Wafer Map Failure Pattern Recognition Using Artificial Intelligence

Presenter: Andy Wee
- GLOBALFOUNDRIES

Authors: Lo Wee Tick
- GLOBALFOUNDRIES

Hui Kong Meng
- ISS-NUS Master Student
Background

• Probing methodology getting more complex with:
  – Increased parallelism
  – Optimized probing path to reduce step indexing time

• Adds challenges to wafer disposition:
  – Unnecessary retesting of wafers
    • Leading to loss of test capacity
    • Hardware pre-mature end-of-life (EOL)
    • Added cost to overall test operation
  – Failure to recover overkill dies
    • Revenue leak due to throwing away potential good dies
  – Highly dependent on past experiences
    • Lost domain knowledge
    • Time consuming
Background

• Disposition challenges

Single Site
- Possible probing issues

Multi-Site
- Could be wafer issues

Single Site
- Possible wafer issues

Multi-Site
- Could be probing issues
Background

- What are we trying to solve?
  - Recognition / Decision problem!!

Hunting Season!!!
Objectives

- Create a tool to help recognize wafer failure pattern using AI (Artificial Intelligence) to help:
  - Wafer test disposition decision
  - Reduce wafer test results review time
  - Retain knowledge gained
  - Create a more systematic way for wafer disposition process
Typical Outsourced Disposition Flow

Test House

First Pass Test Completed

Lot Disposition

Retest needed?

Move to next process

Retest selected dies

Retest needed?

Customer Disposition

Customer

Decisions!
Approach

• Extract useable parameters from available test data to compute statistical information.

• Create database for data insertion

• Select suitable AI methodology for model learning e.g. NN (Neural-Network) supervised learning, to identify recoverable failures detect various pattern failure scenarios.

• Develop novel algorithms to detect various pattern failure scenarios.

• Develop a rule-based algorithm on logical assessment of information.
Development overview

Stage 1
- Test data cleansing and parsing

Stage 2
- Test data insertion into database

Stage 3
- Generate lot & wafer level summary statistics and wafer maps

Stage 4
- Apply KE algorithm and establish findings

Stage 5
- Generate report and recommend actions

Synthetic Task
- Design
- Planning
- Assignment
- Modeling
- Scheduling

Analytic Task
- Prediction
- Diagnosis
- Classification
- Monitoring
- Assessment

June 10 - 13, 2012 IEEE Workshop
Phase 1: Data preparation (1)

- **Understanding the data**
  - Decision to use **STDF** (Standard Test Data Format) that is readily available in all major ATEs
  - Decode the STDF binary format to sieve out required data

- **Cleansing the data for database insertion**
  - Filter out flawed data e.g. missing information, misplaced fields etc.

- **Mining the data**
  - Identify critical parameters and compute statistical data as input nodes
  - Select training data sets based on the disposition scenarios
  - Select test data sets for verification
Phase 1 : Data preparation (2)

• Some example of useful data
  – To detect system failure, trouble spots pattern

<table>
<thead>
<tr>
<th>Input</th>
<th>Detection</th>
<th>Verification</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. X-Y coordinates</td>
<td>1. Start with each failure die as seed.</td>
<td>1. Repeated signature in some wafers, adjacent wafers.</td>
<td>Cluster failure True / False? Quadrant location?</td>
</tr>
<tr>
<td>2. Soft Bins</td>
<td>2. Cluster failing dies to determine if there is grouping beyond set a threshold.</td>
<td>2. Independent of tester or H/W used, cluster will still fail.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

– To detect “Known Recoverable Bins”

<table>
<thead>
<tr>
<th>Input</th>
<th>Detection</th>
<th>Verification</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soft Bins</td>
<td>Recoverable list of soft bins</td>
<td>Recovery history on different wafers, first probe, online reprobe etc.</td>
<td>Probability of subjecting to retest</td>
</tr>
</tbody>
</table>
**Phase 1: Data preparation (3)**

- **Some example of useful data**
  - To detect Parametric signature

<table>
<thead>
<tr>
<th>Input</th>
<th>Detection</th>
<th>Verification</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. X-Y coordinates</td>
<td>Map failing parametric value to population (?)</td>
<td>Test values recorded near limits?</td>
<td>Probability of recovery after retest.</td>
</tr>
<tr>
<td>2. Bin failure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Failing parametric value</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Phase 2 : System selection

- Modular system upon which new KE (Knowledge Engineering) techniques can be added.
- Self-learning system upon which new models can be added.
- Configurable with different rules for different devices, different knowledge.
- Configurable thresholds to detect various failure patterns.
- Web-based reports, easier access.
- Database approach for scalable repository.
Phase 3: System Structure (1)

- Overview – Data Processing

- Test Definition
- Parametric Test
- S/B to H/B mapping
- Bias Conditions

- Bin Limit
- Test Program
- Prober Step Seq.

