Modern Physical Design:
Algorithm
Technology
Methodology
(Part III)
Stan Chow Ammocore
Andrew B. Kahng UCSD
Majid Sarrafzadeh UCLA

Goals of the Presentation

- Various methodologies and interaction between sub-tools
- Classification of Techniques
PART III: Interaction with Upstream Floorplanning and Logic Synthesis

- Interaction classification
  - Definitions
  - Pros and Cons

Design Trend

DSM: Design Global Wires

Local wires
Global wires

DSM: Crosstalk

Interconnect Geometry

Coupling vs. Substrate capacitance

Observation 1

- Deep Sub-micron (DSM) is a problem
- All facets of design are getting more complex (continuously)
- Therefore, we need to make continuous (means incremental?) improvement to tools/design methods.

SOC / DSM Design Dilemma

Accuracy

Abstraction

Require detailed analyses to understand physical interactions

Need abstraction levels to manage complexity
Feature Size and Iterations

IC/ASIC Place & Route Iterations by Process Geometry, North America 1999

Clock Speeds and Iterations

IC/ASIC Place & Route Iterations by Highest Digital Clock Speed
North America, 1999

Sources:
Collett 3rd; 1999 IC/ASIC Physical Design & Layout Verification Study.
Data based on 224 North American IC/ASIC product development teams.
Observation 2

- Therefore, we need to make continuous (means incremental?) improvement to tools/design methods.
- As designs get more complex, number of iteration increases rapidly.
- Incremental (or no) improvement to tools will work (if we are willing to wait 365 days for the result)…. And then the marketing tells us that we need a new feature.
An Easy Solution: Re-use cores

Hard IP
- The easiest path to SoC (?)
- Hard blocks makes the assembly more difficult
  - see results in the next two slides
- No resizing capability to fix timing during assembly
**Soft IP**

- “Soft” IP will allow better Global Optimization
- Final assembly may solve shape, pin, and global timing problems causing reduced design iterations.

**Prediction/Construction heuristics**

- Balance with respect to area and flexibility (if there are a lot of flexible/soft IP).
### Results

