FastRoute 2.0: A High-quality and Efficient Global Router

Min Pan and **Chris Chu**

Iowa State University

Work partially supported by SRC under task ID 1206
and NSF under grant CCF-0540998

FastRoute 2.0: Overview

- A high-quality and fast global router based on:
  - Monotonic routing
  - Multi-source multi-sink maze routing
- On global routing benchmarks from Dr. Kastner (Labyrinth)
  - Compared to *FastRoute* [ICCAD06], *Labyrinth* [TCAD 02] and *Chi Dispersion* router [DAC 03]
    - Achieve 0 overflow for 6/9 circuits with total overflow being reduced by more than 10x
    - 73% slower than *FastRoute*, but still 78x and 37x faster than *Labyrinth* and *Chi Dispersion* router, respectively

Contributions

- Traditional Rip-up and reroute approaches:
    1. Construct Steiner minimal tree or minimum spanning tree for nets
    2. Break each routing tree into tree edges (2-pin nets)
    3. Apply pattern routing and maze routing to route all 2-pin nets one by one

- Our Approaches:
    1. Construct congestion-driven Steiner trees for nets
    2. Monotonic routing replaces pattern routing
    3. Multi-source multi-sink maze routing replaces single-source single-sink maze routing

Inaccurate Interconnect Information

- Placement becomes a critical step in VLSI design flow
- Inaccurate interconnect models are used in placement
  - Star model
  - Clique model
  - Half-perimeter of bounding rectangle
- Accurate information for interconnect cannot be obtained
  - Wirelength
  - Routing congestion
  - Timing
  - Buffer positions and sizes

Why Global Routing?

- Inconsistency between:
  - Interconnect models used in placement
  - Real implementation of interconnects in routing

Global Routing is desirable inside Placement !

But TOO Expensive !!

- Need high-quality and fast Global Router
  - Can be used for interconnect estimation in placement process as well as after-placement global routing
  - High quality so that it can be used as real global router
  - Fast enough to be integrated in placement process

Previous Work: FastRoute

- *FastRoute*: An extremely fast global router [ICCAD06]
- On global routing benchmarks from Dr. Kastner (Labyrinth)
  - Compared to *Labyrinth* [TCAD 02] and *Chi Dispersion* router [DAC 03]
    - 132x and 64x faster than *Labyrinth* and *Chi Dispersion* router, respectively
    - less total overflow than both
  - Even faster than highly-efficient congestion estimator *FaDGloR* [SLIP 05]

FastRoute Flow

Phase 1: Congestion map generation

1) RSMT construction – FLUTE
2) 50% probability L-routing

Phase 2: Congestion-driven Steiner tree construction

1) Congestion-driven Steiner tree topology generation
2) Edge shifting

Phase 3: Pattern routing and Maze routing

1) Z-shaped pattern routing
2) Edge-by-edge Maze routing

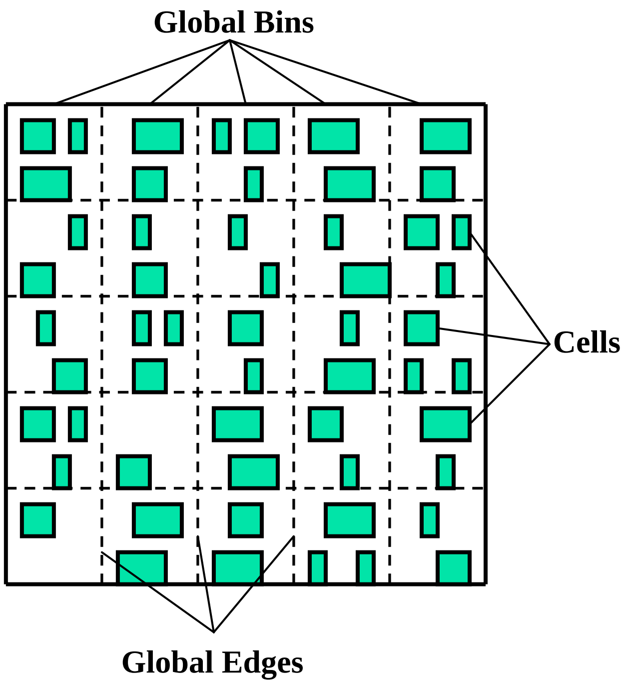
Improvement over *FastRoute*

- *FastRoute*
  - Extremely fast
  - Quality can be further improved
- New Router
  - Improve the routing solutions quality of *FastRoute*
  - High speed comparable to *FastRoute*
- Two major techniques:
  - Monotonic routing - substitutes pattern routing
  - Multi-source Multi-sink maze routing

