ASP-DAC 2015

Machine Learning and Pattern Matching in Physical Design

Bei Yu[1], David Z. Pan[1], Tetsuaki Matsunawa[2], and Xuan Zeng[3]

[1]UT Austin; [2]Toshiba; [3]Fudan University

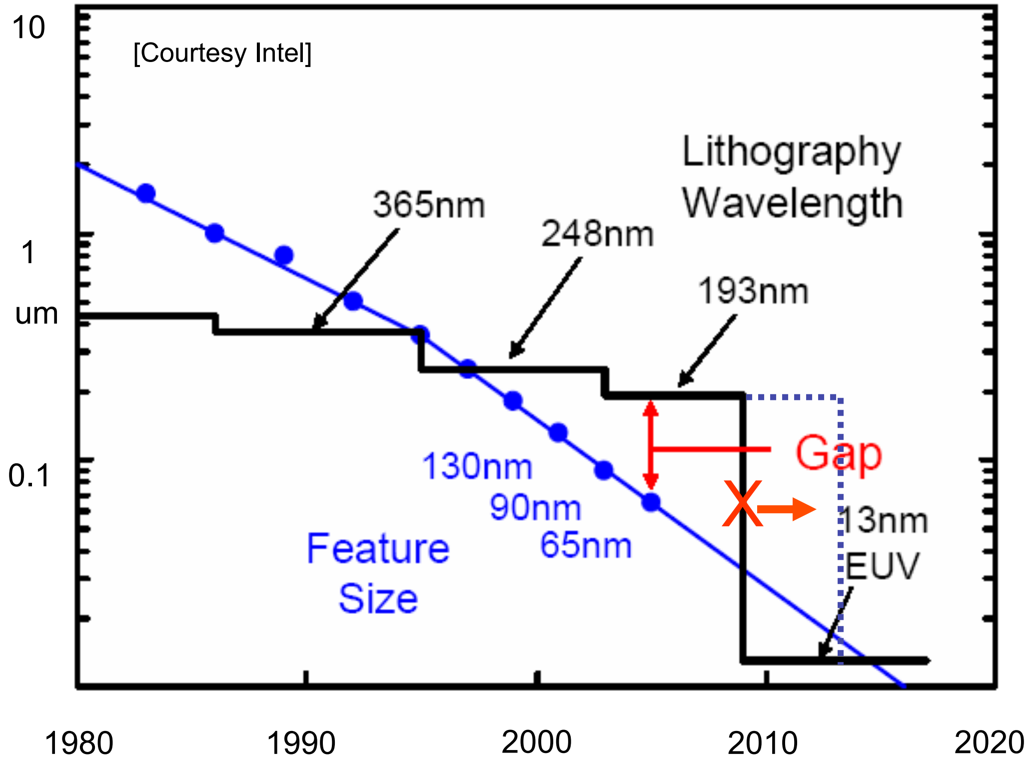
http://www.cerc.utexas.edu/utda

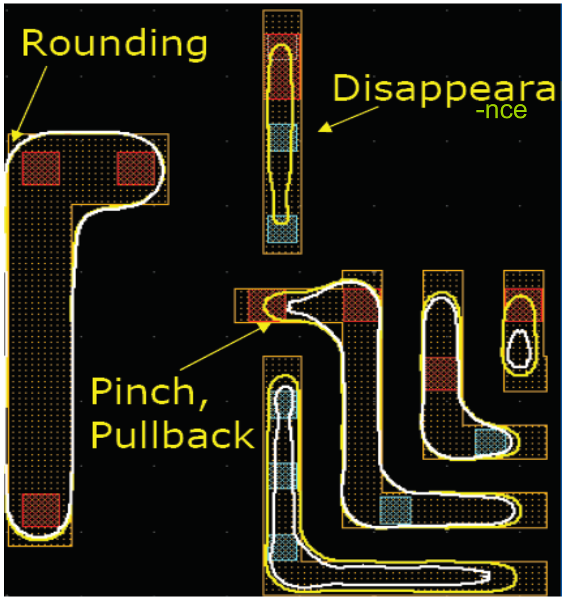
Supported in part by NSF, SRC, IBM, Toshiba, NSFC, SSTC

Outline

- Modern VLSI Challenges
- Machine Learning and Pattern Matching 101
- Applications in VLSI Design and Verification
- Some Advanced Issues
- Conclusion

VLSI Manufacturing Challenges





- The industry forced to extend 193nm lithography
  - Feature size is much smaller than the wavelength
  - Deep sub-wavelength design and manufacturing

Machine Learning 101

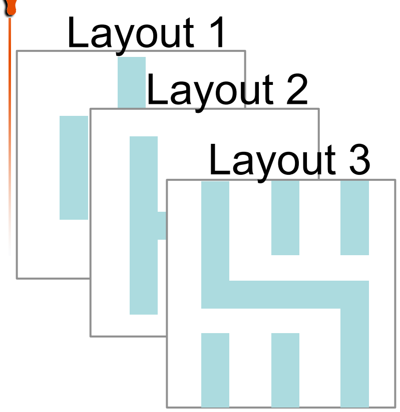
- Study of algorithms that can learn from data

$$y = f(x)$$

y : output
x : input data
f : function

- Supervised learning (labels (y) are given)
  - Classification : y is categorical data
  - Regression : y is continuous data
- Unsupervised learning (no labels are given)
  - Clustering, etc.

Machine Learning 101 (cont'd)

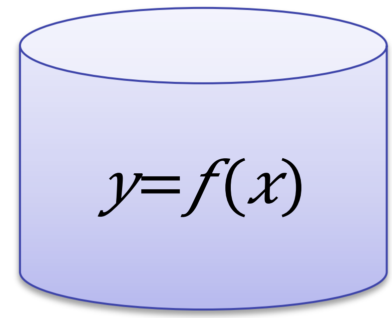


Learning data

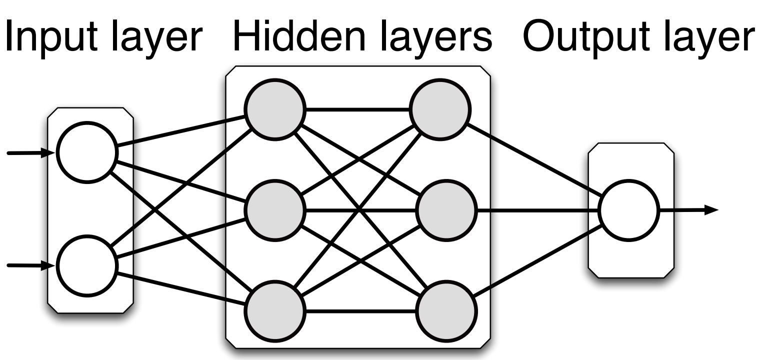
Abstracted vector data

$x_1 = (0,\ \ 1,\ \ 0,\ 1.5,\ \cdots)$
$x_2 = (2,\ 0.5,\ 0.1,\ -1,\ \cdots)$
$x_3 = (1,\ -1,\ \ 0,\ 0.3,\ \cdots)$
....

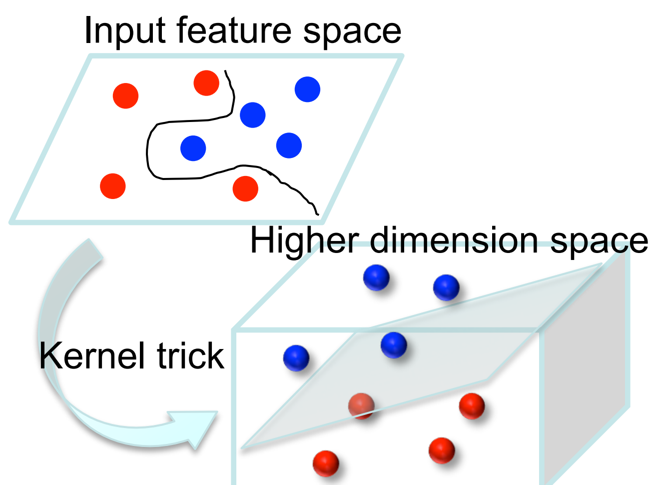
Feature extraction



Model training



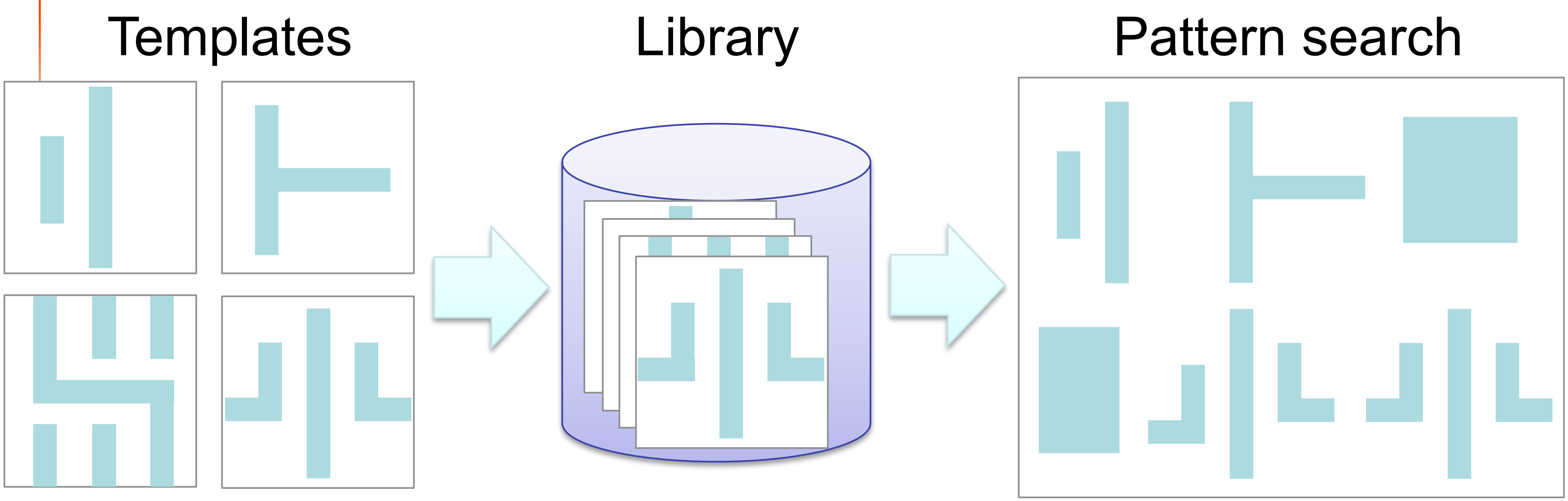
ANN
(Artificial Neural Network)



