High Speed Floating Point Adder for DSP Applications

L. Pavan Kalyan[1] S. Madhava Rao[2]

[1]M.Tech student, [2]Associate Professor

[1,2]ECE Department, Mallineni Laksmaiah Engineering College, JNTUK, Prakasam(D), AP

Abstract:

A fast and energy-efficient floating point unit is always needed in electronics industry especially in DSP, image processing and as arithmetic unit in microprocessors. Many numerically intensive applications require rapid execution of arithmetic operations. Addition is the most frequent operation followed by multiplication. The demand for high performance, low Power floating point adder cores has been on the rise during the recent years. Since the hardware implementation of floating point addition involves the realization of a multitude of distinct data processing sub-units that endure a series of power consuming transitions during the course of their operations, the power consumption of floating point adders are, in general, quite significant in comparison to that of their integer counterparts. Speed is the key parameter for a specific application. The objective is to design a 32 bit single precision floating point adder/subtractor on the IEEE 754 standard floating point representations. The projected plan is to instantiate a good design and modify it for low power and high speed. The pipelining of these designs target high throughput computations.

Key words: Floating point unit, DSP, Addition, 32 bit single precision floating point adder, IEEE 754 standard.

1. Introduction

There are several ways to represent real numbers on computers. Floating-point representation, in particular the standard IEEE format, is by far the most common way of representing an approximation to real numbers in computers because it is efficiently handled in most large computer processors. Binary fixed point is usually used in special-purpose applications on embedded processors that can only do integer arithmetic, but decimal fixed point is common in commercial applications. Fixed point places a radix point somewhere in the middle of the digits, and is equivalent to using integers that represent portions of some unit. Fixed-point has a fixed window of representation, which limits it from representing very large or very small numbers. Also, fixed-point is prone to a loss of precision when two large numbers are divided. Floating-point solves a number of representation problems. Floating-point employs a sort of "sliding window" of precision appropriate to the scale of the number. This allows it to represent numbers from 1,000,000,000,000 to 0.0000000000000001 with ease. The advantage of floating-point representation over fixed-point (and integer) representation is that it can support a much

wider range of values. Floating-point representation the most common solution basically represents reals in scientific notation. Scientific notation represents numbers as a base number and an exponent. For example, 123.456 could be represented as $1.23456 \times 10^{2.}$ Floating-point numbers are typically packed into a computer datum as the sign bit, the exponent field, and the significand (mantissa), from left to right. In computing, floating point describes a system for representing numbers that would be too large or too small to be represented as integers. Numbers are in general represented approximately to a fixed number of significant digits and scaled using an exponent. The base for the scaling is normally 2, 10 or 16. The typical number that can be represented exactly is of the form:

$$\text{Significant digits} \times \text{base}^{\text{exponent}}$$

The term floating point refers to the fact that the radix point (decimal point, or, more commonly in computers, binary point) can "float"; that is, it can be placed anywhere relative to the significant digits of the number. This position is indicated separately in the internal representation, and floating-point representation can thus be thought of as a computer realization of scientific notation.

2. Floating Point Adder

Floating point addition is the most frequent floating point operation. FP adders are critically important components in modern microprocessors and Digital Signal Processors. The design of Floating Point Adders is considered more difficult than most other arithmetic units due to the relatively large number of sequentially dependent operations required for a single addition and the extra circuits to deal with special cases such as infinity arithmetic, zeros, NaNs.
Standard Floating Point Addition requires following steps:

- Preshift for mantissa alignment
- Mantissa addition/subtraction
- Post shift for result normalization
- Rounding

Addition of floating point numbers involves the prealignment, addition, normalization and rounding of significands as well as exponent evaluation. Significand prealignment is a pre-requisite for addition. In floating point additions, the exponent of the larger number is chosen as the tentative exponent of the result. Exponent equalization of the smaller floating point number to that of the larger number demands the shifting of the significand of the smaller number through an appropriate number of bit positions.

