Design and Implementation of a High Speed and Low Cost Hybrid FPS/LNS Processor Using FPGA

Assist. Prof. Dr. Dhafer R. Zaghar
Department of Computer & Software Engineering/College of Engineering/University of Al-Mustansiriyah /Iraq
Email-drz_raw@yahoo.com

Abstract

In the world of the computer data processing there are two main groups of processors first the microprocessor group that use the floating point system (FPS) and the TMS processor group that use logarithmic number system (LNS). There are many works and ideas to improve the two types and mixed between them but the main drawback of these works is that "there are no common rules to measure the efficiency of each work and compare between them". This paper presents some logical and fair rules to measure the efficiency of the processor as a first step on the true way to implement a good process. Hence, this way has three main phases. First, classify the mathematics operations and deduce the approximation weight of each operation in the computer data processing such as general digital signal processing (DSP) fields, fast Fourier transform (FFT), filtering and neural network (NN). The second phase is proposing the design of an optimal process that has a high speed and low cost. The third phase is modifying the optimal design to implement it in the field programmable gate array (FPGA) media. Then, this paper will use the new rules to measure the efficiency of the proposed design and compared it with previous works. Also it will give the most important conclusions that will to steer the designer to implement a high speed and low cost processor.

Key words : ***Floating point, LNS, DSP, FFT, NN, FPGA, processor, mathematical operations, piecewise.***

الخلاصـــــة

يوجد في ميدان معالجة البيانات بالحاسبة مجموعتان من المعالجات الاولى تعتمد على نظام الفارزة الحرة العددي (FPS) و الاخرى تعتمد على النظام العددي اللوغارتمي (LNS). ان هناك عدد كبير من الاعمال التي تتناول هذا الميدان و تستعرض كم كبير من الطرق و الافكار التي تساعد على تحسين اداء هذه المعالجات او المزاوجة بينها, الا ان المشكلة المشتركة بين هذه الاعمال تكمن في عدم و جود قواعد متفق عليها لقياس الكفاءه لكل عمل و المقارنة بينها. سيحاول هذا البحث و ضع قواعد عادلة لقياس الكفاءة كخطوة اولى في الطريق الصحيح لبناء معالج جيد, و هذه العملية تتضمن ثلاث خطوات اساسيه الاولى تقوم بتصنيف العمليات الرياضية و تحاول تخمين المقدار التقريبي لنسبة الاستخدام لهذه العمليات و ذلك في المجالات الحاسوبية المختلفة ضمن نطاق معالجة البيانات مثل معالجة الاشارة الرقمية (DSP) و تحويل فورير السريع (FFT) و الترشيح (filtering) و الشبكات العصبية (NN). اما الخطوة الثانية فتتضمن وصف تصميم معالج ذو سرعة عالية و كلفة قليلة. اما الخطوة الثالثة فتقوم بتحوير التصميم لغرض بناءه بواسطة مصفوفة البوابات المبرمجة الواسعة (FPGA). بعد ذلك سنستخدم القواعد الجديدة لاختبار كفاءة المعالج المقترح و مقارنتة مع الاعمال السابقة و اخيرا سيعطي البحث خلاصة تساعد المصمم لبناء معالج ذو سرعة عالية و كلفة قليلة.

1- Introduction

Most papers and thesis that deal with the design and implementation of FPS, LNS or hybrid processors discuss the result from one side. As an example M. Haselman et al. [1] discuss the speed and cost for operations without log and exp converter cost and time. While some papers that use FPS discuss the result with additional log and exp (cost and time) for each operation processing. However, the two sides are not fair because the ratio of converter (log and exp) to operations is not one-to-one and not very small ratio (cannot be neglected) at the same time.

The second drawback in the discussion is that it compared the operations with the same weights for each operation while that is not true in the computer data processing. For example, the integer addition operation is use a more than 4 times larger than the division operation as will be shown in section 4.

The third drawback is found in some works like Chen and Chow [2]. Here, the comparations of the speed depend on the number of clocks, but a fair comparations must depend on the execution time that depend on the number of clocks in additional to the time of each clock (note that some works use a low storage internally that result in small number of clocks with large clock time hence a long execution time).

This paper will discuss the two main types of numeric systems for FPS that used IEEE 754 and the format of the internally LNS in section 2. Section 3 classify the operations in a symmetric groups and calculate the number of clocks for each operation in FPS and in LNS, the weight of each operation in the computer data processing field will be deduced in section 4. Section 5 shows the parameters of the optimal processor, with its design in section 6, while the implementation of this process is discussed in section 7. Section 8 compare the result of the optimal process with the most powerful previous works. Section 9 has the conclusions that can be used to improve the characteristics of the processor.

2- FPS and LNS

2.1 Floating point [3]

The floating point number has a multi formats, the most formats used is the IEEE 754. In this format a floating point number F has the value:

$$\mathbf{F = -1^{S} \times 1.\, f \times 2^{E}} \qquad (1)$$

Where **S** is the sign, **f** is the fraction, and **E** is the exponent, of the number. The mantissa is made up of the leading "1" and the fraction, where the leading "1" is implied in hardware. This means that for computations that produce a leading "0", the fraction must be shifted.

