ASPDAC 2013

L-Shape Based Layout Fracturing for E-Beam Lithography

Bei Yu, Jhih-Rong Gao, and **David Z. Pan**
Dept. of Electrical & Computer Engineering
University of Texas at Austin

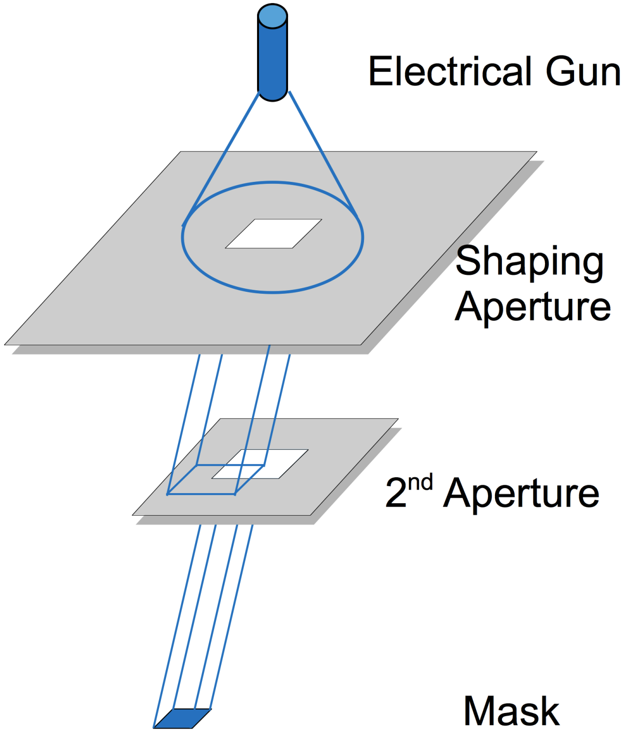
Supported in part by NSF and NSFC

Outline

- Introduction
- Problem Formulation
- Algorithms
  - Rectangular Merging (RM) Algorithm
  - Direct L-Shape Fracturing (DLF) Algorithm
- Experimental Results
- Conclusion

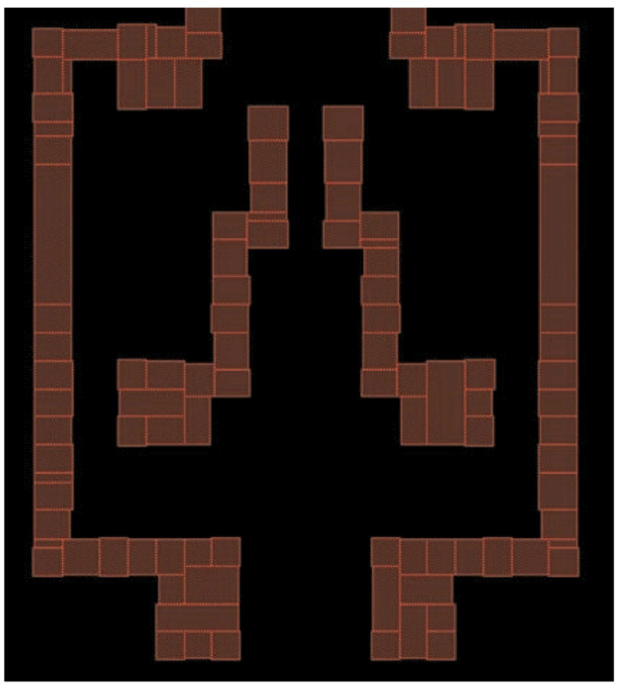
EBL

- E-Beam lithography (EBL)
  - Widely deployed in mask manufacturing
  - Promising candidates for sub-22nm
- Conventional EBDW: variable shaped beams (VSB)



Layout Fracturing

- Fundamental step before EBL writing
- Decompose layout pattern => non-overlapping rectangles
- Shot number dramatically increases for sub-22nm
  - More complicated OPC



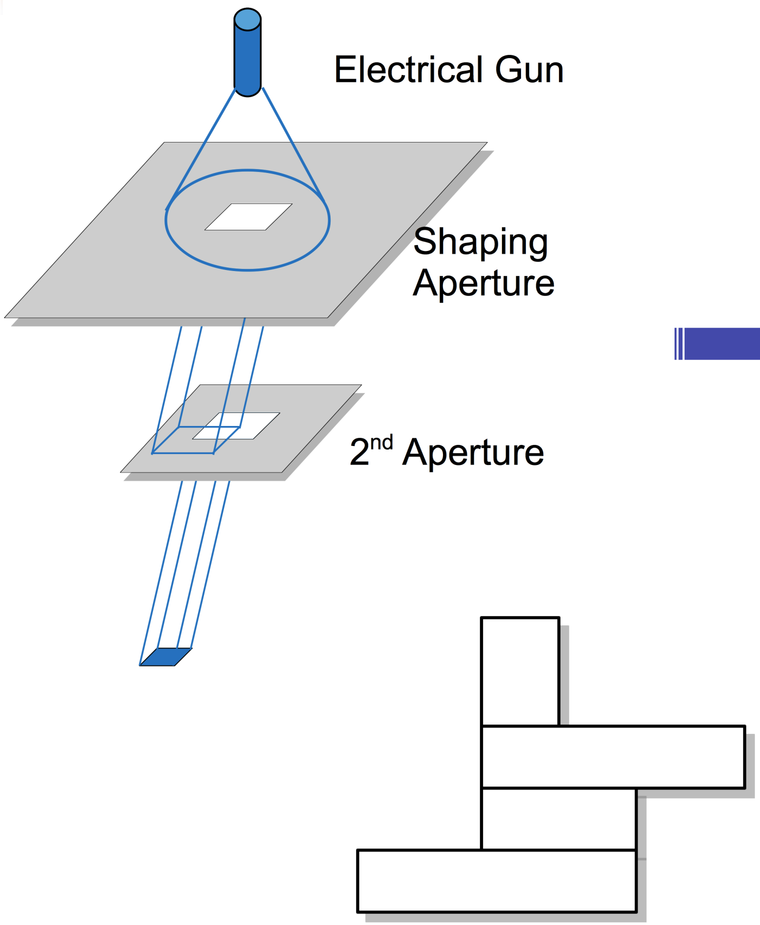
E-beam Shot Count Estimates by Node
(note: all shot count numbers = billions)

