Path Selection for Monitoring Unexpected Systematic Timing Effects

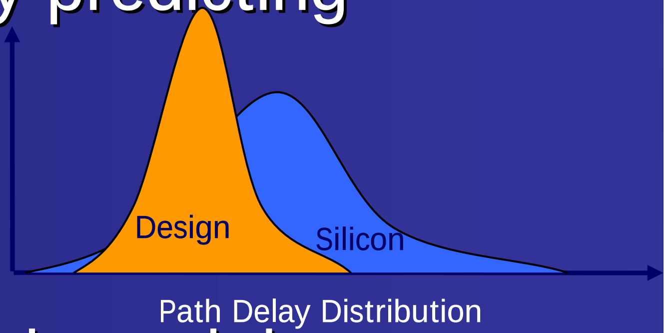
Nicholas Callegari, Pouria Bastani, Li-C. Wang
University of California - Santa Barbara

Sreejit Chakravarty, Alexander Tetelbaum
LSI Corporation

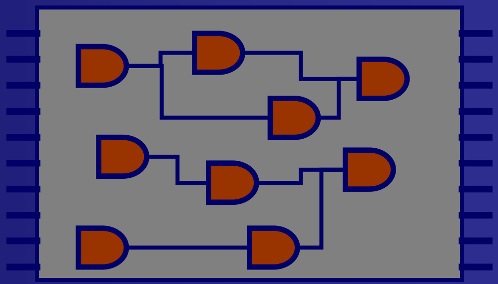
Jan. 22
ASPDAC 2009, Yokohama, Japan

Design-silicon mismatch

- Models and design methodology are constantly changing for accurately predicting silicon



- Current methodologies focus to;
  - Hypothesize potential causes of mismatch in silicon
    - Unexpected timing effects
  - Diagnose the inaccuracy in the model
  - Correct the model based on the diagnosis

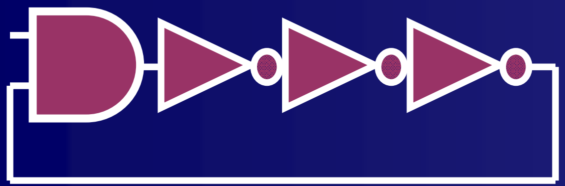


Design-silicon mismatch hypothesis

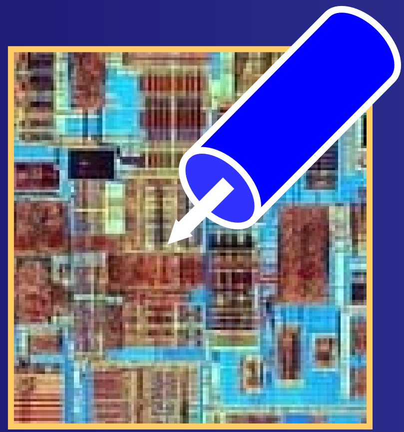
- Size and complexity of silicon today makes it impossible to test every source of unexpected timing effects
  - Needle in a haystack

- Designers identify high risk areas to test
  - Densely packed macro blocks
  - Long paths
  - Hot regions identified by models
- Time consuming, expensive, difficult to predict accurately

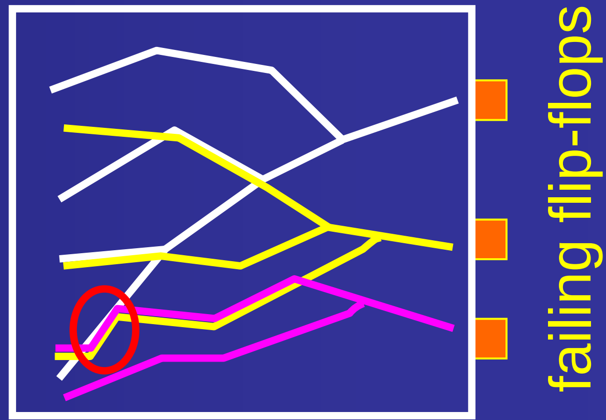
Traditional Approaches



Ring Oscillators



Physical debug



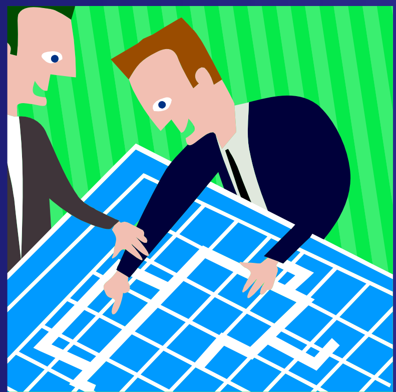
Location-based

- Hypothesize the causes and develop a methodology to check for the hypothesis
- Either it is high cost (manual effort, high cost equipment)
- Or only dealing with effects that are location based
- Need a flexible, low cost methodology to find unexpected timing effects on volume data efficiently

Selecting paths for delay test

- Traditional approach-
  – Obtain Static Timing Analysis (STA) report
  – Manually select critical paths based on design incite

STA Report

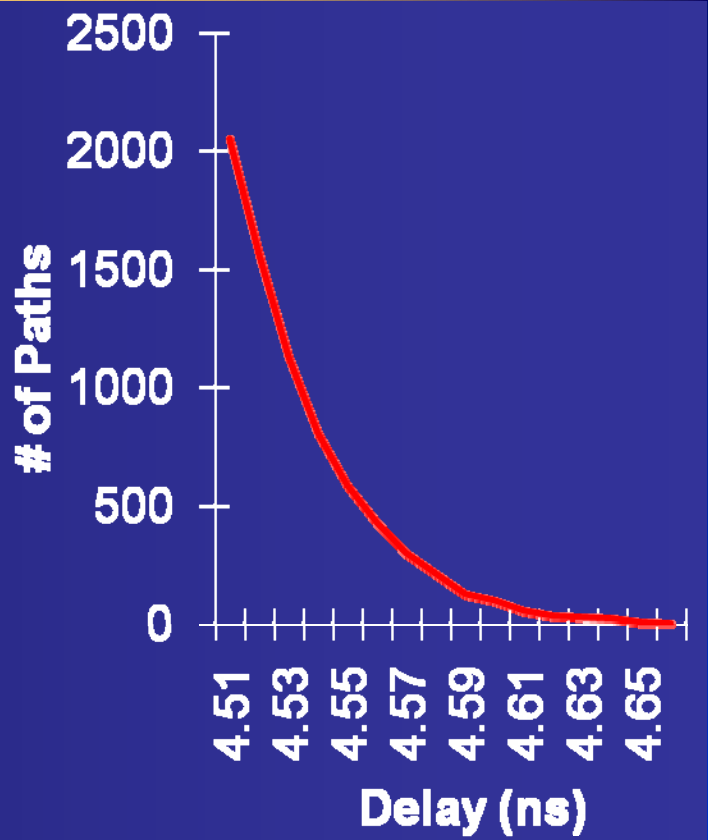


Designers

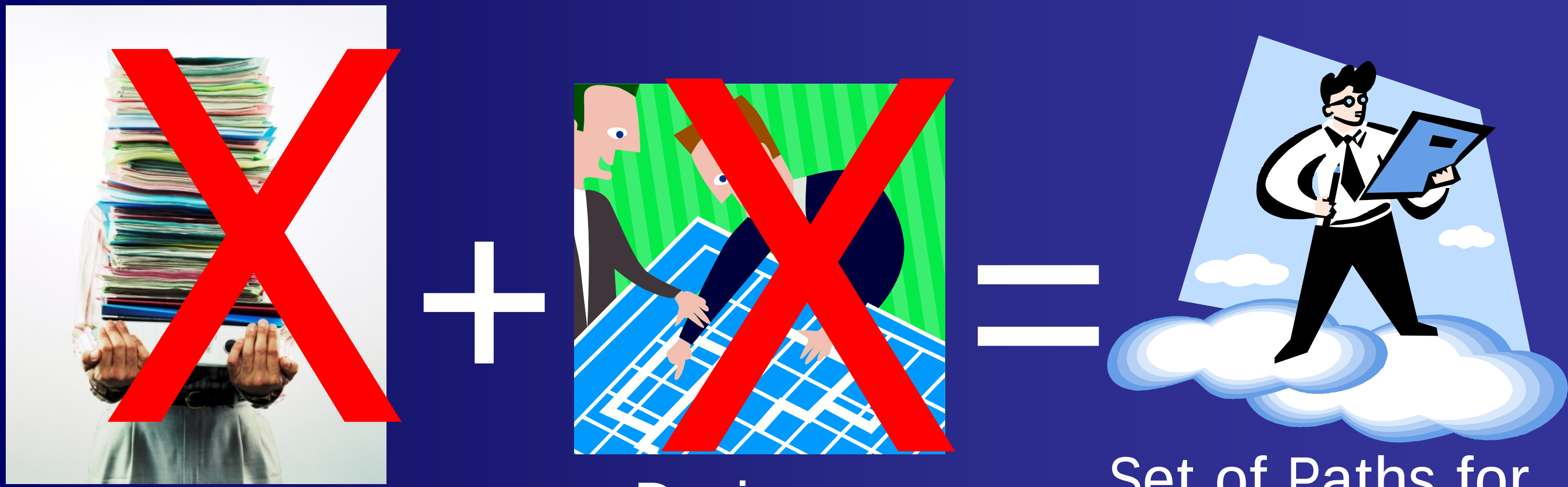
Set of Paths for Delay Test

Issues with traditional approach

- Number of paths
  - Although design dependent, the number of paths increases exponentially.
- Manual effort
  - Design engineers manually identify paths based on unexpected timing effects which is costly.
  - Path selection techniques are not consistent between different designer's.
- Limitations
  - Identification of such effects is only as good as the accuracy of the STA report, and knowledge of the engineers.



