CS3120 Introduction of Integrated Circuit Design

Final Project: Digital Computation-in-Memory

Due Date: 2024/12/29(Sun) 23:59:59

Background

Digital Computation-in-Memory (DCIM) is an innovative computing paradigm that performs computations directly within memory arrays, addressing the data transfer bottlenecks of traditional architectures. By integrating storage and processing, CIM significantly improves energy efficiency and computational speed. This technology is particularly suited for data-intensive applications such as machine learning, image processing, and large-scale data analytics, offering a scalable and efficient solution for next-generation systems.

Description

In this final project, please design a 4-rows x 2-columns DCIM macro for vector-matrix multiplications as shown in Figure 1. The input side includes an input vector **I1, I2, I3, I4** (each of the element have 4 bits, and should be sent 1 bit per cycle, from the most significant bit to the least significant bit, over a total of 4 cycles to generate an output), three control signals **CLK** (clock, period: 2ns), **RST** (Asynchronous active-low reset), and **IN_VAL** (active -high for input valid). The output side consists of two 10-bit output signal **O1, O2** (10-bit, represent the result for column 1 and 2), and **OUT_VAL** (active -high for output valid). The power supply voltage VDD is 1.8V, and VSS is 0V.

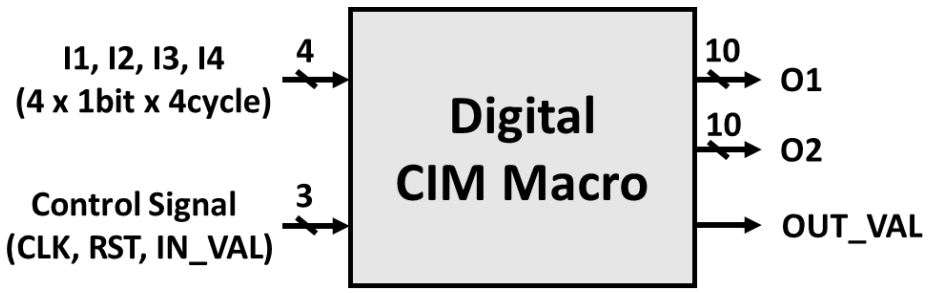

Fig. 1 Block diagram

This DCIM macro multiplies the inputs **I1–I4** with the weights **W** stored in memory and sums the results to produce outputs **O1** and **O2**, as shown in Figure 2. Memory cells are represented using latches. In each cycle, one bit is sent to the memory cell for each row. After multiplying the inputs and weights within the latches, an adder

tree combines all the partial results. The sum **P** is stored in a register for shifting and adding with the next bit's result in the following cycle. Once the calculations for all 4 bits are completed, the results are sent to **O1** and **O2**, and the output valid signal is set high to indicate the computation is complete.

