Abstract—This work proposes a knowledge discovery process for analyzing design and test data, enabling the extraction of interpretable and actionable knowledge beginning with raw test measurements and ending with valuable insight. A variety of preprocessing, data mining, and visualization methods are used to build a unified knowledge discovery framework with applications in test. Silicon measurements and design data from a recent high-performance dual-core SOC are analyzed to evaluate the products timing analysis flow and explain unexpected delay test results. Through extensive analysis we develop a knowledge discovery process that generates human interpretable rules, subgroups, and models explaining complex failure mechanisms.

I. Introduction

As integrated circuit technology continues to scale, new and complex failure mechanisms occur forcing test to evolve from traditional pass/fail screening to adaptive data-driven methods capable of diagnosing and predicting failures, modeling highly nonlinear processes, and coping with variability [1]. Detecting and understanding unexpected test results associated with issues such as power droop, parametric shifts, systematic effects, and various design decisions requires a holistic perspective for informed decision making. An effective diagnostic paradigm should leverage measured test data to uncover valuable knowledge that can explain results and correct issues. Such an approach requires more than just applying a single data mining method to uncover statistically significant findings in test data. A collection of data mining methods must be integrated to build a flow that enables efficient search for interpretable and actionable knowledge explaining test data. This integrated approach is called a knowledge discovery process (KDP) [2].

Figure 1 compares the algorithm-centric data mining perspective to the data-centric KDP perspective. Given the algorithm-centric perspective, one is concerned with finding the best data mining algorithm to extract the most statistically significant results from test data. This perspective, while important, is insufficient for implementing a practical flow. In practice, statistically significant findings do not necessarily provide actionable or interpretable knowledge. If they cannot be interpreted and justified, or they do not lead to feasible actions, such findings become meaningless.

In practice, a successful data mining task not only depends on the mining algorithm used, but also on the information content within the data as well as the formulation of the target question being asked. This data-question-algorithm interdependence is crucial in the following ways:

- **Data-Question**: The data needs to contain sufficient information to answer the question. The question formulation depends on the information content within the data.
- **Question-Algorithm**: The question formulation needs to match the capability of the data mining algorithm.
- **Data-Algorithm**: The data must be in a form suitable for the algorithm. For example, if a binary classification algorithm is chosen, data samples need to be divided into two classes. How the decision is made to split the data crucially impacts the effectiveness of the algorithm.

Because of this interdependence, a data mining task usually cannot be accomplished through a single data mining step as shown in the algorithm-centric perspective. Instead, it has to be an iterative data-centric process as shown in Figure 1, where in each step, the data-question-algorithm interdependence must be considered carefully.

The data-algorithm aspect alone had been studied in the past few years with different applications. For example, the work in [3] proposed a hypothesis pruning and ranking algorithm to analyze silicon speed paths and non-speed paths. The work in [4] used the one-class support vector machine algorithm to rank and predict speed paths. Outlier analysis algorithms have been explored to improve delay test quality in [5] and [6]. Various learning algorithms were studied to correlate structural...
Fmax to functional or system Fmax in [7][8]. Feature ranking methods for diagnosing design-silicon timing mismatch were described in [9]. A survey of data mining algorithms in the context of diagnosis can be found in [10].

This work is an attempt to implement a practical knowledge discovery process taking into account the data-question-algorithm interdependence. Our objectives are twofold:

1) Evaluate a collection of data mining methods and algorithms to determine feasibility and applicability in each step of the iterative knowledge discovery process.

2) Prototype a general knowledge discovery process that can be used in various applications.

In this work, the application focuses on understanding unexpected test results from path delay measurements. We analyze test measurements and design data from a recent high-performance dual-core SOC to evaluate the timing analysis flow. The application serves as an example to further develop the general knowledge discovery process. This paper documents our experience and describes how interpretable and actionable knowledge was discovered through the process.

The rest of the paper is organized as follows: Section II discusses related works and the choice of various data mining methods and algorithms. Section III describes the general knowledge discovery process and how it can be applied to mine test data. Section IV explains the implementation specifics for the target application. Section V discusses and evaluates the extracted knowledge and Section VI concludes.

II. RELATED WORKS AND DATA MINING ALGORITHMS

A variety of data mining algorithms are available for extracting useful information in the form of rules, ranks, clusters, or subgroups. Table I illustrates the multitude of popular algorithms currently available. Given a particular test data mining task, choosing a suitable algorithm is nontrivial. Most work thus far has avoided this problem by assuming a specific data format and problem formulation limiting the application scope to one algorithm.

