1-1-2011

Measuring the Power Efficiency Of Subthreshold FPGAs For Implementing Portable Biomedical Applications

Shahin S. Lotfabadi
Ryerson University

Measuring the Power Efficiency of Subthreshold FPGAs for Implementing Portable Biomedical Applications

By

Shahin Sanayei Lotfabadi
Bachelor of Engineering, Ryerson University, 1998

A thesis
presented to Ryerson University
in partial fulfillment of the requirements
for the degree of Master of Applied Science
in the program of
Electrical and Computer Engineering

Toronto, Ontario, Canada

Author's Declaration

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Signature

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Signature

Abstract

Measuring the Power Efficiency of Subthreshold FPGAs for Implementing Portable Biomedical Applications

Shahin Sanayei Lotfabadi

Master of Applied Science

Department of Electrical and Computer Engineering

Ryerson University, 2011

Power is a significant design constraint for implementing portable applications. Operating transistors in the subthreshold region can significantly reduce power consumption while reducing performance. The low frequency nature of biosignals makes a FPGA operating subthreshold region a good candidate. In this work, I investigate the feasibility of designing such a device and the trade-off between power consumption and performance for FPGA routing resources operating in the subthreshold region. For the 32nm Predictive Technology Model studied in this work, it was observed a power reduction of 197.7 times (or power-delay-product reduction of 3.3 times) for operating under a supply voltage of 0.4 volts (as compared to normal operation in the saturation region using a 0.9V). Under a supply voltage of 0.4 volts, the FPGA can operate at 2.0 MHz while allowing signals to propagate unregistered through 20 routing tracks which meets the real-time requirement for processing 20000 samples per second.

Acknowledgments

I would like to express my deep gratitude to Professor Sridhar Krishnan and Professor Lev Kirischian at Ryerson University and Professor Eric Peskin at the Center for Health Informatics and Bioinformatics of New York University, for their knowledgeable guidance, encouragement, and constant support.

Finally, I would like to especially thank my family for their nonstop and warm support.

Contents

List of Figures

List of Tables

List of Acronyms

AR - Autoregressive

ARMA - Autoregressive moving-average

ASIC - Application-specific integrated circuit

DSP - Digital signal processing

ECG - Electrocardiogram

EDA - Electronic design automation

EEG - Electroencephalogram

EGG - Electrogastrogram

EMG - Electromyogram

FFT - Fast Fourier transform

FPGA - Field programmable gate array

HDL - Hardware description language

IP - Intellectual property

LMS - Least-mean-square

MA - Moving-average

RLS - Recursive least-squares

RLSL - Recursive least-squares lattice

STFT - Short-time Fourier transform

SBB - Swapped Body Biasing

VAG - Vibroarthrogram

VHDL - VHSIC hardware description language

VLSI - Very large scale integrated circuit

Chapter 1

Introduction

1.1 Motivation

Digital signal processing has played a significant role in the diagnostic and research activities in the health care field. Field Programmable Gate Arrays (FPGAs) are an important platform for implementing a variety of digital applications due to their short time to market, re-programmability and low non-recurring engineering costs. While FPGAs have been successfully used in many applications including digital signal processing, aerospace, medical imaging, computer vision, speech recognition, and ASIC prototyping, they have not been widely used in portable applications. FPGAs are not widely used in portable applications primarily due to their significant power consumption. In particular, previous studies [1], [2] have shown that applications implemented on FPGAs can consume significantly more power than the same applications implemented on ASICs. Portable applications, however, demand long battery life and consequently require an ultra low power implementation platform.

One way to extend the battery life of portable applications is to design digital systems that operate in the subthreshold region. Previous works have shown that, by operating in the subthreshold region, digital circuits often achieve minimum power-delay product thus

minimizing their overall energy consumption [3]. Operating circuits in the subthreshold region, however, does significantly reduce their performance. This reduction in performance limits the applicability of subthreshold design in many applications. Performance reduction, however, has significantly less impact on biomedical applications due to the low frequency nature of biosignals. In this research, I investigate the feasibility of designing a specialized FPGA that operates in the subthreshold region in order to reduce the power consumption of biomedical applications while still maintaining real-time signal processing capabilities.

This research is based on a case study of the Burg algorithm [4], a widely used signal processing algorithm in biomedical applications. I first implemented a scalable RTL implementation of the Burg algorithm targeting Autoregressive (AR) modeling applications [5], [6], which is not possible using automated tools such as C-to-FPGA [22], Stateflow diagram to VHDL (SF2VHD) [23], or Simulink-to-FPGA [24].

Based on the design, the maximum operating frequency that guaranties the real-time processing of biosignals is calculated. This maximum operating frequency is then used as the performance constraint in the design of FPGA routing resources that operate in the subthreshold region. The power efficiency of the subthreshold design is then measured by determining the minimum supply voltage that is required to meet the real time performance constraint. These performance and power consumption figures are compared to the performance and power consumption of the conventional FPGA routing resources to quantify power efficiency.

1.2 Research Objectives

The objective of this research is to employ Burg algorithm [4], a widely used signal processing algorithm in biomedical applications, and to investigate subthreshold circuit design for an FPGA that result in reduction of its power-delay product and ultimately longer battery life

of the device. Figure 1.1 indicates the overview of the design. This design assumes that non-stationary biomedical signals have already been converted into stationary frames through an adaptive segmentation filter. The AR modeling generates processes stationary signals to generate coefficients which could co-relate to the physiological sources of the signal.

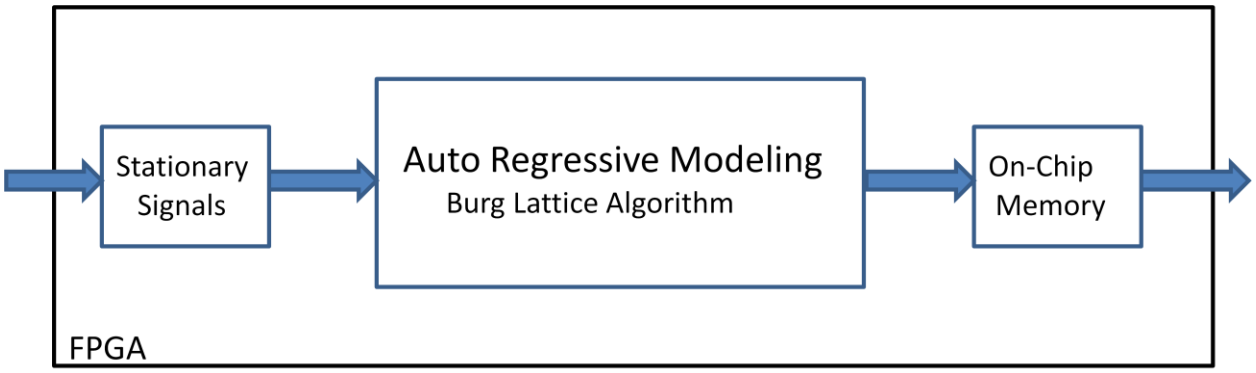


Figure1.1: System Level Block Diagram of the Research

This design is capable of storing parameters generated by AR modeling for up to 24 hours collection of non-stationary signals on a hand held device. The design is also facilitated with the capability to select the number of stages of the filter and hence increase number of parameters generated per input frame. Hence the design caries the same flexibility of a microprocessor based designs, and offers better energy efficiency.

1.3 Original Contributions

The main contributions of this research are described as follows:

Autoregressive Modeling

- Proposing the architecture for a generic Burg-lattice algorithm that can programmed to set the desired stages of the model and to generate AR coefficients.

- Using the proposed model to reduce the area and increase performance of design.
- Developing the required software using Very High Speed Description Language (VHDL), simulating and implementing the design on a Xilinx Virtex 5 FPGA (XC5VLX110-3FF676C).

Subthreshold Circuit Design

- Developing an HSPICE simulation model for the FPGA routing track to find the power-delay product.
- Investigating the performance of the circuit in subthreshold region and the effect of transistor sizing to achieve the desired performance.

1.4 Thesis Organization

This thesis consists of five chapters:

- Chapter 1 introduces the significance of biomedical signal analysis and suggests a design methodology as well as subthreshold design to improve power efficiency of the design. It also states the objectives of the research, the contribution of the author and how the thesis is organized.
- Chapter 2 starts with a review the method of processing biomedical signals presented in this thesis and advantages of this method of implementation. It provides an overview of parametric modeling, particularly AR modeling and the advantages of Burg-lattice algorithm for hardware implementation. It also presents a review of subthreshold circuit design and a study of the power-delay product for this application.

- Chapter 3 presents the design specification and design methodology including details of the architecture implementation, simulation, synthesis, and place and route. It also provides a discussion of advantage of sequential design versus structural designs offer by automated tools such as Simulink-to-FPGA.
- Chapter 4 presents the architecture of routing tracks of an FPGA designed for biomedical signal processing. It also presents simulation of an HSPICE model of this architecture to determine minimum power consumption of the routing track while performance constraint is met.
- Chapter 5 presents a discussion of the acquired results of this research project and offers conclusion and future work for the thesis.

Chapter 2

Review

The main focus in this research is to investigate ways to reduce total power consumption of biomedical applications. As it was discussed earlier low power consumption is a crucial factor in designing devices to process biosignals. For example the battery life of a permanent pacemake lasts 5-15 years and a surgical procedure is required to replace the battery. Hence reduction in power consumption for such devices results in longer battery life that eliminates the burden of a surgical procedure for a patient to replace the battery. In this research it is shown that parametric modeling that is often considered for feature extraction can also be used to compress the biomedical signal. This reduces the size of required memory and ultimately results in less power consumption. Selecting an appropriate algorithm that is less computationally extensive may also optimize the dynamic power consumption of a device. However, in chapter 3, it will be shown that a significant amount of power consumed by a device is static. To reduce the static power consumption of a device subthreshold circuit design is investigated in this research. A review of parametric modeling and subthreshold circuit design is presented in the following sections.

2.1 Parametric modeling

Signals and systems can be concisely and efficiently represented using parametric modeling technique. Parametric modeling is a technique to find the parameters of a mathematical model which describes a signal or a system [4]. The aim of the parametric modeling is to analysis the biosignal of interest, because modeling of biomedical signals provides parameters which could co-relate to the physiological sources of the signal. In parametric modeling the present value of the output is the sum of linear combination of several past output values and present and past values of the input as it shown in the following equation [4]:

(2.1)

where =1, is the input to the system, and is the output of the system. The transfer function of the equation 2.1 can be obtained applying z-transform:

(2.2)

In most cases the system can be fully characterized by parameters and , and not by the gain [4]. Hence from these two parameters it can be determined if the system is an all-pole system, an all-zero system or a pole-zero system. The main modeling methods are: AR (Autoregressive), MA (Moving average), and ARMA (Autoregressive moving-average) modeling. If the values for in Equation (2.2) are all equal to zero, an autoregressive models is formed [4]. In this research we are interested in AR modeling because of the following properties of an AR model:

- Biomedical signals such as speech signal have an underlying autoregressive structure.

- Any signal can be modeled as an AR process provided that an appropriate model order is selected.
- There are efficient algorithms available to compute the solution of a linear system of equations to estimate parameters of the model.

2.1.1 AR Modeling

The autoregressive (AR) spectral estimation method has a better predictive power and result in higher resolution spectral estimation in comparison with the Fast Fourier Transform (FFT) method. The autoregressive (AR) modeling is also computationally efficient and requires less memory for implementation. These advantages have convinced researchers to use parametric spectral analysis methods in biomedical signal processing [4], [5], [6].

AR modeling can be classified as a time-series analysis which is based on modeling a signal as a linear combination of its past values and the present input to a system whose output is the given signal.

(2.3)

The AR transfer function can be found applying the z-transform to the above equation as follow:

(2.4)

In many biomedical signals, a hypothetical input is considered since the input is actually unknown. Hence, a linear combination of past values of the output can be used to predict the approximate value of current output . Therefore the equation 2.3 may be written for approximate predicted output as:

(2.5)

The error in the predicted value can be determined as:

(2.6)

Figure 2.1 indicates the signal flow diagram of the AR model.

