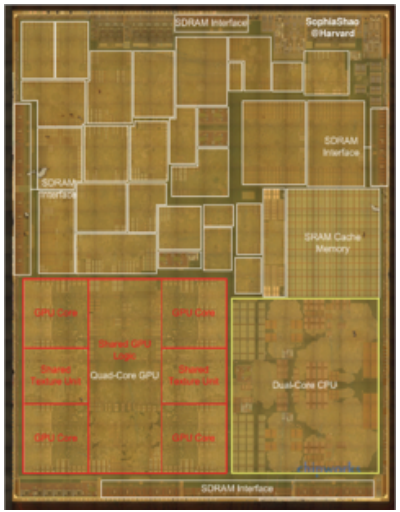
Accelerator-level Parallelism: Mobile SoCs as Harbinger of the Future

Mark D. Hill, Wisconsin & Vijay Janapa Reddi, Harvard

ISPASS FastPath, March 2019

Outline

I. From ILP to Accelerator-level Parallelism
II. Mobile SoCs as Harbinger
III. Gables ALP SoC Model



Thanks for Google Mobile Silicon Group internship
Ideas are the authors' & not necessarily Google's

Accelerator-level Parallelism: Mobile SoCs as Harbinger of the Future

Mark D. Hill [1] and Vijay Janapa Reddi [2]
[1] University of Wisconsin-Madison
[2] Harvard University

Abstract:
This talk will first discuss how computer systems are transitioning from homogeneous parallelism to heterogeneity: ILP, TLP, and new Accelerator-level Parallelism (ALP). It will then discuss systems on a chip (SoCs) for mobile computing, and why they may be a harbinger of computer systems's ALP future. It will conclude presenting the Gables model largely developed at Google's Mobile Silicon Group for HPCA 2019's Industrial Session. Gables seeks to make SoC selection and design more scientific via extending Roofline and bottleneck analysis to provide the first answers, not the final answers. Mark D. Hill will present the work.

Biography:
Mark D. Hill (http://www.cs.wisc.edu/~markhill) is John P. Morgridge Professor and Gene M. Amdahl Professor of Computer Sciences at the University of Wisconsin-Madison, where he also has a courtesy appointment in Electrical and Computer Engineering. His research interests include parallel-computer system design, memory system design, and computer simulation. He is a fellow of IEEE and the ACM. He serves as Chair of the Computer Community Consortium (2018-19) and served as Wisconsin Computer Sciences Department Chair 2014-2017. Hill has a PhD in computer science from the University of California, Berkeley.

Outline w/ Key Points

I. From ILP to Accelerator-level Parallelism

- ALP = Parallelism among workload components concurrently executing on multiple accelerators (IPs)

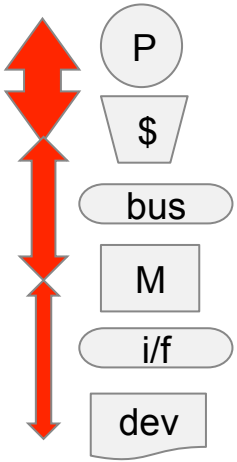
II. Mobile SoCs as Harbinger

- Mobile SoCs already have ALP
- Some Pitfalls already emerging

III. Gables ALP SoC Model [HPCA'19 Industrial Session]

- Some "first answers" to multi-IP questions

A Computer Architecture History

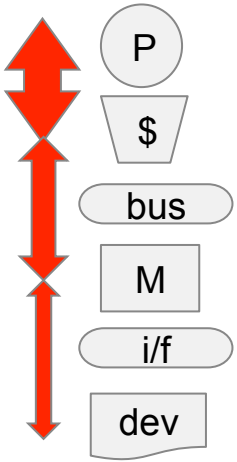


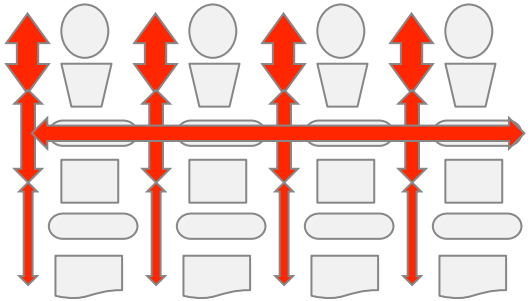
1 CPU

ILP

Instrn-Level Parallelism

A Computer Architecture History





1 CPU

ILP

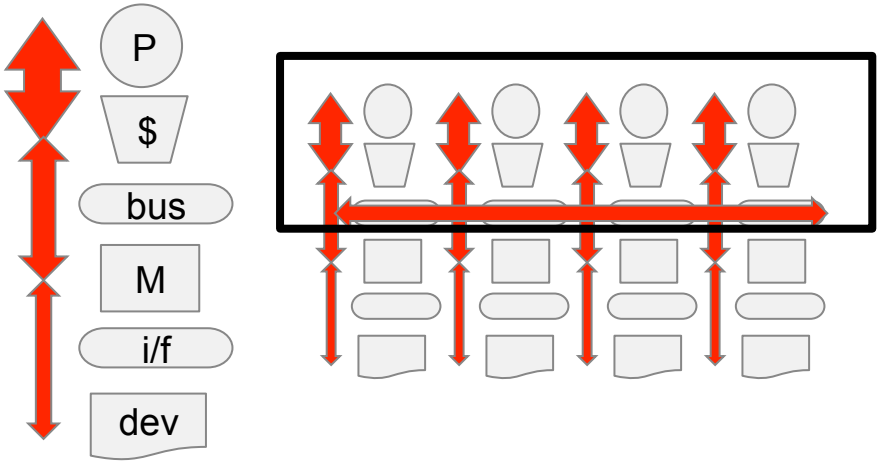
Instrn-Level Parallelism

Multiprocessor

+ TLP

Thread-Level Parallelism

A Computer Architecture History



1 CPU

ILP

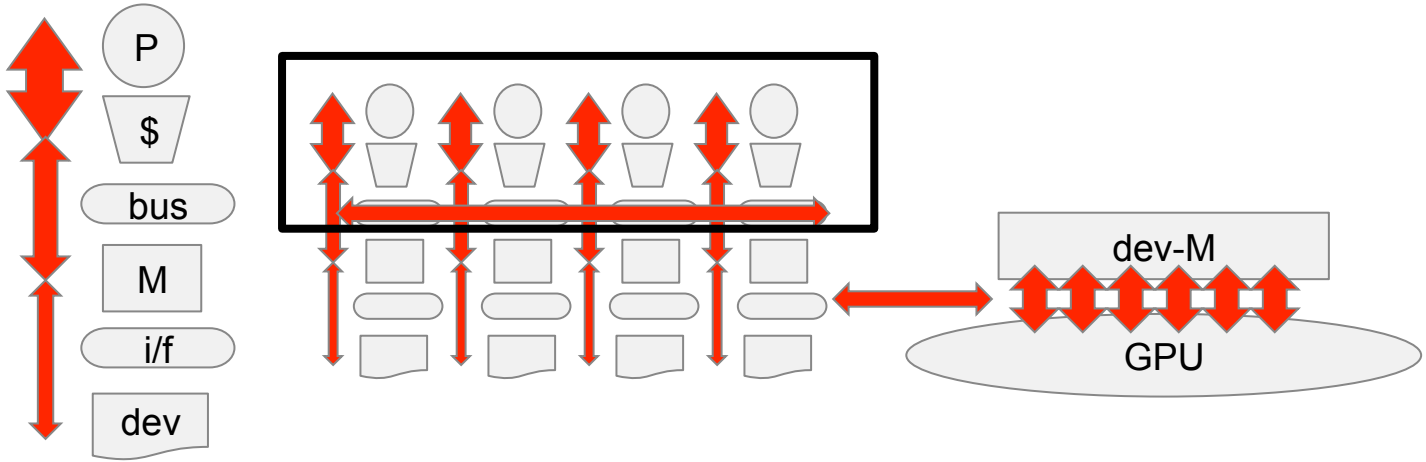
Instrn-Level Parallelism

Multicore

+ TLP

Thread-Level Parallelism

A Computer Architecture History



1 CPU

ILP

Instrn-Level Parallelism

Multicore

+ TLP

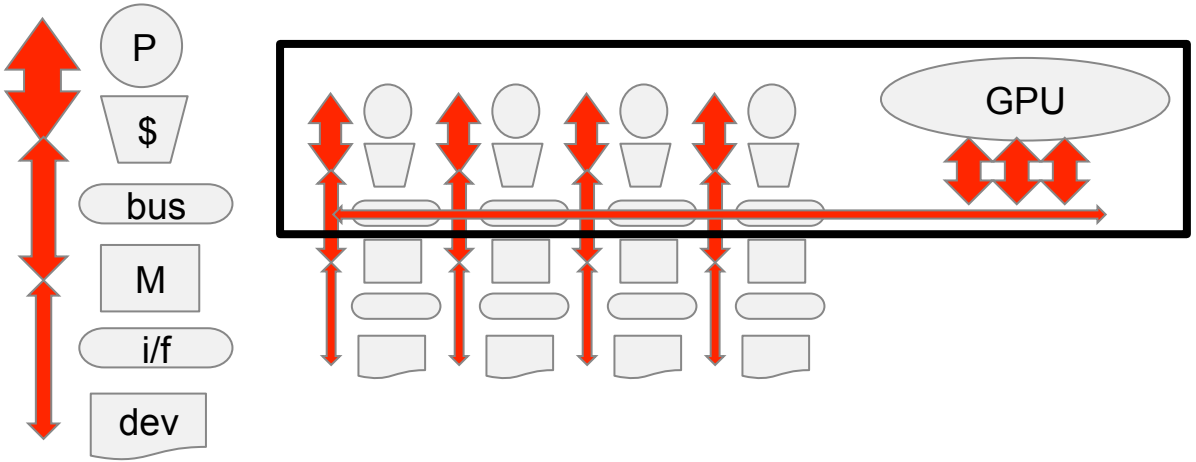
Thread-Level Parallelism

\+ **Discrete GPU**

+ DLP

Data-Level Parallelism

A Computer Architecture History



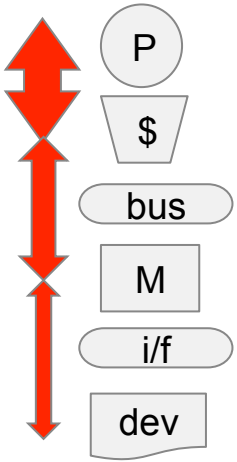
1 CPU | **Multicore** + **Integrated GPU**

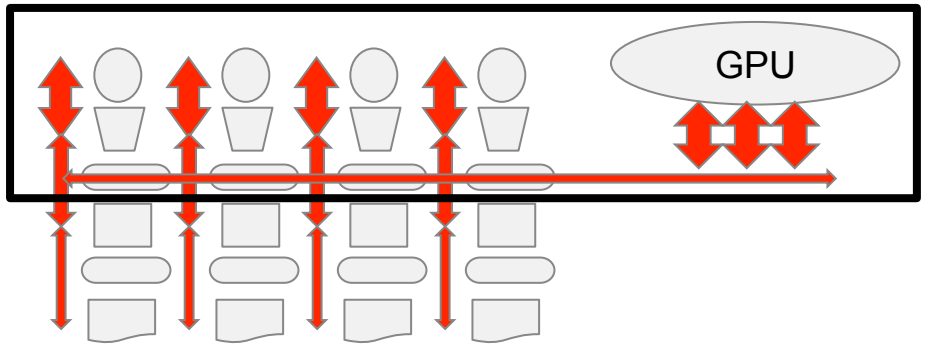
ILP
Instrn-Level Parallelism

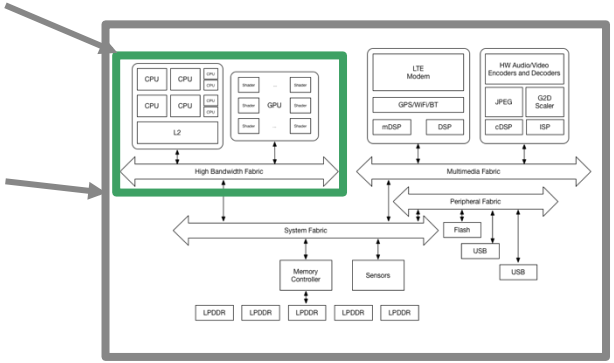
+ TLP
Thread-Level Parallelism

+ DLP
Data-Level Parallelism

A Computer Architecture History







1 CPU

ILP

Instrn-Level Parallelism

Multicore + **Integrated GPU**

+ TLP

Thread-Level Parallelism

+ DLP

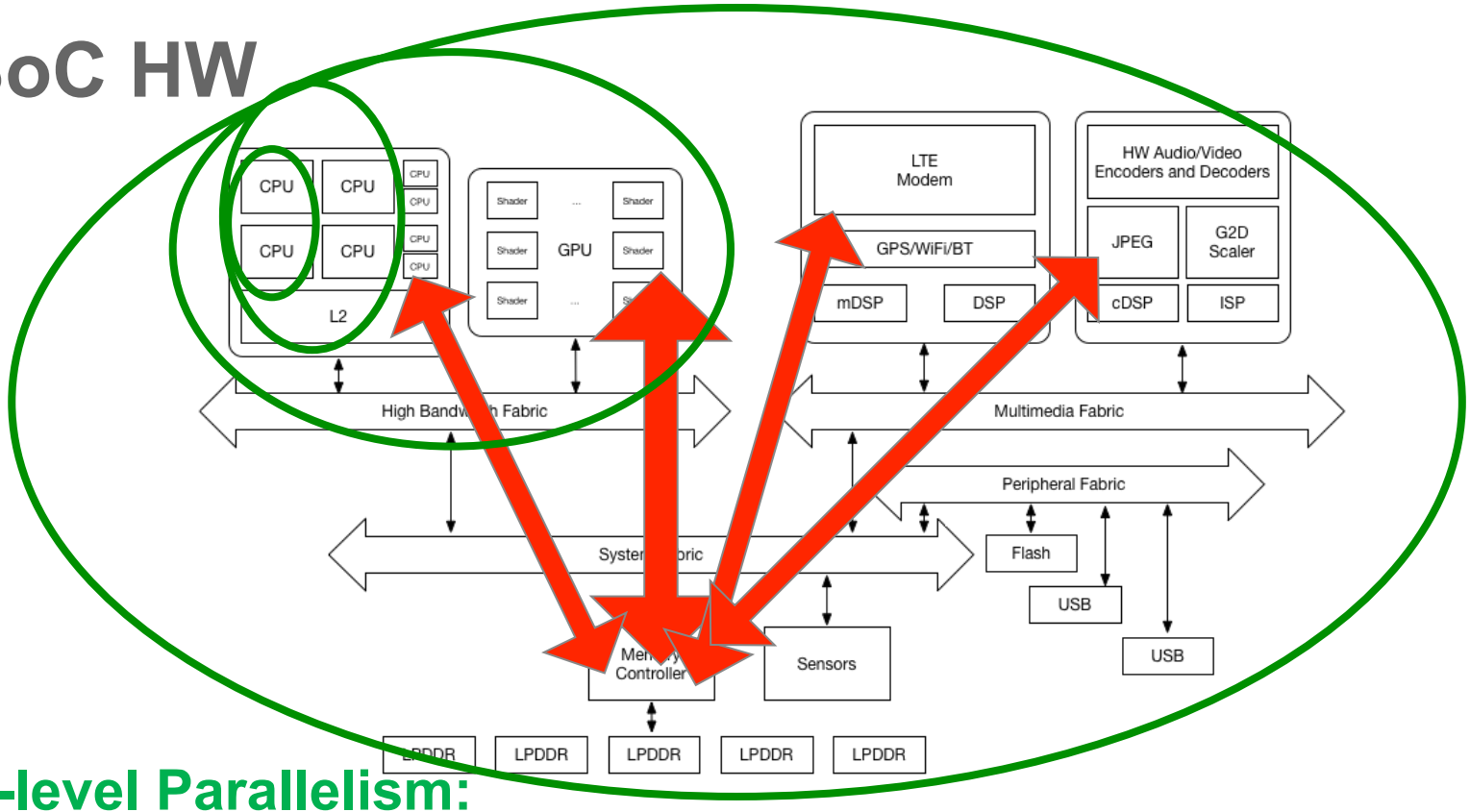
Data-Level Parallelism

System on a Chip (SoC)

