A Parallel Architecture for Secure FPGA Symmetric Encryption

E.J. Swankoski, R.R. Brooks
Applied Research Laboratory
Information Science & Technology Division
[ejs224,rrb5]@psu.edu

V. Narayanan, M. Kandemir, M.J. Irwin
Computer Science and Engineering Department
Pennsylvania State University
[vijay,kandemir,mji]@cse.psu.edu

Abstract

Cryptographic algorithms are at the heart of secure systems worldwide, providing encryption for millions of sensitive financial, government, and private transactions daily. Reconfigurable computing platforms like FPGAs provide a relatively low-cost, high-performance method of implementing cryptographic primitives. Several standard algorithms are used: the Data Encryption Standard (DES), its cipher block chained counterpart (3DES), and the Advanced Encryption Standard (AES). Conventional high-performance architectures utilize loop-unrolled approaches where internal hardware functions are duplicated.

We propose a parallel architecture in which internal hardware functionality is not duplicated but reused. This creates a reasonably compact single block, which is ideal for duplication. This allows multiple users to share the same hardware, as spatial isolation is achieved by the physical separation of individual encryption blocks. Also, this allows for a greater degree of scalability, and system throughput becomes limited only by available physical resources and available I/O resources. We conclude that this parallel encryption architecture allows for comparable performance compared to conventional pipelined architectures with greater flexibility and hardware efficiency.

We show that a pipelined encryption system cannot be used in a physically secure environment as it does not protect the keys adequately. Temporal isolation of the key is achieved using the parallel architecture. Indirect key storage is accomplished using principles of controlled physical random functions, which make all key values fully transient and never hardware-resident. Thus the parallel architecture achieves a high level of physical and design security within the FPGA, protecting the key from both invasive and non-invasive physical attacks.

1. Introduction

The recent evolution of powerful FPGA hardware has made their suitability for cryptoprocessor systems more evident. Additionally, most cryptographic algorithms have ease of hardware design as a main design goal, which makes them particularly well suited to implementation in Verilog HDL. The high cost of cell-based and full custom cryptography chips makes them prohibitive. Also, the inefficiency and low throughput of software implementations prevents their widespread use. FPGAs present an ideal compromise in that they retain the reconfigurability and control of software approaches while also achieving high throughputs near those of custom-designed ASICs.

All high-throughput cryptographic block cipher implementations have utilized a pipelined approach, where inner-round functions (such as those in AES or DES) are duplicated. This allows for both high throughput and efficient use of hardware. However, key control logic becomes complex should one desire to change keys during encryption, as either the pipeline must be emptied and the key changed or additional logic is required to detect and adapt to the change. We propose a high-throughput parallel processing an alternative to pipelined cryptoprocessor architectures. In addition to the pipelining benefits of hardware efficiency and high throughput, it allows for scalability and controllability of the resulting architecture. Pipelined FPGA cores do not utilize the entire chip; our parallel architecture allows for maximum utilization given sufficient I/O resources.

Conventional pipelined implementations of the AES standard can achieve data rates up to about 17.5 Gbps [1]. Pipelined implementations of DES can achieve data rates of up to about 10 Gbps, depending on both the target architecture and the design entry method [1,2]. By comparison, a high-speed software implementation of DES would likely achieve a throughput of about 250 Mbps. Our parallel architectures have indicated memoryless throughputs of 18.8 Gbps for AES, 9.00 Gbps for DES and 8.631 Gbps for 3DES. Using the Virtex-II Pro's Block RAM resources for the AES substitution boxes, we achieve a throughput of 17.7 Gbps. This paper aims to evaluate the performance and implementation details of parallel processing architectures based on the AES and DES symmetric key block ciphers. Verilog HDL modules are synthesized on the Virtex-II Pro FPGA platform to evaluate performance and security of parallel cryptoprocessing applications.

2. Related Work

Much work has been done in the area of high-performance FPGA implementation of block ciphers. Most work has been done on DES and AES; work on AES has become more prominent recently since the adoption of the Rijndael algorithm as the standard. Also, work on pipelined DES has faded with its security becoming weaker with the advent of greater computing power. This is not to say that DES is no longer relevant – it is a fast and economical solution for low-security encryption applications. Triple DES is inherently sequential, and as such is a logical extension of DES. Hence much of the work related to DES and pipelined DES applies both directly and indirectly to Triple DES.

