GENERATION OF FLOATING-POINT MICRO-OPERATIONS USED IN GRAPHICS, IMPLEMENTED WITH FPGAs

SUMMARY

Ovidiu SICOE

Coordinator:
Prof. Dr. Ing. Mircea POPA

Faculty of Automatics and Computers
University Politehnica Timișoara

Timișoara
2018

1 Introduction

A big part of today's mobile devices have screens to interact with the human users. Having this, sustained work is in progress in order to improve the performance of those devices, regarding speed of execution, consumed power and miniaturization. As an extra concern, reusing hardware resources would significantly reduce the cost of production for those devices. A possible implementation of this reusability would come with using FPGAs(Filed Programmable Gate Arrays).

The graphics content that is presented to the end user is getting richer and richer, thus the need to continuously improve the graphical processors. In order to achieve this, not only the graphical processor, but also the regular processors began incorporating better and optimized algorithms and also arithmetic operators for composed mathematical operations.

Another solution would be to use imprecise arithmetic operators in the applications where the fidelity of the end result allows this. Extending this idea, one can also use lower precision representations for real numbers, so that the end result is accepted by a human user. Those precision losses come up with a gain in the execution time, along with a serious power economy.

Considering the above, the Thesis which summary follows introduces a series of ideas, studies and solutions that would help in the development and improvement of graphical processors used in mobile devices.

Initially, we have developed a series of architectures for matrix to matrix operators used in a graphics pipeline, implemented for FPGAs. Those architectures are generic and can be configured taking into consideration the size of the operators, the parallelization degree and the target frequency.

We have also studied the impact of using multiple real number representation formats along the stages of a graphics pipeline. In order to achieve this, we have used a proprietary implementation of the OpenGL ES 1.1 specification, which we have adapted so that it can use multiple representations of real numbers along the pipeline. For this study, we have designed and implemented a couple of software modules. Those modules helped in specifying and running a process that generated a large volume of data to be analysed. We have use a part of those software modules to analyzed the produced data and to extract multiple statistics.

In the end, after analyzing the gathered data, based on some already existing metrics for comparing similar images, we introduced a new metric. The newly introduced metric has an semantic for its values that is easy to understand and interpret. Additionally it is also easy to compute.

2 State Of The Art

This chapter presents the conclusions that were reached after analysing some relevant scientific articles for the domain of the Thesis. They can be grouped into three main cathegories:

- Algorithms and arithmetic operators implemented for FPGAs, for different mathematical operations, simple or composed
- Imprecise arithmetic operators.
- Metrics for comparing similar images.

In the first category, we have presented a series of particular arithmetic operators for simple mathematical operations and also the improvements brought by composed arithmetic operators, like the multiply-add operator. The later one has the following advantages:

- There is no need of separate rounding for the multiply and add operation, respectively.
- Only one unpacking and packing is necessary, from the IEEE 754 format to the internal format and the other way around, for each composed operation.
- Reducing the implementation costs by using the same components both for multiplication and addition.

We have also analysed a series of articles that described specialized algorithms for different particular applications. Those algorithms were specially developed for the particularities of FPGAs, so that it took benefit of their advantages. As an analysis of the performance of those algorithms, they were compared with the versions implemented for general purpose CPUs and, every time, the FPGA versions outperformed the CPU counterparts. They did this both as it regards the execution speed and the used memory bandwidth, although they operated at lower frequencies.

Continuing, we have analysed articles that highlighted the low power consumption of imprecise algorithms, implemented for FPGAs, compared to the high precision alternatives. The results of those imprecise operators were analysed taking into consideration their acceptability by using a number of metrics that quantified the differences relative to their precise counterparts.

Those imprecise operators were verified and validated using applications from the digital image processing domain. Because the end result is evaluated by a human user, we have found a mathematical evaluation of the precision loss from the final result only in a few articles.

This way, we have reached the final category of articles that we have studied, the ones related to the metrics that are used in comparing two similar images,

produced by using some low precision algorithms or arithmetical operators. This way of analysing the differences, by using metrics is well suited for an automatized analysis and can be easily scaled to a great number of results.

Based on already existing metrics that are often used in the articles we have studied, we have analysed the results obtained in our study and, based on those metrics, we have introduced a new metric.