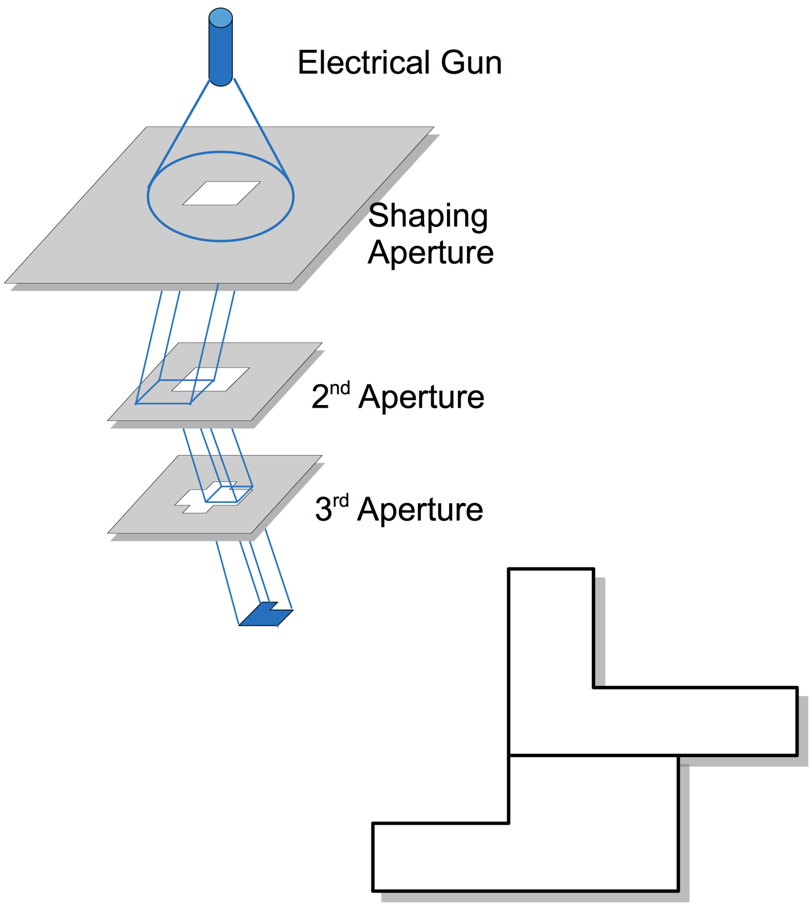
Node	M1 actual	M1 (2x scaling per node)	M1 (4x)	M1 (8x)
14		620	2480	9920
22		310	620	1240
32	155			
45	70			

Courtesy IBM

L-Shape E-beam Shot

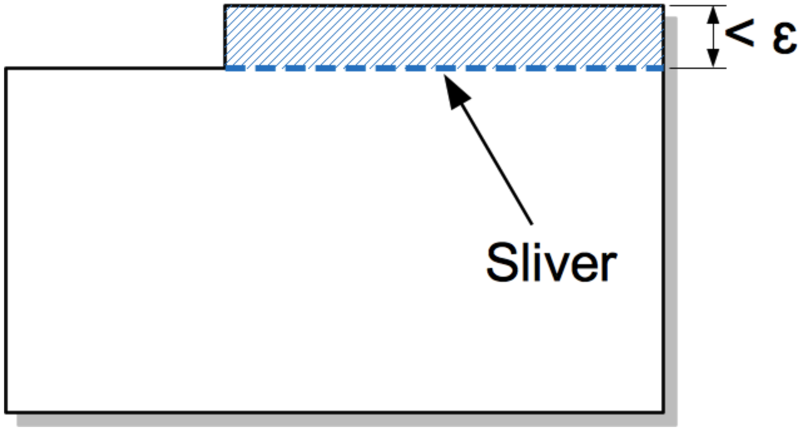
- One more aperture cf. rectangular shots
- Potentially reduce shot number by up to 50%





Previous Works

- Rectangular fracturing
  - ILP [Kahng, SPIE'04, SPIE'06] or heuristic methods [Dillon, SPIE'08; Ma+ SPIE'11]



- L-shape fracturing
  - Report w/o detail algorithms [Sahouria, SPIE'10]
  - In geometrical science, heuristic horizontal slicing
  - However, sliver minimization not considered

Problem Formulation

- Input:
  - Layout (a set of polygons)
- Output:
  - Fracture the layout into a set of non-overlapping L-shapes and rectangles
- Objective:
  - Minimize the shot count (L shapes or rectangles)
  - Minimize the silver length of fractured shots

Outline

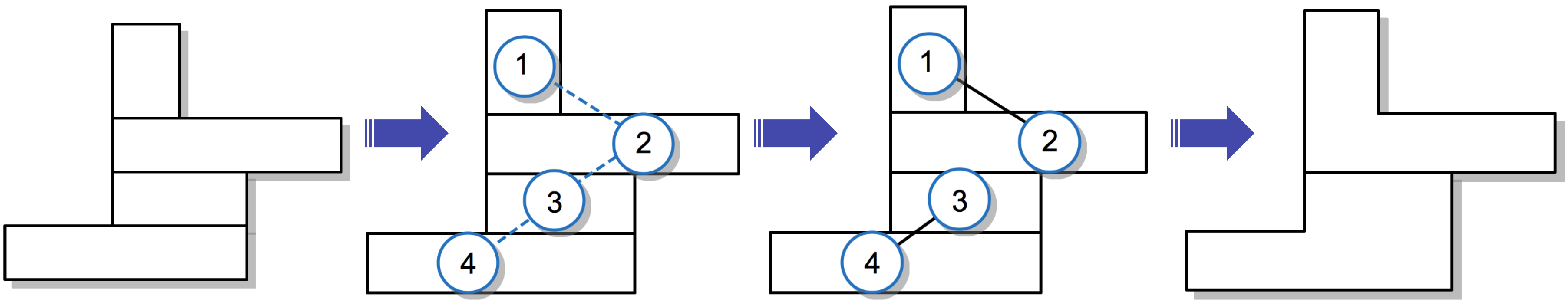
- Introduction
- Problem Formulation
- Algorithms
  - Rectangular Merging (RM) Algorithm
  - Direct L-Shape Fracturing (DLF) Algorithm
- Experimental Results
- Conclusion