For example, authors in [9] propose diagnosing design-silicon mismatch by ranking path specific design features such as cell type, interconnect type, physical location, etc. Support vector machine (SVM) regression was employed to compute a rank of the most important features contributing to design-silicon mismatch. While the results are promising, given another test data-mining task requiring the use of feature ranking, it is unclear how alternative ranking methods such as Gaussian process regression [11], chi-squared test [12], or forward/backward selection [13] would perform compared to SVM regression. Among these methods, chi-squared test has the lowest computational cost while forward/backward selection has the highest.

Furthest neighbor clustering is used in [14] to uncover systematic failures grouped by similarly failing latch signatures captured from scan test results. Similarly, for another task involving data clustering, furthest neighbor methods may not be the most effective, and one may want to consider other algorithms such as k-means clustering [12], SVM clustering [15], or association rule learning [16].

Power gating optimization is performed in [17] where the extension matrix approach is used to discover circuit behavior invariants from power benchmarks to better gate inactive blocks. In such an application, subgroup discovery [18] or association rule learning are feasible alternatives and may provide additional perspective.

Optimizing yield, reducing cost, and increasing device performance is proposed in [19] where the popular classification and regression tree algorithm (CART) [20] is used to detect anomalies during manufacturing. The system is currently installed at the IBM 300 mm fab and claims to save millions of dollars by detecting opportunities for yield and performance improvement. One may wonder if SVM classification [21], or outlier analysis methods [22][23][24] can provide other useful solutions.

Authors in [25] extract cross-domain effects responsible for design-silicon mismatch using subgroup discovery algorithms; however, viable alternatives such as association rule learning [16], and decision tree classification [20] were not considered.

The above works are similar in the sense that they apply one particular mining algorithm suited to the problem at hand; however, as Table I shows many alternatives exist. Suppose one is given test data in a raw format and asks a high-level question to motivate test data mining. Reaching the point of confidently knowing which mining algorithm(s) to apply is nontrivial. This is because such a choice depends on the specific question formulation, the data to be analyzed, prior domain knowledge, and final objectives.

<table>
<thead>
<tr>
<th>Task</th>
<th>Algorithm</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification</td>
<td>NU-SVC</td>
<td>[21]</td>
</tr>
<tr>
<td></td>
<td>C4.5</td>
<td>[28]</td>
</tr>
<tr>
<td></td>
<td>CART</td>
<td>[20]</td>
</tr>
<tr>
<td>Regression</td>
<td>NU-SVR</td>
<td>[21]</td>
</tr>
<tr>
<td></td>
<td>GPR</td>
<td>[11]</td>
</tr>
<tr>
<td></td>
<td>CART</td>
<td>[20]</td>
</tr>
<tr>
<td>Transformation</td>
<td>PCA</td>
<td>[26]</td>
</tr>
<tr>
<td></td>
<td>LDA</td>
<td>[2]</td>
</tr>
<tr>
<td></td>
<td>SVD</td>
<td>[13]</td>
</tr>
<tr>
<td>Feature Ranking</td>
<td>Chi-Square</td>
<td>[12]</td>
</tr>
<tr>
<td></td>
<td>SVM</td>
<td>[21]</td>
</tr>
<tr>
<td></td>
<td>GPR</td>
<td>[11]</td>
</tr>
<tr>
<td>Outlier Analysis</td>
<td>One-Class SVM</td>
<td>[22]</td>
</tr>
<tr>
<td></td>
<td>KNN</td>
<td>[23]</td>
</tr>
<tr>
<td></td>
<td>LOF</td>
<td>[24]</td>
</tr>
<tr>
<td>Clustering</td>
<td>K-Means</td>
<td>[12]</td>
</tr>
<tr>
<td></td>
<td>Hierarchical</td>
<td>[12]</td>
</tr>
<tr>
<td></td>
<td>SVC</td>
<td>[15]</td>
</tr>
<tr>
<td>Rule Learning</td>
<td>Rule Induction</td>
<td>[20]</td>
</tr>
<tr>
<td></td>
<td>CN2-SD</td>
<td>[18]</td>
</tr>
<tr>
<td></td>
<td>FP-Growth</td>
<td>[16]</td>
</tr>
</tbody>
</table>

Table I: Overview of Data Mining Algorithms
III. KNOWLEDGE DISCOVERY IN TEST DATA

In the literature, knowledge discovery (KD) is defined as “the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data” [2]. In this work, we split the KD process into five iterative steps:

1) Understanding the problem (question formulation)
2) Database creation (including both design and test data)
3) Data mining (integrating multiple mining algorithms)
4) Evaluation of discovered results to distill knowledge
5) Application of discovered knowledge

The KD process is iterative because knowledge discovered in steps (3)-(5) may lead to a better understanding of the problem. For example, discovering a more effective perspective to approach test data analysis. Different questions may also be formulated, which can result in collecting additional test measurements, creating a different database, or applying a different set of mining algorithms.

This section explains each step in the KD process, introduces several relevant algorithms, and gives specific examples in the context of evaluating a timing analysis flow beginning with silicon path delay measurements.

