Using an FPGA as a Prototyping Platform for Multi-core Processor Applications

Christopher R. Clark, Ripal Nathuji, Hsien-Hsin S. Lee
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA
{cclark, rnathuji, leehs}@ece.gatech.edu

In this work, we investigate the use of an FPGA to assist in the application development process for multi-core processors. We implement a basic multi-core architecture in an FPGA and develop a software application that takes advantage of multiple processors. We also investigate the use of processor extensions by attaching application-specific custom logic to each core.

A. FPGA Multi-core Architecture

Our architecture consists of multiple processor cores sharing a common data bus and a 32-bit memory address space. We use both hard-IP cores and soft-IP cores. A hard-IP embedded PowerPC (PPC) core is attached to a 64-bit Processor Local Bus (PLB) and accesses instructions and data from a 16 KB memory block connected to the PLB. A bridge connects the PLB to the 32-bit On-chip Peripheral Bus (OPB). The OPB bus hosts one or more MicroBlaze [1] soft-IP processor cores. Each MicroBlaze (MB) processor has a 16 KB dual-ported local memory for storing its program and its static and dynamic data. The PPC and all of the MB processors can access a shared 8 KB memory via the OPB bus. We have implemented this architecture on an Avnet development board containing a Xilinx Virtex-II Pro 2VP-20 FPGA. All logic, including processors and buses, are connected to a 100 MHz clock. The various on-chip memory regions are built using one or more hard-IP BlockRAMs.

B. Sample Application: DNA Sequencing

To study the feasibility of using an FPGA for the development and testing of software for multi-core processors, we implement a DNA sequencing algorithm because this task is well-suited to parallelization. Since our goal was to study functionality rather than achieve high performance, our sequencing algorithm uses a simple dynamic programming approach. The scenario we consider is where a single query sequence is compared against numerous sequences from a database. Due to the data-parallel nature of the operations, we utilize a single program multiple data (SPMD) programming model.

In our system, the PPC core acts as the central coordinator by assigning database sequences to the individual MB cores and reading back results. Inter-processor communication is conducted using the shared memory block, which contains synchronization data structures. Each MB copies a database sequence from the work queue in shared memory to local memory. It then executes the sequencing algorithm and writes the results to the shared memory. With multiple processors performing comparisons in parallel, the time to search an entire database is reduced. We have measured the time required to process a database with 1000 entries, where each entry is 256 characters in length. Table I presents the execution time and speedup for designs with one, two, and four MicroBlaze cores.

The small amount of local memory available to store the state table places a practical limit on the length of compared sequences. It would be possible to compare longer sequences by iteratively running the algorithm on overlapping sections of the sequence, but this would make the number of operations $O(n^2)$ rather than $O(n)$.

C. Processor Extensions Using Custom Logic

An interesting feature of an FPGA multi-core testbed is the ability to quickly implement and evaluate processor extensions using the available programmable logic. We have developed a pattern-matching co-processor that connects to the MicroBlaze through two Fast Simplex Link (FSL) connections. The FSL channels allow data to be transferred between the processor's register file and external circuitry. The pattern-matcher is based on a circuit presented in [2] that is capable of detecting approximate matches between two sequences. This is ideal for DNA sequencing, where we are looking for closely-related sequences with a small number of character substitutions (nucleotide mismatches) and character insertions (gaps).

We modified the MB software to use the co-processor to enable the comparison of longer database sequences. The idea is to use the custom logic to quickly scan the sequence and determine interesting sub-sequences for further processing by the software. This prevents unnecessary computation by allowing the software to ignore sections of the database sequence that would not lead to a close match. Table II shows the execution time of the algorithm using the pattern-matching extension for 1000 database entries of various lengths. The results show that longer sequences can be compared with about a factor of two increase in execution time, significantly better than the factor of n increase that would be incurred by a software-only approach.

TABLE I
PERFORMANCE SCALABILITY WITH MULTIPLE CORES

Number of Cores	Execution Time (sec)	Speedup
1	23.89	1.0
2	11.94	2.0
4	5.96	4.0

TABLE II
PERFORMANCE WITH PATTERN-MATCHING EXTENSION

Length of Database Sequences	Execution Time (sec)		
	1 Core	2 Cores	4 Cores
128	47.93	23.99	12.00
256	57.78	28.92	14.40
512	115.6	57.73	28.92

D. Conclusion

We have found that FPGAs are a useful platform for studying issues related to multi-core processors. Their flexibility allows different designs to be evaluated (e.g. MPSD or streaming architectures), and their ability to run full-length programs provides an advantage over software simulators. This type of configurable platform enables research on multiple issues related to multi-core processors, such as helper threads or the allocation of different functions to different cores (e.g. security, networking).

REFERENCES

[1] "MicroBlaze Soft Processor Core," *http://http://www.xilinx.com/xlnx/xebiz /designResources/ip_product/details.jsp?key=microblaze*.
[2] C.R. Clark and D.E. Schimmel, "A Pattern-Matching Co-Processor for Network Intrusion Detection Systems," *In Proceedings of IEEE International Conference on Field-Programmable Technology*, 2003.