Retraction

Retraction: Reduced Instruction Set Computer (RISC): A Survey (J. Phys.: Conf. Ser. 1916 012040)

Published 23 February 2022

This article (and all articles in the proceedings volume relating to the same conference) has been retracted by IOP Publishing following an extensive investigation in line with the COPE guidelines. This investigation has uncovered evidence of systematic manipulation of the publication process and considerable citation manipulation.

IOP Publishing respectfully requests that readers consider all work within this volume potentially unreliable, as the volume has not been through a credible peer review process.

IOP Publishing regrets that our usual quality checks did not identify these issues before publication, and have since put additional measures in place to try to prevent these issues from reoccurring. IOP Publishing wishes to credit anonymous whistleblowers and the Problematic Paper Screener [1] for bringing some of the above issues to our attention, prompting us to investigate further.

[1] Cabanac G, Labbé C and Magazinov A 2021 arXiv:2107.06751v1

Retraction published: 23 February 2022
Reduced Instruction Set Computer (RISC): A Survey

Malti Bansal¹, Harsh¹
¹Department of Electronics and Communication Engineering, Delhi Technological University, Delhi - 110042, India
¹maltibansal@gmail.com

Abstract. Today’s modern machines are designed to process real time problems and to do so designers try to make them more performance efficient but while doing this, the complexity of design also increases. So, to maintain a balance between both we have to manage this complexity. Now, a question arises how can we do this? The answer resides in our own life. For instance, when we have a complex task to do, we often break it down into simpler sub-tasks and then work upon them individually. How well we are able to do this depends upon the efficiency of our brain. Same analogy we can apply to design processor architecture. To further improve the efficiency, RISC methodology can be proven to be a prominent choice. The RISC is a design methodology which plays an important part in modern embedded systems. From our day-to-day applications like mobile phones to some of the world’s fastest supercomputers like FUGAKU, all are based on RISC architecture. With the advancements in the scaling field and reduction in the price of the ICs, the market starts filling with the RISC based processors. This paper is focused on providing a basic idea about RISC Architecture based processor design and laying out a path that one can follow to design their own RISC processor. Also, Power outages in ICs are the major problem in the VLSI industry so various techniques and methods are also discussed that can be used for optimizing power and improving speed. Some of the major applications are also discussed which provide an idea about the extent of applications, RISC based architecture can be designed.

Keywords: RISC, MIPS, ARM, Clock Gating, Wallace Tree Multiplier, ALU

1. INTRODUCTION

To design a machine specific to an application, we know that it has some inputs and according to these inputs we want the appropriate outputs. If the machine is very small then we can hardwire it but for large machines we require a processor which will process the inputs with efficiency and provide outputs without any error. Also, it should have the capability to handle the runtime interrupts. So, by definition, we can say that the processor is a FSM that process commands stored in memory. The status of a program is interpreted by the data stored in the memory and the data stored in the registers that are located within the processor itself. Each command explains how the whole situation should change and also explains what command should be given next [1]. Now to design a processor, we should know what kind of instructions we are going to pass and according to it we will design each unit. For example, if we want to perform an addition then flow should be like fetch the instruction from memory -> decoding of fetched commands -> generate execution signals like enable the arithmetic unit and pass operand to it -> perform operation -> store the result in memory -> fetch next instruction. So, according to the above path we have to design some units like program counter to fetch instruction form memory, a decoder to decode and so on. So, the conclusion is we require a basic design/structure/architecture one can follow to design its own processor with modifications and advancements according to the application. Reduced Instruction Set Computer commonly can be used as a basic architecture for designing a processor.
because of its many advantages over other architecture. RISC uses load and store operation only to get instructions and data from memory. The instructions other than memory dependent are based on registers. This leads to instruction set architecture know as ISA more understandable and easier as we do not require to access memory every time also it makes the processing of commands to be done one command per clock cycle which increase the speed [2]. Simple addressing modes allows faster addressing computation. Because Reduced Instruction Set Computer based processors practice register to register based commands, most instructions use register-based addressing. Another major advantage is large register set. A huge register set provides number of operations for compiler to minimize the performance and it also reduce the problem of associated with procedure and call return. Instruction format in RISC is also easier as compare to other architecture. All the above properties lead to the improvement in speed and reduction in complexity which are the main aspect that a designer wants to improve to get better efficiency. This paper includes the detailed analysis of research papers over 5 five years, this will help to understand the basic principle of RISC design methodology from scratch and it also includes different techniques/methods that one can refer in designing like 5 stage pipelining, modified ALUs, clock gating technique and more. All these techniques lead to the better performance of processor. This paper also includes some major applications to get a broader view of RISC based designs. The flow of this paper is: Section 2 presents the history of RISC design and how it achieves its popularity. Section 3 presents the basic principle of RISC design. Section 4 includes different techniques to get better performance of RISC based processor. Section 5 presents current going RISC based applications and future trends. Sections 6 presents the conclusion and references.