The only exception for a leading one is for gradual underflow. The exponent is usually kept in a biased format, where the value of **E** is

$$\mathbf{E = E^{true} + bias.} \qquad (2)$$

The most common value of the bias is

$$\mathbf{2^{e-1} - 1}$$

Where **e** is the number of bits in the exponent. This is done to make comparisons of floating point numbers easier. Floating point numbers are kept in the following format:

S	Exponent E	Unsigned Significant f
1 bit	e bits	m bits

The IEEE 754 standard sets two formats for floating point numbers: single and double precision. For single precision, **e** is 8 bits, **m** is 23 bits and **S** is one bit, for a total of 32 bits. The extreme values of the exponent (0 and 255) are for special cases, so single precision has a range of $\pm(1.0 \times 2^{-126})$ to $\pm(1.11... \times 2^{127}) \approx \pm 1.2 \times 10^{-38}$ to $\pm 3.4 \times 10^{38}$ and resolution of 10^{-7}. For double precision, where **m** is 11 and **e** is 52, the range is $\pm(1.0 \times 2^{-1022})$ to $\pm(1.11... \times 2^{1023}) \approx \pm 2.2 \times 10^{-308}$ to $\pm 1.8 \times 10^{308}$ and a resolution of 10^{-15}.

2.2 LNS [4]

Logarithmic numbers can be viewed as a specific case of floating point numbers where the mantissa is always 1, and the exponent has a fractional part. The number **A** has a value of

$$\mathbf{A = -1^{SA} \times 2^{EA}} \qquad (3)$$

Where **SA** is the sign bit and **EA** is a fixed point number. The sign bit signifies the sign of the whole number.

EA is a 2's complement fixed point number where the negative numbers represent values less than 1.0. In this way LNS numbers can represent both very large and very small numbers. The logarithmic numbers are kept in the following format:

SA	EA	
Sign	integer	fraction
1 bit	k bits	j bits

If k=e and j=m the LNS has a very similar range and precision as floating point numbers. For k=8 and l=23 (single precision equivalent) the range is $\pm 2^{-129+ulp}$ to $\pm 2^{128-ulp}$, $\approx \pm 1.5 \times 10^{-39}$ to $\pm 3.4 \times 10^{38}$ (ulp is unit of least precision). The double precision equivalent has a range of $\pm 2^{-1025+ulp}$ to $\pm 2^{1024-ulp}$, $\approx \pm 2.8 \times 10^{-309}$ to $\pm 1.8 \times 10^{308}$. Thus, an LNS representation covers the entire range of the corresponding floating-point version.

2.3 Computer Arithmetic Operations in LNS [4]

Arithmetic operations in LNS are different from the normal due to the nature of the logarithmic number. If $X = \log_b |X|$ and $Y = \log_b |Y|$ then,

$$Z = X \times Y \Rightarrow Z = X + Y$$

$$Z = X \div Y \Rightarrow Z = X - Y$$

$$Z = X + Y \Rightarrow Z = X + \log_b \left(1 + b^{X-Y}\right) \qquad (4)$$

$$Z = X - Y \Rightarrow Z = X + \log_b \left(1 - b^{X-Y}\right)$$

The operations for multiplication and division will become addition and subtraction. Multiplication and division now can be done within one instruction which will be a lot faster in floating point number system.

3- Execution Time of the Operations

There are a large number of methods to classify the operations such as the priority, type of output, extra. This section will classify the operations in groups depending on the requirements of execution in the processor. These groups are:

a- integer add/sub, **b-** float add/sub, **c-** integer multiplication, **d-** float multiplication, **e-** division (normally float), and **f-** other operations such as x^a, a^x, log x, exp x, cos x, sin x, etc. These operations in fact is a functions (normally floating).

The second aim of this section is finding the execution time for the above operations in FPS, LNS and hybrid FPS/LNS processors measured by the number of clocks for each operation. The number of clocks for the operation depends on the type of operation and the characteristics for the processor. The number of execution clocks depends on the number of bits in ALU, number of registers, additional units and finally the control hardware and software design. The characteristics for the processors that will be use in this section are the ALU has 23-bit, 10 registers each one has 23-bit, extra auxiliary units and ideal hardware design and software for the control units.

The number of clocks for each operation using IEEE 754 standard and processor of 23-bit ALU can be summarized in table (1).

Note that, all the operations in table (1) has a specific characteristics therefore it has a fixed number of clocks except the type 6 (other) such as functions solved by Taylor Series [5] and has an approximate number of clocks. For example a high accuracy exponential function [5] to be solved in Taylor Series required 20 terms each has in average 10 multiplications, 1 division and 1 addition. So in total the exponential function required 200 multiplications, 20 divisions and 19 additions (not all functions has this requirements). However, the other operations will need as an average 100 floating multiplications, 10 divisions and 10 floating additions.

Table (1): The number of clocks using IEEE 754 and 23-bit ALU processor.

Operation	FPS clocks	LNS clocks	Hybrid FPS/LNS
Integer add/sub	1	25*	1
Floating add/sub	4	125*	4
Integer multiplication	15	1	1
Floating multiplication	24	4	4
Division	72	4	4
Other*	3160	1690	480
FPS to LNS converters (FLC)	----	M1**	N1**
LNS to FPS converters (LFC)	----	M2**	N2**

* Approximation value.

