Design of local minima/maxima circuit

Dipanjan Swarnakar

ID: 021172010
Dept. of EEE
United International University
Dhaka, Bangladesh
dswarnakar172010@bseee.uiu.ac.bd

***Abstract*— Digital logic circuit (DLC) diagrams form one of the primary steps in the design process of ASIC design. So, it is vital to convert logic diagram into Register-Transfer Level (RTL) language such as Verilog code. Then it will be flow into simulation medium to visualize the operation of the circuit diagram.**

Keywords— max, min, acc, clear, clk.

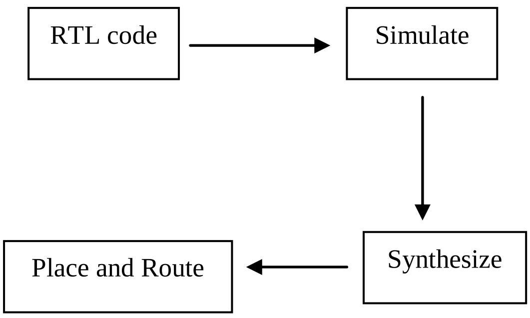
I. Introduction

The purpose of this project is to design a simple digital circuit that accumulates and finds the maximum and minimum of a series of input numbers.

II. Method of Simulation

Operation of the chip begins by setting clear=1 for at least one clock cycle to clear all three outputs. The output acc is the accumulated sum, max is the current maximum, and min is the current minimum of all inputs since the last clearing operation.The design will run at 1 GHz frequency. It's clean of violations,area is minimized;have balanced clock tree.

III. Algorithm and Discussion



A. RTL Code

I design the RTL code as per my outcomes and run it on cadance software. Then fix some errors.

B. Simulate

After fixing the errors, the simulation part comes up(Fig.1).

C. Synthesize and Place and route

Synthesis means 'to generate'. It is a step to generate circuit hardware in cadence. By synthesize I generate netlist from a circuit design model. By using the netlist I managed to generate its equivalent layout view (Fig.4).

D. Figures And Results

a) It shows the simulation from the code I design .

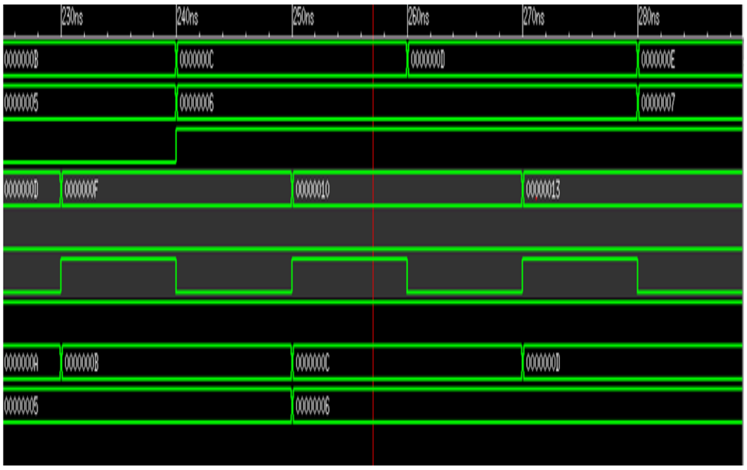
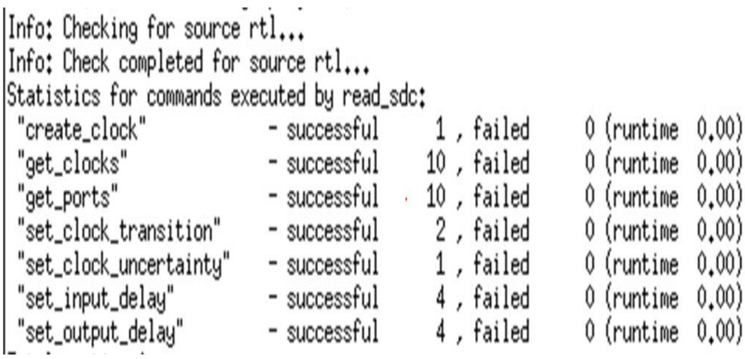


