International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 14
October 2016

In-Field Test for Permanent Faults in Fifo Buffers Of NOC Routers

CHAMANTULA RADHIKA
DEPARTMENT OF ECE
HOLYMARY INSTITUTE OF TECHNOLOGY AND SCIENCE

D.PADMA SRI
Assistant Professor
DEPARTMENT OF ECE
HOLYMARY INSTITUTE OF TECHNOLOGY AND SCIENCE

ABSTRACT

This brief proposes an on-line transparent test technique for detection of latent hard faults which develop in first input firstouptput buffers of routers during field operation of NoC. The technique involves repeating tests periodically to prevent accumulation of faults. A prototype implementation of the proposed test algorithm has been integrated into the router-channel interface and on-line test has been performed with synthetic self-similar data traffic. The performance of the NoC after addition of the test circuit has been investigated in terms of throughput while the area overhead has been studied by synthesizing the test hardware. In addition, an on-line test technique for the routing logic has been proposed which considers utilizing the header flits of the data traffic movement in transporting the test patterns.

Index Terms—FIFO buffers, in-field test, NoC, permanent fault, transparent test.

INTRODUCTION

Multiprocessor systems-on-chips (MPSoCs) have emerged in the past decade as an important class of very large scale integration (VLSI) systems. An MPSoC is a system on- chip a VLSI system that incorporates most or all the components necessary for an application that uses multiple programmable processors as system components. MPSoCs are widely used in networking, communications, signal processing, and multimedia among other applications .A trend of multiprocessor system-on-chip (MPSoC) design being interconnected with on-chip networks is currently emerging for applications of parallel processing, scientific computing, and so on.

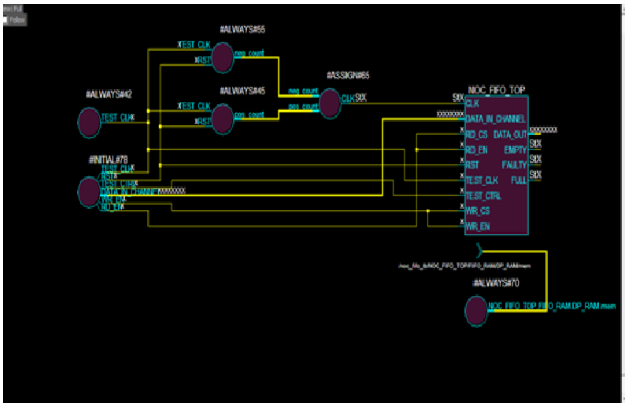
Network on chip

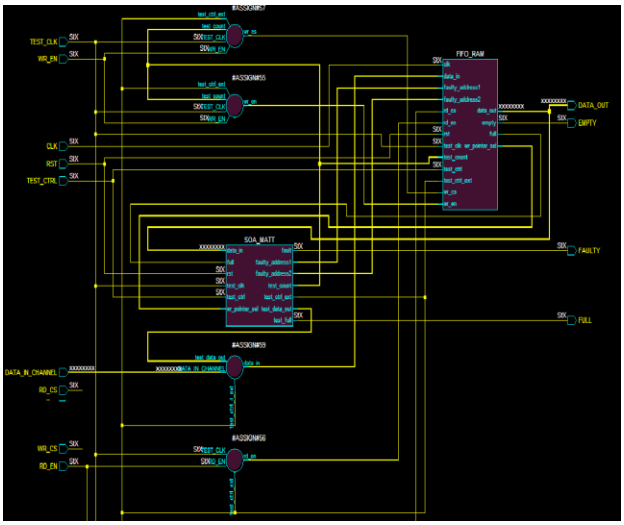
Network on chip or network on a chip (NoC or NOC) is a communication subsystem on an integrated circuit , typically between intellectual property (IP) cores in a system on a chip (SoC). NoCs can span synchronous and asynchronous clock domains or use unclocked asynchronous logic. NoC technology applies networking theory and methods to on-chip communication and brings notable improvements over conventional bus and crossbar interconnections. NoC improves the scalability of SoCs, and the power efficiency of complex SoCs compared to other designs.

