Dynamic Programming Based FPGA Hosted Control-Flow Integrity (CFI)

DESIGN DOCUMENT

sdmay21-10
Professor Akhilesh Tyagi
Advisers: Ananda Biswas and Zelong Li
Greg Wendt/Meeting Scribe
Cole Schumacher/Meeting Facilitator
Nickolas Mitchell/Chief Engineer
Samuel Henley/Chief Engineer
Maxwell Wangler/Test Engineer
Tristan Duyvejonck/Report Manager
sdmay21-10@iastate.edu
http://sdmay21-10.sd.ece.iastate.edu

Revised: 9/16/2020 v1.0

Executive Summary

Development Standards & Practices Used

We use the following standards for interfacing with the FPGA:

- FMC (FPGA Mezzanine Card): Most popular and versatile interface standard
- Digital PMOD: A nice, simple alternative to FMC
- SYZYGY: newer standard that provides a middle ground between these two
- Standard design practices:
    - Well-documented
    - Efficient use of resources
    - Using proper IP's and memory cores

We use the following standards in regards to Control-Flow Integrity:

- Current CFI strategies
    - Stack Canaries
        - Non-executable memory
    - Write XOR eXecute
    - Code randomization
- Coarse-grained CFI
- Fine-grained CFI
- We want the best possible security with the lowest possible performance impact

Summary of Requirements

A high level overview of the function requirements is given below:

- We will create an algorithm to match two sets of strings
- One string is generated based on program control flow, and is compared to a defined string

- It will use a two dimensional grid with its sized based on the number of basic blocks in the program

A high level overview of the nonfunctional requirements is given below:

- The algorithm must perform fast. The goal is O(nlog(n)).

Applicable Courses from Iowa State University Curriculum

List all Iowa State University courses whose contents were applicable to your project.

New Skills/Knowledge acquired that was not taught in courses

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum in order to complete this project.

Table of Contents

List of figures/tables/symbols/definitions (This should be the similar to the project plan)

1 Introduction

1.1 Acknowledgement

Work on this project is taking place along with Dr. Akhilesh Tyagi, and graduate assistants Ananda Biswas and Zelong Li. All of which have a significant expertise in the computer engineering and software engineering field. Our team is working closely with these assistants to familiarize ourselves with the project and the problems we need to solve.

1.2 Problem and Project Statement

General Problem Statement:

> We are trying to determine if we can detect a break within the control flow of a program.

General Solution Approach:

> To do this, we are trying to simplify the problem into one similar to a DNA sequencing problem using dynamic programming.

1.3 Operational Environment

The Operational Environment for this program will be lab rooms. No extreme conditions are expected.

1.4 Requirements

A high level overview of the function requirements is given below:

- We will create an algorithm to match two sets of strings
- One string is generated based on program control flow, and is compared to a defined string
- It will use a two dimensional grid with its sized based on the number of basic blocks in the program

A high level overview of the nonfunctional requirements is given below:

➢ The algorithm must perform fast. The goal is O(nlog(n)).

1.5 Intended Users and Uses

As the program is experimental, we don't have a set user base in mind at the moment beyond the Professor and Graduate Students.

1.6 Assumptions and Limitations

Assumptions:

- The solution for DNA sequencing can be applied to this problem.

Limitations:

- We must use dynamic programming to determine if we lost control flow integrity.
- We must use a FPGA board to test our solution.
- The FPGA board shouldn't cost more than $100.

1.7 Expected End Product and Deliverables

The deliverable for this project is an FPGA board with a working program. The program shall compare two strings: one generated based on program control flow and a defined one. The expected delivery date for this is April 30, 2021.

The expected delivery dates are as follows:

- Ideate solutions: October 26, 2020
- Test solution in higher-level programming language: November 20, 2020
- Implement race conditions logic: February 12, 2021
- Implement solution in FPGA: April 30, 2021

2 Project Plan

2.1 Task Decomposition

1. Develop solution (Meant for understanding race logic/cfg)
    a. Write pseudocode for solution
    b. Develop solution in higher-level programming language
    c. Test solution
    d. Look at binary/cfg (LLVM)
2. Implement solution in FPGA board
    a. Implement using FPGA emulator
    b. Implement on actual FPGA
3. Features (TBD at a later date with the clients)

2.2 Risks And Risk Management/Mitigation

- Possibly override code of other team members
    - The use of git prevents this to a high degree.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Our first key milestone includes, completing development on the software solution which will detect whether there is a significant breach in Control Flow Integrity with above 50% accuracy. Note, the target accuracy is subject to change based on testing. Our team's last milestone will be completing implementation on hardware with accuracy similar to our software solution.

