Unification of Partitioning, Placement and Floorplanning

Saurabh N. Adya,
Shubhyant Chaturvedi, Jarrod A. Roy,
David A. Papa, and Igor L. Markov

MichiganEngineering

Outline

- Introduction
- Comparisons of classical techniques
  - Partitioning, floorplanning, and placement
- Unification
  - Application to large-scale floorplanning
  - Application to mixed-size placement
  - Application to free shape floorplanning
- Our implementation
- Summary

“Hard macros will revolutionize SoC design”

Enno Wein & Jacques Benkoski, *EEDesign*, Aug 20, 2004

- Hundreds of predesigned macros
  - Embedded memories, analog circuitry, IP blocks
  - Existing layout tools are having problems
- **Macro placement is usually separate from standard cell placement** (done once & never repeated)
- Lower utilization, larger dies, lower yield, higher cost

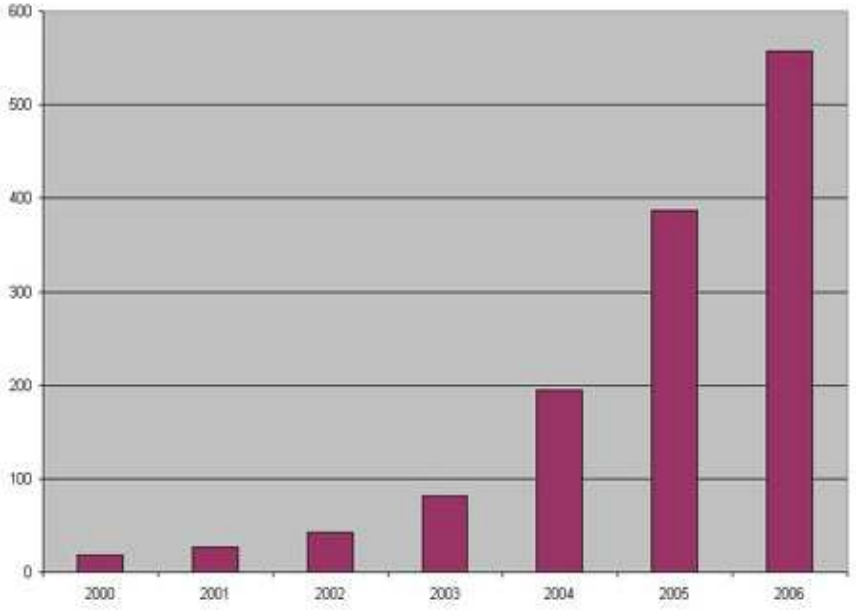


Figure 1 — Growth in the number of hard macros in SoC designs

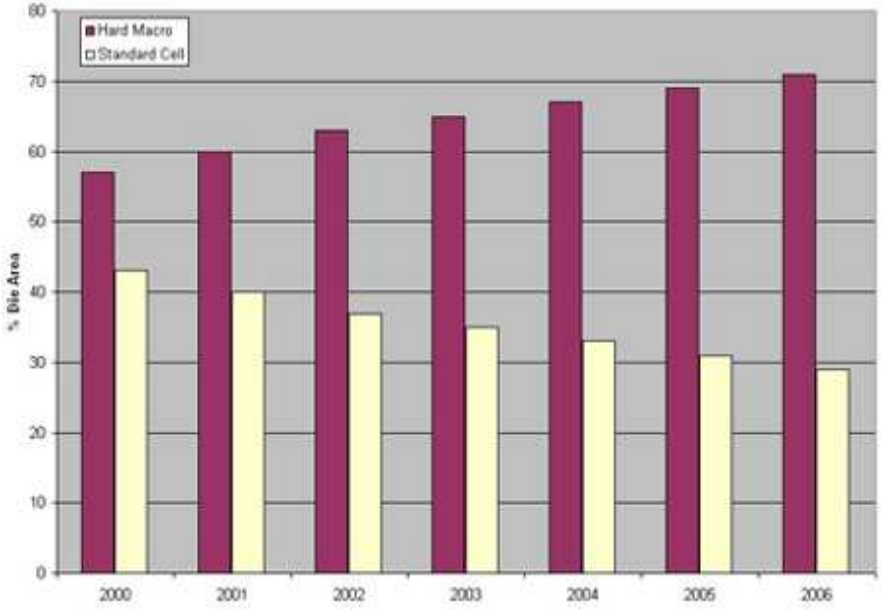
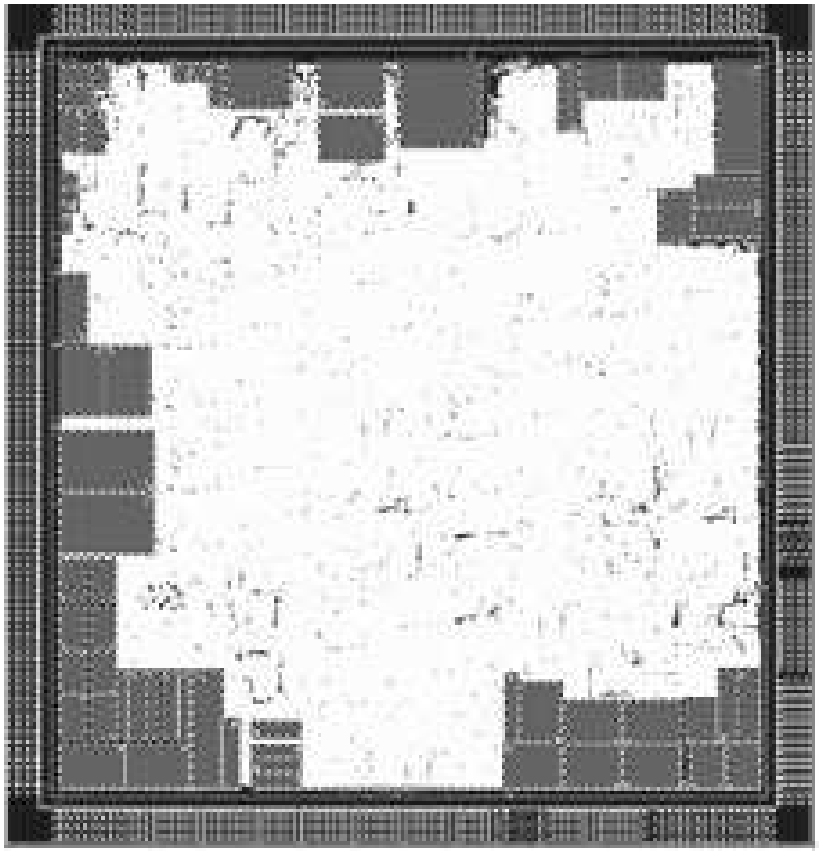
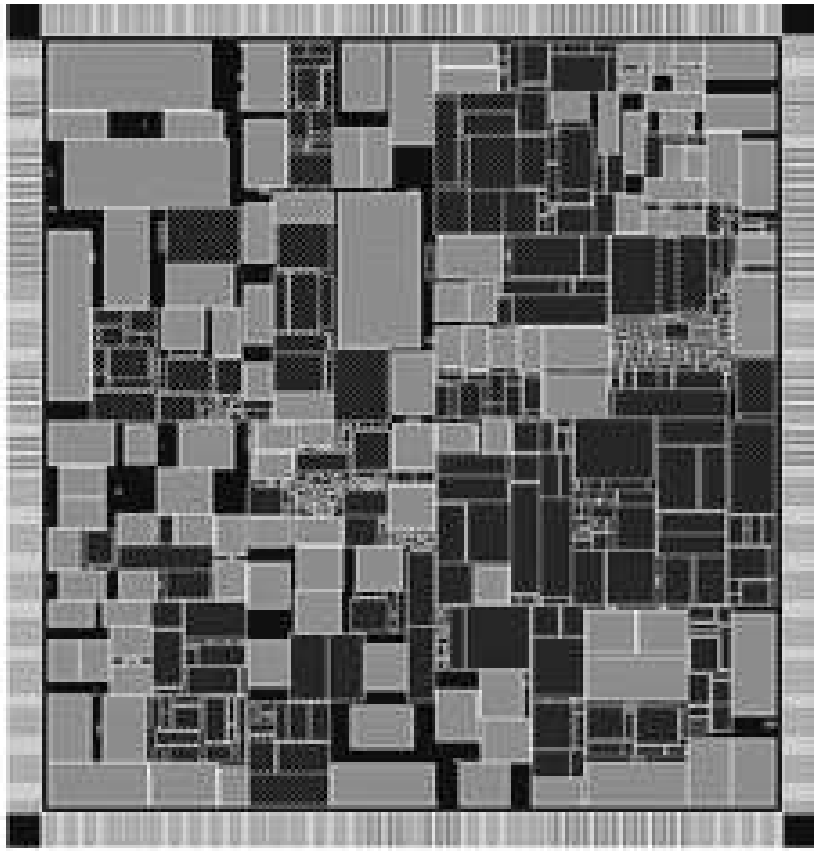