Advances in computing power and FPGA architecture have been reflected in the most recent encryption designs, with many implementations achieving throughputs of greater than 10 Gbps. The Xilinx implementation using Jbits achieves significant throughput advances by removing the key schedule and mode select hardware from the datapath [2]. Thus, any desired key change or mode switch requires a reconfiguration of the chip. Though not a pure reconfigurable DES core, this compromise allows its throughput to exceed the record-holding Sandia National Laboratories DES ASIC chip [1]. Most AES designs are conventional pipelines similar to pipelined DES. Throughput is dependent on the target device, as a pipelined Virtex-E device achieves a throughput of 4.12 Gbps [3] whereas a pipelined Virtex device can achieve a throughput of 12.2 Gbps [4]. The most recent high-performance AES implementation uses a Virtex-II device, and achieves a throughput of 17.8 Gbps [1]. Other factors affect performance as well, such as design entry method (Verilog or VHDL), design optimizations, and implementation optimizations. A multithreaded approach proposed by Alam and Badawy uses multiple pipelined units to achieve a throughput of 7.68 Gbps on a Virtex-II device; this approach is inefficient because it requires a very large device to achieve maximum performance [5]. Also, this leads to underutilization of I/O resources as well as fragmentation of the FPGA's logic cells.

3. Overview of Parallel Architecture

We propose the parallel architecture as a method of achieving maximum utilization of the FPGA's logic cells and I/O resources. Symmetric key block ciphers consume a very large amount of silicon area with respect to their I/O usage; for example, a memoryless 128-bit AES encryptor proposed in [1] uses roughly 12 times more area than an arbitrary 64-bit multiplier despite having the same amount of I/O usage. Typically, larger FPGAs are required for implementation of these block ciphers, and larger FPGAs have by definition higher numbers of I/O pins.

As we will show, pipelined architectures require considerably more area than a single parallel encryption block; however, a fully parallel encryption architecture requires more area than a pipelined architecture. Parallel blocks allow a far greater degree of flexibility when designing an encryption system as we will detail based on its area flexibilities and security advantages. If two pipelined architectures cannot fit within a given device, an arbitrary amount of FPGA resources, both logic and I/O, will remain unused as the pipelined block cannot be split. However, individual parallel blocks are considerably smaller and can be used to reduce fragmentation and increase utilization of the FPGA's logic and I/O resources.

We will also show that parallel architectures provide both performance and utilization benefits in area-constrained devices. If the available area is an integer multiple of the area required for a pipelined architecture, pipelined systems have a performance advantage. However, for spaces larger or smaller than this, pipelined systems become inefficient. Note we make the assumption that additional I/O resources are always available. However, in a black-and-white area comparison, pipelined architectures are considerably smaller than fully parallel architectures. This is unavoidable, as the key hardware must be duplicated for each block. In the case of AES, the key scheduling module is rather large; this represents a direct tradeoff between area and security of the system.

Table 1: Summary of Related FPGA Encryption Implementations

Design Origin	Implementation	Target Architecture	Throughput
Belfast, DSiP Labs	Pipelined DES	Virtex	3.87 Gbps
Xilinx	Pipelined DES	Virtex	10.7 Gbps
Sandia National Labs	Pipelined DES	ASIC	9.28 Gbps
Tampere University, DCS Lab	Pipelined 3DES	Virtex	364 Mbps
Rodriguez, Saqib, Diaz	Pipelined AES	Virtex-E	4.12 Gbps
GMU	Pipelined AES	Virtex	12.2 Gbps
Helsinki UT	Pipelined AES	Virtex-II	17.8 Gbps
University of Calgary	Multithreaded AES	Virtex-II	7.60 Gbps

It is important to both the security and functionality of the system that the keys are kept separate. This requires that each individual parallel block have its own key hardware, which enforces spatial isolation of the keys. This allows multiple independent encryptions to process simultaneously. As a consequence, the parallel encryption blocks suffer an area penalty with respect to the pipelined architecture. It should be noted that is infeasible to use shared key hardware among the parallel encryption blocks since each block is limited by design to only one output per cycle. Key sharing is impossible because even if two encryption blocks share a common key, no two parallel blocks are the same point in the encryption and hence would require separate key values.

