DraXRouter: Global Routing in X-Architecture with Dynamic Resource Assignment

Zhen Cao[1], Tong Jing[1], Yu Hu[2], Yiyu Shi[2], Xianlong Hong[1], Xiaodong Hu[3], Guiying Yan[3]

[1] Computer Science & Technology Department
Tsinghua University
Beijing 100084, China
Phone: +86-10-62785564
Fax: +86-10-62781489
e-mail: caoz@mails.tsinghua.edu.cn

[2] Electrical Engineering Department
UCLA
Los Angeles, CA 90095, USA
Phone: (310) 267-5407
Fax: (310) 267-5407
e-mail: {hu, yshi}@ee.ucla.edu

[3] Institute of Applied Mathematics
Chinese Academy of Sciences
Beijing 100080, China
Phone: +86-10-62639192
Fax: +86-10-62574529
e-mail: {xdhu, yangy}@amss.ac.cn

Speaker: Zhen Cao

Outline

- Introduction
- Preliminaries
- Dynamic Resource Assignment
- DraXRouter Routing Algorithm
- Experimental Results
- Conclusions

Introduction (1)

- X-Architecture
  - Interconnect delay has become a significant factor affecting IC performance.
  - The optimization capability of algorithms under Manhattan Architecture is limited.
  - X-Architecture reduces wire length by 20%, improves chip performance by 10% and reduces power by 20%.
  - X-Architecture is not widespread due to the lack of X-Architecture-based placement and routing algorithms.

Introduction (2)

- Recent work
  - Liquid Routing Technology [T. Mitsuhashi, 2001].
  - Multilevel Full-Chip Routing for the X-Based Architecture [T.Y. Ho, Y.W. Chang *et al*, DAC, 2005].
  - COCO Algorithm [Y. Hu, T. Jing *et al*, Samos, 2005].
- Our work
  - The global routing problem in X-Architecture.
  - Accurate and efficient routing resources assignment method.
  - Routing algorithms with good routing performance.

Contributions of This Paper

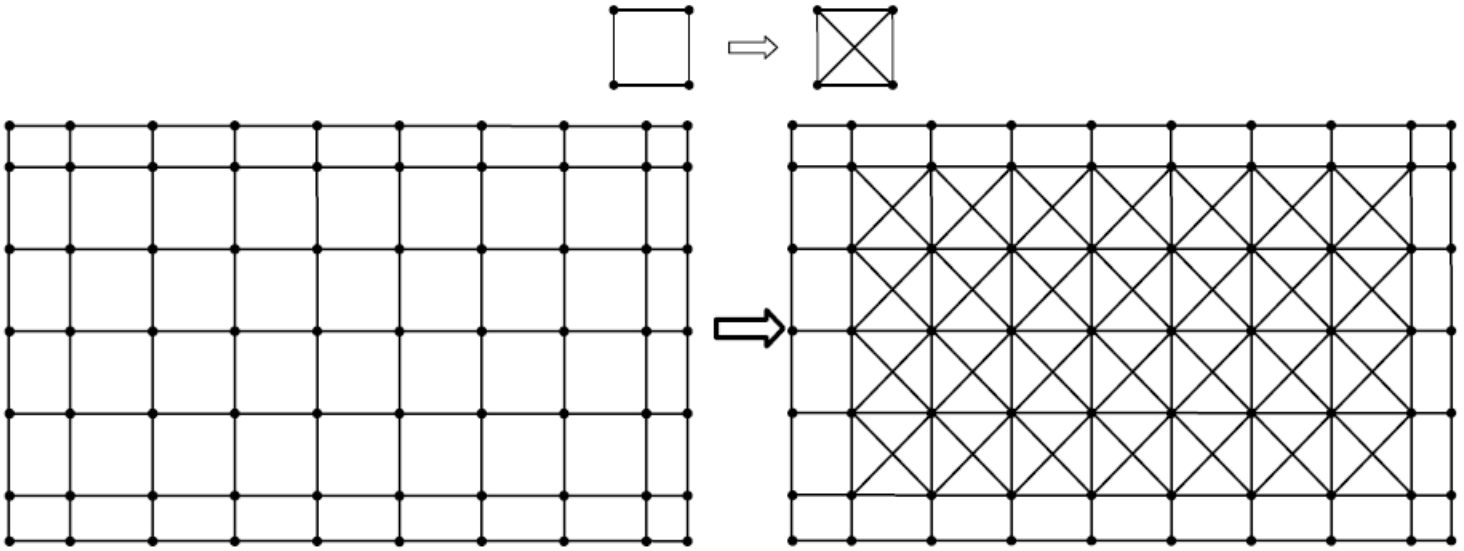
- We present a dynamic resource assignment (Dra) method to guide liquid routing in X-Architecture, which utilizes the routing resources more reasonably under liquid routing model.
- We design a global router, namely DraXRouter, in which we adopt a dynamic-tabulist-based (DTB) tree construction algorithm and the stochastic optimization strategy to gain good routing performance.
- Compared with x-labyrinth [the X-implementation of UCSB Labyrinth], DraXRouter improves utilization of routing resource with a shorter wire length. At the same time, the runtime of DraXRouter is 33% shorter.
- Compared with COCO algorithm [Y. Hu, T. Jing *et al*, 2005], DraXRouter gains a better routing result with an acceptable runtime.

