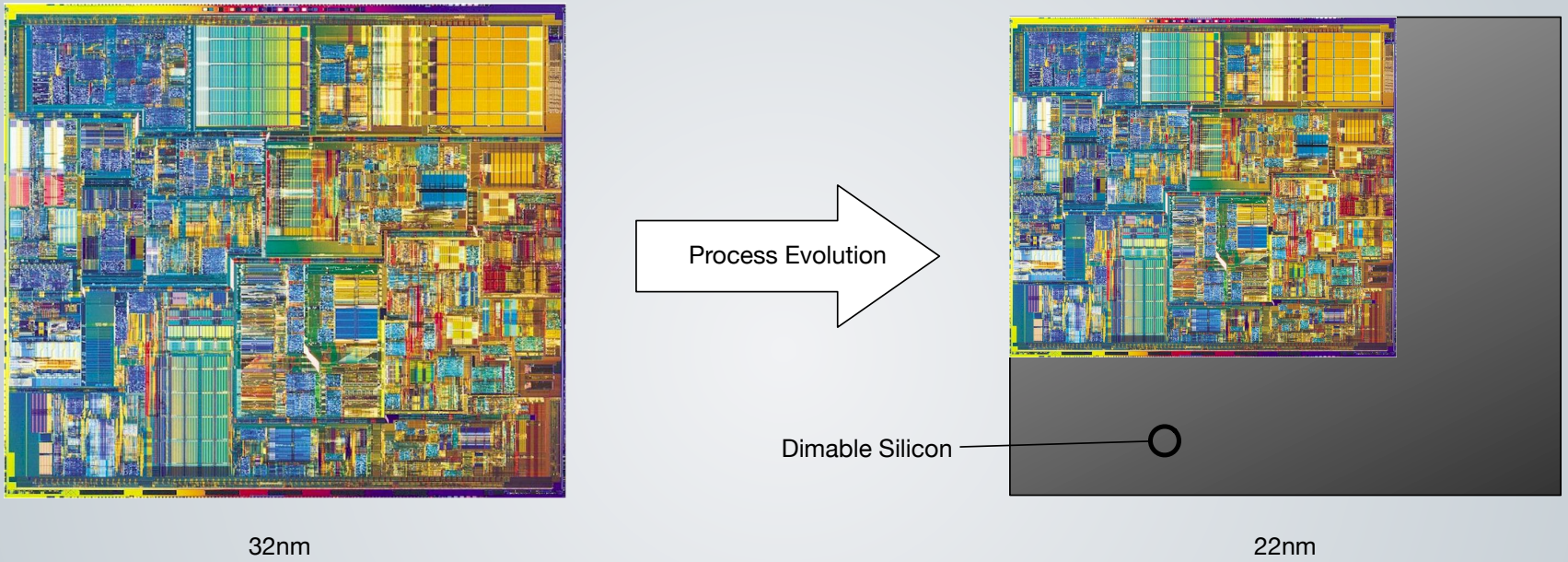

22nm

# . . . parts of the chip need to go dim (clocked down or off)!



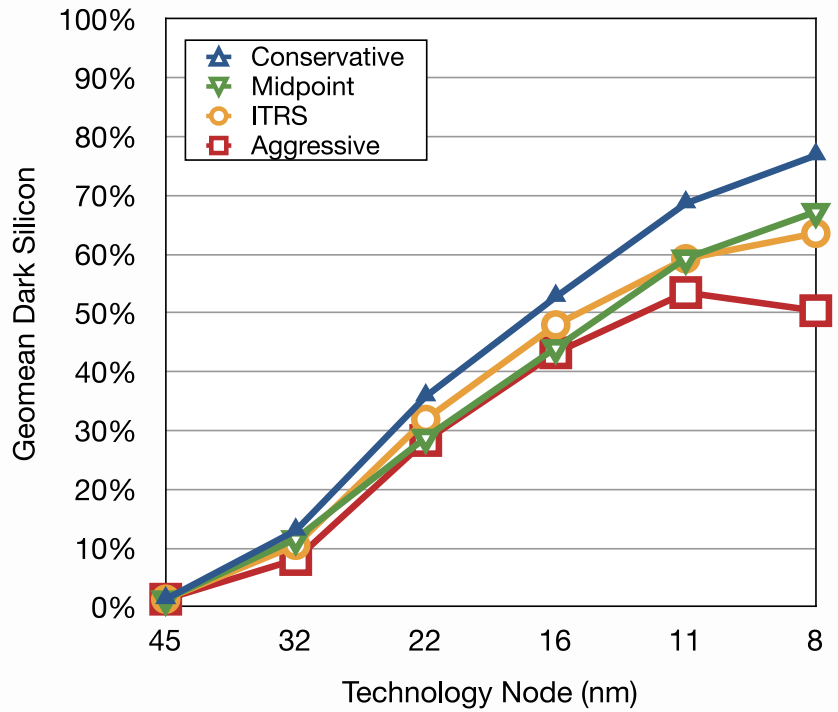

- Let's call these bits "dimable silicon"

# . . . parts of the chip need to go dim (clocked down or off)!



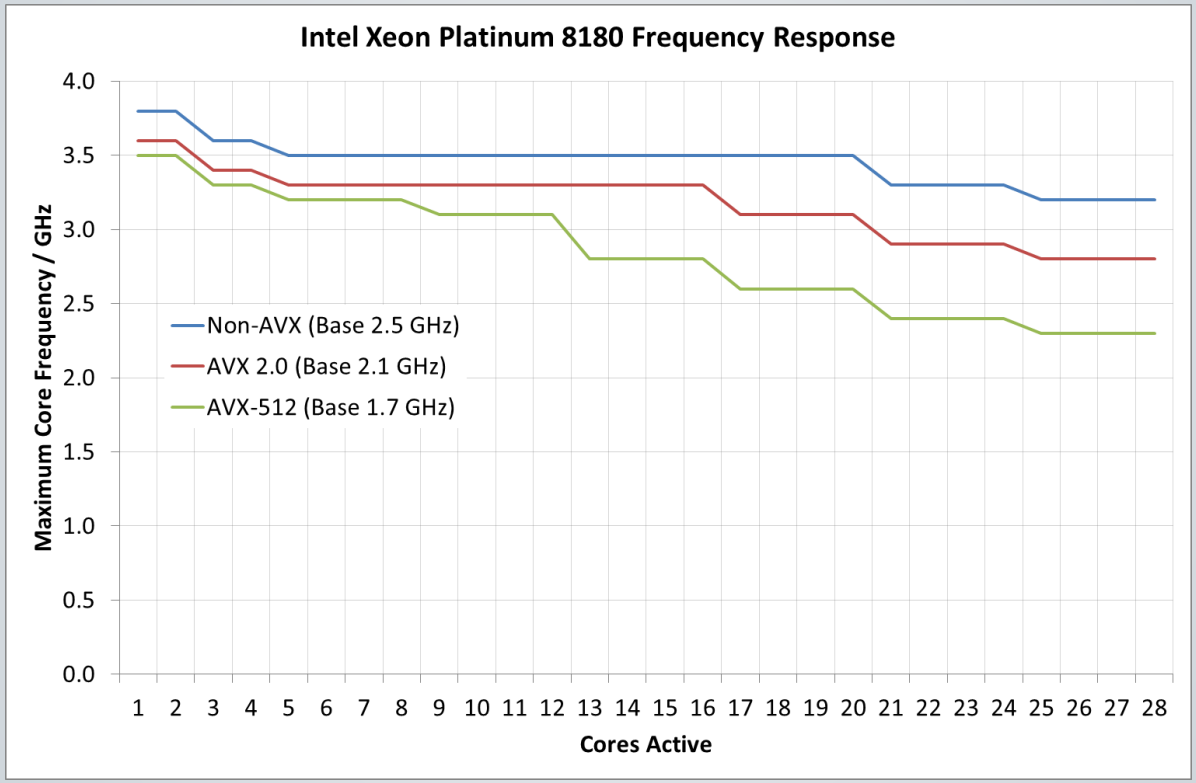

- Let's call these bits "dimable silicon"
- Parts that are inactive at runtime are called "dark"

# Dark silicon in theory...




- Projections for CPU-like cores
- [Esmaeilzadeh et al., TOCS '12]

# Dim silicon in practice...



What functionality should be implemented in dimable silicon?

More cores?

