Control Flow Integrity of Dynamic Programming Using an FPGA Board

sdmay21-10

Gregory Wendt - Meeting Scribe
Cole Schumacher - Meeting Facilitator
Nickolas Mitchell - Chief Engineer (FPGA)
Sam Henley - Chief Engineer (Software)
Maxwell Wrangler - Test Engineer
Tristan Duyvejonck - Report Manager

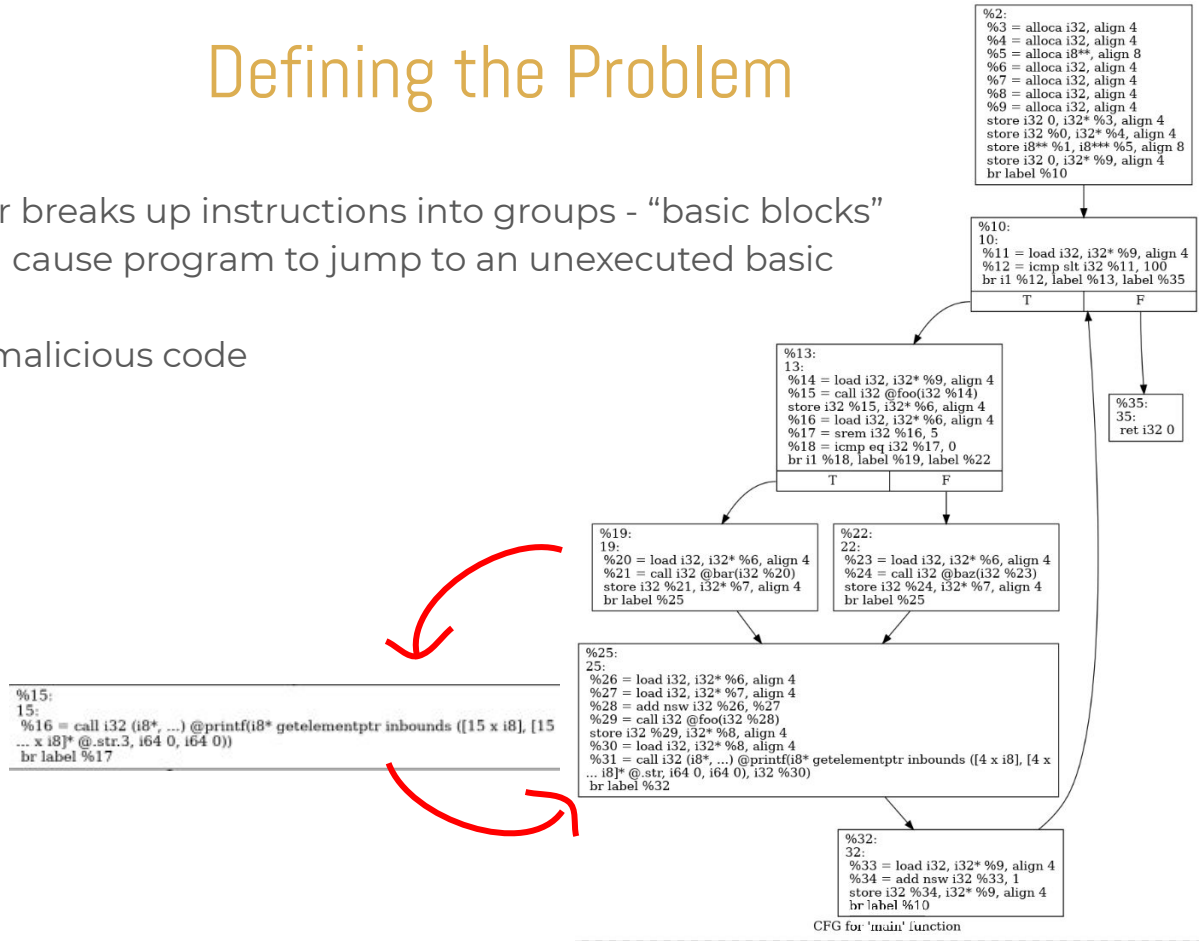
Contact:
sdmay21-10@iastate.edu

Client:
Dr. Akhilesh Tyagi

Faculty Advisors:
Zelong Li
Ananda Biswas

Defining the Problem

- The compiler breaks up instructions into groups - “basic blocks”
- Attacker can cause program to jump to an unexecuted basic block
  - Inject malicious code

