PROJECT USE CASE SCENARIOS #4 – FPGA BASED ML EDGE

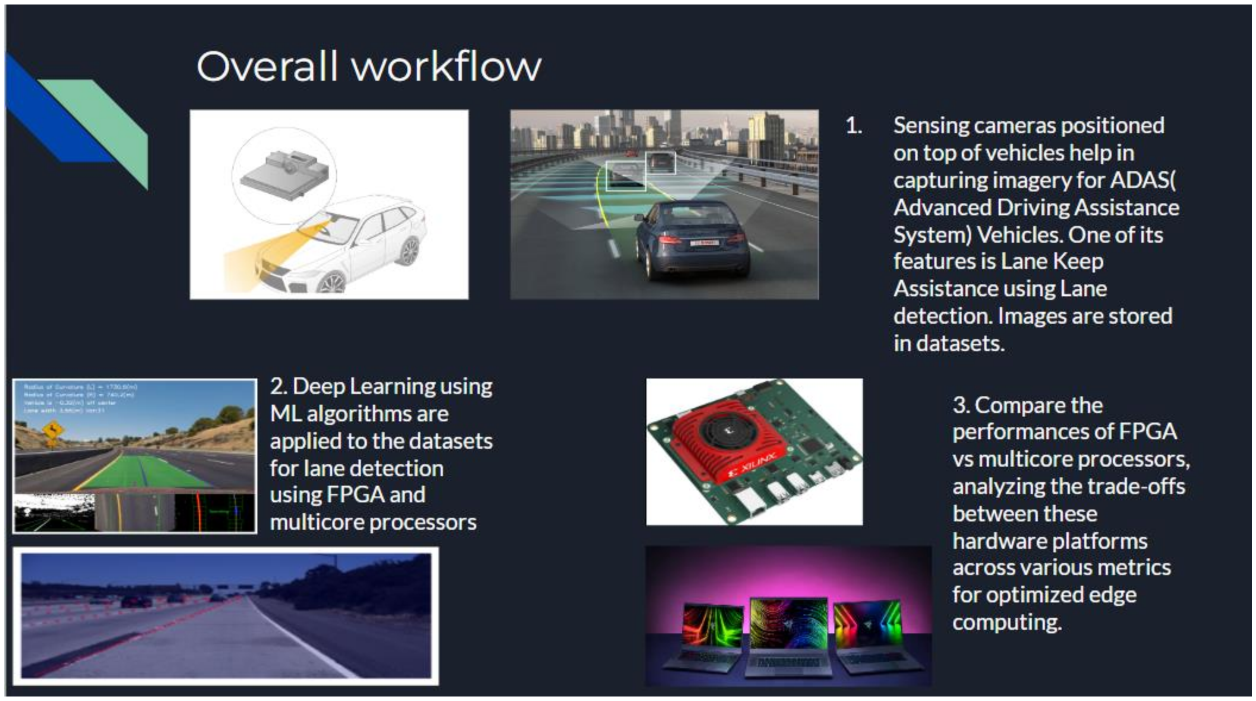
Names of Team Members, Team Member Role, UCI email addresses:

1. Vishwanath Singh (vishws1@uci.edu), Manish Sudumbreaker (msudumbr@uci.edu)

UPDATED VISUAL REPRESENTATION OF PROJECT

Include your updated visual from the initial project proposal here (should be your own interpretation of the prototype). At this point, no more hand-drawn images. No AI Generated images.

Overall Workflow of the project using FPGA for lane detection



Overall workflow

1. Sensing cameras positioned on top of vehicles help in capturing imagery for ADAS(Advanced Driving Assistance System) Vehicles. One of its features is Lane Keep Assistance using Lane detection. Images are stored in datasets.