2. HISTORY OF RISC DESIGN

In the 1970s and early 1980s, Complex Instruction Set Computers commonly known as CISC design dominated the market as they use complex instructions. But the question arises why designers prefer to use such complex design? The reason is that memory and processor are most valuable resources of computer system in those days. In fact, the cost of 16KB memory was about 500$. This makes designer to think of a design that works on high-density codes i.e., each instruction assigned to do more work in order to reduce the size of program. But along this, complex instructions increase the complexity of hardware. In early 1950s, Wilkes proposed a micro programmed control which act as an interpreter between ISA and hardware. The micro program executes the complex instruction in a sequence of small and simple commands which do not need any complex hardware. Also using different micro program, variation of ISA can be implemented this leads to the reduction of interpretation space between machine language and upper language. Now a question arises in mind, what designers motivate to think of RISC methodology? So, the answer is that CISC is developed to make processors more efficient and cost effective under the constraints of small and expensive memory. But as the time passes advancements in technologies takes place with respect to software and hardware. These type of motivation makes designer to think of an alternative of CISC design. In the year of 1980, Patterson and Ditzel published a paper entitled “The Case for the Reduced Instruction Set Computer”. In their work, they pass on the statement that power and performance efficient architectures of multi-chip processor may not be same as that of single chip processor. In University of California, Berkeley, a postgraduate was working on a project in which he is designing a processor. In this project he also includes the RISC architecture. The results of his project prove the statement passed by Patterson and Ditzel. This designed architecture is known as the Berkeley RISC I. The complexity in the designing of RISC is much less than that of CISC processor. The performance delivered by the RISC is similar but the effort requirement is very much reduced in comparison with CISC. The key features of proposed RISC I architecture were consists of huge number of register bank; each register is of 32-bit. The load and store architecture are used for memory access commands. Instruction’s size is fixed with 32-bit unlike the instructions in CISC and the formats in RISC are much less as compare to that of CISC. With all the above-mentioned features, RISC design easily earned the coin having power efficient and price as its two faces. Because of features mentioned above, RISC established its place in market rapidly and after that many companies start manufacture their own processor based on RISC methodology according to their applications. ARM
was the first company which uses the RICS concept as its base for designing its own processors. After that, many companies start using RISC at large or low extend for their benefits Figure 1.

Figure. 1: Timeline of RISC based instruction set architectures

3. REDUCED INSTRUCTION SET COMPUTER (RISC)

The finest approach to perceive the RISC design is to consider it as an idea in spite of framing it as an architecture. Moreover, earlier RISC processors had a smaller number of instructions set but the modern-day RISC based processors has thousands of instructions, many are more complex than that of CISC processors but all the RISC based processors follow some certain principals which are discussed below:

3.1 SIMPLE INSTRUCTION SET ARCHITECTURE

The main aim of RISC methodology is to design simple instruction set which can run in single cycle. This property simplifies the design and makes processor faster and more efficient. The complexity of an instruction also affects the critical path delay. Frequency of operation or in simple words speed of a processor is a very important parameter which can easily be calculated from the concept of critical path. The value of critical path should be small [3]. Simple instruction set eliminate the need of microcode and hardwired operations. To store these instructions, separate cache memory will increase the overall execution efficiency.

3.2 LOAD AND STORE ARCHITECTURE

This architecture is used to run commands in which we have to access memory and in rest of commands we use internal register bank for data transfer. This leads to the increase in speed as each instruction completes in one cycle. Unlike CISC which have many types of operations only increases the complexity in most of the cases as only memory to register and register to register are widely used instructions in any processor. Also, we can use this simple format to design complex ones.

3.3 SIMPLE ADDRESSING MODES

The architecture of RISC methodology includes simple addressing modes. Two of them are register and memory addressing respectively. As we know RISC includes huge number of registers; each of fixed length. All these large register set are used for register addressing mainly to transfer different variable values from one function unit to other. Only the instructions based in loading and restoring uses memory addressing. Another advantage of a large register set is that we can reduce further associated call and refund procedures. To speed up the process calls, we can use registers to keep local dynamics and transfer issues. Local variation is only achieved through the process in which it is declared. This mutation occurs during the pleading process and dies when the process exits.

3.4 INSTRUCTION FORMATS WITH SPECIFIED LENGTHS

The most valuable advantage of using instruction formats of specified lengths is reduction in commands run time. As all instructions are of same size, compiler doesn’t need to check the size of command every time and distinguish them according to it. Also, all the opcode generated are of same size. All this makes the decoding stage to be done in more efficient manner. the lagging also reduces because of fixed length. Now, we will discuss two architectures namely, MIPS (Microprocessor without Interlocked Pipelined
Stages) and ARM (Acorn RISC Machine) both are based on RISC methodology to get more clear vision on RISC.

