Low-Power Warp Processor for Power Efficient High-Performance Embedded Systems

Roman Lysecky
Department of Electrical and Computer Engineering
University of Arizona
rlysecky@ece.arizona.edu

Abstract

Researchers previously proposed warp processors, a novel architecture capable of transparently optimizing an executing application by dynamically re-implementing critical kernels within the software as custom hardware circuits in an on-chip FPGA. However, the original warp processor design was primarily performance-driven and did not focus on power consumption, which is becoming an increasingly important design constraint. Focusing on power consumption, we present an alternative low-power warp processor design and methodology that can dynamically and transparently reduce power consumption of an executing application with no degradation in system performance, achieving an average reduction in power consumption of 74%. We further demonstrate the flexibility of this approach to provide dynamic control between high-performance and low-power consumption.

Keywords

Warp processing, low-power, hardware/software partitioning, dynamically adaptable systems, embedded systems.

1. Introduction

Field programmable gate arrays (FPGAs) are increasingly becoming more popular and mainstream. FPGAs have moved from primarily being used as a prototyping and debugging platform to being incorporated within many computing domains, from high-performance supercomputers to consumer electronics, even including electric guitars. FPGAs can implement any hardware circuit simply by downloading bits for the hardware circuit, much in the same way that microprocessors can execute any software program simply by downloading a new application binary. The programmability, flexibility, rapid prototyping, and debugging advantages that FPGAs offer over ASICs have enabled more designs to be implemented using FPGAs. This trend will likely continue as FPGAs become more advanced with larger capacities, higher performance, domain specific architectural features, and decreasing costs. For many low volume designs, the primary advantages of FPGAs are low NRE and unit costs.

With the continuing evolution of FPGAs, several single-chip microprocessor/FPGA devices have emerged, incorporating one or more microprocessors and an FPGA into a single device along with efficient mechanisms for communication between the microprocessor, FPGA, and shared memory. Such devices, available from Xilinx, Altera, and Atmel, are ideally suited to hardware/software partitioning.

Hardware/software partitioning is the task of partitioning an application between software executing on a microprocessor and hardware coprocessors, which can be efficiently implemented within an FPGA. Extensive research has demonstrated the benefits that can be obtained by re-implementing a software application's critical kernels as a custom hardware coprocessor on an FPGA. For many applications, researchers and commercial vendors have observed overall application speedups of 10X-100X [1][5][7][10][19][20], and approaching 1000X [2][22] for some highly parallelizable applications. Hardware/software partitioning can also reduce energy consumption by up to 99% [8][17][23]. While hardware/software partitioning offers the potential for tremendous speedups, automated partitioning compilers require a significant departure from mainstream software tools. Furthermore, such partitioning tools often require extensive design expertise and effort to maximize results.

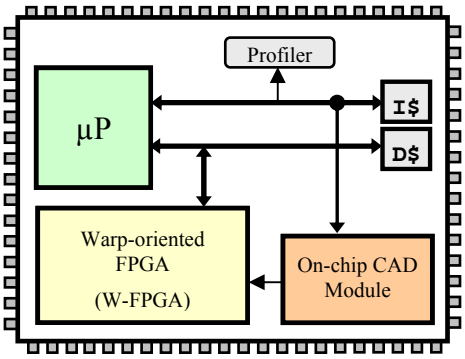
Recently, researchers demonstrated the feasibility of dynamic hardware/software partitioning that dynamically and transparently improve the speed and energy consumption of a software binary executing on a microprocessor, a process called warp processing [12]. A warp processor dynamically detects a software binary's critical kernels, re-implements those kernels as a custom hardware circuit in an on-chip FPGA, and replaces the software kernel by a call to the new hardware implementation of that kernel, all without any designer effort or knowledge thereof. Not all applications can be improved using warp processing, but many can, with application speedups of 2X-10X compared to software only execution on a low-power embedded processor. The increased performance of the warp processors also provides an average reduction in energy consumption of 66%, however, the impacts of warp processing on power consumption has not been considered.

