DpRouter: A Fast and Accurate Dynamic-Pattern-Based Global Routing Algorithm

Zhen Cao [1],Tong Jing [1, 2], Jinjun Xiong [2], Yu Hu [2], Lei He [2], Xianlong Hong [1]

[1] Tsinghua University
[2] University of California, Los Angeles

Outline

- Introduction
- Preliminaries
- Dynamic Pattern Routing
- Global Routing Algorithm
- Experimental Results
- Conclusions

Introduction

- The success of future integrated circuit designs requires the consideration of physical impact. Congestion-aware global routing is crucial to achieve this goal.
- Recent progress on RSMT problem
  - FLUTE [Chu, ICCAD04]
  - Creating and exploiting flexibility [Kastner, TCAD03]
- Recent progress on routing algorithm
  - Labyrinth 1.1 [Kastner, TCAD02]
  - Fengshui 5.1 [Hadsell, DAC03]
  - BoxRouter [Cho, DAC06]
  - FastRoute 1.0 [Pan, ICCAD06]

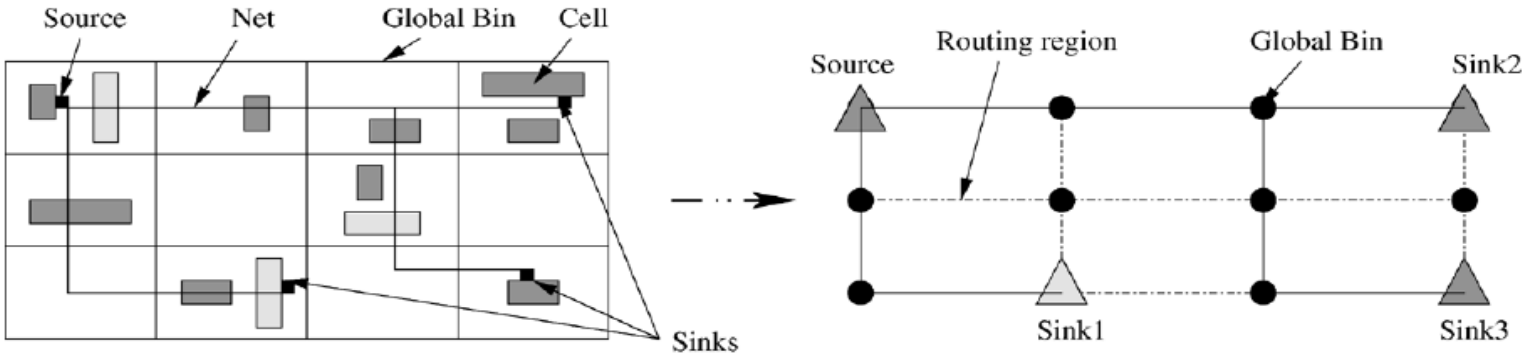
Major Contributions of This Paper

- **An efficient dynamical pattern routing (Dpr) technique to achieve the optimal routing solution for two-pin nets.**
- **A segment-move technique to extend the searching space for routability driven RST problem.**
- **A congestion-aware global routing algorithm that combines the above two techniques to achieve high speed and high optimization capability.**
- **Compared with labyrinth 1.1 and Fengshui 5.1, we simultaneously reduce total overflow by 89.76% and 51.71%, total wire length by 14.49% and 1%, while achieving 64x and 38x speed up of runtime, respectively.**
- **Compared with BoxRouter and FastRoute 1.0, we obtain better WL and significant speedup (20x) over BoxRouter, with similar congestion; we obtain better WL and congestion than FastRoute 1.0, with similar runtime.**

Outline

- Introduction
- Preliminaries
- Dynamic Pattern Routing
- Global Routing Algorithm
- Experimental Results
- Conclusions