+ ALP

Accelerator-Level Parallelism

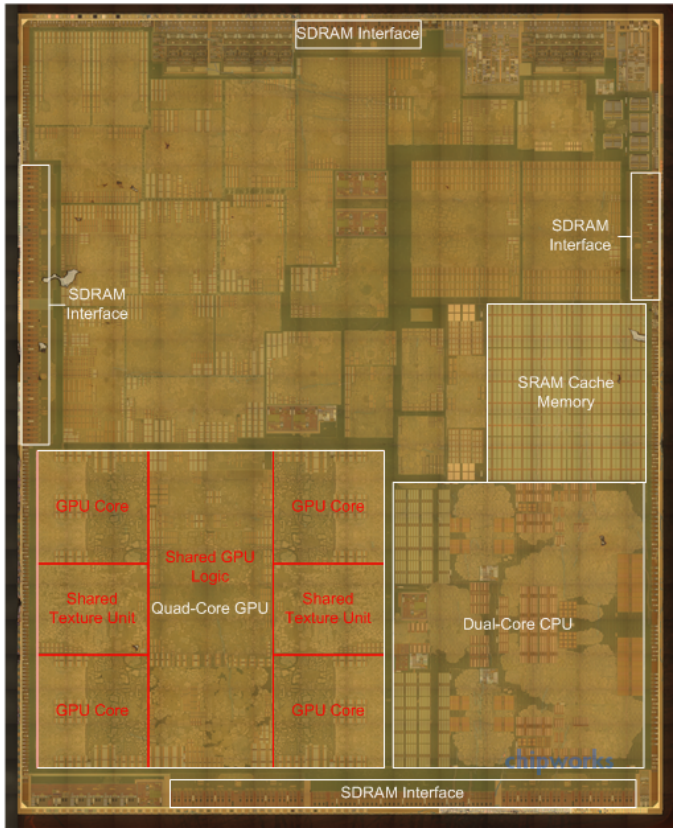
Mobile SoC HW



Accelerator-level Parallelism:

- Parallelism among workload components concurrently executing on multiple accelerators (IPs)

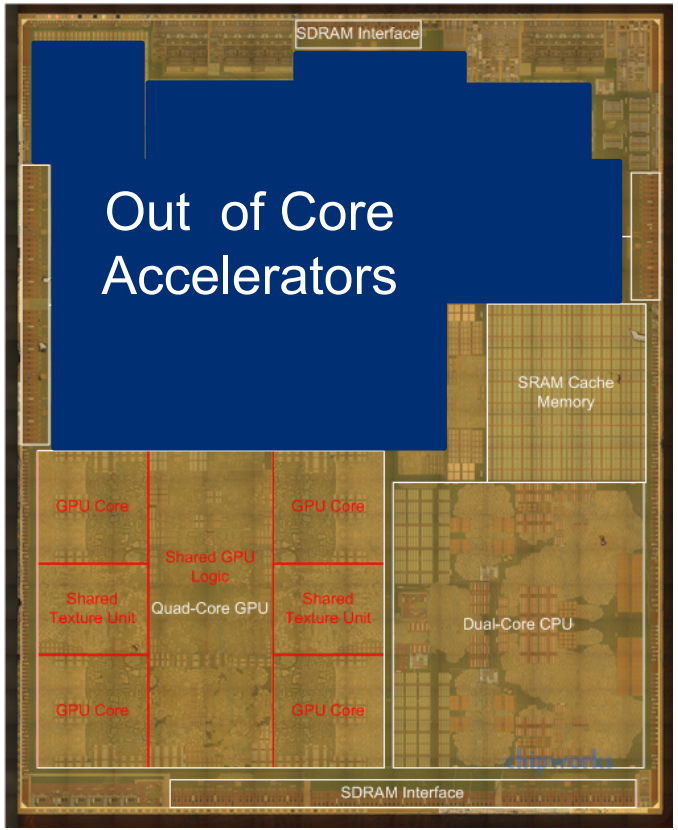
CPU, GPU, xPU (i.e., Accelerators)

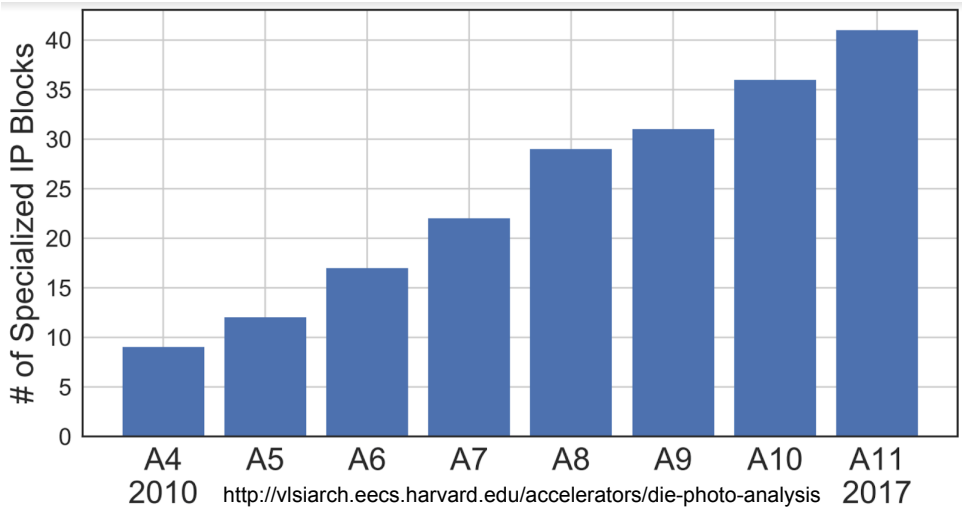


Apple A8

The CPU and GPU occupy less than 50% of the die area. What's the rest?

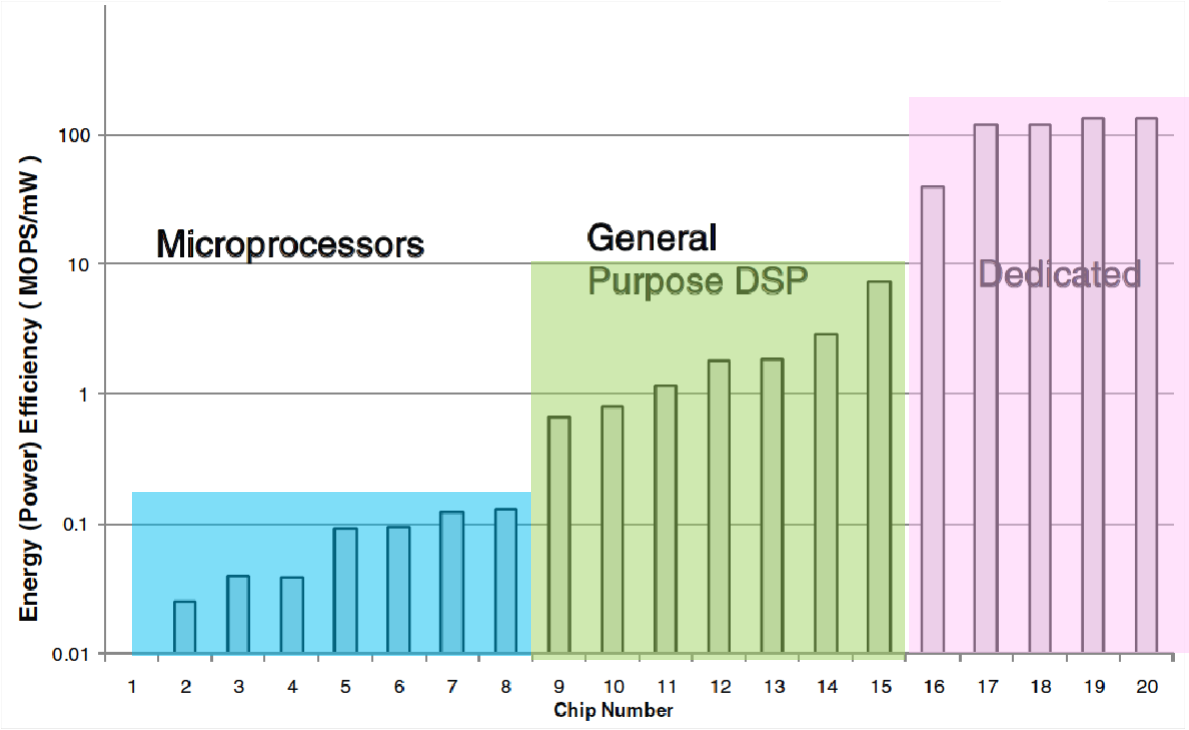
CPU, GPU, xPU (i.e., Accelerators)





The rapid rise of hardware accelerators in smartphone chips.

Potential for Specialized Accelerators



16
Encryption
17
Hearing Aid
18
FIR for disk read
19
MPEG Encoder
20
802.11 Baseband

[Brodersen and Meng, 2002]

X-level Parallelism and Software

Instruction-Level Parallelism: ILP transparent to SW; SW flourishes

Thread-Level Parallelism: TLP SW was a crisis; mixed success even today

Data-Level Parallelism: DLP SW specialized w/ point successes

- Vectorization (pioneering), CPU SIMD (intrinsics), **GPU SIMT (Cuda)**

NEW Accelerator-level Parallelism: point success for Mobile SoCs

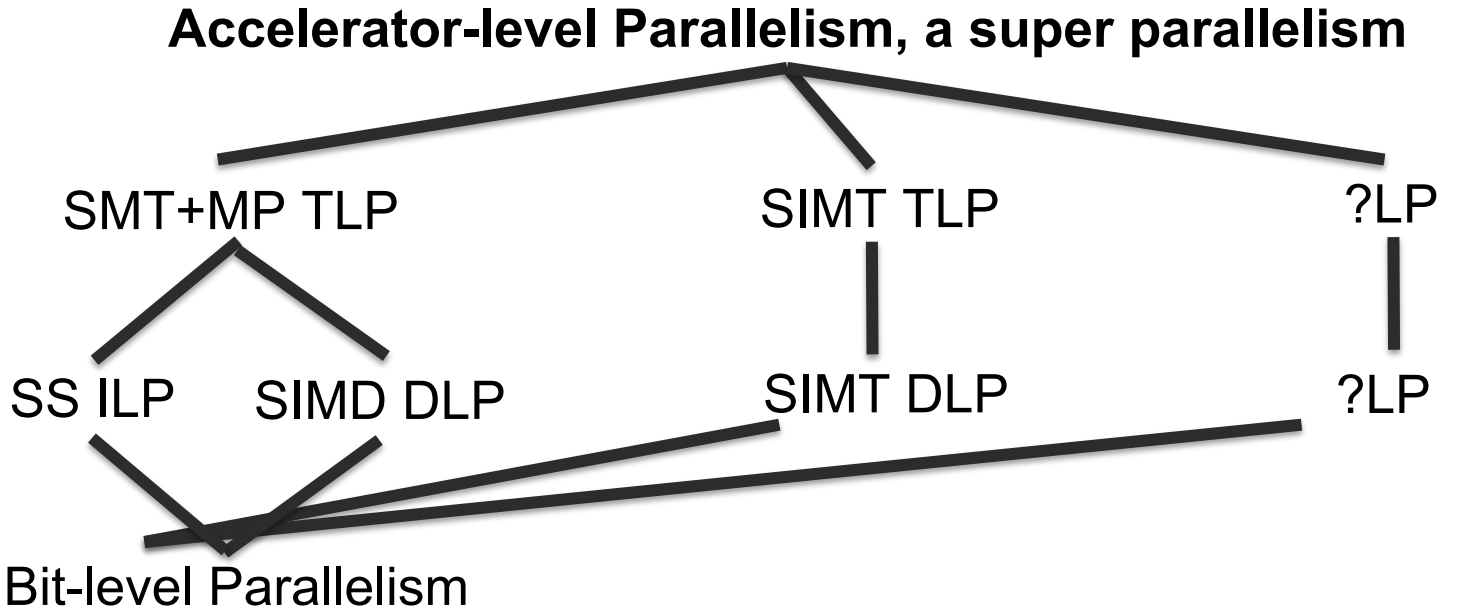
- Lacking SW/HW "science"

Hypothesis: More ubiquitous ALP will happen

- Driven by scaling perf., constrained power, & slow tech change (like SoCs)

