ORIGINAL ARTICLE

An SDPF RISC-V Processor With 55.9% Dhrystone Improvement Using Two-Stage Pseudo-Pipelined Architecture for IoT Applications

Wenji Mo | Jingjing Liu | Yuchen Wang | Feng Yan | Bingjun Xiong | Jian Guan

School of Electronics and Communication Engineering, Sun Yat-Sen University, Shenzhen, China

Received: 30 July 2024 | **Revised:** 12 February 2025 | **Accepted:** 26 February 2025

Funding: This work was supported by the National Natural Science Foundation of China (62174181).

Keywords: IoT | low area | low power | pseudo pipelined | RISC-V | serial data path

ABSTRACT

Embedded Internet of Things (IoT) nodes are required to perform lightweight tasks such as information monitoring and simple signal processing. A crucial component of low-power embedded IoT sensors is the low-power processor. Due to the constraints of operating conditions, there are stringent requirements on the power consumption and area of the processor. To minimize the power and area, this paper proposes a low-power RV32I processor based on the RISC-V instruction set architecture (ISA), which adheres to a serial data path. To enhance the energy efficiency of the serial data path followed (SDPF) processor, this paper proposes a pseudo-pipeline architecture. By partitioning and combining certain instruction lifecycle tasks, a two-stage pseudo-pipeline structure is implemented, thereby reducing the cycles per instruction (CPI) of the SDPF processor. The proposed processor is designed using Verilog HDL, and FPGA prototype validation demonstrates that the proposed processor achieves at least 18% fewer resource compared with the traditional parallel 32-bit RISC-V processors and 55.9% performance improvement compared with the previous SDPF processors. The proposed processor is implemented using a standard 0.18-µm CMOS process. The post-layout simulation results indicate that it has at least 9.6% less area and 37.5% lower dynamic power consumption compared with traditional parallel 32-bit processor. Additionally, it achieves a 40.9% increase in the performance to power ratio compared with the previous SDPF RV32I processor.

1 | Introduction

The rise of the Internet of Things (IoT) has driven an increased demand for information monitoring sensor devices, such as wearable medical devices and natural disaster monitoring equipment. Among various IoT information monitoring devices, energy harvesting systems have drawn much attention because they can increase the sensor life time. With the rapid development of integrated circuit technology, sensors with low power microprocessor can autonomously operate in conjunction with energy harvesting systems, forming intelligent self-powered systems. System-in-fibers (SiF) are one such IoT sensor nodes designed specifically for energy harvesting applications [1, 2]. By integrating small intelligent self-powered systems into clothing fibers, it is possible to continuously monitor physiological signals and environmental conditions over extended periods without the need for frequent battery replacements. However, the design of intelligent self-powered IoT systems, such as SiF sensors, poses certain challenges. On the one hand, these systems must meet miniaturization requirements to allow for embedding into fibers or various scenarios. On the other hand, environmental changes and physical displacement can lead to significant fluctuations in harvested energy. As one of the core components of intelligent self-powered IoT sensor nodes, microprocessors not only need to meet the requirements of low area overhead but also exhibit extremely low power consumption. Therefore, it is essential to design a miniaturized, low power microprocessor to suit this self-powered system.

Thus far, various low-power processors and techniques have been reported to optimize the energy efficiency. From an energy perspective, reducing the supply voltage to near the threshold voltage has been demonstrated as an effective method to lower the power consumption [3–6]. However, further reducing the supply voltage is no longer effective in lowering power consumption, as leakage power consumption becomes the dominant factor. To optimize the leakage power, a dynamic leakage-suppression (DLS) logic has been proposed and implemented [7–10]. By integrating additional constraint transistors into the basic logic unit, the leakage current can be mitigated effectively. However, these approaches have certain drawbacks. On the one hand, excessively low supply voltage severely limits the operating frequency of the system [11]. On the other hand, a considerable timing margin is required to achieve an error-free execution [12]. Moreover, the DLS and its derivative logic necessitate the customization of standard cell libraries, complicating the design process. In energy harvesting based self-powered IoT systems, another effective method for controlling the power consumption is to utilize the power management unit (PMU) to regulate the energy draw of the processor during non-harvesting modes [13–15]. A recent report has introduced a triple mode PMU, which implements energy sensing, performance sensing, and minimum energy point tracking at the same time [16]. This approach allows the minimum energy point voltage to be the minimum power supply voltage required for the processor operation, thereby maintaining the processor at its lowest power consumption level.

From the perspective of processor microarchitecture, retaining only the RV32I integer instruction set is a crucial approach to avoid the additional hardware overhead associated with implementing extra instruction sets. Furthermore, various microarchitecture designs can be employed to reduce the processor's power consumption and area. A single-cycle design, which processes a single instruction within one clock cycle, can reduce the use of flip-flops and effectively decrease power consumption; however, it may result in timing criticality of the critical path [17]. Another strategy involves using a multi-cycle non-pipelined design, where instructions are processed over several cycles [18–20], thereby reducing the toggle rate of data paths and lowering dynamic power consumption. Nevertheless, this method introduces more registers, thereby increasing the chip area. Besides, a novel approach involves designing processors based on custom instruction sets [21, 22], thereby reducing the data transmission bit-width. Compared with the standard 32-bit or 64-bit processors, the 8-bit or 16-bit processors occupy fewer registers, thus reducing power consumption. However, the mainstream compilers are predominantly designed for 32-bit or 64-bit processors, making them challenging to adapt to custom instruction sets with lower bit widths. This presents a significant challenge for embedded software design, consequently extending the design cycle.

On the other hand, the traditional processor designs typically employ parallel data path transmission techniques, limited by the number of registers and size of arithmetic units, thereby restricting their ability to reduce power consumption and minimize area. Conversely, the serial data path followed (SDPF) processor present an attractive alternative [23]. Processing instructions within a serial data path significantly reduces the use of registers. Compared with the parallel data path followed (PDPF) processors, this method achieves lower power consumption and smaller area footprint, making it suitable for low-frequency, low-power applications. However, in the RV32I benchmark [24], converting 32-bit wide data into serial format and transmitting it through a serial path require at least 32 cycles, resulting in a significantly higher cycles per instruction (CPI) than conventional pipelined processors and degrading core performance.

To address the aforementioned issues, this paper proposes an improved SDPF RV32I processor, which adopts a pseudo two-stage pipeline structure. By employing additional shift registers (SRs), the processor partitions and merges part of the tasks within an instruction cycle into two stages of equal clock cycle duration, thus simulating a two-stage pipeline structure and reducing the CPI of the proposed SDPF processor. While providing superior performance, the proposed design maintains the characteristics of low power consumption and small area, thereby improving the energy efficiency ratio of the SDPF processor. The structure of this paper is arranged as follows: Section 2 discusses the SDPF processor and the proposed SDPF processor with a pseudo two-stage pipeline, Section 3 elaborates on the microarchitecture design details of the proposed processor, Section 4 presents the FPGA prototype verification results and post layout simulation results, and Section 5 concludes the paper.

