Design And Implementation of A MIPS Processor with Signal Processing Extensions On FPGA

Heba C Josy*, Kelvin Tony†, Krishnendu R‡ and Liz Abraham§
Rajagiri School of Engineering and Technology
Email: *hebajosy98@gmail.com, †kelvintony2@aol.com, ‡kichuravi28@gmail.com, §lizabraham98@gmail.com

***Abstract*—The purpose of this paper is to design a single cycle central processing unit with signal processing extensions in an FPGA. The processor will be able to handle different instructions, including R-type, I-type and J- type. To the existing instruction set, we are going to include signal processing functions viz. Fast Fourier Transform (FFT). This design tries to meet the faster processing demand in consumer electronics. Also with this design easy debugging of architecture is possible and therefore making it an ideal teaching tool.**

***Index Terms*—Fast Fourier Transform, twiddle factor, Microprocessor without Interlocked Pipelined Stages**

I. INTRODUCTION

MIPS Technologies in the United States developed the Microprocessor without Interlocked Pipelined Stages (MIPS) which uses the reduced instruction set computer (RISC) architecture. MIPS architecture was first developed by John L. Hennessy at Stanford University and proved that performance can be improved greatly with their prototype microprocessor with five-stage execution pipeline and cache controller which could be integrated onto a single silicon chip.

One of the key features of the MIPS architecture is the regular register set. It consists of the 32-bit wide program counter (PC), and a bank of 32 general-purpose registers called r0-r31, each of which is 32-bit wide. All general-purpose registers can be used as the target registers and data sources for all logical, arithmetical, memory access, and control-flow instructions.MIPS is a load or store architecture (also known as a register-register architecture), except for the load or store instructions used to access memory, all instructions operate on the registers. This work presents a hardware design architecture of a 32 bit single cycle MIPS processor along with signal processing extensions(FFT). MIPS Processor consists of blocks like control unit, program counter, data memory, ALU, register file, adder,multiplexers,sign expansion etc.

II. IMPLEMENTATION OF SINGLE CYCLE DATAPATH

The single-cycle micro-architecture proposed here executes an entire instruction in one cycle. It follows the standard design principles of MIPS processor and therefore making it easy to explain and has a simple control unit. Because it completes the operation in one cycle, it does not require any non-architectural state. However, the cycle time is limited by the slowest instruction. The main state elements used in this processor are control unit, program counter, data memory, ALU, register file, adder, multiplexers, sign expansion etc.

A. State Elements

The state elements in the proposed design include the basic building blocks of the design and they consists of the program counter, the assorted registers and the ALU. The proposed design makes use of the standard state elements of the MIPS architecture to maintain the simplicity and efficiency. The new states are calculated from the current states using the different combinational logic between the various state elements.

1) Program counter: The program counter, a register, contains the address location of the next instruction that is to be executed. When the current instruction gets fetched by the processor, the program counter gets incremented by one. When the fetch stage is completed, the program counter calculates and stores the address of the next instruction in the sequence. The program counter is set to 0 when the processor is restarted or is reset. The program given to the processor to execute is usually a set of ordered instructions and each instruction is stored in an address in the memory and since the program counter stores the address of the next instruction some refers the program counter as address pointer.

2) Instruction Register: Every instruction to be executed by the processor is stored in the instruction register. The instruction is stored in the instruction register while the control unit decoded the instruction and finally executed which might be a several step process. The outputs of the IR are given to the control unit to generate various timing and control signals to different state elements involved in the instruction execution stage. The instruction register is loaded with instruction which is pointed by the program counter.

3) Register File: The proposed design has a 32-element 32-bit register file. It consists of one write port and two read ports. The A1 A2 are each 5-bit port each specifying the address of source operands. RD1 and RD2 outputs data read from any of the 32-bit registers. A3 which is the write port takes a 5-bit address input and writes the data in the WD on the rising edge of the clock to the specified register if the WE3 is enabled.

