LOW-COST ENHANCED FOLDED PIPELINE MULTIPLIER DESIGN AND IMPLEMENTATION INTEGRATION FOR IOT DEVICES

Karmel Aqqad[1]
Department of Electrical and Computer Engineering
Birzeit University
Ramallah, Palestine
1191379@student.birzeit.edu

Kamilia Aqqad[2]
Department of Electrical and Computer Engineering
Birzeit University
Ramallah, Palestine
1191348@student.birzeit.edu

Arwa Doha[3]
Department of Electrical and Computer Engineering
Birzeit University
Ramallah, Palestine
1190324@student.birzeit.edu

Yaman Kurdi[4]
Department of Electrical and Computer Engineering
Birzeit University
Ramallah, Palestine
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, Layout.**

I. 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 number of bits needed for digital picture representations requires certain strategies in order to make efficient use of them. [2]

I. 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]

II. 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]

III. 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]

IV. 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]

A. 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]

B. 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]

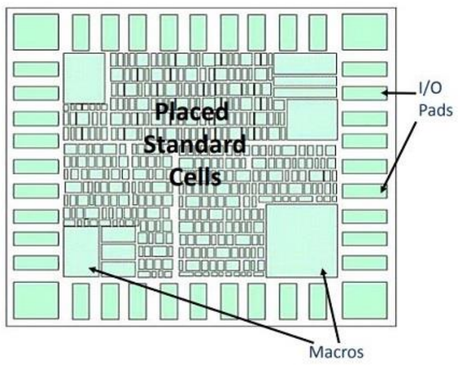


Figure 1: Placement [8]

C. 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]

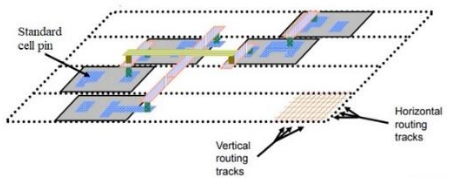


Figure 2: Routing [9]

D. 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]

II. 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.

III. 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.

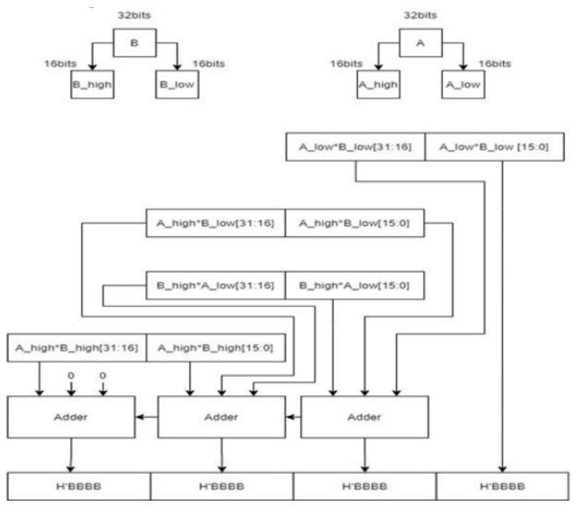


Figure 1: Folded Pipelined Multiplier Design

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.

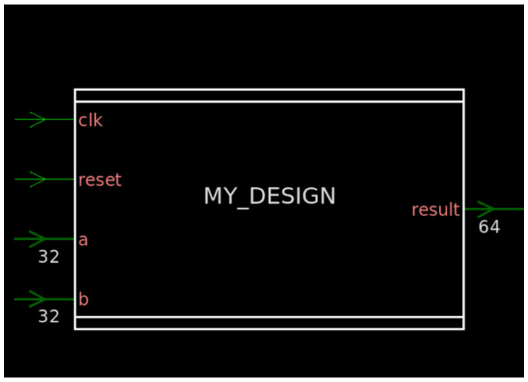


Figure 2: 64-Bit Multiplier Circuit Design

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.

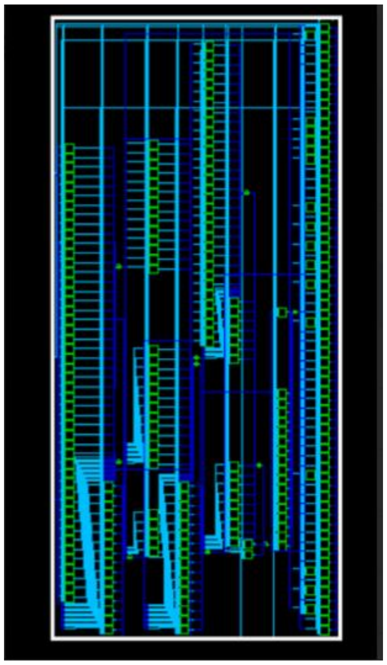


Figure 3: Schematic layout of the code

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.

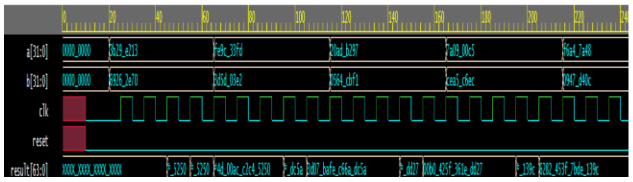


Figure 4: waveform of Circuit Design

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

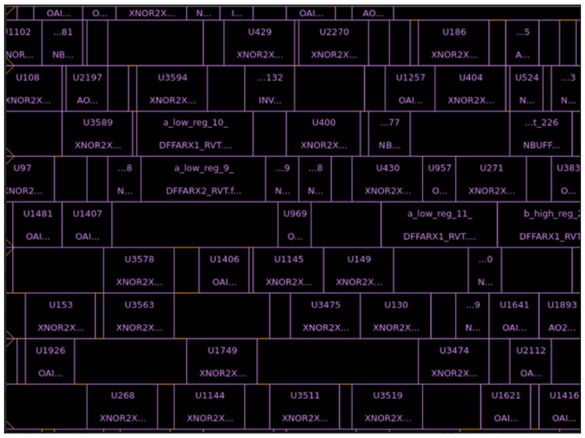


Figure 5: layout zoom in

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.

