BITMAP TEST DATA IN STDF FILE

Costantino Amico
STMicroelectronics
Via C. Olivetti 2, 20041 Agrate Brianza (Mi) Italy

Biography

Dr. Costantino Amico received the degree in electronics from Istituto Tecnico Industriale Feltrinelli Milano and then Economics Laurea degree from Universita' Statale Milano. He joined STMicroelectronics 1989, coming from a previous experience of 4 years in a software house: Data-Management s.p.a, where he was involved in bussines software development. His activity in STM has been always dedicated to find state of the art solutions on Engineering Test Data Analisys and Test Programs management supporting Operations and Supply Chain activities. Participation to an ESPRIT project to develop an Engineering Data-Analysis tool based on SAS (ENGWS) . Good knowledge of the overall IC fabbrication and testing mainly focused on data test data quality and processing.

Main achievements:
- TPMS Test Program Management System, a world wide web based tool to manage test programs developments and distributions
- SWAT Stdf via Web Analysis Tool
- X,Y EWS traceability in Final Test data
- Product Characterization tools

Abstract:
The project - STDF is a binary test data format file proposed as standard several years ago by Teradyne. The basic concept is that having a standard format to store test data allows tester suppliers and semiconductors companies to concentrate more on their core business rather then test data format problems.
STM has, since 2000, shared this idea and adopted STDF as its internal standard format to store test data, asking to all ATE suppliers to have test data produced in such format. Standard data-analysis tools and data-bases have been then built on top of STDF. Clearly this kind of standardization has given impressive improvements on problem detecting and yield improvements along last years.
STDF stands for "Standard Test Data Format" so it must be the main support for all type of data produced by testing stages (Probe and Final), in reality Bitmap data, for historical reasons, have been not defined and stored in STDF while bitmap data are per sure test data.
Due to the above weakness, Bitmap data have been produced since now in many different formats either binary or ASCII with different compression algorithms and granularity (by wafer, by die, by memory test). This lack of standardization is producing delays and inefficiencies on data-analysis whenever memory bitmap information has to be deeply investigated. Clearly this impacts Manufacturing ramp up, time to quality and time to volume.
So we started a project to embed memory bitmap data in STDF.
Project had been divided in 2 steps:
1. Definition an effective GDR (Generic Data Record) record format to store bitmap data inside STDF in a compressed binary format. Inside STDF specification there is a general purpose record called GDR that can be customized to store new kind of information not present in the standard records set.
2. Implementation of the solution found in all ATE to have Bitmap GDR natively produced by testers during test operation

Step 1, that has the most added value, had been already completed. GDR had been defined and validated with following main expected results:

- STDF size is less that the sum of STDF + Original bitmap file
- Data Analysis and Data-Base tools impact is negligible
- Data flow is totally under control and easy to manage
- Bitmap Data Analysis efficiency dramatically improves

Step 2, the deployment and small adaptation to production environment, just started and will be completed within 2006.

The GDR – The defined GDR record is able to store information of bitmap in two different compressed formats:

Format 0=Rectangular homogenous out
Format 1=Bit-image stream

As well as BIST summary information in the header part.

This flexibility allows to adapt the approach to the most common way in whch bitmaps are generated by testers or by external test program functions. Reducing the test time impact and allows an easy implementation by ATE in varius tester platforms.

Rectangular homogenous compressed format – Defines a rectangular area of the memory matrix (Xmin,Xmax,Ymin,Ymax) that have produced the same failed pattern. Very useful to compress failed rows or colums, for instance. In case of single cell just X and Y coordinates are stored in the record to reduce at maximum occupied space.

Bit-image stream – Defines a BitImage stream, compressing the not failed words to a single bit=0. The BitImage stream if defined by Xmin, Ymin, Xmax, Ymax, as in the Rectagular format, but with the addition of boudaries TOP,BOTTOM or LEFT,RIGHT (depending of the defined scanning direction: ROWS or COLUMNS).

GDR Layout – Following the GDR version 2.5 layout:

For each memory test, for each DIE (X,Y), there is one or more GDR. Each GDR is composed by an Header + one or more bitmap Data Blocks.

