Design and Implementation of RISC Processor using FPGA for Convolution Application

Purandara Babu.N#1, Manoj Kumar.K*2, Triveni.M#3

P.G. Scholar (M. Tech), Dept. of ECE, AVR & SVR Engineering College, Kurnool, A.P, India

puri.hyd@gmail.com afcatmanoj@gmail.com mrudalatriveni@gmail.com

I.V.Rameswar Reddy M.Tech rameswar.iv@gmail.com

Abstract— In this paper, we propose a 16-bit non-pipelined RISC processor, which is used for signal processing applications. The processor consists of the blocks, namely, program counter, clock control unit, ALU, IDU and registers. Advantageous architectural modifications have been made in the incrementer circuit used in program counter and carry select adder unit of the ALU in the RISC CPU core. Furthermore, a high speed and low power modified Wallace tree multiplier has been designed and introduced in the design of ALU. The RISC processor has been designed for executing 27-instruction set. It is expandable up to 32 instructions, based on the user requirements. The processor has been realized using Verilog HDL, simulated using Modelsim 6.2 and synthesized using Synopsys. Power estimation and area estimation is done using Synopsys Design Vision using SAED 90nm CMOS technology and timing estimation is done using Synopsys Primetime. In this paper, we have extended the utility of the processor towards convolution application, which is one of the most important signal processing application. The simulations depict the total dissipated power by the processor to be approximately 329.3 µW with the total area of 65012 nm^2.

Keywords— ***CISC, RISC, Convolution, Wallace Tree Multiplier.***

I. INTRODUCTION

The trend in the recent past shows the RISC processors clearly outsmarting the earlier CISC processor architectures. The reasons have been the advantages, such as its simple, flexible and fixed instruction format and hardwired control logic, which paves for higher clock speed, by eliminating the need for microprogramming. The combined advantages of high speed, low power, area efficient and operation-specific design possibilities have made the RISC processor ubiquitous.

The main feature of the RISC processor is its ability to support single cycle operation, meaning that the instruction is fetched from the instruction memory at the maximum speed of the memory. RISC processors in general, are designed to achieve this by pipelining, where there is a possibility of stalling of clock cycles due to wrong instruction fetch when jump type instructions are encountered. This reduces the efficiency of the processors. This paper describes a RISC architecture in which, single cycle operation is obtained without using a pipelined design. It averts possible stalling of clock cycles in effect [1]-[3].

The development of CMOS technology provides very high density and high performance integrated circuits. The performance provided by the existing devices has created a never-ending greed for increasingly better performing devices. This predicts the use of a whole RISC processor as a basic device by the year 2020. However, as the density of IC increases, the power consumption becomes a major threatening issue along with the complexity of the circuits. Hence, it becomes necessary to implement less complex, low power processor designs.

Energy recovery is proving to be a promising approach for the design of low power VLSI circuits. In recent years, studies on adiabatic computing have grown for low power systems and several adiabatic logic families have been proposed [4]-[6]. In this regard, we have utilized 2N-2N2P quasi-adiabatic logic in the design of incrementer circuits and carry select adder circuits along with architectural changes, in order to prove the power efficiency of the proposed structures with those found in the literatures.

Program counter is one of the most complex building blocks of the processor design. It performs mainly two operations, namely, incrementing and loading. In order to address this issue, the present work establishes a novel design of an incrementer structure [7]. It is realized using the quasi-adiabatic 2N-2N2P logic structure. The structure incurs 17 times reduction in power, while comparing against conventional CMOS counterpart. A speed increase of about 25% and a marginal amount of 8% area saving are achieved.

The second part of this work concentrates on the complexity reduction in ALU by optimizing the design of arithmetic circuits. The previous works in literature focus on energy efficient arithmetic circuits. In order to increase the operating speed and power efficiency of the processor, we have come out with a novel design of a carry select adder structure [8]. This has also been compared with the conventional adders to validate the design in terms of power-delay product.