** times to transfer from number system to another.

4- Weights of the Operations

This section will try to answer on the question "What is the ratio of used (execution in computer) for each operation in section 3 in the real world?".

The answer comes from two sources. The first source comes from a statistical search by using the internet to deduce these ratios in the field of computer data processing. The second source comes from using the MS-DOS instructions to monitor some computers in the library of the computer department. The final result in table (3) is the average of both results for the two sub-sections given below.

4.1- Statistics of Operations in Computer Data Processing

This sub-section will try to answer on two questions, first "what are the main fields in the world of the computer data processing and what is the weight of each one?" and the second is "what is the mathematical requirements of each field?".

The answer to the first question will be deduced from the ratio of the works (paper and thesis) for these fields on the internet.

The answer of the second question will require making some logical rules then applied them with examples. These rules can be summarized as, first rule (all the results will be calculated statistically), the second rule (the statistical calculation will be specialized to the most famous data processing fields such as FFT, wavelet, filtering, neural networks and other applications such as DCT, coding and matrices). The third rule (take the most used case in each type). The fourth rule (normalized the number of operations and multiplies them by 100%). These rules can be explained in the following examples.

Example 1: FFT

To calculate the number of operations in FFT [6] we must decide on the type of FFT, the number of points and the type of input data. Applying rule three will select Radix-2, 256 point with integer input data. The execution of this transform require that 256 integer add/sub for the first stage, 256x255x2 floating add/sub for other stages, 256 other (256 exp to calculate W_{256}^{i}) and 256x256x2 (the number 2 in here is for complex) floating multiplications. The fourth rule for 256-point Radix-2 required 0.1 integer add/sub, 49.8 floating add/sub, 0.1 others and 50 floating multiplications. This example also required a 256 FLC and 256 LFC for LNS processor and 256x256x2 FLC and 256x 256x2 LFC for hybrid FPS/LNS processor.

Example 2: NN

This example deal with a neural network (NN) [7] that has three layers to detect the voice of 40 persons. This NN has 256 nodes in input layer, 200 nodes in hidden layer and 40 nodes in the output layer.

The execution of this NN require 256x200 integer add/sub, 256x200 floating multiplications and 200 activation function (other) for the hidden layer, 200x40 floating add/sub, 200x40 floating multiplications 40 activation function (other) for the output layer.

Under fourth rule the NN required 43 integers add/sub, 6.8 floating add/sub, 50 floating multiplications and 0.2 activation functions (others). This example required also 256 FLC and 40 LFC for LNS processor and 256x200+200x40+40 FLC and 256x200+200x40 LFC for hybrid FPS/LNS processor.

The second part of this section is to use the above results in order to calculate the approximation results as in table (2). The final results for the last column in this table are calculated using equation (5).

$$\text{Total weighted sum} = \left(\sum \text{Weight of operation} * \text{Operation ratio}\right) / 100\% \qquad (5)$$

Table (2): The most computer data processing fields with their weights.

operation	FFT	Wavelet	filtering	neural networks	Other applications	Total weighted sum
Using Weight of the operation	14	12	7	16	51	100
Integer add/sub	0.1	57	0	42.8	28	28
Floating add/sub	49.8	0	50	7	20	22
Integer multiplication	0	43	0	0	12	11
Floating multiplication	50	0	50	50	26	32
Division	0	0	0	0	9	4.5
Other	0.1	0	0	0.2	5	2.5
M1*	0.1	14	0.1	0.3	12	8
M2*	0.1	13	0.1	0.1	9	6
N1*	50	50	50	51	41	45
N2*	50	50	50	50	38	44

*see table (1)

4.2- Monitoring of the Operations in Computer

The results of monitoring some computers in the library of computer department using an assembly program can be summarized in the third column of table (3). This program uses some facilities of MS-DOS with program-counters [8, 9]. This program counts each assembly instruction at the time of fetching. Then the program calculates the number of operations in the compiled programs and stores the results in a file. The program is designed to monitor the classic operations (add, sub, mult and div) without possibility of separation for the sources of these operations. The result of monitoring is used to improve the result given in table (2).

Table (3): The results of monitoring in the library with the results of table (2).

Operation	Statically	Monitoring	Approximation total results
Integer add/sub	28	69	30
Floating add/sub	22	NA^+	20
Integer multiplication	11	26	10
Floating multiplication	32	NA^+	30
Division	4.5	5	7
Other	2.5	NA^+	3
M1*	8	---	8
M2*	6	---	6
N1*	45	---	45
N2*	44	---	44

NA^+ means not available value.

