Design Methodology for 2.4GHz Dual-Core Microprocessor

Noriyuki Ito, Hiroaki Komatsu, Akira Kanuma, Akihiro Yoshitake,
Yoshiyasu Tanamura, Hiroyuki Sugiyama, Ryoichi Yamashita,
Ken-ichi Nabeya, Hironobu Yoshino, Hitoshi Yamanaka, Masahiro Yanagida,
Yoshitomo Ozeki, Kinya Ishizaka, Takeshi Kono, Yutaka Isoda
Fujitsu Limited
4-1-1 Kamikodanaka, Nakahara-ku,
Kawasaki, 211-8588, Japan
ito.noriyuki@jp.fujitsu.com

ABSTRACT

This paper presents a design methodology that was applied to the design of a 2.4GHz dual-core SPARC64^{TM} microprocessor with 90nm CMOS technology. It focuses on the newly adopted techniques, such as efficient data management in dual-core design, fast delay calculation of the noise-immune clock distribution circuit, enhanced signal integrity analysis of a large-scale custom macro design, and enhanced diagnosis capability using a logic BIST circuit.

Keywords: Microprocessor, dual-core, clock, custom macro, signal integrity, test, BIST

1. Introduction

As a successor of a 2.16GHz single-core microprocessor [1] with 90nm CMOS technology, a dual-core SPARC64 microprocessor [2] achieves 2.4GHz using the enhanced 90nm CMOS technology. The CAD system [3], which has supported the high performance microprocessor design, is enhanced especially in data management, timing analysis, custom macro design, and test. In data management, design of identical dual cores is made efficient. In timing analysis, delay calculation of a clock distribution circuit with split and shielded wires to reduce noise is sped up. In custom macro design, signal integrity analysis related to coupling noise and Tr/Tf analysis is enhanced. In test, not only good/no-good can be determined separately in each core, but also capability of fault diagnosis is available even if a logic BIST circuit is used. By these enhancements, we completed the 2.4GHz dual-core microprocessor in a short time of about one year.

The paper is organized as follows. Section 2 presents an overview of the microprocessor to which our design methodology is applied. Section 3 presents our concepts when we constructed our CAD system. In Sections 4 through 7, we describe the newly adopted techniques, namely physical hierarchy for dual-core design, clock design, custom macro design, and testing. Finally, we conclude the paper in Section 8, with directions for future work.

2. Overview of dual-core microprocessor

The specification of the 2.4GHz SPARC64 microprocessor is as follows:

Process: 90nm, Cu metallization, 10 metal layers
Frequency: 2.4GHz
Die size: 20.38mm x 20.67mm
Transistor count: 540M
Level 2 on-chip cache: 6MB
I/O signals count: 412
Power dissipation: less than 120W

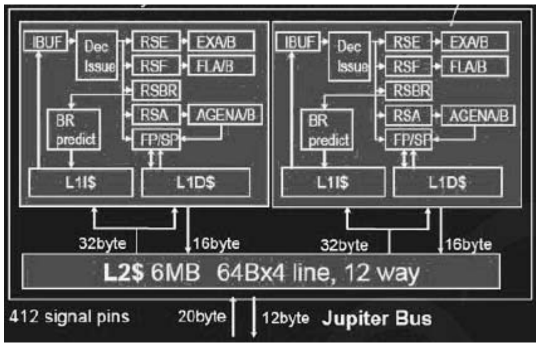

Figure 1. Dual-core microprocessor [2]

