PV200 Semestral project

Tomas Jaros

February 5, 2023

---

1 Introduction

In this semestral project, the task was to use hardware description languages, such as Verilog or VHDL, and try to implement nan MD5 cracker. My solution tried to implement two versions. One mostly in software programmed in C and running on softcore CPU communicating with the MD5 core through several registers available in the custom functions unit of the CPU. The second one is supposed to be running purely without any software.

2 Softcore version

The softcore version uses MD5 core from [4], which was my choice because the code is understandable to me and, thus, easier to work with. As was mentioned, the setup consists of a softcore processor which runs C code to communicate with the MD5 implementation. The CPU used is called NEO430 [3]. It's a 16-bit CPU, which provides several peripherals, such as UART, to transfer data from the computer to NEO430 and possibly interact with the code running on it.

The setup for the MD5 cracker was that we needed to implement communication between the MD5 core and the code running on the CPU. The CPU provides a peripheral called CFU, which stands for custom function unit, where the users can implement their own extensions. The CFU communicates with the outside world through eight 16-bit registers. What I did was that I defined some kind of "protocol" to control the MD5 core, pass data to it, and receive the outputs. Then in the C code, I used the fact that I now know how to control and manipulate the MD5 core to implement the software API. Then I coded a simple letter combination algorithm to try possible combinations sequentially.

This description does sound very ineffective and slow. And in fact, it is; the performance of the cracker is terrible. There are several reasons for that. The algorithm of the hacker is in software; the CPU needs several clock cycles just to process the instructions given to it when it communicates with the MD5 core. Then the communication itself takes time. The MD5 core implementation, in this case, isn't the fastest, such as those which can perform one hash within a clock cycle.

3 Hardware version

As the version that uses the softcore CPU in the way I implemented it is slow, I wanted to try a bit faster version. The first thing that I needed to do was to make some kind of communication between my computer and the FPGA available. As I was advised and given a USB2UART bridge, it seemed like a good choice to use UART. I tried several implementations of UART in Verilog or VHDL. I tried to implement one myself, but that didn't work out well. So then I settled for one implementation [1], which was finally well documented and worked well.

The next part was finding the appropriate MD5 core and implementing the letter combination circuit. This task was a bit daunting, and I did not manage to accomplish anything worthwhile. By chance, I found out about the master's thesis [2] of a former student Maruthi Gillela at my faculty whose objective was parallelization of the MD5 brute force attack, so I thought about giving it a try to at least implement a wrapper to use his implementation. I created such a thing, saving the data received from UART and controlling Gillela's implementation. The problem was that to use his implementation and make it work would require a very powerful FPGA, and the compilation times for even a smaller part of his design, with halved-down alphabet and cores takes several hours and does not fit into FPGA, which I was assigned for this project.

4 Conclusion

I did create an MD5 cracker, albeit a very slow one, with no practical use. I tried to make a faster version of it work, but I didn't succeed due to several limitations, such as my limited knowledge of FPGA programming. However, I would argue that this project positively affected my experience with the FPGA ecosystem.

References

[1] Jakub Cabal. uart-for-fpga. https://github.com/jakubcabal/uart-for-fpga.

[2] Maruthi Gillela. Parallelization of brute force attack on md5 hash algorithm in fpga [online]. Diplomová práce, Masarykova univerzita, Fakulta informatiky, Brno, 2019 [cit. 2023-02-04]. SUPERVISOR : Václav Přenosil.

[3] Stephan Nolting. The neo430 processor. https://github.com/stnolting/neo430.

[4] Joachim Strömbergson. md5. https://github.com/secworks/md5.