

DESIGN AND FPGA IMPLEMENTATION OF A HIGH-PERFORMANCE RISC-V EMBEDDED PROCESSOR USING KOGGE-STONE ADDER FOR LOW-LATENCY ARITHMETIC OPERATIONS

¹KOPPISETTI SRI SAI KRISHNA KARTHIKEYA, ²Dr.E.DEEPTHI

¹Student, Department of Electronics and Communication Engineering, Malla Reddy Engineering College.

karthikkoppisettkarthik@gmail.com

²Assistant Professor, Department of Electronics and Communication Engineering, Malla Reddy Engineering College.

ABSTRACT:

Ripple carry and carry look-ahead adders are commonly used in the Arithmetic Logic Unit (ALU) of traditional RISC-V processors, significantly impacting processor speed and resulting in a substantial carry propagation delay during arithmetic calculations. In this work, we address these constraints with the design and FPGA implementation of a multi-cycle RISC-V Embedded Processor with high performance using the Kogge-Stone Adder for low latency arithmetic operations. The suggested processor is designed using the Verilog Hardware Description Language (HDL) and implemented with the Xilinx Vivado Design Suite. For reducing carry propagation delay and accelerating arithmetic calculation, a 32-bit Kogge-Stone Adder (KSA) is used in the ALU and Program Counter (PC). A multi-cycle control architecture coordinates the various parts of the processor, such as the Control Unit, Datapath Unit, Register File, ALU, Instruction Memory, Data Memory, Program Counter and Kogge-Stone Adder, to make sure they perform the instructions correctly. Behavioral simulation, RTL synthesis, timing analysis and implementation of the suggested hardware design are used to validate the accuracy and stability of the design for different instruction contexts. Experimental results indicate faster arithmetic processing, reduced processing delay, efficient usage of FPGA resources and lower on-chip power consumption when compared to the traditional RISC-V processor architectures. The design of the system can be scaled up and down for future improvements like pipelined execution, cache memory integration, floating-point arithmetic, AI accelerators, cryptographic extensions, and Internet of Things (IoT) embedded applications. Overall, the proposed FPGA-based RISC-V processor offers a high-speed, scalable, and power-efficient computing solution for modern embedded systems, real-time digital signal processing, edge computing, industrial automation, robotics, and beyond.

Keywords: : Xilinx Vivado, RISC-V Processor, FPGA, Verilog HDL, Kogge-Stone Adder (KSA), Arithmetic Logic Unit (ALU), Multi-Cycle Processor, Low-Latency Computing, Parallel Prefix Adder, Hardware Acceleration, Digital System Design.

Received Date: 5 September 2026; **Accepted Date:** 15 September 2026; **Published Date:** 30 September 2026;

This is an open-access article distributed under the terms of the Creative Commons Attribution 4.0 International License (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author(s) and the source are properly cited.

1.INTRODUCTION

The need for high-performance processor architectures that can carry out intricate computational tasks with low latency and high energy efficiency has greatly increased due to the quick development of embedded computing, the Internet of Things (IoT),

artificial intelligence, industrial automation, and real-time digital systems. Its modular design, adaptability and royalty free nature makes it one of the most flexible and scalable instruction set architectures available today, the Reduced Instruction Set Computer Version

Five (RISC-V). While proprietary processor architecture can be optimized for specific applications, RISC-V is suitable for different application-specific customization of various processor components, which allows using it in embedded systems, edge computing, robotics, automotive electronics, devices for healthcare, aerospace systems, and industrial control. Although RISC-V processors have many advantages, conventional multi-cycle RISC-V processors still experience performance bottlenecks in terms of ALU delay associated with carry propagation delay when employing conventional ALU architectures like Ripple Carry Adders (RCA) and Carry Look-Ahead Adders (CLA). The execution times, throughput and overall efficiency of the computation are among the major bottlenecks at high operating frequencies of processors that affect the instruction execution time, with arithmetic delay being one of them. This latency is getting reduced in the modern digital system design with the increasing reliability of the digital hardware, a new research area in digital system design. Field Programmable Gate Arrays (FPGAs) provide an ideal hardware environment for the realization of optimal processor designs thanks to their capability of parallel execution, reconfiguration of the hardware, short execution latency and efficient utilization of programmable logic resources. The FPGA-based processor can be enhanced with high-speed arithmetic circuits to achieve faster computations without adding too many complexities. Therefore, designing a parallel-prefix adder based more efficient RISC-V processor would be a possible solution to improve system performance, reduce arithmetic latency, and accelerate the processor, as shown in [1], [2] and [3].

Numerous researchers have suggested different ways to augment the ability of RISC-V processors to execute instructions faster by optimizing the arithmetic units, control logic, memory organization and the way instructions are executed. The Ripple Carry Adders are usually employed in the conventional arithmetic logic units due to its simple hardware implementation. However, the carry signals have to be propagated sequentially across each bit-position for larger bit-width operations, which introduces significant propagation delay.

Keep an Eye on the Future: Adder's hardware complexity grows rapidly as the size of the operands increases, making them less suitable for big arithmetic units, but they can enhance the speed of the arithmetic unit by generating carry signals in parallel. A carry tree can be used to perform carry computation in parallel, and parallel prefix adders, particularly the Kogge-Stone Adder (KSA), can circumvent these problems, with latency of only $\log(N)$, where N is the number of addends, and greatly enhance arithmetic performance. The Kogge-Stone Adder has become one of the most successful adder designs for high-performance CPUs due to its regularity of connectors, its balanced fan-out and high speed operation. The researchers are able to introduce the KSA architectures with a success rate into the digital signal processor, cryptographic acceleration units, and arithmetic units, in order to shorten the computation time and boost the computing power. A significant number of existing RISC-V implementations on FPGAs, however, still rely on the traditional adder design, thereby impacting the performance of the processor in complex arithmetic operations, and also introducing an arithmetic latency. Also, most of the current solutions focus on the correctness of function, rather than on optimizing temporal performance, power consumption and FPGA resource usage. A multi-cycle optimised RISC-V processor with a high-speed arithmetic unit, that will still be modular, scalable and hardware efficient is therefore sorely needed. This type of architecture may enable future changes that are needed for current embedded computing applications and enhance instruction execution speed without increasing the complexity of the architecture [4, 5, 6].

