Requirements Document
for Imagination Technologies

Pavitar Singh Devgon
Jake Humphrey
Zifan Guo
Weng Lio
Raj Mukherji
Nikolaos Dionelis

Department of Electrical and Electronic Engineering
Imperial College London

Project: FPU Design
Supervisor from Imagination Technologies: Dr Theo Drane
Supervisor from Imperial College: Dr George Constantinides

May 23, 2014

Statement of originality

We confirm that this submission is our own work. In it, we give references and citations whenever we refer to or use the published, or unpublished, work of others. We are aware that this course is bound by penalties as set out in the College examination offenses policy.

Signed: Pavitar Singh Devgon, Jake Humphrey, Zifan Guo, Weng Lio, Raj Mukherji, Nikolaos Dionelis

Abstract

This Requirements Document outlines how the design group interprets the user specification and is agreed upon by both parties before embarking on the project, as an understanding of the end product. Upon the completion of the project, the Requirements Document will be referenced in the final evaluation to determine the success of the project.

Within this document, we outline how the FPU will be designed, with regard to the mathematical operations that will be performed. We will also evaluate the FPU in term of verification and application constraints.

Contents

1 Revision Sheet

Version	Primary Author(s)	Modifications	Date
1	The Group	Initial Revision	2014-05-09
2	The Group	2^{nd} Revision	2014-05-16

2 Document Approval

Name	Title	Signature	Date

3 Introduction

Technology is evolving with tremendous speeds and the demand for high accuracy systems is apparent. Imaginations PowerVR graphics technology division designs the graphics processing units for a wide range of applications. Notably, the PowerVR graphics processors constitute a major component of Apple's iPhone devices [2], as well as of Samsung, Intel, Sony, Qualcomm, and others. As a result, efficiency in throughput, latency, area and power is critical when designing the FPU and an optimal compromise has to be found between these metrics.

For the implementation of the FPU on mobile real-time graphics, no parameter can be sacrificed. For instance, in order for graphics to be photorealistic, ray-tracing is used. Ray-tracing requires many mathematical operations that are performed on the FPU.

3.1 Purpose

The aim of this project is to analyze, design and create an efficient FPU architectural structure that will meet the requirements that are presented in Section 4.2. Raw computational power is typically measured in floating-point operations per second (Flops) and new chips will be benchmarked by analysing flop performance of the Floating Point Unit (FPU) [1]. The Group will develop efficient algorithms for specific floating-point operations. Equivalent C++ and VHDL code will be generated and will be optimized with respect to the synthesis tool performing sanity checking verification [1]. Moreover, the correctness of the FPU will be tested and verified, and an optimum FPU configuration will be proposed.

3.2 Points of Contact

During the kick off meeting, it was decided that the Group would be acting in the role of consultants to a user specification. In this role, the Group would act fairly autonomously and not be expected to contact Imagination too often. However, due to unfamiliarity with FPU design, the Group may need to consult Imagination about certain phases of the design and test process. The Group maintains that each side will respond to emails in a timely fashion (2 working days). The Group's main contact at Imagination will be Theo Drane, who can be contacted via email: Theo.Drane@imgtec.com.

The supervisor at Imperial will be the first point of contact for smaller problems in understanding. The Group will keep regular contact with the supervisor, meeting at least once a fortnight, so that all members of the Group are kept up to date with work and are not falling behind. The Group's supervisor is Dr George Constantinides, who can be contacted via email: g.constantinides@imperial.ac.uk.

4 Specification

4.1 Number Format

All input and output numbers will be represented in the `IEEE 754` floating point format. Each number will be composed of 32 bits; 1 bit for sign, 8 bits for exponent and 23 bits for mantissa. The bias will be set as 127. The significand is defined as the mantissa appended to a 1, i.e 1.m. Conversion to standard decimal value will then follow the formula: $(-1)^s \times (2^{e-127}) \times (1.m)$.

However, there are special cases where the value will not follow this rule. Table 1 presents these cases.

Table 1: Special cases for the floating point number format

Special Value	Sign Bit (s)	Exponent bits (e)	Mantissa (m)
+0	0	0000 0000	All 0s
-0	1	0000 0000	All 0s
$+\infty$	0	1111 1111	All 0s
$-\infty$	1	1111 1111	All 0s
NaN	x	1111 1111	Non-0[a]

[a] Any non-0 mantissa with all-1 exponent number is NaN

The maximum value that can be reached using this convention is given when $s = 0$, $e = 254$, $m =$ all 1s, i.e $2^{127} - 2^{104}$, or 3.402×10^{38} (`max_float`). The minimum value that can be represented is $s = 0$, $e = 1$, $m =$ all 0s: 2^{-126}, which means no numbers between 2^{-126} and zero can be expressed. In order to increase the range at the lower bound, denormals will be used such that the minimum value that can be represented will be as low as 2^{-149} (`min_float`).

