Volume: 5, Issue: 1
Page: 69-77
2023

Journal of Scientific Reports

Journal homepage: ijsab.com/jsr

IJSAB International

Design Pipeline SHA-3 MIPS Processor using FPGA

May S. Al-Rabi

Abstract:
The purpose of this study is to implement and evaluate the security of the Keccak hash function in a Microprocessor without Interlocked Pipelines (MIPS) processor using Field Programmable Gate Arrays (FPGA). The design methodology involves selecting a set of instructions that are required to run the SHA-3 Keccak algorithm on the MIPS processor. The findings of this study demonstrate that the Keccak hash function can be efficiently implemented on a MIPS processor using FPGA technology. The research limitations include the fact that the study only focuses on the implementation of the Keccak hash function on a MIPS processor using FPGA, and does not consider other platforms or technologies. The practical implications of this study are that it provides a cost-effective solution for implementing secure hash functions on embedded systems that require low power consumption and high performance. The originality and value of this study lie in its contribution to the field of cryptography by demonstrating the feasibility of implementing secure hash functions on low-cost processors using FPGA technology.

JSR
Accepted 20 October 2023
Published 27 October 2023
DOI: 10.58970/JSR.1027

ISSN: 2708-7085

Keywords: *VHDL, MIPS Processor, SHA-3, Keccak, FPGA.*

About Author (s)

May S. Al-Rabi, Computer science department, University of technology, Iraq.

Introduction

The cryptographic hash function known as Keccak is the topic of this paper. Hash functions in cryptography are essential to today's modern cryptography and find usage in various practical applications. These applications include the verification of message integrity, the authentication of messages, and the safe storage of passwords. In cryptographic protocols, hash functions are utilized to protect the integrity of a message or to provide authentication by computing a short identifier for the message. This identity is referred to as a hash value, and inside the protocol, it serves as a representation. A cryptographic hash function can take an input of any possible length as long as it is finite, and it can produce an output of a size that has been set in advance. Due to the fact that the input domain is typically much bigger than the output domain, the functions in question have a mapping that is many-to-one in nature. As a direct result of this, it is impossible to prohibit the occurrence of two different messages with the same outcome from happening at the same time. As a consequence of this, for a hash function to be considered safe, it should be computationally difficult to find collisions between different hash values. At the moment, the hash algorithms that are employed the most commonly are SHA-1, SHA-256, and SHA-512, all of which have been given the stamp of approval by NIST. They are designed in accordance with the MD4 and MD5 design principles, and they are a component of many different standards. Over the course of the previous few years, considerable advancements in cryptanalysis have been accomplished, and vulnerabilities in a variety of functions have been discovered. Practical collisions have been shown to be possible with MD4 (Wang, et al., 2005a), MD5 (Wang, et al., 2005b), and SHA-0 (Wang, et al., 2005c). These collisions have been demonstrated. The security bound is significantly lower than was anticipated (Wang, 2005d), despite the fact that the amount of computational work needed to build collisions for SHA-1 is still outside the realm of what is practically possible. It is theoretically possible to initiate attacks on reduced rounds, and actual instances of this happening in practice have been demonstrated (De Cannière, et al., 2007). As a result of this, there is a substantial amount of interest in the development of new secure hash algorithms. In this article, the design of the hash function Keccak in the MIPS processor architecture is discussed. Keccak was the submission in the SHA-3 competition that the National Institute of Standards and Technology (NIST) considered being the most successful overall. The acronym FPGA stands for field programmable gate arrays. It refers to a type of digital integrated circuit (IC) that has programmable customizable pieces of logic, and programmable interconnects between these blocks. Design engineers can configure (program) such gadgets to carry out a dizzying array of responsibilities. Many FPGAs can be reprogrammed on-site, where it is possible to carry out the programming process (De Cannière, et al., 2007). Because of the reprogrammable nature of FPGAs, they are ideally suited for usage in educational settings. This quality enables students to do as many iterations as necessary to rectify and improve the performance of their processor design.

