Lecture

19

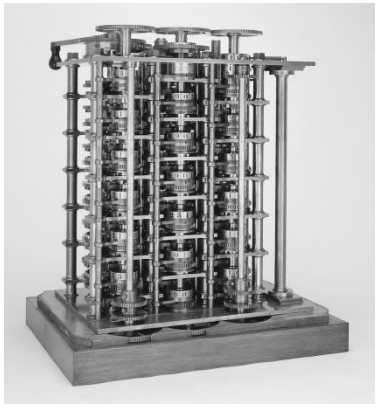
ECE M216A

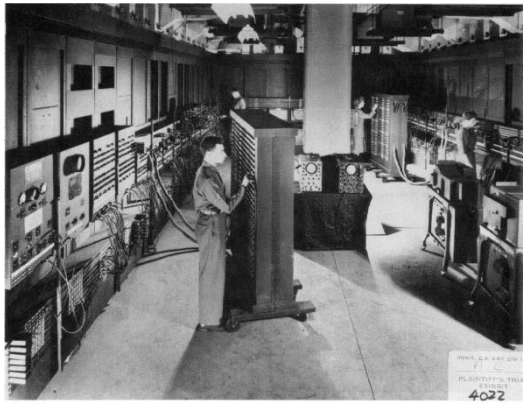
Compute Architectures

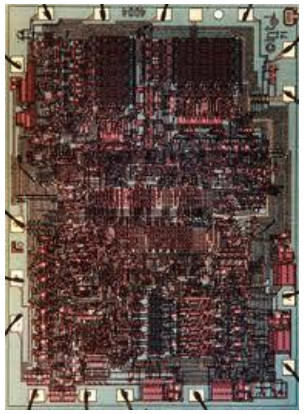
Prof. Dejan Marković

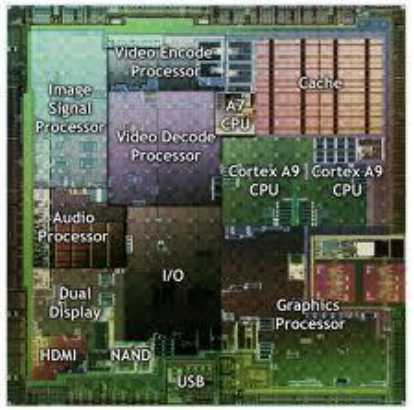
ee216a@gmail.com

A Changing Technology Landscape

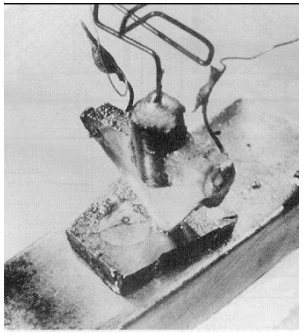


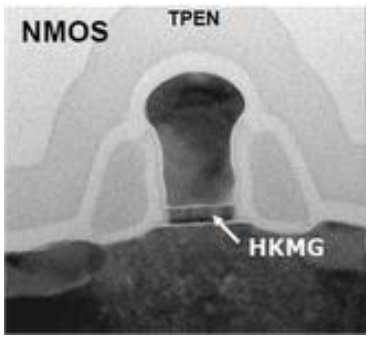






1835 1946 1971 2012





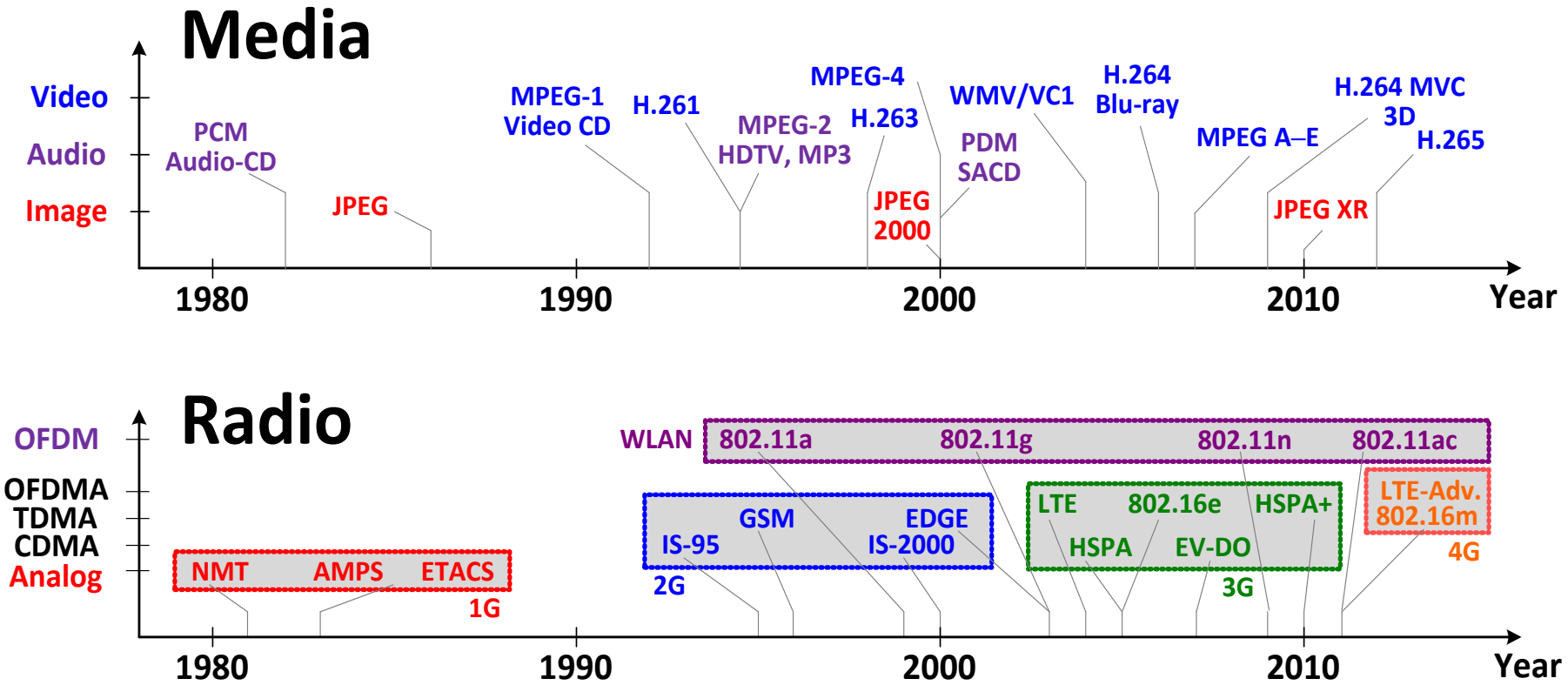
You will be soon designing with "post-CMOS" devices

Signal processing content expanding

Specialized hardware for energy efficiency

Keeping up with Standards

New standard = New chip?



An SoC Example: CPUs + Accelerators

Flexibility (CPUs) + Efficiency (Accelerators) side-by-side

NVIDIA Tegra 2

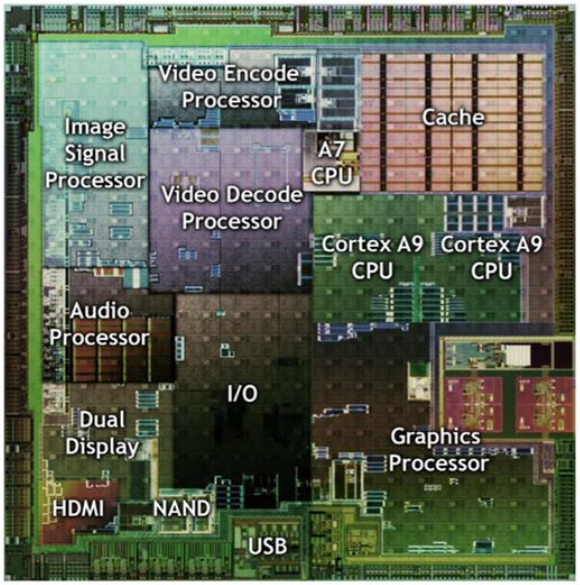
Accelerators (UnCore):

- **Increasingly larger fraction of chip area**
- **Low area utilization a.k.a. DARK silicon**
- **Accelerators for fixed standards**

Some Insights | Evolution of Tegra Chips

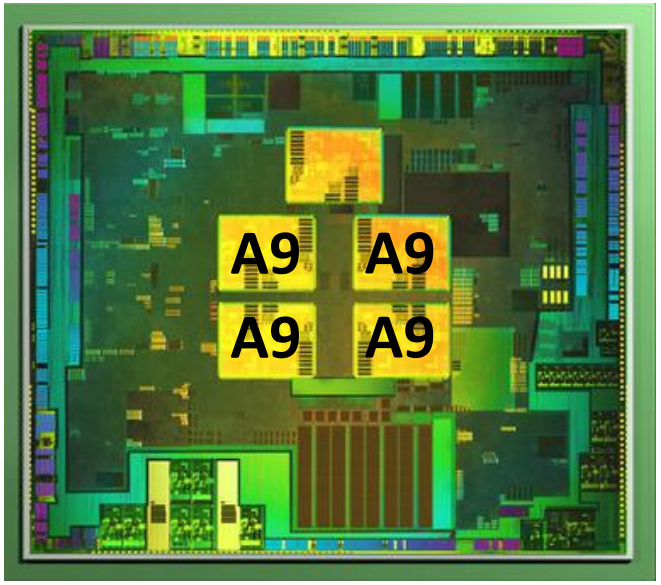
Customization, increasing number of cores…

Tegra 2 (2011)



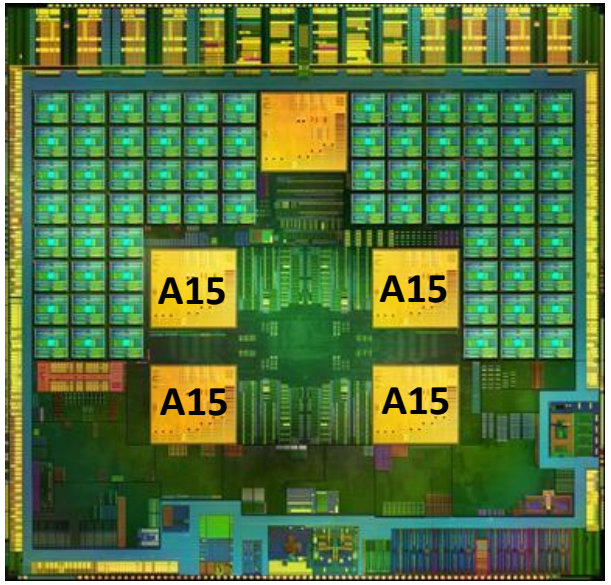
- **Dual-A9**

Tegra 3 (2012)



- **Quad-A9**
- **Power-saver core**

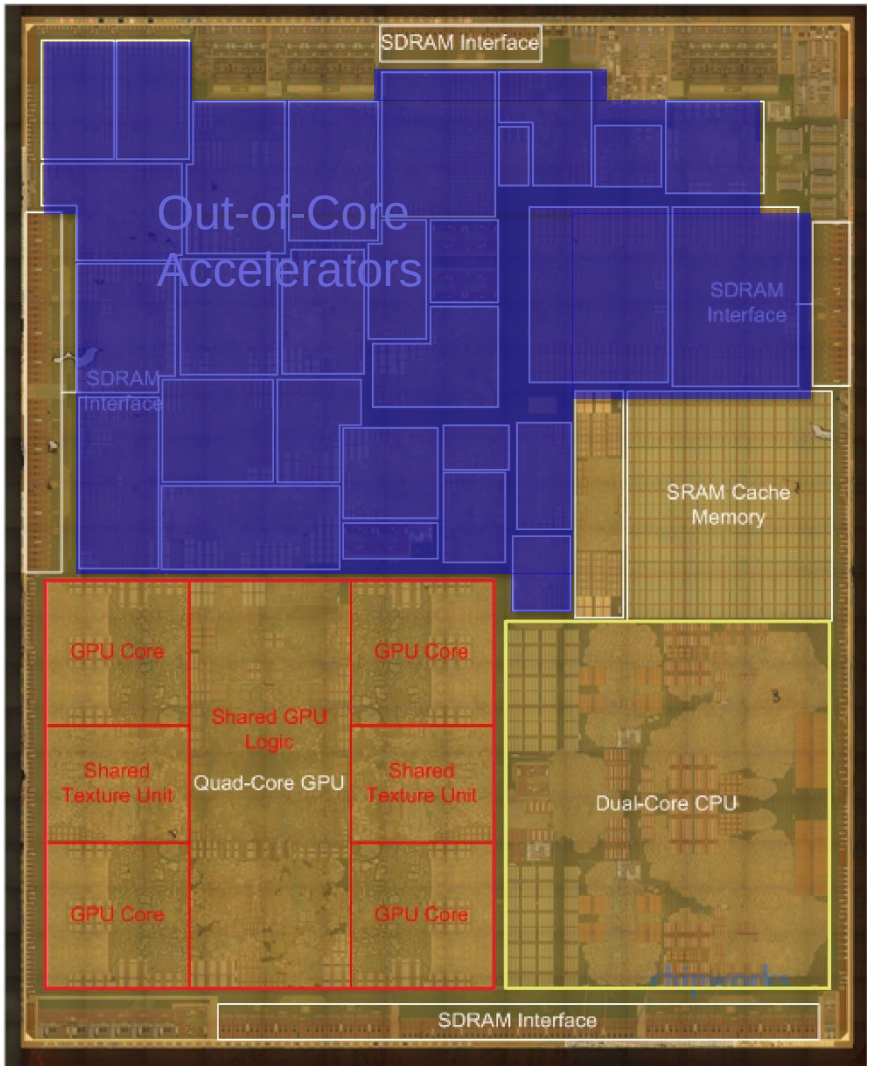
Tegra 4 (2013)

