# A RISC-V ISA Compatible Processor IP for SoC

Suseela Budi, Pradeep Gupta, Kuruvilla Varghese, Amrutur Bharadwaj
Indian Institute of Science (IISc), Bangalore
Email: {suseela, pradeep, kuru, amrutur}@iisc.ac.in

***Abstract*—The emergence of System-on-Chip technology has brought in opportunities in the form of reduced cycle time, superior performance and time-to-market considerations. Our work focusses on a new processor for a System on Chip. The system has a 32-bit, 5-stage pipelined processor, memory subsystem with virtual memory support, interrupt controller, memory error control module, and UART. The processor is based on RISC-V ISA. It supports Integer, Multiply, and Atomic instructions. Memory subsystem includes split caches and translation lookaside buffers. Interrupt controller supports four levels of preemptive priority and preemption can be programmed for individual interrupts. Memory error control module provides single error correction and double error detection for main memory. Wishbone B.3 bus standard is adopted as on-chip bus protocol. The design is implemented on Virtex-7 (XC7VX485tffg1761-2) board and achieves a peak clock frequency of 100MHz.**

## I. Introduction

Processor design starts by choosing an Instruction Set Architecture (ISA). There are two options, commercial and open source ISAs. Most of the commercial ISAs are closed source and proprietary. Its licensing is expensive. Hence, it was decided to build a processor using an open source ISA, compliant with current industry standards. The advantage of this methodology would be to have full control over every aspect of the design, potentially allowing for maximal efficiency. Table 1 shows comparison of available open source ISAs [1]. Among them, RISC-V ISA is chosen for the current design.

| | | | | Address | | | Software | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| ISA | Base+Ext | Compact code | Quad FP | 32-bit | 64-bit | 128-bit | GCC | LLVM | Linux | QEMU |
| SPARC V8 | | | ✓ | ✓ | | | ✓ | ✓ | ✓ | ✓ |
| OpenRISC | | | | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ |
| RISC-V | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

TABLE I: Comparison of open source ISAs

RISC-V is an emerging open source ISA. Release of "Rocket Chip" [2] triggered development of processors based on RISC-V. A soft core based on chisel [3] called "ZSCALE" [2] is released based on RISC-V RV32IM extension. The verilog version of "ZSCALE" is released as "VSCALE" [4]. A microcontroller based on RISC-V ISA named "mRISC-V" is described in [5]. A system with the combination of cache, translation lookaside buffer (TLB), virtual memory, support for atomic instructions, error handling, and configurable nested interrupt handling capability is not available till date. Hence, it was decided to build a system having all the features mentioned above.

RISC-V supports four privilege levels [6]. They are machine, hypervisor, supervisor and user levels. Machine level has the highest privileges and mandatory for a RISC-V hardware platform. RISC-V hardware thread is running at some privilege level encoded as a mode in one or more Control and Status Registers (CSRs). Machine level CSRs are implemented in the design to support machine mode.

Programs exhibit temporal and spatial localities [7]. Cache memories exploit this property of the programs to give improved performance.

Virtual memory is a memory management technique that maps memory addresses used by a program, called virtual addresses, into physical addresses in computer memory. Address translation hardware in the CPU, often referred to as a memory management unit (MMU), automatically translates virtual addresses to physical addresses using page table. Accessing page table is time consuming, as it resides in main memory. TLB is a memory cache that is used to reduce the time taken to access page table.

Our design is a standard RISC-V page-based 32-bit virtual-memory system (Sv32 [6]). Virtual address is divided into a Virtual Page Number (VPN) and page offset, as shown in Figure 1 a. The 20-bit VPN is translated into a 22-bit Physical Page Number (PPN) by using a two-level page table. The 12-bit page offset is untranslated. PPN and page offset together forms physical address.

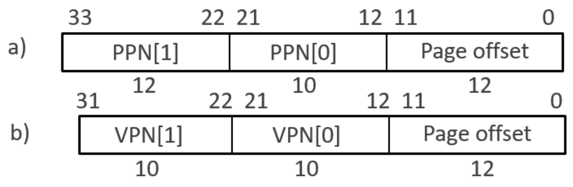

Fig. 1: a) Virtual Address b) Physical Address

System-on-Chip (SoC) design requires a standard bus interface for IP cores to communicate with each other. There are many bus protocols, but AMBA [8], CoreConnect and WISHBONE [9] are well known and commonly used SoC bus architectures. WISHBONE supports Single and Block Read/Write cycles, Read-Modify-Write (RMW) transfer. Hence, WISHBONE is chosen as the bus architecture for our design. Wishbone B.3 is the bus standard.