Hypothesis: More ubiquitous ALP **desperately** needs more SW/HW "science"

A Parallelism Lattice



Acronyms
Accelerator-level Parallelism
Bit-level Parallelism
Central Processing Unit
Data-level Parallelism
Instruction-level Parallelism
Simultaneous Multithreading
Multiprocessor
Single Instruction Multiple Data
Single Instruction Multiple Threads
Superscalar
Thread-level Parallelism

CPU **GPU** **xPU**

Outline w/ Key Points

I. From ILP to Accelerator-level Parallelism

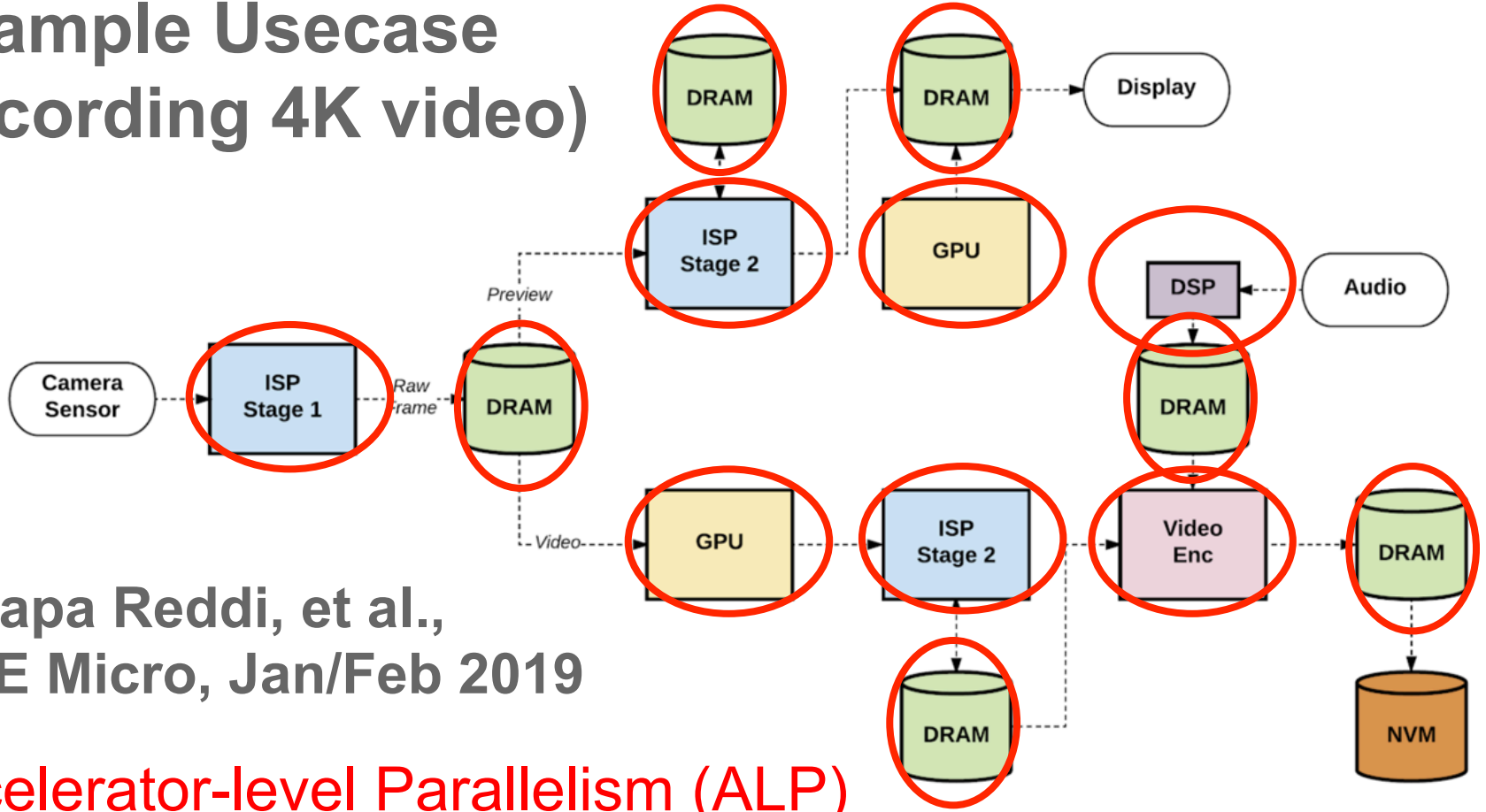
- ALP = Parallelism among workload components concurrently executing on multiple accelerators (IPs)

II. Mobile SoCs as Harbinger

- Mobile SoC already have ALP
- Some Pitfalls already emerging

III. Gables ALP SoC Model [HPCA'19 Industrial Session]
Some "first answers" to multi-IP questions

Example Usecase (recording 4K video)



Janapa Reddi, et al.,
IEEE Micro, Jan/Feb 2019

Accelerator-level Parallelism (ALP)
& repeated off-chip bandwidth use!

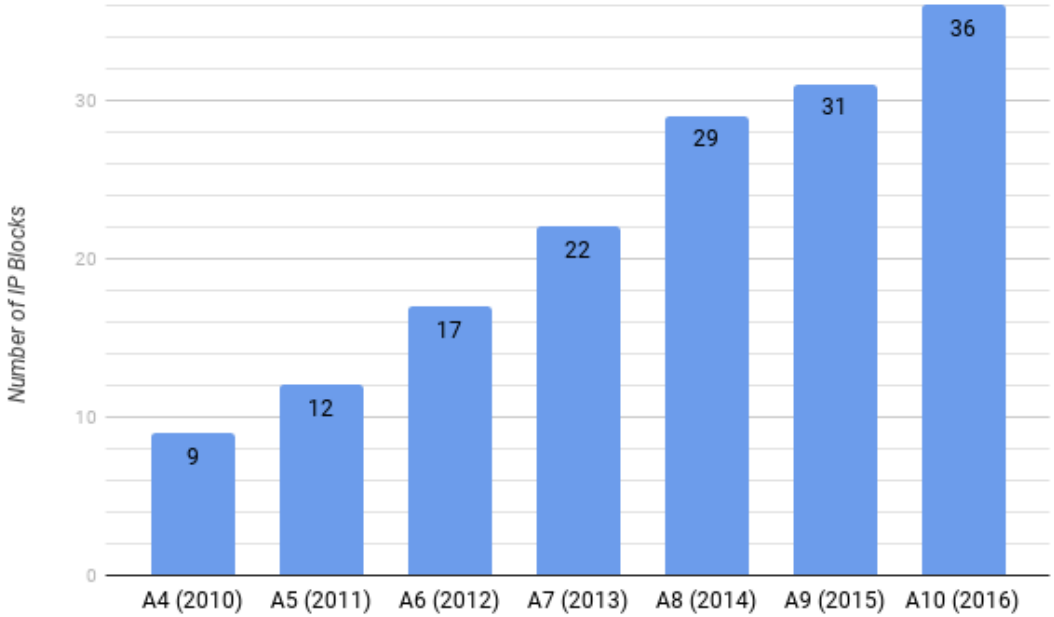
Mobile SoCs Run Usecases

	AP	Display	G2DS	GPU	ISP	JPEG	IPU	VDEC	VENC	DSP
HDR+	X	X		X	X	X	X			
Videocapture	X	X		X	X				X	
VideocaptureHDR	X	X		X	X				X	
VideoplaybackUI	X	X	X	X				X		
Google Lens	X	X	X	X						X

Must run each usecase sufficiently fast -- no need faster
Must run all usecases – average irrelevant (esp. real-time)
A usecase uses IPs concurrently: **more ALP** than serial
For each usecase, how much IP[i] acceleration needed?

Mobile SoCs Hard To Design

Envision usecases
(2-3 years ahead)
Select IPs
Size IPs
Design Uncore



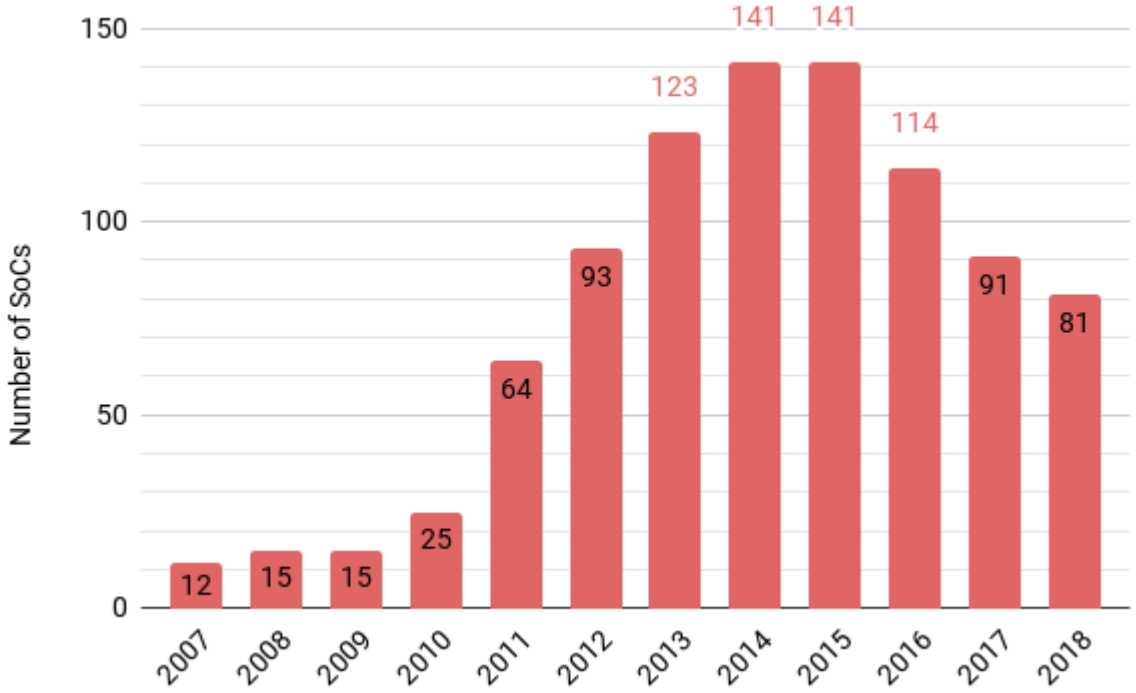
Some **Pitfalls** emerging for SoCs & Beyond

Mobile SoCs Hard To Select

Envision usecases (years ahead)
Port to many SoCs??

Port to few finalists?

Early downselect?



IP diversity hinders use [Facebook, HPCA'19]

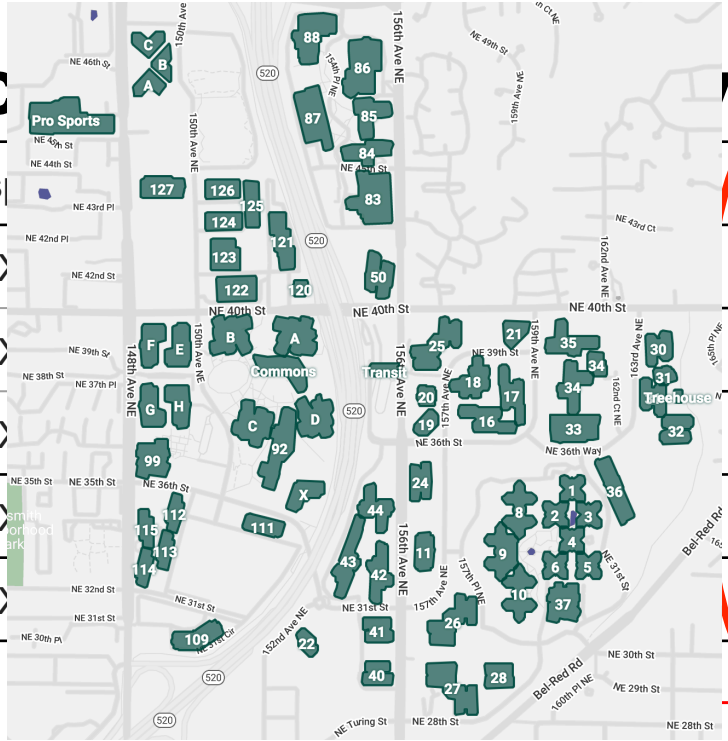
Pitfall 1: Succ ’s Law

	AP	Disp	IPU	VDEC	VENC	DSP
HDR+	X	X	X			
Videocapture	X	X			X	
VideocaptureHDR	X	X			X	
VideoplaybackUI	X	X		X		
Google Lens	X	X				X

TEAM

TEAM

TEAM



Conway’s Law [1967]: Software (& HW) ends up "shaped like" the organizational structure it's designed in

Pitfall 1: Succumbing to Conway's Law

	AP	Display		GPU	ISP	JPEG	IPU	VDEC	VENC	DSP
HDR+	X	X		X	X	X	X			
Videocapture	X	X		X	X				X	
VideocaptureHDR	X	X		X	X				X	
VideoplaybackUI	X	X		X				X		
Google Lens	X	X		X						X

Recommend:

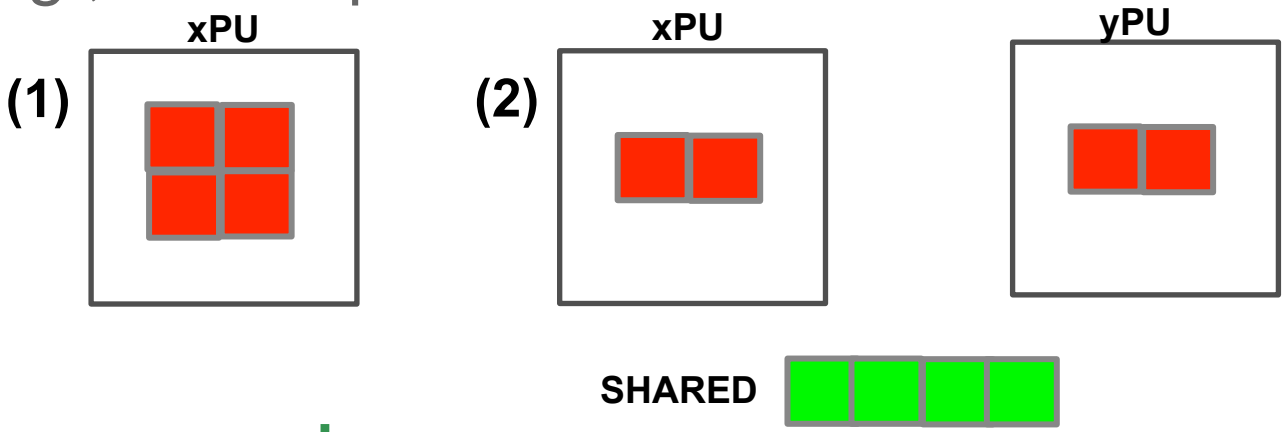
Develop org. mechanisms to combat Conway's Law

E.g., usecase teams for SoC

Pitfall 2: Optimize IPs in Isolation

Many locally opt. IPs ⇒ Globally opt. SoC??