Selecting paths for delay test



HOW DO WE IDENTIFY A SET OF PATHS FOR DELAY TEST?

- Narrow range of critical paths from STA not sufficient due to timing mismatch.
- Manual identification is costly and only as accurate as designers domain knowledge

Reality

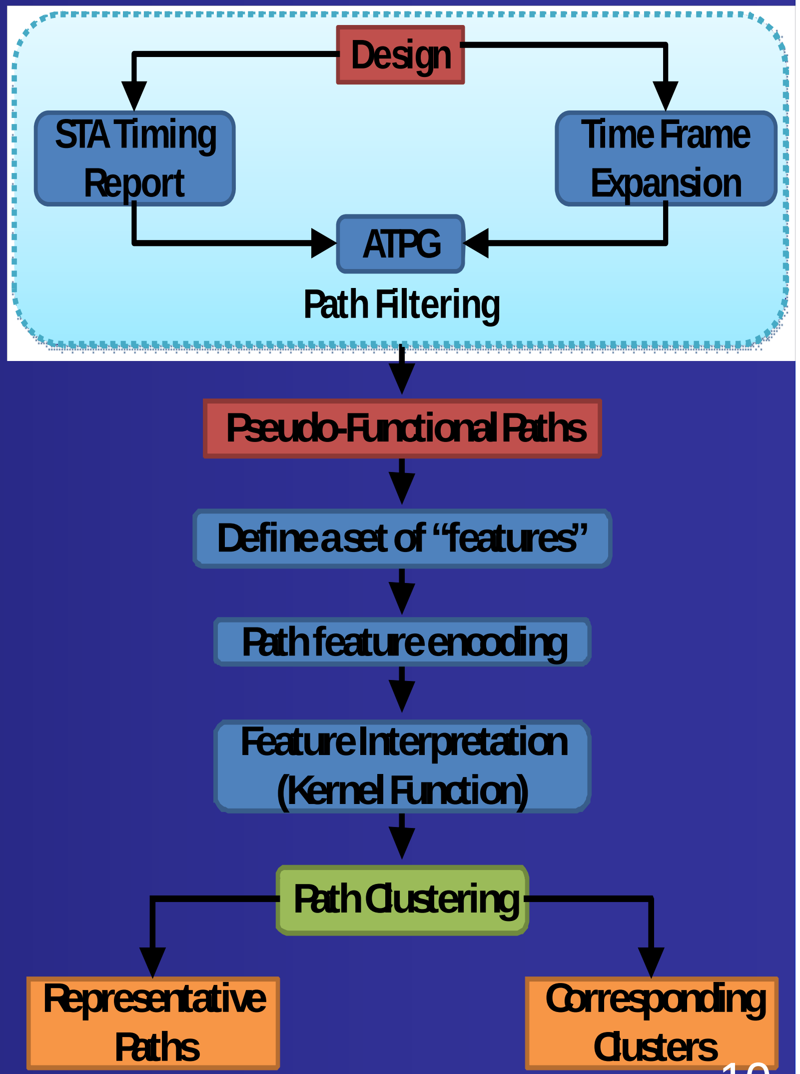
- STA is the best option we have
  - Need to utilize STA more effectively
    - Select a wider range to account for unexpected timing effects
- Designer’s knowledge is important
  - We just cant rely solely on designer’s knowledge
- WHAT ELSE DO WE NEED?
  - We need to examine the root of the cause of unexpected timing effects
  - We need to be able to efficiently and effectively identify paths to maximize observability

Path selection for monitoring unexpected systematic timing effects

- Instead of solving a diagnosis problem, we solve a path selection problem
- Goal: To develop a methodology to select an optimal set of paths to be measured for path delay
- Quantitative measurement of success: A path set is evaluated based on size and coverage of the space of unexpected timing effects

Methodology

- Design/netlist
- Path filtering obtains psuedo-functional paths
- Features encoding is applied
- Kernel modification is applied to properly interpret the features
- Path clustering is applied to produce representative paths



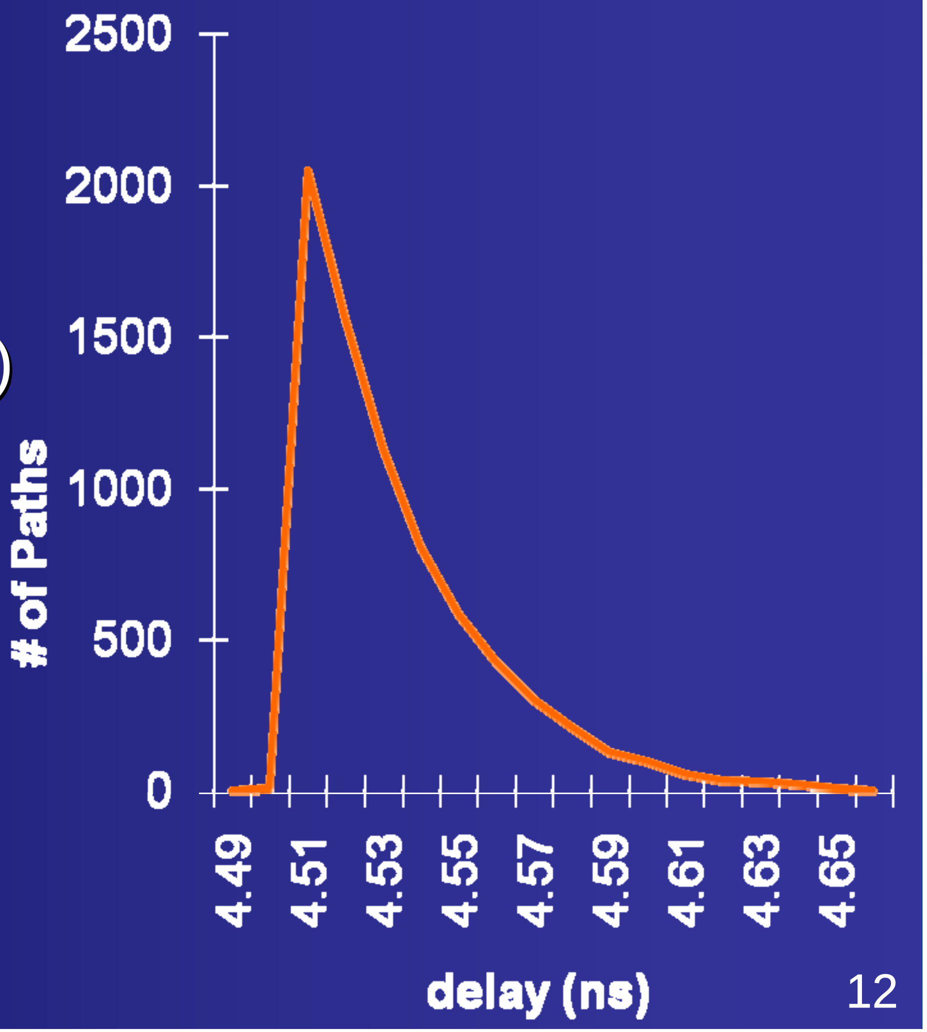
Identifying sensitizable critical paths

- Extract path delay timing report for most critical paths in Primetime
- Use Fastscan to identify which of the most critical paths are:
  - Sensitizable
    - Robust Testable (DR-det_robust)
    - Simulation Testable (DS-det_simulation)
    - Functional Testable (DF-det_functional)
  - Unsensitizable
    - ATPG_Untestable (AU-atpg_untestable)

