DES Encryption and Decryption on the XC6216

XAPP 106 February 2, 1998(Version 1.0)

Application Note by Ann Duncan

Summary

This note describes the design and implementation of DES (Data Encryption Standard) encryption/decryption using the XC6216

Xilinx Family

- XC6200

Demonstrates

- High performance cryptography
- Use of FastMap interface for register access
- Suitability of 6200 for register rich design
- Codesign using the XC6200DS PCI Board
- Hardware/software co-design process
- CBUF clocking

Table of Contents

Introduction

In the field of electronic data transfer, the application of secure cryptographic functions is becoming increasingly important. This note describes the design and implementation of encryption/decryption methods on the XC6216. The designs are realized using XC6200 Development System.

Encryption and decryption of DES require very similar designs. Encryption is described in this document, along with the decryption design evolved from it. This DES encryptor is a pipelined design, where high clock speed is essential for the maximum throughput of plaintext data. The XC6200, being rich in registers, is ideal for this.

A codesign approach supported by the XC6200DS PCI board can be used. The XC6200DS allows a design to be debugged directly onto the XC6216 under the control of a C++ program. Thereby allowing both hardware and software design to take place in parallel.

The DES Function

For a full description of the DES algorithm implementation read Schneier [1]. A summary is given here. Data is encrypted into blocks: 64 bits of plaintext produce 64 bits of ciphertext. The algorithms for encryption and decryption are exactly the same and use the same 56-bit key. For encryption the key is scheduled in one direction; for decryption the scheduling is reversed.

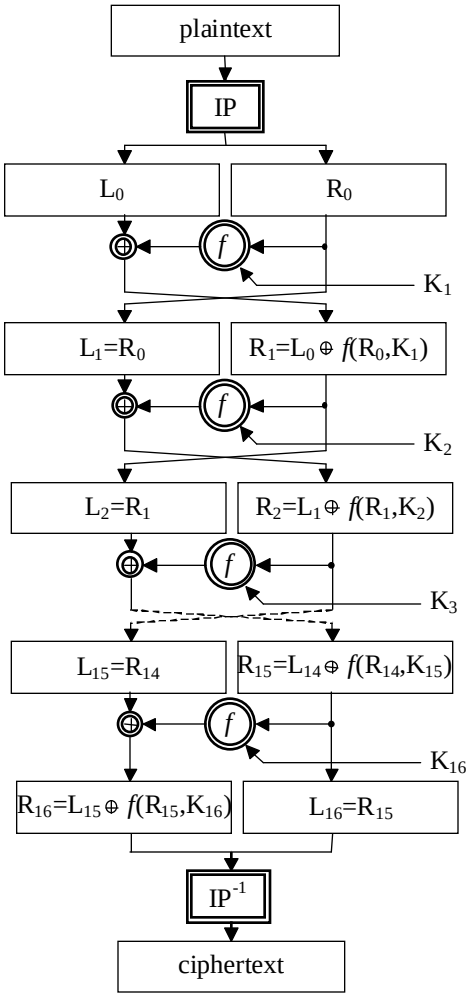

Figure 1: DES flow

The DES encryption algorithm

Step 1 takes plaintext and reorders the bits according to the Initial Permutation IP. See Fig 1. The block splits into a left half and a right half each of length 32 bits. The right half is then combined with the key and a function *f* is applied. The result of this is XOR'd with the left hand block and the two halves are then swapped. The process is then repeated fifteen times, each time applying different versions of the key to the right half. At the end of 16 rounds the left and right halves are recombined and the inverse of the initial permutation, IP^{-1}, is applied to complete the algorithm.

The Function f

The function *f* first expands the 32 bit data R to 48 bits via an expansion permutation. The shifted and permuted key is then XOR'd with the data. This value is pushed through 8 S-boxes to generate 32 new bits; these are then permuted. Details of these permutations and S-box values are given in Appendix A.

Shifting the Key

At each round a different version of the key is applied. Keys are generated as follows.

