FPGA Implementation Of Low Power Parallel Multiplier

U.Meiyarasan[1], G.Sathishkumar[2], M.Selvakumar[3], Dr. J.Rangarajan[4]
[1, 2, 3, 4] Dept of ECE
[1, 2, 3, 4] Muthayammal Engineering College, Namakkal,Tamilnadu, India

Abstract- *In the fast growing communication field, requirements of low power designs are increasing to reduce the power losses and decrease the thermal losses in the same ratio.Multiplier is an arithmetic circuit that is extensively used in common DSP and Communication applications. This paper presents low power multiplier design methodology that inserts more number of zeros in the multiplicand thereby reducing the number of switching activities as well as power consumption. Use of look up table is an added feature to this design. Modifying the structure of adders further reduces switching activity.The design process involves integrating clock gating logic into a ripple-carry adder architecture and implementing it using hardware description language (HDL). The Verilog design is simulated and validated using ModelSim, ensuring accurate functionality and verifying the clock gating's effect on power dissipation. Additionally, synthesis is performed using industry-standard tools to assess the area, power, and timing trade-offs. The results demonstrate a measurable reduction in power consumption compared to a conventional adder, making the design suitable for low-power applications. This document provides a comprehensive overview of the design methodology, implementation details, and performance analysis of the clock-gated adder. The findings underline the importance of power optimization techniques in digital design and highlight the potential for further exploration in power-efficient arithmetic units*

Keywords- Low Power, Multiplier, Reduced Switching.

I. INTRODUCTION

As we get closer to the limits of scaling in complementary metal–oxide–semiconductor (CMOS) circuits, power and heat dissipation issues arebecomingmore and more important. In recent years, the impact of pervasive computing and the internet have accelerated this trend. The applications for these domains are typically run on battery-powered embedded systems. The resultant constraints on the energy budget require design for power as well ass design for performance at all layers of system design. Thus reducing power consumption is a key design goal for portable computing and communication devices that employ increasingly sophisticated and power hungry signal processing techniques. Flexibility is another critical requirement that and dates the use of programmable componts like FPGAs in such devices. However, there is a fundamental trade-off between efficiency and flexibility, and asa result, programmable designs significant performance and power penalties compared to application specific solutions. Consequently various digital signal- processing chips are now designed with low power dissipation.

Signal processing applications typically exhibit high degrees of parallelism and are dominated bya few regular kernels of computation such as multiplication, that are responsible for a large fraction of execution time and energy. In such systems, multiplier is a fundamental arithmetic unit[1]. Shrinking feature sizes are responsible for increasing thermal-related problems as well. The on-chip temperature in current processors can vary by as much as several tens of degrees from one portion of the chip to the other with the maximum temperature reaching as high as 100 degree C. The temperature gradient formed by such units can be a major source of inaccuracy in delay and clock skew computations [2].

II. LITERATURE SURVEY

The efficient implementation of low-power parallel multipliers on Field-Programmable Gate Arrays (FPGAs), addressing the growing demand for energy-conscious digital signal processing (DSP) and high-performance computing. Multipliers are fundamental building blocks in numerous applications, and their power consumption significantly impacts overall system efficiency, particularly in portable and embedded devices. FPGAs offer a flexible platform for prototyping and implementing custom hardware, enabling the exploration of diverse low-power techniques.

The research begins by reviewing essential multiplication algorithms, including array, Booth, Wallace, and Dadda multipliers, analyzing their trade-offs in terms of speed, area, and power. It then delves into FPGA architecture, focusing on configurable logic blocks (CLBs), DSP blocks, and interconnect resources, and how these elements can be effectively utilized for multiplier implementation.

Understanding power consumption in FPGAs, encompassing static and dynamic power, is crucial for developing low-power designs.

The core of the paper lies in the comprehensive literature survey, examining various low-power techniques applied to FPGA multipliers. These techniques encompass algorithm-level optimizations, such as modified Booth encoding, optimized partial product reduction, and operand decomposition, which aim to minimize switching activity and computational complexity. Circuit-level optimizations, including clock gating, voltage scaling, and low-power adder designs, are also explored to reduce dynamic and static power dissipation. FPGA-specific optimizations, such as exploiting DSP blocks, resource sharing, pipelining, and power-aware placement and routing, are analyzed to leverage the unique characteristics of FPGA architecture.

The implementation of clock gating in arithmetic circuits has also been investigated. Liu et al. designed a clock-gated adder and compared its power consumption with a conventional adder.Their findings indicated a significant reduction in switching activity without compromising performance. In another study, Zhang et al. analyzed the trade-offs between power, performance, and area (PPA) in clock-gated arithmetic units, emphasizing the importance of careful design considerations.

