Comparison of Two Floating Point Arithmetic Units for a Precomputer in a Graphics System for Real Time Simulation

Reinhard Möller

Bergische Universität
Gesamthochschule Wuppertal
Fachbereich Elektrotechnik

This paper compares two realized concepts of floating point units in a visual system for a traffic simulator. The hardware structure of a Transformation Processor is described, with which a set of 1x4 vectors can be floating point multiplied by a 4x4 matrix autonomously in real time. It will be shown, that the speed of graphics computations can be advanced enormously by using specially designed parallel graphics hardware but also requires the elimination of some design constraints given by the available building blocks in VLSI design today.

1. Introduction

In the past four years at the Institute for Automation Techniques of the Fachbereich Elektrotechnik of the BUGH Wuppertal a visual system for a traffic simulator has been developed, which is marked by the successful usage of standardized (MOS) electronic components [3, 4]. It is a low cost system, which garants the real-time response for a simulator. In this visual system a lot of microprocessors are working parallel, transforming the picture data base to be visualized. Most of the transformation calculations have to be done as 32-Bit floating point operations, so special concepts for rapid processing of the graphic data in hard- and software were developed, which allow the normally time consuming floating point operations.

The Precomputer used for the transformation calculations is described, two specially for this task developed floating point arithmetic units are shown and their performance will be compared.

2. Description of the Precomputer

2.1. Tasks of the Precomputer

The visual system for the traffic simulator shows the transformed picture data in real time. This is reached with a modular concept of parallel working transformers, each of them processing one object to be visualized.

A transformer consists of a Precomputer and one or more elementary generators. One object can be processed by more than one transformer, which are organized in a so called "Module".

The Precomputer does the following tasks:

- reception and storage of the object describing data,
- reception and storage of the static data (Data, which are constant over a longer time period),
- reception of the dynamic data (Data, which vary with every frame cycle),
- calculation of the visualisation transformations, basing on the received data,
- calculation of the visual parts of the object to be visualized,
- sorting of the outputlists to the visualisation generators,
- transferring of the processed data to the generators.

With the example of a Precomputer for the visualization of a car this shall be demonstrated:

> The describing data of the car, that are the points p_i of a wire frame carmodel, will be stored in the Precomputer at the beginning of the simulation. Now the Precomputer is activated for the simulation, that means, it is ready for receiving the dynamic data and the transformer is runnable. The first received data while simulation are the position data (matrix T_{OW}) of the car model, with which the car is localized in the simulation world. The dynamic data consist of the observers orientation coordinates and angles, forming a 4x4 matrix T_{WB}. With the multiplication of the orientation matrix by the position matrix the general transformation matrix T_{OB} is created, by which now all object points p_i must be multiplied. The results will be perspective projected into screen coordinates.

The sorting and output operations following should not be discussed in more detail.

2.2. Hardware Concept of the Precomputer

The Precomputer consists of five parts, as shown in figure 1. The CPU receives (controlled by hardware interrupt) data from a 16-bit parallel port (CIO). Then it stores and processes these data in its main memory and afterwards it transfers the results to the 16-bit parallel outputport (FIO).

An other link for data communications exists with the serial communications controller (SCC). With it the operating system functions of the Precomputer can be controlled and the data link between the Precomputer and a host computer is possible.