The rest of the paper is organized as follows. Section II explains design of all the blocks in the system. Section III discusses various cases for verification. Results of FPGA implementation are given in Section IV, followed by future scope in Section V and conclusion in Section VI.

Authorized licensed use limited to: LIVERPOOL JOHN MOORES UNIVERSITY. Downloaded on April 21,2023 at 10:25:32 UTC from IEEE Xplore. Restrictions apply.

## II. Processor System Design

The system architecture is shown in Figure 2.

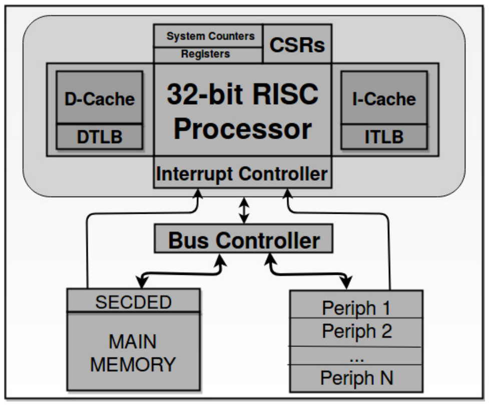

Fig. 2: Processor Architecture

The system has a 32-bit processor based on RISC-V ISA. It supports Integer (I), Multiply/Divide (M), and Atomic (A) instructions. It includes 32, 32-bit general purpose registers and machine level Control and Status Registers (CSR). It also supports three system counters, which are *CYCLE*, *TIME*, and *INSTRET*. *CYCLE* counts the number of clock cycles executed by the processor from an arbitrary start time in the past. *TIME* tracks integer value corresponding to the wall-clock real time that has passed from an arbitrary start time in the past. *INSTRET* count the number of instructions retired by this hardware thread from some arbitrary start point in the past.

To avoid stalls due to structural hazards, instruction and data caches (I-Cache and D-Cache) are provided separately. TLB is implemented to speed up the address translation. As the system has Instruction and data caches, Instruction TLB (ITLB) and Data TLB (DTLB) are provided respectively. I-Cache and ITLB forms instruction memory subsystem. Similarly, D-Cache and DTLB forms data memory subsystem. Interrupt controller with four levels of pre-emptive priority is implemented for processing internal and external interrupts. The pre-emptive priority is programmable. The system follows wishbone protocol. Bus arbiter can be programmed as priority based or round robin. Instruction memory subsystem and data memory subsystem are two masters. Main memory and UART are slaves. Memory data corruption is often fatal to the operation of a system. To overcome this problem, Single Error Correction and Double Error Detection (SECDED) is provided to main memory as error control mechanism.

### *A. Processor*

The processor is a 32-bit, 5-stage pipeline architecture with support for RV32IMA [10] instructions. The pipeline stages are Fetch (F), Decode (D), Execute (E), Memory (M), and Write-back (W) as shown in Figure 3. Address of the instruction to be executed resides in Program Counter (PC). In the fetch stage, PC value is generated, either by an internal counter or toggled by external select lines like 'branch' and 'IRQ'. PC is given to instruction memory subsystem to fetch the instruction. Instruction from I-Cache is given directly to the decode stage. The decode stage also generates select signals for the multiplexers, which do data forwarding in the execute stage. These signals are generated by comparing current source registers and previous destination registers. In case of Atomic Memory Operations (AMO), decode stage is responsible for reading data from memory. Dedicated hardware takes care of DTLB and D-Cache accesses in decode stage.