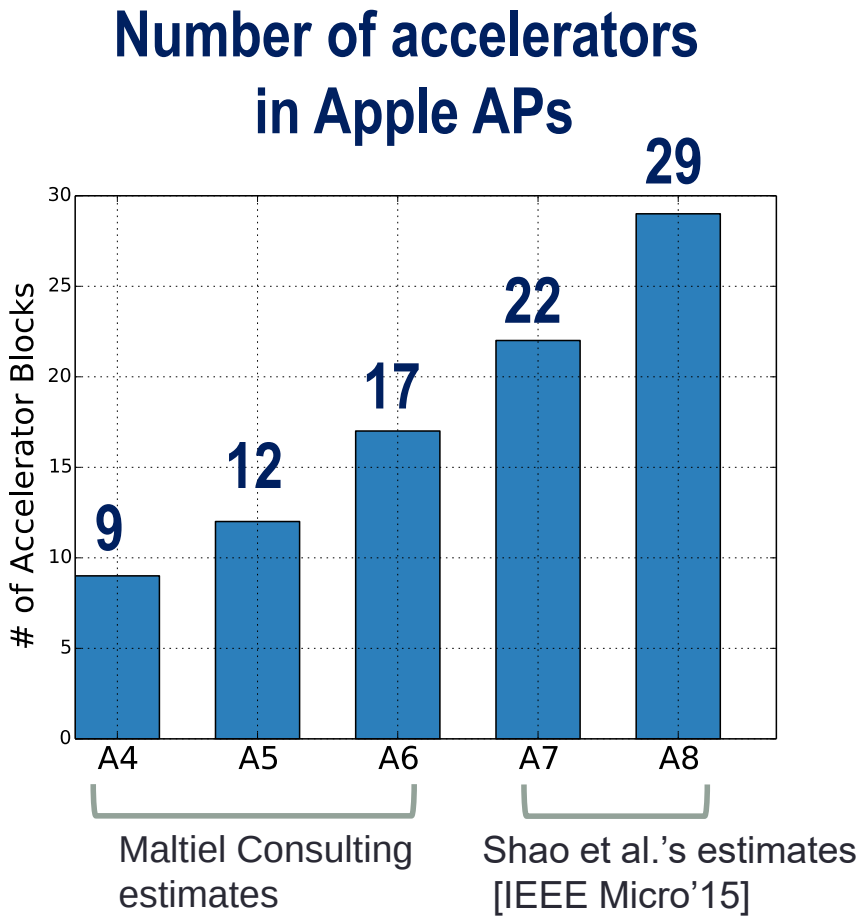


- **72 GPU cores**
- **LTE modem**
- **Computational camera**

From: Google Images

Heterogeneous Computing in Mobile SoCs





Courtesy: D. Brooks (Harvard)

Chip Utilization Drops Every Generation

Power-limited scaling = DARK silicon

Parameter	Classical const-E scaling	Leakage-limited scaling
Threshold, V_T	**1/S**	**1/U**
Supply, V_{DD}	**1/S**	**1/U**
Quantity, Q	S^2	S^2
Frequency, F	S	S
Capacitance, C	1/S	1/S
Power, P	**1**	**S^2/U^2**
Utilization = 1/P	**1**	**U^2/S^2**

$P \propto Q \cdot F \cdot C \cdot V_{DD}^2$

Utilization drop: **S^2/U^2**

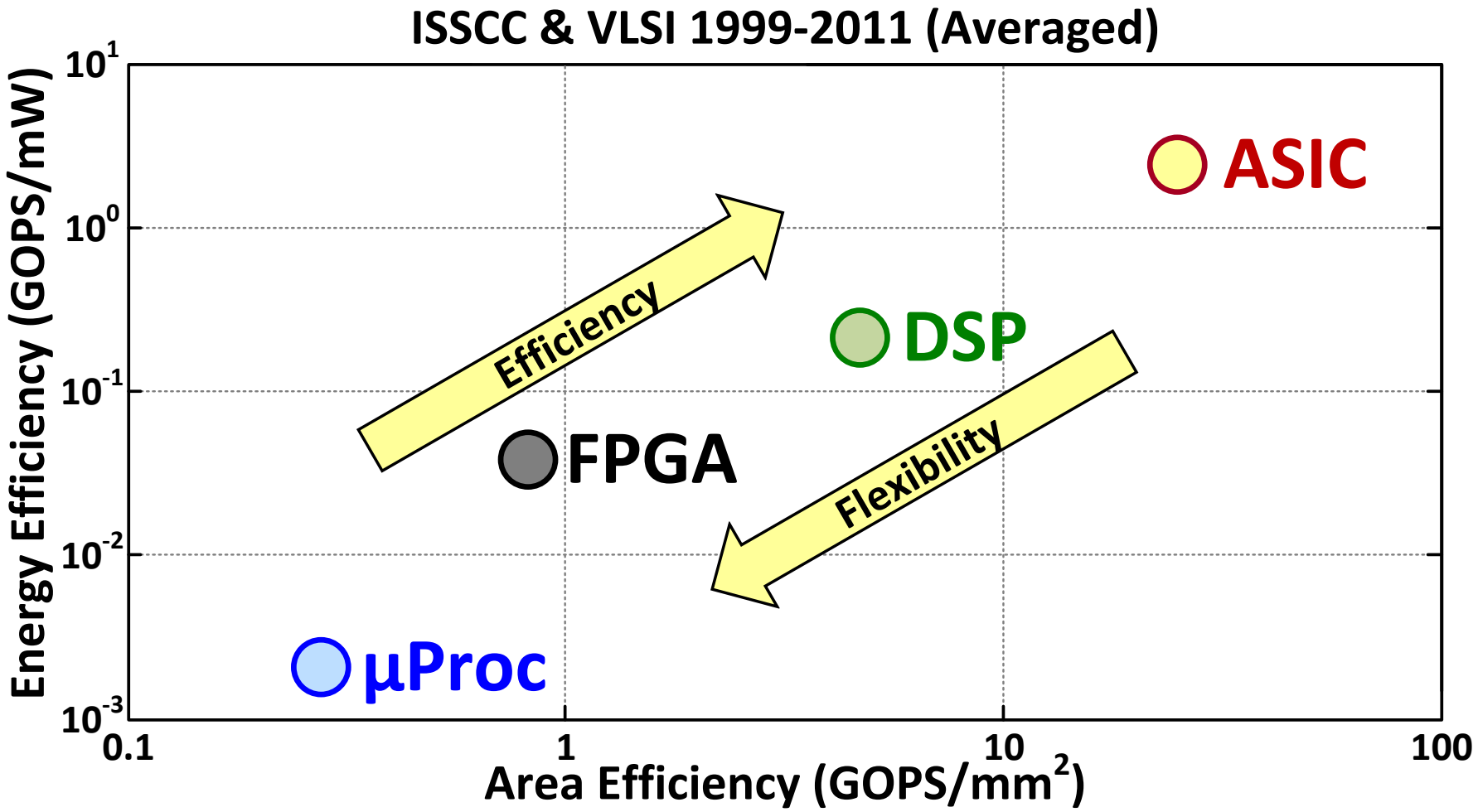
The Utilization Wall (Assume U = 1)

- **Voltage scaling is broken (U < S)**
- **Improvements for a fixed chip size**
  - Computing capability: **2.8x**
    - Transistor count: 2x
    - Operating frequency: 1.4x
  - Energy efficiency: **1.4x**

2x shortfall ⇨ **DARK** silicon

Power-limited scaling

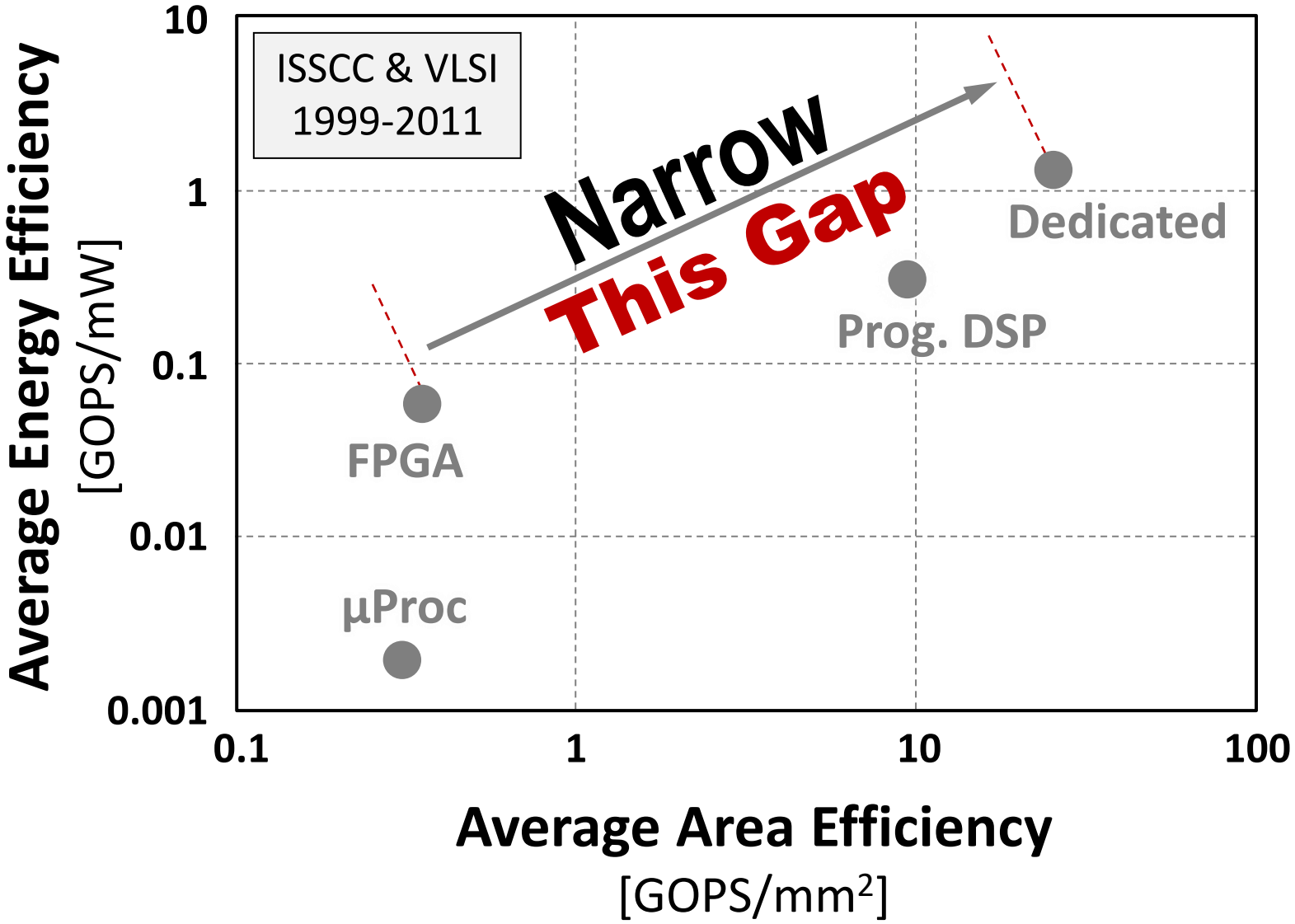
Efficiency vs Flexibility: Inherent Tradeoff



Ways to Achieve FLEXIBILITY

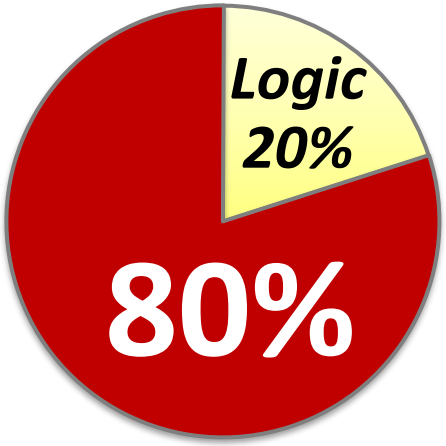
	Software	Hardware
Feature	Programmable DSP	FPGA (Flexible DSP)
Architecture	Fixed	Reconfigurable
Operations	Conditional	Repetitive
Multi-core	Hard	Easy
Throughput	Low/mid	High

Efficient & Flexible Hardware?

