TEST SCHEDULING OPTIMIZATION FOR EMBEDDED CORE TESTING

K. CHAKRAPANI [1], DR. P. NEELAMEGAM [2]
School of Computing SASTRA University
School of EEE SASTRA University

ABSTRACT

Test scheduling is crucially important for optimal SoC test automation in allocating the limited available test resources. To assign test resource, this paper introduces a fuzzy based engine. Test pipelining is used to minimize the time of testing the SoC. The power consumption during the test process should be under control, without exceeding the maximal power and damaging the whole systems. Concurrent behaviors are due to the adept of process algebra .Based on the procedure, the parallel test actions are mapped into concurrent process to outline the test scheduling scheme for SoC core concurrent test. Power Swarm optimization based optimum renders solution to the multiple constraints like test power dissipation, test resources and test priorities prevail in the algorithm for SoC test scheduling based on process algebra.

Keywords: - *System-on-Chip, Test Access Mechanism, Process Algebra, Fuzzy Logic.*

I. INTRODUCTION

Large-scale integration has added enormous complexity to the process of testing modern digital circuits. Besides, during the past several years, integrated circuit technology evolved from chip-set philosophy to embedded cores based system-on-a-chip (SoC) concept [1], which simply refers to an IC, designed by stitching together multiple stand-alone VLSI designs to provide full functionality for an application. Though many aspects of these embedded cores-based systems and SoC are still evolving, they are revolutionizing the electronics industry. These innovations are already on their way to the next generation of cell phones, multimedia devices, and PC graphics chipsets. The cores-based design, justified by the necessity to decrease time-to-market, has created a host of challenges for the design and test community [4][9]. The core test integration is a complex problem – the chip integrator can modify the test and add design for test (DFT) and built-in self-test (BIST) features, if necessary. Specifically, in the context of embedded cores-based system testing, electrical isolation involving the input and output ports of the core from the chip or other cores is a necessity. The fundamental items of interest in core test is access, control, and isolation, and these are the issues which were addressed by the IEEE Technical Council on Test Technology Working Group 1500[2] entrusted with the responsibility of developing standard architecture for their solution. The embedded core test requires, in general, hardware components like wrapper around the core, a source and a sink for test patterns (on-chip or off- chip) and an on-chip test access mechanism (TAM) to connect the wrapper to the source or sink. The cores could be without boundary scan or with boundary scan. For design and test reuse, ASIC manufacture has suggested certain characteristics. In general, different DFT and BIST schemes like scan, partial scan, logic BIST and scan-based BIST are used to test various logic blocks within a SoC like microprocessor or microcontroller. However, the main problem is still the resulting area overhead and performance penalties. Structural test methods like scan and BIST are desirable for test reuse, portability, and test integration into the SoC test set. The TAM includes on-chip test generation logic for cores with BIST. The DFT techniques involve adding optimized test logic within cores and at the chip level to enhance testability and DFT logic helps test pattern generation and application, and assist in the support test environment. In this paper, test methodologies are proposed for embedded cores-based system-on-a-chip (SoC) digital systems comprising of wrapper and TAM. The fault model used is the conventional single stuck-fault model. The nature of faults is single stuck faults. Thus each line can have only two types of stuck faults: stuck-at-1 and stuck-at-0. The IEEE 1500-compliant [2] wrapper separates the core under test from other cores. The TAM plays a vital role in transporting the test patterns to the desired core and the core responses to the output pin of the SoC. The TAM was implemented as a plain signal transport medium, which is shared by all the cores in the SoC. Once the compilation of the cores was done, the fault simulation was carried out with the test patterns feeding the cores through the TAM. The selection of the appropriate core is taken care of by the program running in the background. The simulation process is completely automatic, and requires no intervention from the designer during the test generation process. This paper describes the architecture of the wrapper and test access mechanism, together with models of the SoCs being used, based on application environment.

II. IEEE 1500 BASED SoC TEST INTEGRATION ARCHITECTURE

The IEEE 1500 wrapper [2][3] has various modes of operation. There are modes for functional (nontest) operation, inward facing (IF) test operation, and outward facing (OF) test operation. Different test modes also determine whether the serial test data mechanism (WSI–WSO) or the parallel test data mechanism (WPI–WPO), if present, is being utilized. Instructions loaded into the Wrapper Instruction Register(WIR), together with the IEEE 1500 wrapper signals, determine the mode of operation of the wrapper and possibly the core itself. A minimum set of instructions and corresponding operations shall be supplied. Optional instructions and their corresponding behavior are also defined, together with the requirements for extension of the instruction set. All instructions that establish test modes that utilize the parallel port WPI and WPO are optional, as the presence of this port is optional. Furthermore, IEEE 1500 also allows for user-defined instructions.

