LOW POWER VLSI DESIGN OF A FIR FILTER USING DUAL EDGE TRIGGERED CLOCKING STRATEGY

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology in
VLSI Design and Embedded System

By

SAKSHI GUPTA
Roll No: 20607016

Department of Electronics & Communication Engineering
National Institute of Technology
Rourkela
2008

LOW POWER VLSI DESIGN OF A FIR FILTER USING DUAL EDGE TRIGGERED CLOCKING STRATEGY

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology in

VLSI Design and Embedded System

By

SAKSHI GUPTA

Roll No: 20607016

Under the Guidance of

Prof. K. K. MAHAPATRA

Department of Electronics & Communication Engineering

National Institute of Technology

Rourkela

2008

ACKNOWLEDGEMENTS

It is a pleasure to thank many people who made this thesis possible.

I would like to take this opportunity to express my gratitude and sincere thanks to my supervisor **Prof. K. K. Mahapatra** for his guidance, insight, and support he has provided throughout the course of this work.

I am grateful to our teachers **Prof. G.S. Rath**, **Prof. G. Panda, Prof. S.K. Patra** and **Dr. S. Meher**. From these teachers I learned about the great role of self-learning and the constant drive for understanding emerging technologies, and a passion for knowledge.

My special thanks go to Phd scholar **Mr. J.K. Das**, research scholars , friends at NIT Rourkela for their encouragement and help throughout the course.

I would like to thank all faculty members and staff of the Department of Electronics and
Communication Engineering, N.I.T. Rourkela for their extreme help throughout course. Finally, I am forever indebted to my mother, sisters, and to my best friend Sonali Gupta for their love understanding, endless patience and encouragement when it was most required. I am also grateful to my friends Manish Ajmeria, Jiju M.V. and Leslin Varghese for their support and encouragement to carry out this work.

Sakshi Gupta

CONTENTS

ABSTRACT

Digital signal processing is an area of science and engineering that has developed rapidly over the past 30 years. This rapid development is a result of the significant advances in digital computer technology and integrated–circuit fabrication. DSP processors are a diverse group, most share some common features designed to support fast execution of the repetitive, numerically intensive computations characteristic of digital signal processing algorithms. The most often cited of these features is the ability to perform a multiply-accumulate operation (often called a "MAC") in a single instruction cycle. Hence in this project a DSP Processor is designed which can perform the basic DSP Operations like convolution, fourier transform and filtering. The processor designed is a simple 4-bit processor which has single data line of 8-bits and a single address bus of 16-bits. With a set of branch instructions the project DSP will operate as a CISC processor with strong math capabilities and can perform the above mentioned DSP operations. The application I have taken is the low power FIR filter using dual edge clocking strategy. It combines two novel techniques for the power reduction which is : multi stage clock gating and a symmetric two-phase level-sensitive clocking with glitch aware re-distribution of data-path registers. Simulation results confirm a 42% reduction in power over single edge triggered clocking with clock gating.

Also to further reduce the power consumption the a low power latch circuit is used. Thanks to a partial pass-transistor logic, it trades time for energy, being particularly suitable for low power low-frequency applications. Simulation results confirm the power reduction. This technique discussed can be implemented to portable devices which needs longer battery life and to ASIC's .

List of Figures

Chapter 1

INTRODUCTION

1.1 NEED OF LOW POWER

The density and speed of integrated circuit computing elements has increased roughly exponentially for a period of several decades, following a trend described by Moore's Law. While it is generally accepted that this exponential improvement trend will end, it is unclear exactly how dense and fast integrated circuits will get by the time this point is reached. Working devices have been demonstrated that were fabricated with a MOSFET transistor channel length of 6.3 nanometres using conventional semiconductor materials, and devices have been built that used carbon nanotubes as MOSFET gates, giving a channel length of approximately one nanometre. The density and computing power of integrated circuits are limited primarily by power dissipation concerns.

The increasing prominence of portable systems and the need to limit power consumption (and hence, heat dissipation) in very-high density ULSI chips have led to rapid and innovative developments in low-power design during the recent years. The driving forces behind these developments are portable applications requiring low power dissipation and high throughput, such as notebook computers, portable communication devices and personal digital assistants (PDAs). In most of these cases, the requirements of low power consumption must be met along with equally demanding goals of high chip density and high throughput. Hence, low-power design of digital integrated circuits has emerged as a very active and rapidly developing field of CMOS design.

The limited battery lifetime typically imposes very strict demands on the overall power consumption of the portable system. Although new rechargeable battery types such as Nickel-Metal Hydride (NiMH) are being developed with higher energy capacity than that of the conventional Nickel-Cadmium (NiCd) batteries, revolutionary increase of the energy capacity is not expected in the near future. The energy density (amount of energy stored per unit weight) offered by the new battery technologies (e.g., NiMH) is about 30 Watt-hour/pound, which is still low in view of the expanding applications of portable systems. Therefore, reducing the power dissipation of integrated circuits through design improvements is a major challenge in portable systems design. The need for low-power design is also becoming a major issue in high-performance digital systems, such as microprocessors, digital signal processors (DSPs) and other applications. Increasing chip density and higher operating speed lead to the design of very complex chips with high clock frequencies. Typically, the power dissipation of the chip, and thus, the temperature, increase linearly with the clock frequency.

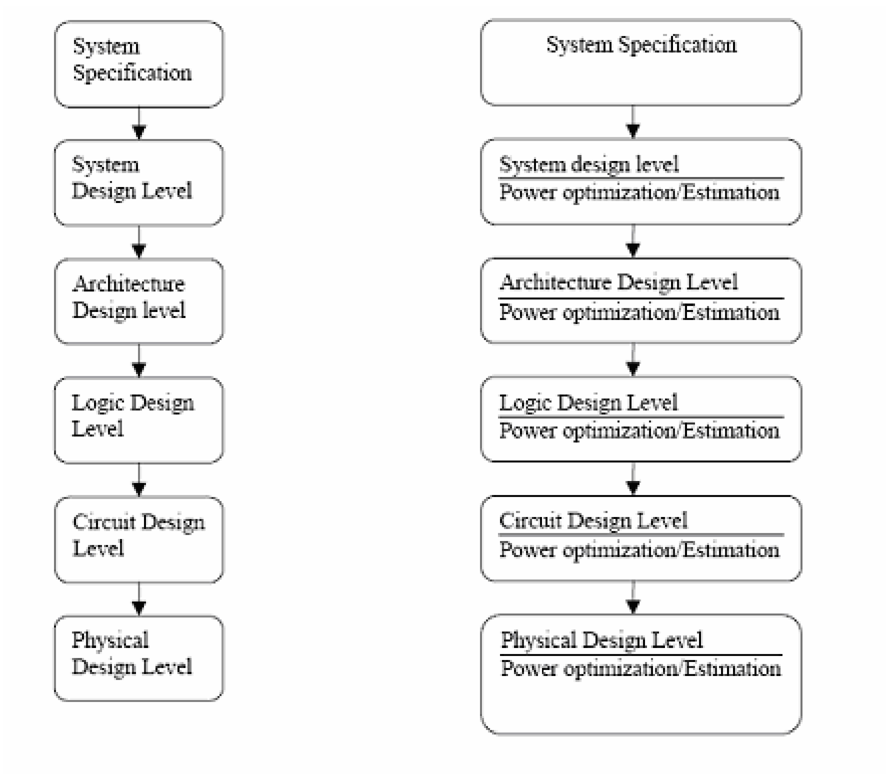
ULSI reliability is yet another concern which points to the need for low-power design. There is a close correlation between the peak power dissipation of digital circuits and reliability problems such as electro-migration and hot-carrier induced device degradation. Also, the thermal stress caused by heat dissipation on chip is a major reliability concern. Consequently, the reduction of power consumption is also crucial for reliability enhancement.

1.1.1 Design flow with and without power

A top-down ordinary VLSI design approach is illustrated in Fig1.1. The figure summarizes the flow of steps that are required to follow from a system level specification to the physical design. The approach was aimed at performance optimization and area minimization. However, introducing the third parameter of power dissipation made the designers to change the flow as you shown in the right-hand side of the Figure 1.1.

In each of the design levels are two important power factors, namely power optimization and power estimation. Power optimization is defined as the process of obtaining the best design knowing the design constraints and without violating design specifications. In order to meet the design and required goal, a power optimization technique unique to that level should be employed. Power estimation is defined as the process of calculating power and energy dissipated with a certain percentage of accuracy and at different phases of the design process. Power estimation techniques evaluate the effect of various optimizations and design modifications on power at different abstraction levels.

Generally a design performs a power optimization step first and then a power estimation step, but within a certain design level there is no specific design procedure. Each design level includes a large collection of low power techniques. Each may result in a significant reduction of power dissipation. However, a certain combination of low power techniques may lead to better results than another series of techniques.



Design Parameters

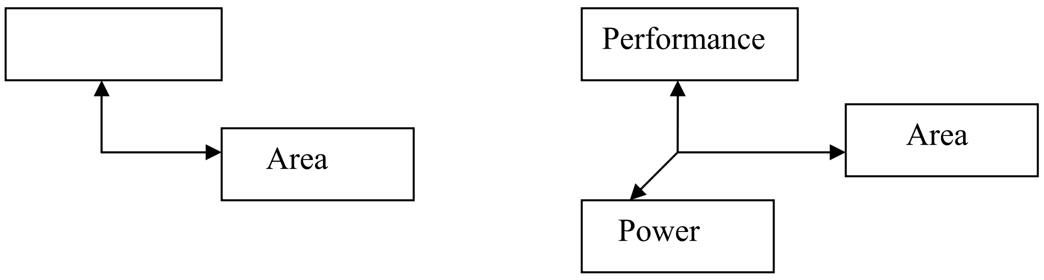


Fig 1.1 VLSI Design Flows

1.1.2 Overview of Power Consumption

The average power consumption in conventional CMOS digital circuits can be expressed as the sum of three main components, namely, (1) the dynamic (switching) power consumption, (2) the short-circuit power consumption, and (3) the leakage power consumption. If the system or chip includes circuits other than conventional CMOS gates that have continuous current paths between the power supply and the ground, a fourth (static) power component should also be considered.

(1) Dynamic (switching) power dissipation

As the name indicates it occurs when signals which goes through the CMOS circuits change their logic state. At this moment energy is drawn from the power supply to charge up the output node capacitance.Charging up of the output capacitance causes transition from 0V to Vdd.Considering an inverter exaple power drawn from the power supply is dissipated as heat in pMOS transitor. On the other hand charge down process causes NMOS transistor to dissipate heat. Output capacitance of the CMOS logic gate consists of below components:

1)*Output node capacitance of the logic gate:* This is due to the drain diffusion region.

2)*Total interconnect capacitance:* This has higher effect as technology node shrinks.

3)*Input node capacitance of the driven gate:* This is due to the gate oxide capacitance.

To find the avearage power energy required to charge up the output node to Vdd and charge down the total output load capacitance to ground level is integrated. Applied input periodic waveform having its period T is assumed to be having zero rise and fall time. Note that average power is independent of transistor size and characteristics.

The dynamic power dissipation of the system is given by the equation

$$P = C_{total}\, V^2 f \quad \text{..............................} \text{ Eq(1.1)}$$

How to reduce dynamic power?

1)reduce power supply voltage Vdd
2)reduce voltage swing in all nodes
3)reduce the switching probabilty (transition factor)
4)reduce load capacitance

(2) The short-circuit power consumption

A CMOS inverter (or a logic gate) is driven with input voltage waveforms with finite rise and fall times, both the nMOS and the pMOS transistors in the circuit may conduct simultaneously for a short amount of time during switching, forming a direct current path between the power supply and the ground. The current component which passes through both the nMOS and the pMOS devices during switching does not contribute to the charging of the capacitances in the circuit, and hence, it is called the short-circuit current component. This component is especially prevalent if the output load capacitance is small, and/or if the input signal rise and fall times are large. The nMOS transistor in the circuit starts conducting when the rising input voltage exceeds the threshold voltage VT,n. The pMOS transistor remains on until the input reaches the voltage level (VDD - |VT,p|). Thus, there is a time window during which both transistors are turned on. As the output capacitance is discharged through the nMOS transistor, the output voltage starts to fall. The drain-to-source voltage drop of the pMOS transistor becomes nonzero, which allows the pMOS transistor to conduct as well. The short circuit current is terminated when the input voltage transition is completed and the pMOS transistor is turned off. An similar event is responsible for the short- circuit current component during the falling input transition, when the output voltage starts rising while both transistors are on.

Note that the magnitude of the short-circuit current component will be approximately the same during both the rising-input transition and the falling-input transition, assuming that the inverter is symmetrical and the input rise and fall times are identical. The pMOS transistor also conducts the current which is needed to charge up the small output load capacitance, but only during the falling-input transition (the output capacitance is discharged through the nMOS device during the rising-input transition). This current component, which is responsible for the switching power dissipation of the circuit is responsible for short-circuit power consumption.

(3) Leakage Power Dissipation

The nMOS and pMOS transistors used in a CMOS logic gate generally have nonzero reverse leakage and subthreshold currents. In a CMOS VLSI chip containing a very large number of transistors, these currents can contribute to the overall power dissipation even when the transistors are not undergoing any switching event. The magnitude of the leakage currents is determined mainly by the processing parameters.

Of the two main leakage current components found in a MOSFET, the reverse diode leakage occurs when the pn-junction between the drain and the bulk of the transistor is reversely biased. The reverse-biased drain junction then conducts a reverse saturation current which is eventually drawn from the power supply. Consider a CMOS inverter with a high input voltage, where the nMOS transistor is turned on and the output node voltage is discharged to zero. Although the pMOS transistor is turned off, there will be a reverse potential difference of VDD between its drain and the n-well, causing a diode leakage through the drain junction. The n-well region of the pMOS transistor is also reverse-biased with VDD, with respect to the p-type substrate. Therefore, another significant leakage current component exists due to the n-well junction. A similar situation can be observed when the input voltage is equal to zero, and the output voltage is charged up to VDD through the pMOS transistor. Then, the reverse potential difference between the nMOS drain region and the p-type substrate causes a reverse leakage current which is also drawn from the power supply (through the pMOS transistor)

1.2 BASIC PRINCIPLES OF LOW POWER DESIGN

Conservation and trade-off are the philosophy behind most low power techniques. The conservation school attempts to reduce power that is wasted without a due course. Low power should be applied at all levels of design abstraction and design activities. Chip area and speed are the major trade-off consideration but a low power design decision also affects other aspects such as reliability, design cycle time, reusability, testability and design complexity.

1.2.1 Reducing Switching Voltage

The dynamic power of digital chips expressed by Eq (1.1) is generally the largest portion of power dissipation. It consist of three terms voltage, capacitance and frequency. Due to the quadratic effect of the voltage term reducing the switching voltage can achieve dramatic savings. The easiest method is to reduce the operating voltage of the CMOS circuit. There are many trade-offs to be considered in voltage reduction. Performance is lost because MOS transistors becomes slower at lower operating voltages. The main reason is that the threshold voltages of the transistors do not scale accordingly with the operating voltage to avoid excessive leakage current.

1.2.2 Reduce Capacitance

Reducing parasitic capacitance in digital design has always been a good way to improve performance as well as power. The real goal is to reduce the product of capacitance and its switching frequency. Signals with high switching frequency should be routed with minimum parasitic capacitance to conserve power. Nodes with large parasitic capacitance should not be allowed to switch high frequency.

1.2.3 Reduce Switching Frequency

For the sake of power dissipation, the techniques for reducing switching frequency have the same effect as reducing capacitance. Again frequency reduction is best applied to signals with large capacitance. The techniques are often applied to logic level design and above. Those applied at a higher abstraction level generally have greater impact. One effective method of reducing switching frequency is to eliminate logic switching that is not necessary for computation.

1.2.4 Reduce Leakage and Static Current

Designers have very little control over the leakage current of the digital circuit. The leakage power problem mainly appears in very low frequency circuits with “sleep modes” where dynamic activities are suppressed. Most leakage reduction techniques are applied at low level design abstraction such as process, device and circuit design.

1.3 MOTIVATION

Energy consumption is a product of average power and delay. And power is linearly proportional to the square of supply voltage. Voltage reduction is an efficient way to reduce power consumption; yet, it also leads to logic speed reduction. However, for signal processing applications, such as digital filter, it is important to maintain a given level of computation or throughput. Hence, parallel architectures should be used to maintain the throughput at a reduced supply voltage . Dual-edge triggered flip-flops is a device level realization of this concept. It can obtain the same data throughput with one half of the clock frequency, thus relaxing the power and clock uncertainty requirements. All these motivated me to carry on this project work.

1.4 OUTLINE OF THE THESIS

In chapter 1 a brief introduction to low power and need of low power is discussed. In chapter two need, advantages of DSP and design of a DSP Processor is discussed. In chapter 3 the advantages of digital filter and FIR fiter architecture is discussed. Chapter 4 discusses in detail about the clock generation, clocking schemes, clock gating techniques. The advantages and disadvantages of various clocking techniques is also discussed. Finally it gives the details of the proposed clocking technique used in FIR filter to reduce the power consumption. Chapter 5 produces the various simulation results. Lastly in chapter six the conclusion and future work is discussed.

Chapter 2

DIGITAL SIGNAL PROCESSOR

2.1 INTRODUCTION

Signal processing is the analysis, interpretation, and manipulation of signals. Signals are electrical representations of time-varying or spatial-varying physical quantities, either analog or digital, and may come from various sources. Signals of interest include sound, images, biological signals such as ECG, radar signals, and many others. Processing of such signals includes filtering, storage and reconstruction, separation of information from noise (for example, aircraft identification by radar), compression (for example, image compression), and feature extraction (for example, speech-to-text conversion). Following are the two subfields of signal processing.

1) Analog Signal Processing
2) Digital Signal Processing

Digital signal processing is an area of science and engineering that has developed rapidly over the past 30 years. This rapid development is a result of the significant advances in digital computer technology and integrated–circuit fabrication. The digital computers and associated digital hardware of three decades ago were relatively large and expensive and, as a consequence, their use was limited to general-purpose non real time (off-line) scientific computations and business applications. The rapid developments in integrated-circuit technology, starting with medium-scale integration and progressing to large scale integration & now very- large scale integration of electronic circuit has spurred the development of powerful, smaller, faster, and cheaper digital computers and special-purpose digital hardware. These inexpensive and relatively fast digital circuits have made it possible to construct highly sophisticated digital systems capable of performing complex digital signal processing functions and tasks, which are usually too difficult and/or too expensive to be performed by analog circuitry or analog signal processing systems. Hence many of the signal processing tasks that were conventionally performed by analog means are realized today by less expensive and often more reliable digital hardware.

Not only do digital circuits yield cheaper and more reliable systems for signal processing they have other advantages as well. In particular, digital processing hardware allows programmable operations. Through software, one can more easily modify the signal processing functions to be performed by the hardware. Thus digital hardware and associated software provide a greater degree of flexibility in system design. Also, there is often a higher order of precision achievable with digital hardware and software compared with analog circuits and analog signal processing systems.

For all these reasons, there has been an explosive growth in digital signal processing theory and applications aver the past three decades. The influence of digital signal processing (DSP) is expanding at a dramatic pace. DSP is a key enabling technology for many applications in fields such as telecommunications, consumer electronics, disk drives, and navigation. DSP functions can be implemented using a range of implementation approaches: ASICs, FASICs, general-purpose processors, and programmable digital signal processors are all commonly used.

2.1.1 Advantages of Digital over Analog Signal Processing

Digital signal processing techniques have numerous advantages. Digital circuits do not depend on precise values of digital signals for their operation. Digital circuits are less sensitive to changes in component values. They are also less sensitive to variations in temperature, ageing and other external parameters. Digital processing of a signal facilitates the sharing of a single processor among a number of signals by time-sharing. This reduces the processing cost per signal. Also multi-rate processing is possible only in digital domain. Storage of digital data is very easy. Digital processing is much more suited for processing very low frequency signals.

2.1.2 Basic Elements of a Digital Signal Processing Systems

Most of the signals encountered in science and engineering are analog in nature. That is, the signals are functions of a continuous variable, such as time or space, and usually take on values in a continuous range. Such signals may be processed directly by appropriate analog systems (such as filters or frequency analyzers) or frequency multipliers for the purpose of changing their characteristics or extracting some desired information. In such a case we say that the signal has been processed directly in its analog form, as shown in Fig 2.1.

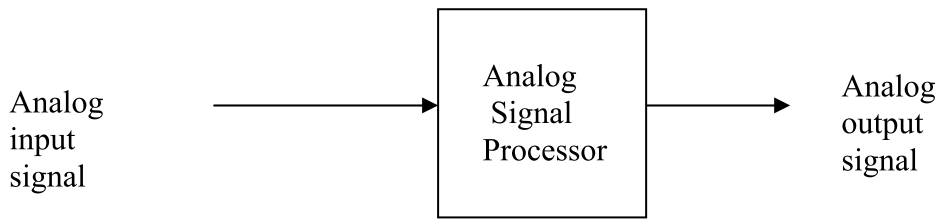


Fig 2.1 : Analog Signal Processing

Both the input signal and the output signal are in analog form. Digital signal processing provides an alternative method for processing the analog signal as shown in Fig 2.2. To perform the processing digitally, there is a need for an interface between the analog signal and the digital processor. This interface is called an analog to digital converter. The output of the A/D converter is a digital signal that is appropriate as an input to the digital processor.

