

From Statecharts to Hardware FPGA and ASIC Synthesis

Paul Clemente
Peter Runstadler
Larry Specter
Kerin Walsh

Raytheon Company
1001 Boston Post Rd.
Marlborough, MA 01752

Abstract

One of the current challenges in the design of digital systems is the transition from specification to hardware. Major advances in the development of requirement analysis tools in recent years have aided this transition. These advances coupled with the advent of logic synthesis tools have provided a strong foundation for development of a seamless, requirements driven design methodology that ensures specification compliance across all levels of design abstraction.

A methodology has been developed at Raytheon to ensure design integrity from specification to hardware implementation for the development of both Field Programmable Gate Arrays (FPGAs) and ASICs. This paper addresses the development of this methodology, which relies on the use of i-Logix's EXPRESS VHDL as the front end requirements analysis tool. Automatic generation of synthesizable RTL VHDL from EXPRESS VHDL's statechart notation enables the exploration of multiple synthesis paths. This paper addresses the exploration of these multiple synthesis paths and the various CAE tools, such as Synopsys, Exemplar, Xilinx, and Mentor, used to achieve the automatic generation of logic.

1.0 Introduction

The primary purpose of the first stages of system development must be to understand the problem and to communicate that understanding to others. This communication has traditionally been accomplished in the form of a written, textual specification. As today's electronic systems become more complex, however, the written specification fails to adequately meet these communication needs. Ambiguities or omissions in a textual requirements specification may not be uncovered until a system has been assembled and tested. In addition, different interpretations of a single requirement may result in designs that are viable by themselves, but that cannot be integrated successfully at the system level. Aside from textual specifications, a more powerful and efficient means of communication is via a graphical method. Recent advances in the CAE market have produced powerful tools that can be used at the front end for requirements analysis via behavioral modelling of the system under design. The power of these tools lies in their ability to transition to detailed design via the automatic generation of VHDL. These system design tools along with the maturation of logic synthesis tools enable development of a seamless design methodology from requirements to the generation of logic.

2.0 Requirements Driven Seamless Design Methodology

All system design methodologies begin with some knowledge of system requirements, i.e. what is the system supposed to do, how fast is it supposed to do it, etc. The requirements driven design methodology, unlike traditional system design methodologies, begins with the generation of an executable specification of the system requirements. An executable specification possesses two important characteristics: it is graphical and it is able to be simulated. A graphical representation of system requirements has powerful advantages over the traditional written specification. By virtue of its graphical nature, it is inherently more readable and more easily reviewable. Its executable feature provides a formal method for removing ambiguities and contradictions from the specification.

Generation of this executable specification is accomplished using i-Logix's EXPRESS VHDL. This interactive environment for system analysis allows creation of a graphical model of the system function and behavior which can be validated via model execution. This process of formal specification of system requirements and subsequent model validation creates an executable specification.

Raytheon's corporate CAD group (RAYCAD) initiated development of a cooperative working relationship with i-Logix, beginning with the development of the automatic VHDL generator from statecharts. This relationship has expanded to include RAYCAD and Division funded activities related to the design and development of an integrated EDA environment.

Systems are described with EXPRESS VHDL using three graphical languages: module-charts, activity-charts, and statecharts. The module-chart represents the structural view of the system. This chart is a hierarchical system block diagram that includes data flow between system components. The purpose of the activity-chart is to define system processes or functions. The activity-chart allows the user to define data and control that flows between the resident functions. It is the statechart view that defines the behavior of these processes or functions. Statecharts represent the modes, or states, that a system may be in. For example, a switch can be in the open or close state; it can not be in both states at the same time. Statecharts are an extension of state transition diagrams, with the added notions of hierarchy and concurrency.

Once statecharts are created to describe the system control, the statechart model may be validated using the model execution capability of EXPRESS. During model development, the system designer continually verifies that the statechart model reflects the true system requirements by performing a series of interactive simulations. Once model validation is achieved, test drivers may be created to simulate the application of test data. Test drivers consist of additional statecharts or simulation control programs(SCPs). The final result is an executable specification: a description of system requirements that can be simulated for system validation.