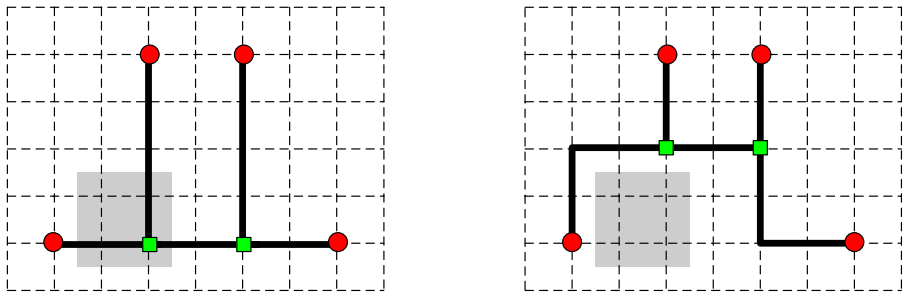
Problem Formulation



- Connect all nets in the routing graph, with the goal of
  - Minimizing (rectilinear) wire length (WL)
    - Performance
  - Minimizing congestion
    - Routability, signal integrity, areas
  - Reducing runtime
    - Integrated physical synthesis flow

Basic Definitions

- *Segment:* a segment is a horizontal or vertical edge line.
- *Topology:* describe the connection relationship between pins and Steiner points.
  - For a given topology, there may exist many embedded trees.
- *Edge of a tree:* the routing between two vertexes.
- *Flexibility*: a flexible edge is an edge by which two vertexes are connected by more than one segment.
  - Flexible edges allow more than one minimum length routing solutions, which is more suitable for global routing than non-flexible edges.

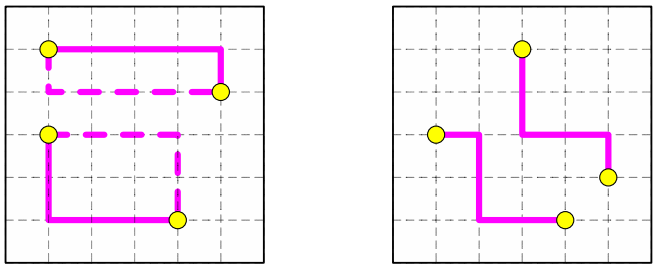


Outline

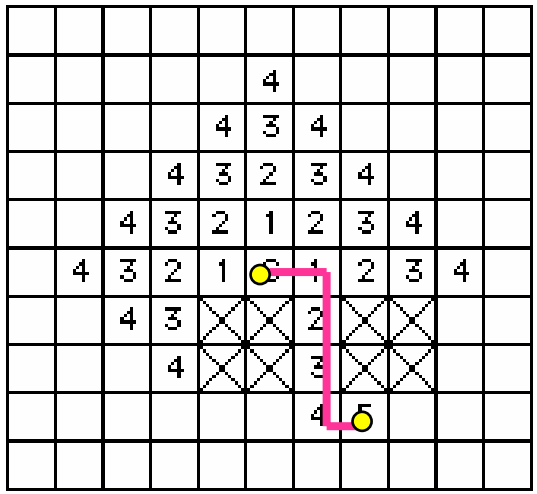
- Introduction
- Preliminaries
- Dynamic Pattern Routing
- Global Routing Algorithm
- Experimental Results
- Conclusions

Two-pin Net Routing

- In global routing, a multiple pin net is usually decomposed into a few two-pin nets to be routed
- Pattern routing
  - L-pattern (1-bend), Z-patten (2-bend) [Kastner, TCAD02]
  - Fast, but with limited routing flexibility

