MEMOCODE 2008 Co-Design Contest

Patrick Schaumont
ECE Dept Virginia Tech
schaum@vt.edu

Krste Asanovic
EECS Dept UC Berkeley
krste@eecs.berkeley.edu

James C. Hoe
ECE Dept Carnegie Mellon University
jhoe@ece.cmu.edu

Abstract

The second MEMOCODE hardware/software codesign contest invites participants to solve a practical hardware/software codesign problem within the time span of one month. The larger objective for this contest is to be a showcase of advances in co-design tools and methodologies, in combination with design ingenuity and creativity. In the second installment of the contest, we received 9 submissions. In this short writeup, we review this year's design problem, and we consider relevant contest statistics.

1 Introduction

The first MEMOCODE hardware/software codesign contest was organized in 2007. The motivations underpinning this contest and its general format are discussed in [1]. Briefly, the hardware software codesign contest engages participants to build a performance-optimized prototype on a platform of choice, starting from a reference implementation running on a Xilinx XUP2VP prototyping board. In the design contest, the major degrees of design freedom are the design flows, the tools and methodologies, and the creativity and expertise of the contestants. The early success of the 2007 event triggered the organization of the second MEMOCODE design contest in 2008.

The second MEMOCODE hardware/software codesign contest started on February 8 2008 with the announcement of a *secret* design problem (involving AES and sorting). Contestants had one month to prodice working HW/SW co-designed solutions. By the conclusion of the contest, 8 teams from around the world (including US, Europe and Asia) submitted a solution, out of the 27 teams that started. The contest has announced a total of three $1000 cash prizes in three different categories. Two $1000 cash prizes are offered for the Highest Performance Design and The Most Efficient Design. In addition, Xilinx is sponsoring a third special $1000 cash prize for the best entry employing a high-level design methodology.

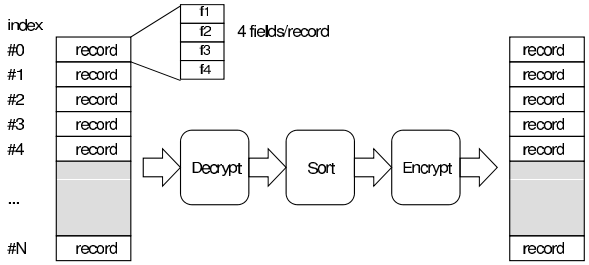

Figure 1. Cryptosorter Application

This year's contest was organized by Krste Asanovic, James Hoe, and Patrick Schaumont. The panel of 5 judges that evaluated the submissions included Kees Vissers (Xilinx), Satrajit Chatterjee (Intel), as well as the three organizers. The contest is sponsored by Nokia, Xilinx, Bluespec, and IEEE CEDA.

In the following, we describe the design problem, some observations on the design, as well as the final outcome[1]

2 Design Problem 2008

Overview. The objective of the 2008 MEMOCODE Hardware Software Codesign Contest is to sort an encrypted database of records as fast as possible (Figure 1). Sorting an encrypted database requires that each record is decrypted, that a proper index for each decrypted record is determined, and that each record is re-encrypted and stored at the proper index. The record length in the database is 4 words. A record consists of four fields $\{f1, f2, f3, f4\}$. Each field is one word long and encrypted. Encryption and decryption transforms the four fields of a record $\{f1, f2, f3, f4\}$ into four decrypted fields $\{g1, g2, g3, g4\}$. Each of $g1$, $g2$, $g3$, and $g4$ is one word long.

Sorting. The sorting key of a decrypted record is determined by weighted numerical ordering of the decrypted

[1]The published design problem and contest rules may be consulted at the contest website http://rijndael.ece.vt.edu/memocontest08/

1-4244-2417-7/08/$20.00 ©2008 IEEE

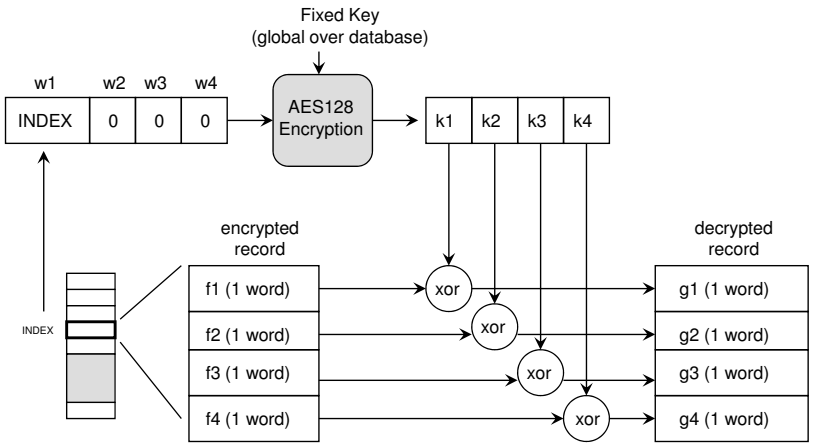

Figure 2. Encryption/Decryption Flow

words of the record. g1 is the most-significant word and g4 is the least significant word. g1 through g4 are unsigned numbers. The database does not contain any duplicate records, so that there is always a unique ordering possible. As an example, the following would be a valid sorting key relation: $\{10, 10, 1, 0\} > \{10, 10, 0, 1\}$.

