International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 10, October- 2014

VHDL Implementation of 20-Bit RISC and DSP Operations in FPGA

M. Sethu, M.E. Digital Signal Processing
Dept. of ECE,
GKMCET,
Chennai – 600 063, Tamilnadu, India.

***Abstract* - The Reduced Instruction Set Computer (RISC) is a smaller instruction set used widely in the microprocessors and microcontrollers. By this RISC core is designed to perform some arithmetic operation and perform some DSP operations such as Discrete Cosine Transform (DCT), Inverse Discrete Cosine Tranasform (IDCT) and Fast Fourier Transform (FFT). The design of a Reduced Instruction Set Computer (RISC) and the Digital Signal Processor (DSP) system described using VHDL and is implemented in a Field Programmable Logic Array (FPGA). This 20 bit processor system has high general purpose register (GPR) orthogonality and communicates to peripheral devices via a serial bus.**

Keywords - Arithmetic Logic Unit (ALU), Central Processing Unit (CPU), Control Unit (CU), Field Programmable Logic Array (FPGA), General Purpose Register (GPR), Instruction Register (IR), Program Counter (PC), Reduced Instruction Set Computer (RISC), Register Set (RS), Multiply and accumulates (MACs), Very Large Instruction Word (VLIW).

I. INTRODUCTION

Reduced Instruction Set Computer (RISC) is a type of microprocessor architecture that utilizes a small, highly-optimized set of instructions, rather than a more specialized set of instructions often found in other types of architectures.

RISC statergy based on the insight that simplified instructions can provide higher performance if this simplicity enables much faster execution of each instruction. It use fewer instructions with simple constructs, therefore they can be executed much faster within the CPU without having to use memory as often. [2]

Currently, RISC has three major IP suppliers - ARM, MIPS & PowerPC. Each has its own characteristics and flexibility. PowerPC is a standard RISC architecture developed by the IBM, Motorola and Apple alliance known as AIM.

RISC can be described as a philosophy with three basic levels:

- All instruction will be executed in single cycle.
- Memory will only be accessed via load and store instruction.
- All execution units will be hardwired with no micro coding.

The RISC provides higher performance in computing because of little need of the external fetches, which take significant amount of processor time and also because of hard-wired instruction implementation.

Main features of a RISC processor are –

- Load/Store design
- Few addressing modes
- Fixed instruction size
- Few instruction formats
- Few operand sizes
- Better compilation
- Many instruction that access memory directly
- Variable length instruction encoding
- Pipelining can be implemented easily.[5]

II. RISC ARCHITECTURE

The Architecture of RISC system is shown in Fig. 1. It includes Decoder, fetch machine, Arithmetic and logic machine, and register set.

RISC consists of: This system can be separated into several states as shown in Figure 1. Each state describes the current operation or process being performed by the CPU and is described in a VHDL module. This system is the hardware within a computer system which carries out the instructions of a computer program by performing the basic arithmetical, logical, and input/output operations of the system.

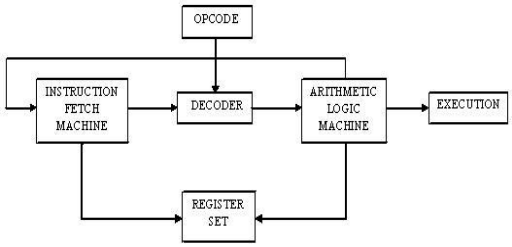

Fig. 1 RISC Architecture

Register Set (RS): In this information is encoded, stored, and retrieved. The RS of this system contains the following registers:

- IR - holds the current instruction.
- PC - holds the address of the next instruction.
- Load - holds data loaded from memory.
- Store - holds data being stored to memory.
- SR - when an operation involves two operands, the status signals are updated. The SR can also the used as an operand in arithmetic and logical operations.
- GPR[x] -up to 64 GPRs can be used in this architecture.

All GPRs and the SR can be used in any operation except for the load and store instructions. Only GPR can be used for loading and storing.

Instruction Fetch Machine: This machine fetches an instruction from external memory, and upon completion of the instruction fetch cycle this machine signals the decoder to decode the instruction. This machine utilizes a 3-bit up counter with an active low reset. The CPU changes states and begins to decode the instruction.

