DES Developed In Handel-C

M Mylonas, D.J Holding, K.J.Blow
Aston University

Abstract: This paper aims to give experiences gained in developing DES using the Handel-C development suite called DK1 from Celoxica. Handel-C is used to design circuits to be programmed onto FPGAs. It was found that care had to be taken when describing a circuit using Handel-C, as several descriptions of the same hardware produced very different results when compiled. The author could not achieve the same circuit speed and size for DES as that of an implementation given in a Celoxica white paper [1]. Neither the author's circuit size or that given in [1] could match that of an implementation of DES in VHDL using the Xilinx Foundation Series software.

1 Introduction.

The work being carried out by the author concerns the speeding up of routers in computer networks, be they in the internet or mobile ad hoc networks. Special reconfigurable hardware called Field Programmable Gate Arrays (FPGAs) are used to speed up router functions that are traditionally executed in software. The FPGAs can implement in hardware the function of a user's choosing.

The circuit that is implemented on a FPGA has traditionally been designed using a hardware description language such as VHDL. This type of language needs an understanding of hardware and is therefore used by hardware engineers. More recently other languages have been developed that are based on traditional software programming languages such as C. Handel-C is one such language and aims to allow designers with a limited hardware background to design a circuit for a FPGA.

The DES encryption algorithm was implemented in both VHDL and Handel-C by the author. DES was chosen for implementation because it is the standard encryption algorithm used in networks and is sufficiently complex to provide a good test for designing circuits using the hardware description languages.

This paper aims to give experiences gained in developing DES using the Handel-C development suite called DK1 from Celoxica. DES was verified in hardware on a FPGA using the Celoxica RC1000 development board, which contains a Xilinx Virtex XCV2000E-6 FPGA.

The rest of the paper will be structured as follows: Section 2 will give a brief introduction to the DES encryption algorithm, Section 3 will give implementation results and analysis using Handle-C, Section 4 will give the conclusions.

2. The DES Encryption Algorithm.

DES is a block cipher. It operates on blocks of 64-bits in size. A 64-bit input block of plaintext will be encrypted into a 64-bit output block of ciphertext. It is a symmetric algorithm, which means the same algorithm and key are used for encryption and decryption. The security of DES rests in the 56-bit key. The DES algorithm functions as follows. The plaintext block is taken in and put through an initial permutation. The key is also taken in at the same time. The key is presented in a 64-bit block with every 8^{th} bit being a parity check. The 56-bit key is then extracted ready for use. The 64-bit plaintext block is split into two 32-bit halves, named the right half and left half. The two halves of the plaintext are then combined with data from the key in an operation called Function F. There are 16 rounds of Function f, after which the two halves are recombined into one 64-bit block, which is then put through a final permutation to complete the operation of the algorithm and a 64-bit ciphertext block is outputted. A block diagram of the Function f is shown below in Figure 1 [2].

As mentioned before the plaintext input is split into two 32-bit halves, L and R. The right half is expanded to 48 bits through a permutation, the extra bits being duplicates of certain input bits. The expanded right half is then XORed with 48 bits selected from the key. The key bits are selected by splitting the 56-bit key into two 28-bit halves, which are shifted to the left by either one or two bits depending on the number of the round. The two shifted halves are then recombined to form the key to be used for the next round, and is also put through a compression permutation to obtain the 48 bits to be XORed with the expanded permuted right half of the plaintext. The result of

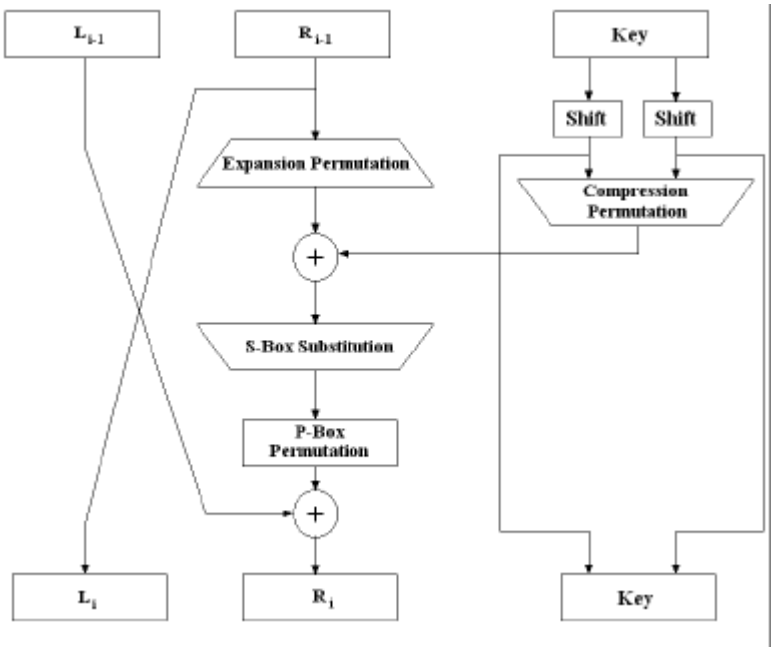

Fig. 1 Function f (One Round of DES) [2]

the XOR operation is then split into eight 6-bit segments which are fed into eight substitution boxes (S-Boxes). Each S-Box works in the same manner – it uses the 6-bit input to address a lookup table and output a 4-bit result. The S-Box outputs are then joined together to form a 32-bit block, which is put through a permutation (P-Box). The output of the P-Box is then XORed with the left half of the plaintext to make the new right half for the next round. The right half then becomes the new left half for the next round. The Function f is repeated in 16 rounds of DES.

