ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 6, June 2017

Reduced Instruction Set Computer Processor for Cryptographic application

[1] Vijayalaxmi P, [2] Chandrakumar H S, [3] Mohankumar B N, Assistant Professor, RRIT, Bangalore

***Abstract*— Security is one of the most important features of industrial products. Cryptographic algorithms are mainly used for this purpose to obtain confidentiality and integrity of data in industry. One of the main concerns of researchers in designing cryptographic algorithms is efficiency in either soft- ware implementation or hardware implementation. However, the efficiency of some well-known algorithms is highly questionable. The main goal of this paper is to present a novel processor architecture called CIARP (stands for Crypto Instruction-Aware RISC Processor) being feasible for high speed implementation of low throughput cryptographic algorithms. CIARP has been designed based on a proposed instruction set named Crypto Specific Instruction Set (CSIS), that can speed up encryption and decryption processes of data.**

***Index Terms*— CSIS, RISC, CIARP, Cryptographic algorithms**

I. INTRODUCTION

Old-time necessity for security and data protection against unauthorized access to classified information in many indus- tries especially in military application is undeniably sobering. Hence, Cryptography plays a significantly important role in the security of data transmission.

On one hand, with developing computing technology, im- plementation of sophisticated cryptographic algorithms has become feasible. On the other hand, stronger cryptographic specifications are needed in order to be reluctant to possible threats. Some well-known examples of cryptographic algo- rithms are DES and AES. The Data Encryption Standard (DES) has been the U.S. government standard since 1977. However, now, it can be easily cracked inexpensively. In 2000, the Advanced Encryption Standard (AES) was substituted for the DES to meet ever-increasing requirements for security.

General purpose processors are mostly used to speed up data manipulation and information processing in systems. Never- theless, these processors are not performance efficient when they are utilized for data encryption and decryption, mainly because many cryptographic algorithms need bit-oriented op- erations in contrast with what general purpose processors work based on. Data are manipulated by word-oriented operations in processors. Consequenty, bit-oriented manipulating by means of them poses undesirable time overhead although some ap- proaches like hardware/software co-design can mitigate the problem. By the way, the most well-known approaches which are used for implementations of cryptographic algorithms can be enumerated as:

- General purpose processors and DSP.
- Application Specific IC (ASIC).
- Reconfigurable hardware or encryption systems (FPGAs).
- Crypto processor and Crypto-Coprocessor.

The main goal of this paper is to present an especial pur- pose instruction set called CSIS (Crypto Specific Instruction Set) that can be utilized for cryptographic algorithms. This instruction set is structured based on some bit and byte- oriented operations. In order to show their appropriateness, a cryptographic processor named CIARP (Crypto Instruction- Aware RISC Processor) has been designed

By this novel pro- cessor, the time order of the standard cryptographic algorithms has been reduced compared to those run on general purpose processors. We believe that CSIS and CIARP open a new field in implementation of cryptographic algorithms.

The rest of the paper is organized as follows: a general description of CIARP is presented in section II. Afterwards, The assembler of CIARP is presented in section III. Section IV describes the simulation and implementation results of the CIARP processor. Finally, the paper is concluded in section V.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 6, June 2017

II.CIARP PROCESSOR

CIARP is a 32-bit processor that its architecture has been designed in a way to be modular. In other words it is composed of some basic building blocks such as multiplexers, decoders, registers and counters and whole processor has been implemented by interconnecting these modules that makes it a fully-synthesizable processor.

Each instruction cycle in CIARP is composed of four T-states. The CIARP instruction set includes two categories of computing instructions. One group contains word-oriented instructions like those can be found in general purpose proces- sors such as arithmetic and logical operations. Another class of instructions is bit and byte oriented so that they can be exploited in implementing algorithms needing such features.

CIARP reaps the benefits of RISC architectures to speed up running programs by utilizing fewer instructions and address- ing modes and more registers as well as pipeline technique and register windowing in its structure. The memory is accessible via only two instructions (load and store). Furthermore, this processor has some control instructions like unconditional and conditional branch instructions.

