2015 International Conference on Soft Computing Techniques and Implementations- (ICSCTI)
Department of ECE, FET, MRIU, Faridabad, India, Oct 8-10, 2015

Tackling SoC Abstraction Gap: Raising the Abstraction Level of DUT Verification

Nishit Gupta, Sunil Alag
R&D in Electronics Group, Department of Electronics & Information Technology
Ministry of Communications and Information Technology, Government of India
New Delhi, India

Abstract— To meet the ever increasing demand of consumer electronics industry, the complexity of System on Chip has increased rapidly and thus, in the past few years efforts are being made to reuse IPs at different abstraction levels from different vendors designed using various hardware description languages. Complex IPs, with different data Payload Structures and Models of Computation, are integrated in SoC making it difficult to trace the simulation issues of Design Under Test, i.e. DUT at system level. To address this issue, TLM 2.0 suggests modeling the System on Chip at higher abstraction level. TLM 2.0 advocates the usage of TLM models as Golden Reference for RTL SoC. The TLM Model of the DUT developed as a part of TLM test bench can be conveniently used at system level for analyzing simulation issues related to RTL DUT but comparing and subsequently analyzing the simulation results of TLM and RTL test benches is a cumbersome task and has yet not standardized. Thus, almost all the Electronics System Level users are devising customized methodologies convenient to them for comparing the simulation results of TLM and RTL test benches.

In this work, a novel approach is proposed to list the discrepancy between the TLM and RTL test benches and compute the divergence percentage for functional verification of DUT. No emphasis is given to the information held in internal structure of DUT and only functional aspect of DUT is taken into consideration. Selective parameters of Transaction Payload are thus, conveniently compared during the simulation time across the various abstraction levels to find out the ambiguous behavior of DUT. Based on empirical results, we argue that the annotations proposed causes reasonable overhead, but provide convenient approach for functional verification of complex DUT.

Keywords—AXI, Bus Cycle Accurate, DUT, SCV, SST2, System C, System on Chip, TLM

I. INTRODUCTION

The classical System on Chip design flow starts from the RTL design which involves writing the register accurate and clock-cycle accurate complex RTL models. At system level, considering the huge efforts required in debugging the register accurate and clock-cycle accurate complex RTL, DUT and their high simulation time, various efforts [1, 2, 6, 7] have been made in the last few decades to raise the abstraction level of SoC modeling for making available a lighter version of SoC. The TLM based methodology raise the level of abstraction for designing and verification of SoC. Many verification strategies [12, 15] advocate preparing a golden reference model using procedural or object oriented languages such as C and C++. Instead, with a little effort [3, 9, 11, 14] accurate TLM models can be developed.

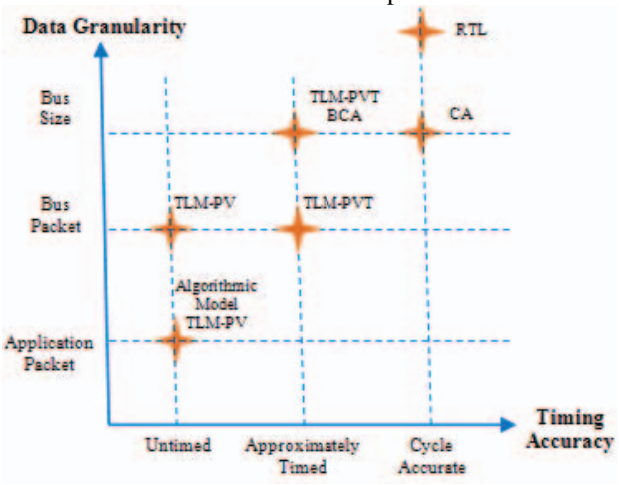

Fig. 1. Model of Computations: Timing accuracy and Granularity

However, this little effort introduces reusability of design. As shown in Fig. 1, TLM Model though relatively inaccurate, can be conveniently developed with lesser efforts and duration as compared to RTL. Thus, the system level developers are predominantly advocating developing the TLM Model of DUT acting as Golden Reference model. The simulation is then carried out with TLM model of DUT under the TLM test bench. All test scenarios, associated input stimuli and the simulation results are saved as Golden reference after extensive verification. Subsequently, the RTL model of the DUT is integrated in same TLM test bench and simulation is carried out with the same stimuli and test scenarios obtained earlier as shown in Fig. 2. The obtained simulation results are then compared to that obtained earlier from pure TLM test bench. While carrying out such a comparison each vendor adopts a customized or non-standard approach suiting to their requirement.

