

STDF Fail Datalog Standardization ITC 2007 - Update

Ajay Khoche
24 Oct 2007



Agenda

- Welcome
- Background
- “New Standard Name” announcement
- Review of Assumptions and Processes
- STDF Quick Review
- Details of the New standard
- Next Steps and Timelines

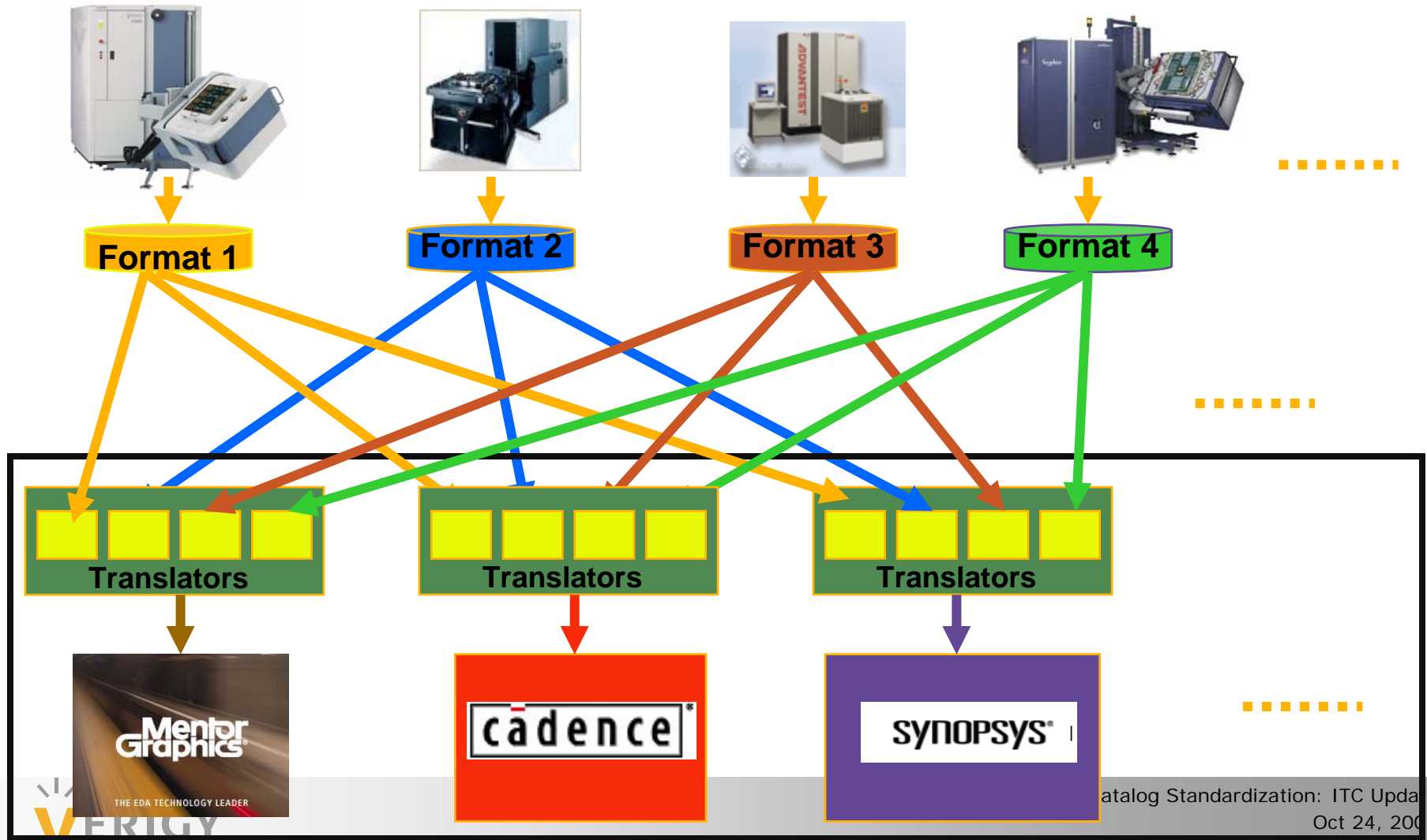
Working Group Members



Thank You!!!

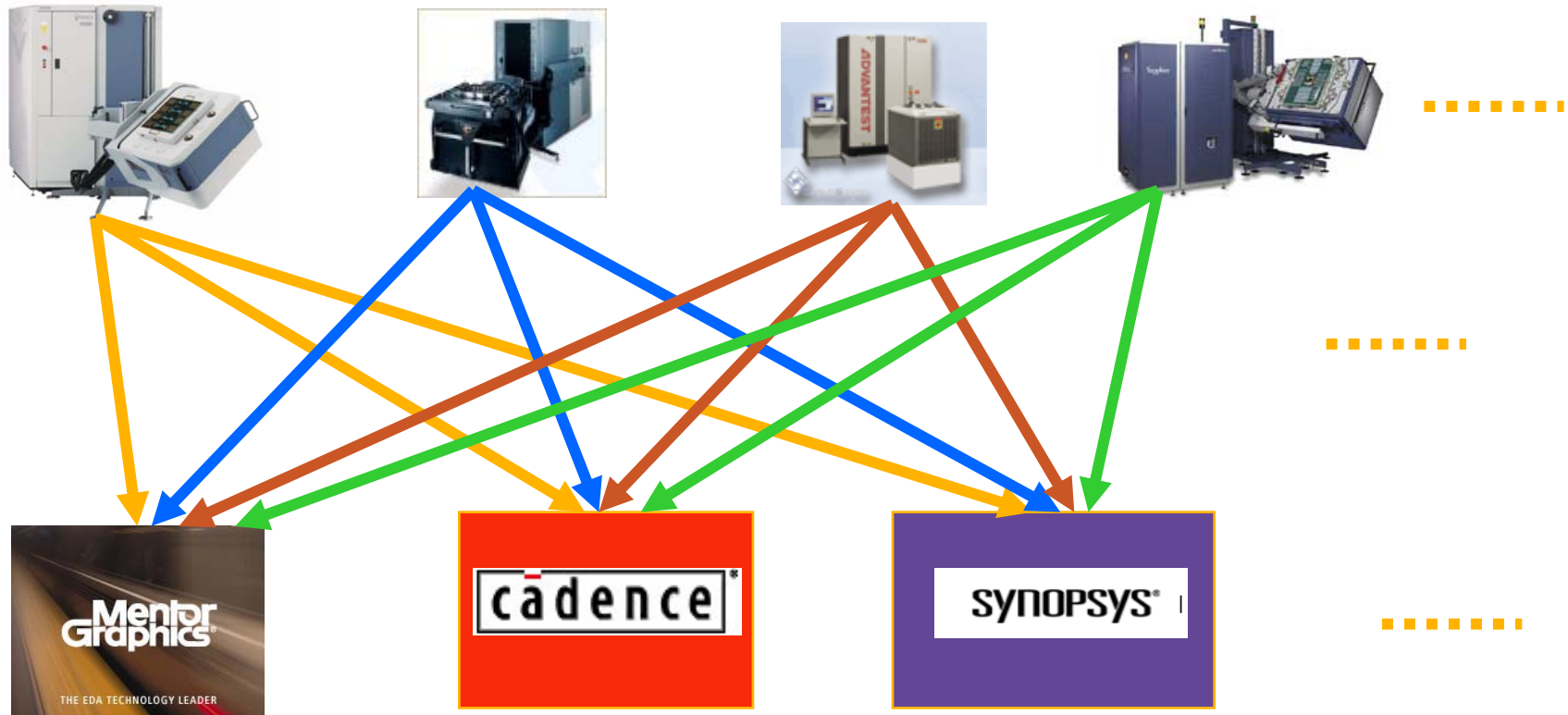
Format Mesh (...Mess?)