In order to employ the processor for signal processing applications, we have integrated a modified Wallace tree multiplier that uses compressor circuits to achieve low power, high speed operation [9], in the ALU. Reference [10] suggests that it is possible to achieve the high speed, low power and area efficient operations by reducing the stronger operations such as multiplication, at the cost of increasing the weaker operations such as addition.

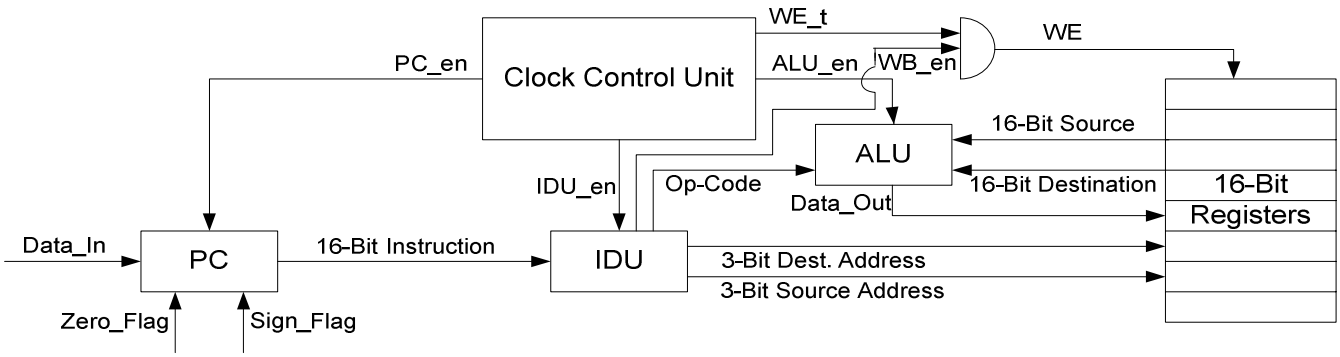

Fig.1 16-bit Non-Pipelined RISC Processor

Convolution is an important signal processing application which is used in filter design. Many algorithms have been proposed in order to achieve an optimized performance of the filters by optimizing the convolution design. Modified Winograd algorithm is notable among them. They require 4 multiplication operations only, when compared to 6 for a 3*2 normal convolution methodology with an additional rise in the number of adders to 9 from 4.

In this work, we have designed and developed a 16-bit single cycle non-pipelined RISC processor. In order to improve the performance, modification on incrementer circuit and carry select adder circuit have been done and modified structure has been integrated into the design and the performance is validated. A multiplier structure has been developed and modified Winograd algorithm is executed in order to validate our claim.

The rest of the paper is organized as follows. Section II presents the design of the RISC CPU. Section III presents the implementation of modified Winograd convolution algorithm. Section IV gives the ASIC implementation results and analysis. Section IV concludes.

II. DESIGN OF 16-BIT RISC CPU

A. Architecture

The architecture of the proposed RISC CPU is a uniform 16-bit instruction format, single cycle non-pipelined processor. It has a *load/store* architecture, where the operations will only be performed on registers, and not on memory locations. It follows the classical von-Neumann architecture with just one common memory bus for both instructions and data. A total of 27 instructions are designed as a first step in the process of development of the processor. The instruction set consists of *Logical, Immediate, Jump, Load, store* and *HALT* type of instructions. The Halt instruction acts as a border line between the instruction and data memory. This offers the flexibility to the programmer, who uses this processor core to define their own instruction and data memory within the allotted 64 memory registers. Each of the register is of 16-bits width capacity. The bit widths of each unit are as follows.

Instruction Unit	: 16 bits
Execution Unit	: 8 bits
Memory Unit	: 16 bits
Op-code Width	: 5 bits.

B. Detail of Logical Blocks

Figure 2 illustrates the block diagram of the 16-bit RISC CPU. The proposed RISC CPU consists of five blocks, namely, Arithmetic and Logical Unit (ALU), Program Counter (PC), Register file (REG), Instruction Decoder Unit (IDU) and Clock Control Unit (CCU). The data-path of the proposed CPU in Fig. 1 is explained as follows.

