Design and Analysis of Multicycle Test Set Based on a Two-Cycle Test Set With Constant Primary Input Vectors for Increased Fault Coverage

V. Sunilkumar[1], Mr M. Devadas[2]
[1] PG Scholar, Department of ECE, Vaagdevi College of Engineering, Bollikunta, Warangal.
[2] Assistant Professor, Department of ECE, Vaagdevi College of Engineering, Bollikunta, Warangal.
Email - sunilkumarwgl.v@gamil.com, das.deva@gmail.com

Abstract: In this paper we are implementing the Multicycle Test Set with the verified S298, S641 Test bench circuit with more accurate fault coverage. Test compaction can be achieved by using multicycle tests. To avoid the computationally intensive process of sequential test generation, multicycle tests can be generated by extending two-cycle tests. However, the scan-in state of a two-cycle test is not always effective for a multicycle test when the primary input vectors are held constant during the functional clock cycles of a test. This paper studies the extent of this issue by considering exhaustive two-cycle and multicycle test sets with constant primary input vectors for finite-state machine benchmarks. Based on the results of this study, it describes an efficient test compaction procedure that modifies selected two-cycle tests in a given test set in order to make them more effective as a source for multicycle tests with constant primary input vectors. Experimental results are presented to demonstrate the importance of this step to test compaction. Test generation procedures for n -detection test sets improve the quality of a test set by adding tests that increase the numbers of detections of target faults. A different approach to n-detection test generation increases the numbers of detections of target faults within the bounds of the number of tests of a single-detection test set. Multicycle tests provide the flexibility of improving the quality of a test set by increasing the number of clock cycles in each test, without increasing the number of tests. Improved test quality is thus achieved with limited increases in test application time and test data volume due to the larger numbers of clock cycles in each test. This paper describes a procedure that starts from a compact one-detection single-cycle test set for single stuck-at faults and produces a multicycle test set with the same number of tests, but increased numbers of clock cycles and improved test quality. The procedure uses only one-detection fault simulation of single stuck-at faults. A similar procedure is applied starting from a two-cycle test set and considering transition faults. The procedures produce tests with constant primary input vectors to accommodate tester limitations.

Key Words: Broadside tests, multicycle tests, test compaction, test generation, transition faults.

INTRODUCTION:

One of the significant challenges to RTL designers is to identify complete timing exceptions upfront. This becomes an iterative process in complicated designs where additional timing exceptions are identified based upon critical path or failing path analysis from timing reports. This approach leaves a significant number of timing paths which may not be real, but these never get identified, since they may not come up in the critical path report. However, synthesis and timing tools will continue to expend resources optimizing these paths when it is not needed. At the same time, it can also impact area and power consumption of the device. The intent of this document is to provide examples of false and multi cycle path exceptions that are easily missed by even experienced designers, and are identified through iterations on timing reports. Broadside tests are used for detecting delay faults in standard-scan circuits. A broadside test starts with a scan-in operation. This is followed by two consecutive functional clock cycles. The first functional clock cycle is applied under a slow clock in order to allow signal-transitions that started because of the scan-in operation to subside. The second functional clock cycle is applied under a fast clock in order to activate delay faults and capture their effects. The test ends with a scan-out operation. In a multicycle broadside test, there may be more than two functional clock cycles between

the scan operations. For at speed test application, the first functional clock cycle is applied under a slow clock. The remaining functional clock cycles are applied under a fast clock.
Multicycle scan-based tests have several applications. They were shown to be useful for circuits with multiple clock domains, for test compaction, to enhance defect detection, and to avoid over testing of delay faults [1]–[12]. Test compaction was considered in [1], [2], [9], and [11]. The use of multicycle tests contributes to test compaction as follows. A functional clock cycle between the scan operations of a test is associated with a present-state and a primary input vector that together define an input pattern to the combinational logic of the circuit. With more patterns between the scan-in and scan-out operations, and by applying these patterns at speed, the test can detect more faults. This allows the number of tests to be reduced. A reduction in the number of tests implies a reduction in the number of scan operations required for applying the test set. This contributes to a reduction in the test application time. It also reduces the test data volume since fewer scan-in and scan-out states need to be stored.

