Journal of Philosophy, Psychology, Linguistics, Law and Demography

https://www.harrisonacademicpress.com/index.php/JPPLD

Vol 1, Issue 1, 2026

DESIGN AND IMPLEMENTATION OF HIGH SPEED 32-BIT MAC (MULTIPLY ACCUMULATE) UNIT

Radhika Rayeekanti, D. Akshaya, K. Bhanurekha, P. kavya

Associate Professor, Department of Electronics and Communication Engineering, Bhoj Reddy Engineering college for women, Hyderabad, Telangana, India.

radhiece2022@gmail.com

[1]dashinolaakshaya231@gmail.com,[2] kbhanurekha77@gmail.com ,[3]ponnurukavya3@gmail.com

Student, Department of Electronic and Communication Engineering, BhojReddy Engineering college for women, Hyderabad, Telangana, India.

Abstract

This paper explores the design and implementation of a high-speed 32-bit MAC (Multiply Accumulate) unit, optimized for digital signal processing (DSP) applications. The MAC unit is a critical component in various computational tasks, particularly in machine learning, image processing, and telecommunications, where performance and efficiency are paramount. We detail the architecture of the MAC unit, focusing on parallel processing techniques, pipelining and optimized data paths that enhance throughput and minimize latency. The proposed design leverages modern semiconductor technologies to achieve high clock rates while maintaining energy efficiency. We present simulation results demonstrating significant performance improvements over traditional MAC implementations, highlighting metrics such as throughput, power consumption, and area efficiency. Additionally, the MAC unit's integration within a larger DSP system is discussed, illustrating its impact on overall system performance. In today's smart and fast computing world, the designing of high speed and low energy consumption based Digital Signal Processors (DSPs) is a realistic and ever embryonic area of research. The multiplier, adder, accumulator are the fundamental construction sub-units for MAC units. This paper has presented the implementation of novel 32-bit MAC unit consisting of Vedic Multiplier using Urdhva Tiryakbhyam sutra and efficient adder circuit using Modified Weinberger adder technique. From comparative analysis, the MAC unit designed was found to be proficient in terms of delay and energy consumed.

1. INTRODUCTION

The multiply-accumulate (MAC) unit is a fundamental block for digital signal processing (DSP) applications. Especially, in recent years, the development of real-time edge applications has become a design trend. Thus,there is a strong demand for high-speed low-power MAC units. A conventional MAC unit is composed of two individual blocks: multiplier and an accumulator (i.e., an accumulate adder). A multiplier usually here has three steps. The first step is the partial product generation (PPG) process. For example, AND gates can be used to generate a partial product matrix (PPM) for an unsigned multiplication. The second step is the partial product reduction (PPR) process. By using the Dadda tree approach or the Wallace tree approach, the PPM can be reduced to become two rows. The third step is the final addition. An adder (called the final adder) is used to perform the summation of the final two rows.

2. LITERATURE SURVEY

Approximate Adders for approximate multiplication: Document The gap between capabilities of CMOS technology scaling and requirements of future application workloads is increasing rapidly. There are several promising design approaches that jointly can reduce this gap significantly. Approximate computing is one of them and in recent years, has attracted the strongest attention of the scientific community. Approximate computing exploits inherent error-resilience of applications and features high-performance energy-efficient software and hardware implementations by trading off computational quality (e.g., accuracy) for computational efforts (e.g., performance and energy). Over the decade, several research efforts have explored approximate computing throughout all the layers of computing stack, however, most of the work at hardware level of For an N-bit multiplier, a (2N-1)-bit adder is required for the final addition. Various adder architectures have been proposed for the trade-offs among delay, area, and power. Furthermore, various MAC unit models can be developed by replacing the multiplier as well as the accumulator (adder) with various architectures. The arithmetic units are not only reduced in complexity, but carries also taken that error value is maintained low, helps in achieving better accuracy, reduced logic complexity of approximate arithmetic units consumes less power and area. The proposed multipliers out performs the existing multiplier designs in terms of area, power, and error, and achieves better peak signal to noise ratio (PSNR) values in image processing application. Error distance (ED) can be defined as the arithmetic distance between a correct output and approximate output for a given input. abstraction has been proposed on adders. In [1], a comparative survey of state of-the-art approximate adders is provided. And it also provides comparison based on both conventional design metrics as well as approximate computing design metrics.