3 Preliminary notions

In this chapter, we introduced a series of concepts used along the thesis in order to develop the proposed solutions and to accomplish the proposed study. This way, we have presented the following concepts:

- A graphics pipeline and the OpenGL ES 1.1 specification
- The representation of 3D objects using polygons spread along their surface
- Coordinate transformations used along the graphics pipeline
- Representations of real numbers using fixed point and floating point

For the graphics pipeline and the OpenGL ES 1.1 specification we have introduced basic concepts, like the main stages and the link between them. We have also shortly presented how the final 2D projections are obtained, starting from 3D objects. Following this, we have detailed the aspects of representing 3D objects as entry data that is accepted by the graphics pipeline. This specification describes the 3D entries using triangles in a 3D space. The triangles are represented by the coordinates of its vertices. This way, the triangles can be provided independently, in a triangle strip or a triangle fan, according to the specification.

For the coordinate transformations we have presented the mathematical concepts behind those transformations. We have also presented some examples based on the translation and scaling matrices in a 2D space.

In the end of the chapter, we have introduced the basic concepts and the mathematical formulas behind two real number representations: fixed point and floating point, respectively.

4 Matrix to matrix operators used in a graphics pipeline

Using the mathematical models presented in the previous chapter, we have taken advantage of the characteristics of matrices corresponding to the translation and scaling transformations. Having this, we have developed an architec-

ture for matrix to matrix operators, FPGA synthesizable, that could be configured with the following parameters:

- The size of real numbers. We have used the floating point representation described in the IEEE 754 specification, being able to configure both the size of the mantissa and of the exponent:
  - exponent 5 bits; mantissa: 10 bits(half precision float)
  - exponent 8 bits; mantissa: 23 bits(float)
  - exponent 11 bits; mantissa: 52 bits(double precision float)
- The target frequency for synthesis. This parameter determines the number of stages for the operator's pipeline:
  - 100 Mhz
  - 200 Mhz
  - 300 Mhz
  - 400 Mhz
- The parallelization degree. This parameter directly influences the resources used in implementing the operator:
  - Maximum
  - Medium
  - Minimum

These configuration parameters allow obtaining of a series of operators that are customized for a large number of applications, according to the specific needs of each one. For each architecture for translation and scaling, respectively, according to each combination of parameters, we have obtained a total of 36 operators.

This way, according to the parallelization degree, for each geometric transformation, we have created operators containing:

- Six processing units. Every unit computes the result corresponding to a position in the result matrix. The result matrix has nine entries, but three of them are identical to the input matrix, because of the particular form of the transformations matrices.
- Two processing units. Every unit computes the results for one of the two different lines in the result matrix. Having this, in the k clock cycle, the first position of each line is prepared, in the $k+1$ clock cycle the second position will be ready and in the $k+2$ clock cycle the last position would be ready.

- One single processing unit. In this case, the processing unit produces the results for all the six positions of the result matrix. Similar to the previous model, it would take six clock cycles to provide the final results.

In order to implement those architectures, we have used the FloPoCo library, which offers an infrastructure for generating the VHDL code used for the FPGA synthesis of those operators. The needed model for this library was developed using C++ and it included a part for verification and validation of the generated operators. This testing part consisted in the generation of the testbench corresponding to each particular operator.

In the end of the section for each translation and scaling operator, respectively, we have presented a table containing the synthesis results for the operators. Those results sustain the main idea that these operators cover a large number of application classes, corresponding to the needs and constraints of each, individually.

5 Implementation of approximate computations

This chapter of the thesis presents the software modules developed for creating a process used to accomplish the study of the impact of using multiple real number representation along the same graphics pipeline.

In the first stage of the graphics pipeline described by the OpenGL ES 1.1 specification, the vertices of the input triangles for the 3D object are processed. This processing consists in applying some affine geometric transformations. This is achieved by successive matrix to matrix multiplications. Having this, we have chosen to modify a proprietary implementation of the OpenGL ES 1.1 specification in order to be able to use one representation for real numbers in the first stage and another, different representation, in the rest of the pipeline.

In order to have a satisfying volume of input data, we have created a software module that scans a series of web pages dedicated to 3D objects created by a variety of graphical artists. Using this, we have collected about 3000 files containing descriptions for 3D objects. Taking into consideration that the OpenGL ES 1.1 specification supports only triangles as polygons for describing a 3D surface and some of the collected files used polygons with four or more vertices, we had to create an extra software module that transformed those files so that they contained only acceptable data. Some of the files that were unable to transform were eliminated. After this first stage, we have gathered around 2400 3D valid objects that were recognized by the available graphics pipeline implementation.