Outline

- Introduction
- Preliminaries
- Dynamic Resource Assignment
- DraXRouter Routing Algorithm
- Experimental Results
- Conclusions

Preliminaries – GRG Generation

- Map all physical pins into each tile.
- Utilize placement result in Manhattan Arch.



Preliminaries – Problem Formulation

- Overflow

$$overflow_e = \begin{cases} d_e - c_e, & if\ d_e > c_e \\ 0, & otherwise \end{cases}$$

- Total overflow

$$tof = \sum_{e \in E} overflow_e$$

- Find routing solution for each edge *e*, to minimize *twl* while *tof = 0.*

Preliminaries – Liquid Routing

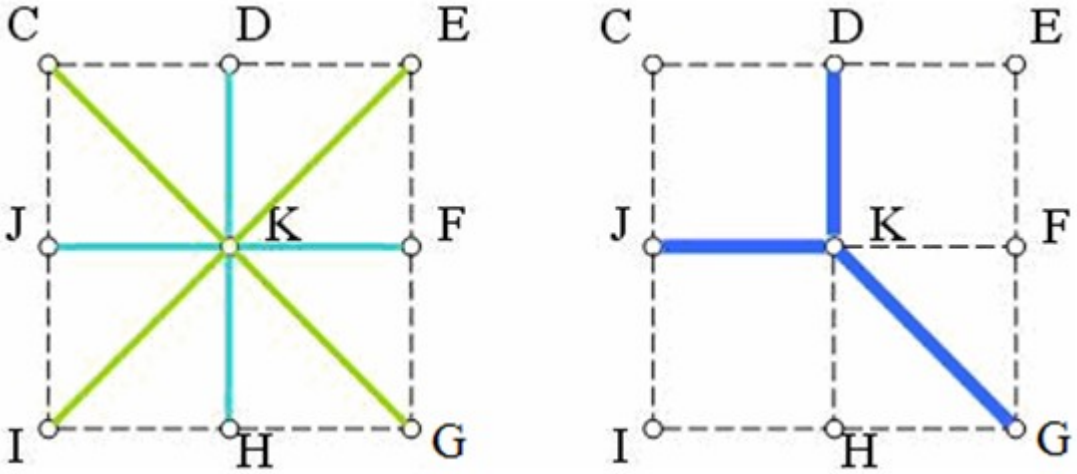
- Routing with preferred direction in X-Architecture will cause increase of via number.
- Liquid routing utilizes the gridless octilinear routing technology.
- In the detailed routing phase, a layer assignment process should be employed to assign the intersectant segments into different layers.

