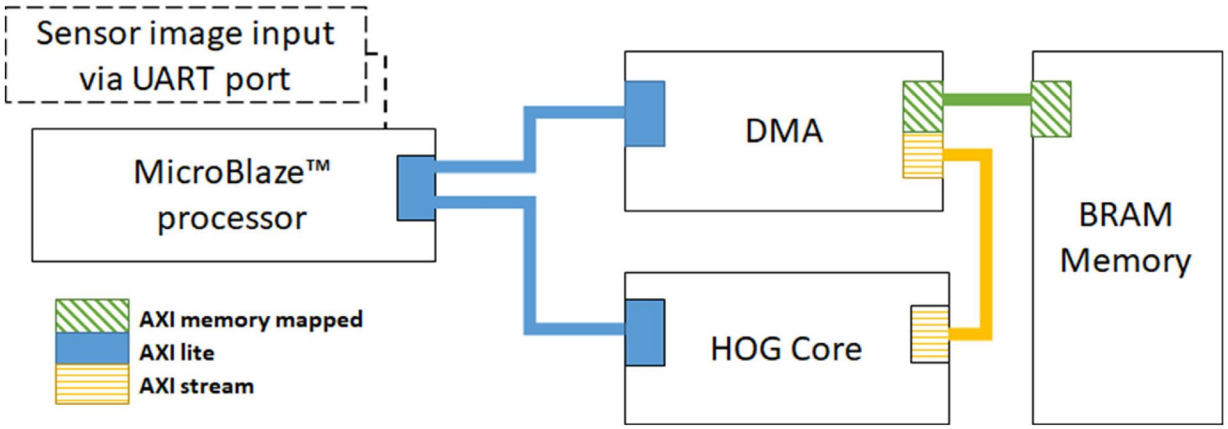
RTL Design and Verification Projects

Outline:

- **Hardware-Software Co-design of HOG-SVM classification**
- **A Fully Pipelined FPGA Architecture for Multiscale BRISK Descriptors**
- **Pipeline MIPS processor**
- **Implementation of a digital Oscilloscope based on FPGA**
- **Implementation of neural network using High Level Synthesize**

1. Hardware-Software Co-design of HOG-SVM classification

- For more detail, click here to open the publication on this work.
- In this project, I developed a hardware–software co-design system of the HOG algorithm, which can receive input data from a digital image sensor, extract the HOG features and make a decision based on those features.
- This design attains a frame rate of 115 frames per second on a Xilinx® Kintex® Ultrascale™ FPGA while using less hardware resources in comparison with other existing work.

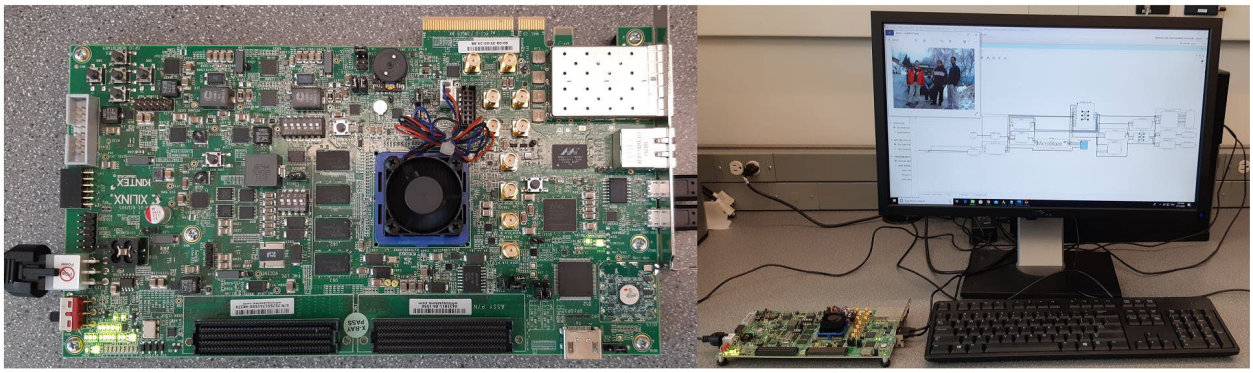


The overall architecture and interfaces.