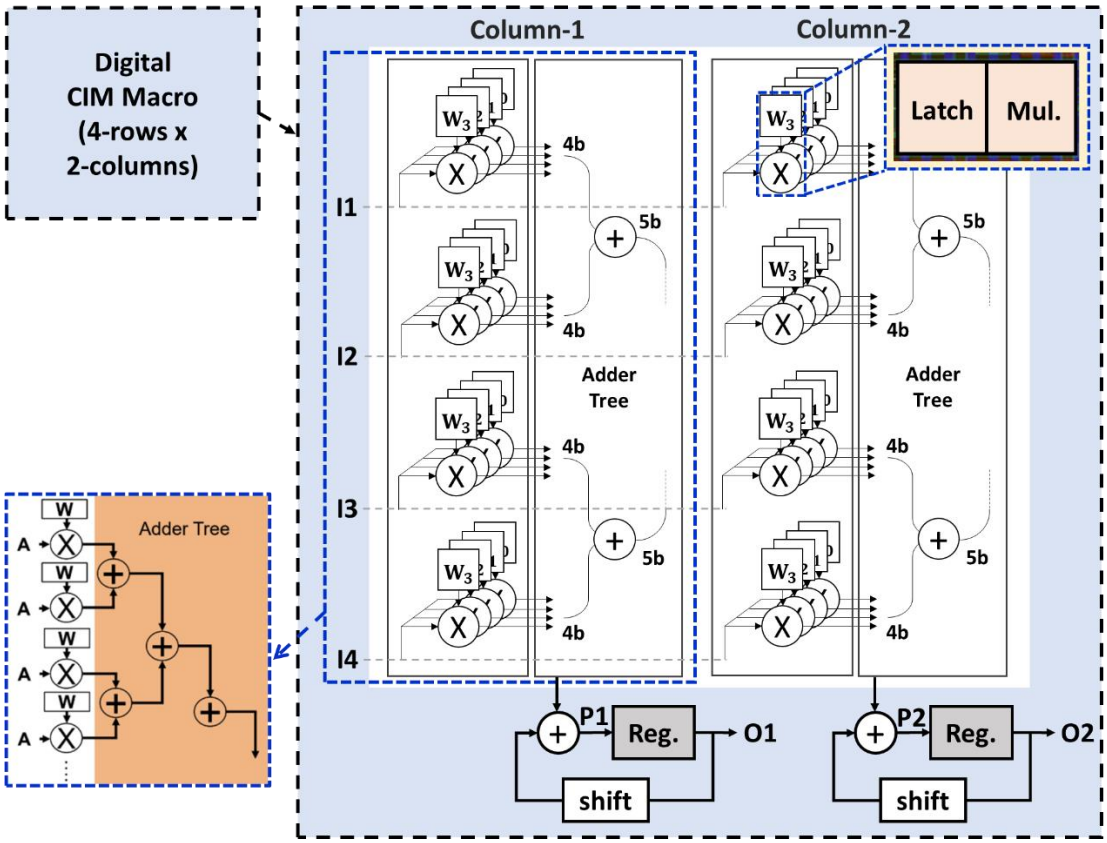

Fig. 2 Digital CIM macro architecture

function definition

- Inputs vector (I1, I2, I3, I4):
  - I1, I2, I3, I4 are 4 bits
  - Give the input bit serially:
    4 inputs (4 bits each) should be sent to the DCIM macro, 1 bit per cycle, over a total of 4 cycles to generate an output
- Weights matrix (W):
  - 4 rows x 2 columns: $(W11, W12, W13, W14)_1$, $(W21, W22, W23, W24)_2$
  - All weights are 4 bits
  - Weights should be stored in latches by initial conditions (.ic)
- Output (O1, O2):
  - O1, O2 are 10 bits
  - $O1 = (I1 \times (W11)_1) + (I2 \times (W12)_1) + (I3 \times (W13)_1) + (I4x (W14)_1)$
  - $O2 = (I1 \times (W21)_2) + (I2 \times (W22)_2) + (I3 \times (W23)_2) + (I4x (W24)_2)$

Signal Specification

All signals and their definitions are listed in Table 1. All input, output, and combinational signals in your program must be named exactly as specified in Table 1. If a signal is wider than 1 bit, each bit should be named by appending *n* to the signal name, where *n* represents the bit position (with 0 indicating the least significant bit). For example, the output signal **O1** should be named **O10**, **O11**, ..., **O19**, where **O10** is the least significant bit, and **O19** is the most significant bit.

Table. 1 signals information

Input signal	Bit width	Definition
I1, I2, I3, I4	1 bit each	Input serial signals
CLK	1 bit	Clock, period: **2 ns**
RST	1 bit	Asynchronous active-low reset
IN_VAL	1 bit	Inputs are valid
Output signal	Bit width	Definition
O1, O2	10 bits each	Output signals
OUT_VAL	1 bit	O1 and O2 are valid (finished)
Combinational signal	Bit width	Definition
ii1, ii2, ii3, ii4	1 bit each	The input from flip-flop
P1, P2	10 bits each	The output before flip-flop

This circuit operates on the positive edge of the clock, and the value of the output signal will be checked at the negative edge. Flip-flops/registers must be placed both before the input signals **I** and after the output signals **O** (as shown for the output register in Figure 2). To measure the delay time of your circuit, the inputs after the flip-flops should be named **ii1–ii4**, and the outputs before the flip-flops should be named **P1** and **P2**, as listed in Table 1. The delay time of your circuit will be measured from **ii1** to **P10**.

The clock period in this program is set to 2 ns, with a rise and fall time of 1 ps for all signals. Please note that for the output logic:

- A logic "1" requires the voltage to be greater than 0.9 VDD.
- A logic "0" requires the voltage to be less than 0.1 VDD.

(Hint: You may add buffers to strengthen the output voltage if necessary.)

Weight initialization

In this program, latches are used to represent memory cells, and weights are stored in the latches through initial conditions (.ic). Please set the weights as follows (described in decimal format):

- Column1: $(W11, W12, W13, W14)_1 = (1, 3, 7, 15)_1$
- Column2: $(W21, W22, W23, W24)_2 = (2, 6, 8, 12)_2$

The method for setting initial conditions is provided in the tutorial on ee-class.