In this work, a novel approach is elaborated using a toolset named DiffxT, as shown in Fig. 3, to list the discrepancy between the TLM and RTL test benches and compute the divergence percentage. Using DiffxT, user can select the attributes to consider while carrying out the comparison. All transaction attributes like address, opcode, timing, data etc. are treated equally in DiffxT. Due to this feature, DiffxT can compute the divergence of transactions occurring at different simulation times in test benches modeled using different model of computation. DiffxT provides a

filtering mechanism to filter out the extra transactions and comparing only relevant transactions as specified by the tags defined in the input configuration file.

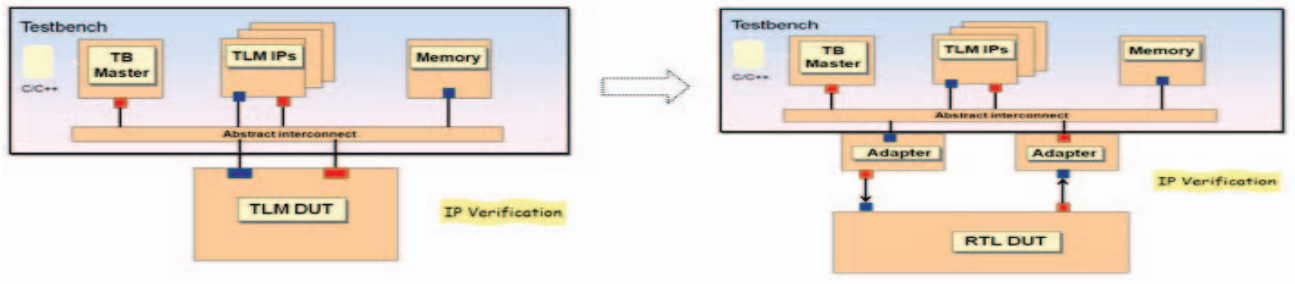

Fig. 2. Analysing RTL DUT in TLM test bench

II. RELATED WORK

While the concept of deploying the Transactional Level Test bench for the verification of RTL DUT is not new [1], not much effort has been put in the direction of automation and standardization of the process of find the mismatch between the RTL and TLM test benches. Thus, almost all the ESL users are devising customized methodologies convenient to them for comparing the simulation results of TLM and RTL test benches. The proposed approach strongly advocates the usage of executable specifications, i.e. TLM test bench, and also identifies the crucial transaction payload parameters to be compared. As a result, a toolset is implemented which can non-intrusively carried out the desired task.

III. TLM GOLDEN REFERENCE: APPROACH FOR SOC VERIFICATION

TLM Golden Reference model approach adopts a redundancy approach in which the duplication of critical components or functions of a system is carried out with an intention of increasing reliability of the system as shown in Fig. 3. In this approach, the TLM and RTL designs are carried out separately by two different teams of engineers, i.e. hardware design engineering team and system level developer team. Such a dual interpretation of the same SoC, done independently by two unrelated teams, can catch discrepancy between the behaviour of test benches at TLM and RTL level. The ultimate goal of this approach is to functionally verify the RTL DUT against the TLM DUT with an assumption, supported by inputs from system level developer team, that the simulation results of TLM results are accurate and as expected. The other aspects of the verification are not taken into consideration. To achieve this, the TLM model of DUT is simulated in TLM test bench to obtain the Golden reference results. RTL model of DUT is then simulated in the same TLM test bench. The obtained simulation results of TLM test bench simulating the RTL DUT, are compared to the Golden reference subsequently. Simulations can be compared by following methods:

1. Simulation-dump comparison: In this approach, transactions from simulation dump are compared.
2. Simulation-run comparison: In this approach, TLM and RTL simulations are compared at particular instance of times by running them simultaneously.

The proposed approach in this paper adopts the first method described above by simulating the RTL DUT in the TLM test bench and comparing the simulation traces obtained with that of TLM test bench developed as Golden Reference. The Bus Functional Models or adaptors are used to raise the level of abstraction of RTL transactions as shown in Fig. 2. As the transactions compared in two simulation databases, i.e. RTL and TLM test benches, have different timing, and may have different attributes, thus a filtering mechanism and selective comparison needs to be performed.