Extract path delay information from STA

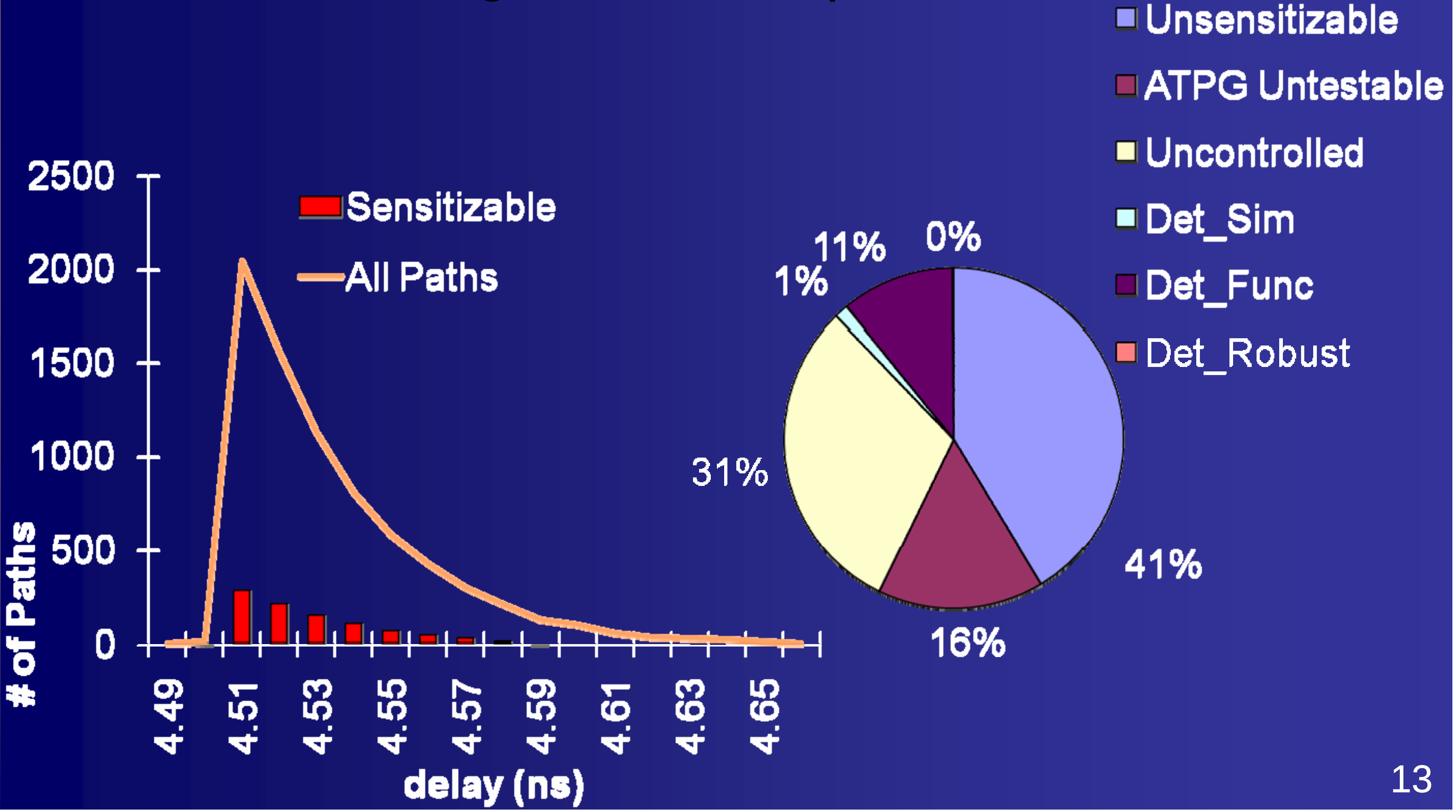
- A6K is a ASIC design with the following characteristics
  - ~6000 cells
  - ~7000 internal nets
  - ~100 flops
  - Maximum observed delay ~4.65ns (215mhz)

- Extract the top 7,400 most critical paths



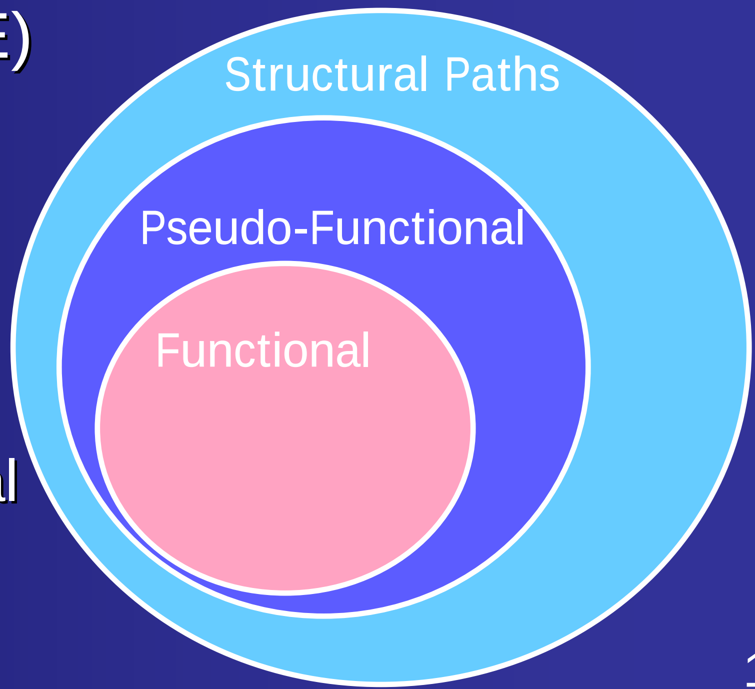
Sensitizable paths

- The new delay test set was passed into Fastscan, and the resulting sensitizable paths were identified



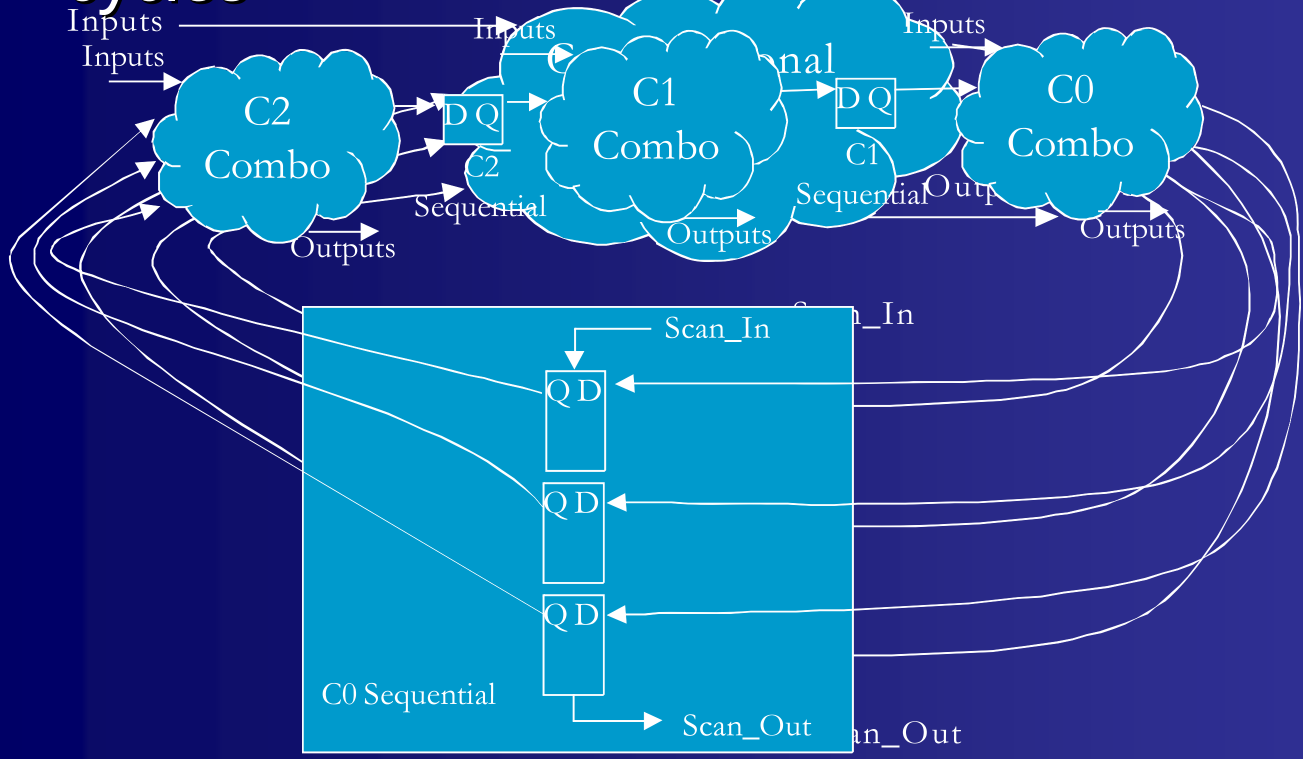
Pseudo-functional path identification

- Goal: Identify functional paths from a set of structural paths
- Reality: Currently there is no automated way to identify functional paths from structural
- Next best thing:
  - Time Frame Expansion (TFE)
    - Unroll the combinational circuit to simulate multiple clock cycles
    - Guarantee no functional paths are removed
    - Reduce the set of structural paths.



