A Dynamical Machine Control System using Simple Processing Unit Based-on FPGA

F W Wibowo\(^1\)

\(^1\)Informatics Department, Universitas Amikom Yogyakarta, Indonesia 55238

E-mail: ferry.w@amikom.ac.id\(^1\)

Abstract. This paper aims to design a dynamic machine control system device. The proposed dynamic device models are a combination architecture of memory and Arithmetic-Logic Unit (ALU). The configuration of the memory has used a random access memory (RAM), and the configuration of ALU is the core of the central processing unit (CPU), which consists of two functions, namely the arithmetic unit and the logical unit. In this paper, the arithmetic unit consists of addition, subtraction, multiplication, and division, whereas the logical units are AND, OR, NOT, and XOR. The shifter circuit in this paper was also applied to complete the programming code instructions in a software approach. The arithmetic unit and logic unit selector path in the ALU configuration employed a multiplexer circuit. This paper used a top-down design technique by determining the components in implementation. By knowing the function and work of these components, it is much easier to combine the components employing the configuration code of the hardware description language, namely a Very High-Speed Integrated Circuit Hardware Description Language (VHDL). The implementation of the VHDL code was configured into the Xilinx Spartan-3E Field Programmable Gate Array (FPGA) to determine the component consumption and the resulted delay time. The results obtained from the configuration of the ALU components consume 26 slices, 50 4-input look-up table (LUT), 17 input/output (I/O), and 17 Bonded IO, while the component delay time is 11,413 ns.

1. Introduction

The development of technology in hardware and software in the field of computing must have economic aspects and efficient in its implementation. The implementation of machine control often uses a microcontroller as a controller because the device can be programmed easily and economically [1]. However, for controller conditions that have faster response specifications, the use of a microcontroller needs to be considered further. It is because of the many different types and specifications among the microcontrollers. The speed aspect, if required in a dynamic machine control field by utilizing reduced instruction set computer (RISC) modeling, it can be configured using Field Programmable Gate Arrays (FPGAs) [2]. The RISC model can mitigate the instructions used and make the design process more manageable. The principle of
hardware platforms can be relied upon in hardware problems with special assignments. The dynamic aspect can be done using a software approach because the programming is flexible. So in this paper utilizing the hardware approach to configure software-based.

Many innovations are poured in various aspects of scientific disciplines, but in this case, they can not be separated from developing and finding an algorithm. The increasingly complex and compelling performance of the algorithms in the world of computer science will have an impact on the implementation of better designs. The description of the nature of the algorithm, especially in the design of very large scale integration (VLSI) technology, is essential in the development and innovation of the integrated circuit (IC) forms. To implement the microprocessor as a core in the central processing unit (CPU) or microcontroller, primarily when used to determine compile techniques which configure a component for storing data instructions, random-access memory (RAM) [3], which varies as a data storage programmatically [4].

Data communication between components has an essential role in processing instructions and signaling. Determination of the path/bus and signal activation of a component must be done properly. If it is not right or not suitable, the resulting conditions will cause errors, including the determination of the time delay, so that the noise of digital data (glitch) can be anticipated in the design of the configuration [5]. The arithmetic logic unit (ALU) design implemented in the FPGAs has a significant contribution in determining the gates and registers it consumes. The choice of configuration and programming code has a significant impact on determining the consumption of components used, in this case ignoring compile techniques and determining routing and placement. Configuring very high speed integrated circuit hardware description language (VHDL) codes in FPGA designs in some specific models has the use and placement of the same components, such as the design of read-only memory (ROM) [4]. The methods of top-down and bottom-up design in FPGA implementation will have a significant effect on the ease and speed of design. The top-down design technique is done by determining the components that will be used in the implementation by knowing the function and work of the component first, then determining the configuration code in the form of hardware description languages, such as VHDL and Verilog. This case is contrary to the bottom-up design technique that uses a hardware description language to make the components needed in the design. Then the components are integrated for FPGA configuration. Many FPGA implementation designers are more likely to use top-down design techniques rather than bottom-up because this technique is relatively easy to understand and construct [5]. This paper uses the VHDL as the hardware description language for FPGA configurations. For FPGA designers, the ability to develop cases and signal logic is a crucial point in determining the design components and the use of VHDL code. The proper implementation of the VHDL code for the FPGA configuration that will be generated is quite strictly related to the CPU usage on the FPGA itself. VHDL code will be compiled with various stages so that it will result in the consumption of components along with their delay time, consumption of basic gates, and registers; then, it is relatively contributing to the
design [6].

