

VLSI Trends in Microarchitecture

Past, present and future

TAU university

January 24, 2006
Uri Weiser

Agenda

- **Microarchitecture**
 - **VLSI**
 - **Trends Past and present:**
 - » **Pipeline, superpipeline**
 - » **Out of Order**
 - » **Branch prediction**
 - » **Caches**
 - » **Trace cache**
 - » **Threads and Chip Multiprocessing**
 - **Future**
 - » **Asymmetric**
 - » **Accelerators**

“[In the beginning] we had little idea of what we had started. ...I remember... saying, ‘Okay, we’ve done integrated circuit. What do we do next?’”

Gordon E. Moore

TRENDS IN VLSI

**Sources:
Shekhar Borkar
Uri Weiser**

Technology trend

Process Technology

1.5μ 1.0μ 0.8μ 0.6μ 0.35μ 0.25μ 0.18μ 0.13μ

Processor

Intel386™ DX
Processor

Intel486™ DX
Processor

Pentium®
Processor

Pentium® Pro
Processor

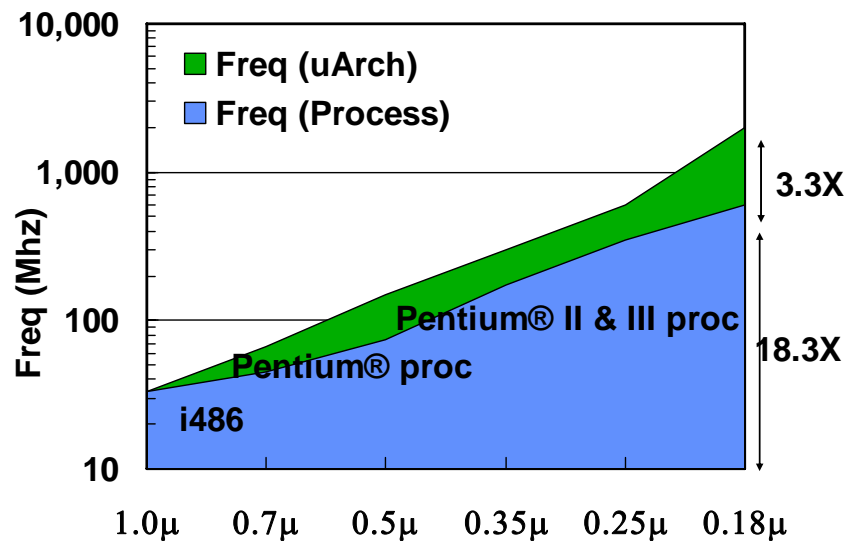
Pentium® II
Processor

Pentium® III
Processor

Pentium® 4
Processor



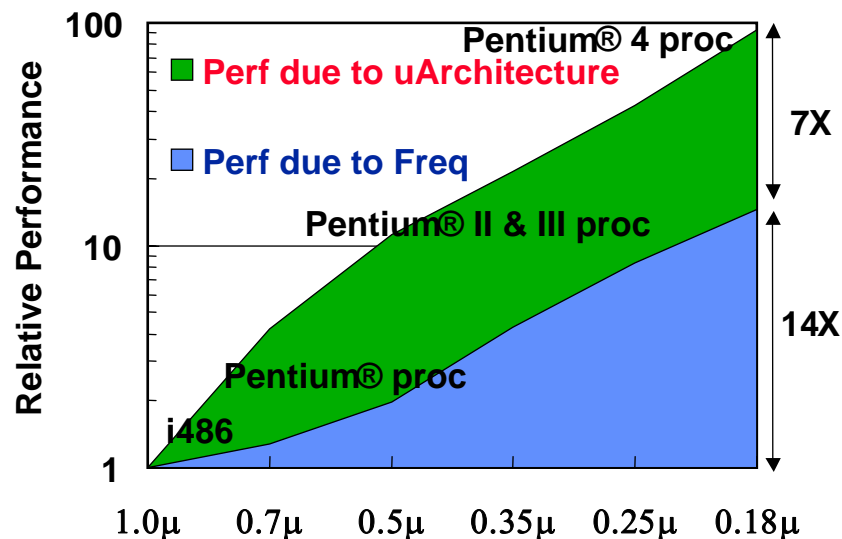
Performance History



1.0 μ -0.18 μ , 1989-2001

Frequency increased 61X

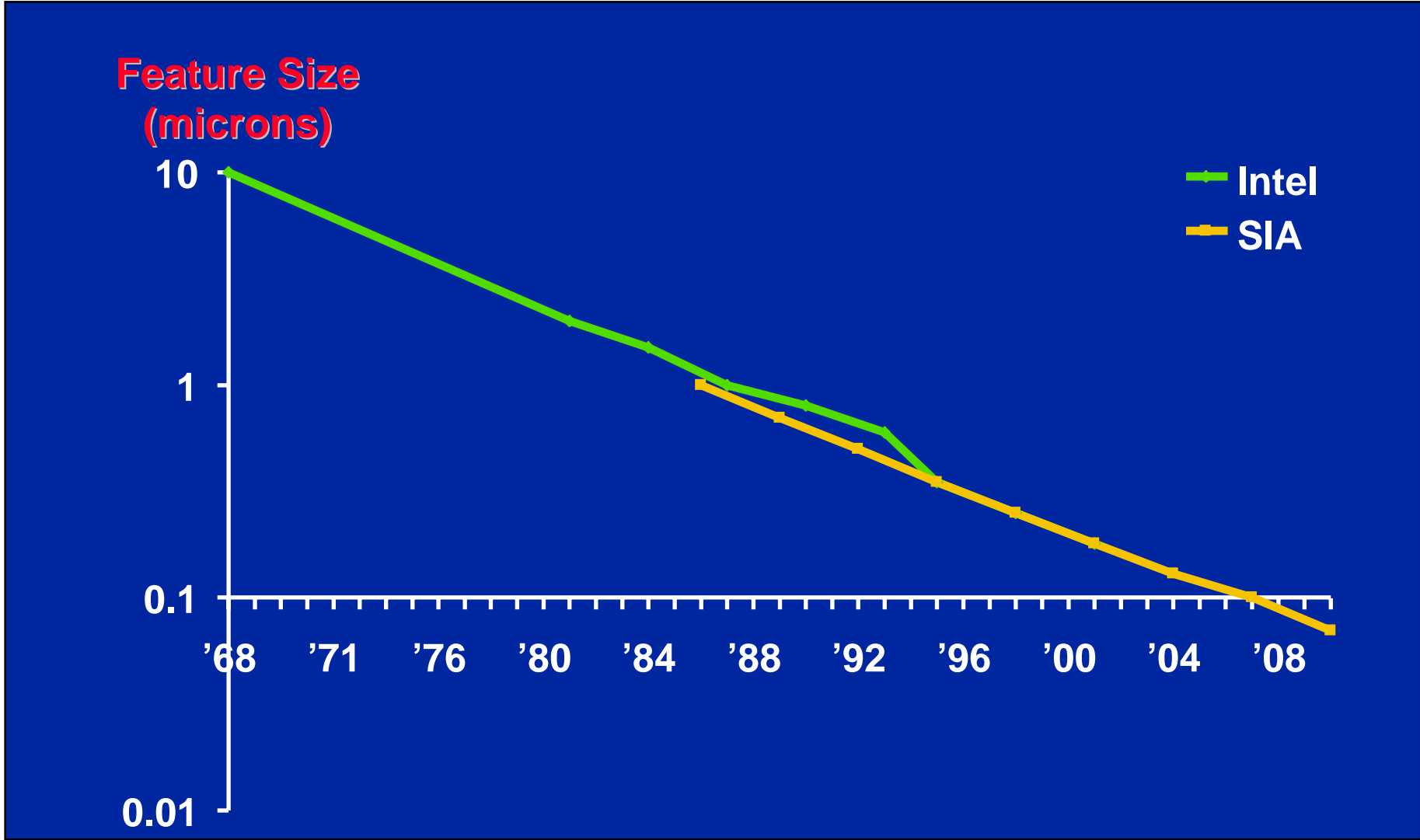
1. 18.3X due to process technology
2. Additional 3.3X due to uArch



Performance increased ~100X

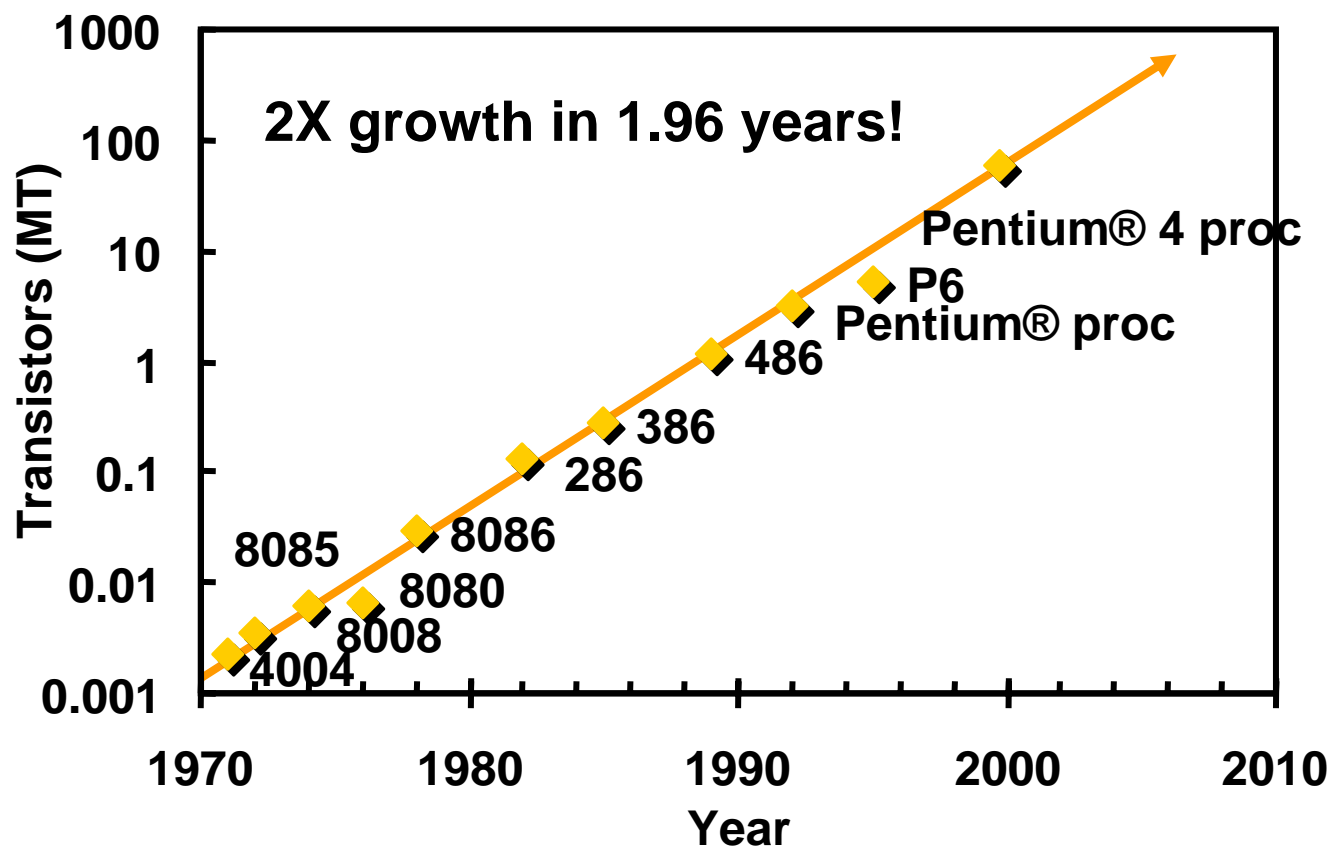
1. 14X due to process tech
2. Additional 7X due to uArch & design

Process Technology: Minimum Feature Size



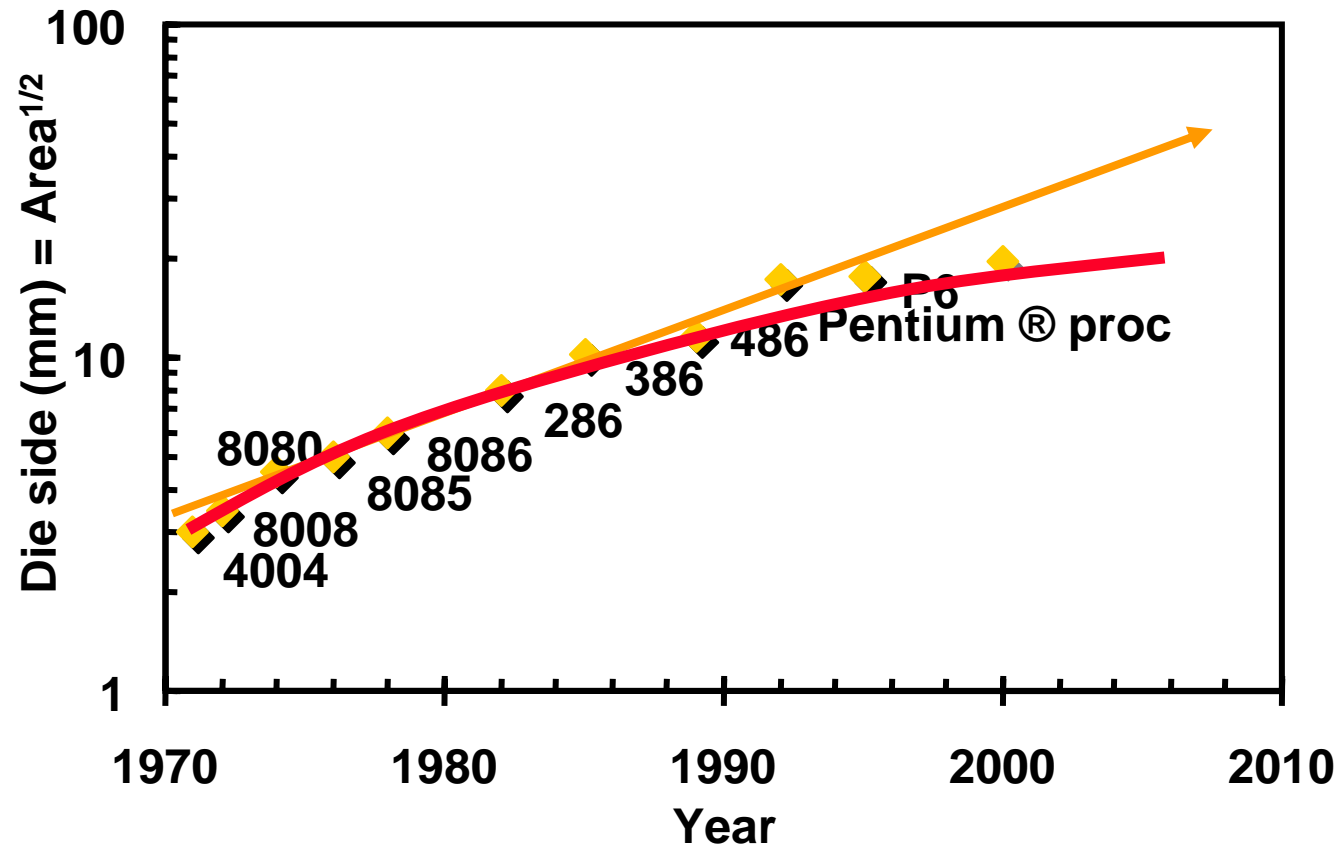
Source: Intel, SIA Technology Roadmap

Transistors on a Chip



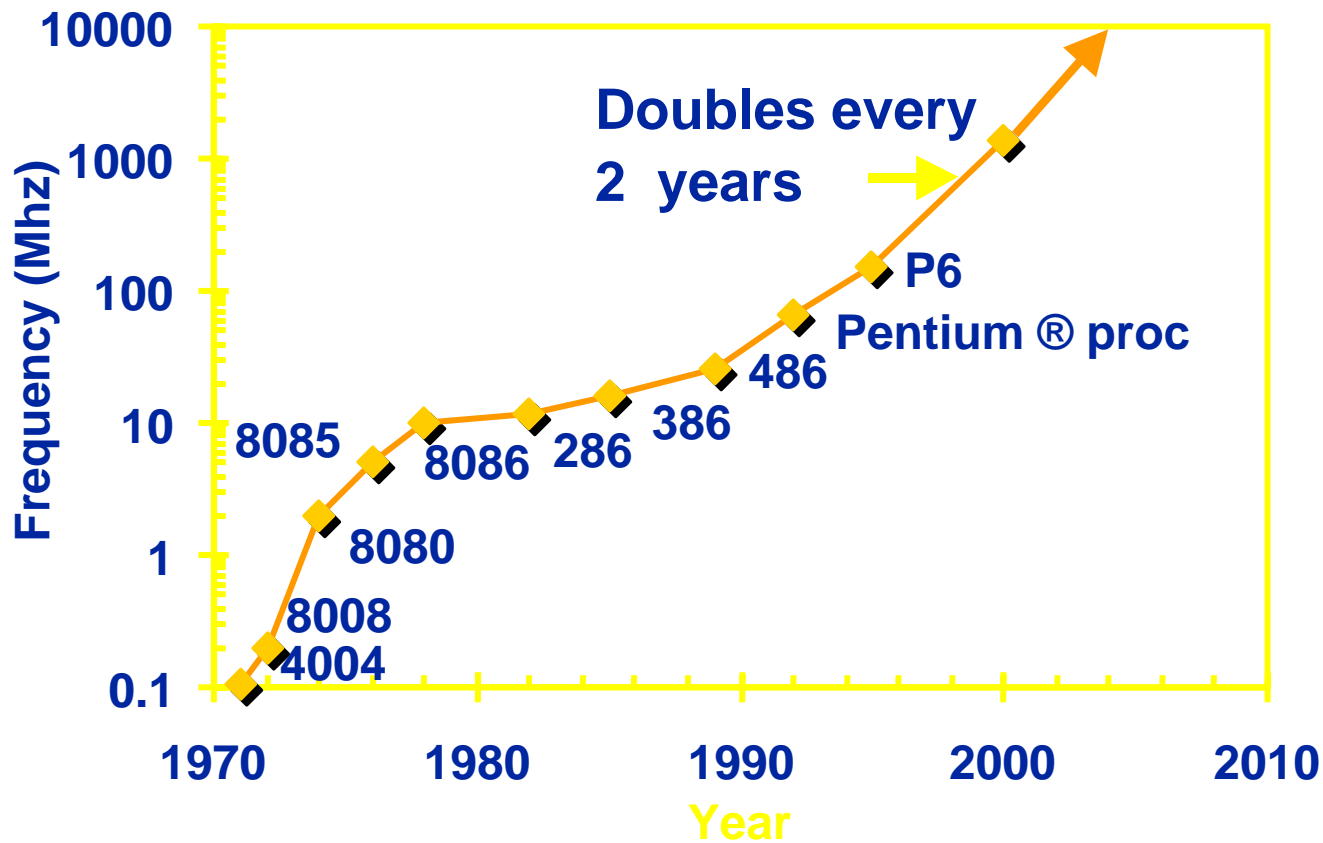
Transistors on a chip doubled every two years

