Instruction Level Reverse Engineering (Disassembly) through EM Side Channel

DESIGN DOCUMENT

Team Number: 09
Client: Prof. Akhilesh Tyagi
Advisers: Varghese Vaidyan

Team Members:
Matthew Campbell
Noah Berthusen
Cristian George
Jesse Knight
Jacob Vaughn
Evan McKinney

sdmay21-09@iastate.edu
https://sdmay21-09.sd.ecec.iastate.edu

Revised: 10/02/2020

Executive Summary

Development Standards & Practices Used

- Agile Scrum model used
- IEEE UART protocols
- Python PEP 8 Style
- Arduino Style Guidelines
- IEEE Code of Ethics

Summary of Requirements

- Program collects EM data and converts it to a usable format
- Model will predict opcodes with 90%+ accuracy and operands with 80%+ accuracy.
- Written in Python
- Well-documented code
- Predictions are formatted in a user-friendly format
- Large amount of data used to train the model

Applicable Courses from Iowa State University Curriculum

- COM S 311 (Introduction to the Design and Analysis of Algorithms)
- COM S 474 (Introduction to Machine Learning)
- CPRE 288 (Embedded System I)
- CPRE 381 (Computer Organization and Assembly Level Programming)
- CPRE 482x (HW Design for Machine Learning)
- EE 224 (Signals and Systems I)
- EE 321 (Communication Systems I)
- EE 201 (Electrical Circuits)
- EE 230 (Electronic Circuits and Systems)

New Skills/Knowledge acquired that was not taught in courses

- Convolutional neural networks (CNNs), Markov chains
- Side-channel observation of processors

Table of Contents

1 Introduction

1.1 Acknowledgement

We would like to thank Vhargese Vaidyan for sharing his knowledge about working with the EM side channel for reverse engineering, and for sharing his equipment and experience with us in order to jump start the project. Additionally, we'd like to thank ETG for letting us check out an oscilloscope for the entire Fall semester allowing us to begin long-term data capture.

1.2 Problem and Project Statement

General problem statement:

We want to be able to determine the assembly level code that is currently running on a processor by only reading the electromagnetic radiation that comes off of the processor. This kind of research has cyber security implications in that you could bypass a lot of security if you could find out what code is running just by measuring the physical electromagnetic (EM) radiation that the processor gives off.

General solution statement:

Our solution is to capture data using an electromagnetic probe and send that data to a machine learning algorithm. The machine learning algorithm will be able to look at the data and the surrounding data points to determine with a degree of certainty what opcode and operand is being ex

1.3 Operational Environment

The resulting end product from this project will be used in a laboratory environment with minimal electromagnetic interference. At the moment, the operational environment will be in 301 Durham. 301 Durham has an oscilloscope and EM antenna powerful enough to capture the radiation emitted by our microcontroller.

1.4 Requirements

Requirements:

- Program collects EM data and converts it to a usable format
- Model will predict opcodes with 90%+ accuracy and operands with 80%+ accuracy.

Non-functional requirements:

- Written in Python
- Well-documented code
- Predictions are formatted in a user-friendly format
- Large amount of data used to train the model

1.5 Intended Users and Uses

Our single intended use is to measure EM radiation from a Cortex Arm M4 processor and output the corresponding opcodes and operands. The intended user for our project is our client Akhilesh Tyagi and other EM side-channel researchers.

1.6 Assumptions and Limitations

Assumptions:

- End users will have access to the necessary hardware and software
- Design is running on a Cortex Arm M4 processor with a 4 stage pipeline running at 200 MHz
- EM is measured using same tools used to train the machine learning model

Limitations:

- Program will need to run on a high-end GPU
- Input data to model must be in a specific format
- Processor must have at least a 4 stage pipeline

1.7 Expected End Product and Deliverables

1. A machine learning algorithm capable of 90% opcode, and 80% operand detection:

   The machine learning algorithm will be delivered at the end of the project, approximately May 2021. The algorithm will be created using python and will include the datasets used to test, train, and validate the algorithm. Additionally, the algorithm will include documentation describing how to redeploy the algorithm to a separate system.