2.4 Project Timeline/Schedule

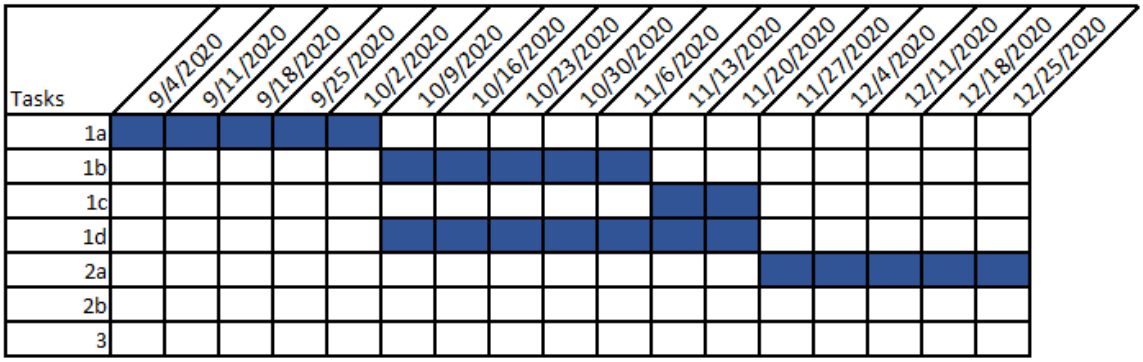


Table 2.1: Gantt Chart for tasks (2020)

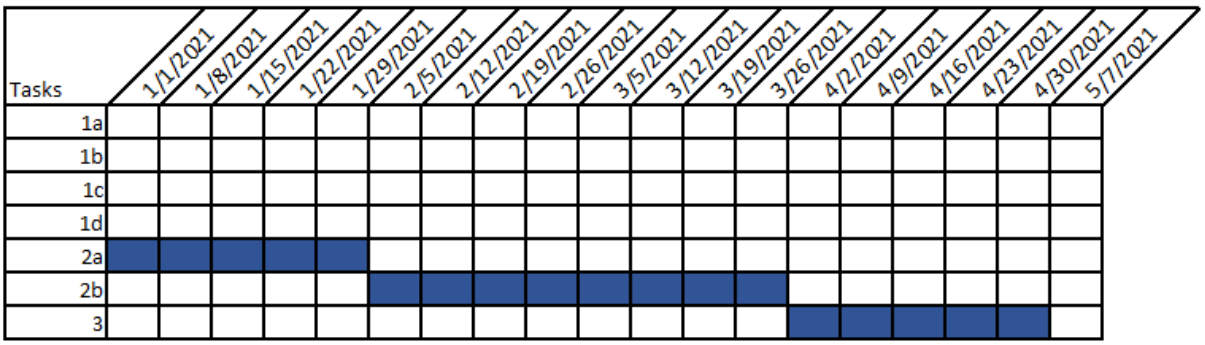


Table 2.2: Gantt Chart for tasks (2021)

Tables 2.1 and 2.2 show the proposed schedule for completing tasks laid out in Section 2.1.

2.5 Project Tracking Procedures

- Our group is using Slack to communicate with the team
- Our group is using a Google Drive to store team documents.
- Our group is using Github to store files relating to the project.
- Our group is a combination of GMAIL, Webex, and ZOOM, to communicate with our clients.

- Our group is using Trello to keep track of assigned tasks and due dates.

2.6 Personnel Effort Requirements

Task	Hours Per Person	Team members assigned	Total person-hours
1a	4	1	4
1b	5	2	10
1c	5	3	15
1d	15	3	45
2a	10	3	30
2b	15	2	30
3	TBD	TBD	TBD

Table 2.3: Effort Estimate by task

Our team will expect all team members to complete the tasks assigned to them by a predetermined deadline determined by the team. Our team expects total weekly hours per member to not exceed around 40 hours in week given several tasks to be completed in said week. In general, our team expects all members to give an appropriate number of hours to provide quality while still time managed work.

