Low Power Adaptive Equaliser Architectures for Wireless LMMSE Receivers

Mark P. Tennant

A thesis submitted for the degree of Doctor of Philosophy.
The University of Edinburgh.
October 2007

Abstract

Power consumption requires critical consideration during system design for portable wireless communication devices as it has a direct influence on the battery weight and volume required for operation. Wideband Code Division Multiple Access (W-CDMA) techniques are favoured for use in future generation mobile communication systems. This thesis investigates novel low power techniques for use in system blocks within a W-CDMA adaptive linear minimum mean squared error (LMMSE) receiver architecture. Two low power techniques are presented for reducing power dissipation in the LMS adaptive filter, this being the main power consuming block within this receiver. These low power techniques are namely the decorrelating transform, this is a differential coefficient technique, and the variable length update algorithm which is a dynamic tap-length optimisation technique.

The decorrelating transform is based on the principle of reducing the wordlength of filter coefficients by using the computed difference between adjacent coefficients in calculation of the filter output. The effect of reducing the wordlength of filter coefficients being presented to multipliers in the filter is a reduction in switching activity within the multiplier thus reducing power consumed. In the case of the LMS adaptive filter, with coefficients being continuously updated, the decorrelating transform is applied to these calculated coefficients with minimal hardware or computational overhead. The correlation between filter coefficients is exploited to achieve a wordlength reduction from 16 bits down to 10 bits in the FIR filter block.

The variable length update algorithm is based on the principle of optimising the number of operational filter taps in the LMS adaptive filter according to operating conditions. The number of taps in operation can be increased or decreased dynamically according to the mean squared error at the output of the filter. This algorithm is used to exploit the fact that when the SNR in the channel is low the minimum mean squared error of the short equaliser is almost the same as that of the longer equaliser. Therefore, minimising the length of the equaliser will not result in poorer MSE performance and there is no disadvantage in having fewer taps in operation. If fewer taps are in operation then switching will not only be reduced in the arithmetic blocks but also in the memory blocks required by the LMS algorithm and FIR filter process. This reduces the power consumed by both these computation intensive functional blocks. Power results are obtained for equaliser lengths from 73 to 16 taps and for operation with varying input SNR.

This thesis then proposes that the variable length LMS adaptive filter is applied in the adaptive LMMSE receiver to create a low power implementation. Power consumption in the receiver is reduced by the dynamic optimisation of the LMS receiver coefficient calculation. A considerable power saving is seen to be achieved when moving from a fixed length LMS implementation to the variable length design. All design architectures are coded in Verilog hardware description language at register transfer level (RTL). Once functional specification of the design is verified, synthesis is carried out using either Synopsys *DesignCompiler* or Cadence *BuildGates* to create a gate level netlist. Power consumption results are determined at the gate level and estimated using the Synopsys *DesignPower* tool.

Declaration of originality

I hereby declare that the research recorded in this thesis and the thesis itself was composed and originated entirely by myself in the School of Engineering and Electronics at The University of Edinburgh.

Mark Tennant

Acknowledgements

I would like to thank Professor Tughrul Arslan and Dr John Thompson for their help and guidance during my research.

Thanks also to Dr Ahmet Erdogan and to my lab colleagues for their help and interesting discussion.

Contents

List of figures

List of tables

Acronyms and abbreviations

ASE	Accumulated Squared Error
AWGN	Additive White Gaussian Noise
BEP	Bit Error Probability
CDMA	Code Division Multiple Access
CMOS	Complimentary Metal Oxide
DCM	Differential Coefficient Method
DF	Direct Form
DS	Direct Sequence
DSP	Digital Signal Processor
FDMA	Frequency Division Multiple Access
FH	Frequency Hopping
FIR	Finite Impulse Response
FSM	Finite State Machine
HDL	Hardware Description Language
IPI	Inter-Path Interference
LFSR	Linear Feedback Shift Register
LMMSE	Linear Minimum Mean Squared Error
LMS	Least Mean Squares
LUD	Length Update
MAC	Multiplier Accumulator
MAI	Multiple Access Interference
MF	Matched Filter
MLSD	Maximum Likelihood Sequence Detector
MRC	Maximal Ratio Combining
MSE	Mean Squared Error
PN	Pseudo-Random Noise
RAM	Random Access Memory
RLS	Recursive Least Squares
RTL	Register Transfer Level

SDF	Standard Delay Format
SNR	Signal to Noise Ratio
SS	Spread Spectrum
TDF	Transposed Direct Form
TDMA	Time Division Multiple Access
VL-LMS	Variable Length Least Mean Squares
W-CDMA	Wideband Code Division Multiple Access
WUD	Weight Update

Nomenclature

clk	clock signal
f	clock frequency
P_{sw}	switching power consumption
S_w	switching activity
C_{load}	load capacitance
V_{dd}	supply voltage
$x(n)$	input sampled sequence
$y(n)$	output sampled sequence
b_i	FIR filter coefficient
i	filter coefficient index
N	FIR filter order
e	adaptation error signal
d	data signal
μ	LMS step-size
c	PN spreading code signal
s	spread spectrum signal
G	processing gain
t	continuous time
T	signal period
T_c	chip period
τ	delay
g	channel estimation
m	transform order
δ_i	decorrelating transform filter coefficients
$\mathbf{R}$	auto-correlation matrix
$\mathbf{p}$	cross-correlation matrix
J	mean squared error
M	sub-filter index
P	sub-filter order

W	active sub-filter segments
η	additive white gaussian noise
K	mobile users
L	propagation paths
A	signal amplitude
j	chip index
S	samples per chip
r	received signal
h	original transmitted data symbol
$\hat{h}$	estimate of original transmitted data symbol
E	signal energy
g	channel impulse response
$\delta(t)$	Dirac's delta function
$\mathbf{w}$	adaptive receiver filter coefficient vector
$\mathbf{r}$	received signal vector
∇	gradient vector
s	sampled signature sequence

Chapter 1
Introduction

1.1 Motivation

The popularity of portable, battery-powered consumer devices is continuing the high demand for low-power, high performance electronic components. This is especially so in the marketplace for wireless communication devices such as 2.5G/3G mobile phones, wireless enabled laptops and other data enabled devices which require ever more powerful computation and real-time signal processing capabilities.

As bandwidth for wireless communications systems continues to be a limited resource, the development of new technologies that are more spectrally efficient or can reuse parts of the spectrum that have already been allocated, becomes even more necessary. At the same time, these new devices are expected to display higher data transfer rates, have lower power requirements and service many users simultaneously. This time averaged power consumption in particular, requires critical consideration during design of portable wireless devices as it has a direct influence on the battery weight and volume required for operation.

Supporting applications which require high speed computation and real-time processing capabilities has the effect of increasing the need for higher clock rates and gate counts. All this is at the cost of power consumption which, added to the physical limitations of battery technology, packaging and thermal management, raises the issue of low-power design at every turn.

1.1.1 The Need for Wideband Technologies

Recent demand for the ability to access data without being constrained by physical location has fuelled the need for high-bandwidth, wireless, convergent computing devices. This is driven by the desire for high speed exchange of information. Our dependence on computers, mobile telephones, personal digital assistants (PDA's) and other internet enabled multi-media

devices, is growing rapidly which of course presents the challenge to industry to provide the most competitive and timely solutions to this market.

An increasing number of the population also rely on being mobile with more people than ever having jobs that require mobility more of the time. Many people cannot afford to be out of touch and use applications which require real-time access to critical information in any location and at any time. It is also argued that wireless data technology is highly successful in business applications, increasing productivity, customer satisfaction and in many cases providing a competitive advantage.

1.2 Contribution

For future generations of wideband wireless devices the design optimisation of the receiver architecture is critical to the power consumption and data throughput of the overall system. From a review of published material carried out, it has been found that the optimisation of W-CDMA receivers is worthwhile and, more specifically, the adaptive LMMSE-RAKE receiver has been identified and chosen for optimisation in this work. The literature reviewed which relates to these topics, addresses a number of ways to carry out improvements to receiver sub-components individually rather than a complete receiver system.

The ultimate aim of this project is to develop a novel adaptive LMMSE receiver architecture in HDL which uses algorithms that have been optimised to reduce power consumption and at the same time retain or have improved performance characteristics in W-CDMA applications. Optimisation would take place in the form of selecting and applying one or more of low-power techniques described in Chapter 2 with the improvements in performance gained from careful application of these new algorithms.

1.2.1 Objectives

The objective set out for this thesis were:

1. Produce a standard functional adaptive filter HDL implementation that uses the LMS algorithm[1] for updating filter weights. This will be used as a conventional implementation for comparison for future low power designs.

2. Implement an optimised algorithm into an LMS adaptive FIR filter core. Use of the decorrelating transform technique will be investigated and its suitability assessed.

3. Investigate the implementation of a variable tap length LMS adaptive filter, firstly through development of the conventional equaliser, then through investigation of translating to a low power optimised design.

4. Produce an HDL implementation of an adaptive LMMSE receiver[2] using functional blocks created in the preceding design stages resulting in an optimised low power receiver architecture for W-CDMA systems[3].

An investigation into how concepts, such as those described in Chapters 2 and 3, can be applied to this architecture will be made. The introduction of the variable tap length adaptive filter, along with how this can be applied in an adaptive LMMSE architecture presents the optimisation of its switching activity in real-time. The concept of switching RAKE fingers in or out of operation according to performance criteria is also investigated. If a given functional block is not needed due to favourable performance conditions then it could be switched out of operation thus saving power. The inherent ability of the variable tap length equaliser to switch taps in to or out of operation also presents scope for power optimisation and programmability.

1.3 Structure

The structure of this thesis is set out as follows:

- Chapter 2 presents a review of the research carried in the area of low power techniques and architectures. Low power techniques are reviewed specifically for the direct form FIR filter and the LMS adaptive filter.

- Chapter 3 presents an overview of RAKE receivers and LMMSE receiver architectures. It begins with an introduction to CDMA principles[4] and explains the motivation for wideband-CDMA communication systems. A review of the techniques proposed for reducing the switched capacitance of these multi-user receivers is also given.

- Chapter 4 presents the technique of using the existing decorrelating transform when applied to the LMS adaptive filter. An outline of its operation is given and the ability of this particular method being applied to the LMS adaptive filter to reduce power

consumption is presented. The implementation developed in this chapter is compared to a conventional implementation of an LMS adaptive filter.

- Chapter 5 presents a method of varying the tap length of the LMS adaptive filter. A practical implementation is developed with the aim of analysing the power consumption saving that applying this technique to the adaptive filter will produce. An existing length update algorithm controls the dynamic increase or decrease in the tap-length of the LMS adaptive filter. The aim is to remove unnecessary switching in the arithmetic blocks present in the various functional blocks.

- Chapter 6 presents the application of the variable length LMS adaptive filter in the implementation of a low power adaptive LMMSE receiver. Detail of the implementation of the new implementation is given. Operation of the new implementation is verified against a model of the system and power results are presented again in comparison with an equivalent conventional implementation of the same receiver.

- Chapter 7 presents the summary and outlines the conclusions of this thesis. Several areas for potential future research are also suggested.

- Appendix A presents the conference papers published and awaiting acceptance as a result of the work carried out in completion of this thesis.

- Appendix B shows the Matlab code written to model the DECOR LMS adaptive filter.

- Appendix C shows Verilog code for the main components in the DECOR LMS adaptive filter implementation.

- Appendix D shows the Matlab code written to model the Variable Length LMS adaptive filter.

- Appendix E shows the Verilog code for the main components in the Variable Length LMS adaptive filter implementation.

1.4 Summary

Research into low-power design is critical to the continuation of advancement in high complexity circuits and systems. The development of low-power processing blocks is a valuable contribution to efficient system design and integration. This thesis studies the techniques and architectures that can be used to reduce the power consumption in an adaptive LMMSE receiver core, the use of which is proposed in W-CDMA communication systems. In particular, the LMS adaptive filter block level design is studied and the application of low-power techniques to this functional block is investigated.

Chapter 2
Low Power Techniques

2.1 Introduction

This chapter explains the sources of power consumption in CMOS integrated circuits. It gives a review of the algorithmic methods employed to reduce power consumption by minimising switched capacitance and an overview of the architectural techniques used. Low power techniques and architectures for FIR filters are then discussed followed by that of the low power techniques used in adaptive filters. The following chapters present the application of one or more of these techniques in the HDL implementation of critical functional blocks within the proposed architectures.

2.2 Power Consumption in CMOS technology

Switching power in CMOS circuits accounts for around 80% [5] of the total power consumption and is given by the relation:

$$P_{sw} = \frac{1}{2} S_w C_{load} V_{dd}^2 f$$

Where V_{dd} is the supply voltage, f is the clock frequency, C_{load} is the load capacitance of the gate and S_w is the switching activity factor. S_w is defined as the average number of times that the gate makes a logic transition (1 - 0 or 0 - 1) in each clock cycle. The product $S_w C_{load}$ is defined as the switched capacitance.

It can therefore be seen that switching power P_{sw}, is directly proportional to each of these terms and proportional to the square of the supply voltage V_{dd}. A reduction of any of these terms will then result in a proportional reduction in switching power. Since the clock frequency f, and the supply voltage V_{dd} are generally restricted due to the platform technology or application of the chip, it is the switched capacitance that holds the greatest scope for study at the algorithmic and architectural levels of design, especially in memory and mathematically intensive DSP

applications. Leakage power reduction is also critical to the continued scaling of CMOS circuits and now accounts for an ever more significant proportion of IC power budgets. As CMOS design continues well into sub-micron gate lengths, leakage power is expected to dominate. This is being addressed by using new material and system design techniques such as the use of high dielectric gate materials. While switching reduction remains important, the progression towards sub-90nm technology presents significant challenges[6] to system design.

2.3 General low-power techniques

As the major power consumption in CMOS technology occurs during switching, it is essentially the minimisation of switching activity that allows computation to be achieved which will provide the overall reduction in power consumption. The following sections briefly summarise the methods that can be used to reduce the switched capacitance of circuit blocks at the algorithmic and architectural level.

2.3.1 Clock gating

Clock gating is used to disable unused modules within a system saving power by preventing unnecessary switching activity in functional logic blocks as well as eliminating power dissipation in the clock distribution circuitry[7].

2.3.2 Operation substitution

Commonly used operations that have alternative realisations may offer power efficiency gains[8]. Provided there is not a serious compromise in overall performance, an alternative way to realise a given operation is beneficial.

2.3.3 Operation minimisation

Switched capacitance can be effectively reduced by minimising the number of operations carried out within an algorithm. This is normally achieved by a transformation of the algorithm itself such that it can be realised by using less hardware and therefore fewer gates, such as memory or arithmetic units[8].

2.3.4 Reducing glitching activity

Any node can undergo numerous transitions in a given clock cycle due to glitching before settling to a stable logic level, all of which consume power[9]. Glitching can be minimised in a number of ways such as balancing signal paths or retiming to compensate for different path delays[10] along with restructuring multiplexer networks or clocking control signals[11]. In [12] a technique for glitch minimisation in combinational circuits is presented. Here, the total number of glitches is reduced by replacing some existing gates with functionally equivalent gates, called F-Gates, that can be 'frozen' by asserting a control signal. A frozen gate will not propagate glitches to its output. Further to this, glitch elimination by gate freezing, gate sizing and buffer insertion is presented in [13]. The proposed method unifies gate freezing, gate sizing and buffer insertion into a single optimisation process to maximise the glitch reduction.

2.3.5 Input and constant coefficient ordering

Switched capacitance can be reduced by reordering the inputs in a chain of operations such that the higher activity inputs enter the chain at a later stage[8]. A power saving can be made for example if constant coefficients are ordered according to their Hamming distance in certain architectures.

2.3.6 Pre-computation

By selectively pre-computing the output logic values of a circuit one clock cycle before they are required and using these pre-computed values gives a reduction in the internal switching activity during the succeeding clock cycle[14, 15]. The output for a subset of input conditions is calculated allowing the original circuit to be switched off in the subsequent clock cycle. The trade-off between the size and power dissipation of the pre-computation logic must be taken into account against the size and power dissipation of the original circuit.

2.3.7 Data representation

The digital representation of data has an effect on the switching activity. In twos complement representation for example, the signal transition from positive to negative or vice-versa, causes the MSB sign-bits to switch, resulting in high switching activity. Switching activity is therefore

very high when the signals being processed switch around zero[10]. The data representation used can also be chosen to yield power savings in certain applications like speech coding or image processing.

2.3.8 Bus encoding

Coding of data to be transmitted on a bus can be employed, depending upon the application, to reduce switching activity. Gray coding for example is employed to sequential and highly correlated data as an alternative to standard binary coded addressing[16, 17]. Other types of coding may also be used such as Bus-invert coding which uses a control bit which is set according to the Hamming distance between successive bus values.

2.3.9 Wordlength reduction

The number of bits, or wordlength, used in data and for addressing is critical to a design when considering speed, area and power[8]. If the number of bits in a binary word can be reduced, the switching activity will be reduced accordingly. This therefore reduces the switched capacitance. Shorter binary words also result in fewer bus lines and decrease the average interconnect length and capacitance.

2.3.10 State assignment

State assignment methods have been presented[18] which minimise the number of bit changes during state transitions in FSMs. The probability of state transitions is calculated using the given input switching probability and used to find an encoding that minimises the switching probability of the state variables.

2.3.11 Scheduling and resource binding

Algorithms have been proposed[19] to minimise the number of transitions on the signals applied to functional units i.e. adders, multipliers, muxes and registers etc. which effectively minimise the switched capacitance. Scheduling is used to bind nodes to the same resource with the candidate nodes being selected such that there are no change in values of the operands between consecutive operations of the same functional unit.

2.3.12 Memory partitioning

The memory in a system can be partitioned such that highly accessed locations are mapped to a small and efficient memory block[16]. Average power is decreased because a large proportion of accesses are concentrated on the most power efficient memory areas. At the same time, switching activity in other memory blocks can be reduced by disabling them in a given clock cycle if they are not addressed.

2.3.13 Selection of appropriate gate level implementation

Switching activity is also dependent upon the gate-level layout that different realisations of the same function can have. The simulation of different logic arrangements for the same functional block allows the switching activity to be evaluated according to performance. When applied to arithmetic units such as adders[20] or multipliers[21, 22] for example, the best construction can be chosen according to desired performance characteristics, such as relative throughput or area, against the reduction in switching activity achieved.

2.3.14 Physical capacitance reduction

Switched capacitance is directly proportional to the physical capacitance being switched. Switched capacitance can be reduced by reducing any of the capacitances present in CMOS technology. This may involve minimising gate count, using smaller devices and shorter/fewer interconnects which can be carried out by logic minimisation or gate re-sizing and by using appropriate placement, partitioning and wire sizing[9].

2.3.15 Technology decomposition and mapping

There are many possible circuit level implementations for every gate level implementation each having its own switching behaviour. Technology decomposition and mapping techniques have been proposed for minimising the switching activity in Boolean networks[23, 24].

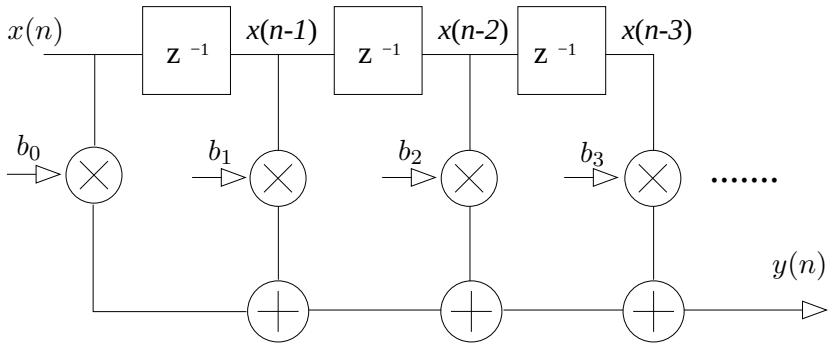


Figure 2.1: *Block Diagram of Conventional DF FIR Filter*

2.4 FIR Filters and Low Power Techniques for FIR Filter Architectures

A filter is described as any system which alters the characteristics of a signal to enhance desired signal components and/or to suppress unwanted signal components. In communication systems filters are used to extract information from a signal or to selectively separate signal components that have previously been combined to provide efficient transmission through a channel. A digital filter can achieve almost any filtering effect that can be represented by a mathematical algorithm and can be implemented in either hardware or software according to overall system constraints.

The Finite Impulse Response (FIR) digital filter is fundamental to signal processing theory and is widely used in many applications. The *N*-tap FIR filter is represented by the following convolution equation:

$$y(n) = \sum_{i=0}^{N-1} b_i x(n-i) \tag{2.1}$$

where b_i's are the filter coefficients, $x(n)$ and $y(n)$ are the nth terms of the input and output sequences, respectively. The direct form FIR filter is shown in Fig.2.1.

The response of the FIR filter to an impulse will always ultimately settle to zero. From the diagram of Fig.2.1 it can be seen that in this Direct Form (DF) the filter is non-recursive meaning that the present $y(n)$ filter output is dependent only on the present $x(n)$ and $x(n-N)$ previous inputs and not previous output values. It is this that ensures that the FIR filter is inherently stable. In it's direct form the FIR filter is simply a delay line made up of a series

of discrete delay elements (z^{-1}), the output of each being multiplied by it's respective filter coefficient, referred to as a filter tap. An *N*-tap FIR filter utilises N-1 delayed samples and *N* multipliers with the filter order being directly related to the number of taps present. Therefore as the number of taps is increased the FIR filter response approaches the ideal filter response. The output $y(n)$ is then calculated by performing the summation of every coefficient multiplier output.

The following sections briefly summarise basic principles of methods that can be used to reduce the switched capacitance of FIR filter architectures.

2.4.1 Coefficient ordering

By reordering the FIR filter coefficients it is possible to reduce the number of logic transitions between successive multiplication thus reducing the overall power consumption. Analysis of the minimum Hamming distance between coefficients[25, 26] for example will minimise the switching transitions at the multiplier input. The ordering of coefficients in this way is however computationally complex for practical size filters and the process of determining this ordering will require heuristic searching or the use of a genetic algorithm.

2.4.2 Coefficient segmentation

Here, individual filter coefficients are decomposed into two primitive sub-components[27] such that one part is produced which can be implemented using a single shift operation leaving the other part, which has reduced wordlength, being applied to the coefficient input of the multiplier unit. This reduction in wordlength results in significantly lower switching activity in the multiplier and therefore reduces the power consumed. Examples of this in practical FIR filters have shown a power saving of up to 63% in the multiplier.

2.4.3 Block processing

A major source of power consumption in digital signal processing is transition activity in multiplier units. In the direct form FIR filter, each delayed input data sample is multiplied by the appropriate filter coefficient at every clock cycle and accumulated in a multiply and accumulate unit. In this situation switching activity is high due to the inputs of the multiplier

unit being presented with new data upon every clock cycle. Power consumption will be directly reduced by any technique that reduces this switching activity. Along with this, another source of power consumption is transition activity on data and address buses. Every time a data sample is multiplied by a filter coefficient, the data and address buses encounter a high switching activity. Due to the fact that bus capacitancies are typically several orders of magnitude greater than that of internal circuit gates, the power consumed is large. A considerable power saving can be achieved if filter outputs are processed in blocks[28]. This will reduce transitions on the data and address buses and also ensure that data at the input to multipliers is retained for more than one clock cycle.

2.4.4 Approximate processing

In general, adaptive filtering algorithms are concerned with dynamically changing the values of the filter coefficients only and do not alter the order of the filter. In approximate processing, the algorithm dynamically adjusts the order of the FIR filter in accordance with the stop-band energy of the input data signal[29]. Filtering solutions are therefore reached in which the stop-band energy of the filter output signal may be kept below a given threshold while using the lowest filter order possible. As it can be shown that filter order is directly proportional to power consumption, this technique achieves a power reduction when compared to a fixed order filter whose output is similarly specified to have a stop-band energy below the given threshold. The filter order is dynamically adjusted by observing the energy of the input signals' stop-band component. When this increases it is necessary to increase the stop-band attenuation of the filter by increasing the filter order. Likewise, if the stop-band energy decreases the filter order is reduced.

2.4.5 Multi-rate architectures

Computationally efficient multi-rate architectures for FIR filter implementation have been proposed [30]. These involve the implementation of FIR filters in terms of decimated sub-filters. Winograd's algorithms are used to reduce computational complexity of polynomial multiplication in the determination of the filter. For a DF *N*-tap filter structure, *N* multiplications and $N-1$ additions per output are required whereas in the multi-rate architecture, $3N/4$ multiplications and $(3N+2)/4$ additions are required per output sample. This reduced computational complexity in multi-rate architectures allows the reduction of

clock frequency and supply voltage while achieving the same throughput as the conventional DF FIR filter, hence significantly reducing power consumed.

2.4.6 Coefficient scaling and optimisation

Coefficient scaling preserves the filter characteristics in terms of pass-band ripple and stop-band attenuation but does result in an overall magnitude gain equal to the scaling factor. Coefficients are scaled in the first stage such that the total Hamming distance between successive scaled coefficients is minimised. This is followed by modifying the scaled coefficients in the second stage such that the total Hamming distance is reduced while maintaining filter characteristics. This is an iterative process and continues until no further reduction in Hamming distance is achieved[31].

2.4.7 Filter realisation through differential coefficient

Typically, the FIR filter coefficients are used directly to calculate the convolution with the input data. The differential coefficient technique involves the use of various orders of differences between adjacent coefficients along with stored intermediate results to compute the output convolution result[32]. Although the memory requirement and number of memory accesses is increased in this technique compared to a conventional FIR filter, the net computation required per convolution is reduced. The result is a net power reduction in the multiplier unit due to the lower magnitude of coefficients presented to it. This technique relies on the differences in value between adjacent coefficients being small when compared to the values of the coefficients themselves. If this is satisfied then the multiplier selected for use can be implemented with reduced wordlength. When the power saving in the multiplication is greater than the cost due to memory overhead then a net power saving is achieved.

2.4.8 Reduced two's complement data representation

A reduced two's complement representation for numbers has been proposed which avoids the use of sign-extension and therefore the switching of the sign-extended bits[33]. The maximum two's complement magnitude of a number is detected and its reduced representation is generated to represent the signal. A constant error is introduced by this reduced representation although it is however compensated for. This technique is more useful for filters in which the

coefficients are correlated and have small magnitude.

2.4.9 Sharing multiplication

This technique is based on the development of a computation sharing multiplier which targets the reduction of redundant computations in FIR filtering[34]. In vector-scalar product operations, sets of short bit sequences are identified such that the multiplication result can then be obtained by using only add and shift operations. These sets are chosen such that the full collection of small sets spans the entire coefficient vector. Using a transposed DF (TDF) filter implementation, a so called pre-computer block contains a set of multipliers used for the multiplication of input data vector with the short bit sequences, representing the coefficient vector. *N* select/shift units, *N* being equal to the number of taps in the filter, and an adder are used to calculate the final output. The sharing of this pre-computer block containing all the multipliers leads to the power saving achieved in this architecture.

2.4.10 Algorithmic Low Power Cores

The authors Erdogan, Hasan and Arslan in [35] present novel architectures for low power FIR filter cores using algorithms that minimise the switched capacitance in the multiplier and on data buses. This is carried out by exploiting data and coefficient correlation within the device. These algorithms are mapped onto the arithmetic processing units and the description of the overall FIR architecture is given.

Initially, two algorithms are outlined. One detailing the implementation of a coefficient segmentation algorithm whereby a 16-bit filter coefficient is segmented into two numbers for computation. The other details a block processing algorithm which reduces switching activity by processing data in blocks. Switching activity is reduced by holding data constant at component inputs for two clock cycles rather than only one. A third proposal is then made of a combination of the two algorithms which results in a compounded saving in power consumption.

It is stated that the coefficient segmentation algorithm yields a 22% power saving with an circuit area overhead of 4% while the block-processing core yields a power reduction of 26% with a 3% increase in area. Combining the two methods of coefficient segmentation and block processing results in a 39% power reduction at the expense of a 7% increase in area.

2.5 Adaptive Filters and Low-Power Techniques for Adaptive Filter Architectures

Linear equalisers are typically implemented using adaptive finite impulse response (FIR) filters [1] with filter coefficients being recursively updated using either the recursive least squares (RLS) or more commonly the least mean squares (LMS) algorithm, as is used in this study. The conventional adaptive filter is typically made up of two functional blocks. The weight update (WUD) block and an FIR filter block which is modified to accept updated filter coefficients from the WUD block used in calculation of the filter output.

Taking the LMS algorithm, the weight update equation for the adaptive LMS filter is

$$b_i(n+1) = b_i(n) + \mu e(n) x(n-k) \tag{2.2}$$

where μ is the step size and $e(n)$ is the adaptation error given by

$$e(n) = d(n) - y(n) \tag{2.3}$$

where $d(n)$ is the desired output $y(n)$ of the filter. This computation updates the filter coefficients b_i seen in the FIR filter equation (2.1).

The number of taps in the FIR structure has a critical influence on the performance and computational complexity of the equaliser. An equaliser with too many taps will be computationally inefficient and may introduce a degradation in mean squared error (MSE) performance due to limitations of the LMS algorithm whereas, an equaliser with too few taps will be unlikely to reach its true potential level of distortion mitigation. Coupled with this is the time variant nature of wireless channels which ideally necessitates the ability of the equaliser to alter its number of taps with time.

The following sections briefly summarise basic principles of methods that can be used to reduce the switched capacitance of adaptive filter architectures.

2.5.1 Low Power Multiplication in FIR Filters

In a paper by Nicol and Larsson[36] a FIR filter is described which uses Booth encoded multiplier blocks. It is stated that by selecting the correct multiplier configuration for a given application a power reduction of 50% or more is achieved depending upon the filter response. In the Booth encoded multiplier the number of partial products is halved thus reducing the number of bit transitions in the operation and reduces delay in the circuit.

In this particular case each tap in the FIR filter is initially assigned a multiplier and in a subsequent design the multipliers are time-multiplexed to compute a number of filter coefficients thus reducing the number of multipliers used. It is noted that the power saving achieved it dependent on the precision of the filter coefficients and optimising these coefficients results in a further power saving. The concepts here are said to be easily translated for use in an adaptive filter however no details are explained.

2.5.2 Variable Length Equalisers

In [37] the authors Riera-Palou, Noras and Cruickshank present the concept of the variable length equaliser. The motivation for this being that the control over length, being number of filter taps, can improve performance and will also reduce power consumption. Here the investigation of variable length is carried out on linear equalisers using the least mean squares (LMS) algorithm for updating filter coefficients.

Following this, the same authors present a simple method for dynamically adjusting the number of taps in a linear equaliser to suit channel conditions[38]. This is based on the use of segmented filters where equalisation is carried out by splitting a FIR filter structure into concatenated sub-filters each with their own outputs. This structure affords the use of extra information provided by the multiple equaliser outputs. These outputs are used to compute a corresponding error signal which is used to obtain an output MSE value for each segment. The performance of each segment can then be evaluated and used to control the number of active segments.

Further to this [39] and [40] present alternative methods for the variation of equaliser length. In [39] an algorithm is described which is derived from the stochastic gradient (SG) algorithm and modified to allow dynamic allocation of filter coefficients. In this the order of the filter along with the adaptation step size are changed automatically according to a certain level of

performance. The benefits of this are both fast convergence and good steady state performance which, in normal cases are traded off against one another. In [40] variable LMS algorithms are presented based on the time constant concept in which the step size is changed (VS-LMS algorithm) and filter length is varied (VL-LMS algorithm). The time constant concept relates to the change in step size or filter length from one iteration to the next. For example, in the VL-LMS algorithm the filter length is kept small to aid fast convergence then increased upon each subsequent iteration to provide better steady-state performance.

2.6 Summary

This chapter has described general and functional block specific techniques currently proposed for reducing the switched capacitance in CMOS circuitry. These techniques aim to optimise the circuit operation or to exploit certain properties of a conventional implementation method. This might involve direct optimisation of hardware area through some sort of transform, or by optimisation at the algorithmic level using scheduling or pre-computation for example. Alternatively, properties such as redundancy or correlation in the signals present on buses and used for computation are exploited to reduce switched capacitance. This thesis will present the low power techniques which can be applied to the LMS adaptive filter and then go on to explore the RAKE and LMMSE receiver structures.

Chapter 3
RAKE Receiver and LMMSE Receiver Architectures

3.1 Introduction

Spectrum availability is a growing problem, especially in metropolitan areas, and is a major factor in the design of wireless communication systems. Advanced techniques are needed to cater for the requirements of operators who need to supply better services to users while being constrained by a finite frequency spectrum. New receiver techniques have been proposed which offer the ability to increase the capacity of wireless networks. In this thesis the architecture of one such receiver is studied which is based on an emerging wideband code-division multiple access (W-CDMA) technique. This technique offers the multi-user interference suppression and cancellation required to give the desired increases in network capacity.

This chapter describes some of the background principles and techniques used to increase the capacity of wireless networks. An introduction into the principles of operation is given and the new techniques proposed to counter their limitations are outlined, namely code-division multiple access and wideband code-division multiple access techniques and principles. Finally, this chapter presents a background overview of the techniques proposed for RAKE and LMMSE receivers and the novel architectures employed in their design and low power techniques that can be used to reduce switched capacitance.

3.2 Code-Division Multiple Access

Users can be separated in a number of ways, normally either in frequency, such as in frequency-division multiple access (FDMA) or in time as in time-division multiple access (TDMA). In both of these schemes, the maximum number of users is limited by the number of orthogonal time or frequency slots available. Code-division multiple access (CDMA) systems differ from this in that all users share the same frequency band at all times with separation of

users being achieved by assigning each user with a unique and specific spreading sequence onto which the transmitted data is modulated[4]. This is in contrast to FDMA or TDMA where there exists a frequency or time separation only. Provided there are enough exclusively unique spreading codes available for adequate user separation then code domain separation will result in an marked increase in channel capacity.

CDMA employs spread spectrum (SS) techniques to separate different users, the most common methods of which used in practical spread spectrum communications systems are direct-sequence (DS) and frequency hopping (FH). Spread spectrum techniques require that the transmission signal occupies a bandwidth greater than that of the minimum required to send the information. In frequency hopping spread spectrum systems the transmission frequency is altered at regular intervals according to the spreading code which is normally a pseudo-random noise (PN) sequence. This spreading sequence must ensure that no two users are able to transmit on the same frequency simultaneously. In a direct-sequence spread spectrum system, a high frequency pseudo-random sequence modulates the data stream. The high frequency spreading sequence will have a far greater spectrum bandwidth than the data signal, this relationship being controlled by the spreading factor. The same PN code is then used at the receiver for de-spreading and recovery of the desired data signal. It follows then that the receiver must be synchronised with the transmitter. De-spreading is carried out in the receiver by correlating the incoming received signal with the synchronised PN code replicated by the receiver. The processing gain of the system is defined as the ratio of the bandwidth of the spread spectrum signal to the spectrum bandwidth of the original data signal. In commercial systems, direct sequence (DS) systems are the favoured option.

In CDMA systems, the data signal $d(t)$ is modulated by the high frequency PN spreading code $c(t)$ which is made up of +1 and -1 signal level 'chips' as shown in the diagram of Fig.3.1. In this case this modulation is simply a modulo-2 addition and also acts as a phase modulator. CDMA code sequences have similar statistical signal properties to sampled white noise and are generated by a linear feedback shift register (LFSR). The processing gain (G) in DS-CDMA is defined as the ratio of the CDMA code frequency $(1/T_c)$ to that of the data frequency $(1/T)$. The spread signal $s(t)$ now has no resemblance to the original data sequence $d(t)$ and is therefore impossible to recover from the transmitted signal without the knowledge of the PN spreading code sequence $c(t)$. The bandwidth of the spread spectrum signal is now G times that of the data signal $d(t)$. The processing gain G also gives an indication of the amount

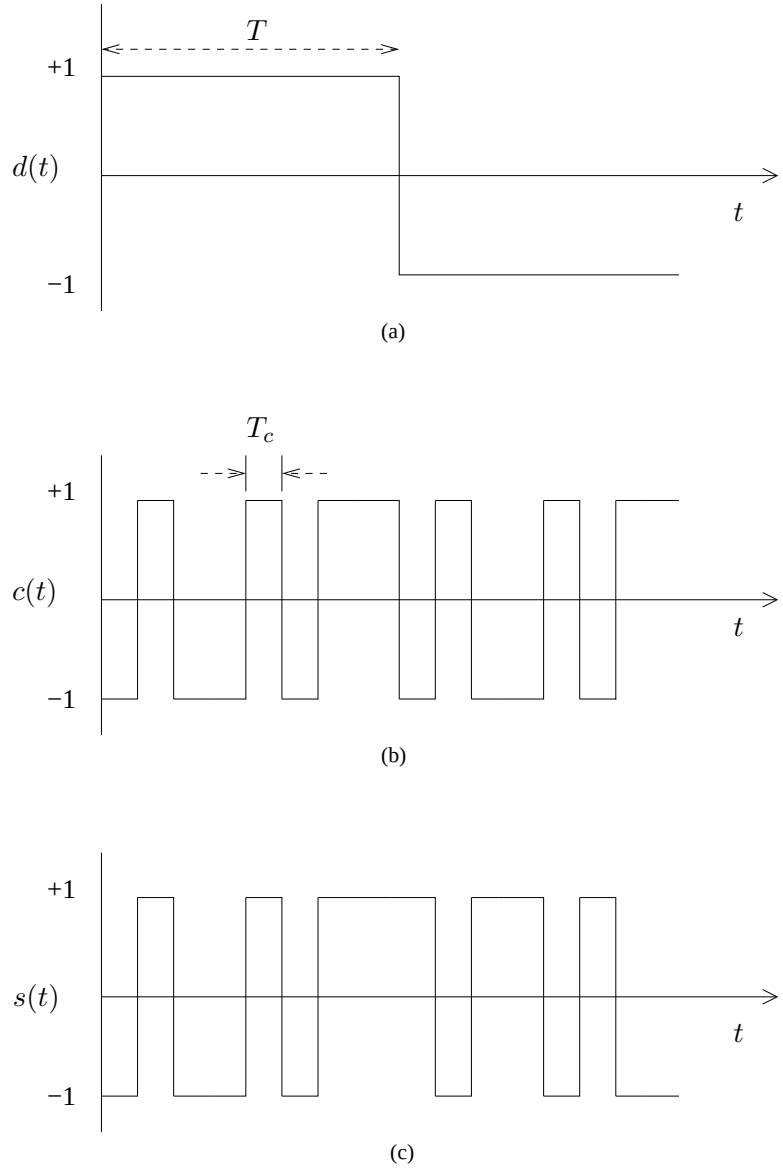


Figure 3.1: *a) Data signal b) PN code signal and c) Spread spectrum signal*

of interference protection provided by the PN spreading code as it is a measure of the amount by which the power in the data signal is spread over the entire transmitted bandwidth.