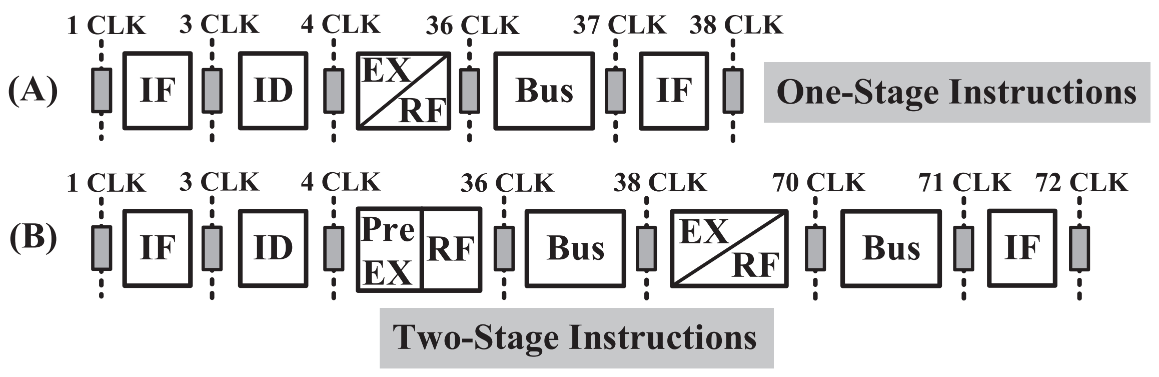


FIGURE 1 | Instruction processing flow of SERV: (A) one-stage instructions and (B) two-stage instructions.

2 | Design of the Proposed SDPF RISC-V Processor

In the literature, the RISC-V processors following a serial data path were first introduced in an open-source RISC-V processor SERV [23]. In SERV, multiple SRs are employed to convert the data required for computation into a serial format and transmit it to various processor modules for execution. The core processes only one bit of data per clock cycle, and the execution results are immediately written back to the register file (RF) in a serial manner. Furthermore, SERV employs either a single-stage or a two-stage method for processing instructions, depending on the instruction type in RV32I. Figure 1A illustrates the single-stage instruction processing flow in SERV, which is used for the instructions that require computation on a single set of data signals throughout their lifecycle, such as ADDI and LUI instructions. The core completes the instruction fetch (IF) and instruction decode (ID) phases in 3 clock cycles and spends 32 clock cycles performing the required operations of the instruction. Figure 1B depicts the two-stage instruction processing flow, which is used for the instructions that depend on preconditions for execution (EX), such as branch type and comparison type instructions. Compared with the single-stage instruction processing, the two-stage instruction processing incurs an additional 32 clock cycles for preliminary data handling, such as condition judgment or memory address calculation. The microarchitecture of SERV is similar to the multi-cycle non-pipelined design in the traditional processors, which necessitates the complete execution of the current instruction before fetching a new one, leading to further significant performance degradation.

This paper proposes a pseudo-pipeline architecture to optimize core implementation and enhance the performance of the SDPF RISC-V processor. Figure 2 illustrates the proposed pseudo two-stage pipeline structure. Similar to SERV, the proposed processor core selects either structure A or B for instruction processing based on the specified RV32I instruction type. Both structure A and structure B initiate the instruction processing flow at the IF phase. Each IF and ID tasks require one clock cycle. The core determines the instruction type based on the decode results; if the current instruction is a conditional dependency type, it is processed according to structure B. All other instruction types follow the processing flow of structure A. In Structure A, the core determines the sequence of subsequent execution tasks based on the decode results. If the current instruction is non-source-register dependent, it directly enters the EX phase; otherwise, it must first perform RF reading operations. The RF reading, EX, and write-back (WB) tasks are embedded within the same time frame, with the RF transmitting data bit-by-bit starting from the least significant bit (LSB). Two cycles are required for the first

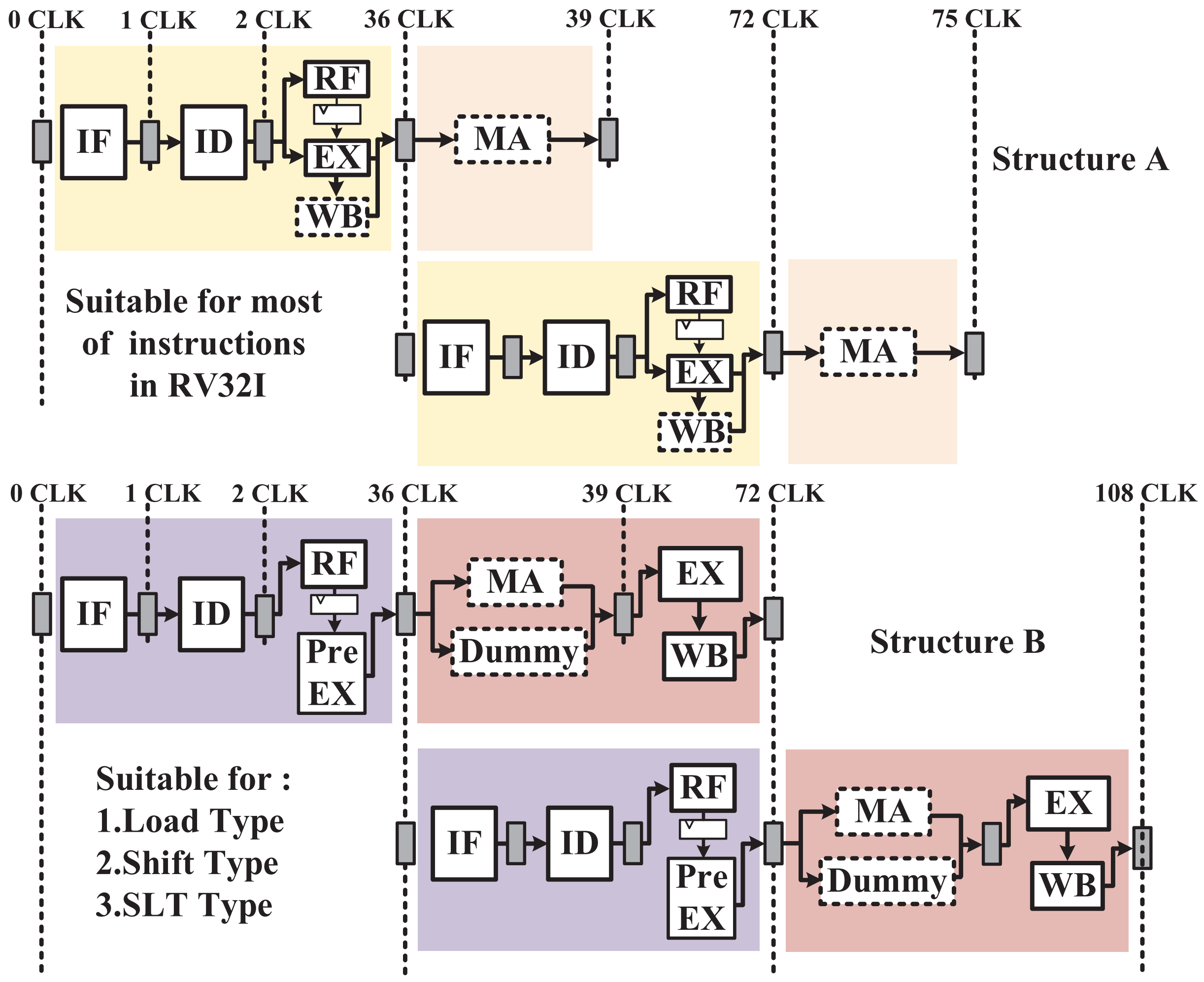


FIGURE 2 | The two-stage pseudo pipeline structure of the proposed SDPF RISC-V processor.

bit of data to reach the EX module. During this phase, the core performs bit-wise computation on the serially input data in each clock cycle and continuously writes the new EX results bit-by-bit back into the RF, which will take a total of 32 clock cycles. However, the WB phase is not necessary for every instruction under the pseudo-pipeline structure A. WB is only bypassed when handling branch-type instructions and store-type (S-type) instructions. Consequently, the WB phase in structure A is represented by dashed lines in Figure 2. After completing the bit-wise WB, the pipeline enters the second stage, which solely consists of memory access (MA) in structure A. According to the RV32I specification, only load-type (L-type) and S-type instructions involve memory operations. Therefore, the MA phase is implemented only when processing S-type instructions in structure A. The MA phase requires three clock cycles: Two clock cycles are utilized for accessing to the bus and completing handshake with the data memory (DMEM), while the third cycle is used to write the data into DMEM. If the current instruction is not an S-type instruction, the MA phase will be skipped.