E.g., Where put SRAM for xPU?



Recommend:

SoC: Usecase-centric design across many IPs (see **Gables**)

Future: Consider appropriate end-to-end workflows

Pitfall 3: Not Applying Amdahl's Law

Design IPs examining their peak acceleration
Consider new IP **xPU**. If 100% of usecase:

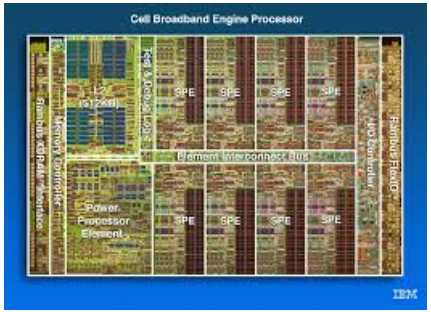
If 25% of usecase:

SoC: End-to-end work fraction at each IP; fast enough? (See **Gables**)

Future: Work fraction again; workload goals?

Pitfall 4: Hyper focus on IP HW Peak Perf.

Zero in “inner-loop” performance @ IP, ignoring:



E.g, Simpler HW ≠ Simpler HW+SW

Simple HW: IP HW has instruction memory
Hard SW: Compiler generates code & runtime “overlays” dynamically into instruction memory

HW++: IP HW instruction cache w/ 4 blocks
EZ SW: Regular compiler; opt. key routines

Pitfall 4: Hyper focus on IP HW Peak Perf.

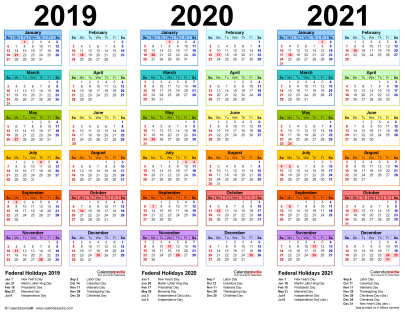
Zero in “inner-loop” performance @ IP, ignoring:

- **Simpler HW ≠ Simpler HW+SW**
- Driver startup/shutdown
- Interrupt latencies
- Time/BW to read input data & write output data
- SW stack for inter-IP communication (e.g., Android) (two device drivers rarely communicate directly)

SoC & Future: Must estimate SW overhead, even if imperfectly (0 is a bad estimate); see LogCA [ISCA’17]

Pitfall 5: Not Managing Co-Design Mismatches

HW takes years
& must work



System SW is similar
& run multiple HW

App SW multiple-month planning
w/ frequent, incremental releases

Pitfall 5: Not Managing Co-Design Mismatches

Apocryphal story:
HW Designer: What will app do in 3 **years**?
App developer: You meant 3 **months** right?

SoC: Careful planning (despite Conway's Law)
& HW flexibility as function of SW/app unpredictability
(e.g., DNNs more unpredictable than MPEG decoding)
Future: Same or TBD

Outline w/ Key Points

II. Mobile SoCs as Harbinger

- Mobile SoC already have ALP
- Some Pitfalls already emerging

Pitfall 1: Succumbing to Conway's Law
Pitfall 2: Optimize IPs in Isolation
Pitfall 3: Not Applying Amdahl's Law
Pitfall 4: Hyper focus on IP HW Peak Perf.
Pitfall 5: Not Managing Co-Design Mismatches

III. Gables ALP SoC Model [HPCA'19 Industrial Session]

- Some "first answers" to multi-IP questions

Modeling Accelerator-level Parallelism

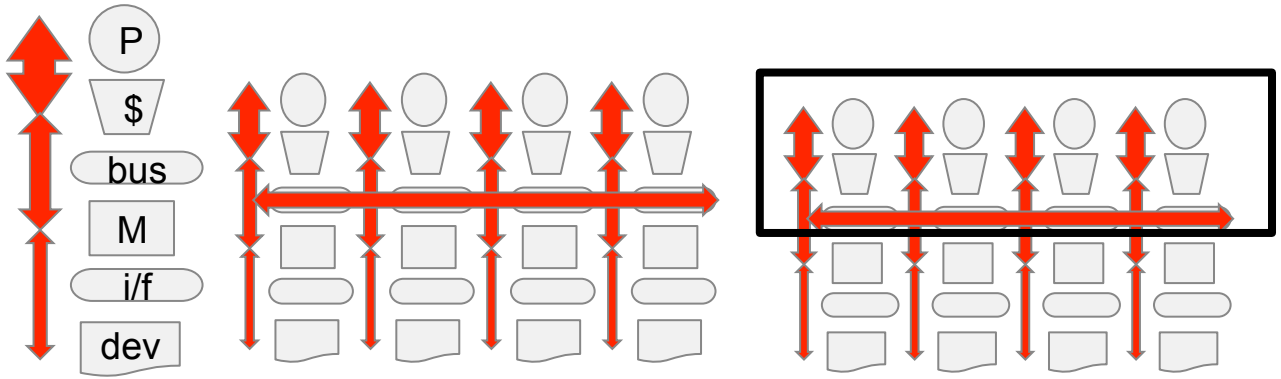
Mobile SoCs have many IPs running in parallel (ALP)

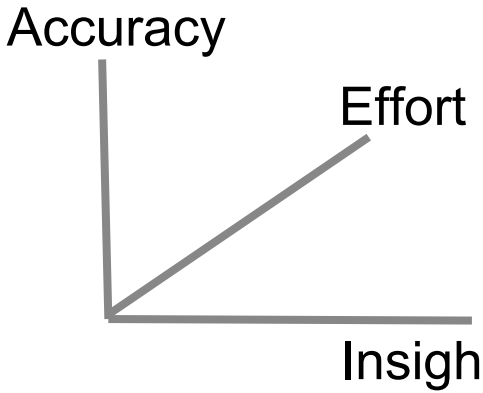
- CPUs, GPUs, DSPs, & 10+ other “IPs” (accelerators)
- Which IPs have potential? How big? How many?
- Need initial answers for IP HW/SW to create/simulate

Gables [HPCA’19 Industrial Session]

- Models give initial answers: Amdahl’s Law & Roofline
- Gables: Roofline per IP & apportion concurrent work
- E.g., balance each IP’s acceleration & communication

Computer Architecture & Models





CPU &
Iron Law

Multiprocessor &
Amdahl's Law

Multicore &
Roofline

Models vs Simulation

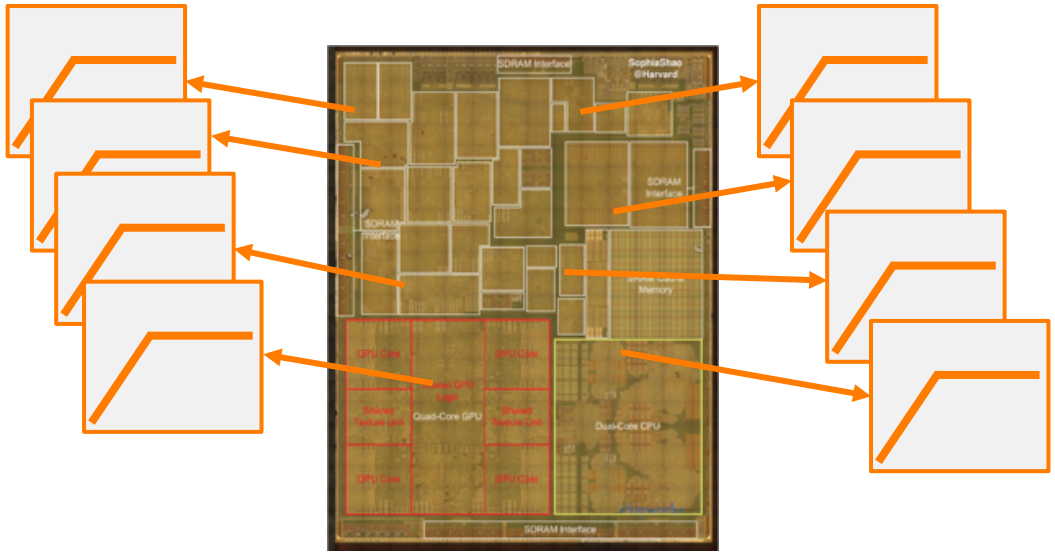
- More insight
- Less effort
- But less accuracy

Models give first answer, not final answer

Gables extends Roofline → first answer for SoC ALP

C.f., https://www.sigarch.org/three-other-models-of-computer-system-performance-part-1/
and https://www.sigarch.org/three-other-models-of-computer-system-performance-part-2/

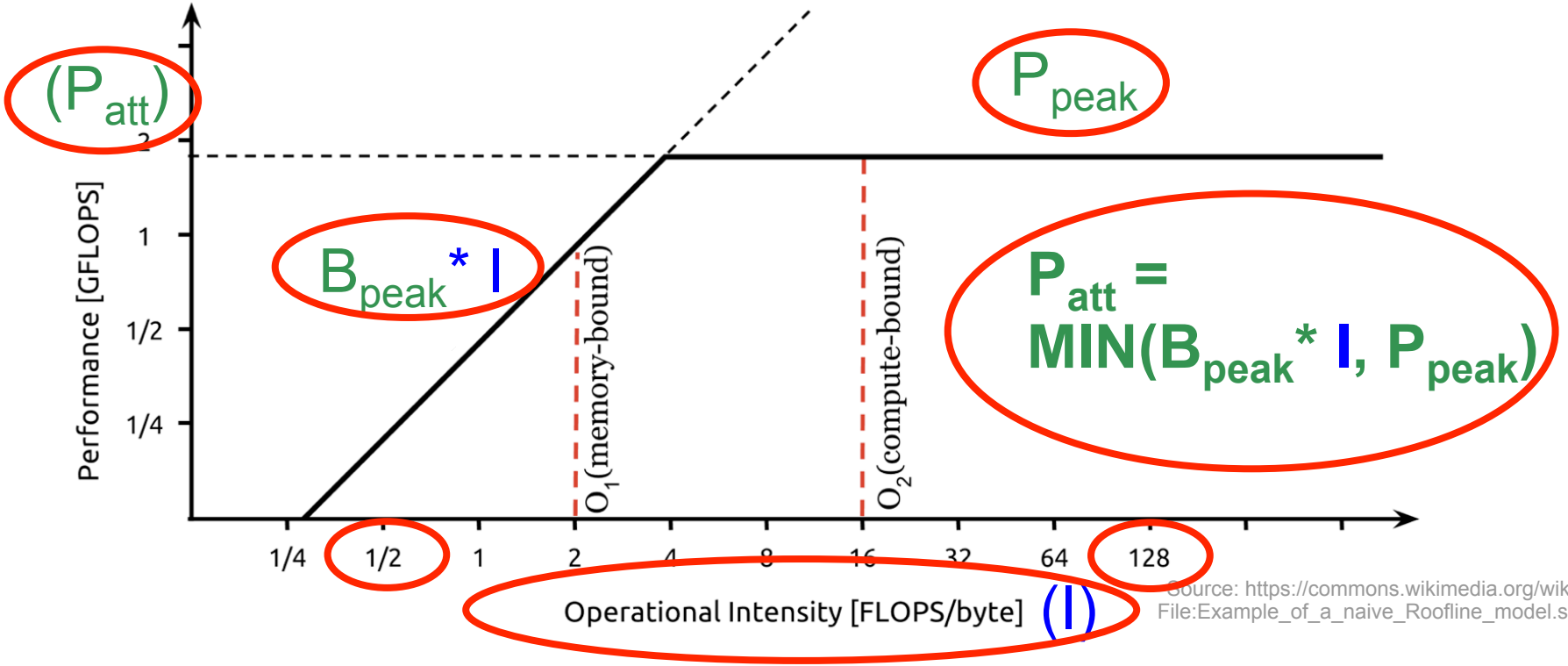
Mobile System on Chip (SoC) & Gables



Gables uses Roofline per IP to provide first answer!

What's a Roofline?