The next step was to provide the data to the application. The application was supposed to project those 3D objects on a plane surface, using our own implementation of the graphics pipeline, according to the OpenGL ES 1.1 spec-

ification. This way, the application consists of the following steps:

1. we clear the drawing 2D surface
2. we draw the object that is rotated with the current rotation angle, starting from zero
3. we obtain a PNG image corresponding to the projection of the rotated 3D object
4. we rotate the 3D object with to degrees around the (-1, -1, -1) axis, resulting in a new rotation angle that will be used in the next frame
5. we resume the drawing process with from the first step in order to obtain the next frame

In order to run the application for all the 3D input objects we needed 21 one days of processing time. We have split this time among three work stations that worked in parallel. Those stations had the following minimal configuration:

- Intel Core i7 processor clocked at ≈2500 MHz, with four cores and eight threads
- 16 GB RAM
- SSD

After running the application, we have gathered more than three million frames. About 450000 images are reference images. Having this, in order to analyse the differences between altered images, that were obtained when using less precise representation formats for real numbers and the reference images, we have created a software module for comparing the two similar images. This module produces a difference image and a file with the values of some metrics that were taken into consideration for the study.

Finally, we needed a software module to analyse the data that resulted from the previous step. For this, we have created a dedicate software module that consisted of two main subsystems: one for reading the data and one for processing the data. Due to the great amount of time that was needed to process the data, the read module was statically loaded by the processing application and the processing module was dynamically loaded, each time the data was requested to being processed. This way, we could read the data only once during one run of the processing application and the processing module could be continuously adapted, rebuilt and reloaded, without needing to read the data again.

For this study, the data was obtained by using the following combinations of formats for representing real numbers:

In the previous table, F_0 represents the format used in the first stage of the graphics pipeline and F_1 represents the format used in the rest of the stages.

[1]Floating Point]

[2]fix point format specified as size of integer part : size of fractional part

	F_0	F_1
1	FP[1]	FP
2	16:16[2]	16:16
3	16:16	FP
4	FP	16:16
5	32:32	32:32
6	32:32	FP
7	FP	32:32

Table 1: Used formats

THe first line in the table represents the combination of maximum precision that was used in obtaining the reference images. The rest of the lines describe combinations of precision in which at least one of the formats is less precise.

6 Results of approximate computations

This chapter of the thesis presents the results obtained following the analysis process described in the previous chapter.

In the beginning of the chapter we have presented the 3D object library that we created and used to accomplish the study. We presented the characteristics of the objects and a series of charts and histograms that describe their distribution. We thus highlighted the variety of objects that we have used in the study, ranging from a small volume, around the unit of volume to objects fitting into a volume of the order of millions of volume units. We have also had objects described by two triangles to objects described by about 450000 triangles.

Following this, we have presented the results of a short subjective analysis, accomplished by a human observer. This was done by visually comparing the altered image to its precise counterpart. We could observe this way that such an analysis is not good enough to highlight the, sometimes insignificant, differences between the obtained images. Thus, we have continued by introducing some metrics described in the speciality literature into the analysis: MSE(Mean Square Error) and PSNR(Peak Signal-to-Noise Ratio).

$$MSE(f,g) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(f_{ij} - g_{ij})^2 \tag{1}$$

$$PSNR(f,g) = 10\log_{10}(\frac{255^2}{MSE(f,g)}) \tag{2}$$

In the previous equations, f and g represent the pixel set of the two compared images while M and N represent the width and height of the images. The values

of MSE represent the average of the squared distance of the pixels of the two compared images. In the same time, PSNR is defined on a logarithmic scale, based on MSE. PSNR was introduced like this to limit the values obtained by using MSE.

We have tried following the values of those metrics and extracting some patterns. The option that seemed most suited for this was through charts for the values of the metric corresponding to each object, in each frame, for each combination of representation formats. We have observed this way a dependency between some characteristics of the 3D objects, like their volume, between the shape of the values' distribution and the combination of representation formats that was used. We have also observed that the values of these metrics vary a lot, from one object to another, for different combinations of representation formats, making a difficult task out of the comparison between the obtain results.

