On Low Power Floating Point Data Path Architectures

Rajan V. K. Pillai

A Thesis
in
The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy at
Concordia University
Montreal, Quebec, Canada

April 1999

Abstract

On Low Power Floating Point Data Path Architectures

Rajan V. K. Pillai
Concordia University, 1999

This work targets development of higher level design methodologies for the implementation of low power floating point units - adders, multipliers and multiply - accumulators. Philosophically, higher level design starts with the characterization of the behavior of the system under consideration. In this work, the design philosophy of floating point adders and multiply accumulators is centered around the significand alignment driven power behavior of these units. Transition activity scaling of these units, taking into account the behavior of exponents, provides a viable high level approach for the design of these units. The exponent behavior based design partition also allows simplification of data paths, so that the speed performance of these units are enhanced. With transition activity scaling, the switching activity of the floating point units is mapped to a limited subset of the hardware real estate, so that the time averaged power consumption of these units is optimal. In order to validate the design methodology, high level power/delay models that reflect the architecture of the functional units are developed. The behavior of the architectures is validated through simulations. Instrumented digital filter programs that emulate the multiply - accumulate segment of DSP systems form the core of our experimental platform.

Acknowledgments

I am extremely indebted to my Professors A.J. Al - Khalili and D. Al - Khalili for their supervision and support. Professor A. J Al - Khalili introduced me to the exciting discipline of VLSI research. It is indeed a hard task to condense my acknowledgments towards my Professors in a few words. Without their support and patience, this dissertation might have not taken the present shape. During the course work phase, I had been extremely fortunate to do courses with Professor Baher Haroun. Professor Haroun's courses had been of immense help in understanding the state of the art in VLSI DSP design. I would like to extend my sincere thanks to David Hargreaves and Girish Patel for their support. During my association with ECE Department, I have enjoyed the good friendship of my fellow graduate students; Ganesh Babu, Shereef Shehata, Khalid Ammar, Ramesh G. R. and others.

Besides my teachers and friends at graduate school, there are a few from my Karma Kanda whom I would like to thank. My Guru - H.H. Swamy Theraviyam, my brother, mother, wife and children, and friends had been of immense help during my Aranaya Kanda.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Introduction

Floating point units (FPUs) are essential building blocks of modern arithmetic/logic units (ALUs) and embedded systems. Since the power consumption and throughput of these units, to a large extent, depend on the quality of floating point units, it is imperative that design efforts be directed towards the optimal implementation of these units. With low power CMOS VLSI design, the magnitude of power reduction/performance enhancement attainable through higher level approaches are quite significant. One of the fundamental driving forces behind the exploration for higher level design optimization strategies is the reusability of architectures/algorithms and methodologies. While the results of lower level approaches like circuit techniques, device scalings etc. perish with technology migration, the higher level strategies survive through a few generations of technology migration. However, development of higher level design methodologies can seldom be considered an isolated activity, rather the development of higher level approaches taking into account the technological idiosyncrasies/limitations/volatilities is envisioned. This work addresses the development of higher level design decisions for the realization of low power floating point adders, accumulators and multiply - accumulators (MAC). Owing to the lack of a full control over all possible choices of low power design strategies, the design efforts are concentrated towards achieving power economy by choosing appropriate architectures, number systems and algorithms.

1.2. Motivation

Numerical computation involving floating point operands is fundamental in many scientific applications. In the past, floating point computations were mainly performed by higher end computing platforms. With advances in VLSI technology, this situation, however, has changed. Even small (and relatively low cost) modern day computers can perform sophisticated floating point computations. Most of these platforms work with dedicated floating point math processors in conjunction with general purpose CPUs. Even this situation is rapidly changing. The CPUs themselves are equipped with floating point units these days. The evolution of dedicated digital signal processing devices (DSPs) is another offshoot of VLSI technology during the past. While integer/fixed point DSPs dominate the market for many real time signal processing applications, the demand for sophisticated floating point DSPs had been increasing during the recent years, particularly for the military applications [1].

With time, while the CPUs/floating point processors/DSPs had been boasting new performance heights, the insight gathered during this evolution paved the way to a new arena of floating point computation, dedicated system solutions. With the conventional CPU - math processor/DSP setup, there are obvious technological bottlenecks that make it hard to achieve computational speeds which many applications demand. In bus oriented systems, the performance of the bus is an obvious limitation. With super scalar CPU architectures and parallel CPU configurations, the computational throughput of functional units can be accelerated, but the performance limit of the system is always dictated by the slower bus speeds. The power consumption of bus interface devices is also of serious concern for modern day designs. The larger the number of bus lines and the higher the bus speed, the

higher the power demand. With instruction driven architectures, power consumption occurs due to instruction as well as data traffic. The on chip power dissipation of control units, caches, data RAMs, DMA controllers etc. adds another dimension to this problem. While the instruction driven, bus oriented architectures are seldom satisfactory for many high performance (and/or low power) applications, dedicated system solutions are the ultimate choice. When general purpose floating point processors/DSPs are claiming performance heights of a few hundred MFLOPs, dedicated systems have already reached GFLOPS capabilities [1].

Table 1.1: Expected power gains in logic synthesis for various levels of abstraction

Level	Transformation	Expected Power Gain
Algorithmic	Algorithm selection	Orders of magnitude
Behavioral	Concurrency	Several times
Register Transfer Level	Structural transformations	10 - 15%
	Clock control	10 - 90%
Data/signal encoding		20%
Technology independent	Extraction/decomposition	15%
Technology dependant	Technology mapping	20%
	Gate sizing	20%
Layout	Placement	20%

In dedicated system design, the power - performance - area triad dictates the viability of a product. With these systems, the designers define appropriate data formats, hardware/software partitions and/or even analog/digital or mixed signal design strategies such that the final solution meets the required performance and power goals. Since the system solution

is more or less application specific, the economies of scale and production volumes may not support exhaustive full custom design approaches. Also, the design turnaround time should be minimal in such situations. This scenario demands a high degree of design automation, that takes into account the higher level power/performance/area optimalities and limitations of the system under consideration. None of the modern day logic synthesis tools are tailor made for low power synthesis. The absence of such tools necessitates intensive low power design efforts to develop appropriate architectures, number systems, data formats, control units and power management strategies.

The power consumption of VLSI ICs can be reduced through a variety of design approaches. The level of freedom enjoyed by full custom designers is the maximum, as far as low power design approaches are concerned. The higher the level of design abstraction, the less the degrees of freedom. Kurt Keutzer et al. [2] discuss the possibilities for power reduction in a hierarchical design environment. Table 1.1 lists the details.

Among all possibilities, the right algorithm gives the maximum power savings which can be orders of magnitude in comparison with other feasible alternatives. Instruction driven architectures versus dedicated system approaches can be considered as a typical example for this category of design decisions. Though instruction driven architectures offer flexibilities in terms of programmability, re-configurability etc., this approach warrants power demand for the supporting hardware units like control units, instruction decoders, bus interface units, cache memory, on chip RAM/ROM memories etc. If the right algorithm for such an application can be translated directly to dedicated hardware units, the power consumption of the overall system can be reduced by orders of magnitude. In behavioral

level power optimization, the possibilities for savings is again several hundred percent in comparison with other approaches. RTL level approaches can yield a good 10 to 90% savings while data/signal encodings can give another 20%.

In higher level synthesis, a good amount of power saving can be achieved by choosing the right system level design approaches. For example, a proper design of clocking scheme itself (e.g. multi phase vs. single phase clocking) can save a substantial amount of power. Activities like extraction/decomposition, technology mapping, gate sizing, placement etc. are best handled by the synthesis tools and hence developments in this direction is gathering momentum. As far as synthesis of low power arithmetic units (e.g. floating point unit) is concerned, power optimization can be achieved through algorithm selection, operational concurrence and structural transformations. Operational concurrence, in general, can result in performance acceleration. The extra speed realized through this technique can be traded for power.

The behavioral and RTL level transformations listed in Table 1.1 can be applied for realizing power efficient arithmetic units. The best example of concurrency for performance enhancement/power reduction of floating point units is the multiply accumulate - fused architecture reported by E. Hokenek et al. [3]. This approach gives significant speed enhancements, which can be traded for power, if desired. Structural and clocking strategies can also yield good power savings. Though not explicitly stated in Table 1.1, efficient power management strategies also result in power efficient logic implementations. The other possibility is activity reduction through logic minimizations, architectural modifications, low power encodings etc.

To summarize, with floating point hardware design, the development of higher level design decisions - architectures, algorithms, number systems etc. - for power/performance optimization is rewarding. Architectural/system level design decisions that are insensitive to technology migration provide results that doesn't perish in the short time.

1.3. Floating point hardware

Numerical computation of floating point operands requires hardware resources for handling the exponent and significand parts of operands. Over the years, the architecture of floating point units hasn't changed much, though the data formats went through some evolution. The IEEE standards [4] [5] [6] [7] define a set of data formats for floating point computation. In general, a given floating point format uniquely offers certain dynamic range and numerical resolution. Floating point representation of numbers involves some degree of approximation. Theoretically, it is not possible to represent an infinite continuum of real data into a precisely equivalent set of floating point values. The IEEE standards also specify guidelines for effecting numerical approximations (rounding) on the results of floating point computation.

The IEEE standards specify a limited set of floating point formats, viz., single precision, single extended, double precision and double extended. In real life, given a set of specifications for establishing certain computational task (for example, a typical signal processing application), the data formats need not be any of that specified by the IEEE standards. In such scenarios, even IEEE specified rounding schemes need not make sense. That means, depending on the specifications of certain application, system solutions can work with non IEEE data formats as well as rounding schemes such that the real life stimulus/

response signals satisfy the data processing goals demanded by the target application.

Floating point hardware for general purpose computing had been around for many years. The early floating point hardware units were built as part of higher end computers. The number systems followed by many of them were distinct in different vendor implementations. Some of the shortcomings of hardware units were rectified by suitable compiler designs, appropriate algorithms etc. With the introduction of IEEE standards, more and more manufacturers are following these standards for the design of their hardware units. Most of the floating point implementations reported during the last ten years deal with IEEE compatible units. The architectural concepts of floating point units are explained in [8] [9]. Since the concentration of this work is towards the development of scalable floating point multipliers/adders/multiply - accumulators, we will consider floating point units of this category only.

The computational algorithms for the realization of floating point operations didn't witness much of a change during the years. Traditionally, floating point units had been implemented as a combination of distinct integer units (binary adders, multipliers, comparators, barrel shifters etc.), and this approach is still very popular in industry. One reason for this bias is the availability of a rich array of knowledge regarding the implementation of integer/fixed point arithmetic units. The other reason is the availability of optimized integer building blocks in the form of standard cells, macro building blocks etc. Notable works in this field include the carry acceleration techniques [10] [11] [12] [13] [14] [15] and partial product generation [16] [17] [18] and reduction techniques [19] [20] [21].

During the recent years, an enormous number of publications on the hardware implemen-

tation of floating point units have appeared. Most of the work done in the past on floating point units had been related to full custom designs. With the shrinking of device feature size, process enhancements and innovative circuit styles (full CMOS, pass transistor logic styles etc.), the design of floating point units have changed a lot during these years, though the basic architectures remain more or less the same. Because of this, many of the works reported in literature during the recent years can be categorized as redesign approaches. References [3] and [22] to [44] list some significant work in this area during the past few years.

E. Hokenek et al. [3] [22] [23] discuss the implementation of a multiply - accumulate fused floating point unit. The throughput improvement offered by this scheme is quite attractive. The authors also present the development of a new leading zero anticipation scheme (LZA) [22] which facilitates concurrent evaluation of significand addition and computation of normalization shift requirements (instead of slower sequential leading zero counting) so that the latency of the FPU is reduced. The leading zero anticipatory logic proposed by the authors makes use of some of the signals of conventional carry look ahead adders (the carry generate, propagate and kill signals) and processes these signals concurrently. Recently Suzuki et al. [24] reported a new LZA scheme that outperforms the scheme reported in [22]. The work of Oberman et al. [25] [26] [27] reflects a new approach, multiple data path based variable latency architecture, for floating point addition.

Davies, T. C. et al. [28] [29] present the design of a processing element for systolic array applications. These processing elements can be configured as a multiplier - accumulator or

inner product step processor. The device has integrated scan path and built in self test features.

Young Surk Lee [30] (The original article is published in Korean language, but the abstract form is available in English) reported the design of an IEEE compatible floating point ALU. Instead of the traditional 2's complement arithmetic, this design uses 1's complement numbers for addition/subtraction. The same adder is reused for rounding as well.

Fujii et al. [31] reported the development of a floating point cell library. The library contains cells of floating point ALU, floating point multiplier, instruction RAM and register file. These cells are good for the design of instruction driven processors. Reference [32] presents the implementation of a CMOS pipelined ALU for superscalar processors while [33] discusses the design of a floating point unit that forms part of a RISC processor. Gillam, K.R et al. [34] discuss the architecture and implementation of an IEEE compatible adder that can be programmed to perform the addition of single or double precision numbers while [35] presents the design and implementation of a multi - mode pipelined multiplier. This multiplier can preform multiplication of 32 bit 2's complement integer numbers or IEEE single or double precision numbers. This multiplier uses octal Booth encoding for the generation of partial products.

Chai, P. et al. [36] reported the design of a 60 MHz CMOS floating point unit with a peak processing speed of 120 MFLOPs with double precision operands. This FPU can perform addition, multiplication, division, square root and inverse operations. Reference [37] presents the design of a superscalar RISC floating point processor with Harvard style bus organization, which can support two concurrent instructions during every clock cycle.

Hsu. P. Y. T. [38] reported the design of a superscalar (two floating point units inside one chip), Tremendous Floating Point (TFP), microprocessor. John A Kowaleski Jr. et al. [39] reported the development of an 866 MFLOPs dual execution pipelined CMOS processor which supports IEEE and VAX data types and rounding modes. This processor can perform two floating point operations per cycle. This processor also uses octal Booth encoding for the generation of partial products. Reference [40] describes the design of a self timed floating point unit using FPGAs and semi - custom ICs. The design supports completely self timed operation without clocking. The design has eliminated the need for a global control unit. Local control units perform all the timing and control functions taking into account the local constraints.

Reference [41] presents the design of a floating point multiplier that performs floating point scaling on integer R, G, B pixel data in computer graphics applications. Narasimhan, D. et al. [42] reported the implementation of an 8 stage pipelined FPGA based floating point adder which adds 13 bit floating point operands. Reference [43] discloses the design of a high speed, sub - nanosecond, CMOS adder suitable for handling the addition of double precision floating point numbers. This adder follows a different architecture for fast carry evaluations, the concept of Ling's adder [44] is followed. References [45] to [49] discuss the implementation of integer multipliers for handling the significand part of floating point operands while [50] and [51] discuss the pipelining issues of arithmetic units.

1.4. Low power design

Most of the work reported during the past few years in the area of low power arithmetic circuits are related to full custom designs which discuss new circuit topologies and occa-

sionally give some new architectural approaches. A few approaches on low power logic synthesis are also reported, but most of them deal with logic minimization problems, technology mapping, activity/timing driven input re-ordering clock system design etc. References [52] to [74] represent some interesting work as far as design optimization for low power logic synthesis is concerned. Power optimization and architectural modifications, in general, requires some knowledge regarding the power implications of logic circuits, architectures, number systems etc. and hence power estimation and model validations are essential. Reference [74] presents some approaches for power estimation.

Design for transition activity reduction is discussed in [53] and [54]. Reference [53] discusses the general issues of logic synthesis and technology mapping for low power and gives emphasis to reduction in spurious transitions (through delay balancing) and gate sizings (i.e. burying high activity nodes into low capacitance gate outputs). Transistor sizing and input ordering (taking into account the arrival times of signals at the input of a logic block such that the total power dissipation due to spurious transitions are minimized) for low power designs reported in [55] can be thought of as a circuit level approach, though some of these techniques can be applied to the design of CAD tools as well. Reference [58] discusses the applicability of power estimation techniques for low power synthesis while [59] presents an activity driven Boolean function minimization for low power designs. References [62] and [63] discuss about low power coding strategies for I/O interfaces. E. Olson et al. [64] reported the development of a low power state encoding strategy for finite state machines. This technique aims activity reduction in the 'present state' signal lines so that highly probable state transitions result in the switching of a minimum number of signal lines. With such an approach, the effective capacitance being switched

during highly likely state transitions is minimized. Reference [67] presents a discussion of low power design possibilities in high level synthesis. M. Alidina et al. [70] discuss the possibilities for precomputation to reduce power dissipation in logic circuits. Certain pre-computed signals can be used for enabling/disabling circuits down in the pipeline so that these circuits need be operational only if it is essential. For example, consider the case of a multiplier. If one of the operands is zero, the output can be asserted zero, irrespective of what happens in the multiplier array. This approach sounds good, but there are some hidden issues. The delay of these logic units can be higher than that of conventional units. References [72] and [73] presents schemes for race/hazard reduction and power management respectively for RTL designs.

1.5. Summary of design approaches

From the preceding paragraphs it is clear that many of the traditional approaches for low power CMOS design are not applicable for higher level design. In general, architectural design for low power/high performance targets the minimization of capacitance switching (time averaged). Philosophically, minimization of switched capacitance or fanout weighted switching activity can be achieved through the minimization of either of the factors - capacitance or activity factor. Also, design decisions that increase the area and hence the total capacitance are not judged as bad choices as long as there is a reduction in switched capacitance (per unit of time). Minimization of switching activity can be interpreted as a strategy that increases the correlation of logic variables. The higher the temporal correlation of a logic variable, the less the switching activity and vice versa. Correlation enhancement can be achieved by partitioning the problem into certain sub - problems, so that the signal correlation of the sub - units are better than that of the original

scheme. Partitioning of the problem in such a way that they can be handled by independent logic blocks, the operations of which constitute a mutually exclusive set is fundamental in the development of transition activity scaled architectures. With transition activity scaling, only one set of logic sub-units are active during any given operation cycle, during which time, the assertion status of the nodes of the other units are preserved at their previous states. Such an approach guarantees activity reduction. In essence, data path replications in conjunction with clock gating offers viable schemes for transition activity scaling.

In order to give a quick summary of how effective transition activity scaling is for the design of low power floating point units, a bird's eye view of certain salient results are presented. With our transition activity scaled triple data path floating point adder scheme (TDPFADD), the time averaged power consumption of the proposed scheme (worst case) is around 65% of that of other schemes. These figures had been arrived at through architectural simulations and validation of high level models. Throughout this work, the architectures being compared against our proposals are that of E. Hokenek et al. [3] [22] [23] and Suzuki et al. [24], which reflect recent thoughts in this area. With floating point multiply accumulate fused architecture, our scheme offers a worst case fanout weighted switching activity reduction of better than 49%. The speed performance of the proposed floating point adder is at least 40% better than that of [24]. With the multiply accumulator also, the speed performance can be better than that of IBM RISC/6000 [3] [22] [23], but there are certain restrictions as well. With our scheme, the architectural power/delay measures are the least, provided that there is room for pre-computation based transition activity scaling. With instruction driven processors, this is a viable approach. However, with stand alone systems or dedicated system solutions, the feasibility of pre-computation based activity

minimization has to be addressed from a system level perspective. Regarding hardware realization of significand alignment units (barrel shifters), our investigations suggest that delay balanced and transition activity scaled barrel shifter organizations are attractive as far as power/delay minimization of these units are concerned.

With significand multipliers, our investigations suggest that partial product generation using simple AND gates results in power/delay optimal multipliers. In contrast to modified Booth algorithm (MBA) [16] [17], AND gate based partial product generation in conjunction with partial product processing using 4:2 compressors yields a worst case power reduction of better than 10% with a corresponding reduction in energy delay of better than 15% for 8 bit integer multipliers. Apart from the power/delay advantages, the simpler partial product generation method also yields better layout regularity.

With floating point hardware design, evaluation of operations of the type $|A - B|$ involving integer data A and B is best addressed by using 1's complement adders. Compared to 2's complement adders, the 1's complement adders evaluate $|A - B|$ using half the hardware, by virtue of which the power consumption and silicon area of such realizations are roughly 50% of that of their 2's complement counterparts. The inherent data comparison measure offered by these units render them all the more attractive.

While the designers do have control over the design of data/control paths of instruction driven processors, they however do have no control on the type of application programs that may run on their processors. In order to validate the effectiveness of some of the methodologies we developed in our work for power minimization, we undertook an investigation into the impact of different realizations of the same algorithm on power. Taking

IIR filters as a typical example. our investigations suggest that direct form realizations are power optimal. the worst case reduction in switching activity is better than 24%. The insight gathered through this investigation is useful in developing system level methodologies for power minimization.

1.6. Thesis organization

The rest of the thesis is organized in the following way. Chapter 2 introduces an overview of some of the proven strategies for low power CMOS VLSI design. In particular, the concept of transition activity scaling is presented. Chapter 3 gives an overview of floating point basics, with particular emphasis on algorithms that implement arithmetic operations. The IEEE recommendations for floating point design are presented. Chapter 4 presents the behavior of exponents/exponent differences encountered in floating point computation, with particular emphasis on temporal correlation characteristics, effect of structural transformations of algorithms, arithmetic optimality etc. Chapter 5 highlights the applicability of 1's complement arithmetic for the design of power/performance optimal floating point units. Chapter 6 presents the development of new approaches for power/performance optimal design of significand multipliers and barrel shifters. Chapter 7 presents our proposal for a low power/low latency architecture for floating point addition. Chapter 8 presents the architectural design of our proposed multiply - accumulate fused FPU. Chapter 9 presents the structural power implications of floating point additions during IIR filtering. Chapters 4 to 9 also blends the results of our investigation as well as the underlying models that are relevant to the topic under consideration. Chapter 10 concludes the thesis. This chapter also presents some proposals for future work.

1.7. Conclusion

Development of higher level design methodologies for the realization of level low power scalable floating point arithmetic units has a definite place in the design of embedded system solutions as well as the design of microprocessors/DSPs. Since the degrees of freedom in this area is less compared to full custom designs, power reduction is targeted through architectural optimizations as well as power management strategies.

Chapter 2

Low Power CMOS VLSI Design

2.1. Introduction

In CMOS circuits, algebraic evaluation of arithmetic/logic functions is accomplished by subjecting MOS devices into switching operations, which invariably results in power consumption. Power conscious design approaches target reductions in operational power demand through a variety of techniques. The first and foremost among such approaches is the reduction in power wastage. For example, it is possible to implement a certain Boolean function in more than one way while it is not impossible that the different implemenatations consume different amounts of power though the end results given by all such implementations are identical. The operational power demands of various implementations can be evaluated by estimating the respective switching activities. The lower the switching activity of a certain implementation, the more desirable the implementation. Apart from switching activity reduction, low power design also targets power reduction through signal swing and operating voltage reductions. System level power reduction approaches include technology scaling, clocking strategies, algorithmic optimizations etc. As far as circuit level power optimization is concerned, pass transistor based logic families [75] [76] [77] [78] [79] [80] [81] [82] [83] and adiabatic logic circuits [84] [85] [86] [87] [88] [89] [90] [91] [92] [93] [94] offer power savings. With design synthesis for low power, design decisions related to architecture, algorithms, number systems, clocking strategies and problem redefinitions etc. are viable while circuit level design decisions are not, owing to the fact

that. the final implementation may use any available target technology and/or cell libraries. The best strategy the designer can resort to for circuit level power optimizations is to choose appropriate architectures which render good layout regularity so that the routing complexity is minimal (and hence power wastage in parasitic capacitances) which also comes with another added bonus, viz. time savings in automated routing. The following paragraphs give a brief overview of the design metrics and architectural level approaches for low power CMOS VLSI design.

2.2. Metrics for characterizing low power designs

Low power design strategies are directed towards the realization of computational structures with the minimum energy overhead per unit of computation. The power consumed by a computing structure. by itself, can in no way distinguish its merits from others as far as different design implementations are concerned. The power consumption of a circuit can be reduced by simply slowing down the circuit. Since the primary goal in all low power design efforts is the realization of energy efficient computing structures without undue compromise on performance, the power consumption of a structure by itself is meaningless unless speed performance figures are explicitly stated for each case.

Since energy overhead and computational speed of logic structures speak everything about the desirability of a specific structure, it becomes a natural choice to consider these two factors for forming appropriate design validation metrics. Energy delay product is a simple and straight-forward combination of the above factors and serves well in identifying energy efficient - performance optimal computational structures. The energy delay product of a circuit cannot be reduced by slowing down the operational speed of a circuit. Energy

$delay^2$ is another metric that can be employed to validate power efficient logic designs.

2.3. Activity reduction

In CMOS logic implementations, algebraic evaluation of logic functions requires a minimum number of switching events for any implementation. But in practice, the actual number of switching events taking place in a logic network can be much higher than the minimum required. In many cases, these extra switching events need not go through full logic swings. The presence of unwanted switching events is a major cause of power consumption. The problem is quite acute for densely connected logic networks with long wiring paths. Reduction of switching activity is of paramount interest in low power designs. Activity reduction is possible through architectural refinements, circuit modifications and device sizing, data encoding, power down techniques, algebraic transformations, boolean function minimization etc.

In a complex logic network, the arrival times of inputs play a crucial role in the generation of unwanted transitions. If the delays of signal paths can be equalized so that the arrival times of the decision variables at the inputs of a logic structure are the same, the presence of spurious transitions can be minimized. In actual circuit implementations, the statistical spread in device parameters and interconnect sizes due to process tolerances may lead to unwanted transition activity [57]. In this case, the delays of logic paths are perturbed because of device parameter variations as well as changes in interconnect dimensions which may lead to the generation or suppression of hazards. Circuit simulations taking into account the process tolerances and device parameter spreads using worst case models becomes important for low power designs since the designers are usually constrained with

a tight power budget. As far as circuit design for low power is concerned, the device dimensions, interconnect widths etc. should be designed in accordance with statistical tolerance schemes [95] so that the final fabricated circuit will meet the performance goals with an acceptable spread in target specifications.

Dynamic circuits, by and large, exhibit advantages over static ones, as far as unwanted switching activities are concerned. These circuits can be designed such that races and hazards are minimum. Data latches can be used in static only designs also for activity reduction. The timing of these latches is critical as far as activity reduction is concerned. Self timing of circuits [96] is another alternative for activity reduction. In self timed circuits, the latch control signals are generated by the logic network itself so that the results are transferred to the next stage only if they are stable and valid. The important consideration in all these approaches is the power efficiency of the scheme. Additional circuits intended to reduce unwanted activities by itself shouldn't drain too much power or still worse, shouldn't increase the overall switching activity in the system.

Algebraic evaluation of logic functions, taking into account the probabilities of input decision variables can result in a reduction of the switching activities in any logic implementation. The switching activities at circuit nodes are determined by the probabilities of the input variables as well as the transmission delays of logic paths. As far as Boolean function evaluations are concerned, certain function decompositions give less switching activity than others. Since signal transmission delays can be addressed separately at the circuit level, activity reduction by Boolean function decomposition (without considering circuit delays) [58] [59] [60] by itself can be pursued for minimizing the power consumption of

logic structures. This technique is equally applicable to full custom design and standard cell based approaches.

If the signal probability of a gate g is given by p_g, the average number of signal transitions at the gate output is proportional to $p_g(1-p_g)$, which incidentally, represents the variance of the signal. The total average dynamic power consumption of a logic structure is proportional to [58]

$$\phi = \sum_{g=1}^{G} p_g(1-p_g) \tag{2.1}$$

where G represents the number of gates comprising the structure. In order to highlight the effect of Boolean function decomposition on the power consumption of logic structures, we will consider the following example [58].

Consider the Karnaugh map (k-map) of a completely specified boolean function given in Figure 2.1. The probabilities of the respective input combinations are also given in the k-map. For example p_0 represents the probability of application of the signal $\overline{X}_1\overline{X}_2\overline{X}_3\overline{X}_4$. The realization given in Figure 2.1(a) represents a canonical sum of product (SOP) form having 20 literals. Considering the transition activity of the resulting logic structure, the total number of signal transitions and hence the dynamic power consumption will be proportional to

$$\Phi_a = p_4(1-p_4) + p_1(1-p_1) + p_5(1-p_5) + p_{13}(1-p_{13}) + p_7(1-p_7) \tag{2.2}$$

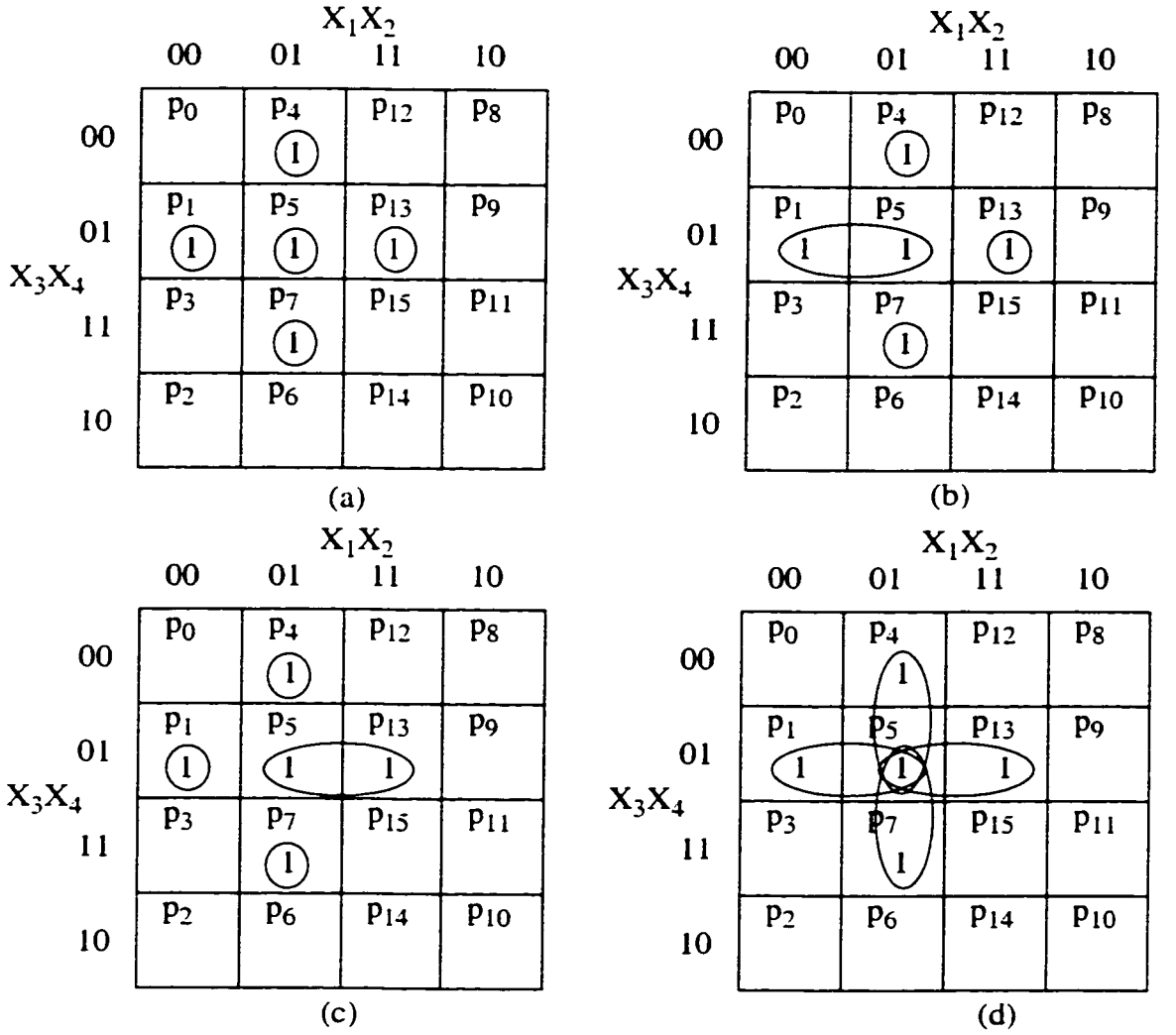


Figure 2.1 - Four realizations with different switching activities

Alternative realizations of the above are given in Figures 2.1(b), (c) and (d). The realizations shown in Figures 2.1(b) and (c) are again SOP realizations, but the number of literals is less, i.e., 15, in each case. The total number of signal transitions corresponding to these realizations are proportional to

$$\Phi_b = \Phi_a - 2p_1p_5 \tag{2.3}$$

$$\Phi_c = \Phi_a - 2p_5p_{13} \tag{2.4}$$

From the above equations, it can be easily seen that whenever two vertices i and j are merged together to form a 1-cube, the switching activity of the resultant realization is less than that of the one without merging the vertices, the actual reduction being proportional to $2p_ip_j$ where p_i, p_j are the probabilities associated with vertex i and j respectively. Figure 2.1(d) shows the minimal SOP realization which requires 12 literals, the least among all realizations shown in Figure 2.1. The signal transition activity in this case is proportional to

$$\sigma_d = \sigma_a - 2p_1p_5 - 2p_4p_5 - 2p_5p_7 - 2p_5p_{13} + 3p_5(1-p_5) \qquad \textbf{(2.5)}$$

Whenever the same vertex i is covered by n_i cubes, the switching activity of the resultant realization increases by an amount proportional to $(n_i - 1)p_i(1 - p_i)$. From the equations given above, we can easily see that realizations (b) and (c) will exhibit fewer signal transitions than realization (a) irrespective of the input signal probabilities. Larger cubes need not give an optimal activity. Realization (d) can give an average number of signal transitions that is greater than or less than that of other realizations depending on the actual value of signal probabilities. For the special case of equally likely inputs (primary signal probabilities are 0.5) the average dynamic power dissipation resulting from realization (d) will be more than that of any of the other three realizations. In other words, the optimal realization is rather a function of the input probabilities, and the conventional Boolean minimization scheme need not necessarily give the best solutions. Considering the power economy of various Boolean function decomposition schemes, the most advantageous schemes are the ones in which each vertex in the on set is covered by the largest possible

cube while every vertex is covered by the fewest possible number of cubes.

Data encoding for low power operation is yet another strategy to reduce the transition activity in logic networks. Chandrakasan et al. [61] discuss the advantages of sign magnitude representation of numbers over 2's complement representation for certain signal processing applications. Most of the arithmetic circuits work with data represented by 2's complement notation. The sign extension of the MSB bits of data give rise to unwanted activity on a number of data lines. This is particularly significant when the dynamic range of the signals being processed is much smaller than the maximum value allowed by the bitwidth of the arithmetic circuit. In this case, sign extension affects a number of MSB bit lines. Sign magnitude representation is attractive in such situations so that only one bit - the sign bit - toggles when the signal goes through a positive to negative transition and vice versa. The reduction in transition activity in sign magnitude system over 2's complement system is a function of both the dynamic range of the signals as well as the signal correlation co efficient. Sign magnitude representation gives best results for those applications that exhibit a large negative correlation between adjacent data samples and the dynamic range of the signal is much less compared to the maximum possible.

Activity reduction is also possible by selectively disabling the clocks to those logic blocks that need not be functional during certain intervals of time. This scheme is attractive because of power savings from the disabled logic blocks as well as the reduced power drain from the clock network. Activity on bus lines can be reduced by low power data encoding schemes [62] [63], so that the toggling of bus lines is minimum. The energy savings realizable through such an approach can be quite significant since the capacitances

associated with on chip buses and I/O pins are much higher than the node capacitances of logic blocks. The overhead in all these schemes is the power consumption of the extra circuits that effect activity reduction.

2.4. Transition activity scaling

As discussed earlier, activity reduction through clock gating offers promising results as far as architectural design for low power operation is concerned. In this approach, the hardware implementation of algorithms are partitioned into a set of mutually exclusive, distinct, gated clock data paths. During any computing cycle, the operations are mapped into only one gated clock logic block. While the target logic block evaluates the requisite logic function, the logic assertion status of the circuit nodes of all other data paths are preserved at their previous states, so that the switching transitions within these data paths are inhibited. Such an approach had been first proposed by Chandrakasan et al. [61]. In CMOS logic implementations, the time averaged dynamic power consumption is related to the switching activities of circuit nodes by the following equation.

$$P \propto ESW \tag{2.6}$$

In equation (2.6), *ESW* represents the expected number of switching transitions. Architectural design for low power operation targets the minimization of *ESW*. To begin with, let us assume that the hardware implementation of an algorithm is partitioned into k data paths. With transition activity scaling, *ESW* is minimized. The power consumption of such a scheme can be represented by

$$P = \sum_{i=1}^{k} P(i)P_i \tag{2.7}$$

where $P(i)$ represents the probability that the operation is mapped into the ith data path while P_i represents the power consumption of the ith data path. Without activity scaling, the power consumption can be as high as $\sum P_i \ldots \forall i$. In equation (2.7). $P_i \propto P(i)$. Because of this, the question of power minimization cannot be formulated into a linear optimization problem. With architectural design, however, power minimization can be achieved through heuristics. The mapping of the algorithm into distinct data paths is governed by factors like: silicon area, layout regularity, data path simplifications, delay reduction, delay equalizations, pipelineability, functional units reusability etc.

2.5. Conclusion

A brief overview of low power design strategies with particular emphasis on those that are viable in architectural design is presented. Design for low power is different from that for high performance or minimum area. Aggressive low power design strategies make use of every possible trade-off for reaching the specified goal. Many a time, silicon area and/or throughput are/is traded for achieving low power solutions. Crucial design decisions can be arrived at through the formulation and analysis of suitable architectural models.

Chapter 3

Overview of Floating Point Arithmetic

3.1. Introduction

The introduction of floating point number systems had been motivated by the desire to represent an infinite set of real numbers into a finite bit field. Even though floating point number representations facilitate the squeezing of an infinite set of numbers into a workable bit field (by way of certain compromises as far as the precision of the number representation is concerned), it is seldom possible to accurately represent all real numbers into an equivalent floating point number. Rather, floating point representations are approximations of real numbers. In general, a floating point number system is completely specified by specifying a suitable base, precision and sign. Table 3.1 lists the format parameters of floating point numbers specified by the IEEE 754 [4] [5] [6] [7] standard.

Table 3.1: Format parameters of IEEE 754 floating - point standard

Parameter	Format			
	Single	Single Extended	Double	Double Extended
Precision (p)	24	≥ 32	53	≥ 64
Maximum value of exponent (e_{max})	+ 127	$\geq$ + 1023	+ 1023	> + 16383
Minimum value of exponent (e_{min})	-126	$\leq$ -1022	-1022	$\leq$ - 16382
Exponent width in bits	8	≥ 11	11	≥ 15
Format width in bits	32	≥ 43	64	≥ 79

As far as hardware implementation of floating point units are concerned, the choice of certain floating point format is governed by factors like the dynamic range requirements, maximum affordable computational errors, power consumption etc. The dynamic range offered by floating point units are much greater than that offered by fixed point units of equivalent bit width. The bit width of the exponent dictates the dynamic range of floating point numbers while the significand bit width decides the resolution. Larger dynamic ranges can be of significant interest in many computing applications. For example, the multiply - accumulate operation is fundamental in digital signal processing applications. An FPU having a higher dynamic range can handle MAC operations over a large number of data samples. In fixed point units, handling of an equivalent computational task may demand the scaling of data samples so that overflow conditions may not slow down the computation. The main problem associated with overflow handling in instruction driven microprocessor/DSP architectures is the instruction overhead. Many of the digital signal processing applications, particularly the ones which work with real time data, demand zero overhead looping, in case overflows do occur which demands significant hardware resources to handle this situation. Though floating point units of wider exponent bit widths are good for many applications that require a large dynamic range, it is not necessary that wider bitwidths should be provided for all applications. The actual bitwidth required in many applications need not match precisely the ones provided by IEEE standards even. In any case, custom specification of floating point format do find quite a lot of applications. The question of format scalability of floating point units becomes important in such situations.

Since floating point numbers are approximations of real numbers, there is no way to guar-

antee absolute accuracy in floating point computations. Even though the concept of absolute accuracy is a mirage in floating point computations, it is natural to expect that computational operations on floating point operands do not introduce errors that are appreciably greater than that introduced while real operands were converted into floating point operands. IEEE 754 standard do prescribe computational measures to ensure 'acceptable accuracy' of floating point computations. The first among such measures is the question of rounding. The IEEE standard requires that numerical operations on floating point operands produce exactly rounded results. In actual operations, this requirement necessitate the computation of exact results and the rounding of such results to the nearest floating point number using the 'round to nearest/even' approach. In practice, with limited precision hardware resources, though it is impossible to compute exact results in its true sense, the introduction of guard digits ensure the computation of results within acceptable accuracy using certain minimum overheads. In general, numerical accuracy of results within acceptable limits can be guaranteed by providing two guard digits and a third sticky bit.

3.2. Floating point addition algorithm

Ref. [97] lists an addition algorithm for IEEE compatible floating point adders. The addition algorithm presented in the following paragraphs is suitable for the implementation of floating point adders whose significands are represented as signed-magnitude numbers (the significand and exponent data fields can contain any number of bits). The applicability of this algorithm is restricted to the specific category of floating point numbers whose significands are invariably normalized, by virtue of which a leading 1 is guaranteed in the significand data field. This algorithm, as such, cannot be applied for the addition of denormalized floating point numbers. In subsequent discussions, the following notation is fol-

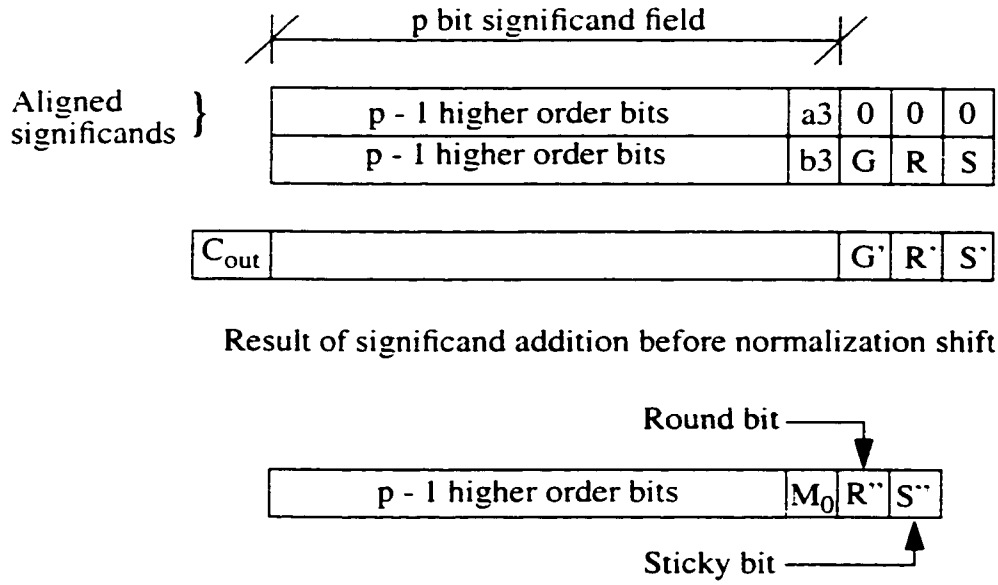


Figure 3.1 - Significand addition, normalization and rounding

lowed. *e1* and *e2* represent the exponents of the floating point operands while *s1*, *s2* represent their corresponding significands. Here, it is assumed that, the hidden leading bits of the significands are already introduced. Addition of two floating point numbers *NUM1* and *NUM2* is a five step process, the details of which are explained in the following paragraphs.

- *Step 1: Evaluate the relative magnitudes ($e1 > e2$ or $e1 \leq e2$) of the exponents as well as the difference between them ($|e1 - e2|$). Take the larger exponent as the tentative exponent of the result.*

- *Step 2: Shift the significand of the smaller number right through a number of bit positions that is equal to the exponent difference. The aligned significand should have two guard bits. Or in other words, for p bit significands, the effective width of*

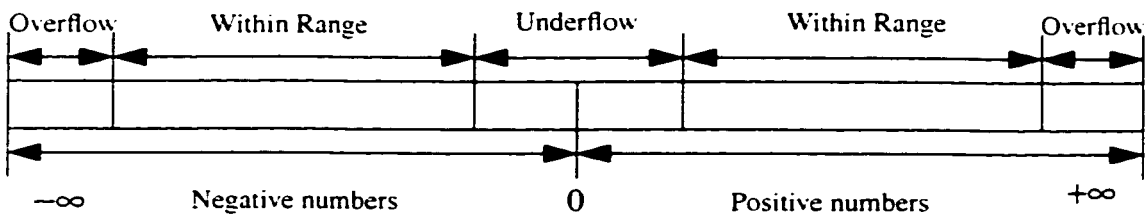


Figure 3. 2 - Range of floating point numbers

the aligned significand must be p + 2 bits, so that two of the shifted out bits are retained as guard (G) and round (R) bits. Append a third bit, namely the sticky bit (S), at the right end of the aligned significand. The sticky bit is the logical OR of all shifted out bits.

- *Step 3: Add the two signed-magnitude significands using a p + 3 bit adder. The bits at G, R, S positions of the significand of the large number must be set to zeros before the sign-magnitude addition. The result of this addition is referred to as PSUM. (PSUM is also a magnitude).*

- *Step 4: Check PSUM for carry out (C_{out}) from the most significand bit position during addition. Shift PSUM right by one bit position if a carry out is detected. During subtraction, if there is no carry out, check PSUM for leading zeros. Shift PSUM left until the MSB of the shifted result is a 1. Correct the tentative exponent set earlier, according to the normalization shift conditions. Set the new round bit (R") as the (p + 1)st bit from the left end of the normalized significand and the new sticky bit (S") as the logical OR of all bits towards the right of the R' bit.*

- *Step 5: Round the result if the logical condition R"(M0 + S") is true, where M0*

represents the pth bit of the normalized significand (or LSB of a p bit significand). If the rounding condition is true, a 1 is added at the LSB position (pth bit) of the normalized significand.

In Step 3, the signed magnitude addition may fall into any of the following categories. Both the significands can have the same sign (two cases) or the significands can have opposite signs (two cases). Depending on the sign and magnitude of the numbers, the result can be positive, negative or even zero. In any case, the final result should be presented in sign magnitude form. In Step 4, the normalization shift can be a 1 bit right shift (in case of a carry out from the MSB position due to addition) or a left shift through a number of bit positions (as dictated by the number of leading zeros, the leading zeros occur due to subtraction) or the shift can be even zero. In Step 5, rounding can sometimes result in the generation of a carry out (this happens if all the bits of the normalized significand are 1s). Rounding in Step 5 can result in the addition of 1 to the magnitude of the normalized signed magnitude sum of significands. For positive results, this leads to an addition of 1 to the significand while the opposite is true for negative results.

Figure 3.1 illustrates the process of significand addition, normalization and rounding. In step 3 of the addition, the *G*, *R* and *S* bits [8] facilitate the precision enhanced addition. Once the process of signed magnitude significand addition is completed, the immediate operation that follows this step is normalization. With the normalized significand, the bit towards the right of LSB becomes the round bit while the logical OR of all other bits towards the right of this bit is the sticky bit.

3.3. Floating point multiplication

Compared to addition, the handling of floating point multiplication is far less complicated. With normalized floating point numbers, the significands are multiplied and the exponents are summed together. With biased exponents the sum of two exponents is doubly biased, hence bias removal of this sum is essential. In case the magnitude of the product of significands is greater than or equal to 2, the exponent is incremented and the product is right shifted through one bit position. With the normalized product (for p bit significands the product is $2p$ bits long), rounding is performed. Here again, the G, R and S bits facilitate rounding.

3.4. Multiply - add

The computation of dot products (or multiply - adds; 'MAC') is essential in many scientific computing applications. With the IEEE standard, the operations of multiplication and addition should produce results that are 'IEEE compliant'. This means, two rounding operations, both the product and sum are individually rounded.

