A System for Behavior Extraction from FPGA Implementations of Synchronous Designs

Mahesh Mehendale
Texas Instruments (India) Pvt. Ltd.,
71 Miller Road, Bangalore, INDIA 560 052

Abstract

The paper presents a system for extracting behavioral representations from FPGA (Field Programmable Gate Array) implementations of synchronous sequential designs. Behavior extraction offers advantages in the areas of simulation, verification, optimization and technology migration. A generic FPGA architecture has been defined. Necessary parameters can be specified to derive the desired architecture. The architecture details and the fuse-map describing the design implementation, form the inputs to the system. The behavior is extracted at two levels of abstractions, 1. boolean equations for the combinational logic and 2. finite-state-machine. The overall extraction flow with details of the different steps in the flow are presented. The results for six designs are presented. They clearly demonstrate the importance and usefulness of behavior extraction, especially for technology migration from FPGAs to the mask programmable gate arrays.

1 Introduction

Field Programmable Gate Arrays (FPGAs) combine the best features of PLDs and mask programmable gate arrays, by providing gate-array type architectures and field programmable interconnect schemes.[1] More and more designers are beginning to use FPGAs for quick prototyping and also for low volume production. A typical design flow involves implementing a system level behavior description in terms one or more FPGAs. The process of translating behavior into logic is not fully automatic. In a possible design scenario, the design is realized in terms of FPGA macro functions and the netlist is partitioned into a set of FPGAs based on their capacity. Thus a behavioral model for each FPGA may not always be available. Any changes to specifications or detection of bugs while at or after the logic design stage, get implemented at the logic netlist level and are not always reflected in the behavior. Extracting behavior from the FPGA implementations of the design offer significant advantages. They are in the areas of simulation, verification, optimization and technology migration.

The system described in this paper provides extracting behavior at two levels of abstractions. The first level is in the form of boolean equations which represent design outputs and sequential logic inputs, in terms of design inputs and sequential logic outputs. The second level of abstraction is in the form of a finite state machine and is extracted from the first level behavior representation.

The system works off a generic FPGA architecture. Necessary parameters can be specified to derive the desired architecture. The paper describes the FPGA architecture, the overall behavior extraction flow and the details of various stages in the flow. Technology migration results via behavior extracted at both the levels of abstraction are presented.

2. FPGA Architecture

The architecture used in the system consists of rows of logic modules, with horizontal routing tracks between the rows of modules. Vertical tracks connect the inputs and outputs of the logic modules to the horizontal tracks via one-time programmable antifuses located at their intersections. This approach has been used in FPGAs from Actel[2] and QuickLogic[3]. The architecture supports two types of logic modules, i. fully combinational and ii. sequential. The combinational logic module implements an N input, one output function. The sequential logic module consists of a combinational logic driving a D flip-flop. It has a control signal which enables bypassing the D flip-flop, so that if needed the sequential logic module can be used to provide combinational functions. In a case where no combinational logic exists, the sequential logic module maps onto a simple D flip-flop.

The target architecture specified in terms of the equations of the logic module and the fuse-map representing the connectivity form the input to the system.

3. Behavior Extraction Flow

Figure 1 shows the overall flow for behavior extraction. In the Function Extraction phase the specific functions, such as 2 input AND, 3 input NOR, 2:1 multiplexor etc., realized by each of the logic modules are derived. The system uses a BDD (Binary Decision Diagram) package for function extraction and also later to extract behavior in terms of boolean equations, In the Netlist Extraction phase, the fuse-map information is used to derive the connectivity of the logic modules. Net delays are calculated from the routing information...
and the necessary process parameters such as antifuse capacitance, capacitance per unit area of the routing tracks etc. The circuit is then partitioned into combinational and sequential blocks. Level-1 Extraction phase derives boolean equations for the combinational block and also calculates delays for each of the outputs from the logic block inputs. VHDL models can be generated from the behavior extracted at this level of abstraction.

![Figure 1: Behavior Extraction Flow](image)

In the Level 2 Extraction process, the logic netlist is partitioned and mapped onto a Mealy representation of the Finite State Machine (figure 2). The outputs of the D flip-flops are mapped on to the "states" of the finite state machine. The boolean equations are used along with the "state" definitions to generate an exhaustive state table. This table is minimized, redundant states are eliminated to get the finite state machine representation of the design. Corresponding RTL or VHDL models can be generated from this representation.

![Figure 2: Partitioning into a Mealy FSM representation](image)

### 4. Results

Table 1 gives the results for 6 MCNC benchmark designs. FPGA architecture used to implement these designs has alternating combinational and sequential logic modules in its rows. The functionality of the combinational logic module is the same as that of Actel-1010 logic module[4]. The sequential logic module consists of Actel-1010 combinational block feeding a D flip-flop. The details of the extracted state-machines are given. The designs were migrated to TGC100[5] gate-array library by re-synthesizing the extracted behaviors at both the levels of abstraction.

The results show that "FPGA design -> behavior extraction-> FSM -> re-synthesis -> GA design" path provides more area efficient solutions for migration of designs from FPGA to mask programmable gate array. The main reason for this is that in most cases, for an N state FSM, using Log2N flip-flops gives the best results for gate arrays, but using close to N flip-flops gives best results for FPGAs. The migration through FSMs enables mapping an N flip-flop design into Log2N flip-flop design, thus providing global optimization.

### Table 1

<table>
<thead>
<tr>
<th>Design</th>
<th>FPGA area #LMs</th>
<th>Extracted FSM data #in,#out,#states</th>
<th>TGC100 area level 1 migration</th>
<th>TGC100 area FSM migration</th>
</tr>
</thead>
<tbody>
<tr>
<td>bbse</td>
<td>50</td>
<td>7,7,16</td>
<td>183</td>
<td>131</td>
</tr>
<tr>
<td>mark1</td>
<td>50</td>
<td>5,16,15</td>
<td>157</td>
<td>128</td>
</tr>
<tr>
<td>ex2</td>
<td>55</td>
<td>2,19</td>
<td>187</td>
<td>146</td>
</tr>
<tr>
<td>donfie</td>
<td>79</td>
<td>2,1,24</td>
<td>234</td>
<td>193</td>
</tr>
<tr>
<td>styr</td>
<td>208</td>
<td>9,10,30</td>
<td>464</td>
<td>450</td>
</tr>
<tr>
<td>sand</td>
<td>218</td>
<td>11,9,32</td>
<td>481</td>
<td>460</td>
</tr>
</tbody>
</table>

### 5. Conclusion

Behavior extraction for FPGA based designs, provides significant benefits in the areas of system level simulation, verification, optimization and technology migration. The system described in this paper extracts the behavior at two levels of abstractions, level 1 : boolean equations for combinational logic of the design and level 2: finite state machine – a higher level abstraction extracted from level 1 behavior. A generic antifuse based channelled architecture for FPGAs is proposed. Necessary parameters can be specified to derive the desired architecture from the generic architecture. The results on six designs run through "FPGA design -> behavior extraction -> FSM -> re-synthesis -> GA design" flow, show that the FSM level behavior extraction is key to the efficient technology migration.

### References