At each round, a different version of the key is applied. The key is split into a left half and a right half. Each new version is generated by applying a circular shift to each half of the key, shifting it either one or two places depending on the round. During encryption, shifting is to the left. During decryption the key versions must be applied in the opposite order so the key is shifting to the right. At the end of the 16 rounds the key has been shifted back to the original position ready for the next block of plaintext.

Implementation of DES for the XC6216

The Encryption Design

The DES Encryption algorithm can be implemented as a pipelined design. As the initial and final Permutations perform no cryptographic operation, they can be eliminated from the design. Pipelining through one round of the design is six stages long, allowing six different keys to be applied to six different blocks of data at once. This design could also be used to encrypt six different blocks of data with the same key. Fig 2 shows the structure of the design.

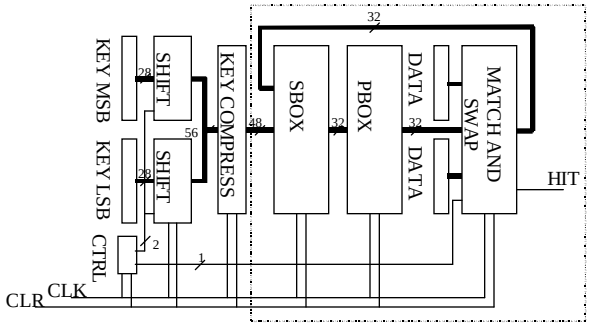

Figure 2: Structure of DES Encryption Design

The logic in the dotted box (Figure 2) implements one round of encryption as a 6-stage pipeline. Blocks KEY LSB and KEY MSB are 28 bit registers. The operation of the circuit is described as follows. Six keys and six blocks of plaintext are written to registers. A CBUF clock, controlling the clocking of the data into the design, is attached to the data and to the key registers. Writing to either of these registers from the software control program generates a clock pulse. After 12 CBUF clock pulses the clock is switched, by means of a MUX, to GCLK and the design become free running for the following 90 clock cycles.

Six keys are applied to the plaintext data for 16 rounds. A shift control bit (dependent on the round) is generated and ensures that the correct key version is ready to be applied at each round. Once the 16 rounds are complete the ciphertext can be read out. The next batch of keys and data can then be written to the design and the process repeated. The scheduling for the keys is given in Appendix A.

The Decryption Design

The design to perform decryption works in exactly the same way as that for encryption. Ciphertext is written to the data input registers and the plaintext is generated. The only differences are in the key Shift block. In decryption the key permutations must be applied in reverse order. This implies that they must be shifted to the right instead of the left. See Appendix A. The Key Shift block is implemented as shown in Fig 3.

Key Shift

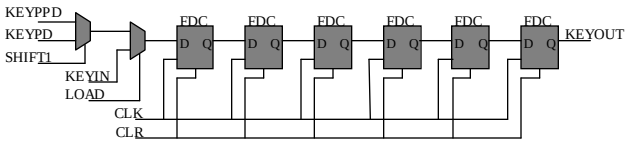

Figure 3: The Key shift block.

The KEYP signal is the KEYOUT of the adjacent key bit. The KEYPP is the next but one key bit. The KLOAD and SHIFT1 signals are each valid for six clock cycles, which is the delay through the pipeline for one round. Six registers delay each new version of the key for six clock cycles. This is the pipeline delay through one round of DES before applying the next version of the key. These six different keys and six different lots of data can be processed simultaneously.

The S-Boxes

Within S-boxes a 6-bit input is used to index a value in the 16x4 table. The first and last bits are a row index and the middle four bits are a column index. The value indexed gives a 4-bit output. Appendix A shows the S-Box tables.

Example: If the 6-bit input to S-Box 1 is 101100, then the row selected would be row 2 and the column selected would be column 6. The output would therefore be 0010.

