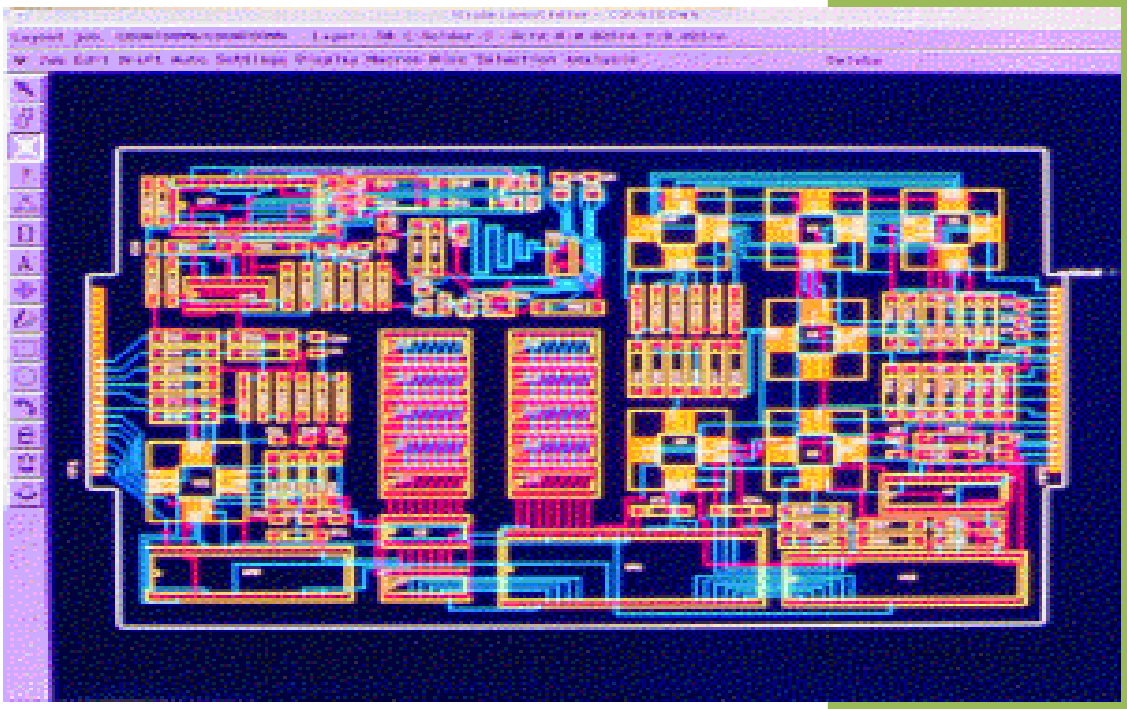
Computer Engineering Department
Faculty of Engineering
Cairo University

Spring 2021
CMP 305
VLSI

DCNN Accelerator (I/O Team)

2021



VLSI Semester Term Project
I/O Team

VLSI Project

Computer Engineering Department

Objectives

- To better understand Digital Design Flow
- To be proficient at VHDL
- To create real world hardware applications
- To understand the mapping between Algorithm Specifications & Hardware Implementation
- To understand Design Trade-offs
- To learn how to optimize Hardware Designs

Introduction

In real world IC designs, massive amount of data is transferred from and to an IC (ex: processor). Most of the current designs use serial communication protocols (as SPI, I2C, USB), however; there still some designs that use parallel communication protocols (as PCI). you will build the interfacing between the main chip and the CPU as discussed in the overview project description.

In this project, you will design a detailed low-level design of an I/O chip interface that uses parallel communication protocol and compression to pass the testing image and the CNN classifier info. The loaded data is saved in a RAM to be processed by the main chip logic. You will also experience the whole cycle of Design & Fabricating a system on chip, best described as in Figure 1.

