Quantitative Evaluation of the Impact of Floating Point Arithmetic Units on the Performance of DSP Structures

Wassim Tout

A Thesis
in
The Department
of
Electrical And Computer Engineering

Presented in Partial Fulfilment of The Requirements
for The Degree of Master of Applied Science at
Concordia University
Montreal, Quebec, Canada

August 2003

Your file Votre référence
ISBN: 0-612-91126-8
Our file Notre référence
ISBN: 0-612-91126-8

Abstract

Quantitative Evaluation of the Impact of Floating Point Arithmetic Units on the Performance of DSP Structures

Wassim Tout

Arithmetic operations traditionally used fixed-point processing because it makes them less expensive. In integer and fixed-point arithmetic, multipliers are larger, slower and consume much more power than adders, which are often neglected in performance evaluation of DSP systems. In floating-point arithmetic that is not true and in this thesis we show that multipliers and adders are equally important. The thesis also emphasizes low power design. For that reason, some of the basic digital filter network structures, built with FP arithmetic units, are revisited to map their performance with different filtering functions. This thesis presents digital filter network structures' performance with different filtering functions. It presents filter network structures transformed from their original form to accommodate pipe-lined arithmetic units. These filter structures can also be implemented with fixed-point arithmetic units because of the speed advantage they provide. Several experiments, through hardware synthesis of the structures, show that FIR filter Direct form structure using an adder tree consumes less power than Direct form structure using a chain of adders and its Transposed form. They also show that for IIR filters, Direct form II using standard floating-point arithmetic units is power optimal. This research work is intended to provide designers with information on the performance of these structures with different applications in an effort to help reduce the "design gap".

Acknowledgment

I am grateful to Dr. Asim J. Al-Khalili for his supervision and his support, without which this work would not have been accomplished. I wish to thank Rajan V.K. Pillai for providing the audio data samples, Syed Y.A. Shah for providing the VHDL code of the FP arithmetic units and Ted Obuchowicz for his technical assistance with the different CAD tools used to perform the experiments. I am also grateful to my family whose help and support made it all possible.

Table of Content

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Until a few years ago, floating-point (FP) processing was only available in large work stations. Nowadays, FP arithmetic is available in the smallest of computers. The major advantage of FP arithmetic is the high computational accuracy, in the context of truncation necessary in integer arithmetic, it provides along with a significantly wider dynamic range than fixed-point arithmetic. However, FP arithmetic is far more complex than fixed-point arithmetic and using it results in increased chip area, higher power dissipation and reduction in speed. Low power design became crucial with the wide spread of portable information and communication terminals. High performance electronics, in addition, suffer from a permanent increase of the dissipated power per square millimetre of silicon, due to the increasing clock-rates. The increase in power dissipation limits the performance [1][2].

Full-custom implementation and technology-specific optimization strategies are viable choices for low power, high speed digital design. However, reuse of such low level techniques is not possible with technology migration, and their results fade rapidly. High level design optimization techniques survive through generations of technology migration because of their inherent low sensitivity to the implementation technology. In contrast, power reduction and speed enhancement attained with high level design strategies are very significant. The power savings at the algorithm level are orders of magnitude [3].

1.2 Low Power Design

The increasing need for low power consuming systems stems, in large part, from the huge demand for mobile digital electronics. New digital applications such as digital mobile communication devices (phones and PDAs), which utilize complex speech compression algorithms, and sophisticated radio modems in a pocket-sized device require methodologies for low power design. Considerable research efforts are focusing on increasing the life of batteries for mobile devices [4]. Numerous low power design techniques exist for different levels of the design flow. They range from the highest levels like algorithm and architecture all the way down to the physical layers.

At highest levels, the literature presents techniques that deal with sorting the coefficients of FIR filters [5], techniques that exploit the low power capabilities of parallel architectures [6], techniques that try to optimize arithmetic units with one constant operand [7] and transition activity scaling techniques to reduce the switching activity within the design [3][8]. At lower levels, we find such techniques as logic minimization and state encoding [9], hazardous activity elimination by latch repositioning [10] and by gate sizing [11], clock-gating [12][13], output pre-computation [14], gate resizing [15], signal-to-pin assignment and I/O encoding and the use of networks "don't cares" [16]. Of course, optimizing the high-level DSP architecture to allow selection of the optimal low-power process does not imply that lower level optimization techniques are no longer useful. Several low-level optimization techniques reduce power beyond that of architectural extensions. Further optimization can be achieved through process enhancement [17][18][19]. It has been argued that the right algorithm gives the maximum amount of power saving, which can be orders of magnitude

in contrast to other alternatives [3][20][21]. Using the correct algorithm, the right architecture and the proper clocking scheme, a considerable amount of power can be saved without having to use circuit level technology-dependent techniques.

1.3 Motivation

FP arithmetic is gaining in popularity and because of certain assumptions about FP arithmetic units' performance, we deemed necessary to investigate some digital signal processing applications using FP arithmetic units. The simple extension of conclusions from the fixed-point domain to that of the floating-point can lead to unpleasant surprises. An example case of such extensions is the obviously common belief that in the floating-point domain, as in the fixed-point domain, multipliers are more important than adders. Such assumptions can be detrimental to a design cycle.

Other information of critical importance is the performance of different structures with different applications. The transfer function of a digital filter can have different implementations without changing the input/output behaviour. Even if area and delay are the same for all the implementations, the power dissipation is likely to be different. That is due to the fact that different structures have different node capacitances and different switching activities. The designer would not need to investigate the performance of the different implementations if the information is already available. That results in reducing the design cycle time, thus increasing productivity [1]. This research effort is meant to help the designer make design trade-off decisions based on power, area and delay performances of a number of network structures before even starting the design process.

1.4 Design and Experimentation Methodology

In total, three Finite Impulse Response (FIR) and three Infinite Impulse Response (IIR) filter network structures are examined in this work. A few basic configurations using FP adders and multipliers are also examined. Modelling of the different designs is done at Register Transfer Level (RTL) in VHDL. RTL code is technology independent and can be translated into hardware with available technology libraries. Functional verification of the different designs is performed at both register transfer and Gate levels. As the power consumption of controlled data path architectures is highly dependent on the utilisation of different data paths, the role of input data becomes very important. Simulation of all the designs is done using real data obtained through conversion of real audio signals to IEEE single precision floating-point format.

MATLAB is used to obtain the input data and to calculate the parameters for the different filters. It is also used to implement reference or "golden" models and to simulate them with the input data. Simulation results of the reference models are then stored into "golden" files and used to verify the results of the RTL and gate-level simulations. The VHDL System Simulator (VSS) is used to simulate RTL and gate-level netlists and to capture the switching activity. Design Compiler is used to synthesise the designs. Power Compiler, in conjunction with Design Compiler, uses the switching activity captured during simulation to optimize the designs simultaneously for area, delay and power and to analyse their power consumption. VSS, Design Compiler and Power Compiler are all products of Synopsys Inc. After generating the filter parameters and input data and RTL modelling of the architectures, the power optimization/estimation flow consists of several steps. We start with area

and delay optimization during the first pass mapping phase of the design. The design is then simulated at gate level to capture the switching activity, which is then back-annotated onto the gate-level design for simultaneous area and delay and power optimization during a second pass mapping phase. The flow is best illustrated in Figure 1.1. The switching activity of the different designs is captured during gate-level simulation and back-annotated onto the design nets for power optimization and power estimation. Power analysis using switching activity from gate-level simulation is very accurate. It can predict average power to within 10 to 25 percent of power analysis results, using a transistor-level simulator such as SPICE [22].

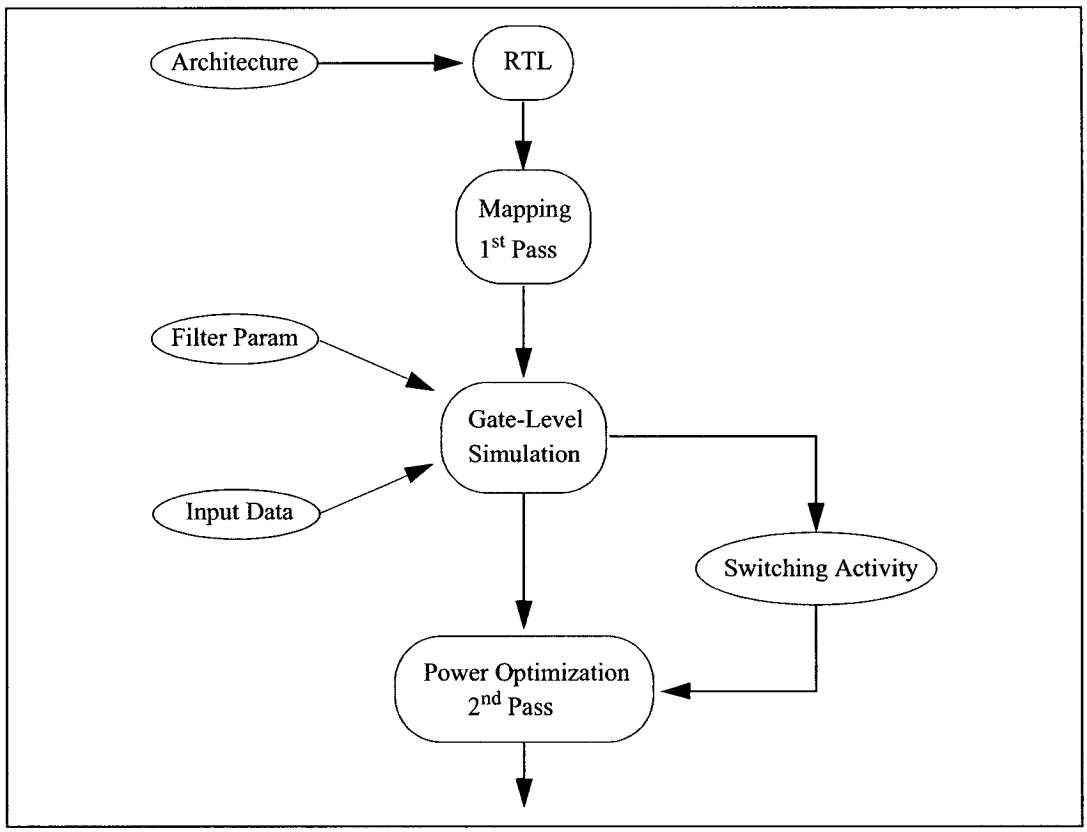

Figure 1.1 Power Optimization Flow

Although the floating-point arithmetic units used for this work are quite large circuits, the gate-level approach does not present any serious problem. The higher accuracy it provides makes it a more reliable approach than that of the RTL's. In fact, we found the results obtained with RTL approach to be inconclusive. It is better, actually, to skip that approach altogether, and work directly with gate-level. The time spent gathering results with the RTL approach should have been spent on the gate-level approach because it is more reliable. To reduce execution time, the building blocks are synthesised individually. Once that is done, they are connected together to form the larger designs. This saves us significant amount of time when synthesizing the large designs.

1.5 Design Implementation

To prove that the findings of this work are not technology dependent, two target technology libraries are used during synthesis of the designs; 0.35 um and 0.18 um CMOS from TSMC provided by the Canadian Microelectronics Corporation (CMC). The operating voltage is 3 V for 0.35 um CMOS technology and 1.62 V for 0.18 um CMOS technology. During synthesis, the same set of constraints is used for the different designs. This ensures that only the difference in architecture is affecting the final results. The filter network structures used are basic common structures slightly modified, where necessary, to accommodate the pipelined floating-point arithmetic units. A detailed examination of these network structures can be found further in this thesis. While the main purpose of this work is to evaluate the power consumption of these different structures, each architecture can be further optimized for power by modifying the algorithm or by using circuit level optimization techniques or by customizing the synthesis constraints for each of the architectures.

Frequency distributions of pre-alignment and normalization shifts, their rate of change and the relevant bit level activities and other activity statistics have also been gathered [23]. This is made possible by carefully inserting counters and flags at particular locations within the designs. These "monitors" allowed for greater visibility of the type of operations being performed by the different components of the design. The extra code needed to implement the monitors is ignored during synthesis and only used to gather the statistics listed above during register transfer level simulation.

Simulation and synthesis of all the designs in this work require significant amount of run-time regardless of the type of platform they are running on. Taking a given design through the power optimization process presented in Figure 1.1 required anywhere between three to eight days of execution time on a six Ultra-SPARC 400 MHz CPU server with 6 GB of RAM, depending on the size of the design. The memory requirement often reached 1.8 GB of RAM with the largest designs. A particular family of designs required several weeks of execution time, mostly for gate level simulation. Experiments performed on this particular family of designs were limited to only the smallest of these designs. Because the experiments required significant amount of time and computing resources, a considerable amount of design automation scripts had to be developed to support this work. Simulation and verification flow scripts were written in addition to power optimization flow scripts to run these jobs in the background without having to rely on interactive shells. Additional scripts were developed for text editing and data extraction and status reporting at each phase of the different flows. Simple UNIX command files, Bourne shell, C shell and PERL were used to develop the design automation scripts.

1.6 Contributions

The contributions of this thesis work can be summarised in the following few bullets:

- Contributions towards addressing assumptions about the performance of arithmetic units in the floating-point domain. We evaluated the performance of a number of floating-point arithmetic units in three different applications. The results showed that multipliers and adders are equally important in the floating-point domain and the performance of the adder cannot be neglected as it is the case in integer and fixed-point arithmetic.
- We present a simple and easy way of pipe-lining digital filter structures by using pipelined arithmetic units.
- The evaluation of high level design decisions for performance optimal realisations of floating-point FIR and IIR filters. We compare the power/performance of several digital filter network structures with different filtering functions and classify them in accordance with their performance.
- We compare performance figures of different implementations of the filter structures and draw conclusions as to performance optimal conditions.
- The results of this work can help in the selection of the best suitable network structure for a given application, thus help reducing the "design gap".

1.7 Thesis Organisation

The thesis is organised in the following manner: Chapter 2 introduces digital filter basics, types, common network structures and applications. Chapter 3 presents an in-depth investigation of area, delay and power performance of floating-point adders and multipliers [8].

Chapter 4 presents three common FIR network structures and how they were adapted to use pipe-lined floating-point arithmetic units in [8]. It also investigates the performance of these network structures with different filtering functions and with different arithmetic units in DSP applications. Chapter 5 presents the Second-order-Section (SOS) implementation of three common IIR network structures and their area, delay and power performance with different filtering functions and with the different floating-point adders and multipliers in DSP applications. Chapter 6 concludes this thesis and presents proposals for future related work.

1.8 Summary

This chapter presented several aspects of the problem of digital design especially for low power. We started out by presenting arguments explaining why there is a need for new low power design methodologies. Next we presented a review of a number of low power design methodologies and approaches that are either being used in the industry or currently under development. The low power design methodologies discussed were divided into separate categories depending on which level of the design flow they are applied. Some approaches apply to physical level, gate-sizing for example, others apply to technology mapping or synthesis level as in the case of clock-gating. The most important approaches for this thesis work are those that apply to algorithmic level. The last part of this chapter discussed the motivation behind this research effort and presented the approach and tools used in the implementation.

Chapter 2

Digital Filters

2.1 Introduction

In DSP applications, filtering of data samples is a very demanding operation. 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 aforementioned exercise does remain the same, different implementations of the same transfer function are possible [24]. Digital Signal Processing (DSP) is one of the most powerful technologies that will shape science and engineering in the near future. Revolutionary changes have already been made in a broad range of fields: communications, medical imaging, radar & sonar, high fidelity music reproduction, and oil prospecting, to name just a few. Each of these areas has developed a deep DSP technology, with its own algorithms, mathematics, and specialised techniques. This combination of "breadth and depth" makes it impossible for any one individual to master all of the DSP technology that has been developed [25]. DSP education involves two tasks: learning general concepts that apply to the field as a whole, and learning specialised techniques for the particular area of interest. The rest of this chapter presents theoretical digital filter basics, usage, types and some of the common network structures.

2.2 Filter Basics

In signal processing, the function of a filter is to remove unwanted parts of the signal (separation of signals), such as random noise, or to extract useful parts of the signal (restoration

of signals), such as the components lying within a certain frequency range. Analog, or electronic, filters can be used for these tasks; however, digital filters can achieve far superior results [25]. Digital filters are a very important part of DSP. In fact, their extraordinary performance is one of the key reasons that DSP has become so popular. Signal separation is needed when a signal has been contaminated with interference, noise, or other signals. For example, a device for measuring the electrical activity of a baby's heart (EKG) while still in the womb. The raw signal will likely be corrupted by the breathing and heartbeat of the mother. A filter might be used to separate these signals so that they can be individually analysed. Signal restoration is used when a signal has been distorted in some way. For example, an audio recording made with poor equipment may be filtered to better represent the sound as it actually occurred. Another example is the de-blurring of an image acquired with an improperly focused lens, or a shaky camera [25].

These problems can be approached with either analog or digital filters. Analog filters are cheap, fast, and have a large dynamic range in both amplitude and frequency. Digital filters, in comparison, are vastly superior in the level of performance that can be achieved. For example, a low-pass digital filter can have a gain of 1 +/- 0.0002 from DC to 1000 hertz, and a gain of less than 0.0002 for frequencies above 1001 Hertz. The entire transition occurs within only 1 hertz, which is not likely to be expected from an op-amp circuit. Digital filters can achieve thousands of times better performance than analog filters [25]. This affects dramatically the way filtering problems are approached. With analog filters, the emphasis is on handling limitations of the electronics, such as the accuracy and stability of the resistors and capacitors. In comparison, digital filters are so good that the performance of the filter

is frequently ignored. The emphasis shifts to the limitations of the signals, and the theoretical issues regarding their processing.

The most straightforward way to implement a digital filter is by convolving the input signal with the digital filter's impulse response. All possible linear filters can be made in this manner. When the impulse response is used in this way, it is called the **filter kernel**. **Recursion** is another way to implement digital filters. When a filter is implemented by convolution, each sample in the output is calculated by weighting the samples in the input, and adding them together. Recursive filters are an extension of this, using previously calculated values from the output, besides points from the input. Instead of using a filter kernel, recursive filters are defined by a set of recursion coefficients. All linear filters have an impulse response, even if it is not used to implement the filter. The impulse response of a filter is simply the output of the filter when the input is an impulse. The impulse responses of recursive filters are composed of sinusoids that exponentially decay in amplitude. In principle, this makes their impulse responses infinitely long. However, the amplitude eventually drops below the round-off noise of the system, and the remaining samples can be ignored. Because of this characteristic, recursive filters are also called **Infinite Impulse Response** or **IIR filters** [25]. In comparison, filters carried out by convolution are called **Finite Impulse Response** or **FIR filters**.

The following list gives some of the main advantages of digital over analog filters [25].

- A digital filter is programmable, i.e. its operation is determined by a program stored in the processor's memory. This means that the digital filter can easily be changed without

affecting the circuitry (hardware). An analog filter can only be changed by redesigning the filter circuit.

- Digital filters are easily designed, tested and implemented on a general-purpose computer or work-station. The characteristics of analog filter circuits (particularly those containing active components) are subject to drift and are dependent on temperature. Digital filters do not suffer from these problems, and so are extremely stable with respect to both time and temperature.
- Unlike their analog counterparts, digital filters can handle low frequency signals accurately. As the speed of DSP technology continues to increase, digital filters are being applied to high frequency signals in the Radio Frequency (RF) domain, which in the past was the exclusive preserve of analog technology.
- Digital filters are very much more versatile in their ability to process signals in a variety of ways; this includes the ability of some types of digital filter to adapt to changes in the characteristics of the signal.

Fast DSP processors can handle complex combinations of filters in parallel or cascade (series), making the hardware requirements relatively simple and compact in comparison with the equivalent analog circuitry.

2.3 FIR Systems

For causal FIR systems, the system function has only zeros (except for poles at z = 0) and is given by Eq (2.1):

$$y(n) = \sum_{k=0}^{M} b_k x(n-k) \qquad \textbf{Eq (2.1)}$$

also known as the discrete convolution of x(n) with the impulse response:

$$h(n) = \begin{cases} b_n & n = 0, 1, \ldots, N \\ 0 & otherwise \end{cases} \qquad \textbf{Eq (2.2)}$$

It is important to notice that a FIR filter uses memory to recall past inputs and integrate them with a specific weighting function (i.e. b terms). Thus, a FIR filter uses no feedback, is inherently stable and will not oscillate [24]. In some applications, such as speech processing, the zero phase (or linear phase) characteristic of a filter is not very critical. The human auditory system responds to short time spectral magnitude characteristics, so the shape of a speech waveform can sometimes change drastically without the human listener's being able to distinguish it from the original.

In image processing, the linear phase characteristic appears to be more important. Our visual world consists of lines, scratches, etc. A non-linear phase distorts the proper registration of different frequency components that make up the lines and scratches. This distorts the signal shape in various ways, including blurring. The zero-phase characteristic is quite useful in applications such as image processing. And zero-phase is very easy to achieve with FIR filters due to the symmetry with respect to the origin. In addition, design and implementation are often simplified if we require zero-phase [26].

Advantages of using FIR filters are [24]:

- FIR filters can easily be designed for constant phase delay and/or constant group delay.

- Stability is inherent and limit cycling is not a problem provided the user implements the filter with non-recursive techniques.
- Round off errors can be controlled in a straightforward fashion in order to keep their effects insignificant.

Drawbacks of using FIR filters are [27]:

- A FIR generally requires many stages to obtain sharp filter bands.
- Additional stages add to memory requirements and often slow processing speed.

2.4 Basic Network Structures for FIR Systems

It will not be necessary for the purpose of this work to examine all known FIR structures. Instead, examining some of the basic and most common network structures will be sufficient for the scope of this thesis work.

2.4.1 Direct Form

Because of the chain of delay elements across the top of the diagram, this structure is also referred to as tapped delay line structure or a transversal filter structure [24].

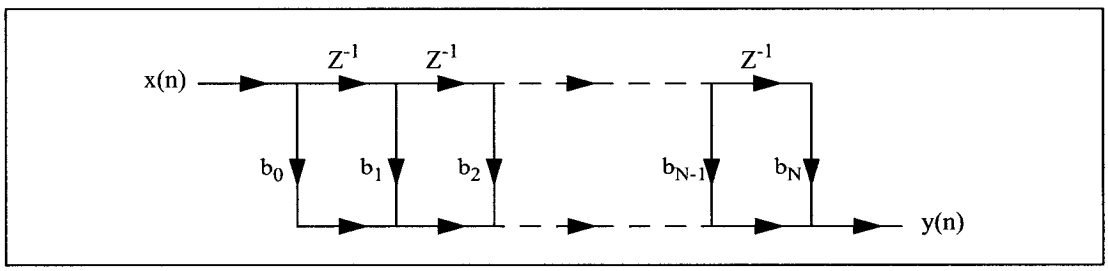

Figure 2.1 Direct Form of a FIR system

2.4.2 Transposed Form

Transposition of a flow graph is accomplished by reversing the directions of all branches in the network while keeping the branch transmittances as they were and reversing the roles of the input and output so that the source nodes become sink nodes and vice versa. For single-input/single-output systems, the resulting flow graph has the same system function as the original graph if the input and output nodes are interchanged [24]. In the Direct form structure, the delay elements are across the input data path. In the Transposed form, the delay elements are across the computed data path. The Transposed structure is illustrated in Figure 2.2.

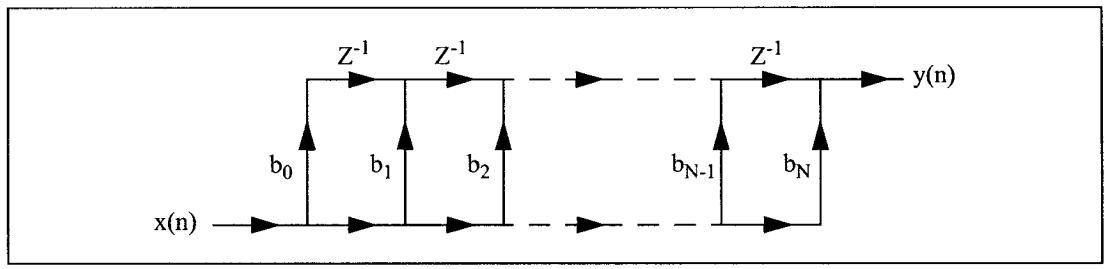

Figure 2.2 Transposed Form of a FIR system

2.5 IIR Systems

An infinite impulse response (IIR) digital filter has an impulse response that is infinite in extent. A linear time-invariant IIR system has input and output that satisfy a difference equation of the form [24]:

$$y(n) - \sum_{k=1}^{N} a_k y(n-k) = \sum_{k=0}^{M} b_k x(n-k) \qquad \textbf{Eq (2.3)}$$

Like FIR filters, IIR filters use memory to recall past inputs and integrate them with a specific weighting function (i.e. b terms). IIR filters, however, also recall past outputs and integrate them with specific weighting function (i.e. *a* terms). Recalling the past outputs is implemented by a feedback loop. With an IIR filter, controlling the phase characteristic is very difficult. It is impossible for a single recursively computable IIR filter to have zero-phase. If zero-phase is necessary, two or more IIR filters are combined to obtain zero-phase for the overall filter [26].

The main advantage of using IIR filters is that they require smaller number of coefficients, i.e. arithmetic operations, to meet a particular magnitude specification than FIR filters [26]. However, the drawbacks of using IIR filters are [26]:

- Testing filter stability and stabilizing an unstable filter without affecting the magnitude response are very big tasks.
- Zero-phase is difficult to achieve.
- High sensitivity to filter coefficient quantization errors.

2.6 Basic Network Structures for IIR Systems

An IIR filter with an arbitrary impulse response h(n) cannot be realised, since computing each output sample requires a large number of arithmetic operations. As a result, in addition to requiring h(n) to be real and stable, we require h(n) to have a rational z-transform corresponding to a recursively computable system [26]. It will not be necessary for the purpose of this work to examine all known IIR structures. Instead, examining some of the basic and most common network structures will be sufficient for the scope of this thesis work.

2.6.1 Direct Forms

The coefficients in the Direct form structures correspond directly to the coefficients of the numerator and denominator polynomials. The corresponding rational system function to Eq (2.3) is given by Eq (2.4):

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}} \qquad \textbf{Eq (2.4)}$$

From Eq (2.4) we can draw two structures: Direct form I and Direct form II illustrated in Figure 2.3 and Figure 2.4 respectively.

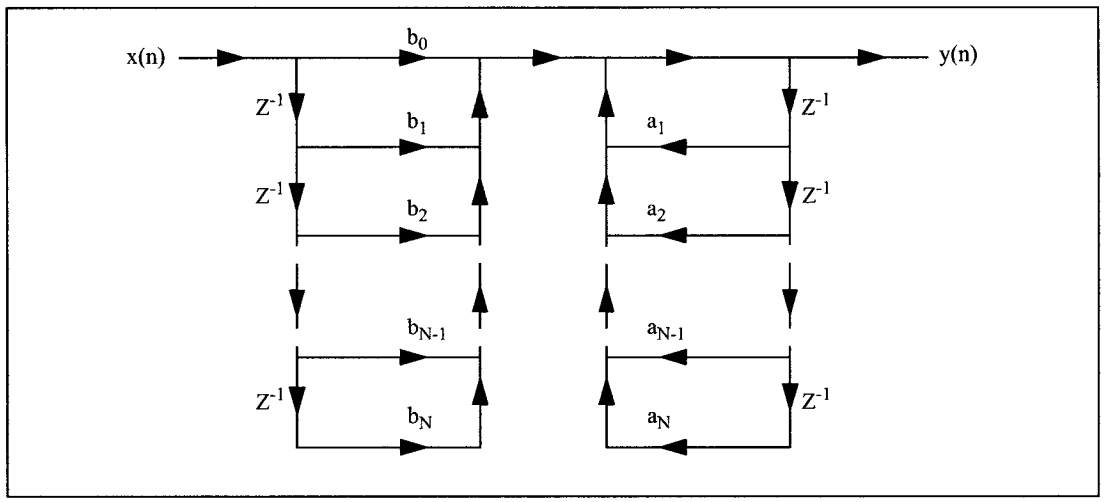

Figure 2.3 Direct Form I of an IIR system

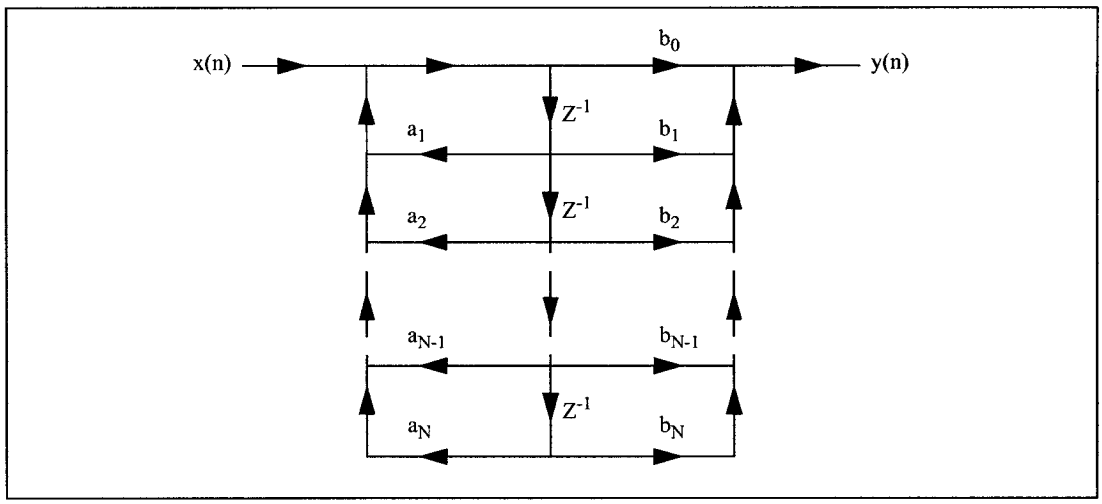

Figure 2.4 Direct Form II of an IIR system

2.6.2 Transposed Direct Forms

Applying the transposition theorem, discussed in Section 2.4.2, to Direct form II, we end up with a new network structure: Transposed Direct Form II illustrated in Figure 2.5.

