Performance and Power Evaluation
of an In-line Accelerator

Alejandro Rico
Barcelona Supercomputing Ctr
Barcelona, Spain
alejandro.rico@bsc.es

Jeff H. Derby
IBM Corporation, STG
Research Triangle Park, NC
jhderry@us.ibm.com

Robert K. Montoye
IBM TJ Watson Research Ctr
Yorktown Heights, NY
montoye@us.ibm.com

Chenyong Cher
IBM TJ Watson Research Ctr
Yorktown Heights, NY
chenyong@us.ibm.com

Pradip Bose
IBM TJ Watson Research Ctr
Yorktown Heights, NY
pbose@us.ibm.com

ABSTRACT
In this paper we evaluate the performance and power of a processor-attached in-line accelerator. The accelerator provides high-performance SIMD computing and power efficiency by means of a very large register file and a set of vector multimedia extensions based on IBM's PowerPC VMX. Our experiments show significant performance improvements and power reduction, mainly due to the drastic decrease of memory accesses caused by the software-managed locality of the very large register file. Total execution time is, on average, reduced by 61%, while consuming 55% less energy.

Categories and Subject Descriptors
C.1.2 [Multiple Data Stream Architectures (Multiprocessors)]: Single-instruction-stream, multiple-data-stream processors (SIMD)

General Terms
Experimentation, Performance

Keywords
PowerPC, SIMD, VMX, accelerator

1. INTRODUCTION
iVMX (indirect VMX) [1] is an extension to the VMX (also known as Altivec) architecture. iVMX provides single-instruction multiple-data (SIMD) execution pipelines operating on 128-bit registers that can be used as four 32-bit, eight 16-bit or sixteen 8-bit elements, providing the same capabilities as previous VMX implementations. On top of that, iVMX features a high-bandwidth very large vector-scalar register file (VSRF) with up to 4096 128-bit registers. To reference such a large amount of registers, iVMX includes a novel indirection mechanism so that it can dynamically address up to 4096 registers. To feed the VSRF, iVMX includes a cache-line-wide path connected directly to the second level (L2) cache. Load/store block instructions are added to the regular load/store quadword operations to move one cache line (128B) or two consecutive cache lines (256B) between the L2 and the VSRF.

In [1], iVMX is embedded in a wide issue machine, where each iVMX unit issues up to 3 instructions per cycle and includes a VSRF with 1024 registers. In this work, we use iVMX as a flexible accelerator that can work as an auxiliary processor unit (APU) which can be connected to the APU interface of an in-order multi-threaded processing core, similarly to the SIMD floating-point unit for BlueGene/L presented in [2]. We assume that the APU interface performs single in-order issue per iVMX unit. Since the number of issue instructions per cycle is decreased from 3 to 1, the number of ports in the VSRF can be reduced and, therefore, capacity can be increased. This way, we use a VSRF with 4096 VMX registers, which is the maximum allowed by the indirection mechanism specifications.

2. iVMX ARCHITECTURE MODELING
For the performance and power evaluation, a cycle-accurate trace-based processor simulator has been extended to incorporate a model for iVMX. The base simulator models a multi-threaded PowerPC core and its cache hierarchy. Processor pipelines are modeled in detail with built-in simultaneous multi-threaded (SMT) capabilities. First level instruction and data caches, as well as the second level cache, are also implemented in a cycle-accurate way. The simulator generates an extensive report of performance and utilization statistics of the application execution on the multiple threads in the core.

The performance and utilization data resulting from simulation feed a pre-VHDL model for active and leakage power to provide a power estimation of the core running a particular workload. The unconstrained active power of each logic macro is estimated based on latch counts and area. The switching power associated with latches is the most significant part of each macro's active power. This is obviously proportional to the latch count. The logic data switching power has been found to correlate strongly with the macro area; hence there is a component of active power that is dependent on area as well. The actual active power (which includes array power) is modulated by the clock-gating factors computed from simulation. This allows us to generate workload-dependent power results, under the assumption that fine-grain, pipeline stage-level clock-gating is im-
implemented in the design. Clock-gating factors for the logic in a particular macro is computed based on the utilization of the pipeline stages contained by that macro. Finally, the total resulting active power is scaled for the target implementation process, which, in this work, is assumed to be 45nm CMOS SOI technology.

