# Global Routing and Parallelism

Roger Caputo, Diego Tumelero, Marcelo Johann, Ricardo Reis

SIM 2013

UFRGS

INF INSTITUTO DE INFORMÁTICA UFRGS PPGC

UFRGS PGMICRO

# Outline

- Introduction
- Background
- Most famous (academic) global routers
- Parallelizing the Global Routing
- Graphics Processing Units
- Work developed at the UFRGS
- Open challenges

# Background

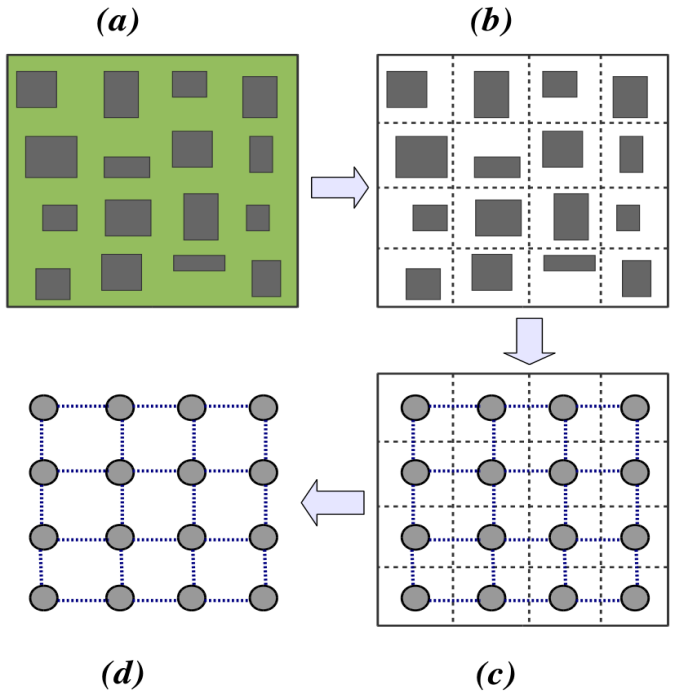
Placement ⇒ **GR** ⇒ Detailed Routing

Distributes the interconnections of the netlists

Defines routing regions

Satisfying constraints like

- Timing
- Wirelength
- Overflow
- Runtime



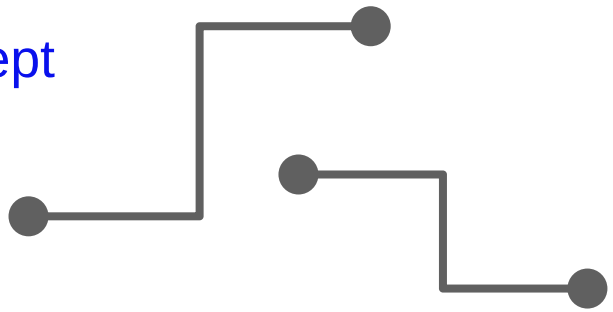
# Basic Algorithms

## Maze routing

- Brute-force method
- Employs BFS, Dijkstra's algorithm and A*

## Monotonic routing

- Sequential method
- Based on the monotonic function concept



# Basic Algorithms

## Pattern routing

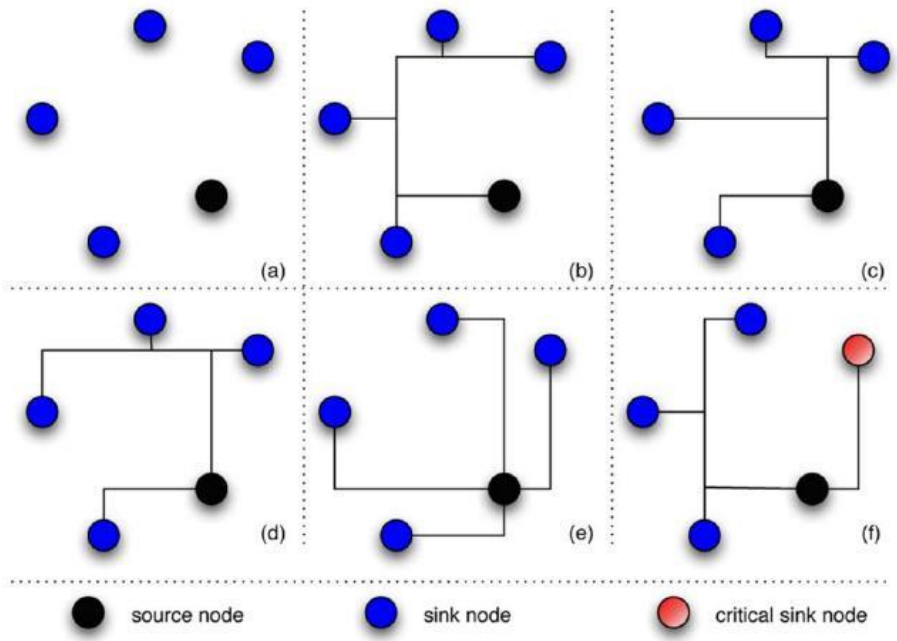
- Used alongside Maze routing
- Makes 'L' and 'Z' paths

## Multi-commodity flow

- Multi-constraint
- Employs ILP

## Steiner trees

- For multi-pin nets
- Topologies to reduce delay



# Most Famous Global Routers

## Fast Route, 2.0 and 4.0

- Deforms a Hanan grid during Steiner tree generation
- Enhanced with monotonic routing and maze routing

## BoxRouter and BoxRouter 2.0

- Progressively expands an initial box