Structure B employs a pseudo two-stage pipeline when processing L-type, shift-type, and set-less-than (SLT) type instructions. Unlike Structure A, after the RF reading in the first stage of the pseudo-pipeline, it requires a pre-execution (Pre-EX) step, with the EX positioned in the second stage of the pipeline. The Pre-EX is used to compute data that will be utilized in subsequent pipeline stages. In this phase, if a L-type instruction is being processed, the core calculates the address signal for MA; if a SLT-type instruction is being processed, the core compares the two specified numbers. For the shift-type instructions, although conditional judgment is not required, 32 clock cycles are needed prior to the shift operation, in which the core fetch the data to be shifted and store it in an SR. The specified shift operation is then executed in the second pipeline stage based on this SR. The second stage of the pseudo-pipeline encompasses three tasks: MA, EX, and WB. Among the three types of instructions processed under Structure B, only the L-type instructions require the MA. Similar to the S-type instructions, L-type instructions also require three cycles to complete data retrieval. To maintain consistent pipeline timing with L-type instructions, three dummy clock cycles are inserted during the original MA timeframe when processing the shift-type and SLT-type instructions. In Structure B, the EX and WB tasks are integrated into the same phase, where the result obtained during the EX phase is immediately written back to the RF in a serialized manner.

The fundamental difference between Structure B and Structure A lies in their handling of conditional dependency instructions. Structure A is unable to perform both conditional computation and WB data calculation simultaneously within the serial data path. To maintain the same pipeline architecture, the two phases of Pre-EX and EX in Structure B, that consume the most clock cycles, are placed in separate stages of the pseudo-pipeline. For both Structure A and Structure B, as the earlier processed instructions enter the second stage of the pseudo-pipeline, the next instructions are fetched and a new instruction cycle starts. This cyclical process of handling instructions continues, forming a pseudo-pipeline structure.

CPI is a crucial parameter for evaluating processor performance, reflecting the number of clock cycles required for a processor to execute a single instruction. For a scalar RV32I processor with sequential instruction fetching, the CPI can be expressed by the following equation:

$$CPI = \frac{\sum \left(cpi_k \times f_k\right)}{\sum f_k} \tag{1}$$

where cpi_k represents the average number of cycles required to process the kth type of instruction, and f_k represents the frequency of the kth type of instruction within a given program. Therefore, the numerator represents the total number of cycles spent executing the program, while the denominator represents the total number of instructions in the program. Given that the SERV processor processes instructions using either a single-stage or a two-stage approach, and its design resembles a multi-cycle non-pipelined structure, the CPI of the SERV processor can be expressed as followed:

$$CPI_{SERV} = \frac{37f_1 + 71f_2}{f_1 + f_2} = 37 + \frac{34}{1 + \frac{f_1}{f_2}} \tag{2}$$

Here, f_1 represents the number of single-stage instructions and f_2 represents the number of two-stage instructions. In practice, f_2 is nonzero. As it can be inferred from Equation (2), the CPI of the SERV processor is influenced by the ratio of different types of instructions. The minimum CPI of the SERV processor exceeds 37, and if $f_1 = f_2$, the CPI can reach as high as 54.

On the other hand, for the proposed pseudo two-stage pipelined processor, starting from the 36th cycle after power on, an instruction completes the EX or WB phase every 36 cycles, which means that the lifecycle of an instruction is completed. Therefore, under ideal conditions, the CPI of the proposed processor is 36. This implies that the proposed pseudo two-stage pipelined processor exhibits a lower and more stable CPI, thereby achieving higher instruction processing efficiency compared with the SERV processor.

3 | Design of the Microarchitecture

3.1 | The Data Path of the Proposed Processor

The data path architecture based on the proposed pseudo-pipeline structure is illustrated in Figure 3. The modules with a blue background indicate that data will be processed and computed in a serial manner, while modules with a red background indicate that data is stored or computed in a 32-bit format. Thick data lines in the figure represent data transmission in a 32-bit parallel manner, whereas thin data lines represent data transmission in a 1-bit serial manner. The purple module indicates RF reading operations, while the yellow module represents RF writing operations.

The lifecycle of an instruction begins at the IF phase, where the Process Counter (PC) Unit sends the address signal to the instruction memory (IMEM). The address signal is computed based on the SR and is sent to the IMEM in a 32-bit parallel format. The IMEM responds within one cycle and sends the corresponding instruction signal to the Decode Unit. The instruction

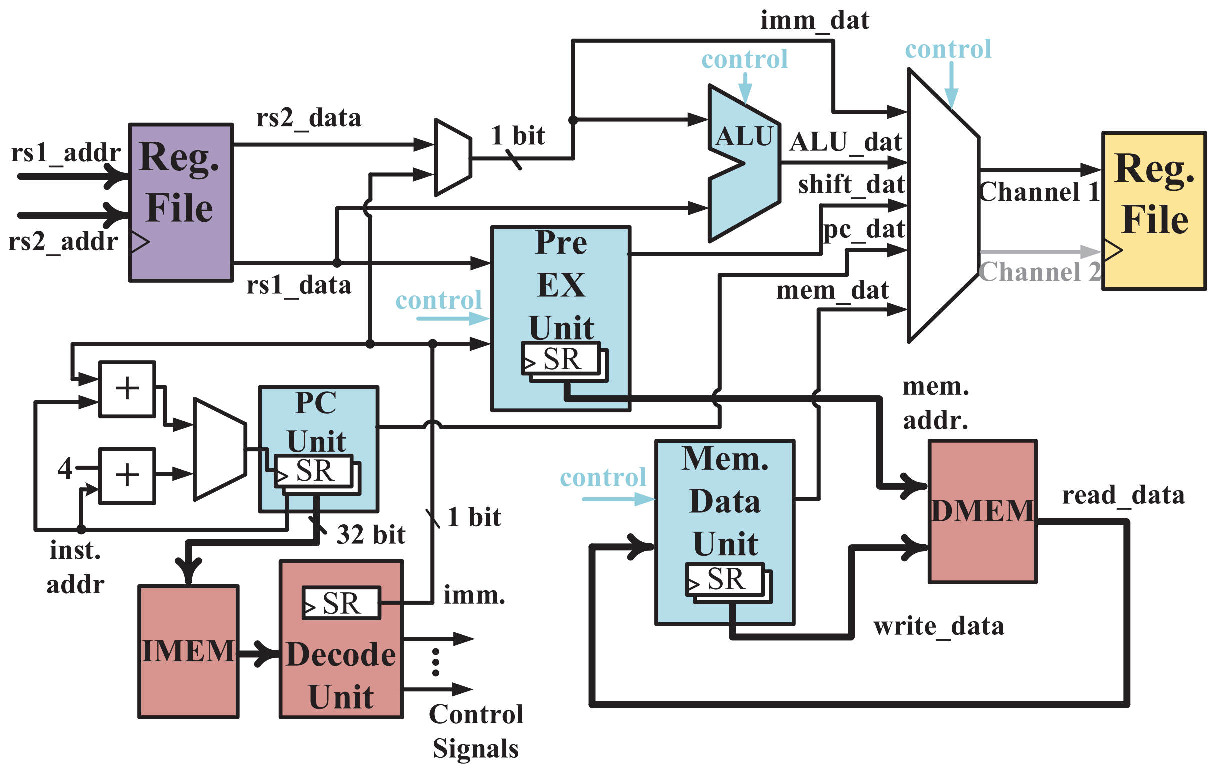


FIGURE 3 | The data path of the proposed SDPF RISC-V processor.