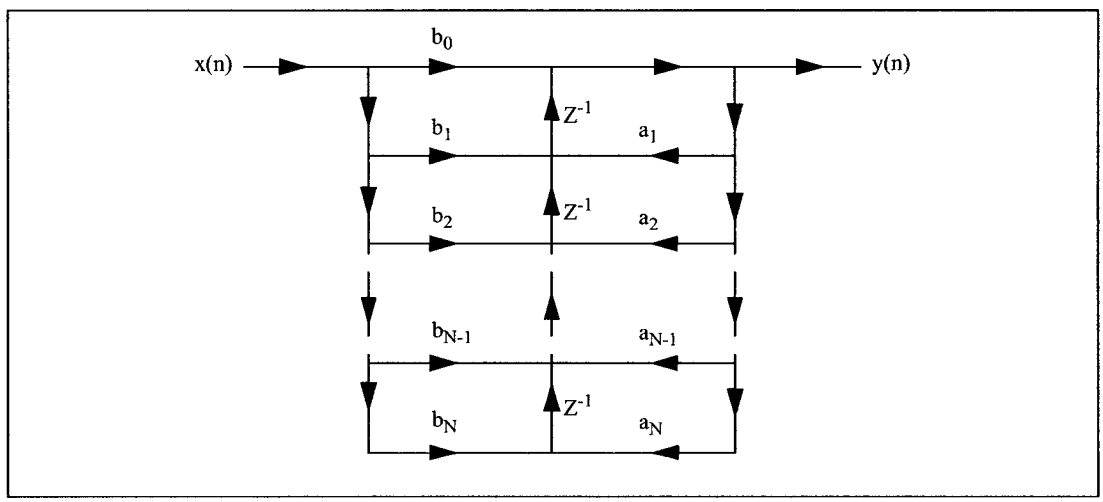

Figure 2.5 Transposed Direct Form II of an IIR system

2.6.3 Cascade Forms

If we factor the numerator and denominator polynomials in Eq (2.4), we can express H(z) in the following form [24], where $M = M_1 + M_2$ and $N = N_1 + N_2$:

$$H(z) = A\frac{\left(\prod_{k=1}^{M_1}(1-g_k z^{-1})\right)\prod_{k=1}^{M_2}(1-h_k z^{-1})(1-h^*_k z^{-1})}{\left(\prod_{k=1}^{N_1}(1-c_k z^{-1})\right)\prod_{k=1}^{N_2}(1-d_k z^{-1})(1-d^*_k z^{-1})} \qquad \textbf{Eq (2.5)}$$

In this expression, the first-order factors represent real zeros at g_k and real poles at c_k, and the second-order factors represent complex conjugate pairs of zeros at h_k and h^*_k and complex conjugate pairs of poles at d_k and d^*_k. This represents the most general distribution of poles and zeros when all the coefficients in Eq (2.4) are real. Eq (2.5) suggests a class of structures consisting of a cascade of first- and second-order systems. In practice, however, it is often desirable to implement the cascade realisation using a minimum of storage and computation. A modular structure that is advantageous for many types of implementation is obtained by combining pairs of real factors and complex conjugate pairs into second-order factors so that Eq (2.5) can be expressed by Eq (2.6) [24], where N_s is the largest integer contained in (N+1)/2:

$$H(z) = \prod_{k=1}^{N_s}\frac{b_{0k} + b_{1k}z^{-1} + b_{2k}z^{-2}}{1 - a_{1k}z^{-1} - a_{2k}z^{-2}} \qquad \textbf{Eq (2.6)}$$

The individual second-order sections (SOS) can be implemented using any of the Direct form structures in Figure 2.3, Figure 2.4 and Figure 2.5. The resulting network structures are illustrated in Figure 2.6, Figure 2.7 and Figure 2.8 respectively.

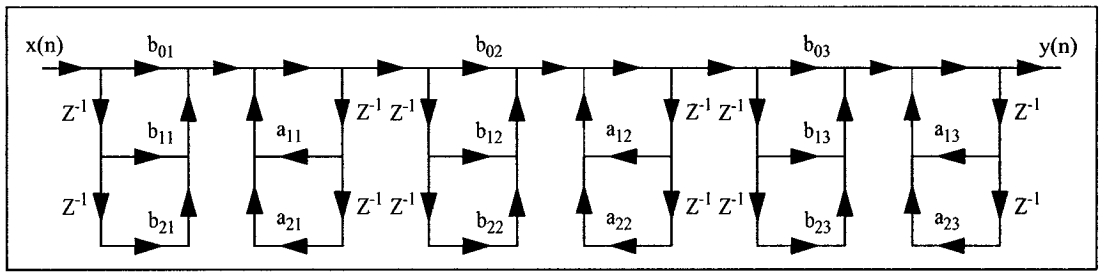

Figure 2.6 Cascade structure of a sixth-order system with Direct Form I realisation

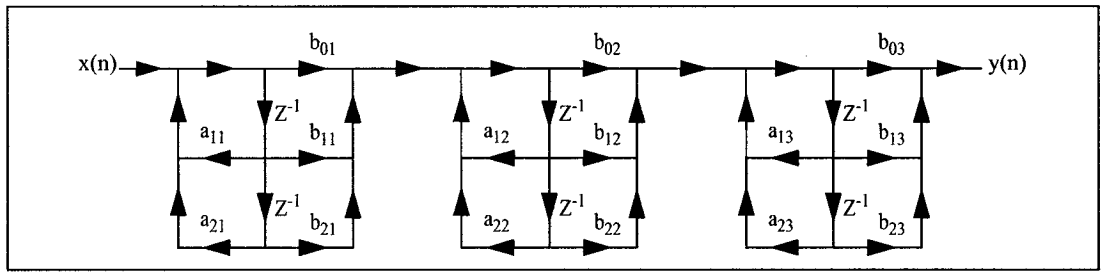

Figure 2.7 Cascade structure of a sixth-order system with Direct Form II realisation

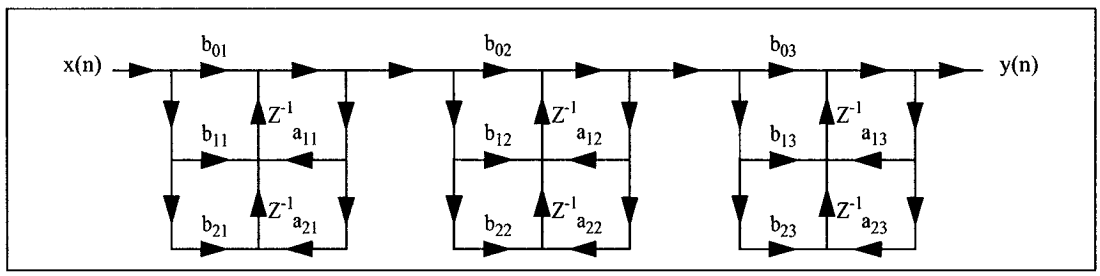

Figure 2.8 Cascade structure of a sixth-order system with Transposed Direct Form II realisation

2.7 Summary

In this chapter we have presented the basic concepts of digital filtering. We began with a brief review of the different types of digital filters and their applications. We followed by discussing the advantages and disadvantages of each type of digital filters and reviewing some of their basic common network structures. The digital filter network structures pre-

sented in this chapter are the basis for the work discussed in Chapter 4 and Chapter 5. Since these two chapters contain numerous references to the network structures discussed in this chapter, it became necessary to familiarise the reader with these structures to better understand the implications of the results reported in this thesis.

Chapter 3

Arithmetic Units

3.1 Introduction

The wide dynamic range of floating point arithmetic is attracting more and more signal processing applications to the floating point domain. This trend has become noticeable in recent years, with the advent of high speed floating point DSP integrated circuits. Advances in VLSI technology made it possible for even small computers to perform sophisticated floating point computations. In integer and fixed-point arithmetic and for the same bit width, addition takes a back seat to multiplication whether for area or for delay or for power consumption.

The objective of the present chapter is to show that in floating point arithmetic, both multiplication and addition operations are on an equal footage of importance. A secondary objective is to show that Transition Activity Scaling Algorithm [3] based floating-point arithmetic units [8] consume less power than their standard counterparts [28] [29]. Both goals are technology independent and that is demonstrated through the use of two different CMOS process technologies, 0.35 um and 0.18 um from TSMC. For all the designs, the same set of constraints is used during synthesis to ensure that it is only the difference in architecture that is affecting the final results. The rest of this chapter is organized in the following manner. Section 3.2 presents the experimental approach used to simulate and synthesize the different designs. Section 3.3 presents the results of the different experiments along with analyses of those results. Section 3.5 concludes this chapter.

3.2 Experimental Approach

Different experiments are performed to realise the aforementioned objectives. All the designs are implemented with VHDL coding and synthesised using "Design Compiler" and "Power Compiler" from Synopsys. The adder and multiplier [8] are IEEE 754 floating-point standard (single precision) compliant floating-point arithmetic units. This means that the data buses in the designs are 32-bit wide. The floating-point multiplier and adder are treated as separate entities. Rounding is performed after each operation, addition or multiplication, which complies with IEEE standard. Two sets of FP arithmetic units are evaluated, Single Data Path (SDP) units [8][28][29] and Multiple Data Path (MDP) units [3][8][22].

3.2.1 Input Data Samples

An assorted collection of audio signal samples in "wav" and "au" formats are listed in [22]. Due to the excessive amount of time required to execute each experiment, we had to limit ourselves to processing only three randomly chosen audio samples out of the batch of samples listed in [22]. The audio signal samples used are namely: Graphon.au, Quichits.wav and Danube.au. Also used is X-files.wav, a 606 KB wav-format sample of the X-files theme that was down-loaded off the world-wide-web. Moreover, to reduce the execution time of the simulation and synthesis procedures, only 64k data input vectors in floating-point format are used. All data samples are converted to IEEE 754 floating-point standard (single precision) format to be compatible with the VHDL coded designs. Converting to floating point format is performed with especially written MATLAB routines.

3.2.2 SDP Configurations

The block diagrams of the SDP FP arithmetic units are shown in Figure 3.1 and Figure 3.2. Please note that these are non pipe-lined arithmetic units.

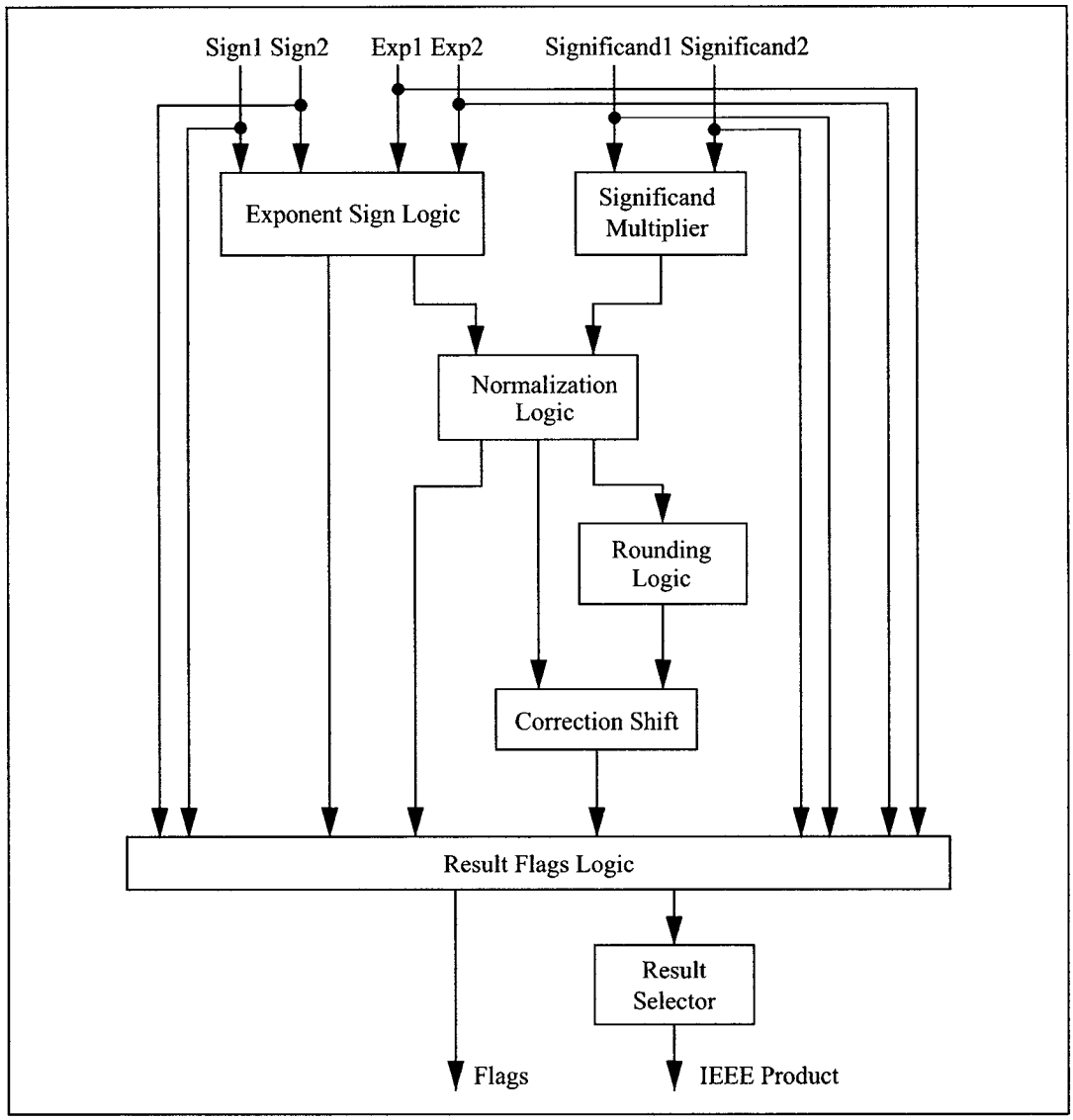

Figure 3.1 Block diagram of the Single Data Path FP multiplier

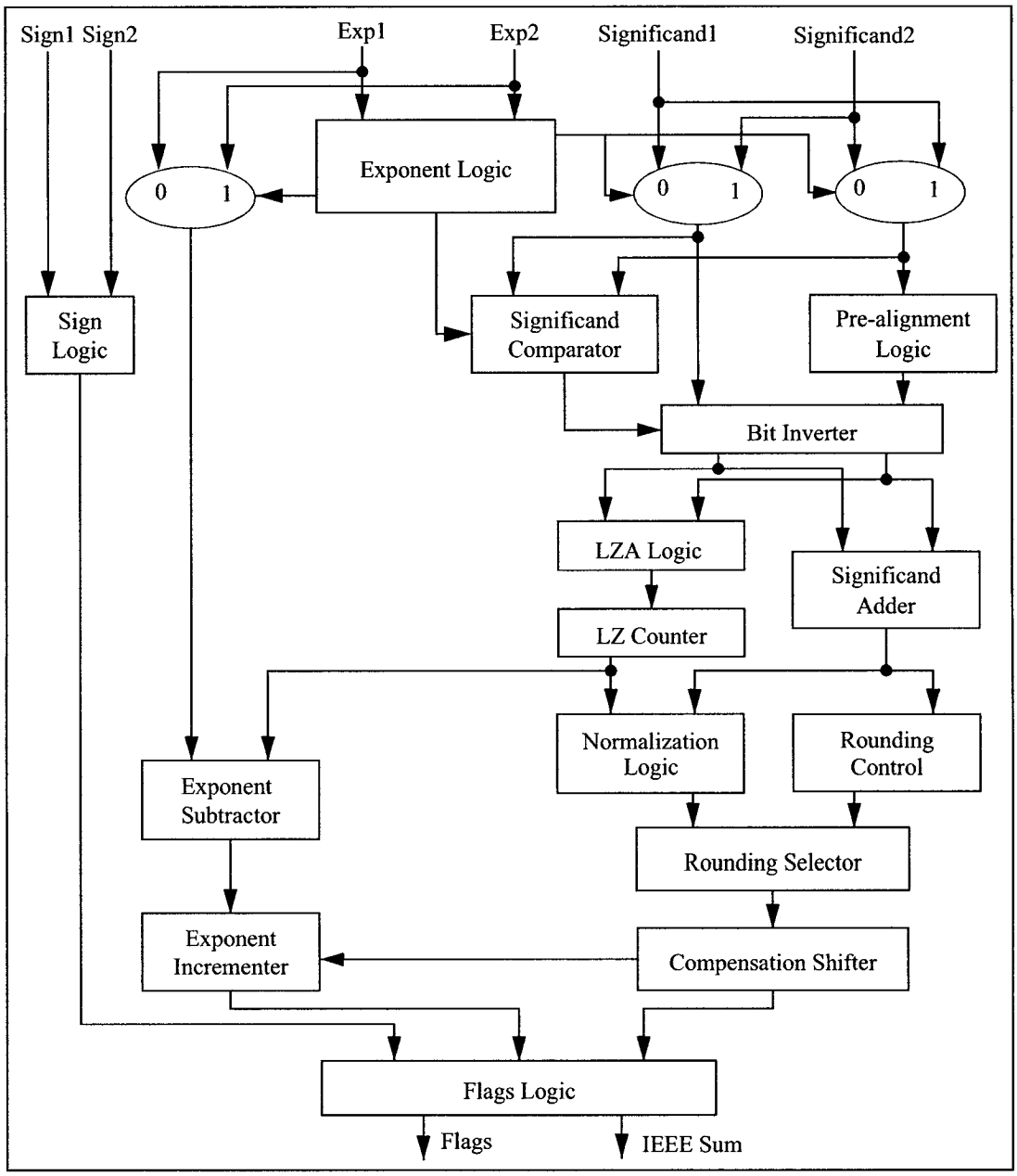

Figure 3.2 Block diagram of the Single Data Path FP adder

The arithmetic units [8] are simulated and synthesized in three different configurations. The first configuration consists of a simple component, adder [Figure 3.3, (a) & (b)] or multiplier [Figure 3.3, (c) & (d)]. The second consists of a general purpose Multiplier-Accumu-

lator (MAC) [Figure 3.4]. And the third consists of a Multiplier-Accumulator (F-MAC) having the same function we see in a transversal filter structure [Figure 3.5].

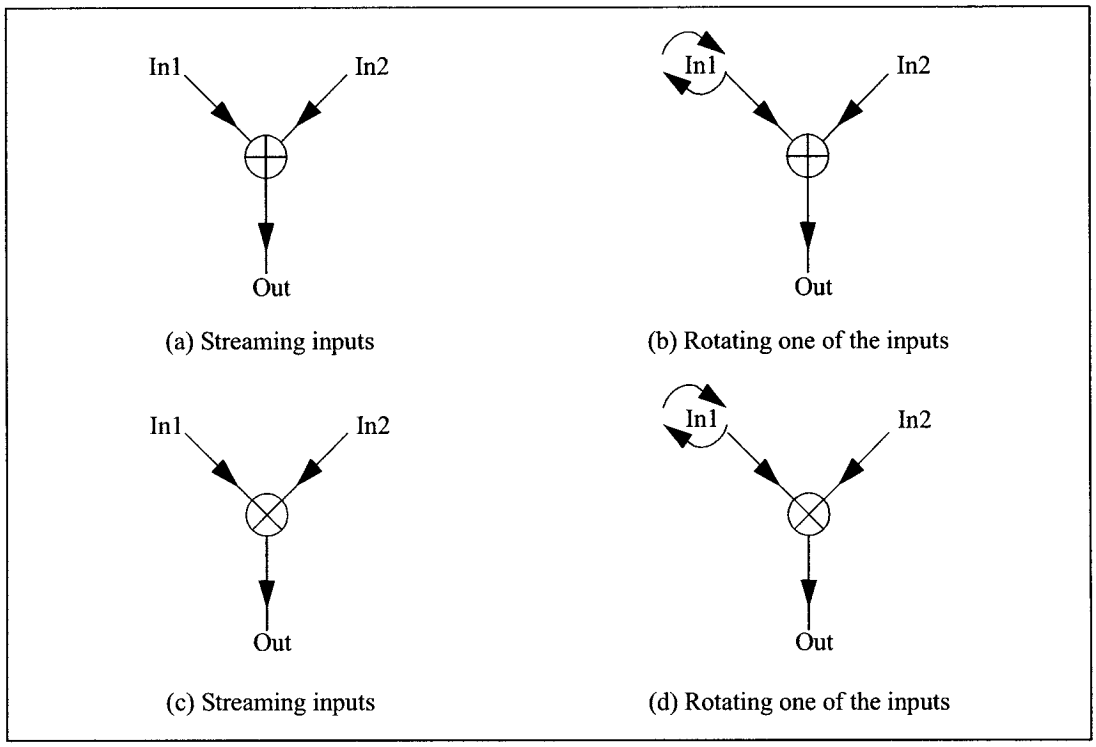

Figure 3.3 Simple Adder and Multiplier Configurations

Two simulation environments are used to capture the switching activity for the adder, the multiplier and the general purpose MAC. The first simply connects two data streams to the input [Figure 3.3 (a) & (c) and Figure 3.4 (a)]. The purpose of the second is to influence the switching activity throughout the design and to reduce the amount of occurrences of special numbers (0, denormal, $\pm\infty$ and NaN [3]). A simple and sufficient way of doing just that is to rotate a set of non-special data numbers at one of the inputs [Figure 3.3 (b) & (d)][Figure 3.4 (b)]. The other input is the same as in the first simulation environment.

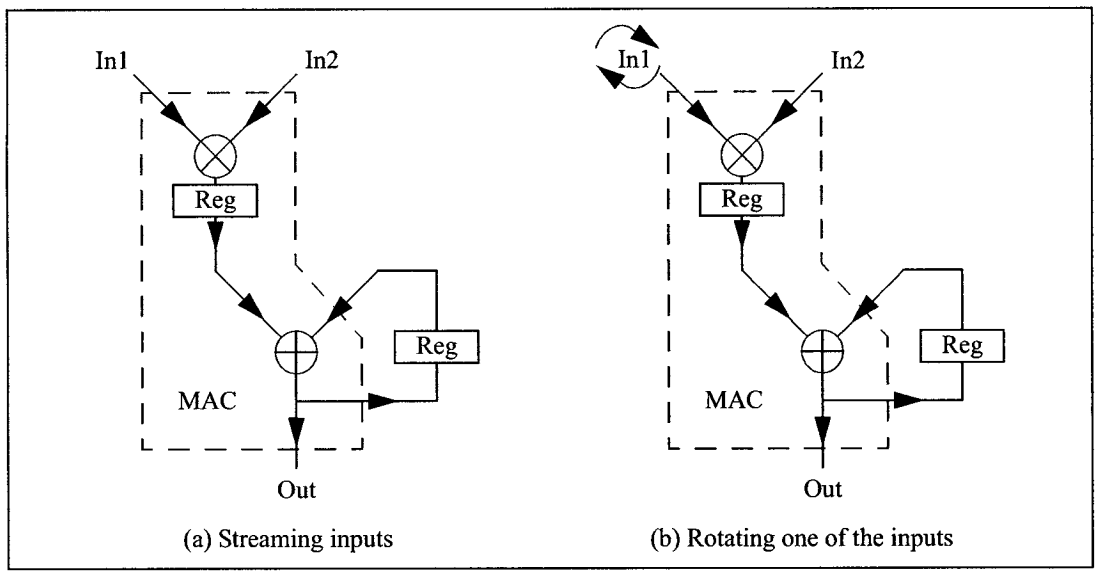

Figure 3.4 General Purpose Multiplier-Accumulator (MAC) configurations

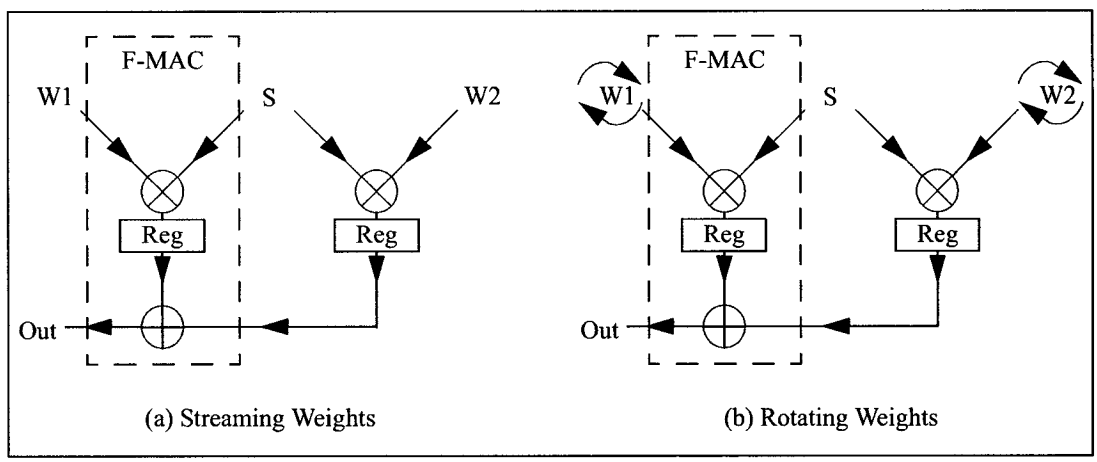

Figure 3.5 Transversal Filter Multiplier-Accumulator (F-MAC) configurations

The third configuration has two simulation environments also, slightly different from each other. One has constant multiplication weights [Figure 3.5 (a)] while the other rotates a set of weights [Figure 3.5 (b)]. For the three configurations, the rotated weights are those of an order 8 low pass FIR filter with a cut-off frequency of 0.2.

3.2.3 MDP Configurations

The block diagrams of the MDP FP arithmetic units are shown in Figure 3.6 and Figure 3.7. Please note that these are pipe-lined arithmetic units and the pipe-line register is an integral part of the design and cannot be removed without compromising the functionality of these units.

