Available online at www.sciencedirect.com

SCIENCE DIRECT®

Microprocessors and Microsystems 30 (2006) 388–397

MICROPROCESSORS AND MICROSYSTEMS

www.elsevier.com/locate/micpro

Dynamic clock-frequencies for FPGAs

J.A. Bower [a,*], W. Luk [a], O. Mencer [a], M.J. Flynn [b], M. Morf [b]

[a] Department of Computing, Imperial College, 180 Queen's Gate, London SW7 2BZ, UK
[b] Computer Systems Laboratory, Department of Electrical Engineering Stanford, CA 94305, USA

Available online 28 February 2006

Abstract

Most FPGA designs run at a fixed clock-frequency determined through static analysis in FPGA vendor supplied tools. Such a clocking strategy cannot take advantage of the full run-time potential of an application running on a specific device and in a specific operating environment. This paper describes methods for using dynamic clock-frequencies to overcome this limitation. We begin by describing a methodology for designing systems which allow dynamic clock-frequencies in FPGAs. We then present a framework for exploring the dynamic behaviour of suitable clock-frequencies for a number of FPGA applications in varied operational environments. Finally we introduce our AutoTEA system, which automatically adds circuitry to arbitrary FPGA designs for dynamically adjusting clock-frequency to a safe limit given current operating conditions. Our results show that dynamically clocking designs can lead to a speed improvement of 33–86% compared to using a fixed, statically estimated clock.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Timing analysis; Better than worst-case performance; Over-clocking; Power-saving; High-performance computation

1. Introduction

FPGAs enable the implementation of adaptive high-performance applications. Such custom hardware designs require a custom clock-frequency which balances performance with reliable circuit operation.

In a traditional FPGA design flow, clock-frequency is determined through static analysis of netlists for a device performed by software. After place-and-route a timing analyser program locates the longest combinatorial path between RAMs, I/Os and flip-flops [1]. This path, termed the 'critical-path', is the main bottleneck of the system and as such a design's clock-frequency can be derived from this.

We identify two sources of wasted performance with a statically determined clock-frequency. First, the manufacturing process for FPGAs is not perfectly uniform, and different physical devices have different characteristics. Second, the actual propagation delay through any path in a device changes during operation. Such changes are caused either by varying environmental conditions or even internal temperature changes due to different inputs varying power consumption.

Static timing tools deal with the above issues by employing worst-case models for estimating delays in hardware designs – leading to conservative clock frequencies. Another problem with using a fixed, statically determined clock-frequency is that all possible operating environments must be supported, however in some situations the environment may exceed conservative error margins. For example, a battery powered device running low on power and suddenly placed into a hot environment may fail although sufficient power would have been available to continue processing at a lower rate.

Dynamic clock-frequency schemes are alternatives to static estimates and have the potential to improve both high-performance and low-power applications. High-performance applications can run at the maximum physically attainable speed, while low-power applications can in general manage power-consumption by optimally balancing

[*] Corresponding author.
E-mail addresses: Jacob.bower@imperial.ac.uk (J.A. Bower), w.luk@imperial.ac.uk (W. Luk), o.mencer@imperial.ac.uk (O. Mencer), flynn@ee.stanford.edu (M.J. Flynn), morf@snow.stanford.edu (M. Morf).

0141-9331/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2006.02.006

clock-frequency, voltage supply, and demand for computation.

This paper contributes towards realising practical FPGA-based systems with dynamic clock-frequency. In particular, we provide:

- A methodology for creating dynamic clock-frequency systems which provide user confidence in correct operation.
- LIMIT: A hardware framework and experiments for exploring the behaviour of maximal safe dynamic clock-frequencies in FPGA designs.
- AutoTEA: An automated implementation of a technique for dynamically adjusting clock-frequencies to their optimal value for arbitrary FPGA designs.
- Experimental results from our LIMIT and AutoTEA systems applied to a diverse set of FPGA applications under varied environmental conditions.

The remainder of this paper is organised as follows. In Section 2 we review related work. We present our methodology for designing systems which implement dynamic clock frequencies in Section 3. In Section 4 we present our LIMIT and AutoTEA systems for implementing and evaluating dynamic clock-frequencies. Finally, in Sections 5 and 6 we present results of our experiments with AutoTEA and LIMIT, and our conclusions.

2. Previous work

Dynamic clock-frequency is already common in modern microprocessors and high-end ASICs [2]. “Over-clocking” in such systems pushes clock-frequency beyond vendor specifications [3]. While such a manual and brute-force approach is not suitable for serious computing systems, research in this area aims to develop solutions that reliably and dynamically adapt clock-frequency to optimal limits. In this paper, we limit ourselves to a discussion of dynamic clock-frequency systems for FPGAs.

We categorise some examples of this work into error tolerating and error avoiding systems and compare them to our own work in Table 1. From the table, three systems use error detection and correction techniques to tolerate errors in over clocked logic: TIMERTOL [4], Razor [5,12] and DIVA [6]. In the TIMERTOL and Razor systems, errors are detected by sampling inputs to pipeline register stages at two different times. The idea is that early samples can be used to continue processing and later outputs, which are more stable, can be used to detect if the early samples are erroneous. DIVA describes a microprocessor with a combined system of simple checker logic and a complex processing core. The simplicity of the checker logic allows it to be aggressively optimised for high-speed operation. In this system the complex processing logic is over-clocked using the high-speed checker to catch all errors. The common idea in all these schemes is to allow logic to run over-clocked, and check for errors to prevent committing erroneous outputs. Clock-frequency changes dynamically to minimise error rates.

Two systems which avoid errors by continuously re-evaluating and adapting clock-frequency to maintain correct functionality are: a self-timed PIC16C57 compatible microprocessor [7] and the TEAtime system [8]. In the PIC16C57 microprocessor, execution is paused while worst-case inputs exercise the system critical-path and the results are checked for errors. Clock-frequency is adapted to eliminate errors in the critical-path. TEAtime applies a similar idea, except that it allows continuous operation of a microprocessor design by creating a duplicate critical-path for checking. This duplicate (or false) critical-path is a one-bit wide version of the longest flip-flop-to-flip-flop path in the main design with additional delay. The idea is that the false critical-path, with its extra delay, will fail before the main design so clock-frequency can be adjusted based on observing errors in the false critical-path.

Other related research areas include: implementing low temperature designs, designing for “average case performance”, dynamic voltage scaling (DVS) and adapting clock-frequency to computation. The idea of designing for average case performance is to create hardware which can achieve higher clock-frequencies when inputs are “average case”, and scaling the clock during worst-case inputs [9]. Designing for low temperature enables higher clock frequencies. A novel method for lowering temperatures in FPGAs is to use dynamic reconfiguration to prevent single areas of a chip from getting too hot [10]. Schemes for adapting clock-frequency to specific computation have also been developed with clock period altered each cycle depending on which units are currently active [11]. Dynamic clock-frequencies are also applicable to systems implementing DVS [14] as they allow frequency to be optimally tailored to match the dynamic voltage.

Table 1
Comparison of our AutoTEA system to related efforts

	AutoTEA (ours)	TIMERTOL [4]	TEATime [8]	DIVA [6]	Razor [5]	PIC16C57 [7]
Error avoidance/tolerance	Avoidance	Tolerance	Avoidance	Tolerance	Tolerance	Avoidance
Overhead scales with design size	No	Yes	No	Yes	Yes	No
Automated implementation	Yes	No	No	No	No	No
Potential for automation	High	Medium	High	Low	Medium	Low
Prototype technology	FPGA	FPGA	FPGA	ASIC	ASIC [12]	ASIC