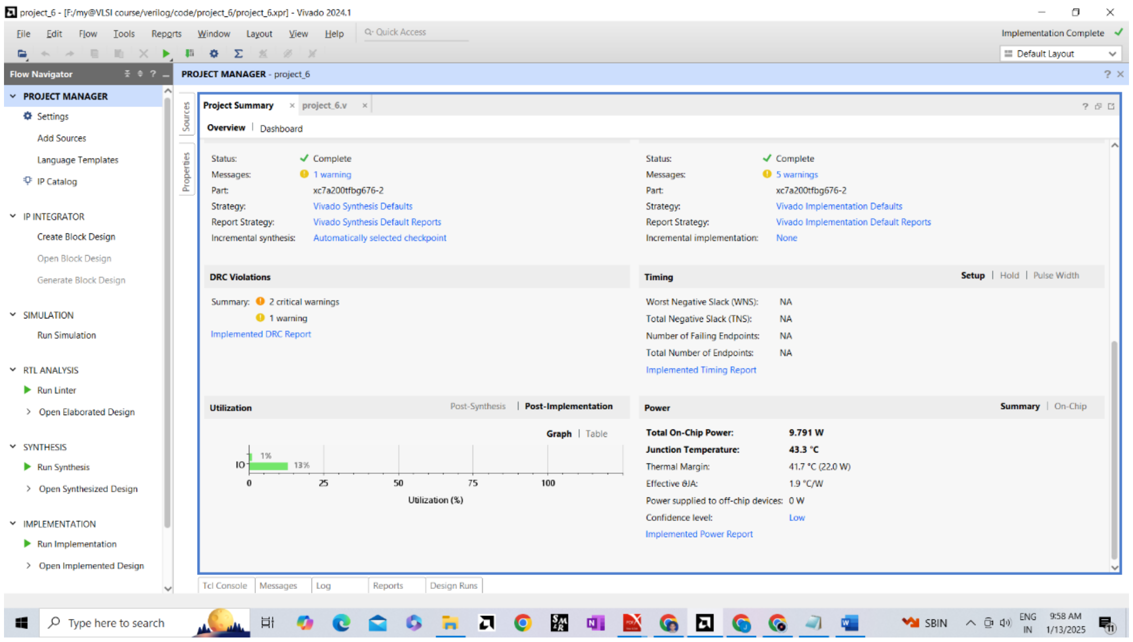
16bit Carry skip Adder

```
module CSK_16bit( input[15:0]a,input c,input[15:0]b,output[16:0]s);
 wire t1, t6, t7;
wire [3:0] xor_ab;
RCA_4bit x1(.a(a[3:0]), .b(b[3:0]), .cin(c), .cout(t1), .sum(s[3:0]));
genvar i;
generate
  for (i = 0; i < 4; i = i + 1) begin : xor_gen
    assign xor_ab[i] = a[i] ^ b[i];
  end
endgenerate
assign t6 = &xor_ab;
assign t7 = t6 ? c : t1;
  wire q1, q6, q7;
wire [3:0] xor_ab2;
RCA_4bit x2(.a(a[7:4]), .b(b[7:4]), .cin(t7), .cout(q1), .sum(s[7:4]));
genvar j;
generate
  for (j = 0; j < 4; j = j + 1) begin : xor_gen2
    assign xor_ab2[j] = a[4+j] ^ b[4+j];
  end
endgenerate
assign q6 = &xor_ab2;
assign q7 = q6 ? t7 : q1;


  wire m1, m6, m7;
wire [3:0] xor_ab3;
RCA_4bit x3(.a(a[11:8]), .b(b[11:8]), .cin(q7), .cout(m1), .sum(s[11:8]));
```

```
genvar k;
generate
    for (k = 0; k < 4; k = k + 1) begin : xor_gen3
        assign xor_ab3[k] = a[8+k] ^ b[8+k];
    end
endgenerate
assign m6 = &xor_ab3;
assign m7 = m6 ? q7 : m1;


   wire n1, n6;
wire [3:0] xor_ab4;
RCA_4bit x4(.a(a[15:12]), .b(b[15:12]), .cin(m7), .cout(n1), .sum(s[15:12]));
genvar l;
generate
    for (l = 0; l < 4; l = l + 1) begin : xor_gen4
        assign xor_ab4[l] = a[12+l] ^ b[12+l];
    end
endgenerate
assign n6 = &xor_ab4;
assign s[16] = n6 ? m7 : n1;


endmodule


 module RCA_4bit(
    input [3:0] a, b, input cin,output cout,output [3:0] sum);
    wire [3:0] carry;
    genvar i;



    generate
        for (i = 0; i < 4; i = i + 1) begin: adder_block
            if (i == 0)
               FA_1bit ff ( .a(a[i]),.b(b[i]),.c(cin),.carry(carry[i]),.s(sum[i]));


            else if (i == 3)
                FA_1bit ff (.a(a[i]),.b(b[i]),.c(carry[i-1]),.carry(cout),.s(sum[i]));
            else
                 FA_1bit ff (.a(a[i]),.b(b[i]),.c(carry[i-1]),.carry(carry[i]),.s(sum[i]));
             end
    endgenerate

endmodule


module FA_1bit(
    input a, b, c,output carry, s );
    assign s = a ^ b ^ c;
    assign carry = (a & b) | (b & c) | (c & a);
endmodule
```



