MINI PROJECT REPORT

TOPIC: ALU DESIGN USING 4-BIT ADDER AND 4-BIT SUBTRACTOR

Submitted By:

ANANYA JAIN

Internship cum Training program

VLSI Expert Private Limited

June-July 2024

PROBLEM STATEMENT:

Create an rtl and synthesis it using DC for an ALU with the following specification:

- 0 0 -> alu is completely off

- 0 1 -> 4-bit full adder (using one 1 bit full adder)

- 1 0 -> 4-bit full sub (depends how you want to design)

- 1 1 -> No Operation

- Use Flipflop to control all inputs and output, so that it can be synchronized properly. (All input the data, output, carry_in carry_out, control input

- use any of the flip flop and time period according to you

Outcome:

synthesize the gate-level netlist.

Make a report also for each step with proper schematic and rtl and waveforms and code.

- **RTL CODE ->**

(Tool: VNC Terminal)

```
// Module for a single-bit full adder.
module bit_adder (
    input x, y, carry_in,
    output sum_out, carry_out
);
    assign {carry_out, sum_out} = x + y + carry_in;
endmodule

// Module for a 4-bit ripple carry adder
module four_bit_adder_module (
    input [3:0] input_x, input_y,
    output [3:0] sum_result,
    output carry_out
```

```
);
    wire carry1, carry2, carry3;
    bit_adder adder0 (.x(input_x[0]), .y(input_y[0]), .carry_in(1'b0), .sum_out(sum_result[0]), .carry_out(carry1));
    bit_adder adder1 (.x(input_x[1]), .y(input_y[1]), .carry_in(carry1), .sum_out(sum_result[1]), .carry_out(carry2));
    bit_adder adder2 (.x(input_x[2]), .y(input_y[2]), .carry_in(carry2), .sum_out(sum_result[2]), .carry_out(carry3));
    bit_adder adder3 (.x(input_x[3]), .y(input_y[3]), .carry_in(carry3), .sum_out(sum_result[3]), .carry_out(carry_out));
endmodule

// Module for a 4-bit ripple carry subtractor
module four_bit_subtractor_module (
    input [3:0] input_x, input_y,
    output [3:0] difference,
    output borrow_out
);
    wire borrow1, borrow2, borrow3;
    bit_adder sub0 (.x(input_x[0]), .y(~input_y[0]), .carry_in(1'b1), .sum_out(difference[0]), .carry_out(borrow1));
    bit_adder sub1 (.x(input_x[1]), .y(~input_y[1]), .carry_in(borrow1), .sum_out(difference[1]), .carry_out(borrow2));
    bit_adder sub2 (.x(input_x[2]), .y(~input_y[2]), .carry_in(borrow2), .sum_out(difference[2]), .carry_out(borrow3));
    bit_adder sub3 (.x(input_x[3]), .y(~input_y[3]), .carry_in(borrow3), .sum_out(difference[3]), .carry_out(borrow_out));
endmodule

// Top module for the Arithmetic Logic Unit (ALU)
module arithmetic_logic_unit (
    input [3:0] input_x, input_y,
```

```
    input [1:0] control_signal,
    input clk,
    output reg [3:0] output_result
);
    wire [3:0] sum_result, difference;
    wire carry_out, borrow_out;

    four_bit_adder_module adder_instance (.input_x(input_x), .input_y(input_y),
.sum_result(sum_result), .carry_out(carry_out));
    four_bit_subtractor_module subtractor_instance (.input_x(input_x), .input_y(input_y),
.difference(difference), .borrow_out(borrow_out));

    always @(posedge clk) begin
        case (control_signal)
            2'b00: output_result <= 4'b0000;
            2'b01: output_result <= sum_result;
            2'b10: output_result <= difference;
            2'b11: output_result <= output_result;
        endcase
    end
endmodule
```