IV. APPROACH FOR SELECTIVE COMPARISON OF SIMULATION DUMPS

As the result of the comparison, two files are produced with the name DiffxT_RESULT.TXT and DiffxT_TRANS. log. DiffxT_RESULT.TXT shows the percentage of deviation between the compared fibers and percentage of deviation among the attributes of transactions. The transactions are compared based on the attributes specified in the configuration file. DiffxT_TRANS.log shows all the transactions for all the fibers given in the configuration file. For each transaction it mentions the status of the operation carried on it. The configuration file, which is passed by user as argument to DiffxT contains the information about the databases to be compared, the fibers which will be compared and attributes on whose basis the comparison is carried out. The format of configuration file is explained in next section. The present mode of operation allows following database type comparisons:

1. scv-scv
2. sst2-sst2

The type of database should be specified in the configuration file. The simulation dumps (database) can be from different level of abstractions or for different buses. One

typical usage would be to compare the transaction dumps for embedded s/w simulation over transaction level platform with RTL platform from different level of abstractions or for different buses.

V. CONFIGURATION FILE SYNTAX

The configuration file, provided to DiffxT library as a command line argument, contains:

1. Names and paths of the databases to be compared.
2. Name of the Fiber to be compared. It also mentions that whether the Fiber is to be treated as reference fiber.
3. Name of the attributes along with mask values to be compared.
4. Different filtering tags as explained in next section.
5. Different selection tags as explained in next section.
6. Criteria of selection, if any, to be used as explained in next section.

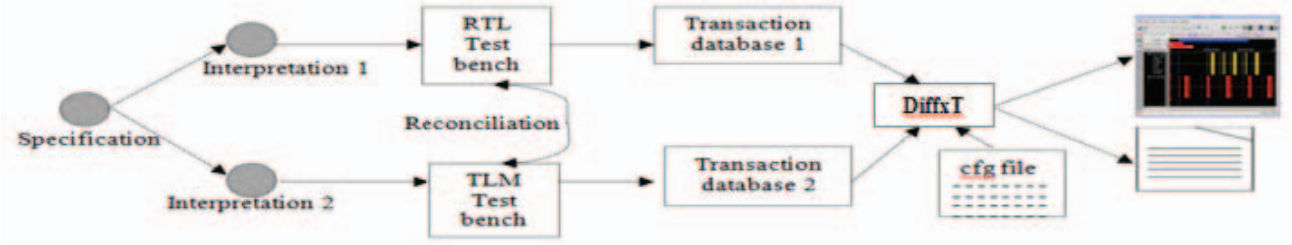

Fig. 3. xDiff approach for comparing TLM and RTL test benches

An example configurable file is shown below:

```
<DiffxT>
<Database ref = “database1” ref.val ="scv" cmp =
“database2” cmp.val ="scv" >
<Fiber ref = “Fiber1” cmp = “Fiber2”>
<Refine>
<Attribute ref = “Master” ref.val=”INIT1” cmp
=Slave cmp.val = "TARGET1" />
</Refine>
<Compare>
<Attribute ref = “Address” cmp = “addr” />
<Attribute ref = “Opcode” cmp = “opc” / >
</Compare>
</Fiber>
</Database>
</DiffxT >
```

VI. SELECTIVE COMPARISON

As the simulation dumps can be from different level of abstractions or for different buses with different simulation timing thus a mechanism of selective comparison is devised by way of supporting tags in xml configuration file. The various tags applied to a fiber to achieve selective comparison are described below:

1. *Refine:* When refining is defined for a fiber then only transactions which meet these refining criteria are compared. Any transaction which doesn’t meet refining criteria is dropped. Refining has two main use cases:
   a) Comparison with single fibers: Refining allows comparing transactions only if they meet refining criteria as show in Fig 4.

```
<Refine>
<Attribute ref = “Master” ref.val=”INIT1” cmp
=Slave cmp.val = "TARGET1" />
</Refine>
```

   b) Comparison with multiple fibers: Transaction of fiber on which refining is defined get compared with different fibers based on which refining criteria is satisfied as shown in Fig. 5.

```
<Refine>
<Attribute ref = “Address” ref.val=”0x10” />
</Refine>
```

2. *Poll:* a condition where a particular location is continuously queried till a valid response is obtained. Each request corresponds to a transaction. Polling ends whenever the location responds back with valid response. Poling is specified by begin condition, end condition and criteria as shown in Fig. 6.

