Buletin Teknik Elektro dan Informatika
(Bulletin of Electrical Engineering and Informatics)
Vol. 3, No. 1, March 2014, pp. 51~54
ISSN: 2089-3191

A Hybrid Hardware Verification Technique in FPGA Design

Mojtaba.Dehghani Firouzabadi*[1], Hossein Heidari[2]
Azad University of Meybod/Departement of Computer Engineering, Yazd, Iran
*Corresponding author, email: Mojtaba_f84@yahoo.com[1], hosseinhdh@yahoo.com[2]
Hp: 09130718835[1], 09357419120[2]

Abstract

Assertion-based verification (ABV) is best emerging technique for verification of industrial hardware. Property Specification Language (PSL) is one of the most important components of ABV. In this paper we present a method to emulate hardware that is capable of support ABV that in it assertion expressions mapped to HDL. We simulated this method by an applicable example by Modelsim software. Test results indicate that this method performance is good.

Keywords: Verification, Assertion Circuit, Assertion, Hardware Emulation

1. Introduction

The everyday increasing of complexity of integrated circuits needs a powerful verification method to be a key for increased productivity and design quality as well as a shorter time to market. A verification that properly implemented circuit regarding its expected behavior is a complex task. ABV (Assertion Based Verification) is a modern verification paradigm that is proper like formal and simulation based methods. Assertion is a verification directive that needs to consider a specific property of verification scheme. Property Specification Language (PSL) is rapidly emerging as a key of Hardware verification language (HVL) [2,3].

PSL aims to provide a tool to formally obtain attribute in higher level than of abstraction that obtains by standard hardware description language (HDL). PSL with its formally defined rules allows designers and verification engineers to model module proper behavior in clear and unambiguous style. Then these properties used in verification tool in verification process. Verification could be performed by formal methods. Because of the state explosion problem, formal methods are worse than simulation methods. One of the common verification methods is to simulate a large collection of test vectors or test tables to ensure correct behavior. When the simulation time is too large designers often turn to emulate using programmable logic devices such as FPGA.

ABV is obviously introduced significant overhead for simulation. For a large circuit, beside the circuit simulation, large number of fairly complicated assertions should be considered. For this reason, the use of hardware emulation in ABV is particularly attractive. Instead of adding assertions in series with the simulator, they can be placed in parallel with the circuit for verification. For inclusion of assertions in the emulation process, the PSL statements must be converted to effective HDL code to properly align with the verification plan.

Figure 1 shows an example that a logical assertion belonging to input or output signals of a counter with an additional circuit has been detected. Without reviewing assertion, a design error in verification scheme can affect the very low blocks and a number of clock cycles. Assertions, in addition to improving visibility, provide recognition of errors exactly in appearing place so error exclusion process will be significantly easier.[1]

PSL assertions could be found in simulators such as Modelsim of Mentor Graphics Corporation. In this paper we present circuits for use in emulation. Section 2 is devoted to an overview of the PSL. In section 3 Mapping of PSL assertions to desired HDL are presented. Section 4 discusses how to use assertion circuits in emulation hardware and shows a practical example of the hardware assertions. Test results are provided in Section 5.

Received November 30, 2013; Revised January 8, 2014; Accepted February 6, 2014

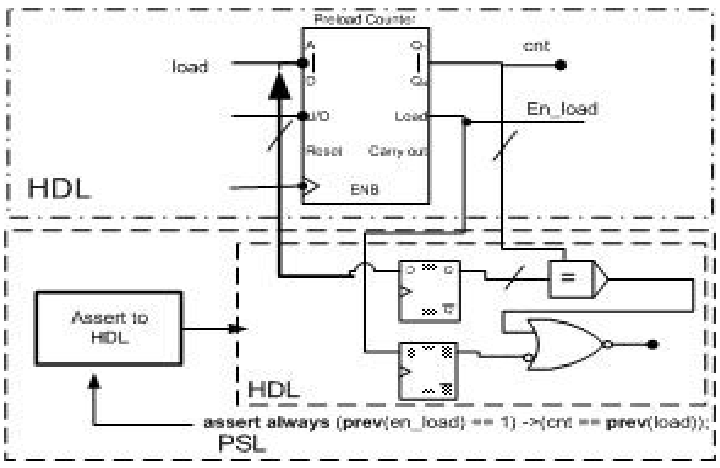

Figure 1. Assertion circuit

2. Property Specification Language (PSL)

Property shading Language (PSL) allows designers to obtain design properties and characteristics and use them in verification process. PSL allows that properties could be used with either formal verification tool (Static verification) or simulation tool (Dynamic verification). In formal verification specification there is no need for stimulation and the properties are evaluated mechanically. For example, the properties with branch time logic are clearly not simulated, so a simple subset of PSL used instead of that.

