ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 9(7), DOI: 10.17485/ijst/2016/v9i7/87835, February 2016* ISSN (Online) : 0974-5645

Pseudo-System-Level Network-on-Chip Design and Simulation with VHDL: A Comparative Case Study on Simulation Time Trade-Offs

Negin Mahani*

Zarand Higher Education Complex, Computer Engineering Department, Shahid Bahonar University, Kerman, Iran; Negin.mahani @uk.ac.ir

Abstract

Background/Objectives: To meet the challenge of increasing design complexity, designers are turning to System Level Design Languages (SLDLs) to model systems at a higher level of abstraction. **Methods/Statistical Analysis:** Now there are some system level languages like SystemC but hardware developers prefer HDL based languages like VHDL for coding. So focusing on methods for extending VHDL for system-level modeling is the issue of hardware modeling researches. VHDL itself has some high level structures to model near system-level. Here we have implemented a 9 switch Network-on-Chip (NoC) with processing and communication elements like FIFOs and we have tried to eliminate signals as communication elements between processing components and using high level structures in addition to resolution function (for the first time in NoC structure) in routing algorithm. **Finding:** Resolution function can decrease simulation speed as in the literature mentioned so we have applied some improving techniques for simulation accelerating to see the result of these tradeoffs. All in all the one with resolution function and other high level structures besides applying improving speed rules has better performance and we have gain about 28% speed up and 35% speed up in contrast to the latter one without eliminating possible signals. **Conclusion/Application:** All in all by using accelerating rules we had no simulation time penalty and gained 28% speed up. Resolution function is a high level structure which is used for hardware purposes, it is preferable than simply implementing the routing algorithm by other common statements

Keywords: High Level Structures, NoC, Resolution Function, System Level Design, VHDL

1. Introduction

In hardware design, we are familiar with several levels of abstraction: transistor, gate, Register Transfer Level (RTL) and Transaction Level Modeling (TLM). It has been apparent that the Register Transfer Level is too low for the size of hardware systems which are being fabricated, due to increasing chip capacity. So designers prefer more abstract levelsy[1].

System simulations are vital for pre-silicon development. One of the bottlenecks that face current developers is the massive simulation times that arise with such embedded systems especially that the sizes and complexities of such circuits are increasing. A way to decrease simulation times is to change the abstraction level in which the system is defined, such that the system still performs the same function but with less simulation complexity. Simulating a complex system defined in RTL would mean that the simulator will have to monitor each internal register in the system at each clock cycle and compute how the register value should change. An abstraction level like TLM would offer some communication abstraction to the system such that the intermediate signals that connect different modules of the system can be removed. By applying such abstraction to an RTL system, simulation times of a System-on-Chip (SoC) with high communication rates between its modules would be reduced significantly[2].

Object oriented languages make creating new levels of abstraction in a single language particularly easy and in fact, that is the primary virtue of C++. SystemC is nothing more than a hardware level of abstraction implemented

* *Author for correspondence*

by extension in C++. More important is that using C++, additional data types and operations can be defined to create higher levels than the base SystemC level. TLM is also an extension to SystemC[3,4].

Design flow must utilize Hardware Description Languages, synthesis and co-simulation capabilities to achieve these goals. Hardware Description Languages (HDL) such as VHDL has been developed that allow the description of the behavior and the structure of a digital system in a simulatable and synthesizable form. The aim of this paper is to focus on previous works in extending VHDL for system level modeling extracting VHDL challenges in system level. Besides we have implemented a NoC infrastructure close to TLM level by using processing and communication elements and tried to eliminate using communication signals as much as possible. We have proposed to use resolution function as a built in language structure in NoC routing algorithm development besides other high level structures and observe the effect of these high level structures in simulation time. Section 2 is about system level HDLs. Section 3 is about VHDL system level challenges and high level structures. Next section includes applying high level structures in order to increasing the abstraction level in VHDL models and simulation accelerating techniques. Section 5 describes implementing a NoC structure close to system level with VHDL. The simulation result is come in the next section and the final part is the conclusion.

