DEVELOPMENT OF BLOCK-CIPHER LIBRARY FOR RECONFIGURABLE COMPUTERS

Miaoqing Huang¹, Tarek El-Ghazawi¹, Brian Larson² and Kris Gaj³

¹ECE Department, The George Washington University, 801 22nd St NW, Washington, DC 20052, USA

²Silicon Graphics, Inc., 2750 Blue Water Road, Eagan, MN 55121, USA

³ECE Department, George Mason University, 4400 University Drive, Fairfax, VA 22030, USA

email: ¹{mqhuang,tarek}@gwu.edu, ²bgl@sgi.com, ³kgaj@gmu.edu

ABSTRACT

Reconfigurable computing is gaining rising attention as an alternative to traditional processing for many applications. Data encryption and decryption is one of these applications, which can get tremendous speedup running on FPGAs instead of microprocessors. We have developed a block-cipher library that covers 15 most popular encryption algorithms, and generated 35 bitstreams running on the SGI's latest version of a reconfigurable computer, RASC RC-100. The end-to-end throughput of 1.136 GB/s have been demonstrated for almost all ciphers, and was limited only by the input/output interface, rather than the FPGA processing time. The library is written in Verilog-HDL, and can be easily ported to other reconfigurable computing platforms. It provides means for cryptographers and computer scientists to program reconfigurable computers without the need for detailed knowledge of hardware design.

1. INTRODUCTION

Reconfigurable computing, as an alternative to traditional microprocessor-based computing, is getting more attention in recent years. Because of the inherent parallelism of hardware logic, the throughput speedup can be of orders of magnitude for certain applications.

Industry and academic community are looking for a uniform model to program the reconfigurable hardware efficiently. Hardware Description Language (HDL) is the primary means to code the FPGA. Meanwhile several HLLs, e.g. Mitron-C, Impulse-C, Handel-C, and MAP-C, are in the development to target the new platform. Except MAP-C that is exclusive with SRC-x reconfigurable computer, the other C-like languages are supposed to support different vendors' platforms.

To make it easy to program reconfigurable computing platforms, it is necessary to provide the libraries of certain complex algorithms, particularly for those who are not familiar how to explore the parallelism of hardware. A library of block cipher is a very good candidate to be implemented on reconfigurable computers. The first popular encryption

algorithm, DES, was designed specifically for hardware implementations 30 years ago. Most of the algorithms designed afterwards are very efficient in hardware as well.

In Section 2 we describe the basic concepts of block ciphers. In Section 3 the block-cipher library itself is presented, and we show the applications designed based on the library and their performance on RC-100 reconfigurable computer in Section 4. Finally, we summarize our work and discuss our future effort.

2. BLOCK CIPHER

In cryptography, a block cipher is any cipher which operates on fixed-length groups of bits, termed *blocks*, with an unvarying transformation, and the output block is only a function of the corresponding input block and does not depend on any previous input blocks. When encrypting, a block cipher might take a (for example) 128-bit block of plaintext as input, and output a corresponding 128-bit block of ciphertext. The exact transformation is controlled using a second input – the secret key. Decryption is similar: the decryption algorithm takes, in this example, a 128-bit block of ciphertext together with the secret key, and yields the original 128-bit block of plaintext.

Most block ciphers are constructed by repeatedly applying a simpler function. Each iteration is termed a *round*, and the repeated function is termed the *round function*; anywhere between 4 to 32 rounds are typical. Many block ciphers can be categorised as *Feistel networks*, or, as more general *substitution-permutation networks*. Arithmetic operations, logical operations (especially XOR), S-boxes and various permutations are all frequently used as components. More recently, the multiplication in *Binary Galois Field* has been introduced into the design of these new block ciphers, such as AES and LOKI97.

To encrypt messages longer than the block size (128 bits in the above example), several modes of operation may be used. The most popular modes include ECB, CBC, OFB and CFB. ECB mode is the most straightforward way to encrypt the plaintext. This mode of operation is adopted in the sim-

plest cryptanalysis means, the brute-forth attack, in which all the possible keys are tried given the plaintext and ciphertext. For other three modes of operation, there is data dependence between two subsequent data blocks, which prevents the processing of the second data block until the encryption of first data block is finished. In the application based on the block-cipher library, the ECB mode is adopted to take the benefit of hardware implementation in this paper.

3. DEVELOPING BLOCK-CIPHER LIBRARY

Following is the list of block-cipher algorithms included in the library (in alphabetic order),

3-Way (96 bits data block).

AES (128 bits data block): Advanced Encryption Standard.

Camellia (128 bits data block).

DEAL (128 bits data block): Data Encryption Algorithm with Larger blocks.

DES (64 bits data block): Data Encryption Standard.

FEAL (64 bits data block): Fast Data Encipherment Algorithm of 16 rounds.

IDEA (64 bits data block): International Data Encryption Algorithm.

LOKI97 (128 bits data block).

RC5-32/12/16 (64 bits data block).

RC6-32/20/16 (128 bits data block).

SAFER SK-64 (64 bits data block): Secure And Fast Encryption Routine of 10 rounds.

Serpent (128 bits data block).

