Verilog Implementation and Analysis of Adder Architectures

June 1, 2025

SUBMITTED BY

SREENESH K.S[1]
Rajagiri School of Engineering & Technology (RSET)

[1] u2201199@rajagiri.edu.in

Contents

1 Introduction

This project involves the design and implementation of various adder architectures using Verilog HDL. The designs were synthesized and analyzed based on timing, power consumption, and Area utilization. The architectures considered include:

- 4 Bit Ripple Carry Adder (RCA)
- 4 Bit Carry Lookahead Adder (CLA)
- 4 Bit Carry Skip Adder (CSKA)
- 4 Bit Carry Select Adder (CSA)

2 Simulation and Synthesis Setup

- Clock Period: 10 ns
- Tools Used: Vivado

3 Timing Analysis

Table 1: Timing Summary

Adder Type	Worst Negative Slack (ns)	Hold Slack (ns)	Delay (ns)
Ripple Carry Adder	7.812	0.208	2.188
Carry Lookahead Adder	8.282	0.123	1.718
Carry Skip Adder	7.931	0.149	2.069
Carry Select Adder	8.083	0.131	1.917

Inference: Carry Lookahead adder [CLA] has the most positive WNS i.e., the shortest critical path delay Ripple carry adder is the slowest

SLACK= Required Time-Arrival Time

4 Power Analysis

Table 2: Power Summary

Adder Type	On-Chip Power (W)	Device Static Power (W)	Dynamic Power
Ripple Carry Adder	0.074	0.070	0.003
Carry Lookahead Adder	0.115	0.112	0.004
Carry Skip Adder	0.074	0.070	0.003
Carry Select Adder	0.088	0.081	0.007

Inference: For **Ripple Carry Adder** [RCA] due to low transistor count and minimal hardware power consumption.

For the **Carry Skip adder**, as it optimizes carry propgation by skipping blocks when possible and power consumption is usually similar to or higher than RCA but less than CLA & CSA ,as 4bit the powerconsumption is slightly similar to RCA

For **Carry Lookahead adder** , Transistor count is high and generate carry signals in parallel. Therfore higher power consumption

For **carry select adder** the hardware is not minimal therfore it consumes more power due to duplicated hardware and parallel operation as per theoretical background. but after syntesis it gives less power consumption maybe because of less bit used.

5 Resource Utilization

Table 3: Utilization Summary

Adder Type	**Slice LUTs**	**Slice Registers**	**Bonded IOBs**
Ripple Carry Adder	5	14	16
Carry Lookahead Adder	6	14	16
Carry Skip Adder	6	14	16
Carry Select Adder	11	14	16

Inference: The Ripple Carry Adder uses the fewest LUTs (5), making it the most area-efficient. Carry Select Adder uses the most LUTs (11) due to parallel computation blocks and multiplexers. All architectures use the same number of registers (14) and IOBs (16), indicating that differences lie mainly in combinational logic, not in I/O or sequential elements.

6 Result Comparison

The performance of the four 4-bit adder architectures—Ripple Carry Adder (RCA), Carry Lookahead Adder (CLA), Carry Skip Adder (CSKA), and Carry Select Adder (CSA)—is compared based on three major parameters: **timing**, **power consumption**, and **resource utilization**.

Timing Performance:
The **Carry Lookahead Adder (CLA)** has the lowest delay (1.718 ns), making it the fastest among the four. This is due to its parallel carry generation logic, which minimizes the critical path delay.

Power Consumption:
The **Ripple Carry Adder (RCA)** and **Carry Skip Adder (CSKA)** consume the least dynamic power (0.003 W) due to their simpler structures and minimal switching activity in a small 4-bit implementation.

The **Carry Select Adder (CSA)**, despite its theoretical complexity, consumes slightly more dynamic power due to parallel operations. However, in this 4-bit design, the power is still relatively low.

Resource Utilization:
The **Ripple Carry Adder** uses the fewest LUTs (5), indicating high area efficiency.

The **Carry Select Adder** uses the most LUTs (11) due to duplicated logic blocks and multiplexers, making it the least area-efficient among the four.

Conclusion:
For **low-power and area-efficient** designs, especially in low-bit applications, **RCA** or **CSKA** is preferable.

For **high-speed operations**, **CLA** is ideal, despite slightly higher power and area usage.

CSA, while theoretically faster in higher bit-widths, shows higher resource consumption with no significant timing benefit in the 4-bit range, and is less efficient in this case.