After model validation is achieved, EXPRESS VHDL provides for the automatic generation of VHDL from the system model. VHDL may be generated in either a behavioral or a register transfer level(RTL) form. The generation of behavioral VHDL may provide the framework for further system development via the enhancement of these behavioral models. The automatic generation of RTL VHDL enables the transition to detailed design via the use of logic synthesis. The synthesizable RTL VHDL may be used as input to one of several logic synthesis tools for the automatic generation of logic, targeted to a specific technology library. The technology independent nature of the EXPRESS VHDL model enables exploration of various target technologies at the synthesis phase via the automatic generation of vendor specific register transfer level(RTL) synthesizable VHDL. The statechart models serve as both implementation and foundry independent design models. As a result, quick turnaround implementations through automatic VHDL and synthesis may be explored.

The resulting netlist that is generated may then be passed to place and route tools either in the FPGA or ASIC environment depending on the target chosen technology.

3.0 Pathfinder as Proof of Seamless Design Methodology

In order to prove the Statecharts to FPGA seamless design methodology, a pathfinder was performed using a requirements specification for a Raytheon design as a starting point. The path to implementation consists of behavioral modelling using statecharts, automatic VHDL generation, gate level synthesis and FPGA place / route, configuration and test. Two different seamless design flows were investigated with this pathfinder, as indicated in Figures 3-1, 3-2.

The major difference between these design flows is the synthesis tools used. The methodology utilizing the Synopsys synthesis tool (Figure 3-1) also uses Xilinx DS343 Mentor interface package as an intermediate step between synthesis and the Xilinx tool set. This intermediate step provided a useful verification point for the pathfinder. The Exemplar based methodology as shown in Figure 3-2. This methodology outputs the synthesized gate level design directly to the Xilinx place and route tools via the Xilinx Netlist Format.