Decoder: Upon completion of the instruction fetch cycle, the instruction is decoded. The decoder reads bit 3 down to 0 of the IR, decides which of the sixteen operations the CPU needs to performs, and signals one of the next states to begin its operation.

Move Machine: The move machine controls all register movement. The most basic of these movements is the movement of data from one GPR to another GPR. On completion of the movement of data, a new instruction is fetched.

Arithmetic Logic Unit: The ALU performs arithmetic and logical operations on data. The data is taken from two GPRs and is moved to the ALU. The result is stored in a GPR. For operations that involve one operand, a GPR can be specified to store the result. The ALU supports two's complement data.

Figure 2 shows Spartan-3 FPGAs, The Spartan-3 family architecture consists of five fundamental programmable functional elements:

- Configurable Logic Blocks (CLBs) contain RAM-based Look-Up Tables (LUTs) to implement logic and storage elements that can be used as flip-flops or latches. CLBs can be programmed to perform a wide variety of logical functions as well as to store data.
- Input / Output Blocks (IOBs) control the flow of data between the I/O pins and the internal logic of the device. Each IOB supports bidirectional data flow plus3-state operation. Twenty-six different signal standards, including eight high-performance differential standard. Double Data-Rate (DDR) registers are included. The Digitally Controlled Impedance (DCI) feature provides automatic on-chip terminations, simplifying board designs.
- Block RAM provides data storage in the form of 18-Kbit dual-port blocks.
- Digital Clock Manager (DCM) blocks provide self-calibrating, fully digital solutions for distributing, delaying, multiplying, dividing, and phase shifting clock signals.

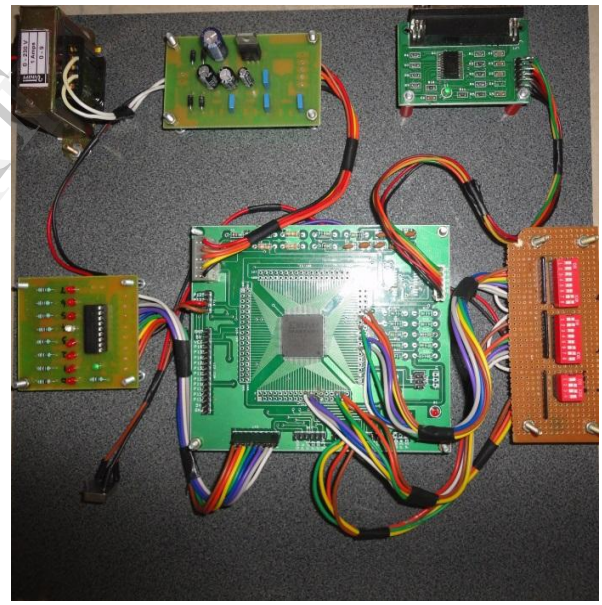

Figure 2 Spartan – 3 FPGA

III. Instruction Sets for RISC Processor

An instruction set, or instruction set architecture (ISA), is the part of the computer architecture related to programming, including the native data types, instructions, registers, addressing modes, memory architecture, interrupt and exception handling, and external I/O. The instruction set describes an abstract version of a processor.

Table 1 shows the instruction set for RISC processor.

TABLE I INSTRUCTION SETS

Instruction	Opcode	Operation performed
OR	0000	OR operation of two registers
AND	0001	AND operation of two registers
NAND	0010	NAND operation of two registers
NOR	0011	NOR operation of two registers
XOR	0100	XOR operation of two registers
XNOR	0101	XNOR operation of two registers
ADD	0110	ADD operation of two registers
SUBTRACT	0111	SUBTRACT operation of two registers
NOT	1000	NOT operation
INCREMENT	1001	Increment the value by 1
DECREMENT	1010	Decrement the value by 1
DCT	1011	Perform DCT Operation
DFT	1100	Perform DFT Operation
FFT	1101	Perform FFT Operation

IV. INSTRUCTION FORMAT

The RISC machine fetches an instruction from the memory. Each instruction decodes by internal decoder and the value of each instruction is 20 bits. In those 0 to 3 bits is the opcode which decide the operation to be performed.

TABLE II INSTRUCTION FORMAT for INPUT

R[y]		R[x]		OPCODE	
19	12	11	4	3	0

Table no 2 shown the instruction format (input) for RISC processor. [5]

The instruction format for output is shown in Table no 3.