The PN sequences chosen should have an impulse-like auto-correlation response for detection purposes, and a low cross-correlation with other spreading codes to maximise separation between users. The data is recovered at the receiver by correlating the incoming received SS signal with a locally generated replica of the PN code. The receiver must transform this wide-band signal into the original signal bandwidth using the correlation properties of the PN code. If ideal transmission is assumed and the incoming PN code and receiver PN code are

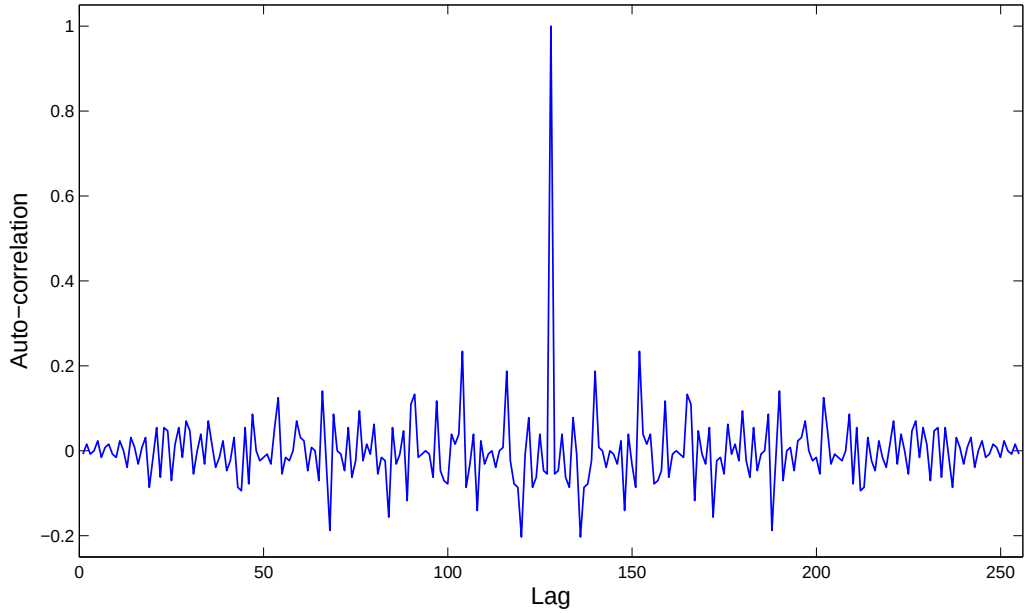


Figure 3.2: *Auto-correlation function of 128chip PN sequence.*

identical, then the correlator will output a positive peak. If the incoming PN code is 180° out of phase with the receiver PN code then a negative peak will be output. Since each user has a unique orthogonal spreading code, the SS signals of the other users are suppressed by correlator. However, each additional user does increase the overall noise level in the system.

The auto-correlation response of the PN sequence is important to allow multi-path signal components to be more easily resolved then combined in a RAKE type receiver. This type of receiver consists of a bank of correlators which receive several multi-path components simultaneously. The auto-correlation response is of even greater importance in frequency-selective fading channels [41]. An example of the auto-correlation response of a 128 chip PN sequence can be seen in Fig. 3.2. The RAKE receiver is designed to counter the effects of multi-path fading, doing so by resolving the time-delayed signal in a number of receiver fingers. Each finger de-spreads the incoming received signal by correlating it with a locally replicated versions of the known PN codes, this correlation of the PN sequence with the incoming received sequence is then output from each finger and combined to retrieve the channel compensated symbols. A functional diagram can be seen in Fig.3.3.

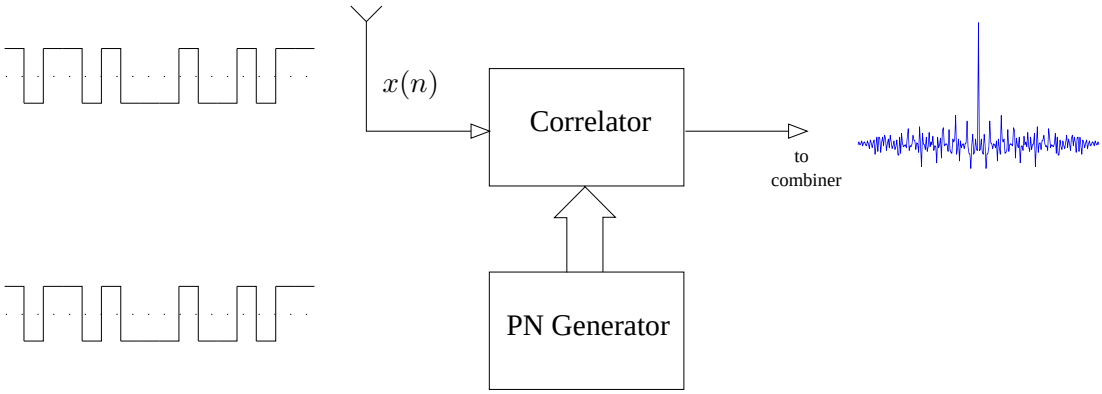


Figure 3.3: *Functional Diagram of RAKE Finger.*

Interference from other users is limited by the low cross-correlation of the PN sequence. The level of interference however is maintained by careful power control, this being a critical requirement for direct-sequence CDMA. The phenomenon of the near-far effect is introduced by inaccurate power control resulting in a weak user's signal being buried in the multiple-access interference (MAI) by the stronger signal of another user. The conventional single-user receivers are highly sensitive to the amount of MAI present with serious degradation resulting from small amounts of MAI[42]. It is this problem that has led to the effort into the development of enhanced receivers with the introduction of multi-user interference suppression receivers and cancellation receivers which exploit the nature of the interference itself to improve performance and increase channel capacity[42].

For single-user communication systems, the optimal receiver in multi-path channels which cause inter-symbol interference is the maximum likelihood sequence detector (MLSD)[2], which requires that the channel is known. In practice however, the channel must of course be estimated. Correlator receivers are the most simple of the sub-optimal single user DS systems[43]. The most widely used receiver in CDMA systems is the RAKE receiver[41]. RAKE receivers are traditionally used in CDMA systems where the spreading factor is large and therefore cross correlation between codes is low.

In CDMA systems it is conventional to neglect MAI and the near-far effect. This however does place limits on the capacity of the system in question. As optimal receivers are impractical in terms of implementation, several methods for sub-optimal receivers have been proposed[44–46]. In the downlink receiver only the desired signal is intended to be demodulated while the interference from other users is suppressed.

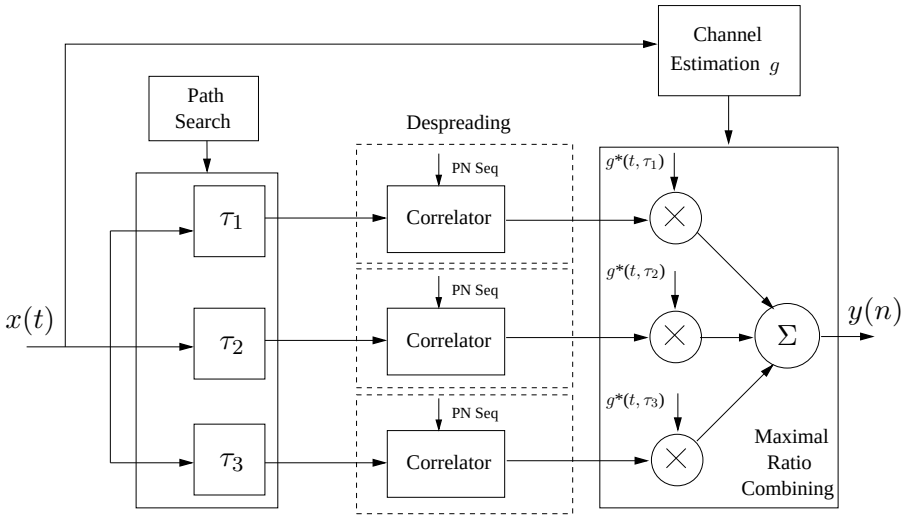


Figure 3.4: *Functional Diagram of RAKE Receiver showing Maximal Ratio Combining.*

Existing methods used for the reception of signals in wireless channels and their architectures are constantly being investigated as data rates and receiver complexity increases. Most wireless receivers employ a CDMA baseband processor which includes a correlator array and a DSP core[47]. Correlators are used for the de-spreading of received signals and are important components in RAKE receivers which capture the energy of multiple individual signal paths. Several multi-path components are captured by a bank of correlators and combined to produce an optimum signal level for decision thresholding. The method of combination depends upon the application, however in third generation CDMA systems maximal ratio combining (MRC) is favoured[48]. Conceptually, the RAKE receiver is shown in Fig.3.4 with the received signals of the multi-path components being combined proportionally according to the channel estimation of the given path. The RAKE receiver fingers are shown with the dotted outline in the diagram.

3.3 Linear Minimum Mean Squared Error Receivers

The linear minimum mean squared error single-user receiver is one type of sub-optimal receiver which has been proposed[2]. The LMMSE receiver minimises the mean squared error between the receiver output and the original desired data signal. The advantage of this type of post-combining receiver is that it is capable of suppressing inter-path and inter-channel interference under severe near-far scenarios. The coefficients of the LMMSE receiver are dependent on the channel coefficients of all users and must be adapted dynamically as the channel changes. In a rapidly fading channel the LMMSE receiver must be adapted

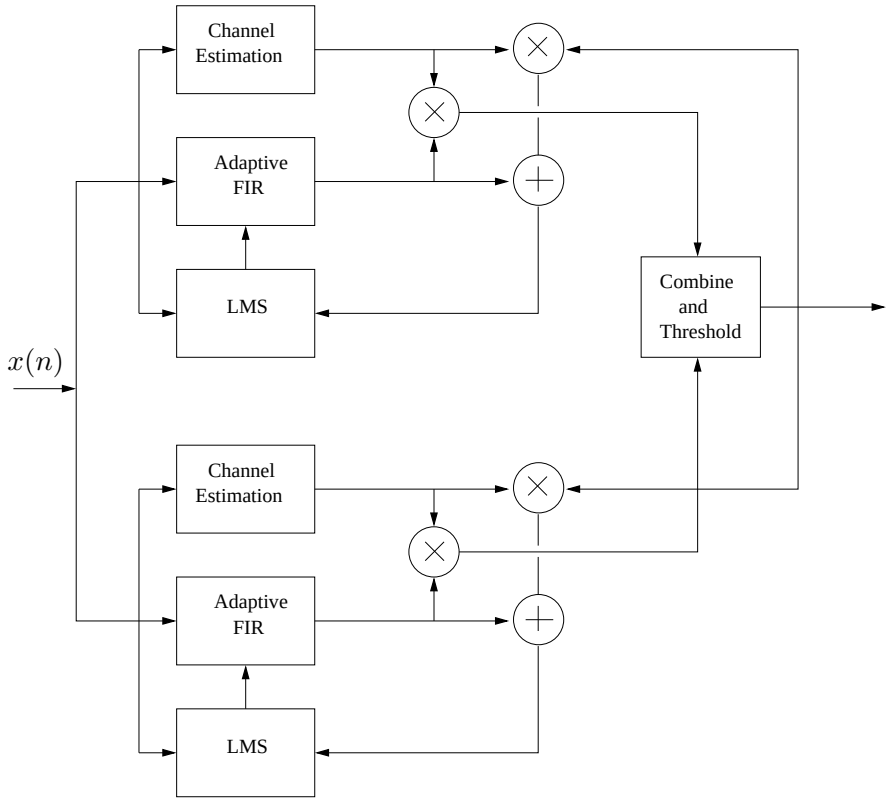


Figure 3.5: *Functional Diagram of LMMSE Receiver.*

continuously and will suffer convergence problems if the channel fades too fast. However the LMMSE receiver can still be used if the rate of fading is sufficiently low in relation to the data rate.

The pre-combining LMMSE receiver requires the knowledge of the spreading codes and the delays of all users. For this to be achieved, a computationally intensive direct matrix inversion calculation must be carried out. Added to this is the fact that all the spreading codes and delays for all users may not be known at the mobile terminal. It is for these reasons that the LMMSE receivers are usually solved iteratively for each user by using some adaptive algorithm such as the least mean squares (LMS) algorithm. However, the W-CDMA proposals are based on conventional RAKE receivers where there is no provision for a training sequence to be provided. The adaptive LMMSE-RAKE receiver can be implemented using the symbol decisions and the channel estimates of the conventional RAKE receiver, therefore no training sequence is required and the adaptive LMMSE-RAKE receiver can be considered to be blind.

Future W-CDMA techniques however offer methods of increasing the data rate available to users without increasing the spectrum bandwidth. This in turn means that the performance of the RAKE receiver is degraded as interference is introduced due to the mechanisms used. The capacity of the system is limited by multiple-access interference (MAI)[49]. Data rate can be increased by either assigning a number of parallel data channels with independent channel codes or by decreasing the spreading factor in a single channel, with each method introducing different interference problems.

The act of assigning multiple data channels results in the multi-path signals being subject to what is referred to as multi-channel interference or MCI. This behaves in the same way as the interference introduced by multiple users in conventional CDMA systems (MAI) and results in the degradation of performance of the RAKE receiver as the number of channels increases. Alternatively, decreasing the spreading factor in a multi-path channel will result in inter-path interference (IPI) which brings loss of diversity between signal path components. The continual decrease in spreading factor therefore results in increasing IPI[48]. Both the techniques detailed here are also sensitive to the near-far problem when a RAKE receiver is used. It can therefore be seen that in W-CDMA systems the performance of a RAKE receiver is limited by the data rate as performance is degraded when data rate is increased. It is therefore useful to investigate multi-user receiver techniques which enhance performance by combating the effects of these different forms of interference.

In this respect, a suitable solution lies in the linear minimum mean squared error or LMMSE receiver and is among the sub-optimal receivers proposed in [44],[45] and [46], optimal receivers being too complex for practical implementation. It's operation is based upon the principle of minimising the mean squared error between the receiver output and the desired transmitted data sequence and is capable of dealing with inter-path and inter-channel interference along with interference caused by near-far scenarios. As the operation of the LMMSE receiver relies on the channel coefficients of all users, the receiver coefficients must be adapted as the channel changes and must be updated continuously in fading channels.

Motivation for further work would be the implementation of a power efficient uplink receiver architecture comprising the LMMSE-RAKE (or adaptive-RAKE) structure that displays good convergence properties. The advanced nature of this receiver will obviously introduce a burden on power consumption therefore when used in a up-link mobile receiver, power optimisation is worthwhile.

3.4 Low Power Techniques for RAKE and LMMSE receiver architectures

It is said that in DSSS systems, the RAKE receiver being one of the key blocks is the most complex and dissipates a large proportion of the total power[50]. It is therefore important that the circuit complexity and power consumption of the RAKE receiver be minimised within the system. The pre-combining LMMSE receiver, or LMMSE-RAKE receiver, is based upon the structure of a conventional RAKE receiver. The following low-power techniques can therefore be directly applied to this type of LMMSE receiver.

3.4.1 Power-scalable RAKE core

In the scheme proposed by Bianco, Dassatti et al.[51] the CDMA receiver outlined is made up of a number of functional blocks, one of which being a RAKE receiver core. This core is made up of a number of RAKE fingers, an adder and a programmable comparator. Power optimisation is achieved by the use of a control unit which selectively turns on the adder according to the number of fingers being served. The control unit can also determine the number of fingers to be activated according to the output of the comparator and the confidence level that it determines based on the SNR. Re-configurability is based upon parameters such as input data width, RAKE adder width, multiplier pipe depth, adder pipe depth, maximum number of fingers and the number of thresholds in the comparator. Further re-configurable architectures are also presented in [52] and [53] and the effects of modifying parameters is studied.

3.4.2 Multi-code correlator array

In the work presented by Ku, Kuo, Chen and Chen [47] a low-power strategy is put forward which describes a correlator architecture for multi-code CDMA systems. In this, the input is de-spread with multiple PN sequences concurrently resulting in considerable power savings. This multi-code correlator architecture is based on the principle of computing partial correlation results which are stored in local registers, these partial results are then delivered to an adder/subtractor network from which the correlation results are output. Other low-power methodologies are also used to achieve a reduction in power consumption such as code grouping or transformation and clock gating in certain functional blocks. The dual-code variation of this multi-code architecture has proved optimal in terms of power consumption,

demonstrating a 41% reduction when compared to a conventional 2's complement single code correlator and 23% reduction when compared to a sign-magnitude single code correlator. Code numbers greater than three show diminished reductions in power. Two of the authors in [54] have progressed this work, using the tri-code correlator they have described to develop a programmable correlator array with a DSP architecture for use in 3G wireless communication applications[55]. This correlator array is reconfigured to cater for the separate code acquisition and code tracking phases during de-modulation. The functions of a chip-matched filter, used for synchronisation and the correlator bank used to determine code group information and determine scrambling code are configured in the code acquisition phase. In the code tracking phase a RAKE receiver is configured for the purpose of eliminating multi-path effects. This work demonstrates in detail the successful implementation and performance of the low-power programmable correlator core with the proposed DSP. No reference however is made to actual power consumption results here.

3.4.3 Shared component approach

The authors Lee and Ha [56], present a parallel operation technique in their RAKE receiver core. The architecture presented relies on the sharing of components between all fingers in the receiver. The first method described uses shared code generators to eliminate de-skew blocks and also allows the code generators to run at lower clock frequencies. The resultant reduction in circuit complexity reduces the total power consumption by 55.2% when compared with a conventional RAKE receiver with no shared components between fingers. A reduction in area is also reported without any degradation in performance. The authors then go on to demonstrate a further architecture in [56] whereby a single common parallel de-spreader is used to pre-compute symbols for RAKE fingers which allows each finger to operate at a lower clock frequency, processing multiple bits for each clock cycle. In this case, each code generator produces four bits in parallel with each de-spreader operating on four data items and computing 16 sub-symbols. The key principle of this architecture relies on the single de-spreader being shared by all fingers. This architecture demonstrates a reduction in power consumption of 37% when compared to a conventional RAKE receiver core.

The proposal by Lee and Kim[57] is also based upon the idea of component sharing. Their multi-finger structure uses shared arithmetic units and a pre-combining time de-skew buffer. The de-skew buffer takes demodulated symbol data from multiple fingers and adds this data to

previously stored data aligned to the same timing reference present in the buffer. The combined symbol data is then stored in the local registers before being presented to the combiner block. Effort here has been concentrated on the hardware 'cost', this will have power benefits but no results are provided and no direct reference is made to power consumption. A reduction in hardware complexity of 49.4% is achieved when compared with a conventional RAKE receiver.

3.5 Summary

This chapter has given an overview of CDMA and wideband CDMA techniques currently proposed for use in third generation mobile communications networks. The progression of the concepts used in RAKE and LMMSE receiver principles is also discussed. A description of low power techniques proposed for use in W-CDMA receivers is also given and how these low power techniques are then applied to the RAKE and LMMSE receiver structures. The following chapters in this thesis will present the application of low power techniques to the LMS adaptive filter and the LMMSE receiver.

Chapter 4
Low Power Differential Coefficient Technique in the LMS Adaptive Filter

This chapter investigates the use of the differential coefficient technique for reducing the power consumed in the HDL implementation of an LMS adaptive filter.

This chapter is organised into six sections. Section 4.1 introduces the motivation for the concept of differential coefficient methods and provides detail of accompanying research on the topic. Section 4.2 gives an overview of the decorrelating transform chosen for implementation with sections 4.3 and 4.4 describing the hardware architecture design. Section 4.6 details the power consumption results collected by the comparison made between the conventional and decor implementations of the LMS adaptive filter. It is important to note that 16-bit two's complement fixed point number representation is used in this chapter unless otherwise explicitly stated.

4.1 Differential coefficient methods

The implementation of complex systems which facilitate wireless communication requires the use of efficient and flexible cores in their design. These cores often involve the repetitive implementation of FIR filters and/or adaptive filters which include an FIR core[1]. In the implementation of FIR filters and thus the adaptive filter, there are two approaches, sequential and parallel. The parallel implementation can maximise throughput at the cost of considerable additional hardware such as adders and multipliers. On the other hand, the sequential implementation is cost and area-effective in hardware although however does suffer a bottleneck in throughput.

Power optimisation is a crucial part of FIR filter design with an ever increasing number of published techniques to reduce the power consumption of FIR filters. The authors in [58] optimise word-lengths of the input and output data samples and coefficient values. This involves the use of a general search based methodology which is based on statistical precision analysis

and the incorporation of cost/performance/power measures into an objective function through word-length parameterisation[59]. In [60], Mehendale et al. present an algorithm for optimising the coefficients of an FIR filter to reduce the power consumption in its implementation on a programmable DSP. The use of coefficient segmentation, block processing and combined segmentation and block processing algorithms for low power FIR filter implementations have been shown in [61]. High throughput FIR implementations have also been described by the authors in[62] and [63].

In most implementations of FIR filters the filter coefficients are used directly to compute the convolution with the input data. The differential coefficient method (DCM)[32] uses various orders of differences between coefficients along with stored intermediate results rather than the coefficients themselves in the computation of the convolution. If fewer bits are required to represent the differences compared to the actual coefficients, the size of the arithmetic unit in the filter can be reduced, hence reducing power consumption. This method does however have an overhead of $N-1$ additional latches (for storage of intermediate results) and $N-1$ additional adders (for addition of intermediate results) for an N tap filter. Although, greater orders of differences have smaller magnitudes, the overhead required by the DCM increases as the order of differences is increased. There is therefore a point beyond which the gains due to smaller coefficient magnitudes are less than the overall cost of overheads[32]. To minimise the overhead while retaining the benefit of DCM, differential coefficient and input method (DCIM)[64] and decorrelating (DECOR) transforms[65] have been proposed. Some of the advantages of DECOR over DCM are listed as (a) lower overheads for a given filter order, (b) overheads being independent of the filter order and (c) power savings over a wider range of filter bandwidths. It can also be seen in [65] that the DECOR transform is proposed for use in adaptive filtering.

4.2 The decorrelating transform

In this chapter the decorrelating transform (DECOR) has been chosen for its advantages over DCM and because of its proposed suitability for use in LMS adaptive filters. The overhead of the DECOR method is lower than that of the DCM method in hardware terms and also in storage terms.

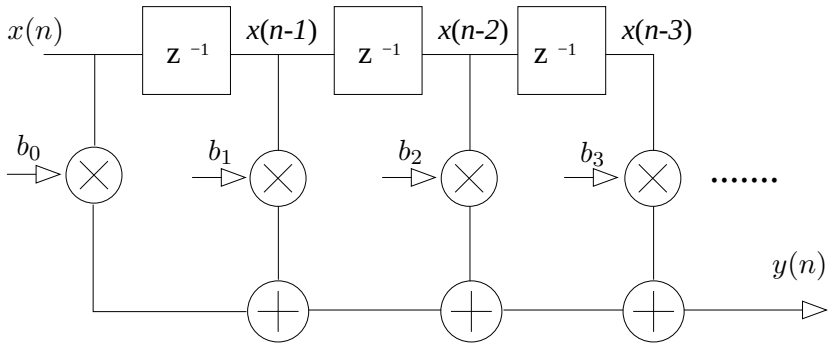


Figure 4.1: *Block Diagram of Conventional DF FIR Filter*

4.2.1 First-order decorrelating transform applied to FIR filters

As has been shown previously, the following convolution is performed by the *N*-tap FIR filter:

$$y(n) = \sum_{i=0}^{N-1} b_i x(n-i) \tag{4.1}$$

where b_i's are the filter coefficients, $x(n)$ and $y(n)$ are the nth terms of the input and output sequences, respectively. The direct form FIR filter is shown in Fig.4.1.

The z-transform of (4.1) is given as:

$$Y(z) = H(z)X(z) \tag{4.2}$$

where $Y(z)$, $H(z)$ and $X(z)$ are the z-transforms of the output, filter and input respectively. In DECOR, the transfer function $H(z)$ is multiplied and divided by the polynomial:

$$T(z) = (1 + z^{-1})^m \tag{4.3}$$

where m represents the order of coefficient difference. The z-transform of the output is therefore given as:

$$Y(z) = H(z)\frac{(1 + z^{-1})^m}{(1 + z^{-1})^m}X(z) \tag{4.4}$$

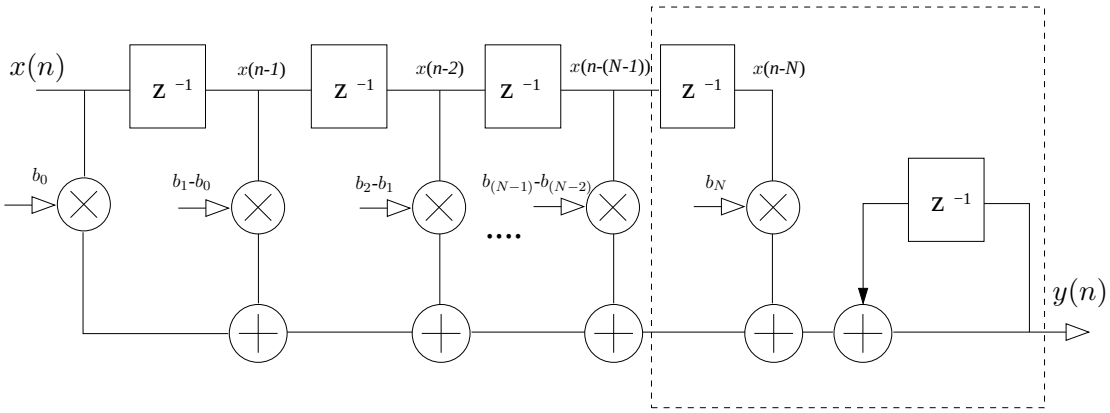


Figure 4.2: *Block Diagram of DECOR FIR Filter*

The frequency response of the filter is not altered by multiplying and dividing the transfer function $H(z)$ by this polynomial. For example, the z-transform of the first order low-pass FIR filter is given by:

$$Y(z) = \frac{(1 - z^{-1})H(z)}{1 - z^{-1}}X(z) \tag{4.5}$$

which when rearranged gives:

$$Y(z) - Y(z)z^{-1} = H(z)X(z) - H(z)X(z)z^{-1} \tag{4.6}$$

According to (4.1) and (4.6) the transformed filter can be expressed as:

$$y(n) - y(n-1) = \sum_{i=0}^{N-1} b_i x(n-i) - \sum_{i=0}^{N-1} b_i x(n-i-1) \tag{4.7}$$

Re-arranging (4.7) we can obtain the following equation for first order (m=1) differential coefficients:

$$y(n) = b_0 x(n) + \sum_{i=1}^{N-1}(b_i - b_{i-1})x(n-i) - b_{N-1}x(n-N) + y(n-1) \tag{4.8}$$

Clearly as (4.8) shows, for first order differential coefficients, the filter outputs can be obtained using the differences between adjacent coefficients (other than the first and last coefficients) and the previous filter output. The transformed filter also requires an additional multiplication and subtraction operation to realise the term $(-b_{N-1}x(n-N))$ in (4.8). Therefore, this together with adding the previous filter output represents the overhead for DECOR using first order differential coefficients. The DECOR FIR filter diagram can be seen in Fig.4.2 with the overhead section shown outlined. An example of the effect of the DECOR transform on the coefficient distribution of a low-pass FIR filter can be seen in Fig.4.3. This clearly shows the resultant reduction in magnitude of the filter coefficients. It can also be seen in equation (4.5) that an extra pole-zero pair is introduced to the original transfer function $H(z)$ at the position $z = 1$ in the z-plane. The z-plane plot in Fig.4.4 shows that of a conventional 72 tap FIR filter, with the plot in Fig.4.5 showing the resultant pole-zero plot of the DECOR transformed filter. The additional pole-zero pair at $z = 1$ on the unit circle can be seen. To ensure stability of the filter, the pole and zero introduced on the unit circle must cancel exactly. This additional pole-zero pair is a result of the DECOR transform introducing the feedback path, or recursive section, at the output stage. The numerator polynomial $(1 - z^{-1})H(z)$ creates the filter with differential coefficients while the denominator $(1 - z^{-1})$ results in the feedback path.

4.2.2 Second-order decorrelating transform equation

In the same way, it can be shown that the output of the second order (m=2) DECOR filter can be derived from the transfer function:

$$Y(z) = \frac{(1 - z^{-1})^2 H(z)}{(1 - z^{-1})^2} X(z) \tag{4.9}$$

with the output $y(n)$ expressed as:

$$\begin{aligned} y(n) = {} & b_0 x(n) + (b_1 - 2b_0)x(n-1) + \sum_{i=2}^{N-1} (b_i - 2b_{i-1} + b_{i-2})x(n-i) \\ & + (b_{N-2} - 2b_{N-1})x(n-N) + b_{N-1}x(n-N-1) \\ & + 2y(n-1) - y(n-2) \end{aligned} \tag{4.10}$$

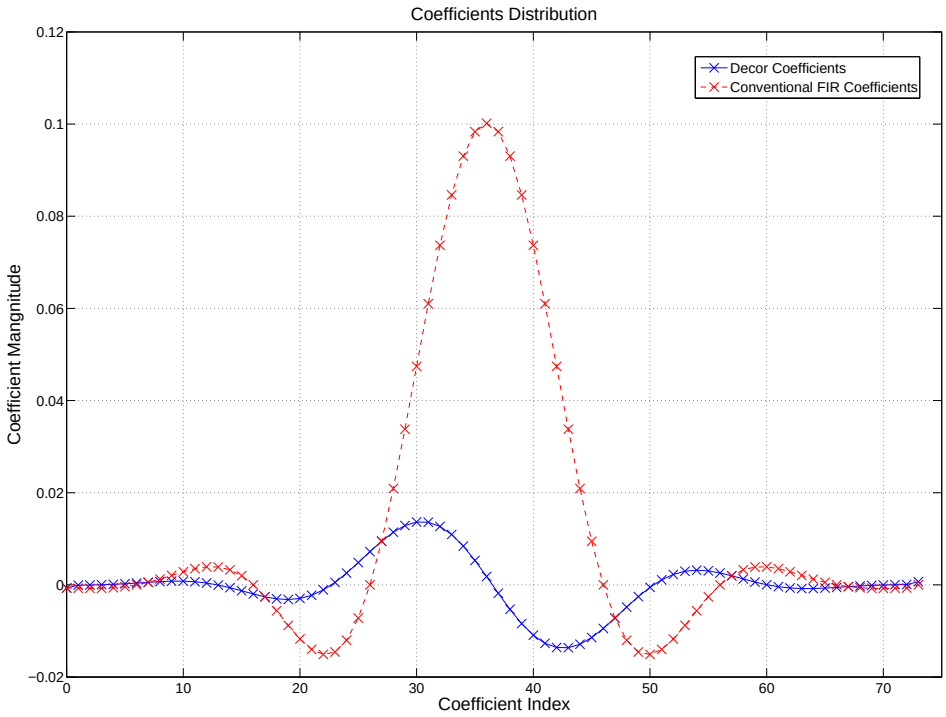


Figure 4.3: *Coefficient distribution plot*

However as the order of the DECOR filter m is increased so is the overhead needed. For the purposes of this chapter, only the first order transform will be implemented as it has been shown already for the DECOR FIR filter that the first order transform provides the best power reduction against the overhead incurred[66].

4.2.3 Modified DECOR transform

In the modified DECOR transform, the transfer function $H(z)$ is multiplied and divided by the polynomial:

$$T(z) = (1 + \alpha z^{-\beta})^m \tag{4.11}$$

where m represents the order of coefficient difference, α and β are parameters chosen depending on the type of FIR filter.

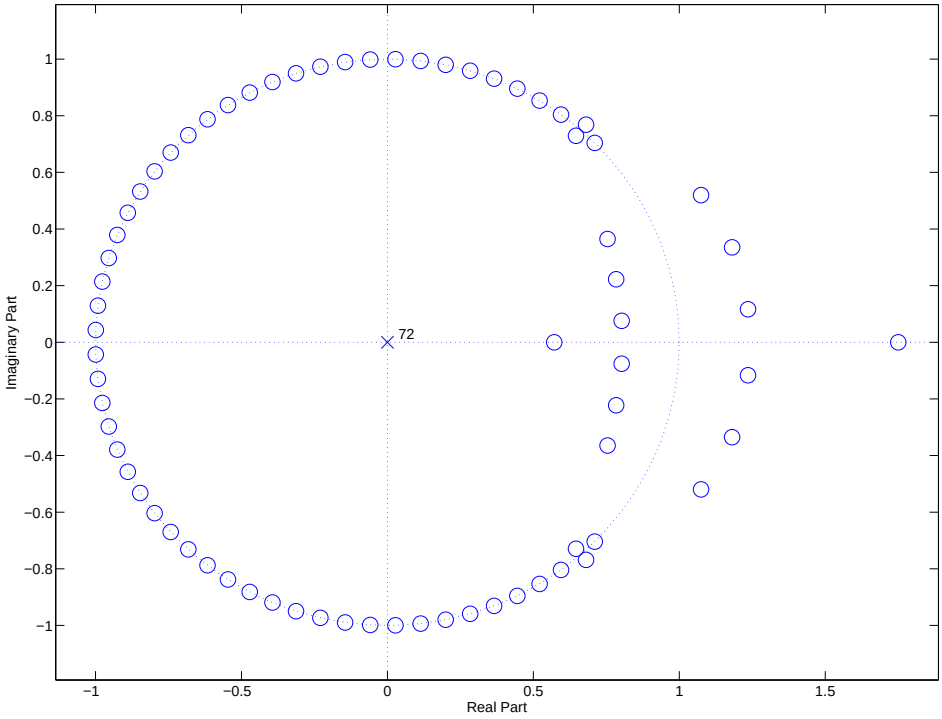


Figure 4.4: *Conventional FIR filter pole-zero plot*

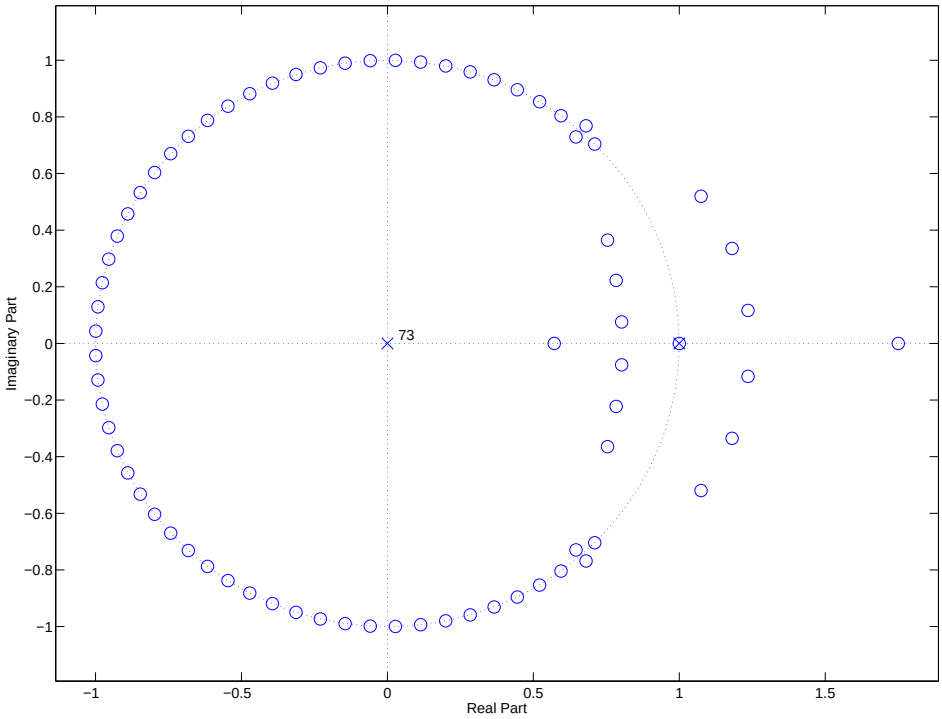


Figure 4.5: *DECOR filter pole-zero plot*

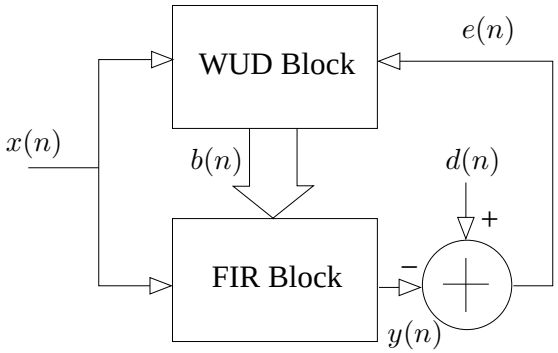


Figure 4.6: *Block Diagram of LMS Core*

The z-transform of the output is therefore given as:

$$Y(z) = H(z)\frac{(1+\alpha z^{-\beta})^m}{(1+\alpha z^{-\beta})^m}X(z) \tag{4.12}$$

Multiplying and dividing the transfer function by the polynomial $(1+\alpha z^{-\beta})^m$ therefore introduces m poles and m zeros at each of the β roots of $-\alpha$. If $|\alpha|$ is equal to 1 then all new poles and zeros lie on the unit circle. To guarantee stability all new poles must exactly cancel with the corresponding new zeros. If $|\alpha|$ does not equal 1 but $-1 \leq \alpha \leq 1$ then the DECOR transform will place a pole-zero pair on $z = \alpha$ inside the unit circle rather than $z = 1$. This therefore provides a solution to guarantee stability in the event that a fixed precision implementation results in a pole-zero pair not cancelling exactly.

4.2.4 The decorrelating transform applied to LMS adaptive filters

The conventional LMS adaptive filter is typically made up of two functional blocks. The weight update (WUD) block and an FIR filter block which accepts filter coefficients from the WUD block. The conventional adaptive filter block diagram is shown in Fig.4.6.

The weight update equation for a least-mean squares (LMS) filter is

$$b_i(n+1) = b_i(n) + \mu e(n)x(n-i) \tag{4.13}$$

where μ is the step size and $e(n)$ is the adaptation error given by

$$e(n) = d(n) - y(n) \tag{4.14}$$

where $d(n)$ is the desired output of the filter.

Applying the DECOR transform to an adaptive filter involves the derivation of the following from (4.1):

$$y(n) = -\alpha y(n-\beta) + \sum_{i=0}^{N+\beta-1} \delta_i(n)x(n-i) \tag{4.15}$$

where δ_i is the DECOR filter coefficient. The complete derivation of which is provided in [65]. The $\delta_i(n)$ DECOR coefficients are defined as:

$$\delta_i(n) = \begin{cases} b_i(n), & 0 \leq i < \beta \\ b_i(n) + \alpha b_{i-\beta}(n-\beta), & \beta \leq i < N \\ \alpha b_{i-\beta}(n-\beta), & N \leq i < N+\beta \end{cases} \tag{4.16}$$

Using this, the DECOR adaptive filter can be constructed. There are three blocks in the DECOR adaptive filter, one to compute each of the equations in (4.13), (4.15) and (4.16). These blocks being a conventional WUD block using equation (4.13), the DECOR block which calculates the decorrelated coefficients using (4.16) and the DECOR filter block employing the equation (4.15). The block diagram of the DECOR adaptive filter can be seen in Fig.4.7.

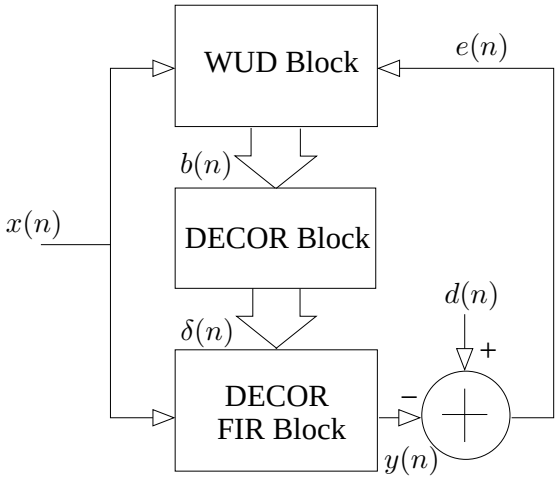


Figure 4.7: *Block Diagram of DECOR LMS Core*

In 4.16 it can be seen that the first β differential coefficients are the same as the original β coefficients calculated by the WUD block. The last β coefficients are the last β coefficients from the WUD block multiplied by α. The centre $N - \beta$ coefficients are either the sums or differences (dependent upon the value of α being 1 or -1) of the original coefficients.

