Schlumberger Automated Test Equipment

ASAP Parallel Testing Setup

Version 0.1

August 2, 2002

Table of Content

1 Introduction

This document describes the DeFT's parallel testing programming techniques.

2 ASAP Environment Setup

To enable a test program to run in parallel mode, you will need:

To add into your .ASAP_defaults or test_program.ASAP_defaults file the environment variable ASAP_PARALLEL_TEST.

setenv ASAP_PARALLEL_TEST TRUE

3 ASAP Programming for Parallel Test

As for a single test program, the parallel test will support multiple package.

All Pindefs informations except the tester channel are taken from the test program side.

The tester channel information are taken from the parallel test loadboard file <program_name>.lbd. This file should be located in the same directory as the executable .b9k[rs]4 file.

3.1 Load Board File

The load board file is used to define the tester channel assignment for all sites under test in parallel mode.

An executable test program becomes parallel if the environment variable ASAP_PARALLEL_TEST is defined as TRUE and a <program_name>.lbd file is found at load time.

3.1.1 Format

It must have the same name than the test program with the extension .lbd In fact only the Pindef has to be expanded.

Use the keyword NUMBER_OF_SITES = n;

to specify the number of site. This allow to test the syntax for each pindef.

If the NUMBER_OF_SITES is greater than the number of sites detected in the pindef, an error message is displayed in the test program IO at load time.

The syntax for each pindef of the .c9k is as follow:

```
pindef_name(0..x) = {
        {tc0_site0, tc1_site0, ..... tcx_site0},
        {tc0_site1, tc1_site1, ..... tcx_site1},
...
        {tc0_site1, tc1_site-n, ..... tcx_site-n}
```

```
}
```

The tester channel for site0 overwrite the one defined in the .c9k file. In fact the tester channel definition of the .c9k file is never used in parallel mode.

If the .c9k file defines more than one package, then the .lbd file must define the same number of package.

3.1.2 Example of a .lbd file

Pindef of the .c9k file:

```
pindef mega1 {
      <02:23:1995 11:35:02>,
      PACKAGE DIP18 {
            ras         = IN ,F1 ,HEX  ,{60},{"3"},
            cas         = IN ,F1 ,HEX  ,{52},{"16"},
            add(0..9)  = IN ,F2 ,ADJACENT,HEX,
                        {40,42,44,46,32,34,36,38,48,50},
                        {"5","6","7","8","10","11","12","13","14","15"},
            wr          = IN  ,F1  ,HEX  ,{58},{"2"},
            data_in       = IN  ,F1  ,HEX  ,{56},{"1"},
            data_out      = OUT ,F1  ,HEX  ,{54},{"17"},
            tf            = IN  ,F1  ,HEX  ,{62},{"4"},
            vcc            = PWR ,{DPS0},{"9"},
            vss            = PWR ,{GND},{"18"}
      },
      PINGROUP {
            all_in   = {data_in,wr,ras,tf,add,cas},
            all_out  = {data_out},
            all_pin  = {all_in,all_out}
      }
}; /* end of PINDEF_TABLE mega1 */
```

Pindef in the .lbd file

```
NUMBER_OF_SITES =2;

pindef mega1 {
      <2:23:1995 11:35:2>,
      PACKAGE DIP18 {
            ras        = {{18},{178}},
            cas        = {{26},{186}},
            add(0..9) =
                        {{6,4,2,0,14,12,10,8,30,28},
                        {166,164,162,160,174,172,170,168,190,188}},
            wr         = {{20},{180}},
            data_in    = {{22},{182}},
            data_out   = {{24},{184}},
            tf         = {{16},{176}},
            vcc        = {{DPS1},{DPS5}},
            vss        = {{GND},{GND}}
 }
}; /* end of pindef table */
```

3.2 Strategy File

The strategy file is used to define all specifics parallel test informations.

After the load board file has been loaded, if a file <program_name>.stg is found in the same directory as the executable .b9k[rs]4 file then it is loaded as the strategy file for this test program

This file is a source file and can be created with an editor.

The <>.stg file can define more than one strategy for this test program. In that case the strategy to be used can be defined using strategytool or by defining the environment variable ASAP_PARALLEL_STRATEGY in the .ASAP_defaults file.

Syntax:

setenv ASAP_PARALLEL_STRATEGY <strategy_block_name>

If this environment variable is not set then the first strategy block in the strategy file is used as default strategy.

3.2.1 Purpose of the Strategy_block

- Which test or group of test has to be executed in Serial / Parallel mode. The default value is Parallel test for all tests.
- Which test has to be executed again in Serial mode if it fails in Parallel mode.
- In case of Serial mode what is done on the other devices while one is under test.
- Define all global variables which need to be expanded. Expanded mean a global variable which will hold a value per site under test.
- Points of synchronization in the test flow.

