A Performance-Driven Standard-Cell Placer Based on a Modified Force-Directed Algorithm*

Yih-Chih Chou Youn-Long Lin

Department of Computer Science

National Tsing Hua University

Hsin-Chu, Taiwan, R.O.C

* Supported in part by the National Science Council, R.O.C

Outline

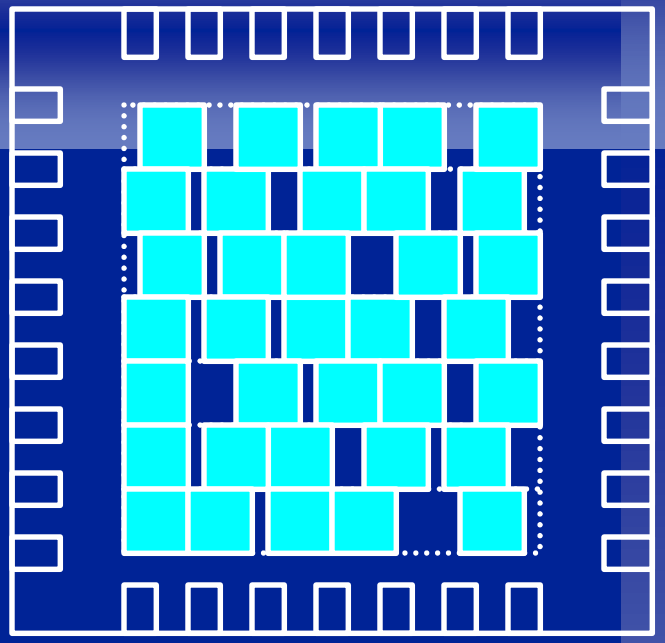
- Motivation
- Graph Model
- Proposed Approach
- Illustrated Example
- Experimental Flow
- Experimental Results
- Conclusions and Future Work

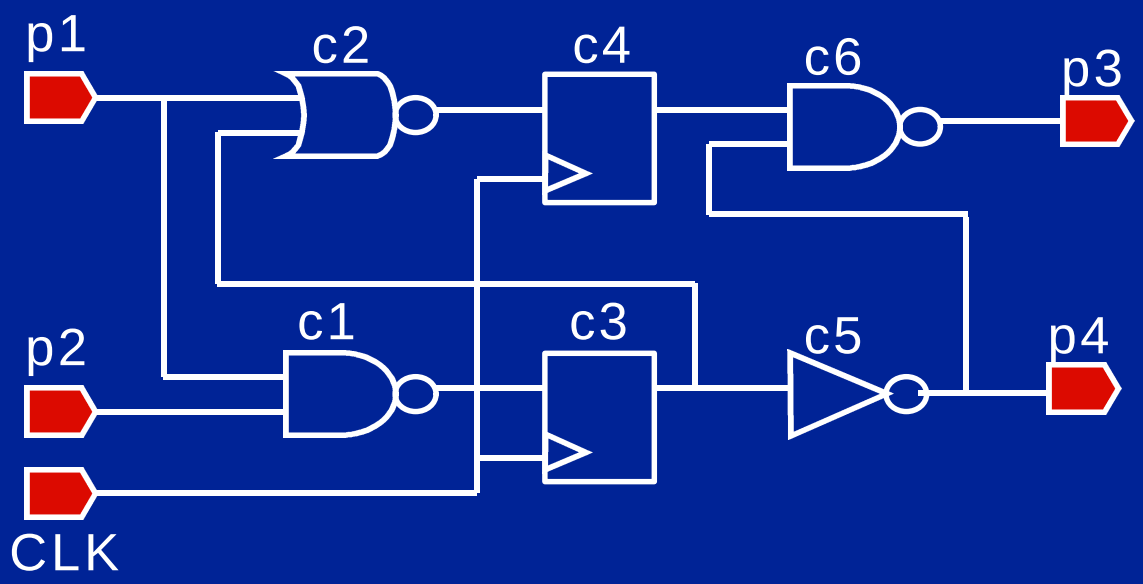
Motivation

- Force-Directed Iterative Refinement
  - Traditional approach:
    - Have to resolve overlapping; Convergence problem
  - This work:
    - We allow overlapping to get a relative placement
    - Move all cells until force-equilibrium is reached
- Path Delay Constraint for Performance-Driven Placement
  - Traditional Approach:
    - Indirectly handled by distributing timing slack among nets along the path (Zero-Slack)
  - This work:
    - We introduce pseudo link between start and end points of each path

