Pre-Eminance of Open Source Eda Tools and Its Types in The Arena of Commercial Electronics

Geeta Yadav
VLSI Design Group
Department of EECE, ITM University
Gurgaon (Haryana), India

Neeraj Kr. Shukla
VLSI Design Group
Department of EECE, ITM University
Gurgaon (Haryana), India

***Abstract*—Digital synthesis with a goal of chip designing in the commercial electronics arena is packed into large EDA Software providers like, Synopsys, Cadence, or MentorGraphics. These commercial tools being expensive and having closed file structures. It is also a financial constraint for the startup companies sometimes who have their budget limitations. Any bug-fixes or add features cannot be made with ease; in such scenario the company is forced to opt for an alternative cost effective EDA software. This paper deals with the advantages of using open source EDA tools like Icarus Verilog, Verilator, GTKwave viewer, GHDL VHDL simulator, gEDA, etc. that are available as a free source and focuses on the Icarus Verilog simulator tool. It can be seen as a big encouragement for startups in Semiconductor domain. Thereby, these open source EDA tools make the design process more cost-effective, less time consuming and affordable as well.**

Keywords—Open source EDA; MentorGraphics; Cadence; Icarus Verilog; GTKwave viewer; Verilator; GHDL VHDL simulator; gEDA; Linux; Github; VPI

I. INTRODUCTION

A. *Open source EDA tools*

The open source plays a key role in the EDA tool development. A set of tools known as Digital synthesis flow is used to turn a circuit design written in Verilog or VHDL a high-level behavioral language into a physical circuit, which can either serve to be configuration code for a Xilinx or Altera chip, or a layout in a specific fabrication process technology [1]. In the commercial electronics arena, digital synthesis with the application of a chip design is usually packaged into large EDA software systems like MentorGraphics or Cadence or Synopsys which are very expensive. So the designers need to maintain cutting-edge performance as these commercial tool chains get more and more expensive. Another disadvantage of working with the closed file structures is that it becomes difficult to add customized features to the tool. Also, for small customers it becomes very difficult and time consuming for them to fix the bugs appearing in the tool. An alternate to this, is to go for new software that is more user friendly and easier to deal with [2].

The oldest of these are probably VIS and SIS, two software tools developed at Berkeley for Verilog parsing, logic verification, and mapping of logic onto a digital standard cell library. They perform the task of logic optimization and cell mapping admirably [3].

B. *Linux Platform*

Linux serves as a open source platform for the various EDA tools. It is very advantageous as compared to the other operating systems as it is free to obtain, while Microsoft products are available for a hefty and sometimes recurring fee, the security aspect of Linux is much stronger than that of Windows as free from virus, the power to control just about every aspect of the operating system, flexibility, its use as a firewall, a file server, or a backup server [4].

C. *Github*

It is a web based hosting service which is used for software development projects using the Git version control system. Git is a software version control tool which is mainly used for proper management of the various versions of a particular tool. It keeps track of the dates when the changes are being made to the tool, what are those changes, and who has made those changes. This way, we can easily follow the continuous changes being made to the tool. For private repositories, Github provides paid plan whereas for open source projects it is free. The site provides social networking functionality such as feeds, followers and the social network graph to display how developers work on their versions of a repository.

II. TYPES OF OPEN SOURCE EDA TOOLS

A. *Icarus Verilog Simulator*

Icarus verilog is an open source synthesis and simulation tool. It works as a compiler and compiles the source code written in Verilog(IEEE-1364) into a target format. The Icarus Verilog compiler is written by Stephen Williams. He is still working on it. This tool supports a waveform viewer named GTKWave. This tool has various released versions; one of its latest released versions is version 0.9.6 [5].

