Hardware Based Virus Scanning Acceleration

Rangana De Silva[1], Iranga Navaratna[2], Malitha Kumarasiri[3], Hasindu Gamaarachch[4]
Department of Computer Engineering, Faculty of Engineering, University of Peradeniya
Peradeniya, Sri Lanka
[1]ranganades@gmail.com
[2]iranganavaratna@gmail.com
[3]malithakumarasiri@gmail.com
[4]hasindu2008@gmail.com

***Abstract*-A hardware-based system to accelerate virus scanning is proposed in this study. The number of files stored in a storage device increases day by day. Hence, the number of files to be scanned also increases. This causes the virus scanning process to run more slowly.**

Virus scanning can be done using methods such as heuristic-based detection, signature-based detection, behavioral detection and cloud-based detection. In signature-based detection, a signature is generated for every file. The process of generating this signature is called hashing. Currently hashing is done using general purpose instructions, but this can be accelerated by adding new application-specific instructions to the processor. Experimental results demonstrate that the virus scanning process is accelerated by this system.

***Keywords*-Hardware based acceleration, Signature based detection, Hashing, Virus scanning, Application-specific instructions, Hotspots**

I. INTRODUCTION

A clear difference in processing speed can be experienced when using a computer with and without antivirus software. This difference is caused by the virus scanning process.

A computer virus is a type of malicious software program ("malware") that, when executed, replicates by reproducing itself (copying its own source code) or infecting other computer programs by modifying them

Computer viruses affect the system performance. Viruses access data illegally and delete data stored in a computer. These data may be important user data or system files. The illegal deletion of system files may cause a system breakdown. Antivirus software has been developed as a remedy for this.

Virus scanning is the process of searching a storage device for viruses. Antivirus software is used for this process. Currently, virus scanning is purely software based. Scanning a large set of files takes a significant amount of time. Also as the number of viruses increases rapidly, the virus scanning process would take more and more time.

There are different methods to detect a virus. Heuristics-based detection, Behavioral-detection, Signature-based detection and Cloud-based detection are some of them.

Signature-based detection is one of the oldest methods which is still used to detect computer viruses. The significance of this method is that it is highly effective towards well-known attacks. It can be performed quickly on modern systems as it reduces the amount of power required to perform these checks. This detection method consists of two main stages, calculating the hash (the signature) and checking the hash with the known virus database. A hash is a unique value for any given file. MD2, MD4, MD5, MD6, SHA-1are some of the hashing algorithms used to calculate the hash.

This project is done to accelerate the hash calculation part of the virus scanning process. MD5 was selected as the hashing algorithm.MD5 [2] is a widely used hashing algorithm which takes any file as input and produces a 128-bit signature as the output. This algorithm was designed by Ronald Rivest in 1991. It has many applications. One of the applications is virus scanning. The 128-bit signature produced by this algorithm can be used to check if a file is corrupted. MD5 takes a variable length input and breaks it into 512-bit blocks. Then, the input is padded so that its length is divisible by 512. Padding is performed by appending a single "1" bit, and then "0" bits to the message. After operating on these 512-bit chunks, MD5 produces the 128-bit output.

Normally all the calculations are done in the processor using the arithmetic logic unit (ALU). Addition, multiplication, AND operation and OR operation are some of the basic arithmetic functions of an ALU. These functions of the ALU are repeatedly used when calculating the hash value in virus scanning. The processing time can be decreased by introducing specialized hardware components to the processor and using one clock cycle to do repeatedly used operations.

The basic processor which uses general purpose instructions is known as the base processor. This processor is later modified by adding new application specific instructions. A five stage pipelined MIPS processor [3] is used in this project as the base processor. It has 32-bit instructions. This processor is designed using Verilog. Verilog is a hardware description

language, which is used to model electronic systems. This language used in this project to extend the base processor and to simulate it. Later, the new processor is tested on a Field programming gate array (FPGA).