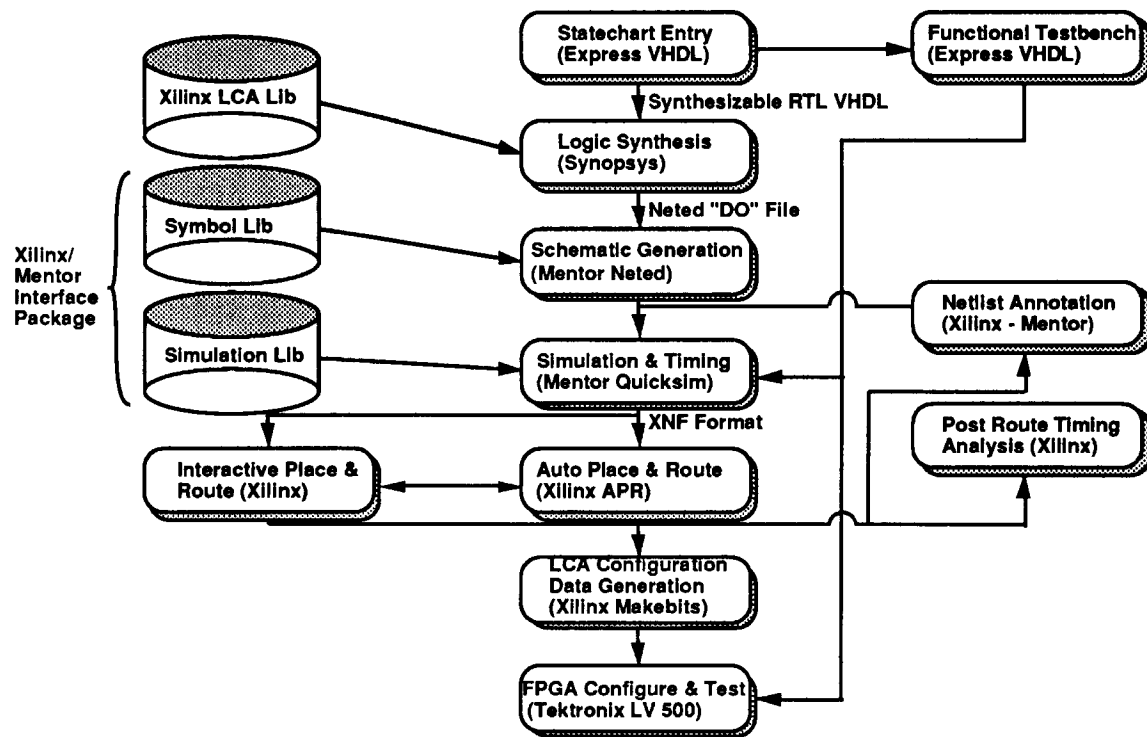


Figure 3-1: Seamless Methodology Using Synopsys Synthesis

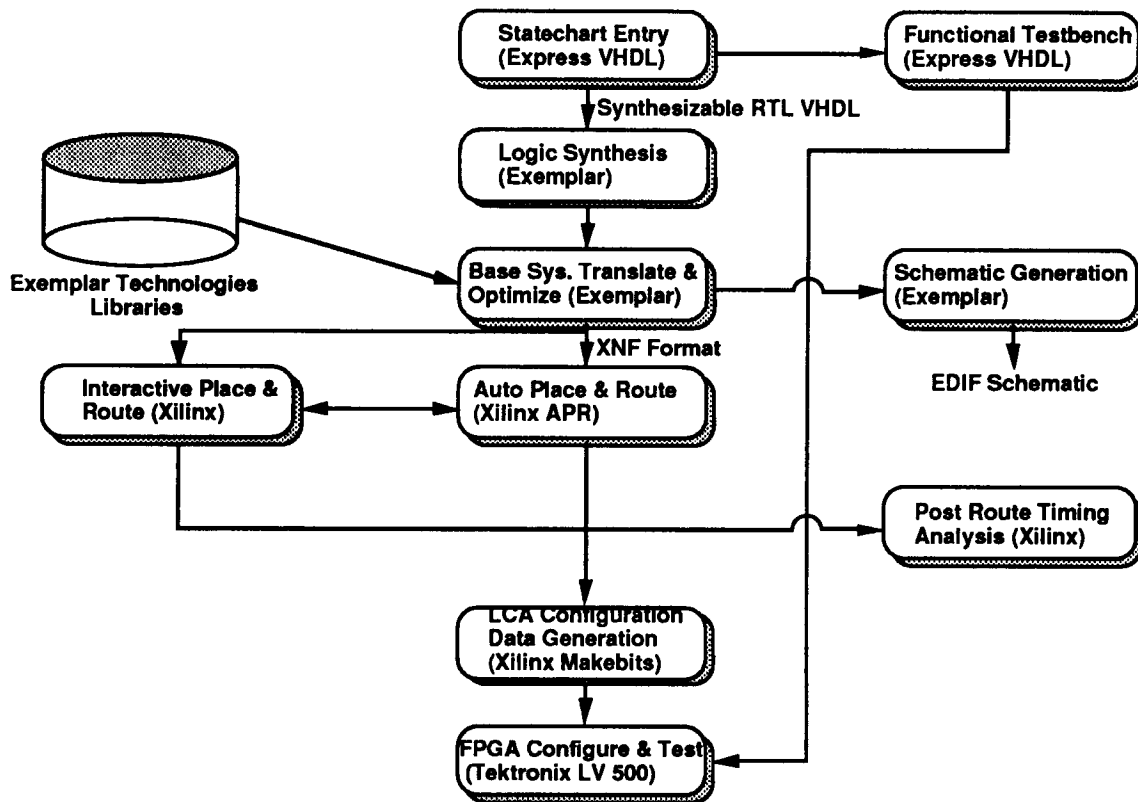


Figure 3-2: Seamless Methodology Using Exemplar Synthesis

3.1 Executable Specification

A written ASIC specification for a Raytheon design to be utilized in a military computer product was the requirements specification used to drive the pathfinder. This specification describes the requirements for a cpu to peripheral bus interface function. An implementation of this interface function would require equal portions of control and datapath logic with a gate complexity of roughly 8,000. The first step in the pathfinder was to use the written requirements specification to generate an executable specification for the IC.

