4KB SRAM Memory Controller with Parameterized CRC Accelerator

Alenkruth Krishnan Murali[1] and Khyati Kiyawat[2]

[1]Ph.D. Student, Department of Electrical and Computer Engineering, University of Virginia
[2]Ph.D. Student, Department of Computer Science, University of Virginia

March 27, 2023

1 Overview of the project

In this course project, we aim to design a simple memory controller with a parameterized Cyclic Redundancy Check (CRC)[1]–[3] accelerator for a 4KB SRAM[4] designed with OpenRAM[5]. We plan to use the same CRC Accelerator design for both the FPGA and ASIC sections of the course.

We will create a minimal, area-optimized memory controller with an integrated CRC accelerator. We will integrate the CRC Accelerator into the simple memory controller and interface the memory controller with a 4KB SRAM created with OpenRAM using SkyWater130[6] PDK. We will use the OpenLane[7] tool flow for our project and will try to optimize the design to meet the eFabless-MPW shuttle submission requirements.

2 Motivation

Cyclic Redundancy Check (CRC) is widely used as an alternative to parity and checksum calculations for checking and correcting errors in data transmissions as well as stored data. We propose to design an accelerator to perform CRC which speeds up the performance of the entire system. In order to keep the complexity tractable, we will integrate the accelerator into a simple memory controller for a 4KB SRAM array.

Unlike the current open-source CRC accelerators[8], [9], our design will be parameterized to work on 8, 16, 32 and 64-bit polynomial widths. We also plan to support user-specified CRC polynomials since the choice of the polynomial is paramount to the efficacy of the procedure.

Finally, we are excited to be able to create a simple, area-optimized, CRC-enabled memory controller with the open-source tooling available, and hopefully, be able to submit our project to the eFabless shuttle. Moreover, we are excited to explore the OpenRAM framework to create a 4KB SRAM with the SkyWater130 PDK.

3 Outline of the project

The CRC method for error detection and correction treats the data frame as a huge binary number. The binary number is divided (at the CRC generation end) by a fixed binary number (the CRC generator polynomial) and the resulting remainder of this division (CRC value) is appended to the end of the data frame. The receiver upon reception of the data frame repeats the calculation and compares its calculated CRC value with the CRC value attached to the data frame.

3.1 Memory Controller

We will create a simple memory controller that can handle a single memory read/write request at any given point in time. Since our focus is on creating the CRC Accelerator and the 4KB SRAM array, our memory controller will only contain an address decoder coupled with a control logic signal generator. The block diagram of the overall system is shown in Figure.1.

3.2 SRAM Array

We will be using OpenRAM to create the 4KB SRAM array. Since we do not have a lot of experience with the OpenRAM framework or with SRAMs, we plan to follow the project report from a previous project from 2022[10](http://venividiwiki.ee.virginia.edu/mediawiki/index.php/OpenRAM:_a_4KB_SRAM_array_-_Jerry_Yin,_Xuanjia_Bi). We will follow the same steps to try and create a memory array with the Skywater130 PDK.

3.3 CRC Accelerator

The traditional method for implementing a CRC unit uses a shift register with XOR gates and feedback taps. We use the same fundamental hardware gates and will be trying to modify their architecture to obtain better performance.

3.3.1 Parameterizability

The current open-source CRC accelerators operate on a fixed polynomial. While this approach makes the design easier, it is not the most efficient solution. As the CRC polynomial can be varied with the performance and reliability requirement of the system, we propose to design a configurable CRC accelerator. The CRC polynomial can be entered as a parameter to our design, thereby having a **parameterized CRC accelerator** that will read data from an interface register. The user can also provide the polynomial width. Otherwise, we will compute CRC with the default polynomial(32-bit).

3.3.2 Performance Optimization

The CRC unit will be pipelined to operate at the highest possible frequency. Since the polynomial is fixed per operation, we can keep the same copy of the polynomial and perform a series of XORs to obtain the final CRC. We can also use buffers to hide the latency of the unit from the main processing unit.

For the ASIC section of the project, we will be prioritizing area over performance. Hence we might deviate from the previously listed performance optimizations, if necessary.