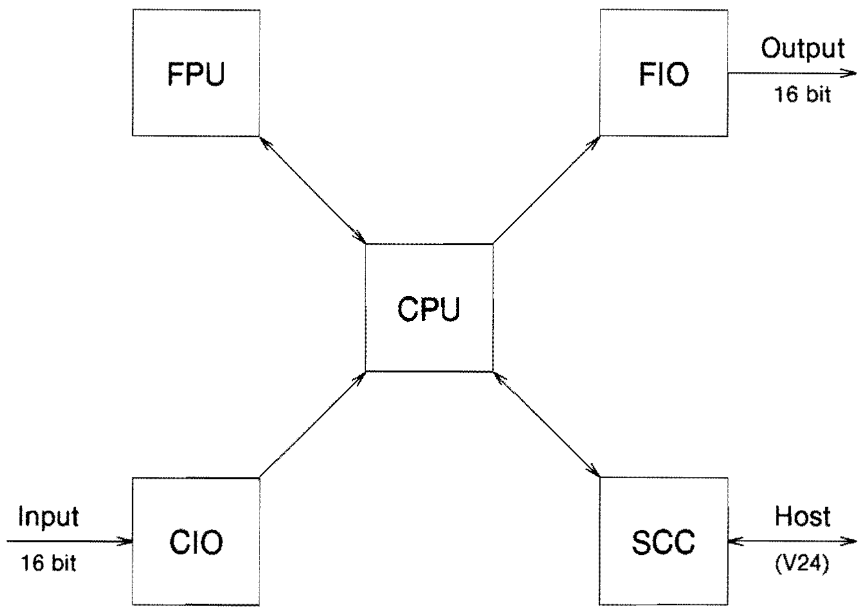


Figure 1: Components of the Precomputer.

One of the most important components of the Precomputer is the floating point unit (FPU). The Precomputer in the described visual system mostly has to compute the transformation equations, and for that it must do exactly calculations on floating point numbers. The computing power of the Precomputer can be measured by the amount of points, which can be transformed in a given time, that means, how many new 4x1 vectors can be obtained through multiplication of a given set of 4x1 vectors by a 4x4 matrix. This led to the Multi FPU Solution, which is described in the following paragraph.

2.2.1. Multi FPU Concept

The floating point unit, which is realized until now, consists of four equal processors. The multiplication of a 4x1 vector p_1 by a 4x4 matrix M can be done with 12 steps:

Starting point: $p_2^T = p_1^T \star M$

FPU1	FPU2	FPU3	FPU4
$f_1 := m_{11}$	$f_1 := m_{12}$	$f_1 := m_{13}$	$f_1 := m_{14}$
$f_2 := m_{21}$	$f_2 := m_{22}$	$f_2 := m_{23}$	$f_2 := m_{24}$
$f_3 := m_{31}$	$f_3 := m_{32}$	$f_3 := m_{33}$	$f_3 := m_{34}$
$f_4 := m_{41}$	$f_4 := m_{42}$	$f_4 := m_{43}$	$f_4 := m_{44}$
$f_1 := f_1 \star x_1$	$f_1 := f_1 \star x_1$	$f_1 := f_1 \star x_1$	$f_1 := f_1 \star x_1$
$f_2 := f_2 \star y_1$	$f_2 := f_2 \star y_1$	$f_2 := f_2 \star y_1$	$f_2 := f_2 \star y_1$
$f_3 := f_3 \star z_1$	$f_3 := f_3 \star z_1$	$f_3 := f_3 \star z_1$	$f_3 := f_3 \star z_1$
$f_4 := f_4 \star h_1$	$f_4 := f_4 \star h_1$	$f_4 := f_4 \star h_1$	$f_4 := f_4 \star h_1$
$f_1 := f_1 + f_2$	$f_1 := f_1 + f_2$	$f_1 := f_1 + f_2$	$f_1 := f_1 + f_2$
$f_1 := f_1 + f_3$	$f_1 := f_1 + f_3$	$f_1 := f_1 + f_3$	$f_1 := f_1 + f_3$
$f_1 := f_1 + f_4$	$f_1 := f_1 + f_4$	$f_1 := f_1 + f_4$	$f_1 := f_1 + f_4$
$x_2 := f_1$	$y_2 := f_1$	$z_2 := f_1$	$h_2 := f_1$

The f_i are contents of the FPU registers, x_i, y_i, z_i, h_i are the components of the vectors p_i and m_{ij} the components of the transformation matrix M.

The advantage in time while processing the transformation calculations is significantly (normally 16 multiplications and 12 additions = 28 operations are necessary) and will be increased by freeing the CPU for other tasks during the time the FPU is calculating: It is possible to produce *real* parallel processing with efficient programming, that means, the CPU does operations, that do not require the FPU, while the floating point units process an arithmetic statement of the CPU [7].