FPGA is an integrated circuit which can be used to solve any computable problem. It contains programmable logic blocks, which can be configured to perform combinational functions. These blocks contain smaller components, including flip-flops, look-up tables, and multiplexers. Hardware description languages are used to configure these logic block. Digital signal processing, software-defined radio, application specific integrated circuits (ASIC) prototyping, cryptography, computer vision and computer hardware emulation are some of the applications of this device.

The aim of this project is to create separate hardware components and extend the base processor in order to accelerate the virus scanning process. Separate hardware components are designed specifically to decrease the processing time of the virus scanning process. These new hardware components are designed to support and accelerate the MD5 hashing algorithm. The number of clock cycles needed can be reduced using these new hardware components. This allows the hash to be calculated faster.

II. Related Work

Baojun Zhang, Jiebing Wang and Xuezeng Pan [4] have done a study on accelerating the virus scanning process. This study has been concentrated on the string matching part of virus scanning while our study is focused on accelerating the hashing part of virus scanning. In their system, the files on the client are uploaded to the server, where the server executes the virus scanning process. This requires a network connection. Our method which is standalone does not require a network connection.

Kimmo Järvinen, Matti Tommiska and Jorma SkyttäHelsinki [5] have implemented a separate hardware module for the MD5 algorithm. This has been tested on an FPGA. This module has given a very high throughput. However, this method would be costly to implement physically. This is designed specifically for MD5. If the virus scanning process starts to use a new hashing algorithm, this module would be useless. Our project is done with the intention of adding new instructions to the processor. These operations can be used for any other similar algorithm as well.

Thipakar Sabapathipillai, Sinthuja Kopalakirushnan, Dhammika Elkaduwe and Roshan Ragel [6] have done a study on accelerating virus scanning using parallel processing. This study has been done to accelerate Aho-Corasick and Boyer-Moore algorithms using GPU. It is focused on the pattern matching part of signature-based detection. Our study is done to accelerate the hash calculating process.

Changxin Li, Hongwei Wu, Shifeng Chen, Xiaochao Li and Donghui Guo [7] have done a study on accelerating the MD5-RC4 algorithm using NVIDIA GPU. The results show that the MD5-RC4 can be accelerated by 3-5 times using GPU. However, this acceleration can be done only on computers which have a GPU.

III. Methodology

A. Hash Algorithm

A hash, which is also called a digest, is a signature for a stream of data. Hashes, compile a stream of data into a small digest. It is a one-way operation. The size of the hash is fixed regardless of the size of the input. MD5 and SHA-1 are well-known hash functions that are currently in use.

MD5 was the hashing algorithm used in this study since it is a commonly used hash function which produces a unique hash value for any file. It is quick to generate compared to SHA-1 but not secure as SHA -1.

When calculating the MD5 hash value for a given file or message, the algorithm takes the input and breaks it into chucks of 512-bit blocks. The loop in the MD5 algorithm executes once for every 512-bit chunk. And within this loop is another loop which executes for 64 times

B. Identifying Hotspots

As the first part of this study, hotspots in the algorithm were identified. Hotspots are the instructions which execute multiple times in a program. The operations shown in FIGURE1 and FIGURE2 are found in the inner loop of 64. Since this loop executes 64 times for every 512-bit chunk, these operations are used for 64*N times in a single hash calculation. Therefore, theseoperations were selected as hotspots.

```
if (i < 16) {
    f = (b & c) | ((~b) & d); // hotspot 1
    g = i;
} else if (i < 32) {
    f = (d & b) | ((~d) & c); // hotspot 1
    g = (5*i + 1) % 16; // hotspot 2
} else if (i < 48) {
    f = b ^ c ^ d; // hotspot 3
    g = (3*i + 5) % 16; // hotspot 4
} else {
    f = c ^ (b | (~d)); // hotspot 5
    g = (7*i) % 16; // hotspot 6
}
```

FIGURE1: Hotspots in MD5 hash algorithm

```
b = b + LEFTROTATE((a + f + k[i] + w[g]), r[i]); // hotspot 7
a = temp;#FFFFFF
```

FIGURE2: Hotspots in MD5 hash algorithm

These hotspots use multiple clock cycles to execute, but the newly added instructions are designed to execute these operations using lesser number of clock cycles. Most of the hash algorithms use similar operations. Hence, the new application specific instructions can also be used in other hash algorithms too.

