XAPP270 (v1.0) August 03, 2001

High-Speed DES and Triple DES Encryptor/Decryptor

Author: Vikram Pasham and Steve Trimberger

Summary

The future of network security depends on encryption provided in the crucial building blocks, like switches, routers, bridges, and other communication equipment. All broadband applications need high-speed cryptosystems to speed up high-bandwidth data transfers and to protect privacy. DES cryptographic hardware is used to protect civilian satellite communications, gateway servers, set-top boxes, Virtual Private Networks (VPN), video transmissions, and numerous other data transfer applications. An application of DES in VPNs is shown in Figure 1.

This application note describes the implementation of high-performance Data Encryption Standard (DES) and Triple DES encryptor/decryptor in Virtex™-II and Virtex-E devices. This implementation operates at 15 Gb/s, making it the fastest encryptor/decryptor available today, surpassing the previous record holder, a custom ASIC [Sandia]. This design fits into an XC2V1000 or an XCV400E. Three copies of the DES engine can be combined to triple the bandwidth to 45 Gb/s or to permit Triple-DES encryption at the same data rate, but with additional latency.

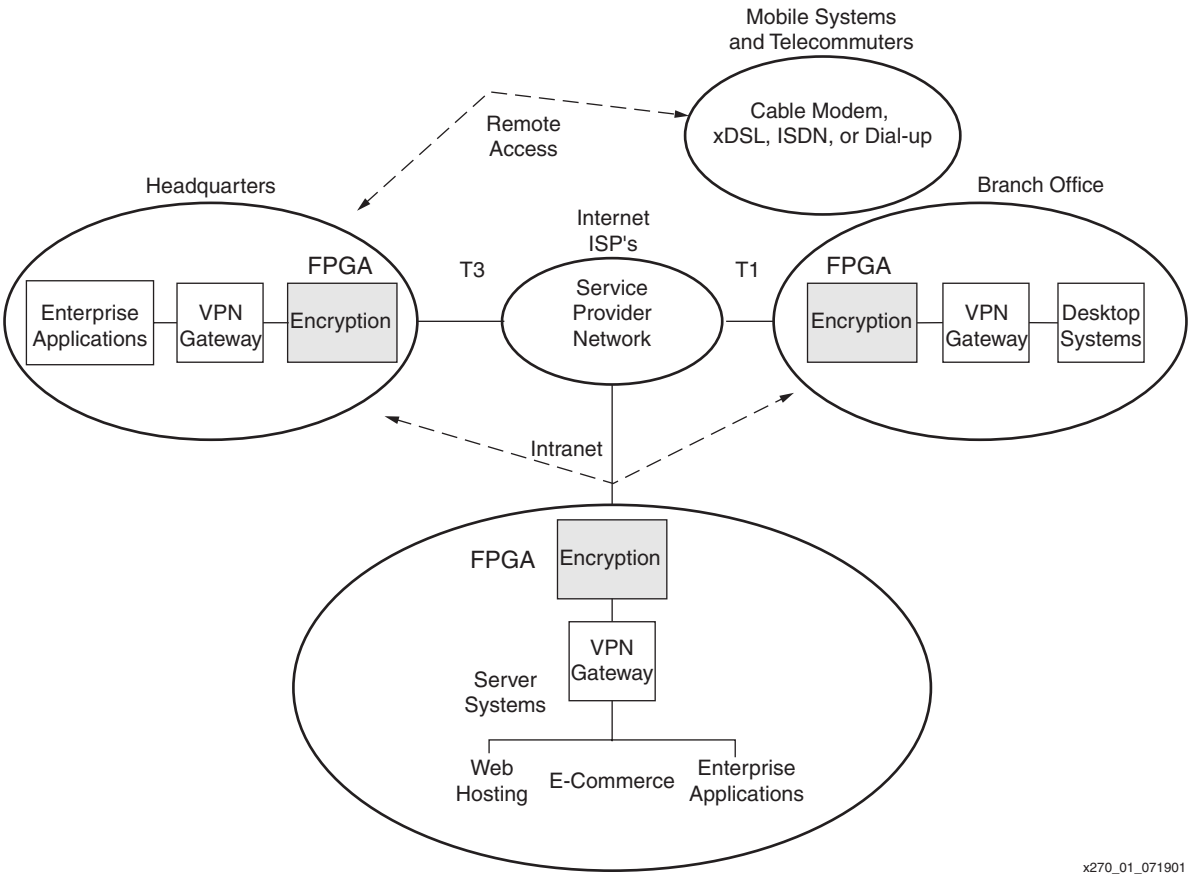

Figure 1: **Encryption in Virtual Private Networks (VPN)**

Introduction

The rapid growth of networking is driving high-bandwidth data transfers all over the world. Today, all the financial transactions, video surveillance, and e-commerce are performed online. All data transfers are carried over networks like LAN, WAN, and ATMs, which are interconnected with routers, switches, bridges, and other network equipment. The growth of virtual private networks (VPNs) and IP security solutions (IPSec) has heightened demand for secure, high performance data transfers. The biggest question is *how* secure are these transactions and will this security feature be a bottleneck in networks?

