Conservation Cores: Reducing the Energy of Mature Computations

Ganesh Venkatesh, **Jack Sampson**, Nathan Goulding,
Saturnino Garcia, Vladyslav Bryksin, Jose Lugo-Martinez,
Steven Swanson, Michael Bedford Taylor

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

The Utilization Wall

- Scaling theory
  - Transistor and power budgets no longer balanced
  - Exponentially increasing problem!
- Experimental results
  - Replicated small datapath
  - More ‘Dark Silicon’ than active
- Observations in the wild
  - Flat frequency curve
  - “Turbo Mode”
  - Increasing cache/processor ratio

Classical scaling

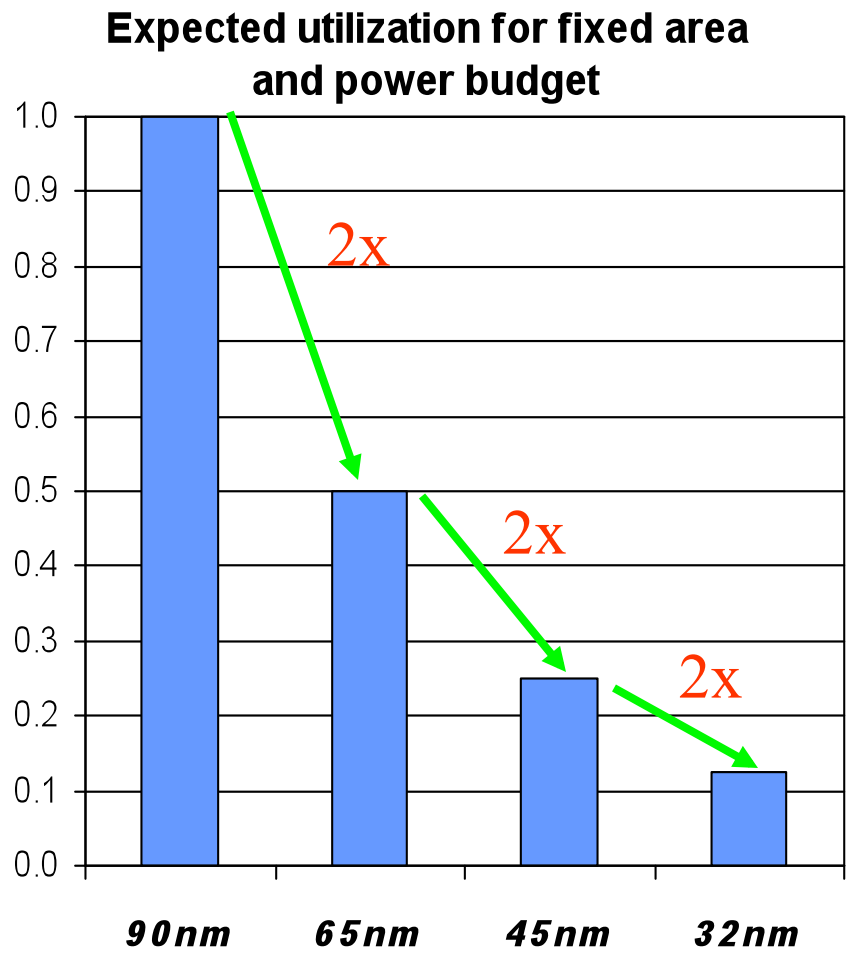
Device count	S^2
Device frequency	S
Device power (cap)	1/S
Device power (V_{dd})	$1/S^2$
Utilization	**1**

Leakage limited scaling

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

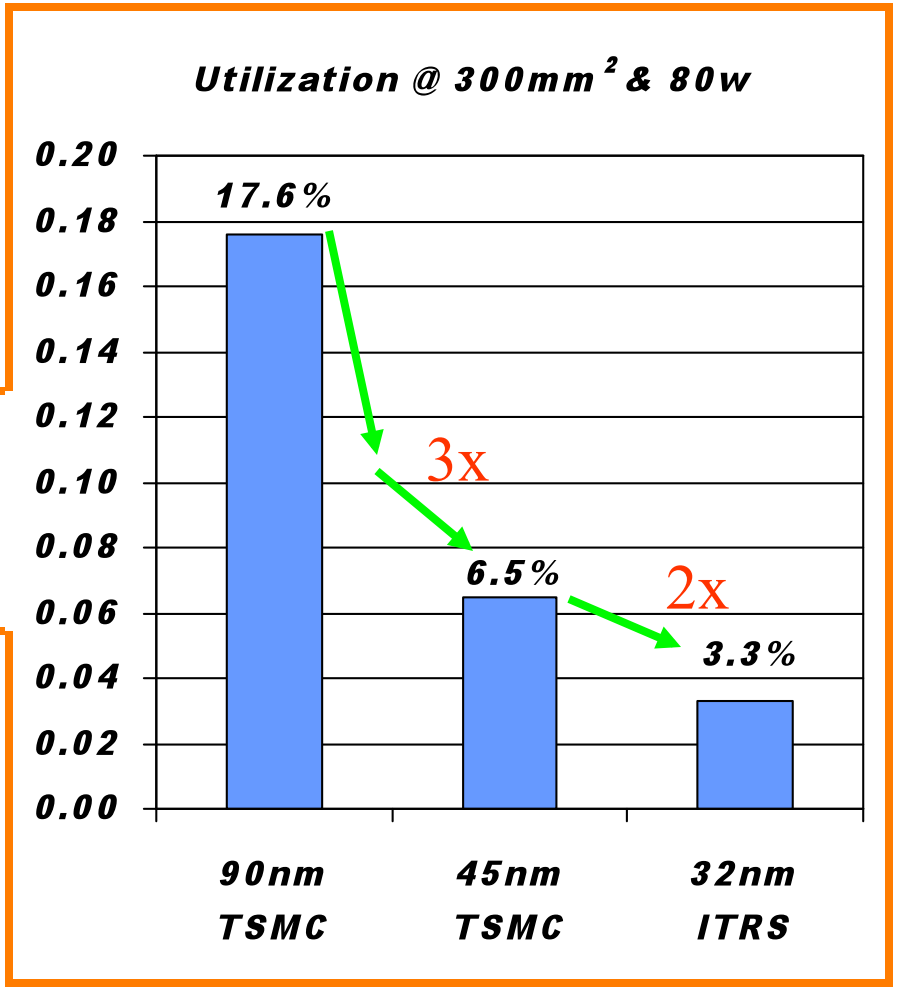
The Utilization Wall

- Scaling theory
  - Transistor and power budgets no longer balanced
  - Exponentially increasing problem!
- Experimental results
  - Replicated small datapath
  - More ‘Dark Silicon’ than active
- Observations in the wild
  - Flat frequency curve
  - “Turbo Mode”
  - Increasing cache/processor ratio