Compressors for Multiplication: Approximate computing is an attractive paradigm for digital processing at nanometric scales. Inexact computing is particularly interesting for computer arithmetic designs. The analysis and design of two new approximate 4-2 compressors are explained in for utilization in a multiplier. These designs rely on different features of compression, such that imprecision in computation (as measured by the error rate and the so-called normalized error distance) can meet with respect to circuit-based figures of merit of a design (number of transistors, delay and power consumption). Four different schemes for utilizing the proposed approximate specified number of times. A number (multiplicand) is added itself a number of times as specified by another number (multiplier) to form a result (product). Multipliers play an important role in today's digital signal processing and various other applications. Multiplier design should offer high speed, low power consumption.we propose a low-power high-speed pipeline multiply-accumulate (MAC) architecture. In a conventional MAC, carry propagations of additions (including additions in multiplications and additions in accumulations) often lead to large power consumption and large path delay. To resolve this problem, we integrate a part of additions into the partial product reduction (PPR) process. In the proposed MAC architecture, the addition and accumulation of higher significance bits are not performed until the PPR process of the next multiplication. To correctly deal with the overflow in the PPR process, a small-size adder is designed to accumulate the total number of carries. Compared with previous works, experimental results show that the proposed MAC architecture can greatly reduce both power consumption and circuit area under the same timing constraint. The high-speed 32-bit Mac unit was an essential stepping stone in the evolution of Apple's computing technology. This section provides an overview of different methodologies for classifying hyper-spectral images, as well as brief explanations of the algorithms adopted by the researcher.

3. TOOLS ANALYSIS

3.1 XILINX

The Integrated Software Environment (ISE™) is the Xilinx® design software suite that allows you to take your design from design entry through Xilinx programming. The ISE Project Navigator manages

and processes your design through the following steps in the ISE design flow.Design entry is the first step in the ISE design flow. During design entry, you create your source files based on your design objectives. You can create your top-level design file using a Hardware Description Language (HDL), such as VHDL, Verilog, or ABEL, or using a schematic. You can use multiple formats for the lower-level source files in your design.After design entry and optional simulation, you run synthesis. During this step, VHDL, Verilog, or mixed language designs become netlist files that are accepted as input to the implementation step.

After synthesis, you run design implementation, which converts the logical design into a physical file format that can be downloaded to the selected target device. From Project Navigator, you can run the implementation process in one step, or you can run each of the implementation processes separately. Implementation processes vary depending on whether you are targeting a Field Programmable Gate Array (FPGA) or a Complex Programmable Logic Device (CPLD).



Fig 3.1 Project Navigator

- Toolbar
- Sources window
- Processes window
- Workspace
- Transcript window

USING THE SOURCES WINDOW

The first step in implementing your design for a Xilinx® FPGA or CPLD is to assemble the design source files into a project. The Sources tab in the Sources window shows the source files you create and add to your project, as shown in the following figure. For information on creating projects and source files, see Creating a Project and Creating a Source File.

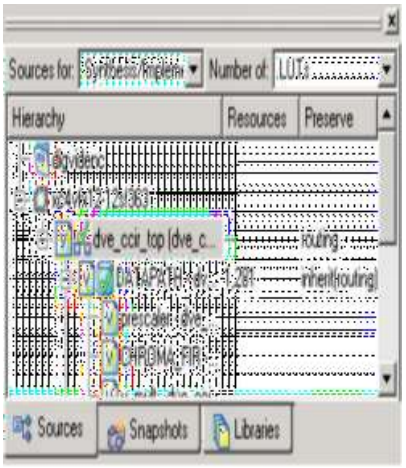


Fig 3.2 Sources Window

The Design View ("Sources for") drop-down list at the top of the Sources tab allows you to view only those source files associated with the selected Design View (for example, Synthesis/Implementation). For details, see Using the Design Views

3.2 Verilog

Introduction

Verilog HDL is a Hardware Description Language (HDL). A Hardware Description Language is a language used to describe a digital system, for example, a computer or a component of a computer. One may describe a digital system at several levels. For example, an HDL might describe the layout of the wires, resistors and transistors on an Integrated Circuit (IC) chip, i.e., the switch level or, it might describe the logical gates and flip flops in a digital system, i.e., the gate level. An even higher level describes the registers and the transfers of vectors of information between registers. This is called the Register Transfer Level (RTL). Verilog supports all of these levels. However, this handout focuses on only the portions of Verilog which support the RTL level.

