Parallel Cross-Layer Optimization of High-Level Synthesis and Physical Design

James Williamson, Li Shang (University of Colorado, Boulder)
Yinghai Lu, Hai Zhou (Northwestern University)
Xuan Zeng (Fudan University)

January 27, 2011

Outline

Problem Introduction

Modern VLSI Design Flow

- Modern VLSI design follows a top-down hierarchical flow
- Design abstraction decreases throughout the flow
- Design layers are addressed sequentially
- Each design layer guides the next

Problem Introduction

The Design Closure Problem

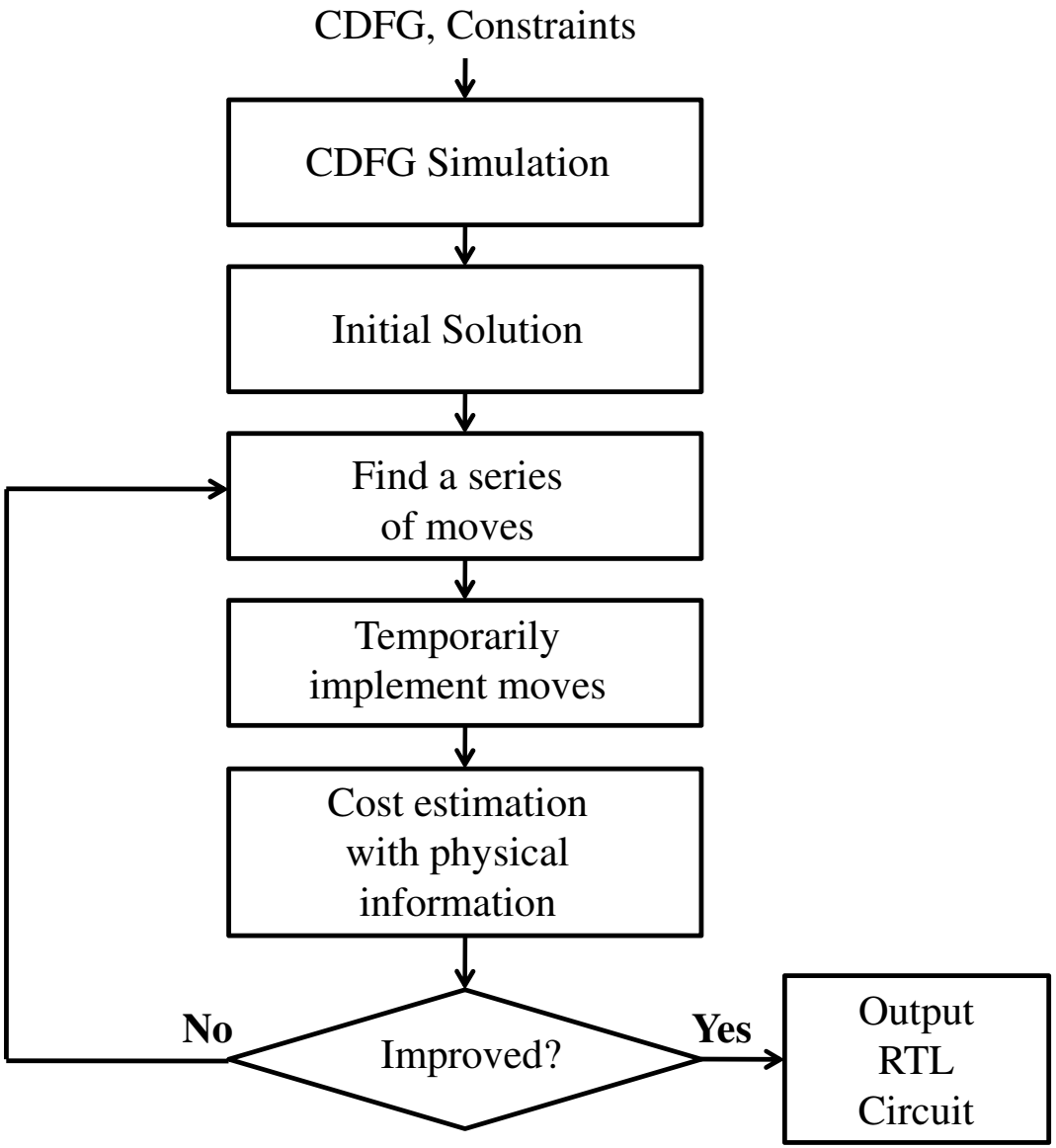
- High-level abstractions can cause incorrect decision making
  - Large portions of the design space are irreversibly pruned
  - Errors found when physical information (interconnect, area) becomes known, i.e. too late
  - Entire design flow repetition is required to ensure design closure – costly!
- The outlook is not good
  - Growing transistor counts, technology scaling, etc. ensure design closure will become increasingly difficult