2.7 Other Resource Requirements

A basic FPGA board will be the only required resource.

2.8 Financial Requirements

Our team plans on no financial requirements being necessary.

3 Design

3.1 Previous Work And Literature

A. Madhavan, T. Sherwood and D. Strukov, "Race Logic: Abusing Hardware Race Conditions to Perform Useful Computation," in *IEEE Micro*, vol. 35, no. 3, pp. 48-57, May-June 2015, doi: 10.1109/MM.2015.43.

This paper was useful in understanding the algorithm we use in the system. The authors applied race conditions to solve a string-matching problem related to protein sequencing. We want to adapt the algorithm to analyze control flow integrity.

3.2 Design Thinking

To define the problem that shaped our design, we worked closely with the professor to understand the needs and core issues of monitoring Control Flow Integrity. Our final goal is to implement the solution on an FPGA board, but first we wanted to focus on developing a solution that could be implemented and tested in a high-level environment.

When we ideated solutions for the problem, we discussed ideas with our graduate assistants. We focused on one solution and built on it until we could determine if the solution was feasible.

3.3 Proposed Design

We are not using any particular design standards at this time. Our first step was to design pseudo code and come up with ideas that theoretically solve the problem. After we discuss the pseudocode with the client, we want to try to implement the pseudo code in a higher-level programming language and test its functionality.

3.4 Technology Considerations

One design alternative available is to use an ASIC system instead of an FPGA. We think the project would be worse off with an ASIC as the processing cycles would be interrupted by the primary compute unit. The ASIC unit would also be more difficult to scale.

3.5 Design Analysis

We have not gotten to a point where we can test the design yet.

3.6 Development Process

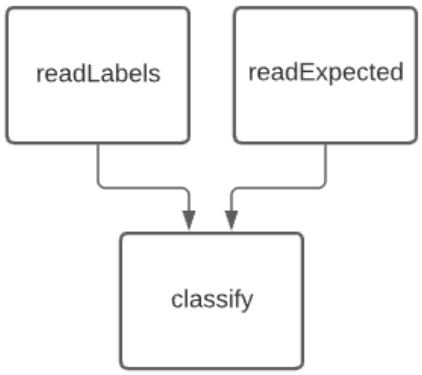
Our team is using a waterfall model development process. We decided the nature of our project was not suited for an agile development process with small, incremental iterations. Our project will move through each stage of the development process once and in order.

3.7 Design Plan

The system is composed of three modules:

1. Reading labels from binary
2. Read expected labels
3. Classifier

The first module reads control labels from the executed binary. The labels could be the PMU data from the processor, or another label to be determined. The second module reads the expected control flow model for the executable. The data from both modules is passed to the classifier, which determines whether there is a breach in control flow integrity.



4 Testing

4.1 Unit Testing

Software we plan to test in an isolated environment includes our dynamically programmed race logic algorithm, and any CFG programs we design. Hardware we plan to test in an isolated environment includes an FPGA board running our solution.

4.2 Interface Testing

The communication between the reading modules and the classification module will be tested by recording the data output by the reading modules and analyzing the behavior of the classification module.

4.3 Acceptance Testing

To verify the functionality of the system, develop input sets for best-case and worst-case scenarios. Apply the input sets to the system. For example, create a scenario when the model and the input data match exactly. Upon completion of the project, we will demonstrate the system to the client during one of our meetings.

4.4 Results

Thus far, our team has yet to produce any meaningful results during our testing phase. Please note, our testing phase is ongoing.

5 Implementation

Describe any (preliminary) implementation plan for the next semester for your proposed design in 3.3.

6 Closing Material

6.1 Conclusion

Summarize the work you have done so far. Briefly reiterate your goals. Then, reiterate the best plan of action (or solution) to achieving your goals and indicate why this surpasses all other possible solutions tested.

6.2 References

List technical references and related work / market survey references. Do professional citation style (ex. IEEE).

6.3 Appendices

Any additional information that would be helpful to the evaluation of your design document.

If you have any large graphs, tables, or similar data that does not directly pertain to the problem but helps support it, include it here. This would also be a good area to include hardware/software manuals used. May include CAD files, circuit schematics, layout etc,. PCB testing issues etc., Software bugs etc.