# More cores?

# More cores?

- you can only have so much of the chip active

# More cores?

- you can only have so much of the chip active
- what use are 40 **identical** cores if you can only use 4 at a time

Okay, not cores!

Functional units, then!

Which of the following should be implemented in dimable silicon?

# Which of the following should be implemented in dimable silicon?

The instruction scheduler Schedule instructions to the ALU

# Which of the following should be implemented in dimable silicon?

The instruction scheduler Schedule instructions to the ALU

The SIMD unit Process multiple floats with one instruction

# Which of the following should be implemented in dimable silicon?

The instruction scheduler Schedule instructions to the ALU

The SIMD unit Process multiple floats with one instruction

A sort-merge-join accelerator Make SM-Joins go 80% faster


# Which of the following should be implemented in dimable silicon?

The instruction scheduler Schedule instructions to the ALU

The SIMD unit Process multiple floats with one instruction

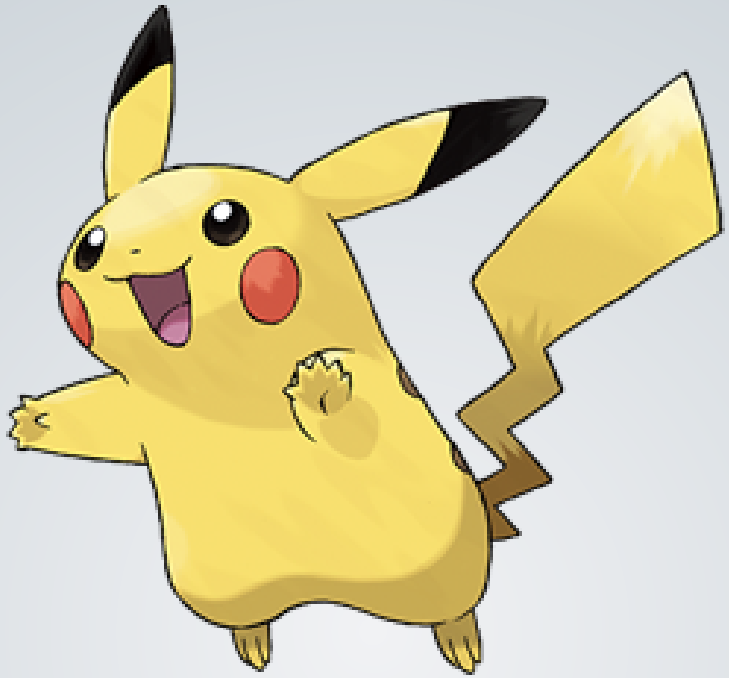
A sort-merge-join accelerator Make SM-Joins go 80% faster

A pokeball support unit Draw a pokeball on your screen in 42 cycles

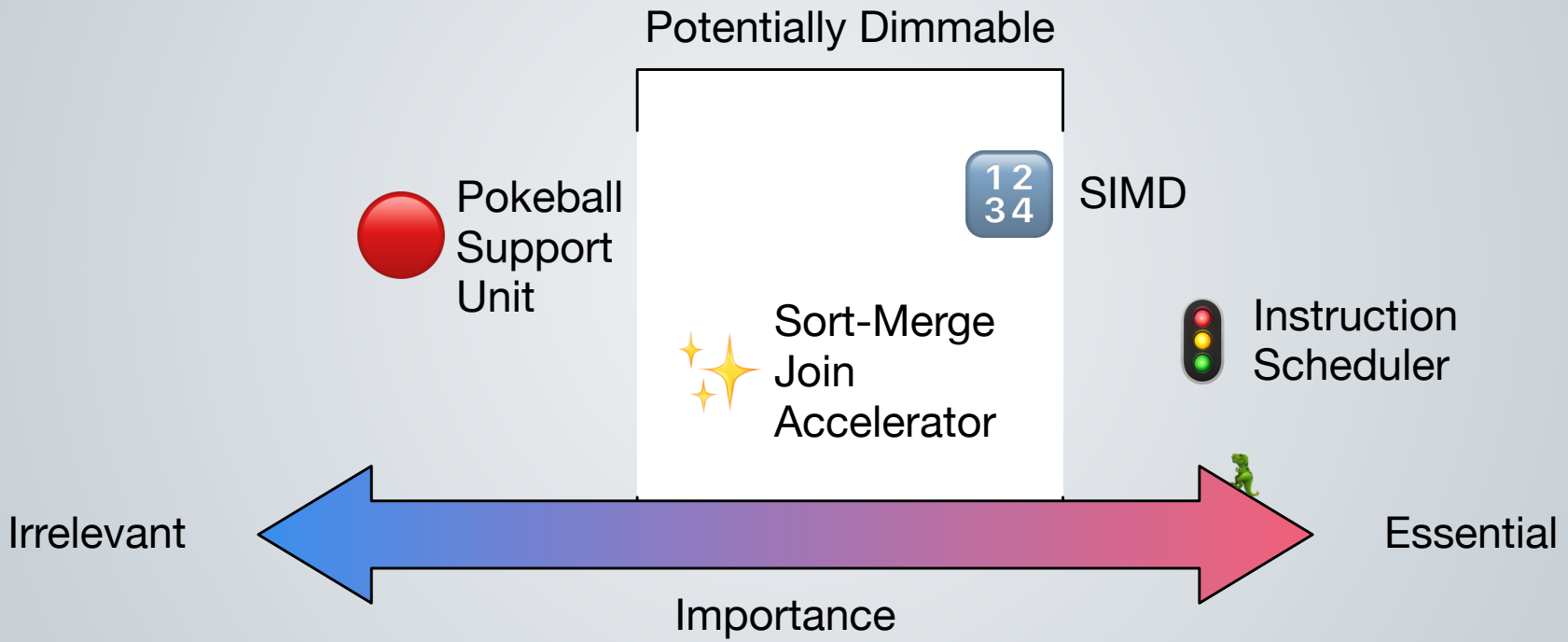



You don't know what a pokeball is?

I envy you your life!



# What functionality should be implemented in dimable silicon?



Should Data Management be in dimable Silicon?

# What is Data Management (in terms of hardware primitives)

- Joins (sort-merge, hash, radix)

# What is Data Management (in terms of hardware primitives)

- Joins (sort-merge, hash, radix)
- Index structures (trees, bloom Filters)

# What is Data Management (in terms of hardware primitives)

- Joins (sort-merge, hash, radix)
- Index structures (trees, bloom Filters)
- Graph algorithms (triangle counting)

# What is Data Management (in terms of hardware primitives)

- Joins (sort-merge, hash, radix)
- Index structures (trees, bloom Filters)
- Graph algorithms (triangle counting)
- Incremental aggregation (stream processing)

# What is Data Management (in terms of hardware primitives)

- Joins (sort-merge, hash, radix)
- Index structures (trees, bloom Filters)
- Graph algorithms (triangle counting)
- Incremental aggregation (stream processing)
- . . . you probably have your own ideas

