ELECTRONICS AND ELECTRICAL ENGINEERING
ISSN 1392 – 1215 *2007. No. 3(75)*
ELEKTRONIKA IR ELEKTROTECHNIKA

SIGNAL TECHNOLOGY
T121
SIGNALŲ TECHNOLOGIJA

Verifiable Template Development for HDL- Descriptions

Y. Syrevitch, D. Zinchenko
DA dep., KhNURE, Kharkov, 61726, UKRAINE, e-mails: syr_Jane@rambler.ru; darijaz@ukr.net

Introduction

In modern CAD tools the basic way of device description is usage of hardware description languages, i.e. VHDL or Verilog, which allow making SOC design process faster. World companies – vendors of digital circuits, are forced to decrease their time-to-market. According to vendors' evaluations, verification (functional as well) takes up to 80% of labor expenditures in the design cycle. There is a big demand for tools of functional verification of devices models on a step of their description in hardware description language (HDL) on behavioral level. Verification systems, used in the world (HDL Score, Verix™, Hammer 100, SpyGlass**,** Questa AFV), do not work on behavioral level. So creation of functional verification system is an issue of the day. To obtain good effectiveness of verification, it is necessary to have information about VHDL description type and its capability to be verified. Requirements of design-for-verification can be divided into several big groups.

Problem statement. It is necessary to formalize verification strategy, proposed in [2, 5], and to develop a template of HDL-model of digital device, which will fit verification objectives (similarly to design-for-test rules).

Verification Strategy on a Base on Graph Model and Path Sensitization Method

The proposed strategy is based on origin HDL-model transformation into graph model, which is a composition of two graphs. First - information - describes dataflow and their conversion (similarly to an operational automaton in classical composite model with microprogram handle) without the registration of conditional branches. The second graph is developed as a network of conditions. The dataflow I-graph contains vertexes of 2 types: operands and functions. Arcs connect vertexes in the following manner: a source vertex is connected to a functional vertex, and then the arc goes out of the functional vertex and comes into a destination vertex. The arcs, which come into destination vertexes, can be conditional or non-conditional. Conditional arcs correspond to the operators, which are inside conditional expressions of VHDL. Conditional arcs contain labels, which code conditions of arc transition operation. In its turn, the second graph (a control one) contains conditional constructions from the origin device description. Each predicate in the condition is modeled as a subgraph, which has a specific label [7]. Functional elements (FE) are multibit logic and arithmetic functions, which correspond to VHDL operators.

Principles of structural testing are used for functional verification. These principles are based on path sensitization in a device model. One of the necessary definitions in the proposed strategy is the definition of distinguishing sequences. Distinguishing sequences (DS) are those, when being driven onto different functions they will give different outputs for same inputs. DS allows finding errors, connected with VHDL OPERATION SUBSTITUTION [2].

The main proved theorem says, that to identify all functional elements in I-graph it is necessary and enough to activate all paths in a graph which cover it, starting from the 1^{st} rank to graph outputs or control points. Thus, it is possible to make a conclusion, that identification of all FEs in VHDL-model checks data processing mechanism. Assembly of tested data processing mechanisms presents verifying model to specification correspondence. On the assumption of formed and proved lemmas, statements, and theorems, general verification strategy, based on functional elements sensitization, starting from 1^{st}-rank element. It consists of the following steps [5].

1. Activation of the i^{th} FE of 1^{st} rank is carried out. Distinguishing sequences are driven directly from external inputs of a graph model. External inputs (outputs) of I-graph are operand vertexes, which are ports in HDL-model.
2. Sensitized path is build from activated FE to either external outputs in a graph (output ports in HDL-models) or a control point.
3. Activation is finished, if set operand vertex is reached or path sensitization is impossible.
4. Steps 1-4 repeat for all FEs of the 1^{st} rank.
5. After finishing the 1^{st} rank FEs activation next k^{ary} FE of the p^{ary}-rank ($p>1, k=\overline{1,n}$), which do not belong to any already sensitized paths, is activated.
6. Activation is carrying out until all FEs are activated.

Classification of digital devices by types of their language descriptions

HDLs allow describing devices of different complexity and purpose. Quality of verification depends on device type and possibility of etalon restoration. Types of device description s can be divided into:

1. **Simple**:

– Descriptions of logic and arithmetic combinational devices – Boolean equations, encoders /decoders, multiplexers, adders/ subtractors, comparator, parity control, shifters, multipliers/dividers, etc;

– Descriptions of sequential devices – flip-flops, counters, registers, memory (RAM, ROM);

– Descriptions of state machines – «pure» state machines (Mealy, Moore) in a case, when "automata" template is used.

