Designing of different types of multipliers

DESIGN AND IMPLEMENTATION OF DIFFERENT MULTIPLIERS USING VHDL A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Bachelor of Technology in Electronics and Communication Engineering By MOUMITA GHOSH Department of Electronics and Communication Engineering National Institute of Technology Rourkela 2007 National Institute of Technology Rourkela CERTIFICATE

This is to certify that the thesis entitled, " DESIGN AND IMPLEMENTATION OF DIFFERENT MULTIPLIERS USING VHDL " submitted by Ms Moumita Ghosh in partial fulfillments for the requirements for the award of Bachelor of Technology Degree in Electronics and Communication Engineering at National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by her under my supervision and guidance. To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma. Date: Prof. Dr. K.

K Mahapatra Dept. of Electronics and Communication Engineering National Institute of Technology Rourkela - 769008 ACKNOWLEDGEMENT I would like to articulate my profound gratitude and indebtedness to my project guide Prof. Dr. K. K Mahapatra who has always been a constant motivation and guiding factor throughout the project time in and out as well. It has been a great pleasure for me to get a opportunity to work under him and complete the project successfully. I wish to extend my sincere thanks to Prof. Dr. G Panda, Head of our Department, for approving our project work with great interest.

I would also like to mention Mr. Jitendra Behera, Ms Durga Digdarshani and Mr Sushant Pattnaik assistance. M. Tech Student, for their cooperation and constantly rendered It is my pleasure to refer VHDL, Acrobat Reader and Microsoft Word exclusive of which the whole process, right from simulation to compilation of this report would have been impossible. An undertaking of this nature could never have been attempted with our reference to and inspiration from the works of others whose details are mentioned in references section.

I acknowledge my indebtedness to all of them. Last but not the least, my sincere thanks to all of my friends who have patiently extended all sorts of help for accomplishing this undertaking. CONTENTS

Abstract..... 05 List of Figures.....ii List of Tables..... iii
Introduction..... 09
VHDL 14
Filters..... 26
Type of filters..... 29
Adders.... ....31 Binary multipliers..... 41
Results..... 54
Conclusion..... . 57
Refrences..... 59

ABSTRACT: Low power consumption and smaller area are some of the most important criteria for the fabrication of DSP systems and high performance

systems. Optimizing the speed and area of the multiplier is a major design issue. However, area and speed are usually conflicting constraints so that improving speed results mostly in larger areas.

In our project we try to determine the best solution to this problem by comparing a few multipliers. This project presents an efficient implementation of high speed multiplier using the shift and add method, Radix_2, Radix_4 modified Booth multiplier algorithm. In this project we compare the working of the three multiplier by implementing each of them separately in FIR filter. The parallel multipliers like radix 2 and radix 4 modified booth multiplier does the computations using lesser adders and lesser iterative steps.

As a result of which they occupy lessr space as compared to the serial multiplier. This a very important criteria because in the fabrication of chips and high performance system requires components which are as small as possible. In our project when we compare the power consumption of all the multipliers we find that serial multipliers consume more power. So where power is an important criterion there we should prefer parallel multipliers like booth multipliers to serial multipliers. The low power consumption quality of booth multiplier makes it a preffered choice in designing different circuits

In this project we first designed three different type of multipliers using shift snd method, radix 2 and radix 4 modified booth multiplier algorithm. We used different type of adders like sixteen bit full adder in designing those multiplier. Then we designed a 4 tap delay FIR filter and in place of the multiplication and additions we implemented the components of different

multipliers and adders. Then we compared the working of different multipliers by comparing the power consumption by each of them. The result of our project helps us to choose a better option between serial and parallel multiplier in fabricating different systems.

Multipliers form one of the most important component of many systems. So by analyzing the working of different multipliers helps to frame a better system with less power consumption and lesser area. CHAPTER 1 INTRODUCTION LOW POWER CONSUMPTION INTRODUCTION Multipliers are key components of many high performance systems such as FIR filters, microprocessors, digital signal processors, etc. A system’s performance is generally determined by the performance of the multiplier because the multiplier is generally the slowest clement in the system.