To protect privacy and safeguard sensitive data transfers from hackers and spies, U.S. government agencies recommend Data Encryption Standard (DES) and Triple DES algorithms [1], [2]. There are several encryption algorithms like DES, Blowfish, RC4/RC5, International Data Encryption Algorithm (IDEA) and others, each having strengths and weaknesses. DES is the most widely used open standard cryptosystem, offering excellent performance. Above all, it is the recommended encryption standard by U.S. government agencies, making it the *de facto* industry standard.

Several high-speed DES hardware implementations have been reported in technical journals[4],[5],[6]. Sandia National Labs announced their fastest DES custom ASIC operating at 6.7 Gb/s [3],[4]]. The implementation described in this application note surpasses all the existing DES implementations and offers up to 15 Gb/s, almost twice the performance of any other existing DES solution. The most directly comparable prior design implemented in an FPGA has complete loop unrolling and encrypts at 3.05 Gb/s [5]. Xilinx Alliance partners also offer alternate DES and Triple-DES IP cores. The IP center on the Xilinx support web gives further information at: **http://support.xilinx.com/ipcenter/index.htm**.

Concerns about DES vulnerability are driving further standardization efforts, so any encryption hardware that is deployed today might become obsolete in a few months. In contrast, an encryption engine residing in an FPGA could be updated in the field with a new encryption algorithm when it becomes available. Recently, the US Department of Commerce selected Advanced Encryption Standard (AES) as a replacement for DES. AES is not yet in widespread use. An FPGA solution is ideal in this case where compatibility with DES is required now and an upgrade to AES will be desirable in the future. Although there are concerns about the security of DES, Triple DES is still considered secure and is recommended by the US Department of Commerce.

Data Encryption Standard (DES) Algorithm

The DES block cipher algorithm was developed by researchers at IBM and was fine-tuned by government agencies, the National Security Agency (NSA) and the National Institute of Standards and Technology (NIST). The American National Standards Institute (ANSI) adopted DES as the federal standard for encryption of commercial and sensitive data. This is defined in Federal Information Processing Standards (FIPS 46, 1977) published by NIST[1], [2].

The DES algorithm has a regular structure that lends itself to pipelining and simple data manipulations to permit fast operations. DES is a symmetric encryption algorithm where the same key is used for both encryption and decryption. DES takes a 64-bit key and a 64-bit block of data as inputs, and outputs 64-bits of encrypted data. The actual key is only 56-bits and the remaining bits, i.e., the least significant bit (LSB) in every byte can be used as parity. The same basic design can be used for both encryption and decryption. A brief description of the DES algorithm in Electronic Code Book (ECB) mode is presented below. For complete details and more in-depth study, see the references listed at the end of this application note[8].

As shown in Figure 2, the DES algorithm begins with an initial permutation *(IP)*, encrypts in sixteen “rounds”, followed by the inverse of the initial permutation *(IP^{-1})*. In each round, the right-side 32 bits of the block are transformed with the function labeled “*f*” and the key, then XOR'ed with the left-side 32 bits.

Let the 64 bits of the input block to an iteration consist of a 32-bit block *L*, followed by a 32-bit block *R*. Using this notation, the input block is then *LR*. Let *K* be a block of 48 bits chosen from the 64-bit key. Then, the output *L'R'* of an iteration with input *LR* is defined by:

$$\begin{aligned} L' &= R' \\ R' &= L \oplus f(R,K) \end{aligned} \qquad (1)$$

The key for each round is a subset of the original 64-bit key with bits permuted. At each iteration, a different block *K* of key bits is chosen from the 64-bit key designated by *KEY*. Let *KS* be a function which takes an integer *n* in the range from 1 to 16 and a 64-bit block *KEY* as input. This yields 48-bit block output K_n which is a permuted selection of bits from *KEY;*

$$K_n = KS(n, KEY) \qquad (2)$$

with K_n determined by the bits in 48 distinct bit positions of KEY.

KS is called the key schedule, because the block *K* used in the *nth* iteration of equation 1 is the block K_n determined by equation 2. For the *nth* iteration, left and right blocks are defined by:

$$\begin{aligned} L_n &= R_{n-1} \\ R_n &= L_{n-1} \oplus f(R,K_n) \end{aligned} \qquad (3)$$

where *n* is in the range of 1 to 16. The pre-output block is $R_{16}L_{16}$.