The procedure from [1] generates a compact multicycle test set targeting single stuck-at faults. The procedure from [2] compacts a given single-cycle test set by combining pairs of tests into multicycle tests targeting single stuck-at faults. The procedure from [9] generates a compact multicycle test set targeting transition faults starting from a two-cycle broadside test set. The procedure extends the tests into multicycle tests by adding subsequences of primary input vectors, which are applied in functional mode, between the scan operations of the tests. It then removes tests that become unnecessary. The procedure from [11] also starts from a given two-cycle test set. It extends a test into a multicycle test by adding runs of the same primary input vectors to create a special type of test that allows changes in the primary input vector to occur under a slow clock.

The advantage of starting from a single-cycle or two-cycle test set is that generating multicycle tests directly requires sequential test generation, which is significantly more computationally intensive than the generation of single-cycle or two-cycle tests. Specifically, for a circuit with G lines, a multicycle test with L clock cycles requires the test generation procedure to consider GL lines. The number of lines is G for a single-cycle test and $2G$ for a two-cycle test. Since the worst-case computational complexity of test generation (for example, using the D-algorithm) is exponential in the number of lines, avoiding the use of a test generation procedure for multicycle tests is advantageous. The procedures from [2], [9], and [11] require conventional test generation for the single-cycle or two-cycle test set. They obtain multicycle tests without performing sequential test generation.

The multicycle tests generated by the procedures from [2], [9], and [11] use scan-in states and primary input vectors from a given single-cycle or two-cycle test set. Some of the scan-in states may not be effective as scan-in states of multicycle tests. For circuits where many of the scan-in states of a single-cycle test set are not effective as scan-in states of multicycle tests, the procedure from [2] produces test sets where most of the tests are single-cycle tests. As a result, it does not benefit fully from the ability of multicycle tests to provide test compaction. The procedure from [9] compensates for this effect by allowing arbitrary primary input subsequences, which are not based on the two-cycle test set, to be applied during the functional clock cycles between the scan operations of a multicycle test. The procedure from [11] also allows a multicycle test to include several different primary input vectors.

The option of compensating for the scan-in states with arbitrary primary input subsequences does not exist when the primary input vector is held constant during all the functional clock cycles of each test. This is sometimes necessary for addressing tester limitations that prevent primary input vectors rom being changed at the speed of a fast clock during a test.

This paper describes an approach for addressing this issue without performing sequential test generation. This paper consists of two (independent) parts. The first part considers exhaustive two-cycle and multicycle test sets with constant primary input vectors for transition faults in finite-state machine benchmarks. These test sets demonstrate the extent to which scan-in states from a two-cycle test set are effective as scan-in states for a multicycle test set. The second part of this paper describes an efficient test compaction procedure that generates a compact multicycle test set for transition faults starting from a given compact two-cycle test set.
This test set can be generated by any test generation procedure for two-cycle tests (the test compaction procedure does not use exhaustive test sets and it is applicable to larger circuits). The test compaction

procedure includes a step where it modifies a scan-in state, and the corresponding primary input vector, in order to make them more suitable for a multicycle test. This step is applied selectively when it appears that a two-cycle test is not effective as a source for a multicycle test. Experimental results demonstrate the importance of this step.

The modification of scan-in states can also be applied with the test compaction procedures from [2], [9], and [11]. However, it is more important when the primary input vectors are held constant during the functional clock cycles of a test. The test compaction procedure that modifies scan-in states is expected to provide higher levels of test compaction than a sequential test generation procedure with dynamic test compaction for the following reason. A sequential test generation procedure with dynamic test compaction uses only unspecified values of a test to detect additional faults. The modification of a scan-in state allows specified values to be complemented if this leads to the detection of more faults. The possibility of complementing specified values provides a higher degree of flexibility for the procedure to detect more faults with every multicycle test.

To address the constraints of a test data compression method [13]–[16], the two-cycle test set can be generated under such constraints. The modification of a two-cycle test can be performed under the same constraints. Specifically, a modification can be avoided if the resulting test does not satisfy the constraints. The application of several consecutive functional clock cycles at-speed under a multicycle test requires a delay fault model where the extra delay of a fault is considered explicitly [17], [18]. In this paper, transition faults with an extra delay of a single clock cycle are considered.

EXHAUSTIVE TEST SETS:

This section studies the extent to which the scan-in states that are effective for a two-cycle test set are also effective for a multicycle test set by considering exhaustive test sets for finite-state machine benchmarks.