Extracted opcodes from decode stage are given to execute stage to perform Integer-Multiply-Atomic (IMA) operations. Forwarding lines toggle the multiplexers so that either the forwarded data or the data from register/CSR/memory is used in execute stage. Forwarded data could be either from memory stage or execute stage itself. Fetch and decode stages are stalled in case of multiply/divide instructions, which takes 2, 32 clock cycles respectively. Results are calculated in execute stage based on control signals from decode stage. These results are given to memory stage to perform data transactions and are also used for updating CSRs. Data transactions happens in load/store/atomic instructions. In register operations, memory stage forwards the results to write-back stage. In write-back stage, register file is updated, if needed. Currently processor supports 32, 32-bit general purpose registers. Two registers can

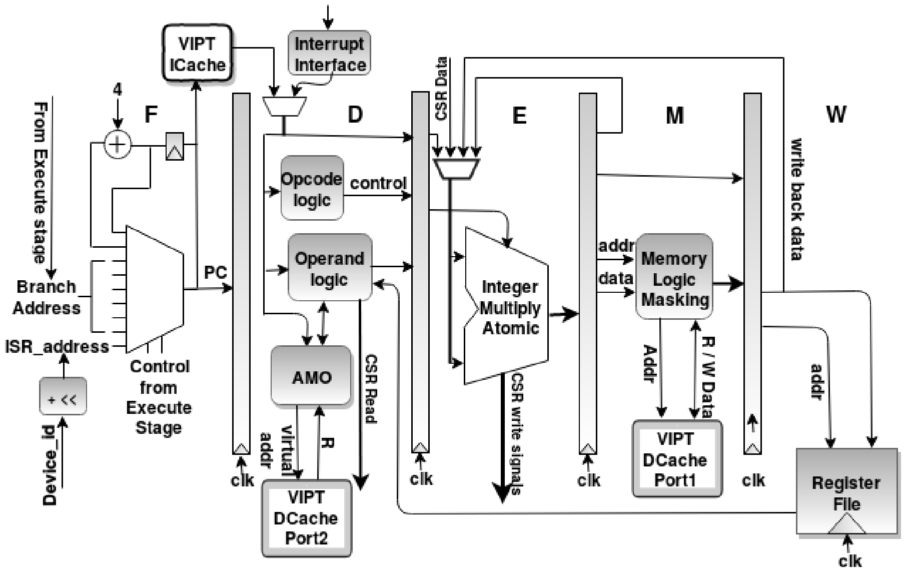

Fig. 3: Pipeline Stages

Authorized licensed use limited to: LIVERPOOL JOHN MOORES UNIVERSITY. Downloaded on April 21,2023 at 10:25:32 UTC from IEEE Xplore. Restrictions apply.

be read out concurrently in the decode stage and one can be written in the write-back stage. According to RISC-V standard register zero is hardwired to zero. For atomic instructions, the memory address is directly taken from the register.

RISC-V ISA sets aside a 12-bit encoding space for up to 4,096 CSRs. It has dedicated instructions for CSR read/write. Machine level CSRs (includes system counters) are incorporated in to the design. Interrupt controller registers are also implemented in CSR address space.

## *B. Data Memory Subsystem*

Data memory subsystem consists of D-Cache and DTLB. Load/store and atomic instructions access memory for data. In load/store instructions, address will be generated in execute stage and used to get data from memory. In atomic instructions, data is read from memory in decode stage. Accessing data memory might happen when reading data in decode stage or reading/writing data in memory stage. Read/write in memory stage is handled by port1 Finite State Machine (FSM). Requests from the decode stage are processed by Port2 FSM. As shown in Figure 4, load/store opcode (‘*lsu_op*’) and memory address (‘*proc_adr*’) signals are given to D-Cache for reading/writing data. Port1 and port2 signals from