To overcome these problems, this work is forward-looking to design and implement a high-performance multi-cycle RISC-V embedded processor on FPGA for low latency arithmetic operation using Kogge-Stone adder. Once implemented using the Xilinx Vivado Design Suite and built in the Verilog Hardware Description Language (HDL), the suggested

processor is then implemented on an FPGA. The architecture features a 32-bit Kogge-Stone Adder (KSA) both in the Arithmetic Logic Unit (ALU) as well as the Program Counter (PC) to speed up bits arithmetic calculations and also to lessen carry propagation delay during the execution of instructions. The Control Unit, Datapath Unit, Register File, Instruction Memory, Data Memory, Program Counter, ALU, Immediate Generator, Multiplexers, and Kogge-Stone Adder are some of the functional modules that make up the CPU. All the modules are controlled with a multi-cycle control mechanism. During the execution of each instruction, there are a set of sequential steps: Instruction Fetch, Instruction Decode, Operand Execution, Memory Access, and Write-back. With the improved KSA, it performs the arithmetic computations much quicker. The accuracy and stability of the proposed hardware architecture have been tested with the help of behavioural simulation, RTL synthesis, implementation, timing and power analysis with the Xilinx Vivado Design Suite. Experimental results are compared with the conventional implementation of RISC-V processors to prove the arithmetic execution speed improvement, the critical route latency reduction, efficient utilization of FPGA resources and power saving. The modular FPGA design also has the potential to be extended in the future with pipelined execution, floating-point arithmetic units, vector processing extensions, cache memory, Internet of Things (IoT) enabled embedded systems, AI and machine learning accelerators, and cryptographic processors. The recommended processor is therefore a scalable computing platform and has high performance, energy efficiency and is suitable for next generation embedded applications, real-time digital signal processing, industrial automation, autonomous systems, robotics and smart electronic devices [7, 8 and 9].

II SURVEY OF RESEARCH

Typical RISC-V processors use an arithmetic logic unit (ALU) that includes ripple carry or carry look-ahead adders that slow the processor speed and can result in a large carry propagation delay during arithmetic operations. Here, a way to address these limitations is proposed to design and implement a high-performance multi-cycle RISC-V embedded processor using a Kogge-Stone adder for low latency arithmetic operations. The suggested processor is implemented in the Verilog Hardware Description Language (HDL) by Xilinx Vivado Design Suite. ALU and Program Counter (PC) use a 32 bit Kogge-Stone Adder (KSA) to remove propagation delay from carries and to increase the speed of the arithmetic operations. A multi-cycle control architecture that coordinates the different parts of the processor: the Control Unit, the Datapath Unit, the Register File, the ALU, the Instruction Memory, the Data Memory, the Program Counter and the Kogge-Stone Adder to execute instructions effectively. For different instructions, correctness and stability of the proposed hardware design are checked by performing behavioral simulation, RTL synthesis, timing analysis and implementation. From the experimental results, it can be seen that the proposed architecture of a processor is found to be better than the conventional RISC-V processor architectures in terms of arithmetic execution time, processing delay, FPGA resources utilization, on chip power consumption. The modular design of the chip makes it easy to add future extensions, including pipelining, cache memory, floating-point, AI accelerators, support for cryptographic extensions, integration into applications with an Internet of Things (IoT) connection and more. The proposed FPGA based RISC-V processor therefore is a very good prospect for next generation of smart hardware applications such as real time digital signal processing, Edge computing, industrial automation, robotics etc.

The topics covered are: Xilinx Vivado, RISC-V Processor, FPGA, Verilog HDL, Kogge-Stone Adder (KSA), Arithmetic Logic Unit (ALU), Multi-Cycle Processor, Low-Latency Computing, Parallel Prefix Adder, Hardware Acceleration, Digital System Design.

Given its open source nature, moduled design, and customizability to meet the requirements of embedded system design, the RISC-V Instruction Set Architecture (ISA) has been a popular choice in the world of computing. One important difference from proprietary processor architectures is that RISC-V processor architectures allow the freedom to design processors to match the application needs without the burden of licensing. RISC-V processors have demonstrated their exceptional scalability and versatility, and they are suitable for use in edge computing, robotics, industrial automation, artificial intelligence and the Internet of Things (IoT). There are a number of implementations of RISC-V processors that are able to verify them using Verilog HDL and VHDL that are successful. The designs can provide reliable instructions execution and modular hardware implementation, but still many processors implement traditional arithmetic units which will create more processing delays. Hence, the researchers have concentrated on optimizing the performance of the processors involving the use of optimized arithmetic circuits and hardware acceleration techniques without modifying the instruction set architecture (ISA) of the processors with the standard RISC-V architecture (RV64IMcF) [2], [1].

FPGAs are one of the most popular embedded computer hardware platforms because of its programmability, parallel processing and flexibility. In order to test the performance of the fabricated processor before the actual fabrication of ASICs, different embedded processors have been designed on FPGA platforms by using Verilog HDL. In addition to custom integrated circuits, other advantages of FPGA implementation include fast prototyping,

low development cost and hardware debugging. FPGA devices provide a number of built-in capabilities that can dramatically boost the performance of a processor such as configurable logic blocks, embedded memories, DSP slices, and high-speed interconnections. The study has also demonstrated the ability of an FPGA-based RISC-V processor to emulate software and enhance the timing performance and efficiency of the hardware. But most of the currently available solutions are based on functional verification and not aimed at optimizing the arithmetic speed and minimizing the execution time. Therefore, to increase the arithmetic efficiency of the processors, FPGA based processors with high speed arithmetic designs were found to be a large research area [3],[4].