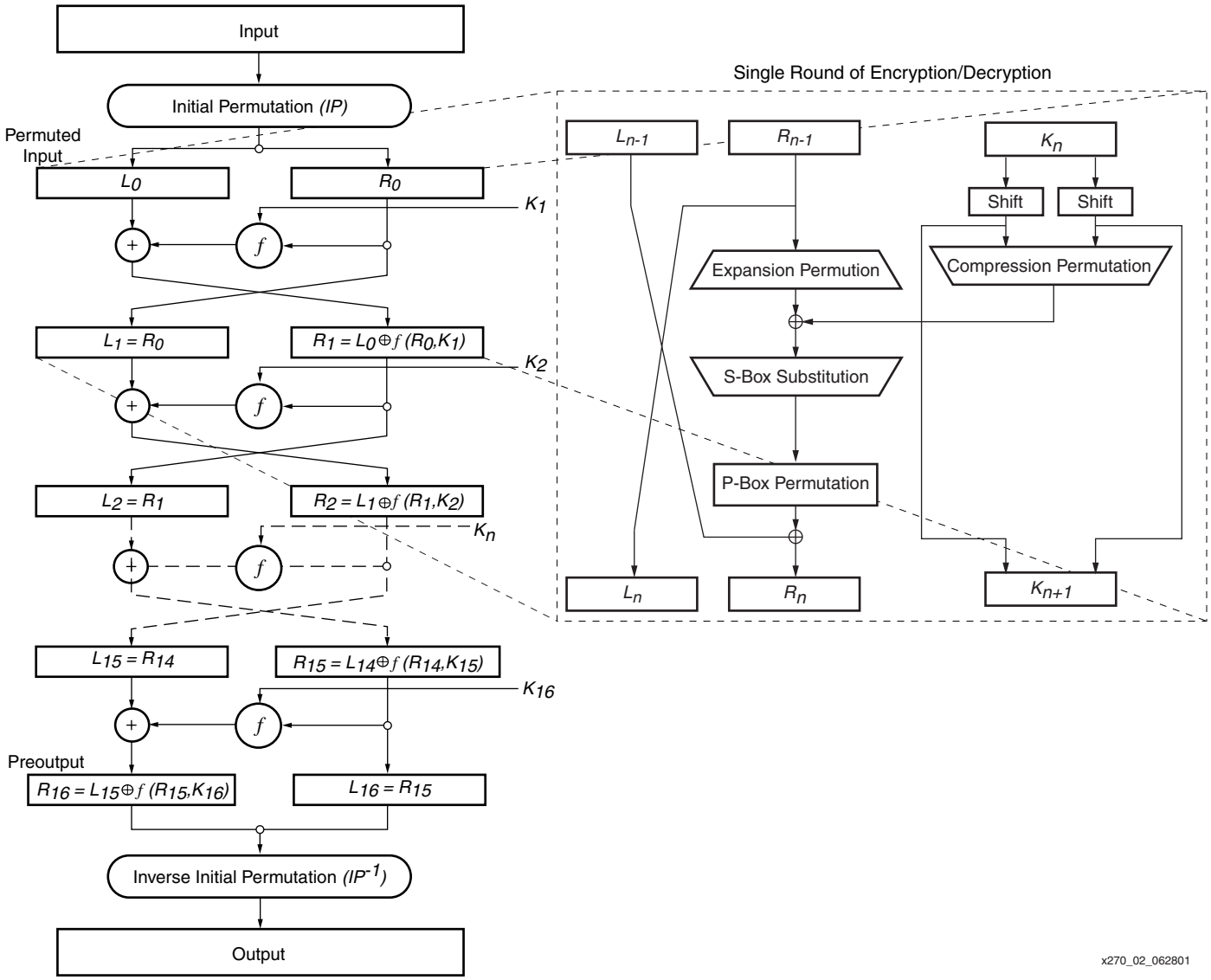

Figure 2: **DES Algorithm Overview**

The Cipher Function *f*

The *"f"* function expands the right side *(R_n)* to 48 bits, XOR's those bits with the key, and divides the resulting 48 bits into eight 6-bit fields. Those fields are used as addresses into eight 64-word by 4-bit memories, called S-boxes. The eight 4-bit *S*-box outputs are re-assembled into the 32-bit word and the bits permuted *(P)* and XORed with the left side *(L_n)* of the block. This is illustrated in Figure 3.

