International Journal of Soft Computing and Engineering (IJSCE)
ISSN: 2231-2307, Volume-3 Issue-5, November 2013

A Novel RISC Processor with Crypto Specific Instruction Set

Bhagyasree. P , C. Silpa, M. J. C. Prasad

Abstract*— *Old-time necessity for security and data protection against unauthorized access to classified information in many industries 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, implementation 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 algorithms are DES and AES.

One of the main concerns in designing cryptographic algorithms is efficiency in either software or hardware implementation. General purpose processors are mostly used to speed up data manipulation and information processing in systems. Nevertheless, these processors are not performance efficient when they are utilized for data encryption and decryption.

A novel RISC processor with Crypto Specific Instruction Set has been designed such that the processor is a Crypto Instruction-Aware RISC Processor, that makes the encryption and decryption processes of data faster, with the help of techniques like pipelining, register windows and a special architecture of barrel shifter. The main goal of this paper is to present a novel processor architecture being feasible for high speed implementation of low throughput cryptographic algorithms.

Keywords*—*Cryptographic Algorithms, Pipeline Technique, Register Windowing Technique, RISC Processor.

I. INTRODUCTION

In this age of universal electronic connectivity, of viruses and hackers, there is a need to protect data and resources from disclosure, the disciplines of cryptography have been matured, that lead to the development of practical, available applications to enforce the security. One such application is the proposed technology. This field has continued innovations and improvements. With many advantageous features, this novel RISC Processor with crypto specific instruction set has been designed. A special purpose instruction set called Crypto Specific Instruction Set that can be utilized for cryptographic algorithms. This instruction set is structured based on some bit and byte-oriented operations. A proper appropriateness of this instruction set is shown using the processor named as novel RISC processor with Crypto Specific Instruction Set. With this processor, time order of standard cryptographic algorithms has been reduced compared to those that run on general purpose processors.

The rest of the paper is organized as follows: a general description of the proposed processor is presented in section II. Then the architecture of the processor is explained in section III. Afterwards, the assembler of the processor is presented in section IV. Section V describes the simulation and implementation results of the Crypto Specific Instruction Set Processor. Finally, the paper is concluded in section VI.

II. PROPOSED PROCESSOR

This novel RISC processor with Crypto Specific Instruction Set is a 32-bit processor that reaps the benefits of RISC architectures to speed up running programs by utilizing less number of instructions and fewer addressing modes and with more registers as well as pipeline technology and register windowing in its structure. This processor is designed based on the Harvard architecture. 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 processor has the barrel shifter that can shift a word as many as it is desired in one clock pulse. The RISC processor with crypto specific instruction set includes two sets of instructions that can be computed. One class of the instruction set contains the word-oriented instructions like those can be found in general purpose processors such as arithmetic and logical operations. The other group of instructions is bit and byte oriented, that can be used to implement the algorithms, in which such features are needed. The most important instructions of the processor are BITP, BYTP and RXOR that will be discussed further.

III. PIPELINE STRUCTURE OF THE PROCESSOR

Pipelining is a technology 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. This RISC Processor takes the advantages of a 4-stage pipeline structure. Each of fetch cycle, decode cycle, execute cycle, and write back cycle has been considered as one stage in this processor pipeline architecture. Hence, all the instructions run in four clock cycles that is each instruction cycle is composed of four T-States in this processor. (The number of stages in which one instruction of a program is executed completely is called an instruction cycle). Fig.1 illustrates the pipeline architecture of the proposed processor.

Manuscript Received November, 2013.

Bhagyasree. P, Electronics and Communications Engineering, Malla Reddy Engineering College, Secunderabad, India.

C. Silpa, Electronics and Communications Engineering, Malla Reddy Engineering College, Secunderabad, India.

Dr. M.J.C.Prasad, Electronics and Communications Engineering,, Malla Reddy Engineering College, Secunderabad, India.