TABLE I
EXHAUSTIVE TEST SETS FOR *bbsse*

i	s_i	2	3	4	5	6	7	8
4	0100	1	1	1	1	1	1	1
6	0110	1	1	1	1	1	1	1
8	1000	1	1	1	1	1	1	1
13	1101	1	1	1	1	1	1	1
14	1110	1	1	1	1	1	1	1
1	0001	1	1	0	1	1	1	1
2	0010	1	1	0	0	0	0	0
3	0011	1	0	0	0	0	0	0
5	0101	1	0	0	0	0	0	0
7	0111	1	0	0	0	0	0	0
10	1010	1	0	0	0	0	0	0
11	1011	1	1	1	0	0	0	0

TABLE II
EXHAUSTIVE TEST SETS FOR *b*01

i	s_i	2	3	4	5	6	7	8
4	00100	1	1	1	1	1	1	1
6	00110	1	1	1	1	1	1	1
8	01000	1	1	1	1	1	1	1
14	01110	1	1	1	1	1	1	1
0	00000	1	1	0	0	0	0	0
2	00010	1	0	0	0	0	0	0
10	01010	1	0	0	0	0	0	0
29	11101	1	0	0	0	0	0	0
12	01100	0	0	0	0	0	1	1
13	01101	0	0	1	1	1	0	0
28	11100	0	1	0	0	0	0	0

TABLE III
EXHAUSTIVE TEST SETS FOR $ex3$

i	s_i	2	3	4	5	6	7	8
1	0001	1	1	1	1	1	1	1
0	0000	1	1	0	0	0	0	0
2	0010	1	0	1	1	1	1	1
3	0011	1	1	0	0	0	0	0
4	0100	1	1	0	0	0	0	0
8	1000	1	0	1	1	0	0	0
13	1101	1	1	0	0	0	0	0
14	1110	1	0	0	0	0	0	0
15	1111	1	0	0	0	0	0	0
7	0111	0	0	1	1	1	1	1
10	1010	0	1	0	0	1	1	1

Test Compaction Procedure: This section describes an efficient test compaction procedure that accepts a compact two-cycle test set *T*init, and a limit *L*>2 on the number of functional clock cycles in a multicycle test. The procedure produces a compact multicycle test set *T*multi where the tests have at most *L* functional clock cycles.

TABLE IV
EXAMPLE OF UNMODIFIED TESTS

i	t_i	det
13	$< 001101101001011, 011010101, 4 >$	271
14	$< 001100010001111, 001111001, 4 >$	293
15	$< 000010111111001, 010101111, 4 >$	348
16	$< 000101000001010, 011011010, 4 >$	376
17	$< 011011011000101, 010010100, 4 >$	413
18	$< 001100110001101, 011001010, 4 >$	418
19	$< 010101111000010, 000101100, 4 >$	436
20	$< 110100100010111, 000110111, 4 >$	491
21	$< 000000000101111, 010101000, 4 >$	499

TABLE V
EXAMPLE OF A MODIFIED TEST

test	det
$t_5 =< 011010111001110, 000100010, 2 >$	$n_5 = 3$
$t_{5,1} =< 011010111001110, 000100010, 3 >$	$n_{5,1} = 3$
$t_{5,2} =< 110010110001111, 000100010, 3 >$	$n_{5,2} = 7$

Multi-cycle implementation of MIPS: Revisit the 1-cycle version shown in below figure

