Creating Software for a Scanning Probe Microscopy

Nicolas Azzi

Department of Physics | Florida State University | Research Mentor: Dr. Ni, Guangxin

Abstract

We created custom software for a scanning probe microscope (SPM) to support various FSU research projects and experiments. We will begin by giving a brief overview of SPM and its potential applications in scientific research. Following that, we'll talk about why we need specialized software and how it fits our needs. We will also discuss the time and effort required to create this type of machinery. Our presentation will primarily concentrate on the software development process and the key features of the software that we developed. In addition, we will go over the programming languages used for this project and why they were chosen over others.

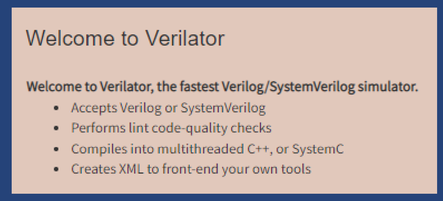
Introduction

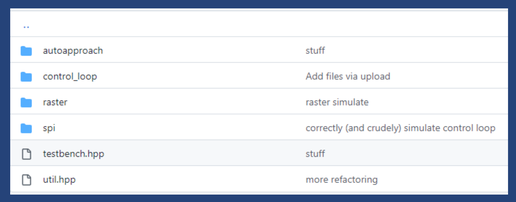
Building a working scanning probe microscopy (SPM) that uses a probe that can scan the surface of a certain element and build an image from that interaction. The probe must be able to magnify and construct the image to a nanoscopic level. Creating a SPM would allow researchers at the National Magnet Laboratory to do important research that requires a microscopy. It would give researchers high resolution images on a nanoscale level on the samples being research on. It could be used in different fields such as materials science, biology, electronics, chemistry, etc., this equipment could be a steppingstone for major innovation.

Methodology

The language the machine software used was Verilog, C++, C, and python. Learning Verilog was extremely difficult as the resource to learn about the language was scarce. Verilog is a low-level language that was used along with some C++ code to run on the machine. It took hours trying to find resources that explained what Verilog was and how to use it. I learned about FPGAs and ASICs as it was associated with the language. It took a while to learn the syntax, planning and designing, and how to optimize the code in Verilog. After learning Verilog, I am now in the process of how to design the software for the microscopy and the way we come at it is through thinking what exactly would be helpful for the researcher to have when running test on the SPM.

```
int main(int argc, char **argv) {
        init(argc, argv);

        int Con,                        // Continue option for user
        P = 3,                          // Default P value
        I = 0,                          // Default I value
        Delay = 20,                     // Default Delay value
        SetPt = 10000,                  // Default SetPt value
        Status = 1,                     // Default Status value
        Option_Picked;                  // Option user picked

        do
        {

                mod = new ModType;
                Transfer func = Transfer{150, 0, 2, 1.1, 10, -1};

                mod->clk = 0;

                // Testing this char for P value
                char Data_Change_S = '0b1101011100001010001111010111000010100011';

                // Default value for P, some type of error where it won't accept any string, int | Try char next time
                set_value(0b1101011100001010001111010111000010100011, CONTROL_LOOP_P);

                // Default value
                set_value((V)6 << CONSTS_FRAC, CONTROL_LOOP_I);
                set_value(Delay, CONTROL_LOOP_DELAY);
                set_value(SetPt, CONTROL_LOOP_SETPT);
                set_value(Status, CONTROL_LOOP_STATUS);

                // Menu
                do
                {
                        printf("%15s\n", "Menu");
                        printf("1) Set Control loop P    Current value: %d\n", P);
                        printf("2) Set Control loop I    Current value: %d\n", I);
                        printf("3) Set Delay             Current value: %d\n", Delay);
                        printf("4) Set setpoint          Current value: %d\n", SetPt);
                        printf("5) Set status            Current value: %d\n", Status);
                        printf("6) Continue\n");

                        // Checks for what option user picks
                        scanf("%d", &Option_Picked);

                        // Clears unix screen
                        system("clear");
```





To run the Verilog and C++ code we used a program called Verilator. Verilator tested the software we created and allowed us to debug anything that we needed testing with. This is an extremely important aspect of coding, without testing our code it could lead to vulnerabilities and issues with the software.

Conclusion / Results

Although our study project is still in its early stages, we have made some advancements and are pleased with our results thus far. Within the following years, our group will have finished developing the SPM. Once completed, this hardware will be a useful tool to further the development of other research projects and act as a steppingstone for other researchers in the field. We are eager to carry out this research and observe how it might affect the scientific community.

Acknowledgements

Special thanks to Dr. Ni for letting me be a part of this project, and Peter McGoron for guiding me throughout this learning experience and being an extra mentor.

Thanks to Gisselquist Technology for creating free and well done educational tutorials on verilator.

References

Russell. “Verilog Tutorial.” Nandland, 30 June 2022, https://nandland.com/introduction-to-verilog-for-beginners-with-code-examples/.

Snyder, Wilson. “Verilator.” Veripool, https://www.veripool.org/verilator/.

Technology, Gisselquist. “Verilog Tutorial.” Verilog, Formal Verification and Verilator Beginner's Tutorial, https://zipcpu.com/tutorial/.