Mapping Algorithms to the Amalgam Programmable-Reconfigurable Processor

Jeffrey J. Cook, Derek B. Gottlieb, Joshua D. Walstrom, Steven Ferrera, Chi-Wei Wang, Nicholas P. Carter
Center for Reliable and High Performance Computing
University of Illinois at Urbana-Champaign
{jjcook, dgottlie, walstrom, ferrera, cwang12, npcarter}@crhc.uiuc.edu

The Amalgam programmable-reconfigurable processor is designed to provide the computational power required by upcoming embedded applications without requiring the design of application-specific hardware. It integrates multiple programmable processors and blocks of reconfigurable logic onto a single chip, using a clustered architecture, similar to the one used on the M-Machine [1] to reduce wire length and delay and allow implementation at high clock rates. The clustered architecture provides tremendous flexibility, allowing applications to exploit parallelism at whatever granularity is best-suited to the application, while the combination of reconfigurable logic and programmable processors delivers much higher performance than could be achieved through programmable processors alone.

This abstract presents the results of our initial experiments in hand-mapping applications onto Amalgam. Five applications (IDCT, Rijndael encryption, nQueens, DNA sequence comparison, and image dithering) have been implemented, achieving speedups ranging from 8.7x to 23.2x over the performance of a single programmable cluster by using the complete resources of an Amalgam chip.

Figure 1 shows a block diagram of the Amalgam processor. Four programmable clusters, which act as independent programmable processors, and four reconfigurable clusters, which contain blocks of reconfigurable logic are integrated onto the chip and communicate with each other and the shared memory system through an on-chip network.

As shown in Figure 2, each programmable cluster contains a two-issue in-order processor that executes an instruction set based on the MIPS ISA. Each reconfigurable cluster contains a 32-row by 32-bit block of reconfigurable logic and a register file. Clusters are independent, relying only on local information to make instruction issue decisions. This eliminates the need for inter-cluster communication on a cycle-by-cycle basis, allowing Amalgam to be implemented at higher clock rates than architectures which require global communication on each clock cycle.

Operations executing on a cluster (programmable or reconfigurable) may only read from that cluster’s register file, but may target registers in any cluster’s register file. This provides a low-latency mechanism for inter-cluster communication that allows communication and synchronization to be combined, since a cluster will block on the execution of an operation that is waiting for data from another cluster in the same way that a conventional processor delays the execution of an instruction that is waiting for data from memory or for the result of another operation. Studies [2] have demonstrated that this register-based inter-cluster communication mechanism can provide significantly better performance than architectures in which clusters communicate only through the memory system.

Amalgam’s clustered architecture is extremely flexible. In this paper, we treat pairs of clusters (one programmable plus one reconfigurable) as independent processors, exploiting parallelism across the four pairs of clusters on an Amalgam chip. This is purely a software convention, and the clusters can be grouped in other ways or treated as eight independent processors.

Five applications have been written to demonstrate the performance of Amalgam: IDCT, Rijndael encryption, nQueens, DNA pattern matching, and image dithering. These applications were hand-coded in assembly and hand-mapped into the reconfigurable array. These applications exhibit diverse levels of partitioning granularity, frequency of synchronization and memory reference intensity. Each application has been written to benchmark various configurations of Amalgam by varying the number of clusters and the composition of clusters used from exclusively programmable clusters to programmable-reconfigurable cluster pairs.

Our implementation of the 2-D IDCT takes advantage of its decomposition into a series of 1-D IDCTs. For the programmable cluster’s 1-D IDCT, we implemented the Loeffler algorithm [3] in 32-bit fixed-point math, which requires only 11 multiplications. We employ the popular technique of distributed arithmetic to achieve a compact yet high-performance IDCT implementation in the reconfigurable version, with a total latency of 14 cycles; 13 cycles to bit-serially process the 13 bit fractional 2’s compliment inputs and 1 cycle to load the shift registers from the register bank. With two IDCT cores implemented on each reconfigurable cluster, although not completely independent, the total amortized cost of an IDCT is just 15 cycles, compared to a minimum of 52 cycles for computation alone on our programmable cluster, with an overall maximum speedup of 14.97 as shown in Figure 3.