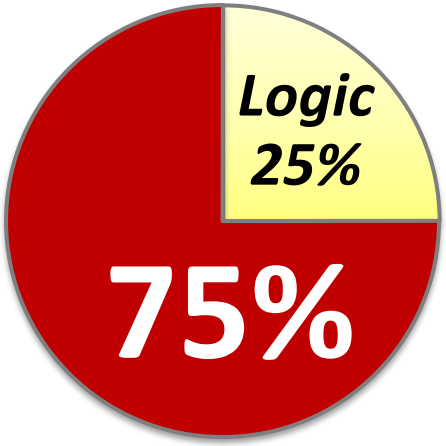


WHY are FPGAs **Inefficient?**

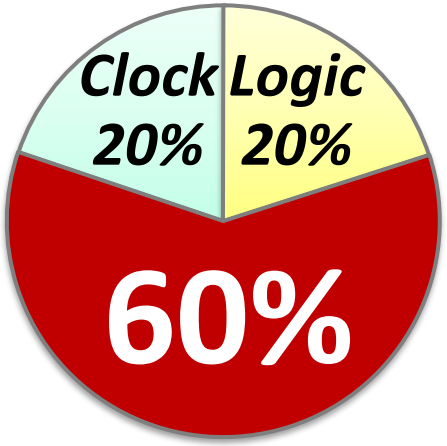
Area



Delay



Power



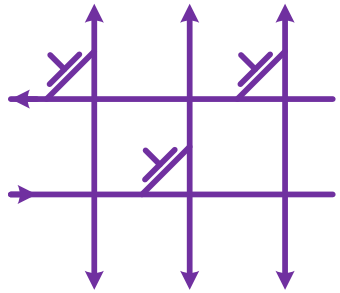
Interconnect

I. Kuon, et al., Found. & Trends in Elec. Design Automation 2007.

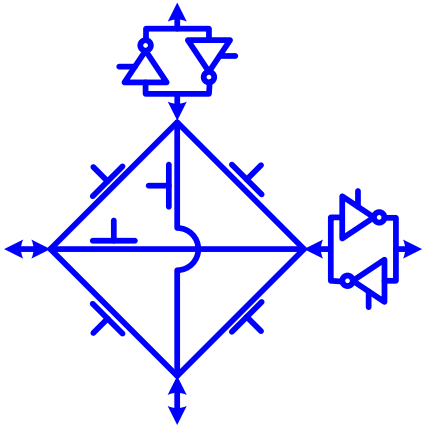
I. Bolsens, MPSOC 2006; B. Calhoun, et al., Proc. IEEE 2010.

