IIS5008 Hardware Security
Programming Assignment 3
Trusted Execution Environment of Multi-core System
(Due:2025/06/06 23:59:59)

Introduction

A **System-on-Chip (SoC)** is a design that integrates multiple system modules onto a single chip, aiming to embody the concept of "**one chip per system**." To address the issue of **inefficient communication between modules** within SoCs, the **Network-on-Chip (NoC)** architecture was introduced. It replaces traditional **bus-based communication** with a network-like structure, using **routers** and **data packets** for transmission. NoC employs **topological architecture** such as **Mesh**, **Torus**, and others, enabling **simultaneous communication between multiple modules**, which provides **higher bandwidth** and **lower latency**.

In this assignment, you are expected to gain an understanding of the **NoC architecture**, distinguish between **binary computing** and **stochastic computing**, and explore their respective performance in terms of **fault tolerance**.

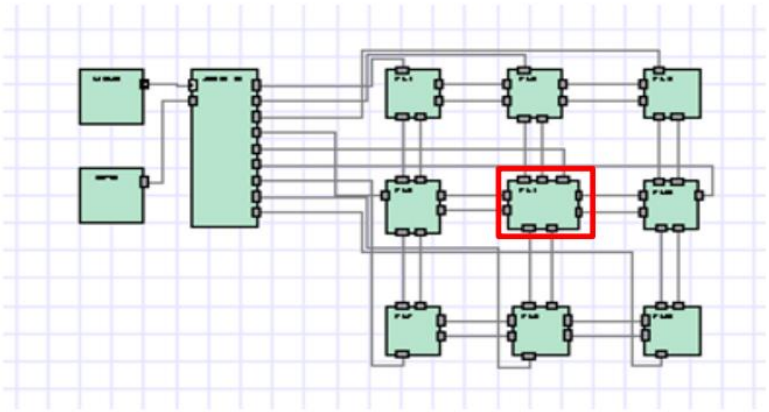

Fig 1. Block diagram of the NoC

Background

1. Approximate Computing

Approximate Computing is a computing paradigm that prioritizes speed, area, power efficiency, or fault tolerance over absolute accuracy. In many applications—such as image recognition or neural networks. Certain levels of error are tolerable, meaning we don't need ultra-precise computation at every step. Approximate arithmetic units, reduced-precision operations, or even logic simplifications can be applied to reduce hardware complexity and energy consumption. Stochastic

Computing (SC) is one of approximate computing.

2. Stochastic Computing

SC is a type of digital computation where values are represented probabilistically and operations are performed using simple logic gates, such as **AND**, **MUX**, and **XNOR**. SC has recently regained attention due to its hardware friendliness, making it attractive for low-power, fault-tolerant, and edge computing applications.

In SC, a value is not represented in binary form, but rather through a bit-stream. Each bit in the stream does not indicate a fixed position—instead, it represents an instance of a probabilistic event. SC doesn't prioritize precision but rather focuses on delivering acceptable results under conditions of high fault tolerance, ultra-low hardware requirements, and resource-constrained environments.

- Unipolar: encodes values between 0 and 1, commonly used for probabilistic multiplication.
- Bipolar: uses symmetric encoding from -1 to 1, allowing it to handle both positive and negative values, making it suitable for neural network applications.

Problem Description

In this experiment, we explore how SC can be integrated with the Platform Architect simulation framework to implement a fault-tolerant LeNet neural network inference system.

By injecting faults into a specific processing element (e.g., PE4), we compare the accuracy differences between the traditional binary computing method and the novel stochastic computing approach under faulty conditions.

Our target is to use stochastic computing to decrease the impact of error injection. Please follow your PA step by step.

- Step 1
  Constructing the environment on Platform Architect. Please refer to the PA3 tutorial provided in the course on eeclass. (PA3_tutorial.pdf)
- Step 2
  Using the stochastic computing to implement multiply-accumulate operation. (sc_function_bipolar.h)
- Step 3

Comparing results after error injection between binary computing and stochastic computing. (error_injection.h)

- Step 4
  Connect all the related files used for constructing the environment of LeNet model on Platform Architect to your design, sc_function_bipolar.h and error_injection.h.
- Step 5
  Simulate both results to comparing the accuracy after attacking with stochastic computing and conventional binary computing.
- Step 6
  Complete your report and submit to eeclass.