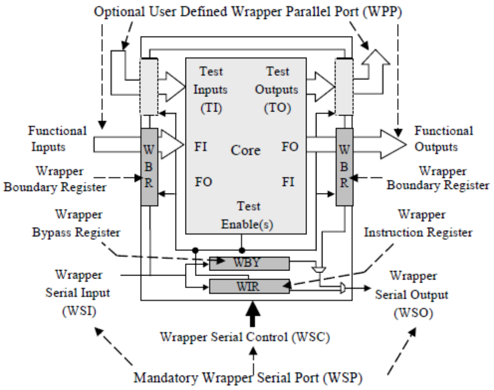
Optional User Defined Wrapper Parallel Port (WPP)
Test Inputs (TI)
Test Outputs (TO)
Functional Inputs
Functional Outputs
W B R
FI
Core
FO
FO
FI
W B R
Wrapper Boundary Register
Wrapper Boundary Register
Test Enable(s)
Wrapper Bypass Register
Wrapper Instruction Register
WBY
WIR
Wrapper Serial Input (WSI)
Wrapper Serial Output (WSO)
Wrapper Serial Control (WSC)
Mandatory Wrapper Serial Port (WSP)

Figure1: Standard IEEE 1500 wrapper

Components

IEEE 1500[2] has a set of instructions that are defined to use only the serial interface (WSP) and a corresponding set of instructions that are defined for the parallel interface. IEEE 1500 must allow accessibility to test the core. There is one main core test instruction—Wx_INTEST (user-specified core-test instruction)—that is flexible enough to allow any core test to execute. There are two other instructions that are mandatory: an instruction for functional mode (WS_BYPASS) and an instruction for external test mode (WS_EXTEST). WS_BYPASS puts the wrapper into the bypass configuration and allows access to all functional terminals of the core shown in Figure 1. WS_EXTEST is the serial EXTEST configuration of the wrapper. Even if there is a WP_EXTEST mode (for parallel access), there must still be a WS_EXTEST instruction capability. The signal connected to the WRCK terminal is a dedicated clock used to operate IEEE 1500 functions.

III. TEST ACCESS MECHANISM (TAM) AND WRAPPER

The design of test access mechanism (TAM)[4] and test wrapper is of critical importance in terms of system integration since they directly impact hardware overhead, test time, and tester data volume. The main issues in this context are wrapper optimization, core assignment to TAM wires, sizing of TAM, and TAM wire routing.

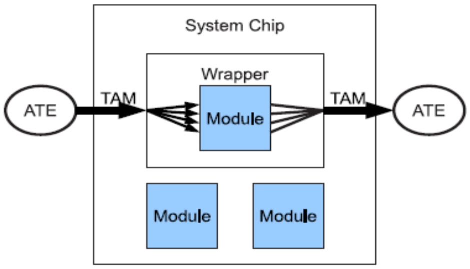

Figure 2: TAM Architecture

There are two important concepts related to TAM, viz. test pattern source and sink, and core wrapper. The test pattern source is responsible for generating the test vectors or test stimuli for the desired core under test. The test pattern sink compares the fault-free response to the faulty response of the core under test. The test pattern source and sink can be built on-chip or off-chip. In our case, we implemented this by using the fault simulator that generates the test vectors, and after getting the test response, compares it with the fault-free response. We will briefly discuss about fault simulation later. There are several ways to design and implement a TAM [5]. The common TAM architectures are: 1) daisy chained TAMs (that use serial shifting of test data); 2) bussed TAMs (based on use of complex protocols); 3) direct access TAMs (for cores with many inputs and outputs); and 4) multiplexer (MUX)-based direct access TAMs. In this paper, MUX-based direct access TAM architecture, as shown in Figure 2 is implemented. The TAM is used to drive the test vectors from the test source [5][6], that is, from the fault simulator to the desired core under test and to transport the test response from the core back to the fault simulator. The selection of the core in the SoC was implemented as part of the TAM architecture. The width of the TAM by the core that has the maximum number of input/output (I/O) pins within the SoC is determined. There are other issues for consideration at this phase, viz. the bandwidth of the TAM versus the cost of extra wires needed for its implementation, total test time depending on the TAM bandwidth, test vectors from the source, and ultimately test data for the individual core. The obvious mechanism to make embedded cores testable from the IC pins [10] is to make the core under test directly accessible from the IC inputs. Though this approach is mostly practiced for embedded memory cores, many block-based ASICs also use this test strategy.

