

Tool Development for Multi-Million Gate Designs

Jarrold A. Roy, David A. Papa, James F. Lu,
Aaron N. Ng, Igor L. Markov

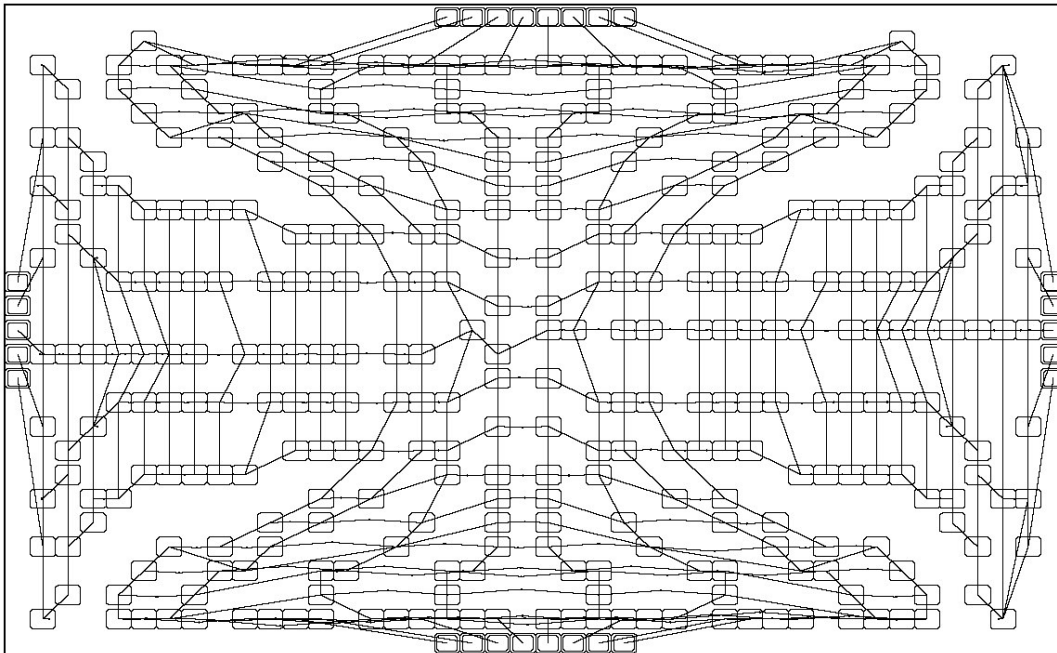
EECS Department
University of Michigan, Ann Arbor

Introduction

- Scalable algorithm design is important
- Runtime and memory requirements must be near-linear
- EDA tool development must be modular to handle large designs
- New practical problems arise with increasing design size
- Logistics of testing and debugging is more involved
- Need automated diagnostics

VLSICAD Placement

- Produce non-overlapping locations for gates
- Minimize wirelength to improve design performance
- Tool examples come from experience with the physical design tool CAPO



Motivation

- Tools need to work on large designs
 - Recent released designs have 2.5 million movable objects
- Practical problems arise at large scales
 - Memory requirements can exhaust 32-bit address space
 - Visualizing, Testing, and Diagnosing problems
- EDA is a fast moving field
 - Requirements constantly change