1) Program Counter: The Program Counter (PC) is a 16-bit latch that holds the memory address of location, from which the next machine language instruction will be fetched by the processor. The proposed PC is the largest sub-block and second to the control unit in complexity. It controls the flow of the instructions execution and it ensures the logical operation flow of the processor. It performs the two operations, namely, incrementing and loading. For most instructions, the PC is simply incremented in preparation for the following instruction or the following instruction nibbles. In general, a normal conventional adder circuit will be used for incrementing action. However, it leads to increased hardware use along with more power dissipation. Hence, this work strives for a low power and novel incrementer circuit design.

In this design, we employ a 6-bit pointer to indicate the instruction memory. It additionally uses a 6-bit pointer to point to the data memory, which will be used only when a Load/Store instruction is encountered for execution.

2) Arithmetic and Logic unit: The arithmetic and logic unit (ALU) performs arithmetic and logic operations. It also performs the bit operations such as *rotate* and *shift* by a defined number of bit positions. The proposed ALU contains three sub-modules, viz. arithmetic, logic and shift modules.

The arithmetic unit involves the execution of addition operations and generates Sign flag and Zero flag as per the result shown in the process. In order to reduce the complexity of the adder circuits used in the arithmetic unit of the RISC CPU, a very fast and low power carry select adder circuit has been introduced. The ALU also consists of a modified Wallace tree multiplier, which uses compressor circuits to achieve low power and improved speed of operation. The multiplier is designed to execute in a single cycle. Hence, it satisfies the requirement of the RISC design, to execute single cycle instructions.

The shift module is used for executing instructions such as rotation and shift operations. The shift module is mandatory for signal processing applications, which needs division by 2. This is achieved by a single right shift operation. The logic unit is used to perform logical operations, such as, Ex-or, OR, and AND. The Data out of each ALU operation is written back into the corresponding destination register, along with the flags updated. In order to maintain simplicity of the design, the carry out of the ALU is not taken into consideration.

3) Register File: The register file consists of 8 general purpose registers of 16-bits capacity each. These register files are utilized during the execution of arithmetic and data-centric instructions. It is fully visible to the programmer. It can be addressed as both source and destination using a 3-bit identifier. The register addresses are of 3-bit length, with the range of 000 to 111. The load instruction is used to load the values into the registers and store instruction is used to retrieve the values back to the memory to obtain the processed outputs back from the processor. The Link register is used to hold the addresses of the corresponding memory locations.

4) Instruction Decoder Unit: Our instruction set is limited yet comprehensive. Since our data bus is only 5 bits wide, it was decided to keep the number of instructions supported within 32 for easier implementation. At present, only 27 instructions have been implemented. The rest have been reserved for porting digital processing applications into our processor. The decoder units decodes the instruction and gives out the 3-bit source and destination addresses respectively, depending on the op-code's operation and it also decides whether the write-back circuit has to be enabled or not.

In case of Load/Store instructions, the IDU updates the Link register. In case of Jump instructions, if the conditions are satisfied, the IDU updates the PC register with the new address from where the next instruction has to be retrieved rather than the normal incremented value.

Figure 2(a) shows the instruction format followed by Logical instructions and Data transfer instructions, such as, MOV, AND, OR, XOR, ADD, SUB, SL (Shift Left), RL (Rotate Right), SR (Shift Right), RR (Right Rotate), SWAP and Multiply instructions. Fig. 2(b) shows the instruction format followed by Immediate instructions. This type has the data vested into the instructions, such as LHI (Load 8-bit value into the eight higher significant bits of the given register), LLI (Load 8-bit value into given register's 8 least significant bits), ANDI, ORI, XORI, ADDI, SUBI.

Figure 2(c) depicts the format of Load instruction. The instruction format for Store instruction is given in Fig. 2(d). Fig. 2(e) depicts the instruction format for JUMP, JZ (Jump if Zero), JNZ (Jump if not Zero), JP (Jump if positive), JN (Jump if Negative) instructions. Fig. 2(f) shows the format for the HALT command.

