International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075, Volume-9 Issue-2S, December 2019

Design Environment for Verilog Module Analysis using Open Source Tools

Uma R., Sarojadevi H., Sanju V.

Abstract*: *Network-on-Chip provides possible solutions for the limitations and challenges by the present day architectures for the interconnections. The characteristics of NoCs include energy efficiency, reliability, scalability, reusability and distributed routing decisions. The existence of today's semiconductor industry depends on shorter time-to-market, challenge of meeting increasing transistor density, reduced product life cycle, and operating frequencies getting higher. This paper discusses about a design environment for the analysis of Verilog NoC module. Tools such as Icarus Verilog, GTK Wave, Yosys etc. which are used for compilation, simulation and synthesis of the NoC are also discussed in this paper.

Keywords : Network-on-Chip, Semiconductor, Verilog, Simulation, Synthesis.

I. INTRODUCTION

In Network-on-Chip architecture, Moore's law is quite significant, and it gets extended by the introduction of Multiprocessor architectures. Network-on-Chip (NoC) architecture [1,2,3,4,5,6,7,15,17] resolves the problems in un-synchronized communications, errors in signal integrity and non-scalable wire delays. By the adoption of packet switched communication architecture the two problems reusability and scalability faced by SoC designers are well addressed. By adopting reuse at the network structure level and level of computation, the design time can be lowered, and as a result, time to market the new products can be reduced This paper describes about an Integrated Development Environment (IDE) for Hardware Description Language (HDL) programming, it's compilation, simulation and synthesis. The IDE can be used to develop the system software which could be deployed in Network-On-Chip on a reprogrammable [9,13,14,16,18,19,20,21] device. The system software developed will need to go through a test bench for simulation and analysis of the outcomes.

II. METHODOLOGY

The IDE integrates different open source tools for synthesis [8,10,11,12], compilation as well as simulation. Icarus software is used for compilation. The outcome of the Icarus compiler is then used directly in the GTK Waveform generator to view the simulated output. For Synthesis Yosys software is used, that will create the net list which can be diagrammatically represented in the IDE. This will help to easily reprogram and optimize the Verilog code for better performance. A design shall be developed using Hardware description language (Verilog). The HDL code is synthesized to be translated to logical primitives. This design can further be optimized to meet the requirements such as performance specifications, specified size, and timing constraints during synthesized process.

A. Icarus Verilog

Icarus Verilog [22] is an open source tool that helps in simulation. Also, it provides the features of synthesizing the module under test/design under test (DUT). It acts as a compiler which generates a VVP assembly form for simulating a full test bench for the module under test.

B. GTKWave Analyzer

GTKWave [23] Analyzer is used to view waveforms and is based on GIMP Toolkit. The VVP command generates a VCD file from the output of the Verilog compilation. This output VCD file can be viewed as waveforms in GTKWave Analyzer.

C. Yosys

Yosys [24] generates the Register Transfer Level (RTL) into logic gates-based design. Verilog only describes the transformation of the data as the information passes from Register to Register. This would require a synthesis tool which generates the net list, which would be representative for the logic gates.

The IDE developed takes HDL design input and has a provision of test bench simulation of the module-under-design. This enables viewing the HDL module-under-design to be viewed as netlist and helps simulating and analyzing time and performance factors as well.

III. SYNTHESIS

Synthesis phase is comprised of multiple steps. The synthesis process uses the Verilog code for the design under test (DUT). Using Yosys open source software, it synthesizes the DUT to generate the Netlist. The output netlist file is then converted into an image representation. This image is subsequently displayed on the Window, which provides the layout of the DUT file. The pseudocode for the synthesis process is as shown below.

```
void Synthesize ()
{
    // Read the Verilog code from
DUT text box control and
     // add contents  to a .v file
```

Revised Manuscript Received on December 12, 2019.

***** Correspondence Author
Uma R*, Dept. of CSE, Nitte Meenakshi Institute of Technology, Bangalore, India. Email: urmoorthy@gmail.com
Sarojadevi H., Dept. of CSE, Nitte Meenakshi Institute of Technology, Bangalore, India. Email: hsarojadevi@gmail.com
Sanju V. School of Computing &IT, REVA University, Bangalore, India. Email: sanjuv21@gmail.com

```
    WriteToFile (dutFile, tbVerilogDUTModule.Text)

     // Use Yosys and Perform
    SynthesisFormYosysSynthesisCommand
     (DUTInputFile, synthesizedOutputFile);

    // Convert Netlist to Image
    ExecuteSynthesis () ;

    //Display Image on Bitmap ImageControl
    ConvertNetlistToBitmapImageFile
       ([in]Netlist, [out]BimapFile);

    dotPictureBmp.Image = Image.FromFile (dotImageFile)
}
```