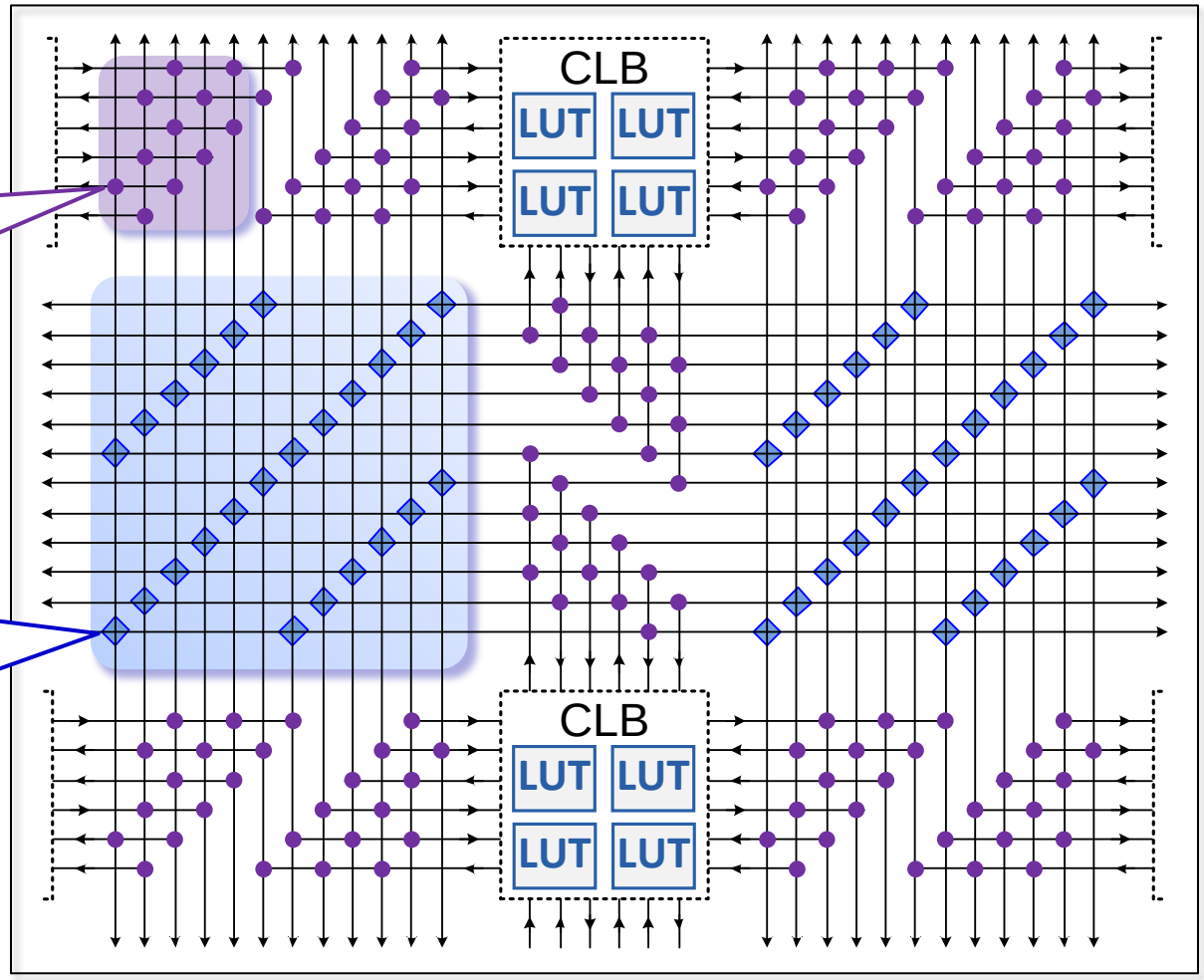
2D-Mesh Interconnects

Connection box



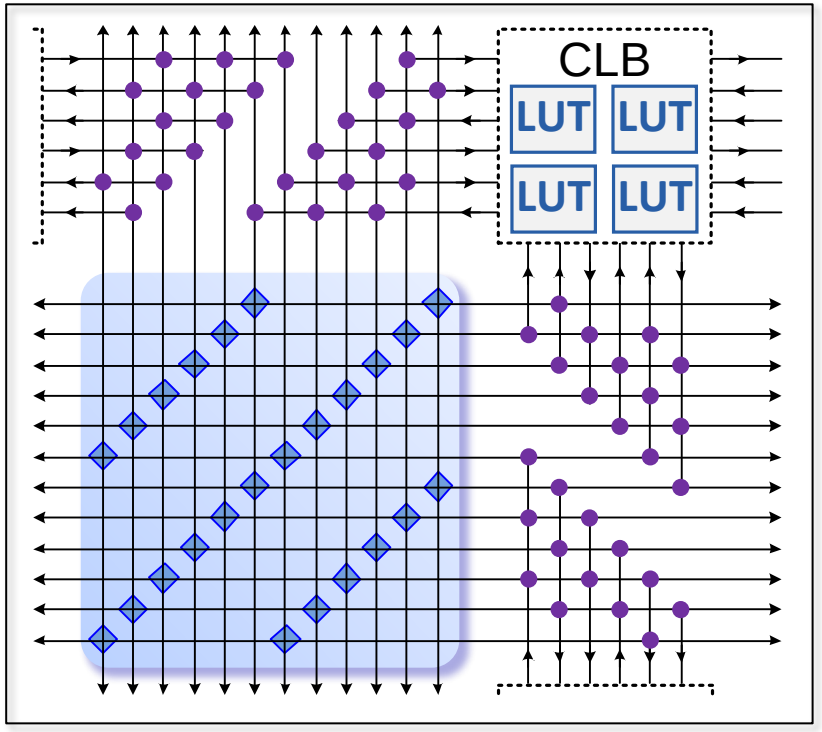
Switch box



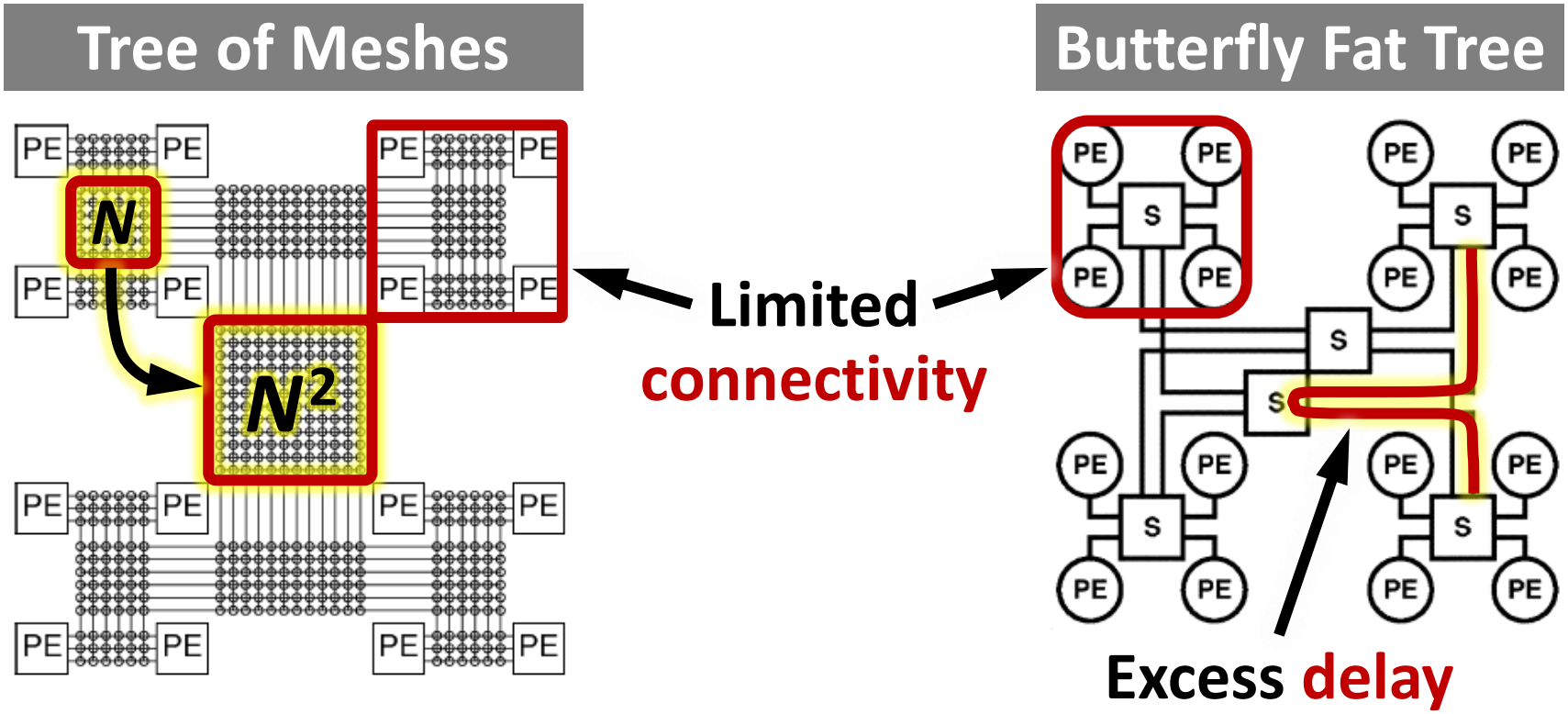


2D-Mesh is NOT Scalable

- From $O(N^2)$ complexity
- Full connectivity impractical



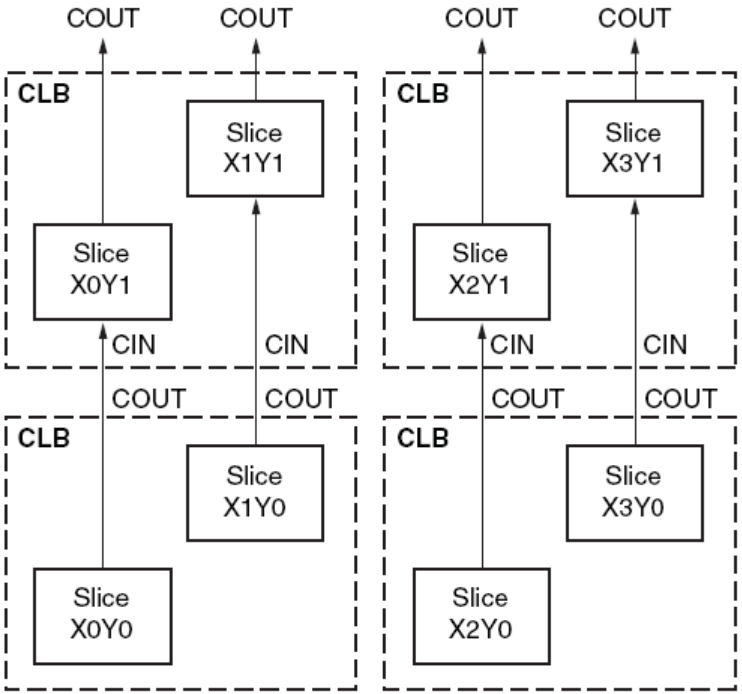
Hierarchical Networks

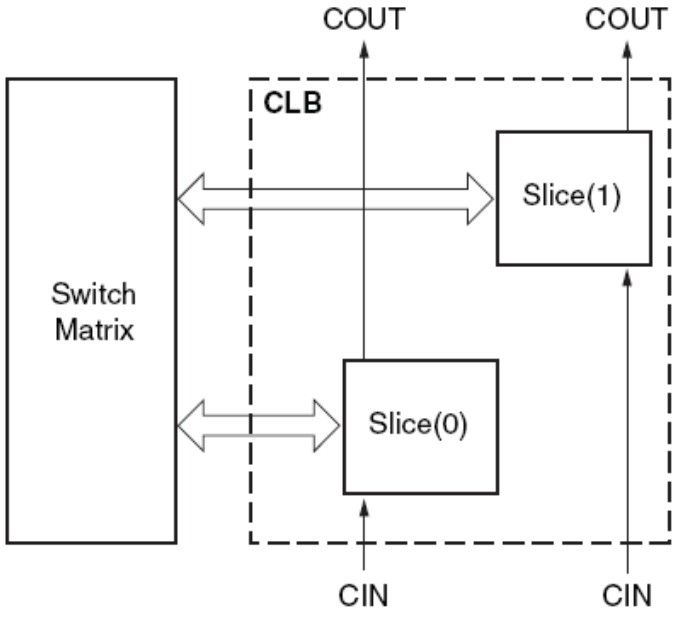


From: A. DeHon, VLSI 10/2004.

Virtex-5 Configurable Logic Blocks (CLBs)

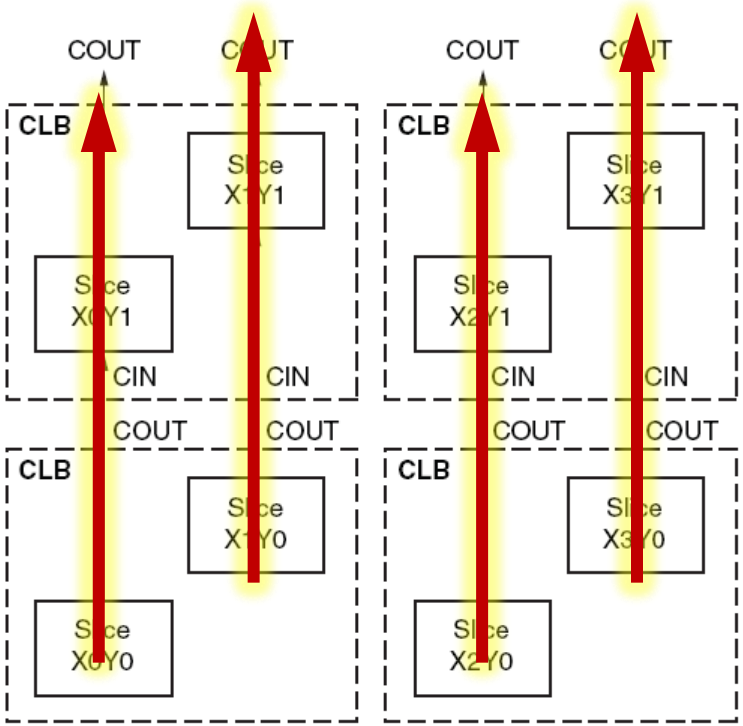
- CLBs implement seq. and comb. functions
- CLB = two unconnected independent slices

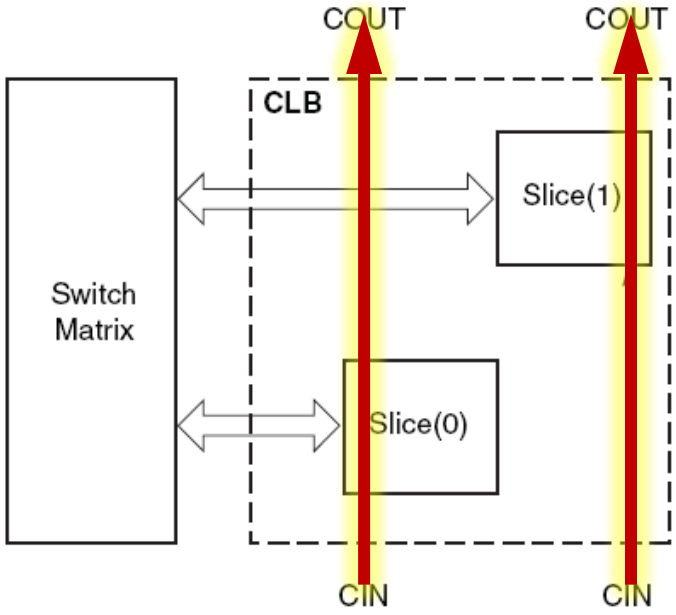




Virtex-5 CLBs: Fast Carry Logic

- Each slice connected to the global routing paths
- Slice columns connected by **fast carry logic**

