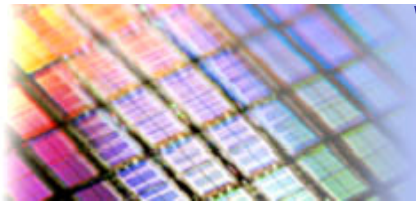
Diffusion-Based Placement Migration

Haoxing Ren, David Pan,
Chuck Alpert, Paul Villarrubia

ECE Dept., UT Austin
IBM Electronic Design Automation
IBM Austin Research Lab

Outline

- Introduction
- Diffusion-based placement migration
  - Mathematical modeling and formulation for placement cell spreading
  - Some implementation details/tricks
    - Velocity computation
    - Initial condition
    - Boundary condition and fixed-macro handling
    - Global versus regional diffusion
- Application to legalization
- Experimental results
- Conclusion and future work



Why Placement Migration?

- Post placement
  - Legalization for post placement buffer insertion, gate sizing, Engineering Change Order (ECO), or decoupling capacitor insertion.
  - Congestion/noise mitigation
  - Thermal/power reduction
- Global placement
  - Cell spreading to reduce overlap

Prior Works

- Network flow [ISPD'04]
- Single row optimization [ICCAD'99, DAC'04]
- Cell movement heuristic [DAC'04]
- Global re-placement starting from a late cut [ICCAD'04]
- However, not intend to honor the "geometry" order
  - Key: smoothly spread out cells

Diffusion

- Physical diffusion is a natural process to diffuse from high-density to low density
- Flux is proportional to the density gradient

$$F = -\nabla D$$

- Density change is inversely proportional to the flow gradient

$$\frac{dD}{dt} = -\nabla F$$

- Therefore

$$\frac{dD}{dt} = \nabla^2 D$$

Diffusion Velocity

- Flux is the product of material density and diffusion speed, thus

$$V = \frac{F}{D} = -\frac{\nabla D}{D}$$

- V can be computed efficiently
- The position of the new cell can be determined by velocity integral