Sequential Incremental Synthesis

Incremental Floorplanning and High-Level Synthesis

- *ISCALP*, Zhong et al, 2002.
- Computes physical information to incrementally guide high-level synthesis
- Iterative approach is better, but takes too long to compute
- True parallelism between layers not exposed

Sequential Incremental Synthesis Flow



Sequential Incremental Synthesis Run-Time Ratio

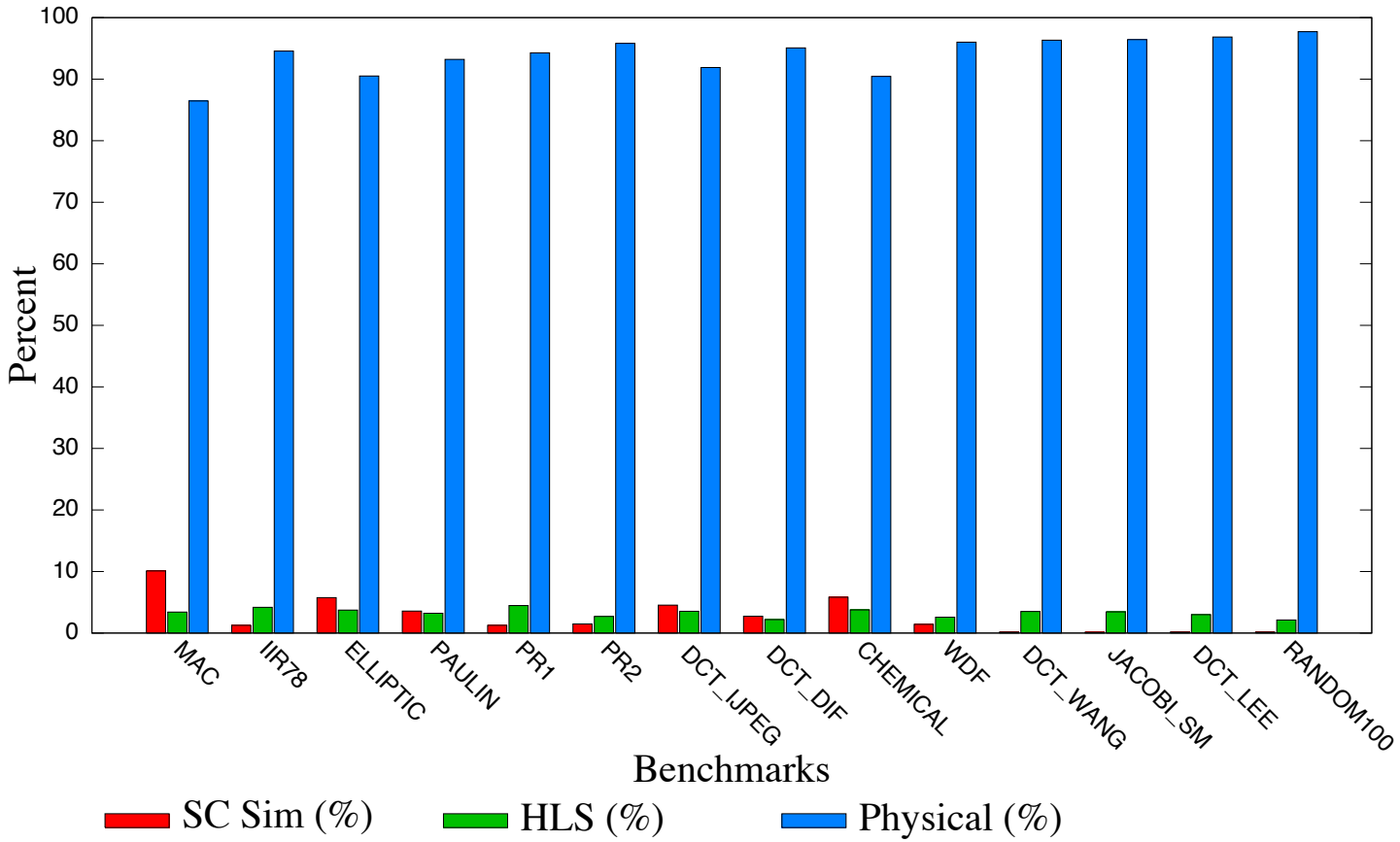


Figure: Computation run-time ratio of sequential incremental synthesis

Breaking the Sequential Flow

Finding Inter-Design Layer Parallelism

- Traditional top-down hierarchical flow needs to be broken
  - Needs vertical integration of layers, holistic approach
- But, how?
  - Parallel programming is very difficult!
- How do we guide design layer decision making under concurrent layer execution?

Research Goals and Contributions

Goals

- Vertically integrate design layers through holistic approach
- Suite heterogeneous characteristics of design layers
- Mitigate the design closure problem

Contributions

- First work for parallel cross-layer optimization
- Leverage parallel heterogeneous power of CPU/GPU
- Novel GPU floorplanner achieves 24% speedup
- Overall 11X on average speed-up over state-of-the-art

Target Design Layers

High-Level Synthesis

- Operations: Rebinding/Merging/Splitting
- Optimizations: Low power
- High sequential control flow dependencies
- Coarse data granularity, low data parallelism

Physical Design

- Operations: Floorplanning, interconnect cost evaluation
- Optimizations: Low power
- Low sequential control flow dependencies
- Fine data granularity, high data parallelism

Proposed Solution

Parallel Cross-Layer Optimization

- Parallelism breaks design layer boundaries, speeds computation
- Nondeterminism guides layer execution and communication
- Cross-layer communication enables real-time design optimization and error correction

Heterogeneous Compute Architecture Mapping

- As a first work, two design layers demonstrate the approach
- HLS $\rightarrow$ Deeply pipelined superscalar CPU
- Floorplanning $\rightarrow$ Massively parallel SIMT GPU

Parallel Incremental Algorithm Flow

Parallel Design Flow

- Nondeterministic transactional model guides layer interactions
- We generate and explore set of (V_{dd}, C_s) pairs, P, for a design
- High-level synthesis attempts moves $\forall p \in P$ configurations
- GPU collectively finds physical impact of $|P|$ high-level moves
- Moves satisfying both levels of abstraction are kept