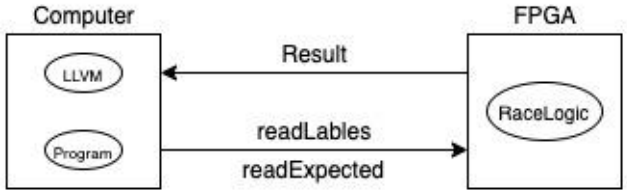


Project Goal

Detect breaches in control-flow integrity using dynamic programming on an FPGA board

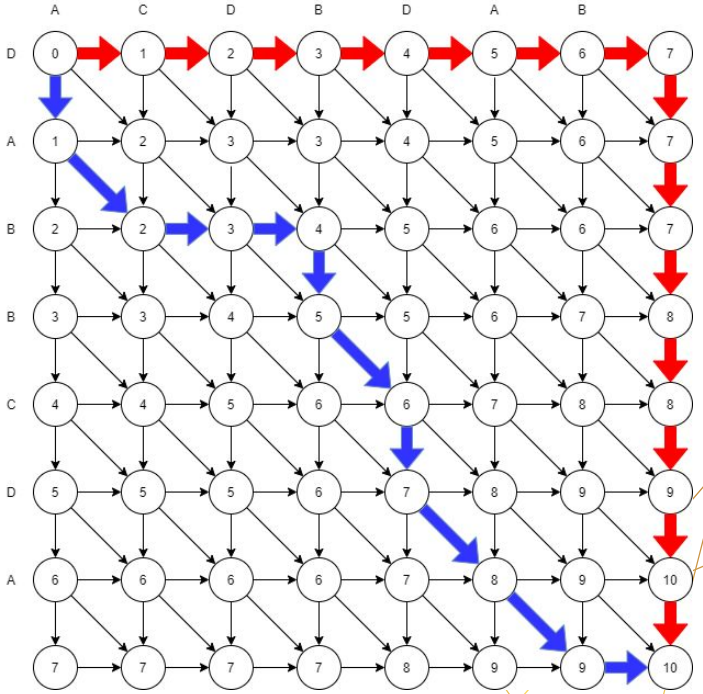
We did this by:

- Creating race logic in a higher level language to test.
- Integrating race logic into hardware (FPGA).
- Set up communicate between computer and FPGA.