4.3 Implementation of conventional LMS adaptive filter core

In order to evaluate the performance of the DECOR adaptive filter a conventional adaptive filter core based on the convolution equation of (4.1) and using the LMS algorithm with it's equations (4.13) and (4.14) was implemented. This conventional adaptive filter has two functional blocks, as seen in Fig.4.6. A weight update block (WUD), which uses the data input samples and an error signal to calculate new coefficients, and a filter block, which is generally an FIR filter and employs the coefficients calculated by the WUD block.

4.3.1 Conventional FIR filter block

A sequential FIR filter implementation was used to minimise hardware cost and reduce complexity at this point. Filter coefficients from the WUD block are clocked into the filter and presented to a single MAC unit which successively multiplies each delayed input sample with the appropriate filter coefficient and accumulates the results to provide the output sample y(n). A diagram of the FIR block structure can be seen in Fig.4.8. This filter implementation

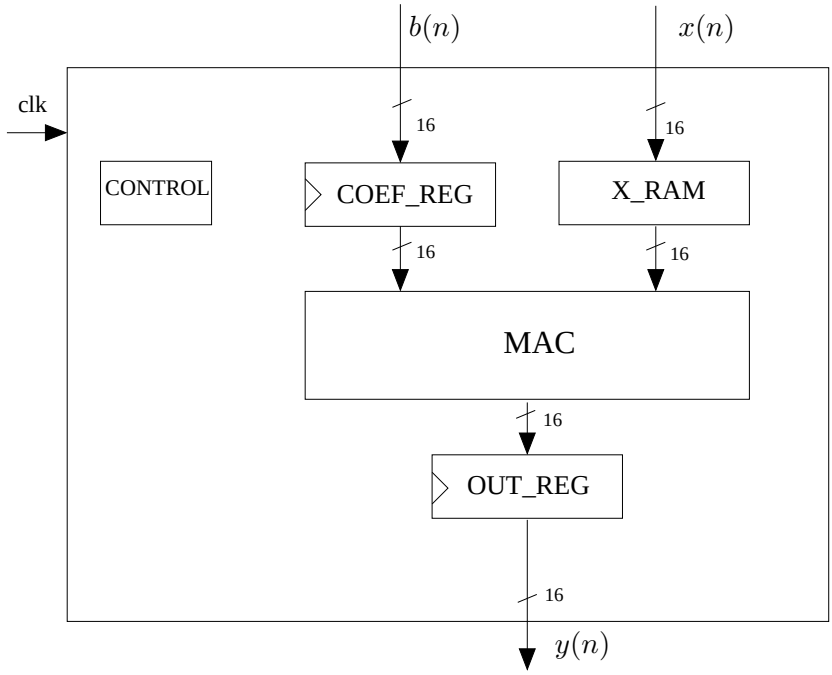


Figure 4.8: *Functional Diagram of FIR Filter Block.*

consists of a single memory block (X RAM) for storing input data $x(n)$, a control block (CONTROL), a number of single word registers for clock synchronisation and an arithmetic block (MAC). More detail about these blocks is given:

- X RAM: This is a RAM used for the storage of the input data $x(n)$. The memory bank is implemented as a latch based circular buffer of 16-bit registers to minimise power consumption. A demultiplexer at the input and a multiplexer at the output of the memory bank are used to address the memory locations and control read and write operation.
- CONTROL: A counter which provides addresses for the RAM blocks and is responsible for the synchronisation of activity for every block in the WUD unit by providing all necessary enable signals for the MAC and single registers.
- MAC: A conventional multiply-accumulate block consisting of a multiplier, an adder and a 32-bit flip-flop based register. A simple multiplexer is used to either feed the output back into the adder for accumulation or to clear the accumulator according to the status of an acc. enable control signal. A representation of the MAC block can be seen in the diagram of Fig.4.9.

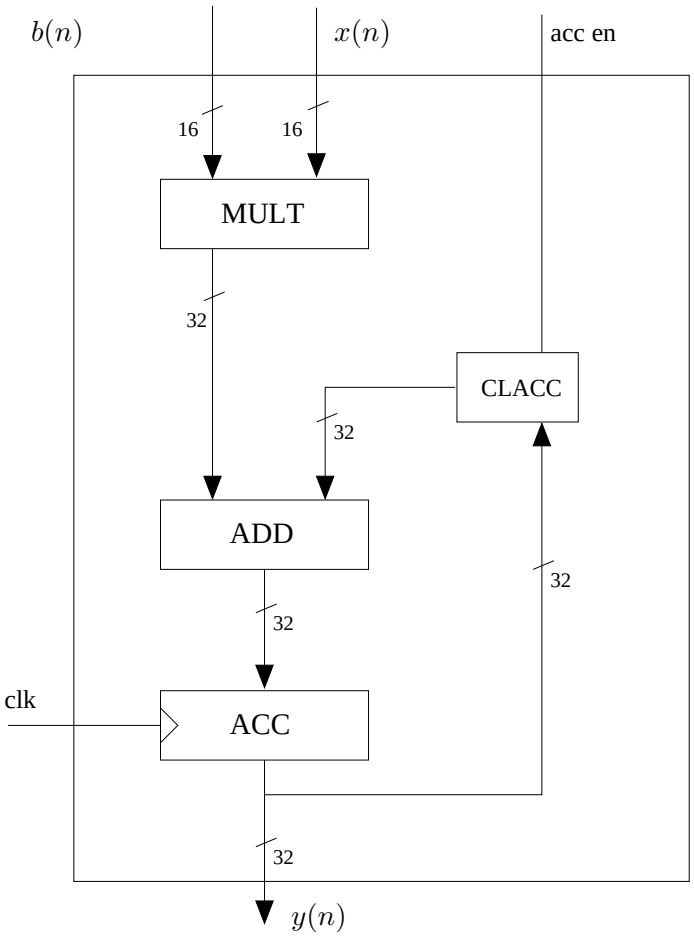


Figure 4.9: *Functional Diagram of MAC Block.*

- REGISTERS: COEFF REG and OUT REG are 16-bit flip-flop based registers used for clock synchronisation and to minimise critical data paths during synthesis. Propagation delay is therefore isolated between memory blocks and critical paths in the arithmetic unit.

4.3.2 Conventional weight update block

Within the WUD block there are a number of functional blocks as shown in Fig.4.10. It consists of three memory blocks for storing the input data (IN RAM), storing the calculated filter coefficients (LMS RAM) and storing the filter weights from the previous $(n-1)$ sample (FIR W RAM), a control block (CONTROL) and a number of distributed arithmetic blocks (AU MULT and AU ADD). A brief description of the main blocks is given below:

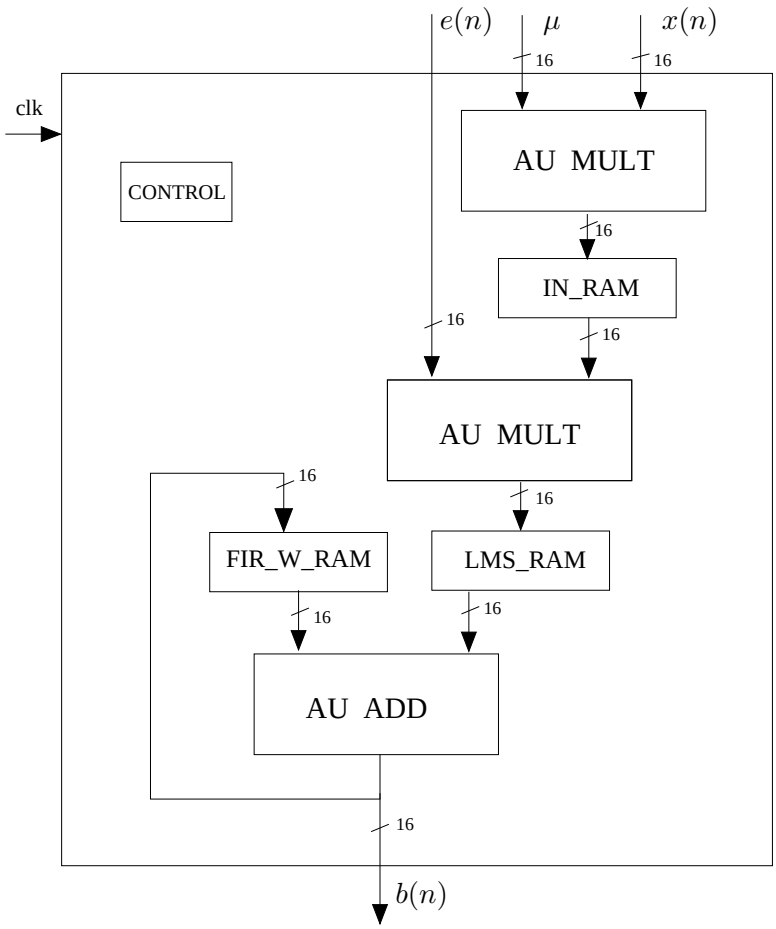


Figure 4.10: *Functional Diagram of Conventional WUD Block.*

- CONTROL: This state machine controller includes a counter which provides addresses for the RAM blocks and is responsible for the synchronisation of activity for every block in the WUD unit.

- IN RAM: This is a RAM used for the storage of the input data $x(n)$, after having been multiplied by the step-size μ. The input data sequence is clocked into this RAM for the use of the next multiplier AU MULT when calculating partial filter weights.

- LMS RAM: Upon calculation of partial filter weights the values are stored in this RAM, being clocked in by the CONTROL block. Values are clocked out at the request of the FIR filter to be presented to the AU ADD unit, the output of which being used directly during calculation of a filter output $y(n)$.

- FIR W RAM: When clocking filter weights out of the LMS RAM (corresponding to the FIR filter addresses signal) the values are immediately added to the filter coefficients from

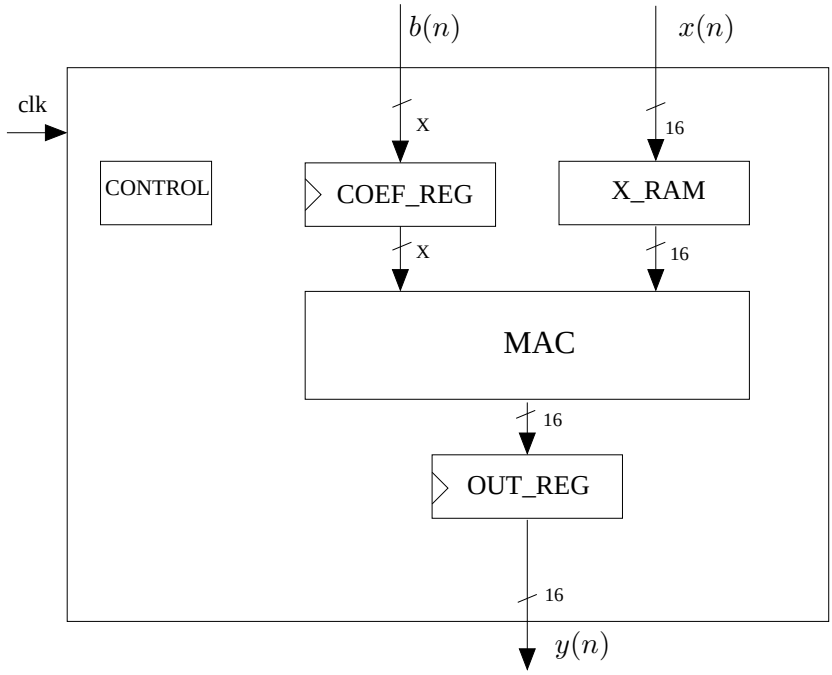


Figure 4.11: *Functional Diagram of DECOR FIR Block.*

the previous $(n-1)$ output sample held in FIR W RAM by AU ADD and written back into this FIR W RAM for subsequent calculation of filter weights for the next $(n+1)$ sample.

A distributed arithmetic approach using the AU MULT and AU ADD units was chosen to allow pipe-lining of coefficient calculation and to ensure that critical timing paths were minimised, this allows increased clock frequency when performing synthesis of the design. This arrangement also minimises the latency required to generate filter coefficients to the extent that regardless of the length of the filter only $N+2$ clock cycles are needed to calculate a complete set of N coefficients.

4.4 Implementation of DECOR LMS adaptive filter core

The DECOR adaptive filter consists of three functional blocks, seen in Fig.4.7. A WUD block, based on an optimised implementation of the conventional WUD block, a DECOR block for calculating the DECOR filter weights δ_i from the weights produced by the WUD block and a DECOR FIR filter.

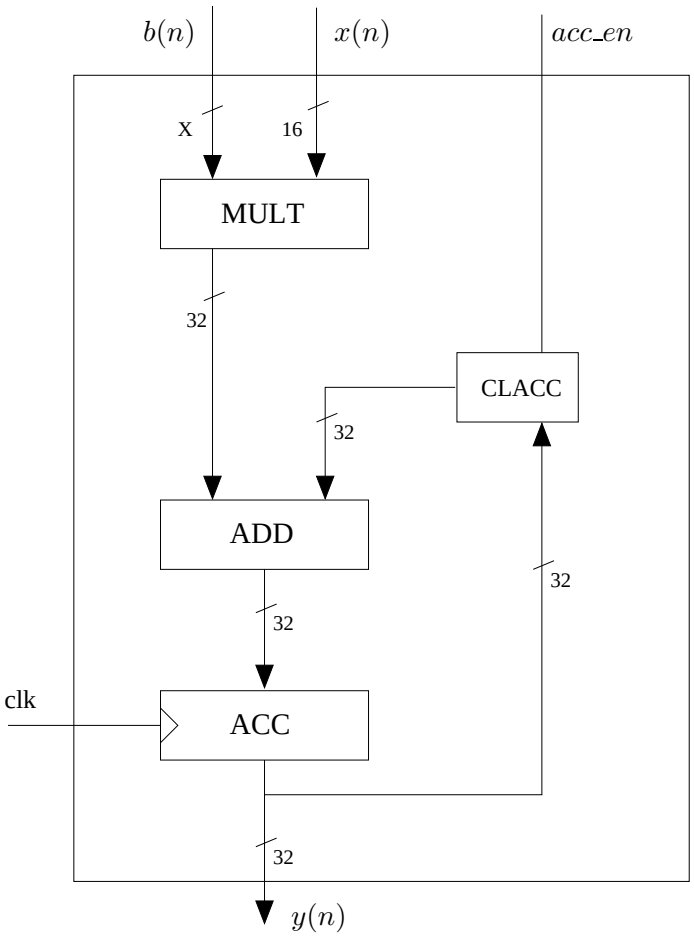


Figure 4.12: *Functional Diagram of DECOR MAC Block.*

4.4.1 DECOR FIR filter block

The general architecture of the DECOR FIR filter used here is identical to the implementation described in Section 4.3.1 with a number of important exceptions. A functional diagram of the first order DECOR FIR filter can be seen in Fig.4.11.

The first order DECOR FIR filter inherently requires one additional filter tap, this being due to the DECOR transform producing the differences between N adjacent filter coefficients. The result being that $N + 1$ DECOR coefficients δ_i are used to calculate one filter output sample. The filter therefore takes one extra clock cycle to calculate the output sample and the overhead needed becomes apparent here due to the single added memory location needed to store the extra coefficient.

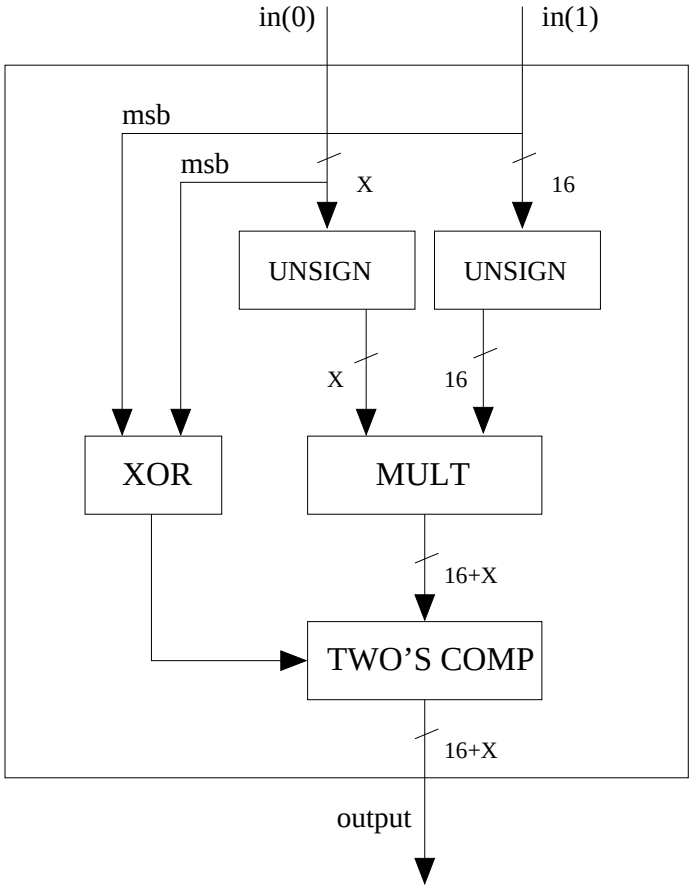


Figure 4.13: *Functional Diagram of Generic Multiplier.*

Contribution to overhead is also contained in the MAC block of the DECOR FIR filter due to the extra multiplication and additions required by DECOR. The addition of the previous $y(n-1)$ sample seen in equation (4.8) means that the clearing logic in the MAC block of the filter is no longer needed during operation. Continual accumulation without clearing the accumulator provides the result of adding the previous filter output, the output of the accumulator only need be sampled at the correct instant to produce output $y(n)$. The CLACC logic can be kept for resetting the MAC. A functional diagram of the MAC block can be seen in Fig.4.12.

There is also a difference in the MAC block of the DECOR FIR filter in that the reduction in δ_i coefficient word-length results in a reduction in the port size of the multiplier therein. The multiplier size is therefore 16 x X where X represents the word-length of the coefficients output by the DECOR block while the data word-length remains equal to 16 bits. This reduction in coefficient word-length results in a smaller multiplier compared to a conventional FIR filter.

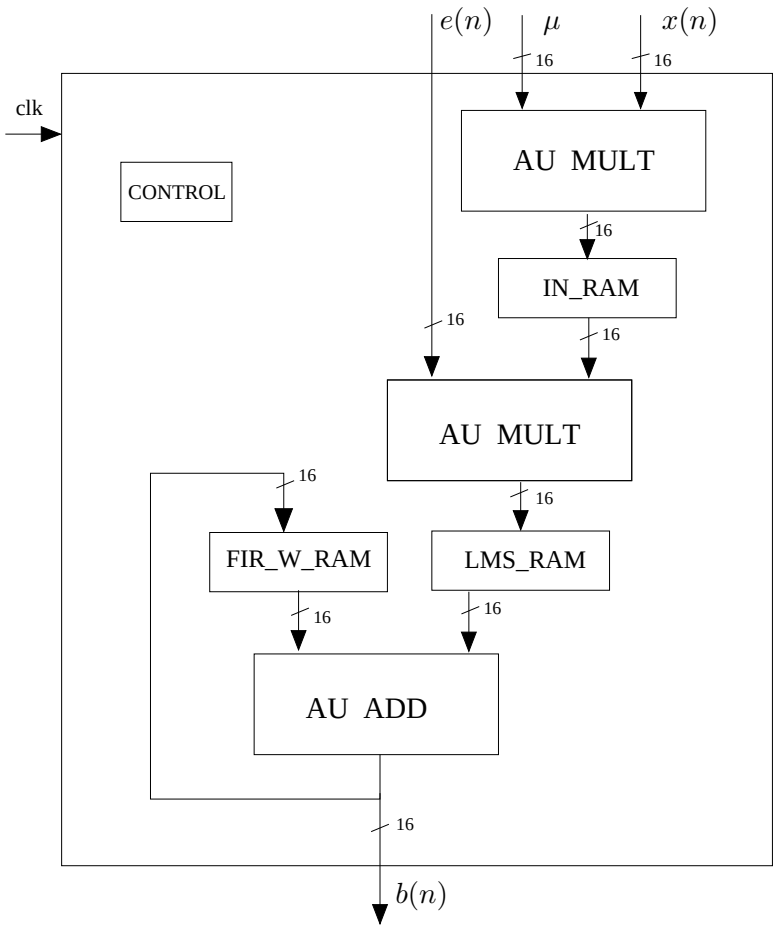


Figure 4.14: *Functional Diagram of DECOR WUD Block.*

Attention should also be given to the type of multiplier used in the MAC Block. It has been shown that the use of a generic unsigned multiplier is best suited for use in the application of the DECOR FIR filter[66]. The aim being that minimising the Hamming distance between coefficients will reduce switching activity in the multiplier. For these purposes then, before the multiplier will be presented with values, the two's complement representation will be converted into its unsigned form for multiplication. There is a penalty to be paid in throughput latency but the overall result will be a greater reduction in power. A functional diagram of the generic multiplier block used can be seen in Fig.4.13.

As only the first order DECOR filter is being investigated at this stage, the hardware implementation can be fixed without having to add the extra hardware needed in higher order DECOR implementation and the overhead required is far lower[66].

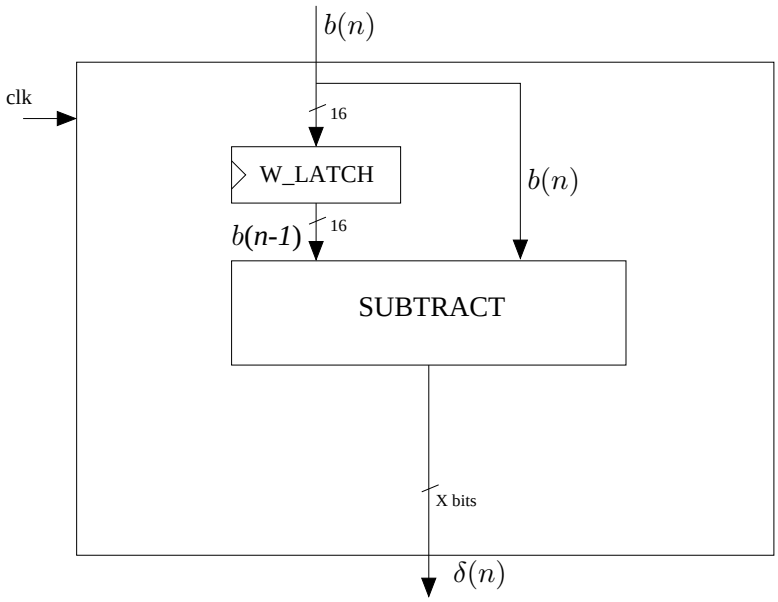


Figure 4.15: *Functional Diagram of DECOR Block.*

4.4.2 DECOR WUD block

A functional diagram of the DECOR WUD block can be seen in Fig.4.14. This DECOR WUD block is largely identical to the conventional WUD block described in section 4.3.2 and shown in Fig.4.10. The difference lies in the state machine controller which is modified to include extra control signals which halt the calculation upon completion of N filter coefficients. This is needed due to the WUD block performing the calculation of N filter coefficients while the transform through the DECOR block calculates the $(N + 1)$ coefficients required by the first-order $(m = 1)$ DECOR FIR filter.

4.4.3 DECOR block

The DECOR block represents part of the overhead in this implementation. The DECOR block used to calculate the δ_i filter coefficients, contains a number of registers for storing the previous δ_{i-1} filter coefficients and also a subtract block. A functional diagram of the DECOR block can be seen in Fig.4.15.

4.5 Design methodology

At this point, discussion of the design flow used to obtain results is important for this and subsequent chapters in this thesis. The basic design flow is shown in the flowchart illustration in Fig.4.16. Conventional and low power architectures are coded in Verilog hardware description language at register transfer level (RTL). Once the functional specification of the design is verified against a Matlab model, synthesis is then carried out using either Synopsys Design Compiler (dc-shell) or Cadence BuildGates to convert the RTL model into a gate level netlist. A standard delay format (SDF) file is also generated by the synthesis tool to provide gate level timing. DC wireload models were used in the synthesis flow targeting a 0.18μ standard cell CMOS library as they can be shown to be accurate at these geometries. For geometries at or below 0.13μ the use of wireload models alone would not be suitable. Design Compiler also generates a timing constraint file in SDF format for the Silicon Ensemble design layout tool. Post-synthesis netlist verification is then carried out determine if the design still meets functional and timing specifications at the gate level using the Cadence Verilog-XL simulator. If any specification is not met the RTL code and/or the timing constraints used during synthesis will be modified. This loop is carried out until the specification is met entirely and no violations are reported during simulation.

Power consumption results can be determined for a given design at the gate level and estimated using the Synopsys Design Power tool. This uses the gate level netlist with the switching activity from every net obtained from gate level simulation to calculate dynamic power. The switching activity for a given design is set in the testbench and output by the Verilog-XL simulator in a SAIF (switching activity file format) file. Power estimation can also be carried out after layout and will provide more reliable results. For this, Silicon Ensemble is used for conversion of the gate level netlist and the timing constraint file into a layout. This resulting modified netlist, post layout SDF and the extracted net capacitances file are then used to verify post layout functionality of the designs and then estimate power consumption using Design Power. Both the internal power and the switching power are detailed and defined as dynamic power consumed. Internal power comprises the power consumed within a cell boundary and is a combination of the short circuit power and the power needed for charging and discharging of the capacitances within the cell. Switching power is defined as the power used in the charging and discharging of the load capacitance at the output of a cell.

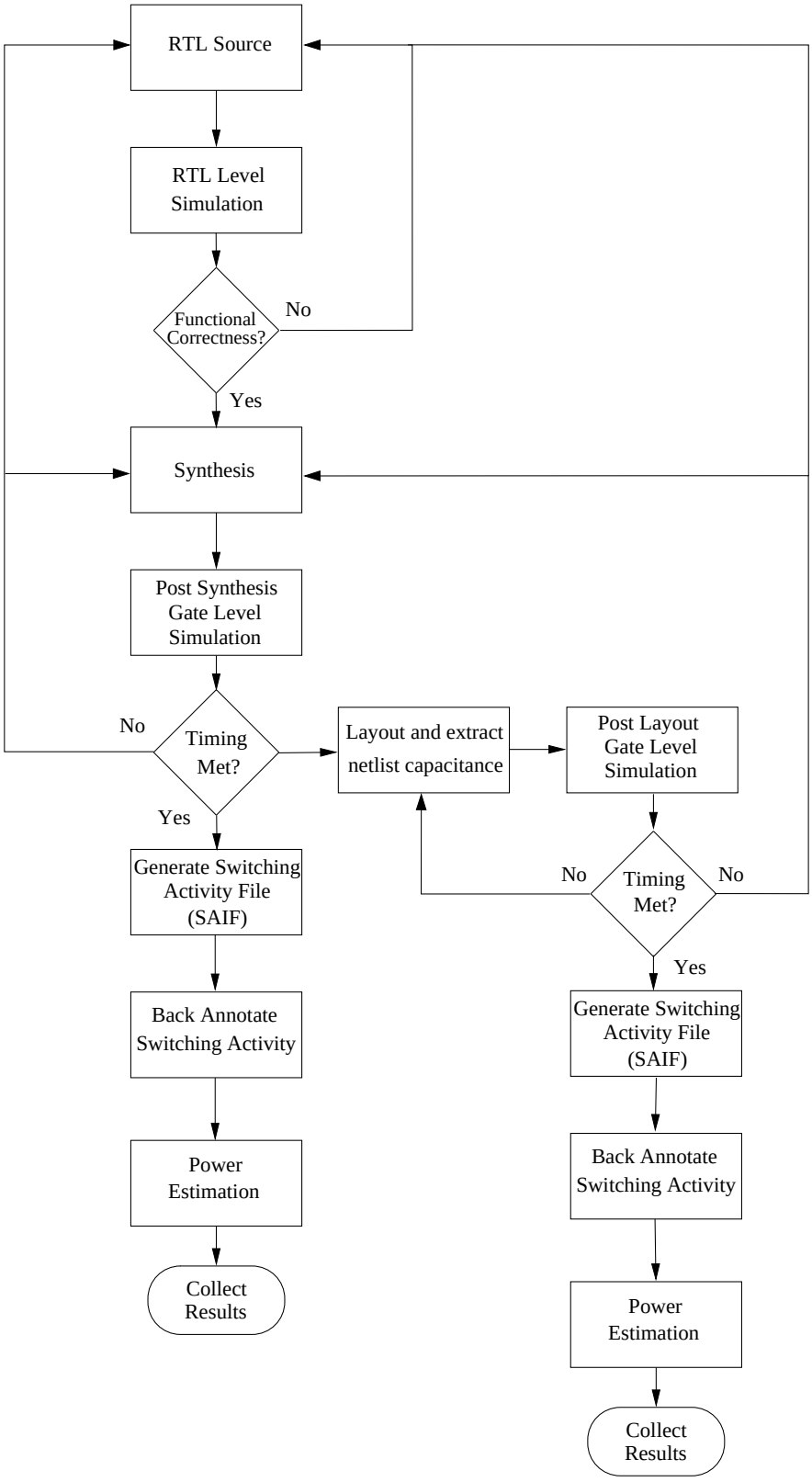


Figure 4.16: *Design Cycle Flow Chart.*

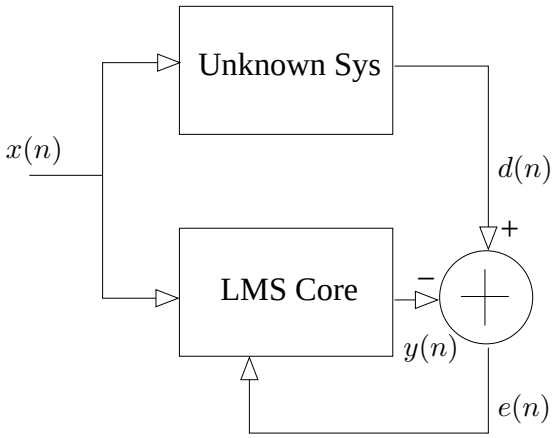


Figure 4.17: *Block Diagram of System Identification Configuration*

4.6 Results

Two different adaptive filter cores have therefore been implemented, the first being the conventional (CON) and secondly the 1st order DECOR. Both have been analysed in terms of area usage and power consumption. DECOR has been implemented for 16bit, 14bit, 12bit and 10bit coefficient wordlengths in order to study the impact of the reduction in size of the multiplier on area and power. For the purposes of this study and to provide a consistent comparison for both architectures, the carry-save array (*csa*) multiplier was selected for use throughout and level-sensitive latch based memory blocks were chosen. The cores were designed using Verilog HDL and then synthesised using Design Compiler^{TM}targeting the UMC 0.18μ standard cell CMOS library. The requirements of the synthesis were identical for all cores. This was vital in order to allow for consistent delay, power consumption and area usage comparisons. A netlist was created for each core and back-annotated netlist simulations for a uniformly distributed random input data of 1000 samples using Verilog-XL^{TM}simulator were performed and verified against Matlab^{TM}simulation results. The resulting data, including switching activity of the circuit nets was then used by Synopsys DesignPower^{TM}to determine power consumption for the different adaptive filter cores. In all of the above stages a clock frequency of 100 MHz and a supply of 1.8 Volts were used.

For the purposes of this chapter, results were collected using the LMS adaptive filter in a system identification configuration. The adaptive filter being used to characterise a system with 'unknown' transfer function. This was done to further verify that the adaptive filter was indeed converging to the correct set of filter coefficients and also to ensure stability. It became clear that for this particular implementation of an adaptive filter using the DECOR transform, stability is a major and ultimately limiting factor in its operation. The experimental configuration can be seen in the diagram of Fig.4.17. In reality the transfer function of the 'unknown' system in this setup is programmable by the user thus validating the functionality of the LMS core by comparison of the programmed transfer function with the steady state transfer function determined by the LMS core. The following power results therefore include only those of the LMS core under investigation.

Attention should also be paid to the reliability of the results obtained from the Synopsys DesignPower tool. These results, while obtained using consistent experimental procedure, are *estimates* of dynamic power consumed due to switching activity calculated from the compiled netlist. The effects of carrying out a physical layout, altering the number of input data samples or particular input data characteristics used for simulation are examples of where variability would be introduced. This therefore places a precision limitation on the power results produced by the tool in relation to the actual power consumed in any given physical realisation of this circuit in silicon. This should be kept in mind upon interpretation of the following results. Any reduction in power presented as a result of design optimisation is relative to a reference design for which results are recorded using an identical design flow and synthesis constraints. The numerical results presented assert a precision consistent with that of the performance of the DesignPower tool estimation, however in reality the precision of such results is questionable in terms of a physical realisation.

Core type	Dynamic Power (mW)
CON 16bit	22.08

Table 4.1: *Power Consumption for conventional adaptive core*

Block	Dynamic Power (mW)
Error	0.04
WUD	13.94
FIR	8.10

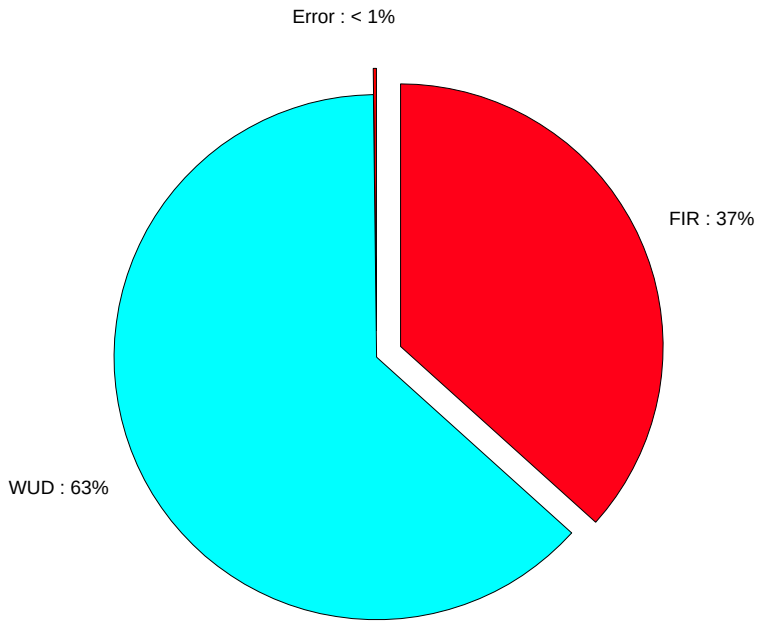
Table 4.2: *Power Consumption for functional blocks in adaptive core*

Block	Dynamic Power (mW)
Control	0.03
AU-MULT	2.58
IN-RAM	1.23
AU-MULT	2.60
LMS-RAM	1.28
Add	0.51
FIR-W-RAM	1.13

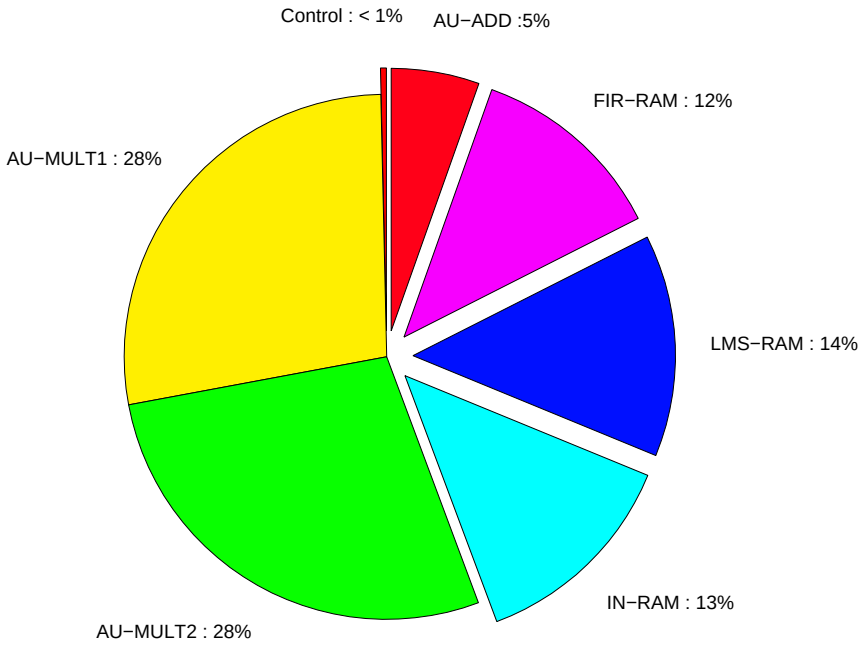
Table 4.3: *Power Consumption of WUD block components*

4.6.1 Power consumption for conventional adaptive core

An analysis of the power consumed by the conventional adaptive core is essential for comparison to be made against the DECOR core and allows the functional blocks consuming the highest power to be identified. Power results are shown in Table 4.1. A 73-tap adaptive filter was analysed with step-size μ = 0.01. The total power consumed by the 16 bit conventional core is 22.08mW. Synthesis constraints allowed the use of latch based memory throughout.



(a)



(b)

Figure 4.18: *a) Power consumed by adaptive core blocks and b) by WUD block components*

Algorithm	Dynamic Power (mW)	% Change
CON 16bit	22.08	-
DECOR 16bit	20.94	-5.16
DECOR 14bit	20.13	-8.83
DECOR 12bit	18.88	-14.50
DECOR 10bit	18.01	-18.43

Table 4.4: *Power Consumption analysis for different coefficient word-lengths*

The power consumed by each of the functional blocks in the adaptive core is illustrated in Fig.4.18(a). It can be seen that the weight update block consumes the most power, this follows as by its nature it is the most computationally complex. The FIR filter is the next highest, containing a single multiply-accumulate block, and the error calculation block is the lowest in terms of power consumed. Already, it is apparent that the DECOR transform technique does not target the largest block level power consumer in the core and relies only upon reducing the power consumed by the FIR block which is the second highest power consumer.

4.6.2 Power consumption for DECOR adaptive core

Power results for the DECOR implementation are shown in Table 4.4. A 73-tap DECOR adaptive filter was analysed with step-size μ = 0.01. Performance analysis was carried out using four different coefficient wordlengths each of which being implemented individually in Verilog. The coefficient wordlengths for each implementation were chosen to highlight the effects of using the DECOR transform and its limitations. Table 4.4 shows that for the 1st order DECOR transform a decrease in coefficient wordlenth results in an increase in the *overall* power saving achieved. As has been indicated, the DECOR transform is implemented within the FIR block of the LMS core only and targets a power reduction therein. The reduction in coefficient wordlength presented to the FIR MAC block therefore reduces the switching activity in the arithmetic units thus reducing the dynamic power dissipated.

