Compositional Verification of IP Based Designs

Hong Peng and Sofiène Tahar

Dept. of Electrical & Computer Engineering,
Concordia University
1455 de Maisonneuve W., Montreal, Quebec,
H3G 1M8 Canada
E-mail: {pengh,tahar}@ece.concordia.ca

Abstract— **In this paper we propose a methodology for the compositional verification of IP-block based designs. We will focus on the reasoning of the compositional verification of the IP based design and on the issue of what interface behavior should be provided with an IP block to make the verification feasible. The proposed approach will be illustrated on the compositional verification of an ATM bit error monitor (BERMON) system.**

I. Introduction

In order to satisfy the fast changing market, a new methodology in the VLSI design, IP (Intellectual Property) block-based design, is widely deployed. Because a large system can be integrated using existing IP blocks, the IC (Integrated Circuit) development process is speeded up. In this type of design, an important issue is to verify whether the integrated system satisfies the required specification. The traditionally used simulation techniques cannot cover all design errors, especially for large systems. The formal verification in system design generally means using methods of mathematical proof rather than simulation and testing to insure the quality of a design, to improve the robustness of a design and to speed up the development [5]. During the past decade, one of the most significant developments in this area is the discovery of model checking for verifying temporal logic properties of finite state processes [4]. Its significance comes from the great ease of use of fully automatic methods, as well as from the fact that many synchronization and communication systems can be modeled as finite state processes. However, model checking suffers from the so called state space explosion problem. Compositional verification [1] [2] is one possible way to address the state explosion problem since it can verify each module separately. State explosion is of course one motivation for pursuing compositional verification. Compositional verification is advocated also for methodological reasons; the modularity of IP blocks in the system can be used in the compositional reasoning and can make the reasoning easily and efficiently. Suppose we are given an IP block-based system, which includes core modules (IP blocks) and customer design modules; the basic problem in the verification of the IP based design is to derive the properties of the whole design (the core modules and the customer design modules) from the properties of each module.

In this paper, we will focus on the compositional verification of IP based designs and on the issue of what interface behavior should be provided with an IP block to make the verification feasible. The proposed approach will be illustrated on the verification of an ATM bit error monitor (BERMON) system [6], which has three modules, a buffer controller (Buf_ctrl), a 32-bit CRC engine ($Crceng$) and a synchronous pulse generator ($Sysres$). Among them, the CRC engine and the pulse generator are IP blocks, but the buffer controller is a customized module. We verified the BERMON using the compositional verification approach and a model checking tool: VIS [3]. Throughout the verification, the compositional reasoning shows the potential to handle a large IP based design, given that the interface behavior of the IP blocks are provided.

The rest of this paper is organized as follows. Section 2 outlines the problem of compositional verification of a large system. The compositional verification approach is presented in Section 3. Section 4 includes the BERMON case study. Section 5 finally concludes the report.

II. Problem definition

The structure of the BERMON is shown in Figure 1. It has two IP blocks, $Sysres$ and $Crceng$, and a customized design, Buf_ctrl. We denote this system $Sysres \| Crceng \| Buf\_ctrl$, for the parallel composition of $Sysres$, $Crceng$ and Buf_ctrl. Given component specifications φ_{IP1} and φ_{IP2} for the IP processes $Sysres$ and $Crceng$, respectively, and, φ_{Buf_ctrl} for Buf_ctrl, we would like to reason according to

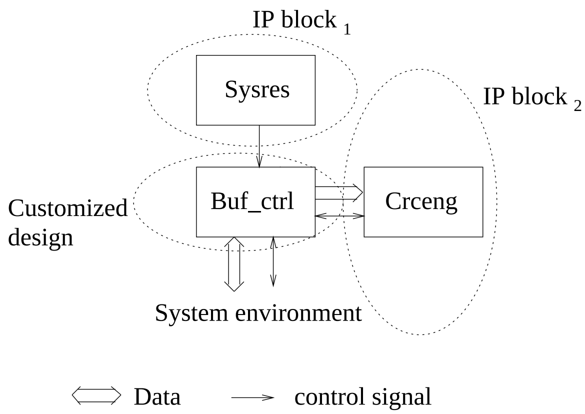

Fig. 1. BERMON architecture

the following schema:

$$\frac{\begin{array}{c}\langle\rangle Sysres \| Crceng \langle \varphi_{IP1} \wedge \varphi_{IP2} \rangle \\ \langle \varphi_{IP1} \wedge \varphi_{IP2} \rangle Buf\_ctrl \langle \varphi_{Buf\_ctrl} \rangle\end{array}}{\langle\rangle Sysres \| Crceng \| Buf\_ctrl \langle \varphi_{Buf\_ctrl} \rangle}$$

Here, we are asserting that if:
1. $Sysres \| Crceng$ satisfies $\varphi_{IP1} \wedge \varphi_{IP2}$; and
2. If the environment of Buf_ctrl satisfies $\varphi_{IP1} \wedge \varphi_{IP2}$, then Buf_ctrl satisfies φ_{Buf_ctrl}
3. Then the composition of $Sysres \| Crceng$ and Buf_ctrl will satisfy φ_{Buf_ctrl}.