The main part of CIARP is IR-Decoder that has been designed to decode the instructions. In other words, according to the given instruction, it determines which unit in the next stage should be activated. One of the distinct features of

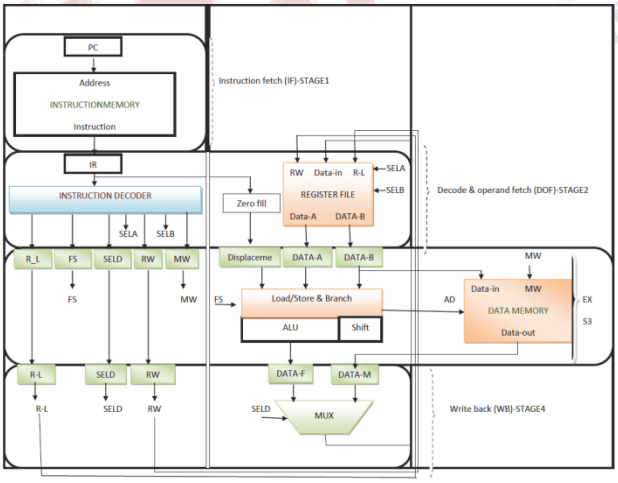

Fig. 1. The internal structure of CIARP.

CIARP is the barrel shifter that is able to shift a word as many as it is desired in one clock pulse. The most important instructions of CIARP are BITP, BYTP and RXOR that will be disscussed further.

A. The Register File

Here, CIARP registers are presented. In all registers, the bit with number zero is assessed as the LSB (Least Significant Bit) while the MSB (Most Significant Bit) is recognized as the bit with number 31.

1) General Purpose Registers (GPR): These 32-bit registers are used as the source or the destination in most of instructions.

2) Permutation Registers (PR): PRs are 32-bit registers that specify the pattern of permutation in crypto-specific instructions. They determine the number of bits or bytes in bit-oriented or byte oriented instructions.

3) Program Counter (PC): It is a 32-bit counter that ascertains the sequence of the program.

4) Address Register (AR): A 32-bit register that is used for memory addressing.

5) Program Status Word (PSW): It is 32-bit register that each bit of it determines a status except some bits that have been considered for future use. Fig. 2 shows its structure and the functionality of each bit.

GT and EQ are used in the comparison instructions and refreshed when a comparison instruction is issued. If the first register is greater than the second, GT bit is set to 1 while EQ bit is set to 1 if both registers are equal. In other cases,

31...10	9...7	6	5	4	3	2	1	0
Reserved	CWP	IE	PF	CF	OF	SF	EQ	GT

Fig. 2. The functionality of each bit in PSW.

both of them are filled by zero. SF is known as the Sign Flag and is set to 1 when the result of an arithmetic instruction is negative. C F is used to save the carry generated by arithmetic operations and by RXOR as well. When overflow occurs, OF is set to 1. In arithmetic and logical instructions, if the result has even parity, P F is set to 1. I E is a writable bit to enable maskable interrupts. Finally, C W P known as the Current Window Pointer of CIARP is a 3-bit field to keep the sequence of window switching in the 8-window register file of CIARP.

B. Addressing Mode

Most instructions work directly with registers and the mem- ory is accessible through load or store instructions like what are designed in RISC processors. There are four addressing modes in CIARP: Immediate addressing, Register Direct Ad- dressing, Register Indirect Addressing, and Index addressing.

1) Immediate Addressing Mode: It can be used for all instructions except for load and store instructions. This kind of addressing mode is shown in Fig. 3.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 6, June 2017

2) Register Direct Addressing Mode: As the former addressing mode, It can be used for all instructions except for load and store instructions. This kind of addressing mode is shown in Fig. 4.

