A HIGH THROUGHPUT FPGA CAMELLIA IMPLEMENTATION

Daniel Denning, James Irvine, Malachy Devlin

Institute of System Level Integration, Alba Centre, Alba Campus,
Livingston, EH54 7EG, UK
E-mail: daniel.denning@sli-instiute.ac.uk

ABSTRACT

In this paper we present a Field Programmable Gate Array (FPGA) implementation of the Camellia encryption algorithm. Our implementation deeply sub-pipelines the algorithm for the FPGA architecture. Camellia has been included in both portfolios of the New European Schemes for Signatures, Integrity, and Encryption (NESSIE) for Europe and the Cryptography Research and Evaluation Committee (CRYPTREC) in Japan. The implementation is the fastest published throughput for the entire block ciphers recommended in both portfolios for NESSIE and CRYPTREC, and runs at a throughput of 33.25Gbit/sec.

1. INTRODUCTION

The Camellia [1] symmetric-key block cipher has been recognised by an open call from NESSIE [2] and CRYPTREC [3] as a cryptographic algorithm to help protect the current and future information society. FPGAs provide a very beneficial platform for implementing cryptographic systems because of the inherent parallelism of the device.

NESSIE was a European Union (EU) initiative and a project with in the Information Society Technologies (IST) of the EU. Camellia was selected with three other block ciphers from the 42-encryption algorithms that were submitted. The other two block ciphers being MISTY1 and SHACAL-2. The AES algorithm was also selected but selected on its evaluation from the National Institute of Standards and Technology (NIST). Other algorithms included digital signatures, identification schemes, public-key encryption, MAC algorithms, and hash functions.

The Camellia algorithm is a 128-bit block cipher jointly developed by NTT and Mitsubishi Electric Corporation. The algorithm has also been submitted to other standardisation organisations and evaluation projects such as ISO/IEC JTC 1/SC 27, IETF, and TV-Anytime Forum. Previous Camellia implementations have been published in [4,5] but do not investigate a sub-pipelining architecture.

2. OVERVIEW OF CAMELLIA ALGORITHM

The Camellia algorithm processes data blocks of 128-bits with secret keys of lengths 128, 192, or 256 bits. Note that Camellia has the same interface as the AES (Advanced Encryption Standard). In our implementation we focus on the algorithm using a key length of 128-bits that is key agile. A key agile core requires that on each clock cycle new data and cipher key must be accepted.

A key length of 128-bits results in an 18 round Feistel structure. After the 6^{th} and 12^{th} rounds FL/FL^{-1} function layers are inserted to provide some non-regularity across rounds. There are also two 64-bit XOR operations before the first round and after the last, also known as pre- and post-whitening. The top-level structure of the algorithm can be seen in Figure 1, as well as the inner 6 round structure. The key schedule, discussed later in this section, generates subkeys for each round, FL/FL^{-1} layers, and pre- and post-whitening.

The FL-function is defined by:

$$Y_{R(32)} = ((X_{L(32)} \cap kl_{L(32)}) <<< 1) \oplus X_{R(32)}, \quad (1)$$

$$Y_{L(32)} = (Y_{R(32)} \cup kl_{R(32)}) \oplus X_{L(32)}, \quad (2)$$

where $Y_{L(32)}$ are the 32 most significant bits of the 64-bit output and $Y_{R(32)}$ are the 32 least significant bits. The FL^{-1} function is just the inverse of FL. Each round can be composed of an *F*-function with a XOR. A different subkey is applied to each *F*-function and the output is XORed with the previous but one result. The *F*-function is defined as

$$Y_{(64)} = P(S(X_{(64)} \oplus k_{(64)}). \quad (3)$$

The *P*-function is constructed only of XOR components and is a linear transformation from 8 input bytes to 8 output bytes. The *S*-function