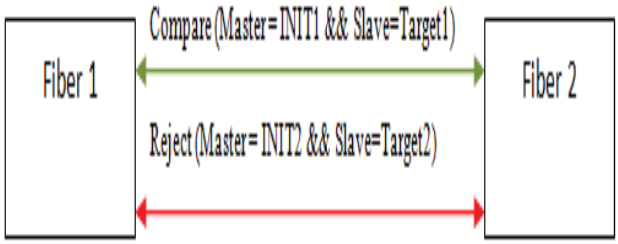

Fig. 4. Refine: Selective comparison with single fiber

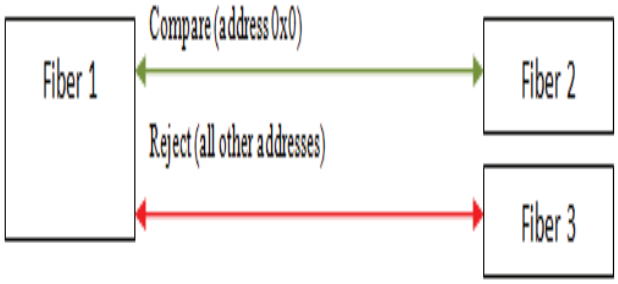

Fig. 5. Refine: Selective comparison with multiple fibers

```
<Poll>
<Criteria>
<Attribute ref = "Address" ref.val= "0x100"/>
</Criteria>
<BeginCondition>
<Attribute ref = "Status" ref.val= "BUSY" />
</BeginCondition>
<EndCondition>
<Attribute ref = "Status" ref.val= "BUSY" />
</EndCondition>
</Poll>
```

3. *Filter:* defined to drop transaction when it meets a defined condition. Attributes with certain value specify a filtering condition. If a transaction matches such a condition then that transaction is dropped and not compared as shown in Fig. 7.

```
<Filter>
<Attribute ref = "Address" ref.val = "0d4" cmp
= "addr" cmp.val = "0d8" />
<Attribute ref = "Data" ref.val = "0d4" />
</Filter>
```

4. *Diff:* is used when transactions in two databases are misaligned due to difference in timing or abstraction level typically characterized by a few transactions which don't match but subsequent transactions match. The databases have actually only a few

```
<Diff>
<Attribute ref = "Address" cmp = "addr" />
<Attribute ref = Data cmp = data />
</Diff>
```

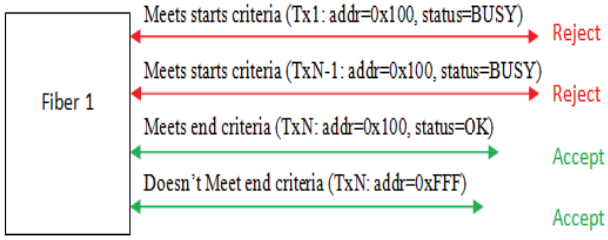

Fig. 6. Polling: Selective comparison

transactions which are extra. In Diff mode these extra transactions are removed before making the comparison as shown in Fig. 8

5. *Match:* This algorithm is useful when one of the fibers either ref or cmp is treated as golden or standard and the other fiber is compared against it. One use case is when it is known in advance that either ref or cmp database has extra transactions on the fibers which are to be compared. To be more precise, both fibers are supposed to match exactly, everywhere but one of the fibers (either the ref or cmp), with config file value as FALSE (Test_Fiber) has extra transactions that are not in the other fiber, with config file value as TRUE (Golden_Fiber) Then DiffxT iterates on Golden_Fiber, and tries to match each transaction with the corresponding transaction of the Test_Fiber. If the corresponding transaction doesn't match, it is reported as an extra transaction (not an error transaction, as it is expected), and the tool tries to match with the following transaction of Test_Fiber, and so on. At the end of the comparison, all the transactions of the Golden_Fiber should have been matched with the transactions of the Test_Fiber. Otherwise, DiffxT reports an error on those extra transactions.

```
<Match ref = TRUE cmp = FALSE >
<Attribute ref = "Address" cmp = "addr" />
<Attribute ref = Data cmp = data />
</Match>
```

6. *Masking:* This feature allows only selected bits/bytes of attribute to be used for comparison. The rest of the bits/bytes are not used for comparison or "masked" out. Masking has two use-cases:
   a) Masking value is specified within the definition of attribute itself.
   b) Mask value is provided by a different attribute.

