OpenSPARC Processor Evaluation Using Virtex-5 FPGA and High Performance Embedded Computing (HPEC) Benchmark Suite

Khaldoon Moosa Mhaidat, Jordan University of Science and Technology, Irbid, Jordan
Ahmad Baset, Jordan University of Science and Technology, Irbid, Jordan
Osama Al-Khaleel, Jordan University of Science and Technology, Irbid, Jordan

ABSTRACT

OpenSPARC is the only 64-bit Chip Multi-Threaded (CMT) processor that has ever been made open-source and non-proprietary. In this paper, the authors present an FPGA-based embedded system and methodology for prototyping and validating the OpenSPARC processor. They also present synthesis and performance evaluation results for OpenSPARC on a Virtex-5 FPGA platform. Light version of OpenSolaris was successfully booted on the platform, and the High Performance Embedded Computing (HPEC) benchmark suite was used to evaluate the performance. Xilinx ISE suite was used for synthesis, implementation, and chip programming. The down-scaled FPGA implementation of the processor runs at 81.3 MHz. The whole processor would require about 176453 Virtex-5 logic slices. To the best of their knowledge, they are the first researchers to report detailed FPGA synthesis results for OpenSPARC and evaluate its performance on FPGA using the HPEC benchmarks. Other researchers may find these results useful when comparing with other processors or studying the impact of a certain design change or addition on performance and cost.

Keywords: ASIC, Chip Multithreading (CMT), FPGA, High Performance Embedded Computing (HPEC), MicroBlaze, OpenSolaris, OpenSPARC, Prototyping, SoC, VLSI

INTRODUCTION

A computer program is either using the processor for computation or accessing data in the memory, disk, or I/O system. Due to the low speed of memory, disk, and I/O system when compared with CPU processing speed, CPU will be idle for long time if not utilized in an efficient way. Because of this speed gap problem, processor architects and designers have tried to come up with many techniques to increase the overall system performance and throughput.

One of the techniques is to allow multiple instructions of the same program (or thread) to run simultaneously to achieve higher re-
source utilization and better performance and throughput. This is known as instruction level parallelism (ILP). Exploiting ILP only is not enough because it is limited by the degree of parallelism in the same thread which is typically low or hard to exploit (Weaver, 2008; Horowitz & Dally, 2004). That is why processor designers have come up with techniques to achieve better utilization of the hardware resources and thus better performance and throughput by exploiting parallelism among threads. This is called thread-level parallelism (TLP) and the processor that can handle multi-threading is described as hardware multithreaded (HMT) (Weaver, 2008). To handle multiple threads, an HMT processor needs to be able to save the state of each thread including the register file, the program counter, the page table, ALU flags, and any other necessary status information in case of a context switch.

HMT however increases the complexity of the pipeline design since it needs to handle multiple threads running on the same core simultaneously and this in turn limits the core frequency and increases power consumption and heat dissipation. Moreover, debugging a complex core would be difficult and expensive to do. The advances in CMOS technology have made it possible for designers to combine multiple simpler single-thread processors into a single die. This approach is known as chip multiprocessing (CMP) (Weaver, 2008). Each processor (called core) inside the CMP has its own execution pipeline that supports only one thread. CMP has the problem that cores may not be utilized efficiently because of the time thread may waste waiting for memory or I/O access to complete. Moreover, cores may need to shares some expensive resources such as the L2 or L3 cache.

To enhance the utilization of the CMP cores, designers proposed to have multi-threading support into every core inside the CMP processor. This approach is known as Chip Multi-Threading (CMT) (Weaver, 2008). Figure 1 depicts the CMP, HMT, and CMT approaches. In the figure, a rectangle with the letter C represents computing time and with M represents memory latency.

In 2006, Sun Microsystems has made its SPARC processor fully open-source and non-proprietary and named it OpenSPARC. Until today, OpenSPARC is still the 64-bit only chip multi-threaded (CMT) processor that has ever been made open-source and non-proprietary. This open-source hardware release is also supported with open-source software, hypervisors, operating systems, and performance, modeling, and analysis tools that researchers can freely use and modify to suite their own interests or needs. There are two releases of OpenSPARC based on SPARC T1 (Leon, et al., 2007; Leon, Langley, & Shin, 2006) and SPARC T2 (Shah, et al., 2007; Nawathe, et al., 2007; Grohoski, et al., 2006) processors. SPARC T1 has 8 cores and can run up to 32 threads; 4 threads per core. SPARC T2 also has 8 cores but supports up to 64 threads; 8 threads per core. Most recent iterations of SPARC are SPARC T5 which has 16 cores and 8 threads per core for a total of 128 threads (Oracle, 2013), and SPARC M5 and M6 which have 6 cores and 12 cores respectively with 8 threads per core for a total of 48 and 96 threads respectively (Oracle, 2013). Both SPARC T5 and M5/M6 are based on the SPARC S3 core architecture. The main differences are the L3 cache size and the core count. M5 and M6 have fewer cores but much bigger L3 cache than T5. M5 and M6 have 48MB L3 cache while T5 has 8MB. This makes them best choice for big-memory applications.

There are three main methods used for pre-silicon validation of large and complex system-on-chip (SoC) designs including processors: simulation, hardware emulation, and FPGA prototyping (ARM, 2013; Synopsys, 2013). FPGA prototyping is much faster than simulation and hardware emulation. Simulation speed is about 100 Hz and hardware emulation speed is about 1 MHz only making FPGA prototyping a necessity nowadays for various reasons: Electronics market is becoming more aggressive than ever and time-to-market is thus very short. The economics of SoC product development including processors is changing. Figure 2 shows product engineering costs for different semiconductor process nodes (Synopsys, 2013). At the 22nm/20nm and 16nm/14nm nodes, the
A Collaborative Approach for Compressive Spectrum Sensing
www.igi-global.com/chapter/a-collaborative-approach-for-compressive-spectrum-sensing/123564?camid=4v1a

Generating Process Network Communication Infrastructure for Custom Multi-Core Platforms
www.igi-global.com/chapter/generating-process-network-communication-infrastructure/65607?camid=4v1a