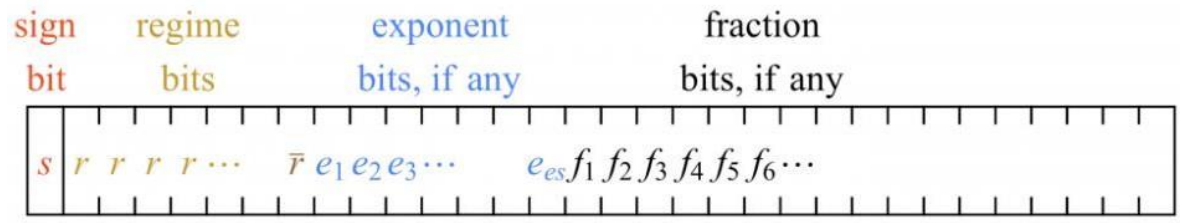
POSIT ALU ASSIGNMENT

AMBESH DIXIT

210102122

POSIT Number system :



For this assignment I have taken 1 signed bit, 2 regime bits, 1 exponent bit and 4 fraction bits. This makes it an 8 bit POSIT number.

To read this POSIT in decimal we use :

Decimal Value = Sign × Regime × 2^Exponent × (1+Fraction/2^4)

Here:

- Sign is either -1 (if the sign bit is 1) or 1 (if the sign bit is 0).
- Regime is calculated based on the 2 regime bits.
- Exponent is determined by the exponent bit.
- Fraction is the binary value represented by the 4 fraction bits.

Here based on the specification of POSIT number system chosen the range of output will be :

$$4 \times 2^1 \times (1 + 15/16) \approx 15.9375$$

So [-15.9375, 15.9375] will be the output range.

IMPLEMENTATION OF ALU :

This POSIT ALU was implemented by first converting the input 8 bit POSIT Number into 16 bit binary representation where, 1 bit is used of sign, next 5 bits for integer part and rest 10 bits for the fractional part.

All the Arithmetic and Logical Operations were than performed in this 16 bit binary representation. At the end the resulting 16 bit number was converted back into 8 bit POSIT format using suitable conversions.

A 3 bit select line will choose the operation to be performed.

5 arithmetic and logical operations were chosen to be implemented :

- 000 = Addition
- 001 = Subtraction
- 010 = Multiplication
- 011 = Bitwise AND
- 100 = Bitwise OR

The complete project was implemented in 5 different modules where each perform a different task :

- alu.v
- posit_binary.v
- binary_posit.v
- floatAdd.v
- floatSub.v
- floatMult.v

The alu.v is the master module which takes in 2 POSIT numbers and select as input and gives a single 8 bit POSIT number as output. This module make instantiation of all the other modules inside it.

floatAdd.v, floatSub.v and floatMult.v takes in two 16 bit floating point inputs in the format specified above and perform the respective operation and return a 16 bit output. All these 3 modules are coded in a way to make sure it follows the rules of arithmetic on floating point numbers.

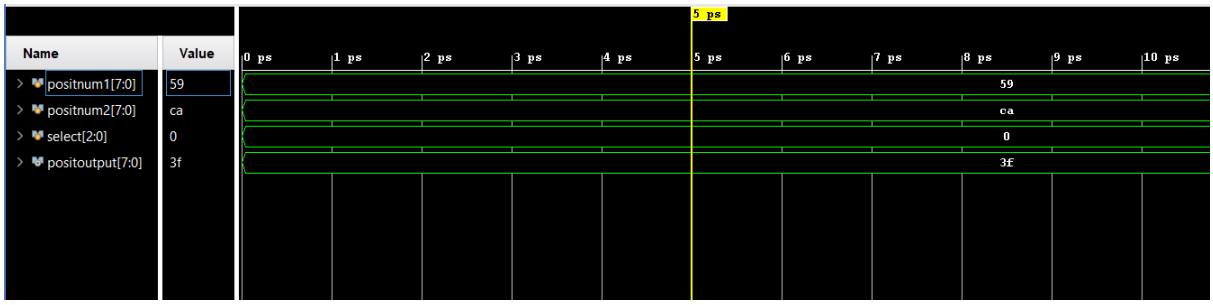
The posit_binary.v module takes in two 8 bit POSIT numbers as input, operates on them and gives two 16 bit floating point numbers as output which are the 16 bit binary representation of POSIT number in a specific format. After performing the required arithmetic or logical instruction on these two 16 bit floating point number, we instantiate the binary_posit.v module which converts the resultant 16 bit binary number back into original 8 bit POSIT system.

