Self-Timed FPGA Systems

Rob Payne
rep@dcs.ed.ac.uk

Department of Computer Science, the University of Edinburgh

Abstract. Recently, there has been a renewal of interest in self-timed systems, due to their modularity, robustness, low-power consumption and average-case performance. Additionally, this paper argues that there are specific benefits to adopting self-timed design for FPGAs. The mapping problems of placement, routing and partitioning are simplified by not having a global clock constraint to meet, so more mappings are available for mapping algorithms to choose from. Hence, there is greater potential for algorithms to improve utilisation and performance of a design, or instead, to increase design turn-around by taking less time to produce a mapping. Furthermore, the ability to perform mappings quickly enables new FPGA applications where the mapping to the FPGA is done on-the-fly. However, currently available FPGAs provide no support for self-timed design. The latter half of the paper describes the STACC architecture, an FPGA architecture targeted at the implementation of self-timed bundled-data systems.

1 Introduction

All current commercial FPGAs are designed for implementing systems synchronously, so have dedicated clock signals. However, the alternative approach of building systems in an asynchronous or self-timed fashion has many potential benefits. This paper examines the case for self-timed FPGA systems and introduces STACC, a dedicated self-timed FPGA architecture.

Section 2 introduces self-timed systems and examines the general advantages of building systems asynchronously. In section 3, the specific benefits of building self-timed FPGA-based systems are considered. Section 4 reviews the current work on self-timed FPGAs and argues that current FPGA architectures do not support self-timed designs. In section 5, the STACC architecture is introduced, concentrating on the development of a timing cell for the architecture, from an unreconfigurable structures to a fully reconfigurable timing cell. Finally, section 6 summarises the paper and looks at future directions for the work.
2 Self-Timed Systems

In synchronous systems (Fig. 1, left), all modules are synchronised through a global clock signal. The clock places a global constraint on the system: all the modules within the system must have their data ready by the next tick of the clock. In asynchronous systems there is no global synchronisation of the modules within the system, but modules do synchronise locally through their communication protocols. Asynchronous communication protocols that use some form of handshake between sender and receiver (Fig. 1, right) are known as self-timed (for a formal definition see [1]).

![Fig. 1. Synchronous and self-timed protocols](image)

Figure 2 shows two common self-timed communication protocols. Both protocols come in two-phase and four-phase variants that attach different significance to transitions within the request/acknowledge handshake. In two-phase or event-based signalling, all transitions are significant. In four-phase signalling, only positive-going transitions are significant, so an idle return-to-zero transition is required.

In the bundled-data protocol (Fig. 2, left), a transition on the request signal indicates that the data is valid. The requirement that the request transition occurs after the data is stable is known as the bundling con-