Carnegie Mellon University
Software Engineering Institute

Co-Design for Edge Artificial Intelligence – Application Specific System on Chip

NOVEMBER 8, 2023

Dr. John G. Wohlbier
Senior Research Scientist
Advanced Computing Lab Lead

Document Markings

DM23-1069

Agenda

- Application-specific system on chips (SoCs)
- Accelerator design flow
- SoC integration
- Prospects
- Resources

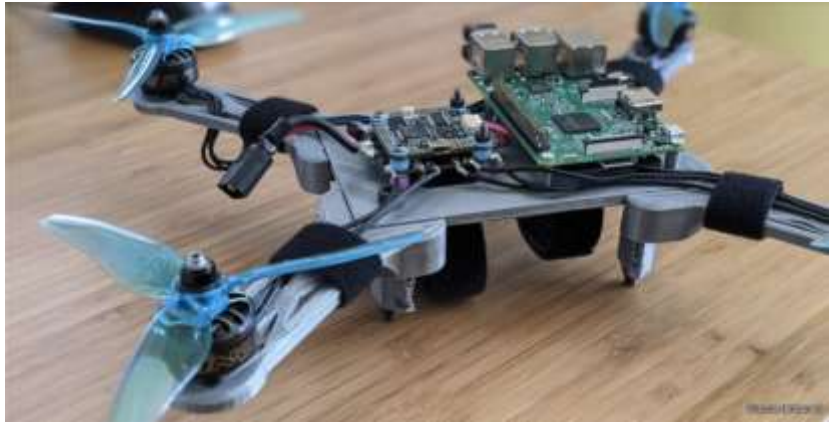
Co-Design for Edge Artificial Intelligence—Accelerator Intellectual Property

Application-Specific System on Chips (SoCs)

Carnegie Mellon University
Software Engineering Institute

Application-Specific SoCs

https://all3dp.com/2/raspbery-pi-drone-simply-explained/



https://microcontrollerslab.com/system-on-chip-soc-introduction/

- Edge devices, such as drones, are powered by microelectronics.
  - Accelerometer
  - Camera
  - Flight controller
  - GPS module
  - Speed controller
  - Radios (Tx/Rx)
- What is an SoC?
  - A chip containing components that comprise a system
  - Components: CPU cores, GPU cores, memory cores, accelerators
  - Accelerators: FFT, NLP, NVDLA, TX/RX

Application-Specific SoCs

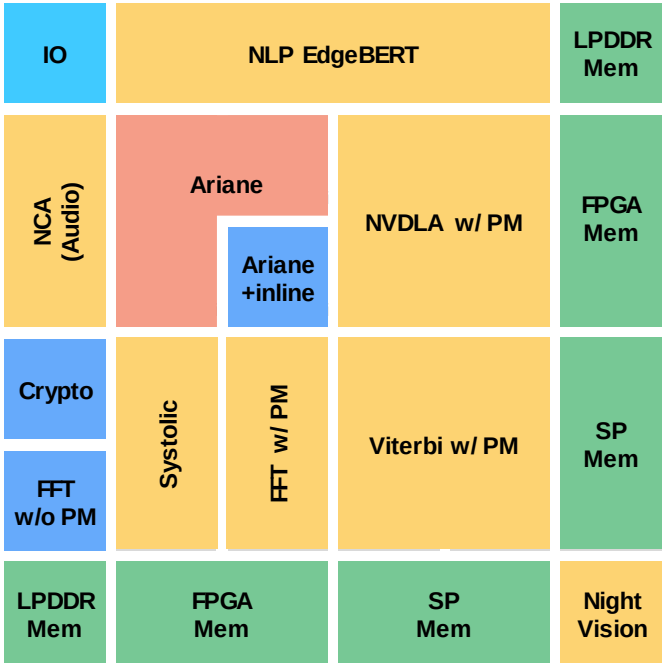


Image courtesy of IBM EPOCHS

- Electronics efficiency plays a major role in system performance.
  - Designing an SoC for a particular deployment can yield major benefit.
- What makes an SoC application specific?
  - The mix of system components is designed from the application.
  - E.g., object detection, object classification, signal characterization
- How do you design an application-specific SoC?
  - Hardware description languages (HDL): Chisel, Verilog
  - High-level synthesis (HLS): Bambu, oneAPI, AMD/Xilinx
  - Frameworks: Chipyard, Embedded Scalable Platforms

"In general, ***compute*** can improve mission time and lower energy consumption by as much as 5X."

Boroujerdian, Behzad; Genc, Hasan; Krishan, Srivatsan; Faust, Aleksandra; & Reddi, Vijay Janapa. Why Compute Matters for UAV Energy Efficiency? *International Symposium on Aerial Robotics*. 2018.

