A New Global Router for Modern Designs

Jhih-Rong Gao and **Pei-Ci Wu**
Synopsys Inc.

Ting-Chi Wang
National Tsing Hua University

Outline

- Introduction
- Problem Formulation
- Methodology
- Experimental Results
- Conclusion

Outline

- Introduction
- Problem Formulation
- Methodology
- Experimental Results
- Conclusion

Introduction

- Fundamental VLSI design flow
  - Placement results
  - Obtain routing topology by global routing
  - Obtain real routes by detailed routing
- Global router
  - Should provide high quality solution so that detailed router is able to find legal solution
  - A fast global router can be a good routability estimator in placement stage

Contribution

- NTHU-Route – provide high-quality global routing result

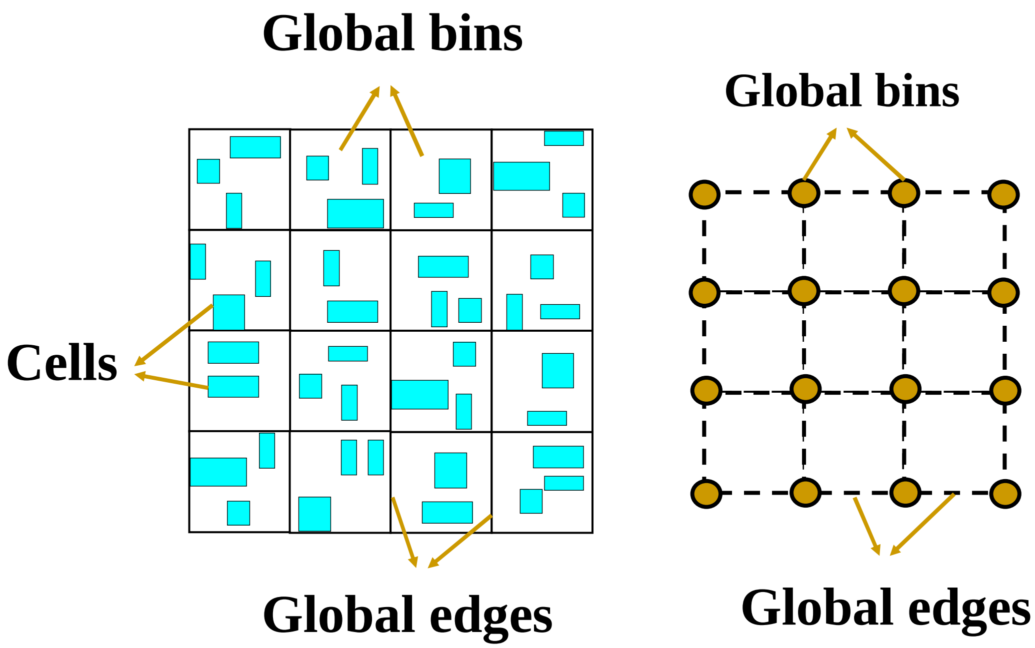
- Iterative rip-up & reroute based global router
- Adaptive multi-source multi-sink maze routing
- Congested region identification method
- Final refinement process for bottleneck

Outline

- Introduction
- Problem Formulation
- Methodology
- Experimental Results
- Conclusion