2. System Level HDLs

The first question which any designer would encounter is that which language should be used for system level hardware design? There are possible choices such as SystemC, System-Verilog, Ada and etc. The answer to this question is very dependent on the purpose of system level modeling but some tradeoffs from inside and outside of the language, such as language constructs and semantics, tool support, third-party IP availability and access to knowledgeable engineers are also interfered in this choice. The most popular languages which are used in hardware design nowadays are VHDL and Verilog which both are suitable for RTL and have substantial disabilities to cover system level. System-Verilog is a system level language from Verilog family with verification purpose in mind from first advent. VHDL is used for high level design, but it lacks abstract timing and communication, genericity and Object Oriented modeling. Some groups like SUAVE have proposed some extensions to VHDL to cover system level. Most of the extensions are added from Ada95 as the base language of VHDL from early development. It shows that Ada has potentials to be used as an HDL. Ada as an HDL has a long history which is out of scope of this paper[5–7].

Among all of the languages mentioned above, SystemC and System-Verilog are more common. The base language of System-Verilog is Verilog and its main focus is on RTL modeling like Verilog. The main purpose of this language is verification. The enhancements related to directed test generation, assertion definitions and coverage metrics are all very valuable capabilities and all are closely tied to the RTL implementation domain.

SystemC is a class library in C++ and TLM is patched after on top of SystemC. Due to lots of patches, there are many problems with debugging which are the most common user problems with this language. Ada is the base language of VHDL and it has inherent concurrency as well as object oriented structures, so it has been chosen to extend VHDL for System-Level modeling in some previous works. The most important advantages of Ada over SystemC are listed as below:

- High readability and well descriptive language.
- Faster and more convenient debugging.
- Inherent concurrency and interface types (not being patchy).
- Having the link to RTL.
- Capability to export and import to/from foreign languages.
- Shorter simulation time[8–10].

2.1 VHDL Unique Advantages

There are lots of benefits of using VHDL in hardware description. Some more significant ones are summarized as below:

- Executable specification.
- Validate spec in system context.
- Functionality separated from implementation.
- Simulate early and fast (Manage complexity).
- Explore design alternatives.
- Get feedback (Produce better designs).
- Automatic synthesis and test generation (ATPG for ASICs).
- Increase productivity (Shorten time-to-market).
- Technology and tool independence (though FPGA features may be unexploited).
- Portable design data (Protect investment).

3. VHDL System Level Challenges and High Level Structures

The first problem in system-level modeling by VHDL is about how to get rid of signals. Signals as a sign of low level hardware description has no place in high level modeling. The second characteristic required for supporting system level in VHDL is addition of object orientation features, since in every system level HDLs like TLM there are roots of object orientation basics. OO-VHDL is a name of a project which has worked on this issue[11]. Other researches for extending VHDL in system level is done by Ashenden called SUAVE project. In these works because of the similarity to Ada syntax they have tried to rent high level structure from Ada programing language. They have presented some requirements document called SUAVE specification for entities called channels for data transport and using function calls for sending and receiving data just like the features which really exist in TLM nowadays[12,13].

There are more constructs and features for high-level modeling in VHDL than there are in most of HDLs. Abstract data types can be used along with the following statements:

- Package statements for model reuse.
- Configuration statements for configuring design structure.
- Generate statements for replicating structure.
- Generic statements for generic models that can be individually characterized, for example, bit width.

All these language statements are useful in synthesizable models[14].

4. Techniques for Increasing the Abstraction Level/Simulation Speed

An important step towards a more efficient design methodology is to increase the abstraction level in the design process. Describing an adder with a '+' rather a network of AND, OR and XOR gates is much more readable and also less error-prone.

