Fixed-outline Floorplanning Through Better Local Search

Saurabh Adya
Igor Markov
(University of Michigan)

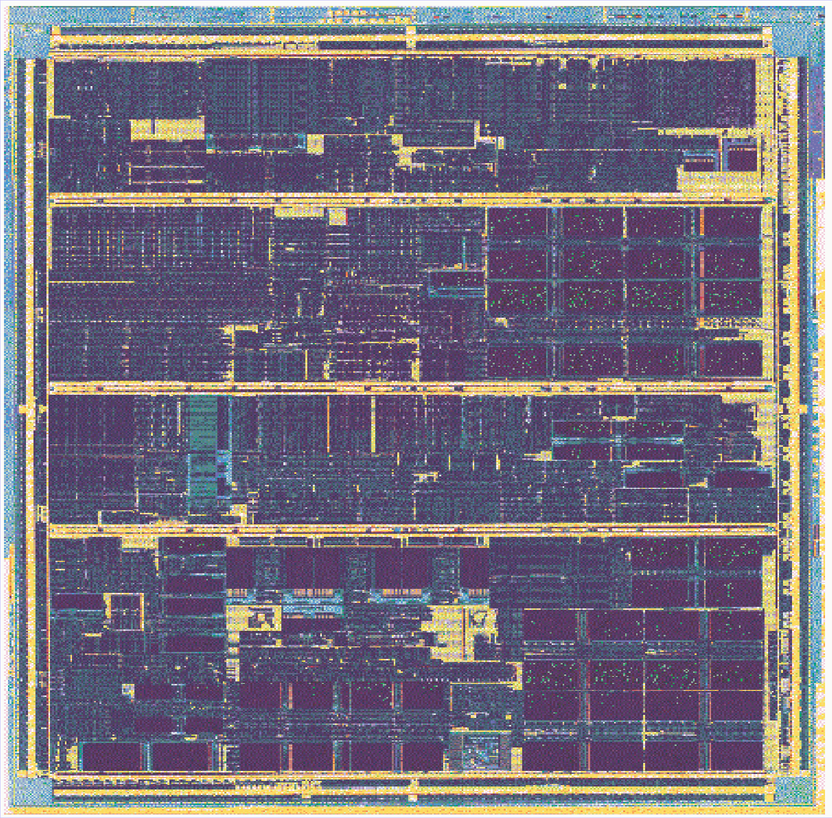
Outline

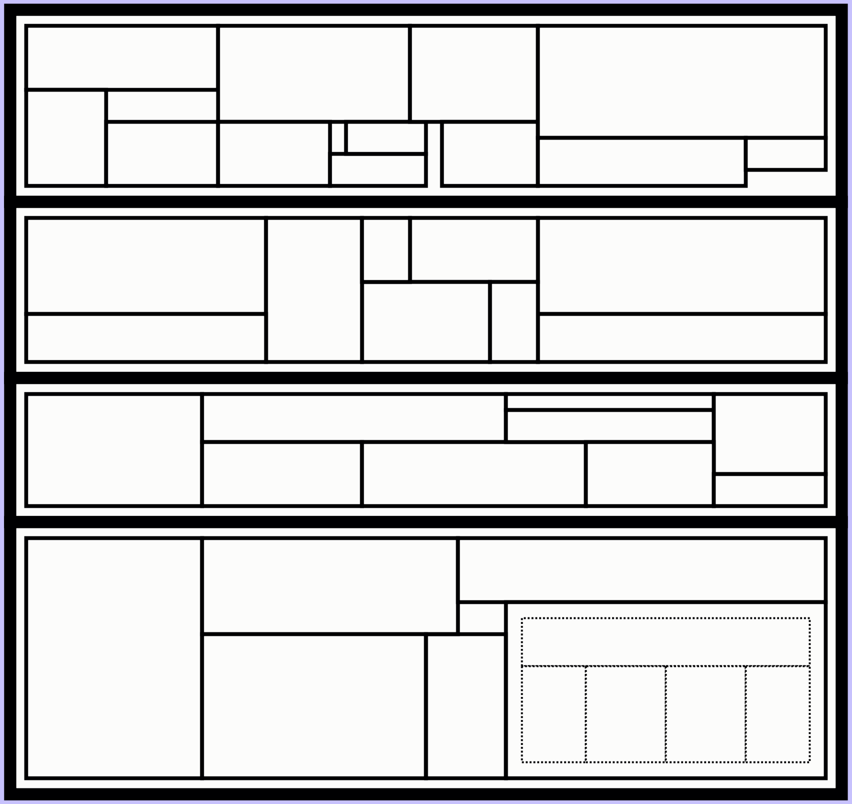
- Classical floorplanning vs. Modern hierarchical floorplanning
- Motivation for new methods
- Better local search
  - Slack-based moves
  - Fixed-outline floorplans using slack-based moves
- Results
- Conclusion