Routing on NoC

Routing on NoC is quite similar to routing on any network. A routing algorithm determines how the data is routed from sender to receiver. Routing algorithms are divided into two groups, oblivious and adaptive algorithms. Oblivious algorithms are also divided into two subgroups: deterministic and stochastic algorithms. Oblivious algorithms route packets without any information about traffic amounts and conditions of the network, deterministic algorithms route packets always along a same route and stochastic routing is based on randomness

BLOCK DIAGRAM





Design Modules

- **NOC FIFO TOP**
- **FIFO SRAM**
- **SOA MATT ALGO**
- **ASYNC DP RAM**

Functional Blocks

The IC's are in charge of processing the incoming probe headers from upstream switches. When an incoming probe header arrives at an input (with Req= "1"), the corresponding IC's monitors the output status through Monitor bus and requests ARBITER to grant it access to the desired OC's through the internal Request bus. Based on output status or the feedback from ARBITER placed in Grant & Answer bus, the IC's operates appropriately and replies to the upstream switch through its Ans_In.

ARBITER

The ARBITER has two roles:

1. First, for cross-connecting Ans_Out signals back to IC's, and second, as a referee for requests from IC. If a requesting IC is accepted, the first role of ARBITER keeps Ans _signal from the requested Ans_Out being connected back to the requesting IC through the Grant & Answer bus.

By this way, in case of backtracking, the ARBITER can direct the probe header to backtrack to the corresponding input from which it arrived (reserved) previously. Contention resolution is required when several IC's, upon observing the Monitor bus, request the same idling output in the same probing clock cycle.

2. The second role of ARBITER (based on a static priority rule) allows only one request to be accepted, while the remainders are answered with "Network Blocked". By receiving this answer, the requesting IC's continues probing other outputs or returns a "Busy Dest." to its corresponding upstream switch, depending on which probed port is the direction port. The OC's, based on the command from the ARBITER placed in the Control bus, make requests to the downstream switches and control the CROSSBAR. When locked with a specific selecting value, the OC's handles the CROSSBAR to establish a direct connection from the Data In to the target Data out.

Programmable Arbitration schemes:

In the previous design of the switching circuit consists of Arbiter as shown in above figure, can be programmed and configurable with different arbitration schemes to overcome drawback in the previous system with arbitration schemes, new design has proposed with new arbitration scheme for better efficiency.

The previous Arbitration schemes

i. Round Robin ii. Fixed Priority

Data encoding schemes

Different encoding techniques are proposed to reduce the power in reference to the bus based architecture. Bus invert method can be applied to encode the randomly distributed patterns.

This encoding scheme is developed by taking the both factor self switching and the cross talk into consideration.

The encoder and decoder are placed in the network interface of the wormhole routed network. The first stage rearranges the data stream in such way that the transition in each link is reduced while in the second stage the inverting of data of depends upon the contribution of the cross couple activity in power dissipation of the link, The proposed encoding scheme

can be applied to the wormhole routed network as the interleaving of flits are not allowed.

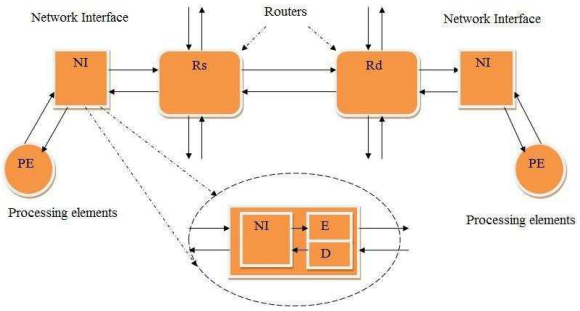
Overview of the technique:

The general scheme diagram is shown below. The basic idea is to apply an encoding technique

end-to-end taking advantage of the wormhole switching technique. In fact, wormhole switching is the most suitable option for on-chip communication. The rationale behind this idea is due to the pipeline nature of wormhole switching. Since all the links of the routing path are traversed by the same

sequence of flits, the encoding decision taken at the network interface guarantees the same switching behavior in each link of the routing path. As shown in below Fig. the NI is augmented with an encoder

(E) and a decoder (D) block. With the exception of the header flit, the encoder encodes the outgoing flits of the packet in such a way as to minimize the power dissipated by the interrouter point-to-point links which form the routing path of the current packet. Since the routers are not equipped with any encoding/decoding logic, the header flit is not encoded as it contains control information (destination address, packet size, and so on) which have to be processed by the routers through the routing path. Similarly to the above description, all the incoming flits in the network interface (with the exception of



Encoding scheme

Power Model

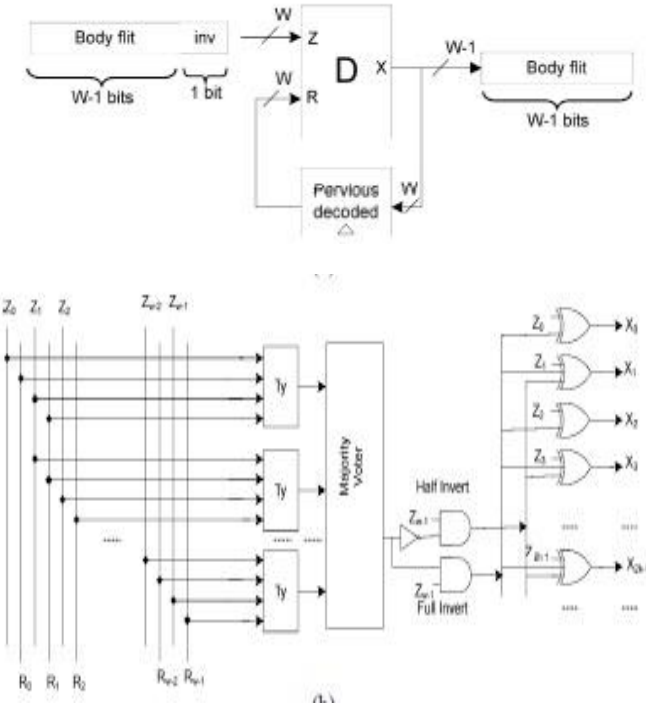
The dynamic power consumed by the interconnects and drivers is given by

$P = [T0\rightarrow1(Cs + Cl) + TcCc] V^2$ dd Fck

where Vdd is the supply voltage, Fck is the clock frequency, Cs is the self capacitance (which includes the parallel-plate capacitance and the fringe capacitance), Cl is the load capacitance, and Cc is the coupling capacitance. T0→1 and Tc are the average number of effective transitions per cycle for Cs and Cc, respectively. They are computed as follows.

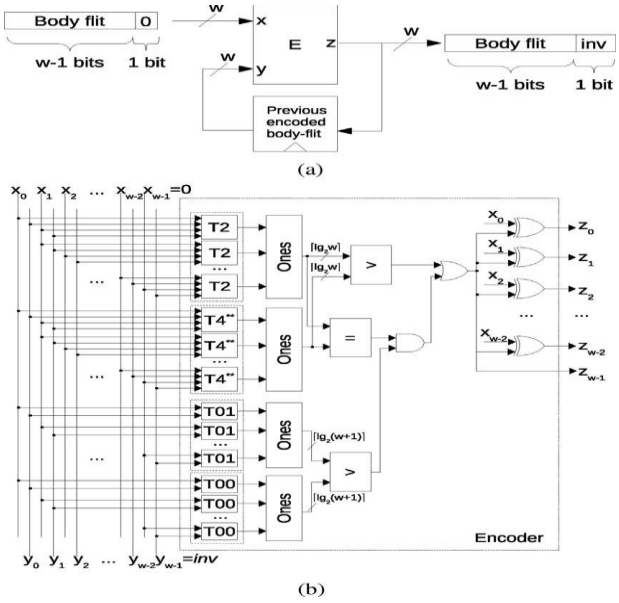
T0→1 counts the number of 0 → 1 transitions in the bus in two consecutive transmissions. Tc counts the correlated switching between physically adjacent lines. Precisely, we can enumerate four types of coupling transitions as follows .