Time Frame Expansion

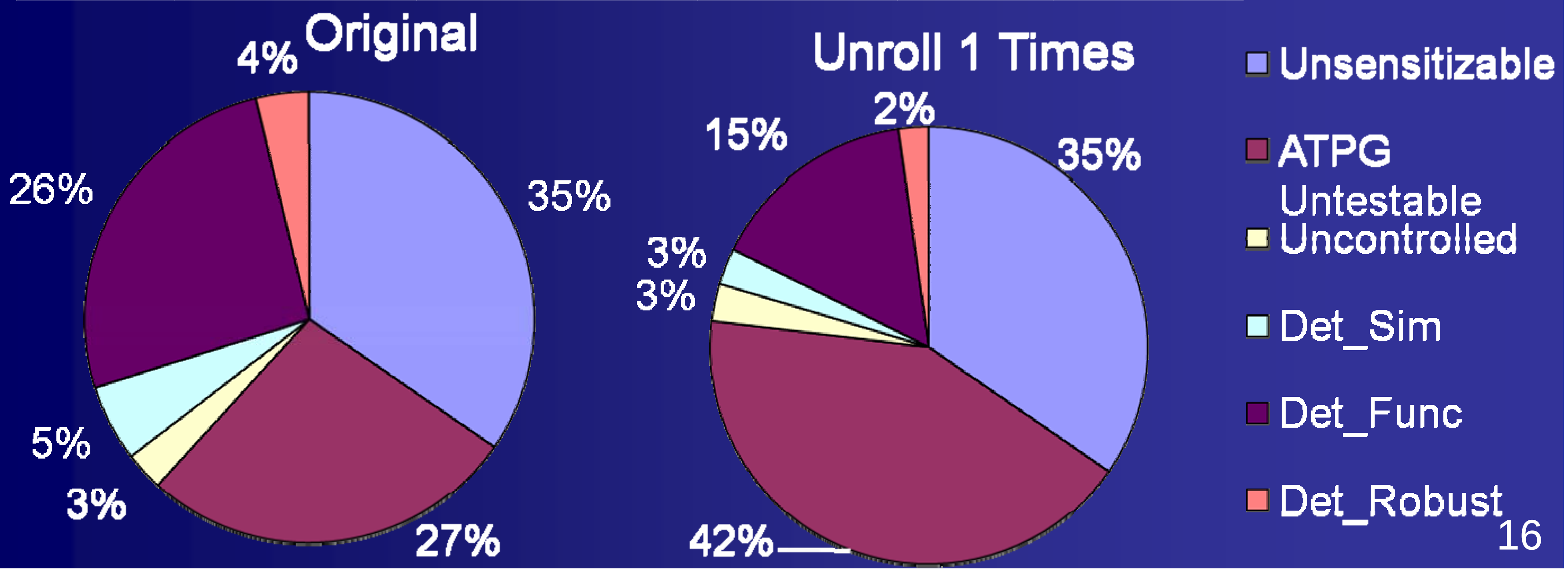
- In Fastscan TFE is accomplished by taking the original circuit and duplicating the combinational
- Careful attention is needed to properly connect the duplicates so the test pattern propagates through cycles



Results for pseudo-functional

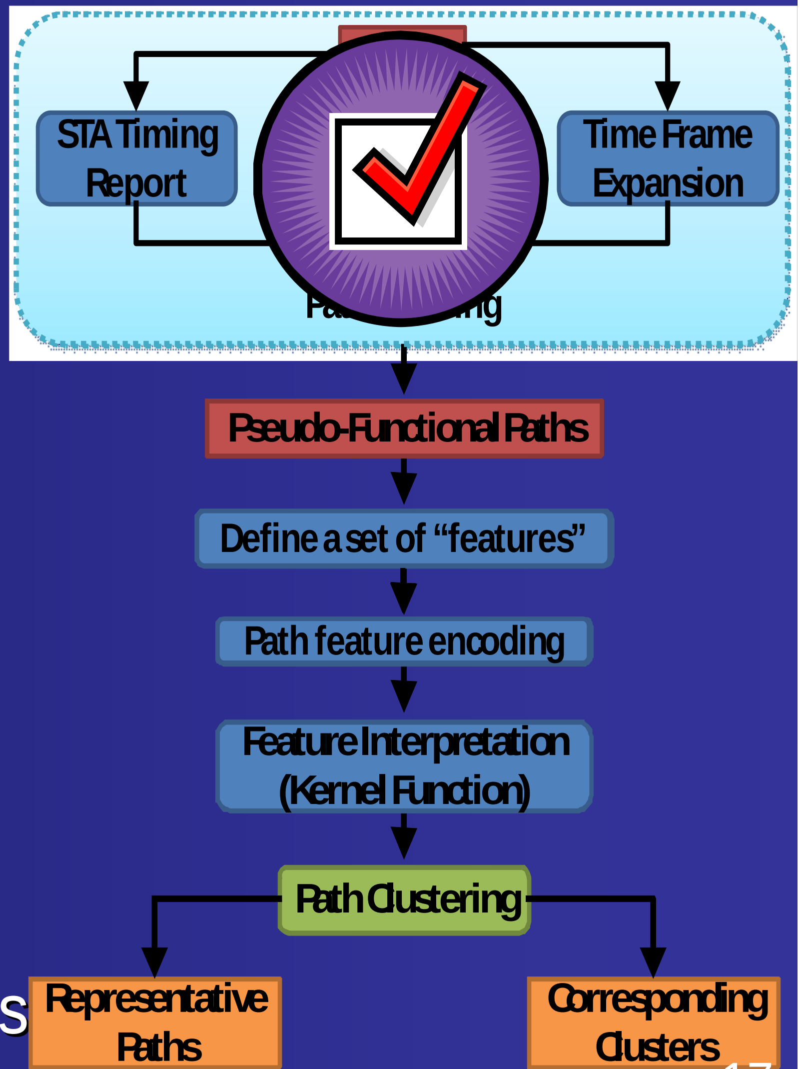
- Results....
  - Sensitizable paths (Total Det) reduced 40%

Circuit	Unsensitizable	ATPG Untestable	Uncontrolled	Total Det	Total Paths
Original	52434	41015	4215	53667	151331
Unroll_1	52434	63955	4195	30747	151331