No matter how fast a simulator gets, the HDL developer can further improve performance by applying a few simple guidelines to the coding style. The key to higher performance is to avoid code that needlessly creates additional work for the HDL compiler and simulator. The following are the general techniques that have a high performance impact; some of them also increase the abstraction level of the design:

- Use Optimized Standard Libraries.
- Use Loop statement (The loop statement is supported by most synthesis tools as long as the loop range is constant).
- Reduce Process Sensitivity.
- Reducing waits.
- Reduce or Delay Calculations.
- Limit File I/O (Reading or writing to files during simulation is costly to performance, because the simulator must halt and wait while the OS completes each transaction with the file system).
- Multiplexing using integer conversion.
- Use State machines.
- Integers vs. Vectors (Arithmetic operations on Standard Logic Vectors (SLVs) are expensive compared to integer operations).
- Avoid Slicing Signals.
- Avoid the "Linear Testbench".
- Use sub-programs.

4.1 Resolution Function an Example of Special Sub-Programs

Using sub-programs (procedures and functions) is a powerful method to hide complexity and improve readability. Tested and reusable sub-programs can be kept in a separate package and use as an IP library of small algorithms (Figure 1).

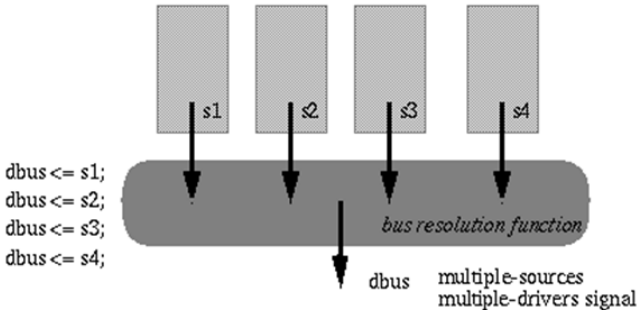

Figure 1. Resolution function.

A resolution function defines how values from multiple sources, multiple drivers, are resolved into a single value. A type may be defined to have a resolution function. Every signal object of this type uses the resolution function when there are multiple drivers. A signal may be defined to use a specific resolution function. This signal

uses the resolution function when there are multiple drivers. A resolution function must be a pure function that has a single input parameter of class constant that is a one dimensional unconstrained array of the type of the resolved signal.

5. VHDL System Level Modeling of Noc Infrastructure (Accelerating Simulation Approach)

In this section we have tried to implement a NoC as high level as possible. To reach this aim we have tried to eliminate using signals when it was possible and try to use high level structures including resolution functions.

In order to accelerating simulation we have done the following steps where it was possible to overcome the overhead of using resolution function and resolved signal.

- Minimizing the number of signal assignment(using variable assignment instead).
- Reducing the number of signals (including implicit signals, Variables should be used instead of signals).
- Avoiding large vectors/ complex records.
- Minimizing the number of concurrent statements(grouping common functions within processes, all registers can be updated in a single process, operations sensitive to the same signals can be grouped in a same process).
- Avoid repeated codes or function calls(reduce computations in the redundant paths by saving temporary results in variables).
- Using numerical data types such as Integers rather than Std_Logic and St_Logic_Vector and Bit Vector.
- Avoid type conversions.
- Enumerated types have better simulation speeds than constrained[15].

5.1 NoC Switch Design

We have designed and implemented a high level model for NoC switches with five identical ports, routing logic and a routing table (here it is a function). Each port contains an input buffer for storing the incoming packets (link list FIFO). Each packet must be a record that includes a header which determines the destination address and a data payload. When a packet arrives, it will be stored in the input buffer. The router continuously checks received packets and according to their destinations routes them to the appropriate output port. The input buffer is a circular FIFO for storing input packets of neighbor switches. We have put an extra field in the packet for an indication of how long it has taken the packet to arrive at its destination. In our parametric design we have used high level structures of VHDL (Figure 2, Figure 3)[16,17].