2.2.2. Bit Slice Concept

The idea to save time by processing equal operations with parallel hardware was realized with the Multi FPU Concept by the help of standard MOS processors. But alternatively a solution with bit slice processors was developed. The question was, how such a concept, compared with the Multi FPU solution could be economically or technically advantageous.

3. Concept of a 4x4 Transformation Processor

3.1. Tasks for a Transformation Processor

The here proposed Bit Slice Processor shall replace the floating point units described before. It was designed for the most complex task, that means, for the 4x4 transformation. Until now all operations of the Precomputer have been realized by MOS components because the traditionally bipolar bitslice technology was well known to be uneconomic and expensive. For the development of the Transformation Processor a new direction was chosen because new electronic components were available on the market, which on one side offer a wide range of complexly functional blocks and on the other side are produced in the advantageously CMOS technology. Former bitslice components mostly used 4 bit slices, which caused a high expenditure in wiring a complex system. The used components consist of 16 bit slices and so they minimize the necessary logic for the desired 32 bit Transformation Processor.

The Transformation Processor (TP) should be used for the following tasks, controlled by the CPU:

- 4x4 transformation of one or a set of vectors,
- arithmetic operations, i.e. addition, subtraction, multiplication, division,
- logic operations,
- transport operations, i.e. internal register loading and data communication between TP and CPU.

3.2. Components of the Transformation Processor

The Transformation Processor (TP) consists of the following components (figure 2): Precomputer Interface; Operation Control memory; Microcode Memory; Sequencer; Data Address generator; data/Coefficient memory; a floating point multiplier; a floating point adder; Timing & control logic.

The *Precomputer Interface* is used for the communication between CPU and Transformation Processor. It consists of two ports, a FIFO port for incoming results to the Precomputer and a latched output port to the TP. The TP can be told by special control words, which kind of operations should be done. Operations can be data transfer to microcode memory, data/Coefficient memory or operation control memory.

The *Sequencer* controls the data flow of the TP and decodes incoming operation codes by using the operation control memory to switch into the opcode related program portion of the microcode memory. The conditions of arithmetic operations are also used for address calculations.

The data to be processed by the Transformation Processor are stored in the data/Coefficient memory, which is controlled by the *Data Address Generator.* The memory is splitted for time optimal data transfer to the arithmetic units. It is

