1. ***Nhóm chủ đề về thiết kế vi mạch điện tử và quang tử***

**1. Mô phỏng và phân tích hiệu suất vi xử lý RISC-V 32-bit qua phần mềm mô phỏng**

* **Mục tiêu:**
  + Nghiên cứu cách thực thi lệnh của vi xử lý RISC-V 32-bit thông qua trình mô phỏng.
  + Đánh giá tốc độ xử lý, số chu kỳ clock cần thiết để thực hiện một tập lệnh cụ thể.
  + So sánh hiệu suất khi chạy các chương trình đơn giản (ví dụ: vòng lặp, thuật toán sắp xếp).
* **Công cụ:**
  + **Spike** (RISC-V ISA Simulator), **QEMU**, **Renode** để mô phỏng vi xử lý.
  + **RISC-V GNU Toolchain** để biên dịch mã.
  + **Gem5** để phân tích hiệu suất và đánh giá số chu kỳ lệnh.
* **Ứng dụng:**
  + Hiểu sâu về cách vi xử lý RISC-V thực thi lệnh.
  + Đánh giá hiệu suất của vi xử lý với các bài kiểm tra chuẩn (benchmarking).
* **Gợi ý phương pháp:**
  + Nhóm **tìm hiểu RISC-V**: Nghiên cứu tập lệnh (RV32I, RV64I) và cách thực thi.
  + Nhóm **lập trình thử nghiệm**: Chạy chương trình bằng C và Assembly, quan sát kết quả trên Spike.
  + Nhóm **phân tích hiệu suất**: Đo số chu kỳ cần thiết để chạy từng đoạn mã và tối ưu hóa lệnh.
  + Nhóm **viết báo cáo**: Tổng hợp kết quả, so sánh hiệu suất với các kiến trúc khác (ARM, x86).

**2. Khảo sát tốc độ truy xuất và băng thông bộ nhớ SRAM và DRAM trong hệ thống RISC-V**

* **Mục tiêu:**
  + Mô phỏng và đánh giá hiệu suất truy xuất bộ nhớ trong vi xử lý RISC-V.
  + So sánh tốc độ đọc/ghi giữa bộ nhớ SRAM và DRAM.
  + Phân tích ảnh hưởng của bộ nhớ cache lên tốc độ xử lý.
* **Công cụ:**
  + **Gem5**, **Renode**, **QEMU** để mô phỏng bộ nhớ trên RISC-V.
  + **DRAMSim3** để mô phỏng hiệu suất DRAM.
  + **RISC-V GNU Toolchain** để biên dịch chương trình test.
* **Ứng dụng:**
  + Hiểu rõ cách vi xử lý RISC-V giao tiếp với bộ nhớ.
  + Đánh giá ảnh hưởng của tốc độ đọc/ghi lên hiệu suất hệ thống.
* **Gợi ý phương pháp:**
  + Nhóm **mô phỏng bộ nhớ**: Sử dụng Gem5 để chạy mô hình SRAM, DRAM, và cache.
  + Nhóm **lập trình benchmark**: Viết chương trình đọc/ghi bộ nhớ với nhiều kích thước dữ liệu khác nhau.
  + Nhóm **phân tích hiệu suất**: Đánh giá thời gian truy xuất bộ nhớ, độ trễ cache.
  + Nhóm **báo cáo kết quả**: So sánh hiệu suất giữa SRAM, DRAM, và cache khi chạy cùng một chương trình.

**3. Phân tích điện năng tiêu thụ và tối ưu hóa tốc độ của vi xử lý RISC-V bằng phần mềm mô phỏng**

* **Mục tiêu:**
  + Đánh giá mức tiêu thụ điện năng của vi xử lý RISC-V khi chạy các tác vụ khác nhau.
  + So sánh hiệu suất giữa các mức xung nhịp (clock speed) khác nhau.
  + Tìm cách tối ưu hóa hiệu suất mà vẫn giảm mức tiêu thụ năng lượng.
* **Công cụ:**
  + **Gem5**, **McPAT** để mô phỏng điện năng tiêu thụ.
  + **RISC-V GNU Toolchain**, **QEMU** để chạy mã nguồn thử nghiệm.
* **Ứng dụng:**
  + Nghiên cứu chiến lược tiết kiệm điện trong thiết kế vi xử lý RISC-V.
  + Ứng dụng trong các thiết bị IoT cần tối ưu điện năng.
* **Gợi ý phương pháp:**
  + Nhóm **mô phỏng điện năng**: Sử dụng McPAT để ước lượng mức tiêu thụ điện của từng lệnh.
  + Nhóm **chạy thử nghiệm**: Chạy các chương trình với xung nhịp 100 MHz, 200 MHz, 500 MHz để so sánh hiệu suất.
  + Nhóm **tối ưu hóa**: Thử nghiệm giảm điện áp, áp dụng các kỹ thuật quản lý năng lượng (Dynamic Voltage Scaling).
  + Nhóm **báo cáo kết quả**: So sánh hiệu suất trên từng mức điện áp và xung nhịp.

