# **Low-Cost Enhanced Folded Pipeline Multiplier Design and Implementation Integration for IOT devices**

Karmel Aqqad1  Kamilia Aqqad2

Department of Electrical and Computer Department of Electrical and Computer

Engineering Engineering

Birzeit University Birzeit University

Ramallah, Palestine Ramallah, Palestine

[1191379@student.birzeit.edu](mailto:1191379@student.birzeit.edu) [1191348@student.birzeit.edu](mailto:1191348@student.birzeit.edu)

Arwa Doha3  Yaman Kurdi4

Department of Electrical and Computer Department of Electrical and Computer

Engineering Engineering

Birzeit University Birzeit University

Ramallah, Palestine Ramallah, Palestine

[1190324@student.birzeit.edu](mailto:1190324@student.birzeit.edu) [1191865@student.birzeit.edu](mailto:1191865@student.birzeit.edu)

***Abstract*—** The paper aims to offer a low-cost design for an Internet of Things (IoT) device high-performance multiplier. Performance is enhanced by the design's use of a folded pipeline architecture, which reduces the critical route length. The implementation is affordable for use in VLSI applications since it makes use of a generic 32nm or smaller CMOS library. Simulation findings show that the suggested design is an attractive option for Internet of Things devices that need quick multipliers since it reduces implementation costs and provides better performance than conventional methods. The effectiveness and performance of system-on-chip (SoC) designs depend heavily on hardware accelerators. Designers of complicated hardware accelerators may quickly generate many performance-cost trade-off solutions for each component by utilizing high-level synthesis (HLS). Finding the Pareto-optimal system-level implementations inside this design space is a challenging optimization problem, nevertheless. We introduce COSMOS, an autonomous process that employs compositional scheduling to coordinate memory optimization and HLS tools for the design-space exploration (DSE) of complicated accelerators. [1]

*Index Terms—*Folded Pipeline Multiplier, Register Transfer Level, Verdi Tool,

# **INTRODUCTION**

In this paper, a generic 32 nm or lower CMOS library is used to propose a low-cost enhanced folded pipeline multiplier for Internet of Things (IOT) devices through design and implementation. IOT applications are calling for more affordable and effective multipliers that meet high performance and low power consumption requirements. The present multiplier design minimizes latency and power consumption by reducing the number of pipeline stages through the use of folded pipeline architecture. The authors used a generic CMOS library to further reduce costs and enable their concept to be used in a range of IOT devices. a promising solution for low-cost, high-performance multipliers in IoT applications, the results of its execution show considerable advantages in terms of power and performance when compared to previous designs.

A staggering amount of information is now available in picture format thanks to advancements in imaging technology. The practice of statistically converting an image from one format to another, usually to alter certain aspects of the picture, is known as image processing. The need to handle digital photographs has grown significantly in the last several years. Reducing the amount of bits needed for digital picture representations requires certain strategies in order to make efficient use of them. [2]

# **Folded Pipeline multiplier:**

Our research presents a new multirate folding transformation that may be utilized to create control circuits for pipelined VLSI designs that execute multirate algorithms in a systematic manner. Multirate folding time-multiplexes the multirate algorithm to hardware in such a way that the resultant synchronous architecture only requires a single-clock signal, even if multirate algorithms feature decimators and expanders that alter the effective sample rate of a discrete-time signal. Two related problems are addressed using the multirate folding equations that are obtained. Memory needs in folded architectures is the first problem. We provide formulas for the least amount of registers needed by a folded architecture that carries out a multirate algorithm. Time-shifting is the second problem. We develop retiming for folding constraints, which specify how a multirate data-flow graph has to be retimed in order for a certain schedule to be viable, based on the noble identities of multirate signal processing. The methods presented in this study may be used to the synthesis of architectures for a broad range of multirate algorithm-based digital signal processing applications, including wavelet transform and subband decomposition-based coding and signal analysis. [3]

## **REGISTER TRANSFER LEVEL (RTL)**

The current thought is that transaction level (TL) modeling is the next development stage for complicated integrated circuit and systems design entry. This implies that automated synthesis tools will support this modeling level definition increasingly as it develops, enabling design capture to begin at a higher abstraction level than it does now. The language SystemC is designed to accommodate hardware transaction level specifications by default. The results of the tests show that TL descriptions offer a quicker route to system validation and that automation of the design cycle may be envisioned from this level of abstraction with little negative effects on the final implementation's quality. [4]