Figure 1 below details the differences between our proposed parallel architecture and conventional pipelined architectures. Each block in the parallel architecture is a completely self-contained encryption unit. The dotted lines indicate the smallest possible unit that can encrypt a block of data. A fully parallel encryption architecture utilizes *n* blocks, where *n* is the number of rounds of the specified block cipher. Note that a pipelined implementation requires all *n* functional blocks whereas a parallel block requires only one. Thus we define a parallel encryption block as a single round function block and a key control module. Furthermore, we define a pipelined encryption as having one key control module and *n* round function blocks. In the parallel case, more than *n* blocks requires an additional I/O allocation. The parallel encryption blocks each have their own independent key hardware. This illustrates the property of *n* independent encryption sessions utilizing *n* independent keys. Also, it follows logically that only one independent encryption block must be present for encryption to proceed. We can see then that the use of *n* independent keys requires a minimum of *n* parallel blocks.

Figure 2 below illustrates the performance comparison of the fastest and most efficient published implementation of AES-128 with our fully parallel architecture. Note that this fully parallel architecture uses Block RAMs to implement the byte substitution boxes. We see that as expected, the pipelined implementation has better best-case performance but is limited greatly in terms of usability. It is clear that when area is constrained, parallel architectures provide improved performance and provide more efficient utilization of the FPGA's resources. The scalability of the parallel architecture makes it suitable for smaller spaces. The pipelined architecture requires approximately 11000 SLICEs and the parallel blocks each require approximately 1300 SLICEs. In this case, where the available area is a multiple of approximately 11000, the pipelined architecture has an advantage. However, for ranges larger and smaller than integer multiples of 11000, the parallel architecture provides greater logic utilization as well as increased overall system performance.

Figure 1: Overview of Parallel and Pipelined Architectures

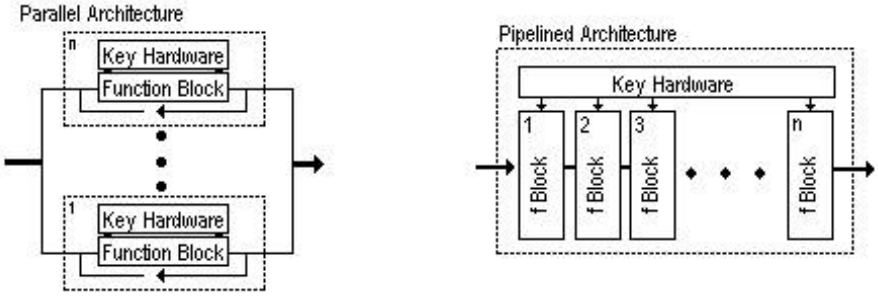
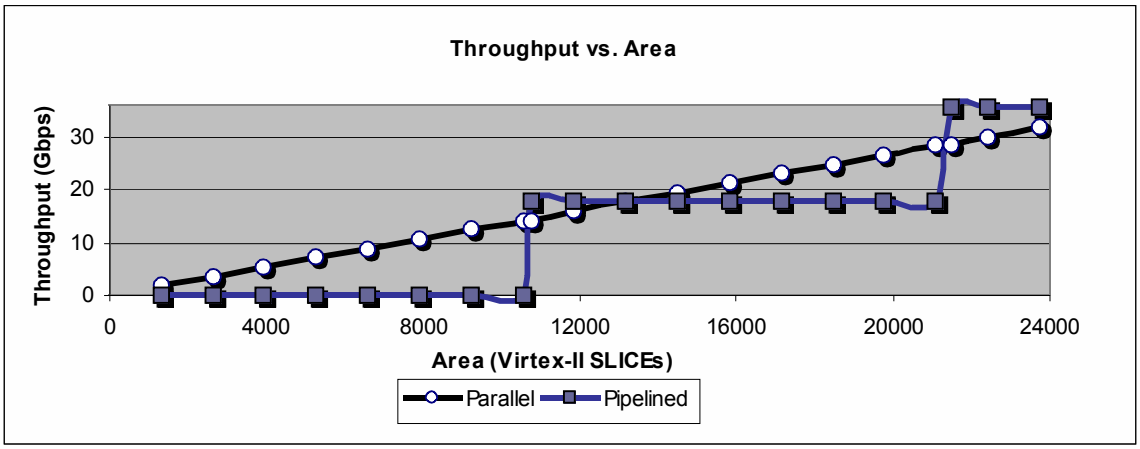