5) Clock Control Unit: Efficient phase scheduling is required to optimize the throughput and the energy consumption of the processor. In this paper, we propose a clock control unit (CCU) which is tasked with efficient phase scheduling, to select the various blocks of the processor.

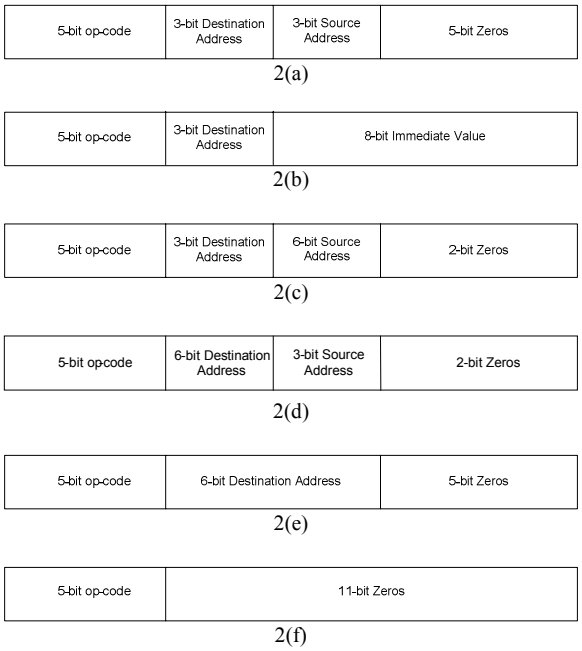

Fig. 2 (a) to (f) 16-Bit Instruction Format

III. Implementation of Modified Winograd Convolution Algorithm

To describe the functionality of the processor towards its use for signal processing applications, we have executed a 3*2 modified Winograd algorithm by implementing its pneumonic code as shown below:

1. Load x0
2. Load x1
3. Load x2
4. Mov x2
5. Add x0, x2
6. Mov x1
7. Add (5), x1
8. Sub (5), x1
9. Mov Reg D to Reg E
10. Load H0
11. Load H1
12. Load H2
13. Load H3
14. Mul H0, X0
15. Mul H1, X1
16. Mul H2, X2
17. Mul H3, X3
18. SR (13) gives S0
19. SR (16) gives S3
20. Add (16), (17)
21. Sub (20), (15)
22. SR (21) gives S1
23. Add (15), (16)
24. Sub (14), (15)
25. SR (24) gives S2
26. HLT

The respective op-codes are initially stored in the instruction memory of the processor. The inputs are stored in

the corresponding data memory, which should, in this case lie beyond the memory location 26. The maximum size of inputs x0, x1, x2 can each be of 6 bits width, while the values of h0 and h1 can each be of 3 bits width.

The execution of the program gives the required results S0, S1, S2, and S3 after 26 clock cycles, thus validating the single cycle instruction execution of the processor. It also proves its use in a typical signal processing application. Repeated execution of multiplication and addition instructions can be utilized in order to execute the MAC operation, which is widely used in signal processing applications. It would thus eliminate the need for a dedicated MAC unit.

IV. ASIC Implementation and Results

The RISC processor described above is designed using Verilog HDL and is simulated using Modelsim 6.2. The proper functioning of the processor is validated. The simulation result shows that the processor is capable of implementing the given instruction in single clock cycle, thereby satisfying the basic requirements of the RISC processor.

In order to evaluate the system performance, usage of synthesis software were used to map the proposed processor on a target library. The target library includes the generic technology mapping information. The tool employed for mapping the Verilog-HDL components to the cell library is the Synopsys design compiler. It mapped the Verilog-HDL components to a SAED 90nm ASIC standard cell library.

The power dissipation report produced by the design compiler is shown in Table I below. The total power dissipation was found to be *7.44mW*. In order to achieve reduced power consumption, a low power design technique called clock gating was employed. To save power, clock gating technique adds more logic to the circuitry to prune the clock tree, thus disabling portions of the circuitry so that its flip-flops do not change state unnecessarily. This technique reduces the power consumption to *3.04mW*. The application specific power dissipation still gets reduced to *6.62μW*, when the switching activity file is given as input to the power compiler.