Die Size Growth



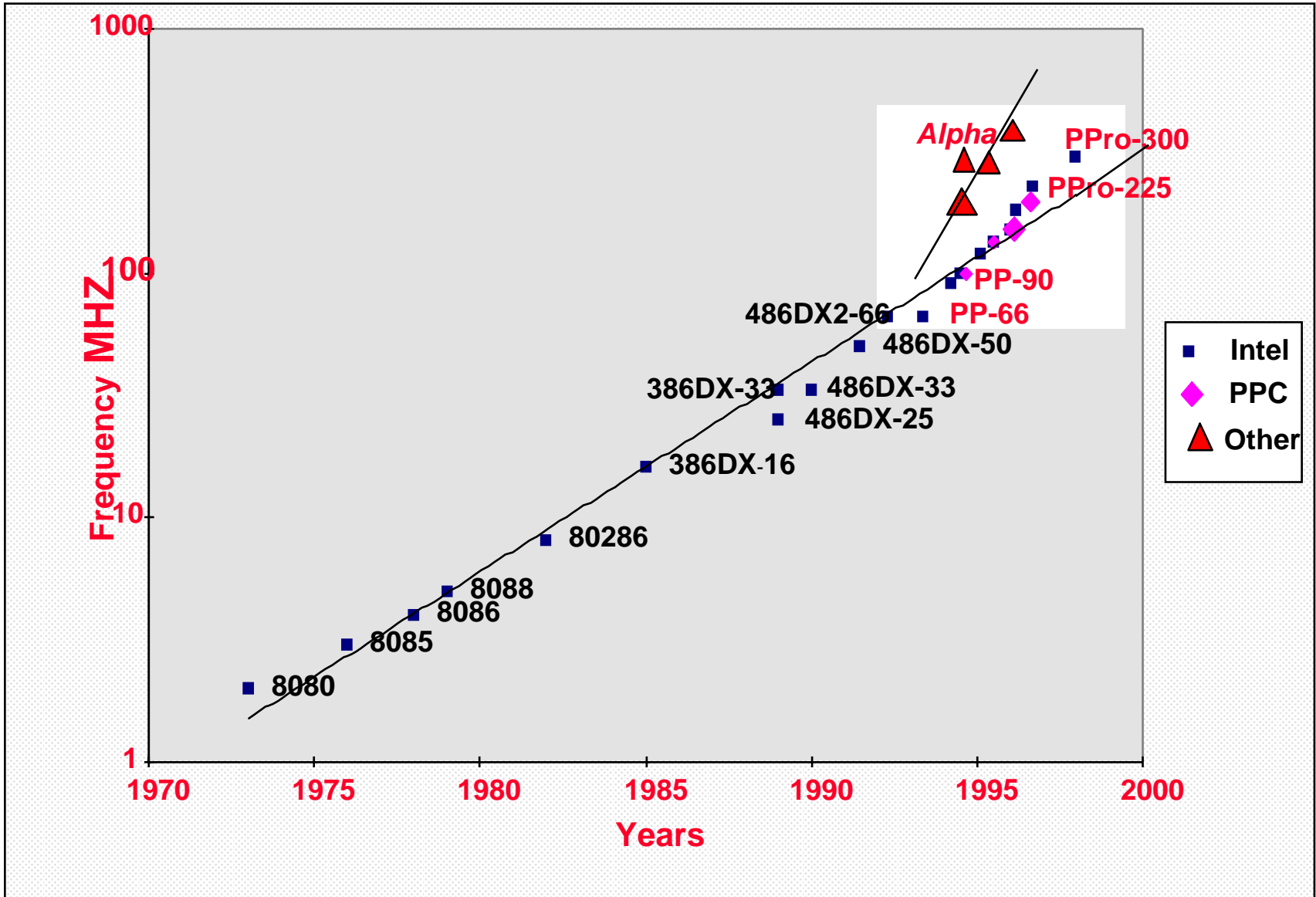
Die size grows? Is it saturated?

Frequency

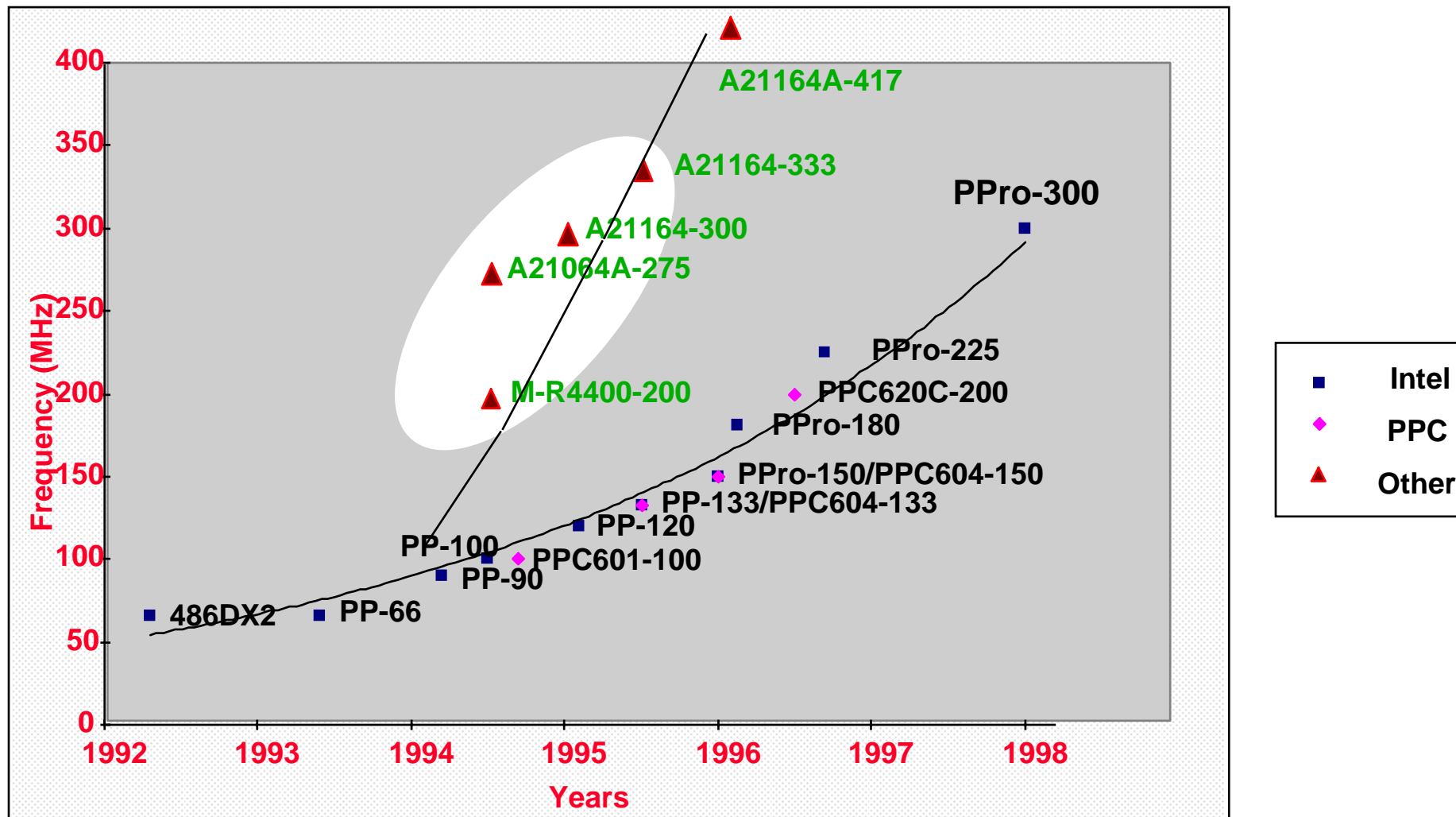


Lead Microprocessors frequency doubles every 2 years

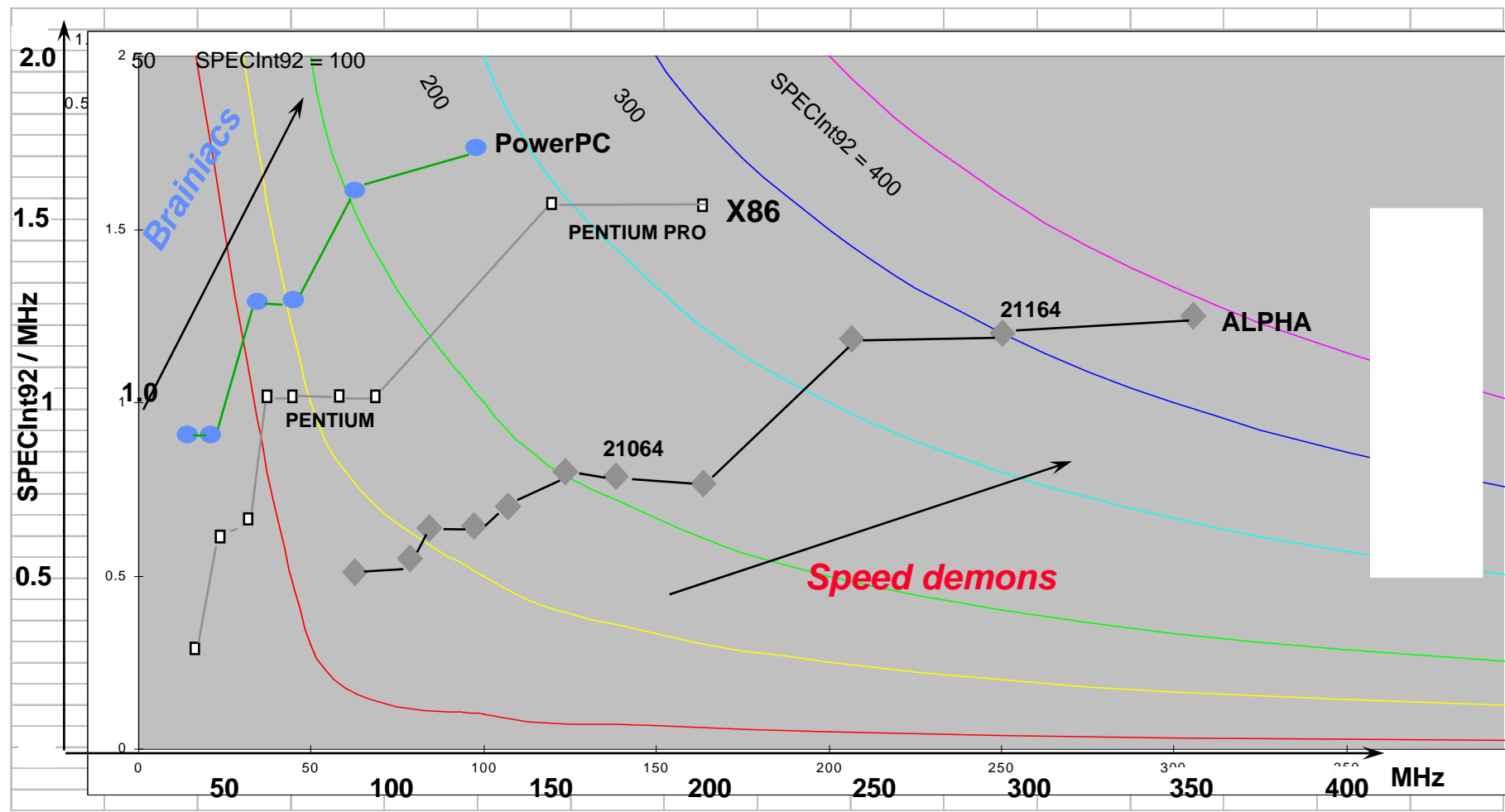
Frequency of Operation



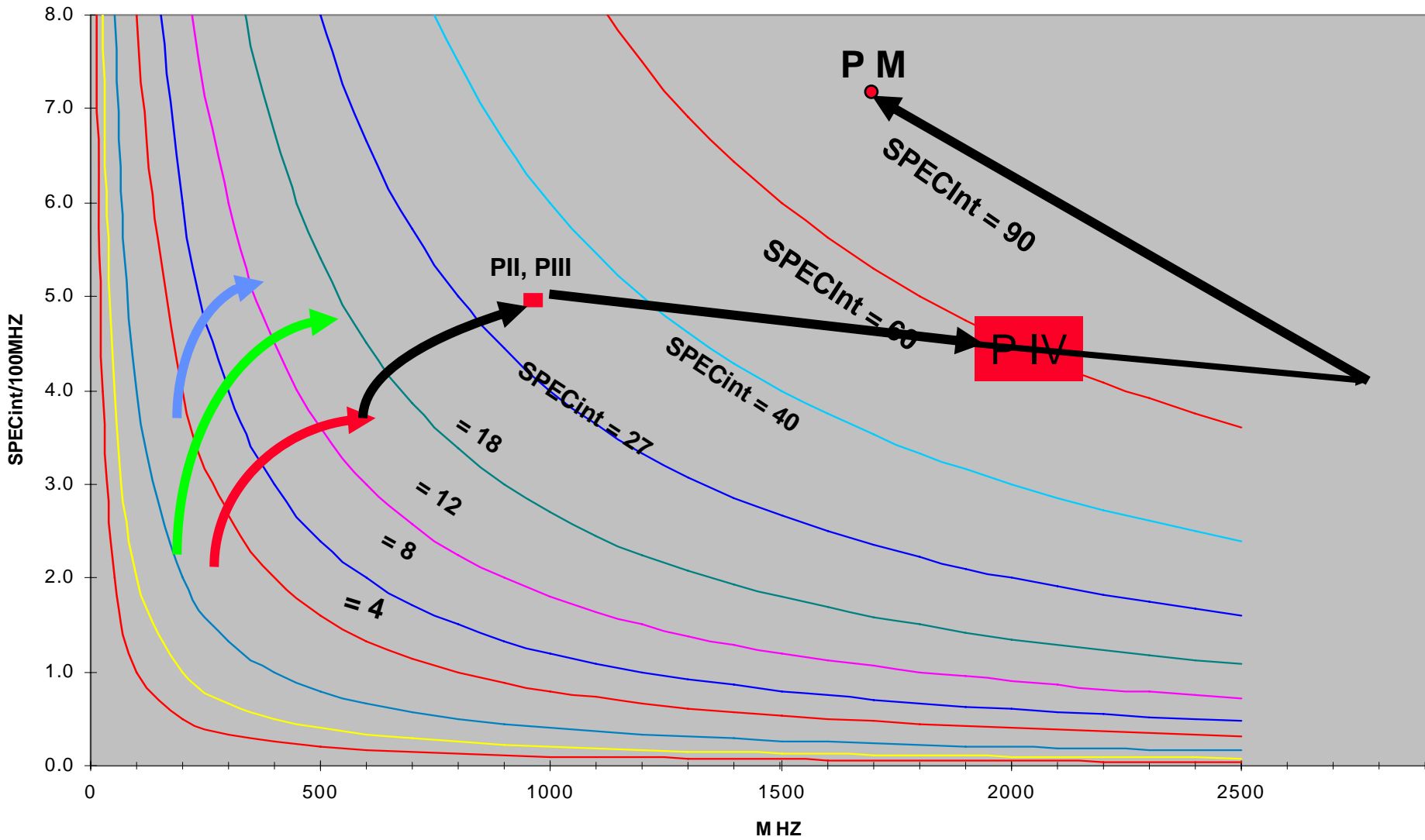
Frequency of Operation (cont.)