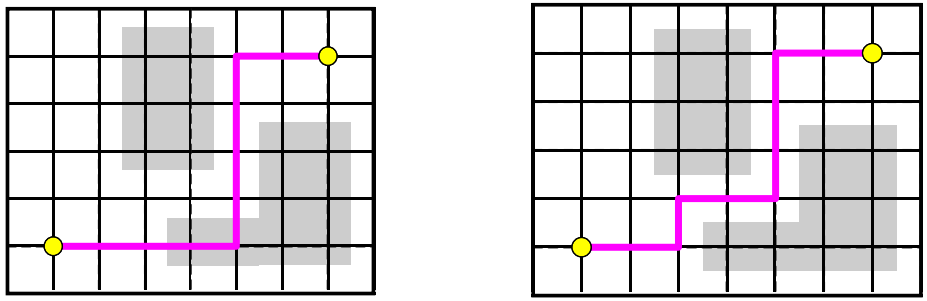


- Maze routing
  - High quality, but too slow

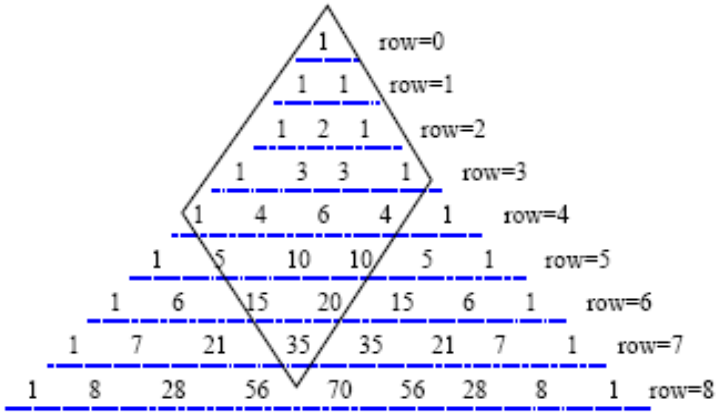


Multi-bend Pattern Routing

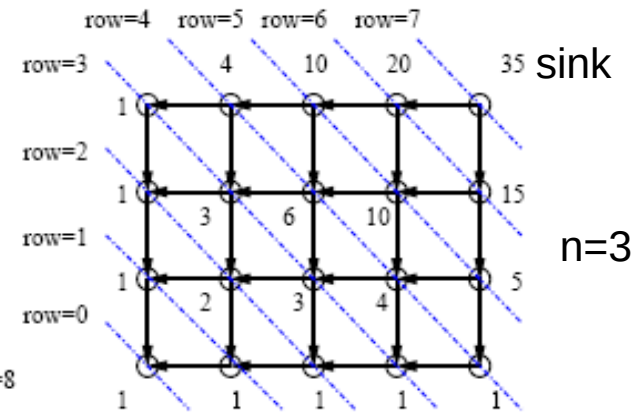
- Multi-bend pattern routing



- There are *C(m+n,m)* different pattern routings to connect two pins with minimum WL