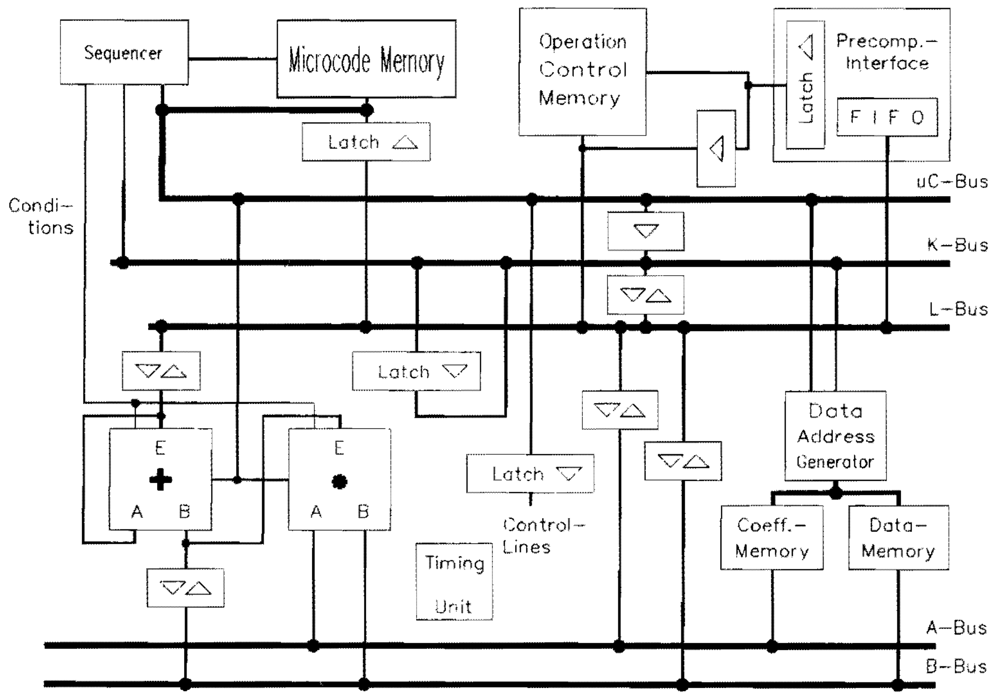
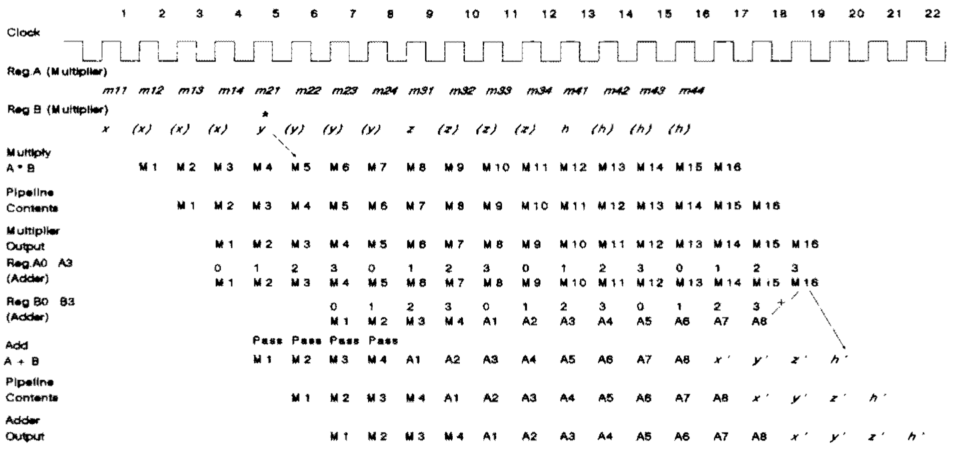


Figure 2: Components of the Transformation Processor

possible to send both a matrix coefficient and a component of a point vector parallel to the floating point units. The Data Address Generator obtains control informations from the L-Bus and the microcode bus (μC-Bus). These data must be hold for a time period, which is done on the K-Bus.

The *Floating point Multiplier* and the *Floating point Adder* are connected via two buses (A-Bus, B-Bus), which are connected to a main L-Bus via bidirectional buffers. Data transfer can be done so in every direction from each port of the multiplier to each port of the adder, and the 4x4 transformation can be optimal calculated with minimum expense. An example of an optimal transformation is shown with the nested operation and transfercycles in figure 3.

With the *Control Logic* the Transformation Processor is able to be used as a "device under development" with all degrees of freedom. It is possible to drive the TP with the normal clock frequency of 10 MHz or in either single step or single clock mode, so that every operation of the TP is controllable.



$$\begin{bmatrix} x' \\ y' \\ z' \\ h' \end{bmatrix}^T = \begin{bmatrix} x \\ y \\ z \\ h \end{bmatrix}^T * \begin{bmatrix} m11 & m12 & m13 & m14 \\ m21 & m22 & m23 & m24 \\ m31 & m32 & m33 & m34 \\ m41 & m42 & m43 & m44 \end{bmatrix}$$

Figure 3: Operation and Transfercycles in the TP

4. Performance Comparison: Bitslice / Multi FPU Concept

The following explanations are basing on:

- an operation frequency of 10 MHz for all components (that means T = 100ns),
- a mean Z8000 memorytransfer cycle of C := 3T,
- a mean Z8000 peripherytransfer cycle of D := 10T,
- the operation time t, expressed in number of cycles T,
- the Multi FPU Arithmetic Unit consists of 4 FPUs.

The following tests have been executed:

4.1. Elementary Calculations

Here the Multi FPU Concept is not advantageous, because only two operands are to be combined. The operation to do is

$$c := a \; op \; b \quad \text{with } op \in \{+,-,*,/\}$$

that means, in every case three steps for the calculation are necessary.

