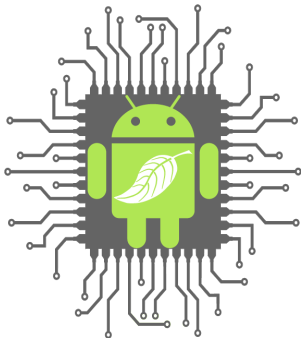
GreenDroid: An Architecture for the Dark Silicon Age

Nathan Goulding-Hotta, Jack Sampson, Qiaoshi Zheng,
Vikram Bhatt, Joe Auricchio,
Steven Swanson, ***Michael Bedford Taylor***



University of California, San Diego

This Talk

The Dark Silicon Problem

How to use Dark Silicon to improve energy efficiency
(Conservation Cores)

The GreenDroid Mobile Application Processor

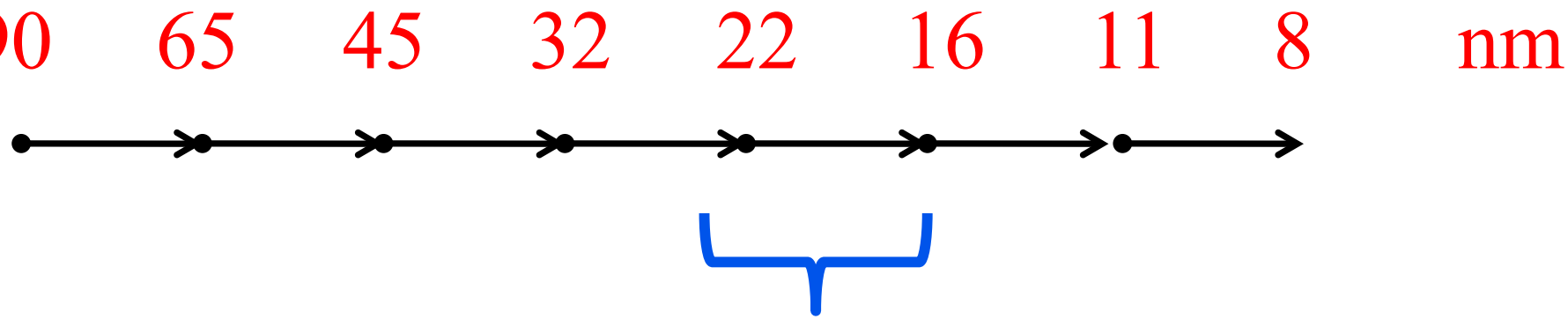
Where does dark silicon come from? And how dark is it going to be?

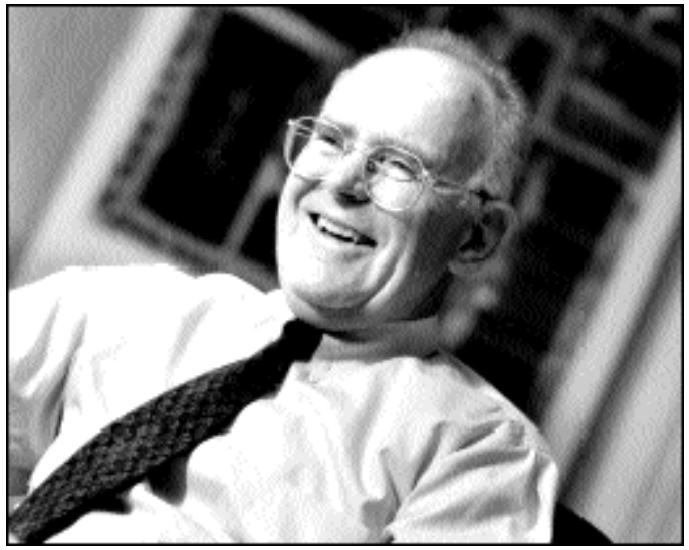
The Utilization Wall:

With each successive process generation, the percentage of a chip that can switch at full frequency drops exponentially due to power constraints.

[Venkatesh, ASPLOS ‘10]

Scaling 101: Moore's Law





$$S = \frac{22}{16} = \sim 1.4x$$

Scaling 101:
Transistors scale as S^2

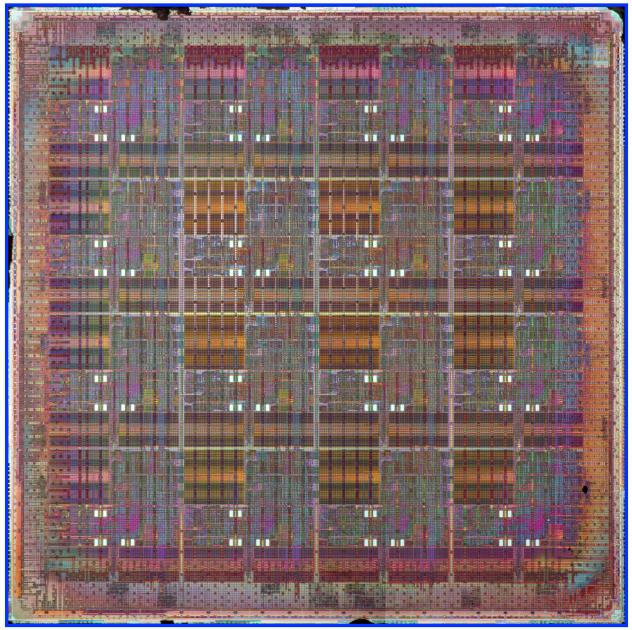
180 nm

16 cores

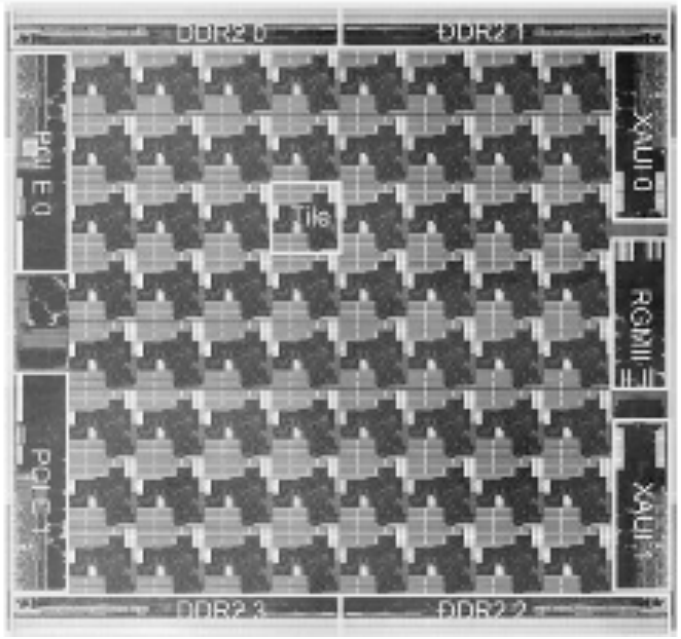
S = 2x
Transistors = 4x

90 nm

64 cores



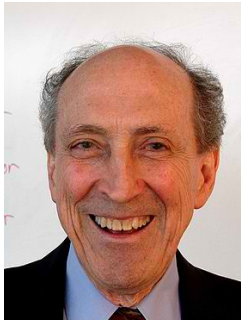
MIT Raw



Tilera TILE64

Advanced Scaling:

Dennard: "Computing Capabilities Scale by $S^3 = 2.8x$"



If S=1.4x …

S^3

S^2

S

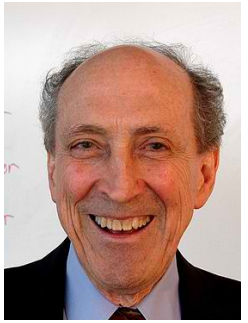
1

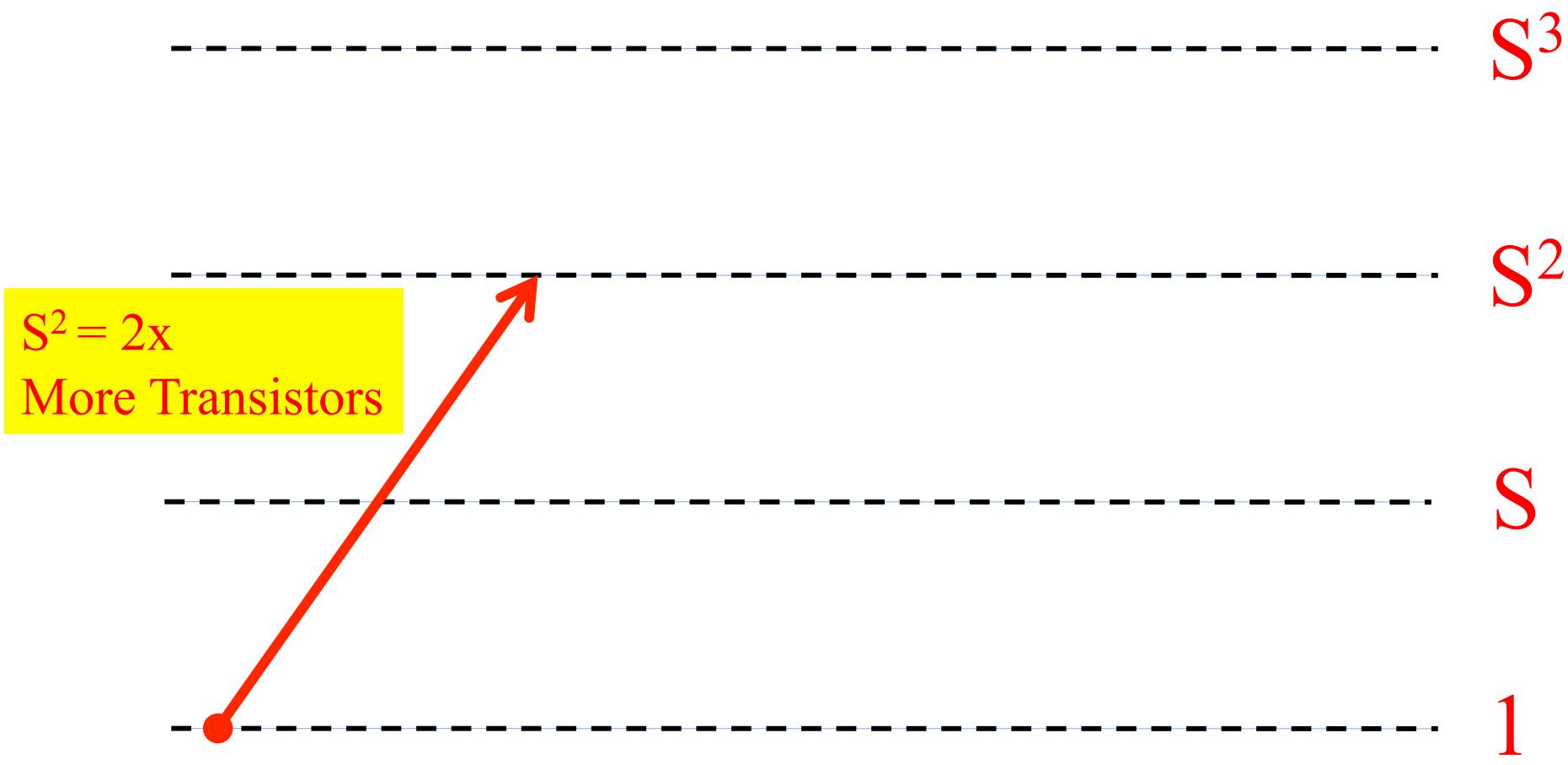
Design of Ion-Implanted MOSFETs with Very Small Dimensions
Dennard et al, 1974

Advanced Scaling:

Dennard: "Computing Capabilities Scale by $S^3 = 2.8x$"

