

The Advanced Encryption Standard on the HC 36m Reconfigurable Computer

Pradeep Kancharla and Duncan A. Buell
Department of Computer Science and Engineering
University of South Carolina
Columbia, South Carolina 29208
 [{kancharl|buell} @cse.sc.edu](mailto:{kancharl|buell}@cse.sc.edu)

ABSTRACT

We describe an implementation of the Rijndael algorithm on the HC 36m reconfigurable computing machine developed by Star Bridge systems. Rijndael is the symmetric key encryption algorithm that was adopted as the Advanced Encryption Standard (AES) by NIST. The HC 36m is a reconfigurable computer that has two 2.4GHz Xeon processors and a reconfigurable resource comprising five Xilinx Virtex II 6000 and two Virtex II 4000 FPGAs. The implementation is done in Viva, an environment developed by Star Bridge Systems, the vendor of the HC 36m. Viva provides the user a graphics editor to develop applications that can be synthesized and mapped to the hardware.

Introduction

In recognition of the fact that cryptography has become an essential part of communications security for electronic commerce, NIST has promulgated standards for cryptography for more than 20 years. When it became clear that DES (Data Encryption Standard) had become obsolete, a successor, the Advanced Encryption Standard, was adopted in the fall of 2001 [1,2]. In this paper we discuss the implementation of this algorithm in hardware on the Star Bridge Systems HC 36m reconfigurable computing machine [3]. Other implementations of AES in hardware have usually focused on minimizing the power consumed or the logic needed, as would be appropriate for a smart card application. The basic goal behind this implementation in hardware is to use the relatively large FPGA resources of the HC 36m, thus not to be constrained by logic, and thus to obtain significant speed up in terms of execution time primarily through parallelism and pipelining.

The Advanced Encryption Standard

Rijndael is a block cipher with possible block and key lengths of 128, 192 and 256 bits. The block to be encrypted and the key can be of different lengths. The encryption is comprised of a variable number of rounds (determined by the key and block lengths) with each round containing four stages: ByteSub, ShiftRow, MixColumn and RoundKeyAddition (in the last round, the MixColumn is omitted). An initial key is expanded to form an expanded RoundKey based on the number of rounds. Since Rijndael is a symmetric cipher, decryption is just the inverse of the encryption.

We have initially implemented AES for a 128-bit block and key size. The 128 bits of the block are viewed as a 4x4 double array containing 16 cells of eight bits each called a State. The key can also be viewed in the same way. It is to be noted that AES was designed to be easily implemented on smart cards with limited computing capability, so most of its operations work on bytes independently.

There are two basic approaches for putting AES in hardware. AES can be viewed as a loop on number of rounds with four (or three in the last round) stages. One could therefore implement each stage once in hardware and iterate over the rounds. This would take the least hardware but obviously doesn't allow pipelining. An alternative would be to unroll the loop and pipeline the computation using multiple instances in hardware of each stage. This allows us to pipeline the computations of different states. In either of the possible implementations one can, inside some of the stages, obtain further parallelism and speed, since the byte-oriented nature of AES allows for independent computations on the bytes. However, this last step in parallelizing can lead to a significant increase in hardware utilization for large block sizes due to the need, as described below, to replicate a large number of lookup tables.

The simplest of the four stages is the ShiftRow, which as its name implies is simply a row-by-row permutation of the block. Its hardware implementation is straightforward and full parallelism is obvious. Next simplest is the RoundKey addition, which is essentially done with XORs. The RoundKey addition takes 329 slices with a maximum pin delay of about 22ns.

The ByteSub and MixColumn stages use $GF(2^8)$ arithmetic. In software and in our initial hardware implementations, this arithmetic (addition and multiplication, respectively) is done with lookup tables. The lookup tables are implemented using the on-chip memory, reading the values from files and synthesizing these values into memory as constants. A lookup table of the necessary 256 values takes 160 slices and has a maximum pin delay of around 17ns.

In these stages we operate independently on each cell of the state. Since these operations are independent of each other, they can either be done iteratively using the same lookup tables or in parallel with separate lookup tables for each byte. In the iterative case, the output of each iteration is registered and passed to the next stage after the completion of all the iterations. This implementation uses fewer resources in terms of silicon--786 total slices for ByteSub with a maximum pin delay of nearly 17ns. The MixColumn stage takes 967 slices and has a maximum pin delay of around 17ns when done in this manner.

Although the HC 36m has substantial reconfigurable resources with its four Virtex II 6000 chips, implementing both ByteSub and MixColumn entirely in a byte-parallel manner would require about 150 lookup tables, each of which requires 160 slices. The 24,000 total slices would be about $\frac{3}{4}$ of a Virtex II 6000 chip. This will first of all limit our ability to unroll the rounds into separate instances of hardware, since complete unrolling would not be possible with only four chips. We note that the use of lookup tables facilitates a software implementation, since the $GF(2^8)$ arithmetic is not supported by the instruction set architecture of microprocessors and since much of the use of the lookup tables has to be done sequentially in software. Given the ability in using a reconfigurable computing machine to design one's own instructions, there may well be an advantage to implementing the arithmetic directly in logic, and we are exploring this possibility.

Finally, as we unroll the loops to pipeline the process, we must spread the design across multiple chips and deal with the problem of moving data between different chips fast enough for the computational part of the design. On the HC 36m we can move data from one FPGA to another using a total of 50 pins and one bit per pin per clock tick (including control bits). We are presently exploring this unrolling and pipelining so as to make optimal use of the entire silicon resource.

We are also implementing an end-to-end application, reading a file in blocks, encrypting the blocks, and then decrypting them in any of the several standard modes (Electronic Code Book, Cipher Block Chaining, etc.), and writing the output to a file. The file handling is done using the Microsoft COM objects on the host processor and the encryption and decryption are done on the HC 36m hardware. Since we have a VHDL model for this computation, we will then be able to compare Viva as a complete tool for implementing an end-to-end application.

References:

- 1) J. Daemen and V. Rijmen. The Design of Rijndael: AES, The Advanced Encryption Standard, Springer Verlag, Berlin, 2001.
- 2) National Institute for Standards and Technology. FIP 197: Announcing the Advanced Encryption Standard, Nov. 26, 2001. <http://csrc.nist.gov/encryption/aes/index.html>
- 3) Star Bridge Systems, Inc., web site, <http://www.starbridgesystems.com>