- TEST
- Data Cleansing
- Report Generation
- STDF

- Data Processing
- Enterprise Database
- Report Generation Analysis

- Device Yield
- H/W Trend
- Parametric Analysis
- Loadboard Probecard

- Wafer Region Analysis
- Retest Pickup Analysis

- H/B wafermap
- S/B wafermap
- Bin Summary

- Retest Pickup Analysis
- Parametric Test Summary
Phase 3: System structure (2)

• Overview – Modules

- Pending
- Archive
- Process
- Error
- Insertion
- Asynchronous background tasks
- Messaging trigger
- Computation
  - Compute statistics
  - Compute KE Algo
  - Generate reports
- Enterprise Database
- User notification once data available and processed
- Data available?
- YES
- NO
- Submit Request
- Formatter
- NO
- YES
- Client or Web-based UI

STDF
Phase 3: System structure (3)

- Overview – Hybrid Intelligence System

- **INPUT**
  - Stack Maps
  - Parametric Data
  - Fail Bin Data

- **Lot Statistical Data**
  - **Wafer Statistical Data**
  - **Electrical Analysis**
    - **BP**
    - **J48**
    - **SVM**
  - **Hybrid NN Component**
    - **Wafer Pattern Recognition (customized search algo)**

- **Predict probability of recovery**
  - **Pattern**
    - Line
    - Segment
    - Arc

- **Rule Based Engine**

Note:
BP - Backward Propagation
SVM – Support Vector Machine

June 10 - 13, 2012
IEEE Workshop
Phase 3: System structure (4)

- Overview – AI System Modelling and Ensemble

<table>
<thead>
<tr>
<th>Input Type</th>
<th>Neural Network Multi-layer Feed Forward Back Propagation (NN-MLFF-BP)</th>
<th>Decision Tree - J48</th>
<th>Ensemble</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>dxy*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bearing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testval1*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testval2*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testval3*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testval4*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testval5*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Count</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Segment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stack*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Note:
dxy – Euclidean distance
Testval – Top parametric fail values
Segment – Neighboring failures
Stack – Stack map data
Phase 3: System structure (5)

- AI Modeling and Ensemble – How it integrate

**Ensemble System**

- BP_Output
- J48_Output
- SVM_Output

\[ f(t) = \text{Bp}_\text{output} \times G_{bp} + \text{J48}_\text{output} \times G_{J48} + \text{SVM}_\text{output} \times G_{svm} \]

\[ f(t) \geq 0.6666, \text{ otherwise} \]

**Trained Values**

<table>
<thead>
<tr>
<th></th>
<th>BP</th>
<th>J48</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values</td>
<td>75.00%</td>
<td>84.00%</td>
<td>81.00%</td>
</tr>
</tbody>
</table>
Phase 3: System structure (6)

- Details of the AI model employed

**NN-MLFF-BP**

- $w'_{(x1)1} = w_{(x1)1} + \eta \delta_1 \frac{df_i(e)}{de} x_1$
- $w'_{(x2)1} = w_{(x2)1} + \eta \delta_1 \frac{df_i(e)}{de} x_2$

**J48 Decision Tree**

- ppl
- met. proc.
- nucleus
- false
- N(422/51)
- regn
- true
- E (637/138)
- false
- N(187/64)
- true
- N(283/88)
- E (362/150)
- false
- E (157/67)

**SVM**

- $K(x_i, x_j) = \phi(x_i)\phi(x_j)$
- Margin = $2 / \sqrt{w^T w}$
- Support Vector
- Margin = $2 / \sqrt{w^T w}$
- Support Vector
Phase 3: System structure (7)