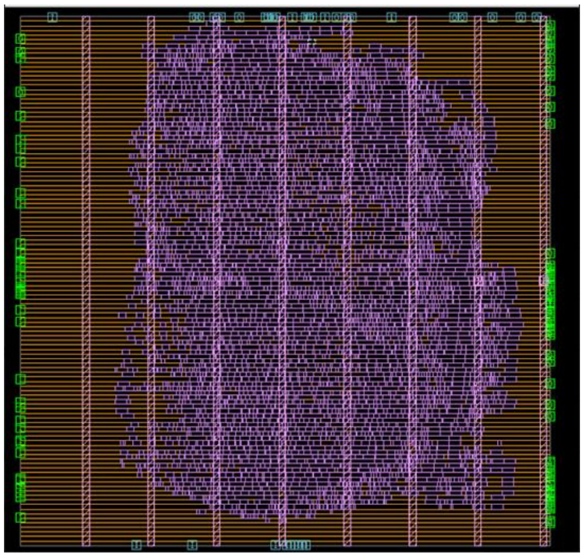
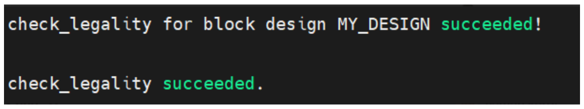


Figure 6: Design metal

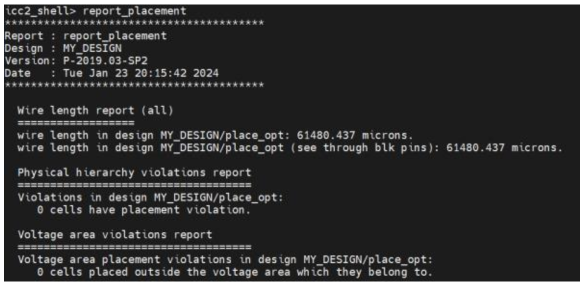
The tool's message "check legality for block design successes" suggests that the design validation process is verifying the compliance and legality of the block-level components, ensuring that the construction and configuration of the blocks align with the specified design criteria and standards.



```
check_legality for block design MY_DESIGN succeeded!

check_legality succeeded.
```

Figure 7: Design meta

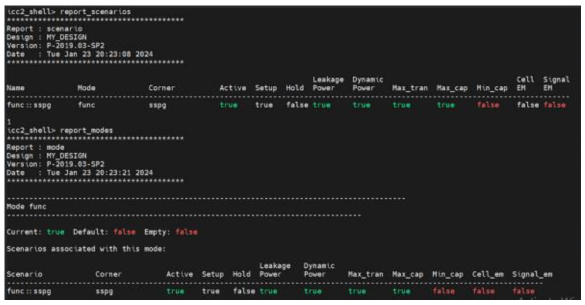
Summary of placement analysis for design MY_DESIGN, revealing wire length metrics, placement violations, and voltage area placement issues.



```
icc2_shell> report_placement
****************************************
Report : report_placement
Design : MY_DESIGN
Version: P-2019.03-SP2
Date   : Tue Jan 23 20:15:42 2024
****************************************

  Wire length report (all)
  ==================
  wire length in design MY_DESIGN/place_opt: 61480.437 microns.
  wire length in design MY_DESIGN/place_opt (see through blk pins): 61480.437 microns.

  Physical hierarchy violations report
  ====================================
  Violations in design MY_DESIGN/place_opt:
     0 cells have placement violation.

  Voltage area violations report
  ====================================
  Voltage area placement violations in design MY_DESIGN/place_opt:
     0 cells placed outside the voltage area which they belong to.
```

Figure 8: Placement Analysis Summary

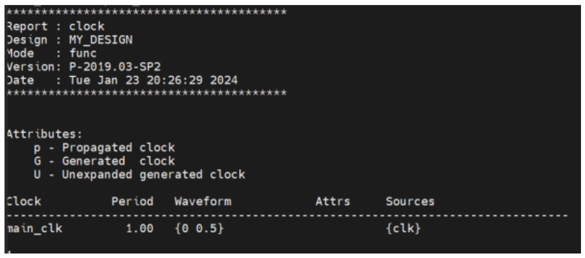
This report provides insights into the power modes and scenarios of design MY_DESIGN. The scenarios highlight power, setup, and corner attributes, with specific configurations for functional and sspg modes. The associated modes, such as Leakage, Dynamic, and Max tran, are detailed, showcasing settings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



```
icc2_shell> report_scenarios
****************************************
Report : scenario
Design : MY_DESIGN
Version: P-2019.03-SP2
Date   : Tue Jan 23 20:23:08 2024
****************************************

                                                                   Leakage  Dynamic                                Cell   Signal
Name            Mode            Corner          Active  Setup  Hold  Power    Power     Max_tran  Max_cap  Min_cap  EM     EM
---------------------------------------------------------------------------------------------------------------------------------
func::sspg      func            sspg            true    true   false true     true      true      true     false    false  false

1
icc2_shell> report_modes
****************************************
Report : mode
Design : MY_DESIGN
Version: P-2019.03-SP2
Date   : Tue Jan 23 20:23:21 2024
****************************************

------------------------------------------------------------------------------------------
Mode func
----------------------------------------------------------------------------------

Current: true  Default: false  Empty: false

Scenarios associated with this mode:

                                                 Leakage   Dynamic
Scenario          Corner          Active  Setup  Hold  Power     Power     Max_tran  Max_cap  Min_cap  Cell_em  Signal_em
------------------------------------------------------------------------------------------------------------------------
func::sspg        sspg            true    true   false true      true      true      true     false    false    false
```

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.



```
****************************************
Report : clock
Design : MY_DESIGN
Mode   : func
Version: P-2019.03-SP2
Date   : Tue Jan 23 20:26:29 2024
****************************************


Attributes:
    p - Propagated clock
    G - Generated  clock
    U - Unexpanded generated clock

Clock          Period   Waveform            Attrs     Sources
--------------------------------------------------------------------------------
main_clk         1.00   {0 0.5}                       {clk}
```

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.