Following this conclusion, we have introduced a new metric, which we have defined relative to the useful number of pixels in the reference image:

$$M(f,g) = \frac{1}{P(f)} \sum_{i=1}^{M} \sum_{j=1}^{N} D(f_{ij}, g_{ij}) \tag{3}$$

where

$$D(f_{ij}, g_{ij}) = \begin{cases} 1, f_{ij} \neq g_{ij} \\ 0, f_{ij} = g_{ij} \end{cases} \tag{4}$$

and

$$P(f) = number\ of\ useful\ pixels\ of\ the\ pixels\ matrix\ f \tag{5}$$

To have a better semantic for the values of this metric, we felt the need to define it as a percent. This way, we have introduced the following definition, that we have used in the subsequent work.

$$M\%(f,g) = M(f,g) * 100 \tag{6}$$

The metric define by the previous equation highlights the percent of pixels that are different in the altered image, relative to its reference counterpart. This way, the distribution of the values for this metric have a similar form with the ones for the MSE and PSNR metrics, just that the values occupy a restrained interval.

Near the end of presenting the results of the study, we have presented a series of charts where we have used aggregations of the previously described metrics, for all the 3D objects that were taken into consideration. We could thus observe some minimum and maximum points corresponding to some frames. Those frames correspond to the projection of the objects rotated with 60, 120, 180 and 240 degrees, respectively. One of the main reasons for those points

could be the lack of precision in representing the values for the trigonometric functions sine and cosine that are used in some of the positions of the rotation matrix used in the graphics pipeline.

7 Contributions, conclusions and perspectives

In the final chapter of the thesis we have presented the original contributions, the conclusions we have reached, along some perspectives in developing further more the presented studies.

Concluding, in the first part of the thesis we have presented a series of matrix to matrix operators, implemented for FPGAs, that could be configured before synthesis according to the following parameters:

- The size of real numbers.
- The target frequency for synthesis.
- The parallelization degree.

Following this, in order to extend the set of developed operators, we studied the influence of using multiple representation formats for real numbers in the stages of a graphics pipeline. This way, we have modified an own implementation of the OpenGL ES 1.1 specification which allows changing the representation formats for real numbers at runtime. We have also created a series of software modules that helped us achieving the following:

- Collecting of 3D objects used as input data for the graphics pipeline.
- Transforming the input data to a format that is accepted by the graphics pipeline.
- A software application that wraps the graphics pipeline implementation and provides the input data to it in order to produce the projections of the 3D objects.
- Comparing the altered images, obtained by using a combination of less precise representation formats, to the reference images obtained by using a combination of higher precision representation formats.
- Analysis of the obtained images and computation of the values for some metrics taken into consideration for the study.
- Filtering, presenting and interpreting the values of the metrics.

Using those modules, we have defined a process to help the proposed study. After obtaining the data resulting from the process, we presented the conclusions by using significant examples, conclusive metrics and charts with the values of the metrics.

Summarizing, during the research for the creation of this thesis, we have obtained the following concrete results:

- We have developed an architecture for the geometric operators for translation and scaling that can operate with different formats of real numbers represented in floating point, including nonstandard formats. Additionally, those architectures can be configured to produce different operators, based on the chosen format, the desired pipeline depth and the target frequency.
- We have created a 3D object library that can be used as input data for a graphics pipeline.
- We have created a process and the corresponding software modules that helped in studying the influence on the final results, of using multiple representations for real numbers in the same graphics pipeline.
- We proposed a new metric with a clear and simple semantic of its values that would in evaluating the difference between an altered image and the corresponding reference image.

In the end, we have presented a series of actions that could be undertaken in order to further develop the research presented in the thesis:

- Creation of configurable matrix to matrix operators for the rotation transformation as well as for other affine geometric transformations used in a graphics pipeline.
- Creation of processing units implemented in FPGA that would implement in hardware the software concept of modifying the graphics pipeline presented in this thesis.
- After the hardware implementation of the concept, a comparative analysis with possible existing solutions, regarding performance.
- Using the combinations of formats at a narrower scale; this way, it would be possible to use many more representations for real numbers in the same graphics pipeline.
- Defining a threshold for the values of M% so that values greater than this threshold denote an unacceptable image for a human user.