*Supported by Cadence*

# Can Recursive Bisection Alone Produce Routable Placements?

Andrew E. Caldwell

Andrew B. Kahng

**Igor L. Markov**

**http://vlsicad.cs.ucla.edu**

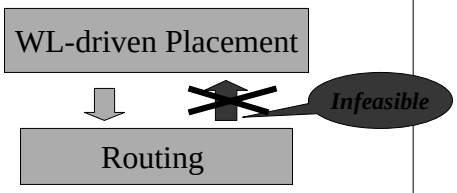
## Outline

- **Routability and the placement context**
- **Placement by recursive bisection**
- **UCLA Capo placer**
- **Empirical results**
- **Conclusions and open questions**

## Routability is a Requirement For Placement

- **VLSI placement is fixed-die, followed by routing**
- ***Routing fails ⇒ placement was not useful***
- **Algorithms that produce routable placements *are more valuable* (no fixes ⇒ cleaner EDA)**
- ***Timing-driven placement***
  - **does not excuse the routability requirement**
  - ***is a harder* problem, not a different problem**
- ***Question: can we achieve routability without increasing wirelength ? (via a better global placer)***

## In This Work:




- **Given**
  - **a circuit**
  - **a** [very good] **black-box router**
- **Definition**
  **a circuit placement is "100% auto-routable"**
  **⇔ the router automatically completes all nets without manual intervention**
- **OUR CRITERION: Less than "100% auto-routability" *==* placement failure**
- **Sets the bar for placer evaluation**

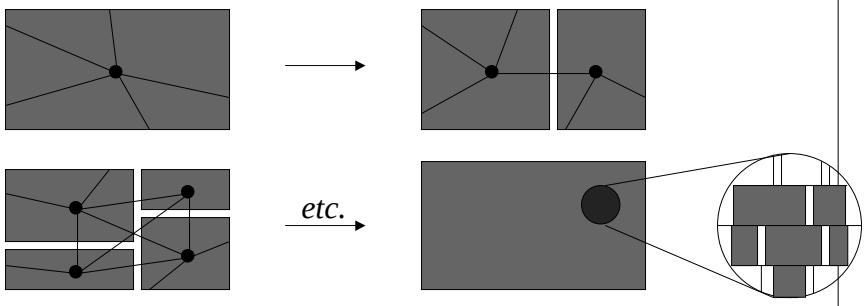
## Fixed-die vs Variable-die

- **Fixed-die P&R (typically, N-layer metal)**
  - **"cell sites" and routing tracks are fixed**
  - **cannot "spread" rows and insert routing tracks**
  - **makes achieving routability much harder**
  - **is implied by modern design techniques**
    - **power/ground planning, hierarchical block methodology**
  - **assumed by Cadence LEF/DEF formats**
    - **assumed by most commercial EDA tools**
- **Variable-die P&R (typically, 2-layer metal)**
  - **row geometries, utilization, area not known in advance**
  - **routability can be traded for area**

## Routability

- **Routability is not a purely "*0-1* property"**
  - **router runtime explodes when routing gets harder**
- **May be *harder* with growing #nets**
  - **need to use large benchmarks (10K cells and up)**
- **May be *easier* with increased # metal layers**
  - **need to use very recent benchmarks**
- **Experimental question: Does decreasing overall WL improve routability and routed wirelength ?**

## Recursive Bisection Placement



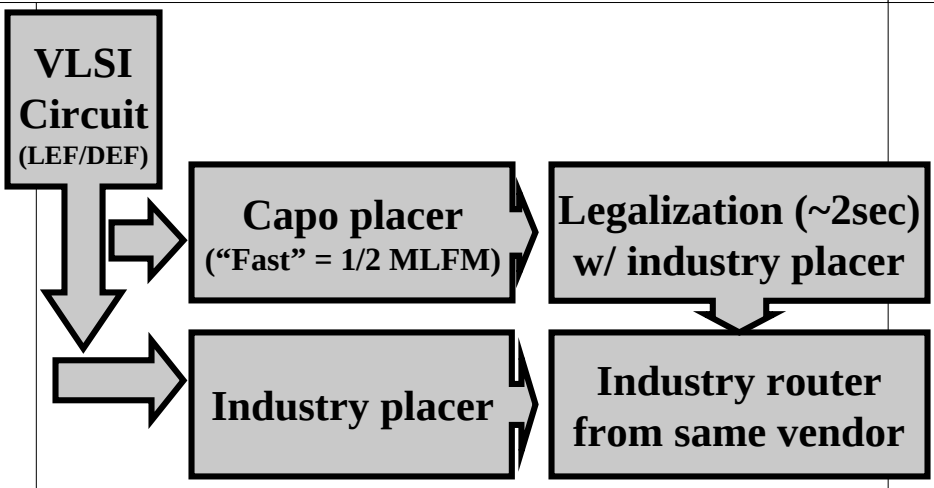

## Recursive Bisection (RB) Placement

- Framework for leading commercial tools
  - fast and scalable
  - can be extended to handle timing
- Key technologies
  - balanced hypergraph bipartitioning
    - UCLA MLPart (Caldwell/Kahng/Markov ASPDAC 2000)
  - end-case processing
    - optimal methods (Caldwell/Kahng/Markov ISPD '99)
- RB placement vastly improved in the last 2 years
  - due to the *multi-level partitioning breakthrough*
- Experimental Q: does better RB improve routability?