$$X = \int V dt$$

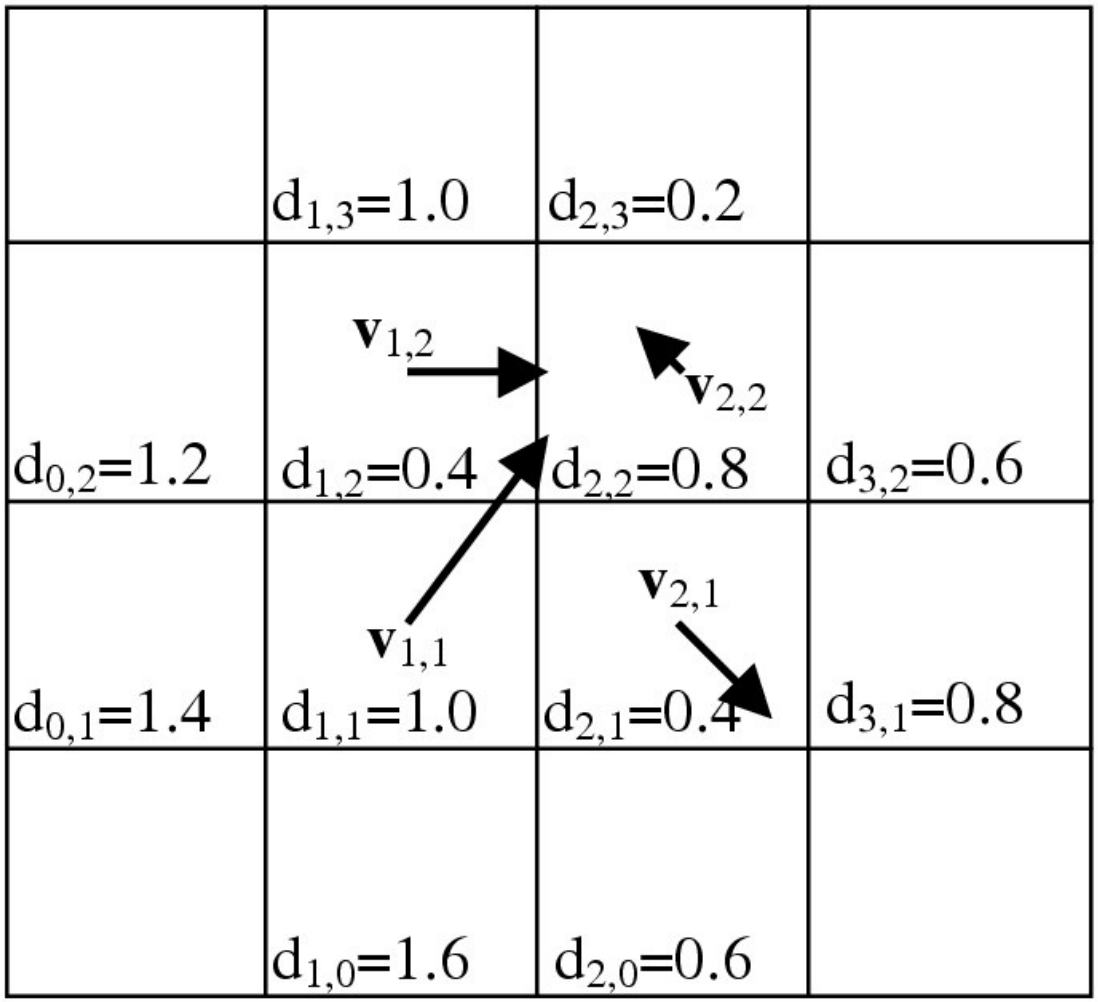
Placement Migration Modeling

- Chip is divided into bins
- Densities are calculated for each bins at the beginning
- Numerically compute the density of each bin by solving the diffusion equation, **do not** recalculate bin density based on real cell placement
- Compute velocity for each bin based on density
- Interpolate velocity inside of a bin
- Compute new placement of cells based on the integral of the velocity field

Pseudo code

- Initialize bin density $d_{j,k}(0)$
- Set $t=0$
- Repeat
  - Compute bin velocity $v_{j,k}(t)$
  - Compute cell coordinates $x_i(t)$ based on velocity integral and velocity interpolation
  - Compute $d_{j,k}(t+1)$ based on $d_{j,k}(t)$
  - Set $t = t+1$
- Until $t=T$ or $max(d_{j,k})<d_{max}$

Bin Density/Velocity Example



Compute Density

- Using Forward Time Centered Space (FTCS) to discretize

$$\frac{dD}{dt} = \nabla^2 D$$

$$d_{j,k}(n+1) = \frac{\Delta t}{2}(d_{j+1,k}(n) + d_{j-1,k}(n) - 2d_{j,k}(n))$$

$$+ \frac{\Delta t}{2}(d_{j,k+1}(n) + d_{j,k-1}(n) - 2d_{j,k}(n))$$

$$+ d_{j,k}(n)$$

Compute Velocity

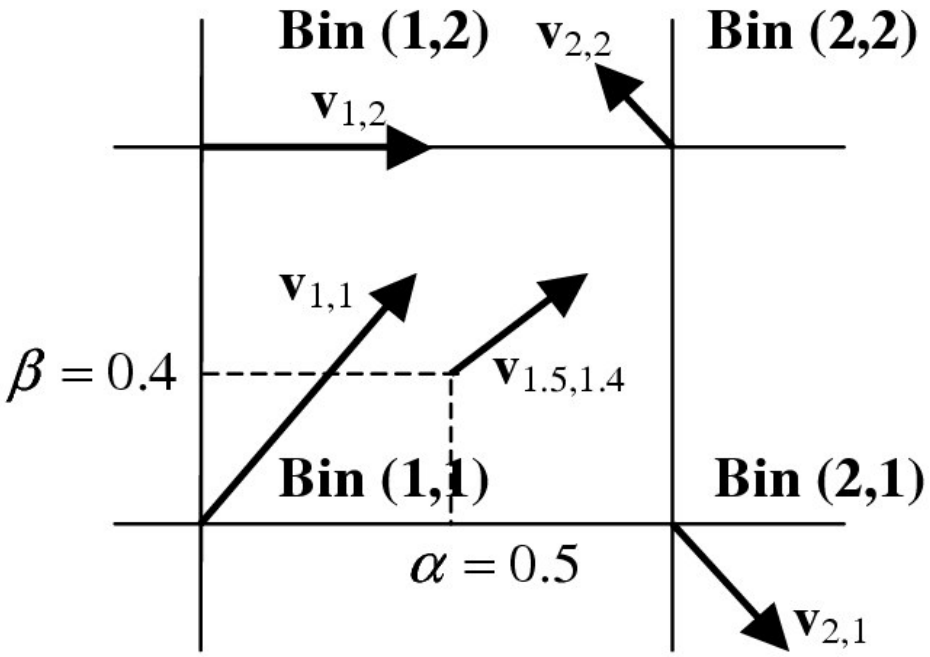
- Discretize

$$V = -\frac{\nabla D}{D}$$

$$vx_{j,k}(n) = -\frac{d_{j+1,k}(n) - d_{j-1,k}(n)}{2d_{j,k}(n)}$$

$$vy_{j,k}(n) = -\frac{d_{j,k+1}(n) - d_{j,k-1}(n)}{2d_{j,k}(n)}$$

Velocity Interpolation



$$v_{x,y}(n) = \alpha(v_{j+1,k} - v_{j,k}) + \beta(v_{j,k+1} - v_{j,k}) + \alpha\beta(v_{j,k} + v_{j+1,k+1} - v_{j+1,k} - v_{j,k+1})$$