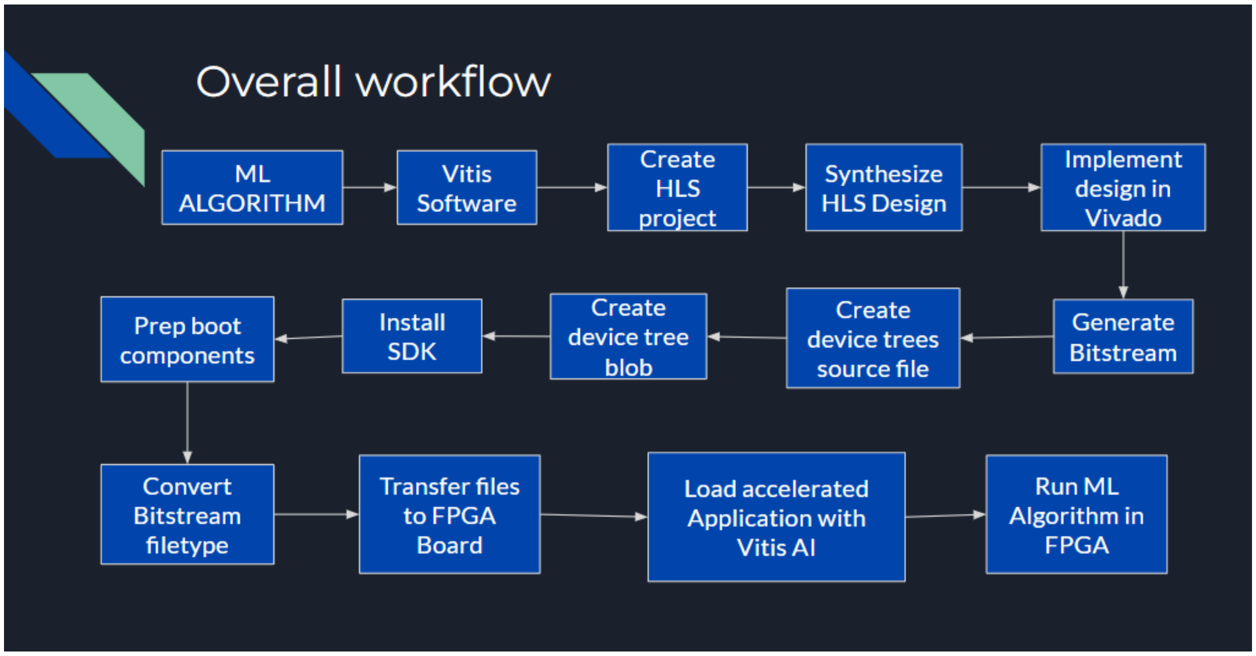
2. Deep Learning using ML algorithms are applied to the datasets for lane detection using FPGA and multicore processors

3. Compare the performances of FPGA vs multicore processors, analyzing the trade-offs between these hardware platforms across various metrics for optimized edge computing.