Decoder technique



Decoder architecture . (a) Circuit diagram. (b) Internal view of the decoder block (D).

Encoder scheme



VERILOG

Verilog HDL is one of the two most common Hardware Description Languages (HDL) used by integrated circuit (IC) designers. The other one is VHDL.

HDL's allows the design to be simulated earlier in the design cycle in order to correct errors or experiment with different architectures. Designs described in HDL are technology-independent, easy to design and debug, and are usually more readable than schematics, particularly for large circuits.

Verilog is a language that can be understood by system designers, RT level designers, test engineers, simulators, synthesis tools, and machines. Because of this important role in design, Verilog has become an IEEE standard. The standard is used by users as well as tool developers. Verilog is a hardware description language for describing hardware from transistor level to behavioral. The language supports timing constructs for switch level timing simulation and at the same time, it has features for describing hardware at the abstract algorithmic level. A Verilog description may consist of a mix of modules at various abstraction levels.

DESIGN METHODOLOGIES

There are two basic types of digital design methodologies:

A top-down design methodology

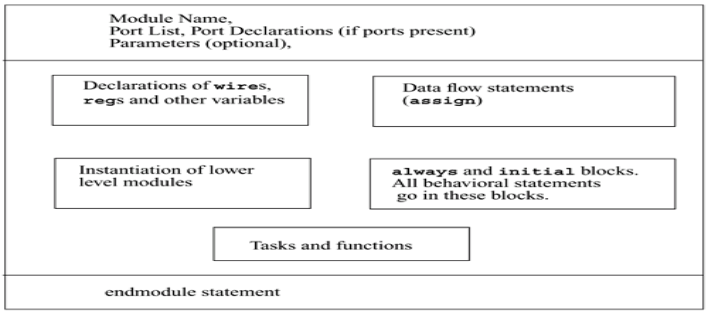
A bottom-up design methodology

MODULES

Verilog provides the concept of a module.A module is the basic building block in Verilog. It can be an element or a collection of low level design blocks. Typically, elements are grouped into modules to provide common functionality used in places of the design through its port interfaces, but hides the internal implementation.

COMPONENTS OF A VERILOG MODULE

The end module statement must always come last in a module definition. All components except module, module name, and end module are optional and can be mixed and matched as per design needs. Verilog allows multiple modules to be defined in a single file. The modules can be defined in any order in the file.



SIMULATION AND FUNCTIONAL VERIFICATION

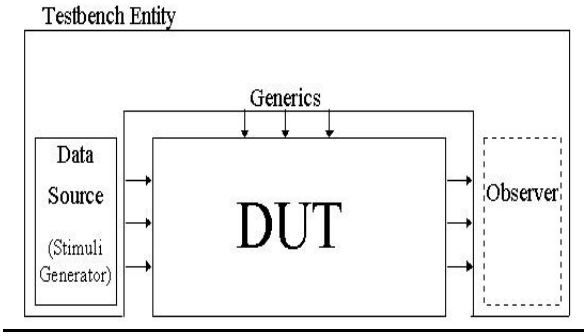
Simulation

Once a design block is completed, it must be tested. Simulation is the process of verifying the functional characteristics of models at any level of abstraction. We use simulators to simulate the the Hardware models. The functionality of the design block can be tested by applying stimulus and checking results. We call such a block the stimulus block. It is good practice to keep the stimulus and design blocks separate. The stimulus block can be written in Verilog. A separate language is not required to describe stimulus. To test if the RTL code meets the functional requirements of the specification, see if all the RTL blocks are functionally correct. To achieve this we need to write testbench, which generates clk, reset and required test vectors. The stimulus block is also commonly called a test bench.