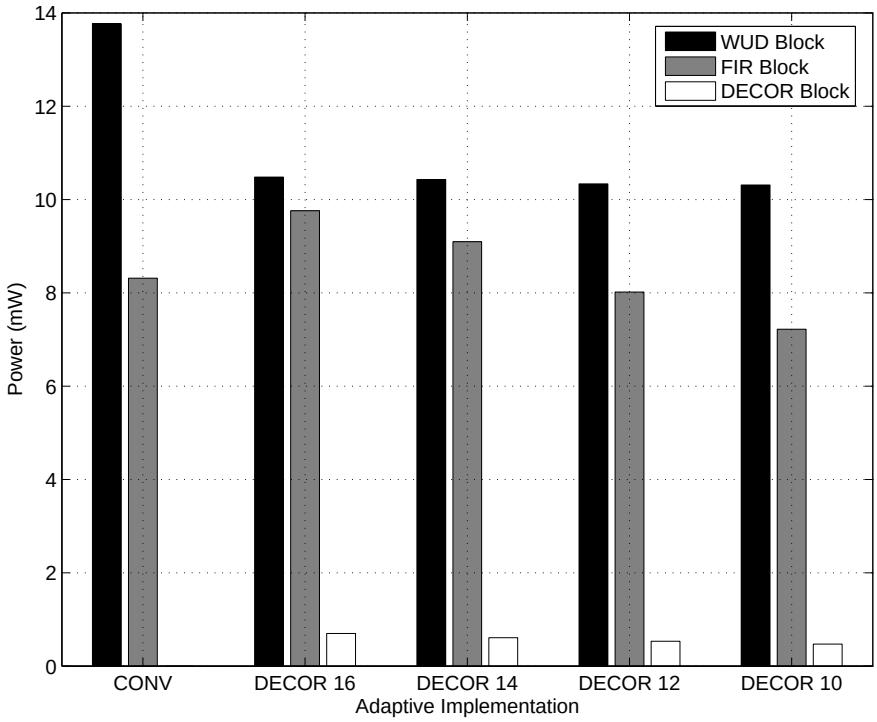


Figure 4.19: *Power Consumption for Main Functional Blocks*

Algorithm	Dynamic Power (mW)
DECOR 16bit	0.70
DECOR 14bit	0.61
DECOR 12bit	0.53
DECOR 10bit	0.47

Table 4.5: *Power Analysis for Decor Block*

A breakdown of the power consumed by different blocks within each adaptive filter design shows where the overhead in this technique lies and where the greatest power savings are made. The graph in Fig.4.19 shows the power consumed by the main functional blocks within the adaptive filter and illustrates the block level power dissipation in each implementation relative to one another. Attention must be brought to the resultant overall reduction in power even though Fig.4.19 clearly shows that the FIR block in the DECOR implementation actually consumes more power than the conventional core until the coefficient wordlength reaches 10 bits. This is dealt with in the following section which draws comparison between the

Algorithm	Dynamic Power (mW)	% Change
CON 16bit	5.185	-
DECOR 16bit	7.07	+36.35
DECOR 14bit	6.57	+26.75
DECOR 12bit	5.75	+10.88
DECOR 10bit	5.15	-0.77

Table 4.6: *Power Analysis for MAC Block Within FIR*

Algorithm	Dynamic Power (mW)	% Change
CON 16bit	8.31	-
DECOR 16bit	9.76	+17.38
DECOR 14bit	9.10	+9.40
DECOR 12bit	8.02	-3.56
DECOR 10bit	7.22	-13.15

Table 4.7: *Power Analysis for FIR Filter Block*

conventional and DECOR implementations. Table 4.5 shows the power consumption of the Decor block for each coefficient wordlength implementation. Due to the number of bits representing the filter coefficients being reduced, the power consumption of this block is reduced accordingly.

4.6.3 Comparison between conventional and DECOR core

A reduction in power consumed by the DECOR FIR block can be seen as the coefficient wordlength is reduced. This being the aim of the DECOR transform in that the filter coefficients presented to the MAC within the filter are directly reduced. Table 4.6 shows the power consumption of the MAC within the CONV filter and within the DECOR filter for each implementation. Table 4.7 shows the power results for the FIR block in each implementation. Not until the coefficient wordlength reaches 10 bits does the MAC in the DECOR implementation contribute to the overall power saving. The overhead present in the extra calculations requiring processing within the MAC block by the DECOR algorithm is not

Algorithm	Area (μm^2)	% Reduction
CON 16bit	1103487.13	-
DECOR 16bit	951924.25	13.7
DECOR 14bit	948440.00	14.1
DECOR 12bit	944398.69	14.4
DECOR 10bit	941235.69	14.7

Table 4.8: *Area analysis for different Coefficient word-lengths*

therefore mitigated by a net power saving until a coefficient wordlength of 10 bits or less is used.

Some minimal optimisation can be made to the WUD block in this implementation. Looking at the difference in power consumption between the 16 bit CON adaptive filter and the 16 bit DECOR adaptive filter highlights this. The X RAM used in the conventional adaptive filter can be replaced by a latch based shift register in DECOR seen in Fig.4.14, which reduces overall power consumption. The implementation of DECOR creates an extra filter coefficient thus allowing the shift register to be used as the timing of control signals is modified. Essentially the extra filter weight created by the DECOR transform allows the control timing of the DECOR WUD block to tolerate propagation delay of the stored coefficients through the shift register.

It can also be seen however that the overhead introduced by the DECOR block in terms of power consumed, is very low in relation to other blocks and changes very little as the wordlength in the DECOR implementations reduces. A small reduction in the power consumed by the DECOR block in each implementation is evident due to the reduction in coefficient wordlength it calculates. A major benefit of this architecture is that the logic gates required by the DECOR block is minimal and therefore does not present a great penalty in terms of power or area.

4.6.4 Area overhead and comparison

Area results are shown in Table 4.8. These results clearly show the considerable overall area overhead present in the implementation of the DECOR FIR filter. A wordlength reduction of 6 bits is needed before the 1st order DECOR filter will demonstrate any power saving in the MAC block. This overhead is due to the extra multiplications and additions needed in the DECOR algorithm. In practice, implementation of a DECOR filter without a reduction in the coefficient

wordlength would never be worthwhile given that the very purpose of the DECOR transform is to reduce this parameter. It has simply been done to point out the overhead present in the design. Nonetheless, it can be seen that due to optimisation in the WUD block an overall power saving can be made for the adaptive filter as a whole. This is due to the optimisation of the $x(n)$ input data storage in the WUD block.

4.7 Summary

This chapter has presented the complete implementation of the DECOR transformation technique for low power adaptive filtering cores. The technique was implemented for different DECOR coefficient word-lengths. The results demonstrate a power saving in the range of 5 - 15 % for coefficient word-lengths varying from 16 bits to 10 bits when compared to a conventional adaptive filter implementation. At the same time, an area saving of up to 15 % is achieved due to optimisation that can be carried out in the design.

This DECOR technique however is not without severe limitations when used in an LMS adaptive filter. Firstly, the DECOR technique targets the FIR filter block within the adaptive core for power reduction. The FIR block is not the highest consumer of power in the design with the weight update block being the more computationally complex. Secondly, the application of the DECOR transform to the LMS adaptive filter results in severe stability issues. It would be difficult and impractical therefore to justify the use of this method of power saving in a real-world application regardless of the proven reduction in power consumption that has been shown here.

Chapter 5
Variable Length Adaptive Filter Core

This chapter describes the use of the variable length algorithm for dynamically updating the tap-length of the LMS adaptive filter to optimise performance and for reducing the power in the adaptive filter core[38]. A novel HDL implementation of this algorithm is presented.

This chapter is organised into five sections. Section 5.1 introduces the motivation for the concept of dynamically updating the tap length of the adaptive filter and provides detail of accompanying research on the topic. Section 5.2 gives an overview of the techniques used in the length update algorithm chosen for implementation with sections 5.3 and 5.4 describing the hardware architecture design. Section 5.6 presents the power consumption results collected from the implementation of this architecture.

5.1 Variable length adaptive filter techniques

Attention here is concentrated on the design of adaptive filtering cores, heavily constrained in power and area requirements for use in mobile wireless applications. In these mobile communication systems channel equalisers are used to mitigate the effects of inter-symbol interference introduced by the wireless channel. In time varying channels the equaliser will track the the change in the channel impulse response. Linear equalisers are typically implemented using adaptive finite impulse response (FIR) filters [1] with filter coefficients being recursively updated using either the recursive least squares (RLS)[67] or more commonly the least mean squares (LMS) algorithm[68] as is used in this study. The performance of the adaptive filter can be measured in a number of ways. Computational complexity, rate of convergence, tracking ability and the resultant steady state mean square error (MSE) are all ways to determine the performance of the filter. The number of taps in the FIR structure also has a critical influence on the performance and computational complexity of the equaliser. An equaliser with too many taps will be computationally inefficient and therefore consume more power than need be, may have reduced tracking and convergence performance and may

introduce a degradation in mean squared error (MSE) performance due to limitations of the LMS algorithm itself. Whereas, an equaliser with too few taps will be unable to reach the desired level of distortion mitigation. If the number of taps in operation is low then MSE performance will improve as taps are added until the optimum number is reached. Coupled with this is the time variant nature of wireless channels which ideally requires the ability of the equaliser to alter the number of taps in operation with time. Optimising the number of taps in operation is highly desirable due to the ability to use as few taps as possible. This will speed the rate of convergence and reduce computational complexity therefore saving power, all without detriment to the MSE performance[68].

The technique of varying the length of the LMS filter was first presented by the authors of [39] with an algorithm which proved that a filter with fewer taps will have a faster convergence than that of a filter with a higher number of taps. This variable length stochastic gradient (VLSG) algorithm demonstrates an LMS adaptive filter which can accomplish a change in its length from being initially low, therefore aiding fast convergence, gradually increasing over time to achieve the low steady state MSE performance characteristic of higher order filters. In [40], Won et al. went on to propose another variable length LMS (VL-LMS) algorithm using a time-constant concept whereby several filter lengths are predetermined and filter length is increased to the next predetermined value when conditions are satisfied.

Both the algorithms referred to previously however offer only the ability to *increase* the filter length over time to satisfy the contradictory goals of fast convergence and good steady state performance. As a progression from the previous methods, the authors Riera-Palou et al. of [38] specifically present a linear equaliser using an algorithm that can dynamically and automatically increase or decrease the length of the filter. Using a segmented FIR filter structure and a weight update algorithm the optimum, and in this case minimum required, number of taps are operated. Further to this, [69] presents a method whereby the optimum length of the adaptive filter is determined. In this case the number of filter coefficients of an unknown system are found using the LMS algorithm in a system identification setup. This method uses the MSE output from a number of individual LMS adaptive filters in parallel to determine the number of unknown system coefficients and their values. This does not lend itself well to a circuit implementation which is constrained in terms of its area and power. Finally in [70], the authors present the most recent proposal for the variable length LMS algorithm. This algorithm uses a method whereby the filter length is varied according to the negative gradient direction of the estimation error.

The familiar gradient decent method is used to track the optimum filter length with constraints included to avoid unexpected behaviour and guarantee convergence.

5.1.1 Adaptive filter length analysis

For the static channel where channel coefficients remain constant, the optimum equaliser coefficients can be found by solution of the Weiner-Hopf equation if the channel impulse response is known. This is given by:

$$\mathbf{w} = \mathbf{R}^{-1}\mathbf{p} \tag{5.1}$$

Where $\mathbf{R}$ is the auto-correlation matrix of $x(n)$ and $\mathbf{p}$ is the cross-correlation matrix between $x(n)$ and $d(n)$, where $x(n)$ is the input vector to the equaliser and $d(n)$ is the original source data.

The LMS algorithm tries to find the Weiner solution for the equaliser by constantly updating the filter coefficients in accordance with the error signal $e(n)$ when the channel impulse response is unknown and varying with time.

From [37], the MSE produced by the LMS filter can be expressed as:

$$J(n) = E[|e(n)|^2] = J_{min} + J_{ex}(n) \tag{5.2}$$

where the MSE, $J(n)$, is made up of the two components J_{min} and J_{ex}. J_{min} is dependent upon channel conditions and the number of taps in the equaliser. Its value will decrease as the number of taps in the equaliser increases. J_{ex} is the excess MSE (EMSE) and is related to the specific adaptive algorithm in use. In the case of the LMS algorithm the EMSE is attributed to inaccurate estimation of the gradient vector and the tracking error in the case of a time varying channel. When the LMS algorithm has reached the steady state, the MSE achieved can be expressed as shown in [1] as:

$$J(\infty) = J_{min} + \mu J_{min} \sum_{k=1}^{M} \frac{\lambda_k}{2 - \mu\lambda_k} \tag{5.3}$$

where λ_k is the kth eigenvalue of the auto-correlation matrix $\mathbf{R}$. If the step-size μ is kept constant then $J(\infty)$ can be reduced by either reducing J_{min} or by reducing the second term in 5.3 and as μ is normally small, the second term has little influence on the overall value for $J(\infty)$. Under normal circumstances, when the SNR level in the channel is high, a longer equaliser will result in a lower J_{min} than that of a shorter equaliser. However when the SNR in the channel is low the J_{min} of the short equaliser is almost the same as that of the longer equaliser. Therefore, increasing the length of the equaliser will not result in better MSE performance and there is no advantage to be gained by having more taps in operation.

5.2 Variable length algorithm

In this chapter the variable length LMS algorithm proposed in [38] has been chosen for its simple and efficient circuit implementation. This variable length algorithm provides the ability to dynamically increase or decrease the number of filter taps in operation using the segmented FIR filter approach. The details of operation of this algorithm will be presented in the following subsections followed by its proposed implementation in the next section.

5.2.1 The LMS adaptive filter

An *N*-tap FIR filter performs the following convolution:

$$y(n) = \sum_{i=0}^{N-1} b_i x(n-i) \tag{5.4}$$

where b_i's are the coefficients of the filter, $x(n)$ and $y(n)$ are the nth terms of the input and output sequences, respectively. The weight update equation for a least-mean squares (LMS) filter is

$$b_i(n+1) = b_i(n) + \mu e(n) x(n-i) \tag{5.5}$$

where μ is the step size and $e(n)$ is the adaptation error given by

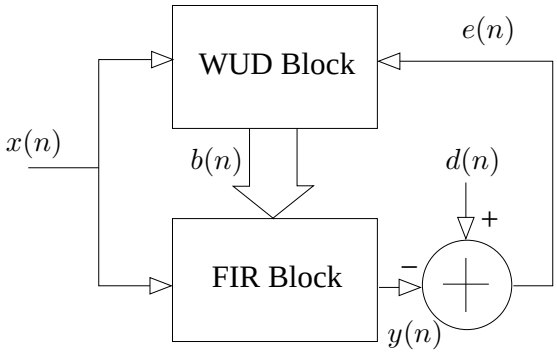


Figure 5.1: *Block Diagram of Conventional LMS Adaptive Filter.*

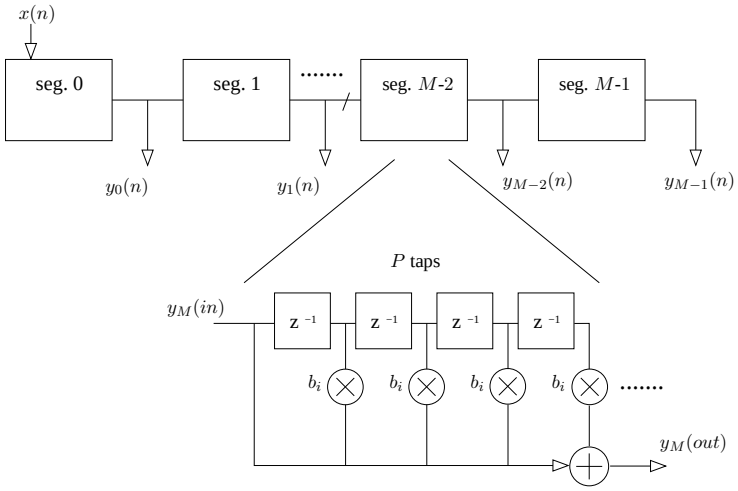


Figure 5.2: *Block Diagram of Segmented FIR.*

$$e(n) = d(n) - y(n) \tag{5.6}$$

$d(n)$ is the desired output of the filter, i.e. the transmitted signal. A block diagram representing the LMS adaptive filter can be seen in Fig.5.1.

5.2.2 Segmented filter

The use of a segmented filter involves splitting an N tap FIR into M concatenated sub-filters of P taps each, such that $N = MP$ as shown in Fig. 5.2. Each segment produces an output $y_s(n)$ (with $0 \leq s < M$) of the input data $x(n)$. The output of the last segment ($y_{M-1}(n)$) will be the same as the output of a conventional N tap FIR filter. In theory the overhead associated with this method however is the need for $M - 1$ extra adders.

5.2.3 Length update equations

When used in a linear equaliser, the segmented filter described in 5.2.2 produces an estimate $y_s(n)$ of the transmitted signal $d(n)$. The information provided by the various equaliser outputs $y_s(n)$, can be exploited to compute a corresponding error signal $e_s(n)$ according to (5.6) such that

$$e_s(n) = \big|\, d(n) - y_s(n) \big| \tag{5.7}$$

The distinct error signals can be squared and averaged to obtain an output MSE measure for each sub-filter. The MSE is defined as:

$$MSE_s(n) = E[\big|d(i) - y_s(i)\big|^2] = \frac{\sum_{i=1}^{n} e_s(i)^2}{n} \tag{5.8}$$

The performance criteria used to evaluate the different sub-filters is the accumulated squared error (ASE) and is defined as

$$ASE_s(n) = \sum_{i=1}^{n} \big|d(i) - y_s(i)\big|^2 = \sum_{i=1}^{n} e_s(i)^2 \tag{5.9}$$

The advantage of using the ASE is that the repetitive computation of division, used in the calculation of MSE, is removed. The aim of the length update algorithm is to detect the sub-filter at which the ASE becomes insignificantly smaller or even larger than the previous sub-filter. The algorithm proposed in [38] to control the number of active sub-filters involved in equalisation, assuming the equaliser has W active segments, is outlined as

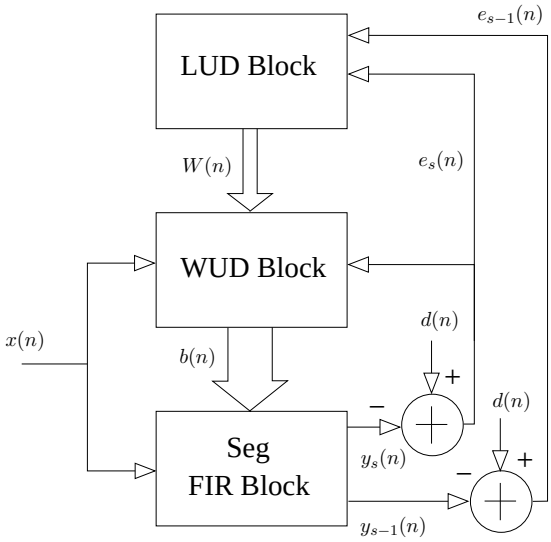


Figure 5.3: *Variable Length Adaptive Filter.*

$$ASE_{W-1}(n) = \sum_{i=1}^{n} \beta^{n-1} \left| d(i) - y_{W-1}(i) \right|^2 \tag{5.10}$$

$$ASE_{W}(n) = \sum_{i=1}^{n} \beta^{n-1} \left| d(i) - y_{W}(i) \right|^2 \tag{5.11}$$

$$\begin{aligned} &\text{If } ASE_W(n) \leq \alpha_{up} ASE_{W-1}(n) \\ &\quad \Rightarrow +1 \text{ sub-filter (P extra taps)} \end{aligned} \tag{5.12}$$

$$\begin{aligned} &\text{If } ASE_W(n) \geq \alpha_{dw} ASE_{W-1}(n) \\ &\quad \Rightarrow -1 \text{ sub-filter (P fewer taps)} \end{aligned} \tag{5.13}$$

where $0 < \alpha_{up} \leq \alpha_{dw} \leq 1$ and determine the amount of worsening or improvement necessary to force the equaliser to expand or contract. β is a forgetting factor and is ≤ 1. A more detailed derivation of this algorithm can be found in [38].

5.3 Implementation of the conventional LMS adaptive filter core

In order to evaluate the performance of the variable length adaptive filter a conventional adaptive filter core based on the convolution equation of (5.4) and using the LMS algorithm with it's equations (5.5) and (5.6) was implemented. The conventional implementation used is exactly the same as that used in Chapter 4, detailed in section 4.3. This conventional adaptive filter has two functional blocks, as seen in Fig.5.1. A weight update block (WUD), which uses the data input samples and an error signal to calculate new coefficients, and a FIR filter block, which employs the coefficients calculated by the WUD block.

5.3.1 Conventional FIR filter block

A sequential FIR filter implementation was used to minimise hardware cost and reduce complexity. Filter coefficients from the WUD block are clocked into the filter and presented to a single MAC unit which successively multiplies each delayed input sample with the appropriate filter coefficient and accumulates the results to provide the output sample y(n). This filter implementation consists of a single memory block for storing input data $x(n)$, a control block, a number of single word registers for clock synchronisation and an arithmetic block. More detail about these blocks has been given in the previous chapter, section 4.3.1.

5.3.2 Conventional weight update block

Within the WUD block there are a number of functional blocks. It consists of three memory blocks for storing the input data, storing the calculated filter coefficients and storing the filter weights from the previous $(n-1)$ sample, a control block and a number of distributed arithmetic blocks. A description of the main blocks has been given in the previous chapter, section 4.3.2.

5.4 Implementation of variable length adaptive filter core

Basing implementation of this design on the algorithm described in [38], the variable length adaptive filter consists of three functional blocks and can be seen in Fig.5.3. A WUD block, based on the implementation of the conventional WUD block but with an enhanced control block, a length update (LUD) block, used to calculate the ASE values and control the increase or decrease in the number of filter taps using the error signals $e_s(n)$ and $e_{s-1}(n)$, and a segmented

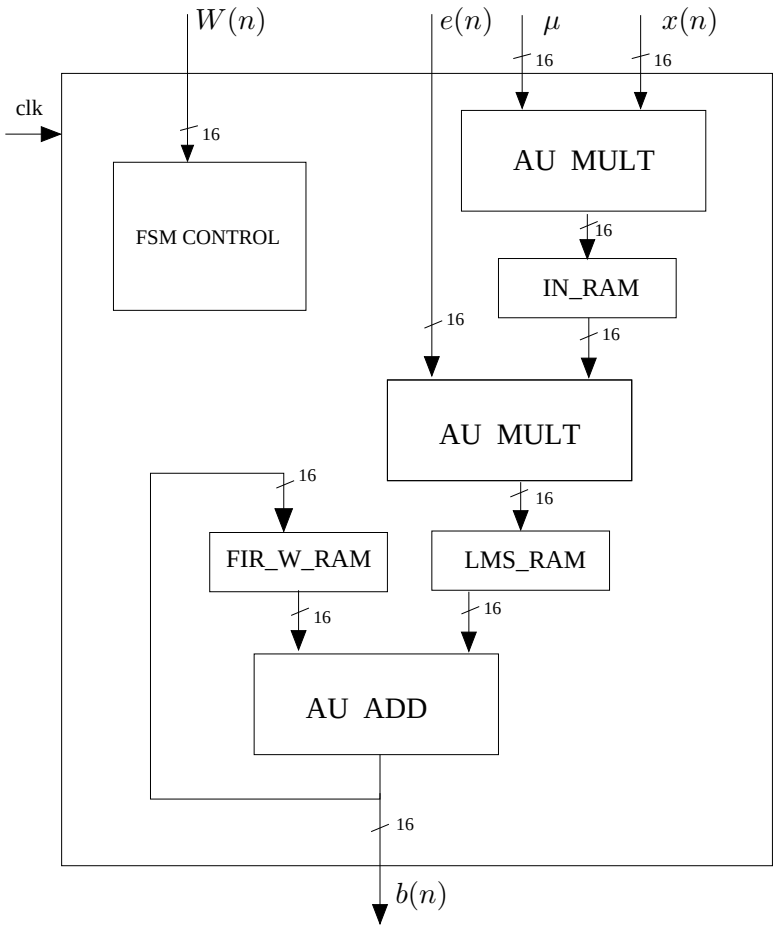


Figure 5.4: *Block Diagram of Varible Length WUD Block.*

FIR filter block are all required.

5.4.1 Weight update block

The finite state machine (FSM) control element in the conventional WUD block is replaced by an enhanced design which accepts the tap-length data from the LUD block, controls all timing signals and correctly addresses the various RAM elements in the WUD block appropriate to the added functionality of the length update algorithm. This enhanced control element is the only alteration made to the WUD block and enables the correct timing and synchronisation of the block to allow the length of the adaptive filter to increase or decrease based on the output value of the LUD block. A diagram of the WUD block used in the variable length core can be seen in Fig.5.4.

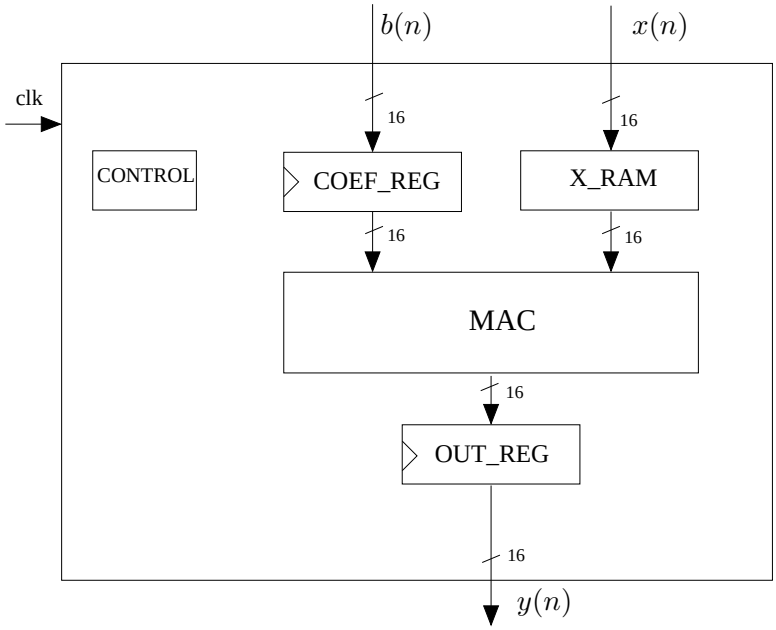


Figure 5.5: *Block Diagram of Segmented FIR Filter.*

5.4.2 Segmented FIR filter

The segmented structure of the FIR filter allows the calculation of sub-filter outputs $y_s(n)$ and thus the error signals used to evaluate the error performance of the N tap filter and $(N-P)$ tap filter (P being the number of taps in one sub-filter or segment). In this design the segmented FIR filter will calculate an output dependent on the number of coefficients presented to it by the WUD block. It is the output of sub-filter results that are critical to the operation of the variable length adaptive filter design. A representation of the structure of the segmented FIR filter can be seen in Fig.5.2 to demonstrate the principle of operation. In the design and implementation of the segmented filter however, it was found that due to the use of a sequential FIR block in the conventional adaptive core implemented previously, that sub-filter results could be captured at the output of the accumulator without having to include any further adders. The accumulator calculates an output value for every filter coefficient, therefore at the appropriate instant any of the sub-filter outputs can be latched and used for error calculation. This is contrary to the structure represented in Fig.5.2 and proposed in [38] where an extra adder is required for every sub-filter output $y_s(n)$ and the number of taps per sub-filter P is fixed, thus a hardware saving is achieved. A block diagram of the implementation chosen for the segmented FIR filter is shown in Fig.5.5.

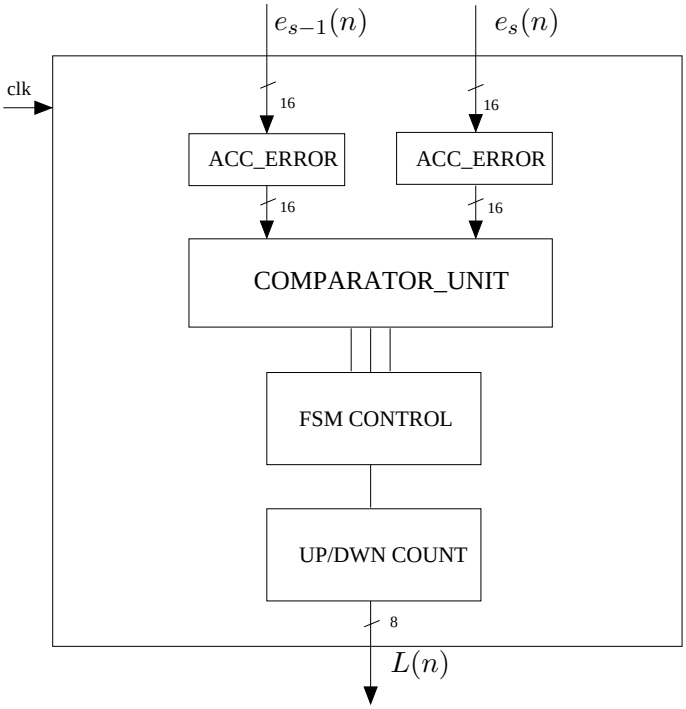


Figure 5.6: *Block Diagram of Length Update Block.*

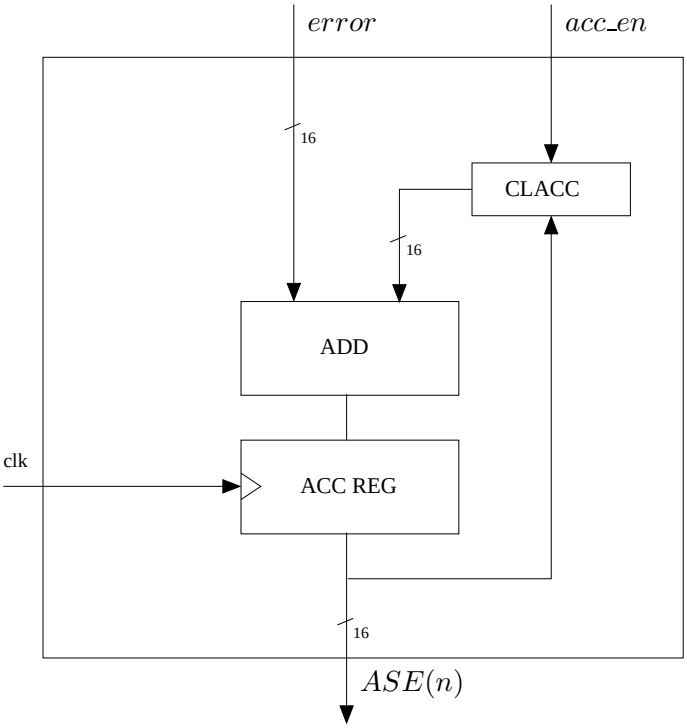


Figure 5.7: *Block Diagram of Accumulator used in LUD Block.*

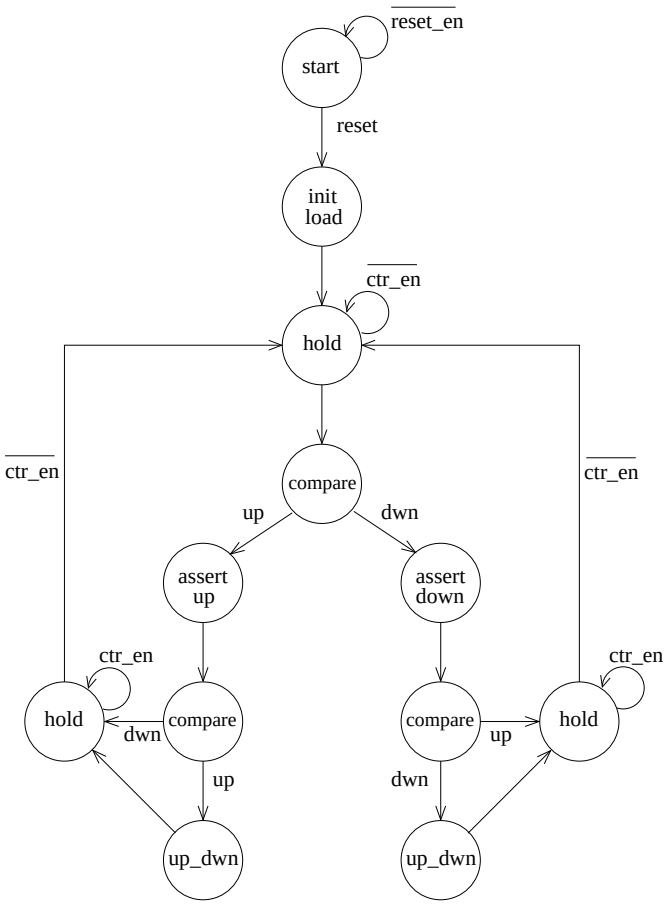


Figure 5.8: *FSM Control used in LUD Block.*

5.4.3 Length update block

The LUD block takes the error signals from the last two FIR sub-filters, $e_s(n)$ and $e_{s-1}(n)$ and calculates the respective ASE of equations (5.10) and (5.11) using accumulator units. A functional block diagram of the LUD block can be seen in Fig.5.6 with the structure of the identical accumulator units shown in Fig.5.7. At this point the LUD algorithm has been modified to reduce complexity when implemented in a fixed point architecture. The windowing function β has been removed and the forgetting factors α_{up} and α_{dw} are substituted by using a threshold value for each rather than performing the multiplication of α_{up} and α_{dw} with the respective ASE values. Simulation showed this to be no detriment to the operation of the algorithm. The number of samples over which the ASE values are accumulated can effectively be varied between 0 and 255, this is determined by the value registered during initialisation which controls the size of the counter used in the accumulator units. The effect of varying the accumulator interval simply determines how often the LUD will update the length of the

equaliser. A greater number of samples can be accumulated simply by extending the range of this counter but it was chosen to be 8 bits wide in this case. The ASE values are latched and presented to a comparator unit in accordance with the length update algorithm. A state machine controller in the LUD block then increments or decrements an up/down counter according to the instantaneous comparator result to provide a tap-length value. This LUD controller initialises the up/down counter and controls timing of the arithmetic units present. The state diagram implemented in the LUD controller can be seen in Fig.5.8. The number of taps initialised at reset is selected by the user and the maximum number of taps is set by the maximum size of the memory blocks and counters in hardware. The algorithm can vary the tap length of the filter to integer multiples of sub-filter size P with the minimum size being P taps.

5.5 Design methodology

The basic design flow is the same as that detailed in section 4.5. Conventional and low power architectures are coded in Verilog hardware description language at register transfer level (RTL). Once the functional specification of the design is verified against a Matlab model, synthesis is then carried out using either Synopsys Design Compiler (dc-shell) or Cadence BuildGates to convert the RTL model into a gate level netlist. A standard delay format (SDF) file is also generated by the synthesis tool to provide gate level timing. Design Compiler also generates a timing constraint file in SDF format. Post-synthesis netlist verification is then carried out determine if the design still meets functional and timing specifications at the gate level using the Cadence Verilog-XL simulator. If any specification is not met the RTL code and/or the timing constraints used during synthesis will be modified. This loop is carried out until the specification is met entirely and no violations are reported during simulation.

5.6 Results

Two different adaptive filter cores have therefore been implemented, the first being the conventional (CON) and secondly the variable length (VAR-LMS). Both have been analysed in terms of area usage and power consumption and in channel equaliser configuration. For the purposes of this study and again to provide a consistent comparison for both architectures, the carry-save array (*csa*) multiplier was selected for use throughout and latch based memory blocks are used. The cores were designed using Verilog HDL and then synthesised using

Core type	Dynamic Power (mW)
CON (64tap)	21.01

Table 5.1: *Power Consumption for conventional adaptive core*

Block	Dynamic Power (mW)
Error	0.041
WUD	13.15
FIR	7.819

Table 5.2: *Power Consumption for functional blocks in adaptive core*

Design Compiler™ targeting the UMC 0.18μ standard cell CMOS library. The requirements of the synthesis were identical for all cores. This was vital in order to allow for consistent power consumption and area usage comparisons. A netlist was created for each core and back-annotated netlist simulations for a uniformly distributed random input bipolar binary data of 10000 samples using Verilog-XL™ simulator were performed and verified against Matlab™ simulation results. The resulting data, including switching activity of the circuit nets was then used by Synopsys DesignPower™ to determine power consumption for the different adaptive filter cores. In all of the above stages a clock frequency of 100 MHz and a supply voltage of 1.8 Volts were used.

Again these results, while obtained using consistent experimental procedure, are *estimates* of dynamic power consumed due to switching activity calculated from the compiled netlist as is described in section 4.6. The numerical results presented are consistent with the precision of the DesignPower tool estimation, however in reality the precision of such results is questionable in terms of a physical realisation.

5.6.1 Power consumption for conventional adaptive core

An analysis of the power consumed by the conventional adaptive core is again made for comparison against the variable length core. The functional blocks consuming the highest power are identified. Power results are shown in Table 5.1. A 64-tap adaptive filter was analysed with step-size μ = 0.01. The total power consumed by the conventional adaptive filter core is

Block	Dynamic Power (mW)
Control	0.03
AU-MULT	2.31
IN-RAM	1.18
AU-MULT	2.60
LMS-RAM	1.21
Add	0.46
FIR-W-RAM	1.08

Table 5.3: *Power Consumption of WUD block components*

Core (No of Taps)	Dynamic Power (mW)	% Difference
CON (64)	21.01	-
VAR LMS (64)	21.89	+4.19
VAR LMS (56)	20.77	-1.14
VAR LMS (48)	19.63	-6.57
VAR LMS (40)	18.48	-12.04
VAR LMS (32)	17.37	-17.33
VAR LMS (24)	16.20	-22.89
VAR LMS (20)	15.62	-25.65
VAR LMS (16)	15.05	-28.36

Table 5.4: *Power Consumption analysis for different equaliser tap-lengths*

21.01mW. As has been shown by the analysis in the previous chapter, the weight update block consumes the most power, this was explained as it is the most computationally complex. The results in Table 5.2 show the power consumed by each of the main functional blocks in the adaptive core. The FIR filter is the next highest, containing a single multiply-accumulate block, and the error calculation block is the lowest in terms of power consumed. The results in Table 5.3 show the power consumed by each of the functional blocks within the WUD block.

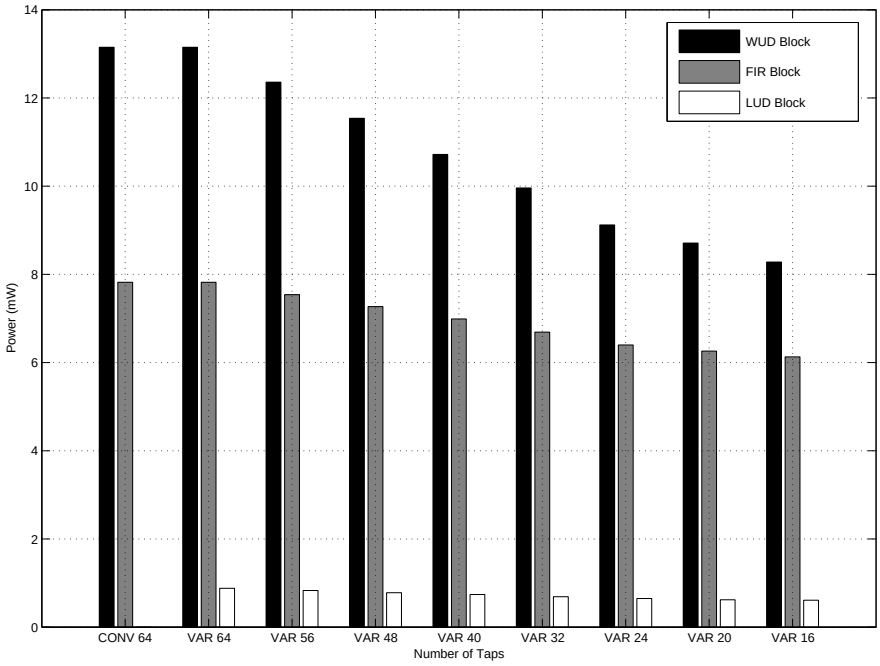


Figure 5.9: *Power Consumed by functional block for different tap lengths*

5.6.2 Power profile for variable length adaptive core

Initial power results for the variable length LMS implementation are shown in Table 5.4. A 64-tap variable length adaptive filter was analysed with step-size μ = 0.01. In this case M = 18 and P = 4 taps/segment. Performance analysis was initially carried out to establish the power profile for a number of discrete tap lengths. Table 5.4 clearly shows an increase in power saving achieved as the number of taps in operation decreases, as would be expected. At this point the LUD block is in operation but the WUD block is not being allowed to alter the number of filter coefficients it calculates for presentation to the filter block. This was done by tying off the length control signal from the LUD block and applying a fixed value to the WUD block controller. These results therefore represent the steady state power consumed by the core once the length update algorithm has converged to an optimum number of taps for operation.