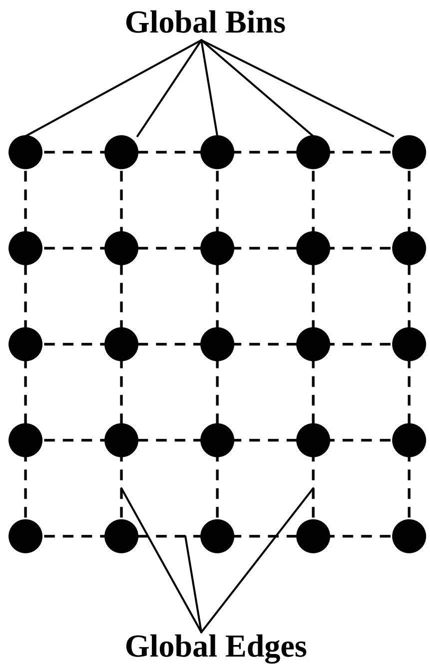
    - substitutes Single-source Single-sink maze routing

Grid Graph Model

- Global Routing problem:
  - Connect all the nets on the grid graph
  - Minimize the total overflow on all global edges



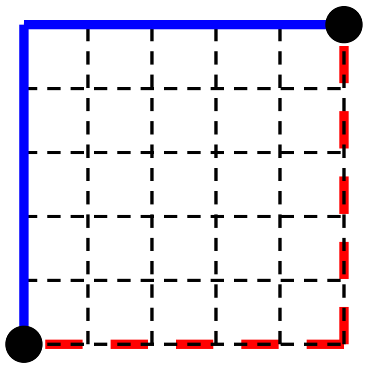
(a)



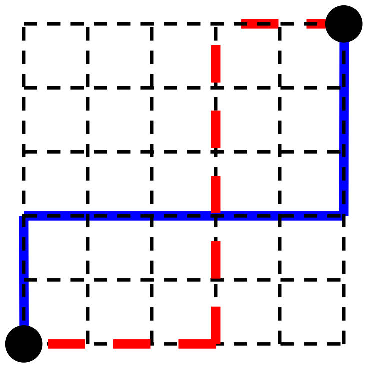
(b)

Pattern Routing

- Use predefined patterns to route 2-pin nets



L-Shaped



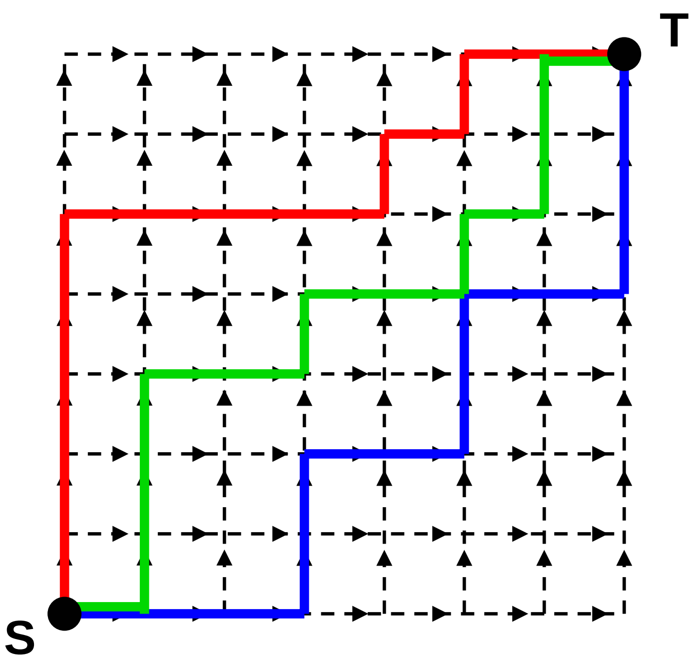
Z-Shaped

- Pros
  - Speed up the routing procedure
  - Constant # bends
- Cons
  - Very limited # paths being searched
  - Solution quality could be much worse than maze routing

Monotonic Routing (1)