Classical Floorplanning

- Objectives: Min Area & Wirelength
- No fixed-outline constraints
- Assumes a variable-die layout and "flat" floorplanning instance
- Representations: O-Tree, B*-Tree, Sequence Pair, TCG, CBL, ECBL etc

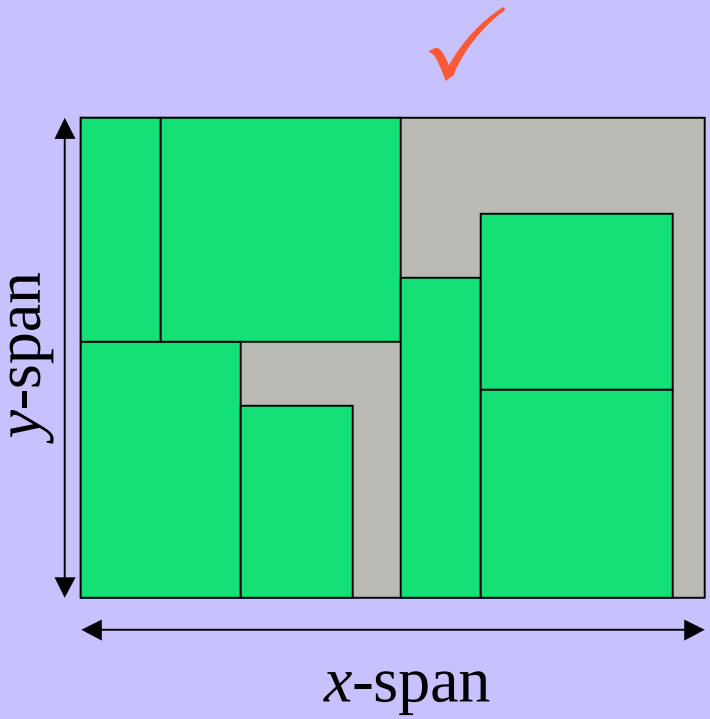
Need For Hierarchical Floorplanning

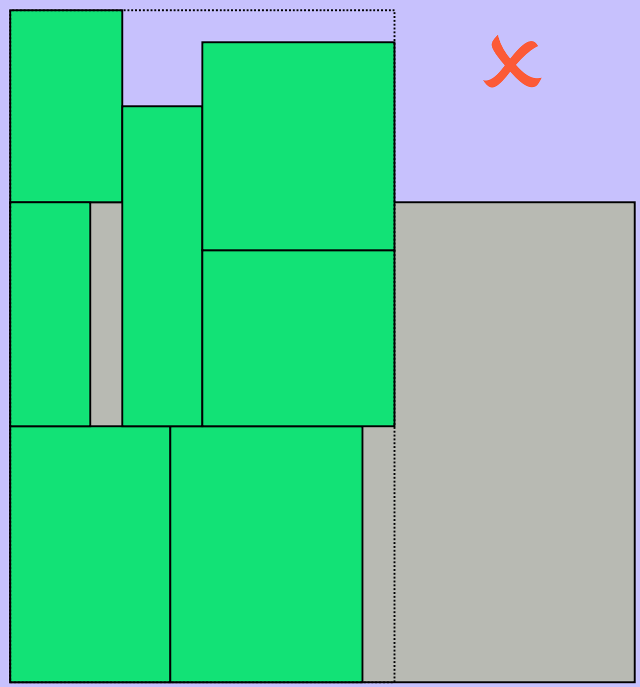




Fixed Outline Constraints

- Not a minimization problem
- Rather a constraint satisfaction problem





Modern Hierarchical Floorplanning

- Target designs: large ASICs, SOCs
  - E.g. multi-million gate designs like graphics cards
  - Hierarchical floorplanning $\Rightarrow$ Fixed outline constraints
  - Fixed-die layout [Kahng, ISPD '00]
- Classical floorplanning still important at top level
- Fixed-outline required at lower levels
- Stronger neighborhood structures helpful
  - Better guide search

Motivation For New Methods

- Existing floorplanners use simulated annealing framework
- Floorplanning with constraints
  - Pre-placed constraints [Young et al, ICCAD '98]
  - Range constraints [Young et al, ISPD '99]
  - Boundary constraints [Young et al, ASPDAC '99]
  - Abutment constraints [Hong et al, DAC '01]