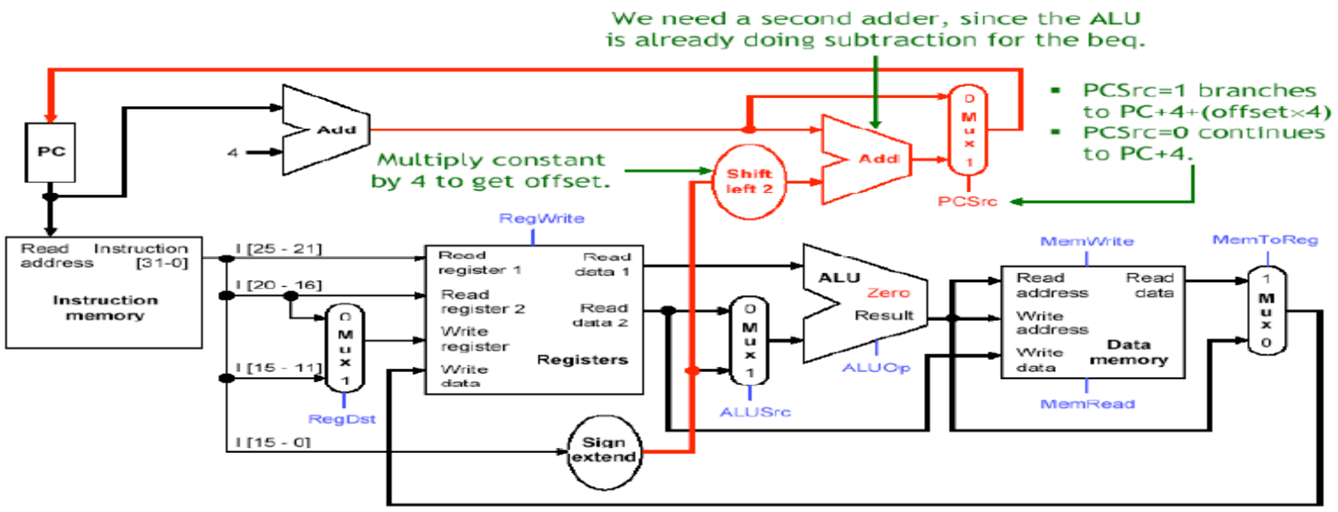


Figure 1: Multi-cycle implementation of MIPS

The multi-cycle version shown in below figure

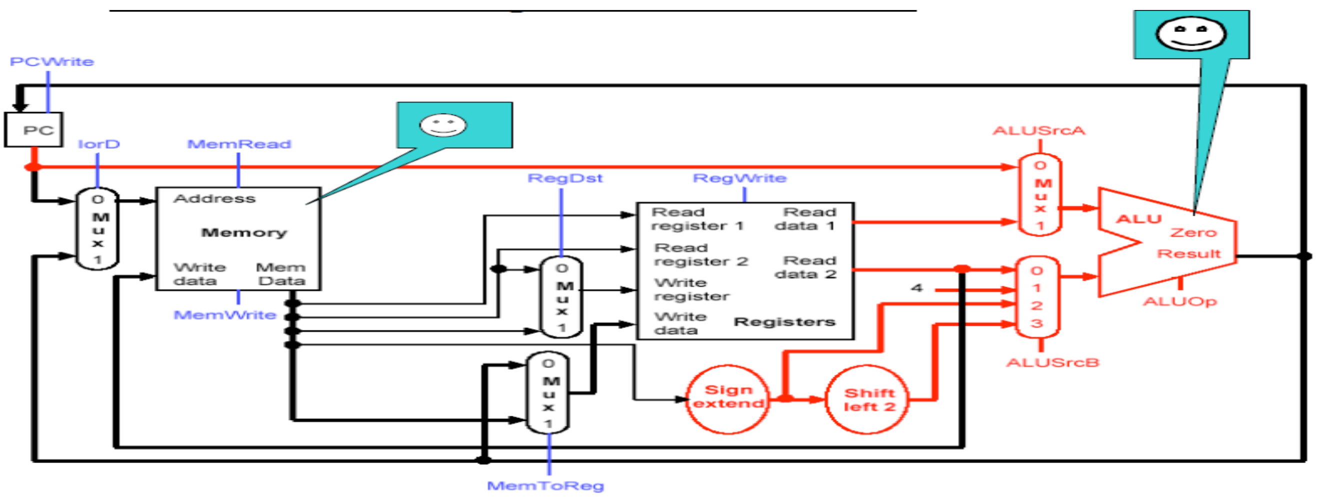


Figure 2: Multi-cycle version 1

Note that we have eliminated two adders, and used only one memory unit (so it is Princeton architecture) that contains both instructions and data. It is not essential to have a single memory unit, but it shows an alternative design of the data path. Intermediate registers are necessary in each cycle; a fraction of the instruction is Executed Five