The seamless design methodology was particularly applicable to this problem for several reasons. The requirements specification was in a state of flux and thus an ideal candidate for behavioral modelling using EXPRESS VHDL 's statechart notation. As changes to the specification were made the statecharts could be easily modified and validated. In addition, the implementation path and target technology had not been chosen. The intent was to implement the function as either an FPGA or an ASIC. Had a traditional design methodology been chosen, initial design work could not begin until both the implementation path and technology had been determined. The power of the seamless design methodology lies in it's ability to enable immediate design development despite lack of a completed specification, firm implementation path and target technology.

3.1.1 Statechart Modeling

The executable specification for the IC was created by modeling the written specification graphically using the Express VHDL tool from i-Logix. In addition to the model of the IC function, a watchdog was created to verify the IC function. This watchdog provided stimulus to the IC model and monitored the responses to verify every requirement in the written specification.

The modeling task took fifteen days with two modelers working in parallel. One engineer created the statechart model for the IC function while the second engineer created the watchdog also using statecharts. The next step was to verify the Statechart models.

3.1.2 Verification

The model debug and verification phase consisted of graphical simulation of the IC model using the watchdog in the Express VHDL analysis environment. Equal amounts of model errors and written specification ambiguities were discovered during these simulations. The model debug and resolution of all specification ambiguities was completed in twenty days.

In addition to the verification of the models, the simulations produced trace files (event log files) which describe every change of state of every element in the model. A post processor was written in C to use these trace files to generate simulation stimulus and expected results for the intermediate verification step with Mentor Quicksim (Section 3.2). The post processor was also used to generate test vectors to be run on the Tektronix LV500 tester during the FPGA configure and test (Section 3.4).

After the model was verified, a section of the model was chosen for the pathfinder in order to avoid complexity and size issues and concentrate on proving the seamless design flow from Statecharts to the FPGA synthesis.

3.1.3 VHDL Generation

In addition to proving the seamless methodology, choosing a piece of the IC function to implement was beneficial in that it demonstrated a methodology for implementing large functions in multiple devices (such as FPGA's). A large Statechart model which has been functionally verified with Express VHDL simulation can be used to generate multiple VHDL entities which can be independently implemented via VHDL synthesis. The partitioning takes place in the VHDL generation setup (compilation profile) and does not require any Statechart modifications. Any combination of states in the model can be combined to produce synthesizable VHDL.

There is, however, a restriction on Statechart modeling targeted for multiple device implementation. It is necessary that internal elements (not ports) of a particular device partition (group of states) be completely defined in that partition. For example, if a condition is assigned true in state A and false in state B, states A and B must be in the same partition.

The VHDL setup and compilation took several minutes for the group of states selected for FPGA implementation. The group of states chosen to be implemented in one FPGA for the pathfinder make up roughly 25% of the entire IC model. This subset of the IC function is called RAY1. The next step in the RAY1 implementation was to use the RTL VHDL generated by Express VHDL to synthesize a gate level design.

3.2 Synthesis

The synthesis stage of the seamless design methodology was investigated using both Exemplar and Synopsys. The Exemplar Logic Synthesis System (ELSS) is capable of producing a Xilinx Netlist Formatted (XNF) netlist which can be input directly to the Xilinx XACT software for place and route and configuration data generation. ELSS incompatibility with the Exemplar formatted VHDL generated by Express VHDL proved to be a problem and is being addressed by Exemplar. As a result of this problem the Exemplar path has not been taken through to completion at this time.

The Synopsys logic synthesizer did successfully generate a gate level implementation of the RAY1 function. Logic synthesis was accomplished by applying as input the Synopsys compatible VHDL generated by Express VHDL and a Synopsys VHDL package from i-Logix. The design was targeted to the Xilinx 3000 series library. The RAY1 synthesis with Synopsys, which included reading in the VHDL and compiling, took approximately two hours to complete on an Apollo 4500 workstation. Because Synopsys is not currently capable of generating XNF formatted netlists, an intermediate format for the gate level design was required to serve as a link to the Xilinx place and route tools.