- Additionaly need to handle fixed-outline constraints [Kahng, ISPD '00]

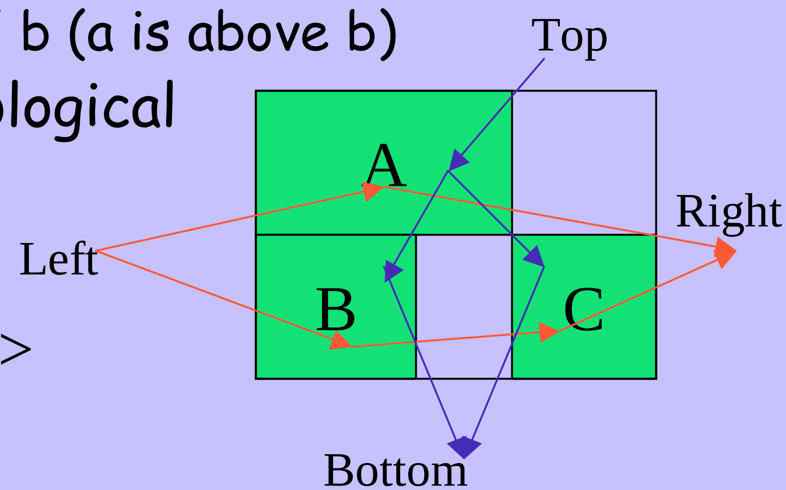
Sequence Pair (SP) Representation

- Proposed by Murata, Fujiyoshi, Nakatake, Kajitani [TCAD '97]
- Two permutations of N blocks capture the geometric relation between each pair of blocks

  $(\langle \ldots a \ldots b \ldots \rangle, \langle \ldots a \ldots b \ldots \rangle) \Rightarrow$ a is to the left of b

  $(\langle \ldots a \ldots b \ldots \rangle, \langle \ldots b \ldots a \ldots \rangle) \Rightarrow$ a is above b
- Horizontal (Vertical) constraint graphs
  - Edge $a \rightarrow b$ iff a is to the left of b (a is above b)
- Compute block locations by topological traversal
  - $O(n^2)$ complexity
  - Slow

<ABC, BCA>



SP Evaluation Via Longest Common Subsequence

- Proposed by Tang, Tian, Wong [DATE '00]
- Longest Common weighted Subsequence (LCS)
  - Weights are copied from block widths (heights)
  - Length of LCS equals *x*-span (*y*-span)
- Floorplan evaluation reduces to LCS computation
- LCS computation algorithms
  - $O(n^2)$ complexity : fast for small floorplans
    - Very little work in inner loop
  - *O(n log n)* complexity
  - *O(n log log n) complexity* [ASPDAC '01]