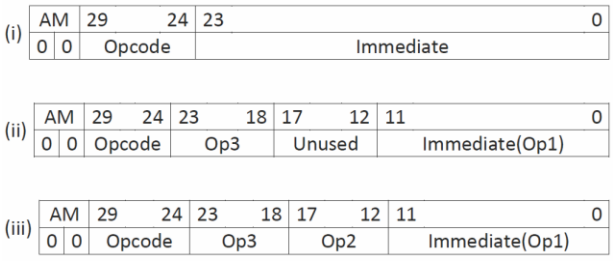

Fig. 3. Immediate addressing mode for (i) Single operand instructions (ii) two operand instructions and (iii) three operand instructions.

Fig. 4. Register direct addressing mode for (i) three operand instructions (ii) four operand instructions and (iii) branch statements.

3) Register Indirect Addressing Mode: It is utilized in the load and store instructions. For the load instruction, the source field refers to a register containing the address of the desired operand. By contrast, in store instruction, the destination field uses this addressing mode.

4) Index Addressing Mode: As the previous addressing mode, it is used in load and store instructions. There are two fields to indicate this kind of addressing mode. One field is reserved for displacement while the other field refers to the index register. The effective address in this mode is as follows:

C. The Pipeline Structure of CIARP

Pipelining is an implementation technique in which the phases of the instruction cycle are executed simultaneously. While an instruction is being fetched from the memory, the former instructions are being processed in following stages.CIARP takes the advantages of a 4-stage pipeline struc- ture [1]. Each of fetch cycle, decode cycle, execute cycle, and write back cycle has been considered one stage in the CIARP pipeline architecture. Hence, all instructions run in four clock periods. Fig. 1 illustrates the pipeline architecture of CIARP.

D. Register Windows

Register Files represent a substantial portion of the energy budget in modern microprocessors [2] and [3]. The techniques for reducing the size include sharing an entry among several operands with the same value [4] and [5], and dividing the register storage hierarchically [6] and [7]. The register model of CIARP is the same as the SPARC architecture and uses the same register windowing mechanism that is described in this section.

The SPARC architecture uses a windowed register file model in which the file is divided up into groups of registers called windows [8]. This windowed register model simplifies compiler design and accelerates procedure calls.

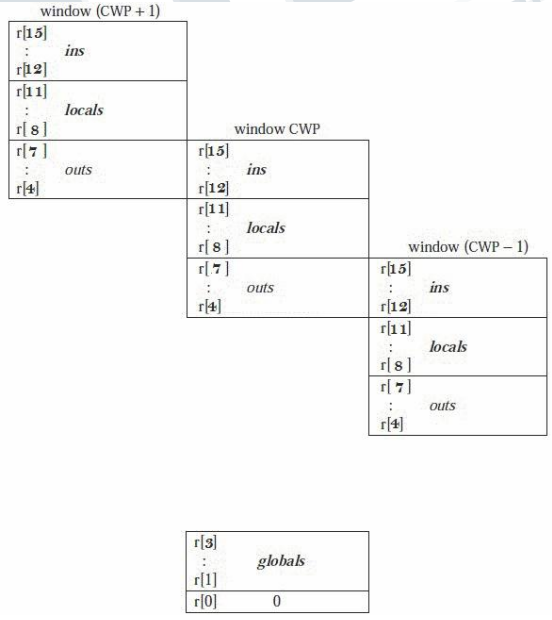

Fig. 5. The functionality of the designed register window.