Figure 2 — Hard macros vs. standard cell area

From a “Sea of Cells” to a “Sea of Hard Macros”

Images from *EEDesign*, August 20, 2004





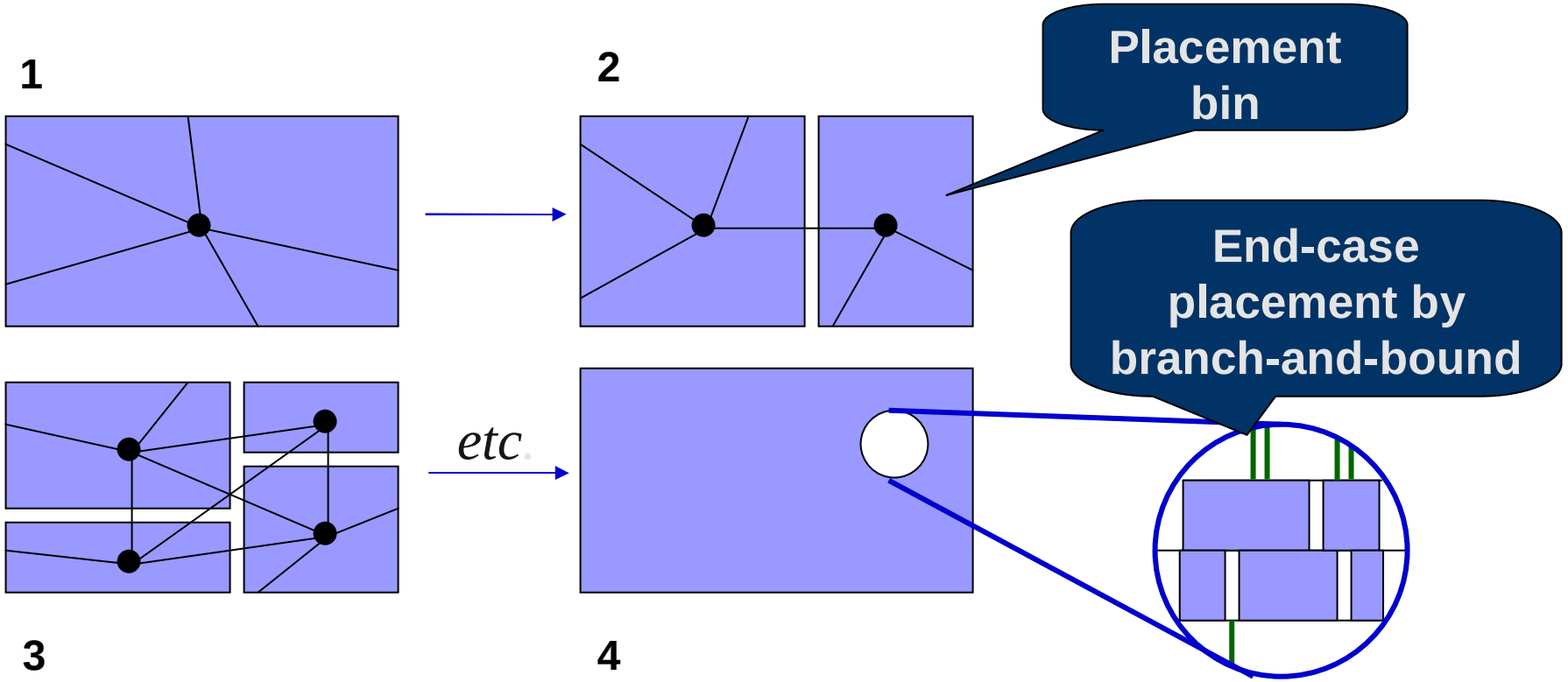
Review: Partitioning & Floorplanning

- **Partitioning**: breaks up the netlist into modules
  - Facilitates a hierarchical design methodology (e.g., for placers that do not scale well)
- **Floorplanning**: seeks non-overlapping locations for modules with fixed & flexible dims (*hard & soft*)
  - Objectives: minimize area and interconnect
  - "Variable-die" or "fixed-die" (full chip or a partition)
- **Partitioning & floorplanning** together facilitate early estimation of interconnect
  - Estimates useful in logic synthesis

Review: Placement vs Floorplanning

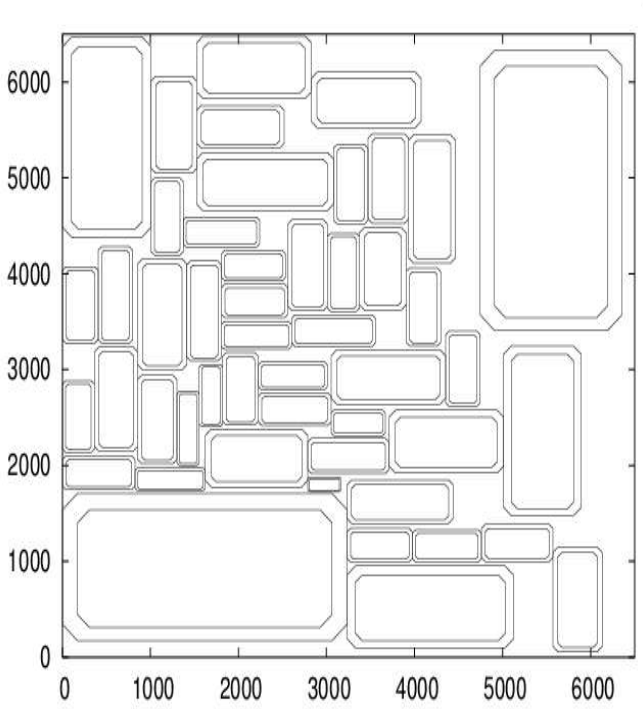
- Mathematically, placement and floorplanning (FP) are the same problem
  - Seek module locations
  - Must avoid overlaps between modules
  - Must observe region constraints
  - Seek to minimize interconnect (power)
  - Seek to satisfy delay constraints
- Main differences
  - Scale (number of objects) and algorithms
- This work: a unified tool (floorplacer) can dynamically invoke FP or partitioning

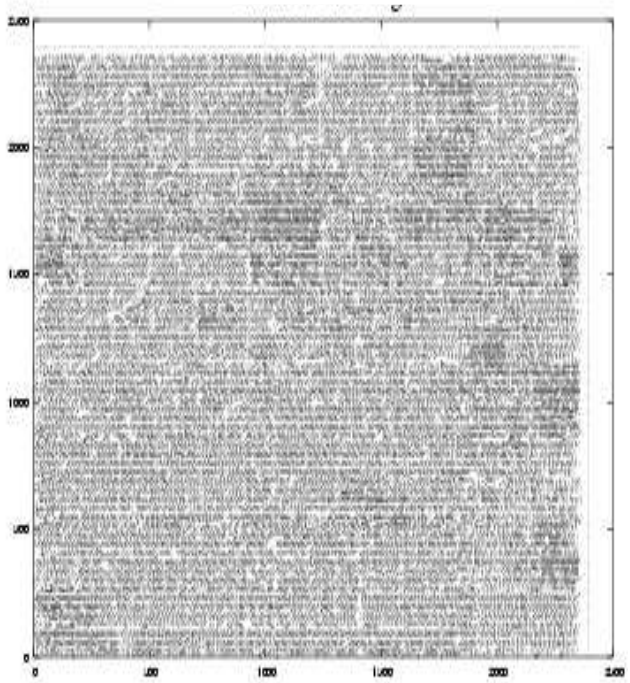
Global Placement by Recursive Min-cut Partitioning