SKIPJACK (64 bits data block).

TDEA (64 bits data block): Triple Data Encryption Algorithm.

TEA (64 bits data block): Tiny Encryption Algorithm.

3.1. Developing Principles

Following principles guide the development of block-cipher library,

(1)Fully pipelined. All the intermediate data and their related subkeys are registered every clock cycle. By this means, the core is ready to take a new data block and its key (or key schedule) every clock cycle.

(2)Simple and generic interface. Besides the clock signal, the inputs consist of data block and key (or subkeys), and the output is the final result. No other control signals exist and key expansion is included internally if it is feasible.

(3)High Speed. All cores run at 200 MHz or above.

(4)Comprehensive coverage. Representative 64-bit block ciphers, such as DES, IDEA, FEAL and RC5, are included in the library. Regarding the 128-bit block ciphers, three of the five finalists of "AES Process", Rijndael, Serpent and

Fig. 1. RASC Blade Hardware.

RC6, and some other popular algorithms are included in the library.

3.2. Defining the hardware cores in the library

For most of block-cipher algorithms, separate modules are written for encryption and decryption respectively. With respect to AES and some 128-bit block ciphers that have three different options of key length, different modules are coded to fit various key lengths. Tabel 1 lists the 35 modules designed for the library.

4. THE ENCRYPTION AND DECRYPTION APPLICATIONS BASED ON BLOCK-CIPHER LIBRARY

4.1. Hardware Architecture

We select SGI's RC-100 to design the applications based on the block-cipher library. SGI integrates its reconfigurable computing resource, FPGA chip, into the system by using its ccNUMA link, and names the technology RASC (Reconfigurable Application-Specific Computing). Multiple RASC blades can be added into the architecture easily as desired.

The latest version of RASC, RC-100, consists of two Virtex-4LX200 FPGA chips, two TIO ASICs, and one loader FPGA, as shown in Figure 1. FPGA chips can run up to 200 MHz.

4.2. Integrating the block-cipher modules onto RASC

Because the block-cipher library is written in Verilog HDL, it is natural to select HDL to program the bitstreams of block-cipher library. As shown in Figure 2, the user's logic is surrounded by the Core Services Block, which is defined by the

Table 1. 35 Modules in Block-Cipher Library

Module Name	Function Description & Key Expansion Implementation (+ or -)	end-to-end throughput (MB/s)			Slices Utilization
		RC-100	Xeon 2.8GHz	Speedup	
aesen	AES encryption, 128-bit key (+)	1,136 (one core)	23.71	48	47,940 (53%)
aesde	AES decryption, 128-bit key (+)	1,136 (one core)	18.19	62	52,852 (59%)
aesen192	AES encryption, 192-bit key (+)	1,136 (one core)	22.91	50	53,090 (59%)
aesde192	AES decryption, 192-bit key (+)	1,136 (one core)	17.02	67	62,897 (70%)
aesen256	AES encryption, 256-bit key (+)	1,136 (one core)	20.29	60	62,743 (70%)
aesde256	AES decryption, 256-bit key (+)	1,136 (one core)	14.86	76	69,841 (78%)
camelliaen	Camelliaen encryption, 128-bit key (+)	1,136 (one core)	3.50	325	40,230 (45%)
camelliade	Camelliaen decryption, 128-bit key (+)	1,136 (one core)	3.52	323	40,214 (45%)
dealen	DEAL encryption, 128&192-bit key, given the 6 subkeys (-)	1,136 (one core)	3.59	316	48,647 (54%)
dealde	DEAL decryption, 128&192-bit key, given the 6 subkeys (-)	1,136 (one core)	3.59	316	48,617 (54%)
deal256en	DEAL encryption, 256-bit key, given the 8 subkeys (-)	1,136 (one core)	2.70	421	61,755 (69%)
deal256de	DEAL decryption, 256-bit key, given the 8 subkeys (-)	1,136 (one core)	2.70	421	61,726 (69%)
des	DES en&decryption (+)	1,136 (two cores)	10.51	108	22,787 (25%)
feal16en	FEAL-16 encryption (+)	1,136 (two cores)	7.14	159	17,291 (19%)
feal16de	FEAL-16 decryption (+)	1,136 (two cores)	6.83	166	17,316 (19%)
ideaen	IDEA encryption (+)	1,136 (two cores)	16.43	69	21,809 (24%)
idea	IDEA en&decryption, given the 52 subkeys (-)	1,136 (two cores)	16.36	69	24,351 (27%)
loki97en	LOKI97 encryption, given the 48 subkeys (-)	1,136 (one core)	6.67	170	45,077 (50%)
loki97de	LOKI97 decryption, given the 48 subkeys (-)	1,136 (one core)	6.54	174	45,788 (51%)
rc5en	rc5(32/12/16) encryption (-)	1,136 (two cores)	47.63	24	20,842 (23%)
rc5de	rc5(32/12/16) decryption (-)	1,136 (two cores)	47.55	24	21,489 (24%)
rc6en	rc6(32/20/16) encryption (-)	1,136 (one core)	29.12	39	37,250 (41%)
rc6de	rc6(32/20/16) decryption (-)	1,136 (one core)	32.26	35	37,732 (42%)
safersk64en	SAFER SK-64 10-round encryption (+)	1,136 (two cores)	2.06	551	37,156 (41%)
safersk64de	SAFER SK-64 10-round decryption (+)	1,136 (two cores)	2.09	544	38,921 (43%)
serpenten	Serpent encryption (-)	1,136 (two cores)	8.99	126	35,188 (39%)
serpentde	Serpent decryption (-)	1,136 (two cores)	10.57	107	35,444 (39%)
skipjacken	SKIPJACK encryption (+)	1,136 (two cores)	1.70	668	56,822 (63%)
skipjackde	SKIPJACK decryption (+)	1,136 (two cores)	1.68	676	56,803 (63%)
tdea	3 DES en&decryption (+)	1,136 (two cores)	3.60	316	44,484 (49%)
teaen	TEA encryption (+)	1,136 (two cores)	23.07	49	26,232 (29%)
teade	TEA decryption (+)	1,136 (two cores)	24.22	47	26,352 (29%)
threewayen	3-Way encryption (+)	852 (one core)	18.91	45	15,335 (17%)
threewayde	3-Way decryption (+)	852 (one core)	8.24	103	15,560 (17%)
rc5key	RC5 (32/12/16) Key Expansion (+)	200 MKeys/s	0.545MKeys/s	367	41,510 (46%)