The Kogge-Stone Adder (KSA) is one of the fastest parallel prefix adders is used in modern digital circuits, because its carry propagation has a logarithmic delay. The Kogge-Stone Adder calculates carry signals in parallel, in a tree structure, as opposed to the Ripple Carry Adders where carry signals are calculated one at a time. The overall speed is increased and the relative decrease of the arithmetic delay is because the carry is computed in parallel. The architectures of KSA have proven to be suitable for researchers working with high performance embedded processors, digital signal processors, cryptographic hardware and image processing systems. Based on operating frequency and latency, it can be seen that FPGA implementations of KSA are better than those of Ripple Carry Adders (RCA) and Carry Look-Ahead Adders (CLA). Despite the advantages of the RISC-V design, there are many processor variants that have used the slower arithmetic designs that are more popular. Thus, the Kogge-Stone Adder can be incorporated into the ALU and Program Counter without compromising the hardware efficiency of the processor, and make it faster, with no extra hardware components added to the system [5], [6].

The most important part of any processor is the Arithmetic Logic Unit (ALU) which carries out logical and arithmetic operations needed to execute instructions. Different studies have been conducted to propose the best possible ALU architectures, so as to improve the processor throughput and reduce the execution time. The various approaches to improving the arithmetic performance are carry select adders, carry skip adders, carry look-ahead adders and parallel-prefix adders. But in high speed applications, parallel-prefix adders are better than other adders in that they can minimize the propagation of the carry without compromising accuracy. Processors' frequency and efficiency have also been improved by tuning their ALUs using FPGA. However, there still exist a few embedded CPU implementations, which use the classic ALU architecture, which is less efficient for computation. Some of the advanced arithmetic units in ALU such as Kogge-Stone Adder are able to perform an arithmetic operation within a shorter time, and can also reduce the processing time of the embedded system and improve the overall performance of the embedded system [7], [8].

Multi-cycle processor architectures execute instructions over a number of processor cycles so that common hardware components (such as ALU, memory units) can be shared in each cycle. The results from the studies have shown that the multi-cycle processors get a better use of the hardware and also have lower implementations cost than single cycle processors, as the hardware is reused in multiple cycles of an instruction. These are typical execution steps, such as instruction fetch, instruction decode, execution, memory access and write-back. These phases are synchronized and sequenced correctly with the help of the controllers that are based on Finite State Machines (FSM). The multi-cycle processor solution with FPGA is much more scalable, adaptable and verifiable, as a hardware design. Even with traditional architectures of adders, however, many multi-cycle CPUs are still plagued with sub-optimal

arithmetic performance. Therefore use of parallel-prefix adders instead of the usual arithmetic units used in multi-cycle processors can greatly enhance the performance of this processor organization.

While much research has been done on parallel prefix architectures, multi cycle processor design, ALU optimization and FPGA based RISC-V processors, there are still several problems with existing parallel-prefix implementations. Most of the CPUs are still utilizing Carry Look-Ahead Adders or Ripple Carry Adders, thus decreasing the computing performance and the arithmetic delay. Furthermore, most of the researches consider only the functional correctness without taking into account at the same time the optimization of time, power utilization and FPGA resources utilization. To overcome these, the proposed work includes an embedded processor based on a High-Performance Multi-Cycle RISC-V (MPRV) processor with a modification of the Program Counter (PC) and Arithmetic Logic Unit (ALU) by embedding a 32-bit Kogge-Stone Adder. The architecture is designed in Verilog HDL and validated by performing behavioral simulation, RTL synthesis, implementation and timing and power analysis in Xilinx Vivado. As it achieves reduced arithmetic latency, high run-time performance, low power consumption and efficient FPGA resource utilization, the suggested processor is appropriate for various applications such as embedded computing, real-time processing, digital signal processing, Internet of Things devices, robotics, and artificial intelligence and next generation high performance computing application [11, 12].

III. PROPOSED SYSTEM

In conventional RISC-V processors, the Arithmetic Logic Unit (ALU) is often configured with ripple carry or carry look-ahead adders that have slow processing speeds and large carry propagation delay in arithmetic

operations. To overcome the aforementioned limit, we introduce the Design and FPGA Implementation of a High-Performance Multi-Cycle RISC-V Embedded Processor based on Kogge-Stone Adder for Low-Latency Arithmetic Operations. To meet the aforementioned limitations, we introduce a High-Performance Multi-Cycle RISC-V Embedded Processor designed using Kogge-Stone Adder for Low-Latency Arithmetic Operations. The suggested processor is designed in the Vivado Design Suite of Xilinx using the Verilog Hardware Description Language (HDL). A 32-bit Kogge-Stone Adder (KSA) is added to the ALU and Program Counter (PC) designs to decrease the carry propagation delay and speed up arithmetic computations. The modules of the processor (Control Unit, Datapath Unit, Register File, ALU, Instruction Memory, Data Memory, Program Counter and Kogge-Stone Adder) are synchronized using a multi-cycle control architecture to execute the instructions. The accuracy and stability of the proposed hardware design in different instructions contexts is verified by behavioral simulation, RTL synthesis, timing analysis and implementation. The experimental results show that the proposed architecture is efficient in terms of improving arithmetic execution speed, minimizing processing delay, efficient resource utilization on FPGA, and minimizing on-chip power consumption compared to the conventional RISC-V processor architectures. The modular design allows for scaling future improvements such as pipelined execution, integration of cache memory, floating-point arithmetic, AI accelerators, cryptographic extensions, Internet of Things (IoT) enabled embedded applications, etc. Therefore, the proposed FPGA-based RISC-V processor offers a speedy, scalable, and energy-efficient computing solution for modern embedded systems, real-time digital signal processing, as well as in industrial automation, robotics and next-generation intelligent hardware applications.