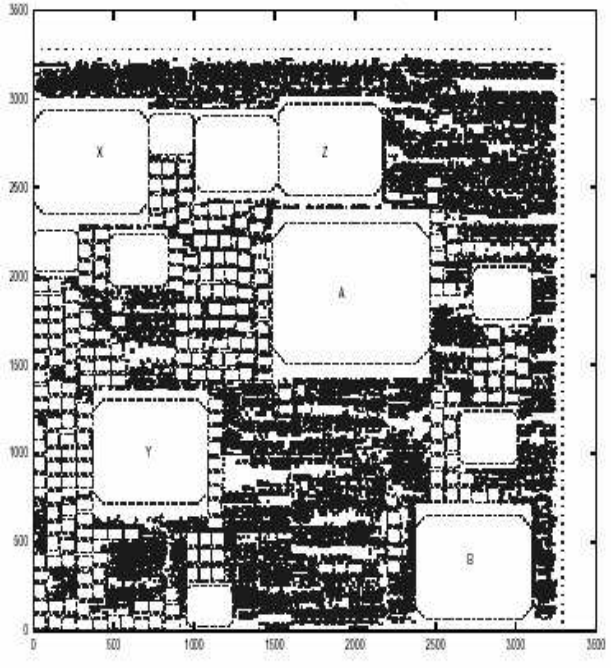


- Placers using min-cut bisection: Capo, FengShui, IBM CPlace, Cadence QPlace

Block-based Design Std-cell Design Mixed-size Design



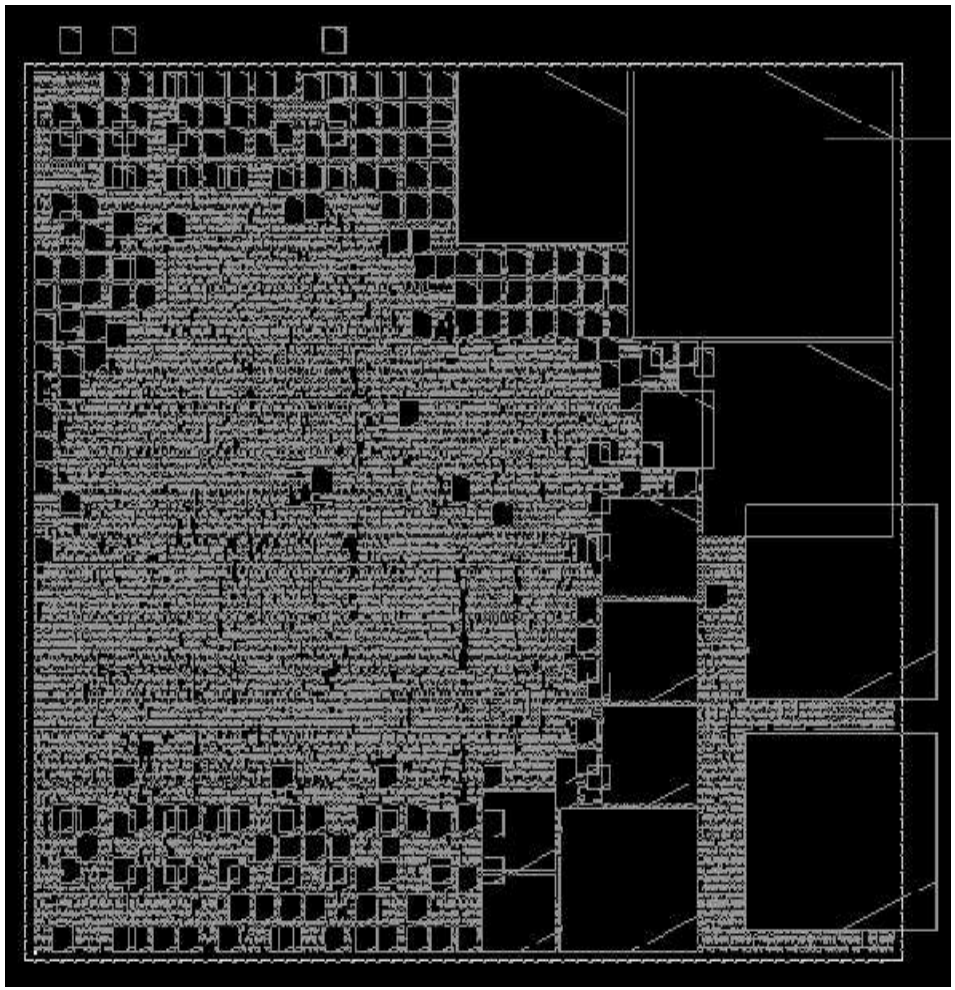


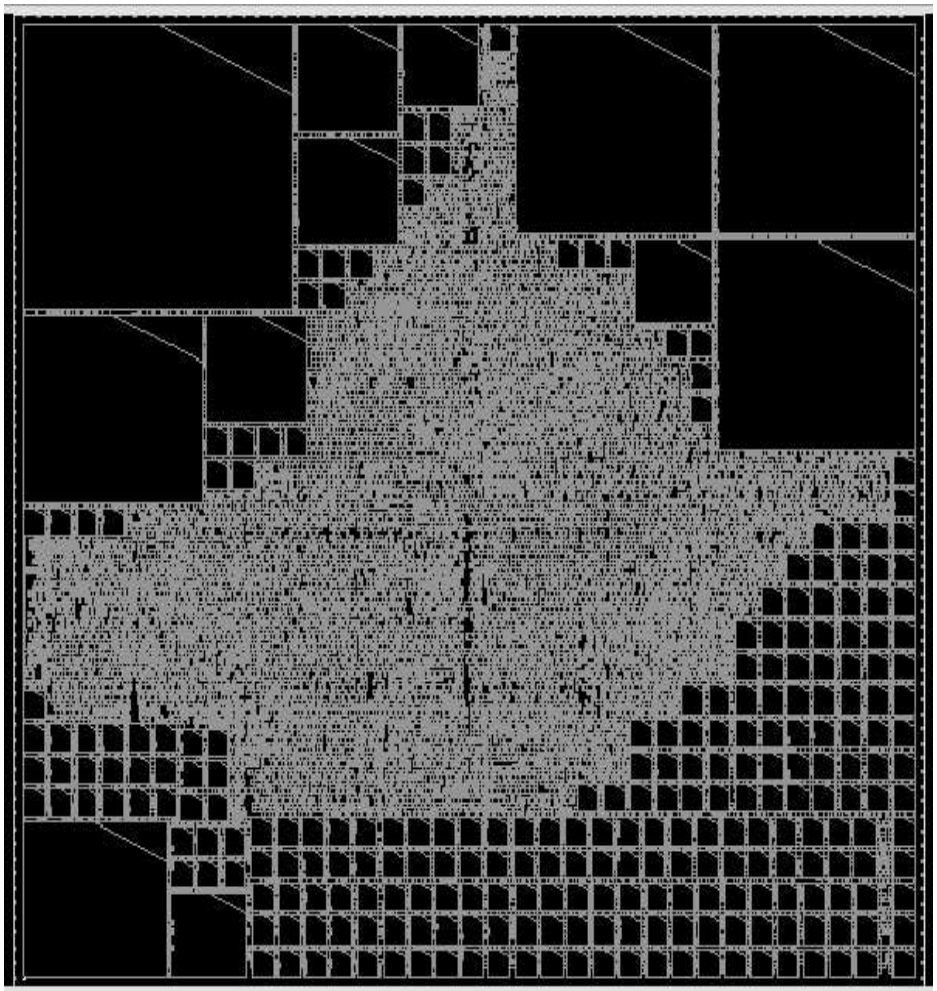


- Large rectangles can represent
  - Intellectual Property (IP): hard or soft
  - Macros, memories, data-paths, analog modules
  - Modules of unsynthesized logic

Cadence SEDSM/QPlace on IBM02

↓ v. 5.1.67 (2002) versus ↓ v. 5.4.126 (2004)

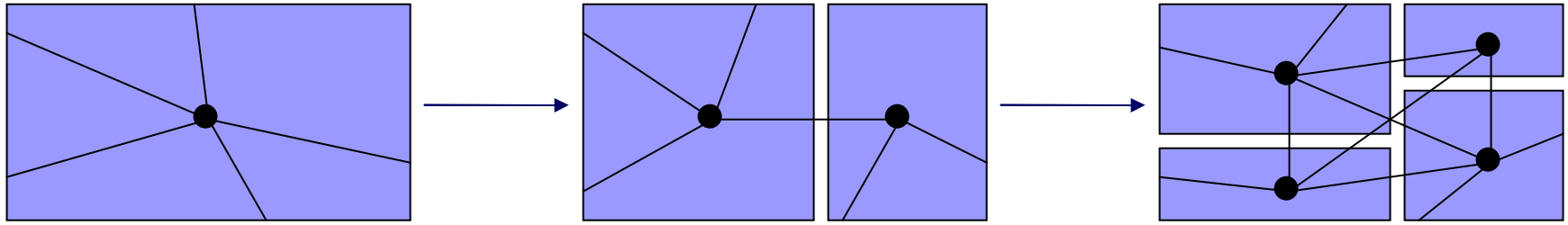




Floorplacement

↓Characteristics	Min-cut Partitioners	Floor-planners	Placers	Floorplacers
Scalable Runtime	Yes	No	Yes	Yes
Scalable Wirelength	N/A	No	Yes	Yes
Explicit non-overlapping constraints	No	Yes	No	Yes
Can handle large modules	Yes	Yes	No	Yes
Routability optimization	No	N/A	Yes	Yes
Can optimize orientation of modules	No	Yes	No	Yes
Support for non-rectangular blocks	Yes	Limited	No	Yes
Support for soft rectangular blocks	Yes	Yes	No	Yes
Handling net weights	Yes	Yes	Yes	Yes
Handling length bounds	No	Yes	Yes	Yes