All data operations in a RISC architecture processor are performed on register data and often apply to the whole register's worth of data. Load and store operations are the only two operations that have any effect on memory. They move information between storage locations, such as memory and a register. This architecture is also known as load-store architecture since it allows load and store operations that load or store less than a complete 32-bit register (for example, a byte, which is 16 bits) (Hennessy & Patterson, 2012a). The number of different formats for instructions is quite low, and the majority of instructions are only available in a single size. Putting it into practice is thus made easier as a consequence of this. MIPS is an example of a general-purpose RISC architecture. MIPS stands for "microprocessor without interlocked pipeline steps," which describes the design of the MIPS microprocessor. A RISC processor having the capability of carrying out full instruction in a single cycle is referred to as

a single-cycle MIPS. The length of time required for each cycle is decided by the instruction that executes the most slowly. Pipelining is a technique for the construction of software in which the execution of a number of different instructions is performed in parallel with one another (Sweetman, 2007). As a direct consequence of this, pipelining may be achieved by partitioning the MIPS for a single cycle into a number of different parts. The term "stage" is used to refer to each individual component. Because of this, a pipelined MIPS with five stages allows for the execution of five instructions concurrently, with one instruction carried out in each stage. Because each stage needs just one cycle of the clock, the amount of speedup that may be accomplished by pipelining is generally equal to the number of stages in the pipeline (De Cannière, et al., 2007b). This is the situation when all conditions are perfect.

MIPS Architecture

An instruction set and some familiarity with registers are the two primary components that make up a CPU's architecture (De Cannière, et al., 2007). The architecture of a 32-bit MIPS processor is discussed in this section of the article. Instruction Set Architecture (ISA) is a term that refers to the actual instruction set that is accessible to the programmer. The ISA can be thought of as the dividing line between software and hardware (Hennessy & Patterson, 2012a). The following are the seven dimensions that make up the MIPS ISA: In terms of the Instruction Set Architectural (ISA), the MIPS processor is classified as a member of the General-Purpose Register (GPR) architecture family. In this particular architecture, the operands may take the form of registers or memory locations at any given time. Memory addressing: The MIPS memory model is byte addressed rather than word addressable since each data byte has its own address. This is in contrast to other memory models, which are word addressable. Other memory types, on the other hand, are word addressable. This is in contrast to that. The 8-bit (byte), 16-bit (half word), and 32-bit (word) operand sizes are all supported by MIPS (Hennessy & Patterson, 2012b). MIPS also supports the following sorts of operands. Operands have the potential to be constant values that are either stored in the instruction itself, in the registers, or in the memory (Hennessy & Patterson, 2012a). The phrase "operations" covers a wide range of topics, including data transport, arithmetic and logic, control, and floating-point operations (Hennessy & Patterson, 2012b). Because it is a simple architecture with an instruction set that can be pipelined with relative ease, MIPS is a good example of a RISC architecture that was used in 2011, and it is a representative example. Instructions for controlling the flow of control are supported by almost every instruction set architecture (ISA) (Page, 2009). These include conditional branches, unconditional jumps, calls to procedures, and returns from those procedures. PC-relative addressing is utilized by each branch, and the branch address is derived from an address field that is added to the Program Counter (PC). Encoding of the Instruction Set Architecture (ISA) Each and every MIPS instruction is 32 bits long, which makes instruction decoding a great deal less complicated. In the MIPS architecture, which makes use of 32-bit addresses, the addressing modes not only identify registers and constant operands but also represent the address of a memory object. In the MIPS architecture, which makes use of 32-bit addresses, the addressing modes not only identify registers and constant operands but also represent the address of a memory object.