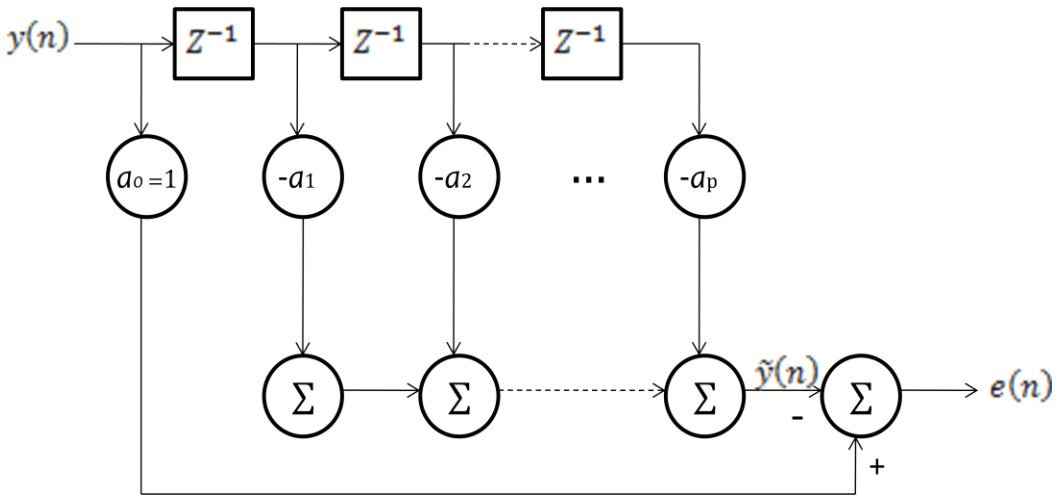


Figure 2.1: Signal-flow diagram of the AR model

There are various techniques that can be used to compute model coefficients (or poles), directly or iteratively. Iterative methods are more computationally intensive to achieve a desired degree of convergence with respect to the direct methods [14]. The most commonly used approaches for direct estimation of model parameters are: the autocorrelation method, the covariance method, the square-root (Cholesky decomposition) method, and the Burg method. In these methods the objection is to solve the normal equations, a set of equations for the predictor coefficients , 1 . Autocorrelation or covariance methods computationally intensive and require large storage. Square-root (Cholesky decomposition) method has less

computation compared with the two previous methods; however the more computationally efficient method with less storage requirement can be achieved by using Levinson-Durbin algorithm. This recursive method provides solution of the set of normal equations [14]. The Burg algorithm is a popular method to process biomedical signals in which the AR parameters satisfy the Levinson-Durbin recursion. Using Burg algorithm the estimate of autoregressive (AR) model can be computed to fit the model to the input data. This algorithm involves by minimizing least squares of the forward and backward prediction errors.

Obtaining the solution of AR parameters of order M using Burg algorithm, it is possible to add one more stage without affecting the earlier computations for previous stages. This is an advantage of Burg method over the Levinson-Durbin algorithm which makes it more suitable for hardware implementation using Field Programmable Logic Arrays (FPGA) or Very Large Scale Integrated Circuit (VLSI) design [14].

2.1.2 Burg Algorithm

The Burg algorithm is based on minimizing least squares of the forward and backward prediction errors. The cost function is given as [4]

(2.7)

where is the order of filter, and are forward and backward predication errors respectively, and N is the length of the input data. The forward and backward prediction error updates are recursively using the lattice structure as follow:

(2.8)

) (2.9)

where is the reflection coefficient of Burg algorithm that can be calculated as:

(2.10)

Figure 2.2 indicates the lattice structure of the recursion equations for one stage of Burg algorithm to update forward and backward prediction errors.

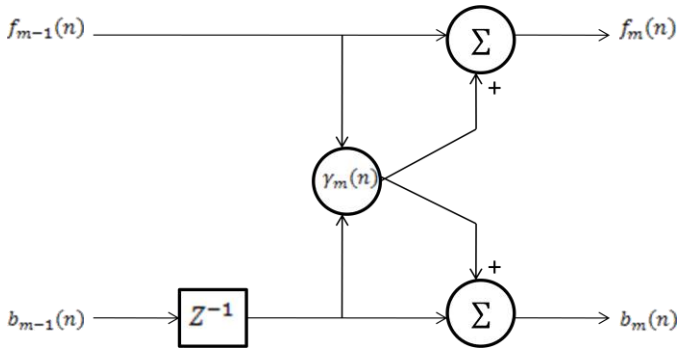


Figure 2.2: The lattice structure of the recursion equations for forward and backward prediction errors

Knowing the reflection coefficient, the AR model parameters for iteration can be computed by following relationship:

(2.11)

The value for the initial AR parameter is always equal to 1, and the parameter can be calculated using following matrix operation:

(2.12)

The least square lattice structure provides stability for real-time estimation of the AR coefficients. Hence a recursive AR model allows for estimation of the AR model coefficients in real time which allows predicting ahead. Burg algorithm is not an adaptive algorithm, since it does not update parameters on sample-by-sample basis (but on block-by-block basis). Equation (2.10) suggests that with increase in system order, the length of data required for calculating reflection coefficient decreases and the number of calculated AR coefficients increases. Hence hardware implementation of the Burg algorithm cannot be accomplished by a structural design approach since the architectures of adjacent stages are not exactly the same. In order to overcome this problem, a sequential data-flow design is suggested in this work as it will be discussed in next chapter.

2.2 Threshold Voltage Effects

AR modeling with Burg algorithm helps to reduce dynamic power consumption. However, as it will be shown in chapter 3 that dynamic power consumption only accounts for about 20% of the total power consumption for this design. Hence reducing static power dissipation can significantly increase the battery life of the device. To accomplish this task the effect of threshold voltage on power-delay product (PDP) of a device is investigated in this research. The threshold voltage is the gate voltage at which an inversion layer is formed in a MOS transistor. Threshold voltage is not constant; it increases with the source voltage and

channel width, and decreases with the body voltage and the drain voltage. Threshold voltage is also depends on the type and thickness of the oxide used in the process, and is directly related to the temperature of a CMOS device.

2.2.1 Body Effect

The transistor is a four-terminal device with gate, source, drain, and body as an implicit terminal. Applying a voltage between the source and body increases the amount of charge required to invert the channel and hence increases the threshold voltage . Equation 2.13 shows how threshold voltage can be modeled:

) (2.13)

where is the threshold voltage when source and body have the same voltage. is the surface potential that can be calculated as it is shown in equation 2.14. is the body effect coefficient that depends on doping level , and oxide capacitance as it is shown in equation 2.15:

(2.14)

(2.15)

For a small voltage applied to the source and body, the relationship between the threshold voltage and can be simplified to (2.16):

(2.16)

where depends on the body effect coefficient and the surface potential.

2.2.2 Subthreshold Leakage

The transistors leak a small amount of current even when the gate voltage is less than the threshold voltage. This leakage is significant for processes less than 90 nm. In this subthreshold region of operation, current drops off exponentially as gate voltage falls below (weak inversion). This region can be used for low power circuit design at the cost of reduced performance [3]. Subthreshold regime can be used for low power circuits at the cost of performance reduction as it will be discussed later in this chapter.

2.2.3 Subthreshold Circuit Design

In subthreshold region, total power consumption reduces with decreasing supply voltage. Delay, on the other hand, increases. Consequently, the power-delay product (PDP) should be used to find the minimum energy operating point for the supply voltage. According to [3], the minimum energy operating point typically occurs at a supply voltage close to 300-500 mV and to reduce both switch capacitance and leakage, all transistors should be initially designed as minimum width. Once the minimum energy operating point is determined, transistor sizing should then be used to trade power to improve performance. In this work, we observe that in the design of FPGA routing tracks, with minimum width transistors, the delay of a track in the subthreshold region is mainly caused by the wiring capacitance and the slow response of the multistage buffers. The issue is examined in detail in chapter 4.

2.3 Related Work

Subthreshold circuit design for low power application has been investigated previously. It has been shown in [7] that operating in the subthreshold region minimizes energy per operation and that the minimum energy-delay product occurs at supply voltage of 3Vt. In [8] an analysis of operating both CMOS and pseudo NMOS logic families in the subthreshold region and a comparison with normal operation in strong inversion region is presented. The results of this research reveal operating in subthreshold region reduces the energy per switching of an 8X8 carry save array multiplier by a factor of two. In [9] different leakage components have been modeled to estimate and reduce the leakage power for low-power applications. In [27] the leakage power consumption of the components which make up the FPGA fabric is studied. They found that about 55% of the total leakage power is consumed by the interconnect multiplexes, 26% by the LUTs and 19% by the flip-flops and other components with the assumption that the leakage of SRAM cells are optimized by using high threshold voltage, and thick oxide transistors. It has been shown in [28] that the used multiplexers in the interconnect fabric are less than 5% of the total multiplexers and the unused multiplexers cause a significant portion of the FPGA leakage power.

There also have been a number of studies to reduce the power consumption of the interconnect fabric on the FPGAs. Some of these studies are focused on the subthreshold design and body biasing techniques. In [10] a subthreshold FPGA that uses a low-swing dual-VDD global interconnect fabric was implemented on a 90nm device. This implementation resulted in 4.7X energy reduction and 14X improvement in speed as compared to a subthreshold FPGA using conventional interconnect. The energy reduction for this implementation was 22X as compared to an FPGA with transistors operating in saturation region. In [11] a stack of half-

width transistors in conjunction with adaptive body biasing has been proposed to reduce the leakage power for a switch block and a switch matrix on an FPGA. In [12] body biasing technique is used at a coarse grained architecture level to reduce leakage power and a clock skew scheduling scheme offered to improve performance. It required 3.35% area overhead to implement clock skew control blocks which are placed at every configurable logic block (CLB). None of the above studies, however, has investigated the performance requirement of biomedical applications on FPGAs.

Chapter 3

FPGA Implementation of AR Burg Algorithm

Biomedical signal processing involves the collection and analysis of signals generated by human and other living organisms for medical diagnosis purposes. In portable applications, these signals often need to be processed in real time. For example, Autoregressive (AR) modeling [4]-[6] is a widely used feature extraction method that is used in biomedical applications to extract key diagnostic features from a range of biomedical signals such as knee joint vibroarthrographic (VAG) signals [5], pathological voice signals [13] and polysomnographic sleep data.

Extracting key features from signals also reduces the amount of memory that is required to store these signals. For portable applications, AR modeling performed in real time can be used to reduce their memory requirements. In particular, instead of directly storing VAG signals from the output of an Analog to Digital Converter (ADC), AR modeling can be used to extracts key features, called Autoregressive (AR) parameters, from the digitized version of these signals and stores only the AR parameters [5], [13] in memory for later diagnostic use [5].

In particular, with 16 KHz sampling rate, 16 gigabits of memory is required to directly store an uncompressed stream of biomedical signals (for an ADC with 12-bit output) for 24 hours. Using a 32-stage AR model, the required memory can be reduced by a factor of 184 to around 90 megabits. This reduction in memory can be extremely important in the design of portable medical devices for monitoring patient activities for an extended period of time.

The AR Burg algorithm has been previously implemented by researchers. Majority of researchers have used a microprocessor base design and some have attempt to use FPGAs or design a custom Integrated Circuits (ASIC), with their attentions mainly focused on improving performance. However, not much attention has been given to low power design. In this thesis a parametric architecture has been designed, suitable to be implemented on an FPGA or an ASIC chip, with the focus on reducing the total power consumption of the device. Although the low frequency nature of biomedical signals (kilohertz range) makes microprocessors a cost-efficient choice to implement AR Burg algorithm, the microprocessors in general provide less control over the power consumption as compare to FPGAs or ASICs. One significant weakness in processor-base design is that it requires an external (off-chip) memory device. Adding an extra device obviously increases the total power consumption of the system. This problem is diminished significantly for FPGAs and ASICs, because of their capability to facilitate an on-chip memory of relatively decent size.

3.1 Implementation of Burg Algorithm

As it was mentioned in previous chapter, the Burg algorithm is based on minimizing least squares of the forward and backward prediction errors. These steps are computed using equations 2.8 to 2.10. Using these equations the reflection coefficient can be calculated and used to first

update the forward and backward prediction errors and then to find the AR Burg model parameters with the aid of equation 2.11. The recursive nature of AR Burg implies that there should be feed-backs in data path. The parameters should be stored to be used later to find the next set of parameters, hence the design requires a memory management unit. It is also required to allocated input and output buffers to read in and send out the data. These buffers can also be used to adjust the data format as well as data rate to match the format and frequency of the units it interfaces with. The Burg algorithm has been previously implemented using MathWorks® FPGA Design Solutions. In the next section this methodology will be discussed.

3.2 Previous Implementation of Burg Algorithm

The Burg algorithm has been previously implemented using Simulink® with Xilinx® System Generator. This is a block level design methodology that generates a VHDL code which produces a structural RTL level design. Figure 3.1 indicates the block diagram of this design [25].

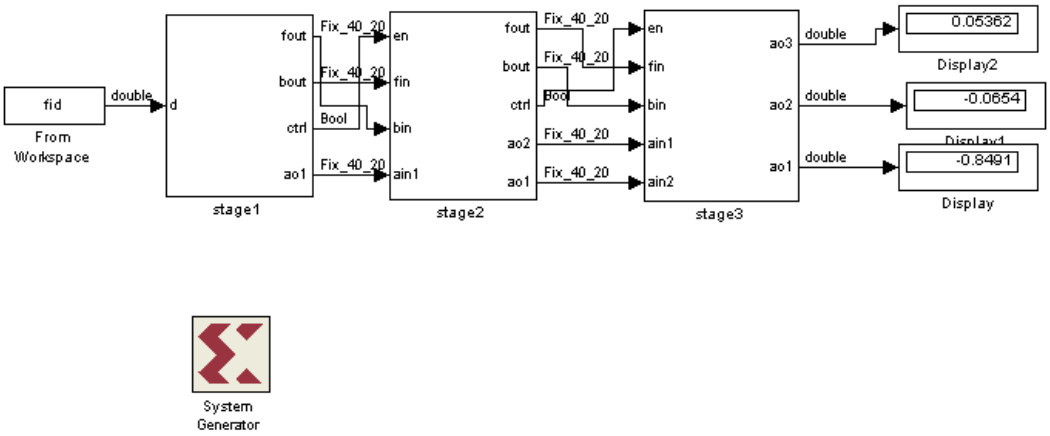


Figure 3.1: Block diagram of previous implementation for 3 stages using MathWorks® FPGA Design Solutions

As it can be seen from the design block diagram only 3 stages has been implemented which can only generate 8 AR model coefficients. In this method each stage has one additional 40-bit data bus with respect to previous stage. Figure 3.1 shows the block diagram of previous implementation for 3 stages using MathWorks® FPGA Design Solutions If this method was used to design an AR model of order 32 or higher, the last stage would have 32 or more 40-bit data bus. Hence the number of required adders and multipliers to complete the computations would increase respectively. Figure 3.2 indicates the design of the first stage of the design.