## **Verdi Tool**

A comprehensive solution to improve and speeds up your design entry, debug, and verification management is the Synopsys Verdi debug and verification management platform. Verdi gives you the capacity to organize, carry out, and assess the coverage of your simulation regressions thanks to its strong features and integration with the most widely used signal database (FSDB). Verdi also has state-of-the-art debugging tools that provide you visibility into all design and verification flows. Strong AI technology included into Verdi makes it simple to traverse a variety of intricate design settings and automate challenging and time-consuming debugging processes. [5]

## **Layout**

A layout is a schematic of the masks that will be used to create your design. Because layout decides whether your design will function or not, it is equally important as defining the specifications of your devices! There are two methods for creating a layout: automatic and manual. When opposed to an automated approach, manual layout typically allows the designer to fit more devices in a smaller space, but it is also more laborious. In contrast, the automated process uses conventional cells and typically requires a larger amount of real estate, but it operates considerably more quickly. [6]

## **Floor planning**

Floorplanning is a crucial design stage since it establishes the dimensions, forms, and placements of the modules within a chip, estimating the chip's overall area, interconnects, and latency. VLSI floorplanning is an NP hard problem in terms of computation. For tackling the VLSI floorplan problem, various researchers have proposed a variety of heuristics and metaheuristic algorithms. A crucial component of the floorplanning stage is the floorplan depiction. The intricacy and search space of the floorplan design are significantly influenced by the floorplan representations. [7]

## **Placement**

The process of placing standard cells inside a design is called placement. Every standard cell's position on the template is established by the tool. These items are positioned by this tool using internal algorithms. The placement improves the design in addition to determining how the available standard cells are arranged in the produced netlist. The placement also establishes if routing the design is feasible.

Numerous factors, including time, congestion, and power optimization, will be taken into consideration while determining placement. [8]

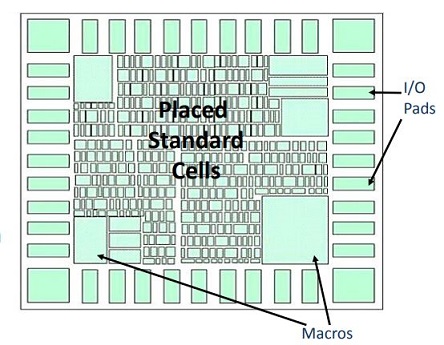


Figure ‎I‑1: Placement [8]

## **Routing**

Routing is the technique of creating physical connections between signal ports through the use of metal layers. Routing is the step that comes after optimization and CTS (Clock Tree Synthesis), during which exact connections between input/output ports, standard cells, and macro units are made. The logical connections included in the netlist—which transform logical connections into physical connections—specify the metals and ports that are used to create electrical connections in the layout. Following CTS, we possess data on all configured cells, obstructions, and flip-flops/latches for the clock tree as well as input/output pins. [9]

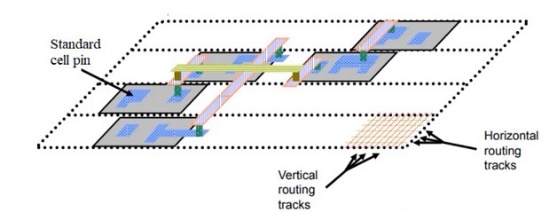


Figure ‎I‑2: Routing [9]

## **Timing analysis**

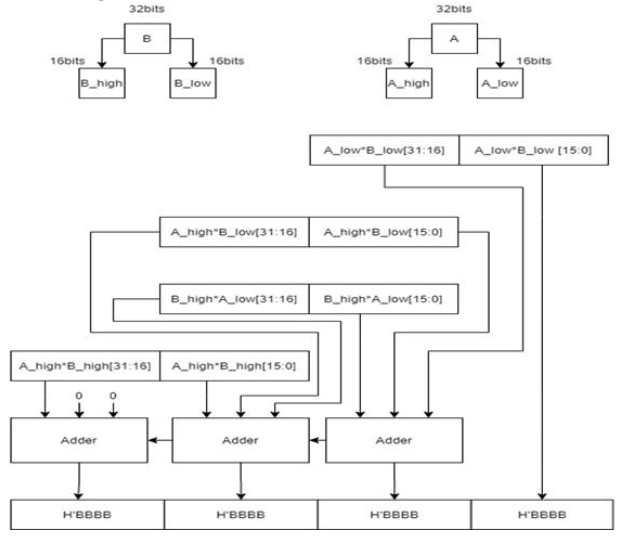
The maximum delay of all pathways from the primary inputs to the primary outputs must be determined for digital VLSI circuits. Statistical Timing Analysis (STA) is a traditional simulation technique used to identify key routes, describe the timing performance of digital circuits, and extract delay information. This comprehensive approach thoroughly examines, troubleshoots, and confirms a design's timing performance. An STA tool typically partitions the whole design into timing pathways, estimates the inter-cell and intra-cell delays to compute the delay on each path, and then investigates each path for timing constraint violations. [10]