- **TESTBENCH->**

```
// Testbench for the ALU module
module alu_testbench;
    reg [3:0] x_operand, y_operand;
    reg [1:0] ctrl_signal;
    reg clock;
    wire [3:0] result;
```

```
 arithmetic_logic_unit alu_instance (
        .input_x(x_operand),
        .input_y(y_operand),
        .control_signal(ctrl_signal),
        .clk(clock),
        .output_result(result)
    );

    initial begin
        // Initialize signals
        x_operand = 4'b0000;
        y_operand = 4'b0000;
        ctrl_signal = 2'b00;
        clock = 0;
        // Generate VCD file for waveform analysis
        $dumpfile("alu_simulation.vcd");
        $dumpvars(0, alu_testbench);
        $monitor($time, " clk=%b ctrl_signal=%b x_operand=%b y_operand=%b result=%b", clock, ctrl_signal, x_operand, y_operand, result);
// Testing all with different values

// Test addition operation
        #10 x_operand = 4'b1010; y_operand = 4'b0101; ctrl_signal = 2'b01;
        #10 clock = 1; #10 clock = 0;

// Test subtraction operation
        #10 x_operand = 4'b1100; y_operand = 4'b0110; ctrl_signal = 2'b10;
        #10 clock = 1; #10 clock = 0;
        #10 x_operand = 4'b0111; y_operand = 4'b0011; ctrl_signal = 2'b10;
```

```
        #10 clock = 1; #10 clock = 0;

// Test no operation
        #10 x_operand = 4'b1111; y_operand = 4'b0000; ctrl_signal = 2'b11; // No operation
        #10 clock = 1; #10 clock = 0;

// Test ALU off
        #10 ctrl_signal = 2'b00; // ALU off
        #10 clock = 1; #10 clock = 0;

        $finish;

    end
endmodule
```

➢ SIMULATION->

(TOOLS USED-> VCS, Verdi)

The design was tested using VCS to ensure proper functionality. Testbenches were created to apply different inputs and observe the ALU's behaviour. Verdi was used for waveform analysis.