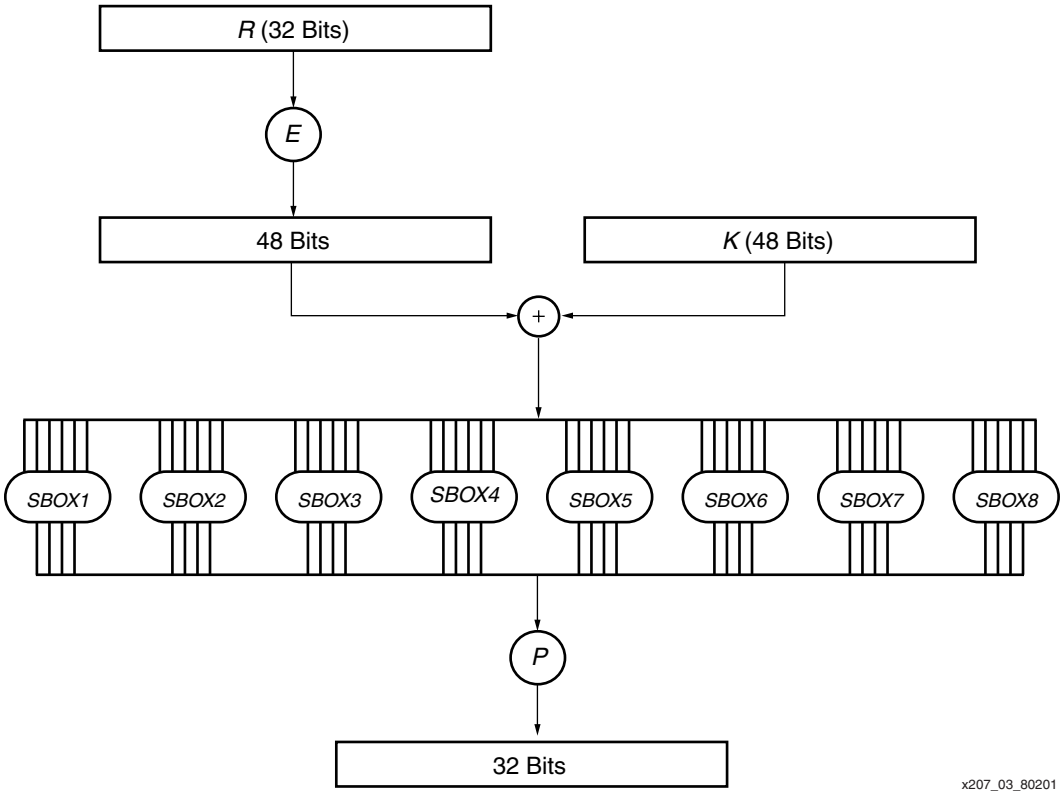

Figure 3: **Calculation of *f (R,K)***

Key Schedule (*KS*) Calculation

The *KS* calculation is shown in Figure 4. The input *KEY* goes through the initial key permutation choice, *PC–1*. In Figure 4, C_0 represents 32 MSB bits and D_0 represents 32 LSB bits of *PC–1*. The blocks C_n and D_n are obtained from the blocks C_{n-1} and D_{n-1}, respectively, for n = 1, 2,..., 16, which is accomplished by adhering to the following schedule of left/right shifts of the individual blocks shown in Figure 4. Key scheduling during encryption is performed using left shifts while decryption uses right shifts.