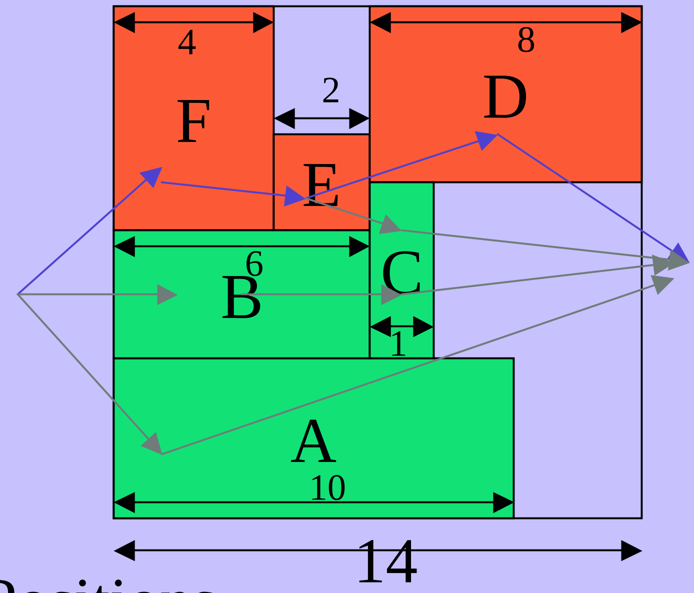
Variables in LCS computation

- Sequence Pair <X, Y>
- MatchX : position of blocks in X seq
- MatchY : position of blocks in Y seq
- Length : maintains the LCS value
- Position : records the position of each block

SP Evaluation Using LCS

```
LCS_ORIG(X,Y,Position,weights)  /*Position[1...N] records block positions*/
  for i = 1 to N   /*Initialize Match Array match*/
  begin
    match[X[i]].x = i;
    match[Y[i]].y = i;
  end
  for i = 1 to N   /*Initialize Length Array Length with 0*/
    Length[i] = 0;
  for i = 1 to N
  begin
    b = X[i];
    p = match[b].y;
    Position[b] = Length[p];
    t = Position[b] + weights(b);
    for j = p to N
      if (t > Length[j]) then Length[j] = t;
      else break;
  end
return Length[N];
```

