128-bit Integer ALU

Saima Absar
Department of CSCE
University of Arkansas, Fayetteville
sa059@uark.edu

***Abstract*—This article demonstrates the semi-custom design flow of a 128-bit integer ALU that performs arithmetic and bit-wise logical operations on integer binary numbers. At first, RTL netlist of the design is written in Verilog. It is then synthesized to a gate-level netlist using Synopsis' Design Compiler. The synthesized netlist is, afterwards, used for standard cell placement and routing with the help of the Cadence Innovus tool. The routed design is checked for DRC violation using Calibre and finally Analysis is done in Primetime. All the netlist has been simulated in ModelSim.**

***Index Terms*—RTL, GTL, Testbench, Synthesis, Placement and Routing, Analysis, Optimization**

I. Introduction

An Arithmetic Logic Unit (ALU) is a combinational logic circuit that can perform basic arithmetic and logical operations on integers. In this project a 128-bit ALU is designed using the semi-custom design flow. All the operations performed by this ALU is summarized in Fig. 1.

M = 0, Logical Bitwise Operations

S1	S0	Function	Comment
0	0	Fi = Ai	Input Ai transferred to output
0	1	Fi = not Ai	Complement of Ai transferred to outp
1	0	Fi = Ai xor Bi	Compute XOR of Ai, Bi
1	1	Fi = Ai xnor Bi	Compute XNOR of Ai, Bi

M = 1, C0 = 0, Arithmetic Operations

0	0	F = A	Input A passed to output
0	1	F = not A	Complement of A passed to output
1	0	F = A plus B	Sum of A and B
1	1	F = (not A) plus B	Sum of B and complement of A

M = 1, C0 = 1, Arithmetic Operations

0	0	F = A plus 1	Increment A
0	1	F = (not A) plus 1	Twos complement of A
1	0	F = A plus B plus 1	Increment sum of A and B
1	1	F = (not A) plus B plus 1	B minus A

Fig. 1: 128-bit ALU operations

II. RTL netlist

A. Synthesizable Verilog Code

The Register Transfer Level (RTL) netlist for this design is written in verilog as shown in Fig. 2. The module **ALU** has six ports: the two input data ports A, B and output F are 128 bits each, the opcode S of 2-bit and the selector pins M and C. The value of M determines whether arithmetic (M = 1) or logical bit-wise operation (M = 0) will be performed, which is implemented here using if-else statement. There are two types of arithmetic operations here, decided by C0, also implemented by if-else. The operation of the op-code S is implemented using the case statement in the netlist.

```
module ALU(A, B, M, S, C0, F);
    input   [127:0] A,B;
    input   M, C0;
    input   [1:0] S;
    output reg  [127:0] F;

    always@(A, B, M, S, C0)
    begin
        if(M==0) begin
            case(S)
                2'b00: F = A;
                2'b01: F = ~A;
                2'b10: F = A ^ B;
                2'b11: F = A ~^ B;
                default: F = 2'bx;
            endcase
        end
        else if(M==1)
            if(C0==0) begin
                case(S)
                    2'b00: F = A;
                    2'b01: F = ~A;
                    2'b10: F = A + B;
                    2'b11: F = ~A + B;
                    default: F = 2'bx;
                endcase
            end
            else if(C0==1) begin
                case(S)
                    2'b00: F = A + 1;
                    2'b01: F = ~A + 1;
                    2'b10: F = A + B + 1;
                    2'b11: F = ~A + B + 1;
                    default: F = 2'bx;
                endcase
                end
```

Fig. 2: RTL netlist, ALU_128.v

B. Verilog Testbench

A testbench, also written in Verilog, is used to verify the functionality of the design. In the testbench, the inputs A and B are assigned random values at appropriate time delays. All the combinations of the M, C0 and S pins are used to verify the output value for all the possible operations of the ALU. The output is analyzed and verified using ModelSim. The waveform generated in ModelSim with all the possible operations is shown in Fig. 3. From the waveform, it can be verified that when the input values of A and B are 5 and 3 respectively, with M = 1, C0 = 1 and S = 2, according to Fig. 1 the ALU performs A+B, so the result F is 13, which is as expected.