FPGA can be easily used to create a microcontroller system configuration by consuming logic block components [7]. Microcontrollers that are made using FPGA is less efficient than using integrated circuits (IC) microcontrollers, but also make it easier to understand the design of the microcontroller systems [8]. ALU implementation varies depending on the bus or number of bit lines, arithmetic, and logic functions that want to be implemented. This paper uses 2x4-bit primary input, consisting of buses A and B, plus 1-bit carry and 5-bit selector. The selector bus is used to select the component configuration to be implemented as a function output.

2. Related Works

In the control system perspective, there are three main compositions, i.e., a sensor, control logic, and actuator. The sensor sends the measured data to the control logic to obtain the embedded optimizer solver using optimization problem formulation, and the result is the control variables that have to be done by the actuators. The process parameters and variables are discrete and also changed at distinct moments in time. The changes, in this case, are defined in advance by the program of instructions. There are two types of control models, i.e., combinational logic control and sequential control. The combinational logic control is using the state of the system has changed. It is more likely as the event-driven changes while the subsequent control is using a certain amount of time has elapsed, so it is closer to the time-driven changes. The FPGAs have been employed as a custom reconfiguration controller for dynamic circuit specialization (DCS) [9].

Arithmetic Logic Unit (ALU) is a core unit of the central processing unit (CPU). It consists of a series of combinational logic that is built from arithmetic and logic operations components using two input operands in computer instruction commands. ALU displays addition, subtraction, integer multiplication, AND, OR, NOT, XOR, and other Boolean operations. Besides that, ALU has a selector input from the encoded CPU instructions to select the components to be operated, and the output data is displayed. The selector path is decoded using ALU which adjusts to the operation number 2n, where n is the encoded input for the selection of the desired operation. An understanding of digital numbers is needed in the design of ALU. For example, if an arithmetic operation is written as \( Y = A + /B + 1 \), then the complement of one B minus A and added 1. This is the same as the reduction of B to A, then added by 1, or in other words A-B + 1. There are three ways to represent signed numbers, namely:

(i) Signed magnitude is a way of representation by negating a number by adding a sign bit in front of the number, where the sign bit 0 represents a positive number, while the sign bit 1 represents a negative number. Suppose the number +1110 is represented as 010112, while the number -1110 is represented as 110112.
(ii) Complement 1 (one) is a way of representing numbers by compressing each bit
number. Suppose the number +1110 is represented as 010112, while the number -1110 is represented as 101002.

(iii) Complement 2 (two) is a way of representing numbers by negating the numbers and then complementing each bit (similar to complement 1), then adding 1. Suppose the number +1110 is represented as 010112, while the number -1110 is represented as 101012.

ALU bit width in the instructions handled, usually the same as the name of the number of processor bits where the external bus can be less, for example, 4-bit processor, processor, 8-bit, 16-bit processor, 32-bit processor, 64-bit processor, etc. Floating-point operations are carried out by floating-point units. Some processors also use ALU to count addresses, such as increment/decrement in the program counter, while some processors also have separate logic. Some processors divide the ALU into two units, namely the arithmetic unit and the logical unit; also, some processors also divide the arithmetic unit into fixed-point operations and floating-point operations. On personal computers (PCs) using floating-point processes are sometimes carried out by floating-point groups on separate chips called numeric co-processors. ALU usually has direct input and output access to the processor controller, central memory (random access memory (RAM)) on a PC, and input/output (I/O) devices. Input consists of instructions (sometimes referred to as machine instruction words) consisting of operation code (op-code), operands, and format code. The operation code instructs the ALU to display the operation. At the same time, the operand is used in the process, for example, the ADD A, B command indicates that the procedure performed is a sum arithmetic operation, where A plus B results are stored in the storage register. Usually, the accumulator is a register where the function usually as an operand A. But the operand must be defined first, whether the value of the operand is fixed-point or floating-point. If the results are in accordance with the initialization of the data, the operation will be successfully operated. Otherwise, the status will indicate that the data will be stored in a permanent place, which is often referred to the machine status word. The bit flow and operations displayed on the ALU are controlled by a series of gates. It is also controlled by a sequence of logical units employing the algorithm for each operation code. In arithmetic units, multiplication and division are carried out by a combination of addition, subtraction, and shift operations [10]. So the design of the ALU is an important part of the processor. It is an approach to accelerate the process of instruction used by the software and always developed on an ongoing basis.