When $e = 0$ and $m \neq 0$, the formula used to convert denormals to decimal value is now: $(-1)^s \times (2^{-126}) \times (0.m)$.

4.2 FPU Requirements

The required specification for the operation of the FPU can be found in Table 2.

The required accuracy is defined in terms of the ulp. 1 ulp is defined as the value that the least significant bit represents if it is 1, that is, it represents the distance between adjacent floating-point numbers.

Table 2: The required operations for the FPU

Opcode[e]	Operation	Mathematical Representation[f]	Accuracy	Throughput (cycles/op)	Latency[g]
0000	No Operation	p	RTE[a]	1	x
0001	Multiply	pq	RTE[a]	1	x
0010	Add	$p+q$	RTE[a]	1	x
0011	Subtract	$p-q$	RTE[a]	1	x
0100	Multiply-Accumulate	$pq+r$	RTE[a]	1	x
0101	Divide	$p \div q$	—[b]	4	$4x$
0110	2D Dot Product	$pq+rs$	nwc[c]	1	$2x$
0111	3D Dot Product	$pq+rs+tu$	nwc[c]	1	$2x$
1000	Square Root	$p^{1/2}$	4 ulp[d]	3	$3x$
1001	Inverse Square Root	$p^{-1/2}$	4 ulp[d]	4	$4x$
1010	2D Euclidean Distance	$\sqrt{p^2+q^2}$	nwc[c]	nwc[c]	nwc[c]
1011	3D Euclidean Distance	$\sqrt{p^2+q^2+r^2}$	nwc[c]	nwc[c]	nwc[c]
1100	Normalised Vector	$\frac{p}{\sqrt{p^2+q^2+r^2}}$	—[b]	nwc[c]	nwc[c]
1101	Unused				
1110	Unused				
1111	Unused				

[a] Round to Nearest, tied to Even

[b] No set accuracy; the Group is to specify how accurate the result will be

[c] No Worse than Chaining constituent operations

[d] Units in Last Place

[e] Opcode decided on an ad-hoc basis; subject to change

[f] p,q,...,u each represents a floating-point number as described in Section 4.1

[g] As yet unspecified, but each operation is measured relatively

4.3 FPU Interface

The FPU, when viewed as a black box, will have a 4-bit input for the operation code and 32-bit inputs for the operands. The maximum number of inputs needed is 6, for the 3D Dot Product. All operations will output one 32-bit number.

4.4 Rounding

When a number requires more precision than can be represented in 32 bits, rounding to a nearby representable number occurs. There are a number of different rounding standards detailing whether to round

up or down.

We will use rounding to the nearest, tied to even (RTE). By RTE convention, a floating-point number is rounded to the nearest representable number. For example, 38.4 will round to 38 and 38.6 will round to 39. However, in the event that the number is equidistant to two representable numbers, the original number will be rounded to the nearest even number. For example, 38.5 will round to 38.

4.5 Out of Range Exceptions

When the output of an operation is out of range, the special case values are used.

Infinity will be used to represent the result of a calculation that is too large. That is, if the calculation could be performed with an arbitrarily large number of exponent bits and after rounding, the result would be larger than `max_float`, we would round the result to +infinity.

When the result of an operation is too small to be represented by the available precision, it will be rounded to either zero or the smallest denormal number, each with the appropriate sign.

4.6 Indeterminate Operation Exceptions

All of the operations we are implementing can be broken down into five basic operations: Multiplication, Addition, Subtraction, Division and Square-Root. When chaining operations together, for example when finding the Euclidean distance, if there is an over- or underflow in the addition, this will propagate to the square root, causing inaccuracies. To prevent this, we will explore algorithms to minimise this.

The results of calculations will agree with arithmetic on the extended real number line[6], in the usual case where the result of that calculation is defined according to that arithmetic.

The existence of positive and negative zeros in our number format is the only significant deviation from the above arithmetic. In this case, we will defer to the IEEE 754 standard[1].