2. Automated data extraction tool to pull data from EMR probe.

   The extraction tool will take data from either the oscilloscope or digitizer and convert it into a format usable by the machine learning algorithm. This tool will allow the client to easily extract additional data to run against the trained algorithm. We expect to have this data extraction tool to be completed by November 2020, but will be delivered to the client with the machine learning algorithm in May 2021.

2 Project Plan

2.1 Task Decomposition

Hardware:

We will need to set up some interface to probe the EM radiation from the processor which will need to be precise and consistent in order to get as clean data as possible. Also we should set up some way to control, monitor and save the processor's executing instructions.

Task:

- **Data collection interface:** Modify existing arduino code and matlab file to be compatible with Nucleo board ISA, for automating data collection
- **Mount**: Design a structure for the probe and processing board to be mounted on for consistent measuring.
- **Single OPs data:** Collect many instances of single opcodes being executed on our processor.
    - Varying Operands Single OPs: data with varying operands
- **Multiple Ops data:** Collect many instances of multiple opcodes being executed on our processor.
    - Varying Operands multiple OPs: data with multiple varying operands
- **Data Interface:** Create an interface between processor chip EM radiation measurements from Matlab and trained ML model for live instruction recognition.

Software:

We need to perform machine-learning experimentation such that we understand what algorithms work best for our data. Including techniques that include some form of recall to understand signals mutating and persisting as they travel through pipeline stages.

Tasks:

- **Classification model for OPs:** We will need a working classification model for opcode prediction when data only includes a single instruction at a time in our processor pipeline.
    - Classification OPs framework subtask: create file that structures classification model, can input data for training and output predictions, but not ready to train effectively yet
    - Classification OPs improvement subtask: polish file such that utilizes the more effective processing and training algorithms
- **Data export automation:** Then we want to use advanced data processing techniques and automate data exportation with multiple instructions in the pipeline.
    - Trained model export subtask: be able to save trained model and load in separate runtime environment to calculate performance analysis

2.2 Risks And Risk Management/Mitigation

For each of the tasks in our project each one comes with an associated risk. Things may not go as well as planned or tools may fail to function. Listed below are some of the risk and risk factors of each of the tasks.

Mounting Hardware: Risk factor 0.2
Risks include not being able to properly mount hardware in order to get a precise and consistent readout.

Single Ops Data: Risk factor 0.35
Risks include not being able to detect an operand vs noise and interference.

Varying Operands Single Ops: Risk factor 0.4
Risks include not being able to properly differentiate between operations detected by the EM antena.

Multiple Ops Data: Risk factor 0.4
Risks include not being able to properly differentiate between operations that are executed one right after another.

Signal Capturing: Risk factor 0.55
Instruments may not be sensitive enough to detect 90% of the opcode and operands. Because of the high risk factor multiple EM antena need to be available for purchase.

Data Conversion: Risk factor 0.2
Risks include conflicts with interfacing sensor output to python.

Filtering Through Machine Learning: Risk factor 0.5
Inability of chosen machine learning technique to properly filter between valid and invalid data. This risk can be mitigated by having an understanding of multiple ML techniques and algorithms.

ML Experimentation: Risk factor 0.1
Risks include the ML algorithm and programming being too difficult to understand and use.

Classification of Ops: Risk factor 0.6
The desired Op classification holds the highest risk because it relies on all other tasks and their associated risk factors. This risk can be mitigated by reducing risk for all other tasks.

Data Export Automation: Risk factor 0.2
Risks include not being able to easily and quickly export data from our EM probe and or the Trained Model.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Observation of Operand and opcode with EM radiation probe with stable and consistent output.

- Successful separation of waveforms for multiple operations in pipeline

Interface between EM radiation probe and software.

- Completely modified arduino and matlab code for Nucleo board compatibility

Collection and identification of many instances of multiple opcodes being executed on the processor.