IV. TAM CONTROLLER DESIGN

As shown in Figure 3, TAM controller is used to provide the dynamic control signals to wrapper [4], i.e., WIP. WIP is composed of six signals: WRCK, WRSTN, SelectWIR, ShiftWR, CaptureWR and UpdateWR. At the exception of WRSTN, all WIP signals are active high. The following are the different signals used.

- *WRCK* (Wrapper Clock): The dedicated test clock.
- *WRSTN* (Wrapper Reset): The dedicated asynchronous reset signal which resets wrapper instruction register (WIR) and puts the wrapper in normal operation mode.
- *SelectWIR:* Selects whether WIR or one of wrapper data registers (WDR) is exclusively connected between WSI and WSO.
- *ShiftWR*, *CaptureWR*: When the corresponding signal is asserted, a shift or capture operation will occur on the rising edge of WRCK.
- *UpdateWR*: When it is asserted, an update operation will occur on the falling edge of WRCK.
- *TransferDR*: TransferDR is required when the WBR includes cells with a transfer capability.

TAM controller can be used, as long as its outputs fully conform to the requirement of WIP. TAM controller is a Finite State Machine (FSM) in nature. Since WIP signals mimic the output of Test Access Port (TAP) controller of IEEE 1149.1[9], here the same state-diagram as TAP (Figure. 3) is used and the output logic of FSM is modified in order to make it suitable for WIP.

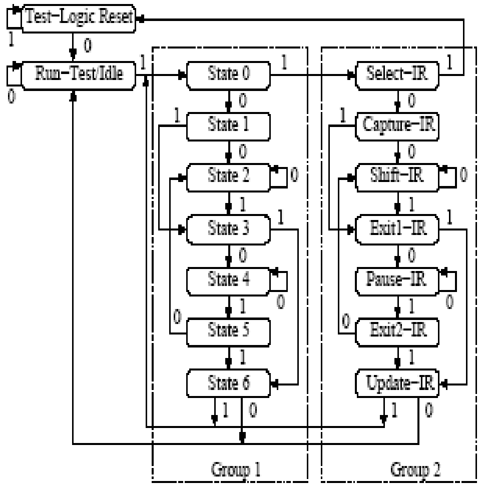

Figure 3: TAM Controller State Diagram

The three inputs (tclk, tms, trst) of TAM controller have just the same meaning as in TAP, that is, TAM controller changes its state on the rising edge of tclk according to tms. In our design, tclk is also used as WRCK.

V. WRAPPER DESIGN

P1500 wrapper consists of WIR, wrapper bypass register (WBY), wrapper boundary register (WBR), gating control logic, mandatory serial path, and optional parallel paths. The serial path is used both for wrapper control by loading instructions into WIR, as well as for low-bandwidth test data access to WBR. Parallel paths are used for internal scan-chains and high-bandwidth test data access to WBR. A minimal library of WBR cells is presented and the design of WBR is not depicted in this paper. WBY used in our wrapper is just an ordinary flip-flop.

Bypass registers both for the serial path and parallel paths are inserted.

WIR is composed of shift register, update register and decoding logic (Fig. 1). At the start of every test cycle, a new instruction is shifted into WIR through the serial path. Consequently, the operation of the wrapper is controlled by both WIP signals, as well as the presently active instruction in WIR. WIR outputs two static control signals for the wrapper. Sel identifies whether WBR or WBY is connected between WSI and WSO and M2_Mode determines whether the test data in WBR apply to system logic or not. WRSTN resets the content in the update register and puts the wrapper in normal operation mode.