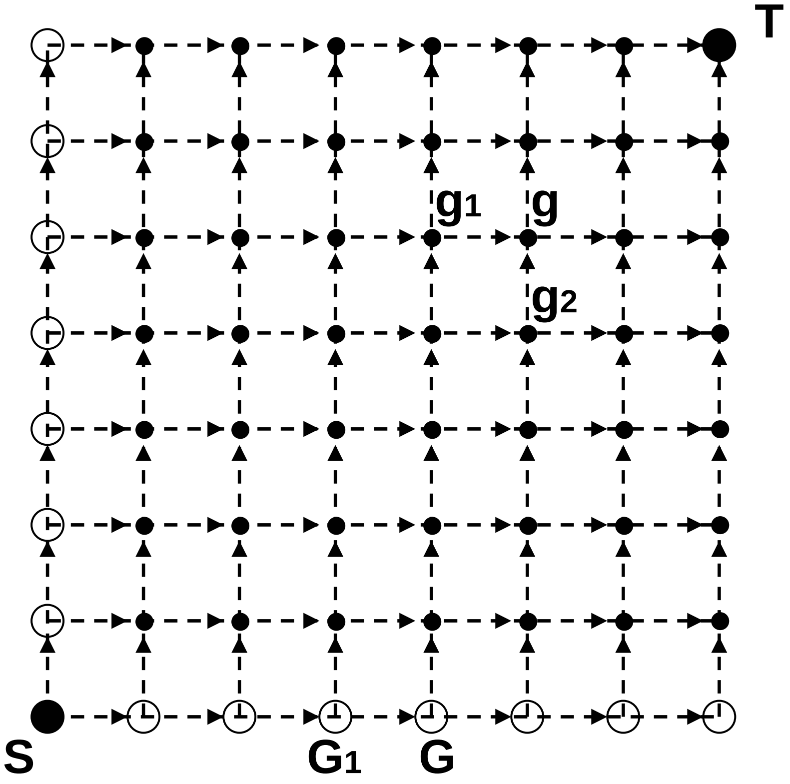
- Routing 2-pin nets (S to T)
- The routing path is monotonic from S to T
- # paths (m×n grids)
  - L-pattern: 2
  - Z-pattern: m+n-2
  - Monotonic