4) Data Memory: The data memory implemented here consists of one read port and one write port. When the write enable, (WE), is set to 1, the data is written into the specified address A on the clocks rising edge. The address A is read onto RD when the write enable is set to 0. The data memory, register file and the instruction memory are all read using combinational logics. That is, if the address changes, the new data appears at RD directly; the clock is not involved in these operations. The data memory is written on the rising edge of

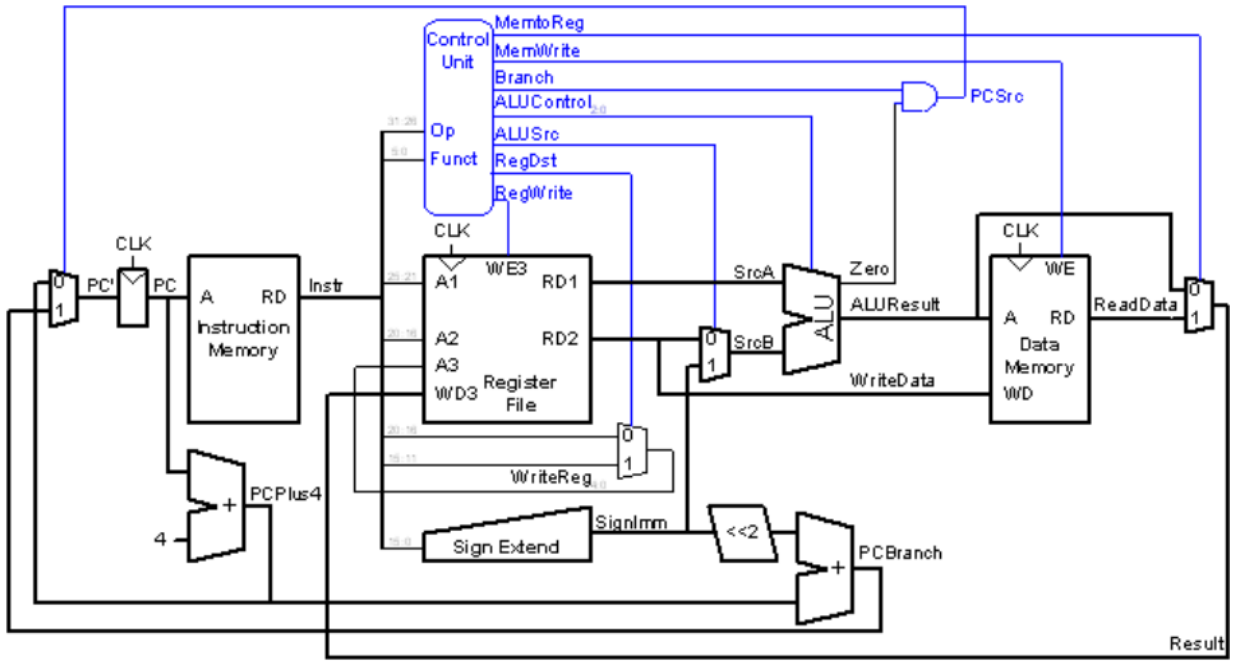

Fig. 1. Single Cycle Datapath

the clock cycle. The different signals i.e. the write enable, data, and address must be setup before the clock edge and after the clock edge it should remain stable.

III. FAST FOURIER TRANSFORM

A. Introduction

Fast Fourier Transform (FFT) is one of the most important and widely used algorithms in the signal processing domain. The signal processing applications need high throughput, and therefore the goal is to reduce the circuit area and then to reduce the delay. The minimization of the power consumption and the amount of hardware used is crucial to the reduction of FFT complexity. James Cooley and John Tukey first introduced the concept of FFT. The Discrete Fourier Transform (DFT) of a sample, or its Inverse Discrete Fourier Transform (IDFT) can be found using the FFT butterfly algorithm. The time domain to frequency domain and vice versa conversions are done by Fourier analysis. The DFT is obtained by decomposing the sequence of values or samples into its components of different frequencies.

FFT is widely used for many applications in engineering, science, and mathematics which include areas such as : communications, signal processing, instrumentation, biomedical engineering, numerical methods, sonic and acoustics, and applied mechanics. It is the most important algorithm in the speech processing methods. The demand for the FFT transform is still growing.