3. CAD system

In the dual-core microprocessor design, two kinds of CAD systems are used: one for chip design based on standard cells and macros, and the other for custom macro design based on transistors. For logic design and verification, EDA vendor tools as well as in-house tools are used. On the other hand, for timing analysis and chip layout, in-house tools are used. In Table 1, we show design steps for which tools from EDA vendors are used. Since EDA vendor tools related to logic design are now mature, we use logic simulator and emulator from EDA vendors as well as an in-house logic simulator. Gradually EDA vendor tools are replacing our in-house logic simulator. As for transistor-level noise analysis, we think that it should be developed as an in-house tool to ensure correct margins. However, we could not afford to develop it due to limited manpower and were forced to use an EDA vendor tool. In Table 2, we show design steps for which in-house tools are used. A high performance microprocessor uses latest cutting-edge circuit and CMOS technologies, which necessitate enhancements in CAD tools such as design rule checkers, placement, routing, and timing/noise analysis. These enhancements are usually not available in vendor tools at the time. So in-house tools are developed to support the state-of-the-art technology.

One extremely important issue is that of the continuity of each CAD tool with technology and over time. In each in-house CAD tool, numerous user requirements and know-how are incorporated. Each of them constructs the originality of our own design methodology. An EDA vendor tool that has been used for a long time may be replaced suddenly with a new tool. If this happens, all of user requirements and know-how are not necessarily carried over to the new CAD tool. Therefore, it is important to set stable in-house CAD tools in the core of design methodology. Indeed our CAD system uses a suite of stable in-house CAD tools, which inherit relevant user requirements and know-how from the era when mainframe computers were developed.

Table 1. Design steps for which EDA vendor tools are used

	Typical cases	Why?
1	Logic simulator, emulator	Tools are mature
2	Editors for cell/macro design, circuit simulator	No competitive advantage with in-house development
3	Noise analysis based on transistors	In-house development is not in time
4	DRC, LVS	Specified as a sign-off tool

Table 2. Design steps for which in-house tools are used

	Typical cases	Why?
1	Logical and physical design rule checkers	Support our original design rules
2	Layout, timing analysis	Tools influence design methodology
3	Routing	Need extensive tuning for the state-of-the-art CMOS process
4	Placement, routing	Essential for high performance
5	Noise analysis based on standard cells/macros	Need to ensure correct margins
6	Clock design, Power grid design	Capability of EDA vendor tools are insufficient

4. Physical hierarchy for dual-core design

In our conventional design methodology for a single-core microprocessor, a chip is divided into sub-chips, each of which is further sub-divided into blocks. One sub-chip corresponds to a unit such as execution unit, instruction unit, or storage unit. In the dual-core microprocessor design, there are three different approaches to represent a core in timing and physical views. The first approach is to design the core as a large macro such as IP or RAM. The second approach is to introduce a new physical hierarchy between a chip and a sub-chip. Finally, the third approach is to place the sub-chips of the core directly at the chip-level. The difference between the first and second approaches is that in the first the core is treated as a macro till the final stage, whereas in the second the new hierarchy is expanded as and when necessary (such as when some analysis is performed on the entire chip). Since logically and physically a core is designed separately from others in the first approach, it is easily handled during chip design. However, it is difficult to ensure the accuracy of analysis at the core boundary when the core is treated as a macro. Although there is a merit of easy handling in both first and second approaches, some modifications to the CAD system are needed to handle the new physical hierarchy. Designers also have to run additional CAD tools in the new hierarchy of the core and have to manage its data. In the third approach, conventional hierarchical structures such as chip, sub-chip, and block are not changed. Therefore, no modification to the conventional CAD system is needed and very little additional work is imposed on designers. We selected the third approach, since the CAD and design overhead is the least as compared to others. We implemented a function that groups sub-chips in the core and places them at another location on the chip as a group with arbitrary flip and rotation. Here, the designer needs to manage only one core data. When timing or noise analysis is performed on the chip, all boundaries of sub-chips and groups are removed and flattened to cells and macros within the CAD tool. In this process, all layout data of the original core is copied to another instantiation of the core according to its flip and rotation. Global wires that pass over cores are designed and routed at the chip-level so that they use exactly the same channel in each core. Since logic outside the cores is not always symmetric, connections from the two cores to other sub-chips are not always the same. Therefore, the cores are expanded into cells and macros during chip analysis to accurately analyze a path from latches outside cores to latches in the cores.