TABLEIII INSTRUCTION FORMAT for OUTPUT

OUTPUT	
7	0

V. OPERATIONS

A) Discrete Fourier Transform

It is a kind of Discrete Transform which is used in Fourier analysis. It transforms one function into another, which is called the frequency domain representation, or simply the DFT, of the original function. The formula for DFT [2] is

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{i2\pi kn}{N}} \quad (k = 0,1, \ldots . N-1)$$

B) Fast Fourier Transform

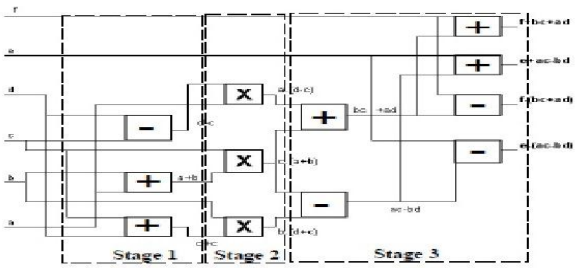

Fig. 3 Montium Butterfly Mapping

Fig. 3 demonstrates the efficiency of the hardware architecture [11]. FFT is an efficient algorithm or fast way to compute a DFT. Radix-2 Decimation-in-time (DIT) Fast Fourier Transform (FFT) is dividing the DFT in to two portions. Using a complex multiplier operation in combination with the flexibility of the Montiumdatapath, it is possible to implement an FFT/IFFT butterfly in a single clock cycle using only 4 arithmetic logic units (ALUs).

C) Discrete Cosine Transform

A Discrete Cosine Transform (DCT) expresses a sequence of finitely many data points in terms of a sum of cosine functions oscillating at different frequencies. The N-point 1-D DCT is defined as

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u, v = 0 \\ 1 & \text{otherwise} \end{cases}$$

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} f(x,y) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2x+1)v\pi}{2N}$$

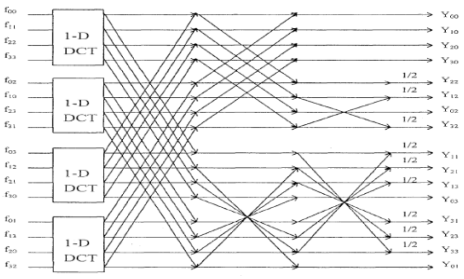

Fig. 4 Butterfly Diagram

Computing the transform directly from the N x N input numbers

- Derive fast DCT algorithms from the signal flow graph (like FFT)
- Based on 1-D DCT
- Larger flow graph

- Global routing
- More temporal storage
- Larger data path

The Figure 4 shows the implementation of the 2-D DCT Butterfly diagram. [7]

D) Ripple Carry Adder

The multiple full adders are used with the carry ins and carry outs chained together then this is called a ripple carry adder because the cout value of the carry bit ripples from one bit to the next. The block diagram of 8-bit Ripple Carry Adder is shown here below in Fig. 4.

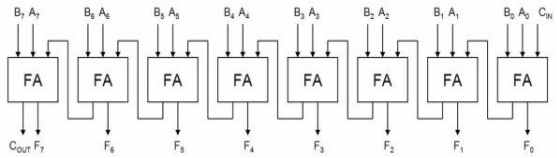

Fig. 5 Ripple Carry Adder

It is possible to create a logical circuit using several full adders to add multiple bit numbers. Each full adder inputs a Cin, which is the Cout of the previous adder. This kind of adder is a Ripple Carry Adder, since each carry bit "ripples" to the next full adder. [14]

E) Carry Save Adder

The Straight forward way of adding together m numbers is to add the first two, then add that sum to the next, and so on. This requires a total of m-1 additions, for a total gate delay of O (m log n). 0