# **Existing Projects and Comparison**

Multiplier designs in IoT devices vary based on specific needs like performance, power consumption, and cost. Common designs include array multipliers, Wallace tree multipliers, and booth-encoded multipliers. Array multipliers use full-adders for basic multiplication, but they can be area-intensive and power-hungry. Wallace tree multipliers reduce partial products for a more efficient design, though carry chain delays can impact performance. Booth-encoded multipliers lessen partial products and additions but have overhead from encoding and decoding. The proposed Low-Cost Enhanced Folded Pipeline Multiplier Design, a modified array multiplier with a folded pipeline, cuts delay and power use. Compared to existing designs, it offers advantages like smaller area and lower power than array and Wallace tree multipliers, and better performance than booth-encoded ones. The design is cost-effective and easily integrated into IoT applications, making it appealing for designers. Implementing it using a 32nm CMOS library ensures a low-cost, high-performance, and low-power solution. While beneficial, further enhancements, optimizations, and real-world testing are needed for higher bit-widths, different operations, integration with accelerators, and improved efficiency in diverse IoT scenarios. Continuous refinement can provide a more efficient and versatile solution for various applications in IoT devices.

# **Simulation and Results**

The development and implementation of our folded pipeline multiplier involve several key stages: Specification, RTL (Register Transfer Level) design, logic synthesis, floor planning, routing, and placement. Initially, the design for our folded pipeline multiplier project was conceptualized. The algorithm was established, outlining the process of taking two 32-bit numbers and dividing each into 16-bit High and 16-bit Low segments. The operation details are illustrated in the figure below.



**Figure 1:** Folded Pipelined Multiplier Design

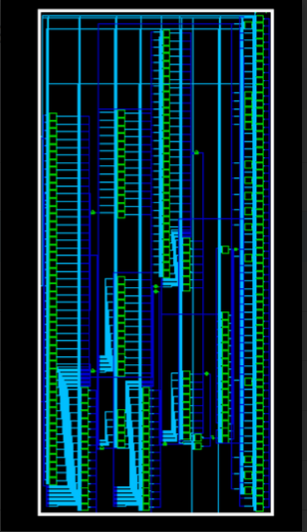
In the initial phase of this project, the Verilog code for the Low-Cost Enhanced Folded Pipeline Multiplier underwent synthesis using the Design Vision tool. Following this, it was transformed into a gate-level netlist. Subsequently, the design underwent the placement and routing process utilizing the ICC2 tool to ensure optimal utilization of the target CMOS library. These methodologies were employed to evaluate the performance of the multiplier design concerning critical criteria, including throughput, latency, power consumption, and space usage. Each stage of this process will be explored in more detail later in this paper. The comprehensive design and implementation of the proposed multiplier, along with simulation and evaluation, are detailed in this document, presented in the sequence in which the tasks were carried out. The verilog code was initially provided and then edited, leading to the final version presented here.

In the realm of digital circuitry, the 64-Bit Multiplier Circuit Design stands as a pivotal element, contributing to the efficient processing of numerical operations. This circuit, adept at handling 64-bit inputs, operates by multiplying two 32-bit numbers, demonstrating its significance in computational tasks demanding precision and complexity. The incorporation of clock and reset signals ensures synchronous and controlled execution.