3.5. Exceptions

IEEE standard specifies a variety of exception flags for handling arithmetic operations on floating point data. Table 3.2 lists the various exception conditions. The meanings of the first three exceptions are clear from their names. The overflow and underflow exceptions are initiated whenever the result of an arithmetic operation exceeds the normal range of floating point numbers. Figure 3.2 illustrates these situations. Since the range of floating point numbers is invariably associated with the magnitudes of their exponents, it is relatively straight forward to detect overflows and underflows. IEEE standard also specifies

exception procedures (trap handlers) to handle the different exception conditions. For IEEE single precision floating point format, a value of 192 is added with an underflowed exponent. For overflow, 192 is subtracted from the overflowed exponent.

Table 3.2: Exceptions in IEEE 754

Exception	Remarks
Overflow	Refer to Figure 3.2, result can be ± ∞ or default maximum value
Underflow	Refer to Figure 3.2, result can be 0 or denormal
Divide by Zero	Result can be ± ∞
Invalid	Result is an NaN
Inexact	System Specified rounding may be required

Table 3.3: Operations that result in NaN

Operation	Remarks
Addition/ Subtraction	An operation of the type ∞ ± ∞
Multiplication	An operation of the type 0 x ∞
Division	Operations of the type 0/0 and ∞/∞
Remainder	Operations of the type x REM 0 and ∞ REM y
Square Root	Square Root of a negative number

Table 3.3 lists all possible scenarios that lead to an invalid exception. During invalid exceptions, the result is set to an NaN (not a number). NaNs are uniquely represented by setting the exponent at its maximum value (255 for single precision) and setting a non zero fraction (significand). Table 3.4 lists IEEE specifications for the representation of special quantities. In Table 3.4, e_{max} and e_{min} represent the maximum and minimum values of

exponents. With exponent biasing, e_{max} = 254 and e_{min} = 1, for single precision floating point numbers. The leading bit of the significand (hidden bit) is a zero for all the special quantities. Although the default result of an invalid operation is an NaN, an NaN operand however, cannot initiate an invalid exception. Or in other words, invalid exceptions are generated only if a relevant operand criterion specified by Table 3.3 is satisfied. Inexact exceptions are raised whenever the result of a floating point operation is not exact.

Table 3.4: Representation of special quantities

Exponent	Significand	Representation
e_{min} - 1	0	± 0
e_{min}	$\neq 0$	0.f X 2^{e}min, denormal
e_{max} + 1	0	$\pm \infty$
e_{max} + 1	$\neq 0$	NaN

IEEE specified flags are 'sticky' in nature, which means that the assertion of a flag remains the same even if the conditions leading to such assertions vanish during subsequent operational cycles. In general, the exception handling logic controls the management of trap handlers as well as flag clearing logic.

3.6. Conclusion

A brief overview of IEEE floating point standards and recommendations for arithmetic operations are presented. IEEE compliance guarantees portability of software between different platforms. Applications that doesn't envisage such portability, need not stick with the compliance requirements.

Chapter 4

Behavior of Exponents

4.1. Introduction

The transition activity scaling of FADDs and MACs is best addressed in terms of the significand alignment behavior as well as operation mix. The architectural design of a transition activity scaled triple data path floating point adder (TDPFADD) is addressed in [98]. Significant among the observations reported in [98] are: (1) The leading zero estimation circuits of FADDs need not be operational all the time and (2) The FADD operation can be bypassed during certain situations. While the validity of the first argument is intuitively clear, this is not the case with the latter. Since the evaluation of the efficacy of the bypass data path in minimizing the power consumption of FADDs demands a complete knowledge of the pre-alignment statistics (which by itself is, by and large application dependant): the absence of such statistics (experimental/analytical) renders the validation difficult. In [25] and [100], the significand alignment statistics of FP additions encountered in general purpose computing (having certain instruction mix, precedence relations etc.) is discussed. While the alignment statistics discussed in these works is rather restricted to general purpose computing, the behavior of exponent differences during repeated summation is of particular interest in DSP applications. The operation of multiply - accumulate (MAC) is fundamental in these applications. During the progress of MAC operations, the partial sums exhibit certain interesting behavior. The behavior of sums and partial sums in accumulation operation had been studied by Kolmogorov,

Andersen. Spitzer and others [99] [101] [102] [103] [104] [105]. In [106]. R. W. Hamming speculated that the behavior of exponents during a sequence of computational operations essentially exhibits non stationary characteristics. A. Lacroix et al. [107] studied the distribution characteristics of exponents and significands while Kari Kalliojarvi et al. [108] highlighted the behavior of exponent differences during floating point addition. Previously. in [109] we characterized the distribution of exponent differences during accumulation operation involving addition of a series of independent. identically distributed (IID) random variables (RVs). In floating point DSP applications. the numbers that are summed during an accumulate operation are, in general, not independent. Once the assumption of IID RVs is violated. the behavior of sums/partial sums can no longer be interpreted on such grounds. In general. the behavior of exponents of partial sums exhibits strong dependencies towards the processing algorithms as well as the distribution of input signals. For example. the sequence of arithmetic operations envisaged by an algorithm can either increase or decrease the correlation of operands of FP addition during the process of accumulation [110]. In such a scenario, simulation based characterization of exponent differences captures the various dependencies.

4.2. Exponent distribution of numbers

While the distribution characteristics of fixed point numbers is a well studied problem, the same is not true with floating point numbers. With floating point numbers, the distribution of exponents and significands require separate treatment. It is generally accepted that the significands are reciprocally distributed [106] [107]. In general, the probability density function (pdf) of exponents of numbers is a function of the distribution of numbers. With IEEE floating point numbers, an analytical expression for the pdf of exponents is given by

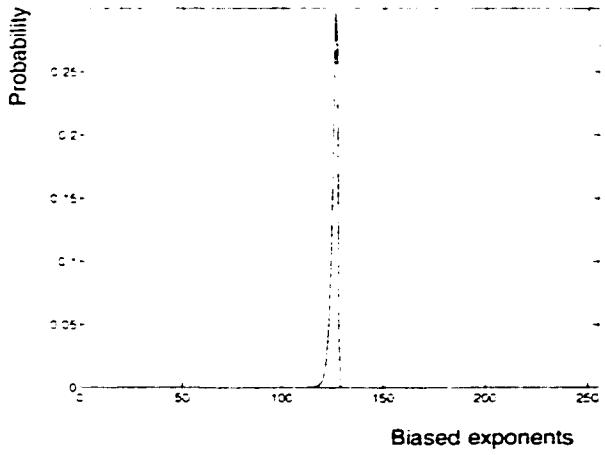


Figure 4.1 - pdf of white noise

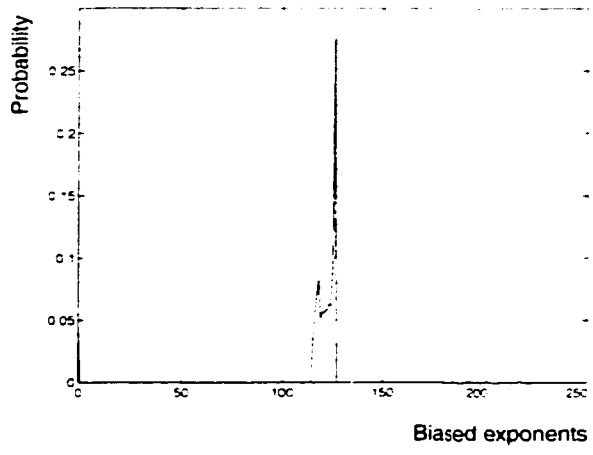


Figure 4.2 - pdf of bipolar audio - I

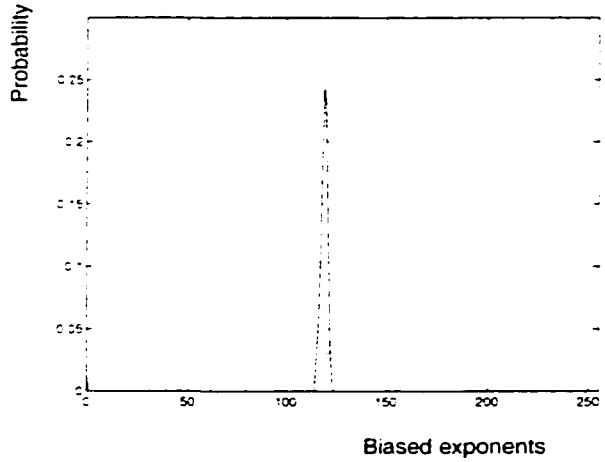


Figure 4.3 - pdf of bipolar audio - II

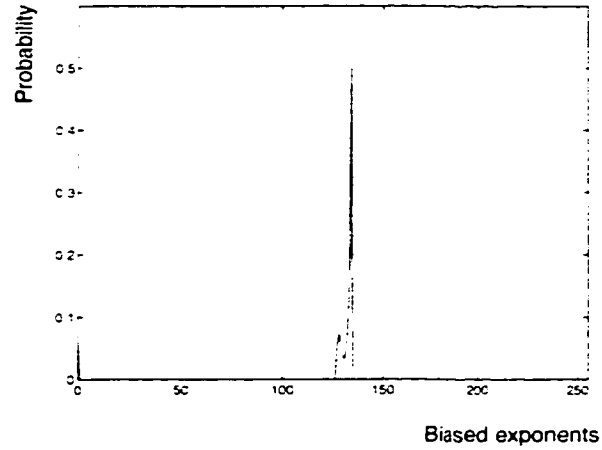


Figure 4.4 - pdf of unipolar audio

$$p(e_i) = \int_{2^{e_i}}^{2^{e_i+1}} g(x)dx \ldots\ldots\ldots\ldots\ldots\ldots (-\infty < x < \infty) \qquad \textbf{(4.1)}$$

where $p(e_i)$ represents the probability that the exponent attains a value (non biased exponent) e_i. $g(x)$ represents the pdf of the input data. Similar expressions for exponent density are given by [107] and [108]. Figure 4.1 illustrates the exponent (IEEE single precision

biased exponents) pdf of 128K samples of Gaussian distributed white noise of zero mean and unit variance ($N(0,1)$). With higher variance signals, the pdf shifts towards right and vice versa. Figures 4.2 and 4.3 illustrate the exponent pdfs of bipolar audio signals of sample sizes 6318040 and 707163 respectively. Figure 4.4 represents the exponent density of unipolar audio data of sample size 8851456. These densities are derived through the normalization of the underlying frequency distributions. As can be seen, the exponent pdfs exhibit sensitivities towards the distribution of the underlying signals, in particular, the dynamic range and power level.

4.3. Accumulation of IID RVs

The following paragraphs present an analysis about the behavior of exponent differences of the current sum and the new number to be added in accumulate operations. The accumulation of a sequence of independent, identically distributed random variables (IID RVs) is considered. Equation (4.2) defines an accumulation operation involving IID RVs (a_is).

Table 4.1: Accumulate loop

$sum = \sum_{i=0}^{n-1} a_i$ **(4.2)**	$sum^{(-1)} = 0$; for i = 0 to n - 1 $sum^{(i)} = sum^{(i-1)} + a_i$; end

Table 4.1 presents a sequential procedure that implements equation (4.2). Let us assume that a_is are Gaussian distributed RVs of zero mean and unit variance ($N(0,1)$). Since the RVs are zero mean, the sum can assume either positive, negative or zero values. In general, unless overflow/underflow exceptions do incur, the accumulate operation progresses and produces a final result that is finite.

During the accumulation of a sequence of IID RVs, the variance of the sum grows with the process of accumulation. With $N(0,1)$ IID RVs, the variance of $sum^{(i-1)}$ during a sequential accumulation process is given by $0,1,2,3,4,.....(n-1)$. With an increasing variance, the distribution of the sum is non-stationary even though the distributions of the RVs - a_is - are stationary. However, even during such a scenario, the process of accumulate operation may still be characterized in terms of certain average behavior. Let us assume the existence of an average distribution for the $sum^{(i-1)}$. For the time being, let us not worry about the precise shape of the distribution of $sum^{(i-1)}$. Let us attach an average variance to the $sum^{(i-1)}$. We will apply the expectation operator to estimate this average variance. The average variance of the $sum^{(i-1)}$ during accumulate cycles $i = 1$ to $(n-1)$ is given by

$$\sigma_{Av}^2 = \frac{1}{(n-1)}(1+2+3+.....(n-1)) = \frac{n}{2} \qquad \textbf{(4.3)}$$

The probability density function of $sum^{(i-1)}$ is comprised of two components. The first component is discrete, defined by a weighted impulse of value $1/n$ at the origin. The reason for this behavior is simple. In the beginning of an accumulate operation, the sum is forced zero, by virtue of which for every n cycle accumulation, $sum^{(i-1)}$ assumes a value of zero with a probability $1/n$ when $i = 0$. The second component of the pdf of $sum^{(i-1)}$ is continuous. This component is $N(0, \sqrt{(n/2)})$, scaled by an appropriate factor (scaling factor $= (1 - 1/n)$ such that the integral of the effective density function (between limits $\pm\infty$) is 1. The pdf of $sum^{(i-1)}$ is given by

$$f(x) = \frac{1}{n}\delta[x] + \left(1 - \frac{1}{n}\right)\left[\frac{1}{\sqrt{\pi n}}exp\left(\frac{-x^2}{n}\right)\right] \qquad \textbf{(4.4)}$$

The process of accumulation can now be modelled as the addition of one $N(0,1)$ RV to an RV whose density function is defined by equation (4.4). These two RVs are independent. The pdf of the exponents of operands can be computed by using equation (4.1). Once the pdfs of the exponents are evaluated, an analytical estimation of the distribution of exponent differences is straight forward. Since the partial sums and the numbers added to this sum are independent, having exponent pdfs $p_s(e)$ and $p_p(e)$, the pdf of exponent differences $p_d(e)$ is given by [108]

$$p_d(e_k) = \sum_{e_i = e_{min}}^{e_i = e_{max}} p_s(e_i)p_p(e_i) \quad \ldots\ldots k = 0$$

$$= \sum_{e_i = e_{min}}^{e_i = e_{max}} p_s(e_i)[p_p(e_i + k) + p_p(e_i - k)] \quad \ldots\ldots (1 \le k \le e_{max}) \qquad \textbf{(4.5)}$$

where, k represents the absolute value of the difference between the exponents while e_{max} and e_{min} represent the maximum and minimum values the exponents may assume. For input RVs with a distribution $N(0,\sigma)$, the expected variance of the current sum is given by $\sigma_{AV}^2 = n\sigma^2/2$.

Based on the above model, the pdf of exponent differences, which is anticipated during the accumulation of 4096 IID RVs with a variance of 50,000 had been computed. Accumulation using double precision floating point addition is envisaged. Figure 4.5 presents a truncated region of the computed density function. Figure 4.6 illustrates the pdf of exponent differences that is experimentally observed. The experiment involved the accumulation

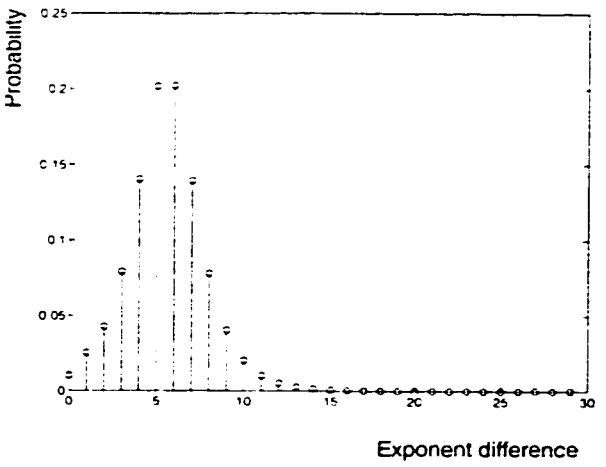


Figure 4. 5 - Accumulation - computed pdf

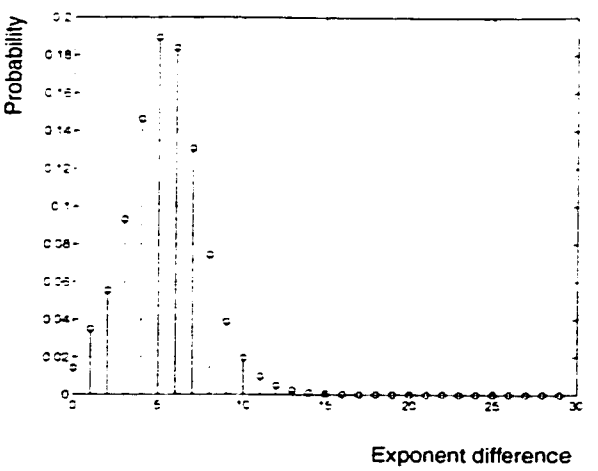


Figure 4. 6 - Accumulation - experimental results

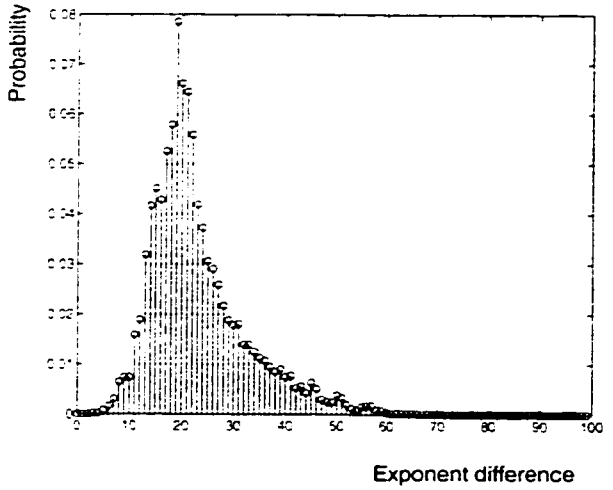


Figure 4. 7 - pdf in Schur norm computation

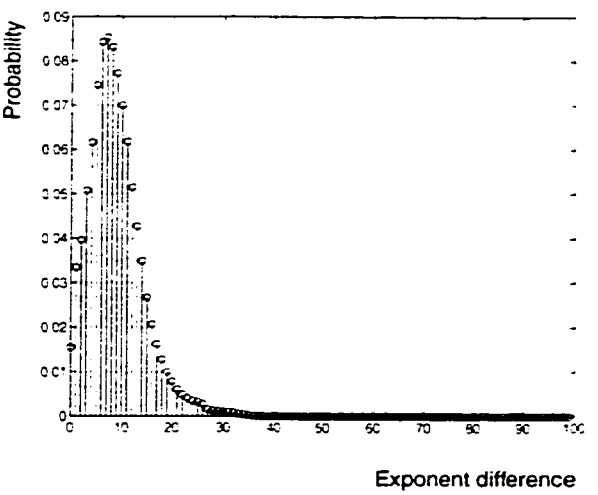


Figure 4. 8 - pdf in matrix inversion (10 iterations

4096 double precision, $N(0, \sigma)$ RVs with the same variance of 50.000. The trials were repeated 500 times. The pdf shown in Figure 4.6 represents the average of all experimental observations. Figures 4.5 and 4.6 are in good agreement as far as the shapes are concerned. (The pdfs represented by Figures 4.5 and 4.6 remain more or less the same even if the input RVs are scaled by a constant - which results in an increase/or decrease in variance). The actual probabilities, however, do differ by small values. The discrepancies between these two results arise mainly due to the following: Though the experiment was performed

over 500 times, the number of trials is small for getting a high degree of accuracy in the statistical averages. The other reason is related to the non linerarities associated with the processing of floating point operations. The theoretical model considers continuous distributions for finding an average variance etc. With continuous distributions, the RVs as well as the sum can assume any real value. However, since the floating point representation of real numbers by itself is an approximation, the computer generated RVs as well as sums are discrete in nature. The theoretical model doesn't take into account the effects of overflow, underflow, rounding problems etc. In other words, the theoretical model is not exact as far as the characterization of exponent distributions involving operations with discrete floating point RVs are concerned.

4.4. Accumulation of non-zero mean numbers

The accumulation of non zero mean RVs also exhibits some interesting properties. For example, in matrix/vector computations, the computation of Schur/Euclidean norms ($\|X\| = \sum x_i^2 ... \forall i$) requires the addition of a series of squares. In this scenario, obviously, we are dealing with the accumulation of non zero mean numbers. As expected, the magnitude of the sum grows with the accumulate operation. Figure 4.7 presents an experimentally observed sample pdf of the exponent differences (actually a truncated version - exponent differences upto 100 only are shown) of the operands during such a situation. The distribution had been observed during the computation of the Schur norm of a 160X160 matrix. In general, the position of the mode of this distribution is dependant on the size of the matrix (or vector) as well as the actual distribution of the exponents of the elements of the matrix (or vector for Euclidean norm). The most interesting aspect of the distribution presented in Figure 4.7 is the occurrence of exponent differences that are

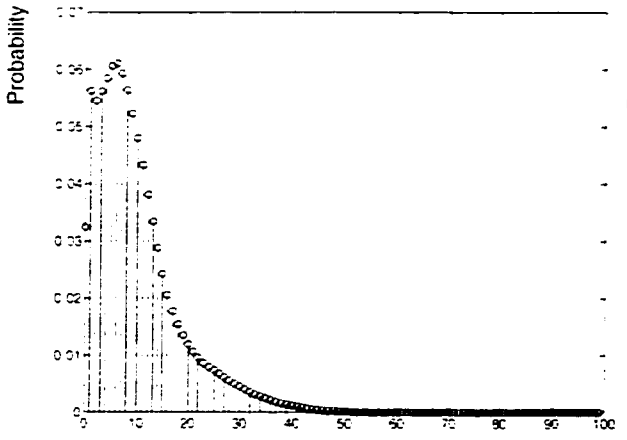


Figure 4. 9 - pdf in matrix inversion (50 iterations)

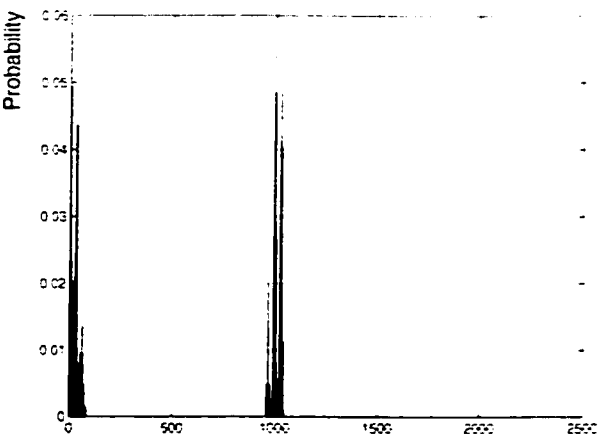


Figure 4. 10 - pdf in determinant computation

greater than the width of significands (53 for IEEE double precision floating point format).

4.5. Matrix computations: multiply - accumulation

Figures 4.8 and 4.9 present two representative pdfs of exponent differences that are experimentally observed in multiply - accumulate operations. These distributions were collected over all matrix multiplications during the computation of the inverse of a 160X160 matrix. Figures 4.8 and 4.9 differs in one aspect, viz. the number of iterations in the computation of the inverse. Figure 4.8 represents a case with 10 iterations while Figure 4.9 represents that with 50 iterations. Between Figures 4.8 and 4.9, as speculated in [106], the non stationary behavior of the pdf of exponent differences is evident. Here again, the spread of the distributions supports the fact that there are certain situations in which the operation of floating point addition doesn't produce any new result other than the larger of the input arguments.

Apart from the distribution characteristics presented by Figures 4.7, 4.8 and 4.9, there

exist a few more interesting aspects in floating point multiply accumulate operations. In MAC operations, whenever one of the operands is zero, the results are known apriori. This aspect is however, not highlighted by Figures 4.7, 4.8 and 4.9 since these Figures do illustrate only truncated regions of the actual distributions. In any floating point addition, whenever one of the operands is a zero, the exponent difference is the exponent of the non zero number. Figure 4.10 illustrates a representative pdf of the exponent differences of floating point MAC operation that involves the summation of a relatively large number of zero terms with non zero terms. This distribution had been observed during the direct computation of the determinant of a strictly diagonally dominant matrix of size 6X6. (For small matrices, compared to iterative methods, the direct computation of determinant is advantageous). In Figure 4.10, the presence of a non zero probability region in the neighborhood of exponent difference of 1000 is attributed to operations of the type $0 \pm$ number.

4.6. IIR filtering - behavior of exponents

In DSP applications, the process of discrete convolution is implemented as a series of multiply-add (MAC or dot product) operations. In order to highlight the significand alignment behavior of FADDs during IIR filtering, we will consider low pass filtering of white noise samples as a typical example. During the discrete convolution operation for filtering, the input/output samples are scaled by the filter co-efficients. Figure 4.11 illustrates a typical pdf of the exponents of filter co-efficients (8th order elliptical filter with a pass band ripple of 0.03 dB, stop band ripple of -100dB and normalized cut-off frequency of 0.2). As can be seen, this pdf exhibits more than one mode, owing to the relatively large difference between the magnitudes of forward and feedback co-efficients. The higher this difference, the larger the separation between the modes. The distribution characteristics of the expo-

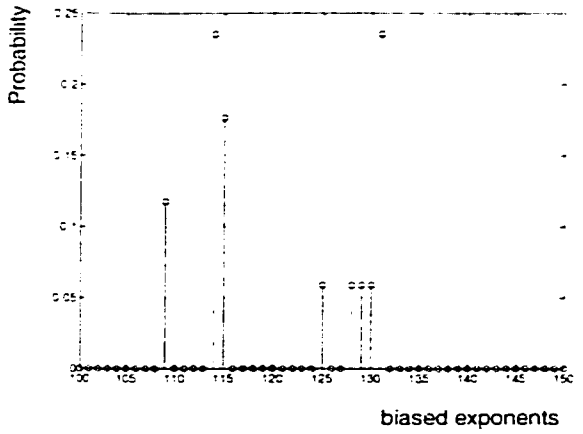


Figure 4.11 - exponent pdf of filter co-efficients

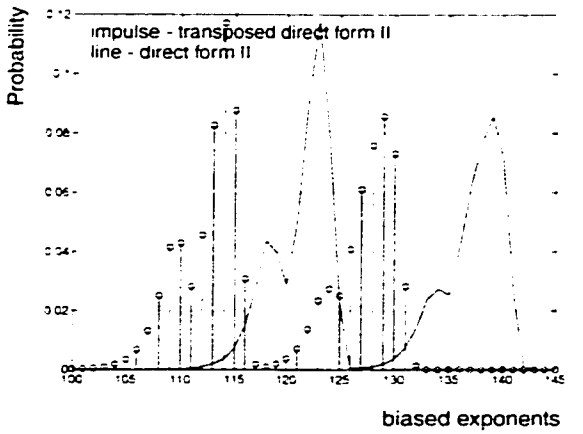


Figure 4.12 - pdf of exponents of products

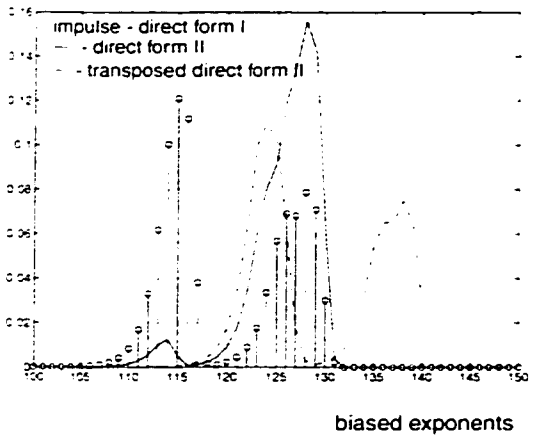


Figure 4.13 - pdf of exponents of partial sums

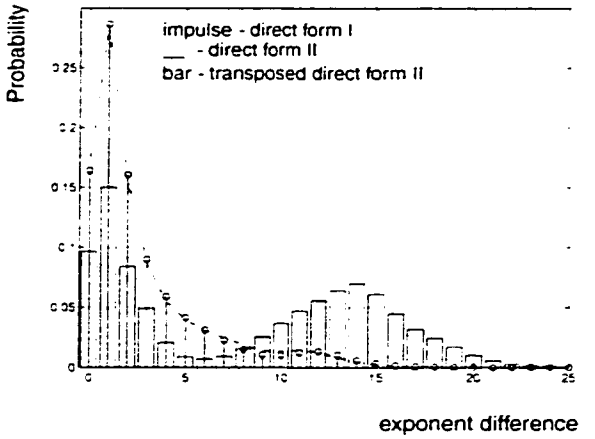


Fig. 4. 14 - pdf of exp. differences during FP adds

nents of filter co-efficients is governed by the filter specifications. Since the input/output signal samples and the filter co-efficients are independent, and the operation of amplitude scaling of the samples prompts the summation of the exponents of the signal and co-efficient: the exponent pdf of the product is obtained through the convolution of the densities of the input/output as well as filter co-efficients. With $p1(e)$ and $p2(e)$ representing the pdfs of the exponents of floating point multiplicands and multipliers, an approximation of the exponent density of the product, $p3(e)$ is given by

$$p3(e) = \lfloor p1(e) \otimes p2(e) \rfloor \tag{4.6}$$

where $\otimes$ represents discrete convolution while $\lfloor\ \rfloor$ represents a range limiting operation which restores the range the exponent of the product can assume into the standard exponent range envisaged by the number system. During multiplication overflows and underflows, the exponents are restored to the system defined default values. The convolution of the densities envisaged by the above equation, however, doesn't capture the effect of exponent increments due to significand normalization.

Figure 4.12 illustrates a truncated region of the pdfs of the exponents of products, that had been experimentally observed during low pass filtering of white noise samples of sample size 128K, by using the 8th order elliptical filter described above. As is evident from this Figure, the pdfs are bimodal. The separation of the modes is a function of the distribution of the exponents of the filter co-efficients as well as input/output signal samples. The solid line curve in Figure 4.12 illustrates the pdf of the exponents of the products when the elliptical filter is realized as direct form II (canonical form). The pdf represented by the impulses illustrates the case of a transposed direct form II implementation. The pdfs of the exponents of the products are more or less identical for the direct form I and transposed direct form II realizations. Since the relative magnitudes of the temporary variables (memory variables, which are scaled by the filter co-efficients) of the direct form II IIR filter are greater than that of the input/output data samples, the exponents of the products of this realization are comparatively large. The average difference between the exponents of the products of the transposed direct form II (as well as direct form I) and the direct form II

filters is evident in Figure 4.12: the pdf for the case of direct form II is right shifted through this average exponent difference. The shapes of the pdfs, however, are more or less the same.

Figure 4.13 illustrates the pdfs of the exponents of partial sums of the discrete convolution operation, for the different filter realizations. With repeated addition of floating point variables, the earlier the partial sum attains a relatively large value, the higher the probability that the exponent of the partial sum is relatively large. With floating point summation, once the exponent of the sum attains a relatively large value, this value remains more or less the same (with the exception of increments/decrements) till another variable that is nearly equal to this sum is subtracted or a relatively larger number is added. With the addition/subtraction of a relatively smaller number to/from a large floating point number, the exponent of the larger number can, at the most change by ±1 (+1 during addition and -1 during subtraction). Owing to the 'sticky' nature of exponents, the exponent behavior of partial sums exhibits sensitivities towards the sequence of additions as well as the exponent distribution of the products that are summed together. Since the MAC operations of direct form realizations are distributed over two separate loops, the exponent pdfs of these filters are bimodal. Each of these modes represent the exponent behavior of partial sums during repeated additions in separate loops. From Figure 4.13, it is clear that the expected values of the exponents of direct form II realizations are higher than that of the direct form I realization, owing to the summation of products, the magnitudes of which are relatively larger. Since the MAC operations of the transposed direct form II realization envisage the summation of forward/feed back products and memory variables within the same loop, the exponents of the partial sums are predominantly distributed around the second mode

(larger exponents) of the pdf of the exponents of the product. which behavior is attributed to the structure of the algorithm. Because of this. in contrast to direct form realizations - summation of product terms. the exponents of which fall around the first mode (smaller exponents in Figure 4.12) with the larger partial sums results in relatively large pre-alignment shifts. Figure 4.14 illustrates a truncated region of the pdf of the difference between the exponents of operands during FP additions encountered in filtering. The bar graph illustrates the case for the transposed direct form II filter. The distributions of direct form I and II filters are nearly identical. Owing to the summation of smaller products with the relatively large partial sums. the distribution of exponent differences of the transposed direct form II realization exhibits two modes. The first mode of the pdf illustrated in Figure 4.14 occurs due to the summation of memory variables with the feedback variables while the second one occurs due to the summation of forward variables with the partial sum.

During normalization. the significand of the sum may be left/right shifted. depending on whether the significand addition produces a result with leading zeros/exponent increment. Figure 4.15 illustrates the pdf of normalization shifts for the different filter realizations. In Figure 4.15. left shifts are represented as negative values while the exponent increment case (right shift) is indicated as a positive shift. The normalization behavior of direct form I and II realizations are more or less identical. For the transposed direct form II realization, the probability for no shift is higher than that of the other cases while the probability for left/right shifts are proportionately lower.

The distributions illustrated so far represent the time averaged behavior of exponents/ exponent differences. In CMOS logic structures, the dynamic power consumption is a

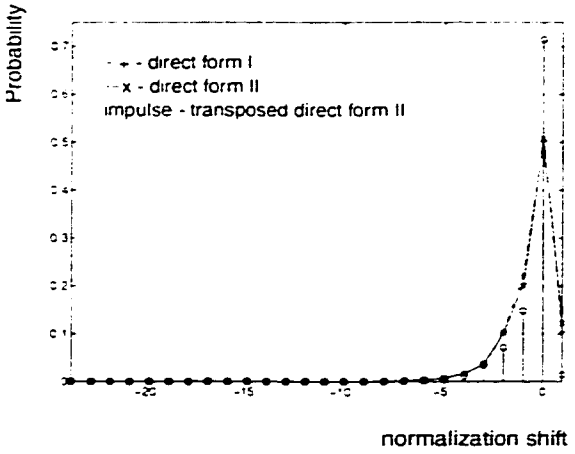


Figure 4.15 - pdf of normalization shifts

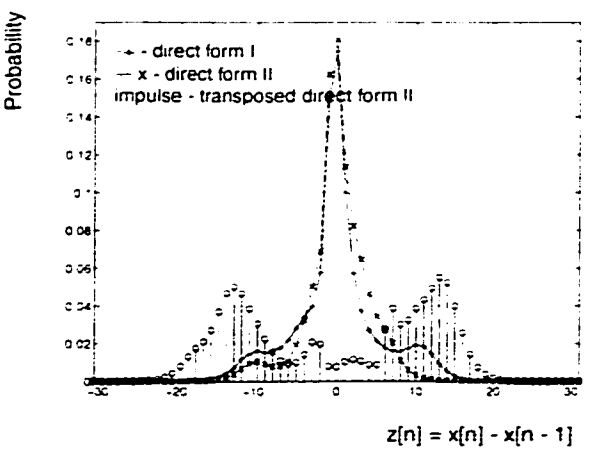


Figure 4.16 - pdf of present shift - last shift

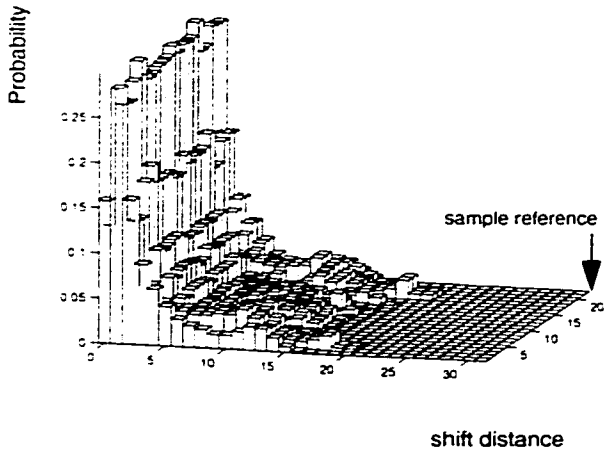


Fig. 4. 17 - pdf of pre-alignment shifts - Direct form I

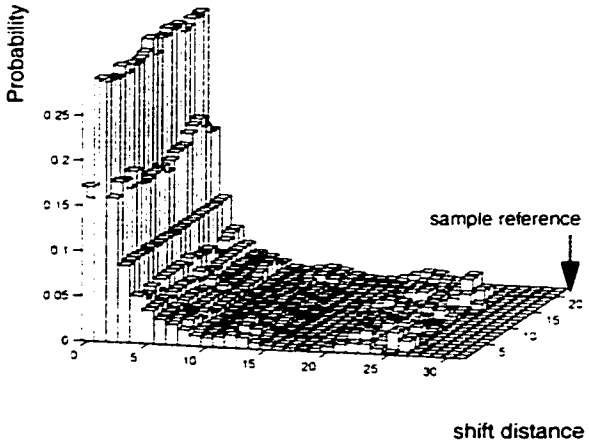


Fig. 4. 18 - pdf of pre-alignment shifts - Direct form II

function of the switching transitions at circuit nodes, which in turn is a function of signal probabilities as well as temporal correlation of logic variables [111]. In order to characterize the alignment driven power consumption of FADDs, the temporal dependance of alignment behavior also need be evaluated. Figure 4.16 illustrates the pdf (time averaged) of the rate of change of pre-alignment shifts (evaluated as present shift - last shift; $z[n] = x[n] - x[n-1]$) for the different filter realizations. With the transposed direct form II realization, the variance of the pdf is large, compared to the direct form counterparts. Also, the

notion of temporal independence assumed in probabilistic power estimation models fails, as far as the behavior of significand alignments is concerned. With independent $x[n]$ and $x[n-1]$, the pdf of $z[n]$ is obtained as

$$p(z[n]) = p(x[n]) \otimes p(-x[n-1]) \tag{4.7}$$

where $p(z[n])$, $p(x[n])$ and $p(x[n-1])$ represent the pdfs of $z[n]$, $x[n]$ and $x[n-1]$. With stationarity assumption, $p(x[n]) = p(x[n-1]) = p(x)$, in which case $p(z) = p(x) \otimes p(-x)$. In other words, $p(z)$ represents the auto correlation of the sequence $p(x)$. With the temporal independence assumption, the pdfs of $p(z)$ should essentially have their maxima at the origin. Apart from this, the pdfs also need be symmetrical about the origin. From the shape of the pdfs illustrated in Figure 4.16, it is clear that both of these characteristics fail. The non stationary behavior of exponents had been first suggested by R.W. Hamming [106]. If the pdfs represented in Figure 4.14 are stationary, the expected values of the pdfs of Figure 4.16 must be zeros. The existence of non zero mean pdfs contradicts the stationarity assumption. The shape of the pdfs also reflects the temporal correlation characteristics of shift behavior. The higher the temporal correlation, the larger the peak at the origin and the smaller the variance of the pdf. With larger, positive temporal correlation between shifts, the probability for the occurrence of situations when present shift - last shift is zero is higher. With direct form I and II realizations, the temporal correlation is higher than that of the transposed realization. From the shape of the pdf of the transposed realization, it is clear that there exists a relatively large negative correlation between present shift and last shift.

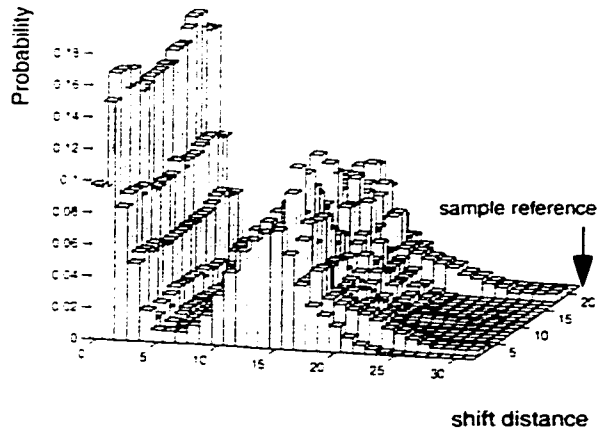


Figure 4. 19 - pdf of pre-alignment shifts - Transposed Direct form II

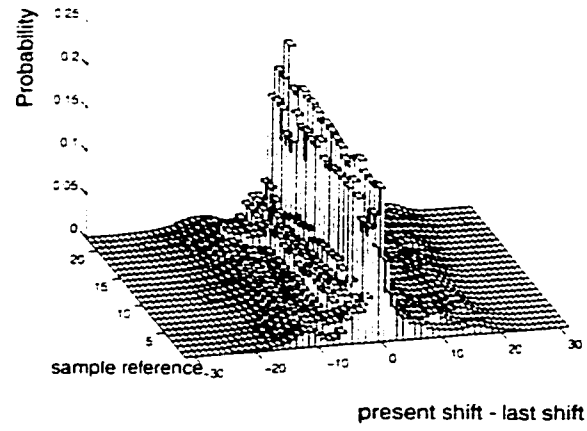


Fig. 4. 20 - pdf of present shift - last shift (Direct form I)

In order to highlight the exponent behavior of FP addition operations during filtering of practical data, we present the significand pre-alignment behavior of the different filter realizations in Figures 4.17 to 4.19. Figure 4.17 represents a family of pdfs of significand pre-alignment shifts that had been experimentally observed during IIR filtering of 22 bipolar audio signal samples [112] (ranging in size 9131 to 6318040 samples), using the direct form I realization while Figures 4.18 and 19 represent the relevant behavior for direct form II and transposed direct form II realizations. In general, in single precision FADDs, the pre-alignment shift is controlled by 5 LSB bits of the relevant exponent difference. Although these shifts are predominantly less than the width of significand, there exists certain situations during which the shifts are even greater. Whenever the exponent differences are greater than 31, the shifts imparted by the pre-alignment barrel shifters are that value of exponent difference that is reflected on the 5 LSB bits. The pdfs represented by Figures 4.17 to 4.19 illustrates the behavior of shifts seen on the 5 LSB bits. The shapes of the pdfs of direct form I and II realizations are, more or less identical. As anticipated, the pdf of the transposed realization is bimodal and exhibits comparatively larger variances. For the

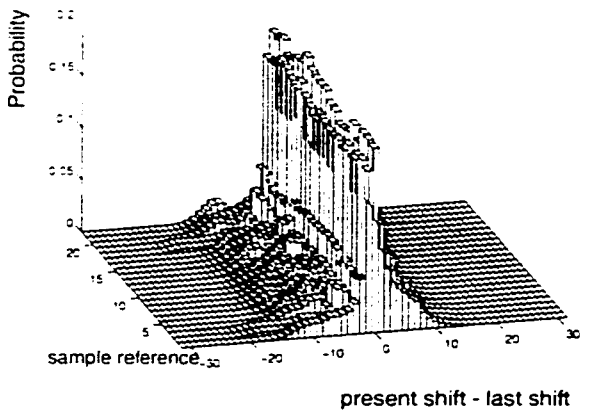


Fig. 4. 21 - pdf of present shift - last shift (Direct form II)

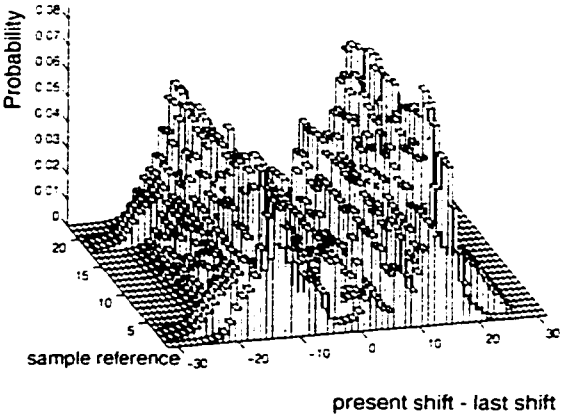


Figure 4. 22 - pdf of present shift - last shift (Transposed Direct form II)

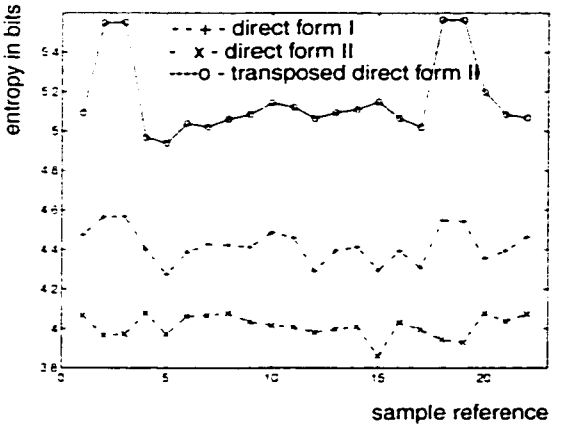


Figure 4.23 - entropy of present shift - last shift

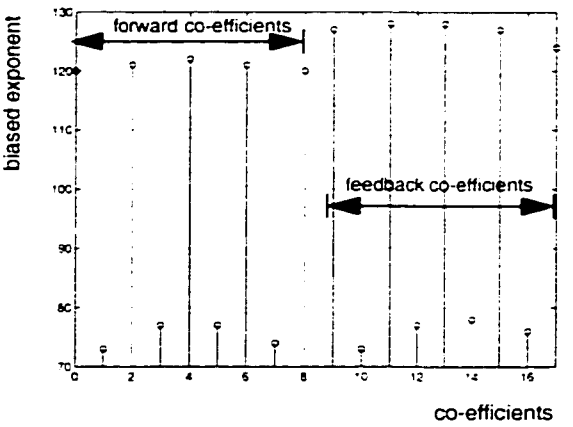


Fig. 4.24 - Exponent quantization of 8th order bandpass filter

direct form II realizations, there exists a finite non zero probability that the pre-alignment shifts exceed the width of significand. For the direct form I realization, the probability for the occurrence of such shifts is marginally low in comparison to that of others.

Figures 4.20 to 4.22 illustrate the temporal behavior of pre-alignment shifts during filtering of audio signals. As expected, the pdf of the transposed direct form II realization exhibits more than one mode. The shapes of these pdfs closely follow that of those obtained during filtering of white noise samples. From the shapes of the pdfs, it is easy to

conclude that the uncertainty between shifts during adjacent cycles is the maximum for the transposed direct form II realization. Among the three pdfs. the entropy measure is the highest for this case. Figure 4.23 shows the experimentally observed entropy of the three filter realizations. The entropy measures illustrated in this Figure are extracted from the pdfs presented in Figures 4.20 to 4.22. The entropies of direct form II realization are always the lowest among the three cases.