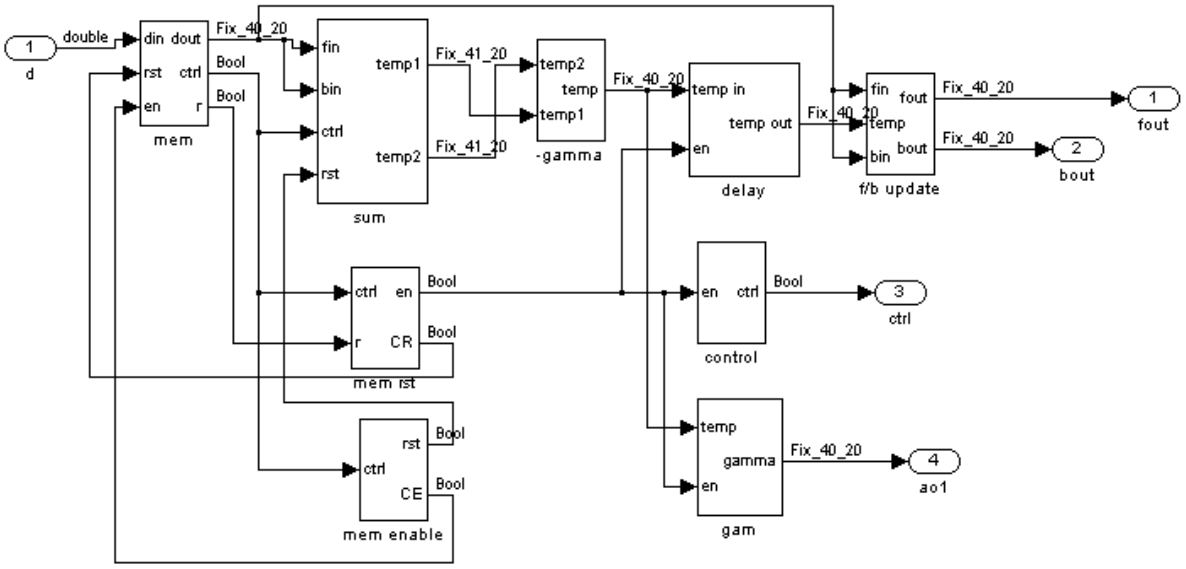


Figure 3.2: Block diagram of the first stage for Burg algorithm using MathWorks® FPGA Design Solutions

Figure 3.2 indicates that this methodology requires dedicated adders and multipliers for each stage and the complexity of the design increase for each additional stage. This design is not parameterized and does not provide flexibility to be able to change the model order or data bus width. Hence it should be designed separately for every desired model order. The power consumption of the device increases as the model order increases. The significance of the limitations of this methodology was the motivation to design an appropriate architecture to

overcome these limitations. More importantly, the design should provide variables to find ways to reduce the power consumption of the device. In the next section the architecture designed in this project is presented.

3.3 A Parameterized Architecture for Implementation of Burg Algorithm

Automated RTL code generators aid to reduce the time to market of a design, however they do not provide an optimized design in terms of area, performance, and power consumption. The alternative is to design a custom architecture. Figure 3.3 is the block diagram of the implementation of the Burg algorithm introduced in this thesis.

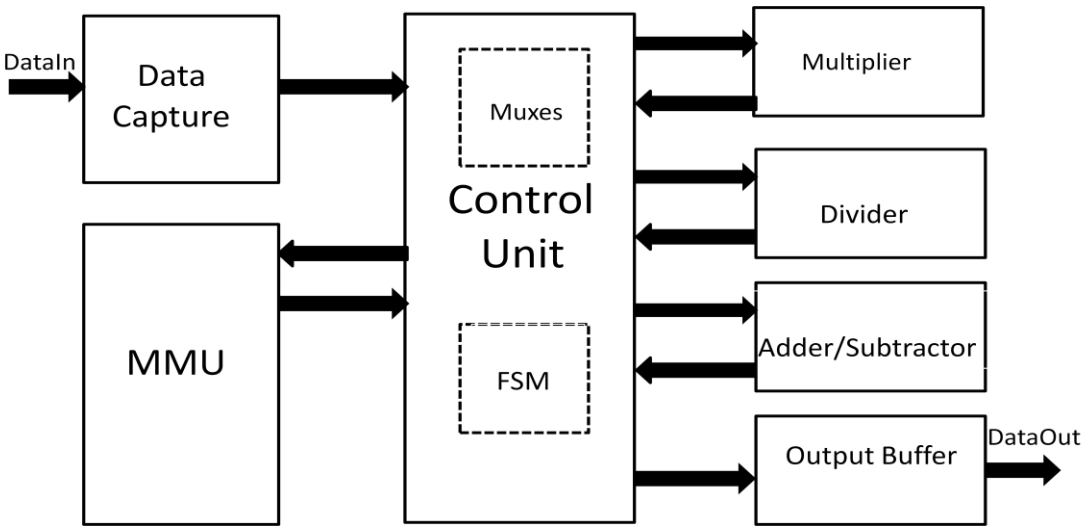


Figure 3.3: Block diagram for the implementation of the Burg algorithm

This architecture offers number of advantages. It is programmable; hence the data bus width and the AR model order can be changed as needed without any limitation. By allocating appropriate number of functional units, it can be configured as a pipelined architecture (temporal parallelism) to minimize the design area, or as a full parallel architecture (spatial parallelism) to improve performance at the cost of area, or a combination of both. The design also maximizes the reuse

of the resources. Finally, it can be mapped to any FPGA or ASIC. Thus this architecture (as it was planned) provides required variables to investigate and determine ways to improve energy efficiency of the design without having any impact on the performance.

Figure 3.4 shows the schematic symbol of the top-level module which includes the IOs.

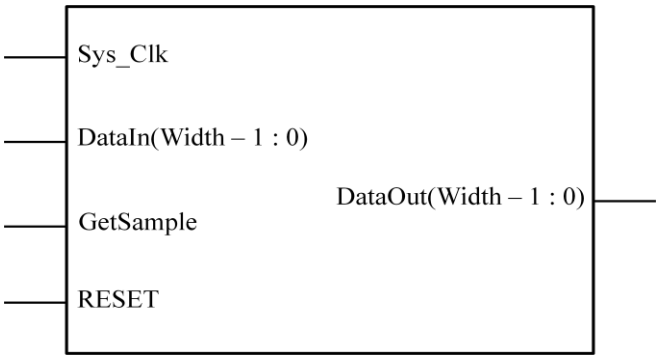


Figure 3.4: Schematic symbol of the top-level module

The top-level module has a simple interface which requires a triggering signal (GetSample) only to start the operation. A typical timing diagram for this design is shown in Figure 3.5.

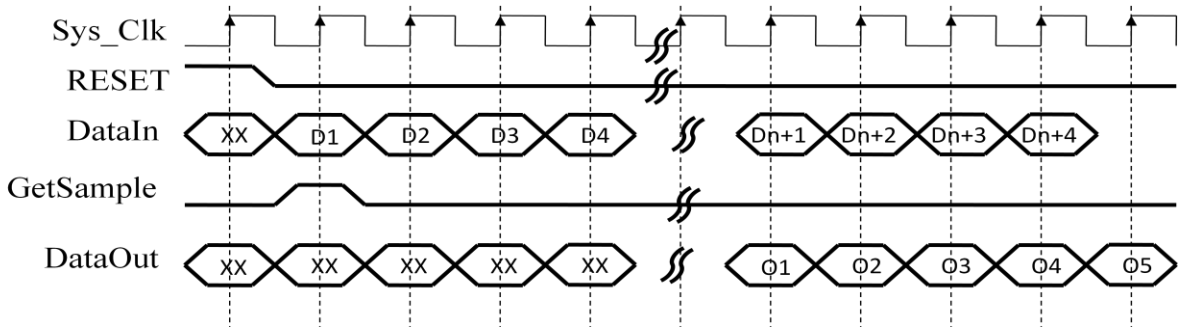


Figure 3.5: Timing diagram of the design

3.3.1 Data Capture Module

This module was designed to read in the data and latch the input data to produce *X(n)* as well as delayed version of the input data *X(n-1)* which are required for the computation of a new set of AR coefficients. For non-stationary biosignals the design assumes that data is already

organized into frames by a Recursive Least Square Lattice (RLSL) filter. This stage of filtering is required prior to computation of the AR coefficients to convert non-stationary biosignals into stationary frames. For stationary biosignals the sampled data can be directly sent without passing through RLSL filter.

3.3.2 Memory Management Unit (MMU)

Each stage of the computation algorithm requires storage of the interim results. The final AR coefficients are also needed to be stored in an on-chip memory before an external device (or module) could read them out. The size of the memory should be adjusted depending on data bus width and model order which in turn defines the number of computational stages. In this design each memory unit is configure as dual port ram using block Rams of Virtex 5. In Virtex 5 FPGAs each block Ram can store up to 32K bits of data. Figure 3.6 depicts the data flow of a dual port ram with 32K bits of storage capacity [26].

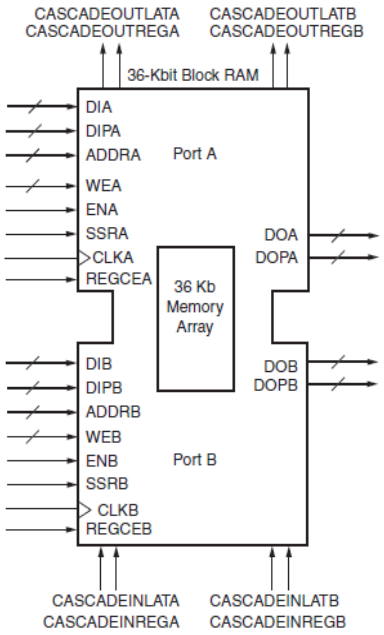


Figure 3.6: Data flow of a Dual Port Ram

The timing diagram of the dual port rams are shown in figure 3.7 [26].

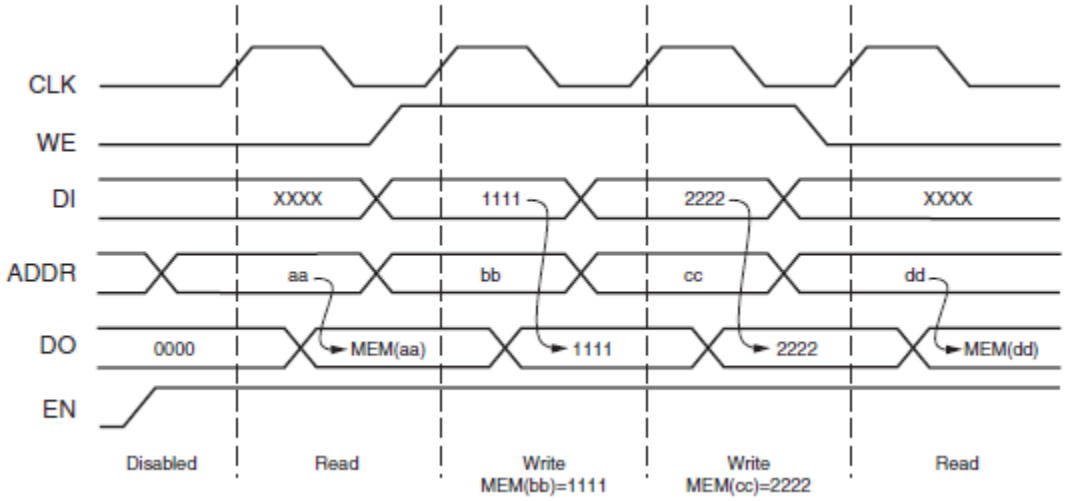


Figure 3.7: Block Ram timing diagram

3.3.3 Functional Units (FUs)

Equations 2.8 to 2.11 suggest that this implementation requires a combination of adders (or subtarctors), multipliers, and dividers. For a pipelined architecture at least one adder, one multiplier and one divider should be allocated. These functional units are designed to perform either fixed point or floating point operations. For spatial parallelism the number of functional units increases as the order of the AR model increases. Increasing the number of functional units results in increase in power composition of the device. Hence in this research more attention is given to a combination of both temporal and spatial parallelism. The objectives of this design could be achieved by allocating three multipliers, three adders, and one divider. In fact the low frequency nature of biosignals allows minimizing number of functional units without any performance penalty.