The mnemonic instructions are easy for humans to write and understand than the machine code. The labels can also be used to identify particular memory locations. The assemblers translate instructions that are understandable to humans into the machine language. From this state, the control unit causes the CPU to execute a sequence of steps correctly, and utilizing the control signals asserted on the various components in making the components active. The control unit keeps things synchronized and make sure that bits flow to the correct components as required.
3. Methodology

The design of the simple controller unit in this paper using the RISC method by employing simple Microprocessor without Interlocked Pipelined Stages (MIPS) architecture, as shown in Figure 1. The design consists of the instruction memory, ALU, shifter, and data memory. The memories used in this paper is using the design of memory, as stated in the [3] and [4]. The arithmetic logic unit (ALU) design and shifter circuit that will be implemented in the field programmable gate array (FPGA) has an 8-bit input consisting of 4-bit input A and 4-bit input B, 4-input selection bit, S, and carry input, Cin. Inputs of A and B function as operands, while input S functions as an operation selector for the ALU components. The components implemented in the ALU design consist of logic units, arithmetic units, multiplexer circuit (MUX), and shifter circuits. The architecture of the simple controller design is illustrated in Figure 1.

![Figure 1. Simple controller unit design.](image)

As shown in Figure 1, there are two 4-bit inputs A and B used as inputs for the logical unit and arithmetic unit components. In contrast, the output selector for the logical unit component and the arithmetic unit has used a multiplexer (MUX), which implements 2-bit selector input, namely $S[1:0]$. The arithmetic unit has a 1-bit carry line, Cin. The arithmetic and logic unit outputs are designed to produce 4-bits. The bit selector in the multiplexer circuit uses the selector path, $S[2]$, where if the value of $S[2] = '0'$ then the multiplexer circuit will select the output from the arithmetic unit, whereas if the value of $S[2] = '1'$ then the multiplexer circuit will choose output from logical units. As input shifter circuit is used data from operand A on the arithmetic unit on the passage function (transfer), the value of the bit sliders of the bus formed consists of two combinations of selector and carry input, Cin. The value issued from the multiplexer for the transfer function consists of 4-bit input.

In contrast, the shift function both left (SHL) and right (SHR) are determined by VHDL configuration with the left shift value. Then the rightmost bit has a reference value of ‘0’ and VHDL configuration with the right shear value, then the leftmost bit
Table 1. Design of ALU operation and shifter

| Arithmetic Unit | Operation | S4 | S3 | S2 | S1 | S0 | Cin | Function |
|-----------------|-----------|----|----|----|----|----|-----|----------|
|                  | Y=A       | 0  | 0  | 0  | 0  | 0  | 0   | Transfer to A |
|                  | Y=A+1     | 0  | 0  | 0  | 0  | 0  | 1   | Increment of A |
|                  | Y=A+B     | 0  | 0  | 0  | 0  | 1  | 0   | Addition |
|                  | Y=A+B+1   | 0  | 0  | 0  | 0  | 1  | 1   | Addition with carry |
|                  | Y=A+/B    | 0  | 0  | 0  | 1  | 0  | 0   | Addition of A with complement of B |
|                  | Y=A+/B+1  | 0  | 0  | 0  | 1  | 0  | 1   | Substraction |
|                  | Y=A-1     | 0  | 0  | 0  | 1  | 1  | 0   | Decrement of A |
|                  | Y=A       | 0  | 0  | 0  | 1  | 1  | 1   | Transfer to A |

| Logic Unit | Operation | S4 | S3 | S2 | S1 | S0 | Cin | Function |
|------------|-----------|----|----|----|----|----|-----|----------|
|            | Y=A and B | 0  | 0  | 1  | 0  | 0  | 0   | AND      |
|            | Y=A or B  | 0  | 0  | 1  | 0  | 1  | 0   | OR       |
|            | Y=A xor B | 0  | 0  | 1  | 1  | 0  | 0   | XOR      |
|            | Y=/A      | 0  | 0  | 1  | 1  | 1  | 0   | Complement of A |

| Shifter Unit | Operation | S4 | S3 | S2 | S1 | S0 | Cin | Function |
|--------------|-----------|----|----|----|----|----|-----|----------|
|              | Y=A and B | 0  | 0  | 0  | 0  | 0  | 0   | Transfer of A |
|              | Y=shl A   | 0  | 1  | 0  | 0  | 0  | 0   | Left shift of A |
|              | Y=shr A   | 1  | 0  | 0  | 0  | 0  | 0   | Right shift of A |
|              | Y=0       | 1  | 1  | 0  | 0  | 0  | 0   | Transfer of 0 |

has a reference value ‘0’. The shifter is equipped with two special conditions, namely the transfer value A and the transfer value ‘0’, in addition to shifting the data bit left or right. If the value of the selector S [4:0] and carry input Cin are equal to ”00000” then the shifter circuit unit only passes data bits from the operand A. If the selector, S[4:3] has an ”11” value and all S[2:0] and Cin has a value of ‘0’, all data released to ”0000”. The function and work of the bit configuration can be written according to table 1.

