# EEDG 6306    ASIC Design

# MSDAP Architecture

### 1.1 Block Diagram

The architecture of MSDAP is given below.  For simplicity only one channel is represented.

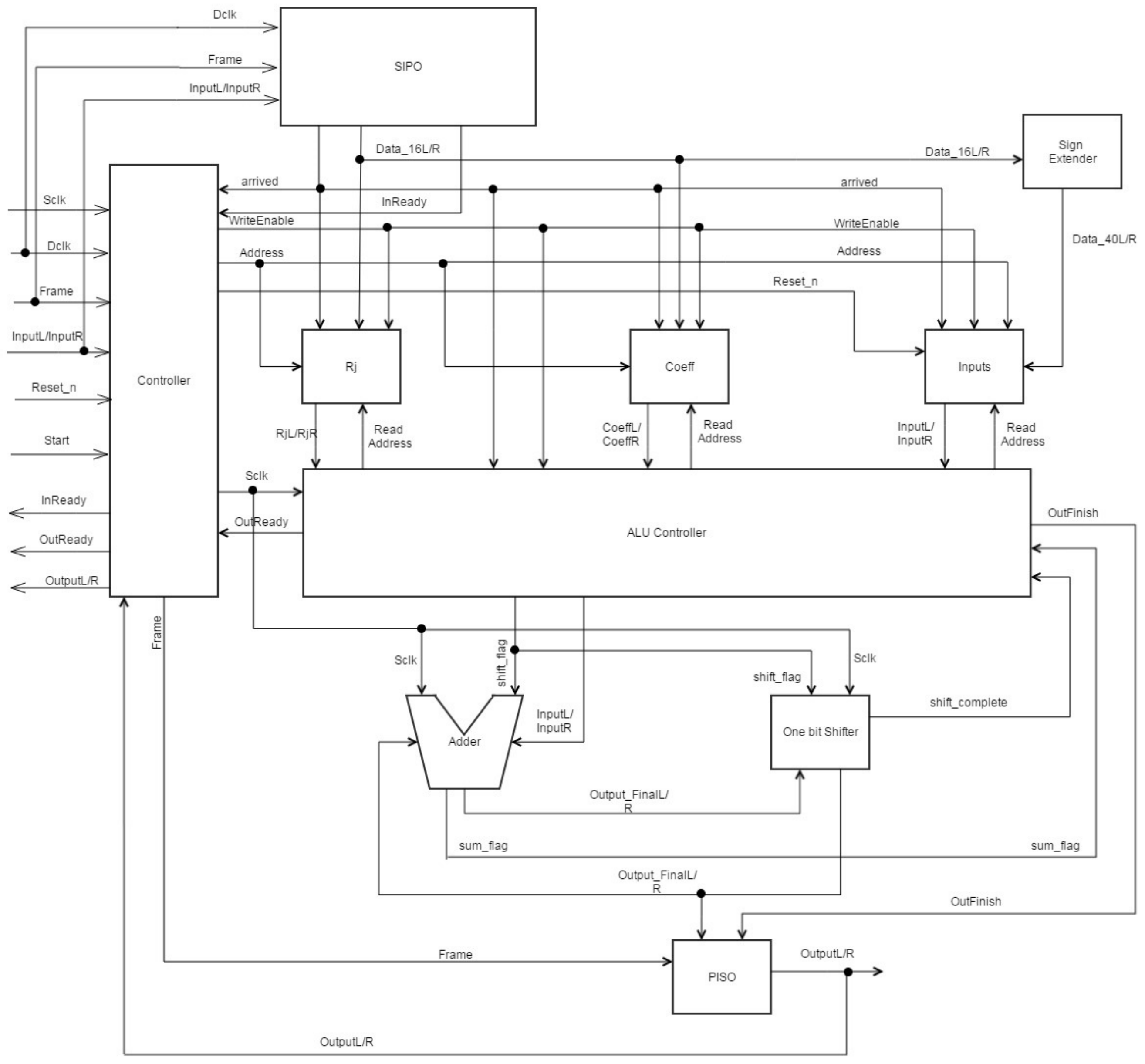

*Fig1. MSDAP Architecture*

## 1.2 ALU Block

The ALU Block controls the functionality of main blocks which perform the computation. This block operates in Sclk frequency and uses internal counters and logic in order to set certain flags which are used for performing various tasks of the entire convolution. The ALU block is responsible for receiving Rj, Coefficients and 40 bit Inputs from the memory and based on number of values in each Rj, the coefficients are accumulated and output is computed through series of addition and shift.

During the addition, ALU controller passes a memory address to memory blocks to receive the inputs. The computation begins when the first 16-bit input is arrived and involves accumulation where Uj's are either added or subtracted based on the Coefficient value, followed by addition of Uj to the output and shifting the output.

## 1.3 Time Budget and working

The system starts when Start signal is high and this signal is asynchronous with Dclk and Sclk. The computation starts as soon as one 16 bit input is received. The total time available for one output to be computed is 16 Dclks, after which MSDAP has to start computation for the second input.