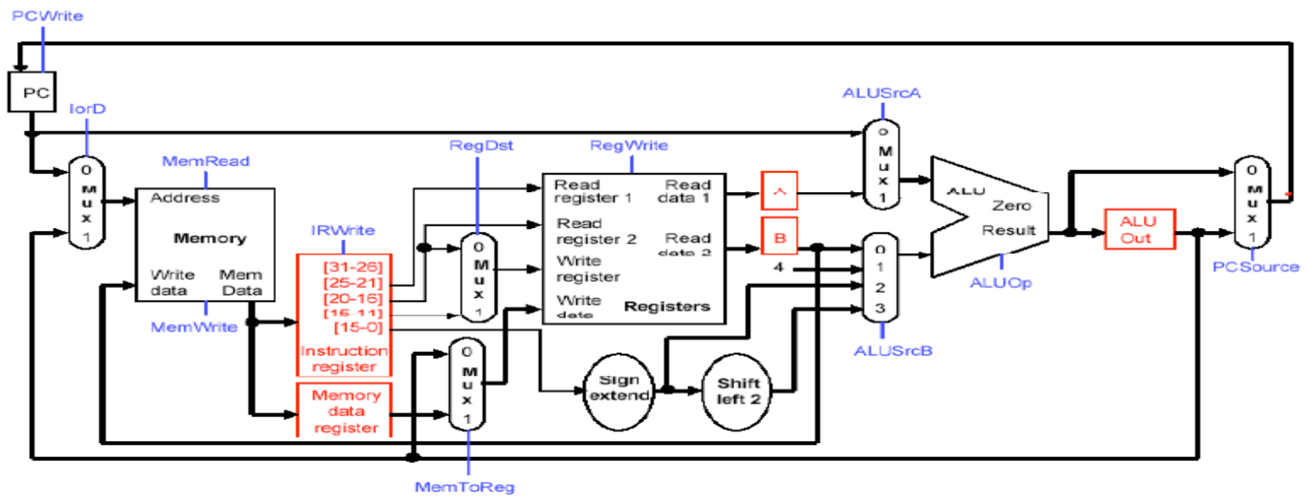
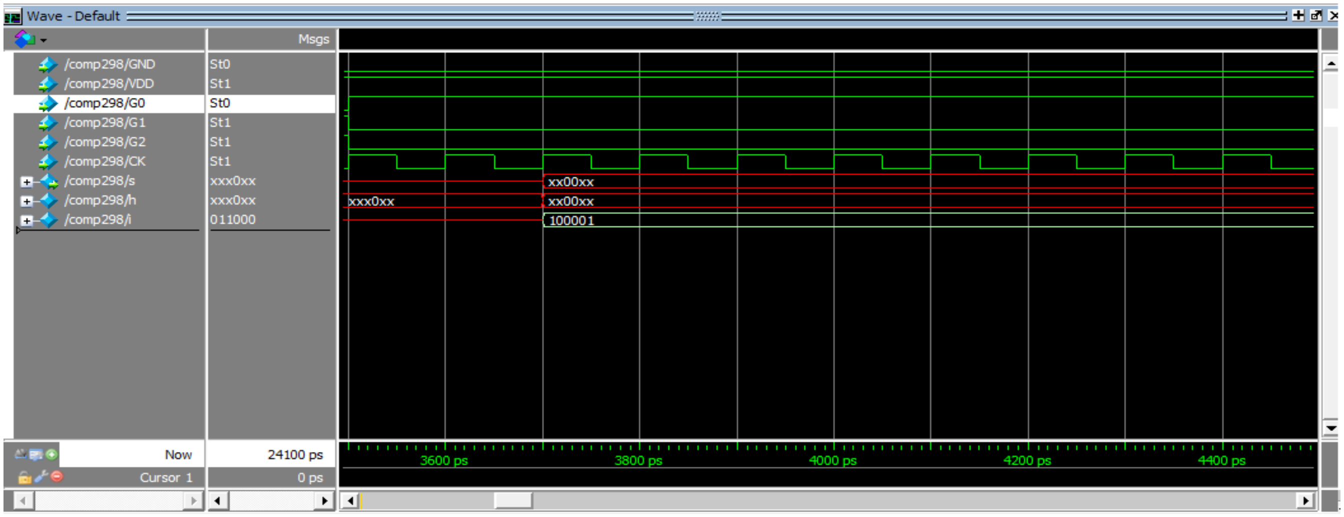


