DESIGN AND ANALYSIS OF APPLICATION-SPECIFIC INTRUCTION SET PROCESSOR FOR HUFFMAN CODING

Nona R Nadibella[1], Nyoman Bogi Aditya Karna[2], Dong Seong Kim[3]
[1,2] Telkom University, Bandung
[3]Kumoh National Institute Technology
nandbella@student.telkomuniversity.ac.id[1], **aditya@telkomuniversity.ac.id[2]**

Abstract

Microprocessors are much important part of computers and IoT. In technology needs, the microprocessor is a tool to help the performance in the system on that technology. Technology requires a microprocessor because the microprocessor has components to produce output according to the commands given to the microprocessor. In the previous microprocessor design, the microprocessor has design to run a computer or technological device that requires a large memory capacity to store data and commands on the microprocessor.

Along with the excessive of using are memory and commands, the microprocessor suffers from a decrease in performance which causes the microprocessor overkill. Overkill is a problem when the microprocessor experience a performance slow down due to the large number of commands and memory that the microprocessor receives. Overkill on the microprocessor also occurs because there are too many bits, so the microprocessor is no longer relevant in improving its performance. To overcome this solution, this research design a microprocessor with an instruction set processor (ASIP) using Huffman Coding so the microprocessor can work according to the instruction ordered.

The software to designing this microprocessor uses Altera Quartus with Verilog HDL as programming language. This simulation to determine the measurement of microprocessor performance and determine can accommodate the instructions of Huffman Coding. The measurement results in Analysis and Synthesis in Altera Quartus, microprocessor design has a spare element capacity of 6272 elements comparison to the intel 8088 microprocessor. From the measurement result, the design of this microprocessor can accommodate the inctructions given from Huffman Coding that the resulting bit do not exceed the required capacity. This thesis is continuation of Nimas Sekar Fatihah Thesis's is discusses Huffman Coding.

Keyword : Microprocessor, Altera Quartus, Verilog HDL, ASIP, Huffman Coding.

1. Introduction

A microprocessor is a computer electronic processing center unit (CPU) made of mini transistors and other circuits on a semiconductor integrated circuit. However, a generic microprocessor originally designed for general purpose application that uses many machine instructions, which in return causes an inefficient power consumption [1]. One aspect that is not used maximally is the instruction set, where not all machine instruction provided by the generic microprocessor are needed to run the algorithm. Therefore need ASIP (Application-Specific Instruction set Processor) which is designed to run the algorithm and its variants.

In designing this microprocessor, it will produce several components on the microprocessor such as Control Unit (CU), Memory and Datapath. In designing this microprocessor, using the Verilog and Quartus programming language methods to display the design output. Verilog is a hardware description language (HDL). It means, by using HDL can describe any digital hardware at any level. This level describes a system by concurrent algorithms (behavioral).

Every algorithm is sequential, which means it consists of a set of instructions that are executed one by one. In designing this microprocessor, using Altera Quartus II as a software to view the output of Verilog, analyze the performance of microprocessor and view the block diagrams. In quartus, there are several features to help user see the output of VHDL coding. These includes Design Entity, Sythetsis, Functional Simulation, Fitting, Timing Analysis, Timing Simulation and Programming and Configuration.

1.2 Problem Formulation

Overkill of microprocessor causes by the use of a simple algorithm on a generic microprocessor cannot provide maximum performance because not all microprocessor capabilities are used by the algorithm. One aspect that is not used maximally is the instruction set, where not all machine instructions provided by the generic microprocessor are needed to run the algorithm.

1.3 Research Purpose

Design ASIP for Huffman Coding and measure its performance.

1.4 Research Scope

Creating a microprocessor with a small number of bits with more velocity.

1.5 Writing Systematic

This undergraduate thesis is written in the following order.

a. Chapter 1 INTRODUCTION
This chapter contains background, scope of the study, research objectives, etc.

b. Chapter 2 BASIC CONCEPTS
This chapter contains an explanation of the basic theory, website, paper and tools.

c. Chapter 3 THE PROPOSED DESIGN
This chapter contains the design block diagram, design microprocessor, and the methods. methods.

d. Chapter 4 RESULT AND ANALYSIS
This chapter contains an analysis of the results obtained from Verilog HDL coding, instructions of Huffman Coding and Analysis and Sythesis from Altera Quartus as performance the microprocessor

e. Chapter 5 CONCLUSION AND SUGGESTIONS
This chapter concludes this thesis and suggestions of the undergraduate thesis.