Encryption and Decryption. Encryption and decryption is done using four keys $\{k1, k2, k3, k4\}$, each one word long. The encryption/decryption operations are performed using XOR operations: $\{f1, f2, f3, f4\} = \{g1 \oplus k1, g2 \oplus k2, g3 \oplus k3, g4 \oplus k4\}$. Based on this mechanism, encryption and decryption of records are identical operations. The four keys $\{k1, k2, k3, k4\}$ are determined using AES-128 (Advanced Encryption Standard [2]) as follows. The record index is encrypted with AES under a fixed, data-base wide key. The encrypted record index yields 128 bits of key material. These 128 bits are split in four words corresponding to $\{k1, k2, k3, k4\}$. The encryption/decryption process is illustrated in Figure 2.

System Testbench. The cryptosorter reference implementation includes a testbench. The testbench generates the content of the database using a pseudo-RNG with a 32-bit seed. The generating polynomial for the PRNG is $P(x) = 1 + x^2 + x^3 + x^{32}$. The testbench uses 8 different test cases. The overall execution time T of a design is defined by the geometric mean of the 8 individual test cases. The overall execution time of the reference implementation is expressed similarly as a geometric mean T_{ref}. The performance metric of a design is then expressed as a speedup factor $\frac{T_{ref}}{T}$.

The 8 testcases are partioned in two groups of 4 testcases each. The 4 testcases in each group differ only in the number of records. The four cases include 2^6, 2^{10}, 2^{14} and 2^{18} records respectively. The first group of 4 testcases uses the random testbench as specified above, with a random generator seed of `0x64646464`. The second group of 4 testcases uses a pathological testbench, defined by the elements of a matrix M as follows. Given a square matrix S with in-order elements in row-major order, then then M is a 90-degree clockwise-rotated version of S. For example, if the T is a 16-element matrix containing the numbers $\{0, 1, 2, .., 14, 15\}$, then M contains $\{12, 8, 4, 0, 13, 9, 5, 1, 14, 10, 6, 2, 15, 11, 7, 3\}$. For a database of size 2^Q, M becomes a square matrix with 2^{Q-1} elements on each side.

Several parameters are hardcoded in the testbench. This includes the record length (4 words), and the database-wide key (equal to `0xB01D FACE 0DEC 0DED 0BA1 1ADE 0EFF EC70`).

Performance Measurements and Restrictions. Contestants are required to obtain performance cycle counts using hardware cycle counting. The cycle counts for the reference software implementation are provided as hardcoded constants to ensure that the reported speedup factors are comparable among teams. The contestants are further not allowed to re-synthesize the design depending on the test case. A single FPGA bitstream configuration must be able to execute all 8 test cases. Contestants are allowed to measure each testcase three consecutive times in order to benefit from running with a warm cache.

Calibration. The contest awards both absolute performance as well as normalized performance. If the contestants use an XUP board (with a -6 XC2VP30) to enter the contest, their performance normalization factor is 1. If the contestants use a more capable platform, the normalization factor will be greater than 1. If they use a less capable platform, however, the normalization factor cannot go below 1. The resulting performance of the testcases then is determined as the product of the measured time and the normal-

ization factor.

The normalization factor is analytically specified and depends upon the number of FPGA's used, the number of CPU's used, the logic capacity of each FPGA, and a technology scaling factor for each CPU and each FPGA. The technology scaling factor is determined by executing a reference program on the target platform (CPU), or by finding the critical path for a reference hardware module (FPGA).

3 Some Observations on the Design Problem

The design problem attempts to strike a balance between challenge and feasibility. If the problem is too simple, the system design challenge disappears and the contest focuses on tiny technological differences. If the problem is too complex, no team will be able to come up with a solution within the limited design time.

By combining two well known problems into one (sorting and encryption), we have tried to create a design space in which the components are well known, but their interactions are unknown. For example, the reference implementation uses a naive encryption strategy that decrypts each record from external memory on an as-needed basis. This results in an increased computational load to cope with encryption/decryption. An alternate strategy would be to decrypt all the records once, then do the sorting, and finally re-encrypt all the records. This however results in strict sequentialization of system tasks, which may result in inefficient utilization of system resources.

Dealing with data moving operations is key in solving this problem efficiently. The size of the database exceeds the on-chip storage resources of the reference platform, so that off-chip memory access time, and use of on-chip storage for cache and local buffering, are important. The problem further allows the sorting strategy to be choosen freely. Sorting is classically thought of as a sequential activity, which introduces another system-level challenge.