While warp processing enables a new computing domain in which software kernels can be dynamically and transparently converted to hardware, previous warp processing work was performance-driven, focusing primarily on increasing application performance. Although increased performance is desired for almost all systems, the dynamic performance improvement of warp processors may be lost for certain application domains. For example, real-time systems must guarantee that task deadlines are met during execution. However, designers must typically be able to guarantee that those deadlines are met statically before executing on the final system. Dynamically improving the performance of such systems may provide some added insurance for meeting deadlines in unexpected situations, but such performance gains may not significantly benefit the system.

Alternatively, as power consumption is continuing to become a dominant design constraint, we propose to extend warp processing by leveraging voltage and frequency scaling to develop a low-power warp processor that can dynamically reduce power consumption with no performance degradation. In this paper, we will first present a high-performance XScale-based warp processor design and highlight the resulting speedup and energy benefits, in addition to analyzing the impacts on

978-3-9810801-2-4/DATE07 © 2007 EDAA

Figure 1: Warp processor architecture overview.



power consumption. We next present our low-power warp processing approach and detail our methodology for utilizing voltage and frequency scaling to reduce power consumption without degrading performance. We present our experimental results demonstrating the effectiveness of our low-power approach, and we conclude by highlighting several future research directions for low-power warp processing.

2. Warp Processing Overview

Figure 1 provides an overview of the previous warp processor architecture [12], consisting of a main processor with instruction and data caches, an efficient on-chip profiler, the warp-oriented FPGA (W-FPGA), and an on-chip computer-aided design module. Initially, a software application executing on a warp processor will execute only on the main processor. During execution of the application, the on-chip profiler monitors the application to determine the critical kernels within the application. After identifying the critical kernels, the on-chip CAD module re-implements the critical software kernels as a custom hardware component within the W-FPGA. Significant previous research has been conducted detailing the warp processor architecture and on-chip CAD tools [12][13][14][18] and are beyond the scope of this paper.

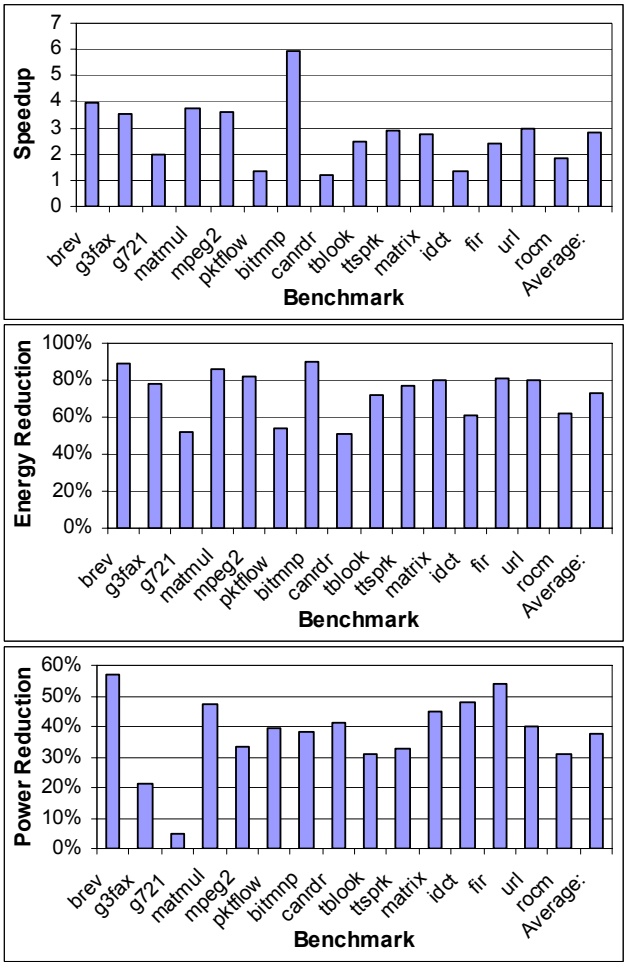
We initially consider a performance-driven warp processor design incorporating a high-performance XScale processor [9] executing at a maximum frequency of 624 MHz, a W-FPGA executing at 200 MHz, and a low-power ARM7 processor executing the on-chip CAD tools. Such a clock frequency ratio between microprocessor and FPGA is very reasonable and representative of a variety of commercially available devices. We simulated the warp processor execution for a number of standard embedded benchmark applications collected from several different benchmark suites, including Powerstone [15], EEMBC [4], MediaBench [11], and NetBench [16], in addition to a lean logic minimization algorithm [13]. Table 1 presents a summary of the embedded benchmark applications considered. We only considered those applications amenable to speedup using FPGA, namely applications whose critical kernels do not use floating point arithmetic, dynamic memory allocation, recursion, function pointers, and regular pointers (other than for array accesses). Simulation and estimation details for these results can be found in Section 4.