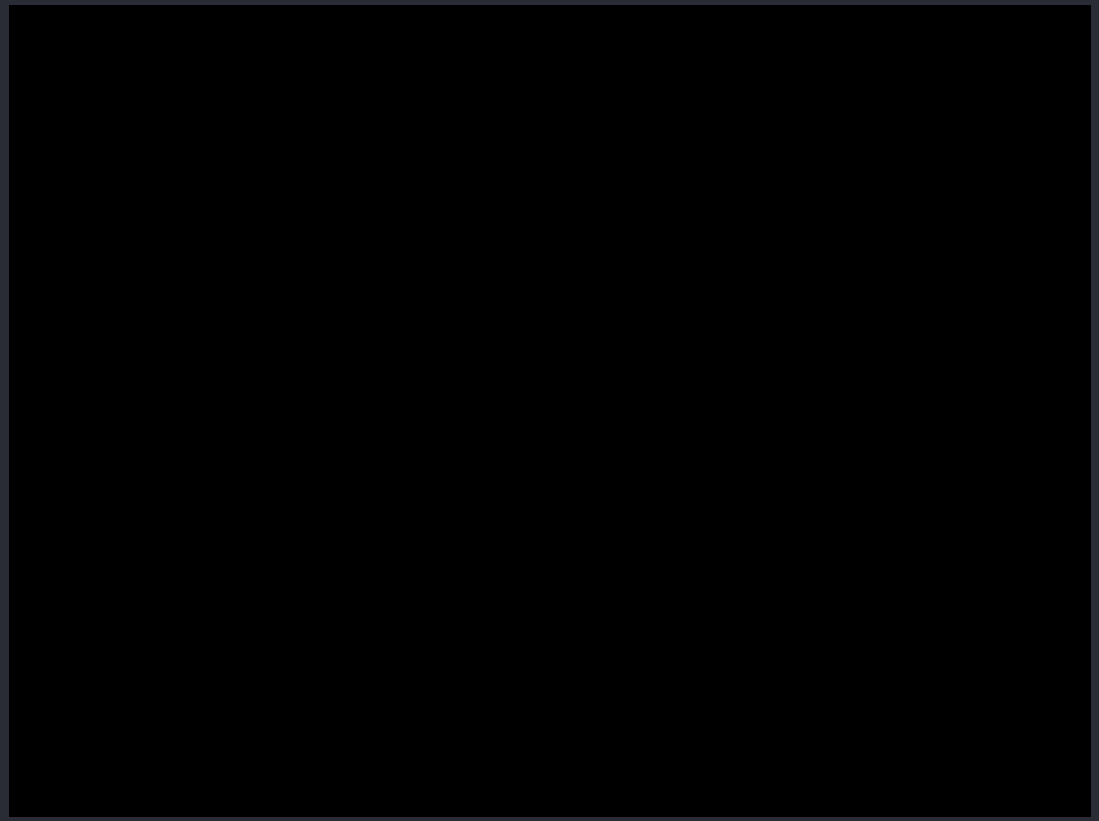


Race Logic

- Goal is to get from the upper left corner to the bottom right in the fewest steps possible.
- Legal moves are one step to the right, one step down or, if the characters at the locations are the same, a step both down and right.
- The blue is one of the most efficient path while the red path is one of the most inefficient.