A. Understanding the Problem

A typical test data mining task starts with raw test data and a high-level question. For example, the raw data can consist of all failing flipflop-pattern pairs based on a test pattern set and a collection of sample chips. The question can be: “What design changes will minimize the observed failures?”.

In a better planned scenario, test data could be collected to answer a particular question. For example, the question could ask how well a timing analysis flow captures silicon reality. To answer this question, thousands of critical paths are measured across a large set of sample chips.

The first step towards knowledge discovery is refining the high-level question into specific inquiries(s) that have the potential to be answered with test data. This step involves understanding the problem domain and the types of design and test data available for analysis. Understanding the problem domain means evaluating measured test results, formulating key question(s), and deciding on an initial set of algorithms to answer those question(s). Managing the interdependence between data, questions, and algorithms is essential for analysis.

Although this step is critical, and impacts the effectiveness of all subsequent steps, it is often overlooked in practice.

Consider applying knowledge discovery to understand path delay measurements for a set of most critical paths selected by static timing analysis (STA). Specifically we have, measured slack from automatic test equipment (ATE) and expected slack from STA. Given these two types of slack data, one often compares their distributions to estimate how accurate STA is relative to silicon measurements. A scatter plot is an effective way to chart measured versus expected slack. Figure 2 shows an example scatter plot where paths above the diagonal line have more than expected slack, and paths below the diagonal have less than expected slack. Each dot in the figure represents one path, and measured slack is the average taken over 30 dies, captured with 1 ps frequency stepping.

Observing a plot like Figure 2, immediately raises questions concerning the particular corner at which STA was run. In this experiment, the nominal corner was used. Although, running the fast corner would have shifted the scatter plot up and running the slow corner would have shifted it down.

One may ask why a large number of paths are underestimated when using the diagonal line as reference. First of all, there can be multiple reasons behind this result. More importantly, these reasons can be sensitive to the nominal corner used to obtain the diagonal line. Because the question is specific to a particular corner, it can be hard to answer.

If one ignores the diagonal line and focuses on the paths falling into two clusters, then the question becomes why one group of paths is more under-estimated than the other. In this view, the position of scatter points relative to the diagonal line is not as important as the relative spacing between them. Such a perspective is more desirable since it leads to results that are less dependent on the particular timing corner.

Another question that can be asked is: “why are a few paths far more under-estimated than the rest?” (those close to the bottom-right corner in Figure 2). This is different from asking the two-cluster question above. In the two-cluster case, both groups contain a large number of paths. When asking why a few paths are special, we view the data as one large cluster and the few remaining paths as outliers. The data mining methods used to answer these two types of questions are different [10].

In this work we explain the two cluster trend not the outliers.
Figure 3 plots the difference between measured and expected slack distributions. The bimodal nature is apparent and divides the distribution into slow and fast clusters of paths. Because this result is unexpected and significant, one should first resolve this issue before trying to answer other more specific and subtle questions. Therefore, we ask the question: “What is causing the unexpected two-cluster distribution?”

Intuitively, the bimodal nature of the problem is best suitable for analysis by classification algorithms shown in Table I. However, a traditional classification algorithm does not provide reasons to explain the difference between two classes. One way to extract such reasons is using a rule learning algorithm to discover rules that explain the data. Additionally, ranking algorithms in Table I can be used to select features most relevant for explaining the observed bimodal distribution. This discussion leads to three viable sets of algorithms that can be applied to explain the unexpected test results observed in Figure 2: Classification, Rule Learning, and Ranking. Once we know the appropriate data mining algorithms to apply, we proceed to the database creation step.

B. Database Creation

Mining algorithms operate on databases to answer a specific target question. Databases contain design and test data as well as other information relevant to the question being asked. For example, the explanation for the unexpected two-cluster distribution can lie in the design or the timing analysis data.

From a design, two basic types of data are available: cell-level netlists and layout files in GDSII and/or Lef/Def format. From a timing analysis tool, cell library and timing reports are available. It is sensible to select aspects of these data files, which have the potential to explain to the unexpected test results. A set of features are derived from these files justified by prior domain knowledge. For example, path delay is modeled as additive where cells, metal interconnects, and vias along a path each contributing some delay. Given this prior knowledge, it is justified to select cell, metal, and via types to develop corresponding features describing each path. Therefore, features based on cell count, via count, and cumulated metal length can be used. Other important feature types may include location on die, STA slew and load rates, cell ordering along paths, etc.

Feature values describing each path are extracted from design files. A database containing the extracted design data and test data is created to feed selected data mining algorithms. While this step seems straightforward, it can be the most time consuming step of the entire process [2]. One challenge lies in combining various types of data into a consistent and reliable format compatible with various data mining algorithms. Other key issues include completeness of data, correlated or noisy data, missing values, high dimensionality, etc.