ISPD 2005

IBM Placement Contest

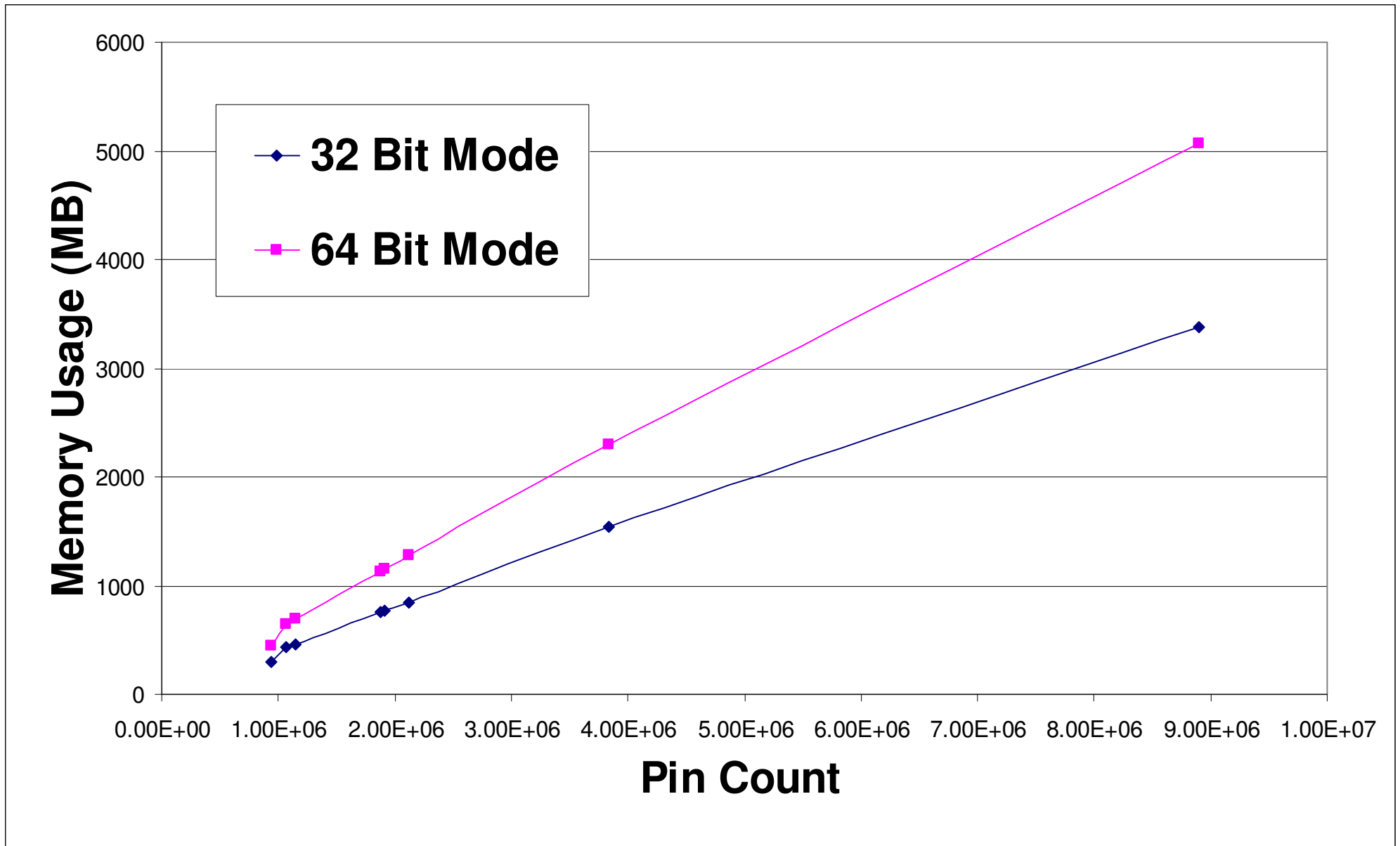
Circuit	# Mov. Objects	# Fixed Objects	# Nets	# Pins	Design Util.
Adaptec 2	254,457	566	266,009	1,069,482	44.32%
Adaptec 4	496,045	1329	515,951	1,912,420	27.23%
Big Blue 1	450,927	723	466,758	1,875,039	33.66%
Big Blue 2	534,782	1329	515,951	1,912,420	27.23%
Big Blue 3	1,095,519	1293	1,123,170	3,833,218	56.68%
Big Blue 4	2,169,183	8170	2,229,886	8,900,078	44.35%

Scaling to 64-bits

- Need 64-bit address space for large designs
 - 32 bits fundamentally limits address space to 4GB
- Memory usage increases in practice
 - Code using pointers *could* double memory requirements
 - We observe approximately 1.5x increase in Capo
- Need 64 bit portable code
 - The following code was observed in a real program!

```
double **d = new int [size]; // Error: works in 32 but unsafe in 64 bits
```
- But, 64 bit operation is faster (~10% observed)