Race Logic Demo

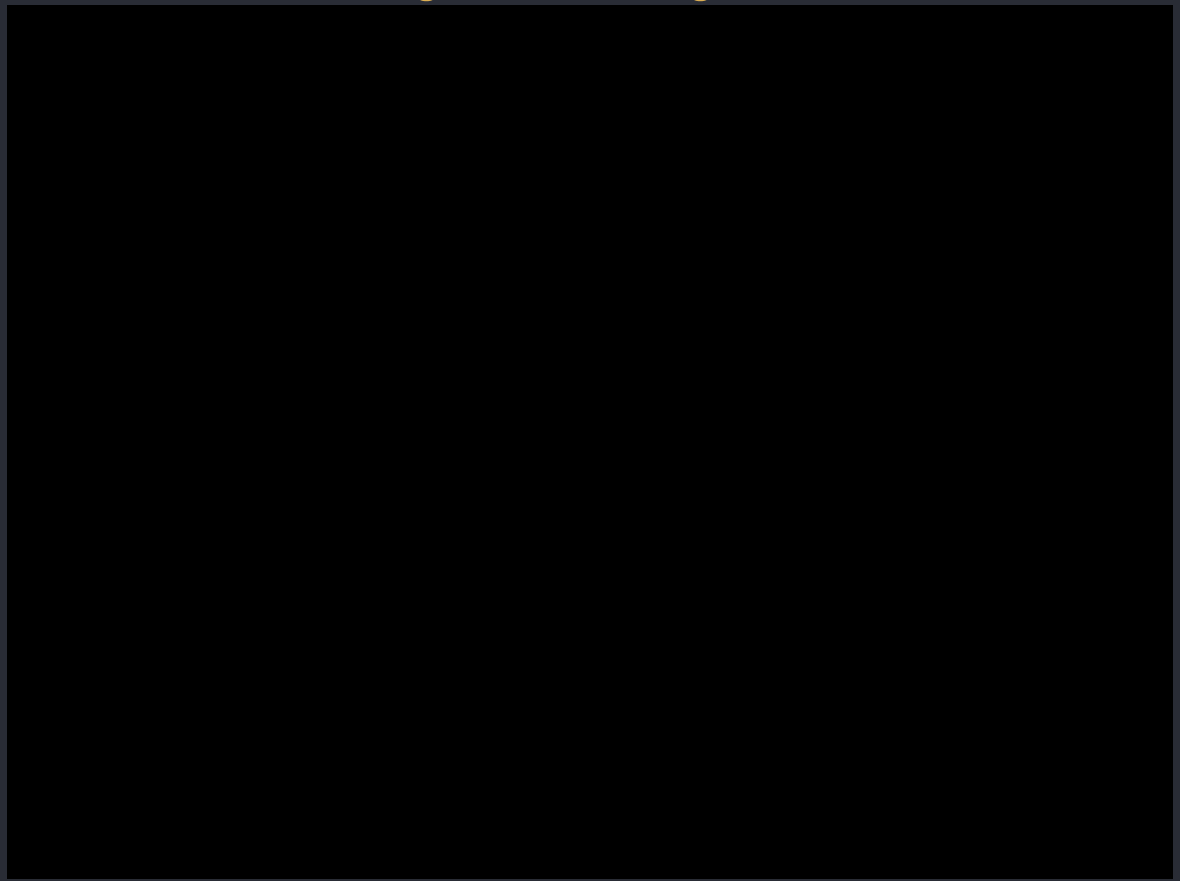


Testing Race Logic

- All types of edge cases were tested
  - Does the algorithm works with different string lengths
  - Does it work for the same string length
  - Does it work if the strings are the same
  - Does it work with mixed characters (letter and numbers)
  - Does it work if a string is empty

- All expected outcomes were manually calculated to compare with

Testing Race Logic Demo



LLVM

LLVM: Compiler and toolchain

Convert source code to Intermediate Representation (IR)

- Portable, high-level assembly language

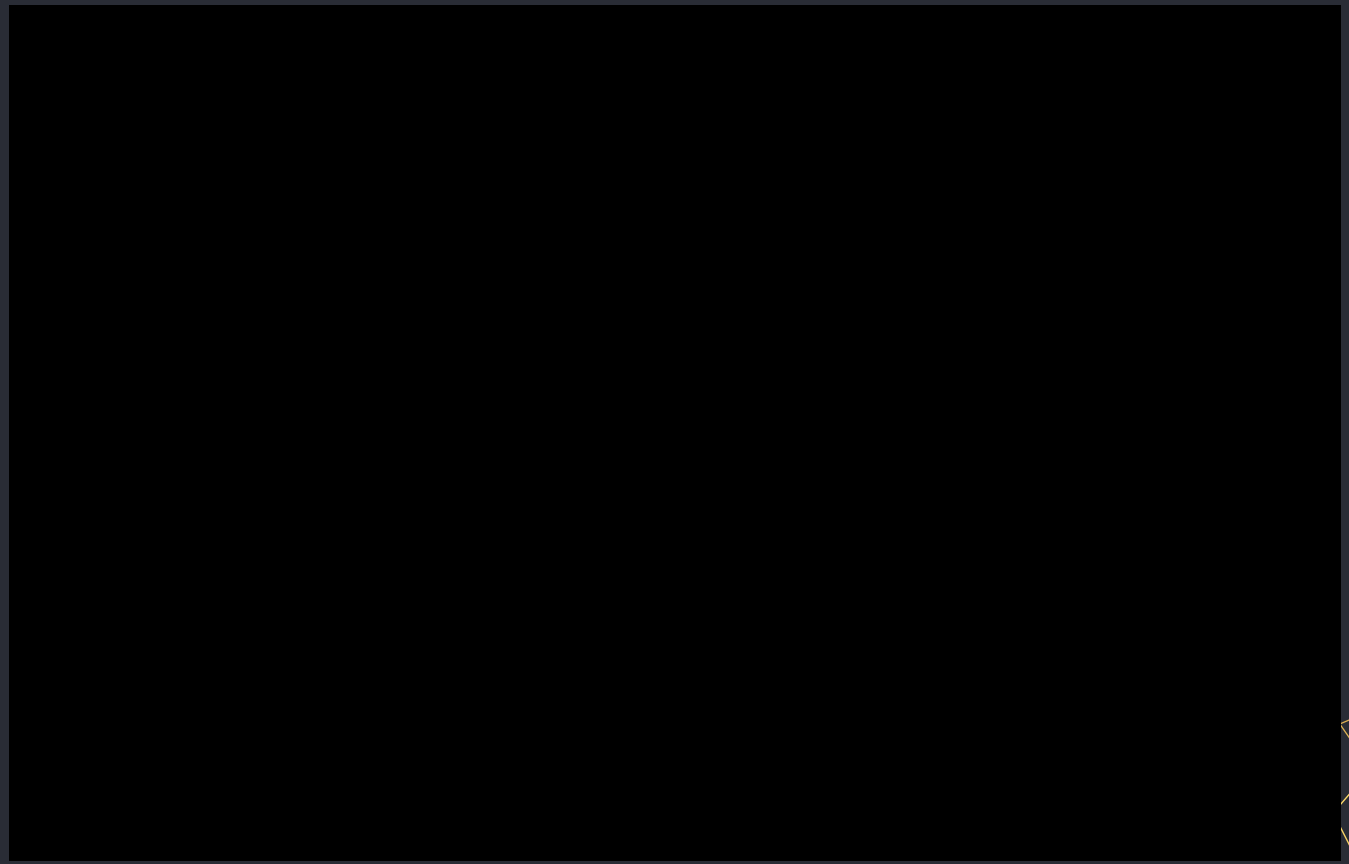
Passes: transformations or optimizations to a program