Generally, there are a number of difficulties involved in developing a verifier that can support this style of reasoning. Firstly, we have to generate the environment assumptions $\varphi_{IP1} \wedge \varphi_{IP2}$, which is often very difficult to obtain. Secondly, in order to avoid erroneous conclusions, all chains of deduction must be well-founded, i.e., the base assumptions must themselves be proved without any assumptions. Many efforts have been presented to solve these problems [1] [2]. However, the IP block-based designs have inherent features to solve the problems because 1) the interface behavior of the IP blocks is clearly specified and 2) the design has a hierarchical structure based on the core modules which are supposed to be already proven correct [5].

III. Analysis of the verification approach

In this section, we develop a description form for IP block-based designs, and then we use compositional methods based on this form.

[Definition 1] System Environment Process. *Let Ω be a system with a property ϕ. A system environment process E^{ϕ} is the environment assumption for property ϕ.*

Such a system environment process will allow us to add some constraints on the inputs of the system so as to make the state space of the verified system smaller. Furthermore, the size of the process is small because it is specified with respect to only one property. It becomes possible to develop an algorithm to generate the process automatically.

Based on the above definition, we give the definition of the abstract environment array as follows.

[Definition 2] Abstract Environment Array. *Let Ω be a system composed of n implementation modules M_i $(i = 1 \ldots n)$. An abstract environment array E_Ω is an $(N+1) \times (N+1)$ array defined as follows,*

$$E_\Omega = \begin{pmatrix} \emptyset & E_1^{\phi} & E_2^{\phi} & \ldots & E_n^{\phi} \\ \emptyset & E_{11} & E_{12} & \ldots & E_{1n} \\ \emptyset & E_{21} & E_{22} & \ldots & E_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \emptyset & E_{(n-1),1} & E_{(n-1),2} & \ldots & E_{(n-1),n} \\ \emptyset & E_{n1} & E_{n2} & \ldots & E_{nn} \end{pmatrix}$$

where E_{ij} stands for a specification of all interface behaviors from module M_i to M_j. E_i^{ϕ} stands for system environment constrains with respect to a property ϕ.

If there are no signals from module M_i to module M_j, E_{ij} is denoted with $\emptyset$.

The array can be seen as a row vector or a column vector as follows,

$$\begin{pmatrix} E_0^I & E_1^I & \ldots & E_n^I \end{pmatrix}$$

where E_i^I represents all the environment input behaviors, and equals to the parallel sum of all the entries of one column in the array, namely $E_i^{\phi} \| E_{1i} \| E_{2i} \| \ldots \| E_{ni}$, or

$$\begin{pmatrix} E_0^O \\ E_1^O \\ \vdots \\ E_n^O \end{pmatrix}$$

where E_i^O represents all the environment output behaviors, and equals to the parallel sum of all the entries of one row in the array, namely $E_{i1} \| E_{i2} \| \ldots \| E_{in}$.

Since the system has outputs to its environment, we define an output process as follows.

[Definition 3] Output Process. *The output process of a system Ω is a vector,*

$$\begin{pmatrix} O_1 & O_2 \ldots & O_n \end{pmatrix}$$

where O_i is the specification of each module's output to the outside of the system.

In order to simplify the verification, we define a special case of the system Ω, which we will use in the sequel.

[Definition 4] Linear System. *A system Ω is linear if its abstract environment array is an upper triangular or lower triangular, namely the entries below or above the main diagonal are all $\emptyset$. An entry on the main diagonal of a triangular array may be $\emptyset$ or not.*

A linear system means that all the interface signals are in the same direction with respect to the order of the modules. In such a system, if there exist signals from E_i to E_j, then no signal exists from E_j to E_i. In this case, the order from M_1 to M_n or vice versa is a well-founded order.

[Composition Theorem 1] *Let Ω be a linear system composed of n modules M_i $(i = 1 \ldots n)$, i.e., $\Omega = \|_{i=1}^n M_i$;*

If

(1) $\forall j \in [1 : n-1], (M_j \| E_j^I) \Rightarrow E_j^O$;

(2) $M_n \| E_n^I \Rightarrow O_n$;

then, $\Omega \Rightarrow O_n$

Here "$\Rightarrow$" can be any logic implication. According to this theorem, in a linear system, if each module in the system composed with its environment process has correct outputs E_j^O, and the module under testing implies the correct outputs, then the whole system can guarantee the correct behaviors.