2. Basic Concepts

2.1 Microprocessor

A microprocessor is a computer processor where the data processing logic and control are included on a single integrated circuit or a small number of integrated circuits. Microprocessor contain both combinational logic and sequential logic. Microprocessors operate on numbers and symbols represented in the binary number system. A microprocessor accepts binary data as input, processes the data, and then provides output based on the instructions stored in the memory. The data is processed using the microprocessors ALU, Control Unit, and register array.

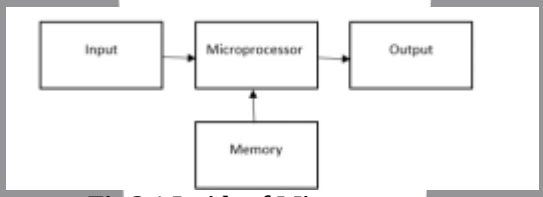


Fig 2.1 Inside of Microprocessor

The implementation of instruction after they are fetched from memory into instruction register is done by control unit. As the same suggests, control unit controls all the input and output signals, and the steps to be carried out for implementation of any program loaded in memory [2]. The ALU is used to carry out arithmetic operations such as Addition Substraction and Logical operations such as AND, OR, NOT, SRL (shift Right Logical) and SLL (Shift Left Logical) [3].

In a microprocessor, after the data is calculated and decode in the ALU, then the data will be forwarded to the data register of register array. A register is one of a small set of data holding places that are part of the computer processor. Registers have several types, one of them is the instruction register. Instruction register is a register which holds the current instruction being executed. It provides opcode to the CU and the ALU [3].

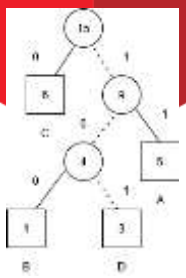
In working on this thesis, it is necessary to use ASIP in the microprocessor because want to create an Autonomous CCTV using images as input which must be transmitted to the server. The purpose of using ASIP is to reduce the size of the image. So that image processing capabilities are needed in the microprocessor. Creating an Autonomous CCTV requires power consumption to compress the image size, this power consumption is in the battery. This will use a lot of transistors on the microprocessor.

2.2 Huffman Coding

Huffman coding is one of the lossless compression techniques. The idea is to assign variable-length codes to input characters, lengths, of the assigned codes are beased on the frequencies of corresponding characters. The most frequent character gets the smallest code and the least frequent character gets the large code. Huffman Coding has higher compression efficiency than

any other methods. The method is shown to be extremely efficient in memory requirements and fast in searching for the symbol [4]. There are 3 core flows in how Huffman Coding works. First, create tree using the frequencies of the characters, then generates code for each characters. Last, decoding is done using the same tree. The steps of Huffman Coding are :

1. Calculate the frequency of each characters in the string.
2. Sort the characters in increasing order of the frequency. Sorted in a priority queue (Q). The characters sorted according to the frequency.
3. Make each unique characters as a leaf node.
4. Create an empty node (Z).
5. Remove these two minimum frequencies from (Q) and add the sum into the list of frequencies.
6. Insert (Z) nodes into tree.
7. Repeat steps 3 to 5 for all characters.
8. For each non-leaf node, assign 0 to the leaf edge and 1 to the right edge.
9.



These instructions were used in Nimas Sekar Fatihah Thesis's to calculate the frequency possessed by each instruction.

Instructions Name	Description	Frequency
ADDI	Add Immediate	9ADDIs
SW	Store Immediate	7SWs
BEQZ	Branch Equal Zero	5BEQZs
BNEZ	Branch Not Equal Zero	4BNEZs
LW	Load Word	3LWs
SUBI	Subtract Immediate	3SUBIs
SGT	Set Greater Than	3SGTs
ADD	Add	2ADDs
SEQ	Set Equal	1SEQs

Table 2.2 List of Instructions used in Huffman Coding Method

Based on how Huffman Coding, the frequency on each value needs to be sorted from the largest to the lowest or vice versa to make it easier to construct Huffman tree. If the value reaches the greatest point, it starts to swap until the last value, so all the frequency values can be sorted from the highest to the lowest [5].

2.3 Altera Quartus II

Altera Quartus II is a programmable logic device design software produced by Altera. Altera Quartus II enables analysis and synthesis of HDL design, which enables the developer to compile the designs, performing timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli and configure the target device with the programmer. In Altera Quartus, there are includes a modular compiler. The compiler module such as Analysis and Synthesis, Partition Merge, Fitter, Assembler, TimeQuest Timing Analyzer, Design Assistant, EDA Netlist Writer, Hardcopy Netlist Writer.