Legalization Problem

- Find a placement with maximum density less than or equal to 1
- Keep the original placement relative order, can be formulated to
  - Minimize total cell displacement (linear or squared)
  - Minimize total net displacement
  - No existing algorithm to optimize for both criteria
- Diffusion is a natural process to achieve density target while maintaining relative order

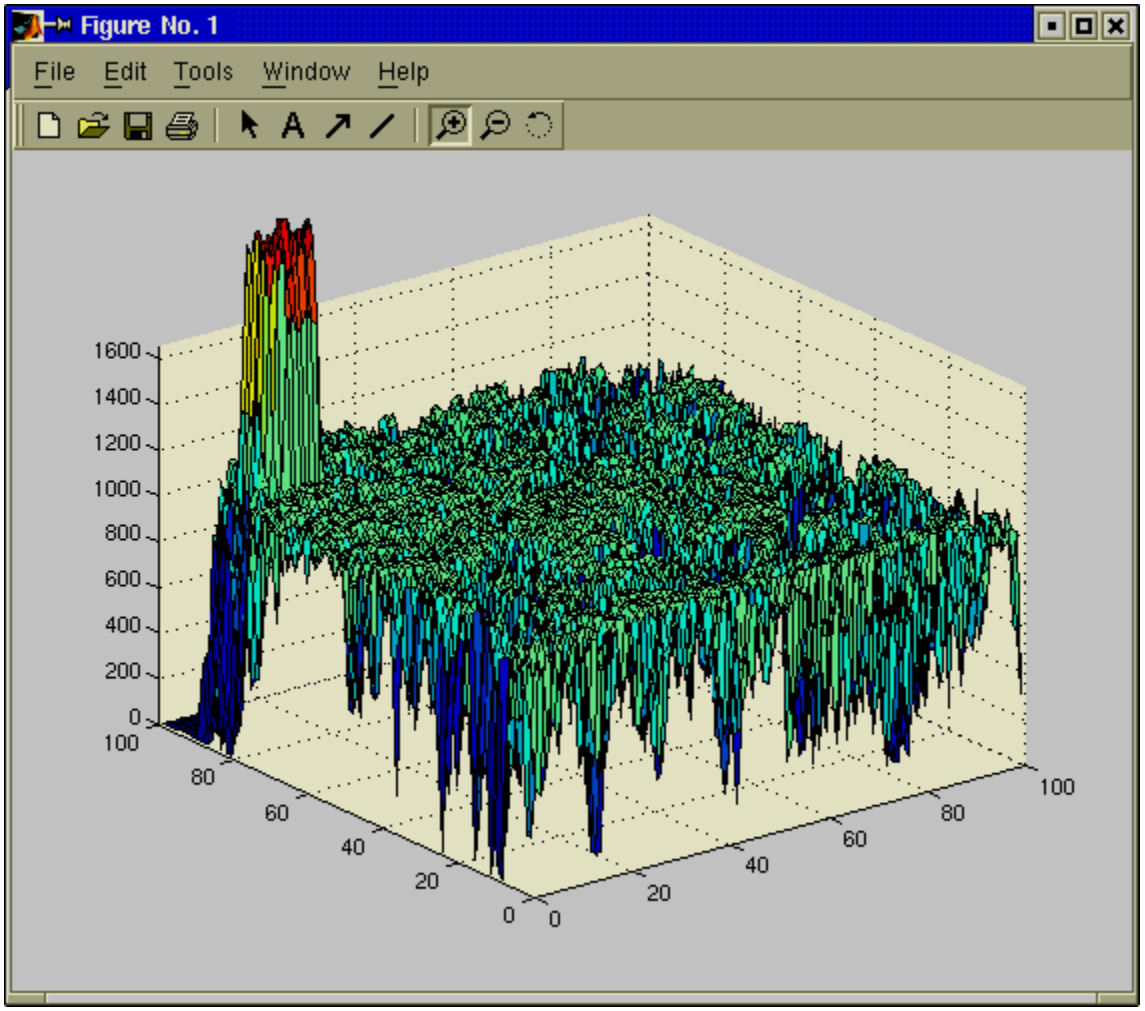
Density Manipulation

- Global diffusion
  - Densities of all the bins are scaled such that the average density of the new densities is equal to the targeted density.
  - Update densities of all the bins
- Regional diffusion
  - Keep the original density value
  - Only update densities of bins adjacent to bins above targeted density