Figure 2 presents the speedup and energy reduction benefits of the performance-driven warp processor for several embedded benchmark applications compared to software only execution on a 624 MHz XScale processor. The performance-driven warp

Table 1: Embedded benchmark applications.

Benchmark	Benchmark Suite	Description
brev	Powerstone	Bit reversal.
g3fax	Powerstone	Group three fax decode.
g721	Powerstone	CCITT voice decoding.
matmul	Powerstone	Matrix multiplication.
mpeg2	MediaBench	MPEG-2 decoder.
pktflow	EEMBC	IP header validation.
bitmnp	EEMBC	Bit manipulation.
canrdr	EEMBC	Controller area network (CAN).
tblook	EEMBC	Table lookup and interpolation.
ttsprk	EEMBC	Engine spark controller.
matrix	EEMBC	Matrix operations.
idct	EEMBC	Inverse discrete cosine transform.
fir	EEMBC	Finite impulse response filter.
url	NetBench	URL-based switching.
rocm	Warp	Logic minimizer.

Figure 2: Speedup, energy consumption reduction, and power consumption reduction of performance-driven warp processor compared to a 624 MHz XScale processor alone.



processor achieves an average application speedup of 2.8X over the already high-performance XScale processor. Applications speedups range from 1.2X for *canrdr* to 6X for *bitmnp*, indicating that warp processing is quite effective in improving embedded system performance, even for high-performance systems. In addition, warp processing results in an average energy reduction of 73%, with a maximum reduction of 90% for *brev*.

The energy reduction of warp processing can be attributed to the significantly reduced application execution time. While reducing energy consumed to perform a given task is important for many applications, power consumption is often a more important design constraint and can be a limiting factor for many embedded systems. However, the significant energy reduction achieved by warp processing does not also imply a corresponding power reduction. We further analyzed the power consumption of the performance-driven warp processor compared to the XScale microprocessor alone. Figure 2 presents the power reduction benefits of performance-driven warp processing for several embedded benchmark applications compared to software only execution on a 624 MHz XScale processor. Although the on-chip FPGA increases both static and dynamic power consumption, the shift from active to idle power consumption of the XScale processor nets an overall power savings for all applications considered. On average, warp processing reduces power consumption by 38%. Many applications benefit more from warp processing than others, with power reductions ranging from as little as 4.8% for *g721* to as much as 57% for *brev*.

3. Low-Power Warp Processor

Performance-driven warp processing is ideally suited for systems in which performance is the dominant design constraint. For systems in which power consumption is the dominant design metric, we present an alternative low-power warp processing approach capable of dynamically partitioning an application to reduce power consumption without any performance degradation compared to the original software execution.

The low-power warp processor architecture is mostly identical to the performance-driven architecture, consisting of a voltage-scalable XScale processor, a frequency-scalable W-FPGA, and a low-power ARM7 processor for executing the on-chip CAD tools. The low-power architecture will take advantage of the voltage and frequency scaling capabilities of the XScale processor and incorporate dynamic clock management to support frequency scaling for the on-chip W-FPGA. While the XScale processor supports many voltage and frequency scaling options, we limit the possible selections for low-power warp processing to a predefined set of voltages and frequencies. Table 2 presents the operating voltage and frequency combinations considered for the XScale processor of our low-power warp processor design, taken from the XScale processor datasheet [9]. For FPGA frequency scaling, we assume we can dynamically adjust the operating frequencies in 5 MHz increments from 50MHz to the maximum frequency of 200 MHz.