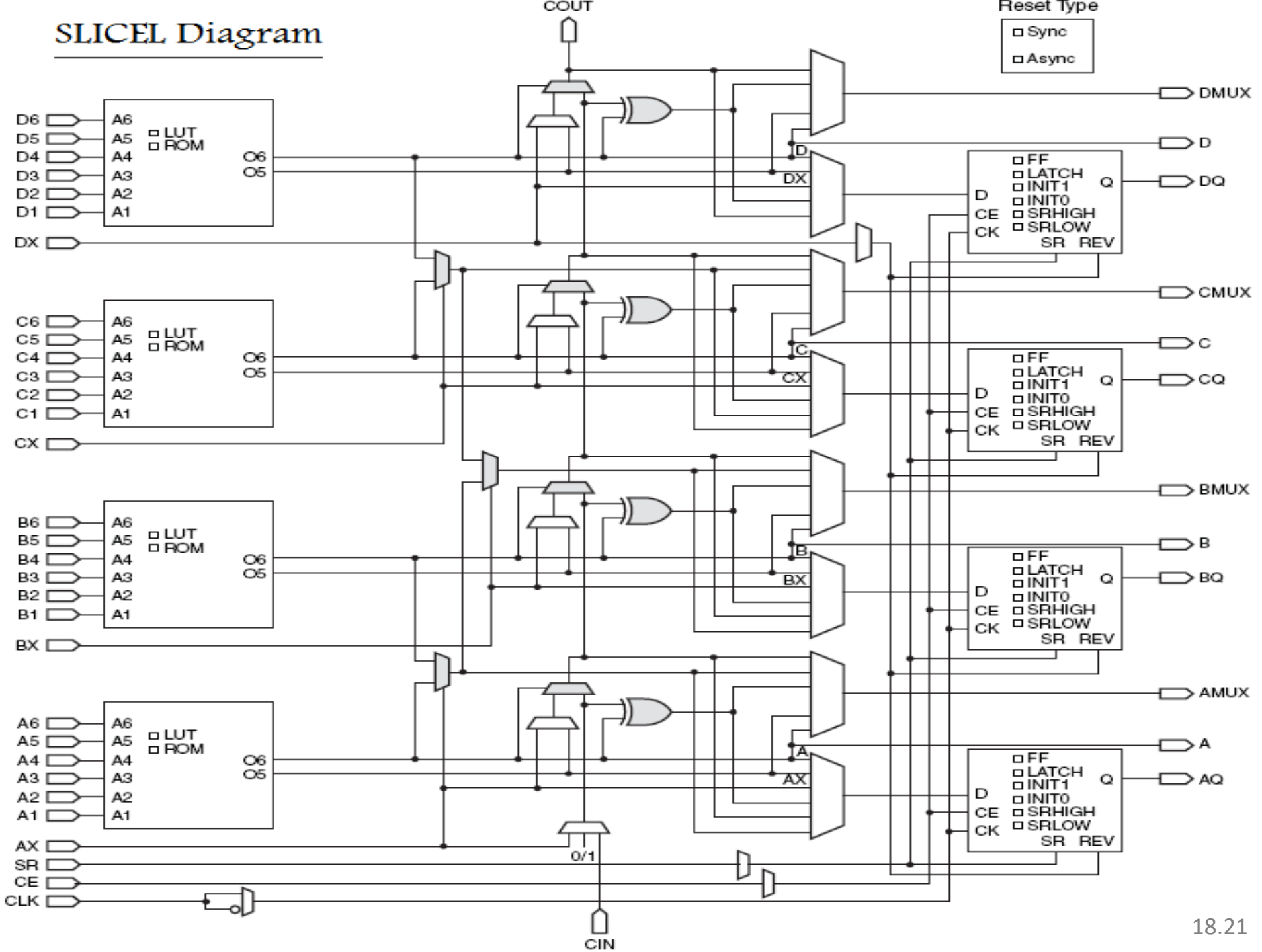


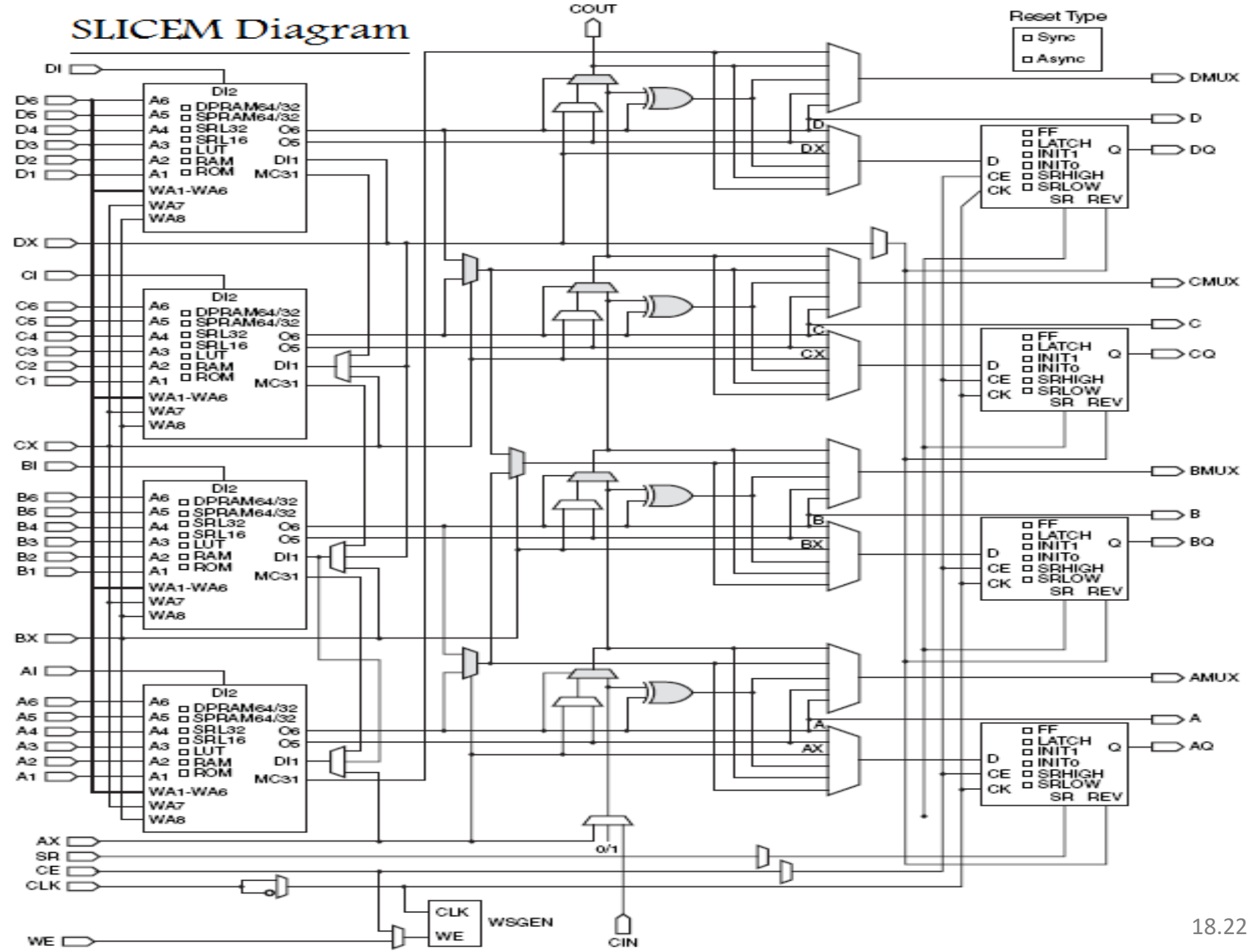


Two Types of Slices

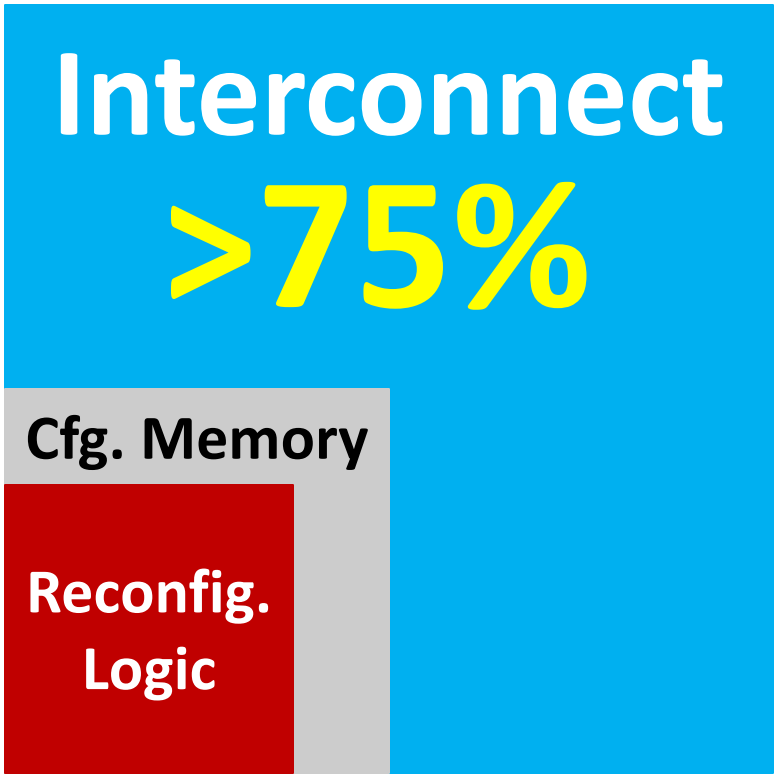
- **SLICEL: regular**
  - Every CLB contains one or two SLICEL
- **SLICEM: more functions**
  - Every other CLB contains a SLICEM

SLICEL Diagram




SLICEM Diagram
COUT
Reset Type
Sync
Async
DI
D6
D5
D4
D3
D2
D1
DX
CI
C6
C5
C4
C3
C2
C1
CX
BI
B6
B5
B4
B3
B2
B1
BX
AI
A6
A5
A4
A3
A2
A1
AX
SR
CE
CLK
WE
DI2
A6
A5
A4
A3
A2
A1
DPRAM64/32
SPRAM64/32
SRL32
SRL16
LUT
RAM
ROM
O6
O5
DI1
MC31
WA1-WA6
WA7
WA8
D
DX
C
CX
B
BX
A
AX
FF
LATCH
INIT1
INIT0
SRHIGH
SRLOW
SR REV
Q
CE
CK
DMUX
D
DQ
CMUX
C
CQ
BMUX
B
BQ
AMUX
A
AQ
0/1
CLK
WE
WSGEN
CIN

Routing Architecture Dominates Chip Area



What is the Cost of Flexibility?

- **We need technical metrics to compare flexible and non-flexible implementations**
  - A power metric because of thermal limitations
  - An energy metric for portable operation
  - A cost metric related to the area of the chip
  - Performance (computational throughput)

Let's use metrics normalized to the amount of computation being performed

Material based on ISSCC 2002 evening session lecture:

R.W. Brodersen, "Technology, Architecture, and Applications," in *Proc. Int. Solid-State Circuits Conf.,* Special Topic Evening Session: Low Voltage Design for Portable Systems, Feb. 2002.

Definitions

Computation

- Operation = OP =algorithmically interesting computation (i.e. multiply, add, delay)
- MOPS = Millions of OP's per Second
- N_{op} = Number of parallel OP's in ***each*** clock cycle

Power

- P_{chip} = Total power of chip = $A_{chip} \cdot C_{sw} \cdot V_{DD}^2 \cdot f_{clk}$
- C_{sw} = Switched Cap / mm^2 = $P_{chip} / (A_{chip} \cdot V_{DD}^2 \cdot f_{clk})$

Area

- A_{chip} = Total area of chip
- A_{op} = Average area of each operation = A_{chip} / N_{op}

Energy Efficiency Metric: MOPS/mW

- **How much computing (number of operations) can we can do with a finite energy source (e.g. battery)?**

$$\text{Energy efficiency} = \frac{\text{Number of useful operations}}{\text{Energy required}}$$

$$= \frac{\text{Number of operations}}{\text{NanoJoule}} = \frac{\text{OP}}{\text{nJ}}$$

$$= \frac{\text{OP/sec}}{\text{nJ/sec}} = \frac{\text{MOPS}}{\text{mW}}$$

$$= \text{Power efficiency}$$

⇨ **Energy efficiency = Power efficiency**

Energy and Power Efficiency

OP/nJ = MOPS/mW

- **Interestingly, the energy efficiency metric for energy constrained applications (OP/nJ) for a fixed number of operations, is the same as that for thermal (power) considerations when maximizing throughput (MOPS/mW)**

- **So let's look at a number of chips to see how these efficiency numbers compare**

ISSCC Chips (22nm – 0.18µm)

Chip	Year	Paper	Description
1	2009	3.8	Dunnington
2	2010	5.7	MSG-Passing
3	2010	5.5	Wire-speed
4	2011	4.4	Godson-3B
5	2013	3.5	Godson-3B1500
6	2011	15.1	Sandy Bridge
7	2012	3.1	Ivy Bridge
8	2011	15.4	Zacate
9	2013	9.4	ARM-v7A

Chip	Year	Paper	Description
10	2012	10.6	3D Proc.
11	2013	9.3	H.264
12	2012	28.8	Razor SIMD
13	2011	7.1	3DTV
14	2011	7.3	Multimedia
15	2011	19.1	ECG/EEG
16	2010	18.4	Obj. Recog.
17	2012	12.4	Obj. Recog.
18	2013	9.8	Obj. Recog.
19	2011	7.4	Neural Network
20	2013	28.2	Visual. Recog.

Chip type:

Microprocessor

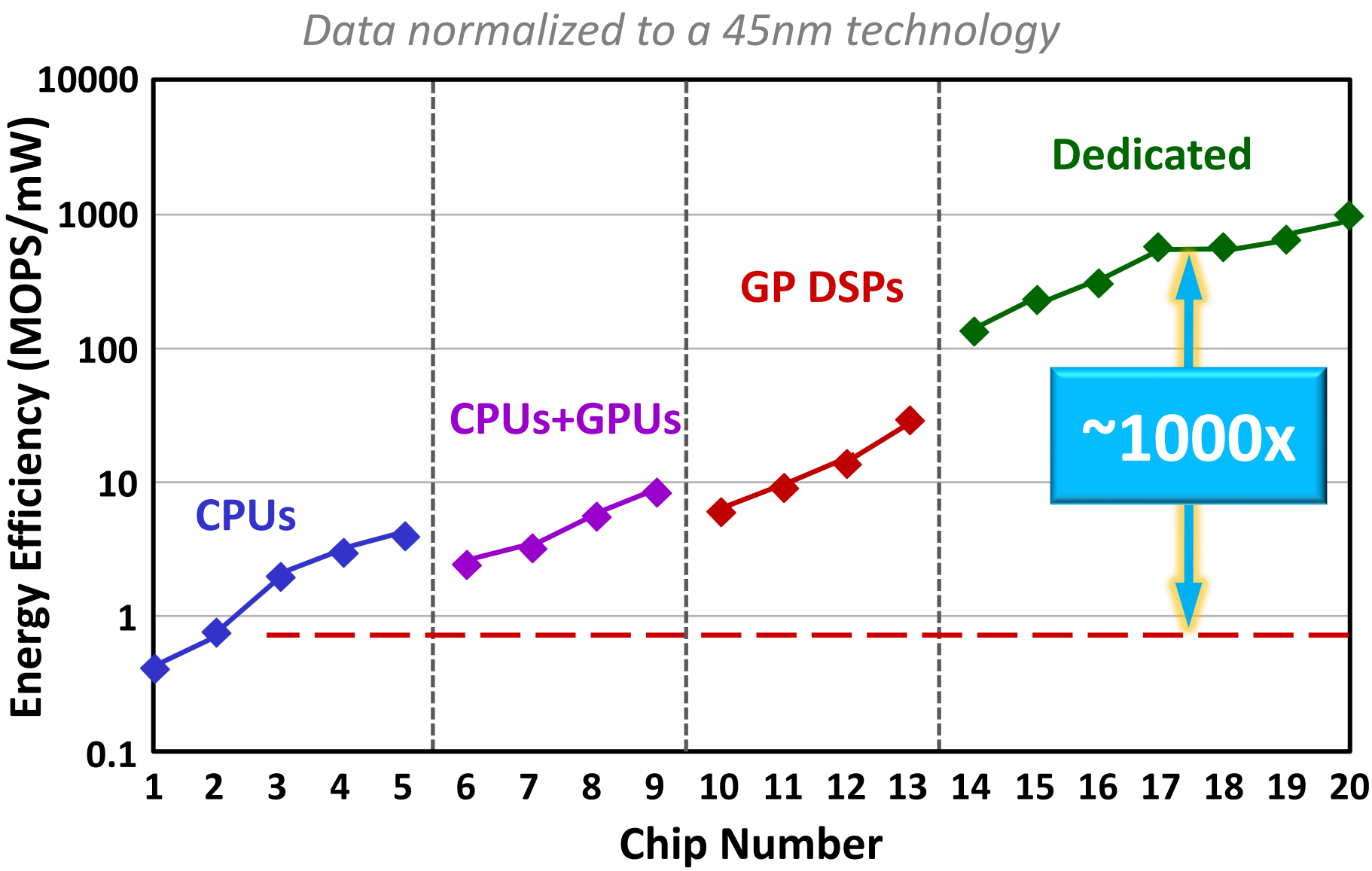
Microprocessor + GPU

General purpose DSP

Dedicated design

Chips published at ISSCC over a 5-year span

Energy Efficiency (MOPS/mW or OP/nJ)



Why Such a Big Difference?