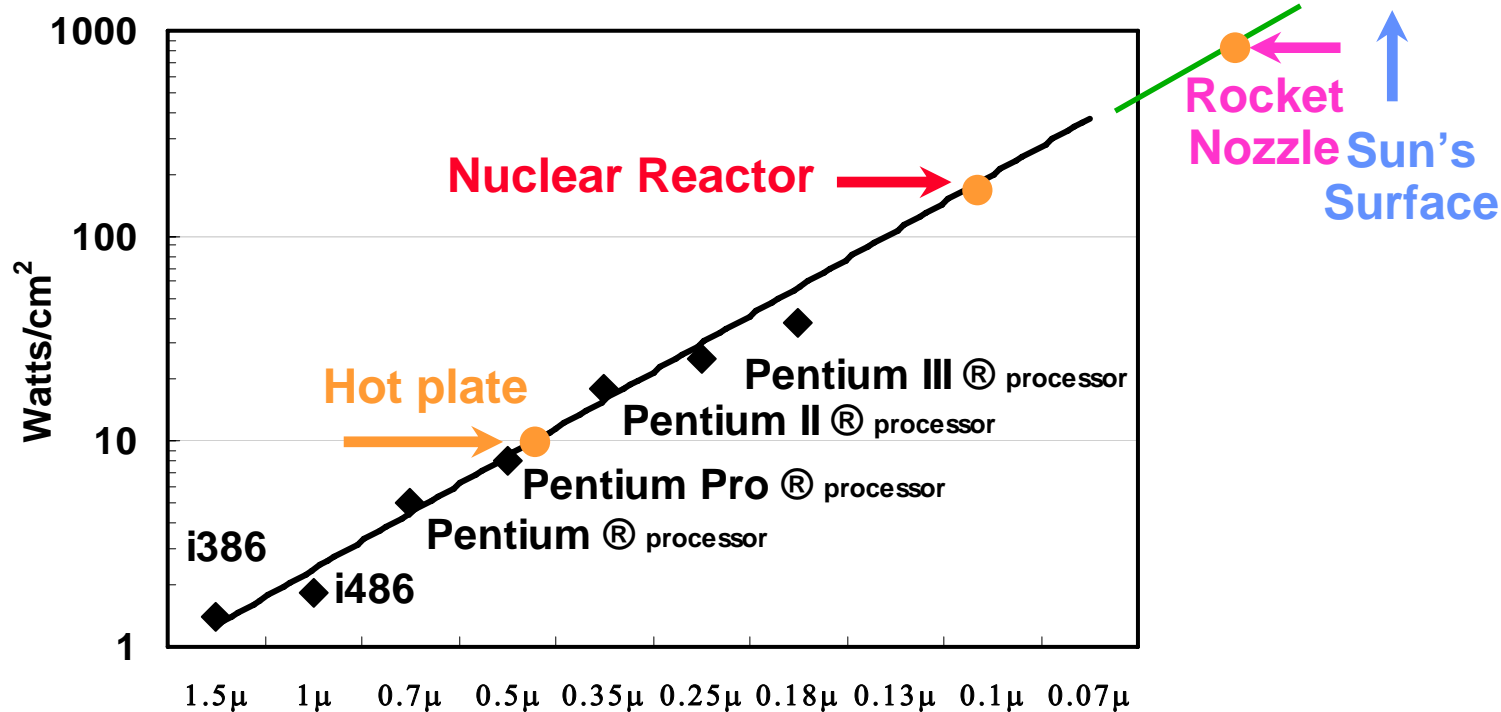
Brainiacs and Speed demons



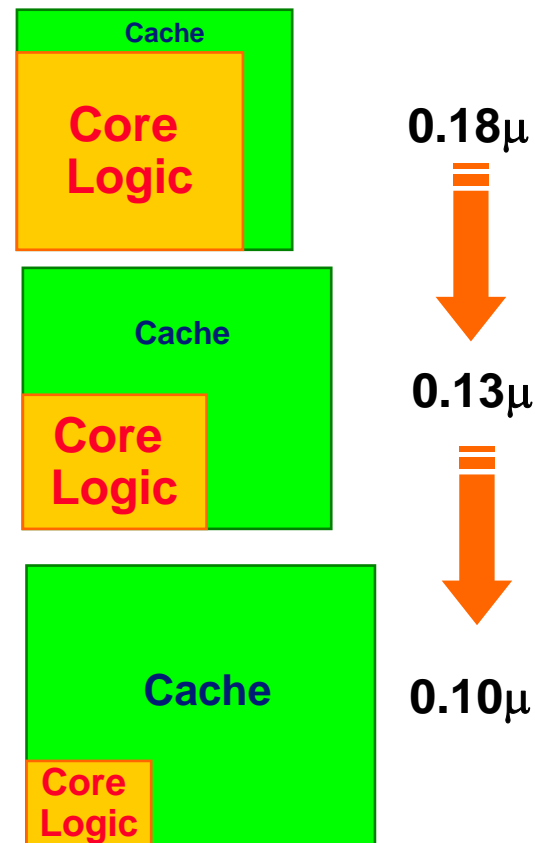
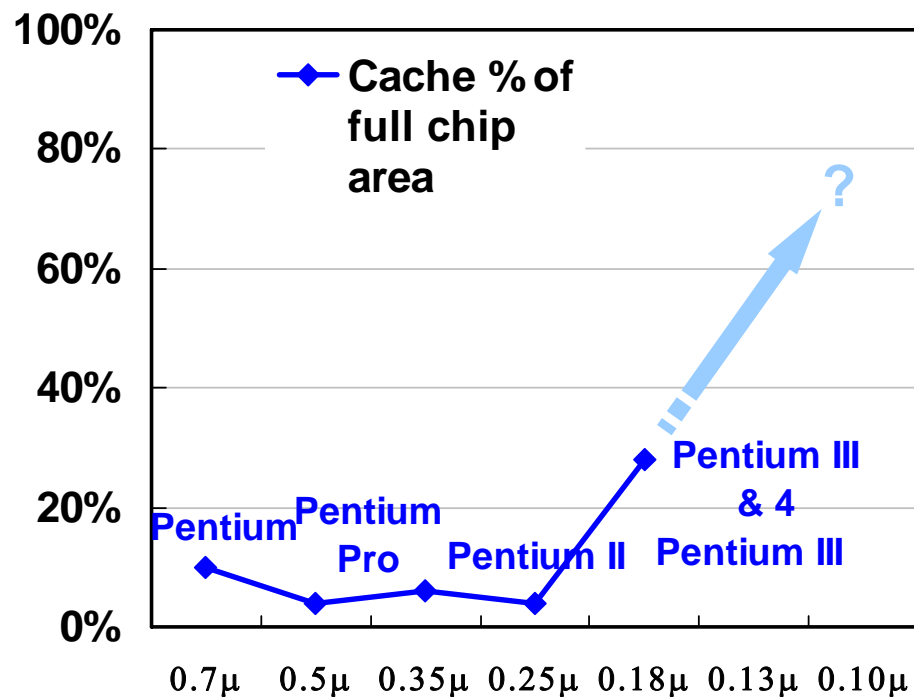
Trends of Future Processors



Power density continues to get worse



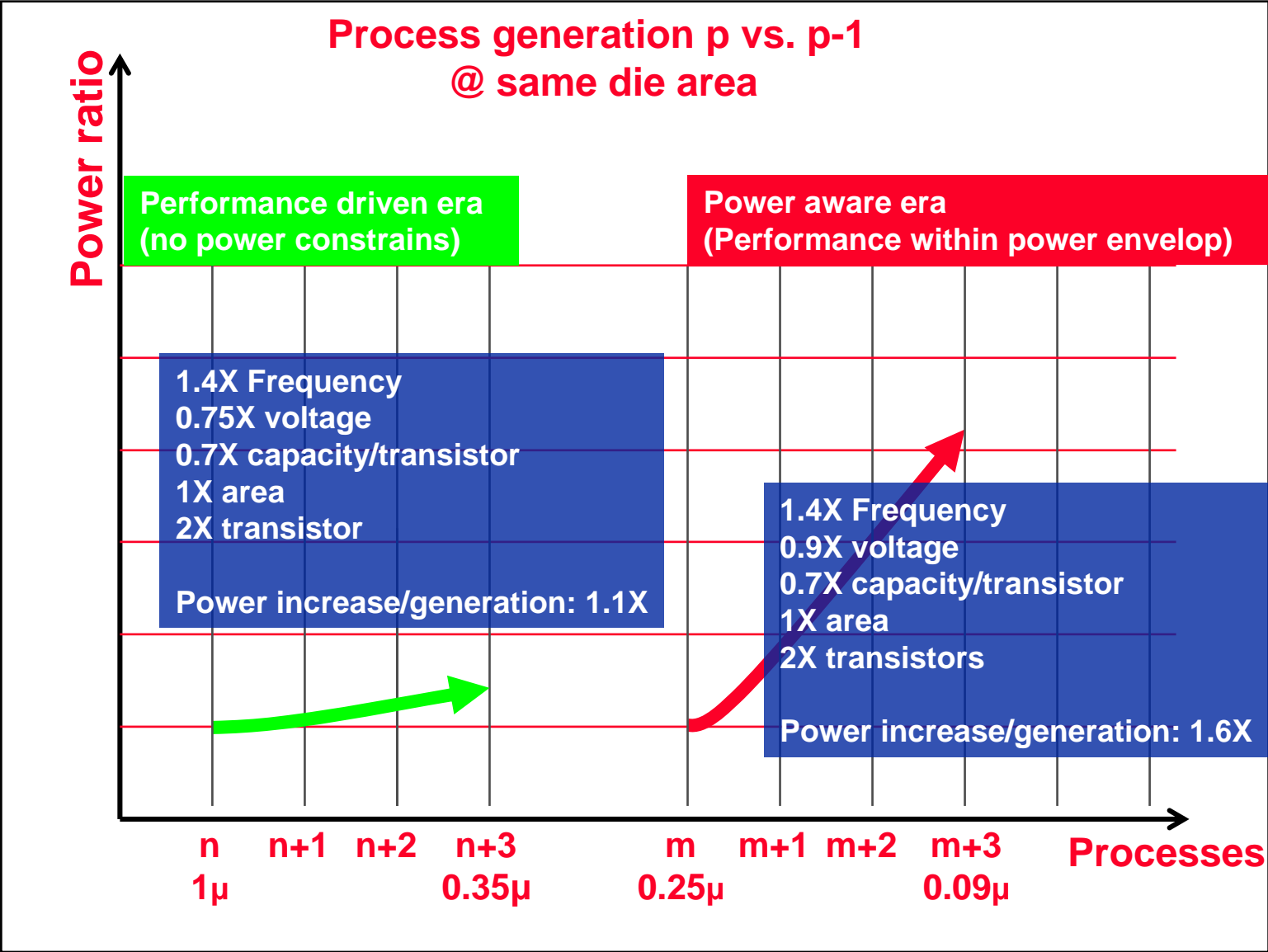
On Die Cache Memory



Larger % of die area will be memory

Process trend – the theory (cont)

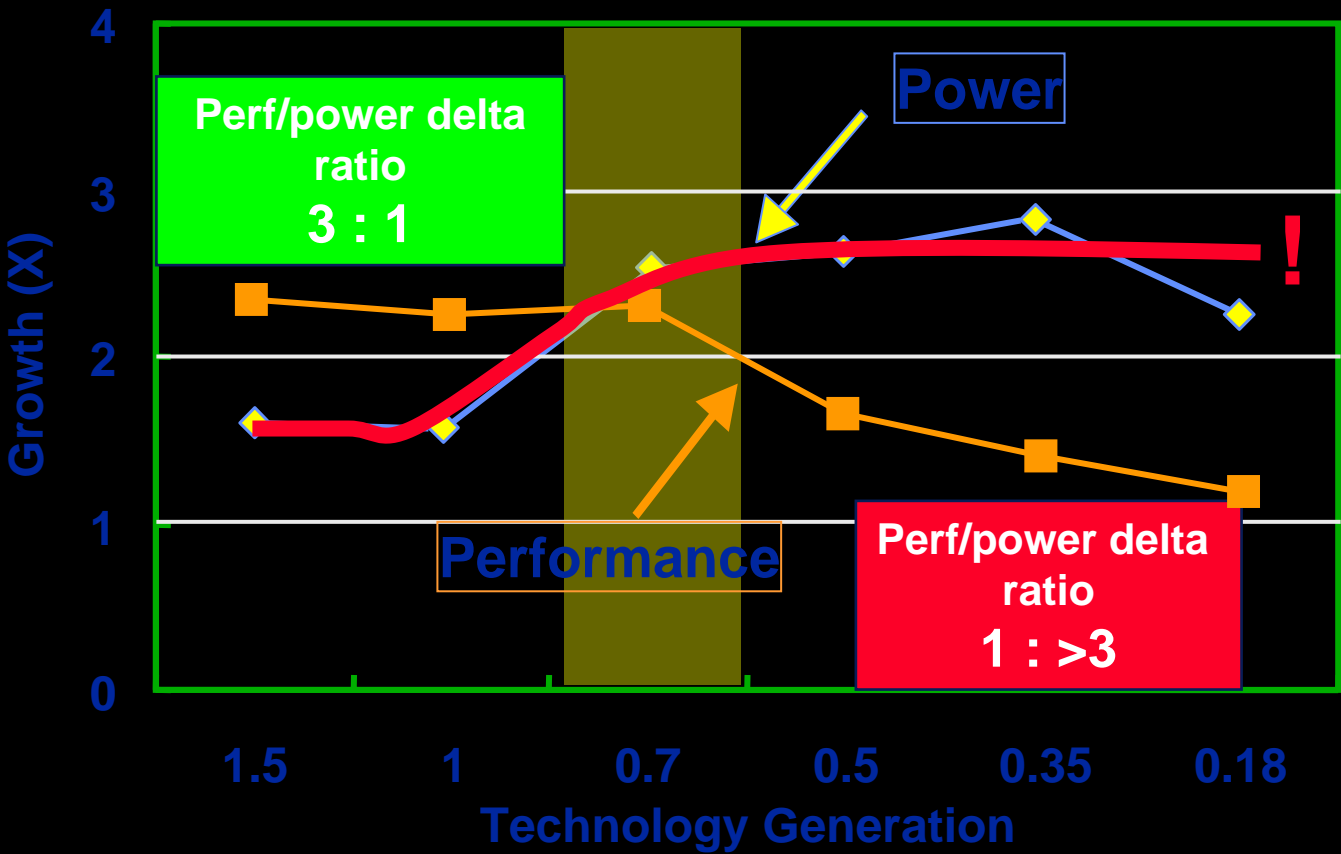
Performance driven era vs. Power aware era



Processor roadmap trend – real life (cont)

Extension of Pollack's Rule (Micro32, 1999)

Processor generation k vs. k-1 compacted
@ the same process technology



Microarchitecture

The Generic Processor

Sophisticated organization to “service” instructions

- **Instruction supply**

- Instruction cache
- Branch prediction
- Instruction decoder
- ...

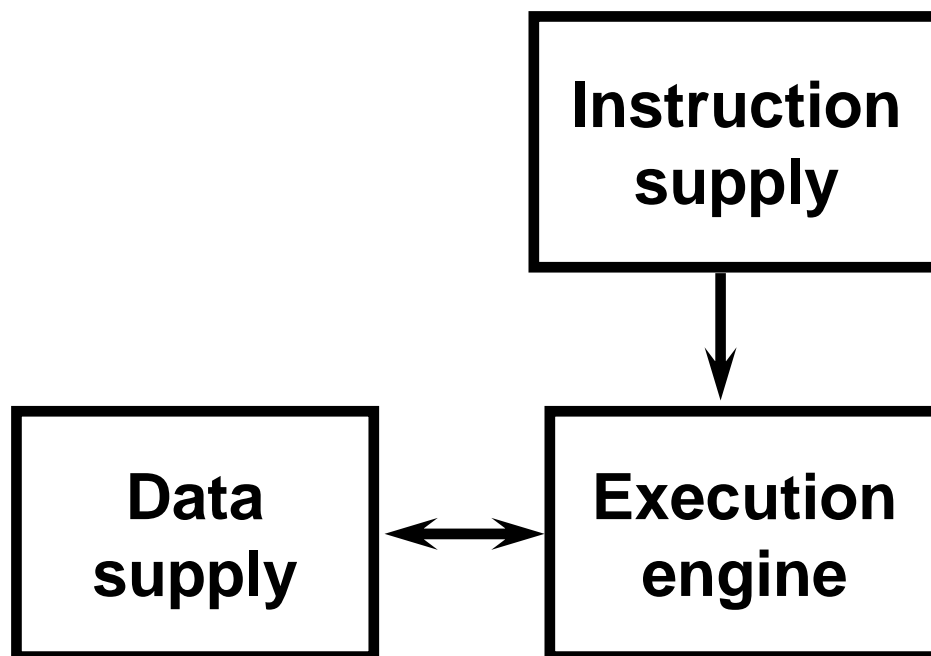
- **Execution engine**

- Instruction scheduler
- Register files
- Execution units
- ...

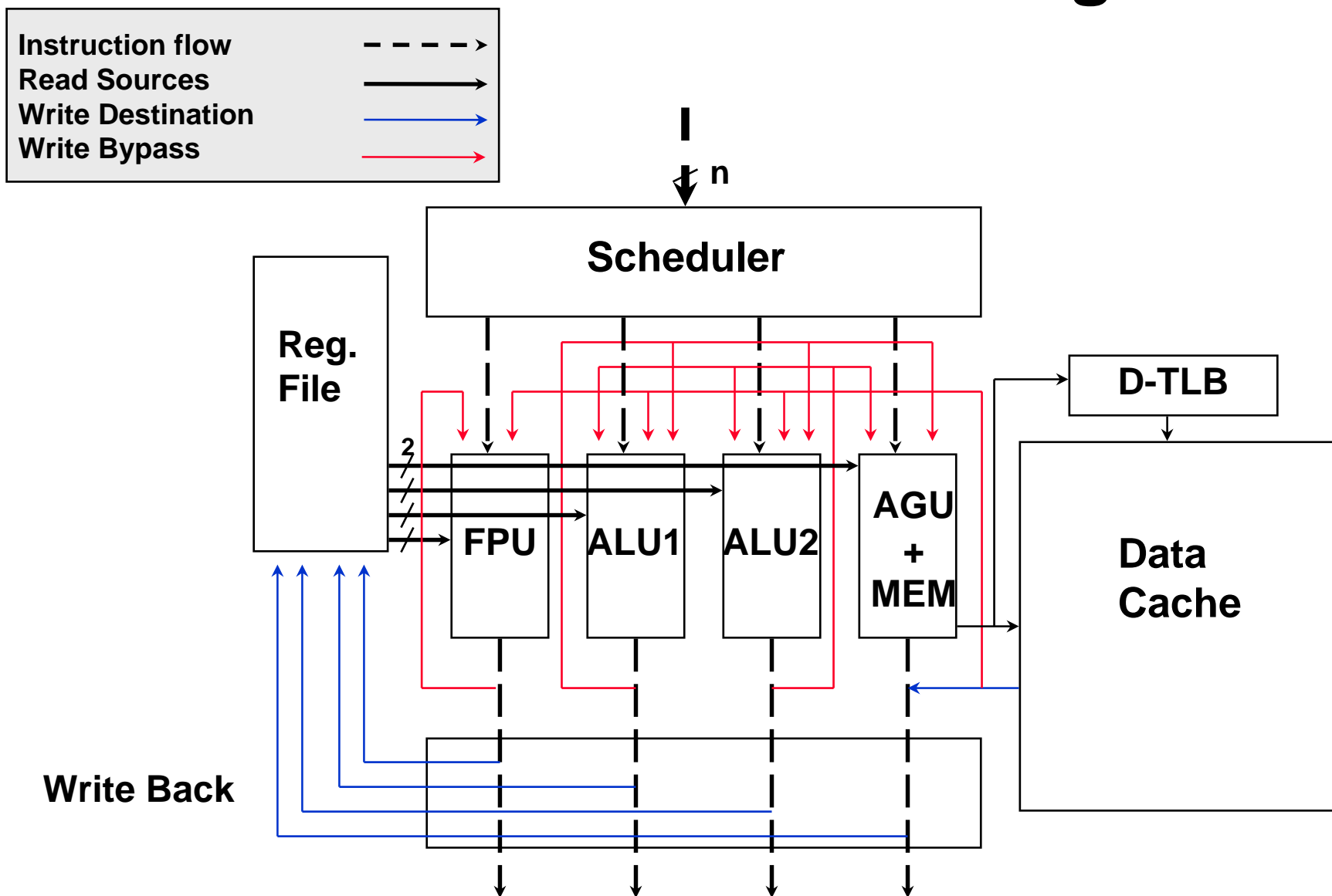
- **Data supply**

- Data cache
- TLB's
- ...