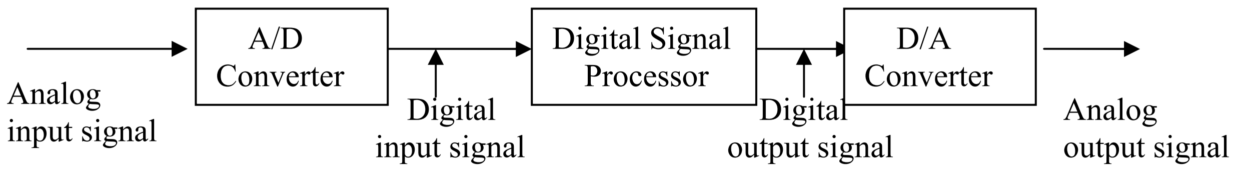


Fig 2.2 : Block Diagram of a Digital Signal Processing System.

The digital signal processor may be a large programmable digital computer or a small microprocessor programmed to perform the desired operations on the input signal. It may also be a hardwired digital processor configured to perform a specified set of operations on the input signal. Programmable machines provide the flexibility to change the signal processing operations through a change in the software, whereas hardwired machines are difficult to reconfigure.

2.2 WHAT IS A DSP PROCESSOR?

While DSP processors are a diverse group, most share some common features designed to support fast execution of the repetitive, numerically intensive computations characteristic of digital signal processing algorithms. The most often cited of these features is the ability to perform a multiply-accumulate operation (often called a "MAC") in a single instruction cycle. A single-cycle MAC operation is extremely useful in algorithms that involve computing a vector dot-product, such as digital filters. Such algorithms are very common in DSP applications. To achieve a single-cycle MAC, all DSP processors include a multiplier and accumulator as central elements of their data-paths. In addition, to allow a series of MAC operations to proceed without the possibility of arithmetic overflow, DSP processors generally provide extra bits in the accumulator to accommodate the bit growth resulting from the repeated additions.

A second feature shared by DSP processors is the ability to complete several accesses to memory in a single instruction cycle. This allows the processor to fetch an instruction while simultaneously fetching operands for the instruction, and/or storing the result of the previous instruction to memory. Typically, multiple memory accesses in a single cycle are possible only under restricted circumstances. For example, usually all but one of the memory locations accessed must reside on-chip, and multiple memory accesses can only take place in conjunction with certain instructions. To support multiple simultaneous memory accesses, DSP processors use multiple on-chip buses, multi-ported on-chip memories, and in some cases multiple independent memory spaces.

To allow numeric processing to proceed quickly, DSP processors incorporate one or more dedicated address generation units. Once configured, the address generation units operate in parallel with the execution of arithmetic instructions, forming the addresses required for data memory accesses. The address generation units typically support addressing modes tailored to DSP applications. For example, these usually include register-indirect modes with post-increment for traversing arrays, and circular ("modulo") addressing for managing circular buffers.

Because many DSP algorithms involve performing repetitive computations, most DSPs provide hardware support for efficient looping. Often, a special loop or repeat instruction is provided which allows the programmer to implement a for-next loop without expending any instruction cycles for updating and testing the loop counter and branching to the top of the loop. Finally, to allow low-cost, high-performance input and output, many DSPs incorporate one more serial or parallel I/O interfaces, and specialized I/O handling mechanisms such as low- overhead interrupts or DMA.

2.2.1 Fixed Versus Floating Point

DSP chip word size determines resolution and dynamic range. In the fixed point processors, a linear relationship exists between word size and dynamic range. The fixed point DSPs are either 16 or 24data bits wide. There are four common ways that these $2^{16} = 65,536$ possible bit patterns can represent a number. In unsigned integer, the stored number can take on any integer value from 0 to 65,535. Similarly, signed integer uses two's complement to make the range include negative numbers, from -32,768 to 32,767. With unsigned fraction notation, the 65,536 levels are spread uniformly between 0 and 1. Lastly, the signed fraction format allows negative numbers, equally spaced between -1 and 1.

The floating point chip perform integer or real arithmetic. Normally, floating point DSP formats are 32 data bits wide and in which 24 bits form the mantissa and 8 bits make up the exponent. This results in many more bit patterns than for fixed point, 2^{32} = 4,294,967,296 to be exact. A key feature of floating point notation is that the represented numbers are *not* uniformly spaced. All floating point DSPs can also handle fixed point numbers, a necessity to implement counters, loops, and signals coming from the ADC and going to the DAC. However, this doesn't mean that fixed point math will be carried out as quickly as the floating point operations; it depends on the internal architecture.

Fixed point arithmetic is much faster than floating point in general purpose computers. However, with DSPs the speed is about the same, a result of the hardware being highly optimized for math operations. The internal architecture of a floating point DSP is more complicated than for a fixed point device. All the registers and data buses must be 32 bits wide instead of only 16; the multiplier and ALU must be able to quickly perform floating point arithmetic, the instruction set must be larger (so that they can handle both floating and fixed point numbers), and so on. Floating point (32 bit) has better precision and a higher dynamic range than fixed point (16 bit) . In addition, floating point programs often have a shorter development cycle, since the programmer doesn't generally need to worry about issues such as overflow, underflow, and round-off error. On the other hand, fixed point DSPs have traditionally been cheaper than floating point devices.

2.2.2 DSP versus General Microprocessors

DSPs differ from microprocessors in a number of ways. Microprocessors are typically built for a range of general purpose functions, and normally run large blocks of software, such as operating systems like UNIX. Microprocessors aren't often called upon for real-time computation. And though microprocessors have some numeric capabilities, they're nowhere near fleet enough for most DSP applications. DSP chips are primary designed for real-time number crunching applications. They have dual(data and program) memories, sophisticated address generators, efficient external interfaces for I/O, along with powerful functional units such as the adder, barrel shifter, and a dedicated hardware multiplier, together with fast registers.? General-purpose microprocessors besides lacking a hardware multiplier and taking several tens of clock cycles to compute a single multiply, also lack the high memory bandwidth, low power dissipation, and real time I/O capabilities of DSP chips, and their cost advantages.

2.2.3 Architecture of Digital Signal Processor

Although fundamentally related, DSP processors are significantly different from general purpose processors (GPPs). To understand why, we need to know what is involved in signal processing. Some of the most common functions performed in the digital domain are signal filtering, convolution and fast Fourier transform. In mathematical terms, these functions perform a series of dot products. This brings us to the most popular operation in DSP: the multiply and accumulate (MAC).

The first major architectural modification that distinguished DSP processors from the early GPPs was the addition of specialized hardware that enabled single-cycle multiplication. DSP architects also added accumulator registers to hold the summation of several multiplication products. Accumulator registers are typically wider than other registers, often providing extra bits, called guard bits, to avoid overflow. Typical DSP algorithms require more memory bandwith than the Von Neumann architecture used in GPPs. Thus, most DSP processors use some forms of Hardvard architecture which has two separate memory spaces, typically partitioned as program and data memories.

Although, this may seem that DSP applications must pay careful attention to numeric accuracy - which is much easier to do with a floating-point data path, fixed-point machines tend to be cheaper (and faster) than comparable floating-point machines. To maintain accuracy without the complexity of a floating-point data path, DSP processors usually include, in both the instruction set and underlying hardware, good support for saturation arithmetic, rounding, and shifting.

Another distinction of DSP processors is specialized addressing modes that are useful for common signal-processing operations and algorithms. Examples include circular addressing (which is useful for implementing digital filter delay lines) and bit-reversed addressing (which is useful for performing a commonly used DSP algorithm, the fast Fourier transform). The generic architecture of a DSP processor is shown in Fig 2.3. The architecture has two separate memory spaces (program and data) which can be accessed simultaneously. This is similar to the Harvard architecture employed in most of the programmable DSP's. The arithmetic unit performs fixed point computation on numbers represented in "2's" complement form. It consists of a dedicated hardware multiplier and an adder/subtracter connected to the accumulator so as to be able to efficiently execute the multiply-accumulate (MAC) operation

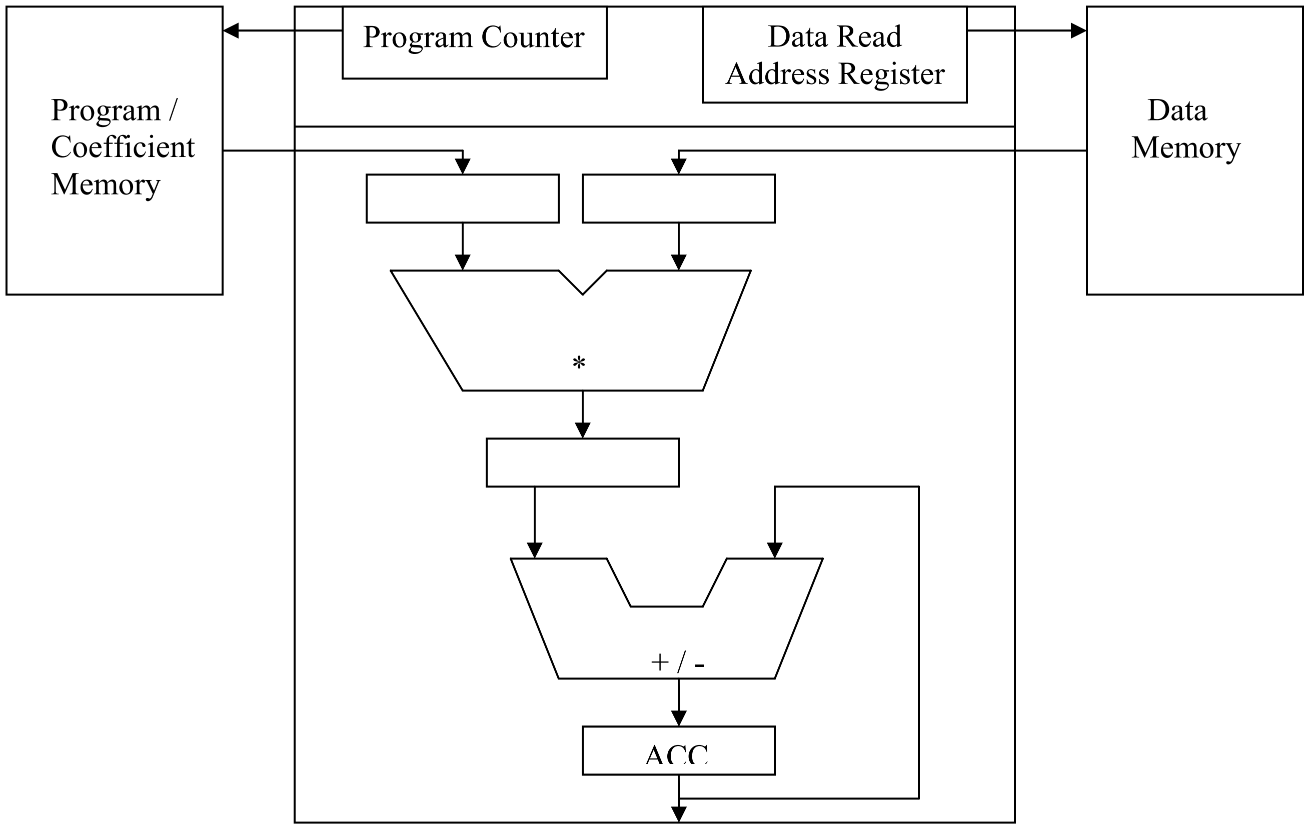


Fig 2.3 : Generic DSP Processor Architecture

2.3 DSP PROCESSOR DESIGN

All processors can be divided into two main categories: general-purpose processors and dedicated processors. General-purpose processors are capable of performing a variety of computations. In order to achieve this goal, each computation is not hardwired into the processor, but rather is represented by a sequence of instructions in the form of a program that is stored in the memory, and executed by the processor. The program in the memory can be easily changed so that another computation can be performed.

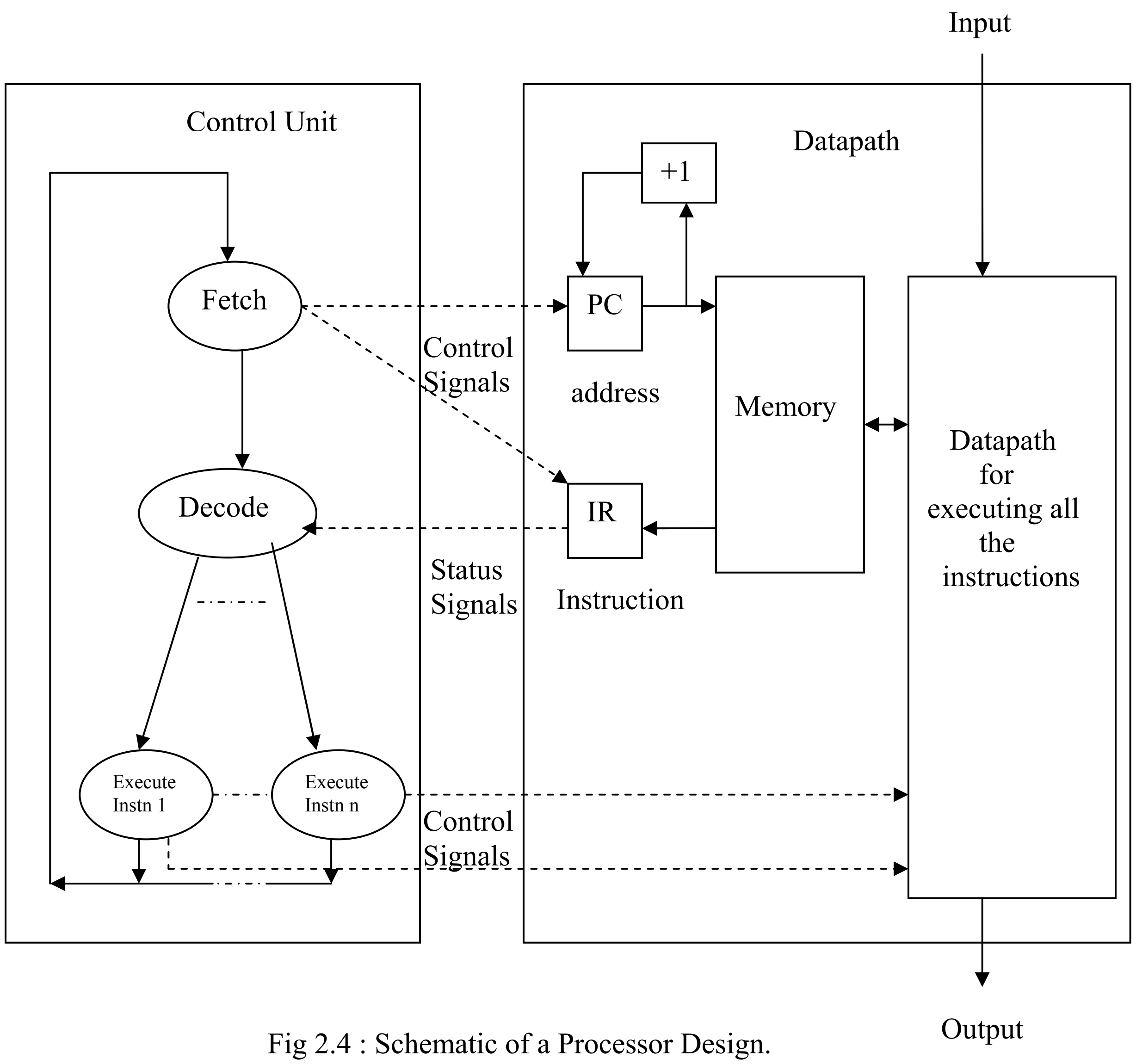


Fig 2.4 : Schematic of a Processor Design.

The design of a processor, can be divided into two main parts, the datapath and the control unit as shown in Fig 2.4. The datapath is responsible for all the operations perform on the data. It includes following

(1) Functional units such as adders, shifters, multipliers, ALU

(2) Registers and other memory elements for the temporary storage of data, and

(3) Buses and multiplexers for the transfer of data between the different components in the datapath.

External data can enter the datapath through the data input lines. Results from the computation can be returned through the data output lines.

The control unit (or controller) is responsible for controlling all the operations of the datapath by providing appropriate control signals to the datapath at the appropriate times. At any one time, the control unit is said to be in a certain state as determined by the content of the state memory. The state memory is simply a register with one or more (D) flip-flops. The control unit operates by transitioning from one state to another – one state per clock cycle, and because of this behavior, the control unit is also referred to as a finite-state machine (FSM). The next-state logic in the control unit will determine what state to go to next in the next clock cycle depending on the current state that the FSM is in, the control inputs, and the status signals. In every state, the output logic that is in the control unit generates all the appropriate control signals for controlling the datapath. The datapath, in return, provides status signals for the next-state logic. Status signals are usually from the output of comparators for testing branch conditions. Upon completion of the computation, the control output line is asserted to notify external devices that the value on the data output lines is valid.

In designing a processor, first its instruction set is to be defined, how the instructions are encoded and executed. The instruction set was designed using the best features of DSPs in combination with general CISC instructions. The combination of these instructions produces a versatile processor. Instructions were broken into six categories:

(1) ALU functions

(2) Special operation functions

(3) Branch functions
(4) Multiplication functions
(5) Shift functions
(6) Load functions.

Instructions are further broken down depending on number of operands, source type, and special functionality.

2.3.1 Instruction Format

Three instruction lengths were chosen: eight-bit, sixteen-bit and thirty-two-bit. The large number of instruction bits are useful in creating the data path though most instructions do not take advantage of the large number of bits. The first instruction byte is loaded into instruction register 1 (*IR1*) and broken into two further parts. The 3-bit opcode (*OP*= *IR1*[7..5]) determines which of the six categories the instruction is from. The five bit function (*func* = *IR1*[4..0]) specifies the specific instruction. Function coding is category dependent and certain bits of function represent either operand length or some other specific related to the instruction. The second instruction byte is loaded into instruction register 2 (*IR2*) and accessed by a number of aliases. These aliases include register and accumulator controls, immediate accessing, and the shift operation function (*shOp* = *IR2*[3..0]). Instruction bytes three and four form a sixteen-bit displacement used by certain instructions.

(1) ALU Functions

ALU functions are represented by an opcode of "000." They are broken into three sub-categories. Two operand instructions (specified when *func*(4) = '0') contain a source and destination registers. One-operand instructions (specified when *func*(4) = '1') contain only a destination register. Immediate instructions operate with the same functions as 2-operand instructions substituting a 6-bit immediate value for the source register. Single operand arithmetic instructions require a single destination register.

(2) Special Functions

Special functions (*OP* = "111") are zero-operand instructions including NOP, STOP, and RET. These three instructions do not affect data path registers. Six other special functions are used to set or clear the ALU flags. These instructions can be used to force certain branches. The functions of the special operations are Gray-coded. Bit zero is used for the carry flag, bit one for the negative and bit two for the zero flag. Bit four determines if the flag is set or cleared.

(3)Branch Functions

Branch functions (*OP* = "011") contain a 16-bit displacement to determine the location of a jump. Normal branch instructions determine whether to branch or not depending upon the values in the flags. The CALL instructions are used for jumping to a subroutine. Special branch instructions including branch-bit-set (BBS), branch-bit-clear (BBC), and hardware loop (RPT) are represented by *func*(4) = '1'.

(4)Multiply Functions

Four multiplication instructions (*OP* = "001") are included: two-operand multiplication, immediate multiplication, square, and multiply-accumulate (MAC) instructions. Gray-scale coding is used for multiplication functions. The MAC instruction format is "00000" which is the same function code for addition.

(5)Shift Functions

Shift functions (*OP* = "010") are the most complicated. Standard one-operand shifting occurs within the data path. Shift instructions include arithmetic (ASL & ASR), logical (LSR) and rotation (ROL &ROR). In addition rounding (RND) and truncating (TNK) are implemented. Rounding and truncating do not work in the normal mathematical sense. Rather round will keep the least significant bits and truncation will keep the most significant bits. In the case of single operand instructions, eight of the ten bits are kept while the remaining bits become zero.

Because of the location of the shifter in the data path it is possible to perform a shift operation and an ALU operation in a single clock cycle. This allows for single clock cycle squares and cubes of accumulator values or single clock cycle divide by two, divide by three. The type of ALU function is determined by *func* while *shOp* determines the shift function. For normal shifting, *func* = "11111" is used, since this passes input-a through the ALU.

Accumulator-to-memory transfers and accumulator-to-register transfers are included as shift instructions since the length of the accumulator needs to be limited in both cases. Accumulator-to-memory transfers include limit, round and truncate (LIMA, RNDA, TNKA). The limit instruction limits the accumulator according to a limitation algorithm that determines where the MSB is. Accumulator-to-memory transfers turn a ten bit value into an eight bit value. The same operations are supported as accumulator-to-register transfers (LIMF, RNDF, TNKF) which create a four bit value from the ten bit value.

(6) Load Functions

Four load instructions (*OP* = "100") exist. Load from memory (LD) loads a register with a number specified by a displacement. Load immediate (LDI) loads a register with a number specified in the instruction. Load direct (LDD) loads a register from a second register while load accumulator (LDA) loads an accumulator with a sign-extended register. Functions for load instructions are Gray coded.

2.3.2 Data path design

The data path is shown in the Fig 2.5. It contains a single data (8-bits) and a single address line(16-bits). The data path can be broken into a number of components. Four-bit values from memory are stored in a register file, which feeds into the multiplier. The ten-bit output of the multiplier connects with the ten-bit ALU / Accumulator file. A limiter is used for store instructions and accumulator-to-register operations. A shifter is along the ALU input-a feedback line. An addressing unit supplies the current address.

