3 Project Plan

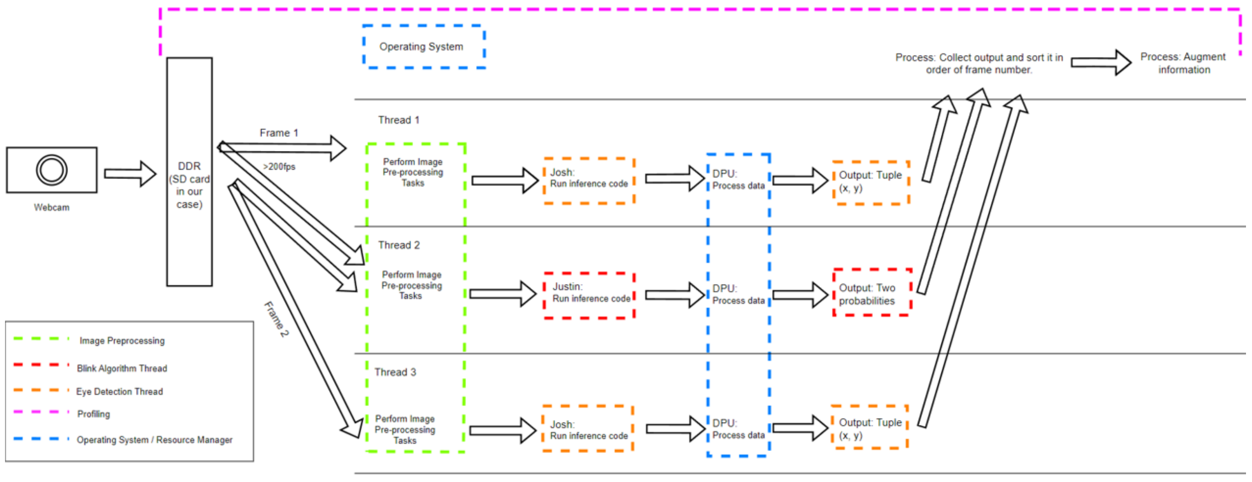
3.1 Project Management/Tracking Procedures

We've chose an Agile project management approach to maximize flexibility and adaptability as we work towards achieving our goal of achieving a throughput of <5ms (200 fps). This methodology allows us to continually reassess our implementations and decisions, ensuring we stay aligned with evolving project requirements.

To facilitate seamless collaboration among team members, we've leveraged platforms like GitHub and Trello. GitHub serves as our primary repository, where our client also has access, enabling them to track our progress in real-time. The source codes and documentation are primary uploaded to our GitHub repository. Meanwhile, Trello enables us to organize tasks, issues, and milestones, ensuring every team member remains on track and focused on key deliverables.

By adopting Agile methodologies and utilizing these collaborative tools, we're not only ensuring transparency and accountability within our team but also fostering a dynamic environment where innovation and iteration thrive.

3.2 Task Decomposition


Operating System
Process: Collect output and sort it in order of frame number.
Process: Augment information
Webcam
DDR (SD card in our case)
Frame 1
>200fps
Frame 2
Thread 1
Perform Image Pre-processing Tasks
Josh: Run inference code
DPU: Process data
Output: Tuple (x, y)
Thread 2
Perform Image Pre-processing Tasks
Justin: Run inference code
DPU: Process data
Output: Two probabilities
Thread 3
Perform Image Pre-processing Tasks
Josh: Run inference code
DPU: Process data
Output: Tuple (x, y)
Image Preprocessing
Blink Algorithm Thread
Eye Detection Thread
Profiling
Operating System / Resource Manager

Design and Implementation

We aim to achieve a throughput of <5ms (200 fps). The diagram shown above is a simplified version of our program design. Our ideal plan is to retrieve each frame from the SD card that our client gave us as input to our multithreading program. The first frame will be fed into Thread 1 and Thread 2. Both threads are making different inferences. The second frame will be fed into Thread 3. The second frame will also feed into Thread 2 once it finishes inferring the first frame. Eventually, three threads will run concurrently. The idea of using three threads is because the Kria KV260 FPGA board has four DDR4 memory. Each of the DDR4 memory is 1 GB in size. Each thread will be assigned one DDR4 memory. The last one, DDR4 memory, will be assigned to the CPU.

In Thread 1, the input frame will first be pre-processed using the image semantic segmentation technique to remove any reflection or imperfection in the frame and output to eye tracking inference code. Then, the eye tracking inference code will infer the pre-processed frame by utilizing DPU and output frame with the pupil coordinate (x-axis and y-axis). The output pupil coordinate frame will then stake with the output frame with the result according to the frame number. The final output will be stored in an array according to the frame number.