These studies collectively establish the effectiveness of clock gating as a power-saving technique. The ongoing advancements in clock gating methodologies, including adaptive and machine learning-based approaches, continue to enhance power efficiency in modern digital circuits. This research builds on these existing works by implementing and evaluating a clock-gated 8-bit adder in Verilog, providing insights into its practical benefits and trade-offs.

III. METHODOLOGY

The this research aims to systematically design and evaluate low-power parallel multipliers for FPGA implementation. Initially, a thorough analysis of prominent multiplication algorithms, including Booth, Wallace, and Dadda, will be conducted to determine their suitability for low-power design and FPGA adaptation. This stage involves theoretical assessment of computational complexity and identification of algorithm-level optimizations, such as modified Booth encoding and optimized partial product reduction, to minimize switching activity. Subsequently, the selected algorithms will be translated into synthesizable HDL code, targeting specific FPGA platforms. Utilizing FPGA vendor-specific tools and libraries, including DSP blocks, the implementations will be optimized for speed, area, and power efficiency. Circuit-level optimizations, such as clock gating, pipelining, and low-power adder designs, will be incorporated to reduce dynamic power consumption. Furthermore, voltage scaling techniques and power-aware placement and routing will be employed to minimize both static and dynamic power dissipation. High Level Synthesis tools will be used to generate some of the designs to compare against traditional HDL based designs.

Power analysis and performance evaluation will be conducted using FPGA vendor-provided estimation tools, focusing on accurately determining power consumption, speed, and area metrics. Simulation and hardware testing on FPGA development boards will be used to validate the functionality and performance of the implemented multipliers. A comparative study will be performed, comparing the developed designs with existing solutions reported in the literature, to assess the effectiveness of the proposed low-power techniques. This comprehensive evaluation will involve analyzing the trade-offs between speed, area, and power consumption, and validating the accuracy of any approximate multipliers used. The impact of voltage scaling on performance and power will be carefully measured and analyzed. Finally, the results obtained from HLS generated designs will be compared to HDL generated designs to determine the effectiveness of each method. This multi-faceted methodology ensures a rigorous and comprehensive investigation into the design and optimization of low-power parallel multipliers for FPGA platforms.

System Architecture

The proposed system consists of the following key components:

Ripple-Carry Adder

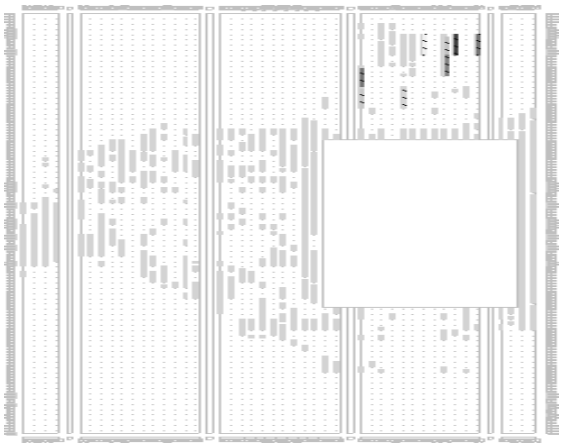
- A simple and robust adder structure is chosen for its ease of implementation.
- Adds two 8-bit binary numbers along with an optional carry input to produce an 8-bit sum and a carry output.
- The operation is sequential, propagating the carry bit through all stages.

Clock Gating Logic

- Implements an AND-based clock gating mechanism:
    - The clock signal is gated (enabled or disabled) using an AND gate.

- An enable signal determines whether the clock signal is propagated to the adder logic.

- When the enable signal is inactive, the clock signal is blocked, preventing unnecessary toggling of flip-flops



Floorplannerviewfor16x16 multiplier in Xilinx xc2vp

.

Control Unit

- Generates the enable signal based on the system's operational state.
- Monitors external inputs or system requirements to determine whether the adder is active or idle.
- Ensures proper synchronization and prevents glitches during transitions

.

Functional Verification

- The design is modeled using Verilog HDL at the RTL level .
- A comprehensive testbench is developed to verify functionality under various input conditions .
- Functional correctness is ensured before proceeding to synthesis.

Power Analysis

- Power estimation is performed using Synopsys Design Compiler and Xilinx Vivadotools .
- Power savings achieved through clock gating are compared against a conventional non-gated adder .

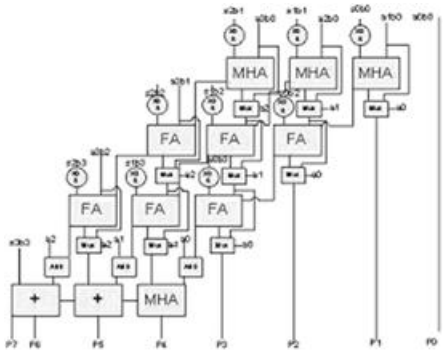
Timing and Performance Analysis

- Simulation is carried out using ModelSim to analyze waveforms and validate clock gating effectiveness [.
- Performance metrics such as propagation delay, setup time, and hold time are measured [14].