Characteristics:

- Simulation engine is efficient
- Portable compiler
- Challenge for commercial tools
- Supported graphics tool like GTKwave
- New compatibility with de facto standards such as library formats and command files

Significance:

The addition of the functionality to the Icarus Verilog tool would lead to the effective and reduction in the design and verification of the circuits through the Hardware Description Languages. List of providers who offer commercial support for Icarus Verilog and/or related products are Dolly Software Private Limited, Embecosm, OCLogic Limited [6].

The tool has various release versions like 0.9.2, 0.9.3, 0.9.4, and so on, latest being the version 0.9.6. The release notes of the tool lists the various bugs in that version which are then worked on. This can be fixed by any user as it is an open source EDA detail. The fixation of these bugs leads to making the verification of the tool more effective.

B. *Verilator*

Verilator is a free Verilog HDL simulator. It compiles synthesizable Verilog into an executable format and wraps it into a SystemC model. Internally a two-stage model is used. The resulting model executes about 10 times faster than standalone SystemC. Verilator has been used to simulate many very large multi-million gate designs with thousands of modules. Therefore we have chosen this tool to be used in the verification environment for the Open RISC processor [7]. *Characteristics:*

- Verilator is the fastest
- It compiles synthesizable Verilog (not test-bench code), plus some PSL, SystemVerilog and Synthesis assertions into C++ or SystemC code
- Verilator has been used to simulate many very large multi-million gate designs with thousands of modules

C. *GHDL VHDL simulator*

GHDL implements the VHDL87 (common name for IEEE 1076-1987) standard, the VHDL93 standard (aka IEEE 1076-1993) and the protected types of VHDL00 (aka IEEE 1076a or IEEE 1076-2000). The VHDL version can be selected with a command line option [7].

Characteristics:

- It has been successfully employed for compiling and simulating the DLX processor and the LEON1 SPARC processor
- It directly creates binaries or executable images, which is the best form for testbenches
- It can be used to pretty print or to generate cross references in HTML

D. *gEDA*

The gEDA project was started by Ales Hvezda in an effort to remedy the lack of free software EDA tools for Linux/Unix. The first software was released on 1 April 1998, and included a schematic capture program and a netlister. At that time, the gEDA project website and mailing lists were also set up.

Characteristics:

- Originally, the project planned to also write a PCB layout program
- The ability to target netlists to PCB was quickly built into the gEDA Project's netlister, and plans to write a new layout program from scratch were scrapped
- The authors of other open source programs became affiliated with the gEDA website and mailing lists, and the collaborative gEDA Project was born

E. *GTKwave viewer*

It is a GTK+ based wave viewer for Unix, Win32 etc. GTK+ is basically a toolkit used for creating graphical user interfaces. It helps in viewing the VCD files. The Icarus Verilog tool uses the GTKwave tool for the graphical representation of the results.

III. VERILOG PROCEDURAL INTERFACE

VPI stands for Verilog Procedural Interface. The use of VPI permits behavioral Verilog code to invoke C functions and C functions to invoke standard Verilog system tasks. VPI is a part of the programming language interface standard IEEE 1364. Users can access information contained in a Verilog design as well as provides facilities to interact dynamically with a software product via the VPI interface routines. VPI interface provides applications such as annotators and connection of a Verilog simulator with other simulation tools and customize the debugging of tasks, delay calculators. The basic functions of the VPI interface can be categorized into two main areas: VPI callbacks usage for dynamic software product interaction. Verilog HDL objects and simulation specific objects can be accessed.