- 1st Pass: read program and get sequence of basic blocks
- 2nd Pass: inject function at the end of each basic block, generate sequence of basic blocks encountered at runtime

Issues:

- Misunderstanding of our passes late in development

LLVM Demo



FPGA

Process

- Development in Quartus Prime using VHDL
- Originally planned to use Modelsim
- Translated C and Java Race Logic into VHDL
- Changes to Race Logic in VHDL
    - 8-bit inputs
    - Non-Changeable string length

Communication between computer and FPGA board

- Using UART over USB
- Worked on VHDL and Visual Studio C++ solution

Requirements and Constraints

Engineering Requirements

- Create an algorithm to match two sets of strings (race logic).
- One string string generated based on control flow, and compared to a defined string.
- Use a 2-D grid of which the size is determined by the number of basic blocks.
- Algorithm should perform in O(n log(n)) runtime.
    - Current performs at O(nm)

Engineering Constraints

- Use dynamic programming to determine if CFI was lost.
- Use a FPGA board to test the solution.
- Total project cost should not exceed $100.

Lessons Learned

- Better communication with client
  - Better defining the problem earlier on
- Spend less time prototyping
  - Creating a better defined project plan would help us save time on the project
  - Developing our algorithm in both C and Java was not necessary. We could have started in C.
- Commitment to plans
  - Planned to work through winter term but little work was done during this period
  - Planned to use Trello but didn't follow through

Lessons Learned

- Better team communication
  - Insure all teammates are receiving real time notifications
- Identify major hurdles
  - Plan to spend more time on the major hurdles
  - Start major hurdles earlier

Work Done - Greg

- Worked on race logic with Max.
- Wrote tests for race logic and fixed algorithm when they didn't pass
- Helped turn it from Java to C and retested it
- Assisted in research and problem solving for VHDL and LLVM

Work Done - Cole

- Meeting Facilitator
  - Helped coordinate and set up weekly meetings
  - Communicated between team and client
- FPGA Research
  - Met with Nick to discuss different FPGA issues
  - Researched different FPGA solution options
  - Experimented with some FPGA simulation techniques

Work Done - Nick

Work Completed

- Implemented Java Race Logic
- FPGA
  - Translated Java and C Race Logic to VHDL
- Researched UART communication between computer and FPGA

Challenges Worked on

- Setting up the FPGA to work with my personal computer
  - Received help from ETG to resolve this
- Setting up communication link between FPGA and Computer
  - Personal computer doesn't recognize FPGA as a serial device
  - UART code doesn't work well with Windows.

Work Done - Sam

- Development of LLVM Passes
  - Learn what LLVM is / how it works
  - Finding examples of LLVM passes
  - Adapting the sample passes to meet our specifications

Work Done - Max

- Created the pseudocode for the Race Logic Algorithm
- Created a test suite for the Java version of the Race Logic Algorithm
- Helped migrate the Java version of the Algorithm to C
- Created a test framework for the C code for the Race Logic Algorithm
- Helped write tests cases for the C code.

Work Done - Tristan

- Report Manager
  - Managed report submissions and content
  - Edited content before final submission of group assignments
- Helped Sam with LLVM tests
  - Helped with understanding the build process

Thank You!

Contact For Question:
sdmay21-10@iastate.edu