Constructing the most relevant features for studying the target question is a key challenge. Often, this task requires extensive domain expertise and may demand help from experts who are intimately familiar with the product design. Methods for automatic feature construction have been studied but are not yet mature for production use. An alternative is to apply feature ranking methods. With feature ranking, a large number of features can quickly be evaluated based on importance. Then, only the most relevant features are selected for subsequent data mining steps.

For storing data, relational database management systems (RDBMS) such as MySQL are a popular choice, and store data in simple flat tables. This format is preferred since the vast majority of data mining algorithms require a flat data table as input. However, this also poses a problem since relevant files such as timing reports, circuit netlists, and GDSII layout files are hierarchal and do not naturally fit into flat tables. To overcome this, standard file formats must be parsed into flat tables before mining can take place.

C. Data Mining

From the perspective of knowledge discovery, data mining is a tool that analyzes large amounts of data to extract actionable knowledge about the domain of interest. There can be two types of actionable knowledge. The first type is for improving the knowledge discovery process. For example, a mining result may suggest insufficient information content in the database, which leads to the action of constructing more features. The second type provides answers to the target question of interest, which suggests actions to diagnose the unexpected test result. For example, answering the two-cluster distribution question can suggest design changes to avoid the problem.

As shown in Table I, there are a wide variety of mining algorithms, each serving a particular purpose in the knowledge discovery process. In this section, we discuss the various types of algorithms, describe one key algorithm from each type, and show its relevance to test data analysis.

$$X = \begin{bmatrix} x_1^1 & x_1^2 & \ldots & x_1^n \\ x_2^1 & x_2^2 & \ldots & x_2^n \\ \vdots & \vdots & \ddots & \vdots \\ x_m^1 & x_m^2 & \ldots & x_m^n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

Fig. 4. Illustration of a typical dataset for mining

Figure 4 illustrates the format of a typical input dataset taken by data mining algorithms. In this illustration, X is data extracted based on features, and y is data extracted from test measurements. In the context of path delay tests, each (x_i, y_i) pair is constructed to describe path i. A tool that analyzes large amounts of data to extract actionable knowledge about the domain of interest. There can be two types of actionable knowledge. The first type is for improving the knowledge discovery process. For example, a mining result may suggest insufficient information content in the database, which leads to the action of constructing more features. The second type provides answers to the target question of interest, which suggests actions to diagnose the unexpected test result. For example, answering the two-cluster distribution question can suggest design changes to avoid the problem.

As shown in Table I, there are a wide variety of mining algorithms, each serving a particular purpose in the knowledge discovery process. In this section, we discuss the various types of algorithms, describe one key algorithm from each type, and show its relevance to test data analysis.

1) Classification and Regression: In classification and regression, a predictive function f(x) is constructed based on the data (X, y) for predicting the y value of an unseen sample \( \bar{x} \) where \( \bar{x} \not\in X \). In regression, y_i's are numerical values. In classification, y_i's are class labels. For example, in binary classification, y_i is either +1 or -1 and f(\bar{x}) is a decision function that predicts the label of \( \bar{x} \).

One of the most popular classification algorithm is the SVM-based \( \nu \)-SVC algorithm introduced in [21]. SVM algorithms work well in practice because of their efficiency in processing large datasets comprised of high-dimensional data.
For binary classification, an SVM decision function is represented as \( f(\vec{x}) = sgn\left(\sum_{i=1}^{m} y_i \alpha_i K(\vec{x}, \vec{x}_i) + b\right) \) which returns the predicted class of \( \vec{x} \). In this function, \( \alpha_i \) measures the importance of \( \vec{x}_i \) (path i) with respect to the class decision boundary. \( K(\vec{x}, \vec{x}_i) \) computes the similarity between \( \vec{x} \) and \( \vec{x}_i \).

A decision function is used for prediction, not for interpretation. Besides sample (path) importance, an SVM function provides no other knowledge other than describing how the decision boundary is constructed. Hence, SVM classification (or regression) cannot be used as the main algorithms for knowledge discovery. However, they can be used as supporting algorithms to check whether a given dataset contains sufficient information content to justify further knowledge extraction.