With one FPU the operation time is t = 50T + 11C + t_R. The calculation time t_R of the FPU lies between 20T (measured mean value) and 70T (value extracted from the data sheet), so that a total calculation time of $\underline{t = 103...153T}$ is given.

The bitslice concept requires t_T = 92T + 24T for the transfer of operand and opcode to, and for the transfer of the solution from the bitslice system. An additional calculation time of 2T for the selected operation must be considered: t = 118T.

For both concepts an additional calculation time of 18T is to be considered, if a floating point division is desired.

4.2. Processing of the 4x4 Transformation

With a Multi FPU solution and the "12-Step-Scheme", as in 2.2.1 described, the following time values are given:

a) Transfer of the transformation matrix. The registers of the four FPUs are loaded (t_1 = 63T + 16C).

b) Four multiplication steps are processed, while the components of the point vector, which is to be transformed, are transferred. For the multiplication a mean time value of 20T was measured (value from data sheet: 44T), so that the whole execution time is t_2 = 48T + 80...176T + 40C.

c) Three steps of addition are executed: t_3 = 33T + 90...210T + 6C.

d) The transformed point vector is transferred to the main memory of the Precomputer (t_4 = 26T + 14C).

The total estimated time for the 4x4 transformation is $t = t_1 + t_2 + t_3 + t_4$, that means t = 568...784T.

It should be noted, that the CPU can do operations in parallel while the FPUs are calculating. If this is not observed, i.e. with programming some FPU operations immediately following another, the CPU performs wait states during the calculation times of the FPUs.

For the 4x4 transformation of a 4x1 point vector with the help of a Transformation Processor the following operations are necessary:

a) Transfer of the transformation matrix. For this task 16 floating point numbers and the transfer operation statement (each 32 bit) are to be transferred from the CPU main memory to the Coefficient Memory of the TP (t_1 = 32T + 34D).

b) Multiplication. It can be seen in figure 3, that 22T clock cycles are necessary for processing the matrix multiplication. This is the result of nested multiply and add operations and the usage of the pipeline structure of the floating point processors in the TP.

The transfer of the MULTIPLY statement (2D), the point to be transformed (8D) and the transformed point (8D) result in: t_2 = 57T + 18D + 22T.

The total time for a 4x4 transformation is given with $t = t_1 + t_2$ as t = 631T.

4.3. Calculation of a Cubic Spline Transformation

With the calculation of cubic spline functions the Multi FPU Concept approximates the speed of the Bit Slice Concept. In [3] a procedure using the Multi FPU Concept was proposed, which does all operations on one vector at one time.

The recursive part of the procedure:

$$\begin{array}{llll} p(k) & := & p(k-1) & + \; D1_{k-1} \\ D1_k & := & D1_{k-1} & + \; D2_{k-1} \\ D2_k & := & D2_{k-1} & + \; D3 \end{array}$$

can be done with the Multi FPU Concept in three steps. The time for calculating these three steps in the Multi FPU arithmetic unit is approximately t = 33T + 6C + t_R clock cycles. Either using the mean measured calculation time (30T per addition) or the calculation time in the data sheet of the FPU (70T) the total calculation time of the FPUs is given as $\underline{t = 141...261\text{T}}$.

A recursion in the Bit Slice arithmetic unit (BSAU) is processed after $\underline{t = 20\text{T}}$ clock cycles (equal to t_R in this case) because nested statements allow multiply and add operations, together with transfer operations every one clock cycle. The special construction of this arithmetic unit allows another advance in speed: It is possible to store all points which are calculated (normally 200 point vectors) in an intermediate memory of the BSAU until the recursion ends. A block transfer of all points to the CPU takes place after all calculations. With that the BSAU concept works five times faster than the Multi FPU Concept because the last one needs transfer operations to the CPU between every two recursions.