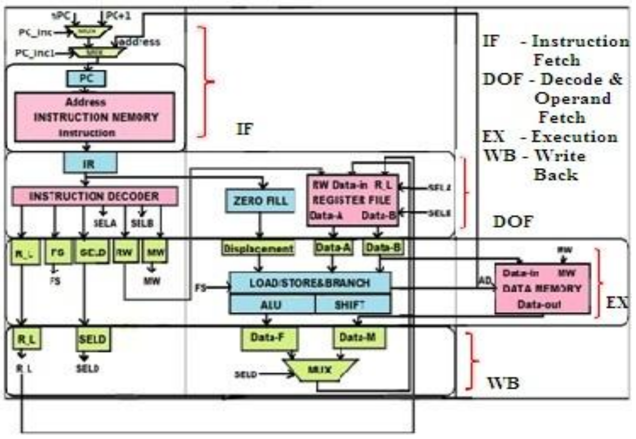

Fig.1: 4-Stage Pipelined Architecture of the RISC Processor With Crypto Specific Instruction Set

A. Instruction Memory

Instruction memory, as the name defines, stores all the instructions that are to be executed. It must be non-volatile and fast. Internal ROM is used as instruction memory, because it is the fastest one and avoided the need for external storage.

B. Instruction Decoder

The instruction decoder is designed to decode all the instructions, that is, it determines which unit in the next stage is to be activated, according to the given instruction. It gives the opcode, SELA, SELB, MW, RW, F_S and R_L. opcode is used to select the which instruction have to execute and SELA, SELB are the address of the register file. MW, it tells the data memory in write condition or read condition:
1. MW =0 memory in read.
2. MW=1 memory in write.
F_S signal is used to select the address of the instruction memory or data memory. RW, R_L signals for the writing and reading the register file.

C. Register File

Here, different registers of this processor are presented. In all registers, the bit with number zero is termed as LSB (Least Significant Bit) while the MSB (Most Significant Bit) is given as the bit with number 31.

1. General Purpose Registers (GPRs): These are 32-bit registers and are used as the source or the destination in most of the instructions.
2. Permutation Registers (PRs): PRs are 32-bit registers which specify the pattern of permutation in crypto-specific instructions. The registers 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 processes the sequence of the program.
4. Address Register (AR): This is a 32-bit register that is used for memory addressing.
5. Program Status Word (PSW): It is a 32-bit register that each bit of it gives the 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.

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

Fig.2: Functionality of each bit in PSW

In other cases, 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. CF is used to save the carry generated by arithmetic operations and by RXOR as well. If any overflow occurs, OF is set to 1. In arithmetic and logical instructions, if the result has even parity, PF is set to 1. IE is a writable bit to enable masks able interrupts. Finally, Current Window Pointer (CWP) of this processor is a 3-bit field which keeps the sequence of window switching in the 8-window register file of this RISC processor.

D. Addressing Mode

Most instructions work directly with registers and the memory is accessible through load or store instructions like what are designed in RISC processors. There are four addressing modes in this proposed RISC Processor. They are: Immediate Addressing, Register Direct Addressing, Register Indirect Addressing, and Index Addressing. According to the application, the required addressing modes can be included. The instruction formats of immediate addressing mode and register direct addressing mode are in various kinds as shown in figures.

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.
2. Register Direct Addressing Mode: As 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.
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. In the similar manner, in *store* instruction, the destination field uses this addressing mode.
4. Index Addressing Mode: As the previous one, it is used in load and store instructions. There are two fields to indicate this kind of addressing mode. One of the fields is reserved for *displacement* while the other field refers to the index register.

31 30	29 24	23 0
AM 0 0	Opcode	Immediate

(a)

31 30	29 24	23 18	17 12	11 0
AM 0 0	Opcode	Op3	Unused	Immediate(Op1)

(b)

31 30	29 24	23 18	17 12	11 0
AM 0 0	Opcode	Op3	Op2	Immediate(Op1)

(c)

Fig.3: Immediate Addressing Mode for (a) single operand instructions (b) two operand instructions and (c) three operand instructions

31 30	29 24	23 18	17 12	11 6	5 0
AM 0 1	Opcode	Op3	Op2	Op1	Unused

(a)

31 30	29 24	23 18	17 12	11 6	5 0
AM 0 1	Opcode	Op3	Op2	Op1	Op0

(b)

Fig.4: Register Direct Addressing Mode for (a) three operand instructions (b) four operand instructions

E. Register Windows

Register file represents a substantial portion of the energy budget in modern microprocessors and the techniques to reduce the size include sharing an entry among several operands with the same value and dividing the register storage hierarchically. The register model of this processor architecture uses the register windowing mechanism in which the file is divided up into eight groups of registers called windows. The compiler design gets simplified and the procedure calls are accelerated with the help of this register windowing technique.