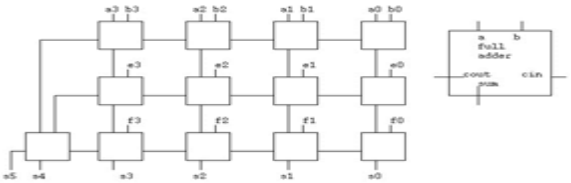

Fig. 6 Ripple Carry Adder

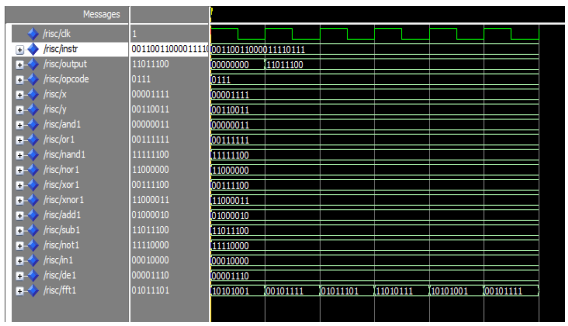

Fig. 6 Simulation result for all opcodes

The basis block for 4-bit carry save adder is shown in figure 6. Carry – Save adders are based on the idea that a full adder really has three inputs and produces two outputs. [14]

VI. SimulationAndResult

All the instructions is simulated correctly and the results are shown in Table no. 4. The simulation result and synthesized using Xilinx ISE version 13.2. The Simulation results is shown in Figures. The Fig. 6 shows the Simulation Result for all opcodes.

TableIV Result

Operation	Input			Output
	X	Y	Opcode	
AND	11011011	00111011	0000	00011011
OR	11011011	00111011	0001	11111011
NAND	11011011	00111011	0010	11100100
NOR	11011011	00111011	0011	00000100
XOR	11011011	00111011	0100	11100000
XNOR	10100000	000000000	0101	11111001
ADD	00000110	00000000	0110	00000110
SUBTRACT	00000111	00000000	0111	11111010
NOT	00000101	00000000	1000	11111010
INCREMENT	00000001	00000001	1001	00000110
DECREMENT	00000000	00000000	1010	00000101
DCT	01011101	00000000	1011	00000011
FFT	00110000	00000000	1101	00000001

The value of input X is 11011011 and input Y is 00111011; the instruction for 0000 is AND, for AND operation output result is 000110011.

The value of input X is 11011011 and input Y is 00111011; the instruction for 0001 is OR, for OR operation output result is 11111011.

The value of input X is 11011011 and input Y is 00111011; the instruction for 0100 is XOR, for OR operation output result is 11100000.

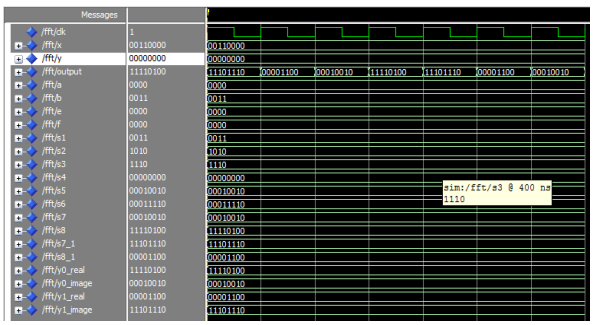

Fig. 7 Simulation result for FFT

The value of input X is 00000110 and input Y is 00000000; the instruction for 0110 is ADD, for ADD operation the instruction is X+Y and the output result is 00000110.

The FFT operation is performed by using the Fig. 3 is performed and the simulation result is shown in fig. 7.

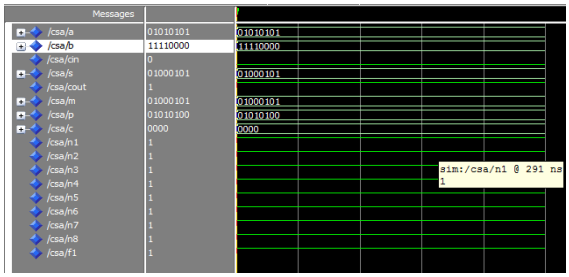

Fig. 8 Simulation result for Carry Save Adder

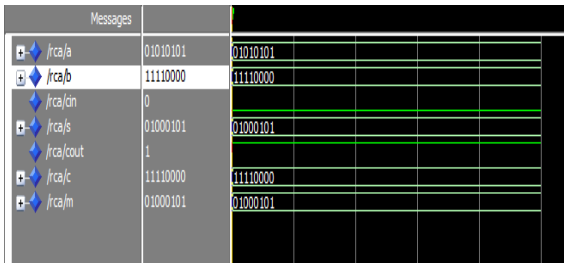

Fig. 9 Simulation result for Ripple Carry Adder

The Carry Save Adder and Ripple Carry Adder operations are performed by using the Fig. 5 and 6 respectively. The simulations result is as shown in fig 8 and 9 respectively. The simulated output using the Spartran 3 FPGA is as shown in figure 10.

