## Classbased Detailed Routing in VLSI Design

#### C. Schulte, T. Nieberg

Research Institute for Discrete Mathematics University of Bonn

CTW 2008

#### Overview of the Talk

- Introduction to VLSI Routing
- Challenges of the new 65nm Technology
- Offgrid Pinaccess
- Equivalence Classes of Circuits
- First Experimental Results

## Introduction to VLSI Routing

- Chip: Realization of a large boolean function.
  Consists of many circuits implementing simple boolean functions (and, or, xor, ...).
- Circuits contain *pins* for I/O that have to be connected by wires.
- First stage of physical VLSI design: Placement of all circuits of the chip on the chiparea.
- We have a *netlist* which partitions the set of all pins into *nets*.
- Routing task: Connect the pins of each net by wires.
  I.e. for each net find a Steiner tree containing its pins which is disjoint from all other nets.
- Traditionally solved on an incomplete 3-dimensional grid graph obtained by removing space already occupied by circuits and power supply.

## Introduction to VLSI Routing

Feasible Routing:

- Many errors have to be avoided such as minimum distance violations, samenet errors,...
- There are so called *ground rules* which determine how a feasible routing looks like.

Optimization Goals:

There are different optimization goals to be considered: Minimizing netlength, power consumption, production yield...

Problem:

- Included Steiner tree packing problem is NP-hard.
- Practical instances are huge:
  - Over 5 million nets with a total of over 20 million pins.
  - Routing grid with over 100 billion vertices.
  - ▶ Up to *1,000m total wirelength* needed.

# Inroduction to VLSI Routing: Chip Example



## Inroduction to VLSI Routing: BonnRoute

Practical solution implemented in our routing tool *BonnRoute*:

- Build Steiner trees by successively searching shortest paths between different connected components of each net.
- Determine smaller searchspaces for the path searches of each net in a prior *global routing* step.
- Use of a sophisticated interval based Dijkstra algorithm with future cost.

Challenges of the new 65nm Technology

- In former technologies all pins were nicely aligned to the routing grid and could be accessed ongrid.
- In newer technologies ≤ 65nm: Pins are no longer guaranteed to contain any ongrid location and may have to be accessed offgrid.
- Additionally there are many new offgrid rules that restrict the way of accessing pins even more.
- Using a *finer routing grid* would be completely *intractable* because of the already huge instance sizes.

# Offgrid Pinaccess

Idea:

- Still use the grid graph for larger distances and *locally* construct simple, small paths from pins to ongrid locations.
- The endpoints of these offgrid paths are used as source/target points for the ongrid pathsearch.
- After an ongrid path has been found, *combine* it with the two corresponding offgrid paths.

# Offgrid Pinaccess

Simply connecting pins to the nearest ongrid point(s) is not feasible because of ground rule restrictions:



- (a): samenet minspace error, i.e. two shapes of the same net are to close together.
- (b) + (c): short edge error, i.e. *both* of two adjacent edges are shorter than a required minimum length.
  Note: The error in (b) does not depend on the length of the north wire.

Offgrid Pin Access: Path Computation Heuristic

- Determine feasible startingpoints for paths inside the pin (*pinshrink*).
- Restriction to *simple paths* (at most 3 segments).
- Compute *minimum run length* of first path segment depending on short edge and samenet minspace rules.
- Extend the path to the nearest ongrid location, also respecting certain minimum lengths to avoid samenet errors when the ongrid path search connects to the end of the path.
- Delete paths that are illegal because of minspace violations to blockages or other nets.

## Offgrid Pin Access: Pinshrink

- Basic Idea: Compute forbidden access regions with the property that any wire connecting to the pin from a certain direction causes a samenet error regardless of its length if it ends within the region.
- ▶ Pinshrink depends on the direction in which the wire leaves the pin. ⇒ Different *shrinktypes*.
- Shrunk pin: *Substract* forbidden access regions from the pin.









## Offgrid Pin Access: Weaknesses

- Our simple path computation heuristic does not find all feasible paths - especially in areas with high pin density.
   More sophisticated approach needed.
- Problem: *Millions of pins* and for each pin offgrid paths are needed *multiple times*.
- Fortunately many pins occur in equal configurations multiple times on the chip.

Idea:

- Build classes of circuits for which we can use the same offgrid paths.
- Precompute and save paths for each class, use table lookup during routing.

## Equivalence Classes of Circuits

We call two circuits equivalent if they have

- the same internal structure (pins and blockages),
- the same outer blockages (for example power supply wires) intersecting its area,
- the same relative offset to the routing grid,

up to translation and certain kinds of rotation and mirroring.

- ➤ ⇒ Computing offgrid paths for the pins of only one representative of each class is sufficient.
- Paths illegal for one circuit are also illegal for all other circuits of the same class.

## Circuit Class Structure



### Circuit Class Example



 $\Rightarrow$ 



## Classbased Offgrid Pinaccess

#### **During initialization:**

- Build circuit classes.
- Compute legal offgrid paths for each pin of one representative circuit of each class and save them in a table.
- Assign these paths also to the equal pins of the other circuits of the class.

#### Before every ongrid path search during routing:

- For each pin: Look up corresponding offgrid paths in constant time.
- Delete offgrid paths that are illegal in the *current* local routing situation.
- Do resulting ongrid routing.

### First Results: Circuit Classes

	# ckts	# classes	$\frac{\# classes}{\# ckts}$ (%)
chip1	16.791	2.711	16.1
chip2	77.688	14.937	19.2
chip3	90.225	8.629	9.6
chip4	159.835	13.807	8.6
chip5	521.919	13.032	2.5
chip6	649.107	19.652	3.0
chip7	3.439.610	31.711	0.9
chip8	5.234.931	37.842	0.7

## First Results: Classbased Offgrid Pinaccess

chip	#opa	#opa classb.	<sup>#opa classb.</sup> #opa (%)
chip1	56455	11231	19.9
chip2	280623	64387	22.9
chip3	247552	33306	13.5
chip4	564035	62142	11.0
chip5	1863850	52940	2.8
chip6	2439338	129912	5.3

In addition a first complete routing run using class based pinaccess showed:

- ▶ 67% of precomputed offgrid paths are *still legal* at the time they are needed.
- Factor 1,4 overall *runtime improvement* compared to on-the-fly method.

## Conclusion and Future Work

Conclusion:

- Circuit classes allow significant speed-up of offgrid pinaccess.
- ► However, impossible to account for all situations ⇒ local on-the-fly offgrid pinaccess still needed.

Future Work:

- *Exact algorithm* to construct offgrid paths.
- ► *Sophisticated choice* of paths from lookup table.



{schulte, nieberg}@or.uni-bonn.de