IV. COMPILATION

In order to check the correctness of the outcomes of a Verilog module being developed, it needs to be used in a test bench, which is also a Verilog module that is consuming the design under test. Using the Icarus compiler, the DUT module along with the test bench module shall be compiled. The compiled output is then used to perform the simulation activity. Simulation process produces a waveform output, which can then be viewed in the waveform analyzer tool. The waveform analyzer tool used in this tool is the GTKWave analyzer, that shall be launched subsequent to the compilation process.

```
    void Compile ()
    {
        // Read the Verilog code DUT text box control and
        // written to the File.
        WriteToFile(dutFile,tbVerilogDUTModule.Text)

        // Read the  Text Box control in Verilog Test Bench
        // add contents to the File.
        WriteToFile(testbenchFile, tbVerilogTestBenchModule
          .Text);

        // Do the compilation process.
        FormCompilationCommandUsingIcarus(dutFile,
         testbenchFile, compiledResult);
        // Execute the compilation command.
        Run Command (command, true);
}
```

V. SIMULATION

Simulation of the design involves the running the Simulation Runtime engine command and then launching the Waveform display.

```
    void Simulate ()
    {
        // Form the Simulation Runtime Engine command.
        FormSimulationRuntimeEngineCommand
        (compiledResult, outputVCDFile);
        // Execute the command.
        RunCommand(command);

        // Form the Simulation Runtime Engine command.
        FormWaveFormLauncherCommand(outputVCDFile);
        // Execute the command.
        RunCommand(command);
    }
```

VI. RESULT AND DISCUSSION

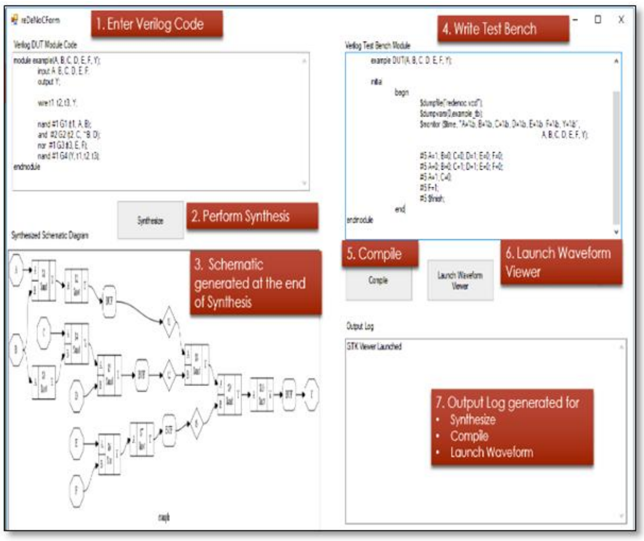
The IDE has the following Text editor options.

• Verilog Module - A module under development can be added in this text editor.
• Verilog Test Bench module - A test bench module for the module under development can be added in the Verilog test bench editor.

The IDE has 3 action controls.
• Synthesis - A netlist output is created using the synthesis option from the Yosys open software.
• Compile- Compiles the Verilog module and the test bench module.
• Launch Waveform Viewer - Executes the simulation and generates a VCD file, which will be used on the GTK Waveform Analyzer.

The IDE has 3 output display options.
• Synthesized Netlist display - Displays the output image.
• Compilation Logs - Displays the output logs of the compilation and synthesize operations.
• GTK Wave Analyzer - Output of the Verilog simulation in the VCD format will be displayed in the GTK Wave Analyzer.

A screenshot of the user interface with its user workflow in the right sequence is as shown in Fig. 1.



```
module example(A, B, C, D, E, F, Y);
        input A, B, C, D, E, F;
        output Y;

        wire t1, t2, t3, Y;

        nand #1 G1 (t1, A, B);
        and  #2 G2 (t2, C, ~B, D);
        nor  #1 G3 (t3, E, F);
        nand #1 G4 (Y, t1, t2, t3);
endmodule
```