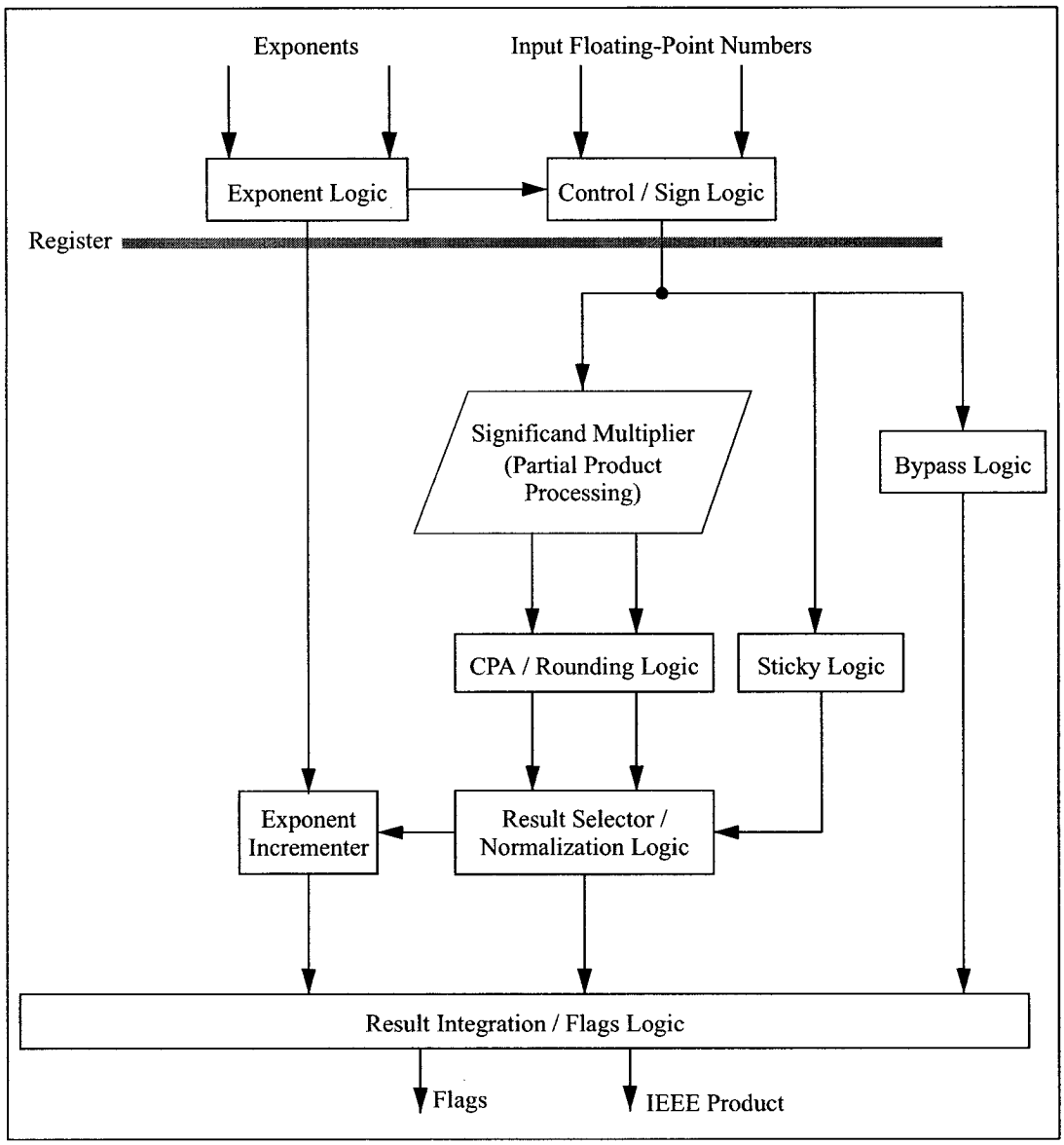

Figure 3.6 Block diagram of the Multiple Data Path FP multiplier

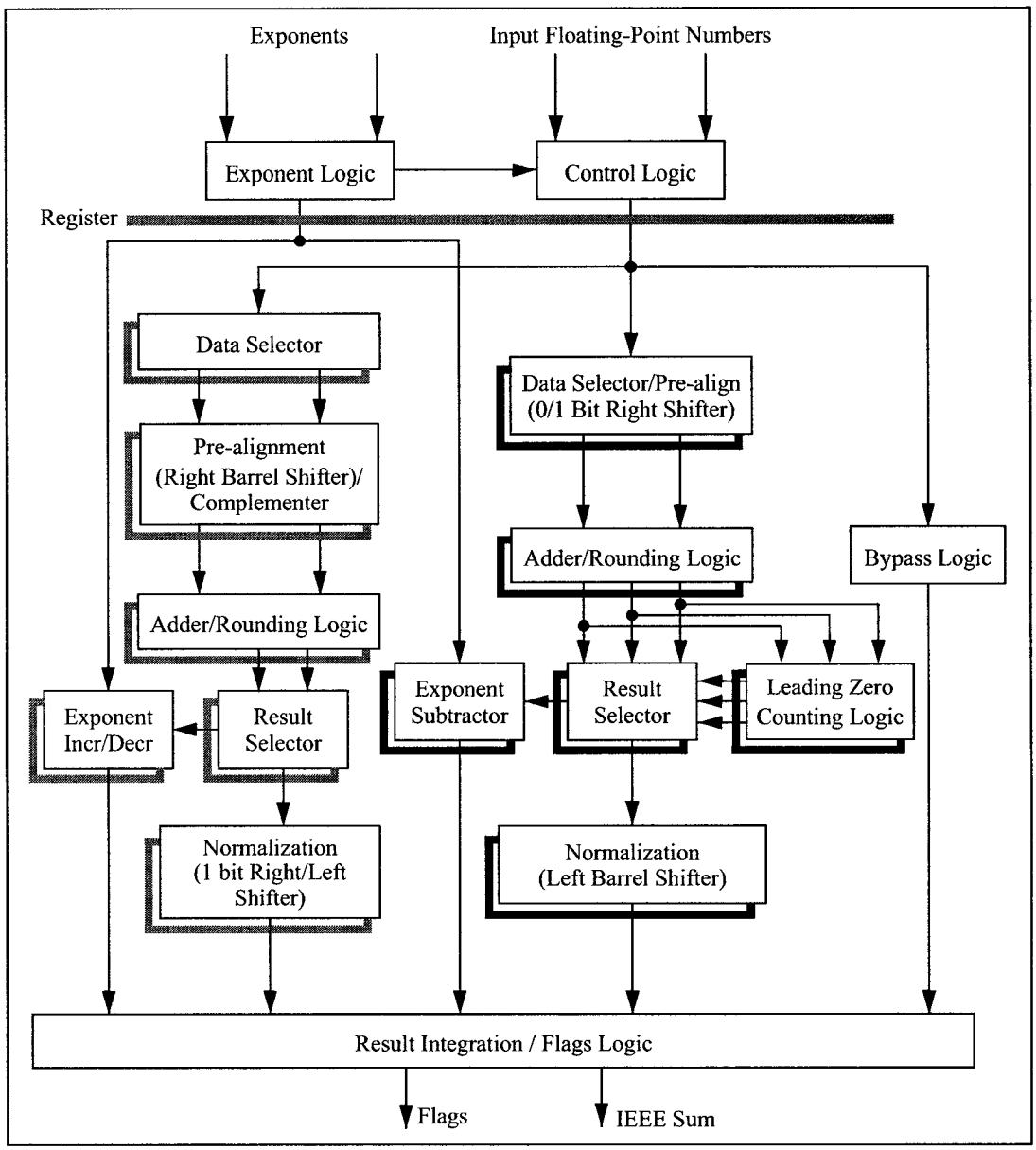

Figure 3.7 Block diagram of the Multiple Data Path FP adder

For the above reason, the design configurations seen in the previous section had to be modified to accommodate these pipe-lined units. Figure 3.8, Figure 3.9 and Figure 3.10 present the modified configurations. Please note that the registers used in the SDP design configu-

rations are now internal components of the MDP arithmetic units. The simulation environments used are the same as in Section 3.2.2.

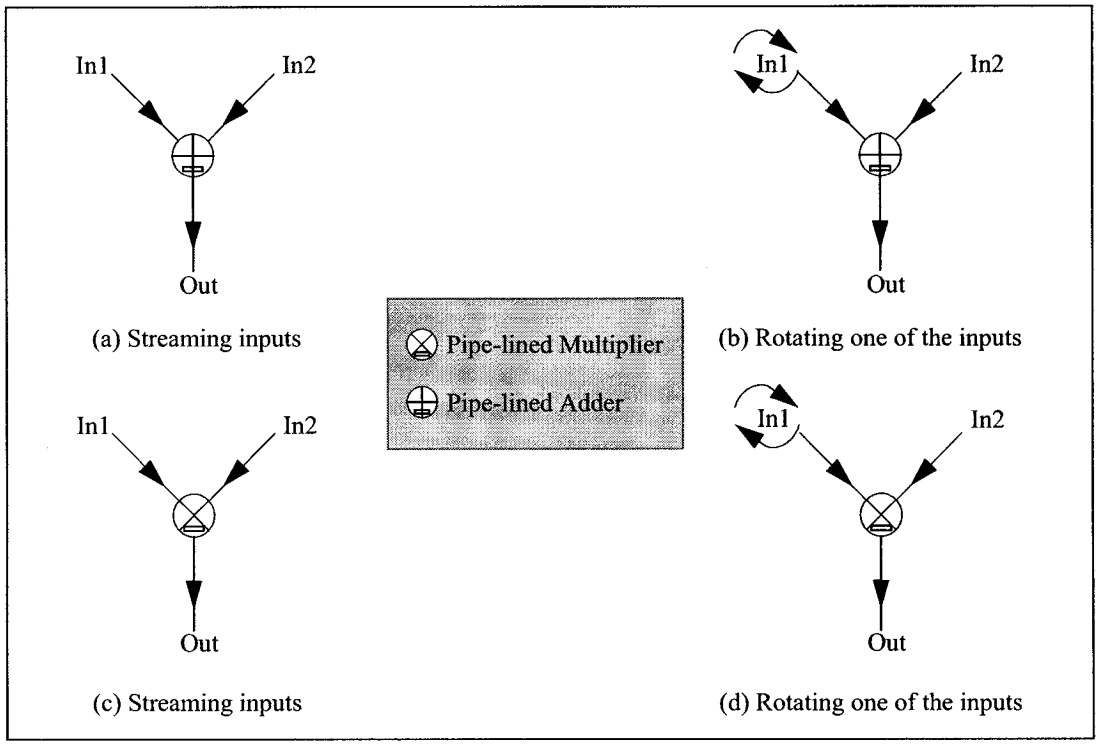

Figure 3.8 Simple Adder and Multiplier Configurations

With these configurations too, two simulation environments are used to capture the switching activity for the adder, the multiplier and the general purpose MAC. The first simply connects two data streams to the input [Figure 3.8 (a) & (c) and Figure 3.9 (a)]. The purpose of the second is to influence the switching activity throughout the design and to reduce the amount of occurrences of special numbers by rotating a set of non-special data numbers at one of the inputs [Figure 3.8 (b) & (d)][Figure 3.9 (b)]. The other input is the same as in the first simulation environment.

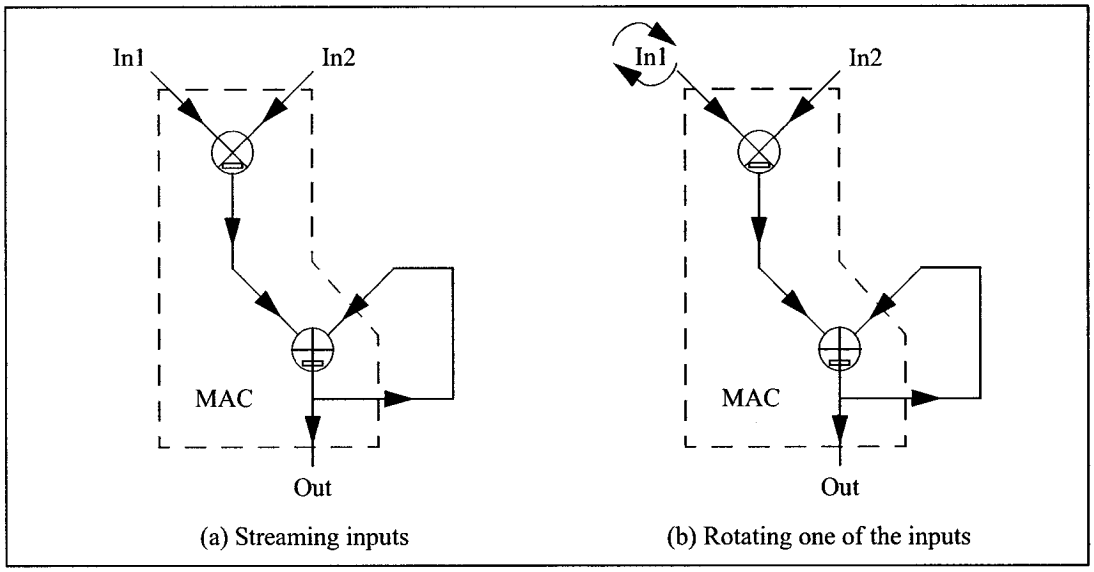

Figure 3.9 General Purpose Multiplier-Accumulator (MAC) configurations

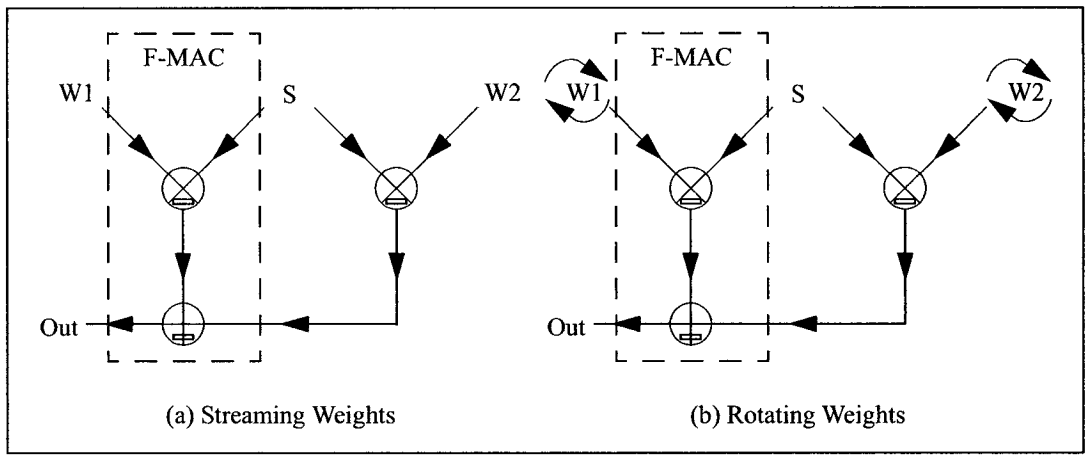

Figure 3.10 Transversal Filter Multiplier-Accumulator (F-MAC) configurations

The third configuration has two simulation environments also. One has constant multiplication weights [Figure 3.10 (a)] while the other rotates a set of weights [Figure 3.10 (b)]. Here too, the rotated weights are those of an order 8 low pass FIR filter with a cut-off frequency of 0.2.

3.3 Results

The reported measurements are those of the area (A), the critical-path delay (T) and the dynamic power (P). Also presented are the calculated values of the AT, AT^2 and PD products in Figure 3.11 and Figure 3.12. Since the leakage power is negligible to the dynamic power, its contribution to the total power consumption has no effect on the results of the comparison work. For that reason, it is not included when considering the power performance of the designs. Please note that for the same design we can have slightly different values, less than 2%, for area and delay if the input is changed. That is due to fine tuning the design during the power optimization procedure [22]. Also, please note that power dissipated by both adder and multiplier is data dependent and the variation of power dissipation with different input data is very high.

3.3.1 Single Data Path FP Arithmetic Units

The block diagrams of the multiplier and the adder are shown in Figure 3.1 and Figure 3.2. The simulation results show the highest critical path delay for the adder and the multiplier to be 59.29 ns in 0.35 um technology. For power comparison, the operating frequency of all the designs and in both technologies is set to 16.67 MHz, corresponding to a clock signal period of 60 ns. The power performance figures are measured at the above frequency.

Table 3.1 A, T and P figures in simple configurations

Simple Config		Adder			Multiplier		
Data	Tech. um	A um^2	T ns	P mW	A um^2	T ns	P mW
Str	0.35	174195	59.15	3.17	441735	59.15	3.73

Table 3.1 A, T and P figures in simple configurations

Simple Config		Adder			Multiplier		
Data	Tech. um	A um^2	T ns	P mW	A um^2	T ns	P mW
Rot	0.35	169855	59.07	5.81	420096	59.15	14.13
Str	0.18	33681	30.57	0.62	78309	34.33	1.40
Rot	0.18	32486	31.16	0.75	78833	32.31	1.77

Table 3.2 A, T and P figures in MAC configuration

MAC		Adder			Multiplier		
Data	Tech. um	A um^2	T ns	P mW	A um^2	T ns	P mW
Str	0.35	170406	57.97	3.83	391833	59.29	7.90
Rot	0.35	166923	58.00	5.18	395246	59.22	23.25
Str	0.18	32689	30.98	0.52	77949	34.34	1.36
Rot	0.18	32297	31.88	0.57	78561	32.82	1.65

Table 3.3 A, T and P figures in F-MAC configuration

F-MAC		Adder			Multiplier		
Data	Tech. um	A um^2	T ns	P mW	A um^2	T ns	P mW
Str	0.35	168875	58.12	5.56	408502	59.19	13.94
Rot	0.35	166932	58.09	6.21	395263	59.23	23.23
Str	0.18	32526	31.73	0.57	77998	33.22	1.59
Rot	0.18	32252	31.43	0.62	78565	32.76	1.65

Examining the results of Table 3.1 through Table 3.3 for both adder and multiplier, in the configurations illustrated in Figure 3.3, Figure 3.4 and Figure 3.5, we clearly see that the adder cannot be considered negligible compared to the multiplier in terms of performance. It is obvious that the adder has a smaller area than the multiplier, but it is only by around 60% in both technologies. Comparing the critical-path delays, we find that they differ by a maximum of only 10%. The Area Delay and Area Delay^2 products presented in Figure 3.11 show a clearer image of the above discussion.

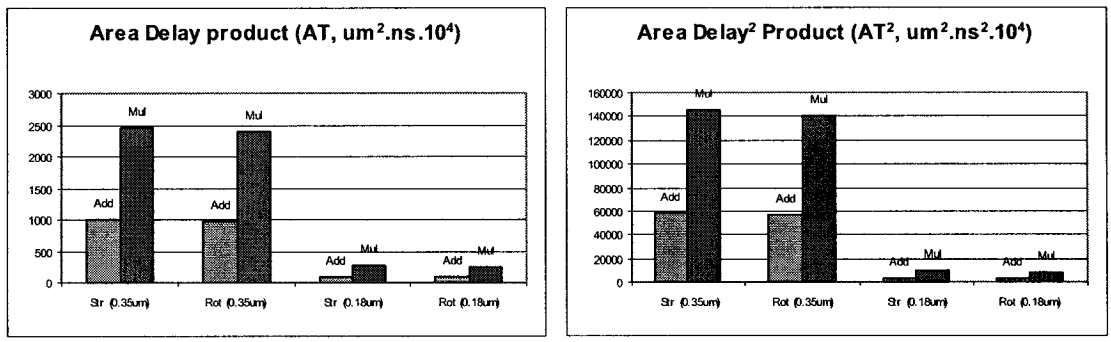

Figure 3.11 Average values of AT and AT^2 products for the SDP units

Comparing the power performance, we find the adder to be consuming less than the multiplier by an average reaching 67% in 0.35 um technology and 64% in 0.18 um technology. That is clearly reflected in the Power Delay product presented in Figure 3.12. Here too, we find the effect of the adder to be non negligible in comparison to that of the multiplier. This is due to the high level of complexity of the FP addition compared to that of the FP multiplication. The added complexity of the addition algorithm narrows the performance gap between adders and multipliers in the FP domain.

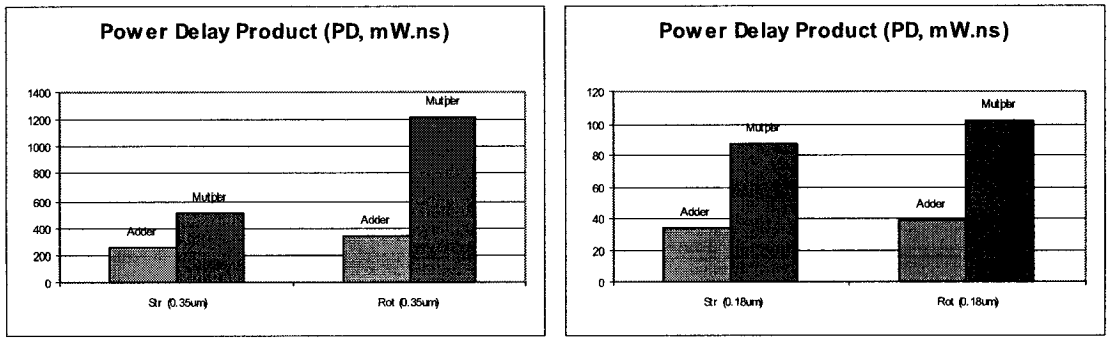

Figure 3.12 Average values of PD product for the SDP units

3.3.2 Multiple Data Path FP Arithmetic Units

The block diagrams of the multiplier and the adder are shown in Figure 3.6 and Figure 3.7. Here too, the power consumption is measured at an operating frequency of 16.67 MHz, corresponding to a clock signal period of 60 ns. Examining the results of Table 3.4 through Table 3.6 for both, adder and multiplier, in the configurations illustrated in Figure 3.3, Figure 3.4 and Figure 3.5, we see once again that the adder cannot be considered negligible compared to the multiplier.

Table 3.4 A, T and P figures in simple configurations

Simple Config		Adder			Multiplier		
Data	Tech. um	A um^2	T ns	P mW	A um^2	T ns	P mW
Str	0.35	271329	30.15	2.78	396314	30.35	3.34
Rot	0.35	259691	29.44	3.85	407479	30.09	8.49
Str	0.18	53227	16.23	0.46	95174	16.96	0.50
Rot	0.18	49771	15.91	0.59	94017	16.74	0.96

Table 3.5 A, T and P figures in MAC configuration

MAC		Adder			Multiplier		
Data	Tech. um	A um^2	T ns	P mW	A um^2	T ns	P mW
Str	0.35	255579	29.99	2.36	394424	29.53	3.04
Rot	0.35	249008	30.53	2.92	405283	30.03	8.06
Str	0.18	53725	15.78	0.33	95578	16.69	0.45
Rot	0.18	51143	16.06	0.41	94263	16.67	0.45

Table 3.6 A, T and P figures in F-MAC configuration

F-MAC		Adder			Multiplier		
Data	Tech. um	A um^2	T ns	P mW	A um^2	T ns	P mW
Str	0.35	244519	28.35	2.62	407313	29.91	4.84
Rot	0.35	238070	28.77	3.20	405554	30.25	8.01
Str	0.18	49600	16.25	0.46	95548	17.41	0.60
Rot	0.18	49310	16.26	0.51	93741	16.62	0.45

The adder, here, has a smaller area than the multiplier, but it is only by around 45% in both technologies. Comparing the critical-path delays, we find that they differ by a maximum of only 4%. The Area Delay and Area Delay^2 products presented in Figure 3.13 reflect the above discussion. Comparing the power performance, we find the adder to be consuming less than the multiplier by an average reaching 64% in 0.35 um technology and 40% in 0.18 um technology. That is clearly reflected in the Power Delay product presented in Figure

3.14. Here too, we find the effect of the adder to be non negligible in comparison to that of the multiplier. In fact, in some cases we find the adder to have a 13% higher power consumption than the multiplier. This is due to the high level of complexity of the FP addition compared to that of the FP multiplication. The higher complexity of the transition activity scaling algorithms, especially the addition's, narrows the performance gap between adders and multipliers in the FP domain. In fact, that gap is considerably diminished to the point where it is possible for adders to dissipate more power than multipliers [Table 3.6].

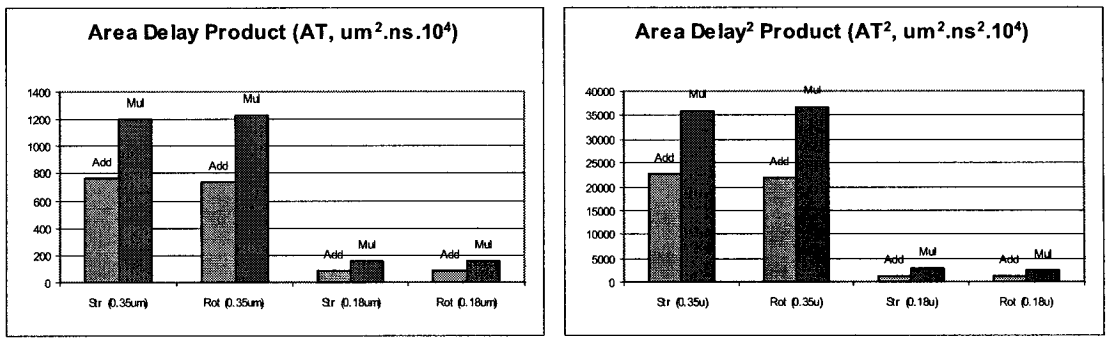

Figure 3.13 Average values of AT and AT^2 products for the MDP units

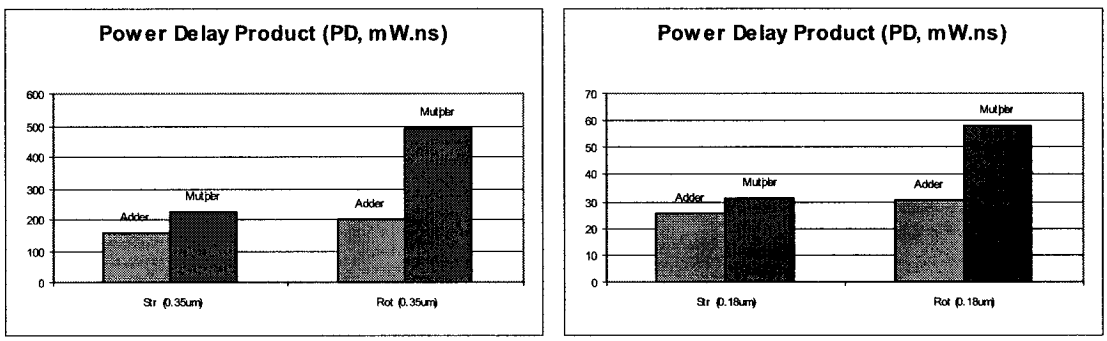

Figure 3.14 Average values of PD product for the MDP units

3.4 Floating-Point vs. Fixed-Point

As we mentioned at the beginning of this chapter, the importance of the performance of multipliers outweigh significantly that of adders in the fixed-point domain. Under the same operating conditions, fixed-point and floating-point units have widely different performances. A typical fixed-point adder has an area around 13% that of a typical fixed-point multiplier while in the floating-point domain it is around 40% [Figure 3.15]. That is mainly due to the increased algorithm complexity of the floating-point addition.

The power performance sees a similar change when migrating from fixed-point to floating-point. In fixed-point, the power dissipation of the adder is around 16% that of the multiplier while in floating-point it is around 40% [Figure 3.16]. In fixed-point, the critical path delay of the adder is around 65% that of the multiplier while it is around 95% in floating-point [Figure 3.17].

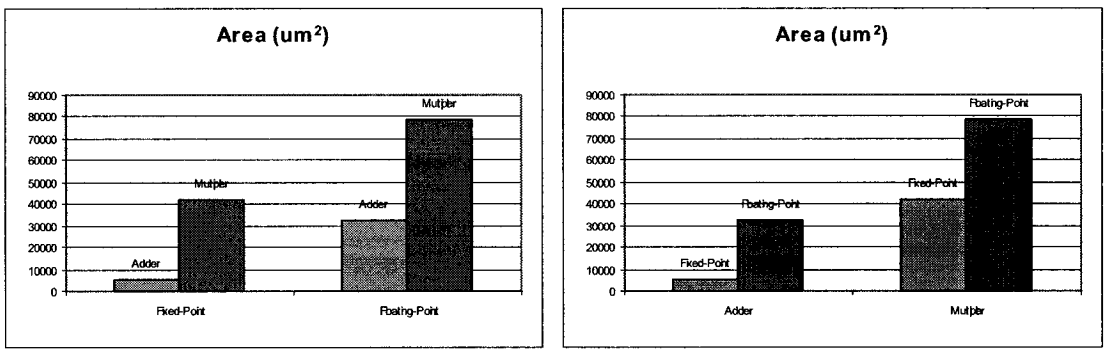

Figure 3.15 Area Performance Figures of the Arithmetic Units

Although migrating from fixed-point to floating-point arithmetic is generally accompanied by an increase in area, delay and power dissipation, the percentage increase differ signifi-

cantly between adders and multipliers. The fixed-point adder is around 17% the floating-point's, has a critical path delay that is around 75% smaller and dissipates around 74% less power. The fixed-point multiplier, on the other hand, is more than half the size of the floating-point's. Its critical path delay is around 60% smaller but it dissipates only around 37% less power. The above is clearly illustrated in Figure 3.15, Figure 3.16 and Figure 3.17.

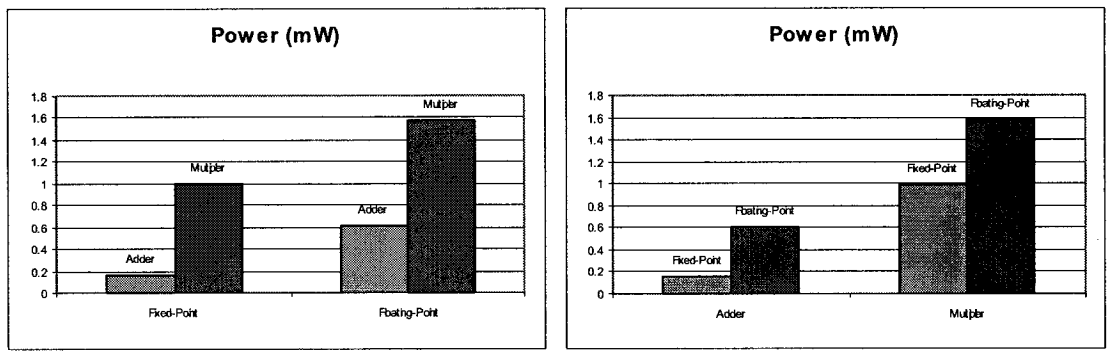

Figure 3.16 Power Performance Figures of the Arithmetic Units

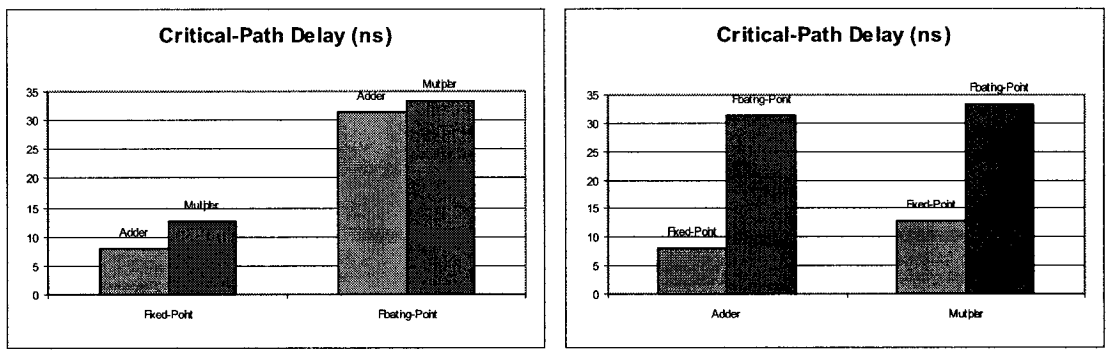

Figure 3.17 Delay Performance Figures of the Arithmetic Units

3.5 Conclusion

From the experiments performed on the floating-point arithmetic units, we can conclude that both, adder and multiplier, units are equally important in the floating-point domain. That is more visible for the MDP designs, where the adder is around 45% smaller in size than the multiplier and consumes less power by at most 64%. For SDP units the adder is 60% smaller than the multiplier and consumes less power by an average reaching 67% in 0.35 um technology and 64% in 0.18 um technology. Critical-path delays for the adder and the multiplier are very close for both MDP and SDP designs. The difference does not exceed 4% for MDP units and 10% for SDP units. Of interest is the fact that in some cases the MDP adder has a larger delay than the multiplier.