Status:	✓ Complete
Messages:	1 warning
Part:	xc7a200tfbg676-2
Strategy:	Vivado Synthesis Defaults
Report Strategy:	Vivado Synthesis Default Reports
Incremental synthesis:	Automatically selected checkpoint

Status:	✓ Complete
Messages:	5 warnings
Part:	xc7a200tfbg676-2
Strategy:	Vivado Implementation Defaults
Report Strategy:	Vivado Implementation Default Reports
Incremental implementation:	None

DRC Violations

Summary: 2 critical warnings
1 warning
Implemented DRC Report

Timing Setup | Hold | Pulse Width

Worst Negative Slack (WNS):	NA
Total Negative Slack (TNS):	NA
Number of Failing Endpoints:	NA
Total Number of Endpoints:	NA

Implemented Timing Report

Utilization Post-Synthesis | Post-Implementation

Graph | Table

Power Summary | On-Chip

Total On-Chip Power:	**9.791 W**
Junction Temperature:	**43.3 °C**
Thermal Margin:	41.7 °C (22.0 W)
Effective θJA:	1.9 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Implemented Power Report

High-Speed and Low-Power 16-bit Carry Skip Adder Implementation in Verilog

Hybrid CSK Adder

```
module inverter_cmos (
    input  a,
    output  out  ,
    input vdd, vss
);
wire t1,y;
 pmos p1 (y, vdd, a);
 nmos n2 (y, vss, a);
 assign t1=~y;
 assign out =~t1;

endmodule
```