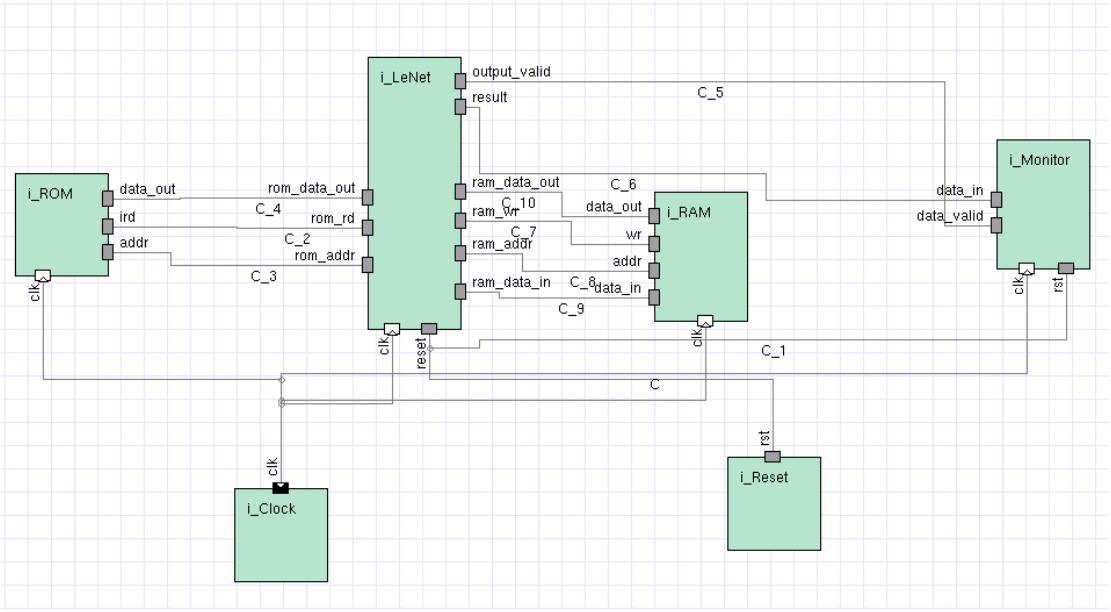

Fig 2. Overall block diagram of the LeNet NoC

Report

Your report must contain the following contents, and you can add more as you wish.

(1) Your name and student ID

(2) How to compile and execute your program and give an execution example.

(3) Explanation of provided LeNet model (HW3.project). You can choose to explain the most important parts. You should explain it in your own words.

(4) Explanation of your design (error_injection.h and sc_function_bipolar.h). You should explain it in your own words.

(5) Conduct different comparative experiments and collect data, as shown in the figure below. Write down your observations with these results.

Error Injection Ratio (%)	5	10	15	20	30
Accuracy of BC					
Accuracy of SC					

Table 1. Comparison of binary computing and stochastic computing.

(6) What have you learned from this homework? What suggestions or thoughts do you have about PA3?

Requirement

Please compress **PA3/** (using tar) into one with the name ${StudentID}_PA3.tar.gz before uploading it to eeclass, For example, 123456789_PA3.tar.gz.

(1) **PA3/** : contains all your systemC code and related files.

(2) **PA3/export/** : contains sim.tcl to simulate your project by only one script.

```
.
├── PA3/                                  # Your PA directory
│   ├── PA3.project
│   ├── config.h / define.h / Tile.h
│   ├── clockreset.h / clockreset.cpp
│   ├── Monitor.h / Monitor.cpp
│   ├── NI.h / NI.cpp
│   ├── Processing_Element.h / Processing_Element.cpp
│   ├── RAM.h / RAM.cpp
│   ├── ROM.h / ROM.cpp
│   ├── Router.h / Router.cpp
│   ├── key.txt / LeNet_All_b.txt / traffic.txt
│   ├── README.md
│   └── error_injection.h / sc_function_bipolar.h
│
├── export/                               # Your output data
│   ├── sim.tcl
│   ├── PA3_result.txt
│   └── 123456789_陳聿廣_PA3.pdf
│
```

(3) ${StudentID}_${Name}_PA3.pdf (ex. 123456789_ 陳 聿 廣 _PA3.pdf) contains all the required contents described in **Report** section. Note that the only acceptable report file format is .pdf, in either Chinese or English. You can use the following command to compress your directory on a workstation:

$ tar -zcvf {StudentID}_PA3.tar.gz

Example:

$ tar -zcvf 123456789_PA3.tar.gz

Grading

The grading is as follows:

(1) 50%: Correctness of your systemC code.

(2) 40%: The report format and content:
Reports that use visual elements such as diagrams and tables to enhance clarity and improve readability will receive higher scores. The explanation of the code should be concise and impactful, focusing on quality over quantity. **You can choose to highlight and explain the most important parts.**

(3) 10%: Demo session:

Please submit your assignment on time. Otherwise, the penalty rule will app b ly:

1. Within 24hrs delay: 20% off
2. Within 48hrs delay: 40% off
3. More than 48hrs: 0 point

Notes

You MUST write this program in **systemC** and use **Plateform Architect** as the simulation platform. You MUST implement the assignment independently. If you refer to any external materials, you MUST cite them properly and demonstrate a thorough understanding of the content.

Contact

For all questions about PA3, please send Email to TA 白琪澤. Email to j37619821765321@gmail.com with the title [HW Security PA3].

Reference

[1] NTU Online SystemC Tutorial PDF
http://media.ee.ntu.edu.tw/courses/msoc/slide/02_SystemC_Tutorial.pdf

[2] Learn SystemC
https://www.learnsystemc.com/
[3] A. Alaghi, W. Qian and J. P. Hayes, "The Promise and Challenge of Stochastic Computing," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 8, pp. 1515-1531, Aug. 2018.