<table>
<thead>
<tr>
<th>Design Data Type</th>
<th>Validation Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell Count</td>
<td>98.65%</td>
</tr>
<tr>
<td>Along Path</td>
<td></td>
</tr>
<tr>
<td>Metal Length</td>
<td>98.97%</td>
</tr>
<tr>
<td>Along Path</td>
<td></td>
</tr>
<tr>
<td>Via Count</td>
<td>98.46%</td>
</tr>
<tr>
<td>Along Path</td>
<td></td>
</tr>
<tr>
<td>Location</td>
<td>98.18%</td>
</tr>
<tr>
<td>On Die</td>
<td></td>
</tr>
<tr>
<td>Least Relevant Features</td>
<td>69.25%</td>
</tr>
</tbody>
</table>

**Table II: Validation of Information Content**

Table II presents results of information content checking performed on five datasets constructed with five types of features and the two clusters of path delay measurements. The \( \nu \)-SVC binary classification algorithm is applied to check for information content in each dataset. Accuracy is calculated using 10-fold cross validation. In 10-fold cross validation, the dataset is randomly split into ten equal-size subsets. In each run, nine subsets are used to build a decision function. This function is applied to predict the class of paths in the one remaining subset. The process repeats ten times so that each path has a predicted class label. Then, an average prediction accuracy is calculated across all paths. An accuracy result above 90% typically indicates sufficient information content is present in the design data features selected to distinguishing between fast and slow paths.

2) **Transformation:** Data transformation has long been used to reduce dimensionality, improve algorithm performance, and enable visualization. In knowledge discovery, transformations such as principle component analysis (PCA) [26], linear discriminant analysis (LDA) [2], and singular value decomposition (SVD) [13] play the critical role of summarizing large quantities of data. Data summarization is a key step in knowledge discovery since it can provide a high level overview of data and facilitate making important analysis decisions.

In PCA, the top ranked principle components account for a majority of the variance in the dataset and therefore can be used to summarize the internal structure of data. For example, the first two principle components can be used to visualize high-dimensional data using a two dimensional scatter plot as shown in Figure 5. This result was obtained by PCA on the combined path delay dataset consisting of all cell, metal, and via features. The figure shows four natural clusters appearing. Interestingly, the majority of slow paths naturally fall into one cluster, while fast paths split into three distinct clusters. This result implies that a systematic effect is the likely cause of the strong bimodal split observed in the path delay test measurements.

Other transformations such as LDA and SVD can also be useful. For example, LDA maximizes class separation which is useful when predicting two known classes of data while SVD removes redundancies in data and is used for feature reduction. Both methods can be used to enhance the speed and accuracy of subsequent mining steps.

3) **Feature Ranking:** To facilitate knowledge discovery, it is desirable to discard irrelevant features. Feature ranking evaluates the importance of features with respect to a dataset. Ranking methods include chi-square test, Gaussian process regression, and forward/backward selection as discussed before. As a special case, linear SVMs can directly be used as a ranking method [27]. Among these methods, the traditional chi-squared goodness-of-fit test is the simplest.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OR4</td>
</tr>
<tr>
<td>2</td>
<td>MUX21</td>
</tr>
<tr>
<td>3</td>
<td>NOR2</td>
</tr>
<tr>
<td>4</td>
<td>MUX21</td>
</tr>
<tr>
<td>5</td>
<td>MUX21</td>
</tr>
</tbody>
</table>

**Table III: Top 3 Features Ranked by \( \chi^2 \) Test**

Table III shows an example of the top three \( \chi^2 \) ranked features for cell, metal, via, and location data types. With respect to the natural split between fast and slow paths, the top features can be interpreted as cells, metals, vias, and locations on the die which are points of interest to be further examined.

4) **Outlier Analysis:** Outlier analysis identifies points numerically distant from the trend established by the rest of the data. In low-dimensional datasets locating outliers may
be as trivial as looking at a scatter plot. For example, in Figure 2, points close to the bottom left corner can be identified as outliers. However, when analyzing high-dimensional data automated methods are needed. Some of the most popular methods are: distance based k nearest neighbor (KNN), density based one-class SVM, and clustering based local outlier factor (LOF) as shown in Table I.

While there is no rigid definition for outliers, the common view is that they represent samples that do not conform to normal behavior. The problem of detecting outliers then becomes that of establishing a metric to accurately compute sample similarity or normality.

In knowledge discovery, outlier analysis methods are useful in that they help separate normal trends from abnormal behavior. Identified outliers can be analyzed individually to uncover their unique characteristics. Non-outliers are analyzed together to uncover their trends characteristics.

Figure 6 shows the results of KNN outlier analysis performed using top cell, metal, and via features. Outliers are marked as red stars and normal samples as blue dots. If we take away the red dots (outliers), it can be observed that the blue dots clearly form a two-cluster distribution. This result further confirms that the two-cluster distribution shown in Figures 23, is likely due to a systematic issue.

5) Clustering: Clustering groups similar samples together to expose regularities in data. Popular clustering algorithms include k-means, hierarchal, and support vector clustering (SVC). In order to group samples, a notion of similarity (between samples) is needed and often computed based on a distance measure, \( d(\vec{x}, \vec{y}) = \frac{1}{2} \sum (x_i - y_i)^2 \). In k-means clustering, k clusters are found by iteratively refining an initial set of k random means. Clusters are defined as the set of samples most similar (closest) to each mean.