4.7. Arithmetically sub-optimal filtering

The exponent/alignment shift behavior presented so far reflects the magnitude correlation of FP operands of addition during low pass filtering. Similar behavior can be anticipated from high pass. band pass and band stop filters as well. However. there exists one category of filters for which the behavior is different. Band pass/stop filters with a normalized centre frequency of 0.5 belongs to this class. Figure 4.24 illustrates the exponent quantization (biased exponents) of the filter co-efficients of an 8th order elliptical bandpass filter (pass band 0.4 - 0.6, pass band ripple 0.03 dB, stop band ripple -100 dB). The filter co-efficients of Chebyshew and Butterworth filters (even FIR filters) also exhibit similar behavior. With this family of filters, the difference between the exponents of alternate filter co-efficients are relatively large in comparison with the width of significands (single precision). Also, the exponents of forward/feedback co-efficients are close to each other. Figures 4.25, 4.26 and 4.27 illustrate the temporal dependance of shift behavior of direct form I, direct form II and transposed direct form II elliptical band pass filters during filtering of white noise. Figures 4.29, 4.30 and 4.31 illustrate the corresponding exponent differences during FP additions. 128K samples of N(0,1) IID RVs had been filtered using 8th order elliptical band pass filters of normalized pass band 0.4 - 0.6; pass band ripple 0.03dB and stop band

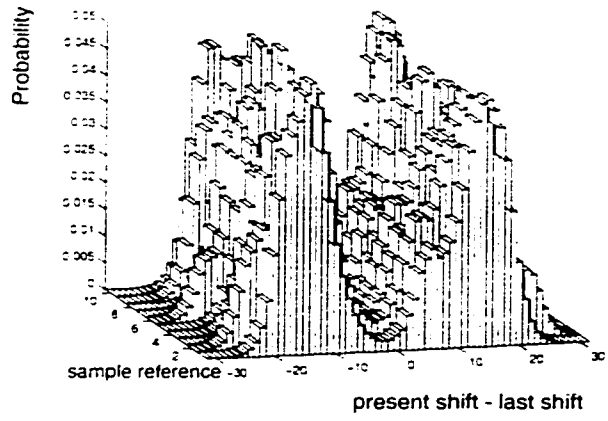


Fig. 4. 25 - pdf of present shift - last shift (I)

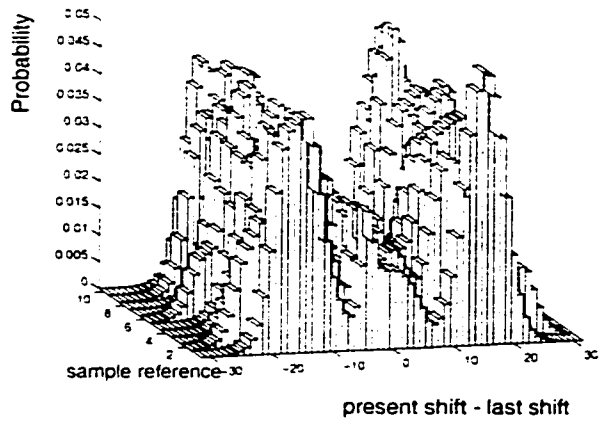


Fig. 4. 26 - pdf of present shift - last shift (II)

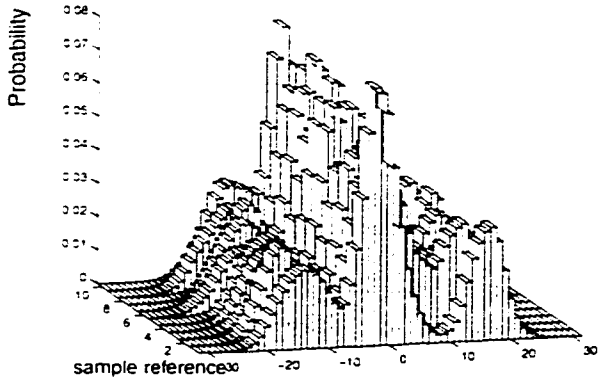


Figure 4. 27 - pdf of present shift - last shift (Transposed Direct form II)

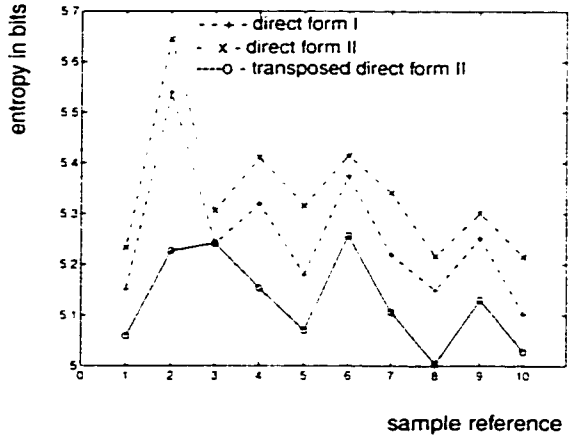


Figure 4.28 - entropy of present shift - last shift (band pass filtering of white noise)

ripples of -125dB, -115dB, -105dB, -95dB, -85dB, -75dB, -65dB, -55dB, -45dB and -35dB respectively. As anticipated, the transposed direct form II realization exhibits better temporal correlation between pre-alignment shifts. Figure 4.28 illustrates the entropy of the various realizations. Here gain, the entropy of transposed direct form II is the least among the three realizations. With the above family of filters, the probability that the FADD endures bypass conditions is more than 50%.

The effect of co-efficient quantization resulting in precision limited FADD bypasses are

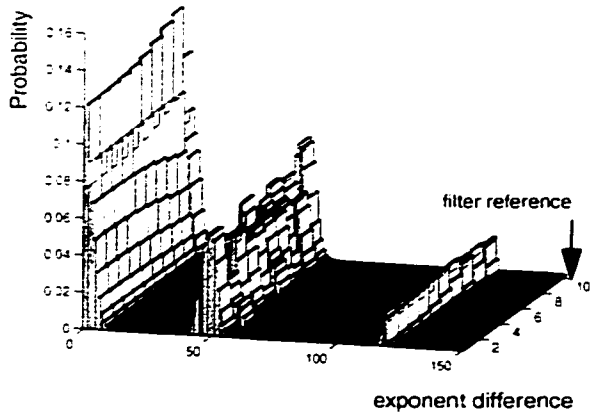


Figure 4.29 - pdf of exp. differences (Direct form I)

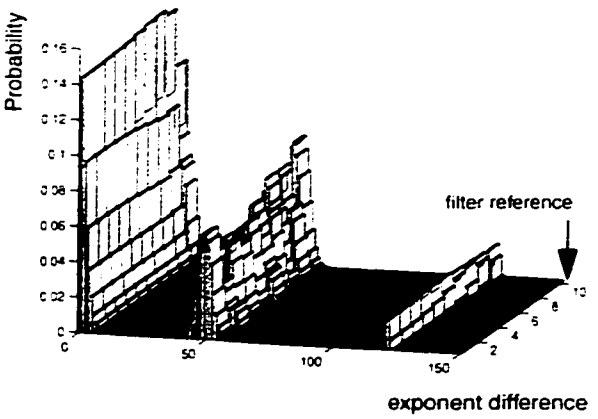


Figure 4.30 - pdf of exp. differences (Direct form II)

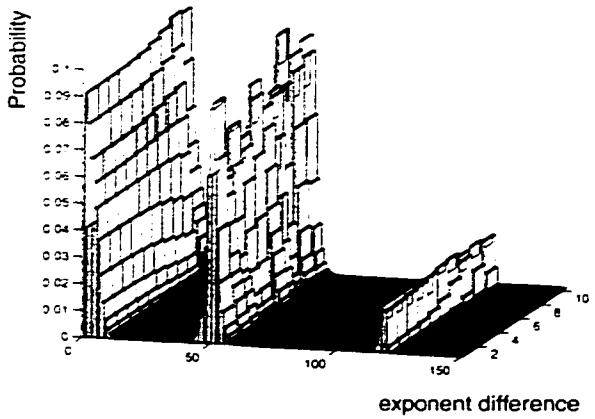


Figure 4.31 - pdf of exp. differences (Transposed Direct form II)

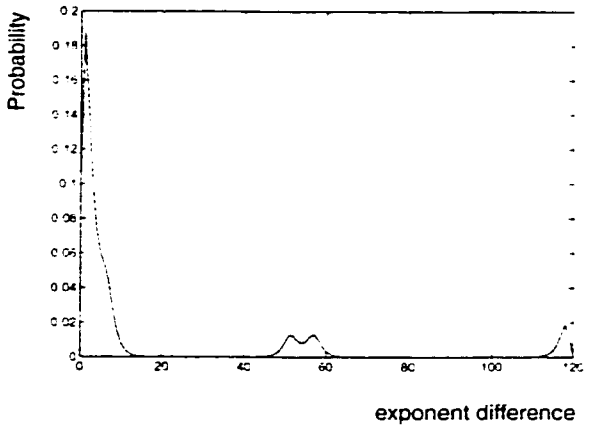


Figure 4.32 - pdf of exp. differences (filtering of white noise using order 16 FIR LPF)

quite relevant in FIR filters as well [110]. As a representative example, Figure 4.32 illustrates a truncated region of the pdf of exponent differences during FP additions in low pass FIR filtering of white noise (16th order LPF with a normalized cut-off frequency of 0.2). In Figure 4.32, the region of pdf in the neighborhood of exponent difference of 50 occurs solely due to the presence of FP operations involving the addition of smaller product terms with relatively large partial sums. Also, the region of pdf in the neighborhood of 120 occurs due to operations of the type $0 \pm$ number. In the beginning of a multiply accumulate

loop, the partial sums are forced zero. In the first cycle of MAC operation, the first product is added with this zero. The exponent of this product (single precision biased exponent) lies in the neighborhood of 120. With lower order filters, operations of this kind are of serious interest in power conscious design. In general, in real time DSP applications, the traffic of signal processing tasks involving filters of lower orders is relatively high. In the case of the order 16 FIR filter, the probability for FADD bypasses are consistently better than 18% while this factor is better than 20% for an order 64 low pass FIR filter.

4.8. Discussion

The foregoing paragraphs presented the pdfs of exponent differences as well as their first order differences during accumulation and MAC operations. Though no two of the distributions presented above do precisely match each other (because of their algorithmic/data dependencies), their shapes, in general, follow the same pattern. All these distributions exhibit non- zero modes. This aspect of the pdfs, however, is not reflected in the exponent behavior in general floating point additions [25] [100]. Also, the spread in the pdfs is another interesting aspect. The occurrence of exponent differences that are greater than the width of the significand is another notable feature. For the case shown in Figure 4.10, the probability for FADD bypass is 0.64 (sum of probabilities for exponent differences greater than 53), which is attributed to the relatively higher traffic of operations of the type $0 \pm$ number. The accumulation of short sequences is another analogous scenario. In this case as well, the probability that the FADD endures a bypass condition is significant. With a probability for FADD bypass of the order of 0.64, the TDPFADD [98] offers a power reduction of better than 5X while the reduction in power delay product is around 10X. In general, the higher the traffic of operations of the type $0 \pm$ number, the higher the power

reduction the TDPFADD can offer. Also, though operations involving non zero floating point operands can as well lead to exponent differences that are greater than the width of significand, the probability that such operations occur is very small.

4.9. Conclusion

The behavior of exponent differences during accumulation and MAC operations is presented. Both theoretical analysis and experimental evidence suggest the existence of pdfs (describing exponent differences) that exhibit non zero modes. Also, the assumption that a floating point adder can be bypassed in certain situations is valid. In floating point accumulate and MAC operations, the higher the traffic of operations of the type $0 \pm$ number, the higher the power reduction attainable through transition activity scaling. Operations of the type $0 \pm$ number, can occur due to the structure of algorithms as well as data dependencies.

Chapter 5

1's Complement Arithmetic for Low Power FPU Design

5.1. Introduction

As discussed earlier, in CMOS logic implementations, the power consumption and speed performance of functional units are, to a large extent, susceptible to algorithmic/architectural design decisions [2]. Traditionally, 2's complement algorithms are extensively used for the realization of adder/subtracters. In general, conversion of 2's complement data into absolute values is expensive as far as operational power demand, speed performance and silicon area are concerned. On the other hand, data conversion of 1's complement operands is rather straight forward. The power/delay overheads of such operations are negligibly small compared to that of their 2's complement counterparts. The following paragraphs examine the applicability of arithmetic structures based on 1's complement algorithms for low power CMOS VLSI design, with specific emphasis on floating point units.

5.2. 1's Complement adder/subtracter

Figure 5.1 gives the block diagram of a generalized significand pre-alignment scheme for floating point additions [28]. In Figure 5.1, *A* and *B* represent the exponents, *S1* and *S2* represent the significands of the aligned numbers, and *NUM1* and *NUM2* represent the floating point numbers (which include sign, exponent and significand). The blocks labelled *A* - *B* and *B* - *A* perform the indicated subtractions using 2's complement arithmetic. Since the magnitude of alignment shift is governed by the absolute value of the dif-

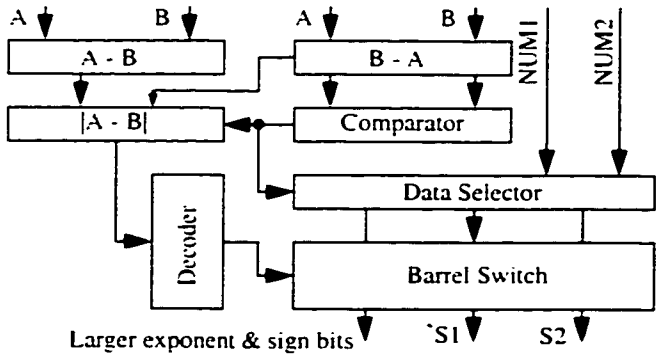


Figure 5.1 - Data alignment unit for floating point addition

Table 5.1: Significance of conditional carry outputs

$C_{out}(0)$	$C_{out}(1)$	Remarks
0	0	$C_{out}(1) = 0 \Rightarrow A < B$
0	1	$C_{out}(0) = 0$ *and* $C_{out}(1) = 1 \Rightarrow A = B$
1	0	*Impossible condition*
1	1	$C_{out}(0) = 1 \Rightarrow A > B$

ference between the exponents, a data selection block is required for the selection of the positive quantity among A - B and B - A. This data select block is controlled by the output of a comparator which performs the evaluation of $A > B$ or $A \leq B$. The output of this comparator also enables the selection of the significand of the smaller number for shifting as well as the routing of the larger number and the sign of the smaller number to the output.

The scheme given in Figure 5.1 is costly as far as design for low power operation is concerned. Arithmetic/logic units based on 1's complement addition are ideal for this kind of applications [114]. In 1's complement addition, an 'end around carry' of 1 indicates that the result is positive and vice versa. For an operation of the type A - B involving two exponents, the end around carry also reveals the truth of the condition $A > B$. The schematic of the conditional [12] carry evaluation logic of a 1's complement adder for the evaluation of $|A - B|$ (A and B represent eight bit exponents) is presented in Figure 5.2. In this scheme, the conditional carry outputs of two bit sections are evaluated in parallel. The carry evaluation logic of the higher order bit position of each two bit section is realized as a combination of an XOR and 2X1 MUXs, the logic of which is analogous to the multiplexor based

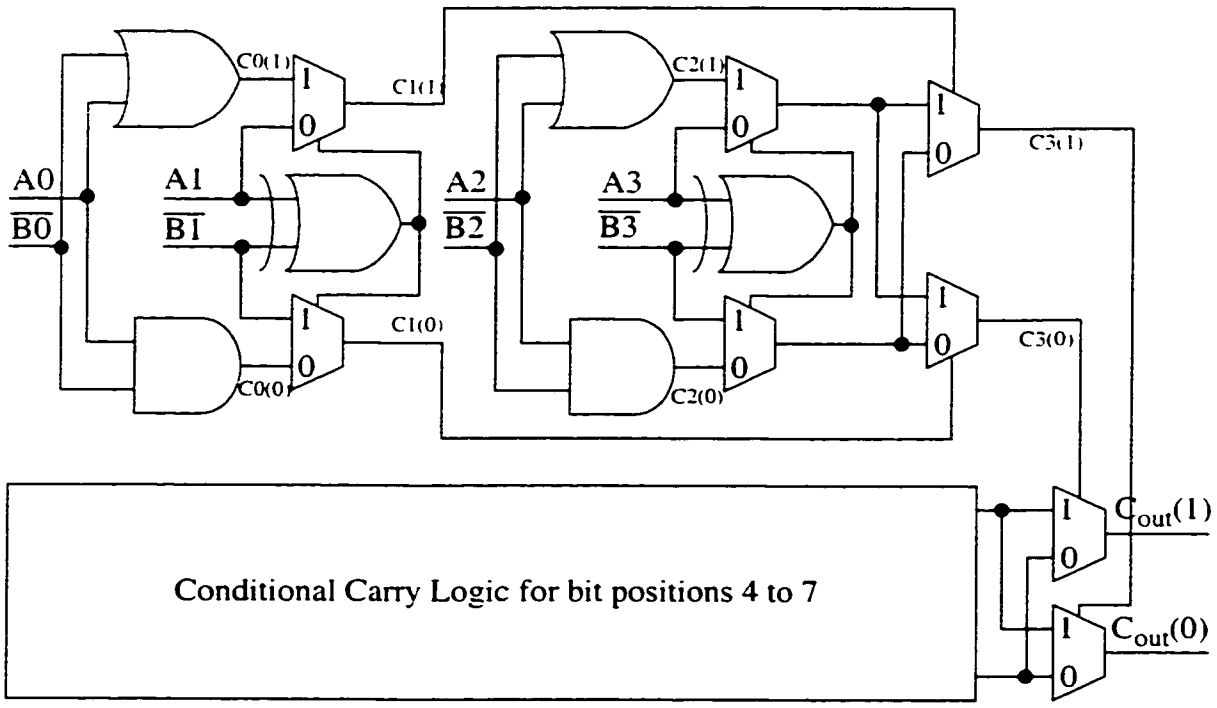


Figure 5.2 - Evaluation of 'end around carry' for |A - B| computation

full adder suggested by Norio Ohkubo et al. [47]. This scheme offers a carry out from the MSB bit position of the adder within $log_2 2n$ gate delays, with n bit exponents A and B.

In general, the conditional carry outputs of a 1's complement adder contain significant information regarding the relative magnitudes of input operands. Evaluation of conditions like $A > B$, $A \leq B$, $B > A$, $B \leq A$, $A = B$ and $A \neq B$ is relatively straight forward. Table 5.1 presents an analysis. In Table 5.1, $C_{out}(0)$ and $C_{out}(1)$ represent the conditional carry outputs from the MSB bit position of the adder anticipating an input carry (at the LSB bit position) of 0 as well as 1. Incidentally, it is worthwhile to note that a bit pattern of the type $C_{out}(0) = 1$ and $C_{out}(1) = 0$ represents an impossible condition.

1's complement adders are ideal for performing the signed magnitude addition of aligned

		0	A7	A6	A5	A4	A3	A2	A1	A0
		1	$\overline{B7}$	$\overline{B6}$	$\overline{B5}$	$\overline{B4}$	$\overline{B3}$	$\overline{B2}$	$\overline{B1}$	$\overline{B0}$
0	0	1	1	1	1	0	0	1	1	1

(a) - Input data for evaluating A - B > K condition

		C7	C6	C5	C4	C3	C2	C1	C0	∅
0	1	0	S7	S6	S5	S4	S3	S2	S1	

(b) - Input data after bit compression

Figure 5.3 - Data presentation for evaluating A - B > K condition

significands as well. The signed magnitude addition of aligned significands can produce a zero result during the subtraction of one floating point number from another when their magnitudes are equal. A direct evaluation of this condition through the decoding of the resulting significand is expensive in terms of silicon area and power. The decoding of the conditional carry outputs of 1's complement adders is attractive for the evaluation of such a condition. A single two input gate can validate the required condition. The power consumption, delay and silicon area associated with such an implementation are negligibly small.

1's complement adders are appropriate for the evaluation of the exponent of the final result. With the conditional carry outputs of these adders, evaluation of overflow/underflow conditions is a trivial operation. During addition of floating point operands, the exponent of the result is conditionally incremented, depending on whether the addition of

aligned significands has resulted in an overflow carry from the MSB position of the significand adder or not. During an exponent increment operation, a $C_{out}(1) = 1$ implies an overflow. Similarly, a $C_{out}(0) = 0$ during the subtraction of a leading zero count from the tentative exponent of the result indicates underflow.

5.3. Evaluation of 'beyond shift range' condition

In floating point additions, the process of addition need not be performed in certain situations. Whenever the magnitude of significand pre-alignment shift exceeds the width of significands, the result of floating point addition is invariably the larger number. Transition activity scaling of floating point adders [98] during such situations can result in substantial power savings. The evaluation of 'beyond shift range' (BSR) condition poses a typical data comparison problem, with possibilities for more than one architectural/algorithmic solution. In the first approach, the evaluation of the required condition - BSR - can be accomplished through the decoding of the magnitude of the difference between the exponents ($|A - B|$). The main drawback with this approach is the delay buildup.

We propose a high speed comparator, which makes use of the 'end around carry' of a three operand - 1's complement adder for the evaluation of conditions like $A - B > K$ (K represents the maximum permissible shift). Since the number of operands involved in the evaluation of $A - B > K$ condition is three, a single row of 3:2 compression logic is required for the pre processing of the input operands - A, B and K - such that a carry evaluation scheme that is analogous to that shown in Figure 5.2 can be used for the evaluation of the required condition. Figure 5.3 (a) shows the input data representation for the evaluation of $A - B > K$. In Figure 5.3 (a), the value of K is chosen as 24, which represents the maximum

allowed shift for the alignment of significands of floating point numbers that conform to IEEE single precision floating point format. Figure 5.3 (b) presents the pre-processed data. The *S*s and *C*s shown in Figure 5.3 (b) are given by

$$S_i = A_i \oplus \bar{B}_i \oplus L_i \ldots\ldots\ldots\ldots\ldots\ldots \forall i \text{ and} \qquad \textbf{(5.1)}$$

$$C_i = A_i\bar{B}_i + L_i(A_i + \bar{B}_i) \ldots\ldots\ldots\ldots\ldots\ldots \forall i \qquad \textbf{(5.2)}$$

where L_i represents the ith bit of the 2's complement of K. Though the expressions given above contain more than two independent variables, a single level of 2 input gates can evaluate the *S*s and *C*s, since the value of K is known apriori. In fact, it is not mandatory to evaluate all *Ss* and *Cs* independently. Some of these signals (or their complements) are already available as logic sub-expressions of the 1's complement adder which evaluates $|A - B|$. Evaluation of $|A - B| > K$ condition can be realized through a logical OR operation of the conditions $A - B > K$ and $B - A > K$. Comparators performing $A - B > K$ and $B - A > K$ conditions can share certain common sub-expressions/their complements. The theoretical minimum delay in establishing $A - B > K$ by this method is $1 + log_2 2n$ gate delays for an implementation using 2 input gates.

5.4. Computation of sticky bit

The default rounding mode specified by the IEEE 754 standard is 'round to nearest - even'. In this mode, the results are rounded to the nearest values and an even value is chosen, in case of a tie. In order to perform a rounding operation of this nature, the aligned significand should have two guard bits (*G* and *R* bits) and a sticky bit (*S*). The generation

of sticky bit involves the formation of the logical OR of all shifted out bits of the aligned significand. Since the guard (G), round (R) and sticky (S) bits are essential for the performance of the signed-magnitude significand addition, these bits must be available as soon as the aligned significands are available. The formation of the guard (G), and round (R) bits is relatively straight forward, two of the shifted out bits of the aligned significand are retained for this purpose. Formation of the sticky (S) bit, however, involves the conditional ORing of a large number of bits of the significand of the smaller floating point number.

M. R. Santoro et al. [115] proposed a concurrent evaluation technique for the computation of sticky bit (through the evaluation of trailing zeros of the significands of input operands) for floating point multipliers. Extending this approach to floating point adders, the sticky bit can be formed through the evaluation of trailing zeros of the significand of the smaller number. In floating point additions, the magnitude of alignment shift of the significand of the smaller number is given by $SH = |A - B|$, where SH represents the magnitude of alignment shift and A and B represent the exponents of the floating point numbers. Since the number of shifted out (discarded) bits is less than the actual shift by an amount that is equal the number of guard bits (G, R bits), the effective shift as far as sticky bit evaluation is concerned is given by $SH_{eff} = SH - 2$. If the effective shift is greater than the number of trailing zeros, the sticky bit can be set to 1, and vice versa. Evaluation of the condition $SH_{eff} > TZ$ can be formulated into: (SH - 2 > TZ), so that the end around carry of a three operand 1's complement adder can be used for the evaluation of the sticky bit. The computation of sticky bit through this approach is attractive as far as modularity and speed of operation are concerned.

5.5. Power/delay measures

Table 5.2: Comparison of Schemes for the Evaluation of $|A - B|$ and $A > B$

Operation	Parameter	1's comp	2's comp	Remarks
Evaluation of $\|A - B\|$	Normalized Area	1	$2^{\dagger}$	†Two identical adders work in parallel
	Normalized Power	1	$2^{\dagger}$	
	Speed of Operation	--	--	Same for both cases
Evaluation of $A > B$	Normalized Area	$0^{\ddagger}$	1	‡No additional Power/Area Overheads
	Normalized Power	$0^{\ddagger}$	1	
	Speed of Operation	--	--	Comparable for both cases

Table 5.2 presents a comparison between the 1's complement subtracter and its 2's complement counterpart. Compared to 2's complement scheme, the 1's complement adder evaluates $|A - B|$ in the same time, using half the hardware. The power consumption of this scheme is around 50% of that of its 2's complement counterpart. The inherent data comparison measure inbuilt with the 1's complement adder facilitates the evaluation of $A > B$ condition using no extra hardware. Assuming that comparators are as complex as adders in implementation, the overall saving offered by the 1's complement based scheme is roughly 66% as far as silicon area and power consumption are concerned.

The evaluation of 'beyond shift range' ($|A - B| > K$) condition through the carry chain evaluation of a 1's complement adder is attractive as far as reusability of logic expressions and speed performance are concerned. Evaluation of this condition through the decoding of $|A - B|$ comes with an added delay penalty. Essentially, a minimum of $1 + log_2 2n$ gate delays

is required for the evaluation of $|A - B|$. Comparison of $|A - B|$ with K takes at least another $log_2 2n$ gate delays. The delay for the evaluation of $|A - B| > K$ condition, using the carry chain evaluation of a 1's complement adder, is $2 + log_2 2n$. For IEEE single precision floating point operands, the percentage reduction in delay is of the order of 33%.

5.6. Conclusion

The foregoing paragraphs presented the merits of 1's complement adders as well as numerical comparators, for the implementation of low power floating point adders, accumulators and multiply - accumulators. The power/delay measures of these units are attractive as far as design for low power/high speed are concerned. The inherent data comparison measure offered by the 1's complement adders render them all the more attractive for low power designs.

Chapter 6

Low Power Multipliers and Barrel Shifters

6.1. Introduction

The power consumption and speed performance of significand multipliers and barrel shifters are of significant interest in floating point data path design. This chapter addresses the merits of different approaches for significand multiplication as well as barrel shifter organization. In particular, the power delay implications of partial product generation methods and transition activity scaled and delay balanced barrel shifter organizations are addressed. Signal transition activity based analytical power delay models that capture the impact of circuit implementation are developed. The validity of these models are verified through circuit/architectural simulations.

6.2. Significand multiplication

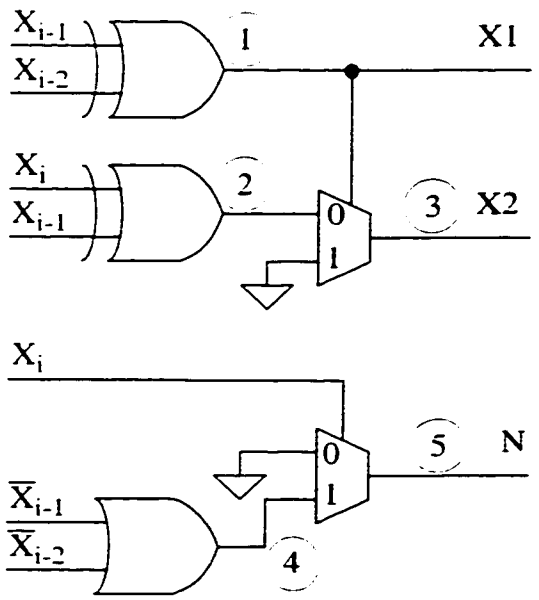
Traditionally, Booth encoding techniques [16] [17] had been widely employed for partial product reduction in parallel multiplier implementations. For parallel multipliers using radix 4 modified Booth algorithm (MBA), the number of partial products are reduced to *n/2* for an *nXn* 2's complement multiplication. This task can be accomplished by the use of 4:2 compressors [116] also, as proposed by D. Villeger et al. [18]. The authors in [18] have shown that 4:2 compressors can reduce the number of partial products to *n/2* in less time and using fewer gates compared to radix 4 MBA. Norio Ohkubo et al. [47] acknowledged the equivalence of partial product generation using MBA and 4:2 compressors, though they favored partial product generation using the MBA scheme owing to the area advan-

tage offered by such a scheme. Even though these authors speculated that partial product generation need not be accomplished by using the MBA scheme alone, they didn't weigh the energy delay measures of various schemes for reaching their conclusions. The following paragraphs address the evaluation of partial product reduction using radix 4 MBA and 4:2 compression technique on the basis of the energy delays of the relevant schemes.

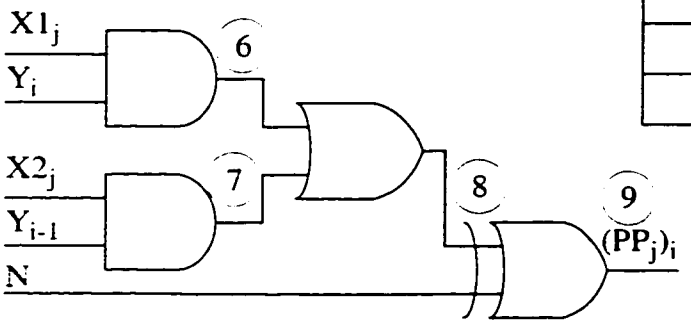
6.2.1 Circuit realization

In radix 4 MBA, partial product reduction is accomplished by selecting *0, +1, -1, +2 or -2* times the multiplicand (Y) as the partial product in accordance with the values of the relevant multiplier (X) bits. There are two possible schemes to implement MBA, MBA I and MBA II. The gate level representations of these schemes are shown in Figures 6.1 and 6.2. The MBA I makes use of three control signals *X1*, *X2* and *N* for effecting partial product bit generations while the MBA II makes use of five control signals, viz. *X1*, *-X1*, *X2*, *-X2* and *Z*. Figures 6.1 and 6.2 also list the signal probabilities (P_g) and fanouts (F_g) of various circuit nodes. Figure 6.3 shows the schematic and signal probabilities of a multiplexor based 4:2 compressor [47] [77]. The circuits had been implemented using CPL building blocks [77].

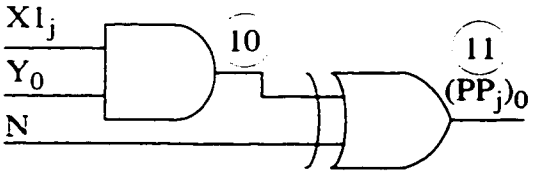
In Figure 6.1, the generation of the negation control signal *N* differs from popular implementations, in which case the logic is over-simplified as $N = X_i$ where X_i represents the MSB of the multiplier bit group being scanned. The partial product must be zero for the bit combination $X_i = X_{i-1} = X_{i-2} = 1$. The assertion of control signals for this condition with $N = X_i$ are $X1 = 0$, $X2 = 0$ and $N = 1$. This condition generates a partial product with all the bits asserted high. The addition of a correction bit of *1* (for 2's complementation) at



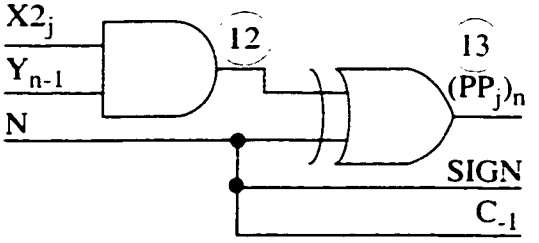
(a) - Generation of control signals



(b) - Partial product bit generation



(c) - Generation of the 0th bit of PP



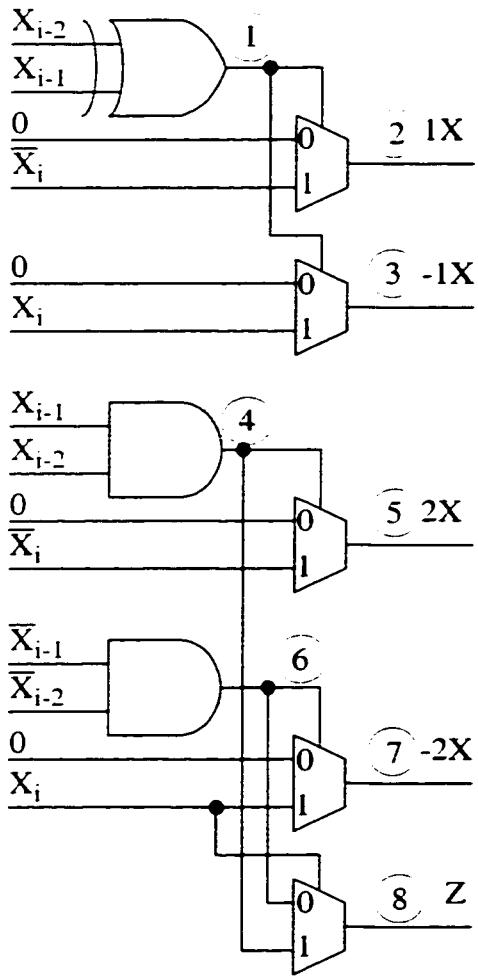
(d) - Generation of the nth, (n+1)th and C-1 bits

Table 6.1: Signal probabilities and fanouts

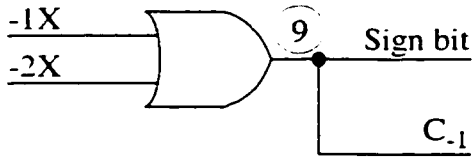
Node	P_g	F_g
1	1/2	n
2	1/2	1
3	1/4	n
4	3/4	1
5	3/8	n+5
6	1/4	1
7	1/8	1
8	3/8	1
9	15/32	2
10	1/4	1
11	7/16	2
12	1/8	1
13	13/32	4

Figure 6.1 - Implementation of modified Booth algorithm - Scheme I

bit position zero results in a partial product that is effectively zero. Though the final partial product is effectively zero and the arithmetic result is correct, this condition generates unwanted switching activity, (which ripples through the entire carry save adder array as



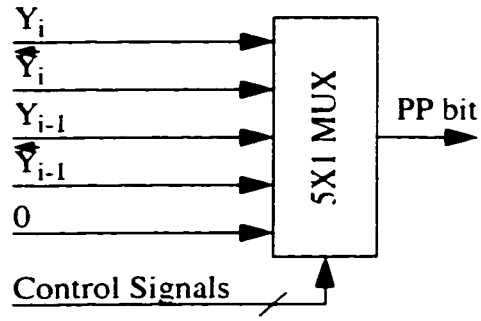
(a) - Generation of control signals



(c) - Generation of sign and carry in (C_{-1}) bits

Table 6.2: Signal probabilities and fanouts

Node	P_g	F_g
1	1/2	2
2	1/4	n+1
3	1/4	n+2
4	1/4	2
5	1/8	n+1
6	1/4	2
7	1/8	n+2
8	1/4	n+1
PP bit	3/8	2
9	3/8	4



(b) Partial product bit generation

Figure 6.2 - Implementation of modified Booth algorithm - Scheme II

well as the final carry propagate adder) resulting in power wastage. The probability of occurrence of this situation is *1/8* (considering a signal probability of *1/2* for the multiplier bits) for every partial product and hence the resulting power wastage is significant enough

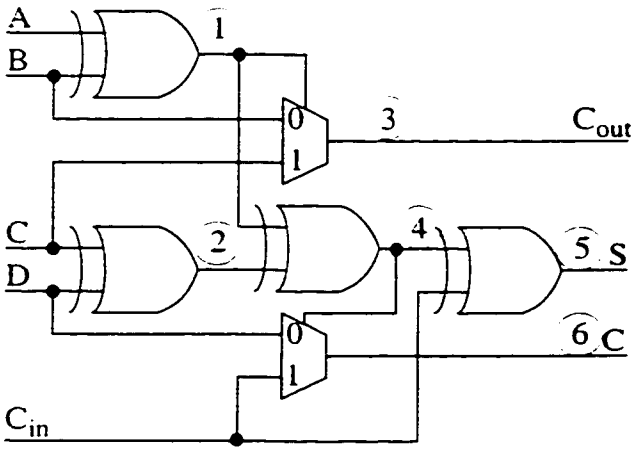


Table 6.3: AF of 4:2 compressor

Node	P_{node}	AF	F_{node}
1	3/8	15/64	2
2	3/8	15/64	1
3	5/32	135/1024	2
4	15/32	255/1024	2
5	245/512	$65415/2^{18}$	2
6	187/1024	$156519/2^{20}$	2

Figure 6.3 - 4:2 compressor

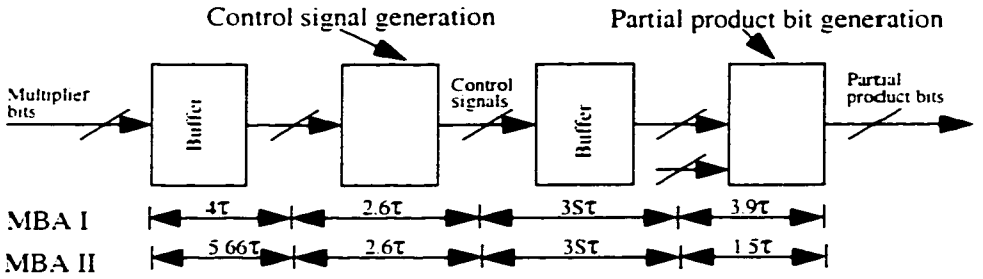


Figure 6.4 - Delay model of partial product generation using MBA

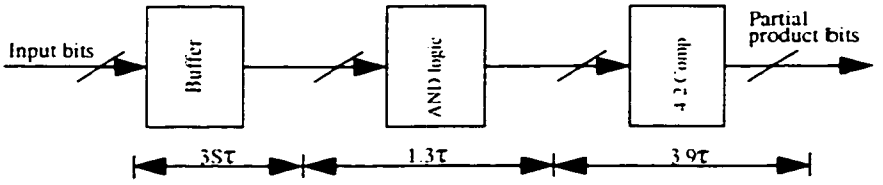


Figure 6.5 - Delay model of 4:2 compressor based partial product generation

to warrant a review of the design decisions related to the generation of *N*.

6.2.2 Delay models

Figures 6.4 and 6.5 illustrate the delay models of MBA and 4:2 compressor based partial product generation schemes. Our delay estimate of MBA I is given by $\tau_{max} = 10.5\tau + 3S\tau$

while that of MBA II and 4:2 based schemes are respectively $\tau_{max} = 9.76\tau + 3S\tau$ and $\tau_{max} = 5.2\tau + 3S\tau$ where τ represents the delay of a minimum sized inverter driving an identical inverter and S represents the stage ratio of the drivers driving the control signals for MBA and multiplier/multiplicand bits for 4:2 compressor based schemes respectively. In our delay analysis, we considered a delay of 1.3τ for a single stage of complementary pass transistor logic (CPL) gate while the delay of the 5X1 CPL multiplexor was estimated as 1.5τ. The stage ratio S of the drivers captures the effect of parasitic capacitance on circuit delays. The larger the parasitic loading, the higher the value of S. The value of stage ratio S is different in various implementations owing to the differences in layout geometries. In the above delay analysis, a three stage driver implementation is assumed for the distribution of the control signals for MBA as well as multiplier/multiplicand bits for 4:2 compressor based scheme. The worst case fanout of multiplier bits for MBA I is four and hence a single stage driver is envisaged for the distribution of these signals resulting in a delay of 4τ. For MBA II, the worst case fanout of multiplier bits is eight, resulting in a driver delay of 5.66τ considering a two stage driver for the distribution of these signals.

6.2.3 Energy delay analysis

The time averaged power consumption at the output of a CMOS logic structure is given by, $P = AF f C_L V_{DD}^2 = P_g(1 - P_g) f C_L V_{DD}^2$, where AF is the activity factor, f is the operating frequency, P_g is the probability of finding a logic high at the node under consideration, C_L is the capacitive loading at the node and V_{DD} is the power supply voltage. Here, it is assumed that the logic variables of circuit nodes exhibit temporal independence. The load capacitance at the output of a gate is proportional to the fanout of the gate. The total energy exchange (during one cycle of operation) due to signal dynamics at circuit nodes,

in any logic structure can be expressed by the following relation.

$$E \propto \sum_{\forall g} P_g(1 - P_g)F_g \tag{6.1}$$

where F_g represents the fanout of the gth gate. The right hand side of the above equation represents the energy measure of logic circuits. The energy delay product of the logic structure is given by

$$ED \propto \sum_{\forall g} P_g(1 - P_g)F_g\tau_{max} \tag{6.2}$$

where τ_{max} represents the delay of the critical path of the circuit. The above relation can be used for evaluating the energy delay measures of gate level logic representations on the basis of signal probabilities, fanouts and circuit delays. The following expression gives an estimate of the energy delay measure of MBA I, for the generation of two partial products.

$$EDM_1 = \left(\left[\frac{527}{256}n + \left(\frac{1+k}{\eta_1}\right)\frac{15}{8}n\right] + \left[\left(\frac{1+k}{\eta_2}\right)\frac{15}{32}(n+5) + \frac{1491}{256}\right]\right)(10.5\tau + 3S\tau) \tag{6.3}$$

where n represents the bitwidth of multiplier and multiplicand data, k represents the coefficient of parasitic loading (for long interconnects, in which case $C_L = (1 + k)C_g$ where C_g represents the actual gate capacitance loading the interconnects), η_1 represents the efficiency of the driver interfacing the control signals $X1$, $X2$ and multiplicand bits while η_2 represents the efficiency of the driver interfacing N, and S represents the stage ratio of the drivers. In the above analysis, the signal probabilities of the multiplier and multiplicand bits had been assumed to be $P_1 = P_0 = 1/2$. The corresponding energy delay measures for

MBA II and 4:2 compressor based schemes are given by

$$EDM_{II} = \left(\left[\left(\frac{1+k}{\eta}\right)\left(\frac{41}{16}n + \frac{69}{32}\right)\right] + \left[\frac{15}{16}n + \frac{47}{4}\right]\right)(9.76\tau + 3S\tau) \quad \textbf{(6.4)}$$

$$EDM_{4\to 2} = \left([3.575n - 1.9832] + \left(\frac{1+k}{\eta}\right)2n\right)(5.2\tau + 3S\tau) \quad \textbf{(6.5)}$$

The energy measures given by the above equations take into account architectural/algorithmic as well as circuit implementation issues. In general, signal probabilities, activity factors and fanouts capture the effects of the architecture/algorithm as well as logic design. The parameter k, on the other hand, is solely dependant on the actual implementation. The higher the wiring complexity of the implementation, the higher the value of k. The coefficient of parasitic loading k and fanout F_g decide the value of the stage ratio (S) of drivers, as given by $S = exp\ [(ln\ ((1+k)F_g))/3]$ for a three stage buffer. The efficiencies of buffers (η) are functions of stage ratios as well as number of stages.

6.2.4 Circuit simulation

CPL realizations of the various partial product generation schemes for an 8X8 integer multiplier had been simulated using HSPICE with level 3 device models of Nortel 0.8 micron BATMOS process. The simulations were carried out for the generation of two partial products using each scheme. The test vectors (X and Y) for the simulation had been generated using random number generation. By setting the MSB of X and Y as zeros, the arithmetic equivalence of integer and 2's complement multiplication (MBA) is ensured. In our experiments, we generated the random test vectors using a uniform distribution and the circuits were simulated for forty cycles of operation. The HSPICE measurements of

time averaged power and delays of critical paths had been used for the computation of the energy delays of various schemes.

6.2.5 Results

Figures 6.6 and 6.7 give the percentage reduction in energy delay of the 4:2 compressor based partial product generation scheme in comparison with MBA schemes, as given by the energy delay models of equations (6.3) (6.4) and (6.5). Figure 6.6 shows the reduction with respect to MBA I while Figure 6.7 shows the reduction with respect to MBA II. From Figures 6.6 and 6.7, it is clear that the energy delay improvement of 4:2 compressor based scheme is better than 20% for a 24X24 bit unsigned multiplier, considering a parasitic capacitance contribution that is equal to gate load capacitances for longer interconnects. The relevant figures for 8X8, 16X16 and 32X32 multipliers are of the order of 39%, 27% and 18% respectively.

Figures 6.8 and 6.9 illustrate the power/performance improvement of the 4:2 compressor based scheme on the basis of HSPICE simulations of the various schemes. From Figure 6.8, it can be seen that the 4:2 compressor based scheme offers an energy delay reduction of better than 36% in comparison with MBA I and 15% against MBA II while the reduction in power consumption is better than 26% and 10% respectively, for various instances of parasitic loading on long interconnects. Figure 6.9 gives the corresponding performance advantages with voltage scaling, considering a parasitic capacitance loading of 0.2 pF. The energy delay advantage of the 4:2 compressor based scheme is better for lower operating voltages, suggesting that this scheme is better suited for low voltage operation. SPICE simulations of the various schemes give performance figures that are not appreciably dif-

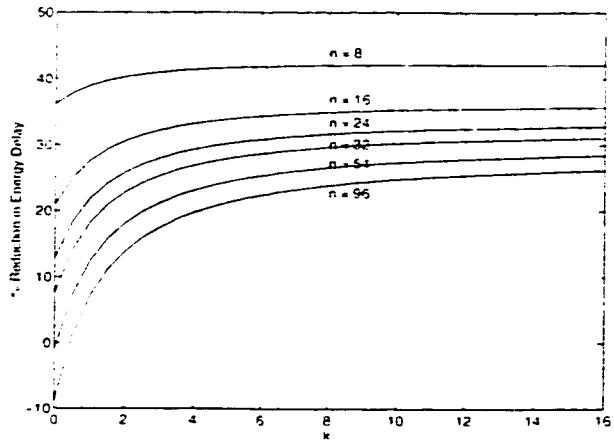


Figure 6.6 - % Reduction in ED (MBA I)

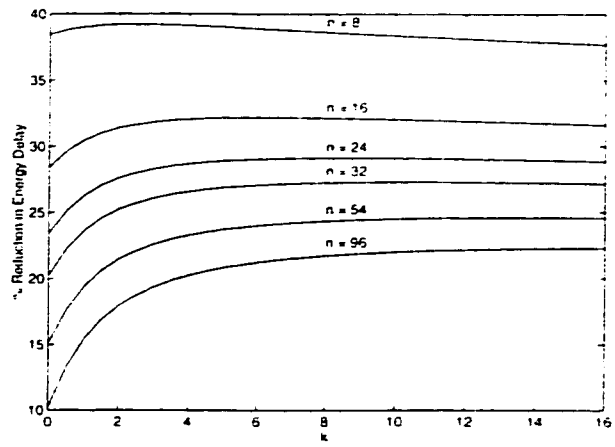


Figure 6.7 - % Reduction in ED (MBA II)

ferent from that theoretically predicted. MBA schemes are vulnerable to power losses due to spurious transitions - owing to the dissimilar delays of control signals and multiplicand bits, by virtue of which the power consumption of these schemes is greater than that theoretically anticipated. Another subtle difference from theoretical analysis occurs as far as signal transmission delays are concerned. The CPL cells used for this analysis exhibited delays greater than 1.3τ because of which the delay of MBA I scheme had been appreciably greater than that of the 4:2 compressor based scheme while the delay of MBA II had been found comparable. Because of these issues, the energy delay and power advantages of 4:2 compressor based scheme over MBA I scheme is better than that theoretically anticipated while these figures are less than that theoretically anticipated in comparison with MBA II.