COMMANDS USED->

vcs ALU.v -full64 -debug_access+all -lca -kdb

./simv -Verdi

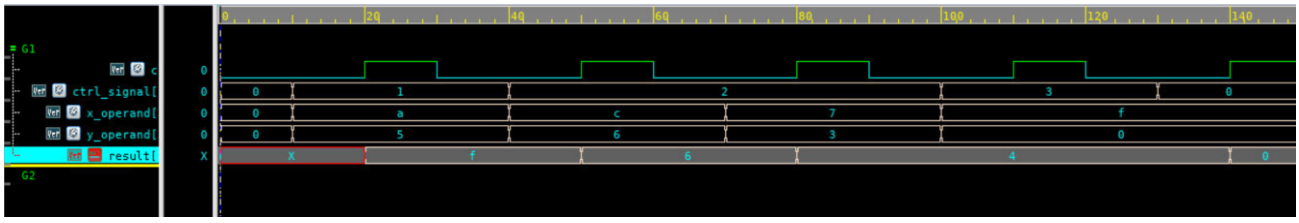
Verdi -alu_simulation.vcd -nologo

VCS SIMULATION OUTPUT->

```
Verdi KDB elaboration done and the database successfully generated: 0 error(s), 0 warning(s)
[til_49@til internship_rtl2gds]$ ./simv -Verdi
Info: [VCS_SAVE_RESTORE_INFO] ASLR (Address Space Layout Randomization) is detected on the machine.
ity, ASLR will be switched off and simv re-executed.
Please use '-no_save' simv switch to avoid re-execution or '-suppress=ASLR_DETECTED_INFO' to suppres
Chronologic VCS simulator copyright 1991-2023
Contains Synopsys proprietary information.
Compiler version V-2023.12_Full64; Runtime version V-2023.12_Full64;  Jul 19 08:26 2024
VCS Build Date = Nov 27 2023 08:44:06
Start run at Jul 19 08:26 2024

                   0 clk=0 ctrl_signal=00 x_operand=0000 y_operand=0000 result=xxxx
                  10 clk=0 ctrl_signal=01 x_operand=1010 y_operand=0101 result=xxxx
                  20 clk=1 ctrl_signal=01 x_operand=1010 y_operand=0101 result=1111
                  30 clk=0 ctrl_signal=01 x_operand=1010 y_operand=0101 result=1111
                  40 clk=0 ctrl_signal=10 x_operand=1100 y_operand=0110 result=1111
                  50 clk=1 ctrl_signal=10 x_operand=1100 y_operand=0110 result=0110
                  60 clk=0 ctrl_signal=10 x_operand=1100 y_operand=0110 result=0110
                  70 clk=0 ctrl_signal=10 x_operand=0111 y_operand=0011 result=0110
                  80 clk=1 ctrl_signal=10 x_operand=0111 y_operand=0011 result=0100
                  90 clk=0 ctrl_signal=10 x_operand=0111 y_operand=0011 result=0100
                 100 clk=0 ctrl_signal=11 x_operand=1111 y_operand=0000 result=0100
                 110 clk=1 ctrl_signal=11 x_operand=1111 y_operand=0000 result=0100
                 120 clk=0 ctrl_signal=11 x_operand=1111 y_operand=0000 result=0100
                 130 clk=0 ctrl_signal=00 x_operand=1111 y_operand=0000 result=0100
                 140 clk=1 ctrl_signal=00 x_operand=1111 y_operand=0000 result=0000
$finish called from file "ALU.v", line 98.
$finish at simulation time                150
         V C S   S i m u l a t i o n   R e p o r t
Time: 150
CPU Time:      0.680 seconds;       Data structure size:   0.0Mb
Fri Jul 19 08:26:42 2024
[til_49@til internship_rtl2gds]$
```

VERDI WAVEFORM OUTPUT->



- **SYNTHESIS->**

(TOOL USED-> Design Compiler, DC)

The synthesized netlist was generated using the Design Compiler .

RTL file name given in this project is ALU.v and SDC file is alu.sdc .

Target Library used is saed32rvt_tt0p78vn40c.db

Steps for synthesis->

1. Go to DC directory then invoke DC shell using command:

```
dc_shell
```

2. After invoking the tool follow the below given commands

```
source -echo -verbose ./rm_setup/dc_setup.tcl
set RTL_SOURCE_FILES ./../rtl/ALU.v
define_design_lib WORK -path ./WORK
```

3. Now set the don't use cells:

```
set_dont_use [get_lib_cells */FADD*]
set_dont_use [get_lib_cells */HADD*]
set_dont_use [get_lib_cells */AO*]
set_dont_use [get_lib_cells */OA*]
set_dont_use [get_lib_cells */NAND*]
set_dont_use [get_lib_cells */XOR*]
set_dont_use [get_lib_cells */NOR*]
set_dont_use [get_lib_cells */XNOR*]
set_dont_use [get_lib_cells */MUX*]
```

4. analyze -format verilog ${RTL_SOURCE_FILES}

5. elaborate ${DESIGN_NAME}

6. current_design

7. Write your Constraint File->

```
create_clock -name clk -period 0.75 [get_ports clk]

set_clock_uncertainty -setup 0.2 [get_clocks clk]

set_clock_latency -max 0.150 [get_clocks clk]

set_clock_latency -source -max 0.075 [get_clocks clk]

set_clock_transition -max 0.3 [get_clocks clk]

set_min_pulse_width -high 0.5 [all_clocks]
```

8. read_sdc ./../CONSTRAINTS/alu.sdc

9. Compile using command->

compile

compile_ultra (for more optimized report)