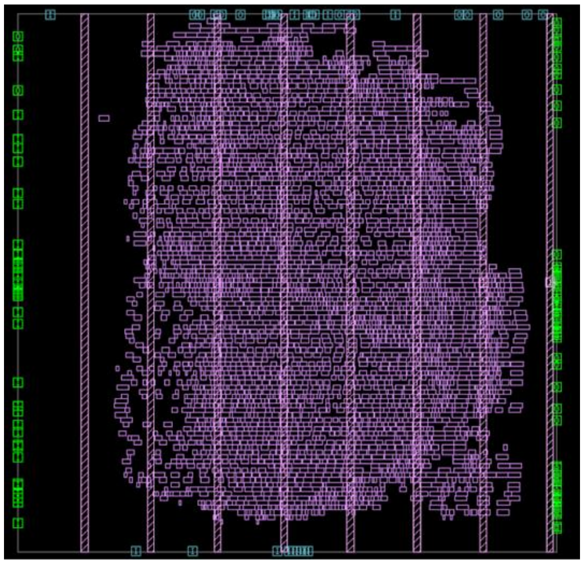
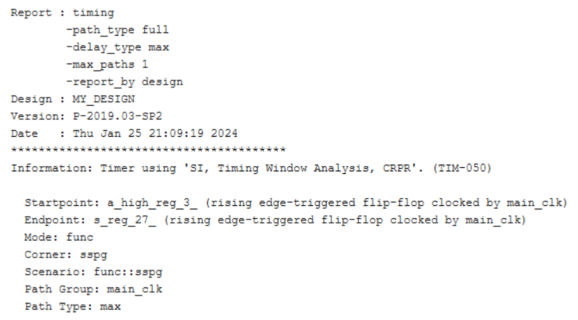


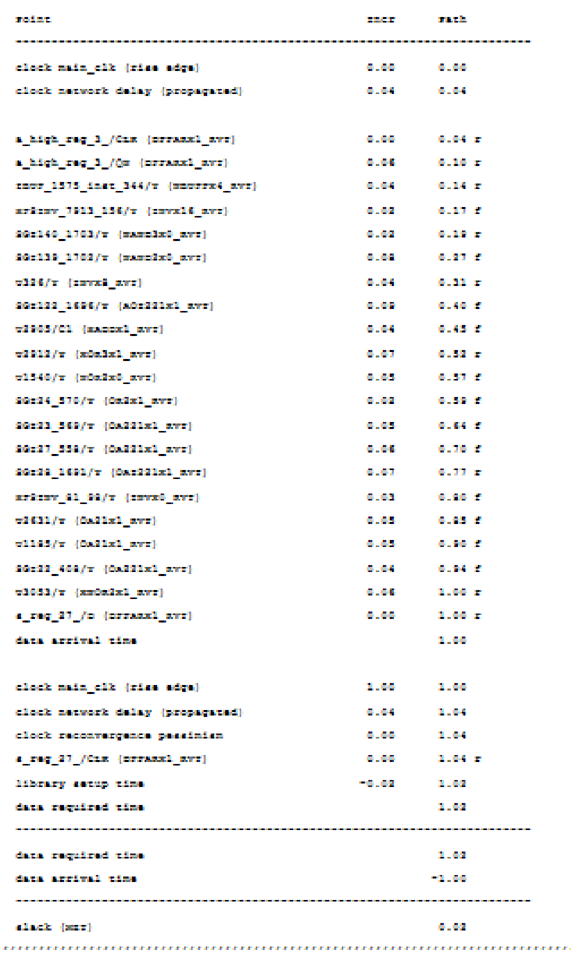
Figure 11: Design after full placement

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.



```
Report : timing
        -path_type full
        -delay_type max
        -max_paths 1
        -report_by design
Design : MY_DESIGN
Version: P-2019.03-SP2
Date   : Thu Jan 25 21:09:19 2024
****************************************
Information: Timer using 'SI, Timing Window Analysis, CRPR'. (TIM-050)

  Startpoint: a_high_reg_3_ (rising edge-triggered flip-flop clocked by main_clk)
  Endpoint: s_reg_27_ (rising edge-triggered flip-flop clocked by main_clk)
  Mode: func
  Corner: sspg
  Scenario: func::sspg
  Path Group: main_clk
  Path Type: max
```

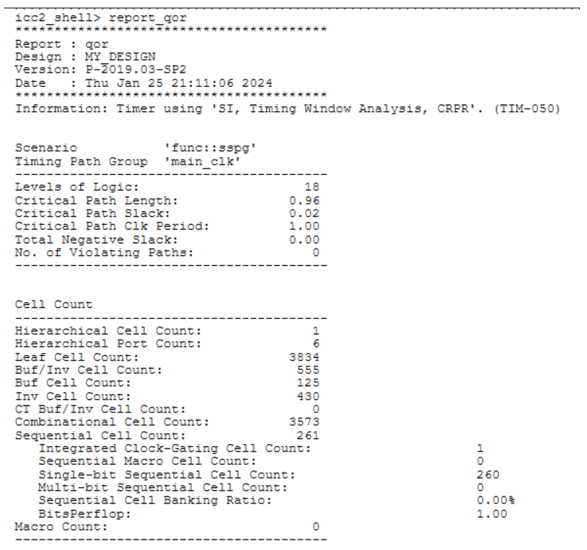
Figure 12: Critical Timing Path Analysis



```
Point                                               Incr      Path
------------------------------------------------------------------------
clock main_clk (rise edge)                          0.00      0.00
clock network delay (propagated)                    0.04      0.04

a_high_reg_3_/CLK (DFFARX1_RVT)                     0.00      0.04 r
a_high_reg_3_/Q (DFFARX1_RVT)                       0.06      0.10 r
ropt_1575_inst_344/Y (NBUFFX4_RVT)                  0.04      0.14 r
xr2inv_7913_156/Y (INVX16_RVT)                      0.02      0.17 f
ZGr140_1703/Y (NAND2X0_RVT)                         0.02      0.19 r
ZGr138_1702/Y (NAND2X0_RVT)                         0.08      0.27 f
U326/Y (INVX8_RVT)                                  0.04      0.31 r
ZGr122_1696/Y (AOI221X1_RVT)                        0.09      0.40 f
U2905/C1 (HADDX1_RVT)                               0.04      0.45 f
U2912/Y (XOR3X1_RVT)                                0.07      0.52 r
U1540/Y (XOR2X0_RVT)                                0.05      0.57 f
ZGr24_570/Y (OA2X1_RVT)                             0.02      0.59 f
ZGr23_569/Y (OA221X1_RVT)                           0.05      0.64 f
ZGr27_558/Y (OA221X1_RVT)                           0.06      0.70 f
ZGr26_1691/Y (OAI221X1_RVT)                         0.07      0.77 r
xr2inv_61_99/Y (INVX0_RVT)                          0.03      0.80 f
U2631/Y (OA21X1_RVT)                                0.05      0.85 f
U1185/Y (OA21X1_RVT)                                0.05      0.90 f
ZGr32_466/Y (OA221X1_RVT)                           0.04      0.94 f
U3053/Y (XNOR3X1_RVT)                               0.06      1.00 r
a_reg_27_/D (DFFARX1_RVT)                           0.00      1.00 r
data arrival time                                             1.00

clock main_clk (rise edge)                          1.00      1.00
clock network delay (propagated)                    0.04      1.04
clock reconvergence pessimism                       0.00      1.04
a_reg_27_/CLK (DFFARX1_RVT)                         0.00      1.04 r
library setup time                                 -0.02      1.02
data required time                                            1.02
------------------------------------------------------------------------
data required time                                            1.02
data arrival time                                            -1.00
------------------------------------------------------------------------
slack (MET)                                                   0.02
```