5. Clock design

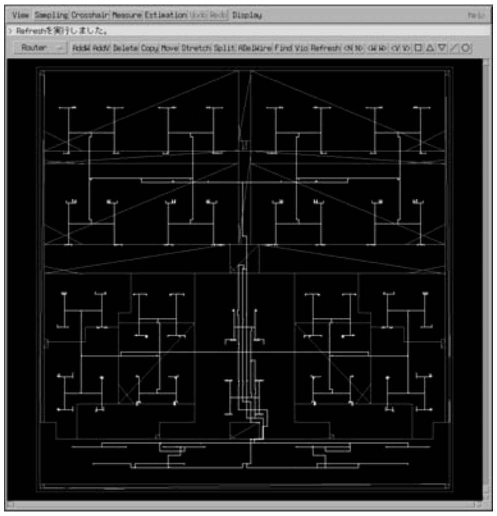

Figure 2. H-shaped clock routing

To reduce clock skew, mesh-shaped layout may be better than H-shaped layout. However, mesh-shaped layout consumes more routing channels and power. Further, delay calculation of a clock net with mesh-shaped layout takes more time. Therefore, we adopt a tree structure with H-shaped layout in the clock distribution circuit, as shown in Figure 2. An H-tree is a regular structure, with symmetric routing patterns, and is easy to construct and route. For all clock nets except connections to latches, split wires with shields in between are used to reduce inductance noise. A wide clock wire is split into several wires, which are shielded by GND wires not only in between, but also at two layers below. Clock nets connected to latches are routed by an automatic router. Other clock nets are routed manually with an in-house interactive P&R editor. When a clock net that is split with shields is routed from one layer to another, routed wires become very complicated due to generated vias, as shown in Figure 3.

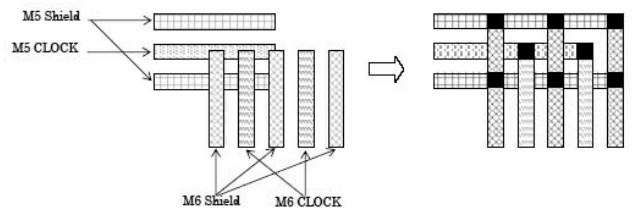

Figure 3. Changing wiring layers in a clock net

Due to this complexity, modification of a clock net takes time if it consists of split and shielded wires. Also, split and shielded wires may have loops due to vias that are generated when the routing layer is changed. These loops make it impossible to calculate delay using simple models such as Elmore delay [4]. This means that the delay of the clock distribution circuit must be calculated by a circuit simulator such as SPICE. However, then the turn-around-time for any clock net modifications becomes large. To solve this problem, clock nets are routed manually as one wire instead of split and shielded wires. Considering the complete design flow, it is only in the final design phase that the designers improve delay by pico-seconds in critical paths. So a SPICE-accurate delay calculation is necessary only in the final phase. In this final phase, each clock net represented by single wires is converted by a CAD tool into split and shielded wires according to the clock design rules. Thus, a clock net is treated as one wire till the final design phase, and its delay is calculated by the conventional Elmore delay model. This achieves the short turn-around-time when the clock distribution circuit is modified. It is also important to minimize the error in the delay calculated when the clock net is treated as one wire, as compared to SPICE simulation results for a split wire. For this purpose, a new dedicated RC table is prepared for extraction in clock nets routed as one wire. During the SPICE simulation of the clock distribution circuit at the final design phase, the clock circuit is divided into three groups from its root to leaves (as explained later). Table 3 shows the results and the errors of three paths for the two calculation methods: 1) each clock wire is treated as one wire and its delay is calculated by Elmore delay model, 2) RC parasitics are extracted after one wire is split and shielded, then its delay is calculated by SPICE.