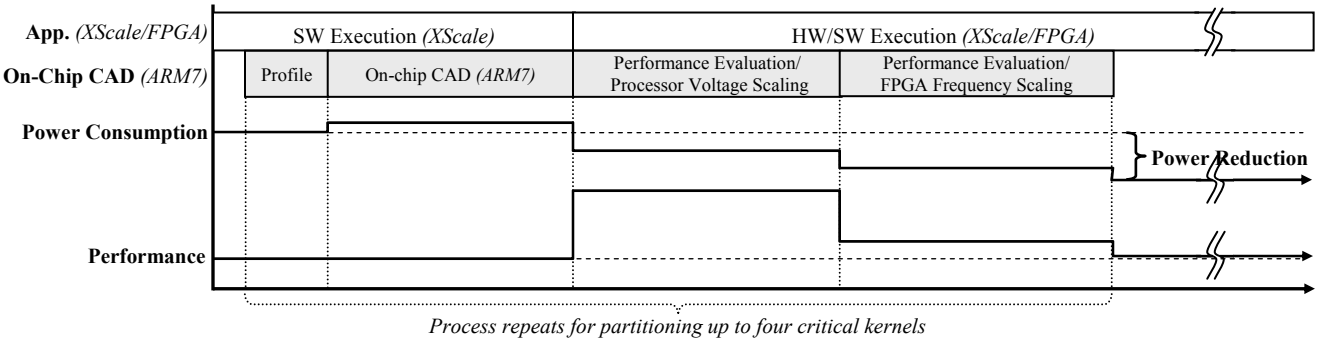
Table 2: XScale operating voltages and frequencies considered for low-power warp processor and the corresponding active and idle power consumption [9].

Voltage (V)	Frequency (MHz)	Active Power (mW)	Idle Power (mW)
0.85	13	44	8.5
0.90	104	116	64
1.05	208	279	129
1.10	312	390	154
1.35	416	570	186
1.45	520	747	222
1.55	624	925	260

The low-power warp processor initially operates much in the same way as the performance-driven warp processor. Figure 3 presents an overview of low-power warp processing, highlighting the overall low-power methodology and providing representative graphs of power consumption and performance during each step. The following details our low-power warp processor methodology:

- *Software Only Execution:* Initially the software application will execute on the XScale processor operating at the maximum frequency of 624 MHz.
- *Profiling:* During the software only execution, the on-chip profiler will continually profile the application to determine the critical kernels of the executing application. The current warp processor's non-intrusive profiler design [6] will determine the top 16 critical kernels and provide a relative ranking of the critical kernels with respect to overall execution time. During this period, the power consumption only consists of the power consumed by the XScale processor, as the FPGA and on-chip CAD module can be entirely shutdown, as they are not yet needed.
- *Dynamic Hardware/Software Partitioning:* After determining the critical kernels within the executing application, the on-chip CAD tools will partition up to four critical kernels to hardware implemented in the on-

Figure 3: Overview and timeline of low-power warp processing and the related dynamic effects on power consumption and performance. *(Not to Scale)*



chip FPGA. During this execution, power consumption will increase while the on-chip CAD tools execute on the low-power ARM7 processor. We note that the CAD tools only require a few seconds of execution time to partition the critical kernels to hardware and can be shutdown after partitioning the critical kernels. Furthermore, the power consumed by the ARM7 processor is less than 3% of the active power consumption of the XScale processor and only requires a few megabytes of memory during execution. As such, the overall power impact of executing the on-chip CAD tools is mostly negligible.