Observe: Min-cut Placement Produces Slicing Floorplans



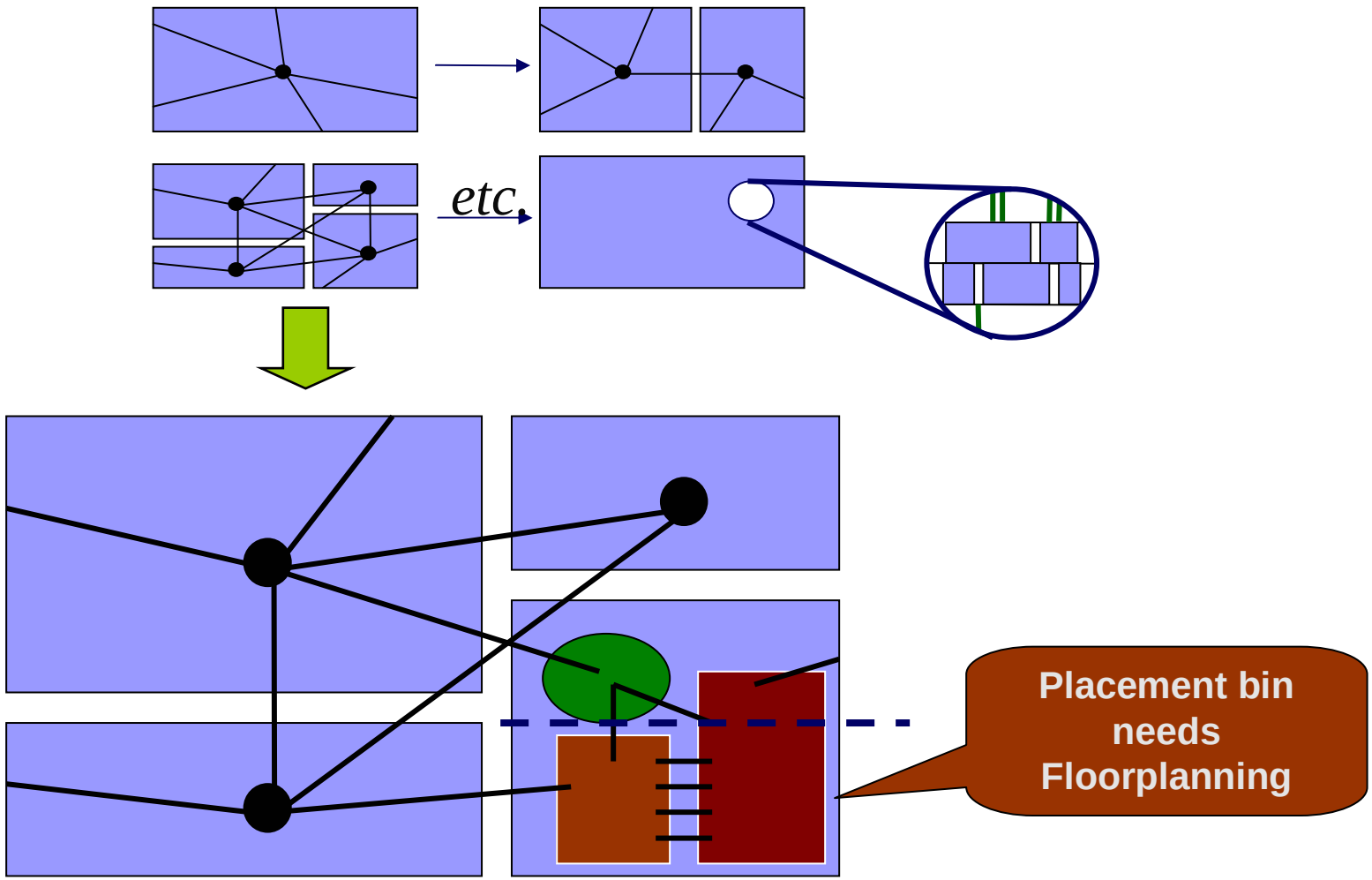
Slicing Floorplan!

- Using this effect in floorplanning may reduce run-time & wirelength by combining partitioning & FPing
  - Recall: traditional floorplanners use Simulated Annealing
- We are not giving up non-slicing FPs either!

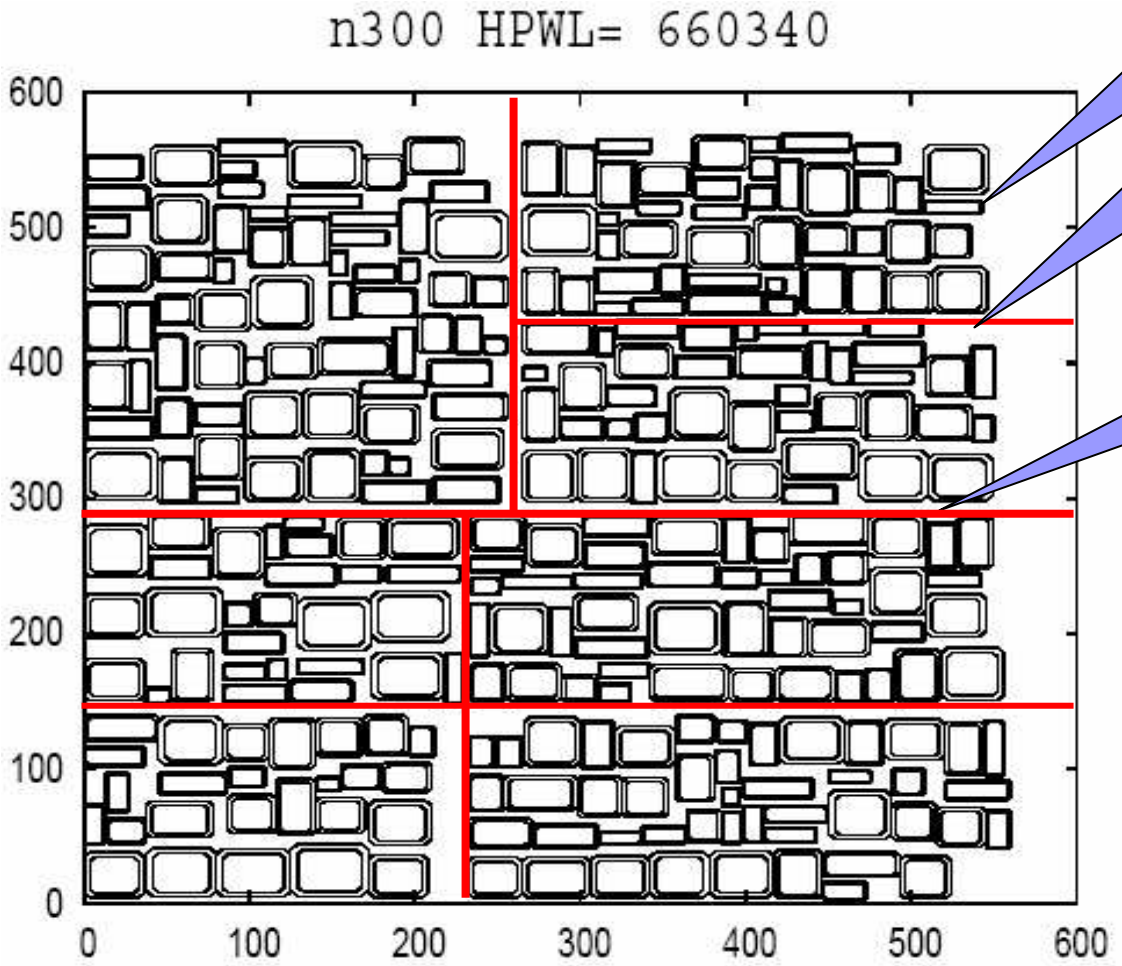
Our Approach: Direct Integration of Placement & Floorplanning

- Perform top-down min-cut placement
- Fall back on floorplanning when necessary
  - ⇒ many "local" calls to a floorplanner
  - In rare cases, packing may be infeasible
  - What can/should be done then?
- Example: to solve mixed-size placement, can start with several slicing cuts
  - Eventually may need to pack blocks (when exactly?)
  - Call fixed-outline floorplanning

Placement by Recursive Bisection + Fixed-outline floorplanning



Example (Min-cut Floorplacement)



Placement bin needs

Merge w Sibling & Re-floorplan

Cut-line (min-cut)

Our Floorplacement Algorithm

```
 Variables: Queue of placement bins
 Initialize queue with top-level placement bin
1  While (queue not empty)
2    Dequeue a bin
3    If bin has large/many macros or is marked as merged
4      Cluster std-cells into soft macros
5      Use fixed-outline floorplanner to pack all macros (soft+hard)
6      If fixed-outline floorplanning succeeds
7        Fix macros and remove sites underneath the macros
8      Else
9        Undo one partition decision and merge bin with sibling
10       Mark new bin as merged and enqueue
11   Else If bin small enough
12     Process end case
13   Else
14     Bi-partition the bin into smaller bins
15     Enqueue each child bin
```