Capo Memory Requirements



Large Scale Benchmarking

- Needs many diverse benchmarks
 - Have to avoid tuning to specific benchmarks
- Requires a lot of CPU resources
 - We use over 100 Pentium workstations
 - With infrastructure to control them (PBS)
- Need to test many configurations
 - Several runs for each valid parameter
 - Each parameter may have multiple values

Randomized (vs. Deterministic) Algorithms

- Pro: More effectively search solution space
 - Find better best-case solutions
- Pro: Independent runs stress separate areas of code
- Con: Testing is more difficult
 - Need many runs to determine average solution quality and locate errors
 - Must be able to reproduce failures
 - Determinism on demand

Modular Tool Infrastructure

- In EDA, change is inevitable
 - Need to minimize cost of change
 - Good code infrastructure requires effort to build
 - e.g. avoiding code duplication requires abstractions
- Allows easy import and reuse of existing code
- Essential for testing and debugging
 - Modules can independently sanity check inputs and outputs

Tool Diagnostics

- Runtime breakdown
 - Quickly identify candidates for speedup
 - Potentially identifies mistakes
 - Modules that take much longer than expected
 - Measure scalability of individual modules
- (Peak) Memory consumption
 - Extrapolate usage for larger inputs
 - When does it make sense to use 64-bits?
 - Identify candidates for space conservation

Sample Self-Profiling

```
CapoPlacer took:                144.506sec
Breakdown by component -
Fidducia-Matheyses Partitioner:   8.99sec   (6.22%)
Multi-Level FMPartitioner:        64.34sec  (44.52%)
Optimal End Case Partitioner:     4.15sec   (2.87%)
Partitioning Problem Setup:       4.78sec   (3.31%)
Optimal End Case Placer:          10.47sec  (7.24%)
End Case Placement Problem Setup:  0.17sec   (0.12%)
Level Statistics:                 0.43sec   (0.30%)
Feedback Processing:              0.49sec   (0.34%)
Block Packing:                    49.28sec  (34.10%)
  Block Packing Clustering:        0.07sec   (0.05%)
  Block Packing Annealing:         44.34sec  (30.68%)
  Number of Block Packing Instances: 97
    (successful: 84, failed: 13)
  The largest Block Packing instance had 9 macros.
  The largest failed Block Packing
    instance had 3 macros.
Total measured runtime: 143.09sec (99.02%)

Minor page faults: 164649 Major page faults: 0
Current resident memory: 19.5078MB
Peak process memory: 29.168MB
Peak process memory observed in
"Multi-Level FMPartitioner after clustering"
```