Generate and initialize P and set *best* as ϕ

loop

Transaction A: *High-Level Synthesis Layer*

$\exists p \in P : p.flag = SYN$

Generate HLS move

if move succeeded $\rightarrow p.flag := PHY$, **else** $p.flag := BRK$

Transaction B: *Physical Design Layer*

$\exists p \in P : p.flag = PHY$

Do floorplan for p, $p.flag := EVL$

Transaction C: *Cost Evaluation (Power and Area)*

$\exists p \in P : p.flag = EVL$

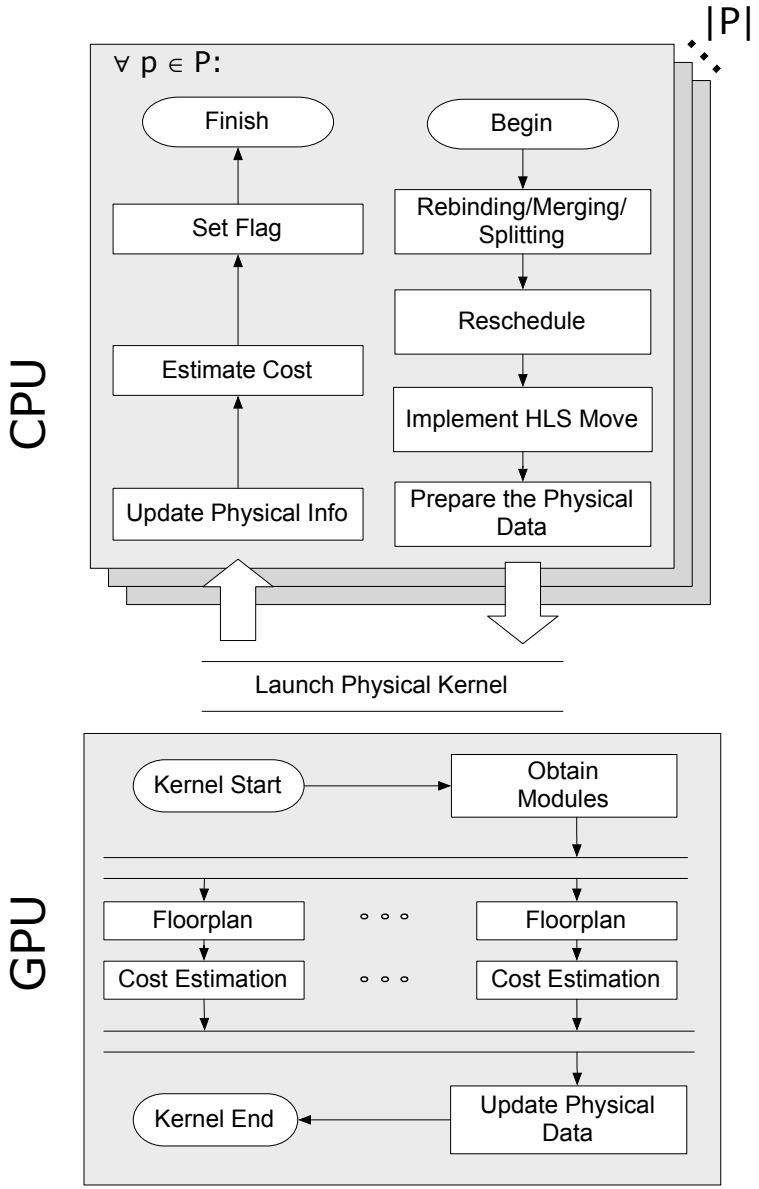
Evaluate design cost

if improved $\rightarrow p.flag := SYN$, **else** $p.flag := BRK$

end loop

Output p with the lowest cost in P as *best*