2. **Complex**:

– Descriptions of devices with microprogramming control (ALU, microprocessor);

– Descriptions of algorithmic devices, which realize algebraic, trigonometric or transformations;

– hardware-oriented descriptions with usage of libraries, specified for chosen hardware implementation;

– Descriptions of interfaces (UART, adapter unit and transmitters/receivers without data transformations);

– Descriptions of compositional devices (operational device with inseparable and valuable data processing part);

– Non-template descriptions – devices, which cannot be put to any group or they contain parts of descriptions of different.

Verification strategy adjustment depending on description types

Quality of verification for different descriptions depends on the end aim of the verification. For type 1 (simple descriptions) it is necessary to check separate components and functions. For type 2 (complex descriptions) verification checks "explicit" and "implicit" modes of work.

Let's consider in details:

1. Descriptions of logic and arithmetic combinational devices – proposed strategy allows to check modes of device work on a base of correct usage of the operators.

2. Descriptions of sequential devices – proposed strategy allows to check conditions of work enable and device functionality as well.

3. Descriptions of state machines – proposed strategy allows, from one side, to carry out standard method of automata check (vertex and transitions pass), and from the other side, to execute diagnostic procedures.

4. Descriptions of devices with microprogramming control – proposed strategy allows to check correspondence between modes inside the code, and modes from the specification.

5. Descriptions of algorithmic devices – not all possible values are driven onto input signals, but only those, which check given modes. Besides, proposed strategy is used for diagnostic procedures.

6. Hardware-oriented descriptions – assume that hardware-oriented libraries do not contain mistakes inside. So after their activation of all libraries, it is necessary to operate in correspondence with its main type.

7. Descriptions of interfaces - proposed strategy allows to check control part (conditions of exchange algorithm forming). However proposed strategy does not check timing parameters.

8. Descriptions of compositional devices – proposed strategy allows to check control part (conditions of modes forming) and operational (operators in modes).

9. Non-template descriptions – it is necessary to set if possible, standard types of descriptions and to apply them correspondingly.

Let's give ground to some of the statements.

Descriptions of logic and arithmetic combinational and sequential devices (close to hardware realization) usually use operators of concurrent signal assignment or simple constructions, such as processes with a set of assignment and conditions checking. Thus, having checked separate elements-operators, it if possible to say, that all specification is checked.

Models of finite state machines, described according to a template, contain operators (for calculating notification signals) either on arcs or in states. Thus, the strategy that checks operators will check conditions of arcs and outputs forming. It coincides with arcs and states traversal.

In description of devices with microprogram control there is a block of microcommand analysis, and blocks, which implement calculations, the proposed strategy allows checking correspondence between modes in a code and modes in specification.

Descriptions of algorithmic devices contain arbitrary written code without templates, where it is impossible to select parts, close to hardware. However there is a benefit in this case: in algorithmic devices it is possible to obtain etalon reactions for any input value. Thus it is possible to use specification mode check.

For proof of approach for algorithmic descriptions, given in paragraph 5, let's formulate the following statement.

Statement. If a function, implemented by a behavioral model of a certain device, is continuous on all possible values of input signals, then within these possible values the model behaves unambiguously and correspondingly to the calculated range space of the function. The proof was based on Bolcano-Koshi theorem and its extension.

Corollary. Number of points at continuous function test doesn't depend on all possible values of input stimuli. Verification strategy usage for continuous function model results in a set of vectors, which checks, the model concerning chosen design error.

A description of interfaces contains a mixture of styles and ways of description, but it is possible to mark blocks, responsible for control (mode selection) and for operations (data processing inside a mode). Correspondingly, having checked an informational graph (and connected with it control graph, as well), all modes from the specification are checked.

Requirements an a Template for Digital Device Verifiable Description

Consider a problem of template formulating. These templates can be used for creating HDL-models, fit to verification. There is a task of principles development, which will allow creating verifiable language models

The growth of number of computers during last 20-30 years is accompanied by continuous increasing of functional possibilities and further complication of element base structure. The necessity in researches of principally new possibilities of qualitative solutions of testing problems has appeared [6]. Methods of design-for-test appeared to be these new possibilities. DFT is, in general case, a way of logic circuit design that provides availability of a circuit to be tested and controlled. Non-testable circuit cannot be either tested adequately, or, if possible, testing will take a lot of time for creating and driving tests [6]. Similar with describing a system as a code. Non-verifiable HDL-model is a code, which is tested with unacceptably long test (time for testing increases); time for test generation is big and/or it needs a complicated algorithm; test quality is unacceptably low (especially for hierarchical models, models with big number of non-functional code and/or with big amount of atypical approaches of programming).