In the case of a non-linear system (system with feedback signals between modules), it is more complicated as we cannot get a well founded order on the modules. We use instead induction over time [8] and propose the following theorem to verify a non-linear system.

[Composition Theorem 2] *Let Ω be a non-linear system composed of n modules M_i $(i = 1 \ldots n)$, i.e., $\Omega = \|_{i=1}^n M_i$;*

If

(1) $\forall j \in [1 : n], M_j \uparrow^0 \Rightarrow E_i^O \uparrow^0$;

(2) $\forall j \in [1 : n], (M_j \| E_j^I) \uparrow^{t-1} \Rightarrow E_j^O \uparrow^t$;

(3) $(M_n \| E_n^I) \uparrow^{t-1} \Rightarrow O_n \uparrow^t$;

then, $\Omega \Rightarrow O_n$

Here, $p \uparrow^\tau$ stand for $\forall t.p$ or "p holds up to time t = τ". In this theorem, because there are feedback signals in the system, we request each module must have at least one unit time delay so that we can reason as following. Given correct inputs at time $t-1$, each module in the system has correct outputs at time t and the system is correct at time 0, we can conclude that the system guarantee the correct behaviors.

In the next section, we will use the above reasoning rules to verify the BERMON case.

IV. Modeling and Verification of the BERMON

A. The BERMON System

Protocol Data Units (PDUs) transported through an ATM network must be segmented into short, fixed-length packets, called cells, at the source end and reassembled at the destination end. Several such adaptation layer protocols are defined by the ATM standards [7], which specify how the process is carried out. ATM Adaptation Layer 5 (AAL5) is one of the ATM standards which is used in the frame-relay services or the signaling information transmission. It has strong bit error detection capabilities because of its 32 bit cyclic redundancy checking (CRC). It also makes better use of the bandwidth, because, unlike other adaptation layer protocols, AAL3/AAL4, it does not require overhead in any of the cells, except for some bytes in the information field of the last one. For these reasons, AAL5 tends to be the preferred method for breaking PDUs into cells. Because data integrity must be ensured end to end, a standard CRC technique is utilized on the entire AAL5 PDU. At the remote end, while reassembly is performed, the PDU is checked for data integrity. The polynomial chosen by the relevant standards bodies to implement the AAL5 CRC is the 32-degree polynomial:

$p(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

All the information in a PDU received at the remote end will be divided by the polynomial. The remainder equals to Hex C704DD7B meaning that the PDU is an exact copy of that one in the source end and no error occurs. The CRC checking and generation are generally carried out on a per-byte basis [7], in an attempt to cope with the dramatic increase of the data throughput of higher-speed lines.

The Bit Error Rate MONitor (BERMON) module is a part of the AAL5 error detection system. It acts as a monitor of incoming AAL5 PDUs on an ATM connection. There is a counter in the BERMON to record the number of PDUs with CRC errors. Whenever the incoming PDU contains CRC errors according to the CRC checking, the BERMON will increase the counter by one.

For monitoring the bit errors of an AAL5 PDU, the PDU is first read byte by byte from the input buffer. At the same time, the BERMON calculates the CRC value of the PDU on a per-byte basis. The calculation of the CRC value of a PDU is done according to the ITU-I.363 protocol standard [7]. Since the BERMON cannot change the content of a PDU as a monitor, the PDU should be sent back to the output buffer. The number of PDU errors is stored in the counter, which is visible to an outside controller.

The BERMON includes three modules, which are shown in Figure 1. The *Sysres* module is responsible for producing a synchronous system start signal. The *Crceng* module is the 32 bit CRC checking engine in the BERMON. It checks whether the CRC of an incoming AAL5 PDU complies with the ITU-I.363 standard. It works on a per-byte basis. Since an AAL5 PDU may contain many bytes, normally 48*N bytes, the byte based CRC checking results will be stored in a shift register array. The *Buf_ctrl* module is the interface between the above core modules and the I/O buffers.

B. Verification of the BERMON

The BERMON as a whole is a non-linear system. In the verification, the system is described using an environment array as described in Section 3, where E^I_{Sysres} and E^O_{Sysres}, E^I_{Crceng} and E^O_{Crceng}, and, $E^I_{Buf\_ctrl}$ and $E^O_{Buf\_ctrl}$ are the input and output interface behavior specifications of the $Sysres$, $Crceng$ and Buf_ctrl blocks, respectively. These specifications are written in Verilog and ACTL (subset of CTL without E path quantification). We verified a set of ACTL properties on this model using VIS [9]. For illustration purpose, we consider in this paper following simple property:

$$\varphi = AG(AF(cdat[7:0] = 0));$$

of Buf_ctrl block [1], which means that no error occurs. Here, we first check the initialization of the three blocks are correct, then we apply the following inference rule assuming there is at least one unit delay between the modules.