is decoded in parallel within one clock cycle. The Decode Unit converts the immediate value contained in the instruction signal and serially transmits them to the data path through the SR for subsequent computations. Additionally, based on the decoded instruction type, the Decode Unit generates a series of control signals to drive the operation of the remaining core modules. The changes in these control signals depend on the instruction type and the pipeline stage, indicating the operation state of each module during the current processing stage. The use of pseudo-pipeline Structure A or Structure B is also governed by the control signals from the Decode Unit, which transmits enable signals based on the respective instruction type. The enable signal for structure A must be maintained for at least 34 clock cycles, whereas the enable signal for structure B must be maintained for 70 clock cycles. Each module executes the corresponding operations based on these control signals. Since the instructions processed using Structure B are more time-consuming, certain control signals must occupy additional registers to remain valid until the instruction processing is complete. For the instructions requiring access to the RF, the Decode Unit sends the access address signal to the RF in parallel to retrieve the corresponding data. To minimize the control logic overhead, the RF is implemented by dividing each required 32-bit register into thirty-two 1-bit registers, which are indexed by specific addresses. Consequently, the read and write operations of the RF are performed serially.

The Arithmetic Logic Unit (ALU) primarily consists of 1-bit Boolean operation units and adders, while the Pre-EX Unit mainly comprises 32-bit SRs. During the EX phase, only the ALU is involved in computations, performing bitwise calculations and directly writing the results back to the register file serially. In the Pre-EX phase, both the ALU and the Pre-EX Unit collaborate, with the ALU performing condition judgments or memory address calculations, and the Pre-EX Unit shifting the computed address signals into the SR for parallel conversion. Specifically, for shift-type instructions, since data is input from the LSB and the number of shifted bits needs to be determined, during the Pre-EX phase, the data from the register file is shifted into the SR within the Pre-EX Unit, awaiting further shifts during the pipeline's EX phase.

The MA operations are primarily handled by the Pre-EX Unit and the MEM Data Unit. As previously discussed, the address for MA is generated by the SR in the Pre-EX Unit, while the data required for S-type instructions is generated by the SR in the MEM Data Unit. The core communicates with the DMEM by sending address signals, data signals, and handshake signals via the bus. In this system, the DMEM consists of a static random-access memory (SRAM) directly connected to the bus. Here, DMEM is not a cache and just set for data storage. This results in an ideal scenario; that is, it takes two clock cycles from the handshake initiated by the core to the response received from DMEM. Additionally, when processing L-type instructions, the data returned from the DMEM is loaded into the SR of the MEM Data Unit, converted to serial output and awaits written back to the register file. All data requiring written back to the RF is connected to a multiplexer controlled by signals generated from the Decode Unit. This multiplexer selects the appropriate data to be written back to the register file based on the instruction type.

During the EX and Pre-EX phases, the PC Unit concurrently performs serial calculations for the next fetch address. Specifically, if the current instruction being processed is a conditional branch instruction, the PC Unit calculates the sequential fetch address and the branch address simultaneously using two SRs. Upon completion of the EX, the PC Unit determines whether a branch should be taken and outputs the corresponding address signal. Additionally, it updates the signals stored in the two SRs to the address of the current instruction after fetching a new instruction, ensuring the base address for subsequent address calculations is accurate.

In the data path, based on different instruction types, the decode unit generates various control signals. Compared with the SERV processor, which also employs a serial data path, the proposed architecture requires additional control logic to implement the pseudo pipeline, primarily for the realization of pseudo pipeline structure B. In the control unit of the proposed processor, two signals, phase1 and phase2, are introduced to indicate the Pre-EX or EX stages. These signals are organized into two sets to facilitate the continuous processing of instructions applicable to Structure B. Additionally, to prevent data hazards, a comparison of the read and write addresses of source registers is incorporated. The outcome of this comparison determines the validity of read and write operations on source registers for the current instruction. The added control logic increases the depth of combinational logic, resulting in a trade-off with operating frequency. To evaluate clock frequency scalability, both cores were synthesized under identical settings using a standard 0.18-μm CMOS process. The static timing analysis results reveal that the critical paths are located between the control unit and the decode unit. The maximum frequencies of the proposed processor and the SERV are 13.827 and 13.728 MHz, respectively, with the difference being within an acceptable range. However, these results also indicate that the maximum operating frequency of the SDPF processor is constrained by the control logic.

3.2 | Data Hazard

Incorporating the pseudo-pipeline structure into the serial data path introduces data hazards commonly found in traditional pipelines. It is imperative to discuss the data hazards in the pseudo two-stage pipeline and their resolution strategies. Herein, instructions processed based on Structure B are referred to as B-type instructions.

3.2.1 | Read-After-Write

Read-after-write (RAW) is a common data hazard in traditional pipeline structures. This hazard occurs when a subsequent instruction reads from a register before a preceding instruction has completed its write operation to the same register, potentially leading to the reading of outdated data. The introduction of a pseudo two-stage pipeline structure can result in simultaneous read and write operations on the same register by consecutive instructions. As illustrated in Figure 4, taking the LW and ADD instructions as examples, during the second stage of the LW instruction, data needs to be written back to the t5 register. Concurrently, during the first stage of the ADD instruction, the value of the t5 register will be read. But due to the delay in updating the register data, the value read is incorrect.

To mitigate such data hazards, the proposed solution involves two key steps: firstly, retaining the WB operation of the preceding instruction and secondly, canceling the read operation of the subsequent instruction to prevent it from reading an incorrect value. Since the value produced by the execution of the previous instruction is exactly what the subsequent instruction requires, this value can be utilized as the data for the subsequent instruction's calculation.

3.2.2 | Write-After-Write (WAW) and Read-and-WAW (RWAW)

The WAW data hazard occurs when two adjacent instructions attempt to write to the same register simultaneously, potentially resulting in the register being updated to an incorrect value. To address this issue, it is theoretically assumed that the value of the register should reflect the write data of the subsequent instruction. Therefore, when adjacent instructions write to a register concurrently, the write operation of the preceding instruction is effectively canceled.

RWAW arise when the WAW and RAW occur concurrently. Addressing this data hazard requires a combined approach that integrates the solutions for both RAW and WAW hazards.

3.2.3 | Data Conflict in SR

Processing consecutive B-type instructions may introduce potential structural hazards. As illustrated in Figure 5A, when handling a B-type instruction, the SR in the Pre-EX unit is utilized for data preprocessing during the first stage

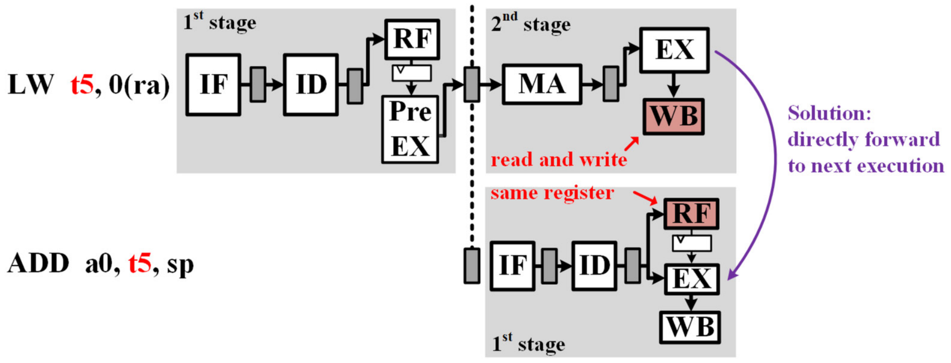


FIGURE 4 | An example of the RAW data hazard.