3.3.4 Output Buffer

This module was designed to send out the data and set the appropriate flag to indicate to an external device (or module) that a new set of AR coefficients is available. The size of this buffer also depends on the order of the AR model. The output clock can be adjusted to match the external device (or module) when a different frequency boundary is required.

3.3.5 Control Unit

This module generates all control signals required by other modules and controls data traffic between them. It includes a Final State Machine (FSM) to generate the control signals as well as multiplexors to select the appropriate data in or out of modules. The FSM has two main loops to control functional units, data path, and memory units to implement equations 2.8 to 2.11. The two loops are quite similar except that the first loop occurs only once per frame, and the second loop may repeat as many times as the order of AR model defines. Figure 3.4 shows this loop.

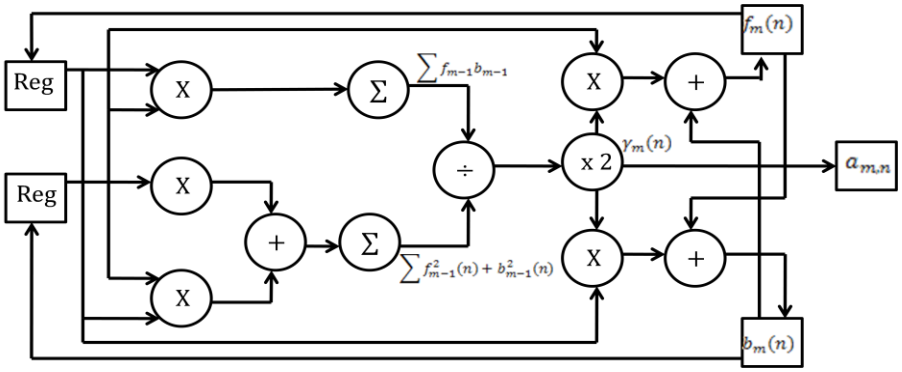


Figure 3.8: The block diagram of the AR coefficients computation loop

Figure 3.9 is the flow chart of the control unit which shows the sequence of activities based on conditions occurs in the state machine.

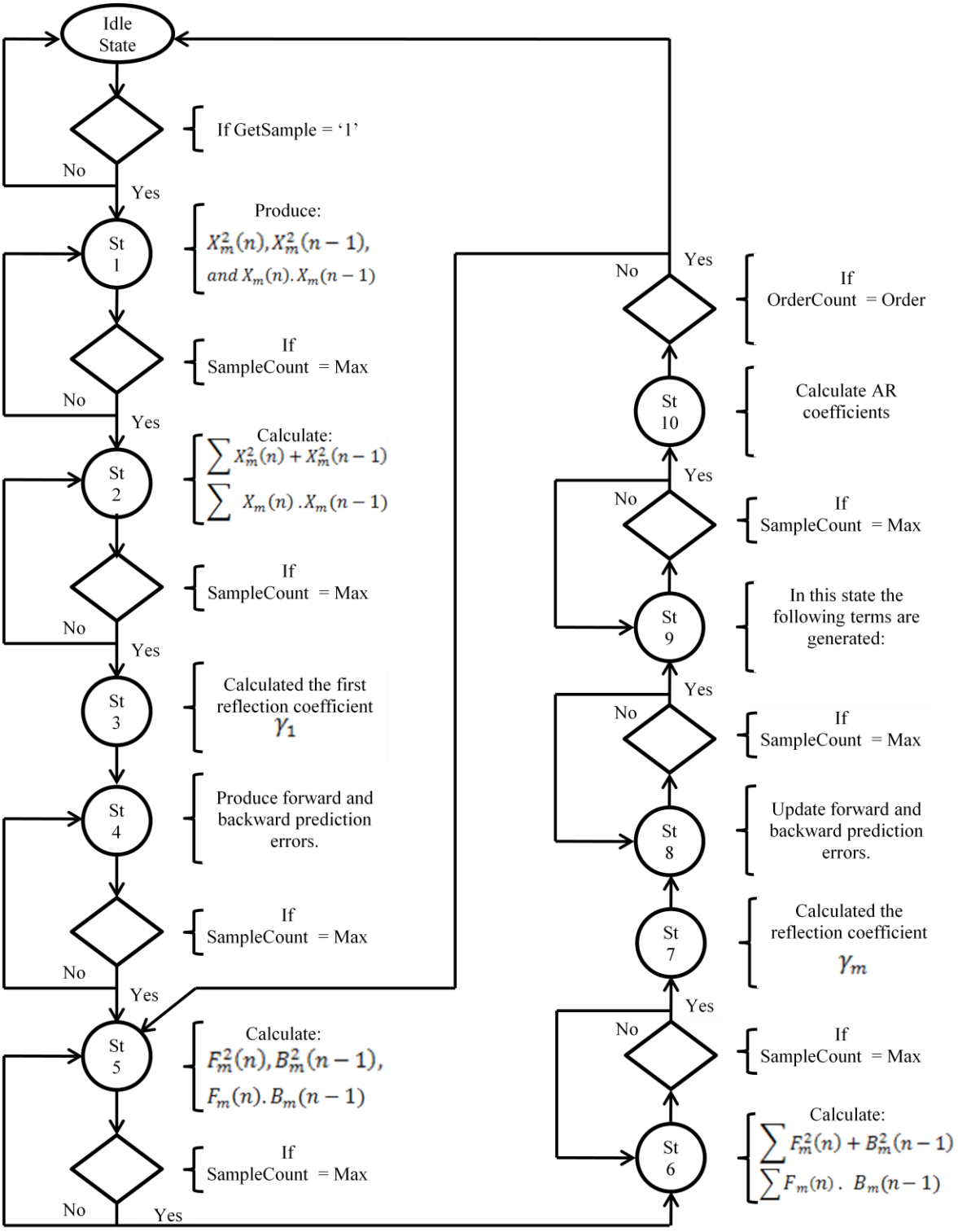


Figure 3.9: Flow chart of the control unit

The first iteration takes the sampled data and produces the first set of forward and backward prediction errors as well as the first reflection coefficient. The loop from state 5 to state 10 consumes previously generated forward and backward prediction errors and the reflection coefficient computed in first iteration to produce AR coefficients and update forward and backward prediction errors as well as the reflection coefficient for next iteration. This loop stops when it has completed the required cycles set by model order. This loop has a significant impact on area reduction as compared with the structural design offered by automated software. The area allocated for the second loop and for functional units remains fixed, hence increase in model order does not cause an increase in resources. Therefore the power consumption remains relatively fixed.

3.4 Results

A comparison between previous implementation and the method suggested in this project reveals a significant improvement in terms of area and performance. Improvements also made in terms of flexibility and functionality of the design. In order to make an accurate comparison this design was mapped into the same FPGA (XC2VP1006FF1704) used in previous implementation. Previous implantation requires a 20 MHz operating clock frequency to generate coefficient for an AR model of order 3. It will be shown that this design at maximum requires a 2 MHz operating frequency to generate coefficient for an AR model of order 32. The design offers flexibility to reduce the operating frequency for lower model orders. Table 3.1 indicates a comparison of resource utilization between the two methods.

Resources Vs. Implementation Method	AR Order	Flip Flops	LUTs	Occupied Slices	Bounded IOBs	Block RAMs	18x18 Multipliers
Previous Design using Simulink-to-FPGA	**3**	**18%**	**20%**	**25%**	**11%**	**20%**	**32%**
Current Design	**3**	**6%**	**10%**	**12%**	**11%**	**17%**	**3%**
Current Design	**32**	**6%**	**10%**	**12%**	**11%**	**17%**	**3%**

Table 3.1: A comparison of resource utilization between two implementation methods: The method using Simulink-to-FPGA tool and the one suggested in this thesis

It can be seen that the implementation method suggested here improves resource utilization from 15% to 90%. The results are specifically significant in terms of using multipliers because in this project the reusability of the resources has been increased. It should be also noted that the data for higher order implementation of the method using Simulink® with Xilinx® System Generator is not available; however the method suggested here requires the same resources for any model order.

According to (3.1) dynamic power consumption of an FPGA can be limited by reducing the operating frequency:

(3.1)

The variation of supply voltage and capacitive load that requires architectural change will be discussed in next chapter.

In order to find minimum operating frequency, cycle count for various numbers of samples was extracted and collected in Table 3.2.

Number of samples	20	100	200	400	2000	8000
Number of Cycles for AR8	554	2474	4874	9674	48074	192074
Number of Cycles for AR16	1114	5014	9814	19414	98014	392014
Number of Cycles for AR32	2214	10014	19614	38814	196014	760414

Table 3.2: Number of cycles required per frame for various model orders

It can be seen from table 3.2 that a fixed 14 cycle overhead is required for processing of each frame. The number cycles increase linearly with an increase in AR model order or an increase in the number of samples per frame. Figure 3.5 is a graphical representation of the data provide in table 3.2:

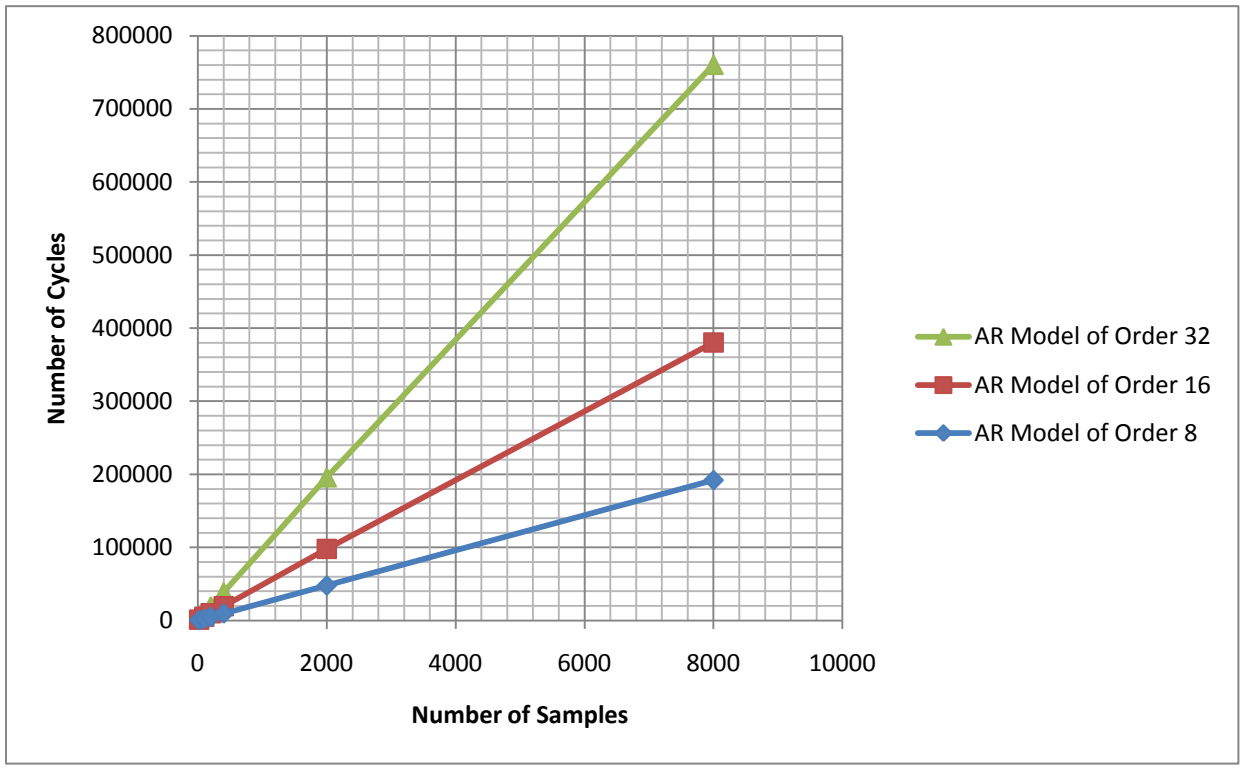


Figure 3.10: Number of cycles vs. number of samples for various AR model orders

Note that biomedical signals are analog signals of relatively low frequency that require digitization. The bandwidth of biomedical signals is limited to a few tens to a few thousand Hertz. According to Nyquist Sampling theorem a signal must be sampled at a rate at least twice the rate of the highest-frequency component present in the signal. Hence the sampling rates for the digital processing of biomedical signals range from 100 Hz to 20 kHz [4].

The actual data processed for AR modeling in this design is organized into frames of maximum 8000 samples. For real time operation, a maximum of 8000 samples must first be acquired and stored in memory. These samples are then processed to generate the AR coefficients while a new

frame of data is being sampled by an Analog to Digital Converter. The maximum operating frequency required to process the data in real-time can therefore be calculated as:

(3.2)

(3.3)

Referring to Table 3.2, for an AR model with order of 32 the design requires 760414 cycles. The above equation shows that for a maximum sampling frequency of 20 KHz (worst-case design), to process 8000 samples per frame an operating frequency of 2MHz is required. Figure 3.6 shows that the operating frequency and sampling frequency are linearly related. This figure can be used to predict the operating frequency with respect to sampling frequency used for any type of biosignals.