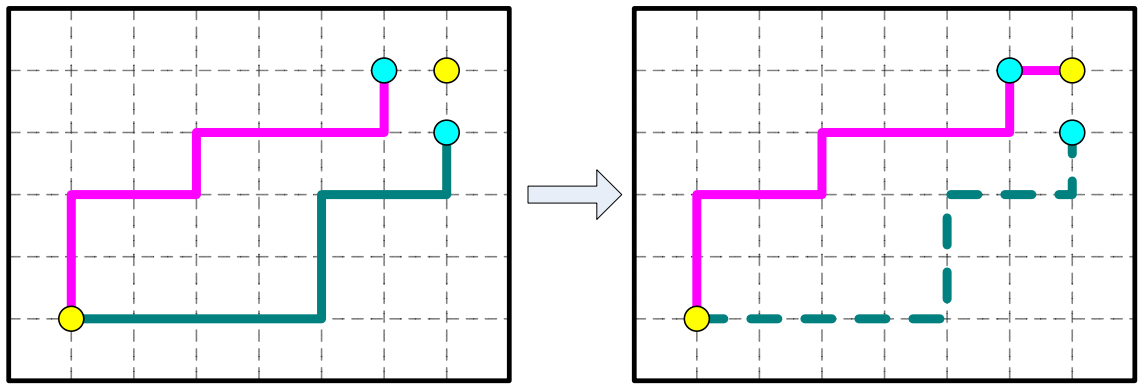


Pascal Triangle



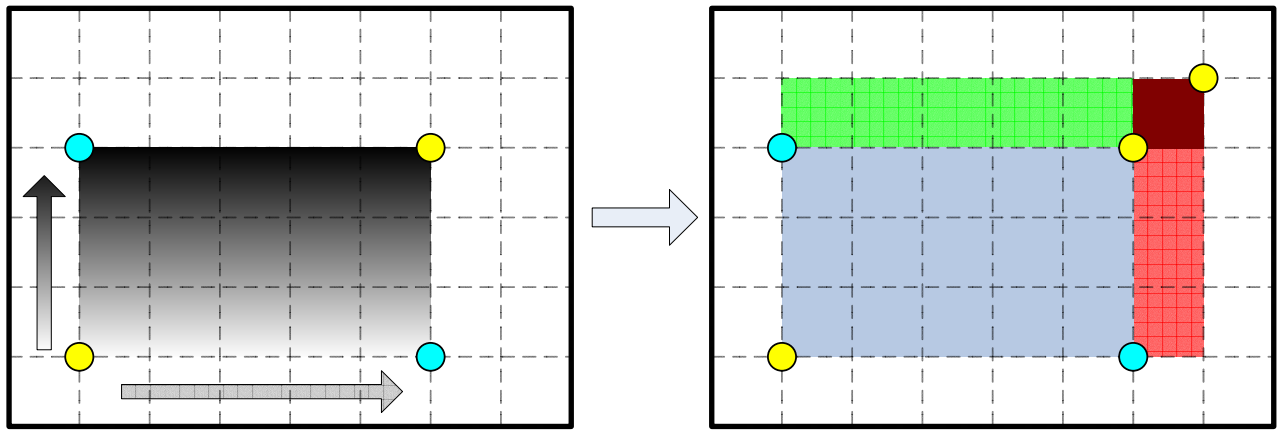
Dpr — Dynamic Pattern Routing

- **We use dynamic programming to solve multi-bend pattern routing**
  - Optimal solution must be optimal in its sub-problems
  - we define proper cost function
  - The optimal solution can be constructed efficiently
- **The idea**
  - OPT(A,B) = min (OPT(A,C)+f(CB), OPT(A,D)+f(DB))
- **Same time complexity as Z-pattern, *O*(*mn*)**
  - With same time complexity, Dpr explores much more routing space than Z-pattern routing, *C(m+n,m)* vs *m+n+2*



Dpr-M: Dpr with Movable Pins

- By utilizing the memorization property of dynamic-programming, we can handle cases with movable pins efficiently
- An example
  - First find optimal solutions from A to all nodes within gray box.
  - When B moves to E, only need to incrementally compute optimal solutions from A to all nodes in green / pink / brown areas.



Outline

- Introduction
- Preliminaries
- Dynamic Pattern Routing
- Global Routing Algorithm
- Experimental Results
- Conclusions

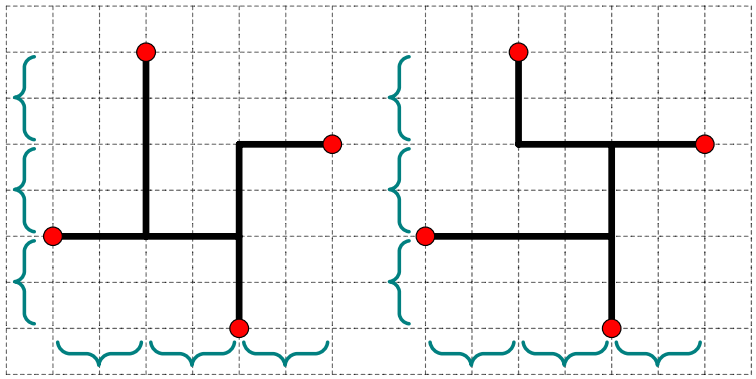
Routing Flow

Algorithm 2 DpRouter
Input: Routing graph G and nets set N
Output: Routing solutions for every net in N

```
1:  congestion estimation;
2:  for every net list l in N do
3:      get the POWVs set P of l;
4:      for every POWV v in P do
5:          use FLUTE to get the original topology t for vector v of
            net list l;
6:          find independent movable segment sets M of t;
7:          use Dpr and Segment-Move techniques to move segments
            in M to find the best routing solution s for v, put s in set S;
8:      end for;
9:      select the best solution s' in S, route it;
10:  end for;
11: rip-up and reroute every net in congested area, using method the
    same of line 3-8, for several iterations;
12: rip-up and reroute every net in congested area, tree topology is se-
    lected using method of line 3-8, but maze routing is utilized in rout-
    ing;
13: return;
```

