Carnegie Mellon University

FPGA-AMP

Project Proposal

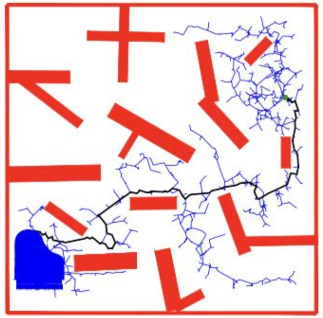
Matt Ngaw, Yufei Shi, Chris Stange
Team C5

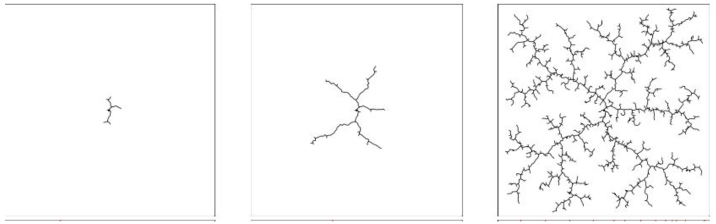
Background

- Recent advances in computing have enabled the development of more sophisticated robots which are increasingly important in our society.

- **Motion planning**: Critical step in the robotics pipeline [2-5].
    - Guides motion of the robot
    - Finds collision-free trajectories from a start position to a goal position
    - Run A* or Dijkstra's on the set of collision-free trajectories in order to find optimal route

- **Our focus**: Rapidly-exploring Random Trees [1]


[1]


[1]

Carnegie Mellon University

Use Case and Motivation

- Problem
  - For complex, latency sensitive robots, RRT is too slow on CPUs and too power-inefficient on GPUs
- Solution
  - Use FPGAs to accelerate motion planning while also consuming less power
- ECE Areas
  - Hardware and Software

Use Case Requirements

- Accuracy
  - RRT is not deterministic so we have to ensure FPGA-AMP generates a collision free path.
  - **>95% accuracy** on this task will mean FPGA-AMP is reliable for real world scenarios.
  - We will also measure how similar the FPGA-AMP solution is to that of our CPU reference solution and a manually generated optimal path. We want to ensure that the path generated by FPGA-AMP is at least as close to optimal as the reference solution.
  - **>95% accuracy** on this task will mean that there is no difference in usability between the reference solution and FPGA-AMP.

- Speedup (Latency, seconds)
  - Research has shown that 1000x speedup over CPUs is attainable. These papers used underpowered CPUs and FPGAs that are more powerful than what we have available [4-5].
  - We hope to achieve **10x speedup** over a CPU implementation.

- Power Efficiency (watts)
  - Power while servicing motion planning query
  - **70% decrease** (130W CPU vs 30W FPGA)
  - The cost of a 130W CPU and a 30W FPGA are comparable (~$350). While a 130W CPU is not a direct competitor in an embedded scenario, it puts an upper limit on the types of systems we want to outperform.

- Energy Efficiency (latency x power, joules)
  - Energy consumed to complete motion planning query
  - If the speedup and power efficiency targets are hit, a **98% decrease** in energy consumption per query should be achieved.

Technical Challenges

- Motion Planning is hard!
  - It is non-trivial to generate motion plans for robots with many degrees of freedom

- Realizing the accelerator microarchitecture in high-level synthesis
  - Easy to copy-paste C code into HLS and get hardware
  - Hard to make the hardware good
    - Meet latency goals
    - Exploit data parallelism

- Data transfer between CPU and FPGA slows us down
  - This is the sequential part of our program. Amdahl’s Law

- System integration

Carnegie Mellon University

Solution Motivation

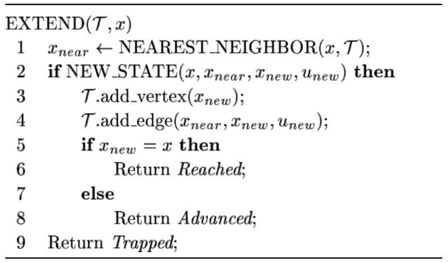
- Why FPGA?
  - Hardware acceleration of an algorithm can be much faster than execution on a CPU
  - Motion planning algorithms (RRT) are highly data-parallel which maps well to FPGA
  - Easy for prototyping, fast iteration cycles
  - GPUs can achieve similar performance enhancements to FPGAs but achieve even worse power figures than the CPU

- Why accelerate motion planning?
  - Motion planning makes up a majority of compute time on a robot [3,4]

- To the best of our knowledge, no open source implementation of RRT for FPGAs exists

Carnegie Mellon University

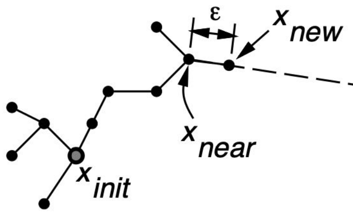
Solution Algorithm (Rapidly-exploring Random Trees)



```
EXTEND($\mathcal{T}, x$)
  $x_{near} \leftarrow$ NEAREST_NEIGHBOR($x, \mathcal{T}$);
  if NEW_STATE($x, x_{near}, x_{new}, u_{new}$) then
      $\mathcal{T}$.add_vertex($x_{new}$);
      $\mathcal{T}$.add_edge($x_{near}, x_{new}, u_{new}$);
      if $x_{new} = x$ then
          Return Reached;
      else
          Return Advanced;
  Return Trapped;
```

[1]

Basic algorithm for the extension of RRT.
Starting at a given node, the function attempts to
generate a new node closer to the target.