This link was provided by the Xilinx DS343 Mentor interface package. The DS343 package provides a Xilinx 3K Series library for the Mentor Neted and Quicksim tools. The package also includes programs to convert to and from the Xilinx place and route tools. A RAY1 Mentor database was created using a Mentor Neted command script generated by Synopsys. The command script was generated in about five minutes by Synopsys and took approximately fifteen minutes to execute in Mentor Neted.

Slight manual intervention (seam in the methodology) was required at this point to add I/O pads to the Xilinx design using Mentor Neted. This intervention was required because I/O pad requirements are not modeled with the Statecharts. I/O pad selections are often driven by factors which are external to the model and synthesis. Alternatively I/O pads could be added as structural VHDL code by editing the VHDL output from EXPRESS VHDL.

The RAY1 design was then simulated using Mentor Quicksim as an additional step to ensure that the function modeled with the Express VHDL Statecharts was identical to that implemented in gates with Synopsys. The Quicksim stimulus and expected results were provided by the Express VHDL Simulator trace file post processor (Section 3.1.2). This additional simulation step is not required, and was performed in this pathfinder simply to identify possible problems early on should they exist (they did not).

The XNF netlist was generated from the RAY1 Mentor database in less than five minutes. The resulting netlist file was then used as the input to the Xilinx place and route tools.

3.3 Place and Route

The XNF netlist generated by the Xilinx Mentor Interface software was used as an input to the Xilinx CLB mapping tool, xnfmap. Xnfmap was run in order to map the gate level design synthesized by Synopsys into Xilinx 3K Series CLB's. A Xilinx XC3090 pin grid array Logic Cell Array (LCA) was chosen as the target device. This device is the largest of the 3K family and was required because of the I/O count in the RAY1 design (112 functional I/O). Xnfmap took three minutes to complete on a 386 PC platform with 6.5 Meg. of memory.

The Xilinx map2lca program was then run on the RAY1 mapping as the final step in the preparation for the Xilinx Automatic Place and Route tool. Map2lca took two minutes to complete.

The Xilinx Automatic Place and Route (APR) tool was then run on RAY1 design using a manually generated constraints file to specify pinout. The APR took six hours and seventeen minutes to complete. Of this time, six hours and six minutes were spent on placement and eleven minutes was spent on routing. The placed and routed design contained 311 Configurable Logic Blocks (CLB's), 112 Input/Output blocks (IOB's), and 354 nets.

3.4 Configuration and Test

Configuration and test of the Xilinx 3090 LCA was performed with the use of a Tektronix LV500 Tester. The LV500 was chosen because it is capable of configuring as well as functionally testing the LCA device without any additional hardware. The LV500 Quick Connect PGA DUT card was used which includes a pre-wired 19 by 19 PGA ZIF socket.

Configuration data for the RAY1 design was generated by the Xilinx makebits software using the rawbits option to create an ASCII configuration bit stream file. A simple post-processor was then written to convert the makebits output to the Tektronix Ewav event format which included the clock, reset, and data inputs necessary for configuration of the Xilinx part in Slave Mode.

As a result of this translation software, the log files from the Mentor Quicksim simulations were easily used for functional verification of the RAY1 Xilinx implementation. The LV500 contains software to translate the Mentor log file event format to the Tektronix Ewav event format. Alternatively, the trace output from the Express VHDL Statechart simulation could be post processed to the Tektronix Ewav format.

4.0 Conclusions

In the quest to transition from specification to hardware, the Requirements Driven Design Methodology offers an integrated top down solution to the generation of FPGAs and ASICs. Our experience has shown that this methodology works best for control oriented designs or designs that have control associated with datapath. An integral part of this methodology is the common stimulus used at both the statechart level and the hardware verification level. This provides for direct traceability and validation of the hardware to the executable specification. It offers a seamless methodology with the exception of the manual I/O pad instantiation. Our pathfinder has shown the engineering effort is focused mainly in the generation and validation of the executable specification which is technology independent. The technology mapping phase, logic synthesis and below, only takes a relatively short amount of time, in this case eleven hours. This Requirements Driven Design Methodology provides for an implementation independent executable specification to be targeted or retargeted to multiple technologies.