Net Decomposition

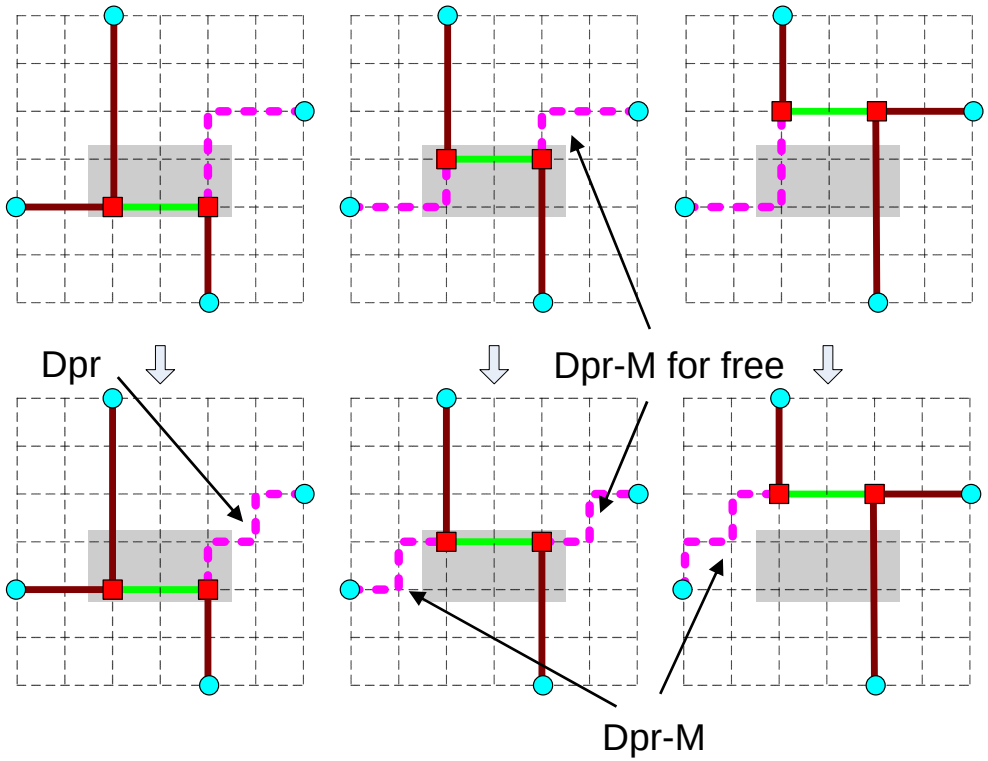
- In our routing flow, we design to obtain RSMT topology first, then decompose it into 2-pin nets.
- In our implementation, we use FLUTE [Chu, ICCAD04].
  - A routing can be encoded by Potentially Optimal WL Vector (POWV), from which WL can be computed efficiently.
  - For small nets (<10), optimal topology is stored in LUT.
  - For big nets, use net breaking technique to recursively break it into small ones.



(a)POWV (1, 1, 1, 1, 2, 1) (b) POWV (1, 2, 1, 1, 1, 1)

Segment-Move Technique

- In FLUTE, only one RSMT topology is stored for each POWV
  - Limit solution space to be explored
- We design **Segment-move technique** to explore more RSMT topologies for routability
  - Movable segment: edge between two adjacent Steiner points
- Combined Dpr and Dpr-M techniques to find routing solutions efficiently



Pins

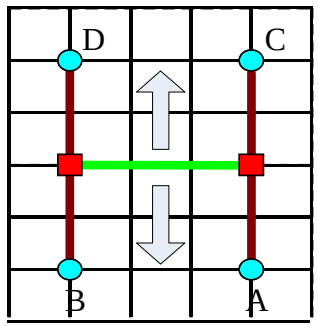
Steiner points

Flexible edges

Movable segments

Combination of Dpr and Segment-Move (1)

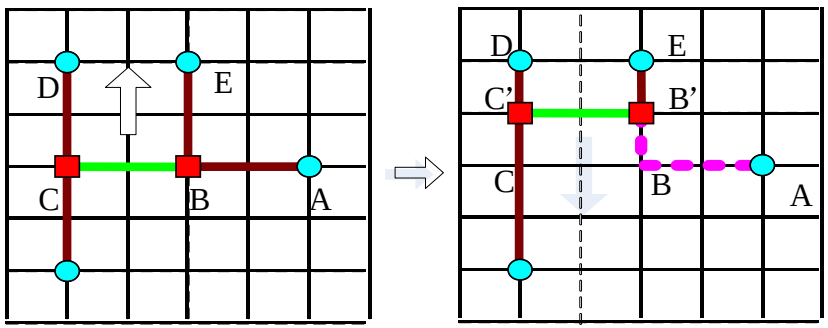
- Three types of movable segments
  - Type A: movement will not change the flexibility of adjacent edges
  - Solution: calculate cost on each movable position



