Implementation and Performance Evaluation of Bit Manipulation Extension on CVA6 RISC-V

Muhammad Ijaz
10xEngineers
Lahore, Pakistan
muhammad.ijaz@10xengineers.ai

Fatima Saleem
10xEngineers
Lahore, Pakistan
fatima.saleem@10xengineers.ai

Umer Shahid
University of Engineering and Technology
Lahore, Pakistan
umershahid@uet.edu.pk

Saad Waheed
10xEngineers
Lahore, Pakistan
saad.waheed@10xengineers.ai

Jean-Roch Coulon
Thales Group
Meyreuil, France
jean-roch.coulon@thalesgroup.com

ABSTRACT

An embedded system requires two conflicting attributes, low power and high performance. Embedded controllers and Internet of Things (IoT) applications are seeing a paradigm shift from using x86 or ARMv8 to the widely accepted RISC-V architecture. Bit manipulation Instructions improve the speed of bit manipulation by providing better system performance due to its improved code density, low power consumption, and improved runtime efficiency. RISC-V base Instruction Set Architecture (ISA) doesn't support bit manipulation instructions. However, due to RISC-V architecture's modularized instruction extension support, processor designers can add bit manipulation instructions to support low-power embedded applications. In this paper, we have implemented the bit manipulation extension (B-extension) of RISC-V on OpenHW's application class SoC CVA6. We have synthesized the design for both the Kintex-7 FPGA board using Xilinx Vivado ISE 2018.2 and the TSMC 65 nm cell library. We have performed quantitative analysis on size and power improvement with the help of FPGA and ASIC synthesis data. The performance and code size reduction capacity is presented by showing the performance results of Dhrystone standard benchmarks against the standard CVA6 ISA (IMAFDC or IMAFC) for both RV64 and RV32 configurations. The result shows 4% improvement in dynamic power usage for RV64, 12.5% improvement in code size while building Linux image for RV64IMAFDCB at the cost of 4% increase in LUTs for FPGA implementation and 3% increase in gate count for ASIC implementation. We have seen 18% speed-up and 4% code size reduction for the Dhrystone benchmark.

CCS CONCEPTS

• **Hardware → Application specific instruction set processors**; Power and thermal analysis; • **Computer systems organization → Data flow architectures**; • **General and reference → Performance**; **Design**.

KEYWORDS

CVA6, RISC-V 'B' extension, Dhrystone, Linux Image, ASIC Implementation, FPGA Implementation

ACM Reference Format:
Muhammad Ijaz, Fatima Saleem, Umer Shahid, Saad Waheed, and Jean-Roch Coulon. 2023. Implementation and Performance Evaluation of Bit Manipulation Extension on CVA6 RISC-V. In *20th ACM International Conference on Computing Frontiers (CF '23), May 9–11, 2023, Bologna, Italy.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3587135.3591439

1 INTRODUCTION

RISC-V is an open-source ISA [3] whose base extension supports word-sized, and half word sized operations. The RISC-V Bitmanip extension (short for "Bit Manipulation") [2] is a standard extension of the RISC-V instruction set architecture that adds instructions for bit-level operations such as bit counting, bit field extraction and insertion, and bitwise logical and arithmetic operations. The Bitmanip extension provides a set of instructions for manipulating individual bits or groups of bits within a register. These operations are often used in cryptography, data compression, and other performance-critical applications.

CVA6 [5], also known as Ariane Core, is OpenHW's application class processor. It is one of the most recognized open-source RISC-V application class cores [1] and was developed on PULP open-source platform. It is a six-stage pipeline core that implements RV64IMAFDC architecture with three privilege levels. In this paper, we have implemented bit manipulation instructions support for CVA6 and evaluated its performance by running Linux image and Dhrystone v2.0 benchmark codes [4]. The results are compared against standard implementation for RV64 and RV32.

2 CVA6 BIT MANIPULATION SUPPORT

The microarchitectural changes to CVA6 for supporting hardware Bitmanip instructions is illustrated in Fig 1. To add support for these extensions, we have extended/modified the Decoder, ALU, and Multiplier modules of the CVA6 (highlighted in the figure). The major changes have been done to the Decoder module which includes adding the RISC-V Bitmanip opcodes and selecting the appropriate CVA6 Functional Unit, Multiplier or the ALU, depending upon the type of bit manipulation instruction. So, for instance, the sh1add - shift by 1 add instruction's functionality consists of a shift and

Table 1: ASIC and FPGA Synthesis and benchmark results for CV64A6_IMAFDC_SV39, CV64A6_IMAFDCB_SV39, CV32A6_IMAFC_SV32, and CV32A6_IMAFCB_SV32