MD5 algorithm uses two arrays which contain 64 elements. Both arrays are accessed 64 times when generating a hash value. These parts can also be accelerated by implementing the arrays in hardware. However, these were not implemented in hardware as it would not be practical to add a register file containing 128 entries when the data memory has only 32 entries.

C. Selection of the Processor

Selection of this processor was made based on the architecture of it. MIPS is a simple processor, and its architecture can be easily understood. A five stage pipelined 32-bit MIPS processor was used in this study for testing as it contains less number of separate instructions.

Further, the source code of the selected processor is available online [3].

In this study, the hardware descriptive language (HDL) used to modify the base processor was Verilog [8]. Icarus Verilog was used as the HDL simulation tool.

D. Alterations to the MIPS Processor

The processor selected contains fifteen Verilog modules. These modules were modified to accommodate the new instructions. Details of the modifications are given in TABLE I.

TABLE I
MODIFIED VERILOG MODULES

Verilog module	Comments
alu.v	New ALU operations were implemented An additional input was added
alu_control.v	Function values were implemented for new ALU operations
cpu.v	An addition input was added
im.v(instruction memory)	Necessary instructions were hardcoded for testing
regm.v(register file)	Modified to support an additional instruction
regr.v(used to store data of pipelined stages)	Modified to support an additional instruction