EDA Tool Translates ATE Formats

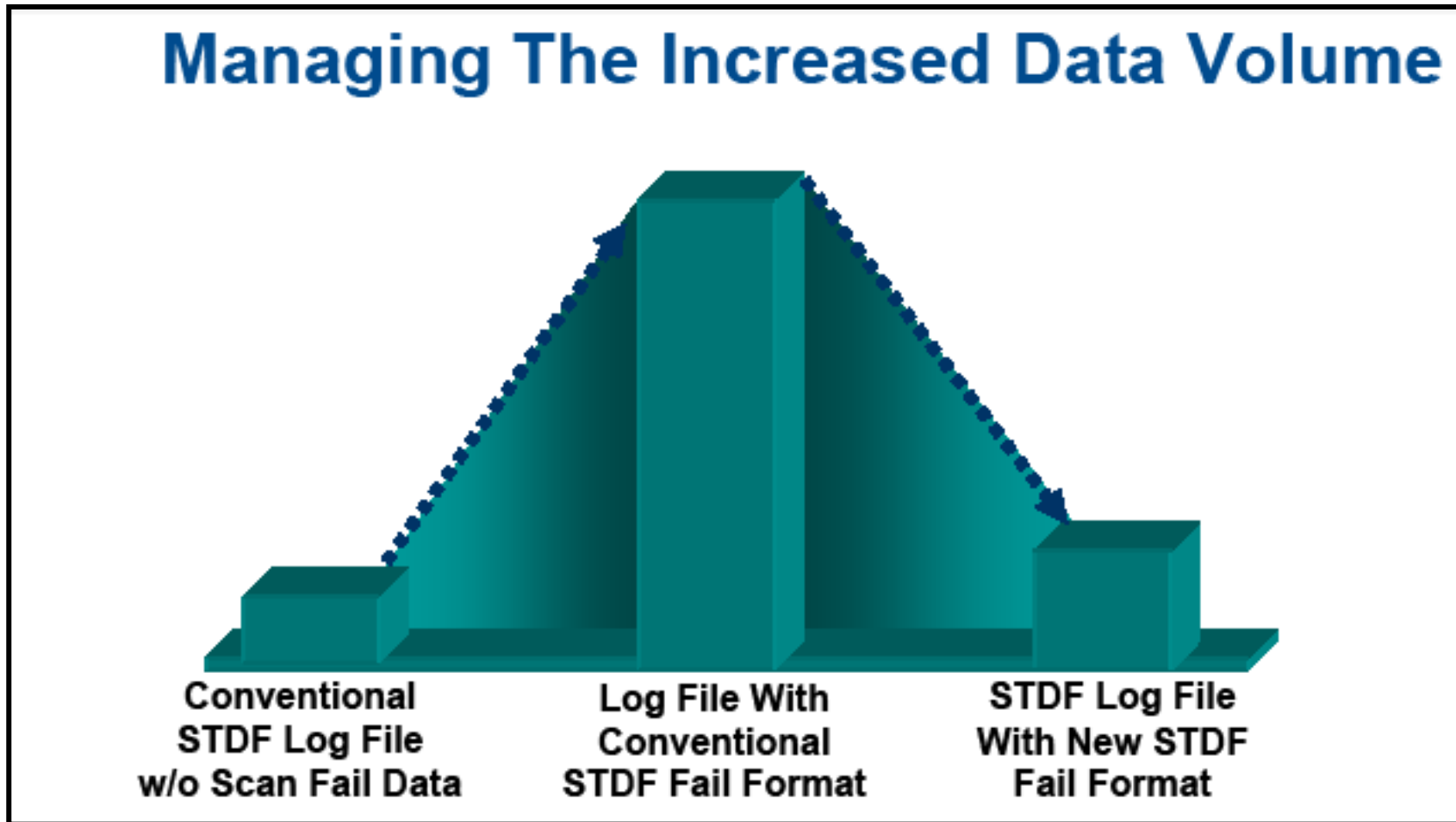


Format Mesh (...Mess?)