VI. POWER SWARM OPTIMIZATION BASED OPTIMUM SEARCH

The PSO model is a new population based optimization strategy introduced by J. Kennedy et al. in 1995. It has already shown to be comparable in performance with traditional optimization algorithms such as simulated annealing and the genetic algorithm. However, PSO does exhibits some disadvantages: it sometimes is easy to be trapped in local optima, and the convergence rate decreased considerably in the later period of evolution; when reaching a near optimal solution, the algorithm stops optimizing, and thus the accuracy that the algorithm can achieve is limited. Theoretical results have shown that the particle positions in standard PSO oscillate in damped sinusoidal waves until they converge to points in between their previous best positions and the global best positions discovered by all particles so far. If some point visited by a particle during this oscillation has better fitness than its previous best position, then particle movement continues, generally converging to the global best position discovered so far. All particles follow the same behavior, quickly converging to a good local optimum of the problem. However, if the global optimum for the problem does not lie on a path between original particle positions and such a local optimum, then this convergence behavior prevents effective search for the global optimum. It may be argued that many of the particles are wasting computational effort in seeking to move in the same direction towards the local optimum already discovered, whereas better results may be obtained if various particles explore other possible search directions.

Each particle *Pi* consists of a position vector *xi* which represents the candidate solution of the optimization problem, a velocity vector *vi* and a memory vector *pibest* of the best candidate solution encountered by the particle. Each particle flies in the dimensional problem space with a velocity which is dynamically adjusted according to the flying experiences of its own and its colleagues

. The position of a particle is updated by

$$x_i\,(t_1)=x_i\,(t)\; v_i\,(t_1) \qquad \text{-------(1)}$$

and its velocity according to

$$v_i\,(t+1) = w\, v_i\,(t) + c_1 r_1\,(\,p_{ibest}\,(t) - x_i(t)\,) + c_2 r_2\,(\,p_{gbest}\,(t) - x_i(t)\,) \qquad \text{-------(2)}$$

Algorithm ***Power Swarm optimization***

```
Create and initialize a n-dimensional swarm S
and set t = 0 ;
repeat
    for each particle i=1,…,s
        if fi <fibest
            S.pibest = S.xi ; fibest = fi ;
        else if fi >fiworst
            S.piworst = S.xi ; fiworst = fi ;
        endif
        if fibest < fgbest
            S.pgbest = S.pibest ; fgbest = fibest ;
        Endif
    Endfor
    for each particle i=1,…,s
        compute its current activity activity(Pi(t)) and
        inertia weight wi(t);
        if activity(Pi(t))<a(t)

            update its velocity vi(t) using Eq.(6.1);
        else
            update its velocity using Eq.(6.2);
        endif
        update its current position xi(t) using Eq.(2);
    endfor
    t++;
until stopping condition is true;
return S.pibest ;
```

The scheduling algorithm selects wrapper design for each core, assigns a start time, an end time and which TAM wires to use for each core in such a way that the test application time is minimized while all constraints are satisfied[15]. The *Optimal Time* is a lower bound that represents the “ideal” situation, but, due to the TAM structure and the wrapper design, this limit is almost never reached. The *Optimal Time* is calculated using the formula:

$$\mathbf{OptimalTime} = \left\lceil \frac{\sum i\ bestPareto}{Wmax} \right\rceil$$

where W_{max} is the number of available pins for test access (the TAM bandwidth)[9].

The *Optimal Time* gives the lower bound of the total test time of the system when no constraints but TAM width limitations are considered. In the ideal case, the schedule does not contain any idle times (i.e. there is no cost loss between tests in the test schedule), and it is therefore the best test application time that can ever be achieved.

VII. EXPERIMENTAL RESULTS AND DISCUSSIONS

We have applied our wrapper design, TAM design and test scheduling algorithms to the ITC'02 benchmarks. The results for system d695 are given in Table 2, respectively. For each TAM width, the first group of columns gives the results of the multiplexed approach, in terms of the test time indicating the quality of the test schedule, and the CPU time used to generate the solution. The last group of columns gives the corresponding results of our efficient approach. The Pareto-optimal points are chosen as close as possible to the TAM width limit. The experimental results show clearly that our approach outperforms the multiplexed approach in terms of test scheduling lengths (on average, the test time of our approach is 36% smaller than that of the multiplexed approach). When compared with the other approach, our algorithm consumes only a tiny fraction of CPU time needed, while producing relatively comparable test scheduling results.