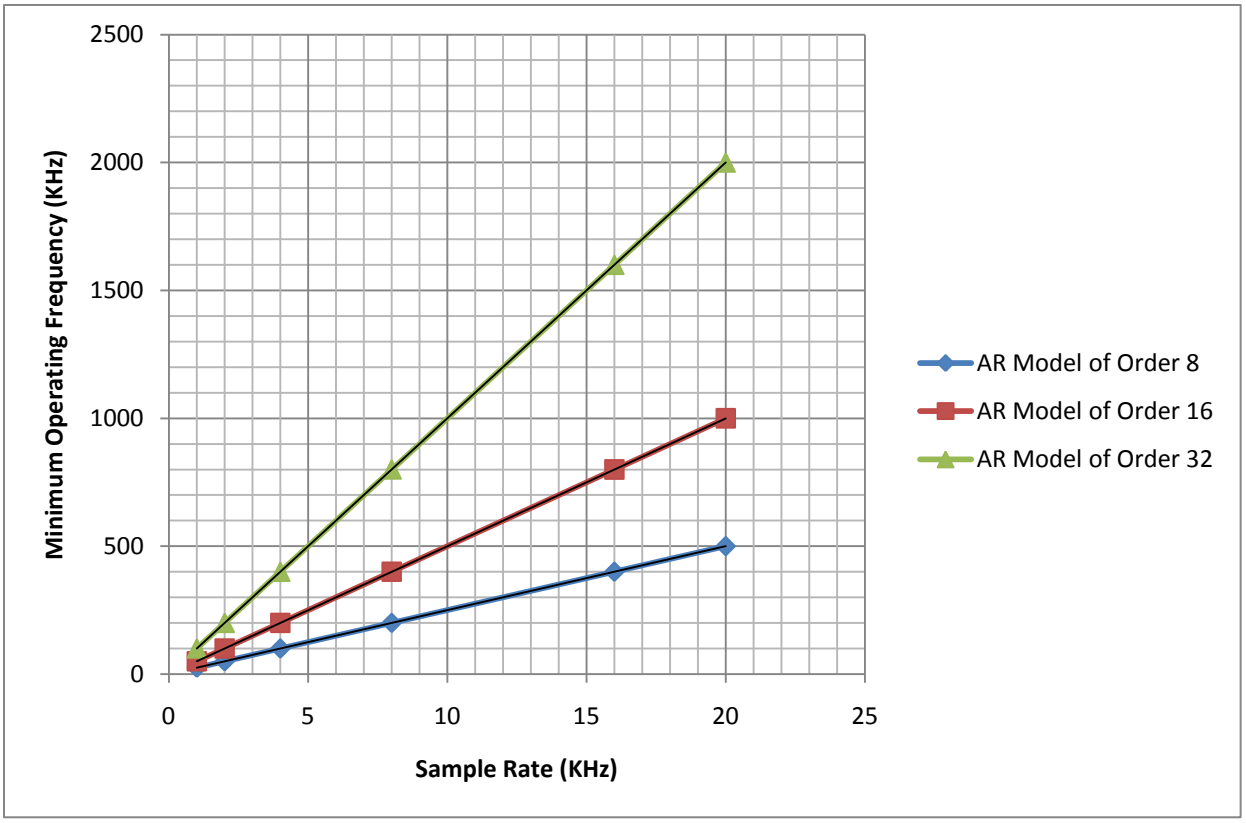


Figure 3.11: Graph of Sample Rate vs. Minimum Operating Frequency

The operating frequency can be reduced to half by doubling the processing module. Since it was shown that this design requires relatively small area of an FPGA, it is feasible to place two units to reduce the frequency of operation. According to equation 3.1 adding additional units will not increase dynamic power consumption because increase in overall switching capacitors due to increase in area will be compensated by reduction in frequency. This leads to next issue which is measuring the total power consumption of the design.

As it was mentioned before, this design was implemented on a device of Virtex II Pro family of Xilinx FPGAs (XC2VP1006FF1704) to compare this design with previous implementation method. In order to study the total power consumption of the design a more advanced device from Virtex 5 family (XC5VLX110-3FF676C) was selected. Two variation of the design was placed and routed on this device in separate steps to investigate the power consumption for a 32-bit single precision floating point implementation versus a 64-bit double precision floating point implementation. I used XPower Estimator tool included in ISE Design Suite provided by Xilinx to measure the power consumption of the design. Figure 3.7 shows the total power consumption for the 32-bit floating point and the 64-bit floating point implementations over a range of operating frequencies from 0 Hz to 10MHz. As shown, for both implementations, reducing operating frequency proportionally reduces power consumption, which is the result of reduction in dynamic power dissipation as dynamic power scales proportionally with clock frequency [3]. Figure 3.7 also shows that both implementations still consume a significant amount of static power even when operating at very low frequencies.

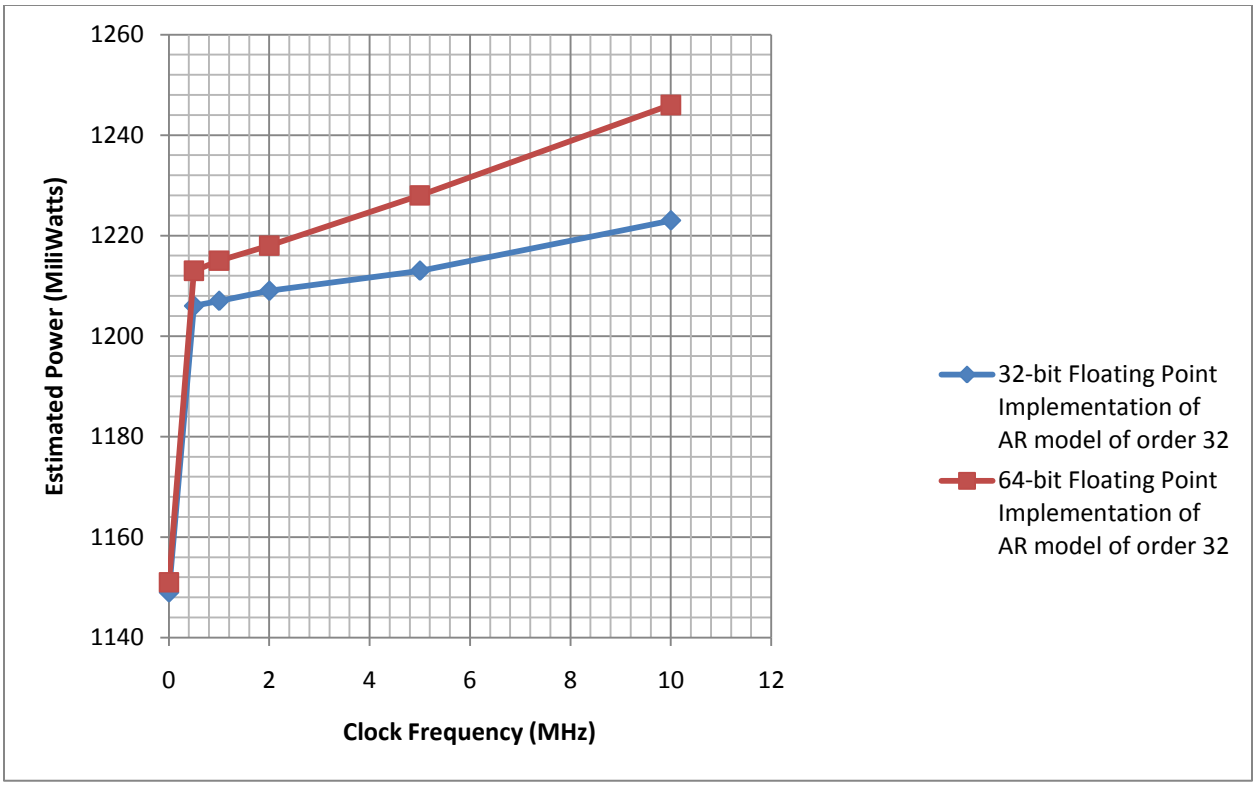


Figure 3.12: Comparison of power estimation for 32-bit and 64-bit floating point implementations

The graphs for both implementations linearly increase for operating frequencies over 1 MHz, however 64-bit floating point implementation has higher slope. Nevertheless what is significant in this figure is the large amount of static power consumption as compared to dynamic power consumptions. This experiment indicates that although it is possible to reduce the amount of dynamic power consumption of an FPGA by applying low power design techniques, yet a significant amount of static power is consumed by the device. This is true because most FPGAs available in the market are SRAM based, and the SRAM cells require power to maintain their values whether they are utilized in the design or not. All other transistors used in routing track architecture and logic blocks also introduce leakages power. One viable remedy for this problem is reducing the power supply as it has been practiced in at least past two decades. Latest technologies are now utilizing 0.9 volts supply voltage. However, the decrease in power supply

is limited by the threshold voltage of a transistor (typically 0.4 volts). Hence further power reduction requires investigation in subthreshold regime. In next chapter a study of subthreshold design and power-delay product for an FPGA is presented.

Chapter 4

Subthreshold Circuit Design

The main focus in this research is to investigate ways to reduce total power consumption for implementation of biomedical applications. The results of the study of power consumption on our FPGA implementation in previous chapter indicated that a significant amount of power consumed in an SRAM based FPGA is static power. Reducing static power dissipation can significantly improve battery life of a device. To accomplish this task, the effect of threshold voltage on the power consumption of FPGAs is investigated in this work. Threshold voltage is the gate voltage at which an inversion layer is formed in a MOS transistor. Threshold voltage is not constant; it increases with the source voltage and channel width, and decreases with the body voltage and the drain voltage. Threshold voltage also depends on the type and thickness of the oxide used in the process, and is directly related to the temperature of a CMOS device [3].

In order to conduct this research a Simulation Program with Integrated Circuit Emphasis (SPICE) model was designed and simulated. The results of this simulation are discussed in this chapter.

4.1 FPGA Architecture

As in most commercially available SRAM based FPGAs, we assume an Island-style architecture consist of arrays of logic blocks connected together and separated by horizontal and vertical programmable routing channels. Figure 4.1.depicts an island-style FPGA in which logic block are surrounded by routing channels.

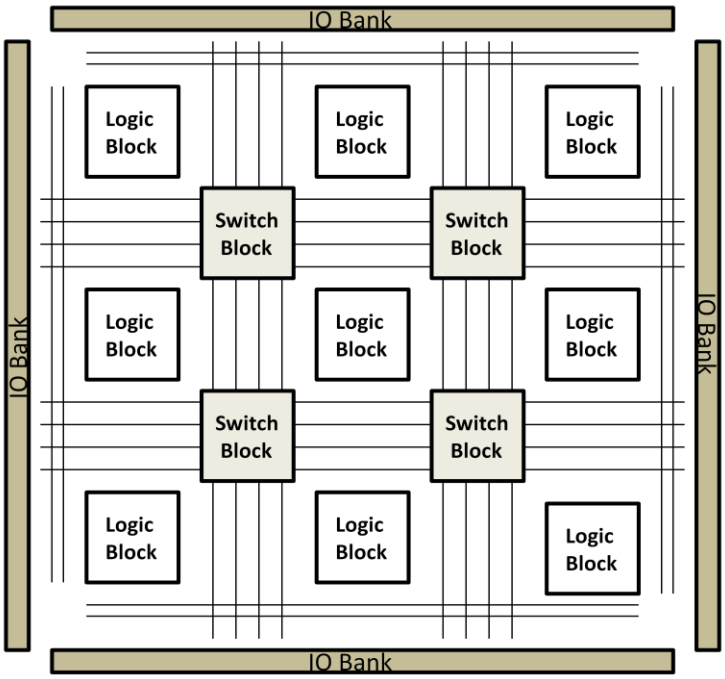


Figure 4.1: Island-Style FPGA Architecture

The routing channels consist of pre-fabricated wiring segments. The horizontal and vertical channels are connected through programmable switches which are called switch blocks [16]. There are multiple wire segments between the switch blocks depends on the architecture of particular FPGA. In this architecture, most of the area of an FPGA is consumed by the routing channels which are also responsible for most of the circuit delay. To investigate power-delay product of an FPGA, it is required to make an accurate simulation model of the routing channels. In next section the architecture of the routing channel will be discussed in details.

4.2 Routing Channel and Delay Path Model

Figure 4.2 depicts the routing channel structure adapted in this research. Each routing track is assumed to be connected to a set of multiplexor 4-to-1 in series with a multistage buffer at both ends. For the conventional routing architecture, for each logic block one isolation buffer is placed to electrically isolate the routing tracks from the input connections of a logic block [16]. Each logic block was modeled as a buffer driving a capacitive load. Wiring capacitors were also considered and placed.