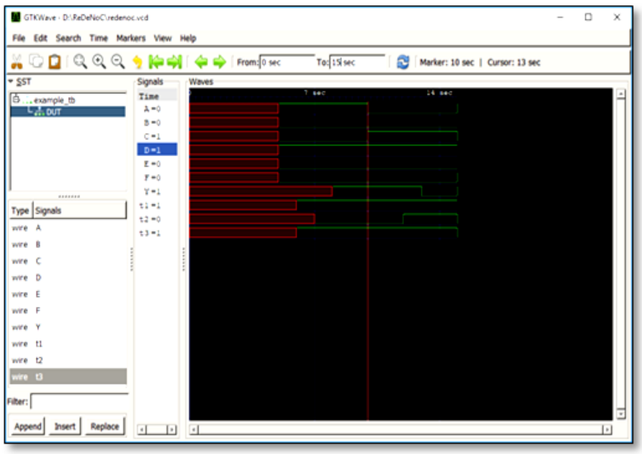
```
        example DUT(A, B, C, D, E, F, Y);

        initial
                begin
                        $dumpfile("redenoc.vcd");
                        $dumpvars(0,example_tb);
                        $monitor ($time, "A=%b, B=%b, C=%b, D=%b, E=%b, F=%b, Y=%b",
                                                        A, B, C, D, E, F, Y);

                        #5 A=1; B=0; C=0; D=1; E=0; F=0;
                        #5 A=0; B=0; C=1; D=1; E=0; F=0;
                        #5 A=1; C=0;
                        #5 F=1;
                        #5 $finish;
                end
endmodule
```

```
GTK Viewer Launched
```

Fig. 1. UI of the Integrated Development Environment

The output of the compilation process abstracts the compilation. The VVP command used for execution to simulate generates the VCD files. This VCD file is viewable in the GTKWaveform Analyzer.

The IDE provides a provision to launch the GTKWaveform viewer with the output generated. Once the registers or wires in the DUT are added to the signals which are to be monitored, then the user will be able to view the waveform for all the parameters in consideration. This waveform will help validate if the module under test is generating the right outcome. Fig. 2 shows the GTKWaveform output for full adder design.


GTKWave - D:\ReDeNoC\redenoc.vcd
File Edit Search Time Markers View Help
SST
Signals
Waves
7 sec
14 sec
Append
Insert
Replace
Filter:

From: 0 sec To: 15 sec | Marker: 10 sec | Cursor: 13 sec

example_tb
DUT

Time
A =0
B =0
C =1
D =1
E =0
F =0
Y =1
t1 =1
t2 =0
t3 =1

Type	Signals
wire	A
wire	B
wire	C
wire	D
wire	E
wire	F
wire	Y
wire	t1
wire	t2
wire	t3

Fig. 2. GTK Waveform Analyzer output

When the waveform analyzer is displayed, it launches with the full-time cycle for the simulation performed. The time view can be modified by adjusting the "From" and "To" time in the toolbar. Similarly, when the user wants to see the value of any time scale, the same can be viewed by placing the Marker at the desired location.

VII. CONCLUSION

NoC, in general, offers great performance and power advantages. A software for providing an environment for NoC design and reconfiguration can be of significant importance for faster and cost-efficient solutions. This paper discusses an Integrated Development Environment for use in NoC based Reprogrammable devices. The simulation and synthesis of the modules performed in this IDE helps in faster programming and diagnosis of the issues in the development of new modules.

ACKNOWLEDGMENT

We thank our Director, Principal, HOD and Staff of NMIT for their support in publishing this paper.