- ILP considering L-shapped patterns
- Improved with A* and *rip-up and re-route*

# Most Famous Global Routers

- FGR
  - Based on Path-finder router for FPGAs
  - Performs fast layer assignment
- MaizeRouter
  - Edge shifting and Edge retraction
  - Sequential approach based on Interdependent net decomposition
- NTHU-Route 2.0
  - Rip-up and re-route
  - History-based cost function to distribute overflow

# Parallelizing the GR

**GR could be accelerated using various processors**

The success of parallelism falls on the algorithmic efficiency?

Scalable algorithm does not mean high performance

Sometimes serial approach is better than the parallel!

# Parallelizing the GR

Routing every net independently

- Not all nets are independent
- Sequential blocks are generated

Partitioning the circuit

- How to partition the circuit?
- How to interconnect the boundaries parts?

# Graphics Processing Units

Why to use GPUs?

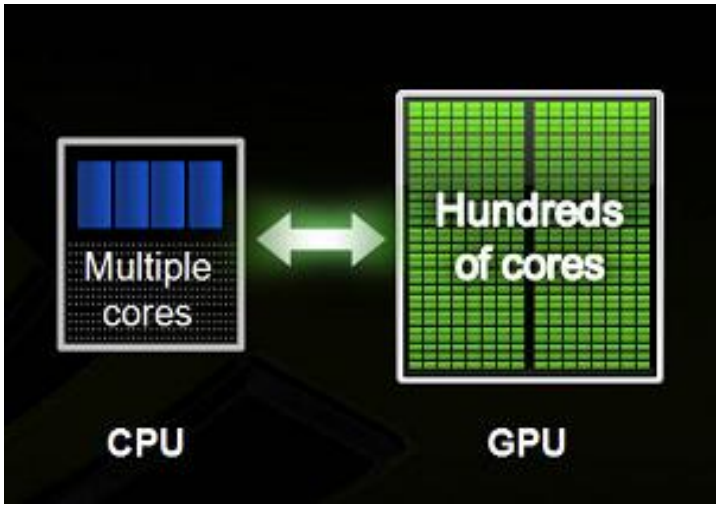
- Pollack's rule ($performance \approx \sqrt{complexity}$)
- 1 GPU ≈ 10 CPUs

Hundreds of cores

GR can be a SIMD task

High throughput memory

Gustafson's law



# Graphics Processing Units

## CUDA

- Better performance
- Only on nVIDIA products


## OpenCL

- CPUs, DSPs, GPUs, FPGAs...
- The performance is “bounded by the lower”