- Waveforms exported into basic format (Excel) with opcode instruction along with accompanying EM characteristics

Interface between processor chip EM radiation measurements and trained ML model

- Collected data converted into format for ML
- Process of data collection -> ML format can be completed automatically

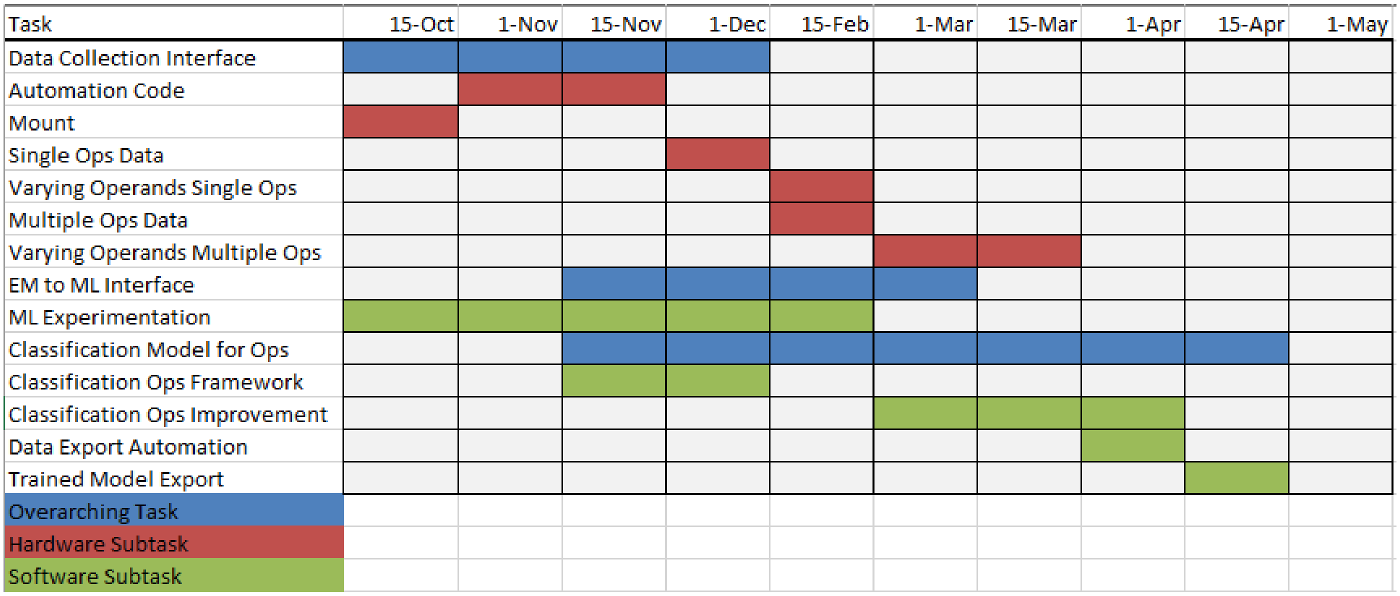
Working classification model for opcode prediction

- 90%+ accuracy in opcode detection

Working classification model for opcode AND operand prediction

- 90%+ accuracy in opcode detection
- 80%+ accuracy in operand detection

2.4 Project Timeline/Schedule*/*-*-



Task	15-Oct	1-Nov	15-Nov	1-Dec	15-Feb	1-Mar	15-Mar	1-Apr	15-Apr	1-May
Data Collection Interface	🟦	🟦	🟦	🟦						
Automation Code		🟥	🟥							
Mount	🟥									
Single Ops Data				🟥						
Varying Operands Single Ops					🟥					
Multiple Ops Data					🟥					
Varying Operands Multiple Ops						🟥	🟥			
EM to ML Interface			🟦	🟦	🟦	🟦				
ML Experimentation	🟩	🟩	🟩	🟩	🟩					
Classification Model for Ops			🟦	🟦	🟦	🟦	🟦	🟦	🟦	
Classification Ops Framework			🟩	🟩						
Classification Ops Improvement						🟩	🟩	🟩		
Data Export Automation								🟩		
Trained Model Export									🟩	
🟦 Overarching Task										
🟥 Hardware Subtask										
🟩 Software Subtask										

2.5 Project Tracking Procedures

To track progress the team will be using GitLab's issue tracking feature. This feature creates a kanban-like board that allows for the assignment of tasks and progress tracking for groups and individuals. Additionally, the team has also chosen to use Slack for IM communications and WebEx teams for voice/video communications.