**Microprocessor without Interlocked Pipelined Stages (MIPS)**

Microprocessor without Interlocked Pipelined Stages is a general-purpose architecture whose main goal is to provide high performance while executing the compiled code. The basic philosophy of MIPS is to introduce a set of instructions which is a computer-operated small engine computer. Therefore, little or no specificity is required and the commands are very consistent with the minor code commands. The processor has a pipe but does not provide the hardware for the pipe connection; this function must be provided by the software. It has grown significantly in research conducted at Stanford. Along with the work of Berkeley RISC, these projects have influenced the development of the processor in relation to the structure of RISC. MIPS is very important for installed applications including digital cameras, digital TVs, Sony PlayStation 2, network routers, and so on. In 1990s, MIPS has legalized their design around two basic structures: MIPS32 and MIPS64. MIPS follow store / store design to provide memory functionality as RISC processors. This structure provides a wide range of upload and store instructions for data transfer of various sizes: byte, halfword, and so on. The development of MIPS32 provides 32 general purpose registers, Program Counter (PC), and two special purpose registers. In the language of the meeting, these registers are identified as $0, $1, ..., $31. Two general purpose registers - the first and the last - are reserved for a specific task: Register $0 is used for zero value and the final $31 register is used as a connection register for the connection and connection order, which is used to request the process. Apart from this, two special purpose registers - called HI and LO - are used to capture the results of number multiplication and division of instructions. MIPS, being a RISC processor, uses a limited format teaching format. Each command is 32 cubits long. It uses three different teaching formats: Instant (Type), Jump (J-type), Register (R-type). The use of a limited number of command formats facilitates command planning. However, three instructional formats and one speaking mode mean that complex operations and coping strategies will need to be integrated by the compiler. If these services and local approaches are not widely used, we may not be able to pay the full penalty. This is an encouragement from RISC processors.

**Advanced RISC Machine (ARM)**

ARM development started following previous RISC machines of that time which were Stanford developed MIPS and Berkeley’s RISCs. Before these architectures, some designs had shared many properties which were then linked to RISC, like IBM, Cray-I and Digital PDP-8. Berkeley RISC was chosen as a precursor to ARM as it took several features from it like 32-bit instructions of fixed length, load and store architecture and three address instruction formats. It also rejected few of them like not executing all instructions in a single clock cycle. Also, ARM doesn’t use delayed branching approach since it dismantles the atomicity of each individual instruction. Furthermore, ARM doesn’t use large Register Windows like Berkeley RISC does since it increases chip area and hence drives up the cost.

**4. TECHNIQUES**

In today’s fast-growing technology, processing of real time tasks is the main concern and to do so complex processors are required. To design such processors, we have two approaches; first is that we design a processor from scratch and second is that we modify available processor architecture by including performance enhancement techniques. Moreover, these techniques help in increasing the functional capabilities but along this it also increases the power requirements of a processor which in turn makes a problem so to balance a trade-off between both low power techniques are also necessary. Discussed below are some most efficient methods that one can follow in order to design a processor.

**4.1 FIVE STAGE PIPELINE ARCHITECTURE**
Pipelining, a standard feature of RISC architecture general purpose processor is an efficient course of action of utilizing concurrency. It is defined as a technique in which the processor executes the current instruction has finish before the next instruction. It is a worthwhile way of improving the usage of hardware resources. Now, consider an analogy of laundry to understand the basic concept of pipelining. Let’s say we have three clothes need to be washed, dried and folded, all three tasks take 10 minutes to complete. To complete the whole process, one way is that we take first cloth perform all three tasks which will take around 30 minutes. Then take second cloth and repeat all three tasks again. In the way it takes around 90 minutes to complete the whole process. However, this is not the appropriate approach. The smarter way would be that we put second cloth for washing after first cloth was washed and revolve in the dryer. Therefore, in the pipeline of laundry first load will be folded, after that second load would dry, and in the end 3rd load. Using this way, whole process will take only 50 minutes in total to complete. In the same way, the operation of pipeline in RISC processor take place, however depending upon the type of processor the stages of pipeline are different. Each instruction is being executed by the processor in a proper step. While different numbers of stages might have different processors, many of below mentioned in five stages sequence: Instruction Decode -> Instruction Fetch -> Execution -> Memory Back -> Write Access. So, the reason is this the why we called it 5 stage pipelining as we execute an instruction in five steps. In general, we can devise N stage pipelining by executing an instruction in N steps. More the number of stages, more efficiently the processor will work. Figure 2 and Figure 3 shows the with and without five stage pipeline MIPS architecture.
instruction decode stage but because of pipeline instruction next to branch instruction got executed which cause problem it is not decided that which instruction to be fetched next. This kind of hazard is known as Control Hazards. Amit Pandey proposed that there are two techniques for remove this hazard named as, Dynamic Branch Prediction and Static Branch Predictions [5].

4.2 ASYNCHRONOUS 5 STAGE PIPELINING

A synchronous design consists of registers which are diverse in nature which are known as concurrent pipeline. A stage register might be serving a dual behaviour comprises of transparent and opaque controlled by the clock. Transparent refers to driving of data from input to output and opaque nature leads to the hold capability of register. When clock get triggered while rest of the time the register become transparent while passing the information to the next stage, it remains opaque. With increase in the number of transistors on a single chip and decrease in transistor size, it remains blurred to makes a difficult job in maintaining a system of global synchronization within the reasonable power constraints. Clock Skew and power consumptions are the main motivation of designing a system with asynchronous pipeline feature. Mainly clock gating and skew reduction are used to overcome the problem of power and clock skew respectively but by using proposed pipeline we can fix both the drawbacks stated above. The elementary idea to achieve asynchronous pipeline is that there will be a sender and receiver communication which can be done by using request and acknowledgment signals. After placing data on data line, sender sends a request signal through control line the transmitted data is being read by the receiver. After the data successfully received a acknowledgement has been send back the receiver. As there is no clock present, data advancement depends on individual stages. So, the NTH stage will store the data if (N-1)TH stage gives information about the latest data to the next stage (N+1)TH stage which is stored in the preceding data. To design the circuitry based on above proposed idea, four asynchronous pipeline latches are included in ordinary 32-bit RISC architecture namely as IF/ID, ID/EXE, EXE/MEM and MEM/WB which are working according to the principal of Catch and Proceed Protocol. Then asynchronous pipeline based 32-bit RISC architecture is shown Figure 4 and Figure 5.