# Work developed at the UFRGS

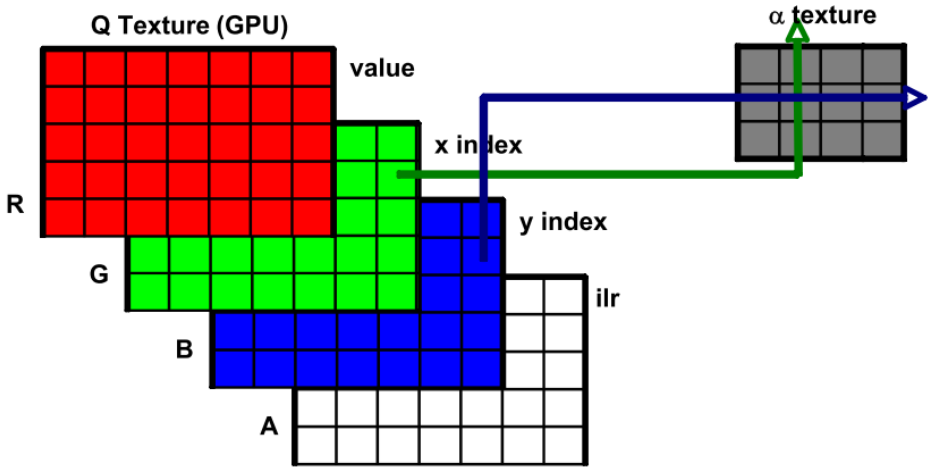
## Flach's placer

GPUs are used to treat the cell placement problem.

Implements a quadratic placement on *OpenGL.*

Execute matrix operations on GPUs to manipulate elements of texture.

The results present significant improvements in algebra operations performance.



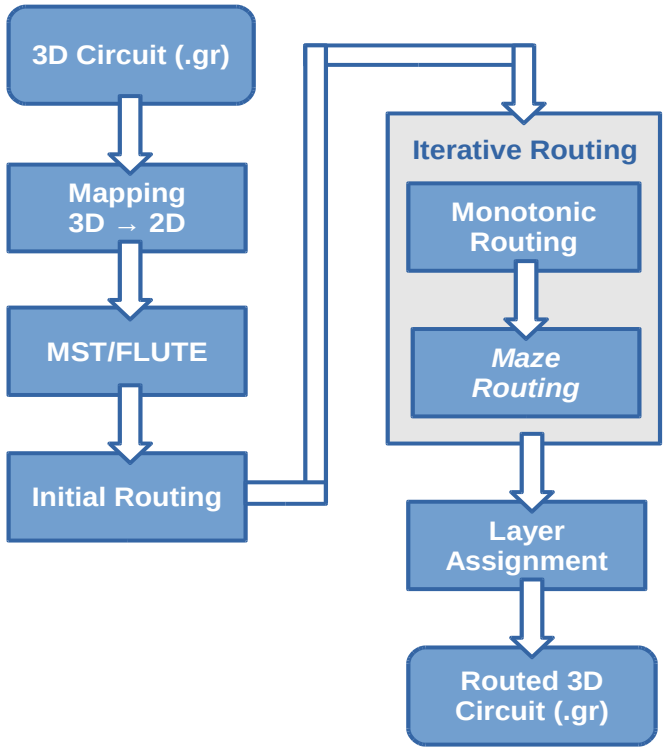
# Work developed at the UFRGS

**Reimann's router**

Implements *rip-up and re-route* technique.

*WL Version* targeting the shortest wirelength

*RT Version* to convergence as fast as possible




Interesting results compared to others from ISPD'08

# Conclusions

GRs are a combination of well-known techniques and algorithms.

GPUs and parallel models are approaches that could uncover excellent solutions to reduce execution time and improve router performance in EDA.

Resources and restrictions must be considered before *going parallel*.

# Global Routing and Parallelism

Roger Caputo, Diego Tumelero, Marcelo Johann, Ricardo Reis

## Questions?

SIM 2013

UFRGS

INF INSTITUTO DE INFORMÁTICA UFRGS PPGC

UFRGS PGMICRO