2.6 Personnel Effort Requirements

Task	Total-Person Hours	Explanation
Data Collection Interface	50	Creation of EM capture and waveform separation method.. Includes validation of signal data after using technique.
Automation Code	10	Automation of data collection technique created, testing of correct functionality.
Mount	5	3D Modelling and Print Time

Single Ops Data	30	Identification of Ops
Varying Operands Single Ops	10	Runthrough and identification of all relevant Ops
Multiple Ops Data	40	Identification of operations in a pipeline, differentiation between consecutive Ops
Varying Operands Multiple Ops	10	Generate data with randomized instruction operands for data variance
EM to ML Interface	40	Set up automation code to connect processed probe readings to trained model for live classification
ML Experimentation	80	Study advanced ML techniques and practice applications on other datasets
Classification Model for Ops	30	Train Models to classify Ops
Classification Ops Framework	30	Design code framework that sets up basic interface to ML packages
Classification Ops Improvement	30	Work on implementing advanced techniques to improve prediction
Data Export Automation	30	Automate data processing on instructions with multiple instructions and varied operands
Trained Model Export	5	ML Package save to file and performance statistical analysis
Total	400	42; answer to life, the universe and everything

2.7 Other Resource Requirements

- STM32H7 Nucleo-144 (MB1364) board
- Power supply, capable of powering above-mentioned microcontroller
- TBPS01 Electromagnetic Probe Clone

- Digital Oscilloscope/Digitizer with a bandwidth of at least 200MHz

2.8 Financial Requirements

- $27 - NUCLEO-H743ZI2 Development Board
- $16 - Semi Rigid Coax Cable (EM Antena part)
- $5 - SMA female to BNC female adapter (EM Antena part)
- $8 - BNC Male to BNC Male Cable (EM Antena part)
- $3 - 2x 4.29mm Cable Ferrites (EM Antena part)
- Up to $70 to purchase another microcontroller
- $129 Total

3 Design

3.1 Previous Work And Literature

Several previous works have investigated electromagnetic side channel information leakage. Works such as *Electromagnetic Emission Measurement of Microprocessor Units* (Maćkowski) have investigated what signals can be measured from the power supply lines. This is slightly different from our project, where we are directly measuring the microprocessor for signals. More closely related to our project is the work *Electromagnetic Side Channel Information Leakage Created by Execution of Series of Instructions in a Computer Processor* (Yilmaz). However we make the distinction that in that work, while signals were being collected, there was no effort to recreate the actions of the processor. This is where our project expands on previous work; we are measuring the electromagnetic radiation from a microprocessor and attempting to recreate the executed assembly by using machine learning.

The task of identifying opcodes and operands as they move through a processor is no easy task. Since our chosen processor has a six-stage pipeline, this means that the EM signals can be a combination of six opcodes/operands. Including the temporal locality of the opcodes in our model will be essential in getting good results. The work *Convolutional neural network-based hidden Markov models for rolling element bearing fault identification* (Wang) is a fairly close example of what we must do, although the input data will be more complex and mixed up.

3.2 Design Thinking

Define - reframe the project in terms of data collection for ML application and a ML classification problem. This splits us nicely into hardware and software projects too.