In knowledge discovery, clustering finds naturally occurring groups leading to more efficient descriptions of data to facilitate subsequent analysis such as rule learning. For example, Figure 7 shows a k-means clustering result where each sample is colored according to the cluster it belongs to. Again, each path is described based on top cell, metal, and via features. The result is similar to that shown in Figure 5, where one cluster of slow paths and three clusters of fast paths appear, confirming our previous findings obtained with PCA. Our intuition about the systematic issue observed during analysis is confirmed since almost all slow paths fall into one cluster.

6) Rule Learning: All data mining algorithms discussed above do not directly extract knowledge in a form that implicates causes of the underlying issue(s) associated with the unexpected test data. To uncover this type of knowledge, a rule learning algorithm should be applied.

Following the two-cluster example shown in Figure 2, a rule learning algorithm applied to this binary class of data can produce rules of the form: “if A and B are both true, then path C has a 99% chance to be in the slow cluster” where A and B constitute features and conditions on their values. For example, A could be based on the feature “maximal consecutive MUX count (MCMC)” that counts the number of consecutive MUX cells on a path, and A is the condition that MCMC \( \geq 3 \).

Rule learning lies at the heart of knowledge discovery since results are simple to understand, interpret, and often lead to actionable knowledge. Popular algorithms include classification and regression trees (CART), C4.5 decision tree learner [28], and CN2-SD subgroup discovery.

In decision tree learning, each node in a tree represents a split on one feature (e.g. MCMC) and each edge represents a splitting condition (e.g. MCMC \( \geq 3 \)). Leaf nodes of the tree represent classification or regression outputs. A path from the root to a leaf node represents a conjunction rule. The key to building a tree is choosing a variable at each node that best splits the data. Choosing the best split is performed using the information gain ratio introduced in [28].

7) Summary: Below we summarizes the typical uses of the six types of algorithms in a knowledge discovery process:

- **Classification** (or regression) can be used to check for sufficient information content in a dataset.
- **Transformation** is helpful for data visualization and dimensionality reduction.
- **Feature Ranking** is used to select relevant features or filter out irrelevant features to enhance the effectiveness in subsequent data mining steps.
- **Outlier Analysis** differentiates systematic trends and outliers so that they can be analyzed separately.
- **Clustering** uncovers natural groups in data leading to efficient data descriptions that facilitate rule learning.
**Rule Learning** extracts rules to explain data where rules are interpretable and applicable in the domain of study.

### D. Evaluation and Application of Extracted Knowledge

Evaluating and applying extracted knowledge means judging the validity, novelty, and utility, of extracted rules, models, or clusters so they can be applied to improve the knowledge discovery process, or design/test quality. In most cases, these two steps require interaction with domain experts to discuss findings and provide valuable feedback that can be used to refine learning algorithms and analysis strategies.

To evaluate findings, further analysis can be performed by simulation, inspection, or measurement. For example, if a discovered rule indicates that a particular cell type is potentially problematic, SPICE simulation may be used to double check that STA models accurately predict cell behavior. Additionally, inspection of the cell layout can rule out possible design issues. If a particular location is implicated as a problem, inspection of the layout area may help identify placement or alignment issues. Sometimes, the evaluation may lead to generation of detailed features dedicated to the issue for further analysis. Once discovered knowledge is validated, it becomes key information to direct engineering resources and improve design quality.

### IV. IMPLEMENTATION

Practical application of knowledge discovery on test data needs to be automated. Every aspect from the access of silicon and design data, to the use of various data mining algorithms, should be integrated so that minimal analyst intervention is needed. This section discuss how we implement the knowledge discovery flow for analyzing the unexpected path delay test results shown in Figure 2.

#### A. Silicon Data Gathering

Silicon path delays were measured from a recent high-performance dual-core system-on-chip design. The motivation for the experiment was to target a large number of paths to acquire broad data for calibrating a timing analysis flow. In the experiment, generation of test patterns was performed via a commercial automatic test pattern generation (ATPG) tool for the 10,000 most critical paths reported by STA. Then, delays of 2142 paths were measured based on 1 ps frequency stepping on ATE. No specific blocks were targeted. Measured paths were selected from the overall timing run as well as the availability of test patterns. Core paths were excluded because they are timed and tested separately.

#### B. Rapid Prototyping

With the growing variety of available data mining algorithms, it becomes difficult to choose just one to perform test data analysis. On the other hand, multiple algorithms offer flexibility and the potential to confirm result by independent methods. In this work, we use RapidMiner [29], an open source rapid prototyping environment for machine learning and data mining experiments. RapidMiner behaves like a high level language for describing analysis flows and is well suited for automating knowledge discovery. Our implemented KD flow is shown in Figure 8, where each box represents a particular operator and each arc represents a data path to another operator. There are operators for database integration, data selection, data mining, and report generation. Constructing the knowledge discovery flow was performed by selecting and combining various operators. Depending on the goal of analysis, a flow can be as simple as three blocks or as complex as the one shown in Figure 8.