The absolute value of the difference between the exponents of the numbers decides the

magnitude of alignment shift. Addition of significands is essentially a signed magnitude addition, the result of which operation is also represented in signed-magnitude form. Signed-magnitude addition of significands can lead to the generation of a carry out from the MSB position of the significand or the generation of leading zeros or even a zero result. Normalization shifts are essential to restore the result of the signed-magnitude significand addition into standard form. Rounding of normalized significands is the last step in the whole addition process. Rounding demands a conditional incrementing of the normalized significand. The operation of rounding, by itself can lead to the generation of a cany out from the MSB position of the normalized significand. That means, the rounded significand need be subjected to a correction shifting in certain situations.

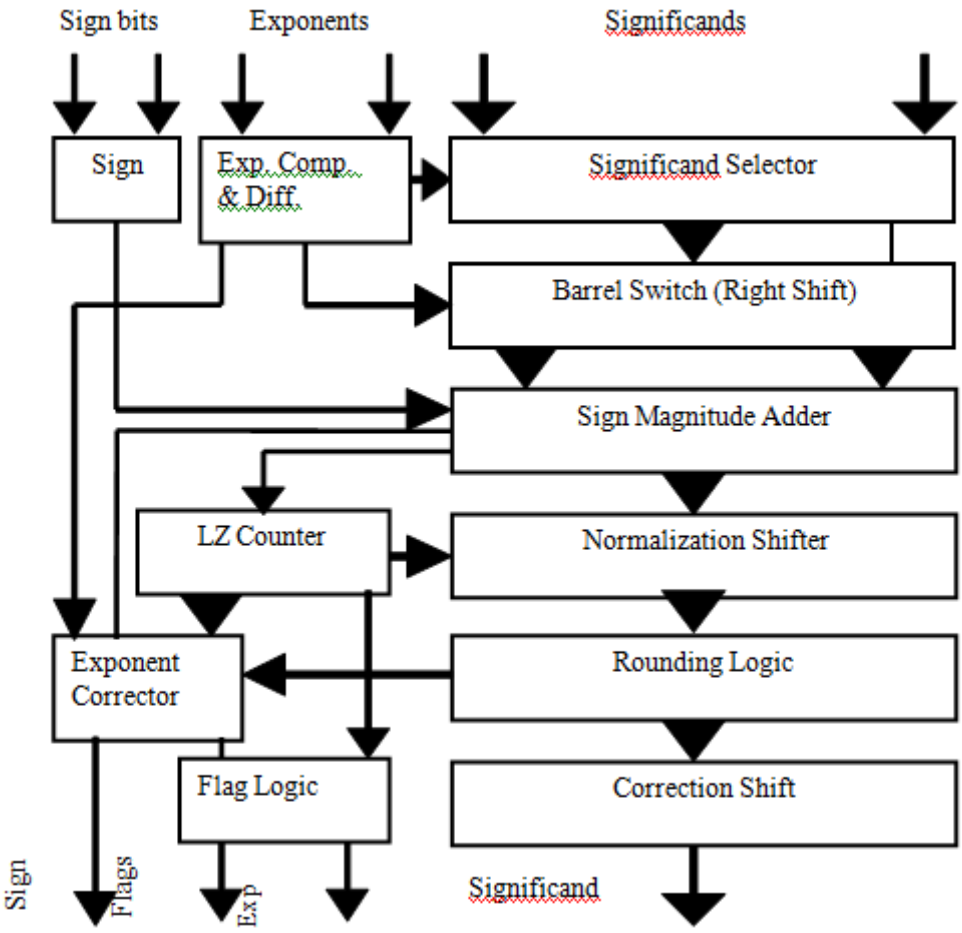


Figure 2.1: Block diagram of Floating Point Adder

Figure 2.1 illustrates the block diagram of a floating point adder. The exponent comparator and differencer block evaluates the relative magnitudes of the exponents as well as the difference between them. The significand selector block routes the significand of the larger number to the adder while routing the significand of the smaller number to the Barrel Switch. The Barrel Switch performs the requisite pre-alignment shift (right shift). The sign magnitude adder performs significand additiodsubtraction, the result of which is again represented in sign-magnitude form, the leading zero counter evaluates the number of leading zeros and encodes it into a binary number. This block, essentially controls normalization shifts. The rounding logic evaluates rounding conditions and performs significand correction whenever rounding is necessary. The correction shift logic performs a right shift of the rounded significand if the process of rounding has resulted in the generation of a carry out from the MSB position of the significand. The exponent corrector block evaluates the value of the exponent of the result. The flag logic asserts various status flags subject to the validity of various exception conditions.