In Thread 2, the input frame will first be pre-processed by cropping the region of interest to reduce data size and output to blink detection inference code. Then, the blink detection inference code will infer the pre-processed frame by utilizing DPU and output frame with the result (blink or no blink).

In Thread 3, the input frame will first be pre-processed using the image semantic segmentation technique to remove any reflection or imperfection in the frame and output to eye tracking inference code. Then, the eye tracking inference code will infer the pre-processed frame by utilizing DPU and output the pupil coordinate (x-axis and y-axis). The output pupil coordinate frame will then stake with the output frame with the result according to the frame number. The final output will be stored in an array according to the frame number.

Components

Pre-processing

- This component will be the first component in every thread.
- It will pre-process all the input frames for other inference codes to speed up the process of inferring.
- This component uses semantic segmentation to remove reflection and imperfection in the content of the frame to ensure the result is accurate.

Eye-Tracking

- This component will be used in two threads as the inferring duration is longer than blink detection.
- This component will utilize the DPU to speed up the inferring process.

Blink Detection

- This component will only be used in one thread as the inferring process is faster than eye-tracking due to minimum data need to be inferred.
- This component will utilize the DPU to speed up the inferring process.

Deep Learning Processing Unit (DPU)

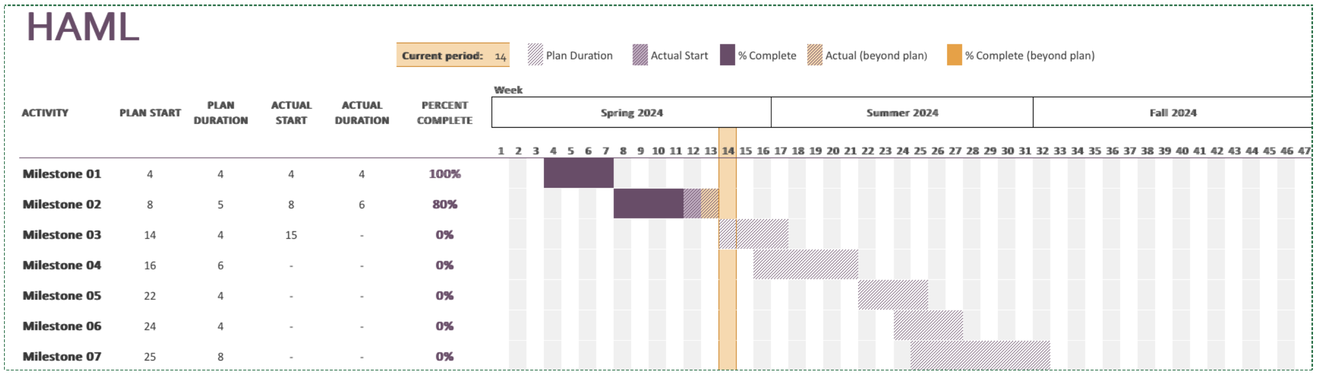
- The DPU model is DPUCZDX8G_ISA1_B4096.
- It is a machine learning accelerator in FPGA.
- It is a specialized hardware accelerator designed specifically for deep learning tasks, aiming to improve performance and efficiency compared to traditional CPUs or GPUs.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Key Milestones	Evaluation Criteria
1. Understand previous team's code.	Fully understand previous team's code.
2. Combine image pre-processing, blink detection algorithm, eye-tracking algorithm, into one program (serially).	Fully implement and test each algorithm for accuracy.
3. Implement parallelism into program.	Achieve throughput of <5ms (200 fps).
4. Implement semantic segmentation for image pre-processing and retrain model.	Achieve model accuracy of top-5 95% (while maintaining <5ms throughput).
5. Implement eye tracking algorithm with the pre-processed image	Achieve the accuracy of output from 86% to 92%.

6. Implement blink detection algorithm (new) with the pre-processed image	Achieve the accuracy of output 99%.
7. Run three threads concurrently by initializing each thread to a memory	Successfully run three threads concurrently without fighting for memory.

3.4 Project Timeline/Schedule



HAML

Current period: 14

▨ Plan Duration ▨ Actual Start ■ % Complete ▨ Actual (beyond plan) ■ % Complete (beyond plan)

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Milestone 01	4	4	4	4	100%
Milestone 02	8	5	8	6	80%
Milestone 03	14	4	15	-	0%
Milestone 04	16	6	-	-	0%
Milestone 05	22	4	-	-	0%
Milestone 06	24	4	-	-	0%
Milestone 07	25	8	-	-	0%