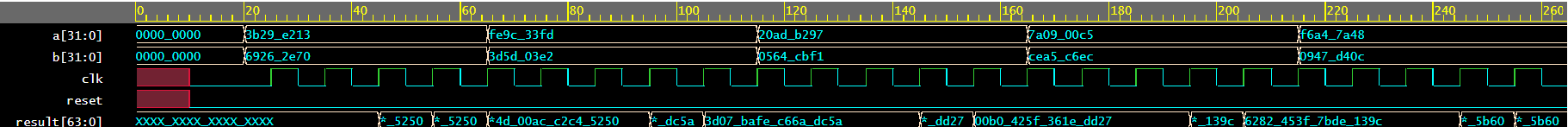
**Figure 2:** 64-Bit Multiplier Circuit Design

In the design process, we utilized several commands to establish specific guidelines: 'create\_clock' fashioned a clock named "clk" with a 10 ns period and a waveform featuring a 2 ns high phase and a 7 ns low phase; 'set\_max\_area' constrained the maximum area to 2000 units; 'set\_max\_fanout' limited the connections for instances labeled "multiplier" to 6, and 'set\_max\_transition' imposed a 3 ns cap on the transition time for these instances. Once these constraints were applied, the design underwent compilation, code conversion into gates, resulting in summary schedules and a schematic layout illustrating the design's configuration.



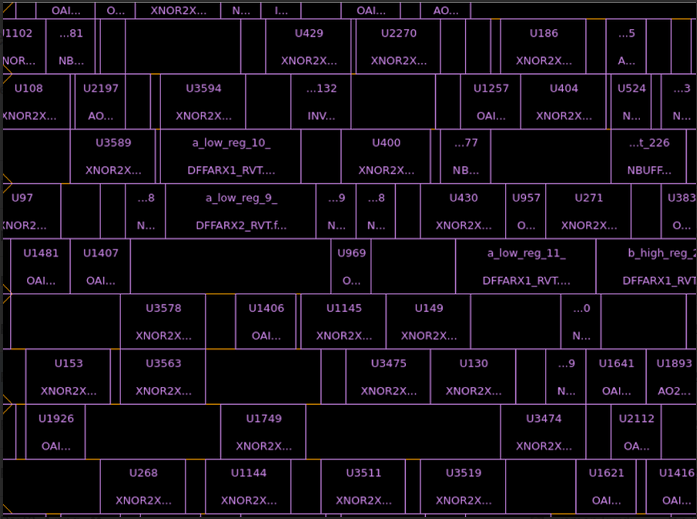
**Figure 3:** Schematic layout of the code

The Verdi waveform for the design shown in the below figure which it contains the values for the clock, reset, input 'a,' input 'b,' and the resultant output.



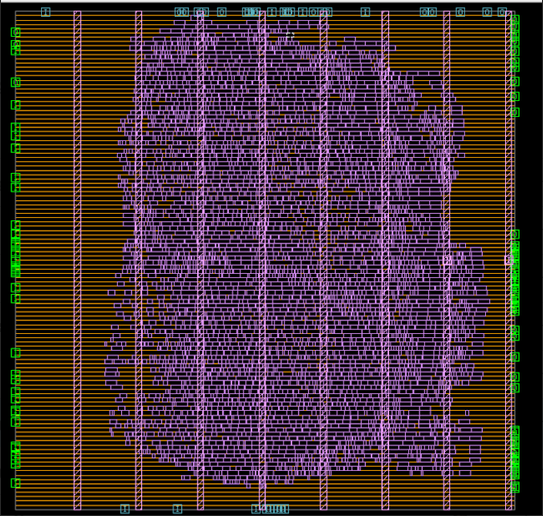
**Figure 4:** waveform of Circuit Design

Zooming in on the schematic layout depicted in the screenshot above reveals a detailed examination of the design's intricate components and their interconnections, providing a closer look at the physical arrangement and relationships within the circuit



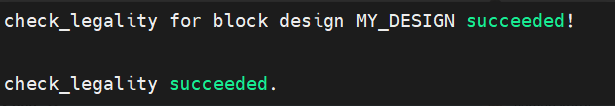
**Figure 5:** layout zoom in

The displayed image showcases the metal layer of the design, offering a detailed view of the physical arrangement and connections within the circuit's metal components.



**Figure 6:** Design metal

The tool's message "check legality for block design successes" suggests that the design validation process is verifying the compliance and legality of the block-level components, ensuring that the construction and configuration of the blocks align with the specified design criteria and standards.