Problem Formulation

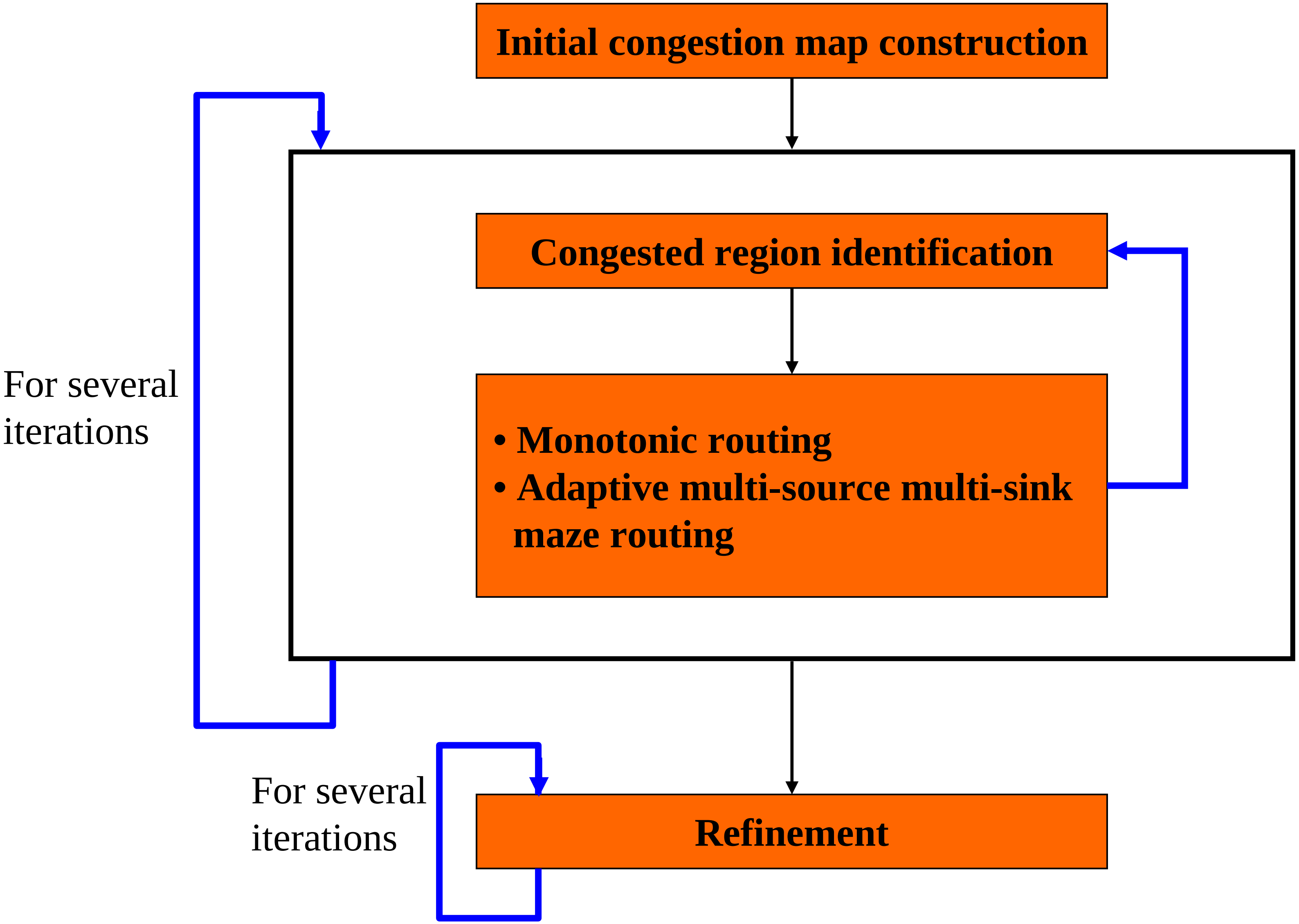
- Input: a set of nets to be routed over a grid graph $G(V, E)$
- Output: Steiner tree topologies for all nets