Figure 2: Area / Throughput Comparison of Parallel and Pipelined Architectures

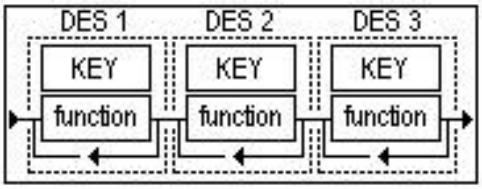


4. Design of Parallel DES & 3DES Block Ciphers

The Data Encryption Standard (DES) was adopted as U.S. Government standard in July of 1977. It uses a 56-bit key and operates on 64-bit blocks of data. Encryption and decryption are symmetric in that they use the same functions with a reversed key schedule. The basic operations of DES include permutations, compressions, expansions, and shifts using 32-bit operands. The data blocks in DES are split in half. There are 16 rounds in DES; a round consists of 3 32-bit XOR operations, three 32-bit permutations, and 8 4-bit substitutions. Following an initial permutation, 15 identical rounds are performed; the last round is slightly different. An ending permutation finishes the computation [6,7,8]. The design allows for relatively easy pipelining, as identical hardware could be duplicated 15 times in succession. The design is not pipelined for several reasons, the most notable being the ease of expansion to Triple DES.

Though the initial and final permutations are not essential for the security of DES, they are implemented. The round function is implemented once and once only – the same hardware is used 16 times per encryption or decryption operation. This reduces area by approximately a factor of 16 opposed to a larger, pipelined version, and it has the added effect of greatly reducing the size and complexity of the key management control logic. The round function itself is self-contained. It is designed combinatorially to allow for ease of lookup tables, which optimizes both speed and area.

Figure 3: Expansion of DES to 3DES



The Triple DES block increases the security of DES greatly by expanding the key length from 56 to either 112 bits (2 key Triple DES) or 168 bits (3 key Triple DES). The Triple DES algorithm consists solely of three single DES operations in sequence: encryption with key 1, decryption with key 2, and encryption with key 3 to produce the cipher text. Figure 3 above shows a pipelined-block design used for Triple DES – the single DES hardware is duplicated three times in succession. There is minimal control logic involved – the valid output of the first encryption block should trigger the operation of the second decryption block, and the valid output of the second decryption block should trigger the operation of the final encryption block. Also, as a pipelined design, the valid output of the first block indicates it can accept a new block of data to encrypt. Despite the fact that single-block latency is tripled, this allows for throughput levels identical to single DES provided the keys do not change.

A single DES block (or Triple DES block) can be replicated and scaled to create a controllable high-throughput parallel cryptoprocessor. Conventional pipelined architectures replicate the internal round structure; typical DES pipelines include hardware for 16 rounds. The downside of this approach is the complexity of key hardware, as more hardware is required to store and select individual round keys. Additionally, sequential encryptions are required to use the same key. In a multi-user environment, this may not be desirable. A higher security approach would allow each sequential encryption the option of using any one of the keys (or key sets, in the case of Triple DES).

A parallel architecture is proposed which, at a small penalty to area, provides a high-throughput, high security cryptoprocessing environment. A parallel architecture as part of a network on a chip provides a high level of controllability, as each data block in the encryption queue can theoretically select its key or key set. This architecture includes 17 separate parallel DES or Triple DES blocks. Based on available space, more or less than 17 blocks can be implemented; 17 was chosen as it represents a zero-latency design. Additional I/O resources would allow other configurations, such as 34, 51, and so on; only designs with multiples of 17 have zero latency. It is feasible to use an encrypted header to indicate the desired key or key set; also, in the case of DES or Triple DES, this header can select between encryption and decryption. An initial key setup period is required; this time period ranges from a minimum of one cycle to a maximum of n cycles, where n is the number of parallel blocks implemented. Note that key setup time is still one cycle provided all parallel blocks use the same key. It is feasible to use an encrypted header to indicate the desired key or key set; also, in the case of DES or Triple DES, this header can select between encryption and decryption. The security of the header need not be equivalent to that of the data itself; invalid or tampered headers would not compromise the data.