In designing this microprocessor are using the Analysis and Sythesis feature to see how VHDL works on this microprocessor. Aims to find out the performance using Analysis and Synthesis. Quartus II is a development tool of Field Programmable Gate Array (FPGA) and Complex Programmable Logic Device (CPLD) launched by Altera, provides a fully integrated development package environment that has nothing to do with the circuit structure but with all the

features of digital logic design. In Quartus II, the design of combinational logic circuit is carried out [6].

2.4. Verilog HDL

Verilog HDL (Hardware Description Language) is a hardware description language that can model the behavior and structure of digital systems at multiple levels of abstraction, ranging from the system level down to that of logic gates, for design entry, documentation and verification purpose. To implement all the VHDL modules, it is necessary to use VHDL packages which provide all the necessary data types, operators and functions. It is essential as it provides *std_logic* and *std_logic_vector* data types including their type conversations [7]. The program is loaded into memory as part of the VHDL model [8].

3. Proposed Design

3.1 System Model

The system model will explain the parts of the microprocessor and how it works.

3.1.1 Microprocessor

In the microprocessor, there are several components to perform input and output as well as executors and drafters. These components are Control Unit (CU), Datapath and Memory. The microprocessor works because of the instructions that will be ordered by the control unit, then translated by the ALU and stored in the data register. The final process, these instructions stored in memory.

3.1.1.1 Control Unit

Control Unit as control all work on the microprocessor.

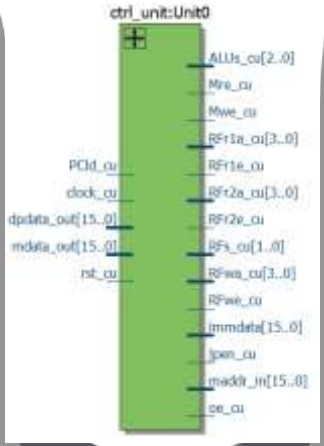


Fig 3.1 Control Unit

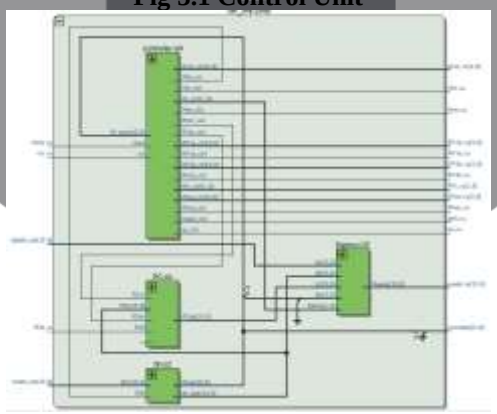


Fig 3.2 Inside of Control Unit

- Program Counter / Instruction Pointer : register recognize the address of the data/instructions entered into the microprocessor.
- Controller : retrieve the address of data/instructions that are input into the microprocessor.
- Instruction Register : fetch the address of the data/instruction from the controller and take it to be decode on the instruction decoder to find out the instructions that were orderd. Instruction register as a temporary storage place containing binary numbers.

3.1.1.2 Datapath

Datapath as a link register in the process of executing an instruction, which is included in the instruction set.

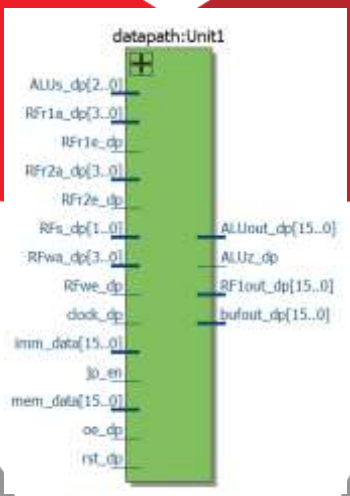


Fig 3.3 Datapath in microprocessor

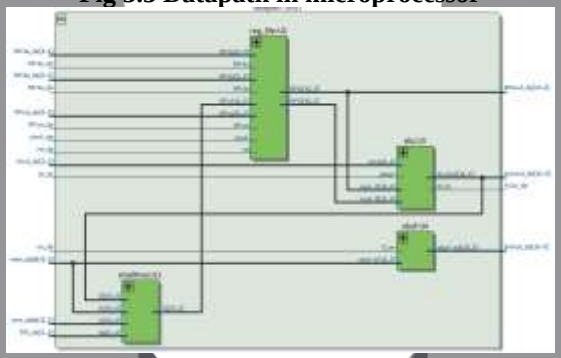


Fig 3.4 Inside of Datapath

- Data Register/Register File : logs I/O locations to transfer data to and from I/O locations. Instructions that are temporarily stored in the instruction register will be sent to the data register/register file to determine and record the I/O that will transfer these instructions to the ALU.
- ALU : perform arithmetic and logical operations. Instructions on data registers are sent to the ALU to be translated into numbers and logic (such as logic and binary gates).

