Exploring High Throughput Computing Paradigm for Global Routing

Yiding Han, Dean Michael Ancajas, Koushik Chakraborty, Sanghamitra Roy
USU BRIDGE Lab
Routing

• Problem
  – Given a placement, and a fixed number of metal layers, find a valid pattern of horizontal and vertical wires that connect the terminals of the nets
  – Levels of abstraction:
    • Global routing
    • Detailed routing

• Objectives
  – Cost components:
    • Congestion (relates to routability)
    • Area (channel width)
    • Wire delays
    • Number of layers (fewer layers $\rightarrow$ less expensive)
    • Execution time
    • Additional cost components: number of bends, vias
Routing Anatomy

Top view

3D view

Symbolic Layout

Note: Colors used in this slide are not standard
Global vs. Detailed Routing

• **Global routing**
  – Input: detailed placement, with exact terminal locations
  – Determine “channel” (routing region) for each net
  – Objective: minimize area (congestion), and timing (approximate)

• **Detailed routing**
  – Input: channels and approximate routing from the global routing phase
  – Determine the exact route and layers for each net
  – Objective: valid routing, minimize area (congestion), meet timing constraints
  – Additional objectives: min via, power
Global Routing

• Increasing complexity
  – Explosion in design rules
  – Additional constraints

• Exploit parallelism to tackle complexity
  – Multi-core solution
    • Liu DAC ‘10, PGRIP DAC ‘10
  – Massive parallel architecture (GPU)
    • This work.
Parallel Global Routing

• Partition-based Parallelism
  – Cut routing grid into partitions
  – Route each partition in parallel
  – Load balancing
  – Cut on long nets

• Net-level Parallelism
  – Individual nets divided across many threads
  – Route nets simultaneously, good load balancing
  – Shared resources among nets cause collision
  – Collision degrades route quality, convergence speed
Exploit net-level parallelism

Routing Map:

- **Parallel Route WITHOUT net dependencies**
  - Schedule subnets with original order
  - Collision

- **Parallel Route WITH net dependencies**
  - Identify concurrent subnets
  - Collision-free route

Collisions

Task Queue:

<table>
<thead>
<tr>
<th>Subnet</th>
<th>Subnet</th>
<th>Subnet</th>
<th>Subnet</th>
<th>Subnet</th>
<th>Subnet</th>
<th>Subnet</th>
<th>Subnet</th>
<th>Subnet</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
Design Flow

Start
- Project to 2D-layer Design
- FLUTE Net Decomposition
- Edge Shifting
- Initial Routing

Post Refinement
- Layer Assignment
- Finish

Congestion Level Identification
- All levels done?
  - Yes
    - Congested Region Identification
    - Concurrent Net Identification
    - Task Scheduling
  - No
    - Rip-up and Re-Route
      - CPU: A* Search
      - GPU: Lee Algorithm

Overflow?
• *Scheduler*, *router* threads execute in parallel
• *Scheduler* dynamically populates the GPU and CPU task pool.
Scheduler

• Build net dependency tree
  – Routing order dictates dependencies among routing tasks.

• Identifies concurrent subnets
  – Subnets without overlapping routing region.

• Dynamically examine the dependencies among nets
  – Release concurrent workload on-the-run.
Net Dependency Construction

- Tile coloring to uncover parallelism in ordered subnet queue
Scheduler Implementation

- Recursively uncover net dependency
- Producer-consumer relation with routing threads
- Optimizations
  - Reduce search area: only search within the congested region
  - Scheduling window: restrict the number of nets being examined
• Breadth-First Search on GPU.
• Exhaust searching frontiers in routing region to find lowest cost path.
Evaluation Platform

• CPU: Intel Q6600 Kentsfield
  – Quad-core 2.4GHz
  – Cache 2 x 4 MB

• GPU: Nvidia GeForce GTX 470 (Fermi)
  – 448 Cuda cores
  – 1280 MB GDDR5
  – Driver Ver. 270.41.19
  – CUDA Toolkit 4.0

• System Memory: 8GB DDR2
• OS: CentOS 5.6 Final (64-bit)
Results

Speedup compared to NTHU-Route 2.0 on ISPD benchmarks

- 4-core
- GPU + 4-core

Adaptec
- Newblue
- Bigblue
Conclusion

• GPU-CPU hybrid global router to exploit concurrency.
• Novel scheduler design to dynamically uncover net-level parallelism.
• Promising speedups with comparable solution quality
Questions