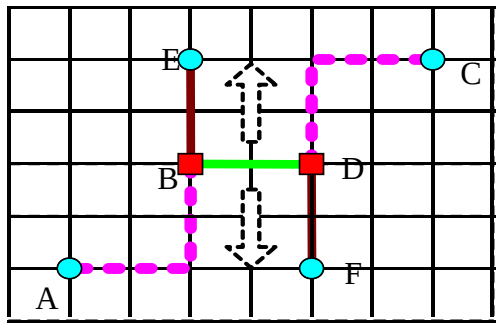
Flexibility not changed

Combination of Dpr and Segment-Move (2)

- Three types of movable segments
  - Type B: movement will change the flexibility of adjacent edges
  - Solution: combined with Dpr-M to calculate cost on each position



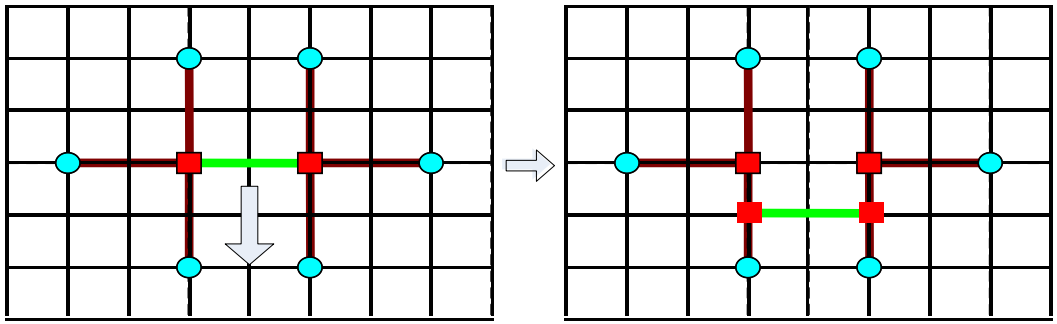
Increase flexibility of AB



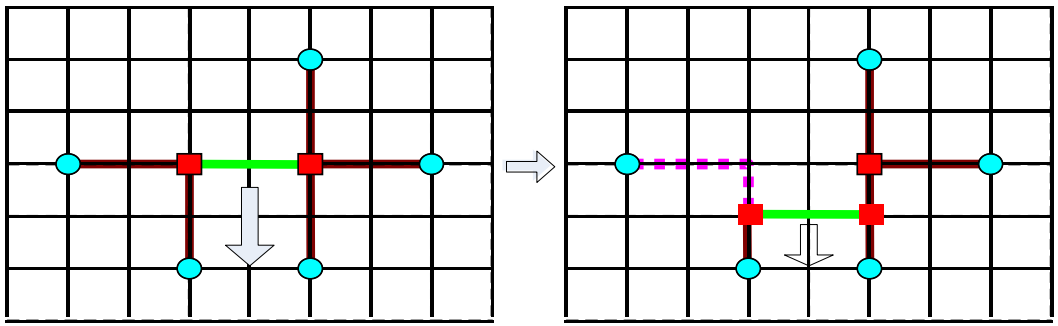
Increase/decrease flexibility of AB/CD

Combination of Dpr and Segment-Move (3)

- Three types of movable segments
  - Type C: movement will change the routing topology
  - Solution: change topology, reclassify into Type A or B