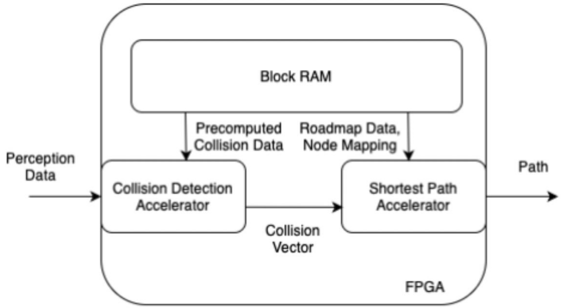


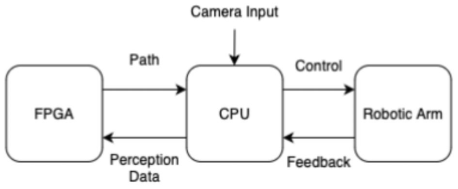
[1]

Graph demonstrating the EXTEND function.

https://en.m.wikipedia.org/wiki/File:Rapidly-exploring_Random_Tree_%28RRT%29_500x373.gif

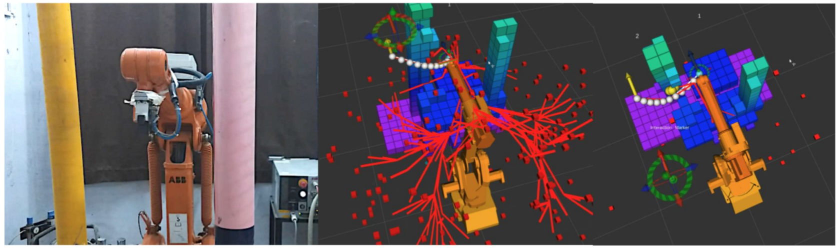
Solution Approach


Block RAM
Precomputed Collision Data
Roadmap Data, Node Mapping
Perception Data
Collision Detection Accelerator
Shortest Path Accelerator
Path
Collision Vector
FPGA



The diagram on the top is for the FPGA-AMP accelerator. This accelerator will be tested in a simulation environment in which the CPU is used to run Linux and feed perception data (generated by a python script) to the FPGA. The generated path will be visualized and metrics will be collected. This is our **MVP**.

The diagram on the on the bottom is the FPGA integrated with a real robotic arm. The CPU will be used to run ROS and will use open source perception, localization and mapping libraries.



https://github.com/indraneelpatil/bottleneck_guided_rrt/blob/master/images/Intelligent%20Exploration.png

- The diagram above shows a robotic arm running RRT to generate a path, allowing it can maneuver around obstacles.
- We plan on having our robotic arm doing **pick and place**. This is an extremely common tasks for robotic arms that are used in a wide range of areas such as manufacturing.
- FPGA-AMP will truly shine when it comes to **dynamic collision detection scenarios**.
  - In static scenarios, only one path must be generated.
  - In dynamic environments, paths must be continually updated.

Testing, Verification, and Metrics

Requirements	Testing	Metrics
Generate Collision Free Paths	Check that the path generated avoids collisions	Avoid collisions on >95% of scenes tested
Generate Optimal Paths	Ensure the delta between the optimal path and the reference solution is similar to the delta between the optimal path and FPGA-AMP	Similar deltas on >95% of scenes tested
Low Latency	FPGA-AMP generates paths significantly faster than the reference solution	10x speedup vs. reference solution
Power Efficient	FPGA-AMP generates paths while being significantly more power efficient than reference solution	70% decrease in power consumption vs. reference solution
Energy Efficient	FPGA-AMP generates paths while being significantly more energy efficient than reference solution	98% decrease in energy consumption per path vs. reference solution

