An FPGA Implementation of A Portable DNA Sequencing Device Based on RISC-V

Zhongpan Wu, *Member, IEEE,* Karim Hammad, *Member, IEEE,* Abel Beyene, *Member, IEEE,* Yunus Dawji, *Member, IEEE,* Ebrahim Ghafar-Zadeh, *Senior Member, IEEE,* and Sebastian Magierowski, *Member, IEEE*

***Abstract*—Miniature and mobile DNA sequencers are steadily growing in popularity as effective tools for genetics research. As basecalling algorithms continue to evolve, basecalling poses a serious challenge for small computing devices despite its increasing accuracy. Although general-purpose computing chips such as CPUs and GPUs can achieve fast results, they are not energy efficient enough for mobile applications. This paper presents an innovative solution, a basecalling hardware architecture based on RISC-V ISA, and after validation with our custom FPGA verification platform, it demonstrates a 1.95x energy efficiency ratio compared to x86. There is also a 38% improvement in energy efficiency ratio compared to ARM. In addition, this study also completes the verification work for subsequent ASIC designs.**

***Index Terms*—DNA, Sequencing, RISC-V, ISA,energy-efficiency, Oxford nanopore, basecalling, Viterbi, FPGA, ASIC.**

I. Introduction

DNA, is the material basis of genes and the most important molecule of life, as a specific carrier of biological inheritance and characteristics. The sequencing technologies used for measuring DNA are now playing an important role in many fields, such as medicine and pharmaceuticals, disease control and biological research [1]. DNA sequencing technology has continued to evolve for decades and has dramatically increased in precision, speed and cost. In particular, the next generation sequencing technology, nanopore-based sequencing, has made miniaturized sequencing devices and high-throughput sequencing a reality [2]. Briefly, nanopore sequencing technology employs cylinder-shaped protein-molecule sensors (“nanopores” or “pores”) through which DNA molecule pass. As a DNA strand goes through a pore it interferes with a constant ionic background current also passing through the pore. The resulting current fluctuations, reflections of the DNA’s nucleobase make up, are measured and this data is used to sequence the corresponding DNA molecule [3]. This sequencing process allows nanopore-based sequencing devices to offer several benefits: first, very long DNA fragments can be measured in one shot, thereby improving fragments assembly into complete genomes, reducing mapping certainty, and detection of structural variants [4].Second, nanopore-based sequencing devices can be made smaller, more portable, which not only enriches the application scenarios, but also makes the devices cheaper and lighter. For example, a commercial nanopore-based sequencer, the MinION, weighs only 87 g and is powered via USB [5]. Since the MinION still requires connection to a processing terminal for real-time sequencing, more recent released include so-called ”all-in-one” sequencers, the MinION Mk1C. This product not only has the traditional MinION, but also integrates an ARM microprocessor and a GPU, as well as DRAM and SSD. These allow it to process nanopore data independently without the need for other computing resources [7]. However, due to the closed source and licensing fees of GPU and ARM, the price of MinION Mk1C is 5 times of the normal MinION price. Therefore this paper argues that there is still a lot of potential research to be done in the area of embedded chip designs targeting mobile molecular analysis technologies. In this paper our focus is on DNA sequencing, in particular we investigate the utility of embedded processors to serve as nanopore-signal basecallers, a critical first computing step in a DNA sequencing bioinformatics ‘pipeline’ [8].

The authors are with the Department of Electrical Engineering and Computer Science, York University, Toronto, Canada. E-mails: {zhongpan, egz, magiero}@eecs.yorku.ca. and khammad2@yorku.ca.

Fig. 1. The Oxford nanopore MinION portable sequencing devices[6], the upper side is MinIOn Mk1C, the lower side is MinION.

As the RISC-V based open source ISA becomes more and more popular [9], this article will also explore the experimental studies on the implmentation of running basecaller under the use of the mature RISC-V architecture. The specific contributions of this paper are summarized as follows:

- A new energy-efficient basecaller built on open source RISC-V ISA is presented.
- The proposed basecaller considers different complexities for the basecalling algorithm which practically corresponds to different nanopore models.
- The RISC-V basecaller under test with other system components have been verified on our custom FPGA test platform.
- The performance of the whole system has been fully tested and the corresponding experimental results & conclusions have been drawn.

II. Viterbi Basecaller

The Viterbi algorithm is a very commonly used dynamic programming algorithm, among its applications are problems in telecommunication, speech recognition and bioinformatics [10]. Its principle is to compute a hidden Markov chain containing hidden states by computing each posterior probability of the observed sequence in processes amenable to description by a hidden Markov model [11].

