MEMOCODE Hardware/Software Co-design Contest

Cryptosorter

Tallinn University of Technology team

documentation

Uljana Reinsalu

Sergei Devadze

Artur Jutman

Anton Chertov

Tallinn

March 2008

Note

Our solution is the minimum, what we could do. However, it represents a working hardware/software design, so we would send it just for statistics.

Task definition

The objective of this contest is to sort an encrypted database of records as fast as possible. 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 is 4 words. A record consists for 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.

Task analysis and methodology

The assignment can be clearly partitioned into two tasks: AES crypto algorithm and data sorting.

The most computation intensive is the AES encryption algorithm. Therefore it was decided to implement it on hardware. Reuse of IP cores is a trend and state of the art system design, therefore we decided to use existing open source IP core for AES 128 encryption. We assume that this IP core is written well and optimized for fast computation, thus it is not the primary goal for optimization.

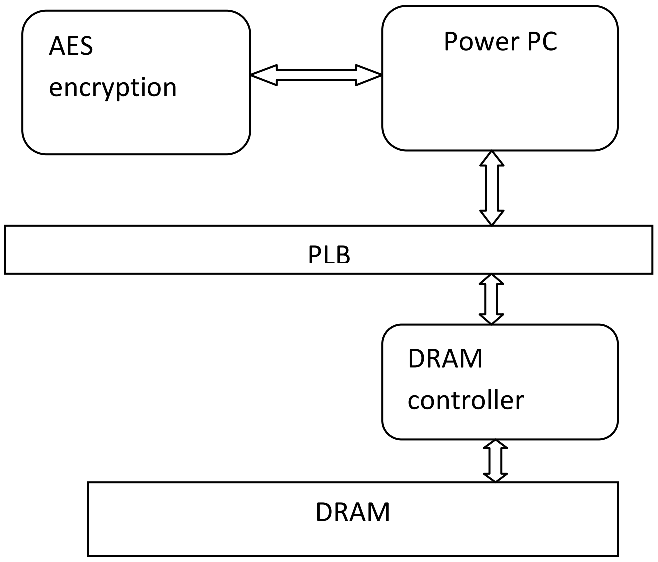
The sorting is the opposite case. Since the speed of a particular sorting algorithm may depend on the structure and statistical distribution of the input data, one or another specific algorithm will be more efficient. Hence, we decided to implement the sorting algorithm in software so that it can be easily and quickly modified when specially required. The importance of this decision was proven by the fact that the contest's test-bench was changed several days prior to the submission deadline.

According to the task we have a lot of data which can only be fully stored in DRAM. For sorting we need a lot of data exchange with memory and thus communication with memory should be also optimized.

Since we currently have no high-level synthesis tools to use we had to restrict our design methodology to the ideas described above.

General structure of implementation

We decided to store keys in the memory so that we can generate the keys once and get them



from memory by index. We decided to make it this way, because the task description does not address this issue. The keys are generated by sending proper index from processor to AES encryption core, which is realized on hardware and AES module returns generated key to processor, which stores it to proper location. Sorting is done in processor.