A breakdown of the power consumed by different blocks within the design shows where the overhead in this technique lies and where the greatest power savings are made. Table 5.5 and the graph in Fig. 5.9 show the power consumed by the WUD block, LUD block and segmented FIR block within the variable length adaptive filter and illustrates the block level power dissipation

Core (No of Taps)	WUD Block (mW)	LUD Block	FIR Block	Error Calc
CON (64)	13.15	-	7.82	0.041
VAR LMS (64)	13.15	0.88	7.82	0.041
VAR LMS (56)	12.36	0.83	7.54	0.039
VAR LMS (48)	11.54	0.78	7.27	0.037
VAR LMS (40)	10.72	0.74	6.99	0.035
VAR LMS (32)	9.96	0.69	6.69	0.032
VAR LMS (24)	9.12	0.65	6.40	0.030
VAR LMS (20)	8.71	0.62	6.26	0.028
VAR LMS (16)	8.28	0.61	6.13	0.027

Table 5.5: *Power Consumption analysis for functional blocks*

Core (No of Taps)	Dynamic Power (mW)	% Difference
CON (64)	13.15	-
VAR LMS (64)	13.16	+0.07
VAR LMS (56)	12.36	-6.01
VAR LMS (48)	11.55	-12.17
VAR LMS (40)	10.72	-18.48
VAR LMS (32)	9.96	-24.26
VAR LMS (24)	9.11	-30.72
VAR LMS (20)	8.71	-33.76
VAR LMS (16)	8.28	-37.03

Table 5.6: *Power Comparsion for CON WUD and VAR-LMS WUD Blocks*

for various tap lengths.

5.6.3 Comparison between conventional and variable length core

Comparison of the power consumed by the CON 64 and VAR 64 cores shown in Table 5.4 highlights the overhead in the implementation of the length update algorithm. The difference in total dynamic power between the cores is 0.88mW which is an increase of 4.19%. This is accounted for by additional logic in the LUD block and an extra error calculation. From this it can be seen that the penalty of the overhead is easily compensated for by the potential saving

Core (No of Taps)	Dynamic Power (mW)	% Difference
CON (64)	7.82	-
VAR LMS (64)	7.82	-
VAR LMS (56)	7.54	-3.54
VAR LMS (48)	7.27	-7.05
VAR LMS (40)	6.99	-10.65
VAR LMS (32)	6.69	-14.41
VAR LMS (24)	6.40	-18.11
VAR LMS (20)	6.26	-19.98
VAR LMS (16)	6.13	-21.61

Table 5.7: *Power Comparison for CON and VAR-LMS FIR Filter Blocks*

in power that can be achieved.

A reduction in power consumed by the WUD block and the FIR block can be seen as the tap length decreases. This being the aim of the variable length core in that the computations carried out in the arithmetic blocks within each are reduced directly as the number of taps in operation is reduced. Table 5.6 shows the power consumption of the WUD block within the CONV filter and within the VAR-LMS filter for different tap lengths. Table 5.7 shows the power results for the FIR block in each implementation. The power reduction in percentage terms for both blocks individually can clearly be seen. These results demonstrate the effectiveness with which the variable length algorithm targets the WUD block as the highest consumer of power and also provides a reduction in power in the FIR filter block. A reduction in power of between 6% and 37% can be achieved in the WUD block and between 3.5% and 21.6% in the FIR filter block over the range of tap lengths.

It can also be seen that the overhead introduced by the VAR-LMS implementation in terms of power consumed, is very low in relation to the CONV adaptive filter and does not change as the number of taps decreases. The power overhead is therefore easily offset by the benefit of being able the optimise the tap length of the filter.

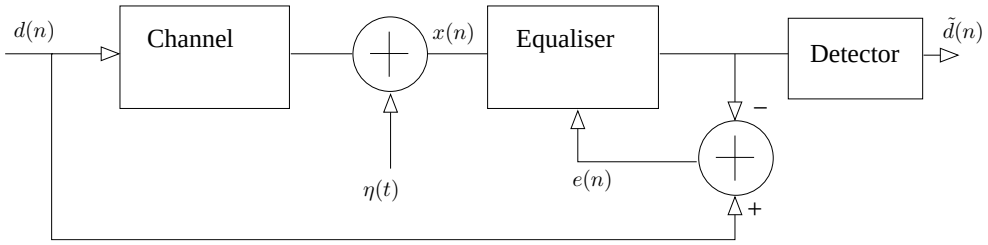


Figure 5.10: *Block Diagram of Equaliser System Model*

E/N_0	No of Taps	Dynamic Power (mW)
5	12	14.80
10	16	15.32
15	20	15.87
20	28	16.98
25	36	18.07
30	52	20.26

Table 5.8: *Power Consumption analysis for different input* E/N_0

5.6.4 Power analysis for dynamic tap length optimisation

When the length update algorithm is employed to determine the optimum number of taps, fundamentally power is saved due to the fact that a filter of greater fixed length will not achieve a better steady state MSE than the optimised case. Again the 64-tap variable length adaptive filter core was analysed with step-size μ = 0.01, M = 18 and P = 4 taps/segment. The ASE is accumulated over 100 samples, i.e. the length update algorithm will increase or decrease the number of filter taps in operation every 100 data symbols. To achieve the optimisation operation of the length update algorithm the core is used to equalise a channel of fixed profile with varying levels of noise present. This is quantified by generating and input signal $x(n)$ adding calculated E/N_0 levels and allowing the filter to converge to a steady state. The system model used to gather these results is shown in the diagram of Fig.5.10. The channel tap coefficients are set to random values exponentially decreasing in average power with exponent powers of 0 to -4 in unity steps.

The graph of Fig.5.11 shows the optimised steady state filter length of the VAR LMS core for various input E/N_0 levels. Again, in this case M = 18 and P = 4 taps/segment. The power consumption results can be seen in Table 5.8.

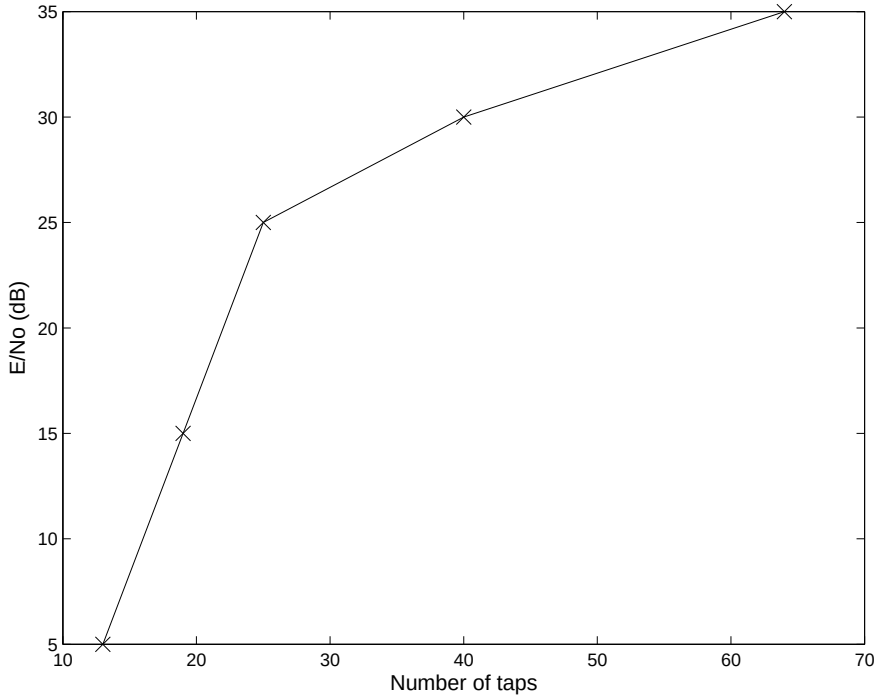


Figure 5.11: *Number of taps for varying* E/N_0

Core	Area (μm^2)	% Difference
CON	451255.0	-
VAR LMS	479232.81	+6.2

Table 5.9: *Area analysis for different adaptive filter cores*

5.6.5 Area comparison and overhead

Area results are shown in Table 5.9. These results show there is little area penalty in the implementation of the variable length algorithm. The overhead is present in the design due to the logic required by the LUD block, which consists of two accumulator blocks, a comparator, the FSM control and an up/down counter. There is also a minor addition in logic terms made to the WUD block FSM control for the purpose of timing and synchronisation. It can be seen that the overhead in core area amounts to an increase of 6.2% in the VAR LMS implementation over the CON implementation. The clock frequency chosen in the verification of these implementations reflects the frequency that the synthesis constraints will comfortably allow without timing violation in critical paths. A reduction in the core clock frequency used will result in a reduction of power consumed accordingly.

5.7 Summary

This chapter has presented a novel architectural VLSI implementation of a dynamically length optimised LMS adaptive filter for use in channel equalisation. The technique was implemented in a 64 tap adaptive filter core and demonstrates length optimisation with varying input E/N_0. The results demonstrate a power saving is achieved by optimising the number of taps in operation. Results have shown a power saving of 28% can be achieved for a variable length architecture optimised to 16 taps over a conventional 64 tap fixed length adaptive filter architecture. It has also been shown that the low-complexity of the additional circuitry needed for the variable length adaptive filter presents minimal overhead for this architecture.

Chapter 6
Low Power LMMSE receiver architecture

This chapter proposes the use of the variable length adaptive filter core developed and explained in detail previously in Chapter 5 for use in a hardware optimised low power implementation of the adaptive LMMSE receiver. A novel HDL implementation of this architecture is presented.

This chapter is organised into seven sections. Section 6.1 introduces the motivation for the concept of the adaptive LMMSE receiver and provides detail of accompanying research on the topic. Section 6.2 describes the system model used for verification of the LMMSE receiver design. Sections 6.3 and 6.4 give an overview and analysis of the techniques used in the operation of the receiver chosen for implementation with sections 6.5 and 6.6 describing the hardware architecture design. Sections 6.7 and 6.8 present the design-flow and power consumption results collected from the implementation of this architecture.

6.1 LMMSE Techniques

As has been discussed in Chapter 3, linear minimum mean square error (LMMSE) techniques can be used to obtain near-far resistant receivers in DS-CDMA systems and overcome the limitations of conventional RAKE receivers. Investigation of such receivers is important as their use has been proposed for application in wideband-CDMA (W-CDMA) systems. The conventional approach in CDMA systems is to ignore multiple-access interference (MAI) and the near-far problem which in turn constrain the system capacity. As has been discussed, a more efficient way to detect multiple users is to implement some form of multi-user detection. With optimal multi-user receivers proving too complex to realise practically, several sub-optimal multi-user receivers have been proposed [44] [45] [46]. Among the group of sub-optimal receivers, the adaptive LMMSE has been proposed for DS-CDMA systems [71] [72] [73]. The LMMSE receiver will minimise the mean square error between the filter output and the true transmitted data symbols. The coefficients of the LMMSE receiver are dependent on the

channel coefficients of all users and must be adapted dynamically as the channel changes. In a rapidly fading channel the LMMSE receiver must be adapted continuously and will suffer convergence problems if the channel fades too fast. However, the LMMSE receiver can still be used if the rate of fading is sufficiently low in relation to the data rate.

The authors of [53] and [74] present a modified LMMSE receiver structure that employs an adaptive-LMMSE technique to improve the performance of a conventional RAKE receiver. This modified LMMSE receiver assumes that the channel coefficients of the desired user are estimated as is the case in the conventional coherent RAKE receiver.

In this chapter it has been chosen that the adaptive LMMSE receiver proposed in [74] will be implemented using the variable length adaptive filter core developed previously in Chapter 5. This receiver has been chosen for implementation due to its use of the LMS algorithm to calculate the LMMSE filter coefficients and its suitability to the use of the variable length LMS algorithm. A conventional adaptive LMMSE receiver will be implemented for comparison to our new adaptive LMMSE receiver. This will provide a measure of performance the optimisations proposed can achieve.

6.2 Modeling of the adaptive LMMSE receiver

Modeling of the adaptive LMMSE receiver core is carried out to functionally verify the operation of the RTL implemented in development of the new adaptive LMMSE core proposed. Each of the functional blocks within the new implementation has been functionally verified and it is not the purpose of the application of this variable length adaptive filter to introduce performance improvement to the receiver from a channel utilisation point of view but to optimise the core hardware and reduce the total power consumed over time.

6.2.1 System Model

The system model is defined to allow consistent testing of the adaptive LMMSE receiver core. A standard DS-CDMA model for a system with K users and L propagation paths is assumed. As has been outlined in section 3.2, the kth user data bits are spread by multiplying the data by a binary pseudo-random noise (PN) sequence. This is expressed as

$$s_k(t) = \sum_{j=0}^{G-1} s_k(j) p(t - jT_c) \tag{6.1}$$

where G is the number of chips per symbol, $s_k(j)$ is the jth chip of the kth user, $p(t)$ is the chip waveform, T_c is the chip interval and t is continuous time.

The received signal can be expressed as

$$r(t) = \sum_{n=0}^{N_b-1} \sum_{k=1}^{K} A_k h_k^{(n)} s_k(t - nT) * g_k(t) + \eta(t) \tag{6.2}$$

where the signal amplitude A_k is given by

$$A_k = \sqrt{\frac{E_k}{T}} \tag{6.3}$$

and where N_b is the number of received symbols, K is the number of users, $h_k^{(n)}$ is the nth transmitted data symbol, $s_k(t)$ is the kth user's spread signal, E_k is the energy per chip and T is the symbol period.

$\eta(t)$ is additive white Gaussian noise (AWGN) and the channel impulse response is given by

$$g_k(t) = \sum_{l=1}^{L_k} g_{k,l}^{(n)} \delta(t - \tau_{k,l}) \tag{6.4}$$

for the kth user. L_k is the number of propagation paths, $g_{k,l}^{(n)}$ is the complex gain of the kth user's lth path during the nth symbol period. $\tau_{k,l}$ is the propagation delay and $\delta(t)$ is the Dirac's delta function.

The received signal can therefore be written as

$$r(t) = \sum_{n=0}^{N_b-1} \sum_{k=1}^{K} \sum_{l=1}^{L_k} A_k h_k^{(n)} g_{k,l}^{(n)} s_k(t - nT - \tau_{k,l}) + \eta(t) \tag{6.5}$$

This received signal is then sampled at

$$T_s^{-1} = \frac{SG}{T} \tag{6.6}$$

where S is the number of samples per chip.

6.2.2 Matlab Model

The system model described was used to create Matlab code which was used to generate test vectors for the RTL implementation as well as to provide input vectors for the Matlab model of the device under test. The Matlab code developed for the variable length LMS core is reused here for the model of the adaptive LMMSE receiver core and forms the main part of the receiver functionality.

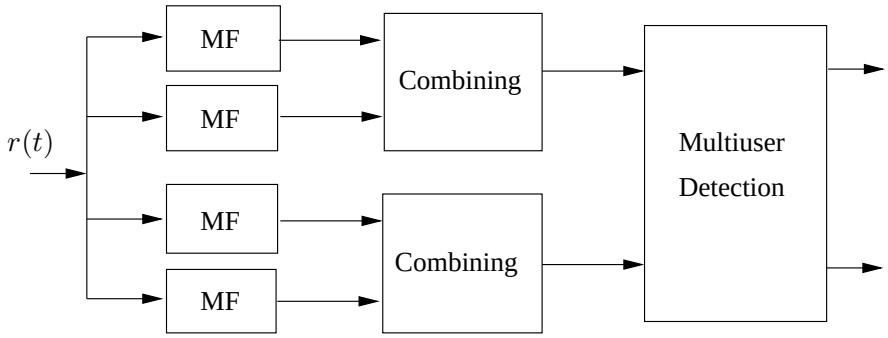
6.3 Adaptive LMMSE Receiver Architectures

In order to evaluate the performance of the new adaptive LMMSE receiver a conventional adaptive LMMSE core based on the work presented in [74] is studied. The LMMSE receiver core is implemented using verilog developed and verified in previous chapters for lower level functional blocks.

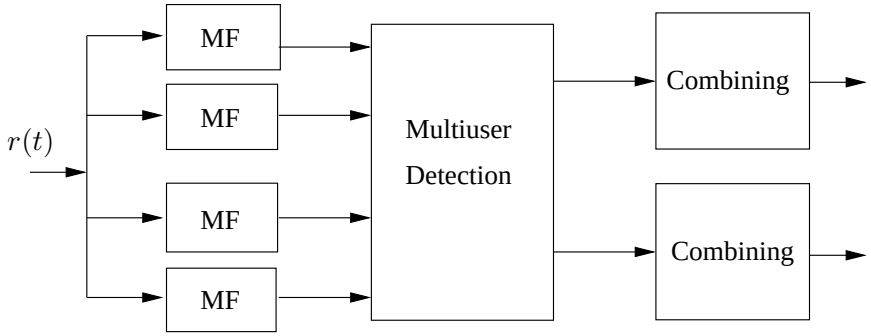
6.3.1 Pre-combining and Post-combining Receivers

The architectures proposed for the implementation of the adaptive LMMSE receiver fall into the categories of either the pre-combining interference suppression type receiver or the post-combining interference suppression receiver. Functionally these architectures can be seen in the diagrams of Fig.6.1. As can be seen, multi-user filtering can take place before combining the multi-path components or after it.

The conventional LMMSE receiver, or post-combining LMMSE receiver, will minimise the mean square error between the receiver output and the true transmitted data symbols. These receivers are capable of handling both inter-path and inter-channel interference under severe near-far situations. As this type of receiver requires the channel coefficients of all users and



(a)



(b)

Figure 6.1: *a) Post combining and b) Pre combining LMMSE receivers*

must be adapted as the channel changes. If the channel is changing rapidly then the receiver coefficients must be updated continuously. These receivers can have convergence problems in this instance but this can be controlled by modifying the optimisation criterion of the coefficient update algorithm.

Optimisation of this post-combining receiver has led to the development of the pre-combining LMMSE receiver which minimises the mean square error between the receiver output and the cross-correlation of the data sequence with the channel coefficients for each multi-path component. It can therefore be seen that this receiver requires the knowledge of the channel coefficients for the user and they must be estimated. In this case only the averaged channel profiles of all users are required and as the path delays and the average channel profiles change relatively slowly, the adaptation performance of the adaptive pre-combining LMMSE receiver is significantly less critical than that of the performance of the adaptive post-combining

LMMSE receiver [75]. The arrangement of the functional blocks in both the pre-combining and post-combining receiver is seen in Fig.6.1. A bank of matched filters (MF) is used for de-spreading and synchronisation.

An analysis of the performance characteristics of the pre-combining and post-combining LMMSE receivers is presented in [48] in fading multi-path channels. The bit error probability (BEP) of the pre-combining LMMSE detector is compared to the BEP of the post-combining LMMSE detector. It is also stated that the post-combining LMMSE receiver has a greater channel capacity potential but due to convergence problems, can only be used to exploit this property at very high data rates and when the rate of fading is low. Pre-combining LMMSE receivers on the other hand do not have such stringent convergence requirements and no restriction on their use in fading channels.

The effect on performance for both of these approaches has been analysed in [76]. This work has shown that the order in which multi-user filtering and multi-path combining is carried out does not have a significant effect upon the decorrelator output BEP as long as the number of users (K) and the number of multi-path components (L) is low. If both parameters become larger, then combining prior to multi-path filtering becomes most favoured, this does however make channel estimation more difficult.

6.4 Precombining adaptive LMMSE receiver analysis

As has been explained, the precombining LMMSE receiver performs multi-user detection before the combining stage which operates then at the symbol level. The adaptive case of the precombining LMMSE receiver requires knowledge of the spreading sequence, data bits and channel coefficients of each multi-path component for which estimates are needed.

The following analysis is carried out for the adaptive LMS version of the precombining LMMSE receiver and is based on the structure of the conventional RAKE receiver. This is the done so that channel estimation information is available for the calculation of each multi-path component. In this adaptive LMMSE receiver each receiver finger is adapted independently to suppress multiple access interference.

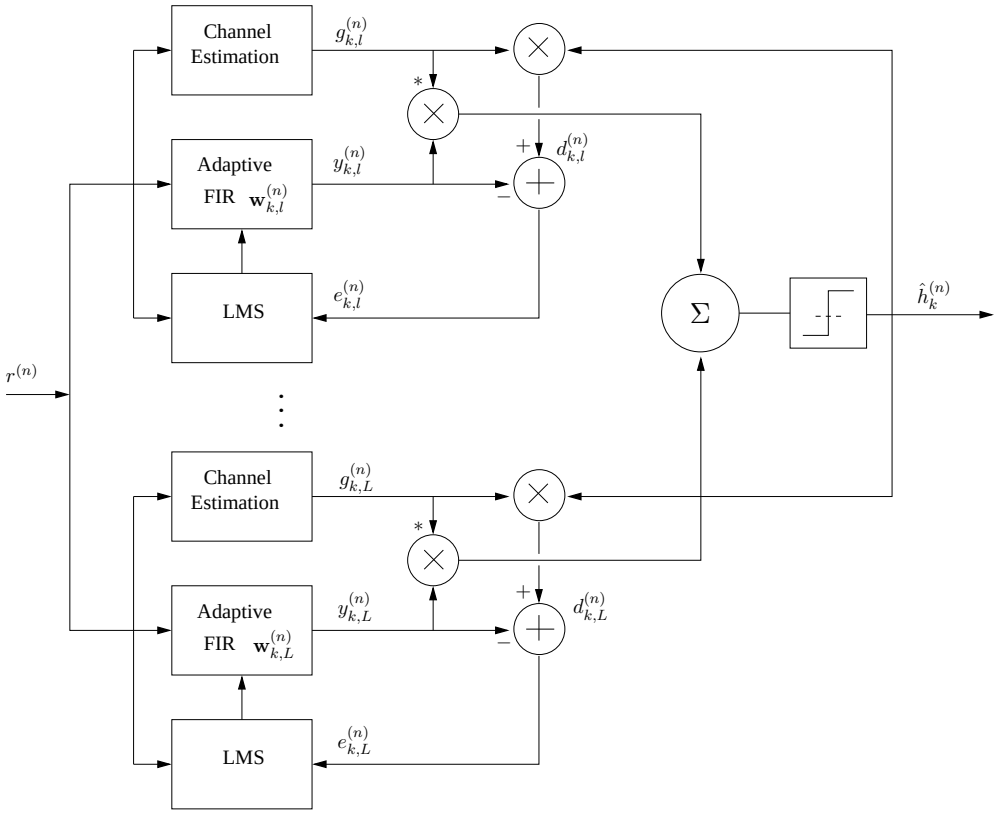


Figure 6.2: *Conventional adaptive LMMSE receiver functional blocks*

A block diagram of the adaptive LMMSE-RAKE structure is shown in Fig.6.2. The output of the lth receiver branch can be expressed as

$$y_{k,l}^{(n)} = \mathbf{w}_{k,l}^{H_{(n)}} \mathbf{r}^{(n)} \tag{6.7}$$

and the decisions are made in the adaptive LMMSE receiver according to

$$\hat{h}_k^{(n)} = \text{sgn}\left(\sum_{l=1}^{L} g_{k,l}^{*_{(n)}} y_{k,l}^{(n)} \right) \tag{6.8}$$

where $\hat{h}_k^{(n)}$ represents the estimate of the original transmitted data symbol, sgn$(\cdot)$ represents the signum function and $g_{k,l}^{*_{(n)}}$ represents complex conjugate of the estimated channel coefficient.

The LMMSE filter coefficients are found by satisfying the MSE criterion (E[$|e_{k,l}^{(n)}|^2$]), which requires the solution of the Weiner-Hopf equation to find the optimal filter coefficients. The estimation of gradient of the error function and using the gradient of steepest decent are used in the solution of the optimal filter coefficients. The filter weights are updated iteratively using the update equation

$$\mathbf{w}_{k,l}^{(n+1)} = \mathbf{w}_{k,l}^{(n)} - \mu\nabla_{k,l} \tag{6.9}$$

where μ is the update step-size and $\nabla_{k,l}$ is the gradient of the MSE with respect to the filter coefficients. Using the stochastic approximation of the steepest decent algorithm such that

$$\nabla_{k,l} \approx -2\mathbf{r}(g_{k,l}\hat{h}_k)^* + 2\mathbf{r}y_{k,l}^* \tag{6.10}$$

it can be shown[48] that the LMS algorithm applied here takes the form

$$\begin{aligned}\mathbf{w}_{k,l}^{(n+1)} &= \mathbf{w}_{k,l}^{(n)} + 2\mu\mathbf{r}^{(n)}\left(g_{k,l}^{(n)}\hat{h}_k^{(n)} - y_{k,l}^{(n)}\right)^* \\ &= \mathbf{w}_{k,l}^{(n)} + 2\mu\mathbf{r}^{(n)}e_{k,l}^{*(n)}\end{aligned} \tag{6.11}$$

The error signal

$$e_{k,l}^{(n)} = d_{k,l}^{(n)} - y_{k,l}^{(n)} \tag{6.12}$$

calculated using the difference between the reference signals and the LMMSE filter output is used to update the filter weights using the LMS algorithm.

At this point the receiver vector can be split into adaptive and fixed component parts such that

$$\mathbf{w}_{k,l}^{(n)} = s_{k,l} + \mathbf{x}_{k,l}^{(n)} \tag{6.13}$$

where $\mathbf{x}_{k,l}^{(n)}$ is the adaptive filter weight vector and $s_{k,l}$ is the fixed spreading sequence for the kth users lth path.

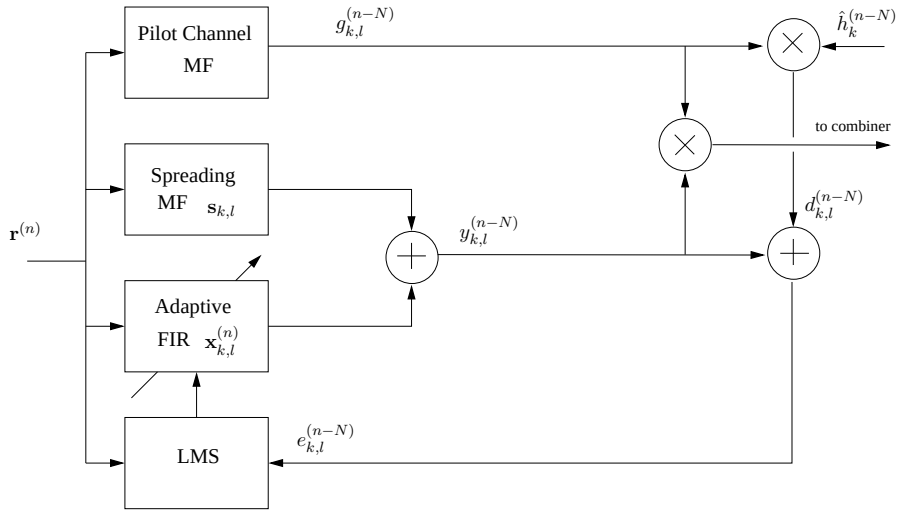


Figure 6.3: *Block diagram of single adaptive LMMSE receiver finger*

The functionality of the conventional RAKE receiver can now be included for optimisation in the implementation of the adaptive LMMSE receiver finger with the updates of the adaptive component now expressed as

$$\begin{aligned}\mathbf{x}_{k,l}^{(n+1)} &= \mathbf{x}_{k,l}^{(n)} + 2\mu_{k,l}^{(n)}\left(g_{k,l}^{(n)}\hat{h}_k^{(n)} - y_{k,l}^{(n)}\right)^*\mathbf{r}^{(n)} \\ &= \mathbf{x}_{k,l}^{(n)} + 2\mu_{k,l}^{(n)}e_{k,l}^{*(n)}\mathbf{r}^{(n)}\end{aligned} \tag{6.14}$$

where $\mu_{k,l}^{(n)}$ is the step-size parameter for the kth users lth path. The block diagram of a single receiver finger in the adaptive LMMSE receiver can be seen in Fig.6.3.

6.5 Implementation of conventional adaptive LMMSE core

In order to evaluate the performance of the our new LMMSE receiver core, a conventional LMMSE receiver core based on the principles discussed in section 6.4 is developed and implemented. The conventional implementation is assembled using the equivalent functional blocks detailed in previous chapters, such as the adaptive LMS core detailed in section 4.3.

Mathematically, the output of the pre-combining LMMSE filter requires the spreading codes and delays of all users in its computation. Processing intensive matrix inversion of the channel covariance matrix is also required which is ideally avoided in practical implementation because

of its computational burden. This, along with the fact that the spreading codes and delays of all users may not be known at the user receiver, is the reason that the LMMSE receiver is normally solved iteratively for each user by some adaptive algorithm such as the LMS algorithm. As a result, the implementation of this type of receiver is referred to as the adaptive LMMSE receiver.

An adaptive LMS based pre-combining LMMSE receiver is presented in [74] with a similar structure to that of a conventional RAKE receiver. This practical solution results in having separate LMMSE receivers for each multi-path component and for this reason is referred to as an adaptive LMMSE-RAKE receiver. To satisfy the MSE solution, this adaptive implementation of the pre-combining LMMSE receiver therefore needs to acquire the knowledge of the spreading sequence timing and channel coefficients for each multi-path component to generate an estimate of the data symbols.

6.5.1 Adaptive LMMSE finger structure

This adaptive LMMSE core has multiple receiver fingers in the same manner as a conventional RAKE receiver, the arrangement for one of which can be seen in Fig.6.3. Implementation of each receiver finger is carried out by arranging and connecting the components developed in previous chapters to carry out the adaptive LMMSE-RAKE computation and designing the matched filters. Once one finger has been implemented it can be replicated to carry out the detection and synchronisation of data symbols for each multi-path component and the output of each finger combined to resolve the true estimated data symbol.

This implementation consists of an LMS weight update block (WUD) which uses the data input samples and an error signal to calculate new coefficients, and a FIR filter block, which employs the coefficients calculated by the LMS WUD block. This is used for detection using the calculated receiver coefficients. Matched filters (MF) are also present and are used for channel estimation and de-spreading. More detail about these blocks is given:

- WUD: The LMS weight update block is replicated from 4.3.2 and carries out the calculation of the receiver filter coefficients iteratively using the LMS algorithm.
- COEFF. FILTER: The calculated receiver coefficients are employed by this FIR filter to detect the user data symbols. Its implementation is the same as the FIR filter structure described in 4.3.1.

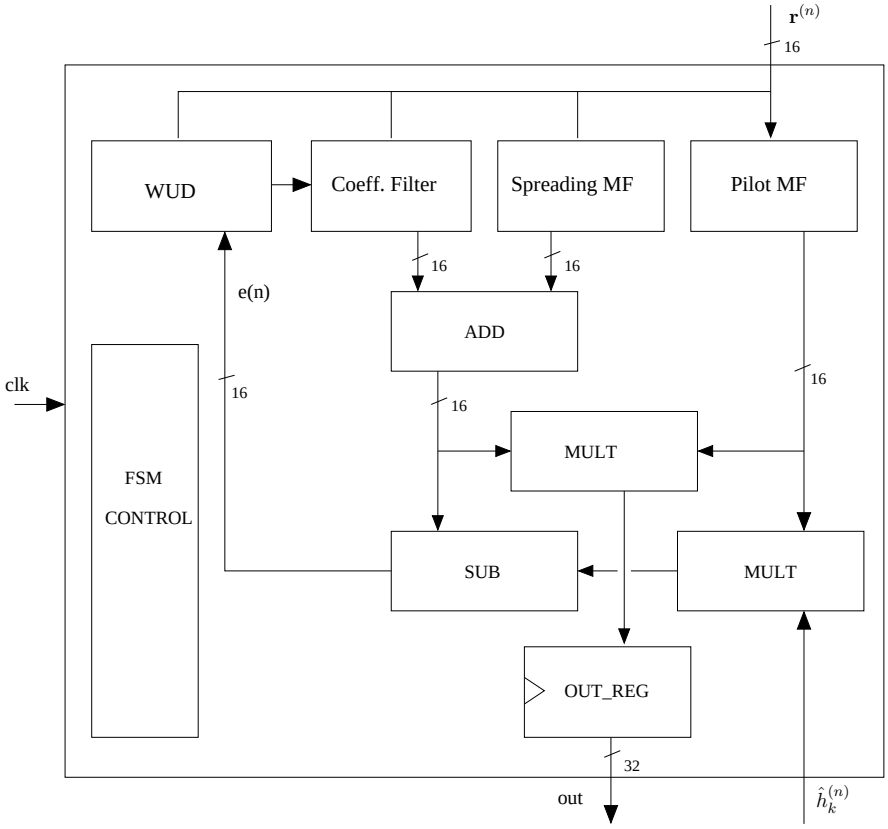


Figure 6.4: *Functional diagram of adaptive LMMSE finger structure*

- SPREADING MF: The spreading matched filter is used for de-spreading and synchronisation of the received multi-path signal. The filter taps are matched to the PN spreading code.

- PILOT MF: The pilot channel matched filter is used to determine an estimation of the wireless channel coefficients. This uses the pilot channel code to estimate the channel coefficients for the lth path. Both the MF sub-blocks are based on an FIR filter structure in terms of their hardware implementation.

- AU: There are a number of distributed arithmetic units for calculation of the required adaptive LMMSE solution components. There are two adder units and a multiplier, all clocked at the symbol rate.

- FSM CONTROL: This finite state machine control block synchronises timing of all the components according to the symbol rate and determines the delay synchronisation of the matched filters.

The functional blocks within the adaptive LMMSE receiver finger implementation can be seen in the diagram of Fig.6.4.

6.5.1.1 Digital Matched Filtering

Looking at the instantiation of the spreading matched filter, the matched filter is used to calculate a value for the correlation between the received spread signal and the known PN sequence with which the transmitted data was originally spread. Its structure is that of an FIR filter that has the same number of taps as the spreading code has chips. It therefore performs the correlation

$$y(n) = \sum_{j=0}^{G-1} s_j r(n+j) \tag{6.15}$$

where s_j is the jth chip, $r(n)$ is the spread sequence sample and G is the number of chips per symbol i.e. spreading factor or processing gain.

The digital matched filter is the preferred method of code acquisition and synchronisation in a DS-spread spectrum system such as in W-CDMA where relatively long spreading sequences are proposed [77]. This is due to their flexibility and power efficiency as a significant proportion of the power budget in the baseband processing is devoted to the digital matched filter. Several implementations of the digital matched filter have been proposed [78] [79] [80] and [81] with low power implementations specifically discussed in [80] and [81].

A diagram of the spreading matched filter block structure can be seen in Fig.6.5. This filter implementation consists of a single memory block (IN RAM) for storing input data vectors $r(n)$, a PN code generator matched to the code for the particular user, a control block (CONTROL), a number of single word registers for clock synchronisation and an arithmetic block (MAC).

The implementation of the pilot matched filter is structurally identical to that of the spreading matched filter but with its purpose being to determine a value of the correlation between the received pilot signal and the predetermined pilot channel code known to both the receiver and the transmitter.

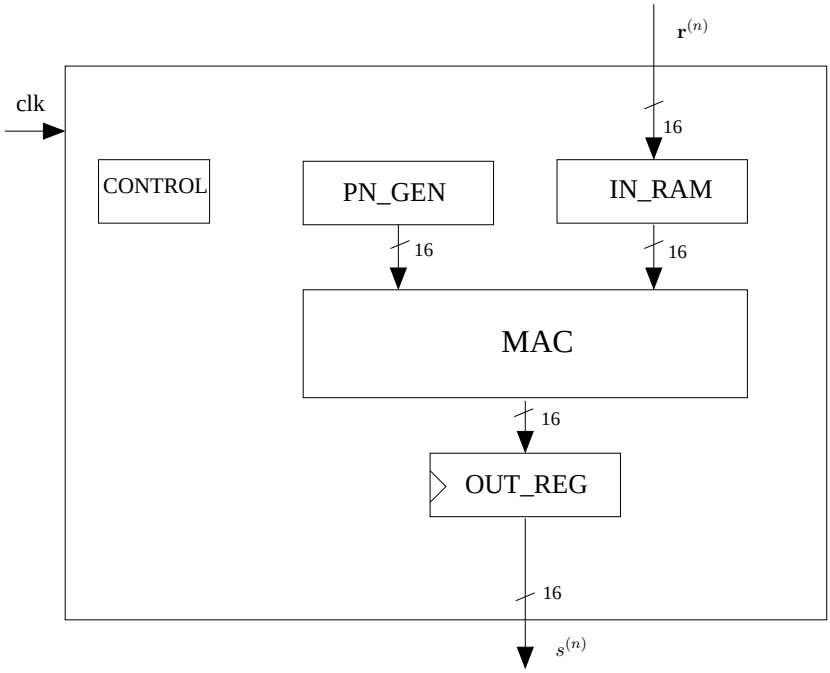


Figure 6.5: *Functional diagram of digital matched filter structure*

6.6 Implementation of new adaptive LMMSE core

Following on from the implementation in the previous section, the adaptive LMMSE receiver is modified to include the variable length adaptive core, developed and verified in section 5.4, in its implementation. In each adaptive LMMSE-RAKE finger the LMS WUD block can be replaced by the variable tap length LMS WUD block. Use of this variable length LMS algorithm will optimise the computation required for calculation of receiver coefficients in every finger of the receiver and therefore reduce the overall power consumed.