Condition empirically determined

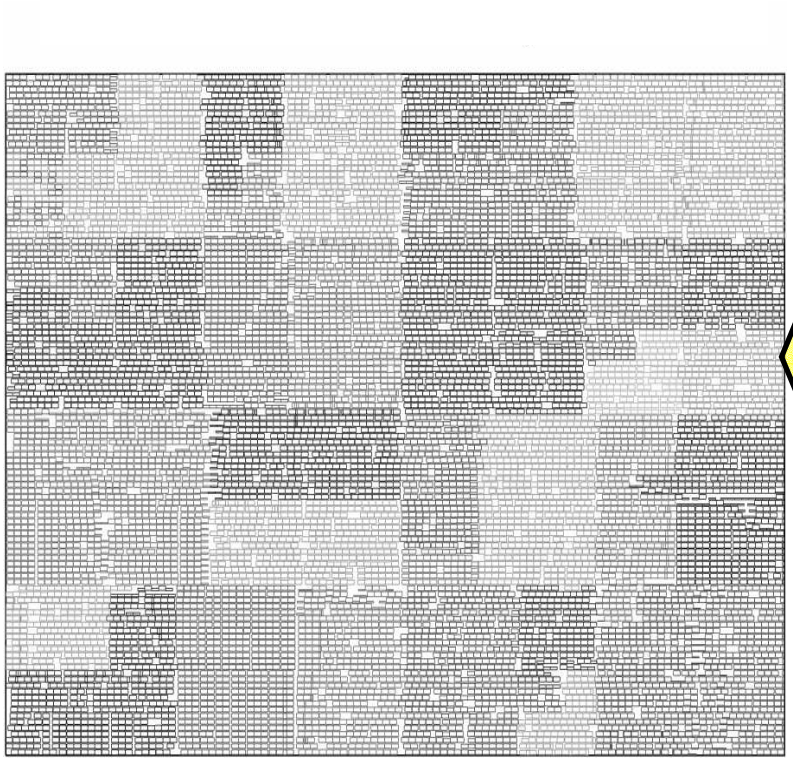
Lines 3-10 are different from traditional min-cut placement

Early Criteria for Block Packing

- Large-macro tests (used to improve runtime)
  - At least 1 macro does not fit in child bins
  - <30 macros total, with total area > 80% of bin area
- What if fixed-outline floorplanning fails ?
  - Return to previous level of placement hierarchy
    - Merge two child bins to form a parent bin
  - Try area-only floorplanning
  - Else final placement has overlaps
    (can try legalizing it at the end!)
- Above conditions detect block-based designs, std-cell and mixed-size designs

Free-Shape Floorplanning

(see details in the paper)

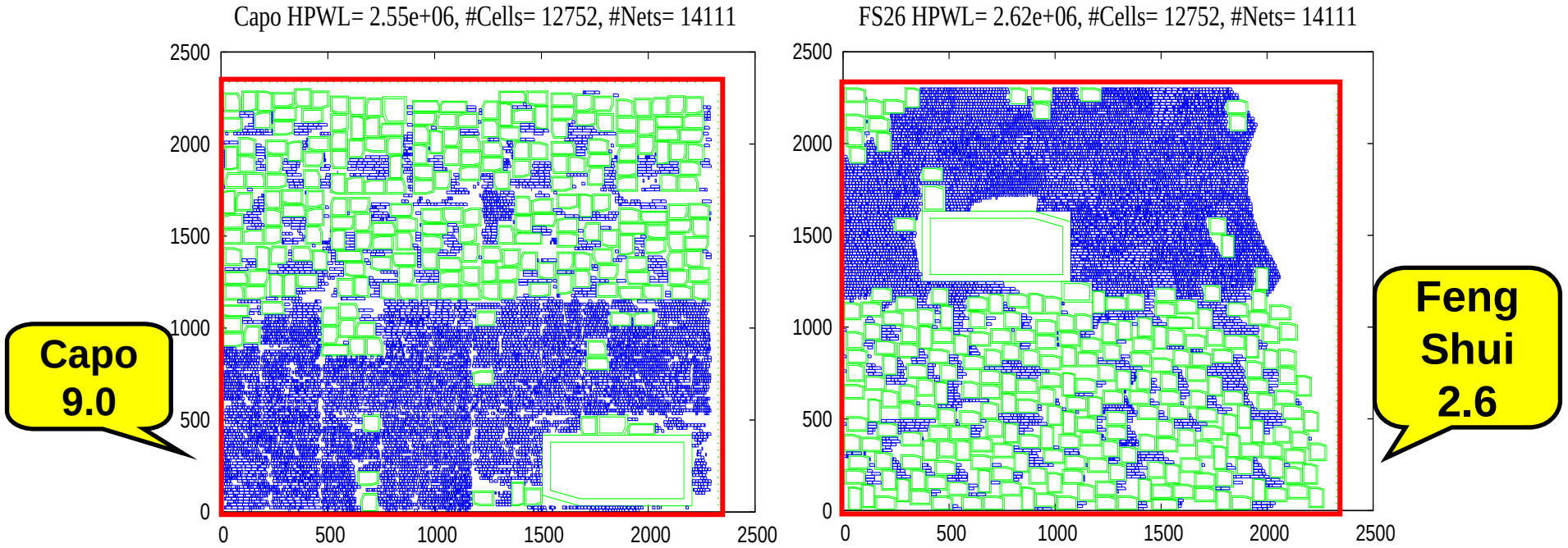


Circuit	Parquet2.0 (rectangles) HPWL	Capo 9.0 (free-shape) HPWL	Improvement in HPWL
ami33	76987	46072	40.1%
ami49	895560	469476	47.5%
n50	202240	87957	56.5%
n100	350593	157548	55.0%

- Shorter interconnect can improve timing & power

New Benchmarks: IBM Mixed-Size wPins

- IBM-MixedSize 2002 (IBM-MS) suite
  - **All large modules are square**
  - **All pins for modules are in the center**
- The new suite (IBM-MS w Pins)
  - **Non-square blocks (aspect ratios ∈ [0.5,2.0])**
  - **Pins uniformly distributed around cell periphery**
- URL: http://vlsicad.eecs.umich.edu/BK/ICCAD04bench/

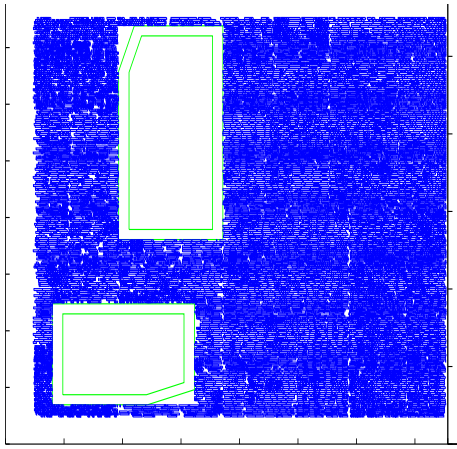


New M-S Benchmarks With Routing Info

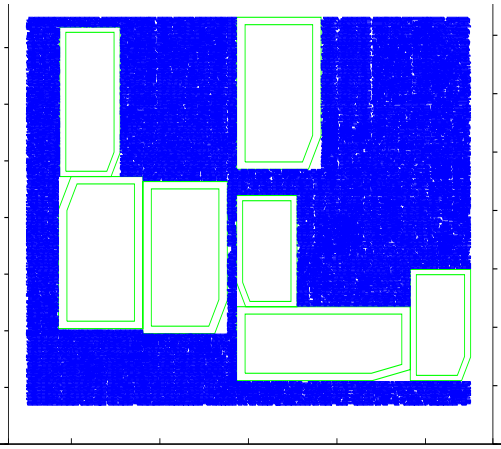
- Derived from circuits posted online by Faraday Corp.
- **Routing information provided to run Cadence WarpRoute**
- http://vlsicad.eecs.umich.edu/BK/ICCAD04bench/

↓Circuit	#Nodes	#Nets	#IO's	Utilization	#Macros	%Area in macros
DMA	11734	13256	948	95.43	0	0
DSP1	26299	28447	844	90.66	2	21.98
RISC1	32615	34034	627	93.94	7	41.99
DSP2	26279	28431	844	90.05	2	6.96
RISC2	32615	34034	627	94.09	7	37.31