Figure 3: Multi-cycle version 2

Five stages of instruction execution Cycle 1 Instruction fetch and PC increment Cycle 2 Reading sources from the register file Cycle 3 Performing an ALU computation Cycle 4 Reading or writing (data) memory Cycle 5 Storing data back to the register file Why intermediate registers? Sometimes we need the output of a functional unit in a later clock cycle during the execution of an instruction. (Example: The instruction word fetched in stage 1 determines the destination of the register write in stage 5. The ALU result for an address computation in stage 3 is needed as the memory address for lw or sw in stage 4.) These outputs must be stored in intermediate registers for future use. Otherwise they will be lost by the next clock cycle. (Instruction read in stage 1 is saved in Instruction register. Register file outputs from stage 2 are saved in registers A and B. The ALU output will be stored in a register ALUout. Any data fetched from memory in stage 4 is kept in the Memory data register MDR.)
The Five Cycles of MIPS (Instruction Fetch) IR:= Memory[PC] PC:= PC+4 (Instruction decode and Register fetch) A:= Reg[IR[25:21]], B:=Reg[IR[20:16]] ALUout := PC + sign-extend(IR[15:0]] (Execute|Memory address|Branch completion) Memory reference: ALUout:= A+ IR[15:0] R-type (ALU): ALUout:= A op B Branch: if A=B then PC := ALUout (Memory access | R-type completion)
LW: MDR:= Memory[ALUout] SW: Memory[ALUout]:= B R-type: Reg[IR[15:11]]:= ALUout
(Writeback) LW: Reg[[20:16]]:= MDR

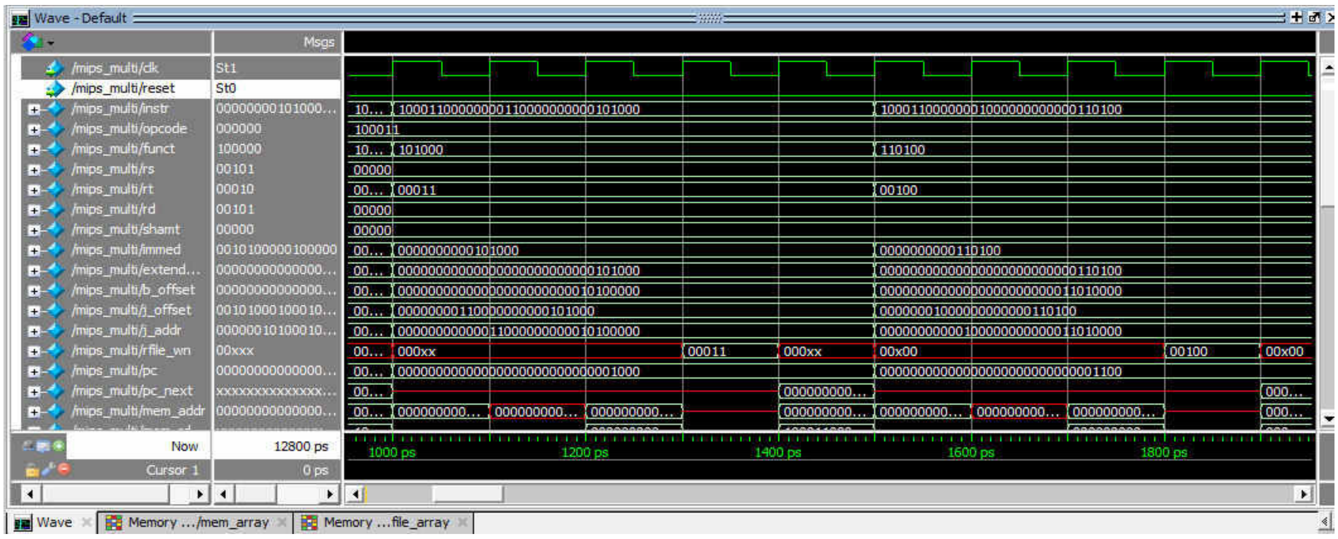
Experimental Results: The test compaction procedure was applied to benchmark circuits as described in this section. The implementation of the procedure does not use any commercial tools. The fault simulation process that it is based on considers one fault and one test at a time. No re-synthesis was applied to the benchmark circuits that may affect their sets of faults or the numbers of tests required for detecting them. The test compaction procedure was applied to compact two cycle broadside test sets with constant primary input vectors that were generated for transition faults in benchmark circuits. Only benchmark circuits where the transition fault coverage is at least 70% were considered. For other circuits, the requirement to use broadside tests with constant primary input vectors causes high fault coverage loss. The below figures 4,5 shows the simulation results for multicycle MIPS and comparison s298 bench mark circuit.



```
Wave - Default
                     Msgs
/comp298/GND         St0
/comp298/VDD         St1
/comp298/G0          St0
/comp298/G1          St1
/comp298/G2          St1
/comp298/CK          St1
/comp298/s           xxx0xx
/comp298/h           xxx0xx
/comp298/i           011000

Now                  24100 ps
Cursor 1             0 ps
```

