MONTAGE: An FPGA for Synchronous and Asynchronous Circuits

Scott Hauck, Gaetano Borriello, Steven Burns, Carl Ebeling
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195

Abstract. Field-programmable gate arrays are frequently used to implement system interfaces and glue logic. However, there has been little attention given to the special problems of these types of circuits in FPGA architectures. In this paper we describe Montage, a Triptych-based FPGA designed for implementing asynchronous logic and interfacing separately-clocked synchronous circuits. Asynchronous circuits have different requirements than synchronous circuits, which make standard FPGAs unusable for asynchronous applications. At the same time, many asynchronous design methodologies allow components with greatly different performance to be substituted for one another, making a design environment which migrates between FPGA, MPGA, and semi-custom implementations very attractive. Similar problems also exist for interfacing separately-clocked synchronous circuits. We discuss these problems, and demonstrate how the Montage FPGA satisfies the demands of these classes of circuits.

1 Introduction

Field-programmable gate arrays provide an ideal implementation medium for system interface and glue logic. They integrate large amounts of random logic and simple data paths, and can be easily reprogrammed to reflect changes in system components. Unfortunately, most of the effort in designing FPGA architectures has ignored the special problems of these types of circuitry. Interface and glue logic require support for interfacing asynchronous logic to synchronous logic, interconnecting separately-clocked synchronous components, and controlling certain circuit delays [1], all of which are largely ignored by current architectures.

Asynchronous circuits are also not well served by current FPGAs. Implementations of asynchronous logic must consider hazards in the logic, synchronization and arbitration of events, and strict adherence to the timing assumptions of the design methodologies [5, 6]. Unfortunately, it is not possible to implement these circuits in a robust manner in current FPGAs. Some of the elements required (most importantly, arbiters that resolve conflicts between two concurrently arriving signals) are not implementable in the standard digital logic found in these devices. In addition, the logic and routing elements must be designed more carefully in order to avoid extra "glitches" on lines, since in asynchronous circuits
every transition is important. Finally, routing resources must have predictable, optimizeable delays in order to meet timing assumptions in the design methodologies.

However, the problems are not restricted simply to the architectures themselves. The mapping software also must be altered to handle the demands of asynchronous logic. Primarily, there are much stricter timing demands that must be upheld. Bundled data and isochronic forks both require signals to be routed with special delay demands, demands that impact placement of logic cells as well. Also, the logic decomposition used to break logic blocks down to the size required by an FPGA (the covering problem) cannot simply use the algorithms for synchronous logic. For quasi-delay insensitive circuits, where the only timing assumptions made are those of isochronic forks, standard synchronous logic decomposition techniques can add extra levels of logic incorrectly, putting hazards into the circuit.

Related work [2] has looked at mapping asynchronous circuits to Actel, antifuse-based, programmable gate arrays. Although the paper outlined an implementation strategy based on a fairly rich library of macro blocks, the underlying limitations of the Actel parts made it difficult to handle arbitration and synchronization, and the complex routing structure did not allow adequate control of routing delays. In addition, we feel that any antifuse-based FPGA is less desirable for asynchronous circuits because of the strict assumptions that must be made about circuit delays. These may require chip delay testing, which is impossible in an unprogrammed antifuse system as path delays cannot be measured until after programming.

2 The Architecture

The Montage FPGA is a version of the Triptych architecture designed to handle synchronous interface and asynchronous circuits. Since much of Montage is identical to Triptych, we direct readers unfamiliar with this architecture to [3]. Like Triptych, Montage is an electrically reconfigurable FPGA, which is preferable to antifuse-based FPGAs because it allows the chip to be programmed for delay testing without permanently configuring it.

The Montage global routing structure is identical to the Triptych routing structure, with diagonal connections between local cells, augmented with vertical segmented channels (see Fig. 1). This structure has proven to be effective for mapping general synchronous circuits. It is even better suited to asynchronous circuits, where one expects to find much more tightly connected subcircuits, and in general less random global routing. Also shared with Triptych is the general philosophy of allowing mappings to fix the tradeoff between logic and routing resources by having logic blocks capable of performing routing functions.

A Montage RLB (shown in Fig. 2) has three inputs and three outputs, and a functional unit (FU) which operates on the three inputs. There are two different types of functional units. The first is a logic block, which implements logic functions and stateholding elements. As shown in Fig. 3, the logic block has a