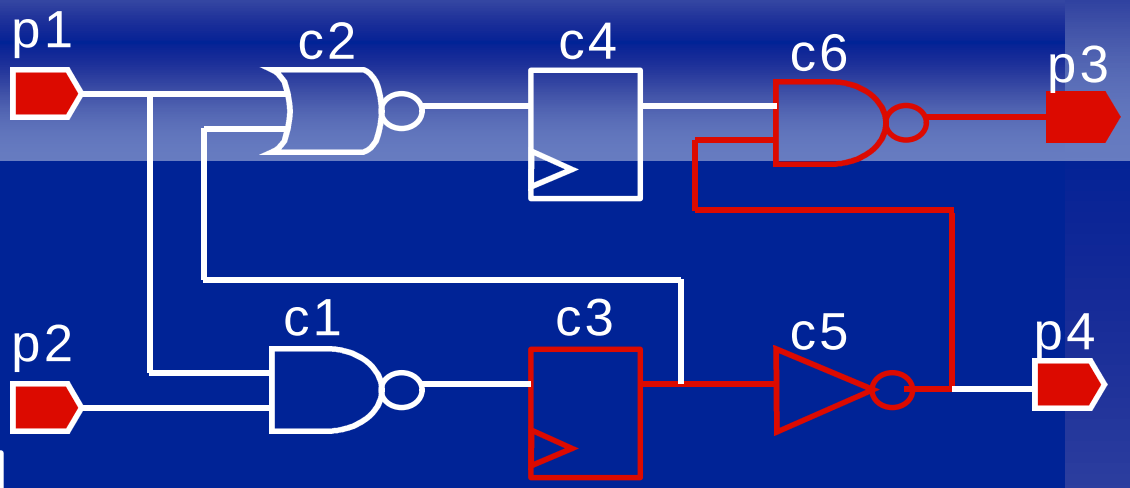
Preliminaries

- Core cells C = $\{c_1, c_2, \ldots, c_m\}$
- I/O pad cells P = $\{p_1, p_2, \ldots, p_n\}$
- Nets N = $\{n_1, n_2, \ldots, n_k\}$