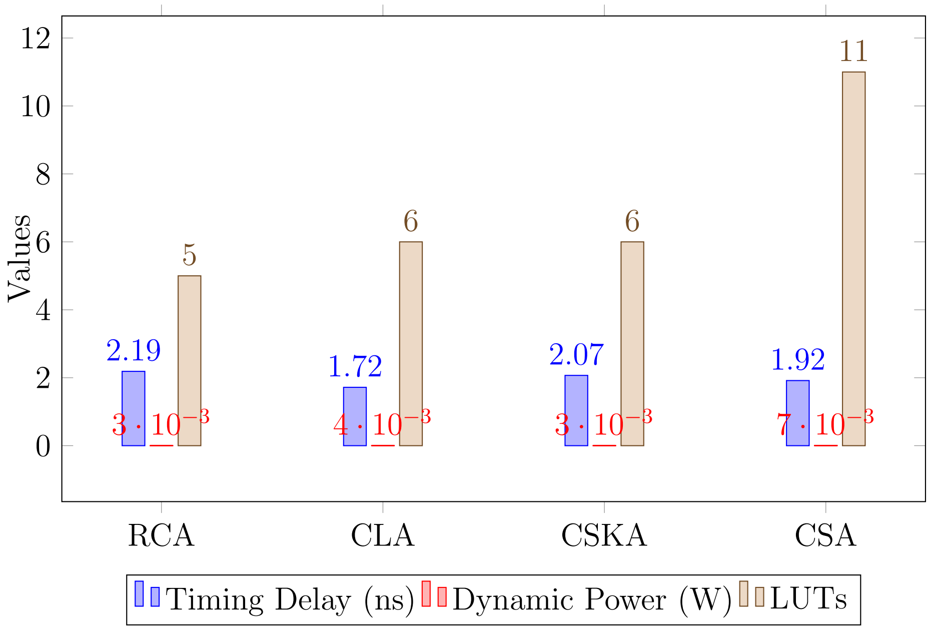
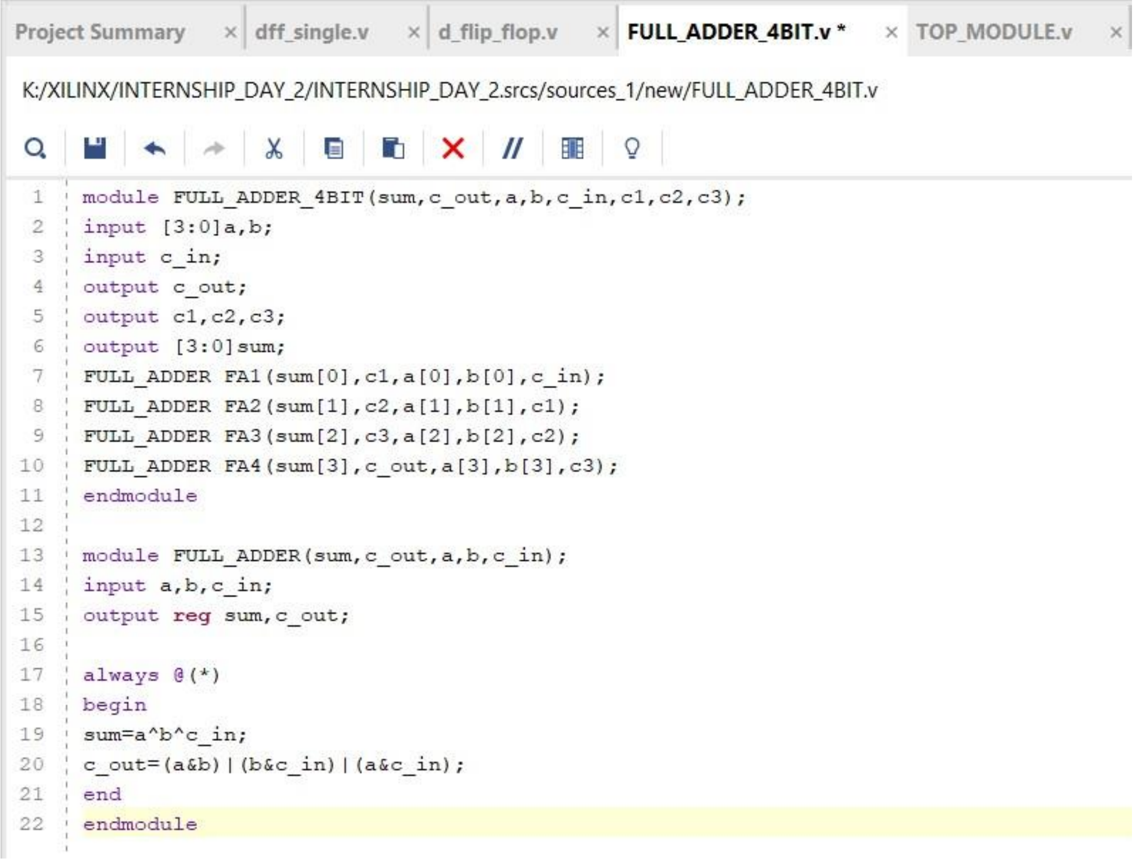


