Detecting Multi-Layer Layout Hotspots with Adaptive Squish Patterns

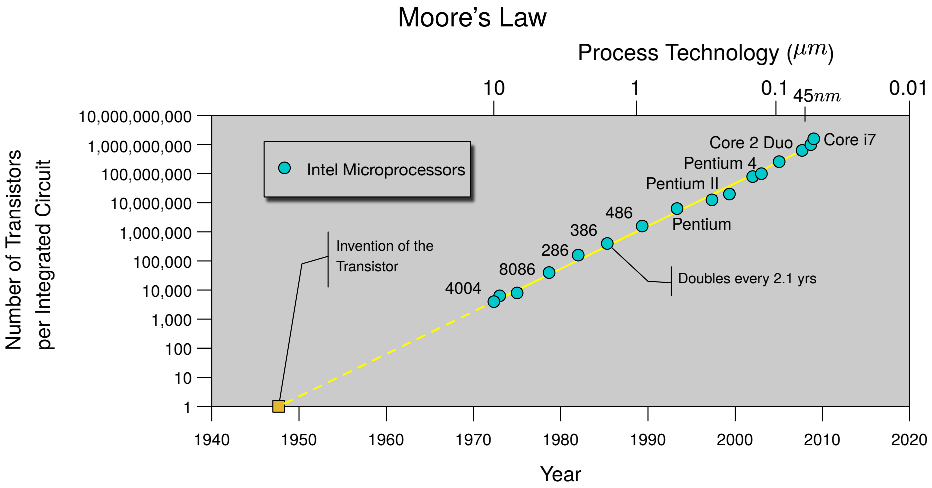
Haoyu Yang[1], Piyush Pathak[2], Frank Gennari[2], Ya-Chieh Lai[2], Bei Yu[1]

[1]The Chinese University of Hong Kong
[2]Cadence Design Systems Inc.

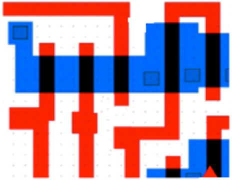
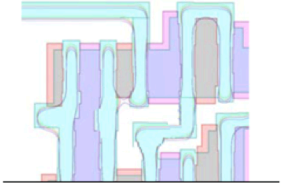
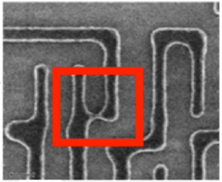
Outline

Outline

Moore's Law to Extreme Scaling

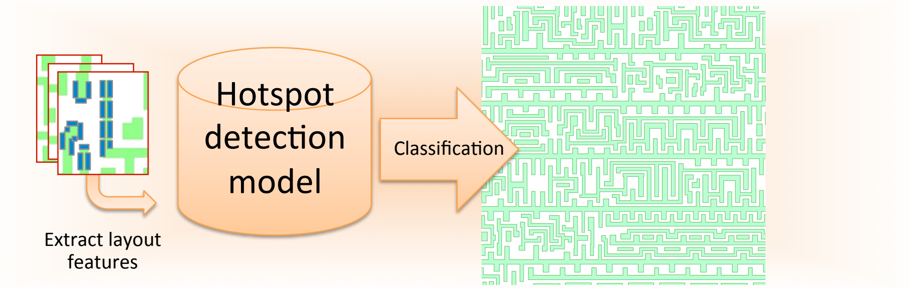


Lithography Proximity Effect

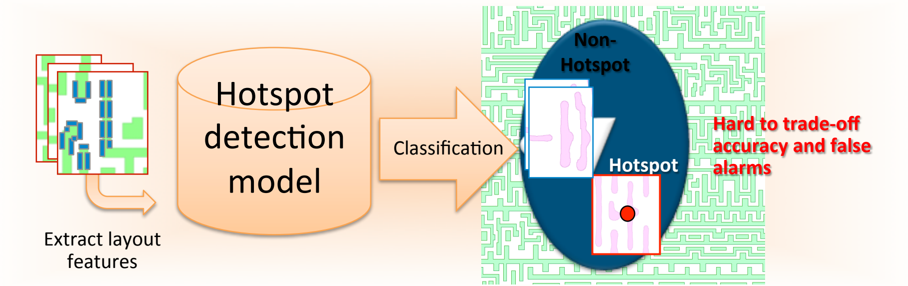




- What you see $\neq$ what you get
- Diffraction information loss
- RET: OPC, SRAF, MPL
- Worse on designs under 10nm or beyond