We first calculate delay based on a Steiner tree constructed for each clock net. The maximum error of delays calculated by Elmore delay model is 10.4% when compared to the results of SPICE simulation. When clock nets are actually routed as one wire, the error becomes less than 3%. When the clock wire is treated as one wire and its delay is calculated by Elmore delay model, no inductance is considered. When the delay is calculated by SPICE, however, we include inductance in the model. Since inductance is not considered in calculation by Elmore delay model, delay is calculated 2-4% larger compared to SPICE simulation with inductance. To speed up SPICE simulation, we divide the clock distribution circuit from the root to leaves into three groups. SPICE simulation begins from the first group and proceeds to the third group. In order to accelerate the simulation, the second group is further divided into 7 sub-groups based on the sub-trees in the second group. Similarly, the third group is divided into 450 sub-groups. This is done for both model size reduction and higher parallelism. SPICE simulation takes about 5 hours on a 1.3GHz UNIX server with 5 jobs running in parallel. We estimate power dissipation of clock distribution circuit to be about 11% (excluding power consumed by clock circuits within latches and custom macros) of the chip total power dissipation.

Table 3. Errors in delay calculation

	By SPICE for split	By Elmore for one wire			
	Routing	Steiner	error	Routing	error
Path1	246.5ps	272.1ps	10.4%	255.1ps	3.5%
Path2	144.2ps	148.8ps	3.2%	138.9ps	-3.7%
Path3	181.6ps	183.0ps	0.8%	176.7ps	-2.7%

6. Custom macro design

Custom macro design is a key in implementing our 2.4GHz dual-core microprocessor, since only it can yield the required high performance and density for a large fraction of the microprocessor. In this section, we describe the CAD flow and techniques especially for timing and Tr/Tf analysis in the custom macro design.

6.1 CAD flow

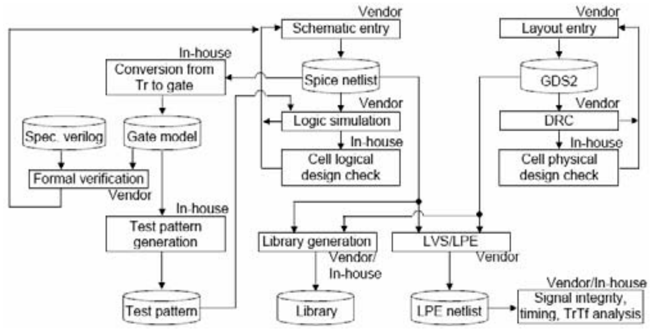

Figure 4. CAD flow for custom macro design

Figure 4 shows the CAD flow of our custom macro design. After schematic entry, logic simulation and logical DRC are performed. Then, the transistor-level netlist is converted to a gate-level netlist G by a CAD tool. This gate-level netlist G is compared to the reference gate-level design prepared by designers. Test vectors are generated by an ATPG tool and are applied on G to check whether undetected faults remain or not. After modifying logic so that all faults are detected, the final schematic design is obtained. This is then laid out by a polygon editor and checked by DRC and physical design rule check tools. The physical design rule check tools check for rules such as routability from each terminal, maximum parallel length of two wire segments (for crosstalk noise), etc. After both schematic and layout are completed, a netlist is extracted from the layout by performing LVS (Layout versus Schematic) and LPE (Layout Parasitic Extraction). Signal integrity, timing, and Tr/Tf analyses are performed on this extracted netlist. Each box of Figure 4 is annotated with the information whether that step is performed by an EDA vendor tool or by an in-house tool. In-house tools mainly cover logical/physical design rule checks, conversion of a netlist from transistors to gates, static timing analysis, and Tr/Tf analysis.

If a custom macro is large (i.e., more than 100K transistors), leaf cells are first developed and characterized as a library in the custom macro design CAD environment. Then, the custom macro is laid out in the chip design CAD environment using these leaf cells. Timing and other analyses are also done in the chip design CAD environment.