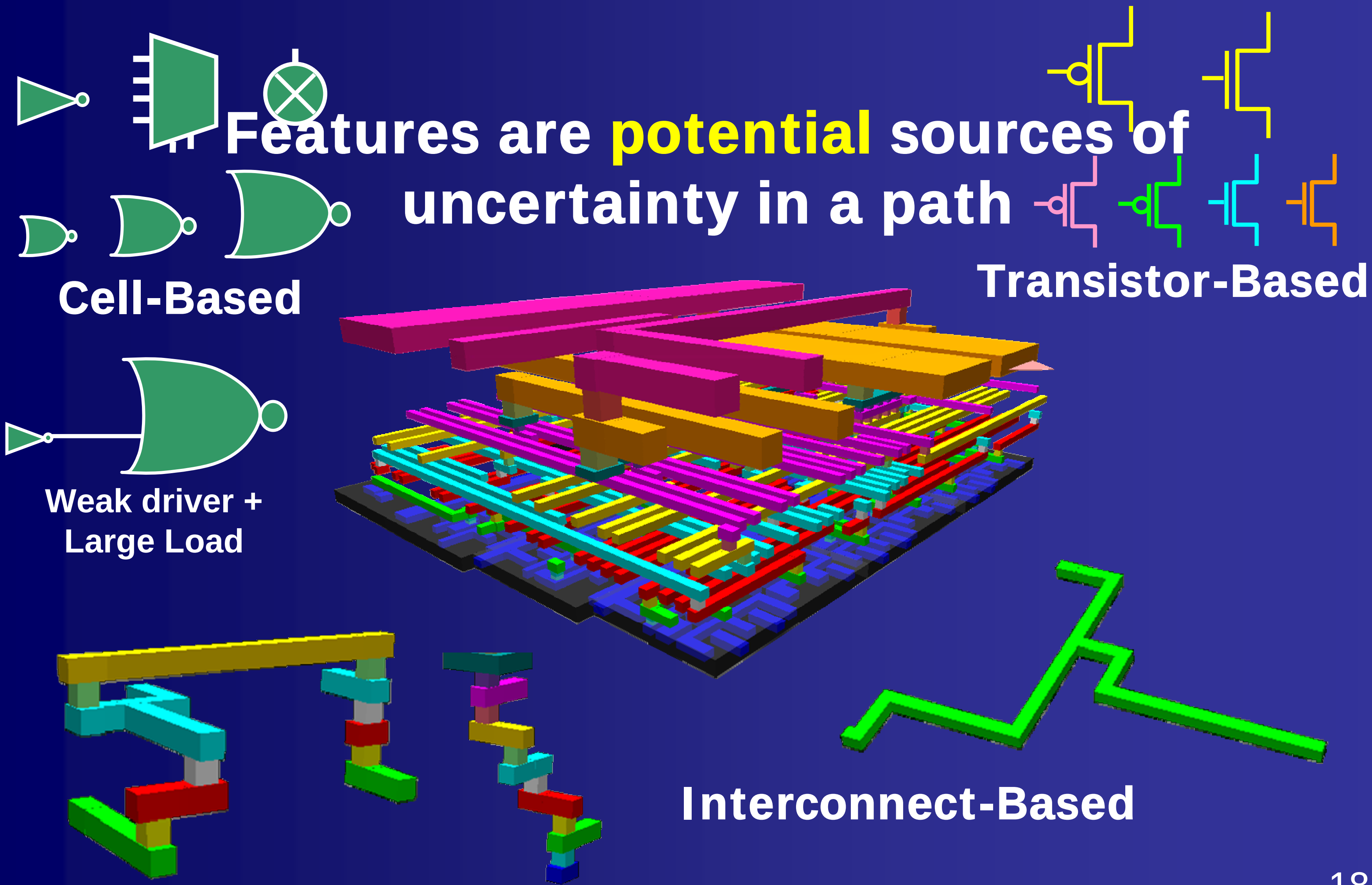
Methodology

- Design/netlist
- Path filtering obtains psuedo-functional paths
- Features encoding is applied
- Kernel modification is applied to properly interpret the features
- Path clustering is applied to produce a representative paths

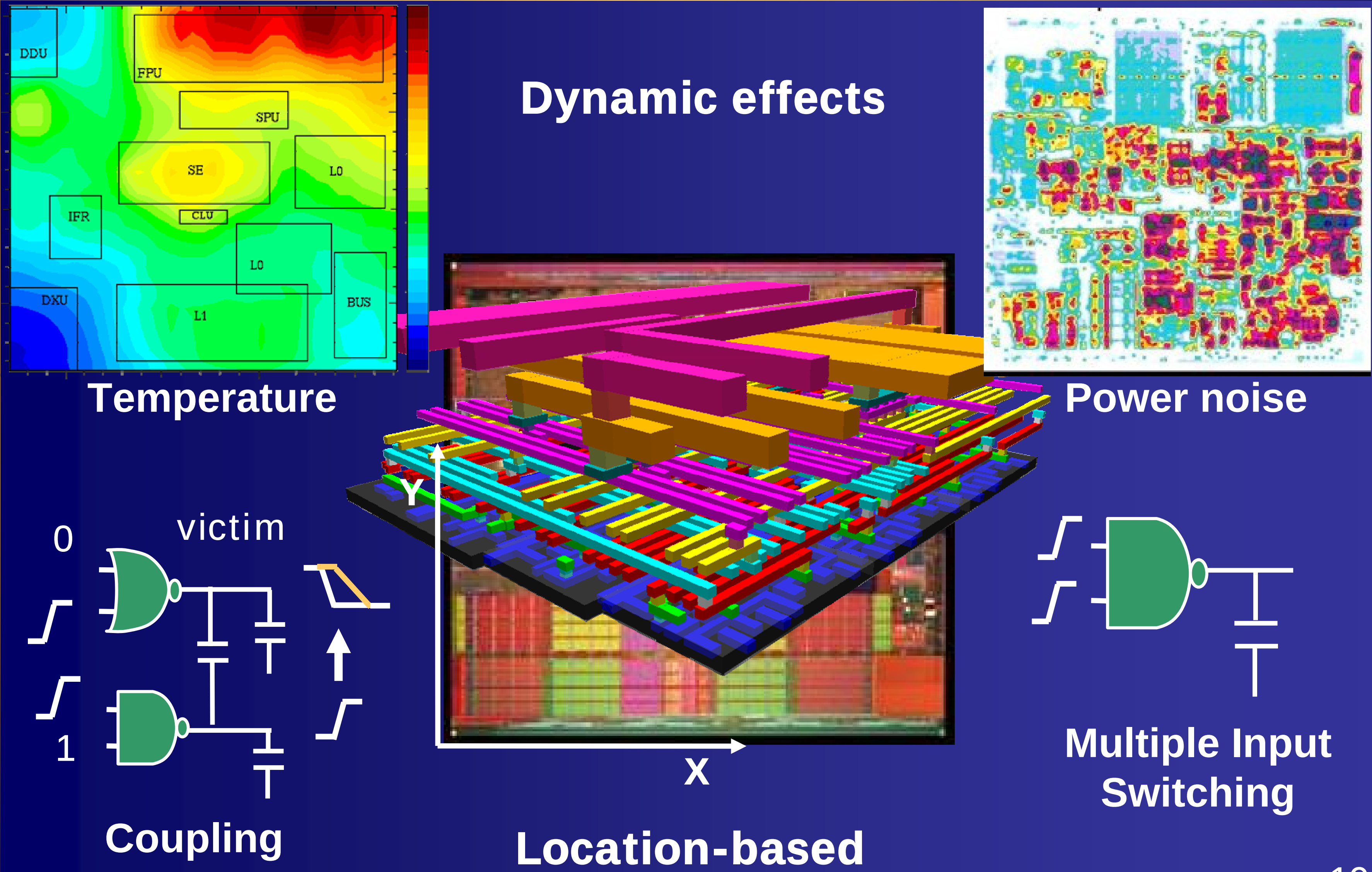


Feature Generation

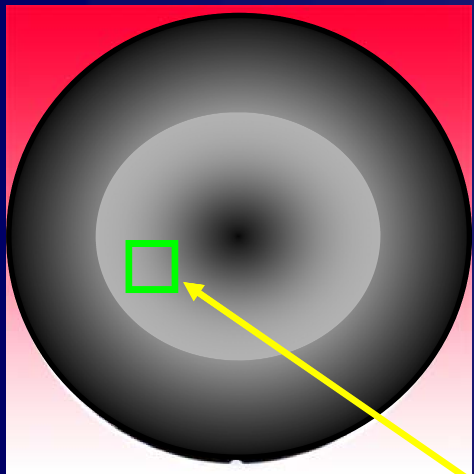
Features are potential sources of uncertainty in a path