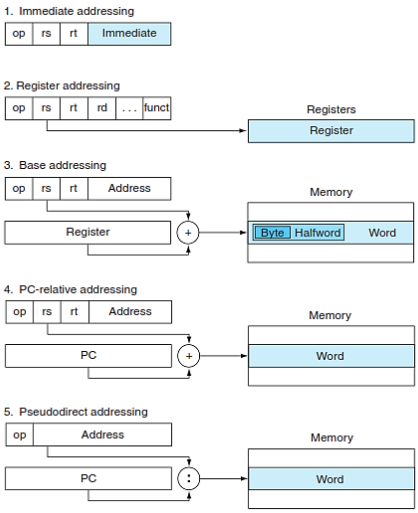

Figure 1: Addressing modes of MIPS

Each instruction execution in a Pipelined MIPS processor is split into five stages, making it possible for the processor to carry out up to five different instructions during a single clock cycle (Hennessy & Patterson, 2012b). Pipelined MIPS processors are classified as 5-stage pipelined processors. Therefore, the single-cycle datapath shown in Figure 2 must be divided into five parts, each of which is given a name that represents a specific phase in the execution of an instruction. The various phases include instruction fetches (IF), decoding instructions (ID), executing instructions (EXE), accessing memory (MEM), and writing results back (WR) (WB). The utilization of pipeline registers is essential to the successful implementation of single-cycle pipelined MIPS. These registers are used to partition the data path into the five sections IF, ID, EXE, MEM, and WB, and they are also responsible for storing the values that are required by instruction as it moves through the subsequent stages. Pipelined MIPS is the key to successfully implementing single-cycle pipelined MIPS. The stages determine the labels of the pipelined registers in MIPS that they separate (Valli, et al., 2012). Figure 2 shows the 5-stages, pipelined MIPS processor with the following pipelined stages:

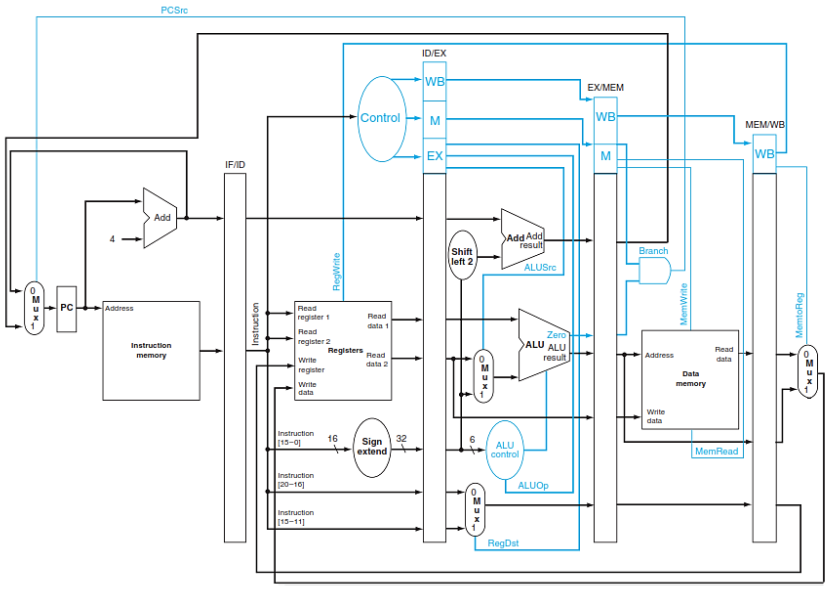

Figure 2: The simple pipelined datapath with the control unit [8].

1) Instruction fetches (IF): the content of the PC is read from memory to retrieve the instruction, and then the value of the PC is increased by 4 to point to the next instruction. The instruction, as well as the now-incremented PC, are both saved in the register for the IF/ID pipeline.
2) Instruction decode (ID): this step involves decoding the instruction that is stored in the IF/ID register, after which the necessary operands are recovered and then delivered to the ID/EXE pipeline register.
3) Execution (EX): either carries out the necessary operation or calculates an address by making use of the operands that have been taken from the ID/EXE pipeline register. The result is saved in the EXE/MEM pipeline register when it has been processed.
4) Memory access (MEM): This stage is only utilized by lw or sw instructions that access the data memory for read or write using the address from the EXE/MEM pipeline register. During this stage, the data memory can be read from or written to. The MEM/WB pipeline register is loaded with data when the lw instruction is executed.
5) Write back, or WB, occurs when the results from the MEM/WB pipeline register are written into the register file (Hennessy & Patterson, 2012b) (Kaur & Gulati, 2013).
In a pipeline processor, the stage that operates at the slowest speed determines the clock rate, also known as the clock cycle period. The control signals that are valid for the single-cycle scheme are also valid for the pipelined scheme, as shown in tables 2-4 and 2-6. However, the pipelined scheme inherently requires sequencing, and the control signals that are generated in the decode stage must go with the instruction throughout the pipeline and waste up until the last stage, as shown in figure 3 (Robio, 2004).