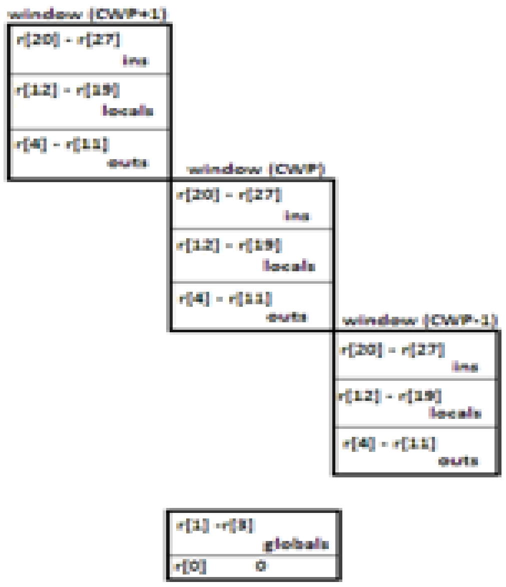

Fig.5: Register Windowing Technique

The proposed processor contains 68 GPRs and 64 PRs (32-bit) and is divided into a set of 128 window registers and a set of 4 global registers. The 128 window registers are made into 8 sets of 12 GPRs and 12 PRs called windows. Hence, the register file consists of 8 register windows in which each window has a set of 24 registers. As 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 are used. The next 8 registers are the *local* registers (local8-loca11 (GPR) and local8-local11 (PR)) called scratch registers, used for anything when a function executes. The last 8 registers are *out* registers (out12-out15 (GPR) and out12-out15 (PR)) which the function uses to pass arguments to functions that it calls. At a given time, a program has to access an active 28-register window. Current Window Pointer (CWP) is used to locate the present window. The incrementing or decrementing the CWP results in an eight register overlap between windows as shown in fig.6. This overlap is used to pass parameters from one window to the next. Here w0-w7 is the eight registers windows.

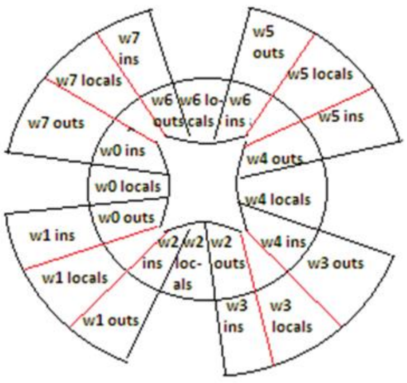

Fig.6: Overlapping of Register Windows

F. ALU & Shift Block

The arithmetic and logic unit (ALU) performs all the arithmetic and logical operations like addition, subtraction, multiplication, division, etc.; AND, OR, NOT, XOR, NAND etc, respectively. The shift block performs all the shift operations like SHL, SHR, ROL, ROR, ROLC, and RORC. To perform the all kind of the shifting operations, a barrel shifter with a special architecture is used in this processor.

G. Structure of Barrel Shifter

The key objective of today's circuit design is to increase the performance without the proportional increase in power consumption. Barrel shifters, which can shift and rotate multiple bits in a single cycle. This has become a common design choice for high speed applications. For this reason, this processor takes the advantages of the method that uses multipliers in its barrel shifter.