Experimental Results

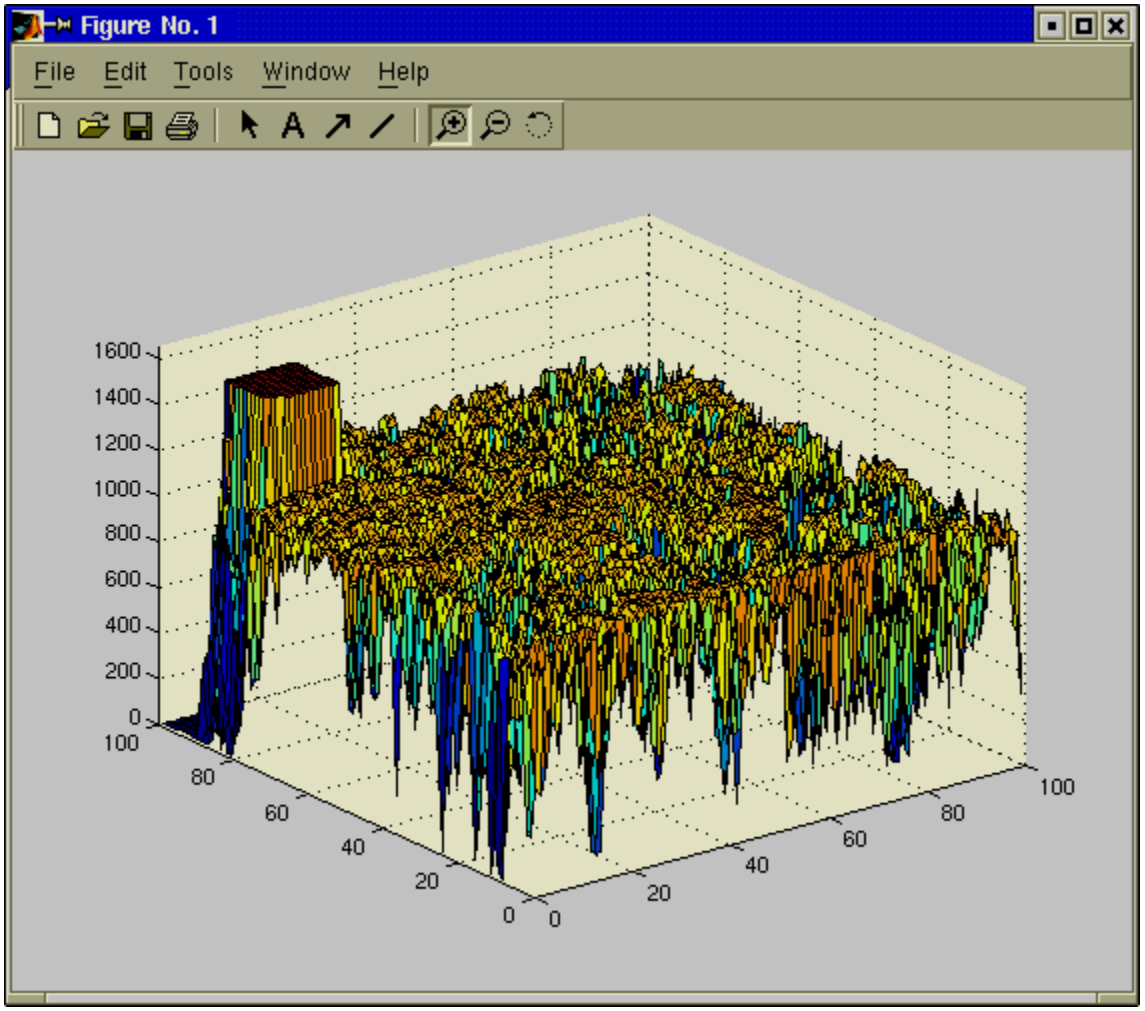
- Example of bin density simulation during diffusion
- Example of relative order before/after legalization
- Legalization result for designs with overlaps, comparing with flow based & greedy legalizer
  - By pseudo expansion
  - By real post placement transforms, such as buffering, re-powering, logic restructuring
  - Concentrated or distributed
  - Runtime