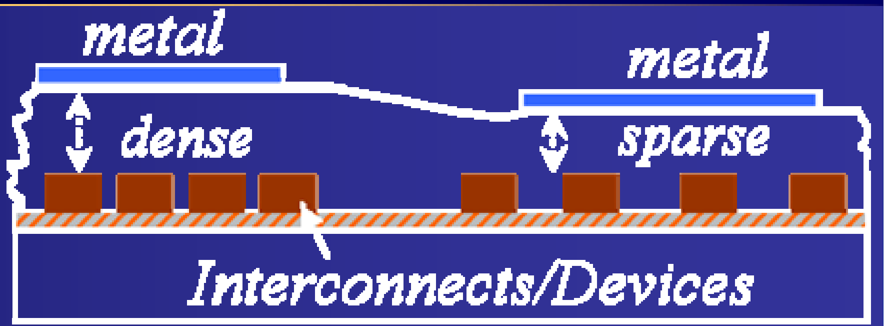
Feature Generation Continued



Feature Generation Continued

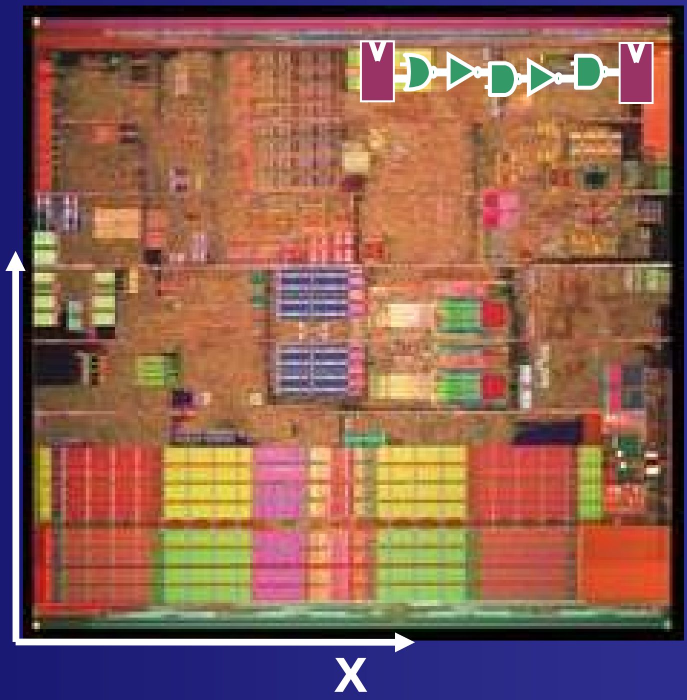


Die-Index



Path Area

Inter-die vs Intra-die



Location-based

Feature Encoding

- m paths are encoded based on potential sources of unexpected timing effects defined by n features.
- Features are provided by the design engineer's knowledge, they can be any source of uncertainty, any possible unexpected timing effect

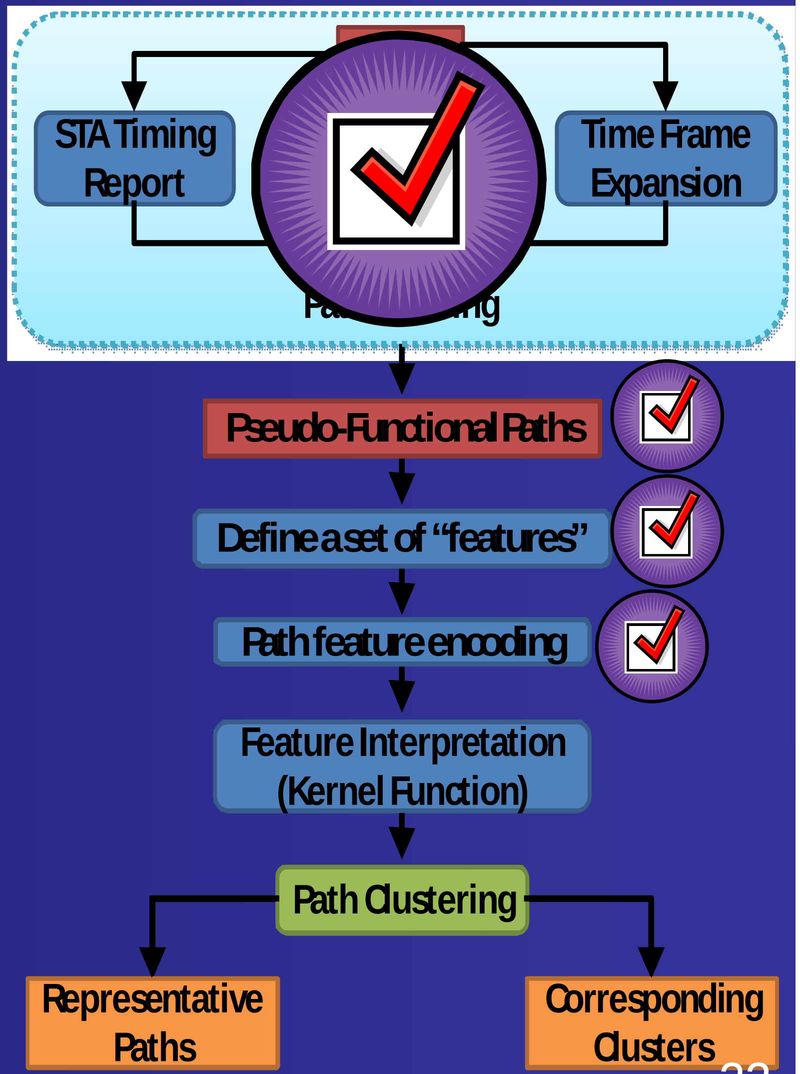
Path Features

$$
\begin{array}{c|cccc|}
 & f_1 & f_2 & \dots & f_n \\
\hline
p_1 & x_{11} & x_{12} & \dots & x_{1n} \\
p_2 & x_{21} & x_{22} & \dots & x_{2n} \\
\vdots & & \vdots & & \\
p_m & x_{m1} & x_{m2} & \dots & x_{mn}
\end{array}
$$

***n* feature definitions**

Methodology

- Design/netlist
- Path filtering obtains psuedo-functional paths
- Features encoding is applied
- Kernel modification is applied to properly interpret the features
- Path clustering is applied to produce representative paths

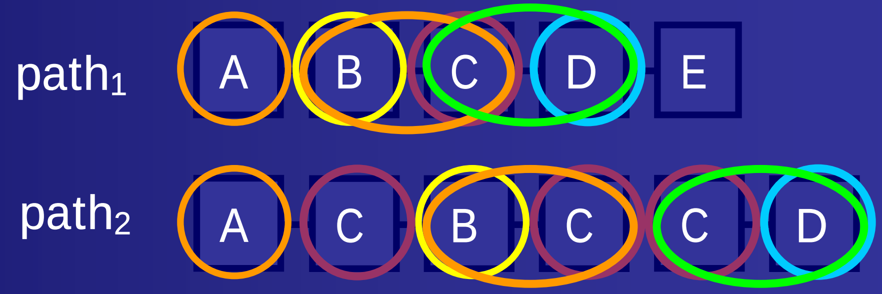


Kernel function

- How do we properly interpret different feature values?
- We want to identify a kernel function that can
  - Properly assess similarity based on different features
  - Take into account high order effects
    - Hypothesis: Unexpected timing effects can be due to a single cell, first order effects, or a combination of *j* cells connected in a in a certain order, high order effects
- Because features are paths based we want to identify a kernel function that takes into account feature ordering

p-Spectrum Kernel

- *p*-Spectrum Kernel – for cell ordering
  - Analyzed how many contiguous sub-paths of length *p* two paths have in common



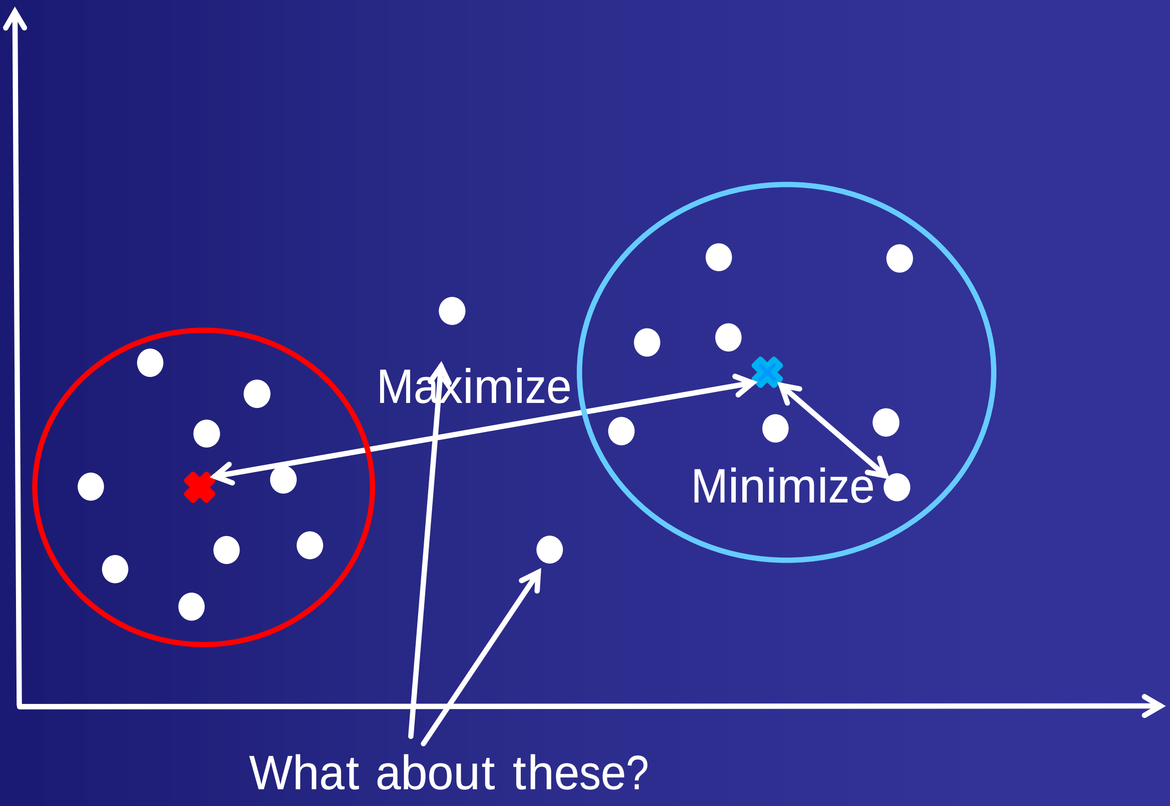
- Example:
  - $path_1$ = [ABCDE] , $path_2$ = [ACBCCD]
  - p =1: $K(p_1,p_2)$ = [A,B,C,D] = 4
  - p =2: $K(p_1,p_2)$ = [BC,CD] = 2
- The more *p* sub-paths the higher the similarity