In the existing nanopore model, although the diameter of the pore is small and can accommodate one DNA molecule, the pore channel inside the sensor is relatively long and can accommodate about 10 DNA molecules at a time [12]. Therefore, in the proposed Viterbi algorithm, the concept of K-mer is introduced to express how many DNA molecules are realistically detected in the pore at the same time, instead of the ideal situation where only one DNA molecule is usually assumed to be detected at a time. Secondly, three different pore detection scenarios are used to express a more accurate nanopore model, `Stay`, `Step` and `Skip`. `Stay` expresses the problem when a new DNA molecule is not detected in the pore for some reasons, such as a change in the DNA flow rate (too slow) within the pore or a blockage of the pore due to some large molecule. In the `step` case, each new DNA molecule that flows into the well is grouped with previously registered DNA molecules (containing K-mer DNA) and is detected by the nanopore sensor normally. `Skip` takes into account an another special case when the nanopore sensor misses a DNA group since the DNA flows too fast in the pore, or the current influence is not enough to be captured.

However, Viterbi as a base-calling algorithm is more suitable for the study of microchip architectures in this paper, which is attributed to its relatively simpler computational process and easier implementation. It also allows us to gain more experience in future research on neural network architectures.

III. Proposed RISC-V Basecaller

In previous work, we implemented the Viterbi algorithm for basecalling on an FPGA accelerator we designed [13], [14]. This FPGA hardware uses the PCIe bus on a desktop computer's motherboard to communicate with a conventional x86 processor. In this design the x86 is tasked with the execution of the basecalling program (in C++) but has access to libraries that allow it to schedule computationally heavy basecalling tasks on the FPGA. In essense, a programmer would be required to include special commands in their code to schedule the basecalling computations on the FPGA accelerator. In this paper, we explore a new basecaller built entirely out of a custom embedded processor utilizing the RISC-V ISA. This processor is implemented on one FPGA and carries out all the tasks previously handled by the x86 and FPGA accelerator. Thus, no special high-level language commands are needed in the basecalling program executed on our embedded processor.

The proposed basecaller is based on the RISC-V ISA open source microprocessor project - Rocket [15]. Unlike traditional hardware description languages, Rocket is implemented in a new hardware 'construction' language called `Chisel`, which has typical high-level programming language features and allows developers to focus more on the design of the architecture by trying to dilute the basic concepts of hardware design [16].

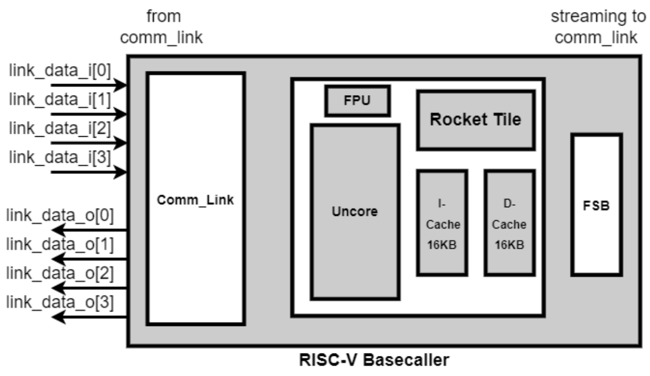

Fig. 2. The architecture diagram of the proposed RISC-V basecaller architecture.

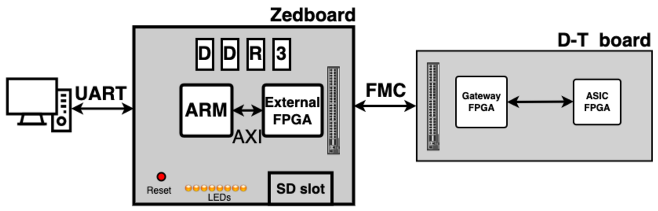

Fig. 3. The test system platform.

The system architecture of the RISC-V basecaller proposed in this paper is shown in Fig. 2, all data buses to and from the RISC-V basecaller are divided into four channels, each of 8 bits, by a high-speed source-synchronous IO interface called "Comm_Link" developed by [17]. This creates a 32-bit data bandwidth, and the incoming data is then passed directly to the microprocessor in the middle. It is also responsible for communicating with the other functional components of the off-chip logic (referring to the gateway FPGA to be mentioned below). The Uncore module is mainly responsible for communicating with the previously mentioned Comm_Link, and then storing the incoming instructions and data into the corresponding 4-way 16-KiB L1 cache for the decoder and ALU in the Rocket Tile to perform instructions. The front side bus (FSB) on the right side connects the off-chip components to the SoC.