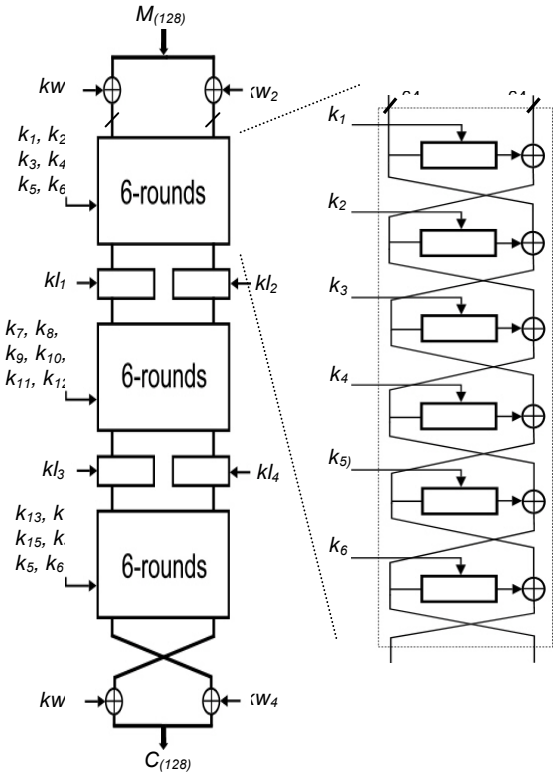

Figure 1. Top-level procedure of Camellia (left-hand side) showing inner structure of 6 Rounds (right-hand side)

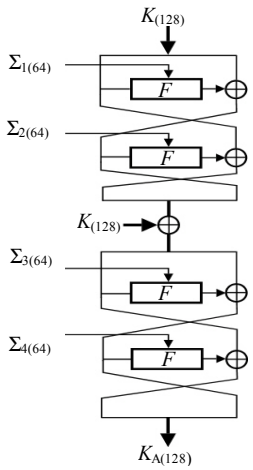

Figure 2. Key schedule architecture for Camellia

represents a substitution using one of 4 s-boxes that are defined by:

$$S_1(x) = h(g(f(x \oplus a))) \oplus b, \quad (4)$$

$$S_2(x) = S_1(x) <<< 1, \quad (5)$$

$$S_3(x) = S_1(x) >>> 1, \quad (6)$$

$$S_4(x) = S_1(x <<< 1), \quad (7)$$

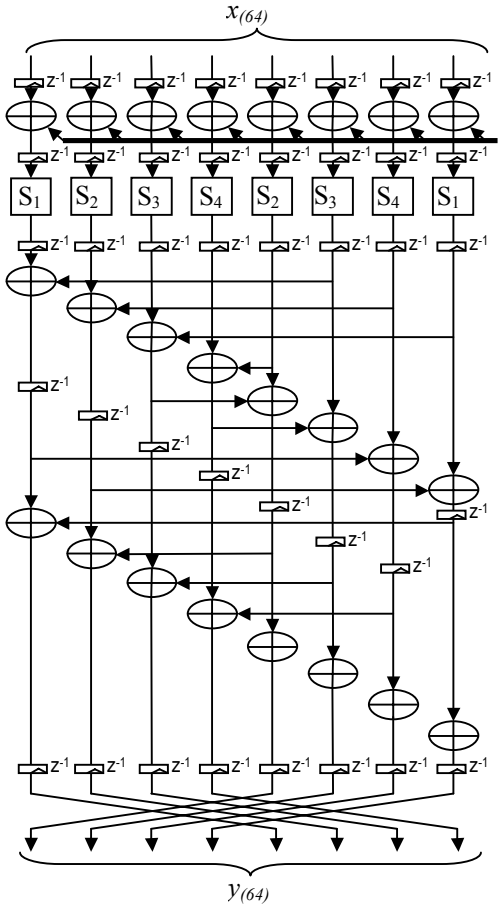

Figure 3. Sub-Pipelined F-Function

where f and h are linear mappings, g is an inverse over GF(2^8), and a, b are fixed constants.