Thus either it is impossible to verify adequately , or if possible, it takes a lot of time for test generation and testing itself. Requirements of design-for-verification can be divided into several big groups. We added requirements, directed to increasing of effectiveness of the proposed strategy.

1. Structural organization requirements:

– in describing devices with microprogram control it is necessary to organized concurrent analysis of microcommand fields;

– in state machine description, it is necessary to use one-, two-, or three-process templates (along with using IF and CASE operators) .

– in describing algorithmic unit, it is important to use, as minimum, three blocks – a block of processing input signals, a block of algorithm processing, and a block of output values correction, etc.

2. General software and functionally specific organization requirements. Good style of programming is: a program is written structurally, is readable, with correct usage of all language's resources and with all lists of full texts.

3. VHDL verification requirements.

– Do not use long if-then-else constructions, use case operator instead (it decreases the depth of C-graph "spinup").

– Do not use default (or initialized) values. Use reset for initializing variables and signals (it increases "flexibility" of I-graph).

– Use additional signals with **out** mode for reading output values instead of using **buffer** mode (it decreases number of feedbacks in I-graph).

– Do not write long lines of a code. Write one operator per line. Use no more that 50 operators per each block of statements (it eases I-graph creation).

– Nested constructions should have no more than triple depth (it eases creation of C-graph).

– Use addressing to a vector range instead of subtype usage. Do not indicate vector dimension, when using the full one (it allows to avoid operand vertexes fragmentation).

4. Observability requirements. In order to increase code observability, it is desirable to use additional signals with type out in a way to produce values on them in the end of a process.

5. Interpretability requirements. Output values should be understandable and easily interpretive by a coder, an engineer, and a verifier. In a case, when implementation was not a loan translation, for values, calculated in a code, it is necessary to find corresponding dependencies in a specification.

Thus, we have formulated expanded requirements for models descriptions.

Implementation

Here are the results of diagnostic experiment for 3 types of devices [5]: control device b06 from ITC benchmark library, sectional microprocessor KP1804BC1, and sequential device s27 from ISCAS'95 benchmark library. Fig. 1 shows histograms of test length dependence from test type. In the figure: 1 – test with all possible values ($2^n \times 2^m$, where n – number of input bits, m – number of memory elements); 2 – program code test (2^n); 3 – specification modes test ($2^n \times M$, where n – number of input bits, M – number of modes). As it is shown in the histogram, not in all cases the proposed strategy gives valuable decreasing of test length, however, even giving little growth of quantity, tests on a base of path sensitization give 100% of errors coverage (design errors of "Operator Substitution" type).

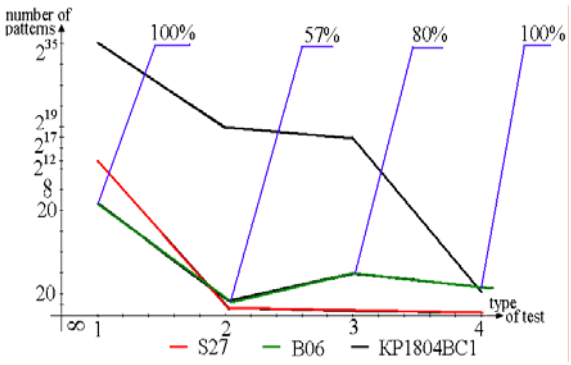

Fig. 1. Dependence of test length from the type of testing for S27, KP1804BC1, B06

Analysis was done on a base of path sensitization and with accounting verifiability of a model.