Another conclusion we can draw from the experiments performed on the floating-point arithmetic units. We can conclude that the multiple data path units have a definite advantage over their single data path counterparts. We see, from the results we have gathered, that the MDP units consume less power than the SDP units, and require less time to perform their function. They suffer, however, from increased area, especially in the case of the adder, which is around 1.5 times the size of its SDP counterpart. When considering the performance products (AT, AT^2 and PD) presented in Figure 3.18, Figure 3.19 and Figure 3.20, we find the MDP units to have a significant advantage over their SDP counterparts by a large percentage. This is true with both, 0.35 um and 0.18 um CMOS technologies. The result of the work presented in this section justifies revisiting some of the basic digital FIR and IIR filter architectures and reporting on their performance.

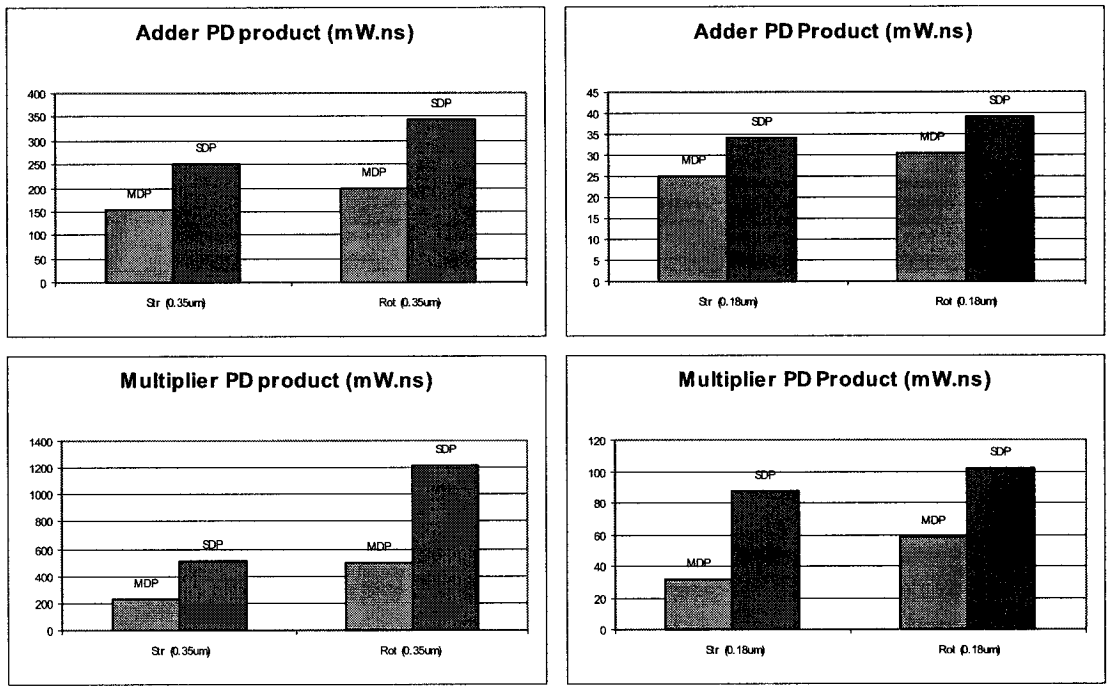

Figure 3.18 Average values of PD product for the FP units

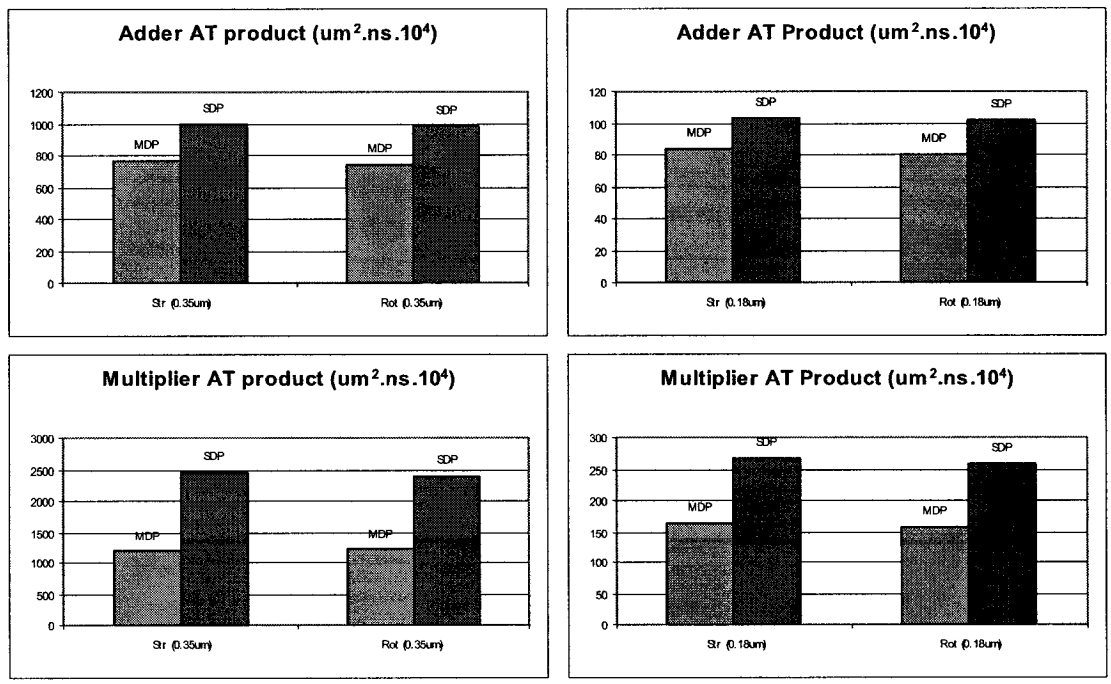

Figure 3.19 Average values of AT product for the FP units

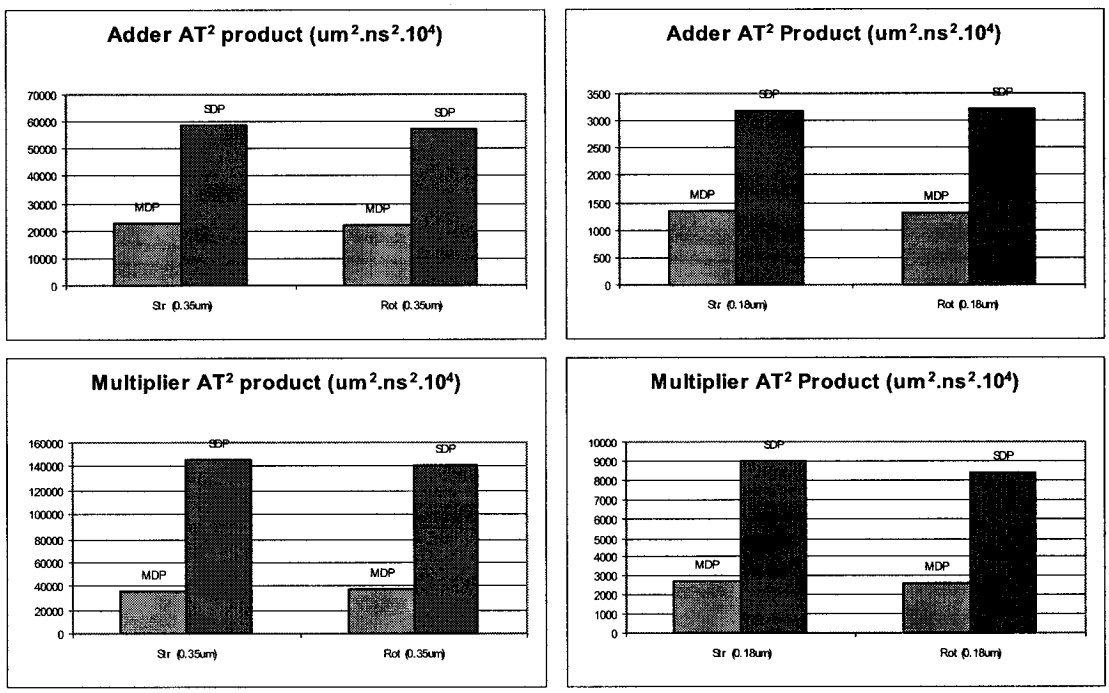

Figure 3.20 Average values of AT^2 product for the FP units

3.6 Summary

This chapter considered the difference in performance between adders and multipliers in the floating-point domain. We presented the different configurations in which the arithmetic units were evaluated. The data samples are also listed to avoid any future ambiguity. The results obtained for each of the configurations are presented in the form of tables and analysed. The conclusion of this chapter presents two points. First, both adders and multipliers are equally important in terms of power consumption, delay and area in the floating-point domain. The second point is that arithmetic units using transition activity scaling algorithm consume less power than other floating-point arithmetic units under the same operating conditions.

Chapter 4

Finite Impulse Response Filters

4.1 Introduction

In DSP applications, filtering of data samples is a very demanding operation. With the help of widely available CAD tools and with a given set of filter specifications, an appropriate filter can rapidly be obtained. The wide dynamic range and higher accuracy of floating point arithmetic is attracting more and more signal processing applications to the floating point domain. This trend has become noticeable in recent years, with the advent of high speed floating point DSP integrated circuits. Advances in VLSI technology made it possible for even small computers to perform sophisticated floating point computations.

A filter transfer function can have different implementations without modifying the filter characteristics. In Chapter 2, we have seen that FIR filters can be implemented in Direct and Transposed forms while the filter transfer function remained the same. The memory requirement for the different realisations is not necessarily the same. In general, differences in realisations imply differences in performance as well. In this chapter we will consider the different FIR filters realisations discussed in Chapter 2. We will show how the different FIR filters network structures presented in Chapter 2 were modified to accommodate the pipe-lined floating-point arithmetic units. A comparative study of area, delay and power consumption performance of the different realisations is discussed in this chapter. In this chapter, we attempt to realise several implementations of the same filter and quantify their performance with the objective of classifying them in accordance with their performance.

The rest of this chapter is organized in the following manner. Section 4.2 presents an area, delay and power performance evaluation of three standard FIR filter network structures. Section 4.3 presents an area, delay and power performance evaluation of three pipe-lined FIR filter realisations and shows how they were derived from the structures presented in Section 4.2. Section 4.3 also presents the experimental approach used to simulate and synthesize the different designs along with analyses of the results. Section 4.5 draws conclusions from the results while Section 4.6 presents a summary for this chapter.

4.2 Standard FIR Filter Network Structures

The common and basic network structures can be found in any literature discussing digital filters [24][30]. We consider three common FIR filter realisations; namely Direct, Transposed and Direct with an adder tree forms. The tree arrangement of the adders reduces by half the critical path delay across the chain of adders of the standard Direct form structure. This structure will be referred to as "Direct Tree" in the remainder of this thesis.

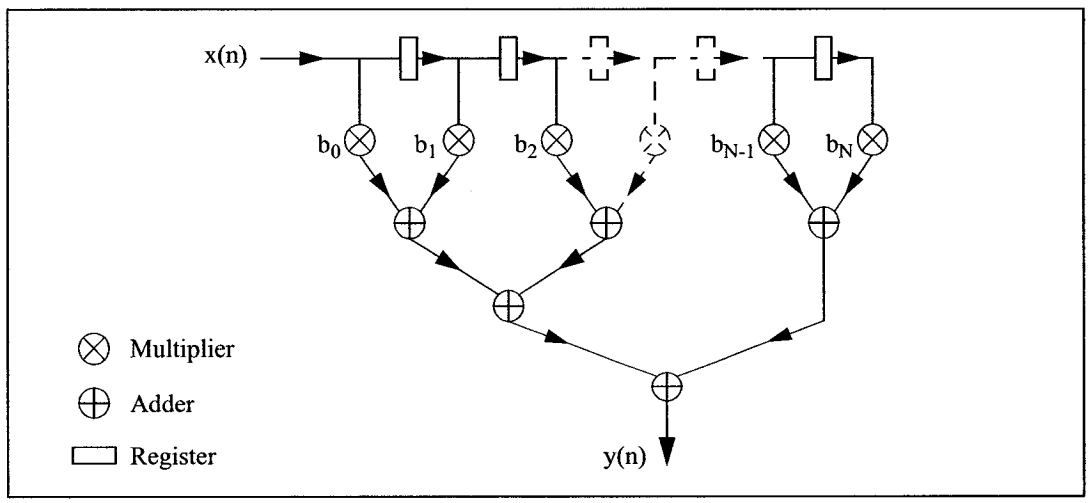

Figure 4.1 Standard Direct Tree Form

In standard form structures the arithmetic units are purely combinational, i.e. no pipe-line stages. The one clock cycle delay between two adjacent filter taps is achieved by inserting delay elements between the taps, whether across the input data path (Direct form) or across the computed data path (Transposed form). Figure 4.1, Figure 4.2 and Figure 4.3 illustrate the standard Direct Tree form, Direct form and Transposed form realisations respectively.

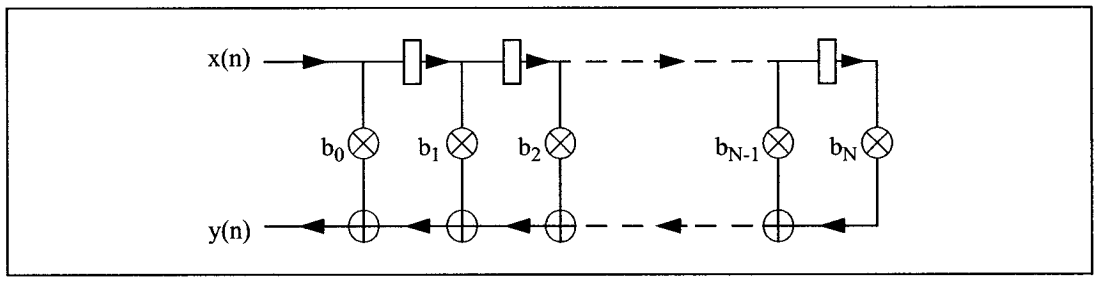

Figure 4.2 Standard Direct Form

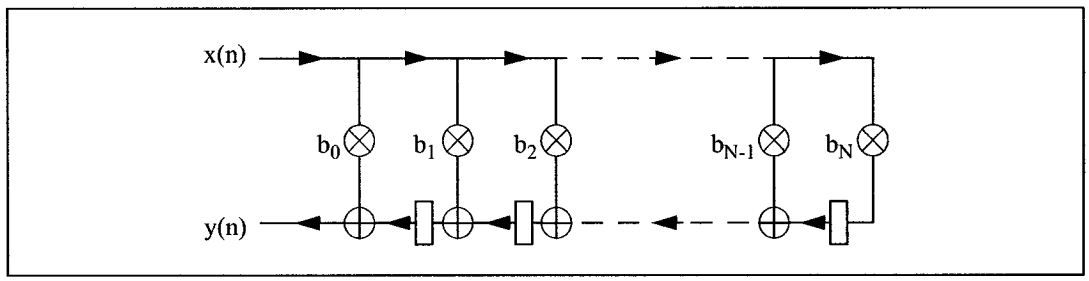

Figure 4.3 Standard Transposed Form

4.2.1 Experimental Approach - Experiment 2

The filter coefficients used are those of order 8 low pass, high pass, band pass and band stop filters. LP and HP filters have a cut-off frequency of 0.2 while the BP and BS filters have a centre frequency of 0.2. Due to very time consuming gate-level simulations, only order 8 filters are simulated in the interest of saving time. The operating frequency used is 5.5 MHz

corresponding to a clock signal period of 182 ns. This frequency corresponds to the critical path delay of a Direct form order 8 FIR filter implemented in 0.35 um technology. The 0.35 um technology was discarded in favour of 0.18 um technology when it became available for research purposes. Two audio signal samples are used, namely: Graphon.au, X-files.wav mentioned earlier. Here too, 64k data input vectors in floating-point format are used. Area, delay and power performance of the three filter structures were evaluated with standard non pipe-lined single data path FP arithmetic units [8][28][29], shown in Figure 3.1 and Figure 3.2.

4.2.2 Results (non pipe-lined)

From Table 4.1 and Table 4.2, and as expected, we see that Direct form realisation has the largest critical path delay, which is due to the adders chain. Direct Tree form realisation has a smaller delay as the signal traverses a smaller number of adders. Transposed form realisation has the smallest delay of the three because of the registers placed at the output of each tap. Differences in area are negligible, less than 0.5%, and are mostly due to synthesis optimization. Power performance of Direct form and Direct Tree form realisations are quasi equal with a difference of less than half a percentage point.

Table 4.1 A, T and P figures for order 8 low pass and high pass filters

Structure	Low Pass			High Pass		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct	1648464	105.69	9.86	1648444	105.69	9.97
Direct Tree	1648354	62.07	9.86	1648435	62.07	9.99
Transposed	1644532	28.20	13.37	1644552	28.20	13.65

Table 4.2 A, T and P figures for order 8 band pass and band stop filters

Structure	Band Pass			Band Stop		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct	1648444	105.69	9.86	1648444	105.69	9.41
Direct Tree	1648435	62.07	9.86	1648435	62.07	9.43
Transposed	1644573	28.20	13.48	1644573	28.20	13.17

Transposed form realisation, however, has the worst power performance with a dissipation of around 30% higher than Direct form and Direct Tree form realisations. Pipe-lining is a standard technique for reducing power dissipation [30] and critical path delay. Since the emphasis of this work is low power, it was decided to evaluate the effect of pipe-lining on the performance of the Direct, Direct Tree and Transposed form realisations.

4.3 Pipe-lined FIR Network Structures

The objective here is to implement the different FIR filter network structures discussed in Chapter 2 using the arithmetic units from Chapter 3. In Chapter 3, two sets of FP arithmetic units were presented. One set consisted of non pipe-lined units [8][28][29], shown in Figure 3.1 and Figure 3.2., while the other consisted of pipe-lined arithmetic units [3][8][22] shown in Figure 3.6 and Figure 3.7. Pipe-lining can be introduced to the non pipe-lined units simply by buffering their outputs. The output buffers result in pipe-lining the arithmetic units and, subsequently, the filter structures. Introducing pipe-lining in the adder and the multiplier is a standard technique of enhancing the performance of these units.

4.3.1 Problem Definition

A simple way of looking at a pipe-lined component is to consider it as a purely combinational component that has a delay unit at its output (or input, the behaviour would be the same). Figure 4.4 and Figure 4.5 show the structures of Figure 4.2 and Figure 4.3 with added delay units at the output of each arithmetic unit to simulate the effect of pipe-lining.

When closely examining the structures of Figure 4.4 and Figure 4.5, we notice that, instead of the desired one clock cycle delay, the delay between two adjacent filter taps is now two clock cycles. This means that the next result of tap I will be transmitted to tap I-1 before the previous result of tap I+1 is even ready. That results in the structure no longer performing the FIR filter function.

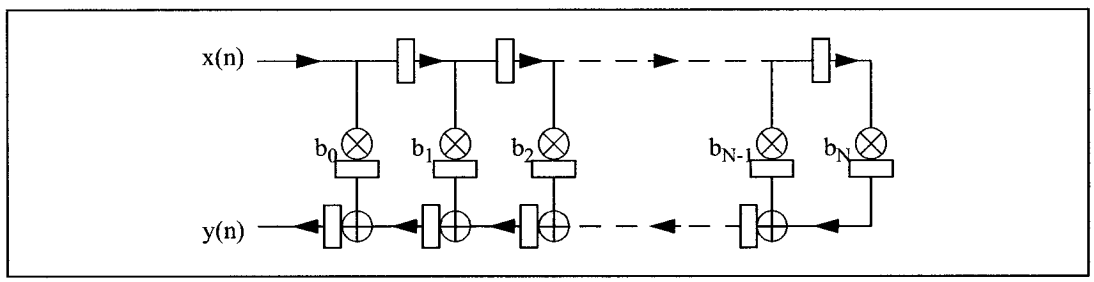

Figure 4.4 Standard Direct Form of a FIR system with simulated pipe-lined components

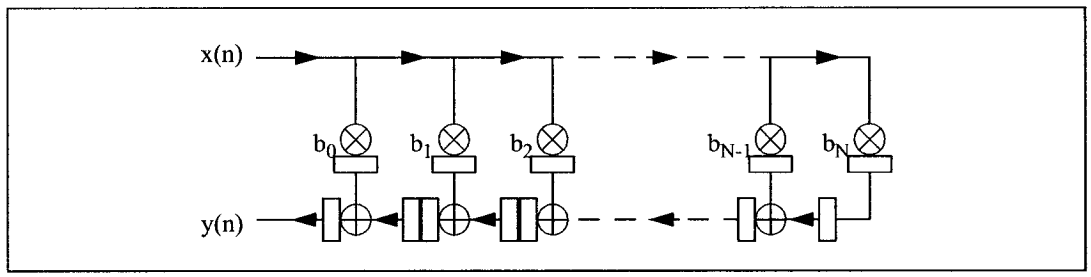

Figure 4.5 Standard Transposed Form of a FIR system with simulated pipe-lined components

4.3.2 Problem Solving (Re-timing)

To restore the correct filter function, we need to re-time the structures to have only one clock cycle delay between two adjacent filter taps. Two approaches were considered and compared. However, only one of them was adopted after comparison of results from each.

4.3.2.1 First Approach

The first approach to re-time the structures consisted of inserting delay elements in each tap. Figure 4.6 and Figure 4.7 show the re-timed structures of Figure 4.4 and Figure 4.5.

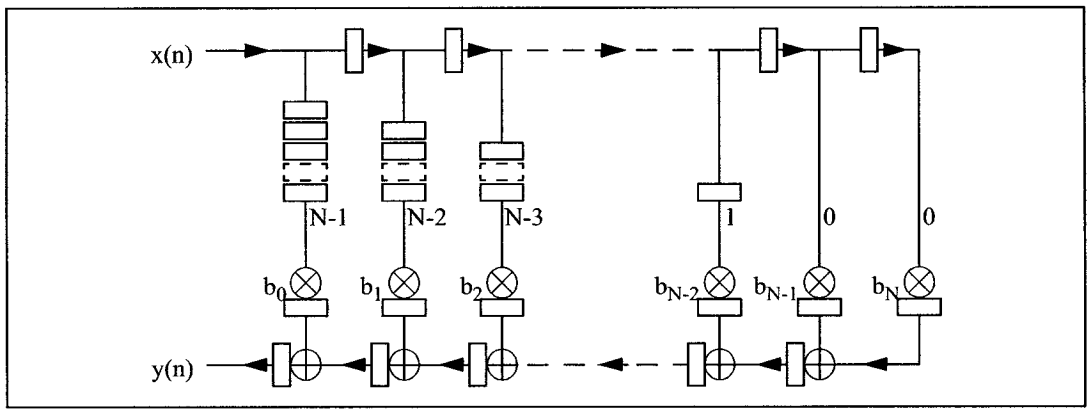

Figure 4.6 Approach 1 re-timed FIR Direct Form with simulated pipe-lined components

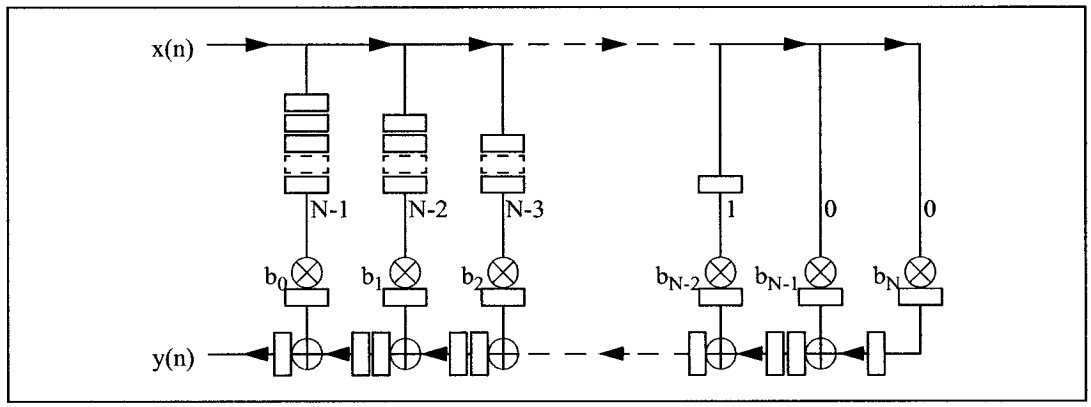

Figure 4.7 Approach 1 re-timed FIR Transposed Form with simulated pipe-lined components

The number of delay elements to be inserted in tap I is equal to (N - 1 - I), with N being the order of the filter and $I \in [0, N-1]$. Each of our pipe-lined arithmetic units is equivalent to one combinational arithmetic unit combined with one delay unit. Figure 4.8 and Figure 4.9 show the re-timed Direct and Transposed form structures using pipe-lined arithmetic units. Adding extra delay elements does solve the problem of re timing-the filter structures.

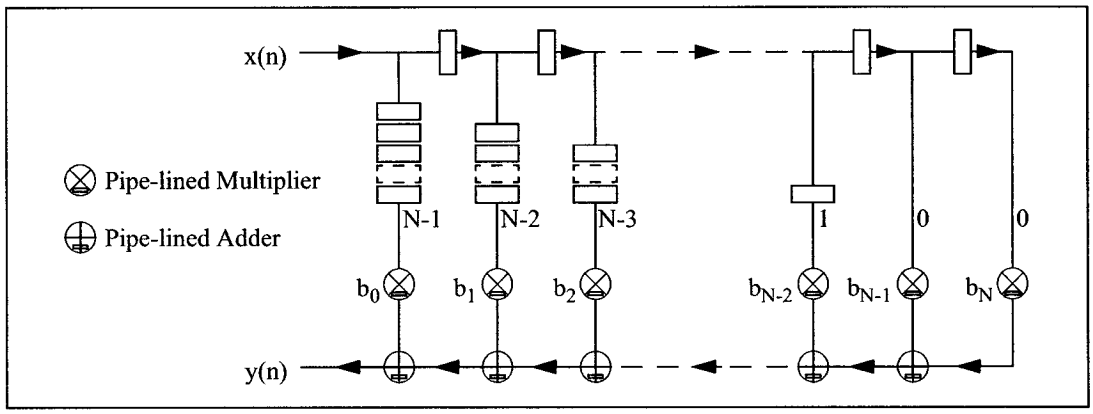

Figure 4.8 Approach 1 re-timed FIR Direct Form with actual pipe-lined components

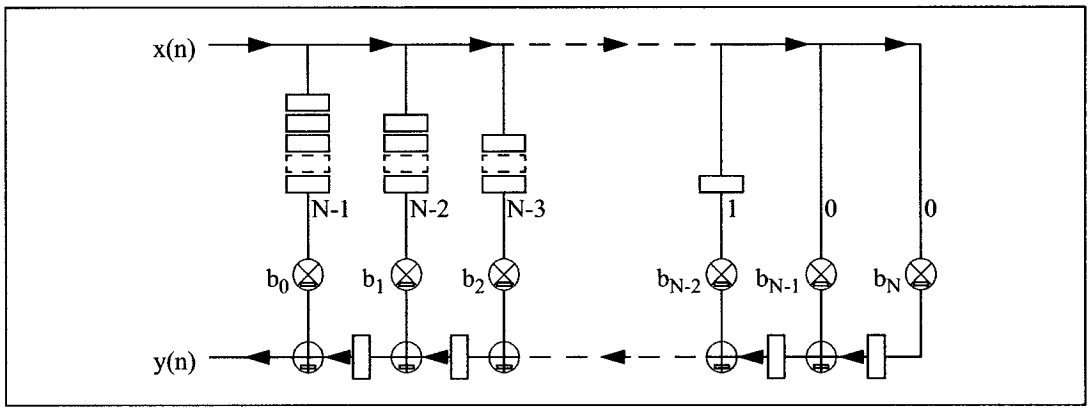

Figure 4.9 Approach 1 re-timed FIR Transposed Form with actual pipe-lined components

However, this approach has obvious drawbacks. The amount of delay elements to be added is proportional to the order of the filter. The total number of delay elements to be added is

the same in both Direct and Transposed form structures. It is equal to the arithmetic series of Eq (4.1), with N being the order of the filter:

$$\sum_{j=0}^{N-1} j = \frac{N(N-1)}{2} \qquad \textbf{Eq (4.1)}$$

This approach yields 820 delay elements required to properly time the structure of an order 40 FIR filter. That number would translate into 820*32 (single precision floating point numbers) = 26240 extra flip-flops in the design. Considering that the extra registers are required across the data path and are to be enabled at all time, regardless of the values at their input, we can anticipate that they will be adding to the amount of power consumed by the design rather than help reduce it.

4.3.2.2 Second Approach

To get rid of the extra delay elements and to solve the problem of correctly timing the structures, we take a closer look at the arithmetic units we are using. As seen earlier, these components, or units, are pipe-lined, which means that they already have embedded delay elements. The objective of this approach is to take advantage of that feature to re-time the structures and to restore the FIR filter function. Taking advantage of the presence of implicit delay units, or elements, in the arithmetic units results in the removal of all but a few absolutely necessary explicit delay units from the filter structures. Both Direct and Transposed forms would have only one explicit delay unit, which is placed across either the input data path (Direct form) or the computed data path (Transposed form). Figure 4.10,

Figure 4.11 and Figure 4.12 show the modified structures using simulated pipe-lined components.