Visualization

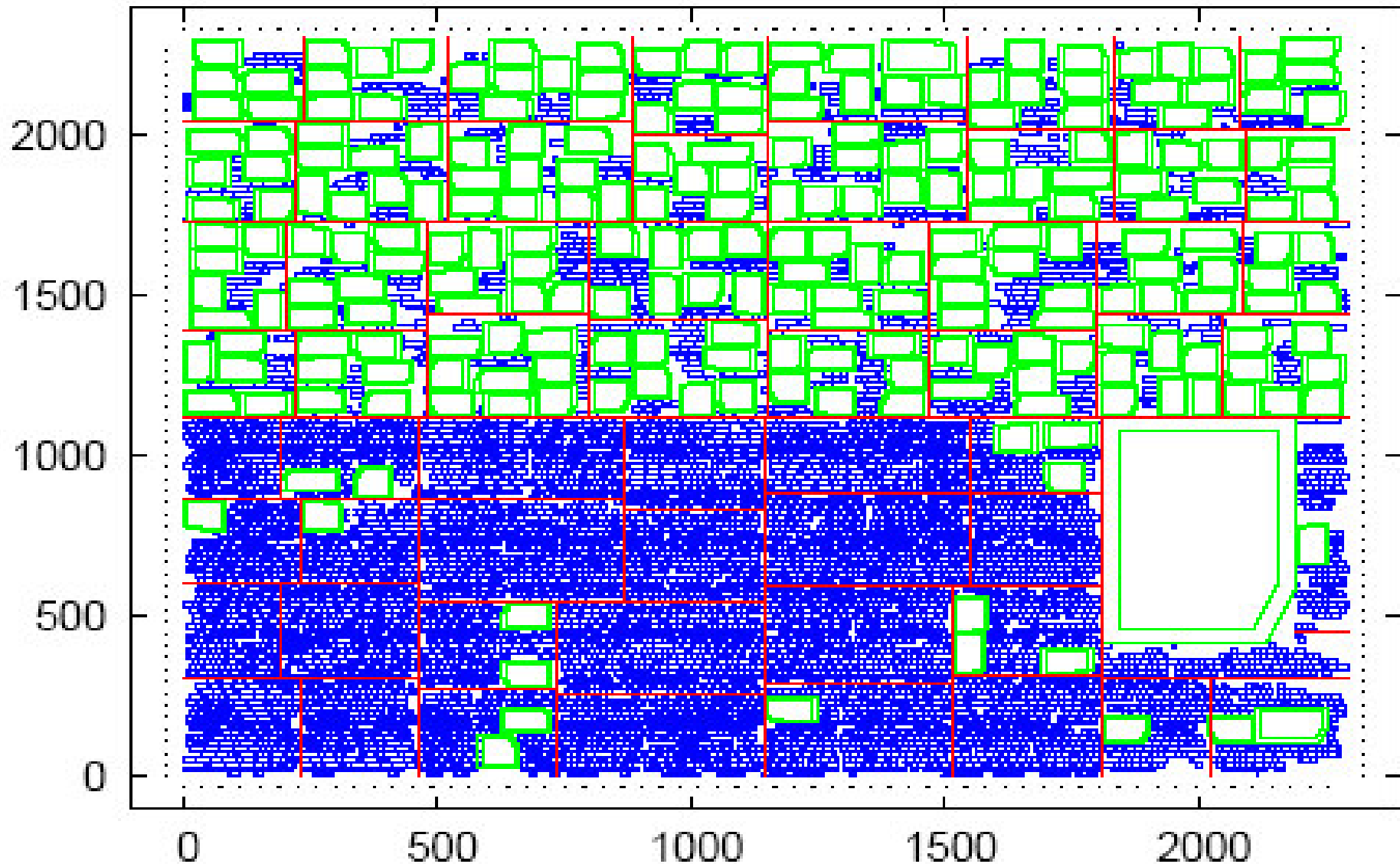
- A picture is worth more than a thousand words
- Captures intermediate steps of tool
- Reading logs is time consuming and potentially ineffective
- People's brains work well visually
- Many problems are easily identified visually

Choice of Visual Representation

- Should be easy to use and have appropriate level of abstraction
- We use GNUplot (<http://www.gnuplot.info>)
 - High level input format simple and text based
 - Allows for easy modification by hand and script
 - Easily overlay multiple data sets on same plot
 - Highly portable
 - Free for all purposes