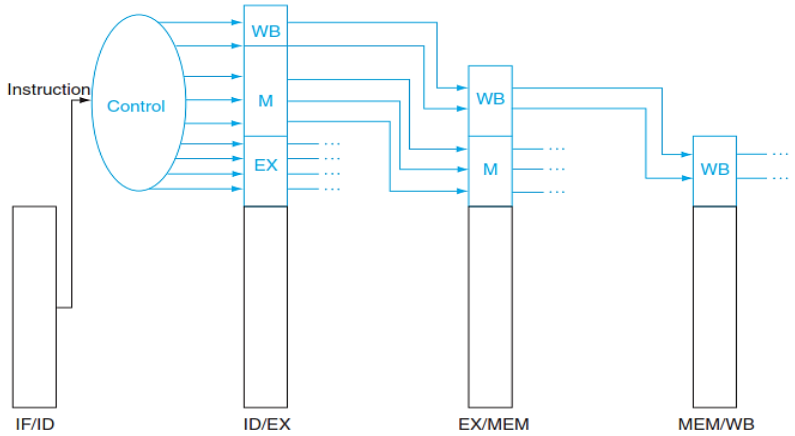

Figure 3: The control lines for the final three stages.

Hash Architecture

A family of hash functions known as Keccak is based on a construction known as the sponge, which allows the state to have a size b. (25; 50; 100; 200; 400; 800; 1600). It makes use of the Keccak-f permutation, as well as the padding scheme. Keccak[r; c; nr] is the notation for a particular instance of the Keccak cryptographic algorithm, where r denotes the rate, c is the capacity, and nr indicates the number of rounds. The Keccak algorithm uses the permutation f, which acts on a three-dimensional state and has elements in the F2 field (see Figure 4). The parameters for this condition are 5 by 5 by w, where w is the width (1; 2; 4; 8; 16; 32; 64). Using this method, each lane can be represented as a w-bit word (Karmani, et al., 2020) (Stinson & Paterson, 2017). Sponge construction is a method of operation that builds a function that accepts input of arbitrary size and output of arbitrary size. The input and output sizes can be chosen at will. It is based on a permutation with a fixed size, f, has a padding rule, and takes two parameters: the rate r and the capacity c. The sponge creation is done iteratively, and the

internal state S has the dimensions b = r + c. Initially, make use of the padding rule to partition the input m into blocks of the sizes r, starting with M0 and going up to Mn. The initial state will be set to S = (0.... 0), and the input will be processed through the phases titled Absorb and Squeeze in the following order. Figure 4 provides a high-level overview of the process that will be followed. When a random permutation is used, the sponge construction can be found (Pachghare, 2019) (Martino & Cilardo, 2020).