- **Goal - Maximum throughput – balanced design**

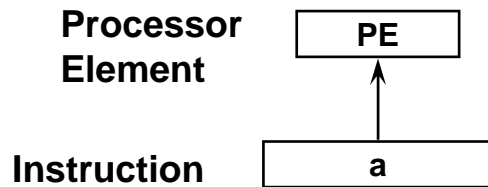


“The Core” - A Block Diagram

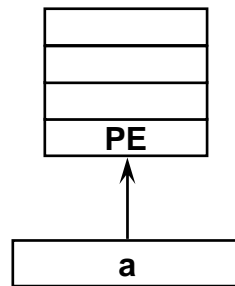


Parallelism Evolution

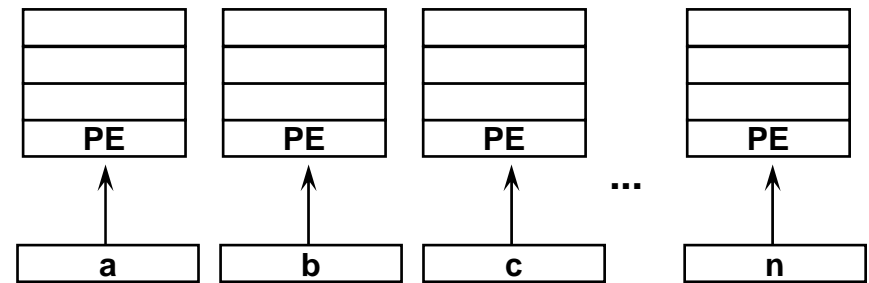
Basic configuration



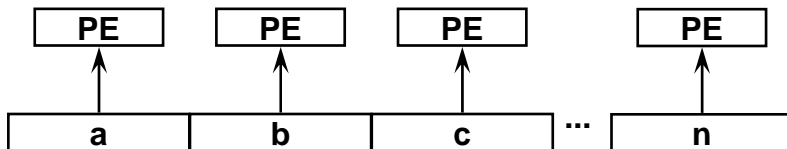
Pipeline



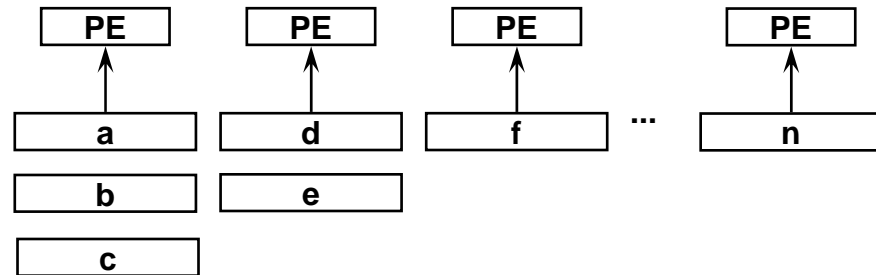
Superscalar - In order



VLIW

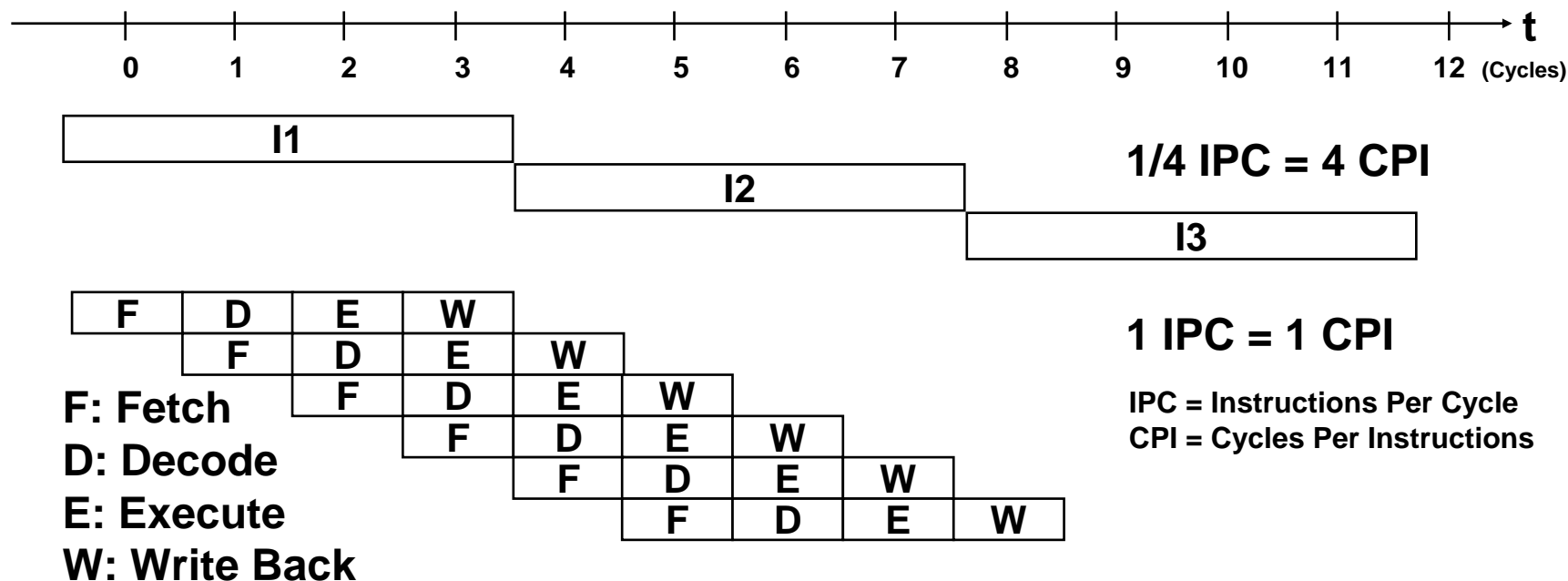


Superscalar - Out of Order



Pipeline

- Break the work to smaller pieces



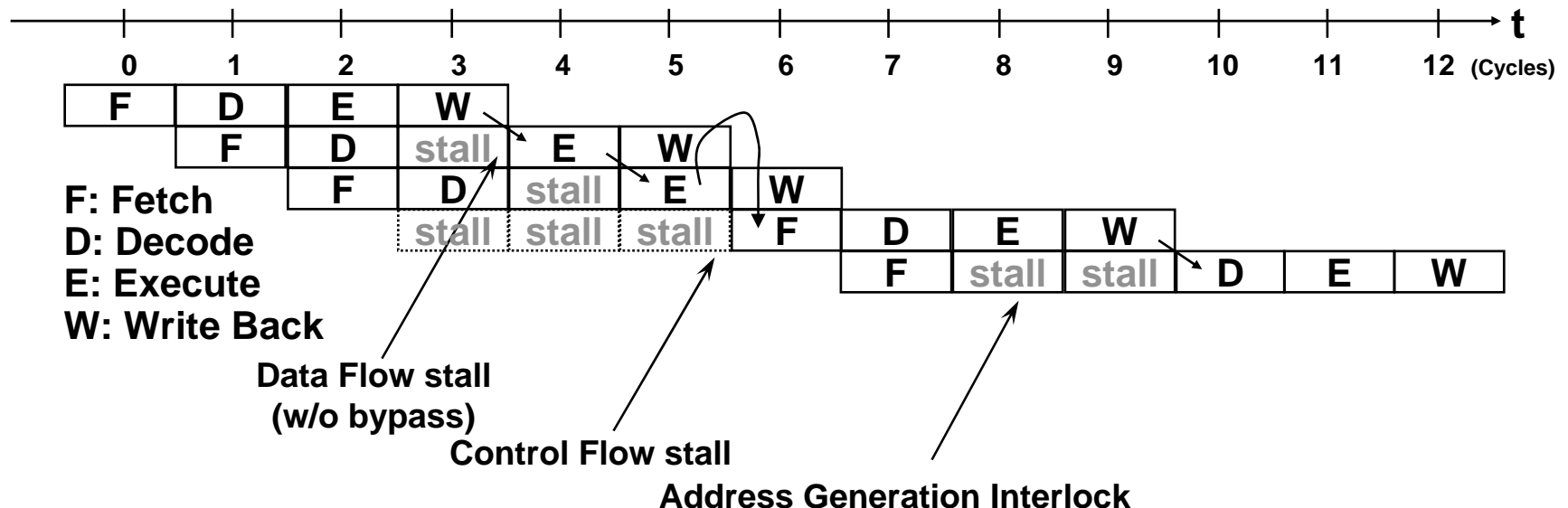
- Increased throughput
 - increased # of completed instructions per cycle and reduces cycle time
 - Number of stages varies
 - Small: 4-5 (Pentium), “Superpipeline” ~14 (Pentium Pro), “ultra-pipeline” ~25 (PIV)

- Calls for good balancing among stages

Examples
Intel 486
NS 32532

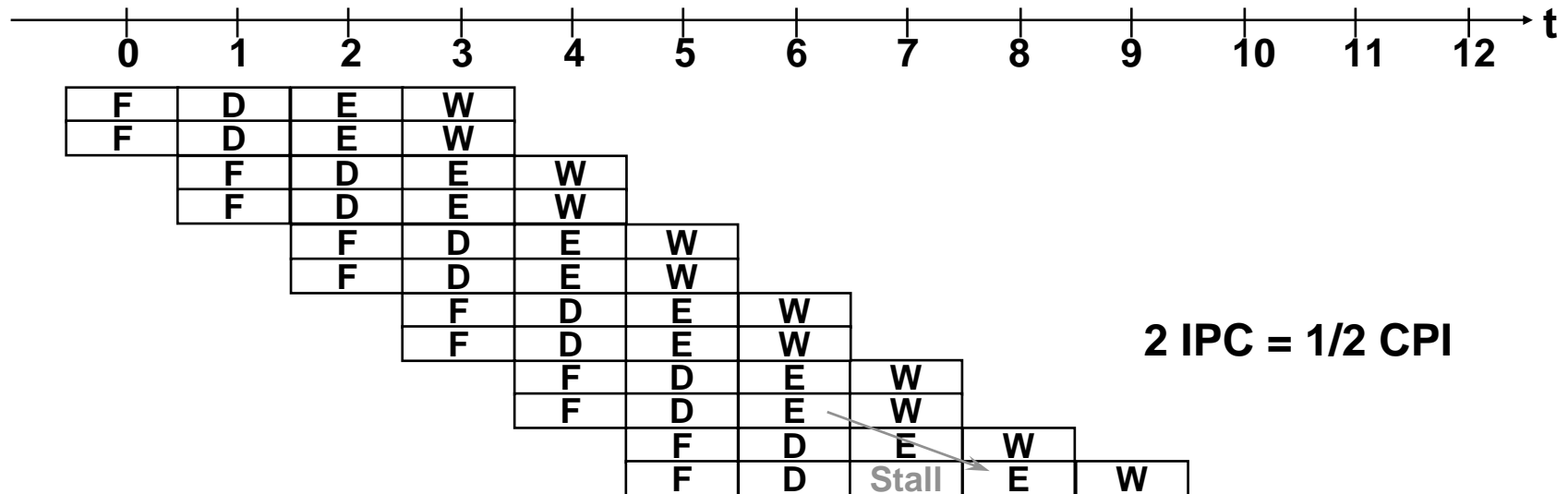
Pipeline Stalls

- But there are “stalls” in the pipeline
 - “Data Hazards”: Data flow dependency (instructions output/input)
 - » Solved by: bypasses, renaming
 - “Control Hazards”: Control flow dependencies
 - » Solved by branch prediction
 - “Structural Hazards”: Limited resources
 - Other (Cache misses, long latency instructions, page faults....)



Super Scalar

- Performs more in a single cycle



- Ideally, can multiply the throughput
 - But stall occurs more frequently

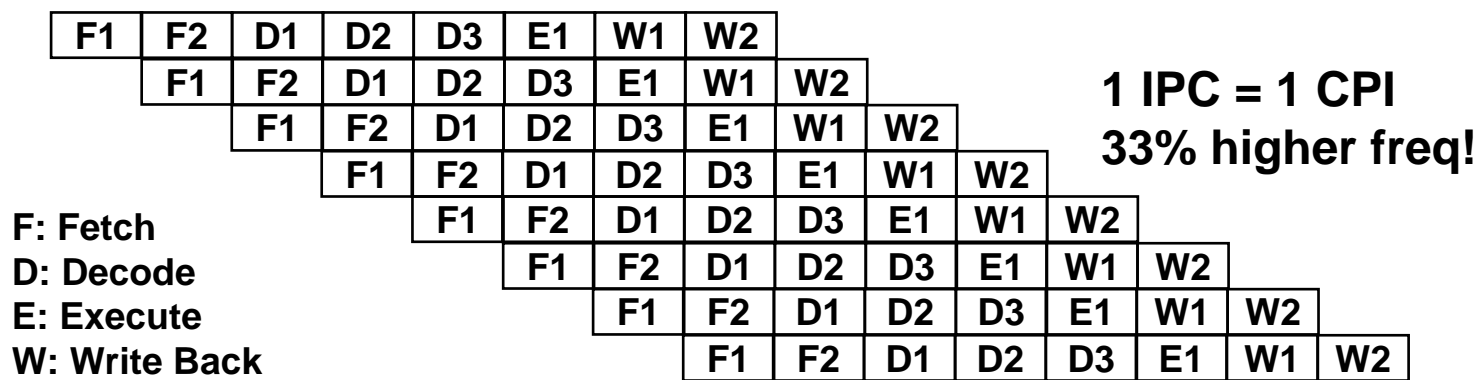
Examples
Intel Pentium® Proc.
Alpha 21164

Super Pipeline

- Split to shorter stages - allows higher frequency

Old clk = 0 1 2 3 4 5 6 7 8 9 10 11 12

New clk = 0 1 2 3 4 5 6 7 8 9 10 11 12



- Ideally, can (again) multiply the throughput, but
 - Stall penalties do not scale (e.g., control flow stall, cache misses)
 - Clock setup/hold reduces net cycle time - each instruction takes longer!
- ⇒ In the example above: 2X stages, but performance gain is <33%

Examples:
Intel Pentium® II/III/4

Out Of Order Execution

- **In Order Execution:** instructions are processed in their program order.
 - Limitation to potential Parallelism.
- **OOO:** Instructions are executed based on “*data flow*” rather than program order

Before: src -> dest

```

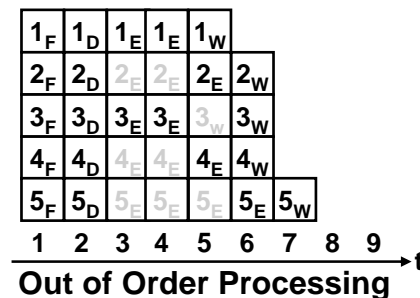
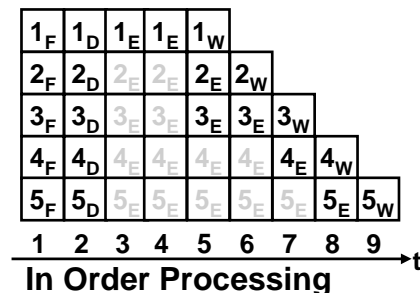
(1) load    (r10), r21
(2) mov     r21, r31      (2 depends on 1)
(3) load    a, r11
(4) mov     r11, r22      (4 depends on 3)
(5) mov     r22, r23      (5 depends on 4)
  
```

After:

```

(1) load    (r10), r21;   (3) load    a, r11;
    <wait for loads to complete>
(2) mov     r21, r31;     (4) mov     r11, r22;
    (5) mov     r22, r23;
  
```

- Usually highly superscalar



Examples:

Intel Pentium® II/III/4
Compaq Alpha 21264

In Order vs. OOO execution.

Assuming:

- Unlimited resources
- 2 cycles load latency

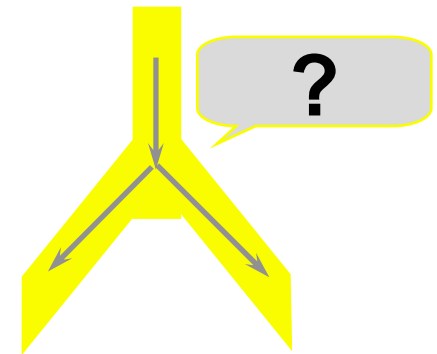
Out Of Order (cont.)

- **Advantages**
 - Help exploit *Instruction Level Parallelism* (ILP)
 - Help cover latencies (e.g., cache miss, divide)
 - Artificially increase the Register file size (i.e. number of registers)
 - Superior/complementary to compiler scheduler
 - » Dynamic instruction window
 - » Make usage of more registers than the Architecture Registers
- **Complex microarchitecture**
 - Complex scheduler. Involves also
 - » Large instruction window
 - » Speculative execution
 - Requires reordering back-end mechanism (*retirement*) for:
 - » Precise interrupt resolution
 - » Misprediction/speculation recovery
 - » Memory ordering

Speculation

Branch Prediction

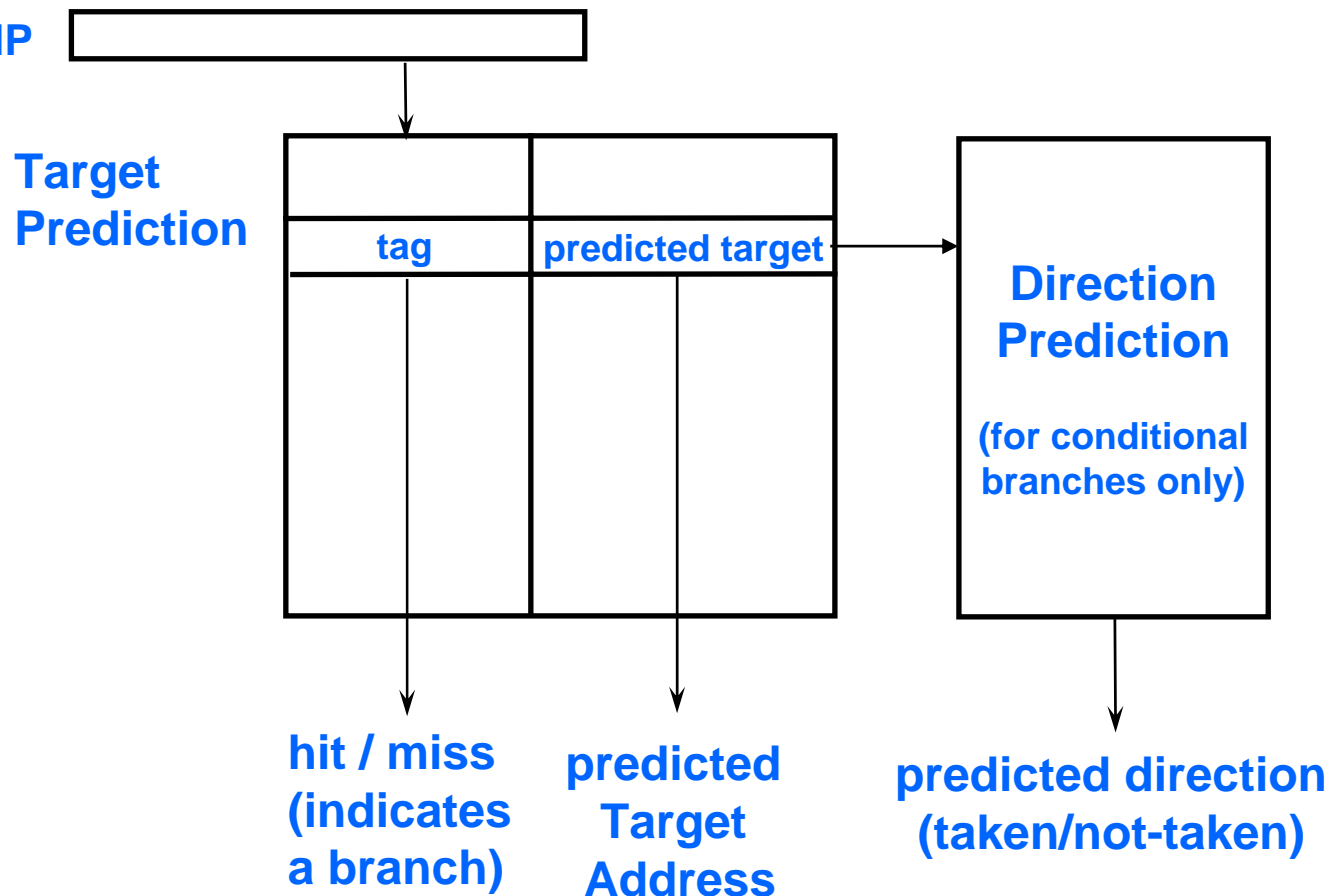
- Goal - ensure enough instruction supply by correct prefetching
- In the past - prefetcher assumed *fall-through*
 - Lose on unconditional branch (e.g., call)
 - Lose on frequently taken branches (e.g., loops)
- Branch prediction
 - Predicts whether a branch is *taken/not taken*
 - Predicts the branch target address
- Misprediction cost varies (higher w/ increased pipeline length)
- Typical Branch prediction rates: ~90%-96%
 - ➔ 4%-10% misprediction,
 - ➔ 10-25 branches between mispredictions
 - ➔ 50-125 instructions between mispredictions
- Misprediction cost increased with
 - Pipeline depth
 - Machine width
 - » e.g. 3 width x 10 stages = 30 inst flushed!