Fig. 3: Simulation Waveform of RTL netlist

TABLE I: Synthesis Report Summary

Timing	
Data Required Time	1.00 ns
Data Arrival Time	-1.00 ns
Slack (MET)	**0.00 ns**
Area Group	
Number of ports	388
Number of nets	8424
Number of cells	8159
Number of Combinational Cells	8159
Total cell area	**18007.509883** um^2
Power Group (1GHz)	Combinational
Internal Power	3.6851 mW
Switching Power	2.5928 mW
Leakage Power	7.2897e+04 mW
Total	**6.3508 mW**

III. Synthesis

In this project, FreePDK45 is used as the target library. The synthesis was done following the standard steps as given in [2]. In the tcl script written for this sysnthesis, at first, the target and link library is set to "gscl45nm.db" and the symbol library to "generic.sdb". Then, the verilog netlist is analyzed, elaborated and linked. Since this is a combinational circuit, a virtual clock of time period of 1ns is created. The compilation is done with 'Compile Ultra' for optimization. A combined report for power, area and timing is generated and saved.
At this step, several clock periods was used to check if slack was met after synthesis, but 1 ns was the best producing a slack of 0 ns. Table I summarizes the report generated after synthesis. The generated gate-level netlist was simulated in Modelsim to verify it's functionality. Fig. 4 shows the waveform where it can be seen that the gate delays are in effect.

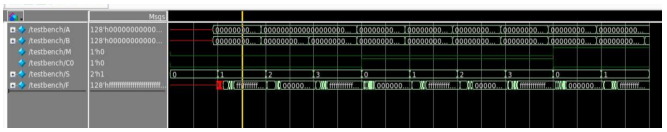
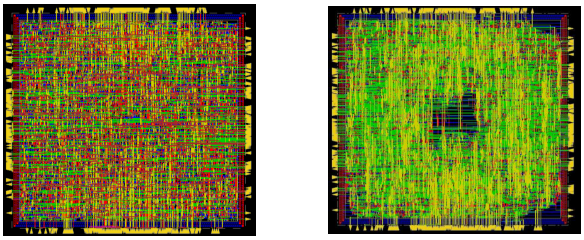


Fig. 4: Post Synthesis netlist simulation

IV. Placement and Routing

Placement and Routing is done with a floorplan specification that has an Aspect ratio of 1, Core uitilization of 0.75 and Core to IO Boundary of 5 on each side. At first, the placement was done without I/O pin constraint, but after routing it was seen that there are a lot of DRC violations. This is because, the circuit has many I/O pins (388 pins) and without any constraint some of them were placed too close to the cells which violated minimum spacing rules. In order to avoid this the I/O pins were constrained to a minimum spacing of 3 tracks with the command:
"setPinConstraint -pin * -layer {M3 M4} -spacing 3".
Fig. 5a show the physical view of the layout after placement with I/O pins contrained to Metal layers 3 and 4.



(a) Placement with I/O pins (b) Routing

Fig. 5: Placement and Routing

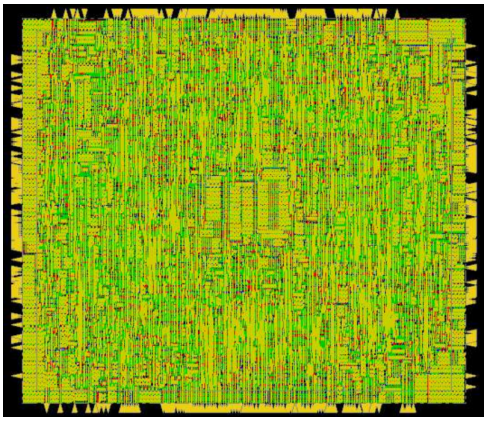


Fig. 6: Layout after sign off

As mentioned before, this is a combinational circuit, so CTS is not applicable. So after Pre-CTS optimization, the next step is routing. NanoRouting was perfomed with 4 Metal layers and 'Fix Antenna' turned off. The layout after routing is shown in Fig. 5b. It can be seen in the layout that there is an empty space in the middle, this is probably because the cells are drawn towards the i/o pins in order to minimize the wire length. After Post-Routing, filler cells and Metal Fills were inserted to 4 metal layers using the 'metalfill.cmd' file available on the course website. All the designs have been saved at each step and the timing reports were generated. After Timing sign-off, the summary report, netlist, parasitic extraction, and GDS were generated and saved. A summary of the report is depicted in Table I. It can be seen in the table that there is a negetive slack of arount 0.4 ns, but at this stage it was ignored since timing analysis will be done more accurately in Primetime. The physical view of the circuit layout after signoff is shown in Fig. 6. The ModelSim simulation of the generated netlist is also shown for verification, in Fig. 7.