Normalization

Normalisation stage left shifts the significand to obtain leading '1' in the MSB, and adjusts the exponent by the normalisation distance. For true subtraction, the position of the leading '1' of the significand result is predicted from the input significands to within 1-bit using an LZA [1,2] which is calculated concurrently with the significand addition. The normalisation distance is passed to a normalization barrel shifter and subtracted from the exponent.

Example:

-100100101001.0012 (binary)

= -1.00100101001001 x 2^11 (normalized binary)

3. Proposed Pipelined Triple Paths Floating Point Adder

A new architecture of speed and power optimized floating point adder is presented.. The functional partitioning of the adder into three distinct, inhibit controlled data paths allows activity reduction. During any given operation cycle, only one of the data paths is active during which time, the logic assertion status of the circuit nodes of the other data paths are held at their previous states. Critical path delay and latency are reduced by incorporating speculative rounding and leading one anticipation (prediction) logic. The latency of the floating point addition can be improved if the rounding is combined with addition/subtraction. LOP anticipates the amount of shift for normalization in parallel with addition [5].

Pipelining

Pipelining is one of the popular methods to realize high performance computing platform. It was first introduced by IBM in 1954 in Project Stretch. In a pipelined unit a new task is started before the previous is complete. This is possible because most of the adders, high speed multiplies are combinational circuits. Once a gate output has been set it remains fixed for the remainder of the operation. Pipelining is a technique to provide further increase in bandwidth by allowing simultaneous execution of many tasks.

A pipelined system divides the logic into stages separated by a set of latch registers. As soon as the output of a stage is latched, a new task can be started at the inputs to the stage. The latches keep the value of the old task which is input to the next stage. The bandwidth of the pipelined unit is the inverse of the latency per stage, not the latency of entire unit. As stage length is shortened bandwidth increases but as the number of stages increases, so does the number of latches necessary. While pipelining can result in sizeable increases in bandwidth, so can operating many units in parallel. Since a pipelined system processes many tasks at one time, the overall bandwidth / cost function should increase. In a pipelined system, the total latency is not as important as the latency per stage. It is important to reduce the total no of latches required since latches become a significant portion of the circuitry in a highly pipelined system.