If S=1.4x …

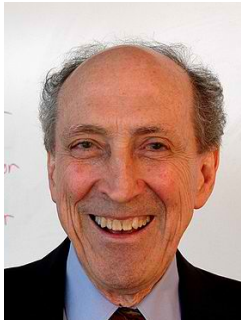


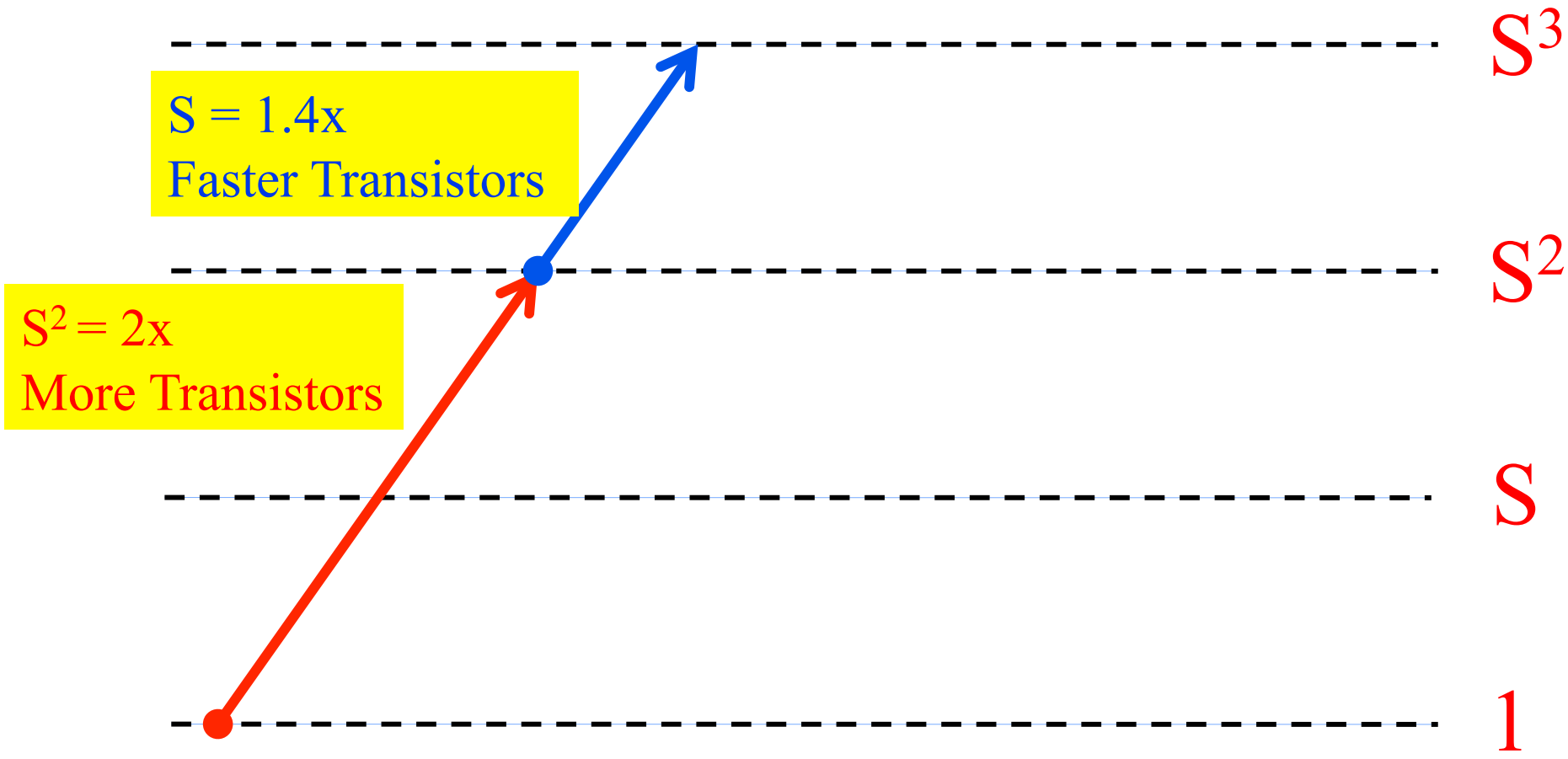


Advanced Scaling:

Dennard: "Computing Capabilities Scale by $S^3 = 2.8x$"

If S=1.4x …

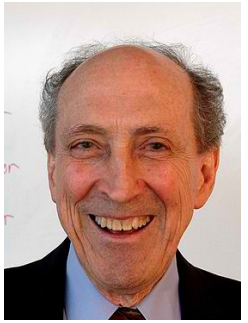


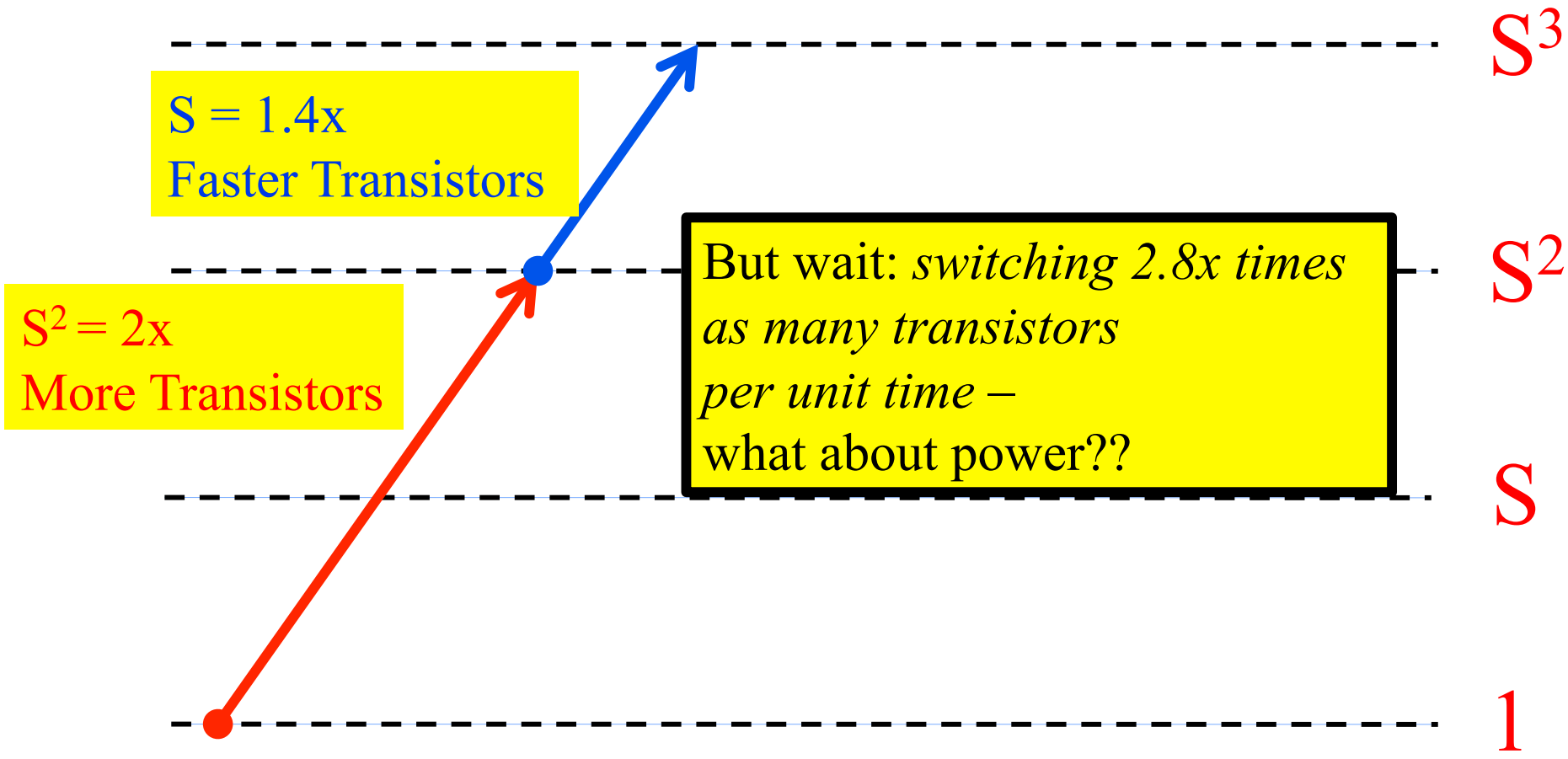


Advanced Scaling:

Dennard: "Computing Capabilities Scale by $S^3 = 2.8x$"

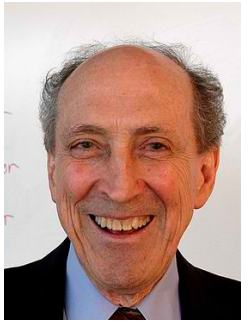
If S=1.4x …

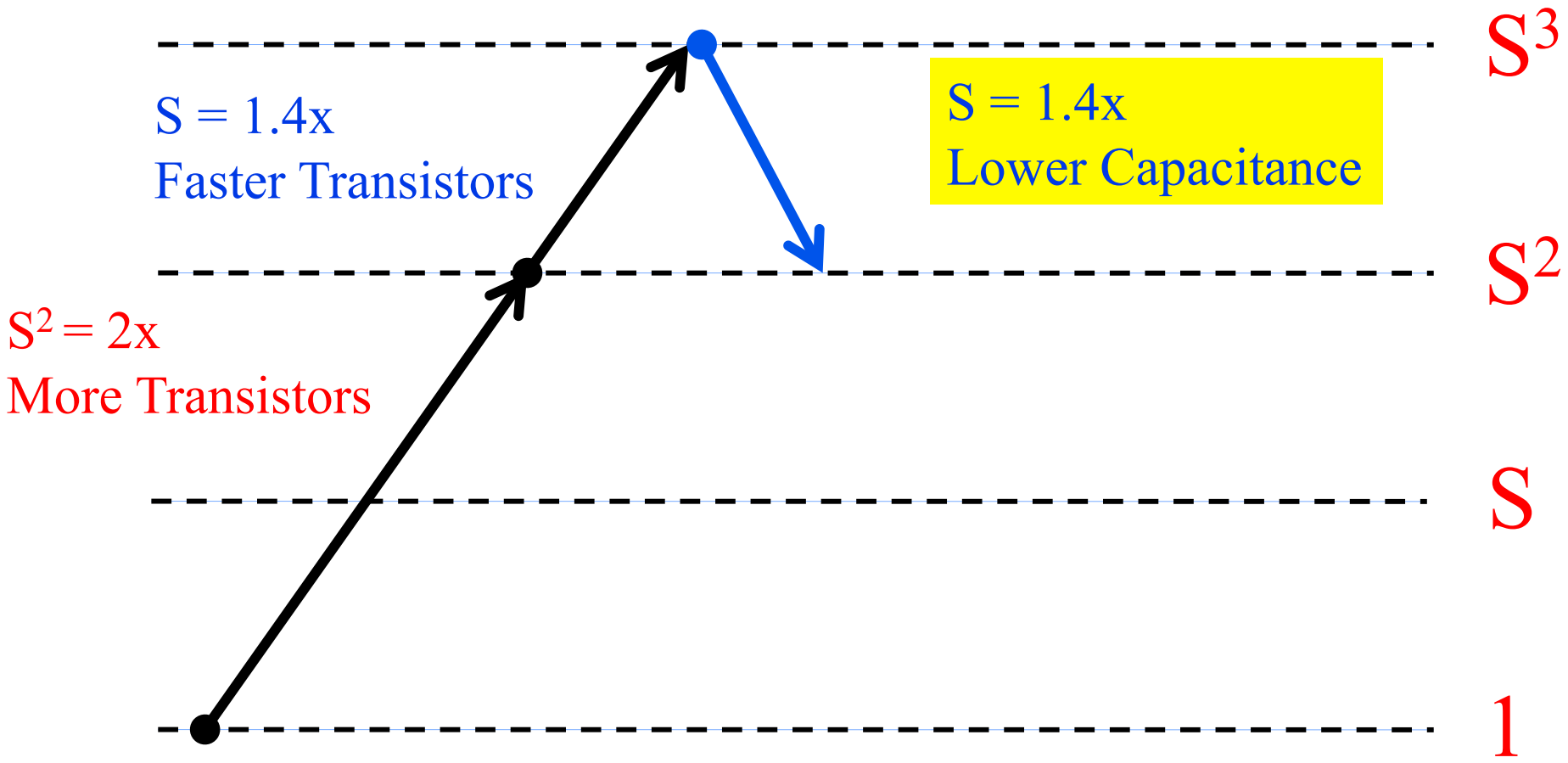




Dennard:

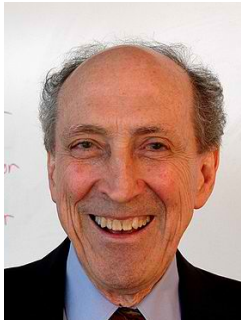
"We can keep power consumption constant"

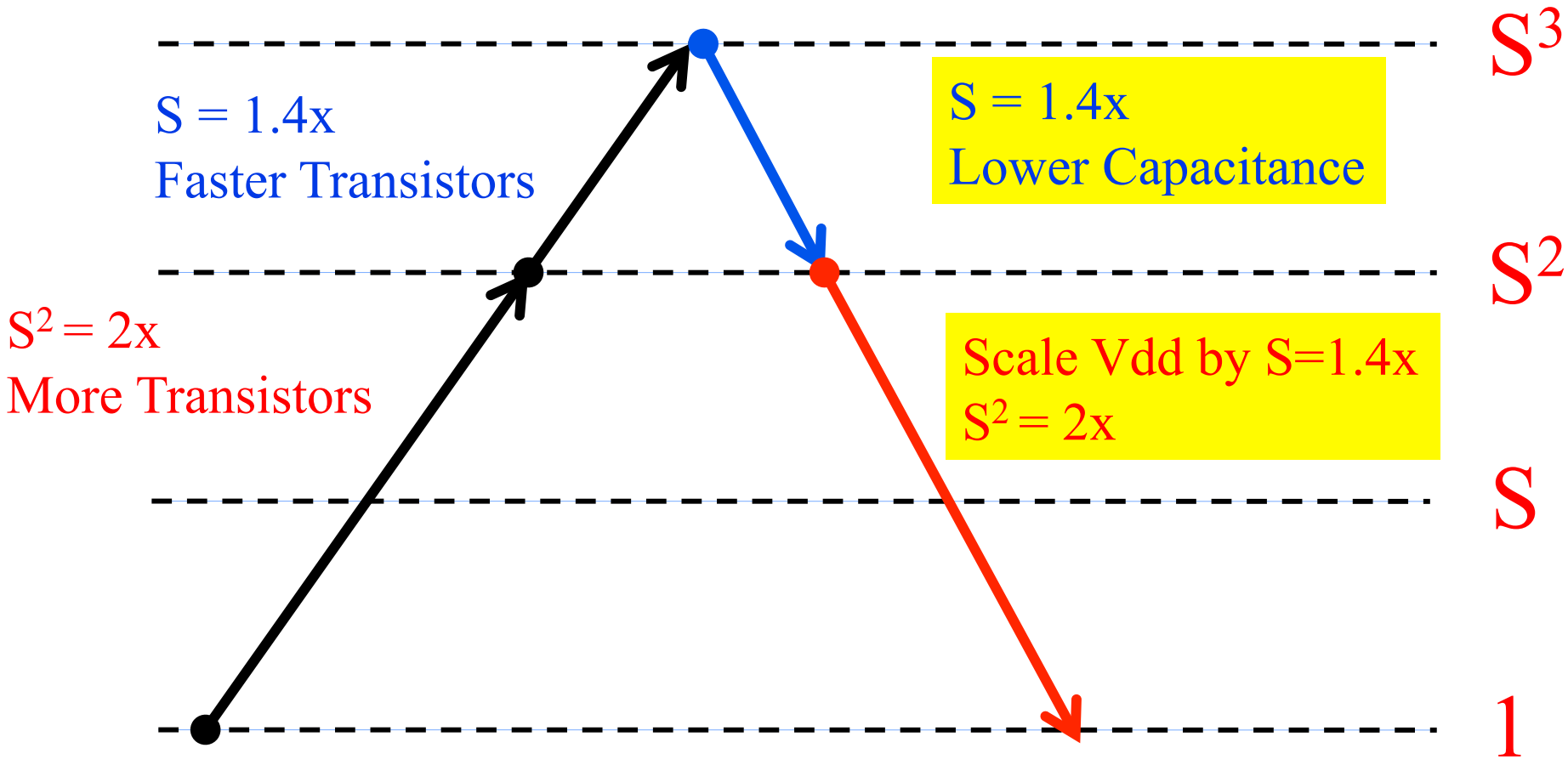




Dennard:

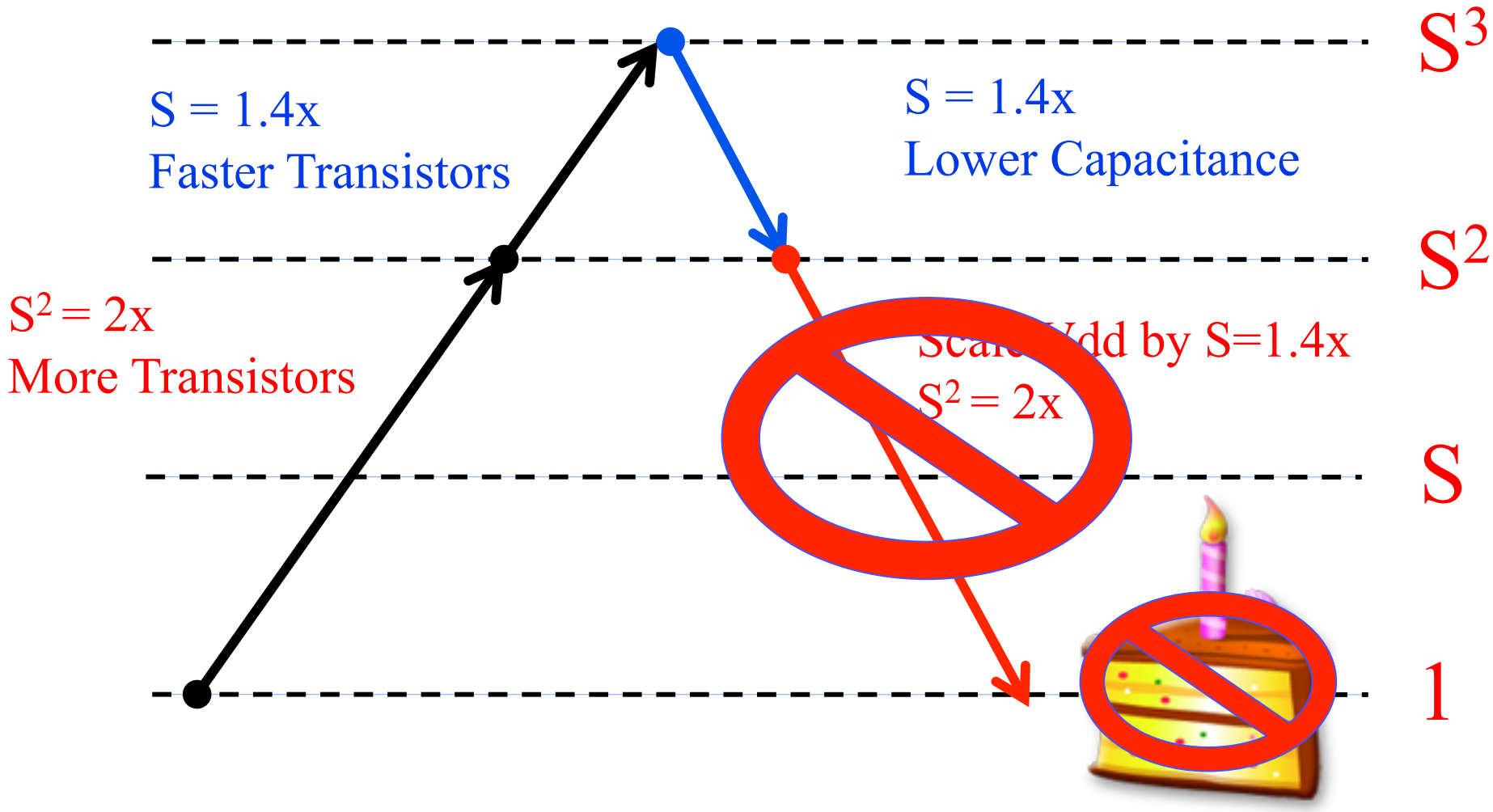
"We can keep power consumption constant"



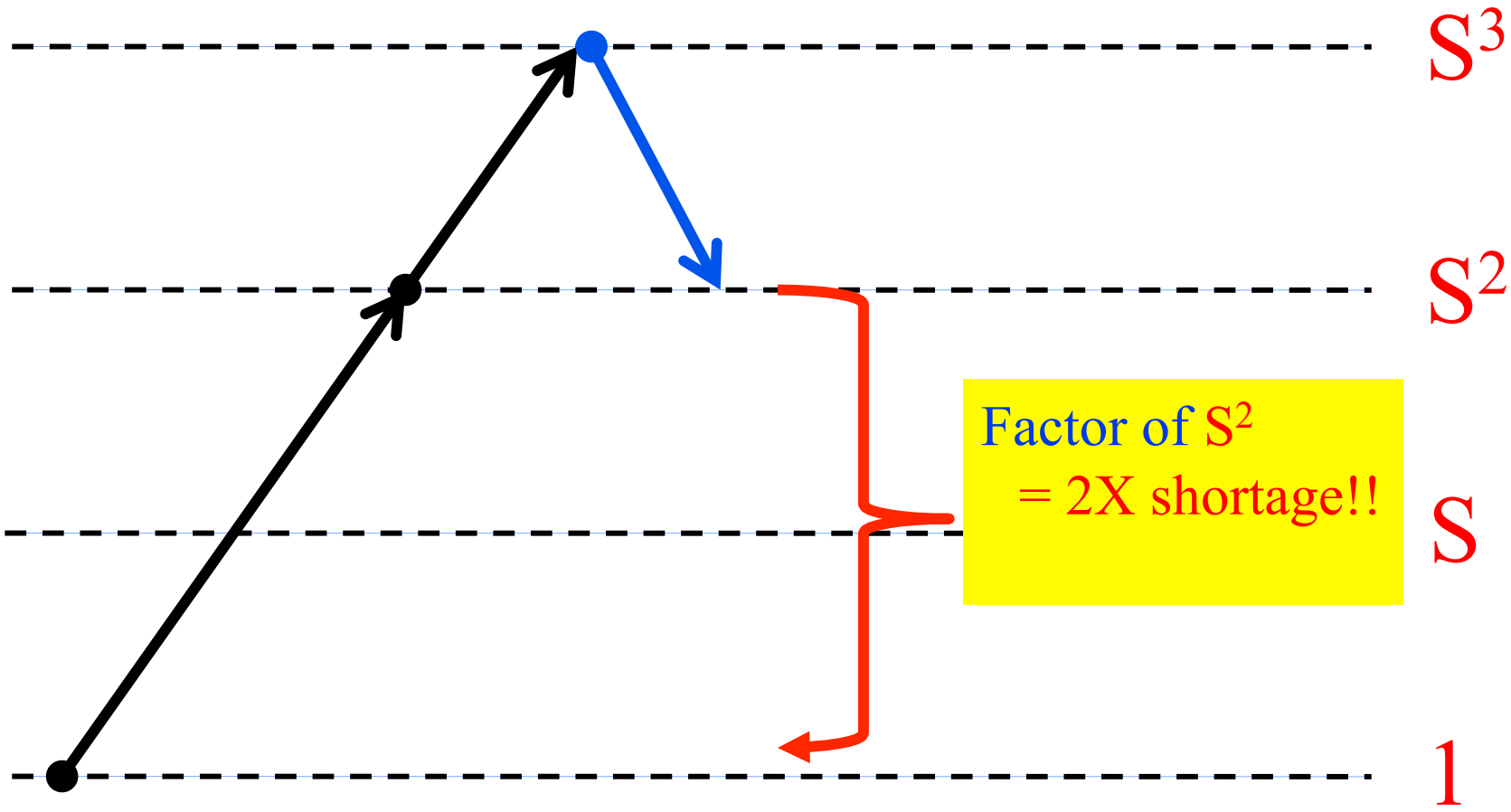


Fast forward to 2005:

Threshold Scaling Problems due to Leakage Prevents Us From Scaling Voltage



Full Chip, Full Frequency Power Dissipation Is increasing exponentially by 2x with every process generation

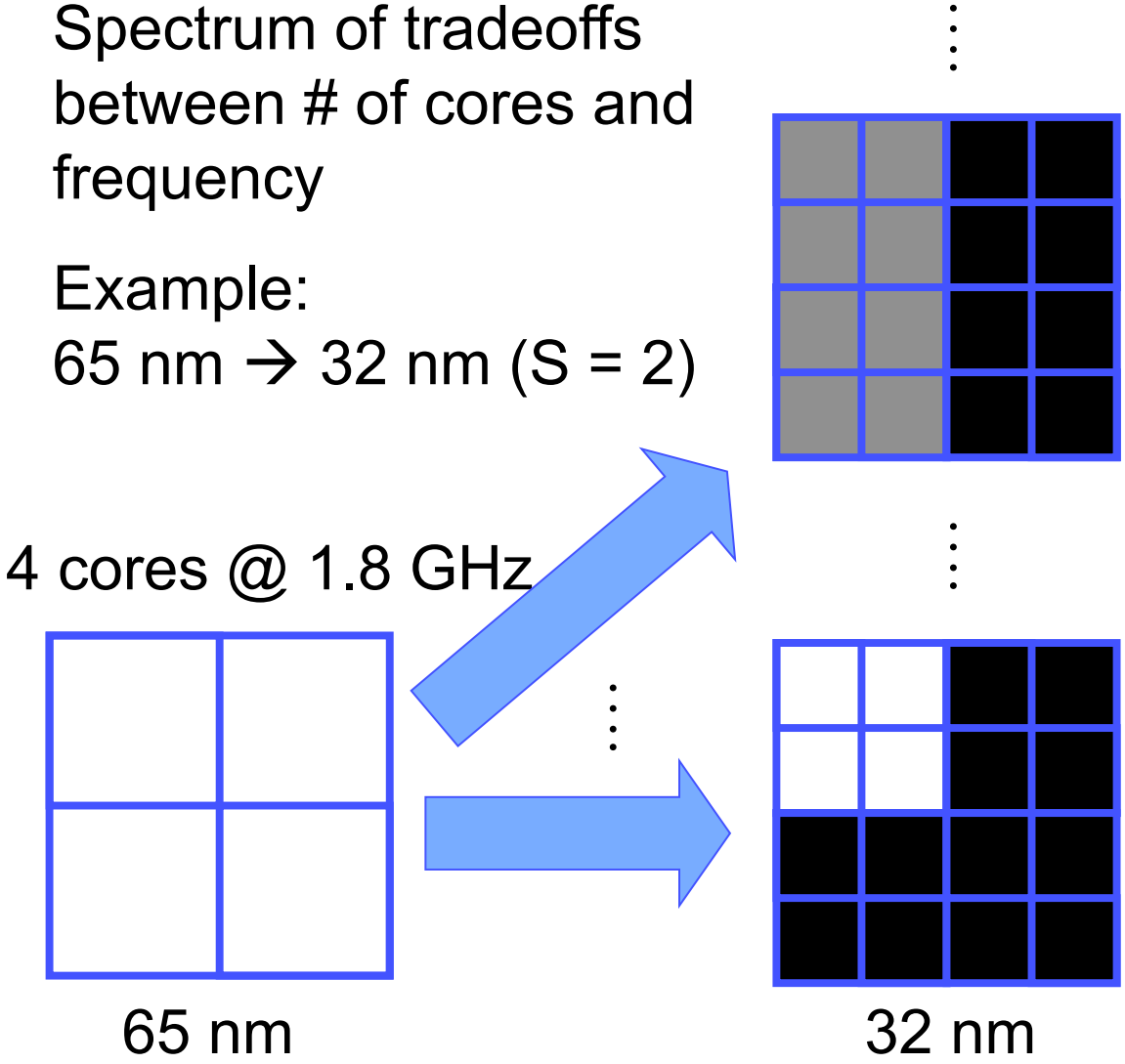


[ASPLOS 2010, Venkatesh]

Multicore has a lot of Dark Silicon in its Future

Spectrum of tradeoffs between # of cores and frequency

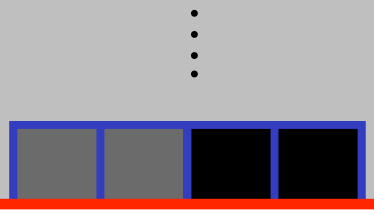
Example:
65 nm → 32 nm (S = 2)



[Hotchips 2010]

Multicore has a lot of Dark Silicon in its Future

Spectrum of tradeoffs between # of cores and frequency



2x4 cores @ 1.8 GHz

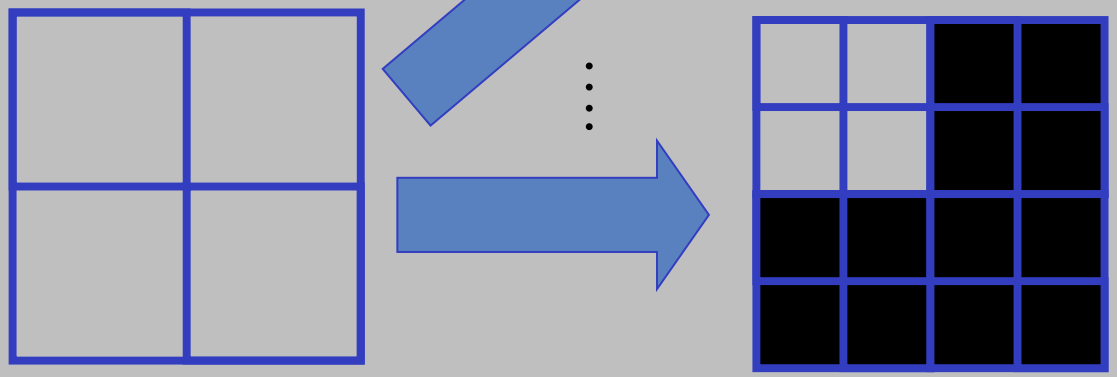
m)

Exa

65 r

The utilization wall will change the way everyone builds chips.

4 cores @ 1.8 GHz



4 cores @ 2x1.8 GHz
(12 cores dark)

65 nm

32 nm

[Hotchips 2010]

This Talk

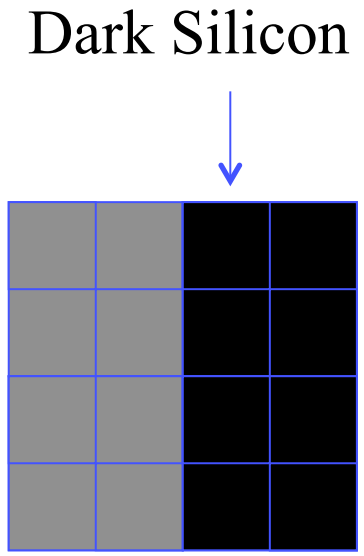
The Dark Silicon Problem

How to use Dark Silicon to improve energy efficiency (Conservation Cores)

The GreenDroid Mobile Application Processor

What do we do with Dark Silicon?

- Idea: Leverage dark silicon to "fight" the utilization wall



- Insights:
  - Power is now more expensive than area
  - Specialized logic can improve energy efficiency by 10-1000x
- Our Approach:
  - Fill dark silicon with *Conservation Cores*, or c-cores, which are specialized energy-saving coprocessors that save energy on common apps
  - Execution jumps from c-core to c-core
  - Power-gate c-cores that are not currently in use
- *Conservation Cores provide an architectural way to trade dark area for an effective increase in power budget!*

C-cores compared to Accelerators

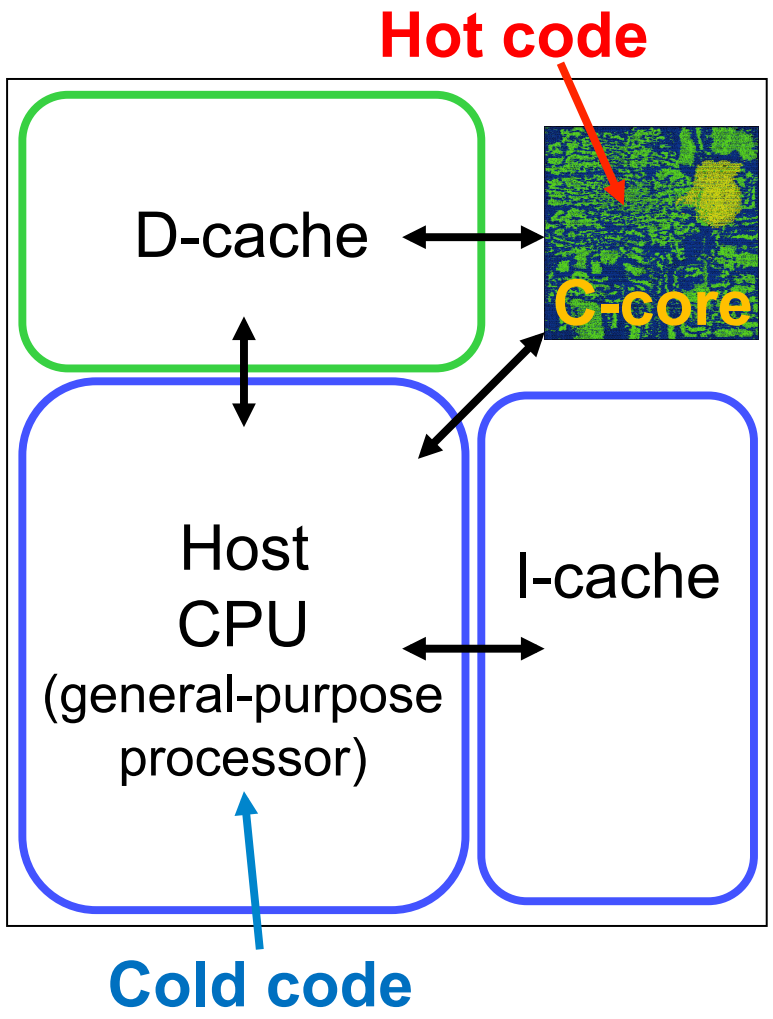
- Although they also tend to be more energy-efficient, accelerators focus on attaining speedup and target codes which are relatively easy to parallelize: medium or high parallelism, high regularity and a relatively small code base.
- Many applications use irregular, non-parallelizable code:

- Amdahl's Law: Overall energy efficiency *depends on the fraction of the total code execution that is optimized!*
- To gain large energy savings through specialization:
  - We need to target irregular code as well as regular code, and
  - We need many, many such coprocessors to get high coverage
    - need to solve both design effort and architectural scalability problems