- *Hardware/Software Execution:* After partitioning the critical kernels to hardware, the application will execute as software executing on the processor and hardware executing on the FPGA. During this initial partitioned execution phase, the application performance will increase and the power consumption will decrease, as demonstrated by performance-driven warp processing.
- *Performance Evaluation and Processor Voltage Scaling:* The low-power warp processor now monitors and evaluates the performance of the partitioned application. By measuring the performance compared to the original software only performance, the low-power warp processor will continue to reduce the voltage and frequency of the XScale processor as long as the performance does not fall below the original software only performance. The voltage and frequency scaling performed during this stage significantly reduces the overall power consumption at the expense of reduced performance. However, due to the coarse granularity of the predefined voltage and frequency options available for the processor, the resulting application performance can still be as much 2X faster than the original software only execution.
- *Performance Evaluation and FPGA Frequency Scaling:* Finally, the warp processor will again monitor and evaluate the performance and continually reduce the frequency of the FPGA as long as the performance does not fall below the original software only performance. The frequency scaling of the FPGA will further help to reduce the overall power consumption, and the resulting performance will closely match the original software only execution.

By re-implementing critical software kernels more efficiently as hardware within the on-chip FPGA, the low-power warp processor can significantly reduce power consumption compared to software only execution of the XScale processor. The low-power warp processor currently supports partitioning up to four critical kernels within the W-FPGA, after which the profiler and on-chip CAD module can be shutdown, thereby amortizing the power consumption of profiling and partitioning over the entire execution of the application. While the current approach does not support continuous profiling and re-partitioning of critical kernels, future work includes investigating this direction.

4. Results

We evaluate the performance and power consumption of our low-power warp processor for the embedded benchmark applications listed in Table 1. For all applications, we simulated the performance-driven and low-power warp processors utilizing an XScale port of the SimpleScalar simulator [3] to determine software execution cycles and gate-level simulations of the W-FPGA to determine hardware cycles for the partitioned critical kernels. All software and hardware execution times include the communication cycles required to communicate and synchronize between the processor and FPGA.

We calculated the power consumed before and after dynamically partitioning the critical kernels to hardware using the following equations:

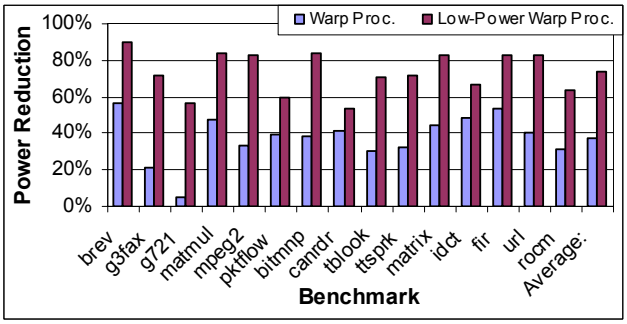
$$P_{Total} = P_{Proc} + P_{FPGA}$$

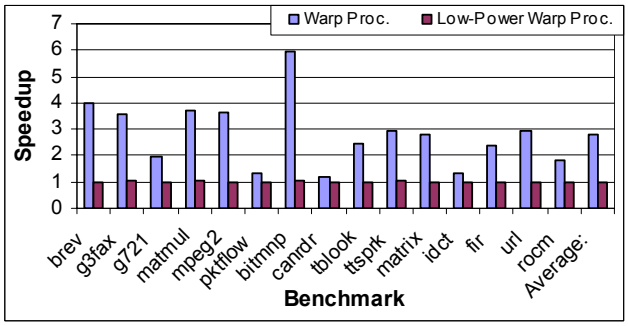
$$P_{Proc} = P_{Proc\ (active)} \frac{t_{Proc\ (active)}}{t_{Total}} + P_{Proc\ (idle)} \frac{t_{Proc\ (idle)}}{t_{Total}}$$

$$P_{FPGA} = P_{FPGA\ (active)} \frac{t_{FPGA\ (active)}}{t_{Total}} + P_{static}$$