Test inputs for testbenches

Any digital system has to carry out a number of activities in a defined manner.once a proper design is done, it has to be tested for all its functional aspects. The system has to carry out all the expected activites

.Futher it should not malfunction under any set of input functions. Functional testing is carried out to check for such requirements.
Test inputs can be purely combinational, periodic, numeric sequences, random inputs or combinational inputs or combination of these.
As the circuit design proceeds, one develops smaller blocks and groups them together to form bigger circuit units. The process is repeated until the whole system is built up. Every stage calls for tests to see whether the subsystems at that layer behaves in the manner expected. Such testing calls for two types of observations:
1. study of signals within a small unit when test inputs are given to the whole unit.
2. isolation of small element and doing local test to facilitate debugging.



SYNTHESIS

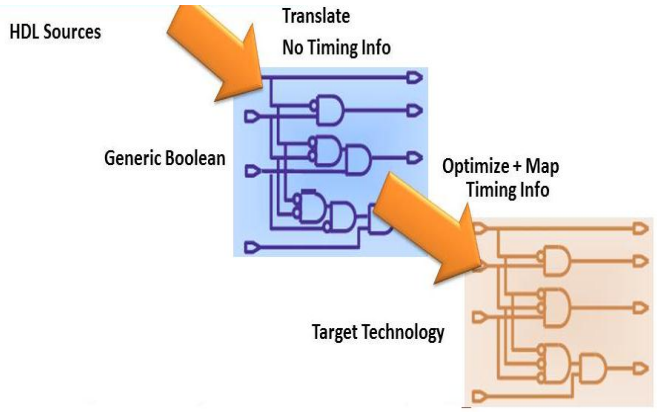
logic synthesis deals with the synthesis and optimization of circuits at the logic gate level.Synthesis is the process of transforming your HDL design into a gate-level netlist, given all the specified constraints and optimization settings.

Logic synthesis is the process of translating and mapping RTL code written in HDL (such as Verilog or VHDL) into technology specific gate level representation.

There are 3 steps in Synthesis:

- **Translation**: RTL code is translated to technolohgy independent representation. The converted logic is available in boolean equation form.
- **Optimization:** Boolean equation is optimized using SoP or PoS optimization methods.
- **Technology mapping:** Technology independent boolean logic equations are mapped to technology dependant library logic gates based on design constraints, library of available technology gates. This produces optimized gate level representation which is generally represented in Verilog.

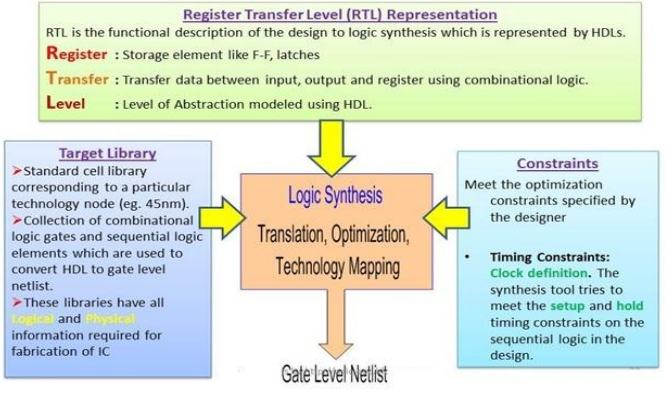
Then the gate level circuit generated is logically optimized to meet the targets or goals set as per the user constraints. The clock frequency target is the number one goal that has to be met by the synthesis operation.



Synthesis = Translation + Optimization + Mapping

Inputs and Output from ASIC Synthesis flow

Outcome of Synthesis is Gate level netlist which is again in Standard Verilog format. Netlists can be simulated as well which we call as Gate Level Simulation.



Register Transfer Level (RTL) Representation

RTL is the functional specification of the design to logic synthesis which is represented by HDLs.

- **Register:** Storage element like F-F, latches
- **Transfer:** Transfer data between input, output and register using combinational logic.
- **Level:** Level of Abstraction modeled using HDL.