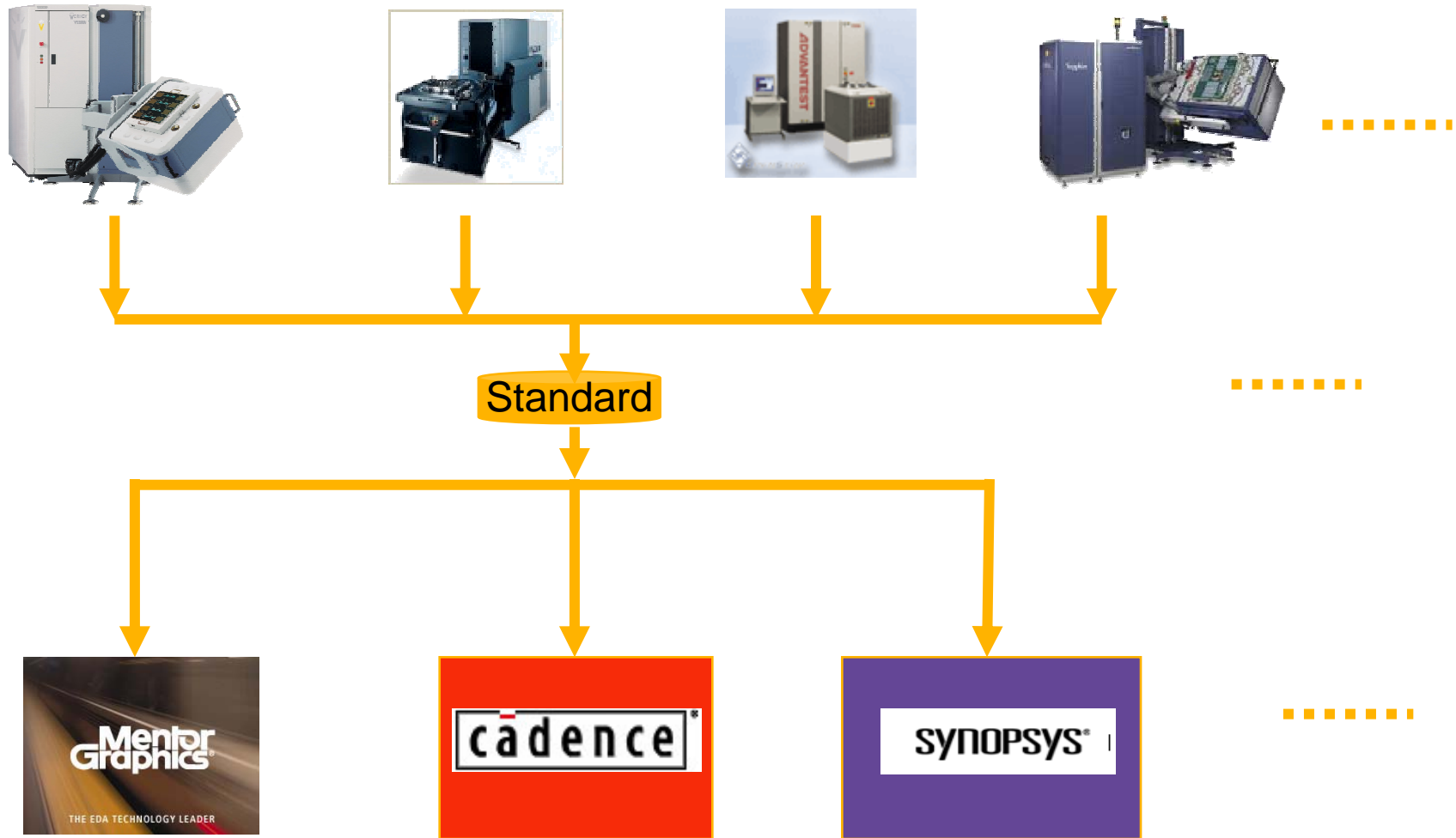
ATE Generates Data in EDA Tool Format



And The Data Volume



Standard Format



The Name



STDF V4 - 2007

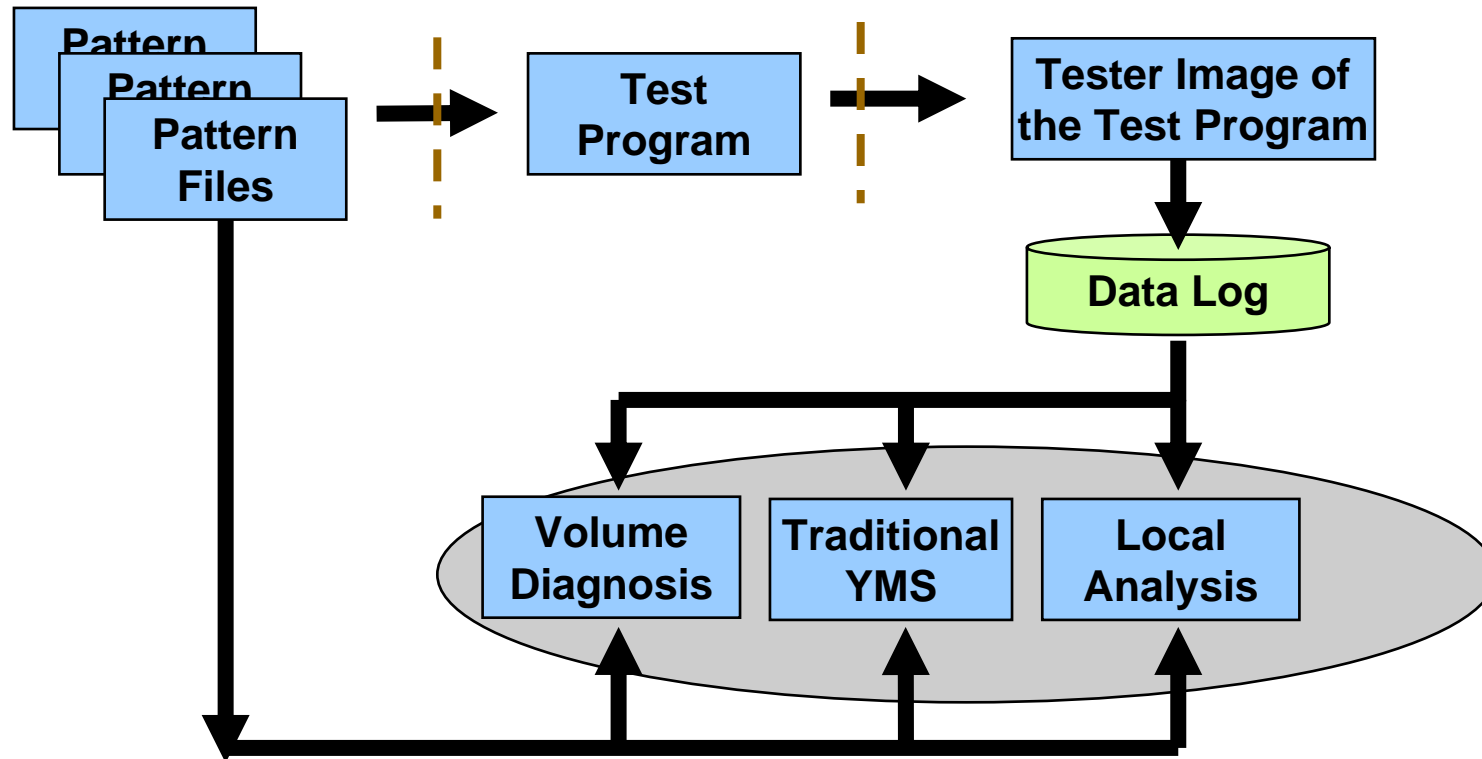
High Level Overview of STDF V4-2007

- Targeted to cater to Volume diagnostics applications
- Leverages and Augments STDF V4
 - Leverage the header information
 - Augment for scan
- Can Coexist with STDF V4
- Three scenarios supported
 - STDF V4 with No Scan Fail (Traditional usage)
 - STDF V4 with traditional and Scan fail information
 - STDF V4 with Scan fail information only

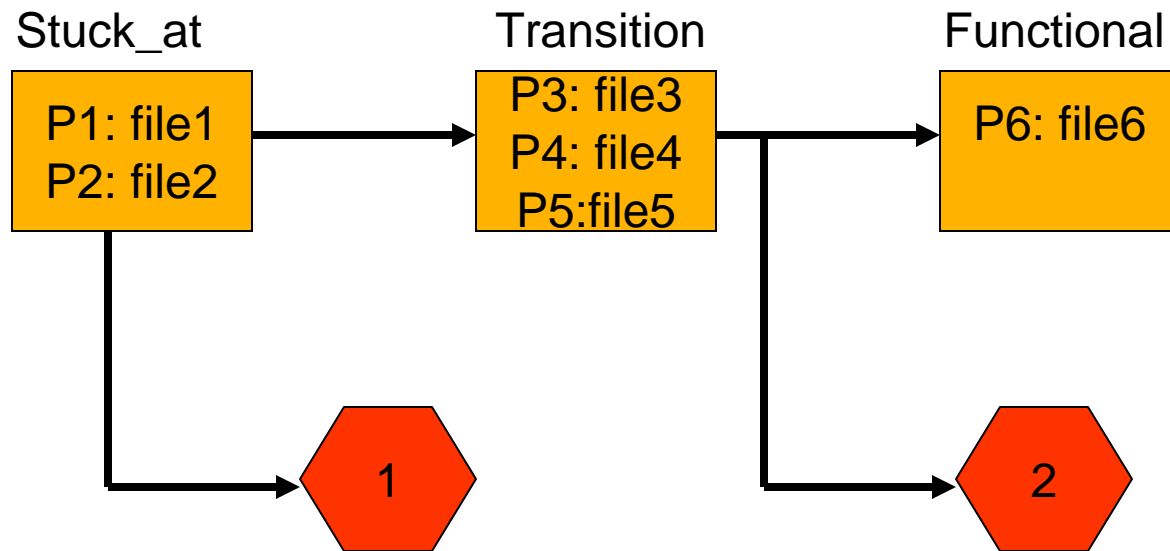
Process Used for Generating the Standard

