Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Implementation of digital electronic arithmetic and its Application

Khader Mohammad, Sos Agaian and Fred Hudson
Electrical and Computer Engineering Department
University of Texas at San Antonio
1 UTSA Circle, San Antonio, TX 78249-0669
Email: hajkhader@gmail.com, sos.agaian@utsa.edu, fred.hudson@utsa.edu

Abstract**—** This Parameterized Digital Electronic Arithmetic (PDEA) model replaces linear operations with non-linear ones. In this paper we introduce a hardware implementation of the parametric image-processing framework that will accurately process images and speed up computation for addition, subtraction, and multiplication. Particularly, the paper presents the design of arithmetic circuits including parallel counters, adders and multipliers based in two high performance threshold logic gate implementations that we have developed. We will also explore new microprocessor architectures to take advantage of arithmetic. The experiments executed have shown that the algorithm provides faster and better enhancements from those described in the literature. Its potential applications include computer graphics, digital signal processing and other multimedia applications.

Keywords**—**Digital Electronic Arithmetic, Logarithmic Number System, Implementation of Logarithmic Image Processing Operations.

I. Introduction

In recent years, improvements in imaging technology have made available an incredible array of information in image format. Image processing is the system of mathematically transforming an image from one form to another, generally, changing some characteristics of the image [1]. In order to utilize digital images effectively, specific techniques are needed to reduce the number of bits required for their representations. Different coding techniques have been developed for use in simplifying and optimizing these reductions. "For a given information source like an image there is a coding technique which permits a source to be coded with an average code length as close as to the entropy of the source as desired," as published in Shannon's Information Theory [2].

Traditionally, image processing uses the concept of linear operations to manipulate images, but since this is inherently a non-linear process, accuracy issues may rise. First, the resulted pixel intensities lie outside the range (0, M), are clipped and causing a loss of information. Second, linear operations typically do not yield results consistent with the physical phenomena (as perceived by human eye/brain). Solving these common problems in an image processing context, the methods can be utilized. Some study shows color still images of 512 by 512 pixels that were first converted to the corresponding monochrome images in order to generate their contours using the sobel filter [30].

This paper consists of several parts. Section II and III present the background and the Parameterized Digital Electronic Arithmetic (PDEA) format, benefits (advantages) and limitation (disadvantages). Section IV presents applications, assumptions and hardware implementations, and Section V presents results and a conclusion.

II. Background: The logarithmic image processing

The logarithmic image processing (LIP) approach was originally developed by Jourlin and Pinoli and formally published in the mid-1980s for the representation and processing of images valued in a bounded intensity range [15]. The Logarithmic Image Processing (LIP) model treats images as light absorption filters, as shown in Figure 1 [4]. The programmable logarithmic image processing PLIP model was developed in [1]. The PLIP basic operations are directly defined as follows:

$$g1 \oplus g2 = g1 + g2 - \frac{g1g2}{\gamma(M)} \qquad (1)$$

$$g1 \Theta g2 = kM\frac{g1 - g2}{K(M) - g2 + \varepsilon} \qquad (2)$$

$$g1 \otimes g2 = \varphi^{-1}(\varphi(g1).\varphi(g2)) \qquad (3)$$

$$\varphi(g) = \lambda(M).\ln^{B}\left[1 - \frac{g}{\lambda(M)}\right] \qquad (4)$$

$$\varphi(g)^{-1} = -\lambda(M).\left[1 - \exp\left(-\frac{g}{\lambda(M)}\right)^{1/B}\right] \qquad (5)$$

Where, the parameters $\gamma(M)$, K(M), and $\lambda(M)$ are linear functions of the type, $\gamma(M)$ = AM + B; A and B are constant parameters.