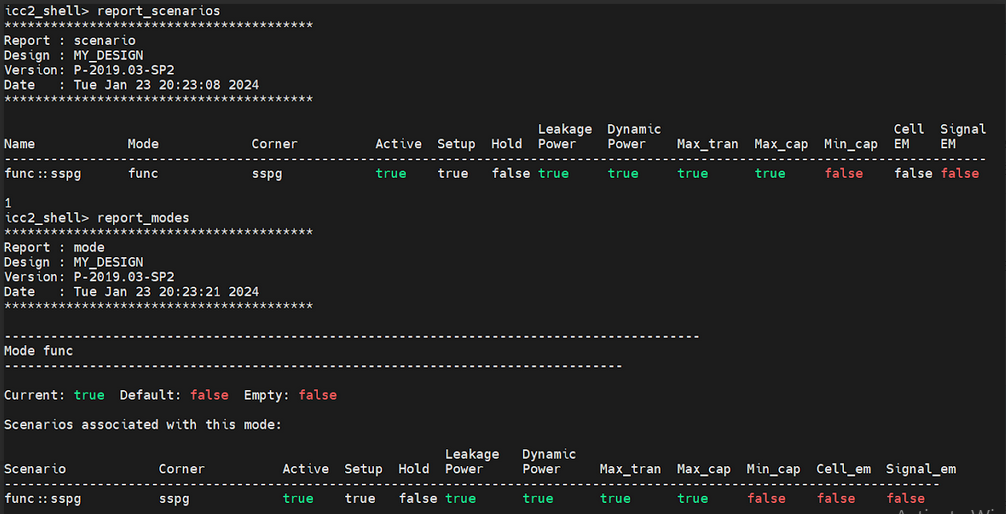
**Figure 7:** Design meta

Summary of placement analysis for design MY\_DESIGN, revealing wire length metrics, placement violations, and voltage area placement issues.



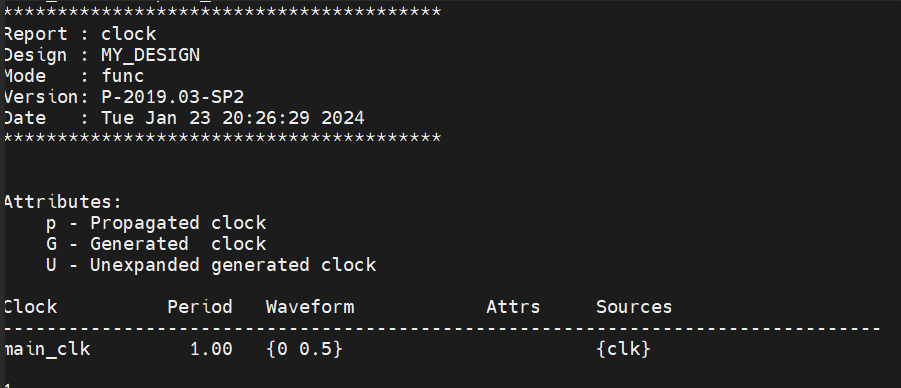
**Figure 8:** Placement Analysis Summary

This report provides insights into the power modes and scenarios of design MY\_DESIGN. The scenarios highlight power, setup, and corner attributes, with specific configurations for functional and sspg modes. The associated modes, such as Leakage, Dynamic, and Max tran, are detailed, showcasing settings for power, setup, and corner attributes. These analyses are crucial for understanding and optimizing the power behavior of the design.



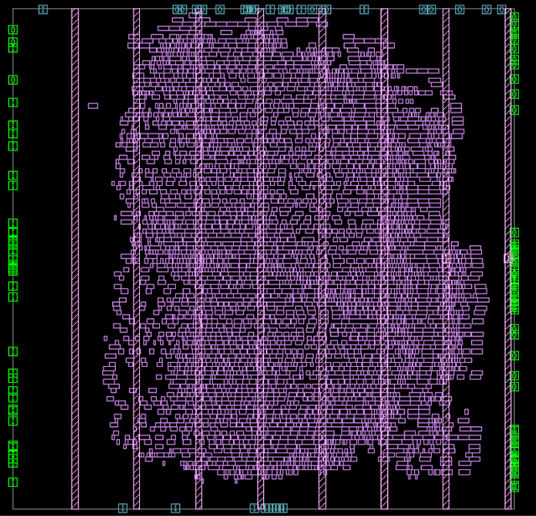
**Figure 9:** Design Power Modes and Scenarios Analysis.

This report details the clock configuration for design in functional mode. The main clock, named 'main\_clk,' has a period of 1.00 units, and its waveform attributes are specified with a 0 to 0.5 transition. The clock is both propagated and generated, and its unexpanded form is denoted as 'U.' This information provides a comprehensive overview of the key clock attributes crucial for the functioning of the design.