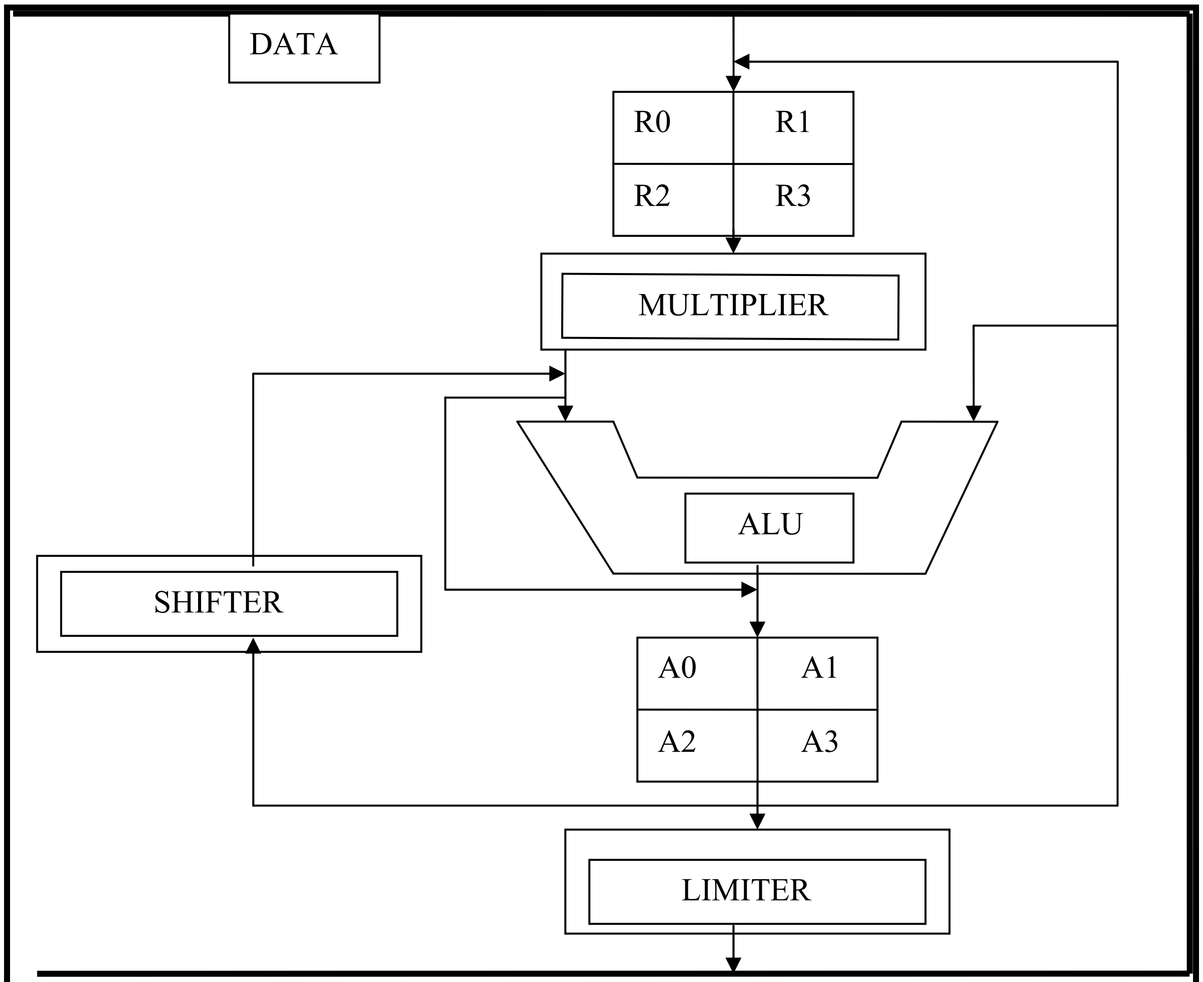


Fig 2.5 : Data Path Block Diagram of DSP Processor

2.3.3 Multiplier

The multiplier shown in Fig 2.6 is a simple unsigned multiplier that shifts and adds the inputs to produce a result. The input come from four 4-bit general purpose registers (R0-R1). The input to the register file is *rin*, *rin* can be a 4-bit data value, an immediate or a current register value. The register enable, *rgEn*, input allows control of storage to the register file. If *rgEn* is high, values can be stored. *RgEn* is controlled by current instruction and the state table output, start. Inputs to the multiplier are multiplexed according the *ra* and *rb*.

The multiplier contains an internal state machine that counts the number of bits shifted and added. It is controlled by a start signal and outputs a finish signal when the product has been calculated. Two four-bit numbers multiplied together will take six clock cycles. For the square instruction *rb* is set equal to *ra*, and for the immediate multiply it is multiplexes an immediate value.

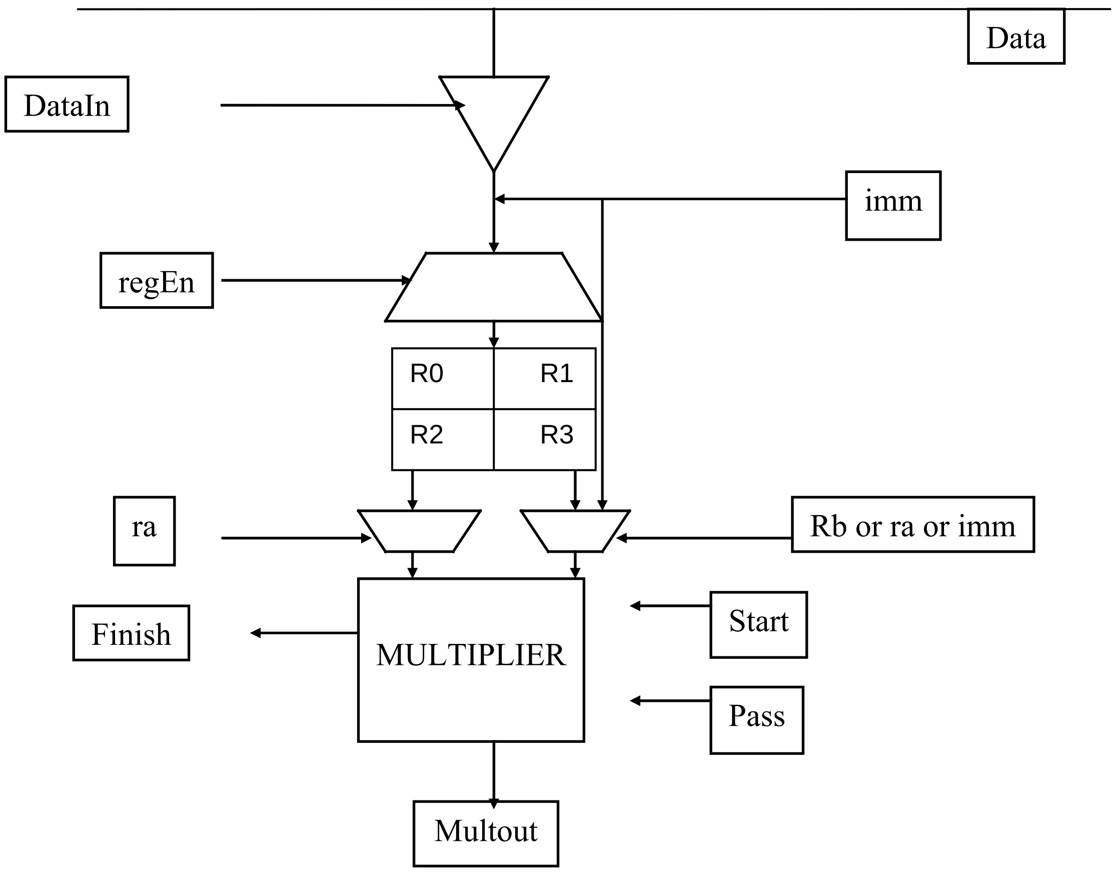


Fig 2.6 : Multiplier Block Diagram of DSP Processor

2.3.4 ALU / Accumulator

ALU is shown in Fig 2.7. Inputs to the accumulator include *ALUI1* and *ALUI2*. *ALUI1* is the output of the shifter unless the instruction register holds the MAC instruction in which case the multiplier output is fed onto *ALUI1*. *ALUI2* is multiplexed by *aca* or *acb* or it multiplexes an immediate value into the ALU. The ALU operation is chosen by *fn* and the ALU outputs four flags dependent on the current instruction: carry, zero, negative, and overflow. The output of the ALU is multiplexed with the multiplier output depending on the instruction and then loaded into the accumulator file.The four 10-bit accumulators are loaded based on the register enable, *regEn*, and *aca*. *RegEn* depends upon *start* and the current instruction.

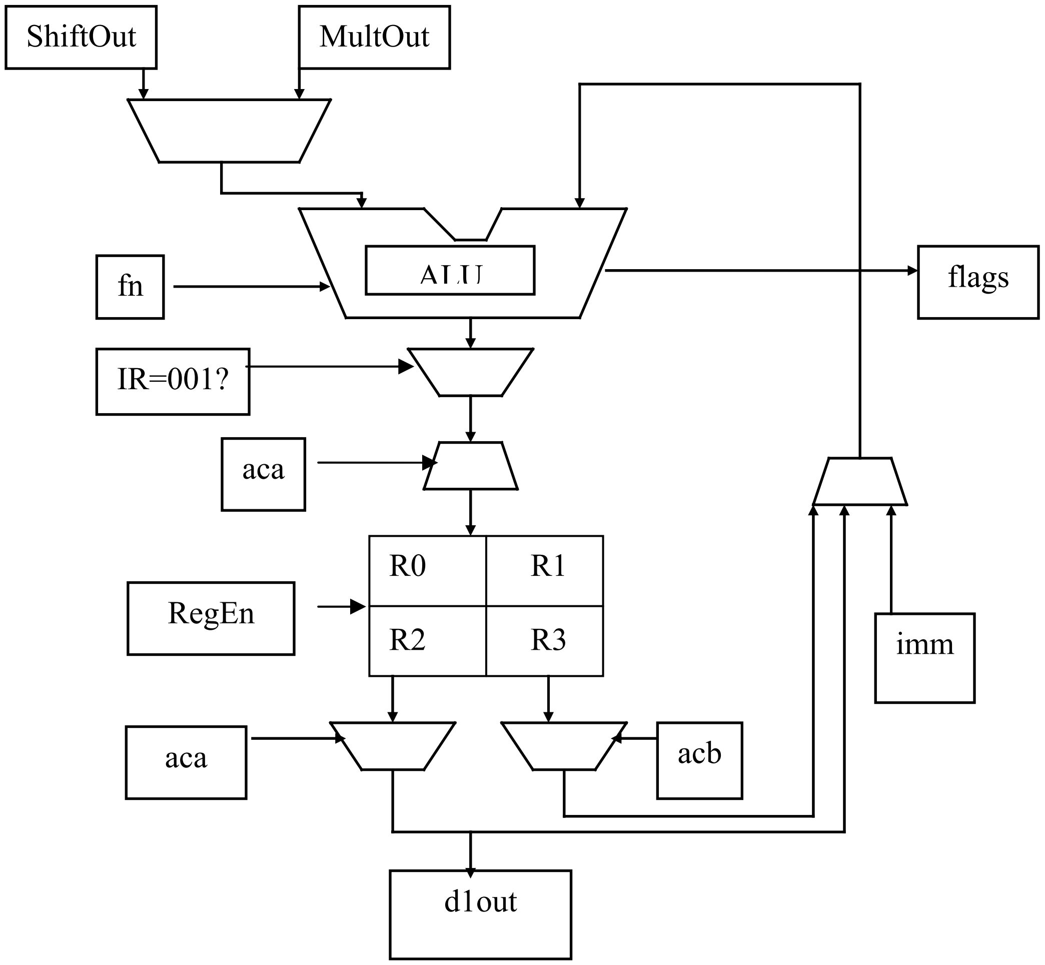


Fig 2.7: ALU Block Diagram of DSP Processor

2.3.5 Limiter

The limiter limits the *aca* accumulator's output dependent upon the function. The function is loaded from *shOp*. The limiter outputs either an eight-bit value to the data bus or a four-bit value to a register. The limiter also outputs a four-bit flag that contains the bits trimmed from the input. Future instructions can use this flag for control purposes.

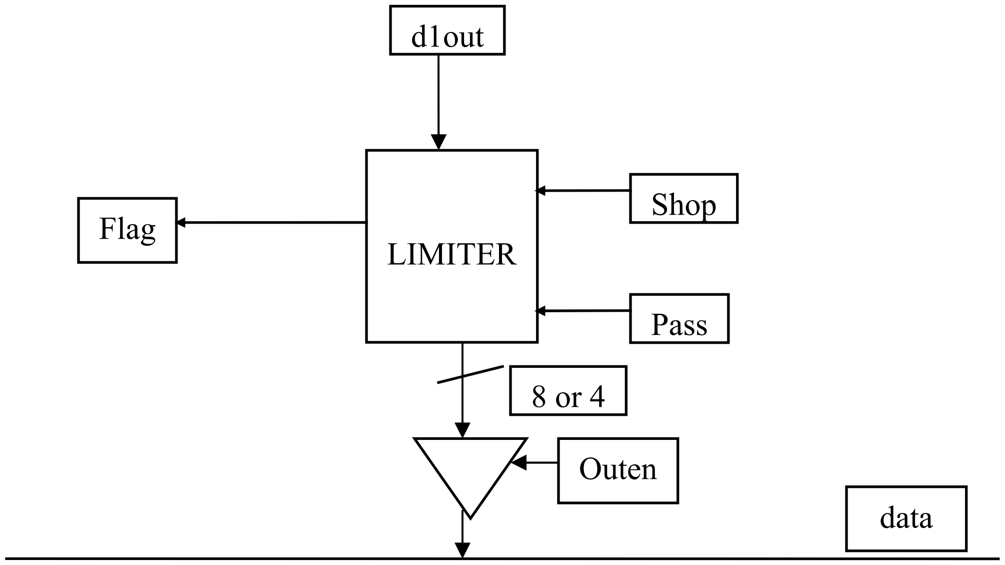


Fig 2.8 : Limiter Block Diagram of DSP Processor

2.3.6 Shifter

The shifter accepts a 10-bit input and shifts it according to *shOp*. The shifter outputs a ten-bit value to the ALU and a single-bit flag, which contains the shifted bit. Similar to the limiter, the flag may be used in future instructions.

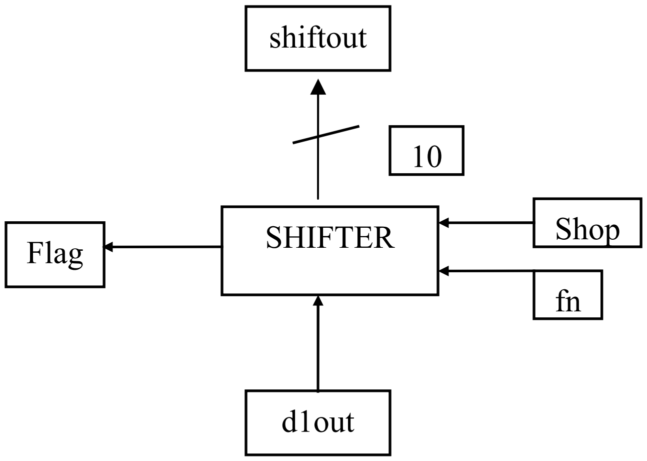


Fig 2. 9: Shifter Block Diagram of DSP Processor

2.3.7 Address Unit

The addressing unit contains a 16-bit program counter, 16-bit stack pointer, 8-bit memory address high register and 8-bit memory address low register. Depending upon the state and instruction the address line receives one of the three.

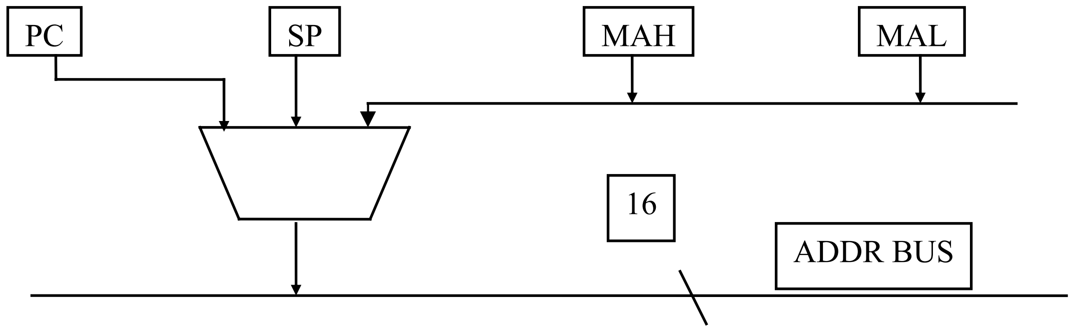


Fig 2.10: Address Unit of DSP Processor

2.3.8 State Machine

The state machine shown in Fig 2.11, is reset by the *reset* input. During reset, state is set to 0 and PC is set to FFFF to allow the next clock tick to bring it to 0000. At each clock tick the state is loaded with the next state and the PC is incremented. The first state loads the first instruction byte and determines if a second byte is needed, the second state loads the second byte and determines whether a displacement is present. States three and four load the displacement. State five sets the start signal and waits for a finish signal. Finish is either set by the multiplier when the product is ready. In the case of an ALU instruction, finish is set the following clock cycle. The machine then jumps to state one and the process repeats.

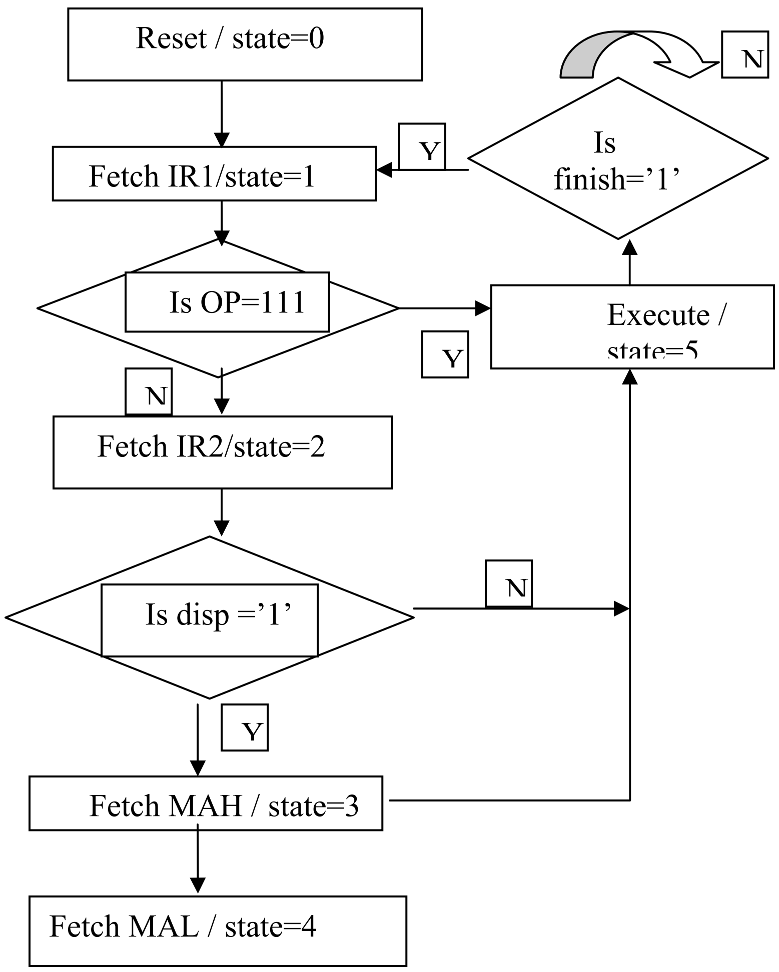


Fig 2.11 : State Machine Block Diagram of DSP Processor

Chapter 3

FIR FILTER

3.1 INTRODUCTION TO DIGITAL FILTERS

3.1.1 Analog and digital filters

In signal processing, the function of a *filter* is to remove unwanted parts of the signal, such as random noise, or to extract useful parts of the signal, such as the components lying within a certain frequency range. The following block diagram illustrates the basic idea.

Raw (Unfiltered) Signal
FILTER
Filtered Signal

Fig 3.1 : Basic Idea of a Filter

There are two main kinds of filter, analog and digital. They are quite different in their physical makeup and in how they work. An analog filter uses analog electronic circuits made up from components such as resistors, capacitors and op-amps to produce the required filtering effect. Such filter circuits are widely used in such applications as noise reduction, video signal enhancement, graphic equalizers in hi-fi systems, and many other areas. There are well-established standard techniques for designing an analog filter circuit for a given requirement. At all stages, the signal being filtered is an electrical voltage or current which is the direct analogue of the physical quantity (e.g. a sound or video signal or transducer output) involved.

A digital filter uses a digital processor to perform numerical calculations on sampled values of the signal. The processor may be a general-purpose computer such as a PC, or a specialized DSP (Digital Signal Processor) chip. The analog input signal must first be sampled and digitized using an ADC (analog to digital converter). The resulting binary numbers, representing successive sampled values of the input signal, are transferred to the processor, which carries out numerical calculations on them. These calculations typically involve multiplying the input values by constants and adding the products together. If necessary, the results of these calculations, which now represent sampled values of the filtered signal, are output through a DAC (digital to analog converter) to convert the signal back to analog form. In a digital filter, the signal is represented by a sequence of numbers, rather than a voltage or current. The following diagram shows the basic setup of such a system.