Clustering attributes

- GOAL: Select a reasonable set of representative paths that provides the best coverage of features, ie. unexpected timing effects
- Clustering Attributes
  – Classify paths into different groups, so that each subset share common features
  – Identify the most centriod vector, or path, to use as a representative for each group
  – Utilize the kernel function to properly identify feature similarity
  – Ability to weigh features based on their assumed importance

Clustering

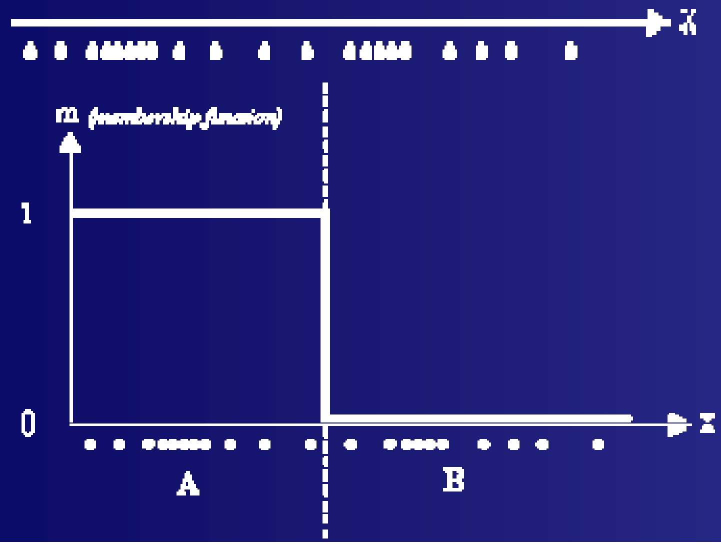
- Objective
  - Maxamize inter-cluster variance
  - Minimize intra-cluster variance

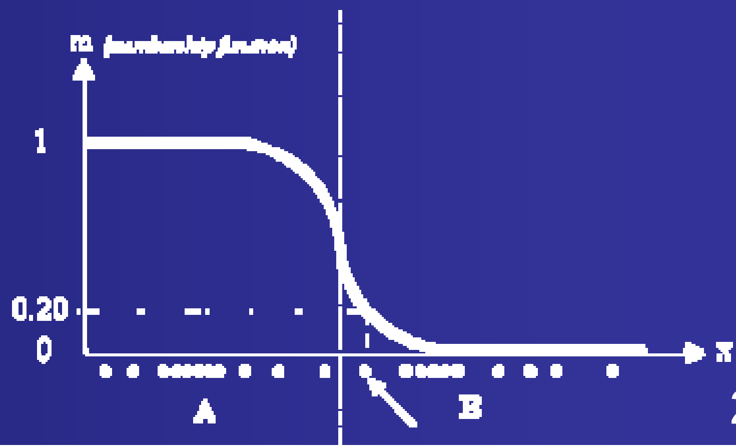


Fuzzy c-means clustering

- Our methodology incorporates fuzzy logic in which each path has a degree of belonging to each cluster
- Objective:
  - Maximize inter-cluster variance
  - Minimize the intra-cluster variance
- Objective function

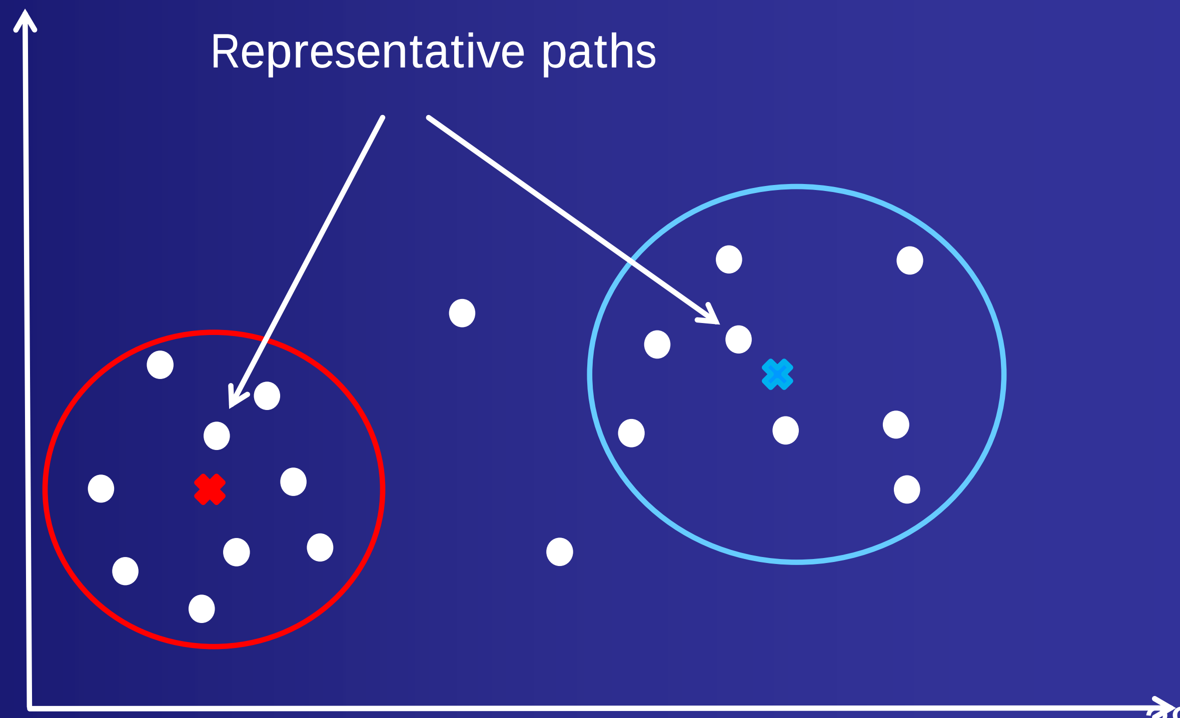
$$J_m(U,V) = \sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik}^m \times dist(x_k, v_i)$$





Path Selection

- Once the clusters are determined, the closest point to the centroid best represents the paths within the cluster.