[1] Notably, IEEE 754 states that $(+0) + (-0) = +0$

Any undefined results will be represented with a NaN, including operations which take NaN as any input. The operations will behave in the following manner:

(x and y are any non-zero, non-infinite and non-NaN numbers)

Table 3: Behaviour of multiplication

Operand 1	Operand 2	Product
x	y	$x * y$
x	0	0
x	∞	∞
0	0	0
0	∞	NaN
∞	∞	∞

[a] The sign of the product follows the standard multiplication convention

Table 4: Behaviour of division

Divdend	Divisor	Quotient
x	y	$x \div y$
x	0	∞
x	∞	0
0	x	0
0	0	NaN
0	∞	0
∞	x	∞
∞	0	∞
∞	∞	NaN

[a] The sign of the quotient follows the standard division convention

Table 5: Behaviour of addition

Augend	Addend	Sum
x	y	$x + y$
x	± 0	x
x	$+\infty$	$+\infty$
x	$-\infty$	$-\infty$
$+0$	± 0	$+0$
-0	$+0$	$+0$
-0	-0	-0
± 0	$+\infty$	$+\infty$
± 0	$-\infty$	$-\infty$
$+\infty$	$+\infty$	$+\infty$
$+\infty$	$-\infty$	NaN
$-\infty$	$+\infty$	NaN
$-\infty$	$-\infty$	$-\infty$

Table 6: Behaviour of Square-root

Radicand	Root
x^a	$\sqrt{x}$
y^a	NaN
0	0
-0	-0
∞	∞
$-\infty$	NaN

[a] $x > 0, y < 0$

The addition table can be used to lookup the behaviour of subtraction due to the rule $x - y = x + (-y)$.

4.7 Optimisation Priority

After implementing the above functions with satisfactory accuracy, optimisation will be performed. The following list indicates the way in which optimisation will be prioritised.

1. Produce the lowest possible value for x (latency)
2. Minimise the power consumption
3. Minimise the area used

Software microcode can be used to implement higher-level machine code instructions to simplify hardware design.

5 Verification Methods

Regarding verification, three main steps will be taken. First, directed testing will be performed. Each directed test targets a specific feature of the design, for instance, specifying inputs that could lead to an overflow can be used to validate our overflow detection and correction mechanism. Passing these tests provides a clear indication of our progress.

Once our design reaches a satisfactory level we can start to increase the test coverage by using constrained-random test. This aims to cover a wide variety of inputs and root out possible corner cases while setting functional boundaries to simplify debug. This method is more efficient since it allows verification of a greater state-space. When the group is convinced that the design is of a high quality, multiple random tests will be used to ensure exhaustive functional coverage of the design, justifying that the whole design works.

These tests will be implemented using testbenches. Code coverage from running simulations on Modelsim will be one of the metrics used to measure the completeness of the tests performed.

The "Golden answer" to which our initial design will be verified upon will be obtained using the arithmetic functions provided in a VHDL floating point package by David Bishop[2]. This library have been chosen since it conforms to the IEEE floating-point standard and includes various settings such as rounding-modes, special numbers handling, number of guard-bits etc. The golden model is subject to change if other libraries or methods are found to be more suitable.

Apart from the aforementioned testing approach, formal tools for verification will be investigated. Using formal tools will allow us to explicitly verify the bit-wise equivalence of our C++ and VHDL designs. However, if these tools require more knowledge and experience than we have time to achieve, or are otherwise inaccessable, we will look into alternatives.

[2] `http://www.vhdl.org/fphdl/Float_ug.pdf`

6 Technology

The RTL hardware will be designed using VHDL and then synthesised. Since it will be difficult to get access to ASIC tools and run them with useful libraries, the Group decided to prototype the FPU design on a high-end FPGA. The FPGA will most likely be one of the following: Xilinx Virtex 7, Altera Stratix IV or Stratix V. The decision will require further discussion with our client, Imagination Technologies. Since FPGA implementations are not the end-goal of this project, the Group will ensure that the RTL does not use any FPGA-specific directives and that the RTL is ASIC-synthesisable.

7 Evaluation Methods

The FPU design will be evaluated in terms of power consumption, area and frequency, as well as of achieving the required accuracy, throughput and latency. Initial evaluation will be based on the result of static timing analysis using synthesis tools, with a focus on resource utilisation and timing analysis. Power will be measured via power analysis tools provided by Xilinx or Altera depending on the FPGA chosen as mention in Section 6. However, these tools might sometimes provide inaccurate results due to statistically estimating certain parameters. As a result, the Group will investigate the aspects of power and frequency further with bench measurements.

By comparing our solution to existing technologies, we will be able to evaluate its effectiveness in the market. One open-source floating-point compiler we have found is FloPoCo. Comparing corresponding operators with their generated VHDL design will be useful for us to benchmark our solution.

8 Timeline and Team

In this part of the report, the Group structure and the Gantt Chart for the entire duration of the project will be presented. The Group has been separated to architects, design engineers, verification engineers and application engineers. The following table shows the timeline for the entire duration of the project.