		Binary Equivalent of No.			
I1I0	No.	D3	D2	D1	D0
00	13	1	1	0	1
01	6	0	1	1	0
10	2	0	0	1	0
11	11	1	0	1	1
	F(I1,I0)	XNOR	NOTI1	OR	XNOR

Figure 4: Table showing logic design for S-Box

The S-Box numbers are gathered into groups of 4. In this example, 13, 6, 2 and 11 from row 2 of S-Box 1 would be grouped together. Table 4 shows the S-Box

values, D3..D0, are generated from the input bits I0 and I1. The bits I2-I5 are used to select the correct S-Box value via a MUX tree. The 64 constants in each S-box require 64 gates and 60 MUXs for selection. For all 8 S-Boxes 992 function units are required. As registers can be mapped to function units with XC6216, the design can be highly pipelined to optimize performance at no extra cost in area.

Advantages of the XC6216

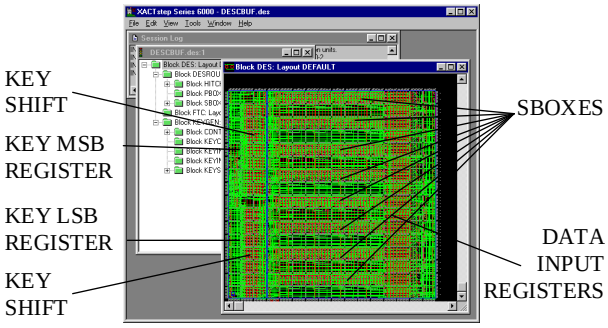

Figure 5: XACT6000 layout of Encryption Design

FastMap™ Interface

The FastMap™ Interface allows addressing to any register or function unit output. This means that no extra routing is required to write to the input registers and access results. So the location of such registers can be in the center of the design, far from IOBs and pin connections e.g. the data registers. See Fig.5.

Fast Debugging Cycle

XC6200 allows design iterations to take place rapidly. Turnaround time for this design, from editing the schematic, through generating the Edif, Placement and Routing using XACT6000, and debugging and testing the design operation on the XC6200DS PCI Board, was of the order of 10 minutes.

Codesign Development and Debugging

The XC6200 Development System PCI board provides an excellent platform for developing designs of this type [3]. Debugging is performed by running the design directly on the XC6216. All function unit outputs can be probed from the software. The board can be clocked in three different ways: free-running mode, single-stepped mode or it can be software clocked.

Clocking Scheme

During data input and output clocking is by means of a CBUF. Each write to a key or data register generates a clock pulse. A counter switches a mux to select a free-running clock after 12 clock cycles i.e. when all the keys and data are written. The free-running clock runs for 16 rounds through the S-Boxes before the mux switches again to select the CBUF clock allowing controlled reading of results and writing of the next round of data.

This clocking scheme suits designs where data is written and read via software and requires software controlled clock but fast clocking is required for intensive data processing.

Performance

Timing Analysis performed on the two versions of the design produced the following data on critical paths.

Encryption

Operation	Clock cycles	Seconds
Writing keys	24	1µs
Writing plaintext	24	1µs
Reading ciphertext	24	1µs
Free-running DES rounds	90	3.72µs
Total	**162**	**6.72µs**

Figure 6: Timing Evaluation for Encryption Design

Device	Blocks per second	Mbytes per second
XC6216	893×10^3	6.25
XC6264	3.572×10^6	25

Figure 7: Possible data throughput for XC6216 & XC6264

Critical path was calculated by the XACT6000 Timing Analyzer to be 41.68ns. The design would run at a maximum clock frequency of 23MHz. Data input and output times would be dependent on the system in which the XC6216 device was resident. In this case, the software control program determined the speed

of data input and output and therefore the speed of clocking during those times. Otherwise for 90 of the 102 clock cycles of one round, the design could be run at maximum clock frequency of 23MHZ. These 90 clock cycles would require 3.72µs.

The fastest speed at which data could be written to the registers would be at a rate 2 GCLK cycles per write [4]. Two writes are required per key and per plaintext value because the maximum size of data is 32bits. Therefore another 24x4 GCLK cycles are needed for writing keys and plaintext.

After encryption or decryption, the results must be read from the output. Alternate reading and writing of new data and processed data can occur. Reading 6 output values, at two reads per value and two clock cycles per register read, would require a further 24 clock cycles. The table in Fig.6 summarizes this.

This equates to an encryption rate of $893x10^3$ data blocks per second or 6.25Mbytes per second. See Fig 7.