Lets look at the components of MOPS/mW

- **The operations per second:**

  $$\text{MOPS} = f_{clk} \cdot N_{op}$$

- **The power:**

  $$P_{chip} = A_{chip} \cdot C_{sw} \cdot V_{DD}^2 \cdot f_{clk}$$

- **The ratio (MOPS / P_{chip}) gives the MOPS/mW**

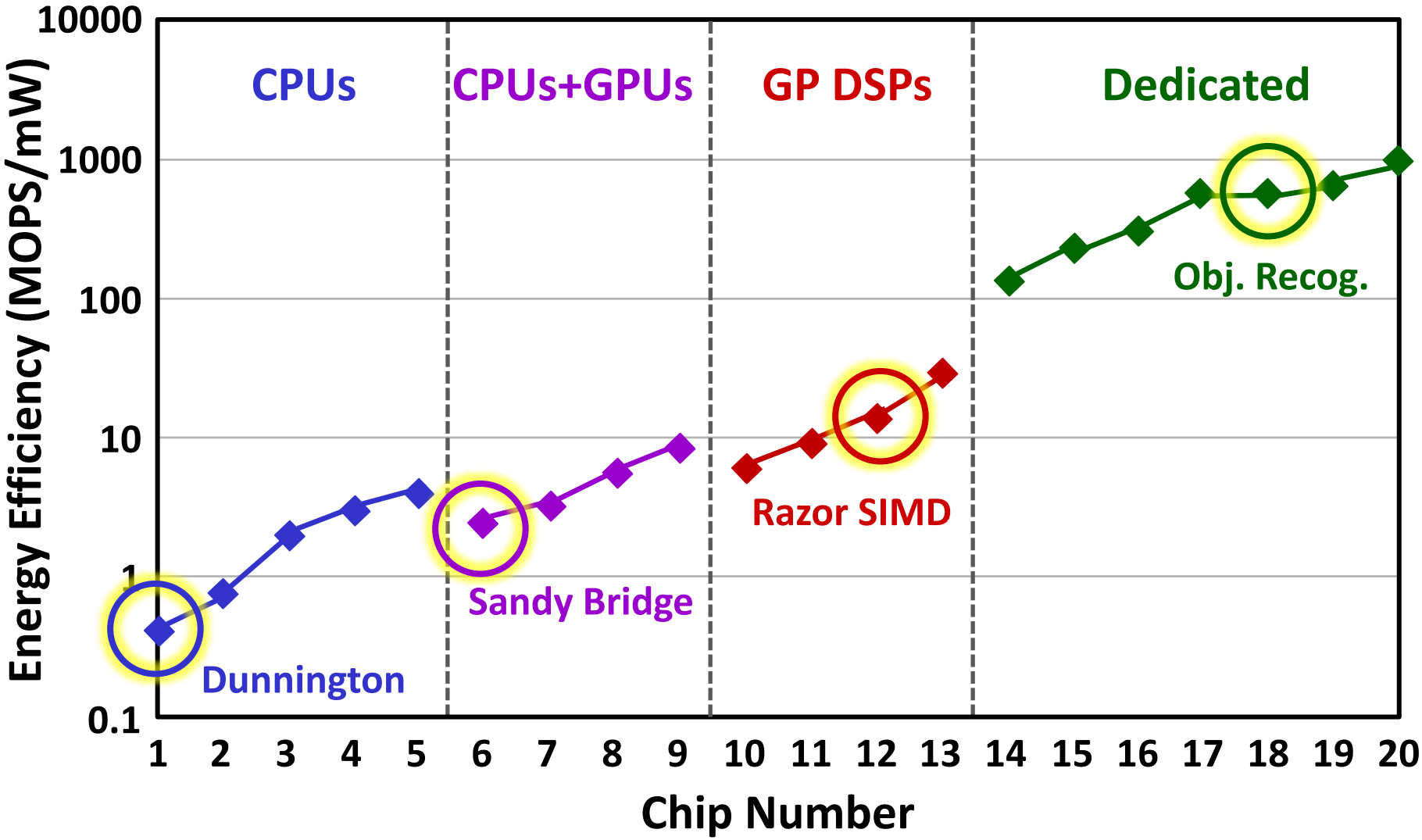
  $$= (f_{clk} \cdot N_{op}) / (A_{chip} \cdot C_{sw} \cdot V_{DD}^2 \cdot f_{clk})$$

Simplifying, $\text{MOPS/mW} = 1 / (A_{op} \cdot C_{sw} \cdot V_{DD}^2)$

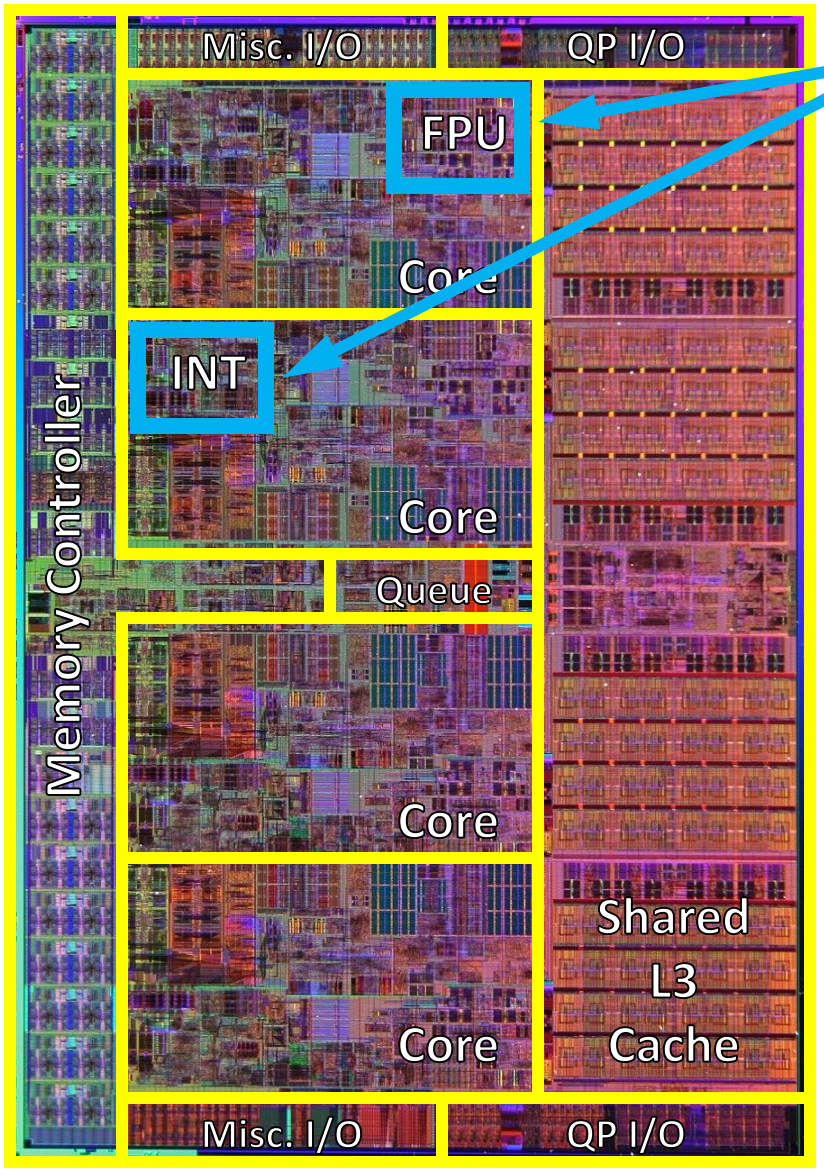
So lets look at the 3 components: V_{DD}, C_{sw} and A_{op}

Let's Look at Some Chips to Actually See the Different Architectures

We'll look at one from each category…



Microprocessor: MOPS/mW = 0.33



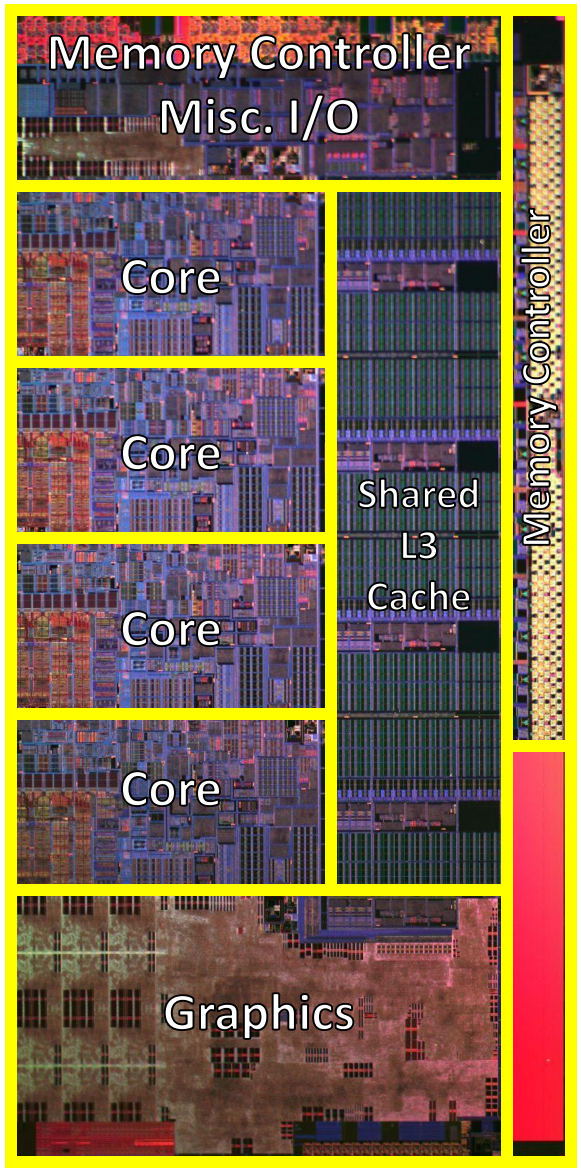
The only circuitry which supports "useful operations"
All the rest is overhead
to support the time multiplexing

N_{op} = 16
f_{clk} = 2.66 GHz
=> 42.56 GIPS

Sixteen operations each clock cycle, so
$A_{op} = A_{chip}$ /16 = 31.4 mm^2

Power = 130 Watts

Microprocessor + GPU: MOPS/mW = 2.46



CPU: 4 cores (8 threads)
=> 4 ops per thread (SIMD)
=> N_{op} = 32
f_{clk} = 3.4 GHz => 108.8 GIPS

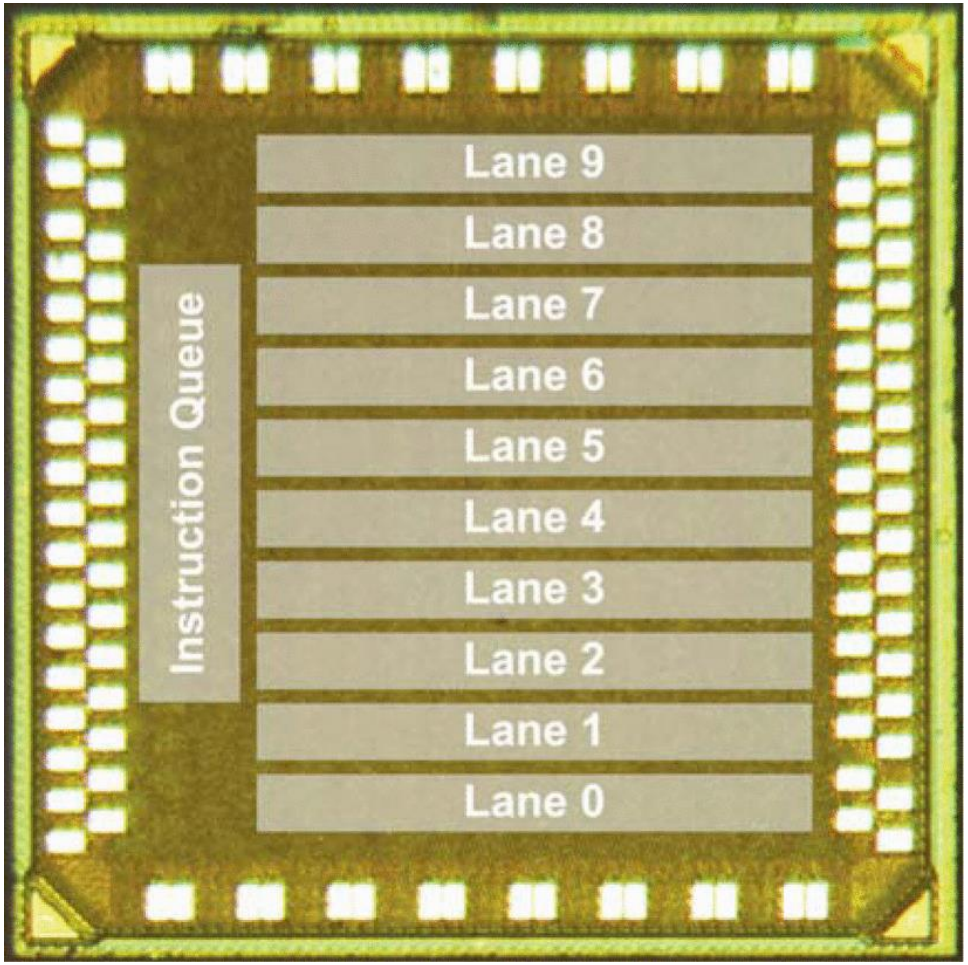
GPU: 12 cores => 8 ops per core (SIMD)
=> N_{op} = 96
f_{clk} = 1.3 GHz => 124.8 GIPS