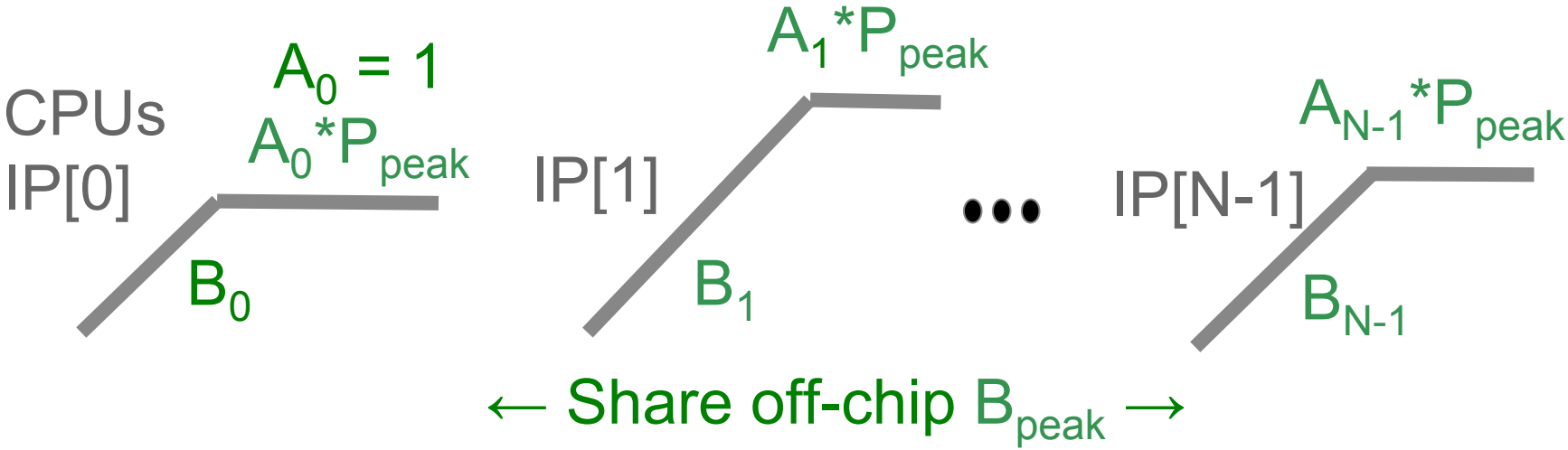
Williams et al., Roofline, CACM 4/2009



Source: https://commons.wikimedia.org/wiki/File:Example_of_a_naive_Roofline_model.svg

Compute v. Communication: Op. Intensity (I) = #operations / #off-chip bytes

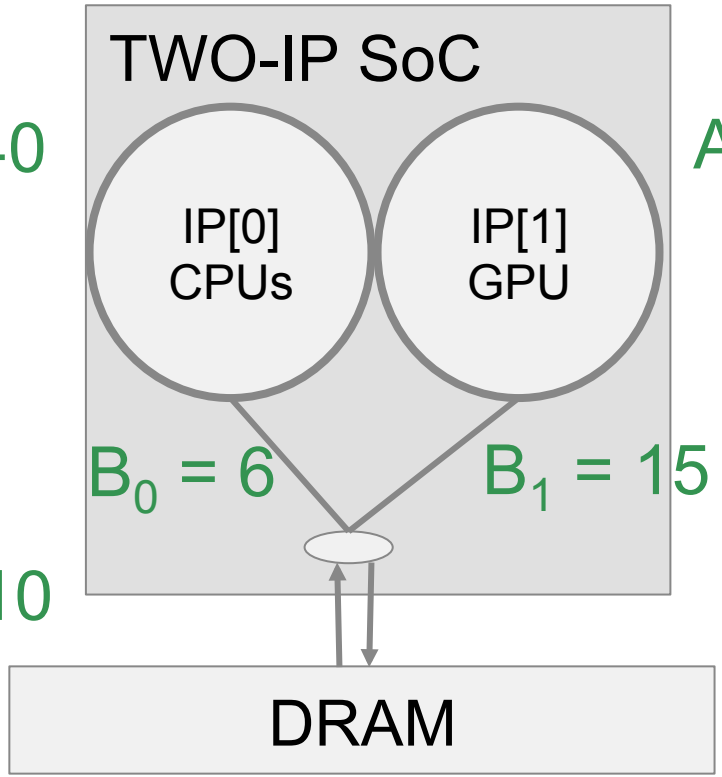
Gables for N IP SoC



Usecase at each IP[i]

- Non-negative work f_i (f_i's sum to 1) w/ IPs in parallel
- Operational intensity I_i operations/byte

Example Balanced Design Start w/ Gables



P_{peak} = 40

B_{peak} = 10

$A_1 * P_{peak}$ = 5*40 = 200

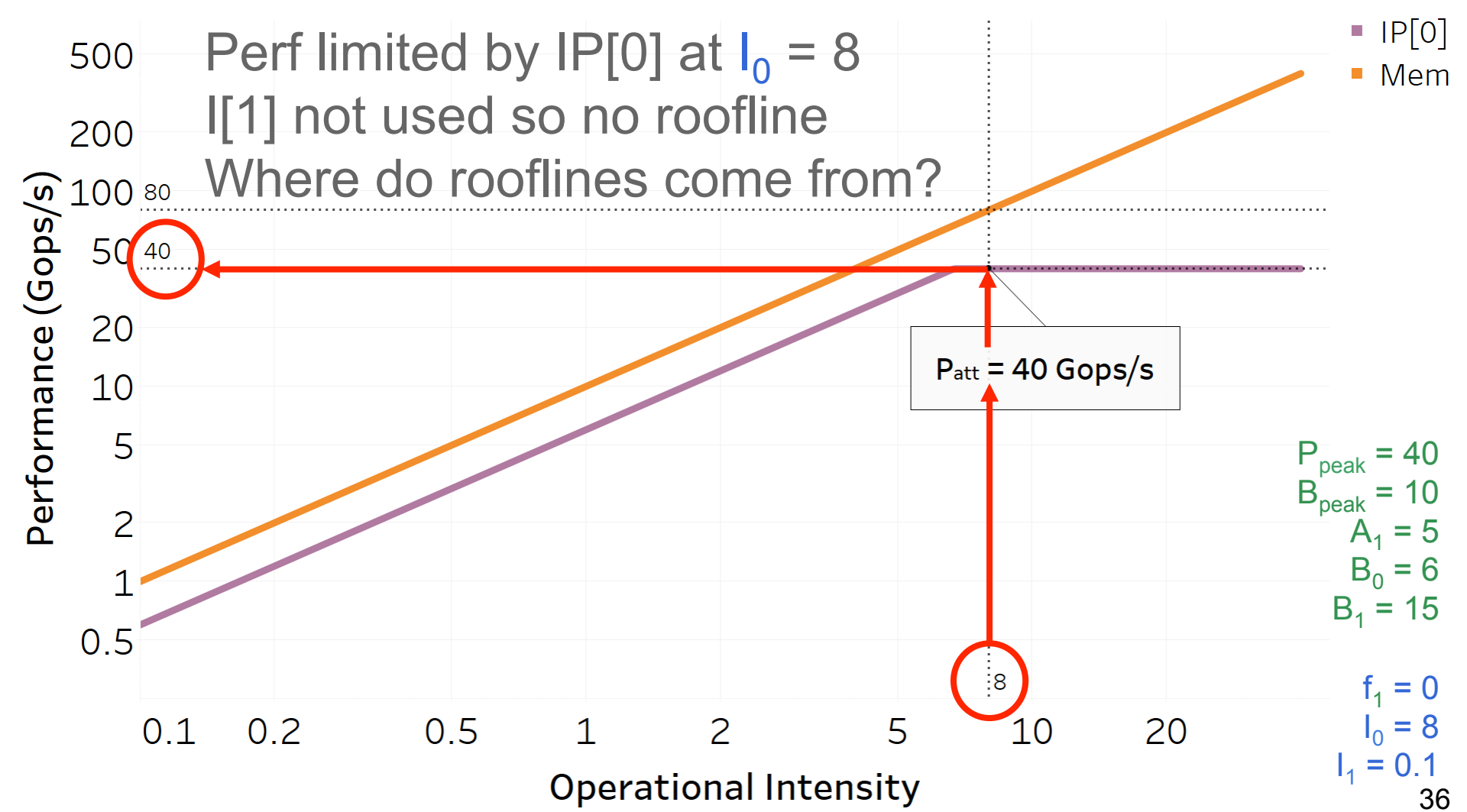
Workload (Usecase):

f_0 = 1 & f_1 = 0

I_0 = 8 = good caching

I_1 = 0.1 = latency tolerant

Performance?


Perf limited by IP[0] at I_0 = 8
I[1] not used so no roofline
Where do rooflines come from?
IP[0]
Mem
Performance (Gops/s)
500
200
100
80
50
40
20
10
5
2
1
0.5
0.1
0.2
0.5
1
2
5
10
20
8
Operational Intensity
P_{att} = 40 Gops/s
P_{peak} = 40
B_{peak} = 10
A_1 = 5
B_0 = 6
B_1 = 15
f_1 = 0
I_0 = 8
I_1 = 0.1

Gables Math: Roofline / Work Fraction

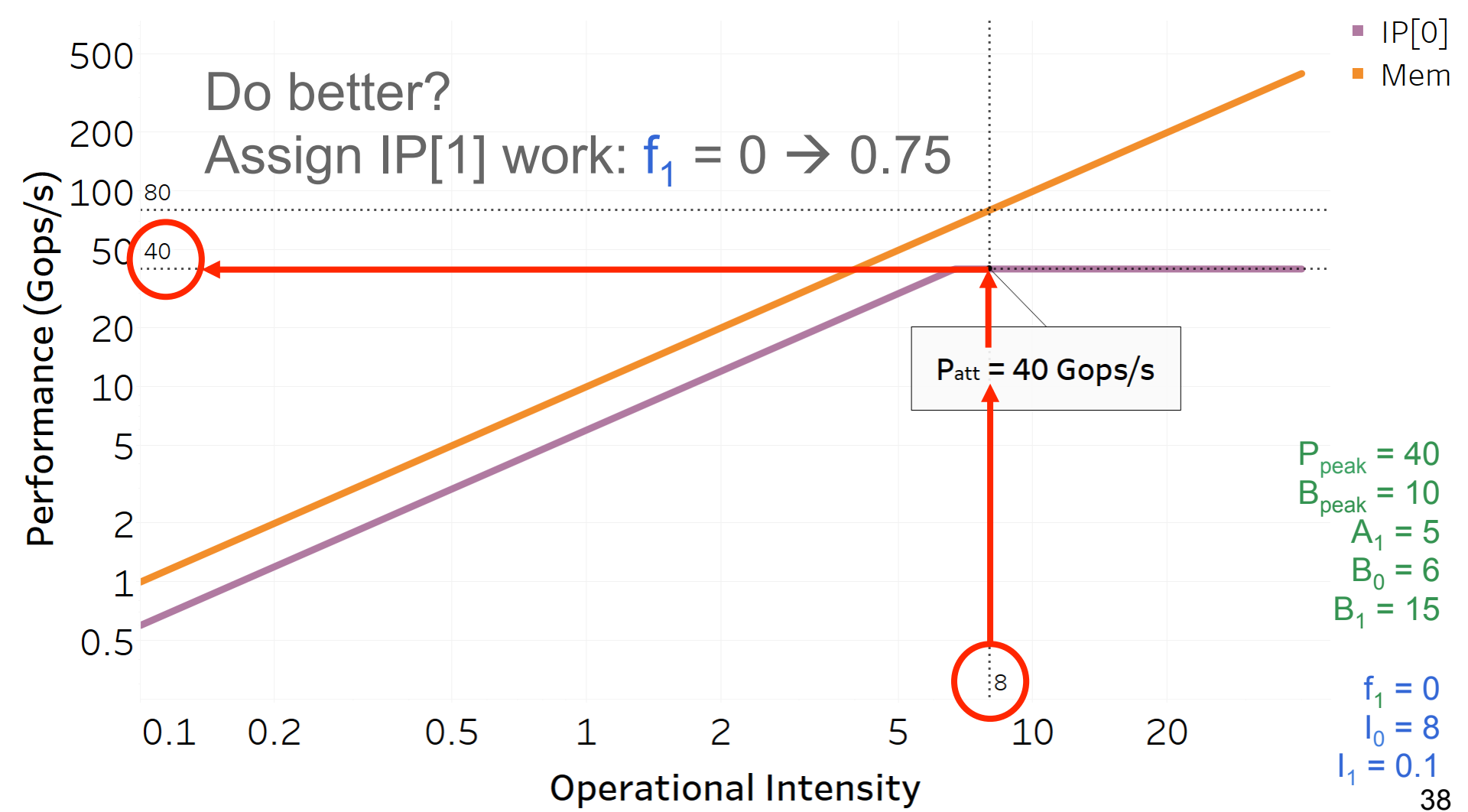
Roofline: $MIN(B_{peak} * I, P_{peak})$

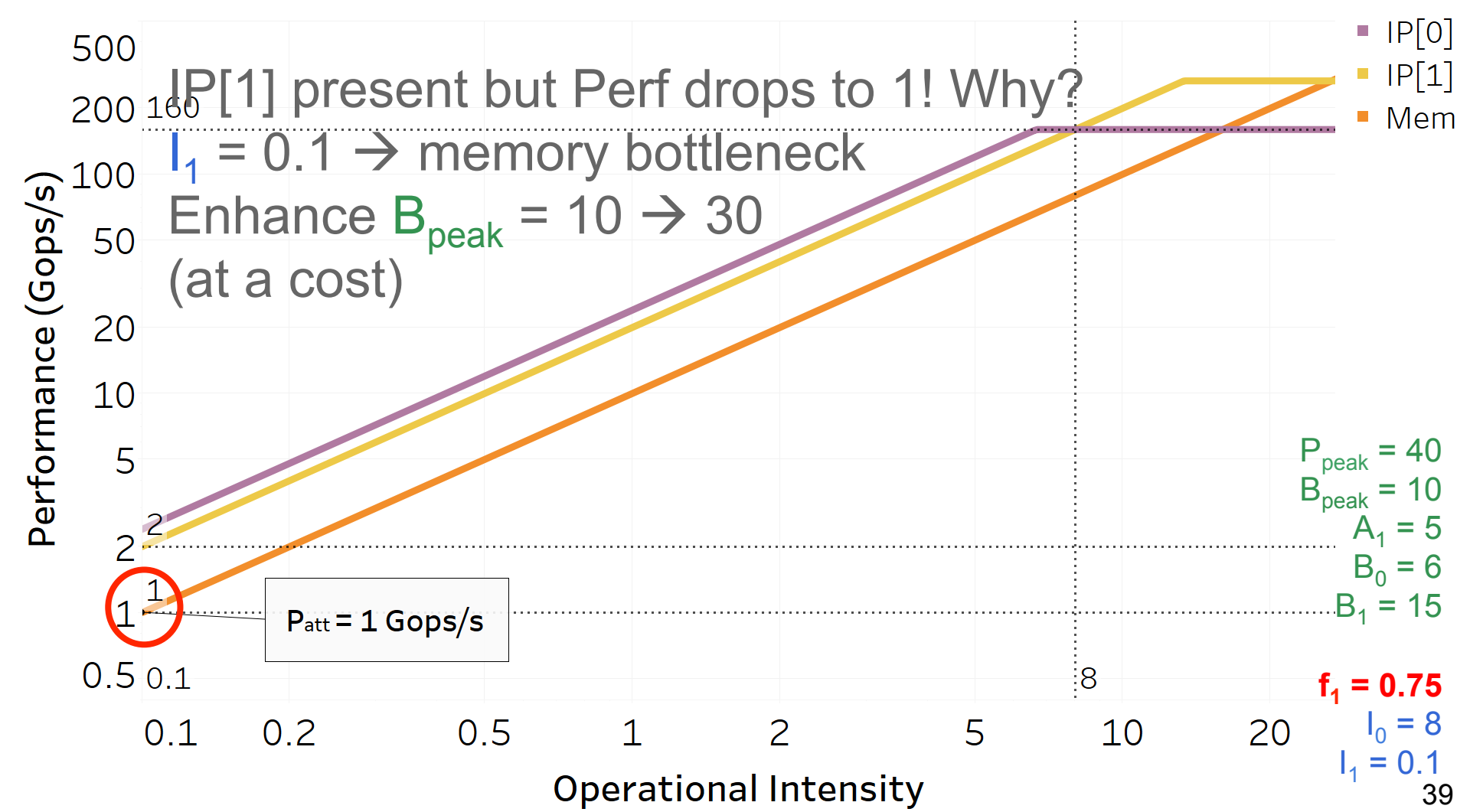
$MIN(B_{peak} * I, 1 * P_{peak}) \;/\; 1$