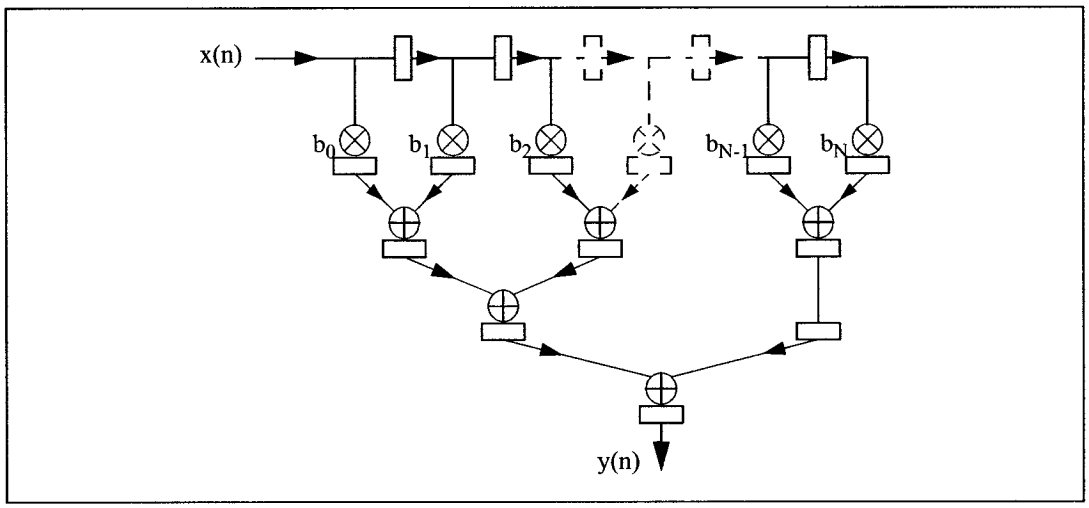

Figure 4.10 Modified Direct Tree Form with simulated pipe-lined components

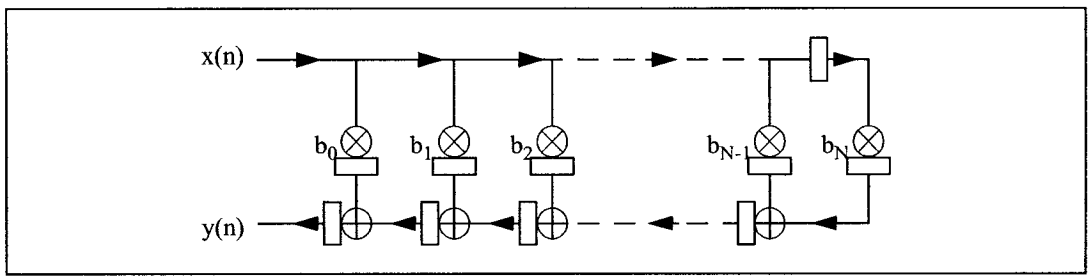

Figure 4.11 Modified Direct Form with simulated pipe-lined components

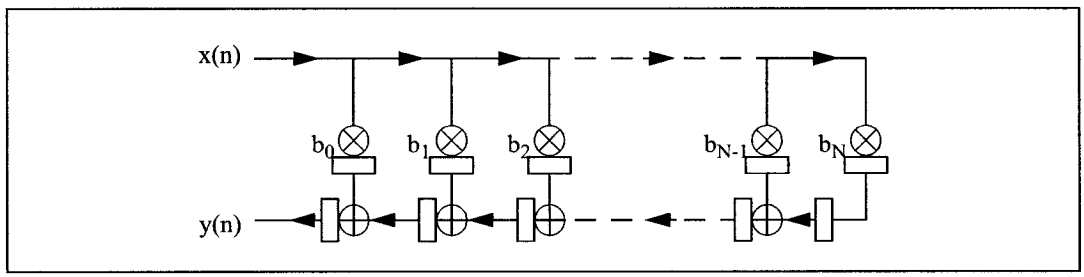

Figure 4.12 Modified Transposed Form with simulated pipe-lined components

Transforming the structures in the manner shown in Figure 4.13, Figure 4.14 and Figure 4.15 results in increased latency especially in the case of the Direct Tree form. The latency of the Direct Tree realisation is given by Eq (4.2), where N is the order of the FIR filter:

$$latency = 2 + \frac{\log N}{\log 2} = 2 + \log_2 N \qquad \textbf{Eq (4.2)}$$

Nevertheless, such transformation reduces dramatically the critical path delay of the Direct form structures. Instead of traversing a chain of adders that is proportional to the order of the filter as it is the case for the standard Direct network structures, the critical path delay is reduced to be that of the slowest component of the network structure. As a result, the traditional speed advantage of the Transposed form over the Direct forms disappears and all three structures have the same critical path delay. Since Direct and Transposed forms have the same area for the same filter order, other factors such as power dissipation become decisive in determining which realisation the designer would implement. The arithmetic units shown in Figure 3.1 and Figure 3.2 have output buffers, which simulates pipe-lining. With pipe-lined components the structures in Figure 4.10, Figure 4.11 and Figure 4.12 are transformed to look like the structures in Figure 4.13, Figure 4.14 and Figure 4.15 respectively. Throughout the rest of this thesis, only this representation is used.

It is obvious and straightforward that this approach is better than the first, area and power wise. Power consumption figures of both approaches were compared for an order eight FIR filter having a cut-off frequency of 0.2 and implemented in Direct form using two sets of

arithmetic units. The first set consisted of standard components (single data path) and the second consisted of the multiple data path components [8].

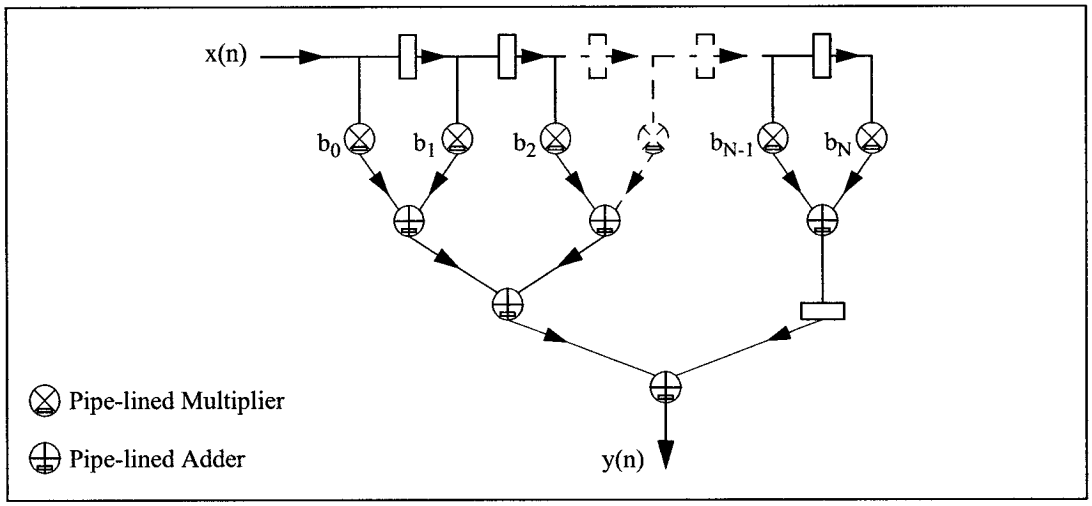

Figure 4.13 Direct Tree Form using pipe-lined components

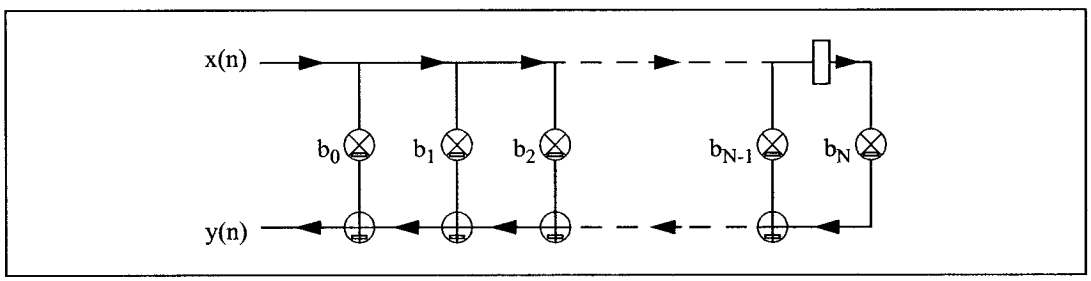

Figure 4.14 Direct Form using pipe-lined components

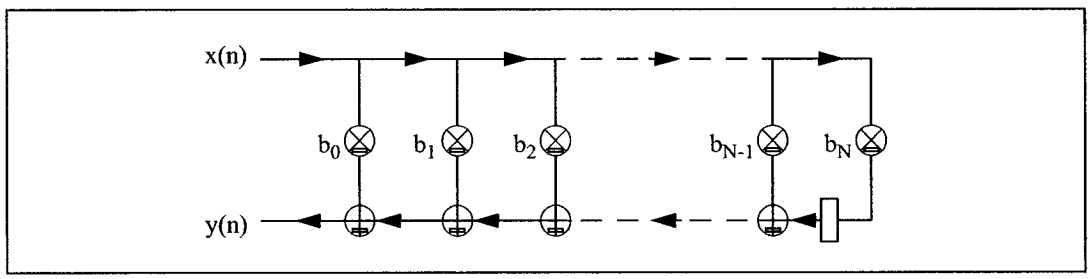

Figure 4.15 Transposed Form using pipe-lined components

The experiment was performed with an operating frequency of 16.67 MHz corresponding to a clock signal period of 60 ns. The circuit was implemented with 0.35 um technology. The first approach's structure had a total dynamic power consumption of 203.632 mW with the first set of components and 198.973 mW with the second. With the second approach the power consumption was only 181.155 mW and 177.276 mW respectively. We clearly see, from these results, that the second approach is power optimal for both sets of components. Naturally, the second approach's area is less than that of the first and, since we use the same pipe-lined arithmetic units in both approaches, the critical path delay remains the same. Moreover, the second approach is much easier to implement, coding wise, since there is a lot less hardware to worry about. The second approach will be adopted to implement all the other structures to be seen later in this thesis. Any difference in the critical path delay between the two approaches will only be due to the power optimization procedure and will neither be significant nor of great consequence to the present work.

4.3.3 Experimental Approach

The filter coefficients used are those of order 8, 16, 32 and 64 low pass, high pass, band pass and band stop filters. The LP and HP filters have a cut-off frequency of 0.2 while the BP and BS filters have a centre frequency of 0.2. Two audio signal samples are used, namely: Graphon.au, X-Files.wav mentioned earlier in Chapter 3. Here too, 64k floating-point format data samples are used. As for the arithmetic units in Chapter 3, the power performance figures are measured at an operating frequency of 16.67 MHz, corresponding to a clock signal period of 60 ns. The designs were synthesized with 0.18 um CMOS technology libraries from TSMC. The reported measurements are those of the area (A), the critical-path

delay (T) and the dynamic power (P). Also presented are the calculated values of the AT, AT^2 and PD products in Figure 4.16 and Figure 4.17. For all the experiments, frequency distributions of pre-alignment and normalization shifts, their rate of change and the relevant bit-level activities had been gathered also [22].

4.3.4 Results (pipe-lined)

4.3.4.1 Single Data Path Arithmetic Units

For all three network structures, regardless of the filter function or the filter order, and from the results presented in Table 4.3 through Table 4.10, we can make three observations. The first observation is totally expected and is that all three filter realisations have equal critical path delays of 36.90 ns. That is a Direct result of the transformation applied to the network structures that allowed the use of pipe-lined building blocks. As expected, the Transposed form no longer has any speed advantage over the Direct forms. This is not achieved by increasing the critical path delay of the Transposed form realisation, but by dramatically reducing that of the Direct forms. Further reducing the critical path delay can be achieved by Optimizing the algorithms and architectures of the arithmetic units that are the building blocks of the filter structures.

Table 4.3 A, T and P figures for order 8 and 16 low pass filters with SDP units

Low Pass	Order 8			Order 16		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct	1006680	36.90	19.18	1929247	36.90	38.90
Direct Tree	1030053	36.90	**14.93**	1973648	36.90	**27.75**
Transposed	1006680	36.90	19.97	1929247	36.90	39.70

Table 4.4 A, T and P figures for order 32 and 64 low pass filters with SDP units

Low Pass	Order 32			Order 64		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct	3774381	36.90	74.56	7464648	36.90	155.14
Direct Tree	3858686	36.90	**52.33**	7626233	36.90	**108.66**
Transposed	3774381	36.90	75.10	7464648	36.90	155.81

Table 4.5 A, T and P figures for order 8 and 16 high pass filters with SDP units

High Pass	Order 8			Order 16		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct	1006680	36.90	19.54	1929247	36.90	39.12
Direct Tree	1030053	36.90	**15.28**	1973648	36.90	**27.97**
Transposed	1006680	36.90	20.68	1929247	36.90	40.24

Table 4.6 A, T and P figures for order 32 and 64 high pass filters with SDP units

High Pass	Order 32			Order 64		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct	3774381	36.90	76.18	7464648	36.90	154.97
Direct Tree	3858686	36.90	**53.42**	7626232	36.90	**111.46**
Transposed	3774381	36.90	77.20	7454548	36.90	155.31

Table 4.7 A, T and P figures for order 8 and 16 band pass filters with SDP units

Band Pass	Order 8			Order 16		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct	1006680	36.90	19.56	1929247	36.90	37.97
Direct Tree	1030053	36.90	**15.14**	1973648	36.90	**27.46**
Transposed	1006680	36.90	20.48	1929247	36.90	38.92

Table 4.8 A, T and P figures for order 32 and 64 band pass filters with SDP units

Band Pass	Order 32			Order 64		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct	3774381	36.90	73.65	7464648	36.90	154.72
Direct Tree	3858586	36.90	**52.66**	7626233	36.90	**109.18**
Transposed	3774381	36.90	74.19	7464648	36.90	155.23

Table 4.9 A, T and P figures for order 8 and 16 band stop filters with SDP units

Band Stop	Order 8			Order 16		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct	1006680	36.90	19.18	1929247	36.90	38.04
Direct Tree	1030053	36.90	**14.50**	1973741	36.90	**27.97**
Transposed	1006680	36.90	19.90	1929247	36.90	38.76

Table 4.10 A, T and P figures for order 32 and 64 band stop filters with SDP units

Band Stop	Order 32			Order 64		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct	3774381	36.90	75.07	7464648	36.90	154.64
Direct Tree	3858686	36.90	**52.73**	7626233	36.90	**108.97**
Transposed	3774381	36.90	75.26	7464648	36.90	155.84

The second observation is also expected and is that Direct form and Transposed form realisations have equal areas while the Direct Tree form has an area that is slightly larger. Direct and Transposed form structures, in our case, are almost identical. In fact the only difference between the two realisations is the location of only one register that is placed either across the data path (Direct form) or across the computing path (Transposed form). Although the Direct Tree form realisation has the largest area of the three, the difference is not due to extra arithmetic units but rather to extra registers across the data path as seen in Figure 4.13. That difference does not exceed 2.2% for an order 64 filter. The Area Delay and Area $Delay^2$ products presented in Figure 4.16 clearly illustrate the first two observations, where the performance of all three network structures is within less than 3% of each other. From the above we see that the area and delay factors are not enough to decide which realisation has more advantage over the others. This means that the third factor, power dissipation, which is the subject of the third observation is the only decisive factor when choosing which realisation to implement.

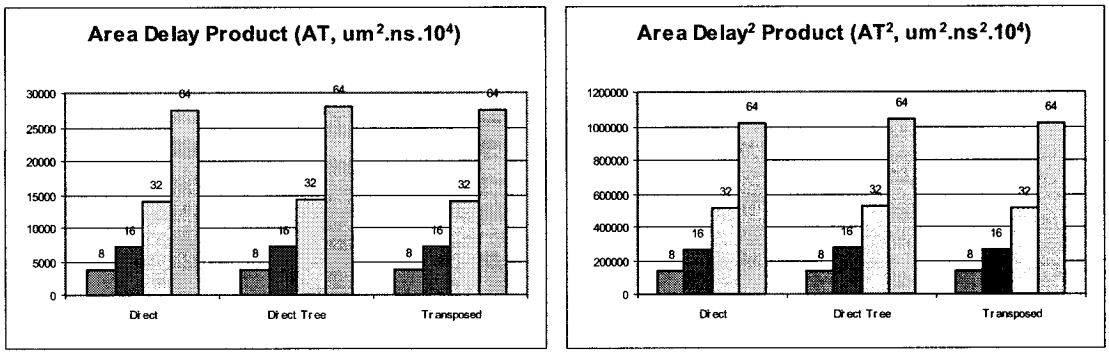

Figure 4.16 Area delay products for the FIR filter realisations using SDP units (pipe-lined)

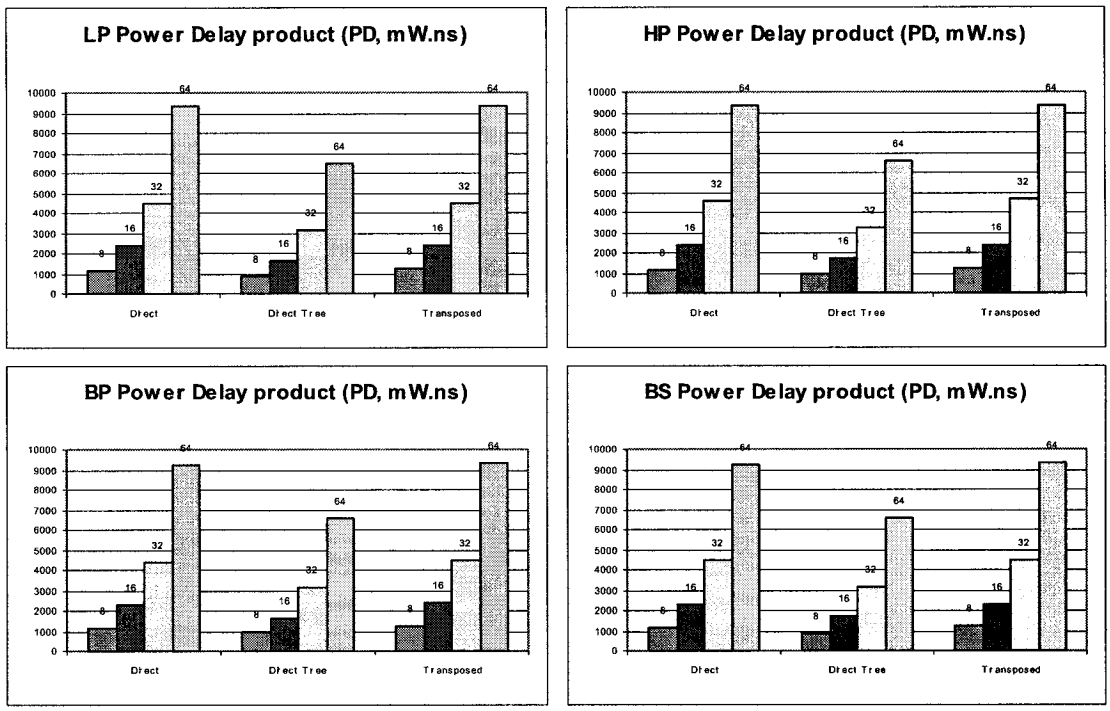

Figure 4.17 Power delay product for the FIR filter realisations using SDP units (pipe-lined)

As a third observation, we notice that Direct form realisation has a consistent power advantage over the Transposed form. This power advantage is inversely proportional to the filter order. It decreases as the filter order increases and is at least 4.6%, 2.6%, 1.3% and 1% for

order 8, 16, 32 and 64 respectively. As FIR filters typically require high number of stages to obtain sharp filter bands [27], the Direct form power advantage diminishes at high filter orders. The structural difference between the two structures can only be found in the Nth tap of the filter, N being the order of the filter. This structural difference translates into the Nth multiplication of the Transposed form realisation being executed one clock cycle behind that of the Direct form. As the filter order grows, the effect of the structural difference on the power performance diminishes in importance compared to the overall power consumption.

Table 4.11 Order 64 low pass filters activity

Low Pass filtering	Graphon.au			X-files.wav		
	Direct	Direct Tree	Transposed	Direct	Direct Tree	Transposed
Simulation time (ns)	3844230	3844710	3844230	3844230	3844710	3844230
Total Adders shift	13247289	7365443	13247289	15573641	8452817	15573641
Total togg.distance	4136266	1317809	4136266	4862915	1577354	4862915
Multipliers. shift	1549289	1548593	1549290	1834820	1834106	1834821
Total shift	14796578	8914036	14796579	17408461	10286923	17408462

Table 4.12 Order 64 high pass filters activity

High Pass filtering	Graphon.au			X-files.wav		
	Direct	Direct Tree	Transposed	Direct	Direct Tree	Transposed
Simulation time (ns)	3844230	3844710	3844230	3844230	3844710	3844230
Total Adders shift	10691138	7374228	10691138	12795909	8436036	12795909
Total togg.distance	4407486	1381650	4407486	5295885	1650020	5295885
Multipliers. shift	1487069	1486400	1487070	1763578	1762879	1763579
Total shift	12178207	8860628	12178208	14559487	10198915	14559488

Table 4.13 Order 64 band pass filters activity

Band Pass filtering	Graphon.au			X-files.wav		
	Direct	Direct Tree	Transposed	Direct	Direct Tree	Transposed
Simulation time (ns)	3844230	3844710	3844230	3844230	3844710	3844230
Total Adders shift	11113711	8925056	11113711	13797210	10349519	13797210
Total togg.distance	5097448	1547518	5097448	6354343	1835899	6354343
Multipliers. shift	1435901	1435253	1435901	1712111	1711455	1712111
Total shift	12549612	10360309	12549612	15509321	12060974	15509321

Table 4.14 Order 64 band stop filters activity

Band Stop filtering	Graphon.au			X-files.wav		
	Direct	Direct Tree	Transposed	Direct	Direct Tree	Transposed
Simulation time (ns)	3844230	3844710	3844230	3844230	3844710	3844230
Total Adders shift	18831955	10294393	18831955	21948410	11833249	21948410
Total togg.distance	5552540	1592016	5552540	6531904	1874914	6531904
Multipliers. shift	1662511	1661732	1662512	1964937	1964192	1964938
Total shift	20494466	11956125	20494467	23913347	13797441	23913348

Direct Tree form realisation, however, presents a significant power advantage over the Direct form and Transposed form realisations with all four filter functions. The adders in the tree structure have the magnitude of significand alignment shifts as well as the rate of change significantly smaller than in the Direct form and the Transposed form realisations. As a result, the Direct Tree form realisation requires less power than the Direct form and Transposed form realisations to perform these shifts [3]. From Table 4.11 through Table 4.14 we see that, in the tree structure, the range of total adders shifts is between 54% to 80%

that of the Direct form and Transposed form realisations. In the same token, the toggling distance, which is defined as the difference between two consecutive shifts, does not exceed one third. This is reflected in the power dissipation figures where Direct Tree form realisation consumes less power than Direct form realisation by at least 27%, 33%, 34% and 34% for order 8, 16, 32 and 64 respectively. The power advantage of the Direct Tree form realisation is clearly displayed in Figure 4.17. From the results gathered, we can conclude that Direct Tree form realisation is power optimal. However, the major disadvantage of the Direct Tree realisation is its latency, which increases with the filter order, as seen in Eq (4.2), while the latency of the other two realisations is constant at two.

4.3.4.2 Pipe-lined vs. Combinational

The results of the experiments are very important in that they clearly show the advantages of transforming the structures in the manner explained in Section 4.3.2.2. The area delay (AT) and area delay^2 (AT^2) products are essential for the comparison of the filter structures using pipe-lined and combinational arithmetic units. Figure 4.18 and Figure 4.19 show the advantage of using pipe-lined units in all three filter realisations.

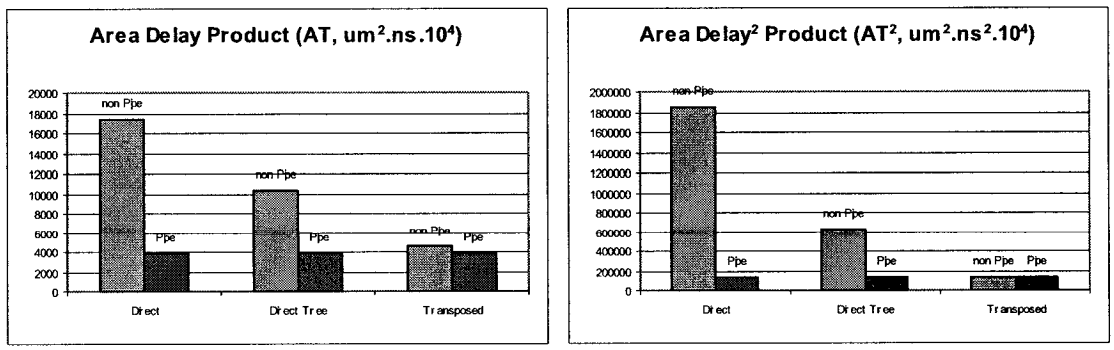

Figure 4.18 Area delay products for the FIR filter realisations using SDP units

Transforming the structures as discussed in Section 4.3 results in around 80% reduction in AT and around 92% reduction in AT^2 for Direct from realisation. For the Direct Tree form realisation the reduction in AT and AT^2 products is around 63% and 78% respectively. Transposed form realisation does not benefit in area and delay as much, from the transformation, as the Direct form and Direct Tree form realisations. In fact, AT is reduced by around 20% but AT^2 sees an increase of around 5%. The above is clearly illustrated in Figure 4.18. The merits of the structure transformation can be further enhanced by customising the synthesis constraints for each filter structure.

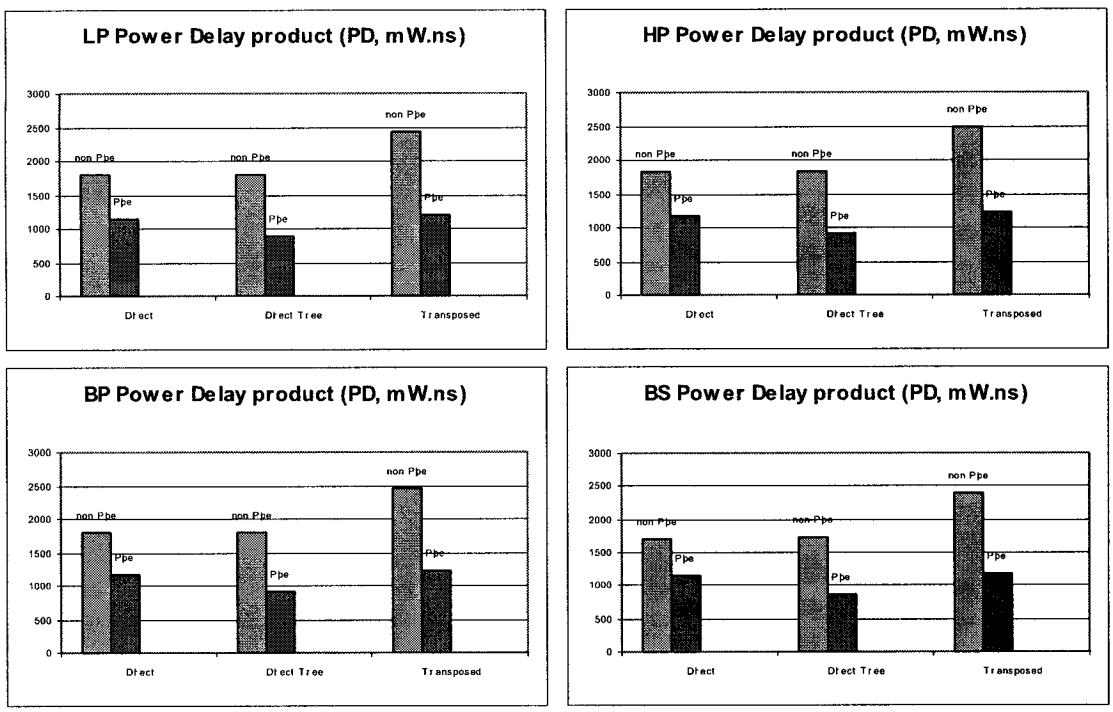

Figure 4.19 Power delay product for the FIR filter realisations using SDP units

Nonetheless, Transposed form realisation's power performance does benefit significantly from the transformation. The pipe-lined implementation of the Transposed form structure

results in around 50% reduction in power delay (PD) product as shown in Figure 4.19. For Direct from and Direct Tree form realisations the reduction is around 35% and 50% respectively. From the results illustrated in Figure 4.18 and Figure 4.19, one cannot dismiss the advantages of the transformation applied to the filter network structures.

4.3.4.3 Multiple Data Path Arithmetic Units

From the results presented in Table 4.15 through Table 4.22 we see once again that, regardless of the filter function or the filter order, all three network structures have equivalent critical path delays. In fact, Direct Tree form realisation has a slightly higher delay than both Direct form and Transposed form realisations. The difference, however, is less than 1% and is only due to the power optimization procedure [22]. As expected, the Transposed form is not showing any speed advantage over the Direct forms.

