GreenDroid: A Mobile Application Processor for a Future of Dark Silicon

Nathan Goulding, Jack Sampson, Ganesh Venkatesh,
Saturnino Garcia, Joe Auricchio, Jonathan Babb[+],
Michael B. Taylor, and Steven Swanson

Department of Computer Science and Engineering,
University of California, San Diego

[+] *CSAIL, Massachusetts Institute of Technology*

We've Hit The Utilization Wall

Utilization Wall:

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

We've Hit The Utilization Wall

Utilization Wall: With each successive process generation, the percentage of a chip that can actively switch drops exponentially due to power constraints.

- Scaling theory
  - Transistor and power budgets are no longer balanced
  - Exponentially increasing problem!

- Experimental results
  - Replicated a small datapath
  - More "dark silicon" than active

- Observations in the wild
  - Flat frequency curve
  - "Turbo Mode"
  - Increasing cache/processor ratio

We've Hit The Utilization Wall

Utilization Wall: With each successive process generation, the percentage of a chip that can actively switch drops exponentially due to power constraints.

- **Scaling theory**
  - Transistor and power budgets are no longer balanced
  - Exponentially increasing problem!

- **Experimental results**
  - Replicated a small datapath
  - More "dark silicon" than active

- **Observations in the wild**
  - Flat frequency curve
  - "Turbo Mode"
  - Increasing cache/processor ratio

Classical scaling

Device count	S^2
Device frequency	S
Device power (cap)	$1/S$
Device power (V_{dd})	$1/S^2$
Utilization	**1**

Leakage-limited scaling

Device count	S^2
Device frequency	S
Device power (cap)	$1/S$
Device power (V_{dd})	**~1**
Utilization	**$1/S^2$**

We've Hit The Utilization Wall

Utilization Wall: With each successive process generation, the percentage of a chip that can actively switch drops exponentially due to power constraints.

- Scaling theory
  - Transistor and power budgets are no longer balanced
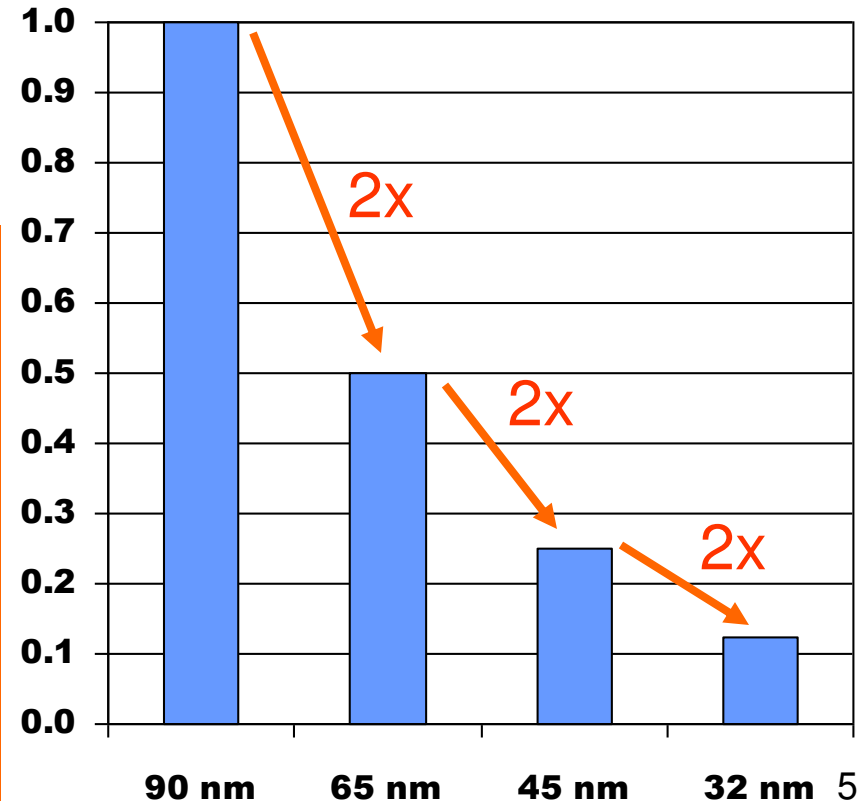 - Exponentially increasing problem!

- Experimental results
  - Replicated a small datapath
  - More "dark silicon" than active

- Observations in the wild
  - Flat frequency curve
  - "Turbo Mode"
  - Increasing cache/processor ratio

Expected utilization for fixed area and power budget

2x

2x

2x

| 90 nm | 65 nm | 45 nm | 32 nm |

We've Hit The Utilization Wall

Utilization Wall: With each successive process generation, the percentage of a chip that can actively switch drops exponentially due to power constraints.
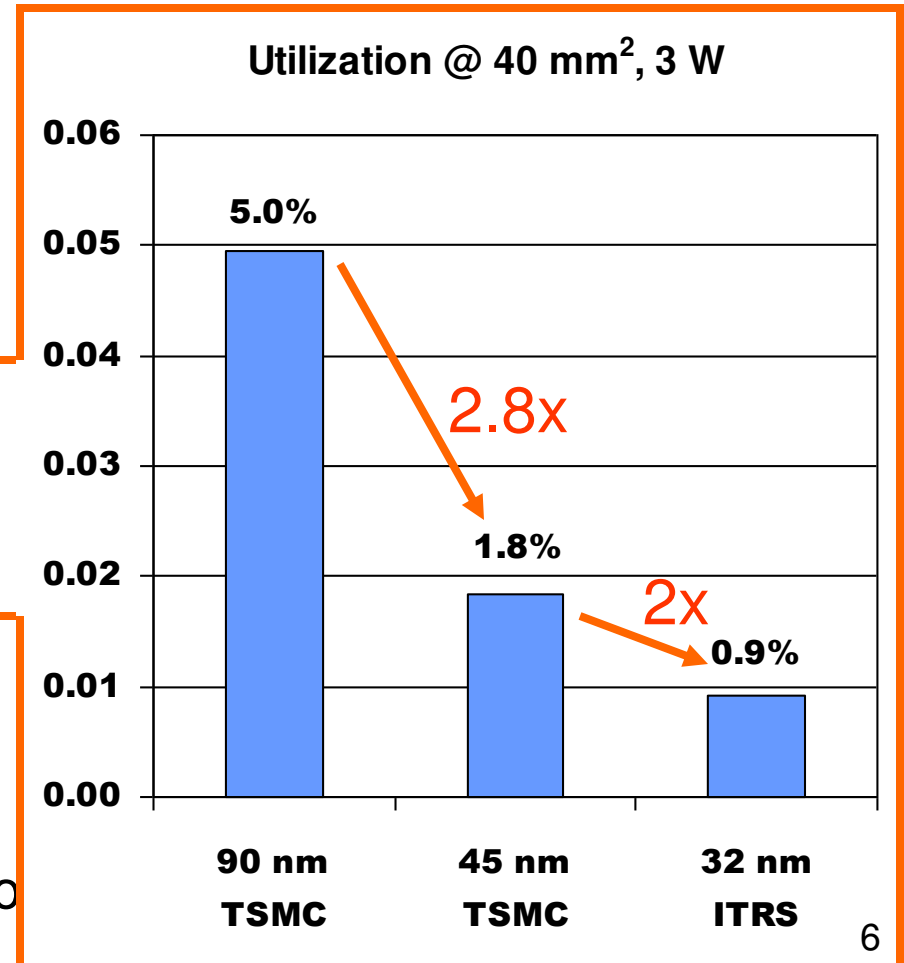
- Scaling theory
  - Transistor and power budgets are no longer balanced
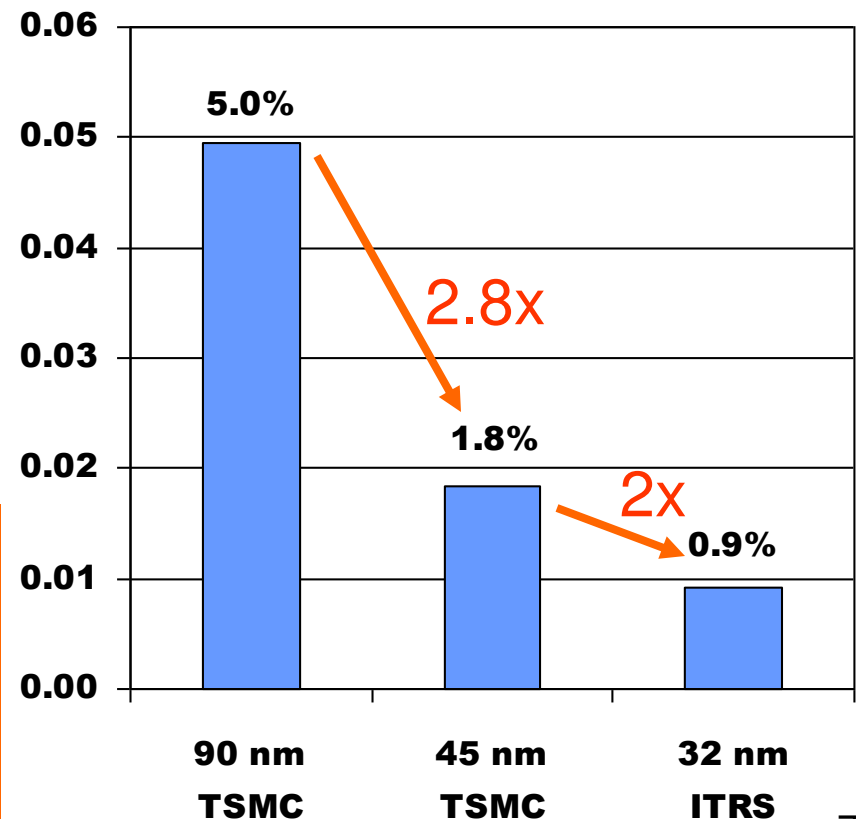 - Exponentially increasing problem!

