Implementation of UCIe2.0 Adapter Layer

Project description

Universal Chiplet Interconnect Express (UCIe), is a new open interconnect standard, aimed to facilitate a high-speed, low-latency communication between chiplets/dies within a multi-chip package or across packages.

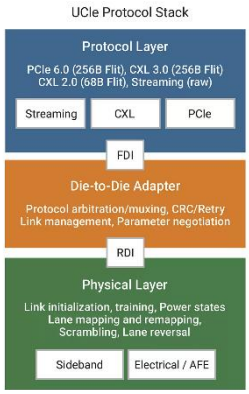
As chip sizes become increasingly constrained and the need for scalability grows, there is a push to partition core functionalities across multiple dies. This approach not only addresses size limitations but also improves yield. Consequently, efficient die-to-die communication becomes essential for integrating these separate functional units effectively.

As semiconductor designs become increasingly complex, traditional interconnect standards face limitations in bandwidth, latency, and scalability. UCIe addresses these challenges by providing a robust, high-speed interconnect standard that enhances data transfer efficiency and coherence between chiplets.

The project scope is to implement the Adapter Layer of the UCIe 2.0 with FDI and RDI interfaces to be integrated with Protocol Layer Support different protocols (Ex. PCIe, CXL2.0, CXL3.0, …. etc.).

The UCIe2.0 Adapter-layer should support

- 64 Byte FDI/RDI interface
- Main-Band Interface Datapath.
- Link initialization/Parameter Negotiation.
- CRC Generation and checking and Link Retry
- Sideband Interface.
- External retry memory.



Ref: https://www.synopsys.com/designware-ip/technical-bulletin/ucie-multi-die-socs.html

Project Activities

- Properly understand the design specs
- Propose suitable design architecture with some performance checking on MATLAB (if needed)
- Understand FPGA/ASIC digital design flow concepts and tools
- RTL design of different system blocks
- System integration and verification
- Logic synthesis for FPGA/ASIC target

Team Size

A team of 6 dedicated and self-motivated students.

How to Apply

Interested students should submit an email to (ramy.raafat@si-vision.com) and (Hossam.fahmy@eng.cu.edu.eg) with the subject: **[SiVi-GP] UCIe2.0 Adapter Layer**.

RISC-V Image Processing and Machine Learning Project Proposal

Introduction

RISC-V, an open standard Instruction Set Architecture (ISA), is rapidly transforming the landscape of computing due to its openness, flexibility, and potential for innovation. Unlike proprietary ISAs, RISC-V is free and open, which has led to the development of a rich ecosystem of tools, simulators, and software. This open-source nature allows for unprecedented levels of customization and experimentation, making it an ideal platform for academic exploration and cutting-edge research. For more information, check https://riscv.org/about/

This project invites students to dive into the world of RISC-V by developing optimized image processing and machine learning (ML) algorithms using the RISC-V Vector Extension (RVV). Students will utilize open-source simulators such as SPIKE to design, test, and optimize their algorithms. This approach provides flexibility, and also deepens students' understanding of the RISC-V toolchain and its ecosystem, offering valuable insights into the future of computing.

Graduation Project Aim

1. **Literature Review**:
    - Conduct a literature review of the RISC-V architecture, with a focus on the Vector Extension (RVV) and its applications in image processing and ML.
    - Explore the open-source RISC-V ecosystem, including simulators, and tools for development and optimization.
2. **Algorithm Development**:
    - Identify and select key image processing algorithms (e.g., edge detection, filtering) and simple ML algorithms (e.g., linear regression, decision trees) that can benefit from vectorization.
    - Develop and optimize these algorithms for the RISC-V Vector Extension using the SPIKE simulator and other open-source tools.
3. **Optimization Techniques**:
    - Apply various optimization techniques to enhance the performance of the developed algorithms on the SPIKE simulator, focusing on vectorization, memory efficiency, and other techniques.
4. **Deployment and Testing**:
    - Deploy the optimized algorithms within the SPIKE simulator environment.

- Measure and analyze the performance improvements in terms of runtime, resource utilization, and efficiency compared to non-optimized implementations.

5. **Experimentation and Evaluation**:
   - Conduct experiments to evaluate the effectiveness of different optimization techniques within the RISC-V ecosystem.
   - Assess the trade-offs between performance and resource usage, providing insights into best practices for RISC-V development.
   - Build simple applications and use cases using the optimized algorithms
6. **Research and Publication**:
   - Compile the findings into a research paper, detailing the development process, optimizations, and results within the RISC-V ecosystem.
   - Submit the paper to relevant conferences or journals for publication.

