VERILOK SOFTWARE FOR PROGRAMMABLE DIGITAL DEVICES

Ruzieva Sokhibjamol Nuralievna
Sotvoldiyeva Sabrina Burxon qizi
Khalikova Madina Shukhratovna

Annotation: *Verilog is a digital circuit description language. In the first lesson, we will get acquainted with the main types of signal sources used in the language. Perhaps it would be a good idea to start our conversation with the concept of a signal. Signals are electrical impulses transmitted by wires between the logical elements of a chain. Wires carry information without any calculations. In a digital loop, signals are important for the transmission of binary data. In Verilog, the main type of signal source is wire. So, if you have an arithmetic or logical expression, you can associate the result of the expression with the named string and then use it in other expressions. This is somewhat similar to variables except that they (like the wires in the circuit) cannot be reconnected quickly and their assignment cannot be changed. The meaning of the wire is a function of what is connected to it. This article presents information about verilok software, virelok software on programmable digital devices and the future of this digital system.*

Keywords: *Verilog, software, FPGA, VHDL, C++, SystemC, Chisel*

INTRODUCTION

Verilog is a hardware definition language (HDL) used to describe digital systems and circuits in code form. It was developed by Gateway Design Automation in the mid-1980s and later acquired by Cadence Design Systems. Verilog is widely used to design and inspect digital and mixed signal systems, including Application-Specific Integrated Circuits (ASICS) and field-programmed gate arrays (FPGAs). It supports a number of levels of abstraction, from structural to behavioral, and is used for simulation-based design and synthesis-based design. The language is used to describe digital circuits in a hierarchical way, starting with the most basic elements such as logic gates and flip-flops and building more complex functional blocks and systems. It also supports a number of modeling techniques, including gate level, RTL level, and behavior level modeling.

Prior to the development of Verilog, the primary hardware description language (HDL) for digital circuit design and verification was VHDL (VHSIC Hardware Description Language). VHDL was developed in the 1980s by the United States Department of Defense as part of the very high-speed integrated circuit (VHSIC) program for the design and testing of high-speed digital circuits. VHDL is a complex language that allows designers to describe digital systems using levels of abstraction ranging from low-level transistors and door levels to complex hierarchical systems. It was designed to be more descriptive and flexible than previous HDLs such as ABEL (extended logic expression language), ISP (integrated system synthesis procedure),

and CUPL (compiler for Universal programmable logic). Despite the development of Verilog and its growing popularity since the 1980s, VHDL remains a widely used HDL, particularly in Europe, in the military and Aerospace Industries. Today, both Verilog and VHDL are widely used in the design and verification of digital circuits, with many companies and organizations using a combination of the two languages.

Verilog introduced several significant improvements over its predecessor languages, which helped make it a more popular and efficient HDL for designing and verifying digital circuits. Several reasons why Verilog is considered better than its predecessor HDLs:

• Simpler syntax: Verilog has simpler syntax compared to VHDL, allowing designers to write code faster and with fewer errors.

• Better support for behavioral modeling: Verilog provides better support for characterizing the behavior and functionality of digital designs. It supports a number of modeling techniques ranging from gate level to behavioral level modeling, which facilitates the description of the behavior of complex digital circuits.

• High level of abstraction: Verilog provides a higher level of abstraction than its predecessors. This allows designers to describe digital circuits using concepts such as modules and ports, making the design process more efficient.

•Better tool support: due to its growing popularity, Verilog supports better tools than its predecessors. Verilog has a number of integrated development environments (IDEs) and simulation tools that facilitate the design and verification of digital circuits.

Verilog creates a level of abstraction, which helps to hide the details of its implementation and technology.

For example, the D flip-flop design requires knowing how to place transistors to achieve positive-edge triggered FF, and how much rise, fall, and clk-Q times are needed to secure the value to the flop. among many other technology-oriented details. Power dissipation, time and the ability to control the network and other flops also require a deeper understanding of the physical properties of the transistor.

Verilog helps us focus on behavior and leaves the rest to be put in order later.

An example of a Verilog code:

The following Verilog code describes the movement of the counter. If the up_down signal is 1, the counter will count up, and if its value is 0, it will go down. If the rstn signal becomes 0, it resets the counter and converts it to an active-low reset.

```
module ctr (input              up_down,
                               clk,
                               rstn,
            output reg [2:0]   out);

    always @ (posedge clk)
        if (!rstn)
            out <= 0;
        else begin
            if (up_down)
                out <= out + 1;
            else
                out <= out - 1;
        end
endmodule
```

The simple example shown above shows how all physical implementation details (the interconnection of basic logic gates such as NAND and NOR) are hidden and at the same time have a clear idea of how the counter works.

the ctr represents an up/down counter module and a real physical execution of the design can be selected from flops of different styles optimized for Area, Power and performance. They are usually concentrated in libraries and will be available for us to select EDA tools at the next stage of the design process.

It is difficult to predict what will be the place of Verilog in the future, but there are several emerging technologies and languages that can affect the future of the design and verification of a digital system.

One technology that can influence the future of digital system design is high-level synthesis (HLS), a method of automatically generating hardware designs from high-level descriptions in languages such as C, C++, and SystemC. HLS allows designers to express their design goals and functionality at a higher level of abstraction, rather than displaying logical door details and registering transfers in Verilog or VHDL. This allows for more efficient and fast design of digital systems and allows designers to explore more design space in a short time.

Another technology that can influence the future of digital system design is machine learning and artificial intelligence (Al), which is able to significantly simplify the process of design and verification of digital systems. For example, machine learning algorithms can be used to automatically optimize and create hardware designs that reduce the need for manual design actions.

In addition, there are new ones that are trying to overcome some of the limitations of Verilog and VHDL, such as Chisel and MyHDL

there are emerging HDLs that are based on more modern programming concepts and provide a higher level of abstraction.

REFERENCES:

1. Cabrera, A.M., Young, A.R., Vetter, J.S.: Design and analysis of cxl performance models for tightly-coupled heterogeneous computing. In: Proceedings of the 1st International Workshop on Extreme Heterogeneity Solutions, ExHET ’22. Association for Computing Machinery, New York, NY, USA (2022). DOI 10.1145/3529336.3530817. URL https://doi.org/10.1145/3529336.3530817
2. Chacko, J., Sahin, C., Nguyen, D., Pfeil, D., Kandasamy, N., Dandekar, K.: Fpga-based latency-insensitive ofdm pipeline for wireless research. In: 2014 IEEE high performance extreme computing conference (HPEC), pp. 1–6. IEEE (2014)
3. Introduction to FPGA Design with Vivado High-Level Synthesis UG998 (v1.0) July 2, 2013
4. Bollinger, JG va N. A. Duffie, kompyuterni boshqarish Mashinalar va jarayonlar, Addison Uesli, Nyu-York, 1988 yil.
5. Astrom, K. J., M. Blanke, A. Isidori, V. Shaufelberger va R. Sanz (tahr.), Murakkab tizimlarda boshqarish, Springer Verlag, Nyu-York, 2000.
6. Holt, Dr, Integratsiyalashgan Ishlab Chiqarish Muhandislik Tizimlari, Mcgrou-Hill Inc., Nyu York, 1992.