Optimization of Module Placement during FPGA Reconfiguration

Introduction

Partial reconfiguration is an option available on the more current models of field programmable logic arrays (FPGAs) which enables them to have more versatility. The previous models only supported static programming – once the board is programmed, it stays constant and cannot be modified midway through execution. Thus, the previous models can only perform one task at a time; to perform a different task, the board must be completely reprogrammed before continuing. With partial reconfiguration, however, the concept of dynamic programming is introduced – part of the board's programming can be changed during execution while other parts remain untouched. The dynamic programming is done by streaming bits and overwriting a subsection (a set of columns) of the board. This then allows a single FPGA board to support multiple tasks without interruption. This situation proves useful when sequential tasks use the same modules.

The Work

I will analyze the relationship between different components and the space it occupies on various Virtex-4™ FPGA boards. For implementation and synthesis, I will be using the Xilinx® ISE v.7.1 software package. The following describes the procedure in detail:

I) Optimal sizing of individual components

Individual components are single logic blocks or modules. In this project, we are considering (but not limited to) the following modules generated using CORE Generator™ (part of Xilinx®): adder, multiplier, and divider. Using empirical data gathered from the standalone module syntheses, we can determine the smallest amount of space each module occupies and subsequently infer mathematical models between the bit-width and space needed for each different type of module. Note that the optimal sizing of the component will be restricted by a time constraint – the module cannot perform over a certain time limit (10 ns) to be considered functional.

II) Optimal sizing of merged components

Once the optimal sizes of the individual components have been determined, they can be combined into a single "super-module". The smallest space available needed for this can then be found empirically. We can then compare this data from **I)** and see if the results are consistent.

III) Design of wrapper-based models for individual components

A wrapper consists of the external wires of a given module. It is normally placed on one (or if needed, two) sides of the module. This is considered overhead and routes all incoming and outgoing signals through that specific space. Using the modules from **I)**, we create wrappers and determine the least amount of space required. The goal of this stage is to determine a wire-to-length ratio. We can then use the correlation between the merged and individual modules to better judge the optimal placement and merging of components.

IV) Design of wrapped-based models for merged components

Following the idea of **III)**, we then create wrappers for merged modules from **II)**. We can then compare the results to **III)**'s and see what the relationship is between individual components with wrappers and merged modules with wrappers. As stated in III), we are ultimately looking for a wire-to-length ratio. As part of the optimization process, we will also look at bit streams needed to reconfigure the same modules but in different arrangements.