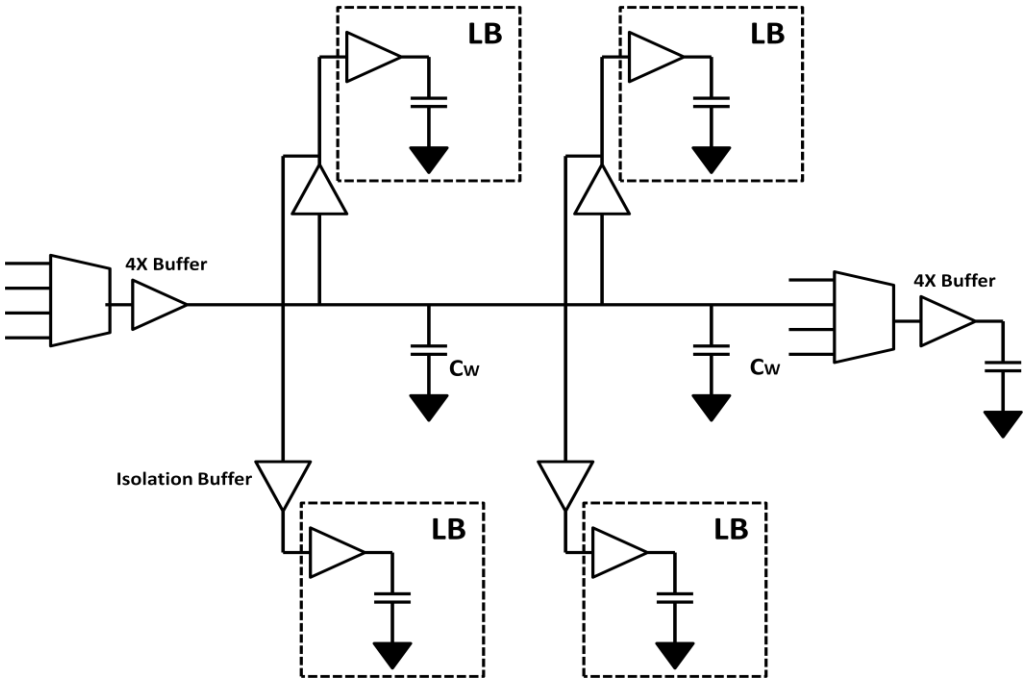


Figure 4.2: Routing track and delay path model

4.2.1 Multiplexers

Multiplexers are implemented in form of a binary tree of pass transistors with SRAM cells controlling the selection of the input data. All transistors are of minimum width size. The schematic of a 4-to-1 multiplexer is shown in Figure 4.3 [16].

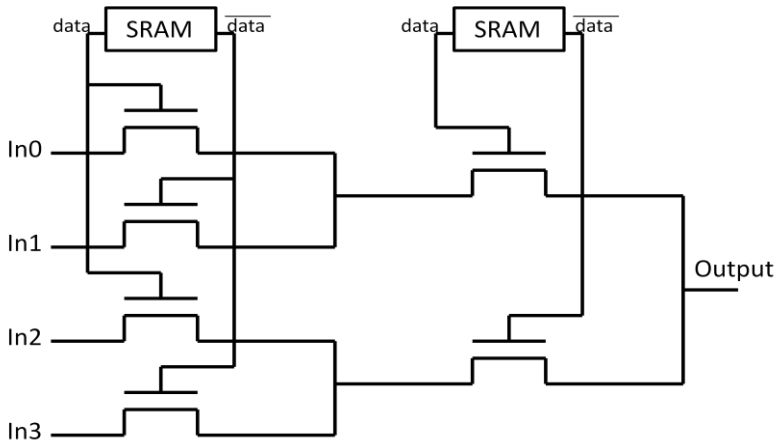


Figure 4.3: A 4-to-1 multiplexer implemented with pass transistors

4.2.2 Buffers

Multistage buffers and isolation buffers are widely used in FPGAs to drive a larger load and to electrically isolate the routing tracks from the input connections of logic blocks. Isolation buffers are simple made of two COMS inverter with minimum-width size transistors connected in series. Multistage buffers however require higher drive strengths that can be accomplish by chaining buffers of gradually increasing size. The multi stage buffers used here are 4X of minimum strength. Figure 4.4 depicts schematic diagram of such buffers.

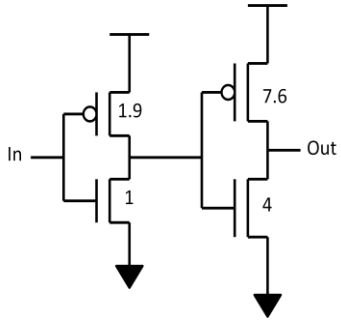
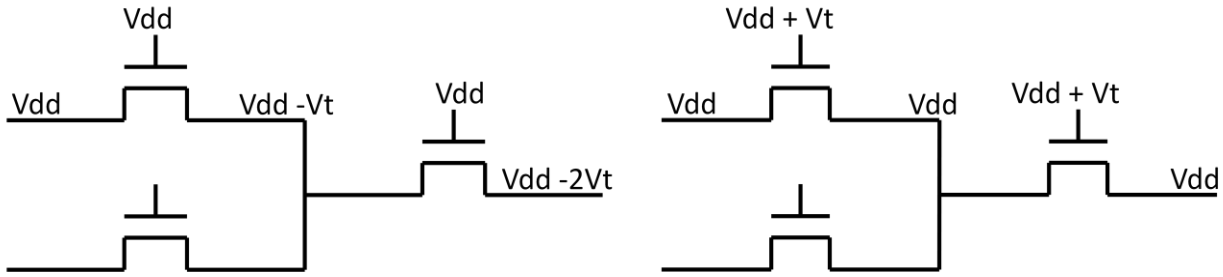


Figure 4.4: A multistage buffer with 4X drive strength

4.2.3 Gate Boosting

An nMOS pass transistor degrades a logic high value by a threshold voltage [16]. The structure of multiplexers used in this work is organized in a binary tree of pass transistors. The logic high value appears at the output of a multiplexer can be significantly degraded (depends on the number of stages of the multiplexer). We boost the gate voltage of a pass transistor by a value equal to threshold voltage. The boost in voltage is applied to pass transistors at each stage of the multiplexer to compensate the voltage drop caused by each pass transistor. Figure 8(a) shows for a supply voltage of 0.9 volts and a threshold voltage of 300 to 400 millivolts the output could be reduced to logic low value where a logic high value is expected. Boosting the gate voltage of the pass transistor by a threshold voltage overcomes the problem as it is shown in Figure 8(b) [16].



a) Pass transistor degrades a logic high value b) Gate boosting to solve voltage degrading of pass transistor

Figure 4.5: (a) Potential problem with pass transistor; (b) Solution for voltage degrading of pass transistor

4.2.4 Transistors and Interconnect Models

In this work, 32nm technology was used. Accurate and customizable model files for NMOS and PMOS transistors that are compatible with HSPICE circuit simulators, were taken from the Predictive Technology Model (PTM) website [17] (which is developed by the

Nanoscale Integration and Modeling (NIMO) Group at Arizona State University). This website also provides RLC values for interconnects by setting the appropriate values as it is shown in Table 4.1.

Metal Type	Cu
Width of the Trace	0.064μm
Separation Between Traces	0.064μm
Length of the Trace	30μm
Thickness of the Trace	0.14μm
Height From Ground	0.14μm
Dielectric Constant	2.2

Table 4.1: Parameters used to determine interconnect capacitance

The wiring capacitance for the routing track model was found to be 4.63fF for a wire length of 30μm that is the equivalent to the length of a side of a tile (a square area containing a logic block and the area occupied by routing tracks). The information related to area of a tile was taken from Intelligent FPGA Architecture Repository (IFAR) website [18]. A typical FPGA has 2 or 4 of such blocks hence the total wiring capacitance is adjusted accordingly.

4.3 Simulation Results

The performance and power consumption of the routing track shown in Fig. 6 were measured over a range of supply voltages. Two types of buffers are considered – the conventional and the Swapped Body Biasing (SBB) [19] buffers – for implementing the 4x buffers shown in the figure.

According to (2.16), the threshold voltage can be adjusted by applying a body bias voltage [3]. In a conventional inverter, the substrate bias voltage is set to zero for both nMOS

and pMOS transistors as it is shown in Fig. 4.6(a). Setting to zero forces the pn junction between the source and body as well as pn junction between the drain and body to be reverse biased. This causes a reduction in the leakage current which is the driving current in the subthreshold region. Hence the performance of the transistor is degraded in subthreshold operations [20], [21].

As shown in Fig. 4.6(b), the source-to-body and drain-to-body pn junctions are forward biased (by applying) in an SBB inverter in order to increase the subthreshold current. Typically, in both the subthreshold and saturation regions, the SBB inverters introduce less propagation delay than the conventional inverters.

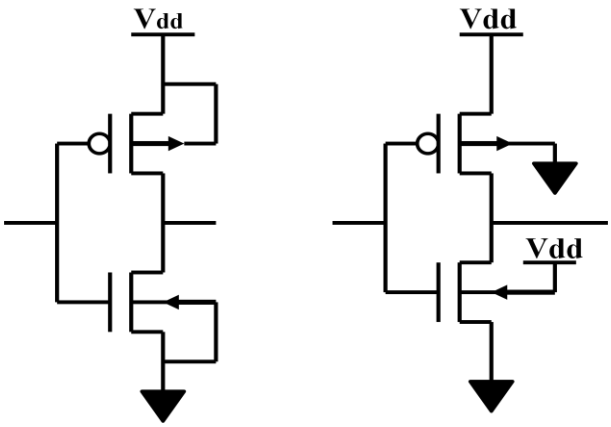


Figure 4.6: (a) A conventional inverter; (b) An inverter with swapped body biasing (SBB) voltage

Using the SBB technique, the conventional 4x buffers shown in Figure 4.2 were substituted by SBB buffers at the output of each multiplexer. Figure 4.7 depicts the structure of this buffer, which is designed by connecting an SBB inverter (to reduce the propagation delay) to a conventional inverter of size 4x (to provide the desired drive strength).

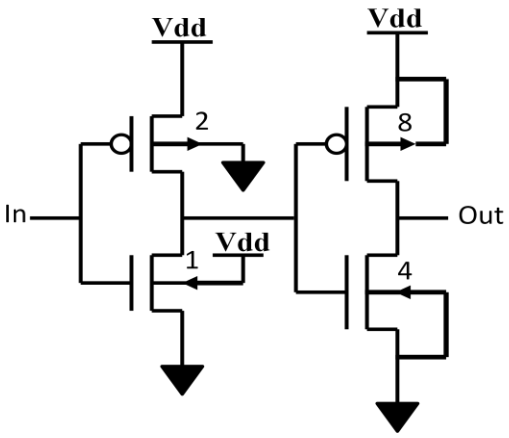


Figure 4.7: Multistage buffers with variable threshold voltage

The PDP graph of the various buffer sizes shows that the optimum size of the secondary stage inverter is four times the minimum size transistors as it is shown in Figure 4.8. Hence the secondary stage inverter of size 4X is used to build the buffers.

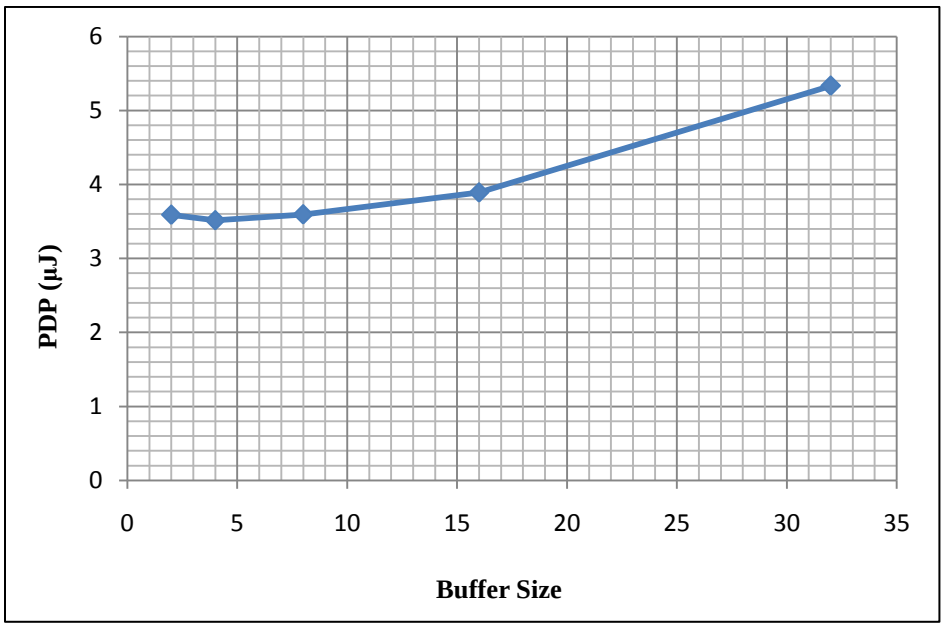


Figure 4.8: Power-Delay Products for various buffer sizes

The average power dissipation and the delay of both buffers for various supply voltages in subthreshold region, near subthreshold region, and saturation region were extracted using the

SPICE model. These measurements are shown in Table 4.2. In this table subthreshold region is marked with (*).