Table 1 can be added several more functions, but the selector function must also be added because the selector path is critical to determine the output data.

4. Result

In the top-down design approach, four arithmetic modules - addition, subtraction, multiplication, and division - have been implemented by combining these arithmetic functions to form a floating-point ALU unit [11]. Each module is divided into sub-modules. Two-bits selector combined to select operands in the ALU design that are realized using VHDL. Design functionality is validated through VHDL simulation. The
principles that have been used in designing the ALU are also applied in this paper. But the difference in its application lies in the algorithm and the selection of more complex design functions. Pipelining was not used in the design because it did not indicate the breakdown of the ALU block.

The reverse logic is applied in the ALU to get high power optimization by determining the appropriate control logic in one parallel sum so that various arithmetic operations can be realized [12]. The use of proper components can speed up the time delay produced by logic delay and route delay to form modules, and the arithmetic units applied in this paper using the sum (+) instead of a combination of existing logic. However, the use of Xilinx Spartan-3E FPGA components is less optimal because it does not have a digital signal processing (DSP) feature. The register transfer logic (RTL) chart generated from the FPGA is depicted in Figure 2.

The consumption of logical blocks consumed by the Xilinx Spartan-3E FPGA configuration is written as in table 2.

5. Conclusion

The results obtained from the implementation of the arithmetic logic unit and related to the shifter on the Xilinx Spartan-3E FPGA consume 26 slices, 50 4-input Look-Up-Table (LUT), 17 Input/Output (I/O) and 17 Bonded IOB. The comparison of delay time between logic and route is 77.4% and 22.6%.

6. References

[1] Wibowo FW and Habib M 2017 A low-cost entry door using database based on RFID and microcontroller ARPN Journal of Engineering and Applied Sciences vol 12 no 17 pp 5073-5077
[2] Scotti G and Zoni D 2019 A Fresh View on the Microarchitectural Design of FPGA-Based RISC CPUs in the IoT Era J. Low Power Electron. Appl. pp 1-19 DOI: 10.3390/jlpea9010009
[3] Wibowo FW 2018 Implementation of FPGA in index data storage as a database International Journal of Engineering and Technology (UAE) vol 7 issue 4 pp 94-97.
[4] Wibowo FW 2010 The Conservative Structure of Synthesizing Read Only Memory Using VHDL on FPGA Proc. of 4th International Seminar on Industrial Engineering and Management (4th ISIEM) pp 231-235
[5] Wibowo FW 2011 Interoperability of reconfiguring system on FPGA using a design entry of hardware description language Proc. of 2011 Computation and Communication Technologies
Figure 2. RTL Diagram of the ALU.

3rd International Conference on Advances in Computing, Control, and Telecommunication Technologies, ACT 2011 - Computer Science Series 1 pp 79-83

[6] Wibowo FW 2015 Implementation of Viterbi algorithm based-on field programmable gate array for wireless sensor network Advanced Science Letters vol 21 no 11, pp. 3521-3525 DOI: 10.1166/asl.2015.6594

[7] Schoeberl M 2011 Leros: A Tiny Microcontroller for FPGAs Proceedings of the 21st International Conference on Field Programmable Logic and Applications (FPL 2011) https://doi.org/10.1109/FPL.2011.13

[8] Nimase AM and Bhalerao DM 2017 Design and implementation of microcontroller in FPGA for IoT International Journal of Advance Research, Ideas and Innovations in Technology vol 3 issue 6 pp 1308-1313

[9] Kulkarni A Kizheppatt V and Stroobandt D 2015 MiCAP: A custom reconfiguration controller for dynamic circuit specialization Proc. of 2015 International Conference on ReConFigurable
[10] Wibowo FW 2018 Comparison of Multiplication Algorithms Based on FPGA Proc. of 2018 2nd Borneo International Conference on Applied Mathematics and Engineering (BICAME) pp 326-331

[11] Dhanabal R Bharathi V Chandrakiran G and Reddy M 2014 VLSI design of floating point arithmetic logic unit Journal of Theoretical and Applied Information Technology vol 64 no 3 pp 703-709

[12] Chinmaye R Amrutha C Vishwanath S Ghosh SS and Sarkar S 2015 International Journal of Engineering and Innovative Technology(IJEIT) vol 4 issue 12 pp 87-92

Computing and FPGAs (ReConFig) pp 1-6 DOI: 10.1109/ReConFig.2015.7393327