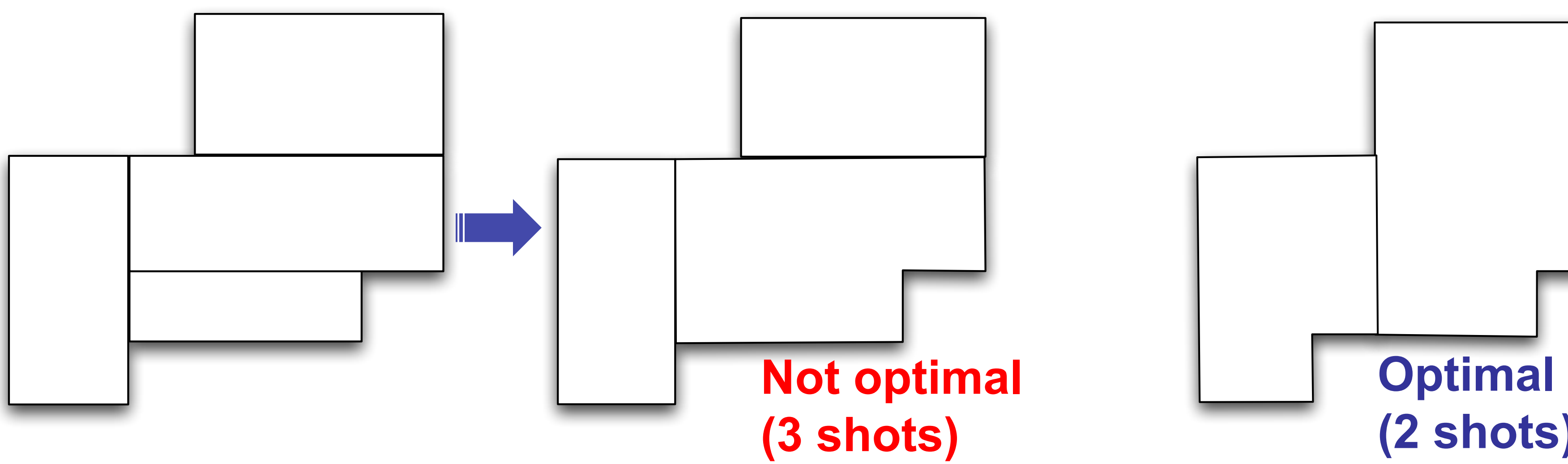
Two Approaches

- Rectangular Merging (RM) Algorithm
  - Re-use previous rectangular fracturing results
  - Merge rectangles into L-shapes
- Direct L-Shape Fracturing (DLF) Algorithm
  - Direct L-Shape Generation
  - Avoid redundant operations
  - Nice properties to reduce problem size/complexity

Rectangular Merging (RM)

- Given input rectangles (through conventional VSB fracturing)
- Construct graph to represent the relationships
- Edge selection through maximum matching O(nmlogn)

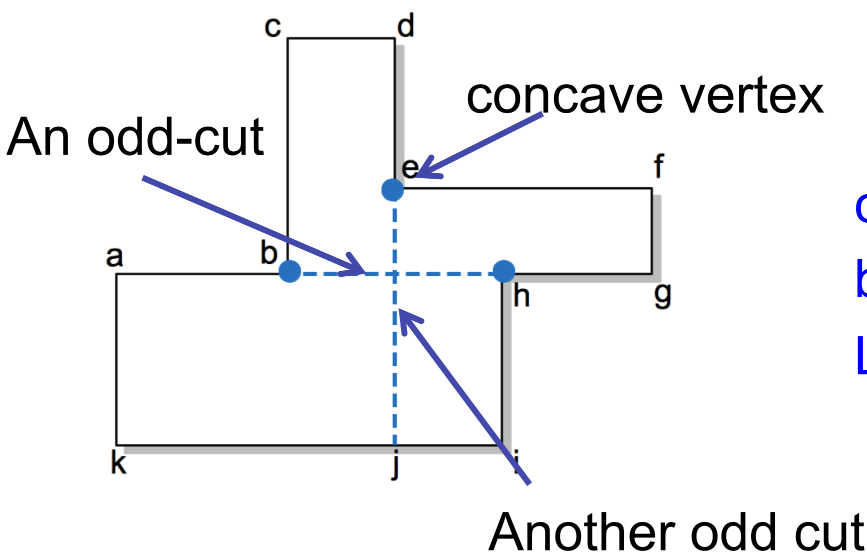




Direct L-Shape Fracturing

- ***Concave vertex***: with internal angle is 270°
- ***Cut***: a horizontal or vertical line segment where at least one of the two endpoints is a concave vertex
- ***Odd-Cut***: a cut that has **odd** number of concave vertices on one or both sides of the cut

Lemma 1: A polygon with *c* concave vertices can be decomposed into L-shapes with upper bound *Nup* = $\lfloor c/2 \rfloor + 1$



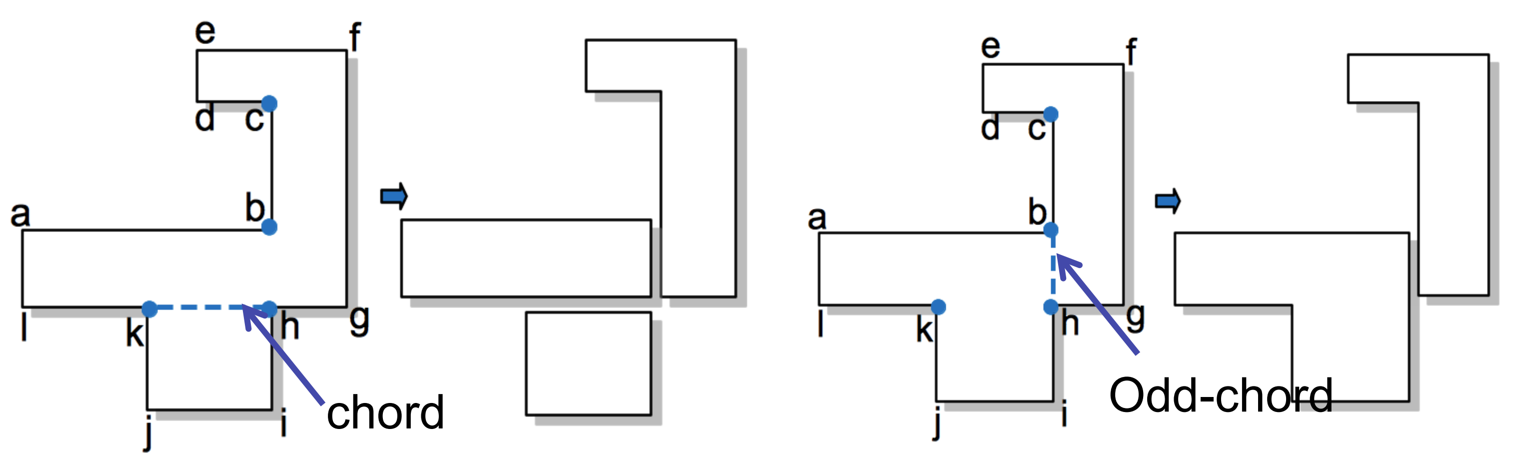
c = 3 ➔ this polygon can be decomposed into two L-shapes

Direct L-Shape Fracturing

- ***Chord***: A special cut whose two endpoints are both concave
- ***Odd-Chord***: a chord that is an odd-cut