In contrast to MBA schemes, the 4:2 compressor based partial product generation scheme offers the best results as far as circuit delays, power consumption and energy delay are concerned. Partial product reduction using MBA I uses six control signals for the genera-

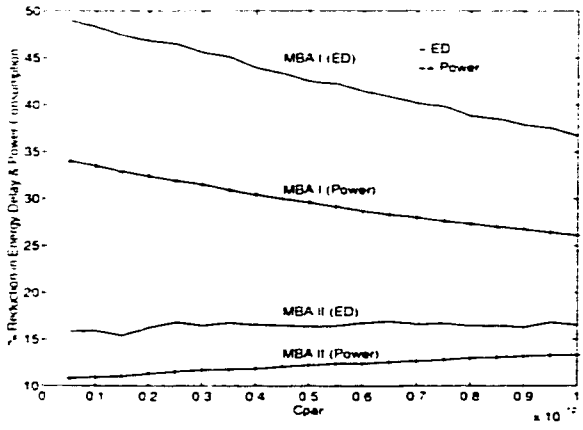


Figure 6.8 - % Reduction in ED and power

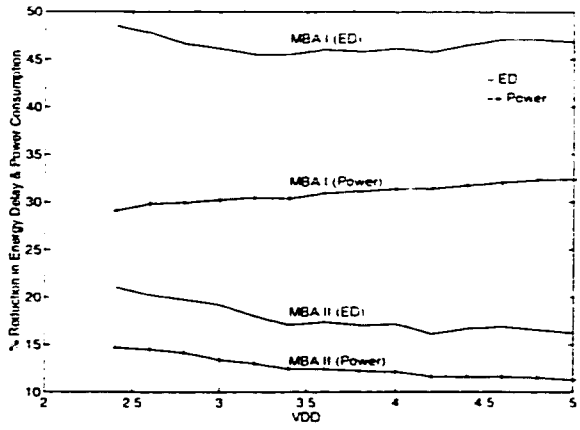


Figure 6.9 - % Reduction in ED and power

tion of two partial products while MBA II uses ten control signals. The effect of parasitic loading on these lines is much greater than that on the four multiplier bit lines of the 4:2 compressor based scheme and hence this scheme is extremely suitable for the implementation of wide multipliers. With MBA, an extra partial product is generated for integer multiplication when n is even. This, together with the sign extension correction bit and 2's complementation bit, effectively increases the number of partial products. Post processing of the extra partial products comes with added power as well as delay penalties. For design synthesis applications, the layout generation can be much faster for the 4:2 based schemes because of better structural regularity. In addition to the power and delay advantages, the computation of the sticky bit for IEEE floating point multipliers turns out to be much simpler with the 4:2 compressor based partial product generation scheme, because of the absence of negative partial products.

6.3. Barrel shifters

Addition of floating point numbers essentially requires the alignment of significands in

accordance with the difference between the exponents. In general, owing to the limited width of significand data fields, shifts beyond the width of significand are not necessary. If the assertion status of the nodes of the barrel switches are preserved during 'no valid shift' conditions, the resulting savings in dynamic power consumption can be substantial. The following paragraphs explain the architectural design and energy delay evaluation of a transition activity scaled barrel switch.

6.3.1 Barrel switch architectures

The shifting of binary data through a number of bit positions can be implemented in different ways [117] [118] [119] [120]. In general, a cascaded array of multiplexors can perform the requisite amount of data shifts. Figure 6.10 presents the block diagram of a barrel switch (BSI) which performs data alignment operations for the addition of IEEE single precision floating point data. In Figure 6.10, alignment shifting is accomplished by routing the data through a cascaded array of 2X1 and 4X1 MUXs. Our investigations[1] suggest that this type of a shifter architecture is advantageous as far as energy delay minimization of barrel switches is concerned. The data selection MUXs at the input of the shifter array present the significand of the smaller number for shifting. This block also performs the additional operation of routing the larger number as well as the sign of the smaller number to the output. In Figure 6.10, A and B represent the exponents while

1. A single stage of nX1 MUXs can also perform the requisite shifts. Though such a scheme appears attractive owing to the reduction in the number of cascaded stages, the energy delay implications of such a scheme can be worse than that of multi stage shifters. Referring to the switch level diagram of a shifter MUX given in Figure 6.14, it can be seen that the switching of the data select transistors constitutes a charge sharing problem between the input nodes and the input of the level restoration inverter. With a large number of data select transistors, the delay accumulation due to this effect can be severe. The operational power demand of such schemes can be worse than that of multi stage shifters owing to the effects of parasitic capacitances due to wiring complexity. The power/delay degradation of such schemes had been highlighted by other researchers also [119].

NUM1 and *NUM2* represent the input floating point numbers. The shift control signals shown in Figure 6.10 are derived through the decoding of the bits of the exponent difference, $|A - B|$. For values of $|A - B|$ exceeding the shift range, the aligned significand must be set to zero. The 'BSR' signal controls this operation. This barrel switch scheme is vulnerable to power losses due to spurious transitions in the control/operand data paths due to the absence of delay balancing schemes. Figure 6.11 presents a delay balanced architecture (BSII). The latches at the input of the shifter array provide the required delay balancing.

As stated earlier, the operational power demand of barrel switches can be reduced by inhibiting power consuming transitions within the shifter array during 'no shift' conditions. The scheme shown in Figure 6.11 can be modified to this effect. Figure 6.12 presents the control/data flow scheme of the proposed barrel switch (BSIII). The timing and control unit performs the following operations. (1) Computes the exponent difference, which eventually determines the magnitude of shifting required. (2) Decodes the exponent difference for controlling the shifter MUXs. (3) Evaluates 'no shift' conditions, viz. equality of exponents, exponent difference greater than shifting range and zero operands. (4) Evaluates the relative magnitudes of exponents. (5) Supplies delay balanced gated clocks for latching the inputs of the shifter array. (6) Generates control signals for effecting output data selection. The input data select block effects significand selection for shifting while the output data select block facilitates presentation of the aligned floating point numbers to the output. Apart from performing alignment shifts, the architecture also supports the evaluation of ceratin status signals that can be of interest in succeeding stages. A

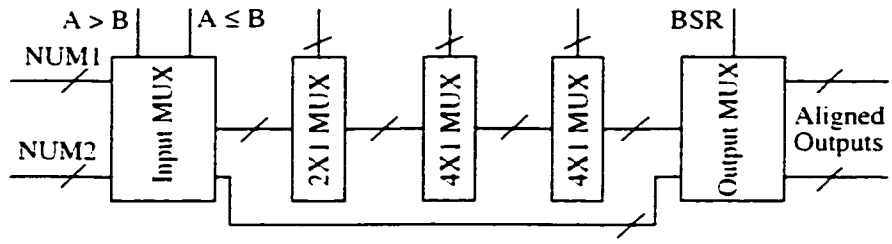


Figure 6.10 - Barrel switch - scheme I (BSI)

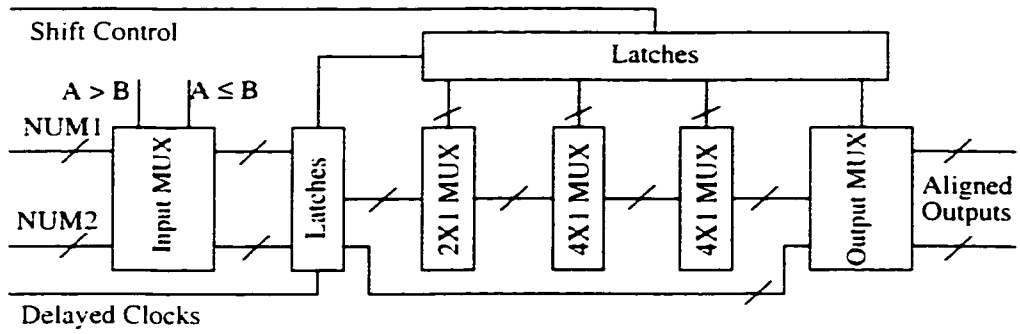


Figure 6.11 - Barrel switch - scheme II (BSII)

condition identical to 'no shift' exists for floating point adders as well, for a certain class of input operands such that the process of addition is not required. The pre computation of 'no add' conditions can be concurrently performed during the process of generation of shift control signals, by making use of ceratin signals that are mandatory for the operation of the shifter. This signal can be useful for effecting power management/control of the significand adder. Apart from this signal, presentation of status signals like zero/infinity operands, 'NaN' (not a number) etc. can also be envisaged.

As stated earlier, the data latches at the input of the shifter array are enabled by properly delayed gated clocks. These clocks are muted during those situations when 'no shift' conditions exist. This type of a control ensures the preservation of the assertion status of the nodes of the shifter array during 'no shift' conditions as well as delay balancing by virtue

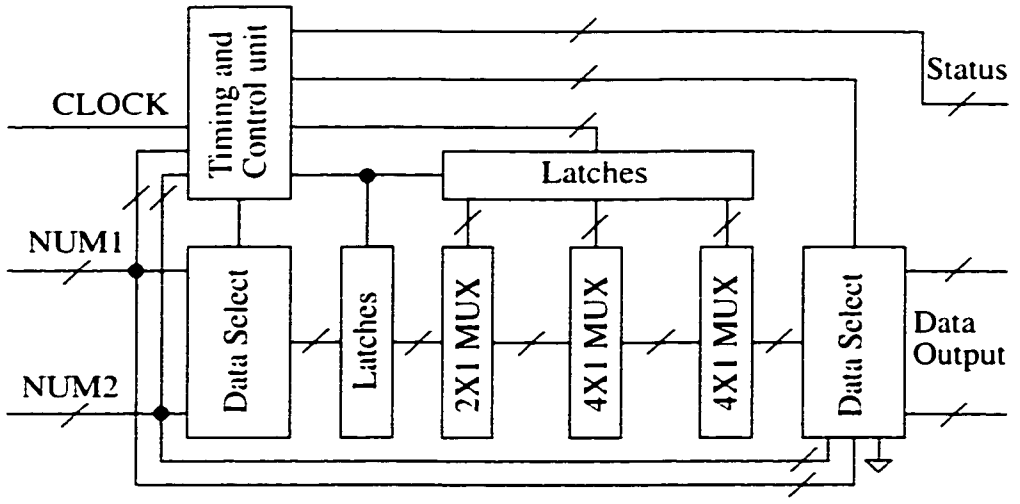


Figure 6.12 - Control/data flow scheme of barrel switch (BSIII)

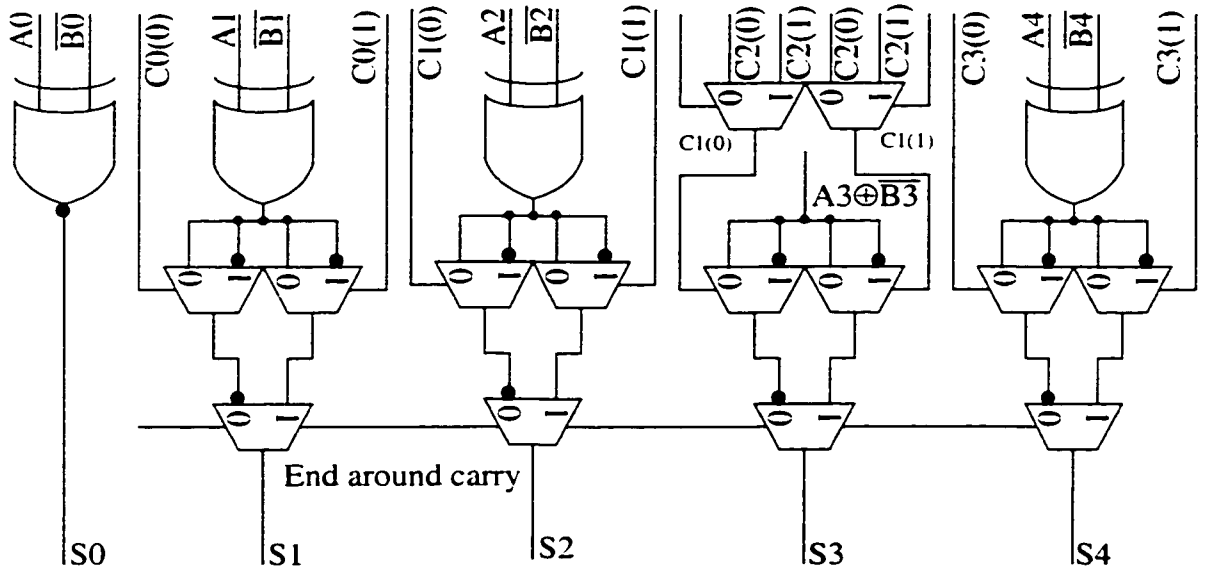


Figure 6.13 - Generation of sum bits for |A - B| computation

of which the operational power demand of the proposed scheme (BSIII) can be significantly lower than that of conventional schemes.

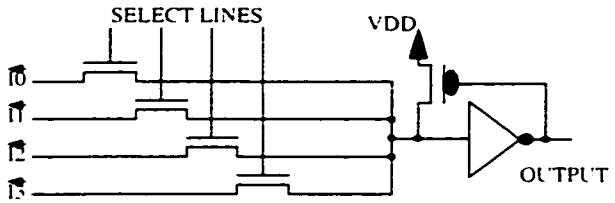


Figure 6.14 - 4X1 multiplexor

6.3.2 Circuit models

As discussed earlier, in floating point additions, the magnitude of significand alignment shift is decided by the difference between the exponents whereas the selection of an appropriate significand for performing the requisite amount of shift is decided by the relative magnitudes of the exponents. As discussed earlier, 1's complement adders are ideal for this operation. (The conditional carry evaluation scheme of an 8 bit 1's complement adder is presented in Figure 5.2). For single precision FADDs, 5 bits are sufficient for effecting significand shifts. Figure 6.13 illustrates a scheme that evaluates 5 of the LSB bits of the exponent difference $|A - B|$. Evaluation of 'no shift' condition can be realized through a logical OR operation of the conditions $A = B$, $A = 0$, $B = 0$ and $|A - B| >$ shift range. The 'no shift' condition can be used for muting the clocks which enable the latches at the input of the shifter array. Pass gate logic structures are ideal for the implementation of shifting as well as data selection MUXs. Figure 6.14 illustrates the circuit diagram of a 4X1 MUX using NMOS pass transistors for data selection.

6.3.3 Energy delay analysis

The following equations give the activity driven (which is analogous to dynamic power consumption) energy measures of various barrel switch schemes. These equations had

been derived through signal probability analysis as well as fanout considerations of the relevant schemes.

$$EM_I = 21.662 + 3.918n + 0.375N + \left[\frac{1}{\eta_1}(1.379n + 0.1788) + \frac{0.15N}{\eta_2}\right](1 + k) \quad \textbf{(6.6)}$$

$$EM_{II} = 35.662 + 1.156n + 0.25N + \left[\frac{1}{\eta_1}(1.439n + 0.089) + \frac{0.15N}{\eta_2}\right](1 + k) \quad \textbf{(6.7)}$$

$$EM_{III} = 63.231 + 0.333n + 0.25N + \left[\frac{1}{\eta_1}(0.3548n + 0.089) + \frac{0.15N}{\eta_2}\right](1 + k) \quad \textbf{(6.8)}$$

where EM_I, EM_{II} and EM_{III} represent the energy measures of BSI, BSII and BSIII (Inhibit controlled or transition activity scaled barrel switch) respectively. In the above equations, k represents coefficient of parasitic loading, η_1, η_2 represent the efficiencies of drivers whose fanouts are of the order n and N respectively, N represents the width of the floating point number (including leading 1 and guard bits) and n represents the width of the significand. The validity of these models are restricted to the specific case of IEEE single precision floating point operands. The energy measures given by the above equations are specific for implementations using MUXs of the type shown in Figure 6.14. For these MUXs (with device widths boosted to double the minimum sizes), the capacitances seen by the select lines are approximately 0.3 times the capacitances seen by the signal inputs, considering device models of 0.5 micron processes. For other types of MUXs, the above equations can be modified by appropriately scaling the terms multiplied by $(1 + k)$.

The delay of BSI is 9τ plus driver delay while that of BSII and BSIII are respectively 10τ plus driver delay and 12τ plus driver delay, where τ represents the delay of a 2 input gate

(a worst case estimate of which is 1.5 times the delay of a minimum sized inverter). A buffer delay of approximately 9τ can be anticipated from a typical three stage driver, having a stage ratio of around 4.4, optimally designed to drive the gate loads of around 26 MUXs of the type shown in Figure 6.14 - considering a coefficient of parasitic loading of 10. The delay of latches is assumed to be equal to that of a 2 input gate.

6.3.4 Results

Figure 6.15 gives a plot of the percentage reduction in power consumption of the proposed barrel switch scheme, for various instances of parasitic loading. The dashed curve represents the percentage reduction in power consumption of BSIII with respect to that of BSII, while the solid line curve represents such a measure in comparison with BSI. The reduction in power consumption of the proposed scheme is better than 55% for a coefficient of parasitic loading of around 10^1. Though the BSII scheme incorporates activity reduction through delay balancing, the power consumption of the delay balancing scheme offsets the power saving through activity reduction. For lower values of k, the power consumption of BSII is less than that of BSI.

Figure 6.16 gives the comparative energy delay reduction of BSIII against that of BSI and BSII. Here again, the dashed curve gives the relative reduction in comparison to BSII while the other curve gives that with respect to BSI. The reduction in energy delay is better than 50% for those values of k that are greater than 10.

---

1. The following example highlights the significance of the value of k. In our energy delay analysis, parasitic loading is restricted to affect only the high fanout nodes, which are essentially excited by drivers. The select lines of shifter MUXs are good examples. For the type of MUXs shown in Figure 6.6, with all device widths boosted to double the minimum required, the parasitic capacitance is around 1 pF for a k of around 13, for the CMC 0.5 micron process.

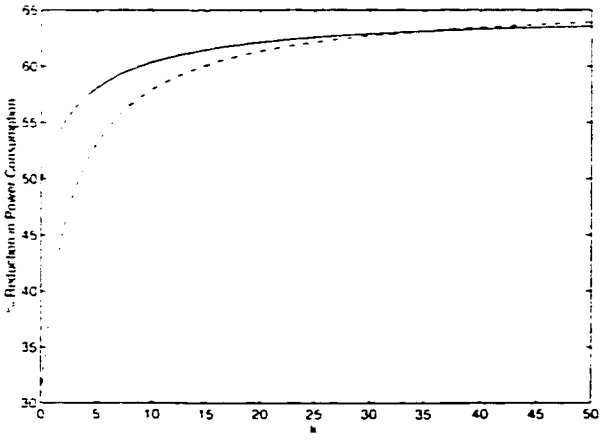


Figure 6.15 - % Reduction in operating power

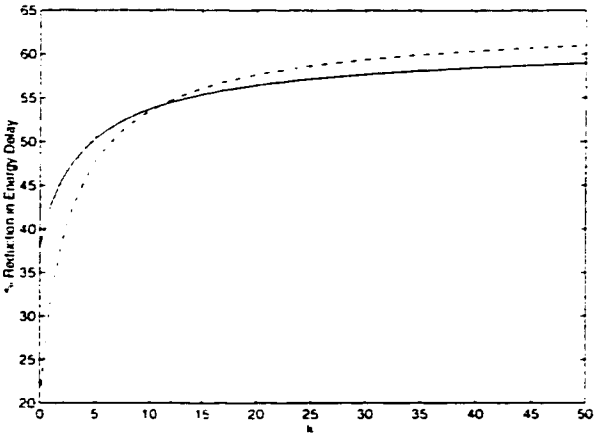


Figure 6.16 - % Reduction in energy delay

In order to highlight the effect of transition activity scaling as far as the power implications of significand pre-alignment is concerned, the power reduction observed during filtering of bipolar audio signals is presented in Figure 6.17 [112]. (The experimental descriptions as well as power models are presented in Chapter 7). During IIR filtering, the reduction in switching activity is around 30% while the probability for barrel shifter bypass is around 5%. For FIR filters, the minimum reduction offered is better than 27.64% while the best is as high as 58%. Figure 6.18 illustrates the barrel shifter model used in our simulations.

In barrel shifters, the dominant component of power consumption is attributed to the switching activities of high fanout nodes and operand data path nodes. Transition activity scaling of these nodes is rewarding as far as design for low power operation is concerned. The delays of drivers which interface signals to the high fanout nodes, forms a major component of the overall delay. Because of these reasons, the additional power/delay overheads attributed to the evaluation of extra control signals for effecting transition activity

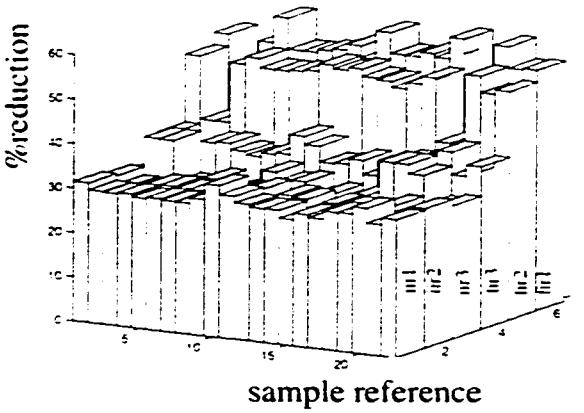


Figure 6.17 - Power reduction during pre -alignment

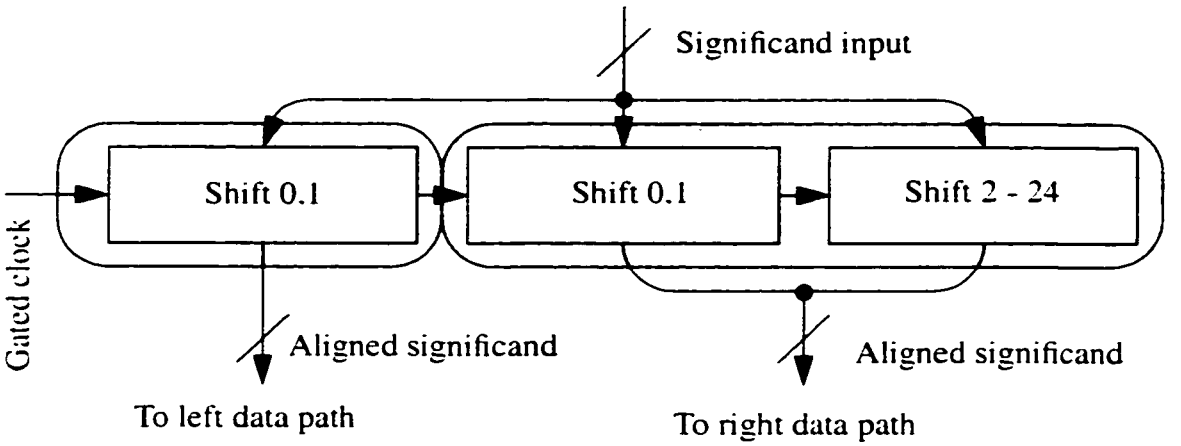


Figure 6.18 - Transition activity scaled pre-alignment barrel switch

scaling is insignificant in contrast to the savings in power wastage attainable through such a control.

6.4 Conclusion

The power/delay implications of partial product generation schemes and barrel switcher organization are presented. The energy delay advantages of these approaches render them attractive for full custom or design synthesis implementations.

Chapter 7

Low Power Transition Activity Scaled Triple Data Path Floating Point Adder

7.1. Introduction

High performance floating point adders are essential building blocks of microprocessors and floating point DSP data paths. Since the hardware implementation of floating point addition (FP 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 [8] [97], the power consumption of floating point adders are, in general, quite significant in comparison to that of their integer counterparts. Owing to the presence of a relatively high traffic of floating point additions in microprocessors [25] [26] [27] and DSPs, the power/performance implications of floating point adders directly impact the power/performance desirability of the target application. In digital CMOS implementations, the power consumption and speed performance of functional units are, to a large extent, susceptible to algorithmic/architectural design decisions [2]. These decisions influence the switching activities, fanouts, layout complexity, logic depths, operand scalability and pipelinability of functional units. Among the above, switching activities, fanouts and layout complexity are directly related to the power consumption. The higher the values of these parameters, the higher the power consumption. Architectural design for low power operation targets the minimization of these parameters. Traditionally, the architecture of floating point adders had been centered around the sequential machine philosophy. With

an ever escalating demand for high performance floating point units, newer design approaches are emerging [3] [22] [23] [25] [26] [27]. With new designs, throughput acceleration is achieved through operational parallelism. In [25] [26] and [27] Oberman et al. proposed a novel architecture that incorporates concurrent, speculative computation of three possible results of an FP addition. These results are typified by the time complexities of their operations: the hardware realization of ceratin cases of floating point addition can be simplified so that the execution time of such operations are reduced. With this, the variable latency architecture reported in [25] [26] and [27] produces results within 1, 2 or 3 cycles, by virtue of which the average latency of FP additions is reduced. The fusion of multiply and accumulate operations in floating point multiply-accumulate units [3], and the concurrent evaluation of leading zeros with significand addition (leading zero anticipation) [3] [22] [24] are other notable examples that exploit parallelism for latency reduction. The operation of rounding is an integral part of floating point addition. While MAC fusion and leading zero anticipatory logic provide throughput acceleration, speculative computing for rounding [121] also enhances the throughput of floating point units. Though the concept of operational parallelism can be put to advantageous use as far as throughput acceleration is concerned, and the trading of speed performance for power reduction is also not infeasible, this approach, however cannot guarantee power reduction in performance critical applications. Coming back to architectures and algorithms, architectural design for low power operation targets the minimization of power consuming transitions through the use of appropriate number systems, algorithms, structural transformations etc. Transition activity scaling of functional units [61] offers promising results as far as architectural design for low power operation is concerned. This chapter

addresses the architectural design of a triple data path floating point adder (FADD) which incorporates a pseudo leading zero anticipatory logic as well as speculative rounding for throughput acceleration and transition activity scaling for power reduction. An abridged version of this work had been presented in [98]. While data path replications in conjunction with concurrent evaluation of different possible results for throughput enhancement of floating point adders is addressed by Oberman et al. [25] [26] [27], our work is different. The salient features of our work are: (1) The data paths are activity scaled and (2) The latency is fixed.

The rest of the chapter is organized in the following order. Section 2 presents a brief overview of floating point addition, hardware organization of floating point adders, performance acceleration techniques etc. Section 3 gives an overview of the architecture of the proposed triple data path floating point adder (TDPFADD). Section 4 presents the salient features of the zero overhead rounding scheme. Section 5 highlights certain aspects (that are relevant to low power designs) of accumulator design using TDPFADD. Section 6 presents the development of power models. The architectural area implications of TDPFADD in contrast to that of other schemes is presented in section 7. Section 8 presents the results. Section 9 presents a discussion highlighting the power/delay implications of the proposed FADD, limitations, worst case scenarios etc. Section 10 concludes the chapter.

7.2. Floating point addition

Addition of floating point numbers involves the alignment, addition, normalization and rounding of significands as well as exponent evaluation. Significand pre-addition-alignment is a pre-requisite for addition (here onwards, we will use the simpler term pre-align-

ment for significand pre-addition alignments). 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 (which may necessitate an exponent correction as well) 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 carry 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 7.1 illustrates a possible 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 shifter. The barrel shifter performs the requisite pre-alignment shift of the significand of the smaller floating point number. The sign magnitude adder performs significand addition/subtraction, the result of which is again represented in sign-magnitude form.

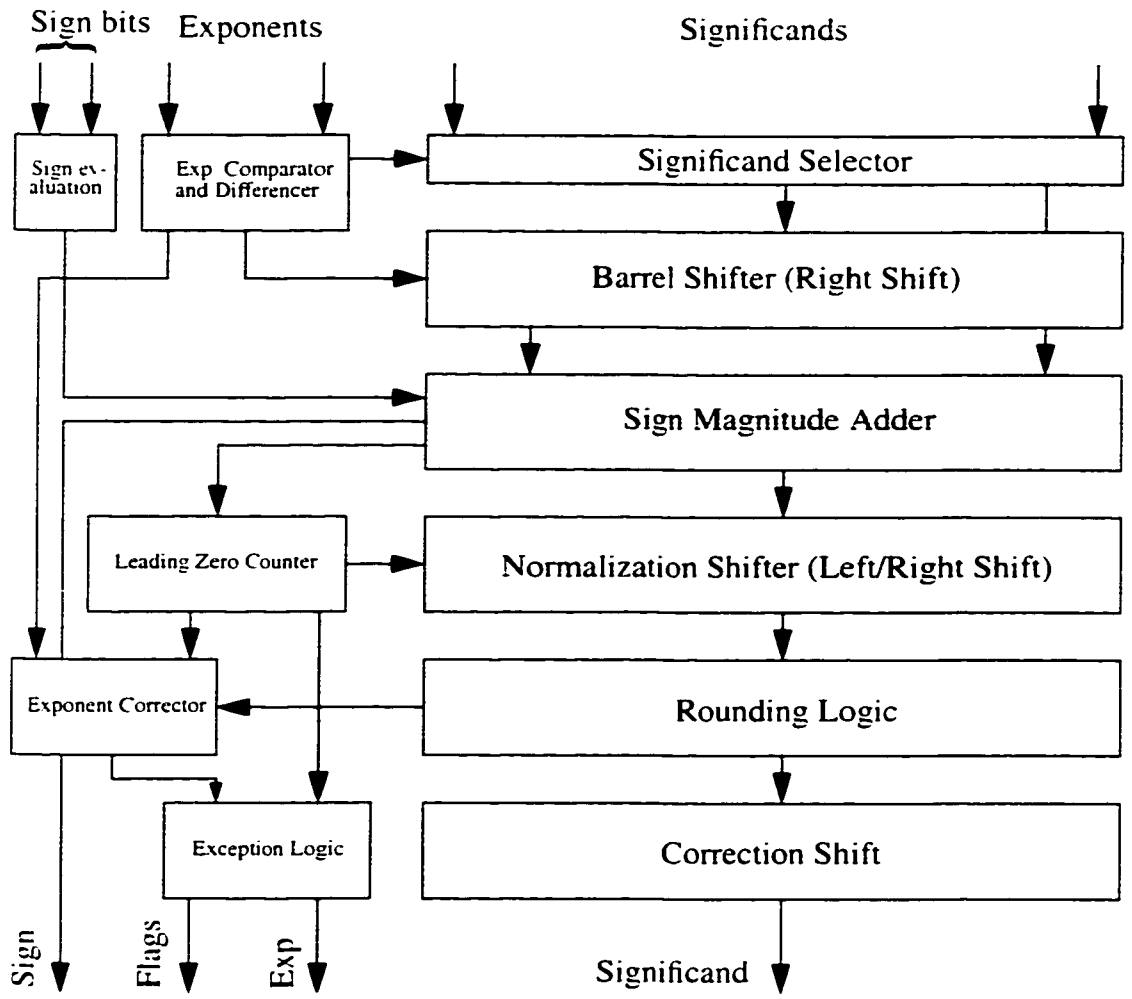


Figure 7.1 - Block diagram of floating point adder

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 exception logic asserts various status flags subject to the validity of exception

conditions.

In [3] [22] [24] [25] [26] and [27], the operations of significand addition and leading zero estimation are concurrently performed, so that the latency is reduced. The resulting performance acceleration comes with power and area penalties. The leading zero anticipatory logic is more complex than a straight forward leading zero counter. Because of this, the power consumption and silicon area of implementations that use leading zero anticipatory logic can be around twice that of schemes that use simpler leading zero counters, as far as significand addition and leading zero estimation are concerned.

7.3. Triple data path FADD architecture

Table 7.1 lists the FP addition algorithm that is mapped into the proposed transition activity scaled triple data path FADD architecture. Figure 7.2 illustrates the control/data flow architecture of TDPFADD. In this scheme, the process of floating point addition is partitioned into three distinct categories and a separate activity scaled data path is envisaged for the handling of each. Among the three distinct data paths, two are computing data paths while the third is a non computing data path. The non computing or bypass data path becomes operational during those situations when the process of floating point addition is guaranteed to produce a result that is known apriori. The criterion for operational partitioning of the computing data paths is the possibility for the generation of a variable number of leading zeros during the signed-magnitude addition of significands. A variable number of leading zeros can be generated only under two circumstances - viz. subtraction of one significand from another when the difference between their exponents is zero or one. Significand pre-alignment shifts for this case are upper bounded by one. For other

Table 7.1: Floating point addition algorithm

Step 1

compare exponents
compute exponent difference |e1 - e2|
evaluate special conditions 0 ± operand, ±∞ ± operand, NaN ± operand
select the tentative exponent of the result
order significands on the basis of the relative magnitudes of exponents
if |e1 - e2| > p or special conditions, generate default result and go to step 2
if |e1 - e2| ≤ 1 and subtraction go to step 3.1
if |e1 - e2| > 1 or addition and neither |e1 - e2| > p nor special conditions go to step 4.1

Step 2
present default result
go to step 5

Step 3. 1
perform 1's complement addition of aligned significands
perform speculative rounding
count leading zeros of the different copies of results
select result and pertinent leading zero count
evaluate exception conditions, if any
go to step 3.2

Step 3.2
normalize significand
compute the exponent of the result
go to step 5

Step 4.1
align the significand

Step 4.2

perform signed - magnitude addition of aligned significands
perform speculative rounding
evaluate normalization requirements: 0 left or right shift
select result and perform normalization
select exponent
evaluate exception conditions, if any
go to step 5

Step 5

select the appropriate copy of result from the relevant data path

cases, signed-magnitude addition of significands can produce at the most one leading zero. Pre-alignment shifts, however, can be between 0 and p bit positions (p represents the width

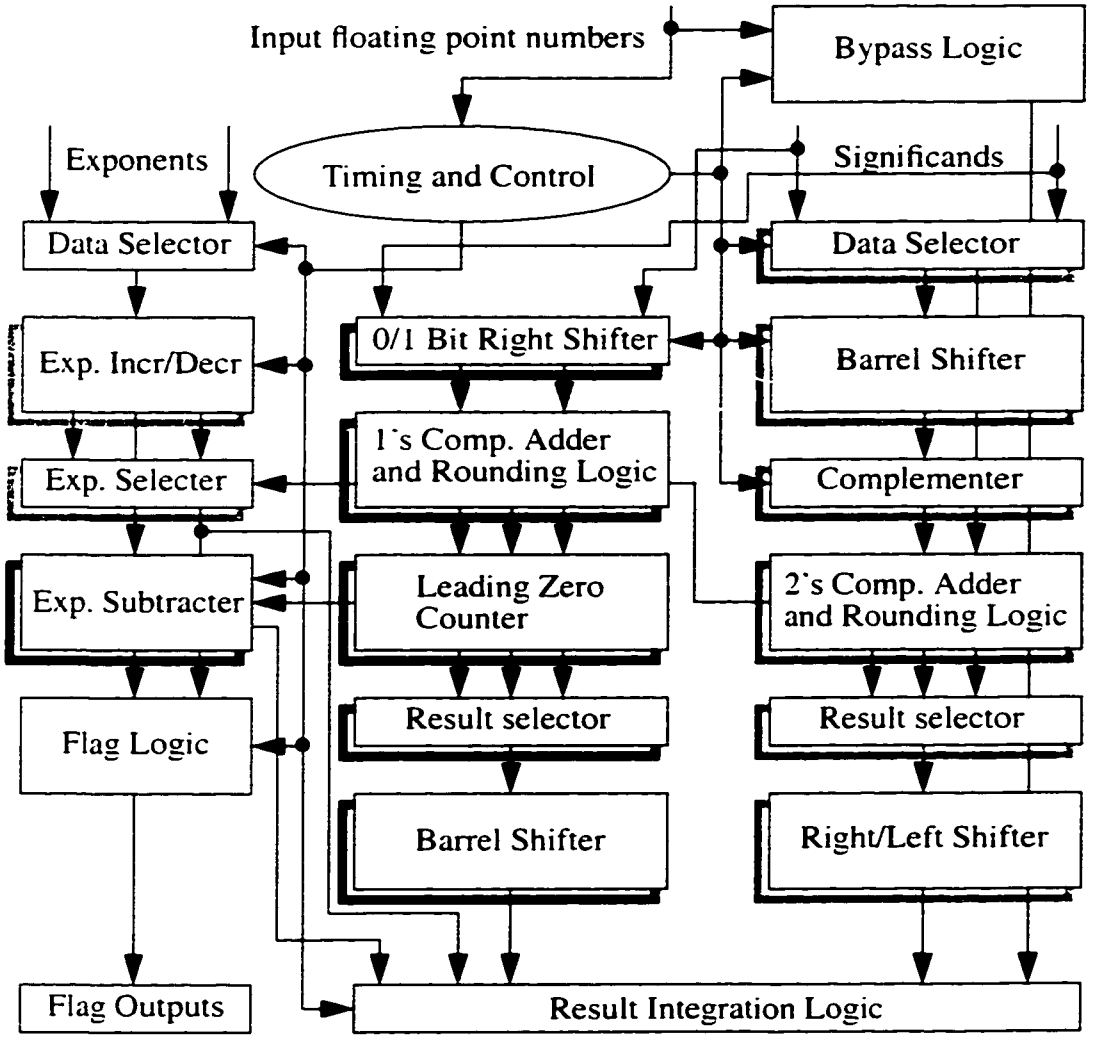


Figure 7.2 - Control/data flow architecture of triple data path floating point adder

of significand data field, including the hidden bit) for such cases.

In addition to the power savings through transition activity scaling, the proposed architecture also offers power savings due to the simplification of data paths. Whenever the significand of a floating point number needs pre-alignment shifts, a barrel shifter is mandatory for performing such operations. In general, shifting of binary data through a variable num-

ber of bit positions demands complex hardware [117] [118] [119] [120] [123], the operational power demand and speed performance limitation of which units are expensive as far as hardware implementation of floating point adders is concerned. Since the proposed architectural partitioning envisages a separate data path - LZA data path which mimics significand addition and leading zero estimation of conventional high speed FADDs - for the handling of arithmetic operations that probably generate a variable number of leading zeros, this data path, however doesn't need a composite barrel shifter for significand pre-alignments; the required alignment shifts can be performed by using a single level of multiplexors (MUXs). Though pre-alignment shifts are ≤ 1 for this data path, the normalization shift requirement can be anywhere between 0 and p, by virtue of which it is mandatory to have a composite barrel shifter for performing such shifts. The leading zero bounded data path (LZB data path - an arithmetic operation in this data path can produce at the most one leading zero), needs a pre-alignment barrel shifter for significand alignments. The normalization shifter for this data path can be realized by using a single level of 3X1 MUXs. In Figure 7.2, the two distinct data paths are characterized by the shading of the blocks. Functional sub-units of the LZA data path are shaded darker than that of the LZB data path. The architecture of the relevant data paths is explained below.

For the LZA data path, the 0/1 bit right shifter block handles the pre-alignment of the input significands. This block also performs the complementation of the appropriate significand. The 1's complement adder performs signed magnitude addition of significands. Pre-computation for rounding is performed concurrently with addition, so that by the time the rounding decisions are known, an appropriate copy of the result can be selected for normalization shift. The leading zero counting logic detects and encodes the number of lead-

ing zeros into appropriate binary words for each of the different copies of the results generated by the 1's complement adder/rounding logic. The proposed FADD scheme employs a 'pseudo leading zero anticipatory logic' approach for the estimation of leading zeros. Since the significand pre-alignment operation is rather fast for the LZA data path (due the absence of a multi-stage barrel shifter), the adder inputs of this data path get asserted at a relatively earlier time compared to that of the LZB data path. The earlier the arrival of significand inputs, the earlier the completion of addition. Because of this reason, a full parallel leading zero anticipatory logic is not mandatory for this data path. Even with a simpler (serial) leading zero counting logic, this data path can complete the process of floating point addition within such time the LZB data path completes an addition. Effectively, this type of a scheme offers a speed performance that is comparable to that of schemes with parallel leading zero anticipatory logic while the complexity/area/power measures are appreciably less. The result selection logic selects an appropriate copy of the result in accordance with the rounding/complementation conditions. This block also selects an appropriate copy of the leading zero count. The output (significand) of this block is shifted by the normalization barrel shifter through an appropriate number of bit positions that is equal to the selected value of leading zero count. The exponent subtracter subtracts the relevant copy of leading zero count from the tentative exponent of the result. The evaluation of underflow and zero result conditions of this data path are discussed in section 5.

For the LZB data path, the input data selector selects significands for pre-alignment operation. The magnitude of pre-alignment shift is encoded by the control unit. The output of the pre-alignment barrel shifter is routed to the significand adder through a complementer.

The significand of the larger number is routed by the input data selector directly to the adder without any complementation. The complementer block performs a conditional inversion (bitwise) of its input data word. Complementation is performed only during subtraction of one floating point number from another. The 2's complement adder performs signed magnitude addition of significands, the results of which operation are guaranteed to be positive. Pre-computation for rounding is concurrently performed with addition. The result selector block selects an appropriate copy of the result in accordance with the rounding/normalization requirements. The normalization shifter for this data path is a simple left/right shifter. Exponent computations of this data path are realized by using exponent increment/decrement logic, which implement a conditional increment (subject to the generation of a carry out from the MSB position of the significand adder during addition) or conditional decrement (subject to the generation of a leading zero during subtraction) of the tentative exponent. Pre-computation of exponent increment/decrement is desirable for speed performance. Once the process of significand addition is complete and the decisions regarding exponent correction are available, the desired copy of the precomputed exponent can be selected. Evaluation of overflow/underflow conditions is rather straight forward for this data path. During subtraction, an underflow condition can arise if and only if the tentative exponent is the smallest representable exponent (defined by the number system) and the result of the signed magnitude significand addition contains a leading zero. Similarly, overflow can occur if the tentative exponent is the maximum representable exponent without overflow, and the significand addition produces a carry out from the MSB end of the significand adder.

The bypass data path essentially selects and memorizes an appropriate floating point number during various bypass conditions. Whenever the difference between the exponents of the numbers is greater than the width of significand, the larger floating point number is latched into the bypass

data path. For operations of the type: 0 ± number also, the larger floating point number is latched. If both the operands are zeros, a zero is latched. During operations of the type ±∞ ± number, an infinity is latched. During situations when an FP addition produces an 'NaN' (not a number) result, an NaN is latched. In pipelined architectures, the results of the bypass data path are not immediately presented to the output. The latched data is presented to the output during an appropriate cycle of pipeline operation.

The timing and control unit, flag logic and result integration logic are common to all the data paths. The timing and control unit evaluates the inputs for various conditions, selects an appropriate data path, routes the inputs to the selected data path, asserts flags and integrates the data from the relevant data paths to the final output. This unit generates the various gated clocks that control the data presentation to/from the various data paths.

7.4. Zero overhead rounding

The design for zero overhead rounding involves the design of two distinct rounding schemes - one for each of the significand addition cases envisaged by the partitioning criterion presented above. Figure 7.3 illustrates the block diagram of the rounding and normalization scheme for the LZA data path. Figure 7.4 (a) illustrates the pre-aligned significands before addition while Figure 7.4 (b) illustrates the result. The data field with a *G* at its LSB position represent the 1's complement of the right shifted significand when the exponent difference between the floating point numbers is 1. When the exponent difference is zero, this data field still represents the 1's complement of one of the significands (in which case *G* will be a 1). The LSB of the shifted out significand is always retained as a guard bit. The bit *a0* is asserted 1 during the subtraction of a right shifted significand.

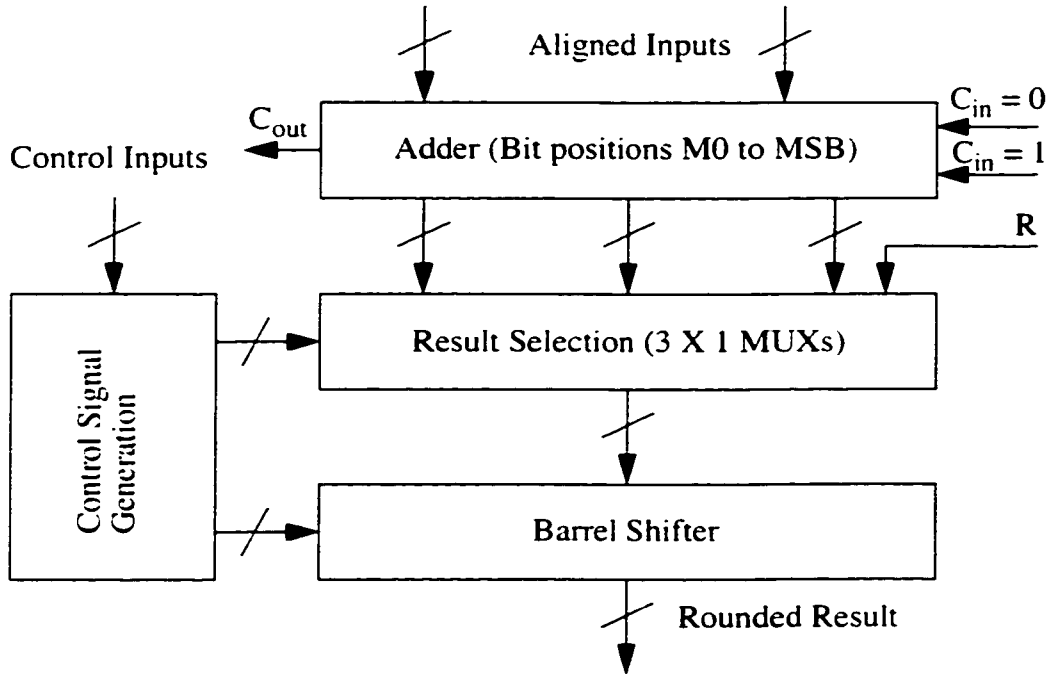


Figure 7.3 - Pre-computation, selection and normalization of rounded results - LZA data path

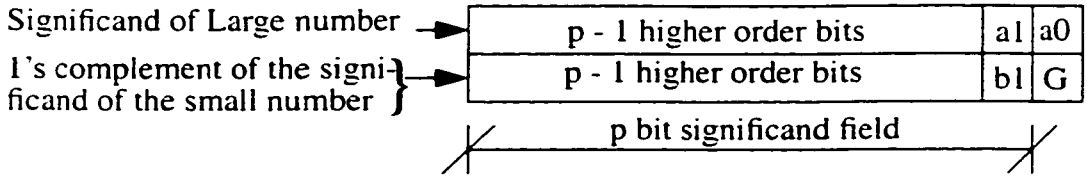


Figure 7.4 (a) - Pre - aligned significands

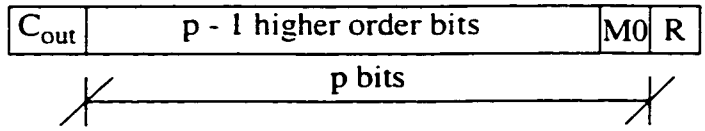


Figure 7.4 (b) - Result of significand addition before normalization shift

Effectively, this type of an operation guarantees a 2's compiement addition of the aligned significands whenever the exponent difference is one, in which case a positive result is guaranteed. Whenever the exponent difference is zero, the *a0*, *G* bits are not relevant in the signed-magnitude addition. The adder block evaluates conditional sum bits for bit posi-

tions M0 to MSB, anticipating block carry inputs of 1 as well as zero. The result selection block accepts the conditional sum bits with anticipated carry inputs of one and zero as well as the 1's complement of the conditional sum bits with an anticipated carry-in of zero. The 1's complement of the conditional sum bits with an anticipated carry-in of zero are required during those situations when the end around carry of 1's complement addition of significands with equal exponents is a zero, indicating a negative result. The barrel shifter shown in Figure 7.3 performs normalization shifts on the selected result such that the leading bit of the final result is always a 1. The control unit evaluates the result selection bits as well as the shift control data word.