- Experimental results
  - Replicated a small datapath
  - More "dark silicon" than active

- Observations in the wild
  - Flat frequency curve
  - "Turbo Mode"
  - Increasing cache/processor ratio

Utilization @ 40 mm², 3 W

	5.0%		

2.8x

1.8%

2x

0.9%

90 nm TSMC	45 nm TSMC	32 nm ITRS

We've Hit The Utilization Wall

Utilization Wall: With each successive process generation, the percentage of a chip that can actively switch drops exponentially due to power constraints.

- **Scaling theory**
  - Transistor and power budgets are no longer balanced
  - Exponentially increasing problem!

- **Experimental results**
  - Replicated a small datapath
  - More "dark silicon" than active

- **Observations in the wild**
  - Flat frequency curve
  - "Turbo Mode"
  - Increasing cache/processor ratio

Utilization @ 40 mm^2, 3 W

90 nm TSMC	**45 nm TSMC**	**32 nm ITRS**

5.0% → 2.8x → 1.8% → 2x → 0.9%

We've Hit The Utilization Wall

Utilization Wall: With each successive process generation, the percentage of a chip that can actively switch drops exponentially due to power constraints.

- Scaling theory
  - Transistor dense and sats
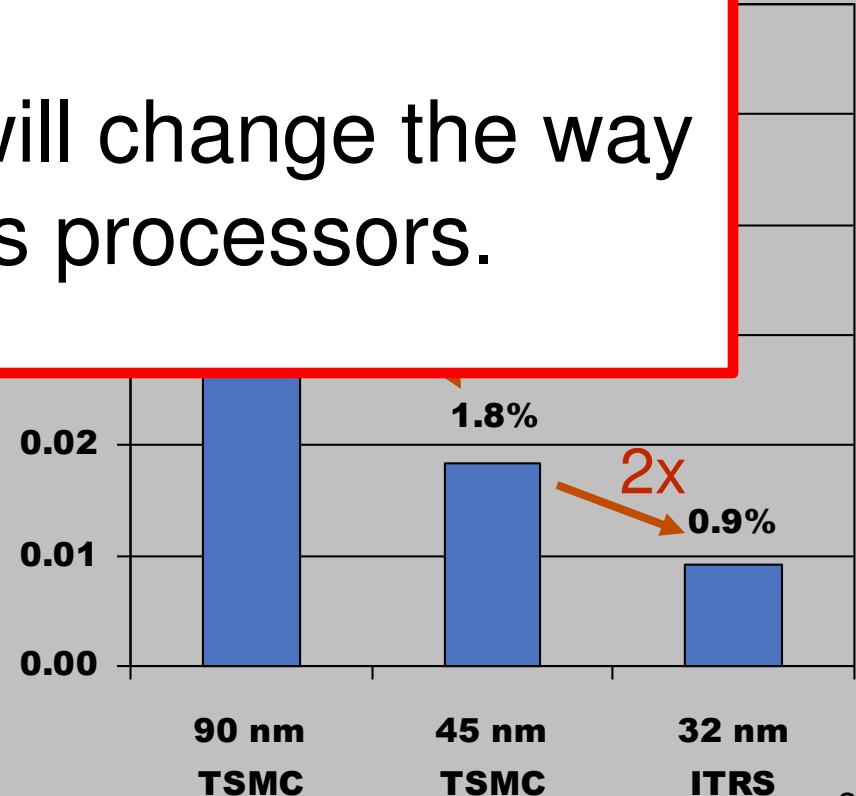
Utilization @ 40 mm², 3 W

The utilization wall will change the way everyone builds processors.

- E
  - Replicated a small datapath
  - More "dark silicon" than active

- Observations in the wild
  - Flat frequency curve
  - "Turbo Mode"
  - Increasing cache/processor ratio

1.8%

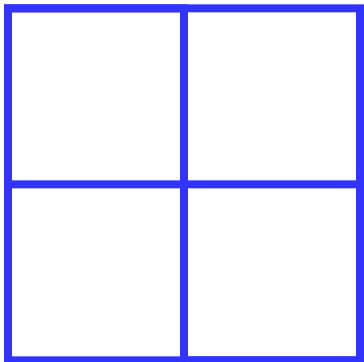2x

0.9%

0.02

0.01

0.00

| 90 nm TSMC | 45 nm TSMC | 32 nm ITRS |

Utilization Wall:
Dark Implications for Multicore

Spectrum of tradeoffs
between # of cores and
frequency
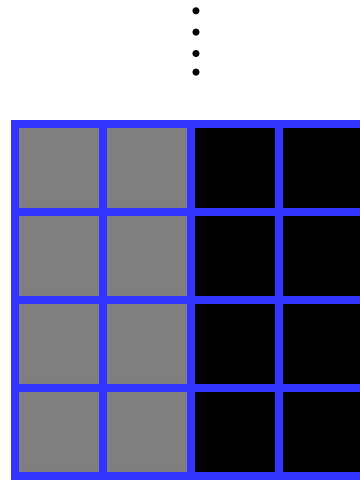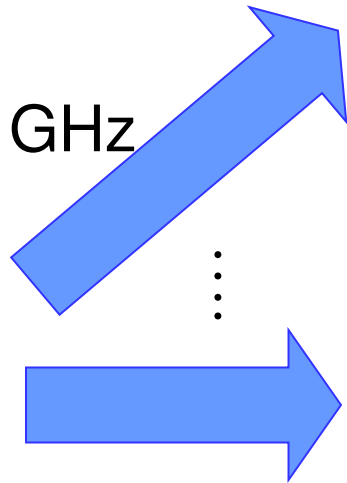
Example:
65 nm → 32 nm (S = 2)
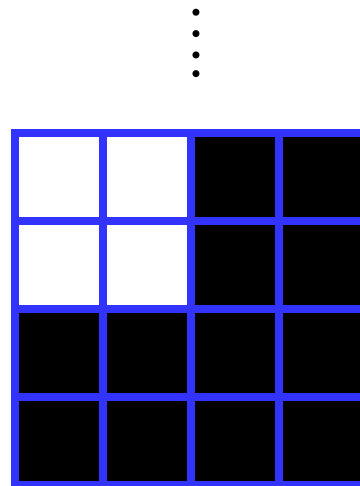
4 cores @ 1.8 GHz

2x4 cores @ 1.8 GHz
(8 cores dark, 8 dim)

(*Industry's Choice*)

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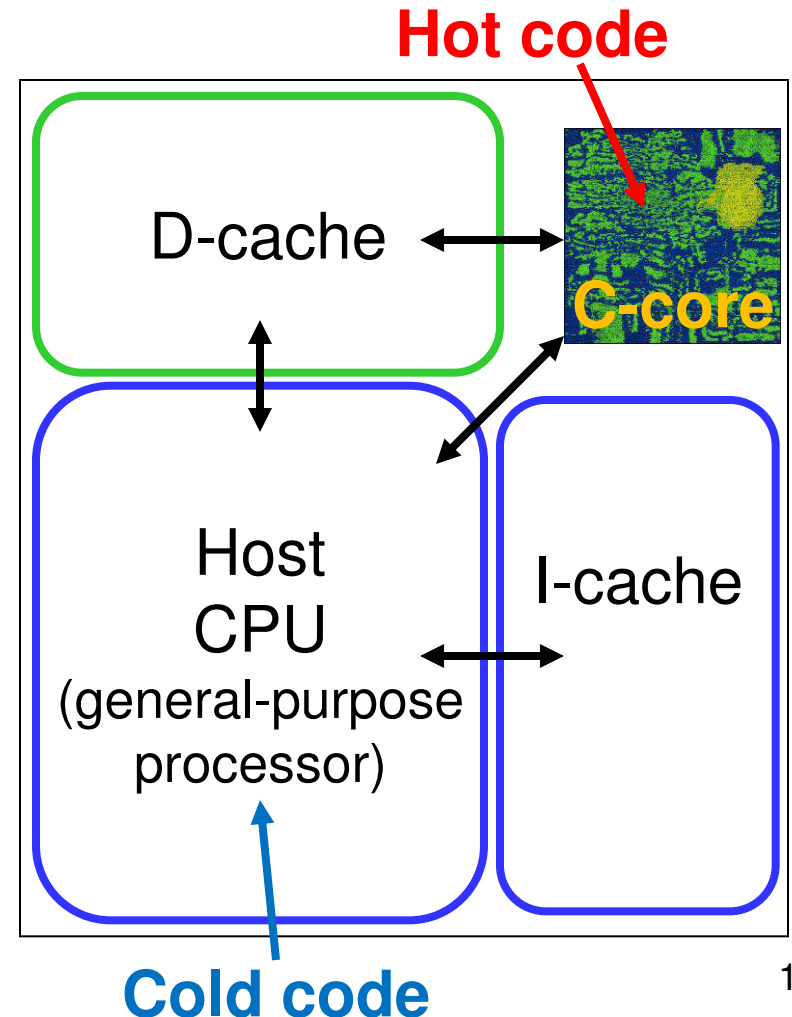(12 cores dark)

65 nm   32 nm

What do we do with dark silicon?