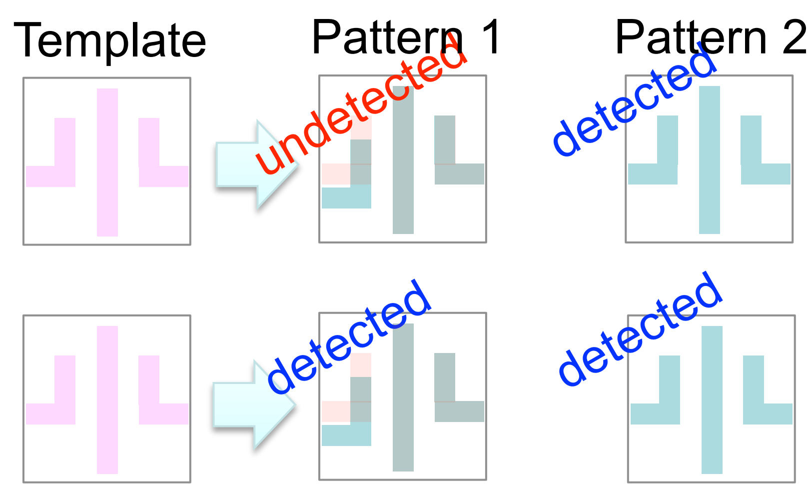
SVM
(Support Vector Machine)

Boosting …

Pattern Matching 101



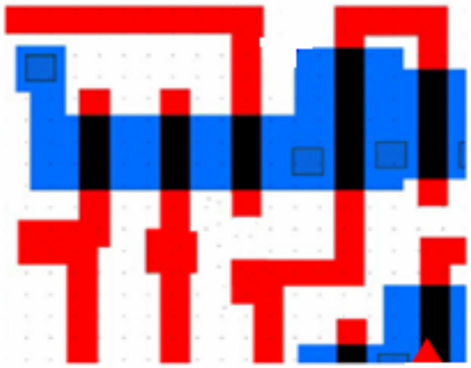
- Exact Pattern Matching
  - Detected pattern = template
- Fuzzy Pattern Matching
  - Detected pattern ≈ template



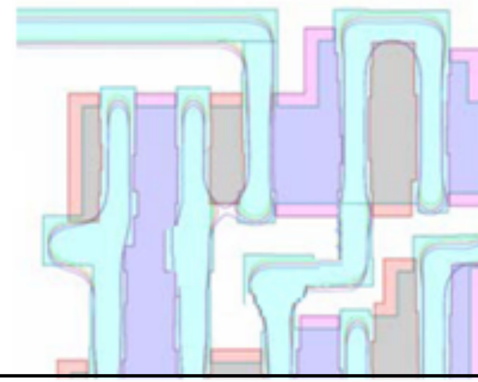
Outline

- Modern VLSI Challenges
- Machine Learning and Pattern Matching 101
- Applications in VLSI Design and Verification
  - Lithography Hotspot Detection
  - Lithography Friendly Routing
  - Datapath Extraction and Placement
- Some Advanced Issues
- Conclusion

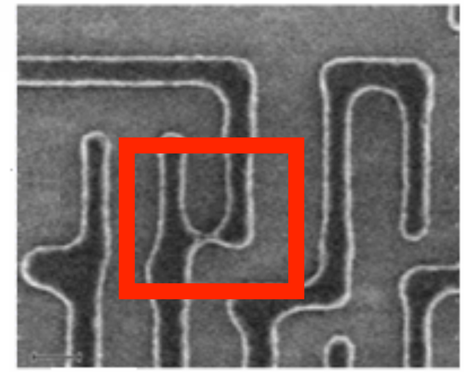
Lithography Hotspot Detection



Layout



Litho simulations

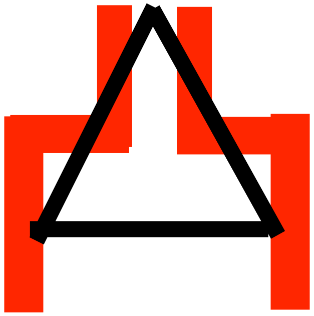


Chip

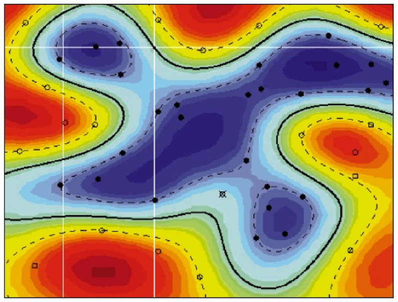
- **Lithographic hotspots**
  - What you see (at design) is NOT what you get (at fab)
  - Hotspots mean poor printability
  - Highly dependent on manufacturing conditions
  - Exist after resolution enhancement techniques
- **Litho-simulations are extremely CPU intensive**
  - Full-blown OPC could take a week
  - Impossible to be used in inner design loop

Various Approaches

[Xu+ ICCAD07]
[Yao+ ICCAD08,
[Khang SPIE06],
etc.



Pattern/Graph Matching



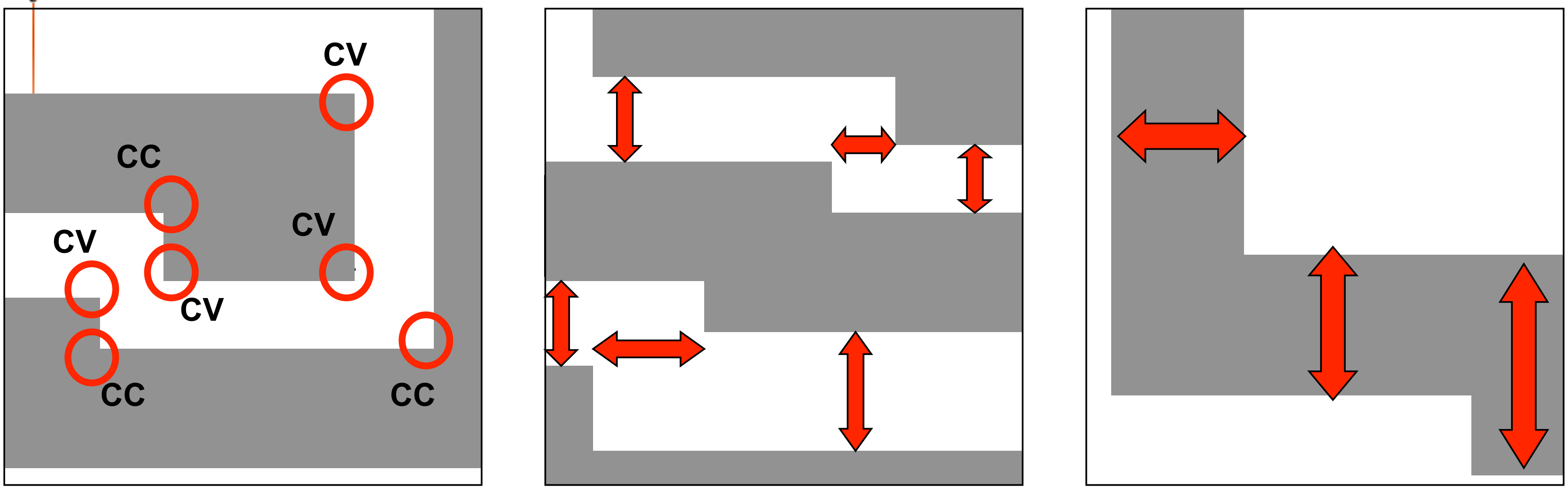
SVM [J. Wuu+ SPIE09]
[Drmanac+ DAC09]
Neural Network Model
[Norimasa+ SPIE07][Ding + ICICDT09]
Regression Model
[Torres+ SPIE09]

Data Mining/Machine Learning

- **Pros and cons**
  - Accurate and fast for known patterns
  - But too many possible patterns to enumerate
  - Sensitive to changing manufacturing conditions
  - High overshoot (false-alarms)

- **Pros and cons**
  - Good to detect unknown or unseen hotspots
  - Accuracy may not be good for "seen" patterns (cf. PM)
  - Hard to trade-off accuracy and false alarms