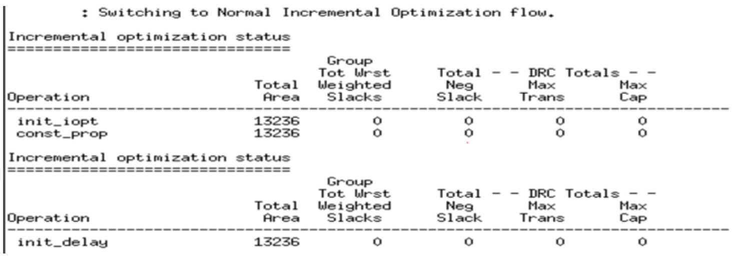
Fig. 1. Simulation of minima /maxima circuit

b)It shows the synthesize part.



```
Info: Checking for source rtl...
Info: Check completed for source rtl...
Statistics for commands executed by read_sdc:
 "create_clock"              - successful      1 , failed      0 (runtime  0.00)
 "get_clocks"                - successful     10 , failed      0 (runtime  0.00)
 "get_ports"                 - successful     10 , failed      0 (runtime  0.00)
 "set_clock_transition"      - successful      2 , failed      0 (runtime  0.00)
 "set_clock_uncertainty"     - successful      1 , failed      0 (runtime  0.00)
 "set_input_delay"           - successful      4 , failed      0 (runtime  0.00)
 "set_output_delay"          - successful      4 , failed      0 (runtime  0.00)
```

Fig.2: Synthesize output



```
        : Switching to Normal Incremental Optimization flow.
Incremental optimization status
===============================
                                    Group
                                   Tot Wrst     Total - - DRC Totals - -
                          Total   Weighted       Neg      Max       Max
Operation                  Area    Slacks       Slack    Trans      Cap
-------------------------------------------------------------------------
 init_iopt                13236        0           0        0         0
 const_prop               13236        0           0        0         0
Incremental optimization status
===============================
                                    Group
                                   Tot Wrst     Total - - DRC Totals - -
                          Total   Weighted       Neg      Max       Max
Operation                  Area    Slacks       Slack    Trans      Cap
-------------------------------------------------------------------------
 init_delay               13236        0           0        0         0
```

Fig.3:Normal increamental optimization flow.

c) It shows the place and route part.

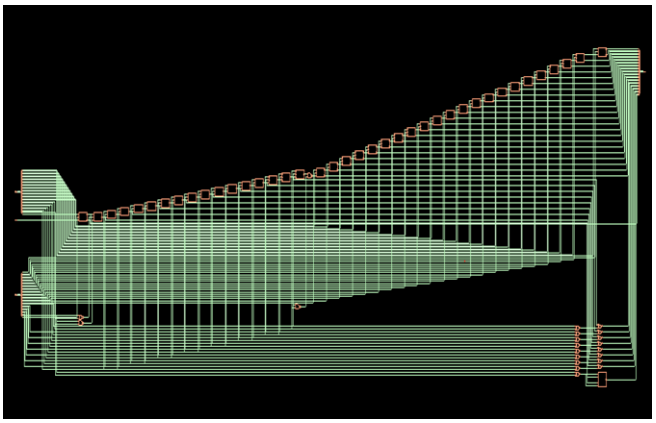


Fig.4:Scamatic from RTL code

IV. Conclusion

This design can simply accumulate the sum also can find the maximum and minimum value of series input numbers. Also this designed circuit runs at 1 GHz frequency. It's clean of violations, minimizing the area; have balanced clock tree.

Acknowledgment

The author learnt valuable information on design local minima/maxima circuit through the EEE442 course at United International University.

References

[1] Lab sheets

Codes

1)

```
module proj (acc,cout,a,b,cin,clk,clear,max,min);
input [31:0]a,b;
input cin,clk,clear;
output reg [31:0]acc,max,min;
output reg cout;
reg[31:0]x,y;
reg c;
always @(posedge clk)begin
if(a>b)
max<=a;
else max<=b;
if(a>b)
min<=b;
else
min<=a;
c<=cin;
x<=a;
y<=b;
if(~clear)begin
acc<=0;
cout<=0;
end
else {cout,acc}<=(x+y+c);
end
endmodule
```

2)

```
module proj_tb;
reg [31:0]a,b;
reg cin,clk,clear;
wire [31:0]acc,max,min;
wire cout;
proj inst0(
.a(a),
.b(b),
.cin(cin),
.acc(acc),
.cout(cout),
.clk(clk),
.clear(clear),
.max(max),
.min(min)
);
initial begin
a=0; b=0; cin=0; clk=0; clear=0;
#10 clear=1;
#5000 $finish;
end
always #10 clk=~clk;
always #20 a=a+1;
always #40 b=b+1;
always #80 a=a-1;
always #160 b=b-1;
endmodule
```