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

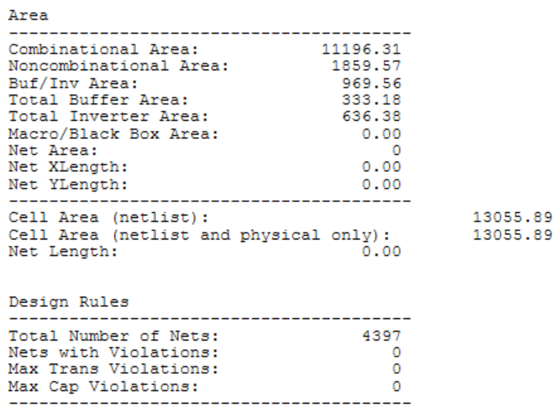
The Quality of Results (QoR) analysis for design MY_DESIGN, conducted using the 'SI, Timing Window Analysis, CRPR' timer, provides a comprehensive overview of various design metrics. Under the functional scenario 'func::sspg' and within the timing path group 'main_clk,' the critical path is identified 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.



```
icc2_shell> report_qor
****************************************
Report : qor
Design : MY_DESIGN
Version: P-2019.03-SP2
Date   : Thu Jan 25 21:11:06 2024
****************************************
Information: Timer using 'SI, Timing Window Analysis, CRPR'. (TIM-050)


Scenario           'func::sspg'
Timing Path Group  'main_clk'
----------------------------------------
Levels of Logic:                    18
Critical Path Length:             0.96
Critical Path Slack:              0.02
Critical Path Clk Period:         1.00
Total Negative Slack:             0.00
No. of Violating Paths:              0
----------------------------------------


Cell Count
----------------------------------------
Hierarchical Cell Count:             1
Hierarchical Port Count:             6
Leaf Cell Count:                  3834
Buf/Inv Cell Count:                555
Buf Cell Count:                    125
Inv Cell Count:                    430
CT Buf/Inv Cell Count:               0
Combinational Cell Count:         3573
Sequential Cell Count:             261
   Integrated Clock-Gating Cell Count:                    1
   Sequential Macro Cell Count:                           0
   Single-bit Sequential Cell Count:                      260
   Multi-bit Sequential Cell Count:                       0
   Sequential Cell Banking Ratio:                         0.00%
   BitsPerflop:                                           1.00
Macro Count:                         0
----------------------------------------
```

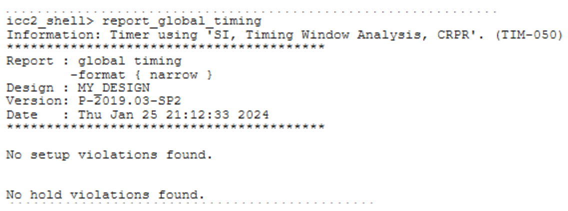
Figure 14: Quality of Results (QoR) Analysis for: Functional Scenario and Timing Path Evaluation

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



```
Area
----------------------------------------
Combinational Area:             11196.31
Noncombinational Area:           1859.57
Buf/Inv Area:                     969.56
Total Buffer Area:                333.18
Total Inverter Area:              636.38
Macro/Black Box Area:               0.00
Net Area:                              0
Net XLength:                        0.00
Net YLength:                        0.00
----------------------------------------
Cell Area (netlist):                              13055.89
Cell Area (netlist and physical only):            13055.89
Net Length:                         0.00


Design Rules
----------------------------------------
Total Number of Nets:               4397
Nets with Violations:                  0
Max Trans Violations:                  0
Max Cap Violations:                    0
----------------------------------------
```

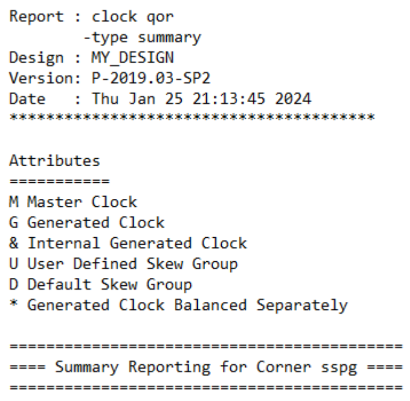
Figure 15: Area Analysis



```
icc2_shell> report_global_timing
Information: Timer using 'SI, Timing Window Analysis, CRPR'. (TIM-050)
****************************************
Report : global timing
        -format { narrow }
Design : MY_DESIGN
Version: P-2019.03-SP2
Date   : Thu Jan 25 21:12:33 2024
****************************************

No setup violations found.


No hold violations found.
```

Figure 16: Global Timing Analysis

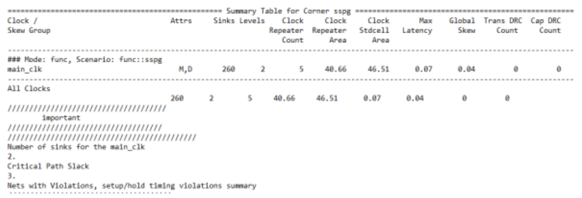
The clock Quality of Results (QoR) summary for design MY_DESIGN under the 'func::sspg' scenario and 'sspg' corner provides an insightful analysis. The main clock, denoted as 'main_clk,' exhibits 260 sinks across 2 levels, with attributes including Master Clock (M) and Default Skew Group (D). The clock 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.



```
Report : clock qor
        -type summary
Design : MY_DESIGN
Version: P-2019.03-SP2
Date   : Thu Jan 25 21:13:45 2024
****************************************

Attributes
===========
M Master Clock
G Generated Clock
& Internal Generated Clock
U User Defined Skew Group
D Default Skew Group
* Generated Clock Balanced Separately

===========================================
==== Summary Reporting for Corner sspg ====
===========================================
```