The implementation of the proposed CIARP processor con- tains 68 GPR registers and 64 PR registers that are 32-bit wide and are classified into a set of 128 window registers and a set of 4 global registers. The 128 window registers are grouped into 8 sets of 12 GPR registers and 12 PR registers called windows. Thus, the register file consists of 8 register windows where each window includes of a set of 24 registers. While a program is running, it has access to 28 32-bit processor registers

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 6, June 2017

which include 4 global registers plus 24 registers that belong to the current register window. As it is shown in Fig. 5, The first 8 registers (GPR and PR) in the window are called the in registers (in4-in7 (GPR) and in4-in7 (PR)). When a function is called, these registers may contain arguments that can be used. The next 8 registers are the local registers (local8-loca11 (GPR) and local8-local11 (PR)) which are scratch registers that can be used for anything while the function executes. The last 8 registers are the out registers (out12-out15 (GPR) and out12-out15 (PR)) which the function uses to pass arguments to functions that it calls. At any given time, a program can access an active 28-register window (24 window registers) and the four global registers. The current active window (the window visible to the programmer) is identified by the Current Window Pointer (CWP). Either incrementing or decrementing the CWP results in an eight register overlap between windows. This overlap of window registers is used to pass parameters from one window to the next.

E. The Structure of Barrel Shifter in CIARP

The key objective of today's circuit design is to increase the performance without the proportional increase in power

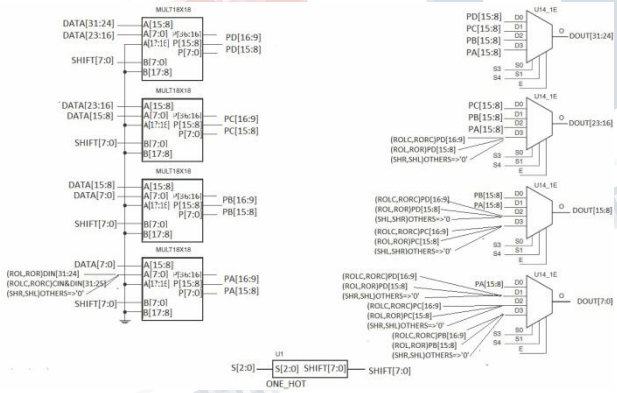

Fig. 6. The barrel shifter based on multipliers.

consumption. Shifting and rotating are required in many operations such as arithmetic and logical operations, address decoding and indexing etc. Barrel shifters, which can shift and rotate multiple bits in a single cycle, have become a common design choice for high speed applications. For this reason, CIARP reap the benefits of the method that uses multipliers in its barrel shifter [9] and [10]. A 32-bit barrel shifter requires thirty-two, 32-to-1 multiplexers. A 32-to-1 multiplexer can be implemented in a Spartan-3a device using two CLBs. Only sixty four CLBs are required to accomplish all the required multiplexing. By using a multiplier-based barrel shifter, a 32- bit barrel shifter is built using four 8-bit barrel shifters and thirty two 4-to-1 multiplexers.

The diagram on the left side of Fig. 6 is a single-cycle, 32- bit barrel shifter. The input bus is broken down into four 8-bit words. The data is processed in two stages. The first stage is constructed of the 8-bit barrel shifters. This stage provides the fine shifting, moving the bits from adjoining bytes. Passing the first stage, the appropriate bits are stored in a byte; however the bytes need to be reordered. The reordering of the bytes, or bulk shifting, is applied in the second stage, as shown on the right in Fig. 6. The 8-bit barrel shifter requires the shift amount being in the one-hot encoded format where three LSBs are used to control the fine shifting, and the two MSBs are used to control the bulk shifting [10].

CIARP supports six kinds of shift operations (SHL, SHR, ROR, ROL, ROLC, and RORC) and uses Hardware sharing technique to reduce the number of slices and the hardware cost without missing performance. Table I shows the synthesis

TABLE I
THE SYNTHESIS RESULTS OF THE BARREL SHIFTER IN CIARP.

	Used	Available	Utilization
Number of slices	117	11264	1%
Number of occupies	107	11264	1%

results of the barrel shifter of CIARP.

F. Interrupt

When an interrupt occurs by peripheral device, an 8-bit vector is placed on the bus Interrupt Vector and the CPU in interrupt cycle jump to the location of the memory which interrupts vector indicates.