Scalability and Integration

- The architecture is tested for scalability by extending it to higher bit-width adders [15].

Feasibility for integration into low-power embedded systems is assessed [16].



4x4Multiplier architecture

IV. SOFTWARE DESCRIPTION FOR VIVADO

Vivado Design Suite is a software suite produced by Xilinx (now AMD) for synthesis and analysis of Hardware Description Language (HDL) designs, specifically targeting Xilinx FPGAs (Field-Programmable Gate Arrays). It's a comprehensive toolchain that covers the entire FPGA design flow, from design entry and synthesis to implementation, verification, and bitstream generation.

Here's a breakdown of Vivado's key software components and functionalities:

1. Design Entry and Synthesis:

- **HDL Editor:** Provides a text editor for writing and editing HDL code (Verilog, VHDL, SystemVerilog).
- **IP Integrator:** A graphical environment for designing systems using pre-built IP (Intellectual Property) blocks, including Xilinx's own IP cores and third-party IP.
- **High-Level Synthesis (HLS):** Allows designers to create hardware designs using high-level languages like C, C++, and SystemC. Vivado HLS translates these high-level descriptions into RTL (Register-Transfer Level) code.
- **Synthesis:** Translates the HDL or HLS code into a gate-level netlist, optimizing the design for the target FPGA architecture.

2. Implementation:

- **Implementation:** Maps the gate-level netlist onto the FPGA's physical resources (LUTs, flip-flops, DSP blocks, etc.). This includes placement (assigning logic elements to specific locations on the FPGA) and routing (connecting the logic elements).
- **Optimization:** Performs various optimizations to improve the design's performance, area, and power consumption.
- **Timing Analysis:** Analyzes the timing characteristics of the design to ensure that it meets the specified timing constraints.
- **Power Analysis:** Estimates the power consumption of the design.

3. Verification and Debugging:

- **Simulation:** Allows designers to simulate the behavior of their designs before implementing them on the FPGA. Vivado supports various simulation techniques, including behavioral simulation and timing simulation.
- **Logic Analyzer (ILA):** Provides an on-chip logic analyzer that allows designers to capture and analyze signals within the FPGA during runtime.
- **Hardware Debugging:** Tools for debugging the implemented design on the FPGA, including signal probing and breakpoint setting.

MagnitudeRegister

0	0	0	1	0	0	0	0	1	1	0	0	0	1	0	0

SignRegister(s)

0	0	0	0	0	0	0	0	*b*	0	0	0	0	*b*	0	0

4. Bitstream Generation and Device Programming:

- **Bitstream Generation:** Generates a bitstream file that contains the configuration data for the FPGA.
- **Device Programming:** Allows designers to program the FPGA with the generated bitstream.

Key Features and Benefits:

- **Integrated Design Environment:** Provides a unified environment for all stages of the FPGA design flow.
- **High-Performance Synthesis and Implementation:** Optimizes designs for performance, area, and power.
- **Advanced Timing Analysis:** Ensures that designs meet timing constraints.
- **Comprehensive Verification and Debugging Tools:** Facilitates design verification and debugging.
- **IP Integration:** Simplifies the integration of IP cores.
- **High-Level Synthesis:** Enables hardware design using high-level languages.
- **Power Analysis and Optimization:** Supports power-aware design.
- **Targeting Latest Xilinx FPGAs:** Support for the newest generations of Xilinx/AMD FPGAs.

Scalability and Customization:

- Vivado is designed to handle very large and complex designs, scaling well with increasing FPGA complexity.
- It allows for a high degree of customization, enabling designers to tailor the design flow to their specific needs.

Tcl Scripting:

- Vivado heavily utilizes Tcl scripting, allowing for automation of design flows and repeatable processes. This is crucial for large projects and team collaboration.
- Designers can create custom scripts to perform specific tasks, such as generating reports, modifying design parameters, or automating implementation steps.

Partial Reconfiguration:

- Vivado supports partial reconfiguration, which allows designers to dynamically change portions of the FPGA design without interrupting the operation of other parts. This is valuable for applications that require dynamic functionality updates.

Ecosystem Integration:

- Vivado integrates with other design tools and platforms, facilitating seamless workflows.
- It supports industry-standard interfaces and file formats, enabling interoperability with third-party tools

Libraries and IP Cores:

- Xilinx provides a rich library of pre-built IP cores, including processors, memory controllers, and communication interfaces.
- These IP cores can significantly reduce design time and effort.

UltraFast Design Methodology:

- Vivado and Xilinx provide the UltraFast Design Methodology to help designers get to optimal results faster. This methodology gives guidelines for design closure, timing closure, and other critical aspects of FPGA design.