The Camellia key schedule for a 128-bit key produces twenty-six 64-bit subkeys, for use in the 18 rounds, pre- and post-whitening and FL/FL^{-1} function layers. Figure 2 shows the first step that involves deriving a 128-bit variable $K_{A(128)}$ from the original secret key K. Then the second step involves generating further round keys by cyclic rotating either K (now K_L) or K_A by 15 or 17.

The Camellia decryption procedure is exactly the same as the encryption, it does not have any inverse functions like with in the AES algorithm, but the keys are needed in reverse order. So the subkey that was needed to encrypt the data block in round 18 is now needed to decrypt in round 1. This can cause a delay in the overall decryption process if the subkeys need to be produced first, before decryption can take place. This can be overcome when the decryption receives the pre-computed subkeys from the sender.

Table 1. Summary of CRYPTREC and NESSIE 128-bit Block Cipher Winners and Finalists on FPGAs

Algorithm	**Authors**	**Device**	**Area Slices** (BRAMS)	**Key agility**	**Throughput Gbits/sec**	**Efficiency Mbps/slices**
Camellia[1]	**Authors**	XC2Vp50	11,287 (88)	YES	**33.25**	-
AES-128[1]	Zambreno et al. [6]	XC2V4000	16938	YES	23.57	1.39
MISTY1[1]	Rouvroy et al. [7]	XCV1000	6,322	YES	19.34	3.06
SHACAL-1[2]	McLoone et al. [8]	XC2V4000	13,729	NO	17.02	1.24
IDEA[2]	Gonzalez et al. [9]	XCV600	12,026 (estimated)	NO	8.3	0.69
RC6[2]	Beuchat [10]	XC2V3000	8,554 (80)	NO (assumed)	15.2	-
Hierocrypt-3 [3]	Rogawski [11]	EPF10K130V	25811 LE	YES (assumed)	0.397	-
CIPHERUNICORN-A[3]	NEC [12]	EP20K1500E	7072 LE (66 ESB)	YES	0.044	-
SC2000[4]	Shimoyama et al. [13]	[0.25µm CMOS ASIC	65K gates	YES (assumed)	0.397	-

3. SUB-PIPELINING ARCHITECTURE

In this section we describe the sub-pipelined architecture but it includes previous classical encryption optimisation techniques to increase the algorithms throughput at the higher level within the algorithm. These include such optimisations as unrolling, round pipelining, and transformation partitioning. The implementation is key agile thus providing very high security and throughput as new data blocks can be encrypted on each clock cycle with a different key.

For the sub-pipelining architecture we have fully unrolled and added pipelining registers between each encryption round. We have then also added 5 stages of sub-pipelining shift registers into the *F*-Function. The registers have been added between the *P*-Function, *S*-Function and the *XOR*. With a pipeline stage also added into the P-Function and a register at the output of the *F*-function. This architecture produces a pipeline delay of 108 clock cycles but does not have an effect on throughput. Further improvements might be made with some algorithm optimisations or floor planning. Adding the registers into the *P*-Function means that the function now has a branch number of 3 or 2 instead of 5 depending on the data path. The architecture of *F*-Function can be seen in Figure 3 on the previous page. The registers can be seen in-between each of the *F*-function operations.

One other important optimisation techniques specific for the FPGA architecture is the use of dual-port block-RAM. Every round has its own associated F-function, as described earlier, and each function utilises 4 different s-boxes for byte substitution. Each F-function makes 2 calls to the same s-box. For this implementation we have chosen to use dual ported block-RAMs for the s-boxes.

4. RESULTS

We have implemented the sub-pipelined architecture on a Virtex-II pro XC2Vp50 device. The core was designed with Xilinx's System Generator, which produced the VHDL and associated files. The VHDL was synthesised with ISE 5.2 and verified with the test vectors that were

[1] Included in both CRYPTREC and NESSIE.
[2] Finalists not included in NESSIE portfolio.
[3] Implemented on a Altera FPGA device
[4] Only able to find ASIC implementation

submitted to NESSIE. The core runs at a frequency of 259MHz that results in a throughput of 33.25GBits/sec. The implementation requires 11,287 slices (47%) of the FPGA and uses 88 block-RAMs (37%). Generally comparisons against other encryption implementations can be difficult and can depend on architecture, device, embedded cores such as block-RAM, key agility, on-chip key scheduling, and other such criteria. Table 1 shows a list of other known high throughput FPGA architectures for NESSIE and CRYPTREC. For implementations that have used block-RAMs we have not included the efficiency. It is possible to calculate the extra number of slices that the block-RAMs would occupy if distributed memory was to be used, but this sort of calculation doesn't take into account the extra routing, which would have an effect on the total throughput and therefore the efficiency.