Constraints

The major objective of the logic synthesis is to meet the optimization constraints specified by the designer. Timing, area and power targets are the optimization constraints.

- **Timing Constraints:** The synthesis tool tries to meet the setup and hold timing constraints on the sequential logic in the design.
- **Area constraints:** Area constraints specifies maximum area for a design.
- **Power Constraints:** Power constraints specifies the maximum power consumption for the design.

Target Library

Target library is standard cell library corresponding to a particular technology node (eg. 45nm). This is a collection of combinational logic gates and sequential logic elements which are used to convert HDL to gate level netlist.

If logic synthesis is carried ou for FPGAs then HDL description is translated and mapped to LUTs, flip-flops and block RAMs. For FPGA implementation separate synthesis tool is required. ASIC synthesis tool can't synthesize the HDL into FPGA omplemtable netlist.

A clean technology independent HDL description of design can be synthesized to any technology node. This can also be targeted for FPGA implementations.

CONCLUSION

In this brief, we have proposed a transparent SOA-MATS++ test generation algorithm that can detect run-time permanent faults developed in SRAM-based FIFO memories. The proposed transparent test is utilized to perform online and periodic test of FIFO memory present within the routers of the NoC. Periodic testing of buffers prevents accumulation of faults and also allows test of each location of the buffer. Simulation results show that periodic testing of FIFO buffers do not have much effect on the overall throughput of the NoC except when buffers are tested too frequently. We have also proposed an online test technique for the routing logic that is performed simultaneously with the test of buffers and involves utilization of the unused fields of the header flits of the incoming data packets for test pattern encoding. As future work, we would like to modify the proposed FIFO testing technique that will allow incoming data packets to the router under test without interrupting the test.

REFERENCES

[1] *International Technology Roadmap for Semiconductors: Interconnect*.(2006) [Online]. Semiconductor Industry Assoc. Available: www.itrs.net

[2] S. Pasricha and N. Dutt, "Trends in emerging on-chip interconnect technologies," *IPSJ Trans. Syst. LSI Design Methodol.*, vol. 1, pp. 2–17,Aug. 2008.

[3] H.-J. Yoo, K. Lee, and J. K. Kim, *Low-Power NoC for High-Performance SoC Design*. Boca Raton, FL: CRC Press, 2008.

[4] S. Borkar, "Thousand core chips: A technology perspective," in *Proc. ACM/IEEE Design Autom. Conf.*, Jun. 2007, pp. 746–749.

[5] G. D. Micheli and L. Benini, *Networks on Chips: Technology and Tools*. San Mateo, CA: Morgan Kaufmann, 2006.

[6] J. A. Davis, R. Venkatesan, A. Kaloyeros, M. Beylansky, S. J. Souri, K. Banerjee, K. C. Saraswat, A. Rahman, R. Reif, and J. D. Meindl, "Interconnect limits on gigascale integration," *Proc. IEEE*, vol. 89, no. 3, pp. 305–324, Mar. 2001.

[7] J. D. Meindl, "Interconnect opportunities for gigascale integration," *IEEE Micro, Special Issue Reliab.-Aware Microarchitecture*, vol. 23, no. 3, pp. 28–35, May 2003.

[8] S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar, "An 80-tile sub-100-W TeraFLOPS processor in 65-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 29–41, Jan.

2008.

[9] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, J.-W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal, "The raw microprocessor: A computational fabric for software circuits and general-purpose programs," *IEEE Micro*, vol. 22, no. 2, pp. 25–35, Mar.–Apr. 2002.

[10] F. Steenhof, H. Duque, B. Nilsson, K. Goossens, and R. P. Llopis, "Networks on chips for high-end consumer-electronics TV system architectures," in *Proc. Conf. Design Autom. Test Eur.*, 2006, pp. 148–153.

[11] A. Ejlali, B. M. Al-Hashimi, P. Rosinger, S. G. Miremadi, and L. Benini, "Performability/energy tradeoff in error-control schemes for on-chip networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 18, no. 1, pp. 1–14, Jan. 2010.