Afterwards, it becomes type A



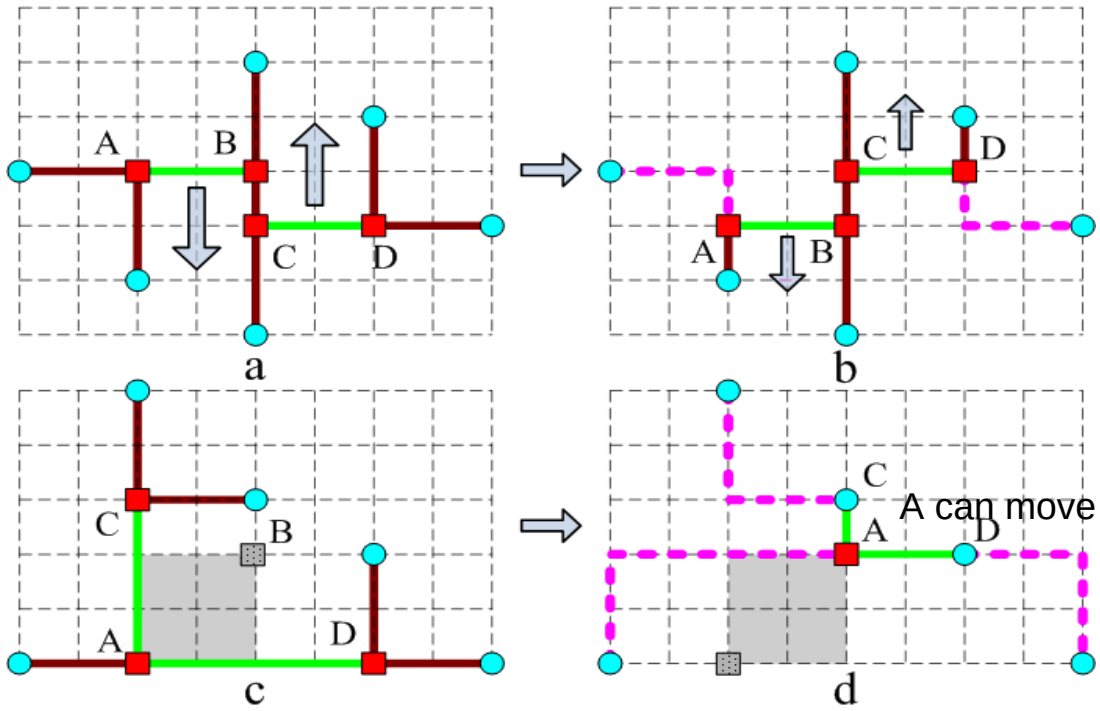
Afterwards, it becomes type B

Combination of Dpr and Segment-Move (4)

- **Movable segments may be overlapped**
  - Independent case: segments can be moved independently, but the tree topology need to change
  - Dependent case: shared vertex is moved in movable area

Independent

Dependent



A is the common points of two movable segments

Cost Function and Net Ordering

- Four components at each routing edge
  - $f = a*ov + b*mid + c*nets + d*esti$
  - Overflow: $ov = max(0, nets – capacity)$
  - Utilization rate: $mid = max(0, nets – k* capacity), 0<k<1$
  - Routed net number: *nets*
  - Estimation of routed net number: *esti*
- Net Ordering, sort the net based on bounding box
  - Initial routing: small→large
  - Rip-up and rerouting: large→small

Outline

- Introduction
- Preliminaries
- Dynamic Pattern Routing
- Global Routing Algorithm
- Experimental Results
- Conclusions

Experimental Results

- DpRouter implemented in C++
- Tested on ISPD98 benchmarks
- Comparison bases (all academic routers)
  - Labyrinth 1.1 [Kastner, TCAD02], binary
  - Fengshui 5.1 [Hadsell, DAC03], binary
  - BoxRouter [Cho, DAC06], published data
  - FastRoute 1.0 [Pan, ICCAD06], published data

  - Both BoxRouter and FastRoute compared their results with those of Labyrinth and Fengshui, so we can use their publication data in our comparison here

Benchmark Characteristics

Benchmark	Grids	Capacity (V, H)	# Cells	# Cells / bin
ibm01	64x64	12, 14	12k	2.9
ibm02	80x64	22, 34	19k	3.7
ibm03	80x64	20, 30	22k	4.3
ibm04	96x64	20, 23	26k	4.3
Ibm05	128x64	42, 63	28k	3.4
ibm06	128x64	20, 33	32k	3.9
ibm07	192x64	21 36	44k	3.6
ibm08	192x64	21, 32	50k	4.1
ibm09	256x64	14, 28	51k	3.1
ibm10	256x64	27, 40	67k	4.1