10. report_timing

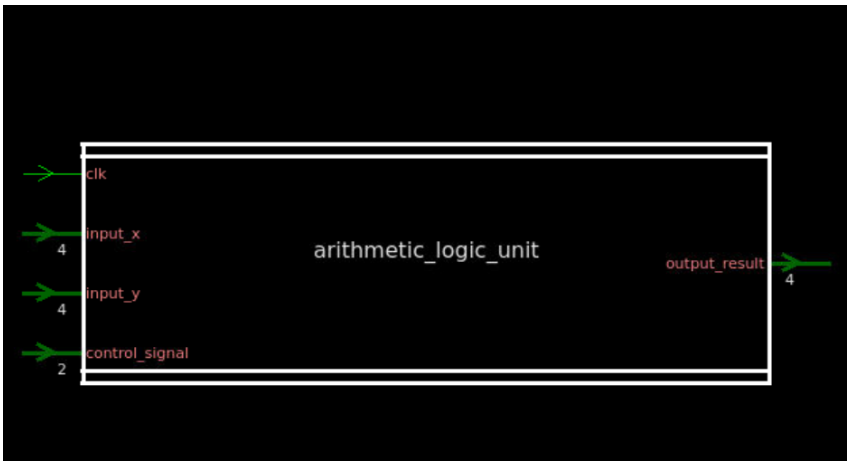
11. report_qor

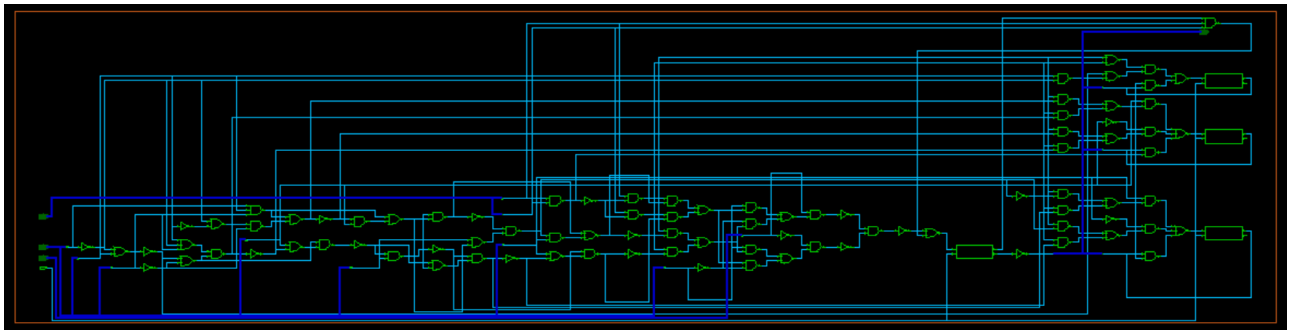
12. Write the output files:

write -format verilog -hierarchy -output ${RESULTS_DIR}/${DCRM_FINAL_VERILOG_OUTPUT_FILE}

write -format ddc -hierarchy -output ${RESULTS_DIR}/${DCRM_FINAL_DDC_OUTPUT_FILE}

- **Schematic of synthesized netlist->**





➢ Timing_report->

```
****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : arithmetic_logic_unit
Version: V-2023.12
Date   : Mon Jul 22 04:40:57 2024
****************************************

Operating Conditions: tt0p78vn40c   Library: saed32rvt_tt0p78vn40c
Wire Load Model Mode: enclosed

  Startpoint: output_result_reg[3]
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: output_result_reg[3]
            (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
```

arithmetic_logic_unit

ForQA saed32rvt_tt0p78vn40c

Point	Incr	Path
clock clk (rise edge)	0.00	0.00
clock network delay (ideal)	0.23	0.23
output_result_reg[3]/CLK (DFFX1_RVT)	0.00	0.23 r
output_result_reg[3]/Q (DFFX1_RVT)	0.29	0.51 r
U132/Y (AND3X1_RVT)	0.10	0.61 r
U128/Y (OR2X1_RVT)	0.08	0.69 r
output_result_reg[3]/D (DFFX1_RVT)	0.01	0.71 r
data arrival time	0.71	
clock clk (rise edge)	0.75	0.75
clock network delay (ideal)	0.23	0.98
clock uncertainty	-0.20	0.78
output_result_reg[3]/CLK (DFFX1_RVT)	0.00	0.78 r
library setup time	-0.07	0.71
data required time	0.71	
data required time	0.71	
data arrival time	-0.71	
slack (MET)	0.00	

➢ Report_qor->

```
****************************************
Report : qor
Design : arithmetic_logic_unit
Version: V-2023.12
Date   : Mon Jul 22 04:37:24 2024
****************************************
```