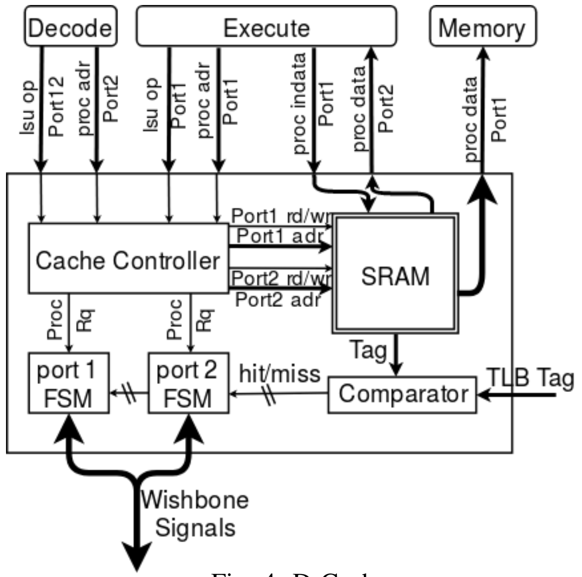

Fig. 4: D-Cache

execute and decode stage respectively are given to cache controller. In case of simultaneous read from decode stage and read/write from memory stage, priority is given for memory stage. Cache controller smoothens the two different accesses to data memory. It stalls request from port2, while processing request from port1. Address and read/write signals are given to SRAM. At the same time processor request (‘*Proc_Rq*’) is given to FSM to inform data access. Tag read from SRAM is compared with the physical tag and hit status will be generated. If it is a hit, data is read/written from/to SRAM. Depending on port request, output data will be either going to execute stage or memory stage. In case of a miss, FSM starts transaction with main memory. Data will be written in SRAM and corresponding tag will be updated. FSM then regenerates the same request and it repeats the process.

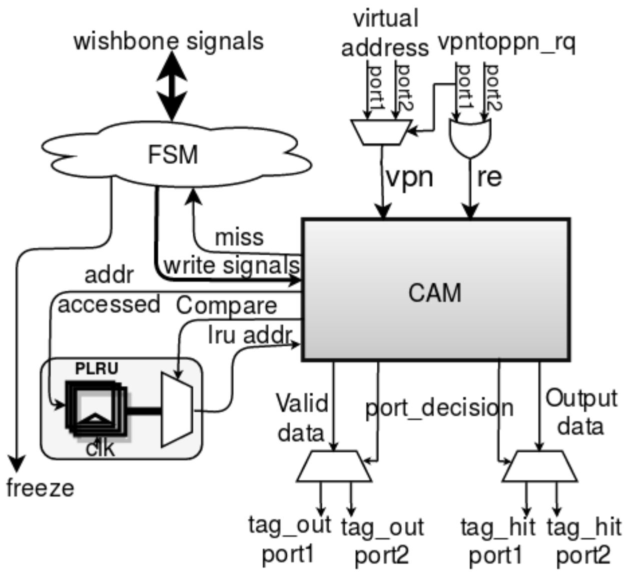

Fig. 5: DTLB

DTLB operation is shown in Figure 5. Virtual address and vpn to ppn request are the inputs to this module. Read enable (‘*re*’) will be generated, if there is a ‘*vpntoppn_rq*’ from any of the ports. VPN will be selected from any of the virtual addresses of the two ports, based on the ‘*vpntoppn_rq*’. VPN and read enable signals are given to CAM module. If the VPN has a PPN in any of the CAM entries, ‘*Valid_data*’ signal will be generated to indicate that ‘*Output_data*’ is valid. ‘*Output_data*’ is demultiplexed as ‘*tag_out_port1*’ or ‘*tag_out_port2*’ depending on ‘*port_decision*’. Similarly, ‘*tag_hit_port1*’ or ‘*tag_hit_port2*’ are generated from ‘*Valid_data*’. If there is no translation for the VPN, then ‘*miss*’ will be generated. FSM will send a request to the main memory for the page table entry. After successful bus transaction, write signals i.e. CAM entry and write enable are given to CAM module. Since CAM has only 32 entries, there might be a need for replacement. The entry to be replaced (‘*lru_addr*’) will be decided by Pseudo Least Recently Used (PLRU) unit. Once CAM replacement is done, them FSM regenerates vpn to ppn request to CAM module. Output data is given out from CAM module with valid signal.

CAM has 32 entries. CAM entry register comprises VPN, PPN and protection bits. When a read enable (‘*re*’) is given to this module, ‘*VPN_in*’ is compared with the VPN of all the CAM entry registers using parallel comparators. If any of the comparisons is successful, PPN and protection bits are given as ‘*Output_data*’ as shown in Figure 6. Otherwise miss will be generated. Upon successful bus transaction, Page Table Entry (PTE) from main memory, write enable and address are used to update the CAM entry.

As shown in Figure 7, virtually-indexed, physically-tagged (VIPT) cache architecture is implemented in the design. Virtual address coming to this block is divided into VPN and page offset with 20 and 12 bits respectively. VPN is given

Authorized licensed use limited to: LIVERPOOL JOHN MOORES UNIVERSITY. Downloaded on April 21,2023 at 10:25:32 UTC from IEEE Xplore. Restrictions apply.