Learning Outcomes

By participating in this project, students will:

1. **Master the RISC-V Ecosystem**: Gain hands-on experience with the RISC-V toolchain, including simulators like SPIKE and QEMU, compilers, debuggers, and optimization tools. This knowledge is crucial as RISC-V continues to gain momentum in both academic and industrial settings.
2. **Develop Expertise in Algorithm Optimization**: Learn to optimize algorithms for the RISC-V Vector Extension, understanding the intricacies of vectorization and how to leverage RISC-V's unique capabilities for enhanced performance.
3. **Understand the Open-Source Advantage**: Engage deeply with the open-source nature of RISC-V, learning how to contribute to and benefit from a collaborative ecosystem that is driving innovation in computing.
4. **Prepare for the Future of Computing**: RISC-V is poised to become a dominant ISA in various domains, including embedded systems, IoT, and high-performance computing. By mastering RISC-V now, students position themselves at the forefront of this evolution, gaining skills that will be highly sought after in the industry.

Why Learning RISC-V is Crucial Now

RISC-V represents a significant shift in the world of computing. Its open-source nature allows for customization and innovation that is not possible with proprietary ISAs. As more companies and research institutions adopt RISC-V, the demand for engineers and developers who understand

this ecosystem will continue to grow. Learning RISC-V now means being prepared for the next wave of technological advancements, from embedded systems to cloud computing.

Moreover, RISC-V's open-source ecosystem encourages a deep understanding of how modern processors work, from the ground up. This is not just about using a toolchain but about understanding and potentially contributing to the very foundation of modern computing. The skills gained through this project will be invaluable, offering a competitive edge in both academic and industrial careers. Synopsys for example announced the formal adoption of RISC-V (https://www.synopsys.com/risc-v.html)

Requirements

1. **Programming and Scripting Languages**: C, C++, Python. Knowing the Assembly (for RISC-V) is a plus.
2. **Knowledge in Image Processing and Machine Learning**: Understanding of basic algorithms and their computational requirements.
3. **Familiarity with RISC-V Architecture**: Basic understanding of RISC-V and its extensions. This is a plus
4. **Experience with Open-Source Tools**: Comfort with using and contributing to open-source tools, simulators, and development environments.
5. **Team Composition**: The team should consist of 4-5 students with a mix of skills in hardware, software, and algorithm development.

How to Apply

Interested students should submit an email to omar.nasr@si-vision.com with the subject: **[SiVi-GP] Graduation Project with Si-Vision (RVV)**. Please include your resume and a brief statement of interest.

Si-Vision Graduation Project
Optimize and Deploy Deep Learning Models

Introduction

Machine learning (ML) models have revolutionized various industries by enabling systems to learn from data, make predictions, and improve over time. From image and speech recognition to autonomous driving and predictive maintenance, the applications of ML are vast and continually expanding. However, deploying these models on hardware presents unique challenges that require careful consideration and optimization to ensure efficient performance. Machine learning models are algorithms that identify patterns and make decisions based on input data. These models range from simple linear regressions to complex neural networks, each suited to different types of problems. The process of developing an ML model involves several steps, including data collection, preprocessing, model selection, training, validation, and evaluation. Once a model demonstrates satisfactory performance on validation datasets, it is ready for deployment. The journey from developing machine learning models to deploying them on hardware involves a series of crucial optimization techniques. These techniques ensure that the models not only perform accurately but also efficiently, making them suitable for a wide range of real-world applications. By addressing the challenges of deployment through careful optimization, we can harness the full potential of machine learning in both edge and cloud environments.

Before deploying machine learning models on hardware, it is crucial to optimize them for efficiency and performance. Optimization techniques can significantly reduce the computational requirements, memory usage, and energy consumption of ML models. Common optimization strategies include:

1. **Quantization:** Reducing the precision of the model's weights and activations, typically from 32-bit floating-point to 8-bit integers. This reduces the model size and computational load without significantly compromising accuracy.