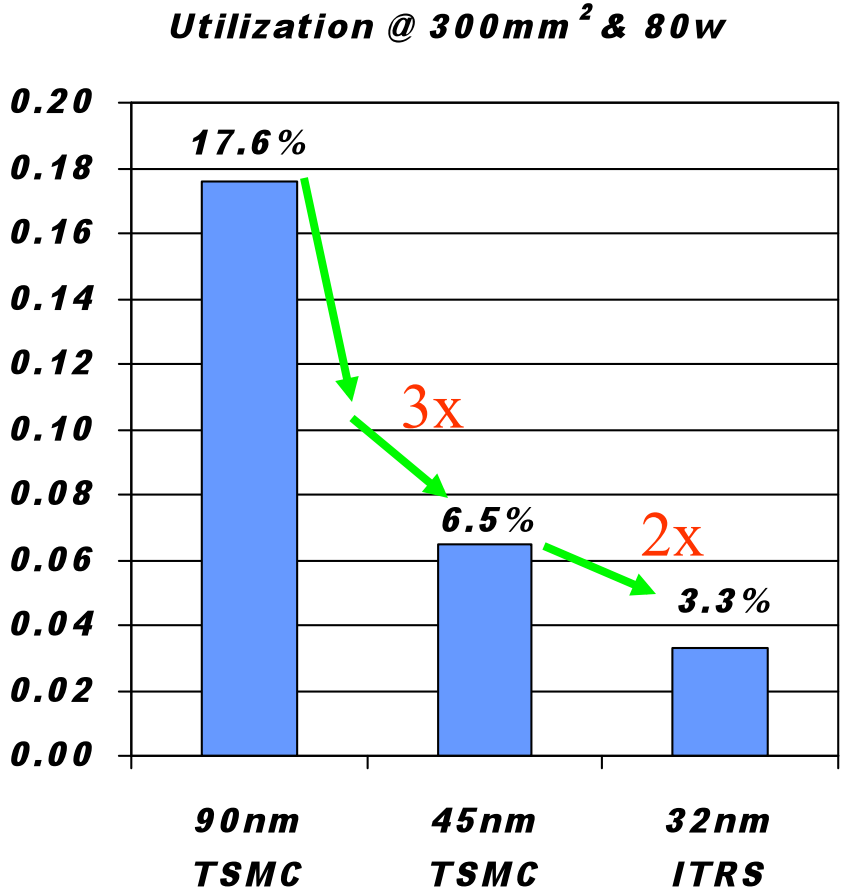
The Utilization Wall

- Scaling theory
  - Transistor and power budgets no longer balanced
  - Exponentially increasing problem!
- Experimental results
  - Replicated small datapath
  - More 'Dark Silicon' than active
- Observations in the wild
  - Flat frequency curve
  - "Turbo Mode"
  - Increasing cache/processor ratio



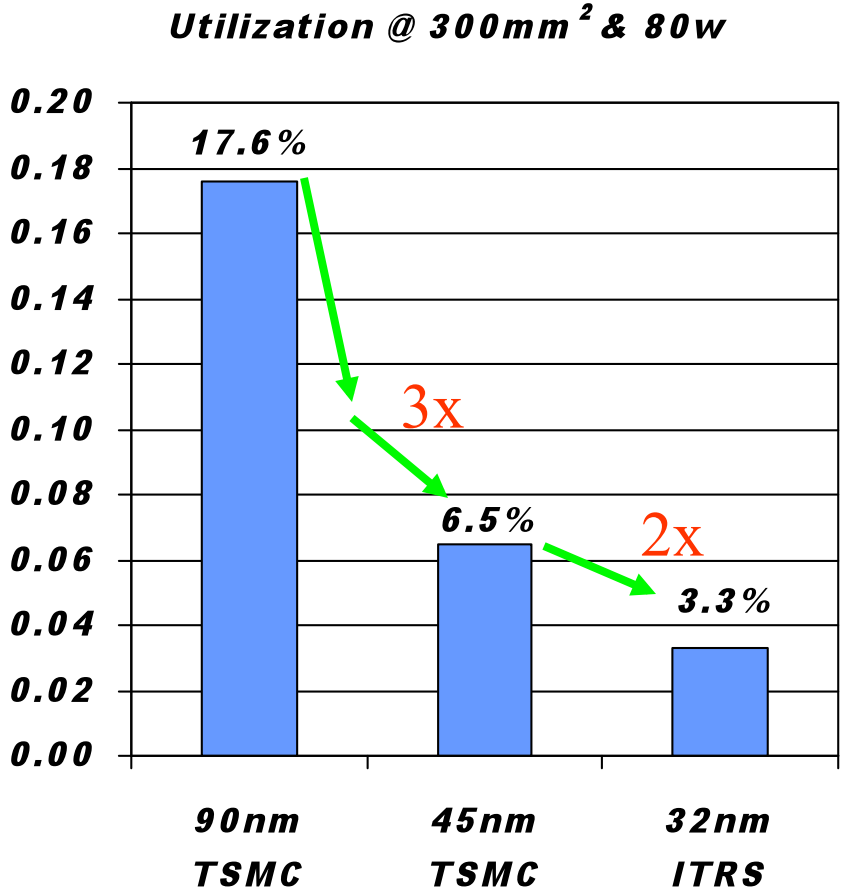
The Utilization Wall

- Scaling theory
  - Transistor and power budgets no longer balanced
  - Exponentially increasing problem!
- Experimental results
  - Replicated small datapath
  - More ‘Dark Silicon’ than active
- Observations in the wild
  - Flat frequency curve
  - “Turbo Mode”
  - Increasing cache/processor ratio



The Utilization Wall

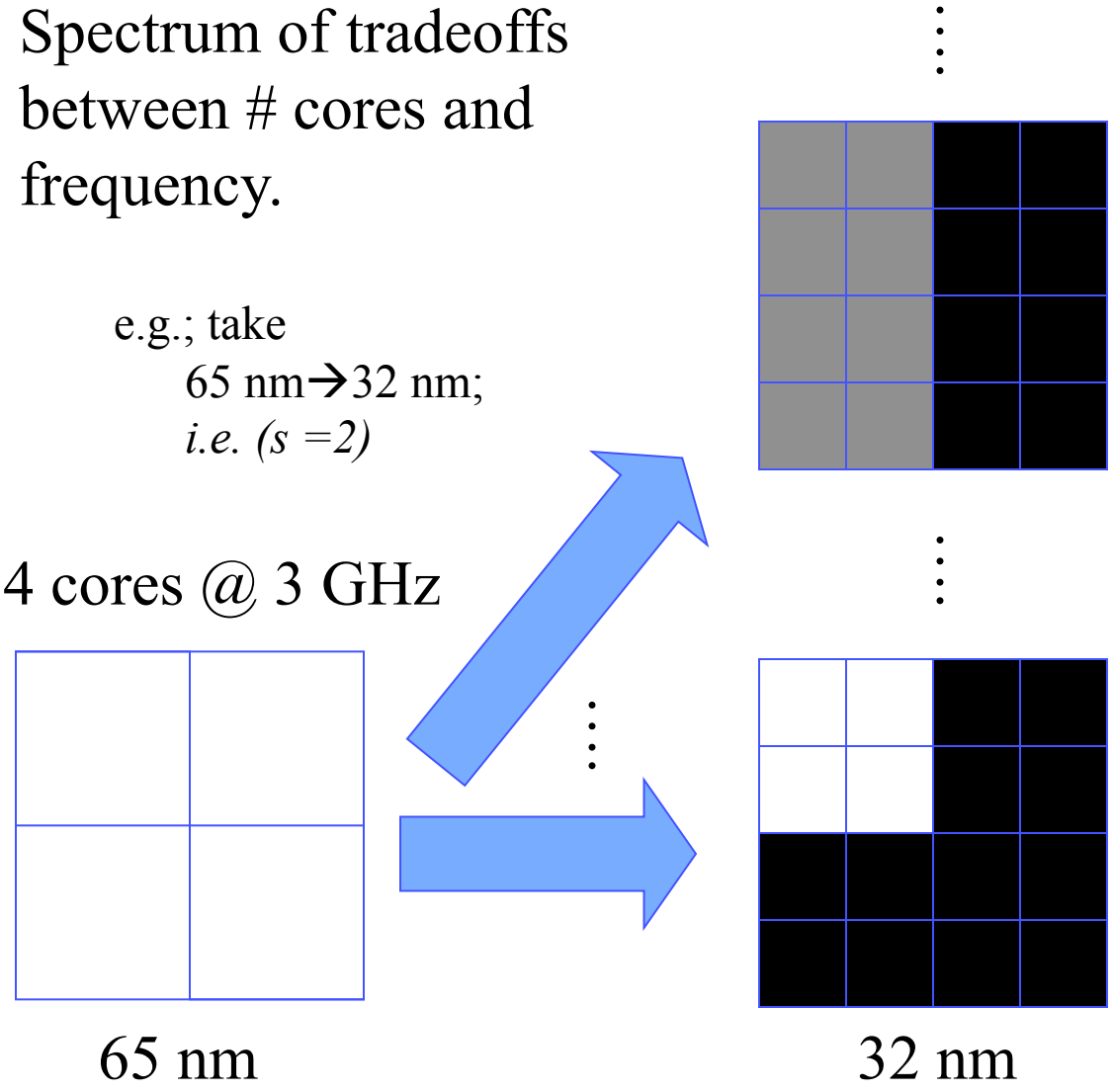
- Scaling theory
  - Transistor and power budgets no longer balanced
  - Exponentially increasing problem!
- Experimental results
  - Replicated small datapath
  - More ‘Dark Silicon’ than active
- Observations in the wild
  - Flat frequency curve
  - “Turbo Mode”
  - Increasing cache/processor ratio
- We’re already here



Utilization Wall: Dark Implications for Multicore

Spectrum of tradeoffs between # cores and frequency.