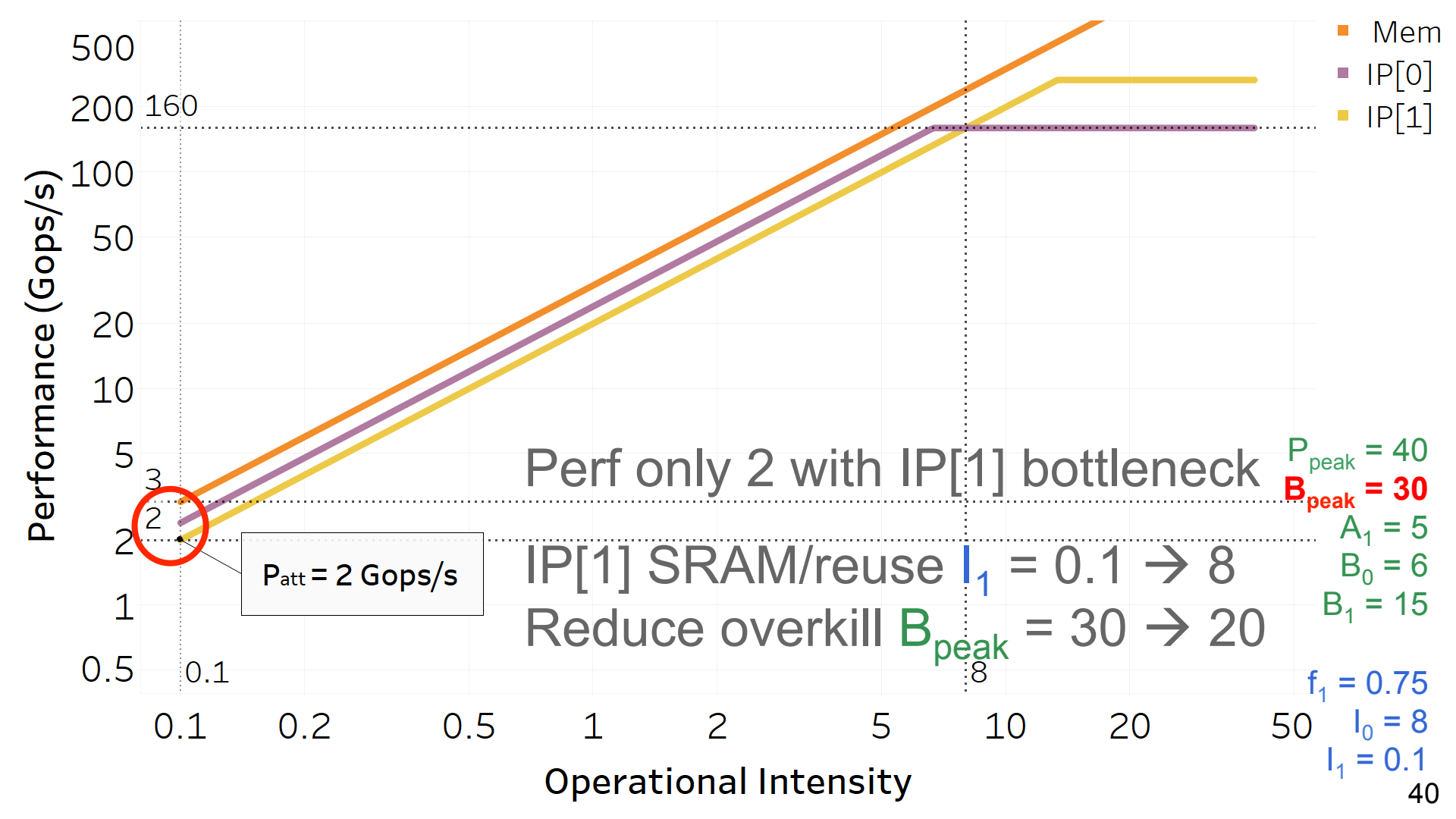
$1 / T_{IP[i]} = MIN(B_i * I_i, A_i * P_{peak}) \;/\; f_i \qquad f_i \neq 0$

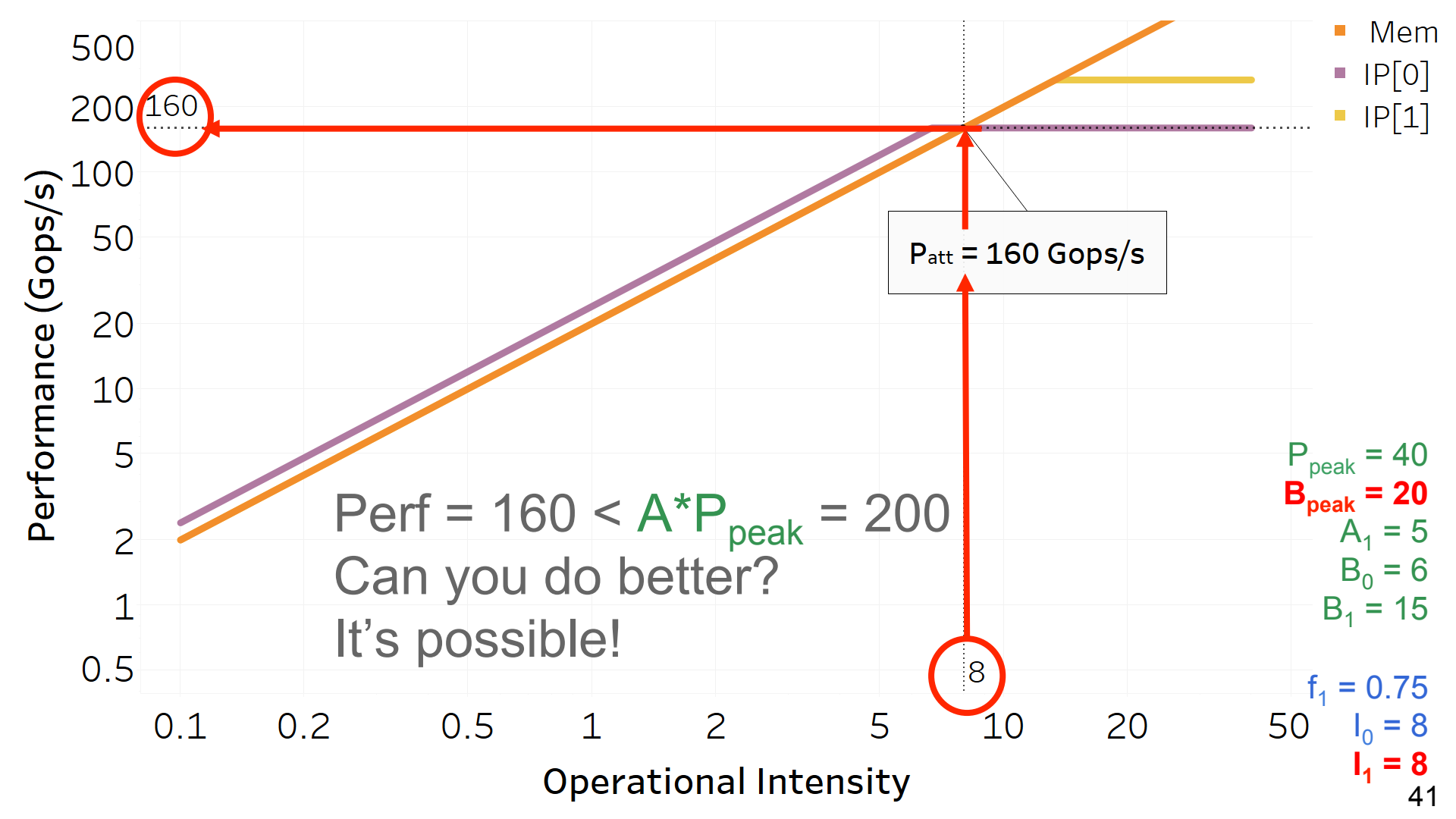
$1 / T_{memory} = B_{peak} * I_{avg} \qquad I_{avg} = 1 / \Sigma_{i=1,N-1}(f_i / I_i)$

$Perf = MIN(1/T_{IP[0]}, \ldots 1/T_{IP[N-1]}, 1/T_{memory})$


Do better?
Assign IP[1] work: f_1 = 0 → 0.75
IP[0]
Mem
Performance (Gops/s)
500
200
100
80
50
40
20
10
5
2
1
0.5
0.1
0.2
0.5
1
2
5
10
20
8
Operational Intensity
Patt = 40 Gops/s
P_{peak} = 40
B_{peak} = 10
A_1 = 5
B_0 = 6
B_1 = 15
f_1 = 0
I_0 = 8
I_1 = 0.1


IP[1] present but Perf drops to 1! Why?
I_1 = 0.1 → memory bottleneck
Enhance B_{peak} = 10 → 30
(at a cost)
IP[0]
IP[1]
Mem
Performance (Gops/s)
500
200
160
100
50
20
10
5
2
2
1
1
0.5
0.1
P_{att} = 1 Gops/s
P_{peak} = 40
B_{peak} = 10
A_1 = 5
B_0 = 6
B_1 = 15
f_1 = 0.75
I_0 = 8
I_1 = 0.1
8
0.1
0.2
0.5
1
2
5
10
20
Operational Intensity


Mem
IP[0]
IP[1]
Performance (Gops/s)
500
200
160
100
50
20
10
5
3
2
2
1
0.5
0.1
0.1
0.2
0.5
1
2
5
10
20
50
8
Operational Intensity
Perf only 2 with IP[1] bottleneck
IP[1] SRAM/reuse I_1 = 0.1 → 8
Reduce overkill B_{peak} = 30 → 20
Patt = 2 Gops/s
P_{peak} = 40
B_{peak} = 30
A_1 = 5
B_0 = 6
B_1 = 15
f_1 = 0.75
I_0 = 8
I_1 = 0.1


Mem
IP[0]
IP[1]
500
200
160
100
50
20
10
5
2
1
0.5
Performance (Gops/s)
0.1
0.2
0.5
1
2
5
10
20
50
Operational Intensity
Patt = 160 Gops/s
Perf = 160 < A*Ppeak = 200
Can you do better?
It's possible!
8
Ppeak = 40
Bpeak = 20
A1 = 5
B0 = 6
B1 = 15
f1 = 0.75
I0 = 8
I1 = 8

A Gables Workflow for a 1st SoC Answer

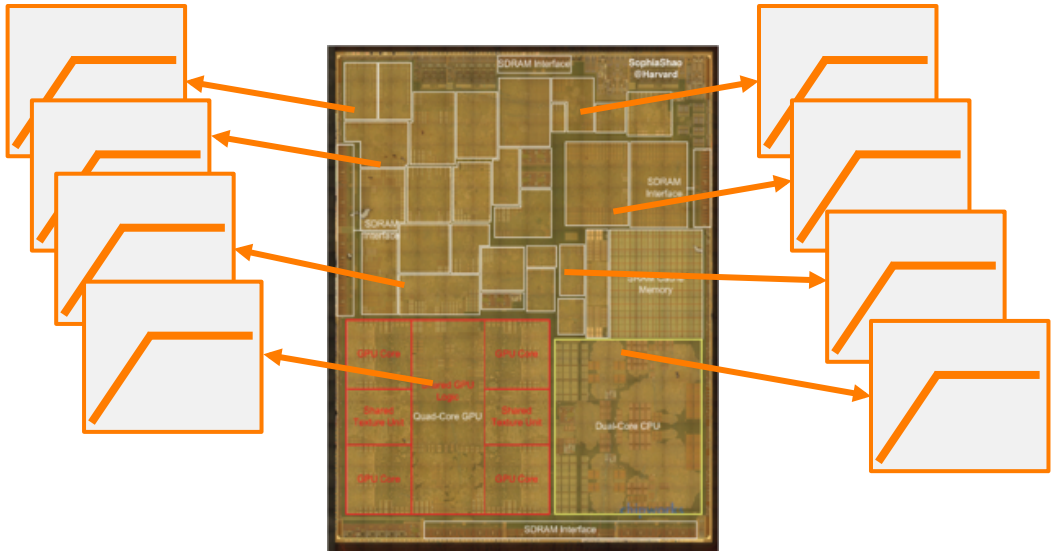
	AP	Display	G2DS	GPU	ISP	JPEG	IPU	VDEC	VENC	DSP
HDR+	X	X		X	X	X	X			
Videocapture	X	X		X	X				X	
VideocaptureHDR	X	X		X	X				X	
VideoplaybackUI	X	X	X	X				X		
Google Lens	X	X	X	X						X

For each usecase repeat until sufficiently fast

- Pick bottleneck IP[i] improve compute/communication
  Pick non-bottleneck IP[i] reduce cost