6.6.1 New adaptive LMMSE finger structure

Taking the variable length LMS weight update functional block and introducing it into the adaptive LMMSE receiver finger is a relatively simple process having already developed the conventional implementation. The new WUD block has been designed to be a drop-in replacement and will calculate the required error signals for the length update control algorithm internally. There are no algorithmic changes to its operation from that described in section 5.4. The FIR filter block which uses the receiver coefficients calculated by the WUD block is also the same as that implemented in section 5.4. All other blocks within a given receiver finger will

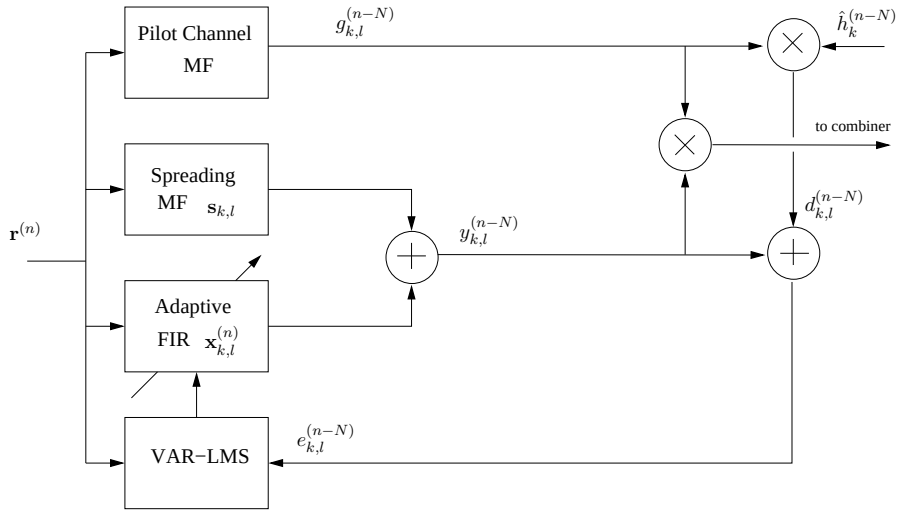


Figure 6.6: *Single adaptive LMMSE receiver finger with variable length LMS functionality*

remain the same as those in the conventional implementation for the purposes of this analysis. There are no changes needed to the FSM controller block as timing of the main functional block in the receiver finger remains the same.

The arrangement of functional blocks in the new adaptive LMMSE receiver finger can be seen in the diagram of Fig.6.6. This implementation is used to verify the use of the variable length LMS algorithm in this application and to determine the power saving that can be made in comparison to the power consumed by the conventional implementation. The following sections present an analysis of the results found by making this comparison.

6.7 Design methodology

Again, the basic design flow is the same as that detailed in section 4.5 and is described briefly here to define the specific flow for this implementation. Conventional and low power architectures are coded in Verilog hardware description language at register transfer level (RTL). Once the functional specification of the design is verified against a Matlab model, synthesis is then carried out using either Synopsys Design Compiler (dc-shell) or Cadence BuildGates to convert the RTL model into a gate level netlist. A standard delay format (SDF) file is also generated by the synthesis tool to provide gate level timing. Design Compiler also generates a timing constraint file in SDF format. Post-synthesis netlist verification is then carried out determine if the design still meets functional and timing specifications at the gate

level using the Cadence Verilog-XL simulator. If any specification is not met the RTL code and/or the timing constraints used during synthesis will be modified. This loop is carried out until the specification is met entirely and no violations are reported during simulation.

6.8 Results

Two adaptive LMMSE receiver cores have therefore been implemented, the first being the conventional (CONV) and secondly the new implementation using the variable length LMS algorithm (OUR). Both have been analysed in terms of area usage and power consumption. The cores were designed using Verilog HDL and then synthesised using Design Compiler^{TM} targeting the UMC 0.18μ standard cell CMOS library. The requirements of the synthesis were identical for all cores to allow consistent power consumption and area usage comparisons. A netlist was created for each core and back-annotated netlist simulations for a uniformly distributed random input bipolar binary data of 10000 symbols using Verilog-XL^{TM} simulator were performed and verified against the Matlab^{TM} receiver core model. The resulting data, including switching activity of the circuit nets was then used by Synopsys DesignPower^{TM} to determine power consumption for the different adaptive filter cores. In all of the above stages a clock frequency of 100 MHz and a supply voltage of 1.8 Volts were used.

A minimum symbol rate of 16 kbits/s is achieved using a 31 chip Gold code was used in simulation with an equal energy two-path ($L = 2$) channel and a maximum delay spread of 10 chip intervals. The number of users K was kept low for this analysis and an unmodulated pilot channel was assumed to be present with equivalent energy to that of the user data channels. A five tap FIR filter is used to model the fading channel. The channel tap coefficients are set to random values exponentially decreasing in average power with exponent powers of 0 to -4 in unity steps.

6.8.1 Power Consumption for Conventional LMMSE core

An analysis of the power consumed by the conventional adaptive LMMSE core is made for comparison against the new receiver core which uses the VAR-LMS adaptive filter. The functional blocks consuming the highest power are identified. A fixed 32-tap adaptive filter was used with step-size μ = 0.01. The total power consumed by the conventional adaptive LMMSE core implementation of one receiver finger is 32.97mW. Power results are shown in

Core type	Dynamic Power (mW)
CONV	32.97

Table 6.1: *Power Consumption for conventional adaptive LMMSE core*

Block	Dynamic Power (mW)
LMS WUD	9.96
FIR	6.69
Pilot MF	6.69
Spread MF	6.69
Mult	1.24
Mult	1.24
Add	0.23
Sub	0.23
Control	0.04

Table 6.2: *Power Consumption for functional blocks in adaptive LMMSE core*

Table 6.1 for the total power consumed by the receiver finger using conventional processing blocks. The results quoted do not include the combining stage of the receiver core to allow direct comparison to be made between critical components in each implementation.

As has been shown by the analysis in previous chapters, the LMS weight update block consumes the most power, this is explained by the fact it is the most computationally complex. The results in Table 6.2 show the power consumed by each of the main functional blocks in a single receiver finger. The FIR filter is the next highest, containing a single multiply-accumulate block as well as memory locations for storing received signal vectors. The pilot matched filter and spreading code matched filter are then next, followed by the multiply and addition blocks, then the control FSM being the lowest in terms of overall power consumed.

Scaling the implementation to multiple adaptive LMMSE receiver fingers yields the power results shown in Table 6.3. It should be noted that the combiner used in this analysis is a behavioural verilog model included in the test bench for simulation purposes and is not synthesised. These results confirm the preconception that the addition of each receiver finger will linearly increase power consumption. Use of the variable length LMS algorithm will

Fingers	Dynamic Power (mW)
1	32.97
2	65.94
3	98.91
4	131.88
5	164.85
6	197.82

Table 6.3: *Power Consumption for multiple conventional adaptive LMMSE fingers*

Core type	Dynamic Power (mW)
OUR	29.77

Table 6.4: *Power Consumption for our adaptive LMMSE core*

clearly target the block within the receiver finger that consumes the greatest amount of power.

6.8.2 Power Consumption for New LMMSE core

Initial power results for the variable length LMS implementation of a single adaptive LMMSE receiver finger are shown in Table 6.4. A 32-tap variable length adaptive filter was used to calculate receiver coefficients with step-size μ = 0.01 as before. In this case $K = 16$ and $P = 2$ taps/segment. A signal to noise ratio of 10 dB was chosen and the variable length LMS algorithm converged to a tap-length of 14 taps. It can clearly be concluded that the optimisation of the filter tap length has resulted in a power saving in comparison to the results for the conventional implementation given in Table 6.1. The multipliers used in the low power LMMSE implementation have been optimised where possible and are implemented using the non-Booth encoded Wallace tree (*nbw*) type multiplier. This has brought a further power saving to the optimised LMMSE finger and is seen in the results shown. Most notably the power consumed by the FIR filter and matched filter blocks has been reduced when compared to the conventional implementation.

The LMS weight update block still consumes the greatest power but this has been targeted directly by the optimisation of the variable length update algorithm. Due to optimisation of

Block	Dynamic Power (mW)
LMS WUD	8.95
FIR	6.07
Pilot MF	6.1
Spread MF	6.61
Mult	1.06
Mult	1.06
Add	0.23
Sub	0.23
Control	0.04

Table 6.5: *Power Consumption for functional blocks in our LMMSE core*

Fingers	Dynamic Power (mW)
1	29.77
2	59.54
3	89.31
4	119.08
5	148.85
6	178.62

Table 6.6: *Power Consumption for multiple adaptive LMMSE fingers using the length update algorithm*

the number of filter taps in operation, the switching activity in the multiplier is reduced and the accesses to the memory blocks storing coefficients within the receiver coefficient filter are also reduced. This is directly responsible for the reduction in power consumed. The results in Table 6.2 show the power consumed by each of the main functional blocks in the receiver finger. There is of course a slight overhead in this implementation due to the control and error calculation logic required by length update algorithm but this is clearly mitigated by the power savings achieved.

Finally, again scaling the implementation to multiple adaptive LMMSE receiver fingers which use the variable length update algorithm yields the power results shown in 6.6. A significant power saving can be achieved as the implementation is scaled up.

Block	*Conv* (mW)	*Our* (mW)	% Power Reduction
LMS WUD	9.96	8.95	10.1
FIR	6.69	6.07	9.38
Pilot MF	6.69	6.07	9.38
Spreading MF	6.69	6.07	9.38
Mult	1.24	1.06	14.07
Mult	1.24	1.06	14.22
Add	0.23	0.23	0
Sub	0.23	0.23	0
Control	0.04	0.04	0

Table 6.7: *Comparison of Power Consumption for functional blocks in both LMMSE core implementations*

Fingers	*Conv* (mW)	*Our* (mW)	% Power reduction
1	32.97	29.77	9.7
2	65.95	59.54	9.7
3	98.91	89.31	9.7
4	131.88	119.08	9.7
5	164.85	148.85	9.7
6	197.82	178.62	9.7

Table 6.8: *Power Consumption comparison between* Conv *and* Our *for multiple adaptive LMMSE fingers*

6.8.3 Comparison between conventional LMMSE core and new LMMSE core

A reduction in power consumed by the WUD block and the FIR block is seen in the receiver core as the tap length of the coefficient filter is optimised by the length update algorithm. Table 6.7 shows the power results for the receiver blocks in each implementation and the percentage power reduction achieved in the main functional blocks within the design.

Core	Area (mm^2)	% Difference
Conv	1.98	-
Our	2.04	+3.03

Table 6.9: *Area analysis for different LMMSE receiver cores*

6.8.4 Area comparison and overhead

Area results are shown in Table 6.9 and power results are shown in Table 6.8. It can be seen that the overhead introduced by *Our* implementation is very low in relation to the *Conv* adaptive LMMSE implementation. When the design is scaled up to include multiple receiver fingers the power saving follows accordingly. Again, it is important to note that the clock frequency chosen in the design flow for both these receiver implementations is the frequency that the synthesis constraints will comfortably allow without timing violation in critical paths. A reduction in the core clock frequency is entirely acceptable and will result in a significant reduction of power consumed provided the desired output data rate for the given end application is still achieved.

6.9 Summary

This chapter has presented a novel architectural VLSI implementation of an adaptive LMMSE receiver core. The novel application of the variable length LMS adaptive algorithm for calculation of channel coefficients in the LMMSE receiver core was presented and has proved successful. The results demonstrate a power saving is achieved by optimising the number of taps used in the LMMSE filter coefficient calculations. Results have shown a power saving of 9.7% can be achieved using our receiver core in comparison with the conventional adaptive LMMSE implementation. It has also been shown that the low-complexity of the additional circuitry needed for the variable length adaptive filter presents minimal overhead for this architecture.

Chapter 7
Summary and Conclusions

7.1 Introduction

The aim of this thesis is to present the investigations made into low power architectures for the LMS adaptive equaliser and its application in a specific type of LMMSE receiver. The LMMSE receiver is proposed for use in W-CDMA communication systems. Algorithmic methods to reduce the switching activity in critical components is targeted. In particular, reducing the switching activity in the arithmetic units of the FIR filter and the LMS weight update blocks is presented in this thesis. Having optimised the low power implementation of these functional blocks, application in the adaptive LMMSE receiver architecture results in an overall reduction in power consumption.

This chapter is organised into four further sections. Section 7.2 summarises the work presented in each of the previous chapters. Section 7.3 outlines the conclusions reached from results presented in the previous chapters. Section 7.4 briefly outlines the achievements made in the work carried out to complete this thesis and finally Section 7.5 suggests potential ideas for future work that are highlighted by the results obtained.

7.2 Summary

The effects of inefficient power usage by electronic devices are clear. Portable devices are expected to have ever increasing battery life and reduce wasted energy such as that lost for example through thermal or switching effects. At the same time, portable devices must maximise use of the wireless transmission spectrum and it is for this reason W-CDMA techniques are favoured for use in future generation mobile communication systems. The object of this thesis has been to investigate low power architectures for functional blocks within a W-CDMA receiver. Specifically, two methods by which to reduce the power consumed by the adaptive LMS equaliser have been investigated. The most successful of which was then applied directly in the implementation of an adaptive LMMSE receiver. The

two methods studied are the Decorrelating transform and the variable length LMS algorithm, of which, the variable length LMS algorithm was chosen for use in the final LMMSE receiver implementation.

Firstly, this thesis has presented the practical implementation of both these methods for reducing the power consumption specifically in the LMS adaptive filter. In the initial review of published work, these techniques had only been proposed for this application and mathematical analysis presented. Critically, the practical implementation of both methods had until now not been carried out. The benefit of doing so here is that their effectiveness in reducing power consumption can be quantified and suitability can be evaluated for application in the LMS adaptive filter.

In this thesis the novel idea of using the variable length LMS adaptive filter in an adaptive LMMSE receiver has been presented. Both the variable length LMS filter and the adaptive LMMSE receiver have been developed independently and it is the combination of these concepts and the practical implementation of the resulting receiver core that provides the basis for study.

Chapter 2 has outlined general and block specific techniques for reducing the switched capacitance and therefore reducing the power consumption of circuit blocks. Current parallel work in the field has been based around the exploitation of correlation properties in signals, through the optimisation of hardware by algorithmic transformation or by the dynamic switching of unused functional blocks of a circuit during operation. Low power techniques were then reviewed specifically for the direct form FIR filter and the LMS adaptive filter. These methods are mainly targeting the arithmetic block level in the filter implementations. Techniques such as block processing, coefficient segmentation, coefficient re-ordering, approximate processing and multi-rate architectures are considered along with methods for coefficient scaling and alternative data representation.

Chapter 3 begins with an introduction to CDMA principles and explains the motivation for wideband-CDMA communication systems. The interference mechanisms present in single-user and multi-user systems are explained and the methods used to combat them are identified along with the methods available to maximise the use of available channel capacity. It goes on to explain the operation of the RAKE and LMMSE receivers proposed for use in W-CDMA and a review of the techniques proposed for reducing the switched capacitance of these multi-user

receivers is given. These include a power-scalable RAKE core that switches receiver fingers in or out of operation, a multi-code correlator approach and a method proposing the sharing of functional components between all fingers in the receiver.

Chapter 4 presents the implementation of the existing decorrelating transform when applied to the LMS adaptive filter. An outline of its operation is given and the ability of this particular method being applied to the LMS adaptive filter to reduce power consumption is questioned. This method is better suited to fixed coefficient FIR filters as the constant updating of the filter coefficients by the LMS algorithm has a serious detrimental effect on stability. A power reduction is however still achieved as the switching activity in the multiplier and in the RAM blocks is reduced due to the smaller wordlength of the calculated filter coefficients. The implementation developed in this chapter is compared to a conventional implementation of an LMS adaptive filter.

The variable length LMS algorithm is outlined in Chapter 5 and a practical implementation is developed with the aim of analysing the power consumption saving that applying this technique to the adaptive filter will produce. The existing length update algorithm controls the dynamic increase or decrease in the tap-length of the LMS adaptive filter. By doing so unnecessary switching is avoided in the arithmetic blocks present in the FIR filter structure and in the calculation of filter coefficients. Power consumed by memory components is also reduced. An outline of the operation of the length update algorithm is given and a detailed description of its development and implementation is given. A comparison is made between the power consumed by the new implementation and a conventional fixed length LMS adaptive filter.

Chapter 6 presents the application of the variable length LMS adaptive filter in the implementation of a low power adaptive LMMSE receiver. This chapter gives further detail about the operation principles of the LMMSE multi-user receiver and existing architectures that have been proposed. Detail of the implementation of the new implementation is given along with practical considerations that were made in its development. Operation of the new implementation is verified against a model of the system and power results are presented again in comparison with an equivalent conventional implementation of the same receiver.

7.3 Conclusions

This thesis has presented the implementation of two low power techniques when applied to the LMS adaptive filter and proposed the novel use of the variable length LMS algorithm in an adaptive LMMSE receiver architecture. It can be concluded from Chapter 4 that a power saving in the range of 5 - 15 % is achieved for coefficient word-lengths varying from 16 bits to 10 bits when compared to a conventional adaptive filter implementation. At the same time, an area saving of up to 15 % due to optimisations in the design. The reduction in coefficient wordlength and therefore switching activity in the filter multiplier block is responsible for this saving in power.

It can be concluded that the DECOR transform method does not directly target the block consuming the greatest amount of power in this design. The application of the DECOR transform to the LMS adaptive filter also results in severe stability issues and therefore it would potentially be impractical to justify the use of this method of power saving in a real-world application regardless of the proven reduction in power consumption that has been shown here.

The results presented in Chapter 5 have concluded that a power saving is achieved by optimising the number of taps in operation. Results have shown a power saving of 28% can be achieved for a variable length architecture optimised to 16 taps over a conventional 64 tap fixed length adaptive filter architecture. It has also been shown that the low-complexity of the additional circuitry needed for the variable length adaptive filter presents minimal overhead for this architecture.

It can be concluded that when the length update algorithm is employed to determine the optimum number of taps, a reduction in power consumed by the WUD block and the FIR block can be seen as the tap length decreases. This being the aim of the variable length core in that the computations carried out in the arithmetic blocks within each are reduced directly as the number of taps in operation is reduced. Power is saved due to the fact that a filter of greater fixed length will not achieve a better steady state MSE than the optimised case.

The results in Chapter 6 conclude that the novel application of the variable length LMS adaptive algorithm for calculation of channel coefficients in the LMMSE receiver core was presented and has proved successful. The results demonstrate a power saving is achieved by optimising the number of taps used in the LMMSE filter coefficient calculations. Results have shown a power saving of 9.7% can be achieved using our receiver core in comparison with the conventional

adaptive LMMSE implementation. It has also been shown that the low-complexity of the additional circuitry needed for the variable length adaptive filter presents minimal overhead for this architecture.

7.4 Achievements

- A low power LMS adaptive filter architecture is proposed and implemented based on the DECOR transform.
- A low power LMS adaptive filter architecture using dynamic tap length optimisation is presented and implemented.
- The use of the Variable Length LMS adaptive filter is proposed for use in the precombining LMMSE receiver.
- A novel hardware architecture for the precombining LMMSE receiver incorporating the variable length LMS adaptive filter is presented and implemented.

7.5 Future work

This thesis has attempted to provide a thorough investigation into the research proposal outlined in Section 7.1. It is the case however that a number of areas of further interest and contribution have been identified and could be investigated. These areas of further interest are:

- An investigation into a DECOR LMS architecture that uses direct calculation of DECOR coefficients rather than transformation would be worthwhile. Any way to address the stability issues encountered would be of benefit.
- Further investigation into the effect of different multiplier types used in the low power architectures presented in this thesis. Characterisation of multiplier type has not been carried out exhaustively for these applications.
- The use of custom memory blocks within the Variable Length LMS adaptive filter implementation could be investigated to extend this work and further reduce power consumption.

- The LMMSE receiver work presented could be extended to incorporate further low power functional blocks that already exist. This would undoubtedly be of benefit to overall power consumption.
- The application of existing methods to dynamically control the number of optimised LMMSE fingers in operation should be explored to achieve further improvements in power consumption.

References

[1] S. Haykin, *Adpative Filter Theory, 4th edn.* Prentice-Hall, 2000.

[2] J. Proakis, *Digital Communications, 3rd edn.* McGraw-Hill, New York, 1994.

[3] H. Holma and A. Toskala, *WCDMA for UMTS: Radio Access for Third Generation Mobile Communications, 1st edn.* John Wiley & Sons, 2001.

[4] A. J. Viterbi, *CDMA: Principles of Spread Spectrum Communication*. Addison-Wesley, 1995.

[5] G. Tiwary, "Below the half-micron mark," in *IEEE Spectrum*, vol. 32, pp. 84–87, Nov 1994.

[6] S. Narendra and A. Chandrakasan, *Leakage in Nanometer CMOS Technologies*. Springer, 2006.

[7] D. Garrett, M. Stan, and A. Dean, "Challenges in clockgating for a low power ASIC methodology," in *IEEE International Symposium on Low Power Electronics and Design*, pp. 176–181, 1999.

[8] A. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. Brodersen, "Optimising power using transformations," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, pp. 12–30, January 1995.

[9] M. Pedram, "Power minimisation in IC design principles and applications," in *ACM transactions on Design Automation of Electronic Systems*, vol. 1, pp. 3–56, January 1996.

[10] A. Chandrakasan and R. Brodersen, *Low Power Digital CMOS Design*. Kluwer, 1995.

[11] A. Raghunathan, S. Dey, and N. Jha, "Glitch analysis and reduction in register transfer level optimisation," in *33rd ACM/IEEE Design Automation Conference*, June 1996.

[12] L. Benini, G. D. Micheli, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "Glitch power minimization by selective gate freezing," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8 no.3, pp. 287–298, June 2000.

[13] H. Lee, H. Shin, and J. Kim, "Glitch elimination by gate freezing, gate sizing and buffer insertion for low power optimization circuit," in *IEEE Industrial Electronics Society Conference, 2004. IECON 2004. 30th Annual*, vol. 3, pp. 2126–2131, 2-6 Nov 2004.

[14] M. Alidina, J. Monteiro, S. Devdas, A. Ghosh, and M. Papaefthymiou, "Precomptuation-based sequential logic optimisation for low power," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 425–436, December 1994.

[15] J. Monteiro, S. Devdas, and A. Ghosh, "Sequential logic optimisation for low power using input disabling precomputation architectures," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 3, pp. 279–284, March 1998.

[16] S. Wuytack, F. Catthoor, F. Franssen, L. Nachtergale, and H. Deman, "Global communication and memory optimising transformation for low power design," in *International Workshop on Low Power Design*, pp. 178–187, April 1994.

[17] C. Su, C. Tsui, and A. Despain, "Low power architecture design and compilation techniques for high performance processors," in *COMPCON Digest of papers*, pp. 489–498, February 1994.

[18] L. Benini and G. Micheli, "State assignment for low power dissipation," in *IEEE Journal of Solid State Circuits*, vol. 30, pp. 258–267, March 1995.

[19] E. Musoll and J. Cortadella, "Scheduling and resource binding for low power," in *International Symposium on Systems Synthesis*, pp. 104–109, 1995.

[20] T. Callaway and E. Swartzlander, "Optimising adders for WSI," in *IEEE International Conference on Wafer Scale Integration*, pp. 251–260, 1992.

[21] T. Callaway and E. Swartzlander, "Optimising multipliers for WSI," in *IEEE International Conference on Wafer Scale Integration*, pp. 85–94, 1993.

[22] G. Keane, J. Spanier, and R. Woods, "The impact of data characteristics and hardware topology on hardware selection for low power DSP," in *International Symposium on Low Power Electronics and Design*, pp. 94–96, August 1998.

[23] C. Tsui, M. Pedram, and A. Despain, "Power efficient technology decomposition and mapping under an extended power consumption model," in *IEEE transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 9, pp. 1110–1122.

[24] C. Yeh, C.-C. Chang, , and J.-S. Wang, "Technology mapping for low power," in *Design Automation conference*, vol. 1, pp. 145–148, January 1999.

[25] P. Merakos, K. Masselos, O. Koufopavlou, S. Nikolaidis, and C. Goutis, "A novel transformation for reduction of switching activity in FIR filter implementation," in *International Conference on Digital Signal Processing*, vol. 2, pp. 653–656, July 1997.

[26] A. Erdogan and T. Arslan, "An order based segmentation algorithm for low power implementation of digital filters," in *IEEE Int. Conference on Acoustics, Speech and Siganl Processing (ICASSP2000)*, pp. D441 – D444, June 2000.

[27] A. Erdogan and T. Arslan, "A coefficient segmentation algorithm for low power implementation of FIR filters," in *IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 359–362, June 1999.

[28] A. Erdogan and T. Arslan, "Data block processing for low power implementation of direct form FIR filters on single multiplier CMOS based DSPs," in *IEEE International Symposium on Circuits and Systems*, pp. D441–D444, June 1998.

[29] J. Ludwig, S. Nawab, and A. Chandrakasan, "Low power digital filtering using approximate processing," in *IEEE Journal of Solid State Circuits*, vol. 31, pp. 395–399, March 1996.

[30] M. Mehendale, S. Sherlekar, and G. Venkatesh, "Low power realisation of FIR filters using multirate architectures," in *9th Internaltional Conference on VLSI Design*, pp. 370–375, January 1996.

[31] M. Mehendale, S. Sherlekar, and G. Venkatesh, "Coefficient optimisation for low power optimisation of FIR filters," in *IEEE Workshop on VLSI signal processing*, pp. 352–361, 1995.

[32] N. Sankarayya, K. Roy, and D. Bhattacharya, "Algorithms for low power and high speed FIR filter realisation using differential coefficients," in *IEEE Transactions on Circuits and Systems-II*, vol. 44, pp. 488–497, June 1997.

[33] Z. Yu, M. L. Yu, K. Azadet, and A. W. Jr., "A low power FIR filter design technique using dynamic reduced signal representation," in *International Symposium on VLSI Technology, Systems and Applications*, pp. 113–116, 2001.

[34] J. Park, W. Jeong, H. Choo, H. Mahmoodi-Meimand, Y. Wang, and K. Roy, "High performance and low power FIR filter design based on sharing multiplication," in *International Symposium on Low Power Electronics and Design*, pp. 295–300, August 2002.

[35] A. Erdogan, M. Hasan, and T. Arslan, "Algorithmic low power FIR cores," in *IEE Proceedings on Circuits, Devices and Systems*, vol. 150, pp. 155–160, June 2003.

[36] C. Nicol and P. Larsson, "Low power multiplication for FIR filters," in *International Symposium on Low Power Electronics and Design Proceedings*, pp. 76–79, Aug 1997.

[37] F. Riera-Palou, J. Noras, and D. Cruickshank, "Variable length equalisers for broadband mobile systems," in *IEEE 52nd Vehicular Technology Conference*, pp. 2478–2485, September 2000.

[38] F. Riera-Palou, J. Noras, and D. Cruickshank, "Linear equalisers with dynamic and automatic length selection," in *Electronics Letters*, vol. 37, pp. 1553–1554, December 2001.

[39] Z. Pritzker and A. Feuer, "Variable length stochastic gradient algorithm," in *IEEE Transactions on Signal Processing*, pp. 997–1001, April 1991.

[40] Y. Won, R.-H. Park, J. Park, and B.-U. Lee, "Variable LMS algorithms using the time constant concept," in *IEEE Transactions on Consumer Electronics*, pp. 655–661, Aug 1994.

[41] R. Price and P. Green, "A communication technique for multi-path channels," in *Proceedings of the IRE*, vol. 46, pp. 555–570, 1958.

[42] A. Duel-Hallen, J. Holtzman, and Z. Zvonar, "Multiuser detection for CDMA systems," in *IEEE/ACM Personal Communications*, vol. 2, pp. 46–58, 1995.

[43] M. Simon, J. Omura, R. Scholtz, and B. Levitt, *Spread Spectrum Communications Handbook*. McGraw-Hill, New York, 1994.

[44] A. Duel-Hallen, J. Holtzman, and Z. Zvonar, "Multiuser detection for CDMA systems," in *IEEE Personal Communications*, vol. 2, pp. 46–58, April 1995.

[45] M. Juntti and S. Glisic, *Advanced CDMA for Wireless Communications*, vol. Wireless Communications: TDMA VERSUS CDMA, pp. 447–490. Kluwer, 4 ed., 1997.

[46] S. Moshavi, "Multi-user detection for DS-CDMA communications," in *IEEE Communications Magazine*, vol. 34, pp. 124–137, 1996.

[47] C.-W. Ku, F.-Y. Kuo, C.-K. Chen, and L.-G. Chen, "Low power strategy about correlator array for CDMA baseband processor," in *IEEE Workshop on Signal Processing Systems,*, pp. 513–522, October 1999.

[48] M. Latva-aho, *Advanced Receivers for Wideband CDMA Systems*. PhD thesis, University of Oulu, October 1998.

[49] M. Majmundar, N. Sandhu, and J. Reed, "Adaptive single-user receivers for direct-sequence spread-spectrum CDMA systems," in *IEEE Transactions on Vehicular Technology*, vol. 49, March 2000.

[50] H.-J. Lee and D. S. Ha, "An area and power efficient RAKE receiver architecture for DSSS systems," in *IEEE International SOC [Systems-on-Chip] Conference Proceedings*, pp. 103–106, September 2003.

[51] A. Bianco, A. Dassatti, M. Martina, A. Molino, and F. Vacca, "A reconfigurable, power-scalable rake receiver IP for W-CDMA," in *Proceedings of the ASP-DAC 2003 Design Automation Conference, Asia and South Pacific*, pp. 499–502, Jan 2003.

[52] L. T. Smit, G. J. Smit, P. J. Havinga, Johann, L. Hurink, and H. Broersma, "Influences of rake receiver/turbo decoder parameters on energy consumption and quality," in *In Proc. of 2002 International Conference On Third Generation Wireless and Beyond*, pp. 227–235, May 2002.

[53] M. Latva-aho, M. Juntti, and I. Oppermann, "Reconfigurable adaptive RAKE receiver for wideband CDMA systems," in *Vehicular Technology Conference, 1998. VTC 98. 48th IEEE*, vol. 3, pp. 1740–1744, May 1998.

[54] F.-Y. Kuo and C.-W. Ku, "Software radio based re-configurable correlator/FIR FILTER for CDMA/TDMA receiver," in *IEEE International Symposium on Circuits and Systems Proceedings. ISCAS 2000 Geneva*, vol. 1, pp. 28–31, 112-115 2000.

[55] C.-K. Chen, P.-C. Tseng, Y.-C. Chang, and L.-G. Chen, "A digital signal processor with programmable correlator array architecture for third generation wireless communication system," in *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, pp. 1110–1120, December 2001.

[56] H.-J. Lee and D. S. Ha, "A new low-power and area efficient RAKE receiver design without incurring performance degradation," in *15th Annual IEEE International ASIC/SOC Conference*, pp. 251–255, September 2002.

[57] S. Lee and J. Kim, "Vlsi architecture design of rake receivers for CDMA2000 systems," in *IEEE Workshop on Signal Processing Systems (SIPS '02)*, pp. 183–188, October 2002.

[58] H. Choi and W. Burleson, "Search based wordlength optimisation for VLSI/DSP synthesis," in *VLSI Signal Processing*, vol. 7, pp. 198–207, 1994.

[59] H. Choi, *Analysing and Exploiting Wordlength Effects in VLSI/DSP Synthesis*. PhD thesis, Dept.of Electrical and Computer Engineering, University of Massachusetts, Amherst, 1994.

[60] M. Mehendale, S. D. Sherlekar, and G. Venkatesh, "Low power realization of FIR filters on programmable DSP's," in *IEEE Trans. on VLSI Systems*, vol. 6, no. 4, pp. 546–553, December 1998.

[61] A. Erdogan, M. Hasan, and T. Arslan, "Algorithmic low power FIR cores," in *IEEE Proceedings - Circuits, Devices and Systems*, vol. 150, pp. 155–160, June 2003.

[62] A. Erdogan and T. Arslan, "Low power implementation of high throughtput FIR filters," in *IEEE Int. Conf. on Circuits and Systems*, pp. 373–376, May 2002.

[63] D. Parker and K. Parhi, "Area efficient parallel FIR digital filter implementations," in *Application Specific Systems, Architechtures and Processors*, pp. 93–111, August 1996.

[64] T.-S. Chang, Y.-H. Chu, and C.-W. Jen, "Low power FIR filter realization with differential coefficients and inputs," in *IEEE Trans. on circuits and Systems-II: Analog and Digital Signal Processing*, vol. 47, no. 2, pp. 137–145, February 2000.

[65] S. Ramprasad, N. Shanbhag, and I. Hajj, "Decorrelating (DECOR) transformations for low power digital filters," in *IEEE Trans. on circuits and Systems-II: Analog and Digital Signal Processing*, vol. 46, no. 6, pp. 776–788, June 1999.

[66] R. Lai, "Design of low power FIR filter by using differential coefficients algorithm," Master's thesis, September 2003.

[67] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*. Wiley, 1996.

[68] S. Haykin and B. Widrow, *Least-Mean-Square Adaptive Filters*. Wiley, 2003.

[69] R. Bilcu, P. Kuosmanen, and K. Egiazarian, "A new variable length LMS algorithm: Theoretical analysis and implementations," in *Proc. 9th Int. Conf. Electronics, Circuits and Systems*, vol. 3, pp. 1031–1034, 2002.

[70] Y. Gu, K. Tang, H. Cui, and W. Du, "LMS algorithm with gradient descent filter length," in *IEEE Signal Processing Letters*, vol. 11, pp. 305–307, March 2004.

[71] P. S. Lewis, "RLS adaptive filtering without a desired signal: Algorithms and architectures," in *Proc. 22nd Asilomar Conference on Signals, Systems and Computers*, vol. 1, pp. 49–53, Oct 31 - Nov 2 1998.

[72] S. Werner and J. Lilleberg, "Downlink channel decorrelation in CDMA systems with long codes," in *Proc. IEEE Vehicular Technology Conference, Houston, USA*, vol. 2, pp. 1614–1617, July 16-19 1999.

[73] K. Hooli, M. Latva-aho, and M. Juntti, "Performance evaluation of adaptive chip-level channel equalisers in WCDMA downlink," in *Proc. IEEE International Conference on Communications, Helsinki, Finland*, vol. 6, pp. 1974–1979, June 11-15 2001.

[74] M. Latva-aho and M. Juntti, "Modified LMMSE receiver for DS-CDMA - part I: Performance analysis and adaptive implementations," in *Spread Spectrum Techniques and Applications, 1998 IEEE 5th International Symposium on*, vol. 2, pp. 652–657, Sept 1998.

[75] P. Rapajic and B. Vucetic, "Adaptive receiver structures for asynchronous CDMA systems," in *IEEE Journal on Selected Areas in Communications*, vol. 12, pp. 21–27, May 1994.

[76] H. Huang, *Combined multipath processing, array processing and multiuser detection for DS-CDMA channels*. PhD thesis, Princeton University, Princeton, NJ, USA, 1996.

[77] S. Goto, T. Yamada, N. Takayama, and H. Yasuura, "A low power digital matched filter for spread-spectrum systems," in *ISPLED'02 Conference*, pp. 301–306, August 12-14 2002.

[78] D. Garrett and M. Stan, "Power reduction techniques for a spread specrtum-based correlator," in *Proc. ISLPED'97*, pp. 85–88.

[79] S.-H. Yen and C.-K. Wang, "A 2V CMOS programmable pipelined digital differential matched filter for DS-CDMA system," in *Proc. the 1st IEEE Asia-Pacific Conference on ASIC*, August 1999.

[80] K. Kitamura, K. Taki, T. Ogata, and Y. Murata, "Low power consumption CMOS digital matched filter," in *Journal on IPSJ*, vol. 42, pp. 1016–1022, April 2001.

[81] M. Liou and T. Chiueh, "A low-power digital matched filter for direct-sequence spread-spectrum signal acquisition," in *IEEE Journal on Solid State Circuits*, vol. 36, pp. 933–943, June 2001.

Appendix A
Publications

A.1 Refereed Conferences

1. M.P. Tennant, A.T. Erdogan, T. Arslan and J. Thompson, *"A Novel Architecture Using the Decorrelating Transform for Low Power Adaptive Filters,"* 19th International Conference on VLSI Design 2006. Held jointly with 5th International Conference on Embedded Systems and Design, Hyderabad, India. 3-7 January 2006.

2. M.P. Tennant, A.T. Erdogan, T. Arslan and J. Thompson, *"A Novel Equaliser Architecture with Dynamic Length Optimisation,"* IEEE Symposium on Circuits and Systems 2006, Kos, Greece. 21-24 May 2006.

3. M.P. Tennant, A.T. Erdogan, T. Arslan and J. Thompson, *"A New LMMSE Receiver Architecture with Dynamic Filter Length Optimisation"* International Symposium on System-on-Chip 2007, Tampere, Finland. 19-21 November 2007.

A Novel Architecture Using the Decorrelating Transform for Low Power Adaptive Filters

Mark P. Tennant, Ahmet T. Erdogan, Tughrul Arslan and John Thompson
School of Engineering and Electronics, The University of Edinburgh
The King's Buildings, Edinburgh EH9 3JL
United Kingdom
M.P.Tennant@sms.ed.ac.uk

Abstract

This paper presents a novel architecture using the decorrelating (DECOR) transformation technique when applied to an LMS adaptive filter. The DECOR transform has been evaluated previously, however no practical evaluation has previously ever been made of how area and power performance is affected by the addition of the DECOR transform to an LMS adaptive filter. Neither has analysis been carried out to determine any tradeoff in increased area this might incur against power saved. This paper presents the first complete architectural VLSI implementation of the decorrelating transform when applied to an adaptive filter and includes a performance study in terms of area and power.

1. Introduction

The high demand for low power, high performance electronic components has been prompted by the popularity of portable, battery-powered end-user devices. This is especially so for wireless communication devices such as mobile phones, PDAs and wireless enabled laptops that require high-speed computation and real-time signal processing capabilities. The implementation of ever more complex systems which facilitate wireless communication requires the use of efficient and flexible cores in their design. In turn these cores often involve the repetitive implementation of FIR filters and/or adaptive filters which include an FIR core, that must have heavily constrained power and area requirements. In the implementation of FIR filters and thus the adaptive filter, there are two approaches, sequential and parallel. The parallel implementation can maximise throughput at the cost of considerable additional hardware such as adders and multipliers. On the other hand, the sequential implementation is cost and area-effective in hardware although however does suffer a bottleneck in throughput.

Power optimisation has become a crucial part of FIR filter design with an ever increasing number of published techniques to reduce the power consumption of FIR filters. The authors in [1] optimise word-lengths of the input and output data samples and coefficient values. This involves the use of a general search based methodology which is based on statistical precision analysis and the incorporation of cost/performance/power measures into an objective function through word-length parameterisation. In [2], Mehendale et al. present an algorithm for optimising the coefficients of an FIR filter to reduce the power consumption in its implementation on a programmable DSP. The use of coefficient segmentation, block processing and combined segmentation and block processing algorithms for low power FIR filter implementations have been shown in [3]. High throughput FIR implementations have also been described by the authors in [4] and [5].

In most implementations of FIR filters the filter coefficients are used directly to compute the convolution with the input data. The differential coefficient method (DCM) inroduced in [6] uses various orders of differences between coefficients along with stored intermediate results rather than the coefficients themselves in the computation of the convolution. If fewer bits are required to represent the differences compared to the actual coefficients, the size of the arithmetic unit in the filter can be reduced, hence reducing power consumption. This method does however have an overhead of $N-1$ additional latches (for storage of intermediate results) and $N-1$ additional adders (for addition of intermediate results) for an N tap filter. Although, greater orders of differences have smaller magnitudes, the overhead required by the DCM increases as the order of differences is increased. There is therefore a

point beyond which the gains due to smaller magnitudes are less than the overall cost of overheads [6]. To minimise the overhead while retaining the benefit of DCM, differential coefficient and input method (DCIM) [7] and decorrelating (DECOR) transforms [8] have been proposed. Some of the advantages of DECOR over DCM are listed as (a) lower overheads for a given filter order, (b) overheads being independent of the filter order and (c) power savings over a wider range of filter bandwidths. It can also be seen in [8] that the DECOR transform is proposed for use in adaptive filtering.