**Figure 4:** Asynchronous pipeline-based architecture

**Figure 5:** Capture Pass Protocol Logic

The Control Pass Protocol based latches have four inputs Capture, Capture Done (Cd), Pass (P), Pass Done (Pd). The request signal and the Pd signal are the inputs of muller element [6]. The output of muller element is the capture port of latch. The latch becomes opaque if there is a change at capture input which means the data of previous stage is stored in the latch. This leads to the toggling of capture done port. The output of Cd port act as a local clock which serves as a request signal for next latch after passing through the delay element. Delay element is added to match the worst-case path delay. The output of capture done port of next latch act as a pass signal of previous latch. As the previous latch gets the pass signal it sends the data to next latch. The pass done toggles as the data gets processed and by the word processed means successful capturing of previous stage and sending of present stage. [7]
proposed the architecture and compared it with synchronous pipeline based 32-bit RISC architecture on the pre layout and post layout power consumption aspects. The results shows that above discussed method is very much power effective. The power results show that asynchronous CPU reduces the power up to 69% compared to the synchronous CPU under the clock frequency constraints of from 100MHz to 400MHz. The area shown by the tool is 0.34 mm². Figure 6.

Figure 6: Comparative study of Power of RISC processors

4.3 DUAL PIPELINE SUPERSCALAR BASED ARCHITECTURE

[8] has proposed a 32-bit RISC processor with 5 stage, two way in order superscalar pipeline is designed. The base architecture used for innovation is RISC-V ISA. RISC-V ISA is chosen because of its upgraded features. The pipeline stages are same as in conventional pipeline system, fetch the instruction from memory -> decoding of fetched commands -> generate execution signals like enable the arithmetic unit and pass operand to it -> perform operation -> store the result in memory -> fetch next instruction. In the Fetch stage, program counter generates the program memory address. Two consecutive instructions are fetched using this program counter from instruction cache these two instructions are send to decode stage for further process. Instruction Issuing Unit (IIU) put either one or two instructions on the pipelines after checking dependencies in between the instructions. As the objective of this architecture to process two instruction at a time, so more hardware will be required in the execution stage. Like in this architecture, two integer and one floating unit are present to process two instructions at a time (two integer type or one integer and one float type). For multiple and divide based instructions, IF and ID stages are stalled as these instructions required more than one number of clock cycle. After the execution stage, results are sent to write back stage for updating of registers or they are forwarded back to execution stage according to the nature of instruction. To reduce the complexity and for proper utilization of resources, some conventions are made by designer that are important to mentions [9]:

- Two instruction which are fetch together should not have any type of dependency on each other. If this happen, only first instruction got executed and second is hold at ID stage.
- Two branch instructions or two memory access-based instruction are not executed together.
- Load, branch, multiply/divide instructions are executed only through first pipeline.

Some of the hardware that are required for the designing of two pipelining based system are Instruction Cache interface for accessing two instruction at a time. Instruction issuing unit for comparing the operands of two instructions and if some dependencies occur, first one is sent to first pipeline and second is hold in a special register. A “rollback” flag is also generated to take care of next PC value. The program counter updates its value by +4 if roll back is generated else by +8. All unit names are taken from the cited research paper [10].

4.4 Some Power Optimization Techniques
Whenever ASIC design is talked about, two names are always mentioned. These are, Power and Performance. These parameters are closely linked to each other and a trade-off exists between them. Therefore, it is the job of the designer to optimize the design for the best balance between the two. Several methods have been developed in the past decade which addresses these issues like clock gating, multiple supply voltages and multiple threshold techniques. Experimental result shows that VDD/2 can act as good candidate for second VDD. The paper studies the various combination of above methods and tries to come up with a suitable combination. These are tested on an 8-bit RISC processor.

**Clock Gating:** The most well-known method to reduce power mostly dynamic is to disable clock when unit is in no use. Clock network is one of the major reasons of dynamic power in synchronous systems. The buffers and clock tress consume a very high portion of total power of around 30% [11]. A register is updated with new data at clock edge when load signal is enabled, else it holds previous values. This means that the edge is still required even when load is not used. This wastes power and can be avoided by creating an enabled clock which is nothing but ANDing of clock and enable signal Figure 7.

![Figure 7: Schematic Diagram showing Clock Gating](image)

**Multi-Voltage:** - Since dynamic power depends on supply voltage VDD, lowering VDD saves power use. But there is again a trade-off. Low VDD also increases delay, so proper optimization needs to be done. In the given work, 1V and 0.7 V are chosen the two supply voltages.