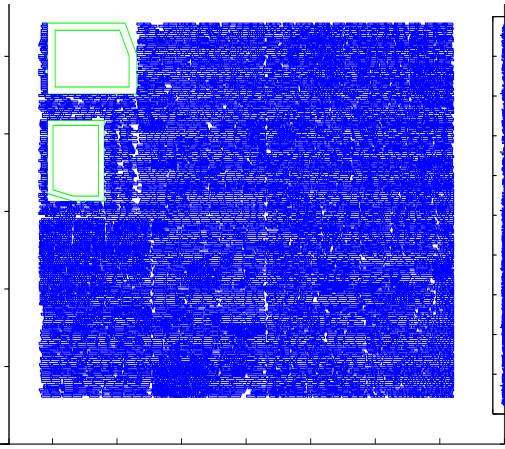
DSP1



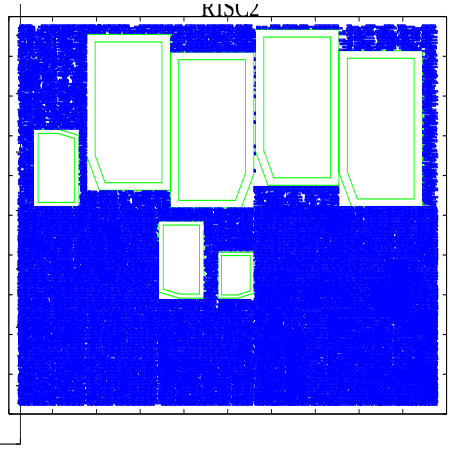
RISC1



DSP2



RISC2



Capo 9.0

- http://vlsicad.eecs.umich.edu/BK/PDtools/
- Source code available for free, for all uses
  - Linux (32/64 bit), Solaris (32/64), Windows (32)
- Reads/writes LEF/DEF
- Bridge to OpenAccess 2.2
- Placements typically routable (e.g., IBMv2 BMs)
- Optimization of soft macros
- Obstacles are supported (see DAC ’00 paper)
  - Blockages are converted to obstacles
  - **Placing macros around fixed obstacles is non-trivial** (sometimes causes overlap, but we are working on this)

Results for Block-Based Designs

Circuit (GSRC)	#Blocks	Parquet 2.1		Capo 9.0		
		HPWL	Time (sec)	HPWL	Time (sec)	# Min-Cut Levels
n10	10	5.58	0.27	5.57	0.37	0
n30	30	17.38	2.35	16.93	1.89	1
n50	50	20.77	8.16	20.34	5.30	1
n100	100	34.53	50.12	32.39	**10.50**	2
n200	200	**62.28**	**240.61**	**56.82**	**27.42**	3
n300	300	**75.69**	**433.92**	**63.62**	**25.21**	3

Up to 16% less interconnect, 20x faster

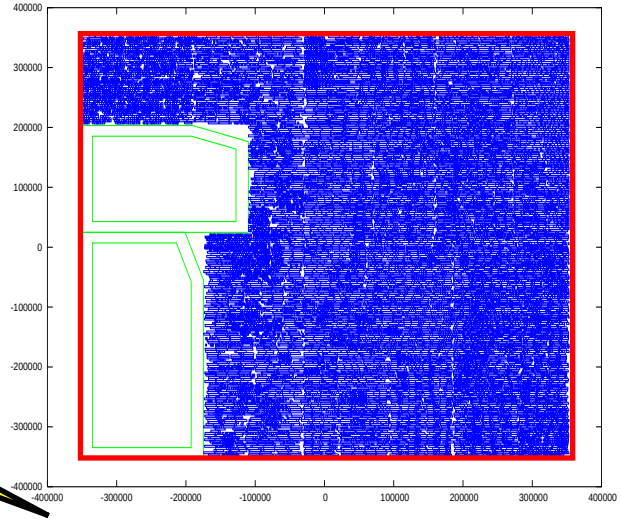
Results: Mixed-size P&R

↓Ckt	SEUltra - Qplace(v5.4.126)					Capo 9.0 -feedback					FengShui 2.6 6/17/04				
	Place		Route			Place		Route			Place		Route		
	WL (e8)	Time (min)	WL (e8)	Time (min)	Viol	WL (e8)	Time (min)	WL (e8)	Time (min)	Viol	WL (e8)	Time (min)	WL (e8)	Time (min)	Viol
DMA	4.7	1	6.3	3	0	**4.4**	2	**5.7**	3	0	4.6	6	6.3	3	0
DSP1	10.5	5	12.7	5	0	**9.8**	24	**11.7**	5	1	10.7	14	14.1	8	0
RISC1	16.7	7	21.6	11	3	**15.7**	21	**21.5**	16	0	19.9	30	**OC**	**OC**	**OC**
DSP2	9.9	4	12.0	6	0	**9.2**	9	**11.1**	5	0	9.2	10	11.6	6	0
RISC2	**15.6**	8	**20.7**	30	333	16.3	19	21.3	11	5	209	25	**OC**	**OC**	**OC**

- Capo placements routable, have the best wirelength in all but one benchmark

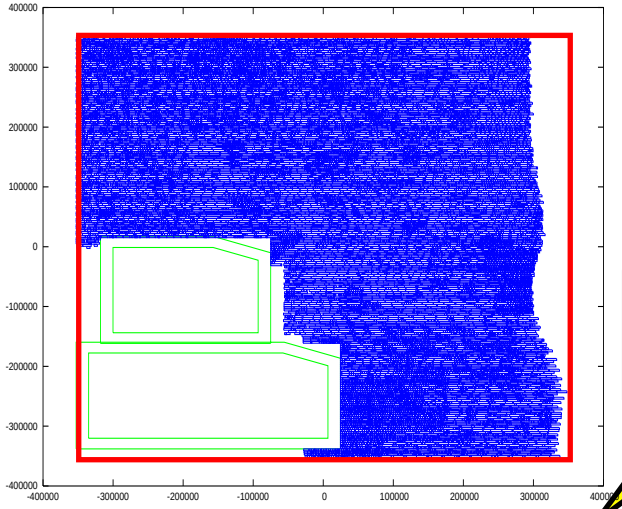
Sample Placements

DSP1 HPWL= 9.84e+08



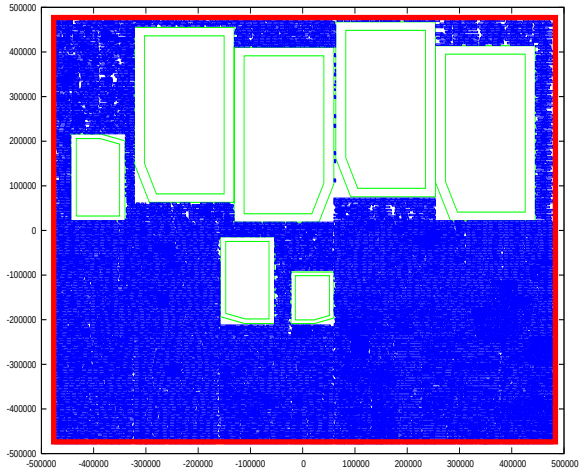
Capo 9.0

DSP1 HPWL= 10.98e+08

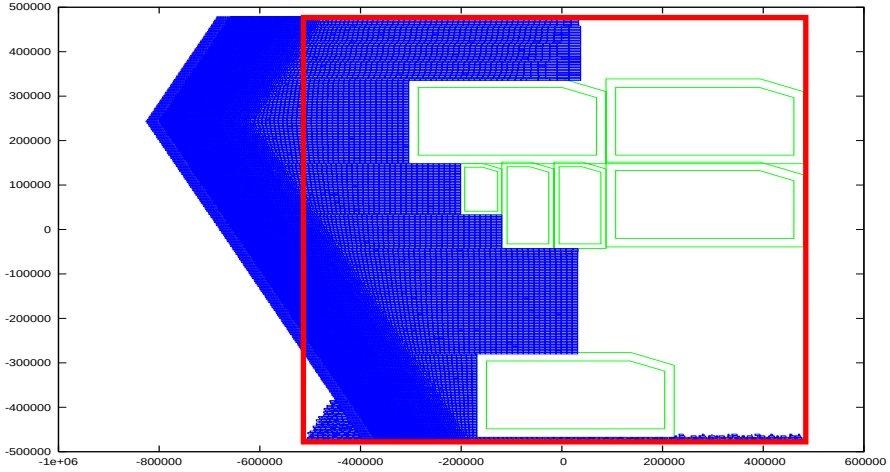


Feng Shui 2.6

RISC2



RISC2_FS HPWL=209.88e+08



Results on Popular IBM-MS BM's

Benchmark Suite	SEUltra v5.1.67 (2002)	SEUltra v5.4.126 (2004)	Capo+ Parquet+ Capo	Capo+ Kraftwerk ECO	mPG	Feng-Shui v2.6 6/17/04	Capo v9.0 -feedback	Capo v9.0 -feedback best-of-2
IBM-MS (ISPD 02)	92.8%	12.2%	19.8%	14.7%	14.2%	-7.9%	0%	-2.0%
IBM-MSwPins (new)	--	15.8%	21.7%	19.7%	--	-3.4%	0%	-1.3%