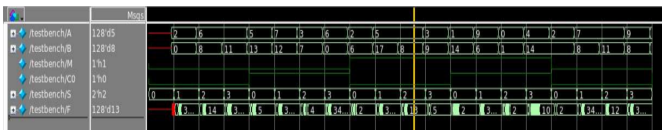


Fig. 7: Post Routing netlist simulation

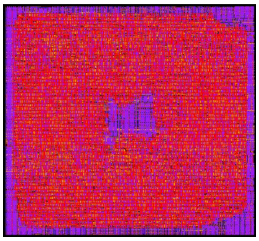
TABLE II: Report Summary

General Design Information	
Design Status	Routed
Design Name	ALU
# Instances	11398
# Std Cells	11398
# Pads	0
# Nets	8829
# IO Pins	388
Floorplan/ Placement Information	
Total area of Standard Cells	20331.484 um^2
Core Width	153.52 um
Core Height	163.4 um
Total area of Core	23510.053 um^2
Total area of Chip	26761.652 um^2
Effective Utilization	8.6480e-01
Number of Cell Rows	62
Wire Length Distribution	
Total metal 1 wire length	2088.2700 um
Total metal 2 wire length	26466.2400 um
Total metal 3 wire length	38402.5050 um
Total metal 4 wire length	20023.9650 um
Total wire length	86980.9800 um
Average wire length/net	9.8517 um
Final WNS	- 0.396 ns
Final TNS	- 18.921 ns

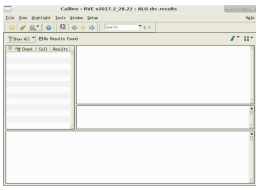
V. Design Ananlysis

A. DRC Verification

Before timing Analysis with Primetime, the routed circuit was verified for DRC violation using Calibre. The design layout and DRC-clean report is depicted in Fig 8.



(a) Design Layout showing all layers



(b) DRC verification

Fig. 8: Design Analysis

B. Timing Analysis by Primetime

The report generated after Primetime analysis is depicted in TableI. As mentioned before, the gscl45nm.db library was used as the target library, and a virtual clock was created since this is a combinational circuit. The maximum delay was set equal to the clock period which is 1 ns. The switching activity of the input was set with a static probability of 50% and toggle rate of 15%. After Report generation, it is seen that the slack is met.

VI. Conclusion

In this report, the design of 128-bit integer ALU using semi-custom design flow is described in details. As shown in the report summary of Table II, the ALU, designed in this project, consumes around 50% of power in routing and the rest 50% in the cells at a frequency of 1GHz. ALthough a negetive slack was shown in the report generated by Innovus, the final report of Primetime verifies that slack is met.

TABLE III: Primetime Report Summary

Frequency (period)	**1GHz (1.0 ns)**
WNS	**0.03 ns**
Power Group	Combinational
Internal Power	5.314e-03 W
Switching Power	4.865e-03 W
Leakage Power	8.288e-05 W
Total	0.0103 W
Net Switching	4.865e-03 W (47.41%)
Cell Internal	5.314e-03 W (51.79%)
Cell Leakage	8.288e-05 W (0.81%)
Total	**0.0103 W (100%)**

Acknowledgment

Thanks to Dr. Yarui Peng for teaching this course so efficiently, and Imam Al-Razi for patiently helping us with the labs.

References

[1] Dr. Yarui Peng, 2019, 3. Synthesizable Verilog, lecture notes, Department of CSCE, University of Arkansas, delivered 30 September 2019.

[2] Dr. Yarui Peng, 2019, 4. Design Compiler, lecture notes, Department of CSCE, University of Arkansas, delivered 8 October 2019.

[3] Dr. Yarui Peng, 2019, 5. Innovus, lecture notes, Department of CSCE, University of Arkansas, delivered 29 October 2019.

[4] Dr. Yarui Peng, 2019, 6. Primetime, lecture notes, Department of CSCE, University of Arkansas, delivered 14 November 2019.