# Who gets to define importance?

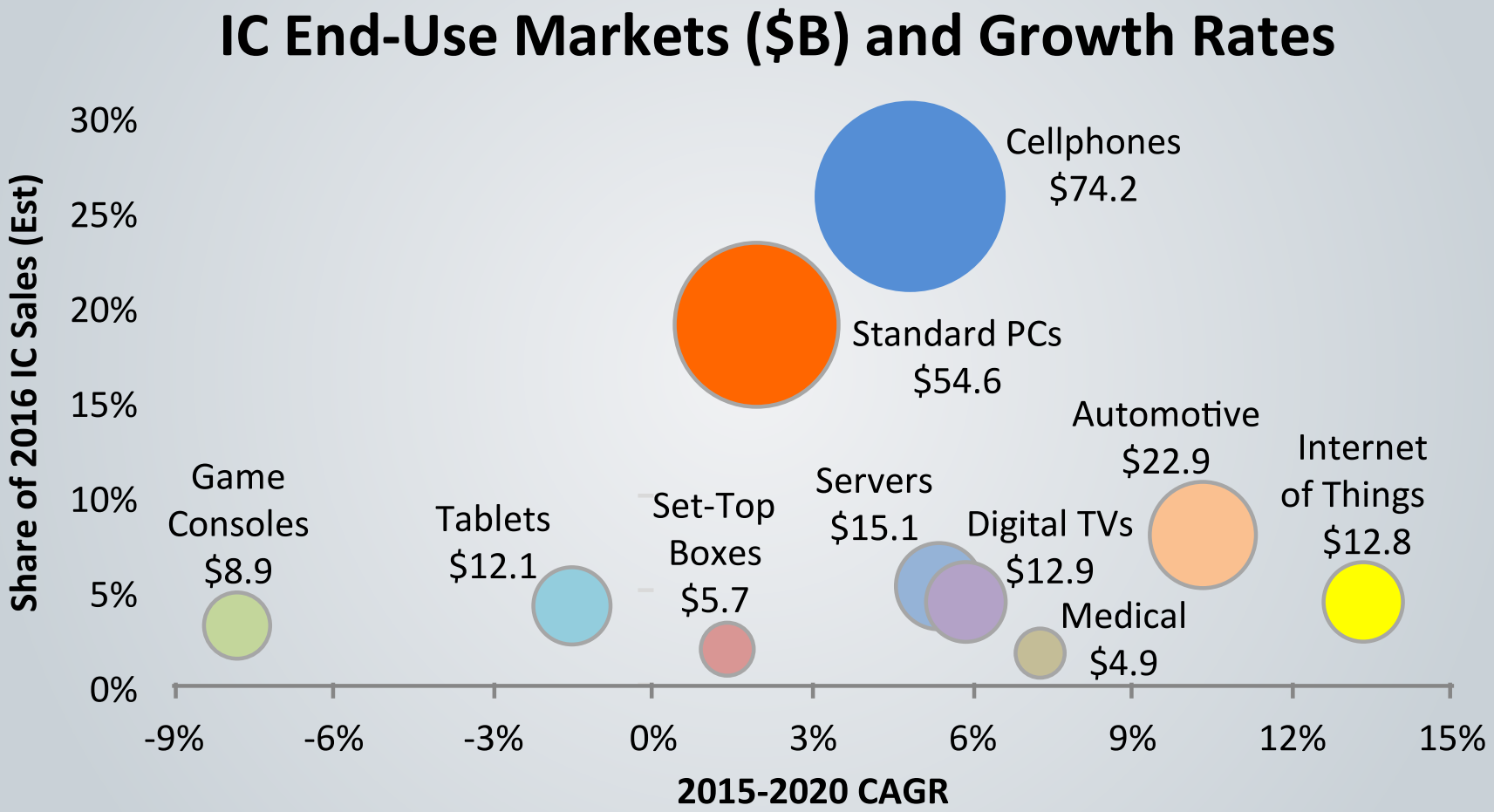
# Who gets to define importance?

*As in all systems, economics is the key part of the objective function that determine[s] design.*
*— Gordon Bell*

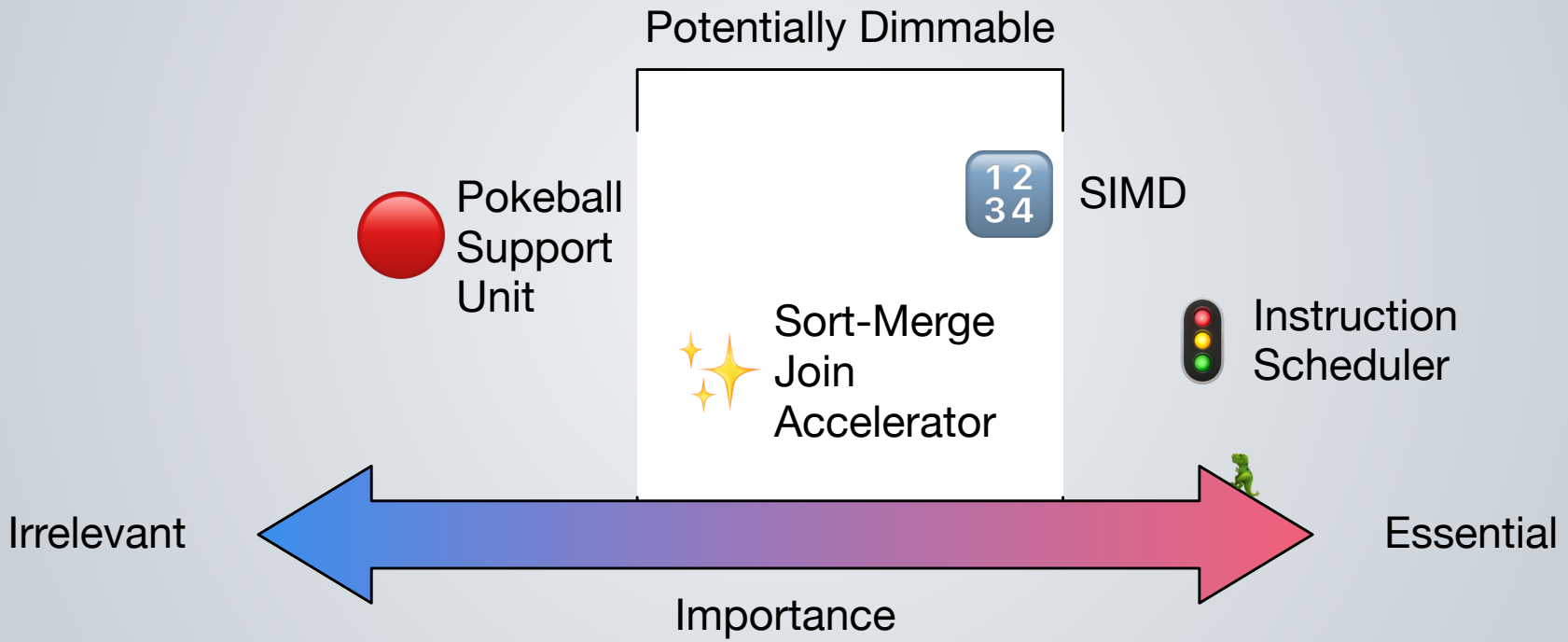
I looked at a lot of economics statistics to prepare this talk

(I have the best numbers! Trust me!)

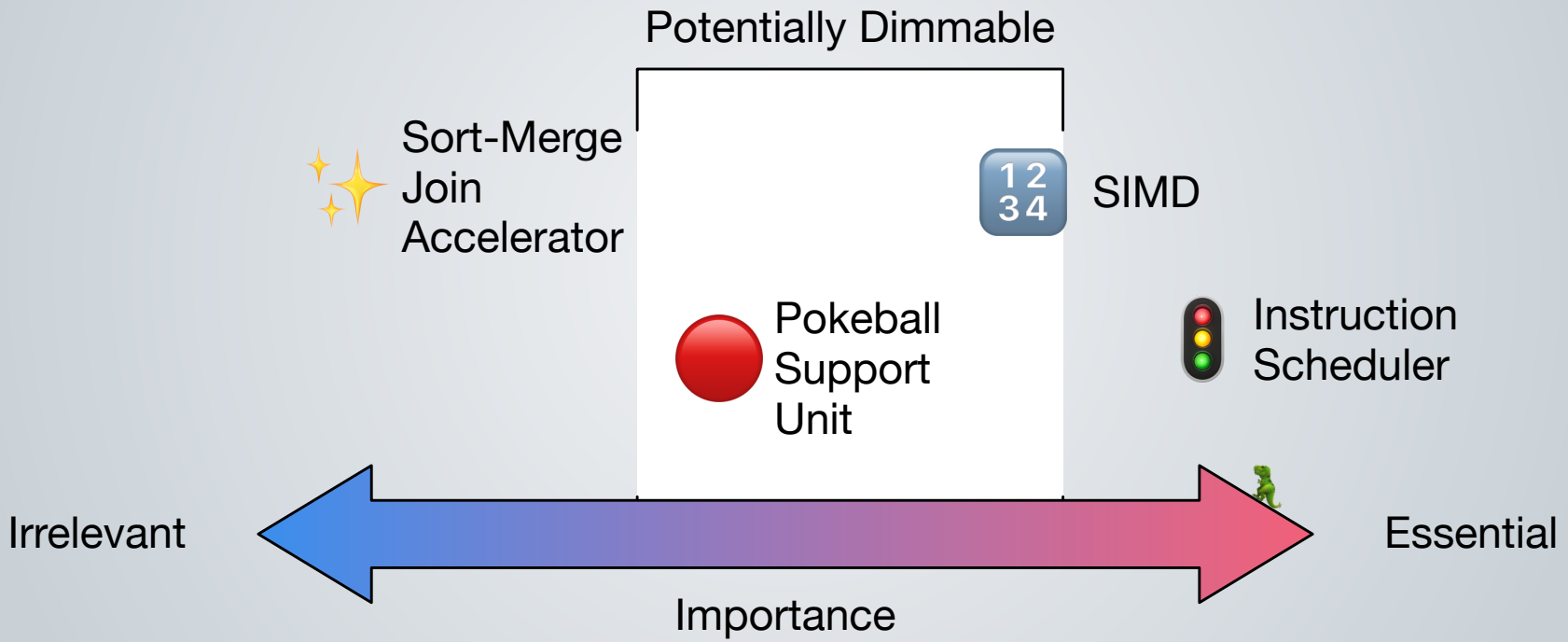
# Demand for microchips is not in servers



# What functionality should be implemented in dimable silicon?



# What functionality should be implemented in dimable silicon?



Well fine, but vendors could specialize chips for servers

They **could**...