The total power consumption (P_{Total}) is the sum of the power consumed by the processor (P_{Proc}) and the power consumed by the FPGA (P_{FPGA}). The power consumed by the processor consists of the active power consumed executing the software applications and the idle power consumed by the processor when the FPGA is active. The active power consumption of the processor is computed as the processor's active power consumption ($P_{Proc(active)}$) multiplied by the ratio of processor's active time ($t_{Proc(active)}$) to the total execution time (t_{Total}). The idle power consumption of the processor is computed as the processor's idle power consumption ($P_{Proc(idle)}$) multiplied by the

Figure 4: Speedup and percentage power consumption reduction for low-power warp processor and performance-driven warp processor compared to a 624 MHz XScale processor alone.





ratio of processor's idle time ($t_{Proc(idle)}$) to the total execution time (t_{Total}). The active and idle power consumption of the XScale processor for the predefined voltage and frequency settings is provided in Table 2. The power consumed by the FPGA consists of the active power consumed while the FPGA is executing the partitioned hardware kernels and the FPGA's static power consumption. The active power consumption of the FPGA is computed as the FPGA's active power consumption ($P_{FPGA(active)}$) multiplied by the ratio of FPGA's active time ($t_{FPGA(active)}$) to the total execution time (t_{Total}). While the static power consumption of the FPGA is consistent across all benchmarks and frequencies, the active power consumption of the FPGA is calculated for each partitioned critical kernels and operating frequency using gate-level simulations of the W-FPGA design synthesized using Synopsys Design Compiler targeting a 0.18 µm technology. We note that the XScale processor is also implemented using a 0.18 µm technology. We further note that the functionality of the W-FPGA design has also been verified through post-layout simulation targeting a 0.13 µm technology as part of the Intel Research Shuttle.

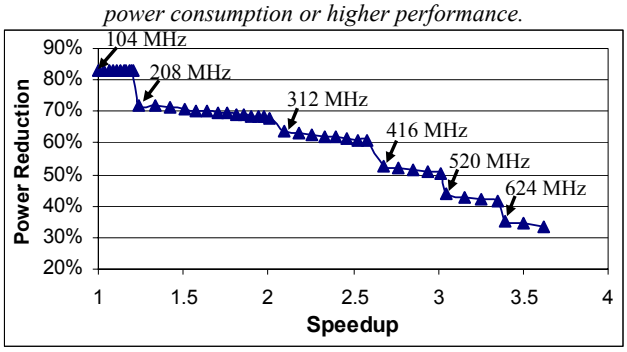
Figure 4 presents the power reduction of our proposed low-power warp processor and the performance-driven warp processor for several embedded benchmark applications compared to software only execution on a 624 MHz XScale processor. The low-power warp processor provides greatly improved power reduction compared to the original performance-driven warp processor. The low-power warp processor increases power savings from an average of only 38% to an average of 74%, almost a 2X improvement in power reduction over the original design. The lowest power reduction achieved by our low-power warp processor is 54% for the application *canrdr*, which is greater than power reduction of the performance-driven warp processor for all but one application. On the other extreme, the low-power warp processor achieves it highest power reduction for the application *brev*, reducing power consumption by an outstanding 90%.

The power consumption of the low-power warp processor is primarily dominated by the XScale processor. On average, the XScale processor consumes 64% of the total low-power warp processor's power consumption after partitioning and voltage and frequency scaling. However, the breakdown of power consumption between the XScale processor and FPGA varies across applications, where the processor accounts for as little as 18% to as much as 80% of the total power.

The significant power savings of low-power warp processing is attributable to the ability to scale back the voltage and frequency of the XScale processor. The level at which the low-power warp processor can scale back the voltage and frequency is directly dependent on the specific application and how effectively the dynamic partitioning can re-implement the application's critical kernels within the on-chip FPGA. For the majority of the application considered, the voltage and frequency of the XScale processor could be scaled back to either 0.9V/104MHz or 1.05V/208MHz. For one application, *brev*, the voltage and frequency was scaled back to lowest possible option of 0.85V/13MHz.