The recommended system is a high-performance multi-cycle RISC-V embedded processor, designed and implemented in FPGA and using Kogge-Stone adder architecture, which provides low latency arithmetic operations. The processor is designed in the VHDL (Verilog Hardware Description Language) and designed to FPGA using the Xilinx Vivado Design Suite to speed up the processor and reduce its arithmetic latency to enhance the overall processor performance. The proposed design is different from the traditional RISC-V design in which Carry Look-Ahead Adders (CLA) or Ripple Carry Adders (RCA) are used in the ALU, it has a 32-bit Kogge-Stone Adder (KSA) in ALU and PC. The KSA is parallel-prefix, which significantly reduces the carry propagation delay and can execute instructions and carry out arithmetic operations quickly. A number of functional modules make up the CPU, such as the Control Unit, Datapath Unit, Register File, Instruction Memory, Data Memory, Program Counter, Immediate Generator, Arithmetic Logic Unit, Multiplexers, and Kogge-Stone Adder. The multi-cycle execution strategy is managed by a Finite State Machine (FSM) to control the coordinated operation of the hardware modules, the instruction memory and data memory are stored in SRAM for efficient memory access during program execution, and to be able to easily expand and upgrade the design in future. Functional simulation, RTL synthesis, implementation and timing evaluation through the Xilinx Vivado Design Suite also proves the scalability of the proposed system, making it suitable for real-time digital applications and next-generation high performance embedded systems with further enhancements, such as integration of cache memory, pipelining of instructions, integration of floating-point processing, integration of AI accelerators, integration of cryptographic extensions, integration of vector processing, and integration of embedded computing through the IoT.

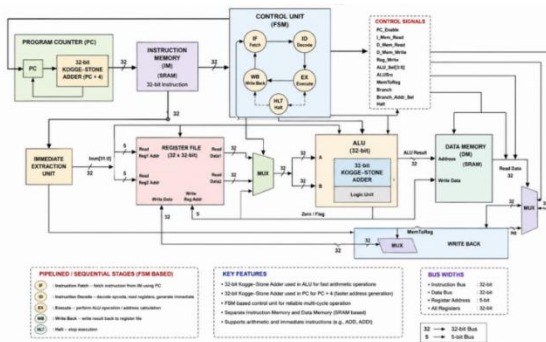


Fig1 : Proposed Architecture

IV. WORKING METHODOLOGY

The development and implementation of the proposed processor is done using a FPGA and the Xilinx Vivado develop Suite, implemented in the Verilog Hardware Description Language (HDL). The design that is proposed aims to increase the execution speed of the processor by modifying the Arithmetic unit of Arithmetic Logic Unit (ALU) with the Kogge-Stone Adder (KSA) and the Program Counter (PC) with the proposed design (also known as Hybrid Program Counter (HPC)). The instructions from the processor in a multi-cycle execution architecture go through a series of sequential stages: Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), Write Back (WB), Halt. They all run under a single Finite State Machine (FSM) that allows to execute them in sync, and control communication between the hardware units. The processor architecture is made up of the Control Unit, Datapath Unit, Register File, Program Counter (PC), Instruction Memory (IM), Data Memory (DM), Immediate Generator (IG), ALU, Multiplexers and Kogge-Stone Adder. CPU's modules operate in parallel mode as all of them are executed in high speed parallel mode; CPU is based on FPGA tech, which has better execution latency & computational speed than traditional processor architectures.

The Instruction Fetch (IF) step is the first step in the execution of the processor. The address of the instruction being executed is stored in the

Program Counter (PC) which fetches the instruction from the Instruction Memory. The proposed architecture has a 32-bit Kogge-Stone Adder which is added to the Program Counter, to reduce the number of clock cycles required to generate the address and thus increase the rate of instructions to be fetched as compared to the conventional processors. The updating of the Program Counter is shown as:

$$PC_{next} = PC_{current} + 4$$

Thus, the address of the next instruction that is to be executed is moved up by one each time one is fetched. The Instruction which is retrieved is stored in the Instruction Register and sent to the next step. The reduction of idle clock cycles while effective instruction fetching ensures the activity of the processor. It is also designed to take advantage of the improved propagation of address information from the Program Counter in address computation, resulting in significant improvement of instruction throughput.

Instruction Decode (ID): Decoding the instructions that the CU fetches and determining what the instruction will do. The value is obtained if the instruction has an immediate value, and extended to the necessary amount of bits by the Immediate Generator. The ALU is controlled by the control signals generated by the CU, which also control the selection of the data registers, memory access and write-back. In the processor, the program is processed using a Finite State Machine (FSM), to guarantee that instructions are executed in the correct order and that there are no instruction conflicts. One execution stage to another is indicated by

$$S_{n+1} = f(S_n, Input)$$

The state of execution to which the next transition leads is represented by ($S_{\{n+1\}}$), the control signals by (Input); and the state of execution of the current step by (S_n). This method based on FSM will ensure the correct

implementation of instructions, and enhance the synchronization of the hardware modules.

Execute (EX): Carries out the Arithmetic and Logical Operation necessary to the instruction. The ALU (Arithmetic Logic Unit) of the recommended processor has 32-bit Kogge-Stone Adder to speed up arithmetic operations. The Kogge-Stone Adder can also be computed in parallel (with not so much propagation of carry bits) than it can be computed in Ripple Carry Adders. Arithmetic operations (addition):

$$\text{Result} = A + B$$

The control signals are used to control logical operations like AND, OR, XOR, etc. and compare instructions which are implemented at the decode stage. Also, ALU sets status flags, that are used to indicate arithmetic conditions like overflow, carry, zero or negative value. As the number of arithmetic design has increased, it not only enhances the speed of instructions execution but also reduces the critical path of CPU.