Target Array + Direction Prediction

- Target and direction are predicted separately
- Tag may be partial

Branch IP



Speculative Execution

- **Execution of instructions from a predicted (yet unsure) path**
Eventually, path may turn wrong.
- **Advantages:**
 - Ensure instruction supply
 - Allow large scheduling window (for out of order)
- **Issues:**
 - Misprediction cost
 - Misprediction recovery

Cache - Motivation & Principle

- Memory consumption is growing about 2X every 2 years
 - Typical size: (Y2000) 64M-128M, (Y2002) 128M-256M
- CPU speed grows faster than memory and buses
 - CPU/Bus grew from 1:1 to 6:1, and still growing

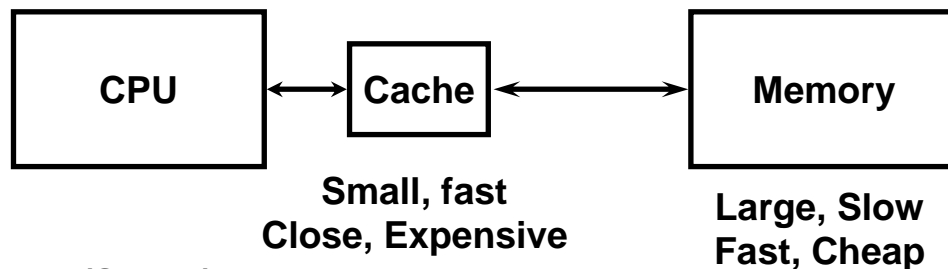
486	Pentium	P-II	P-III	P4
25-66MHz 33MHz	66-233MHz 66MHz	200-450MHz 66-100MHz	0.5-1.33GHz 133-200MHz	1.4-2.4GHz 400MHz

- Memory: DRAM: 60-100ns (“10-16MHz”), Cost: <10\$ per 1M
SRAM is faster but much more expensive

⇒ *Memory becomes the bottleneck for both instructions and data!*
Slow or expensive

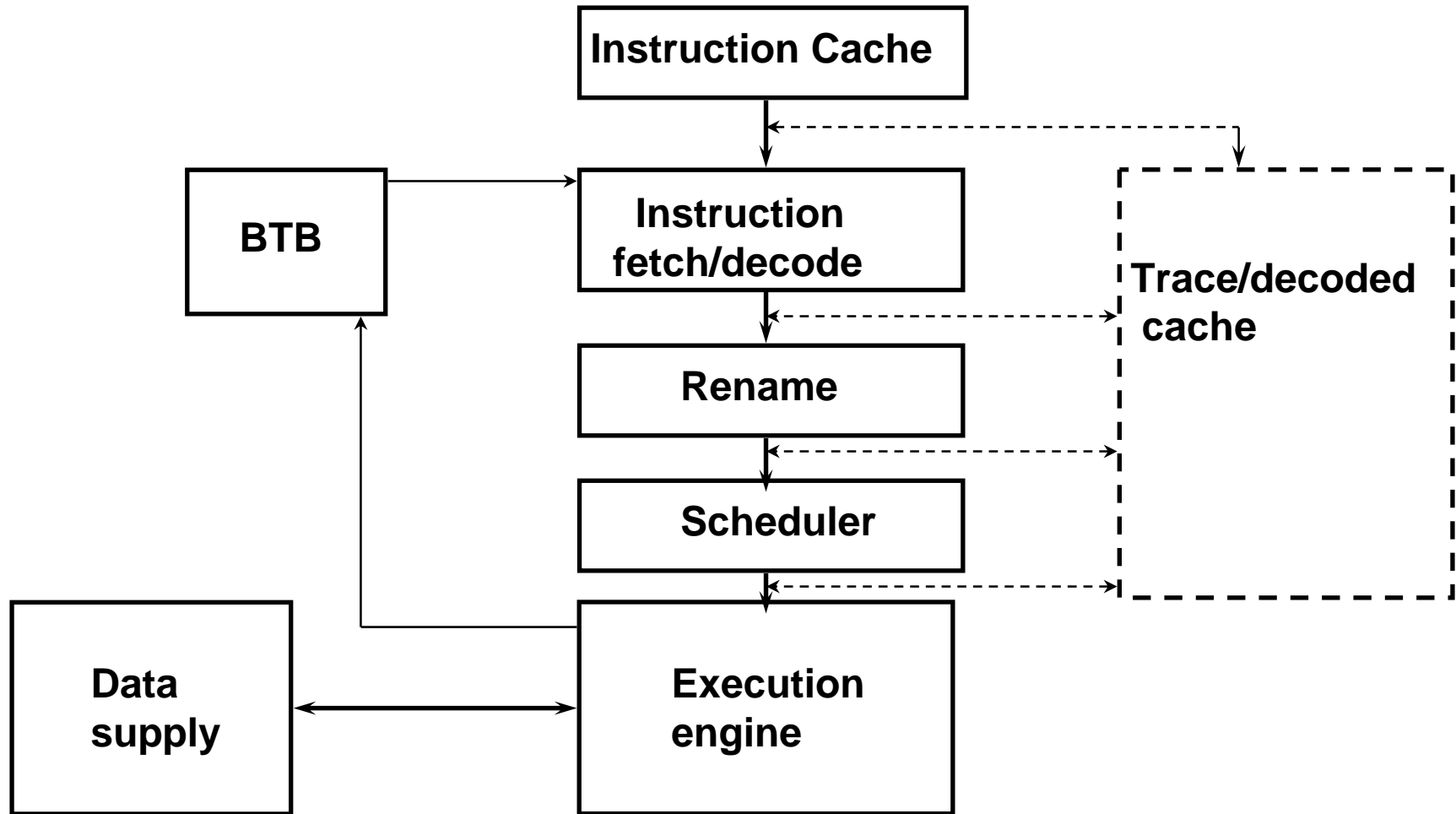
- Solution: Cache - A Small, Fast, Close memory
 - Serves as a buffer between CPU and main memory

- Contains copy of a portion of the main memory
 - Small in size
 - Dynamically changed



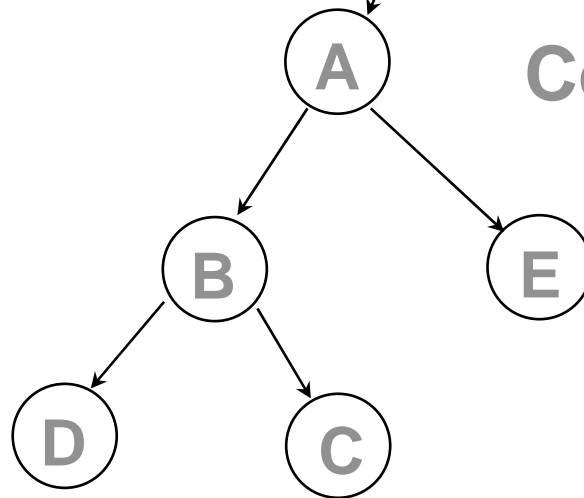
- Exploit space and time locality:
 - Code is fetched sequentially (Space)
 - Code is re-executed (loops, procedures) (Time)
 - Access close or previous data (Space, Time)

The Generic Processor



Fetch bandwidth

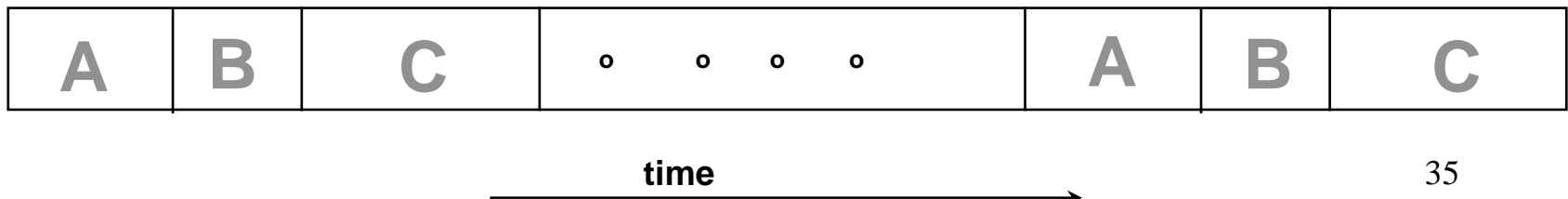
example



Control flow graph

A, B, C are instruction blocks

Dynamic instruction stream

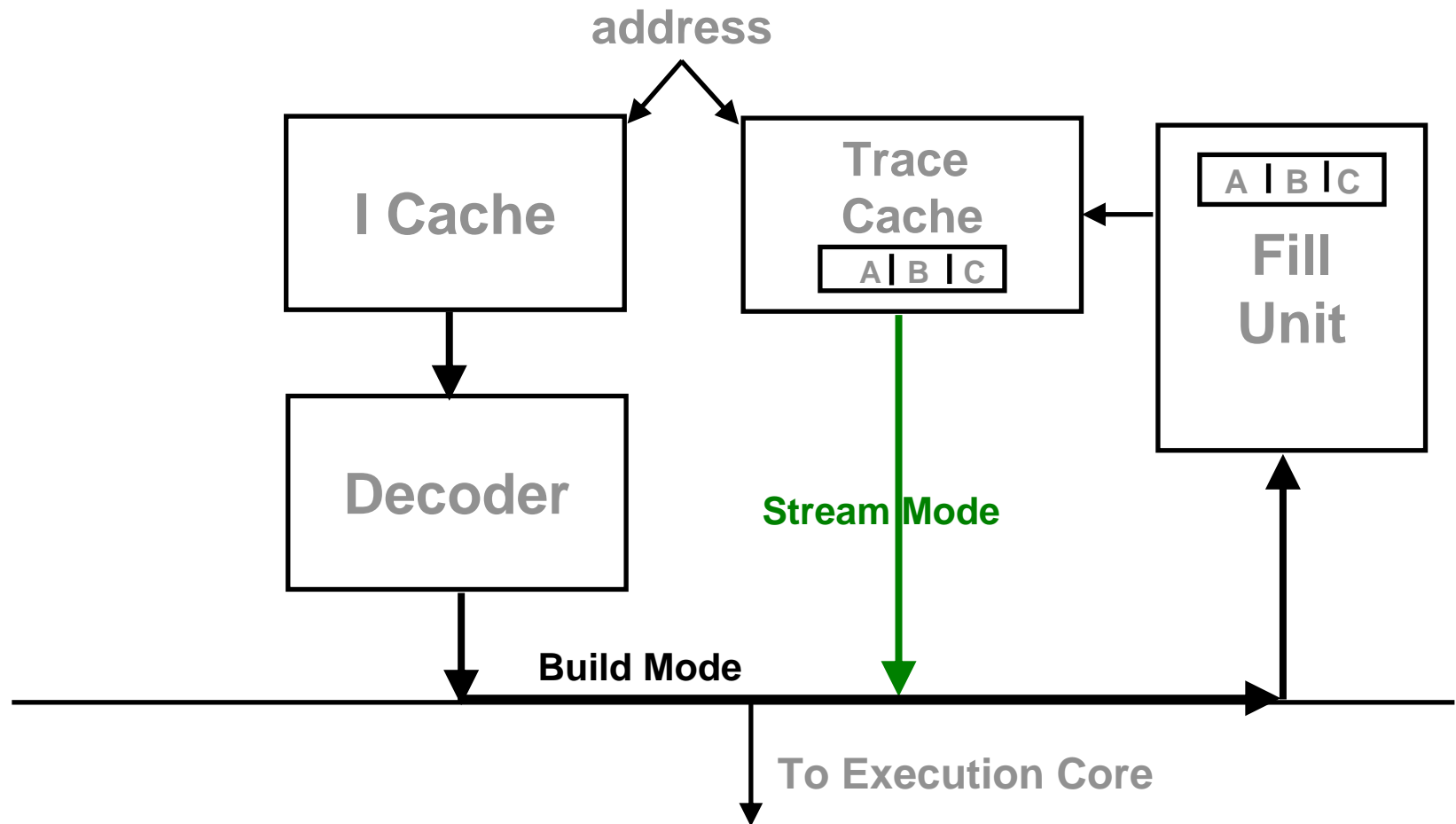


Trace Cache Concept

- **Hold in the “instruction” cache the dynamic stream of the executed instructions**

=> Trace cache acts as “branch predictor” + wide instructions supplier

Trace Cache Overview



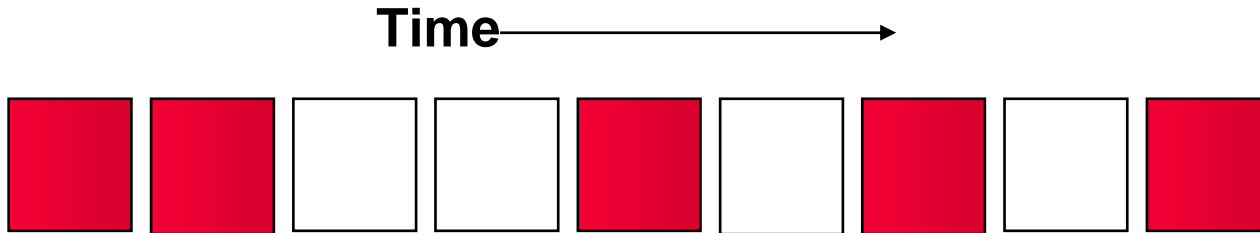
Trace cache line



- **Tag: identifies starting address of trace**
- **N instructions (potentially decoded)**
- **Next address: next fetch address**
- **path info: branch flags (T, NT), number of branches, trace ends w/ branch?,...)**