- Percentages represent differences in HPWL with respect to Capo v9.0
  - Positive percentages indicate larger wirelength than Capo
  - Negative percentages mean smaller wirelength == better performance
- Note that FengShui 2.6 placements are packed to an edge of the core
  - In practical applications, may need to be spread out for routing

Capo 9.0 Runtime Breakdown

(IBM01 mixed-size w pins, 2.4GHz Pentium4)

```
MLPart took:                    83.73sec (45.71%)
FMPart took:                    12.66sec (6.91%)
BBPart took:                     2.77sec (1.51%)
SmPlace took:                   12.56sec (6.86%)
ProblemSetup took:               5.06sec (2.76%)
Level Stats took:                0.32sec (0.18%)
Feedback Processing took:        0.79sec (0.43%)
Floorplanning took:              59.5sec (32.48%)
  Floorplanning Clustering took:    0.26sec (0.14%)
  Floorplanning Annealing took:    56.99sec (31.11%)
  No. Floorplanning Instances: 40 (successful: 38, failed: 2)
  The largest floorplanning instance has 23 macros.
  The largest failed instance has 2 macros.
Total runtime of measured components: 177.4sec (96.83%)
```

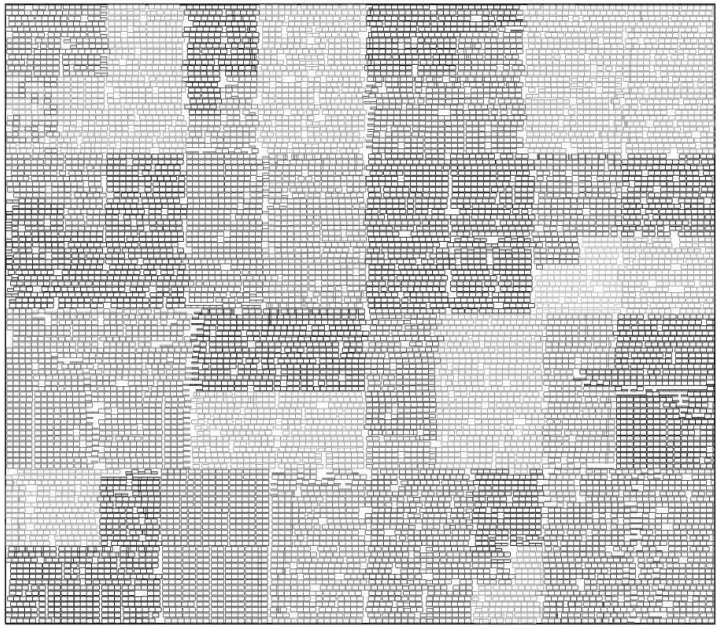
Speed: 4.3K cells & macros / minute (near-linear scaling)

6.3K cells / minute (near-linear scaling)

Summary

- Min-cut floorplacement
  - unifies partitioning, floorplanning & placement
- A working floorplacer is now available (Capo9.0), competitive in these categories
  - Geometric multi-way partitioner
  - Fixed-outline floorplanner with interconnect optimization
  - Large-scale standard-cell and mixed-size placer
  - Free-shape floorplanner (places & shapes modules)
- New benchmarks
  - IBM 01-18 mixed-size with non-zero pin offsets (and non-square blocks)
  - Faraday circuits: complete P&R benchmarks w embed. memories
- Curr. work: adapting floorplacement in design flows

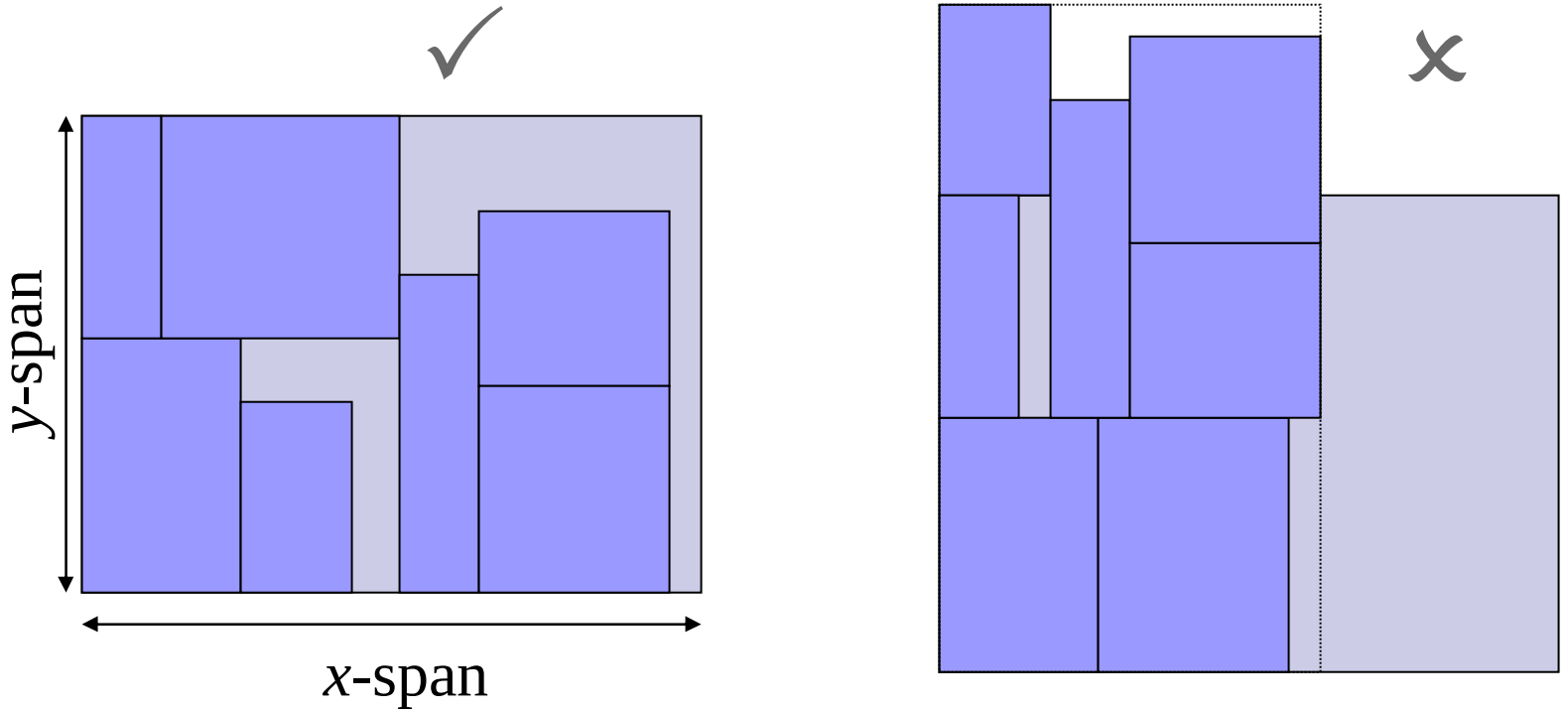
Thank You !



Circuit	Cadence SEUltra Block-Place+QPlace Sun-Blade1000,750MHz I		Capo+Parquet+Capo [2] (Low-Temp. Annealing) Linux/Pentium,2GHz II		Capo+Kraftwerk ECO [2] Linux/Pentium,2GHz III			FengShui v2.6 06/17/04 Linux/Pentium,2.4GHz V		Capo v9.0 -feedback Linux/Pentium,2.4GHz VI	
	HPWL (e6)	Time (min)	HPWL (e6)	Time (min)	HPWL (e6)	Time (min)	% Overlap	HPWL (e6)	Time (min)	HPWL (e6)	Time (min)
ibm01	3.25	12	3.23	18	2.96	5	1.22	2.56	3	2.57	4
ibm02	7.17	31	7.91	12	6.84	13	0.25	6.05	5	5.30	8
ibm03	9.06	28	10.08	57	9.45	13	0.18	8.77	6	8.55	14
ibm04	10.28	31	11.01	12	10.09	15	0.74	8.38	7	9.38	18
ibm05	11.55	24	11.03	5	11.46	5	0	9.94	8	10.78	8
ibm06	8.33	32	8.70	19	9.22	19	0.25	6.99	9	7.12	12
ibm07	13.79	41	14.34	22	14.34	57	0.24	11.37	12	12.67	20
ibm08	17.36	50	17.01	26	17.63	22	1.80	13.51	15	15.63	38
ibm09	16.91	56	19.53	29	21.04	32	0.35	14.12	14	15.55	27
ibm10	43.71	86	53.34	119	49.52	72	4.34	41.96	22	35.09	40
ibm11	24.98	71	25.51	43	25.48	42	0.76	21.19	21	21.71	37
ibm12	46.38	87	54.82	97	61.48	53	0.63	40.84	22	41.81	69
ibm13	33.06	91	34.30	54	32.37	73	0.12	25.45	25	28.00	61
ibm14	45.74	148	48.66	145	47.63	117	0.07	39.93	52	40.87	70
ibm15	68.63	206	70.68	208	62.63	124	0.09	51.96	67	55.09	99
ibm16	75.94	248	75.27	154	78.47	166	2.03	62.77	70	65.89	112
ibm17	92.41	288	87.81	204	85.40	132	0.13	69.38	79	77.99	98
ibm18	57.04	190	54.66	115	57.47	162	0.02	45.59	87	49.22	86
Avg	15.83%		21.71%		19.79%			-3.47%		0%	