5. Conclusion

Two concepts of high speed floating point arithmetic units have been proposed. Their advantages and disadvantages could have been noticed in the comparison of the calculation speeds for some typical tasks of the computer graphics. As it was shown, the parallelising of arithmetic units gives a large advance in speed. The economical expense for the usage of the Multi FPU Concept is much more less than for the realized Bit Slice Concept: The costs of an arithmetic unit consisting of four FPU Chips amount about DM 1.600,--, basing on their price today, compared with the price of DM 5.000,-- for the realized Bit Slice Arithmetic Unit (including a large amount of high speed memory components).

It could be seen in section 4, that the advance in speed with the proposed concept of the Transformation Processor ranges from 5 times for the Spline approximation (worst case) to a minimum of approximately one time in all other cases. This is a result of the large amount of time, used for transfer of variables and operation codes in the BSAU concept. In the Multi FPU Concept the "extended processing instructions (EPI)" capability of the CPU was used. With it the access to the main memory of the Z8000 CPU is about three times faster than for an I/O transfer but restricted to a maximum amount of 16 words per transfer instruction.

An application of the EPI facility to the BSAU Concept would accelerate the speed of the Transformation Processor to the rate expected after the mentioned cubic-spline-test, but alternatively it is planned for the near future to extend the Transformation Processor to a complete Graphics Precomputer.

Some concluding remarks on the equalities between the two concepts and their advantage for Computer Graphics should be mentioned at this point: In both concepts a SIMD/MIMD architecture of parallel working hardware components was chosen, multiprocessor systems with implemented pipeline/dataflow mechanisms where possible. This architecture seems to be sufficient for the speed requirements of Computer Graphics hardware in realtime applications, but it could be extended to the use of systolic arrays of processing elements (PEs) for some applications.

The data manipulation overhead in a multiprocessor system like the Transformation Processor could be minimized by concentrating of all PEs for a graphics task on one special VLSI chip. Graphics tasks could be 4x4 transformation, clipping, curve/surface generation, texture processing, filtering in time and space and other calculation intensive tasks.

Considering the design of graphics VLSI circuits some real limitations should be mentioned:

- The maximum parallelism in graphics hardware is restricted to the existence of parallelism in graphics algorithms,
- The design of integrated circuits for the mentioned tasks is mostly restricted by the number of available gates on a chip, the number of I/O pins, internal busing capabilities and the availability of special function blocks (i.e. microcontrollers, floating point ALU/multipliers/dividers and memory structures).

6. Acknowledgements

The author thanks Analog Devices GmbH for their kindly support of the development of the 4x4 Transformation Processor. Also he thanks cand. Ing. K. Pelzer and cand. Ing. C. Knappstein for their work on the device in hard- and software.

7. References

[1] Clark, J.H. "The Geometry Engine: A VLSI Geometry System for Graphics".",fIACM Computer Graphics. **16** 3, Juli (1982)

[2] Foley, J. and Van Dam, A. "Fundamentals of Interactive Computer Graphics", *Addison-Wesley Systems Programming Series.* Addison-Wesley Publishing Company (1982)

[3] Möller, R. "Entwicklung von Hard- und Softwarekomponenten für das Sichtsystem eines Verkehrssimulators". Dissertation im Fachbereich Elektrotechnik der BUGH Wuppertal, Dezember (1986)

[4] Möller, R. "A Visual System for a Traffic Simulator". W. Strasser (ed.): *Advances in Graphics Hardware I.* Springer (1987)

[5] Newman, W., M. and Sproull, R. F. "Principles of interactive Computer Graphics." ed. H.S.Stone, *McGraw-Hill Computer Science Series.* Tokyo: McGraw- Hill, 2nd ed. (1972)

[6] N.N. Word-Slice^{TM} and Floating Point Components Seminar. Analog Devices Digital Signal Processing Division, Norwood, MA., Fall (1985)

[7] Skiba, T. "Implementation von Floating-Point-Prozessoren in das Mikrorechnerkonzept eines Sichtsimulators." Studienarbeit am Lehrstuhl I für Automatisierungstechnik im Fachbereich Elektrotechnik der BUGH Wuppertal, Februar (1986)