Co-Design for Edge Artificial Intelligence—Accelerator Intellectual Property

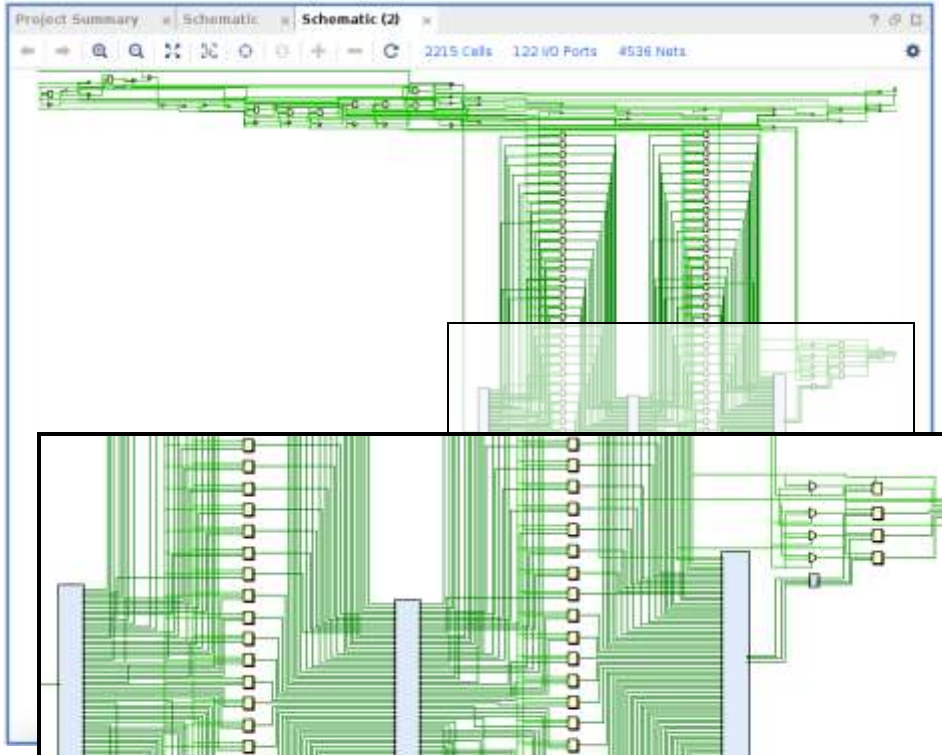
Accelerator Design Flow

Carnegie Mellon University
Software Engineering Institute

Accelerator Design Flow

- Identify hot spots of application(s) to accelerate.
  - Profiling, source code analysis, expert knowledge
- Accelerator design options
  - HDL: code the accelerator by hand (time consuming, but can achieve optimal performance)
  - HLS: tools lower code to HDL (much faster, but unlikely to achieve optimal performance)
- High-level synthesis
  - HLS tools supporting PyTorch and Tensorflow are immature.
- Emerging approaches
  - Multi-Level Intermediate Representation (MLIR) compiler framework for model lowering
  - Dialect abstraction to enable different types of optimizations

Example



- Three-layer neural network implemented in PyTorch—PyTorch Iris
  - Linear/relu/linear/relu/softmax
  - Use hls4ml with AMD/Xilinx backend for synthesis
  - Fixed point quantization: 16 bits, 6 bits left of decimal
  - HLS can be thought of as “code to circuit”
  - Ready for SoC integration

```
class Model(nn.Module):
    def __init__(self, input_dim):
        super(Model, self).__init__()
        self.relu = nn.ReLU()
        self.layer1 = nn.Linear(input_dim, 64)
        self.layer2 = nn.Linear(64,3)
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        x = self.relu(self.layer1(x))
        x = self.relu(self.layer2(x))
        x = self.softmax(x)
        return x
```