RESULT :

The two POSIT INPUTS given were

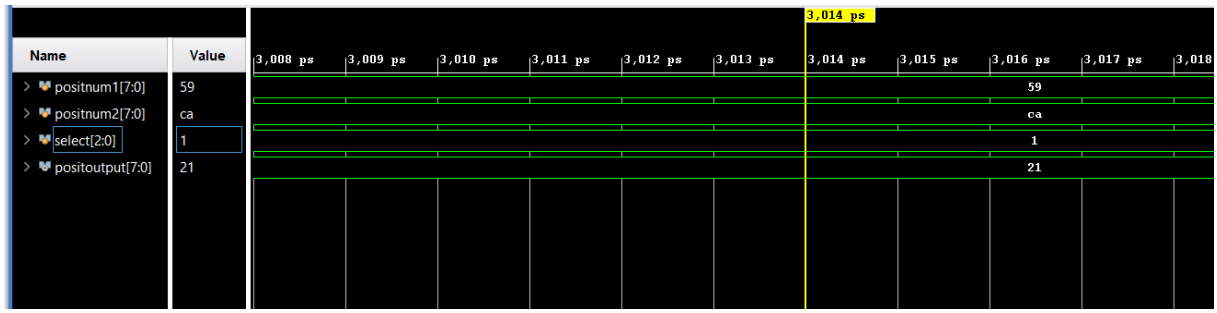
- positnum1 : 8'b01011001
- positnum2 : 8'b11001010