6.2 Signal Integrity

In the custom macro design, noise analysis is carried out by an EDA vendor tool. However, the in-house physical design rule checker checks for additional violations. During chip-level routing, routing channel between the wires of an already-placed custom macro can be used. To check a possible noise error due to two parallel wires, one of which is from the chip and the other from the custom macro, we do the following. After layout of the custom macro, chip-level wires are assumed to be virtually present in the unused routing channels in the custom macro. Then, based on the length of parallel wires, it is checked whether the noise exceeds the limit assuming that the virtual wire from the chip is an aggressor and the wire from the custom macro is a victim. This check is done within the custom macro. Drivability of the wire from the chip is assumed to be N times stronger than a normal driver. If the noise exceeds the limit, the unused channel is prohibited to have a chip-level wire. For the case that the chip-level wire is a victim and that the wire in the custom macro is an aggressor, noise is checked during chip-level analysis. Even when the noise does not exceed the limit, parallel wires can delay the signal transition due to coupling capacitance. A procedure similar to the aforementioned procedure is used in this case as well.

6.3 Timing analysis

When the number of transistors in a custom macro exceeds a few hundreds, it becomes impractical to simulate the whole macro with SPICE. The reason is that a large number of simulation vectors are needed to cover the critical paths, and circuit simulation for each vector takes time. To solve this problem, a custom macro is divided into sub-circuits. Each sub-circuit is characterized as a library cell. Then, the custom macro timing analysis problem can be solved by using a conventional gate-level static timing analysis tool [5]. A sub-circuit is a channel-connected component (CCC). A CCC is the set of all transistors and nodes reachable from the drain or the source of a transistor and surrounded by power and ground lines, gate inputs of transistors and inputs/outputs of the custom macro cell. The timing model of each CCC is determined using SPICE simulation.

To further speed up the timing analysis, we constrain our custom macro to use only a limited number of pre-characterized latches. This is because latch characterization is CPU-intensive. So during transistor-level timing analysis, before dividing the macro into CCCs, we perform library matching to detect latches in the macro. Another technique is used to improve the accuracy and ease of chip-level path timing analysis. The structure of timing library is different for a custom macro without latches and that with latches. The timing library for a custom macro without latches is generated in the same format as regular standard cells. For a custom macro consisting of CCCs/latches, delays calculated for CCCs are saved in the same format as a sub-chip. Since the latches of the custom macro are visible at the chip-level, chip-level timing analysis can analyze a path from or to a latch in the macro. During the analysis, CCCs of the custom macro are treated as regular library cells.

6.4 Tr/Tf analysis

Tr/Tf analysis is used to calculate Tr (rise time) and Tf (fall time) of a signal transition, and to check whether both Tr and Tf are within the limit. The reason why this check is needed is that large Tr/Tf values can cause excessive delay and power dissipation, and can also deteriorate reliability due to hot carrier, etc [6]. Furthermore, errors in the timing values computed by static timing analysis increase if Tr/Tf are outside the limits. In static timing analysis, the cell delay is calculated by interpolating graphs in a timing library using fan-out load, Tr/Tf, etc. If these values exceed the assumed range, the error increases. For these reasons, it is necessary to check Tr/Tf values at each node of CCCs.

Tr/Tf analysis begins with dividing a custom macro into CCCs, as explained in Section 6.3. The CCCs are traversed in a topological order, starting from the macro inputs and proceeding towards the outputs. At a CCC, output transitions are calculated using the values of input transitions generated from fanin CCCs. The calculated transitions are then propagated to fanout CCCs. To calculate output transition for a given input transition in a CCC, all transistors in the netlist are modeled as a table. In this table, the current *Ids* between a drain and a source depends on *Vs*, *Vd*, and *Vg*, which are source, drain, and gate voltages respectively. *Cs*, *Cd*, and *Cg*, which are source, drain, and gate capacitances respectively, are modeled independently of their voltages. Figure 5 shows how an inverter is modeled.