Arithmetic operations in PDEA are different from the normal Boolean approaches due to the nature of the logarithmic number. The operations for addition, subtraction, multiplication and division will become combinations of almost all, but resources can be shared and enabled for the right operation. This will be a lot faster. Originally, no method was used to select optimal parameter values. In this Implementation we selected these parameters as shown on the section to follow. Here is a list of benefits that can be gained from using the PDEA :

1. System has parameters, such as M, B, γ and λ that can be set for the corresponding operation.
2. It can be shown that PDEA operations have associatively and commutatively.
3. Have low power consumption

4. Represent dynamic range and precision comparable to the linear system

We developed a new algorithm that is defined by equations 1, 2, 3, 4, and 5 [1]. Because we are implementing a non-linear process, accuracy issues arise. For example when we compute the exponential or the natural logarithm we iterate up to the precession point. These implementations help improve the solution of common problems in an image processing context. The logarithmic implementation can be utilized to improve many areas of signal processing. Further development can build on this approach to implement other computational functions. Therefore, designing these functions with technology that is high-speed, low power, and regular in layout brings about design approaches that are of substantial research interest. In this regard, when we combined the algorithms in one circuit, we made a significant reduction in the time for computing different functions. Our goal besides having new algorithms is to reduce computation time by sharing resources and reducing area on an IC chip. Accuracy of individual building blocks, as well as overall correct functionality is the main goal of the implementations as is true for any circuit design.

III. Filtering Using Logarithmic image processing

Most of the applications of PDEA being developed today are mainly related to audio applications and other digital signal processing. Besides these applications, researchers are also paying more attention to the potential use of LIP applied to image processing. A low pass filter is a good candidate for calculation to apply the new algorithm we are introducing.

A low-pass filter is an operator that does not affect low frequencies and rejects high frequencies [3]. Applying these functions on a low pass filter and comparing the result is done in Matlab. We change the add/subtract operations and compare the results of the original code (which uses normal operation vs. the modified approach using these new functions). The graphs show the results of the comparison. Figure 2 show the time and frequency impulse response, where A, C are the original response, and B, D are the proposed formula results. Another Time and frequency original step response shown in E, G and F, H with proposed formula. Time and frequency of Random response, I, J is the original, K, L with proposed formula

IV. Hardware Implementations and Performance

Several methods have been suggested [1, 2] for possible incorporation into the hardware of computers to aid in the estimation of numerical significance. These include binary number systems and gray numbers.

A few of the different families of DSP Hardware can be identified [14] as indicated here. DSP processors: These chips

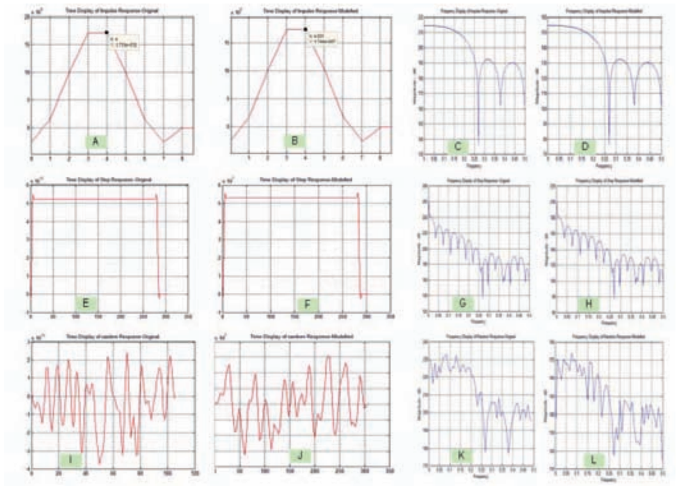


Figure 1. Matlap comparison results

are essentially RISC architecture processors with specialized hardware for DSP algorithms. They can be programmed to perform a wide range of algorithms. DSP cores: These blocks are large macro-blocks with the functionality of a DSP processor that can be embedded in ASICs. These cores include more specialized hardware for signal processing algorithms.

Application Specific DSP Chips: These chips are designed for specific algorithms and are used where programmable approaches cannot achieve the required performance. The nature of DSP algorithms demands different requirements than generic computational algorithms. Multipliers and adders are indispensable blocks of any DSP algorithm. DSP algorithms operate on a fixed rate of data, governed by the level of fabrication technology and the system clock, putting strict restrictions on the speed of individual operations.