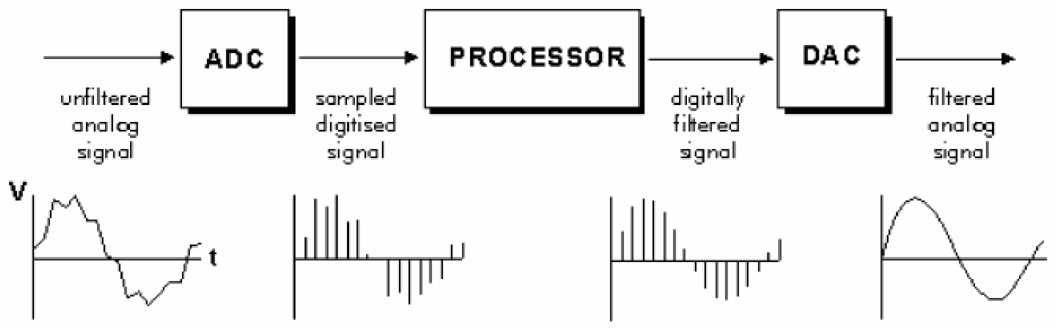


Fig 3.2 : Basic Set-Up of a Digital Filter

3.1.2 Advantages of using digital filters

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

1. A digital filter is *programmable*, i.e. its operation is determined by a program stored in the processor's memory. This means the digital filter can easily be changed without affecting the circuitry (hardware). An analog filter can only be changed by redesigning the filter circuit.
2. Digital filters are easily *designed*, *tested* and *implemented* on a general-purpose computer or workstation.
3. 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 both to time and temperature.
4. 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 RF (radio frequency) domain, which in the past was the exclusive preserve of analog technology.

5. 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.
6. 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.

3.2 FIR FILTERS

Finite impulse response (FIR) filtering is one of the most commonly used DSP functions. It is achieved by convolving the input data samples with the desired unit impulse response of the filter. The output Y(n) of an N –tap FIR filter is given by the weighted sum of latest N input data samples-

$$Y[n] = \sum_{i=0}^{N} A[i] \, . \, X[n-i]$$

The weights A[i] in the expression are the filter coefficients. The number of taps (N) and the coefficient values are derived so as to satisfy the desired filter response in terms of pass-band ripple and stop-band attenuation. FIR filters with symmetric coefficients A[i] = A[N-1-i] have a linear phase response (0 group delay) and are hence an ideal choice for applications requiring minimal phase distortion. A general FIR filter architecture is shown in Fig 3.3.

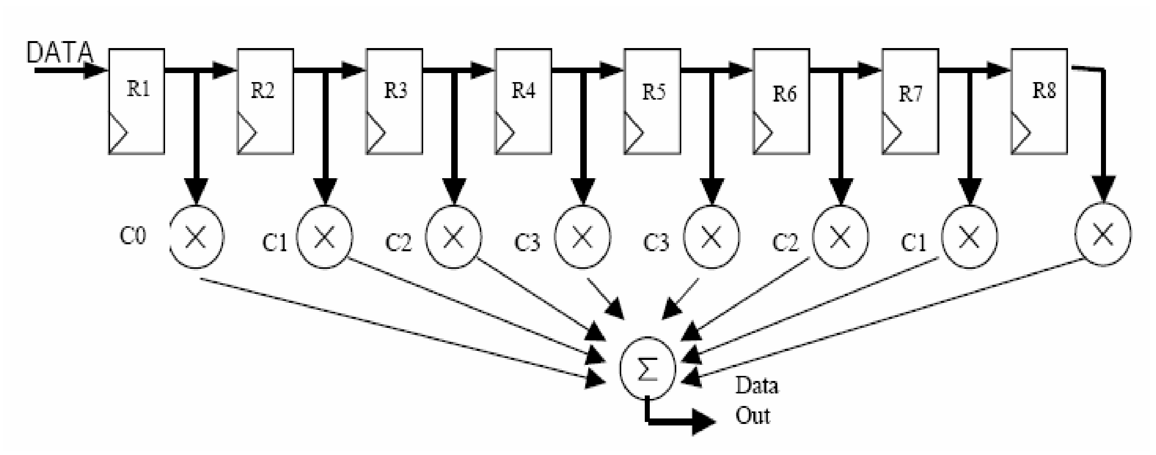


Fig 3.3 : FIR Filter Architecture

The input is shifted through 8 registers (taps). Each output stage of a particular register is multiplied by a known coefficient. The resulting outputs of the multipliers are then summed to create the filter output. Note that the coefficients are symmetric about the center taps. This is true for a linear phase response FIR filter as stated above. So this will allow us to “fold” (folding is an operation done by replacing the independent variable n by –n. it is the reflection of the signal about the time origin n=0. This is done while convolving the signal with another.) the filter in half as shown in Fig 3.4 thus reducing the amount of multipliers to half.

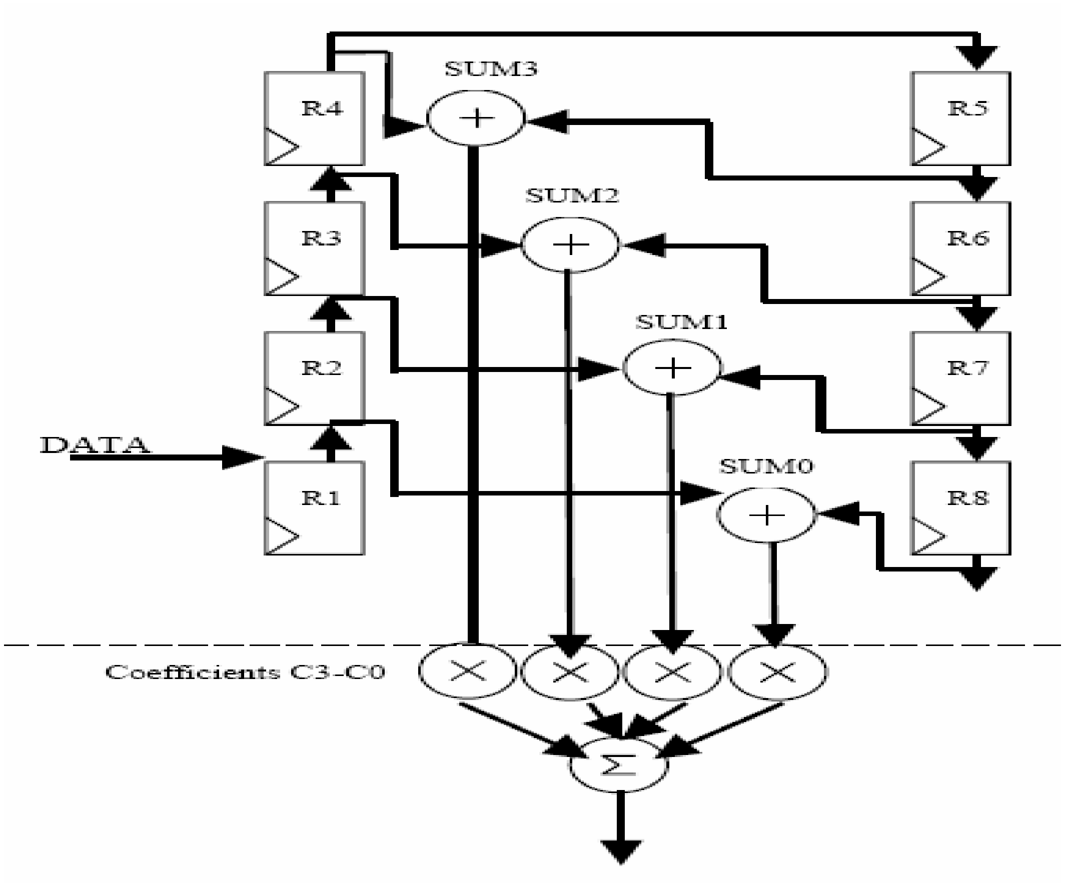


Fig 3.4 Folded FIR Filter Architecture

Chapter 4

DUAL EDGE TRIGGERED GLITCH REDUCING CLOCKING STRATEGY

4.1 CLOCK SUB-SYSTEM

Proper timing and clock system design are one of the most critical components in digital system. The function of the clock in the digital system can be compared to the function of metronome in the music. It designates a beginning of a music score, exact moment when certain notes are to be played by particular instruments in orchestra, designates the end of the part or section, i.e. provides synchronization for various instruments in the orchestra during various parts and periods of the score that is being performed. Similarly, in digital system the clock designates the exact moment when the signal is to change as well as its final value is to be captured, when the logic is active or inactive. Finally, all the logic operations have to finish before the tick of the clock and the final values of the signals are being captured at the tick of the clock. Therefore, the clock provides the time reference point which determines the movement of data in the digital system.

4.1.1 Clock Signals

Clocks are defined as pulsed, synchronizing signals that provide the time reference for the movement of data in the synchronous digital system. The clocking in a digital system can be either single phase, multi-phase (usually two-phase) or edge-triggered, as shown in Fig 4.1:

The dark rectangles in the figure represent the interval during which the bi-stable element samples its data input. Figure shows the possible types of clocking techniques and corresponding general finite-state machine structures:

a) Single-phase clocking and single-phase latch machine, Fig. 4.1 (a)

b) Edge-triggered clocking and flip-flop machine, Fig.4.1 (b)

c) Two-phase clocking and two-phase latch machine with single latch Fig.4.1 (c)

d) Two-phase clocking and two-phase latch machine with double latch Fig. 4.1 (d)

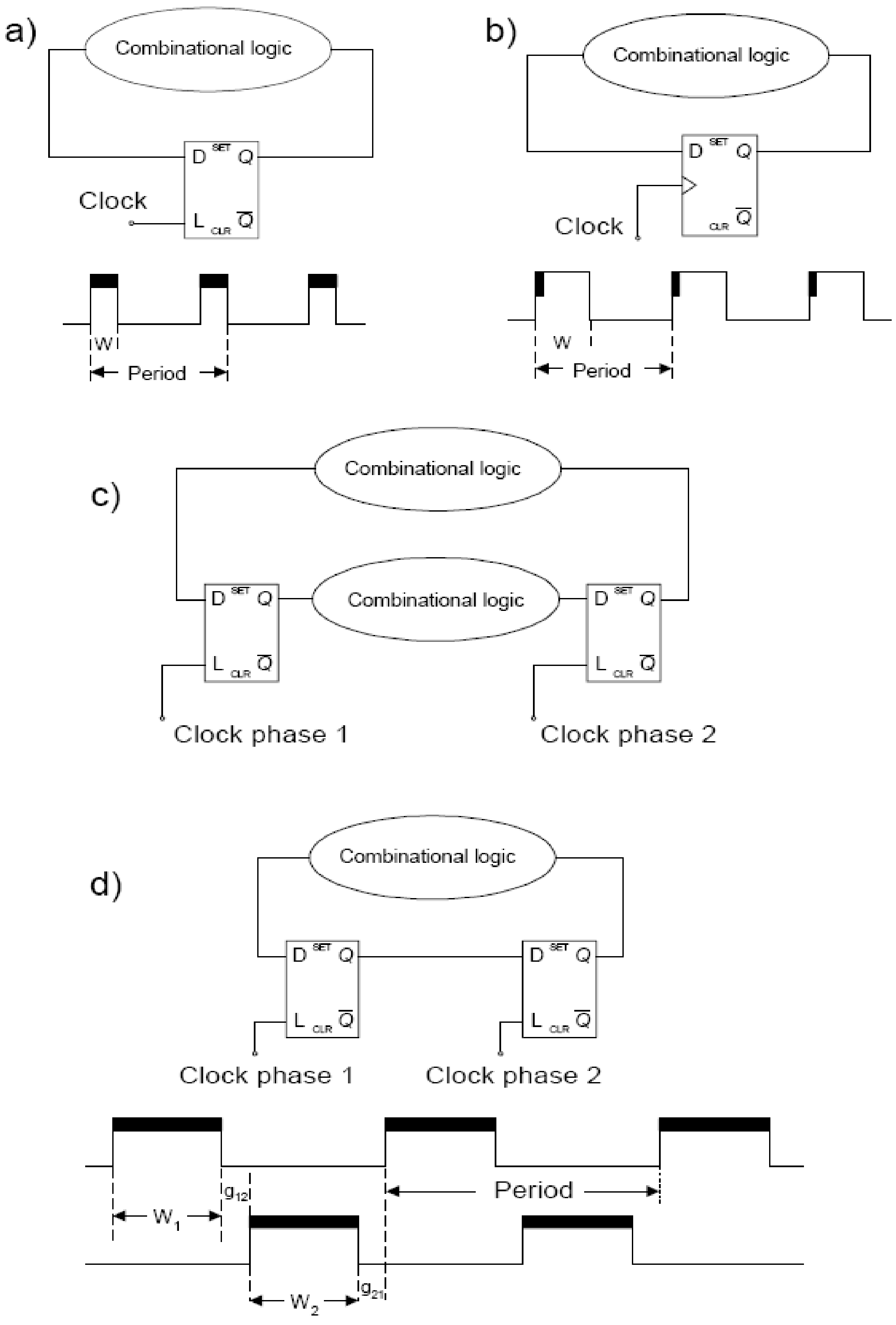


Fig 4.1 : System Clocking Waveforms and General Finite-State Machines Structures

In Fig.4.1 (c) and (d), W_j is the pulse width of the phase j and g_{ij} is the inter-phase gap from phase i to phase j; if $g_{ij} > 0 \Rightarrow$ two-phase, non-overlapping, if $g_{ij} < 0 \Rightarrow$ two-phase, overlapping clocking scheme. The multi-phase design typically extends to three, but not more than four non-overlapping phases. Two non-overlapping clocks provide most reliable and robust clocking system that fits well into the design for testability methodology.

4.1.2 Clock Distribution

The two most important timing parameters affecting the clock signal are:

1) Clock Skew
2) Clock Jitter

Clock Skew is a spatial variation of the clock signal as distributed through the system. It is caused by the various RC characteristics of the clock paths to the various points in the system, as well as different loading of the clock signal at different points on the chip. Clock Jitter is a temporal variation of the clock signal with regard to the reference transition (reference edge) of the clock signal as illustrated in Figure 4.2. Clock jitter represents edge-to-edge variation of the clock signal in time. As such clock jitter can also be classified as: long-term jitter and edge-to-edge clock jitter, which defines clock signal variation between two consecutive clock edges. In the course of high speed logic design we are more concerned about edge-to-edge clock jitter because it is this phenomena that affects the time available to the logic. Typically the clock signal has to be distributed to several hundreds of thousands of the clocked storage elements (also known as flip-flops and latches). levels of amplification (buffering). As a consequence, the clock system by itself can therefore, the clock signal has the largest fan-out of any node in the design, which requires several use up to 40-50% of the power of the entire VLSI chip. Also it must be assured that every clocked storage element receives the clock signal precisely at the same moment in time.

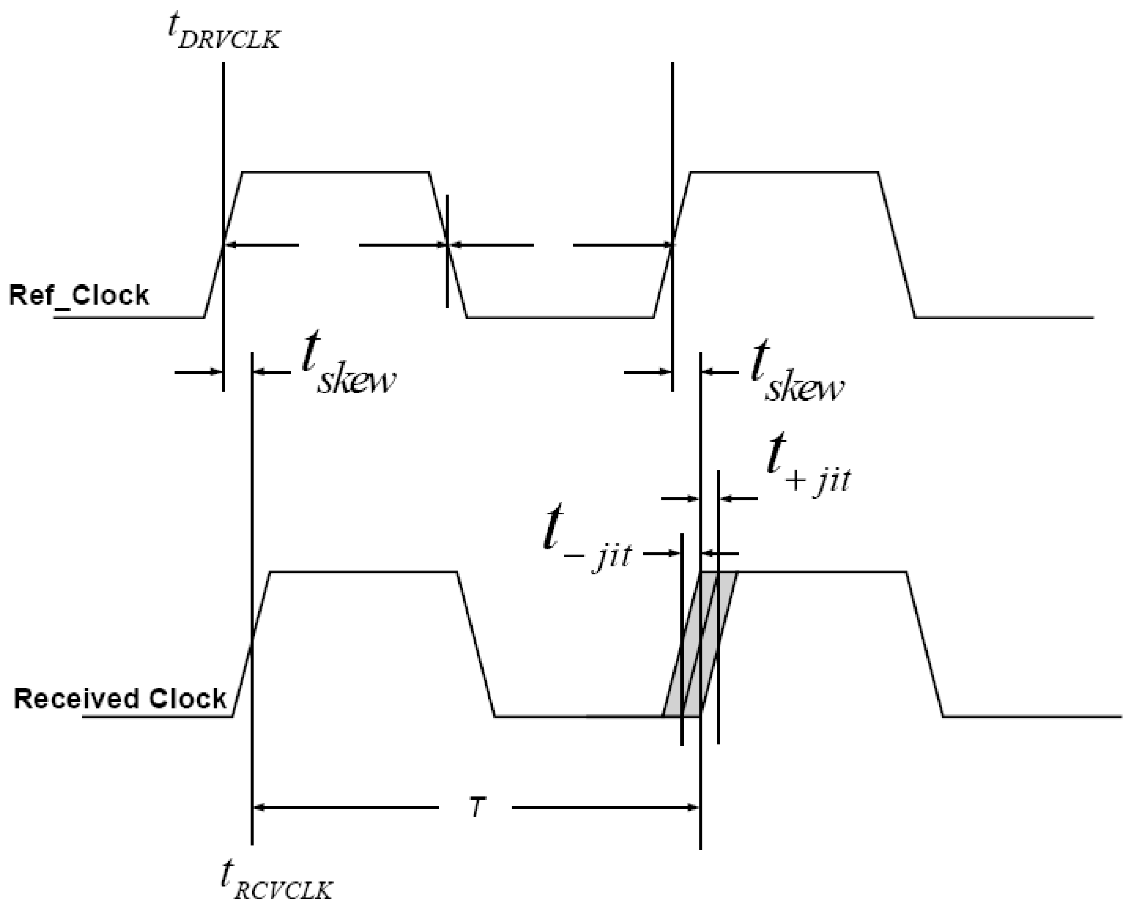


Fig 4.2 : Clock Parameters: Period, Width, Clock Skew and Clock Jitter

4.2 BI-STABLE ELEMENTS

In order to define the clocking of the system properly, the nature and behavior of the bistable elements, often referred as a “latch” or “Flip-Flop” needs to be specified precisely. It is quite common to find both terms “Flip-Flop” and a “Latch” to be used indiscriminately for the bi-stable element used in the synchronous systems. In a synchronous system, operations and data sequences take place with a fixed and pre-determined time relationship. The timing of computations are controlled by flip-flops and latches together with a global clock. Flip-flops and latches are clocked storage elements, which store values applied to their inputs. They are classed according to their behavior during the clock phases. A latch is level sensitive. It is transparent and propagates its input to the output during one clock phase (clock low or high), while holding its value during the other clock phase. A flip-flop is edge triggered. It captures its input and propagates it to the output at a clock edge (rising or falling), while keeps the output constant at any other time. The design of these clocked storage elements is highly depended on the clocking strategy and circuit topology.

Latch

It is a device capable of storing the value of the input D in conjunction with the clock C and providing it at its output Q. The latch has a following relationship between the input D and the clock C: While C=0 the output Q remains constant regardless of the value of D (the latch is "opaque"). While C=1 the output Q=D and it reflects all the changes of D(the latch is *"transparent"*). Often we describe such a behavior of a latch as *"levelsensitive"* (the behavior of the latch is dependent on the value of the clock - not the changes). Behavior of an ideal Latch is illustrated in Fig 4.3.

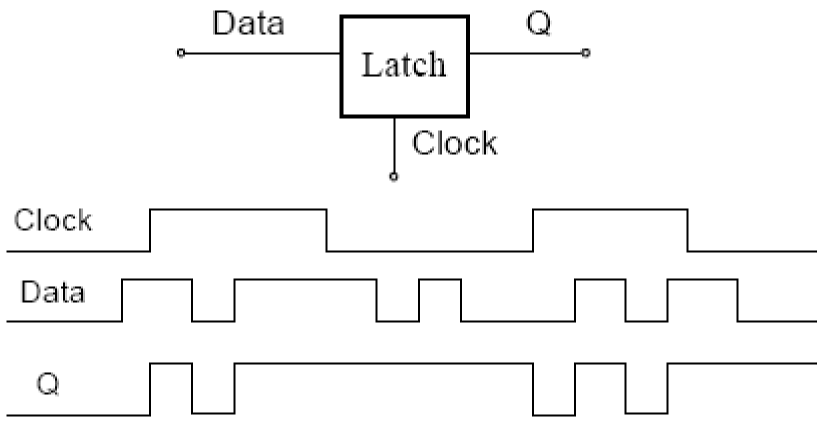


Fig 4.3 : Signal Relationship of an ideal Latch.

Flip-Flop

It is defined as a bi-stable memory element with the same inputs and outputs as a "latch", Figure. However, the output Q responds to the changes of D only at the moment the clock C is making transitions. We define this as being *"edge triggered"*. The internal mechanisms of the "flip-flop" and that of a "latch" are entirely different. We further define a "Flip-Flop" as a *"leading edge triggered"* if the output Q assumes a value of the input D as a result of the transition of the clock C from 0-to-1. Conversely in a *"negative* *edge triggered”* Flip-Flop the output Q assumes a value of the input D as a result of the transition of the clock C from 1-to-0. It is also possible to build *a “double-edge triggered”* flip-flop that responds to both: *leading* and *trailing* edge of the clock C.