<table>
<thead>
<tr>
<th>Test String</th>
<th>Cost</th>
<th>Time (sec)</th>
<th>Cost % Diff</th>
<th>Time (sec)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>ind1(10)</td>
<td>12039</td>
<td>177</td>
<td></td>
<td>12182</td>
<td>1.18</td>
</tr>
<tr>
<td>ind1(20)</td>
<td>12168</td>
<td>173</td>
<td></td>
<td>12324</td>
<td>1.28</td>
</tr>
<tr>
<td>ind1(30)</td>
<td>14971</td>
<td>122</td>
<td></td>
<td>15400</td>
<td>2.86</td>
</tr>
<tr>
<td>ind1(50)</td>
<td>16033</td>
<td>88</td>
<td></td>
<td>17578</td>
<td>9.64</td>
</tr>
<tr>
<td>hway(10)</td>
<td>70571</td>
<td>3420</td>
<td></td>
<td>69611</td>
<td>-1.36</td>
</tr>
<tr>
<td>hway(20)</td>
<td>71838</td>
<td>3515</td>
<td></td>
<td>70448</td>
<td>-1.93</td>
</tr>
<tr>
<td>hway(30)</td>
<td>72274</td>
<td>3524</td>
<td></td>
<td>71900</td>
<td>-0.51</td>
</tr>
<tr>
<td>hway(50)</td>
<td>77653</td>
<td>2564</td>
<td></td>
<td>78696</td>
<td>1.34</td>
</tr>
<tr>
<td>fract(10)</td>
<td>131431</td>
<td>15651</td>
<td></td>
<td>128388</td>
<td>-2.32</td>
</tr>
<tr>
<td>fract(20)</td>
<td>137044</td>
<td>12803</td>
<td></td>
<td>130984</td>
<td>-4.42</td>
</tr>
<tr>
<td>fract(30)</td>
<td>137084</td>
<td>14694</td>
<td></td>
<td>135869</td>
<td>-0.88</td>
</tr>
<tr>
<td>fract(50)</td>
<td>144072</td>
<td>9268</td>
<td></td>
<td>145392</td>
<td>0.91</td>
</tr>
<tr>
<td>prim1(10)</td>
<td>831329</td>
<td>110491</td>
<td></td>
<td>832365</td>
<td>1.2</td>
</tr>
<tr>
<td>prim1(20)</td>
<td>867690</td>
<td>100010</td>
<td></td>
<td>862657</td>
<td>-0.6</td>
</tr>
<tr>
<td>prim1(30)</td>
<td>870456</td>
<td>95299</td>
<td></td>
<td>871623</td>
<td>0.13</td>
</tr>
<tr>
<td>prim1(50)</td>
<td>897120</td>
<td>68303</td>
<td></td>
<td>931694</td>
<td>3.85</td>
</tr>
<tr>
<td>prim2_s1(10)</td>
<td>230703</td>
<td>11899</td>
<td></td>
<td>215193</td>
<td>-6.72</td>
</tr>
<tr>
<td>prim2_s1(20)</td>
<td>235694</td>
<td>11141</td>
<td></td>
<td>217542</td>
<td>-7.70</td>
</tr>
<tr>
<td>prim2_s1(30)</td>
<td>248317</td>
<td>9306</td>
<td></td>
<td>229349</td>
<td>-7.46</td>
</tr>
<tr>
<td>prim2_s1(50)</td>
<td>249489</td>
<td>7445</td>
<td></td>
<td>248566</td>
<td>-0.37</td>
</tr>
<tr>
<td>prim2_s2(10)</td>
<td>323017</td>
<td>38416</td>
<td></td>
<td>283188</td>
<td>-12.33</td>
</tr>
<tr>
<td>prim2_s2(20)</td>
<td>319897</td>
<td>30062</td>
<td></td>
<td>301768</td>
<td>-5.67</td>
</tr>
<tr>
<td>prim2_s2(30)</td>
<td>333313</td>
<td>29632</td>
<td></td>
<td>308023</td>
<td>-7.59</td>
</tr>
<tr>
<td>prim2_s2(50)</td>
<td>354387</td>
<td>21045</td>
<td></td>
<td>323188</td>
<td>-8.80</td>
</tr>
</tbody>
</table>

### Divide and Conquer

- Divide the Problem into Smaller Sub-Problems
- Solve Each of these Separately
- Stitch together the Solutions of the Sub-Problems

10 Million Gate Design => 200 (50k Gate Designs)
Divide and Conquer

- Divide into Logical/Physical Blocks
  - Particularly emphasize the floorplan
  - Iterations between different tools
- Traditional floorplan
  - No flexibility to fix timing problems caused by long wires
  - Overly constrained timing budgets
  - Adds many buffers and oversizes gates on critical paths

Sequential Methodology

- Try to Solve the Problem in Sequential Steps
- Try to Optimize One Functionality at a Time.
  - Optimize Number of Gates at the Logic Synthesis Level
  - Optimize Wire Lengths during Placement
  - Optimize Clock Skew After Placement is Done
  - Optimize Crosstalk during Routing
Sequential Methodology

- Predictive (we lack good predictors)
- Iterative (takes a long time)
- Non-converging (counter productive)

Nets that meet timing in one iteration may fail in the next iteration.