Design Flow Mapping to Heterogeneous Hardware

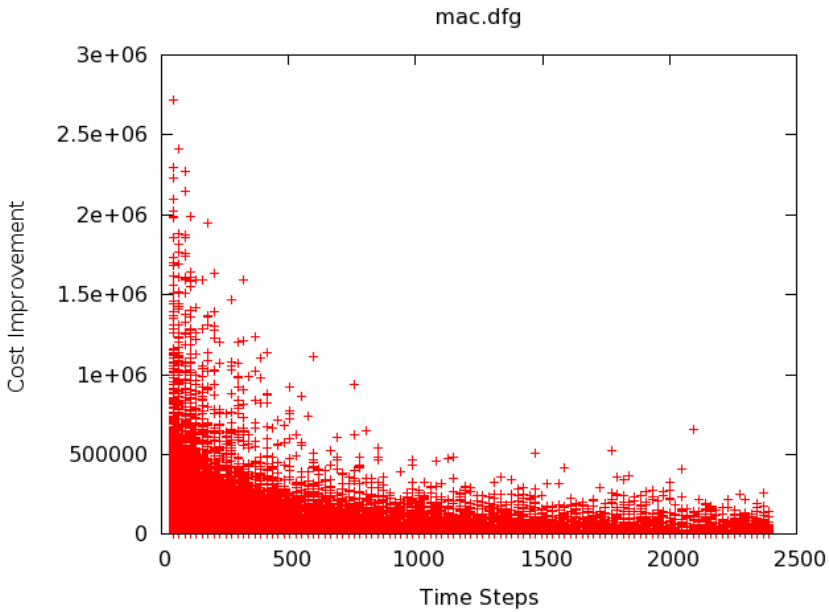


Physical Design on GPU

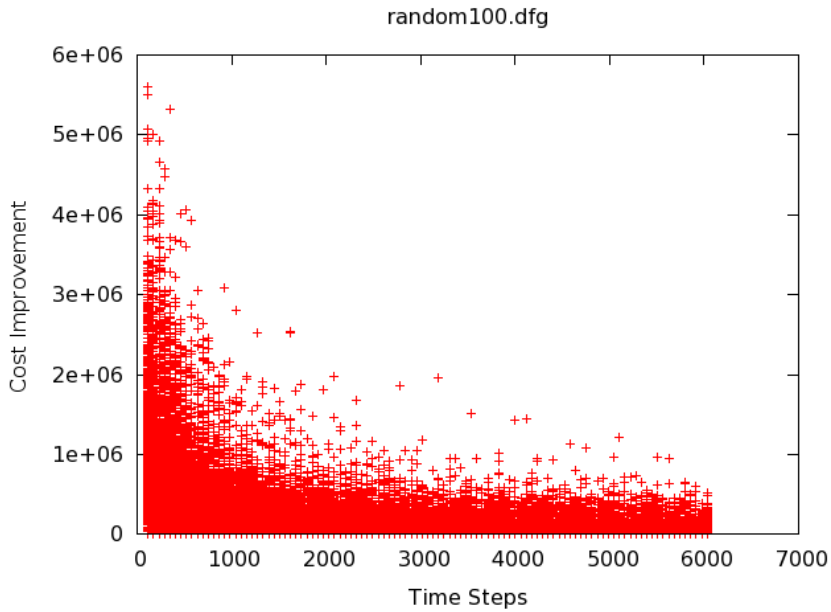
GPU Simulated Annealing-driven Floorplanning

- Up to 4 GPUs concurrently compute thousands of multiple candidate floorplans using SIMT architecture
- Each GPU thread computes a single $p \in P$ floorplan
- **Problem:** GPU kernels are atomic, but different designs require more or less time to reach convergence
- **Solution:** Dynamically maximize design quality and minimize run-time cost using thread- and kernel-level convergence testing