The Gray number system is a non-monotonic representation where two consecutive numbers differ in only one of their digits. For some applications such as counters or low-power address bus decoders, the Gray number system can help reduce the power consumption; when compared to consecutive numbers, there will be fewer transitions (and lower associated power consumption.

The main disadvantage of representing numbers this way is that it is not well suited for standard Boolean arithmetic operations, primarily, because Gray numbers are non-monotonic. The lower power consumption issue is also arguable, as the extra power used for the generation of Gray numbers may decrease the aforementioned power savings considerably [13]

The logarithmic number system (LNS) was introduced in the 1970s. However, not many systems were built that based on LNS. Moreover, one of the major reasons they were not developed is that there are tradeoffs of performance and memory usage when implementing LNS. The highest precision of LNS built so far is 32-bits. LNS often require extensive look-up tables. Higher precision requires a much larger look-up table for computation and the cost may not be worth developing the system. The limitations for LNS [12] are: addition and subtraction are more complex and large-scale look-up is required in order to achieve high precision. We also cannot match completely the effects of conventional floating-point numbers. It is also too expensive to implement since the memory cost goes up and the processor technology becomes more advanced and more complex. In addition, extra logic/software is needed to deal with the zero divide by zero case.

In this paper the implementation of the Logarithmic Image Processing class of algorithms [8-9] is presented as a case study. The operation kernels are identified and their mappings onto an instruction set are described.

Next, some special processor concepts that were used to achieve a good trade-off between performance and flexibility are described in more detail, highlighting the issue of power consumption. In particular, this filtering case study will be used as a test bench for the new implementation of non linear operators. The architecture has been developed to accurately analyze our proposed system; we have coded our design using Verilog hardware description language and have synthesized it for FPGA products. Modelsim is used for simulations in addition to Xilinx ISE. The FPGA chips used are: Spartan 3E from Xilinx.

A. *Proposed Algorithm Hardware Implementation Challenges and Assumptions*

There are different approaches used to implement LIP. The challenges of implementing LIP are mainly related to the approach to managing the look-up table for ln(x), exp(x), for multipliers and for dividers. Some of the approaches for improving or reducing the size of the look-up table include using a pipeline in which we minimize the number of bits and assume some constant values for the parameters in the lookup tables; we picked the architecture to be able to share resources between different functions. Below are the design features implemented with associated assumptions:

- Two input vectors - each 8-bit wide - g1[7:0] and g2[7:0]
- Two single bit inputs (asopr, mult) to represent which operation we are executing
- A decoder is used to select if add, subtract or multiplication is employed
- We implemented and simulated the design assuming these parameters k(M)= 255, Lmpda(M) =256 and ἑ = 0
- Verilog to write the code
- Modelsim used for simulations
- DC and ISE Xilinx tools used for syntheses
- Interface with the associated instantiated blocks shown in figure 2.

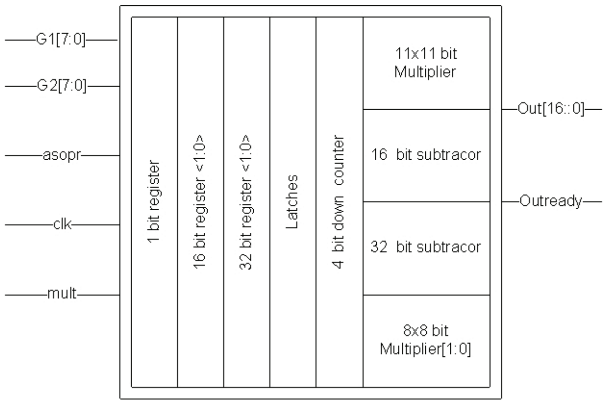


Figure 2. Interface block diagram

B. *Pipeline design*

We have improved the performance of our system by pipelining. The block diagram in Figure 3 shows pipeline architecture implementations of the design. As we can see from Figure 3, we need at least 5 cycles to do addition or subtractions and one more cycle if we are doing multiplications.