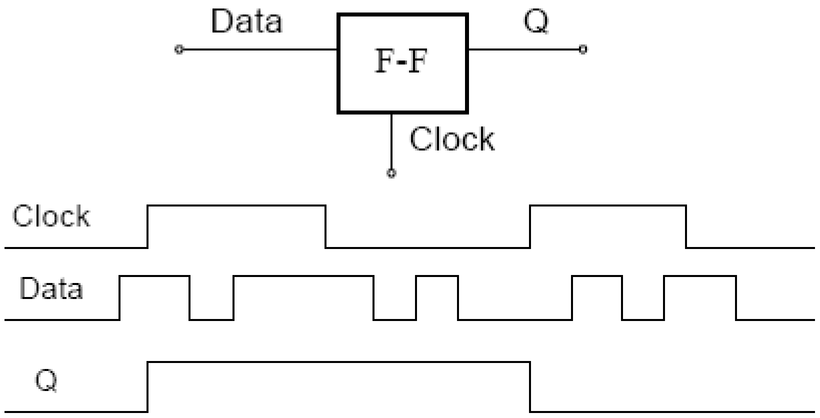


Fig 4.4 : Signal Relationship of an Ideal Leading-Edge Triggered Flip-Flop

4.2.1 Single Edge Triggered Flip Flop and Dual Edge Triggered Flip Flop

When the two latches are connected in series as shown in Fig 4.5 then the flip-flop results is called single edge triggered flip flop. The data are loaded at only one clock edge, either rising or falling.

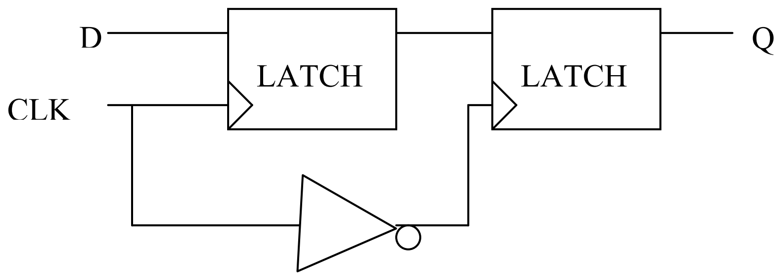


Fig 4.5 : Single Edge Triggered Flip-Flop

And when the two latches are connected in parallel as shown in Fig 4.6 the flip-flop results is called dual edge triggered flip flop. The data are loaded at both rising and falling clock edge. A dual edge triggered flip-flop requires slightly more transistors to implement.

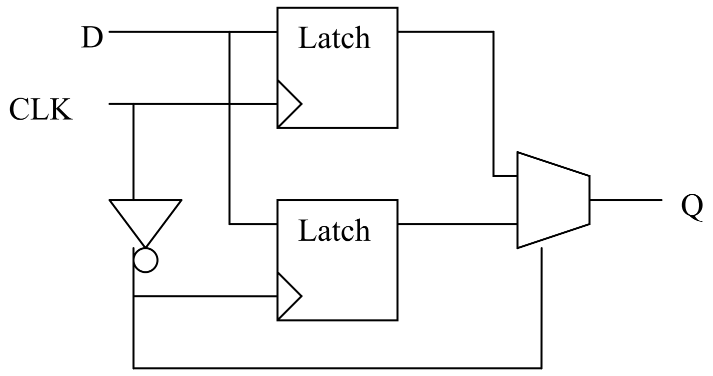


Fig 4.6 : Dual Edge Triggered Flip-Flop

4.3 CLOCK GATING

Low-power techniques are essential in modern VLSI design due to the continuous increase of clock frequency and chip complexity. In particular, the clock system, composed by flip-flops and clock distribution network, is one of the most power consuming subsystem in a VLSI circuit. The three major components of power consumption:

1) Power consumed by combinational logic whose values are changing on each clock edge.
2) Power consumed by flip-flops (this has a non-zero value even if the inputs to the flip-flops, and therefore, the internal state of the flip-flops, is not changing)
3) The power consumed by the clock buffer tree in the design.

As a consequence many techniques have been proposed to reduce clock system power dissipation. Disabling the clock signal (clock gating) in inactive portions of the chip is a useful approach for power dissipation reduction. Clock gating can be applied to different hierarchical levels. It is possible to disable the clock signal that drive a big functional unit reducing power dissipation on both its internal nodes and its clock line. Automatic clock gating is supported by modern EDA tools. They identify the circuits where clock gating can be inserted.

Clock gating works by identifying groups of flip-flops which share a common enable control signal. Traditional methodologies use this enable term to control the select on a multiplexer connected to the D port of the flip-flop or to control the clock enable pin on a flip-flop with clock enable capabilities. Clock gating uses this enable signal to control a gating circuit which is connected to the clock ports of all of the flip-flops with the common enable term. Therefore, if a bank of flip-flops which share a common enable term have clock gating implemented, the flip-flops will consume zero dynamic power as long as this enable signal is false.There are two types of clock gating styles available. They are:

1) Latch free clock gating
2) Latch based clock gating

4.3.1 Latch free clock gating

The latch-free clock gating style as shown in Fig 4.7, uses a simple AND or OR gate (depending on the edge on which flip-flops are triggered). Here if enable signal goes inactive in between the clock pulse or if it multiple times then gated clock output either can terminate prematurely or generate multiple clock pulses. This restriction makes the latch-free clock gating style inappropriate for our single-clock flip-flop based design.

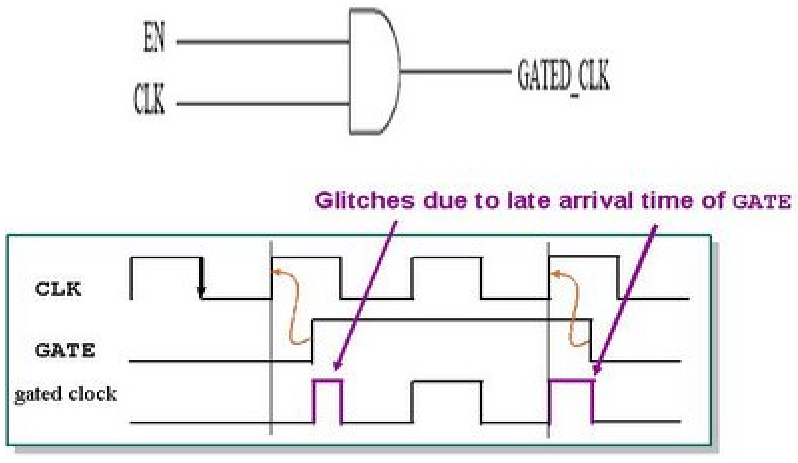


Fig 4.7 : Latch Free Clock Gating

4.3.2 Latch based clock gating

The latch-based clock gating style as shown in Fig 4.8, adds a level-sensitive latch to the design to hold the enable signal from the active edge of the clock until the inactive edge of the clock. Since the latch captures the state of the enable signal and holds it until the complete clock pulse has been generated, the enable signal need only be stable around the rising edge of the clock, just as in the traditional ungated design style.

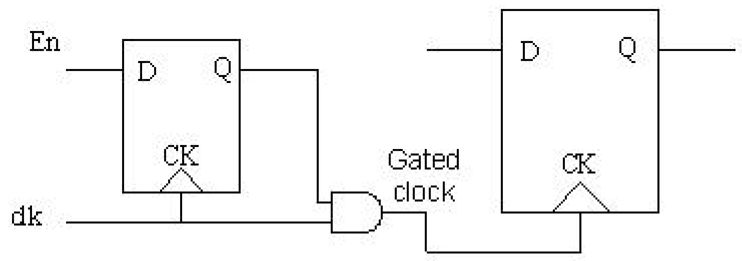


Fig 4.8 : Latch Based Clock

4.3.3 Clocking Schemes

Single-edge-triggered (SET) one-phase clocking is certainly the most common choice. There are five good reasons for this

1) Automata theory maps to SET operation directly

2) It is supported by all common design tools

3) It requires the routing of only one clock tree

4) The basic bistable (SET-FF) is relatively simple,

5) A scan chain is easy to implement for testing

SET clocking has important drawbacks in terms of energy efficiency, however. This clocking scheme has found to have following three drawbacks:

1) Superfluous switching whenever a register is disabled,
2) The presence of an inactive clock edge,

3) The strong vulnerability to clock skew.

The first limitation is commonly addressed with clock gating. Also an idea of dual edge triggered clocking is proposed to overcome the second limitation. Also with clock period reduced to 200 ps nowadays, the importance of clock uncertainties will increase, and complex multiple-phase clocking will become impractical due to increasingly large timing uncertainties and power consumption.

In order to save on clocking power, dual-edge triggered (DET) clocking strategy uses DET storage elements (DETSE) that capture the value of the input after both rising and falling clock transitions. Otherwise, DETSE is nontransparent, i.e., it holds the captured value at the output. Thus, the DET clocking strategy provides a one-time solution to the frequency scaling by retaining the data throughput of single-edge triggered (SET) clocking at halved clock frequency. However, in order to fully exploit the power savings in the clock distribution network, this approach must ensure that the delay and energy consumption of DETSE must be comparable to those of SET storage elements (SETSE). Furthermore, use of both clock edges to synchronize the operation makes timing sensitive to clock duty cycle and increases clock uncertainties generated by the clock distribution system.

4.4 POWER REDUCTION USING DUAL EDGE TRIGGERED CLOCKING STRATEGY

As discussed earlier, clock related power is one of the most significant components of the dynamic power consumption. The total clock related power dissipation in synchronous VLSI circuits is further divided into three major components. The total power dissipation of the clock network depends on both the clock frequency and the data rate, and can be computed based on equation

$$P_{CK} = V^2dd[f_{CK}(C_{CK} + C_{ff,CK}) + f_D C_{ff,D}] \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Eq(4.1)}$$

where

f_{CK} is the clock frequency;

f_D is the average data rate;

C_{CK} is the total capacitance seen by the clock network;

$C_{ff,CK}$ is the capacitance of the clock path seen by the flip-flop;

$C_{ff,D}$ is the capacitance of the data path seen by the flip-flop.

From Eq(4.1) it is obvious that the clock power can be reduced if any of the parameters on the right hand side of the equation is reduced. The reduction of Vdd is already the trend of contemporary design, and it has the strongest impact on the P_{CK} expression. By reducing the overall capacitance of the clock network, C_{CK}, the power dissipation may also be reduced. For instance, the capacitance can be reduced by proper design of clock drivers and buffers. Similarly, by reducing the capacitance inside a flip-flop, $C_{ff,CK}$ and $C_{ff,D}$, power may also be reduced. Furthermore, the clock power dissipation is linearly proportional to the clock frequency. Although the clock frequency is determined by the system specifications, it can be reduced with the use of dual edge triggered flip-flops (DETFFs). As its name implied, DETFF responds to both rising and falling clock edges. Hence, it can reduce the clock frequency by half while keeping the same data throughput. As a result, power consumption of the clock distribution network is reduced, making DETFFs desirable for low power applications. Even for high performance applications, the usage of DETFFs offers certain benefits. Since the clock speed is reduced by a factor of two, one does not need to propagate a relatively high speed clock signal.

4.5 PROPOSED GLITCH REDUCING CLOCKING TECHNIQUE

Glitches are responsible for a significant proportion of overall power dissipation in digital signal processing circuits. Activity-reduction techniques that involve an optimized clocking strategy have been applied to a front-end block in a DSP- adaptive directional microphone for hearing aids.

As a consequence from the third limitation of SET clocking scheme, its operation necessitates careful balancing of clock distribution delays and fast clock ramps. These two requirements are typically met with multiple clock buffers of generous size and power. Level-sensitive two phase clocking does away with the need for fast ramps as the non-overlap phases provide welcome skew margins. Yet the extra load of a second clock tree reduces the benefits. Also the clock distribution never absorbs more than 30%-40% of a chip's overall power dissipation. Even if clock power could be cut in half using DET clocking scheme, this would translate into a mere 15% to 20% overall savings. So a symmetric Level sensitive two phase clocking is undertaken to mitigate spurious switching activities in the data-path too.

This achievement has been made possible by combining two novel techniques:

1) A multi-stage clock gating
2) A symmetric two-phase level sensitive clocking with glitch-aware re-distribution of data-path registers.

The starting point is a symmetric two-phase level sensitive clocking which is again enhanced by two features:

1) Multi-Stage clock gating
2) Stop-Glitch latch barriers

4.5.1 Multi-Stage Clock Gating

Multi-stage clock gating implies that multiple clock gates are cascaded hierarchically to minimize the useless toggling of heavily loaded nodes. While this technique is already sum supported in well-known synthesis tools such as Synopsys for SET clocking, it has never been addressed in the context of two-phase clocking. Non-overlap phases provide ample skew margins and make multi-stage clock gating particularly easy to combine with level-sensitive two-phase clocking. This contrasts favorably with edge-triggered designs, where meager hold margins make it difficult to avoid timing violations.

This multi stage clock gating has been applied to the FIR filters. Accessing a memory in a circular fashion typically involves a counter and an address decoder. If combined with level sensitive two phase clocking and clock gating, a multitude of AND gates result as shown in Fig 4.9 which altogether, are responsible for a heavy capacitive load on the clock net. By re-arranging the decoder in a hierarchical fashion can be significantly lowered. The re-arranged decoder is shown in Fig 4.10.

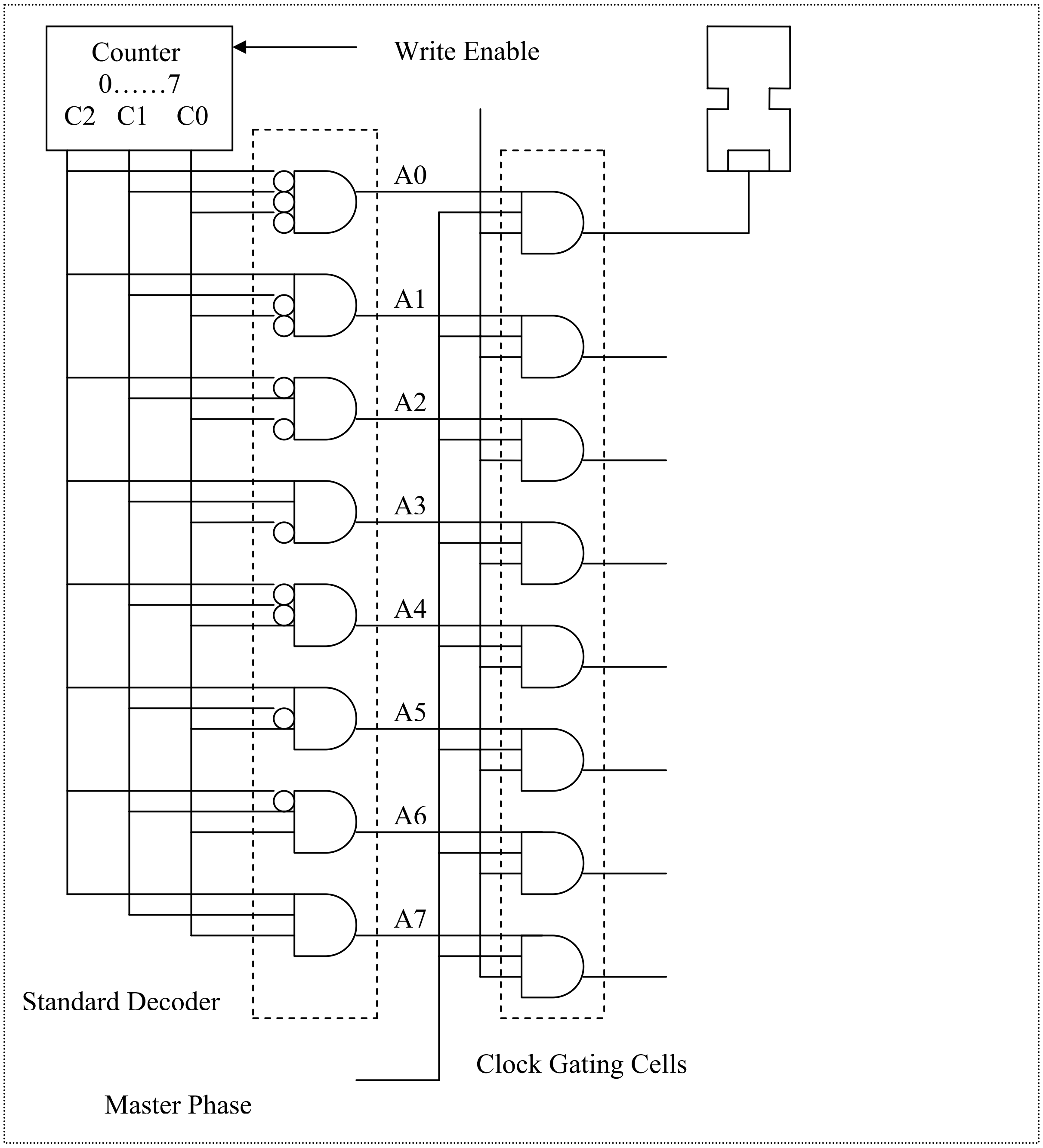


Fig 4.9 : Traditional Clock Gating Applied to a Register File

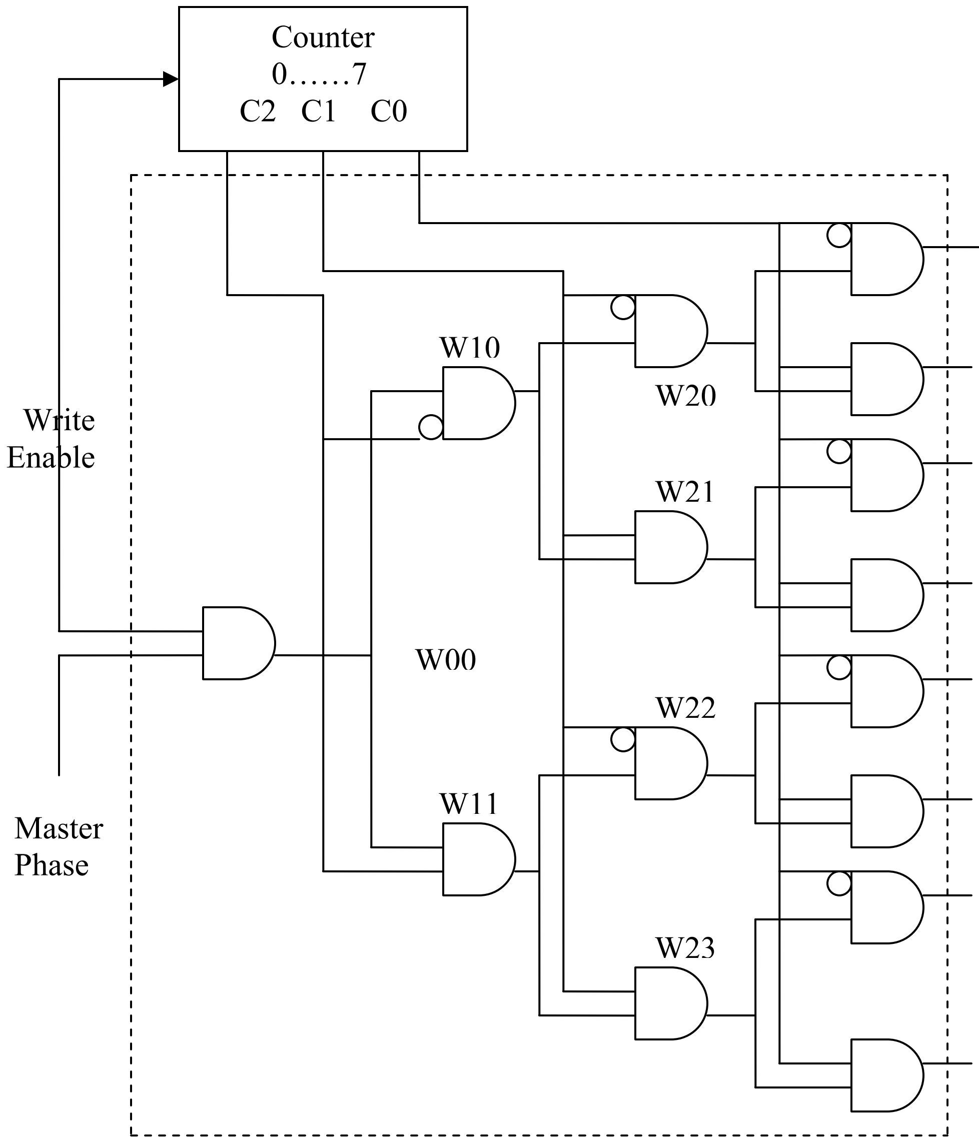


Fig 4.10: Multi Stage Clock Gating Decoder

The relevant quantity is the activity capacitance product summed up over all circuit nodes driven from the clock. This sum of products will be referred to as "effective capacitance". For a traditional address decoder the effective capacitance can be calculated from node switching activities(α).