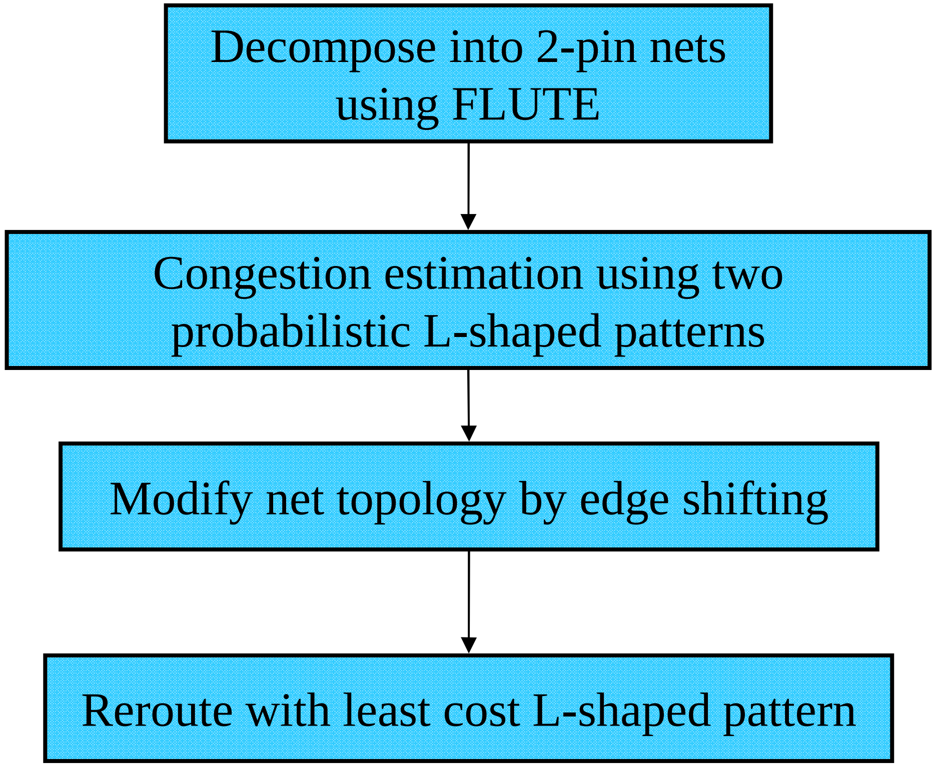
Outline

- Introduction
- Problem Formulation
- Methodology
- Experimental Results
- Conclusion

Our Methodology



Initial Congestion Map Construction



1. FLUTE
   - Generate rectilinear Steiner minimal tree
2. Add ½ or 1 demand to edges
3. Edge Shifting
   - Move some edges to less congested region without increasing wirelength
4. Add 1 demand to edges

Rip-up and Reroute Strategy

- The order of nets to be ripped up & rerouted affects the routing quality very much
- We propose an algorithm to identify congested regions and rip-up & reroute two-pin nets with similar congestion at a time

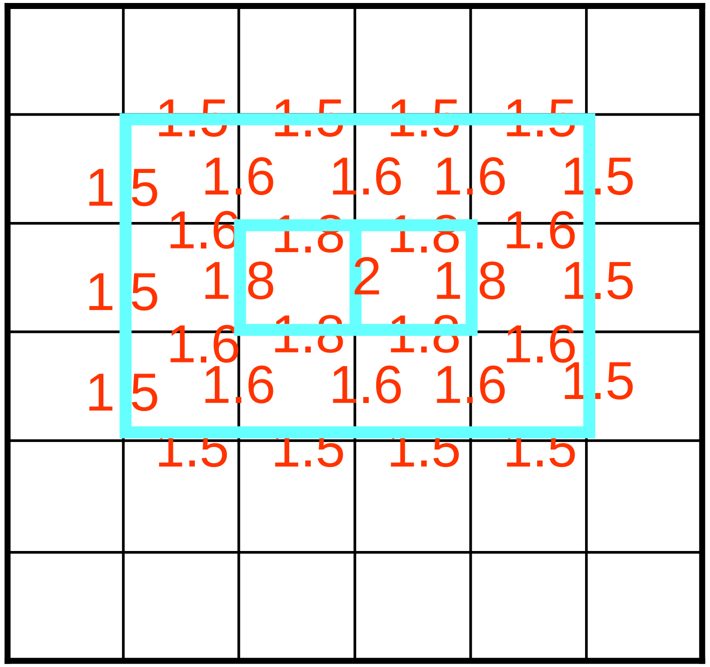
Congested Region Identification

- Partition the interval between the maximum congestion value and 1 into *m* sub-intervals (*m*=10 in our current implementation)
  - Each congested edge belongs to a sub-interval
- For each sub-interval I_k
  - Expand a rectangular region *r* from each edge *e* in I_k until *avg_cong*(*r*) is smaller than the lower bound of I_k

$$avg\_cong(r) = \frac{\sum d(e_i)}{\sum s(e_i)} \qquad e_i \text{ is an edge inside } r$$