**Multi-Threshold:** - Reducing V$_{TH}$ decreases delay as current increases but it increases subthreshold leakage current thereby increasing power dissipation. In the given work, two threshold levels are used. [12] designed a 32-bit 5 stage pipeline RISC processor using standard cell of 180nm CMOS technology in Xilinx 14.5. To get a clear view, a report of power consumption at different frequency before and after inclusion of clock gating is prepared. The frequency range is from 100MHz to 500MHz. The output shown in Table 1 indicates the comparative study of power for addition instruction at different values of clock frequency. Same study is shown in Table 2 for multiplication instruction. It is shown that 56% of total power of processor is consumed by multiplication instruction and this is because of large number of iterations as compare to addition instruction. The observation shows that consumption of power with the factor of 13.6 is done by multiplication instruction as compare to addition without clock gating. But by using clock gating this factor reduced to 6.6. This proves that clock gating technique reduce the power consumption to a large extend [13].

| FREQUENCY (MHz) | POWER (mW) |
|-----------------|------------|
|                 | WITHOUT CLOCK GATING | WITH CLOCK GATING |
| 100             | 7.50        | 5.25           |
| 200             | 13.75       | 11.00          |
| 300             | 19.23       | 18.86          |
| 400             | 24.84       | 23.10          |
| 500             | 31.45       | 30.03          |

Table 1: Power analysis of addition unit without and with clock gating
Table 2. Power analysis of multiplication unit without and with clock gating

| FREQUENCY (MHz) | POWER (mW) WITHOUT CLOCK GATING | POWER (mW) WITH CLOCK GATING |
|-----------------|---------------------------------|------------------------------|
| 100             | 92.45                           | 49.61                        |
| 200             | 200.03                          | 90.28                        |
| 300             | 282.56                          | 112.98                       |
| 400             | 374.66                          | 146.70                       |
| 500             | 451.34                          | 192.21                       |

4.5 FLOATING POINT UNIT FOR FOUR ARITHMETIC OPERATIONS

The objective of designing RISC processor is not only to achieve better performance with simplified or easy instructions but also run instructions with efficiency and optimized power. The Arithmetic and Logic Unit (ALU) is one of the most crucial unit of execution stage. So, it is very important for ALU unit to be more energy efficient. Also, by adding different functional units in ALU increases the processor’s all-over functional capability [14]. DSP, Speech and Image Processing applications require a larger dynamic range and precision than what the conventional fixed-point arithmetic offers. Here, floating point unit comes into the picture. The FPU or floating-point unit contains functionality for floating point addition/subtraction, multiplication and division. As defined in IEEE 754, a 32-bit single precision floating point number is made up of three components: -sign (1-bit), exponent (8-bit), mantissa (23-bit).

Floating Point Addition/Subtraction

To perform addition/subtraction, it is necessary the exponents must be equalized. After mantissa has been aligned, addition or subtraction is done using look ahead carry adder. Rounding must be done to the result to fit it in the bit length. Further, if special result is obtained, provision needs to be made for that like underflow, overflow, infinity, not a number etc. After all this is done, the obtained sign, exponent and mantissa bits are combined to give the final result.

Floating point Multiplication

First of all, if any of the input is zero, output should directly be put as zero. Next to get the sign of the result, the signs of the inputs are XORed. The mantissas are then multiplied by a multiplier (like radix 4 booth recoded multiplier) to get the result and truncated so that it fits the 24-bit Mantissa size fixed. To get the exponent bits, the individual exponents are added and then bias is subtracted from it. All the three-result generated are combined to get the final result. Exception messages are generated if required.

Floating point division

It is quite similar to multiplication. First, it’s checked if the divisor is zero, in that case output is infinity and if both numbers are zero, output is NAN. After this is taken care of, sign bit is calculated similar to Multiplication. Then mantissa of number to be divided is padded with zeroes to make it 48 bits and then divided by the divisor. The exponent is calculated by subtraction of exponents of the inputs and adding bias value. Normalization is done if needed by shifting the mantissa and decreasing the exponent value. The results are then combined to give the final output.

Complex Multiplication

Complex multiplications are very important part of DSP, DIP applications. It is therefore vital to have a special instruction for complex multiplication to be implemented.

It’s as follows:

Let there be two complex numbers \( X_1 = (a+jb) \) and \( X_2 = (c+jd) \) then:

\[ X_1 \times X_2 = (a+jb) \times (c+jd) = (ac-bd) + j(ad+bc) \]
This multiplication requires four multipliers. But this will lead to increased delay and power dissipation. Therefore, a new approach is proposed. Since Real part = (ac - bd) and Imaginary part = (ad + bc). This can be modified as Real part = b (c - d) + c (a - b) and Imaginary part = b (c - d) + d (a + b).

As shown above b (c - d) is used twice, so the total multiplier count can be decreased from 4 to 3. Furthermore, the real and imaginary part can be computed separately. The real part can be calculated when positive edge of clock comes and the imaginary part can be calculated when negative edge of clock comes. Hence using a mux, only two multipliers do the job.