Let's look at three potential data center players:

- Intel
- Nvidia
- ARM

Intel

# Intel's new flagship

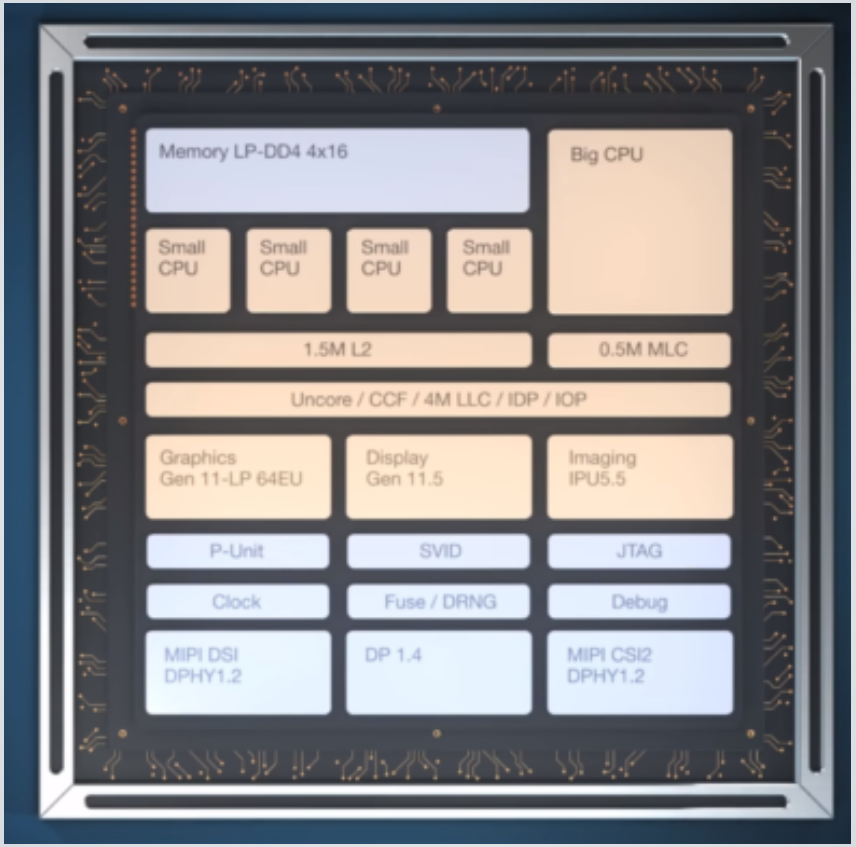
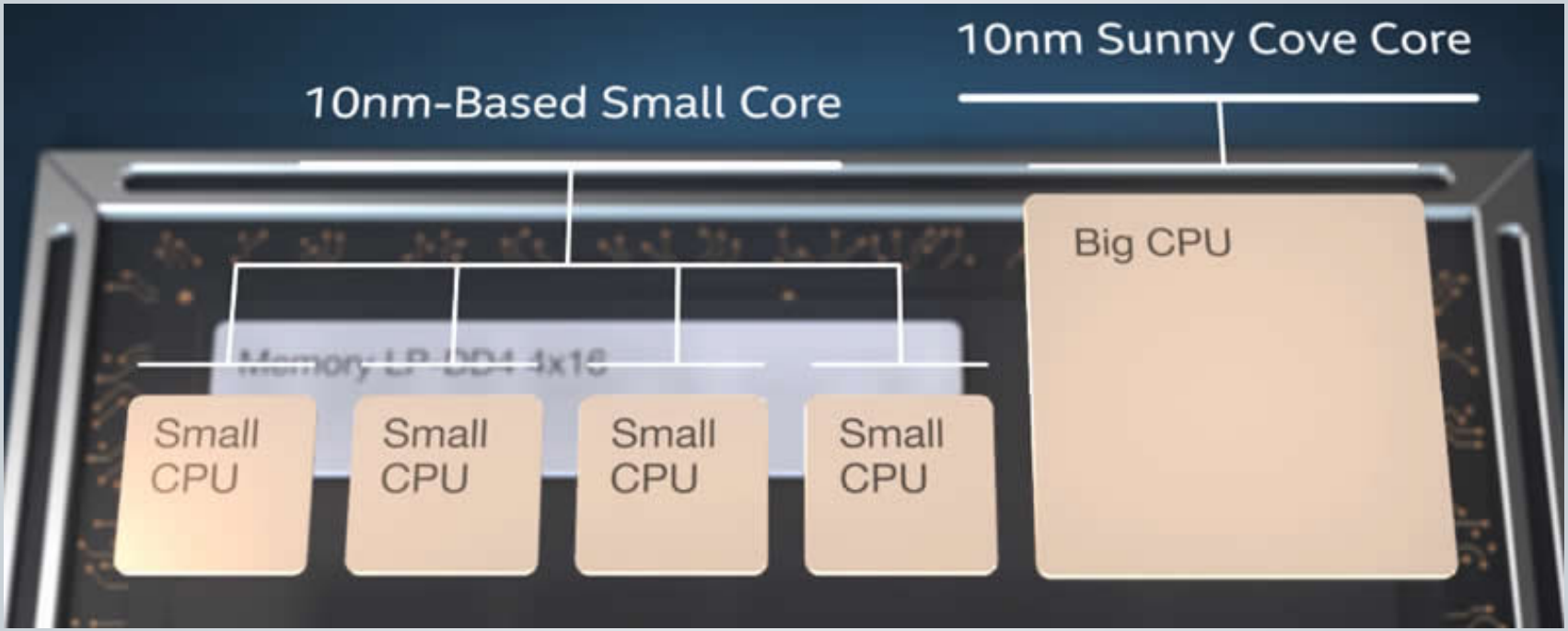
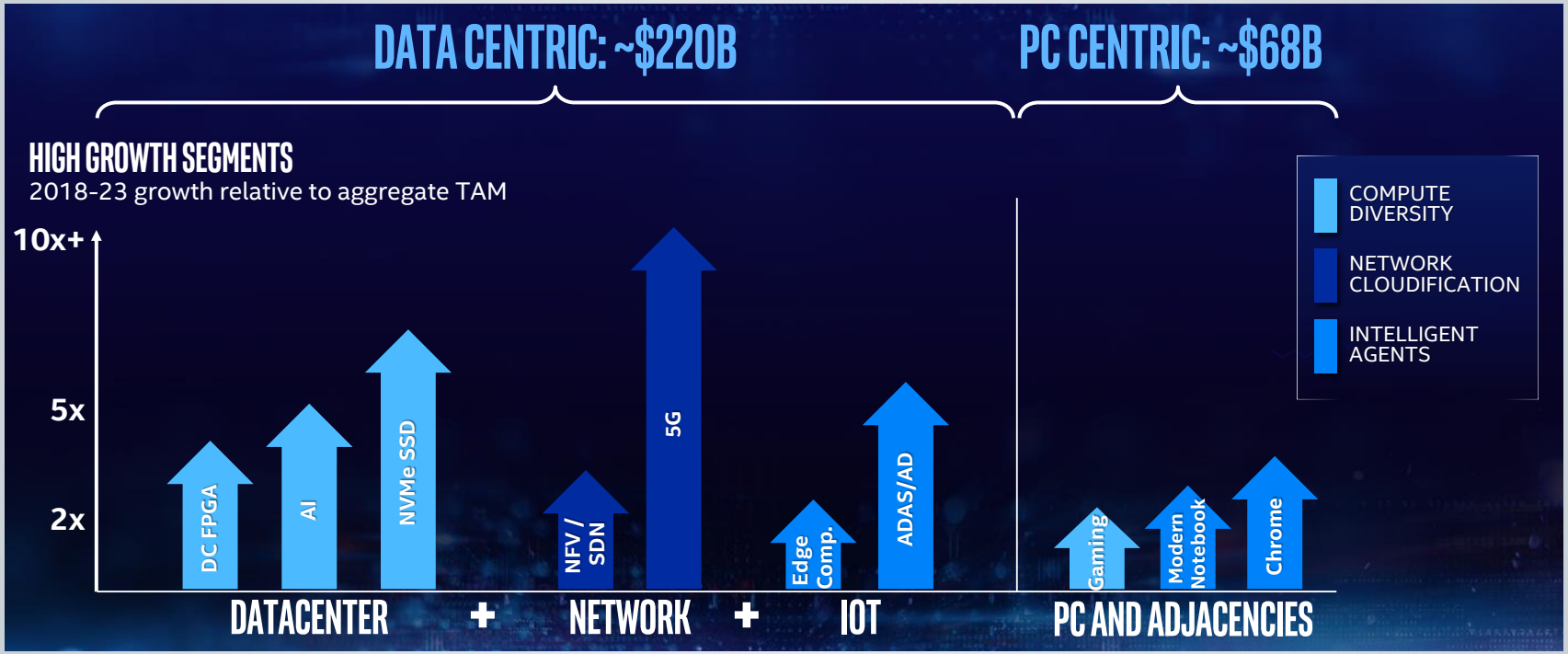