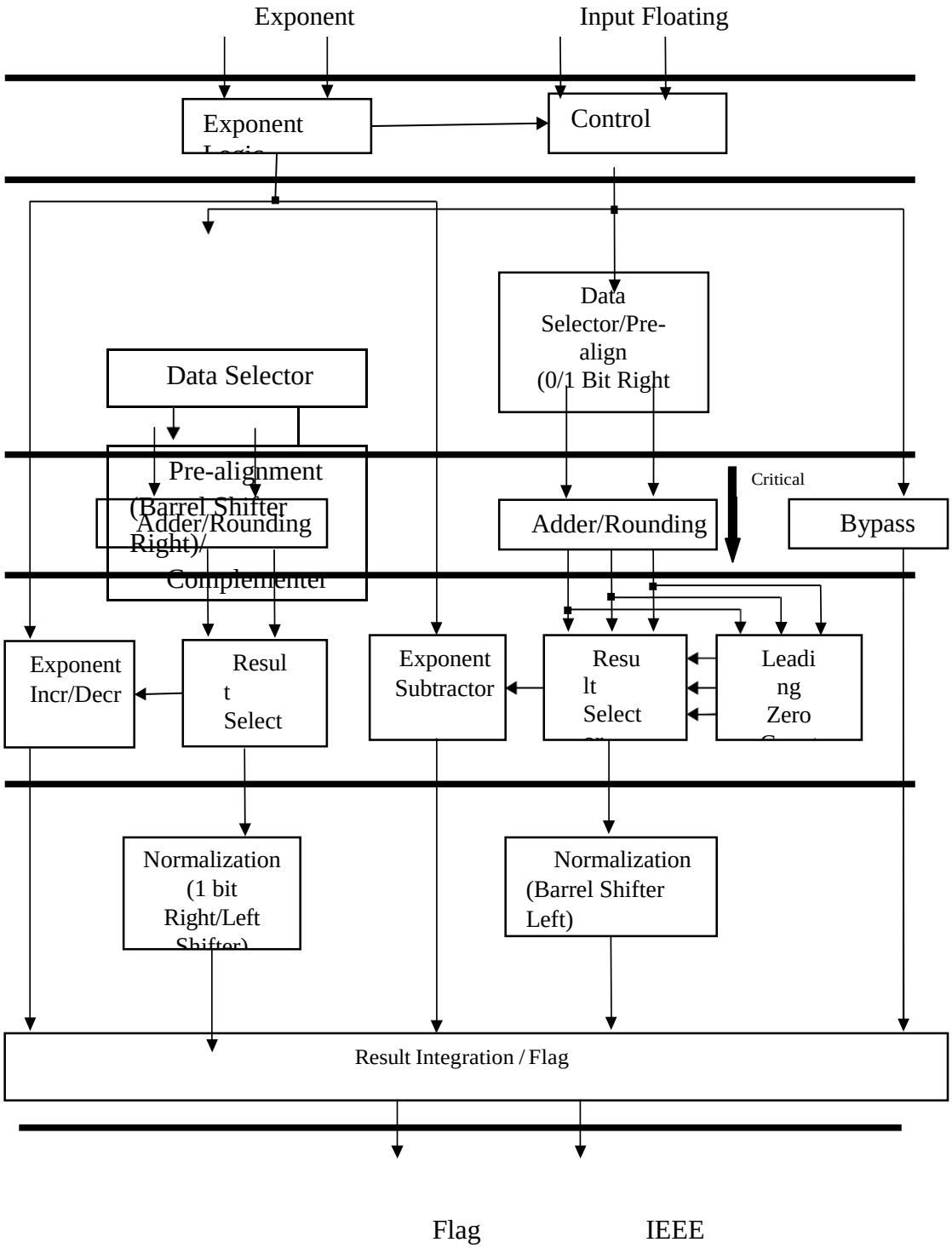


Figure 2.6: Proposed Pipelined Triple Paths Floating Point Adder TDFADD

First in pipelined affects, the latches usually provide the necessary buffering. Second, the maximum fan-out of gates runs through wide range dependent on the manufactures. Implementing pipelining requires various phases of floating-point operations be separated

and be pipelined into sequential stages. Pipelining is a technique where multiple instruction executions are overlapped. Design functionalities are validated through simulation and compilation.

To achieve pipelining input processes must be subdivided into a sequence of subtasks, each of which can be executed by specialized hardware stage that operates concurrently with other stages in the pipeline. The adder design pipelines the steps, complementing, swapping, shifting, addition and normalization to achieve a summation every clock cycle.Each pipeline stage performs operations independent of others. Input data to the added continuously streams in. Speed of operation of pipelined add has been found 3 times more than the unpipelined adder. Successful implementation of pipelining in floating point adder and multiplier using VHDL fulfills the needs for different high performance applications.

4. Implementation on FPGA

A field programmable gate array (FPGA) is a semiconductor device that can be configured by the customer or the designer after manufacturing hence the name "field- programmable". Field Programmable gate arrays (FPGAs) are truly revolutionary devices that blend the benefits of both hardware and software. FPGAs are programmed using a logic circuit diagram or a source code in Hardware Description Language (HDL) to specify how the chip will work. They can be used to implement any logical function that an Application Specific Integrated Circuit (ASIC) could perform but the ability to update the functionality after shipping offers advantages for many applications. FPGAs contain programmable logic components called "logic blocks", and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together" somewhat like a one chip programmable breadboard. Logic blocks can be configured to perform complex combinational functions or merely simple logic gates like AND and XOR. In most FPGAs, the logic block also includes memory elements, which may be simple flip flops or more complete blocks of memory.

FPGAs blend the benefits of both hardware and software. They implement circuits just like hardware performing huge power, area and performance benefits over softwares, yet can be reprogrammed cheaply and easily to implement a wide range of tasks. Just like computer hardware, FPGAs implement computations spatially, simultaneously computing millions of operations in resources distributed across a silicon chip. Such systems can be hundreds of times faster than microprocessor-based designs. However unlike in ASICs, these computations are programmed into a chip, not permanently frozen by the manufacturing process. This means that an FPGA based system can be programmed and reprogrammed many times.