- **http://www.ece.ucsb.edu/~kastner/labyrinth/benchmarks/grids.txt**

Dpr vs L/Z-Pattern Routing

- Total overflow and runtime (1.6GHz PC running Linux)

circuit	L-shape PR		Z-shape PR		Dpr	
	tof	**cpu(*s*)**	***tof***	**cpu(*s*)**	***tof***	**cpu(*s*)**
ibm01	2118	0.13	2053	0.18	1636	0.15
ibm02	4081	0.37	4030	0.45	3585	0.37
ibm03	1016	0.27	983	0.42	480	0.27
Ibm04	3163	0.32	3151	0.46	2356	0.31
Ibm05	51	0.44	94	0.73	0	0.37
ibm06	3424	0.55	3083	0.82	2210	0.57
ibm07	2789	0.60	2644	0.84	2073	0.57
ibm08	2690	0.98	2580	1.45	1229	0.94
ibm09	4065	0.86	3528	1.62	1656	1.09
ibm10	3576	1.00	3306	1.84	2343	1.29
Avg	**1.66**	**0.97**	**1.56**	**1.47**	**1.00**	**1.00**

WL Comparison

	Labyrinth	Fengshui	FastRoute	BoxRouter	DpRouter
ibm01	76517	66006	67128	65588	63857
ibm02	204734	178892	179995	178759	178261
ibm03	185116	152392	151023	151299	150663
ibm04	196920	173241	172593	173289	172608
ibm05	420583	412197	-	409747	413496
ibm06	346137	289276	285882	282325	286025
ibm07	449213	378994	376835	378876	379133
ibm08	469666	415285	412915	415025	412308
ibm09	481176	427556	426471	418615	419199
ibm10	679606	599937	599433	593186	598460
Avg	**1.155**	**1.009**	**1.009**	**1.001**	**1**

- **DpRouter is the best in terms of WL, close to BoxRouter**

Overflow Comparison

Overflow	Labyrinth	Fengshui	FastRoute	BoxRouter	DpRouter
ibm01	398	189	250	102	125
ibm02	492	64	39	33	3
ibm03	209	10	1	0	0
ibm04	882	465	567	309	165
ibm06	834	35	33	0	14
ibm07	697	309	18	53	99
ibm08	665	74	58	0	56
ibm09	505	52	28	0	47
ibm10	588	51	18	0	46
Total	**5270**	**1249**	**1012**	**497**	**555**
Avg	**9.50**	**2.25**	**1.82**	**0.90**	**1**

- **DpRouter beats Labyrinth, Fengshui and FastRoute significantly**
- **DpRouter is similar with BoxRouter.**

Runtime Comparison

Runtime	Labyrinth(s)	Fengshui(s)	FastRoute	BoxRouter	DpRouter(s)
ibm01	35.1	25	0.61	13.74	0.94
ibm02	58.9	81.8	1.74	58.22	2.34
ibm03	63.7	61.8	1.07	29.66	1.44
ibm04	145.5	94.3	1.43	41.65	3.58
ibm05	108.1	191.4	-	-	1.49
ibm06	171.6	131.8	2.25	54.29	4.46
ibm07	408.4	218.8	2.89	91.13	5.44
ibm08	417.5	199	3.17	163.01	6.18
ibm09	673.1	234.4	3.76	119.71	4.74
ibm10	789.6	467.9	5.47	172.35	7.66
Avg	**64.6**	**47.2**	**0.62**	**19.42**	**1**

- **DpRouter beats Labyrinth, Fengshui and BoxRouter significantly**
- **DpRouter is at the same level as FastRoute**

Conclusions

- **DpRouter is a new global router based on**
  - Dpr, dynamic pattern routing
    - Dynamic-programming based technique for multi-bend pattern routing
    - Dpr-M enables incremental updating solution with movable pins
  - Segment move technique
    - Explore more tree topologies and solution space
    - Combined with Dpr/Dpr-M techniques to make routing generation more efficient

- **Compared with four state-of-the-art academic routers, DpRouter achieves**
  - Better WL and congestion with significant speedup (50~60x) over Labyrinth and Fengshui
  - Better WL and significant speedup (20x) over BoxRouter, with similar congestion
  - Better WL and congestion over FastRoute 1.0, with similar runtime.

DpRouter: A Fast and Accurate Dynamic-Pattern-Based Global Routing Algorithm

Thank you

Q & A