G. The Crypto-Purpose Instructions

1) The BITP Instruction: It is a bit-oriented instruction with four operands that is demonstrated in Fig 7. A sample use of BITP is shown below:

BITP GPR1, GPR2, PR1, PR2

It is a bit permutation on lower 16 bits of GPR2 (as the source register) based on the pattern determined by PR1 and PR2 registers, and the result is stored in GPR1 as the target register.

The PR registers are divided into eight parts containing 4 bits. Each part is able to indicate one bit out of lower 16 bits in GPRs. In the BITP instruction, PR2 specifies the bits of GPR2 (the source register) to be copied into those bits of GPR1 (the

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 6, June 2017

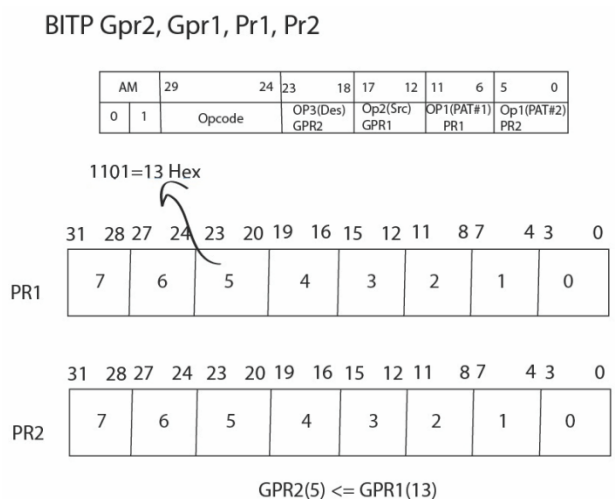

Fig. 7. The functionality of the BITP instruction.

target register) that are designated by PR1. For example if the value of part 5 of PR2 and PR1 is 13 and 5 respectively, the content of the bit 13 of GPR2 will be copied into the bit 5 of GPR1. Any GPR can be substituted for GPR1 and GPR2 as well as PR1 and PR2 which can be replaced by any PR.

2) The BYTP Instruction: It is a byte-oriented instruction with three operands that is demonstrated in Fig 8. A sample use of BYTP is shown below:

BYTP GPR1, GPR2, PR1

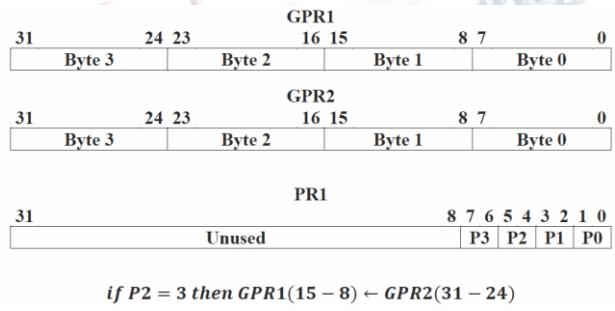

Fig. 8. The functionality of the BYTP instruction.

It is a byte permutation on the four bytes of GPR2 (as the source register) based on the pattern determined by PR1, and the result is stored in GPR1 as the target register.

The PR registers are divided into four parts containing 2 bits. Each part is able to indicate one byte out of the four bytes in GPRs. In the BYTP instruction, PR1 specifies the bytes of GPR2 (the source register) to be copied into relative bytes of GPR1 (the target register).

For example if the value of part 1 of PR1 is 3, the contents of byte 3 of GPR2 will be copied into byte 1 of GPR1. Any GPR can be substituted for GPR1 and GPR2 as well as PR1 and PR2 which can be replaced by any PR.

3) The RXOR Instruction: A sample use of BYTP is shown below:

RXOR GPR1, PR1

As it is shown in Fig 9, this instruction applies the XOR operation on all bits of GPR2 according to contents of PR1 and stores the result into CF which is a flag bit in PSW.