Figure 7.5 gives the block diagram of the significand normalization/rounding scheme for the LZB data path. The input/output data representation for this case is shown in Figure 7.6. The *a0* bit will be a 0 during addition and will be a 1 during subtraction (for 2's complementation of the aligned significand). The *G*, *R*, *S* (guard, round and sticky bits [8] - these bits are essential to implement 'round to nearest/even') and *bi* ($\forall i$) bits are the complements of the bits of the aligned significand during subtraction and are the true bits during addition. For rounding purposes, bit positions *M1* to C_{out} can be treated as a single block. In general, since the rounding decisions are not available till the normalization shift is complete, pre-computation of two sets of sum bits is the best choice for effecting zero overhead rounding - so that by the time the rounding decisions are known, an appropriate set of sum bits can be chosen. The sum bits can be evaluated for incoming carries of zero as well as one. Conditional sum/carry select adders are ideal for this application. As far as the bits towards the right of M1 are concerned, conditional results for all possible cases of addition and rounding need be pre-computed. The combination of result selection/and

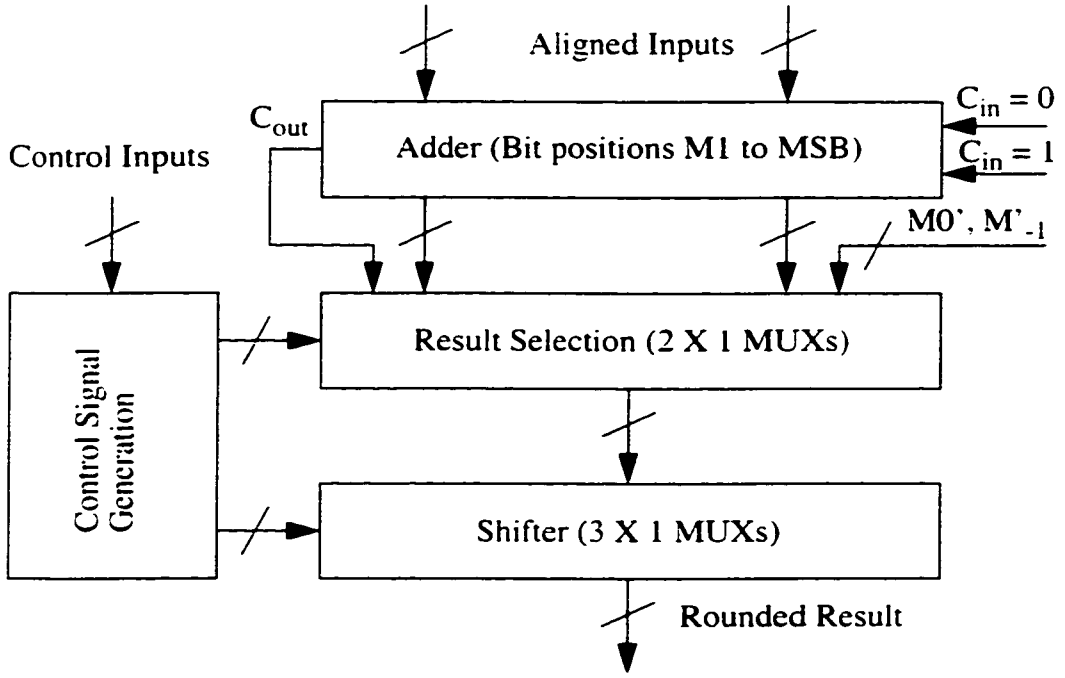


Figure 7.5 - Pre-computation, selection and normalization of rounded results - LZB data path

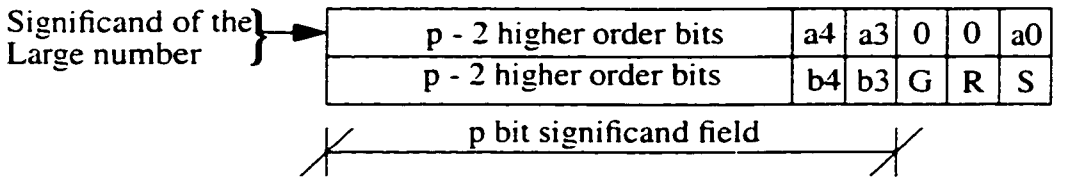


Figure 7.6 (a) - Pre - aligned significands

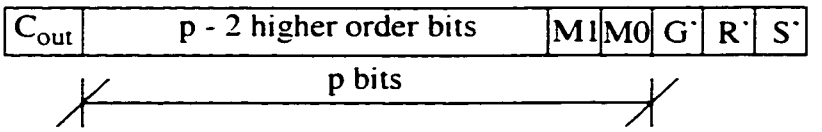


Figure 7.6 (b) - Result of significand addition before normalization shift

shifter blocks append the rounded bits of the lower order bit positions (towards the right of M1) to that of the higher order bit positions (M1 to MSB). The control unit in Figure 7.5 generates the control signals for the selection of the appropriate copy of the pre-computed sum bits and the shift control signals for effecting normalization shift. Sum selection and

normalization shifts are performed by multiplexors.

7.5. Accumulator design with TDPFADD

Considering the power/delay implications of floating point operations as far the implementation of an accumulator is concerned, certain organizational simplifications of the FADD core are advantageous. Essentially, the control unit evaluates various conditions for the excitation of the different data paths. Both the input operands need be evaluated for the detection of zero, infinity and NaN operands. When the FADD is configured as an accumulator, these conditions need be explicitly evaluated with only one of the operands. With the other operand, viz. the current sum, these conditions can be concurrently evaluated while the addition is still in progress. As discussed earlier, with the TDPFADD architecture, the evaluation of the sum for a zero condition is rather straight forward. The generation of a zero result is of special interest in FADD/accumulator design. If the result of a floating point operation is zero, the resulting significand need not be subjected to any normalization shift. That means, the normalization shifter of the LZA data path of the TDPFADD need not be operational during such situations. Transition activity scaling of this barrel shifter can also result in significant power savings. In our design, the normalization shifter of the LZA data path is not excited during those situations when the result of an addition is zero. Apart from this scheme for the handling of a zero result, the accumulator design also envisages the assertion of a zero condition during resets. This operation doesn't necessarily mean that a zero operand is latched into the registers of the data paths during resets. During the process of accumulate operation, whenever the current sum is zero, the input floating point operand is latched into the registers of the bypass data path. With multi stage pipelined organizations, resetting of a large number of pipeline registers

is expensive in terms of power. In our design, upon receipt of a reset signal, not all the registers are reset in the beginning. Instead, the accumulator operates in the bypass mode till the entire pipeline is filled with valid data. Upon completion of this phase, the process of accumulation begins.

Evaluation of infinity and NaN conditions of the current sum is trivial. In any accumulate operation, the current sum can assume a value of infinity or NaN if and only if the previous operation had been of the type $\pm\infty \pm$ number or an operation that leads to an NaN result. That means, the evaluation of infinity and NaN conditions need be done for only one operand, viz. the floating point input. If the current input qualifies to be an infinity or NaN, the result of the present operation will be an infinity or NaN.

7.6. Power measures

In order to validate the effect of transition activity scaling in power minimization, a comparative study of FADDs with and without transition activity scaling is envisaged. The steady state probabilities of data path utilization gives a global perspective of how effective transition activity scaling is. Apart from this global model, our analysis also highlights the significand alignment driven power behavior of FADDs during DSP operations. The following paragraphs present the development of models that reflect the architectural power implications of FADDs.

7.6.1 Data path utilization based power model

The power consumption of a floating point adder can be represented by

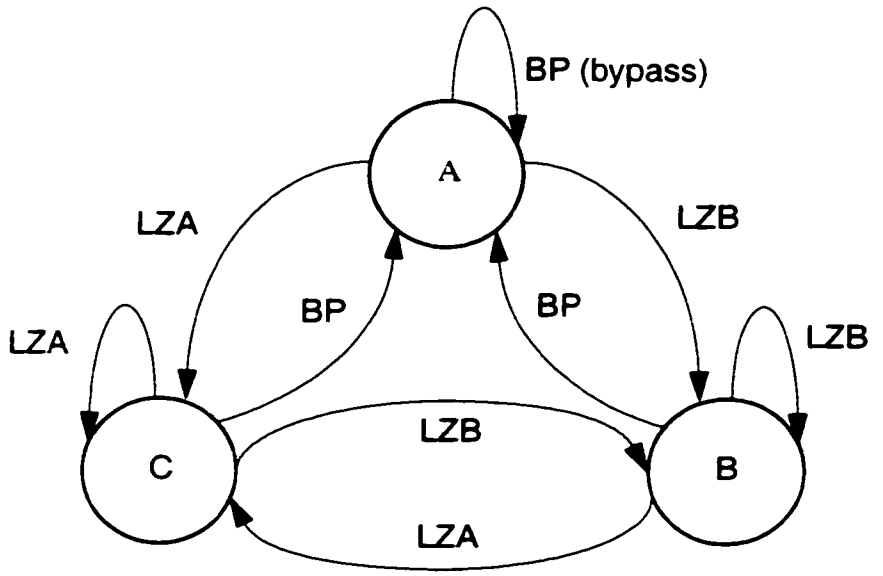


Figure 7.7 - Finite state machine representation of FADD operation

$$P = P_{ADD1} + P_{ADD2} + 2P_{BS} + P_{RND} + P_{LZ} + P_C \tag{7.1}$$

where P_{ADD1} and P_{ADD2} represent the power consumption of the significand and exponent adders respectively. P_{BS} represents the power consumption of barrel shifters while P_{RND} represents the power consumption of the rounding logic. P_{LZ} and P_C represent the power consumption of the leading zero logic and control unit respectively. For floating point adders using leading zero anticipatory logic, since the complexity of these circuits are comparable to that of significand adders, P_{LZ} is comparable to P_{ADD1}; the power consumption of such FADDs can be approximated by

$$P_I = 2P_{ADD1} + P_{ADD2} + 2P_{BS} + P_{RND} + P_C \tag{7.2}$$

Figure 7.7 illustrates a finite state machine representation of the operation of our transition activity scaled FADD. State A represents bypass conditions. State B represents the operation of the FADD during those situations when the signed magnitude addition of significands can produce at the most one leading zero while state C represents the FADD operation that can result in the generation of a significand with a variable number of leading zeros. In TDPFADD architecture, rounding is merged with addition. During any computing cycle, only one barrel shifter is operational. This is true with significand adders also. The power consumption of TDPFADD, taking into account the effect of FADD bypass, can be represented by

$$P_{II} = \beta[P_{ADD1} + P_{ADD2} + P_{BS}] + \alpha P_C + P_{BP}(1-\beta) \tag{7.3}$$

where α represents a scaling factor that reflects the relative complexity measure of the control unit of the TDPFADD core, P_{BP} represents the power consumption of the bypass data path while β represents the activity scaling factor (which is given by $P(B) + P(C)$ where $P(B)$, $P(C)$ represent the probability that the FADD is operating in states B and C respectively). Comparing equations (7.2) and (7.3) under such assumptions that the power consumption of the control unit, bypass data path, exponent logic and rounding logic are negligible in comparison to the power consumption of barrel shifters and significand adders, the power consumption of TDPFADD can be related to equation (7.2) by the following.

$$P_{II} \approx \left(\frac{\beta}{2}\right) P_I \tag{7.4}$$

where $\beta \leq 1$.

7.6.2. Significand alignment based power models

In FP additions, the transition activities of barrel shifter control lines exhibit sensitivities towards the significand alignment behavior of FADDs. Apart from this, the switching activities within the significand data paths also exhibit such sensitivities. The following paragraphs highlight the development of analytical models, that capture the impact of significand alignments on the power consumption of FADDs. Before we go into the specifics of this aspect of power consumption, the following definitions shall be introduced.

Definition 1 - Expected shift: The expected shift of any data alignment operation is defined by

$$E_{SH} = E[x] = \sum_{k=0}^{k_{max}} kP(x = k) = \sum_{k=0}^{k_{max}} kP_k \tag{7.5}$$

In equation (7.5), x represents the shift distance in bits ($x \in \{0, 1, \ldots\ldots k_{max}\}$, which is not necessarily the exponent difference ($|e1 - e2|$). P_k represents the probability that the shift distance is k. With FADDs, in general, shift operations are controlled by a few LSB bits (5 and 6 bits respectively for single and double precision floating point formats) of the relevant exponent difference. With such schemes, even during situations when the exponent differences are greater than the width of significand; the significands may be shifted through certain finite number of bit positions, in case the barrel shifters are not activity scaled. With transition activity scaled schemes, whenever a barrel shifter is activity scaled, the magnitude of alignment shift impressed upon the shift control lines is the last shift.

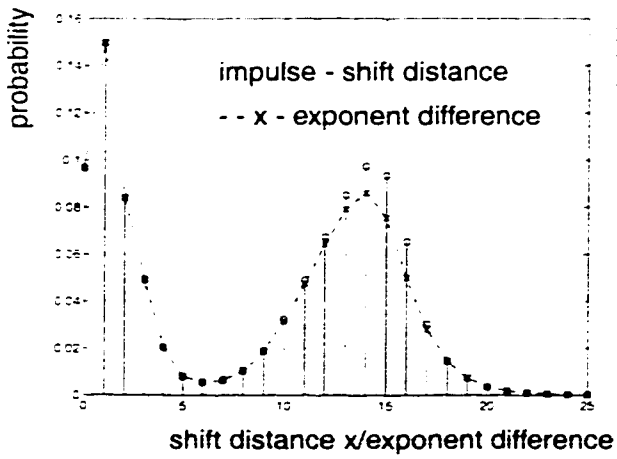


Figure 7.8 - pdf of shift distances

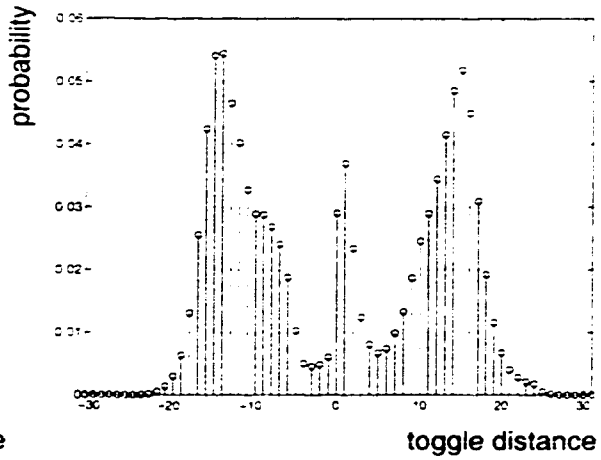


Figure 7.9 - pdf of z[n] = x[n] - x[n - 1]

Because of these reasons, the shift distance x is different from exponent differences; the precise relation between these parameters is a function of FADD data path/barrel shifter organization.

With equation (7.5), it is assumed that x is wide sense stationary (WSS). In floating point DSP operations, strictly speaking, $x[n]$ represents a non stationary random process [106]. Though $x[n]$ is non stationary, the time averaged power consumption of FP units can still be characterized on the basis of an 'average' behavior of $x[n]$ [109]. In our simulation based study, the probability density function (pdf) of x is evaluated on the basis of the frequency distribution of $x[n]$ over the whole set of FP addition operations during the underlying experiment. Figure 7.8 illustrates sample pdfs of x (non activity scaled version) as well as the underlying exponent differences, the relevant frequency distributions of which had been observed during the filtering of white noise ($N(0,1)$). A sequence of white noise samples (128K) had been low pass filtered (single precision FP operations) using an 8th order elliptical filter (transposed direct form II IIR filter - pass band ripple 0.03 dB, stop

band ripple 140 dB, normalized cut-off frequency 0.2). In Figure 7.8, only a truncated region of the pdf is shown, that is, the probabilities are illustrated for shift distances/exponent differences upto 25 only. The expected shift in this case is 8.5886 while the expected exponent difference is 14.2360.

Definition 2 - Toggle distance: Toggle distance is defined as the difference between present shift and last shift: i.e., $z[n] = x[n] - x[n-1]$ where $z[n]$ represents toggle distance.

Definition 3 - Absolute toggle distance: The absolute toggle distance is defined as the absolute value of toggle distance.

Definition 4 - Expected toggle distance: The expected toggle distance is defined as the average number of bit positions through which the shift operation oscillates about the mean or expected shift. The expected toggle distance is computed as the mean value of absolute toggle distance, as given by,

$$E[|z[n]|] = \sum_{l=0}^{l_{max}} l P(|x[n] - x[n-1]| = l) \qquad \textbf{(7.6)}$$

With conventional IEEE single precision FADDs, the index variable l can assume values between 0 and 31 (0 and 63 for double precision). With activity scaled FADDs, l is a function of the organization of barrel shifters.

With transition activity scaled FADDs, most of the parameters discussed above are scaled versions of the relevant parameters of conventional FADDs. Figure 7.9 illustrates a pdf of

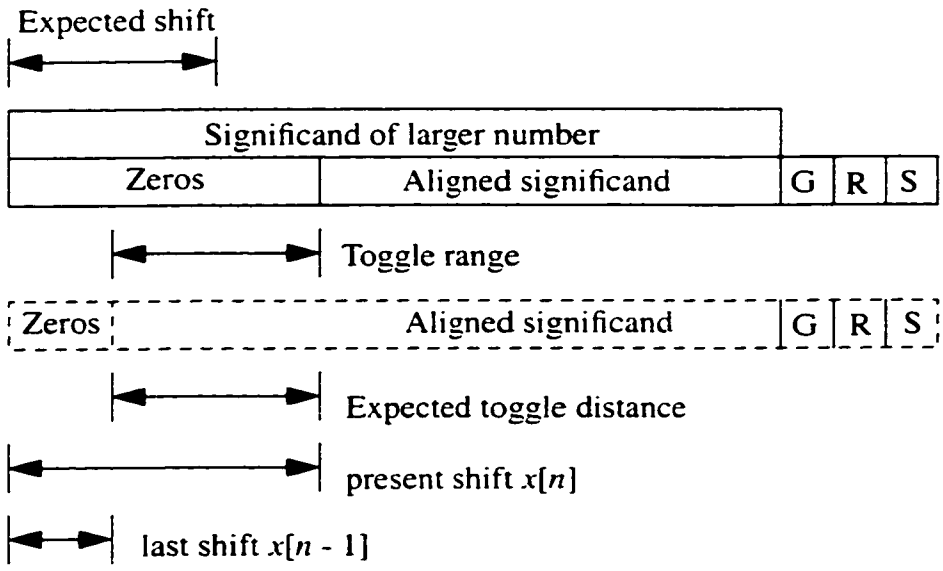


Figure 7.10 - Significand alignment in FP addition

pre-alignment toggle distances of a conventional FADD, that had been experimentally observed during filtering of random noise samples, described earlier. As can be seen, the pdf is not symmetrical about the origin. The expected toggle distance is 10.4004. Figure 7.10 illustrates the various parameters defined above. In general, the behavior of exponent differences as well as toggle distances during FP addition is a function of the structure of algorithms, distribution of data/co-efficients and the architecture of FADDs.

7.6.2.1. Control path switching

The time averaged power consumption of the shift control lines of a barrel shifter can be modelled by

$$P_{BSC} = \frac{1}{2} C_{Li} V_{dd}^2 N_{ADD} \sum_{i=0}^{B-1} t_i \tag{7.7}$$

where N_{ADD} represents the number of floating point additions per second, C_{Li} and t_i repre-

sent the capacitive loading and switching activity (average number of switching transitions during one clock cycle) of the ith shift control line. V_{dd} represents the supply voltage while B represents the width of the barrel shifter control word. In general, the control path power consumption of FADDs is dominated by the power consumption of barrel shifter control lines (both pre-alignment and normalization) as well as various data selection signals that facilitate the presentation of exponents, default results etc. With this, the control path power measure of FADDs can be modelled by

$$P_C \propto \sum_{i \in S} t_i Y_i \tag{7.8}$$

where S represents the set of all control signals, t_i and Y_i represent the transition activity and fanout of the ith control signal. With different architectural schemes, the number of control signals, their transition activity as well as fanouts differ.

7.6.2.2. Alignment driven data path switching

Whenever the position of the aligned significand endures oscillations about the expected shift, the significand data path bits that fall within the toggle range illustrated in Figure 7.12 endure higher transitions than bits which are outside the alignment induced toggling domain. The power consumption due to this phenomenon is proportional to the expected toggle distance. Introducing a factor μ to characterize the alignment driven data path switchings, the relation between μ and expected toggle distance is given by

$$\mu = \frac{E[|z[n]|]}{p} \tag{7.9}$$

In (7.9), E[|z[n]|] represents the expected toggle distance of the relevant FADD scheme while p represents the width of significand. With transition activity scaled FADDs, since the computing data paths are not active during bypass conditions, the power consumption due to alignment toggling is proportional to the expected toggle distance and data path assertion probability.

As discussed previously, during FP subtractions, the 1's (or 2's) complement of the aligned significand is added with the significand of the larger number. During such a scenario, the zeros appearing at the higher order bit positions (due to shift operation) of the aligned significand gets complemented into 1's. If the toggling between addition and subtraction operations is relatively significant, then the power consumption due to this activity is significant. Introducing a factor γ to characterize the power consumption due to the above switching activity; the relation between γ, expected shift and the probability for sign toggling is given by

$$\gamma = \frac{E[x]t_S}{p} \tag{7.10}$$

where, $E[x]$ represents the expected shift and t_s represents the probability for sign toggling. Here again, with transition activity scaling, the magnitude of γ is different. The power consumption of the significand adders of FADDs (which, by and large, reflects the alignment driven data path switching), taking into account the effects of alignment toggling and alignment driven/sign toggling dominated data path switchings can be repre-

sented by

$$P_{SI} = (\hat{P}_{ADD1} + \hat{P}_{LZ})[1 + \mu + \gamma] \tag{7.11}$$

where $\hat{P}_{ADD1}$ and $\hat{P}_{LZ}$ represents the time averaged power consumptions of significand adders and leading zero anticipatory logic under conditions when both μ and γ are zeros. Since the complexity of significand adders and LZA logic are comparable. the above equation can be simplified into

$$P_{SI} = 2\hat{P}_{ADD1}[1 + \mu + \gamma] \tag{7.12}$$

With TDPFADD, the relevant relation is given by

$$P_{SII} = 2\hat{P}_{ADD1}[1 + \mu_C]P(C) + \hat{P}_{ADD1}[1 + \mu_B + \gamma_B]P(B) \tag{7.13}$$

In the above equation. μ_C and μ_B represent the values of the parameter μ for the LZA and LZB data paths respectively of TDPFADD. γ_B represents the value of γ for the LZB data path. Since the LZA data path handles only subtractions, the question of alignment driven sign toggling doesn't arise in this case.

7.7. Area

The complexity of the leading zero anticipatory logic employed in IBM RISC floating point unit [22] [23] renders it an area intensive approach. Though the simplicity of the leading zero anticipatory logic scheme suggested in [24] is attractive for low power

designs, the overall architectural area implications of this FADD scheme (owing to the requirements for dedicated significand comparator and rounding/correction shift logic) makes it sub-optimal as far as area economy is concerned. With the proposed FADD scheme, the merging of rounding operation with significand addition offers area reduction. The area of the pseudo leading zero estimation scheme employed by the proposed architecture is virtually the area of leading zero counters. The absence of a significand comparator is another notable feature. With these area reduction measures, the additional area implications of the LZA data path of the proposed FADD scheme is comparable to the total area overheads of the leading zero anticipatory logic, significand comparator and rounding/correction shift logic of [24]. The use of 1's complement adders for the evaluation of functions like $|e1 - e2|$ (as well as $e1 > e2$, $e1 = e2$ and $e1 < e2$) involving integer data $e1$ and $e2$ [124] is attractive as far as power and area reduction are concerned. With these area optimization measures, the area of the proposed scheme will not be appreciably greater than that of schemes having leading zero anticipatory logic.

7.8. Results

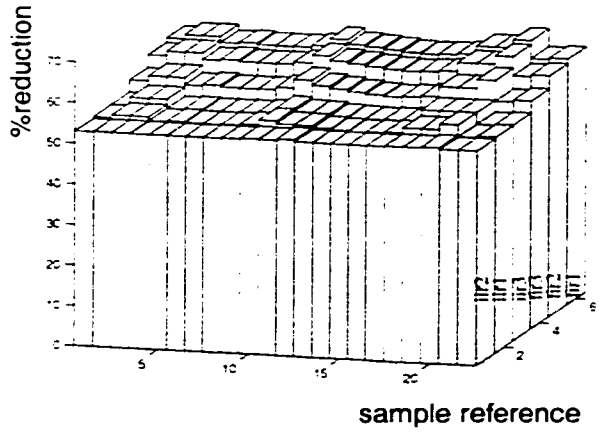
Instrumented digital filter programs, emulating a DSP MAC, had been developed. In our DSP model, the FP multiplier and FADD building blocks are treated as separate entities.The experiments involved the filtering of an assorted collection of data samples - both synthetic and real data [112]. The first among the synthetic signals is a sequence of white noise samples (N(0,1) IID RVs) of sample size 128K, while the second and third are auto regressive signals of the same sample size. Specifically, the AR model of the second signal is $y[n] = x[n] + 0.9*y[n - 1]$ while that of the third signal is $y[n] = x[n] + 0.5*y[n - 1]$. The first three filters are 8th order elliptical filters (low pass), having pass band ripple of 0.03

dB, stop band ripple of -100dB and normalized cut - off frequency 0.2. Filters I, II and III are direct form I, direct form II and transposed direct form II realizations of the same filter. The last three are low pass (normalized cut-off frequency of 0.2) FIR filters of order 64, 16 and 8 respectively. During the course of filtering (single precision floating point operations), frequency distributions of pre-alignment and normalization shifts, their rate of change and the relevant bit level activities had been collected.

Table 7.2: Data path utilization probabilities during filtering of synthetic data

IIR1			IIR2			IIR3			FIR1			FIR2			FIR3		
P(A)	P(B)	P(C)	P(A)	P(B)	P(C)	P(A)	P(B)	P(C)	P(A)	P(B)	P(C)	P(A)	P(B)	P(C)	P(A)	P(B)	P(C)
0.0589	0.5952	0.3460	0.0589	0.6116	0.3295	0.0595	0.7049	0.2356	0.2002	0.6755	0.1243	0.1765	0.6779	0.1456	0.1111	0.7415	0.1474
0.0589	0.6406	0.3005	0.0589	0.6373	0.3038	0.0601	0.6952	0.2447	0.2002	0.6963	0.1035	0.1765	0.7571	0.0664	0.1111	0.8485	0.0405
0.0589	0.6150	0.3251	0.0589	0.6162	0.3250	0.0599	0.7031	0.2370	0.2002	0.7033	0.0965	0.1765	0.7227	0.1007	0.1111	0.7938	0.0950

Table 7.2 presents the data path utilization probabilities of TDPFADD during the filtering of synthetic data. Significant among the observations based on the results presented above is the fact that the probability for the generation of a variable number of leading zeros during significand addition is comparatively low. This means, the leading zero evaluation based partitioning and transition activity scaling of significand data path provides significant power reduction. Figure 7.11 gives an illustrative example of the data path utilization based power reduction of TDPFADD in comparison to that of conventional FADDs, that had been observed during filtering of bipolar audio signals (22 assorted signals ranging in size 9131 to 6318040 samples) [112]. As anticipated, the worst case power reduction is better than 50%. Repetition of the experiments with unipolar signals also substantiates the results [112].



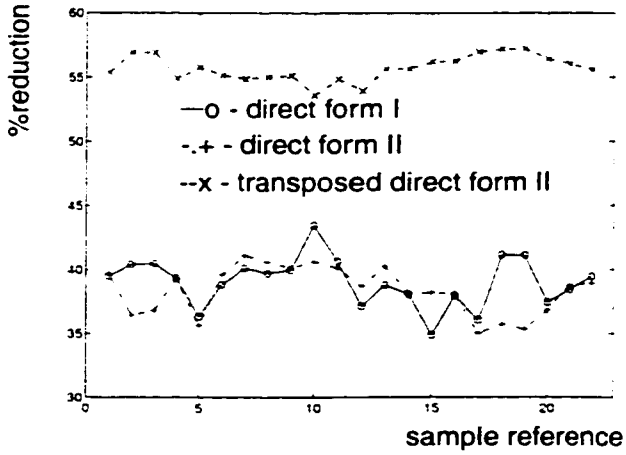
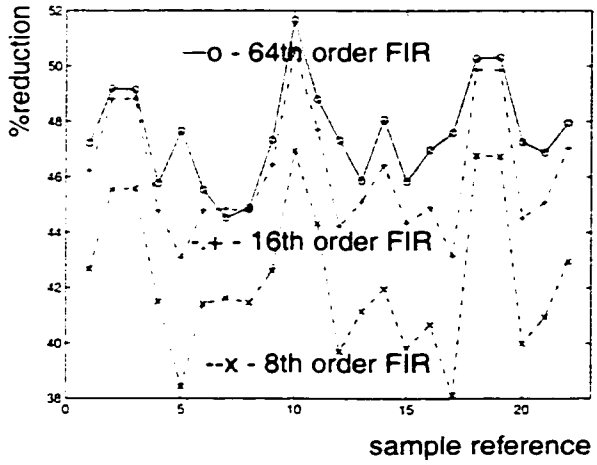


Figure 7.11 - Data path utilization based model **Figure 7.12 - Control path component (IIR)**



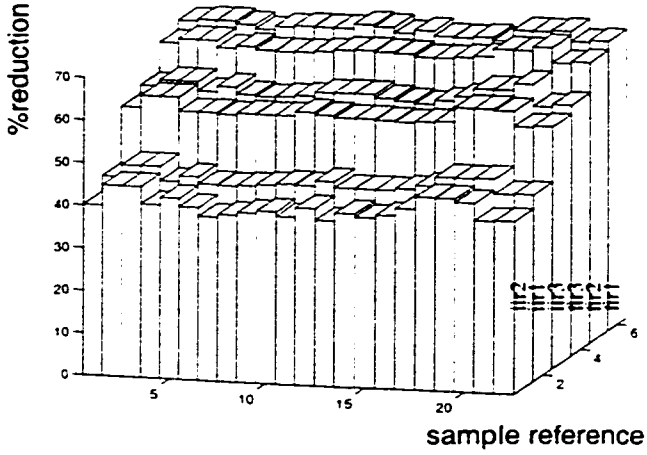


Figure 7.13 - Control path component (FIR) **Figure 7.14 - Data path component**

Figures 7.12 and 7.13 illustrate the reduction in control path power consumption offered by the TDPFADD during filtering of bipolar audio signals. During IIR filtering, the worst case reduction is around 35% while the best is better than 53%. The corresponding figures for FIR filters are 38% and 51% respectively. Figure 7.14 illustrates the reduction in alignment driven data path power consumption offered by the TDPFADD. The worst case reduction is around 40% while the best figures are around 70%.

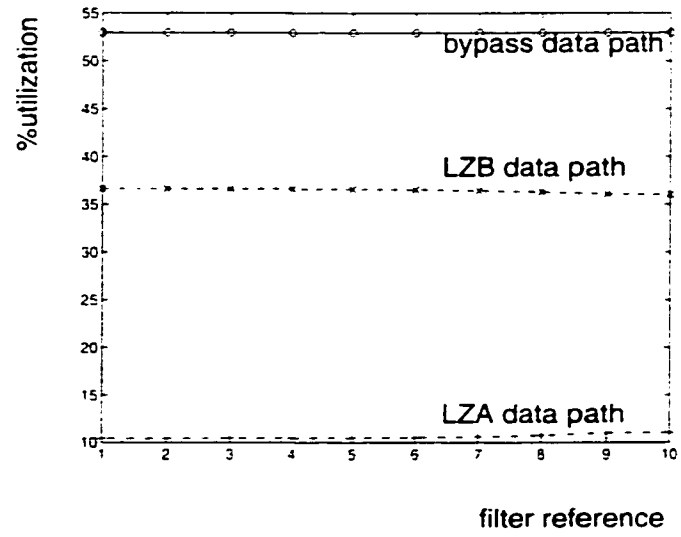


Figure 7.15 - Data path utilization during precision limited IIR filtering

With TDPFADD, the handling of precision limited DSP algorithms is of particular interest as far as the power implications of these units are concerned. In DSP applications, certain arithmetic operations may fail to impact the final result due to the limitations of floating point data formats. Under such circumstances, with FADDs, the question of bypasses gathers significance. Symmetrical band pass/stop filters having a normalized centre frequency of 0.5 exhibits relatively large differences between the exponents of adjacent filter co-efficients. With these filters, the probability that the FADD endures bypass is relatively significant. Figure 7.15 illustrates the data path utilization characteristics of TDPFADD during precision limited IIR filtering of white noise samples. 128 K of white noise samples had been band pass filtered using 8th order elliptical filters (transposed direct form II) of different stop band ripples, but the same pass band (0.4 to 0.6) and pass band ripple (0.03dB). The stop band ripples are -125dB, -115dB, -105dB, -95dB, -85dB, -75dB, -65dB, -55dB, -45dB and -35dB. As is evident, the probability for FADD bypasses are always greater than 0.5. Also, the probability that the LZA data path is asserted is around

0.1. Similar behavior is exhibited by direct form I and direct form II filters (even FIR filters exhibit such behavior) as well. With this class of filters, the estimated power reduction offered by the TDPFADD architecture is around 75%.

7.9. Discussion

As seen from the results presented in the foregoing section, the data path utilization based power measures highlights a global perspective of the power reduction offered by the TDPFADD. The relative power implications of control/data paths indicates the effect of transition activity scaling. With TDPFADD, because of the simplification of barrel shifters, the number of layers of nodes undergoing toggling activity is reduced. For example, the pre-alignment shifting of LZA data path as well as normalization shifting of LZB data path are handled by single stage MUXs. Compared to multi-stage barrel shifters, the data path switching component of these single stage shifters are insignificant.

The expected toggle distances of TDPFADD are appreciably smaller than that of conventional FADDs due to the effect of transition activity scaling. The cumulative effect of reduction in expected toggle distance as well as data path transition activity scaling results in substantial reductions in alignment driven data path switching activity. In our architecture, the question of alignment driven/sign toggling dominated switching activity of aligned significand can be easily addressed. Instead of 2's complement addition, the subtractions handled by the LZB data path can as well be handled by a 1's complement adder, so that the significand of the larger floating point number can be complemented during subtraction instead of the other. Such an approach guarantees the elimination of alignment driven/sign toggling dominated data path switching activity during subtraction.

While the experiments undertaken for the present study does not represent an exhaustive set of DSP experiments. the results however do validate the architectural partitioning scheme and transition activity scaling. Apart from a few representative cases of results presented here. additional results are presented in [112]. Although the alignment behavior exhibits sensitivities towards the actual DSP algorithms. data distribution and co-efficient quantization. the question of FADD bypass is still relevant. Also. the probability that FP additions produce a significand with a variable number of leading zeros is relatively small. Repetition of the experiments using different types of filters having various specifications also substantiates the above findings. The TDPFADD. apart from power reduction. do also offer throughput enhancement. Compared to the FADD reported in [24]. the latency of TDPFADD core is less by around two pipeline stage delays (which accounts to a 40% reduction in delay) - one stage delay reduction due to zero overhead rounding and the other due to the reduction in logic depths of the computing data paths by way of eliminating one barrel switch from the critical path (which is approximately equal to the delay of another pipelined stage). With this. the worst case reduction in power delay product offered by the TDPFADD is 70%. The reduction in number of pipelined stages has an extra dimension on power. The more the number of pipeline registers. the higher the power consumption.

Addressing the question of FADD bypass in the most pessimistic way - assuming that the exponent differences of the numbers are always less than the width of significand so that adder bypass conditions do never occur - the power reduction offered by the TDPFADD scheme is still around 50% (which gives a 3X reduction in power delay product). in comparison to that [24]. This fact is obvious because of the following: Compared to conven-

tional FADD schemes. the computing data paths of the TDPFADD do have only one barrel shifter while all other schemes have two. In contrast to other schemes. the leading zero counting logic of the TDPFADD is operational only during situations that demand its use. The incorporation of zero overhead rounding and data path simplifications accelerate the speed performance of TDPFADD by as much as 66% in contrast to that of [24].

7.10. Conclusion

The architectural design of an activity scaled. triple data path floating point adder is presented. Transition activity scaling of functional units for power reduction offers promising results as far as architectural design for low power operation is concerned. The power/ delay reduction offered by the proposed architecture renders this scheme an attractive choice for DSP applications as well as general purpose computing.

Chapter 8

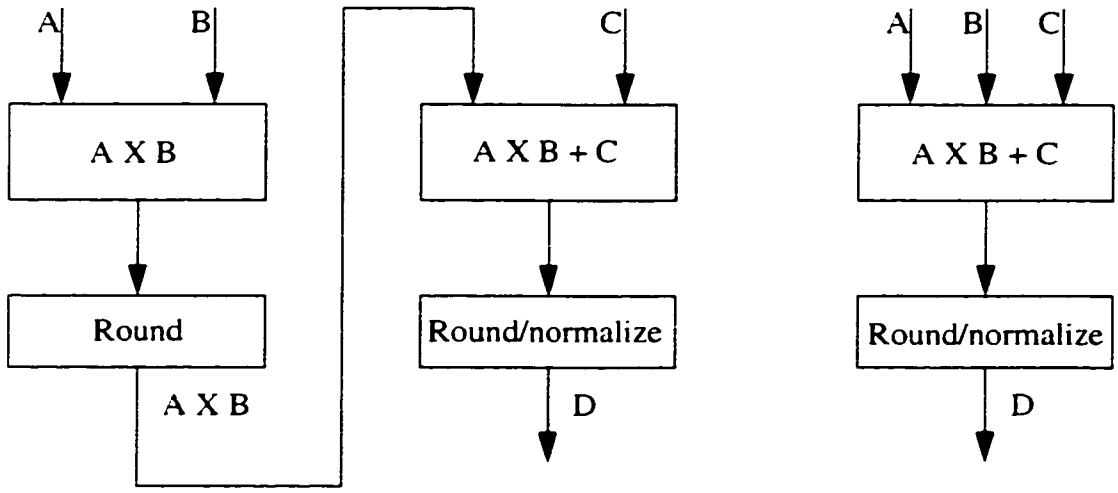
Low Power Floating Point Multiply - Accumulate Fused Architecture

8.1. Introduction

The computation of multiply - accumulate is fundamental in many scientific and engineering applications. Though the number of distinct computational operations involved in the evaluation of the result of a MAC operation are more than one - evaluation of a product and the summation of this product with the previously accumulated sum; these operations can be concurrently performed. From an architectural point of view, hardware units that exploit concurrency for the computation of MAC operations are termed multiply - accumulate fused (MAF) architectures. The fusion of multiply and accumulate operations involving fixed point operands is rather straight forward, while this is not true with floating point operands. In fixed point MAFs, the results of previous MAC operations are treated as an extra partial product. With floating point units, the summation operation is rather complicated. The fundamental difference between floating point addition and fixed point addition arises in the handling of significands. Significand alignments - pre-addition alignment shifts as well as post addition normalization shifts - are mandatory in floating point additions; which demands the use of exponent differencers, barrel shifters and leading zero estimators. Though the concurrent handling of multiply and addition operations involving floating point operands exacerbates the complexity of hardware units; with an ever escalating demand for high performance floating point dot product computing units,

newer approaches are emerging [3] [23]. The IBM RISC/6000 reported in [3] [23] had been the first FPU with multiply - accumulate fused architecture. While the above unit incorporates a novel architecture for MAC fusion; this architecture, however, had never been fully scrutinized from a low power perspective though the power/area/complexity measures of the leading zero anticipatory logic (LZA) reported in [22] had been adjudged sub optimal in light of the findings reported in [24].

As previously stated, since the addition of floating point operands dictates the addition of significands that are appropriately aligned, the MAF architectures envisage the addition of a group partial products of significand multiplication as well as the significand of the previously accumulated sum that are properly aligned in relation to one another. In the IBM RISC/6000 system, the significand of the previously accumulated sum is shifted (left or right) relative to the positions of the partial products of significand multiplication. Though the theoretical validity of such an approach is irrefutable, there exists reasons to suspect that such a scheme results in a sub - optimal architecture as far as the power implications of multiply - accumulate fusion are concerned. The reason behind such skepticism is simple. Once the position of the significand of the product or (rather partial products) is fixed, the significand of the previously accumulated sum need be shifted through a larger number of bit positions. The larger the width of the shifter data path, the larger the power/delay overheads. This chapter addresses the development of an alternative scheme for MAF compared to that used in IBM RISC/6000 so that the architectural power/delay implications of floating point MAC fusion are optimal. The rest of the chapter is organized in the following order. Section 2 gives a brief treatment of the MAF strategy used in IBM RISC/6000. Section 3 presents the architecture of the proposed scheme. The applicability of



(a) Classical multiply - adder **(b) MAC fusion**

Figure 8.1 - Multiply - accumulate fusion

transition activity scaling and data path simplifications for power reduction is also presented. Section 4 presents the significand alignment based power models of MAFs. Section 5 highlights the relative power implications of the proposed scheme in comparison to that of IBM RISC/6000. Section 6 presents a discussion on certain miscellaneous aspects. Section 7 concludes the chapter.

8.2. MAF in IBM RISC/6000

Figure 8.1 illustrates the concept of MAF [3] [23]. In Figure 8.1 (a), the operations of multiplication (to form the product A X B) and summation (A X B + C) are performed sequentially. The floating point product and sum are separately rounded. In Figure 8.1 (b), the multiplication and addition operations are merged into a single operation. Rounding is performed only once. Figure 8.2 illustrates the shift implications for significand pre-alignments. Since the position of the significand of the product is taken as the reference, the

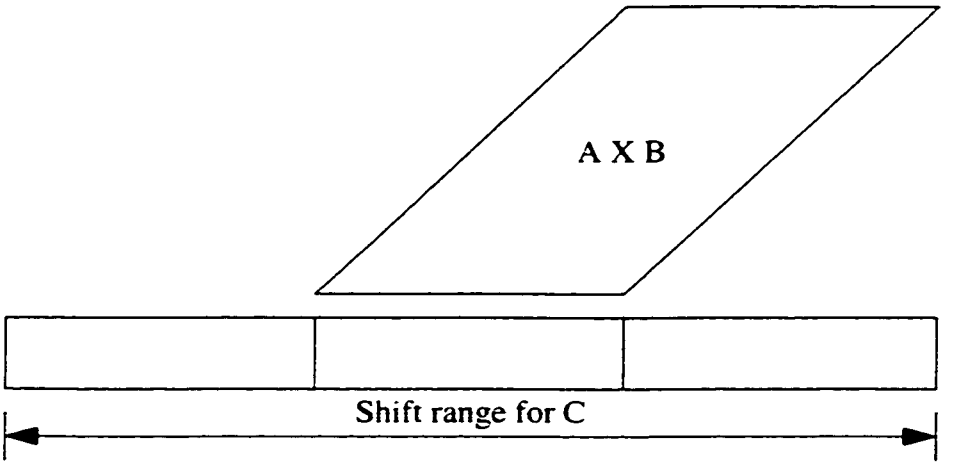


Figure 8.2 - Shift Implications with multiply - accumulate fusion

significand of C gets aligned all the time irrespective of the value of its exponent. For those values of exponents of C that are greater than that of the product A X B, the significand of C is left shifted through an appropriate number of bit positions and vice versa. Since the rounding operation is performed only once, the MAF produces results that are more accurate than the scheme shown in Figure 8.1 (a) would have.

8.3. The proposed MAF scheme

Figure 8.3 illustrates the schematic of the proposed MAF. In this scheme, alignment shifts are not restricted to the significand of C, rather the product of the significands of A and B can also be shifted. The shifting of the product of the significands of A and B, however, is accomplished in an indirect way. The following equation highlights the shifting approach.

$$\frac{AB}{2^{edif}} \equiv A\frac{B}{2^{edif}} \tag{8.1}$$

A right shift of the product of the significands of A and B is equivalent to dividing this

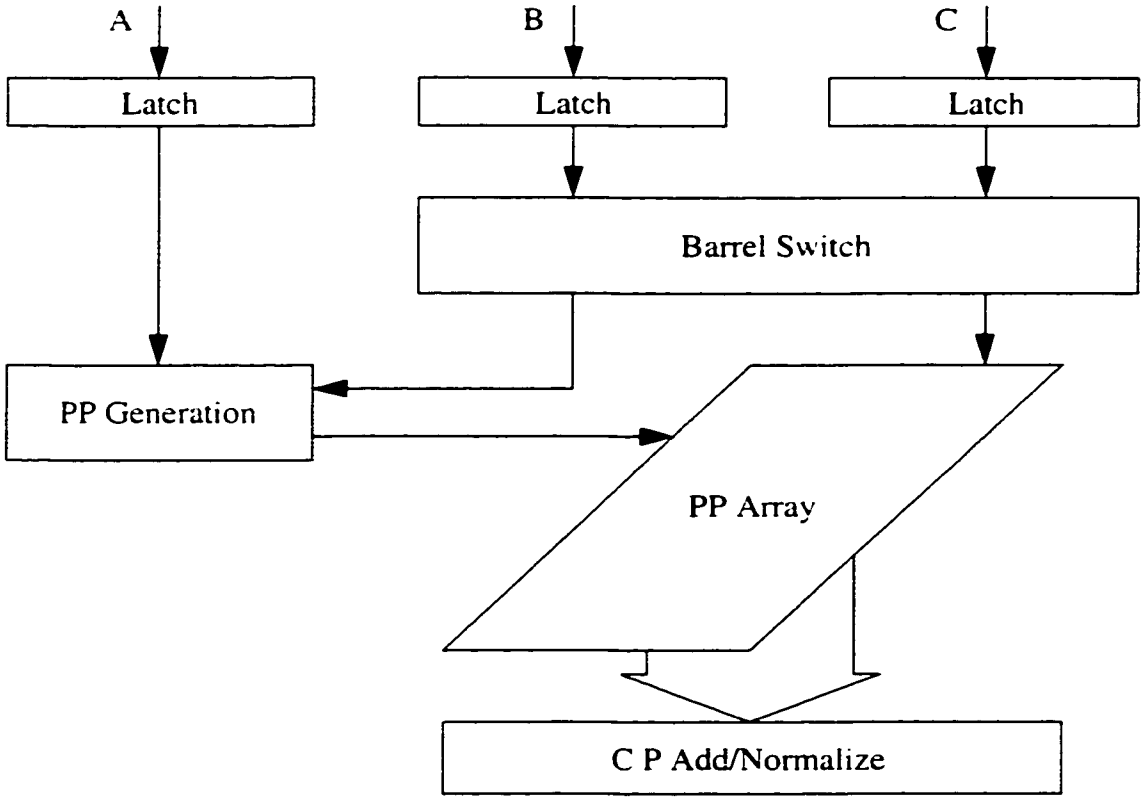


Figure 8.3 - Proposed MAF

product by $2^{e_{dif}}$, where e_{dif} represents the absolute value of the difference between the exponents of the product and C. In essence, shifting of the product is equivalent to shifting one of the significands and then forming the product. Whenever the exponent of the product is greater than that of C, the significand of C is right shifted. The following paragraphs explain the organization of the proposed MAF.

8.3.1. Pre-alignment control

During multiplication of two floating point operands, A and B, the exponent of the product is given by

$$e_{AB} = e_A + e_B - e_{bias} + OV \tag{8.2}$$

In the above equation e_A and e_B represent the exponents of the numbers A and B respectively. e_{bias} represents the value of the exponent bias. OV represents the overflow bit from the MSB position of the significand multiplier. Whenever an overflow does occur from the MSB position of the significand multiplier, the tentative exponent of the product is incremented and the product of significands is subjected to a normalization shift (and rounding).

In general, since an operation of the type A X $B + C$ relies on significand alignment shifts for the computation of the result, an exact value of the shift magnitude is required right at the beginning, so that the operations of multiplication and summation can be concurrently performed. Or in other words, there is no room for any ambiguity as far as the evaluation of the exact value of the exponent of the product is concerned. But the exact value of the product's exponent is not known till the process of significand multiplication is completed. In order to circumvent this difficulty, the following approach is proposed.