Evaluation of *x*-locations by LCS computation



Sequence Pair

<X,Y> = <FEDBCA, ABFECD>

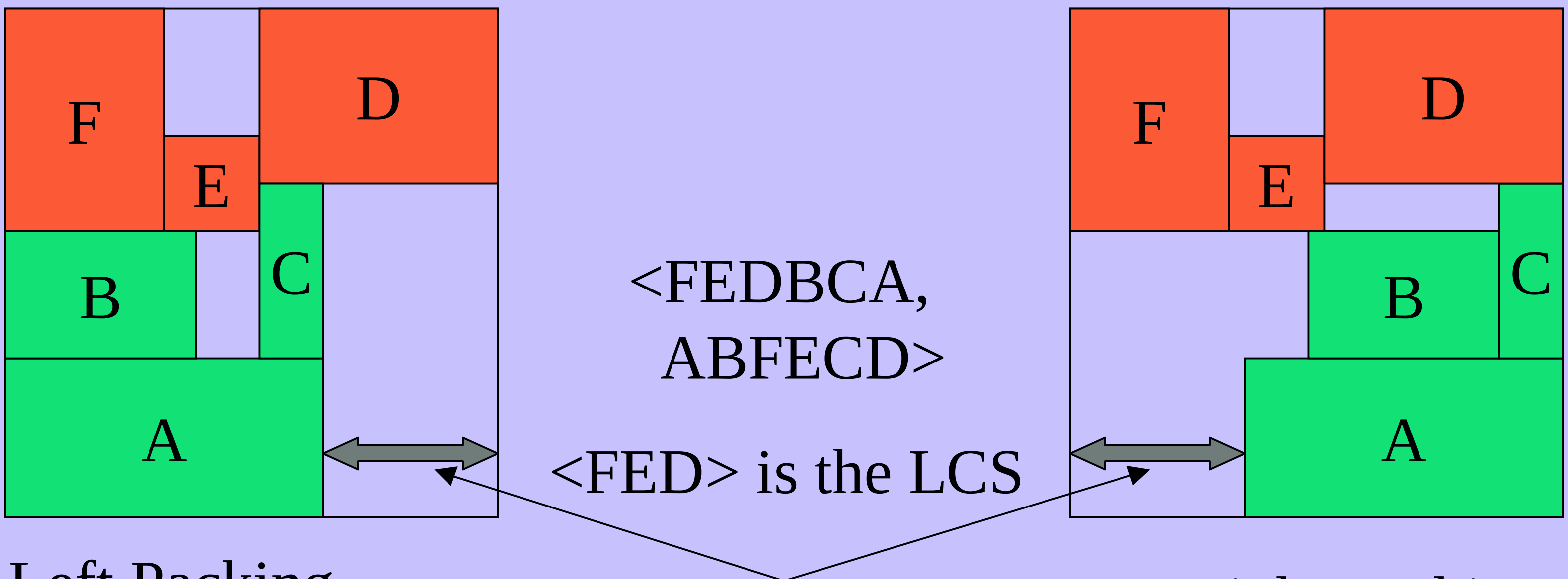
Length Array

1	2	3	4	5	6
0	0	0	0	0	0
0	0	4	4	4	4
0	0	4	6	6	6
0	0	4	6	6	14
0	6	6	6	6	14
0	6	6	6	7	14
10	10	10	10	10	14

Block Positions

A	B	C	D	E	F
					0
				4	0
			6	4	0
	0		6	4	0
	0	6	6	4	0
0	0	6	6	4	0

Floorplan “Slack”