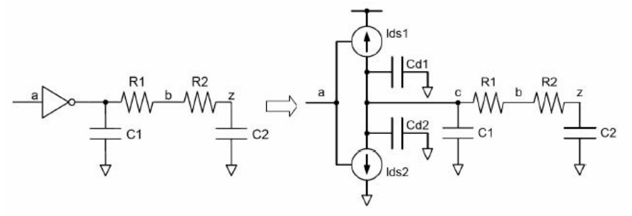

Figure 5. Representation of Tr. by table model

By applying Kirchhoff's laws, a CCC circuit consisting of *Rs*, *Cs*, and transistors is represented by expression (1):

$$C \cdot \dot{v}1(t) + G \cdot v1(t) + Ids(v0(t), v1(t)) = 0 \qquad (1)$$

Here, *v0(t)* is voltage vector of nodes whose voltages are known. Examples of such nodes are *Vdd*, *Vss*, and input nodes of the CCC. *v1(t)* is voltage vector of nodes whose voltages are unknown. These are internal nodes in the CCC. *Ids(vo(t), v1(t))* represents current between the drain and source of a transistor. *C* and *G* represent capacitance and conductance respectively.

Expression (1) is a 1st order differential equation. Assuming that initial voltage of each internal node is 0V at time *t*=0 when Tr is calculated, and *Vdd* at time *t*=0 when Tf is calculated, expression (1) can be solved by applying iteration methods such as Euler method. With this method, the voltage transitions of output nodes in the CCC are calculated. After Tr/Tf values are calculated and checked, fanout CCCs are analyzed. The accuracy of this method is measured using sample circuits in which four stages of cells are connected in series. Seven different kinds of cells are used, e.g., inverter, nand, nor, etc. The transistor width and the length of a net between cells are varied. In all, 105 sample circuits were used. We compare our method with the SPICE simulation. Figure 6 shows the resulting scatter diagram, where X-axis shows SPICE Tr/Tf values and Y-axis, the percentage error of our method with respect to SPICE. When the Tr/Tf values exceed the limit (shown as TfTf NG), the error is within 10%. Even inside the limit, the error is within -15% and +10% above 20 ps, as measured by SPICE. This is a tolerable error range. Our method is very fast and is applicable to large-scale custom macros such as RAMs. The CPU time for the actual analysis is shown in Table 4. The first two macros are logic custom macros, and the last three macros are RAMs. It takes more than one day for the large RAM 5 in Table 4.

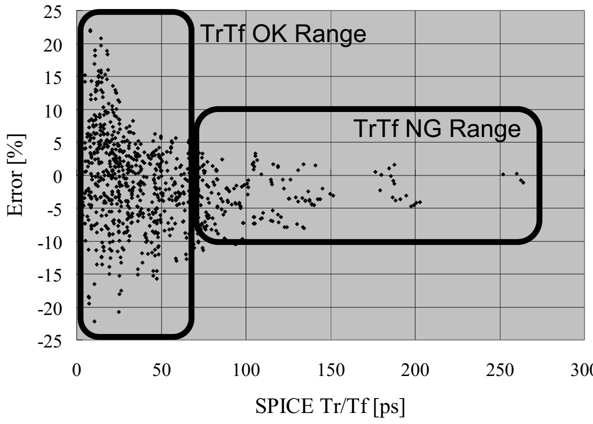

Figure 6. Scatter diagram of Tr/Tf analysis results

Table 4. Execution time in Tr/Tf analysis

Macro	# of Tr	# of R	# of C	CPU time(H)
No.1	12,054	190,740	409,582	0.3
No.2	22,304	319,780	837,945	2.3
No.3	2,056,140	6,421,074	4,777,646	14.5
No.4	1,109,245	3,479,851	2,014,669	10.5
No.5	291,288	3,461,953	8,047,243	27.7