TOTAL: 233.6 GIPS

~69 operations each clock cycle (CPU), so
$A_{op} = A_{chip}$ /69 = 3.14 mm^2

Power = 95 Watts

General Purpose DSP: MOPS/mW = 16



Same granularity (a datapath), more parallelism

10 Parallel processors
(2 for estimation and ECC)

$N_{op} = 8$

$f_{clk} = 550$ MHz

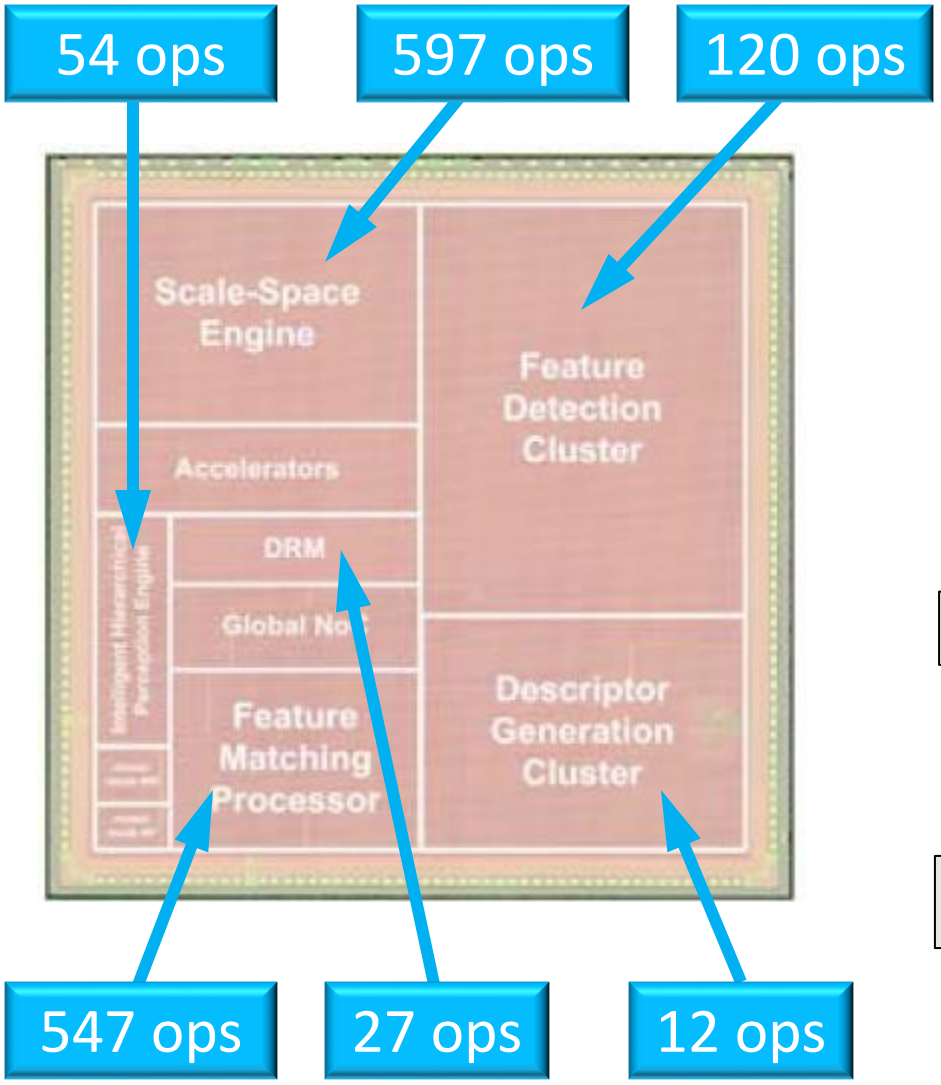
=> 4.4 GOPS

Eight operations each clock cycle, so

$A_{op} = A_{chip} / 8 = 0.5$ mm^2

Power = 275 mW

Dedicated Design: MOPS/mW = 650



Fully parallel mapping of object recognition algorithm. No time multiplexing.

N_{op} = 1357

f_{clk} = 200 MHz

=> 271.4 GOPS

$A_{op} = A_{chip}$ /1357 = 0.02 mm^2

Power = 420 mW

Hardware / Software

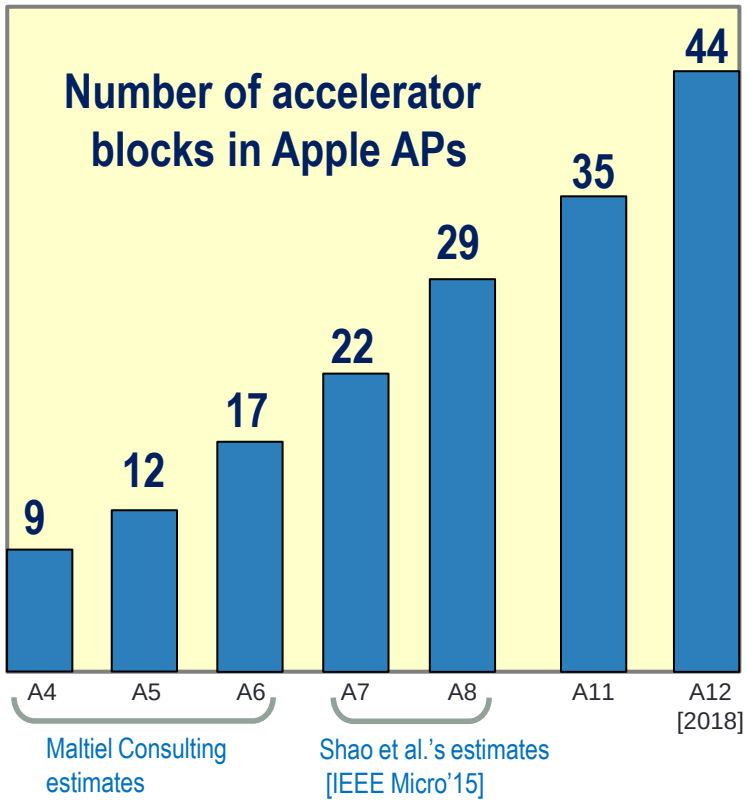
There is no software/hardware tradeoff!

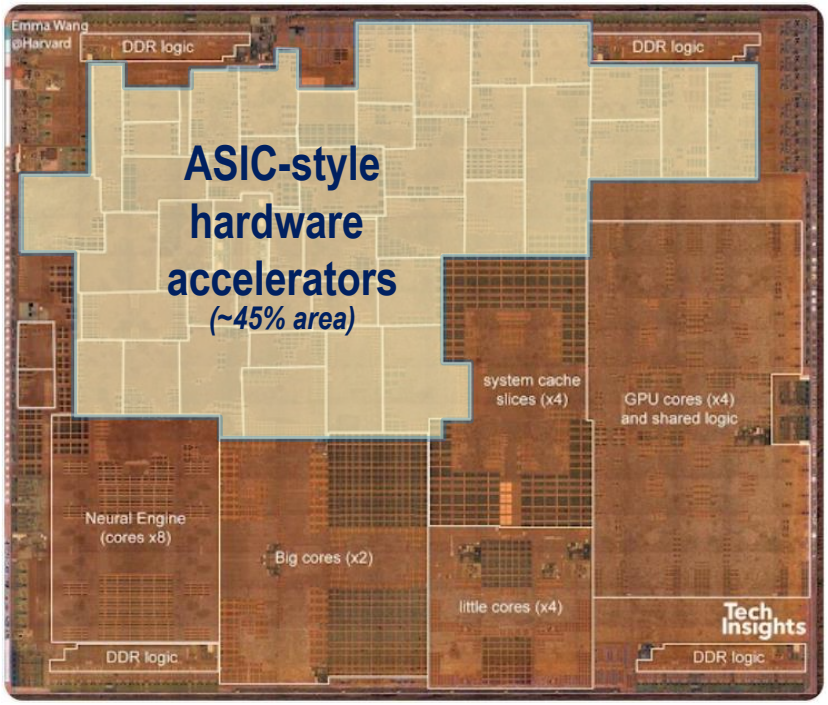
- The difference between hardware and software in performance, power and area is so large that there is no "tradeoff"
- It is reasons other than energy, performance or cost that drives a software solution (e.g. business, legacy, …)
- The "Cost of Flexibility" is extremely high, so the other reasons better be good!

SoC Today: The Apple Approach

Linear growth in the "UnCore" units, exceeding ½ of SoC chip area today (A15)

Increasing "dark silicon" area (A12: ~45%, A15: ~55%), <10% chip is active

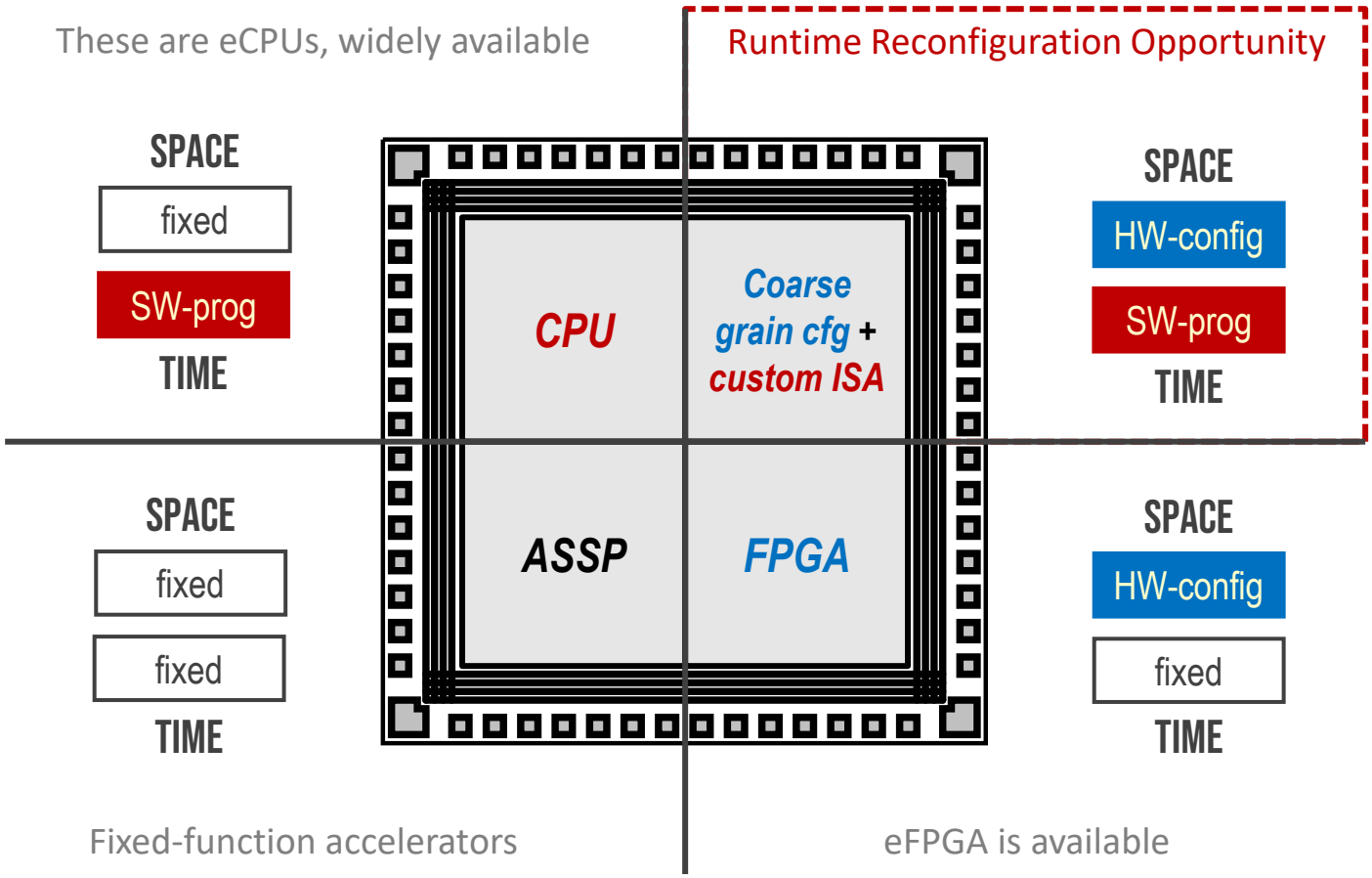




Apple A12 die photo

Space-Time Quadrants of Architecture "X"