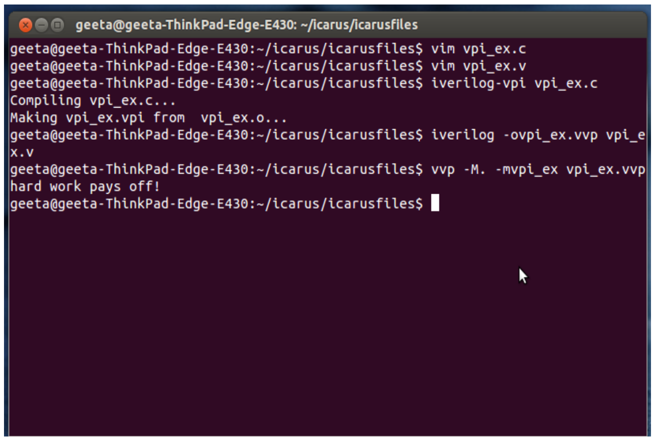
Callbacks in VPI can call a user-defined application by the use of Verilog HDL software product like logic simulator when a specified activity occurs on request of the user. Dynamic software product interaction is accomplished by registered callback mechanism. To understand this take an example like, when a particular net value changes the user can request the calling of the user application my_monitor and can call my_cleanup() routine when the execution of the product of the software completes. The detection of the changes in the occurrence of values, termination of simulation, time advancement, etc. can be detected by using VPI callback facilities which allows the dynamic interaction with software product. The callback feature allows applications of integration with other simulation systems, specializing of the timing checks, complexity of debugging features, etc. There are four basic reasons for callbacks as listed below [9]:

- Simulation event (e.g., change in the value on a net or a behavioral statement execution)
- Simulation time (e.g., the termination of a time queue or after certain amount of time)
- Simulator action/feature (e.g., the end of compilation, end of simulation, restart, or enter interactive mode)
- User-defined system task or function execution

Steps required writing a C function and interfacing it with a Verilog simulator:

- Write a function in C

- Associate the C function with a new system task
- Register a new system task
- Invoke system tasks



```
geeta@geeta-ThinkPad-Edge-E430: ~/icarus/icarusfiles
geeta@geeta-ThinkPad-Edge-E430:~/icarus/icarusfiles$ vim vpi_ex.c
geeta@geeta-ThinkPad-Edge-E430:~/icarus/icarusfiles$ vim vpi_ex.v
geeta@geeta-ThinkPad-Edge-E430:~/icarus/icarusfiles$ iverilog-vpi vpi_ex.c
Compiling vpi_ex.c...
Making vpi_ex.vpi from  vpi_ex.o...
geeta@geeta-ThinkPad-Edge-E430:~/icarus/icarusfiles$ iverilog -ovpi_ex.vvp vpi_e
x.v
geeta@geeta-ThinkPad-Edge-E430:~/icarus/icarusfiles$ vvp -M. -mvpi_ex vpi_ex.vvp
hard work pays off!
geeta@geeta-ThinkPad-Edge-E430:~/icarus/icarusfiles$ 
```

Fig. 1. Example showing use of VPI

IV. Event Queue Standard

The Verilog HDL is defined in terms of a discrete event execution model. A design has a large number of threads connections for execution or processes. Objects that can have state can be evaluated and respond to the changes in the outputs with respect to the changes in the inputs are called Processes. Modules, primitives, initial and always procedural blocks, continuous assignments, asynchronous tasks, and procedural assignment statements are all included in a process.

Update event: Change in value of a every net or variable in the circuit being simulated and named event as well.

Evaluation event: Processes have sensitivity for update events. All the processes that are sensitive to that event are evaluated in an arbitrary order when an update event is executed. Process evaluation is also an event.

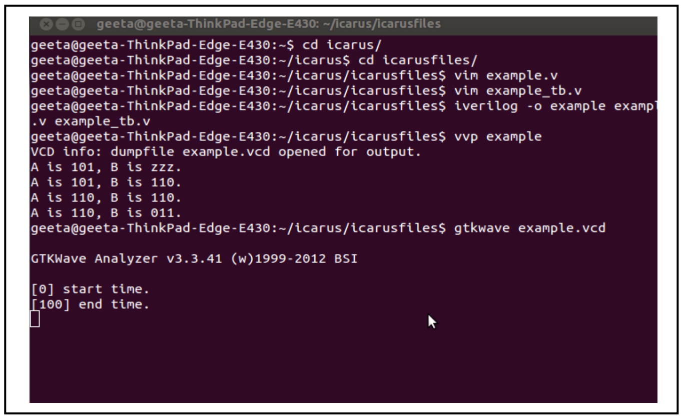
Simulation time: It is the time value maintained by the simulator for modeling the actual time it would take for the circuit in simulation.