7. Test

7.1 Logic BIST

Since the chip has dual cores, it is necessary to reduce time required in chip test. For this purpose, we adopted a logic BIST circuit to reduce the number of test vectors. Our experiments with the test vector generation based on the in-house ATPG tool show that the number of care bits that the ATPG sets in the first few vectors reaches about 80% of the total latches. But it decreases to about 15% in the 100th test vector. This ratio decreases further with the number of test vectors. The average of care bits in all test vectors is 1-5%. The don't care bits are randomly set to 0 or 1. Our logic BIST circuit [7] uses the fact that the number of care bits is small. In each test vector, the random values that are assigned by ATPG to don't care bits are internally generated by a random vector generator. Therefore a large part of the test data can be omitted from the tester storage. The remaining specified values are encoded to apply from outside the chip through the tester. General logic BIST architecture has Pseudo-Random Pattern Generator (PRGP) and Multiple-Input Signature Register (MISR). In our logic BIST architecture, an inverter block is inserted between the PRGP and scan chains. The inverter block is a circuit to invert a bit generated by the PRPG when it is different from the care bit and to supply it to the scan chain. An X-masking block is placed between the scan chain and the MISR. It protects values in the MISR by masking the value X. A decoder block controls the inverter block and the X-masking block by decoding test vectors supplied from ATE (Automatic Test Equipment). To enable the logic BIST circuit to distinguish good/no-good for each core, MISR is assigned to each core as shown in Figure 7. Due to this structure, MISR corresponding to each core can be read independently.

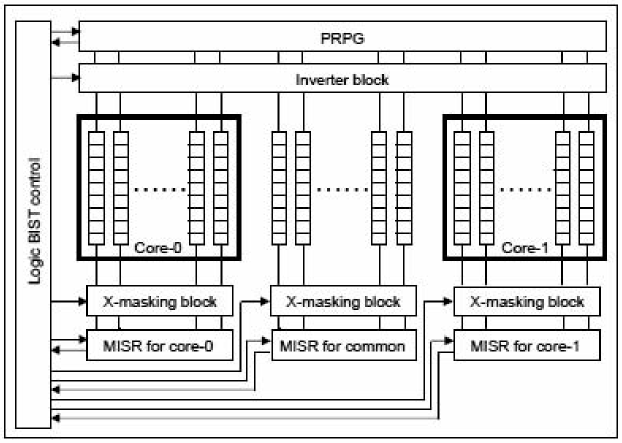

Figure 7. Logic BIST modified for a dual-core chip

When we think of fault diagnosis in a chip, we cannot know which scan cell fails in the structure in Figure 7. In this type of a logic BIST circuit, a MISR compresses the values as a signature. Therefore, it is very difficult to diagnose a fault as it is. In the test of a microprocessor, fault diagnosis is very important to investigate the root cause of a fault. Therefore, we adopted a circuit to switch logic BIST and conventional parallel scan chains. In logic BIST mode, there are 128 scan chains. In the conventional scan chain mode, those scan chains are reconstructed to form 7 scan chains corresponding to 7 sets of SDI/SDO external pins in the chip. Switching circuits are inserted between SDI/SDO external scan pins and scan chains. This reconfiguration is controlled by test port control machine (TPCM).