$$\begin{array}{rcll} Sysres & \Rightarrow & E^O_{Sysres} & (1)\\ Crceng \| E^I_{Crceng} & \Rightarrow & E^O_{Crceng} & (2)\\ Buf\_ctrl \| E^I_{Buf\_ctrl} & \Rightarrow & E^O_{Buf\_ctrl} & (3)\\ Buf\_ctrl \| E^I_{Buf\_ctrl} & \Rightarrow & \varphi & (4)\\ \hline Buf\_ctrl \| Sysres \| Crceng & \Rightarrow & \varphi & \end{array}$$

The verification results are shown in Table 1 including CPU time (Seconds), memory usage (Mega bytes) and the size of BDD nodes generated. This verification were conducted on a Sun Ultra2 workstation with 768MB memory.

Because the state space of decomposed modules are manageable, all verification succeed, hence proving equations (1) to (4) are sound. According to the composition theorem in Section 2, we can conclude that

$$Buf\_ctrl \| Sysres \| Crceng \Rightarrow \varphi$$

Namely, the BERMON system seen as the parallel composition $Buf\_ctrl \| Sysres \| Crceng$ satisfies property φ.

V. Conclusion

This paper exposes a compositional verification approach towards IP block-based designs. The approach is based on the ideas of decomposing a design and using induction over modules or over time. The abstract environment array is defined to describe the interactions of the

[1] Here, AG means "always in all paths"; AF means "eventually in all paths"

Equation	CPU time (s)	Memory usage (MB)	BDDs
$Sysres \Rightarrow E^O_{Sysres}$	0.1	0.24	21
$Crceng \,\|E^I_{Crceng} \Rightarrow E^O_{Crceng}$	252282.2	143	374575001
$Buf\_ctrl \,\|E^I_{Buf\_ctrl} \Rightarrow E^O_{Buf\_ctrl}$	233.9	124	2209944
$Buf\_ctrl\|E^I_{Buf\_ctrl} \Rightarrow \varphi$	114431.8	58	1572712983

TABLE I

Compositional verification of property φ

modules in the complete design and the compositional theorem is used to verify it. This approach is illustrated by the verification of the BERMON system which includes two IP blocks and a customer design module. The entire state space of the design is not tractable in the model checker, VIS. Throughout the analysis and the verification, we can see that compositional verification is a promising verification approach for IP based designs. Contrary to other traditional verification approaches such as simulation, this approach can derive a property of the large design from the description of modules. Moreover, it is practical because the interface behavior is easy to obtain in an IP block-based design. The proposed approach can be used in our case study because the BERMON system is actually a simple case. When the design to be verified becomes larger and many memory elements are involved in the design, further reduction and abstraction techniques have to be used to reduce the state space. We intend in our future work to investigate with our industrial partners whether the proposed approach can permit the verification of much more complicated designs.

Acknowledgments

We would like to thank J. M. Smith and I. Hadzic at University of Pennsylvania, for providing the original BERMON VHDL model. This work is partially supported by a Concordia University graduate student scholarship and an NSERC strategic research grant no. STP0201836.

References

[1] M. Abadi and L. Lamport. Conjoining specifications. *ACM Transactions on Programming Languages and Systems*, 17(3):507–534, May 1995.

[2] R. Alur and T. Henzinger. Reactive modules. In *Proc. 11th IEEE Symposium on Logic in Computer Science*, pages 207–218. IEEE Computer Society Press, 1996.

[3] R. K. Brayton et al. VIS: A system for verification and synthesis. In T. Henzinger and R. Alur, editors, *Eigth Conference on Computer Aided Verification (CAV'96)*, pages 428–432. Springer-Verlag, Rutgers University, 1996. LNCS 1102.

[4] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finitestate concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, January 1986.

[5] A.E.K. Dekdouk, O. A?t-Mohamed, S. Jahanpour, and E. Cerny. About formal verification of ip-based soc designs. In *International Workshop on IP Based Synthesis and System Design*, December 1998.

[6] I. Hadzic and J. M. Smith. P4: A platform for FPGA implementation of protocol boosters. In *Proc. 11th International Workshop on Field Programmable Logic and Applications (FPL'97)*, Sept. 1997.

[7] ITU-T. *Recommendation I.363 on ATM Adaptation Layers,*. International Telecommunication Union (ITU) Telecommunication Standardization Sector, Geneva, Switzerland, July 1992.

[8] K. L. McMillan. Verification of an implementation of Tomasulo's algorithm by compositional model checking. In *Proc. Computer Aided Verification (CAV'98)*, Vancouver, BC, Canada, June/July 1998.

[9] H. Peng and S. Tahar. Compositional verification of an ATM Bit Error Rate MONitor Circuit. Technical report, Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada, March 1999.