Table (3) can be summarized in equation (6) that used to calculate the number of clocks/operation (CPO).

$$\begin{aligned} CPO = (&30*Integer\ add/sub + 20*Floating\ add/sub + 10*Integer\ multiplication \\ &+ 30*Floating\ multiplication + 7*\ Division + 3*\ Other \\ &+\ conversation\ ratio*conversation\ clocks)/100 \end{aligned} \quad (6)$$

That can be rewritten as equation (7) if we take the other operations required in average to 100 floating multiplications, 10 divisions and 10 floating additions.

$$\begin{aligned} CPO = (&30*Integer\ add/sub + 50*Floating\ add/sub + 10*Integer\ multiplication \\ &+ 330*Floating\ multiplication + 37*\ Division \\ &+\ conversation\ ratio*conversation\ clocks)/100 \end{aligned} \quad (7)$$

5- Optimal Processor

The combinations of sections 3 and 4 can be shown in table (4). The calculations of this table come from the multiplication of the results in table (1) with the last column in table (3). The approximation sum in the last row can be calculated from equation (8) below:

$$\text{Approximate sum ratio} = \frac{\sum \text{clocks of the operation for processor}}{\sum \text{clocks of the operation for FPS processor}} * 100\% \quad (8)$$

Table (4): Number of clock*ratio for each operation using IEEE 754 and 23-bit ALU Processor.

Operation	FPS CPO	LNS CPO	Hybrid FPS/LNS CPO*100
Integer add/sub	30	750	30
Floating add/sub	80	2500	80
Integer multiplication	150	10	10
Floating multiplication	720	120	120
Division	504	32	28
Other	9480	5070	1440
FPS to LNS (FLC)	00	8*M1	45*N1
LNS to FPS (LFC)	00	6*M2	44*N2
Approximate sum	10964	8482 +14M	1688 +89N
Approximate sum ratio	100	77 +0.13M	15 +0.81N

The results of table (4) shows that:

1- The speed of the LNS processor depends on the value of M that required at least 4 clocks in [1]. In result the minimum approximation sum ratio is about 77.

2- The speed of the hybrid processor depends on the value of N that required at least 4 clocks in [1]. In result the minimum approximation sum ratio is about 18.

3- The speed of the hybrid processor depends mainly on the value of N. However, the reduction of N will increase the speed directly.

The method of FLC and LFC in [1] is a powerful method because it will give a processor that has a speed equal to 5.5 times higher than the classic (FPS) processor with 10 times higher cost. The additional key to increase the speed is by using a parallel processing.

This paper proposed a new method that depends on two points. First, it uses the parallel processing to separate the FLC and LFC units with a twin memory one for floating form and the other for LNS form. The second form use the piecewise to FLC and LFC with a good optimization between number of pipelines and clock time.

6- The Proposed Design

The study of the results and conclusions reached in section 5 help us to propose a high speed and low cost hybrid processor. The proposed design is a hybrid FPS and LNS which expected to have very fast process, since it depends on four main points:

1- Use parallel processing to process (fetch and execute) the operations and convert the data at the same time as shown in Fig (1).
2- It has a twin memory with the same addresses so each one is a 32-bit/word. The first one store the values of variable in floating point form while the other store the same variables in same address but in the LNS form.
3- Extra fast LFC and FLC that depend on the piecewise approach and direct implementation for multipliers in LFC and FLC units as shown in figures (2- 4).
4- Its units have a pipeline internally approach with a good optimization between the internal pipeline and the clock rate similar to that given in reference [10].

6.1- The Registers Unit

It has 10 registers (R0 to R9) each one is a 23-bit used as a temporizer internally memory. R0 is connected to ALU1 as main register and R1 is connected to ALU2 as main register.

6.2- The 23-Bit General ALUs

They are the execution units of the processor that use for all logic operations (add, sub, etc.) for floating point and LNS variables. For floating variables ALU1 is used for mantissa processing while ALU2 is used for exponential and sign part that work at the same time. For LNS variables ALU1 is used for fraction processing while ALU2 is used for integer and sign part that work at the same time. This will reduce the number of CPO in the processor.

6.3- The I/O Unit

It is the input and output unit for the external data that is controlled by using the central unit. The main component is a buffer with pack processor.

6.4- The Floating Point and LNS Memory

This unit is an internally or externally (but it has a direct connection with processor) unit for processor depending on the media of its implementation. This unit has twin memory the first store the floating form of variables while the second store the LNS form of variables. The two memories word size is 32-bits; it is managed as one memory that has the same address for same variables value.

Each word address has an additional bit in MSB called type-bit used internally to select the type of variable. The value of the type-bit received from the central unit depends on the type of the instruction code. The implementation of this unit required 2 to 16 KB depending on the designer selection or on the available RAM in the platform.

6.5- The FLC Unit

This unit is the more important unit in the processor design because the properties of this unit are the main factor in the processing speed. The basic design idea to implement this unit in Fig (2) depends on the piecewise approach that used to divide the interval from [1, 2) to 128 straight line, the logarithm value of this interval fall in the interval [0, 1). Each line has initial value (shift) found from first 7x7 LUT that used the first 7 MSBs of the floating number in input and the output of the shift LUT is a 7-bits. Also the line has slope value found from second 7x7 LUT that used the first 7 MSBs of the floating number as input and the output slope value is a 7-bits.