7.2 Test generation and verification

In the test generation of the microprocessor with dual cores, the number of test vectors is reduced by 87% by the logic BIST circuit as compared to the conventional scan chains. After test vector generation, we run timing simulation using delay information from static timing analysis. The strobe time, when the simulation results are compared to expected values described in the test vectors, is set to the actual tester strobe time. In the function test, outputs are strobed at a low frequency. In the delay test, outputs are strobed at a high frequency. As for the test vector verification, there are various opinions about its necessity [8]. In our microprocessor design, we perform test vector verification. This vector verification is not pure logic simulation; it includes timing simulation. Delay values are passed from static timing analysis to the test vector verification to cover the verification of cell libraries, design-for-testability circuits, and a timing analysis tool besides test vectors. In timing analysis, a designer can specify constraints such as false path elimination. Usually, these constraints cannot be checked by any CAD tools in the design flow. In our flow, mistakes in these constraints can be detected in this final test vector verification. In fact, this test vector verification detected several problems in our case. We summarize statistics about test vector generation and its verification in Tables 5 and 6 respectively. In test vector generation, the coverage of the function test is 99.7%, and that of the delay test is 91%. It takes one week to generate test vectors. In test vector verification, test vectors for the RAM BIST circuit are generated first, and the verification starts concurrently with the test generation for the remaining test items. Test vector verification was performed using 5 CPUs. It took 3 weeks from the test vector generation to its verification.

Table 5. Test Items

Test	# Faults	# Vectors	Coverage	Time (Hours)
SCAN	17,216,718	3,343	99.7%	0.6
FUNCTION	26,877,639			40.8
RAM BIST	N/A	N/A	N/A	0.1
DELAY	19,705,662	3,971	91.0%	110.0

Table 6. Time Required for Verification and Test

Test	Verification (Hours)	Relative Verification Time
SCAN	402.2	21.1%
FUNCTION	12.6	0.7%
RBIST	1452.5	76.3%
DELAY	35.8	1.9%
Total	1903.1	100.0%

8. Conclusion

In this paper, we presented new techniques used to design a 2.4GHz dual-core microprocessor. Data management for dual cores is improved. In timing analysis, turn-around-time for modification of the clock distribution circuit is reduced by treating split and shielded wires as one wire. In custom macro design, signal integrity analysis is enhanced. In test vector generation and its verification, a new logic BIST circuit is introduced. This reduces the number of test vectors by 87%, and enhances the capability of a fault diagnosis. Together, all these techniques enabled us to design the 2.4GHz dual-core microprocessor in a short time.

We will continue to improve this system so that it can support the development of much higher performance microprocessors with 4 or more cores in future. In particular, we will focus on statistical timing analysis, power grid analysis, and delay test and diagnosis to improve yield and reliability.

9. ACKNOWLEDGMENTS

We would like to thank the technology and CAD development teams for their significant effort in developing the tools. We wish to acknowledge Yuji Yoshida and Aiichiro Inoue for their several ideas and suggestions for our CAD development. Finally we also thank Kaoru Kawamura and other members of CAD group at Fujitsu Labs. Ltd. as well as the members of Advanced CAD Technology group at Fujitsu Labs. of America for their helpful suggestions.

10. REFERENCES

[1] A. Inoue, "SPARC64TM V/VI for Mission-Critical Servers", presented at Fall Processor Forum, 2004.

[2] A. Inoue, "Fujitsu SPARC64 VI: A State of the Art Dual-Core Processor", presented at the Fall Processor Forum, 2006.

[3] N. Ito, H. Komatsu, et al, “A Physical Design Methodology for 1.3GHz SPARC64 Microprocessor”, Proc. ICCD, pp. 204-210, 2003.

[4] W. C. Elmore, “The Transient Response of Damped Linear Networks”, Journal of Applied Physics, Vol. 19, pp. 55 - 63, January 1948.

[5] P. Kulshreshtha, R. Palermo, et al, “Transistor-Level Timing Analysis Using Embedded Simulation”, Proc. ICCAD, pp. 344-348, 2000.

[6] C. Alpert, A. B. Kahng, et al, “Minimum-Buffered Routing of Non-Critical Nets for Slew Rate and Reliability Control”, Proc. ICCAD, pp. 408-415, 2001.

[7] T. Hiraide, K. O. Boateng, et al, “BIST-Aided Scan Test - A New Method for Test Cost Reduction”, Proc. VTS, pp. 359-364, 2003.

[8] R. Raghuraman, "Simulation Requirements for Vectors in ATE Formats", Proc. International Test Conference, pp.1100-1107, 2004.