3.2.2 STRATEGY Blocks Examples

```
strategy First_Strategy {
    <08:09:1994 12:03:43>,
    PINDEF_TABLE = any_one,
    GLOBAL_VARIABLE { "Epsilon", "Delta_1"},
    TEST_STRATEGY {
        {
        TEST = ALL_FTEST,
        SERIAL = YES,
        OTHERS_LEVEL = PowerDown
        },
        {
        TEST = Func_Match_Test,
        SERIAL = IFFAIL,
        IFFAIL_LEVEL = PowerDown
        }
    }
    SYNCHRONIZING_SEGMENT {
        PMU_2_SEG,
        FAST_TEST_SEG,
    },
}; /* End of block */

strategy Second_Strategy {
    <08:09:1994 12:06:05>,
    PINDEF_TABLE = any_one,
    TEST_STRATEGY {
        {
        TEST = ALL_SEARCH,
        SERIAL = NO,
```

```
            IFFAIL_LEVEL = PowerDown
            },
            {
            TEST = ALL_FTEST,
            SERIAL = NO,
            IFFAIL_LEVEL = PowerDown
            },
            {
            TEST = Check,
            SERIAL = IFAIL,
            OTHERS_LEVEL = PowerDown,
            IFFAIL_LEVEL = PowerDown
            }
        },
}; /* End of block */
```

3.2.3 STRATEGY Block Structure Description

strategy *Id* **{**	Keyword used to declare a strategy block. This one is followed by an identifier which is the name of the block. Only one strategy block is used by the TP but several blocks could be describes in development phase.
< Date and Time >	Allows to date the strategy block. The format of date and time is the following: Month**:** Day**:** Year Hour**:** Minute**:** Second with Month [01 .. 12]; Day [01 .. 31]; Year [1973 .. n]; Hour [00 .. 23]; Minute [00 .. 59] and Second [00 .. 59]
PINDEF_TABLE = *Id*	The PINDEF_TABLE field is a dummy field. This information will not be used.
TEST_STRATEGY {	Keyword which specifies that the following braces encompasses information describe the running mode of some tests and the behaviour in case of fail of some tests.
TEST = *Id or Type*	If set to ALL_FTEST, ALL_AC, ALL_DC, ALL_PMU_DC, ALL_DPS_DC, ALL_HCDPS_DC, ALL_ICOMP_DC, ALL_VOH_DC, ALL_VOL_DC, ALL_IOH_DC, ALL_IOL_DC, ALL_VIH_DC, ALL_VIL_DC, ALL_SEARCH or ALL_PLOT, all the test of type FTEST, AC, DC, SEARCH, PLOT will be concerned with the following information. If set to a test identifier, only that test will be take into account.

Note: A single test information overwrote the information of the group in which this one belongs.

SERIAL = *Flag*	If set to NO (which is the default flag), the SERAIL execution mode for the concerned

	test is not allowed even if that test failed. The PARALLEL mode is set and forced. If set to YES, the PARALLEL execution mode (all sites tested in parallel), is disabled. The SERIAL mode is set and forced on the concerned test. If set to IFFAIL, the PARALLEL mode is set until a fail is detected. The PARALLEL mode is set but not forced.
OTHERS_LEVEL = *Id*	This field is used to define the level to apply to non tested sites in SERIAL mode. By default, this is the actual level block which is applied.
IFFAIL_LEVEL = *Id*	This field is used to define the level to apply to a fail device. By default, this is the level at the END of the segment.
}	
GLOBAL_VARIABLE {	Keyword which specifies that the following braces encompasses information pertaining to the global block. It contains the list of the global variables to be expanded versus the number of sites. It allows to maintain site per site parameters.
}	
SYNCHRONIZING_SEGMENT {	Keyword which specifies that the following braces encompasses a list of segments that will be used as points of synchronization in the test flow.
}	
};	

3.3 Test flow priority control

Intent of test flow priority control is to give user a way to control pass/fail site execution priority per global basis or per segment basis other than using post user function code return status.

Default pass/fail site execution priority prior the Raptor release is:

- Take next segment pointed by global pass/fail execution status of the current segment executed at end of the segment. (global pass/fail status is set to fail if there is at least one site failed, which is 0. Thus failed site is always processed first than passed site.)
- Awaiting segment is kept in queue because of different execution status of the sites executed and failed segment is executed first. When this awaiting segment in queue is processed, next segment pointed by highest execution status of suspended sites are taken as next executable segment.

With new scheme, global or per segment pass/fail port priority can be set from strategy block by following rule:

3.3.1 New syntax in strategy file

New keyword is introduced to define flow priority sub block in strategy file syntax are: FLOW_PRIORITY, SEGMENT, ALL_SEGMENTS and four flags keyword P1_FIRST_P0_LAST, P0_FIRST_P1_LAST, HIGH_TO_LOW, LOW_TO_HIGH

```
FLOW_PRIORITY {
        {
        SEGMENT = ALL_SEGMENTS,
        P1_FIRST_P0_LAST,
        HIGH_TO_LOW
        },
        {
        SEGMENT = Test_2,
        P0_FIRST_P1_LAST,
        LOW_TO_HIGH
        }
},
```

Meaning of four flags are as follows:

a. P0_FIRST_P1_LAST forces port0 to be taken first and port1 to be taken last.
b. P1_FIRST_P0_LAST forces port1 to be taken first and port0 to be taken last.
c. LOW_TO_HIGH or HIGH_TO_LOW control the rest of port priority
d. Default scheme is to take port0 first and then go from highest to lowest port.

Combination of four flags and port priority are shown below with four exit port segment example:

	(Default)	P0_FIRST_P1_LAST	P1_FIRST_P0_LAST
(Default)	0, 3, 2, 1	0, 3, 2, 1	1, 3, 2, 0
LOW_TO_HIGH	0, 1, 2, 3	0, 2, 3, 1	1, 2, 3, 0
HIGH_TO_LOW	3, 2, 1, 0	0, 3, 2, 1	1, 3, 2, 0