- **These are canonical building blocks of modern SoCs**



Coarse-Grained Reconfigurable Array (CGRA)

Classification of CGRAs

Architecture	Year	Programming model*	Computation model	Execution model**	Specifications
Xputer [8]	1991	D	SCSD	SSE	
PADDI [9]	1992	I	SCSD	SSE	
PADDI-2 [67]	1993	D	SCMD	DSD	
RAW [68]	1997	I	MCMD	DSD	*More like a multicore processor*
PipeRench [69]	1998	D	SCMD	SSE	
Morphosys [11]	2000	I	SCMD	SSE	
Wavescalar [62, 70]	2003	I	MCMD	DDD	*Dataflow-driven ISA*
PACT-XPP [13]	2003	I/C	SCSD	DSD	
DRP [26]	2004	I	SCSD	SSE	*programmable FSM controller*
ADRES [10]	2004	I	SCSD	SSE	*VLIW controller*
ASH [57]	2004	D	SCSD	SSD	
TRIPS [12]	2004	I	MCMD	DSD	*Dataflow-driven ISA*
CCA [52]	2004	*transparent*	SCSD	SSE	*Runtime-generated configurations*
Tartan [60]	2006	I	MCMD	DSD	*Asynchronous circuit*
TFlex [71]	2007	I	MCMD	DSD	*Dataflow-driven ISA*
RICA [72]	2008	I	SCSD	SSE	
PPA [54]	2009	I	SCSD	SSE	*Polymorphic configurations*
TCPA [50]	2009	D	SCSD	SSE	
C-Cores [73]	2010	I	SCSD	SSE	*Targeted reconfigurability, ASIC-like*
DySER [47]	2012	I	SCSD	SSD	
REMUS [30]	2013	I	SCSD	SSE	
Triggered Inst. [61]	2013	D	MCMD	DSD	
T3 [74]	2013	I/C	MCMD	DSD	*Dataflow-driven ISA*
SGMF [75]	2014	I	MCMD	DDD	
FPCA [76]	2014	I	SCSD	SSE	
DynaSPAM [53]	2015	*transparent*	SCMD	SSD	*Based on PipeRench*
NDA [77]	2015	-	SCSD	SSE	*Process-in-memory*
HARTMP [78]	2016	I	SCMD	DSD	
DORA [51]	2016	*transparent*	SCSD / SCMD	SSD	*Based on DySER*
HRL [79]	2016	D/I	SCSD	SSE	*Process-in-memory, mix-grained*
HReA [16]	2017	I	SCSD	SSD	*General-purpose*
Plasticine [19]	2017	D	SCMD / MCMD	SSD	*Parallel-pattern-based programming*
Stream-dataflow [20]	2017	I	SCSD	DSD	*Vector memory interface*
CGRA-ME [80]	2017	I	SCSD	SSE	*ADRES-like*
Wave DPU [18]	2017	I/C	SCSD	SSD	*Commercial product for DNN*
PX-CGRA [81]	2018	-	SCSD	SSE	*Approximate PEs*
i-DPs CGRA [82]	2018	-	SCMD	SSE	*Double-ALU/Reg. in each PE*
Parallel-XL [83]	2018	I/C	SCMD/MCMD	DDD	*Intel Cilk & work stealing*
dMT-CGRA [84]	2018	I/C	MCMD	DDD	*Based on SGMF*

*I–imperative programming model, D–declarative programming model, C–parallel/concurrent (imperative) programming model, "*transparent*" means that CGRA-related programming is not required, "-" means that programming is not mentioned in that work.
**SSE–static-scheduling sequential-execution, SSD–static-scheduling static-dataflow-execution, DSD–dynamic-scheduling static-dataflow-execution, DDD–dynamic-scheduling dynamic-dataflow-execution.

Required model features

- Programming: I/C (Imperative, concurrent)
- Computation: MCMD (multi-config, multi-data)
- Execution: DDD (dynamic-scheduling, dynamic-dataflow)

Only recent designs have the desired features

- [83] is an FPGA prototype and simulations based
- [84] focuses on the narrow aspect of inter-thread communication (point-to-point), extensions to CUDA

Great need, many open challenges

- No efficient programming paradigm for CGRAs
- More complicated Hw than CPU due to 2D scheduling
- High-level abstraction provides coarse-grain parallelism, which is insufficient to fulfill the hardware potential
- Performance depends on applications; the need for application oriented extensions to the programming model
- Reconfig. speed down to pipeline level (10's of cycles)

L. Liu, et al., "A Survey of Coarse-Grained Reconfigurable Architecture and Design: Taxonomy, Challenges, and Applications," ACM Computing Surveys, Oct. 2019.

Partial FPGA Reconfig. Small-Size & Very Slow

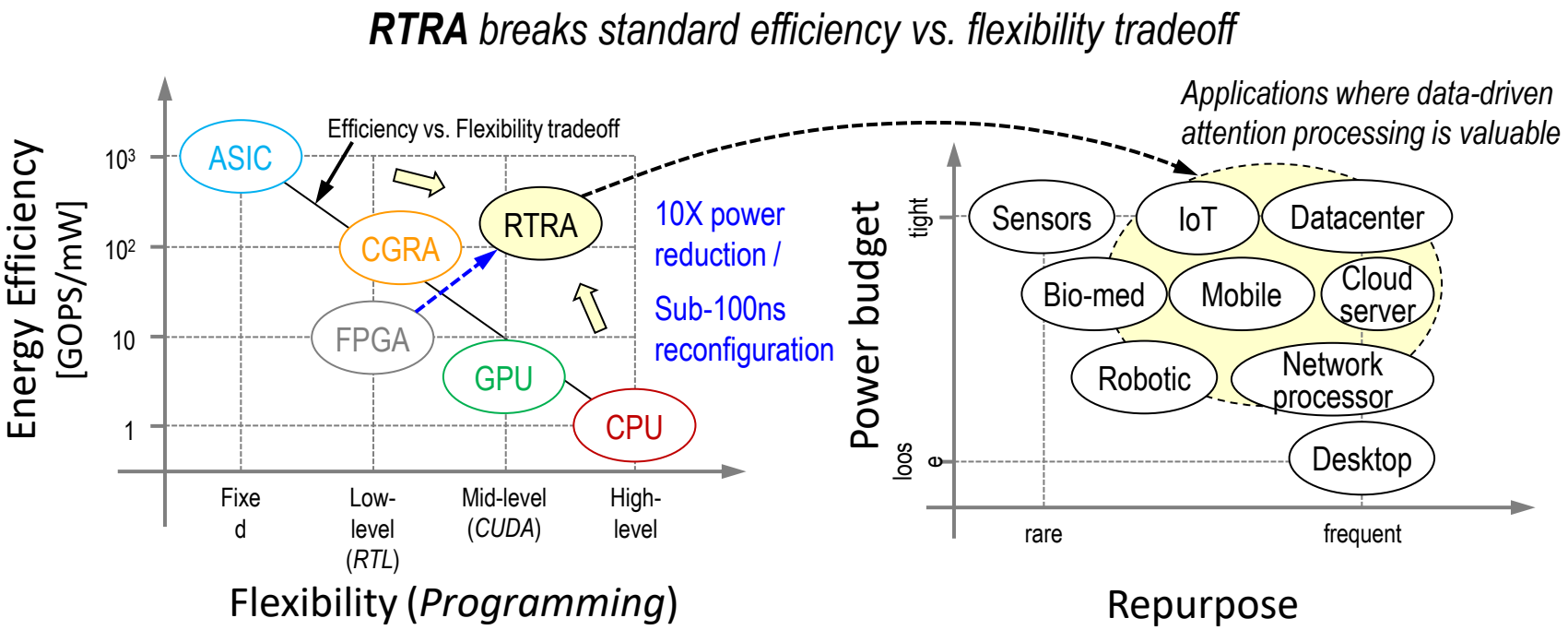
- **FPGA time to dynamic partial reconfigure depends on [1]:**
  - The size of the config. bit-stream ($BitStr_{size}$) – usually in KB
  - The reconfig. path throughput ($RP_{throughput}$) – usually in MB/s

$$T_{dyn-rec} = \frac{BitStr_{size}}{RP_{throughput}}$$

- **Dynamic partial reconfiguration controllers go up to 400MB/s [2]**
- **Usually, a large number of Clk cycles is required for a small amount of logic**
  - 130k Clk cycles to reconfigure 1.5k slices of logic [3] | 0.4ms @ 300MHz Clk
- **An SDR pipeline on a Zynq FPGA uses 3.2k slices of logic, 4-region partition**
  - Largest partial bit-stream size for a region is 324 KB [4]
  - Worst execution time for dynamic partial reconfig. of this region is 1.08ms

[1] G. Valente et al., "Dynamic partial reconfiguration profitability for realtime systems," IEEE Embedded Systems Letters, pp. 1–1, 2020.
[2] S. D. Carlo, P. Prinetto, P. Trotta, and J. Andersson, "A portable open-source controller for safe dynamic partial reconfiguration on Xilinx FPGAs," in Proc. of the 25th International Conference on Field Programmable Logic and Applications (FPL), 2015, pp. 1–4.
[3] L. Pezzarossa, A. T. Kristensen, M. Schoeberl and J. Sparso, "Can real-time systems benefit from dynamic partial reconfiguration?," 2017 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC), Linkoping, 2017, pp. 1-6, doi: 10.1109/NORCHIP.2017.8124984.
[4] A. Kamaleldin et al., "A reconfigurable hardware platform implementation for software defined radio using dynamic partial reconfiguration on Xilinx Zynq FPGA," 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, 2017, pp. 1540-1543, doi: 10.1109/MWSCAS.2017.8053229.

Runtime Reconfig. for Data-Driven Processing

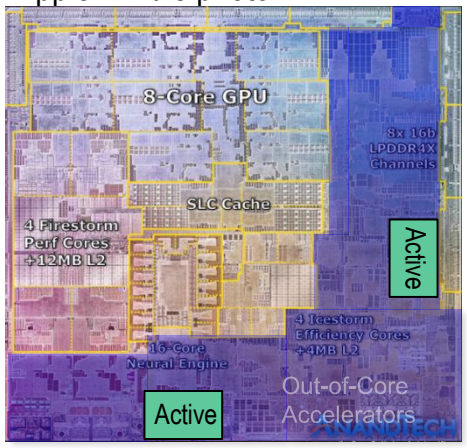


- **Opportunistically repurpose unutilized processor arrays**
  - Multi-step compilation (Sw + Hw) avoids complete program recompile
  - Requires array's network symmetry (for polygon translation/rotation/flip)
  - Support for Sw compilation from Python/C++ base

An Alternative to Accelerators?

- **~7% of *entire* SoC area is active, TSMC 45nm node [*]**
- **Depending on domain specialization, RTRA can be within 2x-10x in area and power vs accelerator**
  - **Note:** system/platform power will not be 2-10x higher; much less (~30-50%)
    - e.g. iPad battery life with H.264 on CPU (3 hours) vs accelerator (10 hours), a 3x system impact with a 1,000x accelerator gain
- **Feasible for new and/or evolving architectures, SDR, etc.**

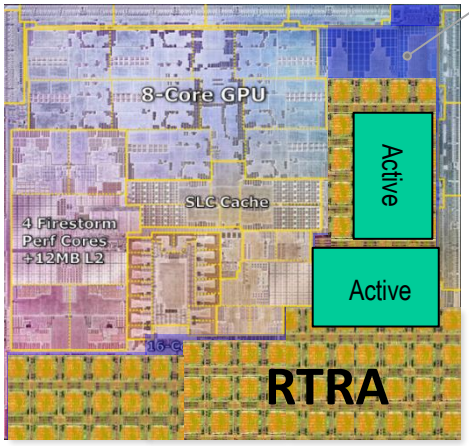
Apple M1 die photo



Replace "UnCore" with RTRA

No area increase (or even saving)

~2 active programs
>10x area overhead



potentially saved area

~2 active programs
Can accommodate more

Algorithm updates w/o chip respin

[*] G. Venkatesh, et al., ACM SIGARCH Computer Architecture News, Volume 38Issue 1 March 2010 pp 205–218 https://doi.org/10.1145/1735970.1736044