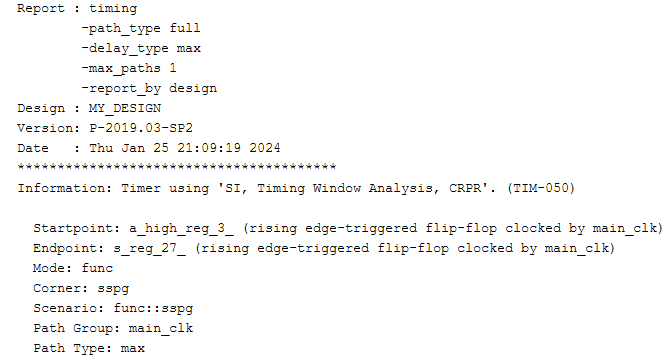
**Figure 10:** Clock Configuration Report

The Fillers Report is generated during the physical verification stage of the VLSI design flow. This report furnishes details about the number, types, and sizes of filler cells used in the layout, along with their respective locations. Additionally, it may include information about the percentage of the layout covered by fillers and highlight any violations of design rules associated with their usage.

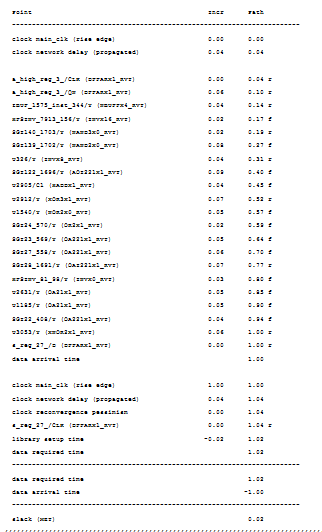


**Figure 11:** Design after full placement

This report analyzes the critical timing path in design, focusing on the maximum delay path under functional mode and the sspg corner scenario. The path, starting at 'a\_high\_reg\_3\_' and ending at 's\_reg\_27\_,' is clocked by 'main\_clk.' Key details include mode, corner, and scenario, providing concise insights into the critical timing characteristics of the design's main clock path.

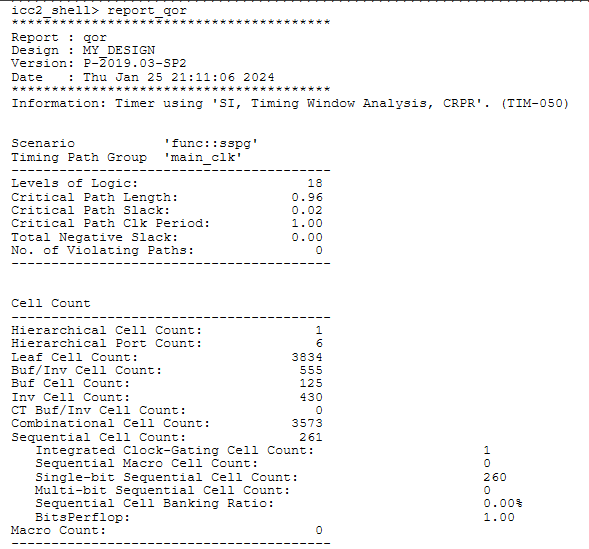


**Figure 12:** Critical Timing Path Analysis



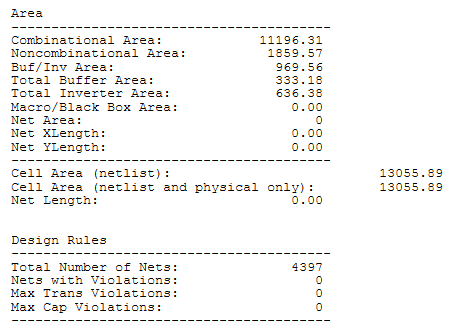
**Figure 13:** Timing Path Analysis Summary: Critical Path and Slack Evaluation

The Quality of Results (QoR) analysis for design MY\_DESIGN, conducted using the 'SI, Timing Window Analysis, CRPR' timer, provides a comprehensive overview of various design metrics. Under the functional scenario 'func::sspg' and within the timing path group 'main\_clk,' the critical path is identified with 18 levels of logic, a length of 0.96, and a slack of 0.02. The critical path's clock period is 1.00, and there are no violating paths, resulting in a total negative slack of 0.00. The cell count analysis reveals 3834 leaf cells, including 555 buffers, 125 buffer cells, and 430 inverters. In terms of area, the combinational area is 11196.31, noncombinational area is 1859.57, and the total buffer and inverter areas are 333.18 and 636.38, respectively. There are no violations in design rules, with a total of 4397 nets and no violations in terms of maximum transition or maximum capacitance. Overall, the design appears to meet critical timing requirements and design rule constraints, reflecting a sound Quality of Results.