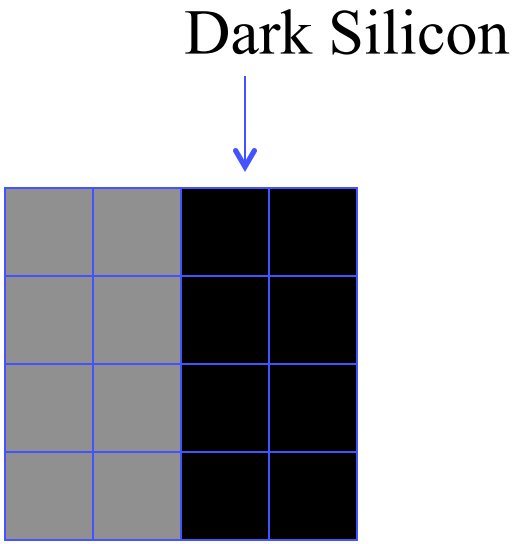
e.g.; take
65 nm→32 nm;
i.e. (s =2)



2x4 cores @ 3 GHz
(8 cores dark)
(Industry's Choice)

4 cores @ 2x3 GHz
(12 cores dark)

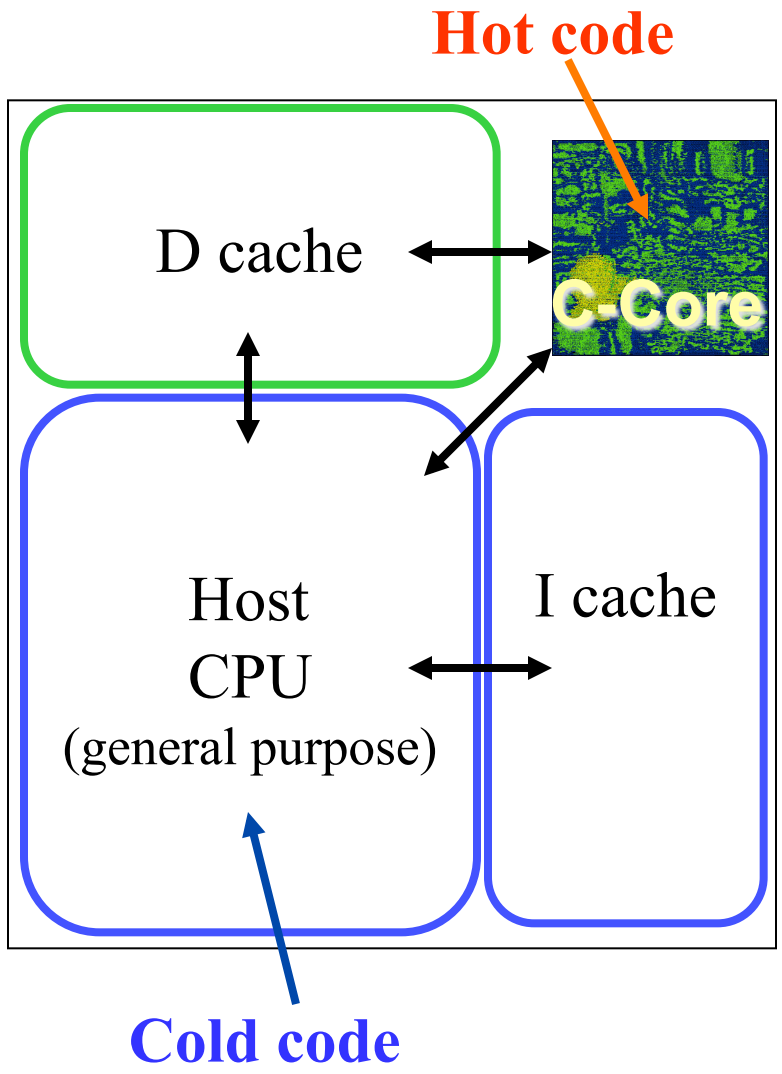
What do we do with Dark Silicon?



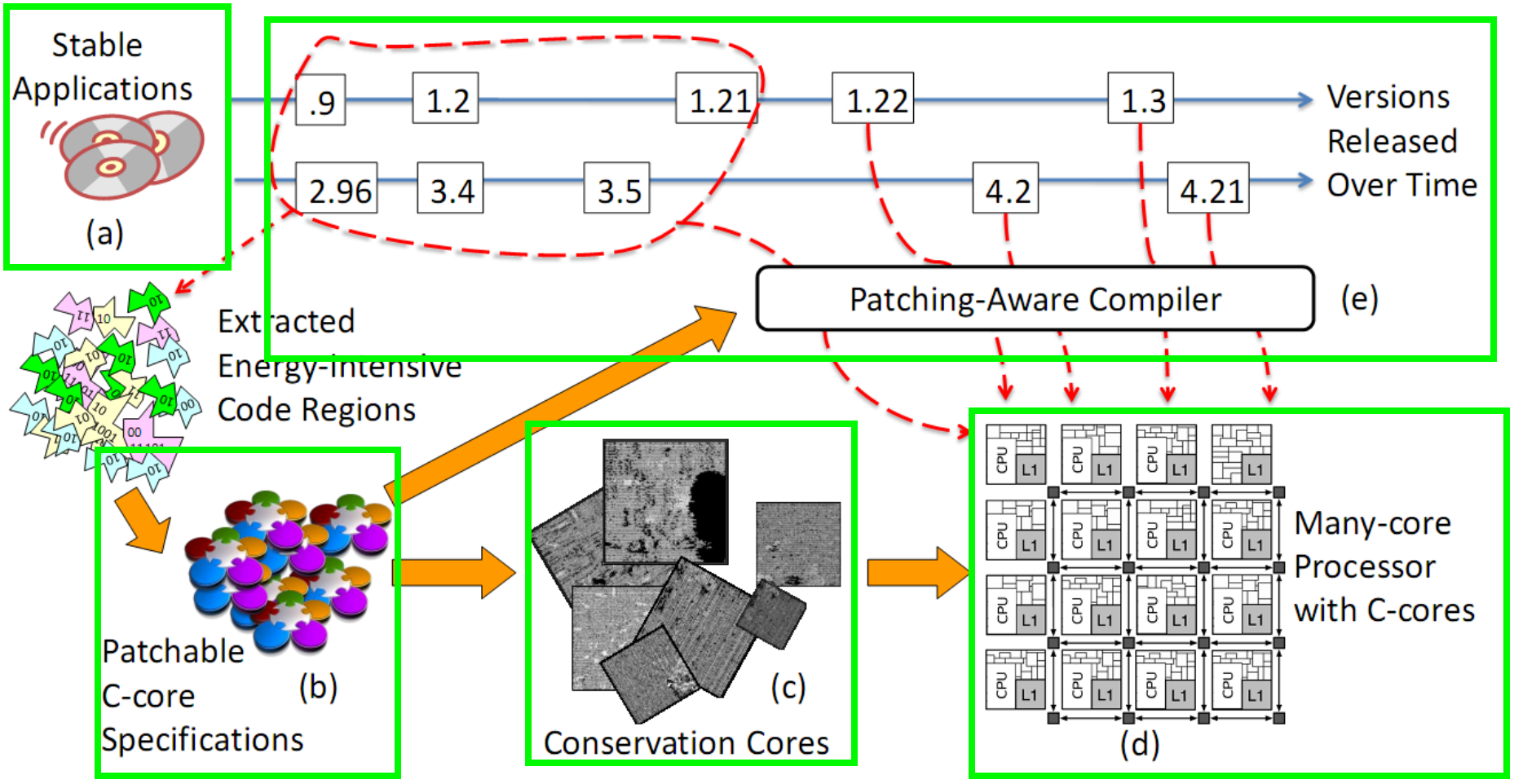
- Insights:
  - Power is now more expensive than area
  - Specialized logic has been shown as an effective way to improve energy efficiency (10-1000x)
- Our Approach:
  - Fill dark silicon with specialized cores to save energy on common apps
  - Power savings can be applied to other program, increasing throughput
- C-cores provide an architectural way to trade area for an effective increase in power budget!