- Domain Knowledge

- Wafer/Lot low yield
  - Wafer fabrication
    - Mask defects
      - Systematic pattern failure
        - Localized spot
      - Process defects
        - Random pattern failure
        - Edge rim failures
    - Random pattern failure
  - Systematic pattern failure
  - Probe card
    - Site to site
    - Planarity
    - Low performance site
  - Prober
    - Site to site
    - Planarity
    - Wrong test program
  - Test setup
    - Low performance site
    - Probe card
    - Prober
    - Wafer prober
    - Wafer/Lot low yield
    - Test setup
  - H/W issue
    - PE cards
    - Connection
  - Interface
    - H/W issue
    - PE cards
    - Connection
Phase 3 : System structure (8)

- Rule-based Engine Logic (translated Domain Knowledge)

<table>
<thead>
<tr>
<th>Yield</th>
<th>Yield</th>
<th>NN</th>
<th>Pattern</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ave, 1st Pass</td>
<td>Ave, Auto Recovery</td>
<td>1st Pass Yield</td>
<td>Auto Reprobe Recovery</td>
<td>Ave. Auto Reprobe Recovery</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>H</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>H</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&gt;0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&gt;0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&gt;0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

June 10 - 13, 2012  IEEE Workshop  20
Phase 4: Pattern Recognition Methodology (1)

- Arc Detection

Definition + Training

Filter results + Identification
Phase 4: Pattern Recognition Methodology (2)

- Line Detection

Definition + Training

Filter results + Identification
Phase 4: Pattern Recognition Methodology (3)

• Segment Detection

Definition + Training

Filter results + Identification
Phase 5: Testing and Results (1)

- Results from AI Model (Training Sets)

<table>
<thead>
<tr>
<th>Items</th>
<th>NN-MLFF-BP</th>
<th>J48</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Instances</td>
<td>38,122</td>
<td>38,122</td>
<td>38,122</td>
</tr>
<tr>
<td>Correctly Classified Instances</td>
<td>27,391 (71.8509 %)</td>
<td>30,932 (81.1395 %)</td>
<td>29,907 (78.4508 %)</td>
</tr>
<tr>
<td>Incorrectly Classified Instances</td>
<td>10,731 (28.1491 %)</td>
<td>7,190 (18.8605 %)</td>
<td>8,215 (21.5492 %)</td>
</tr>
<tr>
<td>Kappa statistic</td>
<td>0.4008</td>
<td>0.6032</td>
<td>0.539</td>
</tr>
<tr>
<td>Mean absolute error</td>
<td>0.3494</td>
<td>0.2584</td>
<td>0.2155</td>
</tr>
<tr>
<td>Root mean squared error</td>
<td>0.423</td>
<td>0.373</td>
<td>0.4642</td>
</tr>
<tr>
<td>Relative absolute error</td>
<td>72.6676 %</td>
<td>53.7533 %</td>
<td>44.8202 %</td>
</tr>
<tr>
<td>Root relative squared error</td>
<td>86.2723 %</td>
<td>76.0688 %</td>
<td>94.6788 %</td>
</tr>
<tr>
<td>Coverage of cases (0.95 level)</td>
<td>99.6695 %</td>
<td>98.0562 %</td>
<td>78.4508 %</td>
</tr>
<tr>
<td>Mean rel. region size (0.95 level)</td>
<td>94.233 %</td>
<td>83.5646 %</td>
<td>50 %</td>
</tr>
</tbody>
</table>
Phase 5: Testing and Results (2)

- Results from AI Model (Test Sets)

<table>
<thead>
<tr>
<th>Items</th>
<th>NN-MLFF-BP</th>
<th>J48</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Instances</td>
<td>3997</td>
<td>3997</td>
<td>3997</td>
</tr>
<tr>
<td>Correctly Classified Instances</td>
<td>69.85245%</td>
<td>69.9775%</td>
<td>71.6537%</td>
</tr>
<tr>
<td>Incorrectly Classified Instances</td>
<td>30.1476%</td>
<td>30.0225%</td>
<td>28.3463%</td>
</tr>
<tr>
<td>Kappa statistic</td>
<td>0.118</td>
<td>0.3124</td>
<td>0.2893</td>
</tr>
<tr>
<td>Mean absolute error</td>
<td>0.4004</td>
<td>0.3319</td>
<td>0.2835</td>
</tr>
<tr>
<td>Root mean squared error</td>
<td>0.449</td>
<td>0.447</td>
<td>0.5324</td>
</tr>
<tr>
<td>Relative absolute error</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Root relative squared error</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Coverage of cases (0.95 level)</td>
<td>99.9249%</td>
<td>97.298%</td>
<td>71.6537%</td>
</tr>
<tr>
<td>Mean rel. region size (0.95 level)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>
Phase 5: Testing and Results (3)