Scheduling an event: Events occur at different times slots, so in order to keep track of the events and to have a surety that they are processed in the correct order, the events are kept on an event queue, as ordered by simulation time. Putting an event on the queue is called scheduling an event.

V. VPI For Inertial Delay In Icarus Verilog Version 0.9.6

This is one of the functional bugs in the 0.9.6 released version. VPI (Verilog Procedural Interface) is used to invoke the C functions from Verilog or vice-versa. Inertial delay is basically the gate delay associated with the design. Icarus Version 0.9.6 does not provide the VPI support for inertial delays i.e. it does not consider the inertial delays associated with the design when interfacing with the other tools.

For the addition of this functionality changes need to be made in the source code. For this bug the scheduler.cc is the part of the source code where changes are needed. An event queue is used for scheduling the active, inactive, and non-blocking and monitors assignments. Hence for inertial delay a second event queue needs to be added providing input to the event loop in the scheduler.cc which will automatically keep a check on the inertial delay of the design. The adding of this functionality leads to the more efficient design verification as the inertial delay will also be encountered.



```
geeta@geeta-ThinkPad-Edge-E430: ~/icarus/icarusfiles
geeta@geeta-ThinkPad-Edge-E430:~$ cd icarus/
geeta@geeta-ThinkPad-Edge-E430:~/icarus$ cd icarusfiles/
geeta@geeta-ThinkPad-Edge-E430:~/icarus/icarusfiles$ vim example.v
geeta@geeta-ThinkPad-Edge-E430:~/icarus/icarusfiles$ vim example_tb.v
geeta@geeta-ThinkPad-Edge-E430:~/icarus/icarusfiles$ iverilog -o example exampl
.v example_tb.v
geeta@geeta-ThinkPad-Edge-E430:~/icarus/icarusfiles$ vvp example
VCD info: dumpfile example.vcd opened for output.
A is 101, B is zzz.
A is 101, B is 110.
A is 110, B is 110.
A is 110, B is 011.
geeta@geeta-ThinkPad-Edge-E430:~/icarus/icarusfiles$ gtkwave example.vcd

GTKWave Analyzer v3.3.41 (w)1999-2012 BSI

[0] start time.
[100] end time.
```

Fig. 2. Example showing inertial delay support without VPI

The inertial delay support is present in the Icarus Verilog version 0.9.6 without the use of VPI, whereas when the VPI is used than the inertial delay is not supported by the tool. Hence this is one of the bug present in the version 0.9.6.

VI. Conclusion

The closed file structures are a real challenge for the startup companies as they are expensive and the bug-fixes as well as the adding of functionality is not possible, in such scenario the company needs to change the whole software. On the other hand open source EDA tools are of great importance, it provides its source code for the users to make changes and use it.

The availability of the source code leads to the faster development of the tool. Icarus Verilog is a very strong simulation tool; the synthesis part is being worked on. Icarus Veilog also has a graphical support in the form of GTKwave viewer. So, open source EDA tools are cost-effective, less time consuming, user friendly with a lot of fun learning as well.

References

[1] URL: http://opencircuitdesign.com/qflow/welcome.html
[2] URL: http://blog.engineersimplicity.com/2008/11/open-source-eda.html
[3] URL: http://opencircuitdesign.com/qflow/welcome.html
[4] URL:http://ubuntu-artists.deviantart.com/journal/8-Advantages-of-using-Linux-overWindows-291681914
[5] URL:]http://iverilog.icarus.com/
[6] URL: https://sites.google.com/site/iverilog/support/support-providers
[7] URL: http://opencores.org/opencores,tools