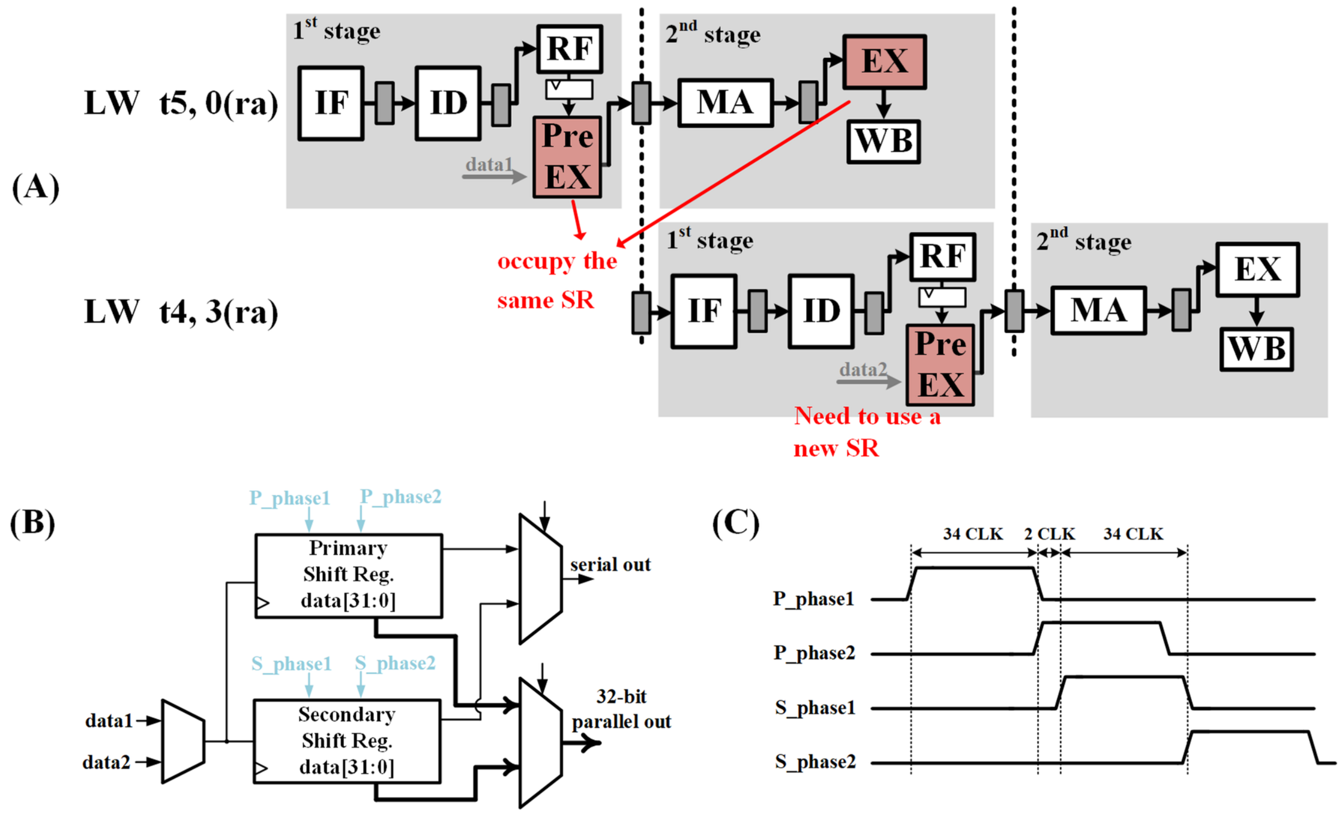


FIGURE 5 | (A) An example of data conflict in SR, (B) schematic of the dual-SR configuration, and (C) timing diagram of the control signal for the dual-SR.

of the pseudo-pipeline. Subsequently, in the second stage of the pseudo-pipeline, the relevant execution operations are performed based on this SR. This implies that the SR will remain occupied until the completion of the current instruction, rendering it unavailable for subsequent B-Type instructions.

A simple and effective solution is to add an additional SR to the two units responsible for processing B-type instructions: the Pre-EX Unit and the Mem. data Unit. Moreover, to avoid unnecessary power consumption, a set of control logic is required to manage the two SRs efficiently. Figure 5B illustrates the schematic of the dual SR configuration. In this figure, Data 1 and Data 2 represent the data to be processed by two consecutive instructions. These data inputs are selected via a data selector to be shifted into either the primary SR or the secondary SR. The main operational logic of the dual SR system is as follows: When only one B-type instruction is present among adjacent instructions, only the primary SR is active, while the secondary SR enters a sleep state. When both adjacent instructions are B-type, the two SRs operate alternately. Figure 5C depicts the timing diagram of the control signals when the two SRs operate alternately. P_phase1 and S_phase1 are the SR update signals; when asserted, they indicate permission to load new data into the SRs. P_phase2 and S_phase2 are the SR active signals; when asserted, they indicate that the SRs are allowed to perform operations associated with the instructions. When processing consecutive B-type instructions, P_phase1 is asserted after the instruction is decoded, allowing the output of the data selector to be sequentially loaded into the primary SR. Subsequently, in the second stage of this instruction, P_phase1 is pulled low and P_phase2 is asserted, enabling the primary SR to process the data. Simultaneously, the next instruction is decoded and S_phase1 is asserted, allowing the output of the data selector to be sequentially loaded into the secondary SR. In the second stage, S_phase1 is invalid and S_phase2 is asserted, enabling the secondary SR to process the data. If the subsequent instruction is also a B-type instruction, P_phase1 is asserted again, initiating a new cycle of execution for the primary SR, and the aforementioned operations are repeated. Thus, by alternately asserting the four control signals, the dual SRs can operate in an interleaved manner. When only one B-type instruction is present among adjacent instructions, the primary SR is controlled solely by P_phase1 and P_phase2, while the control signals for the secondary SR, S_phase1 and S_phase2 remain invalid.

4 | Implementation Results and Discussion

4.1 | Experimental Setup

In this section, the performance, power consumption, and area of the proposed SDPF RISC-V processor are tested and evaluated. The processor's RTL code is synthesized based on both FPGA and a standard CMOS process. The performance metrics of the proposed processor are obtained by running the Dhrystone and CoreMark benchmarks [25, 26]. Dhrystone is a benchmark used to measure the performance of a processor's integer and logical operations, while CoreMark is a similar one.

Dhrystone and CoreMark are two benchmark programs whose scores are positively correlated with performance.

To obtain meaningful experimental results, the proposed SDPF RISC-V processor is tested within a minimalist system-on-chip (SoC), whose block diagram is shown in Figure 6. This SoC includes a 2-kB IMEM and a 4-kB DMEM, which are sufficient to meet the requirements of the target test applications. Multiple communication interfaces, including the universal asynchronous receiver/transmitter (UART) and the serial peripheral interface (SPI), are utilized to transfer results to the host. The internal interconnect of the SoC utilizes the internal chip bus (ICB) of the Hummingbird E203 processor [27], as indicated in the blue-highlighted region. This custom bus is suitable for low-power processor cores and features only two independent handshake channels, allowing users to customize priorities to achieve multi-master and multi-slave peripheral installations.

SERV, an extensively tested open-source RISC-V processor with a serial data path [23], is used as the primary benchmark for comparison. PicoRV32 [19] and ORCA [28] processors are traditional low-power open-source RISC-V processors that based on 32-bit parallel data transmission. The former employs a non-pipelined multi-cycle design, while the latter is based on a four-stage pipelined architecture. These processors, along with the proposed processor, are synthesized and tested within the same SoC environment at a preset frequency of 10 MHz.