III. CIARP ASSEMBELER

An embedded system is a special-purpose computer system designed to perform one or a few dedicated functions. It is usually embedded as part of a complete device including hardware and mechanical parts. Since embedded systems usually use processors with limited computational power and few hardware resources, it is important to write efficient programs controlling the systems [11]. In order to automate generating executable code for CIARP, we have designed an assembler that converts assembly instructions written in a file into machine language instructions [12].

IV. SIMULATION RESULTS

Having been completed the design process, in order to verify its performance, the processor was implemented on hardware using VHDL and Xilinx ISE synthesis tool. The target FPGA used in the implementation was Spartan3a xc3s1400an device from Xilinx. The result of the synthesis showed that the full implementation needs 5,127 (45% of 11264) slices and its critical path limited the maximum clock frequency on 69 MHz. The post-synthesis simulation verified the design goals of CIARP.

V. CONCLUSION

In this paper a novel architecture for crypto processors were presented. The proposed idea of CSIS was implemented by means of the architecture in a way to reduce the time order of the execution of the cryptographic-related instructions. Three instructions called BYTP, BITP and RXOR are introduced. The simulation results show that they are able to speed up symmetric key algorithms realated to stream ciphers,

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 6, June 2017

block ciphers and hash functions.

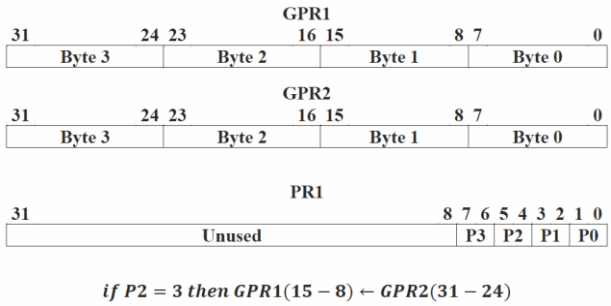

Fig. 9. The functionality of the RXOR instruction.

REFERENCES

[1] D. A. Patterson and J. L. Hennessy, Computer Organization and Design: The Hardware/Software Interface, 3rd ed., Morgan Kaufmann, 2005.

[2] D. R. Gonzales, "Micro-RISC architecture for the wireless market," IEEE Micro, pp. 30-37, July/August 1999.

[3] A. Kalambur and M. J. Irwin, "An extended addressing mode for low power," In Proc. of the IEEE Symposium on Low Power Electronics,pp. 208-213, August 1997.

[4] S. Jourdan, R. Ronen, M. Bekerman, B. Shomar, and A.Yoaz, "A Novel Renaming Scheme to Exploit Value Temporal Locality through Physical Register Reuse and Unification," In Proc. of the 31st MICRO, pp. 216- 225, 1998.

[5] S. Balakrishnan and G.S. Sohi, "Exploiting Value Locality in Physical Register Files," In Proc. of 36th MICRO, pp. 265-276, Dec 2003.

[6] J.-L. Cruz, A. Gonzalez,M. Valero, N.P. Tophanm, "Multiple-Banked Register File Architectures," In Proc. of 27th ISCA, pp. 316-325, June 2000.

[7] R. Balasubramonian, S. Dwarkadas, and D.H. Albonesi, "Reducing the Complexity of the Register File in Dynamic Superscalar Processors," In Proc. of 34th MICRO, pp. 237-248, Dec.2001.

[8] The SPARC Architecture Manual Version 8, SPARC International, Pren- tice Hall, 1992.

[9] I. Hashmi, H.M.H Babu, "An Efficient Design of a Reversible Barrel Shifter," International Conference on VLSI design, pp. 93-98, jan. 2010. [10] Paul Gigliotti, "Implementing Barrel Shifters Using Multipliers," Xilinx Corp. , August 17, 2004.

[11] K. Nakano, Y. Ito, "Processor, Assembler, and Compiler Design Edu- cation Using an FPGA," IEEE International Conference on Parallel and Distributed Systems, pp. 723-728, Dec. 2008.

[12] M. Morris Mano, Computer System Architecture, Prentice-Hall, 1993.