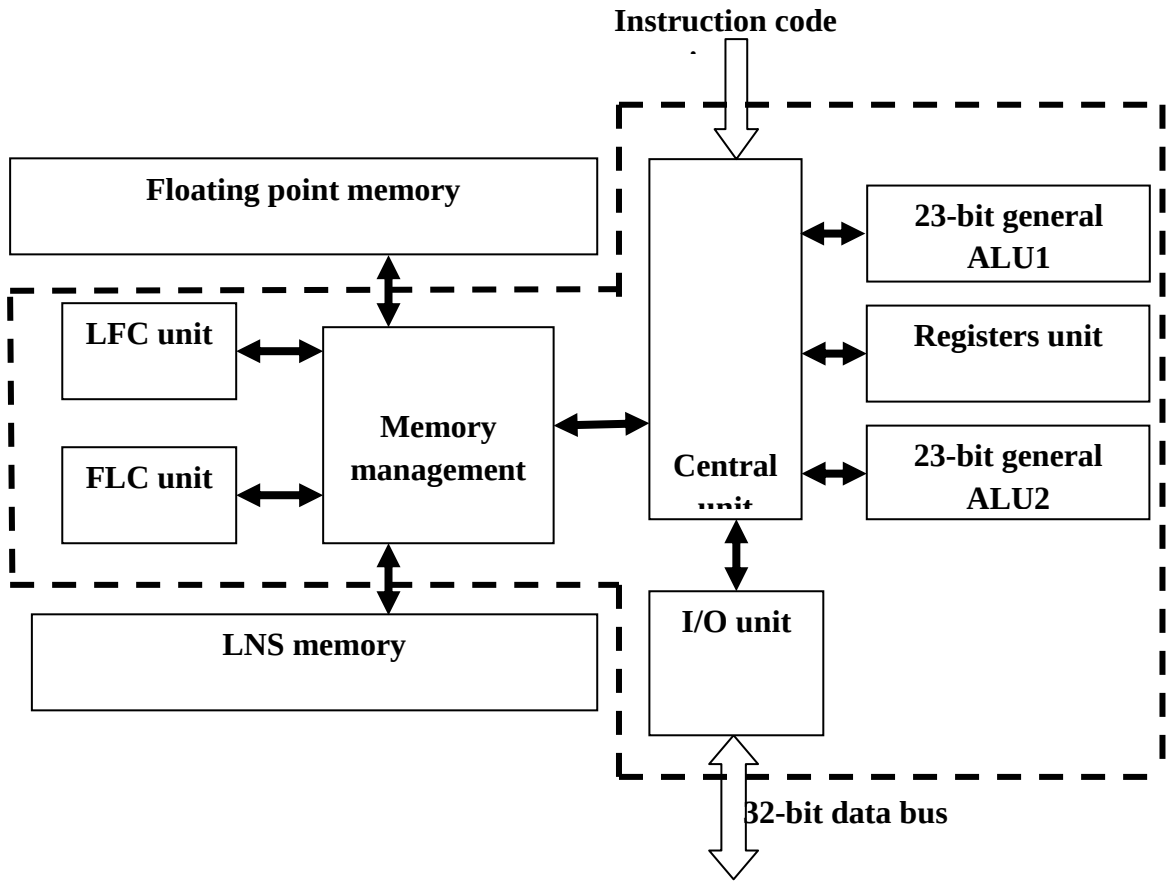


Fig (1): The main blocks of the proposed processor.

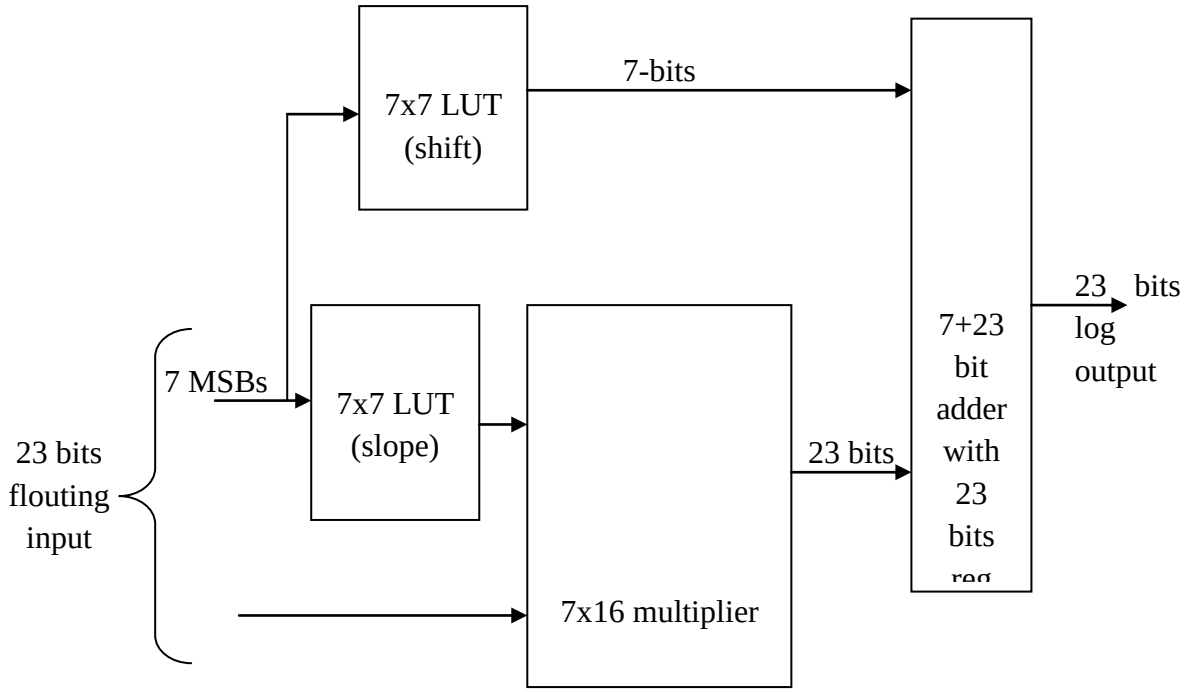


Fig (2): The basic design of FLC unit that used the piecewise approach.