Ideate (design choices) - design our own probe chosen for cost effectiveness, use time series classification in Tensorflow chosen for its good documentation, chose to use matlab to drive triggers in data collection chosen because it was the example given to us

3.3 Proposed Design

We have proposed to collect data from an STM32H743 nucleo-144 board to observe during operation. The board runs at 480 MHz but can be downclocked below 20 MHz. The flexible clock rate of our board allows us to fit within the 100 MHz constraint set by our current project specifications. Additionally, we are using an EM probe built by the team. The oscilloscope we plan on using is an Agilent DSOX2024A which has an operating bandwidth of 200 MHz. The chosen oscilloscope and probe will allow us to observe the electromagnetic radiation without aliasing the signals. The combination of hardware being used will let us observe an ARM M4/M7 class processor at a frequency of at most 100 MHz and collect the observed data for future use in machine learning.

As for software, we plan on using Google's machine learning framework, Tensorflow, to design a neural network capable of identifying both the OPCode and Operand being executed by our processor. The design of the neural network will change as we refine it to achieve greater accuracy, but Tensorflow will allow us to validate, profile, and benchmark our neural network.

3.4 Technology Considerations

- Oscilloscope
  - There are many oscilloscopes to choose from, but we will have to work with what is in the lab that is available to us.
- EMC Probe
  - Purchasing an EMC probe will cost a couple hundred dollars to purchase. While much higher quality, the price makes this unrealistic.
  - Creating an EMC probe out of semi-rigid coax cable allows for a cheaper alternative, but is not quality controlled and provides no method for calibration.
- Development Board Selection
  - There are a variety of different development boards, but since we are not taking advantage of the different features that they have, any microcontroller such as the Nucleo STM743HZI2 that allows us to configure the clock frequency will work.
- IDE Selection
  - STMCubeIDE is the free IDE that is used to program the Nucleo boards. It is a C or C++ environment that requires creating functions on the simplest level. While it gives us direct access to the board and all set up functions, it is very in depth and requires significant embedded systems knowledge which isn't the goal of this project.
  - The Arduino IDE is another choice that has an open source library which allows it to control ARM based microcontrollers rather than AVR based ones with a variety of prewritten