Conservation Cores (C-cores)

"Conservation Cores: Reducing the Energy of Mature Computations," Venkatesh et al., ASPLOS '10

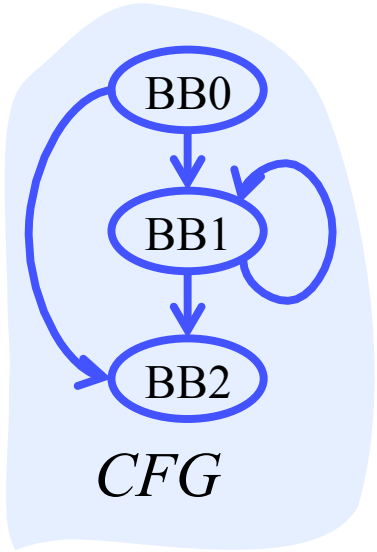
- Specialized coprocessors for reducing energy in irregular code
  - Hot code implemented by c-cores, cold code runs on host CPU;
  - C-cores use up to 18x less energy
  - Shared D-cache → Coherent Memory
  - ***Patching support in hardware***
- Fully-automated toolchain
  - ***No "deep" analysis or transformations required***
  - C-cores automatically generated from hot program regions
  - Design-time scalable
    - Emphasize Quantity over Quality!
    - Simple conversion into HW buys us big gains, no need for heroic compiler efforts.



```
for (i=0; i<N; i++) {
  x = A[i];
  y = B[i];
  C[x] = D[y] + x+y+x*y;
}
```

C-core Generation

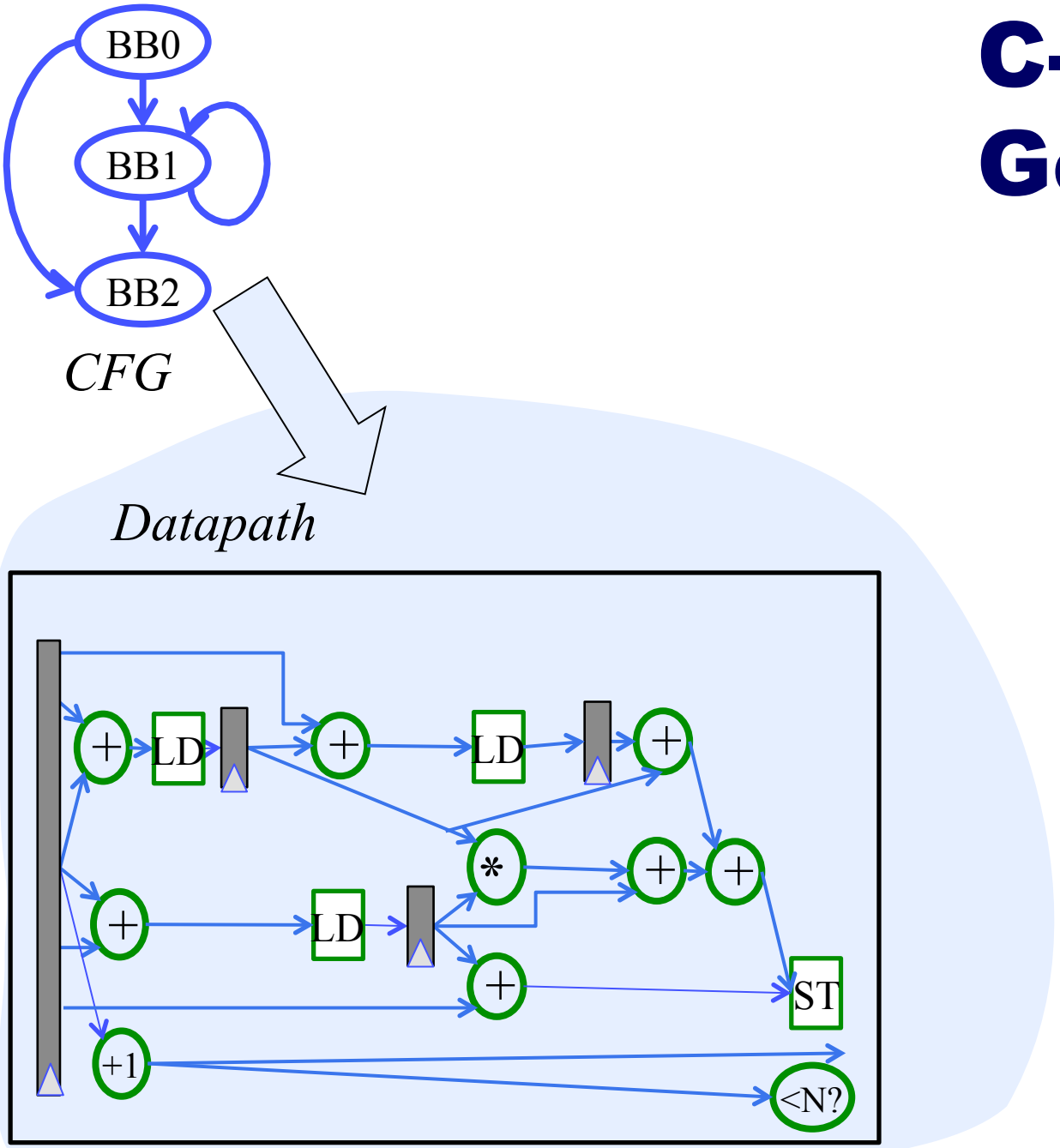
Start with ordinary C code. Irregular or regular is fine. Arbitrary control flow, arbitrary memory access patterns and complex data structures are supported.



C-core Generation

Build a CFG; run ordinary compiler optimizations.

C-core Generation

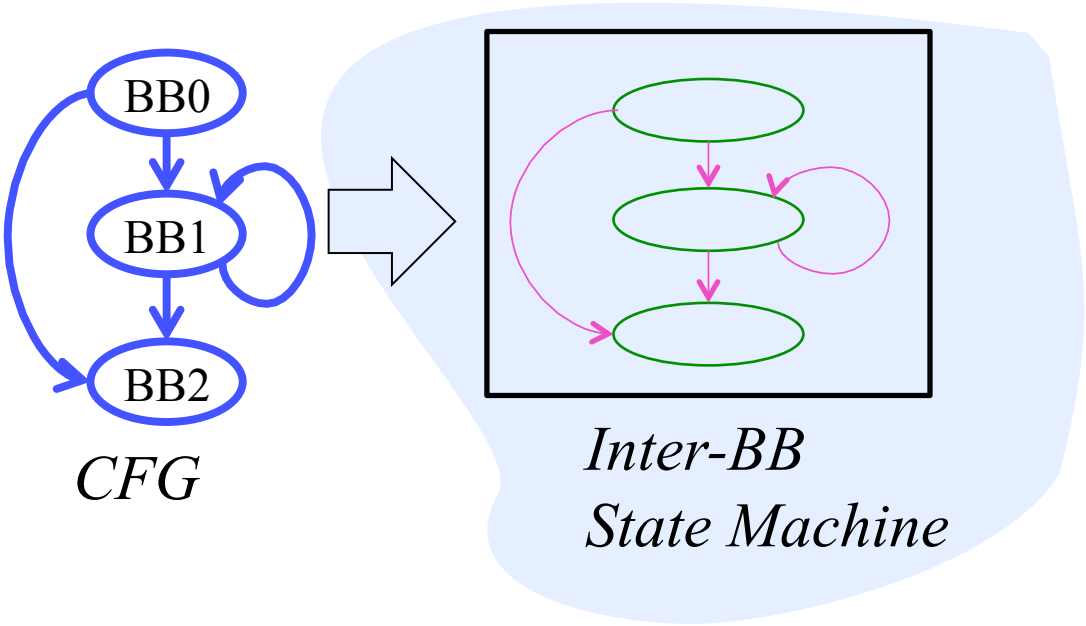


Each BB becomes a datapath; each operator turned into HW equivalent.

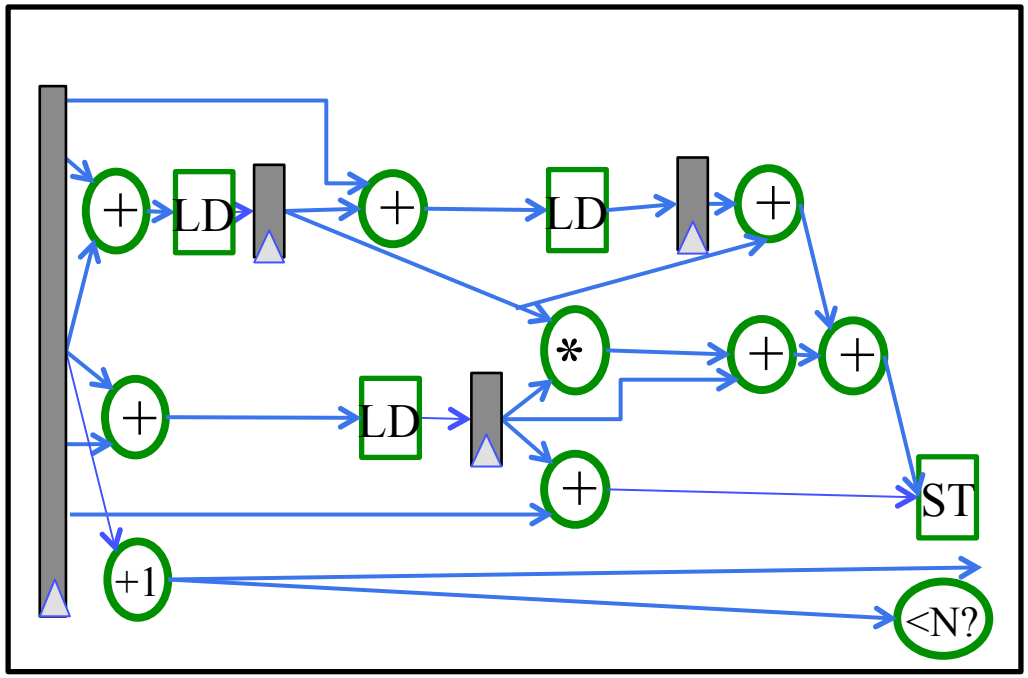
Memory ops mux'd into L1 cache.

Multiplier and FPUs may or may not be shared.

C-core Generation

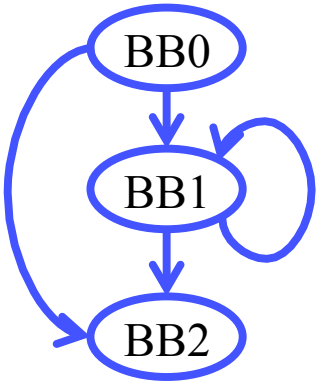


Datapath

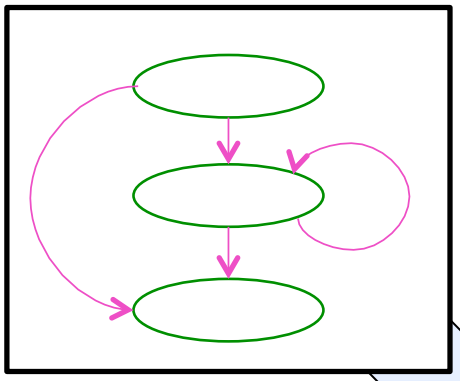


Create a state machine that determines which BB (datapath) is next.