Left Packing

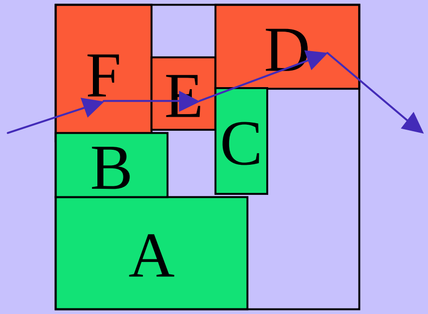
Right Packing

x-slack for block A = $x(A_{right}) - x(A_{left})$

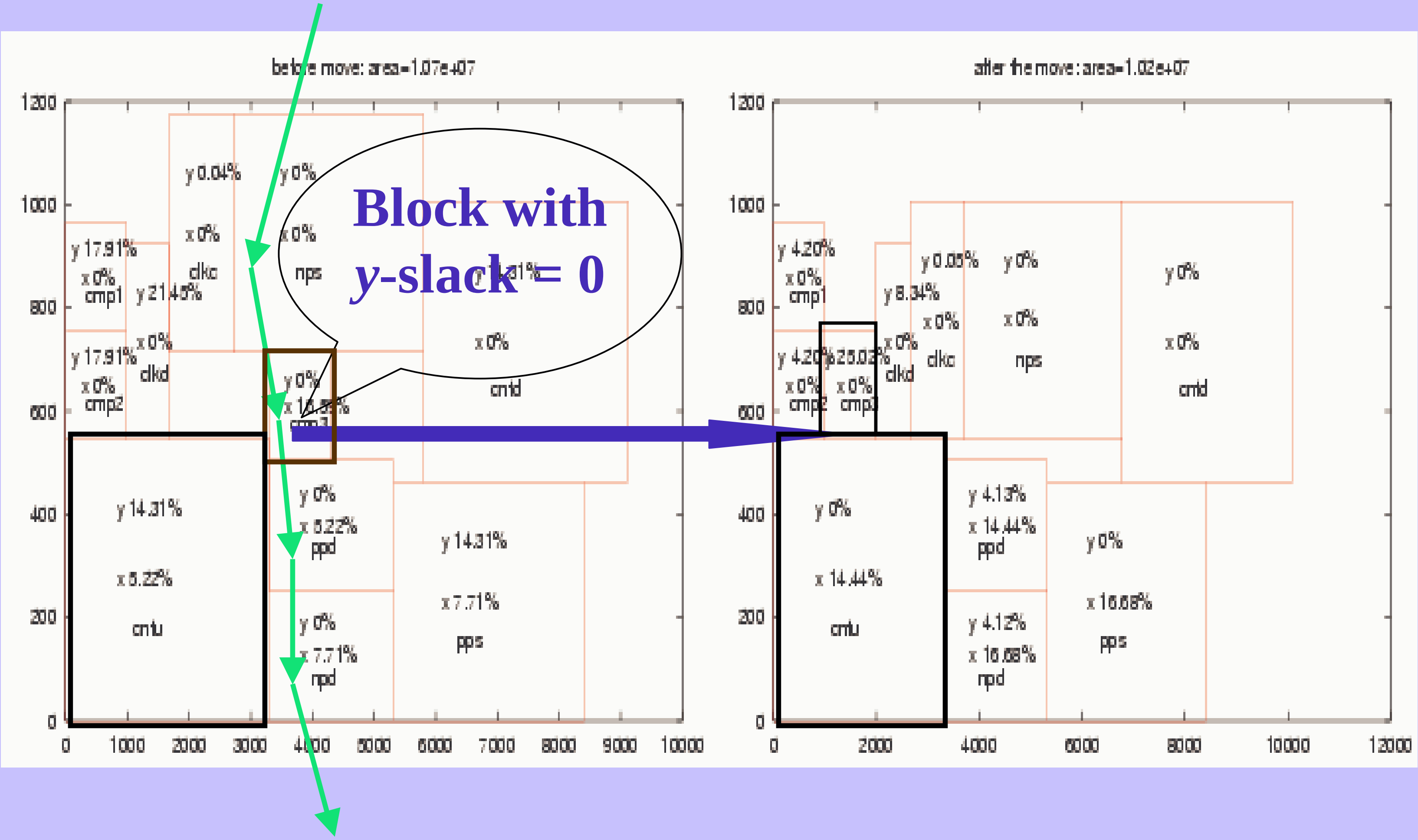
x-Slack Computation

Better Local Search

- Floorplan "Slack" Computation
  - Applicable to any evaluation method which packs left to right and right to left
  - *x*-slack for a block represents the *x*-whitespace
  - Analogy with Static Timing Analysis
- Blocks with zero slack lie on "critical paths"
- Observation : Only moves of blocks with zero *x*-slack (*y*-slack) can reduce the floorplan span
- Slack-based moves helpful for :
  - Satisfying a fixed-outline (this work)
  - Handling soft blocks (see source code)