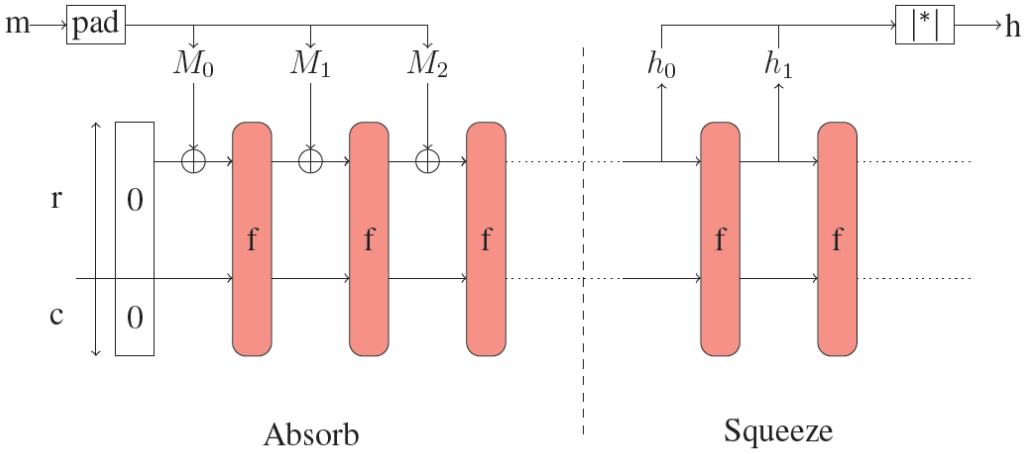

Figure 4: The sponge construction accepts data of arbitrary length as input and computes output that is also arbitrary in length. The input is processed iteratively using a fix-sized invertible permutation denoted by f.

Currently, four versions of the Keccak function family have been approved for the SHA-3 hash function standard: SHA3-224, SHA3-256, SHA3-384 and SHA3-512.in this paper, designs MIPS processors have only instruction that needs to design SHA3-256. The hardware design of a 32-bit Hash MIPS processor by using VHDL, all combinational and sequential elements in the MIPS processor are designed separately as components in VHDL by using Xilinx ISE Design Suite 14.7 software. Then these components are connected to produce the processor. The hardware behaviour of components is described by using VHDL code.

Results

This paper shows the behavioural simulation results which have been gotten from the Xilinx ISim simulator after executing the test programs for each VHDL processor design. Figure 5 through 7 show screenshots of the simulation waveforms. In all of these figures, the first two rows depict the global clock and reset signal. The following rows are executed when the instruction is fetched from the instruction memory. These signals are:

The pc [31:0] value: is used to index the instruction memory. The pcnext [31:0] value: is fed as input to the PC register. The instr [31:0] value: is an instruction indexed out of the instruction memory. The dataadr value: is used to index the data memory the srca [31:0] and srcb [31:0] values: that fed as inputs into the ALU unit, which operates in the execute stage. The aluout value: is used to take the result of ALU operations. The memwrite value: is used to control write data into data memory. The writedata value: is the data which write in data memory. The mem value: it is the Hash Message digest. From the implementation of hash Processor in ISE 14.7 by using VHDL language and testing the execution of all steps for SHA-3 processor is held for a plaintext such as "welcom" to produce a fixed 256-bit hash code" 3f0d883a 14b7a84b 2ab42f75 ce9d695a 42f66a14 18641e16 ef6fed59 82acd8ca", which is completed at t=1030 ns.The testing figure of the SHA-1 160-bits hash function is shown in Fig. 4.1.

