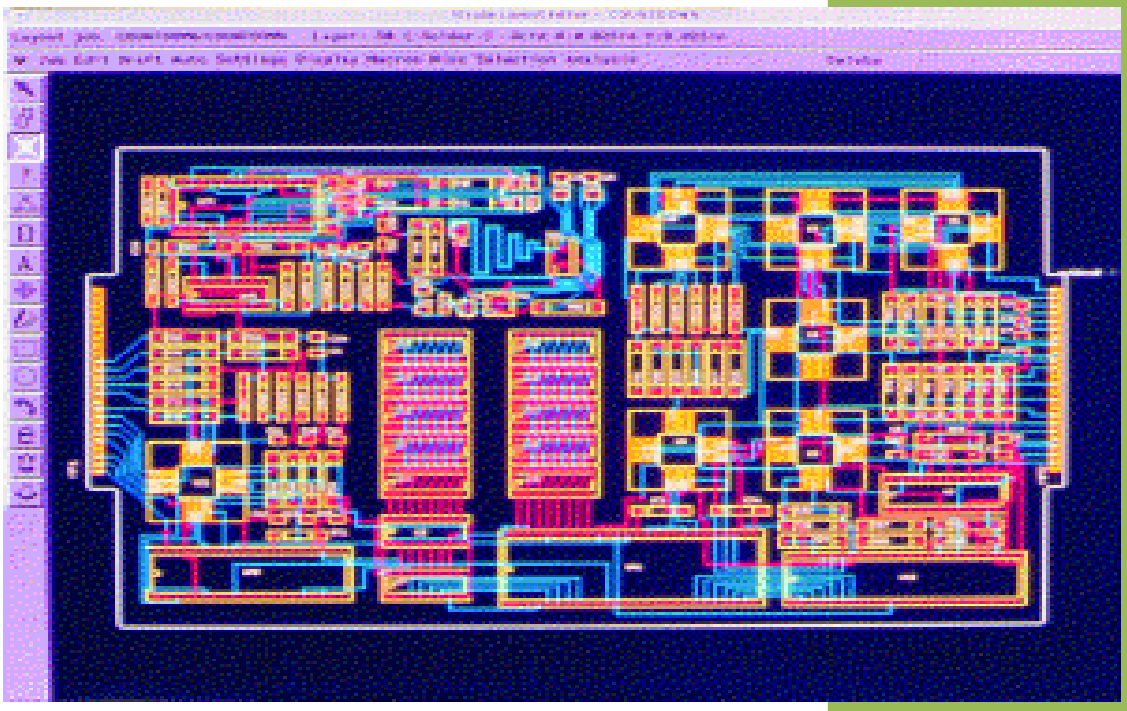
Computer Engineering Department
Faculty of Engineering
Cairo University

Spring 2021
CMP 305
VLSI

DCNN Accelerator (FC Team)

2021



VLSI Semester Term Project

FC 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

Despite CNN core is convolution/pooling layers, additional layers are used as dropout, fully-connected network and soft-max layers. In this project, you will design a detailed low-level design of fully-connected layer and max layers.

You will also experience the whole cycle of Design & Fabricating a system on chip, best described as in Figure 1.