IV. The Test Framework for the RISC-V Basecaller

In order to build the whole test system and pave the way for future ASIC research work, two FPGA development boards (shown in Fig. 3) with three FPGAs were used for the whole test work. one of them is a Xilinx Zedboard ZNQ7020(Z-B), which will provide external memory for our RISC-V basecaller (target) with external memory and communicates with the ARM (host). In addition, the Zedboard is also connected to the host via UART and issues operational commands to the whole system.

The other FPGA test board we made was the daughter board (D-B), which was originally used by UCSD's Basejump

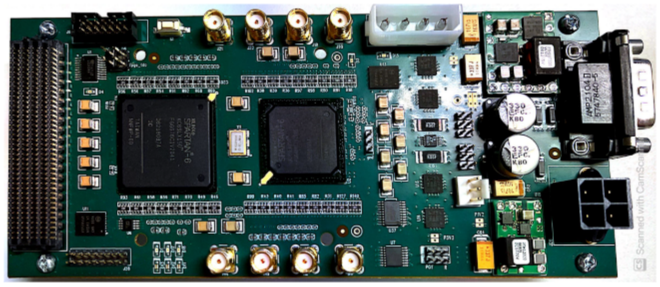

Fig. 4. The physical view of the daughter board (D-B).

project [18]. The D-B we built was equipped with two Spartan-6 FPGA chips, and an FMC interface Within the test system, this interface was used to communicate with the Zedboard connection. The FPGA on the FMC connector side of the D-B board is called the “gateway” (GW).

It is not only responsible for controlling and directing the boot of the D-B board, but also for the communication between the ARM on the ZB and with the ISC-V base converter (again, through the FMC connector). Moreover, it is also responsible for all clock signals. More specifically, it converts the 150-MHz crystal oscillator on the D-B into all the clock signals required by the ASIC by using Xilinx’s clock generator with built-in digital clock manager (DCM) IP. Moreover, it is also responsible for programming the voltage regulators and potentiometers on the board for future ASIC testing. Currently, the GW FPGA contains only a simple micro-blaze system for power management of the D-B board. The FPGA on the other side of the board is used as the ASIC emulator for the chip to test our proposed microprocessor. Our design files will be burned to this FPGA for more verification and testing.

To get the executable already compiled with the RISC-V version of GCC to run on our RISC-V basecaller, the ASIC FPGA used to emulate the RISC-V basecaller, we need to run two other programs. One is a lightweight system application running on RISC-V basecaller, that is, the proxy kernel (pk), which provides the system application interface to the binary executable running on top of it and passes IO-related system calls (syscalls) back to the host, that is, the ARM running the program called `fesvr`. It is responsible for running the syscall handler and for loading the `.elf` file.

After creating the basic flash image (.mcs format) for each of the two FPGAs, we burn the SPI flash corresponding to each FPGA by using the JTAG and setting different jumpers on the D-B board, so that the FPGA will automatically load the bit file from the flash when the D-B board is powered on.

V. Experimental Results

In this section, we use the single RISC-V basecaller core test platform to compare it with two other common computing platforms. The first one is the RISC-V basecaller platform to be tested. The second platform is a general-purpose x86 processor, and here we use a 12-core Intel Xeon E5-2620 v3 processor with a frequency of 2.4 GHz and 32 GB of DRAM for this platform. For the third test platform, we use the ARM

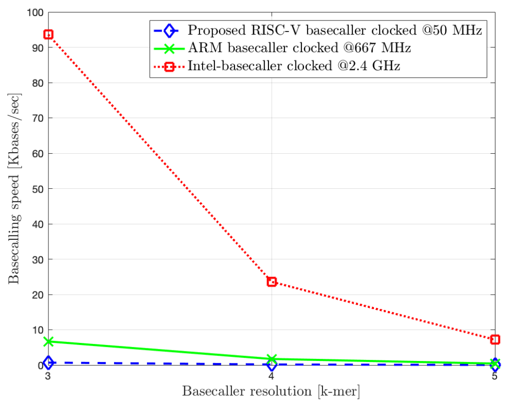

Fig. 5. Basecalling speed measurements (Kbases/sec) on three platforms.

core contained in the Zedboard Zynq chip, which is dual-core Cortex-A9, running at 667 MHz with 32-KB L1-cache, 512-KB L2-cache, 256-KiB on-chip memory. The board also comes with 1 GiB of DRAM. In addition, we evaluate the computational capacity of each platform using 3 different `mer` (mentioned in section II) values of basecaller: 3-mer (64-states), 4-mer (256-states) and 5-mer (1024-states). This means that as the `mer` value increases, the Viterbi algorithm has more branches under the optimal path to be computed, and the number of states to be decoded also increases exponentially.