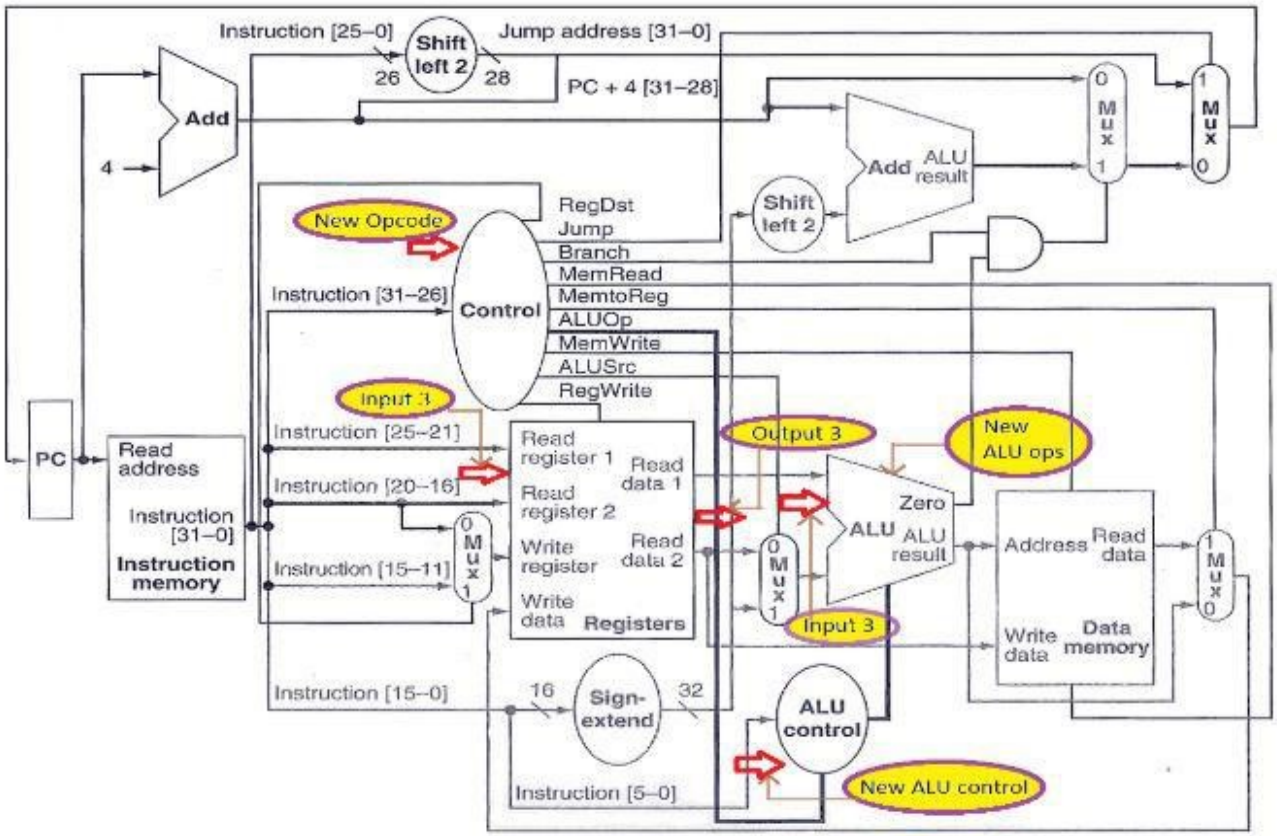

FIGURE3: Modified processor

The changes made in the base processor to accelerate the hashing process are highlighted in FIGURE3.

The ALU of the base processor contains seven basic operations. After examining the MD5 algorithm, four new operations were designed and added to the processor. These operations were designed specifically to accelerate the MD5 algorithm. Changes were done to control blocks and the ALU as shown in FIGURE3.

The ALU was modified to take 3 inputs and the register file was amended to read 3 registers at a time as shown in FIGURE3.

The new instructions designed are given below in TABLE II.

TABLE II
COMPARISION OF THE OLD AND NEW INSTRUCTIONS

Hotspots (New operations)	No. of instructions used in the old processor	No. of instructions used in the new processor	No. of clock cycles saved
a^ (b | (~c))	3	1	2
a ^ b ^ c	2	1	1
(a & b) | ((~a) & c)	4	1	3

IV. TESTING AND VERIFICATION

Icarus Verilog was used for the software verification of new instructions. In order to test these instructions, assembly code of the hotspots were written to an assembly (.asm) file and relevant machine code (.hex) files were generated using a MIPS cross compiler. Then those .hex files were fed to the processor. Extracts of the old and new hex files for the double xor function (a^b^c) are shown in TABLE III. As shown here the last two instructions in the old version has been fused to one instruction.

TABLE III
OLD AND NEW HEX FILES(MACHINE CODE)

Olddoublexor.hex	Newdoublexor.hex
20030003 20080005 20090007 00685026 01495826	20030003 20080005 20090007 00685A43

Both old and new .hex files were simulated using Iverilog.

The old .hex file was executed in the base processor and the new .hex was executed in the extended processor. This simulation process is shown in FIGURE4. The inputs are highlighted in red and outputs are highlighted in green in FIGURE4.