Figure: Lake Field Architecture

# Intel's new flagship



# What is it good for?



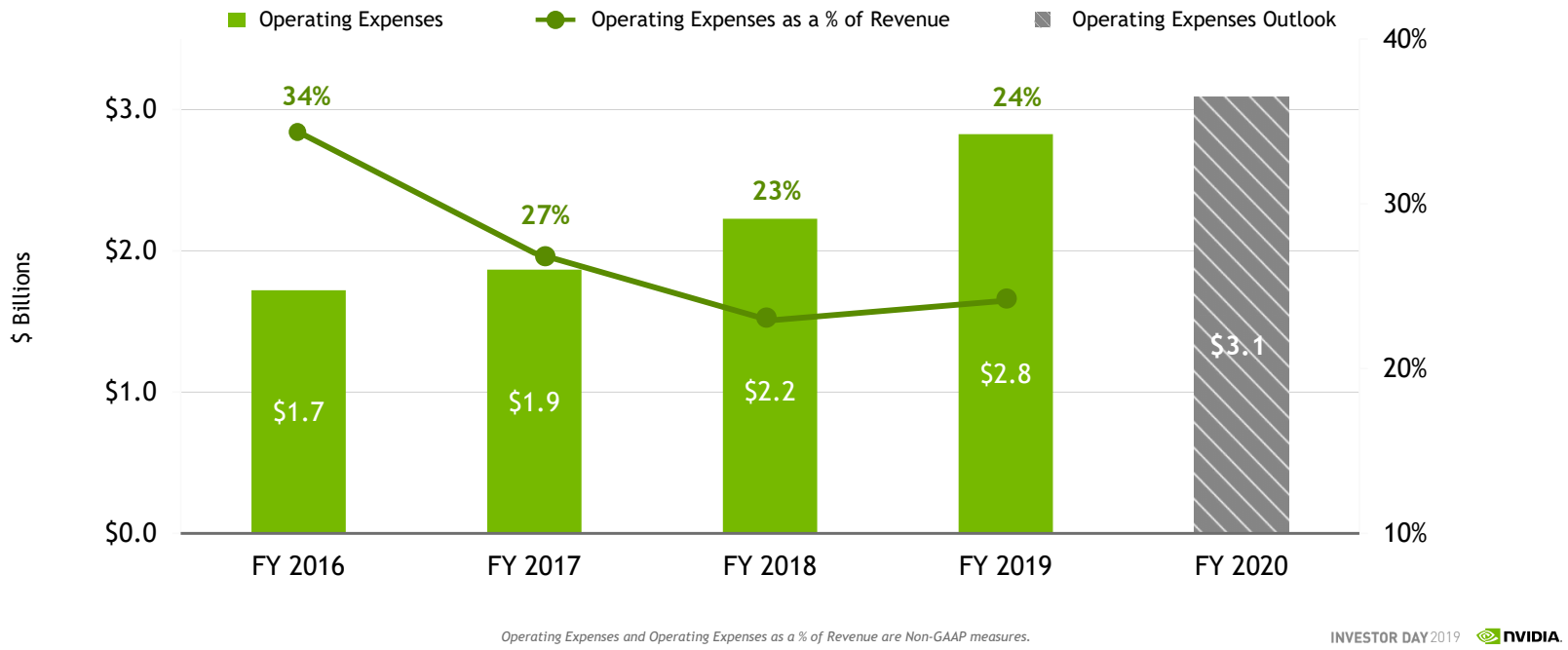

[Intel Investor Meeting 2019]

NVidia

# NVidia

## OPERATING EXPENSES

YoY Investments Focused on Gaming, AI, and Auto. 3-Year CAGR 18%




*Operating Expenses and Operating Expenses as a % of Revenue are Non-GAAP measures.*

INVESTOR DAY 2019 NVIDIA.

# NVidia

*"... that unified architecture allows us to be quite efficient."*
*– Colette Kress (NVidia CFO)*

# ARM

# ARM

- ARM sells chip designs

# ARM

- ARM sells chip designs
- Manufacturers can customize it

# ARM

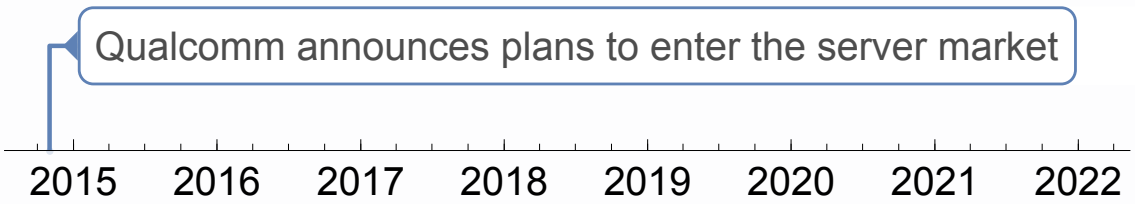
- ARM sells chip designs
- Manufacturers can customize it
- Datacenter providers need reliable partners,

# ARM

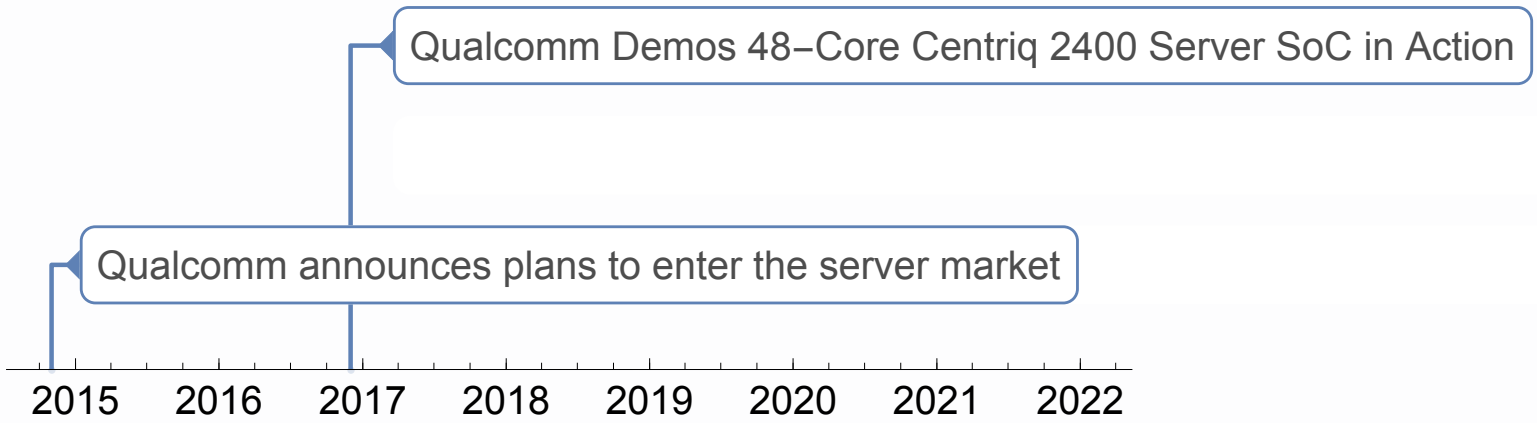
- ARM sells chip designs
- Manufacturers can customize it
- Datacenter providers need reliable partners,
- like Qualcomm