However, the performance of the above methods in terms of power and area, was evaluated analytically using high level models of multipliers, adders and memory units. In [8], in addition to the analytical results, simulation based results are also presented. It must be remembered though that these results were also not based on real VLSI implementation of the filters and no analysis whatsoever is provided for adaptive filters. C simulation models are used, assuming zero and/or unit delays for circuit gates. The work carried out by the authours of [9] presented the VLSI implemenation of the DECOR transform for low power filtering cores and analysed the power, area and speed performance. For more realistic evaluation in this work an adaptive LMS filter using an DECOR FIR core has been implemented in VLSI, targeting 0.18 micron standard CMOS technology. The power and area analysis was made based on this novel architecture being presented consisting of 1st order DECOR in an adaptive LMS filter for different coefficient word-lengths.

2. Implementation

An N-tap FIR filter performs the following convolution:

$$y(n) = \sum_{k=0}^{N-1} b_k x(n-k) \tag{1}$$

where b_k's are the coefficients of the filter, $x(n)$ and $y(n)$ are the nth terms of the input and output sequences, respectively. The z-transfer of (1) is given below:

$$Y(z) = H(z)X(z) \tag{2}$$

where $Y(z)$, $H(z)$ and $X(z)$ are the z-transforms of the output, filter and input respectively. In DECOR, the transfer function $H(z)$ is multiplied and divided by the polynomial:

$$T(z) = (1 + \alpha.z^{-\beta})^m \tag{3}$$

where m represents the order of coefficient difference, α and β are parameters chosen depending on the type of FIR filter. The frequency response is not altered by multiplying and dividing the transfer function $H(z)$ by this polynomial. For example, the z-transfer of the first order low-pass FIR filter is given by (α = -1, β = 1, m = 1):

$$Y(z) = \frac{[(1 - z^{-1})H(z)]}{1 - z^{-1}} X(z) \tag{4}$$

$$Y(z) - Y(z)z^{-1} = H(z)X(z) - H(z)X(z)z^{-1} \tag{5}$$

According to (1) and (5) the transformed filter can be expressed as:

$$y(n) - y(n-1) = \sum_{k=0}^{N-1} b_i x(n-k) - \sum_{k=0}^{N-1} b_i x(n-k-1) \tag{6}$$

Re-arranging (6) we can obtain the following equation for first order (m=1) differential coefficients:

$$y(n) = b_0 x(n) + \sum_{k=1}^{N-1} (b_k - b_{k-1}) x(n-k) - b_{N-1} x(n-N) + y(n-1) \tag{7}$$

Clearly as (7) shows, for first order differential coefficients, the filter outputs can be obtained using the differences between adjacent coefficients (except for the first and last coefficients) and the previous filter output. Also note that the transformed filter requires an additional multiplication and subtraction operation to realise the term $(-b_{N-1}x(n-N))$ in (7). Therefore, this together with adding the previous filter output represents the overhead for DECOR using first order differential coefficients.

The weight update equation for a least-mean squares (LMS) filter is

$$b_i(n+1) = b_i(n) + \mu e(n) x(n-i) \tag{8}$$

where μ is the step size and $e(n)$ is the adaptation error given by

$$e(n) = d(n) - y(n) \tag{9}$$

where $d(n)$ is the desired output of the filter.

Applying the DECOR tranform to an adaptive filter involves the derivation of the following from (1):

$$y(n) = -\alpha y(n-\beta) + \sum_{i=0}^{N+\beta-1} \delta_i(n) x(n-i) \tag{10}$$

where δ_i is the DECOR filter coefficient. The derivation of which is provided in [8].

2.1 Conventional Adaptive Filter Core

In order to evaluate the performance of the DECOR adaptive filter a conventional adaptive filter core based on equation (1) and using the LMS algorithm with it's equations (8) and (9), was implemented. This conventional adaptive filter has two functional blocks, as seen in Fig. 1(a). A weight update block (WUD), which uses the data input samples and an error signal to calculate new coefficients, and a filter block, which is generally an FIR filter and employs the coefficients calculated by the WUD block.

Within the WUD block there are five functional blocks as shown in Fig. 2. It consists of three memory blocks for storing the input data (X RAM), the calculated filter coefficients (LMS RAM) and the filter weights from the previous $(n-1)$ sample (FIR W RAM), an arithmetic unit (AU) and a control block (CONTROL). A brief description of these blocks is given below:

- CONTROL: The controller is based on a counter and is responsible for the synchronisation of activity for every block in the WUD unit.
- X RAM: This is a RAM used for the storage of the input data $x(n)$. The input data sequence is clocked into this RAM for the use of the AU when calculating filter weights.
- LMS RAM: Upon calcuation of filter weights the values are stored in this RAM for the use of the FIR filter.
- FIR W RAM: When clocking filter weights out of the LMS RAM the values are immediately copied into this FIR W RAM for subsequent calculation of further filter weights for the next $(n+1)$ sample.
- AU: This arithmetic unit consists of two multipliers and an adder. This arithmetic unit performs the direct calculation of filter weights according to the weight update equation (8). The calculated values are latched into LMS RAM according to the signals created by CONTROL.

A conventional FIR filter is used, details of the implementation of which can be found in [9]. The conventional FIR filter described here and the implementation of the conventional filter block in [9] are identical with the exception of the removal of the ROM for storing the fixed coefficients in the FIR filter, given that the coefficients are calculated by the WUD block.

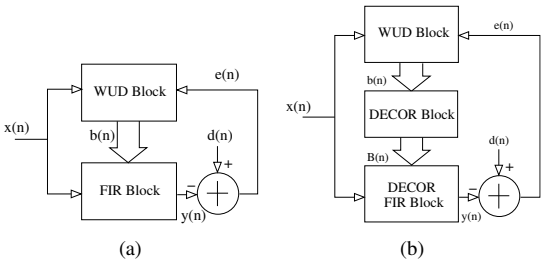


Figure 1. a) Conventional and b) DECOR Adaptive Filters.

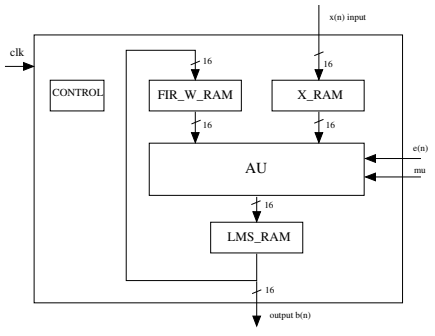


Figure 2. Block Diagram of Conventional WUD Block.

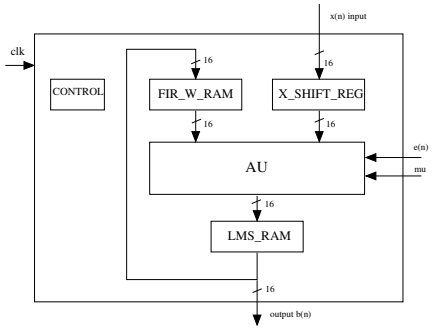


Figure 3. Block Diagram of DECOR WUD Block.

Table 1. Power Consumption analysis for different Coefficient word-lengths

Algorithm	Dynamic Power (mW)	% Reduced
CON 16bit	17.13	-
DECOR 16bit	16.25	5.1
DECOR 14bit	15.62	8.8
DECOR 12bit	14.65	14.5
DECOR 10bit	13.97	18.4

2.2 DECOR Adaptive Filter Core

The DECOR adaptive filter consists of three functional blocks, seen in Fig. 1(b). A WUD block, based on an optimised implementation of the conventional WUD block, a DECOR block for calculating the DECOR filter weights δ_i from the weights produced by the WUD block and a DECOR FIR filter. The DECOR FIR filter used here is again identical to the implementation found in [9] with the exception of the removal of the ROM for storing DECOR filter coefficients given that DECOR filter coefficients will be calculated by the WUD block via the DECOR block. A functional diagram of the DECOR WUD block can be seen in Fig. 3. The DECOR block represents part of the overhead in this implementation, the other part is contained in the MAC block of the DECOR FIR filter due to the extra multiplications and additions required by DECOR. The DECOR block used to calculate the δ_i filter coefficients, contains a number of registers for storing the previous δ_{i-1} filter coefficients and also a subtractor block.

There is also a difference in the MAC block of the DECOR FIR filter in that the reduction in δ_i coefficient word-length results in a reduction in the port size of the mulptiplier therein. The multiplier size is therefore 16 x X where X represents the word-length of the coefficients output by the DECOR block and the data word-length remains equal to 16 bits. This reduction in coefficient word-length results in a smaller multiplier compared to a conventional FIR filter.

3. Results

Two different adaptive filter cores have therefore been implemented, the first being the conventional (CON) and secondly the 1st order DECOR. Both have been analysed in terms of area usage and power consumption. DECOR has been implemented for 16bit, 14bit, 12bit and 10bit coefficient wordlengths in order to study the impact of the reduction in size of the multiplier on area and power. The cores were designed using Verilog HDL and then synthesised using Design Compiler^{TM} targeting the UMC 0.18μ standard cell CMOS library. The requirements of the synthesis were identical for all cores. This was vital in order to allow for consistent delay, power consumption and area usage comparisons. A netlist was created for each core and back-annotated netlist simulations for a uniformly distributed random input data of 1000 samples using Verilog-XL^{TM} simulator were performed and verifed against Matlab^{TM} simulation results. The resulting data, including switching activity of the circuit nets was then used by Synopsys DesignPower^{TM} to determine power consumption for the different adaptive filter cores. In all of the above stages a clock frequency of 100 MHz and a supply voltage of 1.8 Volts were used.

Power results are shown in Table 1 and area results are shown in Table 2. A 73-tap adaptive filter was analysed with step-size μ = 0.001. Performance analysis was carried out using four different coefficient wordlengths. Table 1 shows that for the 1st order DECOR transform a decrease in coefficient wordlenth results in an increase in the power saving achieved. The difference in power consumption between the CON adaptive filter and the DECOR 16 bit filter is due to the optimisation of the WUD block in the DECOR implementaion. The X RAM used in CON can be replaced by a latch based shift register in DECOR seen in Fig. 3, which reduces overall power consumption. The implementation of DECOR creates an extra filter coefficient thus allowing the shift register to be used as the timing of control signals is modified.

A breakdown of the power consumed by different blocks within each apdaptive filter design shows where the overhead in this technique lies and where the

Table 2. Area analysis for different Coefficient word-lengths

Algorithm	Area (μm^2)	% Reduction
CON 16bit	1103487.13	-
DECOR 16bit	951924.25	13.7
DECOR 14bit	948440.00	14.1
DECOR 12bit	944398.69	14.4
DECOR 10bit	941235.69	14.7

Table 3. Power Analysis for FIR MAC Block

Algorithm	Dynamic Power (mW)	% Change
CON 16bit	5.185	-
DECOR 16bit	7.07	+36.35
DECOR 14bit	6.572	+26.75
DECOR 12bit	5.749	+10.88
DECOR 10bit	5.145	-0.77

greatest power savings are made. The graph in Fig. 4 shows the power consumed by the main functional blocks within the adaptive filter.

It can be seen that the overhead introduced by the DECOR block in terms of power consumed, is very low in relation to other blocks and changes very little as the wordlength in the DECOR designs reduces. A small reduction in the power consumed by the DECOR block in each implementation is evident due to the reduction in coefficient wordlength it calculates. A major benefit of this architecture is that the logic required by the DECOR block is very simple and therefore does not present much of a penalty in terms of power or area. A reduction in power consumed by the DECOR FIR block can also be seen as the coefficient wordlength reduces. This being the aim of the DECOR transform in that the filter coefficients presented to the MAC within the filter are directly reduced. Table 3 shows the power consumption of the MAC within the CONV filter and within the DECOR filter for each implementation.

These results clearly show the considerable overhead present in the implementation of the DECOR FIR filter. A wordlength reduction of 6 bits is needed before the 1st order DECOR filter will demonstrate any power saving in the MAC block. This overhead is

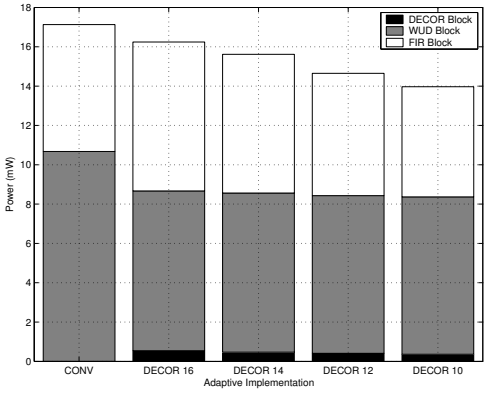


Figure 4. Power Consumption for Main Functional Blocks

due to the extra multiplications and additions needed in the DECOR algorithm. In practice, implementation of a DECOR filter without a reduction in the coefficient wordlength would never be worthwhile given that the very purpose of the DECOR transform is to reduce this parameter. It has simply been done to point out the overhead present in the design. Nonetheless, it can be seen that due to optimisation in the WUD block an overall power saving can be made for the adaptive filter as a whole. This is due to the optimisation of the x input data storage in the WUD block. The CONV implementation of the adaptive filter must use a RAM for this purpose whereas in the DECOR design a latch based shift register can be used as timing restraints are less strict. Essentially the extra filter weight created by the DECOR transform allows the control timing of the DECOR WUD block to tolerate propagation delay through the shift register.

4. Conclusions

This paper has presented a novel architectural VLSI implementation of the DECOR transformation technique for low power adaptive filtering cores. The technique was implemented for different 1st order DECOR coefficient word-lengths. The results demonstrate a power saving in the range of 5 - 18 % for coefficient word-lengths varying from 16 bits to 10 bits when compared to a conventional adaptive filter implementation. At the same time, an area saving of up to 15 % is achieved due to optimisation that can be carried out in the design. It has also been shown that the overhead required by the additional circuitry of the DECOR block is minimal for this architecture.

References

[1] H. Choi and W. P. Burleson, *Search Based Wordlength Optimisation for VLSI/DSP Synthesis*, VLSI Signal Processing. vol. 7, 1994, pp. 198-207.

[2] M. Mehendale, S. D. Sherlekar and G. Venkatesh, *Low Power Realization of FIR filters on Programmable DSP's*, IEEE Trans. on VLSI Systems. Vol. 6, No. 4, Dec. 1998, pp. 546-553.

[3] A. T. Erdogan, M. Hasan and T. Arslan, *Algorithmic Low Power FIR cores*, IEEE Proceedings - Circuits, Devices and Systems. Volume. 150, No. 3, Jume 2003 pp. 155-160.

[4] A. T. Erdogan and T. Arslan, *Low Power Implementation of High Throughtput FIR filters*, IEEE Int. Conf. on Circuits and Systems. May 2002, pp. 373-376.

[5] D. A. Parker and K. K. Parhi, *Area Efficient Parallel FIR Digital filter Implementations*, Application Specific Systems, Architechtures and Processors. Aug 1996 pp. 93-111.

[6] N. Sankarayya, K. Roy and D. Bhattacharya, *Algorithms for Low Power and High Speed FIR Filter Realisation Using Differential Coefficients*, IEEE Trans. on circuits and Systems-II: Analog and Digital Signal Processing. June 1997, Vol. 44, pp. 88-497.

[7] T-S. Chang, Y-H. Chu and C-W Jen, *Low Power FIR filter Realization with Differential Coefficients and Inputs*, IEEE Trans. on circuits and Systems-II: Analog and Digital Signal Processing. Feb 2000, Vol. 47, no. 2. pp. 137-145.

[8] S. Ramprasad, N. R. Shanbhag and I. H. Hajj, *Decorrelating (DECOR) Transformations for Low Power Digital Filters*, IEEE Trans. on circuits and Systems-II: Analog and Digital Signal Processing. June 1999, Vol. 46, no. 6. pp. 776-788.

[9] A. T. Erdogan, T. Arslan and R. Lai, *Implementation of the Decorrelating Transformation for Low Power FIR Filters*, IEEE Workshop on Signal Processing Systems, Oct. 2004, pp. 337 - 342.

A NOVEL EQUALISER ARCHITECTURE WITH DYNAMIC LENGTH OPTIMISATION

Mark P. Tennant*†, A.T. Erdogan†, T. Arslan† and J. Thompson†

†School of Engineering and Electronics, The University of Edinburgh
The King's Buildings, Edinburgh EH9 3JL
United Kingdom

*Email: M.P.Tennant@sms.ed.ac.uk

***Abstract*— This paper presents a novel architecture for tap-length optimisation of the linear LMS equaliser. No analysis has previously been carried out to determine any tradeoff that exists in circuit area against power saving achieved. A low-complexity length update algorithm is employed to dynamically adjust and optimise the number of taps in the linear equaliser according to channel conditions. The results show that the chosen algorithm presents minimal overhead and reduces power consumed due to optimisation of the equaliser length. This paper presents the first complete architectural VLSI implementation of the length optimised equaliser and includes a performance study in terms of area and power.**

I. INTRODUCTION

The popularity of portable, battery-powered consumer devices is continuing the high demand for low-power, high performance electronic components. This is especially so in the marketplace for wireless communication devices such as 2.5G/3G mobile phones, wireless enabled laptops and other data enabled devices which require powerful computation and real-time signal processing capabilities. The implementation of complex systems which facilitate wireless communication requires the use of efficient and flexible cores in their design. Attention here is concentrated on the design of adaptive filtering cores that must have heavily constrained power and area requirements for these applications. Linear equalisers are typically implemented using adaptive finite impulse response (FIR) filters [1] with filter coefficients being recursively updated using either the recursive least squares (RLS) or more commonly the least mean squares (LMS) algorithm used in this study. The number of taps in the FIR structure has a critical influence on the performance and computational complexity of the equaliser. An equaliser with too many taps will be computationally inefficient and may introduce a degradation in mean squared error (MSE) performance due to limitations of the LMS algorithm whereas, an equaliser with too few taps will be unlikely to reach its true potential level of distortion mitigation. Coupled with this is the time variant nature of wireless channels which ideally necessitates the ability of the equaliser to alter its number of taps with time.

The technique of varying the length of the LMS filter was first presented by the authors of [2] with an algorithm which proved that a filter with fewer taps will have a faster convergence than that of a filter with a higher number of taps. This variable length stochastic gradient (VLSG) algorithm demonstrates an LMS adaptive filter which can accomplish a change in its length from being initially low, therefore aiding fast convergence, gradually increasing over time to achieve the low steady state MSE performance characteristic of higher order filters. In [3], Won et al. went on to propose another variable length LMS (VL-LMS) algorithm using a time-constant concept whereby several filter lengths are predetermined and filter length is increased to the next predetermined value when conditions are satisfied.

Both the algorithms referred to previously however offer only the ability to *increase* the filter length over time to satisfy the contradictory goals of fast convergence and good steady state performance. As a progression from the previous methods, the authors Riera-Palou et al. of [4] specifically present a linear equaliser using an algorithm that can dynamically and automatically increase or decrease the length of the filter. Using a segmented FIR filter structure and a weight update algorithm the optimum, and in this case minimum required, number of taps are operated. Further to this, [5] presents a method whereby the optimum length of the adaptive filter is determined. In this case the number of filter coefficients of an unknown system are found using the LMS algorithm in a system identification setup. This method uses the MSE output from a number of individual LMS adaptive filters in parallel to determine the number of unknown system coefficients and their values. This does not lend itself well to a circuit implementation which is

constrained in terms of area and power. Finally in [6], the authors present the most recent proposal for the variable length LMS algorithm. This algorithm uses a method whereby the filter length is varied according to the negative gradient direction of the estimation error. The familiar gradient decent method is used to track the optimum filter length with constraints included to avoid unexpected behaviour and guarantee convergence.

However, none of the above methods have been evaluated in terms of the power or area overhead required in comparison to a standard fixed length adaptive filter. Simulation based results are presented to show how varying filter lengths effects convergence times. In [4] results are presented to show equaliser length changing in accordance to input E/N_0. To realistically analyse power and area requirements, in this work a variable length LMS equaliser has been implemented in VLSI, targeting 0.18 micron standard CMOS technology.

II. Implementation

An N-tap FIR filter performs the following convolution:

$$y(n) = \sum_{k=0}^{N-1} b_k x(n-k) \tag{1}$$

where b_k's are the coefficients of the filter, $x(n)$ and $y(n)$ are the nth terms of the input and output sequences, respectively. The weight update equation for a least-mean squares (LMS) filter is

$$b_i(n+1) = b_i(n) + \mu e(n) x(n-i) \tag{2}$$

where μ is the step size and $e(n)$ is the adaptation error given by

$$e(n) = d(n) - y(n) \tag{3}$$

$d(n)$ is the desired output of the filter, i.e. the transmitted signal.

As is outlined in [4], splitting an N tap FIR into K concatenated subfilters of P taps each, such that $N = KP$, produces an estimate $y_s(n)$ (with $0 \leq s < K$) of the transmitted data ($d(n)$). The various equaliser outputs $y_s(n)$, can be used to compute a corresponding error signal $e_s(n)$ according to (3) such that

$$e_s(n) = \big|\, d(n) - y_s(n) \big| \tag{4}$$

The distinct error signals can be squared and averaged to obtain an output MSE measure for each subfilter. The performance criteria used to evaluate the different subfilters is the accumulated squared error (ASE) and is defined as

$$ASE_s(n) = \sum_{i=1}^{n} \big| d(i) - y_s(i) \big|^2 = \sum_{i=1}^{n} e_s(i)^2 \tag{5}$$

The advantage of using the ASE is that the repetitive computation of division, used in the calculation of MSE, is removed. The aim of the length update algorithm is to detect the subfilter at which the ASE becomes insignificantly smaller or even larger than the previous subfilter. The algorithm proposed in [4] to control the number of active subfilters involved in equalisation, assuming the equaliser has L active segments, is outlined as

$$ASE_{L-1}(n) = \sum_{i=1}^{n} \beta^{n-1} \big|\, d(i) - y_{L-1}(i) \big|^2 \tag{6}$$

$$ASE_L(n) = \sum_{i=1}^{n} \beta^{n-1} \big|\, d(i) - y_L(i) \big|^2 \tag{7}$$

$$\text{If } ASE_L(n) \leq \alpha_{up} ASE_{L-1}(n) \\ \Rightarrow +1 \text{ subfilter (P extra taps)} \tag{8}$$

$$\text{If } ASE_L(n) \geq \alpha_{dw} ASE_{L-1}(n) \\ \Rightarrow -1 \text{ subfilter (P fewer taps)} \tag{9}$$

where $0 < \alpha_{up} \leq \alpha_{dw} \leq 1$ and determine the amount of worsening or improvement necessary to force the equaliser to expand or contract. β is a forgetting factor and is ≤ 1. A more detailed derivation of this algorithm can be found in [4].

A. Conventional Adaptive Filter Core

In order to evaluate the performance of the variable length equaliser a conventional adaptive filter core based on equation (1) and using the LMS algorithm with it's equations (2) and (3) was implemented. This conventional adaptive filter has two functional blocks, as seen in Fig. 1(a).

A weight update block (WUD), which uses the data input samples and an error signal to calculate new coefficients, and a filter block, which is generally an FIR filter and employs the coefficients calculated by the WUD block. Here a conventional direct form FIR filter was used.

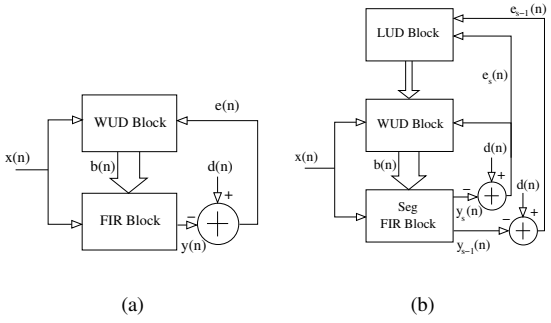


Fig. 1. a) Conventional Adaptive Filter b) Variable Length Adaptive Filter.

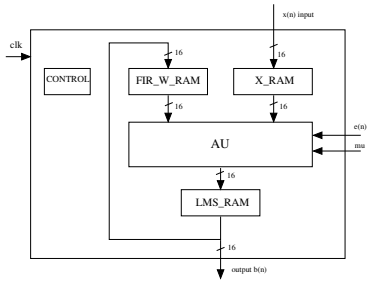


Fig. 2. Block Diagram of Conventional WUD Block.

Within the WUD block there are five functional blocks as shown in Fig. 2. It consists of three memory blocks for storing the input data (X RAM), the calculated filter coefficients (LMS RAM) and the filter weights from the previous $(n-1)$ sample (FIR W RAM), an arithmetic unit (AU) and a control block (CONTROL). A brief description of these blocks is given below:

- CONTROL: The controller is based on a counter which provides addresses for the RAM blocks and is responsible for the synchronisation of activity for every block in the WUD unit.
- X RAM: This is a RAM used for the storage of the input data $x(n)$. The input data sequence is clocked into this RAM for the use of the AU when calculating filter weights.
- LMS RAM: Upon calculation of filter weights the values are stored in this RAM for the use of the FIR filter.
- FIR W RAM: When clocking filter weights out of the LMS RAM (by the FIR filter addresses) the values are immediately copied into this FIR W RAM for subsequent calculation of further filter weights for the next $(n+1)$ sample.
- AU: This arithmetic unit consists of two multipliers and an adder. This arithmetic unit performs the direct calculation of filter weights according to the weight update equation (2). The calculated values are latched into LMS RAM according to the signals created by CONTROL.

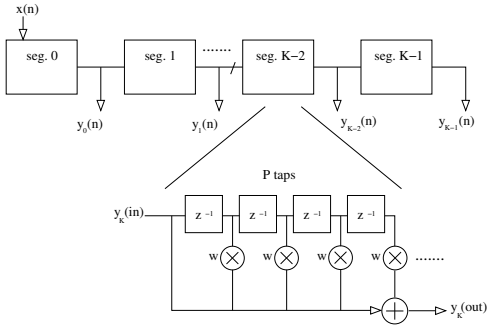


Fig. 3. Block Diagram of Segmanted FIR.

B. Variable Length Adaptive Filter Core

Basing implementation of the design on the algorithm described in [4], the variable length adaptive filter consists of three functional blocks, seen in Fig. 1(b). A WUD block, a length update (LUD) block, used to calculate the ASE values and provide tap-length data to the WUD block, and a segmented FIR filter block. The segmented structure of the FIR filter allows the calculation of subfilter outputs $y_s(n)$ and thus the error signals used to evaluate the error performance of the N tap filter and $(N-P)$ tap filter (P being the number of taps in one subfilter or segment). In this design the segmented FIR filter will calculate an output dependent on the number of coefficients presented to it by the WUD block. It is the output of subfilter results that are critical to the operation of the variable length adaptive filter design. The structure of the segmented FIR filter can be seen in Fig. 3. The control element in the conventional WUD block is replaced by a finite state machine (FSM) design which accepts the tap-length data from the LUD block, controls all timing signals and correctly addresses the various RAM elements in the WUD block. This enhanced control element is the main alteration made to the WUD and enables the length of the adaptive filter to increase or decrease based on the output value of the LUD block.

The LUD block takes the error signals from the last two FIR subfilters, $e_s(n)$ and $e_{s-1}(n)$ and calculates the respective ASE of equations (6) and (7) using accumulator units. The ASE values are latched and presented to a comparator unit in accordance with the length update algorithm of equations (8) and (9). From the comparator result a state machine controller in the LUD block then increments or decrements an up/down counter to provide a tap-length value. This LUD controller initialises the up/down counter and controls timing of the arithmetic units present. The number of taps initialised at reset is selected by the user and the maximum number of taps is set by the

TABLE I
AREA ANALYSIS FOR DIFFERENT ADAPTIVE FILTER CORES

Core	Area (μm^2)	% Difference
CON	451255.0	-
VAR LMS	479232.81	+6.2

TABLE II
POWER CONSUMPTION ANALYSIS FOR DIFFERENT EQUALISER TAP-LENGTHS

Core (No of Taps)	Dynamic Power (mW)	% Difference
CON (64)	21.01	-
VAR LMS (64)	21.89	+4.19
VAR LMS (56)	20.77	-1.14
VAR LMS (48)	19.63	-6.57
VAR LMS (40)	18.48	-12.04
VAR LMS (32)	17.37	-17.33
VAR LMS (24)	16.20	-22.89
VAR LMS (20)	15.62	-25.65
VAR LMS (16)	15.05	-28.36

maximum size of the memory blocks and counters in hardware. The algorithm can vary the tap length of the filter to integer multiples of subfilter size P with the minimum size being P taps.

III. RESULTS

Two different adaptive filter cores have therefore been implemented, the first being the conventional (CON) and secondly the variable length (VAR-LMS). Both have been analysed in terms of area usage and power consumption and in channel equaliser configuration. The cores were designed using Verilog HDL and then synthesised using Design Compiler™ targeting the UMC 0.18μ standard cell CMOS library. The requirements of the synthesis were identical for all cores. This was vital in order to allow for consistent power consumption and area usage comparisons. A netlist was created for each core and back-annotated netlist simulations for a uniformly distributed random input bipolar binary data of 10000 samples using Verilog-XL™ simulator were performed and verified against Matlab™ simulation results. The resulting data, including switching activity of the circuit nets was then used by Synopsys DesignPower™ to determine power consumption for the different adaptive filter cores. In all of the above stages a clock frequency of 100 MHz and a supply voltage of 1.8 Volts were used.

Area results are shown in Table I and power results are shown in Table II. A 64-tap adaptive filter was analysed (CON being fixed and VAR LMS being max possible length) in both cases with fixed step-size μ = 0.005. Power analysis was carried out using an input E/N_0 of 15 dB and fixed channel profile. The channel profile used is identical to that found in [4]. Various equaliser lengths were manually chosen to analyse the effects on power consumption for various tap-lengths. For example, with input E/N_0 of 15 dB, results in [4] show there is no improvement in steady state MSE performance achieved for tap lengths greater than 20. Table II therefore shows that for the VAR LMS core a decrease in tap-length results in an increase in the power saving achieved. These results also identify the overhead present in the VAR LMS architecture. For the VAR LMS core when 64 taps are in operation the power consumed is 0.88 mW higher than in CON when 64 taps are in operation. It can also be seen that the overhead in core area amounts to an increase of 6.2% in VAR LMS over CON. When the length update algorithm is employed to determine the optimum number of taps, power is saved due to the fact that a filter of greater fixed length will not achieve a better steady state MSE. The graph of Fig.4 shows the optimised steady state filter length of the VAR LMS core for various input E/N_0 levels. In this case K = 24 and P = 3 taps/segment.

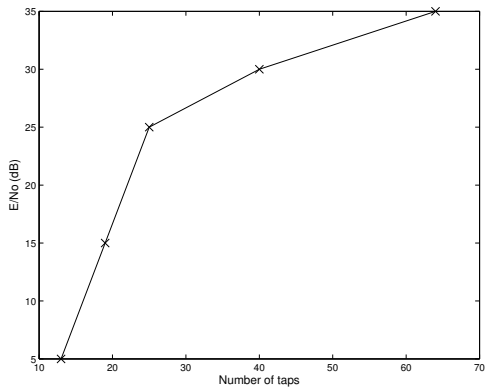


Fig. 4. Number of taps for varying E/N_0

IV. CONCLUSION

This paper has presented a novel architectural VLSI implementation of a dynamically length optimised LMS adaptive filter for use in channel equalisation. The technique was implemented in a 64 tap adaptive filter core and demonstrates length optimisation with varying input E/N_0. The results demonstrate a power saving is achieved by optimising the number of taps in operation. Results have shown a power saving of 28% can be achieved for a variable length architecture optimised to 16 taps over a conventional 64 tap fixed length adaptive filter architecture. It has also been shown that the low-complexity of the additional circuitry needed for the variable length adaptive filter presents minimal overhead for this architecture.

REFERENCES

[1] S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, 2000. 4th edition.

[2] Z. Pritzker and A. Feuer, *Variable length stochastic gradient algorithm*, IEEE Trans. Signal Processing. Vol. 39, pp. 997-1001, Apr. 1991.

[3] Y.K. Won, R.H. Park, J.H. Park and B.-U. Lee, *Variable LMS Algorithms using the time constant concept*, IEEE Trans. Consumer Electron. Vol. 40, pp. 655-661, Aug. 1994.

[4] F. Riera-Palou, J.M. Noras and D.G.M. Cruickshank, *Linear equalizers with dynamic and automatic length selection*, Electron. Lett. Vol. 37, no. 25 pp. 1553-1554, Dec. 2001.

[5] R.C. Bilcu, P. Kuosmanen and K. Egiazarian, *A new variable length LMS algorithm: Theoretical analysis and implementations*, Proc. 9th Int. Conf. Electronics, Circuits and Systems. Vol. 3, pp. 1031-1034, 2002.

[6] Y. Gu, K. Tang, H. Cui and W. Du, *LMS algorithm with gradient descent filter length*, IEEE Signal Processing Lett. Vol. 11, no. 3, pp. 305-307, March 2004.

A New LMMSE Receiver Architecture With Dynamic Filter Length Optimisation

Mark P. Tennant*, A.T. Erdogan, T. Arslan and J. Thompson

School of Engineering and Electronics, The University of Edinburgh
The King's Buildings, Edinburgh EH9 3JL
United Kingdom
*Email: m.p.tennant@sms.ed.ac.uk

***Abstract*— This paper presents a novel architecture for tap-length optimisation of the linear LMS adaptive filter within an LMMSE receiver architecture. No investigation has previously been carried out to determine the suitablility of this concept or the power saving that can be achieved. A low-complexity length update algorithm is employed to dynamically adjust and optimise the number of taps in the adaptive filter present within the LMMSE receiver according to channel conditions. The results show that the chosen algorithm presents minimal overhead and reduces power consumed due to optimisation of the filter length. This paper presents the first architectural VLSI implementation of the LMMSE receiver using the length optimised adaptive filter and includes a performance study in terms of area and power.**

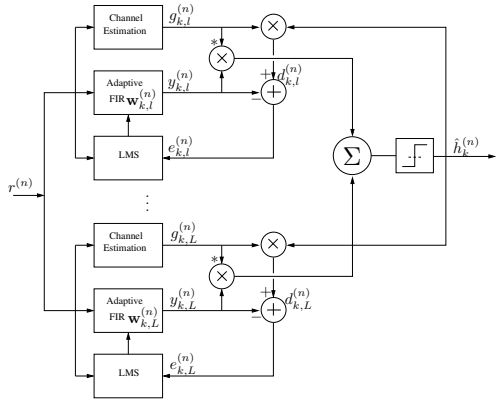


Fig. 1. Conventional adaptive LMMSE receiver functional blocks

I. Introduction

Linear minimum mean square error (LMMSE) techniques can be used to obtain near-far resistant receivers in DS-CDMA systems and overcome the limitations of conventional RAKE receivers. Such receivers have been proposed for application in wideband-CDMA (W-CDMA) systems. With optimal multi-user receivers proving too complex to realise practically, several sub-optimal multi-user receivers have been proposed [1] [2] [3]. Among the group of sub-optimal receivers, the adaptive LMMSE has been proposed for DS-CDMA systems [4] [5]. The LMMSE receiver will minimise the mean square error between the filter output and the true transmitted data symbols. The coefficients of the LMMSE receiver are dependent on the channel coefficients of all users and must be adapted dynamically as the channel changes. In a rapidly fading channel the LMMSE receiver must be adapted continuously and will suffer convergence problems if the channel fades too fast. However, the LMMSE receiver can still be used if the rate of fading is sufficiently low in relation to the data rate. The authors of [6] and [7] present a modified LMMSE receiver structure that employs an adaptive-LMMSE technique to improve the performance of a conventional RAKE receiver. This modified LMMSE receiver assumes that the channel coefficients of the desired user are estimated as is the case in the conventional coherent RAKE receiver.

In this paper the adaptive precombining LMMSE receiver is implemented using the variable length adaptive filter core developed in [8]. This work presents an architectural VSLI implementation of an adaptive LMMSE receiver core, targeting 0.18 micron standard CMOS technology.

II. Implementation

The precombining LMMSE receiver performs multi-user detection before the combining stage which operates then at the symbol level. The adaptive case of the precombining LMMSE receiver requires knowledge of the spreading sequence, data bits and channel coefficients of each multi-path component for which estimates are needed.

The following analysis is carried out for the adaptive LMS version of the precombining LMMSE receiver and is based on the structure of the conventional RAKE receiver. In this adaptive LMMSE receiver each receiver finger is adapted independently to suppress multiple access interference.

A block diagram of the adaptive LMMSE-RAKE structure is shown in Fig.1. The output of the lth receiver branch can be expressed as

$$y_{k,l}^{(n)} = \mathbf{w}_{k,l}^{H_{(n)}} \mathbf{r}^{(n)} \tag{1}$$

and the decisions are made in the adaptive LMMSE receiver according to

$$\hat{h}_k^{(n)} = \text{sgn}\left(\sum_{l=1}^{L} g_{k,l}^{*_{(n)}} y_{k,l}^{(n)} \right) \tag{2}$$

where $\hat{h}_k^{(n)}$ represents the estimate of the original transmitted data symbol, sgn$(\cdot)$ represents the signum function and $g_{k,l}^{*_{(n)}}$ represents complex conjugate of the estimated channel coefficient.

The LMMSE filter coefficients are found by satisfying the MSE criterion (E$[|e_{k,l}^{(n)}|^2]$), which requires the solution of the Weiner-Hopf equation to find the optimal filter coefficients. The filter weights are updated iteratively using the update equation

$$\mathbf{w}_{k,l}^{(n+1)} = \mathbf{w}_{k,l}^{(n)} - \mu \nabla_{k,l} \tag{3}$$

where μ is the update step-size and $\nabla_{k,l}$ is the gradient of the MSE with respect to the filter coefficients. Using the stochastic approximation of the steepest decent algorithm such that

$$\nabla_{k,l} \approx -2\mathbf{r}(g_{k,l}\hat{h}_k)^* + 2\mathbf{r}y_{k,l}^* \tag{4}$$

it can be shown[7] that the LMS algorithm applied here takes the form

$$\begin{aligned} \mathbf{w}_{k,l}^{(n+1)} &= \mathbf{w}_{k,l}^{(n)} + 2\mu\mathbf{r}^{(n)} \left(g_{k,l}^{(n)} \hat{h}_k^{(n)} - y_{k,l}^{(n)} \right)^* \\ &= \mathbf{w}_{k,l}^{(n)} + 2\mu\mathbf{r}^{(n)} e_{k,l}^{*_{(n)}} \end{aligned} \tag{5}$$

The error signal

$$e_{k,l}^{(n)} = d_{k,l}^{(n)} - y_{k,l}^{(n)} \tag{6}$$

is used to update the filter weights using the LMS algorithm.

At this point the receiver vector can be split into adaptive and fixed component parts such that

$$\mathbf{w}_{k,l}^{(n)} = s_{k,l} + \mathbf{x}_{k,l}^{(n)} \tag{7}$$

where $\mathbf{x}_{k,l}^{(n)}$ is the adaptive filter weight vector and $s_{k,l}$ is the fixed spreading sequence for the kth users lth path.