In floating point computation, it is imperative that the alignment of significands is consistent with the binary point reference. The uncertainty with respect to the generation of OV bit in significand multiplication is not a real problem as long as the consistency of binary point is taken into account during hardware design. With the fusion of multiply and accumulate operations, the normalization of the product of significands is not treated in isolation, rather the normalization of the result of a MAC operation is envisaged. Because of this, in contrast to the significand of C, the significand of the product encompasses two bit

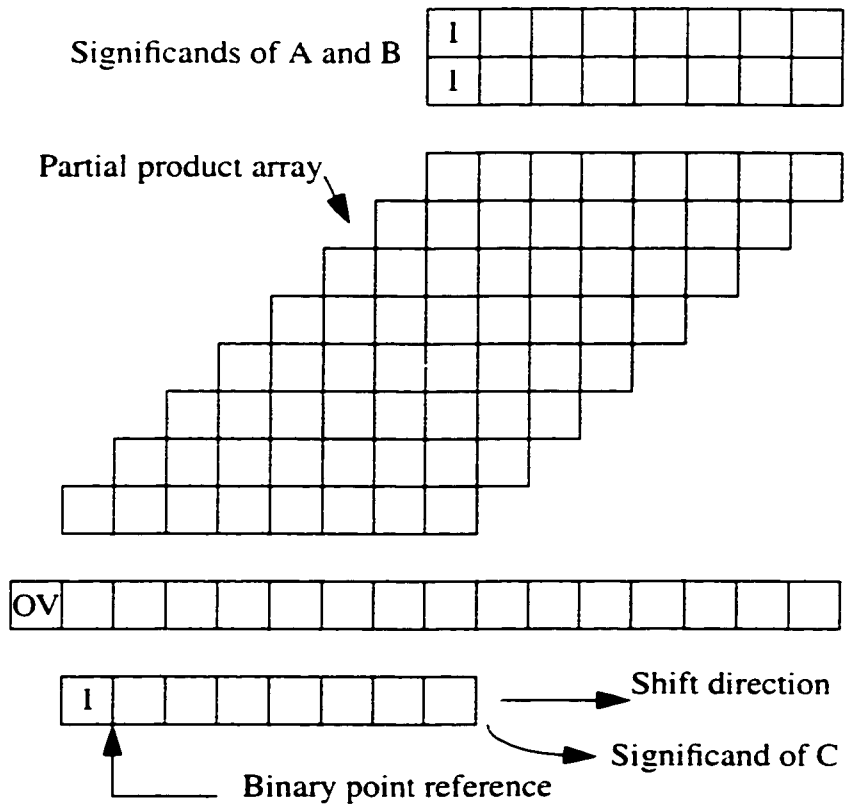


Figure 8.4 - Significand alignment of C

positions towards the left of the binary point reference. As a natural consequence, during situations when a shift of the significand of the product is envisaged, the shift magnitude is always tallied with respect to the *OV* bit. The tentative exponent of the product, for the purpose of shift evaluation is always chosen as

$$e_T = e_A + e_B - e_{bias} \tag{8.3}$$

With the above, the exponent difference is given by

$$e_{dif} = |e_T - e_C| = |e_A + e_B - e_{bias} - e_C| \tag{8.4}$$

Figure 8.4 illustrates a representative case which involves the alignment of the significand of C. Here, for the sake of clarity, the significands are represented as eight bit numbers. Now, if the process of significand multiplication has resulted in the generation of an OV bit at the MSB position of the product, the operation of significand addition will still produce correct results. With the OV bit asserted, the situation is rather equivalent to left shifting the product of the significands of A and B, through one bit position. Post addition normalization shifts during such a situation can be as much as two bit positions.

Figure 8.5 illustrates situations that demand the shifting of the product of significand multiplication. As previously stated, the shifting of the product is accomplished through the shifting of one of the component significands of the product. In this scenario also, the magnitude of shift is specified by equation (8.4). Whenever one of the significands (of A or B) is shifted through a number of bit positions, the product of the shifted significand with that of the other is guaranteed to have a number of leading zeros that is equal to the number of bit positions through which the above shift had been made. Now the assertion of the OV bit has the effect of shifting the product left by one bit position. Or in other words, the assertion of the OV bit is equivalent to annulling a right shift of one bit position. The effective shift in this case, in relation to the leading 1 of the significand of C, is given by

$$e_{dif} = |e_T - e_C| = |e_A + e_B + 1 - e_{bias} - e_C| \qquad \textbf{(8.5)}$$

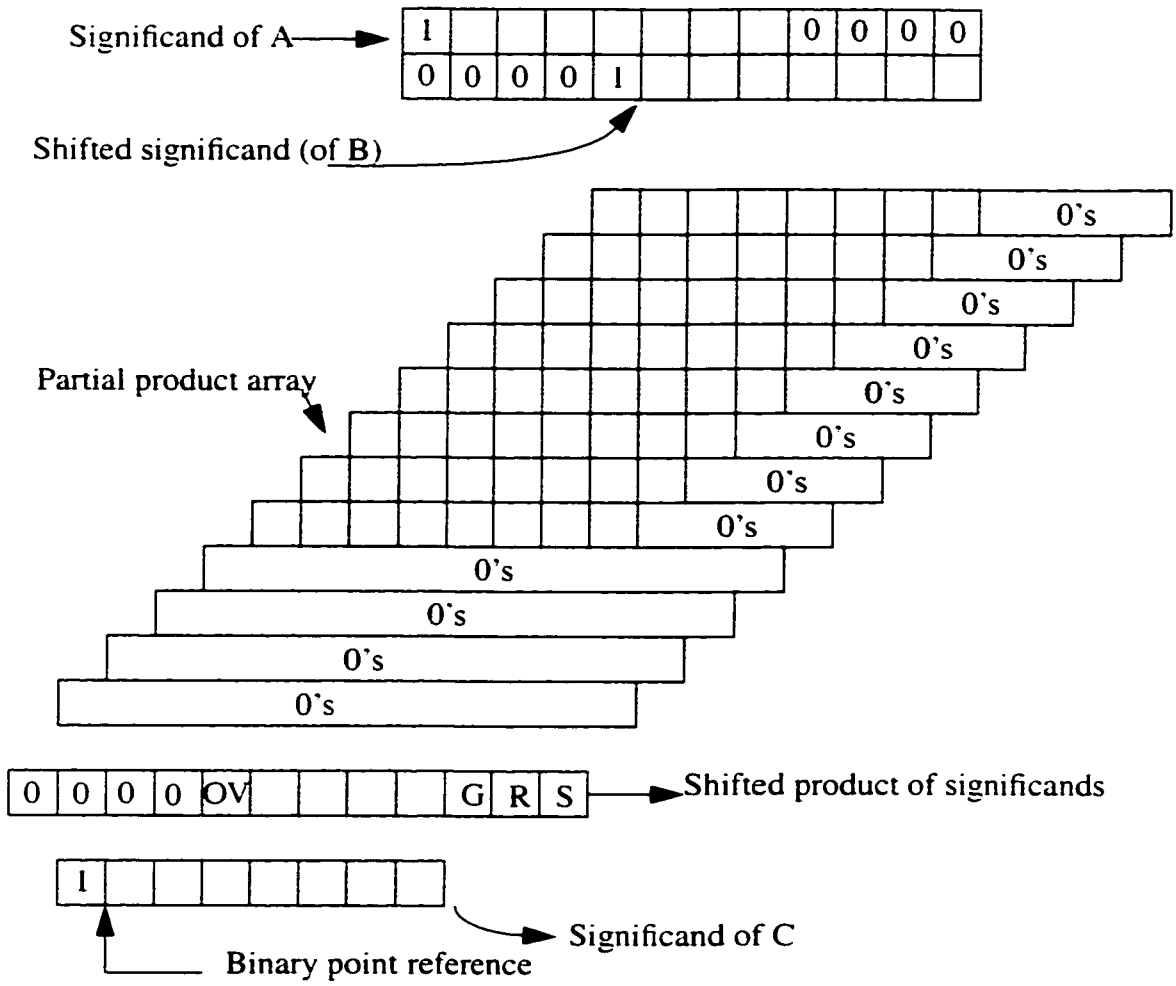


Figure 8.5 - Alignment of the product of significands of A and B

8.3.2. Partial product array design

Since the proposed MAF scheme envisages a conditional shifting of the product of significands (which is indirectly accomplished through the shifting of one of the component significands), a natural question that does arise in the design of the partial product array is the size of the array. In general, integer multiplication[1] involving n bit numbers produces n partial products, each having a length of n bits. Apart from illustrating the alignment

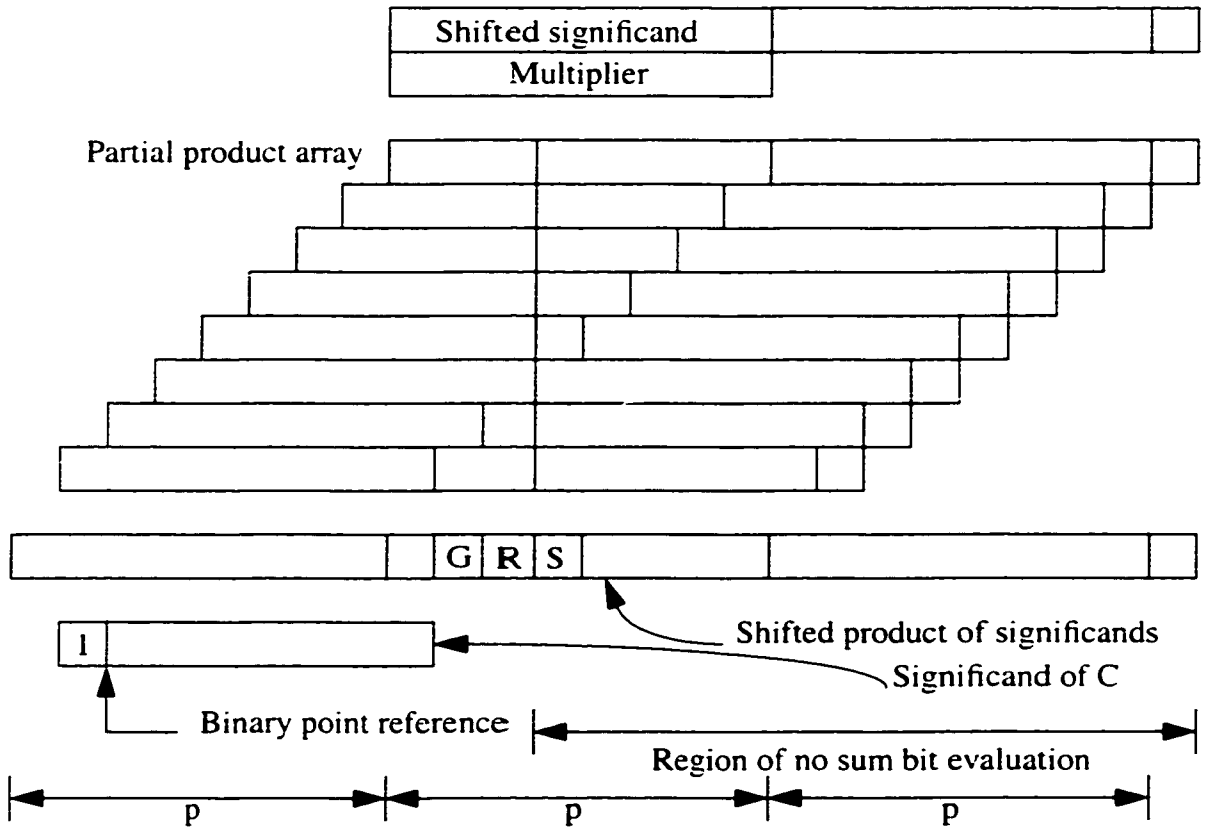


Figure 8.6 - Partial product array - Scheme I

scheme of the product of significands relative to the significand of C, Figure 8.5 also illustrates a possible scenario for partial product generation. As can be seen, the number of partial products in this scheme is more than n, if the shifted significand is used as the multiplier. Partial products can also be formed by using the shifted significand as the multiplicand, as depicted by Figure 8.6. In this case, the number of partial products is always fixed - with p bit significands, the number of partial products is p. If all of the shifted out bits of

1. Modified Booth Algorithm (MBA) [16] [17] is generally used for partial product reduction. Recent investigations [18] [125] regarding the power/delay implications of partial product generation and compression techniques, however indicate that MBA techniques are sub-optimal as far as power/delay minimization is concerned. Because of this reason, the proposed MAF scheme doesn't envisage MBA techniques for partial product reduction.

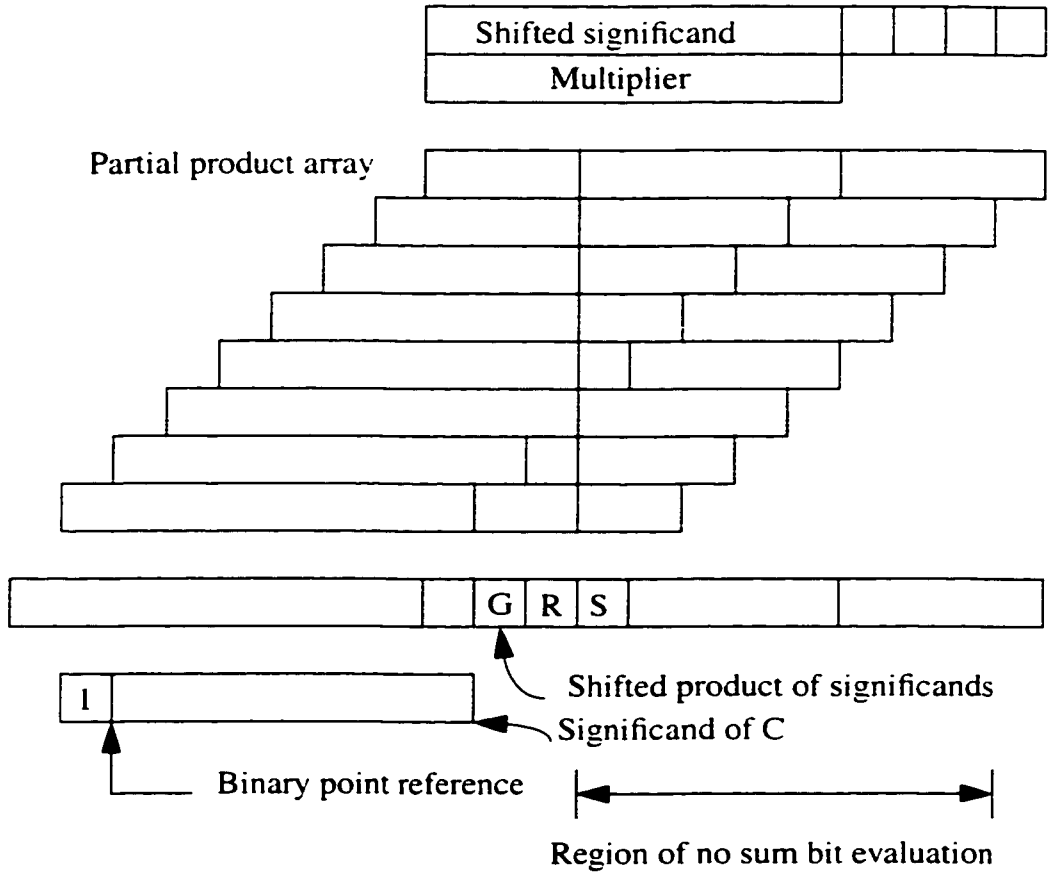


Figure 8.7 - Partial product array - Scheme II

the significand are retained for partial product generation, the width of the partial products is $e_{dif} + p$. However, the upper bound for shifting the multiplicand significand is $p + 1$. With $p + 1$ bit shift, the product is guaranteed to have a minimum number of $p + 1$ leading zeros starting at the MSB end of the product. Since the product of IEEE significands falls within the range, $1 \leq product < 4$, the MSB end of the product occurs at the second bit position towards the left of the binary point reference. With $p + 1$ bit shift of the multiplicand, there is still a product bit at G bit's position as long as $2 \leq product < 4$. This bit decides the conditional rounding of the significand of C. If this bit is zero, i.e., $product < 2$, there is no need to perform any rounding on the significand of C. Shifting of the multiplicand signifi-

cand beyond $p + 1$ bits is guaranteed to produce a product with at least $p + 2$ leading zeros. Or in other words, summation (and subsequent rounding) of this product with the significand of C produces a resulting significand which is guaranteed to be no different from the original significand of C.

With $p + 1$ bit shift of the multiplicand, the width of the aligned product is $3p + 1$, if all the shifted out bits of the multiplicand significand are retained. Among the $3p + 1$ bits of the aligned product, many of the lower order bits are seldom used in the subsequent addition with the significand of C. In fact, during situations when the MAC operation envisages a single rounding step (instead of rounding the product and rounding the sum) $2p - 3$ of the LSB bits doesn't assume any active role in significand summation. The only reason[1], if at all, these bits are to be computed is for the evaluation of the sticky bit of the aligned product. Even though the sticky bit can be computed by logically ORing the $2p - 2$ lower order bits, this, however is not the only choice [115]. For IEEE multipliers, the sticky bit can be generated through an evaluation of the trailing zeros of the product. For binary floating point operands, if the sum of the trailing zeros of the multiplicand and multiplier (significands) is greater than or equal to $p - 3$ ($p - 3$ represents the number of bits towards the right of G (guard) and R (round) bits), then the sticky bit is zero. For floating point MACs, the sticky bit of the shifted product can be set to 1, if the following relation is true.

$$TZ_A + TZ_B < p - 3 + ke_{dif} \tag{8.6}$$

1. For situations that demand higher precision for MAC computation, many of the lower order bits - depending on accuracy requirements - can be used for summation. During such a scenario, an appropriate number of the shifted out bits of the significand of C also has be retained.

In the above relation, *TZA* and *TZB* represent the trailing zero counts of the significands of floating point numbers *A* and *B* respectively. *k* is a 0/1 integer variable, which is zero whenever the exponent of the product is greater than or equal to that of *C* and vice versa. The above modular scheme for sticky bit computation is advantageous as far as complexity reduction of the partial product array is concerned. For bit positions towards the right of *G* and *R* bits, the sum bits need not be evaluated. Essentially, the partial product compression circuits at these bit positions need to compute only the carries injected towards the left of the sticky bit. During situations when, the significand of *C* is right shifted, the sticky bit due to this operation can be asserted if the condition $e_{dif} - 2 > TZ_C$ is true [124].

Although the scheme shown in Figure 8.6 is exemplary for high precision floating point computations, such a scheme may not always be required. The complexity of the partial product array can be reduced by discarding some of the shifted out bits of the aligned significand. Figure 8.7 illustrates a scheme that retains four of the shifted out bits. In this case, the width of the partial products is $p + 4$, instead of $2p + 1$ envisaged by Scheme I. The effect of multiplicand truncation is equivalent to the subtraction of 1/8 ULP (unit in the last place) from the product (worst case). In general, truncation produces a negative bias, and in a worst case scenario, this bias can accumulate. In practice, the rounding error produced in one MAC operation may get cancelled in a subsequent MAC operation, and hence the accumulation of truncation error is never a one to one addition. The accumulation of truncation error can be reduced, by extending the width of the accumulator [126].

The question of guard bit retention for multiplication operation is best addressed from an

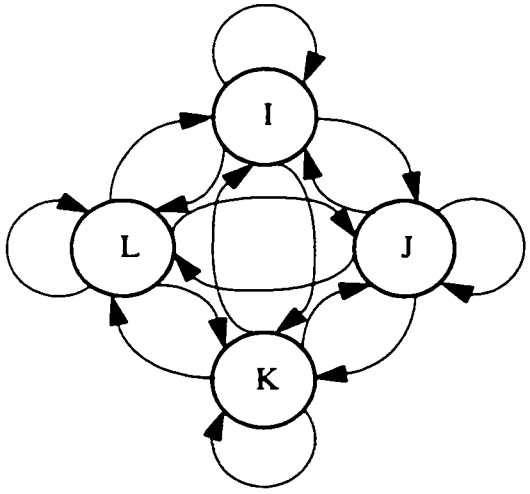


Figure 8.8 - FSM representation of MAF

Table 8.1: State assertion conditions

State	State assertion criterion	Activity scaled blocks
I	$e_c > e_T$ AND $e_{dif} > p + 1$	Entire MAF except control unit and bypass data path
J	$e_c < e_T$ AND $e_{dif} > p$	LZA and large barrel shifters
K	$1 < LZs \leq p$	Pre-alignment barrel shifter (large)
L	$LZs \leq 1$	LZA and normalization barrel shifter (large)

accuracy perspective. If the target application is very sensitive to rounding errors, then all the shifted out bits of the multiplicand need be retained. However, if the target application can tolerate a worst case error of a few ULPs, then a few of the LSB bits of the shifted multiplicand can be discarded. Multiplicand truncation results in compact partial product arrays, that is attractive as far as power/delay/area considerations are concerned. Appendix A presents an analysis regarding the effect of multiplicand truncation on precision.

8.3.3. Transition activity scaling for low power

In CMOS logic implementations, the minimization of expected switching activity is important as far as design for low power operation is concerned. From an architectural/ algorithmic point of view, transition activity scaling [61] of functional units provides a viable approach for the minimization of dynamic power consumption. The applicability of transition activity scaling for architectural power minimization of floating point adders (FADD) is reported in [98]. The power minimization approach outlined in [98] is applicable in the design of floating point MAFs also.

Figure 8.8 illustrates the switching activity function of the proposed transition activity scaled MAF. Table 8.1 lists the criterion for the assertion of various states as well as the modules that are activity scaled in each state. In state *I*, the MAF operates in a full bypass mode. During those situations when the difference between the exponents of the current sum and product ($e_{dif} = e_C - e_T$) exceeds $p + 1$, the results of a MAC operation is not different from the current sum. In such a situation, neither the evaluation of the product nor the addition of the product with the current sum is required. Essentially, the whole of the MAC core except the control unit and bypass data path can be activity scaled during such a situation. In state *J*, the exponent difference is still greater than the width of the significand, but the current sum is relatively insignificant compared to the product. That means, it is essential that the product be evaluated, but the summation of this with the current sum is not required. Whenever, the current sum in no way does affect the MAC operation, there is no need to perform any alignment shifts on such an operand, which means, the pre-alignment barrel shifter can be activity scaled. During such a situation, since the product of the current multiplication replaces the current sum at the end of the present cycle, there is no need to evaluate the presence of leading zeros of the resulting significand. Normalization shifts (right shift) are also upper bounded during such a situation. Hence, the large normalization shifter (left shift) can also be activity scaled.

Table 8.2: Generation of variable number of leading zeros in significand addition

Exponent difference	Remarks
$edif = 0$	OV can be 0 or 1
$edif = 1$ AND $e_C < e_T$	Variable number of LZs iff $OV = 0$

Table 8.2: Generation of variable number of leading zeros in significand addition

Exponent difference	Remarks
$edif = 1$ AND $e_C > e_T$	OV can be 0 or 1
$edif = 2$ AND $e_C > e_T$	Variable number of LZs iff $OV = 1$

In state K, the MAC endures operations that are likely to result in a significand with a variable number of leading zeros. In significand addition, a variable number of leading zeros can be generated during the subtraction of one floating point number from another only if the numbers are nearly equal. In order that the numbers be nearly equal, the exponent difference should not be large. Table 8.2 lists various situations during which a variable number of leading zeros can appear at the MSB end of the result significand. Since the exponent difference is less than or equal to 2, during situations that result in a significand with a variable number of leading zeros, activity scaling of significand pre-alignment barrel shifter offers power reduction.

Table 8.3: Special operations

Control function	Remarks
Product = 0 (A or $B = 0$)	Result of MAC = current sum
$e_A + e_B + 1 - e_{bias} < e_{min}$	Multiplication underflow, result of MAC = current sum
$e_A + e_B - e_{bias} > e_{max}$	Multiplication overflow
$C = 0$ or $< f_{min}$	Addition underflow

In state L, the MAC performs floating point operations that is guaranteed to produce, at the most, one leading zero. The normalization shifts in this state are upper bounded - a maximum right shift of 2 and a maximum left shift of 1. Since the normalization shift distances

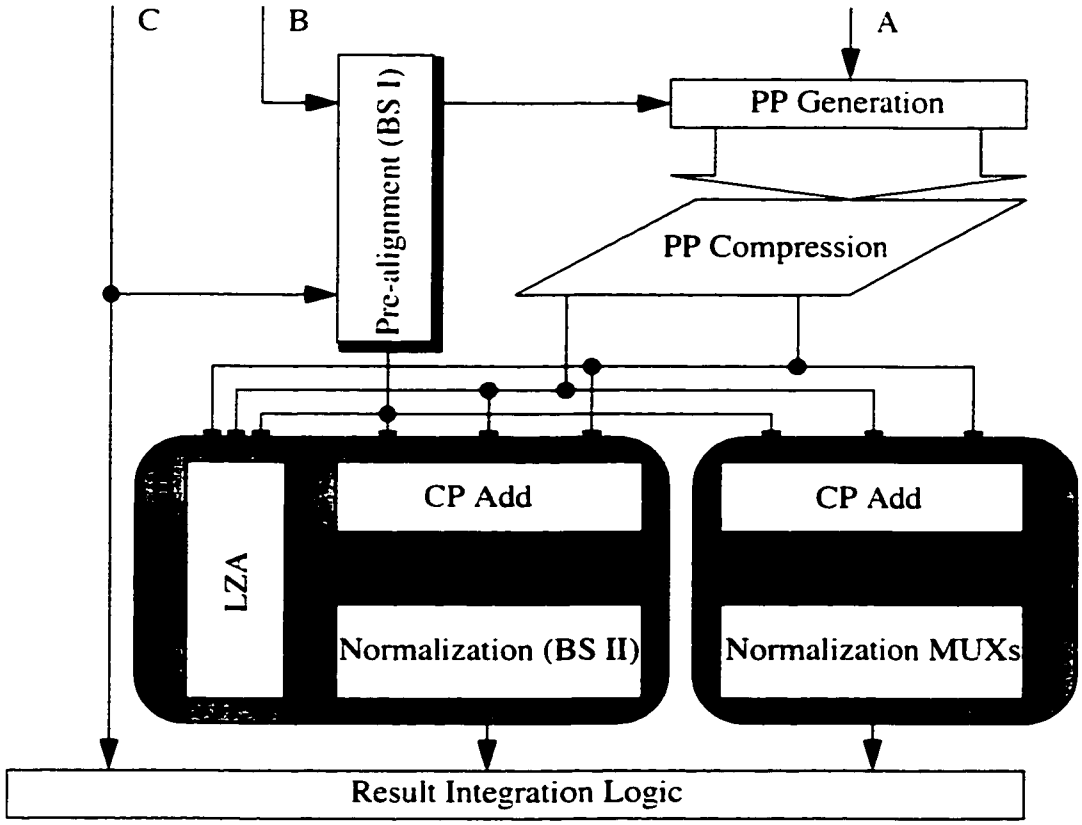


Figure 8.9 - Significand data paths of MAF

are limited, there is no need to excite the large barrel shifter (which is capable of performing left shifts upto p). Activity scaling of this barrel shifter is advantageous as far as transition activity minimization is concerned.

Apart from the activity scaling conditions discussed above, there are a few other situations that are of interest in MAF design. Table 8.3 lists some special operations that are of interest. The first and foremost among such situations is the handling of products that are guaranteed to be zero. If either of the operands - multiplier or multiplicand - is a zero, the multiplication is guaranteed to produce a zero result. The result of a MAC operation during such a situation is known apriori - the previous sum. The whole of the MAC core,

except the control unit and the bypass data path. can be activity scaled during such a situation. The handling of multiplication underflow/overflow, addition overflow/underflow and treatment of special quantities also leads to interesting situations. The evaluation of the multiplier/multiplicand for zero conditions can be merged with the sticky bit evaluation logic. If the number of trailing zeros of significand = p - 1 (the hidden leading 1 is introduced only if the number is not zero) and the exponent is also zero, then the number is zero. The evaluation of the MAC result for a zero condition is trivial with 1's complement conditional sum/carry select adders [124]. Since the evaluation of signals that delineate situations listed in Table 3 is mandatory for MAC operations, their reuse for transition activity scaling is rewarding as far activity minimization of the MAF is concerned.

Figure 8.9 presents the schematic of the significand data path of the proposed transition activity scaled MAF. For the purpose of transition activity scaling, the partial product generation and compression blocks can be treated as a single logic block. The pre-alignment barrel shifter (BS I) itself contains more than one distinct data path. One of these data paths handles significand alignments over a variable number of bit positions (3 to p + 1) while the other handles significand alignments that are upper bounded by two. The significand add/normalize blocks marked LZA/LZB data paths handles significand addition/ normalization during states J, K and L of Figure 8.8. The blocks marked CP Add are carry propagate adders. The operation of rounding is merged with addition [98]. The LZA block computes the number of leading zeros of the significand. The normalization barrel shifter of the LZA data path (BSII) includes a left shifter that can shift through 0 - p bits as well as a 0/1 bit right shifter. With the leading zero bounded data path (LZB), the normalization shifter is a left/right shifter which can handle a maximum left shift of 1 and a maximum

right shift of 2. The result integration logic integrates the results from various data paths, subject to the assertion of various states.

8.4. Power/performance measures

As discussed earlier, with the proposed scheme, significand alignment operations are much simpler compared to that of IBM MAF. Because of this, the switching activity of barrel shifters are expected to be significantly less. Also, because of transition activity scaling, the switching activity within the data paths are minimized. The following paragraphs present the various power models that reflect these aspects. Before we go into the specifics of the power models, an overview of the significand alignment behavior of the IBM architecture shall be presented. Figure 8.10 illustrates the probability density function of the difference between the exponents of C and AB of the operation $Y = AB + C$ of the IBM MAF. The underlying frequency distribution of exponent differences had been observed during low pass filtering of white noise. 128K of N(0,1) IID RVs had been filtered using an 8th order elliptical filter (transposed direct form II) of normalized cut-off frequency 0.2, pass band ripple 0.03 dB and stop band ripple -100 dB. The negative differences indicate situations when $C > AB$ and vice versa. Figure 8.11 illustrates the temporal behavior of pre-alignment shifts evaluated as a pdf of present shift - last shift. The higher the uncertainty with shifts between adjacent cycles, the larger the variance of this pdf. With the proposed scheme, because of architectural modifications and transition activity scaling, the correlation between adjacent shifts is increased, which reflects on power.

Figures 8.12 and 8.13 illustrate the pdfs of normalization shifts as well as its first order difference, the underlying frequency distributions of which had been observed during IIR fil-

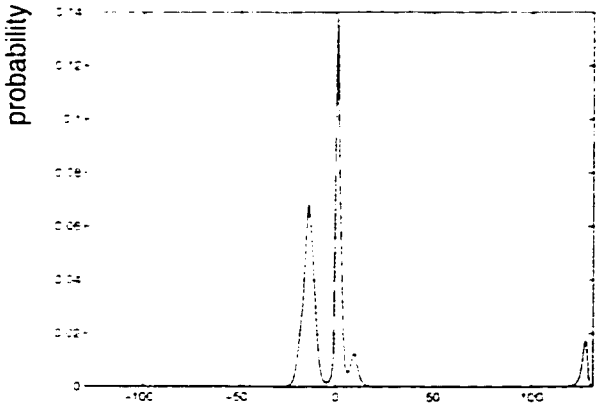


Figure 8.10 - pdf of exp. diff. in IBM MAF

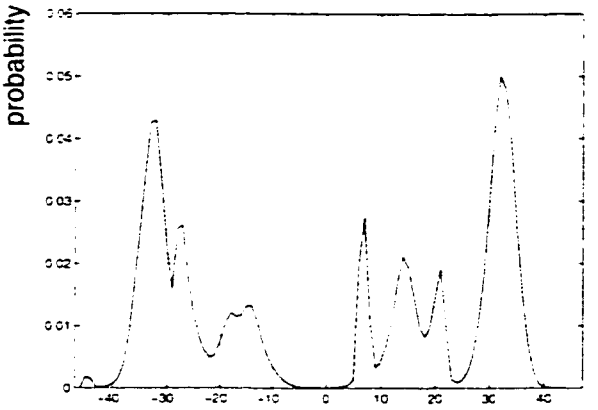


Figure 8.11 - pdf of temporal shift behavior

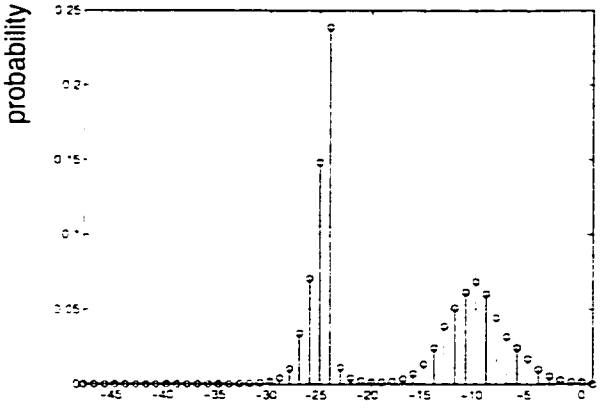


Figure 8.12 - pdf of normalization in IBM MAF

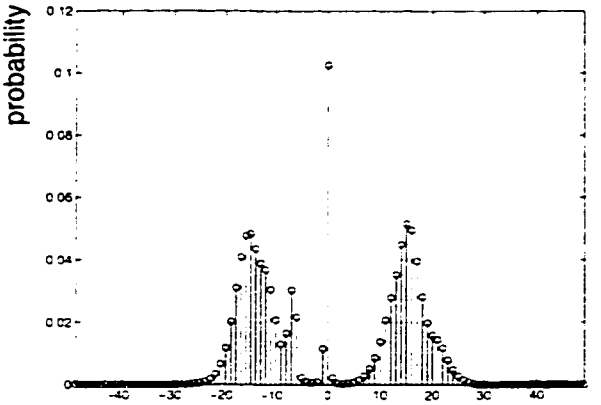


Figure 8.13 - pdf of temporal shift behavior

tering of white noise, using the transposed direct form II realization. In FADDs, normalization shifts through a large number of bit positions are required only during situations when the process of significand addition results in a large number of leading zeros. During all other situations, normalization shifts are limited. However, with the IBM MAF scheme, normalization shifts can be large even during other situations. With this scheme, with p bit significands, the leading 1 after significand addition can occur within a range of $2p$ bits. The reason for such behavior is simple. In this scheme, the significand of C is

aligned all the time irrespective of the relative magnitudes of the exponents of C and AB. With such a scheme, the leading zero estimation logic has to work with the $2p$ bit result and hence the normalization shifts are usually large. In Figure 8.12, the region of normalization shifts between -25 and -48 indicates situations during which a variable number of leading bits of the aligned significands gets cancelled in subtraction. Normalization shifts of 0 to -23 indicate correction shifting of the result of significand addition during situations when the exponent of C is greater than that of AB. With normal FADDs, this region is not relevant.

8.4.1 Significand alignment based power models

The time averaged power consumption of barrel shifter control lines is modeled by equation (7.7). With the IBM scheme, since bidirectional shifts are envisaged, 6 control signals are required for effecting alignment control. Also, the fanout of these lines are around twice that of the proposed scheme. With transition activity scaling, the activity factors of the shifter control lines are scaled down. While equation (7.7) reflects the power measure for effecting barrel shifter control, the switching activity within the significand data paths of barrel shifters is a function of the uncertainty between present shift and last shift. The switching activity measure of a single layer of data path nodes within the barrel shifter can be represented by

$$SW_{DP_i} = \left(1 + \frac{t_{D_i}}{w_i}\right) ESW_i \qquad \textbf{(8.7)}$$

where ESW_i represents the (time averaged) expected switching activity of the ith layer of barrel shifter data path under conditions when present shift is same as last shift, t_{Di} repre-

sents the toggle distance of this layer of nodes and *wi* represents the width of significand. including guard bits. The total data path switching activity measure of barrel shifters can be related to toggle distances by

$$SW_{DP} \propto \sum_{i=0}^{B-1} \left(1 + \frac{t_{Di}}{w_i}\right) \tag{8.8}$$

While the switching activity within the barrel shifter data paths increase with an increase in toggle distance; there exists yet another reason by virtue of which there can be a reduction as well. The effective number of nodes that endure transitions when the toggle distance is zero is equal to the total width minus expected shift; w_i - E[x_i]. With this, the switching activity measure given by equation (8.7) becomes

$$SW_{DPi} = \left(1 + \frac{t_{Di} - E[x_i]}{w_i}\right) ESW_i \tag{8.9}$$

The above is, however not applicable to barrel shifters that wrap around the shifted out bits for data rotation/bidirectional shifts, since the effective data path bits that endure transitions is always *wi*. With these schemes, the effect of alignment toggling affects both ends of the shifted significand, hence the increase in switching activity is proportional to $2t_{Di}$ instead of t_{Di} given in equations (8.7) (8.8) and (8.9).

8.4.2. Significand addition

With the IBM MAF, since the width of significand data path is around twice that of conventional schemes, a 2*p* bit CPA is required for the handling of significand additions. In

general, the wider the width of the adder, the higher the switching activity. Though this scheme envisages wider adders, the switching activity during normal additions is confined to a limited width, viz. the lower order p bits plus the average number of bit positions through which the significand of C is left shifted (expected left shift of the significand of C). The power consumption of this scheme can be represented by

$$P_{ADDI} = 2\left[1 + \frac{E[x_C]}{p} + t_S\right]P \qquad \textbf{(8.10)}$$

where $E[x_C]$ represents the expected left shift of the significand of C, t_s represents the probability that the sequence of operations toggles between {add, sub} and/or {sub, add}. P represents the average power consumption of a p bit significand adder under situations when the shift is zero. The scaling factor 2 is used to capture the power implications of the LZA, which is as complex as an adder as far as switching activity is concerned.

With the proposed scheme, the LZA circuits do operate only during a limited number of cycles. Since the LZA data path is dedicated for subtraction operations, there is no question of sign extension toggling within this data path. The power consumption of the significand adder segment of the proposed scheme is modeled by

$$P_{ADDII} = \left[2P(K) + \left(1 + \frac{E[x]t_S}{p}\right)(P(J) + P(L))\right]P \qquad \textbf{(8.11)}$$

where $P(K)$ and $P(J) + P(L)$ represent the probabilities that the LZA and LZB data paths are asserted. $E[x]$ represents the expected pre-alignment shift.

8.4.3. Partial product array

With the proposed scheme, the shifting of the multiplicand prior to partial product generation results in certain undesirable artifacts as far as the switching activity of the multiplier segment of MAF is concerned. In essence, the shifting of the multiplicand is equivalent to the shifting of all the partial products. Because of this, the effective switching activity of the multiplier array is increased. Such extra activity is expected with partial product generation as well as processing. With the proposed scheme, the switching activity of the multiplier bits are unaffected. However, with the multiplicand, the bits within the toggle range endure higher transitions than bits that are outside the toggle range. Assuming that the significands are independent, uniformly distributed[1] random variables, the bit level signal probabilities are 1/2. Under temporal independence assumption, the activity factors of the significand bits are 1/4. Under situations when the multiplicand is shifted, the multiplicand bit lines that were previously asserted high within the toggle range become zeros, by virtue of which the worst case activity factor of these lines is 1/2. The total increase in switching activity over all multiplicand bits that belong to the toggle range is

$$\Delta AF = \frac{t_D}{2} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (t_D \leq 4) \quad \text{and}$$

$$\Delta AF = \frac{(t_D + 4)}{4} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (t_D > 4) \qquad \textbf{(8.12)}$$

1. Statistical analysis of significand behavior however indicates that significands are reciprocally distributed [106] [107]. With reciprocally distributed significands, the distribution of the lower order bits exhibits maximum uncertainty and hence the distribution of these bits can be approximated into a uniform distribution. With the higher order bits, though the signal probabilities are less than 1/2, owing to the reciprocal distribution of significands, the assumption of uniform distribution is sufficient for reaching worst case estimates of extra switching activities.

With the IBM MAF scheme, the total switching activity measure for pre-alignment barrel shifter control and the set up of multiplier and multiplicand bits is given by

$$EM_I = EM_{BSI} + \frac{p}{2} \tag{8.13}$$

where EM_{BSI} represents the (fanout weighted) switching activity measure of the control lines of the pre-alignment barrel shifter while $p/2$ represents the total activity measure of multiplicand and multiplier bits. With the proposed architecture, the switching activity measure is given by

$$EM_{II} = EM_{BSII} + \frac{(2p + 2t_D + 4)}{4}(1 - P(I)) \quad \ldots\ldots\ldots\ldots\ldots\ldots (t_D \leq 4) \quad \text{and}$$

$$EM_{II} = EM_{BSII} + \frac{(2p + t_D + 8)}{4}(1 - P(I)) \quad \ldots\ldots\ldots\ldots\ldots\ldots (t_D > 4) \tag{8.14}$$

In the above equation EM_{BSII} represents the fanout weighted total activity measure for effecting pre-alignment shifts, t_D represents the expected toggle distance of the product and $P(I)$ represents the probability that the multiplier array of MAF is bypassed. With the proposed MAF, the activity scaled toggle distances are appreciably less than that of IBM MAF.

While the power implications of alignment control and set up of multiplier and multiplicand bits indicates a limited aspect of the effect of multiplicand shifting, the switching activity measures of the actual partial product bits however, capture the power implica-

tions of partial product processing. With conventional MACs/MAFs, since the significand multiplication is performed irrespective of the shift of product, the switching activity measure of this operation is rather fixed. Assuming uniformly distributed IID RVs as multiplier and multiplicand, the total switching activity measure of partial product bits is given by

$$EM_{PPI} = \frac{3p^2}{16} \tag{8.15}$$

Here, it is assumed that the partial product bits are formed by logically ANDing the relevant multiplier and multiplicand bits. With the proposed scheme, the switching activity measure is given by

$$EM_{PPII} = \frac{p}{16}(3p + 2t_D)(1 - P(I)) \quad \ldots\ldots\ldots\ldots\ldots\ldots (t_D \leq 4) \quad \text{and}$$

$$EM_{PPII} = \frac{p}{16}(3p + t_D + 4)(1 - P(I)) \quad \ldots\ldots\ldots\ldots\ldots\ldots (t_D > 4) \tag{8.16}$$

While the switching activity of bits within the toggle range is increased due to alignment toggling, there exists yet another dimension to such behavior. With the average shifts of the multiplicand exceeding the number of guard bits (in this case 4), the switching activity within the multiplier array is confined to a limited subset of the array. With large average shifts, because of this, the total switching activity within the multiplier array is reduced. For expected shifts exceeding 4 bits, the above equations are modified to

$$EM_{PPII} = \frac{p}{16}[3(p + 4 - E[x_{AB}]) + 2t_D](1 - P(I)) \quad \ldots\ldots\ldots\ldots\ldots\ldots (t_D \leq 4) \quad \text{and}$$

$$EM_{PPII} = \frac{p}{16}[3(p+4-E[x_{AB}])+t_D+4](1-P(I)).....(t_D>4) \quad \textbf{(8.17)}$$

where $E[x_{AB}]$ represents the expected shift of multiplicand.

While the equations presented in the preceding paragraphs capture the switching activity measures of multiplicand, multiplier and partial product bits; the usefulness of such activity measures, in isolation, is limited. In CMOS logic structures, the power implications of signal dynamics at circuit nodes is evaluated on the basis of fanout weighted switching activity measures. In the multiplier array, the fanouts of the multiplier and multiplicand bits are of the order of p. The activity factors of these lines are 1/4 while that of the partial product bits are 3/16. Assuming an average fanout[1] of 1.75 for every partial product bits, the fanout weighted switching activity measure of multiplicand and multiplier bits is 1.52 times that of the partial product bits. Also, the extra activity of the multiplicand bits within the toggle range is 4 times the extra activity of the relevant partial product bits. Considering the total fanout weighted activity for pre-alignment barrel shifter control and the setup of multiplicand and multiplier bits, the extra activity of the partial product bits becomes too insignificant, because of the large fanouts as well as high transition activities of the barrel shifter control/multiplier/multiplicand bits.

8.5. Results

Instrumented digital filter programs that envisage single precision FP operations, emulating the two MAF schemes had been developed. An assorted collection of bipolar audio

1. Assuming that 4:2 compressors of the type given in [125] are used for the post processing of partial product bits, the average fanout of partial product bits is 1.75.

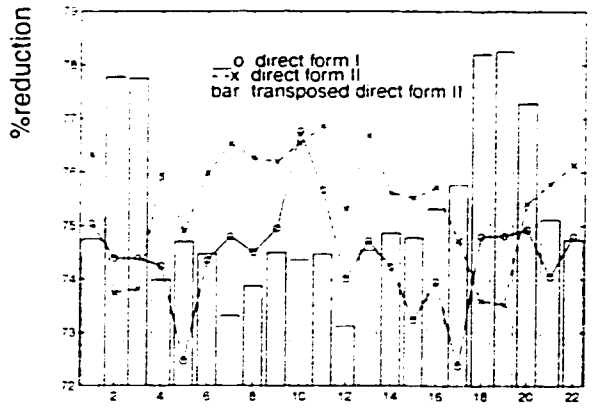


Fig. 8.14 - Reduction during IIR filtering

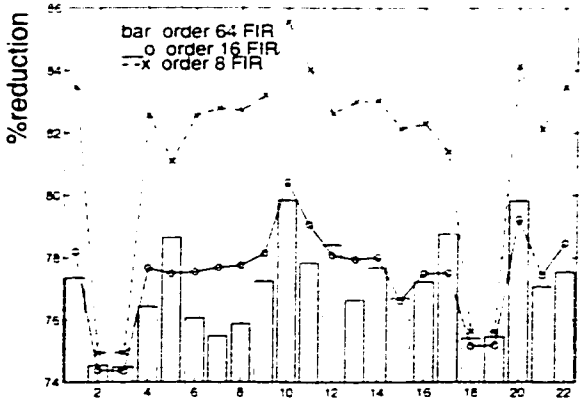


Fig. 8.15 - Reduction during FIR filtering

Activity reduction in barrel shifter control (pre-alignment)

signal samples ranging in size between 8594 and 6318040 samples had been low pass filtered using both FIR and IIR filters. The filter specifications of the IIR filter is the same. as discussed earlier. The samples had been filtered using direct form I. direct form II and transposed direct form II realizations. The normalized cut off frequency of all the three FIR filters are 0.2, while the orders are 64, 16 and 8 respectively. During the course of filtering. the frequency distributions of various data alignment operations, their rate of change and bit level activities of alignment control signals were gathered. The following paragraphs present the relative power implications of the proposed scheme. in contrast to that of the IBM MAF.

Figure 8.14 illustrates the percentage reduction in switching activity offered by the proposed architecture during IIR filtering of bipolar audio signals as far as significand pre-alignment control operation is concerned. Figure 8.15 illustrates the reduction observed during FIR filtering. The worst case reduction in switching activity, observed during architectural simulations outlined above, is better than 72%. With normalization control, the

proposed architecture offers yet higher reduction in switching activity: the reduction is always better than that of pre-alignments. With the transposed direct form II filter discussed above as a typical example, the proposed scheme offers an activity reduction of better than 10X. With all the experiments, the reduction in switching activity offered by the proposed scheme as far as normalization is concerned is always better than such figures for pre-alignment. With significand alignment operations, the toggle distances of the proposed scheme is insignificant in comparison to that of IBM MAF. The worst case reduction in switching activity (irrespective of data path activity scaling) as far as the toggling of the data path nodes of the barrel shifter is concerned is always better than 10X for both pre-alignment as well as normalization barrel shifters.

Figures 8.16 and 8.17 illustrate the experimentally observed activity reduction, offered by the proposed scheme as far as significand addition is concerned. As can be seen, the worst case reduction is better than 44%.