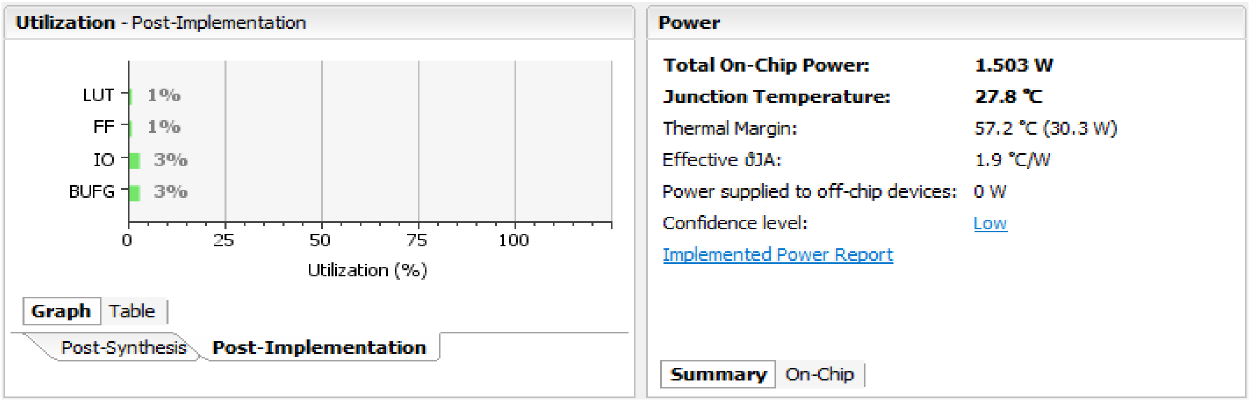
Figure 4: Simulation result of s298



```
Wave - Default
                          Msgs
/mips_multi/clk           St1
/mips_multi/reset         St0
/mips_multi/instr         000000000101000...
/mips_multi/opcode        000000
/mips_multi/funct         100000
/mips_multi/rs            00101
/mips_multi/rt            00010
/mips_multi/rd            00101
/mips_multi/shamt         00000
/mips_multi/immed         0010100000100000
/mips_multi/extend...     000000000000000...
/mips_multi/b_offset      000000000000000...
/mips_multi/j_offset      001010001000010...
/mips_multi/j_addr        000000101000010...
/mips_multi/rfile_wn      00xxx
/mips_multi/pc            000000000000000...
/mips_multi/pc_next       xxxxxxxxxxxxxxx...
/mips_multi/mem_addr      000000000000000...

Now                       12800 ps
Cursor 1                  0 ps
```

Figure 5: Simulation result for Multicycle MIPS



Utilization - Post-Implementation

Power

Total On-Chip Power:	**1.503 W**
Junction Temperature:	**27.8 ℃**
Thermal Margin:	57.2 ℃ (30.3 W)
Effective ϑJA:	1.9 ℃/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Implemented Power Report

Figure 6: Synthesis report for s298

CONCLUSION:

The first part of this paper studied the extent to which scan-in states of a compact two-cycle test set are suitable as scan-in states of a compact multicycle test set. This issue is important for avoiding sequential test generation when constructing multicycle tests. This study was performed by considering exhaustive two-cycle and multicycle test sets for finite-state machine benchmarks. Based on the results of this study, the second part of this paper described an efficient test compaction procedure that uses two-cycle tests in a given test set as a basis for the computation of multicycle tests. The procedure includes the option of modifying scan-in states, and the corresponding primary input vectors, in order to make them more suitable for multicycle tests. This step was applied selectively to tests whose numbers of functional clock cycles were lower than a target. Experimental results were presented to demonstrate the importance of this step to the ability to achieve test compaction using multicycle tests without performing sequential test generation.