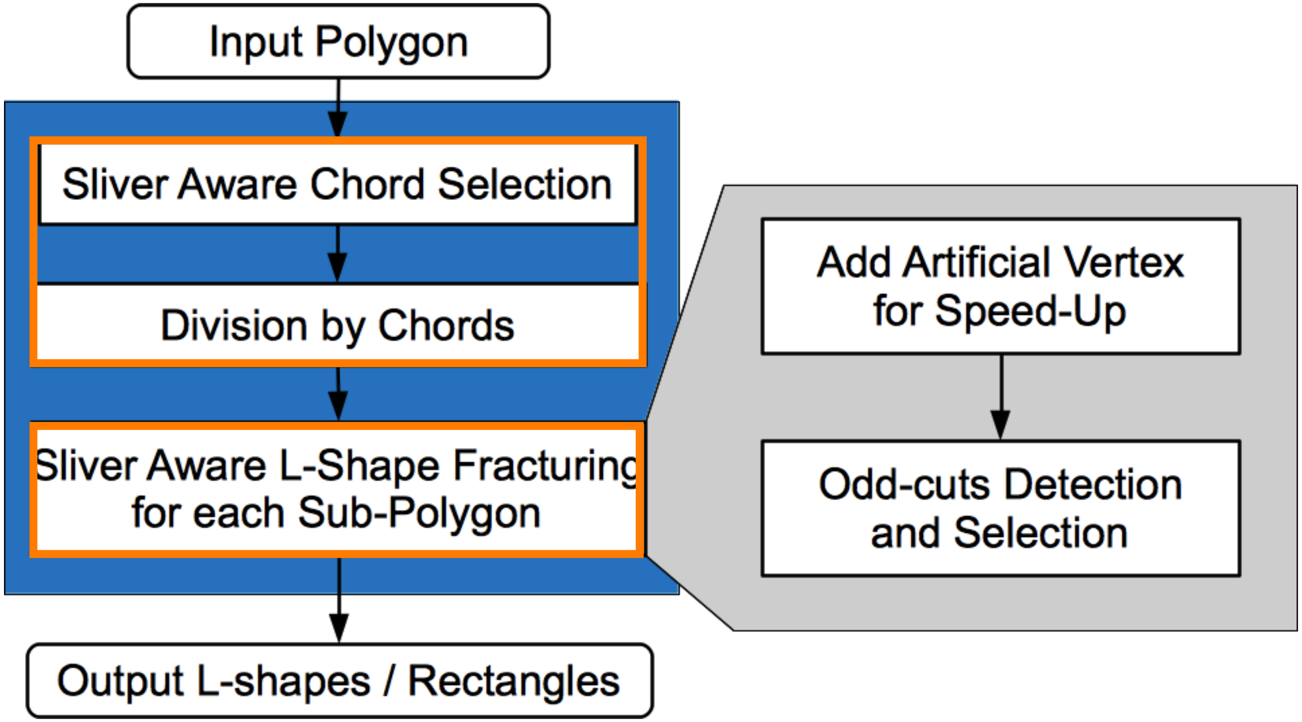
Lemma 2: Dividing a polygon through a chord will not increase *Nup*

Lemma 3: Dividing a polygon with even number of concave vertices through an odd-chord can reduce *Nup* by 1



Direct L-Shape Fracturing Algorithm

- Overall Flow



- Step 1: chord selection and division => independent sub-polygons
- Step 2: odd-cut detection and L-shape fracturing

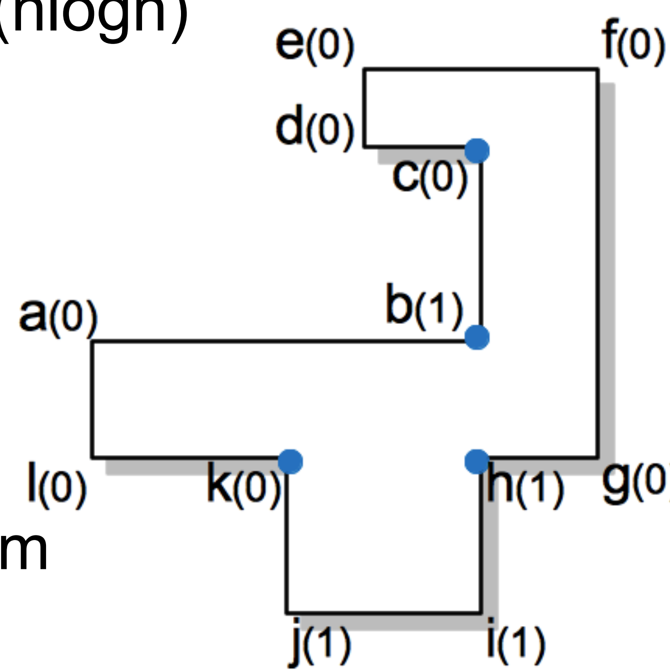
Odd-Chord Detection and Selection

Odd-Chord Detection

- Check whether odd-chord, from O(n) to O(1)
  - Each vertex is associated with parity value p

Theorem 1: In a even polygon, chord *ab* is odd iff $p_a = p_b$

- All odd-chords can be detected in O(nlogn)



Chord Selection

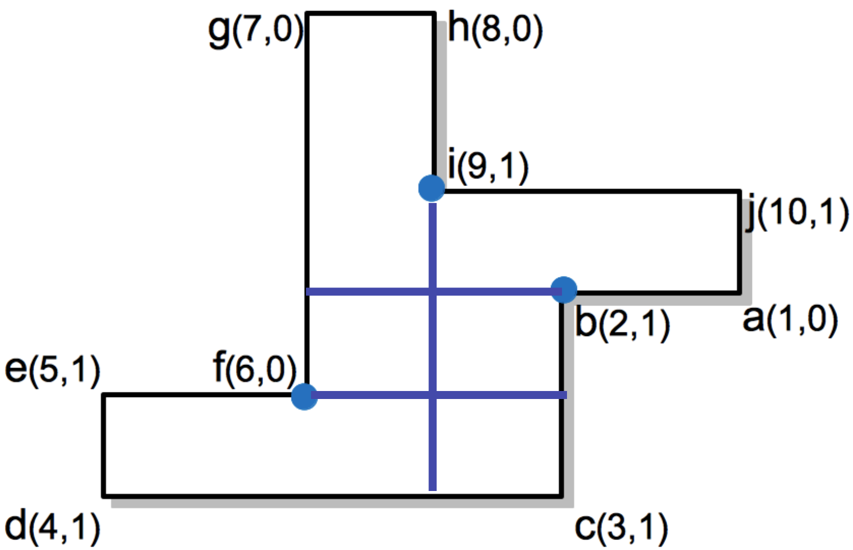
- Prefer odd-chords
  - To reduce shot count Nup
- Sliver minimization
- Maximum weighted matching problem