1. Requirement gathering
2. Data model generation
3. Mapping to records

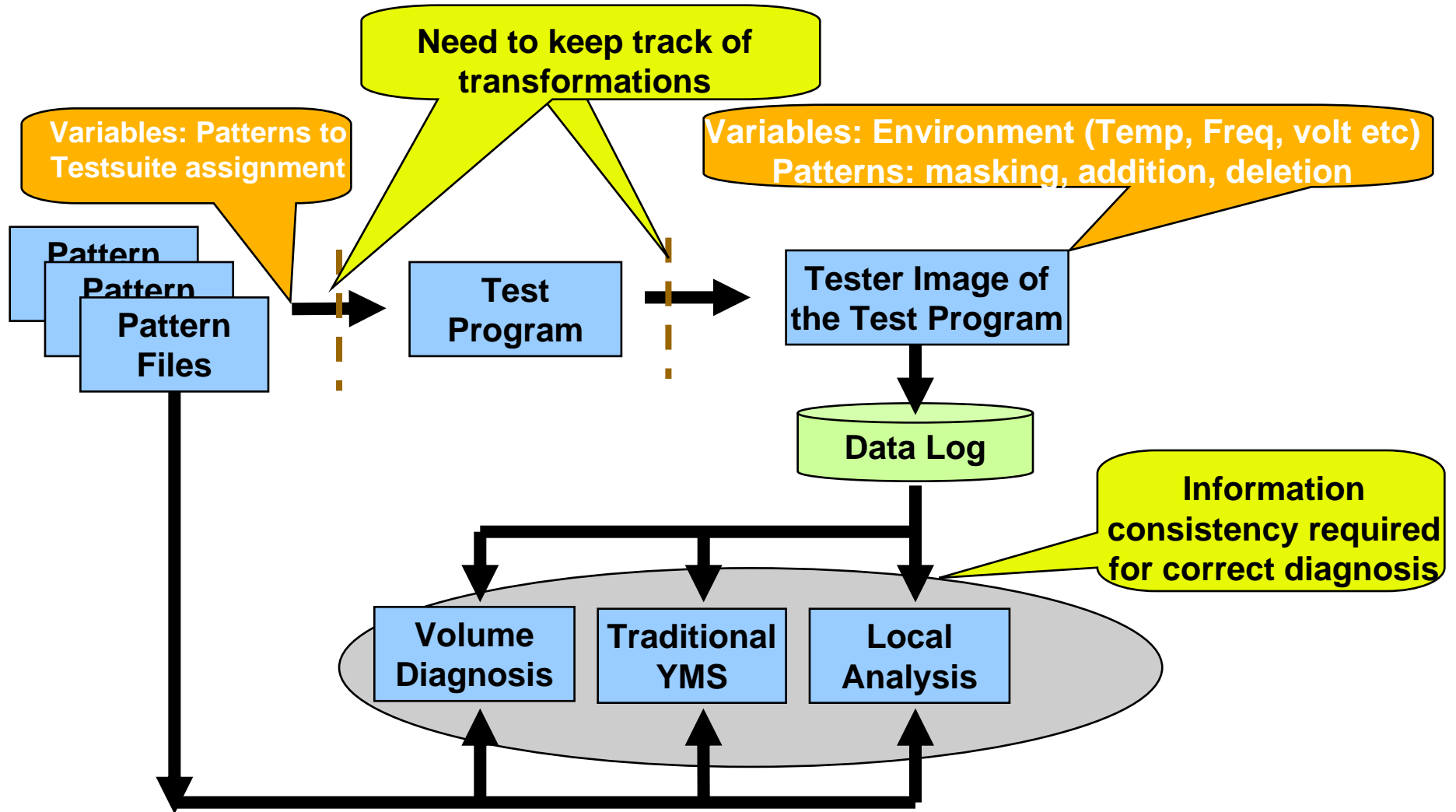
Conceptual Dataflow Diagram



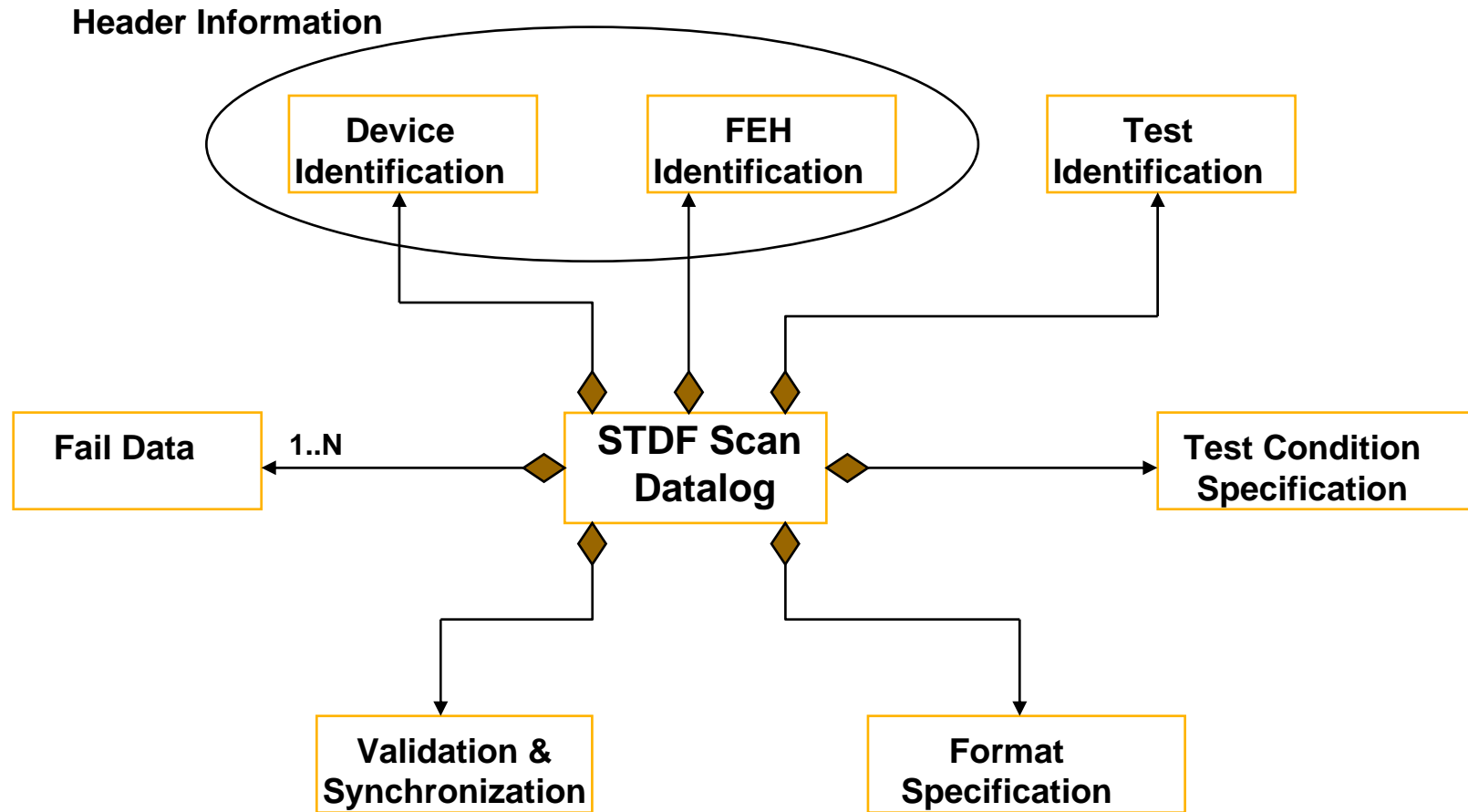
Simple TestFlow



Conceptual Dataflow Diagram



Scan Datalog Data Model



STDF V4 Quick Review

- Organized as series of records
- Record are identified by a type and a sub-type in the header

STDF Record Header

Each STDF record begins with a record header consisting of the following three fields:

Field	Description
REC_LEN	The number of bytes of data following the record header. REC_LEN does not include the four bytes of the record header.
REC_TYP	An integer identifying a group of related STDF record types.
REC_SUB	An integer identifying a specific STDF record type within each REC_TYP group. On REC_TYP and REC_SUB, see the next section.

STDF V4 Record Types

Major Type	Description
0	Information about the STDF file
1	Data collected on a per lot basis
2	Data collected per wafer
5	Data collected on a per part basis
10	Data collected per test in the test program
15	Data collected per test execution
20	Data collected per program segment
50	Generic Data
180	Reserved for use by Image software
181	Reserved for use by IG900 software

Proposal Structure

Purpose	Records
Version Identification	<div data-bbox="913 386 1356 428" style="border: 1px solid black; padding: 2px; text-align: center;">Version Update Record</div>
Per Lot Information	<div data-bbox="913 496 1356 539" style="border: 1px solid black; padding: 2px; text-align: center;">Pattern Sequence Record</div> <div data-bbox="913 555 1356 597" style="border: 1px solid black; padding: 2px; text-align: center;">Name Map Records</div> <div data-bbox="913 613 1356 656" style="border: 1px solid black; padding: 2px; text-align: center;">Scan Structure Records</div>
Device and Test Setup Information	<div data-bbox="913 691 1356 734" style="border: 1px solid black; padding: 2px; text-align: center;">Device Identification Records</div> <div data-bbox="913 750 1356 792" style="border: 1px solid black; padding: 2px; text-align: center;">Test Identification Records</div> <div data-bbox="913 808 1356 850" style="border: 1px solid black; padding: 2px; text-align: center;">FEH Identification Records</div>
Per Test Execution Information	<div data-bbox="913 919 1356 1052" style="border: 1px solid black; padding: 2px; text-align: center;">Scan Test Record</div> <div data-bbox="1108 1065 1136 1156" style="text-align: center;"> ■ ■ ■ </div> <div data-bbox="913 1175 1356 1308" style="border: 1px solid black; padding: 2px; text-align: center;">Scan Test Record</div>

How to Identify the file contains STDF V4-2008 data

- A record subtype is added to describe the presence of V4-2008 information

Old Record

0	Information about the STDF file
10	File Attributes Record (FAR)
20	Audit Trail Record (ATR)

- A record type 30 is added to indicate version of STDF V4-year using string

New Record

0	Information about the STDF file
10	File Attributes Record (FAR)
20	Audit Trail Record (ATR)
30	Version Update Record (VUR)

Version Update Record (VUR)

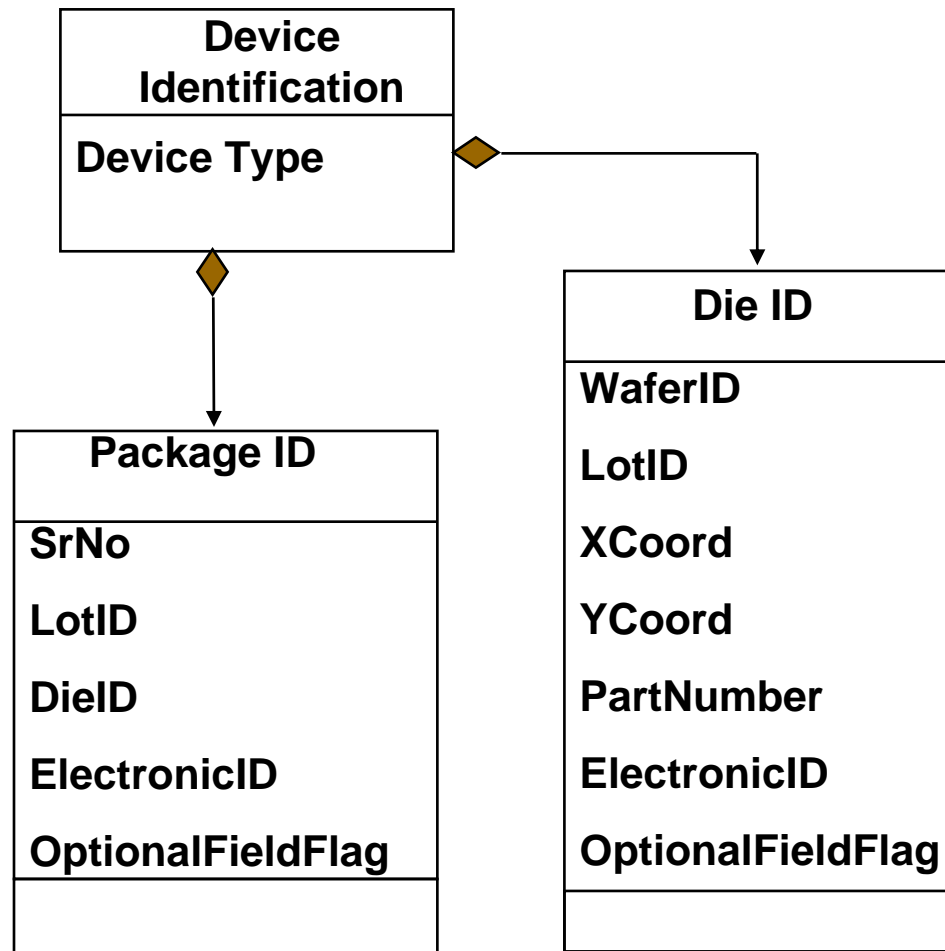
Function: Contains the Version Update Information

Field Name	Data Type	Field Description	Missing/Invalid Data Flag
REC_LEN	U*2	Bytes of data following header	
REC_TYPE	U*1	Record Type (0)	
REC_SUB	U*1	Record sub-type (30)	
NAME_CNT	U*2	No. of Entries	
UPDATE_NAM	C*n	Update Version Name	

Header Information

- Device Identification
- Equipment Identification

Device Identification



Purpose: Uniquely identify the die/part to enable

- Logged per device
- Statistical analysis
- Indexing for retrieval
- Correlation analysis with other measurements

Front-End Hardware

Test Identification
Site ID (E.g. Malaysia)
Cell ID (Test Cell ID)
Prober ID
LoadBoard ID
OptionalFieldsFlag