Layout Analysis

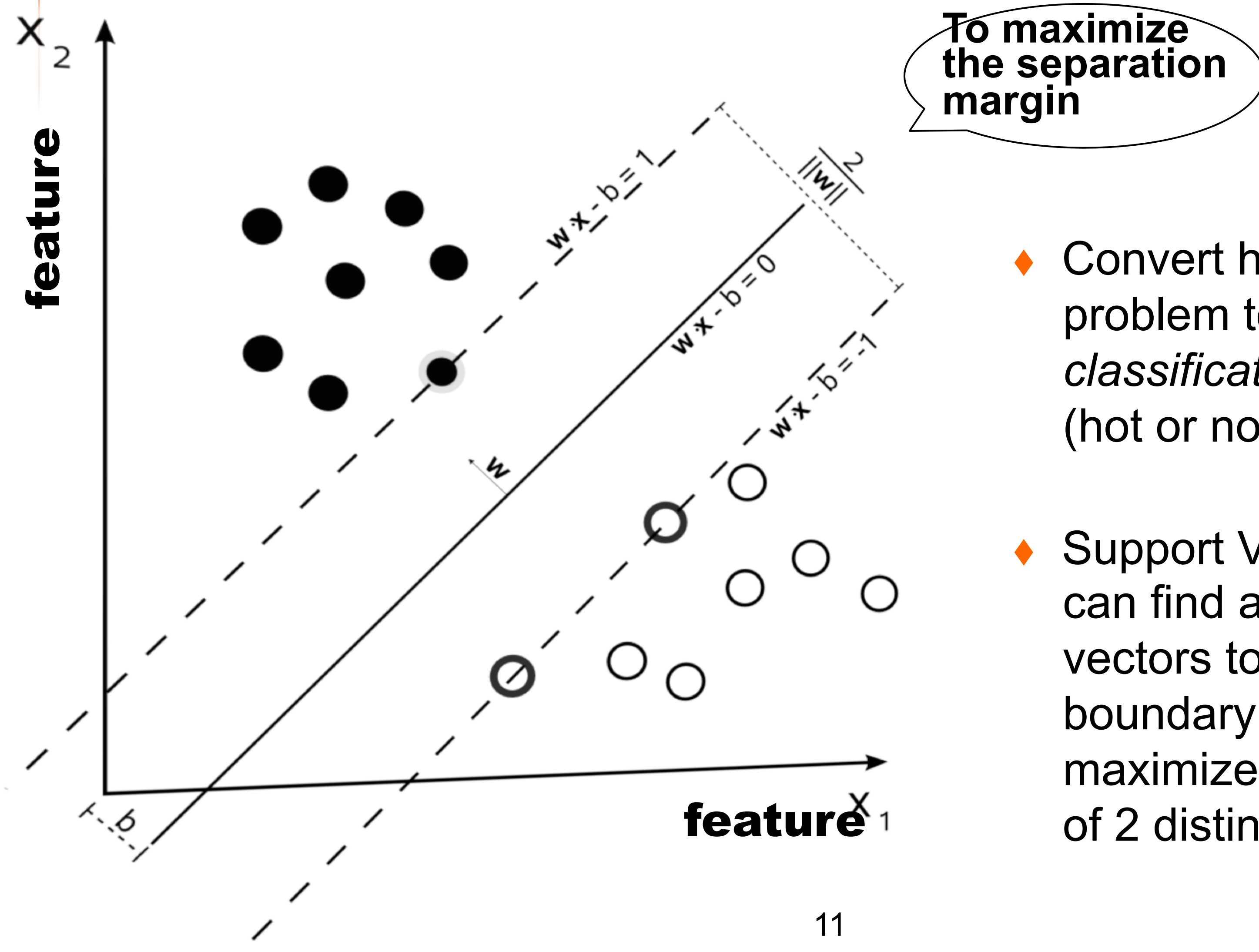


(a) **Corner information** **(b)** **External length** **(c)** **Internal length**

- **Layout Fragmentation**
  - With a pre-defined set of measurement operators
  - Accurate and very fast to apply (e.g., link to CALIBRE API)
  - Full detection to cover whole layout without samplings (cf. window-based approach)
  - Complexity and runtime scale *O(n)*

Machine Learning Kernel - SVM

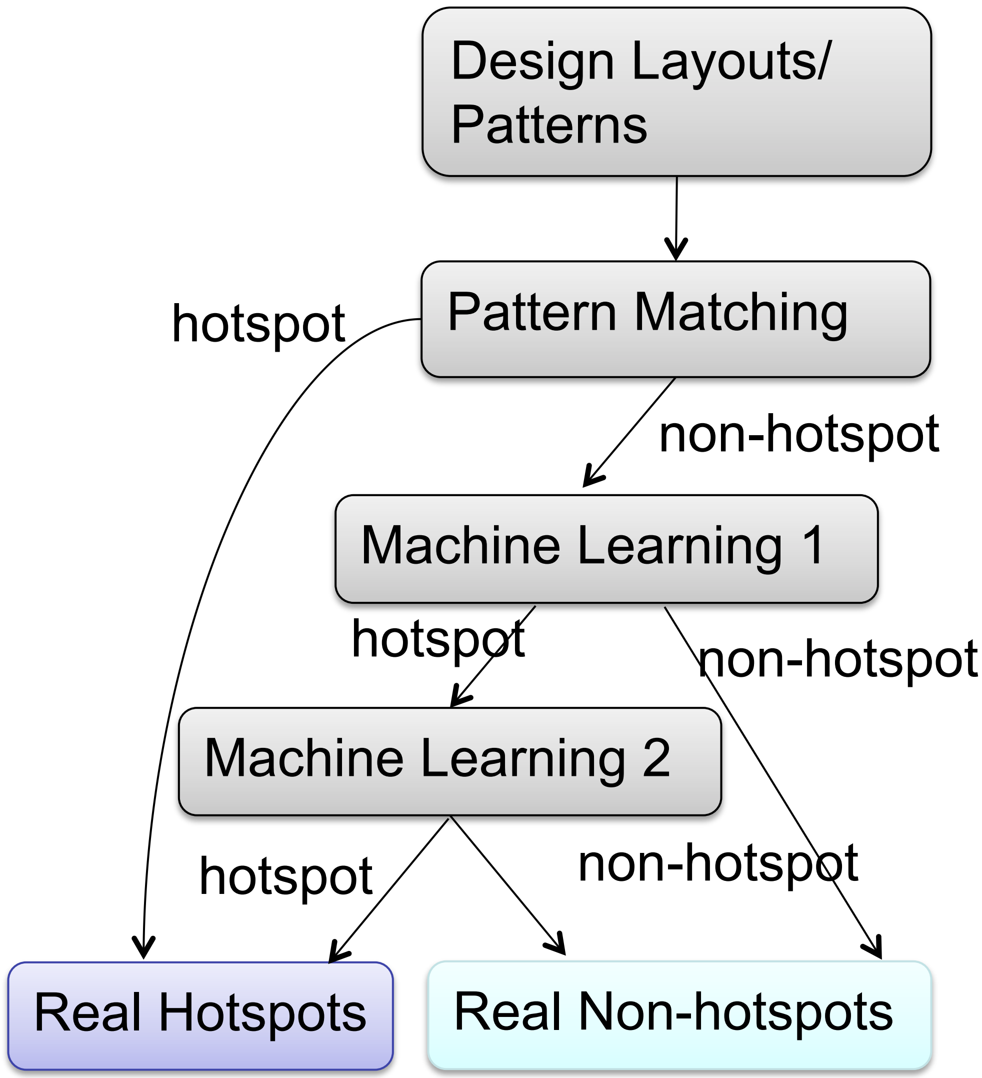
- Support Vector Machine – A linear separation demo



- Convert hotspot detection problem to a *binary classification problem* (hot or nonhot separation)

- Support Vector Machine can find a set of support vectors to construct a boundary plane that maximize the separation of 2 distinctive sets of data

A Naïve Combination Combination of ML and Pattern Matching



Meta-Classification

Pattern Matching Methods
Good **for detecting previously known types of hotspots**

Machine Learning Methods
Good for detecting **new/previously unknown hotspots**

A New Unified Formulation (EPIC)
Good for detecting all types of hotspots with advantageous accuracy/false-alarm (**Meta-Classifier**)

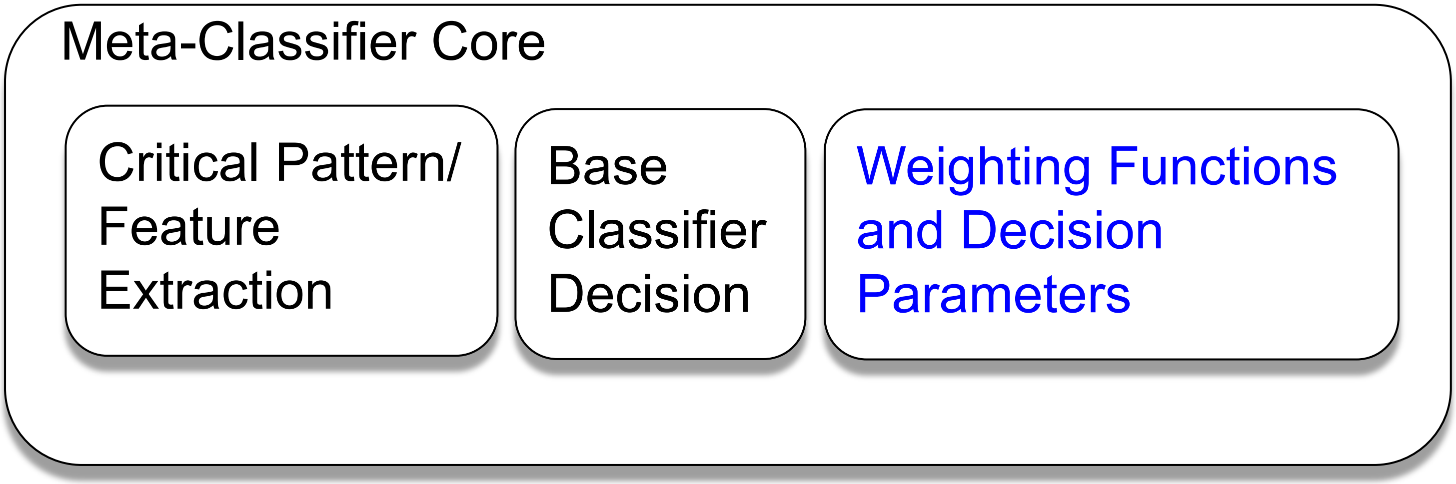
- Meta-Classification combines the strength of different types of hotspot detection techniques

[Ding et al, ASPDAC 2012 BPA]