Waveform Example

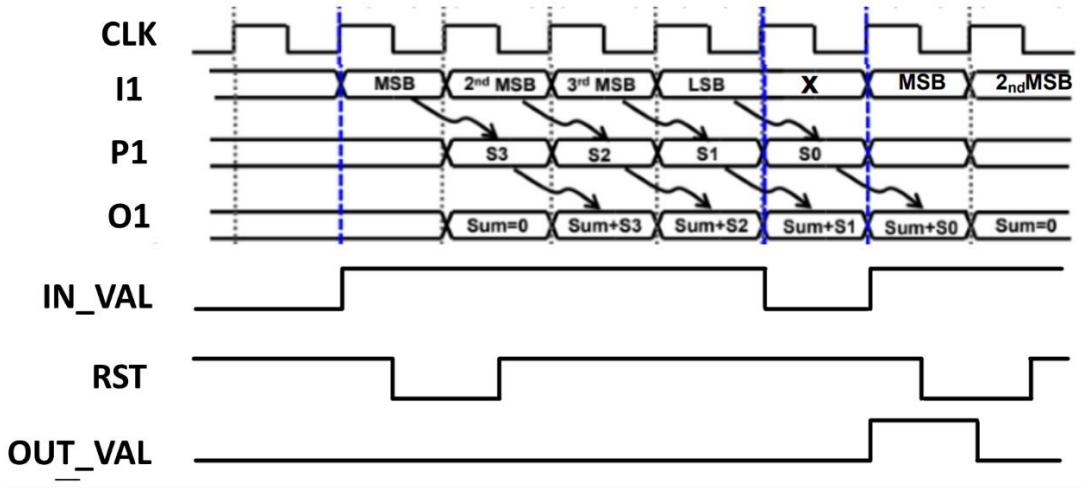

Fig. 3 Waveform example

Bonus

For the bonus version, please design a 32-rows × 16-columns DCIM macro. The functionality should remain the same as the original version. All weights stored in the memory cells should be set to n-1, where n represents the column number. For example, all weights in the first column should be set to 0, while all weights in the 16th column should be set to 15.

If you have implemented the bonus version, please separate the code for the bonus version and the original version, and ensure that you follow the naming rules below.

Limitations

Below are the guidelines that must be adhered to for this assignment. Any violation of these guidelines will result in 0 points for this project.

Environment setup and parameters

- You must complete this project using the process file provided by us. (on the ee-class)
- The naming and pinouts of each input and output must match table 1 provided in this document.
- The operating voltage (VDD) is set to 1.8V, VSS is set to 0V, and the operating temperature is 30˚C.
- For transistors, please set the **L=0.18um**, the width of **NMOS W=0.36 um**, and the width of **PMOS W=0.72um.**
- You must use the “cic018.l” to do this assignment, otherwise you will get 0 point.

Naming rules

- SPICE code file name: DCIM.sp
- SPICE code file name (bonus version): DCIM_bonus.sp
- Primary circuit name: DCIM
- Cell name for your primary circuit: Xdcim
- Signals in the program: named as we mention in **signal specification** part.

Submission Requirement

Please upload the following file to the ee-class:

(1) DCIM.sp
The SPICE code for your program.

(2) StudID_Name_report.pdf (ex: 9862534_陳聿廣_report.pdf)
Your report file. Note that the only acceptable report file format is .pdf, no .doc/.docx or other files are acceptable.

(3) DCIM_bonus.sp
The SPICE code for the bonus version, if implemented.

BE SURE to follow the naming rule mentioned above. Otherwise, your program will be not graded.

We don't restrict the report format and length. In your report, you must at least describe:

(1) The circuit diagram of your design and explaining your design (You can use screenshot to explain)
(2) The transistor level view of your CIM circuit and adder (You can just draw one of them and specified how you connect each)
(3) The bonus circuit if implemented
(4) Waveform of your simulation
(5) The delay and the power of the circuit
(6) The total number of transistors (NMOS and PMOS) you use in this program
(7) The hardness of this assignment and how you overcome it
(8) Any suggestions about this programming assignment

You can also put anything related to the PA in your report.

Grading

The grading is as follows: (110%)

(1) Correctness (50%)
- 5 test cases for each 10%

(2) Performance (20%): 5% (delay time) + 5% (power) + 10% (area)

- Delay time and power by your simulation result
- Area for your number of transistors in your program
- Score function: maximum_score*(your result percentage of the class)
- Your performance score will be 0 if you failed the testcase.

(3) Readability of HSPICE code (10%)

(4) The report (20%)

(5) Bonus (10%)

- 1 test case, the bonus score will be 10 if all correct, otherwise 0.

Please submit your assignment on time. Otherwise, the penalty rule will apply:

- Within 72hrs delay: 20% off
- More than 3 days: 0 point

If you have questions, please E-mail to both me (andyygchen@ee.ncu.edu.tw), and TA 施奕瑄 (ruby68680@gmail.com)