Machine Learning based Hotspot Detection

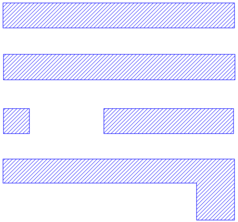


Machine Learning based Hotspot Detection

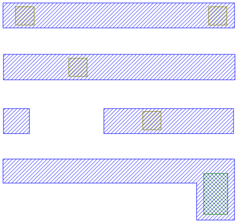


- Predict new patterns
- Decision-tree, ANN, SVM, Boosting, Deep Neural Networks
- [Drmanac+,DAC'09] [Ding+,TCAD'12] [Yu+,JM3'15] [Matsunawa+,SPIE'15] [Yu+,TCAD'15][Zhang+,ICCAD'16][Yang+,DAC'17][Yang+,TCAD]

Multi-Layer Hotspots



(a)

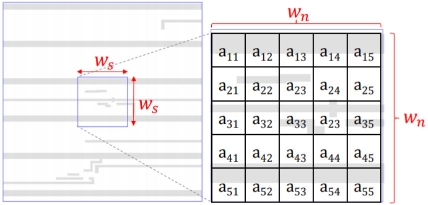


(b)

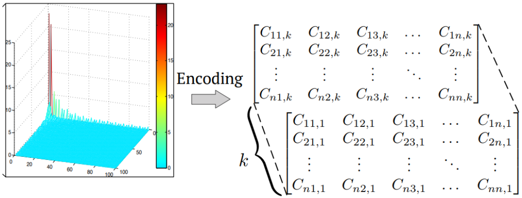
- More complicated patterns
- More failure types (e.g. Metal-to-Via failure)

Layout Representations

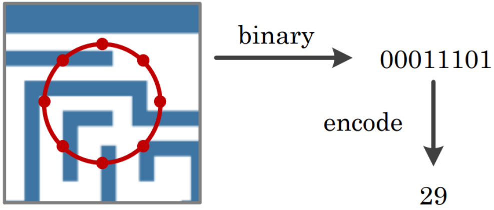
- Density-based features [Matsunawa+,SPIE'15]



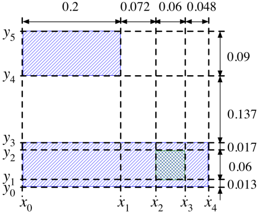
- Feature tensor extraction [Yang+,TCAD]



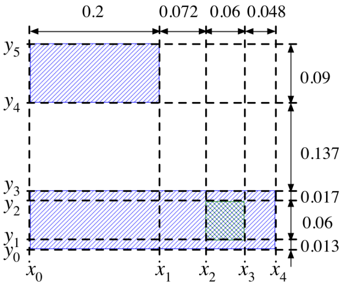
- Concentric circle sampling [Zhang+,ICCAD'16]



- Squish patterns [Gennari+, US Patent]



Squish Patterns



A simple multilayer pattern example with scan lines.

$$\vec{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 3 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix},$$

$$\vec{\delta}_x = \begin{bmatrix} 0.2 & 0.072 & 0.06 & 0.048 \end{bmatrix},$$

$$\vec{\delta}_y = \begin{bmatrix} 0.013 & 0.06 & 0.017 & 0.137 & 0.09 \end{bmatrix}.$$

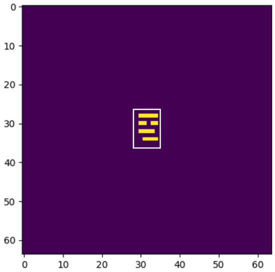
- Lossless
- Storage-friendly
- Incompatible with most machine learning engines.

  Squish representation does not guarantee a fixed tensor dimensionality for a given clip size.

Outline

An Alternative of Padding

- Legacy padding induces large fraction of zeros that are not informative to CNNs.



- Instead of padding, we repeat certain rows or columns of squish topologies.
- $\vec{\delta}$s are adjusted accordingly to make the pattern unchanged.

$$\vec{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 3 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \vec{T}' = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 3 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Which Rows/Columns Are to Be Duplicated/Repeated?

- In machine learning, if some entries of the input are too large/small, there will be bias related to those entries.
- Subtract RGB means in conventional image classification tasks.
- Duplicate rows/columns with larger deltas.

Adaptive Squish Problem:

$$\min_{\vec{s}} \quad ||\vec{\delta'}||_\infty \tag{1a}$$

$$\text{s.t.} \quad \delta'_i = \delta_i / s_i, \forall i, \tag{1b}$$

$$s_i \in \mathbb{Z}^+, \forall i, \tag{1c}$$

$$\sum_i s_i = d. \tag{1d}$$

Repeat Elements

RepeatElements: $\vec{M}' = \texttt{RepeatElements}(\vec{M}, \vec{s}, a)$, which duplicates the columns ($a = 0$) or rows ($a = 1$) of a matrix $\vec{M} \in \mathbb{R}^{a_1 \times a_2}$ by certain times such that the shape of the new matrix $\vec{M}'$ will be increased to a desired value.