5. CONCLUSION

In this paper we have presented a sub-pipelined Camellia implementation that has a throughput of 33.25Gbit/sec. Compared to other published FPGA implementations this is the fastest known throughput of all block ciphers (128-bit) recommended by NESSIE and CRYPTREC. The importance of this algorithm has been proved in the open-call by NESSIE and CRYTREC.

The core is able to receive new secret keys and data on every clock cycle. We have utilised the available dual-port block-RAMs for use in the *F*-Function and been able to pipeline the algorithm to the lowest level and then apply this to the FPGA architecture. The core has a very high throughput and thus could be used in a secure high performance computing application. As the core takes less then 50% of the Virtex 2 pro it might be possible to put two cores down thus being able to obtain a throughput on one FPGA device of around 66Gbits/sec. This will be part of the work for the future.

6. REFERENCES

[1] Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T., "Specification of Camellia – 128-bit Block Cipher", URL: http://info.isl.ntt.co.jp/camellia/#Spec, September 2001.

[2] NESSIE, "NESSIE Project Announces Final Selection of Crypto Algorithms", IST-199-12324, February 2003.

[3] CRYPTREC, Information-Technology Promotion Agency, Japan, "CRYPTREC Report 2002",

[4] Denning, D., Irvine, J., Delvin, M., "A Key Agile 17.4Gbit/sec Camellia Implementation", proceedings of FPL'04, Antwerp, Belgium, Aug. 2004.

[5] Ichikawa, T., Sorimachi, T., Kasuya, T., Matsui, M., "On the Criteria of Hardware Evalution of Block Ciphers(1)", Techn report of IEICE, ISEC2001-53, September 2001.

[6] Zambreno, J., Nguyen, D., Choudhary, A., "Exploring Area/Delay Tradeoffs in an AES FPGA Implementation", proceedings of FPL'04, Antwerp, Belguim, Aug. 2004.

[7] Rouvroy. G., Standaert, F.X., "Efficient FPGA Implementation of Blcok Cipher MISTY1", proceedings IPDPS'03, France, April 2003.

[8] McLoone, M., McCanny, J. V., "Very High Speed 17 Gbps SHACAL Encryption Architecture", in Proc. Of the 13th Int'l Conference on Field-Programmable Logic and its Applications (FPL), Portugal, September 2003.

[9] Gonzalez, I., Lopez-Buebo, S., Gomez, F. J., Martinez, J., "Using Partial Reconfiguration in Cryptographic Applications: An Implementation of the IDEA Algorithm", in Proc. Of the 14th Int'l Conference on Field-Programmable Logic and its Applications (FPL), Portugal, September 2003.

[10] Beuchat, J. L., "FPGA Implementations of the RC6 Block Cipher", in Proc. Of the 13th Int'l Conference on Field-Programmable Logic and its Applications (FPL), Portugal, September 2003.

[11] Rogawski, M., "Analysis of Implementation Hierocrypt-3 Algorithm (and its comparison to Camellia algorithm) using ALTERA devices", Electronic Edition, CoRR cs CR/0312035, 2003.

[12] NEC Corporation, "Self Evaluation Report – CIPHERUNICORN-A", unpublished date.

[13] Shimoyama, T., Hitoshi, Y., Kazuhiro, Y., Takenaka, M., Itoh, K., Yjima, J., Torii, N., Tanaka, H., "Specification and Supporting Document of the Block Cipher SC2000", IST-199-12324, February 2003.