Outline

- Introduction
- Preliminaries
- Dynamic Resource Assignment
- DraXRouter Routing Algorithm
- Experimental Results
- Conclusions

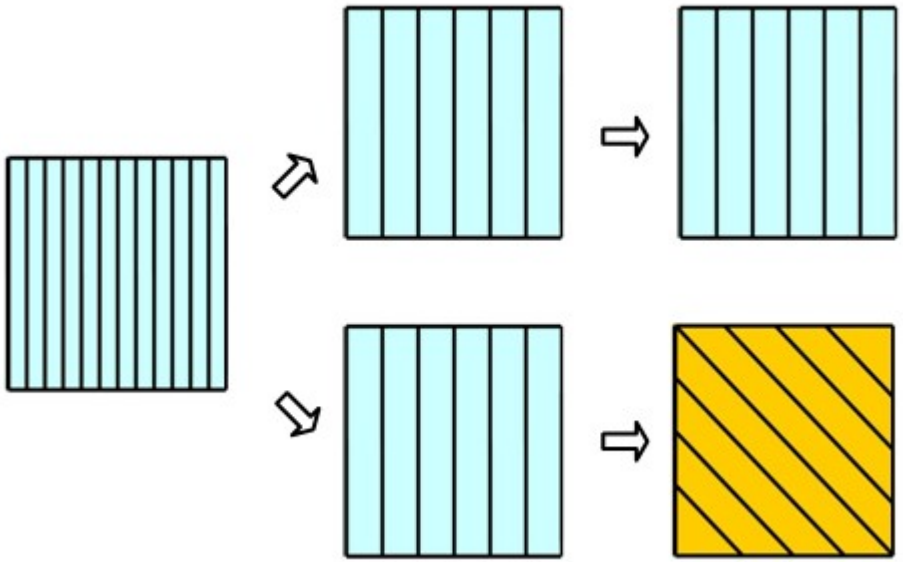
Dra – Capacity Upper Bound C_{eub}

- In liquid routing model, routing in one edge will influence its adjacent edges.
- In Dra, capacity upper bound C_{eub} is given to each edge *e*.
- Routing resources will be assigned dynamically.



Dra – Calculation of C_{eub}

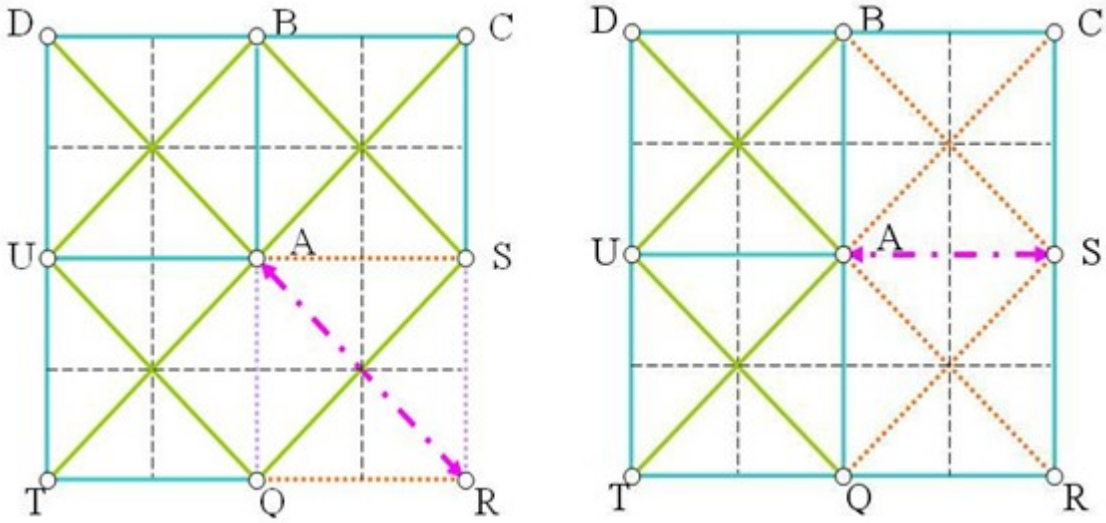
- Calculation of capacities.



- Initial C_{eub} should be twice of the capacity.

Dra – The Assignment Method

- Routing on *AR* will influence *AS, RQ, AQ, SR*.
- Routing on *AS* will influence *AR, SQ, SB, AC*.
- The proportion guarantees the real resources utilized are the same as the traditional methods.



