Chip Multi-Processor Generator

Alex Solomatnikov, Amin Firoozshahian, Wajahat Qadeer, Ofer Shacham, Kyle Kelley, Zain Asgar, Megan Wachs, Rehan Hameed, Mark Horowitz
Stanford University
{sols, aminf13, wqadeer, shacham, kkelley, zasgar, wachs, rhameed, horowitz}@stanford.edu

ABSTRACT
The drive for low-power, high performance computation coupled with the extremely high design costs for ASIC designs, has driven a number of designers to try to create a flexible, universal computing platform that will supersede the microprocessor. We argue that these flexible, general computing chips are trying to accomplish more than is commercially needed. Since design NRE costs are an order of magnitude larger than fabrication NRE costs, a two-step design system seems attractive. First, the users configure/program a flexible computing framework to run their application with the desired performance. Then, the system "compiles" the program and configuration, tailoring the original framework to create a chip that is optimized toward the desired set of applications. Thus the user gets the reduced development costs of using a flexible solution with the efficiency of a custom chip.

Categories and Subject Descriptors
B.7.2 [**Hardware**]: Integrated Circuits – *Design Aids*

General Terms
Algorithms, Performance, Design

Keywords
Chip Multi-Processor, High-Level Design

1. INTRODUCTION
The result of CMOS technology scaling is growing design complexity and thus growing *non-recurring engineering* (NRE) costs associated with the design and verification of the chip. Complexity also makes it hard to predict the time-to-market and increases the risks of application specific integrated circuit (ASIC) design [1]. As a consequence, only IC market segments with large volumes can justify the NRE costs of ASIC design.

One possible approach to improve design productivity, advocated by reconfigurable processor IP providers such as Tensilica, is to use customized embedded processors as a new fundamental building block for complex systems-on-chip (SoC) [2]. Many recent ASIC designs are actually chip multi-processors (CMP) such as the Silicon Packet Processor of Cisco's CSR-1 router [3], TI's OMAP platform [4], and Philips' MSVD [5]. However, the design of such SoCs is far more complex than just customization of reconfigurable processors and optimization of software. Putting together multiple heterogeneous processors and task-specific hardware accelerators requires a lot of hardware and software to interface the units to each other, making it a challenging design and verification problem.

As a result of the difficulty in designing these SoC chips, numerous startups and established companies are trying to create the next generation computing platform, by extending FPGAs with high-level primitives [6], creating more efficient computing for an application domain [7], etc. We argue that there is a middle ground in this conflict. To reduce the design NRE costs, we should create a flexible/configurable abstract system that has been optimized for a broad application domain. The application developer will only need to configure this system, and port his application code to run on this machine. Since this abstract system would have well defined interfaces and high-level primitives, like the next generation computing platforms, the complexity of the design process should be similar to using a programmable engine. The advantage of this approach is two fold. First, and most important, it allows a more powerful set of configuration operations, like creating a tailored functional unit, than would be possible for a reconfigurable hardware solution. Second, it allows tools to identify those portions of the architecture that are unused or underutilized by the target applications and remove them from the implementation.

Clearly there are many issues that need to be addressed before this type of system can be a reality. The most obvious is the requirement for an automated backend toolset that could take the finished logic (with floor-plan information) and create a chip with little manual intervention. More interesting are the tools that perform the optimization of the design. One can imagine some passes that are generic, and that would probably be used in all such systems, while others would be domain specific and tied to the input abstraction used.

In this paper we propose a chip multiprocessor generator based on a reconfigurable CMP architecture. The basic reconfigurable chip architecture is described in the next section. How this system can improve chip efficiency is described in Section 3.

2. RECONFIGURABLE CMP
Our reconfigurable CMP design framework grew out of an earlier effort to build a flexible chip multiprocessor [8]. It is a modular, scalable hierarchical architecture, which integrates a large number of processors and memory blocks on a single die. The system consists of Tiles, each with two Tensilica cores, several reconfigurable memory blocks, and a crossbar connecting them. Tiles are placed in groups of four, forming Quads. Tiles in the Quad are connected to a shared protocol controller. Quads are then connected to each other and to the off-chip interfaces using an on-chip network.

DAC 2007, June 4–8, 2007, San Diego, California, USA.

The memory blocks inside the Tile contain dual-ported meta-data storage, and logic that can perform atomic read-modify-write operations on meta-data bits, FIFO pointers and logic, and a comparator that is used for a single cycle read-compare operation [9]. This allows the memory blocks to be configured to work as caches, local scratchpad memories, or FIFOs. The Tile uses a general crossbar to connect the memory blocks to the access ports of the processors, allowing different cache parameters (including associativity, block size, and coherence) to be set by the user.