5. Performance of Parallel DES & 3DES Block Ciphers

The single DES encryption block is a fast, compact design. It was synthesized on the Virtex-II Pro device to provide the highest performance. The Virtex-II Pro is also very efficient in terms of area, as its logic cell design differs substantially from earlier Virtex and Spartan FPGA architectures. Below are the results of synthesis and translation given 1) an iterative single DES block and 2) a pipelined Triple DES block consisting of three single DES blocks.

Table 2: Performance Details of DES and 3DES Implementations on Virtex-II Pro FPGAs

Algorithm	Number of Parallel DES Blocks	System Frequency	Key Units	Area (Slices)	Throughput
3DES	3	195.198 MHz	3	819	734.9 Mbps
DES	1	203.376 MHz	1	343	765.7 Mbps
3DES	51	134.862 MHz	51	14525	8.631 Gbps
DES	17	140.627 MHz	17	5444	9.000 Gbps

The relatively small size of the Triple DES implementation provides many interesting possibilities on many FPGA architectures. Given that the Triple DES implementation takes up 3.47% of the Virtex-II device and each single DES encryption or decryption operation has a latency of 17 cycles, each Triple DES Block could be duplicated 17 times to create a high-throughput zero latency cryptoprocessor. This has the additional security benefit of allowing 51 separate keys at the added expense of key setup time.

A high-level parallel design incurs some hardware overhead, including the data input and output multiplexers and tri-state buffers. The synthesized clock frequency is greatly decreased to 134.862 MHz, resulting in a throughput of 8.631 Gbps. After an initial key setup period (where separate keys are loaded if required), there is a constant zero-latency effect on the cryptoprocessor. Similarly, a DES cryptoprocessor has a slightly higher throughput at 9.000 Gbps with just over one third the area. The performance of these implementations is second only to the Xilinx JBits implementation; we have the advantage of allowing dynamic key flexibility. A complex system of this magnitude would likely not be ideal for a system on a chip environment, but as a single-chip cryptoprocessor it would be an excellent high-performance option. However, high-capacity FPGAs and ASICs make the use of a parallel processor a real option for a system on a chip. The option to use either DES or Triple DES allows a tradeoff between area and security with little effect on overall throughput. The scaleable throughput T_s can be quantified as follows:

Let N be the number of parallel blocks implemented and F_{sys} be system clock frequency. We then have

(Eq. 1 – DES & 3DES Throughput)

$$T_s = ((64)*F_{sys})*(N/17)$$

Note that the number of available I/O pins limits the effective throughput, as each multiple of 17 requires an additional 128 pins. Assuming a fixed number of pins for control and system operation and an 8-bit data header, each block of data requires 144 bits for input and output. This allows up to 17 blocks at no penalty to throughput; implementing more than 17 blocks with 144 I/O pins results in no gain. It is logical that required I/O can be quantified as follows:

Let P_S be the number of system pins required for proper operation. We then have

(Eq. 2 – Required I/O Pins)

$$P_{IO} = (144)*\lceil (N/17) \rceil + P_S$$

It follows that maximum throughput is attained when the number of parallel blocks implemented is an integer multiple of 17. Note that for Triple DES the single-block latency is three times that of single DES; however, the system latency is still zero. Given that the system latency of a single DES parallel processor is zero, the latency of a Triple DES processor arranged sequentially must also be zero.

6. Design of Parallel AES Block Cipher

The Advanced Encryption Standard (AES) was officially adopted in May of 2002 as the new encryption standard. It is designed to operate on all combinations of data input and keys with lengths of 128, 192, and 256 bits. This design uses a data input length of 128 bits with a key length of 128 bits. A block of data is placed into a 16-byte array, and proceeds through 10 rounds of encryption. Basic operations include byte substitutions, independent row byte shifts, column Galois field multiplications, and key additions. Row shifting and column multiplication use 32-bit operands (one 4 byte row or one 4 byte column) [9,10]. Similar to DES, this design is not pipelined; it is able to achieve reasonable throughput without doing so. Also, space limitations within the context of a mid-size FPGA make pipelining prohibitive. It should be noted that AES is not symmetric for encryption and decryption. The mathematical operations are different and require different hardware [9].