Odd-Cut Detection

- Check whether a cut is odd, in O(1)
- Each vertex is associated (order number, parity)
- ***Theorem 2***: In odd polygon, cut (a, bc) is an odd-cut iff

$$\begin{cases} p_a = p_b, & \text{if } o_a > o_b \\ p_a \neq p_b, & \text{if } o_a < o_b \end{cases}$$

- Odd-cut detection can be finished in O(nlogn)



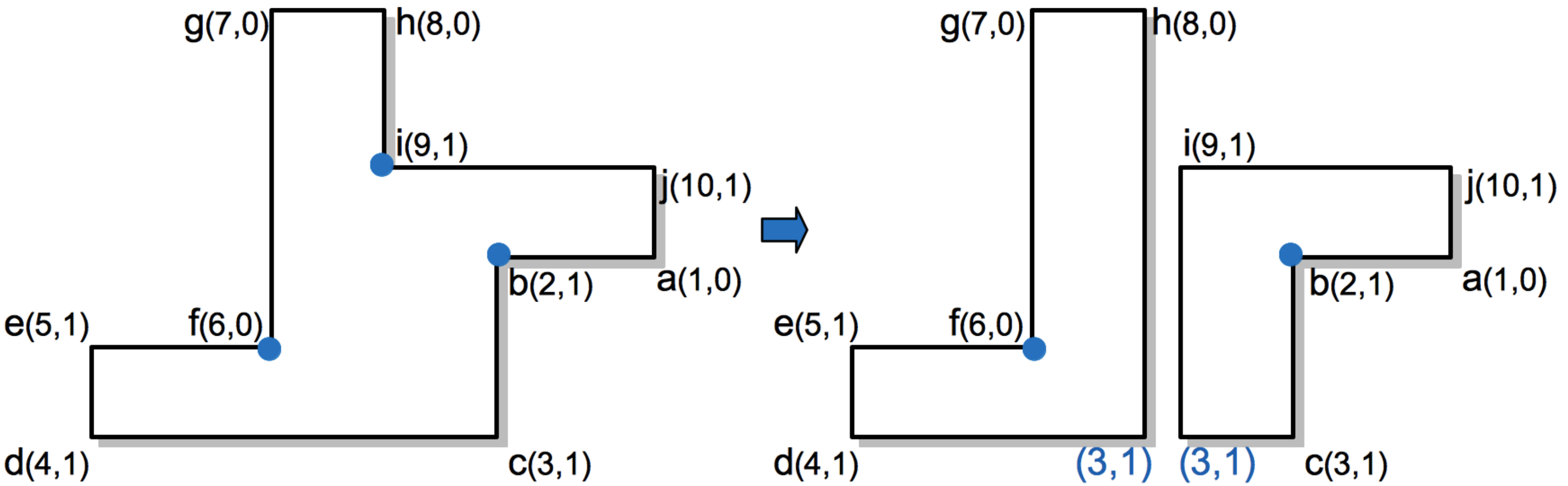
Ob (2) < Of (6), Pb (1) ≠ Pf (0), ✔

Oi (9) > Oc (3), Pi (1) = Pc (1), ✔

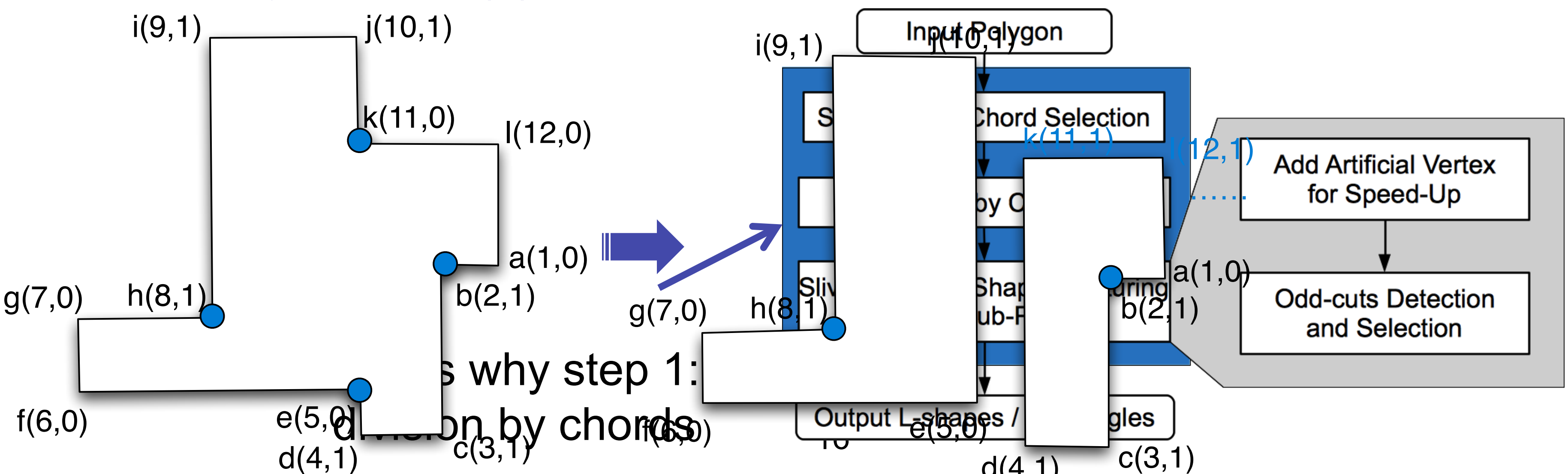
Of (6) > Ob (2), Pf (0) ≠ Pb (1), ✖

Effective Odd-Cut Info Update

- Only update one vertex and four edges, in O(1) time



Update may not be O(1) if odd-cut is a chord

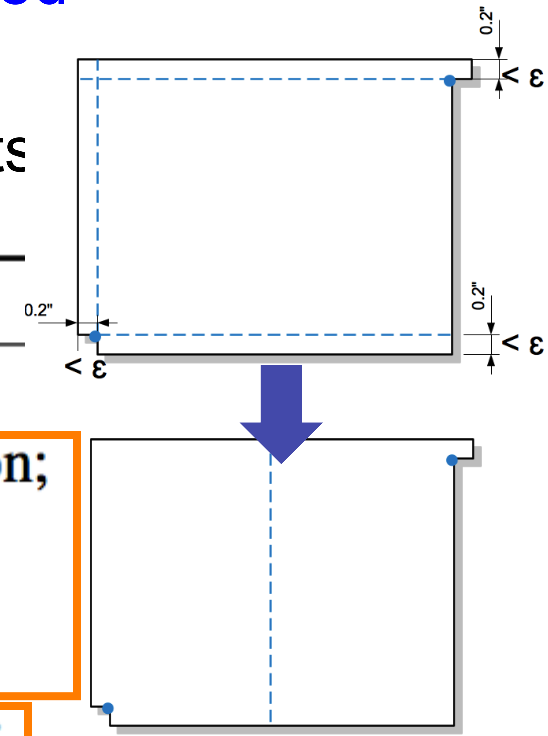


L-Shape Fracturing through Odd-Cut

- After chord selection, initial polygon is divided into a set of sub-polygons
- Fracture each sub-polygon through odd-cuts