C-core Generation



CFG

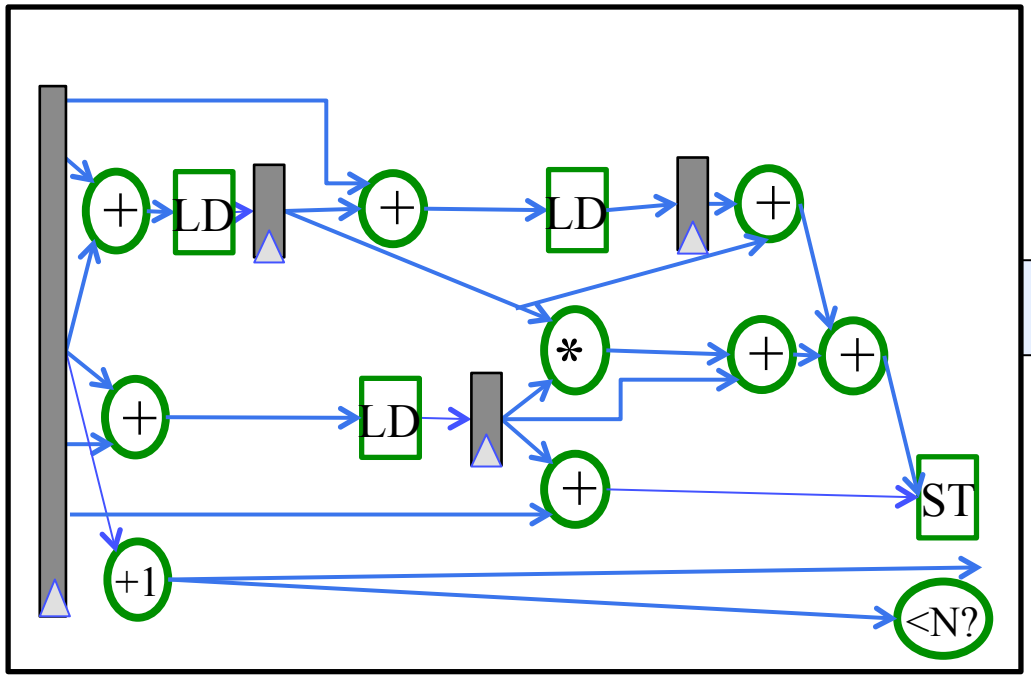


Inter-BB State Machine

.V

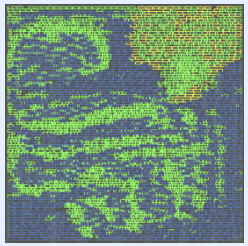
Via verilog, run through standard CAD flow.

Datapath



.V

Synopsys IC Compiler, P&R, CTS



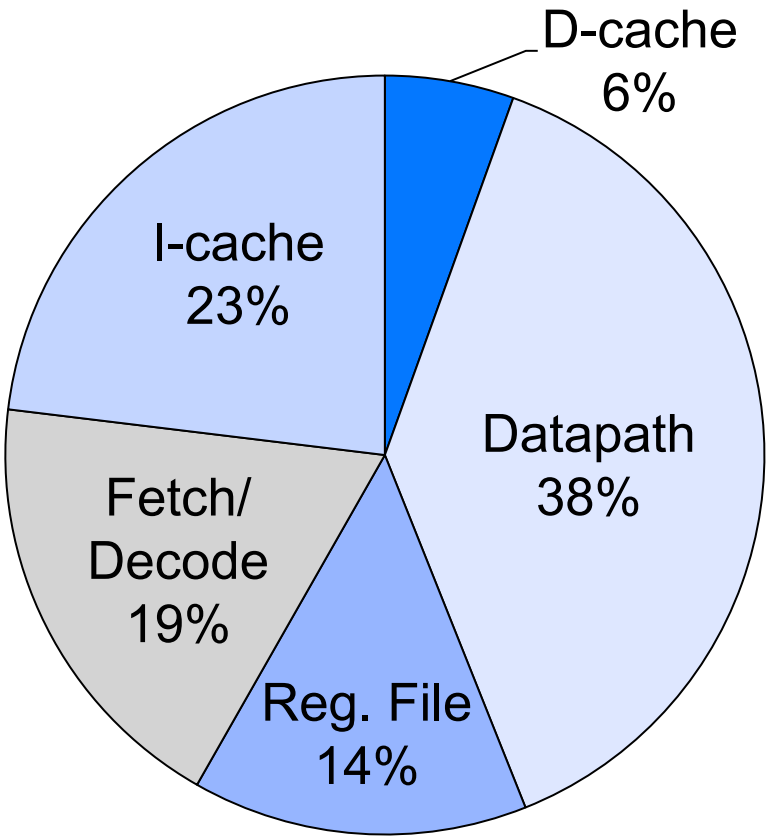
0.01 mm² in 45 nm TSMC runs at 1.4 GHz

C-cores Experimental Data

- We automatically built 21 c-cores for 9 "hard" applications like Spec, Mediabench, etc.
  - 45 nm TSMC
  - Vary in size from 0.10 to 0.25 mm^2

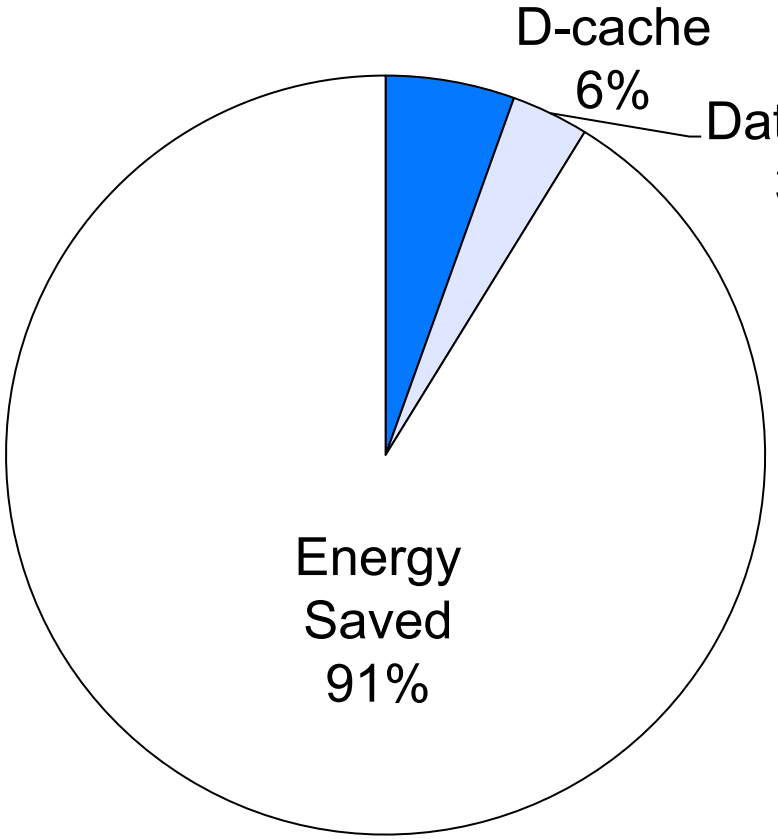
Application	# C-cores	Area (mm^2)
bzip2	1	0.18
cjpeg	3	0.18
djpeg	3	0.21
mcf	3	0.17
radix	1	0.10
sat solver	2	0.20
twolf	6	0.25
viterbi	1	0.12
vpr	1	0.24

Typical Energy Savings



RISC baseline
91 pJ/instr.

← ~11x →



C-cores
8 pJ/instr.

This Talk

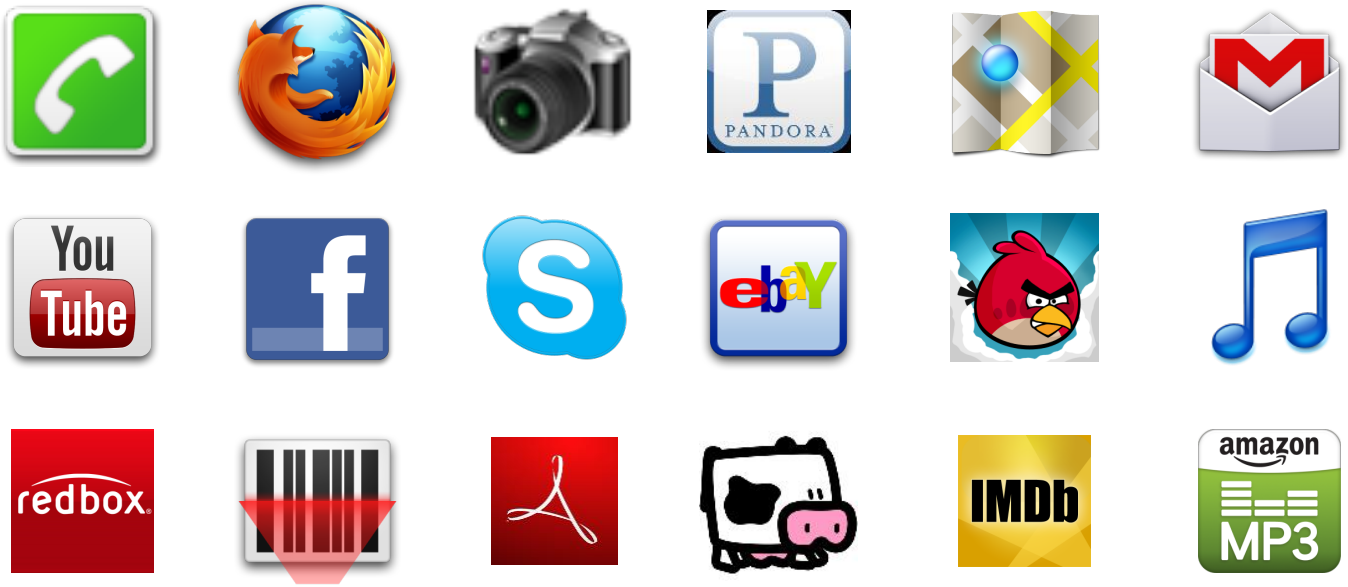
The Dark Silicon Problem

How to use Dark Silicon to improve energy efficiency

The GreenDroid Mobile Application Processor

Examing today's smartphone

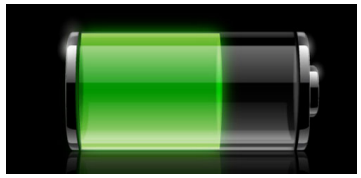
- Lots of irregular code (desktop → mobile)



- **Utilization Wall problem is even worse on mobile!**
  - *Active Power budget is set by*

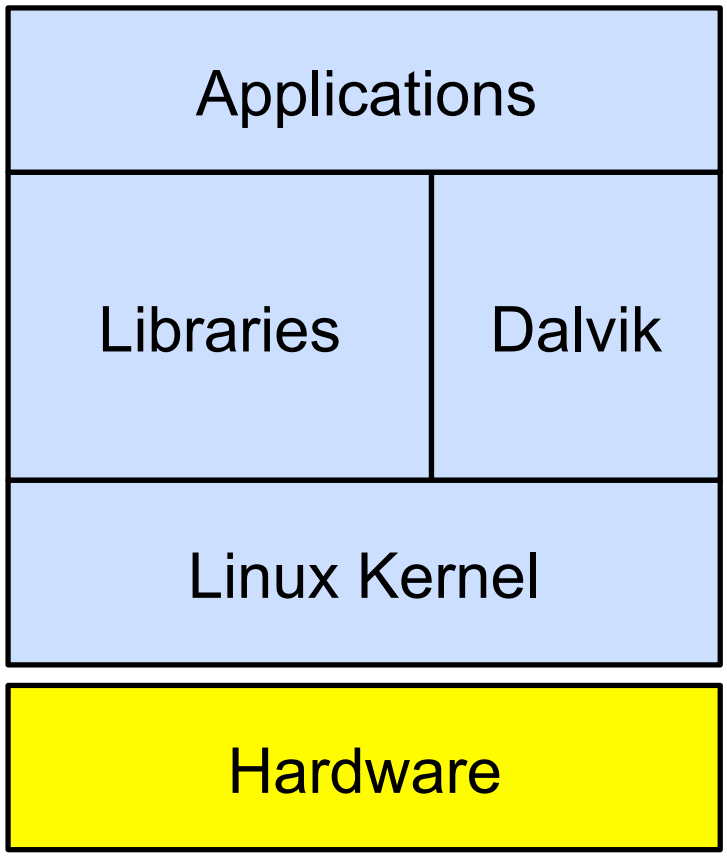
    (battery capacity) / (# hrs active use between recharges)

    rather than thermal design point.