4.6 Wallace Tree Multiplier for RISC Processor

A multiplier block is an important component of microprocessors as well as an essential part of DSP processors. Wallace Multiplier is one of the more popular multipliers designs out there. The Wallace tree multiplier architecture consists of an array of AND. This is used to compute partial products; an adder (typically a carry save adder) is used for adding the partial products which were obtained previously. And a carry propagate adder is used in the final stage of addition. It is possible to improve the design and the same is proposed. The conventional Wallace Tree multiplier consists of adders arranged in different stages. The first two adders don’t depend on carry but subsequent stages do. This causes propagation delay and total latency for an 8-bit multiplier comes out to be 27 with the net total number of stages in the multiplier in the critical path being 13. The proposed novel architecture [15] targets to decrease this latency. It makes the use of uses compressor logics instead of using full adders. Two full adders can be replaced by a single 4:2 compressor. And a 5:2 compressor takes place of three full adders. In the final stage, tree adder is used to reduce power consumption. Sklansky type of tree adder is chosen for this task it also reduces latency. The final latency counts this improved design gives is 15 which is a significant improvement. Furthermore, Transmission Gate Based Mux is used in the compressor therefore reducing the number of transistors. Finally, after the design is implemented, it is shown that for supply voltage of 3.3V and at a frequency of 50MHz, the power consumption of the proposed multiplier circuit is 4.57% less than the conventional multiplier circuit. The corresponding power reduction observed at different frequencies is shown below [16] Figure 8.

![Figure 8: Power Consumption Comparison between conventional and proposed method with respect to the variation of frequency and voltage respectively](image)

4.7 PROGRAMABLE ARITHMETIC LOGIC UNIT

Hardwired General Purpose Processors are widely used in Embedded Systems. However, these are not always suitable for certain applications due to their non-flexibility. Some alternatives are FPGAs and ASIPs but they are limited in one way or the other. The design proposed [17] aims to solve the problem by combining reusability of programmable logic units with computational performance of hardwired design. It consists of a modified a DLX processor which is a 32-bit RISC processor. The DLX architecture consists of a register bank with 32 registers of 32-bit size. The performance is increased by using a 5-stage pipeline which contains the following stages: - fetch the instruction from memory ->
decoding of fetched commands -> generate execution signals like enable the arithmetic unit and pass operand to it -> perform operation -> store the result in memory -> fetch next instruction. Due to pipeline stages, hazards can also arise. These can be when different instructions need to access same resource at the same time, called Structural Hazard. When a branch instruction can affect an instruction present in the instruction in pipeline, called Control Hazard. They can also occur when an instruction needs a data not yet present in the memory, called as Data Hazard. All of these can be avoided in the proposed design. 

The main improvement to DLX is done by remodelling the ALU into a programmable ALU. This is done by using an LUT. For ensuring the perfect execution of existing DLX instruction set, $2^5$ LUTs of size 32 x $2^64$ bits are needed. Since this has a very large memory requirement, further optimizations are done to reduce LUT size and number of LUTs. Final result is obtained that contains 16-page LUTs of size 4 x $2^8$. Out of these 15 pages are reserved for usage for indigenous instructions which DLX has and the remaining one can be configured according to user needs. For configuring the pages there is provision of soft reset (to go back to initial state) and hard reset (for clearing LUTs) [18].

**4.8 POWER OPTIMIZED DUAL CORE**

The idea proposed seeks to design a 32-bit DLX processor having two cores instead of one. This would allow for parallelism which would lead to low power consumption. The first step is to optimize the single core design by removing the non-utilized registers. In the dual core implementation instead of using two data memories, a single one is used so that un-used registers are minimized as much as possible. DLX architecture comprises of a five-stage pipeline and has broadly three types of instructions. jump type, immediate type, Register type. Next, we move on to the dual core aspect of the design. The single core processor executes the commands one after the other but using dual core and separate ROM blocks, they can run parallel. Although it does have a disadvantage that it occupies large area, this can be compensated by sharing same RAM block. This brings about a new problem of dispute between cores over the same memory index, but this can be taken care of manually by specifying the commands or by implementing a Hash Table for collision handling. Finally, the optimized single core and multicore designs are simulated and their comparison is done. Dual Core Shared Memory design occupies about 4 times the area of Single Core design which is to be expected but just consumes 3mW of power more than single core.

**4.9 POWER OPTIMIZATION BASED ON HDL MODIFICATIONS**

Several methods for optimizing power consumption in RISC processor exist in the literature. Many of these are optimizations in the architectural level like clock gating, using Wallace multiplier, advance branching techniques etc. Contrary to these, the technique proposed by Soumya Murthy and Usha Verma their work is focused on HDL modification to save power dissipation. This is modification at the algorithmic level unlike at the architectural level Figure 9. It is difficult to control static or quiescent power. More so, using HDL techniques modification, it is possible to reduce power usage. The another most power usage unit was in FPGA is IOs. This is also dynamic power. This should also be reduced to get maximizing the use of power in good manner. Power management in the FPGA consists of several factors like battery life (for portable devices), environmental factors, digital noise immunity and cooling & packaging costs. Also as discussed above mainly two type of power dissipation sources are there: - 1) Leakage power consumption: The dissipation of power when the system is in idle/standby mode. 2) Dynamic power consumption: Occurs when there is a logic transition of logic gates or flip-flops. The various VERILOG coding methodologies used to implement low power RISC are discussed below:

- Not using a reset in the design as FPGA globally gets a reset automatically.
- Using a low power architcting for multiple Block RAM arrays (using CORE generator).
- Enabling the Block RAM only during active read/write operations.
- Using Synchronous reset in the design.
- Minimizing internal resets.
12

• Building LUTs (<4k bits) based small memory blocks.
• Controlling the Clock Enables in the design.