Complete Flow Experiment

- A6K is a ASIC design with the following characteristics
  - ~6000 Cells
  - ~7000 Internal Nets
  - ~100 Flops
- Ran path filtering steps and obtained the following results
- Due to minimal improvement from 5-cycle TFE, we continue the flow with 1567 paths from 1-cycle TFE

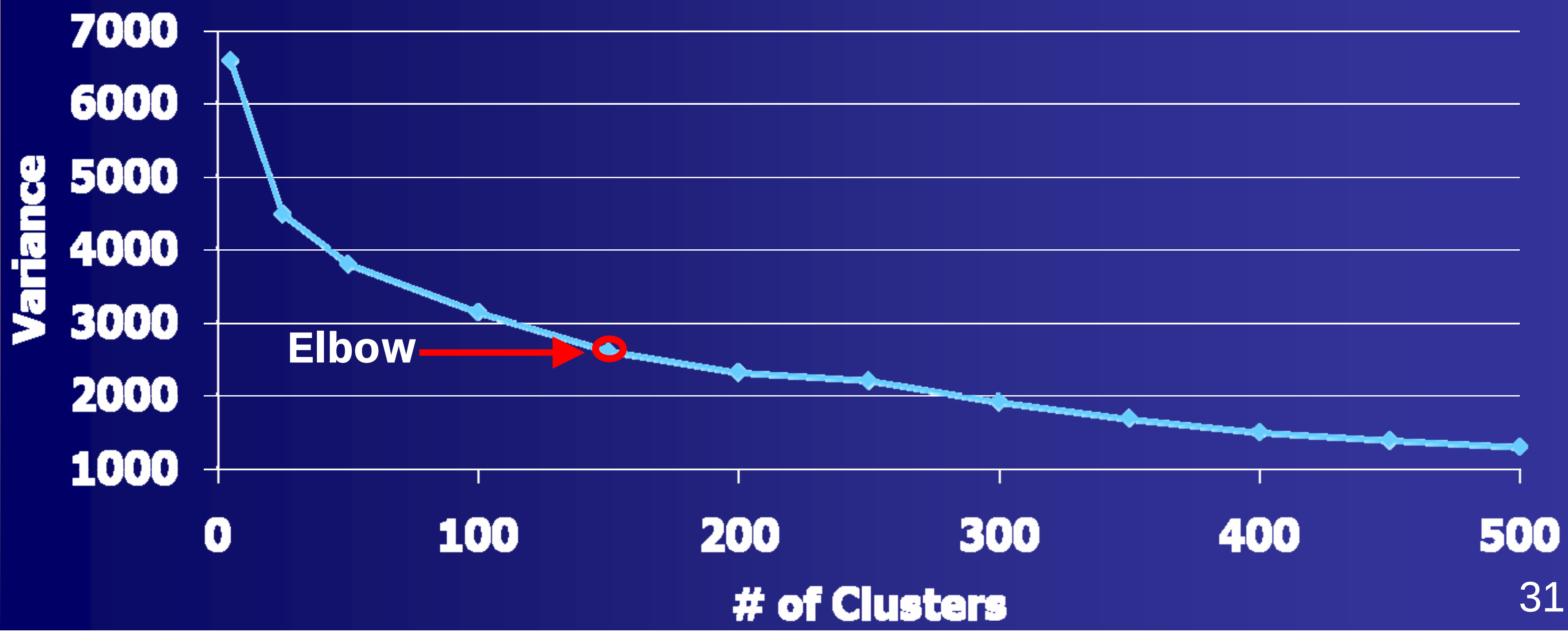
Path Filter Results			
Total Paths	0-cycles	1-cycles	5-cycles
8332	3029	1567	1540
% Reduction	63.65%	81.19%	81.51%

Selecting the number of clusters

- A difficulty accompanied with clustering algorithms is selecting the optimal number of clusters.
  - Too few clusters may not cover all potential unexpected effects
  - Too many clusters may exceed the limit on the number of test patterns
- To obtain the optimal number of clusters we consider:
  - Objective function to minimize intra-cluster variance
  - Quantitative measurement of success, feature coverage

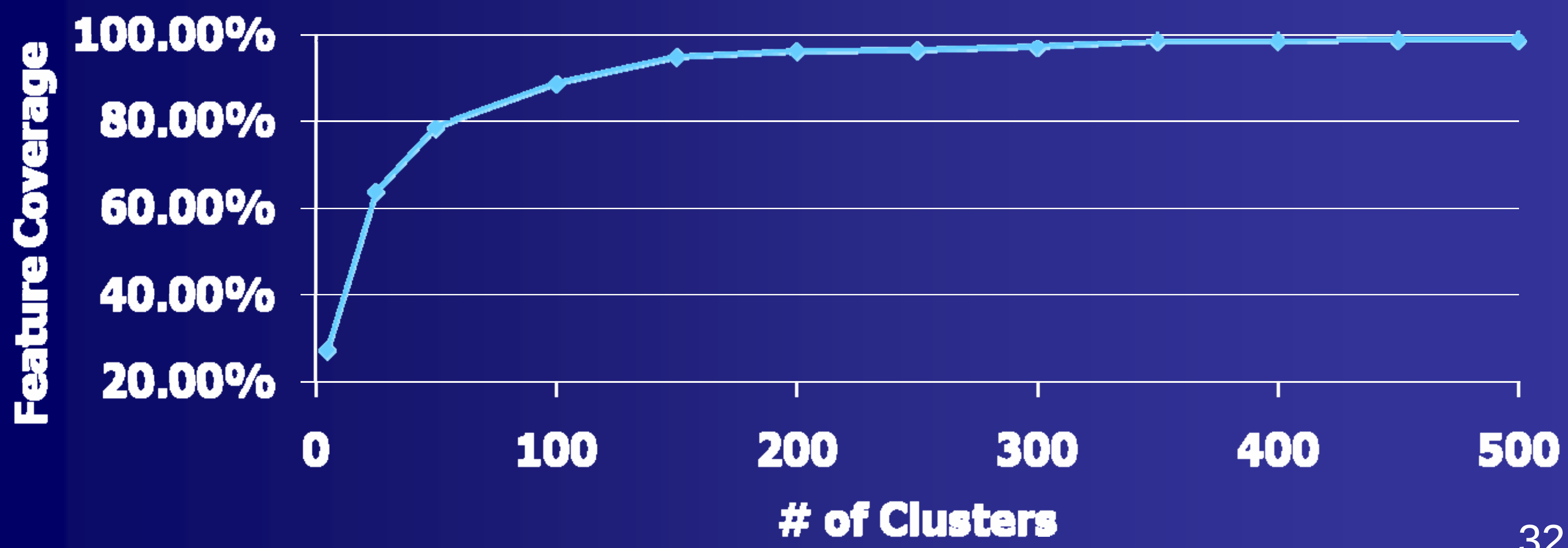
Elbow Criterion

- Selecting the number of clusters in a manner that adding another cluster does not add sufficient information, ie. explain variance.
- Law of diminishing returns, select a optimal number of clusters in a reasonable range



Feature coverage criterion

- Using the quantitative measurement we can analyze our previous selection based on variance
- 150 clusters proves to be a reasonable number of paths to select given our quantitative unit of measure.



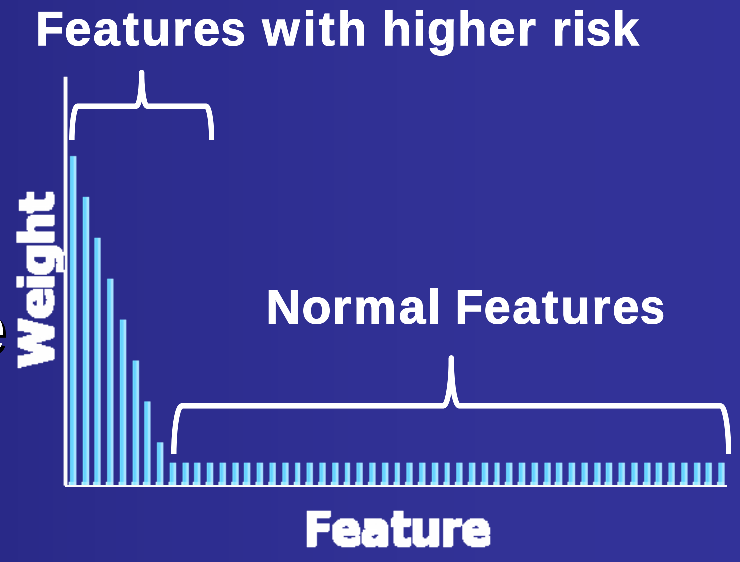
Results

- Obtained the original 1567 pseudo-functional paths
- Test set
  - 150 paths from clustering
  - Randomly select 150 paths
  - Take the top 150 critical paths
- Injected 100 randomly formed timing errors and compared coverage

Error Injection Coverage			Error
Method	**Paths**	**Error Injected**	**Covered**
Clustering	**150**	**100**	**94.80%**
Random	**150**	**100**	**78.26%**
Top Critical	**150**	**100**	**32.00%**

Results Continued

- Selected 10% of the features
- Weighed them based on hypothetical risk
- Injected 10% noise on the distribution and used it as the probability for the randomly formed timing errors



Weighted Error Injection Coverage			
Method	Paths	Error Injected	Error Covered
Clustering	150	100	98.48%
Random	150	100	56.66%
Top Critical	150	100	23.00%

- Thank You