FastGR : Global Routing on CPU-GPU with Heterogeneous Task Graph Scheduler

Siting Liu$^{1,2}$, Peiyu Liao$^{1,2}$, Rui Zhang$^3$, Zhitang Chen$^4$, Wenlong Lv$^4$, Yibo Lin$^1$, Bei Yu$^2$

$^1$Peking University  $^2$Chinese University of Hong Kong  $^3$HiSilicon Technologies Co.  $^4$Huawei Noah’s Ark Lab
Outline

Introduction

Methods

Results

Summary
Outline

Introduction

Methods

Results

Summary
Physical Design

- System Specification
- Architectural Design
- Functional Design and Logic Design
- Circuit Design
- Physical Design
- Physical Verification and Signoff
- Fabrication
- Packaging and Testing
- Chip

Partitioning

Chip Planning

Placement

Clock Tree Synthesis

Signal Routing

Timing Closure
Problem Formulation

Given a placement, a netlist and technology information,

- determine the necessary wiring, e.g., net topologies and specific routing segments, to connect these cells
- while respecting constraints, e.g., design rules and routing resource capacities, and
- optimizing routing objectives, e.g., minimizing total wirelength and reducing congestion.

Netlist

\[ N_1 = \{C_4, D_6, B_3\} \]
\[ N_2 = \{D_4, B_4, C_1, A_4\} \]
\[ N_3 = \{C_2, D_5\} \]
\[ N_4 = \{B_1, A_1, C_3\} \]
Problem Formulation

We use a set of global routing cells (G-cells) with a group of evenly distributed horizontal and vertical grids to represent the global routing region.

A grid graph $G(V, E)$ can be defined by treating each G-cell as a vertex ($v \in V$) and creating an edge ($e \in E$) between every two adjacent G-cells. The edge $e$ has two types: wire edge and via edge.

Global routing is a minimum cost path searching problem on $G(V, E)$. 
FastGR: Motivation

- CPU: Strong controller and ALUs.
- GPU: Grid-based computation resources: max 1024 threads per block.
- GPU: Cheap synchronization within blocks.

- 10+ metal layers.
- Millions of nets.
- Variable objectives and constraints.
FastGR: Motivation - Runtime Breakdown

Runtime breakdown of a modern global router; PATTERN means the pattern routing stage while MAZE means the maze routing stage.
FastGR: Overall Contribution

FastGR is a global routing framework that is accelerated on CPU-GPU platforms with a task graph scheduler and GPU-accelerated algorithms.

- A high-performance task graph scheduler to distribute CPU and GPU tasks for workload balancing and efficiency.
- A novel GPU-accelerated pattern routing algorithm that can route a batch of nets leveraging the massive parallelism on GPU.
Outline

Introduction

Methods

Results

Summary
Parallelism Among Multi-pin Nets

Sample for the bounding box of multi-pin nets.

Sample of 5 levels scheduling.

Conflict graph of the sample.

Sample of 3 levels scheduling.
Batch Scheduler

1. Sort all nets with a sorting strategy.  
   \[ N : \{n_1, n_2, n_3, n_4, n_5, n_6, n_7\}; B : \{\} \]

2. Takes the first net out. 
   \[ N : \{n_2, n_3, n_4, n_5, n_6, n_7\}; B : \{\{n_1\}\} \]

3. choose the independent set. 
   \[ N : \{n_2, n_3, n_4, n_5, n_7\}; B : \{\{n_1, n_6\}\} \]

4. repeat step 2 and step 3. 
   \[ N : \{\}; B : \{\{n_1, n_6\}, \{n_2, n_4, n_7\}, \{n_3, n_5\}\} \]

---

All the tasks are split into two parts:

- Root task batch.
- Non-root task batch.

Only two conditions exist between each pair of conflicting tasks.

- **One task is in the root batch, and the other is not.** The order is from the task in the root batch to the other.

- **Both the tasks are not in the root batch.** The order is from the task with a smaller task ID to the other. Note that the task ID represents the sorting result of all the tasks.
FastGR: Methods - GPU-accelerated Pattern Routing

- the wire connecting $P_s$ and the bend point $B$;
- the vias connecting different metal layers through the bend points $B$ and $B'$;
- the wire connecting the bend point $B'$ and $P_t$. 

Step 1

Step 2

Step 3

Step 4
FastGR: Methods - GPU-accelerated Pattern Routing

Let $L$ be the number of metal layers. $c_i$ be the $i$th step’s cost result vector, with the $j$th entry $c_{i,j}$, $0 < j \leq L$. $c_0$ is a zero vector.

$w_i$ represents the edge weights for step $i$ with size $L$, $i \in \{1, 2, 4\}$.

$W$ means the weights matrix for step 3 with size $L \times L$, where $W_{i,j}$ represents the via costs between the $i$th layer and the $j$th layer.

Step 1, 2, 4:

$$c_i = c_{i-1} + w_i, \quad i \in \{1, 2, 4\}.$$
Let $L$ be the number of metal layers. $c_i$ be the $i$th step’s cost result vector, with the $j$th entry $c_{i,j}$, $0 < j \leq L$.

$c_0$ is a zero vector.

$w_i$ represents the edge weights for step $i$ with size $L$, $i \in \{1, 2, 4\}$.

$W$ means the weights matrix for step 3 with size $L \times L$, where $W_{i,j}$ represents the via costs between the $i$th layer and the $j$th layer.

Step 3:

$$c_{3,t} = \min_{0 < s \leq L} \{c_{2,s} + W_{s,t}\}.$$  

(2)
Each block processes one multi-pin net.

Each multi-pin net includes several two-pin nets.

Each two-pin net has 2 candidate 2D paths ($L \times L$ candidate 3D paths) with L-shape, where $L$ is the number of metal layers.
Experimental Setting

- Pattern routing stage: Task graph scheduler & GPU-friendly pattern routing algorithm.
- Maze routing stage: Task graph scheduler.
- Device: a 64-bit Linux machine with Intel Xeon 2.2 GHz CPU and one GeForce RTX 2080 GPU.
Our GPU-friendly pattern routing algorithm can contribute $10.877 \times$ speedup over the sequential algorithm on CPU.

The task scheduler can contribute to $2.307 \times$ speedup over the widely-adopted batch-based parallelization strategy on CPU.
FastGR: Results - Total Runtime & Score

As for the overall speedup, we can achieve \(2.426\times\) acceleration without solution quality degradation compared with the SOTA global router.
Summary

- Task scheduler plays a very important role in routing problem; Our task graph scheduler alone can contribute $2.307 \times$ speedup on CPU. Also, it can help FastGR to get comparable routing solution quality.

- GPU-accelerated kernel algorithms are vital for the physical design flow; Both the GPU-accelerated pattern routing algorithm and the task graph scheduler can lead to $10.877 \times$ speedup on GPU.
Thank You!