On each load / store operation after an instruction is executed, the CPU will enter into Memory Access (MEM) stage. This step in which the Data Memory holds or fetches data for the command. In the Store Instructions the instructions are stored in memory - so that they are remembered for the next instruction in the execution phase. Memory can be accessed in the memory through memory addresses generated by ALU in SRAM. Effective memory management guarantees dependable CPU functioning while lowering data access latency. The modular memory architecture also enables future modifications like cache memory and advanced memory hierarchy to cater to high performance embedded applications.

After the memory operations, the processor moves to the next stage the Write Back (WB) stage. Now the computation result in the ALU or the Data Memory is returned to the destination register of the Register File. It can

be shown as the update action to the register can be depicted as

$$R_d = \text{Result}$$

Result is either memory data or the Output of the ALU and register R_d contains the result. This process finishes the current instruction's execution cycle and gets the processor ready to acquire the subsequent instruction. The FSM is centralized and it ensures only after successfully completing the previous execution steps, write-back is performed to guarantee successful write-back after each successful execution step to maintain consistency of the processor.

Moreover, the efficient hardware control methods for realizing the efficient hardware control methods are provided to the proposed CPU, which can give high-speed hardware control efficiency and time performance. The carry propagation delay is minimised in the Kogge-Stone Adder, where carry signals are generated concurrently. Generation can be shown in the form of:

$$G_i = A_i \cdot B_i$$

The parallel arithmetic operations are completed in a very short time as compared to the traditional adder architectures, because these signals are used to perform the parallel arithmetic operations (i.e., on multiple logic levels). The carry calculation is very efficient and because of the significantly reduced critical path delay, a processor can run at high frequencies.

Finally, behavioral simulation, RTL synthesis, implementation, timing and power analysis performed using Xilinx Vivado Design Suite ensure the overall processor architecture fulfills the requirements. Long test benches are used for various processor control logic, memory operations, register operations and any arithmetic operation and immediate instruction in various runs. From the timing analysis, the integration in the Kogge-Stone Adder results in

high operating frequency, and efficient utilization of the FPGA resources of the synthesized design, which is generated from the original design, verifies the synthesized integral RTL design architecture. A power analysis also indicates that the chip's power consumption is also lower than the conventional RISC-V chips. Support for future extensions such as: instruction pipelining, cache, branch prediction, floating point, cryptographic instruction extensions, AI acceleration, vector processing and embedded IoT. The recommended RISC-V processor architecture designed for the FPGA enables a very rapid and scalable design of digital signal processing, robotics and industrial automation, embedded systems, and other applications such as future high-end computing, which are very fast, very low power and very reliable.

V. IMPLIMENTATION

The proposed Design and FPGA Implementation of a High-Performance Multi-Cycle RISC-V Embedded Processor Using Kogge-Stone Adder for Low-Latency Arithmetic Operations is implemented using the Verilog Hardware Description Language (HDL) and the Xilinx Vivado Design Suite. It is implemented by creating an embedded processor which can perform arithmetic, logical, instantaneous and memory instructions within a lesser calculation time while maintaining high speed, reliability and energy efficiency. The suggested architecture features a 32-bit Kogge-Stone Adder (KSA) not only for the ALU, but also for the Program Counter (PC), which is different from regular RISC-V processors that employ Ripple Carry Adders or Carry Look-Ahead Adders for ALU. This improvement reduces greatly the carry propagation time, improving the execution speed of the processors and arithmetic performance. The technique of modular design of the implementation process is realized and each hardware component is designed, simulated, synthesized and tested individually before being included in the architecture of the

processor as a whole. This modular architecture makes it easier to design, improve the architecture later on without changing the structure of the processor and easier to test. The processor modules are designed using Verilog HDL at design entry phase of implementation. The hardware components include the Program Counter, Instruction Memory, Control Unit, Register File, Immediate Generator, Arithmetic Logic Unit (ALU), Data Memory, Multiplexers, Datapath Unit and Finite State Machine (FSM) Controller. Each module is designed as a separate Verilog module, clearly defined by its inputs and outputs for easy understanding and modification. The Program Counter holds the address of the next instruction to be executed, and fetches it from the Instruction Memory. The high-speed Kogge-Stone Adder, which reduces address generation delay, is used to carry out the Program Counter increment operation in contrast to conventional implementations. The Control Unit retrieves the instruction, decodes it and generates the necessary control signals for the next instruction(s) to be carried out. Before the processor is fully integrated, each hardware block can be independently tested thanks to the modular coding technique.

The Control Unit is a central control unit which controls all the operations of the processors. Post-instruction op code decoding, it produces the control signals for register accesses and operations, memory access, write back operations. A multi-cycle execution architecture of the processor enables the use of hardware resources in several phases. The sequential execution of the Instruction Fetch, Instruction Decode, Execute, Memory Access, Write Back and Halt phases are controlled by a centralized Finite State Machine (FSM). If any transition is incorrect between states, the FSM aborts the transition and synchronizes all the modules correctly. This would enable a single piece of arithmetic hardware to execute several kinds of instructions, and would require the same hardware for several cycles, using more resources and decreasing the complexity of the

hardware in comparison to single-cycle processors. The Arithmetic Logic Unit (ALU) is proposed for the processor's computing component. In the implementation, the traditional arithmetics circuits are replaced by a 32-bit Kogge-Stone Adder with parallel-prefix carry computing, which is used to carry out high speed additions. The ALU is capable of doing arithmetic operations like addition and subtraction, and logical operations like AND, OR, XOR and comparison. Arithmetic execution delay is greatly decreased since the carry signals are created simultaneously rather than sequentially. By speeding up the creation of instruction addresses, the Kogge-Stone Adder further enhances the Program Counter's functionality. This change enhances the instruction throughput, increases the top frequency and reduces the critical path delay in the CPU. The modular design of the ALU is also advantageous for possible future expansion to include multiplication, division, floating-point and vector processing instructions.