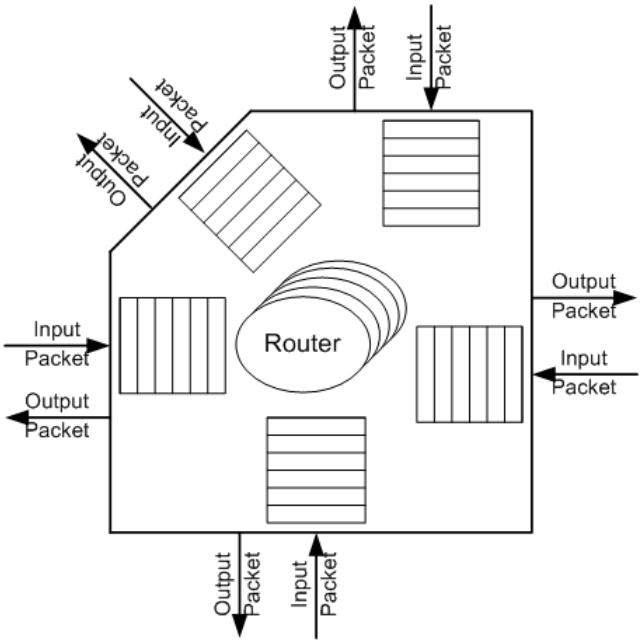

Figure 2. NoC switch block diagram.

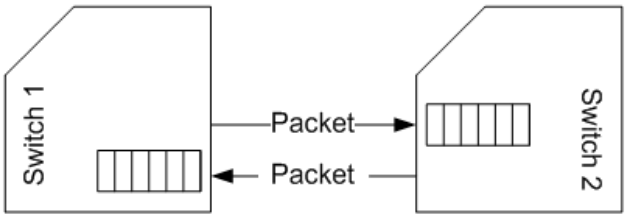

Figure 3. Two NoC switch relation.

5.2 NoC Switch Implementation

In our NoC we have routing function that according to the destination of the packet routs it. If we have some packets from different ports that want to go out from the same out port, we call a resolution function to decide between them. This decision could be based on the life time or based on the order of the ports. We have chosen 'left attribute. Each packet which has been removed from the input FIFO, should be kept to be routed next time. Because of this fact, we have a function to tick the remained packets in

order to be valid and not be destroyed. (The array that shows the port has a valid packet and it has not been routed yet is rout_full when rout full (1) Is one it means that we should not request to the FIFO and a packet is ready, before a request), As it was told before we call the resolution function when we have some packets for the same out port. The code of this function has come below (Figure 4).

```
FUNCTION res_func(drivers:packet_vector)RETURN  packet IS
 VARIABLE pack:packet:=(0,0,0NS,0,0,0,0);
 BEGIN
 report"resolution function is called";
 FOR i IN drivers'RANGE LOOP
 pack: = drivers(drivers'LEFT);
 END LOOP;
 RETURN pack;
END res_func;
```

Figure 4. Resolution function for routing.

```
SUBTYPE resolved_pack IS res_func packet;
TYPE resolved_pack_vector IS ARRAY (NATURAL RANGE <>) OF resolved_pack;
```

Figure 5. Resolved type.

To use this function we have defined resolved_packet_vector as resolved type of packet_vector (Figure 5).

5.3 IP Blocks Implementation

There are six different IP blocks in the NoC. All of them are simple processor units. P1, P9 (Master processors) read the input data from a specified file and send it to all other processors. Also these processors gather responses and write them in separate output files. P2, P4, P6, P8 (Slave processors) get the received packet and process it. Then they make new packet as processing result packet and send it to the proper destination that is mentioned in incoming packet. Processing task of each slave processor is not so important they just put their number instead of the payload of the packet which is going to be processed. Here the processes are not important we have done this for simplicity. Each slave processor sends the incoming packet after processing to one of the master processors according to the following list:

P2, P4 = >P9

P6, P8 = > P1

Figure 6 shows sending packets from P1 to all other processors and gathering responses (dotted lines) in P1 and P9. Also P1 sends the data packets to P9. In this case P9 writes received packets in an output file. P9 processor application is like P1 processor.

