Hierarchical Approach for the Verification of an IEEE-754 Floating-Point Function

Amr T. Abdel-Hamid, Sofiène Tahar, John Harrison
Concordia University
Address: Electrical & Computer Engineering Department, 1455 Maisonneuvve, Montreal, Quebec. H3H 1MG.
Tel: (514) 848-3124   Email: at_abdel@ece.concordia.ca

Abstract. *In this work, we have formalized and verified a hardware implementation of the Table-Driven algorithm for the floating-point exponential function. We have used a hierarchical approach enabling the verification of this function from the gate level implementation up to a behavioral specification adapted from the high level algorithmic description written by Harrison [3].*

1. Introduction

The verification of floating-point circuits has always been an important part of processor verification. Floating-point algorithms are usually very complicated. They are composed of many modules where the smallest flaw in design or implementation can cause a very hard to discover bug, as happened in the Intel's case. Traditional approaches to verifying floating-point circuits are based on simulation. However, these approaches cannot exhaustively cover the input space of the circuits.

Working on the same design path of most electronic products, we will discuss in this paper the formalization and verification of the IEEE-754 table-driven exponential function in all abstraction levels of the design flow. We will start by verifying both RTL and gate level implementation of the IEEE-754 exponential function against a high level behavioral model specification developed before by [3] using the HOL theorem prover.

2. Modelling of the Exponential Function

From the beginning of the verification project, it was noticed that the specification of [3] is too flat for a lower level verification process. So, a new modular behavioral specification was introduced. For instance, we have divided this specification into six intermediate blocks (modules) where the conjunction of these blocks represents the full specification. Trying to achieve maximum modularity for the design, we have tried to minimize the interfaces between different modules. This helps us to divide the verification tasks into well-defined smaller ones. Each of these blocks was also divided into smaller specifications giving us smaller sub-specifications clearly related to the goals (theorems) needed to be proved.

We were faced with the same problem of the flatness in the VHDL implementation developed by *Bui et al.* [1]. The code was so flat as to make it nearly impossible to be modeled, let alone verified. So we had to make some design changes, which keep the same code properties but make the design easier to model and verify. We have aimed mainly in our changes to attack the following criteria:

Logic Complexity: Hierarchical designs reduce the logic complexity in the circuit. Also, in some modules the code could be changed to perform the same function, although it is less complex.

Verification time and effort: it is very hard to model and verify a very large flat design. Redesigning the VHDL code has saved a lot of time and effort needed in the verification process.

The overall verification process is now composed of four main tasks (Figure 1):

1. We prove the correctness of the modular behavioural specification against the flat specification adopted from the one provided in [3].
2. We prove that the modular RTL implementation implies the modular behavioural specification.
3. We prove hierarchically that the synthesized gate level implementation implies the modular RTL description.
4. We link the whole proof in a global one covering the whole design cycle of the floating-point exponential function.

Starting from the gate level implementation, we should hence be ending by the behavioral description of the function. These verification paths are shown in Figure 1, where the shaded boxes are the material provided by [3], and [1], while the white ones represent those developed in this work.

3. Verification of the Exponential Function

The next step is to verify these different levels using a hierarchical proof approach in HOL. This goal cannot be reached directly, due to the very high abstraction gap between the gate and behavioral levels as described above. So, the proof scheme was changed to hierarchically prove that the gate level implies the more abstract RTL. Then this RTL was related, by a formal proof, to a modular behavioral specification. The latter was proved to imply the high level flat behavioral specification. A summary of the verification times for the whole system is given in Table 1.

4. Conclusions

We have demonstrated the use of HOL to model the behavioral and RTL specifications for the IEEE-754 Table-Driven exponential function in a modular form, as well as the modeling of the gate level implementation of the same function. Using a hierarchical verification approach, we have been able to develop a formal proof indicating the correctness of the implementation using the HOL tool. The experimental results showed that other than the time needed to get acquainted with the tool, HOL is able to prove such very deep datapath circuits in a fraction of times of other tools would need.

References

[1] H. T. Bui, B. Khalaf, and S. Tahar, "Table-Driven Floating-Point Exponential Function", CCECE'99, May 1999.

[2] M.J.C. Gordon and T.F.Melham, "Introduction to HOL: a theorem proving environment for higher order logic", Cambridge University Press, 1993.

[3] J. R. Harrison, "Floating-point verification in HOL light: the exponential function", Technical Report number 428, University of Cambridge Computer Laboratory. UK, June 1997.

[4] P.T.P. Tang, "Table-Driven Implementation of the Exponential Function in IEEE Floating-Point Arithmetic", ACM Transactions on Mathematical Software, vol. 15, no. 2, 1989.

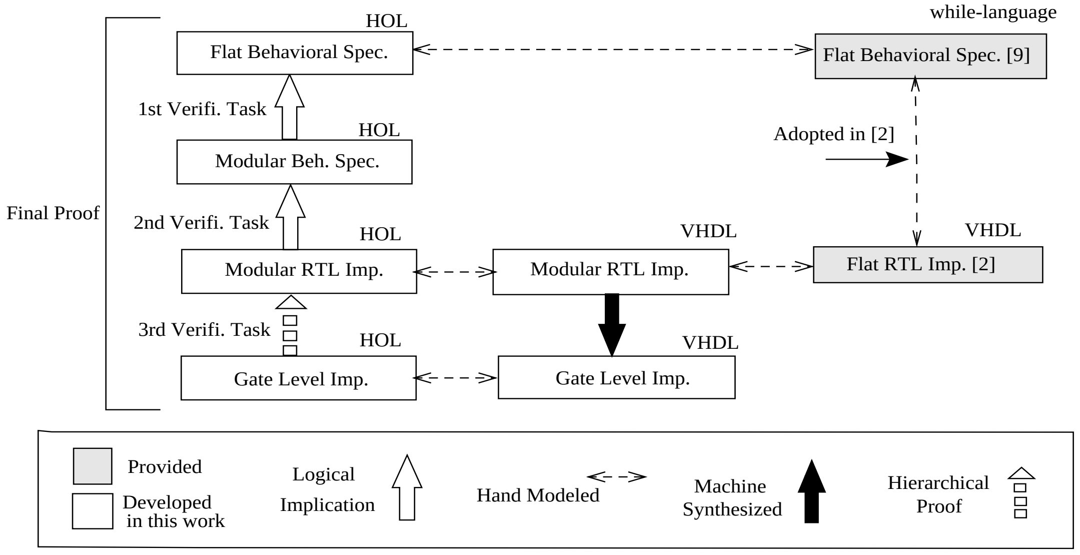

Figure 1. Verification Stages of the Exponential Function

Table 1. Verification Times of Different System Modules

Module Name	CPU Time (in Sec.)
Floating-Point Addition	60.500
Floating-Point Multiplication	30.540
M_J Module	2.120
R1_R1 Module	5.620
P_R Module	3.420
EXP_Cal Module	1.970
IEEE_EXP Module	5.290
Total Verification Time	**214.200**