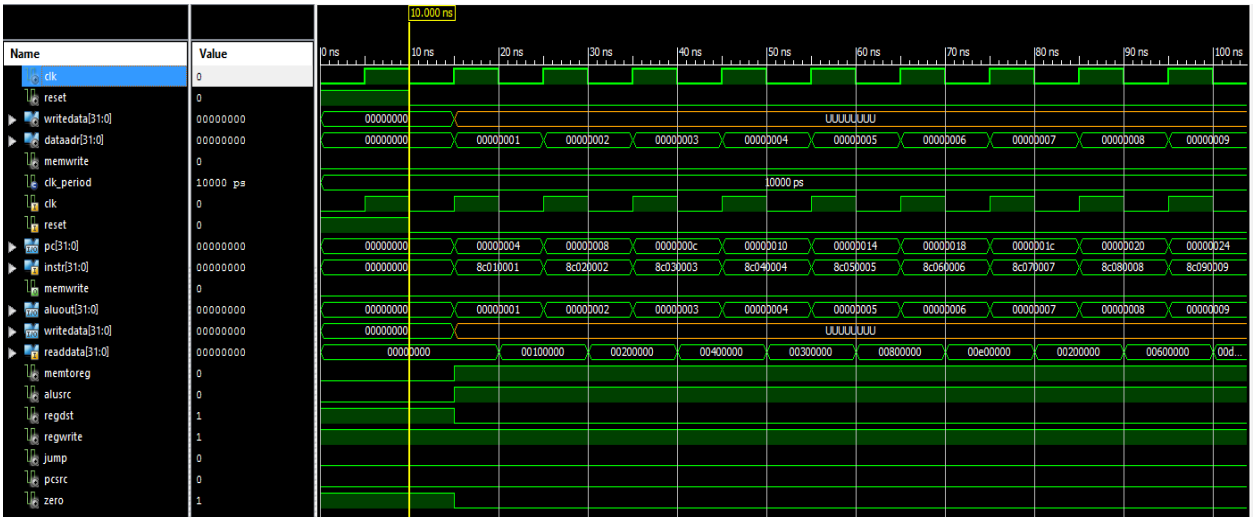

Figure 5: SHA-3 is completed at t=1030 ns.

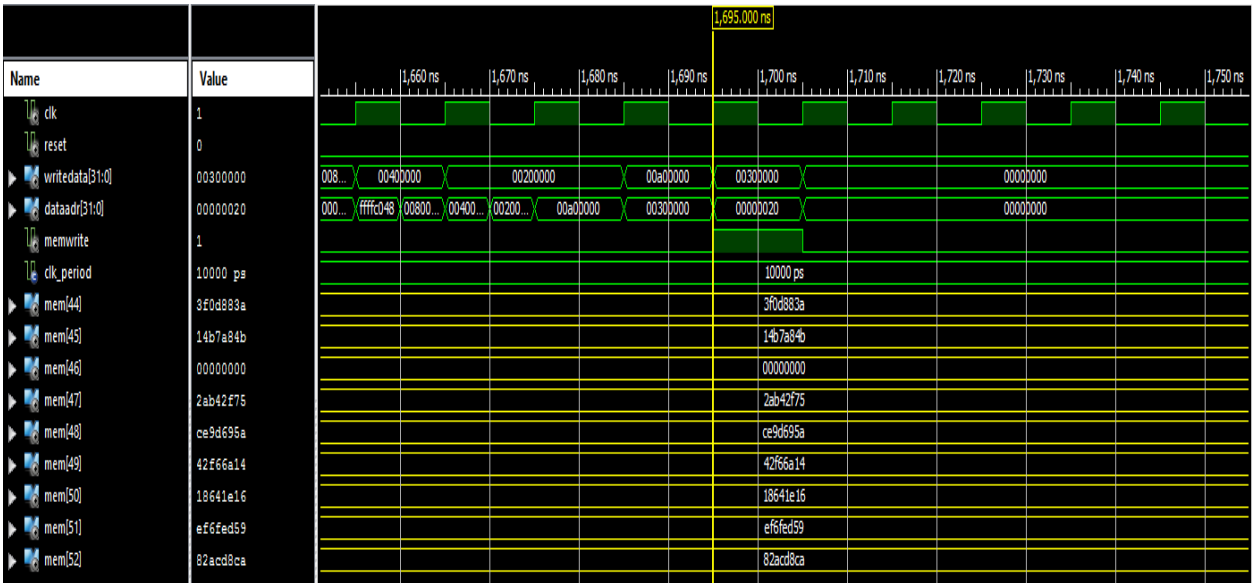

Figure 6. Data memory of the input "welcom".

Device	
Family	Spartan6
Part	xc6slx45t
Package	fgg484
Temp Grade	C-Grade
Process	Typical
Speed Grade	-3

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0
Heat Sink	None
Custom TSA (C/W)	NA

Characterization	
Production	v1.3,2011-05-04

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	1	--	--
Logic	0.000	4353	27288	16
Signals	0.000	4351	--	--
DSPs	0.000	4	58	7
IOs	0.000	67	296	23
Leakage	0.036			
Total	0.037			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	19.1	84.3	25.7

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.200	0.016	0.000	0.015
Vccaux	2.500	0.005	0.000	0.005
Vcco25	2.500	0.002	0.000	0.002

	Total	Dynamic	Quiescent
Supply Power (W)	0.037	0.000	0.036

Figure 7. Power Consumption for the SHA-3

Table 1 shows the number of clocks, throughput, power and execution time of SHA-3. Additionally, table 2 summarizes the FPGA utilization used in SHA-3 Pipeline MIPS Processors.