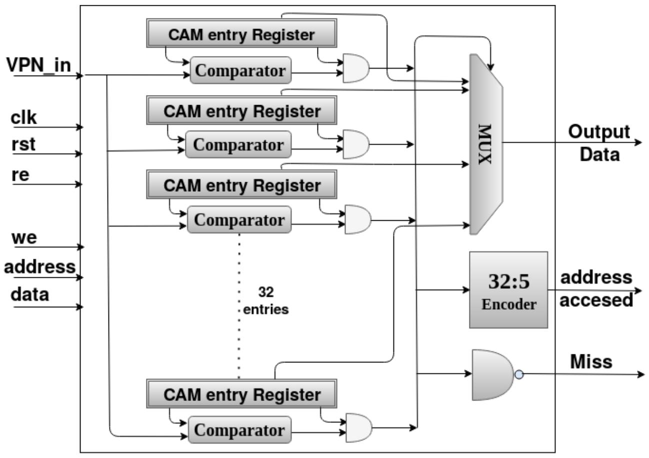

Fig. 6: CAM

to DTLB to get PPN, while the page offset is given to D-Cache to get physical tag. Data will be given out from this module, upon successful matching of PPN and tag. The above process happens in one clock cycle in case of both DTLB and D-Cache hits. In case of DTLB miss, TLB filling unit initiates the bus transaction to get the Page Table Entry (PTE) from the main memory. Meanwhile tag comparison unit waits for the PPN. Dedicated Content Addressable Memory (CAM) module is designed to get the required performance. The CAM has 32-entries, and is fully associative with Pseudo Least Recently Used (PLRU) technique as replacement policy. Once TLB filling is done, tag comparison unit checks for D-Cache hit/miss. If it is a D-Cache miss, bus interface module initiates data transaction between D-Cache and main memory. Once D-Cache is filled, data and hit status will be updated.

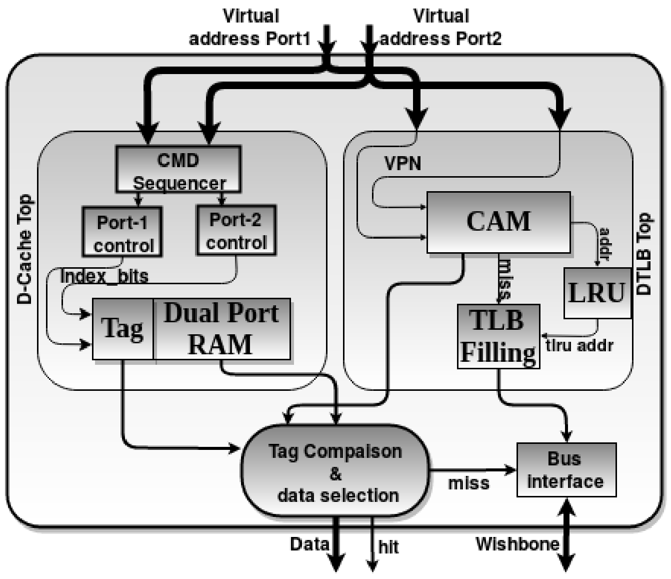

Fig. 7: Data memory subsystem

D-Cache is 8KB, two-way set associative with Least Recently Used (LRU) replacement policy and Cache line is of 32 bytes. In case of AMO, D-Cache is accessed in decode stage as well as in memory stage. Memory stage requires read and write operations, while decode stage requires read operation only. Dual port RAM is used for D-Cache to satisfy the above requirement. Port-1 access is given to memory stage and port-2 access is given to decode stage. Support to handle different port requests is given to DTLB.