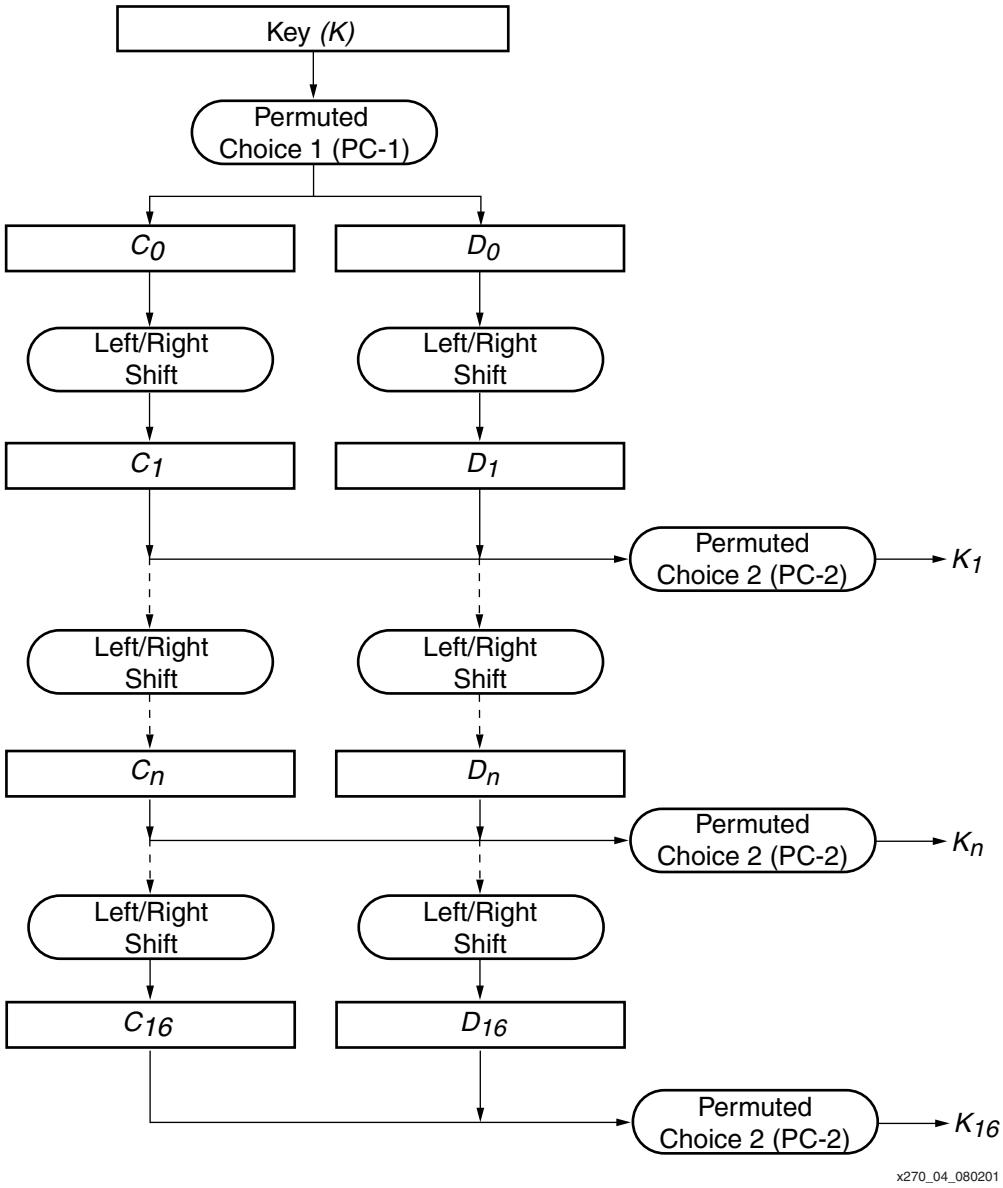

Figure 4: **Key Schedule Calculation**

Decryption

The decryption operation is the same as encryption, but the subkeys K_n are presented in the opposite order. In hardware, this amounts to changing the direction of shift and the shift amounts in subkey generation. The permutation IP^{-1} applied to the pre-output block is the inverse of the initial permutation IP applied to the input. Further, from equation 1 it follows that:

$$R = L' \qquad L = R' \oplus f(L', K) \tag{4}$$

Consequently, to decrypt the message it is only necessary to apply the very same algorithm to an encrypted message block. At each iteration of the computation the same block of key bits K is used during decryption as was used during the encryption of the block. Using the notation of the previous section, this can be expressed by the equations:

$$R_{n-1} = L_n \qquad L_{n-1} = R_n \oplus f(L_n, K_n) \tag{5}$$

Now $R_{16}L_{16}$ is the permuted input block for the decryption calculation and L_0R_0 is the pre-output block. That is, for the decryption calculation with $R_{16}L_{16}$ as the permuted input, K_{16} is used in the first iteration, K_{15} in the second, and so on, with K_1 used in the 16th iteration.

In summary, the DES algorithm consists of 16 identical encryption rounds. Each round contains a significant amount of bit movement, which is simple wire in a hardware implementation, 80 2-bit XORs, and eight lookups in 64-word by 4-bit S-boxes. Each round uses a subset of the key bits with a particular permutation. The permutation depends on the round and on whether the operation is to encrypt or decrypt. This primarily involves wiring, table lookups, and bitwise operations. FPGAs are best suited for such operations and form an ideal platform for DES implementation.

Triple DES Algorithm

The US government agencies had concerns about the vulnerability of the aging DES standard. A new standard, Triple DES, was proposed in 1993 as an alternate to DES. Let *Ek(I)* and *Dk(I)* represent the DES encryption and decryption of *"I"* using DES key *K,* respectively. Each TDEA encryption/decryption operation (as specified in ANSI X9.52) is a compound operation of DES encryption and decryption operations as shown in Figure 5. The following operations are used:

1. TDEA encryption operation is the transformation of a 64-bit block *"I"* into a 64-bit block *"O"* that is defined as follows:

$$O = E_{K3}(D_{K2}(E_{K1}(I))) \qquad (6)$$

2. TDEA decryption operation is the transformation of a 64-bit block *"I"* into a 64-bit block *"O"* that is defined as follows:

$$O = D_{K1}(E_{K2}(D_{K3}(I))) \qquad (7)$$

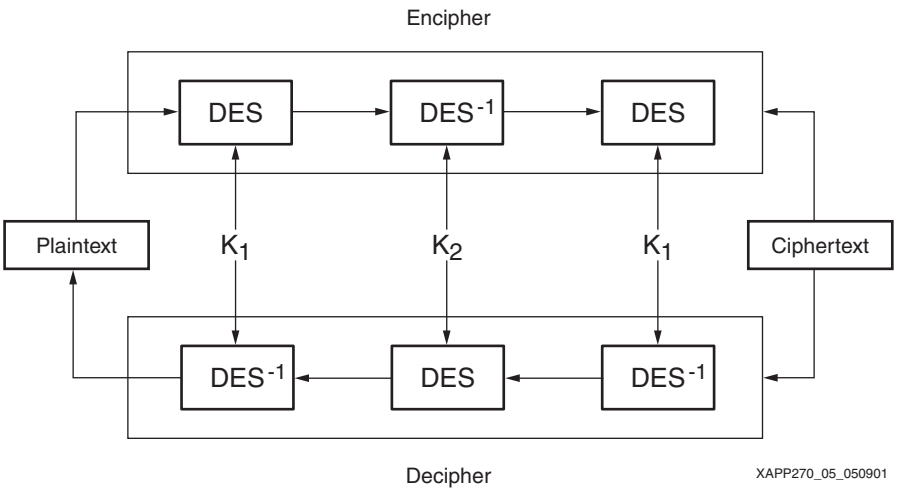

Figure 5: **Triple-DES Block Diagram**

The standard specifies the following keying options for bundle (*K1*, *K2*, *K3*):

1. Keying Option 1: *K1*, *K2,* and *K3* are independent keys.
2. Keying Option 2: *K1* and *K2* are independent keys and *K3* = *K1.*
3. Keying Option 3: *K1* = *K2* = *K3*.

Keying Option 1 provides the greatest security while keying Option 2 is less secure. Keying Option 3 is identical to single DES.

Implementation and Optimization

The main purpose of this design is to implement a high performance encryptor. The complete encryptor is implemented with 16 copies of encryption rounds with deep pipelining to maximize performance. The following sections explain the implementation and optimization steps taken to improve performance.

Loop Unrolling and Pipelining

To gain speed, 16 copies of the round were built to unroll the loop, pipelining the data through the 16 stages. This increases data rate by a factor of sixteen, but at the cost of approximately sixteen times as much logic. The critical path through the round is shown in Figure 6. A multiplexer selects key bits depending on the round and on whether it is encrypting or decrypting. The resulting selected key bits are XOR'ed with the right side of the data block (R_i), and 6-bit fields are used to address into the S-boxes. One bit of one S-box is shown as four LUTs, and selection is made using F5 MUXes. The resulting bits from the S-box are XORed with the left side bits from of the block (L_i) and stored in the pipeline register.

S-Box Mapping to LUTs

Virtex LUTs can be configured as 16 x 1 ROMs. Since this design has a number of constant S-boxes, each bit can be implemented as a 64-word lookup table. The four input lookup table, 4-LUT, can be configured as ROM by using a "case" statement in VHDL or Verilog. SBOX constants are implemented in four LUTs and directly instantiated F5 MUX in HDL to select the result from SBOX. F6 MUX is not instantiated because the F6 function could be merged with the following XOR gate in a single four LUT (circled in Figure 6). The LUT following the SBOX is required in either case to implement the XOR with the left side (Li) of the data block.

Pipelining

Each round of DES is pipelined in three stages to enhance performance. This increases the latency to 48 cycles. The data path is separated from key generation and this reduces the number of logic levels between subsequent pipeline stages. The schematic for a single round of DES is shown in Figure 6.