Given a memory of size n (a power of two), and defining k as $\log_2 n$, the switching activities of the counter outputs c_0…………c_{k-1} are:

$\alpha_{c(0)} = 1 \quad \alpha_{c(1)} = ½ \quad \ldots\ldots\ldots \quad \alpha_{c(k-1)} = 1/2^{k-1}$ ……… Eq(4.2)

The signals q0……qn-1 toggle only twice during a whole writing cycle, therefore:

$\alpha_{qj} = 2/n = 1/2^{k-1} \qquad j = 0, \ldots\ldots, n-1$ …………Eq(4.3)

The input gate capacitance is C_0 and the switching activity of the master phase a,m is two by definition. The effective capacitance of the traditional decoder (C_1) can be expressed through proper combination of Eq.4.2 and Eq.4.3:

$$C_1 = \sum_{j=0}^{k-1}\alpha_{cj} \,.\, n\, C_0 + \sum_{j=0}^{n-1} \alpha_{qj} \,.\, C_0 + \alpha_m \,.\, n\, C_0 = 4\, n\, C_0 \quad \ldots\ldots\ldots\ldots \text{Eq(4.4)}$$

After re-arranging the decoder in a hierarchical way, the switching activity of the nodes w_{ij} can be expressed as:

$\alpha_{w00} = 2 \quad \ldots\ldots \quad \alpha_{wij} = 1/2^{i-1}$ …………………………..Eq(4.5)

Eq(4.2) and Eq(4.5) let express the effective capacitance of a multi-stage decoder(C_2):

$$C_2 = (2\, \alpha_{c(k-1)} + \ldots + n/2.\, \alpha_{c1} + n\, \alpha_{c(0)}).\, C_0 + \sum_{i=0}^{k-1} 2 \sum_{j=0}^{2^i-1} \alpha_{wij} \,.\, C_0 + \alpha_m C_0 = n\, C_0 \,.\, \sum_{i=0}^{k-1} (1/4)^i$$

$$+\ 4k\, C_0 + 2\, C_0 = (4.\, (n^2 - 1)/3n + 4 \log_2 n + 2).\, C_0 \quad \ldots\ldots\ldots\ldots \text{Eq(4.6)}$$

The saving in dynamic energy obtained from multi-stage clock gating can be expressed as follows Eq(4.7):

$$\Delta E/E_1 = C_1 - C_2 / C_1 \xrightarrow{n \to \infty} 2/3 \quad \ldots\ldots\ldots\ldots \text{Eq(4.7)}$$

Eq(4.7) suggests that up to 67% of dynamic energy can be saved when the circular buffers grow very large. Moreover, the multi-stage clock gating is n times more efficient than the flat clock gating scheme when no memory access occurs (write enable remains passive). This happens because the master clock is locked out at the root of the address decoder tree(Fig 4.10), avoiding any unproductive switching activity. Thus, large buffers with relatively few write accesses, such as those found in high-order FIR filters, benefits the most from hierarchical clock gating.

4.5.2 Stop-Glitch Latch Barriers

Symmetric two-phase level-sensitive clocking brings about a degree of freedom unavailable with the other clocking disciplines, namely the exact location of all slave-triggered latch banks. Relocating them can have a large impact on glitch activities and, hence, on the overall energy efficiency. The most advantageous place is just in front of the most energy-hungry circuit blocks in a data-path. The controller may require some changes too, but latency will remain the same. Applying this technique to the FIR filters in the presented test circuit (Fig. 4.11) results in the block diagram of Fig. 4.12. Data out of latch-based memories, triggered on the master phase, are pre-added and converted in sign-magnitude format; glitches are then suppressed before multiplication.

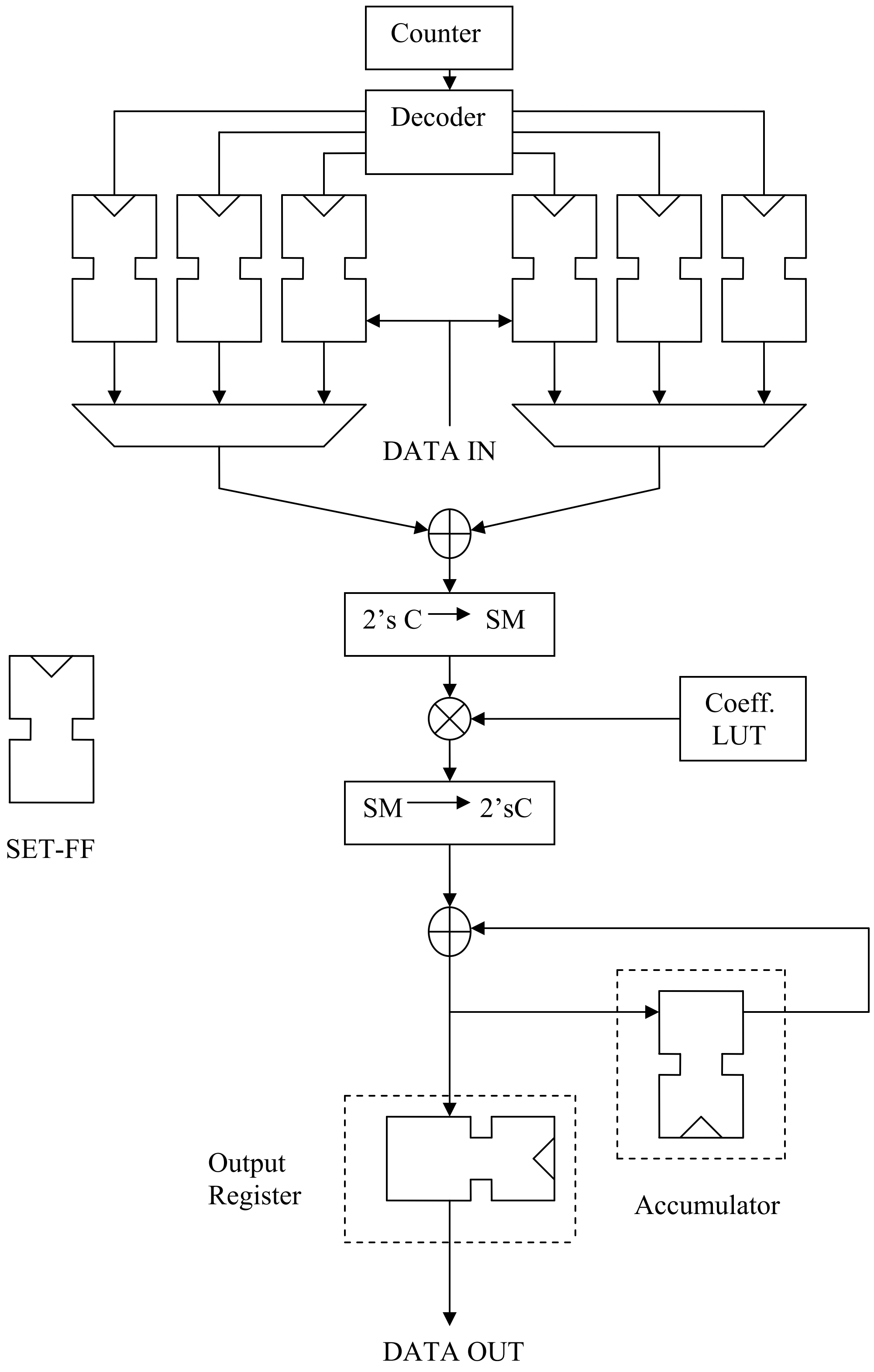


Fig 4.11 : Block Diagram of the FIR Filter with Traditional Clocking

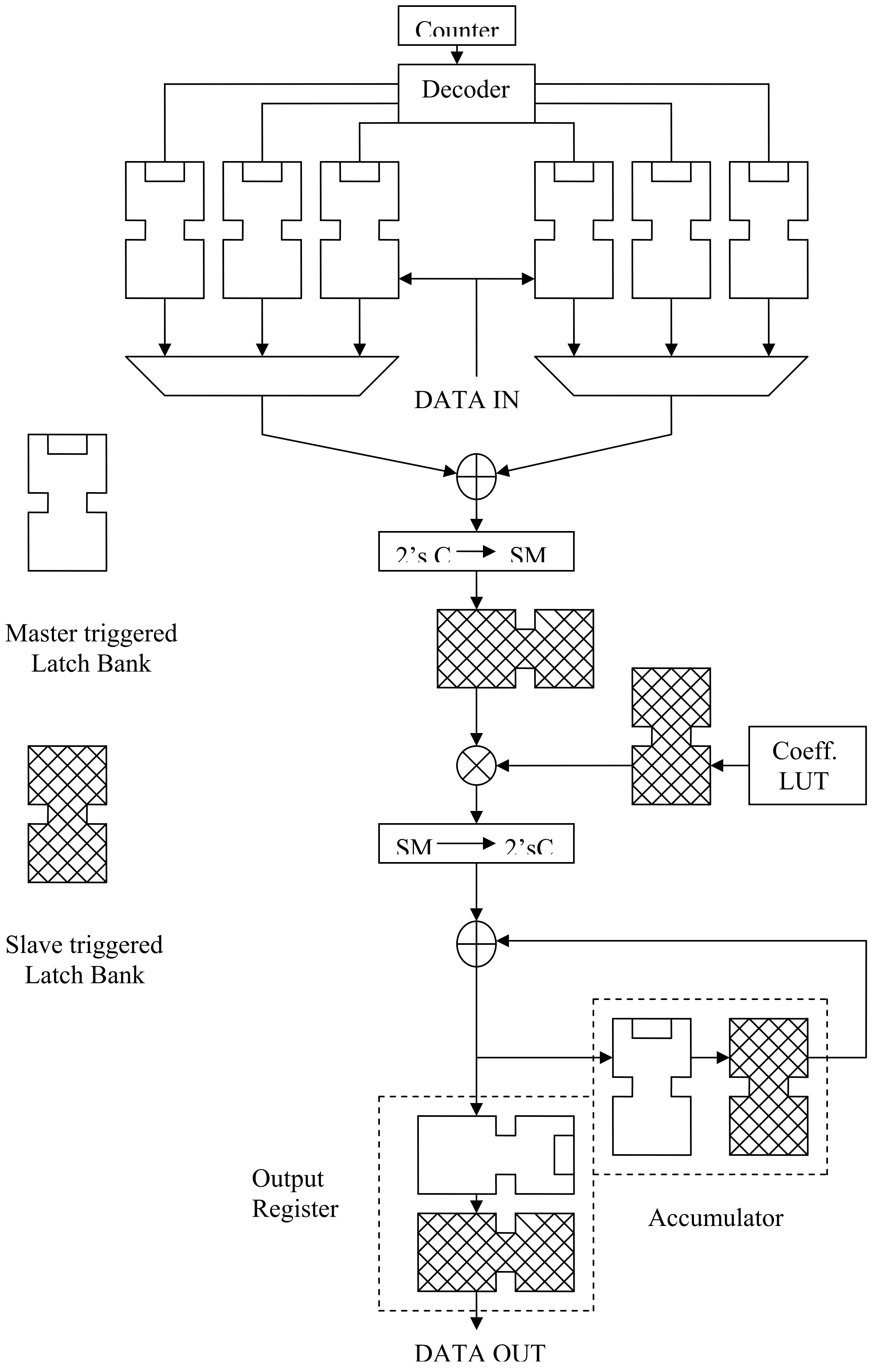


Fig 4.12 : Block Diagram of the FIR Filter with the Proposed Clocking Strategy

4.6 LOW POWER LATCH

In order to exploit the maximum benefits from the clocking techniques introduced before, a low-power latch circuit has been designed. As discussed earlier two latches in parallel can implement a double edge triggered flip-flop. In line with the on-going trend towards low-power, low-voltage operation the circuit of D-Latch shown is original low power circuit. But it suffers from sub-threshold currents if there is a voltage difference across the feedback PMOS pass transistor. Furthermore, the implementation of a weak feedback inverter in Fig. 4.13 will not be beneficial to power saving as it will worsen the low level imbalance across the feedback PMOS pass transistor.

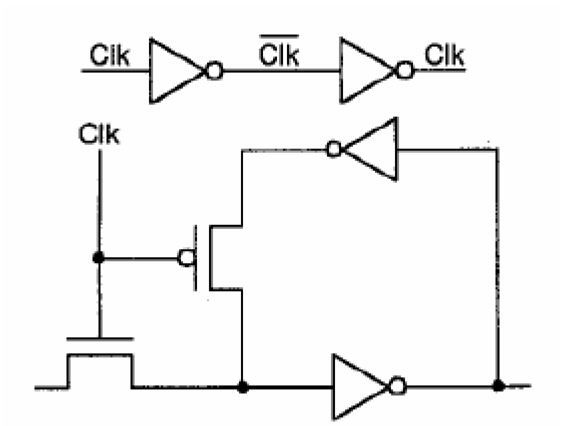


Fig 4.13 : Block Diagram of Original Low Power D-Latch

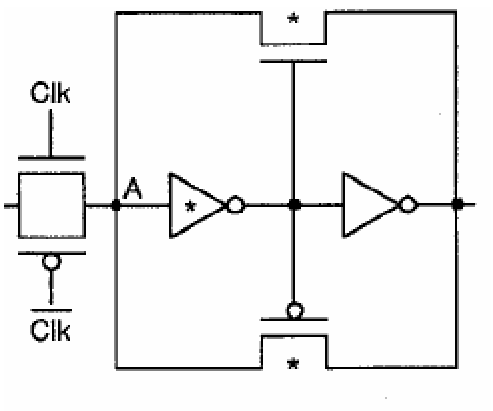


Fig 4.14 : Block Diagram Low Power and Low Voltage D-Latch

Instead, Fig.4.14 shows a circuit with a total of eight transistors. The main advantage of this configuration is that weak inverters and pass transistors marked with * are incorporated to drastically reduce the standby power dissipation. The weak inverter can sufficiently drive the two weak pass transistors and another regular-sized inverter. Similarly, the weak pass transistors only need to maintain the voltage at node **A** when the transmission gate is switched off. Thus, the storage node **A** can withstand a noisy environment and the D-latch is fully static. In addition, both the pass transistors have equal voltage at their sources and drains at all times, eliminating the problem of sub-threshold current leakage. Only the transmission gate and the output inverter need the normal size to maintain the driving capability. Hence, this novel D-latch has a very low standby power dissipation and is suitable for very-low-voltage applications.

Applying this idea a simple 10 transistors low power latch circuit is designed as shown in Fig 4.15. In this circuit a simple pass transistors is substituted for transmission-gates in the feedback and in the D paths in order to avoid local clock inversion or the need for distributing a pair of complementary clocks. This avoided large cross-over contributions and partially traded speed for improved energy efficiency. In this circuit an output inverter is used to provide isolation from glitches that might back-propagate from pin Q to node B, an asynchronous reset to keep the compatibility with our primitive latch. Hence, the final circuit counts 10 transistors against the 21 transistors of the old latch circuit out of the technology library (Fig.4.16), occupying roughly one half of the area. No inverted output was included because, in two-phase asymmetric clocking strategy, at least half of the latches, namely the master-clocked ones, make use only of the Q output. An external low-power inverter was added only when necessary. It should also be mentioned that the transition, for which the novel latch is more attractive in terms of energy dissipation, is when the clock toggles without any change in the input signal D, a very frequent transition in the design. While the new 10-transistor latch would dissipate nearly no energy in this case, the original latch would be highly inefficient due to the inverter at the clock input.

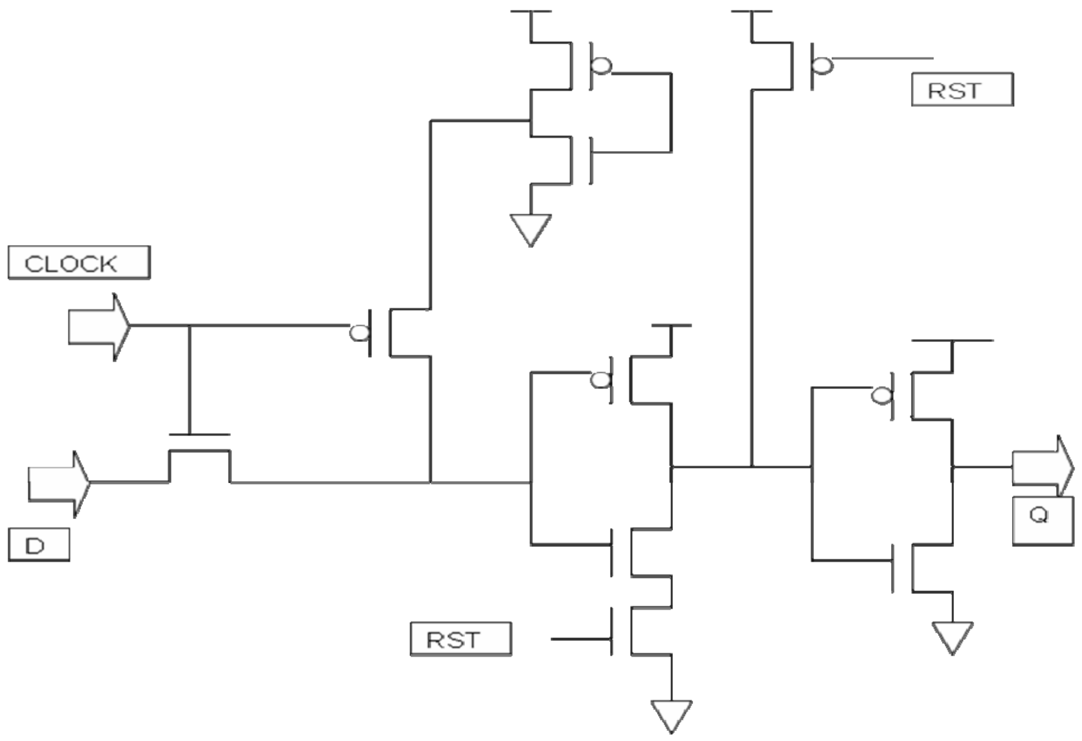
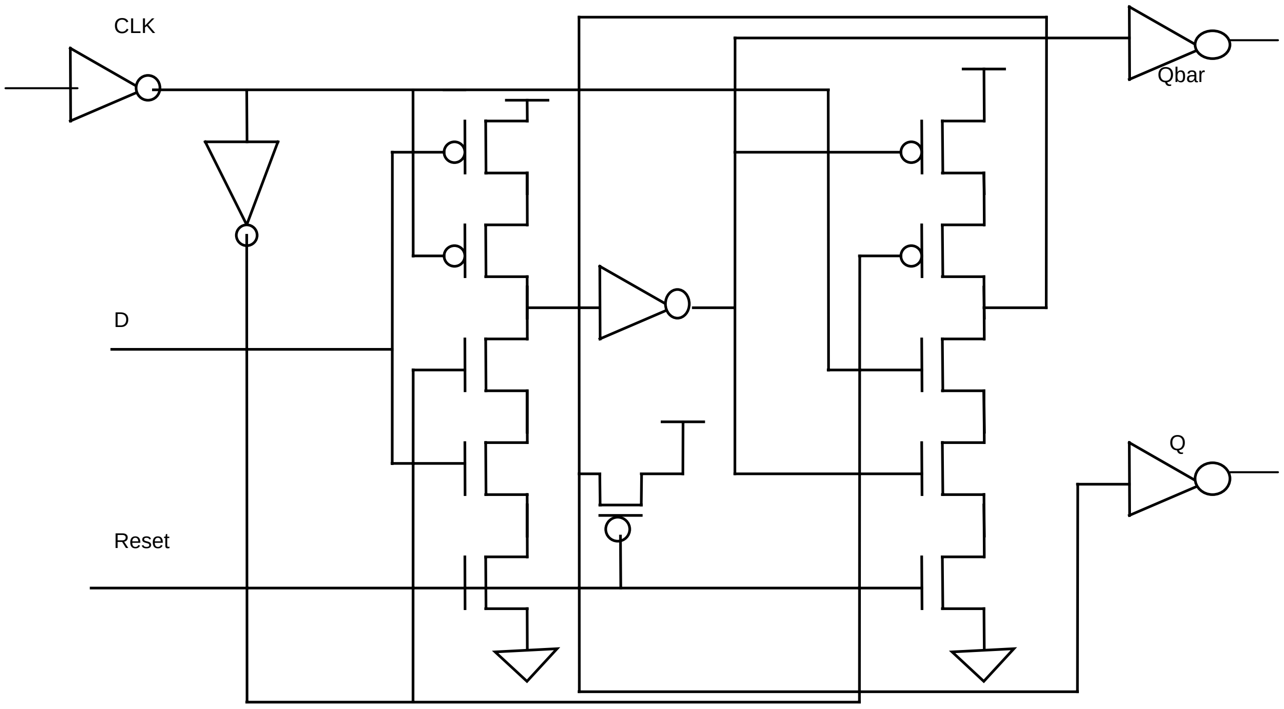


Fig 4.15 : Schematic of 10T Low Power D-Latch



Chapter 5

SIMULATION RESULTS

5.1 Introduction

The simulation results of DSP Processor, FIR Filter with different clocking techniques are given in this chapter. Most of the simulations have done in the VHDL. VHDL stands for *very high-speed integrated circuit* hardware description language, which is one of the programming language used to model a digital system by dataflow, behavioral and structural style of modeling.

VHDL is designed to fill a number of needs in the design process. Firstly, it allows description of the structure of a design, that is how it is decomposed into sub-designs, and how those sub-designs are interconnected. Secondly, it allows the specification of the function of designs using familiar programming language forms. Thirdly, as a result, it allows a design to be simulated before being manufactured, so that designers can quickly compare alternatives and test for correctness without the delay and expense of hardware prototyping.

The Design Vision is a graphical user interface (GUI) to the Synopsys synthesis environment and an analysis tool for viewing and analysing designs at the generic technology (GTECH) level and gate level. Design Vision provides menus and dialog boxes for implementing Design Compiler commands. It also provides graphical displays, such as design schematics.