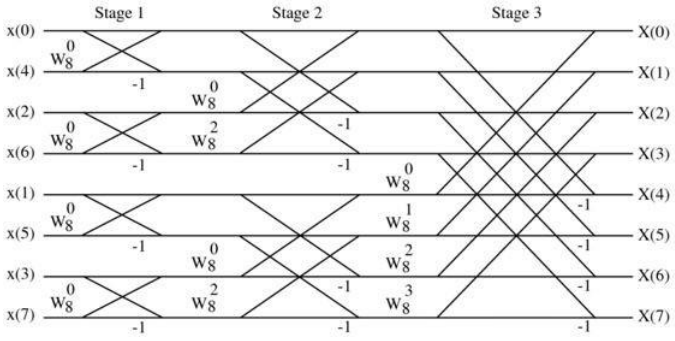

Fig. 2. Butterfly Structure

B. Radix-2 DIT FFT

When length 8 is the input to the FFT the output data is also in the order of 8. It can be represented in the form of a butterfly diagram. The twiddle factor computation is done at each stage of buttery diagram. The input sequence taken in a shuffled order in the DIT algorithm. Here the sequence for which DFT is found is successively divided into smaller sequences and the DFTs of the sub-sequence are combined. For DIT the input is bit reversed, while output is in the original order. The processing of the data is same as in 16-bit radix-2 DIT FFT, the only difference is that the length changes. The number of inputs increase the butterfly units for processing also increases.

Depending on the number of bits, the number of stages and butterflies can be decided. The number of complex mul-