Density Simulation



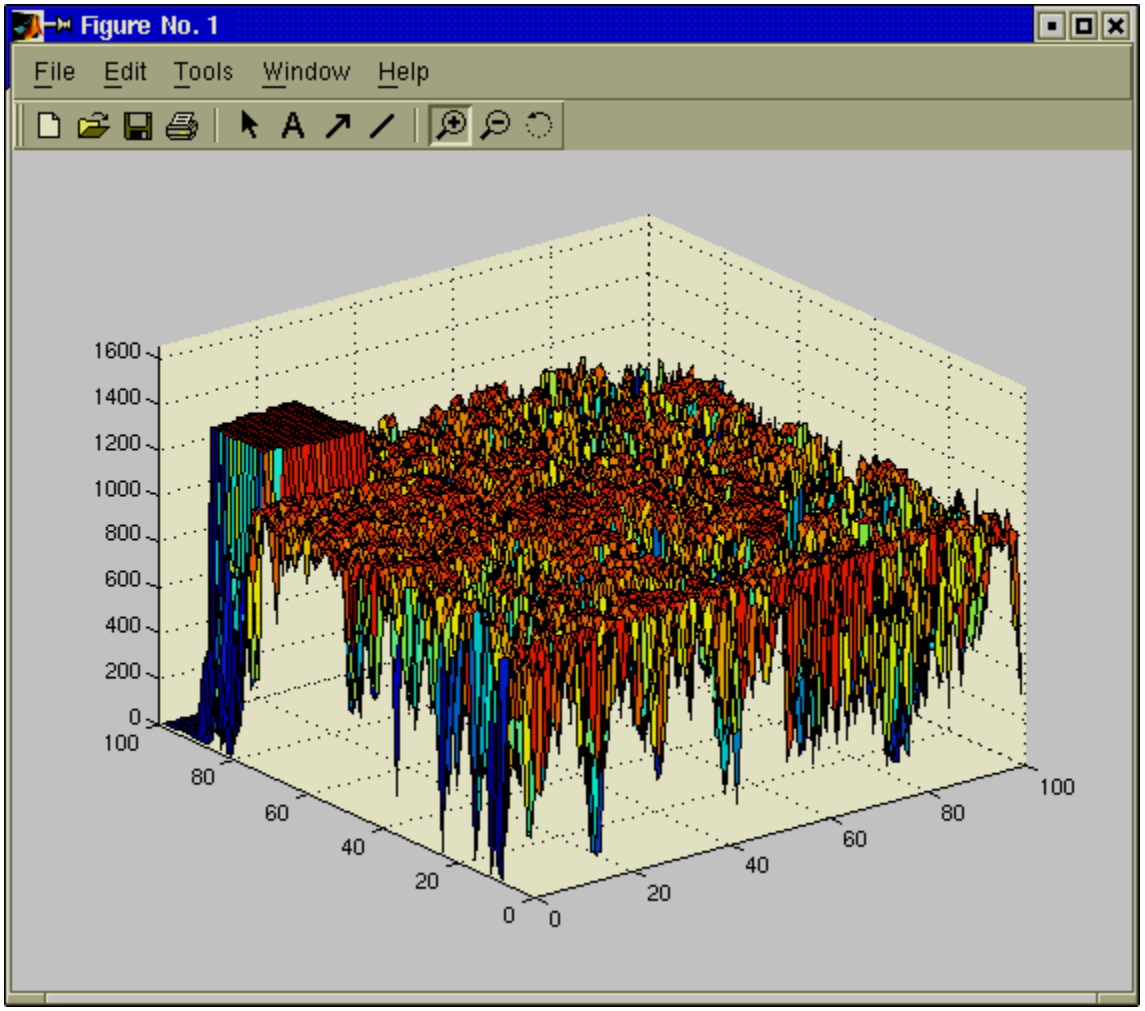
T=0

Density Simulation



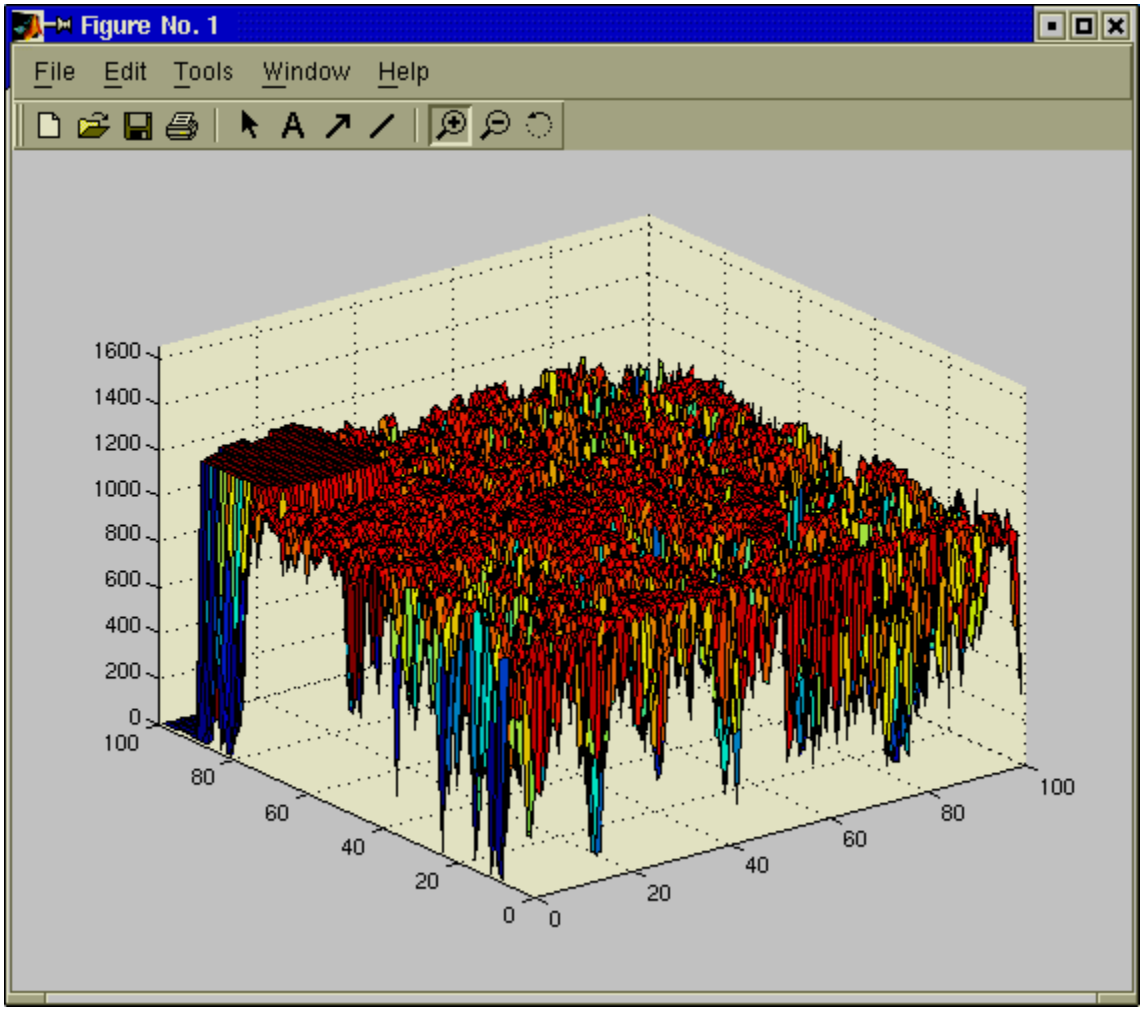
T=100

Density Simulation



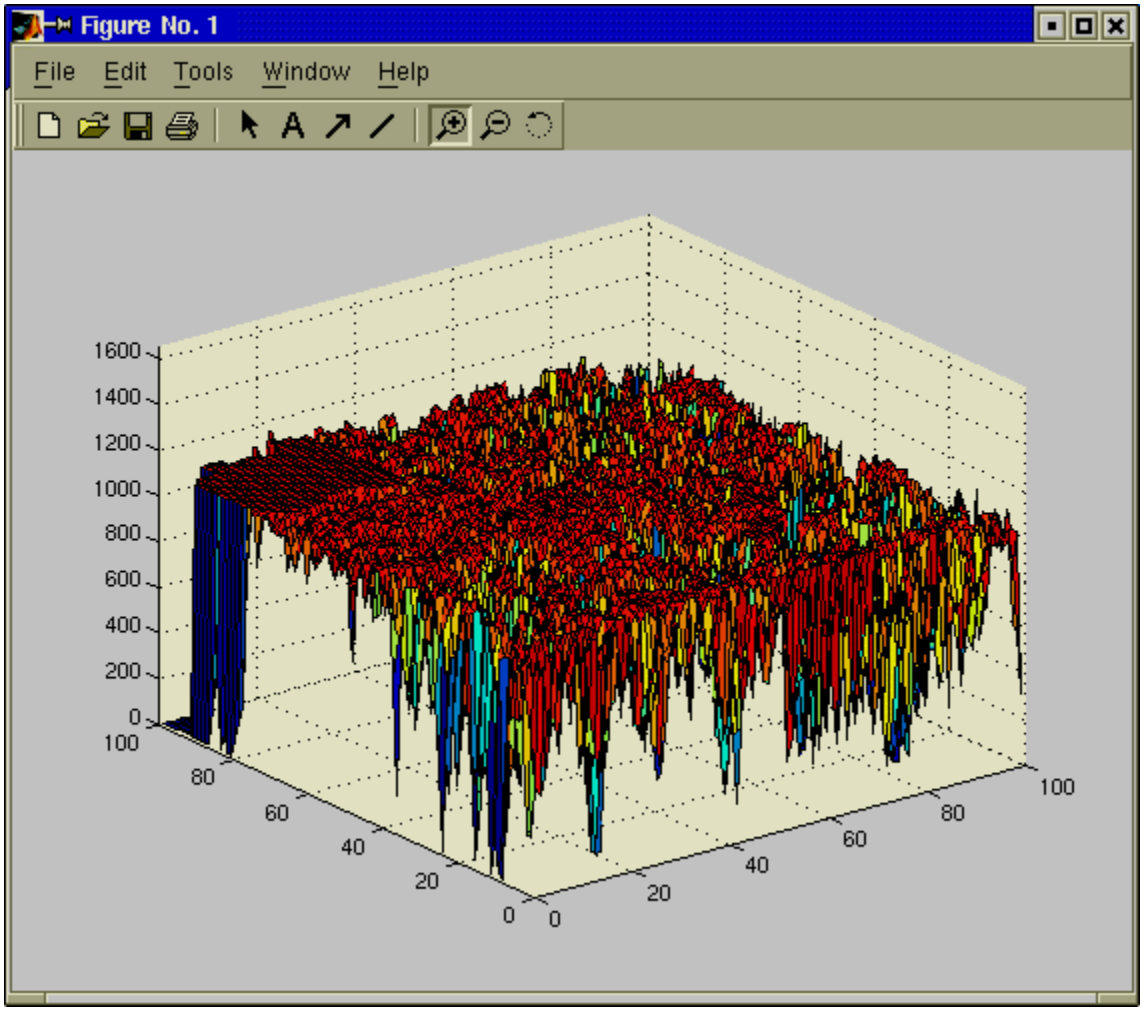
T=200

Density Simulation



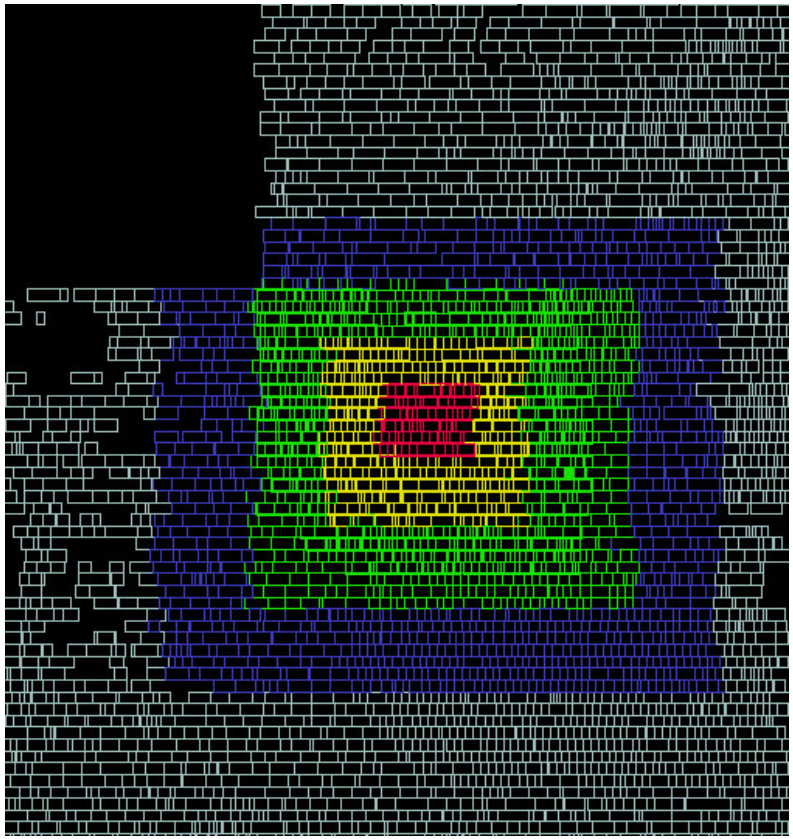
T=300

Density Simulation

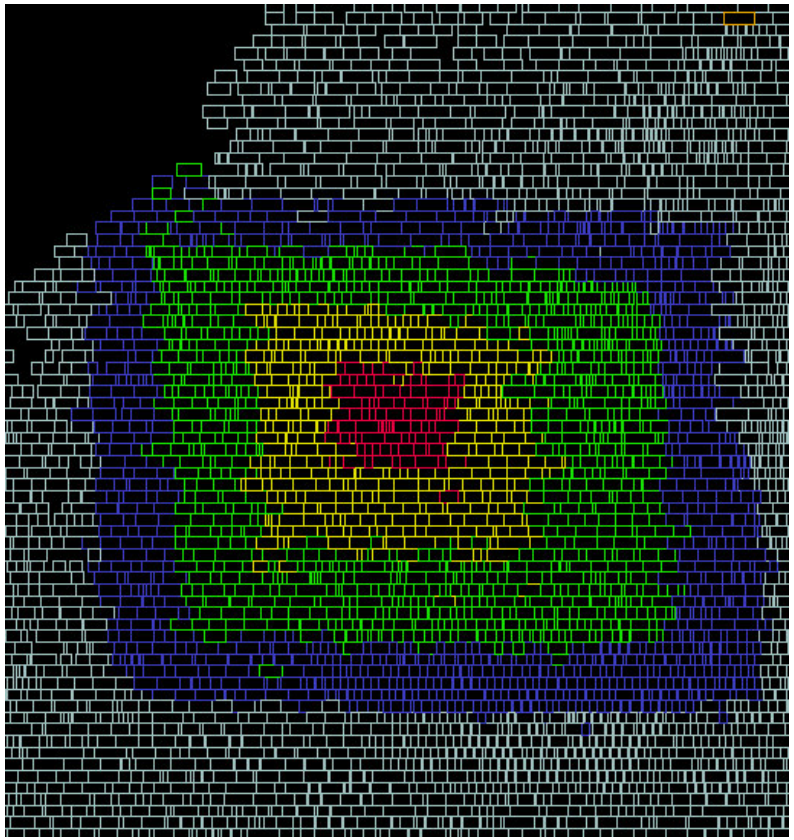


T=400

Relative Order Preserved



Overlapped



After legalization

Expt. Results on IBM Ckts

testcases	Base	GREED	FLOW	DIFF	%improv
ckt1	-0.571	-1.266	-1.497	-0.921	50
ckt2	0.275	-0.287	-0.789	-0.065	40
ckt3	-0.265	-1.155	-1.121	-0.265	100
ckt4	-1.592	-3.569	-3.447	-2.373	58