DES is better suited to hardware implementation, such as in FPGAs, rather than software implementation due to the odd sizes of the operands in the Function f, and the shift operations. The hardware implementations of the individual operations that make up Function f are very simple, consisting of two XOR operations, lookup tables (LUTs), shifts, and wire routing operations (permutations, expansions, compressions).

3. Implementation Results And Analysis Using Handle-C.

Interesting results were obtained when developing DES in Handel-C. The circuits produced by the compiler varied greatly depending on how they were described in Handel-C. The main component of DES that was of interest was the S-Box. This is comprised of six LUTs. In VHDL these LUTs were described using a case statement, which provided the appropriate output for each input. When the Handel-C design of DES was begun, case statements were again used to describe the LUTs. The resulting circuit was found to be many times bigger and slower than that obtained from the VHDL compiler. It was found that the Handel-C compiler interpreted the case statement as logic rather than as a bank of memory and constructed a huge circuit as a result. The LUTs were then described as arrays. This reduced the circuit size greatly, and also increased the speed of the circuit. Using arrays did not achieve the same circuit size as that of the VHDL implementation, with the Handel-C implementation still being much greater. Finally, the LUTs were described as a type called ROM, which is available in Handel-C. Using this ROM type the circuit was reduced in size again and the speed also increased, giving the best results of the three methods used to describe the LUTs. The Handel-C circuit is now approximately equal in speed to the VHDL circuit but still considerably larger in size. Celoxica (the makers of Handel-C) published a white paper [1] on their implementation of Triple-DES. The paper also included implementation results for single DES, which were compared to the author's results. The results from [1]

show that Celoxica was able to achieve much faster circuit performance with a smaller circuit size, but the circuit size was still greater than that of the VHDL implementation. Table 1 below gives a comparison of the results.

TABLE 1 Comparison Of DES Implementation Results

CIRCUIT	NO. OF CLB SLICES	MAXIMUM CLOCK
Pipelined DES VHDL	2,524	68.999MHz
Pipelined DES Handel-C Static Array LUTs	10,327	40.548MHz
Pipelined DES Handel-C Static ROM LUTs	3,813	74.145MHz
Pipelined DES Handel-C Celoxica Implementation	3,025	101MHz

The circuit size is given in CLB slices, which are the reconfigurable blocks that make up a FPGA. The XCV2000 chip used has 19,200 CLB slices. The maximum clock speed of the circuit is given in megahertz. The implementation of DES given in the table is a pipelined version, which has hardware present for each round of DES to give maximum performance. There is not a pipelined version of DES with the LUTs described as case statements because the DK1 Handel-C compiler was unable to compile such a circuit. This is not unexpected when looking at the non-pipelined implementation results which have only one round of DES in hardware – the Handel-C compiler gave gate counts of over 50,000 for the implementation with case statement LUTs, compared to 15,000 for the implementation with LUTs described as ROM.

It can be seen that the Celoxica implementation gives the best circuit performance with 101MHz, which is over 25MHz faster than the author's implementation in Handel-C and over 30MHz faster than the author's VHDL implementation. The VHDL implementation gives the smallest circuit size in terms of CLB slices with a figure of 2524 slices. A big difference can be seen between the Handel-C implementation with LUTs described as arrays and the rest of the implementations – the circuit size is approximately three times bigger and the circuit speed approximately twice as slow. The author's implementation using ROMs was 800 slices bigger and approximately 25MHz slower than that of the Celoxica implementation, which also used ROMs to describe the LUTs. The reason for this is unknown, as the author developed DES using similar Handel-C descriptions to those described in Celoxica's white paper [1].

7. Conclusions.

Handel-C is not as mature as VHDL. Great care has to be taken when describing circuits. Although working circuits can be produced from different descriptions, circuit size and speed can vary greatly. The author could not match the results for a pipelined version of DES described in [1] – the author's circuit was approximately 30% slower and 20% larger. The VHDL description of pipelined DES was approximately the same speed as the author's Handel-C description, but gave the smallest circuit size in terms of CLB slices. Having a hardware background, a white paper, and a VHDL implementation for comparison the author was able to implement a decent version of DES on the Xilinx Virtex FPGA. The Handel-C compiler does not automatically infer an efficient ROM circuit for the LUTs from the most commonly used type of array. The designer has to specifically use the type ROM in order to achieve this. It would be hard to imagine someone without a hardware background and materials for comparison being able to achieve a similar level of performance.

References.

[1] Celoxica Ltd, "Triple DES Hardware Design And Implementation In One Week With Only One Software Engineer" White Paper, Version 1.0, 2001.

[2] B. Schneier, Applied Cryptography, Wiley, 1996, pp 270-278. ISBN: 0-471-11709-9