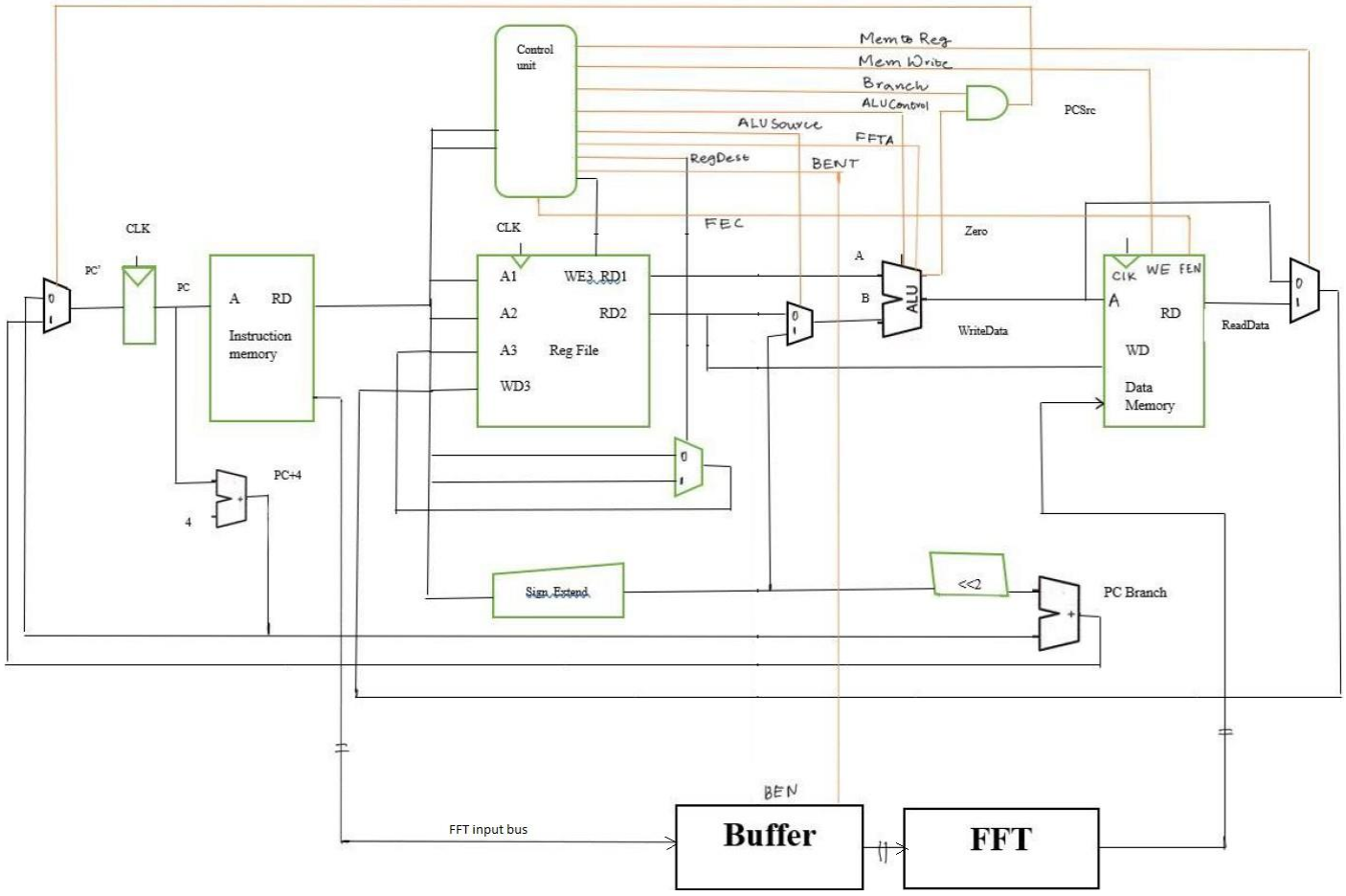

Fig. 3. Single cycle datapath with FFT extension

tiplications is $\frac{N}{2}\log_2 N$ and the number of complex additions is $N\log_2 N$. FFT method recursively breaks down a discrete fourier transform of composite size n = rm into r smaller transforms of size m. These smaller discrete fourier transforms are then combined via size-r butteries, which themselves are DFTs of size r (performed m times on corresponding outputs of the sub-transforms) pre-multiplied by roots of unity known as twiddle factors. This is known as the decimation in time case; the steps can also be performed in reverse this process is known known as the decimation in frequency. Figure 2 shows the butterfly structure for FFT calculation. The randomness of large arrays of partially random numbers can be improved by the butterfly algorithm. Both the decimation in time and decimation in frequency are complementary and has the same amount of complexity.

IV. IMPLEMENTATION OF FFT IN THE DATAPATH

The block diagram in figure 3 shows the proposed architecture with the FFT unit. When the FFT instruction is received the control unit enables the FFT buffer module, and the 8 inputs to the FFT block is stored in the buffer and then given to the FFT block where the calculation happen.

The FFT block (figure 3) mainly consists of a module for fixed point addition fixed point multiplication and an inverter. Separate addition and multiplication blocks(fixed point) are used here to improve the performance. a, b, c etc are the inputs to the FFT block and A, B, C, etc and Ai, Bi, Ci etc are the real and imaginary parts of each FFT output respectively.

The 8 FFT outputs are calculated using the equations of FFT(butterfly algorithm). Ideally the inputs to the FFT block can be given from an external source. In the proposed design the 8 inputs are stored in the instruction memory to test the functionality of the design. We have optimized the structure of this FFT by removing two redundant parts, that is we directly assign outputs Ai and Ei values, where Ai is the imaginary part of the first FFT output and Ei is the imaginary output of the fifth FFT output. After the computations are completed the outputs are stored in the data memory (in the last 16 addresses)for verifying.

Fixed point arithmetic is used to here to improve speed and maintain the simplicity of the design. The fixed point numbers implemented here has 32 bits. The 32^{nd} bit is the sign bit, where 0 represents a positive number and one represents a negative number. The bits from 30 to 15 represents the digits on the left side of the decimal point and the bits from 14 to