Licensing:

- Vivado is available in different editions, each with varying features and capabilities. Licensing options include node-locked and floating licenses.

Continuous Updates:

- AMD regularly releases updates to Vivado, adding new features, improving performance, and supporting the latest FPGA devices. This ensures that designers have access to the most up-to-date tools and technologies.

Documentation and Support:

- AMD provides extensive documentation, tutorials, and support resources for Vivado, including online forums and technical support.

    - Select the Testbench as the top module for simulation.
    - Initialize the simulation using the Simulate menu.
2. Run the Simulation:
    - Run the simulation for a specified duration or until a particular condition is met.
    - Use run commands (run 100ns, run -all, etc.) in the ModelSim console.
3. View Waveforms:
    - Open the Waveform Viewer to observe signals like clock, enable, inputs (a, b, cin), and outputs (sum, cout).
    - Analyze whether the gated clock behaves as expected, blocking unnecessary transitions during idle states.

Debugging

- Use breakpoints, watch windows, and the waveform viewer to debug issues in the design.
- Verify edge cases (e.g., carry-in propagation, clock transitions) using simulation tools.

Features Used in This Project

1. Waveform Analysis:
    - Visualize signal activity and transitions.
    - Confirm the correctness of clock gating and arithmetic operations.
2. Debug Console:
    - Execute commands for step-by-step simulation.
    - Check intermediate values of signals.
3. Simulation Control:
    - Use commands like restart, force, and run to control simulation flow.
4. Log Files:
    - Generate detailed logs of simulation results for documentation and troubleshooting.

Outputs from Vivado

- Simulation Waveforms:
    - Illustrates the clock gating effect (clock transitions disabled during idle states).
    - Shows correct addition outputs for various input combinations.
- Debugging Reports:
    - Identifies and resolves issues in the Verilog design.
    - Validates the enable signal logic for smooth transitions between active and idle states.
- Functional Verification:
    - Confirms that the design operates correctly under all test cases defined in the Testbench.

Advantages of Using Vivado

- Ease of Use: Intuitive interface for waveform viewing and debugging.
- Comprehensive Analysis: Offers detailed insights into signal behavior and logic transitions.
- Integration with Synthesis Tools: Compatible with popular synthesis tools for seamless verification and implementation.

V. CONCLUSION

In this paper we have presented a newmethodology for designing of low power parallel multiplier with reduced switching.Method for increasing number of zeros in the multiplicand is discussed with the help of Binary / Booth Recoding Unit. We use look up tablefor implementing the logic for counting the number of ones and generation of booth recoded multiplicand. Comparing with column bypassing and other techniques our methodology guarantees to have equal or more number of zeros in the multiplicand. Effective implementation of 16x16 multiplier in FPGA is also presented.

REFERENCES

[1] Oscal T. -C. Chen, SandyWang, and Yi-Wen Wu, "Minimization of Switching Activities of Partial Productsfor DesigningLow-Power Multipliers", *IEEETransactionsonVLSISystems,* June 2003 vol. 11, no.3.

[2] Rajendra M. Patrikar, K. Murali, Li Er Ping, "Thermal distribution calculations for block level placement in embedded systems", *MicroelectronicsReliability* 44(2004) 129-134

[3] Hichem Belhadj, Behrooz Zahiri, Albert Tai "Power-sensitive design techniques on FPGAdevices*",ProceedingsofInternationalconferenceonICTaipai* (2003).

[4] A. Wu, "High performance adder cell for low power pipelined multiplier*",inProc.IEEEInt.Symp.onCircuitsandSystems*,May1996, vol. 4,pp.57-60.

[5] S. Hong, S. Kim, M.C. Papaefthymiou, and W.E. Stark, "Low power parallel multiplier design for DSP applications through coefficient optimization*",inProc.ofTwelfthAnnualIEEEInt.ASIC/SOConf.,*Sep. 1999, pp. 286-290.

[6] C. R. Baugh and B. A.Wooley, "A two's complement parallel array multiplication algorithm",*IEEETrans.Comput.,* Dec. 1973,*vol.C-22,* pp.1045– 1047.

[7] I. S. Abu-Khater, A. Bellaouar, and M. Elmasry, "Circuit techniques for CMOS low-power high-performancemultipliers", *IEEEJ.Solid-StateCircuits*, Oct. 1996, vol. 31, pp. 1535–1546.

[8] J. Ohban, V.G. Moshnyaga, and K. Inoue, "Multiplier energy reduction through bypassing of partial products," *Asia-PacificConf.onCircuitsandSystems.*2002.,vol.2, pp. 13-17.

[9] Ming-Chen Wen,Syying-Jyan Wang, and Yen-Nan Lin, "Low Power Parallel Multiplier with Column Bypassing",*Electronicsletters,*10,12May2005 Volume41,IssuePage(s):581–583.