$$\binom{m+n-2}{m-1} = \frac{(m+n-2)!}{(m-1)!(n-1)!}$$

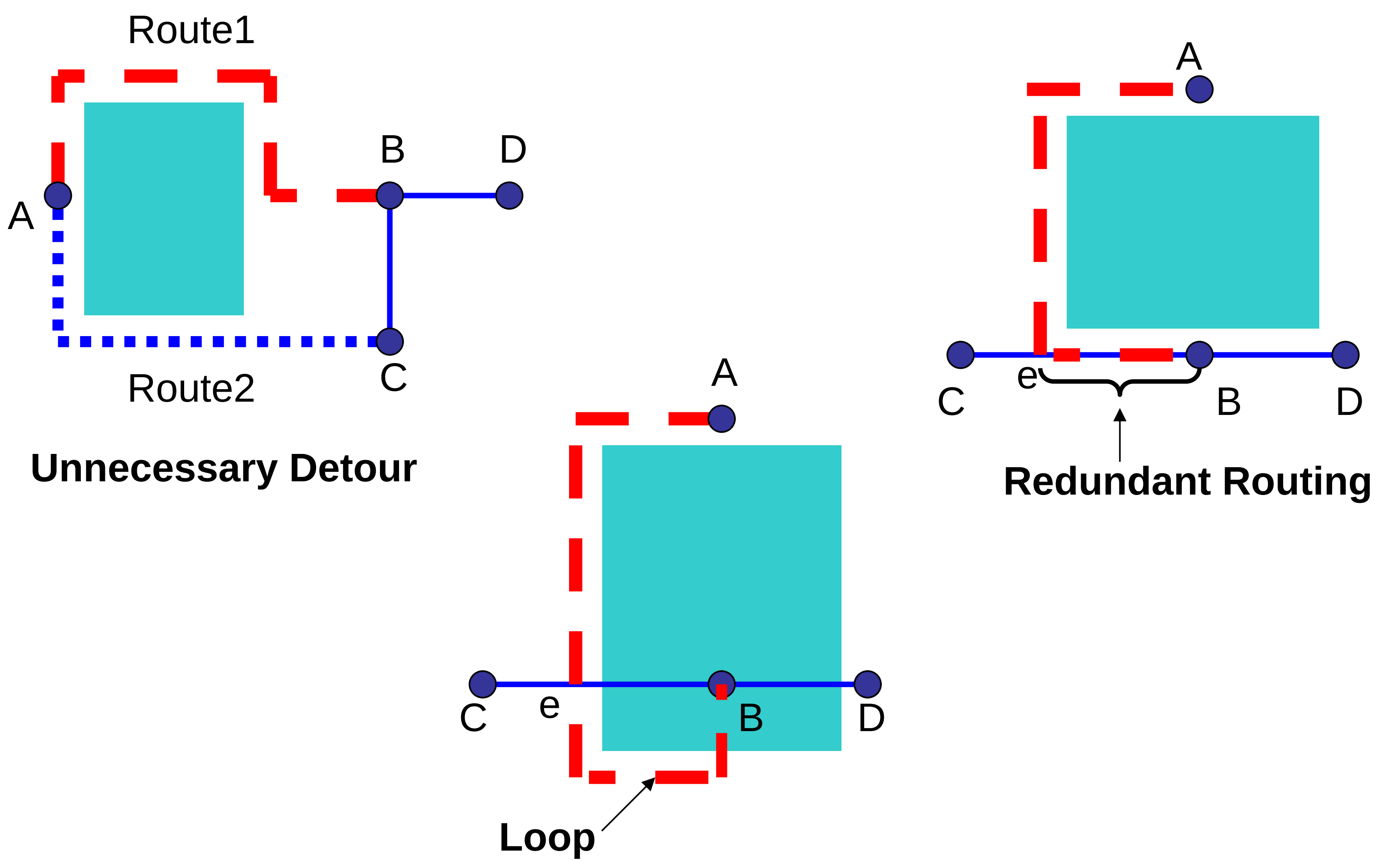


Monotonic Routing (2)

- Least cost monotonic routing path from S to T
  - For any grid points, the least cost monotonic routing path can be found easily
  - Dynamic programming to find the least cost monotonic path
  - Complexity: O(mn)
    same as Z-pattern routing