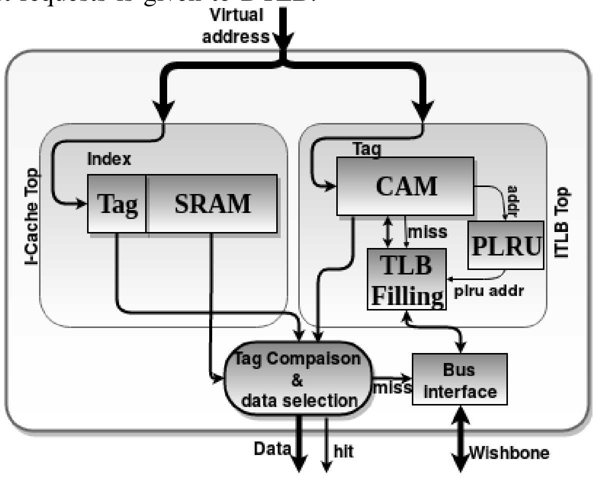

Fig. 8: Instruction memory subsystem

## C. Instruction Memory Subsystem

Instruction memory subsystem consists of I-Cache and ITLB as shown in Figure 8. Instruction memory architecture is same as data memory architecture, except that single port RAM is used instead of dual port RAM. This module is a subset of data memory subsystem. The working is same as explained in Section B. Input for this module i.e. virtual address is generated in fetch stage. Instruction output from this module is supplied directly to decode stage.

## D. Interrupt Controller

The interrupt controller comprises of interrupt edge detector, control logic, interrupt interface, and register bank. Interrupt edge detector receives interrupts from external/internal sources and generates interrupt pulse. Control logic takes the inputs, sorts them according to priority and then updates the Register bank. It sends ‘IRQ’ (Interrupt Request) signal to Interrupt interface, which initiates interrupt processing. Dedicated instructions are directly given to the decode stage by stalling the fetch stage. These instructions preserve the current status of the pipeline. Interrupt interface, then acknowledges control logic through ‘ACK’ signal. At the end of these instructions, fetch stage is released from stall and PC jumps to ISR address. Based on device id, ISR address is calculated. On completion of ISR, interrupt interface takes control of pipeline to restore previous status. It then sends ‘done’ signal to the control logic. The available registers are Interrupt Status Register (ISR), Interrupt Enable Register (IER), Interrupt Active Register (IAR), Interrupt Pending Register (IPR), Interrupt Priority Registers (IPrRx).

## E. Memory Error Control

Main Memory (DRAM) is prone to soft errors. To reduce the failures, SECDED module [11] is implemented. This module is placed in between wishbone interface and main memory. Data (32-bit) to be written into main memory is

Authorized licensed use limited to: LIVERPOOL JOHN MOORES UNIVERSITY. Downloaded on April 21,2023 at 10:25:32 UTC from IEEE Xplore. Restrictions apply.

given for SECDED parity calculation. 39-bit data after adding parity is stored in memory. While reading, 39-bit data from the memory is given to SECDED parity checker. It checks for single/double errors and exceptions are raised accordingly. In case of single/no error, 32-bit data is retrieved from 39-bit data and given to wishbone interface.

### *F. UART Peripheral*

The system has a UART peripheral. UART operates in FIFO mode and supports full duplex communication. Trigger levels can be programmed through FIFO control Register. Baud rate is programmable and can go upto 12.5Mbaud, while operating at 100MHz. Base module is taken from Opencores [12] and modified to support standards like profibus. It supports wishbone protocol. *Rx* and *Tx* are receiver and transmitter signals respectively. Module supports two interrupts i.e. transmitter fifo empty (*tf_empty*) and receiver fifo full (*rf_full*).

## III. Verification

Verification is done at module and integration level. Pipeline is verified for different combinations of instructions and ensured corner cases are covered. Caches are validated against miss-hit combinations and replacement policy. And TLB with Cache is also verified. SECDED is verified by generating intentional errors through a python script. Baud rate and FIFO trigger levels are validated for UART. The whole system is verified in FPGA by applications like viterbi decoder, temperature controller, gauassian random double function etc. FreeRTOS is ported on the system. FreeRTOS software is common for all architectures. The FreeRTOS port files are architecture dependent. Interrupt enabling/disabling has to be handled correctly in the port files. A small mistake in port files will lead to a system crash. RTOS operation is tested in FPGA with multiple tasks. For verification, COE (.coe) file with instructions is loaded into memory rather than test bench. This mimics the real time operation. All the test cases are written in software and COE file is generated using tool chain. This reduces time to write test bench. Test cases from *RISC-V Test* [14] and *Torture* [15] tools are also used.