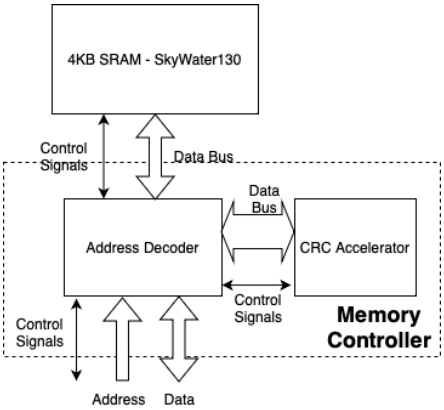

Figure 1: Block diagram of the simple 4KB SRAM memory controller interfaced with the CRC Accelerator. *The original design might differ.*

We will use the OpenLane tooling flow for the memory controller design to obtain the GDSII files. The final goal of the project is to be able to tape out the design through the Efabless OpenMPW program. In order to meet our final goal, we might prioritize the area footprint of our accelerator in the ASIC section of the project. Hence, we expect that we might have two different designs meeting different design requirements (performance in the case of FPGA and area in the case of ASIC). Nevertheless, we will run tests to obtain the performance benefits/overheads of our CRC-integrated memory controller.

4 Background work/Literature Survey

As of now, we are reading through the design manuals and documentation listed in the references to ascertain the feasibility of our project and the time/effort it would entail. We are also checking the already available CRC Accelerator Designs[8], [9], [11] to refine our design. We are also installing and familiarizing ourselves with the open-source tools which we will be using for the project. We have also created a GitHub repository to use for our project at https://github.com/Alenkruth/crc_accelerator

5 Expected Outcome

Through this project, we expect to create an open-source, parameterized CRC Accelerator using System Verilog.

We expect to have created and tested a simple-memory controller for a 4KB SRAM array designed with Skywater130 PDK nodes and also houses an integrated CRC Accelerator. We will redesign the accelerator, if necessary, to meet the area requirements for the OpenMPW shuttle program. We will generate the GDSII file for the system by using only open-source tools and

designs. Finally, we will demonstrate the functioning of the memory controller and CRC unit with a suitable testbench/test program.

6 Expected Challenges

We expect to face a few challenges while working on this project. They are,

1. Floorplan, and Place and route the entire design without violating the 10x10mm^2 area constraint or DRCs.
2. Integrating the SRAM with the memory controller might be difficult.
3. Other Tool-chain problems which we currently do not know of but anticipate for.
4. Methods/approaches to test the system post-layout.

References

[1] W. W. Peterson and D. T. Brown, "Cyclic codes for error detection," *Proceedings of the IRE*, vol. 49, no. 1, pp. 228–235, 1961. DOI: 10.1109/JRPROC.1961.287814.

[2] P. Koopman and T. Chakravarty, "Cyclic redundancy code (crc) polynomial selection for embedded networks," in *International Conference on Dependable Systems and Networks, 2004*, 2004, pp. 145–154. DOI: 10.1109/DSN.2004.1311885.

[3] P. Koopman, "32-bit cyclic redundancy codes for internet applications," in *Proceedings International Conference on Dependable Systems and Networks*, 2002, pp. 459–468. DOI: 10.1109/DSN.2002.1028931.

[4] Wikipedia, *Static random-access memory — Wikipedia, the free encyclopedia*, [Online; accessed 27-March-2023], 2023. [Online]. Available: https://en.wikipedia.org/wiki/Static_random-access_memory.

[5] M. R. Guthaus, J. E. Stine, S. Ataei, B. Chen, B. Wu, and M. Sarwar, "Openram: An open-source memory compiler," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 1–6. DOI: 10.1145/2966986.2980098.

[6] SkyWater Open Source, *Skywater pdk*, [Online; accessed 27-March-2023], 2023. [Online]. Available: https://skywater-pdk.readthedocs.io/en/main/.

[7] The OpenROAD Project, *Openlane*, [Online; accessed 27-March-2023], 2023. [Online]. Available: https://openlane.readthedocs.io/en/latest/.

[8] *PCI Express CRC generator*, https://opencores.org/projects/pci_express_crc, [Online; accessed 27-March-2023].

[9] *Ultimate CRC*, https://opencores.org/projects/ultimate_crc, [Online; accessed 27-March-2023].

[10] J. Yin and X. Bi, "Openram: A 4kb sram array." [Online]. Available: http://venividiwiki.ee.virginia.edu/mediawiki/index.php/OpenRAM:_a_4KB_SRAM_array_-_Jerry_Yin,_Xuanjia_Bi.

[11] M. W. Azhar, T. T. Hoang, and P. Larsson-Edefors, "Cyclic redundancy checking (crc) accelerator for the flexcore processor," in *2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, 2010, pp. 675–680. DOI: 10.1109/DSD.2010.51.