To fairly test the computational speed, only a single core runs the basecaller on all platforms and measures the clock cycles of the run rather than the run-time. For x86 architectures, the same measurement as mentioned in [19], PAPI timers are used to measure the clock cycles of x86 CPUs. For ARM and RISC-V, the number of clock cycles of the basecaller is measured by reading the Control&Status Registers(CSR) inside the chip.

The results of the basecalling data are shown in Fig. 5, with the x-axis showing the three different `mer` values and the y-axis showing the speed in kilo bases per second (Kbases/sec). There is no doubt that the RISC-V basecaller running at 50 MHz is the slowest. This is mainly because the RISC-V basecaller runs at 50 MHz on the FPGA, which is only 7.5% of the speed of the ARM, and the I/O speed of the RISC-V basecaller is also limited (I/O is also 50 MHz).

At 3-mer, RISC-V is 1/10th the speed of ARM, while the fastest Intel basecaller is 14 times faster than the ARM, and runs at 48 times the frequency of RISC-V. At 4-mer, all three basecallers to be tested drop about 4 times compared to the 3-mer. At 5-mer, the speed of the RISC-V basecaller dropped again by 76%, the ARM by 75%, and the faster Intel x86 by only 69% as the computational task was further increased.

In the performance test shown in Fig. 6, we have also varied the amount of input data (in terms of number of signal samples emerging from a nanopore sensor) to the basecaller, as shown by the numbers on the x-axis, feeding data from 10, 100, 1000 to 10,000 samples, and measuring the number of clock cycles for each platform separately at the same number of inputs. It is clear to see that our basecaller basically overlaps with the ARM-basecaller, while the Intel-basecaller is clearly ahead of

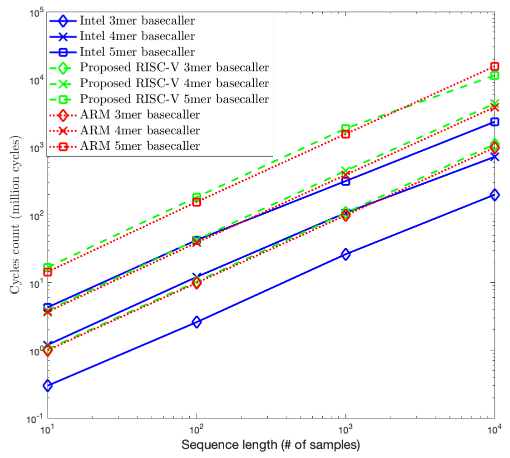

Fig. 6. Basecalling speed measurements (clock cycles) on three platforms.

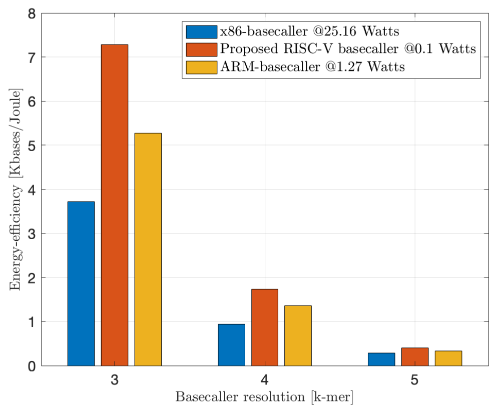

Fig. 7. Energy efficiency (Kbases/Joule) of three platforms.

the other two basecallers, mainly due to its high performance computing core, and large cache.

At the end of the experiment, we recorded the additional power consumption of the three platforms running the basecaller. That is, the power consumption at idle (i.e., the computer is operating, but no basecalling application is actually executing on the machine) compared to the full power consumption when running the basecaller. The additional power consumption of other non-computing core components is not recorded. For example, in idle mode, the Intel x86 CPU consumes 62 W, compared to 2.5 W for the ARMCortex-A9 and only 0.2 W for the ASIC FPGA on the D-B. It should be noted that this value is read from the power supply dedicated to the ASIC FPGA. While running the basecaller, we again recorded the power consumption of each test platform. The most powerful Intel basecaller consumes an extra 25.16 W. In comparison, the ARM Cortex-A9 on the Zedboard consumes an extra 1.27 W. The proposed RISC-V basecaller on the ASIC FPGA consumed only an additional 0.1 W.