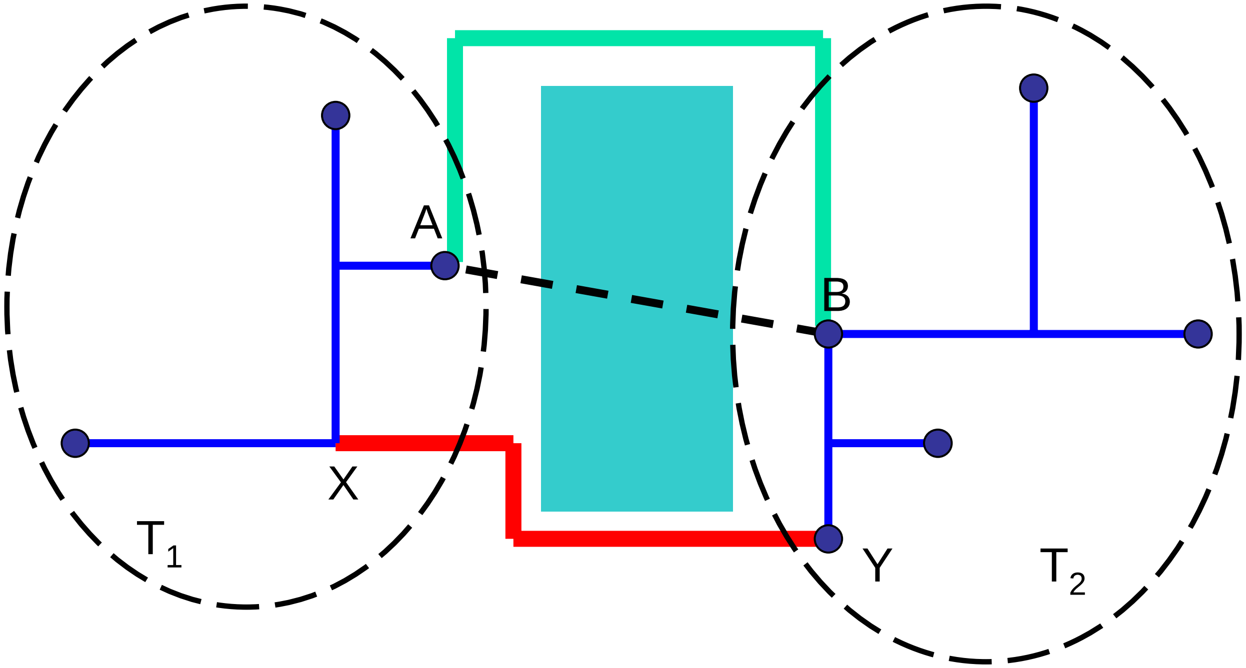


Problems with Edge-by-Edge Maze Routing



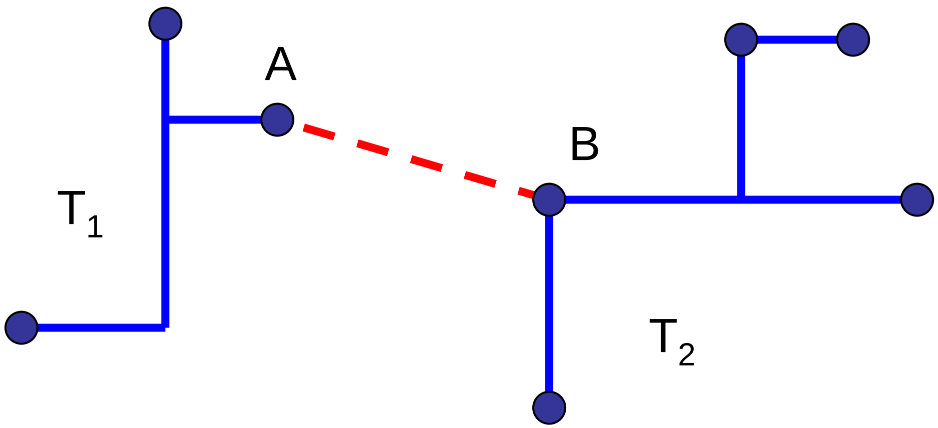
Multi-source Multi-sink Maze Routing

- Maze routing
  - Edge-by-Edge Single-source Single-sink maze routing
    - Route each tree edge one by one, from one endpoint to the other
  - Multi-source Multi-sink maze routing
    - Route between two subtrees



Multi-source Multi-sink Maze Routing Algorithm (1)

- Break the tree edge (A, B) to be routed and get two subtrees T_1 (contains A) and T_2 (contains B)
- First put all points on T_1 into priority queue Q
- Loop similar to Dijkstra's Algorithm to update shortest path from T_1 to the grid points
- Stop when any of the points on T_2 is extracted from Q
- Runtime complexity O($V lg V$) (V is the # grids)



FastRoute 2.0

Phase 1: Congestion map generation

1) RSMT construction – FLUTE
2) 50% probability L-routing

Phase 2: Congestion-driven Steiner tree construction

1) Congestion-driven Steiner tree topology generation
2) Edge shifting

Phase 3: Monotonic routing and Maze routing

1) Route all edges one-by-one by monotonic routing
2) Route the long edge passing congested region by Multi-source Multi-sink maze routing

Phase 3 of FastRoute 2.0

- Phase 3: monotonic routing and multi-source multi-sink maze routing

```
1.  for each net n with Steiner tree T
2.      for each tree edge e in T
3.          Rip-up e and reroute it by monotonic routing
4.  do
5.      for each net n with Steiner tree T
6.          for each tree edge e in T
7.              Rip-up e and reroute it by multi-source
                multi-sink maze routing
8.  Until no significant overflow reduction
```

Experimental Setup

- Benchmarks (From Dr. Kastner (Labyrinth))

Benchmark	Grids	# nets	# routed nets	Max Deg	Avg Deg
ibm01	64x64	11.5k	9.1k	37	3.8
ibm02	80x64	18.4k	14.3k	126	4.4
ibm03	80x64	21.6k	15.3k	49	3.6
ibm04	96x64	26.2k	19.7k	41	3.4
ibm06	128x64	33.4k	25.8k	34	3.8
ibm07	192x64	44.4k	34.4k	22	3.8
ibm08	192x64	47.9k	35.2k	65	4.3
ibm09	256x64	50.4k	39.6k	38	3.8
ibm10	256x64	64.2k	49.5k	32	4.2

- Machine: 3.0GHz Pentium 4 CPU with 2GB RAM
- Experiments
  - Compare with *FastRoute* [ICCAD06], *Labyrinth* [TCAD02] and *Chi Dispersion* router [DAC03]
  - Effect of Monotonic routing technique
  - Investigate Multi-source Multi-sink maze routing

Comparison with Other Global Routers

- OF – total overflow on all global edges
- WL – total routing wirelength (unit - the length of a global edge)
- Time – runtime in seconds.