An Illustrative Example

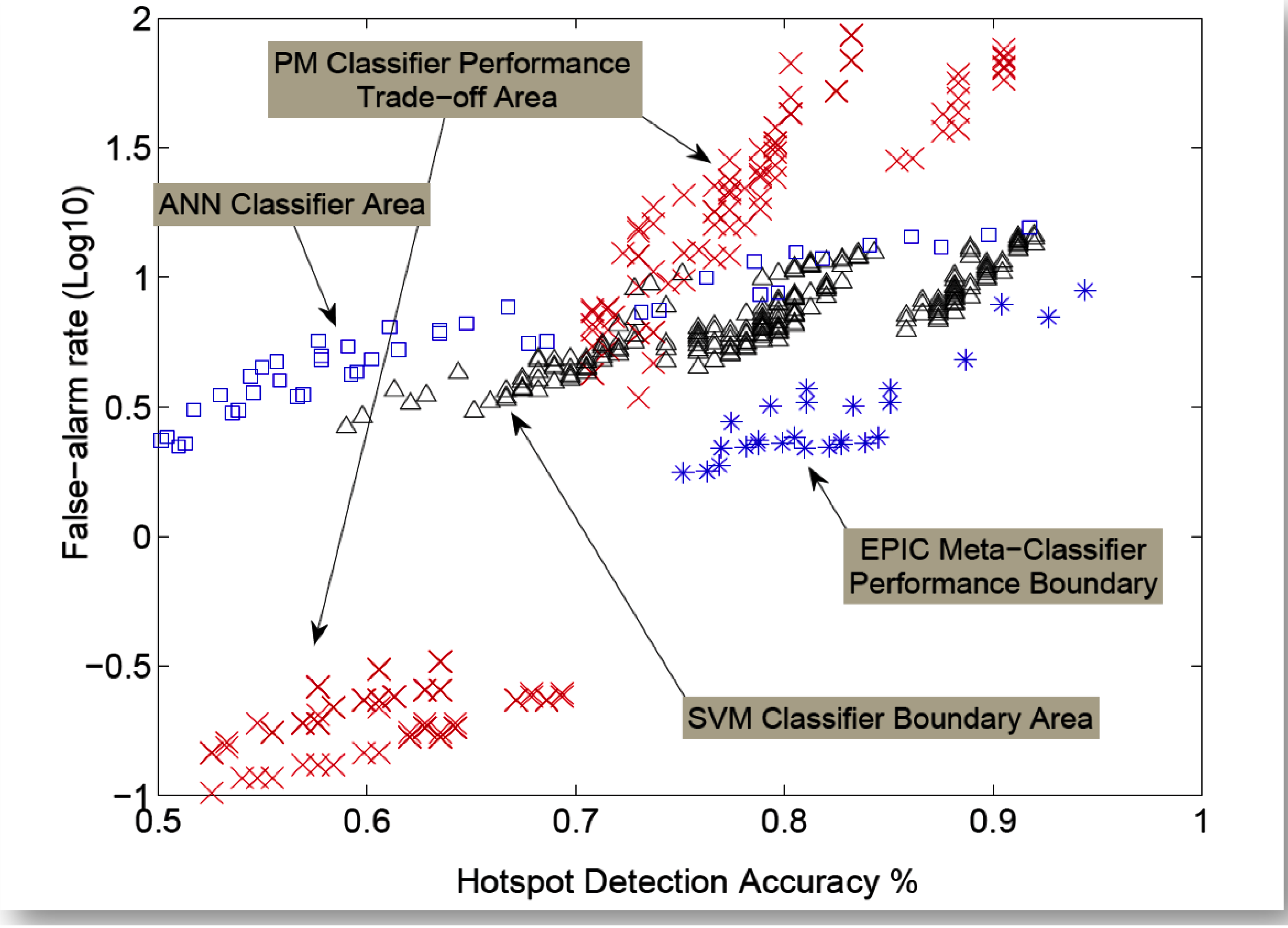
Detection Sub-block	Detection Results (H: hotspot, N: non-hotspot, X: Don't Care)				
Machine Learning 1	X	H	N	H	N
Machine Learning 2	X	H	H	N	N
Pattern Matching	H	N	N	N	N
Final Decision	H	H	N	H	N

Components of Meta-Classifier Core



- Base classifier results are first collected
- Weighting functions to make the overall meta decision (e.g., Minimize Mean Square Error among all samples in data set)
  - Quadratic Programming (QP) formulation
- Accuracy and false-alarm trade-off

False-alarm Rate and Accuracy

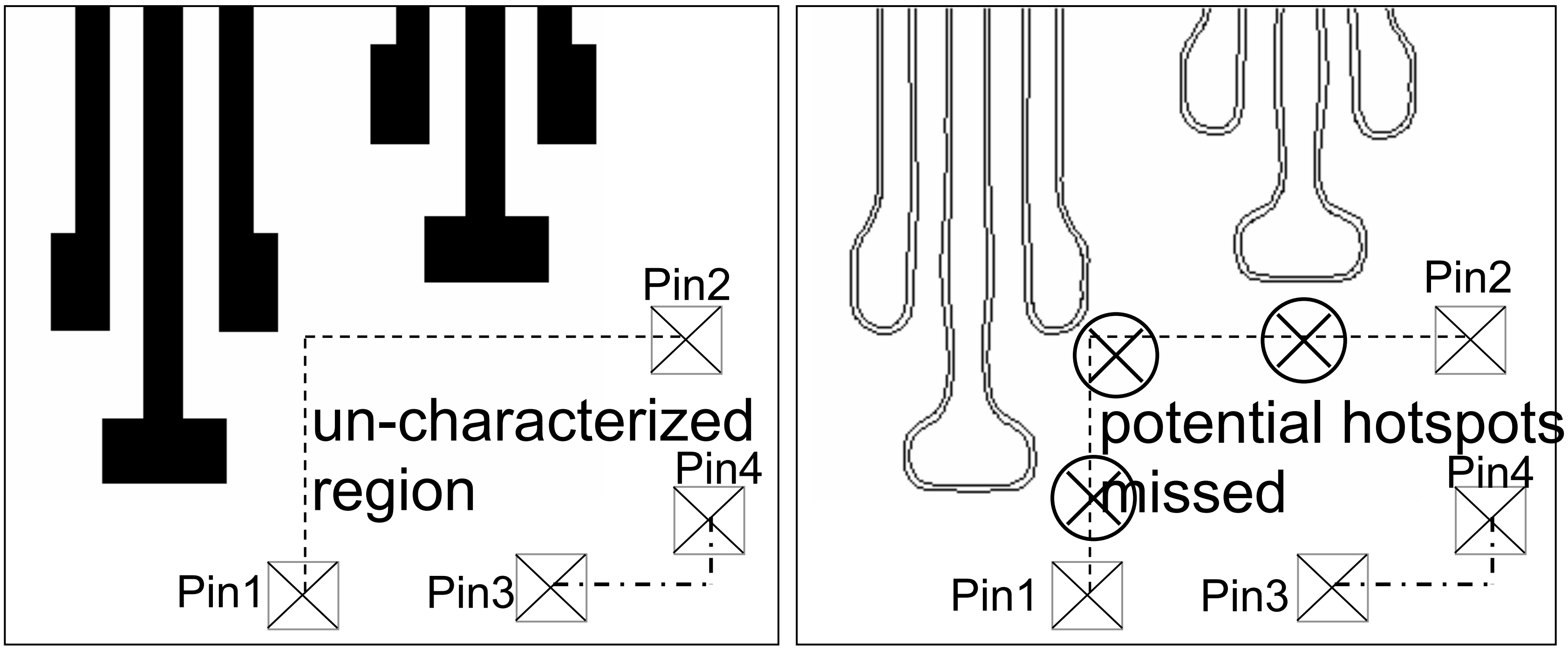


ICCAD'12 Contest

- Released benchmark by Mentor: 2D structures on metal layers with 32 to 28nm processes
- Desired target performance
  - Low false alarm: <100 false hits/mm^2
  - Fast run time: < 1 CPU-hr/mm^2
  - Detection accuracy: > 80%
  - Portability: General calibration strategy
- Publications
  - [Lin et al. DAC 2013]
  - [Yu et al. DAC 2013]
  - [Gao et al. SPIE 2014]
  - ……

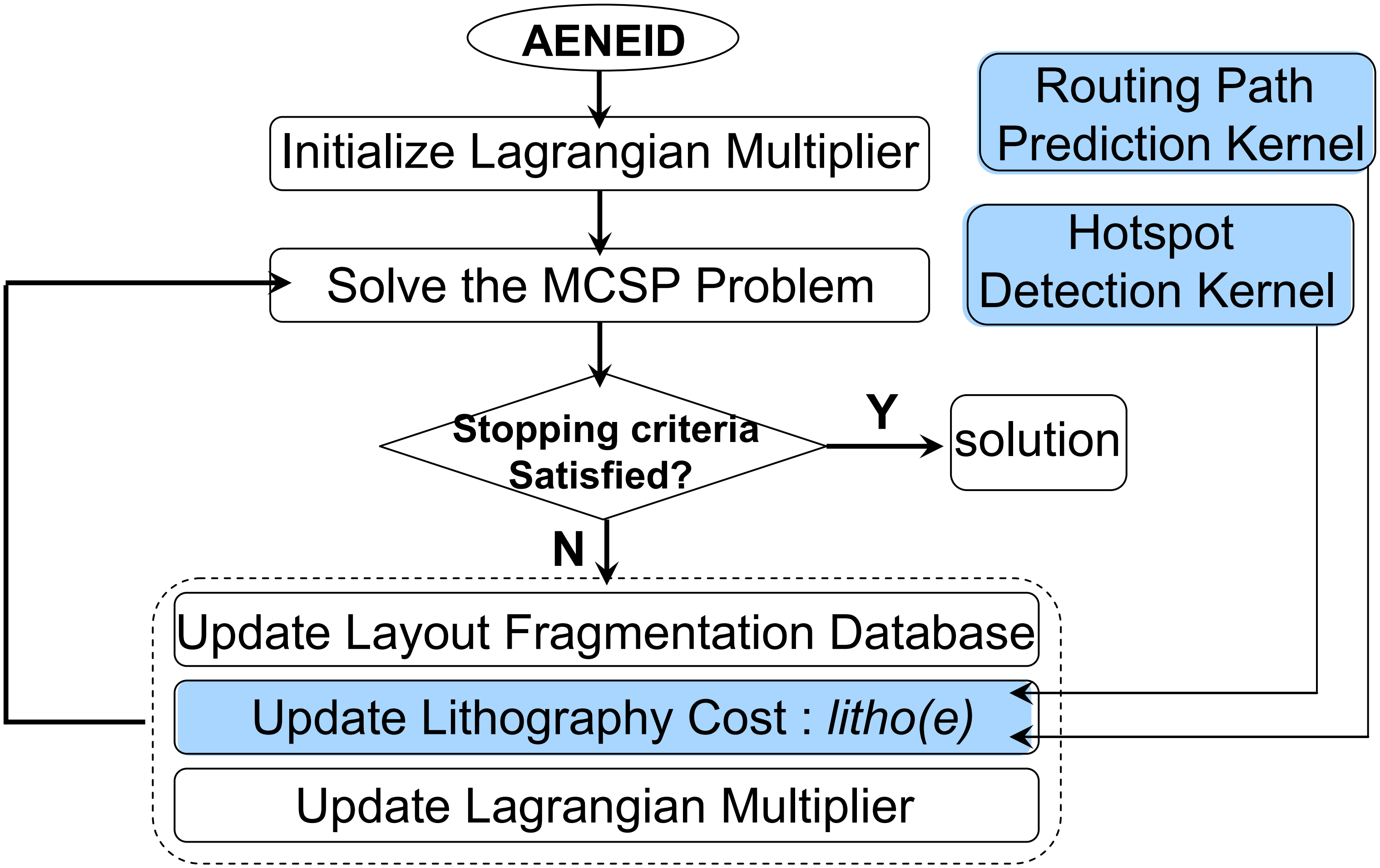
Lithography-Friendly Detailed Routing

- [DAC'11] AENEID: Hotspot learning models in early design stage, used to guide routing