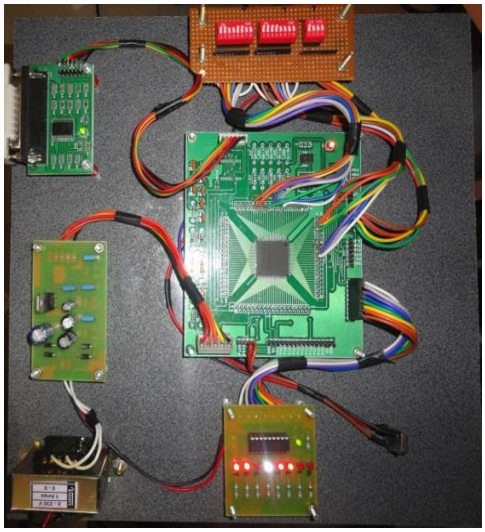

Fig. 10: Simulated output using FPGA Kit.

VII. Conclusion

Thus the simulation and result of this 20-bit RISC processor provides the various features including arithmetic operations and the DSP operations. This design is used in various areas such as android phones. The processor has been designed for executing the instructions of 14 operations in total. The design implemented can be easily implemented using VHDL and simulated with the Xilinx. The value of output and input bit is easily upgraded by increasing the memory of the processor and can be implemented with higher bit values. This RISC processor executes all the instructions in one clock cycle, including jumps, returns from subroutines and external accesses.

Acknowledgment

It gives me immense pleasure to thank the anonymous reviewers for their constructive comments and suggestions.

References

[1] Amit Kumar Singh Tomar, Prof. Rita Jain, "*Implementation of RISC System in FPGA"*, IJETAE, ISSN 2250-2459, Vol 2, Issue 9, September 2012.
[2] Ryszard Gal, Adam Golda, MaciejiFrankiewicz, Andrezej Kos, "*FPGA Implementation of 8-bit RISC Microcontroller for Embedded System"* MIXDES, 323-328, 2011.
[3] LI Xiao-feng, Chen Long, Wang Shihu, "*The Implementation of High-speed FFT Processor based on FPGA",* IEEE, 978-1-4244-7956-6/10, 2010.
[4] AsmitaHaveliya, "*Design and Simulation of 32-Point FFT Using Radix-2 Algorithm for FPGA Implementation",* IEEE, 978-0-7695-4640-7/12, 2012.
[5] Luker, Jarrod D., Prasad, Vinod B, "*RISC System Design I FPGA"*, MWSCAS 2001, vol. 2, pp 532-536, 2001.
[6] M. Vijaya Kumar, M. Vidhya, G. Sriramulu, "*Design and VLSI Implementation of A Radix-4 64 – Point FFT Processor"*, IJRCCT, ISSN 2278-5841, Vol 1, Issue 7, December 2012.
[7] Prof. Shao-Yi Chien, Information Theory and Coding Technique.
[8] Deepak Kumar, K. Anusudha, "*RISC SYSTEM DESIGN IN XILINX", IJAREEIE*, Vol .2, Issue 4, April 2013.
[9] SagarBhavsar, AkhilRao, AbhishekSen, Rohan Joshi, "*A 16-bit MIPS Based Instruction Set Architecture for RISC Processor",* IJSRP, Vol. 3, Issue 4, April 2013.
[10] Anjana R, Krual Gandhi, *"VHDL Implementation of a MIPS RISC Processor*", *IJARCSSE*, Vol. 2, Issue 8, August 2012.
[11] http://www.recoresystems.com/products/montium-reconfigurable-dsp-ip.
[12] Sneha N. Kherde, MeghanaHasamnis, "*Efficient Design and Implementation of FFT"*, International Journal of Engineering Science and Technology (IJEST), ISSN: 0975-5462 NCICT Special Issue Feb 2011.
[13] J. G. Proakis and D. G. Manolakis, " Introduction to Digital Signal Processing", New York: Macmillan, 1988.
[14] R. Uma, VLSI Design, Sri Krishna Publication.
[15] Anuruddh Sharma, MujtiAwad,"A 16-BIT RISC PROCESSOR FOR COMPUTER HARDWARE INTRODUCTION", IRACST, ISSN: 2250-3498, Vol.2, No. 3, June 2012.