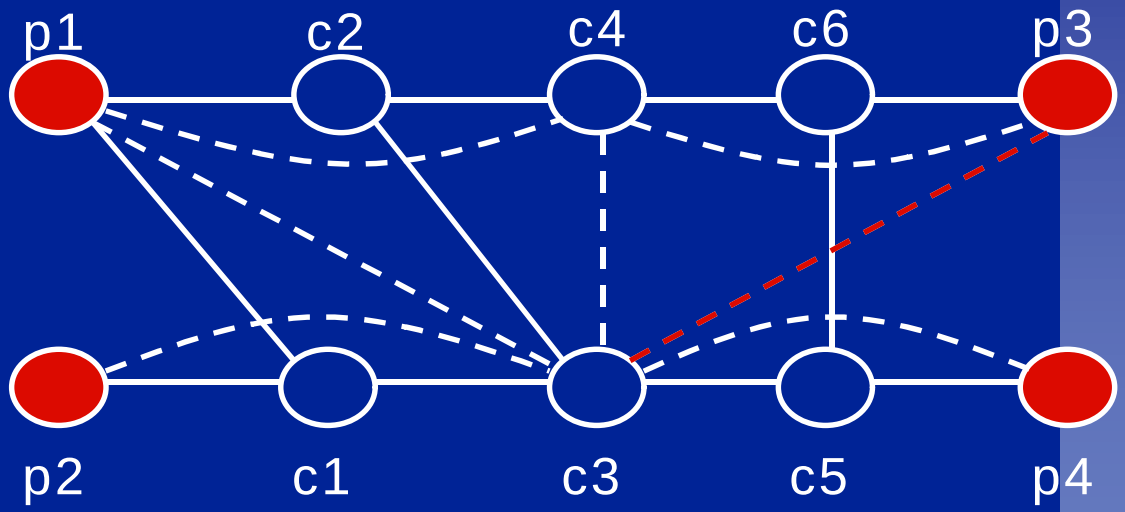
Graph Model



Node: I/O or core cell

Normal (Solid) Link: Cell connectivity

Pseudo (Dashed) Link: One per path



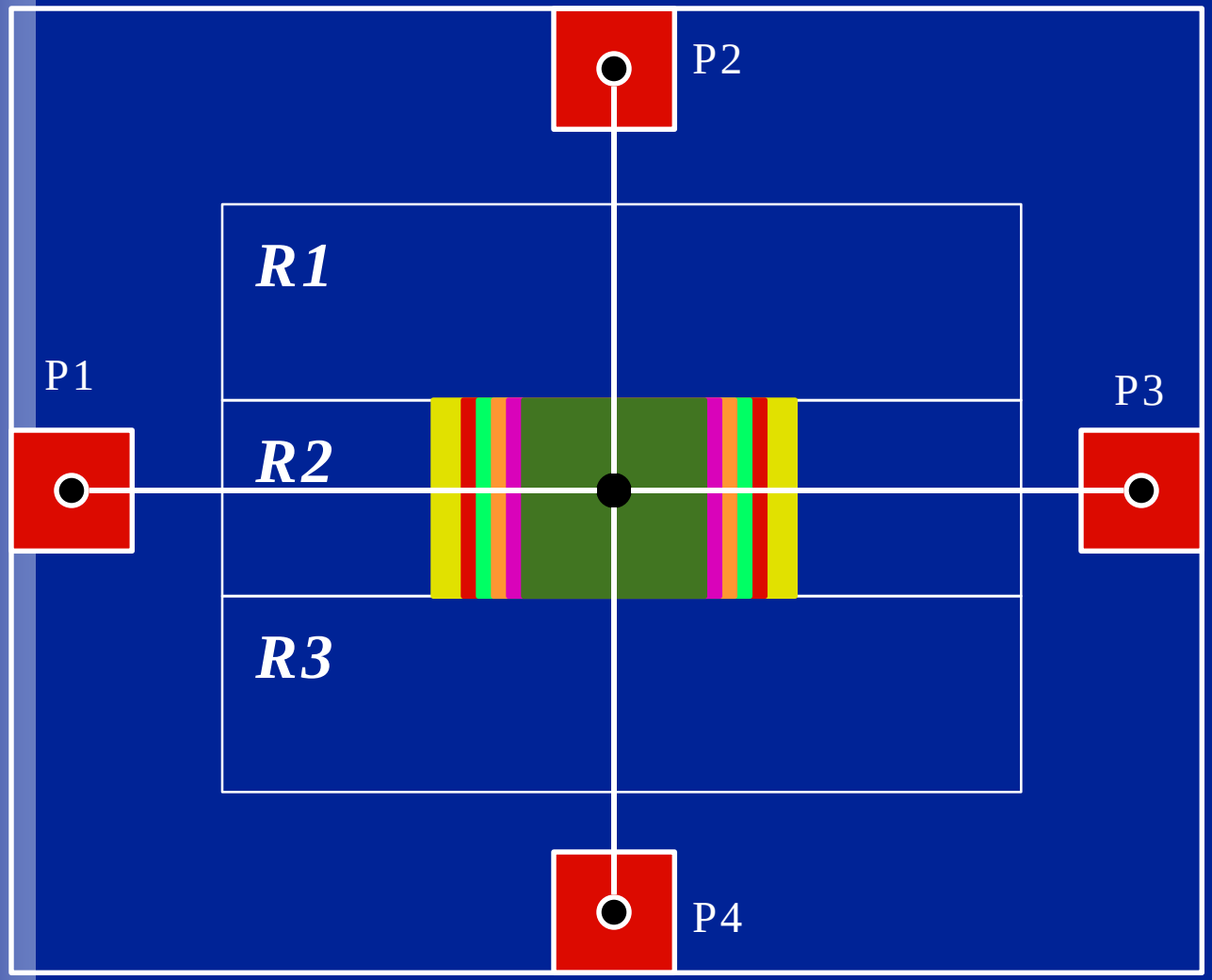
Force Definition

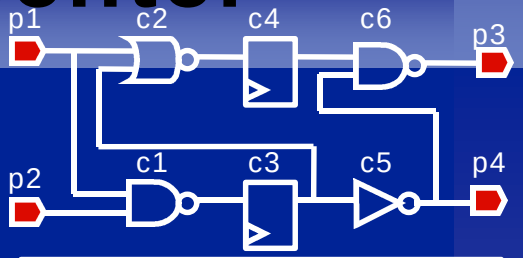
$$f(c_i, c_j) = \begin{cases} (x_i - x_j)^2 + (y_i - y_j)^2 & normal\_link \\ ((x_i - x_j)^2 + (y_i - y_j)^2)(Delay(path_{i \to j}))^a & pseudo\_link \end{cases}$$

Proposed Approach

- Step 1:
  - Floorplan, Fix I/O pad location
  - Construct graph; Add pseudo link between path's starting and ending points
  - Put all core cells at chip center
- Step 2:
  - Iteratively move core cells until all reach force-equilibrium positions
    - Vertically aligned to cell rows
    - Horizontal overlapping is allowed
- Step 3:
  - Form cell rows starting with topmost and bottommost
  - Re-balancing the remaining cells and iterate