Sorting algorithm

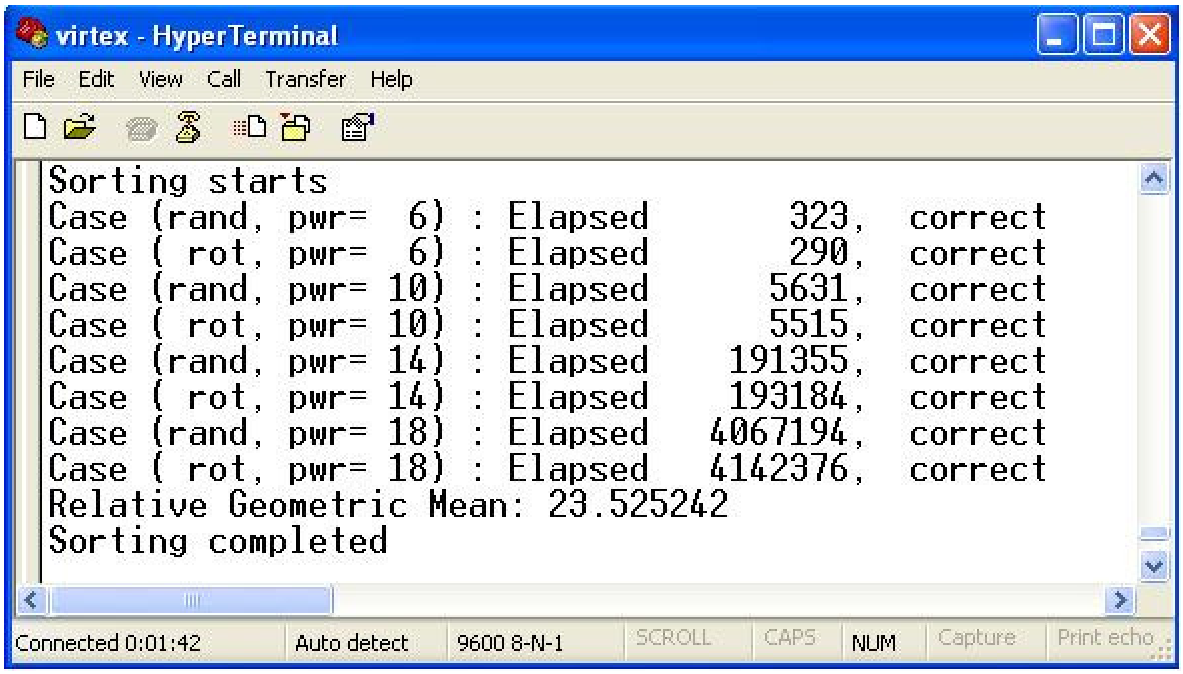
To choose the most effective sorting algorithm, we analyzed the data we need to sort. Since it was written in the task description that data would be generated by LFSR. This means that each data word will be unique and uniformly distributed over the numerical space from 1 to 2^{32}-1. Also having analyzed statistical distribution of data from test-bench, we noticed that in approximately 25% of the cases the first word is replaced by zeros, ~12% of data has zeros in 2 first words, and ~12% have zeros in 3 first words while the remaining does not contain zeros for whole word. Thus we decided to start sorting by dividing data into 4 groups according to the distribution of zeros. Each group will then be sorted separately. Since the first non-zero word in each record is unique because of the LFSR properties, we do not need to sort all 4 words, only the first one. We also made algorithm of being aware of any other data. The algorithm can handle any arbitrary data of four 32-bit words. The analysis of data distribution is performed during data decryption and then either Comb11 sorting is used for groups' data sorting or Comb11 sorting algorithm is used for whole data sorting. Moreover, hashing of n first bits of word is implemented before data sorting. Where n bits is the power of 2 of amount of all data and n can be maximum 16. Initially, hashing was planned to be implemented in hardware, but as we did not get in time correctly working DMA controller hashing is left in software.

Results

Results without hashing in software and taking into account only data distribution according statistics (project WithoutHashing):

```
Sorting starts
Case (rand, pwr=  6) : Elapsed       216,  correct
Case ( rot, pwr=  6) : Elapsed       211,  correct
Case (rand, pwr= 10) : Elapsed      3660,  correct
Case ( rot, pwr= 10) : Elapsed      3718,  correct
Case (rand, pwr= 14) : Elapsed     66161,  correct
Case ( rot, pwr= 14) : Elapsed     69361,  correct
Case (rand, pwr= 18) : Elapsed   1215993,  correct
Case ( rot, pwr= 18) : Elapsed   1294237,  correct
Relative Geometric Mean: 49.819154
Sorting completed
```

Results with hashing in software and taking into account any data distribution (project WithHashing):



```
Sorting starts
Case (rand, pwr=  6) : Elapsed       323,  correct
Case ( rot, pwr=  6) : Elapsed       290,  correct
Case (rand, pwr= 10) : Elapsed      5631,  correct
Case ( rot, pwr= 10) : Elapsed      5515,  correct
Case (rand, pwr= 14) : Elapsed    191355,  correct
Case ( rot, pwr= 14) : Elapsed    193184,  correct
Case (rand, pwr= 18) : Elapsed   4067194,  correct
Case ( rot, pwr= 18) : Elapsed   4142376,  correct
Relative Geometric Mean: 23.525242
Sorting completed
```

What could be done

First thing, what could be done is faster SDRAM access from/to AES IP core. Unfortunately, we could not get in time correctly working DMA controller. Also we had very small amount of resources – only 3-4 person-weeks and the time went also for studying the platform. Secondly, we wanted to make hashing in hardware. It was initially planned, however without DMA controller it was not possible. Also we planned to have at least 2 AES IP cores, generating keys in pipeline. It doesn't make sense to talk about other improvements without checking optimization options mentioned above.