	Description	Type	Notes
	Record id	C*n	BITMAP_DATA_2.5
	Type_flag	U*1	0=Rectangular homogenous out 1=Not compressed bit-image stream
	X_Coord	I*2	
	Y_Coord	I*2	
	Test Number	U*2	0 = missing. Test number not available
	Test Name	C*n	e.g. “5V March” max 32 chars. **Mandatory**
	GDR_Sequence	U*1	GDR sequence number in case we have more than one GDR for the same X,Y,memory test . See detailed description
	I/O	U*2	Word length 8,16,32,64 up to 16535
	Number of blocks present in this GDR	U*2	
	Opt_Flag	U*1	Optional information flag
	Core_Id	C*n	For multi core memories
	Bist_Error	U*1	Used to store Bist error informations
	Memory_Cut	U*1	Memory size in case of Multi size memories
	March_Element	U*4	Used with bist methodology
	Fail_Bits_In_ March_Element	U*4	Used with bist methodology
	Fail_Code	C*n	To identify know fail codes (example: Single Row, Single Column…..)
	Tot_Fail_Words	U*4	Total failed words in the following blocks
	Fail_Bits	U*4	Failed bits in the following blocks
	Fail_Row	U*4	Failed row in the following blocks
	Fail_Col	U*4	Failed col in the following blocks
	Mem_Cols	U*4	Total number of columns in the memory
	Mem_Rows	U*4	Total number of rows in the memory
	Mem_Rows	U*4	Total number of rows in the memory

Following the header section there are Data Block section(s):

- One block for each rectangular section, in format 0
- One for many blocks depending of the size of the bitimage, in format 1

DATA BLOCKS (1 to n)	**Data Info**	B*n
		In case of Type_flag = 0 Data_Info contains the following information: • **Xmin, Ymin, Xmax, Ymax**, (4 bytes for each field that then will be translates to U*4 using big-endian format) • **Out** length is variable depending of I/O size
	Bit_Image This field is repeated **n** times up to reach the Bit_Image_Len	In case of Type_flag = 1 Data_Info contains the following information: • **Xmin, Ymin, Xmax, Ymax**, (4 bytes for each field that then will be translates to U*4 by reader program using CPU TYPE) • **LOT** (4 Bytes, Left Or Top margin of the scan area) • **ROB** (4 Bytes, Right Or Bottom margin of the scan area) • **SD** (1 Byte, Scan direction 0-row scan, 1-column scan) • **BIT_Image_Len** (2 Bytes, Length of bit-image)

Conclusion:
Full implementation of the defined approach will give following benefits:

- Disk space saving 40% to 60%
- Data-flow simplification
- Data integrity improvement
- Distributed analysis on bitmap
- All-In-One data backup/restore
- New integrated analysis directly on STDF (Bitmap vs bins or parameters)
- Is a prerequisite to have a “in-production” bitmap collection

Data:

1 Wafer as example			
Total Number of failed word (16 Bits)		6709707	
Saved Space			
	Not Compressed files	Compressed files	
STDF	11193076	3091611	Bytes
BITMAP	56146356	11493305	Bytes
BITMAP + STDF	67339432	14584916	Bytes
BITMAP in STDF	30279397	9365051	Bytes
Saved PCT	**55.03%**	**35.78%**	

Acknowledgments:
My thanks to Paola Giuffre’, Adriano Donadoni ,Vincenzo Tancorre, Sara Fiorina, Massimiliano Catinoto, Céline Chantome, Mehdi Sebbane, Musumeci Gloria, Francesco Scauso, Maurizio Gruppillo, Giuseppe Bruno , Massimo Tomaini, Lorella Bordogna, Alberto Pagani who have activelly contribuited to the success of this project