# Qualcomm's project Amberwing/Centriq

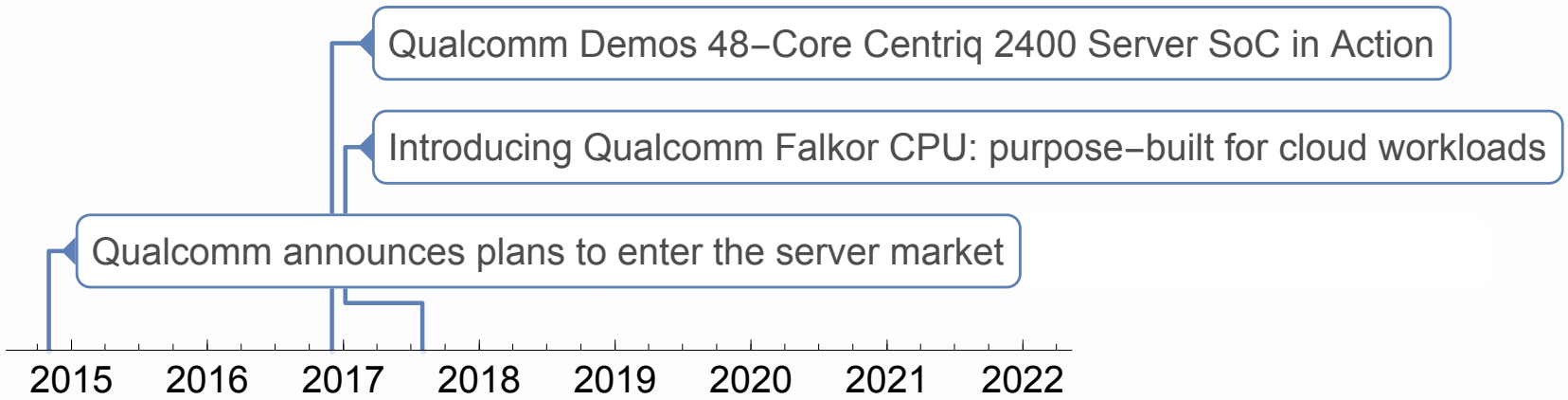
# Qualcomm's project Amberwing/Centriq



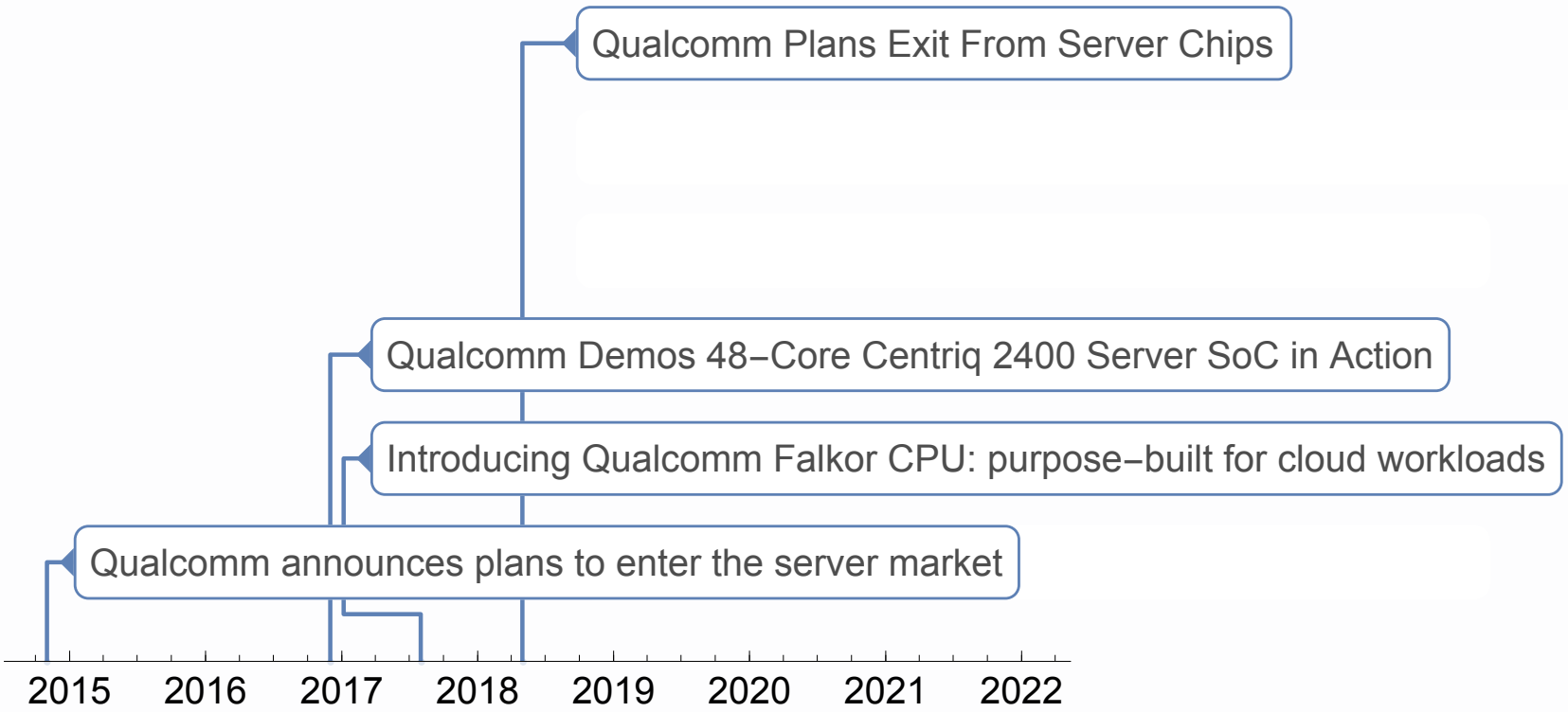
# Qualcomm's project Amberwing/Centriq



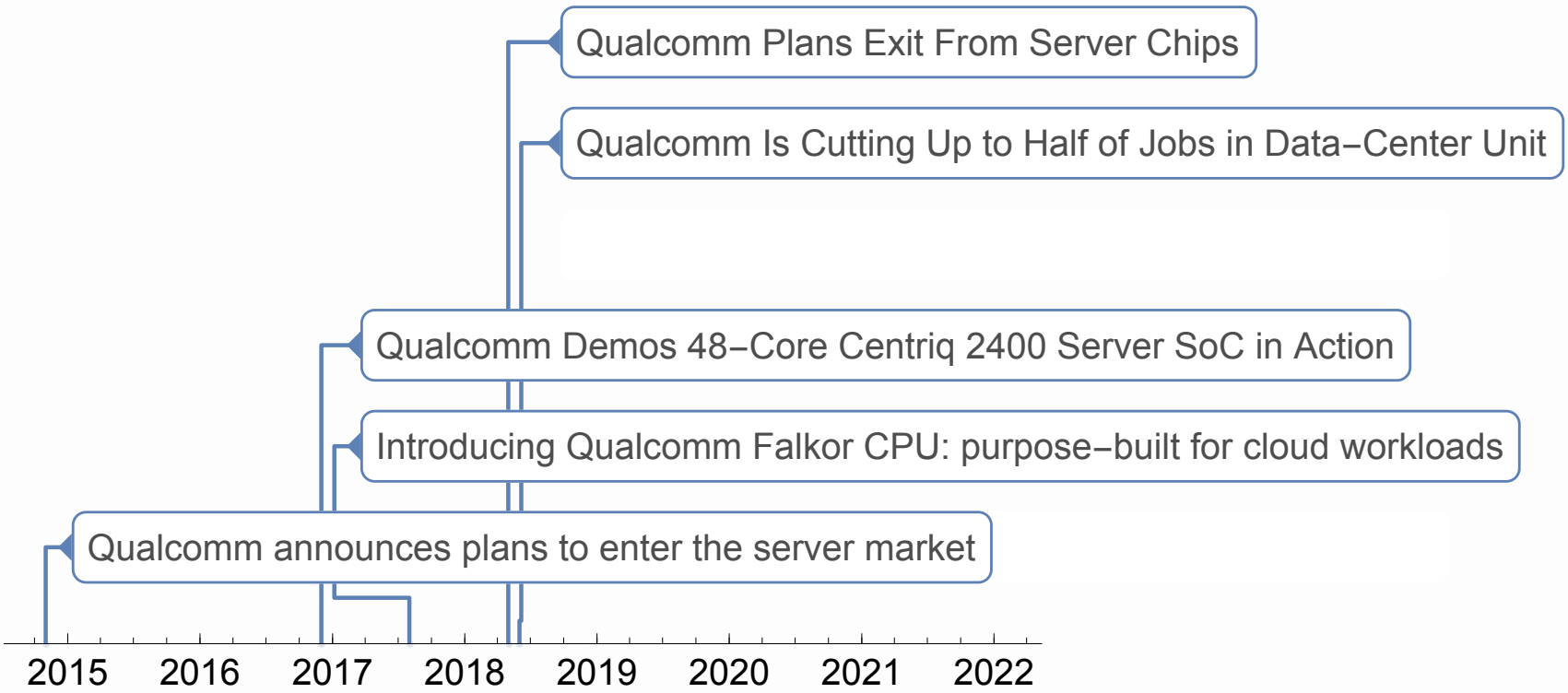
# Qualcomm's project Amberwing/Centriq



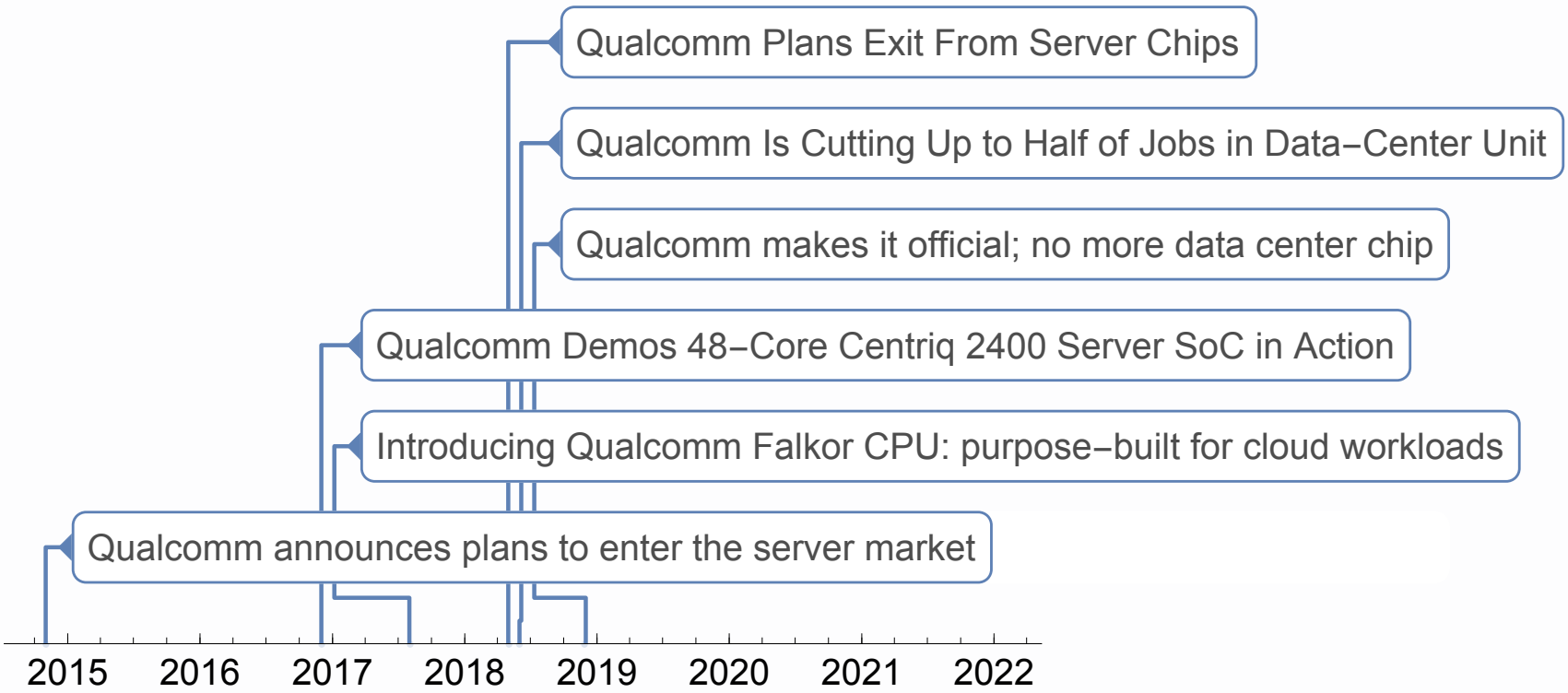
# Qualcomm's project Amberwing/Centriq



# Qualcomm's project Amberwing/Centriq



# Qualcomm's project Amberwing/Centriq



Bottom line: data management is not a killer app

# What are the most important non-essential CPU features?

- Artificial Intelligence

# What are the most important non-essential CPU features?

- Artificial Intelligence
- Security

# What are the most important non-essential CPU features?

- Artificial Intelligence
- Security
- Augmented Reality/Graphics

# What are the most important non-essential CPU features?

- Artificial Intelligence
- Security
- Augmented Reality/Graphics
- Networking (wireless)

# What are the most important non-essential CPU features?

- Artificial Intelligence
- Security
- Augmented Reality/Graphics
- Networking (wireless)
- Modem/DSP

# What are the most important non-essential CPU features?

- Artificial Intelligence
- Security
- Augmented Reality/Graphics
- Networking (wireless)
- Modem/DSP
- **Whatever apple has planned for the next iPhone**

Data Management will not control hardware design!

Let's not sit here and wait for it to happen!

Let's do something

# So, what do we do?

# So, what do we do?

- I like Database Research

# So, what do we do?

- I like Database Research
- What should Database Research cover?

# So, what do we do?

- I like Database Research
- What should Database Research cover?
- Two options:

# So, what do we do?

- I like Database Research
- What should Database Research cover?
- Two options:
  - Redefine Data Management

# So, what do we do?

- I like Database Research
- What should Database Research cover?
- Two options:
  - Redefine Data Management
  - **Hack the planet!**

# Option A: Redefine Data Management!

# Option A: Redefine Data Management!

- Arguably the easier/safer route!

# Option A: Redefine Data Management!

- Arguably the easier/safer route!
- After all,

# Option A: Redefine Data Management!

- Arguably the easier/safer route!
- After all,