Step 1: All 6 cells at chip center





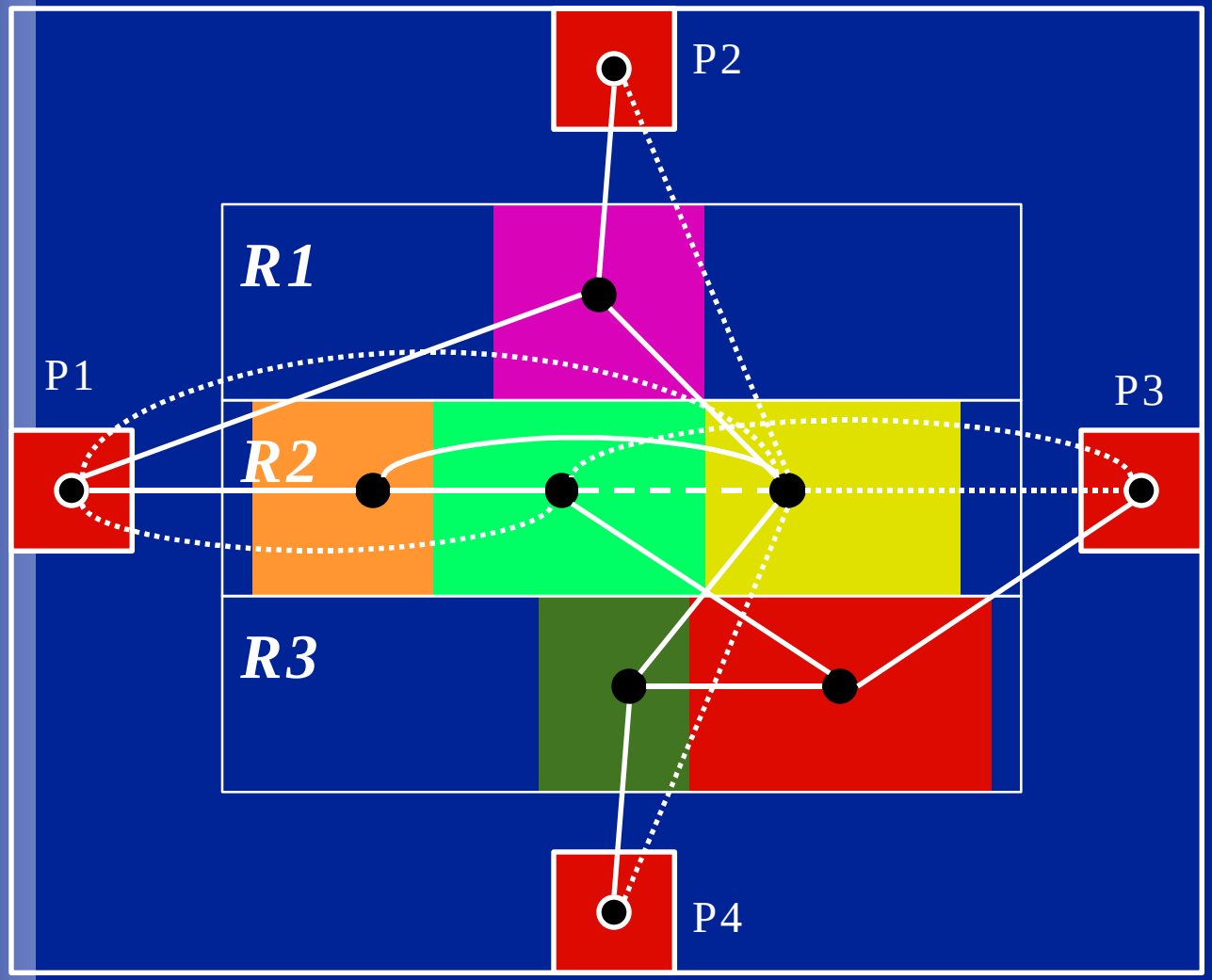
Graph Construction

Force-Equilibrium
Cell Positioning

Cell Row Formation

c1 c2 c3
c4 c5 c6

Completion of Step 2
Overlapping exists



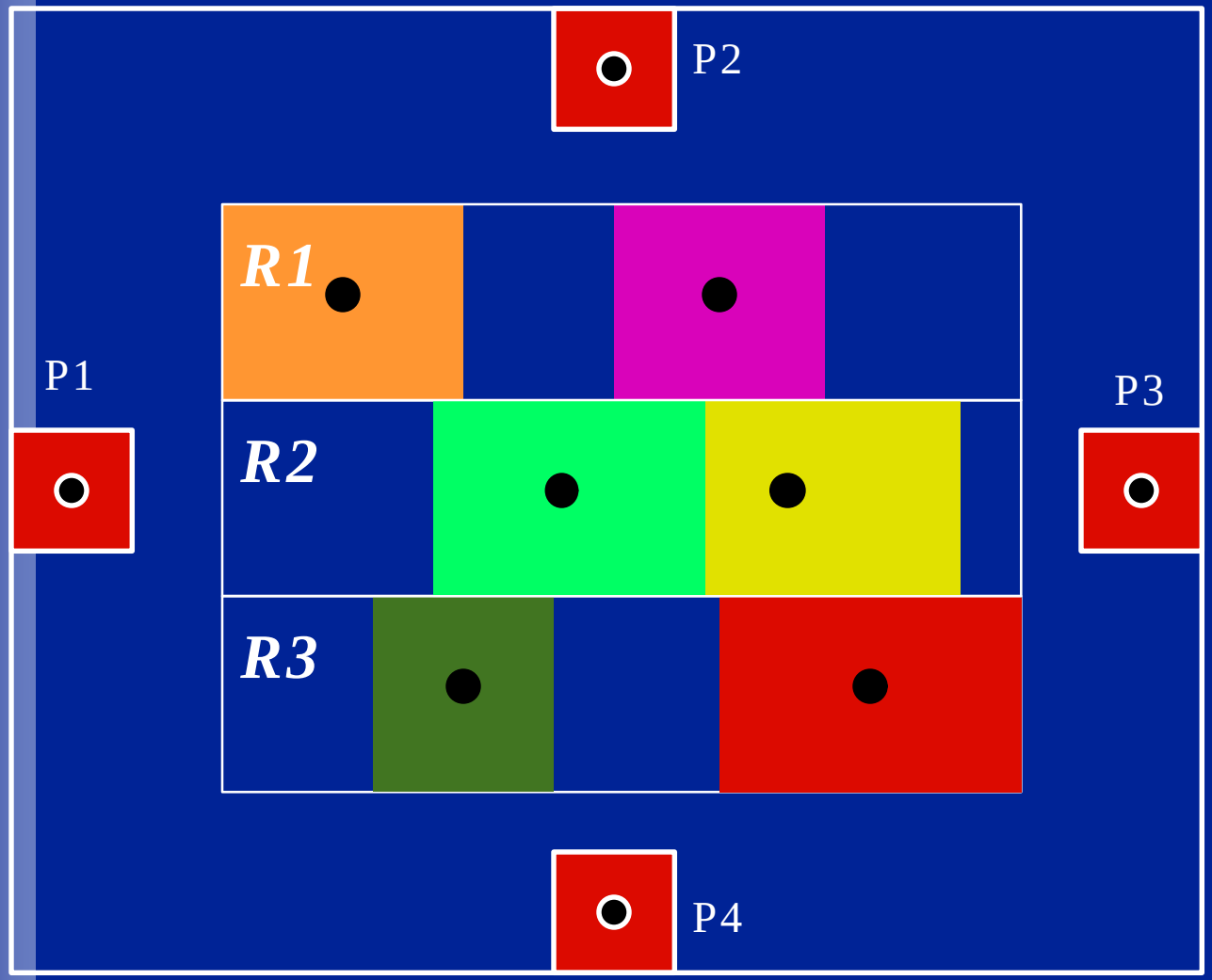
Graph Construction