In simple subsets, operators such as "*∀ paths*" and "*∃ path*" are not permitted. So only one path is exist to next state that is applied by test table or environment. But, anyway, formal methods have many applications even in the field of SOC (Single on chip).[11]

In this paper, the intended application is from assertions to emulate the hardware, so providing and using PSL in subset frame is simple. PSL, to be widely applicable, have 4 flavor: VHDL, Veriloge, System Veriloge and GDL.[2]

For simplification, here we have presented a set of PSL structure. Clock of all of PSL statements will be default time with the following statement: Default clock=(posedge clock_sig);

In the PSL, Boolean layer composed from HDL. So the Boolean implication (->), equivalence (<->) and built-in functions have been added to it. Prefabricated functions include:

Prev (sig) and Prev (sig, N): turns previous value of a signal sig one or N pulse compared to current cycle.

Stable (sig), rose (bit), fell (bit): behavior of a bit is compared with its previous value and returns a Boolean value.

While many of the assertions can be made using the Boolean layer, the expressive power of PSL comes from temporal layer. The main structure in layer is the sequence. Sequence is a regular expression that is a list of Boolean expressions or other sequences that is encountered in successive clock cycles. Sequence is enclosed in {} and its list separator is semicolon (;). The comma (,) can be used to combine sequences with each other so that the last cycle of left sequence overlaps with the first cycle of right sequence.

Repetition operator [* N] is used to construct a sequence and for Boolean or other sequence expression.

A property is constructed by Boolean and temporal expressions. For more time control, simple below temporal operations can be used:

Always X: X feature should always be true

Never X: X feature should never be true.

Next X: X feature should be true in the next cycle.

Eventually! X: X feature should eventually be true.

In general, verification method is of writing features that should be considered by the verification scheme. Assert expression is for reviewing data property. In this paper assertion used for description of verification expressions that is appeared in the PSL form so they converted to HDL form. An assertion signal is output of assertion circuit that will be shown during the emulation. If the process is completed and no assertion signals activated then everything works fine. We should know that verification is powerful enough only in extent of assertions.

3. Producing Aassertion circuits

Incorporation of assertions for emulating hardware included to converting the PSL expressions into a hardware description language for verifying scheme.

3.1. Convert Boolean layer

As previously mentioned, Boolean layer expressions, including expressions of HDL background, equivalent and implicant and built-in expressions. The first two terms are easily converted: HDL expressions will be out directly, the equivalence operator a <-> b converted to (a & b) | (~ a & ~ b). Implicant operator (a -> b) becomes as follows: the left-hand side (LHS) converted to if and right-hand side (RHS) converted to must mode. LHS result is condition of RHS.

3.2. Temporal layer convert

As mentioned in Section 2, the majority of the statements in the temporal layer, use time operators, sequences and implication suffix. This subsection shows how this structure converted to the circuit form.

3.2.1. Conversion of temporal operators

Always keyword is a flip flop (FF) that will be reset to zero and it will be 1 only when condition signals are true. This FF signal condition is known as extension condition signal. After became 1, FF can only be reset by the abort operator. If the condition for starting is signal and abort does not exist no FF can be made and the 1 logic is passed instead of that argument. Never keyword converted as always keyword but complementary of signal returned to argument. Next temporal operator made by passing from the delay in condition signal.

4. Assertions in emulation and simulation

When PSL assertions needed to simulate there is a selection for using PSL supporter simulator. For many reasons, not always possible to use this simulator and other tools should be considered. In these cases, the simulation can be made, and indirectly using verification expression converted to HDL could support PSL.

For large circuits that are not practical for the simulation, circuit emulation made by FPGA and before converted to silicon it becomes widely tested. With the proposed method mentioned in this paper asserted hardware could easily converted to emulated circuit of HDL synthesis code. Because the assertion circuit entered to implementation process, all optimization related to the synthesis of assertion hardware are applied. Because asserted circuits are part of scheme, asserting signals should be supervised when executing outside. These issues informed the verification engineer, parts of the plan that will not meet the desired specifications.

Displaying assertion in emulation can be done as: with routing asserted signals to unused pins, with routing asserted signals to common registers used as interface between the host and the emulator, and with using chain scan techniques for reading the state of the circuit along its assertions when an error is detected.