Mentor Graphics is an Electronic Design Automation (EDA) package. The suite of tools can handle anything from Printed Circuit Board (PCB) to Hardware Definition Language (HDL). In the circuit design area, there are tools for schematic capture, digital and analog simulation, physical layout, and design verification. On the micro (chip) level, there are tools for schematic capture, simulation, physical layout, design verification, and mask generation. Many libraries contain models for popular existing design components

5.2 VHDL Simulation Results

5.2.1 Simulation Result of an Assembly Language Program for a 6-Tap FIR Filter

A simple program for a 6-Tap FIR Filter is written using the instructions of the processors. Basically the program in assembly language for FIR filter consist of load instructions to get the input data and coefficients, then the basic operation involved is multiply and accumulate only. The multiplier designed in the processor takes 6 clock cycles to multiply two 4-bit numbers. The simulation results are shown in Fig 5.1,5.2 & 5.3. The simulation results shows data in hexadecimal (8 bits) which is the opcode value of the instructions used, the input and the filter coefficient value. The pathout is the output port shows the decimal value (8-bits) of the output of the FIR filter. Also the aluflag shows the status of various flags as per the execution of different instructions.

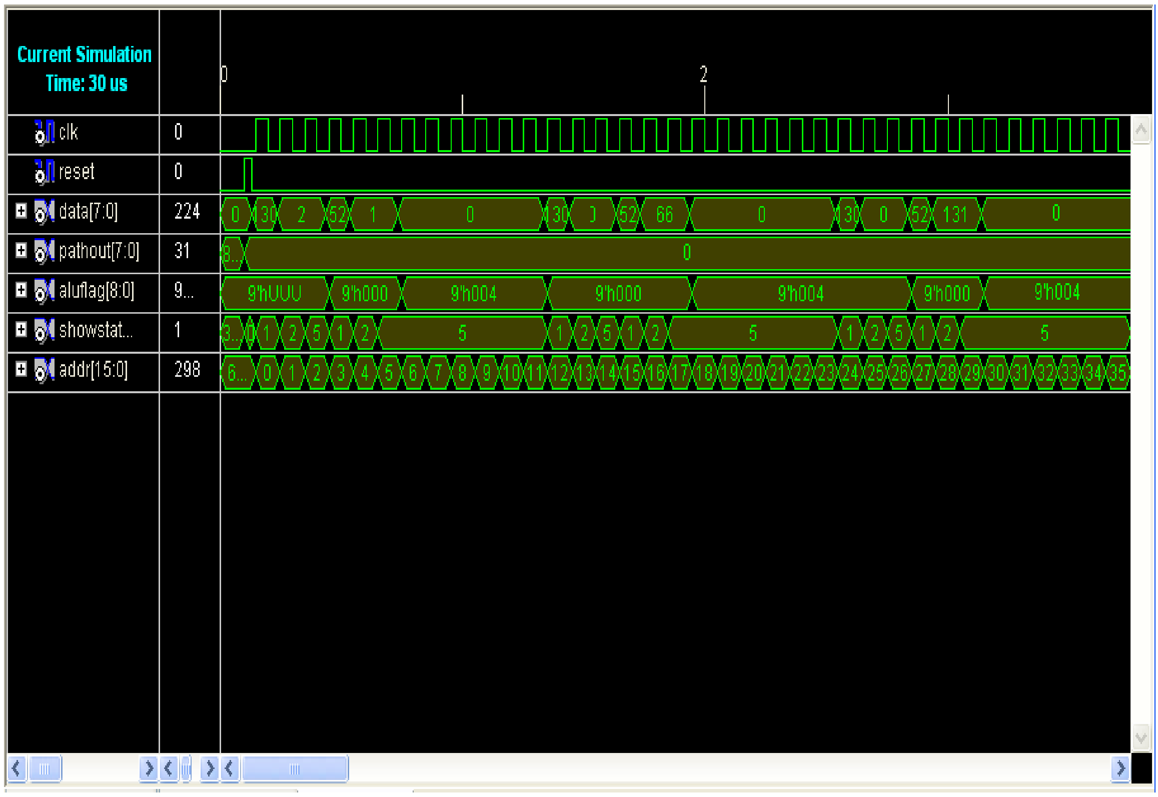


Fig 5.1 : Simulation Output of a 6-Tap FIR Filter Program

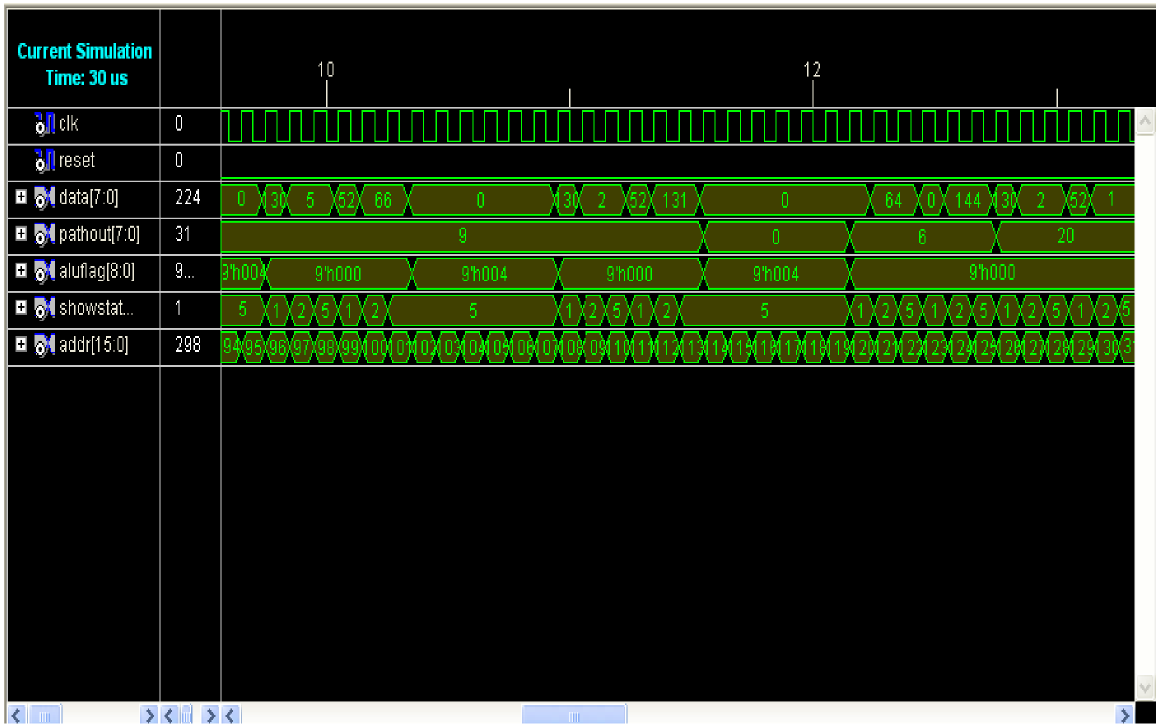


Fig 5.2 : Simulation Output of a 6-Tap FIR Filter Program

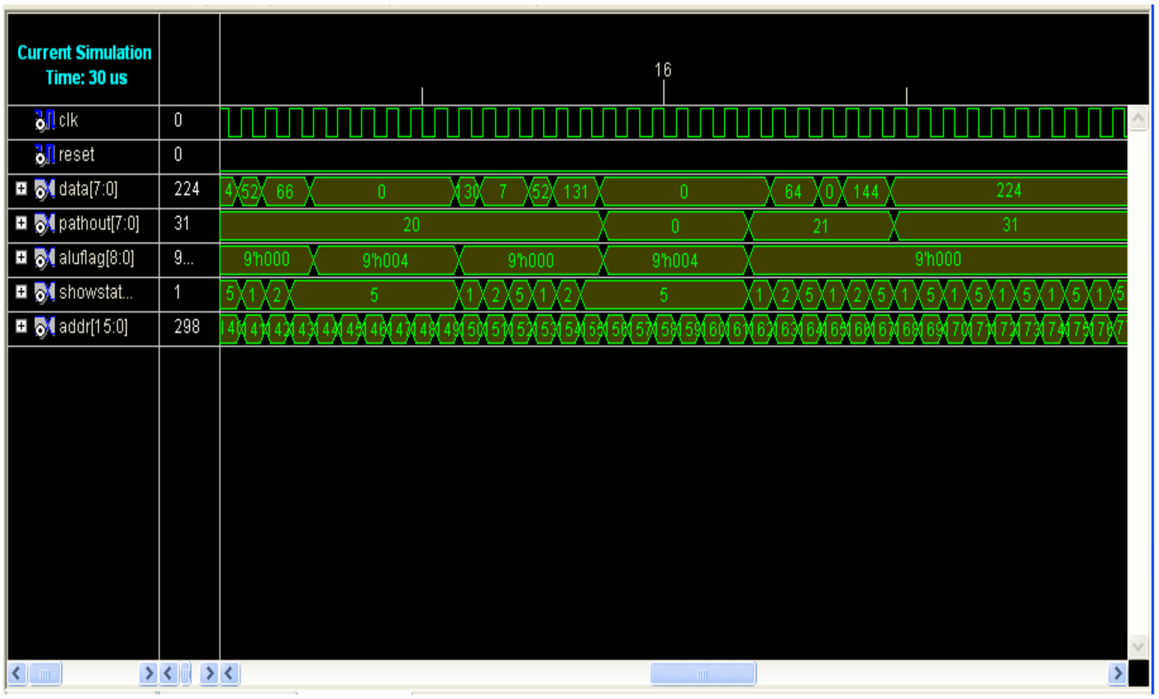


Fig 5.3 : Simulation Output of a 6-Tap FIR Filter Program

5.2.2 Simulation Result of FIR Filter with Single edge Triggered Clocking.

Fig 5.4 shows the simulation output of the FIR Filter Block Diagram shown in Fig 4.10 above. It uses the single edge triggered clocking strategy. It takes less time to produce the output in comparison to the proposed clocking strategy. In the simulation ouput the input port datain takes the input of FIR filter. Another input port coefflut accepts the FIR coefficients. The output port dataout gives the output of FIR filter. All the values of these ports are displayed in decimal.

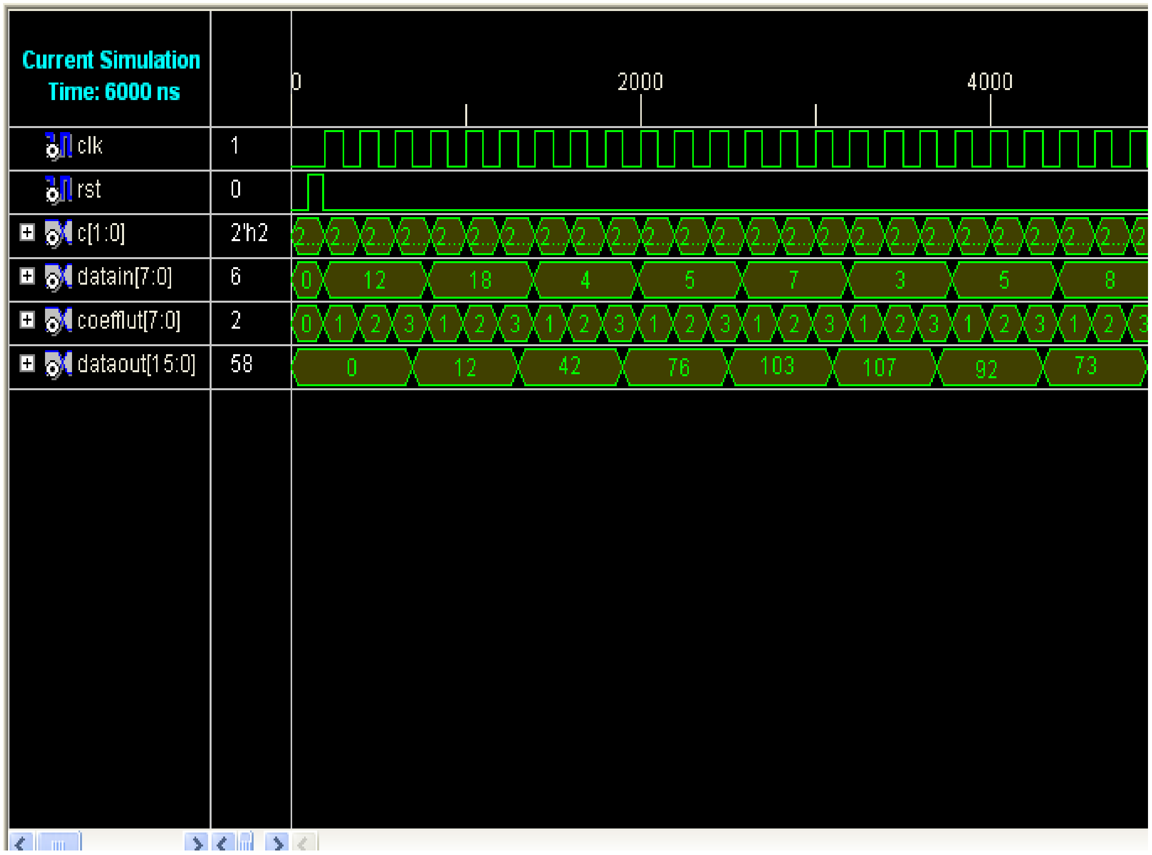


Fig 5.4 : Simulation Output of a 6-Tap FIR Filter Block Diagram Using Single Edge Triggered Clocking Strategy.

5.2.3 Simulation Result of FIR Filter with Single Edge Triggered Clocking.

Fig 5.5 shows the simulation output of the FIR Filter Block Diagram shown in Fig 4.11. It uses the dual edge triggered level sensitive two phase clocking strategy. It takes more time to produce the output in comparison to the single edge triggered clocking strategy. This is because of multi stage clock gating in a re-arranged fashion which takes a longer path as shown in Fig 4.9. In the simulation ouput the input port datain takes the input of FIR filter. Another input port coefflut accepts the FIR coefficients. The output port dataout gives the output of FIR filter. All the values of these ports are displayed in decimal.

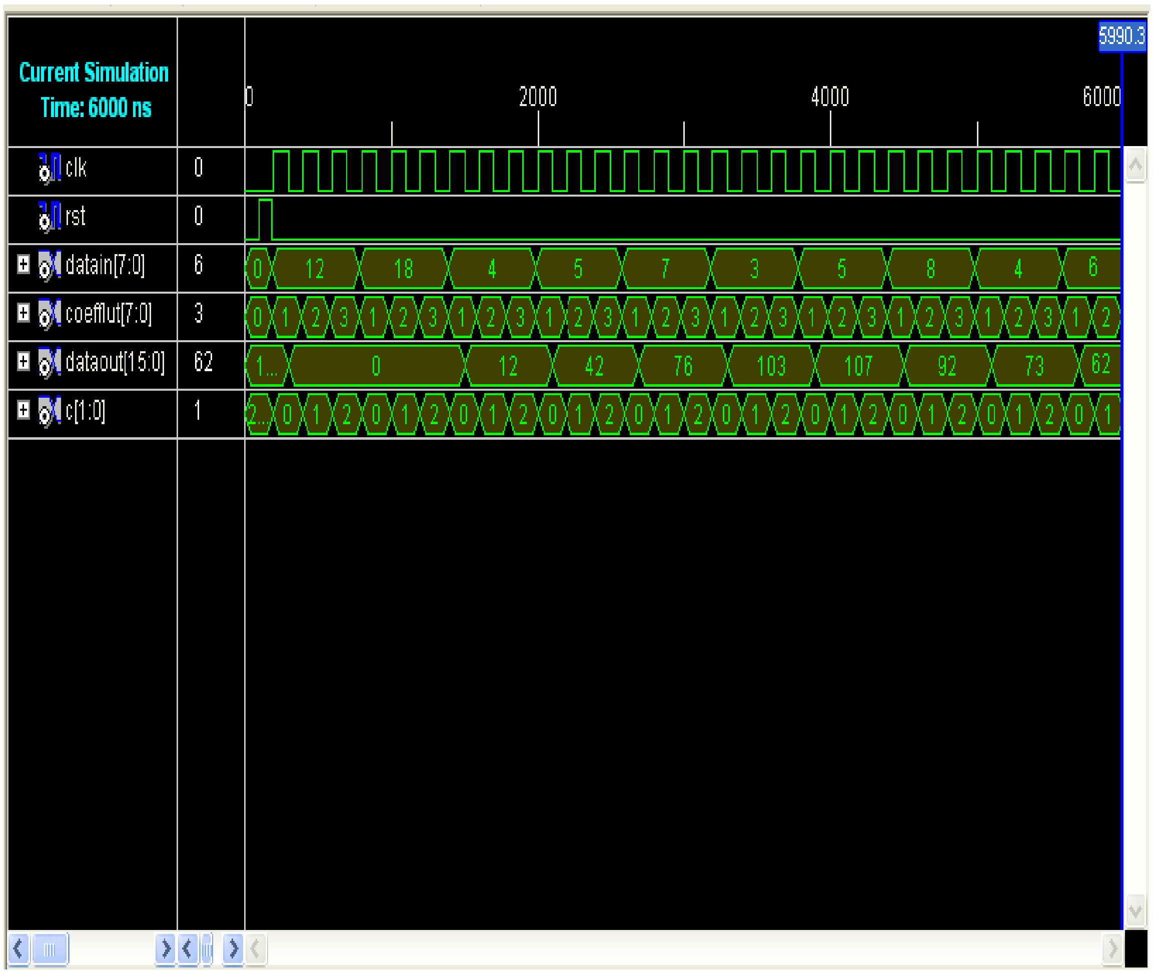


Fig 5.5 : Simulation Output of a 6-Tap FIR Filter Block Diagram Using Proposed Clocking Strategy.

5.2.4 Simulation Output of FIR Filter MAC Unit included inside the Processor

Fig 5.6 & Fig 5.7 shows the output of the FIR Filter MAC using the proposed clocking included inside the DSP Processor. The input data & filter coefficients are stored in the data memory. From the data memory these are supplied to the MAC unit. The signal dataoutalu produces the output of the FIR filter.

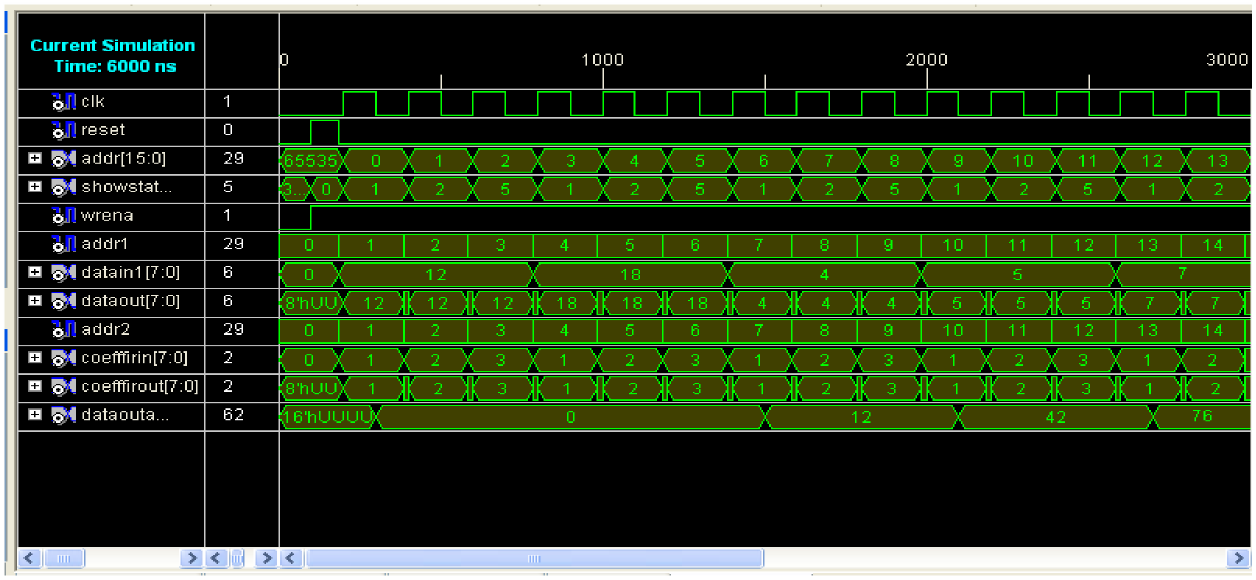


Fig 5.6: Simulation Output of FIR Filter MAC included in the DSP Processor.

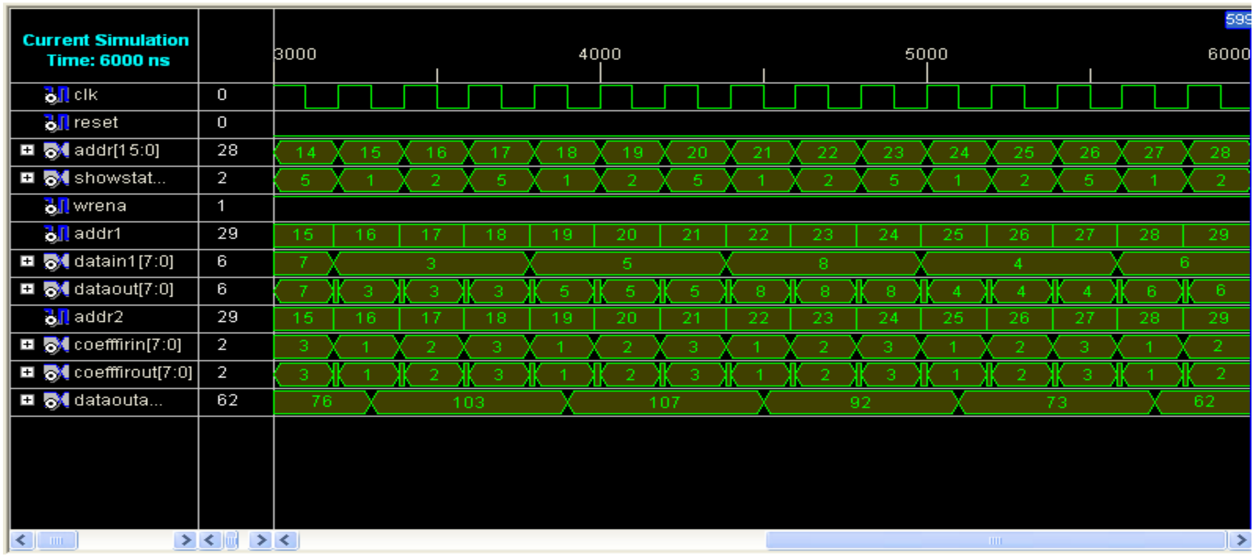


Fig 5.7: Simulation Output of FIR Filter MAC included in the DSP Processor.