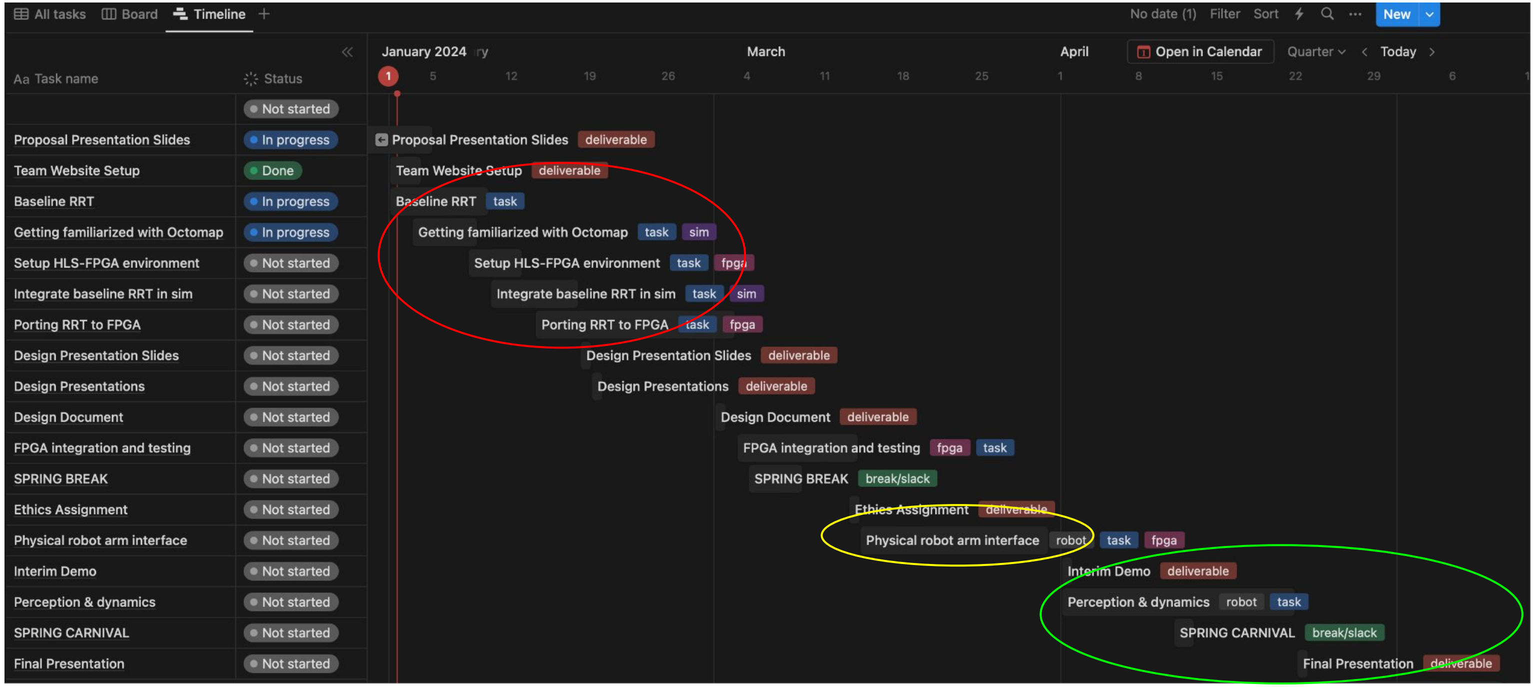
Note: All requirements will be tested in both simulation and with real hardware.

Tasks and Division of Labor

- Baseline RRT (Yufei, Chris)
- **Simulation Environment (Yufei, Chris)**
- HLS-FPGA environment (Matt)
- **Uarch design (Matt, Yufei, Chris)**
- Porting RRT to **HLS (Matt)**
- Optimization (Matt, Yufei, Chris)
- **Perception (Chris)**
- Robotic Arm **Dynamics (Yufei)**
- Robotic Arm & FPGA Integration (Matt)
- Full System **Integration (Matt, Yufei, Chris)**

Carnegie Mellon University

Schedule



Aa Task name	Status
	Not started
Proposal Presentation Slides	In progress
Team Website Setup	Done
Baseline RRT	In progress
Getting familiarized with Octomap	In progress
Setup HLS-FPGA environment	Not started
Integrate baseline RRT in sim	Not started
Porting RRT to FPGA	Not started
Design Presentation Slides	Not started
Design Presentations	Not started
Design Document	Not started
FPGA integration and testing	Not started
SPRING BREAK	Not started
Ethics Assignment	Not started
Physical robot arm interface	Not started
Interim Demo	Not started
Perception & dynamics	Not started
SPRING CARNIVAL	Not started
Final Presentation	Not started

Proposal Presentation Slides deliverable
Team Website Setup deliverable
Baseline RRT task
Getting familiarized with Octomap task sim
Setup HLS-FPGA environment task fpga
Integrate baseline RRT in sim task sim
Porting RRT to FPGA task fpga
Design Presentation Slides deliverable
Design Presentations deliverable
Design Document deliverable
FPGA integration and testing fpga task
SPRING BREAK break/slack
Ethics Assignment deliverable
Physical robot arm interface robot task fpga
Interim Demo deliverable
Perception & dynamics robot task
SPRING CARNIVAL break/slack
Final Presentation deliverable

References

[1] LaValle, S., et al. “Rapidly-exploring random trees: Progress and prospects,'' in *Algorithmic and computational robotics: new directions*, vol. 5, pp. 293–308, 2001.

[2] Liu, S., et al, “Robotic computing on fpgas,'' *Synthesis Lectures on Computer Architecture*, vol. 16, no. 1, pp. 1-218, 2021.

[3] Wan, Z., et al, “Robotic Computing on FPGAs: Current Progress, Research Challenges, and Opportunities,'' in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2022, pp. 291-295.

[4] Murray, S., et al, “The microarchitecture of a real-time robot motion planning accelerator," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2016, pp. 1–12.

[5] Murray, S., et al, “A programmable architecture for robot motion planning acceleration,'' in *2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2019, pp. 185–188.

Carnegie Mellon University