The number of cycles will change based on technology, process, number of bits for which you are going to implement the design, and parameters you chose. In cycle 1, we have implemented a decoder that takes two single bit inputs; the output will decide on whether the operation is *add, sub* or *mult.* The output of the decoder will be the control-enable signals (*sel_operation*) to all parts on cycle1; it will be latched and used in the decision-making and what to run in other pipes. In cycles 2 and 3: the circuit determines what logic to gate into, based on the control signal; if we are performing *mult* then we just access the ln(x) function on cycle 2, multiply on cycle 3 and get the inverse on cycle 5; otherwise we connect to the divider, then subtraction; then on cycle 5 the output is selected.

C. *Syntheses and circuit design*

We implemented the proposed architecture using Verilog and syntheses flow, this approach is appropriate because we can change parameters and re-synthesize quickly. In this paper our design primarily consists of six major modules. They are: adder, sub tractor, decoder, multiplier, divider, and multiplier.

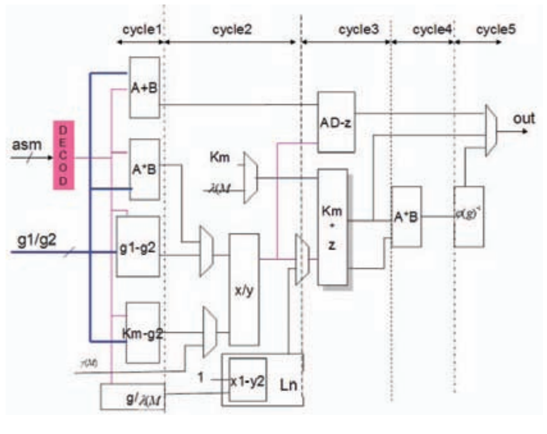


Figure 3. pipeline block diagram

Each part of the circuit is enabled based on the decode selection. Using this approach, the system can take advantages of using a combination of these functions to perform fast multiplication and division. A look-up table will not be needed to implement addition or subtraction; but it will be enabled when multiplications is selected.

There is a trade-off of memory consumption for multiplication (the look-up tables are larger). However, since the system needs to implement both number systems, it will take more space to put a conversion block to switch the number system back and forth and it will increase the time delay of the operations.

Two control signals, asopr and mult use these signals to choose what operation will be executed. Mathematical algorithms are implemented in Verilog to demonstrate the proposed arithmetic operations. Figure 4 shows simulation results of the multiplecation

D. *Sub-circuits/Modules used*

Table 1 shows component with the associated numbers. Three Multipliers are used: one 11x11 bits and two 8x8 bits multiplier. Multipliers are enabled based on the operations. Registers, latches and counters: We used look-up table to find the exp and ln(X) functions.

E. *Area, Power and timing consumptions*

In Table 2 a comparison is shown of the IC area consumed, based on the corresponding operation; most of the area corresponds to the combinational logic. For power, as shown in Tabel 3, the circuit consumes 2.4mW and the cell leakage power is 4.75uW using voltage of 0.85v.

The size of circuit *Cn* is the number of edges (connecting links) of the circuit. The depth of circuit *Cn* is the length of the longest path from any input node to an output node. A circuit family is an innate list of circuits *C = (C1; C2; :::; Cn; :::)* where *Cn* has n inputs. As described above, C computes a family of functions *(f1; f2; :::; fn; :::),* where *fn* is the function of n inputs computed by circuit.

The FPGA chip used is: Spartan 3E from Xilinix. The critical length in the circuit implemented on the Xilinix FPGA had the minimum period for the proposed subsystem is 10.209ns (Maximum Frequency 97.957MHz). Minimum input arrival time before clock is 16.178ns. Maximum output required time after clock is 6.140ns. Addition timing results for addetion are shown in Figure 5, when *asopr* =1, *mult*=1, g1=7, g2=3. It took 95ns to get the result of the add function.

Subtraction function timing results shows when *asopr* =1, *mult* =1, g1=7, g2=3 the out is 3. Once we switched from add to sub it took the sub-tractor (190 -499) 309ns to get the result. The multiplication function timing results when *asopr* =1, *mult* =1, g1=7, g2=3. The output resulted after we enabled the mult function - “*activate m*=1”.