(a) (b)

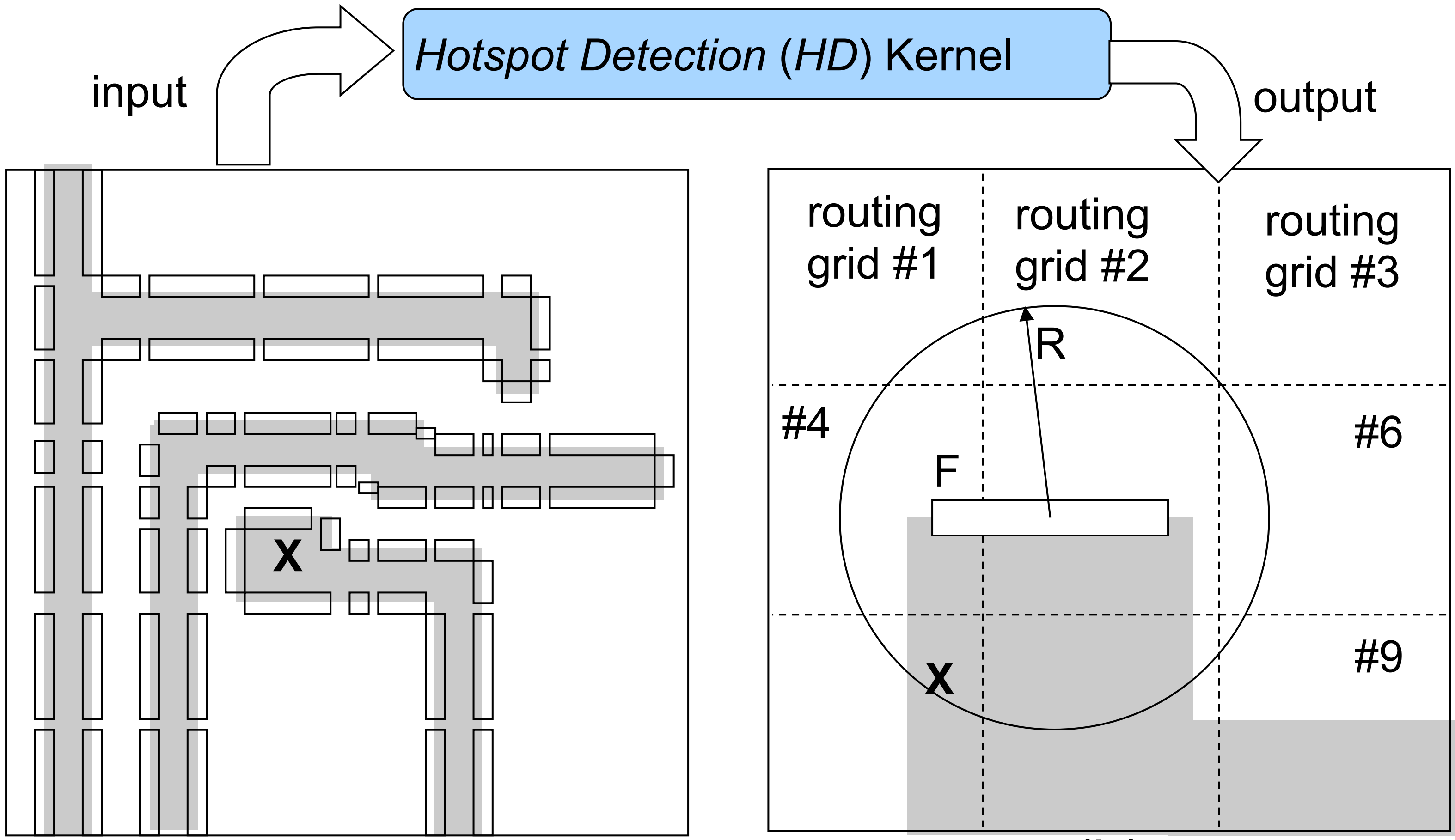
AENEID Overall Flow



- Machine learning models to guide AENEID to avoid hotspot patterns in the early design stages

Cost Function For Detailed Routing-I

$$litho(e) = \boxed{litho(e)^{HD}} + litho(e)^{RPP}$$

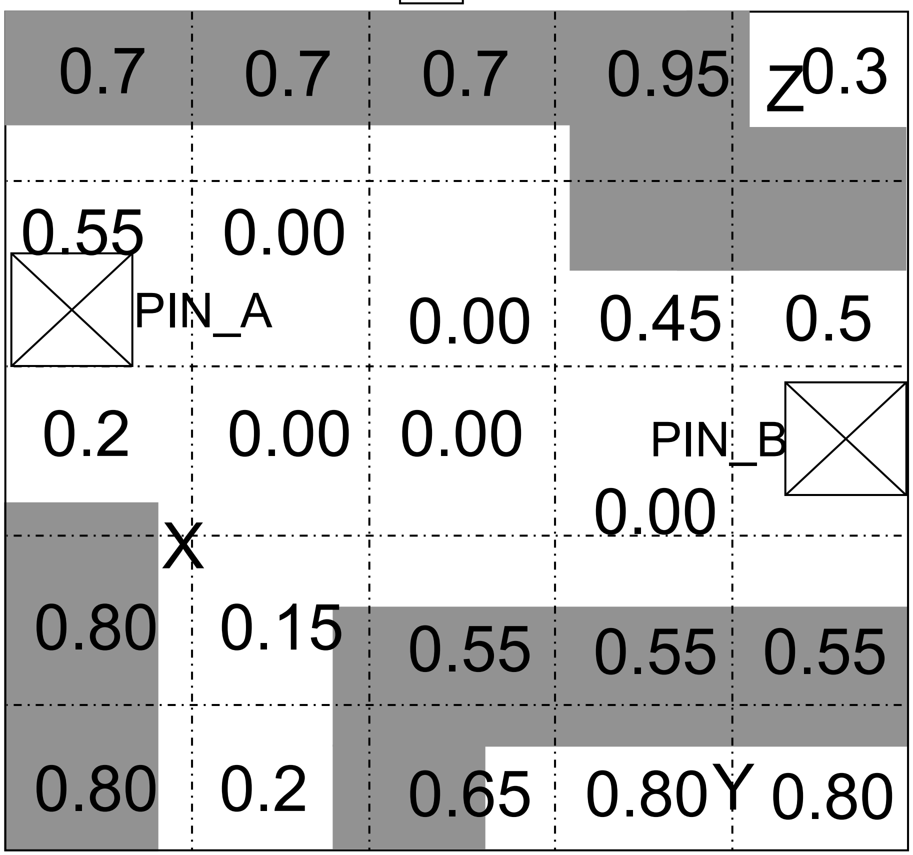


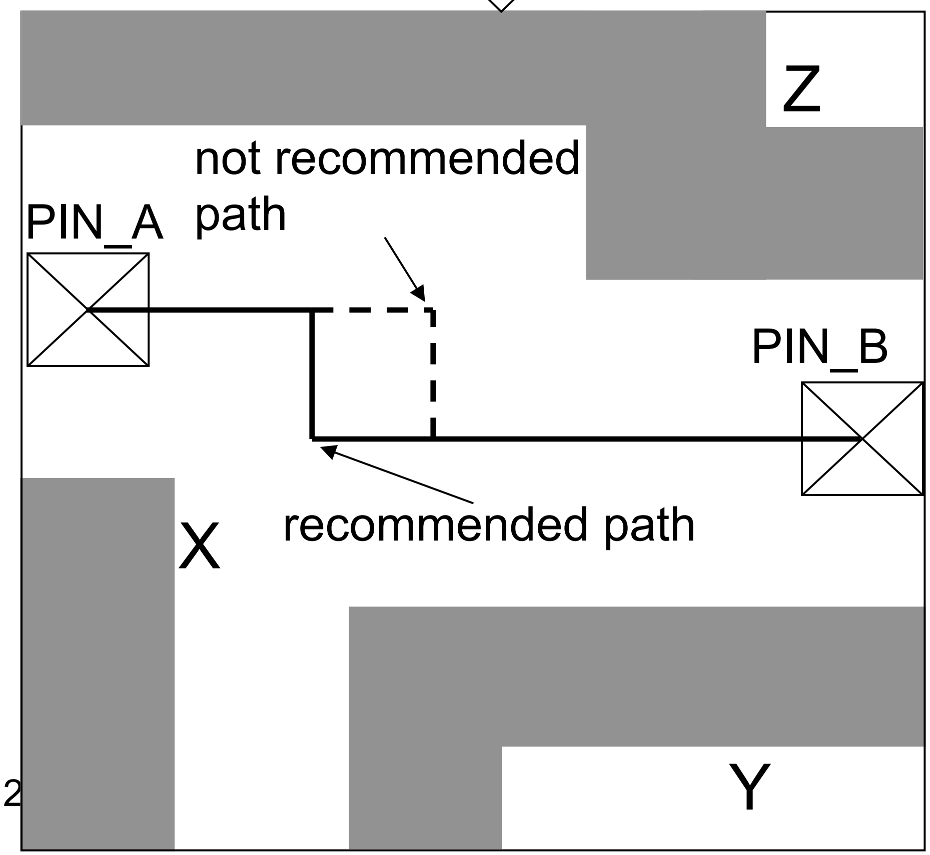
(a) (b)