Figure 4 also highlights the speedup of our proposed low-power warp processor compared to software only execution on a 624 MHz XScale processor. As the low-power warp processor scales back the performance benefits after partitioning the critical kernels to the FPGA, we do not expect to see any

Figure 5: Low-power warp processing speedup versus percentage power reduction for *mpeg2* compared to a 624 MHz XScale processor alone. *Annotation indicates points at which changing operating voltage/frequency results in either lower power consumption or higher performance.*



significant speedups. However, we also must ensure that the low-power warp processor approach does not degrade system performance. As expected, our proposed low-power warp processor achieves an average speedup of only 1.01X, ranging from 1.00X to 1.04X. These results also confirm that overall system performance is not adversely affected for any application.

Therefore, by reducing power consumption and leaving performance almost unchanged, the significant power savings of low-power warp processing directly results in a similar reduction in energy consumption of 74%, on average.

Finally, we analyzed the performance and power tradeoffs that are possible with the low-power warp processor design. By enabling low-power operation, a warp processor provides many possible performance and power consumption configurations. Figure 5 presents the tradeoff between application speedup and power reduction for the *mpeg2* application, showing only those points that result in either improved performance or reduced power consumption. Performance and power reduction range from a speedup of 1X and an 83% reduction in power consumption to a speedup of 3.6X and power reduction of only 33%. With this flexibility, the low-power warp processor can be utilized in high-performance embedded systems, low-power embedded systems, and even systems in the middle that may sporadically require high-performance but where overall low-power consumption is essential. While warp processing is currently targeted at embedded applications, the flexibility of low-power warp processing is perfectly suited to battery-powered computing devices, such as PDAs and laptop computers. For example, when such portable devices are plugged into an electrical outlet, the device can utilize performance-driven warp processing to maximize performance. Alternatively, when running on battery power, low-power warp processing can be utilized to reduce power consumption to extend battery.

5. Conclusions

Low-power warp processing leverages the dynamic partitioning benefits of warp processors and the power saving benefits of voltage and frequency scaling to create a high-performance embedded architecture capable of dynamically reducing power consumption without degrading performance. By focusing on reducing power consumption, our proposed low-power design can increase the power reduction by almost 2X compared to the

original performance-driven warp processor, achieving an average power reduction of 74%. Furthermore, the low-power warp processor design provides great system flexibility, allowing system performance and power consumption to be dynamically modified during execution to meet changing system needs. Such flexibility can provide the equivalence of a 2.5 GHz operating frequency or an 83% reduction in power consumption. The tremendous performance and/or power saving benefits of warp processing open a door to a new computing domain of transparent, dynamically adaptable systems capable of meeting ever increasing performance and power design metric without design expertise or knowledge.

6. Future Work

While both performance-driven and low-power warp processors offer many advantages, warp processing is still in its infancy. Much research is needed to extend and improve warp processing, potentially to boarder application domains including desktops and servers. We are currently pursing further research in area of warp processors, specifically focusing on low-power and real-time systems. While the current low-power warp processor provides substantial power reductions, the partitioning algorithms employed during warp processing are still performance-oriented. We plan to investigate the potential for further improving power reduction by developing partitioning algorithms specifically focused on reducing power consumption. We also plan to investigate the potential of utilizing low-power modes supported by voltage scalable processors to further reduce power consumption as well. Such low-power modes offer tremendous power saving, even over the idle power consumption, but often incur a significant performance penalty entering and exiting such power modes. We also plan to evaluate utilizing a low-power FPGA, which provides significantly reduced active and static power consumption at the cost of reduced performance [21], within the low-power warp processor.

7. References

[1] Berkeley Design Technology, Inc. http://www.bdti.com/articles/info_eet0207fpga.htm#DSPEnhanced%20FPGAs. 2004.

[2] Böhm, W., Hammes, J., Draper, B., Chawathe, M., Ross, C., Rinker, R., Najjar, W. Mapping a Single Assignment Programming Language to Reconfigurable Systems. Journal of Supercomputing, Vol. 21, pp. 117-130, 2002.

[3] Burger, D., T. Austin. The SimpleScalar Tool Set, version 2.0. SIGARCH Computer Architecture News, 1997.