Floorplanning Convergence Points



MAC benchmark:
Convergence point found at
500 iterations



RANDOM100 benchmark:
Convergence point found at
2000 iterations

Convergence Points across Configurations

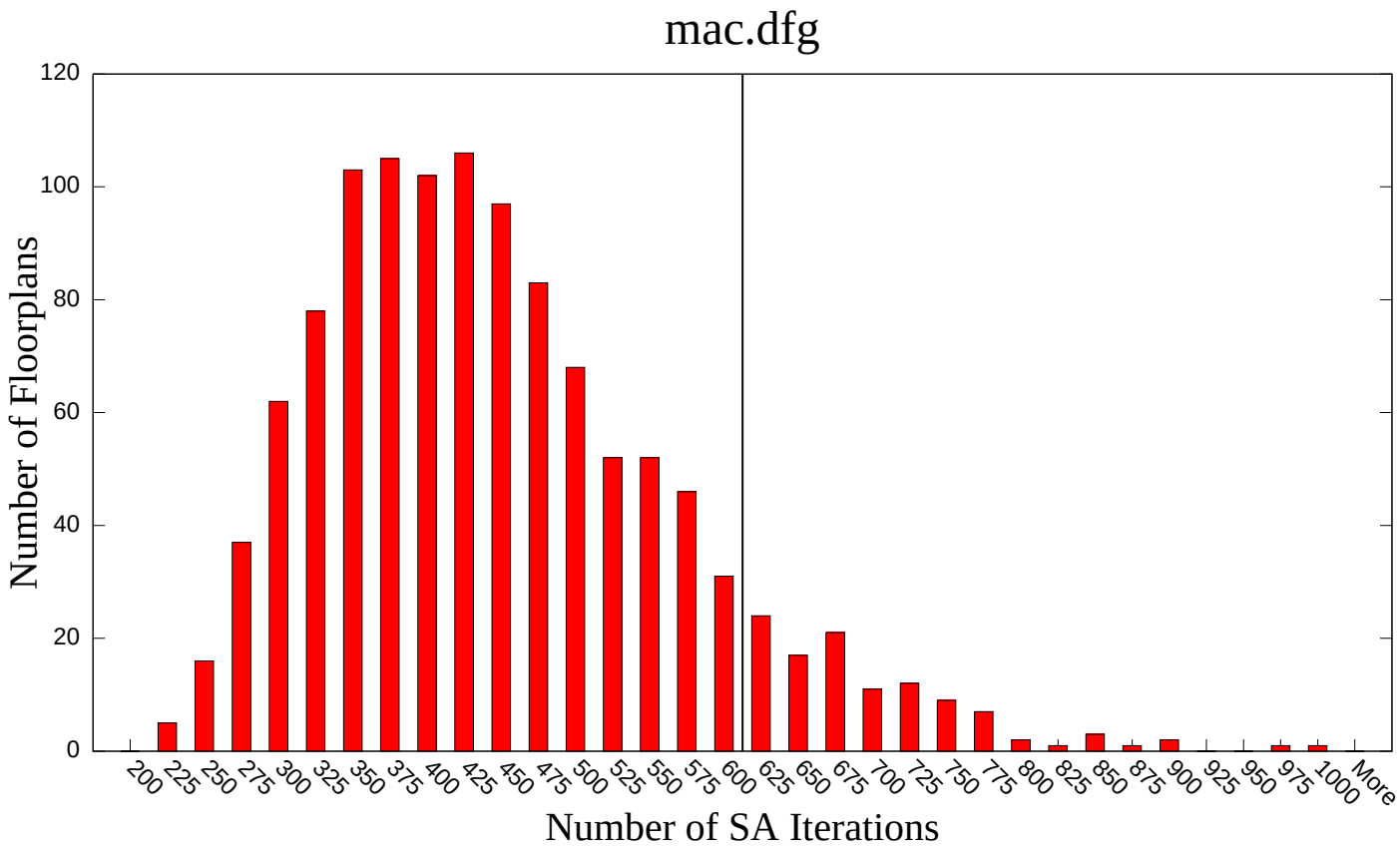


Figure: Detected convergence points for 1,155 MAC configurations; 90% of configurations lie left of the vertical line