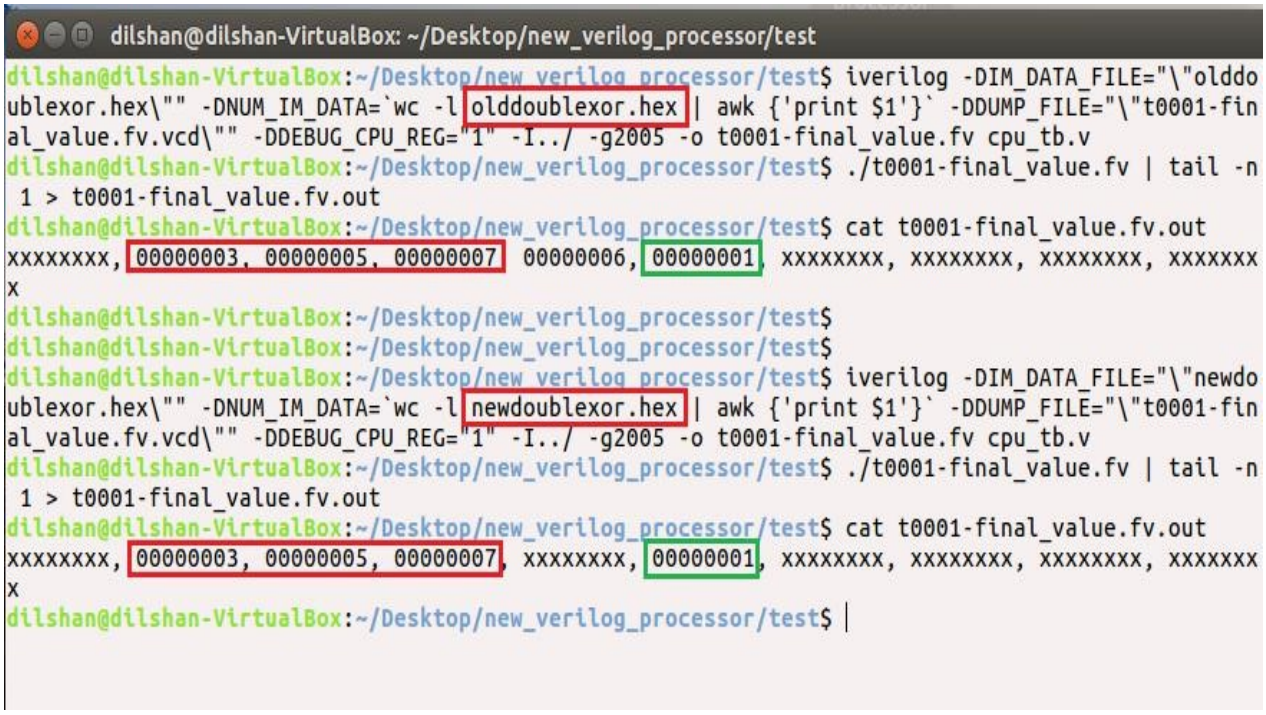

FIGURE4: Results comparison of hotspot double XOR in Iverilog

Since both hex files generated the same output, it was verified that the new instructions generate true results. Other newly added instructions were also tested in the same manner.

Hardware verification of the new processor was done on an FPGA board. The FPGA device used is a Cyclone IV EP4CE115F29 belonging to the Cyclone IV E family. This device has 432 M9K memory blocks. Furthermore, has 128MB of SDRAM, 2MB of SRAM, 8MB flash with 8-bit mode and 32Kb EEPROM.

Both the new application specific instructions and the blocks of old instructions were tested on this FPGA for verification.

Altera Quartus [9] is a programmable logic device design software. HDL designs can be analyzed and synthesized using this software. An HDL design can be compiled and features namely timing analysis and RTL simulation can be used to configure the design with the target device (FPGA).

The new instructions were simulated on the FPGA board using Altera Quartus software for hardware verification. New instructions generated the same output as the old instructions but using lesser number of clock cycles.

V. RESULTS

The extended processor and the base processor were tested with different hotspot instructions, and timing analysis was done separately for every hotspot. These time values of hotspots were benchmarked in both processors to test the performance.

A. Timing Analysis

Timing analysis feature in Altera Quartus was used in finding the maximum allowable frequency for Verilog instructions. Maximum allowable frequency (Fmax) depends on the instruction which takes the most time. Hence, for different hotspots the Fmax value may change. Fmax also depends on slack. Slack depends on arrival time and required arrival time of data.

Arrival time and required time is very useful when verifying the clock requirement for a setup between sequential elements in the design. Arrival time represents the time at which data arrives at the input of the receiving sequential element. Required time represents when data is required to be present at the end latch. FIGURE5 illustrates the setup check.