Table 4.15 A, T and P figures for order 8 and 16 low pass filters with MDP units

Low Pass	Order 8			Order 16		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct	1359239	26.26	14.74	2615688	26.26	28.28
Direct Tree	1382659	26.51	**14.47**	2660146	26.51	**27.49**
Transposed	1359237	26.26	14.84	2615615	26.26	28.42

Table 4.16 A, T and P figures for order 32 and 64 low pass filters with MDP units

Low Pass	Order 32			Order 64		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct	5128446	26.26	53.97	10153890	26.26	112.03
Direct Tree	5212824	26.51	**52.17**	10315466	26.51	**108.12**
Transposed	5128369	26.26	54.20	10153874	26.26	112.37

Table 4.17 A, T and P figures for order 8 and 16 high pass filters with MDP units

High Pass	Order 8			Order 16		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct	1359235	26.26	15.12	2615686	26.26	29.58
Direct Tree	1382653	26.51	**14.91**	2660154	26.51	**27.45**
Transposed	1359237	26.26	15.27	2615615	26.26	29.82

Table 4.18 A, T and P figures for order 32 and 64 high pass filters with MDP units

High Pass	Order 32			Order 64		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct	5128446	26.26	53.46	10153884	26.26	111.56
Direct Tree	5212824	26.51	**51.54**	10315448	26.51	**107.41**
Transposed	5128369	26.26	53.69	10153877	26.26	112.21

Table 4.19 A, T and P figures for order 8 and 16 band pass filters with MDP units

Band Pass	Order 8			Order 16		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct	1359235	26.26	14.99	2615688	26.26	27.91
Direct Tree	1382655	26.51	**14.84**	2660154	26.51	**27.02**
Transposed	1359241	26.26	15.09	2615615	26.26	28.07

Table 4.20 A, T and P figures for order 32 and 64 band pass filters with MDP units

Band Pass	Order 32			Order 64		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct	5128524	26.26	53.36	10153884	26.26	111.11
Direct Tree	5212820	26.51	**52.02**	10315469	26.51	**106.94**
Transposed	5128357	26.26	53.44	10153881	26.26	111.50

Table 4.21 A, T and P figures for order 8 and 16 band stop filters with MDP units

Band Stop	Order 8			Order 16		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct	1359235	26.26	14.89	2615686	26.26	27.46
Direct Tree	1382649	26.51	**14.60**	2660197	26.51	**26.72**
Transposed	1359235	26.26	15.02	2615625	26.26	27.48

Table 4.22 A, T and P figures for order 32 and 64 band stop filters with MDP units

Band Stop	Order 32			Order 64		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct	5128541	26.26	54.44	10153894	26.26	111.60
Direct Tree	5212739	26.51	**52.91**	10315470	26.51	**107.67**
Transposed	5128346	26.26	54.56	10153895	26.26	111.93

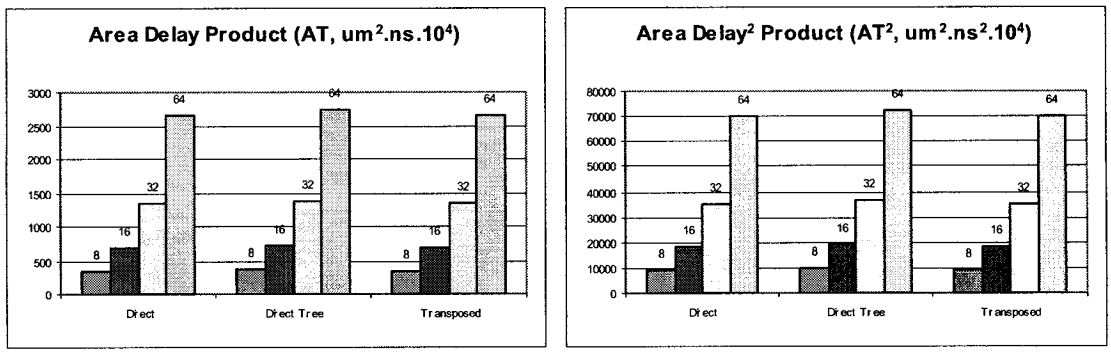

Figure 4.20 Area delay products for the FIR filter realisations with MDP units (pipe-lined)

As with the SDP units, Direct form and Transposed form realisations have equal areas while the Direct Tree form have an area that is slightly larger. Direct and Transposed form structures, in our case, are almost identical as is clearly illustrated in Figure 4.14 and Figure 4.15. The Direct Tree form realisation has the largest area of the three, which is due to extra registers across the data path as seen in Figure 4.13. That difference still does not exceed 2% for an order 64 filter. The Area Delay and Area Delay^2 products presented in Figure 4.20 clearly illustrate the first two observations, where the performance of all three network structures is within 3% of each other. From the above we see that the area and delay factors

are not enough to decide which realisation has more advantage over the others. This means that power dissipation is the only decisive factor when choosing a realisation to implement.

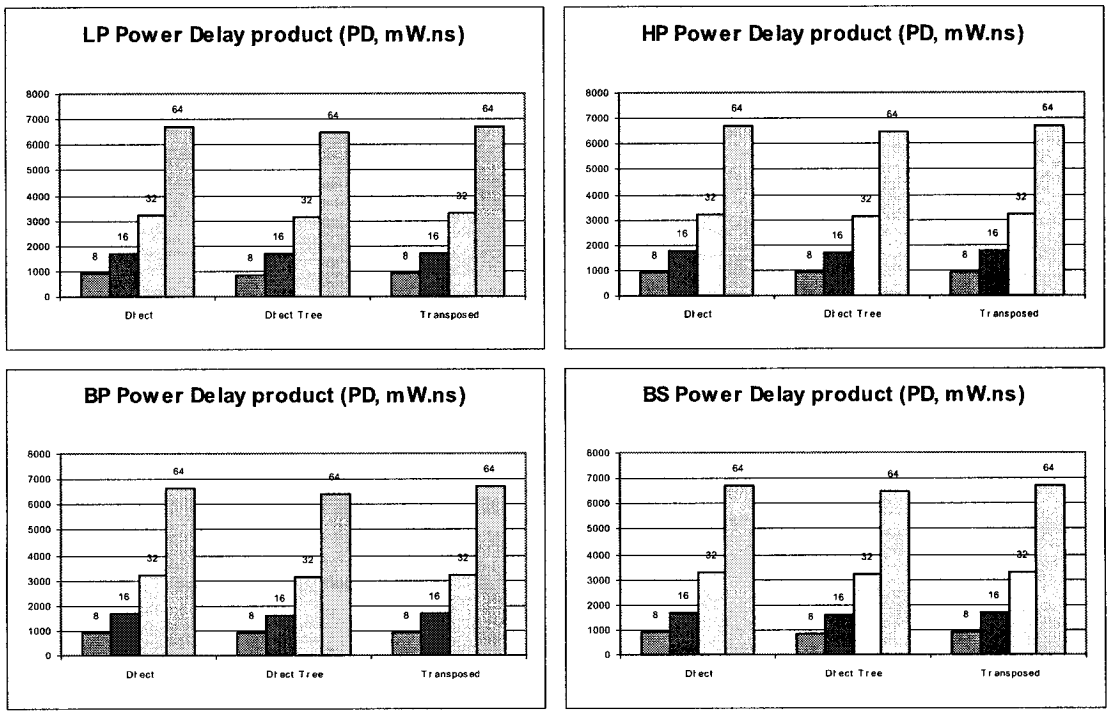

Figure 4.21 Power delay product for the FIR filter realisations with MDP units (pipe-lined)

As seen with the SDP units in Sub-section 4.3.4.1, Direct form realisation has a power advantage, albeit minute but consistent, over the Transposed form realisation. This power advantage is inversely proportional to the filter order. It decreases as the filter order increases and is at least 0.7%, 0.5%, 0.4% and 0.3% for order 8, 16, 32 and 64 respectively. Direct Tree form realisation, however, presents a slightly better power advantage over the Direct form and Transposed form realisations. Here too, the adders in the tree structure have the magnitude of significand alignment shifts as well as the rate of change significantly smaller than in the Direct form and the Transposed form realisations. As a result, the Direct Tree

form realisation requires less power than the Direct form and Transposed form realisations to perform these shifts [3]. This is reflected in the power dissipation figures where Direct Tree form realisation consumes less power than Direct form realisation by at least 1.8%, 2.8%, 3.3% and 3.5% for order 8, 16, 32 and 64 respectively. The power advantage of the Direct Tree form realisation is illustrated in Figure 4.21. From the results gathered, we can conclude that Direct Tree form realisation is power optimal. Again, however, Direct Tree form realisation suffers from its latency, which increases with the filter order, as seen in Eq (4.2), while the latency of the other two realisations is constant at two.

4.3.4.4 Multiple Data Path vs. Single Data Path

Comparing the results of Table 4.3 through Table 4.10 and Table 4.15 through Table 4.22, we can clearly see the power advantage of the MDP units over their SDP counterparts. This advantage is very noticeable in the case of Direct form and Transposed form realisations. These two realisations implemented with MDP units dissipate less power than with SDP units. The power saving realised with MDP units is at least 23% for Direct form and 25% for Transposed form. For Direct Tree from realisation the power advantage of the MDP units is far less important. In fact, the percentage reduction in power dissipation does not exceed 3% for Direct Tree form realisation. Such a small difference makes it less easy to decide which FP arithmetic units to use in the implementation of the Direct Tree form structure. Contrary to what we have seen earlier, it is factors other than power that help make that decision. A summary of the above is illustrated in Figure 4.22, which presents the Power Delay (PD) product of the structures implemented with MDP and SDP arithmetic units.

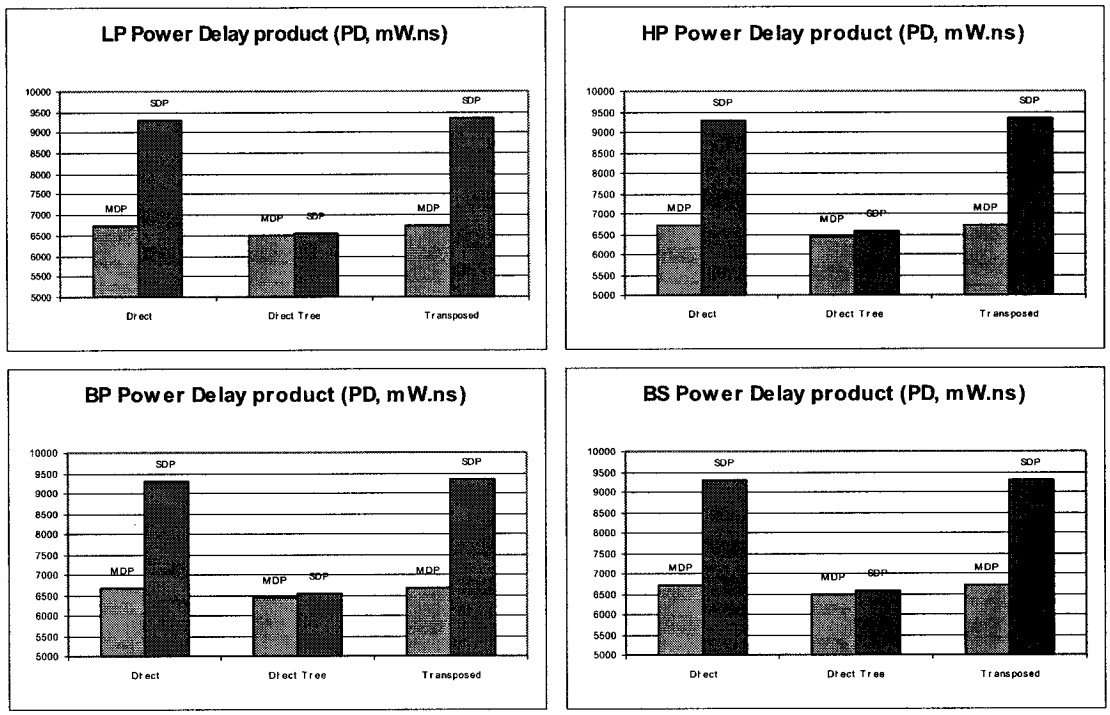

Figure 4.22 Power delay product for order 64 FIR filter realisations (pipe-lined)

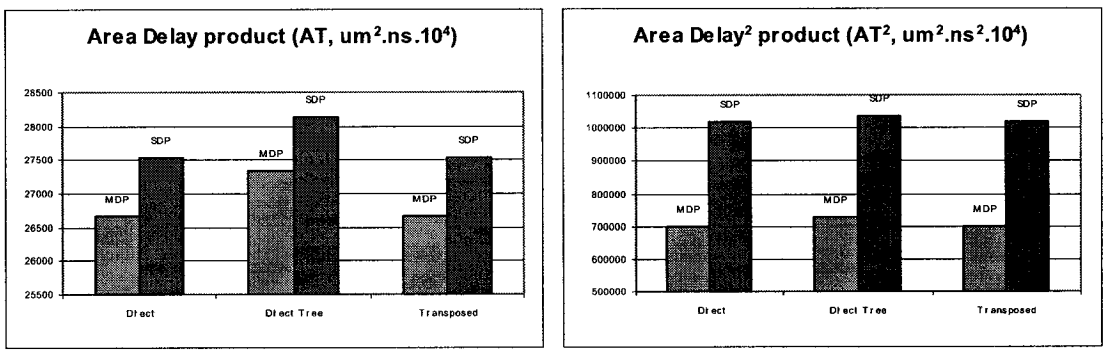

Figure 4.23 Area delay products for order 64 FIR filter realisations (pipe-lined)

The area delay (AT) and area delay^2 (AT^2) products are essential for the comparison of the filter structures using MDP and SDP arithmetic units. Figure 4.23 show the advantage of using MDP units in all three filter realisations. Although the structures are around 35%

larger when implemented with MDP units than with SDP units, their critical path delays are around 30% lower. The Area Delay (AT) and Area Delay^2 (AT^2) products reveal a considerable advantage in favour of the structures implemented with MDP units. Using MDP units reduces AT by around 4%, and AT^2 by around 30%.

4.4 Floating-Point vs. Fixed-Point

With the FIR filters we also see that under the same operating conditions, fixed-point and floating-point units have widely different performances. The same filter structure, order 8 pipe-lined Direct form, the area with typical fixed-point units is between 40% to 50% less than with floating-point units. The fixed point's critical path delay is about 65% smaller than the floating-point's as seen in Figure 4.25.

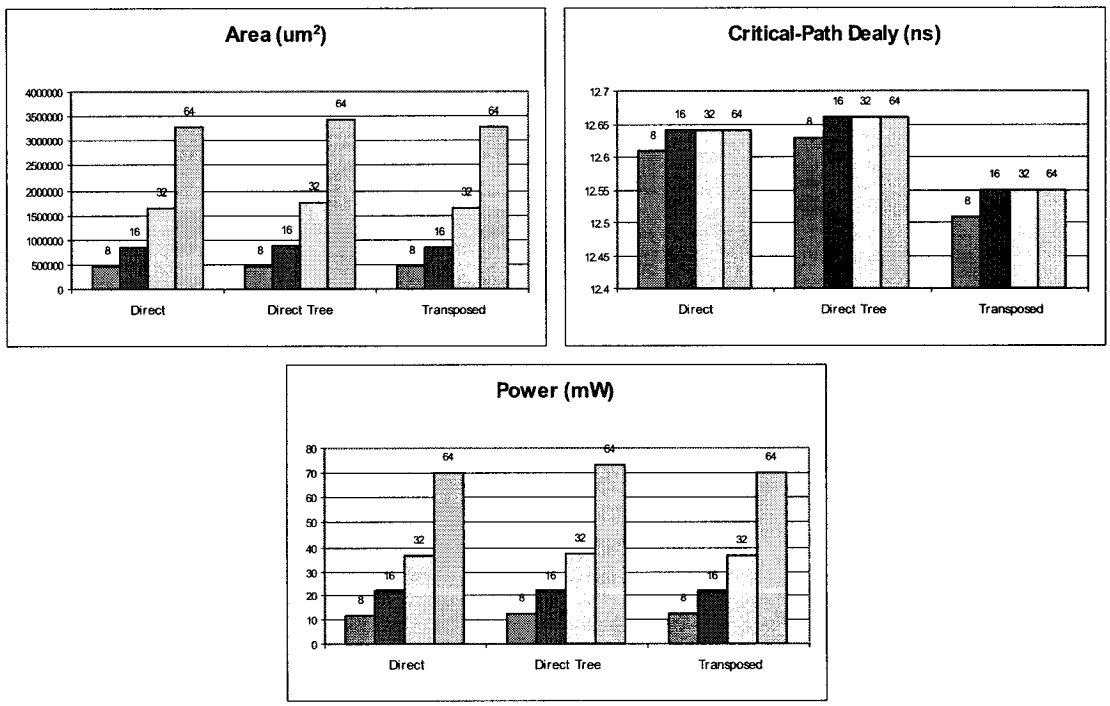

Figure 4.24 Area, Delay and Power Performance of the Fixed-Point FIR filters

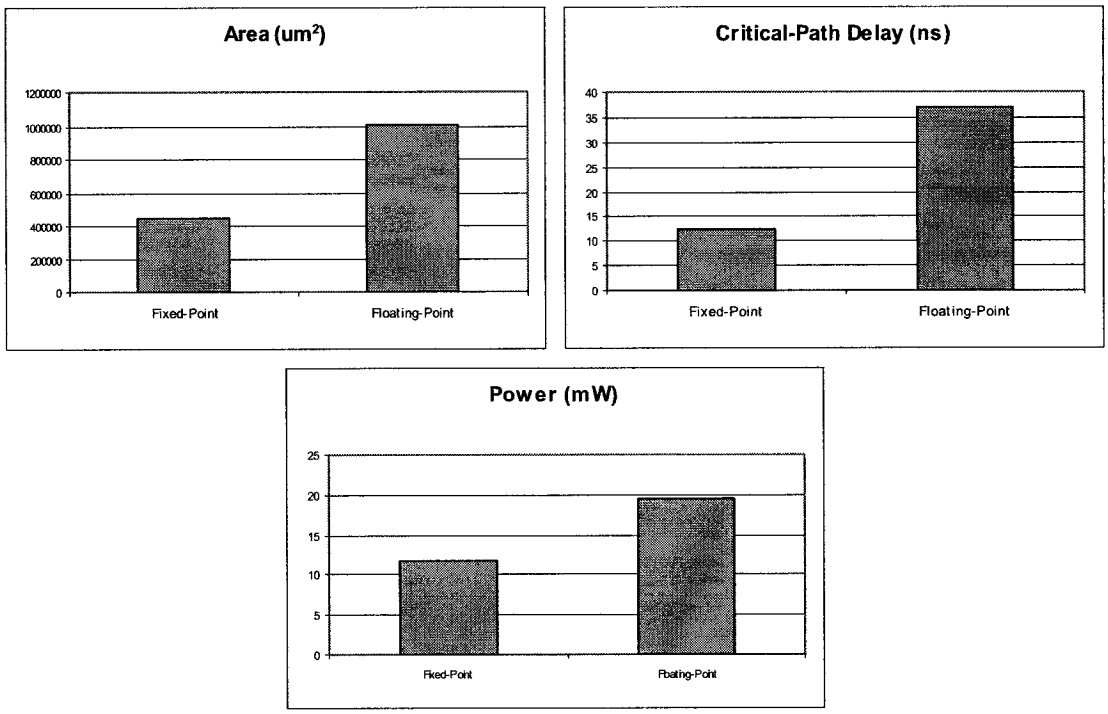

Figure 4.25 Comparison of Performance of the Fixed and Floating-Point FIR filters

With fixed-point units the total power dissipation is around 38% smaller than with floating-point units. The contribution of adders to the total power dissipation is around 10% with fixed-point units and 23% with floating-point units. The multipliers contribution is 76% and 70% respectively. The rest of the power is dissipated by the network interconnections. The results also show that with fixed-point arithmetic units Direct Tree form is not power optimal. Direct Tree from is actually the least desirable of the three structures. Transposed form is showing the smallest critical-path delay followed very closely by Direct form and Direct Tree form. Direct Form, however, has the smallest area of the three structures and

dissipates the least amount of power. The difference in performance between the three structures is negligible and it is less than one percentage point as can be seen in Figure 4.24.

4.5 Conclusion

From the experiments performed on the different FIR filter structures, we notice that in general the Direct form structure has a power advantage over the Transposed form. This is true for all four filtering functions. Area and delay differences between Direct and Transposed forms are negligible when they exist and are only due to CAD tools optimization. Direct Tree form presents an even higher power advantage over Direct and Transposed forms with a minor area and delay penalty (3% with SDP units and 1% with MDP units) for an order 64 filter. Another conclusion we can draw from the set of experiments is that the Multiple Data Path units have a definite advantage over their single data path counterparts. We see, from the results we have gathered, that a FIR filter structure using MDP units consumes less power than the same filter structure using SDP units, and requires less time to perform the filtering function. Structures using MDP units suffer, however, from increased area, which is due to the fact that, as we saw in Chapter 3, the MDP arithmetic units are larger than their SDP counterparts.

To recap the results of the comparison between MDP and SDP units, the filter structures have less power dissipation with MDP units than with SDP units. The percentage power reduction is 23% for Direct form and 25% for Transposed form. For Direct Tree form it is 3% at most. The AT and AT^2 products also benefit from the use of the MDP units as they

are reduced by 4% and 30% respectively. Direct Tree form realisation is power optimal with MDP and SDP units.

4.6 Summary

In this chapter, we measured the area, delay and power performance of three FIR filter structures with different filtering functions, different filter orders and different data samples. We began by presenting the different filter structures to be evaluated and how they were implemented. We, also, listed the data samples used to avoid any ambiguity. The results obtained were compiled and presented in the form of tables and graphs for each of the filtering functions and analysed. We finally drew two conclusions from the results we gathered. First, Direct Tree form structure is power optimal. Second, the MDP arithmetic units, once again, proved to have a significant power advantage over their SDP counterparts in FIR filtering applications. Finally, we saw that the filter structures using fixed-point arithmetic units do not have the same performance with floating-point arithmetic units.

Chapter 5

Infinite Impulse Response Filters

5.1 Introduction

In DSP applications, filtering of data samples is a very demanding operation and with the help of widely available CAD tools and with a given set of filter specifications one can rapidly delineate an appropriate filter. This is true for both FIR and IIR filters where we used the same tool to generate the different filter weights. An IIR filter transfer function can have different implementations without modifying the filter characteristics. In Chapter 2, we have seen that IIR filters can be implemented in Direct form I, Direct form II and Transposed Direct form II while the filter transfer function remains the same. The memory requirement for these different realisations is not the same. In general, differences in realisations imply differences in performance as well. In this chapter we will consider the different IIR filters realisations discussed in Chapter 2. We will show how the different IIR filter network structures presented in Chapter 2 were modified to accommodate the pipe lined nature of the floating-point arithmetic units. A comparative study of area, delay and power consumption performance of the different realisations is discussed in this chapter. The rest of this chapter is organized in the following manner. Section 5.2 presents the hardware implementation of the different IIR filters realisations and shows how they were derived from the structures shown in Chapter 2. Section 5.3 presents the experimental approach used to simulate and synthesize the different designs. Section 5.4 presents the results of the different experiments along with analyses of the results. Section 5.6 draws conclusions from the results while Section 5.7 presents a summary of this chapter.

5.2 Hardware Implementation of IIR Systems

The objective here is to implement the different IIR filter network structures discussed in Chapter 2 using pipe-lined components. In this section, three common IIR filter second-order-section (SOS) realisations are presented. In standard form structures the arithmetic units are purely combinational, i.e. no pipe-line stages. But in our case, the FP arithmetic units to be used as building blocks are pipe-lined. Each unit has one stage of registers, which means that each unit, or component, requires one clock cycle to perform one operation.

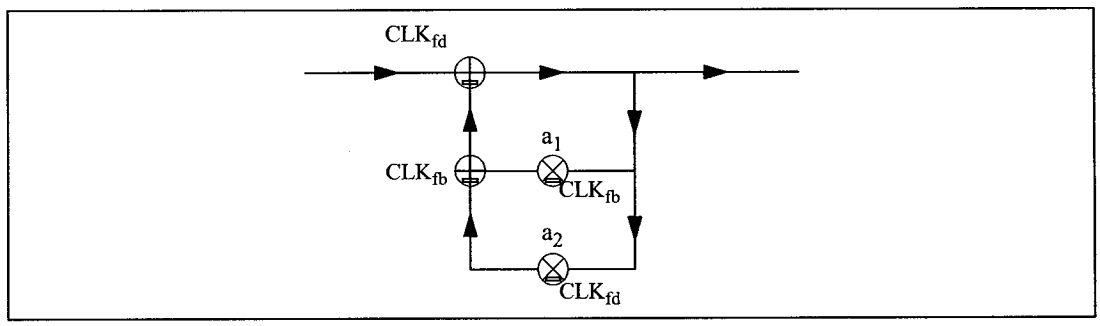

Figure 5.1 Feedback loop of Direct form I illustrating the clocking scheme of the structures

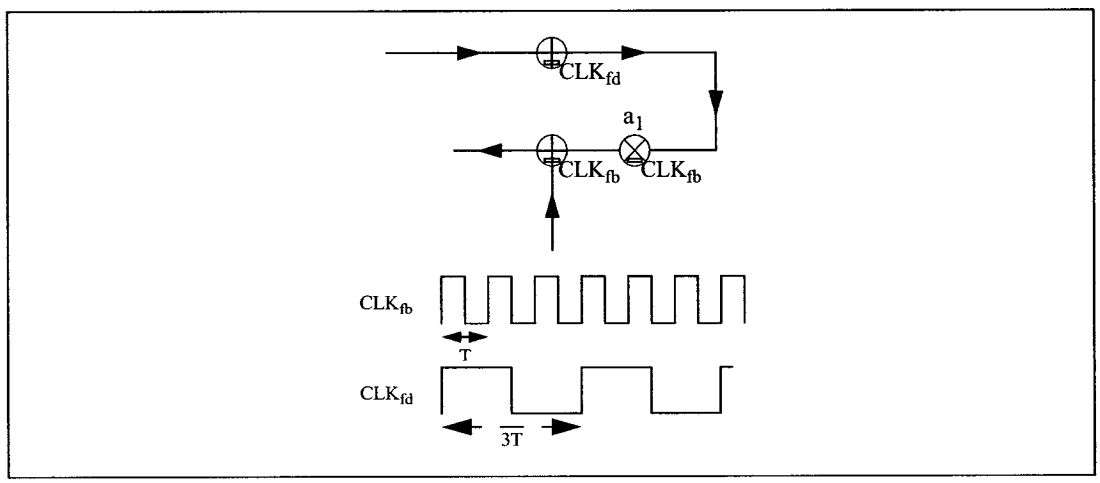

Figure 5.2 Broken feedback loop and waveforms of the clock signals

The implementation of the different IIR filter structures with pipe-lined arithmetic units uses the same approach used to implement the FIR filter structures. However, the challenge with the IIR structures is to solve the problem with the feedback path. Following the same approach used to implement the FIR structures, i.e. using pipe-lined building blocks, we would need to clock the IIR structures with two different clock signals. The first clock signal is used to clock the filter in general. This includes the forward loop and some components of the feedback loop. The second clocks the rest of the components in the feedback loop and is three times faster than the first clock signal. The idea is to have the result of the feedback loop synchronised with the data at its input.

Examining the first loop in Figure 5.1 one can easily see why CLK_{fb} has to be at least three times faster than CLK_{fd}. If we break the loop, as shown in Figure 5.2, and suppose the arithmetic units have the same delay T, the total time that a signal arriving at the input of the first adder would need to get to the output of the broken loop is 3T. Applying the same approach used to implement the FIR structures and the timing approach explained in the previous paragraph, we end up with the IIR filter structures illustrated in Figure 5.3, Figure 5.4 and Figure 5.5.

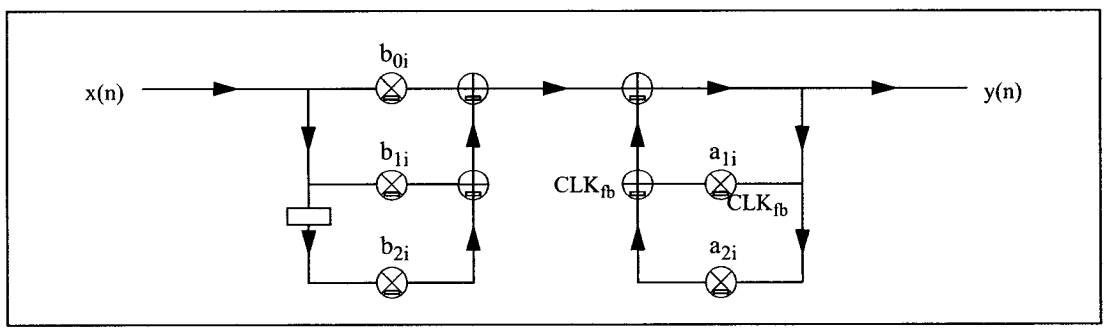

Figure 5.3 Ith Second-order section of a Direct form I IIR filter structure

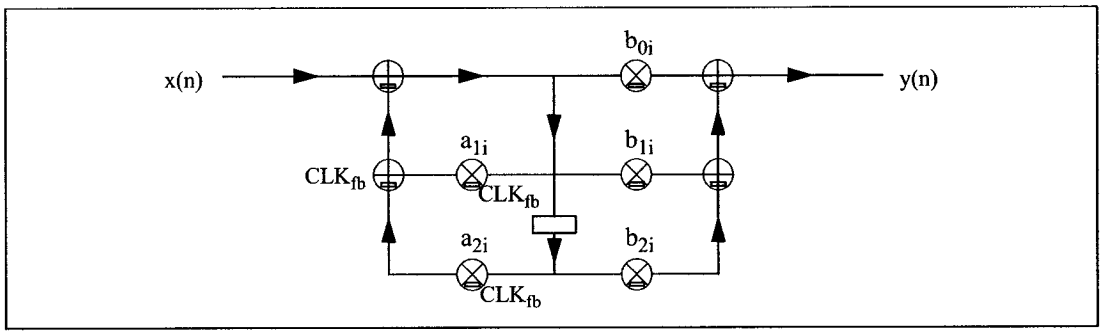

Figure 5.4 Ith Second-order section of a Direct form II IIR filter structure

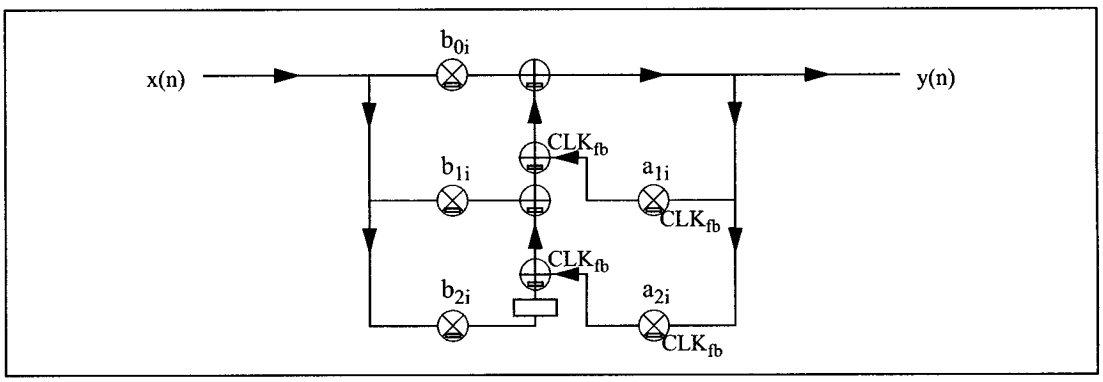

Figure 5.5 Ith Second-order section of a Transposed Direct form II IIR filter structure

5.3 Experimental Approach

The set of experiments encompasses low Pass, high pass, band pass and band stop filtering of real data samples using order 8 elliptical filters. The filters have a pass band ripple of 0.03, band stop ripple of -80db, -100db, -120db and -140db with cut-off frequency of 0.2 for LP and HP filters and centre frequency of 0.2 for BP and BS filters. Two audio signal samples are used, namely: Graphon.au, X-Files.wav. As for the experiments in Chapter 3 and Chapter 4, 64k data input vectors in floating-point format are used. The designs were synthesized with 0.18 um CMOS technology libraries from TSMC. The reported

measurements are those of the area (A), the critical-path delay (T) and the dynamic power (P). Also presented are the calculated values of the AT, AT^2 and PD products. The power performance figures are measured at an operating frequency of 5.56 MHz, corresponding to a clock signal period of 180 ns. For all the experiments, frequency distributions of pre-alignment and normalization shifts, their rate of change and the relevant bit-level activities had also been gathered [22].