Timing Path Group 'clk'

Levels of Logic:	2.00
Critical Path Length:	0.48
Critical Path Slack:	0.00
Critical Path Clk Period:	0.75
Total Negative Slack:	0.00
No. of Violating Paths:	0.00
Worst Hold Violation:	0.00
Total Hold Violation:	0.00
No. of Hold Violations:	0.00

Cell Count

Hierarchical Cell Count:	0
Hierarchical Port Count:	0
Leaf Cell Count:	94
Buf/Inv Cell Count:	22
Buf Cell Count:	0
Inv Cell Count:	22

CT Buf/Inv Cell Count: 0

Combinational Cell Count: 90

Sequential Cell Count: 4

Macro Count: 0

Area

Combinational Area: 168.243335

Noncombinational Area: 26.430977

Buf/Inv Area: 27.955840

Total Buffer Area: 0.00

Total Inverter Area: 27.96

Macro/Black Box Area: 0.000000

Net Area: 16.229180

Cell Area: 194.674312

Design Area: 210.903492

Design Rules

Total Number of Nets: 106

Nets With Violations: 0

Max Trans Violations: 0

Max Cap Violations: 0

Hostname: ti1.trg.vlsiexpert.in

Compile CPU Statistics

Resource Sharing: 0.00

Logic Optimization: 0.07

Mapping Optimization: 0.91

Overall Compile Time: 7.79

Overall Compile Wall Clock Time: 8.46

Design WNS: 0.00 TNS: 0.00 Number of Violating Paths: 0

Design (Hold) WNS: 0.00 TNS: 0.00 Number of Violating Paths: 0

➢ Synthesized gate level netlist:

Arithmetic_logic_unit.mapped.v

```
////////////////////////////////////////////////////////////
// Created by: Synopsys DC Ultra(TM) in wire load mode
// Version   : V-2023.12
// Date      : Mon Jul 22 04:22:12 2024
////////////////////////////////////////////////////////////


module arithmetic_logic_unit ( input_x, input_y, control_signal, clk,
      output_result );
  input [3:0] input_x;
  input [3:0] input_y;
  input [1:0] control_signal;
  output [3:0] output_result;
  input clk;
```

```
  wire   n219, n17, n18, n19, n20, n132, n133, n134, n135, n136, n137, n138,
       n139, n140, n141, n142, n143, n144, n145, n146, n147, n148, n149,
       n150, n151, n152, n153, n154, n155, n156, n157, n158, n159, n160,
       n161, n162, n163, n164, n165, n166, n167, n168, n169, n170, n171,
       n172, n173, n174, n175, n176, n177, n178, n179, n180, n181, n182,
       n183, n184, n185, n186, n187, n188, n189, n190, n191, n192, n193,
       n194, n195, n196, n197, n198, n199, n200, n201, n202, n203, n204,
       n205, n206, n207, n208, n209, n210, n211, n212, n213, n214, n215,
       n216, n217;

  DFFX1_RVT \output_result_reg[3]  ( .D(n20), .CLK(clk), .Q(n219), .QN(n217)
       );
  DFFX1_RVT \output_result_reg[2]  ( .D(n19), .CLK(clk), .Q(output_result[2])
       );
  DFFX1_RVT \output_result_reg[1]  ( .D(n18), .CLK(clk), .Q(output_result[1])
       );
  DFFX1_RVT \output_result_reg[0]  ( .D(n17), .CLK(clk), .Q(output_result[0])
       );
INVX0_RVT U125 ( .A(n178), .Y(n135) );
  INVX0_RVT U126 ( .A(n177), .Y(n134) );
  INVX0_RVT U127 ( .A(n133), .Y(n132) );
  OR2X1_RVT U128 ( .A1(n132), .A2(n136), .Y(n20) );
  AND2X1_RVT U129 ( .A1(output_result[2]), .A2(n214), .Y(n190) );
  AND2X1_RVT U130 ( .A1(output_result[1]), .A2(n214), .Y(n204) );
  AND2X1_RVT U131 ( .A1(n135), .A2(n134), .Y(n133) );
  AND3X1_RVT U132 ( .A1(n219), .A2(control_signal[0]), .A3(control_signal[1]),
      .Y(n136) );
  INVX0_RVT U133 ( .A(n217), .Y(output_result[3]) );
  AND2X1_RVT U134 ( .A1(control_signal[0]), .A2(control_signal[1]), .Y(n214)
       );
  INVX0_RVT U135 ( .A(n214), .Y(n147) );
```

```
AND2X1_RVT U136 ( .A1(control_signal[0]), .A2(n147), .Y(n208) );
AND3X1_RVT U137 ( .A1(input_y[0]), .A2(input_x[0]), .A3(input_x[1]), .Y(n141) );
INVX0_RVT U138 ( .A(input_y[0]), .Y(n209) );
INVX0_RVT U139 ( .A(input_x[0]), .Y(n137) );
OR2X1_RVT U140 ( .A1(n209), .A2(n137), .Y(n138) );
INVX0_RVT U141 ( .A(input_x[1]), .Y(n149) );
AND2X1_RVT U142 ( .A1(n138), .A2(n149), .Y(n139) );
OR2X1_RVT U143 ( .A1(n141), .A2(n139), .Y(n193) );
 INVX0_RVT U144 ( .A(n193), .Y(n198) );
AND2X1_RVT U145 ( .A1(n198), .A2(input_y[1]), .Y(n140) );
OR2X1_RVT U146 ( .A1(n141), .A2(n140), .Y(n142) );
AND2X1_RVT U147 ( .A1(input_x[2]), .A2(n142), .Y(n146) );
INVX0_RVT U148 ( .A(n146), .Y(n144) );
OR2X1_RVT U149 ( .A1(input_x[2]), .A2(n142), .Y(n143) );
AND2X1_RVT U150 ( .A1(n144), .A2(n143), .Y(n184) );
AND2X1_RVT U151 ( .A1(n184), .A2(input_y[2]), .Y(n145) );
OR2X1_RVT U152 ( .A1(n146), .A2(n145), .Y(n161) );
AND2X1_RVT U153 ( .A1(n208), .A2(n161), .Y(n160) );
AND2X1_RVT U154 ( .A1(control_signal[1]), .A2(n147), .Y(n207) );
OR2X1_RVT U155 ( .A1(n209), .A2(input_x[0]), .Y(n148) );