Proposed GPU Floorplanning

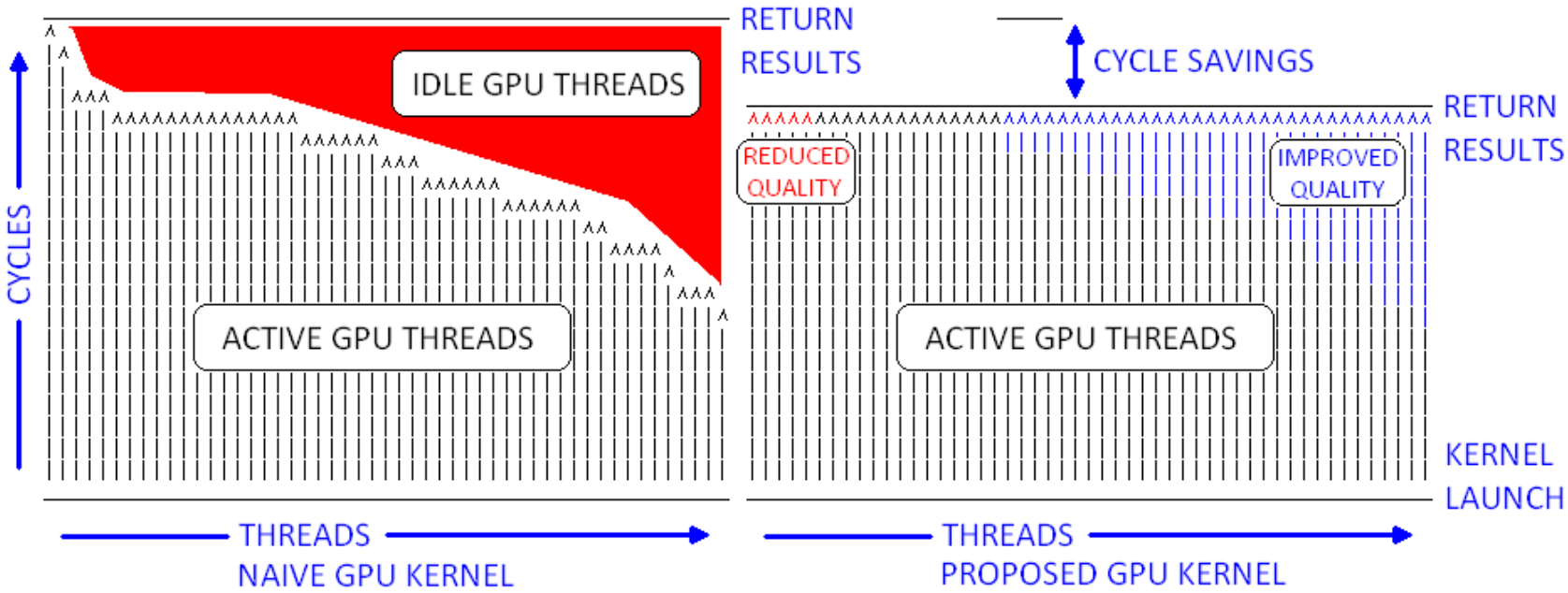


Figure: GPU kernels with and without convergence testing

CUDA Floorplanning Algorithm

if $threadIdx = 0$ **then**
  $counter_{GLOBAL} := 0$
loop
  Generate floorplan *move*, evaluate $move_{COST}$
  if $flag_{CONV} = false$ **then**
    Update *HISTORY* list, update $move_{BEST}$ if appropriate

    Convergence Test 1: *Thread Level*
    if $\forall moves \in HISTORY$, $\leq 5\%$ improve cost **and**
      $\forall$ improved moves, $\leq 10\% \times move_{BEST}$ **then**
      increment $counter_{GLOBAL}$, $flag_{CONV} := true$

  Convergence Test 2: *Kernel Level*
  if $counter_{GLOBAL} \geq 90\% \times |P|$ **then**
    return

end loop

Experimental Setup

- Workstation
  - Intel quad-core Nehalem 2.13GHz processor
  - 4GB memory
  - Quad Nvidia Tesla C1060 GPUs
- ISCALP modified based on nondeterministic transactional model and convergence-aware GPU floorplanner
- 14 0.25 μm-technology benchmarks tested

GPU Floorplanner Run-Time Speed-Up and Result Quality

Table: GPU Floorplanning: Traditional SA vs. Proposed GPU SA

Benchmark	Traditional SA		Proposed GPU SA		Improvement	
	Time	Energy	Time	Energy	Speedup	Energy (%)
MAC	17.84	2214.56	15.75	2215.20	1.13X	100.02
IIR77	67.10	3422.75	51.63	3433.46	1.29X	100.31
ELLIPTIC	32.52	2837.97	26.60	2847.84	1.22X	100.34
PAULIN	20.58	1242.33	19.61	1241.98	1.04X	99.97
PR1	51.93	2693.80	42.43	2708.64	1.22X	100.55
PR2	83.64	4029.71	63.11	4019.39	1.32X	99.74
DCT_IJPEG	53.64	2925.21	41.39	2916.09	1.29X	99.69
DCT_DIF	58.78	2222.45	47.54	2219.13	1.23X	99.85
CHEMICAL	35.21	2592.33	29.16	2593.79	1.20X	100.06
WDF	75.17	2301.84	60.91	2304.28	1.23X	100.11
DCT_WANG	83.45	1820.14	63.70	1837.53	1.31X	100.96
JACOBI_SM	107.15	3646.65	78.85	3661.61	1.35X	100.41
DCT_LEE	99.44	3061.91	78.84	3067.20	1.26X	100.17
RANDOM100	141.16	3715.22	107.78	3683.77	1.30X	99.15
Avg.					**1.24X**	**100.09**