Force-Equilibrium
Cell Positioning

Cell Row Formation

c1 c2 c3
c4 c5 c6

Step 3: Form Rows R1 and R3



Graph Construction

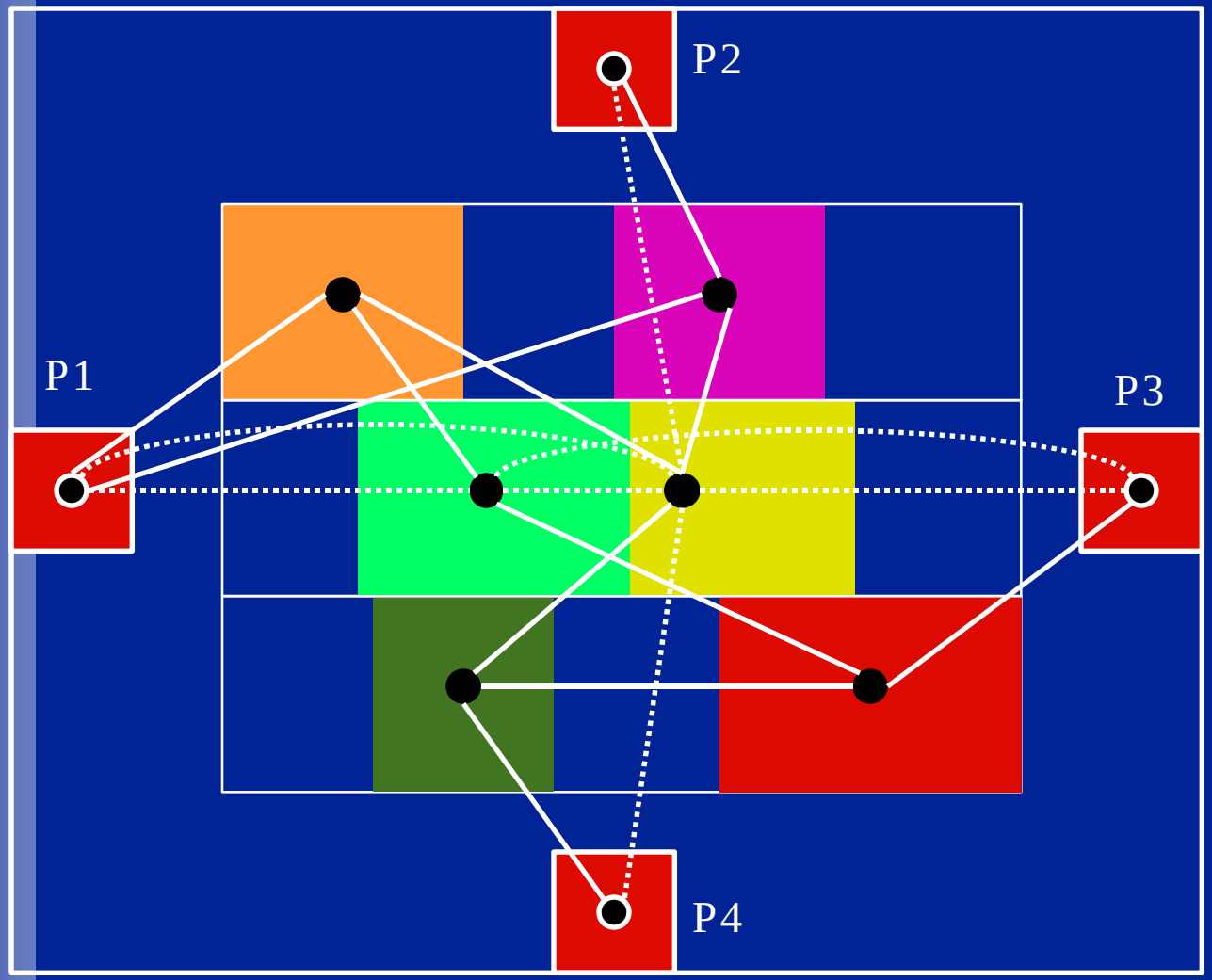
Force-Equilibrium Cell Positioning

Cell Row Formation

c1 c2 c3

c4 c5 c6

Re-Balancing remaining 2 cells in the middle