Outline

- Introduction
- Preliminaries
- Dynamic Resource Assignment
- DraXRouter Routing Algorithm
- Experimental Results
- Conclusions

Routing Alg. – Tree Construction (1)

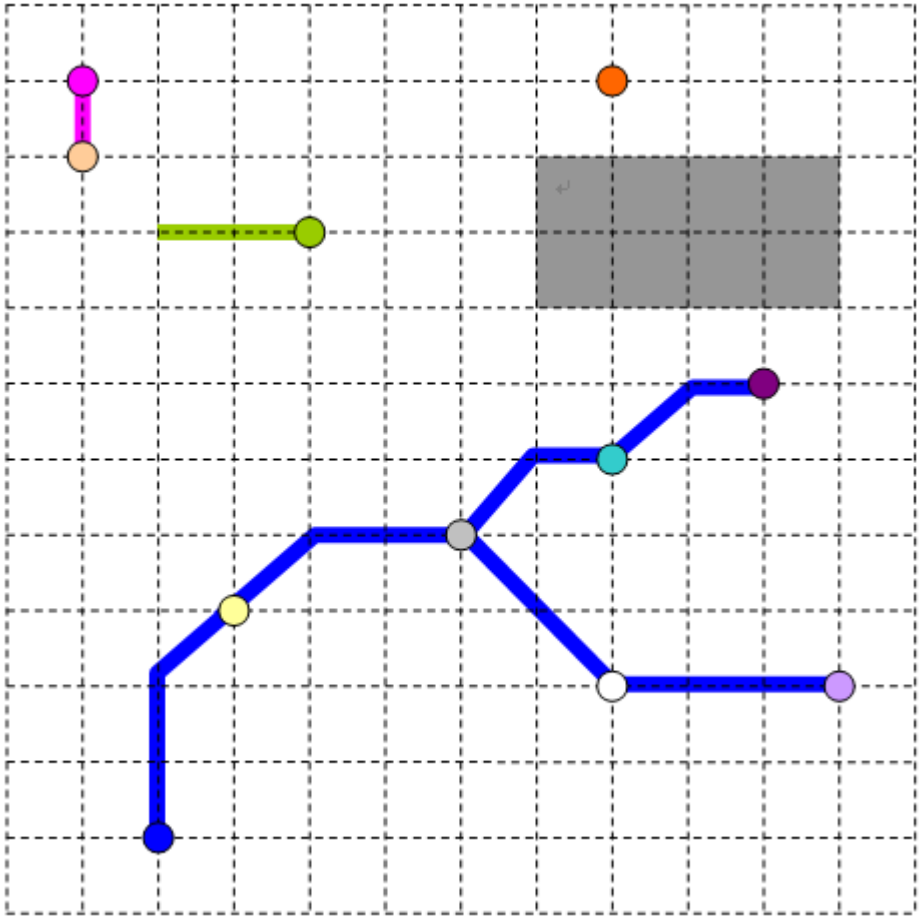
- The heuristic
  - Every pin is initially regarded as a growing point (GP).
  - Each GP will grow toward another GP according to the other GPs' tabulists.
  - When two GPs meet, they will merge into a new GP.
  - Repeat until only one GP left, and the tree is constructed.

Routing Alg. – Tree Construction (2)