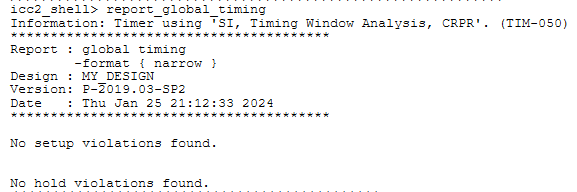


**Figure 14:** Quality of Results (QoR) Analysis for: Functional Scenario and Timing Path Evaluation

Design Area Breakdown: Combinational, Noncombinational, and Buffer/Inverter Areas Analysis

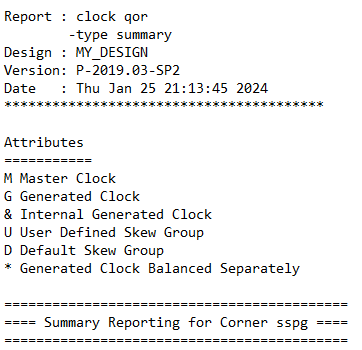


**Figure 15:** Area Analysis



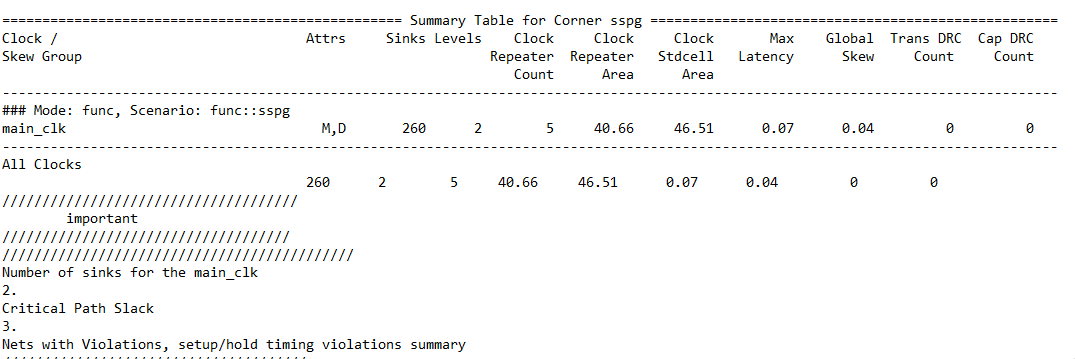
**Figure 16:** Global Timing Analysis

The clock Quality of Results (QoR) summary for design MY\_DESIGN under the 'func::sspg' scenario and 'sspg' corner provides an insightful analysis. The main clock, denoted as 'main\_clk,' exhibits 260 sinks across 2 levels, with attributes including Master Clock (M) and Default Skew Group (D). The clock repeater count is 5, and the corresponding areas for the repeater, standard cell, and global skew are reported. Notably, the critical path slack is highlighted as 0.07, emphasizing the timing margin for critical paths. The summary offers essential details, such as the number of sinks for the main clock (2), and mentions the absence of setup and hold timing violations, providing a concise overview of the clocking characteristics and quality of timing in the design.



**Figure 17:** Clock Quality of Results (QoR) Summary.

This table presents a comprehensive overview of the clocking characteristics for the main clock ('main\_clk') in design MY\_DESIGN under the 'func::sspg' scenario. The attributes include Master Clock (M) and Default Skew Group (D), with 260 sinks distributed across 2 levels. The summary details aspects such as clock repeater count, repeater area, standard cell area, maximum latency, global skew, and counts for both transition and capacitance design rule check (DRC) violations. The critical path slack of 0.07 indicates a favorable timing margin, and the report emphasizes the absence of setup and hold timing violations. This provides crucial insights into the clock quality of results, guiding further optimization efforts in the design.



**Figure 18:** Clock QoR Summary Table for Corner sspg.

# Conclusion

In conclusion, the utilization of a folded-pipelined multiplier design proves advantageous for the swift execution of high-bit count multiplication operations, achieving both time and power efficiency. This approach involves the parallel operation of four 32-bit multipliers, followed by the summation of their results. In contrast, a conventional 64-bit multiplier could be significantly slower, up to 3-4 times, compared to the 32-bit circuits. On a different note, the integration of a low-cost enhanced folded pipeline multiplier design for IoT devices successfully addresses the challenge of creating high-performance, power-efficient, compact, and affordable multipliers. The folded pipeline architecture enables rapid data processing with minimal energy consumption and area usage. Leveraging icc2 and its tools, we optimized the design for superior performance, power efficiency, and area utilization. Overall, our study demonstrates the feasibility and success of the proposed low-cost multiplier design for IoT devices, indicating its potential application in other circuits. This method provides an effective and economical solution for low-power IoT devices, with future investigations aimed at refining the design for additional performance measures and exploring alternative circuit topologies to further enhance efficiency and cost-effectiveness.