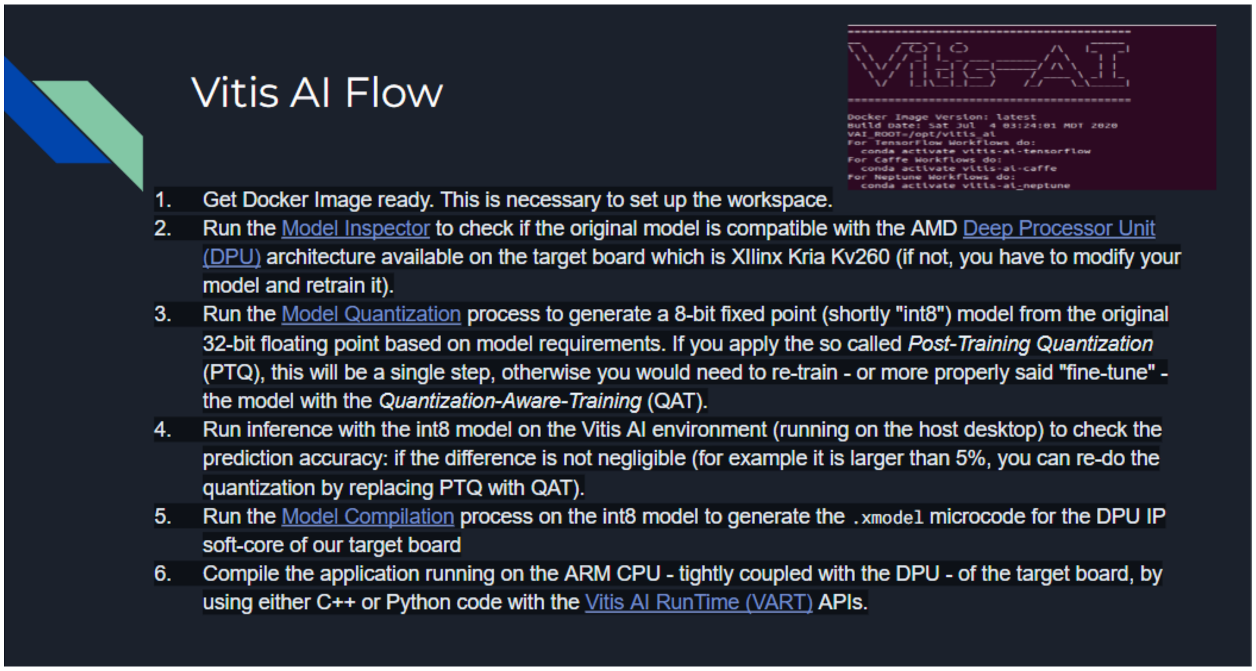
UPDATED USE CASE SCENARIOS

Show more details (break down flowchart boxes, tasks, etc. into manageable chunks that can be completed in days – not weeks/months)

Flowchart of running a model on the Kria KV260 board(Our FPGA Board)



Understanding what exactly happens in Vitis AI software:



Vitis AI Flow

```
=================================================
 \\   /(_)|(_)          /\    |__ __|
  \\ / / | | ___  ___  /  \     | |
   \V /| | | | / __|__/ /\ \    | |
    \ /|_|_|_|\___\   / ____ \ _| |_
=================================================

Docker Image Version: latest
Build Date: Sat Jul  4 03:24:01 MDT 2020
VAI_ROOT=/opt/vitis_ai
For TensorFlow Workflows do:
  conda activate vitis-ai-tensorflow
For Caffe Workflows do:
  conda activate vitis-ai-caffe
For Neptune Workflows do:
  conda activate vitis-ai_neptune
```

1. Get Docker Image ready. This is necessary to set up the workspace.
2. Run the Model Inspector to check if the original model is compatible with the AMD Deep Processor Unit (DPU) architecture available on the target board which is XIlinx Kria Kv260 (if not, you have to modify your model and retrain it).
3. Run the Model Quantization process to generate a 8-bit fixed point (shortly "int8") model from the original 32-bit floating point based on model requirements. If you apply the so called *Post-Training Quantization* (PTQ), this will be a single step, otherwise you would need to re-train - or more properly said "fine-tune" - the model with the *Quantization-Aware-Training* (QAT).
4. Run inference with the int8 model on the Vitis AI environment (running on the host desktop) to check the prediction accuracy: if the difference is not negligible (for example it is larger than 5%, you can re-do the quantization by replacing PTQ with QAT).
5. Run the Model Compilation process on the int8 model to generate the `.xmodel` microcode for the DPU IP soft-core of our target board
6. Compile the application running on the ARM CPU - tightly coupled with the DPU - of the target board, by using either C++ or Python code with the Vitis AI RunTime (VART) APIs.

UPDATED TIMELINE OF TASKS AND PLANNED WORKLOAD SPLIT FOR USE CASE SCENARIOS

Which planned timeline tasks from draft #2 were successfully completed?

- We have completed working on the machine learning model for our FPGA.
- FPGA setup finished early spring quarter.

Which ones were not completed? How will you adjust accordingly during the next 2 weeks?

- Unable to program accelerator on Xilinx Kria kv260 yet. Still working on it.
- Program the accelerator as early as possible and optimize it for the ML algorithm that we have.
- Test the prototype using a dashcam in a car, around UCI Campus

ADDITIONAL COMMENTS / CONCERNS

If there are comments/concerns you'd like to discuss with your advisor or the instructional team, state them here for reference.

- Working on optimizing the 3 models to run on Xilinx kria kv260.
- Tweaking of accelerator parameters.
- Running model on GPU for comparison, since Lane detection in cars is done using GPU.
- After testing and successful completion of model running on Kria, we move to make it real time and compare between GPU and FPGA.