Cost Function For Detailed Routing-II

$$litho(e) = litho(e)^{HD} + \boxed{litho(e)^{RPP}}$$

Routing Path Prediction (*RPP*) Kernel Processing





Testing Benchmarks and Simulation Results

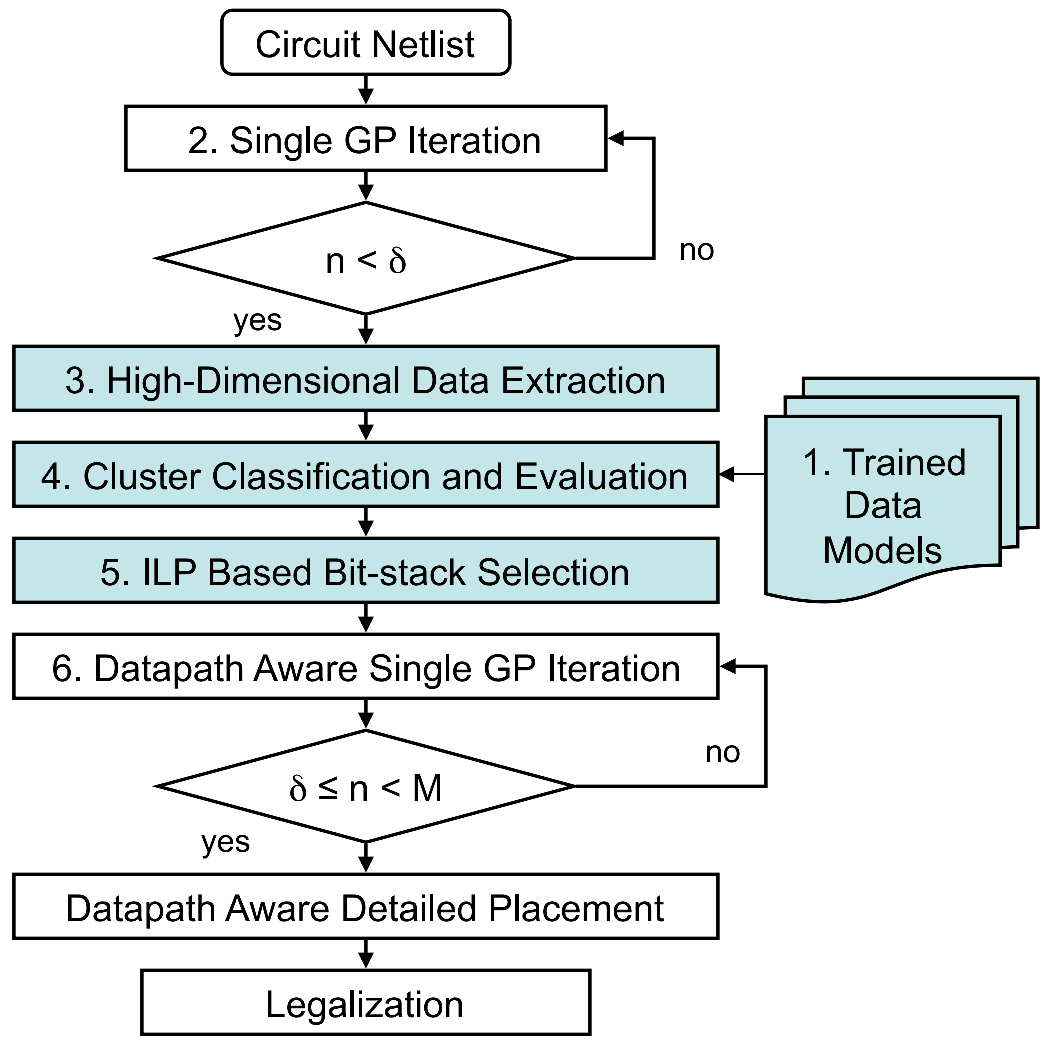
Benchmarks	CK1	CK2	CK3
Layout Size	50X50um^2	100X100um^2	160X160um^2
Nets to Route	0.45K	1.48K	3.4K
M1 Blockage#	1K	8.8K	13.1K
M2 Fragment#	12.2K	41K	152.6K
M2 Blockage#	0.14K	0.47K	2K
M2 Fragment#	0.56K	1.9K	8.3K

Compared with ELIAD (Minsik Cho, et al. TCAD09), AENEID shows 23%-64% hotspot reduction at the cost of 30% extra run-time without penalty on total wire-length

	AENEID											
	HD						HD + RPP					
Circuit	CK1		CK2		CK3		CK1		CK2		CK3	
Circuit Size um^2	50^2		100^2		160^2		50^2		100^2		160^2	
Wire-length um	859.3		5502.0		24797.0		859.1		5502.0		24797.5	
Run-time sec	8		409		3291		8		400		3279	
Run-time overhead %	33		38		19		33		35		18	
Metal layer	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2
Hotspot#	11	2	34	7	90	17	8	2	22	5	58	15
Hotspot reduc %	35	33	48	30	44	26	53	33	66	50	64	35
Avg. hotspot reduc %	36						50					
Avg. extra run-time %	30						29					

Machine Learning for Placement

- Data mining and extraction based on not just graph but also physical information
- We can extract data-path like structures even for “random” logics
- Use them to explicitly guide placement
- Very good results obtained cf. other leading placers like simPL, NTUPlace, mPL, CAPO



[Ward+, DAC’ 12]

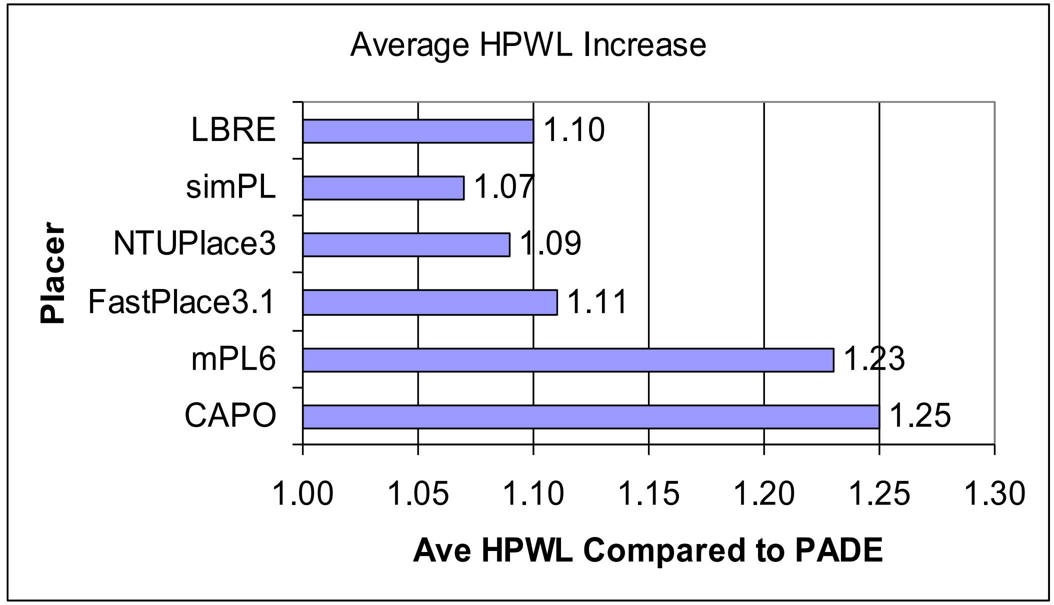
Datapath Placement Techniques

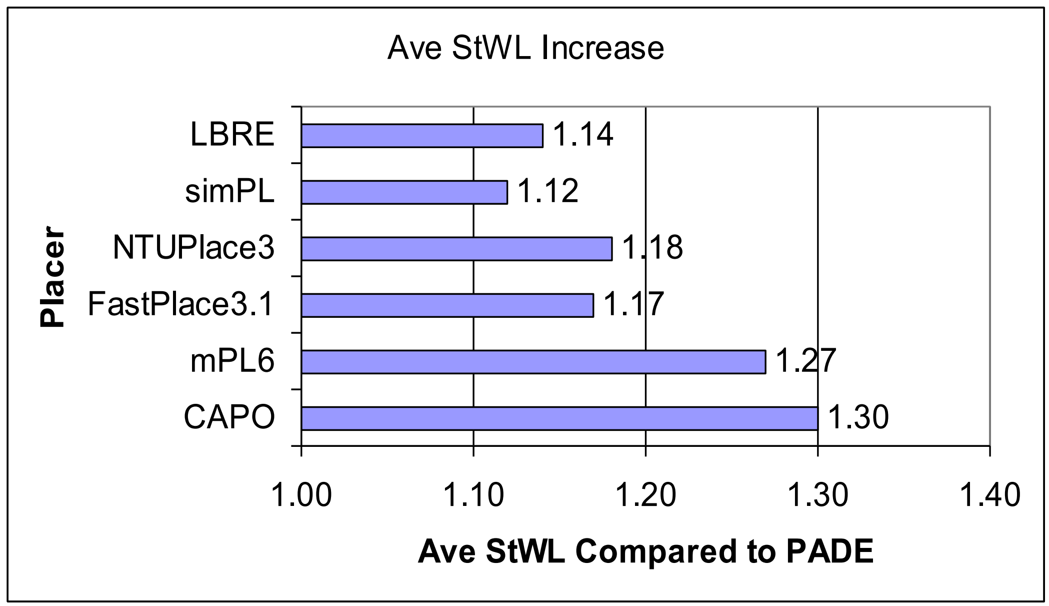
- Steiner wirelength (StWL) improvement through bit-stack alignment
  - Significantly improves total StWL and routing congestion
- Datapath placement techniques
  - Skewed Weighting with Step Size Scheduling
  - Fixed-Point Alignment Constraint
  - Bit-Stack Aligned Cell Swapping
  - Datapath Group Repartitioning
- Integrate alignment constraints into force-directed placement
- Simultaneously place datapath & random logic

[Ward+, ISPD' 12]

PADE: Hybrid Experimental Results

- All numbers are average wirelength ratio cf. PADE
- PADE without datapath extraction generates the simPL wirelength results
- HPWL: 7%+ better
- StWL: 12%+ better





PADE: ISPD2005 Results

- **PADE Wirelength results on the ISPD 2005 Placement Benchmarks**
- **At least 2% better in HPWL**
- **At least 3% better in StWL**
- **Highlights the effectiveness of structure aware extraction**