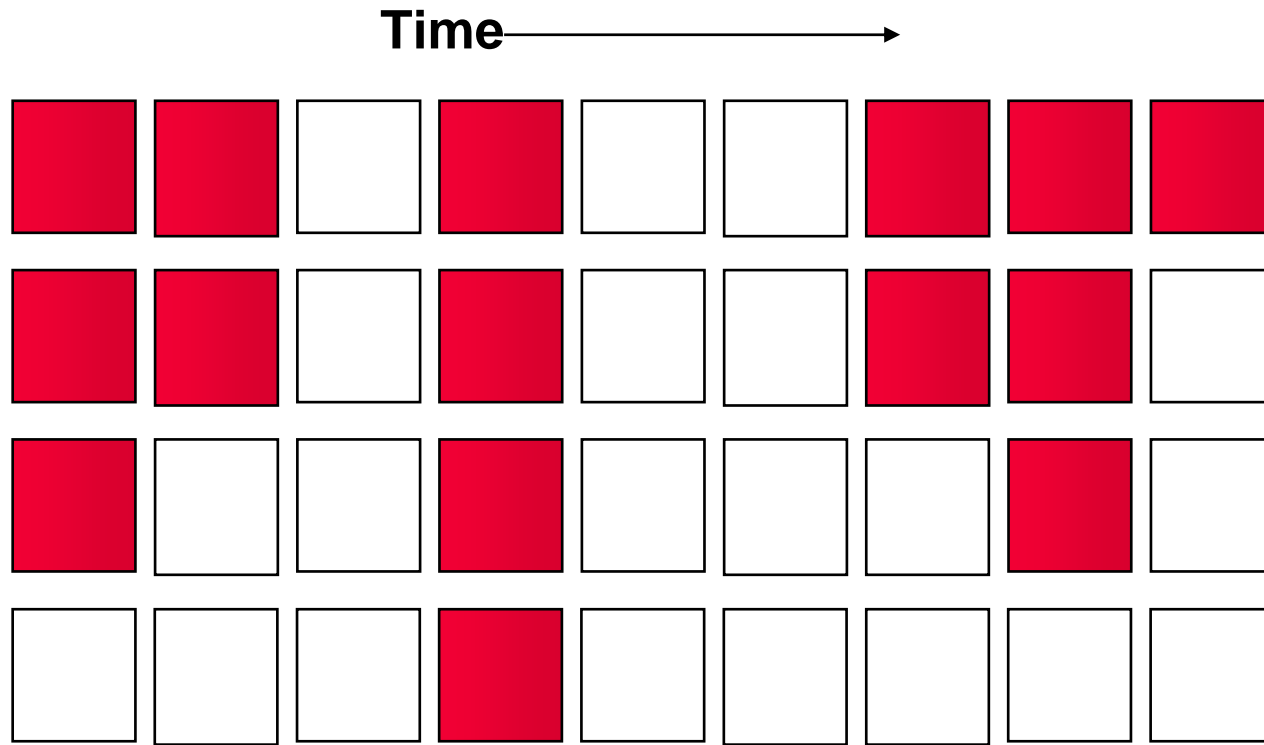
Threads

Scalar Execution



Dependencies reduce throughput/utilization

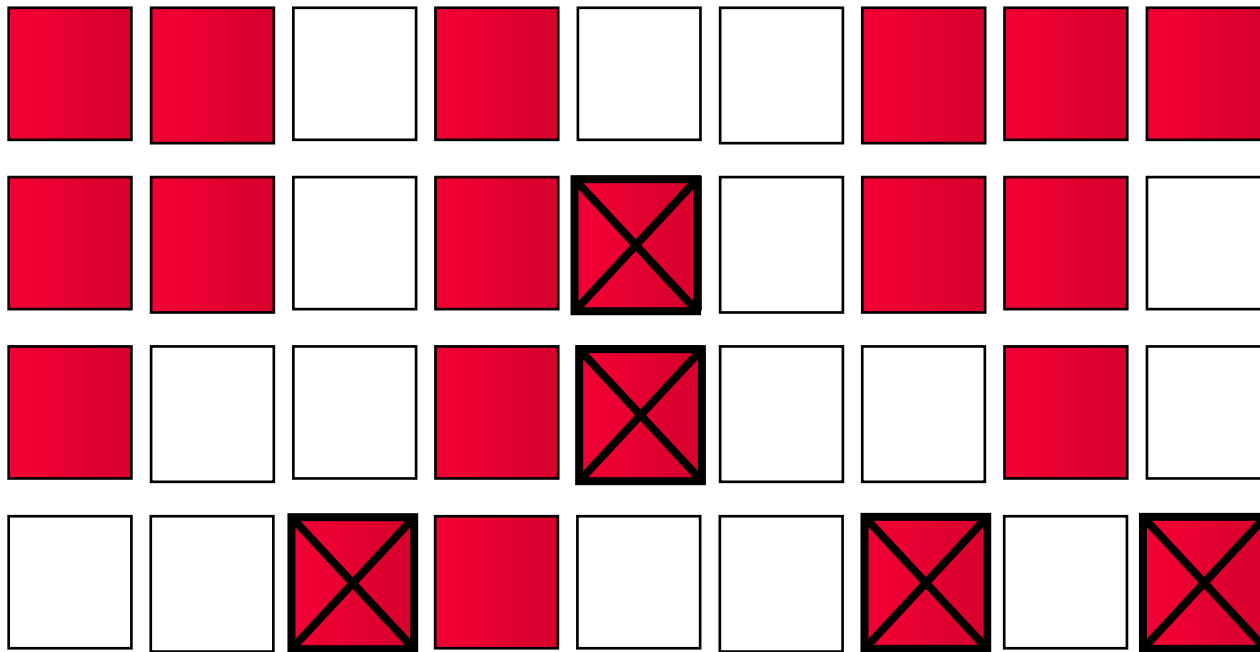
Superscalar Execution



Generally increases throughput, but decreases utilization

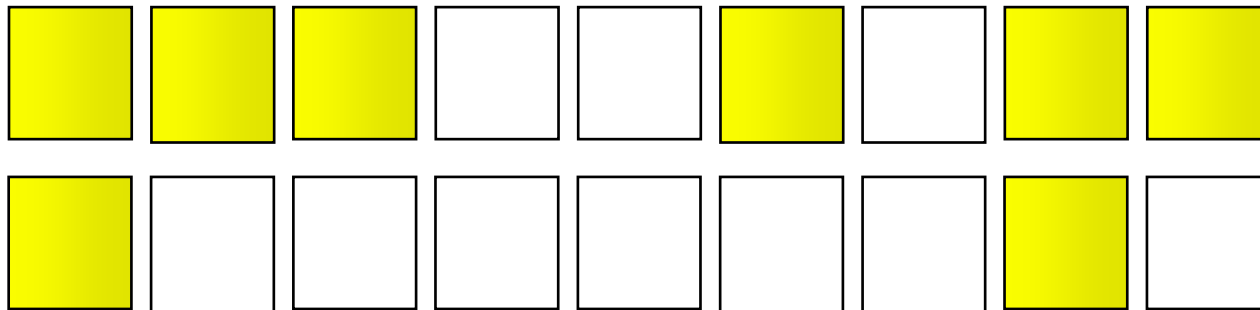
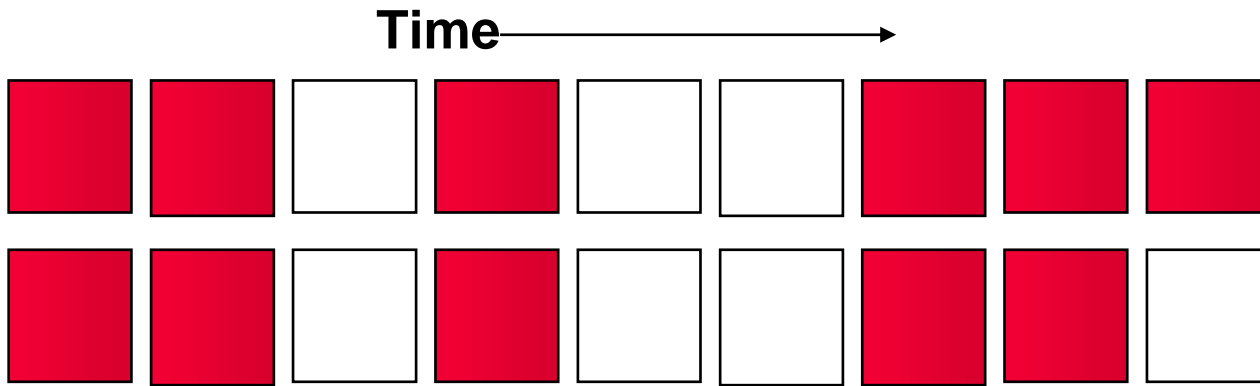
Predication

Time →



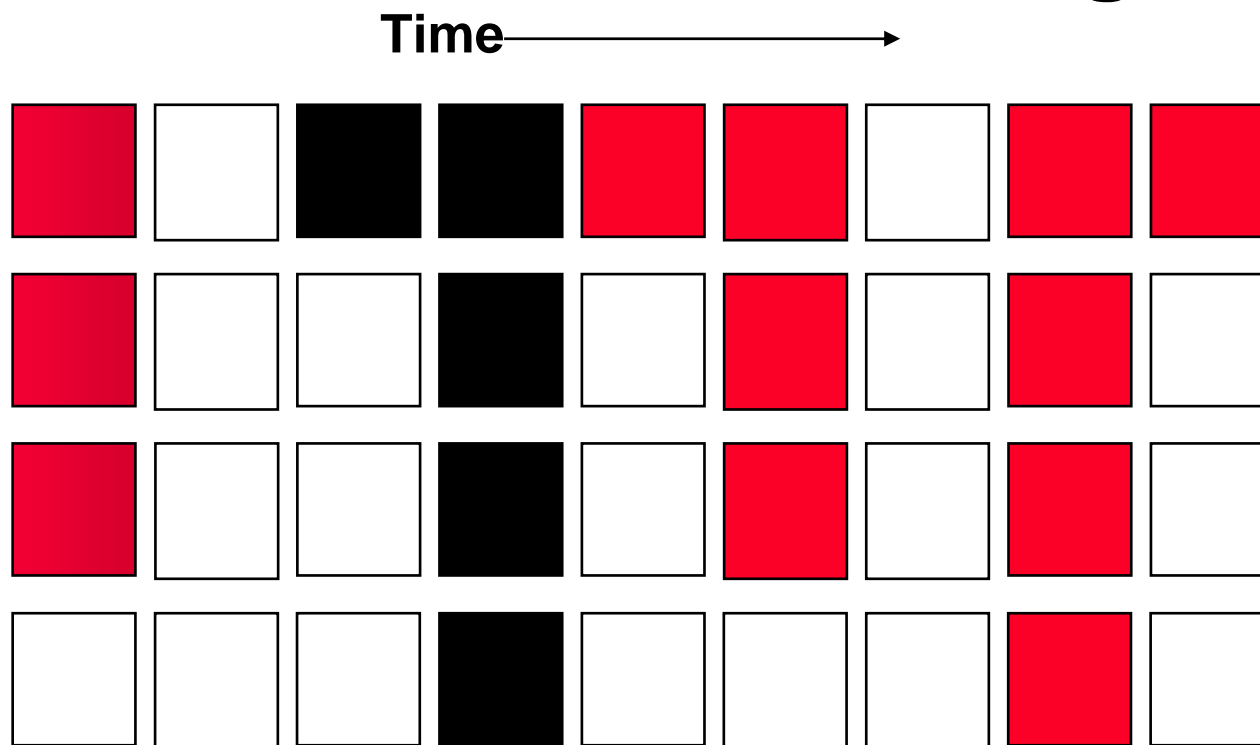
Generally increases utilization, increases throughput less
(much of the utilization is thrown away)

CMP – Chip Multi-Processor



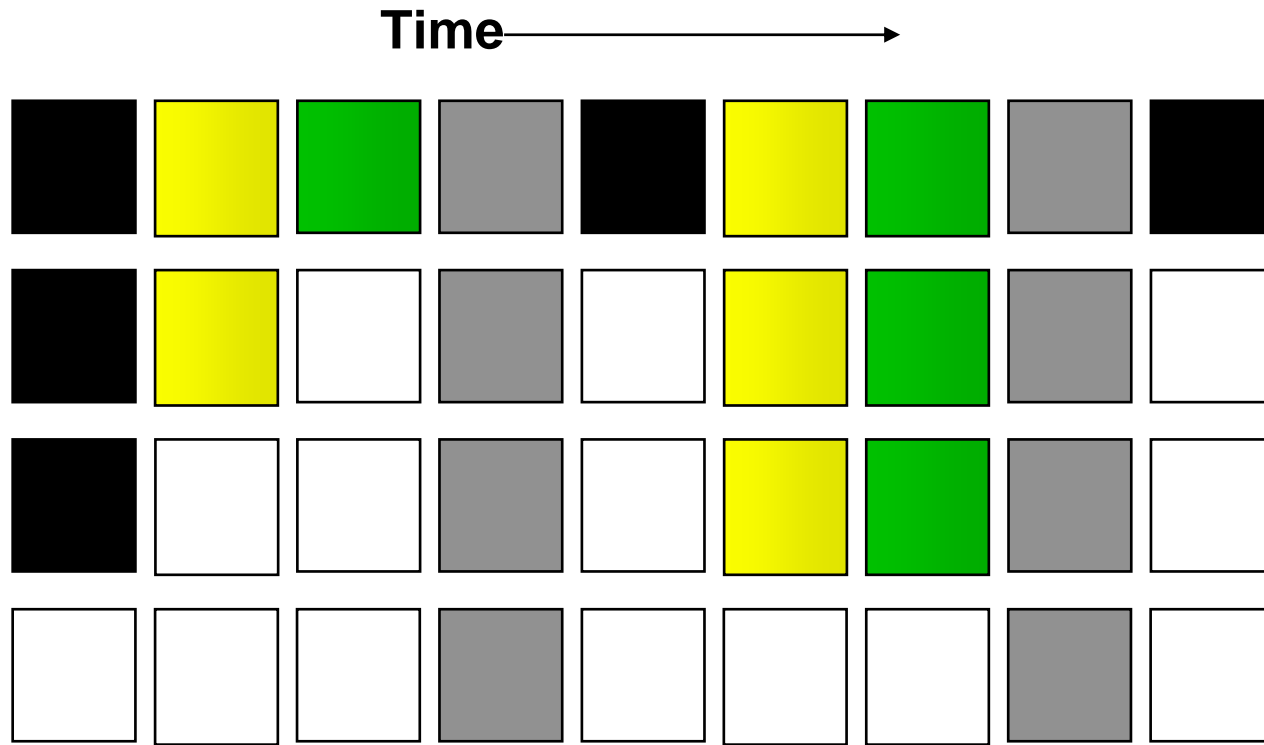
Low utilization / higher throughput

Blocked Multithreading



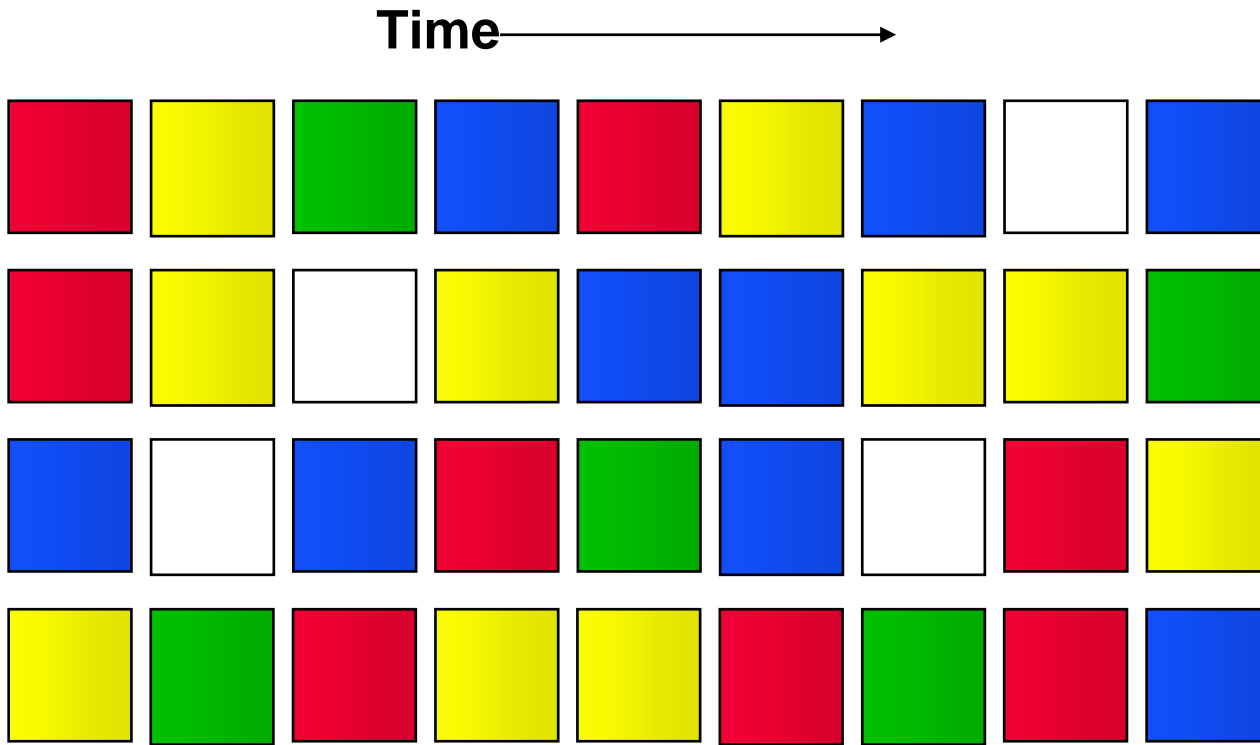
May increase utilization and throughput, but must switch when current thread goes to low utilization/throughput section (e.g. L2 cache miss)

Fine Grained Multithreading



Increases utilization/throughput by reducing impact of dependences

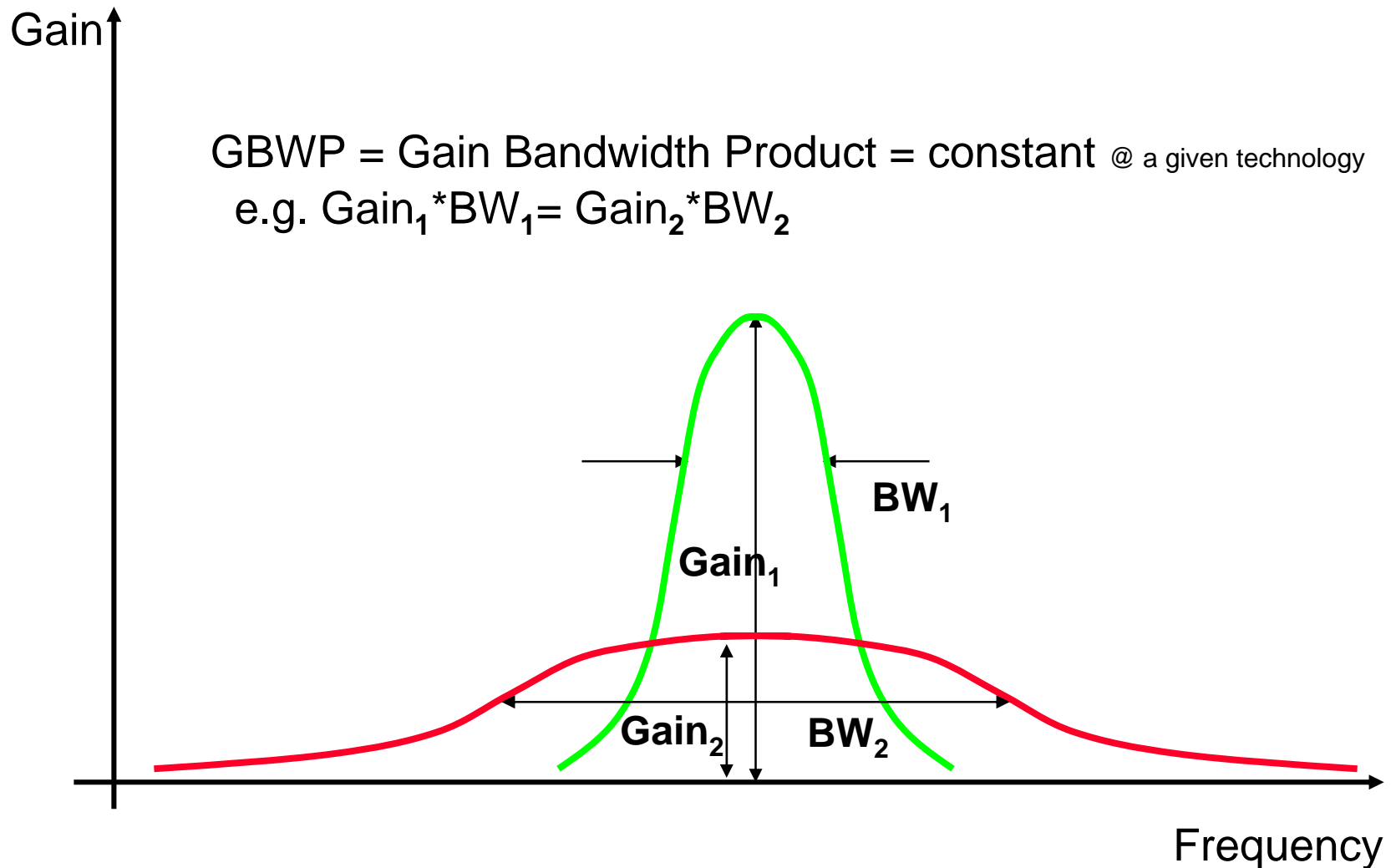
Simultaneous Multithreading



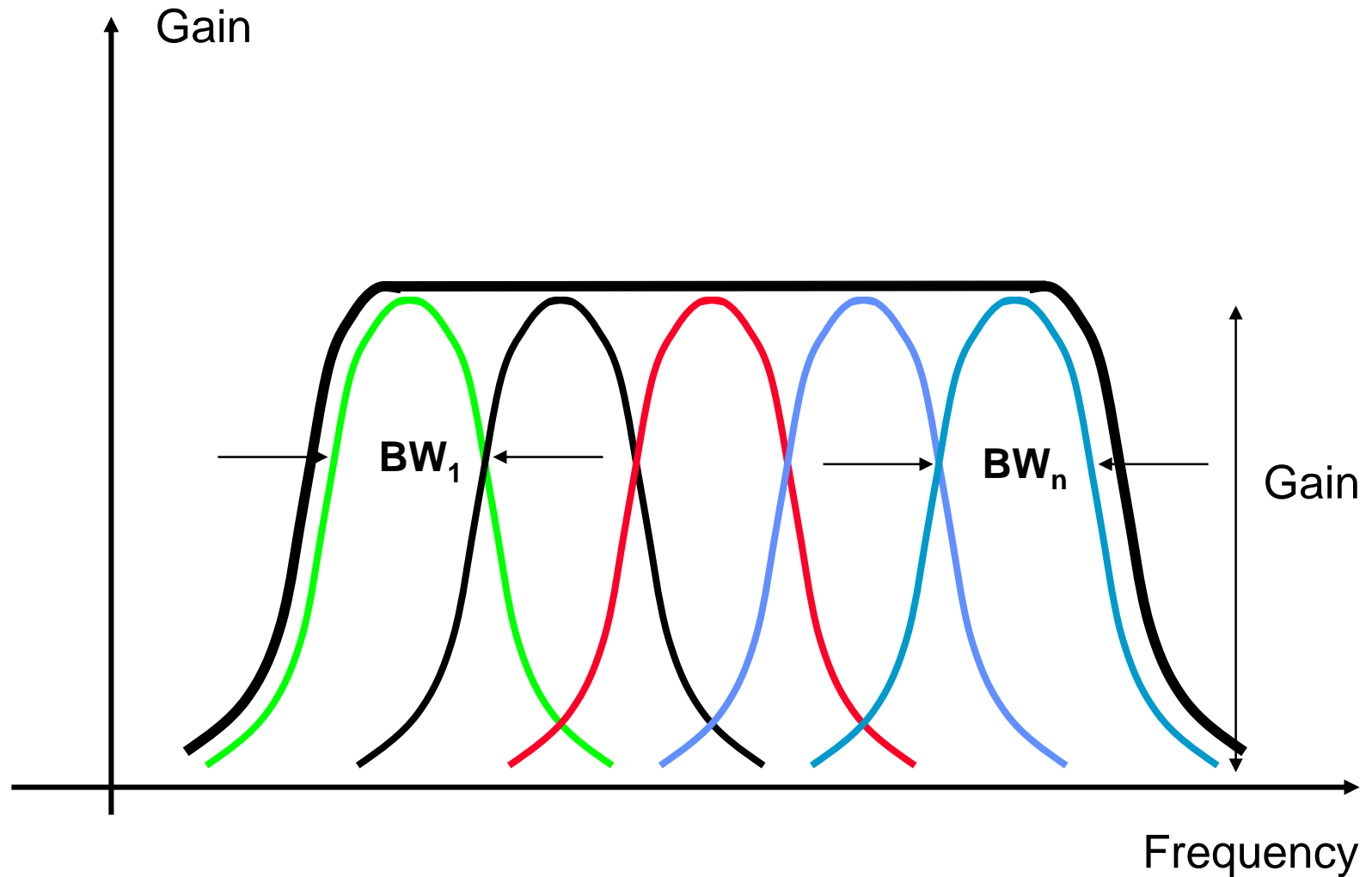
Increases utilization/throughput

Future


Analog Circuit Paradigm



Analog Circuit Paradigm (cont.)