![Figure 9: Bar graph shows the comparative power for different units with and without HDL modifications](image)

Finally, the simulated and synthesized design it shows that the power saved by using the above-mentioned techniques is ~22mW. Therefore, we can say that on comparison, an overall reduction of 13.33% is obtained.

5. APPLICATIONS AND CURRENT TRENDS

The paper so far focussed on understanding the concept of RISC design and methods for the purpose of capabilities enhancement (with regard to both functionality and performance) of a processor. Along this, we also get an idea about the extends of RISC based applications which encompasses convolution application, signal processing, commercial data processing, broader base for tablets and smart phones real-time embedded systems. Now, it’s time to dive deep in the application part of RISC design so that a designer gets more clarity whether to choose RISC as a base for their own project as per the current trends. The objective of this section is to provide a review on the recent published applications-based research papers so that the reader should know the impact of RISC design in different fields. Now days, RISC V Instruction Set Architecture is gaining popularity because of its free and open-source ISA feature. By the word open source means that a designer can manipulate the instructions and add extensions according to the application. Other features of RISC are modularity of ISA, multiple sources for intellectual property. RISC-V also gets the support from high performance CPUs like the supercomputing field and data centre, embedded CPUs like IoT and mobile industry. The RISC V has been adopted by SoC industry and researchers because reduction in manufacturing cost is the main of SoC while working on black box hardware is a major huddle in the research field. The developers also get help by RISC V as they provide proper ecosystem where they can run their whole hardware/software program on their FPGA. Safety and authenticity are the critical issues of robotics field and as RISC V provides open-source operating system makes robotics, a good application of RISC. The foundation of RISC V is functioning on providing cloud-based FPGA which makes difficult for designers to run RISC V on local FPGA. mainly focussed on designing a local FPGA which show an interface between peripherals and ROC code. The proposed structure objective is to take the inputs from outer world and send to the ROS operating system for processing and send back the output via same FPGA board to the outer world. It is consisting of two parts: Software and hardware. Software part includes Robot Operating System (ROS) which in turn consist of RMW (ROS Middle Ware) and RCL (ROS Client Library). The RCL work is to establish a connection between ROS and user application and the programming related to it is done in various language like C, C++, python. On the other hand, RMW role is to manage system operations. It schedules the tasks and manage the communication between external and internal node depending upon the priority set by user. The hardware part includes the implementation of RISC V BOOM (Berkeley Out-of-Order Machine) which is connected internally with other peripherals using UART, I2C, SPI protocols. A Linux distribution project Milli Isletim Sistemi in
Turkish (MILIS) is implemented using Linux from Scratch (LFS) techniques. This project is very much different from GNU/Linux projects because of package manager and its independent base. It has package production and its own self sufficient root file system. Free and easily accessible to use operating system is the goal of this project. MILIS is a voluntarily initiated project because of this its ultimate goal is to be a open source software. To make a complete open-source system, RISC V is the most promising choice and according to the knowledge of M. Numan Ince and his colleagues, no operating system is working locally to support such open-source ISA. So, Ince et al. On RISC-V Build an Open-Source Linux Computing System, proposed an operating system to support RISC V architecture. The proposed operating system is a fork of MILIS project. In the Linux community, their work in known by the name ‘Port’. Another major field which is collaborating with RISC for the implementation of its applications is Software-defined Radio (SDR). Accetti R. et al. present a processor design which is basically a RISC V ISA with a extension of instruction set, idealized for SDR applications. The architecture includes a core processor which consists of the standard multiplication extension, a RISC-V processor of 32-bit with integer base instructions and the Oolong baseband processing accelerator unit. The 3-stage in-order pipeline Harvard architecture is the main processor core used, which uses the integer division and multiplication standard extension (RV32M) and the RISC-V integer base instructions (RV321). In this paper, in order to compare the performance of RISC-V. The 3-stage pipeline ARM cores published results and the Dhrystone 2.1 benchmark results have been compared. A hardware accelerator Oolong was created for the implementation of packed 4x8 bit fixed-point instructions and baseband processing. There are various instructions that can be executed using standard RISC-V instructions. The performance of the core in baseband processing applications would be affected by the number of cycles require to implement those operations along with standard ISA. To accelerate the performance of the Advanced Encryption Standard cryptography algorithm, some instructions were add up to the Oolong extension. The complex-number operations are performed by the custom instructions, designed for baseband processing and complex modulation. The proposed system architecture consists the IO, processor core, a WISHBONE bus interconnection and peripherals, and was pointed to an Alter Cyclone III FPGA that achieved 0.9 DMIPS/MHz except using any Optimizations. With the rapid increase of security threats, many researchers proposed solutions against security attacks but most of them are implemented in software which in turn provides a performance overhead. To overcome this, hardware solutions are also published but the permanent modification is needed to the host processor. So, an interface is needed which can enable the security module with causing a permanent change to the main processor. This requirement is observed by Hyunyoung Oh and he proposed an interface in which he develops a security interface which is capable of extracting the internal information possessed by the host processor, which can be used to recognize the data access patterns and the control flow. It is assumed that the target system is coming in sort of a SoC also it is possible to attach any external hardware during implementation. The security interface extracts the internal information from RISC-V processor that are data address and data access pattern. Each delayed address is multiplexed carefully with the help of control signal, which is used as a selector. Memory region protector (MRP) is a module that establishes the isolation environment by keeping a check on the occurrence of illegal memory access. MRP utilizes access permission matrix to detect an illegal access. There are two types of relationships maintained by the access permission matrix i.e., code versus data region and code versus code region. There are two registers that are provided to set the low and high addresses of the region, namely, data region selector and code region selector. The results indicate that the security module has enabled by the security interface that is operated separately of the processor which has low area overhead and no performance.