Supply Voltage (V)	Average Power (µW)		Delay (nS)	
	Conventional Buffer	SBB Buffer	Conventional Buffer	SBB Buffer
0.9	29.58	93.068	0.40	0.14
0.8	19.665	53.161	0.58	0.27
0.7	11.668	30.513	0.92	0.54
0.6	5.5642	9.0586	1.81	1.10
0.5*	1.5643	1.9623	4.56	3.48
0.45*	0.54538	0.64034	12.96	8.59
0.4*	0.1303	0.14963	41.93	24.5
0.35*	0.024716	0.02836	143	75.5

Table 4.2: Average power dissipation and delay of both buffer types for various values of supply voltages

In this table delay is measured as the difference between the time that the output reaches 50% of its final value and the time that the input reaches 50% of its final value [3]. It should be also mentioned that the wiring resistance for the routing track model in this project was measured as 73.66 Ω; however the wiring resistance was not considered for the delay path modeling of the routing track. This was due to the fact that at low frequency the capacitive reactance is significantly larger than the resistance of the circuit. Hence the impedance of the circuit is predominantly capacitive and the wiring resistance can be ignored. For example for our model the total wiring capacitance is 4.63 fF for 30 µm of the routing track. This capacitance at our target frequency of 20 KHz results in a capacitive reactance of approximately 1720 MΩ,

which is much greater than the 73.66 Ω wiring resistances. Therefore the wiring resistance of the routing tracks can be ignored for measurement of the delay of the circuit. Figures 4.9 through 4.15 are the HSPICE plots for delay measurements of routing tracks with conventional buffers and SBB buffers for subthreshold supply voltages of 0.4V, 0.45V, and 0.5V.

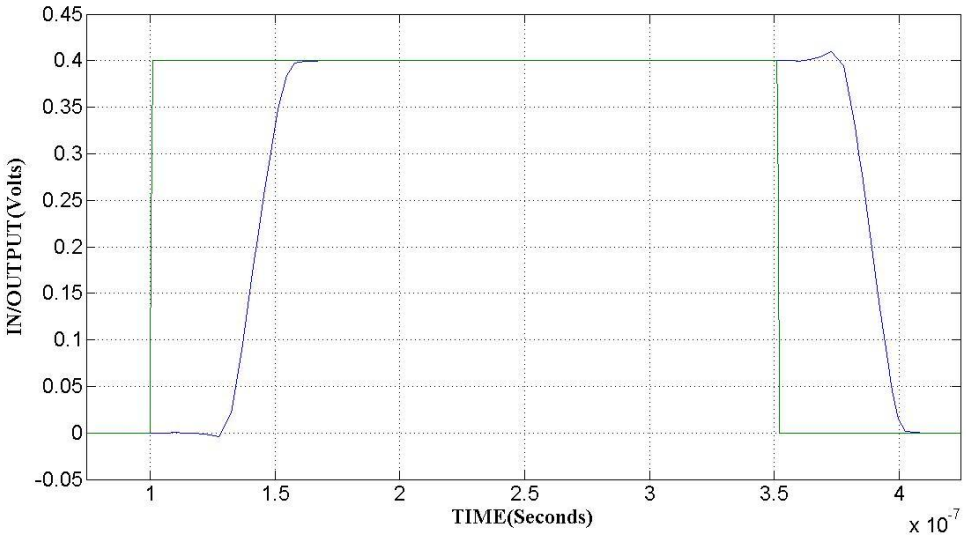


Figure 4.9: Plot of Input vs. Output of the routing track using conventional buffers with 0.4 V supply

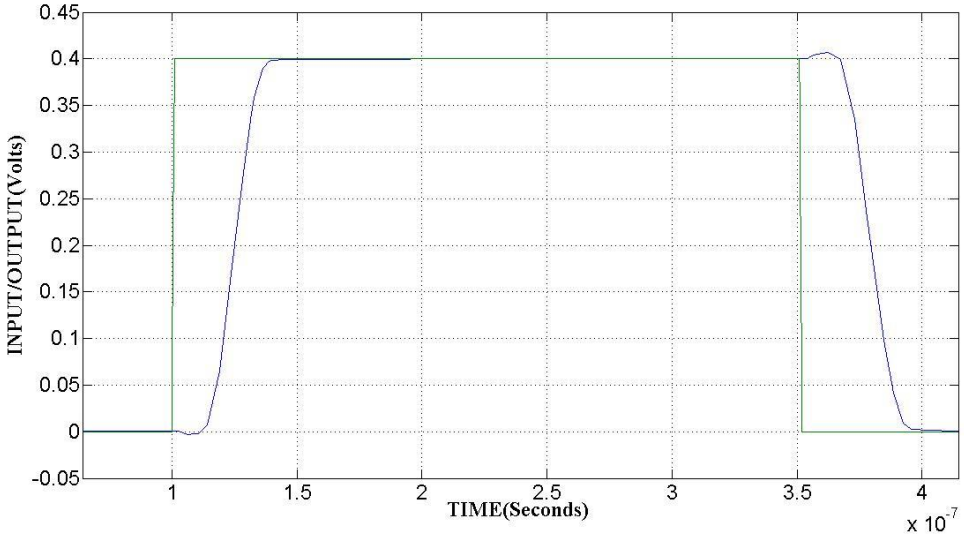


Figure 4.10: Plot of Input vs. Output of the routing track using SBB buffers with 0.4 V supply

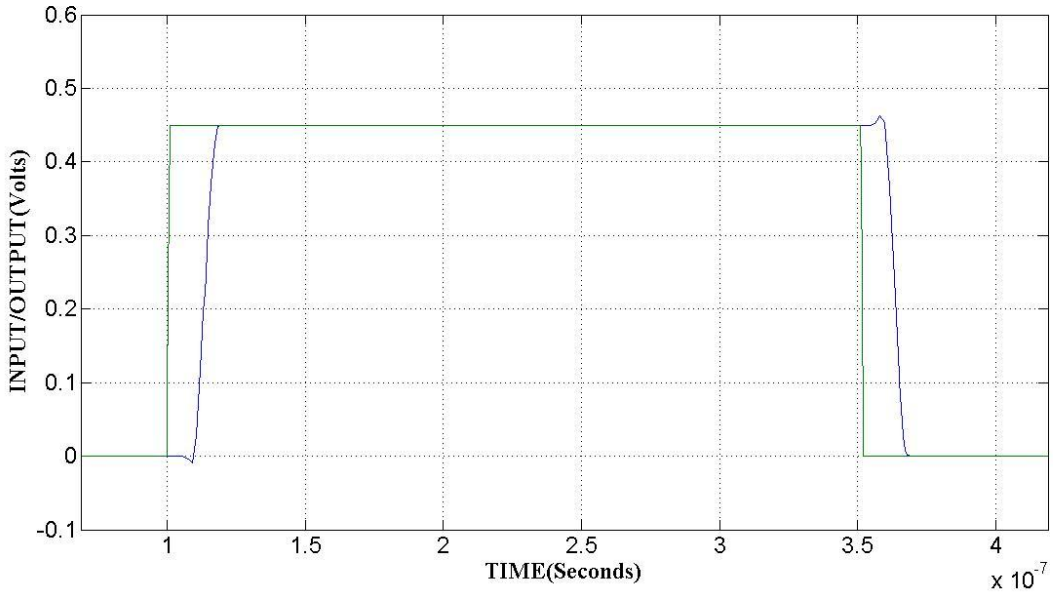


Figure 4.11: Plot of Input vs. Output of the routing track using conventional buffers with 0.45 V supply

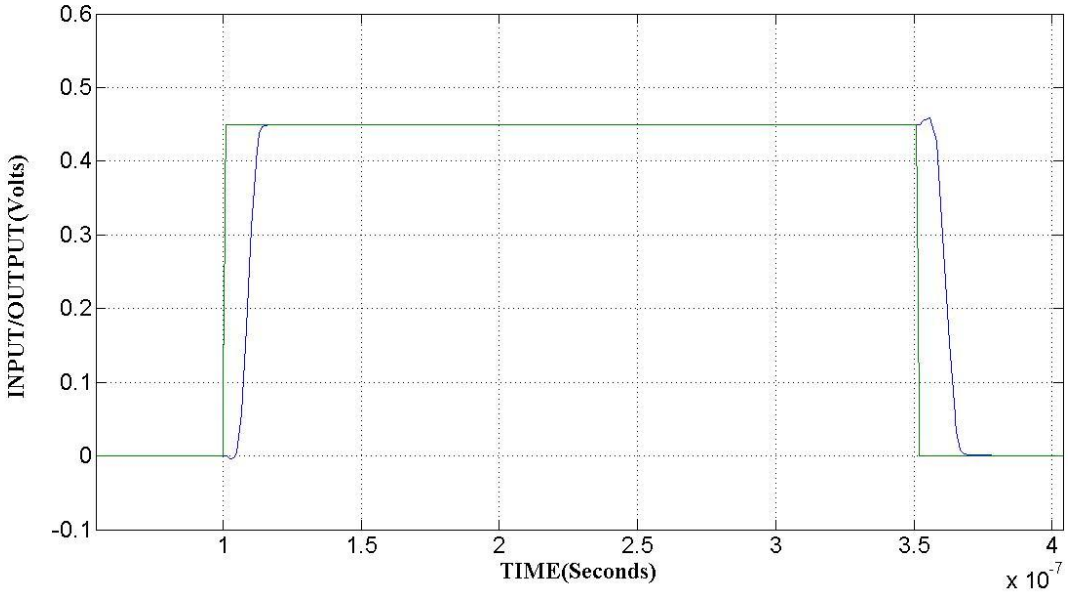


Figure 4.12: Plot of Input vs. Output of the routing track using SBB buffers with 0.45 V supply

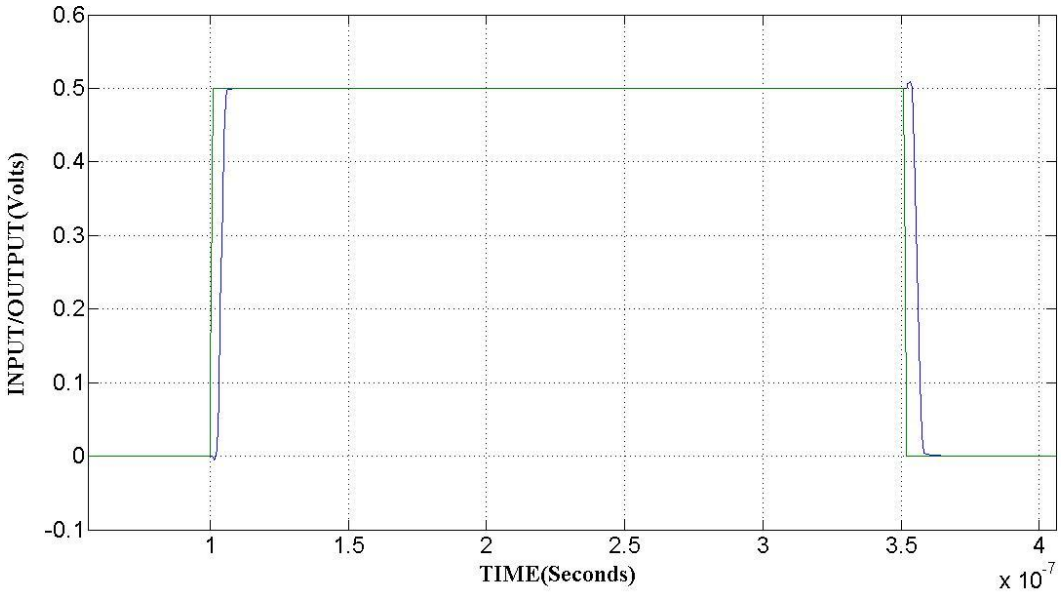


Figure 4.13: Plot of Input vs. Output of the routing track using conventional buffers with 0.5 V supply

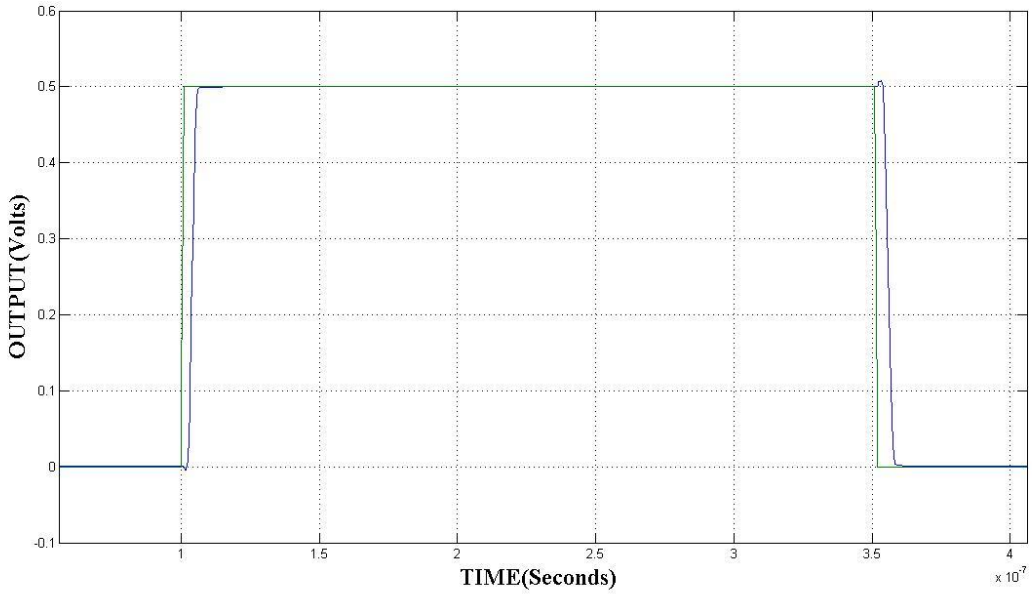


Figure 4.14: Plot of Input vs. Output of the routing track using SBB buffers with 0.5 V supply

In the table 4.2 power and delay figures are shown for supply voltages ranging from 0.9 volts to 0.35 volts below which the outputs of both circuits (with conventional buffers and SBB buffers) become distorted for the 2.0 MHz operating frequency. The model with SBB buffers has lower power-delay product for supply voltages of less than 0.6 volts as it is shown in Figure 4.15.