4.1 Up/Down counter example

As an applicable example, one Ascending - Descending counter will be tested. The Table 1 shows two features for authenticity of verification scheme.

Features 1 and 2 have been implemented to ensure the proper operation of the counter and show that with executing Figure 2 assertions and HDL file of Figure 1, asserted signals released to proper end in design hierarchy to monitor.

Table 1. Assertions

N	Assertion
1	assert always (prev(en ud) == 0 && prev(en load) == 0) -> (stable(cnt));
2	assert always (prev(en load) == 1) -> (cnt == prev(load));

5. Experiment Results

Verification results of previous section example showed in Figure 2 with emulation method. Above features are not verifiable by Modelsim and only simple Boolean expressions could be verified by this simulator. But with hardware emulation method assertion could be verified and the asserted output shows with q_assert.

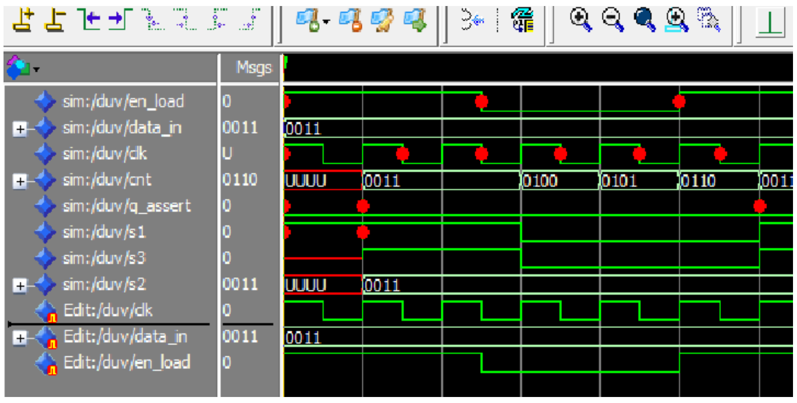

Figure2. Experiment results

It can be seen that in emulation method verification also have been done good, and also in emulation generated codes is easier to read.

5. Conclusion

This methodology introduces improvements of emulation for PSL code and allows assertions transform to the emulation hardware that called assertion circuits. Assertion circuit outputs could be used as triggers or tracing signals and also as a error monitor to find bugs related to mapping or implementation process. In the future , in this method, we can use other HVLs , such as OVA, OVL,…

References

[1] Y Abarbanel, I Beer, L Glushovsky, S Keidar, Y Wolfsthal*. FoCs: Automatic Generation of Simulation Checkers from Formal Specifications. Conference on Computer Aided Verification*. 2000; 538–542..

[2] Accellera. *PSL Language Reference Manual, ver. 1.1. www.eda.org/vfv/docs/PSL-v1.1.pdf*. 2004.

[3] B Cohen, S Venkataramanan, A Kumari. *Using PSL/ Sugar for Formal and Dynamic Verification. Vhdl CohenPublishing, Los Angeles, California*. 2004.

[4] A Dahan et al. *Combining System Level Modelingwith Assertion Based Verification. Intl. Symposium on Quality Electronic Design.* 2005; 3A.1

[5] M Gordon, J Hurd, K Slind. *Executing the Formal Semantics of the Accellera Property Specification Language by Mechanised Theorem Proving. Lecture Notes in Computer Science.* 2003; 2860: 200–215.

[6] A Habibi, S Tahar. *Design for Verification of SystemC Transaction Level Models. Design Automation and Test in Europe.* 2005; *5A.4*.

[7] IBM AlphaWorks. FoCs Property Checkers Generator ver. 2.02. *www.alphaworks.ibm.com/tech/FoCs*. 2005.

[8] Mentor Graphics. *0–in Assertion Synthesis. http://www.mentor.com/products/fv/abv/0-in/index.cfm,* 2005.

[9] K Ng, A Hu, J Yang. *Generating Monitor Circuits for Simulation–Friendly GSTE Assertion Graphs. Intl.Conference on Computer Design*. 2004; 488–492.

[10] J Ruf, D Hoffmann, T Kropf, W Rosenstiel. *Simulation–Guided Property Checking Based on Multi–Valued AR Automata. Design, Automation and Test inEurope.* 2001; 742–748..

[11] A Sen, J Bhadra, V Garg, J Abraham. FormalVerification of a System–on–Chip Using Computation Slicing. *Intl. Test Conference*. 2004; 810–819.