In Fig. 7, we show the energy consumption ratio of the basecaller under the three platforms at different resolutions ($\mathtt{mer} = 3, 4, 5$) and when $\mathtt{mer}$ is 3, it can be concluded that, compared to Intel x86, the proposed RISC-V basecaller has an energy efficiency ratio of 1.95 times better. Even when compared to ARM Core, which has an excellent energy efficiency ratio, it still has a 38% energy efficiency ratio advantage.

VI. Conclusion

In the face of increasingly challenging basecalling tasks for computation, this paper presents a RISC-V ISA implementation of a basecaller, and after implementing our custom FPGA in the verification platform, it fully demonstrates a competitive energy efficiency ratio ahead of other traditional architectures (x86, ARM). Although the current test platform still needs more refinement. It has laid the foundation for our next research, such as the possibility of adding multiple cores, optimizing the logic and boosting the frequency.

References

[1] J. A. Shendure, G. J. Porreca, and G. M. Church, “Overview of dna sequencing strategies,” *Current Protocols in Molecular Biology*, vol. 81, no. 1, pp. 7–1, 2008.
[2] S. Reid, J. A. Clarke, J. White, and G. Harper, “Analysis of measurements of a polymer,” U.S. Patent US2015/0 057 948A1, Feb. 26, 2015.
[3] D. Deamer, M. Akeson, and D. Branton, “Three decades of nanopore sequencing,” *Nat. Biotechnol.*, vol. 34, no. 5, pp. 518–524, May 2016.
[4] MinION Mk1C Technical Specification, April 2020, Available at: MinION Mk1C Technical Specification.
[5] H. Lu, F. Giordano, and Z. Ning, “Oxford nanopore minion sequencing and genome assembly,” *Genomics, proteomics & bioinformatics*, vol. 14, no. 5, pp. 265–279, 2016.
[6] O. nanopore, “Minion,” Aug 2021. [Online]. Available: https://nanoporetech.com/products/minion
[7] P. DAS, R. DADWAL, V. RADHAKRISHNAN, M. PARIHAR, S. BHATTACHARYA, D. MISHRA, and M. CHANDY, “Comparison of four high throughput sequencing platforms in a medical laboratory for gut microbiome research,” *Indian J. Anim. Hlth*, vol. 59, no. 2, pp. 89–99, 2020.
[8] X. Wang, *Next-generation sequencing data analysis*. CRC Press, 2016.
[9] A. S. Waterman, *Design of the RISC-V instruction set architecture*. University of California, Berkeley, 2016.
[10] C. Cheng and K. K. Parhi, “Hardware Efficient Low-Latency Architecture for High Throughput Rate Viterbi Decoders,” *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 55, no. 12, pp. 1254–1258, 2008.
[11] W. Timp, J. Comer, and A. Aksimentiev, “DNA Base-calling from a Nanopore using a Viterbi algorithm,” *Biophysical J.*, vol. 102, no. 10, pp. L37–L39, May 2012.
[12] Z. Wu, K. Hammad, Y. Dawji, E. Ghafar-Zadeh, and S. Magierowski, “Hardware accelerated dna sequencing,” in *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2018, pp. 550–551.
[13] Z. Wu, K. Hammad, E. Ghafar-Zadeh, and S. Magierowski, “FPGA-Accelerated 3rd Generation DNA Sequencing,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 14, no. 1, pp. 65–74, 2020.
[14] K. Hammad, Z. Wu, E. Ghafar-Zadeh, and S. Magierowski, “A Scalable Hardware Accelerator for Mobile DNA Sequencing,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 2, pp. 273–286, 2021.
[15] K. Asanovic, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio, H. Cook, D. Dabbelt, J. Hauser, A. Izraelevitz *et al.*, “The rocket chip generator,” *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17*, vol. 4, 2016.
[16] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avižienis, J. Wawrzynek, and K. Asanović, “Chisel: constructing hardware in a scala embedded language,” in *DAC Design automation conference 2012*. IEEE, 2012, pp. 1212–1221.
[17] M. B. Taylor, “Basejump stl: Systemverilog needs a standard template library for hardware design,” in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.
[18] Michael Taylor, Available at: http://bjump.org/bridge.
[19] J. Koenig, D. Biancolin, J. Bachrach, and K. Asanovic, “A Hardware Accelerator for Computing an Exact Dot Product,” in *IEEE 24th Symposium on Computer Arithmetic (ARITH)*, 2017, pp. 114–121.