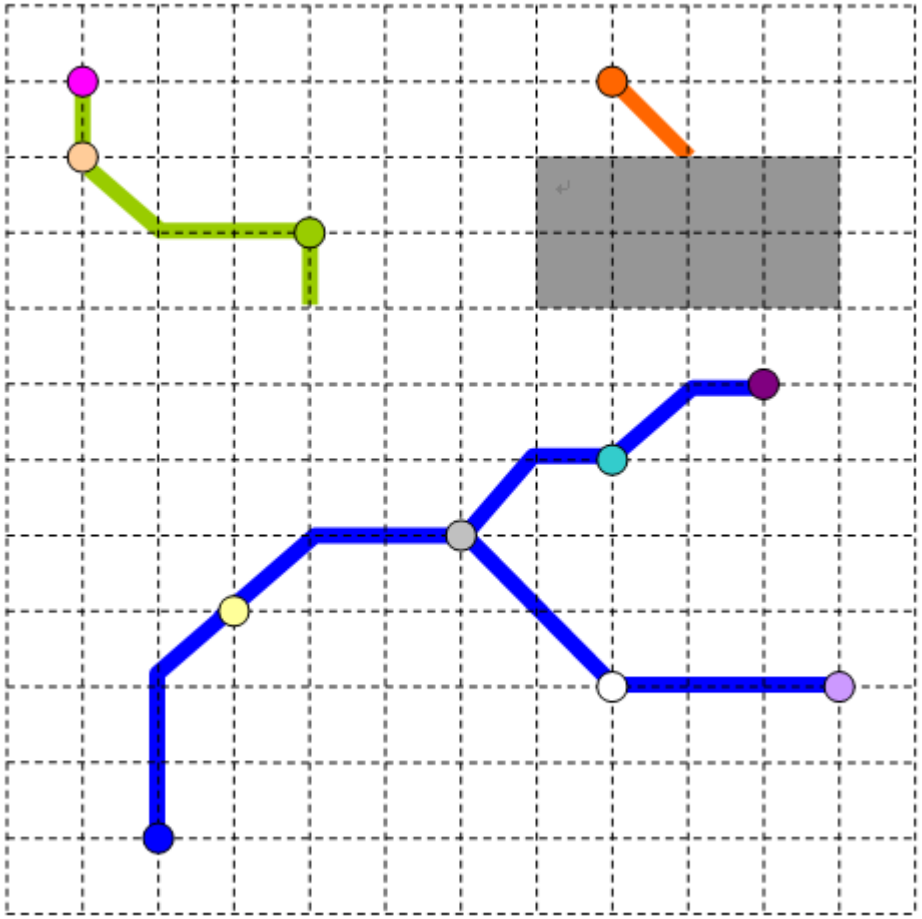
- Dynamic tabulist
  - The current position of a GP and its tabulist is selected dynamically during the construction process.
  - When tabulist of a GP is changed, the influence of this GP to others is changed too.
- Estimation function.

$$f_p(v) = w(p,v) + dis(s,v) + g(p,v)$$

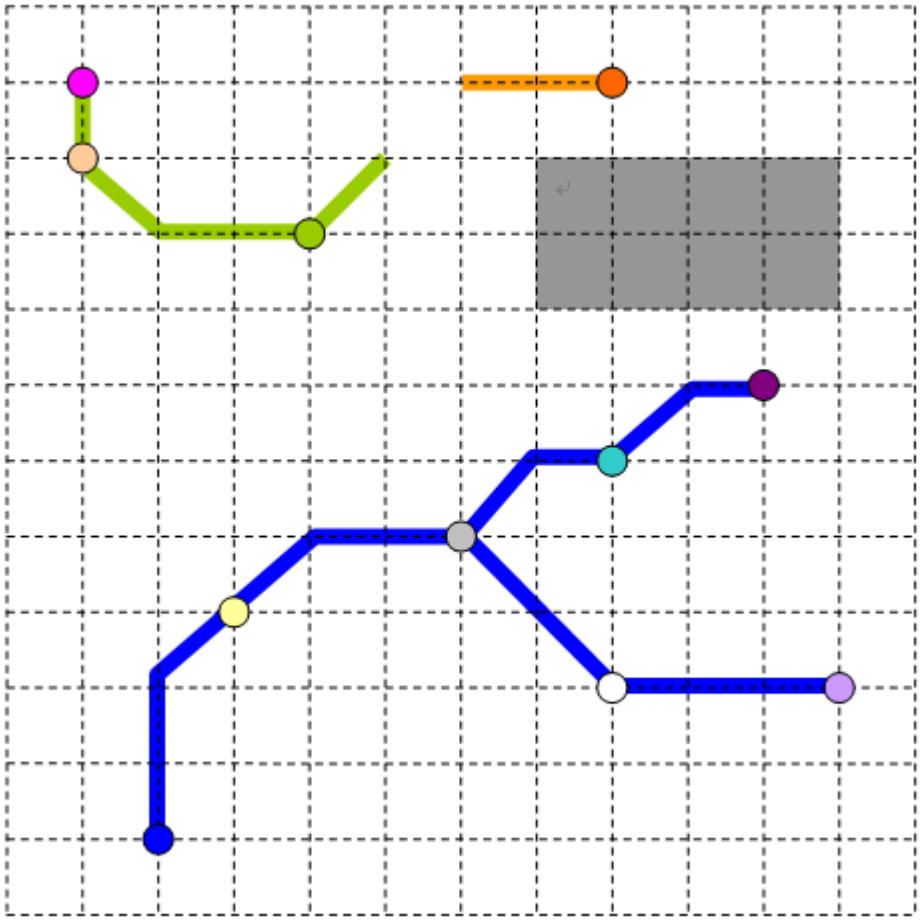
Routing Alg. – An Example (1)



