A Block Based Test Methodology (BBTM) for Verigy 93K Platform

Subbarao Jaldu, Albert Alcorn, Test Engineering

***Abstract*—The 93K graphical testflow interface is difficult to develop, navigate and debug large test programs. Making a testflow change is tedious, error-prone since test conditions are hidden inside testsuites. Block-based test methodology (BBTM) reduces testsuite count and simplifies testflow management using CSV files. It provides programming flexibility with C++, standardizes user-created IP blocks for IP reuse, and creates customizable datalogs which can be imported easily into statistical analysis tools such as JMP. BBTM is implemented on 93K using Test Methods and three input CSV files that contain flow, parameter, and setup information. BBTM is used to successfully characterize five diverse Cypress product families across 3 divisions. BBTM programs are standardized, readable, easy to manipulate. A single testflow can support both production and characterization needs of new silicon. There are currently 45 tIP modules that are used to characterize AC, DC & Analog parameters. The char conditions, tIP and testflow are independent blocks that allow users to add new tests seamlessly with variable levels of high quality data collection at several conditions much faster compared to any other system. BBTM allows the users to port the test techniques very quickly from Test Chips to Full Chip by simply adding a test case in the CSV file.**

BACKGROUND

Verigy 93K Platform is one of the most powerful test platforms available at Cypress. This tester is very versatile and gives a lot of flexibility to control the resources required for a test. The biggest bottle neck on this tester is the GUI based testflow which is hard to develop, understand and interpret.

Typical test program development cycle is primarily dependent on the complexity of the device and the ATE development interface. The secondary impact can be directly associated to the following:

- TE development style : Affects debug
- No IP Reuse : Longer development cycle, Redundant Learning
- Single resource allocation : No concurrent development.

Based on this, Char/Production test development becomes a highly unpredictable task to meet the PR4 requirements.

INTRODUCTION

The Cypress product portfolio started getting very diversified with the introduction of the new mantra: programmability. The devices started becoming versatile supporting multiple interfaces and protocols with the flip of a bit. This drastically increases the char development cycle. Every parameter can add 3-5 days to the char cycle time. This problem was clearly evident during the WB-Antioch char program development cycle which triggered the DCD test engineering team to create a software platform that addresses the following issues:

- ✓ Provides a Simple Interface
- ✓ Complete reuse of tIP
- ✓ Supports Flex resources
- ✓ High Quality test program
- ✓ Portability
- ✓ Standardized test Functions
- ✓ Standardized output format
- ✓ Supports all versions of 93k platform

IMPLEMENTATION

The Test Methodology (BBTM) is targeted to use across several device families at different manufacturing steps. It was intended to be modularized for better understanding and reuse. The following are the key functional components that are integral to this software.

1. TRD (Technical Requirements Document) : This simple to read file replaces the GUI based test flow. This flow encapsulates all the setup conditions and the flow organization at different mfg. steps (which includes Pre Silicon Replay, Char, Sort, Class, QA). TRD is device specific and can be shared with in the device family ex: chops

2. tIP (Standard Test Functions) : These high quality functions are written to perform very specific set of tasks required to test various parameters. These are easy to understand and are standard across all devices. These are developed by a pool of very experienced test engineers.

3. ITERATOR : This is the central nervous system of the BBTM that controls the whole program. This brings the TRD and tIP together to execute a complete test program that confirms to the TRD requirements using the well known tIP.

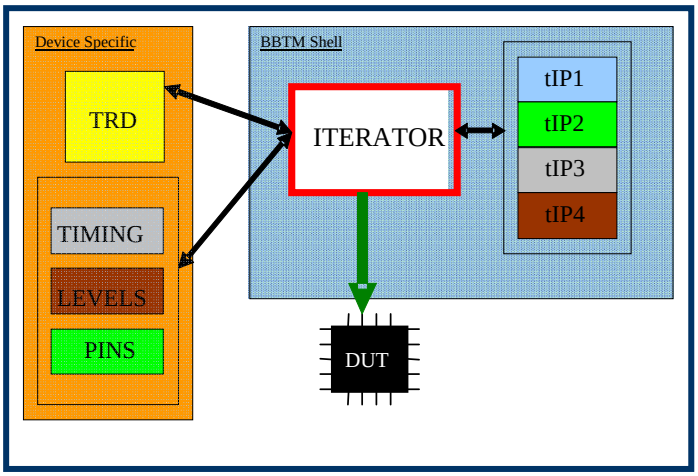
4. PRIMARIES : These are device specific setup files in 93k format that contain the Pins, Levels & Timing information. These files are device specific.

5. DATALOGS : The output datalog follows the generic format of Cypress *DATA_BROWSER* which makes it easy for data mining and char analysis.

A Program developed in this style will have a BBTM Shell and device specific components. The Shell consists of Iterator and tIP blocks that are developed by a core team of DCD test engineers. These blocks went through a 5 staged review process to meet the specific tIP definition and confirm to efficient functional and coding standards. The Shell is then thoroughly tested and released to the BBTM IP Library.

A new user will need spend very minimal time to learn the usage of BBTM Shell for the first product and then spend all the time efficiently on Device Specific development. The TE will need to develop the TRD in a CSV format, Pins, Level setups & Timing setups. This will allow the TE to have more time to learn the Device specifics rather than the tester setup.

Figure 1 Illustrates BBTM implementation.



RESULTS

This software truly provides a platform that is simple to develop and understand. It provides extreme flexibility to support concurrent development. The csv style of interface allows non TE resource to create customized derivative programs on the fly for engineering options. The table below shows the improvement in the content of the code by using the BBTM flow.

		Device1 (Old Flow)	Device2 (BBTM Flow)
DEVICE FACTS	PARAMETERS	30	150
	VDD COMBINATIONS	6	25
	PATTERNS	50	200
PROGRAM FACTS	FILE SIZE (BYTES)	6256459	6400
	TOTAL PAGES	2429	40
	No. of Lines	69854	2104

Although the above table clearly shows that BBTM program is smaller compared to older style, the biggest advantage is the simplicity and readability of the code. Typically flex resourcing is not a productive option since all test engineers will have a learning curve associated with every new program. The BBTM style drives this learning curve to a near zero since TEs are knowledgeable about the Iterator and most tIPs. They are required to learn only new tIPs and read the TRD. This is truly evident in the WB-Benicia (parameters=700, patterns=1000) char development where 4 engineers are simultaneously developing the char program with minimal learning curve.

CONCLUSIONS

The success of this Software can be measured by its adoption rate. Since the inception of this tool, several test engineers started using it at 4 global locations (CSJ, CYWA, CMI, INDC) across 3 different divisions. BBTM is used to develop test programs for 6 different chips in the last one year. There are 45 unique tIPs now and the list will continue to grow as the need arises.

References

[1] Subversion Online Documentation : http://svnbook.red-bean.com
[2] Tortoise SVN : http://tortoisesvn.tigris.org