ISSUES

No	Issues	Solutions
1.	ADD/ SUB: Addition/Subtraction require Post-Normalization for which Synthesizable algorithm must be developed.	LZD can be used for this purpose. MATERIAL: https://www.researchgate.net/publication/324254446_An_Efficient_Base-4_Leading_Zero_Detector_Design
2.	DIV: No Circuitry is available which can be used to perform division unlike addition, subtraction and multiplication.	Algorithms should be developed that can implement division using Addition, Subtraction and Multiplication. MATERIAL: https://en.wikipedia.org/wiki/Division_algorithm
3.	SQRT: Similar to Division no circuitry is there to perform square root.	Algorithms.
4.	F2I: No algorithms is available online for Float (IEEE 754) to Integer conversion.	Algorithm has to be develop according to understanding of conversion, IEEE 754 and language.
5.	F21: For different IEEE standards Algorithm changes entirely.	Efficient algorithm designing.
6.	F21: Ranges for Sign and Unsigned conversions are entirely different, hence resource sharing can prove to be very much difficult.	Through digital logic and intelligent designing resource sharing can be done.
7.	I2F: Similar to Float to Integer no algorithms for Integer to Float are available online.	Algorithm has to be develop according to understanding of conversion, IEEE 754 and language.
8.	12F: Ranges of all IEEE 754 sub-standards are different, hence parameterization is not straightforward.	Efficient algorithm designing.
9.	COMMON ISSUE: Subnormal Numbers add an entirely different dimension to the existing format, algorithms for each and every instruction completely changes when subnormal numbers are incorporated into the instructions. Also limited literature is present online.	When developing algorithms for instructions make sure to incorporate subnormal number logic too, instead of designing algorithms for normal number first and then adding logic for subnormal numbers. To design and develop logic for subnormal number individual must have excellent understanding of IEEE 754.

10.	COMMON ISSUE: Exceptional data such as Infinities and Nans unlike integer representation are also part of the IEEE 754 floating point number standard. Detection of such exceptions and taking proper care of them is one of the major issue face in designing FPUs.	Through developed understanding of IEEE 754 and digital logic knowledge, algorithms of all the instructions must be designed so that they can detect and take care of such data.
11.	CORNER CASES: One of the biggest challenge faced in design and development of an FPU are corner cases. Corner cases are in all the instructions and some corner cases require algorithms to de completely changes if algorithms aren't developed keeping all the corner cases in mind.	Create excellent understanding of IEEE 754 and through manual solving test and check if designed algorithm works fine for both ends of the IEEE754 numbers continuum (very large numbers and very small numbers).
12.	PIEPELING: Instructions are of different complexity levels hence all instructions cannot be of same number of stages/ cycle.	FSM/ Logic must be developed using knowledge of digital designing that can control such pipeline.
13.	DATA DEPENDENCY: Instructions that require more than 1 cycle to get executed have data dependency issues in them.	Stalling!

MATERIAL

No	Topic	Material
1.	IEEE Standard for Floating Point Numbers	IEEE Standard for Floating Point Numbers
2.	IEEE 754 Arithmetic's Algorithms	IEEE 754 Floating Point basics | tutorials
3.	Pipelining	CS6810 -- Lecture 4. Computer Architecture Lectures on Pipelining (only 1st video) Conceptual Introduction to Pipelining
4.	Rounding Modes	https://pages.cs.wisc.edu/~markhill/cs354/Fall2008/notes/flpt.apprec.html

DO AND DONOTs OF FPU

1. Design whole algorithm at once instead of first designing algorithm for normal numbers first and then adding logic for subnormal numbers.
2. Rounding modes should be kept is mind too while designing algorithms, since addition of rounding modes can alter the algorithms.
3. Algorithms must be tested through manual solving first to make sure that all corner cases and exceptions are handle effectively.
4. Algorithms must be designed directly for parameterized instructions instead of first designing algorithms for single standard and then parameterizing them. Since algorithms changes a lot once parameterization is introduced.