International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 07 | July - 2021 ISSN: 2582-3930

Processor using RISC-V ISA

Prof. A.S. Nigade[1], Prof. S.K. Pawar[2], Abhijeet Banerjee[3], Nihar Das[4], Sanket Ghosh[5]

[1]Mrs. A.S. Nigade (Asst. Professor, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, India)
[2]Mrs. S.K. Pawar (Asst. Professor, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, India)
[3]Abhijeet Banerjee (Student, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, India)
[4]Nihar Das (Student, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, India)
[5]Sanket Ghosh (Student, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, India)

Abstract - Majority of all the processors and micro-controllers utilized all over world now are either based on x86 Instruction Set Architecture (ISA) or are ARM based. Both of these ISAs are proprietary and cannot be easily availed by people wishing to design their own CPU or extend the ISA for their specific use case. In light of this problem the RISC-V ISA was designed which is an open specification ISA. In this project we are designing our own CPU utilizing the RISC-V ISA. To realise this project we are not going the usual of route using a generic Hardware Descriptive language(HDL) like VHSIC Hardware Description Language (VHDL) or Verilog. But we are using a scala based library called SpinalHDL to design our CPU. There are various benifits of using SpinalHDL to define a CPU like there is no need for re-stating same objects within the HDL code and having higher level of abstraction for defining the logic.

Key Words: RISC–V ISA, Central Processessing Unit (CPU), HDL, SpinalHDL

1.INTRODUCTION

When a processor is designed first things to be considered is the application environment for the processor. Such as whether the application needs a single core (processor) or multi core design. The designing of the data path takes place next which works as the logic control flow network. The datapath is laid out to handle the necessary capabilities.It is the hardware that performs all the required operations, for example, ALU, registers, and internal buses. Then the instruction set is then evaluated to check if it needs to be extended for a particular task. The designing of the logic control unit is done next. The design of the logic control unit determines how the datapath operates. In this way, the control unit can specify how the data flows through the datapath.The implementaion can be implemented as an instruction register or a instruction decode unit. This is all done while taking into account the address path required for the processor function The cpu designed here is a low performance / low power design and will be usefull for applications needs such as in embedded systems, industrial iot, remote monitering systems etc.

RISC – V is an ISA that is easy to implement, simple and inclinates towards low power designs. Its is a free and open instruction-set architecture (ISA) that is licensed under BSD license.

It is designed to be modular with a 32, 64, 128-bit integer base and provide various optional extensions like floating point. It has been designed while taking into account for all market use cases such as in microcontrollers, image, graphics, and PC/server processors. It is also suitable for implementation ranging from FPGAs to fully custom layouts. With an increasing interest in design and application of accelerators , extensibility has been made an essential part of universality. For an ISA to be easily extensible all the while having to maintain a large software support is tricky as software developers need a consistent target. In order to resolve for this issue the base integer instruction set and a few optional and predefined extensions have been guaranteed to be stay the same and a mechanism for creating various extensions has been provided The base ISA is architected to simplify implementation. To enable extensions, some portions of the instruction encoding space has been set aside for future use cases.

Aside for monetary value a compelling reason behind choosing RISC V is that existing popular instruction set architecture have several drawbacks associated with it.

Drawbacks with x86-64

- Power inefficient : x86 architecture tends to be power hungry when compared to alternatives and therefore naturally, produce more heat.
- Huge and Complex Instruction Set : x86 is decades old architecture and developments took place over decades. 64 bit instruction set AMD64 is just extension of 32 bit x86 architecture. Thus, with changes and revisions instruction set has become unnecessarily complex.
- Scalability : Scaling the x86 architecture is very complex and time consuming to achieve.
- Security : With revisions in instruction set, a lot of bugs are introduced which can be a potential security flaw. These bugs cannot be corrected and present permanently.
- Proprietary IP : x86 is an intellectual property and monopoly of Advanced Micro Devices and Intel. Thus dealing IP is time consuming and complex.

Drawbacks with ARM (A32, A64)

- Proprietary IP : Working with intellectual property is costly and time consuming and often comes with many hindrances.
- Complex decoding : ARM ISA instructions requires a complex decoding process which increases overhead for program execution.
- Complex addressing modes: Various features of the CISC architecture have been incorporated into the ARM ISA such as their multiple load/store

instruction. This has lead to increased complexity of the processor and also increases the processor die size.

2. SYSTEM OVERVIEW

In our project, we are designing a low power processor for utilization in embedded system environment using RV32IM instruction set. RV32I is a base 32-bit integer ISA. It comprises of 47 instructions. There are eight instructions that are system instructions (system calls and performance counters) which if implemented as a single trapping instruction, reduces the number of user-level hardware instructions to 40. The instructions are 32 bits long. They are stored naturally aligned in memory in little - endian byte order.

The pipeline consists of 5 stages (Fetch, Decode, Execute, Memory, WriteBack).