Algorithm 1 LShapeFracturing(P)

Find all odd-cuts;
Choose cut cc considering the sliver minimization;
if Cannot find legal odd-cut **then**
  Generate an auxiliary cut cc;
end if
Cut P through cc into two polygons $P1$ and $P2$;
Update one vertex and four edges;
LShapeFracturing($P1$);
LShapeFracturing($P2$);



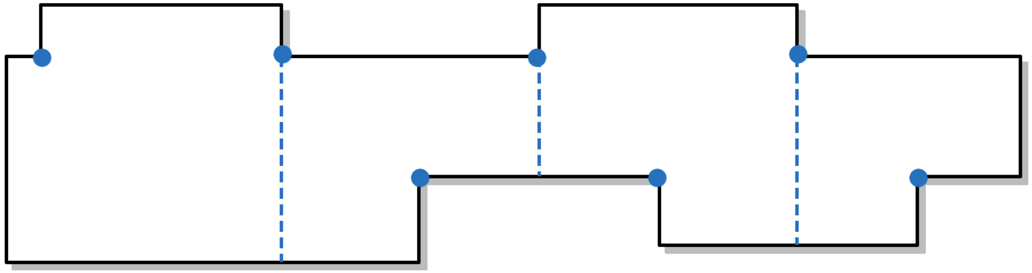
Effective Odd-cut info Update

Runtime complexity O(n^2logn)

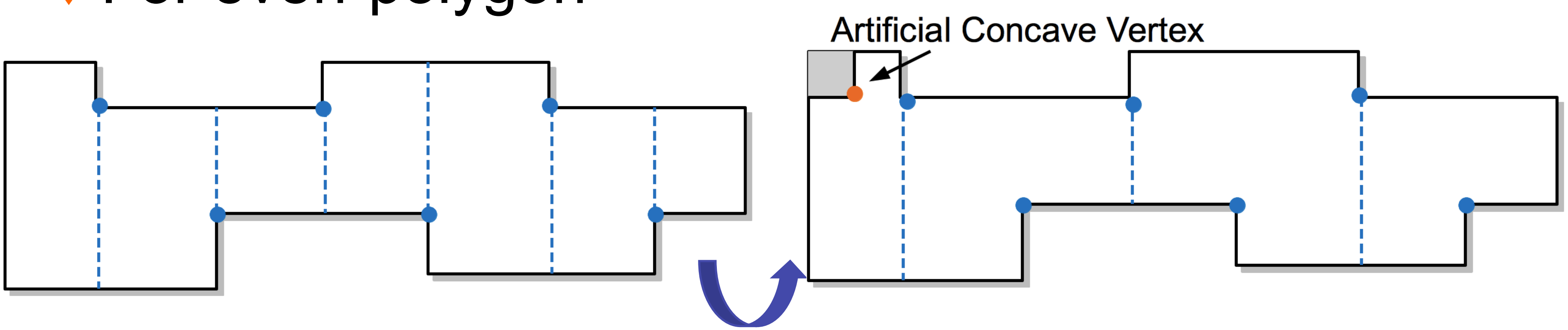
Speed-up Techniques

Select multiple independent odd cuts simultaneously

- For odd-polygon (odd # concave pts)



- For even-polygon

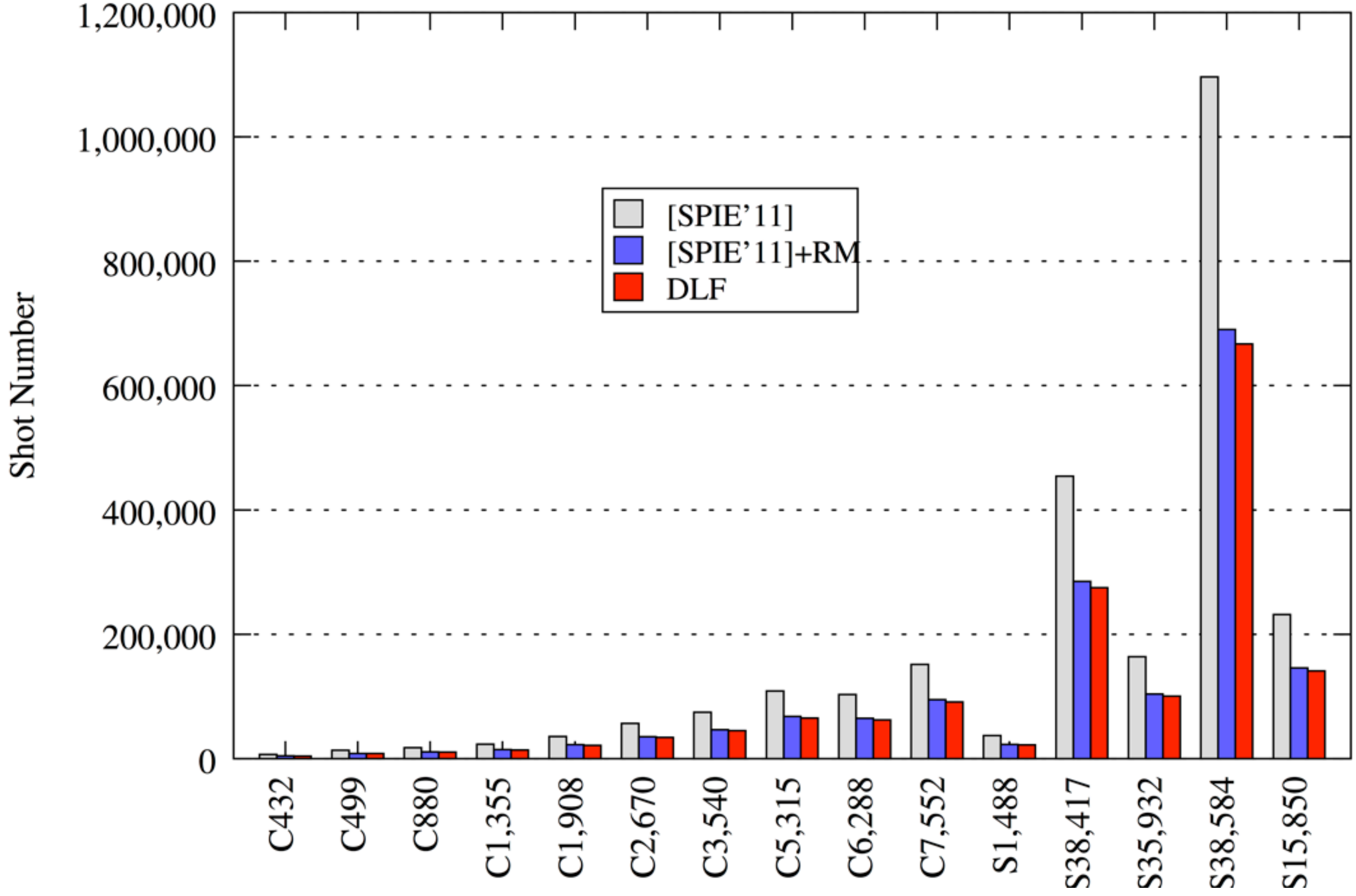


Practical runtime complexity can be reduced to O(nlogn)