- $a = 0:$

$$\vec{m}'_k = \vec{m}_j, \forall \sum_{i=1}^{j-1} s_i < k \leq \sum_{i=1}^{j} s_i. \tag{2}$$

- $a = 1:$

$$\texttt{RepeatElements}(\vec{M}, \vec{s}, 1) = \texttt{RepeatElements}(\vec{M}^\top, \vec{s}, 0)^\top. \tag{3}$$

.

Repeat Elements

For example, if we let $\vec{s} = \begin{bmatrix} 1 & 1 & 2 & 1 \end{bmatrix}^\top$ and $a = 0$, then the `RepeatElements` operation on the topology matrix $\vec{T}$ will result in

$$\vec{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 3 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \to \vec{T}' = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 3 & 3 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{4}$$

Adaptive Squish: Solution 1

Algorithm 1 Obtaining adaptive squish patterns with a greedy procedure.

Input: T, $\boldsymbol{\delta}$, a, d_0, d;
Output: T, $\boldsymbol{\delta}$;

1: **while** $d_0 < d$ **do**
2: $\quad \boldsymbol{s} \leftarrow \mathbf{1} \in \mathbb{R}^{d_0}$, $i \leftarrow \arg\max_i \{\delta_i | i = 1, 2, ..., d_0 - 1\}$;
3: $\quad s_i \leftarrow 2$, $\delta_i \leftarrow \delta_i / 2, \forall i$;
4: $\quad \boldsymbol{\delta} \leftarrow$ `RepeatElements`$(\boldsymbol{\delta}, \boldsymbol{s}, 1)$;
5: $\quad \boldsymbol{T} \leftarrow$ `RepeatElements`$(\boldsymbol{T}, \boldsymbol{s}, a)$;
6: $\quad d_0 \leftarrow d_0 + 1$;
7: **end while**

- Extend a 3×3 squish topology to shape 3×6.

$$\vec{T} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix},$$

$$\vec{\delta}_x = \begin{bmatrix} 28 & 18 & 2 \end{bmatrix}, \vec{\delta}_y = \begin{bmatrix} 16 & 16 & 16 \end{bmatrix}.$$

$$\vec{T}' = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

$$\vec{\delta}'_x = \begin{bmatrix} 7 & 7 & 14 & 9 & 9 & 2 \end{bmatrix}, \vec{\delta}'_y = \begin{bmatrix} 16 & 16 & 16 \end{bmatrix}.$$

Adaptive Squish: Solution 2

Algorithm 2 Deriving an approximate solution of Formula (8) that will be used for generating adaptive squish patterns.

Input: $\boldsymbol{\delta}$, d_0, d;
Output: $\boldsymbol{s}$;

$l \leftarrow \sum_i \delta_i$;
$t \leftarrow l/(d-1)$;
$s_i \leftarrow \max\{1, \texttt{int}(\delta_i/t)\}, \forall i$;
while $\sum_i s_i < d-1$ **do**
  $\delta'_i \leftarrow \delta_i/s_i, \forall i$;
  $i \leftarrow \arg\max_i\{\delta_i | i = 1, 2, \ldots, d_0 - 1\}$;
  $s_i \leftarrow s_i + 1$;
end while
$\delta_i \leftarrow \delta_i/s_i, \forall i$;
$\boldsymbol{\delta} \leftarrow \texttt{RepeatElements}(\boldsymbol{\delta}, \boldsymbol{s}, 1)$;
$\boldsymbol{T} \leftarrow \texttt{RepeatElements}(\boldsymbol{T}, \boldsymbol{s}, a)$;

- Extend a 3×3 squish topology to shape 3×6.

$$\vec{T} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix},$$

$$\vec{\delta}_x = \begin{bmatrix} 28 & 18 & 2 \end{bmatrix}, \vec{\delta}_y = \begin{bmatrix} 16 & 16 & 16 \end{bmatrix}.$$

$$\vec{T}' = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

$$\vec{\delta}'_x = \begin{bmatrix} 9.33 & 9.33 & 9.33 & 9 & 9 & 2 \end{bmatrix}, \vec{\delta}'_y = \begin{bmatrix} 16 & 16 & 16 \end{bmatrix}.$$

Adaptive Squish: Data Preparation

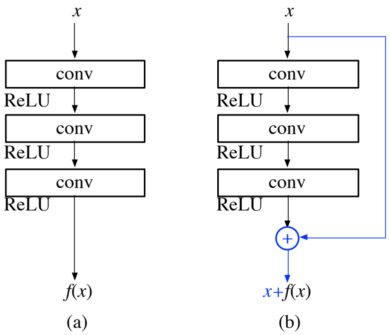
- The squish topology and $\vec{\delta}$s will be stacked together into a 3D tensor $[\vec{T}'; \vec{\delta}'_X; \vec{\delta}'_Y]$ that will be fed into neural networks for training and inference, where,

$$
\vec{T}' = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix},
$$

$$
\vec{\delta}'_X = \begin{bmatrix} 9.33 & 9.33 & 9.33 & 9 & 9 & 2 \\ 9.33 & 9.33 & 9.33 & 9 & 9 & 2 \\ 9.33 & 9.33 & 9.33 & 9 & 9 & 2 \end{bmatrix},
$$

$$
\vec{\delta}'_Y = \begin{bmatrix} 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 \end{bmatrix}.
$$

ResNet Block



- Gradient vanishing problem.
- Allows gradient to be easily backpropagated to early layers.
- Feature reuse.