## UCLA Capo Placement Tool

- Open-source: vlsicad.cs.ucla.edu, openeda.org
- Employs recent advances in recursive bisection
- This paper: nothing used beyond recursive bisection
- Improved "flat" Fiduccia-Mattheyses (FM)
  - ⊙ better performance for small partitioning tolerance
  - ⊙ VLSI Design 2000
- Better Multi-Level Fiduccia-Mattheyses (MLFM)
  - ⊙ improves upon `hMetis` (DAC `97), faster
  - ⊙ ASP DAC 2000
- New block splitting heuristics for vertical cuts
  - → "easier" partitioning instances and increased solution space
- Hierarchical tolerance computation (UCLA TR-200002)

## Experimental Flow




Computed: HPWL, WWL, routed WL and runtimes

| Test case | Cells | Nets | White space | #Metal Layers | Placer | HPWL x1e6 | WWL x1e6 | Place time | Routed WL x1e6 | Route time |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 11471 | 11828 | 24.30% | 3 | Industrial | 2.8 | 3.22 | 182 | **3.43** | 223 |
| + | | | | | UCLA Capo | 2.68 | 3.05 | 269 | **3.3** | 293 |
| | | | | | Capo-Fast | 2.72 | 3.03 | 131 | **3.38** | 336 |
| 2 | 19832 | 22974 | 9.90% | 6 | Industrial | 1.24 | 2.53 | 520 | **2.49** | 833 |
| + | | | | | UCLA Capo | 1.28 | 2.36 | 473 | **2.16** | 500 |
| | | | | | Capo-Fast | 1.31 | 2.12 | 270 | **2.22** | 502 |
| 3 | 20392 | 25634 | 14.20% | 2 | Industrial | 5.93 | 7.7 | 414 | **7.9** | 613 |
| + | | | | | UCLA Capo | 5.6 | 7.18 | 471 | **7.52** | 579 |
| | | | | | Capo-Fast | 5.76 | 7.25 | 239 | **unroutable!** | 3840 |
| 4 | 25995 | 28603 | 8.70% | 3 | Industrial | 10.8 | 13.6 | 427 | **17** | 5382 |
| - | | | | | UCLA Capo | 10.3 | 12.8 | 837 | **17.9** | 6346 |
| | | | | | Capo-Fast | 10.2 | 12.5 | 394 | **17.8** | 5558 |
| 5 | 33917 | 39152 | 29.40% | 4 | Industrial | 5.89 | 7.29 | 990 | **7.9** | 3502 |
| **- infty** | | | | | UCLA Capo | 5.73 | 6.88 | 1197 | **unroutable!** | 4196 |
| | | | | | Capo-Fast | 5.67 | 6.73 | 621 | **unroutable!** | 7355 |
| 6 | 35549 | 44121 | 0.10% | 4 | Industrial | 9.67 | 12.3 | 1765 | **11.8** | 1120 |
| + | | | | | UCLA Capo | 9.3 | 11.4 | 1546 | **11.1** | 1050 |
| | | | | | Capo-Fast | 9.43 | 11.5 | 649 | **11.7** | 1055 |
| 7 | 42352 | 44490 | 29.30% | 5 | Industrial | 37.1 | 47.7 | 981 | **44.3** | 624 |
| = | | | | | UCLA Capo | 36.2 | 45.5 | 1154 | **44.5** | 615 |
| | | | | | Capo-Fast | 34.8 | 45.1 | 657 | **46** | 701 |

# What About MCNC Benchmarks?

- **Too old for meaningful routability evaluation**
  - **> 10 years old, no longer representative (Alpert 98)**
  - **row-based layouts use *variable-die***
- **Most published WLs are unreliable**
  - **solutions not available**
  - **different row configurations used**
  - **some placers place pads (on the boundary?)**
  - **some assume given pad locations (which ones?)**
- **Capo runs on MCNC benchmarks (Bookshelf format)**
  - **you can download Capo, run it and see solutions**
  - **runtimes on a single Pentium III Xeon @550MHz**
    - **avqlarge ( 25K cells, 33K nets) – 4.5min**
    - **golem3 (100K cells, 217K nets) – 37 min**

## What Did Not Work For Us…

- "Overlapping" with bisections
- Fancy terminal propagation
- Explicit top-down look-ahead
- Improvements using analytical placement
  - quadratic wirelength
  - linear wirelength
- Using name-based hierarchies
  - improvement on one example out of many
    - the circuit was unusually hard to partition
    - possible interpretation: need to improve partitioner
- "Placement Vcycling"

## Conclusions

- Capo placer is scalable and competitive
  - freely available for research and commercial use
- Better recursive bisection ⇒ better routability
  - improving RB *still* makes sense (e.g., as proposed)
  - improving min-cut partitioning *still* makes sense
- Weighted wirelength is not a good objective
- "better HPWL ⇒better routability" - not clear!
  - We draw conclusions only about min-cut
  - Folklore: analytical placements are hard to route
  - Other WL minimization techniques may be better *or worse*
  - Min-cut *may be* a better objective !

## Open Questions

- **Need** transparent **routability improvement**
  - **not to affect wirelength of routable placements**
- **Is recursive bisection** [done right] **still the best method ?**
  - **you can download Capo and compare**

  http://vlsicad.cs.ucla.edu/GSRC/bookshelf/Slots/Placement