In an example below, "Data" mask value is binary "100" and for "data" attribute it is given by the value of Byte_Enable attribute.

```
<Attribute ref = Data cmp = data >
<Mask ref = "0b100" cmp = Byte_Enable />
</Attribute>
```

7. *Offset:* This feature allows comparing attributes after addition/subtraction of offset value from the attribute's value. If the offset value is positive then it

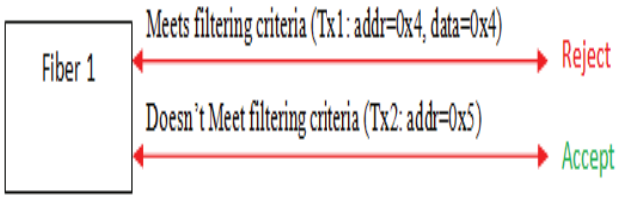

Fig. 7. Filter: Selective comparison

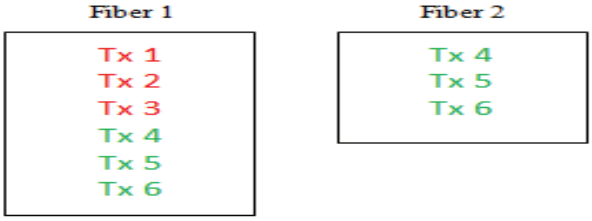

Fig. 8. Diff:Selective comparison

is added else it is subtracted. In an example below, for "Data" Offset value is hex "10" and for "data" attribute Offset value is hex "-20"

```
<Attribute ref = Data cmp = data >
<Offset ref = “0x10” cmp = "-0x20" />
</Attribute>
```

VII. OUTPUT

DiffxT is launched by a simple command as below:

DiffxT –cfg config.cfg

The result files dumped, summarizes the result of comparison carried out along with the providing the below information:

1. The name for ref and cmp fiber
2. Algorithm (Diff/Normal) used while comparison
3. Total no. of transactions found for each set of comparison fibers
4. Transactions removed due to filtering (If filter specified in configuration file)
5. Transactions removed due to polling (If poll specified in configuration file)
6. Transactions removed due to refining (If transactions don’t meet refine criteria)
7. Transactions removed due to Diff, these are transactions which don’t meet "Diff" criteria. They are removed to make the transactions from the two compared fibers aligned.
8. Extra transactions (obtained as above in 4, 5, 6 & 7 serial numbers) which are not compared
9. Transactions Compared, these are the transactions which are actually compared.

```
Ref Fiber:                  TOP.MEMORY
Cmp Fiber:                  AXI3_stream
Comparison Algorithm:       Normal
+-----------------------------------+----------------------+
|                                   |  ref   |   cmp   |
+-----------------------------------+----------+---------+
| Total Transactions found          |  45    |    9    |
| Extra Transactions                |  36    |   0     |
| Total Transactions Compared       |   9    |   9     |
+-----------------------------------+----------+---------+
Total Error Transaction:    9
Transaction Convergence: 80
+----------------------+--------------------+--------------+
|    Attribute         |  Deviation         | First Error  |
+----------------------+--------------------+--------------+
|  Address/addr        |  88.8889           | 20ns/300ns   |
|   Data/data          |  88.8889           | 0s/200ns     |
+----------------------+--------------------+--------------+
Total Deviation Percentage is 88.8889
```

Fig.9. DiffxT Transaction log

10. Total Error Transactions: Gives the transactions out of transactions compared, which don’t match.
11. Transaction Convergences. It is calculated as follows:

(1-min(Ref_Fiber_Transaction, Cmp_fiber_Transaction)/Max(Ref_Fiber_Transaction, Cmp_fiber_Transaction)) *100

Where Ref_Fiber_Transaction & Cmp_Fiber_Transaction = 7 & 8 given above

For each attribute it shows the percentage of transactions in which their values differ and total error percentage for

```
_____________________________________________
Ref:      Fiber1          |
Status:   Filtered        |
Trans     Count1/1        |
Trans.    ID: 1           |
Trans.    Start Time: 0s  |
Trans.    End Time: 20s   |
Address = 0X2             |
Data =    0X1             |
__________________________|__________________
                          |Cmp:          Fiber2
                          |Status:       Polled
                          |Trans. Count  1 /1
                          |Trans. ID:    11

                          |End Time      10s
                          |Address =     0X2
                          |Data =        0X1
__________________________|__________________
```