- Goal: Leverage dark silicon to scale the utilization wall

- Insights:
  - Power is now more expensive than area
  - Specialized logic can improve energy efficiency (10–1000x)

- Our approach:
  - Fill dark silicon with specialized cores to save energy on common applications
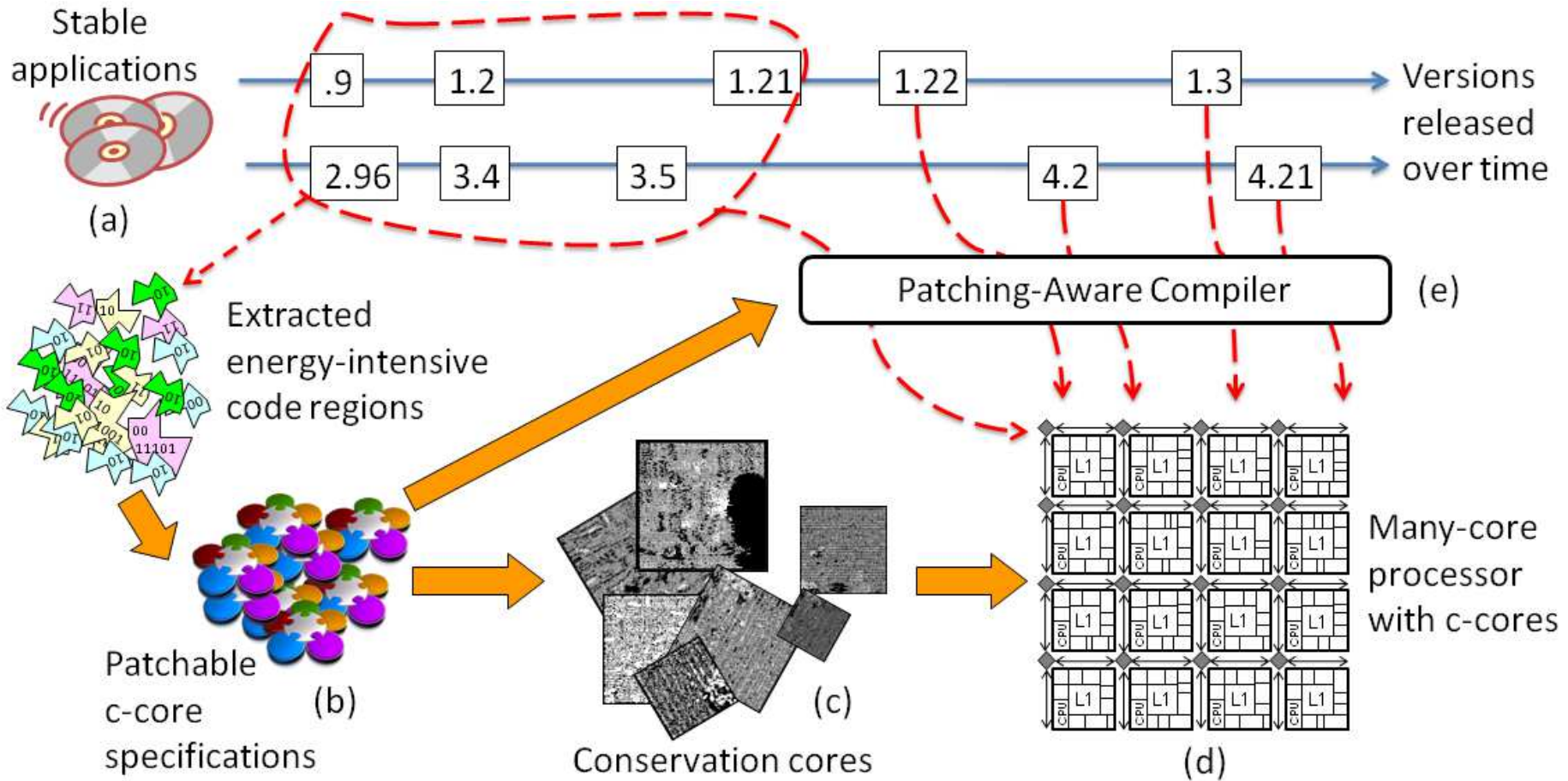 - Provide focused reconfigurability to handle evolving workloads

Conservation Cores

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

- **Specialized circuits for reducing energy**
  - Automatically generated from hot regions of program source code
  - Patching support future-proofs the hardware

- **Fully-automated toolchain**
  - Drop-in replacements for code
  - Hot code implemented by c-cores, cold code runs on host CPU
  - HW generation/SW integration

- **Energy-efficient**
  - Up to 18x for targeted hot code

Hot code

D-cache

C-core

Host CPU
(general-purpose processor)

I-cache

Cold code

11

The C-core Life Cycle

Stable applications (a)

.9 1.2 1.21 1.22 1.3 → Versions released over time

2.96 3.4 3.5 4.2 4.21

Extracted energy-intensive code regions

Patchable c-core specifications (b)

Conservation cores (c)

Patching-Aware Compiler (e)

Many-core processor with c-cores (d)
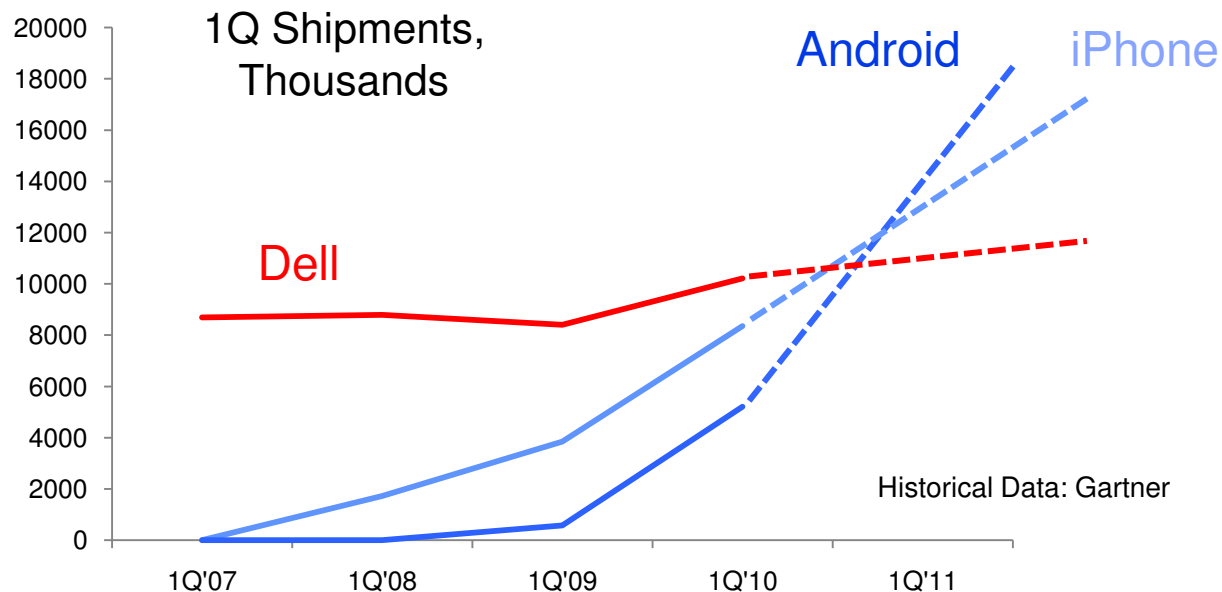
Outline

- Utilization wall and dark silicon

- GreenDroid

- Conservation cores

- GreenDroid energy savings

- Conclusions

Emerging Trends

The *utilization wall* is exponentially worsening the dark silicon problem.

Specialized architectures are receiving more and more attention because of energy efficiency.

Mobile application processors are becoming a dominant computing platform for end users.