Decryption

Timing Analysis on the decryption design gave a critical path of 54.46ns which is equivalent to a maximum clock frequency of 18.4MHz and a theoretical maximum data throughput of 4.8Mbytes per second. Again data input and output controlled the speed of clocking during these times and this is system dependent.

With both of the designs, it would be possible to improve the delay on the critical nets by hand routing them using XACT6000 custom route tool.

Summary

FPGA designs for DES encryption and decryption meet rquirements for increased security in data transfer. The XC6200 family is well suited to accelerating designs of this type

Reference

[1] Applied Cryptography 2nd Edition Protocols Algorithms and Source Code in C, Schnier Bruce. Wiley.

[2] http://limestone.uoregon.edu/deschall.html

[3] XC6200DS PCI Board Documentation

[4] XC6200 Data Sheet

Appendix A

The S-Boxes

S-Box 1

14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7,
0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8,
4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0,
15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13

S-Box 2

15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10,
3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5,
0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15,
13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9

S-Box 3

10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8,
13,7,0,9,3,4,6,10,2,8,5,14,12,11,15,1,
13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7,
1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12

S-Box 4

7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15,
13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9,
10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4,
3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14

S-Box 5

2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9,
14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6,
4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14,
11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3

S-Box 6

12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11,
10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8,
9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6,
4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13

S-Box 7

4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1,
13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6,
1,4,11,13,12,3,7,14,10,15,6,8,0,5,9,2,
6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12

S-Box 8

13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7,
1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2,
7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8,
2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11

P-Box Permutation

15,6,19,20,28,11,27,16,0,14,22,25,4,17,30,9,
1,7,23,13,31,26,2,8,18,12,29,5,21,10,3,24

Key Compression

13,16,10,23,0,4,2,27,14,5,20,9,
22,18,11,3,25,7,15,6,26,19,12,1,
40,51,30,36,46,54,29,39,50,44,32,47,
43,48,38,55,33,52,45,41,49,35,28,31

Key Scheduling for Encoding

1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1

Key Scheduling for Decoding

0,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1

Key Permutation

56,48,40,32,24,16,8,0,57,49,41,33,25,17,
9,1,58,50,42,34,26,18,10,2,59,51,43,35,
62,54,46,38,30,22,14,6,61,53,45,37,29,21,
13,5,60,52,44,36,28,20,12,4,27,19,11,3};

Key Expansion Permutation

31,0,1,2,3,4,
3,4,5,6,7,8,
7,8,9,10,11,12,
11,12,13,14,15,16,
15,16,17,18,19,20,
19,20,21,22,23,24,
23,24,25,26,27,28,
27,28,29,30,31,0

Initial Permutation, IP

57,49,41,33,25,17,9,1,59,51,43,35,27,19,11,3,
61,53,45,37,29,21,13,5,63,55,47,39,31,23,15,7,
56,48,40,32,24,16,8,0,58,50,42,34,26,18,10,2,
60,52,44,36,28,20,12,4,62,54,46,38,30,22,14,6

Final Permutation, IP[1]

39,7,47,15,55,23,63,31,38,6,46,14,54,22,62,30,
37,5,45,13,53,21,61,29,36,4,44,12,52,20,60,28,
35,3,43,11,51,19,59,27,34,2,42,10,50,18,58,26,
33,1,41,9,49,17,57,25,32,0,40,8,48,16,56,24

Limitations And Restrictions

Warning: THIS IS AN UNTESTED DESIGN.

Xilinx, Inc. does not make any representation or warranty regarding this design or any item based on this design. Xilinx disclaims all express and implied warranties, including but not limited to the implied fitness of this design for a particular purpose and freedom from infringement. Without limiting the generality of the foregoing, Xilinx does not make any warranty of any kind that any item developed based on this design, or any portion of it, will not infringe any copyright, patent, trade secret or other intellectual property right of any person or entity in any country. It is the responsibility of the user to seek licenses for such intellectual property right where applicable. Xilinx shall not be liable for any damages arising out of or in connection with the use of the design including liability for lost profit, business interruption, or any other damages whatsoever.