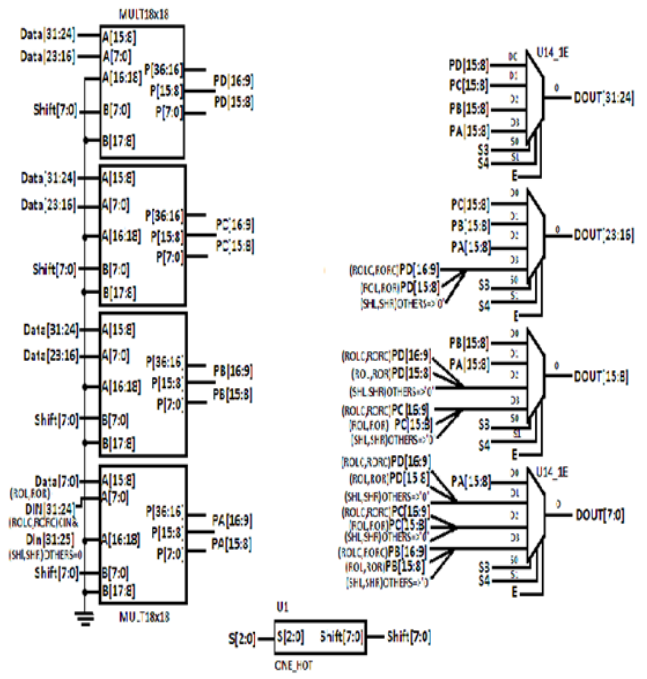

Fig.7: Barrel Shifter based on Multipliers

A 32-bit barrel shifter requires 32 32-to-1 multiplexers. A 32-to-1 multiplexer is implemented in any device using 2 CLBs. So, 64 CLBs are required for multiplexing. By using a multiplier-based barrel shifter, a 32- bit barrel shifter can be built using only 4 8-bit barrel shifters and 32 4-to-1 multiplexers as shown in fig.7. The diagram on the left side of figure6 is a single-cycle, 32- bit barrel shifter. The input data is broken down into four 8-bit words. The data will be processed in two stages. The first stage is constructed of the 8-bit barrel shifters. This stage gives the fine shifting, moving the bits from adjoining bytes. Passing the first stage, the appropriate bits are stored in a byte; however the bytes are to be reordered. This reordering of the bytes, or bulk shifting, is applied in the second stage, as shown on the right in figure7. The 8-bit barrel shifter requires the shift amount being in the one-hot encoded format where the three LSBs are used for fine shifting and the two MSBs are used for bulk shifting. This processor supports all the six kinds of shift operations (SHL, SHR, ROR, ROL, ROLC, and RORC) without extra hardware by just changing the input logic and uses hardware sharing technique to reduce the number of slices and the hardware cost without missing performance.

H. Load or Store & Branch Block

This block generates address for the RAM and ROM memory in store, load and jump instructions. The signal F_S selects the address for particular memory (that means if F_S is high: the generated address is for accessing RAM else the generated address is of ROM address).

I. Write Back

The ALU result is written to an appropriate register in the register file. The register file is the hardware which stores all the registers. At this stage, the MUX will select the final result as shown in the architecture of the processor.

J. Data Memory

In this processor, the memory is accessed during the load and store instruction. For any other instructions, this doesn't do anything. The memory being accessed is the data memory (which is the data cache).

K. 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.

L. Crypto Purpose Instructions

Three crypto purpose instructions have been designed. They are BITP, BYTP and RXOR.

1. BITP Instruction: It is a bit-oriented instruction with four operands that is demonstrated in fig.8. A sample use of BITP is shown below:

BITP GPR1 GPR2 PR1 PR2

It is a bit permutation on lower 16 bits of GPR2 (source register) based on the pattern determined by PR1 and PR2 registers, and the result is stored in GPR1 (target register). The PR registers are divided into 8 parts containing 4 bits. Each part indicates 1 bit out of lower 16 bits in GPRs. In the BITP instruction, PR2 gives the bits of GPR2 (source register) to be copied into those bits of GPR1 (target register) designated by PR1.

For example, if the value of part 5 of PR2 and PR1 is 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.

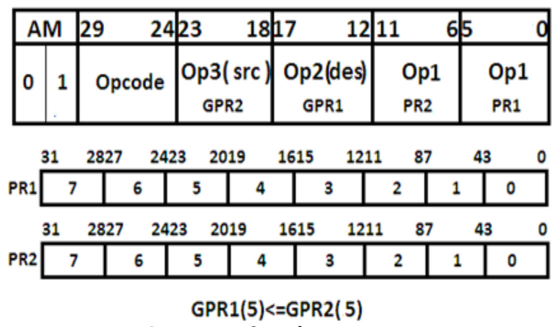

Fig.8: Functionality of BITP Instruction

2. 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

It is a byte permutation on the four bytes of GPR2 (source register) based on the pattern determined by PR1, and the result is stored in GPR1 (target register). The PR registers are divided into four parts containing 2 bits. Each part indicates 1 byte out of the 4 bytes in GPRs. In the BYTP instruction, PR1 specifies the bytes of GPR2 (source register) to be copied into relative bytes of GPR1 (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.