ckt5	-0.623	-6.072	-3.64	-2.047	52
ckt6	-0.387	-3.45	-3.562	-3.305	5
ckt7	-0.796	-1.601	-1.274	-0.796	100

Worst Slack

testcases	Base	GREED	FLOW	DIFF	%improv
ckt1	11.48	13.23	13.4	12.46	44
ckt2	15.06	17.03	17.33	16.65	19
ckt3	47.1	52.47	52.65	51.76	13
ckt4	51.37	59.02	58.67	56.85	25
ckt5	150.8	159	159.2	158.7	3
ckt6	166.6	175.6	175.4	174.8	8
ckt7	367.7	382.7	382.5	381.7	5

TWL

Run Placement Driven Synthesis (PDS) with non-overlap mode, then use cell expansion to generate illegal placement.

Overlaps by Physical Transforms

	GlobalD	Flow	Greedy	Base
ckt1	-0.949	-2.72	-1.64	-0.708
ckt2	-1.561	-1.569	-1.567	-1.567
ckt3	-0.498	-0.493	-0.492	-0.479
ckt4	-0.845	-0.841	-0.845	-0.846
ckt5	0.054	-0.182	-0.047	0.157
ckt6	-1.13	-1.135	-1.132	-1.125
ckt7	-1.221	-2.793	-1.222	-0.612

Worst Slack

	GlobalD	Flow	Greedy	Base
ckt1	-4839	-6614	-5475	-4058
ckt2	-1110	-1098	-1102	-1077
ckt3	-462	-462	-460	-451
ckt4	-390	-396	-400	-196
ckt5	-218	-387	-380	-62
ckt6	-2747	-2820	-2746	-2659
ckt7	-14364	-14340	-14100	-12461

FOM

Run IBM PDS with overlap produced by physical synthesis

Overlap Distribution Effect

	TWL		SLK		FOM	
type(%)	FLOW	DIFF	FLOW	DIFF	FLOW	DIFF
D(23)	13.4	12.46	-1.497	-0.921	-8441	-3883
C(18)	14.46	12.62	-1.976	-1.253	-11822	-4361

Centralized (C) vs. Distributed (D) overlaps

Diffusion algorithm works extremely well for tough (centralized) overlaps

Runtime Comparison

Runtime	RegionalD	GlobalD	Flow	Greedy
ckt1	85	245	60	167
ckt2	163	945	311	132
ckt3	118	506	136	103
ckt4	275	1441	758	217
ckt5	44	219	45	38
ckt6	1409	4789	1886	1010
ckt7	956	2018	900	520

Run on IBM-AIX S85

Conclusion

- Diffusion based placement migration spreads the placement smoothly
  - more likely to preserve the integrity of the original placement.
- It uses local information to compute cell movement, therefore its implementation is both simple and efficient.
- Future work
  - Timing aware diffusion
  - Thermal, congestion, …