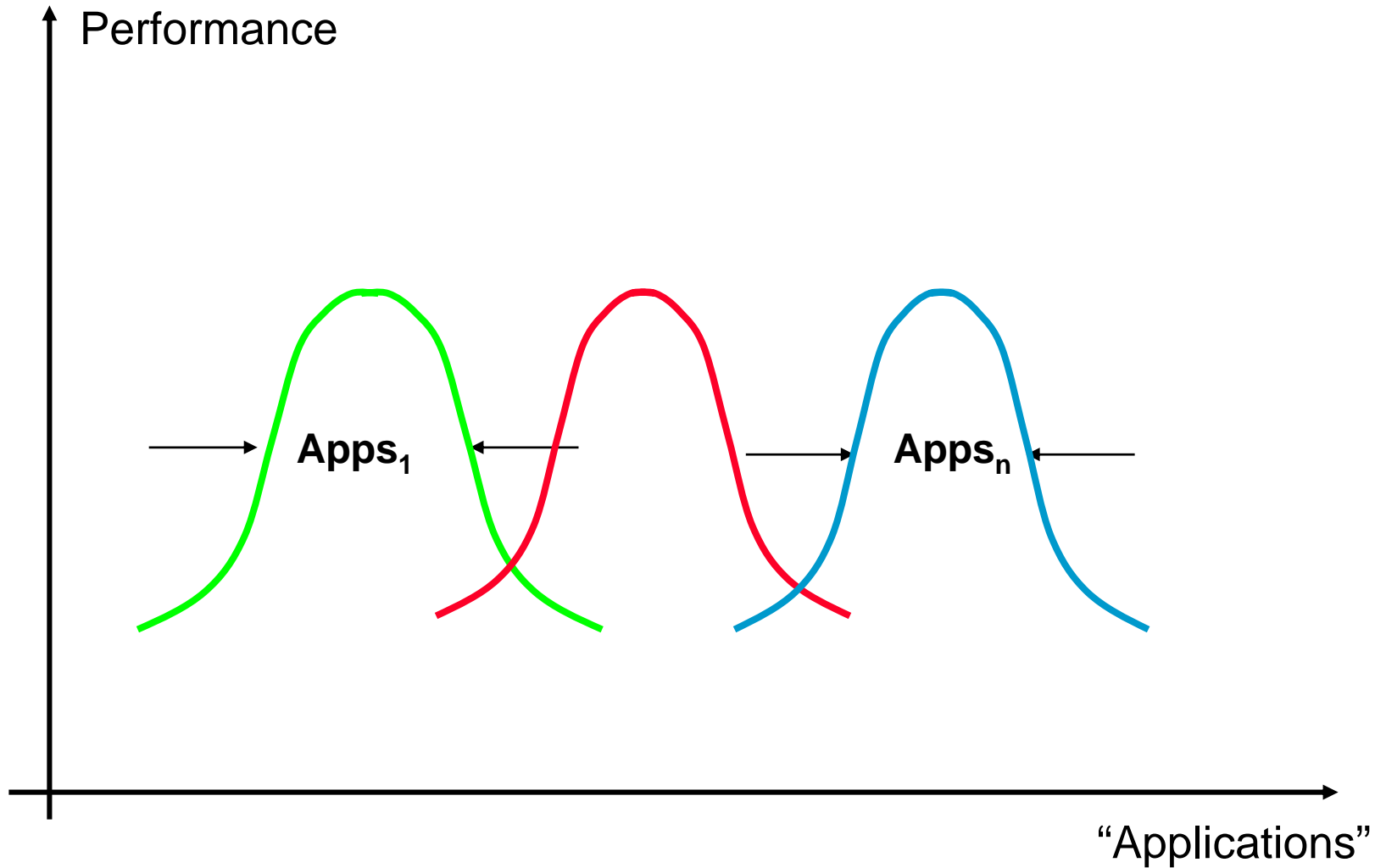


“Theory”

- **Analog Gain Bandwidth Product (GBWP) is constant for a specific technology, this is also true for other “environments”...**
- **A computer structure can excel in performance for a specific application set but not at all applications (also true for benchmarks)**
- **a person can excel in several areas but not at all...**
- 

examples: benchmarks, application in coming foils
people....

Tuning for Applications



Provide Specialized “efficient” MIPS

- Find a way to support the new performance requirements via an efficient “mechanism”
- A tailored solutions (to a specific application set) can provide an “efficient” MIPS
via INTEGRATION, how?

The Need

the environment

- These days is the PC's 20th birthday
 - 835 Million PC sold 1981-2001
 - 138 million PCs in year 2001_(IDC), 10X number of cars, 1.5X of television sold annually
 - 2.2 Billion Email a day, 10X of the first class mail
 - 400 million on line users (200 in Sep99)
 - CPU performance improved ~8000X !!!
- What will be the need for performance in the coming 20 years?
- What will be the technology progress in the coming 20 years? 10 years? 5 years?

Statistics courtesy of Gartner Dataquest, U.S. News & World Report, Jupiter Internet Population Model, and NUA Internet Surveys

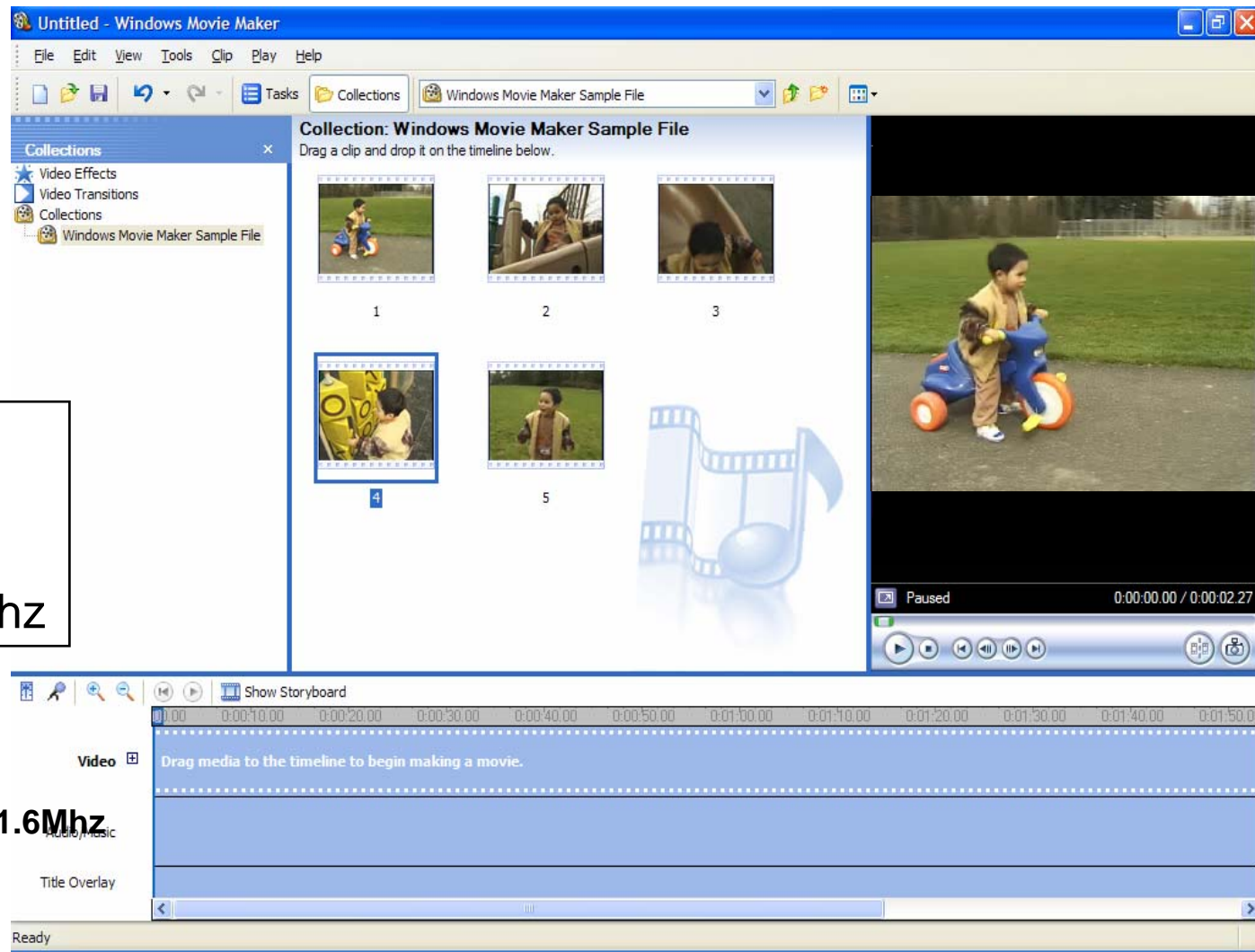
Windows XP examples that needs excessive performance:

- Movie Maker Video Indexing
- Video smoothing

Example 1: Movie Maker Video Indexing

320*240, 30fps
4X slower than
real Time on
Centrino™ @1.6Ghz

1980x1080, 30fps
~100X over Cetrino™ @1.6Mhz.



Video smoothing

5 FPS (Jerky)



30 FPS (Enhanced)



**Example 2:
Emulation of:
Video smoothing
Video Enhancement**

352*240 pixels
CPU usage:
70% of Centrino™ @1.6Ghz

**1980x1080, 30fps
~21X over Cetrino™ @1.6Mhz**

The Need

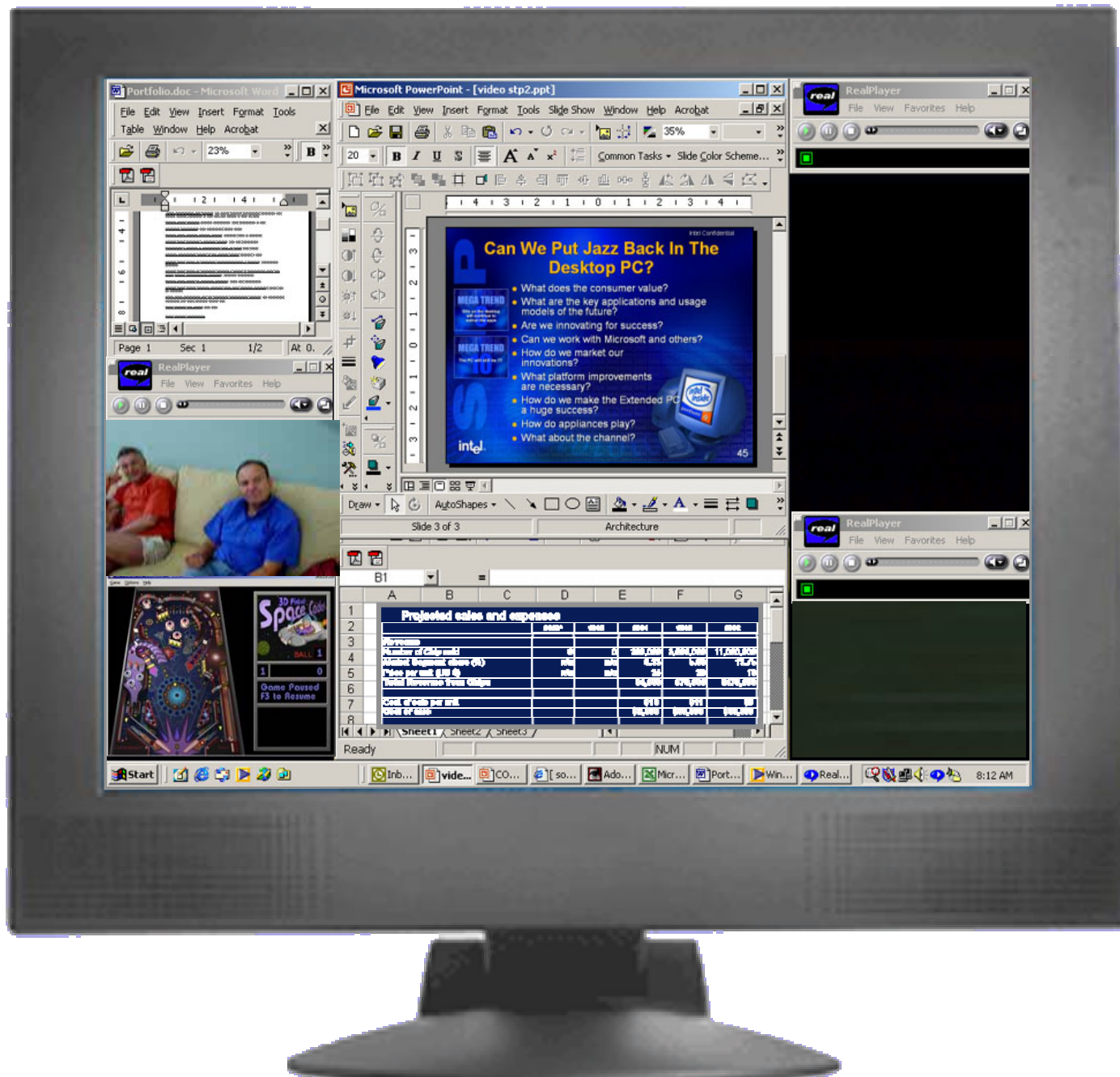


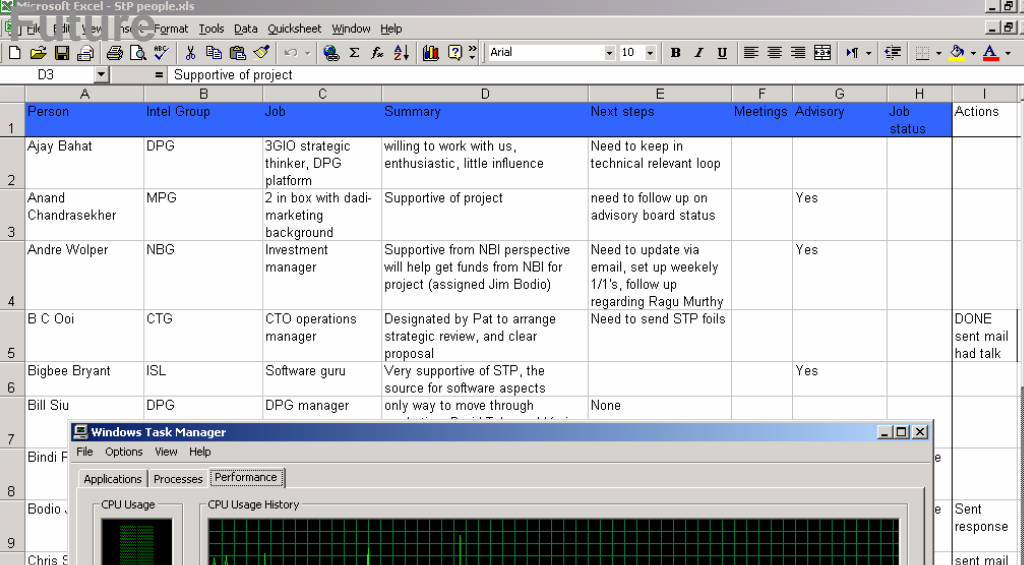
The need: Build a Panorama



M. Brown and D. G. Lowe. Recognising Panoramas. ICCV 2003
Performance: >30min P4 3GHz

Simplified capabilities at Microsoft Digital Image Suite 10 (\$129.95)



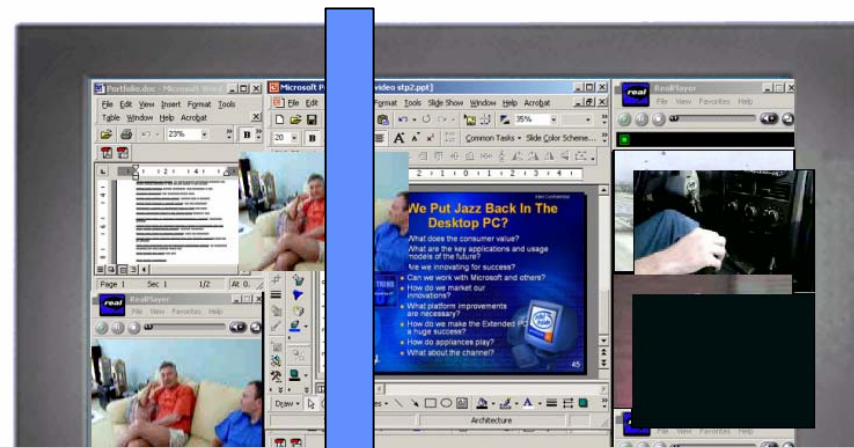


● CPU Usage

an example
(measured on IBM X20)

- Streaming vs. General purpose

Streaming Processing
6 low resolution videos-->
Continuous need for MIPS



Avg.