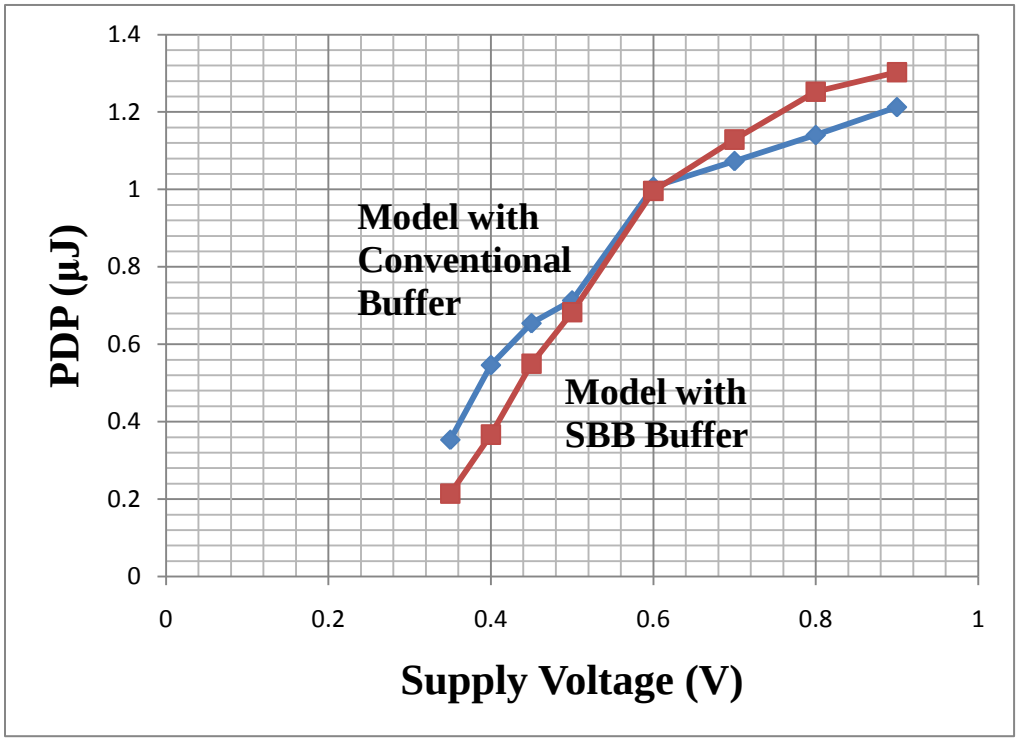


Figure 4.15: Power-Delay Product vs. Supply Voltage for Models with SBB Buffers and Conventional Buffers

Note that this simulation is conducted for only one routing track; however the several routing tracks typically are chained together to connect a signal from its source to its destination on an FPGA. Since the routing tracks investigated in this work are buffered at each end, a linear delay model can be used to calculate the delay of a set of chained routing tracks as shown in Equation 4.1 [16], where is the intrinsic delay of the buffer, is the equalized pull-up/pull-down resistance of the buffer and is the total capacitances that are needed to

be driven by the buffer. Note that the delay values shown in Table 4.2 corresponds to the sum for

.

(4.1)

As shown, the total routing delay increases linearly with a factor of M. For example, with a supply voltage of 0.35 V and SBB type buffers, the delay for a single track is 75.5 ns. Hence the total routing delay for 20 tracks would be 1510 ns which is greater than the 500 ns clock period (2.0 MHz). Using a supply voltage of 0.4V reduces the delay for a single track to 24.5 ns, thus the delay for 20 tracks would be 490 ns. Consequently, a signal with a 10 ns logic delay can still meet the timing constraint of 2.0 MHz operating frequency. Finally, as shown in Table III, using SBB buffers with 0.4V supply voltage improves the power consumption of the circuit by a factor of 197.7 and the power-delay product by a factor of 3.3, as compared to using the conventional buffer at the 0.9 V supply voltage.

Chapter 5

Conclusion and Future Work

The main focus in this research was to investigate ways to reduce total power consumption of biomedical applications, since low power consumption is a crucial factor in designing devices to process biosignals. It was shown that parametric modeling that is often considered for feature extraction can also be used to compress the biomedical signal. This reduces the size of required memory and ultimately results in less power consumption. The AR Burg algorithm has been previously implemented by researchers. Majority of researchers have used a microprocessor base design and some have attempt to use FPGAs or design a custom Integrated Circuits, with their attentions mainly focused on performance. However, not much attention has been given to low power design. In this thesis a custom architecture has been designed, suitable to be implemented on an FPGA or an ASIC chip, with the focus on reducing the total power consumption of the device. The estimated power consumption extracted from the design revealed that a significant amount of power consumed by a device is static. To reduce the static power consumption of a device subthreshold circuit design was investigated in this research. Interconnect power dominates the total power consumption of an FPGA hence the power efficiency of the routing resources is crucial for low-power applications. In this thesis power requirement for implementing a computationally intensive algorithm used for processing

biosignals on an FPGA was investigated. A model for routing tracks of an FPGA was suggested which is able to operate in the subthreshold region while still meeting the timing constraint. In particular, it is shown that using SBB buffers, it is possible to achieve power reduction by a factor of 197.7 and power-delay product reduction by a factor of 3.3 as compared to normal operation in the saturation region using a 0.9 volt supply voltage. The power reduction can significantly increase the battery life of portable devices utilizing FPGAs for biomedical applications.

It should be noted that this work has only considered subthreshold design using 32 nm technology for the routing tracks of an FPGA. In the future, subthreshold design should also be investigated for Logic Blocks, Block RAMs, Functional Units, and other elements that make up the architecture of an FPGA.

Bibliography

[1] F. Li, D. Chen, L. He, and J. Cong, "Architecture Evaluation for Power-Efficient FPGAs," in *Proceedings of the 2003 ACM International Symposium on Field-Programmable Gate Arrays,* Monterey, CA, February 2003, pp. 175-184.

[2] J. Anderson, F. Najm, and T. Tuan, "Active Leakage Power Optimization for FPGAs," in *Proceedings of the 2004 ACM Symposium on Field-Programmable Gate Arrays,* Monterey, CA, February 2004, pp. 33-41.

[3] N. H. E. Weste, D. M. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective 4th Edition,* Addison Wesley, MA, 2010.

[4] R. M. Rangayyan, *Biomedical Signal Analysis: A Case-Study Approach,* Wiley-IEEE Press, NY, 2001.

[5] R. M. Rangayyan, S. Krishnan, G. D. Bell, C. B. Frank, and K. O. Ladly, "Parametric Representation and Screening of Knee Joint Vibroarthrographic Signals," *IEEE Transactions on Biomedical Engineering,* Vol. 44, No. 11, November 1997, pp. 1068-1074.

[6] S. Tavathia, R. Rangayyan, C. Frank, G. Bell, K. Ladly, and Y. Zhang, "Analysis of Knee Vibration Signals Using Linear Prediction," *IEEE Transactions on Biomedical Engineering*, Vol. 39, No. 9, September 1992, pp. 959-970.

[7] J. J. Burr and A. Peterson, "Ultra Low Power CMOS Technology," in *Proceedings of the 1991 NASA Symposium on VLSI Design*, Moscow, ID, October 1991, pp. 4.2.1-4.2.13.

[8] H. Soeleman and K. Roy, "Ultra-Low Power Digital Subthreshold Logic Circuits," in *Proceedings of the 1999 International Symposium on Low Power Electronics and Design*, San Diego, CA, August 1999, pp. 94-96.

[9] K. Roy, S. Mukhopadhayay, H. Mahmoodi-Meimand, "Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits," *Proceedings of the IEEE*, Vol. 91 No. 2, February 2003, pp. 305-327.

[10] B. H. Calhoun, J. F. Ryan, "A Sub-Threshold FPGA with Low-Swing Dual-VDD Interconnect in 90nm CMOS", in *Proceedings of the 2010 IEEE Custom Integrated Circuits Conference*, San Jose, CA, September 2010, pp. 1-4.

[11] G. V. Leming, and K. Nepal, "Low-Power FPGA Routing Switches Using Adaptive Body Biasing Technique," in *Proceedings of the 2009 IEEE International Midwest Symposium on Circuits and Systems*, Cancun, Mexico, August 2009, pp. 447-450.

[12] S. M. Bae, K. Ramakrishnan and N. Vijaykrishnan, "A Novel Low Area Overhead Body Bias FPGA Architecture for Low Power Applications," in *Proceedings of the 2009 IEEE Computer Society Annual Symposium on VLSI*, Tampa, FL, May 2009, pp. 193-198.

[13] D. Ge, N. Srinivasan, and S. M. Krishnan, "Cardiac arrhythmia classification using autoregressive modeling," *BioMedical Engineering Online*, http://biomedical-engineeringonline.com/content/1/1/5, November 2002.

[14] J. Makhoul, "Linear Prediction: A Tutorial Review," *Proceedings of IEEE,* Vol. 63, No. 4, April 1975, pp. 561-580.

[15] L. Griffiths, "A Continuously-Adaptive Filter Implemented as a Lattice Structure", in *Proceedings of the 1977 IEEE International Conference on Acoustics, Speech, and Signal Processing,* Munich Germany, April 1977, pp. 683-686.

[16] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs,* Kluwer Academic Publishers, 1999.

[17] Available online at http://www.eas.asu.edu/~ptm/, Accessed February 2011.

[18] Available online at http://www.eecg.utoronto.ca/vpr/architectures/ , Accessed February 2011.

[19] S. Narendra et. al, "Ultra-Low Voltage Circuits and Processor in 180nm to 90nm Technologies with a Swapped-Body Biasing Technique," in *Proceedings of the 2004 International Solid-State Circuits Conference,* San Francisco, CA, February 2004, pp. 8.4.1-8.4.3.

[20] J. Kao, S. Narendra, A. Chandrakasan, "Subthreshold Leakage Modeling and Reduction Techniques," in *Proceedings of the 2002 IEEE International Conference on Computer Aided Design,* San Jose, CA, November 2002, pp. 141-148.

[21] J. Nyathi,B. Bero, "Logic Circuits Operating in Subthreshold Voltages," in *Proceedings of the 2006 IEEE International Symposium on Low Power Electronics and Design,* Tegernsee, Germany, August 2006, pp. 131-134.

[22] M. Diaby, M. Tuna, J. Desbarbieux, and F.Wajsburt, "High level synthesis methodology from C to FPGA used for a network protocol communication," in *Proceedings of the 15th IEEE International Workshop on Rapid System Prototyping,* 2004., pp. 103–108, June 2004.

[23] K. Camera, "SF2VHD: A stateflow to VHDL translator," Master thesis, UC Berkeley, 2001.

[24] M. A. Shanblatt and B. Fould, "A Simulink-to-FPGA implementation tool for enhanced design flow," in *Proceedings of the 2005 IEEE international conference on microelectronic systems education* (MSE'05), pp. 89–90, 2005.

[25] J. Beibei, "High-level FPGA implementation of adaptive signal segmentation and autoregressive modeling techniques," Master thesis, Ryerson University, 2009.

[26] "Virtex-5 FPGA User Guide V5.3," Available online at http://www.xilinx.com/support/documentation/user_guides/ug190.pdf , Accessed June 2011.

[27] T. Tuan and B. Lai. "Leakage power analysis of a 90nm FPGA," In *Custom Integrated Circuits Conference*, pp. 57-60, San Jose, CA, ,2003.

[28] S. Srinivasan, A. Gayasen, and T. Tuan, "Leakage control in FPGA routing fabric," in *Proceedings of the Asia South Pacific Deign. Automation. Conference*, Shanghai, China, Jan. 2005, pp. 661–664.