20000
18000
16000
14000
12000
10000
8000
6000
4000
2000
0

1Q Shipments, Thousands

Android   iPhone

Dell

Historical Data: Gartner

1Q'07   1Q'08   1Q'09   1Q'10   1Q'11

14

# Mobile Application Processors Face the Utilization Wall

- The evolution of mobile application processors mirrors that of microprocessors

- Application processors face the utilization wall

  - Growing performance demands

  - Extreme power constraints
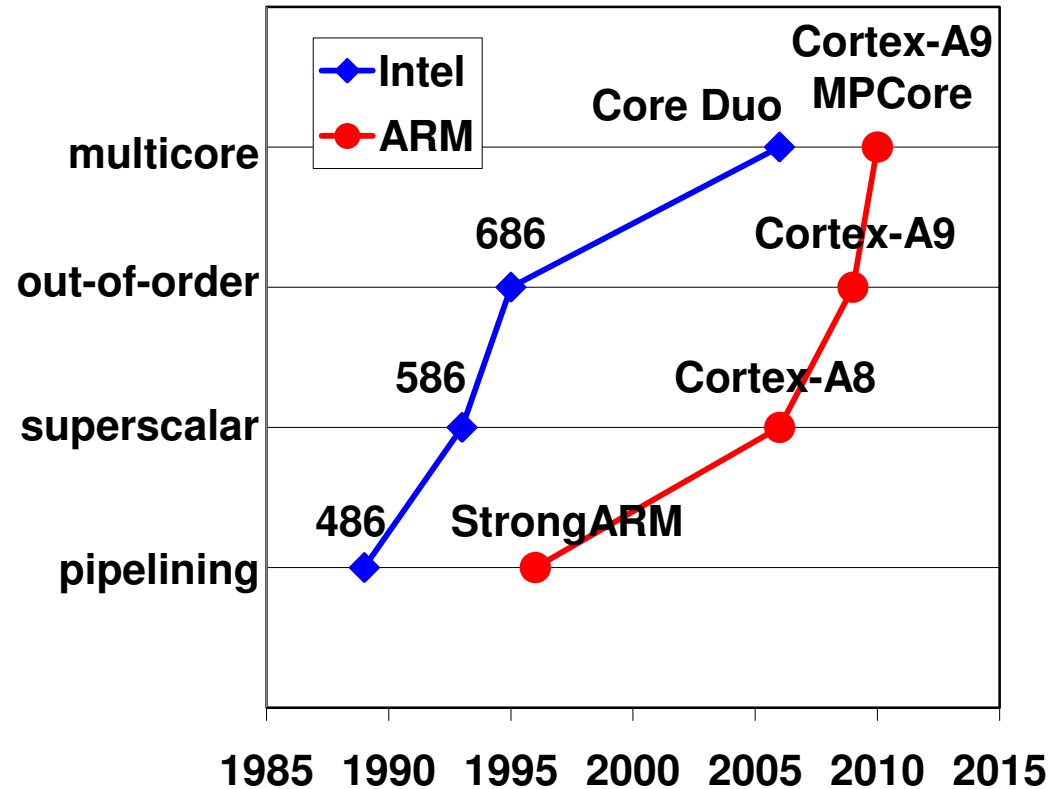
Android™

- 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.)

Applications	
Libraries	Dalvik
Linux Kernel	

| Hardware |
|---|

16
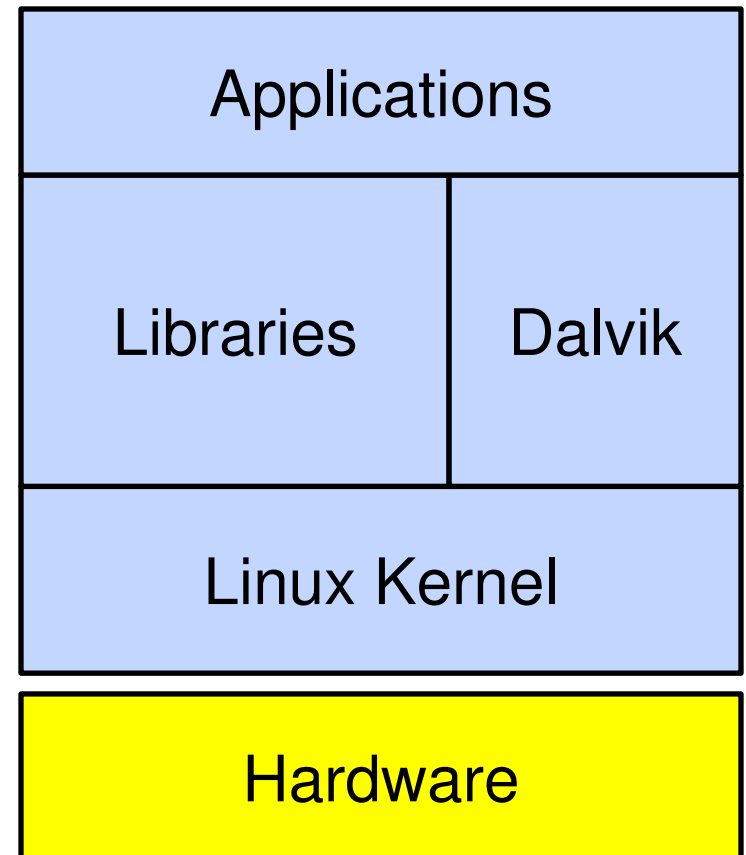
Applying C-cores to Android

- **Android is well-suited for c-cores**
  - Core set of commonly used applications
  - Libraries are hot code
  - Dalvik virtual machine is hot code
  - Libraries, Dalvik, and kernel & application hotspots → c-cores
  - Relatively short hardware replacement cycle