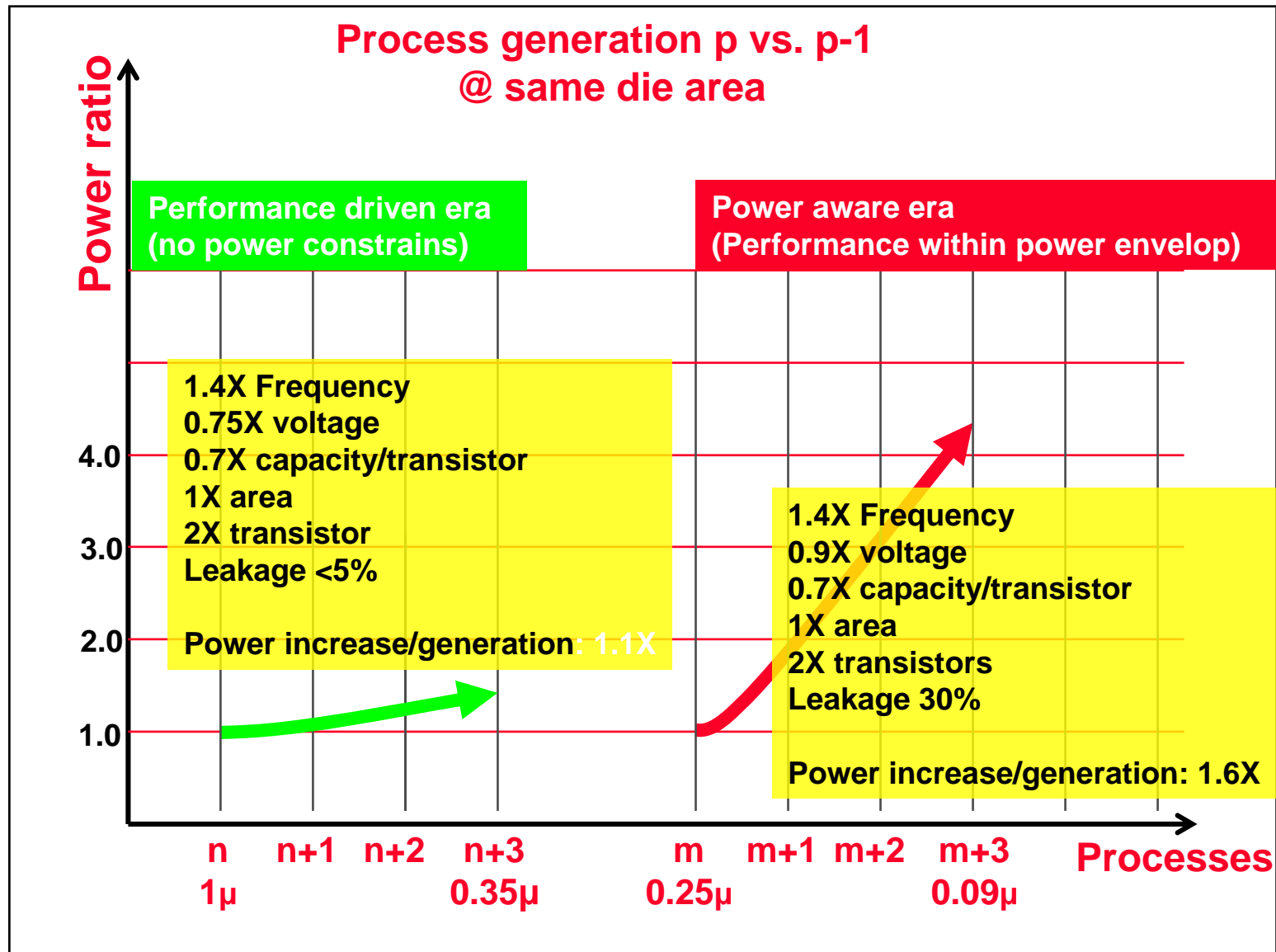
Avg.

General purpose usage
Excel and Outlook -->
Burst need for MIPS



Process trend – the theory (cont)

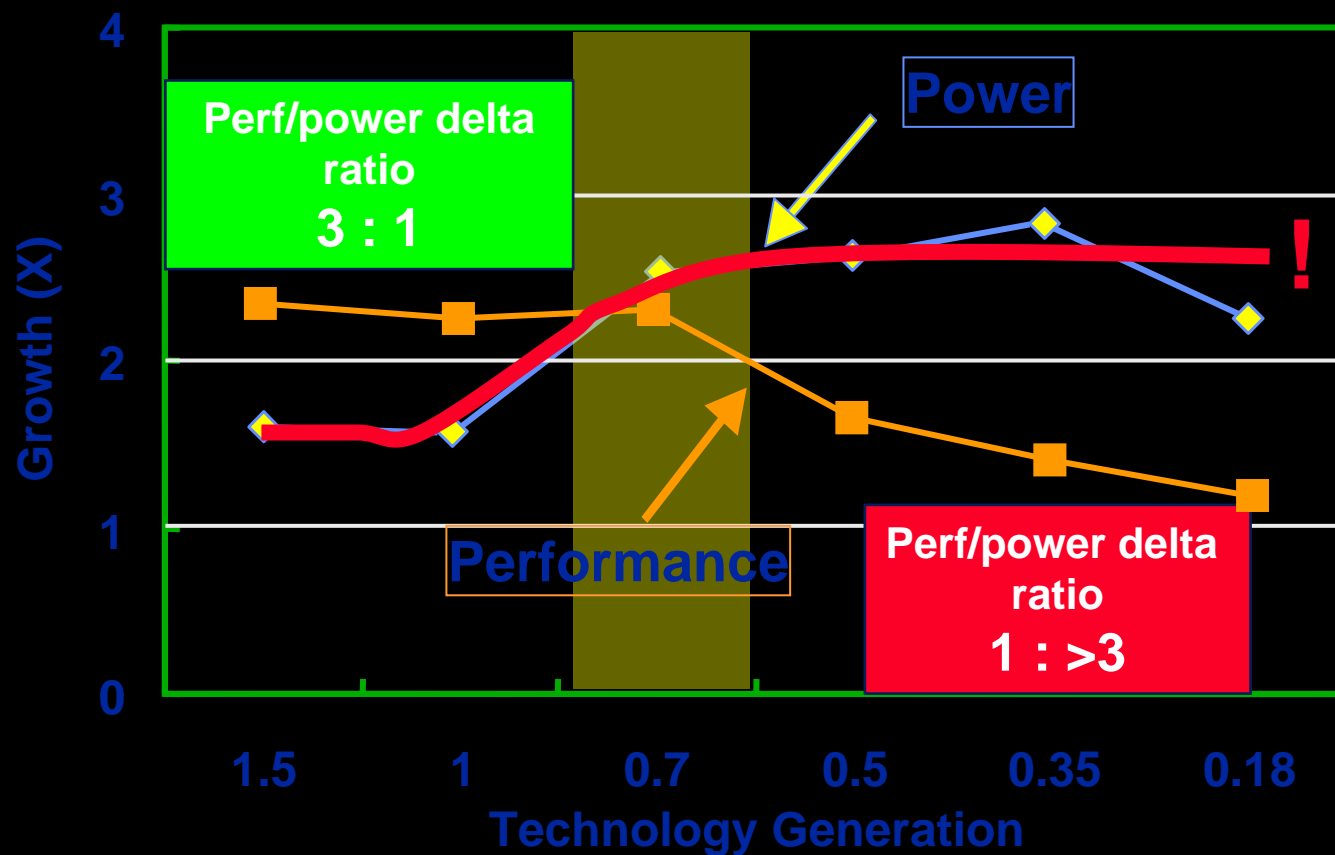
Performance driven era vs. Power aware era



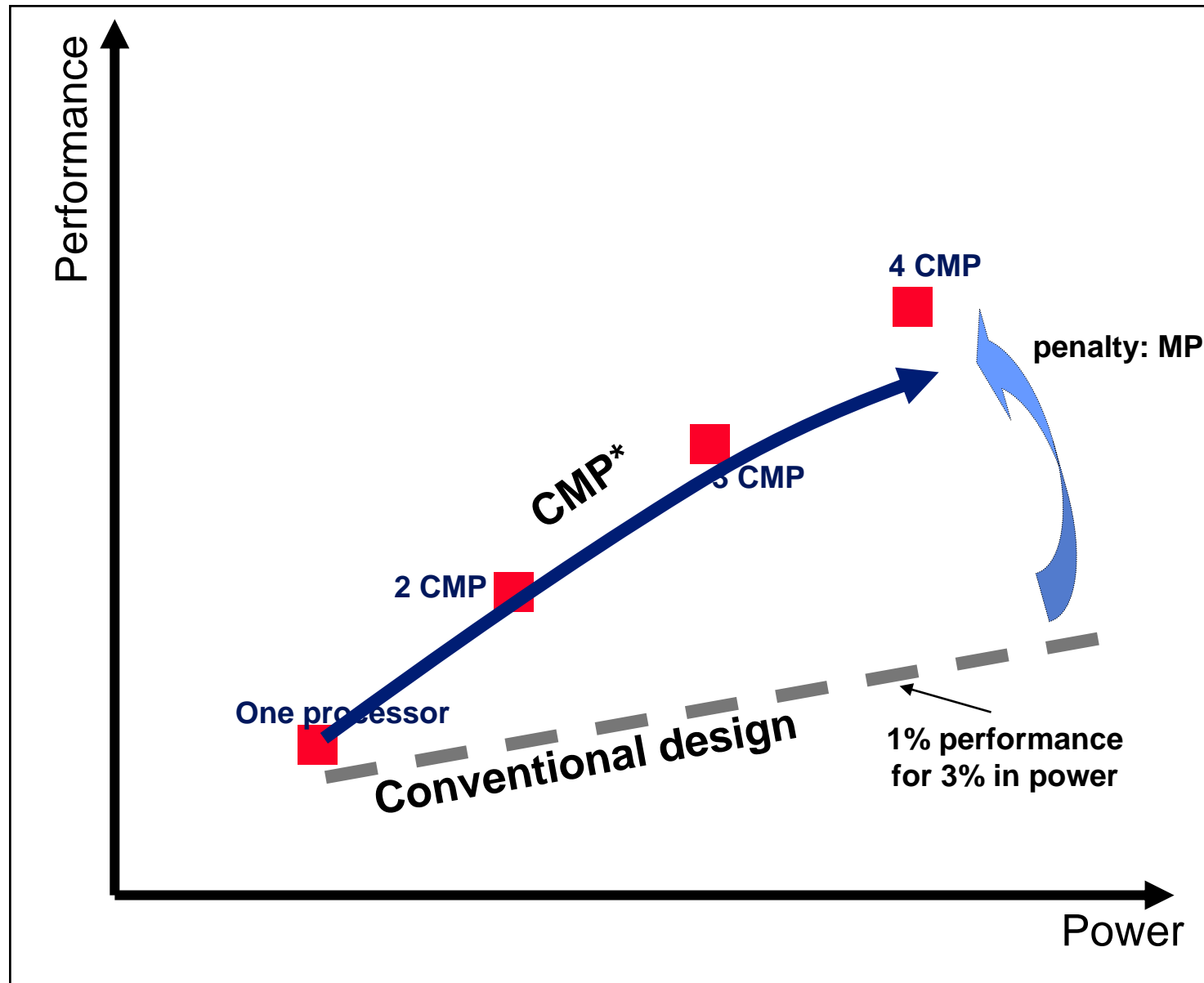
Processor roadmap trend – real life (cont)

Extension of Pollack's Rule (Micro32, 1999)

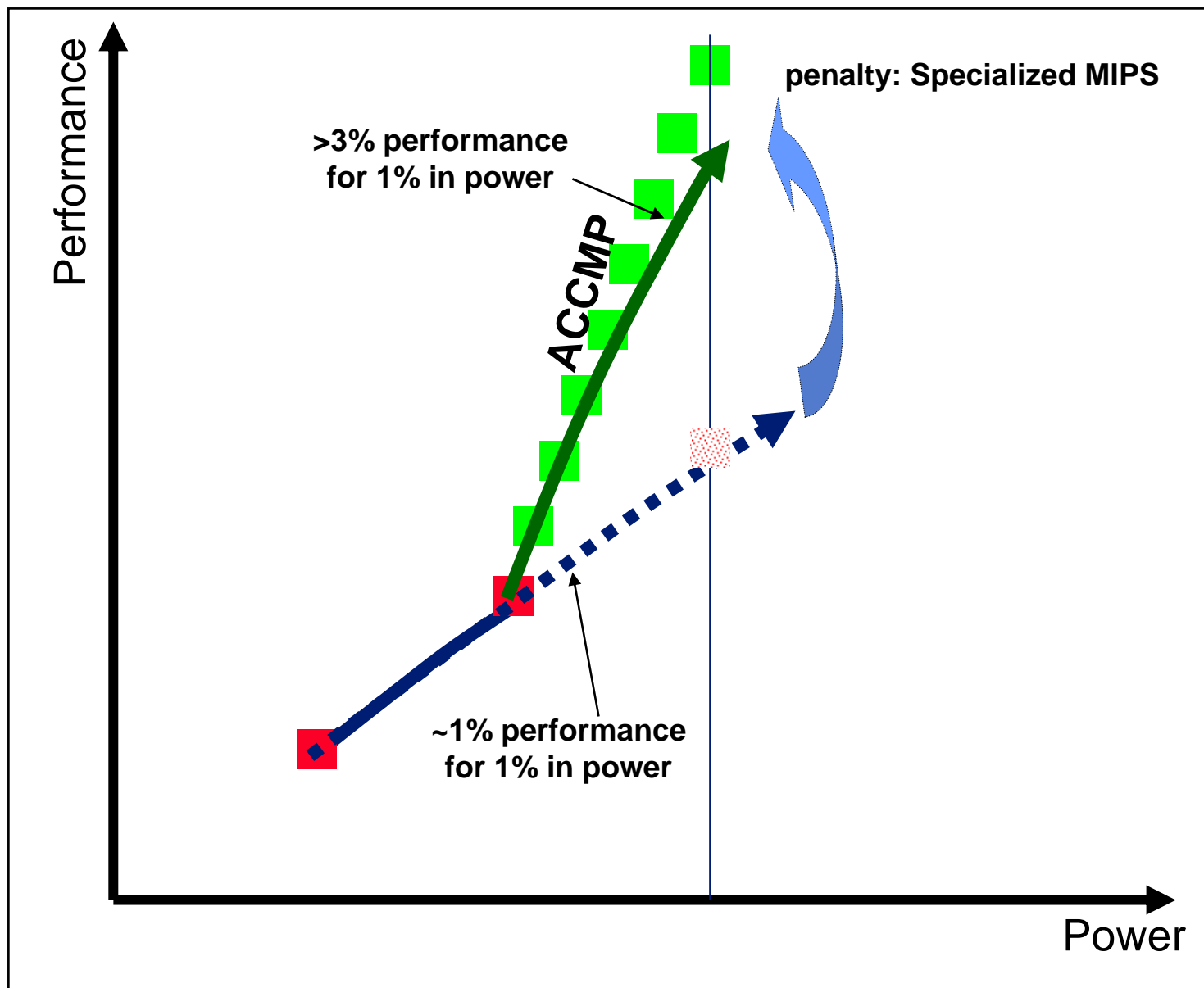
Processor generation k vs. k-1 compacted
@ the same process technology



solution 1: CMP (Chip Multi-Processor)



solution 2: ACCMP (Asymmetric Cluster CMP)

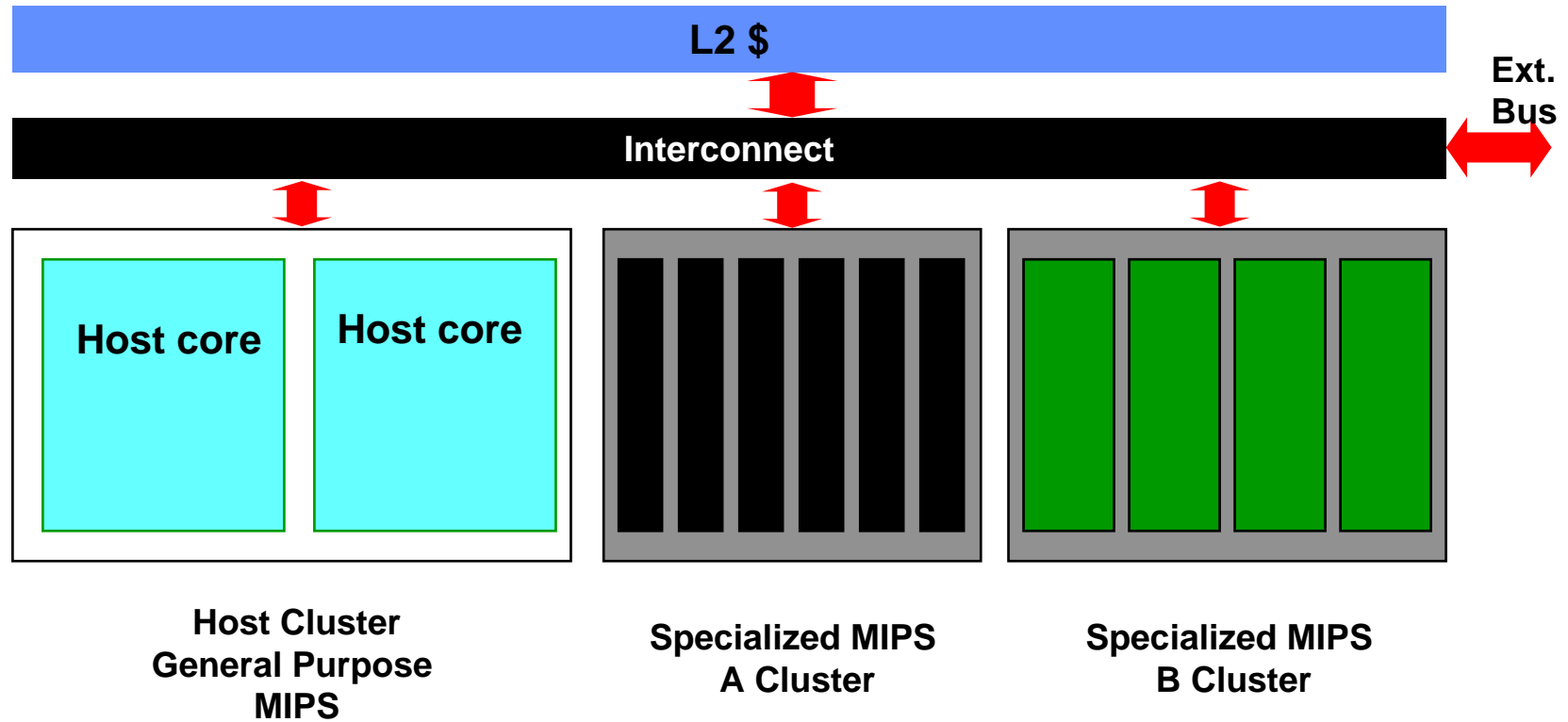


ACCMP

- What is the ACCMP?
 - On Die Asymmetric Clusters of cores
 - Efficient specialized MIPS clusters with >3-4X performance/power over GP cores
 - Compatible ISA?
- Penalties
 - Multi-Processing (tasks or threads)
Specialized MIPS

ACCMP is a solution that enables to continue (for a while)
Moore's performance law within the power envelop

ACCMP



Future - Processors

- **applications need**
- **Specialized MIPS**
- **Detached from the CPU core**
- **Different engines**
- **Mixture of Programmable and fixed function**
- **?**