The leakage power model takes into account two main components of static standby current: subthreshold channel leakage and gate tunneling leakage. Channel leakage is estimated using state-of-the-art equations based on device gate width, threshold voltage ($V_{th}$) and the device efficiency in turning on or off. The oxide layer thickness is also used to estimate gate tunneling leakage. The total leakage current is the sum of both channel leakage and gate tunneling leakage for each device type (i.e., different gate width and $V_{th}$) and factoring the effect of drain-induced barrier lowering in channel leakage and the effects of supply voltage and temperature in both channel and gate leakage.

3. IVMX EVALUATION

In order to measure the actual performance improvement provided by the addition of the large VSRF and the additional VMX extensions, iVMX is compared with a standard VMX implementation. The VMX baseline includes a vector register file with 32 128-bit VMX registers. To perform a fair comparison, we assume the iVMX and baseline VMX units to have the same regular VMX capabilities, meaning same pipeline depths and instruction latencies.

The iVMX configuration is composed of one in-order multi-threaded PowerPC host core with an attached APU containing two iVMX units and performing single in-order issue per iVMX unit. The VMX configuration just replaces the two iVMX units with two standard VMX units.

Thanks to the very large register file in iVMX, applications are able to load large amounts of data at once, operate locally for a long period and store the results at the end. Baseline VMX codes, require the use of many more loads and stores that result from spill code, because of the small size of the register file. iVMX codes therefore require far fewer vector load/store instructions and fewer scalar arithmetic instructions for address computation.

For this evaluation, we use three different application kernels: a 1D FFT, a Viterbi decoder and a pyramid filter. Comparing the iVMX and VMX versions of these kernels, we appreciate a significant reduction in the number of vector loads (up to 26%), vector stores (up to 13%), and scalar arithmetic (up to 33%) instructions, which results in a total code path length reduction of 38% for FFT, 50% for Viterbi and 62% for pyramid filters.

The large reduction of memory instructions in iVMX codes significantly reduces the number of stall cycles due to cache misses. Also, the fact that the distance between memory instructions is much larger in iVMX codes than in VMX-targeted codes, along with the large amount of available registers, gives much more flexibility to the compiler register allocation algorithm, which is able to schedule dependent instructions appropriately, in order to avoid stall cycles due to register dependencies. As a result, iVMX gets CPI reductions between 8% (Viterbi) and 50% (Pyramid), which is consistent across single- and dual-threaded simulations.

The reduced instruction counts in iVMX along with its CPI improvements over VMX result in the total execution time benefits shown in Figure 1. Both FFT and Viterbi kernels for iVMX complete execution in about half the time required by the baseline VMX unit. The maximum boost takes place in Pyramid, which reduces execution time by 80%. These significant improvements result in a practical speedup of up to 5x, and supports the expected benefits of having such a dynamically addressed very large register file.

Figure 2 shows the normalized total energy required to execute both VMX and iVMX experiments across all kernels. Although power consumption for iVMX is higher due to the larger area and additional logic for indirection and extended functionalities, the lower CPI and a reduced utilization of the cache hierarchy leads in most of the cases to a better energy per instruction consumption compared to the VMX configuration. This fact, along with the significant code reduction, results in very large energy savings ranging from 42% (FFT) to 76% (Pyramid).

4. CONCLUSIONS

In this paper, we build a fundamental understanding of the benefits of a very large register file in the context of an in-line vector/SIMD accelerator. The dynamic indirection techniques employed to access such a large register file required a significant enhancement of existing cycle-accurate processor simulation methods. The integrated energy models were built using appropriately scaled baseline versions available from accurate power models maintained in the design of current generation processors. Our study revealed a drastic reduction in compiled application code path length, because of the massive elimination of unnecessary memory accesses. This fact, in conjunction with lower CPI resulting from more efficient register allocation and data manipulation leveraged via dynamic indirection, provides very significant execution time and energy consumption reductions.

5. REFERENCES