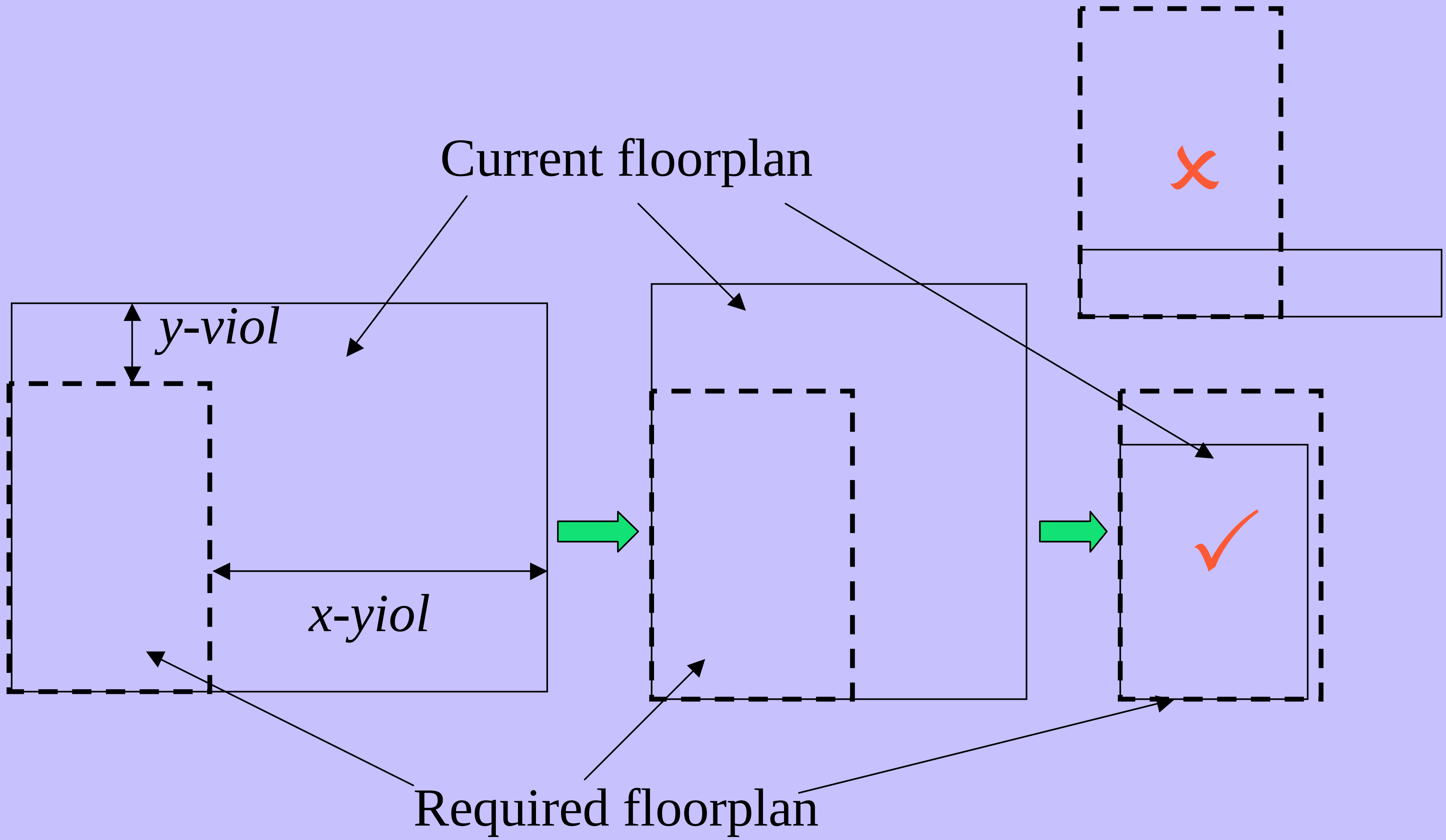
Example: A Slack-based Move



Fixed-outline Floorplanning

- Find target aspect ratio of fixed outline
- During annealing track current aspect ratio
- Apply slack-based moves if current aspect ratio too far from target aspect ratio
  - *(targetAR – currAR)/targetAR > 0.005*
- A sample slack-based move
  - Find block A with zero slack in one dimension
  - Find block B with large slack in other dimension
  - Place A close to B
- Different slack-based moves are possible

Fixed-outline Floorplanning (contd.)



Objective Functions

- Main framework : Simulated Annealing (SA)
- Need an objective
  - Classical min-area objective appears inadequate
- Choose one of 3 objective functions
  - *Min area*
  - *Min (excessive length+ excessive width)*
  - *Min (max of excessive(length,width))*
- Empirical winner (shown later)
  - *Min (excessive length+ excessive width)*