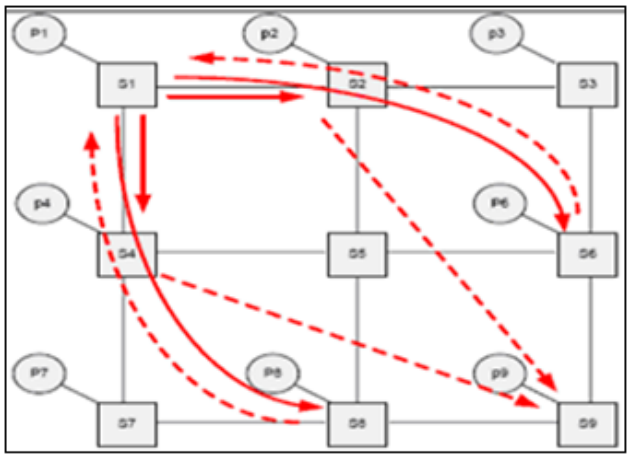

Figure 6. NoC switch scenario.

5.4 Packet Format

Packets must contain the following fields at least in addition to any extra field that is required for computing number of packets that each switch pass and the delay.

Source address => the source node that generates and sends the packet.

Destination address => the destination address for that packet.

Response node address => the address of the node that the response must send to it.

Data => the data payload of each packet (And other necessary fields).

There are two different input files each of which belongs to one of the master processors.

5.5 Output Files Format

Each master processor makes an output file for each of processors that send their responses to it. Master processor writes the information of each received packet in corresponding response file according to the source address field. Information of each packet is kept separately and it contains some information about delay, number of switches and the processed data and some other useful information.

6. Simulation Results

The simulation result wave form is illustrated in Figure 7. By comparing the simulation speed of our NoC model and the one without resolution function and accelerating

techniques (both models were closed to system level with eliminating signals where it was possible) we have yield about 28% speed up. Resolution function is a high level structure which is used for hardware purposes, it is preferable than simply implementing the routing algorithm by other common statements. Again we have done the simulation time comparison between our NoC and the other one without eliminating possible signals (in the latter). This time we have gained 35% speed up. It was not possible for us to eliminate all signals in the project so the gained speed up was not the real speed up in TLM over lower levels but by decreasing the number of signals we must have reached a significant speed up though. This speed up is because that signal modification due to lots of attributes is a time consuming work but variable modification is not the same and some other high level structures are processed more quickly than low level ones. To say more precisely the overhead of each signal is as follows (for useful information about impact of description language, abstraction layer and value representation on simulation performance refer to[18,19]):

- Each signal requires one or more drivers.
- Specific handling and event scheduling.
- Memory storage.
- More instructions to execute.

7. Conclusion

The reason for the importance of being able to model hardware in VHDL is clear. Hardware modeled in one language can also be modeled in the other. The choice of HDL is shown not to be based on technical capability, but on personal preferences, EDA tool availability and commercial, business and marketing issues. VHDL simply addresses the hardware-related characteristics of parallelism and structure. Standard VHDL have good facilities for extending the level of abstraction, attempts to use it for any of the higher levels requires non-standard additions or interpretations of its existing facilities. The amount of code to be written for the design decreases as the level of abstraction increases, which reduces the probability of coding errors. The simulation speed