- Results from Rule-based from WEB UI

![Image of testing results and WEB UI interface with tables and charts.](image-url)
Other Considerations

• Features
  - Dynamic DB creation per lot. Ex.. `lot_<lotnumber>`
  - Childlots insert into parent lot DB.
  - Each STDF is tracked.
  - Traceable first pass, auto-reprobe, offline retest.
  - Each PTR records has its own table: `ptr_<ptrnumber>`
  - Lot DB be dropped to conserve space.
  - Redundancy for performance. XY coordinates included every table

• Performance (Centrino 2 core 2.1Ghz, 4GRam laptop)
  - Size: 30STDFs (121MB), DB size: ~ 4GB, 250 tables.
  - Insertion duration: Approx ~ 10-15mins per STDF.
  - Sql query : 10.22M records, 16850 results, 32 sec
# List of tools used (1)

<table>
<thead>
<tr>
<th>Tools</th>
<th>Info</th>
</tr>
</thead>
</table>
| **MYSQL database** | Widely available robust database with extensive technical help and online information available. Support in many platforms, particularly for Windows and various Linux distributions.  
Test data on database makes it easy to extract and access from most software development language and tools. For this project, a prototype database will be build to store test data, later to be part of the overall system. The database can then be switched to the sponsors' internal database system via an interface.  
[link](http://www.mysql.com/)                                                                                                                                                                                                 |
| **Weka Data-mining** | Weka is a datamining software written in Java. It has a comprehensive set of analysis tools for data exploration and understanding. Weka java API can be access and called to perform data analysis functions and return results. We have successfully tested Weka's connection to MYSQL database.  
[link](http://www.cs.waikato.ac.nz/ml/weka/)                                                                                                                                                                                                 |
| **Oracle Netbeans** | Netbeans IDE for Java application development and complementary to Glassfish.  
[link](http://netbeans.org/)                                                                                                                                                                                                 |
## List of tools used (2)

<table>
<thead>
<tr>
<th>Tools</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LibSTDF</strong></td>
<td>C++ stdf-reader project derived from FreeSTDF project. Link: <a href="http://freestdf.sourceforge.net/">http://freestdf.sourceforge.net/</a></td>
</tr>
<tr>
<td><strong>Rule-based Development tools and libraries</strong></td>
<td>JBOSS DROOLS  link - <a href="http://www.jboss.org/drools">http://www.jboss.org/drools</a></td>
</tr>
<tr>
<td><strong>Neura Network</strong></td>
<td>Using Weka decision tree and NN functions.</td>
</tr>
</tbody>
</table>
List of tools used (3)

<table>
<thead>
<tr>
<th>Tools</th>
<th>Info</th>
</tr>
</thead>
</table>
| Oracle GlassFish Application server | Oracle Glassfish application server community version is an Open Source application server that implements JAVAEE specification and is a potential good platform to build on.  
Glassfish provides a web container for web application, dependency injection to link services to business objects (KE algorithms), message queue for notification, asynchronous operations, timer service for cron jobs and ease of use among many other features. Glassfish allows us to build a proof of concept system that is flexible to change in later part of the project development.  
link: [http://glassfish.java.net/](http://glassfish.java.net/) |
Conclusion

- It is concluded that the AI system deployed can help detect and flag-out underlying wafer process or test related problems. However, the accuracy of the prediction to decide on the retest is currently not able to replace human judgment. Nonetheless, it is a very useful tool to help minimize over-sights and definitely a good tool for knowledge retention. With further fine tuning to the system and by employing improved AI algorithm, it is hoped that the prediction accuracy can improve and plays a bigger complimentary role in the disposition process.
Future Work

- More training sets with new patterns to refine pattern detection
- Possible upstream and downstream integration: Etest data input, Final Test data input
- Explore other KE techniques, models.
Thank You

• We would like thank the following teams for the successful completion of the project:

ISS-NUS KE22 FYP Team:
Supervisors: Miss Fan Zhen Zhen, Dr Famming, Charles Pang
Team members: Tran The Anh, Tai JiaJun, Chew Poh Kwee
http://www.iss.nus.edu.sg/iss/browse.jsp?cid=7&pid=45&rid=45&type=9&name=Our_Staff

GLOBALFOUNDRIES Team:
Sponsor: Yeo Chak Huat
Team members: Lee Eu Jin, Zhong Jie