Conservation Cores

- Specialized cores for reducing energy
  - Automatically generated from hot regions of program source
  - Patching support future proofs HW
- Fully automated toolchain
  - Drop-in replacements for code
  - Hot code implemented by C-Core, cold code runs on host CPU
  - HW generation/SW integration
- Energy efficient
  - Up to 16x for targeted hot code



The C-Core life cycle



Outline

- The Utilization Wall
- Conservation Core Architecture & Synthesis
- Patchable Hardware
- Results
- Conclusions

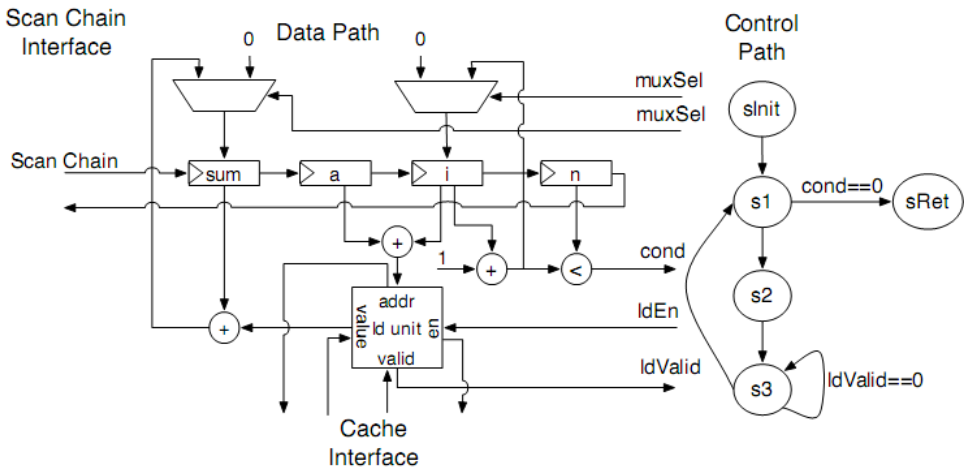
Constructing a C-Core

- C-Cores start with source code
  - Parallelism agnostic
- C code supported
  - Arbitrary memory access patterns
  - Complex control flow
  - Same cache memory model as processor
  - Function call interface

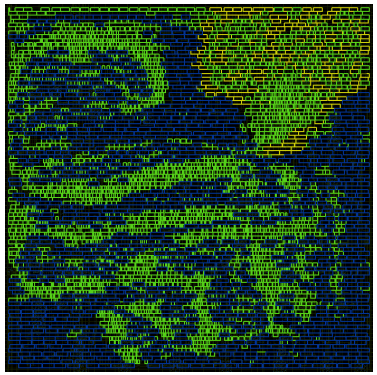
```
sumArray(int n, int *a)
{
    int i = 0;
    int sum = 0;
    for(; i<n; i++)
    {
        sum += a[i];
    }

    return(sum);
}
```

Constructing a C-Core

- Compilation
  - C-Core isolation
  - SSA, infinite register, 3-address
  - Direct mapping from CFG, DFG
  - Scan chain insertion



- Verilog to Place & Route
  - TSMC 45nm libraries
  - Synopsys CAD flow
    - Synthesis
    - Placement
    - Clock Tree Generation
    - Routing