The shift value is the course part of the output while the fine part calculated using the slope value with the 16 LSBs (reminder) of the input multiplied by the slope value. The 7x16 multiplier is the most important sub-unit in this unit because it is the most complex and lower speed part. The implementation of this sub-unit can be made using one 18x18 HMULT but this approach has a low speed. The other approach in Fig (3) required three LUTs used as multipliers and two stage shift and adder with store to sum and sort the output 23-bit data. This combination gives a high speed and low cost unit with low error output. The FLC unit implement to give an optimal storage separation that will balance between the number of clocks (latency) and the maximum delay (clock time). It has two internally storage points first in the output of multiplier and the other in the total output. That will give two latency clocks with a maximum delay of 7 gates if this unit is implemented in FPGA.

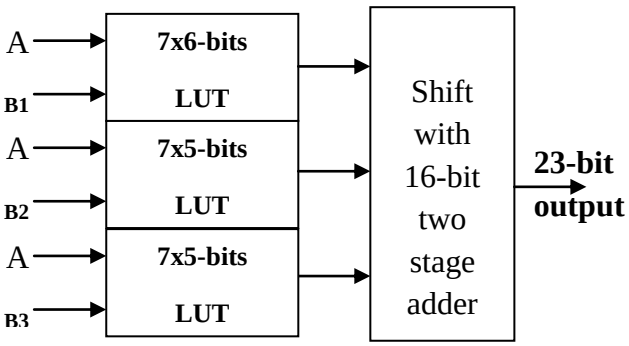
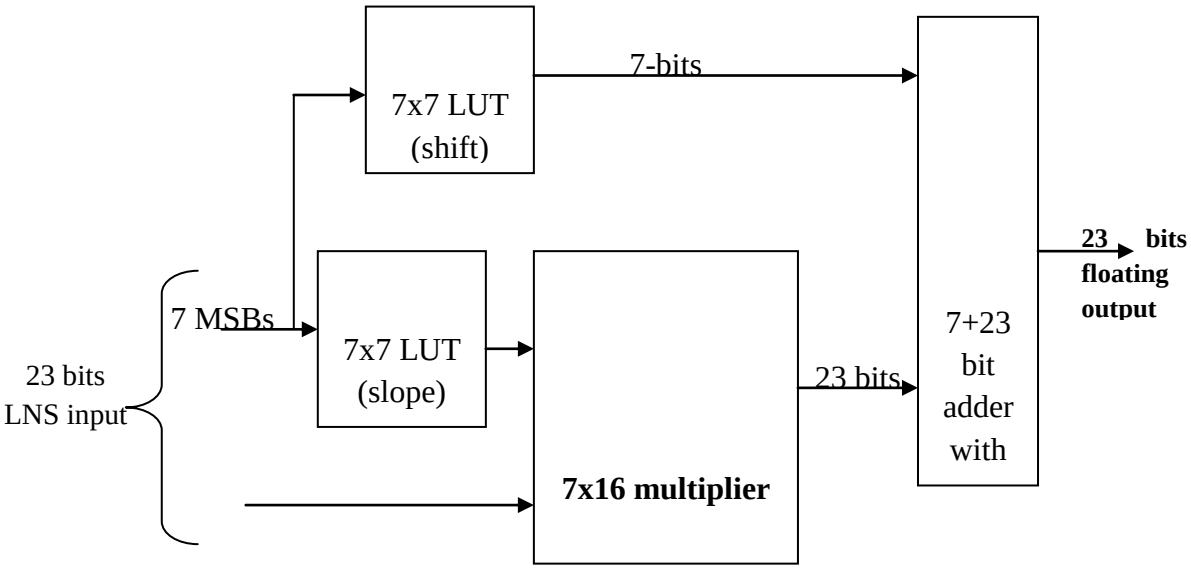


Fig (3): 7x16-bit multiplier using three LUTs as sub-multipliers.

6.6- The LFC Unit

This unit in Fig (4) has been built with the same idea and form of the FLC unit with



reveres intervals of input and output. Also this unit has same speed and cost of FLC.

Fig (4): The basic design of LFC unit that used the piecewise approach.

6.7- The Memory Management Unit

This unit is controlled the read and write in the twin memory also it managed the timing of used LFC and FLC units. Moreover, its tasks can be summarized in the following tasks:

1- Receive the value and type (floating or LNS) of variable from the central unit.

2- If the variable is float store it in floating memory unit and send another copy to FLC, then receive the LNS equivalent value and store it in LNS memory unit at same address of floating memory unit.