	FastRoute 2.0			FastRoute			Labyrinth Predictable router			Chi Dispersion router		
	OF	WL	Time	OF	WL	Time	OF	WL	Time	OF	WL	Time
ibm01	31	68489	0.72	250	67128	0.21	242	76228	16.99	189	66005	8.63
ibm02	0	178868	0.93	39	179995	0.56	214	202235	26.53	64	178892	26.27
ibm03	0	150393	0.60	1	151023	0.43	117	191500	37.92	10	152392	24.71
ibm04	64	175037	1.88	567	172593	0.50	786	198181	80.95	465	173241	32.94
ibm06	0	284935	1.36	33	285882	0.91	130	339379	72.06	35	289276	53.33
ibm07	0	375185	1.60	18	376835	1.05	407	450855	168.41	309	378994	79.61
ibm08	0	411703	2.36	58	412915	1.16	352	466556	154.82	74	415285	72.94
ibm09	3	424949	1.92	28	426471	1.39	310	481841	229.59	52	427556	86.67
ibm10	0	595622	2.79	18	599433	1.98	288	680113	296.70	73	599937	139.61
Total	98	2665181	14.16	1012	2672275	8.19	2846	3086888	1083.97	1271	2681578	524.71
Norm	1	1	1	10.327	1.003	0.578	29.041	1.158	78	12.969	1.006	37

Effect of Monotonic Routing Technique

- Flows for phase 3 routing
  - Flow1: Only Z-shaped pattern routing
  - Flow2: Only Monotonic routing
  - Flow3: Z-shaped pattern routing + Maze routing (m-s m-s)
  - Flow4: Monotonic routing + Maze routing (m-s m-s)

	Z	Monotonic	Z + Maze	Monotonic + Maze
ibm01	1435	1280	40	31
ibm02	2711	2569	0	0
ibm03	260	145	0	0
ibm04	1950	1794	112	64
ibm06	1682	1444	0	0
ibm07	1020	853	0	0
ibm08	963	735	1	0
ibm09	1065	626	21	3
ibm10	1834	1532	2	0
Total	12920	10978	176	98

- Runtime of Monotonic routing is 2.3x slower than Z-routing

Maze Routing Statistics

	Total # of tree edges	Tree Edges Being Maze Routed (%)	Tree Edges w/ Endpoints Changed (%)
ibm01	28116	3.24%	1.12%
ibm02	55361	4.00%	1.25%
ibm03	45582	1.79%	0.45%
ibm04	53308	4.04%	0.88%
ibm06	82283	2.43%	0.62%
ibm07	109175	1.89%	0.44%
ibm08	133222	1.13%	0.30%
ibm09	128185	1.25%	0.45%
ibm10	181432	1.25%	0.47%
Avg		2.34%	0.66%

- Significant portion of the edges (1 out of 3.5) been maze routed have their endpoints changed.

Runtime Comparison with Capo & Dragon

- Runtime comparison with Capo9.1 and Dragon3.01
  - All runtimes are in seconds

	FastRoute 2.0	Capo9.1	Dragon3.01
ibm01	0.72	126	778
ibm02	0.93	280	663
ibm03	0.60	338	633
ibm04	1.88	456	1234
ibm06	1.36	666	1392
ibm07	1.60	1145	1904
ibm08	2.36	1277	4163
ibm09	1.92	1329	3953
ibm10	2.79	2035	3537
Total	14.16	7652	18257
Norm	1	540	1290

  - Incorporating FastRoute into placers will not significantly increase placement runtime

Conclusion and Future Work

- *FastRoute* 2.0
  - High-quality
    - More than 10x less total overflow than *FastRoute, Labyrinth* and *Chi Dispersion* router
  - Speed
    - 73% slower than *FastRoute,* but still 78x faster than Labyrinth and 37x faster than Chi Dispersion router
    - 2~3 orders of magnitude faster than state-of-the-art academic placers
- Future work
  - Incorporate *FastRoute* into multilevel framework
  - Integrate global routing into placement process

Thank You !

Questions?