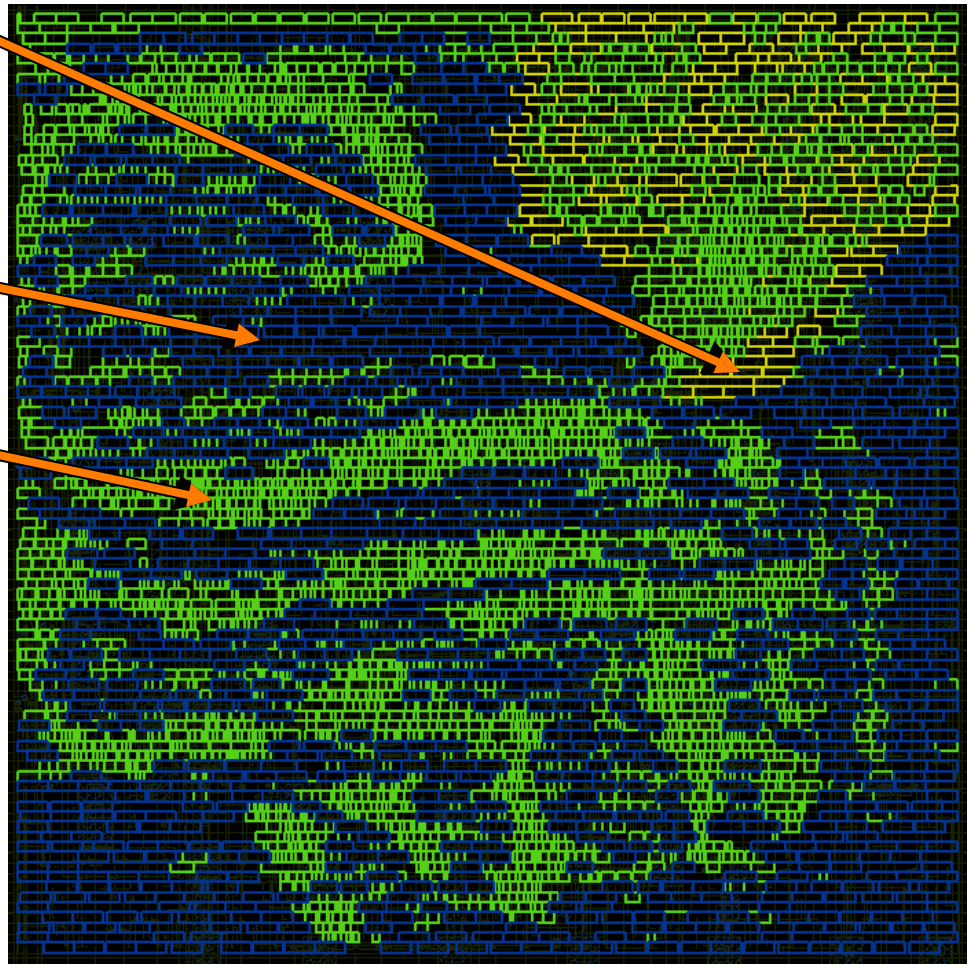


C-Core for *sumArray*

Gold – Control path

Blue – Registers

Green – Data path

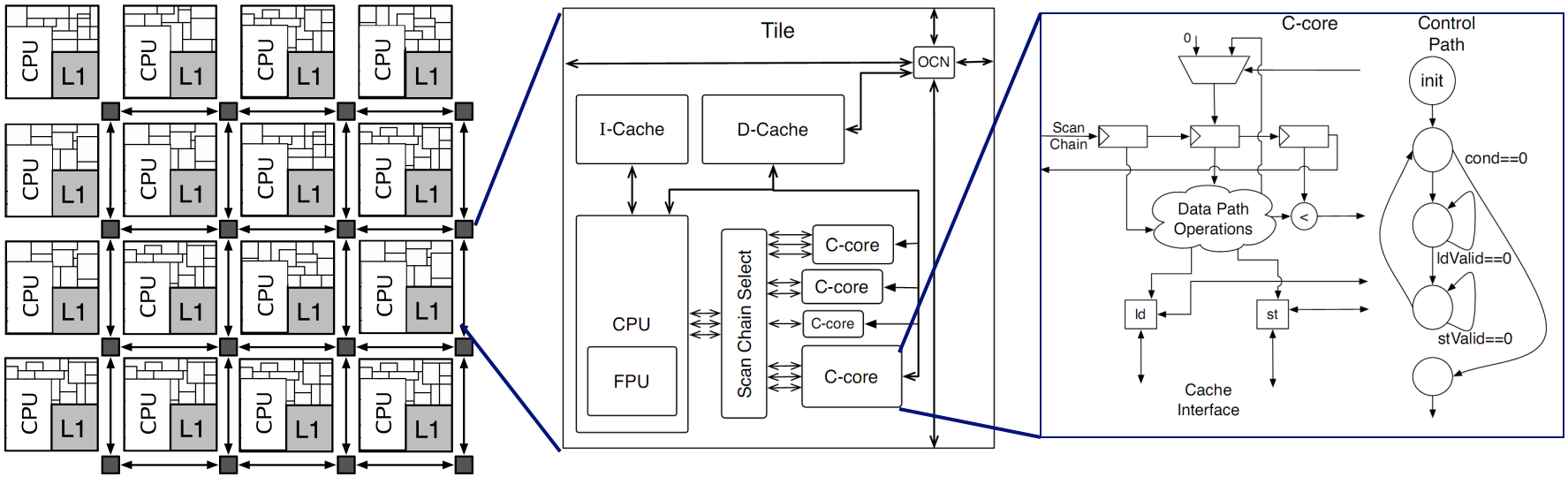


Post-route Std. Cell layout of an actual C-Core generated by our toolchain

0.01 mm^2, 1.4 GHz

A C-Core enhanced system

- Tiled multiprocessor environment
  - Homogeneous interfaces, heterogeneous resources
- Several C-Cores per tile
  - Different types of C-cores on different tiles
- Each C-Core interfaces with 8-stage MIPS core
  - Scan chains, cache as interfaces

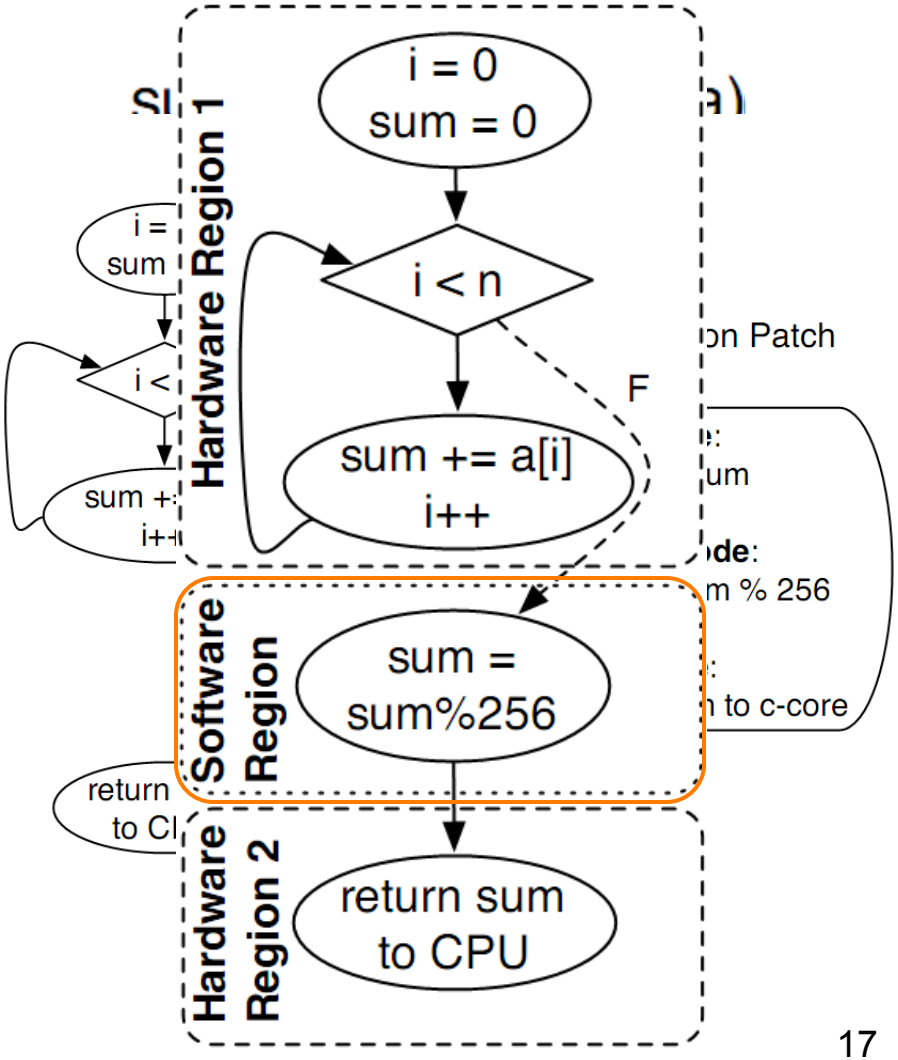


Outline

- The Utilization Wall
- Conservation Core Architecture & Synthesis
- Patchable Hardware
- Results
- Conclusions

Patchable Hardware

- Future versions of hot code regions may have changes
  - Need to keep HW usable
  - C-Cores unaffected by changes to cold regions
- General exception mechanism
  - Trap to SW
  - Can support any changes



Reducing the cost of change

- Examined versions of applications as they evolved
  - Many changes are straightforward to support
- Simple lightweight configurability
  - Preserve structure
  - Support only those changes commonly seen

Structure	Replaced by
adder subtractor	AddSub
comparator(GE)	Compare6
bitwise AND, OR, XOR	BitwiseALU
constant value	32-bit register