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.



```
================================================== Summary Table for Corner sspg ===================================================
Clock /                              Attrs    Sinks Levels   Clock    Clock    Clock      Max   Global Trans DRC  Cap DRC
Skew Group                                                Repeater Repeater  Stdcell  Latency    Skew     Count     Count
                                                             Count     Area     Area
-----------------------------------------------------------------------------------------------------------------------------------
### Mode: func, Scenario: func::sspg
main_clk                               M,D      260      2        5    40.66    46.51     0.07     0.04         0         0
-----------------------------------------------------------------------------------------------------------------------------------
All Clocks
                                     260      2        5    40.66    46.51     0.07     0.04         0         0
/////////////////////////////////////
        important
////////////////////////////////////
//////////////////////////////////////////////
Number of sinks for the main_clk
2.
Critical Path Slack
3.
Nets with Violations, setup/hold timing violations summary
--------------------------------------
```

Figure 18: Clock QoR Summary Table for Corner sspg.

The design verification procedure is initiated by loading the design with the placed, synthesized/routed clock tree, and signal routed design. Using the create chip_finish command, which is used for layout versus schematic (LVS), design rule checks (DRC), and final verification checks. Following the application of this command, the library has nine blocks.

The final design was as shown below and after displaying the route:

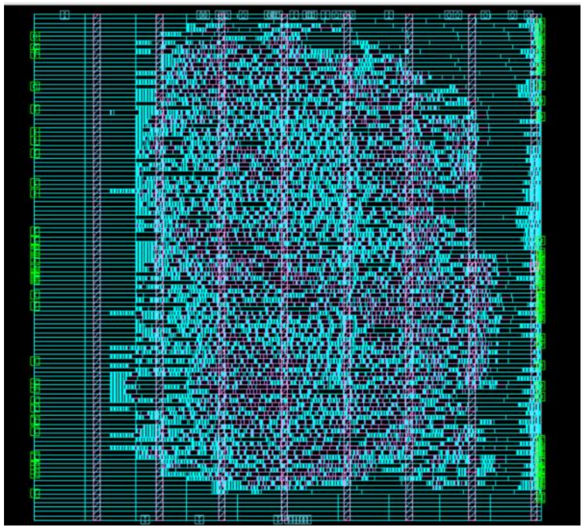


Figure 19: Layout

Identifying the sort of cell:

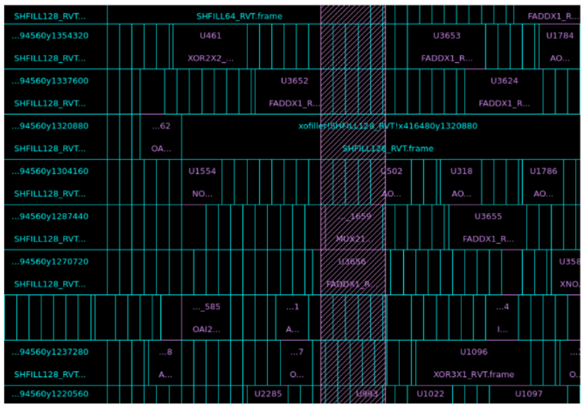
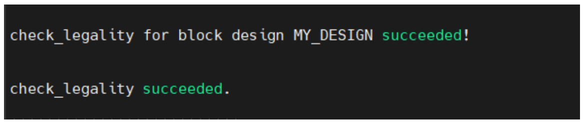


Figure 20: Check the ref_phys_view

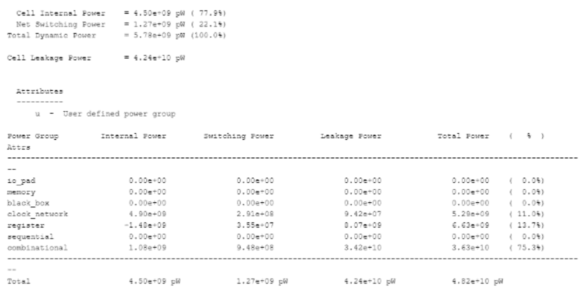
Then running the "check_legality" command to confirm that every cell has been legalized:



```
check_legality for block design MY_DESIGN succeeded!

check_legality succeeded.
```

Figure 21: Success message

Check the power:



```
  Cell Internal Power    = 4.50e+09 pW ( 77.9%)
  Net Switching Power    = 1.27e+09 pW ( 22.1%)
Total Dynamic Power      = 5.78e+09 pW (100.0%)

Cell Leakage Power       = 4.24e+10 pW


  Attributes
  ----------
      u  -  User defined power group

Power Group        Internal Power      Switching Power        Leakage Power         Total Power    (   %  )
Attrs
------------------------------------------------------------------------------------------------------------
--
io_pad                 0.00e+00            0.00e+00             0.00e+00              0.00e+00     (  0.0%)
memory                 0.00e+00            0.00e+00             0.00e+00              0.00e+00     (  0.0%)
black_box              0.00e+00            0.00e+00             0.00e+00              0.00e+00     (  0.0%)
clock_network          4.90e+09            2.91e+08             9.42e+07              5.29e+09     ( 11.0%)
register              -1.48e+09            3.55e+07             8.07e+09              6.63e+09     ( 13.7%)
sequential             0.00e+00            0.00e+00             0.00e+00              0.00e+00     (  0.0%)
combinational          1.08e+09            9.48e+08             3.42e+10              3.63e+10     ( 75.3%)
------------------------------------------------------------------------------------------------------------
--
Total                  4.50e+09 pW         1.27e+09 pW          4.24e+10 pW           4.82e+10 pW
```

Figure 22: Power

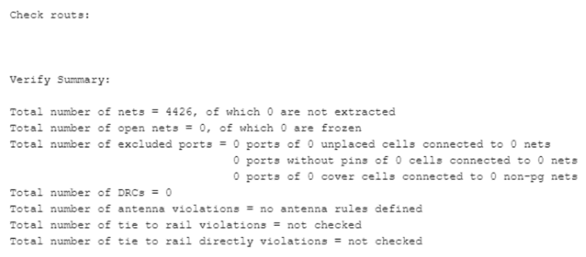
Total Dynamic Power: 5.78e+09 pW

Cell Leakage Power: 4.24e+10 pW

The power used by the cells is known as internal power, while the power used during state changes, or clock toggling, is known as switching power.

The power is divided into several areas, including combinational logic, clock networks, and registers.