Category	Parameters	RV64IMAFDC	RV64IMAFDCB	RV32IMAFC	RV32IMAFCB
Utilization	LUTs	73726 (total) 81 (i_alu) 544 (multiplier) 5874 (scoreboard)	77113 (total) 547 (i_alu) 325 (multiplier) 5927 (scoreboard)	54629 (total) 9 (i_alu) 86 (multiplier) 4224 (scoreboard)	55591 (total) 184 (i_alu) 149 (multiplier) 4657 (scoreboard)
Power	On-Chip Power	2.910 W (total) 2.726 W (dynamic)	2.981 W (total) 2.617 W (dynamic)	2.892 W (total) 2.686 W (dynamic)	2.870 W (total) 2.578 (dynamic)
Code size Reduction	fw_payload.bin	2.4MB	2.1MB	2.1MB	1.9MB
	Image	18.7 MB	17.6 MB	16.8MB	14.2MB
	vmlinux	14.9MB	14.1MB	11.5MB	10.2MB
ASIC Synthesis					
ASIC Synthesis		**RV64IMAFDC**	**RV64IMAFDCB**	**RV32IMAFC**	**RV32IMAFCB**
Total gate count		515 kgates	532 kgates	186 kgates	193 kgates
EX state (multiplier)		21.47 kgates	31 kgates	4.73 kgates	7.158 kgates
EX stage (ALU)		2.836 kgates	9.544 kgates	<2 kgates	4.772 kgates
Dhrystone Benchmark Results for RV64 and RV32					
Options		**RV64IMAFDC**	**RV64IMAFDCB**	**RV32IMAFC**	**RV32IMAFCB**
Time		5.0ms 250kcycles	4.1ms 205kcycles	6.3ms 315kcycles	5.9ms 295kcycles
Code Size		142.8KB	139.9KB	236.9KB	227.9KB

addition and as such is executed in the ALU. On the other hand the clmul - carry-less multiplication instruction performs a carry-less multiplication and so it is executed in the Multiplier unit. This also illustrates that both the ALU and the multiplier units have to be modified to support Bitmanip instructions.

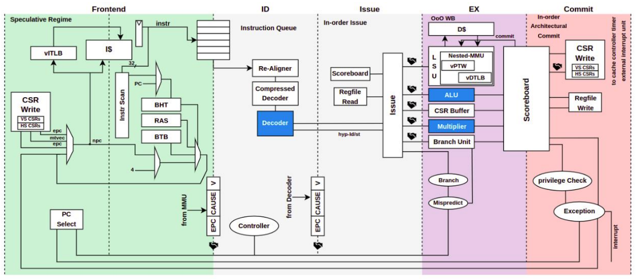

Figure 1: CVA6 core microarchitecture featuring the RISC-V bitmanip extensions

3 RESULTS ANALYSIS

In this section, the ASIC and FPGA implementation results are compared between the standard CVA6 core and the modified core with Bitmanip extensions. Dhrystone benchmark was compiled with RISC-V GCC version 12.2.0 and run on a Linux-booted CVA6 core on a Kintex-7 FPGA board. The FPGA synthesis was done using Xilinx Vivado 2018.1. The ASIC synthesis was done using the TMSC 65 nm PDK and its gate count was compared against the standard implementation. The complete results of the ASIC and FPGA implementations, and Dhrystone benchmark are given in Table 1.

4 CONCLUSION

In this paper we have shown the implementation of the Bitmanip extension support in application class RISC-V processor (CVA6) and evaluated the ASIC 294 and FPGA synthesis data, Linux code size and Dhrystone benchmark results. The data show that a standard CVA6 (RV64IMAFDC or RV32IMAFC) implemented on a FPGA versus a CVA6 with the Bitmanip extensions (RV64IMAFDCB or RV32IMAFCB) results in a 4% improvement in dynamic power for RV64 (5% improvement for RV32) , 12.5% improvement in code-size for RV64 Linux image (9% improvement for RV32) at a cost of 4% increase in LUTs for an FPGA implementation (3% increase in gate count for ASIC implementation.) We have also seen 18% speed-up and 4% code size reduction in the results of the Dhrystone benchmark.

REFERENCES

[1] Alexander Dörflinger, Mark Albers, Benedikt Kleinbeck, Yejun Guan, Harald Michalik, Raphael Klink, Christopher Blochwitz, Anouar Nechi, and Mladen Berekovic. 2021. A comparative survey of open-source application-class RISC-V processor implementations. *Proceedings of the 18th ACM International Conference on Computing Frontiers* (2021). https://doi.org/10.1145/3457388.3458657

[2] Bastian Koppelmann, Peer Adelt, Wolfgang Mueller, and Christoph Scheytt. 2019. RISC-V Extensions for Bit Manipulation Instructions. In *2019 29th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. 41–48. https://doi.org/10.1109/PATMOS.2019.8862170

[3] Andrew Waterman, Yunsup Lee, David A Patterson, and Krste Asanovic. 2011. The risc-v instruction set manual, volume i: Base user-level isa. *EECS Department, UC Berkeley, Tech. Rep. UCB/EECS-2011-62* 116 (2011).

[4] Reinhold P. Weicker. 1984. Dhrystone. *Commun. ACM* 27, 10 (1984), 1013–1030. https://doi.org/10.1145/358274.358283

[5] F. Zaruba and L. Benini. 2019. The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27, 11 (Nov 2019), 2629–2640. https://doi.org/10.1109/TVLSI.2019.2926114

Github Repository Link: https://github.com/openhwgroup/cva6;
Github Pull Request Link: https://github.com/openhwgroup/cva6/pull/878