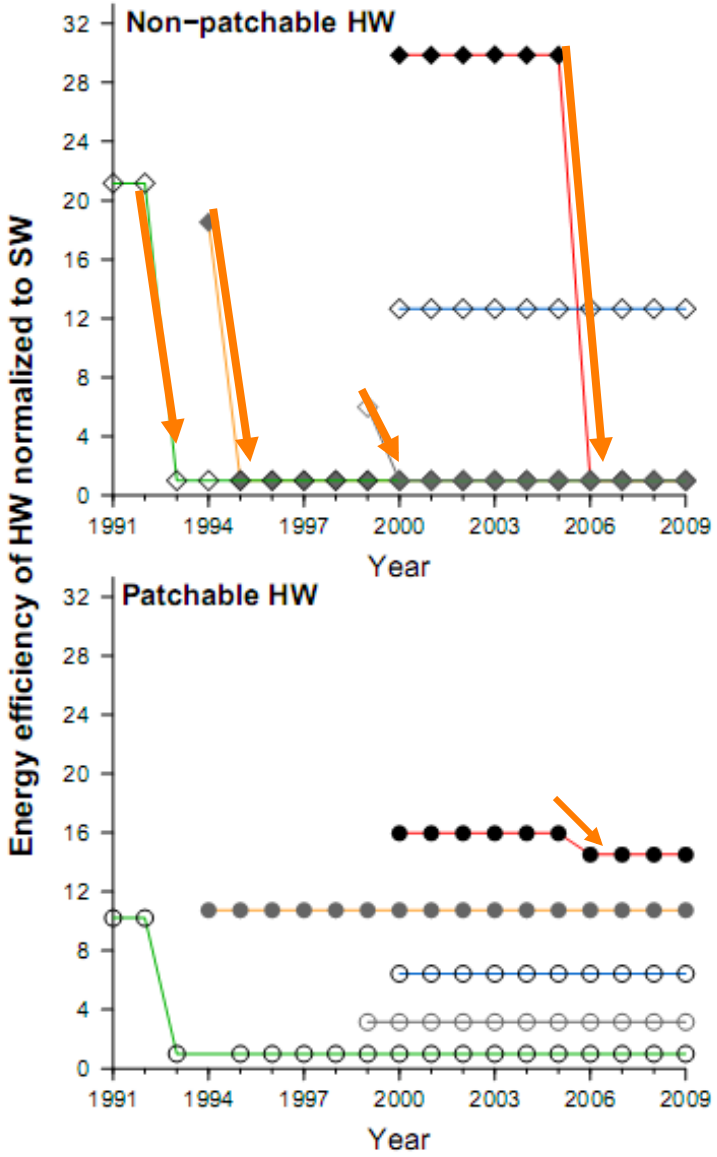
Patchability overheads

- Area overhead
  - Split between generalized datapath elements and constant registers
- Power overhead
  - 10-15% for generalized datapath elements
- Opportunity costs
  - Reduced partial evaluation
  - Can be large for multipliers, shifters

Structure	Area (μm^2)	Replaced by	Area (μm^2)
adder subtractor	270 270	AddSub	365
comparator (GE)	133	Compare6	216
bitwise AND, OR bitwise XOR	34 56	Bitwise	191
constant value	~ 0	32-bit register	160

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

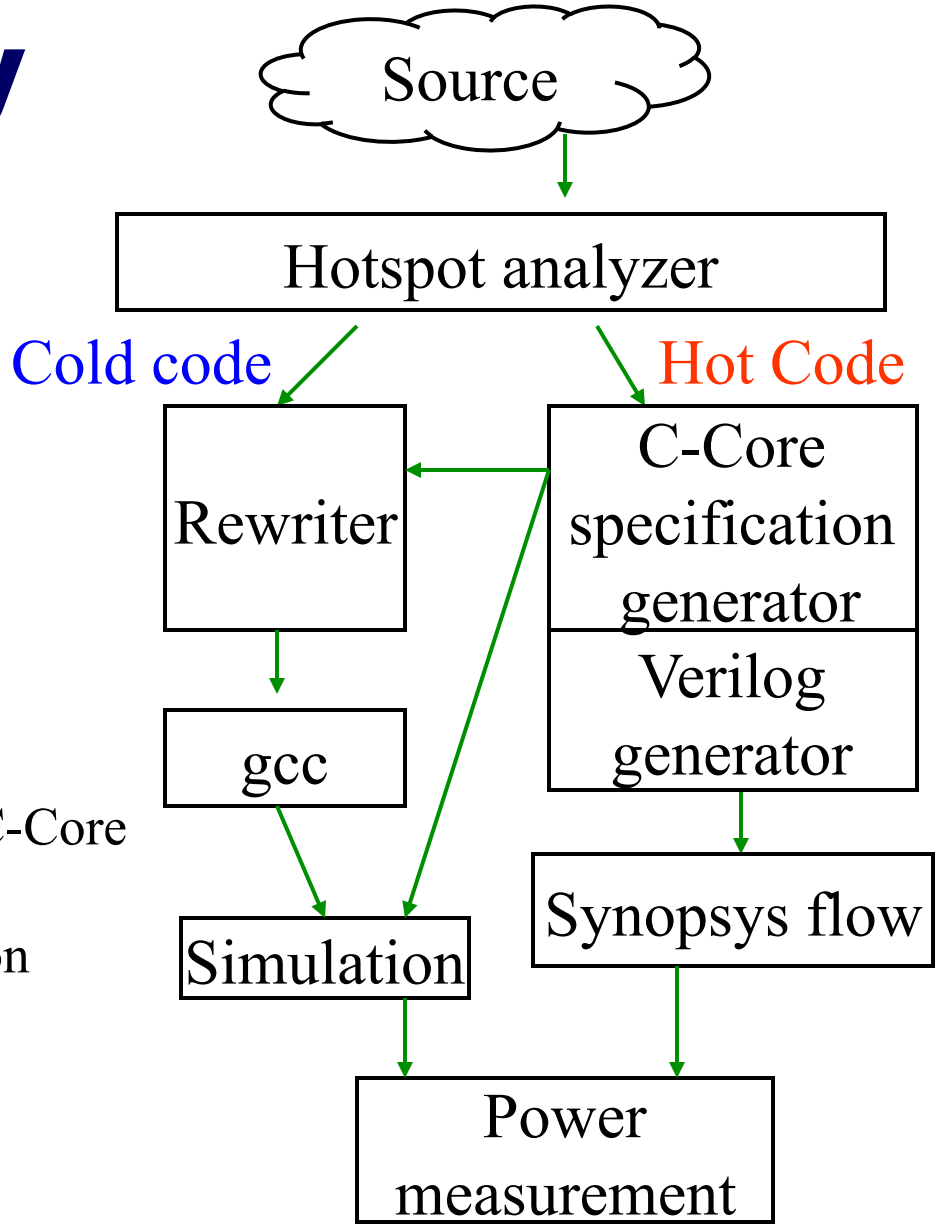


Outline

- The Utilization Wall
- Conservation Core Architecture & Synthesis
- Patchable Hardware
- Results
- Conclusions

Automated measurement methodology

- C-Core toolchain
  - Specification generator
  - Verilog generator
- Synopsys CAD flow
  - Design Compiler
  - IC Compiler
  - TSMC 45nm
- Simulation
  - Validated cycle-accurate C-Core modules
  - Post-route netlist simulation
- Power measurement
  - VCS+PrimeTime