Figures 8.18 and 8.19 illustrate the estimated activity reduction offered by the proposed MAF taking into account the energy measures of pre-alignment control as well as the set up of multiplier and multiplicand bits. As can be seen, the worst case reduction is better than 47%. Figures 8.20 and 8.21 illustrate the percentage reduction in switching activity offered by the proposed scheme as far as the activities of partial product bits are concerned. The effect of MAF bypasses on activity reduction is clear from Figure 8.21. The order 64 and 16 FIR filters do exhibit MAF bypasses between 10 and 20%. However, with the IIR filters, the chances for bypass are negligibly small. Since the expected shifts as well as toggle distances of the transposed direct form IIR filter are large in comparison

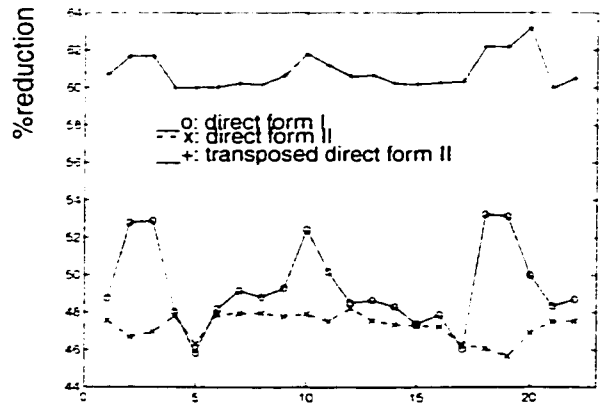


Fig. 8.16 - Reduction during IIR filtering

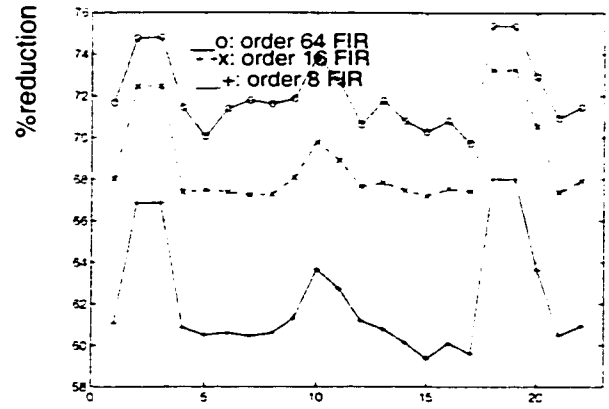


Fig. 8.17 - Reduction during FIR filtering

Activity reduction in significand addition

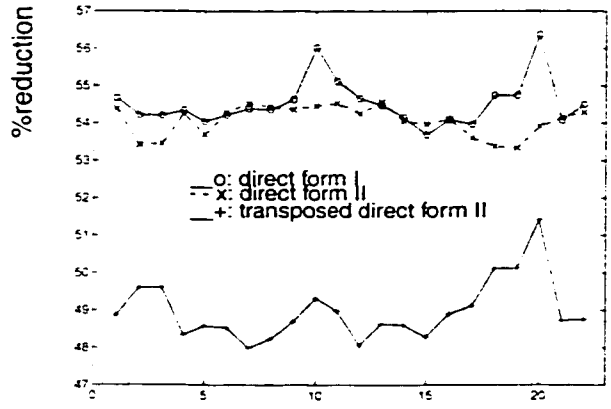


Fig. 8.18 - Reduction during IIR filtering

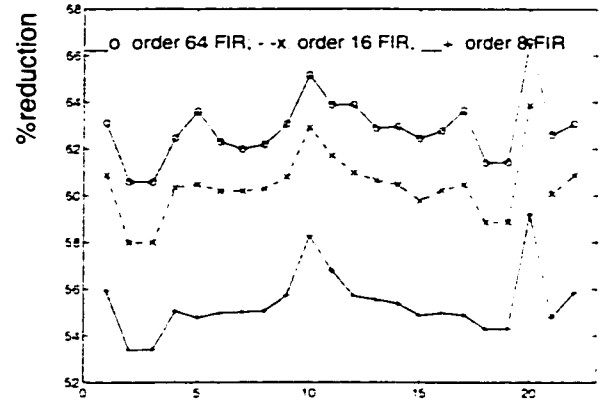


Fig. 8.19 - Reduction during FIR filtering

Combined activity reduction - pre-alignment control and set up of multiplier and multiplicand bits

with the width of significands, the extra activity of this scheme is the worst among all other cases, and the worst case increase in activity is less than 12%.

In general, the power consumption of pre-alignment barrel shifters accounts for the most dominating component of power consumption in FADDs and MAFs. Because of the large fanout of the barrel shifter control lines, the capacitive loading of these lines are large.

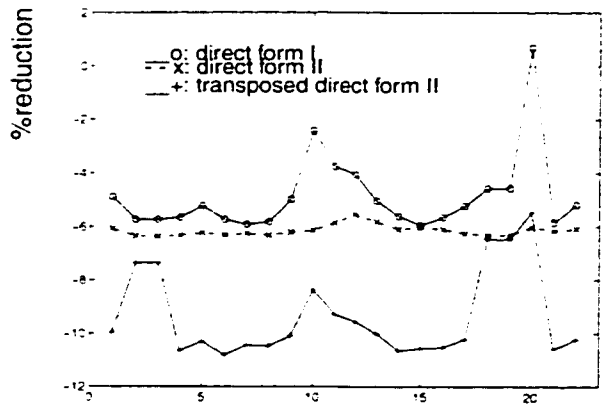


Fig.8. 20 - Reduction during IIR filtering

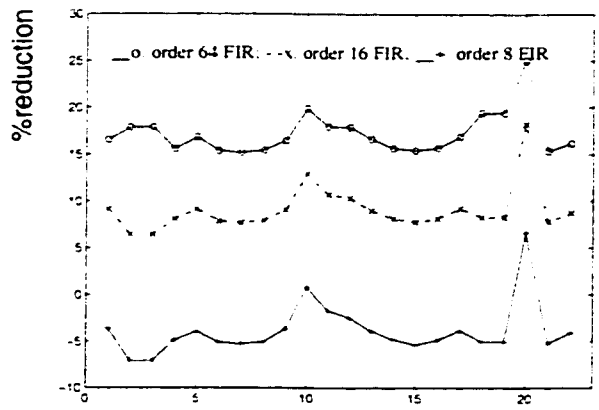


Fig.8. 21 - Reduction during FIR filtering

Activity reduction in multiplier array

Compared to normalization, the activity of pre-alignment barrel shifters is relatively large. The extra activity of the partial product bits affects the switching activities of the partial product compression circuits. However, with increasing logic depths, the relative increase in activity diminishes. Assuming that the fanout weighted activity measure of partial product bits as well as post processing circuits is 5 times that of such measure of partial product bits, the worst case power advantage offered by the proposed scheme is still around 49%. This figure had been arrived at by taking into account the fanout weighted activity measures of pre-alignment and normalization barrel shifter control lines, multiplier/multiplicand lines as well as partial product array. The worst case advantage projected above had been observed during filtering of white noise samples using the transposed direct form II IIR filter. With the filtering of bipolar audio samples also, similar figures are observed.

8.6. Discussion

So far, it had been assumed that the speed performance of IBM MAF and the proposed

scheme are comparable. Actually, with the proposed scheme, since the fanouts of barrel shifter control lines are half that of the IBM scheme, the set up of these signals are faster. This is true with normalization as well. Also, the process of significand addition is faster with the proposed scheme owing to the reduction in significand data path width, in contrast to that of IBM MAF. However, the evaluation of many of the activity scaling conditions may impose extra delay. While this extra delay is a definite limitation with certain applications, this can be easily addressed in instruction driven processors. With these applications, the pre-computation of certain control functions are possible during an early phase of instruction scheduling itself. With such schemes, it is also possible to operate the MAF in an activity scaled variable latency [25] mode (the MAF operates with the minimum latency during bypass conditions), so that the time averaged latency is less than the worst case latency.

While the question of MAF/FADD bypasses are not very relevant in a majority of computing situations, there however, exists a few DSP applications where this is not the case. As previously mentioned, with the order 64 and 16 FIR filters, the probability that the exponent of the product is negligible compared to that of the sum is between 10 and 20%. With symmetrical band pass and band stop filters (the normalized pass/stop bands centered at 0.5), the probability that the MAC operation produces a result that is not different from the current sum is around 47%. The above behavior is observed in both FIR and IIR filters, the only exception being Bessel filters. With bypass dominated filtering applications, the question of FADD/MAF bypasses arises due to the exponent quantization of filter co-efficients - the exponents of some of the filter co-efficients are too small compared to that of the rest. Because of this, during discrete convolution for filtering; the summation of prod-

uct terms whose magnitudes are relatively insignificant compared to that of the partial sums results in pre-alignment shifts that are larger than the width of significands. With MAC bypasses of the order of 47%, the proposed MAF gives a worst case power reduction of better than 72%.

8.7. Conclusion

The micro architecture and architectural power implications of a transition activity scaled MAF is presented. With filtering of audio signals as a test vehicle, the reduction in switching activity offered by the proposed architecture is verified. The proposed architecture is ideal for a variety of applications that doesn't require IEEE compliant floating point results. The worst case power reduction offered by the proposed scheme is better than 49%.

Chapter 9

Structural Power Implications of FP Additions During IIR Filtering

9.1. Introduction

During the recent years, with the advent of high speed floating point DSP ICs, more and more signal processing applications are migrating to the floating point domain. In DSP applications, filtering of data samples is one of the most demanding operations. Given a set of filter specifications satisfying certain signal processing requirements, with the help of modern CAD tools, the designer can rapidly delineate an appropriate filter. While the filter transfer function obtained through the above exercise does remain the same, different implementations of the same transfer function are possible [128]. For example, IIR filters can be implemented as direct form, canonical form, transposed canonical form and cascade realizations. Compared to direct form realizations of IIR filters, the memory requirements for canonical realizations are less. Other than memory optimization, are there any other factors that make certain filter realizations more attractive compared to others? In VLSI system design, the power consumption and speed performance of functional units are all the more important. Even realizations having essentially the same memory requirements and speed performance can have vastly different power implications. This chapter targets characterization of FP additions in floating point IIR filters on the basis of their structure driven power implications.

In low power CMOS VLSI system design, an early exploration for algorithmic/architec-

tural power minimization is important since the scale of power reduction achievable through such approaches is relatively significant [2] [122]. Higher level power estimation and modelling are essential for the validation of design approaches for algorithmic/architectural power minimization. Though gate level power estimation of CMOS circuits is a relatively well studied problem [74], the same is not true as far as higher level power estimation (especially at the system/algorithmic level) is concerned. Recent thoughts on higher level power estimation and modelling are reported in [111] [129] [130], [131] and [132]. In [111], a higher level analytical power estimation method involving word level statistics is suggested, while [131] and [132] provide entropy based higher level characterizations. Though in [111] an analytical characterization of arithmetic building blocks for signal processing applications had been achieved, [133] and [134] reflect more specific work in this area. In [133], S. R. Powell et al. reported a higher level power dissipation model for filter and transform type DSP algorithms that are implemented through a set of linearly connected multiply - add (MAC) based processing elements. J. S. Ward et al. [134] developed technology independent cost functions for the characterization of different DSP algorithms (as well as different implementations of the same algorithm). While [133] and [134] attempted the characterization of embedded fixed point DSP algorithms, our approach is different. As previously mentioned, our aim is the characterization of the structural power implications of floating point IIR filters. The other difference is: our study is directed more towards programmable DSP applications, though some of the findings of this work are useful in architectural design of floating point DSP cores/systems. An abridged version of this work had been reported in [110]. The rest of the chapter is organized in the following way. Section 2 introduces the problem definition. Section 3 presents

the development of significand alignment driven power models of FADDs. Sections 4 outlines the simulation approach. Section 5 presents the results. Section 6 presents a discussion on the power implications of the exponent quantization of filter co-efficients, memory requirements and structure driven rounding implications. Section 7 concludes the chapter.

9.2. Problem definition

In floating point DSP applications, the magnitude and rate of data alignment operations involving significands of floating point operands for addition during multiply - accumulate (MAC) procedure, do dictate the power implications of these applications. For example, if the magnitude of significand alignment shifts as well as its rate of change are relatively small in comparison to the significand width, then the power requirements for effecting such shifts is relatively smaller. Apart from this, the power dissipation due to alignment driven switching of significand data path bits is also less. Intuitively, DSP algorithms that minimize the above factors - alignment shifts and (time) rate of change of shifts - guarantee power minimal realizations. In IIR filter realizations, owing to the presence of structure dependant repeated summation of floating point operands, we have reasons to believe that the structure driven sequence of arithmetic operations influences the magnitude correlation of floating point operands of addition during MAC operations. Figure 9.1 illustrate the signal flow graph representations of an 8th order IIR filter implemented as direct form I, direct form II and transposed direct form II realizations.

In direct form I and II realizations, the MAC operations are distributed over two separate loops. Within each of these loops, the co-efficients involved in multiplication belong to the same set, viz. forward or feedback co-efficients. For the transposed direct form II realiza-

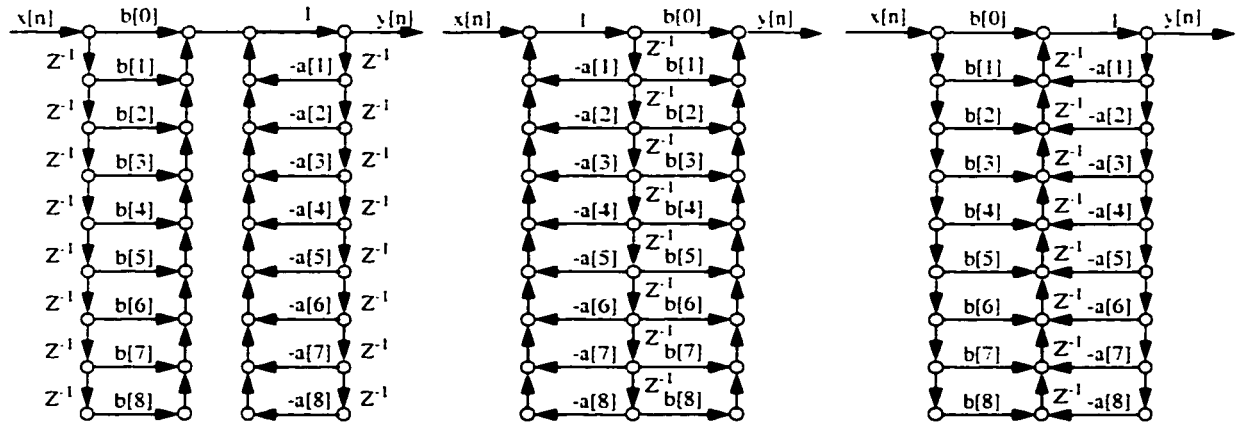


Figure 9.1 - Signal flow graph representation of IIR filters

tion, most of the adder nodes handle a three operand addition. Out of the three operands, two of the operands are the scaled input/output samples while the third is a memory variable. During situations when the magnitude difference between the feedback and forward co-efficients are relatively large (while the input/output samples doesn't exhibit a compensating magnitude variation), the difference between the exponents of the operands can be relatively large. However, during situations when the magnitudes of feedback/forward co-efficients are close to each other or the difference between the magnitudes are compensated by a proportionate difference between the input/output samples, the exponent differences can be smaller. In other words, the significand pre-alignment shifts (and hence the rate of change of shifts) of the transposed direct form II realization exhibits sensitivities towards the relative magnitudes of forward/feedback co-efficients as well as input/output samples. In the case of direct form realizations, the exponent distribution of the same set of co-efficients - i.e., feedback or forward co-efficients - dictates such behavior.

Based on the above arguments, we hypothesize the following. Given an IIR filter design,

realizations that produce relatively well correlated (magnitude correlation) operands of FP addition results in power optimal implementations. Having laid down such a hypothesis, our problem definition is: A study of the algorithmic/structure driven power implications of FP additions in IIR filters.

9.3. Characterization of control/data path switching

Since the central theme of this study is the characterization of the power implications of additions in IIR filters, the following assumptions are valid. The power consumption of the FP multiplier is comparable for the different realizations. The power consumptions of barrel shifters and significand adders/leading zero anticipatory logic are affected by the sequence of arithmetic operations. Since the number of arithmetic operations are the same in all the realizations, the speed performance of various realizations are the same.

The time averaged power consumption of barrel shifter control operation is given by equation (7.7) while the control path component of the power measure is represented by equation (7.8).The alignment driven data path switching is modeled by equation (7.12). These models reflect the significand alignment driven power implications of conventional FADDs. In our architectural model of barrel shifters, the shifting operation is confined between 0 and 31 bits, so that 5 of the LSB bits of the relevant exponent difference effects the requisite data alignment operations. The barrel shifter contains $\log_2 p$ (rounded to the next higher integer) shifter stages. While there exists exponent differences even greater than 31 during the course of FP additions, since the shifting is confined within limits 0 - 31; the expected values of shifts as well as toggle distances reflect shifts within these limits.

9.4. Experimental approach

Single precision instrumented IIR filter programs, emulating a DSP multiply - accumulator (MAC), had been developed for the various realizations. For the present simulation based study, single precision MAC models had been chosen, owing to the proliferation of real time DSP applications using single precision floating point DSPs; so that the results of this study are immediately applicable to a variety of practical signal processing applications. Because of the recent surge in popularity of DSP applications for the processing of audio signals, we selected audio signal processing as a test vehicle. In our DSP model, the FP multiplier and FP adder building blocks are treated as separate entities.

The first set of simulations encompasses low pass filtering of data samples, by using an 8th order elliptical filter (pass band ripple 0.03 dB, stop band ripple -100 dB and cut-off frequency 0.2 dB). Ref. [112] lists an assorted collection of audio signal samples that had been filtered using the instrumented programs. Apart from audio signal samples, the behavior of FP additions during filtering of synthetic data samples (in particular white noise and AR(1) signals) also had been validated. The AR(1) - I signal is $g[n] = f[n] + 0.9*g[n - 1]$ while AR(1) - II is $g[n] = f[n] + 0.5*g[n - 1]$. Each of the signals had been of sample size 128K. In order to verify the impact of filter specifications on the alignment driven power behavior of FADDs, white noise samples had been filtered using elliptical low pass filters of the same pass band specification given above, but different stop band ripples (-140dB to -50dB). To validate the repeatability of results with filters that are not low pass, the experiments were performed at random using band pass, band stop and high pass filters. In this set of experiments also, both audio samples as well as synthetic data had been filtered. During the course of filtering, frequency distributions of pre-alignment

and normalization shifts, their rate of change and the relevant bit level activities as well as the transition between addition and subtraction operations had been gathered.

The second set of simulations comprise band pass/band stop filtering of white noise samples using elliptical filters, the normalized pass band centre frequency of which are 0.5 (pass band 0.4 - 0.6; pass band ripple 0.03 dB; stop band ripple -100 dB). The power behavior of this category of filters is of special interest, owing to the special nature of the exponent quantization of filter co-efficients. With this type of filters, the exponents of adjacent filter co-efficients exhibit relatively large differences - that are actually more than the width of single precision significands. With precision limited arithmetic, whenever the difference between the exponents of operands are more than the width of significand, an FP addition no longer produces a result that is different from the larger operand. Because of this, the arithmetic efficiency of FP additions of these filters are less than 50%. The impact of filter specification changes also had been studied. The behavior of this family of filters with Chebyshew and Butterworth filters also had been verified.

9.5. Results

The following paragraphs present the relative power implications of the different filter realizations that had been experimentally observed during filtering of audio signals as well as synthetic data. The first section of results pertains to low pass filtering. In the second section we present the results of band pass/stop filters having normalized centre frequencies of 0.5.

9.5.1. Low pass filters

Table 9.1 lists the switching activity reduction offered by the direct form I filter in comparison with transposed direct form II realization, that had been experimentally observed during filtering of synthetic data. The reduction in control path switching of pre-alignment and normalization barrel shifter control lines as well the total reduction in control path switching are listed. To characterize the data path switching activity, reduction in parameters α and β as well as the total reduction in data path switching are listed. Table 9.2 lists the corresponding reduction in parameters that is offered by the direct form II realization, in contrast to that of transposed direct form II realization. The direct form I realization offers a reduction of better than 29% as far as the power consumption due to control path switchings are concerned, while the corresponding reduction in data path switching component is better than 26%. In contrast to transposed direct form II realization, the canonical form (direct form II realization) offers better than 33% reduction in control path switching component. The reduction in alignment driven data path switchings are better than 29%. With this, it is clear that the direct form realizations offers worst case power reductions that are better than 25% as far as power implications of FP add operations during IIR filtering are concerned.

Table 9.1: Percentage reduction in switching activity - direct form I

Signal	Control path switching			Data path switching		
	Pre-alignment	Normalization	Total	α component	β component	Total
N(0,1) IID RVs	39.44	-6.73	29.60	65.60	69.21	26.43
AR(1) - I	43.93	13.16	35.72	69.54	82.53	30.24
AR(1) - II	41.12	1.13	31.99	67.03	74.36	28.14

Table 9.2: Percentage reduction in switching activity - direct form II

Signal	Control path switching			Data path switching		
	Pre-alignment	Normalization	Total	α component	β component	Total
N(0,1) IID RVs	44.00	-4.57	33.44	75.57	73.45	29.71
AR(1) - I	47.54	7.64	37.59	77.75	79.74	32.25
AR(1) - II	44.58	-2.67	34.08	76.56	74.95	30.93

From the above results it is clear that during low pass filtering, direct form realizations offers significant reduction as far as both data/control path switching dominated power consumption of FADDs are concerned. Figure 9.2 illustrates the percentage reduction in switching activity of pre-alignment barrel shifter control offered by the direct form realizations (in contrast to the transposed realization) during filtering of various audio signal samples [112] (22 samples). As is evident, direct form I implementation offers a worst case reduction of better than 38% while the corresponding figure for direct form II is better than 41%. Figure 9.3 illustrates the case with normalization. During filtering of certain signals, the direct form realizations impart more switching activity than transposed realization. The reason for such a behavior is the following. Since the expected shift of pre - alignment is relatively large for the transposed realization, the chances for the generation of a variable number of leading zeros (exponent difference ≤ 1 and subtraction) during significand addition are comparatively less. By virtue of having fewer chances for the generation of a variable number of leading zeros, the traffic of normalization shifts involving large shifts (as well as the rate of change of normalization shifts) is relatively low. Because of this, with the transposed realization, the power requirements for effecting normalization

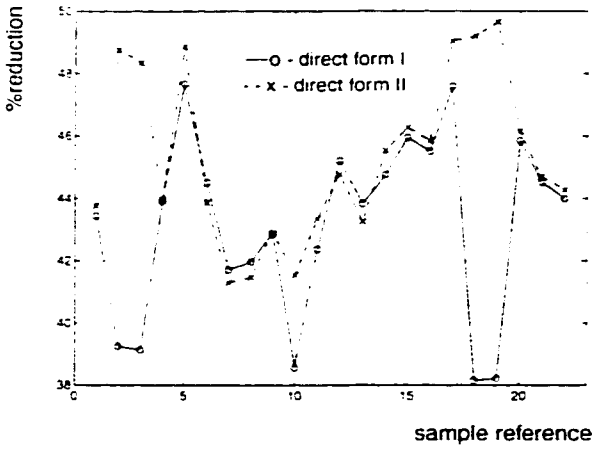


Fig. 9.2 - Activity reduction in pre-alignment barrel shifter control

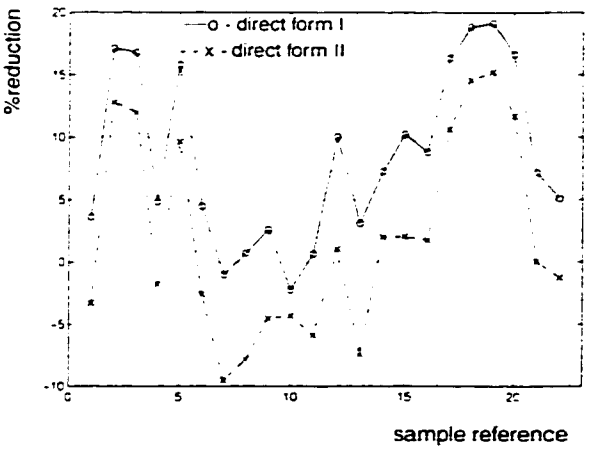


Fig. 9.3 - Activity reduction in normalization barrel shifter control

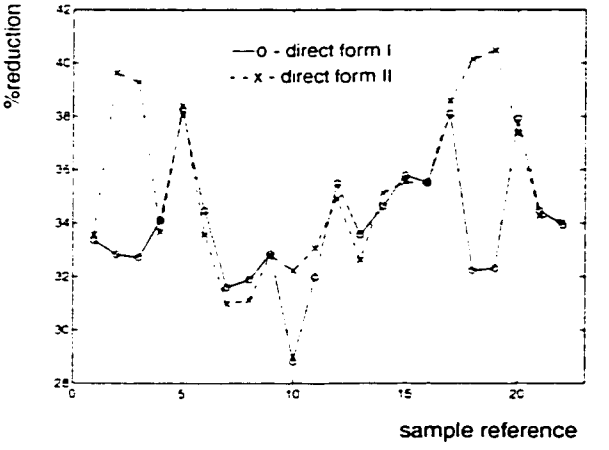


Fig. 9.4 - Reduction in control path switching (22 samples)

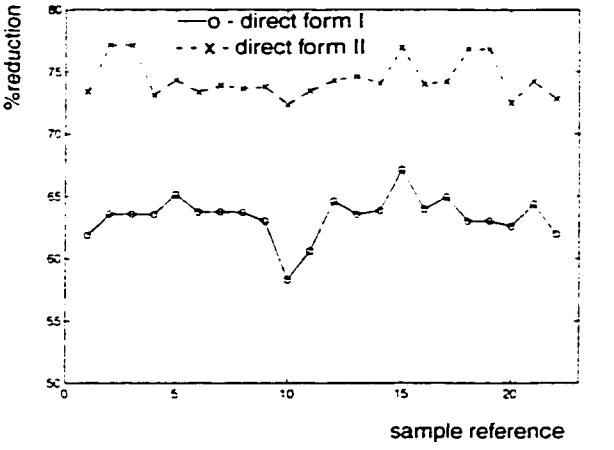


Fig. 9.5 - Reduction in alignment driven data path toggling

shifts are less than that of the direct form realizations. However, this behavior - to a large extent - is governed by the temporal correlation characteristics of input signals as well. From the power reduction characteristics shown in Figure 9.3, this is evident. With nearly 50% of the signals, the power implications of transposed realization is worse than that of direct form realizations.

In contrast to direct form II and transposed direct form II, direct form I realizations are

clearly the winners as far as power implications of normalization shifts are concerned. Though the power advantage offered by the direct form realizations are not so good in this case, the direct form realizations are still attractive owing to the dominance of pre-alignment operations in draining a substantially larger power than normalization. Figure 9.4 illustrates the overall control path power advantage offered by the direct form realizations taking into account the power implications of pre-alignment as well as normalization barrel shifters and the FADD bypass control signal. The direct form I realization offers a power reduction of better than 28% while the corresponding figure for direct form II is better than 30%.

Figure 9.5 illustrates the percentage reduction in power consumption due to the alignment driven toggling of significand data path. The worst case reduction offered by the direct form I realization is better than 57%. The reduction offered by the direct form II realization is consistently above 71%. Figure 9.6 illustrates the power advantage offered by the direct form realizations as far as power implications due to alignment driven/sign toggling dominated data path switching activity is concerned. With direct form I, the worst case advantage is better than 63%. The direct form II offers a worst case reduction of better than 72%. Figure 9.7 highlights the overall reduction in data path switching component of the power consumption of FADD. The worst case reduction offered by the direct form I filter is better than 24% while the corresponding figure for direct form II is better than 29%. Taking into account the reduction in data and control path switchings (and taking the lower figure for worst case analysis), the worst case power reduction offered by direct form I realization can be better than 24%. With direct form II, this figure is around 29%.

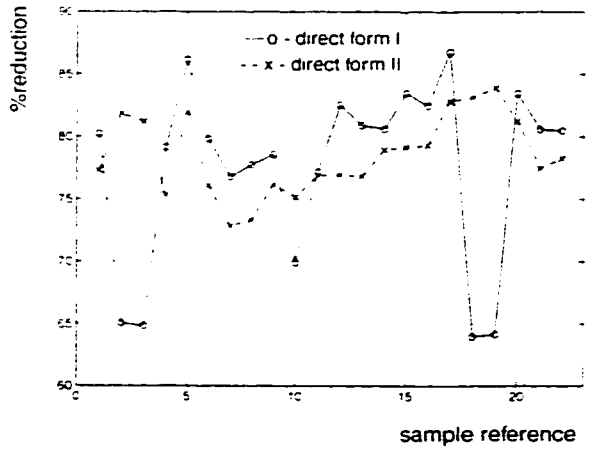


Fig. 9.6 - Reduction in alignment driven/sign toggling dominated data path switching

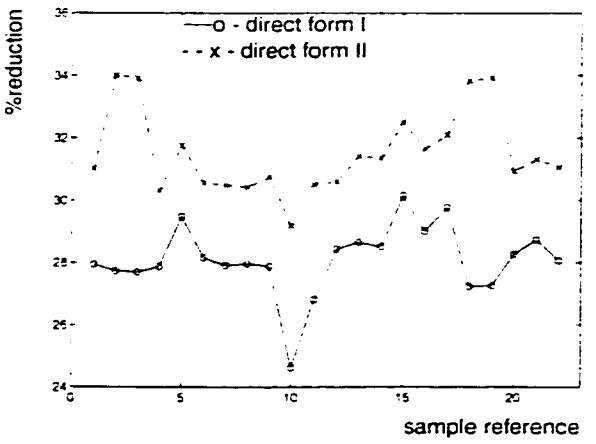


Fig. 9.7 - Reduction in data path switching (22 samples)

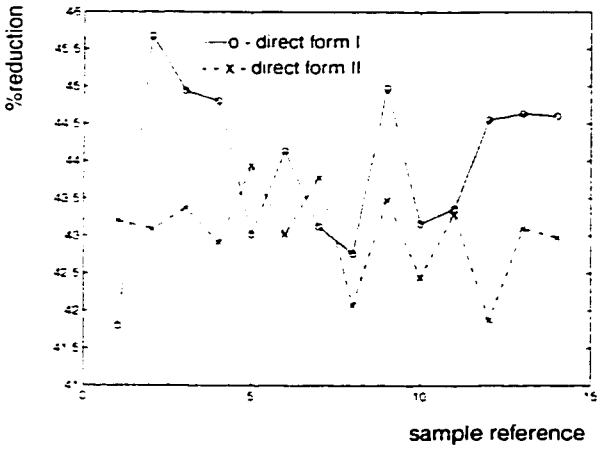


Fig. 9.8 - Reduction in control path switching (14 samples)

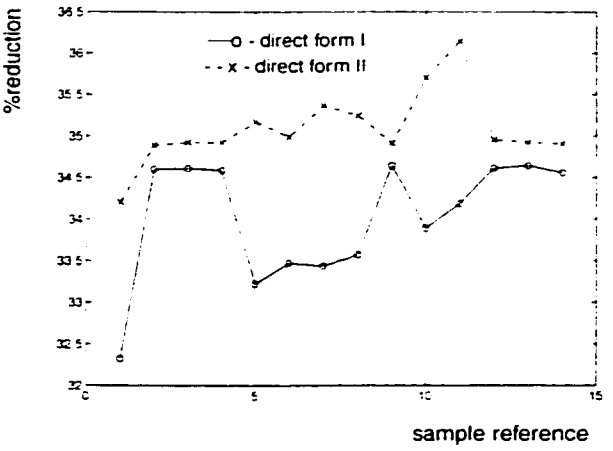


Fig. 9.9 - Reduction in data path switching (14 samples)

In order to validate the repeatability of results, the experiments were repeated with a family of different audio signal samples [112] (14 signals, ranging in size 129264 - 8851456 samples), maintaining the same filter specification. Figures 9.8 and 9.9 illustrate the reduction on control and data path switching activities. From Figure 9.8, it is clear that the worst case reduction in control path switching offered by the direct form I realizations are better than 41%. The corresponding figure for data path switchings are better than 32%. Evidently, the significand alignment driven power consumption of the floating point adder segment of the DSP ALU is appreciably greater while running the transposed canonical

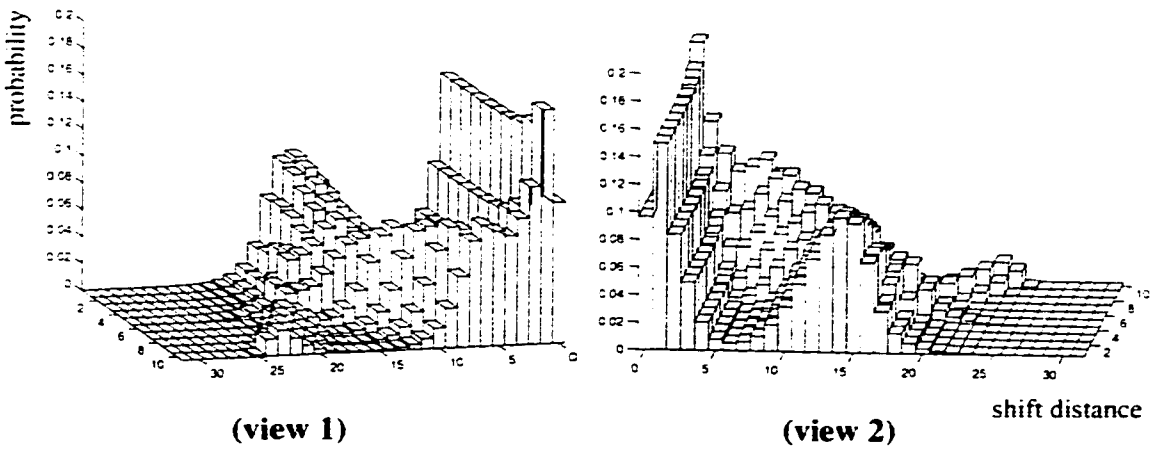


Figure 9.10 - Pre-alignment behavior of transposed direct form II IIR filters

form of filters, in contrast to canonical forms. From the experiments, we also found that the uncertainty with respect to sign toggling of the leading bits of aligned significands is always around 0.5 for the transposed direct form II algorithm while this factor can be as low as 0.25 for the direct form counterparts.

In order to highlight the effect of filter specifications on the significand alignment behavior of FADDs, we present the following: White noise samples had been filtered by using low pass elliptical filters of the same cut off frequency (0.2) and pass band ripple (0.03dB), but stop band ripples of -140dB, -130dB, -120dB, -110dB, -100dB, -90dB, -80dB, -70dB, -60dB and -50dB respectively. Figure 9.10 illustrates the pre-alignment behavior of FADDs during IIR filtering (transposed direct form II) of white noise samples, for different filter specifications (entries 1 to 10 in the z axis). Figures 9.11 and 9.12 illustrate the impact of filter specifications on control/data path power advantages offered by the direct form filter realizations during low pass filtering of white noise samples. As can be seen from Figure

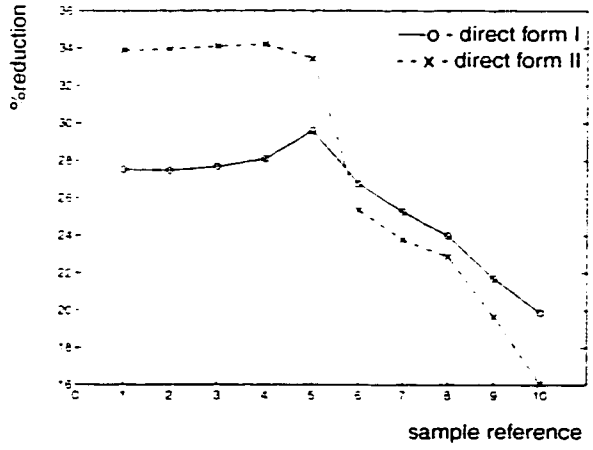


Fig. 9.11 - Reduction in control path switching - LPFs

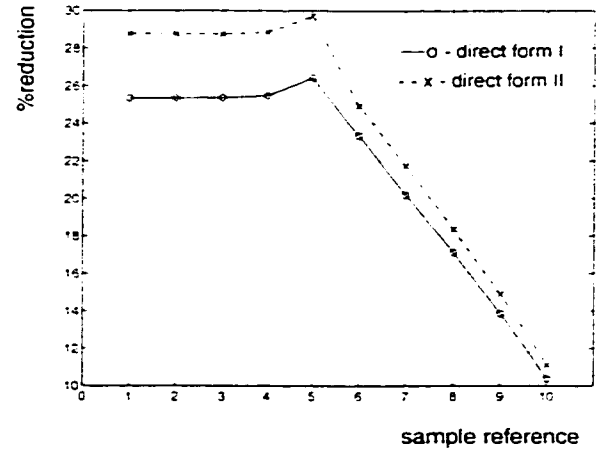


Fig. 9.12 - Reduction in data path switching - LPFs

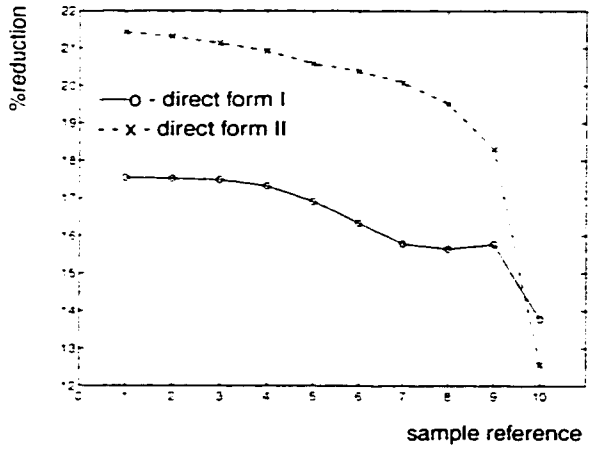


Fig. 9.13 - Reduction in control path switching - BPFs

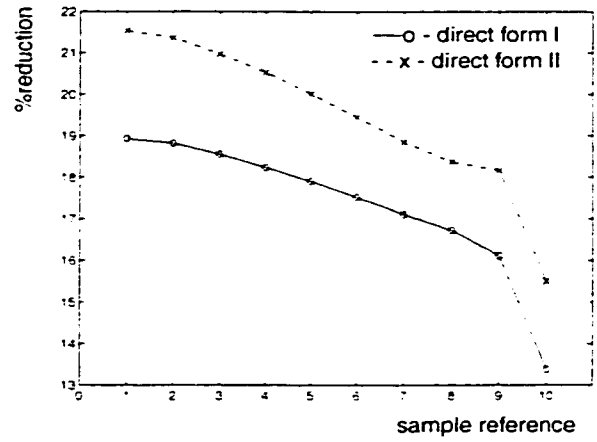


Fig. 9.14 - Reduction in data path switching - BPFs

9.10. the tighter the filter specification, the more the spread in the distribution of shift distances. With relaxed specifications, the 2nd mode of the pdf eventually merges with the first. The higher the variance of the pdf of shift distances, the higher the power consumption of FADDs. With relaxed filter specifications, the power advantage offered by the direct form filters are less. Actually, in such cases, lower order filters of tighter specifications can deliver the desired response. That means, with optimally designed filters, the power advantages offered by the direct form filters remains more or less the same.

The power behavior highlighted so far is in good agreement with other types of low pass filters as well. Figures 9.13 and 9.14 illustrate the power advantages offered by the direct form filters during band pass filtering of white noise samples. White noise samples (128K) had been band pass filtered using 16th order elliptical filters of pass band ripple 0.03dB, normalized pass band of 0.2 to 0.6 and varying stop band ripples of -140dB to -50dB (entries 1 to 10 in the x axis). Here again, with relaxed filter specifications, the power advantage reduces. With optimally designed filters, however, the advantage remains. While the results presented so far deals with low pass and band pass filters, the power implications of high pass filters are also more or less the same and the results are applicable to other filter families as well, i.e., Chebyshev, Butterworth etc.

9.5.2. Arithmetically sub-optimal filters

The results presented so far pertains to filters that are more or less optimally implemented within the limitations of floating point arithmetic. In other words, in these filters, the question of pre-alignment shifts exceeding the widths of significands had been marginally low. However, there exists certain class of filters for which this is not the case. Band pass/band stop filters whose centre frequency is 0.5 is such an example. Elliptical, Chebyshew and Butterworth (even FIR filters) filters all experience the effect of precision limited arithmetic for this case. As discussed previously, with this family of filters, the difference between the exponents of alternate filter co-efficients are relatively large in comparison with the width of significands (single precision). While the alternate filter co-efficients of the same set, i.e., forward/feedback co-efficients, are not well correlated in terms of their magnitude; the correlation between the corresponding members of the forward/feedback co-efficients is very high. With such cases, the transposed direct form II is power optimal.

Actually, the whole class of filters are sub optimal in terms of the effective utilization of precision limited floating point arithmetic.

In order to validate the power behavior of filters that are arithmetically sub optimal, white noise samples had been filtered using band pass/band stop filters. Here again, we used 8th order elliptical filters, the specifications of which are: normalized pass/stop bands 0.4 - 0.6 and pass band ripple - 0.03dB. The stop band ripple of these filters are -125dB, -115dB, -105dB, -95dB, -85dB, -75dB, -65dB, -55dB, -45dB and -35dB. Figures 9.15 and 9.16 illustrate the relative power implications of the band pass filter. Here, for the sake of maintaining uniformity with the results presented so far, the relative power advantages of direct form realizations are presented. In this particular case, the advantage is negative, which means, the transposed realization is power optimal. Figures 9.17 and 9.18 illustrate the power behavior of band stop filters described above. Here again, the transposed direct form II is power optimal.

9.6. Discussion

While the results presented so far doesn't reflect an exhaustive simulation base, however, these results do substantiate the validity of the observation - that the power consumption of FADDs of DSPs exhibits sensitivities towards the structure of algorithms. Though the experimental evaluation and characterization of such behavior demands special tools (instrumented filter programs); a first level evaluation of the power bearing of different filter realizations can be achieved with current state-of-the-art DSP tools. In general, once the exponent quantization of the filer co-efficients are known, a first level evaluation is relatively straight forward. With this approach, the elimination of bad design choices is quite

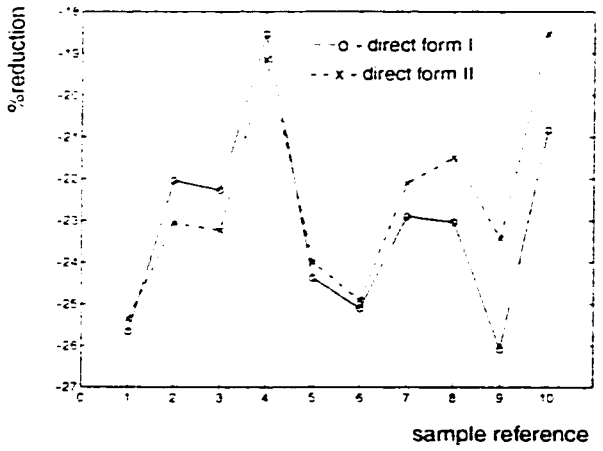


Fig. 9.15 - Reduction in control path switching - BPFs (special case)

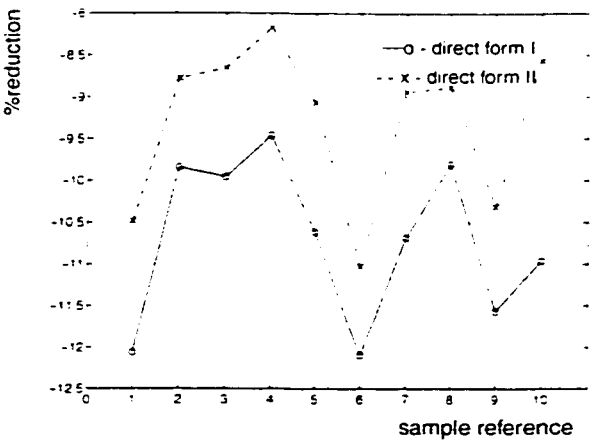


Fig. 9.16 - Reduction in data path switching - BPFs (special case)

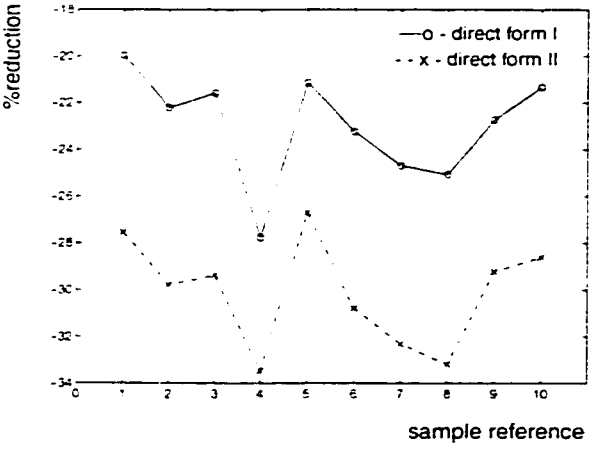


Fig. 9.17 - Reduction in control path switching - BSFs (special case)

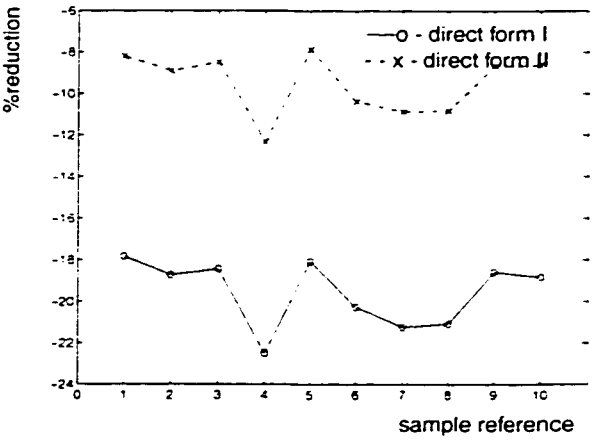


Fig. 9.18 - Reduction in data path switching - BSFs (special case)

rapid. Since the design for power efficient operation is all the more important, an early exploration/analysis of the type presented above for power minimization is rewarding.

So far, the discussions were mainly focussed towards the power implications of the adder segment of floating point DSPs. From a system level perspective, the memory requirements are also important. In case of memory restrictions, direct form II is the preferred choice, which delivers a power optimal and memory optimal solution. Since the number of

arithmetic operations are the same in all realizations, the speed performance can be maintained equal. With modern DSPs, the availability of on chip RAMs/buffers or register files is no real problem. That means, even direct form I realizations are also viable.

The direct form I realization is attractive from yet another perspective. In floating point operations, the generation and accumulation of rounding noise is a function of the magnitudes of operands. For example, a multiplication operation involving operands A and B is realized in a precision limited floating point environment as [127] $fl(AB) = AB(1+\delta)$. In this equation, $fl(AB)$ implies the result of precision limited multiplication of numbers A and B. δ is a random variable that is uniformly distributed in the interval $\pm$ 1/2ulp (unit in the last place). Similarly $A + B$ is realized as $fl(A+B) = (A+B)(1+\varepsilon)$. ε is also uniformly distributed in the interval $\pm$ 1/2ulp. From this, it is clear that the higher the magnitude of product/sum, the higher the rounding error. In floating point DSP applications, minimization of rounding noise essentially demands a minimization of sum/product. During low pass filtering, the rounding errors are the least with direct form I realizations. This is true with most of the other types of filters as well. With direct form II realizations, the effect of rounding noise is the worst (irrespective of the filter type). This observation agrees with the result that had been reported by B. D. Rao [127]. The rounding error due to multiplications is the same for direct form I and transposed direct form II realizations. In the case of precision limited band stop filters, the rounding noise due to floating point addition is the least with transposed direct form II realization.

9.7. Conclusion

The algorithmic/structural power implications of FP additions during floating point IIR fil-

tering is presented. Compared to transposed direct form II realizations, direct form realizations offer reduced power operation, during most of the filtering situations. In addition to reduced power operation, the direct form I realization also offers the least rounding noise. With low pass filtering of audio signals, the worst case power reduction offered by the direct form realizations had been found to be better than 24%. Direct form II offers power and memory optimal realizations. With band pass/band stop filters with a normalized centre frequency of 0.5, transposed direct form II realizations are power optimal.

Chapter 10

Conclusions and Future Work

10.1. Preamble

Contributions towards the development of high level design decisions for power/performance optimal realization of floating point adders, accumulators, multipliers and multiply - accumulators are presented in the preceding chapters. The mapping of arithmetic operations into distinct clock gated functional units allows activity reduction as well as throughput enhancements /latency reduction. Validation of the design decisions are accomplished through architectural simulations and analysis. Analytical models describing the higher level power/delay behavior of the floating point elements are developed. Listed below are the salient contributions to the knowledge of present state - of - the art in floating point DSP data path design.