  - Find two-pin nets within the region
- For each 2-pin net in increasing order of size of bounding box
  - Monotonic routing
  - Adaptive multi-source multi-sink maze routing if necessary

An Example of Range Identification



- Partition [2, 1]

  →{[2, 1.8), [1.8, 1.6), [1.6, 1.4),…, [1.2, 1)}

History Based Cost Function

$$cost_e = 1 + h_e \cdot p_e$$

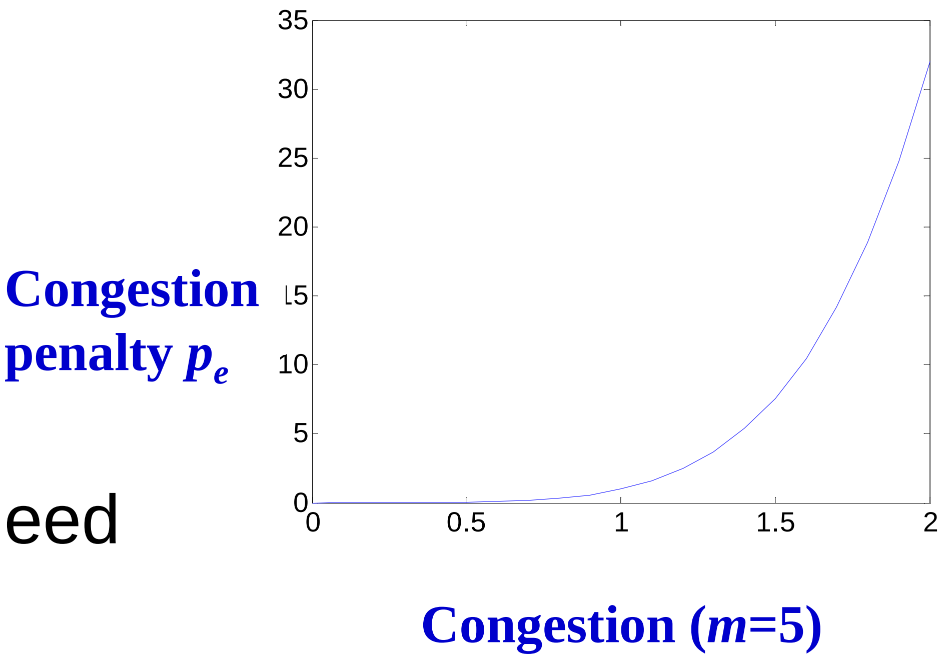
- h_e: the history of congestion on e

$$h_e^{i+1} = \begin{cases} h_e^i + 1 & \text{if } e \text{ has overflow} \\ h_e^i & \text{otherwise} \end{cases}$$

