Phase 1 Report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Slack | Data arrival Time | Area | Power (nW) | Algorithm |
| 41146 | 854 | 69 | 114.152771 | Fixed Point Addition |
| 40997.2 | 1002.8 | 1288 | 1995.057007 | Fixed Point Multiplication |

* We have used the built-in addition and multiplication sign without any implementation as permitted at this phase

1. Assumption:

* Constant scale factor for inputs and outputs equals 7-bits.

1. Limitations in multiplication:

* To be in safe zone, Number of bits in integer parts equal 6- bits, So I can take the result 12 bits and doesn’t occur overflow.

1. Validation:

* Overflow flag set to ‘1’ if there is overflow.
* In multiplication, Overflow doesn’t occur, if the sign bit extend in output in all the rest of bits on the right .

## Tasks

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Task | Hours spent | Problem faced |
| Ahmed Hamdy | Addition | 7 | Dealing with –ve numbers in vhdl how it's work with 2's complement or 1's complement |
| Ayat Mostafa | Multiplication | 14 | How to handle overflow when it occurs in multiplication |
| Mohamed khier | Searching for the algorithms |  |  |
| Abdulrahman khaled | Testing | 30 | Python code was very had and got all of my time and after the team changed the negative number to be 2’s complement I deleted all my work and made another c++ code, and the problem was in the multiplication in very corner cases it got 1’s comp of the result and not having enough time after the modifications |

# Addition design

A 16 bit

ADD

MUX

16 bit result bus

16 bit output

B 16 bit

0

overFlow

output[15]

B[15]

A[15]

1

Or

Not

Not

Not

And

And

* Design for Multiplication:-

INPUT2 (16-Bit)

INPUT1 (16-bit)

Comparator

MULTIPLY

Result (32-bit)

Output (16-bit) overFlow