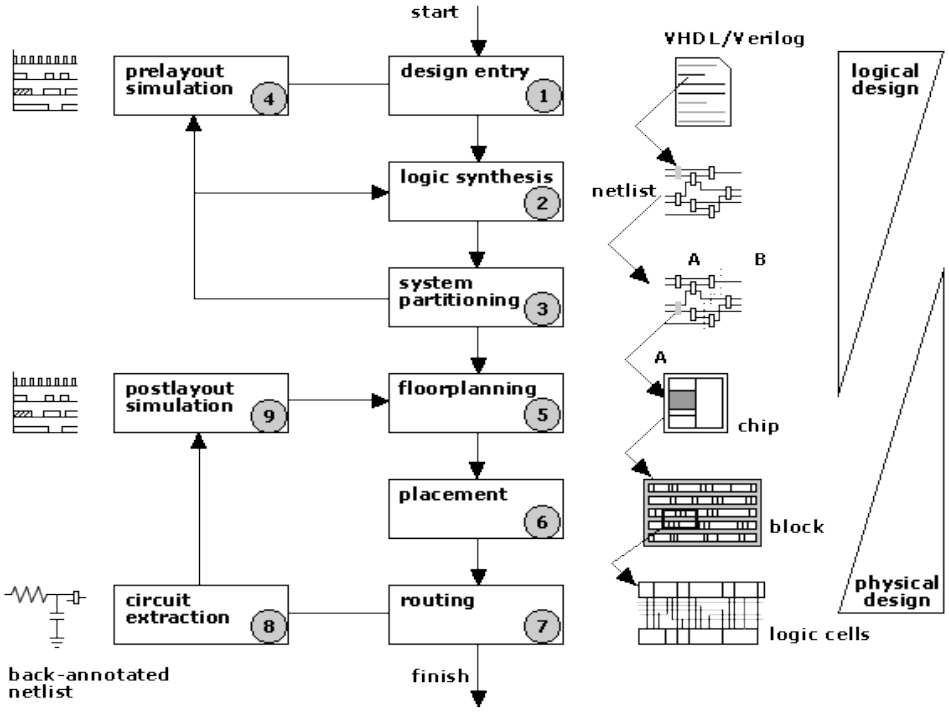


Figure 1: ASIC Chip Design Flow

FC Scenario

Apple company has assigned your team for designing & implementing DCNN accelerator to be added into IPhoneX12. After Convolution and pooling layers are applied (locally & high-level analysis), fully-connected network is applied to analyze the whole image (globally) features extracted from the CNN layers. Fully-connected layers are applied by matrix multiplication of the last CNN layer output & adding a bias. Finally a softmax layer chooses the highest probability label for the image.

The FC layer steps are simple:

1. Loop on all the neurons outputs from the last CNN layer
    a. For each cell multiply it with its corresponding weights
    b. Accumulate result + bias
2. Repeat the loop for each neuron in the FC layer
3. Select the max of FC layer as the output

Note that a single FC layer is still costy. If the last CNN layer generated a 10x10 neurons, then you have a 100 weights (multiplication operation) for each neuron in the FC layer. Additionally, you need to use a high fixed size precision for your fixed-point operations.

Radix-2 Booth's Multiplication Algorithm

Binary multiplication uses shifting operations to compute the multiplication of 2 n-bit **signed** numbers. It could be implemented in a single-cycle (Shift-Add Approach) or multi-cycle (Booth's algorithm) approaches. **You will use Radix-2 Booth Algorithm in your mini ALUs**.

1) Assuming that the 2 operands are called M and R
2) Set values for A, P, S
   Where A is a (2n+1) bits & the highest n bits are set to be M (ex: n=4 A="MMMM00000")
   And S is a (2n+1) bits & the highest n bits are set to be -M (ex: n=4 S="(-MMMM)00000")
   And P is a (2n+1) bits & the bits from (1 to n) are set to be R (ex: n=4 P="0000RRRR0")
3) Loop for N iterations:
    a) Check the Least Significant (Rightmost) 2 bits in P
        i) If LSB == "00" or "11"
            (1) Do Nothing
        ii) If LSB == "01"
            (1) P = P + A
        iii) If LSB == "10"
            (1) P = P + S
    b) Arithmetic Shift P 1 bit to the right
4) Result = P[1 to 2n] (Ignore the least significant bit)

Fixed Point Arithmetic

Fixed points is a binary representation for fraction numbers. The integer part is normal binary $\{2^4 2^3 2^2 2^1 2^0\}$ (increasing powers of 2). The fraction part is a decreasing powers of 2 $\{2^{-1} 2^{-2} 2^{-3} 2^{-4}\}$.

i.e: 1011(-2) ⇒ the number in the register is 1011 & the fraction is the least 2 bits
10.11
$= 1*2^1 + 0*2^0 + 1*2^{-1} + 1*2^{-2}$
= 2 + 0 + ½ + ¼ = 2.75

Normal binary arithmetics (as addition) are applied on fixed point number of same precision (number of bits in the fraction part), however; shifting is needed for different precision numbers.

i.e: 1.125 + 3.5 = 4.625
== 1001(-3) + 111(-1)
== 1001(-3) + 11100(-3)
== 01.001 + 11.100 = 100.101

Additionally, The multiplication and shift operations can be used similarly to the binary operations by adjusting the point position.

i.e: 1.125 * 3.5 = 3.9375
== 001.001 * 011.100 = 000011.111100
== 1001(-3) * 11100(-3) = 111111(-6)
3-bits for integer + 3-bits for fraction ⇒ 6-bits for integer + 6-bits for fraction