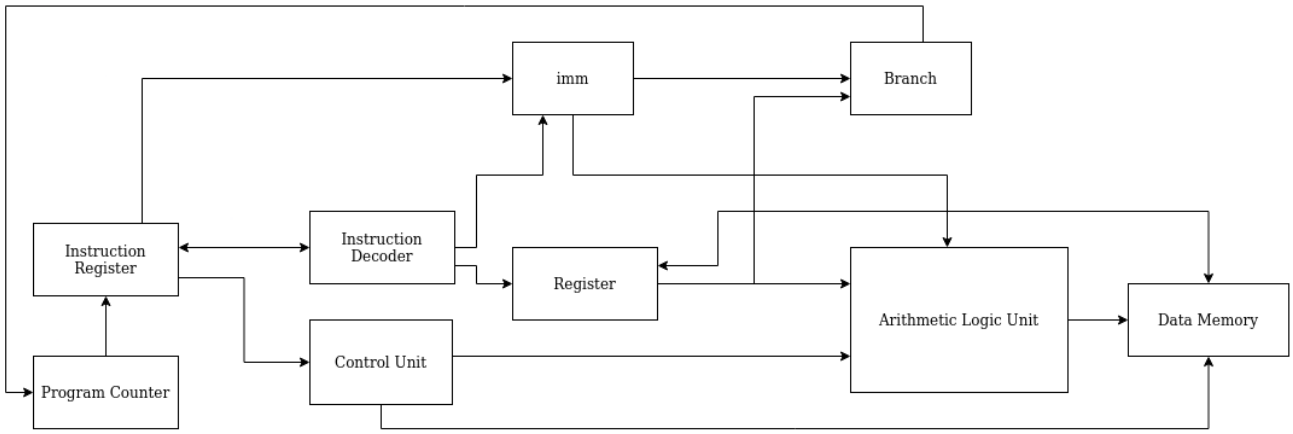

Figure 1: Block diagram

The designing of the processor has been done with SpinalHDL. The scala library SpinalHDL has been used to ease the designing complexity of the processor design stage. The use of modern language paradigms such as those implemented in scala means that with a library for hardware descriptive keywords; it can function as a much better HDL than VHDL or Verilog. It is able to provide a higher abstraction level meaning less lines of code for defining the logic and lesser hindrance with defining all states in every entity.

3. WORKING

Stage 1 : Instruction Fetch
Instruction is fetched from memory.
Instructions from the address given from the Program Counter are sent to next stage. Then the Program Counter is incremented.
Instruction register stores the pre-fetched instructions of the program being executed.

Stage 2 : Instruction Decode
Instruction is decoded in the Instruction Decoder. Then the operator and operands are sent to register.
imm (Immediate) is value added in the instruction itself, instead of register or memory location. This is used in instructions that perform arithmetic or logical operations on constant.
Control Unit gets instructions from instruction decoder and gives control signals and timing signals to the execution units of the CPU such as ALU, registers.

Stage 3 : Execute
ALU (Arithmetic Logic Unit) is digital circuit that performs arithmetic operations such as the add, subtract, increment, decrement; bitwise logical operations such as AND, OR, XOR and ones complement and bitshift operation on the numbers. It is the fundamental block of the CPU where all the principal operations are performed.

Stage 4 : Memory Access
Load and Store instructions are performed in this stage. Data Memory stores the instructions.

Stage 5 : Write back
The Program Counter is updated. Branch makes decision regarding the flow of instruction. In order to change the flow, jumps are performed depending on the conditions. When a jump is done, the Program Counter is updated and instruction that will be executed next are updated.

4. CONCLUSIONS

By utilizing the newer available digital hardware design methods and an open specification ISA the designing and development of complex digital hardware can be made much simpler and easier to implement. With the end result of having the option of improving and securing the previous iteration much quicker than what is seen in the industry today.

5. FUTURE SCOPE

Our project is based on an open ISA, so any improvements done on the instruction architecture can be easily expanded and changes can be easily incorporated in future iterations. Which is something not seen in todays consumer processors as the design and implementaion methodology is highly complex leading to multiple re-implementations of the same logic designs. This is also exacerbated due to the constraints of the currently used HDLs due to its verbosity.

REFERENCES

[1] The RISC-V Instruction Set Manual
https://github.com/riscv/riscv-isa-manual/releases/download/Ratified-IMAFDQC/riscv-spec-20191213.pdf

[2] A RISC-V instruction set processor-micro-architecture design and analysis
DOI: 10.1109/VLSISATA.2016.7593047

[3] Single Cycle RISC-V Micro Architecture Processor and its FPGA Prototype
DOI:10.1109/ISED.2017.8303926

[4] Design of the RISC-V Instruction Set Architecture Technical Report No. UCB/EECS-2016-1
http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-1.htm

[5] Instruction Sets Should Be Free: The Case For RISC-V Technical Report No. UCB/EECS-2014-146
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-146.html

[6] Software tools, risc-v foundation
https://riscv.org/software-tools/

[7] S.P. Dandamudi, Guide to RISC Processors: For Programmers and Engineers. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005

[8]J. Tandon, “The openrisc processor: Open hardware and linux,” Linux J., vol. 2011, no. 212, Dec. 2011.
http://dl.acm.org/citation.cfm?id=2123870.2123876