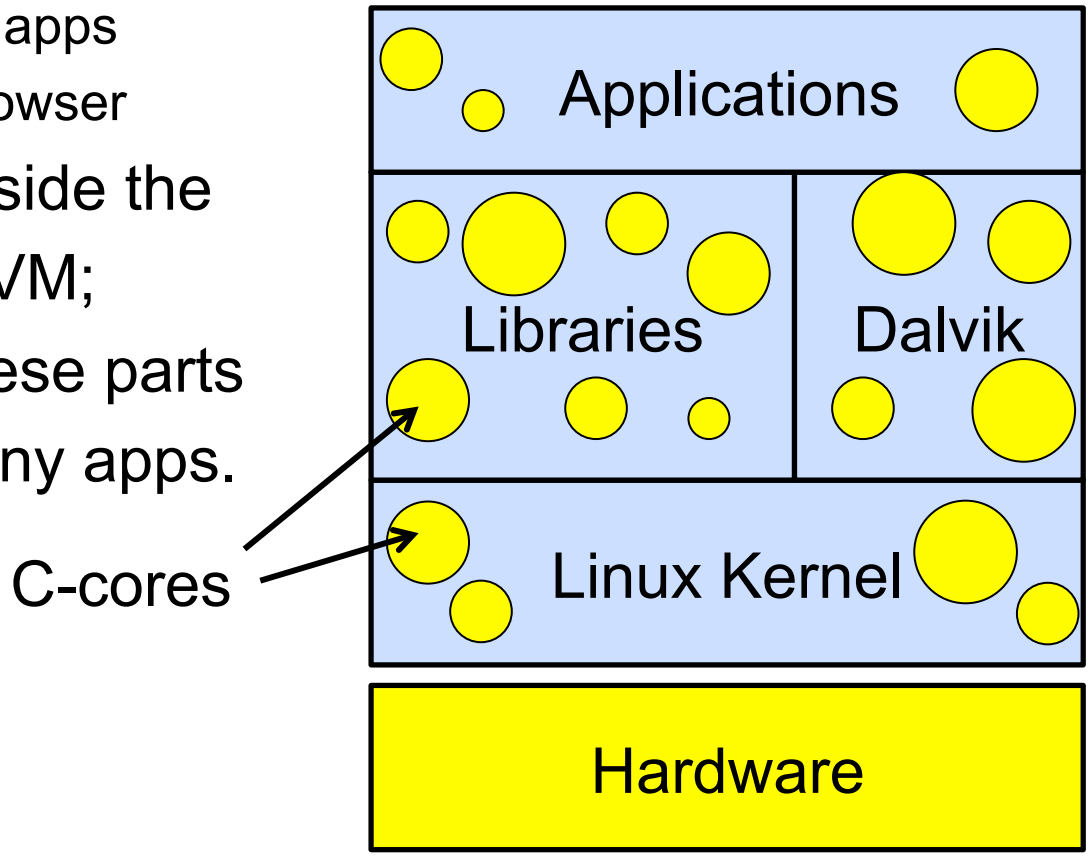
Android™

- Google's OS + app. environment for mobile devices
- Java applications run on the Dalvik virtual machine
- Apps share a set of libraries (libc, OpenGL, SQLite, etc.)



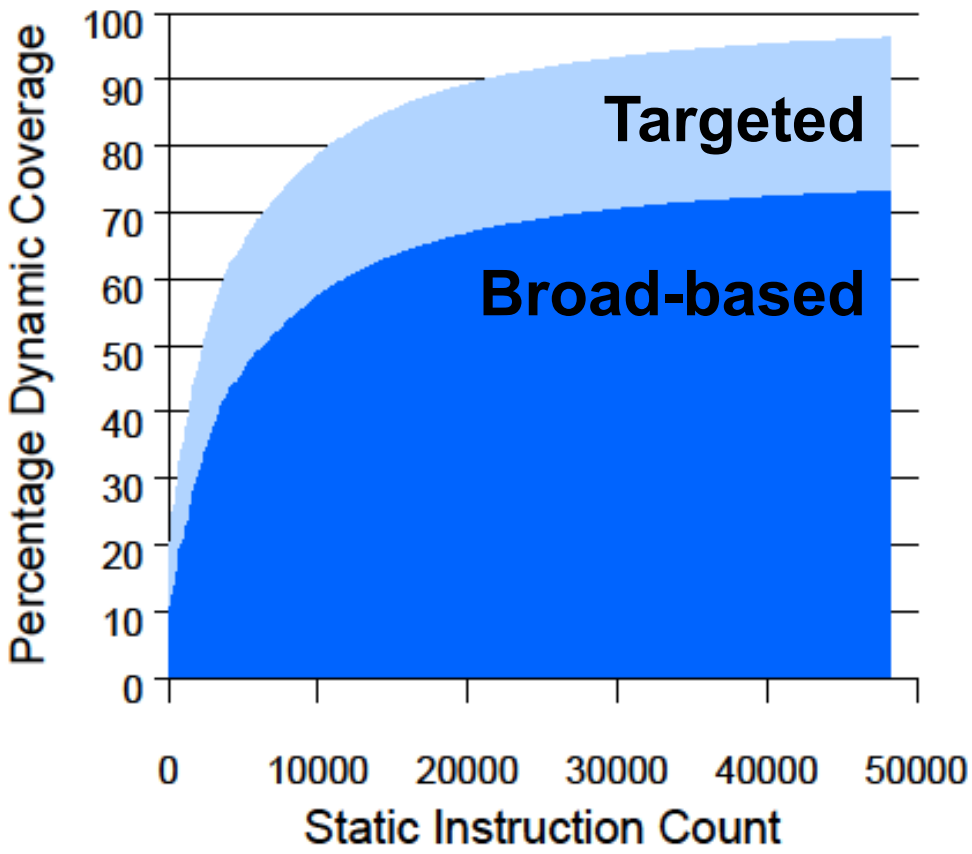
Applying C-cores to Android

- Android is well-suited for c-cores
  - Concentrated set of commonly used applications
    - 73% of time in top 51 apps
    - 33% of time just in browser
  - Lots of hot code is inside the Libraries and Dalvik VM; c-cores that target these parts are reused across many apps.



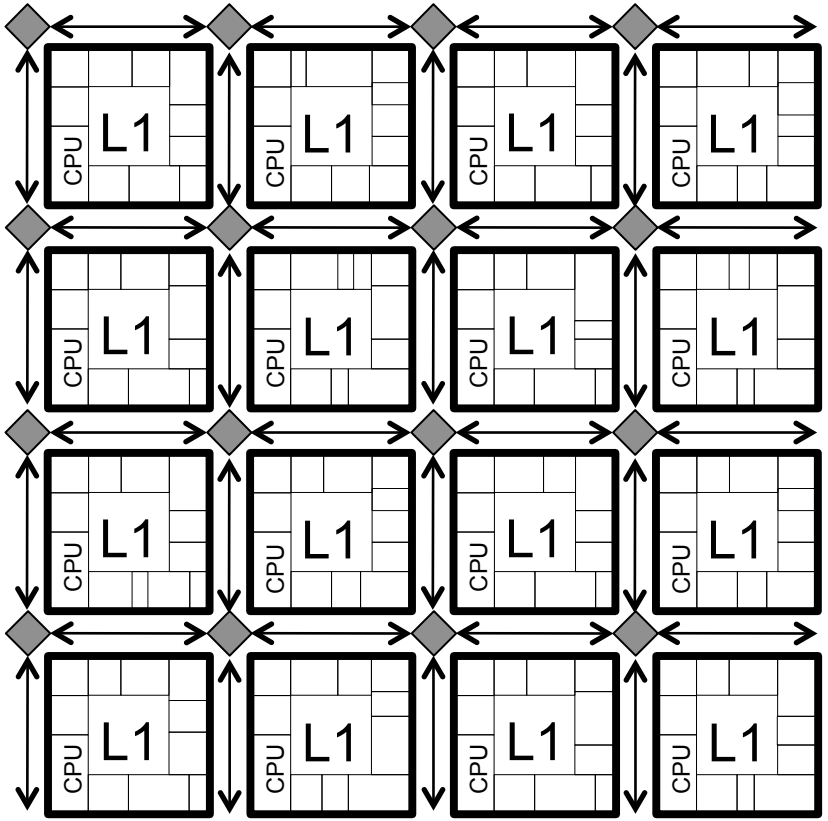
Android Workload Profile

- Profiled common Android apps to find the hot spots, including:
  - Google: Browser, Gallery, Mail, Maps, Music, Video
  - Pandora
  - Photoshop Mobile
  - Robo Defense game
- Broad-based c-cores
  - 72% code sharing
- Targeted c-cores
  - 95% coverage with just 43,000 static instructions (approx. 7 mm^2)



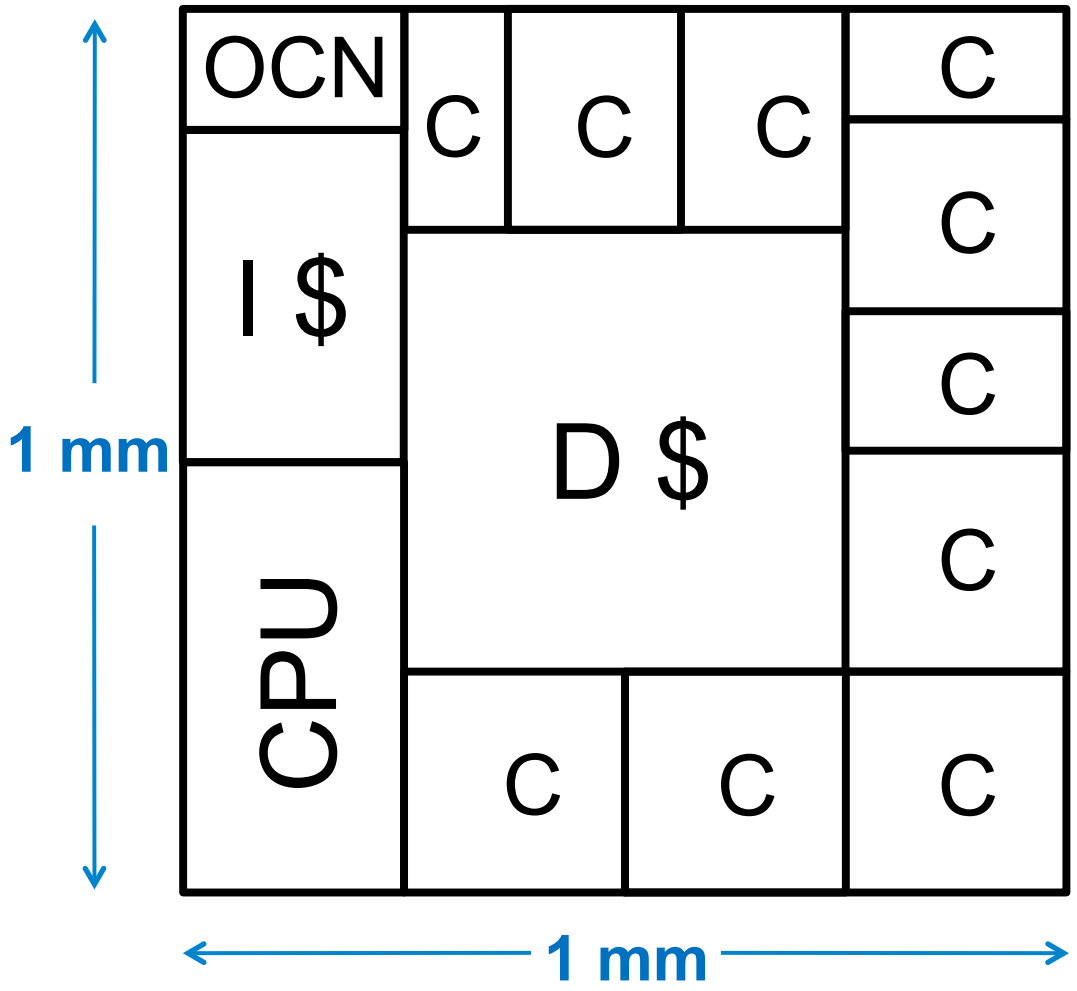
GreenDroid Tiled Architecture

- Scalable C-core fabric
- 16-tiles on a NOC
- Each tile contains
  - 6-10 Android c-cores (~125 total)
  - 32 KB D-cache (shared with CPU)
  - RISC “host” core
    - 32 bit, in-order, 7-stage pipeline
    - 16 KB I-cache
    - Single-precision FPU
  - On-chip network router