BITMAP TEST DATA IN STDF

Costantino Amico
STMicroelectronics

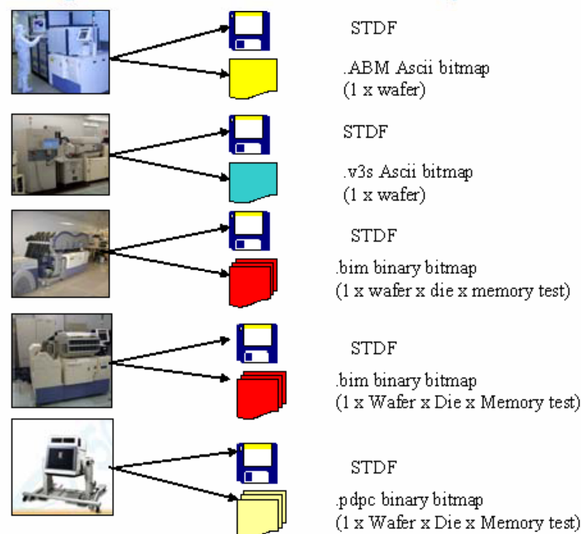
STDF Scope vs bitmap data

- STDF means: Standard **Test Data** Format
- **Test Data** is any data result produced by die testing (EWS/FT) from tester/test program
- Bit Maps data are produced by tester/test program during die testing so they are **TEST DATA**
- BitMap Test Results should be stored to STDF to have a coherent approach and to get immediate benefits of any data flow already implemented for STDF files
- Moreover having binning & test rejects, parametric & functional test results and BITMAP data in the same file allows a very easily implementation of complex and reliable data analysis

DRAWBACKS before BITMAP in STDF

- Not standardization on bitmap format
- Not clear ownership of the data
- Not easy data management. In some cases thousands of bitmap files per wafer
- Not standardization on bitmap data-flow and management
- Data integrity Not guarantee. Link between bitmap and STDF files is not 100% reliable
- Data duplication, unnecessary disk space usage and network overload

Starting condition on different platforms



Project Goal

Define and validate an effective way to store bitmap data inside STDF

The Challenge

- STDF format provides a set of standards records to store Fail counters (BIN & TEST), Parametric and Functional test results, plus some other descriptive and summary record.
- No dedicated records to store BITMAPS
- Bitmap data is usually involving huge amount of data and disk space

The Solution

- STDF format provides a special record called GDR (Generic Data Record) that could be customized to store any kind of information
- With a proper customization of the GDR we were able to define an effective binary compressed format to store BITMAP data

BITMAP GDR Layout

- Header section to store descriptive/summary information
  - Memory Test Name
  - Fail summaries (words, bits, cols, rows)
  - Memory size, word size, bist information
- One or more Data Block(s) to store failed bits

HEADER Layout (details 1/2)

Record id	C*n	BITMAP_DATA_2.5
Type_flag	U*1	0=Rectangular homogenous out 1=Not compressed bit-image stream
X_Coord	I*2	
Y_Coord	I*2	
Test Number	U*2	0 = missing. Test number not available
Test Name	C*n	e.g. “5V March” max 32 chars. **Mandatory**
I/O	U*2	Word length 8,16,32,64 up to 16535

HEADER Layout (details 2/2)

Opt_Flag	U*1	Optional information flag
Core_Id	C*n	For multi core memories
Bist_Error	U*1	Used to store Bist error informations
Memory_Cut	U*1	Memory size in case of Multi size memories
March_Element	U*4	Used with bist methodology
Fail_Bits_In_ March_Element	U*4	Used with bist methodology
Fail_Code	C*n	To identify know fail codes (example: Single Row, Single Column…..)
Tot_Fail_Words	U*4	Total failed words in the following blocks
Fail_Bits	U*4	Failed bits in the following blocks
Fail_Row	U*4	Failed row in the following blocks
Fail_Col	U*4	Failed col in the following blocks
Mem_Cols	U*4	Total number of columns in the memory
Mem_Rows	U*4	Total number of rows in the memory

Data Block Layout (details)

Data_Info B*n			In case of Type_flag = 0 Data_Info contains the following information: •**Xmin, Ymin, Xmax, Ymax**, (4 bytes for each field that then will be translates to U*4 by reader program using big endian format) •**Out** length is variable depending of I/O size
			In case of Type_flag = 1 Data_Info contains the following information: •**Xmin, Ymin, Xmax, Ymax**, (4 bytes for each field that then will be translates to U*4 by reader program using using big-endian format) •**LOT** (4 Bytes, Left Or Top margin of the scan area) •**ROB** (4 Bytes, Right Or Bottom margin of the scan area) •**SD** (1 Byte, Scan direction 0-row scan, 1-column scan) •**BIT_Image_Len** (2 Bytes, Length of bit-image)
This field is repeated **n** times up to reach the Bit_Image_Len	Bit_Image	B*n	Bit-image data the filed size is 255 bytes so that if the Bit_Image_Len is greater than 255 more that one bit_image fields will be present