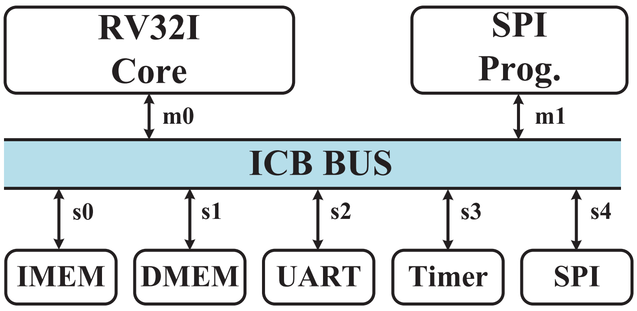


FIGURE 6 | Block diagram of the minimalist SoC implementation for processor measurement.

4.2 | FPGA Implementation Results

This section details the implementation of the processor designed on FPGA, with a focus on the Dhrystone and Coremark benchmarking of the proposed processor and SERV. The test results were obtained using Vivado 2021.2 and a test environment based on the Xilinx KC705 device. Table 1 presents the FPGA resource consumption of each SoC in terms of look up tables (LUTs) and flip-flops (FFs). Compared with the traditional PDPF RISC-V processor, the proposed processor requires fewer logic resources and exhibits lower power consumption. When compared with PicoRV32, which also supports the RV32I, there are reductions of 31.4% in LUTs and 22.3% in FFs.

In comparison with the similarly SDPF-based SERV processor, the proposed processor consumes more LUTs and FFs, and this is because the additional SR utilized in its pseudo-pipeline architecture. These SRs are employed for calculating branch addresses and performing data computations in the Pre-EX Unit, necessitating additional control logic and thereby increasing power consumption. However, the proposed processor achieves an average performance improvement of 55.9% compared with SERV. Therefore, considering the enhanced performance, the additional power and area overhead are deemed acceptable.

4.3 | ASIC Implementation Results

The proposed SDPF RISC-V processor were synthesized and placed-and-routed using a standard 0.18-μm CMOS process with 1.8 V standard cell library. Figure 7 shows the layout of the minimalist SoC. Additionally, for a more effective comparison, SoCs based on SERV, PicoRV32, and ORCA were also synthesized and simulated under the same conditions. Table 2 presents the comparison of post-layout simulation results with other research outcomes. The dynamic power consumption is derived from running the Dhrystone benchmark program. The post-layout simulation results closely align with FPGA prototype validations. Compared with the processors utilizing a parallel 32-bit data path, the proposed processor sacrifices some performance in exchange for lower power consumption and a smaller area

TABLE 1 | Experimental results of RISC-V minimalist SoC implementation on FPGA.

	This work	Integration 2023 [17]	ISCAS 2023 [18]	PicoRV32 [19]	ORCA [28]	SERV [23]
Platform	**KC705**	NEXYS	xc7a100t	KC705	KC705	KC705
ISA	**RV32I**	RV32I	RV32I	RV32I	RV32I	RV32I
Datapath type	**SDPF**	PDPF	PDPF	PDPF	PDPF	SDPF
Frequency (MHz)	**10**	40	N/A	10	10	10
Number of LUTs	**903**	1710	1100	1318	1478	720
Number of FFs	**795**	1024	600	1024	962	631
Dynamic power (mW)	**114**	145.58	N/A	116	116	113
Dhrystone (DMIPS/MHz)	**0.053**	1 MIPS/MHz	0.317	0.355	1.589	0.034
Coremark (CoreMark/MHz)	**0.037**	N/A	N/A	0.263	1.255	0.023

footprint. Compared with [3], the proposed processor achieves a 37.2% reduction in power consumption. Typically, for on-chip environmental monitoring devices powered by energy harvesting systems, lower power consumption means extended operational durations, while a smaller chip area means lower cost.

In the comparison between the proposed processor and SERV, they exhibit similar levels of area and power consumption. The ratio of Dhrystone to dynamic power is utilized as a parameter to reflect the energy efficiency of the two SDPF processors. Additionally, the ratio of Dhrystone to both dynamic power and area is employed as a figure of merit for further evaluation. The proposed processor demonstrates a 40.9% higher performance-to-power ratio compared with SERV and a 31.8% improvement in the performance per power and area. This indicates that the pseudo two-stage pipeline architecture of the proposed processor yields superior performance in the SDPF processor approach and achieves a more optimal trade-off among performance, area and power consumption.

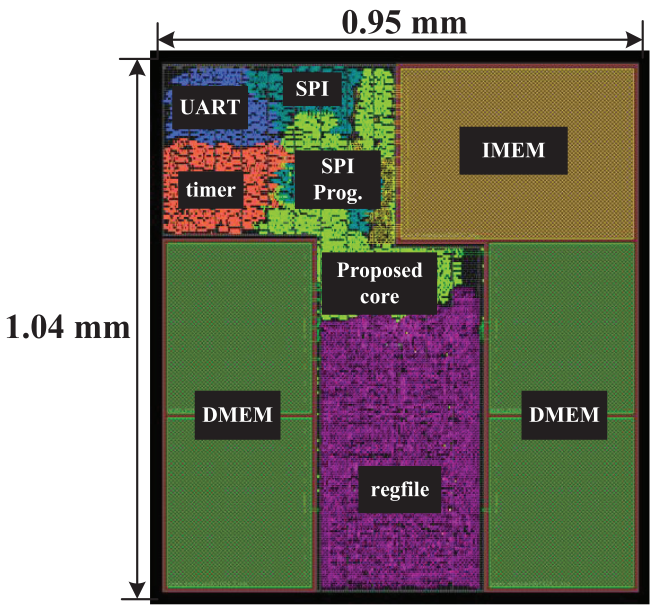


FIGURE 7 | The layout of the minimalist SoC including the proposed processor.

Compared with PDPF processors, SDPF processors, including the proposed architecture, exhibit certain disadvantages in absolute performance. A higher CPI results in lower benchmark performance, leading to reduced energy efficiency. This indicates that SDPF processors are not suitable for scenarios with stringent performance and energy efficiency requirements. However, as a trade-off, the proposed processor demonstrates significant advantages in terms of absolute power consumption and area. In certain battery-less IoT applications, achieving lower absolute power consumption to sustain operation is a more critical challenge than pursuing higher absolute performance.

Although the PDPF processors demonstrates higher energy efficiency compared with the SDPF processors, its larger area and higher power consumption make it difficult to meet the design requirements of self-powered IoT SoC sensors, such as SiF sensors. In comparison with SERV, the proposed processor exhibits a superior energy efficiency area ratio. These characteristics highlight that the proposed processor contributes to the development of more compact, intelligent, and adaptable self-powered IoT SoC sensor nodes.

Furthermore, to obtain more realistic performance and power consumption levels of the SDPF processors, we utilized an actual IoT application program as a workload to test both proposed processor and SERV processor. This workload is a temperature measurement program applied to a fluorescence optical fiber temperature sensor, which determines external temperature by

TABLE 2 | Performance summary and comparison with other works.