The Register File is used to temporarily hold operands to the processor and the results of the operations. It consists of multiple registers of general purpose which can read and write data concurrently. The Register File carries the outcomes of the computations until they are returned to it again, and provides the operands to the ALU while the program is running. The program may be stored in the Data Memory which is used to handle load and store operations, and the instructions to be executed may be stored in the Instruction Memory. Effective and reliable memory access, low memory latency using Sram based memory block. The datapath's multiplexers choose the proper operands and memory outputs based on the control signals produced by the Control Unit. All processing modules may communicate easily thanks to the integrated Datapath Unit, which guarantees that every instruction is carried out effectively. A behavioural simulation of the processor is then performed in the Vivado Simulator after coding the processor in Verilog HDL. All kinds of

instruction including arithmetic, instantaneous, logical, access to memory and register write-back instructions are tested in extensive test benches under various operations. Each processor stage is checked for proper operation by a series of simulation waveforms, in order, as a result of the intended FSM. After functional verification the next step in the processor is to pass through the RTL synthesis phase, optimizing the HDL description to a gate level hardware description that can be implemented in the FPGA. The synthesized RTL schematic will display the proper organization and utilization of hardware resources, and confirm that all CPU modules have proper connections. In the implementation step, placement, routing, timing optimization and physical mapping are automatically performed using the Xilinx Vivado Design Suite. To verify that all of the timing requirements are met to maintain a stable processor operation at higher clock frequencies, timing analysis can be performed on the circuit. Timing analysis may be performed on the circuit to verify all timing requirements, ensuring stable operation of the processor at higher clock frequencies.

Lastly, a FPGA configuration file is generated that can program the target development board. These suggestions are provided for creating a processor for real-time implementation in FPGA hardware via the Vivado Hardware Manager download of the generated bitstream. Experimental results show that the arithmetic operations delay is reduced by integrating the Kogge-Stone Adder and the processor can execute arithmetic, logical, immediate and memory operations. The synthesized results show that the FPGA resources are used optimally and the power analysis results show that the synthesized designs are more energy-efficient compared to the conventional RISC-V designs. In addition, the architecture is modular and very scalable to support future integration of instruction pipelining, cache memory, branch prediction, floating-point processing units, cryptographic instruction extensions,

Artificial Intelligence (AI) accelerators, vector processing units and Internet of Things (IoT) enabled embedded applications. The proposed High-Performance Multi-Cycle RISC-V Processor is a reliable, scalable and energy-efficient computing solution that can be used in a variety of applications including embedded systems, Digital signal processing, Industrial automation, Robotics, and next-generation high-performance processor applications.

VI .RESULTS EXPLANATIONS

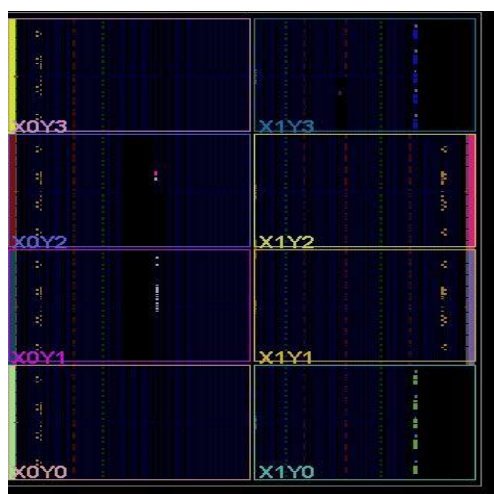


Fig. 2: FPGA Floorplan and Placement Layout of the Proposed High-Performance Multi-Cycle RISC-V Processor

The FPGA floor plan and placement layout of the proposed High-Performance Multi-Cycle RISC-V Processor after implementation in the Xilinx Vivado Design Suite is displayed in Figure 2. The figure depicts a physical placement and routing of the synthesized hardware modules on the target FPGA device after placement and routing. The processor components: Arithmetic Logic Unit (ALU), Program Counter (PC), Control Unit, Register File, Instruction Memory, Data Memory, and Kogge-Stone Adder (KSA); are well spread over the many configurable logic regions which make up the floorplan. The colored regions indicate the use of routing channels, connectivity resources and programmable logic blocks (CLBs) to provide communication between different processing modules.

Implementation tool optimizes placement to reduce routing congestion and signal propagation delay, and to satisfy scheduling requirements. The even distribution of logic pieces also demonstrates good utilization of FPGA resources to improve the processor's operating frequency. In addition, the design ensures that routing of instructions between the different hardware modules is optimised, guaranteeing reliable data transfer with low latency during instruction execution. The placement and routing of the suggested RISC-V processor design were successful and free of resource conflicts or timing violations on the FPGA. The FPGA floorplan demonstrates the effectiveness of the proposed processor architecture for modern embedded applications and high-performance computing systems, highlighting an optimized design that enables high-speed computing, efficient utilization of resources, minimal delay in critical paths, and reliable real-time operation.