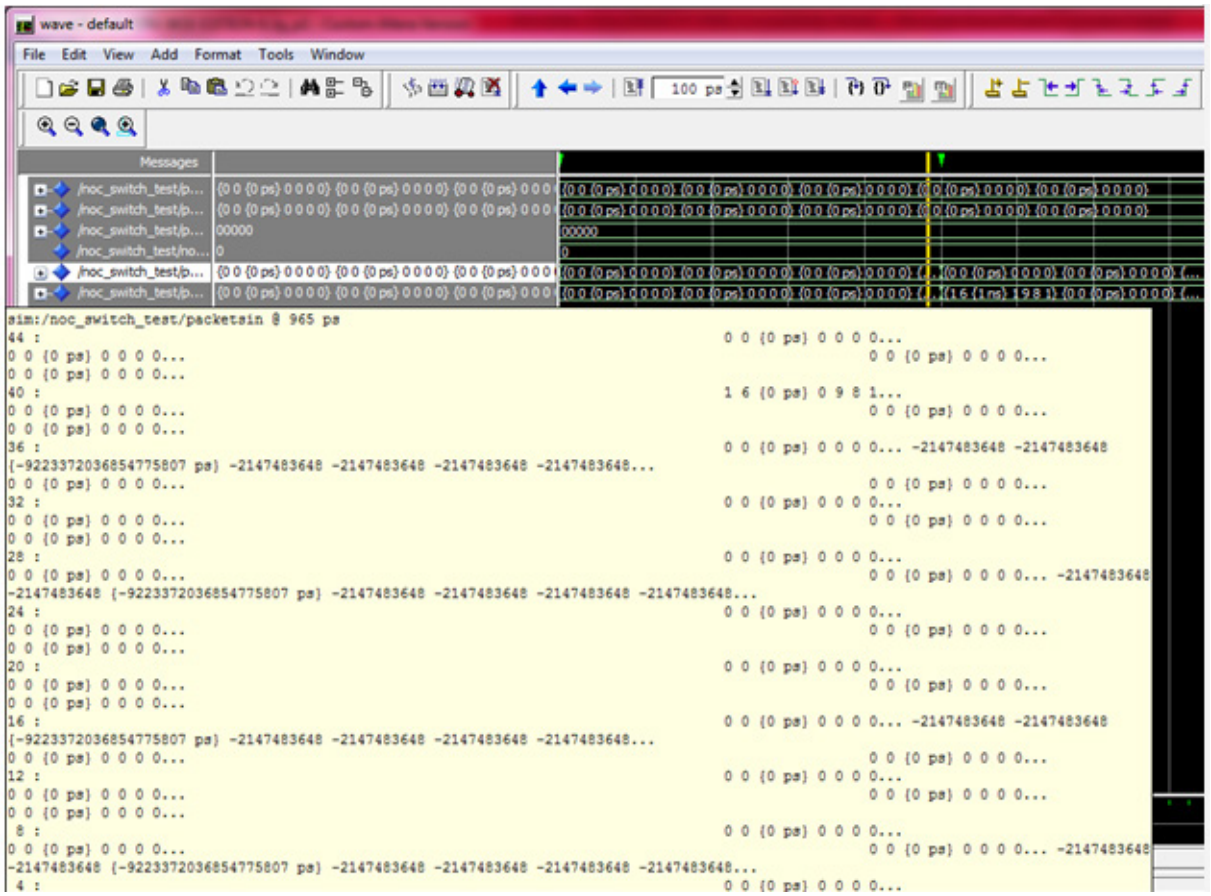

Figure 7. Simulation results of NoC switch.

increases as the level of abstraction increases. In addition, the generality of writing code at higher levels can result in a more general implementation. Using sequential VHDL statements to code the algorithm also allows the use of complex statements and a higher abstraction level. Debugging and analysis is simplified due to the serial execution of statements, rather than the parallel flow used in dataflow coding. In this paper we have summarized VHDL challenges which we have encountered in our NoC design. We have utilized some high level VHDL structures to model our NoC close to system level. Then we have made a simulation time comparison between two equivalent models of the NoC one with using high level structures specially resolution function in routing algorithm the other without using them. Often small changes to a handful of code lines can yield a large performance benefit. Resolution function can decrease simulation speed as in the literature mentioned so we have applied some improving techniques for simulation accelerating to see the result of these tradeoffs. All in all the one with resolution function and other high level structures besides applying improving speed rules has better performance and we have gain about 28% speed up and 35% speed up in contrast to the latter one without eliminating possible signals.