	CAPO	mPL6	FastPlace3.1	NTUPlace3	simPL	PADE
Adaptec1	97.22	86.2	88.75	91.06	87.05	**85.12**
Adaptec2	114.54	100.64	104.03	99.06	102.13	**98.92**
Adaptec3	296.22	235.06	239.7	234.52	228.32	**222.08**
Adaptec4	257.47	208.85	215.02	211.86	201.82	**196.23**
Bigblue1	127.72	108.31	**105.24**	110.02	109.94	106.98
Bigblue2	189.6	174.69	178.44	175.27	168.65	**164.33**
Bigblue3	452.91	370.7	421.31	389.39	369.61	**361.96**
Bigblue4	1105.52	930.63	911.64	974.44	901.85	**883.82**
AVE	1.22	1.04	1.07	1.06	1.03	1.00

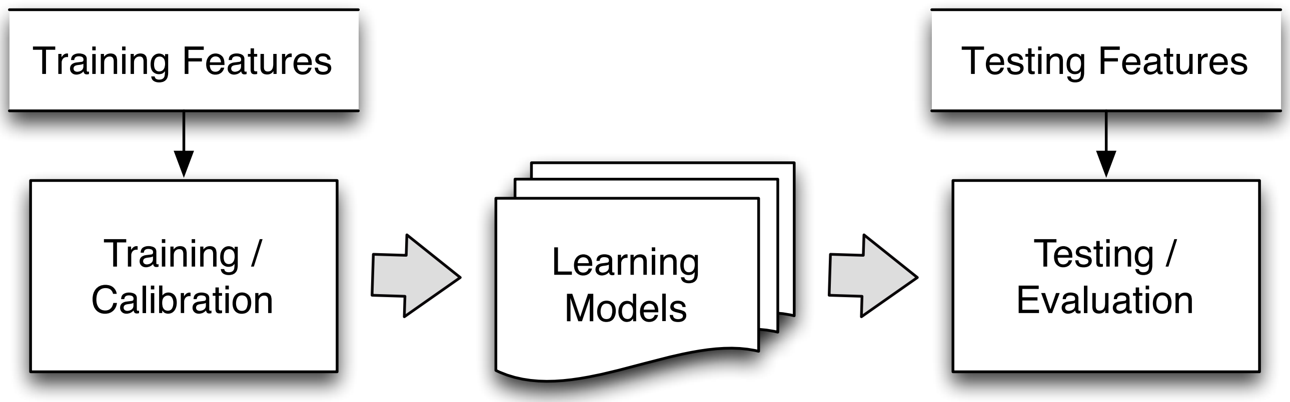
Outline

- Modern VLSI Challenges
- Machine Learning and Pattern Matching 101
- Applications in VLSI Design and Verification
- Some Advanced Issues
- Conclusion

Issue 1: ML or PM?

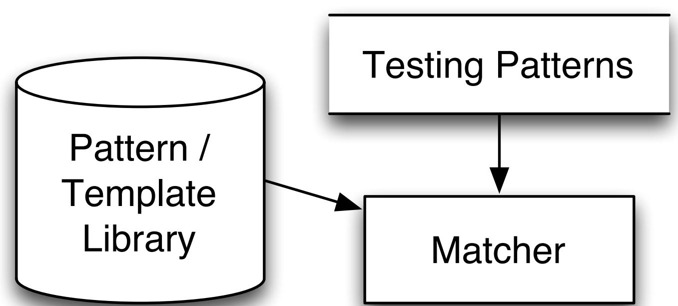
Machine Learning:

- (+) good to unseen data
- (-) longer training time



Pattern Matching:

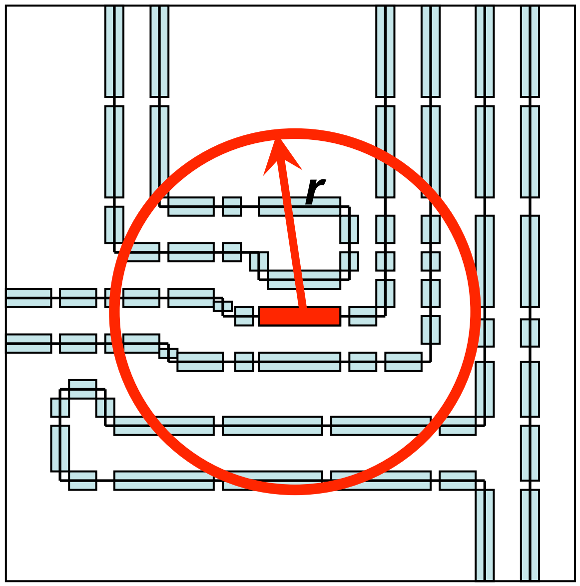
- (+) Easy to implement, fast
- (-) Sensitive to process change

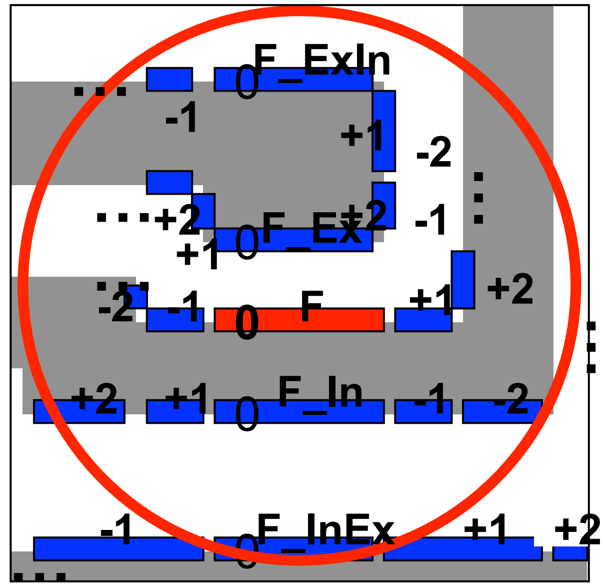


Hybrid approaches are desirable!

Issue 2: Feature Extraction

- Fragmentation based feature [ASPDAC’11, SPIE’14]





$$V_F = \mathop{\Theta}_{i}^{\widetilde{F}_i \in \delta_r^F} \{f_{corn}(\widetilde{F}_i) \oplus f_{ext}(\widetilde{F}_i) \oplus f_{int}(\widetilde{F}_i) \oplus f_{misc}(\widetilde{F}_i)\}$$

$$\widetilde{F} = [F, F\_Ex, F\_In, F\_InEx, F\_ExIn...]$$

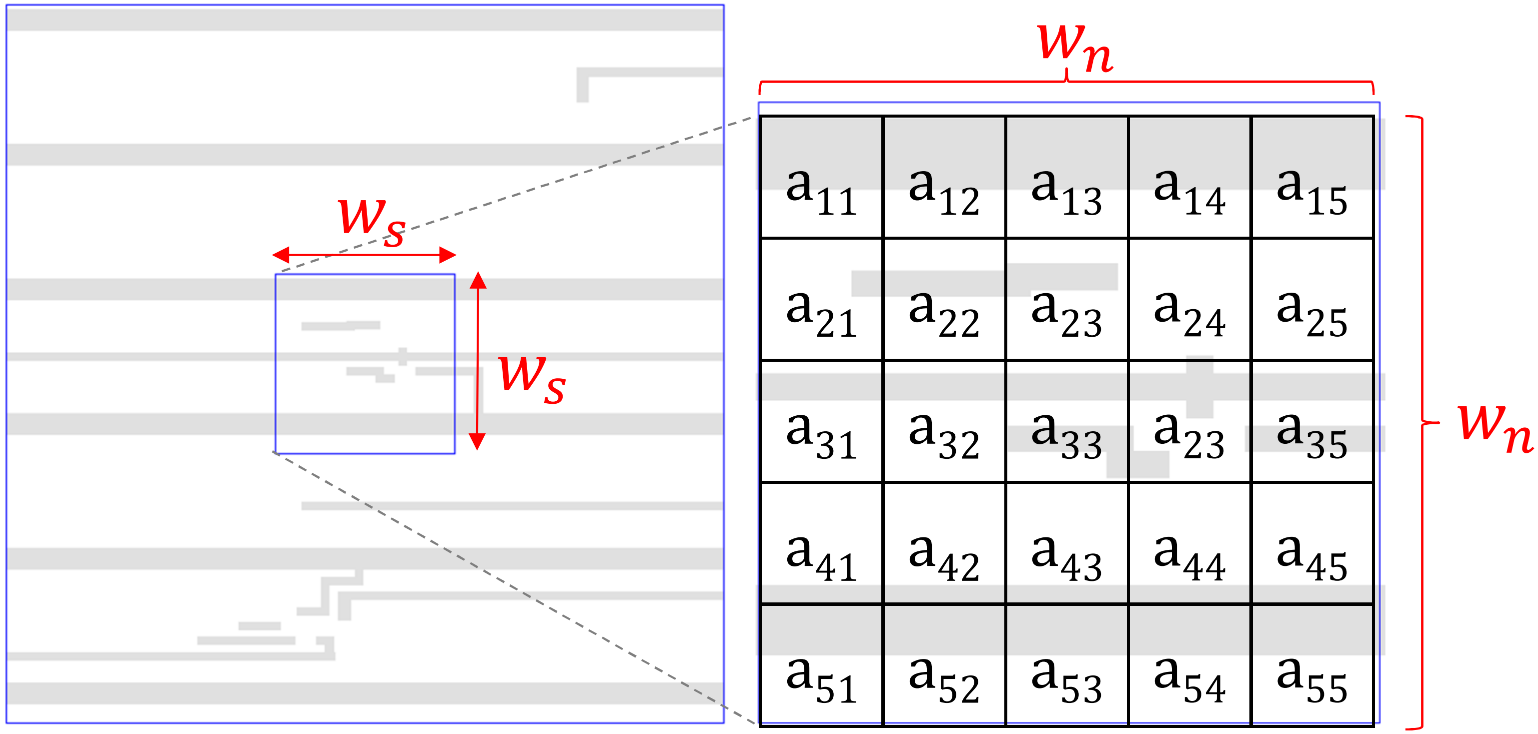
V_F is the feature vector associated with *fragment F*

