**Course: EEEN 5330-00\_ Rapid Prototyping and ASIC Design**

**Term: Spring 2017**

**Topic: SINGLE PRECISION FLOATING POINT MULTIPLIER**

**Last Name: THOTAKURA**

**First Name: ANIL KUMAR**

**K#: 00381028**

**Electrical Engineering and Computer Science**

**Texas A&M University at Kingsville**

**05/01/2017**

**Introduction**

Floating point multiplication is extensively recycled in broad set of experimental and noticeable processing computation. In these calculation multiplication is one of the better used athematic operation in these computations. Further the demand of huge speed multiplier expanding as the of high speed processors are expanding. Large throughput arithmetic activity is crucial to manage the aspire to achievement in more actual time salient and image processing application. Multiplication is the one of the major application in such kind of operation, over the decades the advancement of quick multiplier circuits is processing with concern. It also curtails the power utilization and time postponement.

To perform the real numbers in binary format floating point numbers are one of the achievable way. Usually there are two types of performing the floating-point values. There are single precision and double precision. Now this report contains about the single precision. It contains one bit sign represented by S, a twenty-three-bit fraction denoted by M or Mantissa, and an eight bit exponent sows by E. To get the significand an additional bit is combined to the fraction. If the proponent is exceeded than 0 and lesser than 255, and for a normalized number the significand should contain 1 as the MSB bit.

|  |  |  |
| --- | --- | --- |
| S | E=Exponent | M=Mantissa |
| 1 bit | 8-bits | 23 bits |

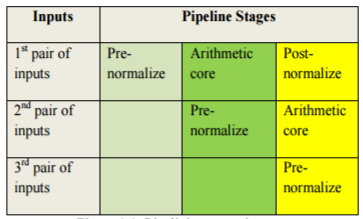
Z=(-1^S) \*2^(E-Bias) \*(I.M)

Bias=127

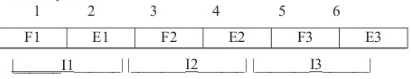
Therefore Value= (-1^sign bit) \*2^(Exponent\*127) \*(I\* mantissa)

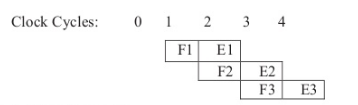
**Floating point Multiplier**

The architecture of multiplier for floating point numbers is expected and planned. So, the proposed floating point numbers In this entity, the inputs are two 64 bit operands, four rounding mode, enable and restart. And the outputs present in this entity are 64-bit output and a ready and this ready goes huge when output is enable. The operands are convert into composition which compose them clear and active to maintain internally which means the pre-normalize. Addition, subtraction or multiplication are the essential arthematic operations done by arthematic core. The result will be normalized and converted in post normalization. The pipelining approaches is mainly belongs in these three stages, initially inputs coming in to the pre-normalizing stage after these are transmitted in to arthematic core after that on the pre normalizing state other inputs are ready to initialize that is it is ready to serve next inputs this process is done circularly.

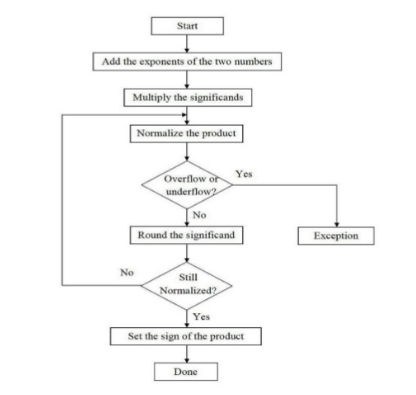


The acceleration of beheading of any direction can be diverse by a statistic of approaches like, accepting a agile automation to frame a processor or to organize the plumbing in a approach so the different activities can be achieved at concurring. Depending on pipelining various actions can be done together without developing the execution period of a guidance. Where the fetching block is denoted by F, and the execution block is represented by E. We can save more clock cycles by improving the instruction count.





Block diagram



This is the basic structure of floating point number first it starts and adds the exponents of the two numbers respectively and after that multiply those significands then normalize the products of the significands then the result shows either overflow or a underflow if it gets overflow then its gets exception if it does not get any overflow or underflow then round the significand and still try to normalize and do the same thing as done before up to get overflow after completing all this things set a sign to the result then the process is done successfully.

**Design entry**

The appropriate circuit has to be express in a VHDL code, name given to the vhdl entity has to match the project name. By using a text editor or by using a Quarts II text editing facilities this code can be copied in to a file and it can be titled with any name but it must hold with an extension vhd. which shows a vhdl file. That analyzed VHDL code arrange the circuit and provoke an extertion of it for the target chip. those tools are composed by compiler. now start the compilation it moves to several steps, after it fineshes it produce a compilation report if there are any errors circuit it pop ups an error message which shows the number of errors after that we have to rectigy those errors until without getting any errors while compiling.