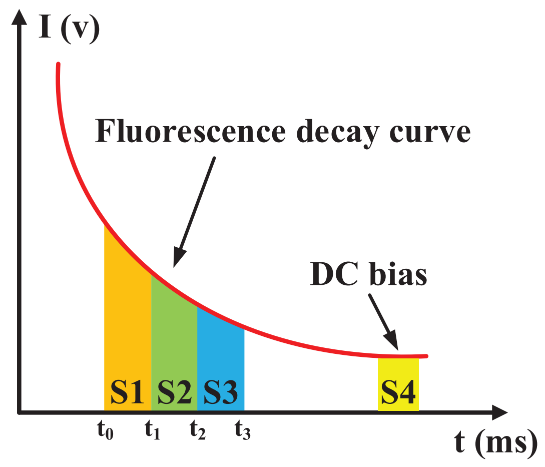
	This work	Integration 2023 [17]	PicoRV32 [19]	ECTM 2023 [20]	ORCA [28]	SERV [23]
Technology (nm)	180	180	180	180	180	180
ISA	RV32I	RV32I	RV32I	RV32I	RV32I	RV32I
Datapath type	SDPF	PDPF	PDPF	PDPF	PDPF	SDPF
Supply voltage (V)	1.8	1.8	1.8	1.8	1.8	1.8
Frequency (MHz)	10	40	10	100	10	10
Average CPI	36	1	4	1	1	>37
Area (mm^2)	0.988	25	1.093	1.33	1.136	0.926
Dynamic power (μW/MHz)	182.1	290	291.3	675	341.9	164.6
Dhrystone (DMIPS/MHz)	0.053	1 MIPS/MHz	0.355	N/A	1.589	0.034
Dhrystone/power (DMIPS/μW)	2.9×10^{-4}	3.4×10^{-3}	1.2×10^{-3}	N/A	4.6×10^{-3}	2.06×10^{-4}
Dhrystone/(Power × Area) (DMIPS/[μW × mm^2])	2.94×10^{-4}	1.38×10^{-4}	1.1×10^{-3}	N/A	4.1×10^{-3}	2.23×10^{-4}

extracting the fluorescence decay constant. Figure 8A shows the fluorescence decay curve after photoelectric conversion, with its mathematical model represented by the following equation:

$$I(t) = I_0 e^{-\frac{t}{\tau}} + B + rn \tag{3}$$

where B represents the DC bias, rn represents the random noise, and τ represents the decay time constant to be determined. The core algorithm of the temperature measurement program is based on the integral ratio algorithm [29], as illustrated in Figure 8B. The post-layout simulation results, shown in Figure 8C, indicate that compared with SERV, the proposed processor exhibits 21.93% higher power consumption but achieves a 58.8% reduction in execution cycles. Therefore, it can be concluded that the proposed processor maintains its advantages in practical IoT scenarios.

Besides, this work presents opportunities for future innovations and improvements to overcome existing limitations. In the current architecture, there are two potential methods to further reduce power consumption. One approach is to modify the instruction set from RV32I to RV32E. The main distinction between RV32E and RV32I is that RV32E supports only 16 source registers, which can lead to reductions in area and power consumption overhead. Furthermore, add the support for the RV32C instruction set.



(A)

$$I(t) = I_0 e^{-\frac{t}{\tau}} + B + rn$$

$$S_1 = \int_{t_0}^{t_1} I(t)dt, S_2 = \int_{t_1}^{t_2} I(t)dt, S_3 = \int_{t_2}^{t_3} I(t)dt$$

$$\Delta S_1 = S_1 - S_2, \Delta S_2 = S_2 - S_3$$

$$\tau = \frac{\Delta t}{\ln(\Delta S_1 / \Delta S_2)}$$

(B)

	Average Power	Cycles Consumption
This work	225.7 μW/MHz	1.581 M
SERV	185.1 μW/MHz	2.512 M

(C)

FIGURE 8 | (A) Fluorescence decay curve, (B) principle of integral ratio algorithm, and (C) the simulation result.

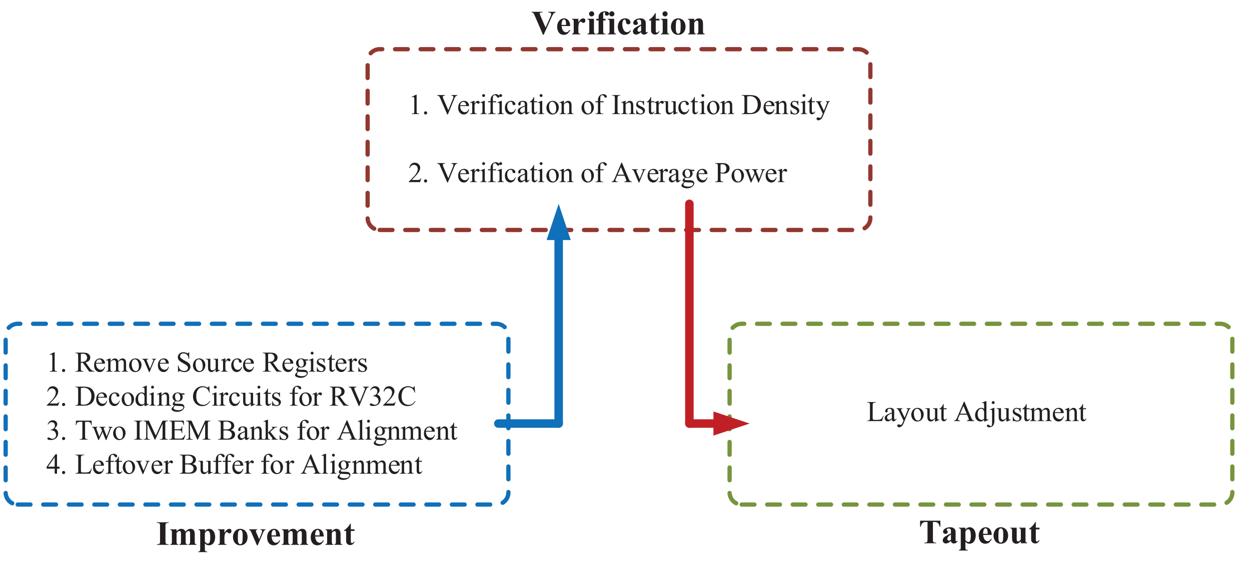

Verification
1. Verification of Instruction Density
2. Verification of Average Power
1. Remove Source Registers
2. Decoding Circuits for RV32C
3. Two IMEM Banks for Alignment
4. Leftover Buffer for Alignment
Improvement
Layout Adjustment
Tapeout

FIGURE 9 | The roadmap for future enhancements using the RV32E/RV32C instruction sets.

RV32C allows for the compression of 32-bit instructions into 16-bit or shorter formats, thereby further decreasing the power consumption during the fetching and decoding processes. Figure 9 presents the roadmap for future enhancements using the RV32E/RV32C instruction sets. Initially, to transition from RV32I to RV32E, 16 source registers will be removed. Subsequently, to accommodate the RV32C instruction set, corresponding decoding circuits must be added. Additionally, to address alignment issues arising from the introduction of the compressed instruction set, the existing IMEM can be split into two interleaved banks, each with a 32-bit width. And a leftover buffer can be incorporated for instruction concatenation and temporary storage. Upon completing these enhancements, it is essential to verify improvements in instruction density and reductions in average power consumption compared with the current architecture. Finally, layout adjustments and tape-out testing will be required. Another approach is to add a switch signal to toggle between a pseudo-pipeline and a non-pipeline mode. Given that there are only two instructions being handled simultaneously in the core, it is feasible to switch between the pipelined and non-pipelined modes by disabling the early IF signal in the core with minimal overhead. Under the low-load conditions, switching to the non-pipelined mode can significantly reduce dynamic power consumption. Although the serial data path architecture exhibits lower power overhead, its energy efficiency is slightly inferior to that of the parallel datapath. Therefore, future work could explore the relationship between the processor data path width, energy efficiency and power consumption while using the 32-bit instruction set.

5 | Conclusion

This paper introduces a pseudo two-stage pipeline SDPF processor, which enhances the performance of SDPF RISC-V processors by reducing CPI in the serial data path through simulated pipeline structures, which achieves improved performance while maintaining low power and area overheads. Implementation results demonstrate a 55.9% performance increase and a 40.9% enhancement in performance-to-power ratio compared with SERV. Additionally, compared with traditional PDPF processors, the proposed processor exhibits lower power consumption and smaller chip area. This work is particularly suitable for energy-constrained IoT nodes requiring low power, compact size and low cost.

