CS310 Project Specification: Using ML to optimise structured mesh-based solvers on FPGAs

Matthew MacDermott

October 2025

1 Problem Statement

Structured mesh stencil computations are widely used to solve problems involving Partial Differential Equations. FPGAs offer an attractive platform for accelerating these computations, providing comparable performance to CPUs and GPUs while consuming significantly less power. However, achieving optimal performance requires tuning a large number of design parameters, resulting in a vast and complex design space that is difficult to explore manually.

Recent work has introduced an automatic code generation framework that uses the OPS (Oxford Parallel library for Structured mesh solvers) domain-specific language to generate optimized High-Level Synthesis (HLS) code for FPGAs. This framework translates high-level stencil descriptions directly into efficient FPGA implementations, applying optimizations such as window buffer chaining and on chip loopback pipelines to achieve high throughput and energy efficiency.

While this significantly reduces development effort, the unoptimised design space remains large and challenging to navigate. It is proposed that machine learning can be used to explore this space and predict optimal FPGA parameter configurations more effectively than current methods, resulting in improved performance.

2 Objectives

2.1 Essentials

The following objectives outline the key steps required to successfully complete this project:

1. **Research High-Level Synthesis (HLS):** Gain a thorough understanding of HLS concepts, with a focus on using Vitis, which will be extensively used throughout the project.
2. **Investigate the OPS Domain-Specific Language (OPS-DSL):** Study how stencil computations are expressed using OPS and how these are mapped to FPGA implementations.

3. **Explore suitable Machine Learning (ML) techniques:** Identify and evaluate ML models best suited for predicting optimal design parameters within an FPGA design space.
4. **Investigate optimal data encoding:** Explore methods for encoding stencil and report data as input to the ML model.
5. **Setup environment:** Configure Vitis and OPS-DSL using existing work [1] to enable automatic translation of OPS stencil code into C++ Vitis HLS code.
6. **Develop an ML-based optimisation framework:** Implement an ML model capable of selecting optimisation parameters (e.g. unrolling, tiling, and pipeline depth) for structured mesh stencil computations.
7. **Simulate and evaluate parameter selections:** Use FPGA simulation tools to estimate performance of the generated designs and test the ML model's predictions.
8. **Iteratively refine the model:** Develop an automated pipeline that feeds simulation results back into the model, enabling it to learn from each iteration and improve its prediction accuracy over time.
9. **Parallelise the training pipeline:** The high build time of the HLS project, both in hardware and emulation modes, significantly slows down model training. To address this, parallel training will be explored to improve the rate of data generation and improve overall training efficiency.
10. **Benchmark on FPGA hardware:** Run the optimal simulated designs on real FPGA hardware and compare performance against static and heuristic-based optimisation techniques [1].

2.2 Extensions

Depending on the workload and progress through the essential objectives, the following extensions may be pursued to enhance the project outcomes. While not required for successful completion, they would provide valuable additional insights and improvements.

11. **Expand the ML optimisation scope:** Enable the machine learning model to predict a wider range of optimisation parameters (e.g. array partitioning and loop fusion) beyond the initial set.
12. **Assess and apply optimal data encodings:** Investigate how different encodings of stencil and report data affect the model's ability to learn performance patterns. Select the encodings that give the best performance.
13. **Optimise ML model for performance and efficiency:** Extend the ML model to optimise for both performance and power efficiency. Evaluate the trade-off compared to optimising solely for performance.
14. **Architecture-specific training:** Develop and train separate models tailored to specific FPGA architectures.

3 Methods & Methodology

3.1 Background reading

The project will initially require a lot of background reading of available articles and papers regarding using machine learning for stencil computations, FPGA optimisations and the Vitis HLS pipeline.

3.2 Experimentation with Existing System

Set up and familiarise with the current system by gaining access to the Owl DCS environment. Begin using Vitis and OPS-DSL to develop optimised stencil computations. Establish and validate the automatic code generation workflow from OPS-DSL to Vitis C++ for FPGA deployment [1].

3.3 ML Model Creation

Develop a reinforcement learning model to integrate with the existing system. The model will take as input an OPS-DSL stencil, a set of possible optimisations and hardware details. It will decide which optimisations to apply along with their associated parameters. For example, the model may choose to implement loop unrolling and select an optimal unroll factor of 4. Initially, to simplify the problem, the range of values for each optimisation parameter will be pre-defined (e.g. unroll factor constrained between 1 and 32).

3.4 ML Model Training

Training will begin with randomly selected optimisation parameters and values. This is both for simplicity at this stage and allowing the model to explore a wide and unbiased range of the design space. These configurations will be simulated using the FPGA emulation environment provided by the Owl DCS system. The results of these simulations will be fed back into the model to iteratively improve its ability to select optimal parameters.

3.5 Evaluation Methodology

For each FPGA configuration and set of optimisations, both throughput and energy consumption will be measured. A key advantage of FPGAs over other hardware, such as GPUs, is their lower power consumption for comparable performance.

Results will be output via the terminal in a text-based format. For clearer presentation, visualisations will be created using Python's Matplotlib library [4]. To reduce the impact of variability and measurement noise, all experiments will be run three times and averaged. Tests will also be performed on both unoptimised code and code optimised using static heuristics, allowing direct comparison of the benefits provided by ML-guided optimisation.

3.6 Methodology

The project will follow an agile development methodology, building the solution incrementally through successive iterations. Initial cycles will focus on setting up the system and integrating a basic ML model. Subsequent iterations will expand the scope to incorporate

advanced optimisation strategies and different data encodings, gradually improving the ML model's predictive accuracy.