INVX0_RVT U156 ( .A(n148), .Y(n211) );
OR2X1_RVT U157 ( .A1(n211), .A2(n149), .Y(n152) );
OR3X1_RVT U158 ( .A1(input_x[0]), .A2(n209), .A3(input_x[1]), .Y(n150) );
AND2X1_RVT U159 ( .A1(n150), .A2(n152), .Y(n194) );
INVX0_RVT U160 ( .A(n194), .Y(n199) );
OR2X1_RVT U161 ( .A1(input_y[1]), .A2(n199), .Y(n151) );
AND2X1_RVT U162 ( .A1(n152), .A2(n151), .Y(n153) );
INVX0_RVT U163 ( .A(n153), .Y(n155) );
AND2X1_RVT U164 ( .A1(n155), .A2(input_x[2]), .Y(n154) );
INVX0_RVT U165 ( .A(n154), .Y(n158) );
OR2X1_RVT U166 ( .A1(n155), .A2(input_x[2]), .Y(n156) );
```

```
  AND2X1_RVT U167 ( .A1(n158), .A2(n156), .Y(n180) );
  INVX0_RVT U168 ( .A(n180), .Y(n185) );
  OR2X1_RVT U169 ( .A1(input_y[2]), .A2(n185), .Y(n157) );
  AND2X1_RVT U170 ( .A1(n158), .A2(n157), .Y(n163) );
  AND2X1_RVT U171 ( .A1(n207), .A2(n163), .Y(n159) );
  OR2X1_RVT U172 ( .A1(n160), .A2(n159), .Y(n172) );
  AND2X1_RVT U173 ( .A1(input_y[3]), .A2(n172), .Y(n168) );
  INVX0_RVT U174 ( .A(input_y[3]), .Y(n171) );
INVX0_RVT U175 ( .A(n161), .Y(n162) );
  AND2X1_RVT U176 ( .A1(n208), .A2(n162), .Y(n166) );
  INVX0_RVT U177 ( .A(n163), .Y(n164) );
  AND2X1_RVT U178 ( .A1(n207), .A2(n164), .Y(n165) );
  OR2X1_RVT U179 ( .A1(n166), .A2(n165), .Y(n170) );
  AND2X1_RVT U180 ( .A1(n171), .A2(n170), .Y(n167) );
  OR2X1_RVT U181 ( .A1(n168), .A2(n167), .Y(n169) );
  AND2X1_RVT U182 ( .A1(input_x[3]), .A2(n169), .Y(n178) );
  INVX0_RVT U183 ( .A(input_x[3]), .Y(n176) );
  AND2X1_RVT U184 ( .A1(input_y[3]), .A2(n170), .Y(n174) );
  AND2X1_RVT U185 ( .A1(n172), .A2(n171), .Y(n173) );
  OR2X1_RVT U186 ( .A1(n174), .A2(n173), .Y(n175) );
  AND2X1_RVT U187 ( .A1(n176), .A2(n175), .Y(n177) );
  INVX0_RVT U188 ( .A(n184), .Y(n179) );
  AND2X1_RVT U189 ( .A1(n208), .A2(n179), .Y(n182) );
  AND2X1_RVT U190 ( .A1(n207), .A2(n180), .Y(n181) );
  OR2X1_RVT U191 ( .A1(n182), .A2(n181), .Y(n183) );
  AND2X1_RVT U192 ( .A1(input_y[2]), .A2(n183), .Y(n192) );
  INVX0_RVT U193 ( .A(input_y[2]), .Y(n189) );
  AND2X1_RVT U194 ( .A1(n208), .A2(n184), .Y(n187) );
  AND2X1_RVT U195 ( .A1(n207), .A2(n185), .Y(n186) );
  OR2X1_RVT U196 ( .A1(n187), .A2(n186), .Y(n188) );
  AND2X1_RVT U197 ( .A1(n189), .A2(n188), .Y(n191) );
```

```
  OR3X1_RVT U198 ( .A1(n192), .A2(n191), .A3(n190), .Y(n19) );
  AND2X1_RVT U199 ( .A1(n208), .A2(n193), .Y(n196) );
  AND2X1_RVT U200 ( .A1(n207), .A2(n194), .Y(n195) );
  OR2X1_RVT U201 ( .A1(n196), .A2(n195), .Y(n197) );
  AND2X1_RVT U202 ( .A1(input_y[1]), .A2(n197), .Y(n206) );
  INVX0_RVT U203 ( .A(input_y[1]), .Y(n203) );
AND2X1_RVT U204 ( .A1(n208), .A2(n198), .Y(n201) );
  AND2X1_RVT U205 ( .A1(n207), .A2(n199), .Y(n200) );
  OR2X1_RVT U206 ( .A1(n201), .A2(n200), .Y(n202) );
  AND2X1_RVT U207 ( .A1(n203), .A2(n202), .Y(n205) );
  OR3X1_RVT U208 ( .A1(n206), .A2(n205), .A3(n204), .Y(n18) );
  OR2X1_RVT U209 ( .A1(n208), .A2(n207), .Y(n213) );
  AND2X1_RVT U210 ( .A1(input_x[0]), .A2(n209), .Y(n210) );
  OR2X1_RVT U211 ( .A1(n211), .A2(n210), .Y(n212) );
  AND2X1_RVT U212 ( .A1(n213), .A2(n212), .Y(n216) );
  AND2X1_RVT U213 ( .A1(n214), .A2(output_result[0]), .Y(n215) );
  OR2X1_RVT U214 ( .A1(n216), .A2(n215), .Y(n17) );
endmodule
```