Experimental Results

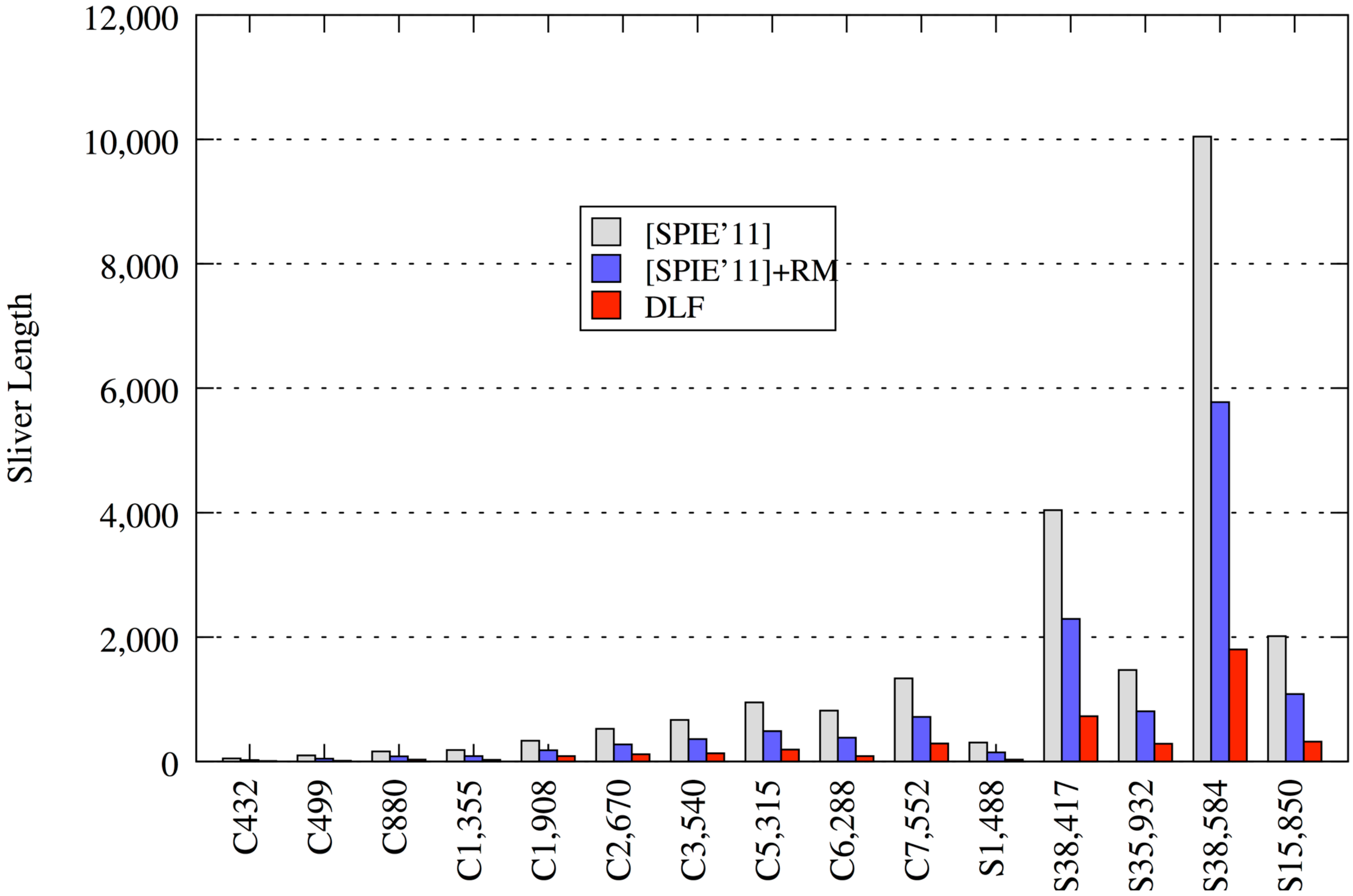
- Implement RM and DLF in C++
- 3.0GHz Linux machine with 32G RAM
- ISCAS 85&89 benchmarks
- Scaled to 28nm nodes
- Lithography simulations and OPC
- Implement rectangular fracturing in [Ma, SPIE'11]
- Sliver parameter ε = 5nm