3.1.1.3 Memory

Memory is an electronic circuit that can stored and provide/represent data/instructions. Instructions that have been decoded by the ALU are stored in memory on the microprocessor. This memory is a container/place to accommodate the actual instructions that have performed arithmetic and logic operations on the microprocessor, which will repeat the instruction command. This condition is called looping, one of the working properties of microprocessor.

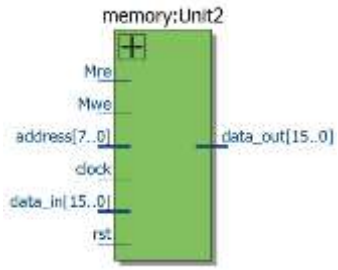


Fig 3.5 Datapath in microprocessor

3.2 System Block Diagram

In this block diagram, it starts from control unit as the brain of processor to control the microprocessor. In control unit, there is controller are obtained next state and control logic with state register. In designing, the VHDL code has been modifying to accommodate Huffman Coding instructions so that the microprocessor can work according to the instructions ordered. In the control unit, the register instruction will send data to the datapath to be interpreted by the ALU of the command. After being translated by the ALU, the instructions are sent to memory to be stored on microprocessor. This way of working continues until the next instruction. Because the working system of the microprocessor itself is looping.

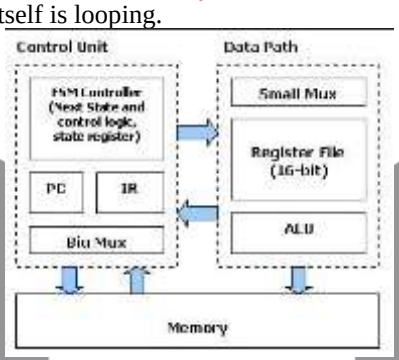


Fig 3.6 Block Diagram of Performance Inside the Microprocessor

3.3 Flowchart

The following flowchart shows the working principle of the proposed system to be executed.

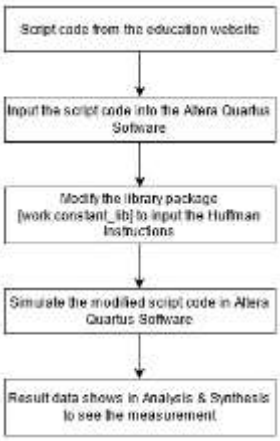


Fig 3.7 Flowchart of executing the microprocessor code

To create the microprocessor in Altera Quartus, it must determine the microprocessor script code that can accommodate these instructions by looking at the package library for storing Huffman instructions into the microprocessor. The author determines the microprocessor script code whose coding architecture is complete so it can modify the library package script code to enter instructions into the microprocessor script code. The microprocessor script code is simulated on Altera Quartus

to see the results of the measurement data on the microprocessor. The simulation results can be seen in the Analysis & Synthesis features, which can show the total number of logic elements, registers, and the number of pins of the microprocessor coding.

3.4 System Performance Parameter

The parameters used in the system as follows :

- Analysis and Synthesis, to determined the performance of microprocessor
- Huffman Coding, to determine the microprocessor can work with Huffman Coding instructions
- Performance Efficiency, achive the performance efficiency due to optimal chosen machine instruction process.

4. Result and Analysis

4.1 Result and Analysis

The Huffman Coding method is carried out to determine the optimal performance of the microprocessor to carry out the instructions given. Microprocessor optimization seen from analysis and Synthesis on Altera Quartus.

4.2 Result Design and Measurement

Data analysis looks from the results of Total Logic Elements, Total Combinational Functions, Dedicated Logic Registers and Total Pins.