- p_e: resultant congestion

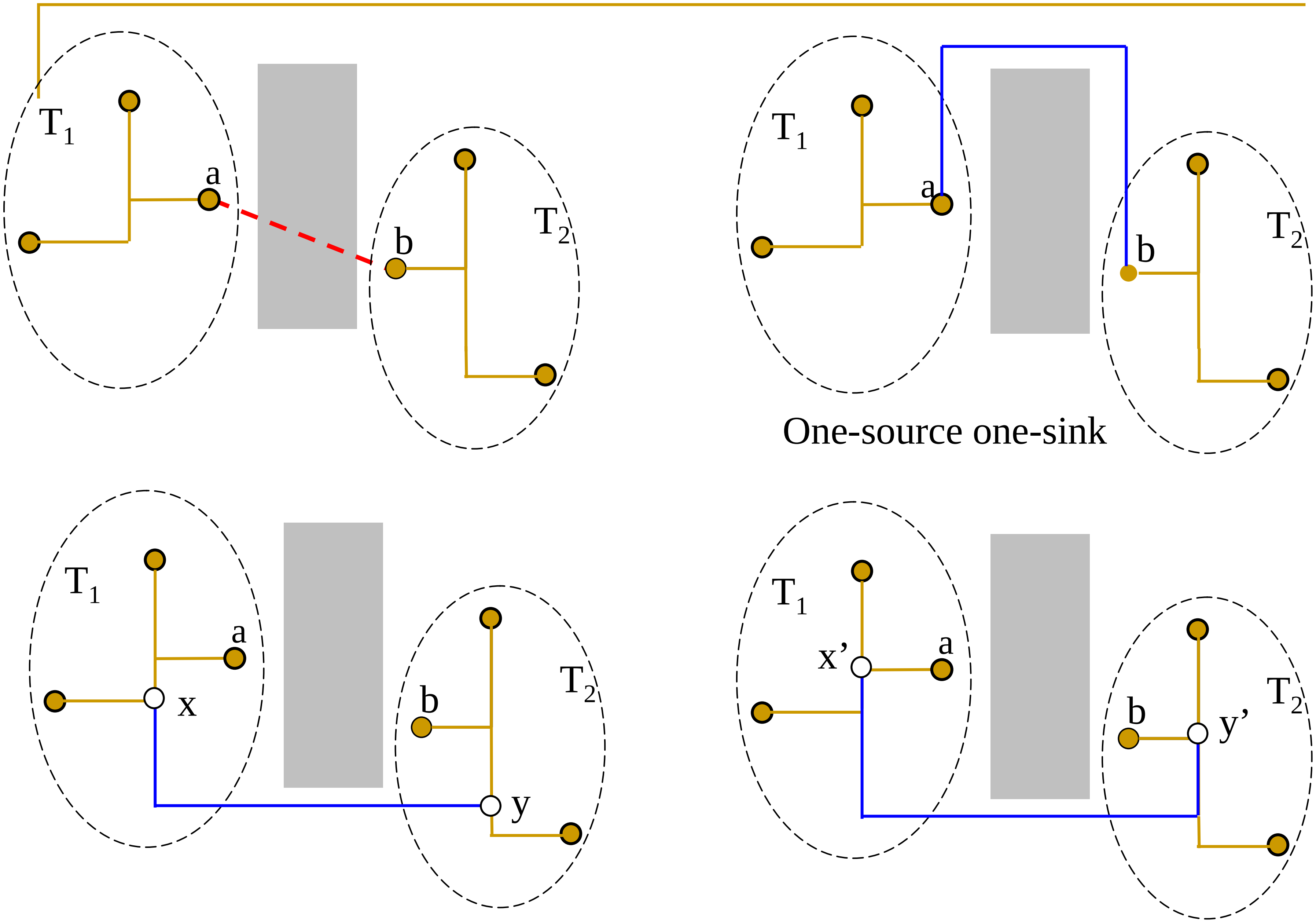
$$p_e = \left(\frac{d(e)+1}{s(e)}\right)^m$$

- Sharing edges does not need additional cost



Adaptive Multi-Source Multi-Sink Maze Routing

- General maze routing
  - One-source one-sink
- Multi-source multi-mink maze routing
  - Treat all grid points on one subtree as sources
  - Another subtree as sinks
- Adaptive multi-source multi-mink maze routing
  - Treat only pins or Steiner points as sources and sinks
  - More efficient