The design uses reusable function hardware, with minimal unnecessary hardware duplication. As was the case with DES, the structure of AES lends itself logically to reusable function block. The four row shifting operations are separate modules, since each operation is a separate shift. All row shifting is done through routing channels; no logic resources are used. They are similar to DES permutations, though entire bytes are shifted rather than individual bits. The column multiplication operations use four separate modules, allowing each column multiplication to proceed in parallel. The byte substitution is a 256x8 ROM lookup, and it is duplicated 16 times to

also allow maximum parallelism. Also, we can use Virtex Block RAMs to implement the byte substitution tables. This saves a considerable amount of space, since a fully combinatorial implementation of a single encryption block requires 1,280 Virtex-II SLICEs for substitution tables alone. A dual-ported Block RAM is used to implement two substitution boxes. This requires 8 Block RAMs per block; a fully parallel block would use 80 Block RAMs. The substitution boxes associated with the key scheduler are implemented combinatorially for performance reasons. All other internal functions are combinatorial. This allows for a parallel architecture, which to some degree sacrifices hardware efficiency for throughput.

The security of AES has been well researched and is widely considered to be more secure than Triple DES. Its longer key length adds to its security capabilities; additionally, key lengths of up to 256 bits allow an even higher level of security. In this design, the 10-round structure and 128-bit block size allow the AES algorithm to encrypt data much faster than a similar Triple DES implementation. Also, AES provides more efficient use of hardware; its performance and security capabilities far offset its somewhat larger area.

7. Performance of Parallel AES Block Cipher

The AES implementation is ultimately not as compact as possible, but duplicates some hardware to achieve higher performance. The byte substitution ROM is implemented 16 times; these modules are re-used each round during encryption and decryption. An extremely area-constrained design could theoretically use only one byte substitution ROM with a huge penalty to throughput. Also, the column multiplication function is repeated four times. This is not as much of an issue, since a single multiplication operation requires roughly half the area of a single byte substitution ROM. The largest component is the key scheduler, which is not duplicated. The bulk of the key scheduler is comprised of four byte substitution ROMs and four 32-bit XORs.

The encryption module is able to attain gigabit throughput, but as a comprehensive module the system must operate at or around the decryption frequency (depending on synthesis results). The overall throughput is reasonable at about 1.6 Gbps. Note that the decryption operation initially incurs a 10 cycle key setup penalty once per key lifetime. The keys are generated sequentially but must be used in reverse order. It should also be noted that encryption and decryption could occur in parallel with a comprehensive module provided the inputs arrive on subsequent cycles. Also, it is assumed that decryption key setup occurs prior to the bulk of the encryption or decryption operations. For this paper, we consider only encryption performance.

Since the design does not waste hardware, we can also construct a dedicated high-throughput AES encryption processor based on duplicated single AES encryption modules. Unconventional approaches have been proposed before, including multithreaded and pipelined approaches. This architecture is similar to a multithreaded approach in that synchronization between "threads" need only occur to prevent data collision at the output. Assuming all modules are identical, collisions are impossible, as a collision would require data arriving simultaneously to two separate units. This is prevented because the input bus is shared.

For a fully parallel encryption architecture, we do not include any decryption modules. Additionally, the inclusion of decryption would reduce the maximum hardware utilization to 50%. At best, it would interleave encryption and decryption operations, likely increasing the overall latency for both operations.

Note that we include synthesis results for both a single AES block and an AES encryption processor with a hard-coded key. This is similar to the JBits implementation of DES mentioned above, and the speed gains are noticeable. Also, a significant area reduction is achieved. However, this approach is impractical for two reasons. The key security is weakened greatly, as all round keys (including the key in its pure form) are stored in either on-chip RAM or ROM. Thus direct memory attacks could intercept the key itself. Also, changing the key requires a partial reconfiguration of the device. This expends a considerable amount of power.

Table 3: Performance of AES Encryption and Decryption on Virtex-II Pro FPGAs

Algorithm	Number of Parallel Blocks	System Frequency	Area (Slices)	Key Units	Block RAMs	Throughput
AES-128	10	146.798 MHz	23979	10	0	18.80 Gbps
AES-128	10	138.122 MHz	14013	10	160	17.77 Gbps
AES-128	2 (1 Encryptor / 1 Decryptor)	124.906 MHz	6184	2	0	1.599 Gbps
AES-128	1	147.973 MHz	2921	1	0	1.894 Gbps
AES-128	1	145.052 MHz	1319	1	16	1.857 Gbps
AES-128	10	150.621 MHz	20249	0	0	19.28 Gbps
AES-128	1	161.577 MHz	2370	0	0	2.068 Gbps

As before, we can quantify the throughput of the AES encryption processor as follows. Let N = number of parallel blocks implemented and F_{sys} = system clock frequency.