Running the command "check_routes":

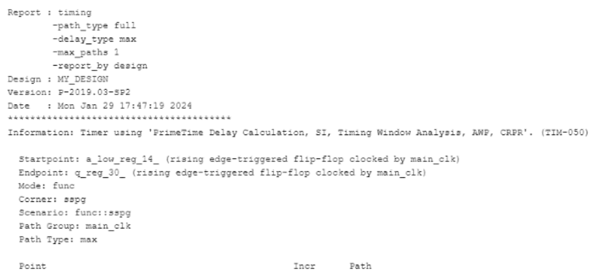


```
Check routes:



Verify Summary:

Total number of nets = 4426, of which 0 are not extracted
Total number of open nets = 0, of which 0 are frozen
Total number of excluded ports = 0 ports of 0 unplaced cells connected to 0 nets
                                 0 ports without pins of 0 cells connected to 0 nets
                                 0 ports of 0 cover cells connected to 0 non-pg nets
Total number of DRCs = 0
Total number of antenna violations = no antenna rules defined
Total number of tie to rail violations = not checked
Total number of tie to rail directly violations = not checked
```

Figure 23: Check routes

The report provides information on the via optimization status for various metal layers (VIA1 through VIA7) in the IC layout. Most of the vias have a weight of 1 and 100% optimization, meaning that all vias—connections between layers—are fully placed, optimized, and mapped, which helps to create an effective interconnect structure. An overview of the verification procedure is provided in the second graphic. It verifies that there are no frozen nets, excluded ports, or open nets, indicating that all connections have been successfully made and that there are no unmovable or unfinished components. It also demonstrates that there are no breaches of Design Rule Checks (DRCs) and that antenna and tie to rail checks are either not relevant or have not been completed. Up to the present verification step, this suggests a clean design in terms of connection and rule compliance, with some tests either postponed or deemed unnecessary. Together, these reports indicate that the IC design is moving along well, with successful routing and initial verification, even though certain infractions have not yet been completed or are awaiting final inspections.

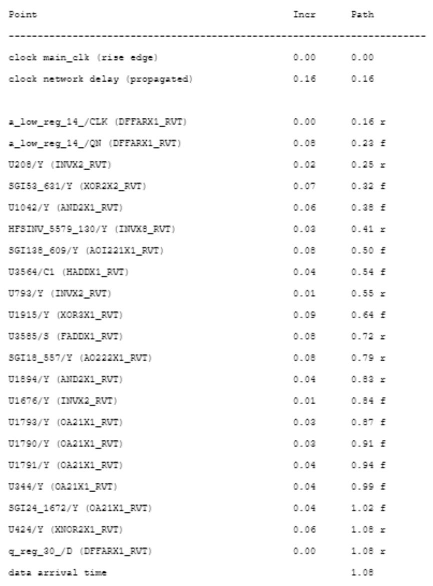
After applying the command report timing:



```
Report : timing
        -path_type full
        -delay_type max
        -max_paths 1
        -report_by design
Design : MY_DESIGN
Version: P-2019.03-SP2
Date   : Mon Jan 29 17:47:19 2024
****************************************
Information: Timer using 'PrimeTime Delay Calculation, SI, Timing Window Analysis, AWP, CRPR'. (TIM-050)

  Startpoint: a_low_reg_14_ (rising edge-triggered flip-flop clocked by main_clk)
  Endpoint: q_reg_30_ (rising edge-triggered flip-flop clocked by main_clk)
  Mode: func
  Corner: sspg
  Scenario: func::sspg
  Path Group: main_clk
  Path Type: max

  Point                                                 Incr      Path
```

Figure 24: Report timing [1]

The crucial path in the design is displayed in the timing report. With a slack (MET) of 0.01 the design satisfies timing constraints. The data arrival time is slightly earlier than the needed time, resulting in positive slack. The report breaks out the delay at each critical path point.

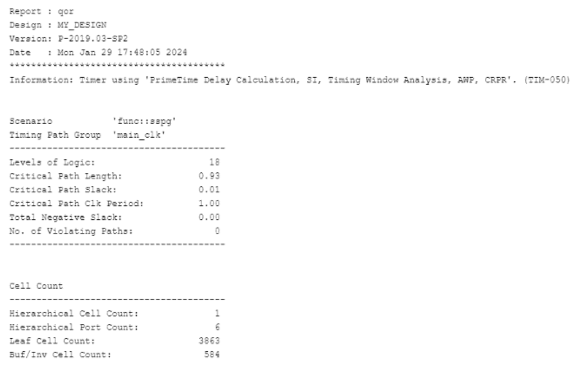


```
Point                                              Incr      Path
------------------------------------------------------------------------
clock main_clk (rise edge)                         0.00      0.00
clock network delay (propagated)                   0.16      0.16


a_low_reg_14_/CLK (DFFARX1_RVT)                    0.00      0.16 r
a_low_reg_14_/QN (DFFARX1_RVT)                     0.08      0.23 f
U208/Y (INVX2_RVT)                                 0.02      0.25 r
SGI53_631/Y (XOR2X2_RVT)                           0.07      0.32 f
U1042/Y (AND2X1_RVT)                               0.06      0.38 f
HFSINV_5579_130/Y (INVX8_RVT)                      0.03      0.41 r
SGI138_609/Y (AOI221X1_RVT)                        0.08      0.50 f
U3564/C1 (HADDX1_RVT)                              0.04      0.54 f
U793/Y (INVX2_RVT)                                 0.01      0.55 r
U1915/Y (XOR3X1_RVT)                               0.09      0.64 f
U3585/S (FADDX1_RVT)                               0.08      0.72 r
SGI18_557/Y (AO222X1_RVT)                          0.08      0.79 r
U1894/Y (AND2X1_RVT)                               0.04      0.83 r
U1676/Y (INVX2_RVT)                                0.01      0.84 f
U1793/Y (OA21X1_RVT)                               0.03      0.87 f
U1790/Y (OA21X1_RVT)                               0.03      0.91 f
U1791/Y (OA21X1_RVT)                               0.04      0.94 f
U344/Y (OA21X1_RVT)                                0.04      0.99 f
SGI24_1672/Y (OA21X1_RVT)                          0.04      1.02 f
U424/Y (XNOR2X1_RVT)                               0.06      1.08 r
q_reg_30_/D (DFFARX1_RVT)                          0.00      1.08 r
data arrival time                                            1.08
```