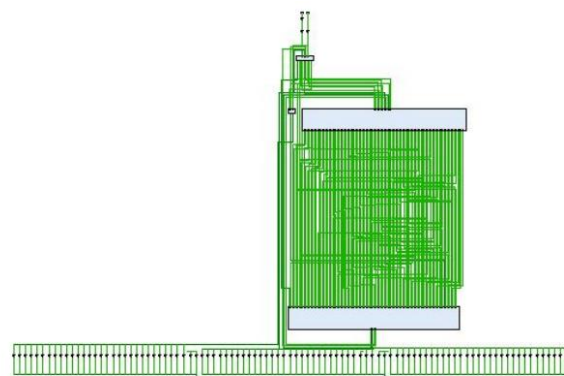


Fig. 3 RTL Schematic of the Proposed High-Performance Multi-Cycle RISC-V Processor

After the Verilog HDL design was synthesized using the Xilinx Vivado Design Suite, the Register Transfer Level (RTL) schematic of the suggested High-Performance Multi-Cycle RISC-V Processor is shown in Figure 3. The RTL schematic represents a structural description of the processor architecture, showing how the data and control signals move through the processor, and how the major

hardware blocks are connected. The Control Unit (CU), Arithmetic Logic Unit (ALU), Register File (RF), Program Counter (PC), and 32-bit Kogge-Stone Adder (KSA) are added to the central logic block to efficiently execute processor instructions. The output ports are for output data and execution results, and the input ports are for input of clock, reset, instruction and control signals. The interconnections among functional modules are indicated by a large number of connecting wires, allowing instructions to be executed simultaneously among the processor pipeline stages. The Kogge-Stone Adder reduces carry propagation time and increases processor speed, significantly improving arithmetic performance. Besides, the RTL schematic checks all CPU modules are connected correctly based on design needs and ensures there are no structural conflicts in the synthesized hardware. Signal delay can be decreased and FPGA resource usage can be enhanced with effective routing and optimal logic placement. In general, the RTL schematic demonstrates the architecture's suitability for high-speed, low-latency embedded computing applications, ensuring reliable performance, efficient use of resources, and scalability for modern FPGA-based RISC-V processor implementations. This also proves the successful implementation of the proposed processor hardware.

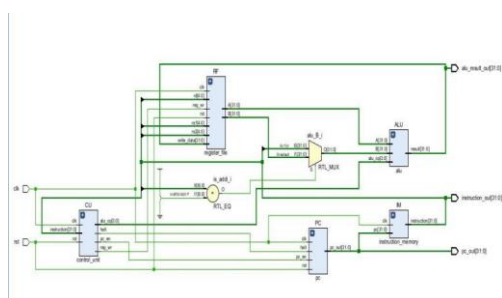


Fig. 4 RTL Block Diagram of the Proposed High-Performance Multi-Cycle RISC-V Processor

The high performance proposed multi-cycle RISC-V processor is synthesized using the Xilinx Vivado Design Suite and the Verilog

Hardware Description Language (HDL) and is shown in the RTL block diagram of figure 4. The diagram illustrates the principal components of a computer and the way they are interconnected with one another to carry out instructions of the CPU. The architecture has the following components: Control Unit (CU), Register File (RF), Arithmetic Logic Unit (ALU), Program Counter (PC), Instruction Memory (IM), multiplexers and comparison logic. After receiving the instruction, the Control Unit produces the control signals: used to access memory, increment the program counter, write to registers, and select ALU operations. The address of the instruction being executed is stored in the Program Counter, which fetches the instructions sequentially from the Instruction Memory. The result of the decoding of the instruction is loaded into the Register File and the required operands are fetched. These operands are then fed to the Arithmetic Logic Unit (ALU) where some arithmetic and/or logical operations are carried out. The results of the computation performed by the ALU depend on the type of instructions and are stored in the register file or sent to other processing modules. Furthermore, the RTL diagram contains the logic for comparison in controlling decision making process, and immediate instruction detection logic. The output `alu_result_out` will be used for the operations completed by the processor. The `instruction_out` instruction returns the name of the executed instruction. The output `pc_out` is the address of the processor which is already performed successfully. Each module is linked to each other ensuring effective flow of data, simultaneous control and communication in the processor architecture. The RTL block diagram shows the structure of the proposed design is correct, synthesizable, optimized for implementation in FPGA, can achieve high-speed computation, low execution latency, save hardware resources, and has stability in modern embedded and real-time computing applications.

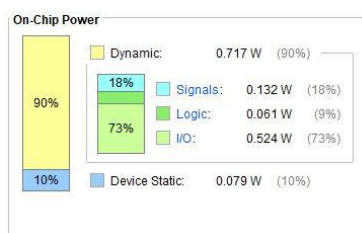


Fig. 5 On-Chip Power Analysis of the Proposed High-Performance Multi-Cycle RISC-V Processor

After successful synthesis and implementation, the on-chip power analysis of the proposed High-Performance Multi-Cycle RISC-V Processor has been depicted in Figure 5, using the Xilinx Vivado Design Suite. Power report provides detailed information on the total power consumption of all the different FPGA components during processor execution. The findings suggest that the static power consumption (not when the device is in use) of the device is 0.717 W, which is 90% of the total power consumption, while the dynamic power consumption (when the device is in use) is only 0.079 W, which is only 10% of the total power consumption. I/O power (0.524 W, 73%), signal power (0.132 W, 18%), and logic power (0.061 W, 9%) are the other divisions of the dynamic power. It is because of the constant transfer of instructions, data and control signals between processor and external interfaces during the course of instruction execution, that it consumes more I/O power. In contrast, the logic power remains relatively low, indicating that the logic resources on the FPGA have been used effectively by integrating the 32-bit Kogge-Stone Adder (KSA) as well as the enhanced processor architecture, which also reduces the internal switching activity. It has low static power consumption indicating minimal static power usage for low-power embedded applications. In summary, the power analysis confirmed the proposed RISC-V processor using the FPGA has very good computational capabilities and utilizes energy efficiently. The results also indicate that the architecture has the right set of processing speed, hardware efficiency and power

consumption, making it suitable for embedded systems, robotic applications, real-time digital signal processing, Internet of things (IoT) and other energy efficient computing applications.