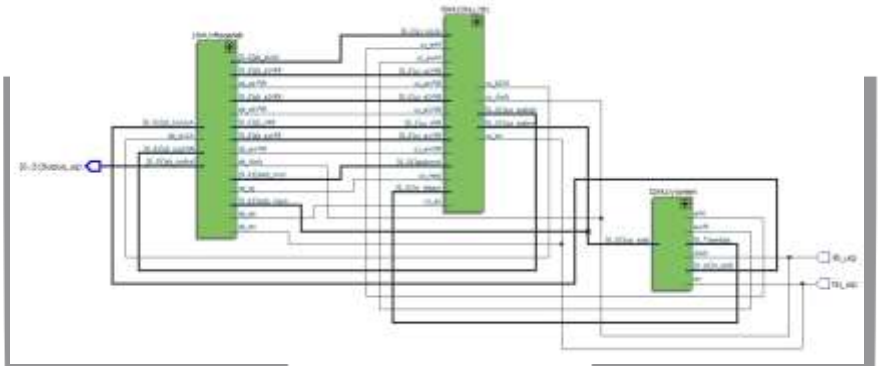


Fig 4.1 Final Design of Microprocessor in Altera Quartus Software

Flow Summary	
Flow Status	Successful - Mon Jul 19 13:54:15 2021
Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013 SJ Web Edition
Revision Name	Microprocessor
Top-level Entity Name	Microprocessor
Family	Cyclone IV E
Total logic elements	3,719 / 6,272 (59 %)
Total combinational functions	3,713 / 6,272 (59 %)
Dedicated logic registers	4,465 / 6,272 (71 %)
Total registers	4465
Total pins	18 / 92 (20 %)
Total virtual pins	0
Total memory bits	0 / 276,480 (0 %)
Embedded Multiplier 9-bit elements	2 / 30 (7 %)
Total PLLs	0 / 2 (0 %)
Device	EP4CE6E22C6
Timing Models	Final

Fig 4.2 Result Performancy Measurement of Design Microprocessor

4.3 Analysis Measurement and Comparison Data

This analysis data uses a comparison between the design of the microprocessor in the thesis and the Intel 8088 Microprocessor. Comparison of the microprocessor using the Intel 8088 because one of an example of a 16-bit microprocessor and includes a processor compatible with computer motherboards on the market.

SPECIFICATIONS	MICROPROCESSOR DESIGN	MICROPROCESSOR INTEL 8088
Logic Elements	5719/6272 elements logic gate = 5719 : 6 = 953	29000 elements logic gate = 29000 : 6 = 4833
Total Registers	4465/6172 registers	14 registers
Total Pins	18/92 pins	40 pins

Table 4.1 Table of Comparison Data Specification on Microprocessor

source : https://en.wikipedia.org/wiki/intel8088

The microprocessor design in this thesis shows a higher number than the Intel 8088 microprocessor because the capacity provided/estimated at Altera Quartus is 6272 more elements. Meanwhile, the intel 8088 microprocessor does not have spare element capacity, which means it only provides element capacity for the entire microprocessor.

The effect of the comparison results of the microprocessor designed in this thesis with the Intel 8088 microprocessor based on the measurement data is that the microprocessor has more spare element capacity, as evidenced by the comparison of the results of the designed elements with the results of the elements provided on Altera Quartus so that they can store instructions and not reduce the performance of the microprocessor itself.

5. Conclusion

In this Thesis, the microprocessor is designed to accommodate 9 instructions. Because the preparation of this thesis refers to the Nimas Sekar Fathihah's Thesis, which is explained using only the instructions ADD, ADDI, LW, SW, BEQZ, BNEZ, SEQ, SGT, and SUBI. As for the Intel 8088 microprocessor itself, it does not have spare capacity, either in the form of reserves in the registers or the logic elements on the pins.

References

[1] N. Karna, N. Fatihah, and D. Kim, "generic," in 2019 International Conference on Information and Communication Technology Convergence (ICTC), 2019, pp. 612–616.
[2] P. Deshmane, M. Lad, and P. Mhetre, "8 Bit Microprocessor Using VHDL," vol. III, no. Iv, pp. 241–246, 2014.
[3] G. R. Gare and K. P. A. L. R, "Custom 8 Bit Microprocessor Designing and Implementation on FPGA Board," pp. 113–118, 2016.
[4] R. Praisline Jasmi, B. Perumal, and M. Pallikonda Rajasekaran, "huffman," in 2015 International Conference on Computer Communication and Informatics (ICCCI), 2015, pp. 1–5.
[5] N. Karna, N. Fatihah, and D. S. Kim, "Evaluation of DLX Microprocessor Instructions Efficiency for Image Compression," ICTC 2019 - 10th International Conference on ICT Convergence: ICT Convergence Leading the Autonomous Future, pp. 612–616, 2019.
[6] H.-y. Shen, J.-h. Liu, and J.-h. Li, "Application of Quartus II in Digital Electronic Technology Teaching," no. Amee, pp. 103–106, 2018.
[7] L. Isola, "Design and VHDL Implementation of an Application-Specific Instruction Set Processor," 2019.
[8] R. J. Hayne, T. Citadel, R. J. Hayne, and C. Engi, "AC 2011-5 : AN INSTRUCTIONAL PROCESSOR DESIGN USING VHDL An Instructional Processor Design using VHDL and an FPGA," 2011.