5.4 Results

5.4.1 Single Data Path Arithmetic Units

5.4.1.1 Low Pass and High Pass Filters

For these filters and from simulation results in Table 5.1 through Table 5.4, we find that Direct form II has the lowest area delay product of the three realisations we are studying. The area delay (AT) product of Direct form II is 3.6% less than that of Direct form I and 6.8% less than that of Transposed Direct form II. The area delay^2 (AT^2) product of Direct form II follows a similar trend and is around 7% less than that of Direct form I and around 13% less than that of Transposed Direct form II [Figure 5.6].

Table 5.1 A, T and P results for low pass filters with stop band ripple -80db and -100db using SDP units

Low Pass	Stop band ripple -80db			Stop band ripple -100db		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct I	2238321	37.17	15.14	2238321	37.17	14.99
Direct II	2238321	35.83	**14.91**	2238321	35.83	**14.65**
Transposed DII	2238321	38.46	17.25	2238321	38.46	17.14

Table 5.2 A, T and P results for low pass filters with stop band ripple -120dband - 140db using SDP units

Low Pass	Stop band ripple -120db			Stop band ripple -140db		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct I	2238321	37.17	15.25	2238321	37.17	14.91
Direct II	2238321	35.83	**15.03**	2238321	35.83	**14.82**
Transposed DII	2238321	38.46	17.35	2238321	38.46	17.03

Table 5.3 A, T and P results for high pass filters with stop band ripple -80db and - 100db using SDP units

High Pass	Stop band ripple -80db			Stop band ripple -100db		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct I	2238321	37.17	16.20	2238321	37.17	16.41
Direct II	2238321	35.83	**15.79**	2238321	35.83	**15.85**
Transposed DII	2238321	38.46	18.87	2238321	38.46	19.03

Table 5.4 A, T and P results for high pass filters with stop band ripple -120db and - 140db using SDP units

High Pass	Stop band ripple -120db			Stop band ripple -140db		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct I	2238321	37.17	16.64	2238321	37.17	16.52
Direct II	2238321	35.83	**16.31**	2238321	35.83	**16.30**
Transposed DII	2238321	38.46	19.44	2238321	38.46	19.09

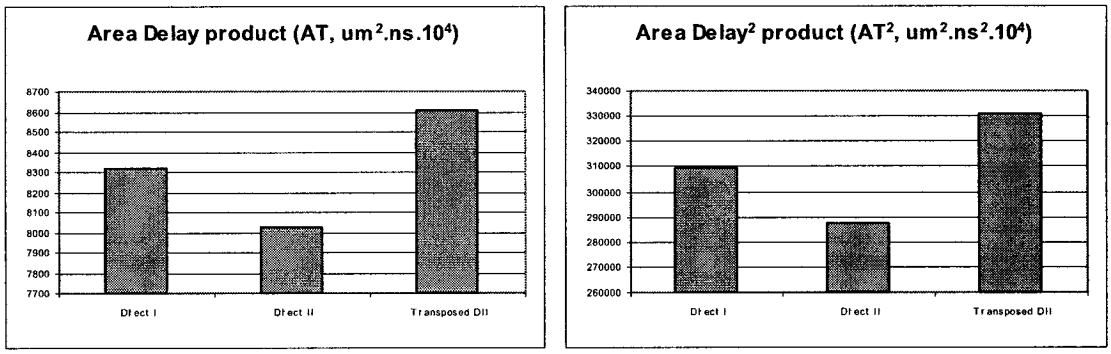

Figure 5.6 Area delay products for low pass and high pass filters using SDP units

The power behaviour of the network structures follows that of the AT and AT^2 products, where Direct form II realisation has the advantage over the other two realisations [Figure 5.7]. The power delay (PD) product of Direct form II is at least 1.5% less than that of Direct form I and 13% less than that of Transposed Direct form II.

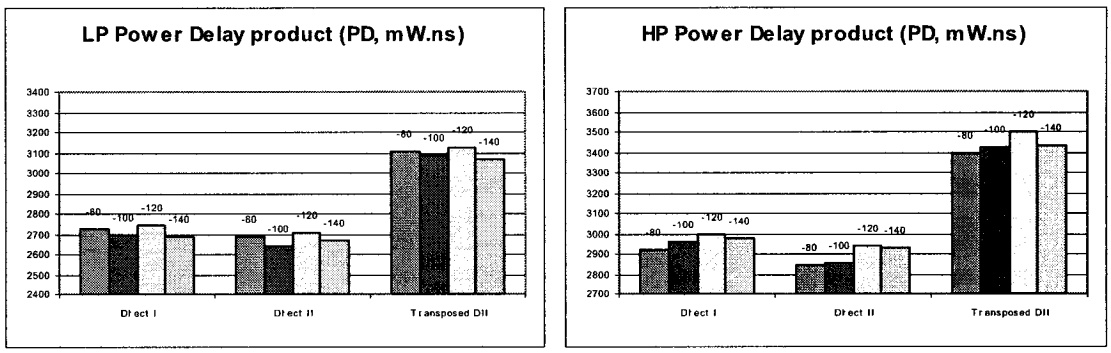

Figure 5.7 Power delay product for low pass and high pass filters using SDP units

5.4.1.2 Band Pass and Band Stop Filters

Experimental results for these filters are presented in Table 5.5 through Table 5.8. With this type of filters, the area increases by a factor of two since the number of weights is double

what it is for low pass and high pass filters. The critical path delays do not change however, as we have seen with FIR filters. With the doubling of the area, we find that Direct form II still has the lowest area delay (AT) product of the three realisations we are studying.

Table 5.5 A, T and P results for band pass filters with stop band ripple -80db and -100db using SDP units

Band Pass	Stop band ripple -80db			Stop band ripple -100db		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct I	4476643.5	37.17	29.66	4476643.5	37.17	30.50
Direct II	4476643.5	35.83	**29.62**	4476643.5	35.83	**30.33**
Transposed DII	4476643.5	38.46	34.11	4476643.5	38.46	34.97

Table 5.6 A, T and P results for band pass filters with stop band ripple -120db and -140db using SDP units

Band Pass	Stop band ripple -120db			Stop band ripple -140db		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct I	4476643.5	37.17	29.15	4476643.5	37.17	31.91
Direct II	4476643.5	35.83	**28.93**	4476643.5	35.83	**31.84**
Transposed DII	4476643.5	38.45	33.66	4476643.5	38.46	35.93

Table 5.7 A, T and P results for band stop filters with stop band ripple -80db and -100db using SDP units

Band Stop	Stop band ripple -80db			Stop band ripple -100db		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct I	4476643.5	37.17	30.45	4476643.5	37.17	30.3041

Table 5.7 A, T and P results for band stop filters with stop band ripple -80db and -100db using SDP units

Band Stop	Stop band ripple -80db			Stop band ripple -100db		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct II	4476643.5	35.83	**29.92**	4476643.5	35.83	**29.91**
Transposed DII	4476643.5	38.46	34.7816	4476643.5	38.46	34.79

Table 5.8 A, T and P results for band stop filters with stop band ripple -120db and -140db using SDP units

Band Stop	Stop band ripple -120db			Stop band ripple -140db		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct I	4476643.5	37.17	31.94	4476643.5	37.17	30.27
Direct II	4476643.5	35.83	**31.53**	4476643.5	35.83	**29.94**
Transposed DII	4476643.5	38.45	36.32	4476643.5	38.46	34.65

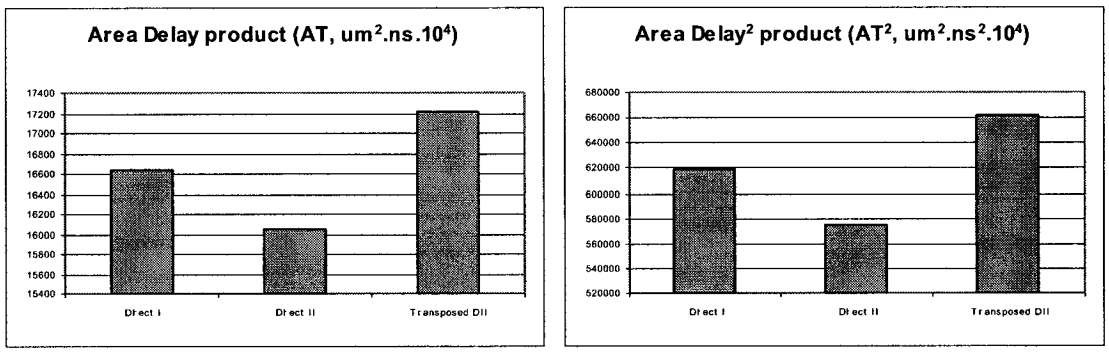

Figure 5.8 Area delay products for band pass and band stop filters using SDP units

For these filters too, the area delay (AT) product of Direct form II is 3.6% less than that of Direct form I and 6.8% less than that of Transposed Direct form II. The area delay^2 (AT^2)

product of Direct form II follows a similar trend and is around 7% less than that of Direct form I and around 13% less than that of Transposed Direct form II [Figure 5.8].

The power behaviour of these filters is not different from that of the low pass and high pass filters. For band pass and band stop filters, Direct form II remains the network structure with the lowest power delay (PD) product [Figure 5.9]. The difference between Direct form II and Direct form I realisations is very small though as the PD product of Direct form II is less than that of Direct form I by 1.7% at the most.

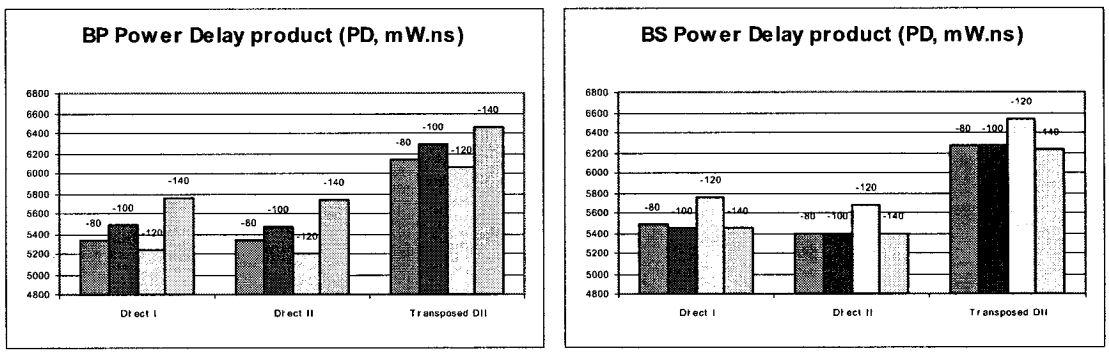

Figure 5.9 Power delay product for band pass and band stop filters using SDP units

In contrast, both realisations have a significant power advantage over the Transposed Direct form II. Power delay (PD) product for the latter is higher than that of Direct form I and Direct form II by at least 13%. From the above we can say that Transposed Direct form II realisation is the least desirable of the three realisations to implement with band pass and band stop filters. Direct form II would be the realisation of choice when considering power, area and delay performance.

5.4.2 Multiple Data Path Arithmetic Units

5.4.2.1 Low Pass and High Pass Filters

For these filters and from simulation results in Table 5.9 through Table 5.12, we find that Direct form II has the lowest area delay product of the three realisations we are studying. The area delay (AT) product of Direct form II is around 1% less than that of Direct form I and around 4% less than that of Transposed Direct form II.

Table 5.9 A, T and P results for low pass filters with stop band ripple -80db and -100db using MDP units

Low Pass	Stop band ripple -80db			Stop band ripple -100db		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct I	2923055	25.46	**16.56**	2923099	25.46	**16.35**
Direct II	2923100	25.26	18.17	2923055	25.26	17.98
Transposed DII	2923087	26.29	18.87	2923055	26.29	18.65

Table 5.10 A, T and P results for low pass filters with stop band ripple -120dband -140db using MDP units

Low Pass	Stop band ripple -120db			Stop band ripple -140db		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct I	2923120	25.46	**16.67**	2923087	25.46	**16.30**
Direct II	2923100	25.26	18.22	2923087	25.26	17.87
Transposed DII	2923055	26.29	18.99	2923055	26.29	18.60

Table 5.11 A, T and P results for high pass filters with stop band ripple -80db and -100db using MDP units

High Pass	Stop band ripple -80db			Stop band ripple -100db		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct I	2923087	25.46	**16.98**	2923087	25.46	**17.03**
Direct II	2923100	25.26	18.70	2923055	25.26	18.92
Transposed DII	2923087	26.29	19.22	2923055	26.29	19.35

Table 5.12 A, T and P results for high pass filters with stop band ripple -120db and -140db using MDP units

High Pass	Stop band ripple -120db			Stop band ripple -140db		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct I	2923120	25.46	**17.17**	2923087	25.46	**17.46**
Direct II	2923087	25.26	19.22	2923087	25.26	19.27
Transposed DII	2923055	26.29	19.37	2923055	26.29	19.70

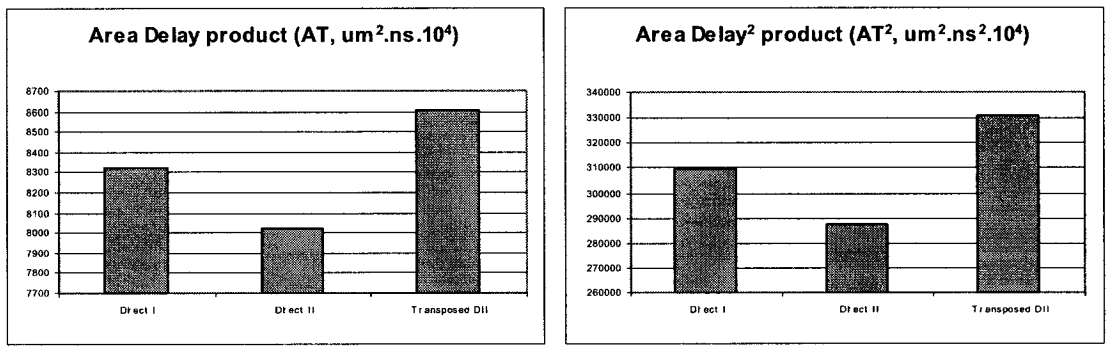

Figure 5.10 Area delay products for low pass and high pass filters using MDP units

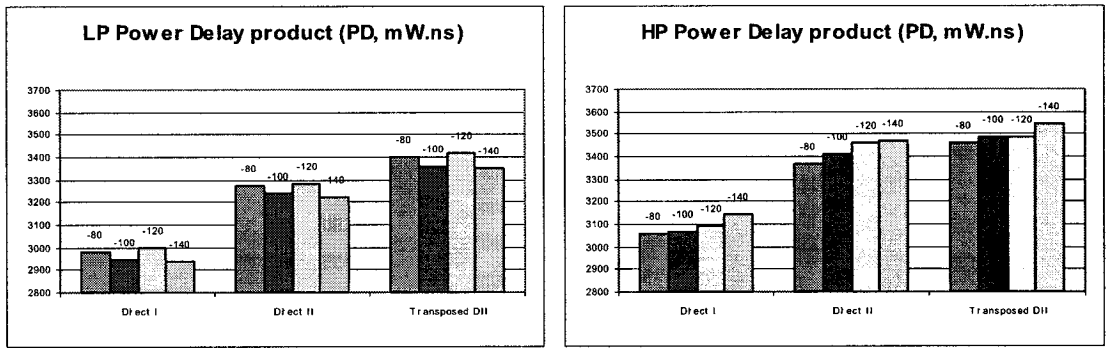

Figure 5.11 Power delay product for low pass and high pass filters using MDP units

The area delay^2 (AT^2) product of Direct form II follows a similar trend and is around 1.5% less than that of Direct form I and around 8% less than that of Transposed Direct form II [Figure 5.10]. Although Direct form II realisation has the lowest AT and AT^2 products, it is Direct form I realisation that has the best power performance of the three realisations [Figure 5.11]. The power delay (PD) product of Direct form I is at least 8.8% less than that of Direct form II and 12% less than that of Transposed Direct form II.

5.4.2.2 Band Pass and Band Stop Filters

Experimental results for these filters are presented in Table 5.13 through Table 5.16. With this type of filters, the area increases by a factor of two since the number of weights is double what it is for low pass and high pass filters. The critical path delays do not change however, as we have seen with FIR filters. With the doubling of the area, we find that Direct form II still has the lowest area delay (AT) product of the three realisations we are studying.

Table 5.13 A, T and P results for band pass filters with stop band ripple -80db and -100db using MDP units

Band Pass	Stop band ripple -80db			Stop band ripple -100db		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct I	5846170	25.46	**33.59**	5846170	25.46	**34.08**
Direct II	5846117	25.26	36.55	5846105	25.26	37.05
Transposed DII	5846105	26.29	38.13	5846105	26.29	38.56

Table 5.14 A, T and P results for band pass filters with stop band ripple -120db and -140db using MDP units

Band Pass	Stop band ripple -120db			Stop band ripple -140db		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct I	5846105	25.46	**32.59**	5846105	25.46	**35.41**
Direct II	5846170	25.26	35.64	5846182	25.26	38.56
Transposed DII	5846105	26.29	37.23	5846170	26.29	39.72

Table 5.15 A, T and P results for band stop filters with stop band ripple -80db and -100db using MDP units

Band Stop	Stop band ripple -80db			Stop band ripple -100db		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct I	5846182	25.46	**33.59**	5846170	25.46	**33.58**
Direct II	5846117	25.26	36.30	5846182	25.26	36.49
Transposed DII	5846105	26.29	38.07	5846117	26.29	38.13

Table 5.16 A, T and P results for band stop filters with stop band ripple -120db and -140db using MDP units

Band Stop	Stop band ripple -120db			Stop band ripple -140db		
	A um^2	T ns	P mW	A um^2	T ns	P mW
Direct I	5846105	25.46	**35.06**	5846105	25.46	**33.78**
Direct II	5846235	25.26	37.96	5846170	25.26	36.52
Transposed DII	5846105	26.29	39.51	5846194	26.29	38.26

For these filters too, the area delay (AT) product of Direct form II is around 1% less than that of Direct form I and around 4% less than that of Transposed Direct form II. The area delay^2 (AT^2) product of Direct form II follows a similar trend and is around 1.5% less than that of Direct form I and around 8% less than that of Transposed Direct form II [Figure 5.12].

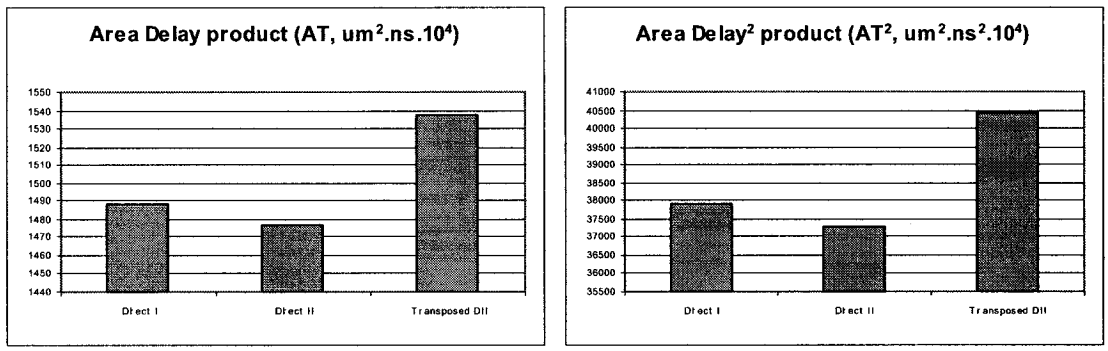

Figure 5.12 Area delay products for band pass and band stop filters using MDP units

The power behaviour of these filters is not different than that of the low pass and high pass filters. For band pass and band stop filters, Direct form I remains the network structure with

the lowest power delay (PD) product [Figure 5.13]. The difference between Direct form I and Direct form II realisations is small, though, as the PD product of Direct form I is less than that of Direct form II by around 8% at the most. In contrast, both realisations have a significant power advantage over the Transposed Direct form II. Power delay (PD) product for the latter is higher than that of Direct form I and Direct form II by around 13% and around 5% respectively. From the above we can say that Transposed Direct form II realisation is the least desirable of the three realisations to implement with band pass and band stop filters. Although Direct form II has the lowest AT and AT^2 products, Direct form I would be the realisation of choice when considering power performance.

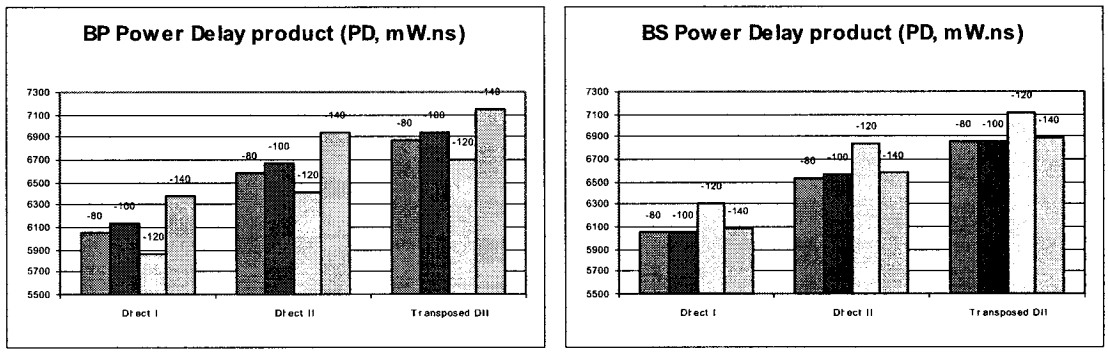

Figure 5.13 Power delay product for band pass and band stop filters using MDP units

5.4.3 Correlation Between Power Behaviour and Internal Activity

To help understand the power behaviour of the structures, frequency distributions of pre-alignment and normalization shifts, their rate of change and the relevant bit level activities and other activity statistics have been gathered [22]. The power behaviour of the different realisations can be explained by considering the activity figures in Table 5.17 through Table 5.24. Although the number of adder shifts of Direct form II is sometimes lower than that

of Direct form I, the difference is very small. For low pass filters the difference is negligible, less than 0.5%, while it does not exceed 3.5% for high pass filters. For band pass filters the difference is around 5%, while it does not exceed 3% for band stop filters. The toggling distance of Direct form II, however, is consistently lower than that of Direct form I by at least 7% for low pass filters and 9% for high pass filters. For band pass and band stop filters, the toggling distance is lower by at least 7% and 18% respectively. For low pass, high pass, band pass and band stop filters the number of multiplier shifts of Direct form II is higher than that of Direct form I by around 35%. The activity figures of Transposed Direct form II realisation are noticeably higher than those of Direct form I and Direct form II realisations as it is clearly illustrated in Figure 5.14.

Table 5.17 Low pass filters activity with -80db and -100db

Low Pass filtering	-80db			-100db		
	Direct I	Direct II	Transposed DII	Direct I	Direct II	Transposed DII
Simulation time (ns)	23053320	23053320	23051880	23053320	23053320	23051880
Total Adders shift	7071894	7076563	10680703	7243370	7219808	11738593
Total togg.distance	2740457	2603138	3480983	2674112	2493449	3474755
Multipliers. shift	1572493	2067754	2107348	1621802	2108872	2158970

Table 5.18 Low pass filters activity with -120db and -140db

Low Pass filtering	-120db			-140db		
	Direct I	Direct II	Transposed DII	Direct I	Direct II	Transposed DII
Simulation time (ns)	23053320	23053320	23051880	23053320	23053320	23051880
Total Adders shift	6783874	6718593	12564841	6873001	6814312	13188914
Total togg.distance	2553384	2334108	3535323	2438109	2249436	3513150
Multipliers. shift	1763051	2230545	2305328	1858618	2332348	2395479

Table 5.19 High pass filters activity with -80db and -100db

High Pass filtering	-80db			-100db		
	Direct I	Direct II	Transposed DII	Direct I	Direct II	Transposed DII
Simulation time (ns)	23053320	23053320	23051880	23053320	23053320	23051880
Total Adders shift	7515686	7791666	11698724	7511600	7769649	11725729
Total togg.distance	4063102	3704986	5478884	4063932	3701644	5441723
Multipliers. shift	1529558	1958647	1945017	1897192	2713058	2701652

Table 5.20 High Pass filters activity with -120db and -140db

High Pass filtering	-120db			-140db		
	Direct I	Direct II	Transposed DII	Direct I	Direct II	Transposed DII
Simulation time (ns)	23053320	23053320	23051880	23053320	23053320	23051880
Total Adders shift	7514329	7795435	11754269	7536771	7778700	11817687
Total togg.distance	4068357	3738971	5424915	4094079	3716547	5446982
Multipliers. shift	2114637	2901508	2866473	2055142	2803943	2783628

Table 5.21 Band Pass filters activity with -80db and -100db

Band Pass filtering	-80db			-100db		
	Direct I	Direct II	Transposed DII	Direct I	Direct II	Transposed DII
Simulation time (ns)	23053320	23053320	23051880	23053320	23053320	23051880
Total Adders shift	15436487	16328623	21882501	15302194	16036465	22061980
Total togg.distance	7682740	7165363	10115609	7621194	7087756	10016629
Multipliers. shift	3582895	5161504	5128417	3484542	5026172	5031374
Total shift	19019382	21490127	27010918	18786736	21062637	27093354

Table 5.22 Band Pass filters activity with -120db and -140db

Band Pass filtering	-120db			-140db		
	Direct I	Direct II	Transposed DII	Direct I	Direct II	Transposed DII
Simulation time (ns)	23053320	23053320	23051880	23053320	23053320	23051880
Total Adders shift	15220121	16008546	22415016	14898184	15634912	23220223
Total togg.distance	7578780	7087869	10039818	7355259	6937506	9751415
Multipliers. shift	3653555	5198517	5190808	4075396	5630074	5613925
Total shift	18873676	21207063	27605824	18973580	21264986	28834148

Table 5.23 Band Stop filters activity with -80db and -100db

Band Stop filtering	-80db			-100db		
	Direct I	Direct II	Transposed DII	Direct I	Direct II	Transposed DII
Simulation time (ns)	23053320	23053320	23051880	23053320	23053320	23051880
Total Adders shift	15230065	15887837	21786762	15191194	15734091	21470465
Total togg.distance	7795372	6376713	9586762	7848581	6367388	9485150
Multipliers. shift	3734614	5161678	5161376	3729837	5077855	5064685
Total shift	18964679	21049515	26948138	18921031	20811946	26535150

Table 5.24 Band Stop filters activity with -120db and -140db

Band Stop filtering	-120db			-140db		
	Direct I	Direct II	Transposed DII	Direct I	Direct II	Transposed DII
Simulation time (ns)	23053320	23053320	23051880	23053320	23053320	23051880
Total Adders shift	15202602	15656282	21443519	15219589	15639249	21464290
Total togg.distance	7883097	6390262	9468191	7907983	6422002	9503164
Multipliers. shift	3840729	5138093	5118644	3653940	4929844	4906117
Total shift	19043331	20794375	26562163	18873529	20569093	26370407

For the structures implemented with the MDP arithmetic units, the power dissipation share of the adders is, at best, 60% less than that of the multipliers [Table 5.25]. This makes the power behaviour of these structures less sensitive to that of the adders in favour of a higher sensitivity to that of the multipliers. The higher sensitivity helps the multipliers dominate the power behaviour of the structures when the overall results are very close. This is greatly displayed in the case of Direct form I and Direct form II realisations. For these two realisations, the activity figures of the adders are very close and do not always follow the power behaviour of the structures as seen in Figure 5.14.