(Eq. 3 – AES Encryption Throughput)

$$T_{AES} = ((128)*F_{sys})*(N/10)$$

Also, we quantify the throughput of non-parallel AES decryption over time as follows:

(Eq. 4 – AES Decryption Throughput)

$$T_{DEC} = ((128)*F_{sys})/(10+(10/O_D))$$

where O_D indicates the number of decryption operations performed with the same key. Throughput of the decryption operation approaches that of the encryption operation provided the key does not change; the initial 10-cycle latency becomes less significant over time.

Table 3 above presents a summary of our AES implementations. We noted initially that the pipelined memoryless AES implementation proposed by Jarvinen in [1] is the fastest published implementation of the AES-128 algorithm with a reported throughput of 17.8 Gbps. Our proposed memoryless parallel encryptor achieves a throughput of 18.8 Gbps and the Block RAM parallel encryptor achieves a throughput of 17.8 Gbps. Our parallel architectures are larger; however, they are able to utilize 10 different keys. Also, as we emphasized before, individual parallel blocks can be placed as area permits where a pipelined architecture cannot fit.

Place and route results indicate a memoryless parallel AES encryption processor with all its included control logic can fit in a Virtex-II Pro 50 (XC2VP50). It is large, occupying nearly all (90-97%) of the available slices, but is able to achieve extremely high throughput. This is impractical for a system on a chip. It is possible that a system could utilize the embedded PowerPC and Block RAM resources without requiring logic, but this is unlikely. The implementation that uses Block RAM resources to implement substitution boxes uses roughly half of the device's logic resources and roughly one third of the device's Block RAM resources. Thus this architecture would be suitable for a system on a chip. Larger Virtex-II Pro FPGAs would be able to integrate a memoryless high-throughput AES processor as part of a system on a chip; this is beyond the scope of this paper.

8. Security

Consider the key scheduling methods used in symmetric key algorithms. The structure of AES has 10 rounds; DES and Triple DES have 17 rounds. Each round has an associated key. Initially, DES and Triple DES use a key derived from the original plaintext key via an algorithmically specified permutation. Similarly, AES uses the plaintext key for the first round. Both algorithms use modified keys in each successive round.

The pipelined architectures duplicate the internal functions of the block ciphers. This effectively prevents temporal isolation of the key, as each round has a constant key output. Essentially, in the structure of a pipelined AES implementation, the key is always present in the chip in its pure form, making it vulnerable to attack and interception through differential power analysis. The same holds for DES and Triple DES. It has been shown that pipelined symmetric encryption methods are inherently insecure, as they are vulnerable to differential power analysis attacks [11].

A controlled physical random function allows each chip to individually generate a device-specific signature. These algorithms are based primarily on delay analysis of self-oscillating loops to generate distinct hardware signatures that are deterministic within the context of a single device. Research has been done in the implementation of controlled physical random functions in FPGAs, and results indicate the signatures can be reliably used to distinguish between separate devices [12,13]. However, these circuits have not yet been perfected. We can use principles of these physical random functions to create an obfuscated key.

To achieve temporal isolation of the key using these signatures, we assume that the value of the physical function is known. Also, we assume that is easily accessible by the device's control functions. We can compute an XOR of the device's signature with the symmetric key; this value can be stored with relative insecurity as it is useless outside the given device. Since this is done at compile time, we do not need to input the key to the device after configuration. Also, the physical signature is not stored explicitly, and it must be computed dynamically when needed. Any physical attempt to probe the value will alter the electromagnetic characteristics of the chip and hence affect the delay, obfuscating the signature. Thus, we do not store the key explicitly; it is virtually impossible to intercept the key through any physical methods.