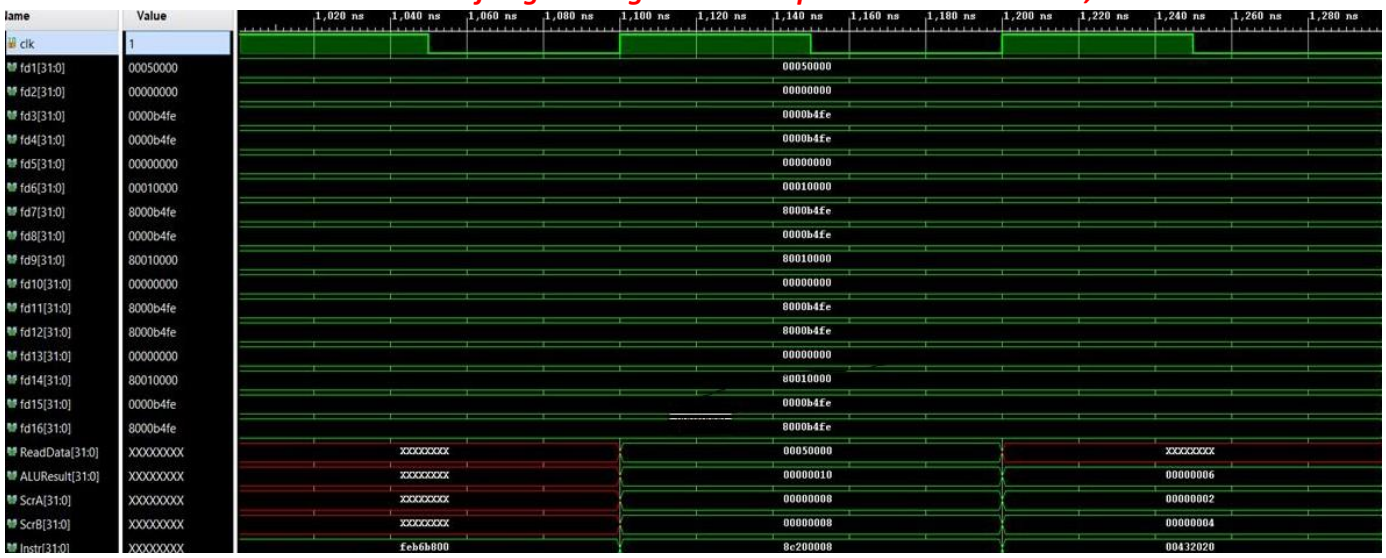

Fig. 4. Simulation results

0 represent the fractional part.

The proposed design was simulated using Xilinx Vivado and the functionality was verified. The figure 4 shows the output of 8 point FFT with the inputs 1,1,1,1,1,1,1 and 3.

V. CONCLUSION

A single cycle central processing unit supporting MIPS ISA can be designed and tested using this architecture.This processor is capable to handle different instructions including R-type, I-type and J-type. We also added a signal processing extension to the existing instruction set. The extension used is Fast Fourier Transform (FFT). Fast Fourier Transform is a computer algorithm used in digital signal processing (DSP) to modify, filter and decode digital audio, video and images. The processor was synthesized using Xilinx ISE platform. It is very rare for a modern processor unit to have a single-cycle design. The reasons for this are the long cycle times, the wasted resources, and the large amount of wasted time in each cycle. It is for this reason that single-cycle processors work as a good teaching tool, but are not often employed in actual designs.

VI. FUTURE SCOPE

All the basic arithmetical operations can be implemented in this processor. And in addition, we had also implemented the FFT function. In future, we can also further develop this into Short Time Fourier Transform. This transform helps in frequency scaling, cross synthesis etc.We can also advance our project as an IP Core. As essential elements of design reuse, IP cores are part of the growing electronic design automation (EDA) industry trend towards repeated use of previously designed components. The IP Core can improve the processing speed of data-intensive functions. Our processor can also be implemented as a single chip by implementing pipeline and parallel architecture.

ACKNOWLEDGMENT

Many thankful to the College management for providing an environment where we could develop and improve our skills and thereby better ourselves emotionally and professionally.We are indebted to our Seminar Guide, Mr. Karunakara P Menon, Asst. Professor, Department of Electronics and Communication Engineering for the constant help and support.

REFERENCES

[1] M. E. A. Ibrahim, M. Rupp , and H.A. H. Fahmy, Power Estimation Methodology for VLSI Digital Signal Processors, in Proc. ACSSC 08, 2008, paper 169593.

[2] K. Anand and S. Gupta, Designing Of Customized Digital Signal Processor B.T. Thesis, Indian Institute of Technology, Delhi, May, 2007.

[3] Gautham P, Parthasarathy R. Karthi, Balasubramanian. "Low Power Pipelined MIPS Processor Design," in the proceedings of the 2009, l2(h international symposium,2009 pp. 462-465.

[4] Harpreet Kaur, Nitika Gulati, "Pipelined MIPS With Improved Datapath", IJERA, Vol. 3, Issue 1, January -February 2013, pp.762-765

[5] K. Karuri and R. Leupers, Application Analysis Tools for ASIP Design: Application Profiling and Instruction-set Customization, 1st ed., Springer, 2011

[6] J. Becker, M. Glesner, A Parallel Dynamically Reconfigurable Architecture Designed for Flexible Application-Tailored Hardware/Software Systems in Future Mobile Communication, The Journal of Supercomputing, vol.19, no.1, 2001.

[7] T. Ferdous, Design, Synthesis and FPGA-based Implementation of a 32-bit Pipelined Digital Signal Processor, International Journal of Scientific and Engineering Research (IJSER), vol. 3, issue 7, 2012.

[8] M. R.S. Balpande, M.R.S. Keote, Design of FPGA based Instruction Fetch Decode Module of 32-bit RISC (MIPS) Processor, in Proc. ICCSNT 11, 2011, paper 10.1109, p. 409.

[9] Neenu Joseph. Sabarinath.S. "FPGA based Implementation of High Performance Architectural level Low Power 32-bit RISC Core", 2009 IEEE.

[10] David Harris, Sarah L. Harris Digital Design and Computer Architecture 2013.