REFERENCES:

1. S. Y. Lee and K. K. Saluja, “Test application time reduction for sequential circuits with scan,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 14, no. 9, pp. 1128–1140, Sep. 1995.
2. Pomeranz and S. M. Reddy, “Static test compaction for scan-based designs to reduce test application time,” in *Proc. Asian Test Symp.*, Singapore, 1998, pp. 198–203.
3. P. C. Maxwell, R. C. Aitken, K. R. Kollitz, and A. C. Brown, “IDDQ and AC Scan: The war against unmodelled defects,” in *Proc. Int. Test Conf.*, Washington, DC, USA, 1996, pp. 250–258.
4. J. Rearick, “Too much delay fault coverage is a bad thing,” in *Proc. Int. Test Conf.*, Baltimore, MD, USA, 2001, pp. 624–633.
5. X. Lin and R. Thompson, “Test generation for designs with multiple clocks,” in *Proc. Design Autom. Conf.*, Anaheim, CA, USA, 2003, pp. 662–667.
6. G. Bhargava, D. Meehl, and J. Sage, “Achieving serendipitous N-detect mark-offs in multi-capture-clock scan patterns,” in *Proc. Int. Test Conf.*, Santa Clara, CA, USA, 2007, pp. 1–7.
7. Park and E. J. McCluskey, “Launch-on-shift-capture transition tests,” in *Proc. Int. Test Conf.*, Santa Clara, CA, USA, 2008, pp. 1–9.
8. E. K. Moghaddam, J. Rajski, S. M. Reddy, and M. Kassab, “At-speed scan test with low switching activity,” in *Proc. VLSI Test Symp.*, Santa Cruz, CA, USA, 2010, pp. 177–182.
9. Pomeranz, “Generation of multi-cycle broadside tests,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 8, pp. 1253–1257, Aug. 2011.
10. Pomeranz, “Multi-cycle tests with constant primary input vectors for increased fault coverage,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 9, pp. 1428–1438, Sep. 2012.
11. Pomeranz, “Multi-cycle broadside tests with runs of constant primary input vectors,” *IET Comput. Digit. Tech.*, vol. 8, no. 2, Mar. 2014, pp. 90–96.
12. Pomeranz, “Design-for-testability for multi-cycle broadside tests by holding of state variables,” *ACM Trans. Design Autom. Electron. Syst.*, vol. 19, no. 2, pp. 1–20, Mar. 2014.
13. K. Lee, J. Chen, and C. Huang, “Using a single input to support multiple scan chains,” in *Proc. Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, 1998, pp. 74–78.
14. C. Barnhart *et al.*, “OPMISR: The foundation for compressed ATPG vectors,” in *Proc. Int. Test Conf.*, Baltimore, MD, USA, 2001, pp. 748–757.
15. J. Rajski *et al.*, “Embedded deterministic test for low cost manufacturing test,” in *Proc. Int. Test Conf.*, Baltimore, MD, USA, 2002, pp. 301–310.
16. N. A. Touba, “Survey of test vector compression techniques,” *IEEE Design Test Comput.*, vol. 23, no. 4, pp. 294–303, Apr. 2006.
17. Pomeranz and S. M. Reddy, “At-speed delay testing of synchronous sequential circuits,” in *Proc. Design Autom. Conf.*, Anaheim, CA, USA, 1992, pp. 177–181.
18. K.-T. Cheng, “Transition fault testing for sequential circuits,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 12, no. 12, pp. 1971–1983, Dec. 1993.
19. T. J. Snethen, “Simulation-oriented fault test generator,” in *Proc. Design Autom. Conf.*, New Orleans, LA, USA, 1977, pp. 88–93.
20. P. Girard, C. Landrault, V. Moreda, and S. Pravossoudovitch, “An optimized BIST test pattern generator for delay testing,” in *Proc. VLSI Test Symp.*, Monterey, CA, USA, 1997, pp. 94–100.
21. K.-H. Tsai, J. Rajski, and M. Marek-Sadowska, “Scan encoded test pattern generation for BIST,” in *Proc. Int. Test Conf.*, Washington, DC, USA, 1997, pp. 548–556.