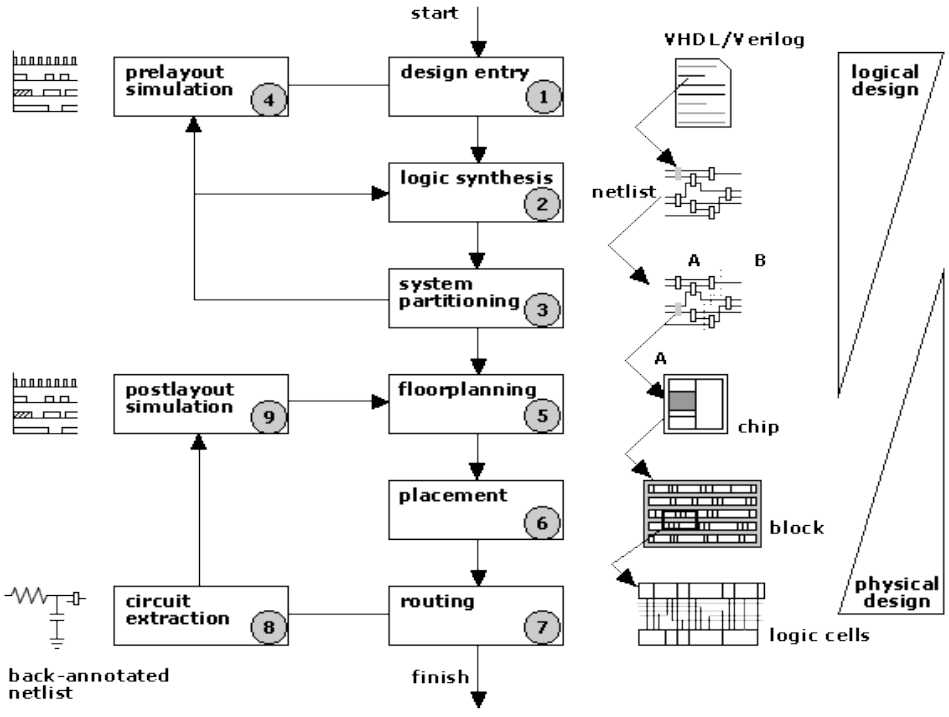


Figure 1: ASIC Chip Design Flow

I/O Scenario

Apple company has assigned your team for designing & implementing DCNN accelerator IP to be added into IPhoneX12s. Since I/O interface needs 2 parts (the sending "source" and receiving "destination"), the source is the PC's CPU that reads the input image & the CNN classifier info and send it to the I/O module (the destination).

The CPU compress the input image (BMP file) and the CNN classifier (JSON file), then send them over a 16-bit parallel bus. Moreover, the CPU uses additional control signals to control the chip (for example: an interrupt signal to notify the chip that a new command will be sent). The CPU sends control signals & data, and receives status signals & result. The I/O module receives the CPU commands & data, decompress the image & CNN, and pass the commands to the rest of the chip.

The CPU control signals includes:

1. System clock: Controls the chip frequency
2. Reset signal: Restart signal that initialize the chip for processing
3. Interrupt signal: A notification for the chip that a command is sent
4. Load/Process signal: A command signal that choose between loading a new data or processing the loaded data.
5. Image/CNN signal: A control signal that chooses the loaded data type (loading image or loading CNN)

You can add any additional signal you need to allow the CPU control the chip (justify the additional signals).

The working scenario should be as follows:

1. The CPU generates clock signal for the chip (constant frequency)
2. The CPU sends a reset signal to restart the chip (pulse signal for 1 clock cycle)
3. The CPU sends Load CNN command + interrupt signal to notify the chip of the command
4. The CPU sends the compressed CNN
5. The I/O sends done loading signal to the CPU
6. The CPU sends Load Image command + interrupt signal to notify the chip of the command
7. The CPU sends the compressed Image
8. The I/O sends done loading signal to the CPU
9. The CPU sends process image command + interrupt signal to notify the chip of the command
10. The I/O passes the command to the chip
11. The I/O passes the done processing signal + result from the chip

Design Requirements

You are required to build a detailed hardware design for the I/O module & integrate your module with the rest of the bigger team. The system is built for sending/receiving data over a 16-bit bus interface with the CPU as shown in figure 2 (IO module is the green box).

Additionally, your team will prepare the software scripts required to prepare the data for testing (compressing the image) and the test-bench for the whole chip (Tcl file for loading the data & reading results).