The quad protocol controller is a programmable message engine that can perform all the required communication between the Tiles, from cache refills and cache coherence protocol to complex DMA requests. We have already mapped conventional multi-threaded models with shared memory and cache coherence, streaming [10] and transactional memory [11] on this platform.

3. CMP GENERATOR

While CMP architecture described in the previous section was designed to be directly implemented as a reconfigurable chip this approach had two significant shortcomings. The first was that the resulting system, while flexible, would not provide the end user the efficiency that was possible had the system been hard-wired. The second was that we could not take full advantage of the customization potential of the Tensilica processor to really match the memory system and the processor to the application.

In thinking more about the efficiency problem, it became clear that we should consider the system we constructed as being used for design input, and not as the RTL for the final design. Between the design input and output we could perform a number of optimization steps to improve the overall efficiency of the design. In addition to the obvious advantage of customizing the size and type of memories (e.g. using a CAM if needed) and using the Tensilica system to optimize the functional units of the processors to the applications, this two step process would also allow the user to compile out much of the flexibility that was in the original design framework. The crossbar between the processor and all the memory mats becomes a set of wires; extra meta-data bits that are not used are removed, buffers can be sized to meet the actual traffic/number of outstanding requests, etc.

We have used the current set of design synthesis tools to remove "dead code" when constants are used as configuration inputs to a flexible design. This approach was needed to map a configuration of our flexible architecture to an FPGA. Since FPGA gates were limited, we mapped the configured logic, and not the full design. The initial results look promising.

Finally, for applications that have exceptionally high performance demands and require custom hardware accelerators that cannot be substituted by configurable processors, one could replace a processor with a specialized functional unit. This functional unit would be connected to the standardized memory interfaces developed for the processors to allow the hardware accelerator to access Tile memory blocks directly.

The main benefit of the proposed CMP generator is reduced design and validation time. Validation should be simplified since all interactions between the interfaces have been already validated. Design time should be reduced because additional functionality is specified in a higher level language like TIE [2], and complex interface/communication issues have already been resolved. This allows the ASIC designer to focus on higher level design issues such as algorithms, and partitioning of data and computation.

The starting point for many of the tools needed in the CMP generator already exists. Design tools that incorporate the verification environment within the code (such as System Verilog) can allow constant propagation into not only the RTL but the environment as well. Current synthesis tools can eliminate much of the redundant logic after customization, but we probably would like a higher-level optimization pass in these types of systems. The generated RTL would still need to be fed into the ASIC place-and-route flow. These systems are getting more automatic, and we expect that for highly structured designs like ours, sometime in the future, generating the layout from highly structured code will be fast and efficient.

4. CONCLUSIONS

We present a novel approach to reducing the very high NRE cost of chip design, by making chip design essentially the configuration/programming of an abstract, flexible machine. Our key insight is that much of the unneeded flexibility in the architecture could be identified and compiled out. This gives the potential of rapid design time, with a high-performance and power efficient silicon implementation. While the construction of the original generator would still be expensive, it could be amortized over many different designs. Experiments with our own flexible CMP system, built using Tensilica processors, look promising.

5. REFERENCES

[1] Bryant, R., et al. Limitations and Challenges of Computer-Aided Design Technology for CMOS VLSI. *Proceedings of IEEE, vol. 89,* 3, March 2001, 341-365.

[2] Rowen, C. Reducing SoC Simulation and Development Time. *IEEE Computer,* 11, December 2002, 29-34.

[3] Dixit, A. Networking Applications for Xtensa Configurable Processors. *Linley Tech 2006,* January 25th 2006.

[4] Wireless Terminals Solutions Guide. Texas Instruments.

[5] Mutz, S., et al. Heterogeneous Multiprocessing for Efficient Multi-Standard High Definition Video Decoding. *Hotchips 18,* August 2006.

[6] http://www.cswitch.com

[7] http://www.streamprocessors.com

[8] Mai, K., et al. Smart Memories: A Modular Reconfigurable Architecture. *International Symposium on Computer Architecture,* June 2000.

[9] Mai, K., et al. Architecture and Circuit Techniques for a 1.1GHz 16-kb Reconfigurable Memory in 0.18um-CMOS. *IEEE Journal of Solid- State Circuits*, January 2005.

[10] Kapasi, U., et al. Programmable Stream Processors. *IEEE Computer, vol. 36,* 8, August 2003, 54-62.

[11] Hammond, L., et al. Transactional Memory Coherence and Consistency. *International Symposium on Computer Architecture,* June 2004.