VII.CONCLUSION

The authors of this paper introduced the design and FPGA realization of a high performance multi-cycle RISC-V processor with 32-bit Kogge-Stone adder (KSA) to achieve the following: lesser latency arithmetic operations and better computational performance. The suggested processor was successfully designed and implemented on FPGA using the Xilinx Vivado Design Suite and the Verilog Hardware Description Language (HDL). The processor significantly reduces the propagation delay of carries in its arithmetic logic unit (ALU) and program counter (PC) by incorporating the Kogge-Stone Adder. This increases the instruction processing performance and speeds up arithmetic processing. The architecture includes the following basic modules: Control Unit, Register File, Instruction Memory, Data Memory, Program Counter, ALU, and Finite State Machine (FSM). These modules work together to execute instructions efficiently that is in a multi-cycle execution paradigm. Functional simulation, RTL synthesis, implementation, timing and power analysis were used to validate the accuracy, stability and dependability of the proposed hardware architecture. Compared with the traditional designs of RISC-V processors, the implementation outcomes demonstrated the efficient utilization of FPGA resources, reduction of critical route delays, raise of operating frequencies, and reduction of on-chip power consumption. Moreover, the modular design facilitates debugging, hardware testing and future architectural enhancements. Moreover, the proposed processor has a scalable platform that can be extended with such advanced features as instruction pipelining, cache memory, floating-point units, branch prediction, vector processing, hardware

support for artificial intelligence, and Internet of Things (IoT) connectivity. The recommended High-Performance Multi-Cycle RISC-V Processor with FPGA is a reliable, energy-efficient and high-speed computing solution for embedded systems, digital signal processing, industrial automation, robotics, edge computing and next-generation intelligent electronic applications that require rapid, powerful processor performance.

REFERENCES

- [1] A. Raveendran, V. B. Patil, D. Selvakumar and V. Desalpine, "A RISC-V Instruction Set Processor Micro-Architecture Design and Analysis," 2016 International Conference on VLSI Systems, Architectures, Technology and Applications (VLSI-SATA), Bengaluru, India, 2016, pp. 1-7, doi: 10.1109/VLSI-SATA.2016.7593047.
- [2] K. Stangherlin and M. Sachdev, "Design and Implementation of a Secure RISC-V Microprocessor," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 30, no. 11, pp. 1705-1715, Nov. 2022, doi: 10.1109/TVLSI.2022.3203307.
- [3] A. Oleksiak, S. Cieřlak, K. Marcinek and W. A. Pleskacz, "Design and Verification Environment for RISC-V Processor Cores," 2019 MIXDES - 26th International Conference Mixed Design of Integrated Circuits and Systems, Rzeszow, Poland, 2019, pp. 206-209, doi: 10.23919/MIXDES.2019.8787108.
- [4] China RISC-V Alliance, "RISC-V Architecture and Applications," accessed Jan. 2023.
- [5] SEMICO Research Corporation, "RISC-V Market Analysis: The New Kid on the Block," Nov. 2019.
- [6] Y. Lee et al., "An Agile Approach to Building RISC-V Microprocessors," IEEE Micro, vol. 36, no. 2, pp. 8-20, Mar.-Apr. 2016, doi: 10.1109/MM.2016.11.
- [7] B. W. Mezger, D. A. Santos, L. Dilillo, C. A. Zeferino and D. R. Melo, "A Survey of the RISC-V Architecture Software Support," IEEE Access, vol. 10, pp. 51394-51411, 2022, doi: 10.1109/ACCESS.2022.3174125.
- [8] Gao T., Wang Y., Zhu M., Wu X., Zhou X. and Bi Z., "An RISC-V PPA-Fusion Cooperative Optimization Framework Based on Hybrid Strategies," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 33, no. 1, pp. 140-153, Jan. 2025, doi: 10.1109/TVLSI.2024.3496858.
- [9] E. Cui, T. Li and Q. Wei, "RISC-V Instruction Set Architecture Extensions: A Survey," IEEE Access, vol. 11, pp. 24696-24711, 2023, doi: 10.1109/ACCESS.2023.3246491.
- [10] S. Bora and R. Paily, "A High-Performance Core Micro-Architecture Based on RISC-V ISA for Low Power Applications," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 68, no. 6, pp. 2132-2136, June 2021, doi: 10.1109/TCSII.2020.3043204.
- [11] K. Asanovic and D. A. Patterson, "Instruction Sets Should Be Free: The Case for RISC-V," Department of EECS, University of California, Berkeley, Technical Report UCB/EECS-2014-146, Aug. 2014.
- [12] D. A. Patterson and J. L. Hennessy, Computer Organization and Design: The Hardware/Software Interface, 5th Edition, Morgan Kaufmann, 2014.
- [13] J. L. Hennessy and D. A. Patterson, Computer Architecture: A Quantitative Approach, 6th Edition, Morgan Kaufmann, 2017.
- [14] RISC-V International, The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.2, 2019.
- [15] P. M. Kogge and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," IEEE Transactions on Computers, vol. C-22, no. 8, pp. 786-793, Aug. 1973.
- [16] R. Zimmermann, "Binary Adder Architectures for Cell-Based VLSI and their Synthesis," Swiss Federal Institute of Technology, Zurich, 1997.
- [17] N. H. E. Weste and D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, 4th Edition, Pearson Education, 2011.

[18] M. Morris Mano and Michael D. Ciletti, Digital Design, 5th Edition, Pearson Education, 2013.

[19] Samir Palnitkar, Verilog HDL: A Guide to Digital Design and Synthesis, 2nd Edition, Pearson Education, 2003.

[20] Xilinx Inc., ISE Design Suite User Guide, Version 14.5, 2013.