In our design, Dclk = 768 MHz and Sclk = 26.88 MHz

Total number of Sclks in one Dclk = $26.88\ \text{MHz}768\ \text{KHz}$ = 35

Hence for one computation, total number of Sclks needed = 16 × 35 = 560

For the output computation, an adder is used for addition and accumulation of coefficients and one bit shifter is used to shift the accumulated coefficients by one bit. Loading of Rj, Coefficients and Inputs are done instantaneously at this stage.

Calculation to find out the computation time is as follows:

Total number of coefficients = 512

Time required to compute Uj(0) to Uj(16) = 512

Addition of Uj to Output and shifting is done in a sequential manner. When Uj(0) accumulation is completed, Uj(1) accumulation starts, and at the same cycle accumulated Uj(0) is added to the Output. In the next cycle, Output is shifted by one bit where in Uj(1) accumulation is still happening. Thereby saving the resources and time. All the above operation is with respect to Sclk.

Hence, total time required for computation = 512 + 1 + 1 = 514 Sclks.

The additional two Sclks are to add Uj(15) to output and shift the output by 1. We have performed optimization techniques to reduce the computation clocks. Now that we have saved 46 Sclks, we can reduce the Sclk frequency from 26.88 MHz to 25.344 MHz so that it consumes less power. Another method is to use 16 adders, which would parallely accumulate all the 16 Uj to increase the computation speed drastically for extra hardware tradeoff.

## 1.4 Functionality of each block and Signals used

### Serial In Parallel Out (SIPO)

SIPO converts the input serial data to parallel output data. The inputs are read  at the frame signal which is aligned to Dclk. The 16 bit inputs are read bit by bit and its converted into parallel input data of 16 bits (Data_16L/R). A flag (Arrived) is set to indicate the arrival of all 16 bits for a particular input. This flag is reset on next Dclk when first bit of next input arrives. InReady signal is set by SIPO to indicate that MSDAP is ready to receive inputs Rj, Coefficients and Input samples.

### Encode & Sign Extender

This block is used for encoding and sign extending the input samples to 40 bits. This is done by assigning bits [15:0] to InputL/R[31:16] and then sign extending by assigning InputL/R[39:32] to 0/1 based on MSB of the received input. The array which stores the input samples are initialized to zero, so zero padding bits InputL/R[15:0] need not require a separate hardware. Input to this block is Data_16L/R and output are encoded and sign extended data of the name Data_40L/R.

### Memory Unit for Rj, Coefficients and Inputs

The memory unit stores the input data. It stores up to 16 16-bit words of Rj, 512 16-bit words of coefficients and 40-bit sign extended Inputs. This block is controlled by signals WriteEnable, Address, Data_16L/R, arrived. The main controller sends the required control signals for this block. When WriteEnable is 00, the data Data_16L/R is written to Rj memory. When WriteEnable is 01, Data_16L/R is written to  Coefficients memory. When WriteEnable is 10, the Data_16L/R is encoded and sign extended to 40 bit Data_40L/R and then stored in the memory. The address where the inputs are stored are sent by the main controller. When Reset_n goes low, only the 40-bit Input memory is cleared.

### ALU Controller

This block provides control signal for blocks like Adder, One bit shifter, PISO and memory and works based on Sclk. The computation starts when WriteEnable is 10 and inputs are received (arrived is 1). The inputs are accessed from the memory using the address. This block is responsible for setting flags shift_flag and OutReady.

### Adder

This block adds two operands with respect to Sclk. Accumulation of Uj and addition of Uj's are performed in this block. The output of this block is fed back to the same block for adding all the accumulated Uj's for a single output. A flag sum_flag is set high when addition is complete and this flag is sent to the ALU controller.

### One Bit Shifter

This block is controlled by the signal shift_flag. When shift_flag is high, the added output Output_FinalL/R is shifted to right by one bit. A flag shift_complete is set high and sent to the ALU controller indicating that shifting operation is complete, so that ALU can clear the shift_flag to start adding the Uj in the next cycle.

**Parallel In Serial Out (PISO)**

This block converts the parallel computed output data to serial data. This block is controlled by signals Outfinish and Frame. The computed data Ouput_FinalL/R which is 40 bit long are transmitted bit by bit when Outfinish is set high and Frame is set low.

## 1.5 Difference between our design and the classic design

In the classic design discussed in the class, the memory used for storing Rj, Coefficients and Inputs are 16 bits. The Inputs have to be sign extended every time it is accessed. As the received Input samples are large in number, total number of sign extension that will be done is very large. This can be avoided by using a different memory design.

In our proposed architecture, memory to store Rj and Coefficients are 16 bits wide (word length) and memory to store Input is 40 bit wide (word length). The Inputs are sign extended and then converted to 40 bits before storing it in the memory array thereby ignoring sign extension every time the input is accessed.