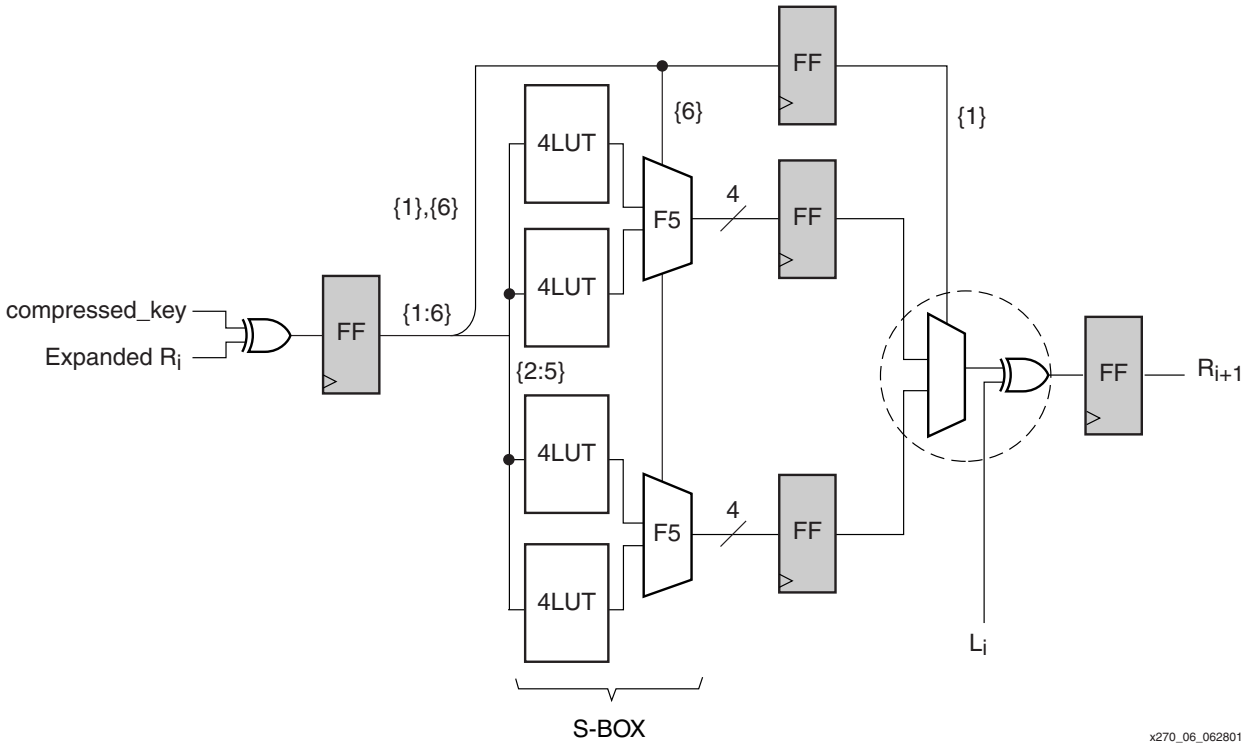

Figure 6: **Single-Round Data Path**

Key Generation

The 64-bit input key is initially permuted and goes through specific number of shift operations, followed by second round of permutation for each round. The key generation is made independent of DES round operation and has a three-stage pipeline to balance the pipelining in each round. The three stage pipelining is achieved by configuring LUTs as Shift Registers (SRL16) followed by a flip-flop. Virtex LUTs can be configured as shift registers and the user can define one to 16 shift registers for each LUT. Virtex-II devices allow cascading eight SRL16s in a single CLB (4 slices, 8 LUTs) to create a 128-bit shift register.

A SRL16 is used for the two-stage pipeline and the third stage pipeline is done in a flip-flop. This offers Place and Route tools more flexibility in placing these SRL16s and flip-flops to achieve better speed. Key generation for a single round is shown in Figure 7.