D695	**Multiplexed Approach**		**Our Approach**	
TAM Width	Test Time	CPU Time	Test Time	CPU Time
80	1987	43	1241	32
64	2456	38	1835	29
48	2739	32	2268	24
32	3234	25	2630	17
24	3956	21	3327	14
20	4156	14	3924	9
16	5467	8	4268	4

VIII. CONCLUSION

The Recent technology development has made it possible to design and manufacture extremely complex systems. In this paper we have proposed a particle swarm optimization based test scheduling technique that minimizes the test application time by allowing tests to be applied as concurrently as possible. The technique takes test power consumption and test conflicts into account when minimizing the test application time. The test conflicts we consider are due to unit testing with multiple test sets, hierarchical SOCs where cores are embedded in cores, and the sharing of test access mechanism (TAM) wires

IX. REFERENCE

1. Larson, E., K. Arvidsson, H. Fujiwara and Z. Peng, 2004. *"Efficient test solutions for core based designs."* IEEE Trans. Comput. Aid. Des. Integrat. Circ. Syst., 23: 758-774. DOI: 10.1109/TCAD.2004.826560

2. Ravi, S., L. Ganesh and N.K. Jha, 2001 "
Testing of core-based systems-on-a-chip". IEEE Trans. Comput. Aid. Des. Integrat. Circ. Syst., 20: 426-439. Digital Object Identifier 10.1109/43.913760

3. Sehgal, A., V. Iyengar and K. Chakrabarty, 2004*" SoC test planning using virtual test access architectures"* IEEE Trans. Very Large Scale Integrat. Syst., 12: 1263-1275. DOI: 10.1109/TVLSI.2004.834228

4. Shao, J., G. Ma, Z. Yang and R. Zhang, 2008 *" Process algebra based soc test scheduling for test time*
5. *Minimization"* Proceeding of the IEEE Computer Society Annual Symposium on VLSI, Apr. 7-9, IEEE Xplore Press, Montpellier, pp: 134-138. DOI: 10.1109/ISVLSI.2008.88

6. Yoneda, T., M. Imanishi and H. Fujiwara, 2007 *" An SoC test scheduling algorithm using reconfigurable union wrappers"* Proceeding of the Design, Automation Test Europe Conference and Exhibition, Apr. 16-20, IEEE Xplore Press, Nice, pp: 1-6. DOI: 10.1109/DATE.2007.364596

7. Shaer, B., D. Landis, and S. Al-Arian, *“Partitioning algorithm to enhance pseudoexhaustive testing of digital VLSI circuits,"* IEEE TVLSI, Volume: 8 Issue 6, Dec. 2000 pp. 750 -754.

8. Brglez, F., and H. Tujiwara, *“A Neutral Netlist of 10 Combinational Benchmark Circuits and A Target*

9. *Translator in Fortran,”* IEEE Int’l Symp. On Circuits and Systems, 1985.

10. Al-Arian, S. and R. Bolling, *“Improving the Testability of VLSI Circuits through Partitioning”* IEEE Sym. on Circuits and Systems, Vol. 4, 1994, pp. 199-202.

11. Dr. S., Venayagamoorthy, G.K., Gudise, V.G., 2004 *“Optimal PSO for collective robotic search applications”* IEEE Congress on Evolutionary Computation, June 20–23, pp. 1390–1395.

12. Gudise, V.G., Venayagamoorthy, G.K., 2004 *“ FPGA placement and routing using particle swarm optimization”* IEEE Computer Society Annual Symposium on VLSI, February 19–20, pp. 307–308.

13. Hu Xiaohui, A., Eberhart, R., 2002 *“ Multi objective optimization using dynamic neighborhood particle swarm optimization”* In: Proceedings of the 2002 Congress on Evolutionary Computation, vol. 2, May 12–17, pp. 1677–1681.

14. Kennedy, J., Eberhart, R., 1995 *“Particle swarm optimization”* In: Proceedings IEEE International Conference on Neural Networks, vol. IV, Perth, Australia, pp. 1942–1948. ISBN 1558605959.

15. Shaer, B., Al-Arian, S., Landis, D., 2000b *“ Partitioning sequential circuits for pseudoxhaustive testing”.* IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 8 (5), 534–541.

16. Shi, Y., Eberhart, R.C., 1998 *“A modified particle swarm optimizer”* In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 69–73.

AUTHOR PROFILE

1. Prof *K. Chakrapani* obtained his B.E in Electronics and Communication from Regional Engineering College, Tiruchirapalli, M.Tech in Computer Science from SASTRA University, Thanjavur and MS in Information Technology from Bharathidasan University, Tiruchirapalli. Currently he is working as an Assistant Professor at TIFAC CORE, SASTRA University, Thanjavur. He is a life member of ISTE. He can be contacted at *kcp@core.sastra.edu* and his hand phone number is +919362637203

2. *Dr. P.Neelamegam* is currently working as a professor at SASTRA University, Thanjavur. He can be contacted at *neelkeer@eie.sastra.edu* and his hand phone number is +919443077327.