  - AI works with data

# Option A: Redefine Data Management!

- Arguably the easier/safer route!
- After all,
  - AI works with data
  - Graphics is data

# Option A: Redefine Data Management!

- Arguably the easier/safer route!
- After all,
  - AI works with data
  - Graphics is data
  - Data security is important

# Option A: Redefine Data Management!

- Arguably the easier/safer route!
- After all,
  - AI works with data
  - Graphics is data
  - Data security is important
- Fine, but boring:

# Option A: Redefine Data Management!

- Arguably the easier/safer route!
- After all,
  - AI works with data
  - Graphics is data
  - Data security is important
- Fine, but boring:
  - The agenda is being defined by the hardware folks $\Rightarrow$ we stop innovating (with respect to hardware)

Option B: Find creative use for "the hardware we have"!


HACK THE
PLANET

No?

Man, I grew up at a special period in human history!

# Option B: Find creative use for "the hardware we have"!

- Seems risky, but

# Option B: Find creative use for "the hardware we have"!

- Seems risky, but
  - much more fun

# Option B: Find creative use for "the hardware we have"!

- Seems risky, but
  - much more fun
  - actually innovative

# Option B: Find creative use for "the hardware we have"!

- Seems risky, but
  - much more fun
  - actually innovative
  - research is risky!

Remember the olden days (of Database Research)?

Database research was a lot more wild west, back then!

# 2005

# Efficient Relational Database Management using Graphics Processors

Naga K. Govindaraju Dinesh Manocha

University of North Carolina at Chapel Hill

{naga, dm}@cs.unc.edu
http://gamma.cs.unc.edu/DB

## ABSTRACT

We present algorithms using graphics processing units (GPUs) to efficiently perform database management queries. Our algorithms use efficient data memory representations and storage models on GPUs to perform fast database computations. We present relational database algorithms that successfully exploit the high memory bandwidth and the inherent parallelism available in GPUs. We implement these algorithms on commodity GPUs and compare their performance with optimized CPU-based algorithms. We show that the GPUs can be used as a co-processor to accelerate many database and data mining queries.