Graph Construction

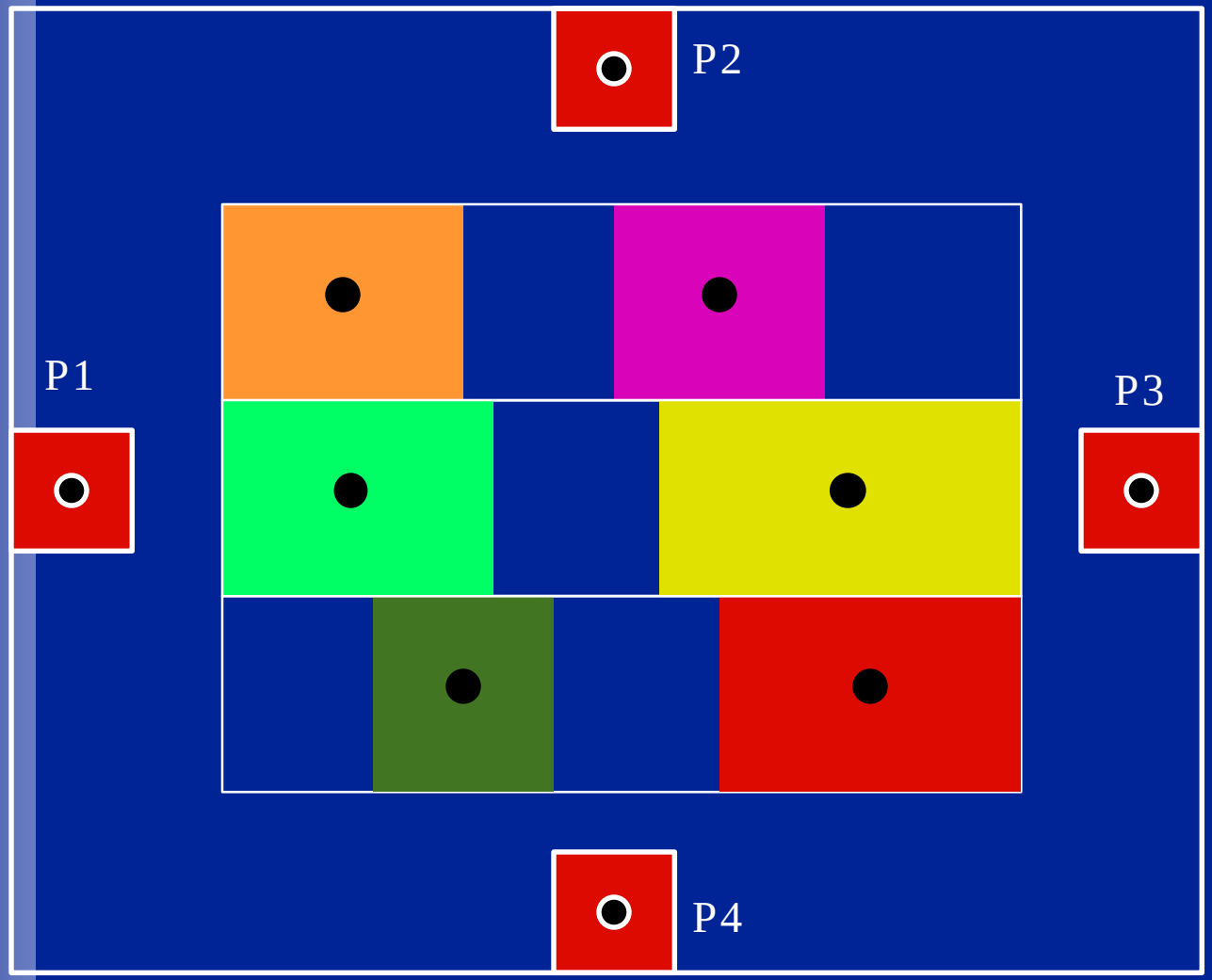
Force-Equilibrium Cell Positioning

Cell Row Formation

c1 c2 c3

c4 c5 c6

Final placement after forming Row R2



Graph Construction

Force-Equilibrium Cell Positioning

Cell Row Formation