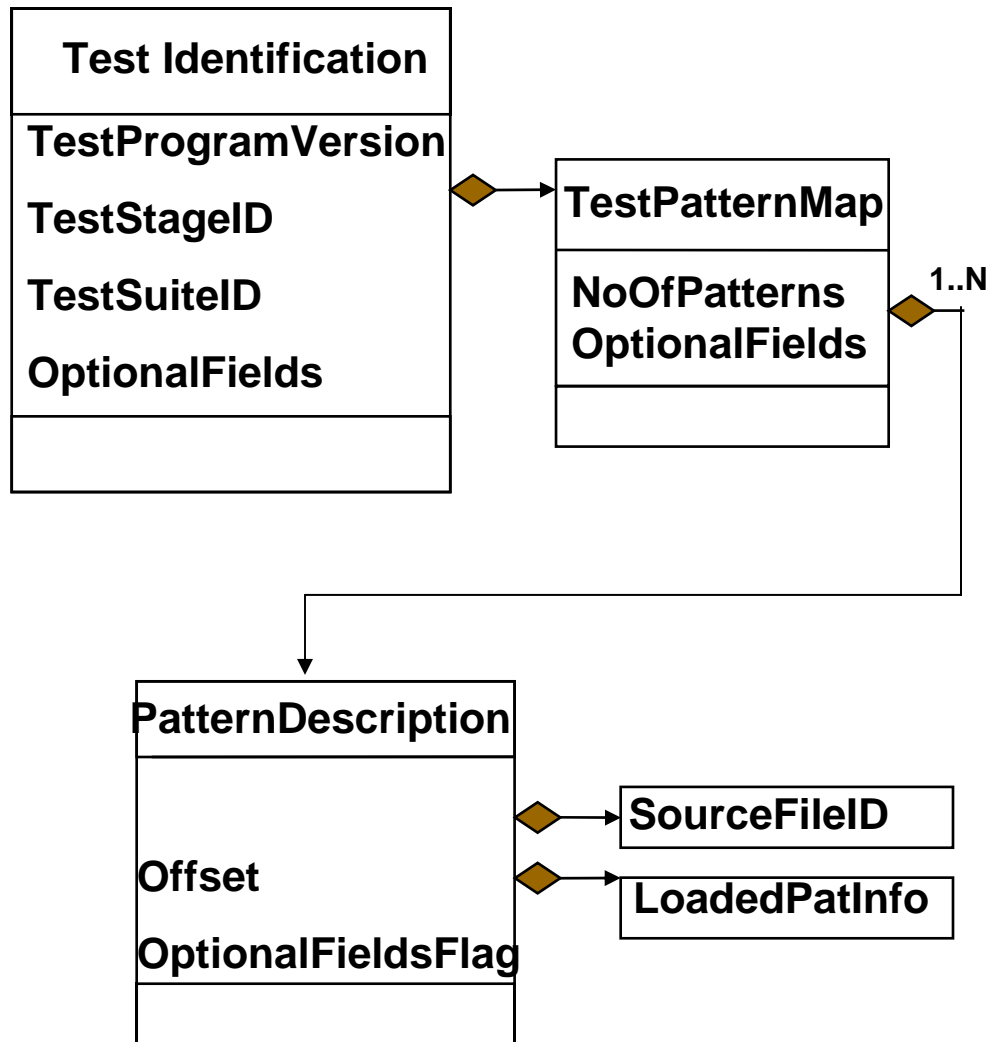
Purpose: To uniquely identify equipment used for testing

- Logged Per STDF file
- Indexing for retrieval
- Correlation analysis across equipment and sites

Header Records

- Proposal Use Existing STDF records for the Header information (Device ID, FEH, Test Location/Program Identification)
- Differences are padded to existing field
 - SrNo is added to the PartID String using "SrNo=Value" format (delimited by ";")
 - DieID/Electronic ID is similarly also added to PartID using the same syntax

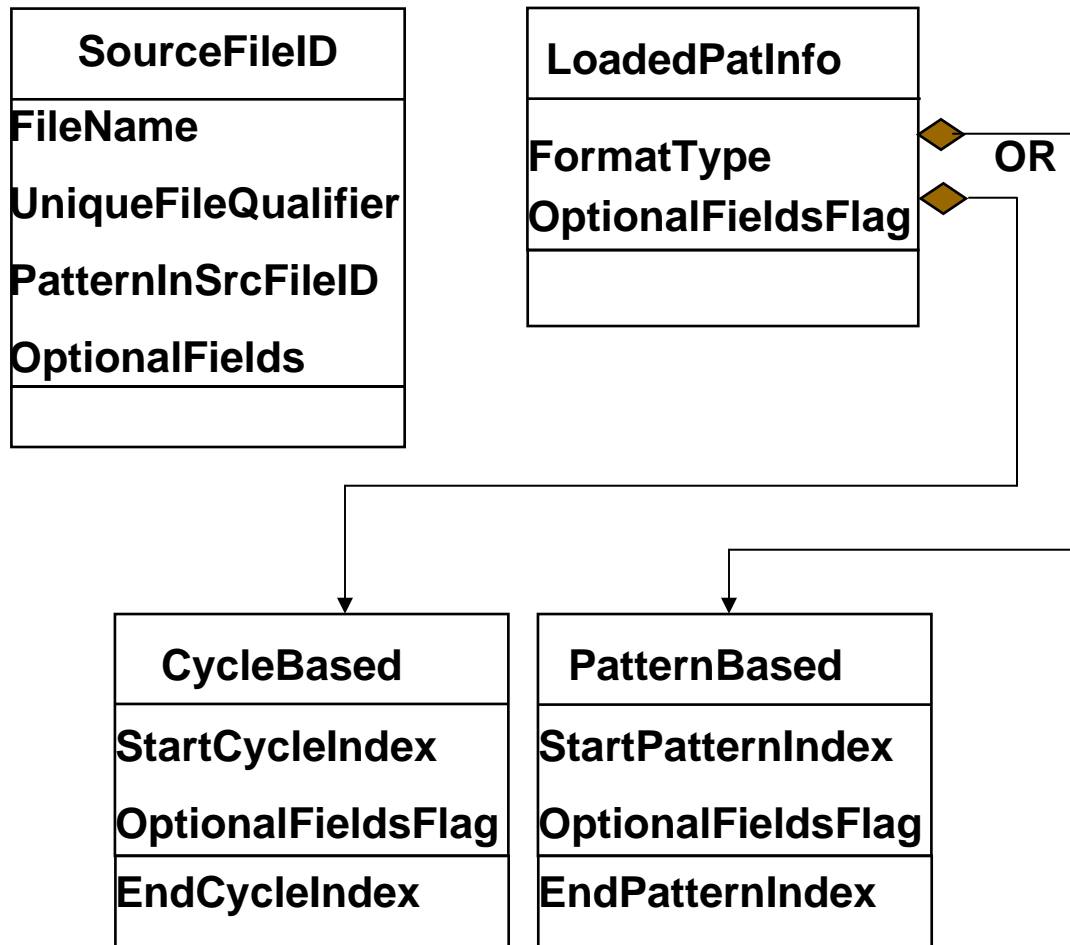
Test Identification



Purpose: To log the test data loaded on the ATE

- Logged per TestSuite
- Provide analysis tools with the tester image of the test data (mapping from individual patterns to test program(s))
- Provide links back to the source of test data
- Correlation analysis across test insertions/test stages

Test Identification



NOTE:

- **Source File ID** provides links to the source pattern(s) and files
- **LoadedPatInfo** provides the mapping of patterns to the test suites

Test Identification Mapping

- A new STDF record under record type 1
- New Record is called Pattern Sequence Record (PSR)

1	Data collected on a per lot basis
10	Master Information Record (MIR)
20	Master Results Record (MRR)
30	Part Count Record (PCR)
40	Hardware Bin Record (HBR)
50	Software Bin Record (SBR)
60	Pin Map Record (PMR)
62	Pin Group Record (PGR)
63	Pin List Record (PLR)
70	Retest Data Record (RDR)
80	Site Description Record (SDR)
90	Pattern Sequence Record (PSR)

Pattern Sequence Record (PSR)

Function: Contains information pertaining to the pattern profile of a specific executed scan test
 It is referenced by the STR (Scan Test Record) through it's PSR_IDX field.
 Specifies the source pattern files created by ATPG tools that constitute a specific scan test

Field Name	Data Type	Field Description	Missing/Invalid Data Flag
REC_LEN	U*2	Bytes of data following header	
REC_TYPE	u*1	Record Type (1)	
REC_SUB	u*1	Record sub-type (80)	
PSR_IDX	U*2	PSR Record Index (used by STR records)	
CONT_FLG	U*1	Continue Flag. Indicates this is a continuation of a preceding PSR record	
INCOMP_FLG	U*1	Incomplete Flag. Indicates that subsequent continuation STR records follow	
TOTAL_CNT	U*2	This field indicates the total no. of pattern information sets over all the PSRs	
LOCAL_CNT	U*2	No of Pattern Information sets in the current PSR	
PAT_BGN	U*8	Cycle # pattern begins on (Cycle #1 is 1st cycle in pattern execution)	Note 1
PAT_END	U*8	Cycle # pattern stops at	Note 1
PAT_FILE	C*n	Pattern File Name	Note 1
FILE_ID	C*n	Unique file identifier key	Note 1
SRC_ID	C*n	PatternInSrcFileID??	Note 1

Notes:

1. A test may consist of one or more pattern blocks derived from one or more ATPG files. The PSR_CNT field specifies the number of following pattern information sets contained within the record. Each pattern information set consists of the following five fields
 - PAT_BGN : The cycle count the specified ATPG pattern begins on. The 1st cycle executed is
 - PAT_END : The cycle count the specified ATPG pattern ends on.
 - PAT_FILE : The name of the ATPG file.
 - FILE_ID: Optional - Unique character string that uniquely identifies the file
 - SRC_ID: Optional - The name of the pattern block in the source file (??)