One-source one-sink

Multi-source multi-sink

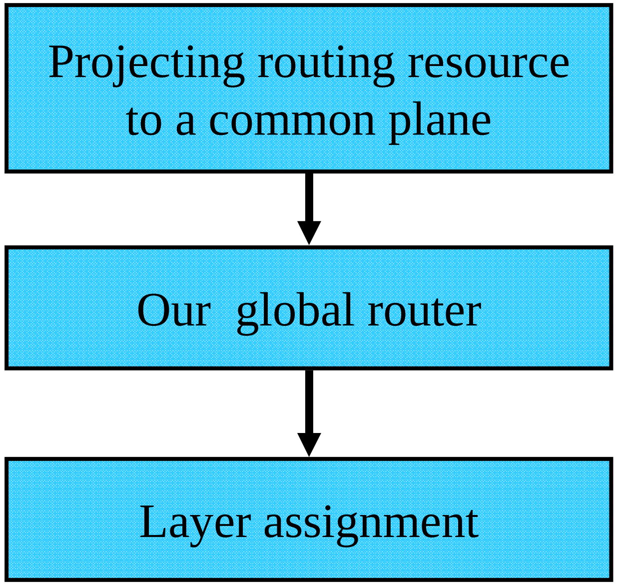
Adaptive multi-source multi-sink

Refinement

- Apply when iterative history based rip-ups and reroutes get stuck
  - $cost_e$=1+$h_e \cdot p_e$
  - h_e dominates edge cost when *e* tends to be congested
- Use another cost function
  - If passing *e* induces overflow → $cost_e$=1
  - Otherwise → $cost_e$=0
- Rip-up & reroute 2-pin nets in decreasing order of total overflow
  - Monotonic routing
  - Adaptive multi-source multi-sink maze routing if necessary

Extension for Multi-Layer Designs

- Multi-layer design
  - Each layer may have preferred routing direction
  - Need via to connect different layers



Outline

- Introduction
- Problem Formulation
- Methodology
- Experimental Results
- Conclusion

Experimental Setup

- Benchmarks

Benchmark	Grids	# nets
ibm01	64x64	11507
ibm02	80x64	18429
ibm03	80x64	21621
ibm04	96x64	26163
ibm06	128x64	33354
ibm07	192x64	44394
ibm08	192x64	47944
ibm09	256x64	50393
ibm10	256x64	64227

ISPD98

Benchmark	Grids	# nets
adaptec1	324x324	219794
adaptec2	424x424	260159
adaptec3	774x779	466295
adaptec4	774x779	515304
adaptec5	465x468	867441
newblue1	399x399	331663
newblue2	557x463	463213
newblue3	973x1256	551667

ISPD07

- Machine: Linux system with 2.2G CPU and 8G memory
- Comparison bases
  - ISPD98: BoxRouter, FastRoute 2.0
  - ISPD07: FGR, MaizeRouter, BoxRouter

Results on ISPD98 Benchmarks

Benchmark	BoxRouter			FastRoute 2.0			Our algorithm			WL reduction over BoxRouter	WL reduction Over FastRoute 2.0
	OF	WL	Runtime(s)	OF	WL	Runtime(s)	OF	WL	Runtime(s)		
Ibm01	**102**	**65588**	**8.3**	**31**	**68489**	**0.72**	**0**	**63321**	**4.17**	**-**	**-**
Ibm02	**33**	**178759**	**34.1**	**0**	**178868**	**0.93**	**0**	**170531**	**7.44**	**-**	**4.66%**
Ibm03	**0**	**151299**	**16.9**	**0**	**150393**	**0.60**	**0**	**146551**	**5.86**	**3.14%**	**2.55%**
Ibm04	**309**	**173289**	**23.9**	**64**	**175037**	**1.88**	**0**	**168262**	**13.61**	**-**	**-**
Ibm06	**0**	**282325**	**33.0**	**0**	**284935**	**1.36**	**0**	**278617**	**12.75**	**1.31%**	**2.22%**
Ibm07	**53**	**378876**	**50.9**	**0**	**375185**	**1.60**	**0**	**366288**	**15.89**	**-**	**2.37%**
Ibm08	**0**	**415025**	**93.2**	**0**	**411703**	**2.36**	**0**	**405169**	**13.17**	**2.37%**	**1.59%**
Ibm09	**0**	**418615**	**63.9**	**3**	**424949**	**1.92**	**0**	**415464**	**11.59**	**0.75%**	**2.23%**
ibm10	**0**	**593186**	**95.1**	**0**	**595622**	**2.79**	**0**	**580793**	**33.72**	**2.09%**	**2.49%**
Average										**1.93%**	**2.59%**