The design compiler report depicting the area consumed by the processor shows the total area occupied by the processor to be *79264.49 nm*^{2}. The clock gating technique still reduces this area to be around *65012.356 nm*^{2}.

The primetime tool of Synopsys is used to measure the timing results of the processor. The constraint provided for the processor is 5ns (200 MHz), and the slack is found to met for both setup time and hold time. The critical path value indicates that the processor can run efficiently at a speed of *200 MHz*.

The application specific processor's specifications have thus found to have:

Speed : 200 MHz
Area : 65012 nm^2
Power Dissipation : 329.3 μW

TABLE I

Power Dissipation

Cell Internal Power	51.1165 nW
Net Switching Power	10.128 nW
Total Dynamic Power	61.2445 nW
Cell Leakage Power	329.2979 μW
Total Power dissipation	*329.3 μW*

V. Conclusions

The design of a single cycle 16-Bit non-pipelined RISC processor for its application towards convolution application has been presented. Novel adder and multiplier structures have been employed in the RISC architecture. The processor has been designed for executing the instruction set comprising of 27 instructions in total. It is shown expandable up to 32 instructions, based on the user requirements. The processor design promises its use towards any signal processing applications.

Acknowledgment

We place our gratitude on record to the Department of Electronics and Communication Engineering, SSN College of Engineering, Rajiv Gandhi Salai, Chennai for the support rendered to us in carrying out this work.

References

[1] Robert S. Plachno, VP of Audio "A True Single Cycle RISC Processor without Pipelining". ESS Design White Paper – RISC Embedded Controller.

[2] Youngjoon Shin, Chanho Lee, and Yong Moon, "A Low Power 16-Bit RISC Microprocessor Using ECRL Circuits", ETRI Journal, Volume 26, Number 6, December 2004.

[3] Yasuhiro Takahashi, Toshikazu Sekine, and Michio Yokoyama, "Design of a 16-bit Non-pipelined RISC CPU in a Two Phase Drive Adiabatic Dynamic CMOS Logic," International Journal of Computer and Electrical Engineering, Vol. 1, No. 1, April 2009 1793-8198.

[4] V. B. Saambhavi and V. S. Kanchana Bhaaskaran, A 16-Bit RISC Microprocessor Using DCPAL Circuits. International Journal of Advanced Engineering and Technology (IJAET), E-ISSN-0976-3945, Vol.II, Issue I, January-March 2011, pp. 154-162

[5] J.S. Denker, "A Review of Adiabatic Computing," IEEE Symp. Low Power Electronics, 1994, pp. 94-97.

[6] H. Mahmoodi-Meinnand, A. Afzali-Kusha, and M. Nourani, "Adiabatic Carry Look-Ahead Adder with Efficient Power Clock Generator," IEEE Proc., vol. 148, 2001, pp. 229-234.

[7] K. Nishimura, T. Kudo, and H. Amano, "Educational 16-bit microprocessor PICO-16," Proc. 3rd Japanese FPGA/PLD design conference and exhibit (Japanese Edition), Tokyo, July 19–21, 1995, pp. 589–595.

[8] Samiappa Sakthikumaran et al., "A Very Fast and Low Power Incrementer and Decrementer Circuits", International Journal of Computer Communication and Information System (IJCCIS) Vol2. No.1 – 2011, pp. 200-203.

[9] Samiappa Sakthikumaran et al., "A Very Fast and Low Power Carry Select Adder Circuits", 3rd International Conference on Electronics Computer Technology - ICECT 2011.

[10] Samiappa Sakthikumaran et al., "A Novel Low Power and High Speed Wallace Tree Multiplier for RISC Processor", 3rd International Conference on Electronics Computer Technology - ICECT 2011.

[11] Keshab K.Parhi, *VLSI Digital Signal Processing Systems*, Wiley India Edition,1999.