The functionality of the conventional RAKE receiver can now be included for optimisation in the implementation of the adaptive LMMSE receiver finger with the updates of the adaptive component now expressed as

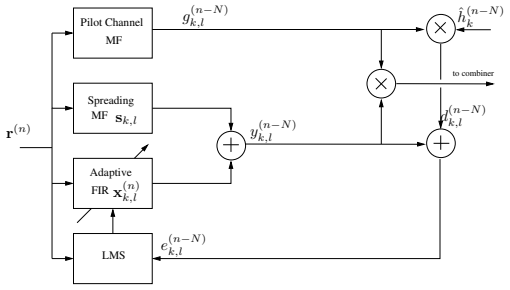


Fig. 2. Block diagram of single adaptive LMMSE receiver finger

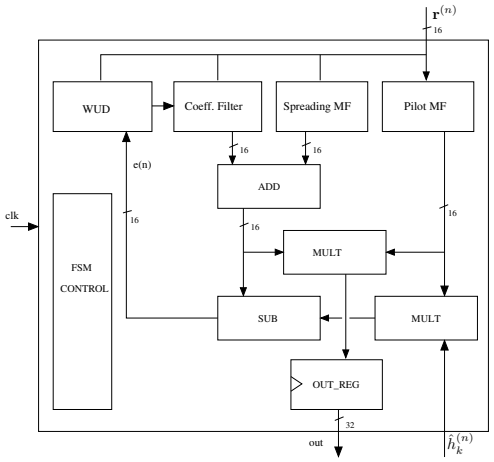


Fig. 3. Functional diagram of adaptive LMMSE finger structure

$$\begin{aligned} \mathbf{x}_{k,l}^{(n+1)} &= \mathbf{x}_{k,l}^{(n)} + 2\mu_{k,l}^{(n)} \left(g_{k,l}^{(n)} \hat{h}_k^{(n)} - y_{k,l}^{(n)} \right)^* \mathbf{r}^{(n)} \\ &= \mathbf{x}_{k,l}^{(n)} + 2\mu_{k,l}^{(n)} e_{k,l}^{*_{(n)}} \mathbf{r}^{(n)} \end{aligned} \tag{8}$$

where $\mu_{k,l}^{(n)}$ is the step-size parameter for the kth users lth path. The block diagram of a single receiver finger in the adaptive LMMSE receiver can be seen in Fig.2.

A. Implementation of conventional adaptive LMMSE core

In order to evaluate the performance of the our new LMMSE receiver core, a conventional LMMSE receiver core is implemented using the equivalent functional blocks without optimisation. Mathematically, the output of the pre-combining LMMSE filter requires the spreading codes and delays of all. Process intensive matrix inversion of the channel covariance matrix is also required and is ideally avoided. The LMMSE receiver is normally solved iteratively for each user by some adaptive algorithm such as the LMS algorithm. As a result, the implementation of this type of receiver is referred to as the adaptive LMMSE receiver.

An adaptive LMS based pre-combining LMMSE receiver is presented in [7] with a similar structure to that of a conventional RAKE receiver. This practical solution results in having separate LMMSE receivers for

each multi-path component. To satisfy the MSE solution, this adaptive implementation therefore needs to acquire the knowledge of the spreading sequence timing and channel coefficients for each multi-path component to generate an estimate of the data symbols. The adaptive LMMSE core therfore has multiple receiver fingers like the conventional RAKE receiver, the arrangement for one of which can be seen in Fig.2.

This implementation consists of an LMS weight update block (WUD) which uses the data input samples and an error signal to calculate new coefficients, and a FIR filter block, which employs the coefficients calculated by the LMS WUD block. This is used for detection using the calculated receiver coefficients. Matched filters (MF) are used for channel estimation and de-spreading. The functional blocks within the adaptive LMMSE receiver finger implementation can be seen in the diagram of Fig.3. More detail about these blocks is given:

- WUD: The LMS weight update block carries out the calculation of the receiver filter coefficients iteratively using the LMS algorithm.
- COEFF. FILTER: The calculated receiver coefficients are employed by this FIR filter to detect the user data symbols..
- SPREADING MF: The spreading matched filter is used for de-spreading and synchronisation of the received multi-path signal. The filter taps are matched to the PN spreading code.
- PILOT MF: The pilot channel matched filter is used to determine an estimation of the wireless channel coefficients. This uses the pilot channel code to estimate the channel coefficients for the lth path. Both the MF sub-blocks are based on an FIR filter structure in terms of their hardware implementation.
- AU: There are a number of distributed arithmetic units for calculation of the required adaptive LMMSE solution components. There are two adder units and a multiplier, all clocked at the symbol rate.
- FSM CONTROL: This finite state machine control block synchronises timing of all the components according to the symbol rate and determines the delay synchronisation of the matched filters.

B. Implementation of new adaptive LMMSE core

Following on from the implementation in the previous section, the adaptive LMMSE receiver is modified to include the variable length adaptive core, developed previously[8]. Use of this variable length LMS algorithm will optimise the computation required for calculation of receiver coefficients in every finger of the receiver and therefore reduce the overall power consumed.

The segemented FIR filter block which uses the receiver coefficients calculated by this WUD block, more

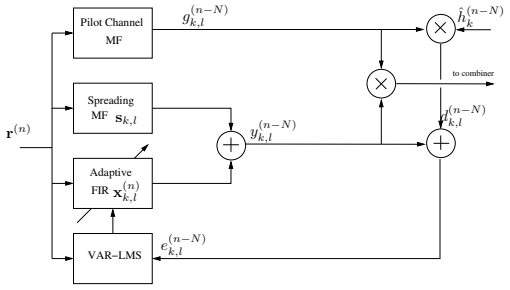


Fig. 4. Single adaptive LMMSE receiver finger with variable length LMS functionality

detail of which is also described there. All other blocks within a given receiver finger will remain the same as those in the conventional implementation for the purposes of this analysis. There are no changes needed to the FSM controller block as timing of the main functional block in the receiver finger remains the same.

The arrangement of functional blocks in the new adaptive LMMSE receiver finger can be seen in the diagram of Fig.4. The following sections present an analysis of the results found by making this comparison.

III. RESULTS

Conventional and low power architectures are coded in Verilog hardware description language at register transfer level (RTL). Two adaptive LMMSE receiver cores have therefore been implemented, the first being the conventional (CONV) and secondly the new implementation using the variable length LMS algorithm (OUR). Both have been analysed in terms of area usage and power consumption. The cores were designed using Verilog HDL and then synthesised using Design Compiler^{TM} targeting the UMC 0.18μ standard cell CMOS library. The requirements of the synthesis were identical for all cores to allow consistent power consumption and area usage comparisons. A netlist was created for each core and back-annotated netlist simulations for a uniformly distributed random input bipolar binary data of 10000 symbols using Verilog-XL^{TM} simulator were performed and verified against the Matlab^{TM} receiver core model. The resulting data, including switching activity of the circuit nets was then used by Synopsys DesignPower^{TM} to determine power consumption for the different adaptive filter cores. In all of the above stages a clock frequency of 100 MHz and a supply voltage of 1.8 Volts were used.

A minimum symbol rate of 16 kbits/s is achieved using a 31 chip Gold code was used in simulation with an equal energy two-path ($L = 2$) channel and a maximum delay spread of 10 chip intervals. The number of users K was kept low for this analysis and an unmodulated pilot channel was assumed to be present with equivalent energy to that of the user data channels. A five tap FIR filter is used to model the fading channel. The channel

TABLE I
POWER CONSUMPTION FOR CONV AND OUR ADAPTIVE LMMSE CORES

Core type	Dynamic Power (mW)
CONV	32.97
OUR	29.77

TABLE II
COMPARISON OF POWER CONSUMPTION FOR FUNCTIONAL BLOCKS IN BOTH LMMSE CORE IMPLEMENTATIONS

Block	*Conv* (mW)	*Our* (mW)	% Diff
LMS WUD	9.956	8.953	-10.1
FIR	6.693	6.065	-9.38
Pilot MF	6.692	6.065	-9.38
Spreading MF	6.692	6.065	-9.38
Mult	1.236	1.062	-14.07
Mult	1.237	1.061	-14.22
Add	0.231	0.231	0
Sub	0.231	0.231	0
Control	0.035	0.035	0

tap coefficients are set to random values exponentially decreasing in average power with exponent powers of 0 to -4 in unity steps.

A. Power Consumption for Conventional LMMSE core

An analysis of the power consumed by the conventional adaptive LMMSE core is made for comparison against the new receiver core which uses the VAR-LMS adaptive filter. The functional blocks consuming the highest power are identified. A fixed 32-tap adaptive filter was used with step-size μ = 0.01. The total power consumed by the conventional adaptive LMMSE core implementation of one receiver finger is 32.97mW. Power results are shown in Table I for the total power consumed by the receiver finger using conventional processing blocks.

The LMS weight update block consumes the most power, this is explained by the fact it is the most computationally complex. The results in Table II show the power consumed by each of the main functional blocks in a single receiver finger.

B. Power Consumption for New LMMSE core

Initial power results for the variable length LMS implementation of a single adaptive LMMSE receiver finger are shown in Table I. A 32-tap variable length adaptive filter was used to calculate receiver coefficients with step-size μ = 0.01 as before. In this case $K = 16$ and $P = 2$ taps/segment. A signal to noise ratio of 10 dB was chosen and the variable length LMS algorithm converged to a tap-length of 14 taps. It can clearly be concluded that the optimisation of the filter tap length has resulted in a power saving in comparison to the results for the conventional implementation given in Table I. The multipliers used in the low power LMMSE implementation have been optimised where possible and are implemented using the non-Booth encoded Wallace tree (*nbw*) type multiplier. This has brought a further power saving to the optimised LMMSE finger and is seen in the results shown. Most notably the power consumed by the FIR filter and matched filter blocks has been reduced when compared to the conventional implementation.

TABLE III
AREA ANALYSIS FOR DIFFERENT LMMSE RECEIVER CORES

Core	Area (mm^2)	% Difference
Conv	1.98	-
Our	2.04	+3.03

The LMS weight update block still consumes the greatest power but this has been targeted directly by the optimisation of the variable length update algorithm. Due to optimisation of the number of filter taps in operation, the switching activity in the multiplier is reduced and the accesses to the memory blocks storing coefficients within the receiver coefficient filter are also reduced. This is directly responsible for the reduction in power consumed. The results in Table II show the power consumed by each of the main functional blocks in the receiver finger. There is of course a hardware overhead in this implementation due to the control and error calculation logic required by length update algorithm but this is clearly mitigated by the power savings achieved. Area results are shown in Table III and it can be seen that the overhead introduced by *Our* implementation is very low in relation to the *Conv* adaptive LMMSE implementation.

IV. CONCLUSION

This paper has presented a novel architectural VLSI implementation of an adaptive LMMSE receiver core. The novel application of the variable length LMS adaptive algorithm for calculation of channel coefficients in the LMMSE receiver core was presented and has proved successful. The results demonstrate a power saving is achieved by optimising the number of taps used in the LMMSE filter coefficient calculations. Results have shown a power saving of 9.7% can be achieved using our receiver finger structure in comparison with the conventional adaptive LMMSE implementation. It has also been shown that the low-complexity of the additional circuitry needed for the variable length adaptive filter presents minimal overhead for this architecture.

REFERENCES

[1] A. Duel-Hallen and J. Holtzman and Z. Zvonar, *"Multiuser detection for CDMA systems,"* IEEE Personal Communications, Vol. 2, No. 2, pp.46-58, April 1995.

[2] M. Juntti and S. Glisic, *"Advanced CDMA for Wireless Communications, Wireless Communications: TDMA VERSUS CDMA,"* Kluwer, 4th Ed. pp. 447-490, 1997.

[3] S. Moshavi, *"Multi-user Detection for DS-CDMA Communications,"* IEEE Communications Magazine, Vol. 34, No. 10, pp.124-137, 1996.

[4] S. Werner and J. Lilleberg, *"Downlink channel decorrelation in CDMA systems with long codes,"* Proc. IEEE Vehicular Technology Conference, Houston, USA, Vol. 2, pp.1614-1617, July 16-19 1996.

[5] K. Hooli and M. Latva-aho and M. Juntti, *"Performance evaluation of adaptive chip-level channel equalisers in WCDMA downlink,"* Proc. IEEE International Conference on Communications, Helsinki, Finland, Vol. 6, pp.1974-1979, June 11-15 2001.

[6] M. Latva-aho and M. Juntti and I. Oppermann, *"Reconfigurable Adaptive RAKE Receiver for Wideband CDMA Systems,"* IEEE Vehicular Technology Conference, 1998. VTC 98, Vol. 3, pp.1740-1744, May 1998.

[7] M. Latva-aho and M. Juntti, *"Modified LMMSE Receiver for DS-CDMA - Part I: Performance Analysis and Adaptive Implementations,"* IEEE 5th International Symposium on Spread Spectrum Techniques and Applications, Vol. 2, pp.652-657, Sept. 1998.

[8] M.P. Tennant, A.T. Erdogan, T. Arslan and J. Thompson, *"A Novel Equaliser Architecture with Dynamic Length Optimisation,"* IEEE Symposium on Circuits and Systems 2006, Kos, Greece. 21-24 May 2006.

Appendix B
Matlab Code for the DECOR LMS Filter

B.1 Main section of code

```
clear;

n_bits = 15;
fs=1000;
fn=fs/2;
taps=73;                        % FIR filter tap length
low1=fir1(taps-1,0.2);          % transition edge 0.1 Fn

t= 10000;                       % The number of input data
bit_width= 16;                  % The coefficient bit_width


% Coefficient generator

coeff_16b = round((low1/max(abs(low1)))*32767); % 16 bit value
coeff_q = low1;

c0 = [coeff_q 0];
c1 = [0 coeff_q];
cin = c0-c1;

cin = (round(cin*2^n_bits));
Cc = cin/2^n_bits;

freqz(coeff_16b);
figure;plot(cin);




% Create input signals
x = ((rand(1,t)>0.5)-.5);


var = std(x);
```

```
noise=0.5*(randn(1,(t))*var);



% Standard FIR filter
FIR_coeff = coeff_q;
FIR_std = filter(FIR_coeff,1,x);

% DECOR filter

DEC_ref = filter(Cc,[1 -1],x);
DEC_approx = filter(Cc,[1 -0.99],x);

den = [-1];                      %den = -1 to implement feedback loop.

q = length(Cc);
p = length(den);

X_n = zeros( 1 , q );           %shift register for x
Y_n = zeros( 1 , p );           %shift register for y


n = zeros(length(x),1);
for n = 1:length(x),
      X_n(2:q) = X_n(1:q-1);    %shift first q-1 values by 1
      X_n(1) = x(n);            %shift in next value of x

      DEC_std(n) = Cc*X_n' - den*Y_n';    %convolution sum

      Y_n(2:p) = Y_n(1:p-1);    %shift first p-1 values by 1
      Y_n(1) = DEC_std(n);      %shift in next value of y
end;



%DECOR LMS code

del = taps+1;
pad = zeros(1,del);

delta = zeros(1,del);

mu = 0.0002;

C = zeros(1,taps+1);                % adaptive coefficients are initialised to zero
W = zeros(1,taps);                  % adaptive coefficients are initialised to zero
```

```
x = [pad, x];
DEC_out = [delta, DEC_std];


n = zeros(length(x),1);
e = zeros(length(x),1);
y = zeros(length(x),1);
dec = zeros(length(x),1);



for n = del:length(x),
   X = x(n:-1:n-del+1);

   dec(n) = C([1])*X([1]) + C([2:del-1])*X([2:del-1])' - C([del])*X([del]);
   dec(n) = dec(n) + 0.99*dec([n-1]);



   X_FIR = X([1:del-1]);
   y(n) = W*X_FIR';


   e(n) = DEC_out(n) - dec(n);


   %lms_X = X([1:del-1]);
   W = W + 2*mu*e(n)*X_FIR;

   %W = round(W*2^n_bits)/2^n_bits;                  % Quantise weights for DECOR


   init = n-taps;
   if (init) < 74
       dec_W = W([1:init]);
       dec_X = X([1:init]);
   else
       dec_W = W;
       dec_X = X([1:del-1]);
   end;

   c0=[dec_W 0];
   c1=[0 dec_W];

   C=(c0-c1);

   sizeof = size(C);
   sizeof = sizeof([2]);
```

```
    pad_C = zeros(1,74-sizeof);

    C = [C pad_C];
    C = (round(C*2^n_bits))/2^n_bits;                % Quantise weights for DECOR

end;
```

Appendix C
Verilog Code for the DECOR LMS adaptive Filter

C.1 Main section of code

```
////////////////////////////////////////////////////////////////////////////////

/* TOP LEVEL LMS CODE */

////////////////////////////////////////////////
// implement LMS algorithm for N filter taps //
////////////////////////////////////////////////

module LMS (clk, error, x_input, mu_val, reset_n, count, scanEnable, scanIn, scanOut, tap_W);

parameter word_size = 'word_size;
parameter counter_width = 'counter_width;

input [word_size-1:0] error;
input [word_size-1:0] x_input;
input [word_size-1:0] mu_val;
input [counter_width-1:0] count;
input clk, reset_n, scanEnable, scanIn;

output  scanOut;

output [word_size-1:0] tap_W;

wire LMS_control_en, coeff_en, write_en, shift_clk, clk;
wire [counter_width-1:0] count, address;
wire [word_size-1:0] coeff_ans;
wire [word_size-1:0] shift_output;
wire [word_size-1:0] W_0, FIR_tap_W;

LMS_control  control   (.count(count), .reset_n(reset_n),
 .LMS_en(LMS_control_en));

LMS_counter  LMS_address  (.out(address), .cnt(LMS_control_en), .reset_n(reset_n), .clk(clk),
.scanEnable(scanEnable), .scanIn(scanIn), .scanOut(scanOut));

coeff_enable coeff_EN (.out_en(coeff_en), .count(address));

shift_enable  enable (.out_en(shift_clk), .count(address));
```

```
multiply_coeff   val  (.err_in(error), .u_in(mu_val), .coeff_en(coeff_en),
.coeff_res(coeff_ans), .reset_n(reset_n), .scanEnable(scanEnable));


input_shift_reg  shift (.x_in(x_input), .shift_clk(shift_clk), .shift_addr(address), .shift_out(shift_output),
.reset_n(reset_n), .scan_En(scanEnable));


arith_unit  calc0    (.coeff_in(coeff_ans), .x_shift_in(shift_output), .W_out(W_0), .W_in(FIR_tap_W));


my_DW_ram_r_w_s_dff W_RAM (.data_out(tap_W), .data_in(W_0), .addr_read(count), .addr_write(address),
.wr_n(1'b0), .reset_n(reset_n), .clk(clk));

my_DW_ram_r_w_s_dff FIR_RAM (.data_out(FIR_tap_W), .data_in(tap_W), .addr_read(address), .addr_write(count),
.wr_n(1'b0), .reset_n(reset_n), .clk(clk));



endmodule // LMS

////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
```

C.2 Decor section of code

```
/* Decor_block.v

21/02/05  M.Tennant

*/


//default word size
`define word_size 16
`define counter_width 7



//////////////////////////////////////////
// 16 bit flip flop with active low reset //
//    //
//////////////////////////////////////////


module d_latch_dec (out, d, reset_n, clk, scanEnable);

parameter word_size = `word_size;

   input [word_size-1:0]  d;

   input    reset_n, clk;

   input    scanEnable;
```

```
   output [word_size-1:0] out;

   reg [word_size-1:0]    out;


   always @(posedge clk or negedge reset_n)

      if (!reset_n)
    out <= #1 0;

      else if(scanEnable)
 out <= {out[word_size-1:1]};

      else
    out <= #1 d;


endmodule // d_latch




//////////////
// Subtract //
//////////////

module dec_sub (diff, in_0, in_1, CI, CO);

  parameter width = ‘word_size;
  parameter coefficient_size = ‘coefficient_size;

  // port decalrations

  output [coefficient_size-1 : 0]  diff;
  output                  CO;

  input  [width-1 : 0]  in_0;
  input  [width-1 : 0]     in_1;
  input                   CI;

  wire  [width-1 : 0] diff_out;



  assign diff = diff_out[coefficient_size-1:0];



DW01_sub #(width) dec_DW01_sub ( .A(in_0),
.B(in_1),
.CI(CI),
.DIFF(diff_out),
.CO(CO));


  endmodule  // DW01_sub;



////////////////////////////////////////////////////////////////////////////////
```

```
//////////////////////////////
// Implement Decor Block //
//////////////////////////////

module DECOR(w, decor, clk, reset_n, scanEnable);

parameter word_size = ‘word_size;
parameter counter_width = ‘counter_width;
parameter coefficient_size = ‘coefficient_size;



   input  [word_size-1:0]  w;
   input scanEnable, reset_n;
   input clk;

   output  [coefficient_size-1:0] decor;


   wire [word_size-1:0] w_1;



dec_sub  sub_unit (.diff(decor), .in_0(w), .in_1(w_1), .CI(1’b0), .CO());

d_latch_dec w_latch (.out(w_1), .d(w), .reset_n(reset_n), .clk(clk),
.scanEnable(scanEnable));


endmodule // subtract
```

Appendix D
MATLAB Code for the Variable Length Equaliser

D.1 Main section of code

```
clear;

delta = 2;

n_bits=15;
t=5000;

x = 0.5*((rand(1,t)>0.5)-.5);
r = 1;


imp = [-4.4 0 -7 -12 -9 -14 -20 -16.5 -20 -22.5 -20];
response = (10.^(imp/10));

quant = round(response*(2^n_bits));
quant = quant/2^n_bits;

channel = [0.4 1 0.2 0 0.06 0.1 0.04 0.01 0.02 0.01 0.005 0.01];
channel = channel.*2;


b = (randn(1,5)).*exp(-(0:4));

c = [1 0 -.3 0 .5 0 -.1 0 0.1];




% Generate noise
snr = 20;

n=1/sqrt(2).*(randn(1,length(x)));
noise = 10^(-snr/20)*n;
noise = 0.25.*noise;
```

```
var = 1;

noise_low_SN=0.55*(randn(1,t*r)*sqrt(var));
noise_low_q=round(noise_low_SN*2^n_bits)/2^n_bits;

noise_high_SN=0.01*(randn(1,t*r)*sqrt(var));
noise_high_q=round(noise_high_SN*2^n_bits)/2^n_bits;

noise = [noise_high_q];




% input signal
input = (filter(c,1,x));
%input = (x);


pad = zeros(1,delta);
pad1 = ones(1,4);

input_LMS = round(input*2^n_bits)/2^n_bits;
input_q = round(input*2^n_bits)/2^n_bits;             %Quantisation for file output


input_des=round(x*2^n_bits)/2^n_bits;



% adaptive filtering
L = 9;               % filter length
P = 3;               % sub-filter size
mu = 0.005;          % step size ( mu < 1/(L*std(x)^2) )
W = zeros(1,L);      % adaptive coefficients are initialised to zero

ASE = 50;

e_array = zeros(1,ASE);
e_sub_array = zeros(1,50);

pad = zeros(1,L);
x_pad = zeros(1,L-3);
x_delay = [pad x];
d = [x_pad, input_LMS];


y = zeros(1,length(x));
```

```
e = zeros(1,length(x));

y_sub = zeros(1,length(x));
e_sub = zeros(1,length(x));

coeff = zeros(1,length(x));

for n = L:length(x),
   X = d(n:-1:n-L+1);
   y(n) = round((W*X')*2^n_bits)/2^n_bits;                 % adaptive filter out



   X_sub = d(n:-1:n-L+(P+1));
   W_sub = W(1:1:(L-P));
   y_sub(n) = round((W_sub*X_sub')*2^n_bits)/2^n_bits;     % adaptive sub-filter out


   % error signals
   e(n) = (round((x_delay(n) - y(n))*2^n_bits))/2^n_bits;
   e_sub(n) = (round((x_delay(n) - y_sub(n))*2^n_bits))/2^n_bits;

   e_array(2:ASE) = e_array(1:ASE-1);                      % calculation of ASE
   e_array(1) = (e(n))^2;

   e_sub_array(2:ASE) = e_sub_array(1:ASE-1);
   e_sub_array(1) = (e_sub(n))^2;

   ASE_res = sum(e_array);
   ASE_res_sub = sum(e_sub_array);

   if (ASE_res <= ASE_res_sub) % evaluate Filter length
    L = L + P; % to add or remove P taps
   elseif (ASE_res >= ASE_res_sub)
    L = L - P;
   end

   coeff(n) = (round(2*mu*e(n)*2^n_bits)/2^n_bits);
   W = W + coeff(n)*X;                                          % LMS update

   W=(round(W*2^n_bits)/2^n_bits);


   e_sq(n) = e(n)^2;

end;
```

Appendix E
Verilog Code for the Variable Length Equaliser

E.1 Main section of code

```
This is the top level adaptive filter module.
*/

//////////////////////////////
// Adaptive Filter          //
//////////////////////////////
‘timescale 1ns/1ps

‘define word_size 16
‘define coefficient_size 16
‘define counter_width 7

‘define counter_size 72

‘define enable_val 71
‘define enable_val_sub 2

module adapt (y, y_dat, data, cnt, error, error_sub, count, size, init_tap, out_en, reset_n, clk, scan_en,
scan_in, scan_out, mu, tap_W, LMS_size);

parameter word_size  = ‘word_size;
parameter coefficient_size  = ‘coefficient_size;
parameter counter_width  = ‘counter_width;

parameter error_val  = ‘enable_val;
parameter error_val_sub  = ‘enable_val_sub;


input [word_size-1:0]      data;
    input [word_size-1:0]    y_dat, mu;
input [counter_width-1:0] size, init_tap;
input     cnt, reset_n, clk;
    input     scan_in, scan_en;


    output     scan_out, out_en;
    output [word_size-1:0]     y, tap_W;
output [word_size-1:0]     error, error_sub;
output [counter_width-1:0] LMS_size, count;



wire   [word_size-1:0]    x_m, tap_W, y_acc;

wire   [counter_width-1:0] count;
```

```
   FIR  filter (.data(data), .x_m(x_m), .cnt(cnt), .reset_n(reset_n), .clk(clk),
    .scan_in(scan_in), .scan_en(scan_en), .acc_round(y_acc),
.out_en(out_en), .y(y), .count(count), .tap_W(tap_W));

   subtract #(word_size, counter_width, error_val) sub  (.y(y_acc), .y_dat(y_dat), .error(error), .count(count),
    .clk(clk), .reset_n(reset_n));


   subtract #(word_size, counter_width, error_val_sub) sub_minus (.y(y_acc), .y_dat(y_dat), .error(error_sub), .count(count),
    .clk(clk), .reset_n(reset_n));


   LMS  adaptive  (.clk(clk), .error(error), .mu_val(mu), .size(size), .reset_n(reset_n),
    .count(count), .x_m(x_m), .tap_W(tap_W), .LMS_size(LMS_size),
.scanEnable(scan_en), .scanIn(scan_in), .scanOut(scan_out));


   tap_size tap_calc (.error(error), .error_sub(error_sub), .count(count),
    .clk(clk), .init_tap(init_tap), .tap_size(),
.alpha_up(), .alpha_dw(), .reset_n(reset_n), .scanEnable(scan_en));


endmodule // ADAPT


////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////
```

E.2 Tap size Calculation

```
////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////

/* tap_size.v */



//////////////
// Adder    //
//////////////


module tap_add (A,B,CI,SUM,CO);

  parameter width = 32;

  // port decalrations

  output [width-1 : 0]   SUM;
  output                 CO;
```

```
  input  [width-1 : 0]   A,B;
  input                  CI;


DW01_add #(width) U_DW01_add ( .A(A),
.B(B),
.CI(CI),
.SUM(SUM),
.CO(CO));


endmodule  // DW01_add;



//////////////////////
// Multiplies a and b //
//////////////////////

module tap_mult (prod, in_0, in_1, tc);



parameter word_size = 'word_size;

parameter word_size_a  = word_size;
parameter word_size_b  = word_size;



input [word_size_a-1 : 0]  in_0;
input [word_size_b-1 : 0] in_1;
input  tc;

output [word_size_a+word_size_b-1 : 0] prod;


DW02_mult #(word_size_a, word_size_b) tap_U_DW02_mult   (       .A(in_0),
.B(in_1),
.TC(tc),     // 1'b1 for two's complement
.PRODUCT(prod));


endmodule



/////////////////
// multiplexer //
/////////////////

module tap_mux(out, in_0, in_1, sel);

parameter word_size = 'word_size;

   input  [word_size*2-1:0]     in_0;
   input       in_1, sel;

   output  [word_size*2-1:0]  out;

   reg  [word_size*2-1:0]     out;

   always @(in_0 or in_1 or sel)
```

```
      begin
 if (!sel)
    out = in_0;
 else
    out = in_1;
      end // always @ (in_0 or in_1 or sel)

endmodule // mux




////////////////////////////////////////////
// 16 bit flip flop with active low reset //
//    //
////////////////////////////////////////////


module d_latch_acc (out, d, reset_n, clk, scanEnable);

parameter word_size = 'word_size;

   input [word_size*2-1:0]  d;

   input    reset_n, clk;

   input    scanEnable;


   output [word_size*2-1:0] out;

   reg [word_size*2-1:0]    out;


   always @(posedge clk or negedge reset_n)
      if (!reset_n)
    out <= #1 0;

      else if(scanEnable)
 out <= {out[word_size-1:1]};

      else
    out <= #1 d;


endmodule // d_latch



////////////////////////////////////////////
// rounds output (32 bits) to 16 bit   //
////////////////////////////////////////////

module round_err (out_16, in_32);

parameter word_size = 'word_size;

   input [word_size+word_size-1:0] in_32;

   output [word_size-1:0]     out_16;

   wire [word_size-1:0]     out_16;
```

```
   reg [word_size:0]     round;

   always @(in_32)
      begin
 round = in_32[30:14] + 17'h 00001;
      end // always @ (in_32)

   assign out_16 = round[16:1];

endmodule // round_adder


////////////////////////////////////////
// rounds output (32 bits) to 16 bit   //
////////////////////////////////////////

module trans_err (out_32, in_32);

parameter word_size = 'word_size;

   input  [word_size+word_size-1:0]  in_32;

   output  [word_size+word_size-1:0]      out_32;

   wire  [word_size+word_size-1:0]      out_32;



   assign out_32 = {3'b 0, in_32[31:3]};

endmodule // round_adder



////////////////////////////////////////////
// create enable signal for accumulator     //
////////////////////////////////////////////

module acc_enable (out_en, count);

parameter counter_width = 'counter_width;

   input [counter_width-1:0] count;

   output       out_en;

   reg       out_en;

   always @(count)
      if (count == 7'h 00)
   out_en = 1;

      else out_en = 0;

endmodule // out_enable



////////////////////////////////////////////
// Counter module to count from zero to 23 //
////////////////////////////////////////////

'define acc_counter_size 254

module acc_counter (out, cnt, reset_n, clk, scanEnable, scanIn, scanOut);
```

```
parameter counter_width = ‘counter_width;
parameter acc_counter_size = ‘acc_counter_size;



   input                     cnt, reset_n, clk;

   input        scanEnable, scanIn;

   output        scanOut;

   output [counter_width-1:0] out;

   reg [counter_width-1:0]    out;


   always @(posedge clk or negedge reset_n)
      if (!reset_n)              // reset
 out <= #1 acc_counter_size;

      else if (scanEnable)
 out <= {scanIn, out[counter_width-1:1]};

      else
 if (cnt)
    if (out == 0)            // if out is zero
       out <= #1 acc_counter_size;         // set out to 23
    else
       if (out > acc_counter_size)         // if out greater than 23
  out <= #1 acc_counter_size;      // reset out to 23
       else
  out <= #1 out - 1; // otherwise subtract one
 else
    out <= #1 out;           // if cnt not high don’t do anything

assign scanOut = out[0];

endmodule // counter




//////////////////////////////
// Implement Error Calc     //
//////////////////////////////


module MS_error_calc (error, acc_error, ase_en, reset_n, scanEnable, ctr_en);

parameter word_size = ‘word_size;
parameter counter_size = ‘counter_size;
parameter counter_width = ‘counter_width;



    input  ase_en;
    input  [word_size-1:0]  error;

    input scanEnable, reset_n;

    output [word_size-1:0] acc_error;
    output ctr_en;
```

```
    wire [word_size+word_size-1:0] ase_32, mux_out, d, accu, comp;
    wire [word_size-1:0] err_hold, rnd_error, ase_hold;
    wire [counter_width-1:0] sample;
    wire ctr_en;


//d_latch_err err_latch (.out(), .d(), .reset_n(reset_n), .clk(ase_en),
// .scanEnable(scanEnable));

tap_mult mult_unit (.prod(ase_32), .in_0(error), .in_1(error), .tc(1'b1));

trans_err compress (.in_32(ase_32), .out_32(comp));

round_err round_ase (.in_32(accu), .out_16(acc_error));

tap_mux  acc_control (.out(mux_out), .in_0(accu), .in_1(1'b0), .sel(ctr_en));

tap_add  add  (.A(comp), .B(mux_out), .CI(1'b0), .SUM(d), .CO(no_connect1) );

d_latch_acc acc_latch (.out(accu), .d(d), .reset_n(reset_n), .clk(ase_en),
.scanEnable(scanEnable));

acc_counter sample_count (.out(sample), .cnt(1'b1), .reset_n(reset_n), .clk(ase_en),
.scanEnable(scanEnable), .scanIn(), .scanOut());

acc_enable acc_ctr (.count(sample), .out_en(ctr_en));



endmodule //

////////////////////////////////////////////////////////////////////////////////////////////////////



////////////////////////////////////////////
// create enable signal for output register //
////////////////////////////////////////////

module ASE_enable (out_en, count);

parameter counter_width = 'counter_width;

   input [counter_width-1:0] count;

   output       out_en;

   reg       out_en;

   always @(count)
      if (count == 7'h 45)
   out_en = 1;

      else out_en = 0;

endmodule // out_enable




//////////////////////
// Up Down Counter //
//////////////////////
```

```
module up_dwn_count (data, up_dn, load, cen, clk, reset, count, tercnt);

  parameter width = 7;

  // port decalrations

  output [width-1 : 0]   count;
  output tercnt;

  input  [width-1 : 0]  data;
  input                   up_dn, load, cen, clk, reset;



DW03_updn_ctr #(width) TAPS_DWO3_updn_ctr (.data(data), .up_dn(up_dn), .load(load), .cen(cen), .clk(clk),
.reset(reset), .count(count), .tercnt(tercnt));


endmodule  // DW01_sub;




////////////////////////////////
// state_machine control       //
////////////////////////////////


module control_up_down (clk, ctr_en, pos_neg, grad, limit_up, limit_dn, reset_n, clk_up_dn, load, up_down, alpha_sel);



//parameter counter_width = 'counter_width;

parameter start   = 0;
parameter init    = 1;
parameter ctr_hold = 2;
parameter comp   = 3;
parameter assert_up   = 4;
parameter assert_dn = 5;
parameter clk_L         = 6;
parameter clk_G   = 7;
parameter wait_L = 8;
parameter wait_G = 9;

input  clk, ctr_en, pos_neg, grad, limit_up, limit_dn, reset_n;

output  clk_up_dn, load, up_down, alpha_sel;

reg  [3:0]  current_state, next_state;
reg load;
reg up_down;
reg clk_up_dn;
reg alpha_sel;



always @(reset_n or clk)begin
if(!reset_n)begin
current_state = 0;
load = 1;
up_down = 0;
clk_up_dn = 0;
```

```
end
end


always @(posedge clk)begin
if (current_state == start) begin
if (reset_n == 1) next_state = init;
else next_state = start;
end


else if (current_state == init) begin
load = 0;
next_state = ctr_hold;
end


else if (current_state == ctr_hold) begin
load = 1;
alpha_sel = 0;
if (ctr_en == 0) next_state = ctr_hold;
else if (ctr_en == 1) next_state = comp;
else next_state = ctr_hold;
end


else if (current_state == comp) begin
if (pos_neg == 1) next_state = assert_dn;
else if (pos_neg == 0) next_state = assert_up;
else next_state = comp;
end


else if (current_state == assert_dn) begin
up_down = 0;
if (limit_dn == 1) next_state = clk_L;
else if (limit_dn == 0) next_state = wait_L;
//next_state = clk_L;
end


else if (current_state == assert_up) begin
up_down = 1;
if (limit_up == 1) next_state = clk_G;
else if (limit_up == 0) next_state = wait_G; //limit
//next_state = clk_G;
end


else if (current_state == clk_L) begin
clk_up_dn = 1;
next_state = wait_L;
end


else if (current_state == clk_G) begin
clk_up_dn = 1;
next_state = wait_G;
end


else if (current_state == wait_L) begin
clk_up_dn = 0;
if (ctr_en == 1) next_state = wait_L;
else if (ctr_en == 0) next_state = ctr_hold;
```

```
else next_state = wait_L;
end


else if (current_state == wait_G) begin
clk_up_dn = 0;
if (ctr_en == 1) next_state = wait_G;
else if (ctr_en == 0) next_state = ctr_hold;
else next_state = wait_G;
end



current_state = next_state;

end

endmodule




////////////////////////////////////////////////////////////////////////////////


//////////////////////////////
// Implement Tap Size Block //
//////////////////////////////

module tap_size (clk, error, error_sub, count, tap_size, init_tap, alpha_up, alpha_dw,
reset_n, scanEnable);

parameter word_size = 'word_size;
parameter counter_width = 'counter_width;

   input  [word_size-1:0]  error, error_sub, alpha_up, alpha_dw;
   input [counter_width-1:0] count;
   input [counter_width-1:0] init_tap;

   input clk, scanEnable, reset_n;

   output [counter_width-1:0] tap_size;

   wire [word_size-1:0] error, error_sub, acc_sub, acc_final, latch_sub, latch_final, ABS_diff, ASE_diff;



ASE_enable  err_enable (.out_en(ase_en), .count(count));



MS_error_calc calc_sub (.error(error_sub), .acc_error(acc_sub), .ase_en(ase_en),
.reset_n(reset_n), .scanEnable(scanEnable), .ctr_en(ctr_en_sub));

MS_error_calc calc_final (.error(error), .acc_error(acc_final), .ase_en(ase_en),
.reset_n(reset_n), .scanEnable(scanEnable), .ctr_en(ctr_en));

d_latch_err ase_latch_sub (.out(latch_sub), .d(acc_sub), .reset_n(reset_n), .clk(ctr_en_sub),
.scanEnable(scanEnable));


d_latch_err ase_latch_final (.out(latch_final), .d(acc_final), .reset_n(reset_n), .clk(ctr_en),
.scanEnable(scanEnable));
```

```
tap_sub  error_diff (.diff(ASE_diff), .in_0(latch_sub), .in_1(latch_final), .CI(1'b0), .CO());

sign ASE_sign_of (.in(ASE_diff), .sign_of(ASE_sign));

absval absval_ASE_diff (.in(ASE_diff), .out(ABS_diff));

ASE_diff_lim_up ASE_lim_up (.in0(ABS_diff), .comp(limit_up));

ASE_diff_lim_dn ASE_lim_dn (.in0(ABS_diff), .comp(limit_dn));

control_up_down control_cnt (.clk(clk), .ctr_en(ctr_en), .pos_neg(ASE_sign), .reset_n(reset_n),
.clk_up_dn(clk_up_dn), .load(load), .up_down(up_dn),
.grad(), .limit_up(limit_up), .limit_dn(limit_dn),
.alpha_sel(alpha_sel));

up_dwn_count  taps (.data(init_tap), .up_dn(up_dn), .load(load), .cen(clk_up_dn), .clk(clk),
.reset(reset_n), .count(tap_size), .tercnt());



endmodule // tap_size

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
```