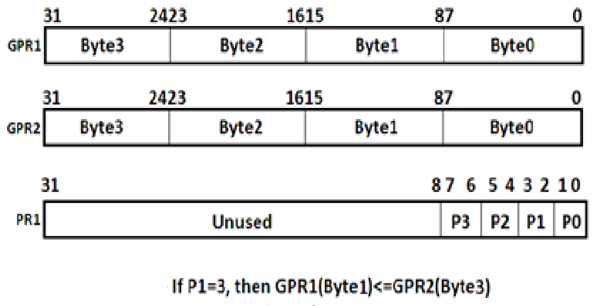

Fig.9: Functionality of BYTP Instruction

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

RXOR GPR1 PR1

As shown in fig.10, 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 Program Status Word (PSW)

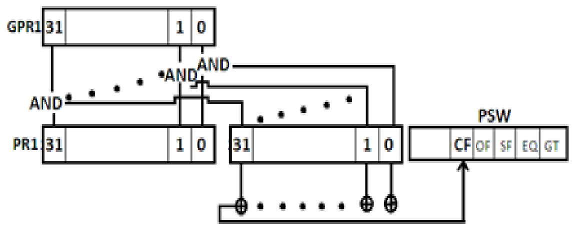

Fig.10: Functionality of RXOR Instruction

III. CIARP ASSEMBELER

An embedded system is a special-purpose computer system designed to perform few required functions. It is embedded on a device as a part of it including hardware and mechanical parts. As embedded systems usually use processors with limited computational power and few hardware resources, it is important to write efficient programs controlling the systems. In order to automate generating executable code for this processor, an assembler that converts assembly instructions written in a file into machine language instructions has been designed.

IV. SIMULATION AND SYNTHESIS RESULTS

Having been completed the design process, in order to verify the processor's performance, the processor was implemented on hardware using VHDL and Xilinx ISE synthesis tool. The target FPGA used in the implementation was Spartan3s xc3s1200E device from Xilinx. The result of the synthesis showed that the full implementation needs 5154 (59% out of 8672 slices) and its critical path limited the maximum clock frequency on 76.447MHz. The post-synthesis simulation verified the design goals of the proposed processor.

V. CONCLUSION

In this paper a novel architecture for crypto processors were presented. The proposed idea of the processor with crypto specific instruction set 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 related to stream ciphers, block ciphers and hash functions as a pipeline architecture is used in this project in contrast with the existing technologies for cryptography.

REFERENCES

[1] Computer Organization and Design: The Hardware/Software Interface, 3rd ed., Morgan Kaufmann, 2005.
[2] 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.
[3] 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.
[4] S. Balakrishnan and G.S. Sohi, "Exploiting Value Locality in Physical Register Files," In Proc. of 36th MICRO, pp. 265-276, Dec 2003.
[5] 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.
[6] The SPARC Architecture Manual Version 8, SPARC International, Prentice Hall, 1992.
[7] K. Nakano, Y. Ito, "Processor, Assembler, and Compiler Design Education Using an FPGA," IEEE International Conference on Parallel and Distributed Systems, pp. 723-728, Dec. 2008.

AUTHOR PROFILE

Bhagyasree.P, presently pursuing final semester M.Tech in Digital Systems & Computer Electronics at Malla Reddy Engineering College, Secunderabad. She is presently working as an Assistant Professor at RRS College of Engineering and Technology, Patancheru. She received her degree of Bachelor of Technology in the stream of Electronics and Communication from Maheshwara Engineering College. Her areas of interest are VLSI, Signal and Image Processing, and Communications.

C.Silpa , presently working as a Associate Professor in the department of Electronics and Communication Engineering, MREC, Secunderabad, Andhra Pradesh, India. She has 7 years of teaching experience. Her areas of interest are Communication Systems, Network Security, Image Processing, Digital Signal Processing.

Dr. M. J. C. Prasad, presently working as a Head of the department of Electronics and Communication Engineering, MREC, Secunderabad, Andhra Pradesh, India. He is having 15 years of teaching experience. His areas of interest are Communication systems, Digital Systems, Image Processing, Digital Signal Processing, Advanced DSP Systems.