Parallel Cross-Layer Run-Time Speed-Up

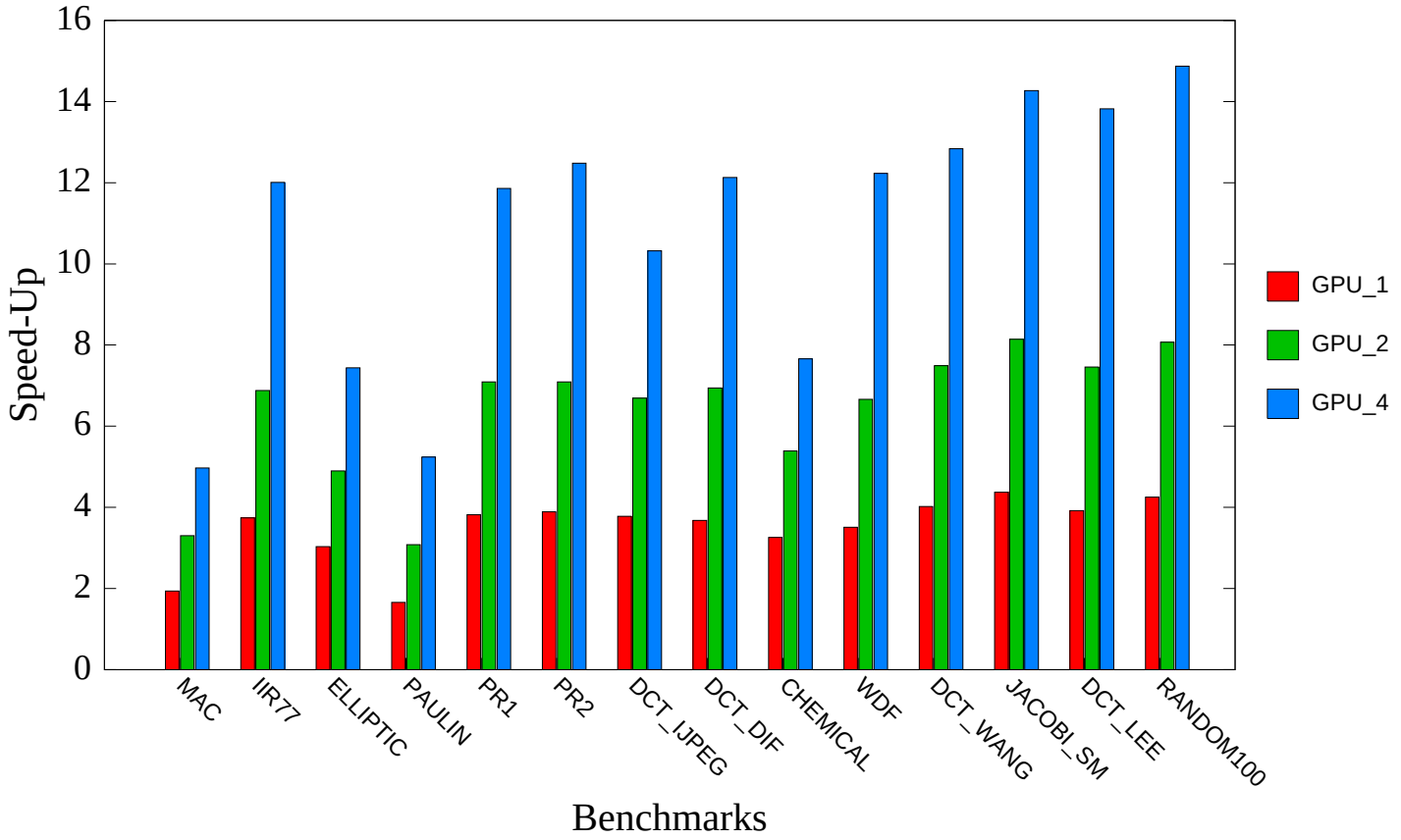


Figure: Speed-up of parallel cross-layer optimization approach, while maintaining comparable result quality

Summary

- Design closure is an increasingly difficult issue
- Parallelism between design layers needs exploitation
- Optimizations made across design layers must agree
- Parallel Cross-Layer Optimization
  - Broken boundaries between layers vertically integrates flow
  - Nondeterministic transactions enable design layer concurrency
  - Heterogeneous architectures mate well across design layers
  - GPU floorplanner dynamically optimizes run-time vs. quality