STDF Record Mapping for Per Test specific Objects

- Data Objects Covered:
 - Test Condition Specification
 - Format Specification
 - Validation and Synchronization
 - Fail data
- A new STDF Record called Scan Test Record (STR) is created under Major type 15

15	Data collected per test execution
10	Parametric Test Record (PTR)
15	Multiple-Result Parametric Record (MPR)
20	Functional Test Record (FTR)
30	Scan Test Record (STR)

Scan Test Record

- Multiple STRs are allowed to accommodate large amount of fails
- Inheritance over the multiple STRs is allowed
- Optional fields are controlled by flags

Scan Test Record Structure

Class of Information	Information Fields
Flags for Optional Fields	Continuation Flag, Incomplete flag
Test Setup Information	Test No, Test Head Number, Test Site, Test Flags Etc (Brought over from FTR)
Validation and Synchronization Information	Buffer fail status, Global Mask Status, Fail limit Info,
Test Condition Specification with Optional Fields Specification	Scan_Freq, Capture Frequency, Voltage, Optional
Datalog Format Specification	Z-handling, Data Field Map, Optional Field Map
Datalog with Per fail optional Fields	Pin Info, Pattern/Cycle no, Measured Data, Expected data, Fail data, optional data

Scan Test Record Details: Test Condition

- Test Conditions are specified as a attribute and value (with units) pairs represented in ASCII
- Reasons
 - Low in volume
 - Compatible with STIL
- Expressions will be support for the value
 - Expression string to start with '='
 - No forward reference beyond the current test condition block
 - Not a must for readers to support
 - Readers can/should ignore if not supported
- Exception: Temperature is not on a per test basis hence will be specified in MIR

Reserved Keywords

- Only two reserved Attr Names
 - SHIFT_FREQ
 - CAPTURE_FREQ
- If these keywords are used then the meaning is the scan frequencies used during the data capture

STDF Record Fields for Test Conditions

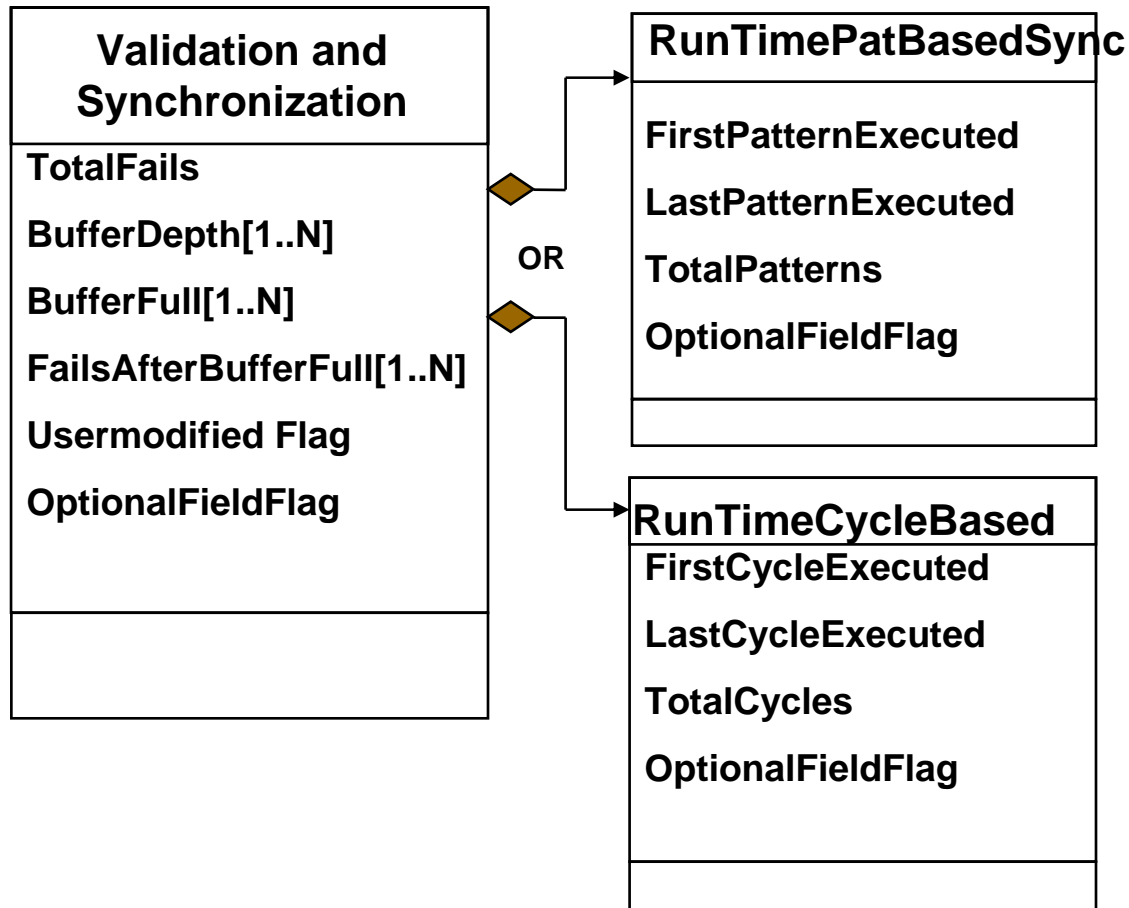
- Represented as two arrays of

COND_CNT	U*2	No of Test Conditions and optional data specifications	
ENV_TXT	kxC*n	Test Condition Description	COND_CNT=0
ENV_VAL	kxC*n	Test Condition Value	COND_CNT=0

- Test Conditions are optional
 - used only for correlation with other measurements
- Length value k = 0 indicates the absence of the test conditions

K
x

Validation & Synchronization



Purpose: To provide the analysis tools with information on execution status of device test to enable data integrity checks and synchronization

- Logged per test execution
- Data on any exception condition encountered during test
- Any buffer capacity related issues
- Any test data change

Validation and Synchronization

- Fields for communicating execution time info
 - Fail after buffer full
 - Mask map
 - Total cycles executed
 - Total fails
- Fields for communicating data transformation through the loop
 - Name Map
 - FlipFlop Name Map
 - Change Map

Fail After Buffer Full

- Field Name: Failure After Logging Map (FAL_MAP)
- Purpose: To detect the status of the pattern after logging is stopped as passing or at least one fail
- 1 bit per pin in the order of PMR indexes

FALMAP D*n Bitmap of failures after buffer full

- Optional field controlled by Flags (2 bits)

00 - No information of buffer status is provided

01 - All fails were logged, FAL_MAP not present

10 - FAL_MAP present

Masking Information

- Name: MASK_MAP
- Purpose: Do not assume the masked pins a passing
- Global masking information per pin
- Optional field, Controlled by 2 bit flag with following behavior

00 Nothing is masked (MaskMap not present)

01 Use the previous MASK_MAP definition
(MASK_MAP in the current record absent)

10 Use the MASK_MAP in the current record
and make is persistent

Buffer Limit

- Name: FAL_LIM
- Purpose: to communicate the current fail logging limits
- Both global and per pin limit specification allowed
- Default and override allowed for the per pin case
- Optional fields
- Description

LIM_CNT **U*2** **Size of Array; 1 for global case**

PMR_IDX **kxU*2** **First entry is always 0;**

LIM_SPK **kxU*4** **Fail limit for the PMR_IDX; 0 th entry
used for global/default case**

Validation and Synchronization Fields

TOT_FAILS	U*4	Total failures (pin x cycle) detected in test execution
LOG_FAILS	U*4	Total fails logged, including continuation records
CYC_CNT	U*4	Total cycles executed in test
MASK_MAP	D*n	Bit map of Globally Masked Pins
FAL_MAP	D*n	Bit map of failures after buffer full
LIM_CNT	U*2	Size of Array; 1 for global specification
PMR_IDX	kxU*2	Indexes of the pins; first location is always 0 and refers to all pins
LIM_SPK	kxU*4	Fail Limits for the PMR_IDXs; 0th location used for global/default value

NameMap

- Name: Signal Name Record (SNR)
- A New Record Sub Type (91) under Major type 1
- Contents: Pairs of PMR Indexes and ATPG Names
- Purpose: Provide the mapping of ATE names to ATPG names
- Optional:
 - only PMR info is available if alternate record is absent and logical name should/would contain the ATPGsignal name
 - If present then use the alternate Name records which would contain ATPG names and would replace the logic name from the PMR record