FPGAs are being incorporated as central processing elements in many applications such as consumer electronics, automotive, image/video processing military/aerospace, base stations, networking/communications, supercomputing and wireless applications.

FPGA for Floating Point Computations

With gate counts approaching ten million gates, FPGA's are quickly becoming suitable for major floating point computations. However, to date, few comprehensive tools that allow for floating point unit tradeoffs have been developed. Most commercial and academic floating point libraries provide only a small number of floating point modules with fixed parameters of bit-width, area and speed [7]. Due to these limitations, user designs must be modified to accommodate the available units. The balance between FPGA floating point unit resources and performance is influenced by subtle context and design requirements. Generally, implementation requirements are characterized by throughput, latency and area.

1 FPGAs are often used in place of software to take advantage of inherent parallelism and specialization. For data intensive applications, data throughput is critical.
2 If floating point computation is in a dependent loop, computation latency could be an overall performance bottleneck.

FPGA Technology Trends

- General trend is bigger and faster.
- This is being achieved by increases in device density through even smaller fabrication process technology.
- New generations of FPGAs are geared towards implementing entire systems on a single device.
- Features such as RAM, dedicated arithmetic hardware, clock management and transceivers are available in addition to the main programmable logic.
- FPGAs are also available with the embedded processors (embedded in silicon or as cores within the programmable logic fabric).

FPGA Implementation

The FPGA that is used for the implementation of the circuit is the Xilinx Spartan 3E (Family), XC3S5000 (Device), FG320 (Package) FPGA device. The working environment/tool for the design is the Xilinx ISE 9.2i is used for FPGA Design flow of VHDL code.

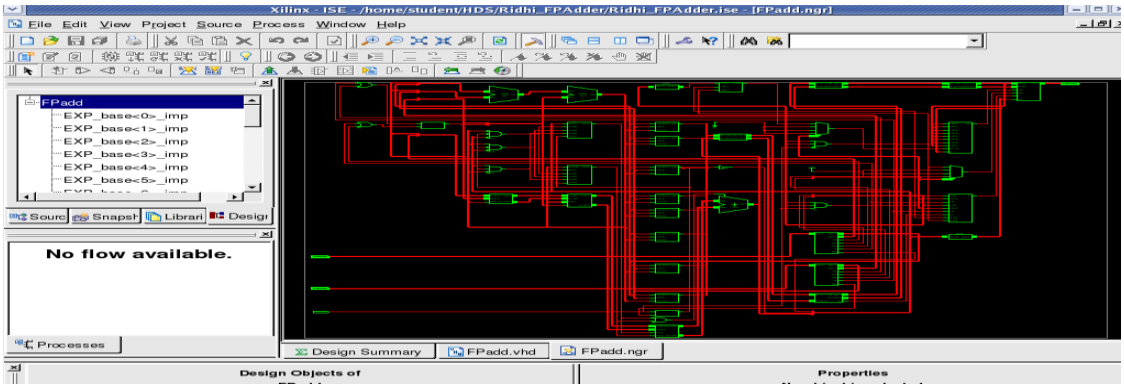
5. Results

Simulation and Discussion

The use of VHDL for modeling is especially appealing since it provides formal description of the system and allows the use of specific description styles to cover the different abstraction levels (architectural, register transfer and logic level) employed in the design. Design is verified through simulation, which is done in a bottom-up fashion. Small modules are simulated in separate test benches before they are integrated and tested as a whole.