C-cores

Applications

Libraries

Dalvik

Linux Kernel

Hardware
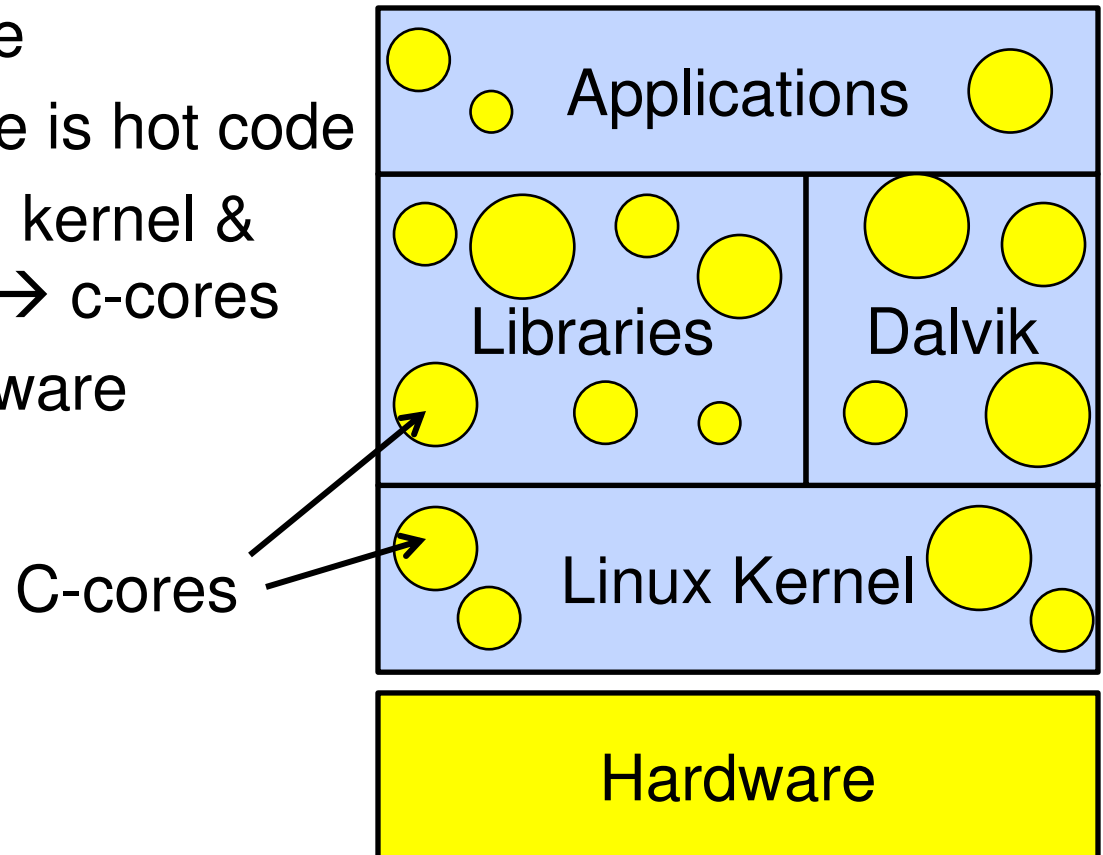
Android Workload Profile

- 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)
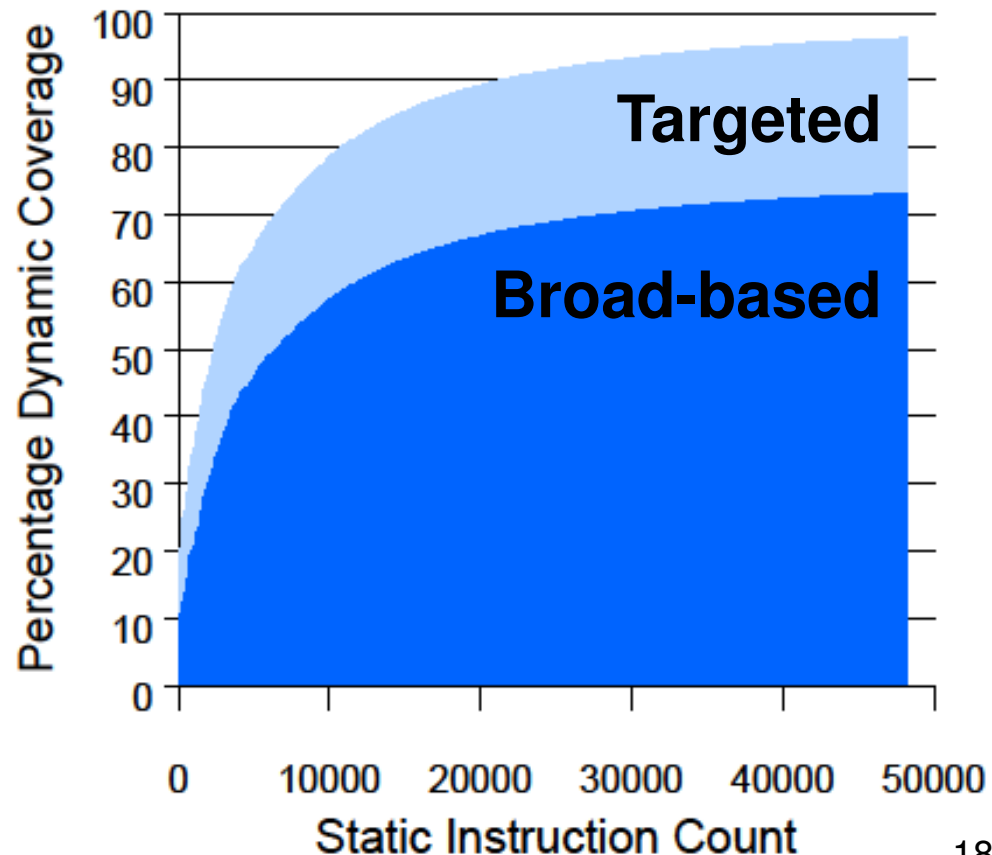
GreenDroid: Applying Massive Specialization to Mobile Application Processors
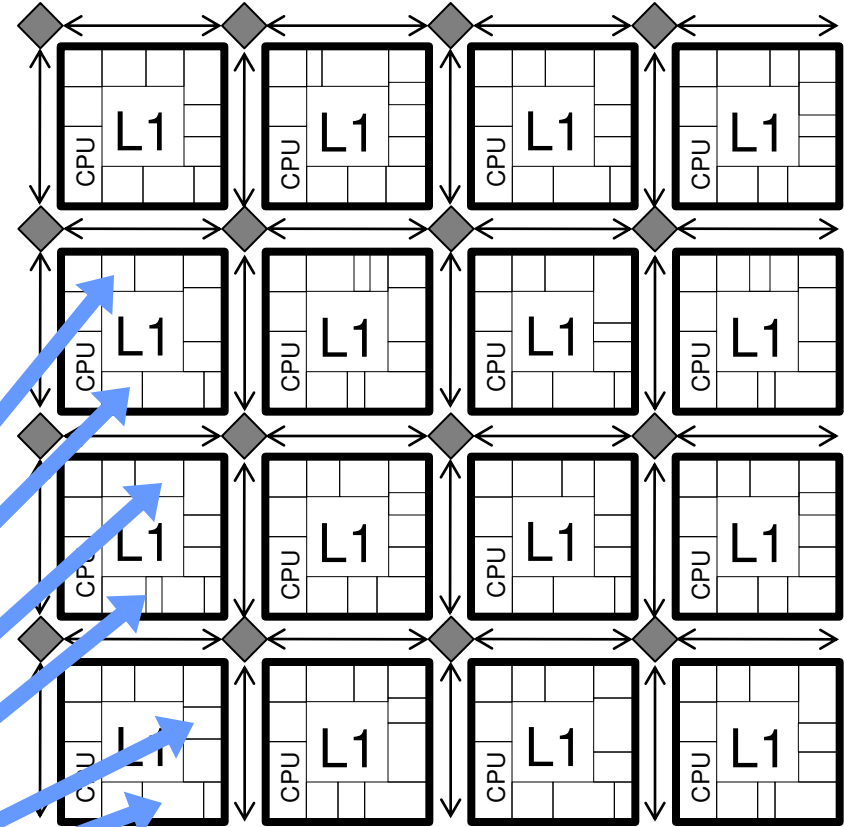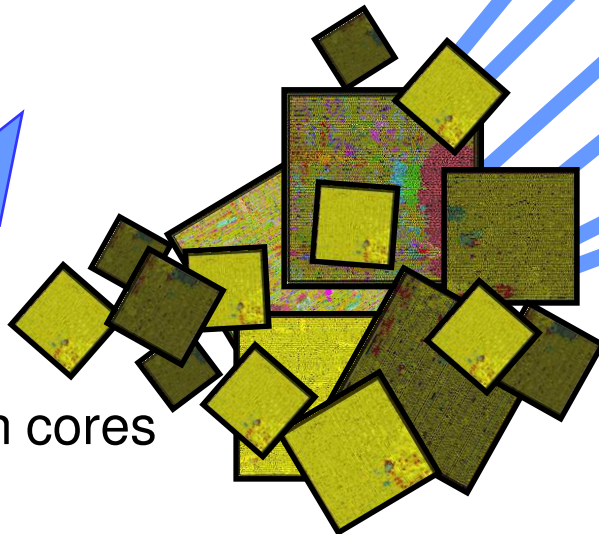
Android workload
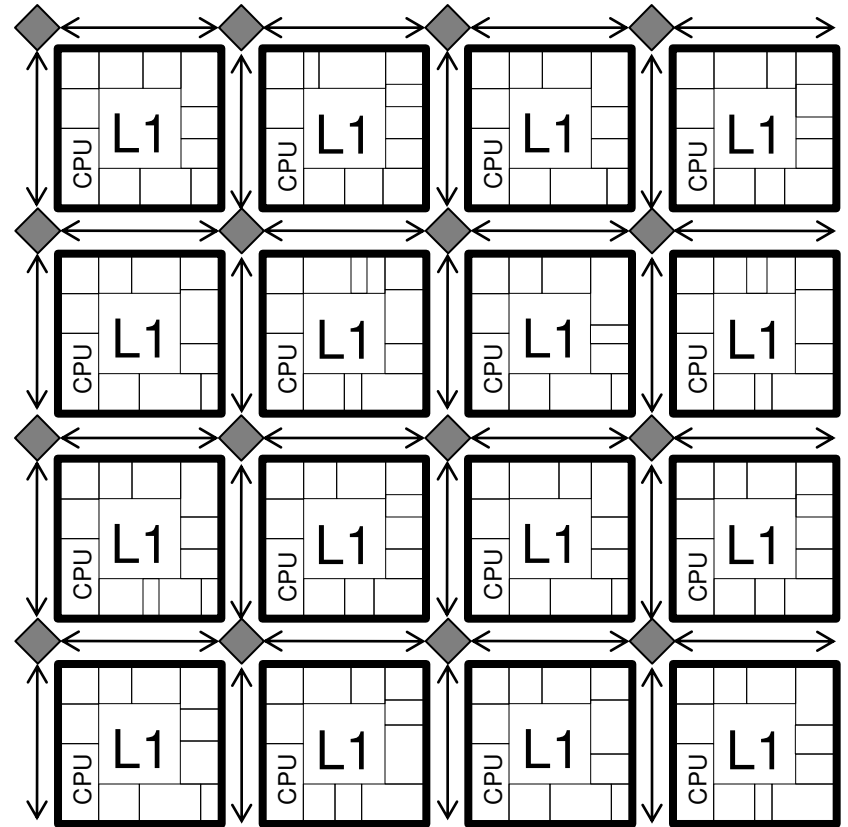
Automatic c-core generator

Conservation cores (c-cores)

Low-power tiled multicore lattice

GreenDroid Tiled Architecture
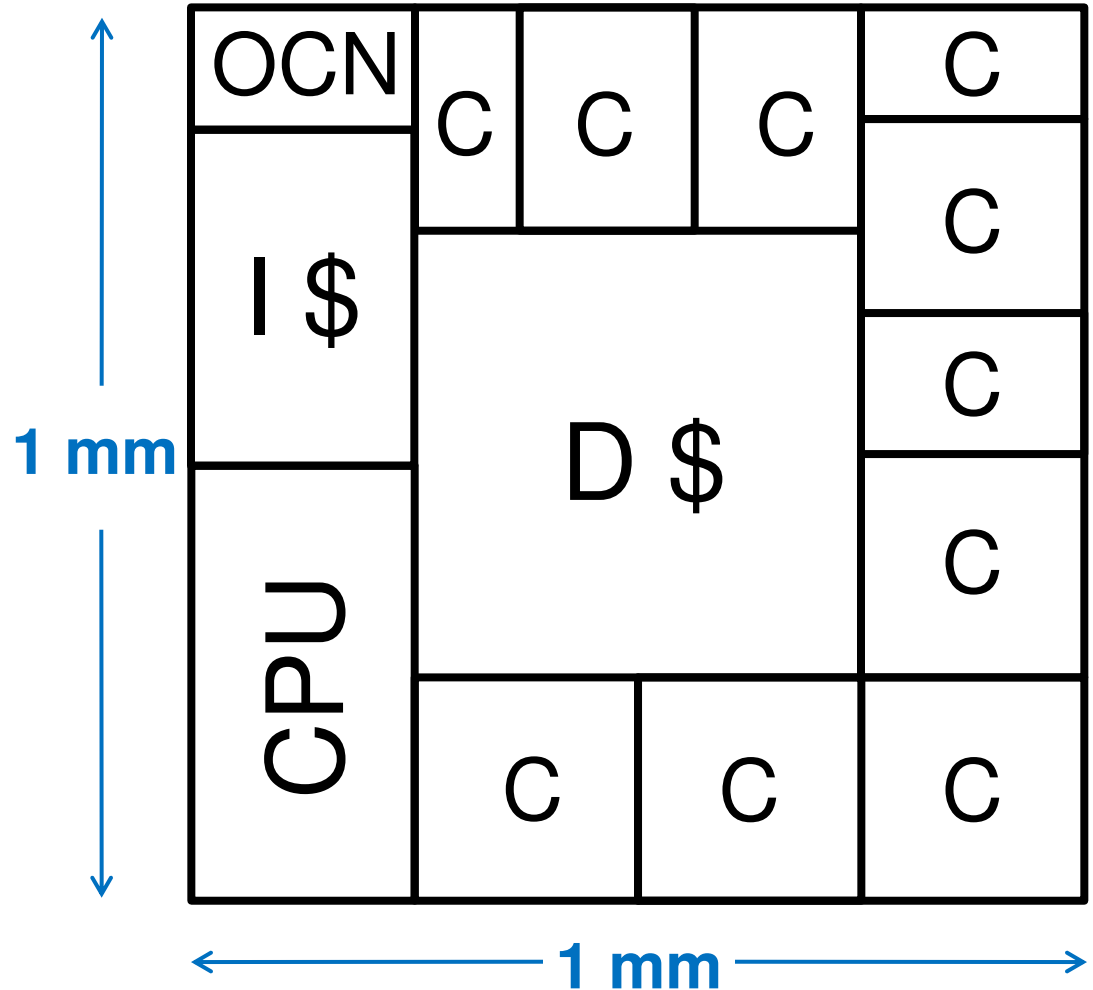
- Tiled lattice of 16 cores
- Each tile contains
  - 6-10 Android c-cores (~125 total)
  - 32 KB D-cache (shared with CPU)
  - MIPS processor
    - 32 bit, in-order, 7-stage pipeline
    - 16 KB I-cache
    - Single-precision FPU
  - On-chip network router