#### C. Process Flow Design

As shown in Figure 8, there are numerous components to a knowledge discovery process flow. Execution proceeds from left to right where results of each operator on the left are passed to the next operator on the right. Independent operations can be performed in parallel to take full advantage of a multi-core processor. Each operator has several configuration parameters and once they are set, the flow can automatically proceed without further analyst involvement.

1) **Database Integration**: The first step in any discovery flow is acquiring data. On the far left of Figure 8, we see six database integration operators that connect to and select data from design and test databases. If databases exist, accessing
them is performed by selecting a schema and table name. If data is only available in standard file formats such as STA timing reports or GDSII layouts, a database must be created by parsing these files.

We had access to timing reports generated by the Cadence Encounter timing system and the top level GDSII layout file. Scripts were written to parse each file and populate a MySQL database. Timing reports were parsed to generate path based tables containing cell type, load, slew, and expected slack. GDSII layouts were parsed to extract cell locations, via types, and cumulated metal lengths along each path. Average measured path delay across 30 dies was also stored in the database as the test result of interest.

2) Data Selection and Preprocessing: Once data is stored in the database, several selection and preprocessing operators are used to format the data for best interpretation by analysis algorithms. Measurement data is split into two classes, slow paths and fast paths justified by Figures 2 and 3.

3) Feature Generation: To perform data analysis, a set of initial features are generated to encode path characteristics. For example, cell and via features were encoded as the count of each type present in a path, while metal layer features were stored as the cumulated metal length in each path. Additionally, path location was encoded as a 100 bit binary vector. Each bit represents the presence or absence of a path in a $10 \times 10$ grid partitioning the die. In our analysis, several hundred features were produced.

Cell, metal, via, and location features are combined and normalized so that no data type has more influence over others during analysis simply because of its large spread in feature values. Additional features can also be generated by combining existing features. For example, computing the ratio of wide to narrow metal length for each path, or the percent time spent in cells versus interconnect.

4) Feature Ranking: Since the volume of data can be quite large, with 2142 samples and a few hundred features, it is necessary to perform feature ranking so several data mining algorithms can run efficiently without memory issues. Feature ranking also provides valuable insight by highlighting features that are most relevant to the test data being analyzed. Chi-squared feature ranking was performed with each feature value discretized into 10 bins. Once a feature rank was established, the top 25 features were selected for further analysis.

5) Data Mining: The data mining step in a knowledge discovery flow is the most computationally involved and time consuming, requiring the most effort to properly setup. Figure 8 shows data mining blocks that include operators for cross validation, decision tree classification, subgroup discovery, rule induction, and SVM classification. Each of these operators has a specific set of parameters that need to be set and slightly different input data requirements.

The first analysis step is one that checks the information content of the data we wish to extract knowledge from. Figure 8 highlights the information content analysis step as a separate box. Evaluation is performed using a cross validation operator, containing the $\nu$-SVC classification algorithm and the 10-fold cross validation strategy discussed earlier. The $\nu$-SVC algorithm uses a Gaussian kernel with $\nu = 0.05$. Some of the validation results are discussed in Table II. Datasets containing enough information content are kept for further analysis. If cross validation cannot achieve high accuracy, new features or ranking methods may be necessary.

Subsequent algorithms applied include C4.5 and CART where classification trees are built to explain the difference between slow and fast paths. An information grain ratio of 0.2 is used to split on critical features and a maximal tree depth of 2 was selected to focus on extracting simple and descriptive rules rather than large trees.

6) Report Generation and Visualization: Since knowledge discovery is a highly iterative process it is necessary to have an automated method to report and visualize results. In this work, we generate feature rank tables, colored scatter plots, decision trees, and performance metrics that can be automatically organized into a report. The right side of Figure 8 shows several report generation operators that are used to generate HTML reports summarizing the findings of knowledge discovery. This greatly helps when communicating results to domain experts for further interpretation and feedback.

7) System Requirements: Data mining and knowledge discovery problems are some of the most resource intensive, requiring large amounts of processing power and memory. For analysis we used a four core 3 GHz Intel CPU with 8 GB of ram. In many cases, intermediate results of complex process flows could not fit into memory so changes had to be made such as performing feature selection prior to memory intensive data mining. To cope with these performance issues we analyze cell, metal, via, and location data separately ensuring that results are obtained and interpreted quickly (in minutes). More complex processes that analyzed large quantities of different data types were executed in batch mode and results stored for later review.