- This project is implemented in VHDL (for the HOG-SVM core) and in C (for the program on the MicroBlaze processor).
- In this project, I developed and interface for the HOG-SVM core to handle AXI Lite bus for connection to the MicroBlaze processor.
- I also developed an AXI streaming interface so that the DMA can read the image pixels from the on-chip Block RAM memory of the FPGA and send it in an streaming manner to the HOG-SVM core.
- The HOG-SVM core is completely in VHDL and implemented on the programmable logic. The bandwidth of the designed streaming channel between the memory and the HOG-SVM IP-core is 1.2 Gbit/s, since the DMA can send each pixel in one clock cycle to the HOG core.
- We use the UART port as a matter of convenience to write the test image in the BRAM memory. Since the BRAM memory can be filled using various methods (depending on the application), this interface could be replaced with another connection interface without affecting the main concepts of this work.
- Our implementation makes four main contributions.
    - First, at the task allocation level, I proposed a well-organized partitioning between different parts in a hardware–software co-design system, which consumes fewer FPGA resources than other comparable hardware–software systems. The idea is to assign the computationally intensive parts of the algorithm, such as gradient and magnitude computation, bin assignment, normalization and classification, to hardware, and delegate the resource-intensive part, which is the windowing stage, to software.

- Second, as an algorithmic-level contribution, to the best of our knowledge, we are the first to propose a logarithm-based bin assignment in the HOG algorithm, which leads to a multiplier-free implementation of the HOG and reduces the overall number of multipliers for the HOG-SVM core.
- Third, we propose to use two parallel histogram computation modules, which save one clock cycle for every 8 pixels. As a result, the HOG core can accommodate the pixel data in a streaming manner on each clock cycle without any pause.
- Fourth, we propose a simpler implementation of the block normalization step, which reduces the IP-core resources.
- Our design has the capability to use several HOG-SVM IP-cores in parallel for one image. In future, we can modify the design to take advantage of this feature and enhance the speed of the system. Another possibility is to use interrupts efficiently to read precomputed window addresses from the memory. In this way, the processor would have more free time to perform other tasks while the HOG-SVM cores and DMAs are processing the image. Another possible enhancement involves developing other variants of the HOG algorithm and their implementation in hardware. There are many other variants of the HOG algorithm, such as HOG-3d [24], which require a high number of computations and can benefit from parallel implementation.

- FPGA platform: KCU105 evaluation board, Xilinx® Kintex® Ultrascale™ FPGA , **XCKU040**
- Software tool: Vivado Hlx