Signal Name Record

Signal Name Record (SNR)

Function: Contains information on the mapping of PMR indexes to the ATPG Names
It is used to store the mapping of ATE names to original ATPG names
If this field is not present then the logic names of the PMR record should contain the ATPG names

Field Name	Data Type	Field Description	Missing/Invalid Data Flag
REC_LEN	U*2	Bytes of data following header	
REC_TYPE	U*1	Record Type (1)	
REC_SUB	U*1	Record sub-type (91)	
NAME_CNT	U*2	No. of Entries	
PMR_IDX	kxU*2	PMR Index of the pin in the mapping	
ATPG_NAME	kxC*n	ATPG Signal Name for the pin	

FF Mapping

- It is done using a table
- A new record is introduced to handle the FF name table
- FF names are indexed by Chain no.(X) and FF no.(Y)
- Name: Scan Structure Record (SSR)
- Record Type: (Major Type = 1, SubType = 92)
- Optional: Needs to be written only if needed
- Contents: On The next slide

Scan Structure Record

Scan Structure Record (SSR)

Function: Contains information on the mapping of Scan Chain, Bit position to the FF name
 It is used to support the applications where the log requires the FF names e.g. Logic Vision
 This is an optional record.

Field Name	Data Type	Field Description	Missing/Invalid Data Flag
REC_LEN	U*2	Bytes of data following header	
REC_TYPE	U*1	Record Type (1)	
REC_SUB	U*1	Record sub-type (92)	
SSR_INDEX	U*2	SSR Index	
SSR_NAME	C*n	Name of the Scan Structure for reference	
CONT_FLG	U*1	Flags for continuation record 1:New and Cont; 2:Contd & Incomplete; 3:Contd & Complete	
FLOP_CNT	U*4	No of Entries	
CHAIN_NO	kxU*2	Scan Chain No	
FF_NO	kxU*2	Bit position	
FF_NAME	kxC*n	Name of the FF	

Format Specification

Format Specification
ZHandlingFlag
FailDataFormat
OptionalFieldsFlag

Purpose: To provide information to the datalog reader tools about the format of the datalog to follow. Allows handling of tester-specific handling of measured data

- Logged per test execution

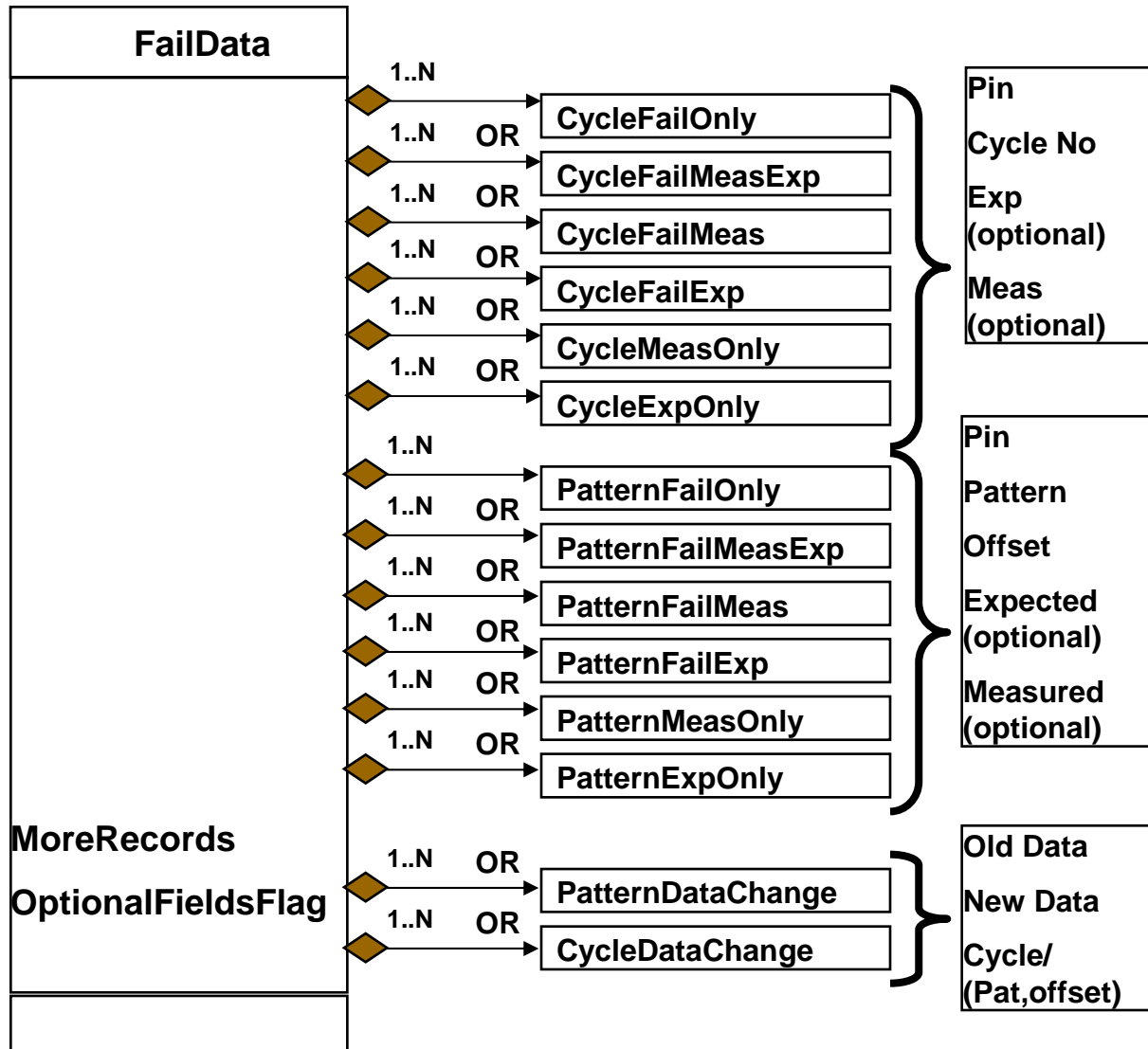
Z-handling values

Enumeration value	condition
0	Z mapped to L
1	Z mapped to H
2	Z mapped to Z
3	Z mapped to X
4	Not handled

- Handled by enumeration type in a byte
- Name: Z_FLAG
- Type C*1

Z_CHAR C*1 Z Handling character

Fail Data



Purpose: To log the data

- Logged per test execution
- Both cycle based and pattern based logging is supported
- Measured and expected data can optionally be recorded for synchronization purposes
- Also used for communicating pattern change data

Format Specification: Fail Data Format

- Fail data is implemented using a super record definition with all the possible fields.
- Each field is represented using an array
- Fields can be omitted from the super record using flags, the entire array is then skipped
- Fail data format is specified by setting the flag bits in the Flag field
- Name: DATA_FLG
- Type: U*1
- Specification: On the next page

Datalog Fields Supported

- Pin Id
- Cycle_no
- Pattern No
- Bit No
- Chain No.
- Pat Data
- New Data
- Measured Data

Datalog Usage

Cycle Based

PMR_INDEX

CYC_NO

Optional

PAT_DATA

CAP_DATA

Pat Based

PMR_INDEX

PAT_NO

BIT_NO

Optional

PAT_DATA

CAP_DATA

FF Map (/w SSR)

CHAIN_NO

BIT_NO

CAP_DATA

Cycle Based (Measured Only)

PMR_INDEX

CYC_NO

CAP_DATA

Cycle Based (Expected Only)

PMR_INDEX

CYC_NO

PAT_DATA

Fail data and Format Specifications

SSR_REF	U*2	SSR Index (Scan Structure Record)	
PSR_REF	U*2	PSR Index (Pattern Sequence Record)	
RCD_FAILS	U*4	Count (<i>k</i>) of fails logged in this record only	
Z_CHAR	C*1	Z Handling character	
OPT_FLG	U*1	Set of two bits specify the Optional fields present	
DATA_FLG	U*1	Individuals bits specify what data arrays follow in this record.	
CYC_NUM	kxU*4	Array of cycle numbers (All cycle formats)	DATA_FLG bit 0 = 0
PMR_IDX	kxU*2	Array of PMR Indexes (All Formats)	DATA_FLG bit 1 = 0
CHAIN_NO	kxU*2	Array of Chain No for FF Name Mapping	DATA_FLG bit 2 = 0
CAP_DATA	kxC*1	Captured Data (Cycle capture format)	DATA_FLG bit 3 = 0
PAT_DATA	kxC*1	Pattern vector data contained in the test pattern	DATA_FLG bit 4 = 0
NEW_DATA	kxC*1	New vector data (Vector change format)	DATA_FLG bit 5 = 0
PAT_NUM	kxU*4	Pattern # (Pattern format)	DATA_FLG bit 6 = 0
BIT_NUM	kxU*4	Chain Bit Position (Pattern format)	DATA_FLG bit 7 = 0

Multiple STRs can be used t if the number of fails exceed a single record limit

Optional Fields: Per Scan Test

- Stored using String representation
- Added to the test conditions (Attr,Value) pairs in two separate arrays
- Stored using multiple C*n fields

Optional Fields Specification: Per Fail

- Optional Fields per fail are stores as array just like other fail record fields
- Field type U*1, U*2, U*4 are allowed
- A new Flag field along the lines of DATA_FLG is used to indicate the data type and order of the optional fields
- Two bits in the flag are used to indicate the type of optional fields

01 – U*1

10 – U*2

11 – U*4

00 – End of optional records

Next Steps

- Document Available - Nov 1st, 2007
- Document review – Nov, 2007
- Prototype development (Vendors) - Jan 2008
- Tool-Tool flow verification – Mar 2008
- Customer Beta site verification – June 2008
(Mini production run)
- Regular product from Vendors - ???