2. **Pruning:** Removing redundant or less important connections (weights) in the neural network. This results in a sparser model that requires less computation and storage.
3. **Knowledge Distillation:** Training a smaller, less complex model (student) to mimic the behavior of a larger, more complex model (teacher). The student model achieves similar performance with reduced resource requirements.
4. **Model Compression:** Combining various techniques like pruning and quantization to compress the model size further while maintaining performance.
5. **Hardware-Specific Optimizations:** Tailoring the model to leverage the specific capabilities of the target hardware, such as using specialized instruction sets or accelerators.

Graduation Project Aim

1. Do a Literature review for the most famous deep learning models.
2. Do a Literature review for the common optimization techniques used.
3. Identify the most famous deep learning models needed to act as the deployed models.
4. Train the deep learning models (if needed).
5. Do Inference to make sure it is runnable (in case of already trained models).
6. Deploy the models on target: Jetson Nano and measure the runtime.
7. Apply optimization techniques over the models.
8. Redeploy the optimized models on target again and remeasure the runtime, then identify the gain out of the optimization techniques.
9. Conduct set of experiments on both machine and target with their runtimes to check which optimization technique has the best effect.
10. Write a research paper out of the project to be published in a conference or a journal.

Requirements

1. Programming and Scripting Languages: C++, python, tinyML.
2. Good knowledge of Machine Learning, Deep Learning, and Computer Vision
3. Good knowledge of Machine Learning optimization techniques.

4. Good knowledge of deployment on targets: Jetson Nano.
5. The team ranges from 4-7 Students.

Please submit an email to (omar.nasr@si-vision.com) and (mohamed.tolba@si-vision.com) with subject: **[SiVi-GP] Deep Learning Deployment** if you are interested in the project.

Si-Vision Graduation Project
Adversarial Attack on Deep Learning Models

Introduction

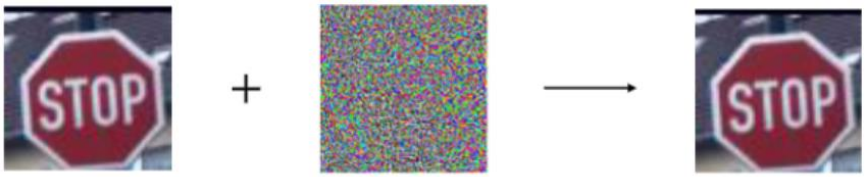
In the rapidly advancing field of machine learning, particularly deep learning, models have demonstrated remarkable capabilities in tasks ranging from image classification and natural language processing to autonomous driving and medical diagnosis. However, despite their impressive performance, these models are not impervious to vulnerabilities. One significant threat that has garnered substantial attention in recent years is adversarial attacks. Adversarial attacks involve deliberately crafting inputs to deceive machine learning models into making incorrect predictions or classifications. These inputs, known as adversarial examples, are often imperceptibly different from legitimate inputs to human observers but can drastically alter the model's output. This phenomenon poses a severe challenge to the reliability and security of machine learning systems, especially in critical applications like security surveillance, autonomous vehicles, and financial fraud detection.

Adversarial attacks exploit the inherent weaknesses and blind spots in machine learning models. These attacks typically involve adding small, carefully calculated perturbations to the original input data. The goal is to create inputs that appear normal to humans but lead the model to misclassify them with high confidence. Adversarial attacks can be classified into various types based on their goals, knowledge of the model, and the nature of the perturbations:

1. **White-Box Attacks:** The attacker has full access to the model, including its architecture, parameters, and training data. This allows for precise calculation of perturbations to maximize the effectiveness of the attack.
2. **Black-Box Attacks:** The attacker has limited or no knowledge of the model. These attacks rely on querying the model and observing its outputs to infer the necessary perturbations. Techniques such as transfer attacks and surrogate models are often employed.

3. **Targeted Attacks:** The attacker aims to mislead the model into making a specific incorrect prediction (e.g., misclassifying a stop sign as a speed limit 70 in autonomous driving).

The implications of adversarial attacks are profound and far-reaching. In security-sensitive applications, such as biometric authentication systems, self-driving cars, and medical diagnostics, adversarial attacks can lead to catastrophic failures, posing risks to safety and privacy. For example, an adversarial attack on a facial recognition system could allow unauthorized access, or a perturbation to an image of a road sign could cause an autonomous vehicle to take incorrect actions as shown in this figure:



Graduation Project Aim

1. Do a Literature review for the famous adversarial attacks' techniques.
2. Identify the deep learning models that will be attacked.
3. Train the deep learning models.
4. Do Inference and have the measurement metrics or KPIs.
5. Attack the deep learning model using different techniques.
6. Redo Inference and measure the new measurement metrics to check the effect of every technique.
7. Conduct set of experiments over deep learning models using different attacking techniques.
8. Propose a technique on how to defend against adversarial attacks (may be through a literature review)
9. Write a research paper out of the project to be published in a conference or a journal.

