

Porting EDIF netlists to the Viva Environment

Sreesa Akella

Duncan A. Buell

James P. Davis

Department of Computer Science and Engineering

University of South Carolina

Columbia, South Carolina 29208

akella@enr.sc.edu, buell@cse.sc.edu

April 23, 2003

Abstract

We describe a procedure to import an EDIF netlist to the Star Bridge Systems Viva design environment. The Viva tool is used for modeling and programming the Star Bridge Systems Hypercomputer system. The EDIF netlist is generated by synthesizing a functionally working VHDL model of a design. Importing EDIF into Viva would provide the ability to implement designs previously modeled in VHDL and to obtain reference points with regard to Viva-versus VHDL performance.

1 Introduction

The HDL-based design methodology is time-tested and provides a very stable design flow. A design can be modeled using any of the hardware description languages like VHDL or Verilog; for our purposes we look at designs modeled in VHDL. The model can be tested for correct functionality using simulation tools like ModelSim. Once the design functionality has been verified, the design can then be synthesized and an EDIF [3] netlist generated. The Star Bridge Systems Viva environment provides us the ability directly to implement this design by importing the EDIF netlist. This provides us the opportunity to compare the performance of VHDL-modeled designs against designs modeled directly in Viva. We describe here a procedure to import successfully an EDIF netlist into the VIVA environment.

2 Process Flow for importing EDIF to Viva

The process flow for importing EDIF to Viva is given in Figure 1. The design is modeled in VHDL. It is better if we have a hierarchical and modular description of a design instead of a single architecture describing the entire design, since a single architecture will lead to greater complexity in the EDIF netlist generated. The model is then verified for its functionality using the ModelSim simulation tool. The verified design can then be synthesized using any of several FPGA synthesis tools; we have used both the Synopsys FPGA Compiler and the Synplicity Synplify Pro tools [1], [2]. The synthesis is one of the important steps in the process, as it requires a certain tweaking of the synthesis options. The options that need to be set are given below.

Device : Virtex II XCV6000
Speed : -4
Package : FF1152
I/O insertion : Disable

The last option of I/O buffer insertion is important. If this option is not disabled, the synthesis tool would insert I/O buffers into the EDIF netlist, and these buffers would then have to be manually removed before the EDIF netlist could be imported into Viva. If this is not done, the Viva environment will generate Xilinx tool errors when the design is implemented.

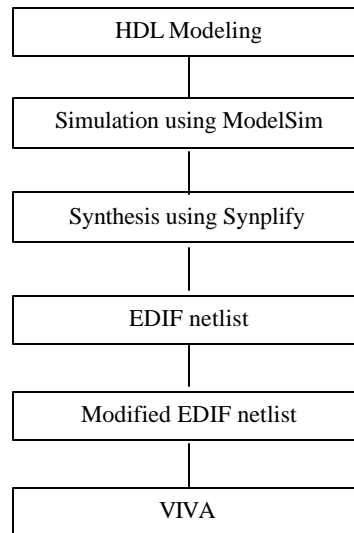


Figure 1. Process Flow for importing EDIF into VIVA

The EDIF netlist generated by this process cannot be directly imported into Viva; several modifications have to be made before the EDIF netlist becomes compatible with what Viva expects. These modifications include changes to library references and port references and exclusion of unnecessary library and cell definitions. We need to pick up the cell definitions of our main module along with related sub modules that are not already defined in Viva. The Viva tool references its basic components through a file named “FPGAStrings”; this file contains EDIF netlists of all the library components that are referenced by the objects in Viva. For importing our EDIF netlist we could place the main cell and related cell definitions of our design into the “FPGAStrings” file or alternatively have them put into a file that the Viva object can reference through an attribute setting. Once we have done this, we could then open the Viva tool and create the signature of our component. An important detail in this is that bit strings or bit vectors are not permitted as input or output; the input and output ports of the component must be only single bit data types. This component could then be converted into a primitive object by setting certain attributes. Once the object is created in Viva, it could be synthesized and its functionality checked. Finally, when the design has successfully been imported, the resulting module can be used as a library component for other designs just as if it were an object created entirely in Viva.

Though the EDIF porting of small designs, such as a 16-bit multiplier, is not a difficult task, larger designs pose problems because a great many modifications have to be made before the EDIF netlist is compatible with Viva. A better approach is to break up a hierarchical design, generate netlists piece by piece, import these individual sub module netlists into Viva, and generate the top design using these sub module objects. Even then, for large hierarchical designs, a number of individual netlists must be modified, and this number would increase as the complexity of the design increases. We have thus been developing an automated process for modifying EDIF netlists using scripts. These scripts written in Perl [4] would take the EDIF netlist and the FPGAStrings file as inputs and generate the modified EDIF netlist. The scripts would need to perform the following modifications:

- Eliminate unnecessary library definitions
- Eliminate cells already present in the FPGAStrings file
- Modify cell, library, view and port references

Using scripts such as these will make the porting of large designs into Viva much easier and EDIF porting in general much simpler. We will use as our test example the importing of the large design for an Elliptic Curve Cryptosystem and the comparison of its implementation in Viva. We will present performance data and compare it with the performance obtained by implementing designs modeled through Viva as well as present conclusions regarding the improvement of the scripts and the EDIF porting process.

References

- [1] Synopsys FPGA Compiler II™ User Guide. Version 2001.08-FC3.7, January 2002
- [2] Synplify Pro™ Reference Manual, October 2001.
- [3] Introduction to EDIF. www.edif.org.
- [4] Randal L. Schwartz, Tom Christiansen. *Learning Perl*. Second edition, O'Reilly publications, July 1997.