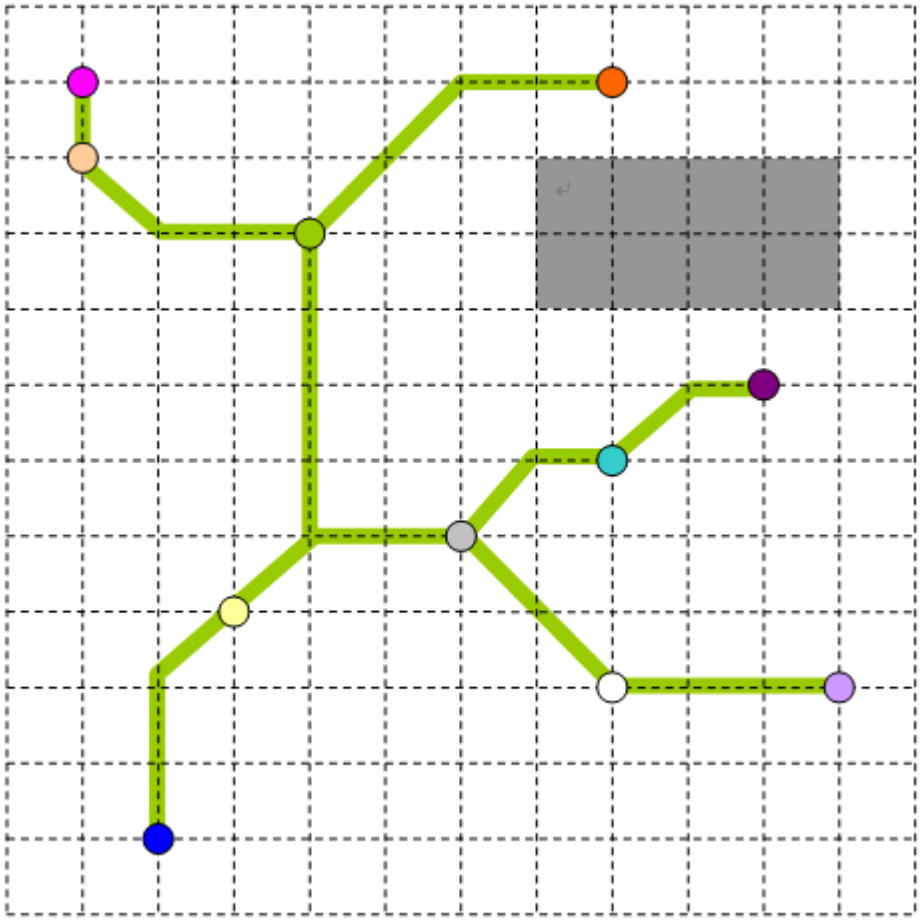
Routing Alg. – An Example (2)



Routing Alg. – An Example (3)



Routing Alg. – An Example (4)



Routing Alg. – Stochastic Optimization (1)

- Rip-up and reroute to refine the solution
  - Pick up all nets over congestion area.
  - Select the nets to be rerouted randomly according to a select function.
  - Reroute these nets.
  - Accept or give up the new solution according to a judgment function.

Routing Alg. – Stochastic Optimization (2)

- Select function $p = P * cnum / tof$
- Judgment function

$$d_{tof} \neq 0 \; and \; d_{twl} \neq 0 \qquad p = 0$$

$$d_{tof} < 0 \; and \; d_{twl} \leq 0 \qquad p = 1$$

$$d_{tof} < 0 \; and \; d_{twl} > 0 \qquad p = e^{-(Q+T)}$$

$$d_{tof} > 0 \; and \; d_{twl} < 0 \qquad p = e^{-(Q+T)}$$

  - p is the probability of acceptance
  - Q is d_{twl} / d_{tof}
  - T is defined as $T = (N / K + 1) * D$

Outline

- Introduction
- Preliminaries
- Dynamic Resource Assignment
- DraXRouter Routing Algorithm
- Experimental Results
- Conclusions

Exp. Results – Experiment Setup

- ISPD'98 benchmarks.
- The X-Implement of UCSB Labyrinth: x-labyrinth.
- Compare DraXRouter with x-labyrinth and COCO.
- Assign the same routing resources.
- Shrink the routing resources.