References for more Informations

1. https://en.wikipedia.org/wiki/Booth%27s_multiplication_algorithm
2. https://en.wikipedia.org/wiki/Binary_multiplier
3. http://www.geoffknagge.com/fyp/booth.shtml
4. http://www.ecs.umass.edu/ece/koren/arith/simulator/Booth/
5. http://cs231n.github.io/convolutional-networks/#fc
6. https://en.wikipedia.org/wiki/Fixed-point_arithmetic
7. https://spin.atomicobject.com/2012/03/15/simple-fixed-point-math/

Design Requirements

You are required to build a detailed hardware design for the FC module. The system is built for applying fixed-point operations (multiplication/addition/comparison) in order to apply a fully-connected network layer and select the max label. The FC module (green box) overview is shown in figure 2. The module should be designed to optimally compute a variable number of neurons in a FC layer with a variable number of inputs, then generate the label for the max neuron value.

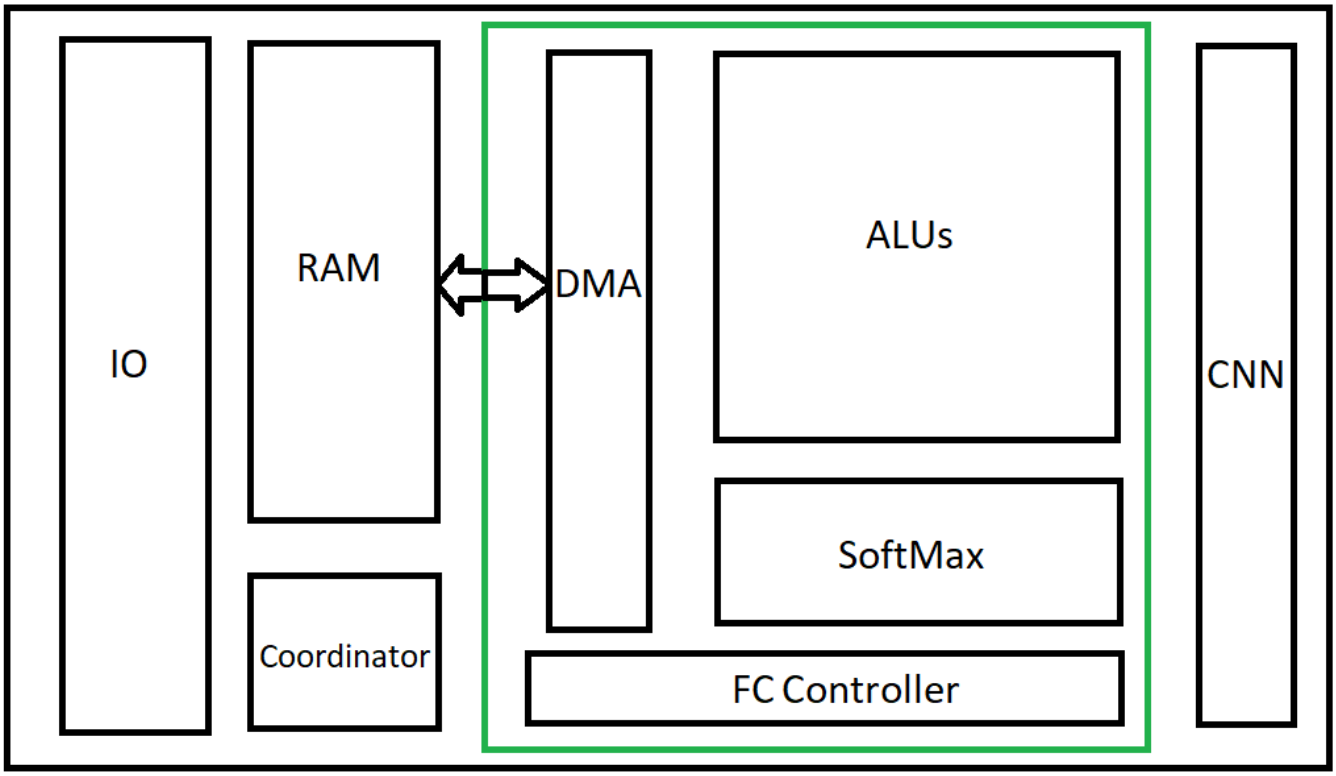


Figure 2: A simplified system design

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.

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.
- You are not allowed to use the multiplication "*" operator.
- 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