3.5 Risks and Risk Management/Mitigation

Risk	Probability	Mitigation Strategies
Not being able to complete project in time	10%	• Regularly review and adjust project timelines during sprint retrospectives. • Implement a robust risk management plan to identify potential delays early on. • Break down project tasks into smaller, manageable chunks to track progress more effectively. • Allocate additional resources or adjust team priorities as needed to meet deadlines.
Combining multiple algorithms into one program can increase the complexity of the software	20%	• Conduct thorough code reviews to ensure clarity, efficiency, and maintainability of the integrated algorithms. • Implement modular design principles to encapsulate individual algorithms, making the codebase more manageable. • Utilize comprehensive testing methodologies to validate the integration of algorithms and identify any potential conflicts or performance bottlenecks.
Implement parallelism into program could lead to synchronization overhead	30%	• Employ effective parallel programming paradigms such as

		task-based parallelism or data parallelism to minimize synchronization overhead. • Utilize synchronization primitives like locks, semaphores, or atomic operations judiciously to avoid contention and bottlenecks. • Profile and optimize critical sections of code to reduce serialization and maximize parallel execution.
Image semantic segmentation could take longer time	10%	• Continuously optimize the machine learning algorithms used for semantic segmentation to improve inference speed without compromising accuracy. Techniques such as model pruning, quantization, and architecture optimization can be explored. • Apply data augmentation techniques and preprocessing steps to the input images to reduce computational complexity without sacrificing model performance. Techniques like image resizing, cropping, and normalization can streamline inference. • Investigate model compression techniques such as knowledge distillation or weight pruning to reduce the computational requirements of the segmentation model while preserving its accuracy. This can lead to faster inference times on resource-constrained hardware.

3.6 Personnel Effort Requirements

Team member	Task	Subtask	Description	Estimated hours
Jonathan Tan	DPU Management, main function	Implement DPU sharing	Because there is only one DPU on board and multiple algorithms share the resource, proper DPU sharing mechanism need to be implemented.	10

		Coordinate dataflow	Coordinate the input and output requirements of different algorithms	5
		Testing	Testing and debugging.	30
Josh Czarniak	Eye tracking algorithm	Modify previous team's eye tracking code	Previous team's eye tracking code relies the RPU, we need to remove that portion of the code and ensure that it still runs.	10
		Manage memory and limit to 1GB per thread	Due to the limited DDR memory on board, each eye tracking thread is only allowed 1GB of memory.	10
		Testing	Testing and debugging.	30
Justin Wenzel	Blink detection algorithm	Write the blink algorithm	Using the blink detection model, write the program that implements the blink detection model.	10
		Manage memory and limit to 1GB per thread	Due to the limited DDR memory on board, each eye tracking thread is only allowed 1GB of memory.	10
		Testing	Testing and debugging.	30
Kai Heng Gan	OpenCV image preprocessing	Implement image semantic segmentation machine learning algorithm	Using semantic segmentation ML algorithm could remove the glare in the image and increasing the accuracy of the output.	10
		Write a program for image pre-processing	This program will take the input fed by the main function and process the image by running the model trained from the semantic segmentation ML algorithm and output the result for other algorithms to infer.	10
		Testing	Testing and debugging the program to avoid errors occurred in the main function.	30
Santiago Campoverde	Profiling	Recompile Petalinux to include VART profiling tools	In order to use VART's profiling tools, we will need to recompile the operating system (Petalinux) to include the tools.	20
		Create script to analyse generated csv data.	Because we will run many tests, a script is to be created to analyse multiple tests at once	5

3.7 Other Resource Requirements

Hardware Resources -

- **Xilinx Kria Evaluation Board** – For development and exectuing program, utlizing built in DPU for model inferences.

- **ETG Provided Development Computer** – Provides a native Linux OS for the team to develop and test programs, including installation of other Xilinx development tools. Allows team to SSH into board from remote locations.

Software Resources -

- **TensorFlow** – An open-source machine learning library developed by Google, allowing developers to easily build and deploy machine learning models/applications.
- **Docker** – An opern-source platform creating an environment that easily allows developers to develop, share, and run applications, by packages everything needed to run the software within a unti called a container.
- **Xiling Vitis** – A software platform that assists in the development and launch of embedded software on Xilinx's different hardwares, including FPGAs in this project.
- **OpenCV** – An open-source computer vision and machine learning library. Used for many different cases but assits the project through image preprocessing.
- Python – A favored language of choice for its vast support and libraries to assist in machine learning and development.
- **C/C++** - Used within the embedded system for implementing low level functionality and interfacing of the and between different hardware components.

Collaborative Tools/Documentations

- **Version Control System** – Using Git with the platform GitHub for code sharing among group members.
- **Project Management** – Using Trello for task management, and progress checks for group members.
- **Microsoft Teams/Telegram** - Enables collborative communication channels group and client communication.
- **Technical Documentation** – Utilizing different technical documents to further understanding of hardware and software tools (DPU, TensorFlow, Docker, OpenCV, etc.).
- **Group Documentation** – Utilizing specific documentation created by previous groups during development and testing, to assits with previous project understanding and provided code base.