This iterative approach is well-suited to the experimental nature of FPGA stencil optimisation. It provides the flexibility to adapt objectives and priorities based on simulation results and hardware benchmarking, ensuring progress remains aligned with project goals and manageable within the available time frame. While agile typically emphasises teamwork and collaborative development, this project is undertaken individually, so the methodology is applied primarily to iteration and incremental improvement rather than team-based practices.

4 Timetable

Date	Tasks
T1 W1-2	Writing and submission of the Project Specification.
T1 W3-4	Research into Vitis HLS, OPS-DSL and ML models.
T1 W4	Setup OPS-DSL and Vitus environment and show manual parameter tuning works. Test out simulation framework and analyse the report produced.
T1 W5-6	Implement a basic ML model that outputs optimisation parameters within specified range.
T1 W7	Research how to encode the stencil and report data for use with the model.
T1 W7-9	Create a reinforcement learning loop that uses the report from previous optimisations as training data.
T1 W8	CS325 Compiler Design buffer time. Parallelise the RL loop to increase training rate.
T1 W9	CS342 Machine Learning buffer time. Train the ML model.
T1 W10-11	Run the ML chosen optimisations on FPGAs and evaluate the results against other methods.
Christmas Holidays	Writing of the progress report and CS345 Sensor Networks coursework buffer time.
T2 W1	Finalise and submission of the progress report.
T2 W2-3	Refine the ML model further to add extra optimisation parameters.
T2 W4-5	Develop specific models for other FPGA architectures, utilising device properties.
T2 W5	Term 2 coursework buffer.
T2 W6	Experiment and evaluate with other ML architectures.
T2 W7	Extend the ML model to optimise for both performance and power efficiency.
T2 W8	Begin writing the final report plan.
T2 W9	Finalisation and submission of the final report plan. Begin writing the final report.
T2 W10	Continue writing the final report and write the project management report.
Easter Holidays	Exam revision and finalising the final report.
T3	Continued exam revision and prep for Viva.

The timetable outlines the anticipated workflow for the project, though it should be considered an estimate due to the evolving nature of the work. It includes buffer time to account for unforeseen circumstances, as discussed in the risks section, as well as for coursework deadlines. Progress will be monitored through weekly meetings with the project supervisor.

5 Resources & Risks

5.1 Resources

- **Vitis** [2] - An open-source FPGA development platform by Xilinx, used for HLS and hardware acceleration of C++ code.
- **OPS-DSL** [5] - An open source domain specific language for expressing structured mesh stencil computations, enabling automated code generation for CPUs, GPUs, and FPGAs.
- **Personal Laptop** - For code development and testing.
- **FPGA** - Field-Programmable Gate Array hardware used to evaluate stencil computations with the optimised parameters.
- **DCS Batch Compute System** - Department provided environment used for training machine learning models and other intensive computations.
- **DCS Owl Node** - Dedicated compute node within the DCS system configured for running Vitis HLS.
- **HACC Cluster** [8] - Computing cluster providing multiple FPGAs for hardware testing.
- **Git** - Version control system used to track code changes.
- **GitHub** [6] - Cloud-based platform for hosting Git repositories and enabling multi-device development.
- **Python with Matplotlib** [4] - Python library for creating data visualisations.

5.2 Risks

Risk	**Mitigation**
Personal laptop breaking or getting lost	There will be a GitHub repo used to remotely store my work and frequent syncs to minimize the amount of uncommitted work which could be lost. The Department of Computer Science lab machines will be able to support my project.
Unexpected delays from illness	The added buffer time in the timetable allows for unforeseen delays caused by illness or similar issues.
Losing project documents	All project documents will be written using Overleaf, which is a cloud-based LaTeX editor. This allows access to documents from any device with an internet connection. In case of loss of data or lack of internet connection, all documents will be synced weekly to GitHub, and in turn, my local laptop.
Slow RL model training due to long HLS build times, limiting progress	The RL model training will be parallelised to increase the amount of data to train the model. Genetic algorithms could be explored to combine learnings from multiple runs.

6 Legal, Social, Ethical and Professional Issues & Considerations

No such risks have been identified.

References

[1] B. Thileepan, S.A. Fahmy and G.R. Mudalige. (2025). Automatic Code-Generation for Accelerating Structured-Mesh-Based Explicit Numerical Solvers on FPGAs. To appear in the International Conference on Parallel Architectures and Compilation Techniques (PACT '25) Nov 03–06, 2025, Irvine, California, USA. ACM, New York, NY, USA.

[2] [Online] Available at: `https://www.amd.com/en/products/software/adaptive-socs-and-fpgas/vitis.html` (Accessed: 13 October 2025).

[3] Q. Sun, Y. Liu, H. Yang, Z. Jiang, Z. Luan and D. Qian, "StencilMART: Predicting Optimization Selection for Stencil Computations across GPUs", 2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Lyon, France, 2022, pp. 875-885, doi: 10.1109/IPDPS53621.2022.00090.

[4] (2024). *Matplotlib - Visualization with Python* [Online]. Available at: `https://matplotlib.org/` (Accessed: 13 October 2025).

[5] *OP DSL Homepage* [Online]. Available at: `https://op-dsl.github.io/` (Accessed: 13 October 2025).

[6] *GitHub: Let's build from here* [Online]. Available at: `https://github.com/` (Accessed: 13 October 2025).

[7] Víctor Martínez, Fabrice Dupros, Márcio Castro, Philippe Navaux, "Performance Improvement of Stencil Computations for Multi-core Architectures based on Machine Learning", Procedia Computer Science, Volume 108, 2017, pp. 305-314, ISSN 1877-0509, doi: 10.1016/j.procs.2017.05.164. (`https://www.sciencedirect.com/science/article/pii/S1877050917307408`)

[8] *Heterogeneous Accelerated Compute Clusters* [Online]. HACC Resources. Available at: `https://www.amd-haccs.io/` (Accessed: 13 October 2025).