Example of Rectangular homogenous out

X/Y	Y=5	Y=6	Y=7	Y=8	Y=9	Y=10
204						
205					11111111	11111111
206					11111111	11111111
207					11111111	11111111
208						
209						
210		01010101				
211		11110000				
212						
213						
214						
215						
216						
217						
218						
219						
220	10101010					

Fail Address	Fail Data (Dec)
220,5	170
210,6	85
211,6	240
205,9	255
206,9	255
206,9	255
205,10	255
206,10	255
207,10	255

220,5,170
210,6,85
211,6,240

205,9,207,10,255

bit-image stream (1/2)

X/Y	Y=5	Y=6	Y=7	Y=8	Y=9	Y=10
204						
205		0	0	0	11111111	11111111
206		0	0	0	11111111	11111111
207		0	0	0	11111111	11111111
208		0	0	0	0	
209		0	0	0	0	
210		01010101	0	0	0	
211		11110000	0	0	0	
212		0	0	0	0	
213		0	0	0	0	
214		0	0	0	0	
215		0	0	0	0	
216		0	0	0	0	
217		0	0	0	0	
218		0	0	0	0	
219		0	0	0	0	
220	10101010	0	0	0	0	

LOT

ROB

bit-image stream (2/2)

X/Y	Y=5	Y=6	Y=7	Y=8	Y=9	Y=10
204						
205		0	0	0	11111111	11111111
206		0	0	0	11111111	11111111
207		0	0	0	11111111	11111111
208		0	0	0	0	
209		0	0	0	0	
210		01010101	0	0	0	
211		11110000	0	0	0	
212		0	0	0	0	
213		0	0	0	0	
214		0	0	0	0	
215		0	0	0	0	
216						
217						
218						
219						
220	10101010					

LOT

ROB

1	1	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0 Pad bit to 32

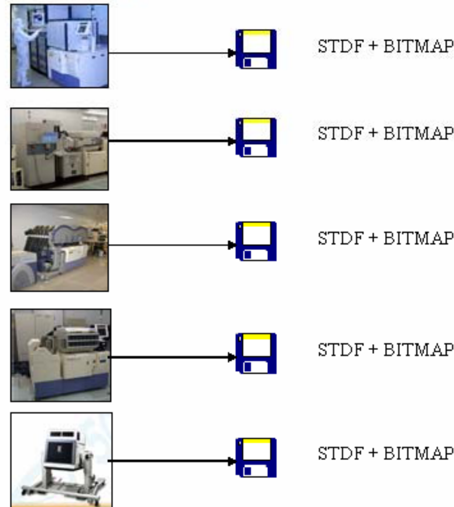
New approach benefits

- Clear ownership of data
- Data flow already existing well managed and known
- Standardization of bitmap format
- No data duplication and network overload
- Data integrity (bitmap link to other test data) is assured by default. Bitmap are embedded in STDF
- Bitmap data online & available on-demand for whole company engineering community
- Open a new scenario giving enhanced capabilities for future bitmap analysis
- Is one of the key elements to go on a real in-production bitmap standard collection

Example of saved space

1 Wafer as example			
Total Number of failed word (16 Bits)		6709707	
Saved Space			
	Not Compressed files	Compressed files	
STDF	11193076	3091611	Bytes
BITMAP	56146356	11493305	Bytes
BITMAP + STDF	67339432	14584916	Bytes
BITMAP in STDF	30279397	9365051	Bytes
Saved PCT	**55.03%**	**35.78%**	

With new approach on different platforms



Acknowledgments

My thanks to Paola Giuffre', Adriano Donadoni ,Vincenzo Tancorre, Sara Fiorina, Massimiliano Catinoto, Céline Chantome, Mehdi Sebbane, Musumeci Gloria, Francesco Scauso, Maurizio Gruppillo, Giuseppe Bruno , Massimo Tomaini, Lorella Bordogna, Alberto Pagani who have activelly contribuited to the success of this project