Fig.10. DiffxT Result log

all the attributes. A set of files, one for each pair of fibers given in the configuration file, is created. These files are simvision scripts which allow the user to directly view the error transactions in simvision window. Fig. 9 and Fig. 10 shows the excerpts from Transaction log and Result log obtained by running DiffxT on RTL AXI3 slave DUT in RTL and TLM test bench, reflecting effectiveness of the approach used. With the results obtained, it can be conveniently deduced that for address and data compared in respective transactions from RTL and TLM test benches, 36 transactions are filtered and in total 9 transactions are reported as erroneous.

VIII. FURTHER WORK

Efforts are being made to add an intermediate step of converting the simulation databases obtained from simulations at various abstraction levels with different HDL into a common ascii format, i.e. SCV. This step will remove the delays in obtaining license of simulators and will introduce interoperability and subsequently standardization among various vendors. Conversion of RTL simulation database into

TLM2.0 compliant simulation database before comparison will be the further step to tackle the wider issues related to difference in abstraction level.

IX. CONCLUSION

As the complexity of SoC is increasing, conventional methods of RTL DUT verification are necessary but not sufficient as verification at RTL level is becoming more complex and time consuming task. The advantages of TLM test bench for embedded software development has been accepted and well adopted across the industry. The same TLM test bench, as Golden Reference, can be deployed efficiently to verify the complex RTL DUT with little overheads by adopting the novel approach describe in this work.

REFERENCES

[1] A. Clouard et al., “Towards Bridging the Precision Gap between SoC Transactional and Cycle Accurate Levels”, in Proc. of Design, Automation and Test in Europe Conf., Paris, France, 2002.

[2] Pasricha, S. “Transaction level modeling of SoC with SystemC 2.0” in Synopsys Users Group Conference (SNUG), 2002.

[3] Grotker, T., Liao, S., Martin, G., Swan, S., System Design with SystemC, Kluwer Academic Publishers, 2002.

[4] O. Ogawa et al., "A practical approach for bus architecture optimization at transaction level", in Proc. of Design, Automation and Test in Europe Conf., 2003, pp. 176-181.

[5] M. Caldari et al., “Transaction-level models for AMBA bus architecture using SystemC 2.0”, in Proc. of Design, Automation and Test in Europe Conf., 2003, pp. 26-31.

[6] L. Cai and D. Gajski, “Transaction Level Modeling: An Overview,” in Proc. of Int’l Conf. Hardware/Software Codesign and System Synthesis”, IEEE press, 2003, pp. 19-24.

[7] H. Jang et al., “High-Level System Modeling and Architecture Exploration with SystemC on a Network SoC: S3C2510 Case Study”, in Proc. of Design, Automation and Test in Europe Conf., 2004, pp. 538-543

[8] W. Klingauf. “Systematic Transaction Level Modeling of Embedded Systems with SystemC”, in Proc. of Design, Automation and Test in Europe Conf., 2005, pp. 566-567.

[9] IEEE Standard SystemC Language Refernce Manual, IEEE Std. 1666, 2005.

[10] D.C. Black and J. Donovan, SystemC: From theGround Up, Springer-Verilag New York, Inc., 2005.

[11] F. Ghenassia., Transaction-Level Modeling with SystemC: TLM Concepts and Applications for Embedded Systems. Springer, 2006

[12] Bacchini, Francine et al., “Building a common ESL design and verification methodology - is it just a dream?”, in Proc. of Design Automation Conference, 2006 43rd ACM/IEEE, pp. 370-371.

[13] Cornet, J., Maraninchi, F. and Maillet-Contoz, L., “A Method for the Efficient Development of Timed and Untimed Transaction-Level Models of Systems-on-Chip” in Proc. Of Design, Automation and Test in Europe, 2008, pp. 9-14.

[14] Accellera Systems Initiative, “SystemC”, 2012, available at https://www.systemc.org.

[15] Harrath, N., Monsuez, B., Barkaoui, K., “Verifying SystemC with predicate abstraction: A component based approach” in Proc. of 14th International Conerence. on Information Reuse and Integration (IRI), IEEE, 2013 , pp. 536-545.