Data Availability Statement

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

References

1. G. Loke, T. Khudiyev, B. Wang, et al., "Digital Electronics in Fibers Enable Fabric-Based Machine-Learning Inference," *Nature Communications* 12, no. 1 (2021): 3317–3325.

2. M. Chen, Z. Wang, Q. Zhang, et al., "Self-Powered Multifunctional Sensing Based On Super-Elastic Fibers by Soluble-Core Thermal Drawing," *Nature Communications* 12, no. 1 (2021): 1416–1425.

3. D. Markovic, C. C. Wang, L. P. Alarcon, T. Liu, and J. M. Rabaey, "Ultralow-Power Design in Near-Threshold Region," *Proceedings of the IEEE* 98, no. 2 (2010): 237–252.

4. R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits," *Proceedings of the IEEE* 98, no. 2 (2010): 253–266.

5. M. Gautschi, P. D. Schiavone, A. Traber, I. Loi, A. Pullini, and D. Rossi, "Near–Threshold RISC–V Core With DSP Extensions for Scalable IoT Endpoint Devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25, no. 10 (2017): 2700–2713.

6. X. Liu, D. S. Truesdell, O. Faruqe, et al., "A Self-Powered SoC with Distributed Cooperative Energy Harvesting and Multi-Chip Power Management for System-in-Fiber. 2023 IEEE International Solid–State on Circuits Conference (ISSCC), (2023): 236–238.

7. W. Lim, I. Lee, D. Sylvester, and D. Blaauw, "Batteryless Sub-nW Cortex-M0+ Processor with Dynamic Leakage-Suppression Logic," 2015 IEEE International Solid–State on Circuits Conference (ISSCC), (2015): 1–3.

8. L. Lin, S. Jain, and M. Alioto, "A 595pW 14pJ/Cycle Microcontroller With Dual-Mode Standard Cells and Self-Startup for Battery-Indifferent Distributed Sensing," 2018 IEEE International Solid–State on Circuits Conference (ISSCC), (2018): 44–46.

9. D. S. Truesdell, J. Breiholz, S. Kamineni, N. Liu, A. Magyar, and B. H. Calhoun, "A 6–140-nW 11 Hz–8.2-kHz DVFS RISC-V Microprocessor Using Scalable Dynamic Leakage-Suppression Logic," *IEEE Solid-State Circuits Letters* 2, no. 8 (2019): 57–60.

10. N. Shavit, I. Stanger, R. Taco, L. Yavits, and A. Fish, "Low Power, Energy Efficient and High Performance Triple Mode Logic for IoT Applications," *2024 19th Conference on Ph. D Research in Microelectronics and Electronics (PRIME)*, (2024): 1–4.

11. D. S. Truesdell, X. Liu, J. Breiholz, S. Gupta, S. Li, and B. H. Calhoun, "NanoWattch: A Self-Powered 3-nW RISC-V SoC Operable from 160 mV Photovoltaic Input With Integrated Temperature Sensing and Adaptive Performance Scaling," 2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits), (2022): 210–211.

12. B. C. Wu, W. T. Chen, and T. T. Liu, "An Error-Resilient RISC-V Microprocessor With a Fully Integrated DC–DC Voltage Regulator for Near-Threshold Operation in 28-Nm CMOS," *IEEE Journal of Solid-State Circuits* 58, no. 11 (2023): 3275–3285.

13. J. Lee, Y. Zhang, Q. Dong, et al., "A Self-Tuning IoT Processor Using Leakage-Ratio Measurement for Energy-Optimal Operation," *IEEE Journal of Solid-State Circuits* 55, no. 1 (2019): 87–97.

14. J. K. Brown, D. Abdallah, J. Boley, et al., "A 65 nm Energy-Harvesting ULP SoC with 256kB Cortex-M0 Enabling an 89.1 μW Continuous Machine Health Monitoring Wireless Self-Powered System," *2020 IEEE International Solid–State on Circuits Conference (ISSCC),* (2020): 420–422.

15. C. J. Lukas, F. B. Yahya, K. K. Huang, et al., "A 2.19 μW Self-Powered SoC With Integrated Multimodal Energy Harvesting, Dual-Channel up to −92 dBm WRX and Energy-Aware Subsystem," 2023 IEEE International Solid–State on Circuits Conference (ISSCC), (2023): 238–240.

16. X. Liu, S. Kamineni, J. Breiholz, B. H. Calhoun, and S. Li, "A Sub-μW Energy-Performance-Aware IoT SoC With a Triple-Mode Power Management Unit for System Performance Scaling, Fast DVFS, and Energy Minimization," *IEEE Journal of Solid-State Circuits* 59, no. 7 (2024): 2272–2284.

17. S. Shukla, P. K. Jha, and K. C. Ray, "An Energy-Efficient Single-Cycle RV32I Microprocessor for Edge Computing Applications," *Integration* 88 (2023): 233–240.

18. J. Saussereau, C. Leroux, J. B. Begueret, and C. Jego, "AsteRISC: A Size-Optimized RISC-V Core for Design Space Exploration," *2023 IEEE International Symposium on Circuits and Systems (ISCAS),* (2023): 1–5.

19. C. Wolf, "Picorv32 – A Size Optimized RISC–V CPU [online]." https://github.com/cliffordwolf/picorv32.

20. F. F. Nascimento, R. N. Wuerdig, A. F. Ponchet, et al., "RISC-V SoC Physical Implementation in 180 nm CMOS with a Quark Core Based on FemtoRV32," 2023 IEEE Seventh Ecuador Technical Chapter Meeting (ECTM), (2023): 1–6.

21. T. K. Dang, K. D. Nguyen, C. K. Pham, and T. T. Hoang, "Accumulator-Based 16-Bit Processor for Wireless Sensor Nodes," *IEEE Transactions on Circuits and Systems II: Express Briefs* 71, no. 7 (2024): 3543–3547.

22. M. Sarmiento, K. D. Nguyen, C. Duran, et al., "A Sub-μW Reversed-Body-Bias 8-Bit Processor on 65-Nm Silicon-On-Thin-Box (SOTB) for IoT Applications," *IEEE Transactions on Circuits and Systems II: Express Briefs* 68, no. 9 (2021): 3182–3186.

23. M. Sarmiento, K. D. Nguyen, C. Duran, et al., "System on a Chip With 8 and 32 Bits Processors in 0.18–μm Technology for IoT Applications," *IEEE Transactions on Circuits and Systems II: Express Briefs* 69, no. 5 (2022): 2438–2442.

24. A. Waterman, Y. Lee, R. Avizienis, D. A. Patterson, and K. AsanoviĆ, "The RISC-V Instruction Set Manual Volume II: Privileged Architecture Version 1.9," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS–2015–49, (2015).

25. R. P. Weicker, "Dhrystone: A Synthetic Systems Programming Benchmark," *Communications of the ACM* 27, no. 10 (1984): 1013–1030.

26. S. Gal–On and M. Levy, "Exploring Coremark a Benchmark Maximizing Simplicity and Efficacy," The Embedded Microprocessor Benchmark Consortium, (2012).

27. Open-source Hummingbirdv2 E203 RISC–V Processor Core and SoC [online], https://github.com/riscv-mcu/e203_hbirdv2.

28. VectorBlox, "ORCA: RISC–V by VectorBlox [online]," https://github.com/VectorBlox/orca.

29. D. Jia, L. Meng, S. Lin, and J. Yu, "The Signal Processing Based on Average and Curve Fit Technique for Optic-Fiber Fluorescent Temperature Measurement System in Hyperthermia Clinical," 2011 International Conference on Electrical and Control Engineering, 5110–5113.