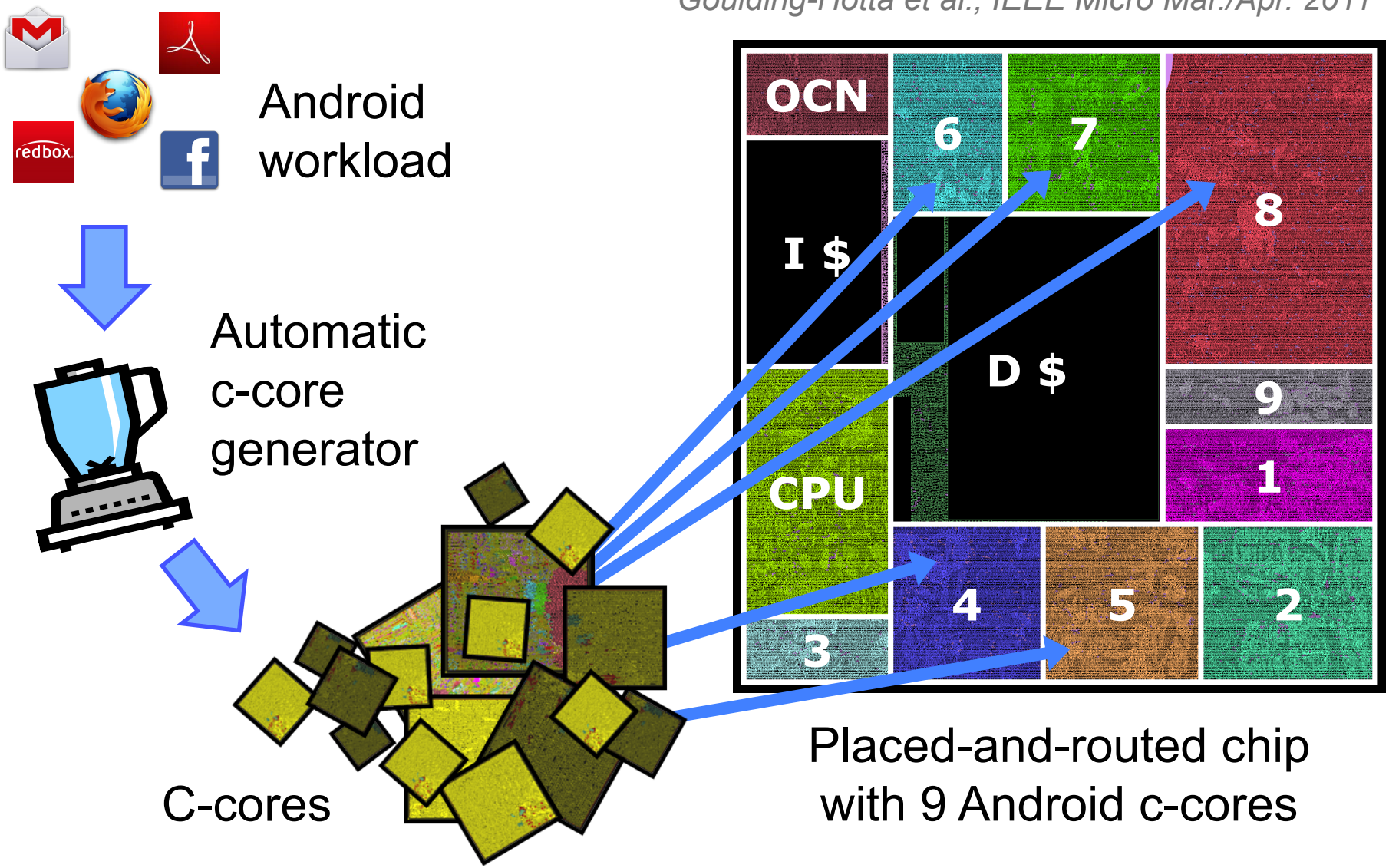
GreenDroid Tile Floorplan

- Norm to 45 nm:
  - 1.0 mm^2 per tile
  - 1.5 GHz
- 25% RISC core, I-cache, and on-chip network
- 25% D-cache
- 50% C-core “fill”



GreenDroid: Using c-cores to reduce energy in mobile application processors

"The GreenDroid Mobile Application Processor: An Architecture for Silicon's Dark Future,"
Goulding-Hotta et al., IEEE Micro Mar./Apr. 2011



GreenDroid: Projected Energy

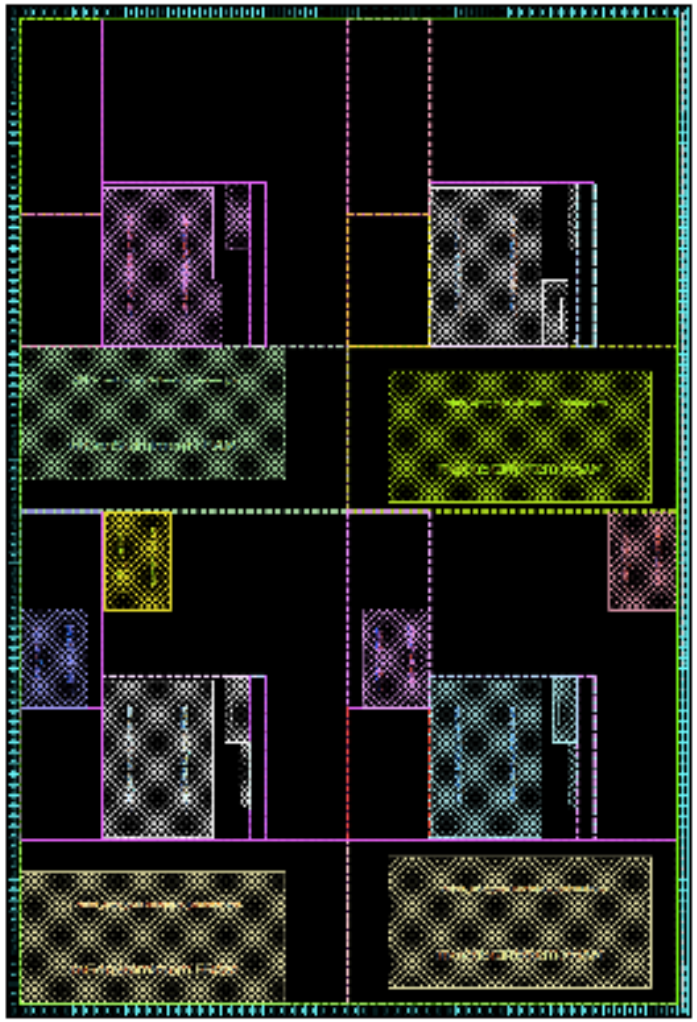
- GreenDroid c-cores use 11x less energy per instruction than an aggressive mobile application processor

- Including cold code, GreenDroid will still save ~7.5x energy

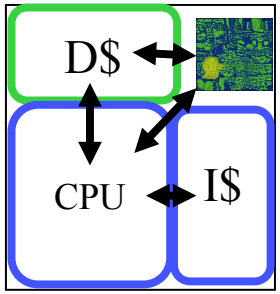
Aggressive mobile application processor (45 nm, 1.5 GHz)	91	pJ/instr.
GreenDroid c-cores, 45 nm	8	pJ/instr.
GreenDroid c-cores + cold code (est.)	12	pJ/instr.

Quad-core GreenDroid Prototype

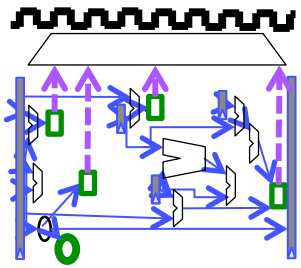
- Four heterogeneous tiles with ~40 C-cores.
- Synopsys IC Compiler
- 28-nm Global Foundries
- 1.5-2 GHz
- 2 mm^2
- In verification stages
- Multiproject Tapeout w/ UCSC



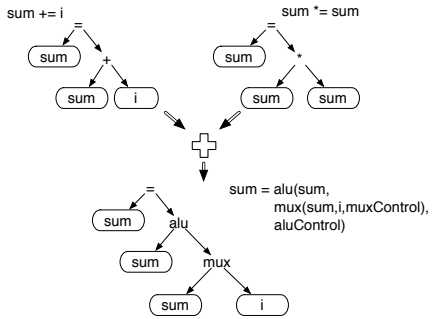
ASPLOS ‘10 → **HPCA ’11** → **MICRO ’11**



C-cores; Patching; Util. Wall

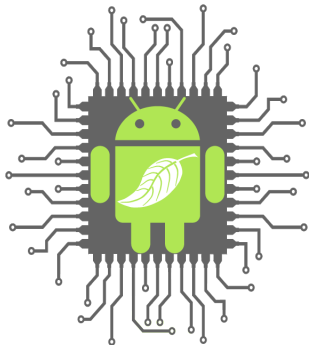


Selective Depipelining; Cachelets

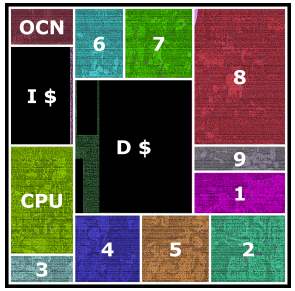


Automatic C-core General-ization

HOTCHIPS ‘10 → **IEEE MICRO ‘11**



GreenDroid; Dark Silicon



GreenDroid P&R Tile

FCCM ’11 → **FPL ’11**

C-cores for FPGAs

Selective Depipelining for FPGAs

The UCSD GreenDroid Team

Prof. Taylor

greendroid.org

Prof. Swanson

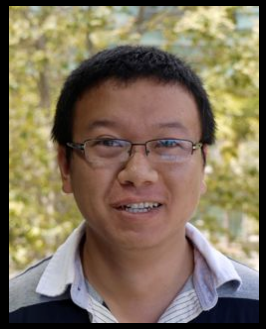
Heroic Students












Conclusions

- Dark Silicon is opening up a whole new class of exciting new research areas. (Submit to DaSi!)
- Conservation cores use dark silicon to attack the utilization wall.
- GreenDroid is evaluating the benefits of c-cores for mobile application processors; a 28-nm prototype is underway.

Patchability Payoff: Longevity

- Graceful degradation
  - Lower initial efficiency
  - Much longer useful lifetime
- Can support any changes, through *patching*
  - Arbitrary insertion of code – software exception mechanism
  - Changes to program constants
  - Changes to operators