3- If the variable is LNS store it in LNS memory unit and send another copy to LFC, then receive the floating equivalent value and store it in floating memory unit at same address of LNS memory unit.

4- Read the values of variables and sending them to the central unit with the specific type depending on the order of the central unit.

The cost and timing of this unit depends on the internal details of the implementation. The basic design shows that this unit required in FPGA to less than 100 slices.

6.8- The Central Unit

This unit is the core of the processor that controlled on the all the fetch and execute instructions in the processor. It has a multi sub-unit such as integer to floating sub-unit, control unit, log/float selector. Moreover, this unit has the following tasks:

1- Receive the data and instruction code.
2- Classify the operations to float or LNS data requirement.
3- Ask the memory management to send or receive the data.
4- Send the data with operation code to ALU1 and ALU2.
5- Read and write the data in registers unit.
6- Convert the integer variables to float form when it's necessary.
7- The other entire control mission in details of the processor.

The cost of this unit depends on the internal details of the implementation while this unit will dissipate 1 clock to fetch the instruction with maximum 3 clock latency. The basic design shows that this unit required in FPGA to less than 200 slices.

7- Implementation of the Proposed Processor

The details for the implementation of the proposed processor using xc2v6000 [11] required about 1500 slices as shown in table (5).

The actual speed for the proposed processor can be calculated from equations (6) and (7). So calculation result for the number of clocks for each operation is shown in table (6).

Table (5): The implementation of the proposed processor using xc2v6000.

Unit name	Cost Slice	Max delay nsec	No of clocks	Latency clocks
Registers	230	1.72	1	0
ALUs	60	14.28	1	0
I/O	64	4.76	1	1
FLC	414	19.04	2	2
LFC	414	19.03	2	2
Memory management	100	4.76	1	3
central	200	4.76	1	2

The overall design	1482	19.24	--	5

Table (6): The details of the fetch and execution speed in the proposed processor.

Operation	Ratio of operation	Number of clocks	Proposed CPO
Integer add/sub	30	1	30
Floating add/sub	20	3	60
Integer multiplication	10	1	10
Floating multiplication	30	2	60
Division	7	2	14
Other	3	250	750
FLC	Δ	3	3Δ
LFC	Δ	3	3Δ
The approximation sum ratio	----	-----	8.5

Δ a small value.

Note that in the ideal case the converting units (FLC and LFC) not effected on the processing time because they work parallel with the other units. But, practically they have some intersection because the latency in the converting units, this intersection depends on the applications and the type of programming that will be used. However, this value is small and can be neglected for most cases.

8- Comparations With Previous Works

This section compares the speed and cost of the most three powerful pervious works and one from the standard floating processor design with the proposed processor as shown in table (7).

1- First work is a standard design floating point processor [2] that requires 1247 slices with deduced max delay under xc2v6000 Xilinx FPGA chip of about 16.66 nsec. The average of the operation can be calculated using the tables of reference with equations (6) & (7). It is equal to 110 clocks per operation (CPO) or 1833 nsec in the average for each operation.
2- Second work is a hybrid FPS/LNS processor by Chichyang Chen and Paul Chow [2] that has more than one model. The most used model require to 3415 slice. The design for this process is very complex and heavy and the deduced max delay under xc2v6000 Xilinx

FPGA chip is about 124 nsec. The average of the operation can be calculated using the tables of reference with equations (6) & (7). It is equal to 14 CPO or 1736 nsec in the average for each operation.

3- Third work is a LNS processor by Mark G. Arnold [12] that has more than one model. The most used model require to 3904 slice. The design for this processor depends on the pipeline approach that increases the speed and reduced the cost but it has a large latency. The average time of the operation can be calculated using the tables of reference with equations (6) & (7). The max delay under xc2v6000 Xilinx FPGA chip is about 95 nsec for each operation.

4- Fourth work is an LNS processor by M. Haselman et al. [1] that is the most powerful processor in the previous works. The sum (from tables) of all its parts required 2054 slice. The design for this process depends on the parallel and independent units approach that increase the speed but increase the cost, also the design use a good approaches to implement the add/sub units and converting units, these approaches will reduce the cost and increase the speed at the same time. However, the average time of operation can be calculated using the tables of reference with equations (6) & (7). The max delay under xc2v6000 Xilinx FPGA chip is about 84 nsec for each operation.

5- The proposed work is a hybrid FPS/LNS parallel processor that required 1482 slice. The design for this process has a max delay under xc2v6000 Xilinx FPGA chip of about 58 nsec.

Table (7): Comparations of speed and cost for pervious works with the proposed processor.

work	Cost slice	Max speed nsec	CPO	Time of operation nsec
Floating point processor [2]	1247	16.66	110	1833
Chichyang Chen and Paul Chow [2]	3415	124*	14	1736
Mark G. Arnold [12]	3904	NA^+	NA^+	95
M. Haselman et al. [1]	2054	NA^+	NA^+	84
proposed	1482	19.24	3	58

- Approximation value.
- NA^+ means not available value.

The results in table (7) are rewritten as rational values with respect to the floating point processor to give a more focusing view as shown in table (8). However, the rational cost or

rational speed can not give a real view to the efficiency of the design. Another factor (efficiency factor) will be added in last column of table (8) that will sum the rational advantages of the processor design.