Figure 25: Report timing [2]

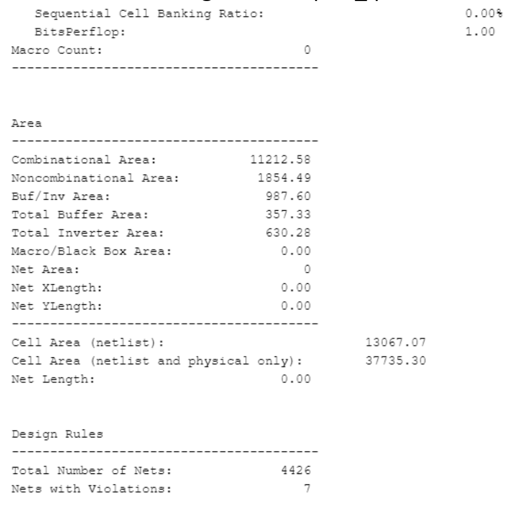
The capacitance scaling factors are universally set to 1, indicating that no scaling is performed to the extracted data, and the design's extraction choices are fully defined. The verification process for the corner scenario'sspg' has determined the late and early capacitance thresholds and ratios. The late_ccap_ratio value of 0.03 indicates sensitivity to capacitive effects. Notably, no cells are left out of the study; 20,767 cells are marked for time consideration in both the early and late scenarios. With the data arrival time precisely meeting the required time of 1.00 time unit, the timing path analysis for the clock'main_clk' yields a clock network delay of 0.02 time units. This results in a zero slack (MET), indicating that the timing for this design path is accurate and falls within the expected constraints for proper functionality.

After run the command report_qor:



```
Report : qor
Design : MY_DESIGN
Version: P-2019.03-SP2
Date   : Mon Jan 29 17:48:05 2024
****************************************
Information: Timer using 'PrimeTime Delay Calculation, SI, Timing Window Analysis, AWP, CRPR'. (TIM-050)


Scenario           'func::sspg'
Timing Path Group  'main_clk'
----------------------------------------
Levels of Logic:                      18
Critical Path Length:               0.93
Critical Path Slack:                0.01
Critical Path Clk Period:           1.00
Total Negative Slack:               0.00
No. of Violating Paths:                0
----------------------------------------


Cell Count
----------------------------------------
Hierarchical Cell Count:               1
Hierarchical Port Count:               6
Leaf Cell Count:                    3863
Buf/Inv Cell Count:                  584
```

Figure 26: Report_qor 1



```
   Sequential Cell Banking Ratio:                      0.00%
   BitsPerflop:                                        1.00
 Macro Count:                        0
 ----------------------------------------


 Area
 ----------------------------------------
 Combinational Area:           11212.58
 Noncombinational Area:         1854.49
 Buf/Inv Area:                   987.60
 Total Buffer Area:              357.33
 Total Inverter Area:            630.28
 Macro/Black Box Area:             0.00
 Net Area:                            0
 Net XLength:                      0.00
 Net YLength:                      0.00
 ----------------------------------------
 Cell Area (netlist):                         13067.07
 Cell Area (netlist and physical only):       37735.30
 Net Length:                       0.00


 Design Rules
 ----------------------------------------
 Total Number of Nets:             4426
 Nets with Violations:                7
```

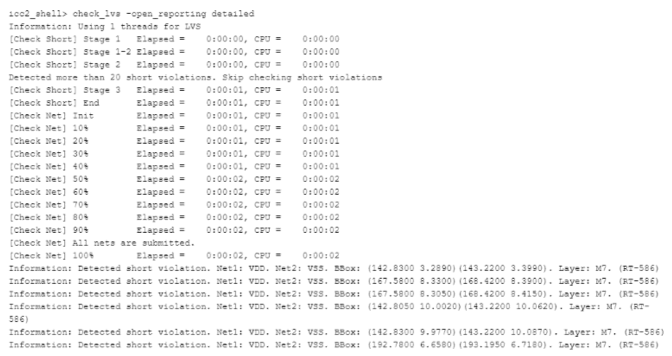
Figure 27: Report_qor 2

Critical Path Length: 0.93

Critical Path Slack: 0.01

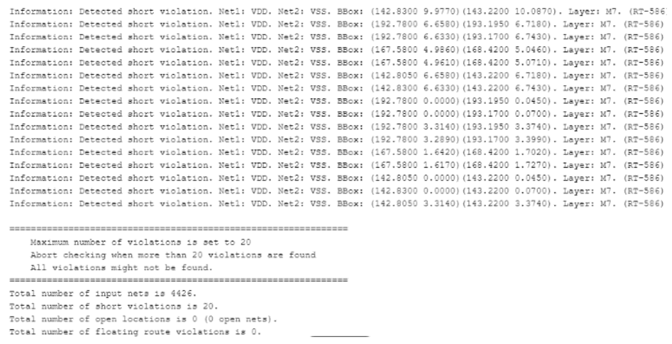
The design has a positive slack, indicating that it meets the timing requirements. Cell count, area, and design rule violations are also provided.

Next, by running the command "check_lvs -open_reporting detailed," which verifies that the layout adheres to the given design's schematic, a routed, clock-synthesized, and schematically arranged design is produced.



```
icc2_shell> check_lvs -open_reporting detailed
Information: Using 1 threads for LVS
[Check Short] Stage 1   Elapsed =    0:00:00, CPU =    0:00:00
[Check Short] Stage 1-2 Elapsed =    0:00:00, CPU =    0:00:00
[Check Short] Stage 2   Elapsed =    0:00:00, CPU =    0:00:00
Detected more than 20 short violations. Skip checking short violations
[Check Short] Stage 3   Elapsed =    0:00:01, CPU =    0:00:01
[Check Short] End       Elapsed =    0:00:01, CPU =    0:00:01
[Check Net] Init        Elapsed =    0:00:01, CPU =    0:00:01
[Check Net] 10%         Elapsed =    0:00:01, CPU =    0:00:01
[Check Net] 20%         Elapsed =    0:00:01, CPU =    0:00:01
[Check Net] 30%         Elapsed =    0:00:01, CPU =    0:00:01
[Check Net] 40%         Elapsed =    0:00:01, CPU =    0:00:01
[Check Net] 50%         Elapsed =    0:00:02, CPU =    0:00:02
[Check Net] 60%         Elapsed =    0:00:02, CPU =    0:00:02
[Check Net] 70%         Elapsed =    0:00:02, CPU =    0:00:02
[Check Net] 80%         Elapsed =    0:00:02, CPU =    0:00:02
[Check Net] 90%         Elapsed =    0:00:02, CPU =    0:00:02
[Check Net] All nets are submitted.
[Check Net] 100%        Elapsed =    0:00:02, CPU =    0:00:02
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (142.8300 3.2890)(143.2200 3.3990). Layer: M7. (RT-586)
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (167.5800 8.3300)(168.4200 8.3900). Layer: M7. (RT-586)
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (167.5800 8.3050)(168.4200 8.4150). Layer: M7. (RT-586)
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (142.8050 10.0020)(143.2200 10.0620). Layer: M7. (RT-
586)
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (142.8300 9.9770)(143.2200 10.0870). Layer: M7. (RT-586)
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (192.7800 6.6580)(193.1950 6.7180). Layer: M7. (RT-586)
```