GreenDroid Tile Floorplan

- 1.0 mm^2 per tile

- 50% C-cores
- 25% D-cache
- 25% MIPS core, I-cache, and on-chip network

GreenDroid Tile Skeleton

- 45 nm process

- 1.5 GHz

- ~30k instances

- Blank space is filled with a collection of c-cores

- Each tile contains different c-cores
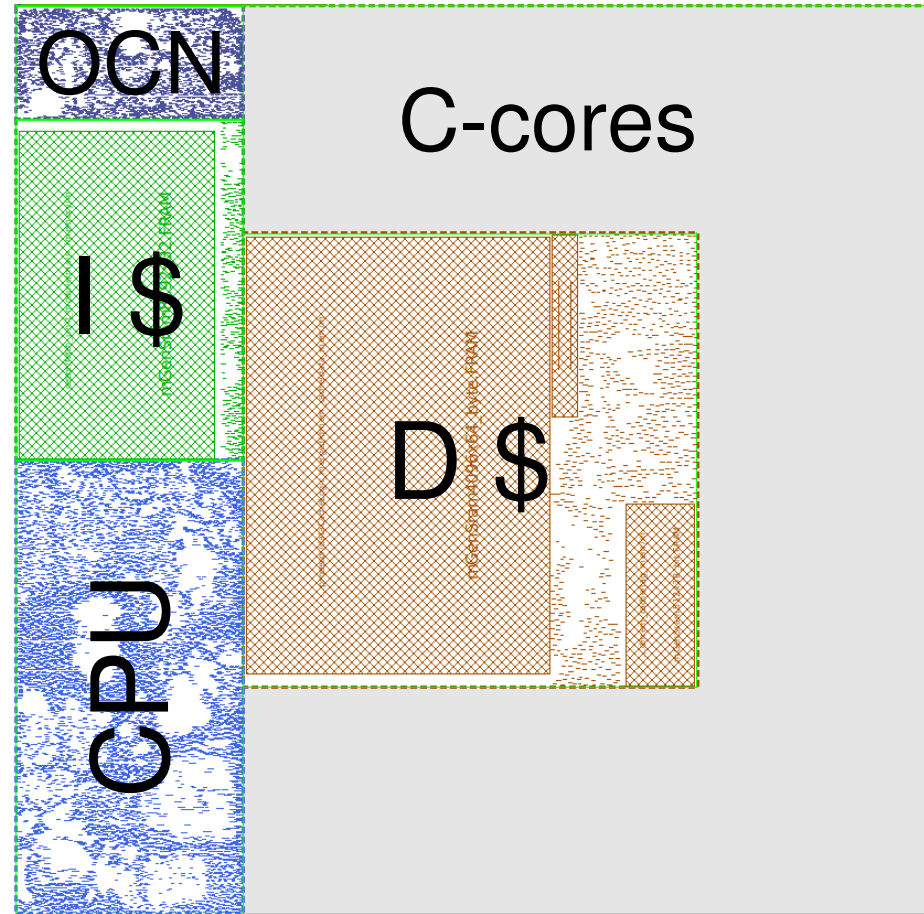
OCN

C-cores

I $

D $

CPU

Outline

- Utilization wall and dark silicon

- GreenDroid

- Conservation cores

- GreenDroid energy savings

- Conclusions

Constructing a C-core

- C-cores start with source code
  - Can be irregular, integer programs
  - Parallelism-agnostic

- Supports almost all of C:
  - Complex control flow
    e.g., goto, switch, function calls
  - Arbitrary memory structures
    e.g., pointers, structs, stack, heap
  - Arbitrary operators
    e.g., floating point, divide
  - Memory coherent with host CPU

```c
sumArray(int *a, int n)
{
  int i = 0;
  int sum = 0;

  for (i = 0; i < n; i++) {
    sum += a[i];
  }

  return sum;
}
```
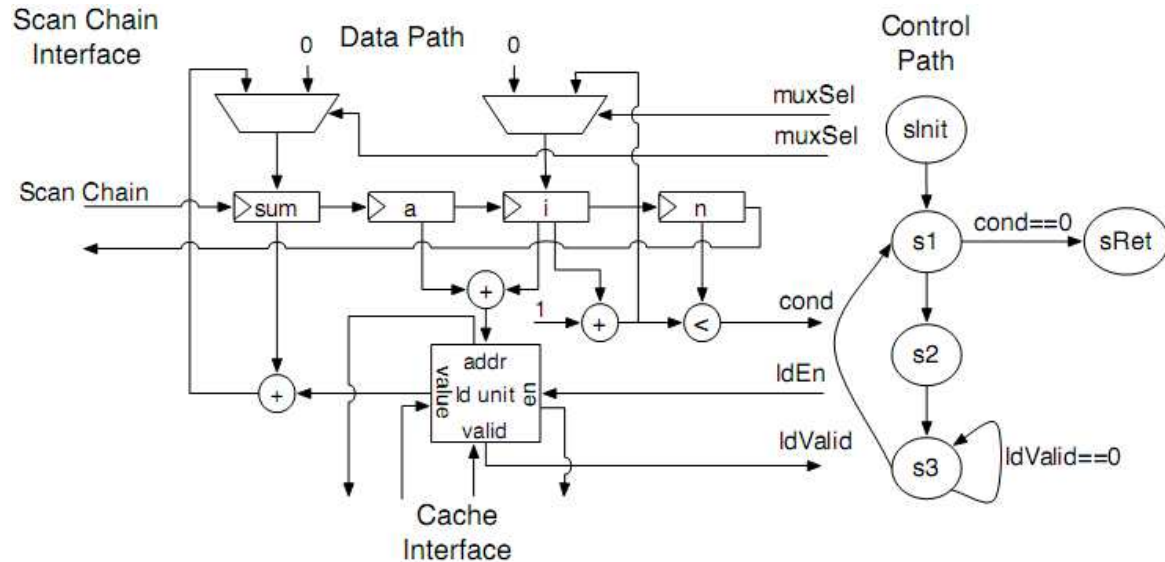
Constructing a C-core
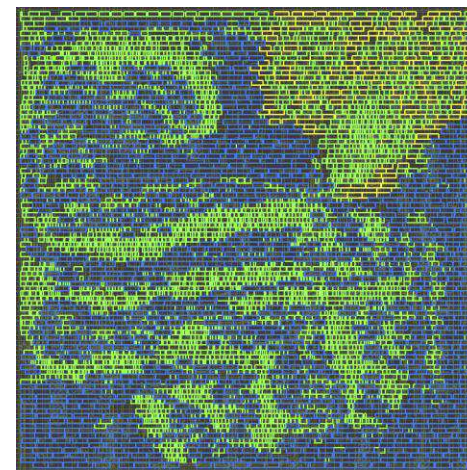
- Compilation
  - C-core selection
  - SSA, infinite register, 3-address code
  - Direct mapping from CFG and DFG
  - Scan chain insertion

- Verilog → Place & Route
  - 45 nm process
  - Synopsys CAD flow
    - Synthesis
    - Placement
    - Clock tree generation
    - Routing

0.01 mm^2, 1.4 GHz

C-cores Experimental Data

- We automatically built 21 c-cores for 9 "hard" applications

  - 45 nm TSMC

  - Vary in size from 0.10 to 0.25 mm^2

  - Frequencies from 1.0 to 1.4 GHz

Application	# C-cores	Area (mm^2)	Frequency (MHz)
bzip2	1	0.18	1235
cjpeg	3	0.18	1451
djpeg	3	0.21	1460
mcf	3	0.17	1407
radix	1	0.10	1364
sat solver	2	0.20	1275
twolf	6	0.25	1426
viterbi	1	0.12	1264
vpr	1	0.24	1074