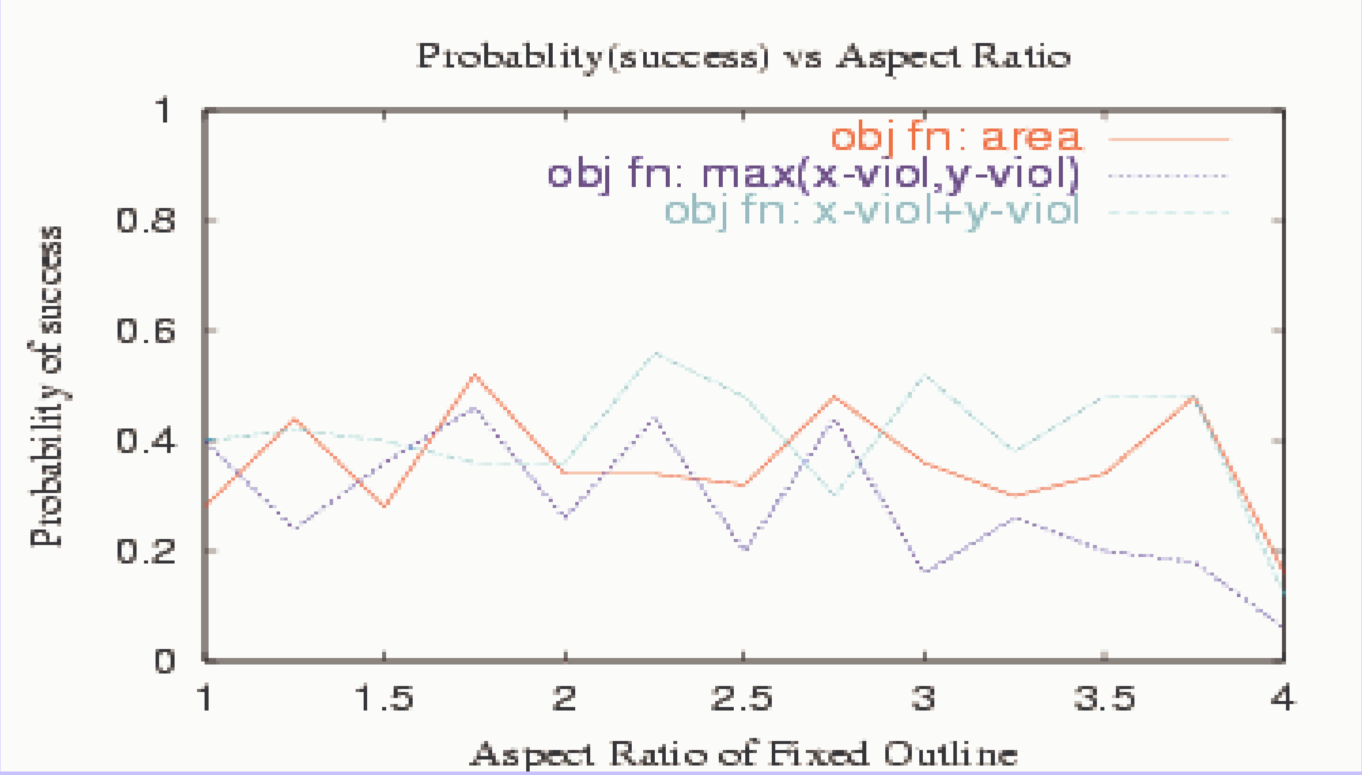
Implementation

- Implementation is in C++
  - Compiled with g++ 2.95.2 –O3
- Hardware
  - 800 MHz PC/Intel system
- Min area results competitive with latest reported

Circuit	Min/Avg Area (mm^2)	Min/Avg deadspace (%)	Avg Time (sec)
Apte	47.07/48.14	1.08/3.28	4
Xerox	19.83/20.73	2.42/6.65	3
Hp	9.14/9.49	3.39/6.95	4
Ami33	1.19/1.23	2.85/6.01	9
Ami49	37.27/38.01	4.91/6.76	16

Results

- Without slack-based moves
  - Not able to satisfy fixed-outline constraints
- With slack-based moves
  - Fixed-outline success rates for ami49



Conclusion

- Fixed-outline floorplanning harder than min area floorplanning
- Known algorithms cannot address fixed-outline constraints
- Slack-based moves help
- New objectives help
- Source code available at:
  - http://www.vlsicad.eecs.umich.edu/BK/parquet
- Ongoing work :
  - Fixed-outline floorplanning with wirelength minimization