Pick IP[i] configs that satisfy all usecases; done if cost ok

Mobile System on Chip (SoC) & Gables

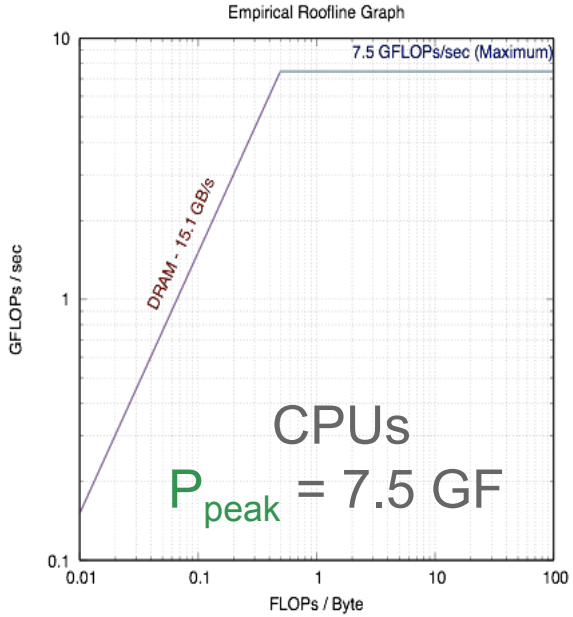


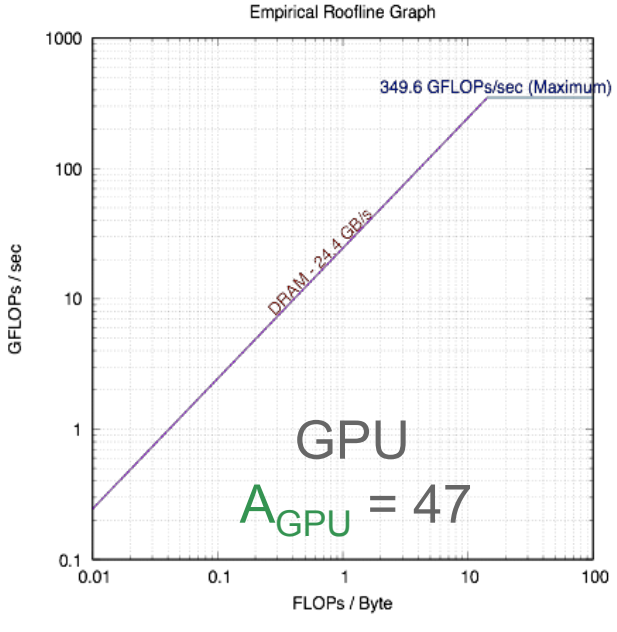
1. Include Accelerator IP[i]? Or give work to enhanced CPUs
2. IP[i] over-provisioned? Make IP[i] acceleration less
3. IP[i] over-communicates? IP[i] less compute; more SRAM

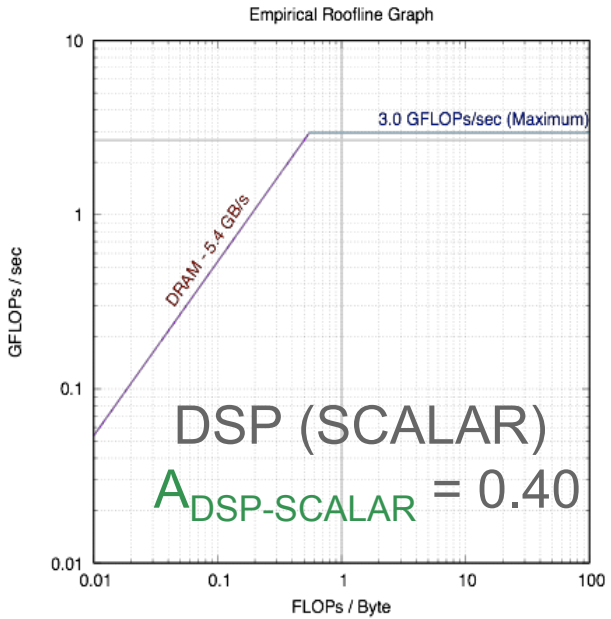
Pixel 2 (Snapdragon 835) w/ Aux. Thermal Mangmt

µBenchmark w/ Qualcomm Snapdragon™ 835

- All elements load from array & vary FP SP op intensity
- Finds empirical lower bound on rooflines

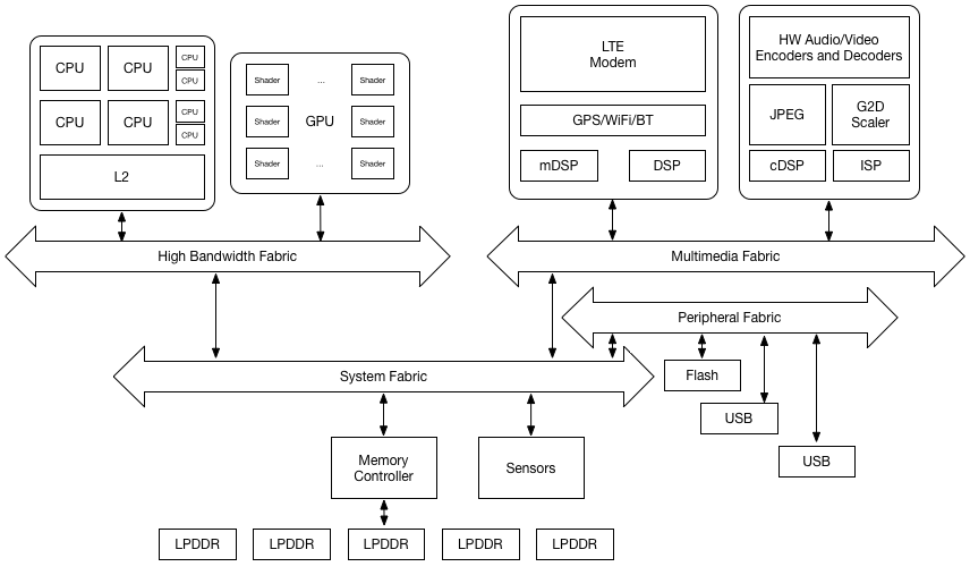


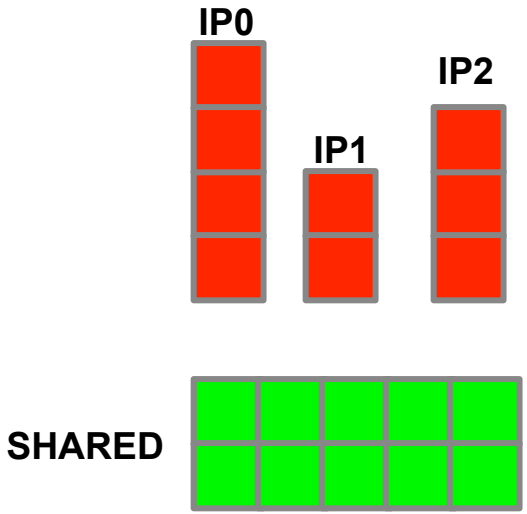




- Preliminary evidence that multiple rooflines useful

Case Study: Allocating SRAM

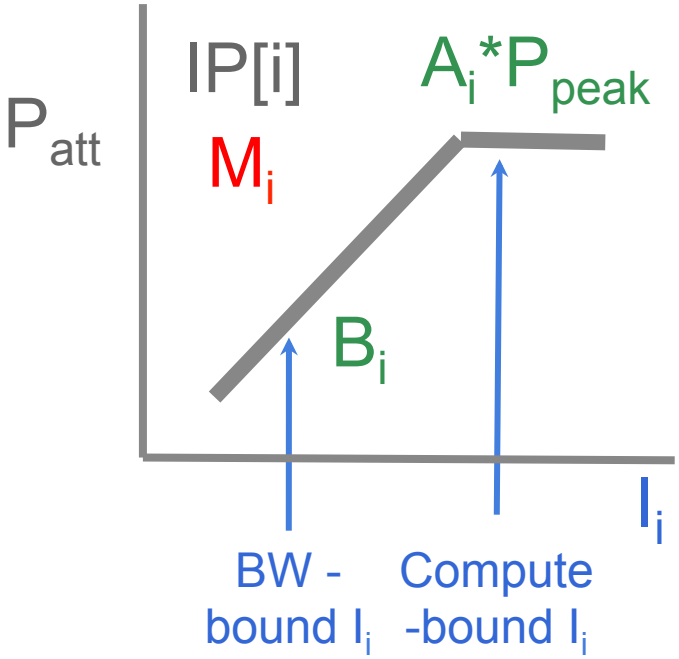




Where SRAM?

- Private w/i each IP
- Shared resource

What determines I_i?



SW Usecase (most important)

- Dense v. sparse matrices
- E.g. vision v. audio ML

Hardware

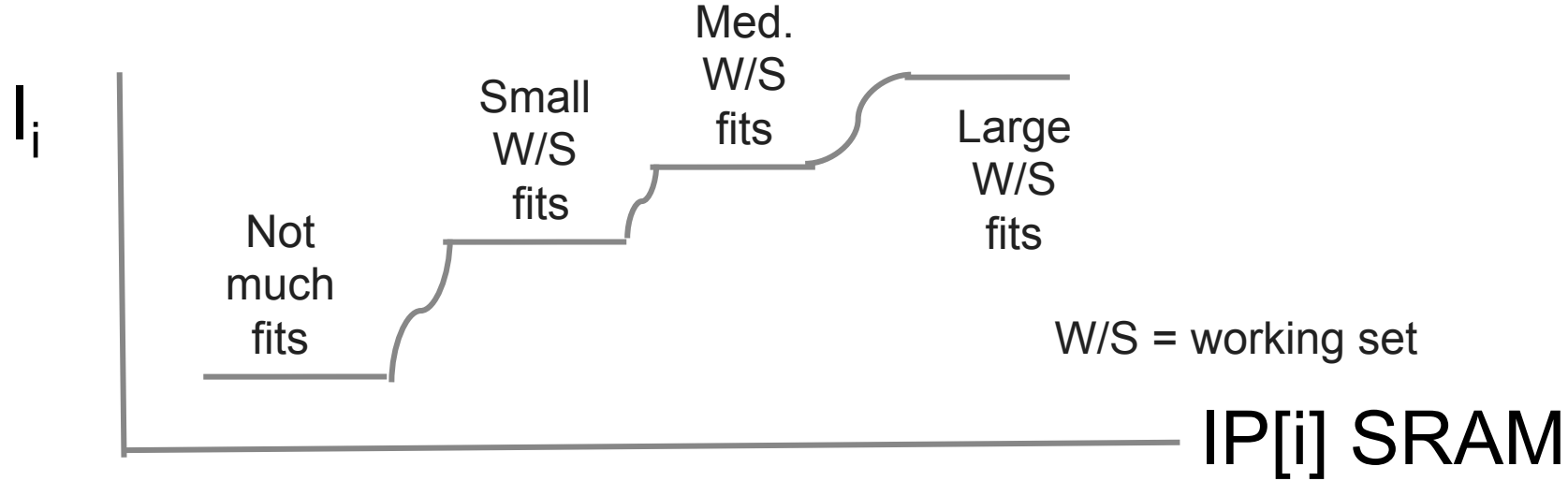
More A_i toward BW-bound (recall f_i too!)

More B_i toward compute-bound

More M_i toward compute-bound **if reuse**

Whither I_i as function of M_i?

Does more IP[i] SRAM help Op. Intensity (I_i)?



Non-linear function that increases when new footprint/working-set fits

Should consider these plots when sizing IP[i] SRAM

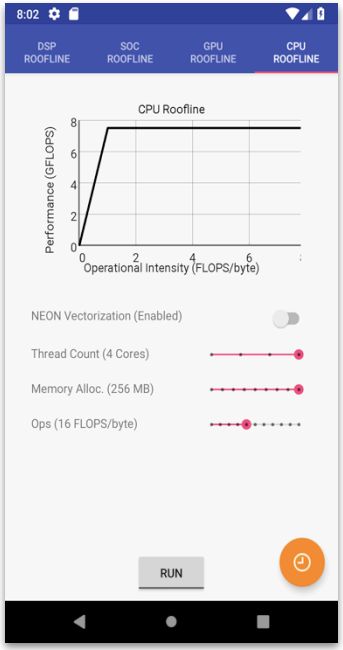
Later evaluation can use simulation performance on y-axis

Gables Paper & Home Page

Extensions: memory-side buffer, interconnect, serial work

Interactive tool for 2-IP & 3-IP SoCs

Gables Android Source at GitHub



http://research.cs.wisc.edu/multifacet/gables/

Gables Executive Summary

All models are wrong, but some are useful.
–George Box, Statistician, 1987

Gables Mobile SoC Model [HPCA'19 Industrial Session]

- Models give initial answers: Amdahl's Law & Roofline
- Gables: Roofline per IP & apportion concurrent work
- E.g., how much IP[i] acceleration needed?

Accelerator-level Parallelism

ALP = Parallelism among workload components concurrently executing on multiple accelerators (IPs)

Mobile SoCs: point successes, lacking SW/HW science

Hypothesis: More ubiquitous ALP will happen

- Due to scaling perf., constrained power, & slow tech change
- Retarded by SW/HW "science" of ALP among CPUs (ILP+TLP), GPUs (+DLP), & many IPs (xLP)

Hennessy & Patterson: A New Golden Age for Computer Architecture

Accelerator-level Parallelism: Research Call

Parallelism Success → Deep Thinking

- ILP: basic blocks too short → branch prediction
- TLP: SW to manage (OpenMP) or hide (SQL)
- DLP: SIMT surpasses SIMD/vectors on <$1K GPUs

Infrastructure

SimpleScalar

gem5

GPGPU-Sim

Let's do Deep Thinking for ALP

- Enhance or coalesce IPs (in progress)
- Create SW/HW for coordination & communication
- SW abstraction/implementation for each IP hard
- SW abstractions/implementations for ALP harder
- **All needed for continued computer performance scaling**

Aladdin++?

Outline w/ Key Points

I. From ILP to Accelerator-level Parallelism

- ALP = Parallelism among workload components concurrently executing on multiple accelerators (IPs)

II. Mobile SoCs as Harbinger

- Mobile SoCs already have ALP
- Some Pitfalls already emerging

III. Gables ALP SoC Model [HPCA'19 Industrial Session]

- Some "first answers" to multi-IP questions

Thanks to Mobile Silicon Team @ Google

Backup Slides

Related Work

Builds on Roofline & Amdahl's Law

Closest: SoC MultiAmdahl [Kelassy et al., CAL'12]
Gables adds BW per-IP & chip & uses concurrent work

Gables can be extended

- CPU-GPU "Valley" [Guz et al., CAL'09]
- LogCA interaction overheads [Altaf & Wood, ISCA'17]
- Richer IP models, e.g., [Jog et al., ISMS'15]

Gables Caveats

Base Assumptions

- SW has perfect Accelerator-level Parallelism
- All IP's concurrent w/ each other & memory BW
- BW limits of Roofline appropriate (proxy for power?)