Table 1 Throughput and consumed power for different types of SHAs.

Hash	Time (ns)	No. of clocks	Throughput (Mbps)	Power (Watt)
SHA-3	1030	103	250	0.037

Table 2: FPGA resource utilization.

Logic Utilization	32-bit MIPS Processor
Number of Slice Flip Flops	13
Number of 4 input LUTs	3568
Number of occupied Slices	1847
Number of bonded IOBs	131

Conclusions

Hash function implementations on hardware is best than implementations on software since software implementations don't satisfy the power, throughput, speed, and security requirements of the complex modern communication systems which used today. In this paper, a hash processor for performing SHA-3 calculations are implemented, specified and analyzed using the hardware description language VHDL. The MIPS processor is chosen because it has simple instruction sets with simple decoding, easily understandable architecture and rich documentations. This design supports only instructions needed for this type of hash. It is easy to design ALU; it consists of gated and some multiplexers. It is necessary in this design to select the appropriate control signals to ensure that the instructions are going in the correct data path.

References

De Cannière, C., Mendel, F., & Rechberger, C. (2007). Collisions for 70-Step SHA-1: On the Full Cost of Collision Search. In Selected Areas in Cryptography (pp. 56-73). Springer.

Hennessy, J. L., & Patterson, D. A. (2012). Computer Architecture: A Quantitative Approach (5th ed.). Morgan Kaufmann.

Hennessy, J. L., & Patterson, D. A. (2012). Computer Organization and Design: The Hardware/Software Interface (4th ed.). Morgan Kaufmann.

Kaur, H., & Gulati, N. (2013). Pipelined MIPS with Improved Datapath. International Journal of Engineering Research and Applications (IJERA), 3(1), 762-765.

Karmani, M., Benhadjyoussef, N., Hamdi, B., & Machhout, M. (2020). A Hardware-Software Codesign Case Study: The SHA3-512 algorithm Implementation on the LEON3 Processor. In 5th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP) (pp. 1-6). IEEE.

Martino, R., & Cilardo, A. (2020). SHA-2 Acceleration Meeting the Needs of Emerging Applications: A Comparative Survey. IEEE Access, 8, 28415-28436

Page, D. (2009). A Practical Introduction to Computer Architecture. Springer-Verlag.

Pachghare, V. (2019). Cryptography and Information Security. PHI Learning Pvt. Ltd.

Robio, V. (2004). A FPGA Implementation of A MIPS RISC Processor for Computer Architecture Education (Master's thesis). New Mexico State University.

Stinson, D. R., & Paterson, M. B. (2017). Cryptography: Theory and Practice. CRC Press.

Sweetman, D. (2007). See MIPS Run (2nd ed.). Morgan Kaufmann.

Valli, B., et al. (2012). FPGA Implementation and Functional Verification of a Pipelined MIPS Processor. International Journal of Computational Engineering Research, 2(5), 1559-1561.

Wang, X., Chen, H., Lai, X., Feng, D., & Yu, X. (2005). Cryptanalysis of the Hash Functions MD4 and RIPEMD. In EUROCRYPT (pp. 1-18). Springer.
Wang, X., Yin, Y. L., & Yu, H. (2005). How to Break MD5 and Other Hash Functions. In EUROCRYPT (pp. 19-35). Springer.
Wang, X., Yu, H., & Yin, Y. L. (2005). Efficient Collision Search Attacks on SHA-0. In CRYPTO (pp. 1-16). Springer.
Wang, X., Yin, Y. L., & Yu, H. (2005). Finding Collisions in the Full SHA-1. EUROCRYPT, 17-36.

Cite this article:

May S. Al-Rabi (2023). Design Pipeline SHA-3 MIPS Processor using FPGA. *Journal of Scientific Reports, 5*(1), 69-77. doi: https://doi.org/10.58970/JSR.1027

Retrieved from http://ijsab.com/wp-content/uploads/1027.pdf

Published by

IJSAB International