Results on ISPD07 Benchmarks

Benchmark		FGR			MaizeRouter			BoxRouter			Our algorithm		
		Total OF	Max OF	Total cost (e5)	Total OF	Max OF	Total cost (e5)	Total OF	Max OF	Total cost (e5)	Total OF	Max OF	Total cost (e5)
2-layer	adaptec1	0	0	55.8	0	0	62.26	0	0	58.84	0	0	57.11
	adaptec2	0	0	53.69	0	0	57.23	0	0	55.69	0	0	54.46
	adaptec3	0	0	133.34	0	0	137.75	0	0	140.87	0	0	137.16
	adaptec4	0	0	126.05	0	0	128.45	0	0	128.75	0	0	128.66
	adaptec5	0	0	155.82	2	2	176.69	0	0	164.32	0	0	160.3
	newblue1	1218	10	47.51	1348	16	50.93	400	2	51.13	352	4	47.78
	newblue2	0	0	77.67	0	0	79.64	0	0	79.78	0	0	79.22
	newblue3	36970	1090	108.18	32588	1236	114.63	38976	1088	111.64	31800	608	111
6-layer	adaptec1	60	2	90.92	0	0	99.61	0	0	104.05	0	0	90.56
	adaptec2	50	2	92.19	0	0	98.12	0	0	102.97	0	0	92.17
	adaptec3	0	0	203.44	0	0	214.08	0	0	235.87	0	0	205.04
	adaptec4	0	0	186.31	0	0	194.38	0	0	211.95	0	0	188.43
	adaptec5	2480	2	264.58	2	2	305.32	0	0	298.08	0	0	265.03
	newblue1	2668	4	92.89	1348	16	101.74	400	2	101.83	352	2	90.91
	newblue2	0	0	136.08	0	0	139.66	0	0	155.07	0	0	136.01
	newblue3	53648	636	168.42	32840	1058	184.4	38976	1088	195.5	31800	204	168.4

Cost Reduction rates and Runtimes

Benchmark (2-layer)	Cost reduction over FGR	Cost reduction over MaizeRouter	Cost reduction over BoxRouter	Runtime(s)
adaptec1	-2.35%	8.27%	2.94%	5579.98
adaptec2	-1.43%	4.84%	2.21%	977.5
adaptec3	-2.86%	0.43%	2.63%	3802.87
adaptec4	-2.07%	-0.16%	0.07%	522.29
adaptec5	-2.88%	-	2.45%	15990.29
newblue1	-	-	-	2251.45
newblue2	-2.00%	0.53%	0.70%	210.21
newblue3	-	-	-	21380.57
Average	-2.26%	2.78%	1.83%	

Benchmark (6-layer)	Cost reduction over FGR	Cost reduction over MaizeRouter	Cost reduction over BoxRouter	Runtime(s)
adaptec1	-	9.09%	12.96%	5595.4
adaptec2	-	6.06%	10.49%	991.34
adaptec3	-0.79%	4.22%	13.07%	3843.31
adaptec4	-1.14%	3.06%	11.10%	558.28
adaptec5	-	-	11.09%	16045.34
newblue1	-	-	-	2261.67
newblue2	0.05%	2.61%	12.29%	230.44
newblue3	-	-	-	21412.59
Average	-0.62%	5.01%	11.83%	

Outline

- Introduction
- Problem Formulation
- Methodology
- Experimental Results
- Conclusion

Conclusion

- NTHU-Route -- A new global routing algorithm
  - Based on Iterative rip-up & reroute
  - Adaptive multi-source multi-sink maze routing
  - Congested region identification method
  - Final refinement process for bottleneck
- Achieve good solution quality
  - Solve all cases on ISPD98 benchmarks
  - Solve 6 of 8 cases on ISPD07 benchmarks
  - Obtain fairly low routing cost

Thank You

Q & A