# **Appendix**

// Code your design here

module folded\_pipelined\_multiplier (

    input wire clk,

    input wire reset,

    input wire [31:0] a,

    input wire [31:0] b,

    output reg [63:0] result

    );

    reg [15:0] a\_low;

    reg [15:0] a\_high;

    reg [15:0] b\_low;

    reg [15:0] b\_high;

    reg [31:0] p;

    reg [31:0] q;

    reg [31:0] r;

    reg [31:0] s;

    reg [1:0] c1,c2;

    always @(posedge clk or posedge reset) begin

        if (reset) begin

        a\_low <= 0;

        a\_high <= 0;

        b\_low <= 0;

        b\_high <= 0;

        p <= 0;

        q <= 0;

        r <= 0;

        s <= 0;

        result <= 0;

    end else begin

      //first cycle

      a\_low <= a[15:0];

      a\_high <= a[31:16];

      b\_low <= b[15:0];

      b\_high <= b[31:16];

      //second cycle

      p <= a\_low \* b\_low;

      q <= a\_low \* b\_high;

      r <= a\_high \* b\_low;

      s <= a\_high \* b\_high;

      //third cycle

      result[15:0] <= p[15:0];

      {c1,result[31:16]} <= p[31:16] + q[15:0] + r[15:0];

      //fourth cycle

      {c2,result[47:32]} <= q[31:16] + r[31:16] + s[15:0] + c1;

      //fifth cycle

      result[63:48] <= s[31:16] + c2;

    end

    end

endmodule

// Code your testbench here

module tb\_top();

  reg clk, reset;

  bit [31:0] a,b;

  wire [63:0] result;

  initial begin

    #10

    reset <= 0;

    clk <= 0;

    #10

    forever begin

      #5

      clk <= ~clk;

    end

  end

  initial begin

    #20

    for (int i = 0; i < 5; i++)begin

      a <= $urandom();

      b <= $urandom();

      repeat(5) @(posedge clk);

      if (result != a\*b)begin

        $display("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nTEST FAILED\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

        $finish();

      end

    end

    $display("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nTEST PASSED\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

    $finish();

  end

  folded\_pipelined\_multiplier a1 (.\*);

  initial begin

    $dumpfile("dump.vcd");

    $dumpvars;

  end

endmodule

# **REFERENCES**

1. Piccolboni, L., Mantovani, P., Guglielmo, G. D., & Carloni, L. P. (2017). COSMOS: Coordination of high-level synthesis and memory optimization for hardware accelerators. ACM Transactions on Embedded Computing Systems (TECS), 16(5s), 1-22.
2. Mohammad, K., Agaian, S., & Hudson, F. (2010). Implementation of Digital Electronic Arithmetics and its application in image processing. Computers & Electrical Engineering, 36(3), 424-434.
3. Denk, T. C., & Parhi, K. K. (1998). Synthesis of folded pipelined architectures for multirate DSP algorithms. IEEE transactions on very large scale integration (VLSI) systems, 6(4), 595-607.
4. Calazans, N., Moreno, E., Hessel, F., Rosa, V., Moraes, F., & Carara, E. (2003, September). From VHDL register transfer level to SystemC transaction level modeling: a comparative case study. In 16th Symposium on Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings. (pp. 355-360). IEEE.
5. <https://www.synopsys.com/content/dam/synopsys/verification/datasheets/verdi-ds.pdf>
6. <https://www.bioee.ee.columbia.edu/courses/cad/html/layout.html>
7. Singh, R. B., Baghel, A. S., & Agarwal, A. (2016, March). A review on VLSI floorplanning optimization using metaheuristic algorithms. In 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) (pp. 4198-4202). IEEE.
8. <https://www.vlsi-backend-adventure.com/placement.html>
9. <https://www.vlsi-backend-adventure.com/routing.html>
10. Avadhani MD, S. (2020). Future of Timing Analysis in VLSI Circuits. International Journal of Electrical Engineering and Technology, 11(4).