Figure 28: Details report [1]



```
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (142.8300 9.9770)(143.2200 10.0870). Layer: M7. (RT-586
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (192.7800 6.6580)(193.1950 6.7180). Layer: M7. (RT-586)
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (192.7800 6.6330)(193.1700 6.7430). Layer: M7. (RT-586)
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (167.5800 4.9860)(168.4200 5.0460). Layer: M7. (RT-586)
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (167.5800 4.9610)(168.4200 5.0710). Layer: M7. (RT-586)
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (142.8050 6.6580)(143.2200 6.7180). Layer: M7. (RT-586)
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (142.8300 6.6330)(143.2200 6.7430). Layer: M7. (RT-586)
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (192.7800 0.0000)(193.1950 0.0450). Layer: M7. (RT-586)
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (192.7800 0.0000)(193.1700 0.0700). Layer: M7. (RT-586)
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (192.7800 3.3140)(193.1950 3.3740). Layer: M7. (RT-586)
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (192.7800 3.2890)(193.1700 3.3990). Layer: M7. (RT-586)
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (167.5800 1.6420)(168.4200 1.7020). Layer: M7. (RT-586)
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (167.5800 1.6170)(168.4200 1.7270). Layer: M7. (RT-586)
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (142.8050 0.0000)(143.2200 0.0450). Layer: M7. (RT-586)
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (142.8300 0.0000)(143.2200 0.0700). Layer: M7. (RT-586)
Information: Detected short violation. Net1: VDD. Net2: VSS. BBox: (142.8050 3.3140)(143.2200 3.3740). Layer: M7. (RT-586)

=================================================================
    Maximum number of violations is set to 20
    Abort checking when more than 20 violations are found
    All violations might not be found.
=================================================================
Total number of input nets is 4426.
Total number of short violations is 20.
Total number of open locations is 0 (0 open nets).
Total number of floating route violations is 0.
```

Figure 29: Details Report [2]

LVS (Layout vs. Schematic) Checking:

Between the ground (VSS) and power (VDD) nets, there have been a few short violations found.

The bounding box coordinates of each violation, together with the metal layer where the short occurred, are provided.

Once the instrument reaches the maximum of 20 infractions, it stops checking.

Explanation:

Timing requirements appear to be met by the design with positive slack.

The configuration has several short violations, particularly in the area between the ground and power nets.

Twenty short violations were found by the LVS tool, and because of the specified limit, the checking operation was terminated.

In conclusion, the analysis of the digital design shows promising results in a number of areas. The design performs well, meeting time requirements with a positive slack. Leakage and dynamic power components define power consumption. Nevertheless, there are 20 short violations between the power and ground nets found in the layout versus schematic (LVS) assessment, which calls for further examination and correction. Resolving these layout problems should improve the overall quality and dependability of the design.

IV. VERIFICATION:

The simulation after the running the codes in Appendix 2.1 and 2.2:

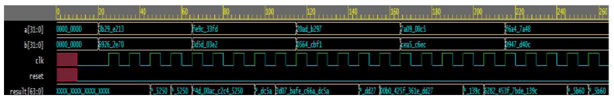


Figure 30: Verification Simulation

The above waveform shows the simulation results of a pipelined multiplier implemented in Verilog. The pipelined multiplier is a type of digital circuit used for multiplying two binary numbers in a way that improves the throughput by overlapping the execution of several multiplication operations.

In the Verilog code, the multiplication operation is broken down into smaller steps, which are then executed in stages or "pipeline stages." Here's a breakdown of the pipelined stages as per your code:

First cycle: The input numbers a and b are split into their lower and higher 16 bits (a_low, a_high, b_low, b_high). This splitting is a preparation for partial products that will be computed in the following cycles.

Second cycle: Four partial products are computed: p (a_low * b_low), q (a_low * b_high), r (a_high * b_low), and s (a_high * b_high). These partial products are the results of multiplying the split parts of the original input numbers.

Third cycle: The lower 16 bits of the result are assigned from the lower 16 bits of p. The sum of the upper 16 bits of p and the lower 16 bits of both q and r are computed, with any carry-out stored in c1.

Fourth cycle: The next part of the result (bits 47:32) is computed by adding the upper 16 bits of q and r, the lower 16 bits of s, and the carry-in c1 from the previous addition. Again, any carry-out is stored, this time in c2.

Fifth cycle: The final part of the result (bits 63:48) is computed by adding the upper 16 bits of s and the carry-in c2 from the previous operation.

The reset logic is also present to initialize or clear all the registers when the reset signal is high. This is common practice to ensure the system starts with a known state upon activation or after an error.

The testbench provided with the code generates random numbers for a and b, waits for the pipeline to process these numbers through its stages, and then checks if the result matches the expected multiplication of a and b. If the multiplication is correct, the test passes; otherwise, it fails.

From the waveform, we can see the clock (clk) signal oscillating, indicating the passage of time in the simulation. The reset signal appears to be a short pulse, probably to initialize the system. The a and b signals change values at certain points, which triggers the calculation in the pipeline. The result signal changes accordingly after the appropriate number of clock cycles, as per the pipelining stages.

Generally, the waveform should show a staggered update to the result signal as the different pipeline stages complete, which corresponds to the expected behavior of a pipelined multiplier.

V. 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.

VI. Appendix

Appendix1:

```
// 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
```

Appendix 2.1:

```
// Code your design here

// 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
```

Appendix 2.2:

```
// Code your testbench here
// or browse Examples


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).