Θ is a concatenate function; $\oplus$ is a sort-n-combine function

δ_r^F includes both proximity and some peripheral information

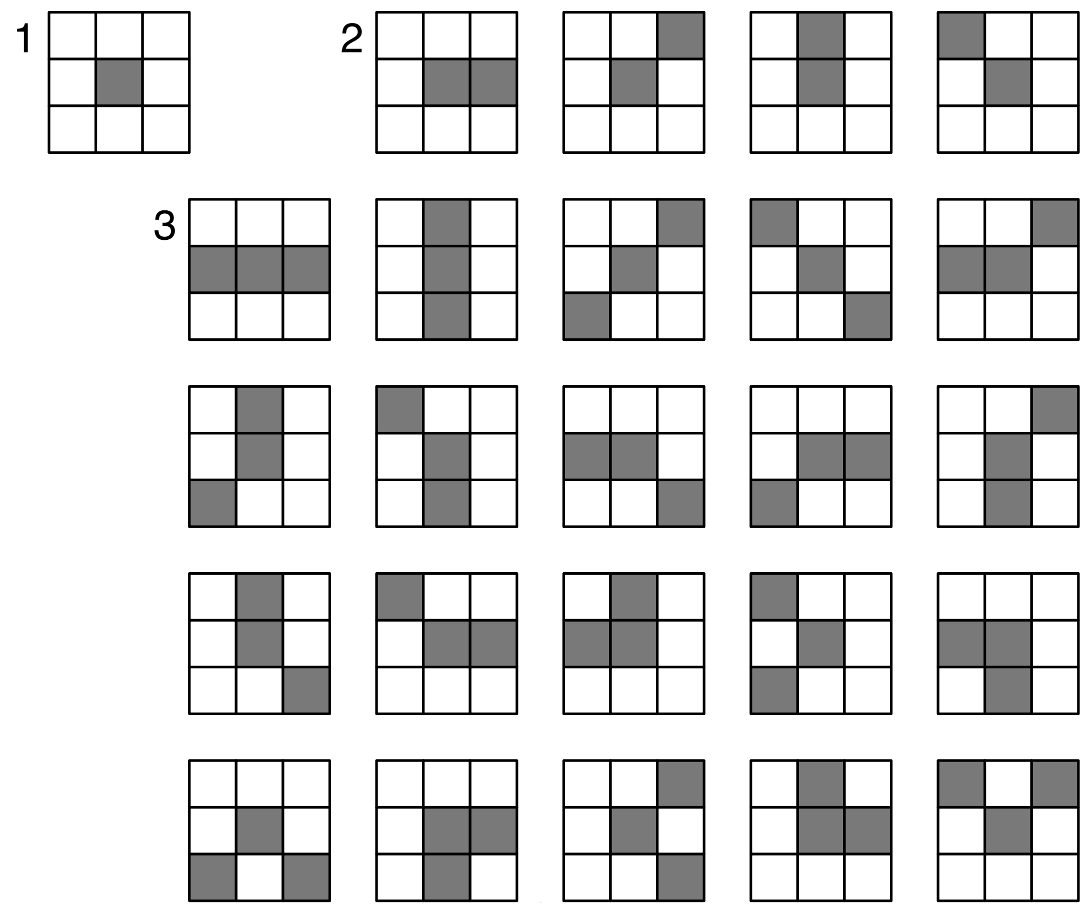
Issue 2: Feature Extraction

- Density based feature [Wuu+, ASPDAC'11; Matsunawa +, SPIE'15]



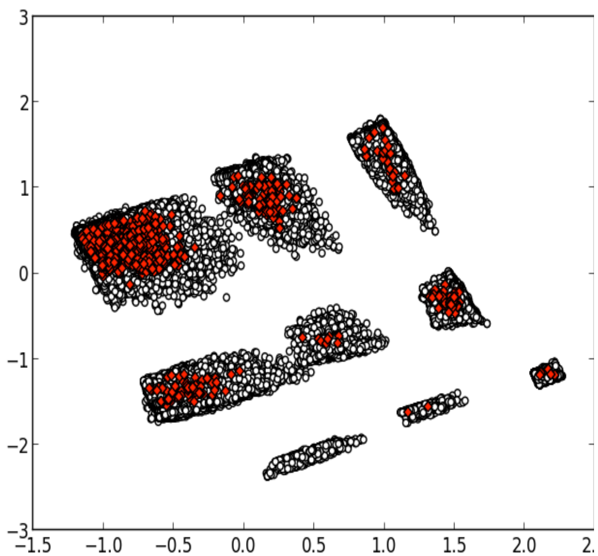
Issue 2: Feature Extraction (cont.)

- HLAC based feature [Nosato+, JM3'14]
  - higher-order local autocorrelation (HLAC)
  - 25 local masks => 25 dimensional vector feature

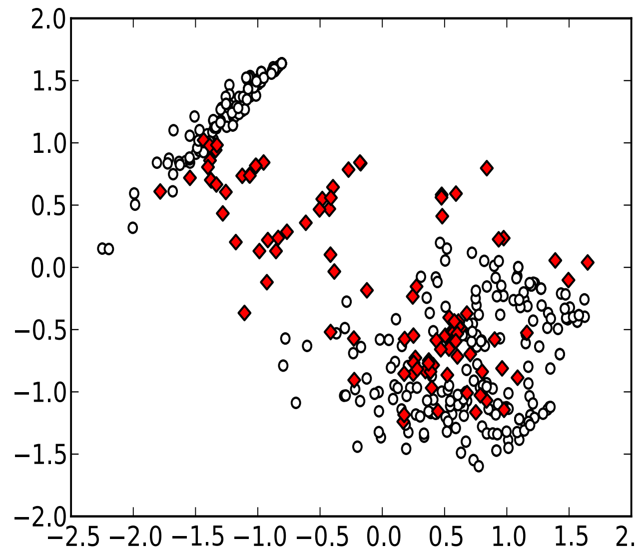


Issue 2: Feature Evaluation

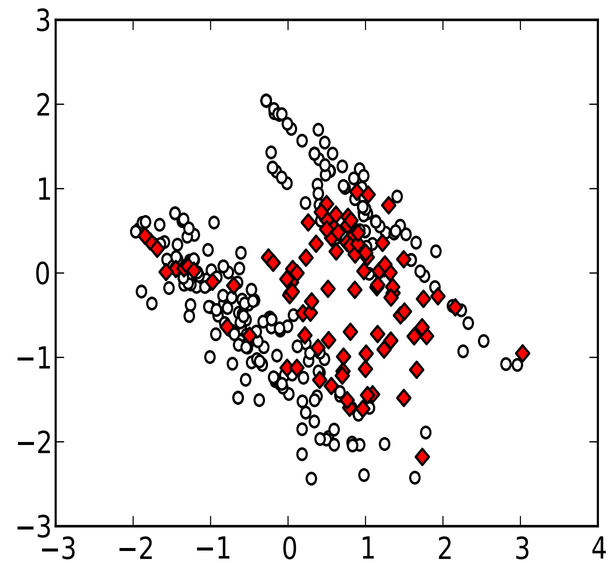
- Analyze Feature Space



Fragmentation based



Density based



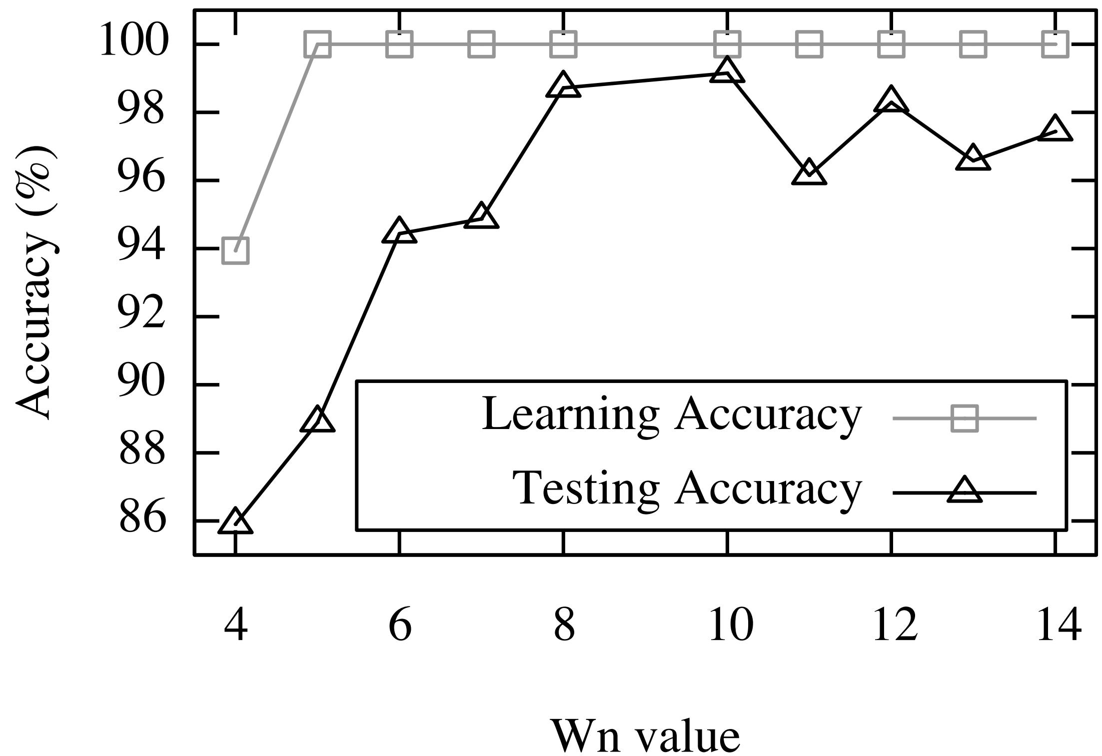
HLAC based

- Measure feature distances [Matsunawa+, SPIE'15]

$$d_i = \frac{\sqrt{(x_i - \mu)^T V^{-1} (x_i - \mu)} - d_{NHS_{min}}}{d_{NHS_{max}} - d_{NHS_{min}}}$$

Issue 3: Overcome Overfitting

- Overfitting: good training, but bad testing



- Possible Solutions:
  - Regularization (additional constraints or objective terms)
  - Cross validation

Conclusion

- Machine learning and pattern matching 101
- Applications in VLSI design and verification
  - Lithography hotspot detection
  - Lithography friendly routing
  - Datapath-like circuit extraction and placement
- Still many open problems and opportunities
  - Hybrid machine learning and pattern matching
  - Feature extraction and classification
  - Overfitting in machine learning
  - Cross-layer applications