Current End

Scan Test Record (STR)

Function: Contains all or some of the results of the single execution of a scan test in the test program. It is intended to contain all of the individual pin/cycle failures that are detected in single test execution. If there are more failures than can be contained in a single record, then the "INCMP_FLG" field will be set and one or more additional STR records will be added containing the remaining failure data.

Field Name	Data Type	Field Descri	Missing/Invalid Data Flag
REC_LEN	U*2	Bytes of data following header	
REC_TYPE	U*1	Record Type (15)	
REC_SUB	U*1	Record sub-type (30)	
CONT_FLG	U*1	(Bit 0 - CONT_FLG, Bit 1- INCMP_FLAG, Bit 2,3 - FALMAP Flag bits) Bits 4,5 - Mask_MAP Flag bits; Bit 7 - UserModifiedFlag	
TEST_NUM	U*4	Test Number	
HEAD_NUM	U*1	Test head number	
SITE_NUM	U*1	Test site number	
TEST_FLG	B*1	Test flags (fail, alarm, etc.)	
TEST_TXT	C*n	Descriptive text or label	length byte = 0
ALARM_ID	C*n	Name of alarm	length byte = 0
PROG_TXT	C*n	Additional Programmed information	length byte = 0
RSLT_TXT	C*n	Additional result information	length byte = 0
SPIN_MAP	D*n	Bit map of enables comparators	length byte = 0
TOT_FAILS	U*4	Total failures (pin x cycle) detected in test execution	
LOG_FAILS	U*4	Total fails logged, including continuation records	
CYC_CNT	U*4	Total cycles executed in test	
MASK_MAP	D*n	Bit map of Globally Masked Pins	length byte = 0
FAL_MAP	D*n	Bit map of failures after buffer full	length byte = 0
LIM_CNT	U*2	Size of Array; 1 for global specification	
PMR_IDX	kxU*2	Indexes of the pins; first location is always 0 and refers to all pins	
LIM_SPK	kxU*4	Fail Limits for the PMR_IDXs; 0th location used for global/default value	
COND_CNT	U*2	No of Test Conditions and optional data specifications	
ENV_TXT	kxC*n	Test Condition Description	COND_CNT=0
ENV_VAL	kxC*n	Test Condition Value	COND_CNT=0

SSR_REF	U*2	SSR Index (Scan Structure Record)	
PSR_REF	U*2	PSR Index (Pattern Sequence Record)	
RCD_FAILS	U*4	Count (<i>k</i>) of fails logged in this record only	
Z_CHAR	C*1	Z Handling character	
OPT_FLG	U*1	Set of two bits specify the Optional fields present	
DATA_FLG	U*1	Individuals bits specify what data arrays follow in this record.	
CYC_NUM	kxU*4	Array of cycle numbers (All cycle formats)	DATA_FLG bit 0 = 0
PMR_IDX	kxU*2	Array of PMR Indexes (All Formats)	DATA_FLG bit 1 = 0
CHAIN_NO	kxU*2	Array of Chain No for FF Name Mapping	DATA_FLG bit 2 = 0
CAP_DATA	kxC*1	Captured Data (Cycle capture format)	DATA_FLG bit 3 = 0
PAT_DATA	kxC*1	Pattern vector data contained in the test pattern	DATA_FLG bit 4 = 0
NEW_DATA	kxC*1	New vector data (Vector change format)	DATA_FLG bit 5 = 0
PAT_NUM	kxU*4	Pattern # (Pattern format)	DATA_FLG bit 6 = 0
BIT_NUM	kxU*4	Chain Bit Position (Pattern format)	DATA_FLG bit 7 = 0

Notes On Specific Fields:

SPIN_MAP This optional field contains an array of bits corresponding to the PMR index numbers of the enabled comparators. The 0th bit corresponds to PMR index 0, the 1st bit corresponds to PMR index 1, and so on. Each comparator that is enabled will have it's corresponding PMR index bit set to 1. Note that this field applies to the present record only plus any following continuation STR records). If this field is missing, then it is assumed that conventional pin enabling as defined in the source pattern source file specified in the referenced PSR records apply.

ENV_TXT This optional field is a string containing one or more test condition parameters/variables delimited by ";" characters, (e.g. "VDD1;VDD2;Period;Capture Freq;"). A corresponding value for each of these parameters is expected to be included in the following **ENV_VAL** field. If the **ENV_TXT** field is omitted, then the data from this field is *inherited* from the last **ENV_TXT** field read that had the same **TEST_NUM** value as this record.

ENV_VAL This optional field is a string containing one or more test condition values delimited by ";" characters, (e.g. "3.5V"; "700mV"; "22ns"; "333MHz;"). A corresponding value is expected for each parameter definition that is in the **ENV_TXT** (contained or inherited). Note that each value entry contains both a numeric and an engineering unit descriptor, and is consistent with the general rules for defining values in IEEE1450, with the exception that the enclosing "" characters are optional. This field also supports expressions as defined in IEEE1450. In general use of these fields it can be assumed that parameter values once defined for a specific test number are persistent and need not be repeated for each subsequent STR record.

TOT_FAILS	The total number of failures that were detected in this test, logged or not. A failure is defined as a pin and cycle, such as in three pins failed in the same cycle, that would be counted as 3 failures. If 0 then it should be inferred that this value was not determinable
CYC_CNT	The total number of cycles that were executed in this test. If 0 then it should be inferred that this value was not determinable
Z_CHAR	The waveform character that was substituted in place of the "Z" in the source pattern files.
LOG_FAILS	The total number of failures that were detected and logged in this test. A failure is defined as a pin and cycle, such as in three pins failed in the same cycle, that would be counted as 3 failures. This count includes the total failures from the test execution and includes any following continuation STR records.
RCD_FAILS	The total number of failures that were detected and logged just in this STR record.
DATA_FLG	A flag that specifies which data arrays are included in the subsequent remainder of this (and any continuation record)
CYC_NUM	An array of 4 byte cycle numbers that correspond to the number of failures logged in this record when logging in the "Cycle" mode (vs. the "Pattern" mode). The first cycle executed is always "1".
PMR_IDX	An array of PMR indexes that correspond either to the CYC_NUM array or to the PAT_NUM array
MEAS_DATA	An array of 1 byte characters that correspond to the CYC_NUM ("Cycle" mode") or PAT_NUM ("Pattern Mode") entries that indicate the data state actually actually observed on the tester pin.
CAP_DATA	An array of 1 byte characters that correspond to the CYC_NUM ("Cycle" mode") or PAT_NUM ("Pattern Mode") entries that indicate the "captured" data state. This function is generally the same as the MEAS_DATA except it is intended to infer that this data does not represent a failure, but rather the data states "captured" by the tester, e.g. in a scan data dump application.
PAT_DATA	An array of 1 byte characters that correspond to the CYC_NUM ("Cycle" mode") or PAT_NUM ("Pattern Mode") entries that indicate the data state that was specified in the source pattern file for this specific pin and Cycle or Pattern/Bit. This data has multiple applications: <ol style="list-style-type: none"> 1) Can be used when it is required to provide the expect data associated with each failure 2) Can be used to provide validation (when required) of the synchronization of cycle #'s to the source pattern file (as a note: Generally only a few initial representative cycles would be required for this function, and only once in an STDF file for each specific PSR utilization.) 3) Can also be used to represent the "Old" data states when used in conjunction with the following "NEW_DATA array when transmitting an STR record for the purposes of defining test pattern modifications

NEW_DATA	An array of 1 byte characters that correspond to the CYC_NUM ("Cycle" mode") or PAT_NUM ("Pattern Mode") entries that indicate any test pattern data modifications made to the original data in the source patterns.
PAT_NUM	When used in the "Pattern" format, an array of 4 byte integers specifying the pattern number. The 1st pattern executed is always "1"
BIT_NUM	When used in the "Pattern" format, an array of 4 byte integers specifying the bit position in the scan chain. The 1st bit position is always "1"

Issues

- Test Program information
- Reserved words for scan freq, capture_freq etc
- Examples of fields
- Do we still Need SPIN_MAP v/s MASK_MAP
-

END