Shot Number Comparison



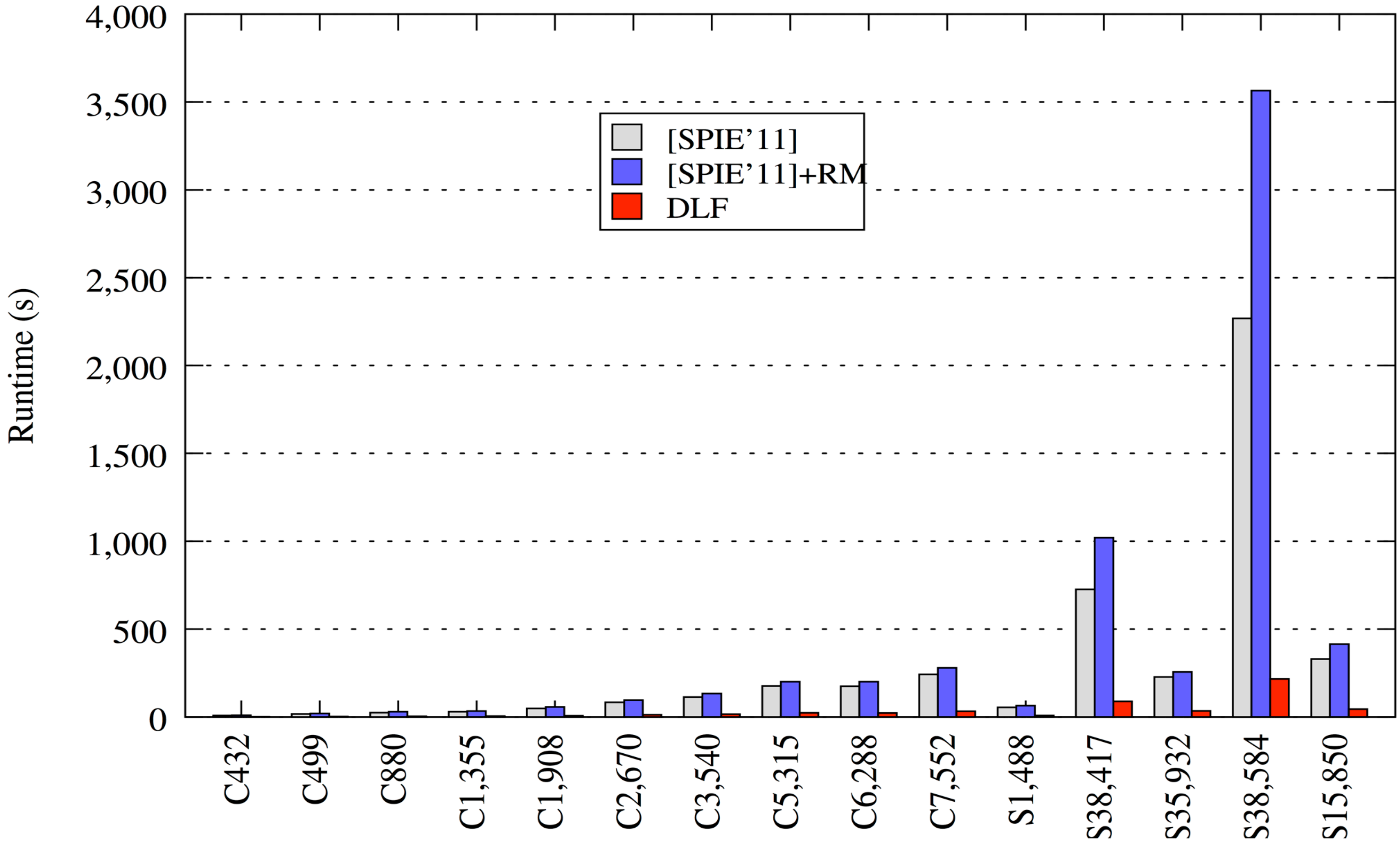
- Compared with [SPIE'11], RM reduces shot no. by 37%
- DLF: reduces 39%

Sliver Length Comparison



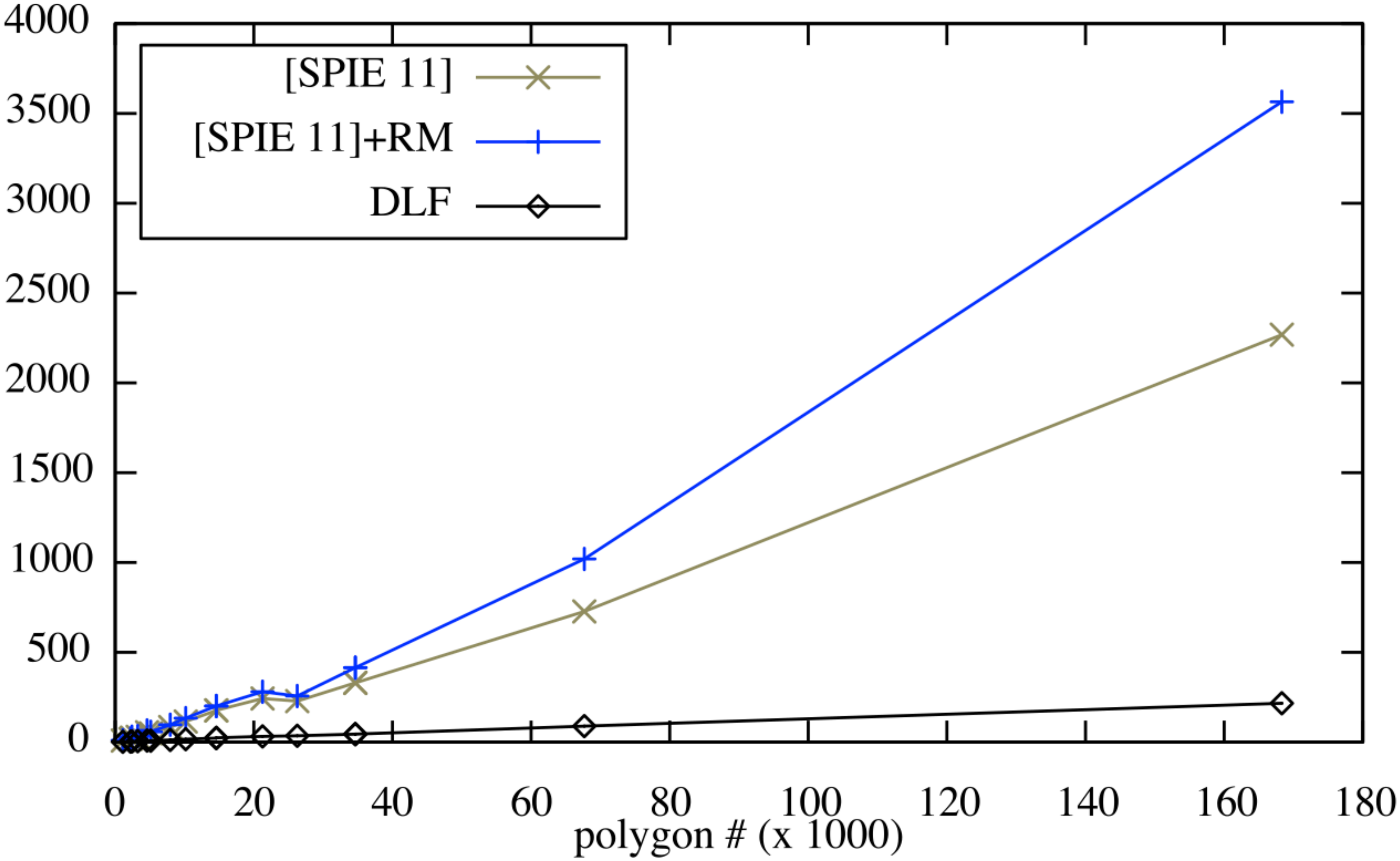
- DLF can reduce sliver by 82% cf. [SPIE'11], 67% cf. RM

Runtime Comparison



- DLF is very efficient, only 11% runtime cf. [SPIE'11]

Runtime Scalability



- DLF scales better than both [SPIE'11] and RM

Conclusion

- This work proposed the first systematic and algorithmic study in EBL L-shaped fracturing
- Two algorithms are proposed: RM and DLF
- Sliver minimization is explicitly considered
- DLF obtained the best results in all metrics
- EBL is under heavy R&D, including massive parallel EBDW.
  - More research needed on EBL-aware physical design