Observation

- There are many “equally good” placement and routing solution. A small change in one, will change the whole things.
- So, cannot trust wire-load models
Traditional Workarounds

- Pessimistic approach
  - For 50K block size, use wire-load model for 100K instead
  - Nets are over-driven
  - Wastes power and area, but reduces number of nets that need fixing after phys design
  - Assumes timing can be met with the pessimistic model (not always the case)
- Over-constrained approach
  - For 80 MHz design, synthesize at 100 MHz
  - After physical design, reset to 80 MHz
  - Nets between 80-100 MHz will “pass”
- Multiple-iteration approach
  - Annotate timing info and phys design info into P&R and synthesis
  - Optimization attempts to minimize changes to accuracy of phys design (usually can’t do)

Semi-Sequential Methodology

- Lots of logic move followed by lots of placement move
- Some logic moves followed by some placement moves
Low Congestion: some logic activities to CORRECT synthesis mistakes

High Congestion: lots of logic activities (panic mode)
Simultaneous Methodology

- “combine” placement and synthesis (& other steps)
- We need to find the right type and location of the move.

Proof of Simultaneous Methodology

- Obviously, the most knowledgeable set of moves
- haven’t been done in the past because
  - history
  - algorithmic complexity
  - need

Timing Optimization

Gate Delays as well as Interconnect delays needs to be an essential part of the design process.

Static Timing Analysis needs to be integrated into the optimization process.
**Timing Analysis**

How do we get the delay numbers on the gate/interconnect?

---

**Timing Metrics**

- How do we assess the change in a delay due to a potential move during physical design?
- Whether it is channel routing or area routing, the problem is the same
  - translate geometrical change into delay change
Iterative Placement

- A placement move changes the interconnect capacitance and resistance of the associated net.
- A net topology approximation is required to estimate these changes.

Placement Algorithms
VLSI Design Flow and Physical Design Stage

Definitions:
- **Cell**: a circuit component to be placed on the chip area. In placement, the functionality of the component is ignored.
- **Net**: an electronic wire to connect several cells.
- **Netlist**: a set of nets which contains the connectivity information of the circuit.

A flow chart for a typical VLSI design.

Placement Problem

A bad placement

A good placement
Global and Detailed Placement

In global placement, we decide the approximate locations for cells by placing cells to global bins. In detailed placement, we make some local adjustment to get the final non-overlapping placement.

Traditional Approaches

- Quadratic Placement
- Simulated Annealing
- Bi-Partitioning
- Quadrisection
- Force Directed Placement
- Hybrid
Congestion Map for a Wirelength Optimized Placement

Routing Algorithms
**Maze Routing/Shortest Path**

Label Source $s = 0$
Label Adjacent grid points using $i + 1$
Get Shortest path to target $t$

---

**Routing**

- Requirements for the DSM Router:
  - N-layer shape-based router
  - Supports gridless and gridded routing
  - Variable wire width for optimal delay constraints
  - Cross-talk avoidance, antenna effects
  - Clock tree sizing for tree balancing
  - Power routing sizing for voltage drop and electromigration
  - Power and clock routing resources reserved early
  - Activity-based optimization
Typical Timing-Driven Approaches

- Net weighting to prioritize timing critical Paths
- Reduction of entire net length

Placynthesis:
Simultaneous Logic/Placement Approach

- Logic optimization concurrently with placement
- Net "placement" with gate placement
Some Placynthesis Moves

- Resizing
- Buffering
- Cloning
- Restructuring

More Placynthesis Moves
Commercial Integration Approach:

- Integrate synthesis with phys design
  - Cadence (Envisia Synthesis, 9/1999)
    - Physically Knowledgeable Synthesis (PKS)
  - Synopsys ("PhysOpt")
  - Monterey ("Dolphin")
  - Magma ("???").

Many other Design Metrics: Power Supply and Total Power

Source: The Incredible Shrinking Transistor, Yuan Taur, T. J. Watson Research Center, IBM, IEEE Spectrum, July 1999
Dual Voltages: A harder problem

- Layout synthesis with dual voltages: major geometric constraints

H -- High Voltage Block
L -- Low Voltage Block

Layout Structure

Cell Library with Dual Power Rails

Conclusion

- Deep Sub-micron (DSM) problems are here and are real
- Traditional Physical Design and Logic Synthesis Algorithms do not work
- Innovation (in algorithms, methodology, tools, etc) needed in all facets.