Conclusions

Scientific novelty and practical usefulness of the obtained results consist in:

1. Classification of digital devices by types of their VHDL-descriptions is carried out to more effective execution of verification;

2. Adjustments of path sensitization verification strategy are done;

3. A list of requirements for HDL-model is proposed, that includes both software requirements and specific criteria.

References

1. **Bergeron J.** Writing Testbenches: Functional Verification Of HDL Models // Kluwer Publishers, 2003. – 354 p.
2. **Syrevitch Yev., Karasyov A., Mehana S.S.** Functional verification quality metrics at HDL-models verification // Radioelectronic systems. – 2006. – Vol. 6. – C. 153–157.
3. **Harry D. Foster, Adam C. Krolnik, David J.** Lacey Assertion-Based Design // Kluwer Publishers, USA. – 363 p.
4. **Tasiran S., Keutzer K.** Coverage Metrics For Functional Validation Of Hardware Designs // IEEE Design & Test Of Computers.– July-August 2001.– P. 36–45.
5. **Shkil A., Syrevitch Yev., Karasev A., Cheglikov D.** Test Verification of Behavioral HDL-models // ASU and pribory avtomatiki. – 2006. –Vol. 134. –C. 4–12.
6. **Bennets R.** Boundary-Scan // Asset Intertech.–2000.– 130 p.
7. **Kryvulya G., Syrevitch Yev., Karasyov A., Cheglikov D.** Internal Model Algorithms For Digital Design Verification of VHDL Descriptions // Proc. of the Intern. Conf. CADSM.–2005. – Lviv-Polyana, UKRAINE. – P. 369–372.

Submitted for publication 2006 12 29

Y. Syrevitch, D. Zinchenko. Verifiable Template Development for HDL- Descriptions // Electronics and Electrical Engineering. – Kaunas: Technologija, 2007. – No. 3(75). – P. 49–52.

Classification of digital devices by types of their language descriptions is introduced. Also, a template of HDL-model of digital device, which will fit verification objectives in a case of using path sensitization methods, is considered. The proposed strategy starts from origin HDL-model transformation into a graph model, which is a composition of two graphs. To identify all functional elements in an informational graph it is necessary and enough to activate all paths in a graph which cover it, starting from the 1st rank to graph outputs or control points. Usage of a template allows building a graph model of HDL-description and further verification easier. Adjustments of path sensitization verification strategy are done. Dependence of test length from the type of testing for S27, KP1804BC1, and B06 benchmarks is analyzed. Ill. 1, bibl. 7 (in English; summaries in English, Russian and Lithuanian).

Е. Сыревич, Д. Зинченко. Разработка шаблона HDL-описаний, удовлетворяющего требованиям верифицируемости // Электроника и электротехника. – Каунас: Технология, 2007. –№ 3(75). – С. 49–52.

Предлагается классификация цифровых устройств в зависимости от типа их языкового описания при верификации. Также предлагается шаблон HDL-модели цифрового устройства, удовлетворяющего требованиям верифицируемости при использовании методов активизации путей. Предлагаемая стратегия начинается с преобразования HDL-модели в графовую, представляющую композицию двух графов. Для идентификации всех функциональных элементов в информационном графе необходимо и достаточно активизировать все пути в графе, которые его покрывают, начиная с первого ранга и до выходов графа или контрольных точек. Использование шаблона позволяет облегчить и упростить построение графовой модели и дальнейшую верификацию. Выполнено уточнение стратегии верификации на основе активизации путей в графовой модели. Анализируется зависимость длины теста от типа тестирования для систем тестов S27, КР1804ВС1 и В06. Ил. 1, библ. 7 (на английском языке; рефераты на английском, русском и литовском яз.).

J. Syrevič, D. Zinčenko. Verifikuojamų HDL aprašų šablonų kūrimas // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2007. – Nr. 3(75).– P. 49–52.

Pasiūlyta skaitmeninių įtaisų klasifikacija pagal jų kalbinius verifikavimo aprašus. Taip pat pasiūlytas skaitmeninio įtaiso HDL modelio šablonas, tenkinantis verifikuojamumo reikalavimus, kai taikomi kelių aktyvinimo metodai. Siūloma strategija prasideda nuo pradinio HDL modelio transformavimo į grafų modelį, kurį sudaro dviejų grafų kompozicija. Norint identifikuoti visus funkcinius informacinio grafo elementus, būtina ir pakanka suaktyvinti visus grafą sudarančius kelius, pradedant nuo pirmojo rango ir baigiant grafo išėjimais arba kontroliniais taškais. Naudojant šablonus, galima palengvinti ir supaprastinti grafų modelio sudarymą ir tolesnį verifikavimą. Patikslinta kelių aktyvinimu pagrįsta verifikavimo strategija patikslinimas. Analizuojama S27, KP1804BC1 ir B06 įtaisų testų trukmės priklausomybė nuo testo tipo. Il. 1, bibl. 7 (anglų kalba; santraukos anglų, rusų ir lietuvių k.).