10.1. Contributions to the state - of - the - art

Characterized the behavior of exponents/exponent differences that are encountered during repeated addition of IID RVs and matrix computations; developed analytical models on the distribution of exponent differences during accumulation of IID RVs. The models are verified through simulations. The non stationary behavior of exponents/ exponent differences encountered in repeated summation is established through theoretical as well as experimental reasoning. The findings are reported in [109].

The applicability of arithmetic structures based on 1's complement arithmetic for the

implementation of low power floating point units is validated. These units evaluate operations of the type $| A - B |$ involving integer data A and B in the same time, using half the hardware; in contrast to schemes that use 2's complement arithmetic. The inherent data comparison measure of the 1's complement based scheme makes it all the more attractive. With 1's complement based conditional sum adders, evaluation of functions like $A > B$, $A < B$, $A = B$, $A \leq B$, $A \geq B$, $A \neq B$ are virtually free. Compared to 2's complement based schemes, the power advantage of the 1's complement scheme is around 50%. The results are reported in [124].

The power/performance desirability of an optimal partial product reduction strategy, originally suggested by D. Villeger et al. [18], for parallel multipliers is established. Through analytical modeling and circuit simulations, it is demonstrated that partial product generation using simpler AND logic is advantageous in terms of power, delay and layout regularity in contrast to schemes using modified Booth algorithm. The findings are reported in [125].

Developed a low power, transition activity scaled organization for the implementation of barrel shifters. Analytical power/delay models that characterize the behavior of barrel shifters is developed [123]. The proposed scheme had been evaluated through architectural simulations and analytical reasoning [112]. During filtering of an assorted collection of audio signal samples, the experimentally observed worst case power reduction offered by the proposed scheme (as far as pre-alignment of significands are concerned) is better than 27%. The power/delay advantages of the proposed scheme renders it an ideal choice for the implementation of low power floating point units.

Developed a low power transition activity scaled triple data path architecture for floating point additions [98]. The proposed scheme, in addition to power reduction due to transition activity

scaling, also offers faster operation due to data path simplifications and speculative rounding. Developed high level power/delay models that describe the alignment driven floating point additions [112] [140] [135]. Validated the proposed architecture through architectural simulations. In contrast to comparable schemes that are recently reported in literature, the new scheme offers a worst case power reduction of around 36%. With the proposed transition activity scaled triple data path floating point as a core, a low power scheme for floating point accumulation is developed. The results are reported in [137].

Evaluated the behavior of floating point additions during digital filtering that are characterized by arithmetically sub-optimal, precision limited discrete convolutions [112] [138]. The notion of FADD bypass in practical DSP applications is verified. As an interesting artifact, the transformation of linear phase FIR filters into a non linear phase version due to limitations of arithmetic precision is demonstrated. During multiply - accumulate operations for the realization of symmetrical band pass/stop filters, the probability that FADDs endure bypass conditions are greater than 50%. The proposed triple data path FADD core offers a worst case power advantage that is grater than 75% during such situations.

Developed a new, low power architecture for floating point multiply - accumulate fusion. The power behavior of the scheme had been validated through architectural simulations and modeling, which substantiate the architectural portioning criterion. During filtering of signals using symmetrical band pass/stop FIR/IIR filters (except Bessel filter), the probability that the entire MAC can be bypassed is around 50% (which translates into a 72% power reduction). Developed analytical models that describe the significand alignment driven power behavior of the proposed architecture.

Characterized the structural power implications of floating point additions during IIR filtering [110] [140] [141]. Based on analytical modeling and simulations, the desirability of direct form realizations of IIR filters for power optimal behavior is established: the worst case power advantage the direct form realizations do offer is better than 24%. The simulations also substantiate the conclusion from other researchers [127] that direct form I realization is more desirable as far as rounding noise minimization is concerned.

10.2. Future work

The higher level design methodologies developed in this work finds applications for CAD tool development. Automatic synthesis of floating point data paths is one area where these methodologies can be immediately applied. Development of VHDL behavioral descriptions is a good starting point. ASIC implementation of the FP units is another interesting problem.

The methodologies developed in this work, however forms a strong foundation for higher level architectural/system design. In particular, the philosophical reasonings of transition activity scaling is of immense desirability in system design. For example, with future processors/systems, it is appropriate to think in terms of instruction driven transition activity scaling. With the control units absorbing the responsibility of the management of transition activity scaling, the scale of power reductions achievable can be very high.

References

[1] Lisa A Coleman. "DSPs break through technology road blocks." *Military and Aerospace Electronics*. pp. 24 - 28. April 1994.

[2] Kurt Keutzer and Peter Vanbekbergen. "The Impact of CAD on the Design of Low Power Digital Circuits." in *Proceedings of the 1994 IEEE Symposium on Low Power Electronics*. pp. 42 - 45. 1994.

[3] Erdem Hokenek. Robert K. Montoye and Peter W. Cook. "Second Generation RISC Floating Point with Multiply - Add fused", *IEEE Journal of Solid State Circuits*. Vol. 15. pp. 1207 - 1213. October 1990.

[4] Coonen. J. T.. "Contributions to a proposed standard for binary floating point arithmetic". *Ph D dissertation*, University of California. Berkeley. 1984

[5] Coonen. J. T.. "An implementation Guide to a Proposed Standard for Floating - Point Arithmetic ", *Computer Magazine*, vol. 13, no 1, pp. 68 - 79, January 1980

[6] "IEEE Standard for Binary Floating - Point Arithmetic", ANSI/IEEE Std 754 - 1985, New York, The Institute of Electrical and Electronics Engineers Inc., August 12, 1985

[7] David Goldberg, "What Every Computer Scientist Should Know About Floating - Point Arithmetic", *ACM Computing Surveys*, Vol. 23, No. 1, pp. 5 - 48, March

1991

[8] Israel Koren, "Computer Arithmetic Algorithms", Prentice Hall, Englewood Cliffs, 1993.

[9] Kai Hwang, "Computer Arithmetic: principles, architecture and design", John Wiley and Sons, 1979

[10] T. Kilburn, D. B. G. Edwards and D. Aspinall, "Parallel Addition in Digital Computers: A New Fast "Carry" Circuit," *Proc. IEE*, Vol. 106, Pt. B, p. 464, September 1959.

[11] M. Lehman, "A Comparative Study of Propagation Speed-up Circuits in Binary Arithmetic Units," Information Processing, 1962, Elsevier-North Holland, Amsterdam, pp. 671, 1963.

[12] J. Sklansky, "Conditional Sum Addition Logic", *Trans. IRE*, Vol. EC - 9, No. 2, pp. 226 - 230, June 1960.

[13] M. Lehman and N. Burla, "Skip Techniques for High - Speed Carry - Propagation in Binary Arithmetic Units," *IRE Transactions on Electronic Computers*, pp. 691, December 1961

[14] R. P. Brent and H. T. Kung, "A regular layout for parallel adders", *IEEE Transactions on Computers*, C - 31, pp. 260 -264, March 1982.

[15] T. Kilburn, D. B. G. Edwards and D. Aspinall, "A parallel Arithmetic unit using a

saturated transistor fast-carry circuit", *Proc. of IEE*, Vol. 107, Pt. B, pp. 573 - 584, November 1960.

[16] Andrew D. Booth, 'A signed binary multiplication technique', *Quarterly J. Mechan. Appl. Math.*, 4: 236 - 240 (1951).

[17] O. L. MacSorley, "High speed arithmetic in binary computers", *Proc. IRE*, Vol. 49, pp. 67 - 91, 1961.

[18] D. Villeger and V. G. Oklobdzija, "Evaluation of Booth encoding techniques for parallel multiplier implementation", *Electronics Letters*, Vol. 29, No. 23, pp. 2016 - 2017, 11th November 1993.

[19] C. S. Wallace, "A Suggestion for a Fast Multiplier", *IEEE Trans. Electron. Comput.*, EC - 13: pp. 14 - 17, 1964.

[20] L. Dadda, "Some Schemes for Parallel Multipliers", *Alta Freq.*, 34: 349 - 356 (1965).

[21] L. Dadda, "On parallel Digital Multipliers", *Alta Freq.*, 45: 574 - 580 (1976).

[22] Erdem Hokenek and Robert K. Montoye, "Leading - zero anticipator (LZA) in the IBM RISC System/6000 floating-point execution unit", *IBM J. Res. Develop.*, vol. 34, pp. 70 - 77, 1990.

[23] R. K. Montoye, E. Hokenek and S. L. Runyon, "Design of the IBM RISC system/ 6000 floating point execution unit", *IBM J. Res. Develop.*, vol. 34, pp. 59 - 70,

1990.

[24] Hiroaki Suzuki, Hiroyuki Morinaka, Hiroshi Makino, Yasunobu Nakase, Koichiro Mashiko and Tadashi Sumi, "Leading-Zero Anticipatory Logic for High-Speed Floating Point Addition", *IEEE Journal of Solid State Circuits*, Vol. 31, No. 8, pp. 1157 - 1164, August 1996.

[25] Stuart Franklin Oberman, "Design Issues in High Performance Floating Point Arithmetic Units", Ph. D dissertation, *Department of Electrical Engineering*, Stanford University 1996.

[26] Stuart F. Oberman, Hesham Al-Twaijry, and Michael J. Flynn, The SNAP Project: Design of Floating Point Arithmetic Units, *in Proceedings of the 13th IEEE Symposium on Computer Arithmetic*, July 1997.

[27] Stuart F. Oberman and Michael J. Flynn, A Variable Latency Pipelined Floating-Point Adder, *Proceedings of Euro-Par'96*, Lyon, France, Springer LNCS vol. 1124, pp. 183-192, August 1996.

[28] Davies, T. C., "A Floating Point Systolic Array Processing Element Using Serial Communication", *Masters Thesis*, Royal Military College, Kingston, CANADA, 1990.

[29] Davies, T. C., Al - Khalili, D. and Swarc, V., "A floating point systolic array processing element with serial communication and built in self test", *Journal of VLSI Signal Processing*, vol. 8, No. 3, pp. 241 - 251, December 1994.

[30] Young Surk Lee, "IEEE standard floating point ALU with 60 MHz clock frequency", *Journal of the Korean Institute of Telematics and Electronics*, vol. 28A, No. 11, pp. 61 - 68, November 1991.

[31] Fujii, H., Hori, C., Takada, T., Hatanaka, N., Demura, T. and Ootomo, G., "A Floating Point Cell Library and a 100 MFLOPs Image Signal Processor", *IEEE Journal of Solid State Circuits*, vol. 27, No. 7, pp. 1080 - 1088, July 1992.

[32] Nobohiro Ide, Hiroto Fukuhisa, Yoshihisa Kondo, Takeshi Yoshida, Masato Nagamatsu, Junji Mori, Itaru Yamazaki and Kiyoji Ueno, "A 320 MFLOPs CMOS Floating - Point Unit for Superscalar Processors", *IEEE Journal of Solid State Circuits*, vol. 28, No 3, pp. 352 - 361, March 1993.

[33] Mealer, C. "An Integrated Floating - Point Unit for a RISC Architecture", *Conference record of WESCON/88*, pp. 1.2 - 1.8, 1988.

[34] Gillam, K. R., and Jones, K. R., "Design and Architecture for a Multi - Mode Pipelined Floating Point Adder", *Proceedings of the IEEE 1991 National Aerospace and Electronics Conference (NAECON 1991)*, Dayton, OH, vol. 1, pp. 73 - 76, 1991.

[35] Enriques, A. B., and Jones, K. R., "Design of a Multi - Mode Pipelined Multiplier for Floating Point Applications", *Proceedings of the IEEE 1991 National Aerospace and Electronics Conference (NAECON 1991)*, Dayton, OH, vol. 1, pp. 77 - 81, 1991.

[36] Chai, P., Chuk, T., Fong, Y. H., Hu, L., Ng, K., Prabhu, J., Quck, A., Samuels, A. and Yeun, J., "A 120 MHz CMOS Floating Point Processor", *Proceedings of the IEEE 1991 Custom Integrated Circuits Conference*, San Diego, pp. 15.1/1 - 4, 12 - 15 May 1991.

[37] Nakano, H., Nakajima, M., Nakakura, Y., Yoshida, T., Goi, Y., Nakai, Y., Segawa, R., Kishida, T. and Kadota, H., "A 80 MHz 64 bit Microprocessor for Parallel Computer", *Proceedings of the IEEE 1991 Custom Integrated Circuits Conference*, San Diego, pp. 15.2/1 - 4, 12 - 15 May 1991.

[38] Hsu, P. Y. T., "Designing the TFP Microprocessor", *IEEE Micro*, vol. 14, No 2, pp. 23 - 33, April 1994.

[39] John A Kowaleski Jr., Gilbert M Wolrich, Timothy C Fischar, Rober J Dupcak, Patricia L Kroesen, Tung Pham and Andy Olesin, "A Dual Execution Pipelined Floating - Point CMOS Processor", *Proceedings of the 1996 IEEE International Solid State Circuits Conference*, pp. 358 - 359, 1996.

[40] Novak, J. H. and Brunvand, E., "Using FPGAs to Prototype a Self - Timed Floating Point Co - Processor", *Proceedings of the IEEE 1994 Custom Integrated Circuits Conference*, San Diego, pp. 85 - 88, May 1994.

[41] Hiroshi Makino, Hiroaki Suzuki, Hiroyuki Morinaka, Yasunobu Nakase, Koichiro Mashiko and Tadashi Sumi, "A 286 MHz 64-b Floating Point Multiplier with Enhanced CG Operation", *IEEE Journal of Solid State Circuits*, vol. 31, No 4, pp.

504 - 513, April 1996.

[42] Narasimhan, D., Fernandes, D., Raja, V. K., Dorenbosch, J., Bowden, M., and Kapoor, V. S., "A 100 MHz FPGA based Floating Point Adder", *Proceedings of the IEEE 1993 Custom Integrated Circuits Conference*, San Diego, pp. 3.1.1 - 3.1.4, 9 - 12 May 1993.

[43] Samuel Naffziger, "A Sub - Nanosecond 64 bit Adder Design", *Proceedings of the 1996 IEEE International Solid State Circuits Conference*, pp. 662 - 663, 1996.

[44] Ling, H., "High Speed Binary Adder", *IBM J. Res. Develop.*, vol. 25, No 3, pp. 156, May 1981.

[45] Mori, J., Nagamatsu, M., Hirano, M., Tanaka, S., Noda, M., Toyoshima, Y., Hashimoto, K., Hayashida, H. and Maeguchi, K., "A 10 ns 54X54 b Parallel structured full Array Multiplier with 0.5 micron CMOS Technology", *IEEE Journal of Solid State Circuits*, vol. 26, No 4, pp. 600 606, April 1991.

[46] Song, P. J. and De Micheli, G., "Circuit and Architecture Trade - offs for High - Speed Multiplication", *IEEE Journal of Solid State Circuits*, vol. 26, No 9, pp. 1184 - 1198, September 1991.

[47] Norio Ohkubo, Makoto Suzuki, Toshinobu Shinbo, Toshiaki Yamanaka, Akihiro Shimizu, Katsuro Sasaki, and Yoshinobu Nakagome, "A 4.4nS CMOS 54X54-b Multiplier Using Pass-Transistor Multiplexer," *IEEE Journal of Solid State Circuits*, Vol. 30, pp. 251 - 257, March 1995.

[48] Makoto Hanawa, Kanji Kanako, Tatsuya Kawashimo, Hiroshi Maruyama, "A 4.3 ns 0.3 μm CMOS 54X54 b Multiplier Using Precharged Pass - Transistor Logic", *Proceedings of the 1996 IEEE International Solid State Circuits Conference*, pp. 354 - 355, 1996.

[49] Vojin G. Oklobdzija, David Villeger and Simon S. Liu, "A Method for Speed Optimized Partial Product Reduction and Generation of Fast Parallel Multipliers Using an Algorithmic Approach", *IEEE Transactions on Computers*, vol. 45, No. 3, pp. 294 - 306, March 1996.

[50] Luigi Dadda and Vincenzo Piuri, "Pipelined Adders", *IEEE Transactions on Computers*, vol. 45, No. 3, pp. 348 - 356, March 1996.

[51] Peter M. Kogge, "The Architecture of Pipelined Computers," McGraw - Hill advanced Computer Science series, 1981.

[52] Anantha P. Chandrakasan, Miodrag Potkonjak, Renu Mehra, Jan Rabey and Robert W. Broderson, "Optimizing Power Using Transformations", *IEEE Transactions on Computer - Aided Design of Integrated Circuits and Systems*, Vol. 14, No. 1, pp. 12 - 31, January 1995.

[53] Miodrag Potkonjak, Pranav Ashar, Sujit Dey, Toshio Misowa and Rabindra K. Roy, "Synthesis Techniques for Low - Power Digital Designs," *NEC Res. & Develop.*, Vol. 36, No.1, pp. 83 - 102, January 1995.

[54] Carl Lemonds and S. S. Mahant Shetti, "A Low Power 16 by 16 Multiplier Using

Transition Reduction Circuitry," in *Proceedings of the 1994 International workshop on Low Power Design*, pp. 139 - 142, 1994.

[55] Chin Hwee Tan and Jonathen Allen, "Minimization of Power in VLSI Circuits Using Transistor Sizing, Input Ordering, and Statistical Power Estimation," in *Proceedings of the 1994 International workshop on Low Power Design*, pp. 75 - 80, 1994.

[56] S. C. Prasad and K. Roy, "Circuit Optimization for Minimization of Power Consumption under Delay Constraint," in *Proceedings of the 1994 International workshop on Low Power Design*, pp. 15 -20, 1994.

[57] L. Benini, M. Favalli, and B. Ricco, "Analysis of Hazard Contributions to Power Dissipation in CMOS IC's," in *Proceedings of the 1994 International workshop on Low Power Design*, pp. 27 - 32, 1994.

[58] Sarma B. K. Vrudhula and Hong-Yu Xie, "Techniques for CMOS Power Estimation and Logic Synthesis for Low Power," in *Proceedings of the 1994 International workshop on Low Power Design*, pp. 21 - 26, 1994.

[59] Rajeev Murgai, Robert K. Brayton, and Alberto Sangiovanni-Vincentelli, "Decomposition of Logic Functions for Minimum Transition Activity," in *Proceedings of the 1994 International workshop on Low Power Design*, pp. 33 - 38, 1994.

[60] Razak Hossain and Alexander Albicki, "Low Power via Reduced Switching

Activity and its Application to PLAs." in *Proceedings of the 7th Annual IEEE International ASIC Conference and Exhibition*, pp. 100 - 103, Rochestor, N.Y., September 19 - 23, 1994.

[61] Anantha P. Chandrakasan, Randy Allmon, Anthony Stratakos, and Robert Brodersen, "Design of Portable Systems," in *Proceedings of the 1994 IEEE International Solid-State Circuits Conference*, pp. 259 - 266, 1994.

[62] Mircea R. Stan and Wayne P. Burleson, "Limited Weight Codes for Low Power I/O," in *Proceedings of the 1994 International workshop on Low Power Design*, pp. 209 - 214, 1994.

[63] Mircea R. Stan and Wayne P. Burleson, "Bus-Invert Coding for Low-Power I/O", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 3, pp. 49 - 58, March 1995.

[64] E. Olson and S. M. Kang, "Low Power State Assignment for Finite State Machines," in *Proceedings of the 1994 International workshop on Low Power Design*, pp. 63 - 68, 1994.

[65] Gary D. Hachtel, Mariano Hermida De La Rica, Abelardo Pardo, Massimo Poncino, and Fabio Somenzi, "Re-encoding Sequential Circuits to Reduce Power Dissipation," in *Proceedings of the 1994 International workshop on Low Power Design*, pp. 69 - 74, 1994.

[66] Yoshinobu Nakagome, Kiyoo Itoh, Masanori Isoda, Kan Takeuchi, and Masakazu

Aoki. "Sub-1V Swing Internal Bus Architecture for Future Low-Power ULSI's." *IEEE Journal of Solid State Circuits*, Vol. 38, pp 414 - 419, April 1993.

[67] Laurence Goodby, Alex Orailoglu, and Paul M. Chau, "A High Level Synthesis Methodology for Low Power VLSI Design," in *Proceedings of the 1994 IEEE Symposium on Low Power Electronics*, pp. 48 -49, 1994.

[68] Milos D. Ercegovac and Tomas Lang, "Reducing transition counts in Arithmetic Circuits," in *Proceedings of the 1994 IEEE Symposium on Low Power Electronics*, pp. 64 - 65, 1994.

[69] R. Iris Baher, Hyunwoo Cho, Gary D. Hachtel, Enrico Macii, and Fabio Somenzi, "An Application of ADD-based Timing Analysis to Combinational Low Power Resynthesis," in *Proceedings of the 1994 International workshop on Low Power Design*, pp. 39 - 44, 1994.

[70] Mazhar Alidina, Jose Monteiro, Sreenivas Devadas, Abhijit Ghosh, and Marios Papaefthymiou, "Precomputation-based Sequential Logic Optimization for Low Power," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 2, No. 4, pp. 426 - 436, December 1994.

[71] Peter H. Schneider and Ulf Schlichtmannet, "Decomposition of Boolean Functions for Low Power Based on a new Power Estimation Technique," in *Proceedings of the 1994 International workshop on Low Power Design*, pp. 123 - 128, 1994.

[72] Anand Raghunathan, sujit Dey and Niraj K. Jha, "Glitch Analysis and Reduction in Register Transfer Level Power Optimization", *Proceedings of the 33rd Design Automation Conference*, June 3 - 7, 1996, Las Vegas, pp. 331 - 336, 1996.

[73] C. Papachristou, M. Spinning and N. Nourani, "An Effective Power Management Scheme for RTL Design Based on Multiple Clocks", *Proceedings of the 33rd Design Automation Conference*, June 3 - 7, Las Vegas, pp. 337 - 342, 1996.

[74] Farid N. Najm, "A Survey of Power Estimation Techniques in VLSI Circuits", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 2, No. 4, pp. 446 - 455, December 1994.

[75] Kazuo Yano, Toshiaki Yamanaka, Takashi Nishida, Masayoshi Sito, Katsuhiro Shimohigashi, and Akihiro Shimizu, "A 3.8-ns CMOS 16X16-b Multiplier Using Complementary Pass-Transistor Logic," *IEEE Journal of Solid State Circuits*, Vol. 25, pp. 388 - 395, April 1990.

[76] D.Ghosh, S. K. Nandy, K. Parthasarathy and V. Viswanathan, "NPCPL, Normal process complementary pass transistor logic for low latency, high throughput applications", in *Proceedings of VLSI DESIGN '93, 6th Int. Conf. VLSI Design*, Bombay, India, pp. 341 - 346, June 1993.

[77] J. C. H. Latour, "High - Speed Complex Multiplier Integrated Circuit with Emphasis on Low Power Design", *Masters Thesis*, Royal Military College, Kingston, CANADA, 1995.

[78] Makoto Suzuki, Norio Ohkubo, Toshiaki Yamanaka, Akihiro Shimizu, and Katsuro Sasaki, "A 1.5nS 32b CMOS ALU in Double Pass-Transistor Logic," in *Proceedings of the 1993 IEEE International Solid-State Circuits Conference*, pp. 90 - 91, 267, 1993.

[79] Makoto Suzuki, Norio Ohkubo, Toshinibu Shinbo, Toshiaki Yamanaka, Akihiro Shimizu, Katsuro Sasaki, and Yoshinobu Nakagome, "A 1.5-nS 32-b CMOS ALU in Double Pass-Transistor Logic," *IEEE Journal of Solid State Circuits*, Vol. 28, pp. 1145 - 1151, November 1993.

[80] Akilesh Parameshwar, Hiroyuki Hara, and Takayasu Sakurai, "A High Speed, Low Power, Swing Restored Pass-Transistor Logic Based Multiply and Accumulate Circuit for Multimedia Applications," in *Proceedings of the 1994 IEEE Custom Integrated Circuits Conference*, 278 - 281, 1994.

[81] John H. Pasternak and C. Andre T. Salama, "Differential Pass-Transistor Logic," *IEEE Journal of Circuits and Devices*, pp. 23 - 28, July 1993.

[82] Norio Ohkubo, Makoto Suzuki, Toshinobu Shinbo, Toshiaki Yamanaka, Akihiro Shimizu, Katsuro Sasaki, and Yoshinobu Nakagome, "A 4.4nS CMOS 54X54-b Multiplier Using Pass-Transistor Multiplexer," in *Proceedings of the 1994 IEEE Custom Integrated Circuits Conference*, pp. 599 - 602, 1994.

[83] Norio Ohkubo, Makoto Suzuki, Toshinobu Shinbo, Toshiaki Yamanaka, Akihiro Shimizu, Katsuro Sasaki, and Yoshinobu Nakagome, "A 4.4nS CMOS 54X54-b

Multiplier Using Pass-Transistor Multiplexer," *IEEE Journal of Solid State Circuits*, Vol. 30, pp. 251 - 257, March 1995.

[84] Roderick T. Hinman and Martin F. Schlecht, "Recovered Energy Logic: A Single Clock AC Logic," in *Proceedings of the 1994 International workshop on Low Power Design*, pp. 153 - 158, 1994.

[85] Saed G. Younis and Thomas F. Knight, Jr., "Asymptotically Zero Energy Split-Level Charge Recovery Logic," in *Proceedings of the 1994 International workshop on Low Power Design*, pp. 177 - 182, 1994.

[86] John S. Denker, "A Review of Adiabatic Computing," in *Proceedings of the 1994 IEEE Symposium on Low Power Electronics*, pp. 94 - 97, 1994.

[87] L. "J." Svensson and J. G. Koller, "Driving a Capacitive Load without Dissipating fCV^2," in *Proceedings of the 1994 IEEE Symposium on Low Power Electronics*, pp. 100 - 101, 1994.

[88] Thomas Indermaur and Mark Horowitz, "Evaluation of Charge Recovery Circuits and Adiabatic Switching for Low Power CMOS Design," in *Proceedings of the 1994 IEEE Symposium on Low Power Electronics*, pp. 102 - 103, 1994.

[89] L. J. Svensson and J. G. Koller, "Adiabatic Charging without Inductors," in *Proceedings of the 1994 International workshop on Low Power Design*, pp. 159 - 164, 1994.

[90] John S. Denker, Stephen C. Avery and Alex G. Dickinson, "Adiabatic Computing with the 2N-2N2D Logic Family," in *Proceedings of the 1994 International workshop on Low Power Design*, pp. 183 - 188, 1994.

[91] W.C. Athas, L. J. Svensson, J. G. Koller, N. Tzartzanis, and E. Chou, "A Framework for Practical Low Power Digital CMOS Systems Using Adiabatic Switching Principles," in *Proceedings of the 1994 International workshop on Low Power Design*, pp. 189 - 194, 1994.

[92] Thad Gabara, "Pulsed Power Supply CMOS - PPS CMOS," in *Proceedings of the 1994 IEEE Symposium on Low Power Electronics*, pp. 98 - 99, 1994.

[93] Thad Gabara, "Quasi-Static CMOS," in *Proceedings of the 7th Annual IEEE International ASIC Conference and Exhibition*, pp. 104 - 107, Rochestor, N.Y., September 19 - 23, 1994.

[94] William C. Athas, Lars "J." Svensson, Jeffrey G. Koller, Nestoras Tzartzanis and Eric Ying - Chin Chou, "Low-Power Digital Systems Based on Adiabatic-Switching Principles," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 2, pp. 398 - 407, December 1994.

[95] Stephen W. Director, Peter Feldmann, and Kannan Krishna, "Statistical Integrated Circuit Design," *IEEE Journal of Solid State Circuits*, Vol. 28, pp. 193 - 202, March 1993.

[96] Uming Ko, P. T. Balsara, and Wai Lee, "A Self-Timed Method to Minimize

Spurious Transitions in Low Power CMOS Circuits," in *Proceedings of the 1994 IEEE Symposium on Low Power Electronics*, pp. 62 - 63, 1994.

[97] David Goldberg, "Computer Arithmetic", in *Computer Architecture: A quantitative approach*, D. A. Patterson and J. L. Hennessy, Morgan Kaufman, San Mateo, CA, Appendix A, 1990.

[98] R. V. K. Pillai, D. Al - Khalili and A. J. Al - Khalili, "A Low Power Approach to Floating Point Adder Design", *in Proceedings of the 1997 International Conference on Computer Design*, pp. 178 - 186, 1997.

[99] The Law of the Iterated Logarithm in "Discrete Stochastics" by Konrad Jacobs, Birkhauser Verlag, 1992

[100] T. Sweeney, "An analysis of floating - point addition," *IBM Syst. J.*, Vol. 4, pp. 31 - 42, 1965.

[101] Andersen, E. S., "On the number of positive partial sums of random variables", *Skandinavisk Aktuarietidskrift*, 32, pp. 27 - 36, 1949.

[102] Andersen, E. S., On the frequency of positive partial sums of a series of random variables, *Mat. Tidskrift* B, pp. 33 - 35, 1950.

[103] Andersen, E. S., "On sums of symmetrically dependent random variables, *Skandinavisk Aktuarietidskrift*, 36, pp. 123 - 138, 1953.

[104] Andersen, E. S., 'On the fluctuations of sums of random variables I, II, *Math.*

Skand. pp. 263 - 283 pp. 125 - 223, 1 1953 and 2 1954.

[105] Spitzer, F. A., "A combinatorial lemma and its application to probability theory", *Transactions of the American Mathematical Society*, 82, pp. 323 - 339, 1956.

[106] R. W. Hamming, "On the Distribution of Numbers", *Bell System Technical Journal*, Vol. 49, No. 8, pp. 1609 - 1625, October 1970.

[107] A. Lacroix and F. Hartwig, "Distribution Densities of the Mantissa and Exponent of Floating Point Numbers", *Proceedings of ISCAS92*, pp. 1792 - 1795, 1992.

[108] Kari Kalliojarvi and Yrjo Neovo, "Distribution of Roundoff Noise in Binary Floating - Point Addition", *Proceedings of ISCAS92*, pp. 1796 - 1799, 1992.

[109] R. V. K. Pillai, D. Al - Khalili and A. J. Al - Khalili, "On the Distribution of Exponent Differences During Floating Point Addition", *Proceedings of the 1998 Canadian Conference on Electrical and Computer Engineering*, Vol. 1, pp. 105 - 108, 1998.

[110] R. V. K. Pillai, D. Al - Khalili and A. J. Al - Khalili, "On the Power Implications of Floating Point Addition in IIR Filters", *Proceedings of the 10th International Conference on Microelectronics*, pp. 200 - 203, December 1998.

[111] Sumant Ramaprasad, Naresh R. Shanbag and Ibrahim N. Hajj, "Analytical Estimation of Signal Transition Activity from Word-Level Statistics", *IEEE Transactions on Computer - Aided Design of Integrated Circuits and Systems*,

Vol. 16, No. 7, pp. 718 - 733, July 1997.

[112] R. V. K. Pillai, D. Al - Khalili and A. J. Al - Khalili, "Power Implications of Additions in Floating Point DSP - an Architectural Perspective", *Technical report of the Department of Electrical and Computer Engineering* (No. 1998-2-pillai), Concordia University, Montreal, CANADA, 1998.

[113] R. V. K. Pillai, D. Al - Khalili and A. J. Al - Khalili, "Power Implications of Precision Limited Arithmetic in Floating Point FIR Filters", to be presented at the *1999 International Symposium on Circuits and Systems*, Orlando, Florida during May 30 to June 2, 1999.

[114] Joseph J. F. Cavanagh, "Digital Computer Arithmetic - Design and Implementation", McGraw-Hill, 1983.

[115] Mark R. Santoro, Gary Bewick and Mark A. Horowitz, "Rounding Algorithms for IEEE Multipliers", *Proc. 9th Symp. on Computer Arithmetic*, pp. 176 - 183, 1989.

[116] A.Weinberger, "4:2 Carry save adder module", *IBM Tech. Disclosure Bulletin*, 23, 1981

[117] G. M. Tharakan and S. M. Kang, "A New Design of a Fast Barrel Switch Network", *IEEE Journal of Solid-State Circuits*, Vol. 27, No. 2, pp. 217 - 221, February 1992.

[118] R. Pereira, J. A. Michell and J. M. Solana, "Fully Pipelined TSPC Barrel Shifter

for High-Speed Applications", *IEEE Journal of Solid State Circuits*, Vol. 30, No. 2, pp. 217 - 221, February 1992.

[119] K. P. Acken, M. J. Irwin and R. M. Owens, "Power Comparisons for Barrel Shifters", Digest of Technical Papers - 1996 International Symposium on Low Power Electronics and Design, pp. 209 - 212, 1996.

[120] Raymond S. Lim, "A Barrel Switch Design", *Computer Design*, pp. 76 - 79, August 1972.

[121] Kohn, L. and S. W. Fu, "A 1, 000, 000 Transistor Microprocessor", in *Proceedings of the 1989 IEEE International Solid State Circuits Conference*, pp. 54 - 55, 1989.

[122] Anantha P. Chandrakasan, Samuel Sheng and Robert W. Brodersen, "Low-Power CMOS Digital Design," *IEEE Journal of Solid State Circuits*, Vol. 27, pp. 473 - 483, April 1992.

[123] R. V. K. Pillai, D. Al - Khalili and A. J. Al - Khalili, "Energy Delay Measures of Barrel Switch Architectures for Pre-Alignment of Floating Point Operands for Addition", in *Proceedings of the 1997 IEEE International Symposium on low power Electronics* and *Design*, 1997

[124] R. V. K. Pillai, D. Al - Khalili and A. J. Al - Khalili, "Evaluation of 1's Complement Arithmetic for the Implementation of low power CMOS Floating Point Adders", *Proceedings of the 1997 Canadian Conference on Electrical and Computer Engineering*, Vol. 1, pp. 153 - 156, May 1997.

[125] R.V.K. Pillai, D. Al-Khalili and A.J. Al-Khalili, "Energy Delay Analysis of Partial Product Reduction Methods for Parallel Multiplier Implementation", Digest of Technical Papers - *1996 International Symposium on Low Power Electronics and Design*, pp. 201 - 204, 1996.

[126] David J. Kuck, Douglass S. Parker and Ahmed H. Ameh, "Analysis of Rounding Methods in Floating - Point Arithmetic", *IEEE Transactions on Computers*, Vol. C - 26, No. 7, pp. 643 - 650, July 1977.

[127] Bhaskar D. Rao, "Floating Point Arithmetic and Digital Filters", *IEEE Transactions on Signal Processing*, Vol. 40, No. 1, pp. 85 - 95, January 1992.

[128] Alan V. Oppenheim and Ronald W. Schafer, "Discrete-time signal processing", Englewood Cliffs, N.J., Prentice Hall, 1989.

[129] Paul E. Landman and Jan M. Rabaey, "Architectural Power Analysis: The Dual Bit Type Method", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 3, No. 2, pp. 173 - 187, June 1995.

[130] Paul E. Landman and Jan M. Rabaey, "Activity - Sensitive Architectural Power Analysis", *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, Vol. 15, No. 6, pp. 571 - 587, June 1996.

[131] Mahadevamurty Nemani and Farid N. Najm, "Towards a High-Level Power Estimation Capability", *IEEE Transactions on Computer - Aided Design of Integrated Circuits and Systems*, Vol. 15, No. 6, pp. 588 - 598, June 1996.

[132] Diana Marculescu, Radu Marculescu and Massoud Pedram, "Information Theoretic Measures for Power Analysis", *IEEE Transactions on Computer - Aided Design of Integrated Circuits and Systems*, Vol. 15, No. 6, pp. 599 - 610, June 1996.

[133] S. Powell and P. Chau, "A Model for Estimating Power Dissipation in a Class of DSP VLSI Chips: The PFA Technique", *IEEE Transactions on Circuits and Systems*, pp. 646 - 650, June 1991.

[134] J. S. Ward, P. Barton, J. B. G. Roberts and B. J. Stanier, "Figures of Merit for VLSI Implementations of Digital Signal Processing Algorithms," *Proceedings of IEE*, Vol 131, Part F, No. 1, pp. 64 - 70, February 1984.

[135] R. V. K. Pillai, D. Al - Khalili and A. J. Al - Khalili, "Power Implications of Additions in Floating Point DSP - an Architectural Perspective", Submitted to AFRICON'99.

[136] R. V. K. Pillai, D. Al - Khalili and A. J. Al - Khalili, "A Low Power Approach to Floating Point Adder Design", *Submitted to IEEE Transactions on Circuits and Systems.*

[137] R. V. K. Pillai, D. Al - Khalili and A. J. Al - Khalili, "A Low Power Floating Point Accumulator", *Proceedings of the Eleventh International Conference on VLSI Design*, pp. 330 - 333, January 1988.

[138] R. V. K. Pillai, D. Al - Khalili and A. J. Al - Khalili, "Arithmetically Sub-optimal

Floating Point Digital Filters - An Architectural Power Perspective". To be presented at the *1999 Canadian Conference on Electrical and Computer Engineering*, May 1999.

[139] R. V. K. Pillai, D. Al - Khalili and A. J. Al - Khalili, "A New Architecture for Floating Point Multiply - Accumulate Fusion". *Submitted to* ICCD '99.

[140] R. V. K. Pillai, D. Al - Khalili and A. J. Al - Khalili, "On the Structure Driven Power Behavior of Additions in Floating Point IIR filters". *Technical report of the Department of Electrical and Computer Engineering* (No. 1998-3-pillai), Concordia University, Montreal, CANADA, 1998.

[141] R. V. K. Pillai, D. Al - Khalili and A. J. Al - Khalili, "On the Structural Power Implications of Additions in Floating Point IIR Filters", *Submitted to IEEE Transactions on Circuits and Systems.*

Appendix A

Effect of multiplicand truncation on rounding

With the proposed scheme, the effect of multiplicand truncation in rounding is applicable only during situations when the shift of the product is greater than 4. Because of this, the truncation of the shifted multiplicand doesn't cause catastrophic errors [7]. Figure A.1 illustrates the relative positions of the aligned significands as well as the result during signed magnitude addition of significands in MAC operation. The *G*, *R*, *S* bits facilitate round to nearest, even [8]. During situations when the shift of the product is greater than 4, the truncation of the multiplicand has the effect of discarding a carry input - *Cin* - at *S* bit position. The assertion of *OV* indicates that the result of significand addition is ≥ 2, which necessitate a normalization shift. The assertion of a 0 at *LB* bit position during subtraction indicates the generation of a leading zero, which situation also demands normalization shift. In Figure A.1, *S'* represent the natural bit at *S* bit position of the product of significands that is being replaced by the *S* bit. Here, product of significands implies the product that is formed through the multiplication of the multiplier and truncated multiplicand.

In the proposed scheme, since the *S* bit is formed through trailing zero evaluation, multiplicand truncation doesn't affect the accuracy of this bit. Also, during situations when the *S* bit is zero, there is no error due to multiplicand truncation. An *S* bit of zero implies that all other bits at and towards the right of this bit position are zeros, which means both *S'* and *Cin* are guaranteed to be zeros during such situations. Whenever *S'* or/and *Cin* are zeros (irrespective of the assertion of *S*) the carry injection at *S* bit position cannot propa-

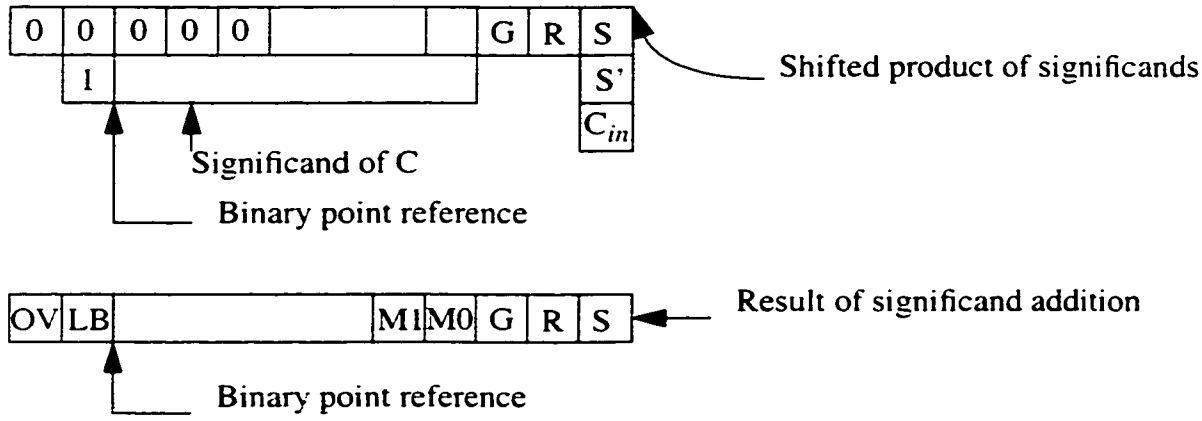


Figure A.1 - Effect of carry injection at S bit position when |AB| < |C/16|

gate to the higher order bit positions of the product. Situations that doesn't come under any of these cases are limited. These cases are highlighted in Table 4.

In Table 4, the error terms ± 1 ULP indicate unnecessary rounding up/down that is prompted due to the truncation of the shifted multiplicand. During situations when the product of significands is shifted through 5 bit positions, the assertion of *OV* bit is possible only if three of the bits of the significand of *C* towards the right of the leading 1 are 1's and the significand addition produces a carry injection at the 3rd bit position. Because of this, the probability that *OV* is asserted is 1/16. However, with shifts exceeding 5 bits, the signal probability of *OV* is reduced. Through similar arguments, it can be shown that the probability that *LB* bit becomes zero during subtraction with a product shift of 5 is 1/16. Assuming signal probabilities of 1/2 for *G*, *R*, *S'*, *Cin*, *M*0 bits and 1/16 for *OV* and *LB* bits and further assuming that *S* bit is always asserted and the chances for addition and subtraction are equally likely, the total probability that a 1 ULP error occurs due to multiplicand truncation is 9/512. Taking into account the probability that $|AB| \leq |C/32|$, the expected extra rounding error (worst case) due to multiplicand truncation is given by

Table A.1: Effect of Multiplicand Truncation on Rounding Error

G	R	S	S'	Cin	Error in addition		Error in subtraction	
					$OV = 0$	$OV = 1$	$LB = 1$	$LB = 0$
0	0	1	1	1	0	0	0	+ 1 ULP
0	1	1	1	1	- 1 ULP	0	+ 1 ULP	0
1	0	1	1	1	0	0	0	+ 1 ULP
1	1	1	1	1	0	- 1 ULP[a]	0	0

a. This happens if and only if the MO bit of the sum = 0. W

$$E(error) = P(Trunc)\frac{9}{512}ULP \tag{8.1}$$

where $P(Trunc)$ represents the probability that multiplicand truncation occurs. In our simulations, the largest value of $P(Trunc)$ observed is less than 0.5.

In precision limited floating point computing, a multiplication operation involving operands A and B is realized as $fl(AB) = AB(1 + \lambda)$ [127]. In this equation, $fl(AB)$ implies the result of precision limited multiplication of numbers A and B. λ is a random variable that is uniformly distributed in the interval ± 1/2ULP (unit in the last place). Similarly $AB + C$ is realized as $fl(AB + C) = (AB + C)(1 + \varepsilon)$. ε is also uniformly distributed in the interval ± 1/2ULP. With FP units that envisage separate rounding of the sum and product,

$$fl(AB + C) = [AB(1 + \lambda) + C](1 + \varepsilon) = (AB + C)(1 + \varepsilon) + AB(\lambda + \lambda\varepsilon) \tag{8.2}$$

The random variables λ and ε are independent. With the proposed scheme, $AB + C$ is real-

ized as $fl(AB + C) = (AB + C)(1 + \gamma)$. where γ represents a random variable that reflects the rounding implications. With $P(Trunc)$ approximated as 0.5 (worst case), the probability density function of γ is given by

$$f(\gamma) = \frac{3}{512}\delta[x - ULP] + \frac{3}{1024}\delta[x + ULP] + \frac{1015}{1025}f(\varepsilon) \tag{8.3}$$

The first two components of the density function given above accounts for the extra rounding error due to multiplicand truncation. $f(\varepsilon)$ represents a uniform distribution within $\pm$ 1/2ULP. In essence, the pdf of γ can be thought of as a perturbed distribution of ε. It can be easily verified that the contribution of rounding error due to the term $AB(\lambda + \lambda\varepsilon)$, in general, outweighs the error due to significand truncation. During situations when $|AB| \geq |C/16|$, the pdf of γ is the same as that of ε. However, with schemes that round the product and sum separately, the error contribution due to the term $AB(\lambda + \lambda\varepsilon)$ is always relevant. With large shifts of the product of AB, the kronecker delta terms of equation (8.3) assume too small values. With relatively large shifts, the probability that *OV* bit is asserted or the *LB* bit becomes zero is marginally low.

Appendix B

List of Abbreviations

ALU	Arithmetic Logic Unit
AND	Logical AND operation
AF	Activity Factor
AR	Auto Regressive
BFLOPS	Billions of Floating Point Operations per Second
CAD	Computer Aided Design
CMOS	Complementary Metal Oxide Semiconductor
CPL	Complementary Pass Transistor Logic
CPU	Central Processing Unit
dB	Decibel
DMA	Direct Memory Access
DSP	Digital Signal Processing
FIR	Finite Impulse Response

FPGA	Field Programmable Gate Array
FP	Floating Point
FPU	Floating Point Unit
FADD	Floating Point Adder
G, R, S	Guard, Round and Sticky bits
IC	Integrated Circuit
IID	Independent, Identically Distributed
IIR	Infinite Impulse Response
I/O	Input/Output
LSB	Least Significant Bit
LZA	Leading Zero Anticipatory Logic, LZA data path
LZB	Leading Zero bounded data path
MAC	Multiply - Accumulator
MAF	Multiply - Accumulate Fused Architecture
MBA	Modified Booth's Algorithm

MFLOPS	Millions of Floating Point Operations per Second
MOS	Metal Oxide Semiconductor
MSB	Most Significant Bit
MUX	Multiplexors
N(-,-)	Normal Distribution
NaN	Not a Number
NMOS	N type MOS transistor
OR	Logical OR operation
OV	Overflow
pdf	probability density function
PSUM	Partial Sum
RAM	Random Access Memory
ROM	Read Only Memory
RISC	Reduced Instruction Set Computer
RTL	Register Transfer Level

RV	Random Variable
SOP	Sum of Products
TDPFADD	Triple Data Path Floating Point Adder
ULP	Unit in the Last Place
VLSI	Very Large Scale Integration

Appendix C

List of Symbols

A.B.C.num	Floating point numbers
α	Alignment driven switching activity measure
β	Alignment driven/sign toggling dominated switching activity measure, probability that the FADD is not operating in bypass mode
γ	Alignment driven/sign toggling dominated switching activity measure
C_L	Capacitive loading
e	Exponents
η	Efficiency of drivers
$E[\ ]$	Expectation operator
EM	Energy measure
EDM	Energy delay measure
ESW	Expected number of switching transitions
f	Frequency variable

F_g	Fanout of a gate
$g()$	probability density function
P, p	Power, signal probabilities, width of significand
σ^2	Variance
$\sum$	Summation operator
Φ	Switching activity measure
$\otimes$	Discrete convolution
$\oplus$	Exclusive OR
k	index variable in loops, co efficient of parasitic loading
S	Stage ratio of drivers
SH	Shift, number of bits
SW	Switching activity measure
$t_{D,}$:	Toggle distance
t_s	probability for sign toggling
TZ	Trailing zero count

τ	Delay variable
V_{dd}	Supply voltage
Y	Fanout
$\lvert\ \rvert$	Absolute value
$\lVert\ \rVert$	Matrix/vector norms