REFERENCES

1. William J. Dally and Brian Towles (2001) "Route Packets Not Wires on chip interconnection network", Proceedings of the 38th annual Design Automation Conference, pp 684-689.
2. Niranjan N Chiplunkar and Sanju V (2008) "Network-on-Chip: A Brief Review", Proceedings of International Conference on Emerging Techniques in Computing, Electronics, Embedded System & VLSIDesign.
3. Muhammad Ali, Michael Welzl and Martin Zwicknagl (2008) "Networks on Chip: Scalable Interconnects for Future Systems on Chip", 4th European Conference on Circuits and Systems for Communications.
4. RobbeVancayseele, Brahim Al Farisi, Wim Heirman, Karel Bruneel and Dirk Stroobandt (2011) "RecoNoC: A Reconfigurable Network-on-Chip", 6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC).
5. Salma Hesham, Jens Rettkowski, Diana Göhringer, Mohamed A and Abd El Ghany (2015) "Survey on Real-Time Network-on-Chip Architectures", SpringerInternationalPublishing Switzerland.
6. Sanju V., Koushika C, Sharmili R., Niranjan Chiplunkar and M. Khalid (2014) "Design and implementation of a network onchip-based simulator: a performance study", International Journal of Computational Science and Engineering, Vol. 9.
7. Ahmed Ben, Achballahm and Slim Ben Saoud (2013) "A Survey of Network-On-Chip Tools", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 4.
8. Razvan Nane, Jongsok Choi, Yu Ting Chen, and Fabrizio Ferrandi (2016) "A Survey and Evaluation of FPGA High-Level Synthesis Tools", IEEE Transactions on Computer-aided Design of Integrated Circuits And Systems, Vol. 35.
9. ScottHauckK (1998) "The Roles of FPGA's in Reprogrammable Systems", Proceedings of the IEEE, Vol 86.
10. Razvan Nane, Vlad-Mihai Sima and Christian Pilato (2016) "A Survey and Evaluation of FPGA High-Level Synthesis Tools", IEEE Transactions on computer-aided design of integrated circuits and systems, vol. 35.
11. C. Iseli and E. Sanchez (2002) "A C++ compiler for FPGA custom execution units synthesis", Proceedings IEEE Symposium on FPGAs for Custom Computing Machines.
12. L. Daoud, Dawid Maksymilian Zydek and Henry Selvaraj (2014) "A Survey of High-Level Synthesis Languages, Tools, and Compilers for Reconfigurable High-Performance Computing", Proceedings of the 18th International Conference on Systems Science (IC, Advances in Intelligent Systems and Computing, Vol 240, pp 483-492.
13. Mohamed S. Abdelfattah and Vaughn Betz (2015) "Power Analysis of Embedded NoCs on FPGAs and Comparison with Custom Buses", IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, Vol 24, pp 165-177.
14. Saourbh Singh Mehra, Rajni Kalsen and Rajeev Sharma (2010) "FPGA based Network-on-Chip Designing Aspects", National Conference on Advanced Computing and Communication Technology.
15. S. Kumar, A. Jantsch and J.-P. Soininen (2002) "A Network on Chip Architecture and Design Methodology", Proceedings IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design.
16. Jan Frigo, Maya Gokhale and Dominique Lavenier (2001) "Evaluation of the Streams-C C-to-FPGA Compiler: An Applications Perspective", ACM/SIGDA International symposium on FPGA.
17. T.A. Bartic, J.Y. Mignolet and V. Nollet (2005) "Topology adaptive network-on-chip design and implementation", IEE Proceedings - Computers and Digital Techniques, Vol 152.
18. Mohamed S. Abdelfattah, Andrew Bitar and Vaughn Betz (2015) "Take the Highway: Design for Embedded NoCs on FPGAs", Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp 98-107.
19. Hadi Mardani Kamali and Shahin Hessabi (2016) "AdapNoC: A Fast and Flexible FPGA-based NoC Simulator", 26th IEEE International Conference on Field Programmable Logic and Applications (FPL).
20. Hadi Mardani Kamali, Kimia Zamiri Azar and Shaahin Hessabi (2018) "DuCNoC: A High-Throughput FPGA-based NoC simulator using Dual-Clock Lightweight Router Micro-Architecture", IEEE Transactions on Computers, Vol 62, pp 208-222.
21. Ke Pang, Virginie Fresse, Suying Yao Otavio and Alcantara De Lima (2015) "Task mapping and mesh topology exploration for an FPGA-based network on chip", Journal Microprocessors & Microsystems, Vol 39, pp 189-199.
22. Icarus Verilog. http://iverilog.icarus.com/home
23. GTKWave. http://gtkwave.sourceforge.net/
24. David Shahy, Eddie Hungz, Clifford Wolf, Serge Bazanski, Dan Gisselquist and Miodrag Milanovi, "Yosys+nextpnr: An Open Source Framework from Verilog to Bitstream for Commercial FPGAs", to appear at IEEE FCCM 2019.

AUTHORS PROFILE

Mrs. Uma R. is pursuing Ph.D. from Visvesvaraya Technological University, Belgaum. She obtained her M.Tech. in Computer Science and Engineering from Nitte Meenakshi Institute of Technology, Bangalore in 2011 and B.Tech. in Computer Science and Engineering from Cochin University of Science and Technology in 2005. She is working as Assistant Professor at NMIT since February 2008. Her area of interest includes Network-on-Chip architectures, Compiler design and Web technologies.

Dr. Sarojadevi is a Ph.D. graduate from the Indian Institute of Science, Bangalore. She has been providing service as teacher/researcher since the time she received her BE & ME qualifications from the University Visvesvaraya College of Engineering, Bangalore. She has published more than 100 research papers in recognized journals/conferences and guided six people for Ph.D. She has about 4 years of industry experience in Siemens Information Systems Ltd., Bangalore. Dr. Sarojadevi is recognized by Marquis Who Is Who in the World in 2016 with Lifetime achievement award and she also won several other awards including CSI India Technext award in 2018 as Best Faculty of the Year. Her main research areas include High performance computing & computer architecture, Cloud computing, Machine learning and Deep learning. She is a life member of the professional bodies like CSI, IISCAA & ISTE.

Dr. Sanju V. obtained his BE in Computer Science and Engineering from NMAMIT, Nitte, India in 2002, M.Tech. in VLSI and Embedded Systems from Department of PG Studies, University Campus, VTU, Belgaum, India in 2005 and his Ph.D. from VIT University, Vellore in 2014, India. His area of interest includes Network-on-Chip architectures, CAD for VLSI, and Embedded systems.