Fig. 2. RASC Algorithm Interface.

Fig. 3. Shift-Register showing the status of each stage in the pipeline.

vendor and in charge of the data movement among user's algorithm block, local memory and common memory.

All the modules are fully pipelined and data-driven. Under ideal scenario, each module is fed one new data block every clock cycle. For a 128-bit block cipher, the maximum throughput is 3.2 GB/s running at 200 MHz. To maximize the end-to-end throughput on RASC, we design the bitstream following two approaches. (1) Use separate local memory banks for storing input data and output data. (2) Use the streaming (*pingpong*) mode of RASC to overlap the data movement and data processing.

To make sure the surrounding logic only reads the valid result no matter when and how a valid data block is fed to the block cipher module, an one-bit wide shift-register of same depth as the latency of block-cipher module is designed besides the block-cipher itself, as depicted in Figure 3.

Every clock cycle the contents of shift-register shifts one bit to the right and "1" (of valid input) or "0" (of dummy input) is put at the head of channel. Now, by observing the content of tail bit of the shift-register, the surrounding logic can tell the content at the output is valid result or dummy data.

4.3. The performance of block-cipher applications on RASC

Although SGI claims the NUMALink-4 can sustain up to 6.4 GB/s bandwidth per FPGA, we found the real end-to-end throughput is far lower than the ideal throughput, 3.2 GB/s. If the FPGA runs at full speed (200 MHz) targeting a big chunk of raw data, say 1 GB, the end-to-end throughput is 1.136 GB/s (see Table 1). In another word, we can say the bitstream is always data-hungry and almost $\frac{2}{3}$ processing power is wasted. The up-bound of I/O bandwidth is hit

currently. But at the same time, we can see there is a big room for lifting the roof of I/O efficiency.

The software implementation of most algorithms are reference codes provided by the authors respectively except AES and IDEA, which are contributed by OpenSSL library. But even we assume the optimized codes are 10 times faster than the reference codes used in this paper, we still see a significant speedup from the FPGA. Furthermore, we can find the performance of hardware doesn't degrade when the complexity of block ciphers increases. For example, the hardware modules of DES and Triple-DES have the same throughput, yet the software throughput of Triple-DES is only $\frac{1}{3}$ of that of DES.

4.4. Porting the Block-Cipher Library to other platforms

The library is written in Verilog-HDL without any platform-specific interface. This feature makes it almost effortless to expand the library to other platforms that can integrate HDL modules directly. For example, we wrote the similar applications based on the block-cipher library on SRC-6 reconfigurable computer. The end-to-end throughput on SRC-6 under Carte release 2.1 is 675 MB/s. Several factors can be blamed for the performance degrade on SRC-6. (1) The frequency of FPGA is fixed at 100 MHz. (2) Carte release 2.1 does not support a complete overlap of data transfer and data processing. (3) The I/O efficiency is slower than that of RC-100.

5. CONCLUSION

A comprehensive hardware block-cipher library has been developed for reconfigurable computers. The hardware modules in the library are fully pipelined to achieve the maximum throughput and the targeting frequency is 200 MHz. Without any platform-specific interface, this library can be ported to various reconfigurable computing platforms, and one case of SGI RASC RC-100 is discussed in detail in this paper. Through the application, we find a good means dealing with data transfer between main memory and local memory is critical to exploit the potential of FPGA's processing capability in an integrated platform. In the future, we plan to expand the library to cover asymmetric-key encryption algorithms and port it to more platforms as well.

6. REFERENCES

- [1] A. Menezes, P. van Oorschot, S. Vanstone, "Handbook of applied cryptography", Boca Raton : CRC Press, pp. 223-282, 1997.
- [2] Silicon Graphics, Inc., "Reconfigurable Application-Specific Computing User's Guide (007-4718-005)", July 2006.