Co-Design for Edge Artificial Intelligence—Accelerator Intellectual Property

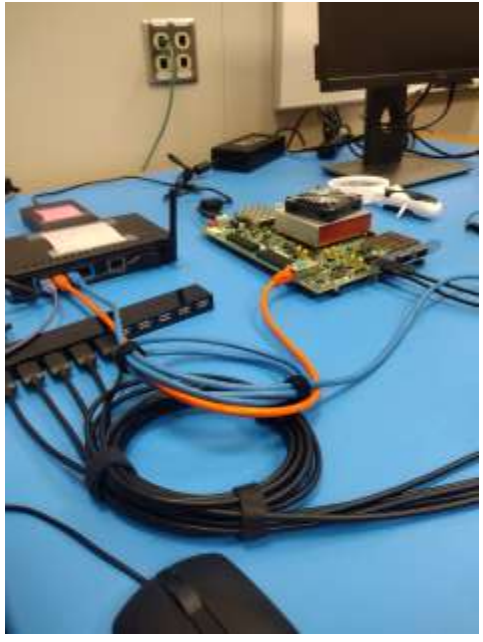
SoC Integration

Carnegie Mellon University
Software Engineering Institute

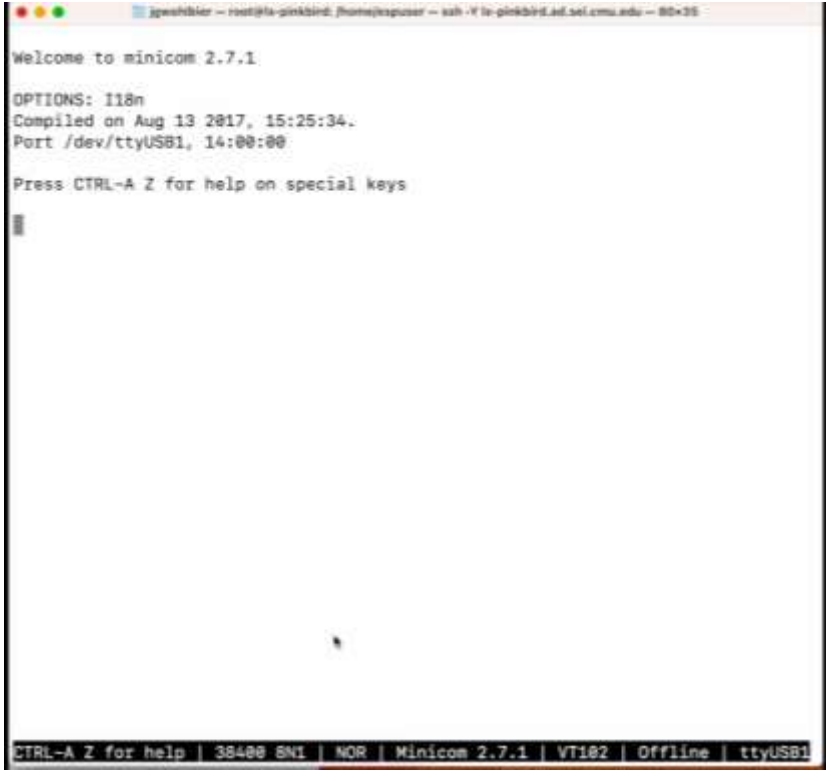
SoC Integration

- Using Embedded Scalable Platforms from Columbia University
  - Includes selectable CPUs, memory cores, I/O cores, selection of accelerators
- Example: four-tile design with an RISC-V CPU core, memory tile, I/O tile, and PyTorch-Iris accelerator
- Implementation on field-programmable gate array (FPGA) with AMD/Xilinx tools
  - Design flow for application-specific integrated circuit (ASIC) also available

SoC Testing—FPGA Implementation

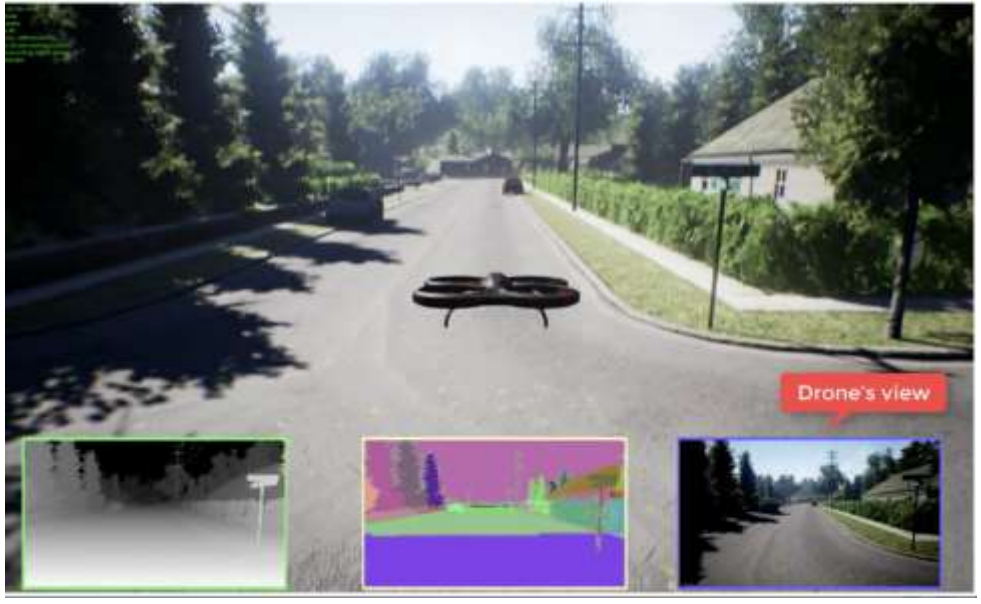


FPGA Dev Board Xilinx VCU118 connected to host. Booting Linux on RISC-V core and then running application, invoking the accelerator.



Prospects

Prospects



- Quantification of performance with real neural networks
- Accelerators for application with digital signal processing and neural network loads
- Integrated simulation of system deployment accounting for energy savings

https://paperswithcode.com/dataset/airsim

Resources

- Reproduce these results using open source code
- CMU-SEI Github
  - PyTorch-Iris – hls4ml branch – simple neural network code
    - https://github.com/cmu-sei/pytorch-iris
  - Docker environment for using hls4ml
    - https://github.com/cmu-sei/hls4ml-docker
  - Docker environment for using esp
    - https://github.com/cmu-sei/esp-docker
- ESP tutorials
  - https://www.esp.cs.columbia.edu/docs/

Contact

Dr. John G. Wohlbier
Senior Research Scientist
Advanced Computing Lab Lead

Telephone: +1 412.268.5800
Email: info@sei.cmu.edu