The problem also included two *Easter Eggs*: optimization opportunities which are not explicitly stated in the assignment but which could potentially have a significant impact on the optimized result. The first Easter Egg is that the database key is fixed, and that the encryption/decryption keys can be evaluated and stored at compile time. Thus, it is perfectly possible to eliminate AES runtime encryption/decryption altogether at the expense of using more storage and introducing more I/O traffic. Since the encryption keys have the same size as data records, pre-calculated keys can be stored in a database as well, only doubling the resulting database in size. The second Easter Egg is that the contents of the records are not uniformly distributed. Contestants needed to examine the reference implementation to see this. 80% of the records generated by the testbench have the first field set to zero, 40% of the records have the first two fields set to zero, and 20% of the records have the first three fields set to zero. This enables contestants to potentially optimize the comparison operations. Several teams tried to exploit the two features mentioned above.

Table 1. Finishing Teams

Team ID	Affiliation	Size
sunita	Nanyang National University	8
uljana	Talinn University of Technology	4
brian	Brian Lindemann	1
vijay	AMD	1
marco	Politecnico Di Milano	13
rob	Old Dominion University	4
kermin	MIT	5
eric	Eric Simpson	1

Table 2. Performance Ranking

Team ID	Speedup	Platform	Language
kermin	1102.4	XUP	Bluespec
brian	100.2	XUP	C
marco	85.4	XUP	C + HDL
uljana	49.8	XUP	C + HDL
sunita (1)	41.1	XUP	C + HDL
vijay	33.0	XUP	C + HDL
rob	23.5	XUP	C + HDL
eric	12.8	XC2VP100	C + HDL
sunita (2)	11.0	XUP	C + Impulse C

4 Results

Ranking. Of the 27 teams that requested access to the reference implementation of the Cryptosorter design, 8 teams provided a solution before the contest deadline on 9 March. Table 1 lists the affiliation and configuration of each finishing team. Table 2 lists, for each team, the overall speedup, the platform used, and the design languages used. The overall speedup is defined by the geometric mean of the set of testbench configurations. The speedup is further normalized according to the platform calibration rules. The table rows are sorted according to the overall speedup. A surprising result is that the two best designs are not the result of hardware-software codesign in the traditional sense. The top design uses Bluespec (parallel design paradigm), while the second design uses C (sequential design paradigm).

All of the submissions were evaluated by a panel of 5 judges. The judges verified if each design worked according to the testbench specification, and they verified the perfor-

Table 3. Elegance Ranking

Rank	Team ID
1 (tie)	team kermin, team vijay
3	team brian
4	team marco
5	team sunita (2)

mance of each design. In addition, each judge evaluated the elegance of each design. Elegance is a subjective appreciation, and it considers factors such as cleverness of the algorithm, exploitation of parallelism, quality and clarity of the design documentation, and quality and clarity of the source code. The individual rankings were then combined, by majority voting, into a top-5 of elegant designs. The resulting elegance ranking is listed in table 3.

Awards. To determine the overall ranking, the performance ranking and elegance ranking are combined. This overall ranking is then evaluated against the award criteria: Highest Performance Design (maximum absolute performance), Most Efficient Design (best performance relative to the platform), and Best Figures of Merit Using a High Level Language.

Based on the highest performance ranking and a top-ranking in elegance, team kermin (MIT) becomes the winner of the $1,000 award for Higest Performance Design. Because all top-level teams use the same XUP2VP platform, the contest ranking for Most Efficient Design award is identical to the ranking for Highest Performance Design award. However, the contest rules specifically support only a single award should a team win in both categories. The prize for Most Efficient Design can therefore not be awarded to team kermin.

The judging panel concluded to award a $500 honorable mention to team vijay, based on the consensus that this design has superior elegance (although this design does not qualify for Highest Performance nor Most Efficient).

The third award, for Best Figures of Merit Using a High Level Language was decided separately under the figures of merit defined by Xilinx, the sponsor of this award. In this category, team kermin was the winner of the $1,000 prize.

The MEMOCODE 2008 Hardware Software Codesign Contest was an exciting event that generated significant outside interest. The contest demonstrated that Hardware Software Codesign remains a challenging topic. The number of finishing teams quadrupled over last year, which opens up the road towards next years' challenge.

5 Acknowledgements

We would like to acknowledge the support of the IEEE Council on Eletronic Design Automation (CEDA), Nokia, Xilinx, and Bluespec. We would like to acknowledge our two outside judges, Kees Vissers from Xilinx and Satrajit Chatterjee from Intel.

References

[1] F. Brewer, J. C. Hoe, "MEMOCODE 2007 Co-Design Contest", in *Proceedings of the Fifth ACM-IEEE International Conference on Formal Methods and Models for Codesign*, p. 91-94, Nice, France, June 2007.

[2] FIPS Publication 197, "Advanced Encryption Standard (AES)", Online at `http://csrc.nist.gov`.