It can give a good idea about the processor properties and efficiency for the method of its design. The efficiency factor can be calculated using the following equation (9).

$$efficiency\ \ factor = \frac{Rational\ speed}{Rational\ \text{cost}} \qquad (9)$$

Table (8): Comparations of the efficiency factor for the pervious works with the proposed processor.

work	Cost slice	Max speed MHz	Rational cost	Rational speed	efficiency factor
Floating point	1247	0.546	1	1	1
[2]	3415	0.576	2.743	1.055	0.385
[12]	3904	10.526	3.136	19.251	6.139
[1]	2054	11.905	1.65	21.802	13.213
proposed	1482	17.241	1.19	31.574	26.533

The results in table (8) can be summarized in the following points:

1- The proposed design is the higher speed for all types. It has a speed equal to bout 1.5 times faster than the maximum speed of the previous works and more than 30 times over the classical floating processor.
2- The propose design is the lower cost for all types. It has a cost equal to bout 0.72 times less than the minimum cost of the previous works and about 1.2 times over the classical floating processor.
3- The efficiency factor for the propose design is the higher value for all types. It has an efficiency factor equal to bout 2 times higher than the maximum gain factor of previous works and more than 26 times over the classical floating processor.

9- Conclusions

This work helps the designer to design and implement a good parameters processor. These conclusions can be summarized as:

1- A fair comparations between the processors must include the converting times and costs.

2- The speed comparations between the processors must depends on the processing (fetch and execution) times and not affected by the number of clocks.
3- The speed comparations between the processors depends on the ratios of using each operation multiplied by its execution time.
4- The hybrid FPS/LNS is the highest speed type while the classic FPO processor is the slower speed.
5- The parallel processing is the optimal solution to increase the speed of the hybrid FPS/LNS processor and increase the value of CPO.
6- The FLC and LFC are the most complex units, so the cost and speed of the hybrid processor depend on those characteristics.
7- The good optimization between the internal pipeline and the maximum delay (clock) time is an important factor to increase the speed of the processor.
8- The twin memory approach that has same addresses to store the values of variable in floating point and LNS form at the same time is a good approach to increase the speed of the processor.
9- The piecewise approach with FPGA LUTs gives a low cost and high speed for the FLC and LFC units.
10- The use of LUTs and adders to design and to implement the multiplier in FPGA increase the speed and reduce the general cost.
11- The processor that used a parallel processing and hybrid approach with a piecewise approach for the FLC and LFC units is the most powerful solutions to increase the speed and reduces the cost of the processor.

References

[1] M. Haselman, M. Beauchamp, A. Wood, S. Hauck, K. Underwood and K. Hemmert, "A Comparison of Floating Point and Logarithmic Number Systems for FPGAs", *IEEE Symposium on Field-Programmable Custom Computing Machines*, pp 181-190, 2005.

[2] C. Chen and P. Chow. "Design of a Versatile and Cost-Effective Hybrid Floating-Point/LNS Arithmetic Processor." In GLSVLSI '07: Proceedings of the 17th Great Lakes Symposium on VLSI, *ACM, pages 540-545, March 2007.*

[3] S. W. Smith. "The Scientist and Engineer's Guide to Digital Signal Processing", Second Edition, California Technical Publishing, 1999.

[4] N. Wing, "Logarithmic number system and its application to image processing", Oregon State University School of Electrical Engineering and Computer Science, ECE 577 - Fall 2003, ngwi@engr.orst.edu.

[5] J. Mathews and K. Fink, "Numerical Methods Using Matlab", 4th Edition, ISBN: 0-13-065248-2, Prentice-Hall Inc. Upper Saddle River, New Jersey, USA, 2004, http://vig.prenhall.com.

[6] C. Van Loan, "Computational Frameworks for the Fast Fourier Transform", Frontiers in Applied Mathematics, SIAM, Ithaca, New York, 1992.

[7] Z. Runxuan, "Efficient Sequential and Batch Learning Artificial Neural Network Methods for Classification Problems", Ph. D thesis, School of Electrical & Electronic Engineering, 2005.

[8] G. Krzysztof, "Program-counter-based prediction in operating systems" Ph.D. thesis, Purdue University, 2005.

[9] R. Hyde, "The Art of Assembly Language Programming", the Art of Assembly Language Programming.htm, 1996.

[10] G. Hinton, D. Sager, M. Upton, D. Boggs, D. Carmean, A. Kyker and P. Roussel, "The Microarchitecture of the Pentium 4 Processor ", Desktop Platforms Group, Intel Corp, Intel Technology Journal Q1, pp 1, 2001.

[11] XC2V6000-6BF957C datasheet pdf datenblatt - Xilinx, Inc – "Virtex-II 1.5V Field-Programmable Gate Arrays", ALLDATASHEET.htm., Contact us Privacy Policy Bookmark Link Exchange Site link Manufacturer, 2008.

[12] M. Arnold, "A VLIW Architecture for Logarithmic Arithmetic" *dsd*, Euromicro Symposium on Digital Systems Design (DSD'03), University Bethlehem, p. 294, 2003, PA 18015 USA marnold@eecs.lehigh.edu.