**4. Kiểm tra và mô phỏng bộ tăng tốc AI đơn giản trên RISC-V**

* **Mục tiêu:**
  + Mô phỏng cách vi xử lý RISC-V thực hiện các tác vụ AI cơ bản như nhận dạng ảnh hoặc xử lý tín hiệu.
  + Phân tích tốc độ thực hiện và yêu cầu bộ nhớ của một mô hình AI đơn giản trên RISC-V.
* **Công cụ:**
  + **TVM**, **TensorFlow Lite** để chạy mô hình AI.
  + **Renode**, **Gem5** để mô phỏng thực thi trên RISC-V.
  + **RISC-V GNU Toolchain** để biên dịch mã chạy trên mô phỏng.
* **Ứng dụng:**
  + Đánh giá khả năng chạy AI trên phần cứng nhúng dùng RISC-V.
  + So sánh hiệu suất giữa RISC-V và kiến trúc khác (ARM, x86).
* **Gợi ý phương pháp:**
  + Nhóm **xây dựng mô hình AI**: Chọn một mô hình đơn giản (ví dụ nhận dạng chữ số MNIST).
  + Nhóm **mô phỏng thực thi**: Chạy mô hình trên trình giả lập RISC-V.
  + Nhóm **phân tích hiệu suất**: Đo thời gian thực thi, tiêu thụ bộ nhớ.
  + Nhóm **tối ưu hóa**: Thử nghiệm với định lượng mô hình (quantization) để tăng tốc độ xử lý.

Reference

Dưới đây là một số nguồn tài nguyên chính mà nhóm sinh viên có thể sử dụng:

**1. Vi xử lý RISC-V và công cụ mô phỏng**

* **Spike (RISC-V ISA Simulator)**
  + GitHub: <https://github.com/riscv/riscv-isa-sim>
  + Mô phỏng thực thi tập lệnh RISC-V ở cấp độ ISA (Instruction Set Architecture).
* **QEMU (RISC-V Emulator)**
  + GitHub: <https://github.com/qemu/qemu>
  + Mô phỏng CPU RISC-V 32/64-bit, hỗ trợ chạy Linux.
* **Renode (RISC-V System Emulator)**
  + GitHub: <https://github.com/renode/renode>
  + Mô phỏng hệ thống nhúng, hỗ trợ cả bộ nhớ, GPIO, UART...
* **Gem5 (Computer Architecture Simulator)**
  + GitHub: https://gem5.googlesource.com/public/gem5
  + Hỗ trợ mô phỏng CPU, cache, bộ nhớ, và đánh giá hiệu suất hệ thống.

**2. Bộ nhớ SRAM, DRAM và cache trong RISC-V**

* **DRAMSim3 (DRAM Simulator)**
  + GitHub: <https://github.com/umd-memsys/DRAMSim3>
  + Giúp mô phỏng tốc độ đọc/ghi và độ trễ của bộ nhớ DRAM.
* **Ramulator (Memory System Simulator)**
  + GitHub: <https://github.com/CMU-SAFARI/ramulator>
  + Dùng để mô phỏng hệ thống bộ nhớ DRAM, LPDDR4, HBM.

**3. Điện năng tiêu thụ và tối ưu hóa hệ thống RISC-V**

* **McPAT (Multicore Power, Area, and Timing)**
  + GitHub: <https://github.com/HewlettPackard/mcpat>
  + Phân tích điện năng tiêu thụ của vi xử lý, hỗ trợ RISC-V.
* **ORCS (Open Source Cycle-Accurate Simulator)**
  + GitHub: <https://github.com/ucb-bar/chipyard>
  + Giúp đo hiệu suất và điện năng tiêu thụ của hệ thống SoC RISC-V.

**4. Chạy thuật toán AI trên vi xử lý RISC-V**

* **TVM (AI Compiler for RISC-V)**
  + GitHub: <https://github.com/apache/tvm>
  + Hỗ trợ chạy AI trên nền tảng phần cứng nhúng, bao gồm RISC-V.
* **TensorFlow Lite (for Microcontrollers)**
  + GitHub: <https://github.com/tensorflow/tflite-micro>
  + Tối ưu hóa mô hình AI cho hệ thống nhúng như RISC-V.

**5. Thiết kế Photonic Integrated Circuits 4x4 Switch**

Hướng nghiên cứu:  
Sử dụng nền tảng Rsoft CAD và công cụ BeamPROP để thiết kế tôi ưu một bộ 4x4 Thermo-optic Switch sử dụng điều khiển nhiệt bằng Metallic thin film trên Al/Cr/Ni/W/Ag/Au…và các Mach-Zehnder Interferometers cho dải bước sóng Telecom window 1550 nm, nền tảng đơn mode và vật liệu SOI

**6. Thiết kế bộ phân kênh 16 bước sóng x 2modes trong C-band của Telecom window**

Hướng nghiên cứu: Thiết kế bộ phân kênh 16 bước sóng x 2modes trong C-band của Telecom window bằng cách sử dụng cấu trúc AWG (Arrayed Waveguide Grating) trên nền vật liệu Si/SOI và công cụ mô phỏng Rsoft Component Suite