But

- Insight but not cycle-level accuracy
- Omits interrupt latencies, etc., to manage IPs
- IP acceleration varying w/ usecase (Roofline ceiling?)
- <your concern here>

Gables Conjectures

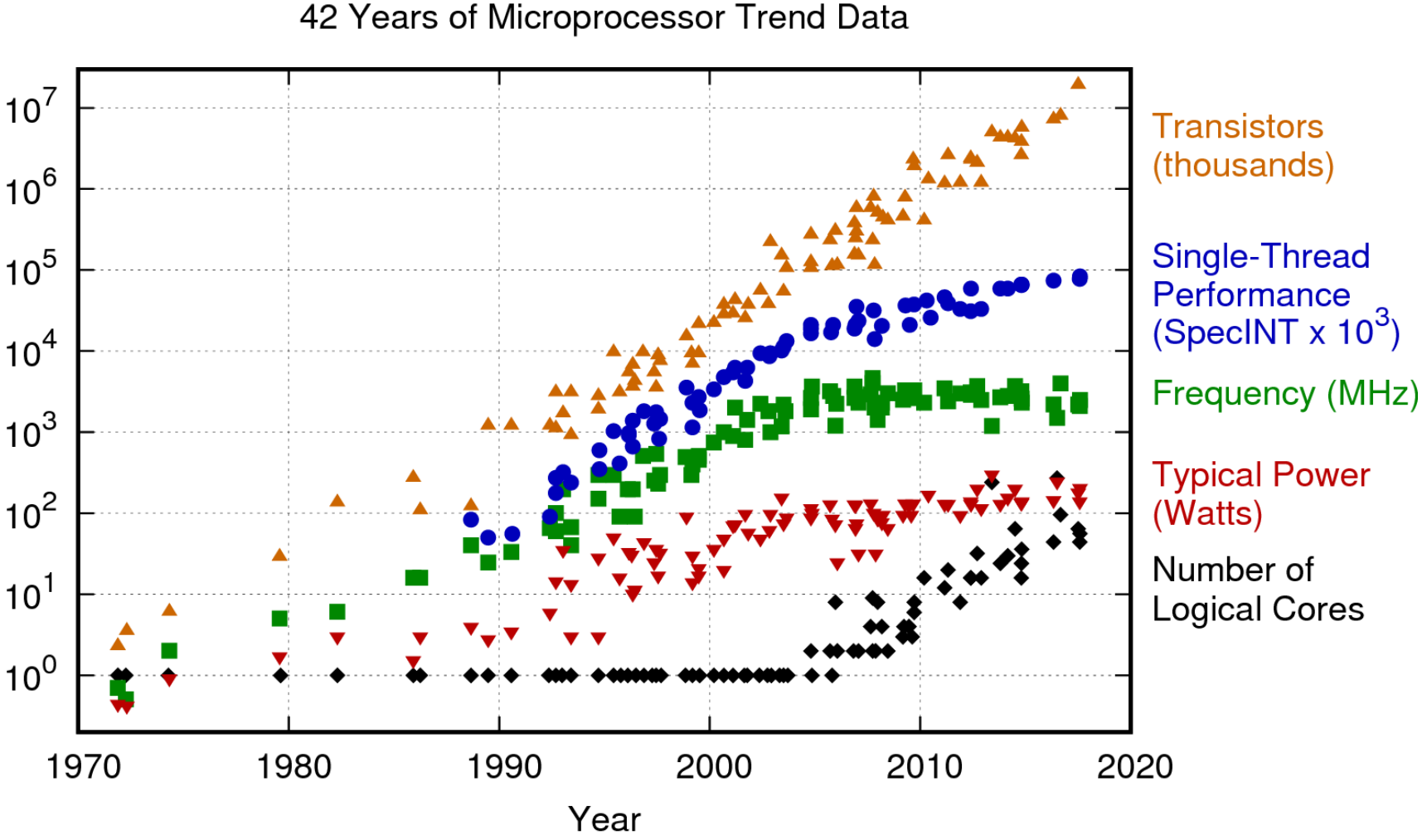
Gables provides a way to conceptualize many-IP SoCs

- Roofline per IP forces early parameter estimation
- Insight for much less work than porting usecases

Operational intensity I_i zeros in on SRAM utility & reuse

Understanding work fraction f_i valuable to estimate the acceleration A_i necessary for each usecase

SoCs harbinger of accel.-level parallelism broadly

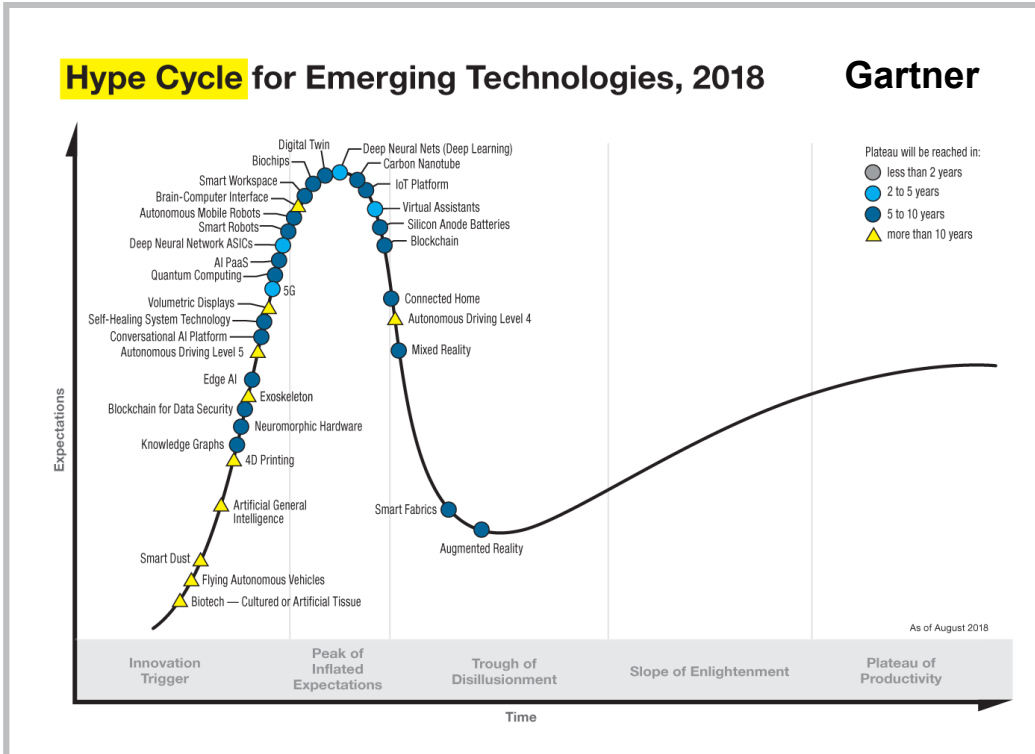


Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp https://www.karlrupp.net/wp-content/uploads/2018/02/42-years-processor-trend.png

Pitfall X: Design for (Hyped) Importance

IPs should target important workloads, but …

Recommend: Provision IP resources (compute & SRAM) only as needed for important usecases



LogCA Perf. Model of HW Accelerator


Inspired by LogP [CACM1996]

Abstract accelerator using five parameters

- **L** Latency: Cycles to move data
- **o** Overhead: Setup cost
- **g** Granularity: Size of the off-loaded data
- **C** Computational index: Work done per data byte
- **A** Acceleration: Speedup ignoring overheads

Gables Glossary

SoC HW Inputs

- P_{peak} & B_{peak} CPU perf. & off-chip BW from Roofline
- A_i & B_i acceleration & BW for each IP[i]

SW Usecase Inputs

- f_i fraction work at each IP[i]
- I_i operational intensity at each IP[i]

Output

- $P_{attainable}$ SoC performance upper bound

Numbers Behind Gables's Example

6**A** ----------
Ppeak = 40 Gops/s, Bpeak = 10 Gbytes/s, A = 5, B0 = 6 and B1 = 15.
I0 = 8 operations/byte on IP[0], I1 =0.1 for IP[1], and f =0.00.

$1 / T_{IP[0]} = MIN(B_0 * I_0, P_{peak}) / (1 - f)$ $f \neq 1$
$1 / T_{IP[1]} = MIN(B_1 * I_1, A * P_{peak}) / f$ $f \neq 0$
$1 / T_{memory} = B_{peak} * I_{avg}$ $I_{avg} = 1/[(1- f)/ I_0) + (f / I_1)]$
Perf = $MIN(1/T_{IP[0]}, 1/T_{IP[1]}, 1/T_{memory})$

$1 / T_{IP[0]} = MIN(6 * 8, 40) / 1.0 = 40$ $f \neq 1$
$1 / T_{IP[1]} = MIN(B_1 * I_1, A * P_{peak}) / f$ $f \neq 0$ MOOT f = 0
$1 / T_{memory} = 10 * 8 = 80$ $I_{avg} = 8$ since f = 0
Perf = MIN(40 , --, 80) = 40

6**B** ----------
Ppeak = 40 Gops/s, Bpeak = 10 Gbytes/s, A = 5, B0 = 6 and B1 = 15.
I0 = 8 operations/byte on IP[0], I1 =0.1 for IP[1], and f =0.75.

$1 / T_{IP[0]} = MIN(B_0 * I_0, P_{peak}) / (1 - f)$ $f \neq 1$
$1 / T_{IP[1]} = MIN(B_1 * I_1, A * P_{peak}) / f$ $f \neq 0$
$1 / T_{memory} = B_{peak} * I_{avg}$ $I_{avg} = 1/[(1- f)/ I_0) + (f / I_1)]$
Perf = $MIN(1/T_{IP[0]}, 1/T_{IP[1]}, 1/T_{memory})$

$1 / T_{IP[0]} = MIN(6 * 8, 40) / 0.25 = 40/0.25 = 160$
$1 / T_{IP[1]} = MIN(15 * 0.1, 5 * 40) / 0.75 = 1.5/0.75 = 2$
$1 / T_{memory} = 10 * I_{avg}$ $I_{avg} = 1/[(0.25/ 8) + (0.75 / 0.1)] = 0.13278$
$1 / T_{memory} = 10 * 0.13278 = 1.3$
Perf = MIN(160, 2, 1.3) = 1.3

6C ----------
Ppeak = 40 Gops/s, Bpeak = 30 Gbytes/s, A = 5, B0 = 6 and B1 = 15.
I0 = 8 operations/byte on IP[0], I1 =0.1 for IP[1], and f =0.75.

$1 / T_{IP[0]} = MIN(B_0 * I_0, P_{peak}) / (1 - f)$ $f \neq 1$
$1 / T_{IP[1]} = MIN(B_1 * I_1, A * P_{peak}) / f$ $f \neq 0$
$1 / T_{memory} = B_{peak} * I_{avg}$ $I_{avg} = 1/[(1- f)/ I_0) + (f / I_1)]$
Perf = $MIN(1/T_{IP[0]}, 1/T_{IP[1]}, 1/T_{memory})$

$1 / T_{IP[0]} = MIN(6 * 8, 40) / 0.25 = 40/0.25 = 160$
$1 / T_{IP[1]} = MIN(15 * 0.1, 5 * 40) / 0.75 = 1.5/0.75 = 2$
$1 / T_{memory} = 30 * I_{avg}$ $I_{avg} = 1/[(0.25/ 8) + (0.75 / 0.1)] = 0.13278$
$1 / T_{memory} = 30 * 0.13278 = 3.98$
Perf = MIN(160, 2, 3.98) = 2.0

6D ----------
Ppeak = 40 Gops/s, Bpeak = 20 Gbytes/s, A = 5, B0 = 6 and B1 = 15.
I0 = 8 operations/byte on IP[0], I1 =8 for IP[1], and f =0.75.
$1 / T_{IP[0]} = MIN(B_0 * I_0, P_{peak}) / (1 - f)$ $f \neq 1$
$1 / T_{IP[1]} = MIN(B_1 * I_1, A * P_{peak}) / f$ $f \neq 0$
$1 / T_{memory} = B_{peak} * I_{avg}$ $I_{avg} = 1/[(1- f)/ I_0) + (f / I_1)]$
Perf = $MIN(1/T_{IP[0]}, 1/T_{IP[1]}, 1/T_{memory})$

$1 / T_{IP[0]} = MIN(6 * 8, 40) / 0.25 = 40/0.25 = 160$
$1 / T_{IP[1]} = MIN(15 * 8, 5 * 40) / 0.75 = 120/0.75 = 160$
$1 / T_{memory} = 20 * 8 = 160$
Perf = MIN(160, 160 , 160) = 160

Numbers Behind Gables's 3-IPExample

TESTING IN PROGRESS

Ppeak = 40 Gops/s, Bpeak = 30 Gbytes/s, A_0 = 1, A_1 =3, A_2 = 5, B_0 = 6, B_1 = 15 and B_2 =10.
I_0 = 4, I_1 = 6, I2 = 8, f_0=20%, f_1 = 30%, and f_2 = 50%.

$1 / T_{IP[0]} = MIN(B_0 * I_0, A_0 * P_{peak}) / f_0 \quad f_0 \neq 0$
$1 / T_{IP[1]} = MIN(B_1 * I_1, A_1 * P_{peak}) / f_1 \quad f_1 \neq 0$
$1 / T_{IP[2]} = MIN(B_2 * I_2, A_2 * P_{peak}) / f_2 \quad f_2 \neq 0$
$I_{avg} = 1/[f_0/ I_0) + (f_1 / I_1) + (f_2 / I_2)]$
$1 / T_{memory} = B_{peak} * I_{avg}$
Perf = MIN($1/T_{IP[0]}$, $1/T_{IP[1]}$, $1/T_{IP[12}$, $1/T_{memory}$)

$1 / T_{IP[0]} = MIN(6 * 4, 1 * 40) / 0.20 = 24/0.20 = 120$
$1 / T_{IP[1]} = MIN(15 * 6, 3 * 40) / 0.30 = 90/0.30 = 300$
$1 / T_{IP[2]} = MIN(10 * 8, 5 * 40) / 0.50 = 80/0.50 = 160$
$I_{avg} = 1/[0.20/ 4) + (0.30 / 6) + (0.50/ 8)] = 1 / 0.1625 = 6.1538$
$1 / T_{memory} = 30 * 6.1538 = 185$
Perf = MIN(120, 300, 160, 185) = 120