## IV. Results

The design is implemented using Xilinx Vivado 2016.2 on Virtex-7 FPGA (xc7vx485tffg1761-2) based VC707 board. Post route resource utilization is shown in Table II. This includes number of Look Up Tables (LUTs), Registers, RAM36, and RAM18 for individual modules as well as complete system. RAM36 and RAM18 refers to SRAM of 36 and 18 Kilo bits respectively. Currently the system is running at 100MHz without atomic operation support and 80MHz with atomic instructions. The percentage of utilization for LUTs and Registers are 4.68 and 1.36 respectively.

Pulpino [16] and Shakti C [17] class processors are similar to our processor and based on RISCV ISA. None of them has virtual memory support. These processors are openly available. We had taken the codes and implemented in FPGA. The table III shows the comparison interms of LUTs and Registers.

| Module | LUTs | Registers | RAM36 | RAM18 |
|---|---|---|---|---|
| Top | 14201 | 8237 | 96 | 2 |
| Pipeline | 5508 | 1968 | 0 | 0 |
| Memory Subsystem | 8198 | 6897 | 26 | 2 |
| Others | 2033 | 994 | 70 | 0 |

TABLE II: Resource Utilization in Virtex-7 FPGA

| Processor core | LUTs | Registers |
|---|---|---|
| Our processor | 14201 | 8237 |
| Pulpino | 16078 | 11642 |
| Shakti C class | 18715 | 14032 |

TABLE III: Comparison of Resources

## V. Conclusion

In this paper, we have discussed about a 32-bit processor system for SoC. Memory subsystem with virtual address support falls under critical part for verification. The system is implemented on Virtex-7 FPGA and results are shown in terms of area and frequency. Linux can be booted by adding Supervisor level. By adding hypervisor level, can run virtual machine. The core is stable and extensively verified. A co-processor for the system can be designed by adding Packed-SIMD instructions (RISC-V future standard extension). Application specific hardware can be added to the system, by implementing dedicated instructions (under RISC-V non-standard extension).

## References

[1] K. Asanovic and D. A. Patterson, “Instruction sets should be free: The case for risc-v,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146, Aug 2014.

[2] A. M. Yunsup Lee, Albert Ou, “Z-scale: Tiny 32-bit risc-v systems with updates to the rocket chip generator.” The International House, Berkeley, 2015.

[3] “Chisel,” https://chisel.eecs.berkeley.edu, the Regents of the University of California, 2015.

[4] “Verilog version of z-scale,vscale,” https://github.com/ucb-bar/vscale, 2016.

[5] C. D. et al., “A 32-bit risc-v axi4-lite bus-based microcontroller with 10-bit sar adc,” in *VII Latin American Symposium on Circuits and Systems (LASCAS)*, 2016.

[6] A. Waterman, Y. Lee, R. Avizienis, D. A. Patterson, and K. Asanovi, “The risc-v instruction set manual volume ii: Privileged architecture version 1.9.1,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-161, Nov 2016. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-161.html

[7] A. J. Smith, “Cache memories,” *ACM Computing Surveys (CSUR)*, vol. 14, no. 3, pp. 473–530, 1982.

[8] he AMBA Bus Architecture, “www.arm.com.”

[9] OPENCORES.ORG, *WISHBONE System-On-Chip (SoC) Interconnection Architecture for Portable IP Cores, Revision B.3,*, 2015.

[10] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovic, “The risc-v instruction set manual, volume i: User-level isa, version 2.0,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-54, May 2014. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-54.html

[11] “Error detection and correction,” Supplement to Logic and Computer Design Fundamentals, pearson Education, 2004.

[12] OpenCores, “Uart16550,” https://opencores.org/project,uart16550.

[13] J. L. Hennessy and D. A. Patterson, *Computer architecture: a quantitative approach*. Elsevier, 2011.

[14] “https://github.com/riscv/riscv-tests.”

[15] “https://github.com/ucb-bar/riscv-torture.”

[16] “https://github.com/pulp-platform/pulpino.”

[17] “rise.cse.iitm.ac.in/shakti.html.”

Authorized licensed use limited to: LIVERPOOL JOHN MOORES UNIVERSITY. Downloaded on April 21,2023 at 10:25:32 UTC from IEEE Xplore. Restrictions apply.