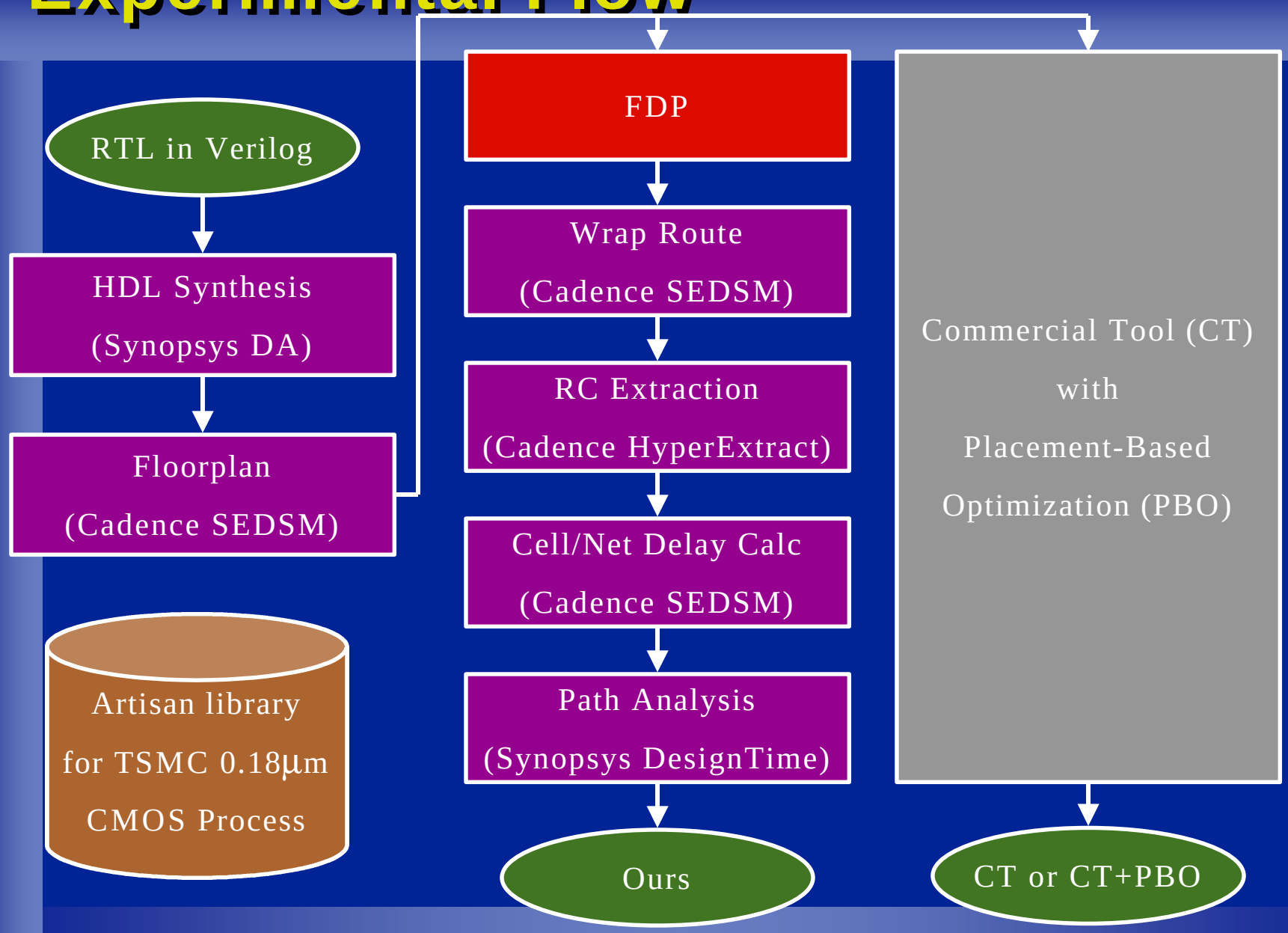
c1 c2 c3

c4 c5 c6

Make Step 2 run faster

- The force associated with pseudo link is much stronger than that with normal link
- First, we only let Flip-Flops move
- Then, we let all cells move
- 15% Reduction in Total CPU Time