Within the context of symmetric-key block ciphers, we can further detail the security effects of physical random functions. DES and Triple DES perform first a permutation of the key, followed by shifts and finally the XOR application the encryption data. Since shifts and permutations are bit-independent and do not alter any values, we can perform the key operations on the XOR value alone. We can replace the 2-input XOR of the DES round operation with a 3-input XOR of the computed key value, the encryption data, and the physical signature. Thus, the actual key values only exist as transient values and results. We incur an additional delay of a 3-operand XOR instead of a 2-operand XOR; also, additional hardware is required to shift and permute the physical signature.

Similarly, AES encryption can avoid the existence of the key at any given time. However, because AES key values depend on the previous round's interim key value, storage is required. It is possible to store only an XOR value; however, this adds two additional levels of logic as XOR operations are required both prior to storage of the key value and prior to computation of the new key value. Since bit values are changed, there is no inherent insecurity of storing interim key values. Also, since the first operation involving the key uses the plaintext key, we can apply a 3-input XOR to remove the existence of the key from the device.

9. Conclusion

We have shown that a parallel architecture for symmetric cipher encryption allows a higher degree of control over conventional pipelined architectures. Also, the parallel encryption architectures allow for multi-gigabit throughput for all symmetric ciphers. Single-chip performance of this parallel approach exceeds most commercially available pipelined cores. The proposed architecture uses parallel encryption blocks to achieve a high throughput zero latency design.

The implementation of the algorithms and encryption processors in Verilog HDL allow for efficient implementation in both FPGA and ASIC mediums. Also, unlike full-custom designs, optimizations and changes can be made quickly and easily. This allows for a high degree of scalability and controllability of the parallel architecture. Additionally, through slight design modifications we can show that the use of Block RAM for substitution boxes improves relative performance.

We have shown also that a parallel architecture provides a greater degree of security than conventional pipelined architectures. We can use controlled physical random functions to generate a device-independent hardware signature. With some slight algorithmic modifications, we can limit the existence of the key to partial transient values, and hence we protect the symmetric key from analysis and interception.

10. References

1. K. Jarvinen, M. Tommiska, and J. Skytta, "A Fully Pipelined Memoryless 17.8 Gbps AES-128 Encryptor," Proceedings of the 2003 ACM/SIGDA Eleventh International Symposium on Field Programmable Gate Arrays, Pages 207-215.
2. C. Patterson, "High Performance DES Encryption in Virtex FPGAs using Jbits," IEEE Symposium on Field-Programmable Custom Computing Machines, 2000, Pages 113-121.
3. F. Rodriguez-Henriquez, N.A. Saqib, and A. Diaz-Perez, "A 4.2 Gbit/s Single-Chip FPGA Implementation of AES Algorithm," Electronics Letters, July 2003, Pages 1115-1116.
4. P. Chodowiec, P. Khuon, and K. Gaj, "Fast Implementations of Secret-Key Block Ciphers Using Mixed Inner- and Outer-Round Pipelining," Proceedings of the 2001 ACM/SIGDA Ninth International Symposium on Field Programmable Gate Arrays, Pages 94-102.
5. M. Alam, W. Badawy, and G. Jullienn, "A Novel Pipelined Threads Architecture for AES Encryption Algorithm," Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors, 2002.
6. J.V. Oldfield and R.C. Dorf, Field Programmable Gate Arrays: Reconfigurable Logic for Rapid Prototyping and Implementation of Digital Systems. John Wiley & Sons, 1995.
7. B. Schneier, Applied Cryptography: Second Edition. John Wiley & Sons, 1996.
8. "Data Encryption Standard," Federal Information Processing Standards Publication 46-2, December 30, 1993.
9. J. Daemen and V. Rijmen, "AES Proposal: Rijndael," AES Algorithm Submission, September 3, 1999.
10. "Advanced Encryption Standard," Federal Information Processing Standards Publication 197, November 26, 2001.
11. H. Saputra, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, R. Brooks, S. Kim and W. Zhang, "Masking the Energy Behavior of DES Encryption", Design Automation and Test in Europe 2003, Munich, Germany.
12. P. Hamalainen, M. Hannikainen, T. Hamalainen, and J. Saarinen, "Configurable Hardware Implementation of Triple DES Encryption Algorithm for Wireless Local Area Network," 2001 IEEE International Conference on Acoustics, Speech and Signal Processing, Pages 1221-1224.
13. B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon Physical Random Functions", Proceedings of the Computer and Communication Security Conference, November 2002.