[4] EEMBC. Embedded Microprocessor Benchmark Consortium. http://www.eembc.org, 2005.

[5] Ernst, R., J. Henkel, T. Benner. Hardware-Software Cosynthesis for Microcontrollers. IEEE Design & Test of Computers, Oct/Dec, pp. 64-75, 1993.

[6] Gordon-Ross, A., F. Vahid. Frequent Loop Detection Using Efficient Non-Intrusive On-Chip Hardware. Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES), pp. 117-124, 2003.

[7] Guo, Z., Buyukkurt, B., Najjar, W., Vissers, K. Optimized Generation of Data-Path from C Codes. Design Automation and Test in Europe Conference (DATE), pp. 112-117, 2005.

[8] Henkel, J. A Low Power Hardware/Software Partitioning Approach for Core-based Embedded Systems. Design Automation Conference, pp. 122 - 127, 1999.

[9] Intel Corp. XScale PXA27x Processor Family. http://www.intel.com/design/pca/prodbref/253820.htm, 2006.

[10] Keane, J., C. Bradley, C. Ebeling. A Compiled Accelerator for Biological Cell Signaling Simulations. International Symposium on Field-Programmable Gate Arrays (FPGA), pp. 233-241, 2004.

[11] Lee, C., M. Potkonjak, W. Mangione-Smith. MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems. International Symposium on Microarchitecture (MICRO), pp. 330-335, 1997.

[12] Lysecky, R., G. Stitt, F. Vahid. Warp Processors. ACM Transactions on Design Automation of Electronic Systems (TODAES), Vol. 11, No. 3, pp. 659 - 681, 2006.

[13] Lysecky, R., F. Vahid. On-chip Logic Minimization. Design Automation Conference (DAC), pp. 334-337, 2003.

[14] Lysecky, R., F. Vahid, S. Tan. Dynamic FPGA Routing for Just-in-Time FPGA Compilation. Design Automation Conference (DAC), pp. 954-959, 2004.

[15] Malik, A., B. Moyer, D. Cermak. A Low Power Unified Cache Architecture Providing Power and Performance Flexibility. International Symposium on Low Power Electronics and Design (ISLPED), pp. 241-243, 2000.

[16] Memik, G., W. Mangione-Smith, W. Hu. NetBench: A Benchmarking Suite for Network Processors. International Conference on Computer-Aided Design (ICCAD), 2001.

[17] Stitt, G., F. Vahid. The Energy Advantages of Microprocessor Platforms with On-Chip Configurable Logic. IEEE Design and Test of Computers, Nov/Dec 2002.

[18] Stitt, G., F. Vahid. New Decompilation Techniques for Binary-level Co-processor Generation. International Conference on Computer Aided Design (ICCAD), 2005.

[19] Stitt, G., F. Vahid, G. McGregor, B. Einloth. Hardware/Software Partitioning of Software Binaries: A Case Study of H.264 Decode. International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pp. 285-290, 2005.

[20] Stitt, G., F. Vahid, S. Nematbakhsh. Power Savings and Speedups from Partitioning Critical Loops to Hardware in Embedded Systems. ACM Transactions on Embedded Computing Systems (TECS), Vol. 3, No. 1, pp. 218-232, 2004.

[21] Tuan, T., S. Kao, A. Rahman, S. Das, S. Trimberger. A 90nm Low-Power FPGA for Battery-Powered Applications. International Symposium of Field Programmable Gate Arrays (FPGA), 2006.

[22] Venkataramani, G., W. Najjar, F. Kurdahi, N. Bagherzadeh, W. Bohm. A Compiler Framework for Mapping Applications to a Coarse-grained Reconfigurable Computer Architecture. Conf. on Compiler, Architecture and Synthesis for Embedded Systems (CASES), 2001.

[23] Wan, M., Y. Ichikawa, D. Lidsky, L. Rabaey. An Power Conscious Methodology for Early Design Space Exploration of Heterogeneous DSPs. ISSS Custom Integrated Circuits Conference (CICC), 1998.