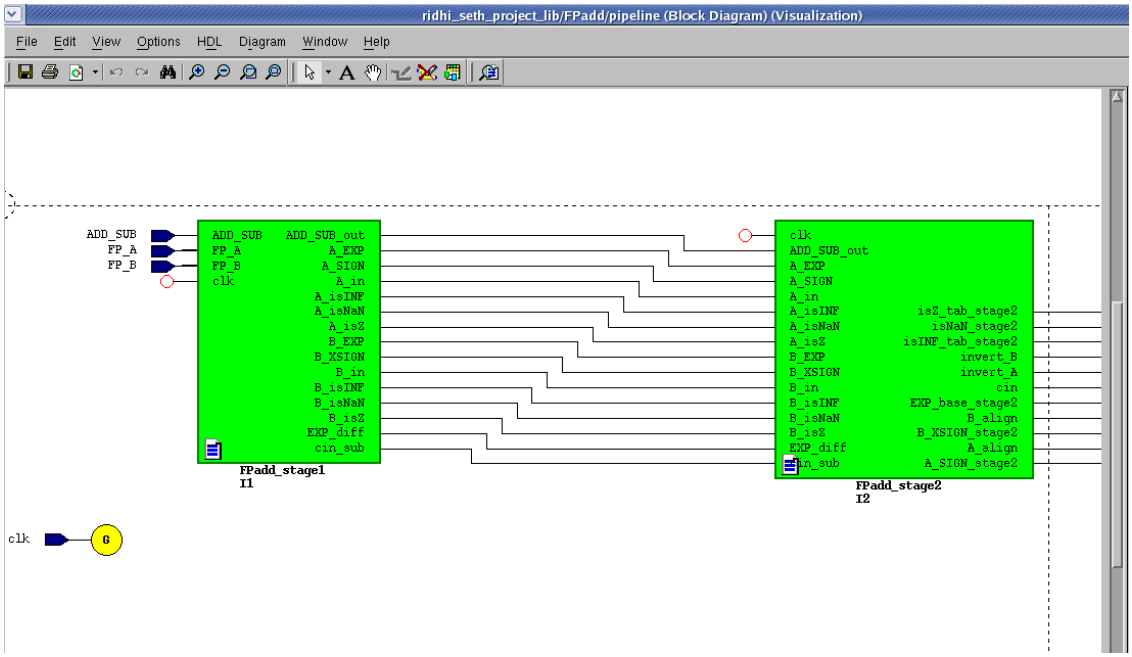
Xilinx Results of propose Floating Point Adder:

RTL of proposed FP adder



Modalism Results of proposed FP Adder:

Block Diagram Visualization of proposed Pipelining in FP adder



Design summary of standard and proposed FP adders:

S.No	Parameters	Standard Floating Point Adder	Proposed Pipelined Triple Paths Floating Point Adder
1.	Number of IOs	105 out of 182(57%)	105 out of 182(57%)
2.	Number of BELs	205 out of 1728(11.8%)	236 out of 1728(13.6%)
3.	Minimum period	2.649 ns	1.937 ns (26.87%)
4.	Max.Frequency(speed)	377.501 MHz	516.262 MHz (36.75%)
5.	Power consumption	38.46mw	41.79mw (8.65%)

6. Conclusion

This work presents design, optimization and implementation of floating point arithmetic modules on FPGA to be used in DSP applications. These modules are floating point adder/subtractor. The architectural design of Pipelined Triple Data Path floating Point Adder incorporating techniques of throughput acceleration is addressed. The latency of the floating point addition can be improved by combining rounding with addition/subtraction. Pipelining of the design targets have high throughput computations. The simulation and synthesis results of modules are provided.

References

[1] IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 7541985.

[2] N. Burgess. The flagged prefix adder for dual additions. In Proc. SPIE ASPAAI-7, volume 3461, pages 567–575, San Diego, Jul. 1998.

[3] M. Farmwald. On the Design of High Performance Digital Arithmetic Units. PhD thesis, Stanford University, Aug. 1981.

[4] R.V.K. PILLAI Concordia University, Montreal, Canada D. AL-KHALILI Royal Military College, Kingston, Canada A.J. AL-KHALILI AND S.Y.A. SHAH Concordia University, Montreal, Canada "A Low Power Approach to Floating Point Adder Design for DSP Applications "Received August 12, 1999; Revised April 20, 2000 pages 195-213.

[5] N. Quach and M. Flynn. Design and implementation of the snap floating-point adder. Technical Report CSL-TR- 91-501, Stanford University, Dec. 1991.

[6] Y. Hagihara, S. Inui, F. Okamoto, M. Nishida, T. Nakamura, and H. Yamada. Floating- point datapaths with online builtin self speed test. IEEE J. Solid-State Circuits, 32(3):444– 449, Mar. 1997.

[7] R.V.K. Pillai, D. Al-Khalili and A.J. Al-Khalili "Power Implications of Additions in Floating Point DSP -an Architectural Perspective" Concordia University, Montreal, CANADA; Royal Military College, Kingston, The IEEE International Conference 1999 Pages: 581- 586.