8. References

1. Paul J, Thomas D, Cassidy A. High-level modeling and simulation of single-chip programmable heterogeneous multiprocessors. ACM Transactions on Design Automation of Electronic Systems. 2005 Jul; 10(3):431–61.
2. Karim E. Modeling of a multi-core Micro-Blaze system RTL and TLM abstraction levels in SystemC. Master Project Nr. 2013 Mar; 3395:1–77.
3. Glass R. Reuse: What's wrong with this picture? IEEE Software. 1998 Mar; 15(2):57–9.
4. Skillicorn D, Talia D. Models and languages for parallel computing. ACM Computing Surveys. 1998 Jun; 30(2):123–69.
5. Haobo Y, Domer R, Gajski D. Embedded software generation from system level design languages. Design Automation Conference, Asia and South Pacific; 2004 Jan 27-30. p. 463–68.
6. Mahani N. Making Alive Register Transfer Level and Transaction Level Modeling in Ada. ACM SIGAda Adaletters. 2012 Aug; , 32(2):9–16.
7. Ecker W, Boettger J. Evaluation of Ada'95 and VHDL for system level modeling. Proceedings of the VIUF '97 Spring Meeting. VHDL the next 10 years; 1997. p. 15–29.
8. Calazans N, Moreno E, Hessel F, Rosa V, Moraes F, Carara E. From VHDL Register Transfer Level to SystemC transaction level modeling: A comparative case study. 16th Symposium on Integrated Circuits and Systems Design, IEEE; 2003 Sep 8-11. p. 355–60.
9. Douglas J. VHDL and Verilog compared and contrasted plus modeled example written in VHDL, Verilog and C. 33rd annual Design Automation Conference; Las Vegas.1996 Jun 3-7. p. 771–6.
10. Ussery C. VHDL is the Phoenix burning. VHDL the next 10 years. 1997. p. 29–35.
11. Pandharpurkar N, Ravi V. Design of BIST using self-checking circuits for multipliers, Indian Journal of Science and Technology. 2015 Aug; 8(19):1–7.
12. Ashenden PJ, Wilsey PA, Martin DE. SUAVE: Extending VHDL to improve data modeling support, IEEE Design and Test of Computers. 1998 Apr-Jun; 15(2):34–44.
13. Mills M, Peterson G. Hardware/Software Co-design: VHDL and Ada 95 code migration and integrated analysis. Annual ACM SIG Ada International Conference on Ada; Washington, D.C., United States. 1998. p. 18–27.
14. van den Hurk J, Semicond P, Dilling E. System level design, a VHDL based approach. European Design Automation Conference; Brighton. 1995 Sep 18-22. p. 568–73.
15. Cohen B. VHDL Answers to Frequently Asked Questions. Springer; US. 1997. pp. 222–37.
16. Elhaji M, Boulet P, Zitouni A, Meftali S, Dekeyser J. System level modeling methodology of NoC design from UML-MARTE to VHDL, design automation for embedded systems. Springer Verlag; Germany. 2012. p. 1–27.
17. Marzbanrad J, Soleimani G, Mahmoodi-k M, Rabiee A H. Development of fuzzy anti-roll bar controller for improving vehicle stability. Journal of Vibroengineering. 2015; 17(7):3856–64.
18. Sinthuja S, Kumar J, Manoharan N. Energy efficient voltage conversion range of multiple level shifter design in multi voltage domain, Indian Journal of Science and Technology. 2014 Oct; 7(S6):82–6.
19. Ecker W, Esen V, Schonberg L, Steininger T, Velten M, Hull M. Impact of description language, abstraction layer and value representation on simulation performance. Design, Automation and Test in Europe; Nice, France. 2007 Apr 16-20. p. 767–72.