The Rijndael block cipher [4] is a symmetric encryption algorithm that iterates using a round transformation, with number of iterations dependant on the block cipher size; here a 128-bit block size with ten iterations. Implementing Rijndael on the programmable clusters was straightforward; the single programmable cluster implementation was easily extended to two and four cluster versions as the round transformation is a parallel operation on each of the four columns in the block. Although a T-box configuration [4] is faster for a programmable...
cluster, an S-box implementation was used for a direct comparison with the programmable-reconfigurable cluster pair configurations. The greatest benefit in accelerating Rijndael in the reconfigurable cluster was the optimization of the Mix Column phase of the round transformation, which required only a single cycle and four rows of the array, resulting in maximal speedup of 16.9.

Our implementation of nQueens represents rows of the chessboard as bit vectors in memory. The algorithm iteratively generates configurations in a depth-first manner. Mapping of nQueens onto the reconfigurable clusters consisted of three modules: a board generator, a board safety checker, and a state machine for global control. Registers in the reconfigurable cluster were allocated to hold the inputs to the board generator and the board safety checker, and the state machine was responsible for moving the appropriate rows of the chessboard into and out of these registers as necessary to implement the algorithm. With unused registers available, the reconfigurable cluster held the chessboard not in memory but in local registers, eliminating all memory references during computation. More significant, though, was the ability of the state machine to use the result of a computation to determine which register to access in each step of the algorithm. This led to an impressive maximal speedup of 23.2 on four cluster pairs.

For DNA sequence comparison [5], the edit distance has been shown to be a convenient way to quantify the similarity of two sequences. This edit distance is defined as the minimum cost of converting one sequence to another through character deletions, character insertions and the substitution of one character for another. The edit distance is computed with a well-known dynamic programming algorithm as described in [6] which is used for both the programmable and reconfigurable implementations. A maximal speedup of 8.66 was observed.

The image dithering benchmark converts an image from a 256 color palette to a 6 color palette for each of the three color components of a pixel, red, green, and blue, resulting in a dithering from 2563 to 63 colors. To dither the palette, Floyd-Steinberg error diffusion [7] is employed.

Figure 3. Benchmark Results

Two primary configurations were used for our benchmarks. The first uses only programmable clusters and is referred to in the figures as *prog*. The second uses programmable-reconfigurable cluster pairs, which are referred to as *rec*. Each configuration then has a version that does inter-cluster communication only through memory and another version utilizing register-register writes when possible, denoted as *forwarding*.

Rijndael, as shown in Figure 4, exhibits trends found in all our benchmarks. The programmable cluster implementations scale with less than linear speedup, due in part to communication overhead. Enabling forwarding between programmable clusters, thus eliminating unnecessary communication overhead, provides a 17% increase in performance for two clusters and a 29% increase for four clusters. Utilizing programmable-reconfigurable cluster pairs, performance again increases dramatically, 36% for a single programmable-reconfigurable cluster pair, and 37% for a two cluster pair. With forwarding enabled, the quad programmable-configurable cluster pair configuration gained an additional 65% performance increase.

Figure 4: Rijndael Encryption Performance

Among the benchmarks presented, the maximum speedup by utilizing all eight clusters ranged from 8.7 to 23.2. In all cases, the availability of reconfigurable logic allowed for significant performance gains, with an average of 270% speedup versus configurations with the same number of programmable clusters. Additionally, the ability to utilize register-register writes (forwarding) provided, on average, an additional 21% speedup. Both of these results are very promising in showing the Amalgam clustered programmable-reconfigurable architecture as an attractive processor for future embedded systems.

Acknowledgements

This work was supported by the Office of Naval Research under grant N00014-01-1-0824.