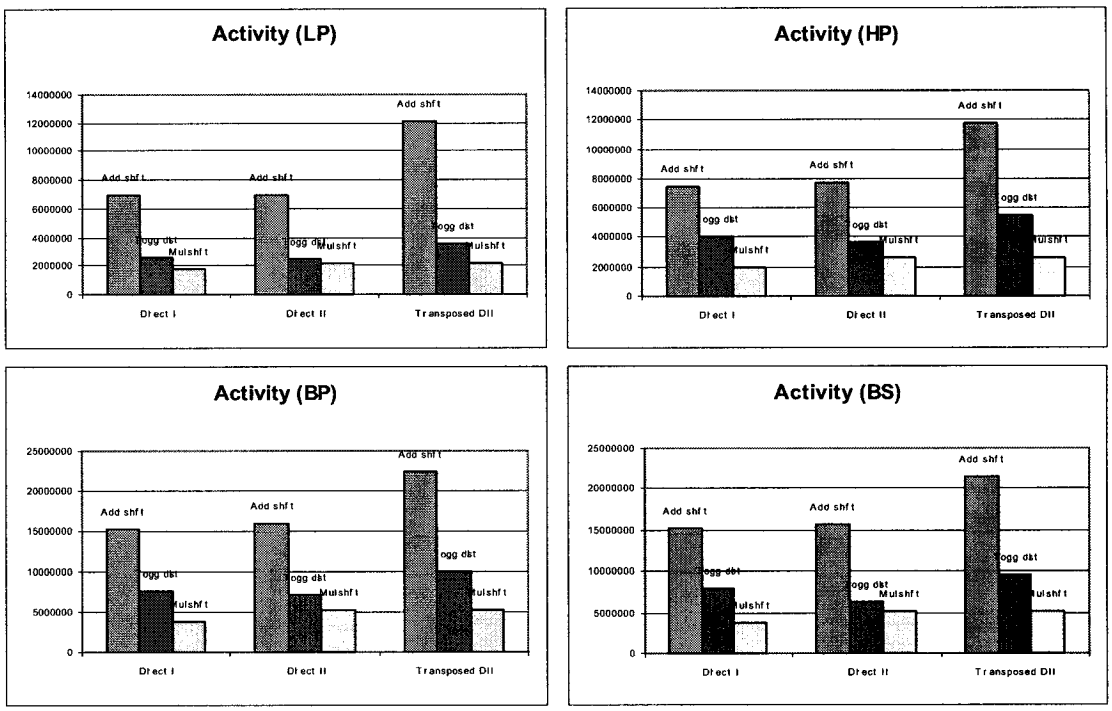

Figure 5.14 Break-down of activity for the IIR filters

The activity figures of the multipliers, however, are significantly different and, due to their greater influence, are the decisive factor in the power behaviour. The activity inside the

adders of the Transposed Direct form II realisation is considerably higher than that of Direct form I and Direct form II realisations. And although the activity of the multipliers of Transposed Direct form II realisation is slightly lower than that of Direct form II realisation, it is not enough to give the former any power advantage. The interconnections inside the Transposed Direct form II realisation account for 25% of the total power dissipation [Table 5.25], which is large enough to compensate for small differences, resulting from either adders or multipliers, between the structures. Direct form I has the magnitude of significand alignment shifts as well as the rate of change smaller than Direct form II and Transposed Direct form II realisations. As a result, the Direct form I realisation requires less power than the Direct form II and Transposed Direct form II realisations to perform these shifts [3] as can be seen in Figure 5.11 and Figure 5.13.

Table 5.25 Contribution percentages to power dissipation with MDP units

Block	Direct I	Direct II	Transposed DII
Adders	25	23	20
Multipliers	60	62	55
Total arithmetic units	85	85	75
Interconnections	15	15	25

For the structures implemented with the SDP arithmetic units, the power dissipation share of the adders is less than that of the multipliers by around 36%, 44% and 61% for Direct form I, Direct form II and Transposed Direct form II realisations [Table 5.26]. This means that with SDP units, the structures are more sensitive to the power behaviour of the adders than with MDP units. Evidently, this results in decreased sensitivity to the power behaviour

of the multipliers. Because of this, the multipliers alone can no longer dominate the power behaviour of the structures when the overall results are very close. This is greatly displayed in the case of Direct form I and Direct form II realisations. For these two realisations, the activity figures of the adders are very close and do not always follow the power behaviour of the structures as seen in Figure 5.14. The activity figures of the multipliers, however, are significantly different but due to their reduced influence cannot, on their own, set the power behaviour of the structures. This situation gives the adders a greater influence over the power behaviour. As can be seen from the results, it is the toggling distance of the adders that is the decisive factor in giving the power advantage to the Direct form II instead of the Direct form I realisation [Figure 5.14][Figure 5.13]. The interconnections inside the Transposed Direct form II realisation, implemented with SDP units, account for 28% of the total power dissipation [Table 5.26]. Here too, it is large enough to compensate for small differences, resulting from either adders or multipliers, between the structures.

Table 5.26 Contribution percentages to power dissipation with SDP units

Block	Direct I	Direct II	Transposed DII
Adders	36	33	20
Multipliers	56	59	52
Total arithmetic units	92	92	72
Interconnections	8	8	28

With SDP arithmetic units, Direct form II has the magnitude of significand alignment shifts as well as the rate of change smaller than Direct form I and Transposed Direct form II realisations. As a result, the Direct form II realisation requires less power than the Direct form

I and Transposed Direct form II realisations to perform these shifts [3] as can be seen in Figure 5.7 and Figure 5.9.

5.4.4 Multiple Data Path vs. Single Data Path

Comparing the results of Table 5.1 through Table 5.8 and Table 5.9 through Table 5.16, we can clearly see that the power advantage of the MDP units over their SDP counterparts, which we have seen in Chapter 3 and Chapter 4, disappears altogether. In fact, the results illustrated in Figure 5.15 clearly show that it is the IIR filters implemented with SDP arithmetic units that have the power advantage instead of those implemented with MDP units. This advantage is very noticeable with all the realisations and it is 8%, 17.5% and 7% for Direct form I, Direct form II and Transposed Direct form II realisations respectively.

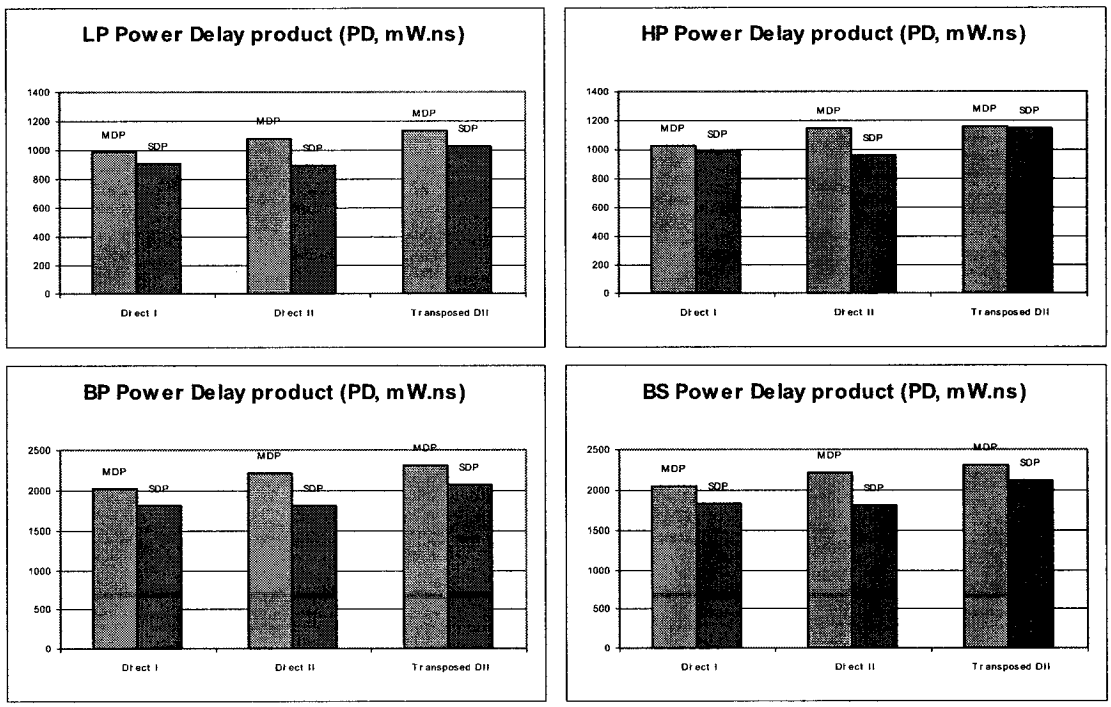

Figure 5.15 Average Power delay product for the IIR filter realisations

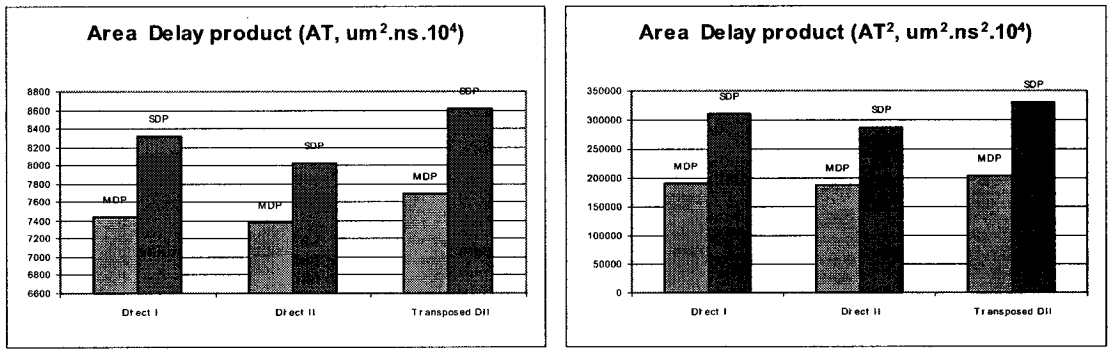

Figure 5.16 Average Area delay products for the IIR filter realisations

The area delay (AT) and area delay^2 (AT^2) products are essential for the comparison of the filter structures using MDP and SDP arithmetic units. Figure 5.16 show the advantage of using MDP units in all three filter realisations. Although the structures are around 30% larger when implemented with MDP units than with SDP units, their critical path delays are around 30% lower. The Area Delay (AT) and Area Delay^2 (AT^2) products reveal a considerable advantage in favour of the structures implemented with MDP units. Using MDP units reduces AT by around 10%, and AT^2 by around 39%.

5.4.5 Second-Order-Section vs. Nth-Order-Section

A brief evaluation of the IIR filter structures in Nth-Order-Section (NOS) implementation was performed to compare its performance with that of the Second-Order-Section implementation used in the experiments discussed earlier in this chapter. A simple experiment confirmed that NOS implementations have smaller area due to the lower number of coefficients needed for these implementations. An order 8 low pass IIR filter with a pass band ripple of 0.03, a stop band ripple of -80db and a cut-off frequency of 0.2 in Direct form I

has an area around 11% smaller in NOS than in SOS implementation. The critical path delay, however, sees an increase of around 14% when implemented in NOS due to the high fanout at the output of the feedback loop. The high fanout has a considerable impact on the the power consumption, which is higher in NOS than in SOS implementation by around 20%. The fanout effect is even higher in Direct form II and Transposed Direct form II realisations as can be clearly seen from Figure 5.4 and Figure 5.5. For a Direct form II realisation the delay with NOS implementation is 44% higher than with SOS implementation. For a Transposed Direct form II realisation, it is 18% higher. Due to the high fanout effect, it was decided from the results discussed in this paragraph that the NOS implementation will not be investigated further since its power behaviour is predetermined to be less interesting than that of the SOS implementation.

5.5 Floating-Point vs. Fixed-Point

With the IIR filters we also see that under the same operating conditions, fixed-point and floating-point units have widely different performances as can be seen in Figure 5.19. The same filter structure, in this case order 8 pipe-lined Direct form I, the area with typical fixed-point units is around 55% less than with floating-point units. The fixed point's critical path delay is about 60% smaller than the floating-point's. With fixed-point units the total power dissipation is around only 13% smaller than with floating-point units. The contribution of adders to the total power dissipation is around 12% with fixed-point units and 33% with floating-point units. The multipliers contribution is 79% and 58% respectively. The rest of the power is dissipated by the network interconnections.

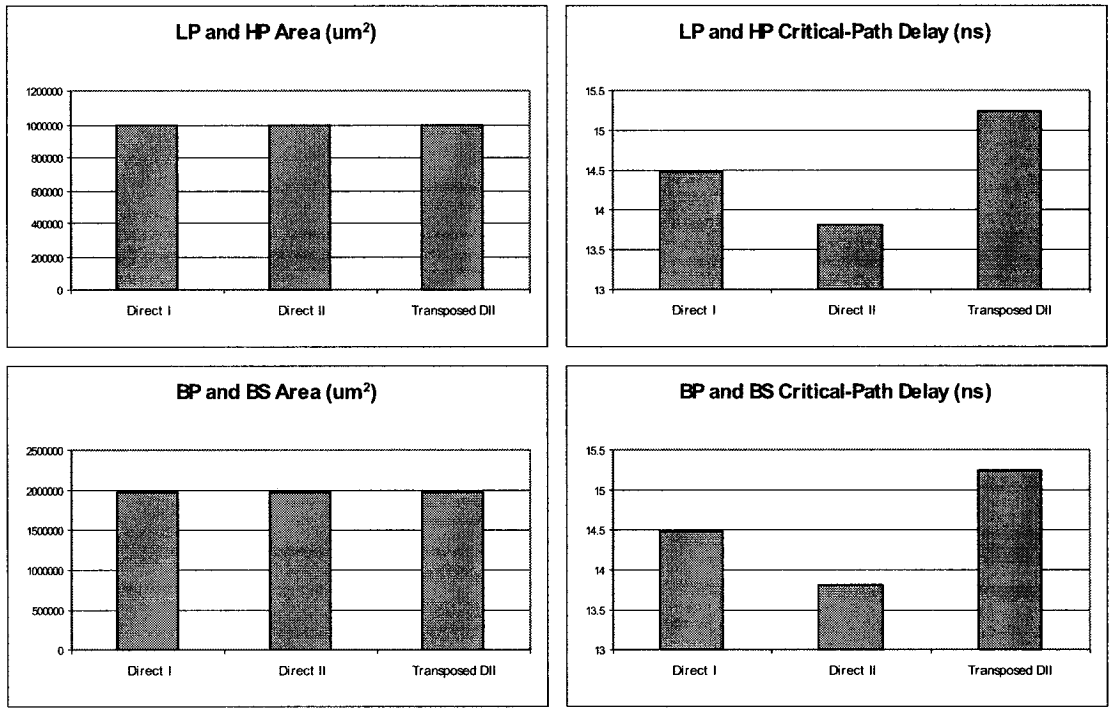

Figure 5.17 Area, Delay and Power Performance of the Fixed-Point filters

The results also show that with fixed-point arithmetic units, Direct form II is delay and power optimal for all four filtering functions. Transposed Direct from II remains the least desirable of the three structures as can be seen in Figure 5.17 and Figure 5.18.

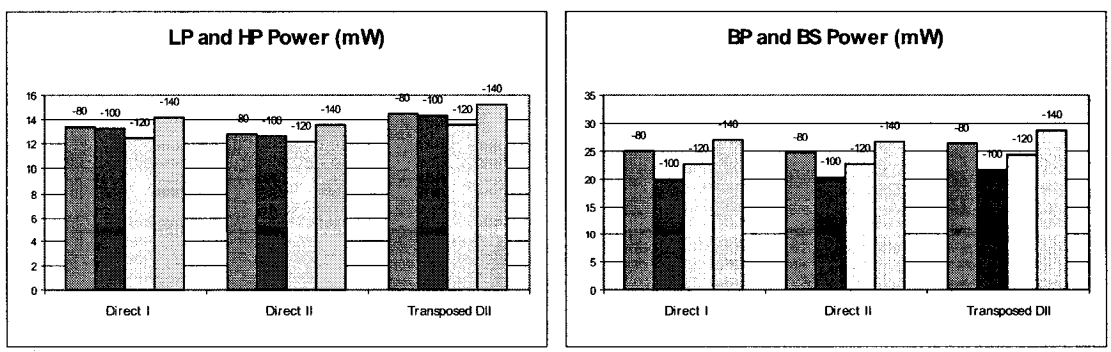

Figure 5.18 Power Performance of the Fixed-Point filters

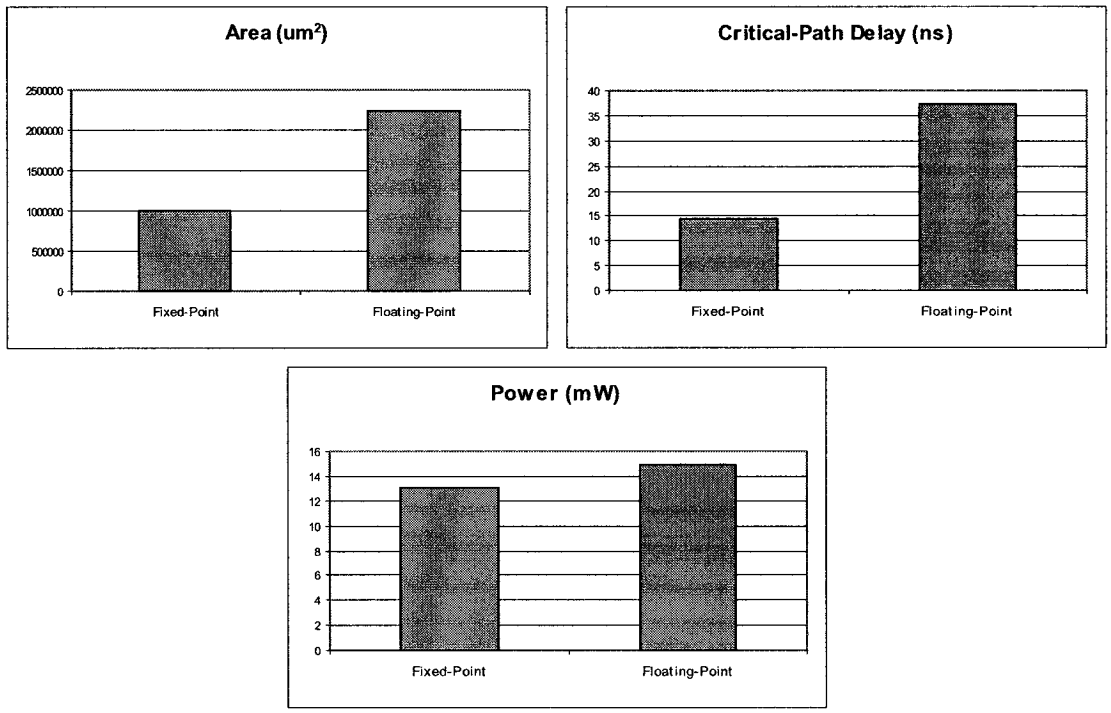

Figure 5.19 Area, Delay and Power Performance of the Fixed and Floating-Point filters

5.6 Conclusion

From the experiments performed on the different IIR filter structures, we notice that in general the Transposed Direct form II structure is the least desirable of the three realisations to implement. Its measured area, delay and power performance figures (AT, AT^2 and PD) are highest with all four filtering functions and with the two sets of arithmetic units, SDP and MDP. In contrast, Direct form II has the lowest AT and AT^2 products. Focusing solely on power performance, Direct form I would be the realisation of choice when using the MDP arithmetic units. When using SDP units, however, it is Direct form II that presented the best power performance. Another conclusion we can draw from the set of experiments is that the Multiple Data Path units have an area and delay advantage over their single data path

counterparts. We see, from the results we have gathered, that an IIR filter structure using MDP units has less AT and AT^2 products than the same filter structure using SDP units. Structures using MDP units, however, consume more power than the same structures using SDP units contrary to what we have seen with the FIR filters in Chapter 4.

5.7 Summary

In this chapter, we measured the area, delay and power performance of three IIR filter structures with different filtering functions, different filter specifications and different input data samples. We began by presenting the different filter structures to be evaluated and how they were implemented. We, also, listed the data samples used to avoid any ambiguity. The results obtained were compiled and presented in the form of tables and graphs for each of the filtering functions and analysed. We finally drew conclusions from the results we gathered. First, Direct form II structure has the best area and delay performance. Second, Direct form II structure is power optimal with SDP units while Direct form I structure is power optimal with MDP units. Third, the MDP arithmetic units have a significantly lower AT and AT^2 products than their SDP counterparts in IIR filtering applications while their power consumption proved to be higher. We also showed that the delay and power performance of IIR filters implemented in SOS are significantly better than in NOS even though the area of the latter is smaller. Last but not least, when using fixed-point arithmetic units the Direct form II structure is delay and power optimal.

Chapter 6

Contribution and Future Works

6.1 Contribution

Contribution towards addressing common errors about the performance of arithmetic units in the floating-point domain and the evaluation of high level design decisions for performance optimal realisations of floating-point FIR and IIR filters were presented in Chapter 3, Chapter 4 and Chapter 5. We evaluated the performance of a number of floating-point arithmetic units in three different applications. The results showed that multipliers and adders are equally important in the floating-point domain and the performance of the adder cannot be neglected as it is the case in integer and fixed-point arithmetic. We presented a simple and easy way of pipe-lining the digital filter structures by using pipe-lined arithmetic units. We compared the power/performance of several digital filter network structures with different filtering functions and classified them in accordance with their performance. We also compared performance figures of different implementations of the filter structures and drew conclusions as to performance optimal conditions. The results of this work can help in the selection of the best suitable network structure for a given application.

6.2 Feasibility Evaluation

Advancement in VLSI technology is allowing more and more complex systems to be implemented on silicon. The large scale of integration that is possible nowadays has made it possible for lap-top computers to have more computing power than room size computer systems from around two decades ago. Mobile phones that weighed few kilos less than two

decades ago are readily available these days in pocket size devices. The costs of the aforementioned devices have seen a significant reduction too. Today, almost everybody has a personal computer while not so long ago it was the privilege of only a few. Complex algorithms are implemented not for the sake of complexity but because they have certain advantages. Floating-point arithmetic's major advantage is its significantly wider dynamic range than fixed-point arithmetic. Its disadvantage, however, is the size or area of its implementation. For the same bit width, 32 bits in our case, and for the same DSP structure, Direct form order 8 FIR filter, the area with typical fixed-point units is between 40% to 50% less than with floating-point units [Figure 4.25]. Bear in mind that, with 32 bits, the dynamic range of the fixed-point arithmetic is insignificant in comparison to the floating-point's. The dynamic range of 32 bits in floating-point arithmetic is equivalent to that of 426 bits in fixed-point arithmetic. It makes no sense at all to implement the design with this bit-width knowing that with 64 bits, the fixed-point design is around 150% larger than the floating-point's. In applications where the dynamic range is of critical importance, it makes more sense to use floating-point arithmetic if the benefits outweigh the costs of the increase in area. If a large dynamic range is not required, fixed-point arithmetic makes more sense as the area, delay and power dissipation figures, for the same bit width, are less than with floating-point arithmetic.

6.3 Future Related Works

The logical next step for this work is to evaluate the impact of using pipe-lined arithmetic units on the filter structures in more complex filtering applications like adaptive filtering for example. The performance of the different designs we evaluated in this thesis can be

further enhanced by customising the synthesis and optimization constraints for each strucutre. The impact of further pipe-lining the arithmetic units, i.e. increase the number of pipeline stages, is certainly worth evaluating. The re-evaluation of the NOS implementations of IIR filters with fanout reducing techniques should also be attempted.

References

[1] Frank Poppen and Wolfgang Nebel, "*Evaluation of a behavioural Level Low Power Design Flow Based on a Design Case*", OFFIS Research Institute, Synopsys Users Group (SNUG) Boston 2001.

[2] SolvNet Doc Id 901293 "*New Methodology for Low-Power*", Synopsys Inc., Mountain View, CA, October 2001.

[3] Rajan V. K. Pillai, "*On Low Power Floating Point Data Path Architectures*" Ph.D. Thesis, Concordia University, Montreal, QC, 1999.

[4] Bill Steele, "*Researchers develop a tiny battery with extra, extra long life*", Cornell Chronicle, front page, November 7, 2002.

[5] A.T. Erdogan and T. Arslan, "*Low Power Implementation of Linear Phase FIR Filters for Single Multiplier CMOS Based DSPs*", In Proceedings of the 1998 IEEE International Symposium on Circuits And Systems (ISCAS), pp. V425-V428, May-June 1998.

[6] An-Yeu Wu and K. J. Ray Liu, "*Algorithm-Based Low-Power transform Coding Architectures: The Multirate Approach*", IEEE Transactions on Very large Scale Integration Systems, Vol. 6, No 4, pp. 707-718, December 1998.

[7] Cheng-Yu Pai, A. J. Al-Khalili and W. E. Lynch, "*Low-Power Constant-Coefficient Multiplier Generator*", Annual IEEE Internatioanl ASIC/SOC Conference, Washington, September 2001.

[8] Syed Y.A. Shah, "*On Synthesis and Optimization of Floating Point Units*", M.A.Sc. Thesis, Concordia University, Montreal, Quebec, Canada, 2000.

[9] C-Y. Tsui, M. Pedram, C-h. Chen, and A. Despain, "*Low power state assignment targeting two- and multi-level logic implementations*", In Proccedings of the 1994 IEEE International Conference on Computer Aided Design, pp. 82-87, November 1994.

[10] J. Monteiro, S. Devadas, and A. Ghosh, "*Retiming sequential circuits for low power*", In Proccedings of the 1993 IEEE International Conference on Computer Aided Design, pp. 398-402, November 1993.

[11] Masanori Hashimoto, Hidetoshi Onodera and Keikichi Tamaru, "*A Power Optimization Method Considering Glitch Reduction by Gate Sizing*", In Proceedings of the 1998 International Symposium on Low Power Electronics and Design (ISLPED), August 1998.

[12] Zia Khan and Gaurav Mehta, "*Automatic Clock Gating for Power reduction*", Synopsys Users Group (SNUG) San Jose 1999.

[13] Harry Veendrick, "*Deep-Submicron CMOS ICs: From Basics to ASICs*", Kluwer BedrijfsInformatie, ISBN 90-5576-128-1, October 1998.

[14] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou, "*Pre computation-based sequential logic optimization for low power*", In Proccedings of the 1994 International Workshop on Low Power Design, pp. 57-62, April 1994.

[15] M. Berkelaar and J. Jess, "*Gate sizing in Mos digital circuits with linear programming*", In Proccedings of the European Design automation Conference, pp. 217-221, 1990.

[16] S. Iman and M. Pedram, "*Multi-level network optimization for low power*", In Proccedings of the 1994 IEEE International Conference on Computer Aided Design, pp. 372-377, November 1994.

[17] C. Moerman, E. Lambers, "*Optimizing DSP: Low Power by Architecture*", www.adelantetechnologies.com.

[18] Ferdinant Sluijs, Kees Hart, Wouter Goeneveld and Stephan Haag, "*Integrated DC/DC converter with Digital Controller*", In Proceedings of the 1998 International Symposium on Low Power Electronics and Design (ISLPED), August 1998.

[19] Roderick Thornton Hinman, "*Recovered Energy Logic: A Logic Family and Power Supply Featuring Very High Efficiency*", Ph.D. Thesis, MIT 1994.

[20] 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, October 1994.

[21] Mike Gladden and Indraneel Das, "*RTL Low Power Techniques for System-On-Chip Designs*", Synopsys Users Group (SNUG) San Jose 1999.

[22] "*Power Compiler Reference Manual*", Version 2001.08 Synopsys Inc., Mountain View, CA.

[23] 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.

[24] Alan V. Oppenheim and Ronald D. Schafer, "*Discrete-Time Signal Processing*", Prentice-Hall, Englewood Cliffs, NJ, 1989.

[25] Steven W. Smith, Ph.D., "*The Scientist and Engineer's Guide to Digital Signal Processing*", California Technical Publishing, 1997.

[26] Jae S. Lim, "*Two Dimensional Signal and Image Processing*", Prentice-Hall 1990.

[27] Robert L. Kay, "*Digital Filters-General form*", Elite Engineering Corp., technical information, June 1996, www.eliteeng.com/digitfil.htm.

[28] 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.

[29] H. Fujii, C. Hori, T. Takada, N. Hatanaka, T. Demura, and G. Ootomo, "*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.

[30] Keshab K. Parhi, "*VLSI Digital Signal Processing Systems: Design and Implementation*", Wiley, New York 1999.

Appendix

List of Acronyms

VLSI	Very Large Scale Integration
IEEE	Institute of Electrical and Electronics Engineer
CMOS	Complementary Metal Oxide Semiconductor
ASIC	Application Specific Integrated Circuit
CAD	Computer Aided Design
SPICE	Software Process Improvement and Capability dEtermination
RTL	Register Transfer Level
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VSS	VHDL System Simulator
MAC	Multiply - Accumulator
PDA	Personal Digital Assistant
EKG	Electrocardiogram, also known as ECG
TSMC	Taiwan Semiconductor Manufacturing Company
CMC	Canadian Microelectronics Corporation
RAM	Random Access Memory
CPU	Central Processing Unit

DC	Direct Current
RF	Radio Frequency
FP	Floating Point
NaN	Not a Number
SDP	Single Data Path
MDP	Multiple Data Path
DSP	Digital Signal Processing
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
NOS	Nth Order Section
SOS	Second Order Section
LP	Low Pass
HP	High Pass
BP	Band Pass
BS	Band Stop
fd	forward
fb	feed-back
MHz	Mega Hertz

GB	Giga Bytes
mW	milli Watts
ns	nano second
um	micro meter (micron)