TABLE 1 CIRCUIT TYPE

Function	Number of instant
16-bit sub tractor	2
32-bit sub tractor	1
11x11-bit multiplier	1
8x8-bit multiplier	2
1-bit register	41
16-bit register	2
32-bit register	2
1-bit latch	2
11-bit latch	2
16-bit latch	3
22-bit latch	1
8-bit latch	3
9-bit latch	1
4-bit down counter	1

TABLE 2 LOGIC FUNCTION USED

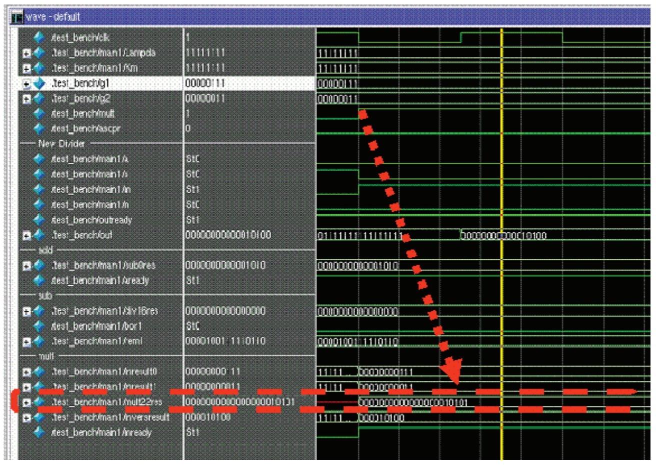
Logic Type	Area(um2)
Combinational	4010.965
Non combinational	608.366
Cell area	4619.33
Net(Interconnect)	25.59349
Total area	4644.924

TABLE 3 AREA

Power Tabel	
Global Operating Voltage	0.85v
Total Dynamic Power	2.3650 mW
Cell Leakage Power	4.7560 uW
Total Power	2.37mW

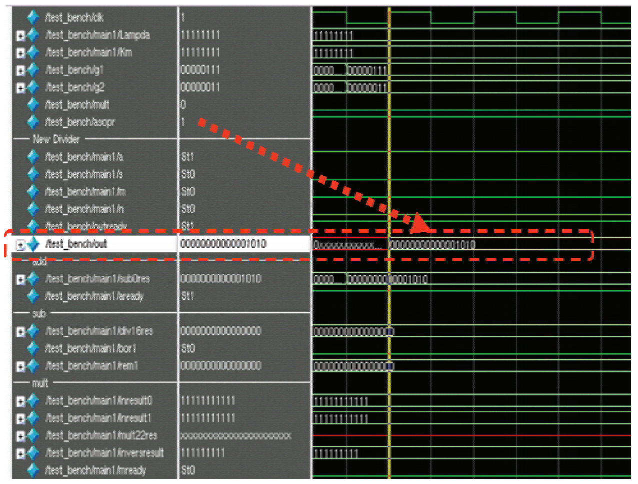
F. *Hardware limitations*

The major problem in implementing division is that the natural logarithm and the exponential are complicated to deal with, so a look-up table is used. As the precision goes up, a larger look up table is needed with a higher level of detail. It consumes memory and also has a long delay time caused by table searching.



```
wave - default
/test_bench/clk                    1
/test_bench/main1/Lampda           11111111
/test_bench/main1/Km               11111111
/test_bench/g1                     00000111
/test_bench/g2                     00000011
/test_bench/mult                   1
/test_bench/ascpr                  0
New Divider
/test_bench/main1/A                St0
/test_bench/main1/s                St0
/test_bench/main1/m                St1
/test_bench/main1/h                St0
/test_bench/outready               St1
/test_bench/out                    0000000000010100
add
/test_bench/main1/sub0res          0000000000001010
/test_bench/main1/aready           St1
sub
/test_bench/main1/div16res         0000000000000000
/test_bench/main1/bor1             St0
/test_bench/main1/rem1             00001001 11110110
mult
/test_bench/main1/nresult0         00000000 11
/test_bench/main1/nresult1         0000000011
/test_bench/main1/mult22res        0000000000000000010101
/test_bench/main1/inversresult     000010100
/test_bench/main1/mready           St1
```