V. Analysis of Results

To ensure easily interpretable and actionable knowledge is discovered, rule learning and feature selection algorithms were used to analyze cell, metal, via, and location data separately. The goal was to explain the heavily bimodal silicon test results seen in Figure 2. Results are presented as decision trees and colored scatter plots showing the impact of discovered design features. Note that a path consists of a data path (from launch FF to capture FF), an uncommon clock path (clock path after the split point, feeding the launch FF and capture FF) and a common clock path. Hence, a discovered important feature could be associated with either of them.

A. Cell Type Analysis

Cell analysis was a key focus of this study with the goal of uncovering specific cells contributing to slow paths. Ranking cell types across each path implicated three critical cells: MUX21, NOR2, and OR4 as shown in Table III, where MUX21 is associated with the common clock path as a clock gate and NOR2 and OR4 are within the data path. Upon further...
analysis it was discovered that these cells only appear in a wrapper module interfacing to a memory controller located in a small region of the die.

To validate the findings, we plot paths that contain all three cells as red dots and the rest as blue stars in Figure 9. A clear split emerges, with the exception of a few extreme outliers. Domain experts provided feedback that the use of MUX21 as clock gate in the wrapper interface module was an unusual design choice. It is also important to note that the discovered rule says that MUX21 by itself is not the issue. Only when it is combined with NOR2 and OR4 on the data path (in the wrapper) would the issue occur.

B. Metal Layer Analysis

For metal layer analysis, the CART algorithm obtained the decision tree shown in Figure 10 (c). The rule says that if cumulated length of metal layer 2 within the data path is grater than $2.285 \times 10^5$ nm, then the path would be slow. This result is consistent with the high rank of metal layer 2 obtained by feature ranking analysis.

Figure 11 shows a color map plotting the distribution of metal layer 2 length across all paths. Paths colored blue have shorter metal layer 2 lengths, and paths colored orange have longer lengths. This shows that slow paths spend significantly more time in metal layer 2.

Metal layer 5 in the clock path was also found to be critical. Figure 12 shows the distribution of metal layer 5 length in the clock path. A clear split is visible where all slow paths have twice as much metal 5 length compared to fast paths.

C. Via Type Analysis

Via types between layers were also studied. Figure 10 (a) shows a decision tree explaining the difference between fast and slow paths by two via types. The tree shows that clock paths with more than 14 single vias between layers 4 and 5 and more than 70 double vias between layers 5 and 6 are consistently slow. This is further backed up by Figure 13 showing that paths containing more single vias between metal 4 and 5 are slow. It is interesting to note that these results are consistent with the analysis of metal layer 5 on the clock path shown above. It is intuitive to see that long total length of metal layer 5 can be highly correlated to more vias used between layers 4 and 5 and between layers 5 and 6.

D. Path Location Analysis

Path location was analyzed to determine if any die area was common to all failing paths. Analysis was performed using 100 element binary vectors encoding path location on a $10 \times 10$ grid dividing the die. Figure 10 (b) shows that paths intersecting grid location 16 are always slow, and Figure 14 confirms all paths passing through grid 16 are in fact slow. This location on the die corresponds to the wrapper interface module, which was discovered independently during cell analysis.

E. Summary

Four types of analysis lead to the following findings: (1) The slow paths occur mostly in the wrapper interface module for a memory controller. (2) These path share several unique characteristics: use of NOR2 and OR4 on the data path, use of...
MUX21 as clock gate, and long interconnect length in metal layer 5 on the clock path. (3) The specific use of MUX21 is unusual and must be further investigated. These are reasonable findings already confirmed by domain experts.

It is important to note that these findings are not necessarily root causes. They are indicators that guide further investigation to find the root cause(s). These findings can also be viewed as potential fixes. For example, a fix could consist of replacing MUX21 with another cell and reducing the delay through metal layer 5 by inserting a few more buffers on the clock path. Both activities are being investigated by the design team.

VI. CONCLUSION

This work presented a knowledge discovery process for explaining significant and unexpected test results. Various feature selection, transformation, and data mining algorithms were integrated to form a knowledge discovery flow capable of extracting interpretable and actionable knowledge. Results showed that automated knowledge discovery could successfully analyze design and silicon test data to explain unexpected bimodal test results. Simple reasons in the form of rules were extracted explaining why the bimodal split exists. Three summarized findings were proposed to the design team indicating issue with a memory controller interface wrapper. Validation by domain experts confirmed the feasibility and diagnostic benefit of the extracted knowledge.

REFERENCES

[9] Pouria Bastani, Nick Callegari, Li-C. Wang, Magdy S. Abadir, Diagnosis of design-silicon timing mismatch with feature encoding and importance ranking - the methodology explained, ITC, 2008.
[16] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao, “Mining Frequent Patterns Without Candidate Generation: A Frequent-Pattern Tree Approach,” Data Mining and Knowledge Discovery, 2004.