C-core Energy Efficiency: Non-cache Operations
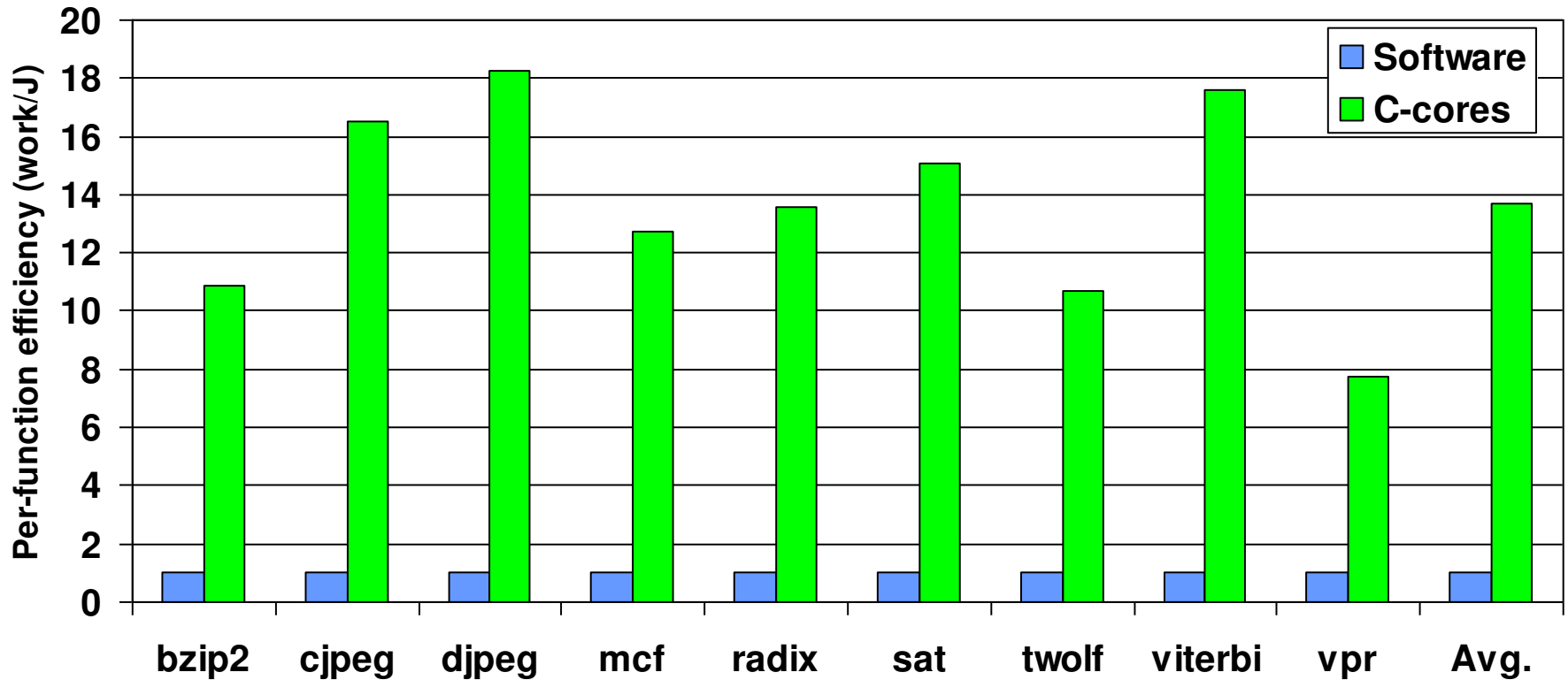
- Up to 18x more energy-efficient (13.7x on average), compared to running on the MIPS processor

Where do the energy savings come from?

D-cache
6%

I-cache
23%

Fetch/
Decode
19%

Datapath
38%

Reg. File
14%
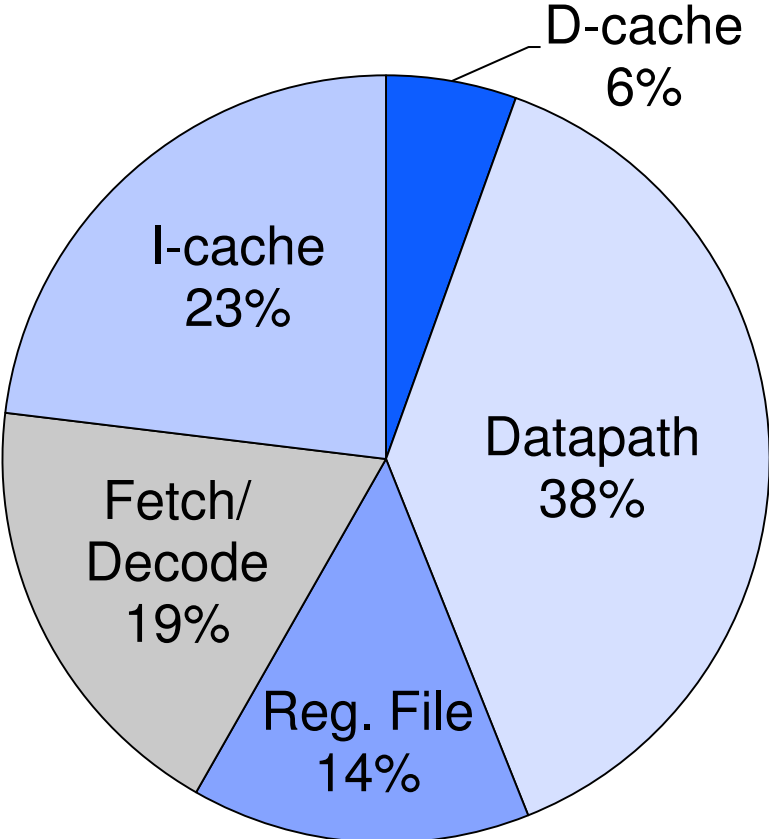
MIPS baseline
91 pJ/instr.

D-cache
6%

Datapath
3%

Energy
Saved
91%

C-cores
8 pJ/instr.

28
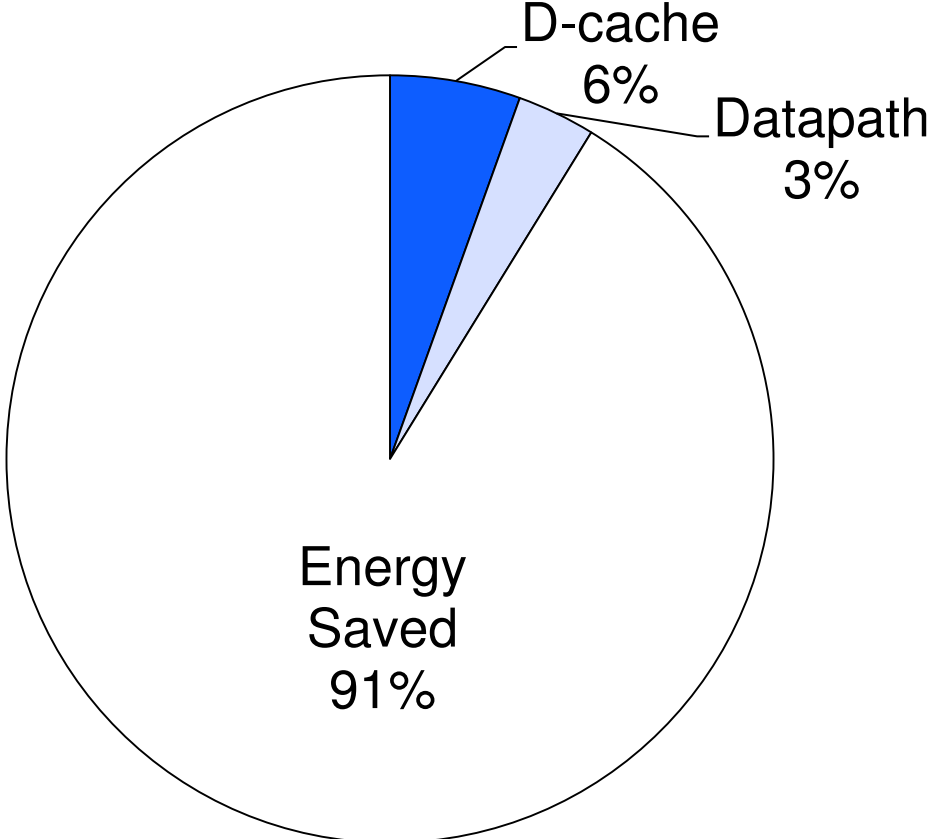
Supporting Software Changes

- Software may change – HW must remain usable
  - C-cores unaffected by changes to cold regions
- Can support any changes, through *patching*
  - Arbitrary insertion of code – software exception mechanism
  - Changes to program constants – configurable registers
  - Changes to operators – configurable functional units
- Software exception mechanism
  - Scan in values from c-core
  - Execute in processor
  - Scan out values back to c-core to resume execution

Patchability Payoff: Longevity

■ **Graceful degradation**
  – Lower initial efficiency
  – Much longer useful lifetime