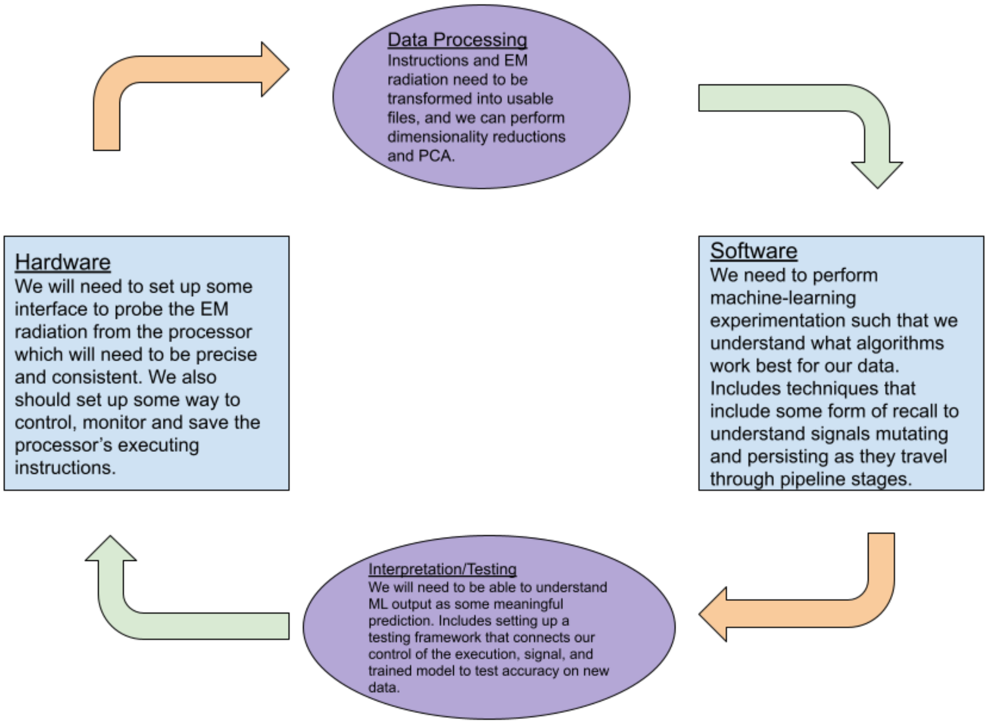
3.5 Design Analysis

We ended up purchasing the Nucleo board for our testing, and controlled it using the Arduino IDE with the homemade EMC probe. The homemade EMC probe was able to capture data from the processor, however the magnitude of the probe was extremely low, and may require an amplifier in the future to capture more accurate data. Other things that were noticed were that other parts of the board were extremely noisy and created undesirable data that could ultimately affect the machine learning. The machine-learning should be able to pick out this data, but it may be important to measure from many different locations along the board to ensure that the various noise sources are considered when the machine learning runs.

3.6 Development Process

Our project's development process is a combination of Waterfall and Agile. The hardware side of our project mainly uses the Waterfall model as tasks need to be sequentially completed in order to work on the next task in line. On the software side of development we use Agile. Agile gives us a flexible and responsive approach to the task at hand as the tasks do not need to be sequentially completed and their importance varies as certain tasks are progressed. Using the Waterfall model to ensure continuous hardware progression and the Agile model to flexibly and reactively respond to problems keeps productivity and progression at a maximum.

3.7 Design Plan



Data Processing

Instructions and EM radiation need to be transformed into usable files, and we can perform dimensionality reductions and PCA.

Software

We need to perform machine-learning experimentation such that we understand what algorithms work best for our data. Includes techniques that include some form of recall to understand signals mutating and persisting as they travel through pipeline stages.

Interpretation/Testing

We will need to be able to understand ML output as some meaningful prediction. Includes setting up a testing framework that connects our control of the execution, signal, and trained model to test accuracy on new data.

Hardware

We will need to set up some interface to probe the EM radiation from the processor which will need to be precise and consistent. We also should set up some way to control, monitor and save the processor's executing instructions.

(1) How did your use cases from Sec 1.5 affect and shape your design plan?

Our use case laid the groundwork for the shape and direction of our design plan.

(2) How are your requirements being handled in the design plan (as well as use cases interpreted within the requirements driven design constraints)?

Because of our design requirements we have chosen to use complex software such as ML in order to achieve the necessary accuracy to meet design requirements.

(3) modules in your design are blocks of functionality of some granularity and abstraction– they compose with each other through interfaces How are these modules tied to requirements and use cases?

Our requirement and use cases come from the software module connected to the testing block since that is where we define the accuracy benchmarks.

4 Testing

4.1 Unit Testing

Software: test a handful of different configurations and classification models

Hardware: test for nop collection, then other opcodes

Acceptance testing for requirements: can we classification with enough accuracy

Unit testing: we will test our automated data collection schemes that make sure communication between matlab, arm board, and oscilloscope are working and data is collected correctly

4.2 Interface Testing

The interface between the nucleo board and the oscilloscope must be tested to ensure that all needed data can be acquired from it. Additionally, we must be able to have the oscilloscope send the data to a Matlab program able to save the EM signals as CSV files.

When gathering the training data for the machine learning model, it is fine that the oscilloscope is separated from the code doing the machine learning. After all, we need to use ISU's GPU cluster to train the model. Once training is completed, though, a goal would be to have the model predict opcodes and operands in real time. This would require the oscilloscope to be able to record EM signals, send them to the computer running the machine learning model, and have the model make predictions. The interface between these two systems will be a point of testing once we have the model trained. If this turns out to be infeasible, then the next best thing would be to run the assembly program, collect the EM signals from the program, and feed that to the model, resulting in a number of predictions.

Relevant interfaces include: Nucleo board, oscilloscope, Matlab, Tensorflow (python)

4.3 Acceptance Testing

The client will be involved in live testing of the design in action. Design requirements are 90% accuracy for opcode and 80% for operand so we will just use 70/30 split for training and verification. Demonstrating to the client that operations on the ARM processor match those observed and processed by our design.