Select = 000 : Addition



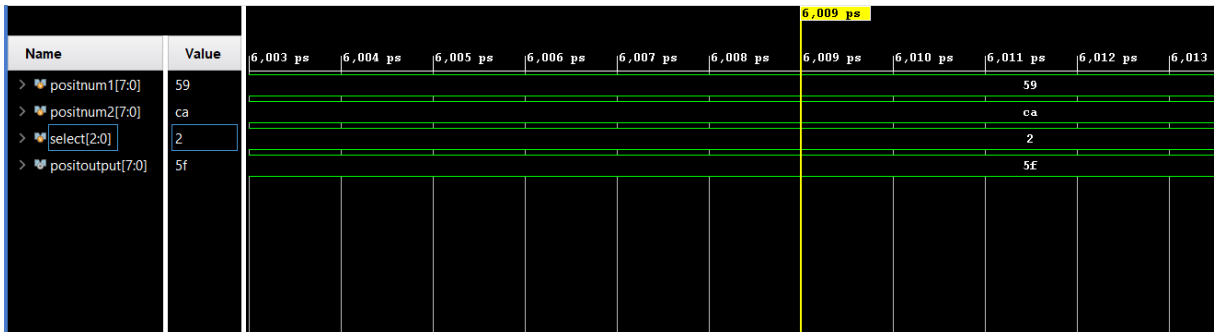
positoutput : 8’b00111111

Select = 001 : Subtraction



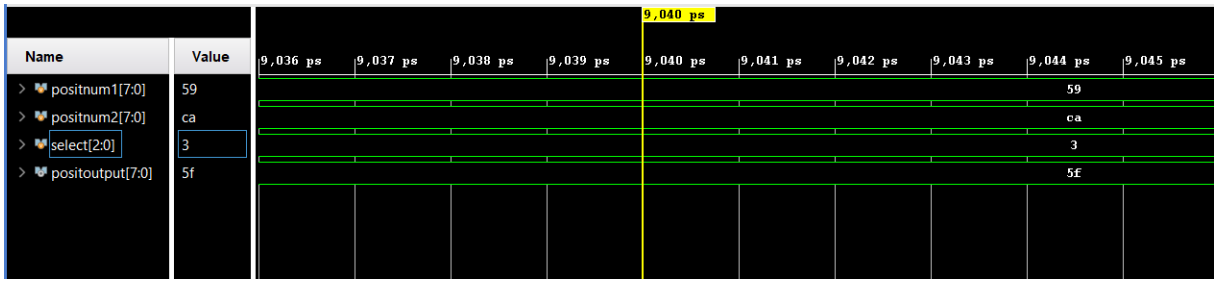
positoutput : 8’00100001

Select = 010 : Multiplication



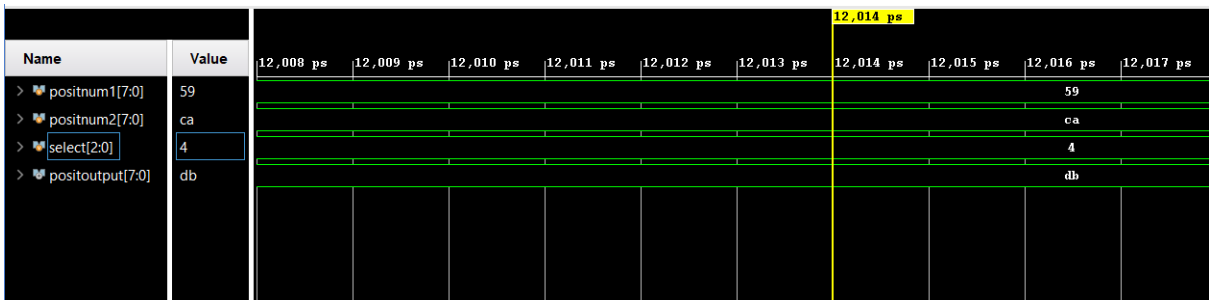
positoutput : 8’b01011111

Select = 011 : Bitwise AND



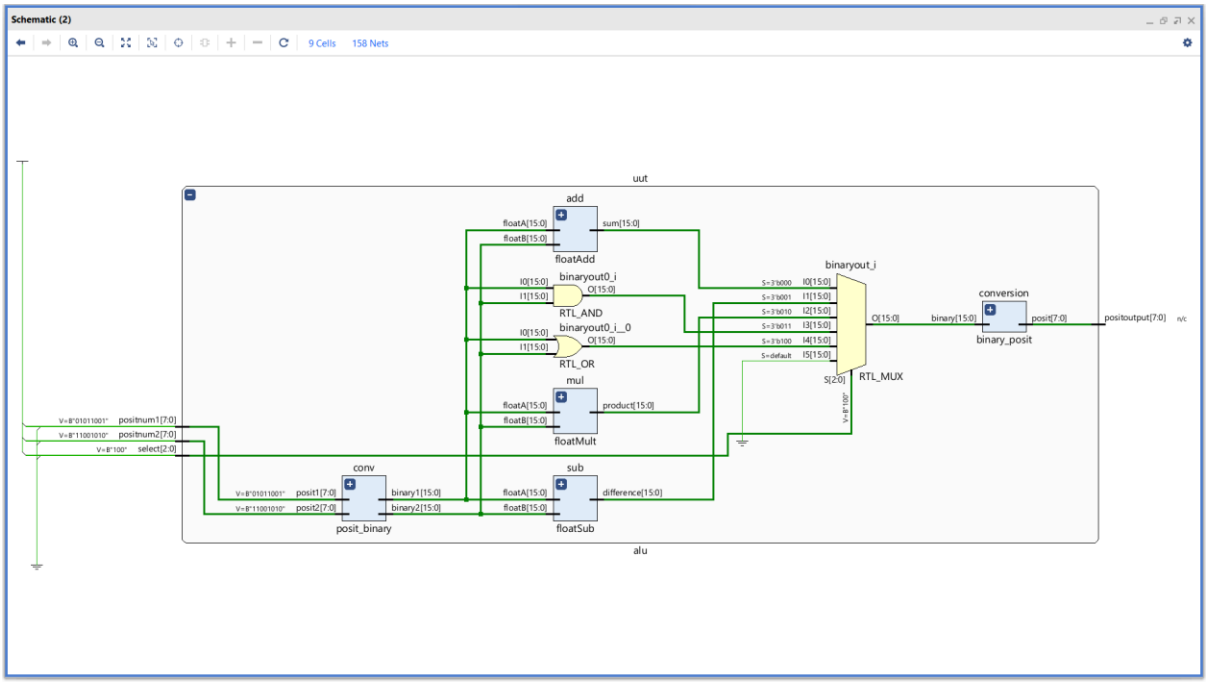
positoutput : 8’b01011111

Select = 100 : Bitwise OR



positoutput : 8’b11011011

DESIGN ARCHITECHTURE :



All the different modules can be clearly seen in the RTL schematic.

The master module alu incorporates every other module inside it.