5.3 Synopsy Power Reports

5.3.1 Power Report Using Synopsy Tool for the FIR Filter Using Single Edge Triggered Clocking Strategy

The report shown below is generated using the Synopsys Design Vision logic synthesis tool. It takes HDL designs and synthesize them to gate-level HDL netlists. Both verilog and vhdl languages are supported.

```
****************************************
Report : power-analysis_effort low
Design : FinalSET
Version: Y-2006.06-SP4
Date   : Wed May  7 17:17:47 2008
****************************************
Library(s) Used:
tcb013ghpwc (File: /home/NIS/MTECH2008/sakshi/tcb013ghpwc.db)
Operating Conditions: WCCOM   Library: tcb013ghpwc
Wire Load Model Mode: segmented

Design        Wire Load Model          Library
------------------------------------------------
FinalSET              ZeroWireload      tcb013ghpwc
FinalSET_DW01_add_0    ZeroWireload      tcb013ghpwc
FinalSET_DW02_mult_0   ZeroWireload      tcb013ghpwc
FinalSET_DW01_add_1    ZeroWireload      tcb013ghpwc
FinalSET_DW01_add_2    ZeroWireload      tcb013ghpwc

Global Operating Voltage = 1.08
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1nW

Cell Internal Power  =   1.8142 mW   (77%)
Net Switching Power  = 527.8910 uW   (23%)
                       ---------
Total Dynamic Power    =   2.6421 mW  (100%)

Cell Leakage Power     =   4.6389 uW
```

5.3.2 Power Report Using Synopsy Tool for the FIR Filter Using Proposed Clocking Strategy

```
****************************************
Report : power-analysis_effort high
Design : Final3FIR
Version: Y-2006.06-SP4
Date   : Wed May  7 16:44:04 2008
****************************************
Library(s) Used: tcb013ghpwc (File: /home/NIS/MTECH2008/sakshi/tcb013ghpwc.db)
Operating Conditions: WCCOM   Library: tcb013ghpwc
Wire Load Model Mode: segmented

Design        Wire Load Model           Library
------------------------------------------------
Final3FIR             ZeroWireload      tcb013ghpwc
Final3FIR_DW01_add_0   ZeroWireload      tcb013ghpwc
Final3FIR_DW02_mult_0  ZeroWireload      tcb013ghpwc
Final3FIR_DW01_add_1   ZeroWireload      tcb013ghpwc
Final3FIR_DW01_add_2   ZeroWireload      tcb013ghpwc


Global Operating Voltage = 1.08
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1nW
Cell Internal Power  =    1.2462 mW   (81%)
Net Switching Power  = 300.8500 uW   (19%)
                     ---------
Total Dynamic Power    =   1.5471 mW  (100%)

Cell Leakage Power     =   5.6700 uW
```

5.4 Mentor Graphics Power Report

5.4.1 Power Report of the Schematic of Low Power 10 Transistors D-Latch

```
1************13-May-2008********ELDO v6.8_2.1 Production(64 bits) (v6.8_2.1)
**********************10:06:23*****************
0* Component: $MGC_WD/dlatchlow1  Viewpoint: ami05a
0****      INITIAL TRANSIENT SOLUTION       TEMPERATURE =   27.000 DEG C
0*******************************************************************
************************************************
  NODE       VOLTAGE     NODE       VOLTAGE     NODE       VOLTAGE
CLK          0.0000      D           0.0000      N$24          9.7994M
N$25         1.2461M   N$26      0.0000      N$27          999.9994M
N$28         1.0000     N$29      73.2123M     Q             3.6593N
RST          0.0000      VDD       1.0000
   VOLTAGE SOURCE CURRENT
NAME      CURRENT      VOLTAGE      POWER
   V2        0.0000              0.0000            0.0000
   V1       -23.8649P          1.0000           -23.8649P
   V4        0.0000              0.0000            0.0000
   V3        567.1607F         0.0000            0.0000
```

TOTAL POWER DISSIPATION: 23.8649P WATTS

5.4.2 Power Report of the Schematic of Old Technology 21 Transistors D-Latch

```
1************13-May-2008**********ELDOv6.8_2.1Production(64bits)  (v6.8_2.1)
**********************10:34:54*****************
0* Component: $MGC_WD/dlatcnorm  Viewpoint: ami05a
0****   INITIAL TRANSIENT SOLUTION          TEMPERATURE =   27.000 DEG C
0*******************************************************************
************************************************
  NODE       VOLTAGE     NODE       VOLTAGE     NODE       VOLTAGE
   CLK            0.0000        D             0.0000       N$450        1.8000
    N$451         1.8000        N$460        1.8000       N$463        3.0266N
    N$466         3.0266N       N$475        1.8000       N$476        16.5812M
    N$478         1.8000        N$479        16.5812M     Q            3.0266N
    QB             1.8000       RST           0.0000      VDD           1.8000
   VOLTAGE SOURCE CURRENT
NAME      CURRENT      VOLTAGE      POWER
   V4         0.0000             0.0000          0.0000
   V3         0.0000             0.0000          0.0000
   V2         0.0000             0.0000          0.0000
   V1        -65.8990P         1.8000          -118.6183P
```

TOTAL POWER DISSIPATION: 118.6183P WATTS

Chapter 6

CONCLUSION & FUTURE WORK

6.1 Conclusion

- The simple DSP processor developed is a good stepping stone for future complex and optimized DSPs. With a set of branch instructions the project DSP will operate as a CISC processor with strong math capabilities. Currently, the biggest strength of the DSP is its shifting speed and internal accumulator operations. With current instructions Fourier transforms, correlations and signal filtering can realistically be solved using this DSP.
- The individual components were tested separately using the simulator. The ALU, multiplier, shifter and limiter were exhaustively tested for all possible inputs and correct outputs. All of these components work perfectly. The fetch state machine was tested in the same manner and worked properly for all implemented instructions. Currently all implemented functions, function correctly.
- Applying the proposed clocking strategy to the FIR filter simulation results shows 42% reduction in the power dissipation.
- Also simulations for the low power latch design confirm the functionality and the energy savings of the design: each latch dissipates five times less in comparison to old latch.

6.2 Future Work

- Advanced DSP features such as hardware looping, dual memory buses, and single clock multiply-accumulate can be added in DSP's structure.
- These techniques can be implemented in various applications of portable devices for example in the front end of a noise suppression circuit that combines signals from two microphones. Such circuits find applications in hearing aids. It can be implemented in many low power ASIC's.

REFERENCES

[1] Flavio Carbognani, Felix Buergin, Norbert Felber, Hubert Kaeslin and Wolfgang Fichtner “42% Power Savings through Glitch Reducing Clocking Strategy in a Hearing Aid Application, IEEE Trans. pp 2941-2944(ISCAS 2006).

[2] N. Nedovic and V. G. Oklobdzija, "Dual-edge triggered storage elements and clocking strategy for low-power systems," IEEE Trans. VLSI Syst., vol. 13, no. 5, pp. 577-590, May 2005.

[3] T. A. Johnson and I. S. Kourtev, "A single latch, high speed double-edge triggered flip-flop (DETFF)," in Proc. IEEE International Conference on Electronics, Circuits and Systems (ICECS 2001), Malta, Sept. 2001, pp. 189-192.

[4] V. Zyuban, "Optimization of scannable latches for low energy," IEEE Trans. VLSI Syst., vol. 11, no. 5, pp. 778-788, Oct. 2003.

[5] P. Mosch, G. van Oerle, S. Menzl, N. Rougnon-Glasson, K. van Nieuwen-hove, and M. Wezelenburg, "A 660-,uW 50-Mops 1-V DSP for a hearing aid chip set," IEEEJ. Solid-State Circulits, vol. 35, no. 11I, pp. 1705-1712, Nov 2000

[6] F. Carbognani, F. Buergin, N. Felber, H. Kaeslin, and W. Fichtner, "Two-phase clocking and a new latch design for low-power portable applications," in Proc. Power and Timing Modeling, Optimization and and Simulation (PATMOS'05), Leuven, Belgium, Sept. 2005, pp. 446-455.

[7] Ching, L., Ling, O.: Low-power and low-voltage D-latch. IEE Electronics Letters **34** (1998) 641-642

[8] Zyuban, V., Meltzer, D.: Clocking Strategies and Scannable Latches for Low Power Applications. ISLPED (2001) 346-351

[9] Arm, C., Masgonty, J., Piguet, C.: Double-Latch Clocking Scheme for Low-Power I.P. Cores. PATMOS (2000) 217-224

[10] Mahesh Mehendale, Sunil D. Sherlekar and G.Venkatesh” Low Power Realization of FIR Filters on Programmable DSPs”, IEE Tran,(VLSI Systems) vol.6,no.4, Dec1998

[11] Circuit Design with VHDL by Volnei A Pedroni.

[12] The National Technology Roadmap for Semiconductors, Semiconductor Industry Association (SIA), 1999–2000.

[13] A. Jain *et al.*, "A 1.2 GHz alpha microprocessor with 44.8 GB/s chip pin bandwidth," in *IEEE Int. Solid-State Circuits Conf. Tech. Dig.*, Feb. 2001, pp. 240–241.

[14] P. Hofstee *et al.*, "A 1-GHz single-issue 64 b powerPC processor," in *IEEE Int. Solid-State Circuits Conf. Tech. Dig.*, Feb. 2000, pp. 92–93.

[15] R. P. Llopis and M. Sachdev, "Low power, testable dual edge triggered flip-flops," in *Int. Symp. Low Power Electronics and Design Tech. Dig.*, 1996, pp. 341–345.

[16] A. Gago, R. Escano, and J. A. Hidalgo, "Reduced implementation of D-type DET flip-flops," *IEEE J. Solid-State Circuits*, vol. 28, no. 3, pp. 400–402, Mar. 1993.

[17] J. Tschanz, S. Narendra, Z. Chen, S. Borkar, M. Sachdev, and V. De, "Comparative delay and energy of single edge-triggered & dual edgetriggered pulsed flip-flops for high-performance microprocessors," in *Int. Symp. Low Power Electronics and Design Tech. Dig.*, Aug. 2001, pp. 147–152.

[18] N. Nedovic, M. Aleksic, and V. G. Oklobdzija, "Conditional pre-charge techniques for power-efficient dual-edge clocking," in *Int. Symp. Low Power Electronics and Design Tech. Dig.*, Aug. 2002, pp. 56–59.

[19] N. Nedovic, V. G. Oklobdzija, M. Aleksic, and W. W. Walker, "A low power symmetrically pulsed dual edge-triggered flip-flop," in *Proc. 28th European Solid-State Circuits Conf.*, Sept. 2002, pp. 399–402.

[20] N. Nedovic and V. G. Oklobdzija, "Timing characterization of dual-edge triggered flip-flops," in *Proc. Int. Conf. Computer Design*, Sept. 2001, pp. 538–541.

[21] V. Stojanovic and V. G. Oklobdzija, "Comparative analysis of masterslave latches and flip-flops for high-performance and low-power systems," *IEEE J. Solid-State Circuits*, vol. 34, no. 4, pp. 536–548, Apr. 1999.

[22] N. Nedovic, W. W. Walker, and V. G. Oklobdzija, "A test circuit for measurement of clocked storage element characteristics," *IEEE J. Solid-State Circuits*, to be published.

[23] D. Markovic, B. Nikolic, and R. W. Brodersen, "Analysis and design of low-energy flip-flops," in *Int. Symp. Low Power Electronics and Design Tech. Dig.*, Aug. 2001, pp. 52–55.

APPENDIX A

DSP INSTRUCTION SET AND OPCODES

(1) ALU Function

ALU Function 2-Operand Op= 000 Func(4)=0				
	Op	Func	aca	acb
ADD	000	00000	xx	xx
SUB	000	00001	xx	xx
AND	000	00010	xx	xx
OR	000	00011	xx	xx
XOR	000	00100	Xx	xx
CMP	000	00101	xx	xx

ALU Function 1-Operand Op= 000 Func(4)=1				
	Op	Func	aca	#N
ADDI	000	10000	xx	NNNNNN
SUBI	000	10001	xx	NNNNNN
ANDI	000	10010	xx	NNNNNN
ORI	000	10011	xx	NNNNNN
XORI	000	10100	Xx	NNNNNN

Arithmetic 1-Operand Op= 000				
	Op	Func	aca	Don’t Care
NOT	000	10110	xx	xx
INC	000	10111	xx	xx
DEC	000	11000	xx	xx
CLR	000	11001	xx	xx
PASS	000	11111	xx	xx
NEG	000	11010	xx	xx
ABS	000	11011	xx	xx

(2) Special Function

	Opcode	Func
NOP	111	00000
STOP	111	11111
SETC	111	10001
CLRC	111	00001
SETN	111	10010
CLRN	111	00010
SETZ	111	10100
CLRZ	111	00100
RETN	111	01111

(3) Multiply Functions

	Op	func	aca	ra	rb
MULT	001	00100	XX	XX	XX
MULTI	001	10100	XX	XX	NNNN
SQR	001	00101	XX	XX	XX
MAC	001	00000	XX	XX	XX

(4) Load Functions

	Op	func	aca	ra	imm	disp
LD	100	10001	------	XX	-------	16
LDI	100	00010	------	XX	NNNN	-----
LDA	100	00100	XX	XX	--------	-----
LDD	100	01000	------	XX	--------	-----

(5) Branch Functions

	Op	func	Don’t Care	disp
BR	011	00000	--------------	16
BEQ	011	00001	--------------	16
BNE	011	00010	--------------	16
BLT	011	00011	---------------	16
BGT	011	00100	---------------	16

Subroutine   func(4 and 3) = 01

	Op	func	Don’t Care	Disp
CALL	011	01000	---------------	16

(6)Shift Functions

1-Operand Shift func=11111 Shop(3)=0				
	Op	func	aca	Shop
ASL	010	11111	XX	0000
ASR	010	11111	XX	0001
LSR	010	11111	XX	0010
ROL	010	11111	XX	0011
ROR	010	11111	XX	0100
RND	010	11111	XX	0101
TNK	010	11111	XX	0110

Shift to Memory

	Op	func	aca	Shop	disp
RNDA	010	----------	XX	1100	16
TNKA	010	----------	XX	1010	16
LIMA	010	----------	XX	1000	16
RNDF	010	----------	XX	1101	16
TNKF	010	----------	XX	1011	16
LIMF	010	----------	XX	1001	16

APPENDIX B

DSP ASSEMBLER

```
$nolist

r0 equ 0

r1 equ 1

r2 equ 2

r3 equ 3

a0 equ 0

a1 equ 1

a2 equ 2

a3 equ 3

orig macro addr

cseg at 8000h

base equ $

cseg at 8000h+addr

LED equ base+4000h

endm

;------- ALU FUNCTIONS (2-op) ------------

zadd macro a,ab

dcb 00h

dcb 00h+(a SHL 6)+ (ab SHL 4)

endm

zsub macro a,ab

dcb 01h

dcb 00h+(a SHL 6)+ (ab SHL 4)
```

```
endm

zand macro a,ab

dcb 02h

dcb 00h+(a SHL 6)+ (ab SHL 4)

endm

zor macro a,ab

dcb 03h

dcb 00h+(a SHL 6)+ (ab SHL 4)

endm

zxor macro a,ab

dcb 04h

dcb 00h+(a SHL 6)+ (ab SHL 4)

endm

zcmp macro a,ab

dcb 05h

dcb 00h+(a SHL 6)+ (ab SHL 4)

endm

;------- ALU FUNCTIONS (1-op)IMMEDIATE ------------

zaddi macro a,i

dcb 10h

dcb 00h+(a SHL 6)+ i

endm

zsubi macro a,i

dcb 11h

dcb 00h+(a SHL 6)+ i

endm
```

```
zandi macro a,i

dcb 12h

dcb 00h+(a SHL 6)+ i

endm

zori macro a,i

dcb 13h

dcb 00h+(a SHL 6)+ i

endm

zxori macro a,i

dcb 14h

dcb 00h+(a SHL 6)+ i

endm

;------- ALU FUNCTIONS (1-op)ARITHMETIC ------------

znot macro a

dcb 16h

dcb 00h+(a SHL 6)

endm

zinc macro a

dcb 17h

dcb 00h+(a SHL 6)

endm

zdec macro a

dcb 18h

dcb 00h+(a SHL 6)

endm

zzero macro a
```

```
dcb 19h

dcb 00h+(a SHL 6)

endm

zpass macro a

dcb 1fh

dcb 00h+(a SHL 6)

endm

;------- SPECIAL FUNCTIONS (0-op)------------

znop macro

dcb 0e0h

endm

zstop macro

dcb 0ffh

endm

zsetc macro

dcb 0e1h

endm

zclrc macro

dcb 0f1h

endm

zsetn macro

dcb 0e2h

endm

zclrn macro

dcb 0f2h

endm
```

```
zsetz macro

dcb 0e3h

endm

zclrz macro

dcb 0f3h

endm

;------ BRANCH FUNCTIONS (normal)-----------

zbr macro addr

dcw 0060h

dcw (addr - base)

endm

zbeq macro addr

dcw 0061h

dcw (addr - base)

endm

zbne macro addr

dcw 0062h

dcw (addr - base)

endm

zblt macro addr

dcw 0063h

dcw (addr - base)

endm

zbgt macro addr

dcw 0064h

dcw (addr - base)
```

```
endm

;------ BRANCH FUNCTIONS (subroutine)-----------

zcall macro addr

dcw 0068h

dcw (addr - base)

endm

zretn macro

dcb 6fh

endm

;------ BRANCH FUNCTIONS (special)-----------

zbbs macro addr

dcw 0070h

dcw (addr - base)

endm

zbbc macro addr

dcw 0071h

dcw (addr - base)

endm

zrpt macro addr

dcw 007fh

dcw (addr - base)

endm

;------ MULT FUNCTIONS-----------

zmult macro a,ra,rb

dcb 20h

dcb 00h+(a SHL 6)+ (ra SHL 4)+(rb SHL 2)
```

```
endm

zmulti macro a,ra,i

dcb 30h

dcb 00h+(a SHL 6)+ (ra SHL 4)+ i

endm

zsqr macro a,ra

dcb 21h

dcb 00h+(a SHL 6)+ (ra SHL 4)

endm

;------ SHIFT FUNCTIONS (1-op) -----------

zasl macro a

dcb 5fh

dcb 00h+(a SHL 6)+ 0

endm

zasr macro a

dcb 5fh

dcb 00h+(a SHL 6)+ 1

endm

zlsr macro a

dcb 5fh

dcb 00h+(a SHL 6)+ 2

endm

zrol macro a

dcb 5fh

dcb 00h+(a SHL 6)+ 3

endm
```

```
zror macro a

dcb 5fh

dcb 00h+(a SHL 6)+ 4

endm

zrnd macro a

dcb 5fh

dcb 00h+(a SHL 6)+ 5

endm

ztnk macro a

dcb 5fh

dcb 00h+(a SHL 6)+ 6

endm

;------ SHIFT FUNCTIONS (to mem) -----------

zrnda macro a,addr

dcb 4000h+(a SHL 6)+ 0dh

dcw (addr - base)

endm

ztnka macro a,addr

dcb 4000h+(a SHL 6)+ 0eh

dcw (addr - base)

endm

;------ SHIFT FUNCTIONS (with ALU) -----------

zaslk macro alu,a

dcb 40h+alu

dcb 00h+(a SHL 6)+ 0

endm
```

```
zasrk macro alu,a

dcb 40h+alu

dcb 00h+(a SHL 6)+ 1

endm

zlsrk macro alu,a

dcb 40h+alu

dcb 00h+(a SHL 6)+ 2

endm

zrolk macro alu,a

dcb 40h+alu

dcb 00h+(a SHL 6)+ 3

endm

zrork macro alu,a

dcb 40h+alu

dcb 00h+(a SHL 6)+ 4

endm

zrndk macro alu,a

dcb 40h+alu

dcb 00h+(a SHL 6)+ 5

endm

ztnkk macro alu,a

dcb 40h+alu

dcb 00h+(a SHL 6)+ 6

endm

;------- LOAD FUNCTIONS --------

zldd macro r,rb
```

```
dcb 88h

dcb 00h+ (r SHL 4) + (rb SHL 2)

endm

zldi macro r,i

dcb 82h

dcb 00h + (r SHL 4)+ i

endm

zld macro r,addr

dcb 80h

dcb 00h+ (r SHL 4)

dcw (addr - base)

endm

zlda macro r,a

dcb 84h

dcb 00h+ (a SHL 6) + (a SHL 4)

endm

zsta macro a,addr

dcb 90h

dcb 00h+ (a SHL 6)

dcw (addr - base)

endm


$list
```