4.4 Results

We are not far enough in the project to post any results from final design testing. We have been able to collect data from the microprocessor, view it in the oscilloscope, and save it in Matlab. Below is an example plot of oscilloscope data. The next steps are to collect data for several opcodes and begin to design a machine learning framework.

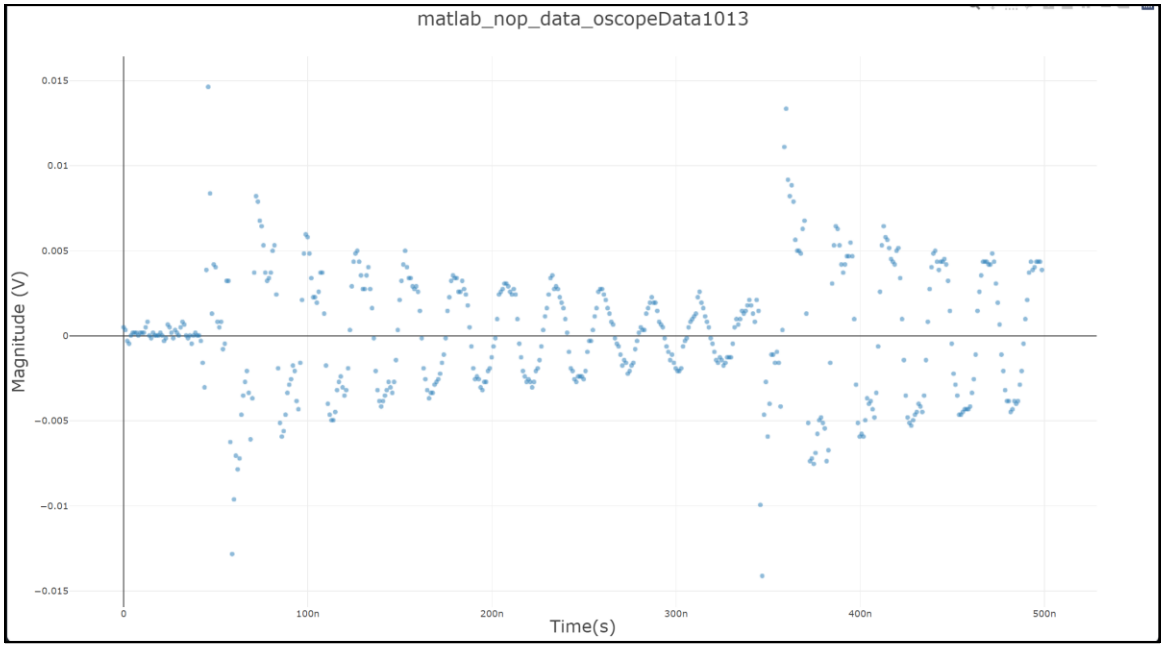


Figure X. Example oscilloscope data of EM radiation caused by NOP opcodes

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).

B. B. Yilmaz, M. Prvulovic and A. Zajić, "Electromagnetic Side Channel Information Leakage Created by Execution of Series of Instructions in a Computer Processor," in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 776-789, 2020, doi: 10.1109/TIFS.2019.2929018.

Andrzej Kwiecień, Michał Maćkowski and Krzysztof Skoroniak Series: Communications in Computer and Information Science, Year: 2012, Volume 291, Page 191

Maćkowski, Michał & Skoroniak, Krzysztof. (2009). Electromagnetic Emission Measurement of Microprocessor Units. 39. 103-110. 10.1007/978-3-642-02671-3_12.

Shuhui Wang, Jiawei Xiang, Yongteng Zhong, Yuqing Zhou, Convolutional neural network-based hidden Markov models for rolling element bearing fault identification, Knowledge-Based Systems, Volume 144, 2018, Pages 65-76, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2017.12.027.

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.