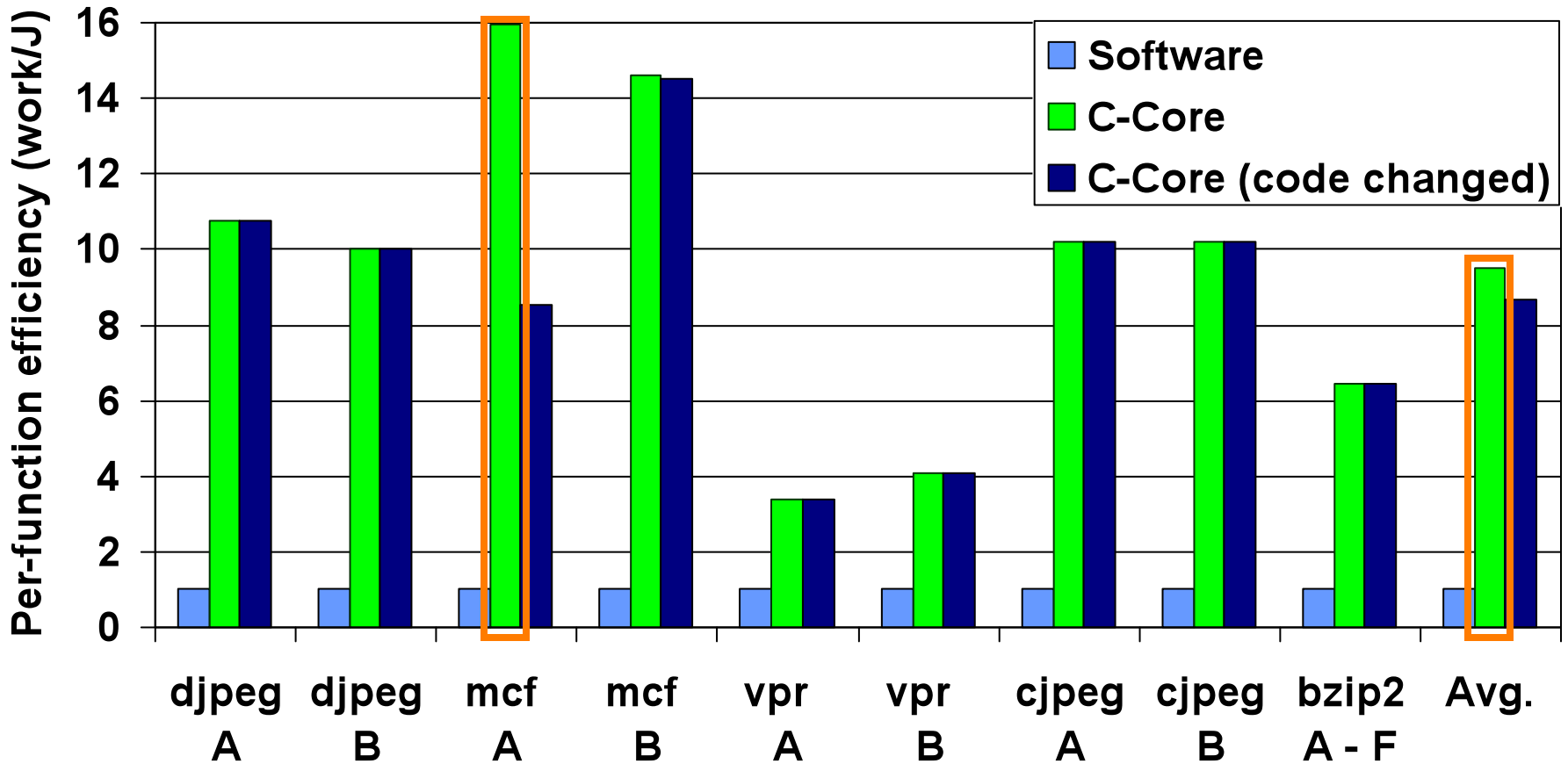


Our cadre of C-Cores

- We built 23 C-Cores for assorted versions of 5 applications
  - Both patchable and non-patchable versions of each
  - Varied in size from 0.015 to 0.326 mm^2
  - Frequencies from 0.9 to 1.9GHz

C-core	Area (mm^2) Non-P.	Patch.	Freq. (MHz) Non-P.	Patch.
bzip2				
fallbackSort	0.128	0.275	1345	1161
fallbackSort	0.128	0.275	1345	1161
cjpeg				
extract_MCUs	0.108	0.205	1556	916
get_rgb_ycc_rows	0.020	0.044	1808	1039
subsample	0.023	0.039	1651	1568
extract_MCUs	0.108	0.205	1556	916
get_rgb_ycc_rows	0.020	0.044	1808	1039
subsample	0.023	0.039	1651	1568
djpeg				
jpeg_idct_islow	0.133	0.222	1336	932
ycc_rgb_convert	0.023	0.043	1663	1539
jpeg_idct_islow	0.135	0.222	1390	932
ycc_rgb_convert	0.024	0.043	1676	1539
mcf				
primal_bea_mpp	0.033	0.077	1628	1412
refresh_potential	0.017	0.033	1899	1647
primal_bea_mpp	0.032	0.077	1568	1412
refresh_potential	0.015	0.028	1871	1639
vpr				
try_swap	0.181	0.326	1199	912
try_swap	0.181	0.326	1199	912

C-Core hot-code energy efficiency

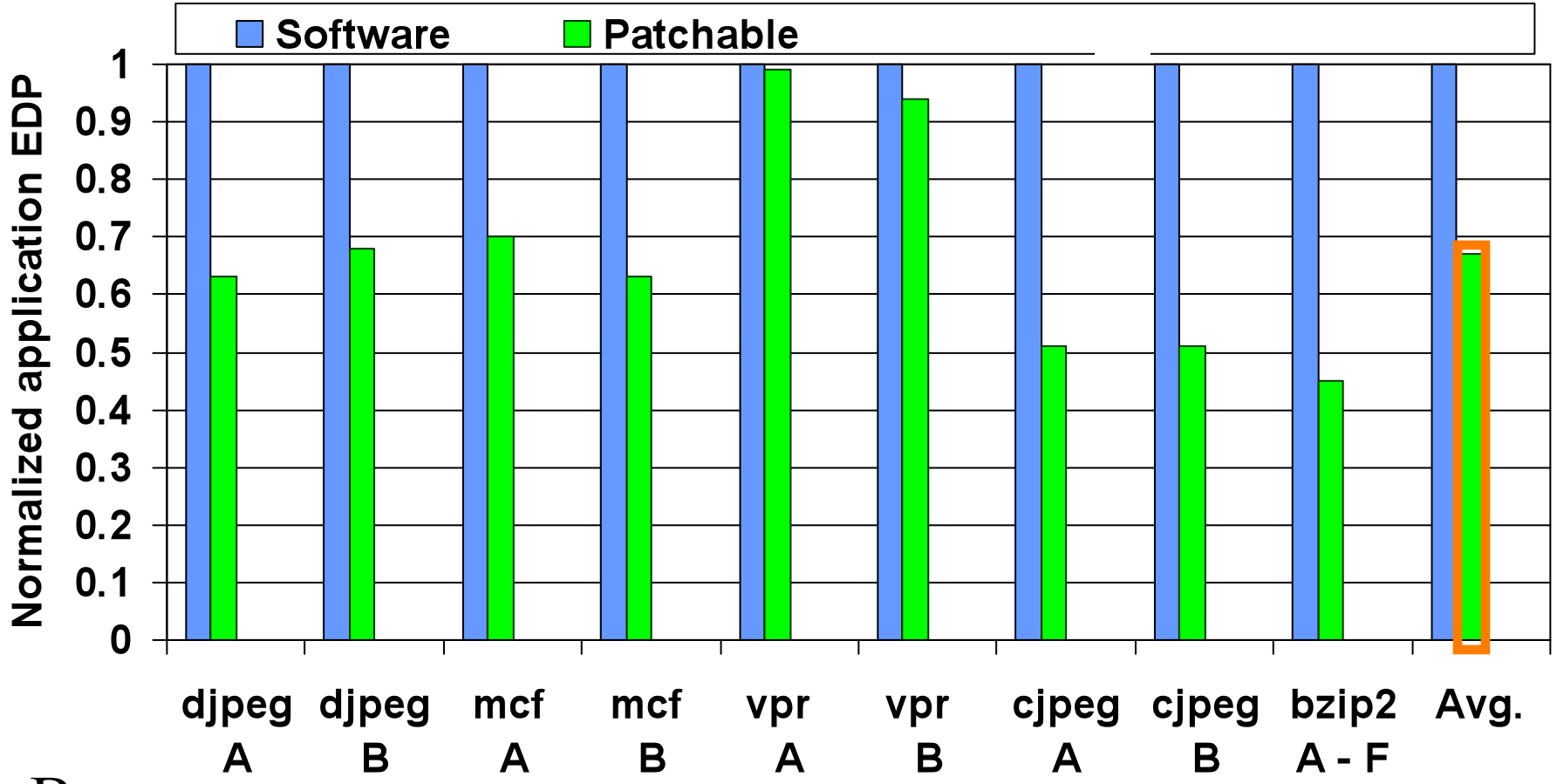


- Up to 16x as efficient as general purpose in-order core, 9.5x on average

System energy efficiency

- C-Cores very efficient for targeted hot code
- Amdahl's Law limits total system efficiency