CODE

```
module newCSKA( input[15:0]a,input c,input[15:0]b,output[16:0]s);



   wire t1, t6, t7;
wire [3:0] xor_ab;
RCA_4bit x1(.a(a[3:0]), .b(b[3:0]), .cin(c), .cout(t1), .sum(s[3:0]));
genvar i;
generate
    for (i = 0; i < 4; i = i + 1) begin : xor_gen
        assign xor_ab[i] = a[i] ^ b[i];
    end
endgenerate
assign t6 = &xor_ab;
 inverter_cmos hyb_mux1(.control(t6),.in1(t1) ,.in2(c),.out(t7));




    wire q1, q6, q7;
wire [3:0] xor_ab2;
RCA_4bit x2(.a(a[7:4]), .b(b[7:4]), .cin(t7), .cout(q1), .sum(s[7:4]));
genvar j;
generate
    for (j = 0; j < 4; j = j + 1) begin : xor_gen2
        assign xor_ab2[j] = a[4+j] ^ b[4+j];
    end
endgenerate
```

```
assign q6 = &xor_ab2;

inverter_cmos hyb_mux2(.control(q6),.in1(q1) ,.in2(t7),.out(q7));


  wire m1, m6, m7;
wire [3:0] xor_ab3;
RCA_4bit x3(.a(a[11:8]), .b(b[11:8]), .cin(q7), .cout(m1), .sum(s[11:8]));
genvar k;
generate
  for (k = 0; k < 4; k = k + 1) begin : xor_gen3
    assign xor_ab3[k] = a[8+k] ^ b[8+k];
  end
endgenerate
assign m6 = &xor_ab3;
inverter_cmos hyb_mux3(.control(m6),.in1(m1) ,.in2(q7),.out(m7));

  wire n1, n6;
wire [3:0] xor_ab4;
RCA_4bit x4(.a(a[15:12]), .b(b[15:12]), .cin(m7), .cout(n1), .sum(s[15:12]));
genvar l;
generate
  for (l = 0; l < 4; l = l + 1) begin : xor_gen4
    assign xor_ab4[l] = a[12+l] ^ b[12+l];
  end
endgenerate
assign n6 = &xor_ab4;
inverter_cmos hyb_mux4(.control(n6),.in1(n1) ,.in2(m7),.out(s[16]));
```

```
endmodule



 module RCA_4bit(
    input [3:0] a, b, input cin,output cout,output [3:0] sum);
    wire [3:0] carry;

    genvar i;
 generate
        for (i = 0; i < 4; i = i + 1) begin: adder_block
            if (i == 0)
               FA_1bit ff ( .a(a[i]),.b(b[i]),.c(cin),.carry(carry[i]),.s(sum[i]));

            else if (i == 3)
                FA_1bit ff (.a(a[i]),.b(b[i]),.c(carry[i-1]),.carry(cout),.s(sum[i]));
            else
                 FA_1bit ff (.a(a[i]),.b(b[i]),.c(carry[i-1]),.carry(carry[i]),.s(sum[i]));
             end
    endgenerate
endmodule





module FA_1bit(
    input a, b, c,output carry, s );
    assign s = a ^ b ^ c;
    assign carry = (a & b) | (b & c) | (c & a);
endmodule
```

```
module inverter_cmos (input  control,in1 , in2, output  out);
wire t1,y;
 pmos p1 (y,in1,control);
 nmos n2 (y, in2,control);
 assign t1=~y;
 assign out =~t1;

endmodule
```

TESTBENCH

```
module testbench;


  reg [15:0] a;
  reg [15:0] b;
  reg c;

  // Declare output
  wire [16:0] s;

  // Instantiate the CSK_16bit module
  newCSKA uut (
    .a(a),
    .b(b),
    .c(c),
    .s(s)
  );
```

```
// Apply stimulus
initial begin
  // Monitor the signals for debugging
  $monitor("a = %h, b = %h, c = %b, s = %h", a, b, c, s);

  // Test case 1
  a = 16'hFFFF; b = 16'h0001; c = 0;
  #10; // Wait for 10 time units

  // Test case 2
  a = 16'h1234; b = 16'h5678; c = 1;
  #10;

  // Test case 3
  a = 16'hABCD; b = 16'h4321; c = 0;
  #10;

  // Test case 4
  a = 16'h0000; b = 16'h0000; c = 0;
  #10;

  // Test case 5
  a = 16'hFFFF; b = 16'hFFFF; c = 1;
  #10;

  // Test case 6
  a = 16'hAAAA; b = 16'h5555; c = 0;
  #10;
```

```
    // Test case 7 (All bits 1, carry-in = 1)
    a = 16'hFFFF; b = 16'hFFFF; c = 1;
    #10;

    // Finish simulation
    $finish;
  end

endmodule
```

Result

```
a = ffff, b = 0001, c = 0, s = 10000
a = 1234, b = 5678, c = 1, s = 068ad
a = abcd, b = 4321, c = 0, s = 0eeee
a = 0000, b = 0000, c = 0, s = 00000
a = ffff, b = ffff, c = 1, s = 1ffff
a = aaaa, b = 5555, c = 0, s = 0ffff
a = ffff, b = ffff, c = 1, s = 1ffff
```