***Faculty of Engineering***

***Computer Department***

***VLSI Project***

***-----------------------------------------------------------------------------------------***

***I/O Module***

***Phase 2***

***Team 2***

***Names:***

***1-Ahmed Salah.***

***2-Khaled Sabry.***

***3-Mahmoud Youssri.***

***4-Mohamed Abdel-Aziz Ebrahim.***

***File Tree :***

IOCHIP🡪 CPU,processenablecircuit , loadenablecircuit , LDONEcircuit , ramenablereadwrite , RamControl , ADDRESSGENERATOR , decompressor , jsondecompressor and MYMUX.

CPU🡪MEMORY.

Processenablecircuit 🡪 Rigester and Gates.

Loadenablecircuit 🡪 Rigester and Gates.

LDONEcircuit 🡪 Gates.

Ramenablereadwrite 🡪 Gates.

RamControl 🡪Gates.

ADDRESSGENERATOR 🡪 FullAdder and Rigesters.

Decompressor 🡪 Counters , Rigesters and Decoder.

Jsondecompressor 🡪 Counters , Rigesters and Decoder.

MYMUX 🡪 Gates.

***Image encoding and Compression:***

- **Input**: image gray level 28\*28.

- **output**: compressed binary text file.

- the image is gray level (0-255) so we needed 8 bits to define the color of the pixel.

- as the data bus is 16 bits, we used the other 8 bits to know how many pixels are similar next to each other.

- we begin calculating the pixels from top left row to top right row then calculate again second top left row to second top right row and so on.

- the compression is done on the byte level.

- the output file each row in it contains 16 bits 0’s and 1’s rows.

**CNN Json encoding: -**

- **Input**: CNN json file.

- **Output**: two files : biString = each row contains 16 bit and each row represents value for each object in the json file.

: idString = identify the meaning of each row in the output biString.

- we take from json file.

1) convolution layer (sx, in\_depth, out\_depth, out\_sx, l1\_decay\_mul, l2\_decay\_mul, pad, layer name, biases {depth, weight ‘w’}, filters weights).

2) pooling layer (sx, in\_depth, out\_depth, out\_sx,in\_sx ”convlayer out\_sx”, pad, layer name).

3) fully connected layer (num\_inputs, biases{weight ‘w’}, filters weights).

- we add also (num of layers, filters count for convolution layer,in\_sx in pool layer which represents out\_sx from convolution layer).

***Design Entites for I/O :***

**1-The load enable circuit :**

**-Job :**

To inform the chip that it is in the loading mode as it generates a signal load that remains one during the whole loading .

**- The circuit design :**

**2-The Process enable circuit :** **-Job :** To inform the chip that it is in the Processing mode as it generates a signal process that remains one during the whole processing and this signal is sent to the modules of CNN and FC .

**- The circuit design :**

**3-LDONE Circuit:**

**-Job:**

Generates a load done PULSE to be sent to the cpu.(ORed with the PDone to generate Done )

**- The circuit design :**

NOTE: The upcoming 3 circuits combined are considered as DMA

**4-Address generator :**

**-Job:**

This circuit tells the address to be used by the the ram to read or write. It allows to choose between the next address and an outer address decided by the process modules.

**- The circuit design :**

**5-Ram eanble reading and writing Circuit:**

**-Job :**

This circuit tells if we are to read from or write to the ram.

**- The circuit design :**

**6- Ram control:**

**Job :**

This file contains the 2 MDR registers 1 as input and the other as output as the ram can read and write at the same time. The file also contains the ram of the chip.

**- The circuit design :**

***Decompression Entity :***

**Demonstration:**

The decompression begins when the decompress signal is high , the first eight bits is the value which is propagated to 28 register each register carries one pixel value and the second eights bits is the number of adjacent pixels which carry same pixel value is the input to a counter this counter is responsible for counting the cycles needed to save pixels in the register when it finishes it makes read signal high to read another line from file and the load of counter signal is high . There is the register counter which is responsible with the mux to choose which register to save the current pixel value when all 28 registers carry pixel values of one row of the image the memory read signal becomes high and the register counter is reseted .

***Inputs and outputs :***

***Flow of the I/O chip :***

1-The cpu MUST send a reset signal to put the chip into idle mode, when ever the reset is set the deevice stops all of its work and return to the starting point.

2-The cpu synchronize the device by sending a clock signal.

3-Once the reset signal is removed the chip is removed the chip will be read to work but it will remain idle till recieving an interrupt.

4-On recieving the interrupt load enable circuit and process enable circuit starts to work to inform the device to enter the loading mode or processing mode (here it will enter the loading image mode).

5-The address generator will indicate the starting address of storing the image (0)

6-The cpu will start sending the compressed image on the 16 bit bus and each clock

Data is decompressed and once we have a whole row of the image ready ,it is stored at input MDR.

7-When the whole row of the image is ready and decompressed it is loaded into ram and then the address is incremented.

8-the previous loading steps are repeated tell the end of the image

9-when the data on data bus are equal to (0000 0000 1010 0001) , the image is loaded and the chip sends a Done pulse to the cpu.

10-Cpu will interupt the chip and send the CNN contents and they will be stored in the ram by the same manner explained above.

11-The chip will send a Done signal to the cpu by the same way.

12-The cpu will raise the process signal but the device will remain idle till recieving the cpu interrupt.

13-On recieving the interrupt the process enable circuit will decide to start processing.

14-The process enable signal is sent to the other modules to start processing.

15-On finishing the processing a Done signal will be sent to the I/O to send it to the CPU and the result is stored at the ram and then I/O sends it to cpu.

***Simulation :***

***For A small Test Case***