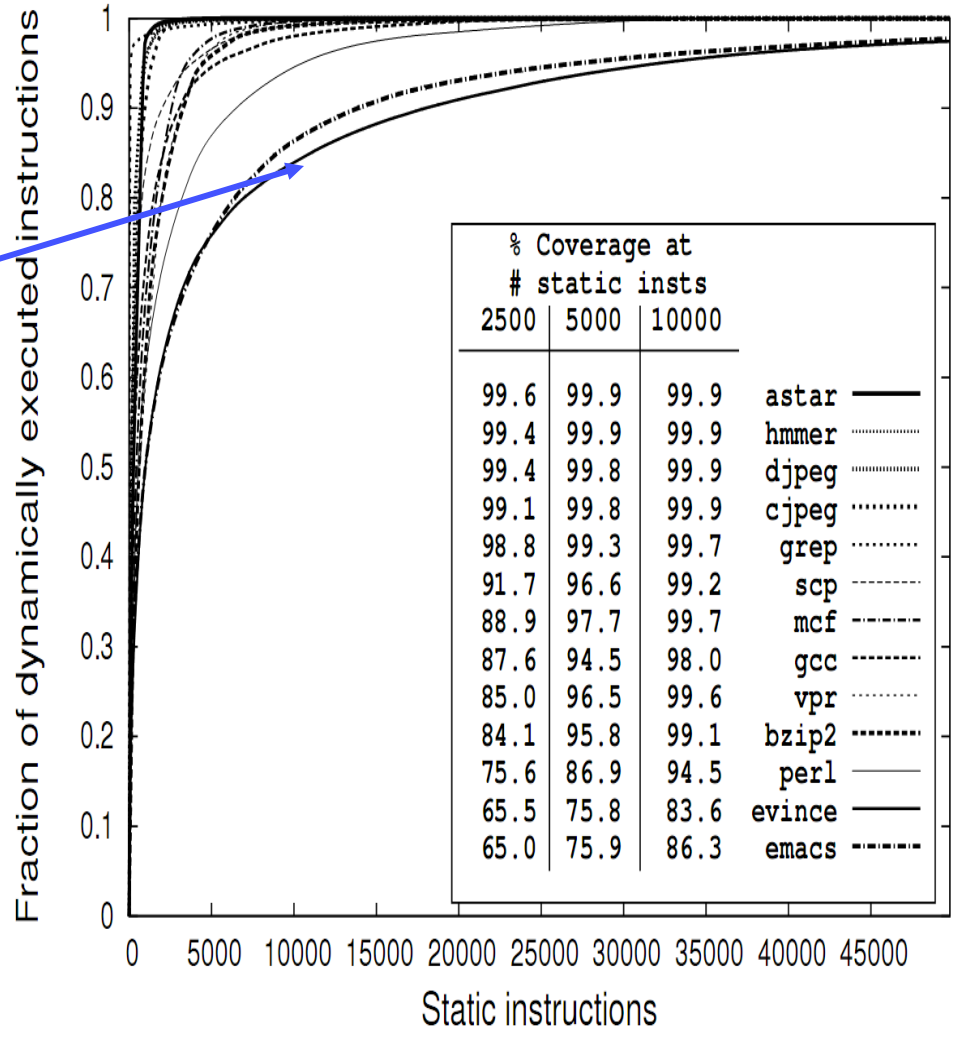
C-Core system efficiency with current toolchain



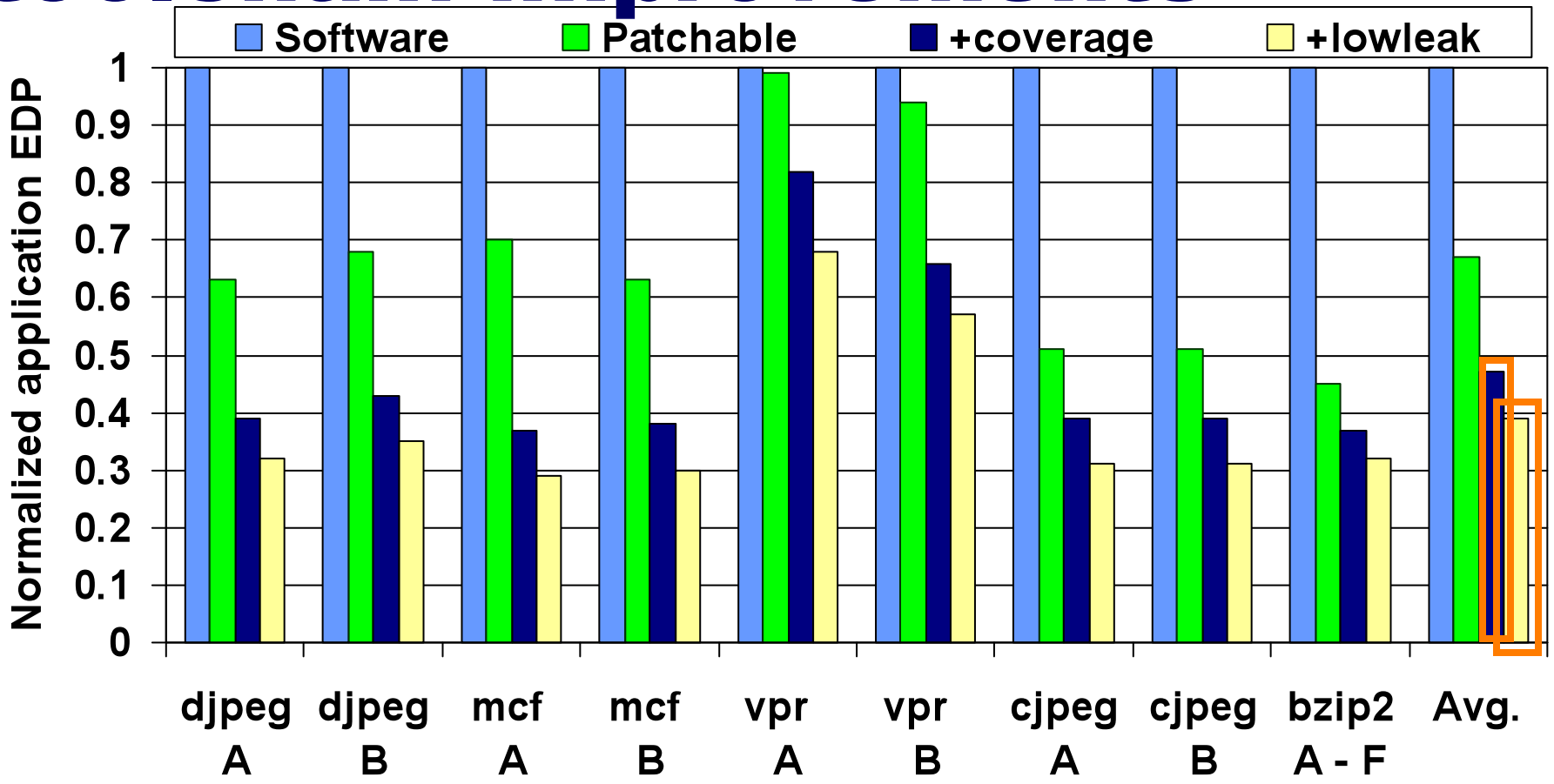
- Base
  - Avg 33% EDP improvement

Tuning system efficiency

- Improving our toolchain's coverage of hot code regions
  - Good news: Small numbers of static instructions account for most of execution
- System rebalancing for cold-code execution
  - Improve performance/leakage trade-offs for host core



C-Core system efficiency with toolchain improvements



- With improved coverage
  - Avg 61% EDP savings
  - Avg 14% increased execution time

With improved coverage system components
- Avg 53% EDP improvement

Conclusions

- The Utilization Wall will change how we build hardware
  - Hardware specialization increasingly promising
- ***Conservation Cores*** are a promising way to attack the Utilization Wall
  - Automatically generated patchable hardware
  - For hot code regions: 3.4 – 16x energy efficiency
  - With tuning: 61% application EDP savings across system
  - 45nm tiled C-Core prototype under development @ UCSD
- Patchability allows C-Cores to last for ten years
  - Lasts the expected lifetime of a typical chip