■ **Increased viability**
  – With patching, utility lasts ~10 years for 4 out of 5 applications
  – Decreases risks of specialization
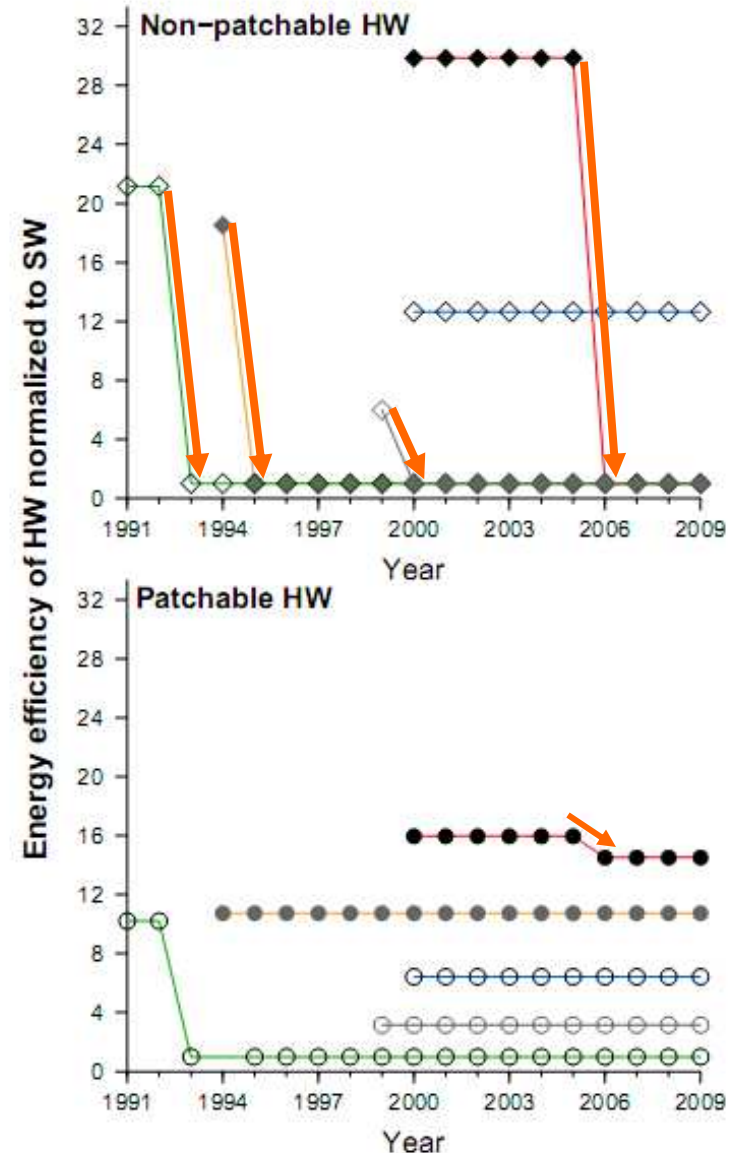
Outline

- Utilization wall and dark silicon

- GreenDroid

- Conservation cores

- **GreenDroid energy savings**

- **Conclusions**

GreenDroid: Energy per Instruction

- More area dedicated to c-cores yields higher execution coverage and lower energy per instruction (EPI)

Average Energy per Instruction (pJ) vs **C-core Area (mm^2)**
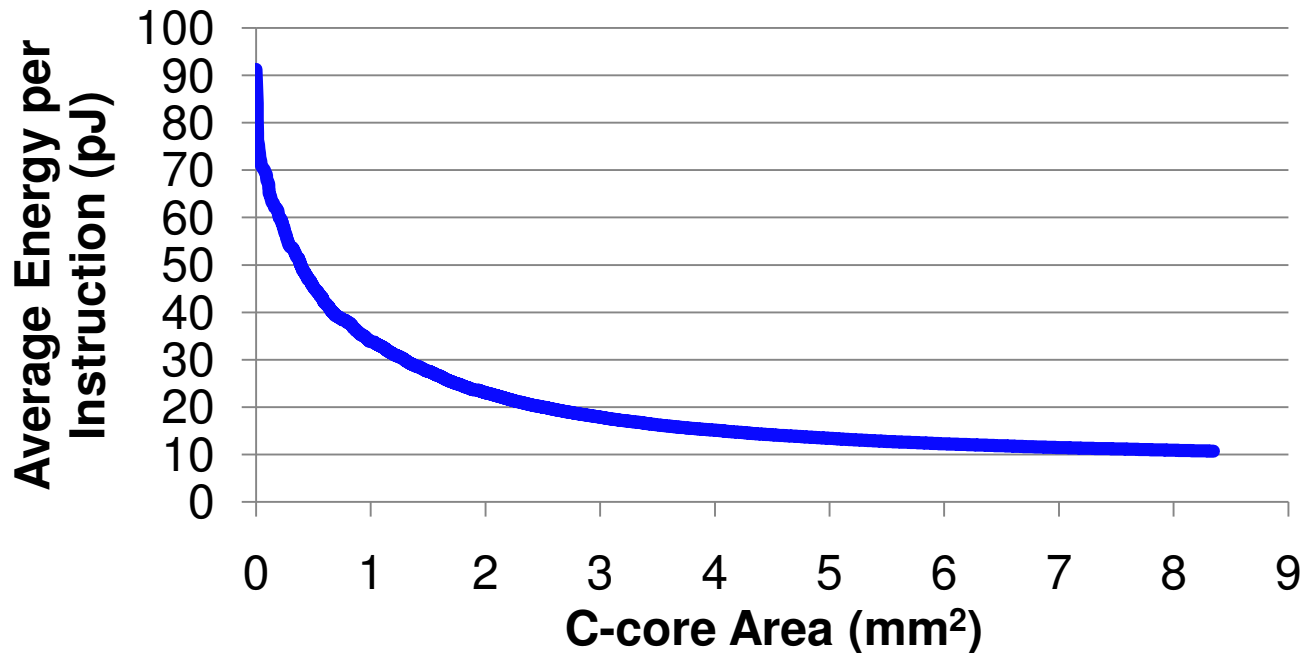
- 7 mm^2 of c-cores provides:
  - 95% execution coverage
  - 8x energy savings over MIPS core

What kinds of hotspots turn into GreenDroid c-cores?

C-core	Library	# Apps	Coverage (est., %)	Area (est., mm^2)	Broad-based
dvmInterpretStd	libdvm	8	10.8	0.414	Y
scanObject	libdvm	8	3.6	0.061	Y
S32A_D565_Opaque_Dither	libskia	8	2.8	0.014	Y
src_aligned	libc	8	2.3	0.005	Y
S32_opaque_D32_filter_DXDY	libskia	1	2.2	0.013	N
less_than_32_left	libc	7	1.7	0.013	Y
cached_aligned32	libc	9	1.5	0.004	Y
.plt	<many>	8	1.4	0.043	Y
memcpy	libc	8	1.2	0.003	Y
S32A_Opaque_BlitRow32	libskia	7	1.2	0.005	Y
ClampX_ClampY_filter_affine	libskia	4	1.1	0.015	Y
DiagonalInterpMC	libomx	1	1.1	0.054	N
blitRect	libskia	1	1.1	0.008	N
calc_sbr_synfilterbank_LC	libomx	1	1.1	0.034	N
inflate	libz	4	0.9	0.055	Y
. . .	. . .	. . .	. . .	. . .	. . .

GreenDroid: Projected Energy

Aggressive mobile application processor   91 pJ/instr.
(45 nm, 1.5 GHz)

GreenDroid c-cores   8 pJ/instr.

GreenDroid c-cores + cold code (est.)   12 pJ/instr.

- 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

Project Status

- **Completed**
  - Automatic generation of c-cores from source code to place & route
  - Cycle- and energy-accurate simulation (post place & route)
  - Tiled lattice, placed and routed
  - FPGA emulation of Android-based c-cores and tiled lattice

- **Ongoing work**
  - Finish full system Android emulation for more accurate workload modeling
  - Finalize c-core selection based on full system Android workload model
  - Timing closure and tapeout

GreenDroid Conclusions

- The utilization wall forces us to change how we build hardware

- Conservation cores use dark silicon to attack the utilization wall

- GreenDroid will demonstrate the benefits of c-cores for mobile application processors

- We are developing a 45 nm tiled prototype at UCSD

GreenDroid: A Mobile Application Processor for a Future of Dark Silicon

Nathan Goulding, Jack Sampson, Ganesh Venkatesh,
Saturnino Garcia, Joe Auricchio, Jonathan Babb[+],
Michael B. Taylor, and Steven Swanson

Department of Computer Science and Engineering,

University of California, San Diego

[+] *CSAIL, Massachusetts Institute of Technology*

Backup Slides

Automated Measurement Methodology

- **C-core toolchain**
  - Specification generator
  - Verilog generator

- **Synopsys CAD flow**
  - Design Compiler
  - IC Compiler
  - 45 nm library

- **Simulation**
  - Validated cycle-accurate c-core modules
  - Post-route gate-level simulation

- **Power measurement**
  - VCS + PrimeTime

Source

Hotspot analyzer

Cold code   Hot code

Rewriter

C-core specification generator

Verilog generator

gcc

Simulation

Synopsys flow

Power measurement