Requirements

1. Programming and Scripting Languages: C++, python.
2. Good knowledge of Machine Learning, Deep Learning, and Computer Vision
3. The team ranges from 4-6 Students.

Please submit an email to (hanan.yousef@eng.cu.edu.eg) and (mohamed.tolba@si-vision.com) with subject: **[SiVi-GP] Adversarial attacks** if you are interested in the project.

Analog Layout Automation

Motivation

Analog layout automation is one of the emerging challenges facing IC (Integrated Circuit) design. Analog layout usually goes through the following order of phases, namely: floorplan, placement, and finally routing. After that, comes the verification phase which includes, but is not limited to, design rule check (DRC), layout versus schematic (LVS), and parasitic extraction (PEX).

Many of these steps are circuit's type constraints dependent, whether it is Power Management, Data Converters (DAC/ADC), Timing Circuits (PLL/CDR) or Radio Frequency Circuits (LNA/PA/Mixers). Each type has its own systematic approaches to layout production based on its own constraints

Automation of analog layout will lead to less design iterations, faster delivery with higher quality layout production.

Responsibilities

- Identify a valid real world analog layout problem that can be automated.
- Innovative usage / adaptation of algorithms (either conventional or AI related) to design and implement an automation solution to the identified problem.
- Test the solution against several situations and identify its limitations.
- State the enhancements needed to overcome the limitations and provide a clear workflow on how to accomplish that.

Intended Outcome

- Gain knowledge of analog layout production.
- Use acquired knowledge to clearly identify one or more valid real world analog layout problems that can be automated.
- Design and implementation of an automation solution to one or several real world analog layout problems.
- Learn good and clean code practices.
- Gain knowledge of version control.
- Apply documentation best practices.

Qualifications

- A Team of 4-6 dedicated and self-motivated students
- Basic understanding of analog design.
- Basic knowledge of IC fabrication and layout design.
- Basic programming skills [C++/Python and Tcl language knowledge is a plus].
- Fair knowledge of IC design tools, Custom Compiler is a plus.
- Fair Knowledge of IC layout verification tools [Calibre and IC verification ICV tool is a plus].

How to Apply

The project will be under the supervision of **Dr. Mohamed Refky.**

Interested students should submit an email to (fady.atef@si-vision.com), (mai.hegab@si-vision.com) and (graduation.project.cu@gmail.com) with the subject: **[SiVi-GP] Analog Layout Automation**

Modeling and Verification of Simplified UCIe Adapter Graduation Project

Description

Universal Chiplet Interconnect express (or simply UCIe) is a new specification that defines a universal communication standard between all chiplets in the industry. It helps in unifying the way that chiplets from different third party companies talk to each other, paving the way to improved chiplet ecosystem and growing chiplets market.

Chiplets are separately fabricated dies that are combined into one package. THe concept of chiplets emerged as a way to workaround the limitations on System-on-Chip size. SOC designs require more area for improved performance and more complicated tasks. The design size is hitting the reticle limit. The idea of chiplets is to divide complicated designs into smaller dies that are separately fabricated but combine them into one package. This is called 'disaggregation' which requires high bandwidth, low latency communication between dies on a package to maintain the overall performance of the design. UCIe standard is created to present a standard specification for chiplet-to-chiplet communication.

In this project, we will work on modeling a simplified version of the UCIe Adapter, a bridge that models UCIe PHY and a testbench for verifying two Adapters connected back-to-back.

Project Activities

- Training required: Systemverilog, UVM, SVA.
- Properly understand the UCIe specification.
- Architect an RTL model for Adapter and corresponding UVM testbench.
- Implement testbench wrappers in Dut-to-Dut topology.
- Implement required UVM components including Fcov and scoreboard.
- Add tests and close code coverage.

Team Size

A team of 6-8 dedicated and self-motivated Students

How to Apply

Interested students should submit an email to (salaheldin.ashraf@si-vision.com) and (Hossam.fahmy@eng.cu.edu.eg) with the subject: **[SiVi-GP] Modeling and Verification of Simplified UCIe Adapter**

Sponsored by Si-Vision