Verilog is one of the two major Hardware Description Languages (HDL) used by hardware designers in industry and academia. VHDL is the other one. The industry is currently split on which is better. Many feel that Verilog is easier to learn and use than VHDL. As one hardware designer puts it, "I hope the competition uses VHDL." VHDL was made an IEEE Standard in 1987, while Verilog is still in the IEEE standardization process.

Verilog Code Structure

The Verilog language describes a digital system as a set of modules. Each of these modules has an interface to other modules to describe how they are interconnected. Usually we place one module per file but that is not a requirement. The modules may run concurrently, but usually we have one top level module which specifies a closed system containing both test data and hardware models. The top level module invokes instances of other modules. Modules can represent bits of hardware ranging from simple gates to complete systems, e. g., a microprocessor. Modules can either be specified behaviorally or structurally (or a combination of the two). A behavioral specification defines the behavior of a digital system (module) using traditional programming language constructs, e. g., ifs, and assignment statements. A structural specification expresses the behavior of a digital system (module) as a hierarchical interconnection of sub modules. At the bottom of the hierarchy the components must be primitives or specified behaviorally. Verilog primitives include gates, e. g., nand, as well as pass transistors (switches). The <module name> is an identifier that uniquely names the module. The <port list> is a list of input, The <declares> section specifies data objects as registers, memories and wires as wells as procedural constructs such as functions and tasks. The <module items> may be initial constructs, always constructs, continuous assignments or instances of modules.

4. CONCLUSION

Computer hardware has grown in power at an amazing pace ever since. The most important computational resources is energy which is deeply linked to the reversibility of the computation. The primary objective of this project was to gain insight into the Reversible Computation and its use for making devices energy efficient for long life. Multiplier is a basic arithmetic cell in computer arithmetic units and it is supposed to be the most power consuming unit when higher order multiplication is to be performed. In this work, looking at advantages of reversibility, we synthesized a parity preserving reversible multiplier circuit with the help of existing fault tolerant Fredkin, F2G and IG gate. We proposed this project which presents low-power high-speed two-stage pipeline MAC architecture for real-time DSP applications. Our basic idea is to integrate a part of additions (including a part of the final addition in the multiplication and a part of the addition in the accumulation) into the PPR process. As a result, critical path delays and power dissipations caused bycarry propagations can be reduced. To correctly deal with the overflow during the PPR process, an α-bit accumulator is used to count the total number of carries. Experimental results consistently show that the proposed approach input and output ports which are used to connect to other modules. works well in practice. The proposed MAC architecture is applicable to both the design of an unsigned MAC unit and the design of a signed MAC unit. Note that the only differences between the unsigned MAC unit and the signed MAC unit are the PPM structure and the α-bit addition mechanism. Moreover, the proposed MAC architecture is also applicable to the systolic array (for performing the matrix multiplication). Implementation data show that, compared with the systolic array based on the conventional PE (i.e., the conventional MAC architecture), the systolic array based on the proposed PE (i.e., the proposed MAC architecture) can greatly reduce both circuit area and power consumption under the same timing constraint. SCS-based multipliers maintain the input and Output operands of the Montgomery MM in the carrysave format to escape from the format conversion, leading to fewer clock cycles but smaller area than FCS-based multiplier. In the existed architecture disadvantages are carry propagation delay and extra clock cycles. To overcome the disadvantages we go for PASTA adder.

5. REFERENCES

V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, “Low-power digitalsignal processing using approximate adders,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 32, no. 1, pp. 124–137,Jan. 2013

.E. J. King and E. E. Swartzlander, Jr., “Data-dependent truncationscheme for parallel multipliers,” in Proc. 31st Asilomar Conf. Signals,Circuits Syst., Nov. 1998, pp. 1178–1182.

K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, “Design oflow-error fixed-width modified booth mutiplier,” IEEE Trans. VeryLarge Scale Integr. (VLSI) Syst., vol. 12, no. 5, pp. 522–531,May 2004.

H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, “Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57, no. 4, pp. 850–862, Apr. 2010.