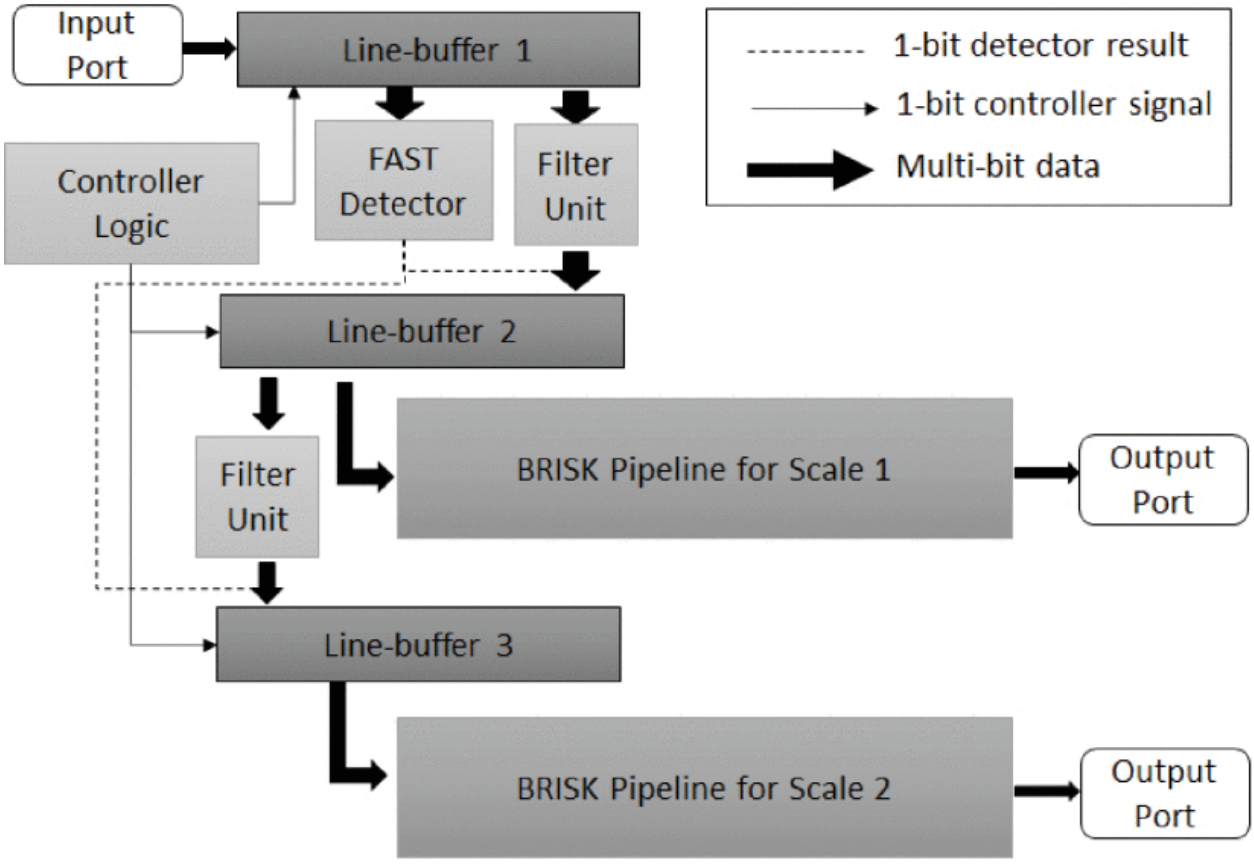


An overview of the hardware used for human detection.

- **Publication**: S. Ghaffari, P. Soleimani, K. F. Li, and D. W. Capson, “A Novel Hardware–Software Co-Design and Implementation of the HOG Algorithm,” *Sensors*, vol. 20, no. 19, p. 5655, Oct. 2020, doi: 10.3390/s20195655.

2. A Fully Pipelined FPGA Architecture for Multiscale BRISK Descriptors

- For more detail, click here to open the publication on this work.
- In this work, I developed a multiscale FPGA-based implementation of the BRISK descriptor. We presented a new sampling pattern for the BRISK algorithm with a similar number of sampling points, which reduced the number of registers for line buffers by more than **50%** and the pipeline registers **up to 73%.** The proposed design is fully pipelined and achieves a maximum operating frequency of **168 MHz**. For images with **1920 × 1080 resolution**, our proposed implementation has a frame rate of **78 fps**.
- This design is implemented in **VHDL** and verified using Vivado simulator. The output files generated by the test bench code are compared to the software model outputs for verification.
- In power consumption report we have considered interenal switcihng activity of the nodes in this design.



The overall architecture of this design.

- In this work, I used techniques such as pipelining, clock gating, and algorithmic level optimization to reduce power consumption and resource utilization and increase the speed of the implemented circuit.
- FPGA platform: KCU105 evaluation board, Xilinx® Kintex® Ultrascale™ FPGA , **XCKU040**
- Software tool: Vivado Hlx

- **Publication**: S. Ghaffari, D. W. Capson and K. F. Li, "A Fully Pipelined FPGA Architecture for Multiscale BRISK Descriptors With a Novel Hardware-Aware Sampling Pattern," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 6, pp. 826-839, June 2022, doi: 10.1109/TVLSI.2022.3151896..

3. Pipeline MIPS processor

- In this project, I developed a pipeline MIPS architecture in Verilog HDL.
- The processor has 5 stages, namely Instruction fetch, Instruction decode, Execution, Memory operations, and Write back stage.
- In order to handle data hazard, I developed a forwarding unit to forward the register values whenever there is a data dependency in the flow of the pipeline.
- There is also a controller based on finite state machine designed for this project.
- This project uses a simple code for memory simulation. The files for implementation on FPGA are not uploaded in the repository.
- Link to the Github repository of this project.

4. Implementation of a digital Oscilloscope based on FPGA

- In this project, I developed a digital oscilloscope module on DE0 Altera FPGA which shows the detected signal on a VGA monitor.
- Details to be added.

5. Implementation of neural network using High Level Synthesize

- In this project, I developed C code for HLS (High Level Synthesize) to implement a convolutional neural network for handwritten digit classification on a Xilinx FPGA.
- Details to be added.
- Link to publication.

- Publication: S. Ghaffari and S. Sharifian, "FPGA-based convolutional neural network accelerator design using high level synthesize," *2016 2nd International Conference of Signal Processing and Intelligent Systems (ICSPIS)*, 2016, pp. 1-6, doi: 10.1109/ICSPIS.2016.7869873.