Exp. Results – Benchmark Data

Circuits	Net#	Grids
ibm01	13k	64×64
ibm02	19k	80×64
ibm03	26k	80×64
ibm04	31k	96×64
ibm05	30k	128×64
ibm06	34k	128×64
ibm07	46k	192×64
ibm08	49k	192×64
ibm09	59k	256×64
ibm10	66k	256×64

Compared with x-labyrinth on *twl*

Circuits	DraX	x-laby	Imp.(%)
ibm01	61011.9	65989.6	7.54
ibm02	168908	169388	0.28
ibm03	148970	163081	8.65
ibm04	163009	174081	6.36
ibm05	408902	421328	2.95
ibm06	275598	280265	1.67
ibm07	349383	355865	1.82
ibm08	389485	394140	1.18
ibm09	412195	416077	0.93
ibm10	565999	570143	0.73
Average	——	——	3.21

Compared with x-labyrinth on *tof, time*

Circuits	DraX		x-labyrinth		*Tof* Imp.(%)	CPU Imp.(%)
	Tof	CPU(s)	*Tof*	CPU(s)		
ibm01	4	36	87	89	95.4	59.55
ibm02	0	78	185	119	100	34.45
ibm03	1	73	217	148	99.54	50.68
ibm04	0	137	199	179	100	23.46
ibm05	0	438	109	587	100	25.38
ibm06	0	331	63	410	100	19.27
ibm07	0	349	80	548	100	36.31
ibm08	0	429	144	701	100	38.8
ibm09	0	734	33	987	100	25.63
ibm10	0	1132	96	1309	100	13.52
Average	——	——	—	——	99.49	32.71

Compared with COCO on *twl*

Circuits	DraX	COCO	Imp.(%)
ibm01	61011.9	74256.8	17.84
ibm02	168908	201484	16.17
ibm03	148970	182767	18.49
ibm04	163009	197607	17.51
ibm05	408902	502694	18.66
ibm06	275598	329378	16.33
ibm07	349383	434956	19.67
ibm08	389485	475549	18.1
ibm09	412195	510459	19.25
ibm10	565999	702568	19.44
Average	——	——	18.15

Compared with COCO on *tof, time*

Circuits	DraX		COCO		*Tof* Imp.(%)
	Tof	CPU(*s*)	*Tof*	CPU(*s*)	
ibm01	4	36	786	18	99.49
ibm02	0	78	299	46	100
ibm03	1	73	494	43	99.79
ibm04	0	137	1022	47	100
ibm05	0	438	900	236	100
ibm06	0	331	748	176	100
ibm07	0	349	503	159	100
ibm08	0	429	425	247	100
ibm09	0	734	1043	312	100
ibm10	0	1132	1183	350	100
Average	——	——	——	——	99.93

Exp. Results – Discussions (1)

- Wire length and Congestion Comparisons
  - Our algorithm can achieve 100% completion for global routing for almost all circuits (8 of 10).
  - Compare with x-labyrinth, DraXRouter reduces total overflow by 99.49%, total wire length by 3.21%, with 32.71% less runtime.
  - Compared with COCO, DraXRouter reduces total overflow by 99.93% and total wire length by 18.15% with an acceptable runtime

Exp. Results – Discussions (2)

- Crossover Number Reduction
  - Crossover number gives a prediction for the number of vias produced by liquid routing.
  - Compare the number of crossed segments produced by our router with and without the Dra Method.
  - Dra method produces 5% lesser crossed segments and the wire length is even shorter.

Outline

- Introduction
- Preliminaries
- Dynamic Resource Assignment
- DraXRouter Routing Algorithm
- Experimental Results
- Conclusions

Conclusions

- Dynamic resource assignment (Dra) method.
- Dynamic-tabulist-based tree construction algorithm.
- Stochastic optimization strategy.
- Compare with two recent global routers, DraXRouter gains a better routing solution.

Thank you.

Q & A