The Neural Network Architecture

JM3 [Yang+,JM3'17]					Ours				
Layer	Filter	Stride	Output	Parameter	Layer	Filter	Stride	Output	Parameter
conv1-1	3×3×4	2	160×160×4	36	conv1-1	5×5×128	2	32×32×128	9600
conv1-2	3×3×4	2	80×80×4	144	conv1-2	5×5×128	1	32×32×128	409600
conv2-1	3×3×8	1	80×80×8	288	conv1-3	5×5×128	1	32×32×128	409600
conv2-2	3×3×8	1	80×80×8	576	conv1-4	5×5×128	1	32×32×128	409600
conv2-3	3×3×8	1	80×80×8	576	conv2-1	5×5×256	2	16×16×256	819200
pool2	2×2	2	40×40×8		conv2-2	5×5×256	1	16×16×256	1638400
conv3-1	3×3×16	1	40×40×16	1152	conv2-3	5×5×256	1	16×16×256	1638400
conv3-2	3×3×16	1	40×40×16	2304	conv2-4	5×5×256	1	16×16×256	1638400
conv3-3	3×3×16	1	40×40×16	2304	conv3-1	5×5×512	2	8×8×512	3276800
pool3	2×2	2	20×20×16		conv3-2	5×5×512	1	8×8×512	6553600
conv4-1	3×3×32	1	20×20×32	4608	conv3-3	5×5×512	1	8×8×512	6553600
conv4-2	3×3×32	1	20×20×32	9216	conv3-4	5×5×512	1	8×8×512	6553600
conv4-3	3×3×32	1	20×20×32	9216	conv4-1	5×5×1024	2	4×4×1024	13107200
pool4	2×2	2	10×10×32						
conv5-1	3×3×32	1	10×10×32	9216					
conv5-2	3×3×32	1	10×10×32	9216					
conv5-3	3×3×32	1	10×10×32	9216					
pool5	2×2	2	5×5×32						
fc1			2048	1638400	fc1			1024	16777216
fc2			512	1048576	fc2			2	2048
fc3			2	1024					
Summary				2746068					59796864

Outline

The Dataset & Configurations

- 14nm metal layer, M3, V3, V4

	Train	Test	Image	Squish
Hotspot	3073	6015	320×320	64×64×3
Nonhotspot	973197	1457830		

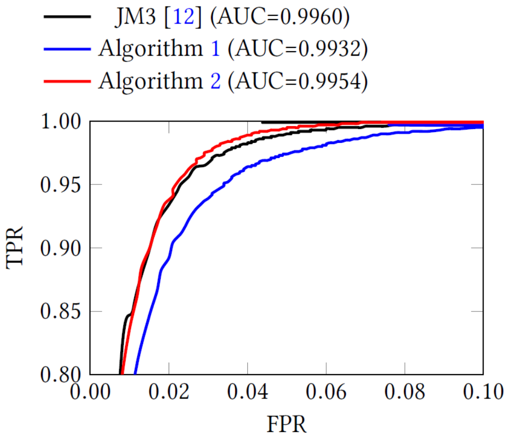
- Initial learning rate: 0.001
- Decay: 0.7 per 2000 steps
- Weight normalization: 0.001
- Xavier, Adam

Results

- Hit : # of hotspot patterns that are predicted as hotspots
- False Alarm : # of good patterns that are predicted as hotspots

Item	JM3 [Yang+,JM3'17]	Algorithm 1	Algorithm 2
Accuracy (%)	98.87	97.51	**99.24**
False Alarm Rate (%)	4.81	5.05	**4.52**
Hit	5947	5865	**5969**
False Alarm	70193	73645	**65926**
Precision (%)	7.81	7.38	**8.30**

Receiver Operating Characteristics



- JM3 behaves even better than Algorithm 2 in terms of area under curve.
- AUC advantages of JM3 comes from the region where the decision threshold is above 0.9.
- Higher confidence on hotspot patterns that can be correctly predicted by classifiers is not necessary.

Outline

Conclusion

- Adaptive Squish Pattern.
  Attains good properties of squish patterns and compatible with most learning machines.
- Multilayer Hotspot Detection.
  First time consider metal-to-via failure.
- ResNet.
  Allow better convergence and model generality.