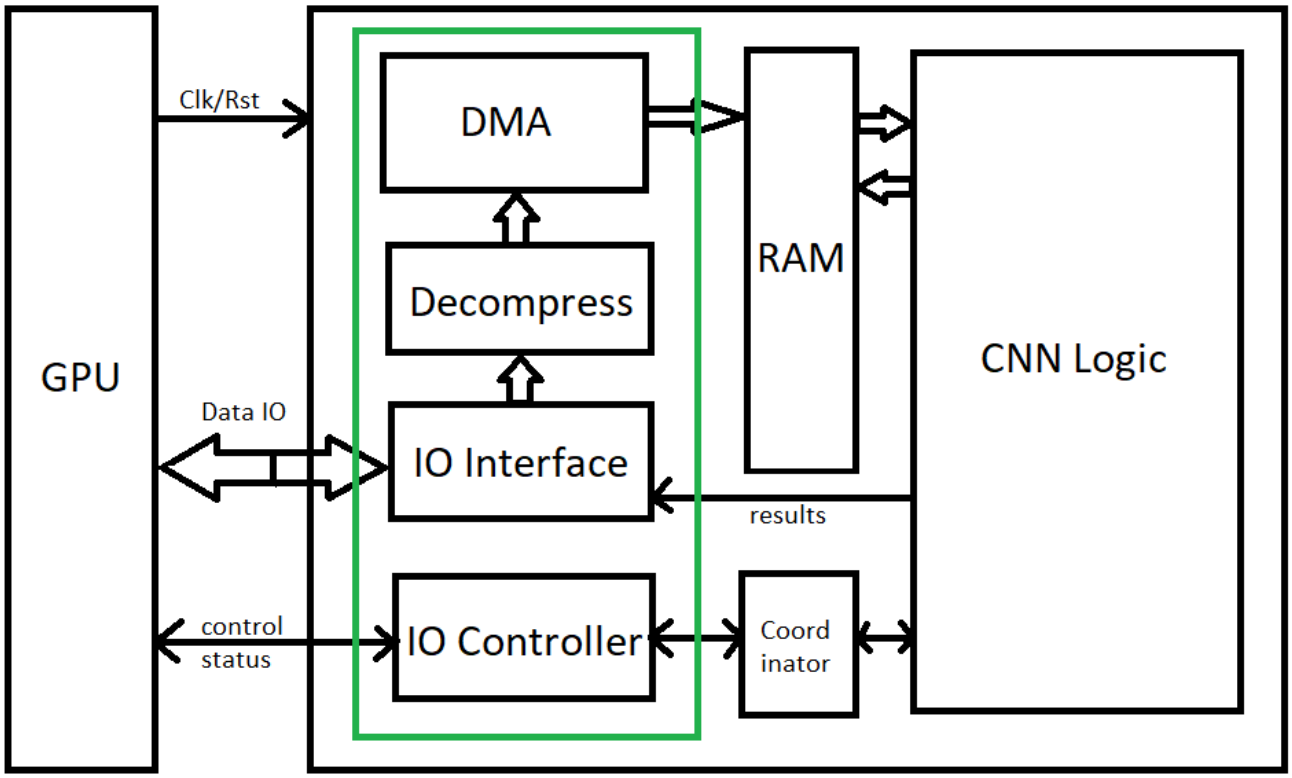


Figure 2: A simplified system design

Note that the RAM/Coordinator are common modules that is used-by/control the 3 main modules in the chip, so you need to coordinate with the rest of the team.

Take into consideration that your design will be implemented in VHDL and synthesized in the next phases. So try to make the design as clear as possible to easily implement it. Prior optimization is a plus.

Compression/Decompression Algorithm

The image is in a bitmap file format (BMP). For compression & decompression you will used bit-level Run-Length Encoding (RLE) algorithm. The compression step is implemented as a software script, while the decompression step is implemented in your decompress module. Additionally, you will need to write a Tcl file (test-bench) that reads the compressed file & send it to the chip.

Compression step is done by the following:

1. Read the raw image file (BMP file)
2. Convert the pixels from integer based numbers into binary numbers
3. Loop on the image row by row & compute the RLE compressed row (bit-level RLE)
4. Save the compressed file to be submitted to the chip in runtime

Submitting the image to the chip is done by the following:

1. Open the compressed file
2. Read Line by line
3. Split the line into the input databus size (16-bit)
4. Generate the control signals required to load the image to the chip

Decompression step is done by the following:

1. Wait for the load image control signals
2. Receive the compressed image line by line
3. For each line, compute the original row using RLE decompression
4. Save the original row in the chip RAM in a specific place

System Specifications

The system should have the following I/O ports:

Port	Direction	Size
Clk	IN	1 bit
Rst	IN	1 bit
Interrupt	IN	1 bit
Load/Process	IN	1 bit
CNN/Image	IN	1 bit
Done	OUT	1 bit

Din	IN	16 bits
Dout	Out	4 bits

References for more Informations

1. https://en.wikipedia.org/wiki/Run-length_encoding
2. https://www.dcode.fr/rle-compression
3. https://www.fileformat.info/mirror/egff/ch09_03.htm

Rules & Regulations

- I/O subteam is 3-4 members.
- You are free to design your system as you want as long as you perform the required functionality.
- You could add any additional I/O ports and modules according to your need, but you have to justify your design choices.
- Your design should be modifiable to meet the design constraints in the next phases.
- Your design should be integratable with the rest of the bigger team.
- Take care that your design is logically mappable to hardware or you will have to repeat it all again.
- Open your mind and don't limit yourself .
- You are not allowed to copy from any external resources in your implementation .
- You are allowed to consult external resources for Design but Do your OWN & you have to fully understand it.
- **Grades are based Mainly on Individual work + your team work . if you didn't work and the project was complete you will still get a zero grade. And we really mean it.**
- The document is variable to change with a previous notification.

Deliverables

- Hardware design documents:
  - Your names
  - Detailed units architecture with connections.
  - Detailed sub-units design in the well-known logic gates Level (counters / adders / ...).
  - Design assumptions and limitation
- A compression script that compress the image file
- A Tcl file that sends the compressed image file & CNN file over the 16-bit databus.