REFERENCES

[1] S. Furber, Arm System-on-chip Architecture, 2th edition, UK: Addison-Wesley, 2000.
[2] Sivarama P. Dandamudi, Guide to RISC Processors, NY: Springer-Verlag New York, 2005.
[3] M. Zaid, P. Mustajab, DESIGN AND APPLICATION OF RISC PROCESSOR 2017 International Conference on Multimedia, Signal Processing and Communication Technologies (IMPACT), pp. 242-246, November 2017.

[4] A. Pandey, Study of Data Hazard and Control Hazard Resolution Techniques in a Simulated Five Stage Pipelined RISC Processor, 2016 International Conference on Inventive Computation Technologies (ICICT), pp. 1-4, Aug 2016.

[5] D. Devikanniga, A. Ramu, and A. Haldorai, Efficient Diagnosis of Liver Disease using Support Vector Machine Optimized with Crows Search Algorithm, EAI Endorsed Transactions on Energy Web, p. 164177, Jul. 2018. doi:10.4108/eai.13-7-2018.164177

[6] H. Anandakumar and K. Umamaheswari, Supervised machine learning techniques in cognitive radio networks during cooperative spectrum handovers, Cluster Computing, vol. 20, no. 2, pp. 1505–1515, Mar. 2017.

[7] Gokulan T, A. Muraleedharan, K. Varghese, Design of a 32-bit, dual pipeline superscalar RISC-V processor on FPGA 2020 23rd Euromicro Conference on Digital System Design (DSD), pp. 340-343, Aug 2020.

[8] S. Qureshi, K. R. Sanjeev, Power and Performance Optimization using MultiVoltage, Multi-Threshold and Clock Gating for LowEnd Microprocessors TENCON 2009 - 2009 IEEE Region 10 Conference, pp. 1-6, Jan 2009.

[9] P. k. Yadav, P. K. Misra, Power Aware Study of 32-bit 5 stage Pipeline RISC CPU using 180nm CMOS Technology 2017 14th IEEE India Council International Conference (INDICON), pp. 1-4, Dec 2017.

[10] S. Palekar, N. Narkhede, 32-bit RISC Processor with Floating Point Unit for DSP Applications IEEE International Conference On Recent Trends In Electronics Information Communication Technology, pp. 2062-2066, May 2016.

[11] D. Babayan, E. Babayan, S. Antonyan, E. Kagramanyan, A. Avetisyan, A new Approach of Multi Voltage and Adaptive Voltage Scaling Techniques for 16 nm FinFET RISC Processor 2017 IEEE 37th International Conference on Electronics and Nanotechnology (ELNANO), pp. 128-131, April 2017.

[12] C. Vinoth, V. S. K. Bhaaskaran2, B. Brindha, S. Sakthikumaran, V. Kavinilavu, B. Bhaskar, M. Kanagasabapathy, A. Sharath, A Novel Low Power and High Speed Wallace Tree Multiplier for RISC Processor 2011 3rd International Conference on Electronics Computer Technology, pp. 330-334, April 2011.

[13] H. Najjar, R. Bourguiba, J. Mouline, A reusuable hybrid RISC processor with programmable instruction set 2018 15th International Multi-Conference on Systems, Signals & Devices (SSD), pp. 1028-1031, March 2018.

[14] S. Pandit, P. Sifka, Design and implementation of power optimized dual core and single core DLX processor on FPGA 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), July 2018.

[15] S. M. Bhatat, S. U. Bhandari, Design and Analysis of 16-bit RISC Processor 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUENA), pp. 1-4, August 2018.

[16] R. Samanth, A. Amin, S. G. Nayak, Design and Implementation of 32-bit Functional Unit for RISC architecture applications 2020 5th International Conference on Devices, Circuits and Systems (ICDCS), pp. 46-48, March 2020.

[17] C. Venkatesan, T. Sulthana M, Sumithra M.G, Design of a 16-Bit Harvard Structure RISC Processor in Cadence 45nm Technology, 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), pp. 173-178, March 2019.

[18] J. Jeemon, Low Power Pipelined 8-bit RISC Processor Design and Implementation on FPGA 2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), pp. 476-481, December 2015.