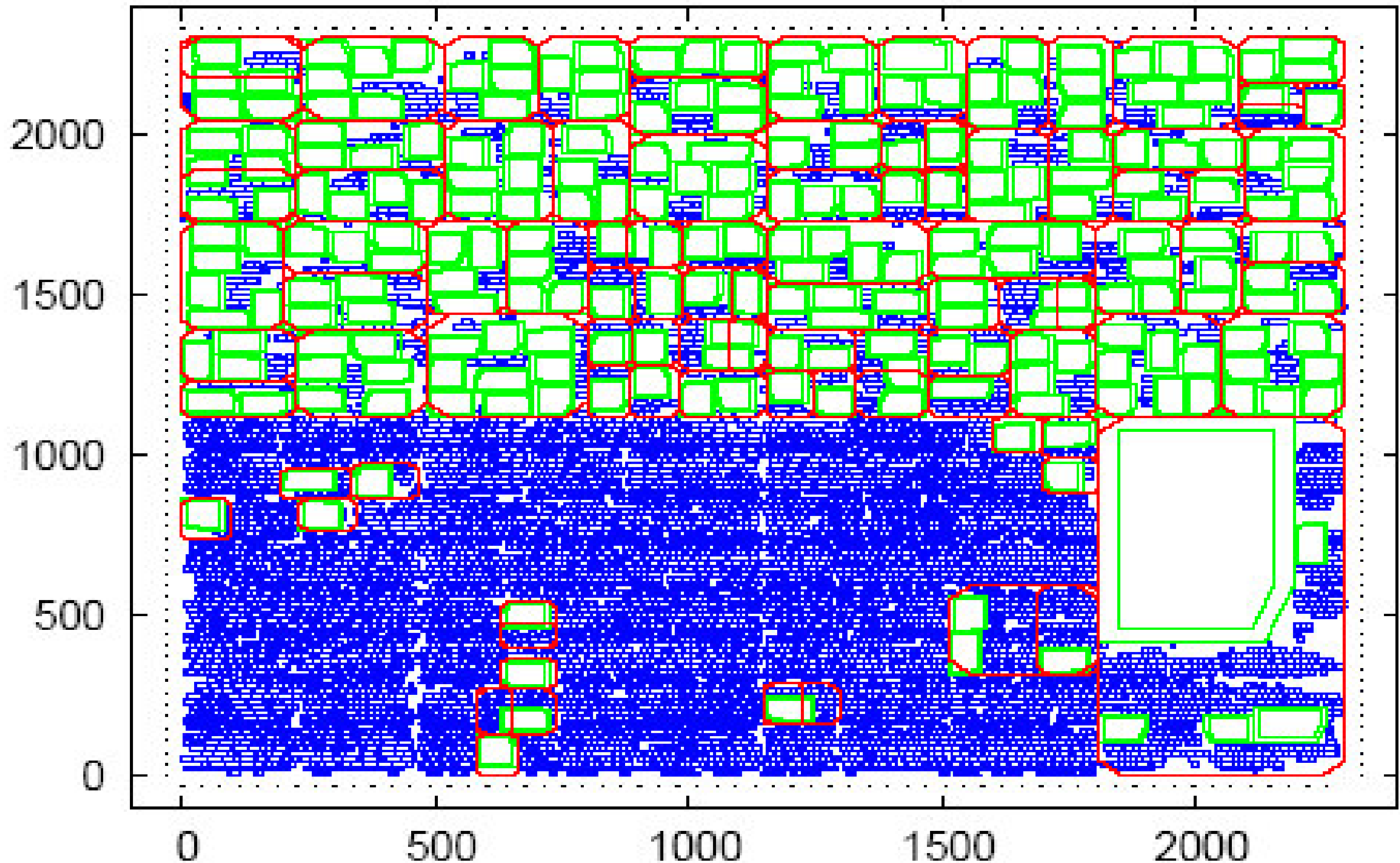
Placement Visual: Cutlines

IBM01 HPWL= 2.491e+06, #Cells= 12752, #Nets= 14111



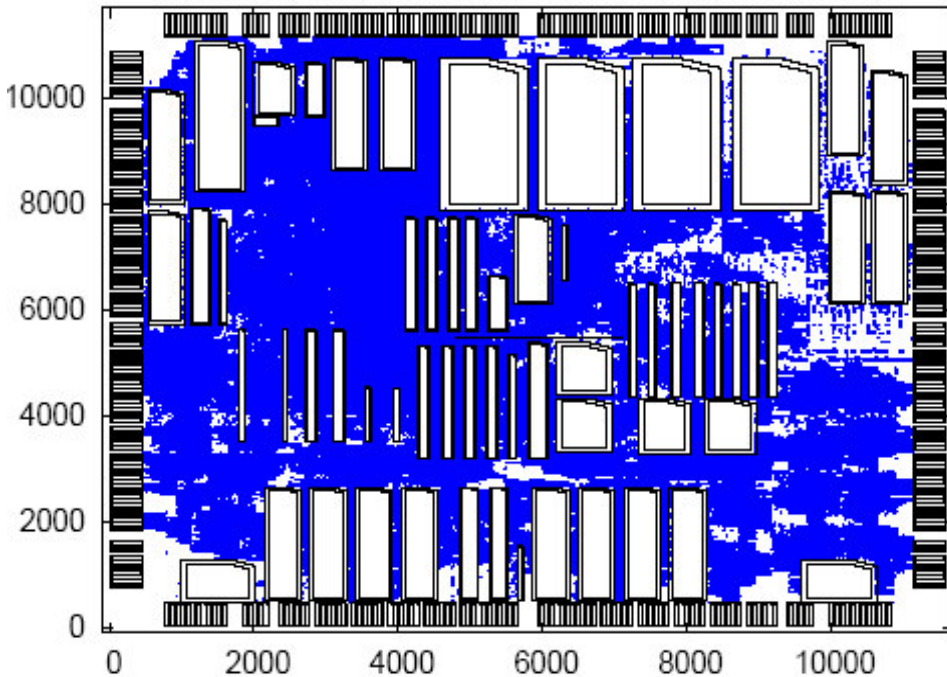
Placement Visual: Floorplanning

IBM01 HPWL= 2.491e+06, #Cells= 12752, #Nets= 14111

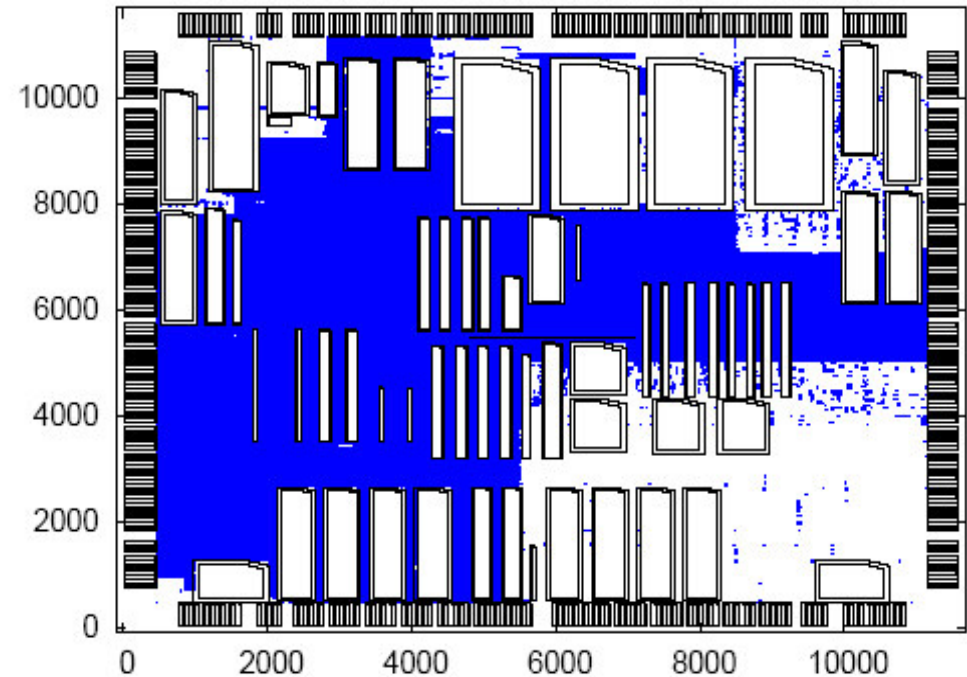


Large-scale Visualization

adaptecl HPWL=9.091e+07, #Cells=211447, #Nets=219794



adaptecl HPWL= 8.666e+07, #Cells= 211447, #Nets= 219794



- Comparison of different whitespace allocation techniques
 - Not visible in run logs
- Plots of large designs warrant compression
 - Less than 1 bit per cell

Conclusions

- Necessary attributes of tools that can effectively handle multi-million gate designs
 - Modular components
 - Large-scale testing
 - 64-bit compatibility
 - Visualization techniques for large inputs
 - Automated diagnostics
 - Memory and runtime growth nearly linear