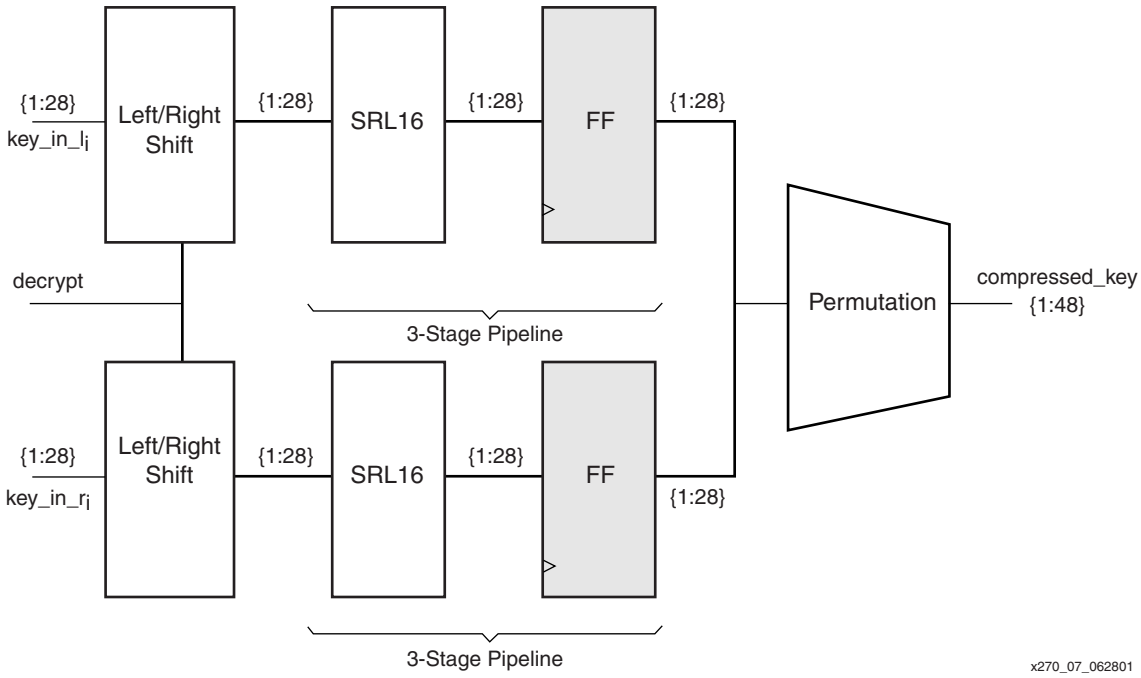

Figure 7: **Single Round Key Generation**

Triple DES Implementation

Three copies of DES are instantiated to realize Triple DES implementation, and are connected as shown Figure 5. The latency for Triple DES implementation is 144 cycles, i.e., 48 cycles for each copy of DES.

DES HDL Code and Simulation

This application note provides reference design in VHDL and Verilog. The reference design also has a sample DES test bench with NIST test vectors to verify DES encryption and decryption. This test bench verifies DES encryption for the following Key and Plain text values:

```
Key: x"00 00 00 00 00 00 00 00"
Plain text Input: x"80 00 00 00 00 00 00 00"
```

The expected encrypted result should be:

```
Encrypted output: x"95 f8 a5 e5 dd 31 d9 00"
```

This encrypted output is decrypted with the same Key to verify decryption. This testbench does *not* cover all NIST test vectors. Xilinx recommends verification of all NIST test vectors for DES compliance. For a complete test suite, refer to NIST test requirements[9].

Implementation Results

This DES design is implemented in Virtex-II devices. The resource utilization and performance results are shown in Table 1 and Table 2. This design is implemented with only a period constraint of 4 ns period and the PAR effort level of 5 is used.

Table 1: **DES Implementation Results**

Device	Number of LUTs (% of LUTs Used)	Number of Slice Flip-Flops (% of Flip-Flops Used)	Performance (maximum clock rate)
Virtex-II XC2V1000-5 FG456	5036 (49%)	5185 (50%)	15.1 Gb/s (237 MHz)

Table 2: **Triple DES Implementation Result**

Device	Number of LUTs (% of LUTs Used)	Number of Flip-Flops (% of Flip-Flops Used)	Performance (maximum clock rate)
Virtex-II XC2V3000-5 FG676	16,181 (56%)	15,759 (54%)	13.3 Gb/s (207 MHz)

Conclusion

DES and Triple-DES are deployed in communication and network equipment where data security is of utmost importance. With networks capable of transferring several gigabits per second, the encryption hardware should not become a bottleneck for the network performance. The implementation described in this application note runs almost twice as fast as a recent ASIC design. This design can run at over 15 Gb/s, making it the fastest DES encryptor.

Export and Usage Restrictions

The U.S. government has imposed certain restrictions on the export of DES encryption software and hardware. ***Xilinx recommends checking with the U. S. government's FIPS-46-3, 1999[2] publication for export restrictions and current government policies on the export of these commodities, software, and technology***. Cryptographic devices and technical data regarding them are subject to Federal Government export controls. Exports of cryptographic modules implementing this standard and technical data regarding them must comply with these Federal regulations and be licensed by the Bureau of Export Administration of the U.S. Department of Commerce.

References

For more detailed information, the following technical documents are recommended:

1. ANSI, "Triple Data Encryption Algorithm Modes of Operation", American National Standards Institute X9.52-1998, American Bankers Association, Washington DC, July 29, 1998
2. FIPS, "Data Encryption Standard", Federal Information Processing Standards Publication 46-3, October 1999 available at: **http://csrc.nist.gov/encryption/tkencryption.html**
3. Sandia National Labs press release, available at **http://www.sandia.gov/media/NewsRel/NR1999/encrypt.htm**
4. C. Wilcox, et. al. "A DES ASIC suitable for network encryption at 10Gbps and beyond." First International Workshop on Cryptographic Hardware and Embedded Systems, Springer Verlag, 1999.
5. FreeIP, **http://www.free-ip.com/DES/index.html**
6. Patterson, C., "High Performance DES Encryption in Virtex FPGAs using Jbits", FCCM 2000, IEEE Computer Society, 2000.

7. S. Trimberger, et. al. “A 12Gbps DES Encryptor/Decryptor Core in an FPGA.” Cryptographic Hardware and Embedded Systems - CHES 2000, Springer Verlag, 2000.
8. Schneier, B., *Applied Cryptography*, John Wiley and Sons, 1996.
9. NIST Test requirements at: http://csrc.nist.gov/cryptval/des.htm.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
08/03/01	1.0	Initial Xilinx release.