## 1. INTRODUCTION

The challenge of real-time data analysis pushes the limits of database

to increasing programmability, the GPUs have evolved into powerful processors that are flexible enough to perform general purpose computations [16]. The inherent parallelism and the high memory bandwidth within the GPUs has been used to accelerate many of the traditional algorithms by an order of magnitude as compared to the CPU-based implementations. For example, a high-end GPU such as NVIDIA GeForce FX 6800 Ultra has a peak performance of 48 GFlops and a peak memory bandwidth of 35.2 GBps. In contrast, a 3.4 GHz Pentium IV CPU is capable of a peak performance of 6.8 GFlops and the DDRII main memories on PCs have a peak memory bandwidth of 6.4 GBps. Furthermore, the computational power of GPUs is progressing at a rate faster than the Moore's law for the CPUs.

# 2007

## Vectorized Data Processing on the Cell Broadband Engine

Sándor Héman Niels Nes Marcin Zukowski Peter Boncz

CWI, Kruislaan 413
Amsterdam, The Netherlands
{Firstname.Lastname}@cwi.nl

**ABSTRACT**

In this work, we research the suitability of the Cell Broadband Engine for database processing. We start by outlining the main architectural features of Cell and use micro-benchmarks to characterize the latency and throughput of its memory infrastructure. Then, we discuss the challenges of porting RDBMS software to Cell: *(i)* all computations need to SIMD-ized, *(ii)* all performance-critical branches need to be eliminated, *(iii)* a very small and hard limit on program code size should be respected.

While we argue that conventional database implementations, i.e. row-stores with Volcano-style tuple pipelining, are a hard fit to Cell, it turns out that the three challenges we use the TPC-H data warehousing benchmark to evaluate the efficiency of running database software on Cell.

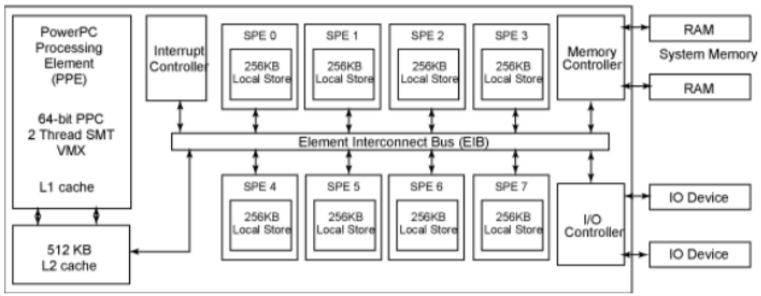



**Figure 1: Cell Broadband Engine Architecture**

2011

# How to Efficiently Snapshot Transactional Data: Hardware or Software Controlled?

Henrik Mühe
TU München
Boltzmannstr. 3
85748 Garching, Germany
muehe@in.tum.de

Alfons Kemper
TU München
Boltzmannstr. 3
85748 Garching, Germany
kemper@in.tum.de

Thomas Neumann
TU München
Boltzmannstr. 3
85748 Garching, Germany
neumann@in.tum.de

## ABSTRACT

The quest for real-time business intelligence requires executing mixed transaction and query processing workloads on the same current database state. However, as Harizopoulos et al. [6] showed for transactional processing, coexecution using classical concurrency control techniques will not yield the necessary performance – even in re-emerging main memory database systems. Therefore, we designed an in-memory database system that separates transaction processing from OLAP query processing via periodically refreshed snapshots. Thus, OLAP queries can be executed without any synchronization and OLTP transaction processing follows the lock-free, mostly serial processing paradigm of H-Store [8]. In this paper, we analyze different snapshot mechanisms: Hardware-supported Page Shadowing, which lazily copies memory pages when changed by transactions, software controlled Tuple Shadowing, which generates a new

product based on the H-Store research prototype [8]. There, costly mechanisms like locking and latching for concurrency control are avoided by executing all transactions sequentially, therefore yielding serializability without overhead. In order to increase the level of parallelism, the database can be logically partitioned to accommodate one transaction per partition [3].

Lockless, sequential execution has proven to be very efficient for OLTP workloads, specifically short transactions that modify only a handful of tuples and terminate within microseconds. With the need for "real-time business intelligence" as advocated by Plattner of SAP [13] among others, serial execution is bound to fail. Long-running OLAP queries cannot be executed sequentially with short OLTP transactions since transaction throughput would be severely diminished every time a long-running query stops OLTP transactions from being executed. To solve this dilemma,

There are more opportunities!

# Did you know that...

- ...your GPUs have onboard DSPs?

# Did you know that...

- ...your GPUs have onboard DSPs?
- ...Intel Vt-X provides cheap and easy sandboxing of code?

# Did you know that...

- ...your GPUs have onboard DSPs?
- ...Intel Vt-X provides cheap and easy sandboxing of code?
- ...GPUs do up to 4 times more floating point multiplies than integer

# Did you know that...

- ...your GPUs have onboard DSPs?
- ...Intel Vt-X provides cheap and easy sandboxing of code?
- ...GPUs do up to 4 times more floating point multiplies than integer
  - (hint: hashing involves multiplications)?

# Did you know that...

- ...your GPUs have onboard DSPs?
- ...Intel Vt-X provides cheap and easy sandboxing of code?
- ...GPUs do up to 4 times more floating point multiplies than integer
  - (hint: hashing involves multiplications)?
- Hardware performance counters have virtually zero overhead?

So, why were/are they not adopted?

# Adoption challenges:

- Increased system complexity

# Adoption challenges:

- Increased system complexity
- Lack of a story for integration:

# Adoption challenges:

- Increased system complexity
- Lack of a story for integration:
  - Hardcoding a prototype is not enough

# Adoption challenges:

- Increased system complexity
- Lack of a story for integration:
  - Hardcoding a prototype is not enough
  - Even extending Postgres seems to not be enough

# Adoption challenges:

- Increased system complexity
- Lack of a story for integration:
  - Hardcoding a prototype is not enough
  - Even extending Postgres seems to not be enough
  - We need to go deeper!

Abstractions are the key

# Abstractions are the key

- Need to be easy to integrate

# Abstractions are the key

- Need to be easy to integrate
- Need to hide complexity

# Abstractions are the key

- Need to be easy to integrate
- Need to hide complexity
- The best are invisible

# Abstractions are the key

- Need to be easy to integrate
- Need to hide complexity
- The best are invisible
- Need to enable the exploitation of hardware features

# Abstractions are the key

- Need to be easy to integrate
- Need to hide complexity
- The best are invisible
- Need to enable the exploitation of hardware features
- Need compiler support

# The problem with abstractions

# My personal take:

- I've worked on abstractions for

# My personal take:

- I've worked on abstractions for
  - co-processing...

# My personal take:

- I've worked on abstractions for
  - co-processing...
  - tunable data analytics...

# My personal take:

- I've worked on abstractions for
  - co-processing...
  - tunable data analytics...
  - cost estimation...

# My personal take:

- I've worked on abstractions for
  - co-processing...
  - tunable data analytics...
  - cost estimation...
  - (AI...)

# My personal take:

- I've worked on abstractions for
  - co-processing...
  - tunable data analytics...
  - cost estimation...
  - (AI...)
- I am currently working on abstractions for

# My personal take:

- I've worked on abstractions for
  - co-processing...
  - tunable data analytics...
  - cost estimation...
  - (AI...)
- I am currently working on abstractions for
  - stream processing...

# My personal take:

- I've worked on abstractions for
  - co-processing...
  - tunable data analytics...
  - cost estimation...
  - (AI...)
- I am currently working on abstractions for
  - stream processing...
  - FPGAs...

# My personal take:

- I've worked on abstractions for
  - co-processing...
  - tunable data analytics...
  - cost estimation...
  - (AI...)
- I am currently working on abstractions for
  - stream processing...
  - FPGAs...
- ...and they are all different

This leaves me with the key question:

Is there a unifying abstraction?

This leaves me with the key question:

Is there a unifying abstraction?
(even if only for data mangement)

# Conclusion

# Conclusion

- Data Management will not determine hardware design!

# Conclusion

- Data Management will not determine hardware design!
- It never has!

# Conclusion

- Data Management will not determine hardware design!
- It never has!
- Folks still made a living (and had fun) by using hardware creatively!

# Conclusion

- Data Management will not determine hardware design!
- It never has!
- Folks still made a living (and had fun) by using hardware creatively!
- Dark silicon will provide more opportunities to be creative

Be creative!

But hide your creativity behind abstractions!

Thank you