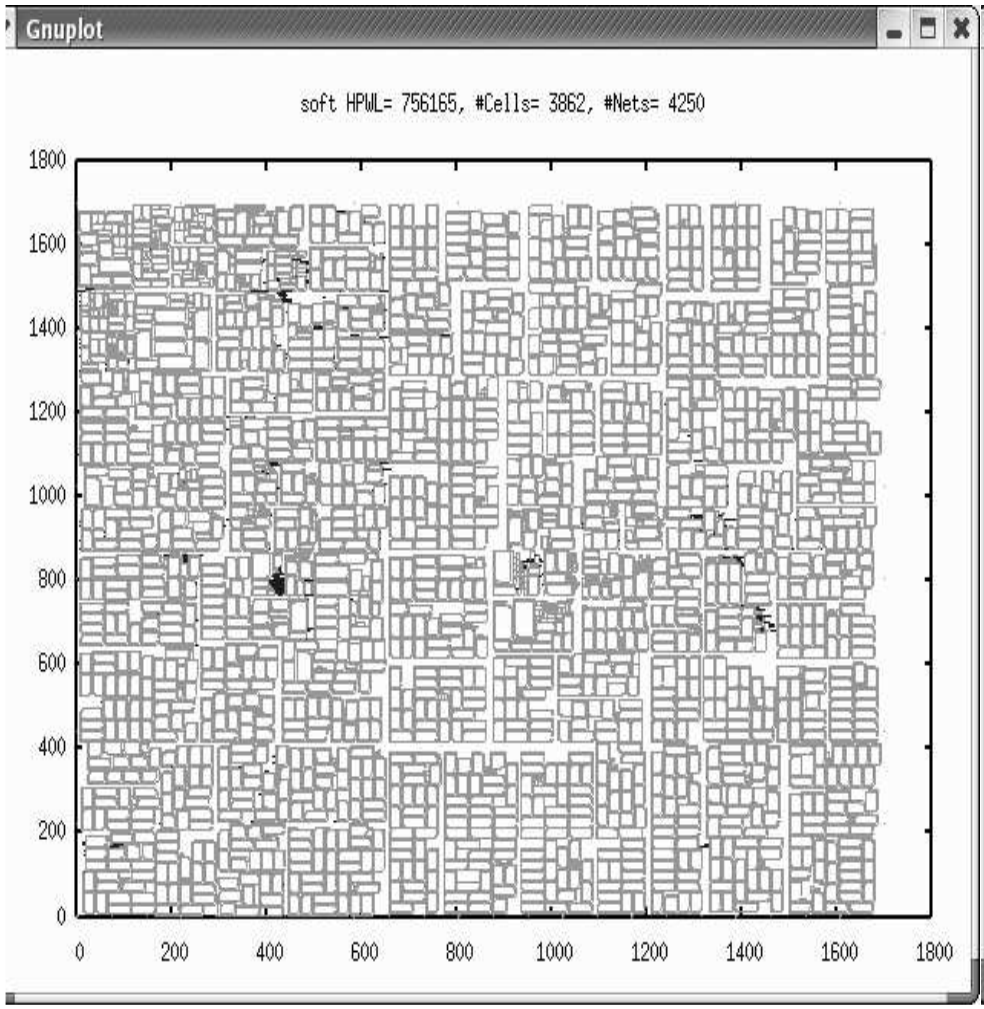
Fixed Outline Floorplanning

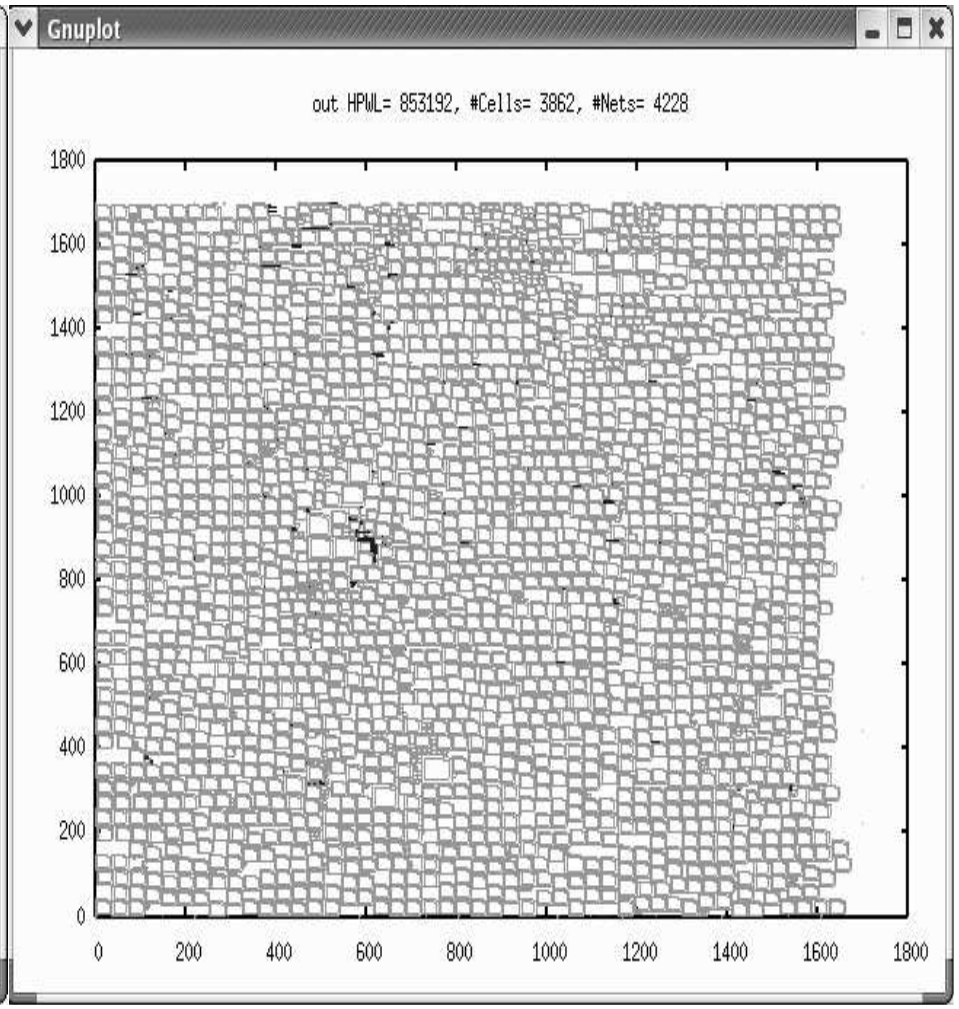
- Not an area minimization problem
  - Rather a constraint satisfaction problem
  - “Classical Floorplanning Considered Harmful” [Kahng, ISPD `00]
- Sample tool: *Parquet* [ICCD`01, TVLSI`03]



Capo 9.0

Fengshui2.6





Why Mixed-size Placement is Difficult

- Mixed-size placement is at least as hard as
  - Standard cell placement (many small movable modules)
  - Floorplanning (large, bulky modules are difficult to pack, especially on a fixed die!)
- Typical optimization heuristics are move-based
  - Each move is “local”, i.e., affects few other objects
  - However, large modules affect many other modules
  - Some moves have ripple-effect on small cells
- Removing overlaps after global placement is not easy, invalidates top-down estimation
  - Need correct-by-construction methodologies

Integrated Partitioning, Floorplanning and Placement

- Traditional design flows apply separate optimizations
  - Mostly a scalability concern for old algorithms
- New generation of fast min-cut placers enable an integrated approach
  - A min-cut partitioner is part of the placer
  - Shifting cut-lines perform floorplanning
  - End result: locations of modules (a placement)

Classical Floorplanning

- Seeks non-overlapping locations of hard and soft blocks
- Objectives: minimize area and/or wirelength
- Core area not pre-defined (variable-die layout)
- Floorplan representations:
  - Location-based versus topological
  - O-Tree, B*-Tree, Sequence Pair, TCG, CBL etc
  - We use SP, but our methods are generally applicable
- Simulated Annealing (SA) used for optimization