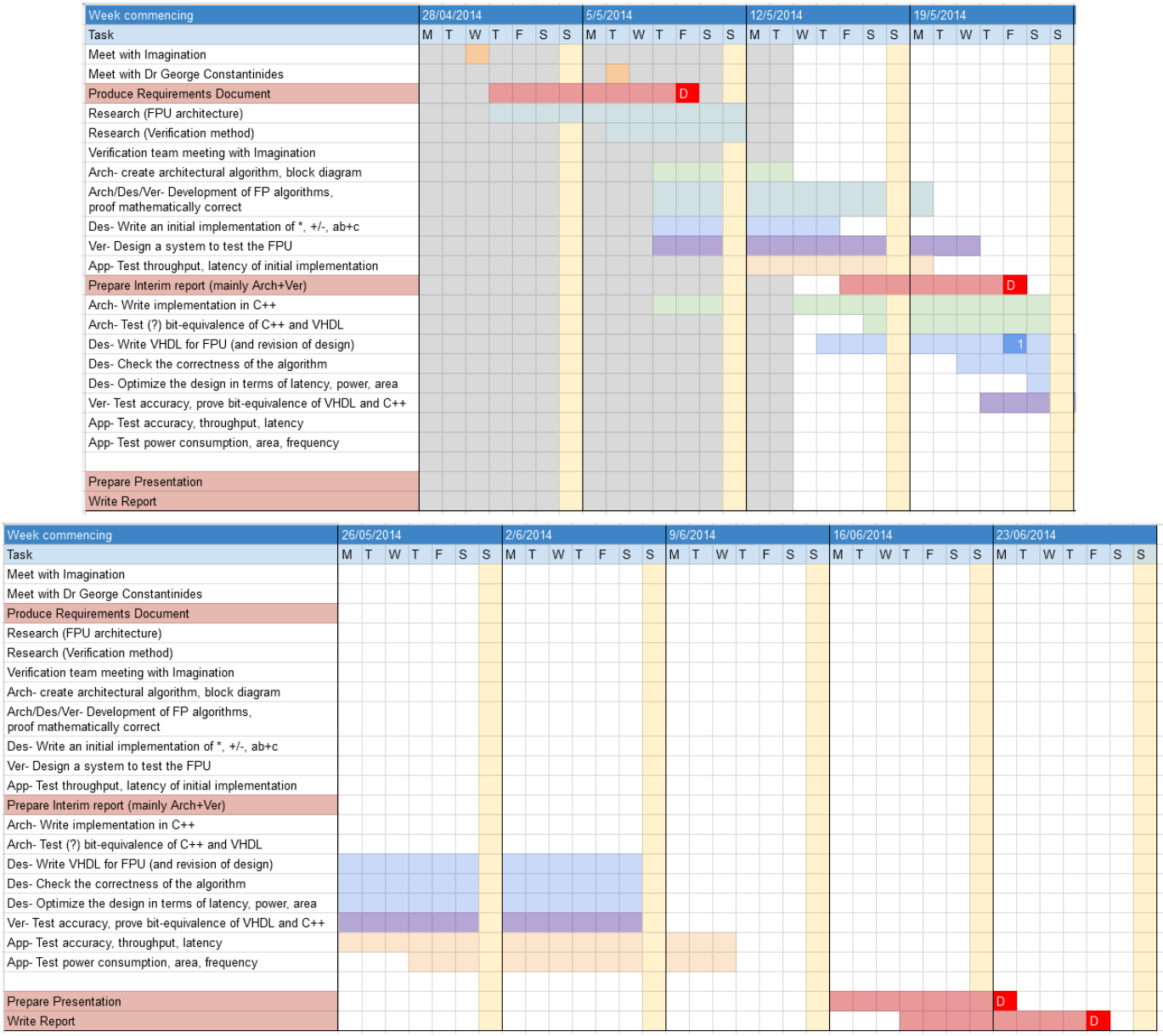

Figure 1: Gantt Chart timeline that will be used for the project.

References

[1] Imagination Technologies, *FPU Design Project Handout*, 2014.

[2] Apple, Imagination Technologies Extend IPhone, IPad Graphics Chip License Pact, *"Apple, Imagination Technologies Extend IPhone, IPad Graphics Chip License Pact."*, N.p., n.d., Web, accessed 05 May 2014.

[3] FPgen Team. *Floating-Point Test-Suite for IEEE. Rep. N.p.: IBM Labs in Haifa, n.d.* Web. `https://www.research.ibm.com/haifa/projects/verification/fpgen/papers/ieee-test-suite-v2.pdf`.

[4] Muller, J. M. *"2. Definitions and Basic Notions." Handbook of Floating-point Arithmetic. Boston: Birkhauser, 2010. N. pag.52.* Print.

[5] *THE MPFR LIBRARY: ALGORITHMS AND PROOFS. Rep. N.p.: n.p., n.d. Web.* `http://www.cs.berkeley.edu/~fateman/generic/algorithms.pdf`.

[6] *Affinely Extended Real Numbers*, Wolfram Mathworld `http://mathworld.wolfram.com/AffinelyExtendedRealNumbers.html`. accessed 15 May 2014.