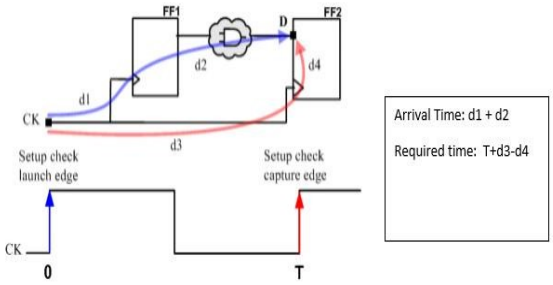

FIGURE5: Arrival time and required time for setup check

Slack is simply the difference between the required time and the arrival time.

Slack = required time – arrival time

If the slack is negative, the path is violating the setup relationship between the two relationship elements. Hence, it is necessary to maintain a positive slack.

The slack analysis was done for both processors under 10ns clock period. A positive slack was obtained on all the occasions. Hence, the data path was not violated in the setup relationship mentioned earlier.

B. Performance Comparison of Hotspots in Old and New Processors

As the slack values are positive, Fmax can be calculated using Quartus timing analyzer without any violations. The verification was done using a frequency of 100MHz. Since the Fmax values of all the instructions were greater than 100MHz, it was concluded the extended processor can be used with a frequency of 100MHz. Table IV shows the detailed results sheet of the comparison of hotspots in both old and extended processor designs.

TABLE IV
PERFORMANCE COMPARISON OF PROCESSORS

Hotspot	Clock cycles reduced	Fmax old(MHz)	Fmax new(MHz)	Time saved (ns)
a^(b|(~c))	2	102.26	104.64	20
a^b^c	1	122.01	105.88	10
(a&b)|((~a)&c)	3	106.77	111.02	30

VI. CONCLUSION

Virus scanning is a vital process in a computer. This process reduces the performance of a computer. A method to accelerate the virus scanning process was presented in this study. The method was to implement separate hardware components in the processor to

speed up the hash calculating process. In this study, a 32-bit MIPS processor was extended by adding new application specific instructions. Separate hardware components were added to accommodate these new instructions. After adding these new components, the extended processor was tested and verified using an FPGA.

When testing this new extended processor, it was seen that the hash calculation could be accelerated using this method.

An average of 4 clock cycles can be saved for each time the inner loop of the MD5 algorithm is executed in the extended processor. Therefore, if there are N number of 512-bit chunks to operate on, 64*4*N number of clock cycles can be gained for a single hash calculation using the extended processor.

There are few more hotspots which were not implemented in this project. The processor can be further extended and the hash calculation can be done faster if those hotspots are also implemented. If this method is combined with an accelerated string matching system, the virus scanning process can be accelerated significantly.

REFERENCES

[1] William, Stallings. *Computer Security: Principles And Practice*. Pearson Education India, 2008.

[2] Rivest, Ronald. "The MD5 message-digest algorithm." (1992).

[3] MIPS processor [Online] Available: https://github.com/jmahler/mips-cpu

[4] Zhang, Baojun, Jiebing Wang, and Xuezeng Pan. "Virus Scan System Based on Hardware-Acceleration." *Computer and Computational Sciences, 2007. IMSCCS 2007. Second International Multi-Symposiums on*. IEEE, 2007.

[5] Jarvinen, Kimmo, Matti Tommiska, and Jorma Skytta. "Hardware implementation analysis of the MD5 hash algorithm." *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*. IEEE, 2005.

[6] Sabapathipillai, Thipakar, et al. "Accelerating Virus Scanning With GPU."

[7] Khan, Shadab Ahmad. "FPGA Implementation of MD5 Algorithm for Password Storage."

[8] The Verilog website [Online] Available: http://www.verilog.com/

[9] Altera Quartus II [Online] Available: https://www.element14.com/community/docs/DOC-40098/l/altera-introduction-to-the-quartus-ii-software