Experimental Flow



Benchmark Characteristics

Benchmark	# cells	# nets	# I/O	area(ìm^2)
matrix	3375	3603	119	227405
sdram_rdr	4125	4559	95	365698
32bMAC	8655	8941	213	362695
VP2	10063	10542	323	657251
64bMAC	27043	27458	417	1210814
a259k	95765	104683	153	4392336
a518k	191592	209354	153	8230874

Available from: http://www.cs.nthu.edu.tw/~ylin/placement.htm

Quality and Run Time Comparison

Benchmark	CT		Ours		Ours (From CT)		CT+PBO		Ours (From CT+PBO)	
	Delay	CPU	Impr. %	CPU	Impr. %	CPU	Impr. %	CPU	Impr. %	CPU
matrix	8.42	6	7.3	9	7.1	8	6.5	10	6.9	8
sdram_rdr	2.81	35	5.9	56	6.2	51	3.4	116	7.4	53
32bMAC	4.85	154	16.2	150	16.9	139	9.3	578	18.5	122
VP2	13.66	276	13.9	229	13.3	207	5.8	662	15.0	211
64bMAC	4.97	509	14.1	486	14.5	430	8.5	1802	17.1	395
a259k	12.35	13196	17.4	11906	17.7	10834	11.3	33122	19.1	9914
a518k	14.26	46313	17.1	41349	16.7	38455	8.7	110688	18.8	36276
total		60489		54185		50124		146908		46979

Delay in (ns)

CPU in (sec) running on Sun UtraSparc80

PBO Area Overhead: 3.81%

How much should we weigh the Pseudo-Link(a) ?

Benchmark	1		1.5		2		2.5		3	
	Delay(ns)	CPU(s)	Delay	CPU	Delay	CPU	Delay	CPU	Delay	CPU
matrix	7.80	8	7.79	9	7.81	9	7.85	9	7.90	11
sdram_rdr	2.64	60	2.69	58	2.64	56	2.61	56	2.88	63
32bMAC	4.29	156	4.19	163	4.06	150	4.30	146	4.35	171
VP2	12.45	241	12.09	263	11.76	229	11.89	218	12.35	358
64bMAC	4.50	507	4.33	531	4.27	486	4.21	492	4.38	625
a259k	11.15	11989	10.83	12697	10.20	11906	10.29	12064	11.03	13802
a518k	12.86	42310	12.14	43356	11.82	41349	12.05	42031	12.94	45163

Force Definition

a = 2 is a good choice

$$f(c_i, c_j) = \begin{cases} (x_i - x_j)^2 + (y_i - y_j)^2 & normal\_link \\ ((x_i - x_j)^2 + (y_i - y_j)^2)(Delay(path_{i \to j}))^a & pseudo\_link \end{cases}$$

Is Pseudo Link indeed Effective? How much is Needed?

Benchmark	Add no Link		Add link for those longer than 90% of the longest		Add link for those longer than 50% of the longest		Add link for all paths	
	Delay	CPU	Delay	CPU	Delay	CPU	Delay	CPU
matrix	8.53	7	8.82	7	8.28	8	7.81	9
sdram_rdr	3.15	46	3.01	45	2.79	46	2.64	56
32bMAC	4.79	87	4.43	98	4.26	125	4.06	150
VP2	13.63	163	12.69	179	12.11	208	11.76	229
64bMAC	5.04	272	4.83	335	4.31	403	4.27	486
a259k	12.65	5834	11.43	6310	10.61	9525	10.20	11906
a518k	14.59	16953	13.10	21088	12.41	32666	11.82	41349
total		23362		28062		42981		54185

Delay in (ns) and CPU in (sec)

Run-Time/Quality Tradeoff

Quality is Essentially Independent of Initial Placement

Benchmark	Chip Center		Random		From CT	
	Delay(ns)	CPU(s)	Delay(ns)	CPU(s)	Delay(ns)	CPU(s)
matrix	7.81	9	7.79	11	7.82	8
sdram_rdr	2.64	56	2.66	53	2.64	51
32bMAC	4.06	150	4.13	192	4.03	139
VP2	11.76	229	11.75	211	11.84	207
64bMAC	4.27	486	4.30	607	4.25	430
a259k	10.20	11906	10.22	12389	10.16	10834
a518k	11.82	41349	11.85	43160	11.88	38455
total		54185		56623		50124

Conclusions and Future Work

- Force-directed performance-driven placement
- Model path delay constraint directly with pseudo link
- Integrated into an industrial flow
- Significant timing improvement
- Computationally efficient
- Quality is independent of initial placement; good initial placement helps a little bit in run time
- Future work
  - ECO capability (buffer insertion)
  - Handle macro and preplaced blocks