Figure 4. multiplication simulation results



```
/test_bench/clk                    1
/test_bench/main1/Lampda           11111111
/test_bench/main1/Km               11111111
/test_bench/g1                     00000111
/test_bench/g2                     00000011
/test_bench/mult                   0
/test_bench/ascpr                  1
New Divider
/test_bench/main1/a                St1
/test_bench/main1/s                St0
/test_bench/main1/m                St0
/test_bench/main1/h                St0
/test_bench/outready               St1
/test_bench/out                    0000000000001010
add
/test_bench/main1/sub0res          0000000000001010
/test_bench/main1/aready           St1
sub
/test_bench/main1/div16res         0000000000000000
/test_bench/main1/bor1             St0
/test_bench/main1/rem1             0000000000000000
mult
/test_bench/main1/nresult0         11111111111
/test_bench/main1/nresult1         11111111111
/test_bench/main1/mult22res        xxxxxxxxxxxxxxxxxxxxxx
/test_bench/main1/inversresult     111111111
/test_bench/main1/mready           St0
```

Figure 1 Addition simulation result

V. Conclusions

In this paper, we presented an optimized implementation of the parameterized Digital Electronic Arithmetic (PDEA) approach for implementing computer arithmetic. This particular model has the advantage of being fine tuned for signal processing; in this case it uses the mean squared error measurement and objective measures of enhancement to achieve a more effective signal processing model. This implementation has the advantage of being optimized based on operation, power and area. To accurately analyze our proposed system, we have coded our design using Verilog hardware description language and have synthesized it for FPGA products using ISE and Modelsim.

References

[1] Eric Whartona, Dr. Karen Panettac, Dr. Sos Agaian, Digital electronic arithmetic with applications, IEEE Inter. Conf., 2007

[2] Design and performance of pixel-level pipelined-parallel architecture for high speed wavelet-based image compression, Computers and Electrical Engineering , 2005.

[3] G. Deng and L. W. Cahill Logarithmic number system and its application to image processing, Department of Electronic Engineering, La Trobe University, Bundoora Victoria 3083 Australia. EEE 1058-6393193 $03.00 0 1993

[4] http://oregonstate.edu/~ngw/LNS_files/frame.htm

[5] http://rsb.info.nih.gov/nih-image/more docs/Engineering/ImgEngr.html

[6] http://wwwisl.stanford.edu/~abbas/group/papers_and_pub/spie00_jeff.pdf

[7] UTSA – Class notes. EE 3523 Signal and System II

[8] T. G. Stockham Jr., “Image processing in the context of a visual model,” Proceedings of the IEEE, vol. 60, no. 7, pp. 828–842, 1972.

[9] D. J. Granrath, “The role of human visual models in image processing,” Proceedings of the IEEE, vol. 69, no. 5, pp. 552–561, 1981.

[10] Marr [D. Marr, Vision: A Computational Investigation into the Human Representation and Processing of Visual Information, W. H. Freeman, San Fransisco, Calif, USA, 1982.

[11] D. G. Myers, Digital Signal Processing Efficient Convolution and Fourier Transform Technique, Prentice-Hall, Upper SaddleRiver, NJ, USA, 1990.

[12] Q.-Z. Wu and B.-S. Jeng, “Background subtraction based on logarithmic intensities,” Pattern Recognition Letters, vol. 23,no. 13, pp. 1529–1536, 2002, N. Hauti`ere, R. Labayrade, and D. Aubert, “Detection of visibility conditions through use of on board cameras,” in Proceedings of IEEE Intelligent Vehicles Symposium (IVS ’05), pp. 193–198, Las Vegas, Nev, USA, June 2005.

[13] C. Piguet, "Ultra Low-Power Digital Design," in CMOS \& BiCMOS IC Design'97 , Lausanne, 1997.

[14] M. A. Bayoumi and E. E. Swartzlander, "VLSI Signal processing Technology," Kluwer Academic Publishers, 1994:

[15] M. Jourlin and J.-C. Pinoli, “Logarithmic image processing,” Acta Stereologica, vol. 6, pp. 651–656, 1987.