Furthermore, it is generally the most area consuming. Hence, optimizing the speed and area of the multiplier is a major design issue. However, area and speed are usually conflicting constraints so that improving speed results mostly in larger areas. As a result, a whole spectrum of multipliers with different area-speed constraints have been designed with fully parallel. Multipliers at one end of the spectrum and fully serial multipliers at the other end. In between are digit serial multipliers where single digits consisting of several bits are operated on.

These multipliers have moderate performance in both speed and area. However, existing digit serial multipliers have been Plagued by complicated switching systems and/or irregularities in design. Radix 2^n multipliers which operate on digits in a parallel fashion instead of bits bring the

pipelining to the digit level and avoid most of‘ the above problems. They were introduced by M. K. Ibrahim in 1993. These structures are iterative and modular. The pipelining done at the digit level brings the benefit of constant operation speed irrespective of the size of’ the multiplier.

The clock speed is only determined by the digit size which is already fixed before the design is implemented. CHAPTER 2 VHDL : THE LANGUAGE EXPERIMENTAL Many DSP applications demand high throughput and real-time response, performance constraints that often dictate unique architectures with high levels of concurrency. DSP designers need the capability to manipulate and evaluate complex algorithms to extract the necessary level of concurrency. Performance constraints can also be addressed by applying alternative technologies.

A change at the implementation level of design by the insertion of a new technology can often make viable an existing marginal algorithm or architecture. The VHDL language supports these modeling needs at the algorithm or behavioral level, and at the implementation or structural level. It provides a versatile set of description facilities to model DSP circuits from the system level to the gate level. Recently, we have also noticed efforts to include circuit-level modeling in VHDL. At the system level we can build behavioral models to describe algorithms and architectures.

We would use concurrent processes with constructs common to many high-level languages, such as if, case, loop, wait, and assert statements. VHDL also includes user-defined types, functions, procedures, and packages. " In many respects VHDL is a very powerful, high-level, concurrent programming

language. At the implementation level we can build structural models using component instantiation statements that connect and invoke subcomponents. The VHDL generate statement provides ease of block replication and control.

A dataflow level of description offers a combination of the behavioral and structural levels of description. VHDL lets us use all three levels to describe a single component. Most importantly, the standardization of VHDL has spurred the development of model libraries and design and development tools at every level of abstraction. VHDL, as a consensus description language and design environment, offers design tool portability, easy technical exchange, and technology insertion VHDL: The language An entity declaration, or entity, combined with architecture or body constitutes a VHDL model.

VHDL calls the entity-architecture pair a design entity. By describing alternative architectures for an entity, we can configure a VHDL model for a specific level of investigation. The entity contains the interface description common to the alternative architectures. It communicates with other entities and the environment through ports and generics. Generic information particularizes an entity by specifying environment constants such as register size or delay value. For example, entity A is port (x, y: in real; z: out real); generic (delay: time); end A;

The architecture contains declarative and statement sections. Declarations form the region before the reserved word begin and can declare local elements such as signals and components. Statements appear after begin

and can contain concurrent statements. For instance, architecture B of A is component M port (j : in real ; k : out real); end component; signal a, b, c real := 0. 0; begin " concurrent statements" end B; The variety of concurrent statement types gives VHDL the descriptive power to create and combine models at the structural, dataflow, and behavioral levels into one simulation model.

The structural type of description makes use of component instantiation statements to invoke models described elsewhere. After declaring components, we use them in the component instantiation statement, assigning ports to local signals or other ports and giving values to generics. invert: M port map (j => a ; k => c); We can then bind the components to other design entities through configuration specifications in VHDL's architecture declarative section or through separate configuration declarations.

The dataflow style makes wide use of a number of types of concurrent signal assignment statements, which associate a target signal with an expression and a delay. The list of signals appearing in the expression is the sensitivity list; the expression must be evaluated for any change on any of these signals. The target signals obtain new values after the delay specified in the signal assignment statement. If no delay is specified, the signal assignment occurs during the next simulation cycle: c