Figure 1: Comparison of 4-bit Adder Architectures

7 Proof Regarding above statements

RIPPLE CARRY ADDER REPORT

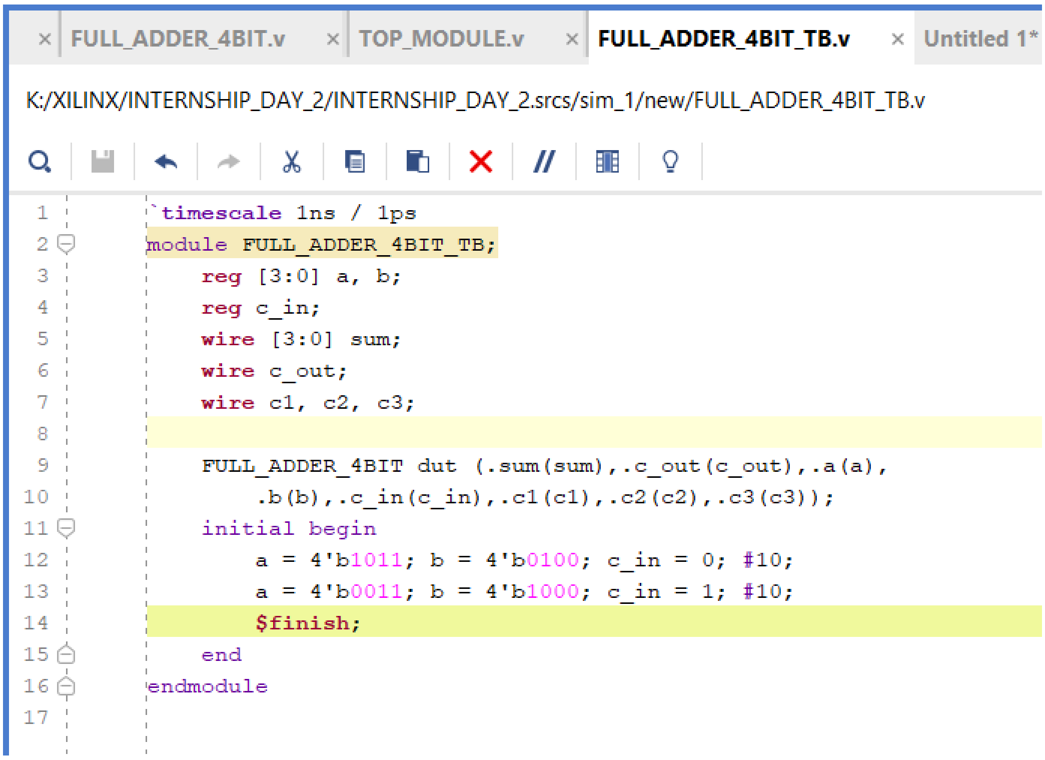
RIPPLE CARRY ADDER CODE:



K:/XILINX/INTERNSHIP_DAY_2/INTERNSHIP_DAY_2.srcs/sources_1/new/FULL_ADDER_4BIT.v

```
module FULL_ADDER_4BIT(sum,c_out,a,b,c_in,c1,c2,c3);
input [3:0]a,b;
input c_in;
output c_out;
output c1,c2,c3;
output [3:0]sum;
FULL_ADDER FA1(sum[0],c1,a[0],b[0],c_in);
FULL_ADDER FA2(sum[1],c2,a[1],b[1],c1);
FULL_ADDER FA3(sum[2],c3,a[2],b[2],c2);
FULL_ADDER FA4(sum[3],c_out,a[3],b[3],c3);
endmodule

module FULL_ADDER(sum,c_out,a,b,c_in);
input a,b,c_in;
output reg sum,c_out;

always @(*)
begin
sum=a^b^c_in;
c_out=(a&b)|(b&c_in)|(a&c_in);
end
endmodule
```

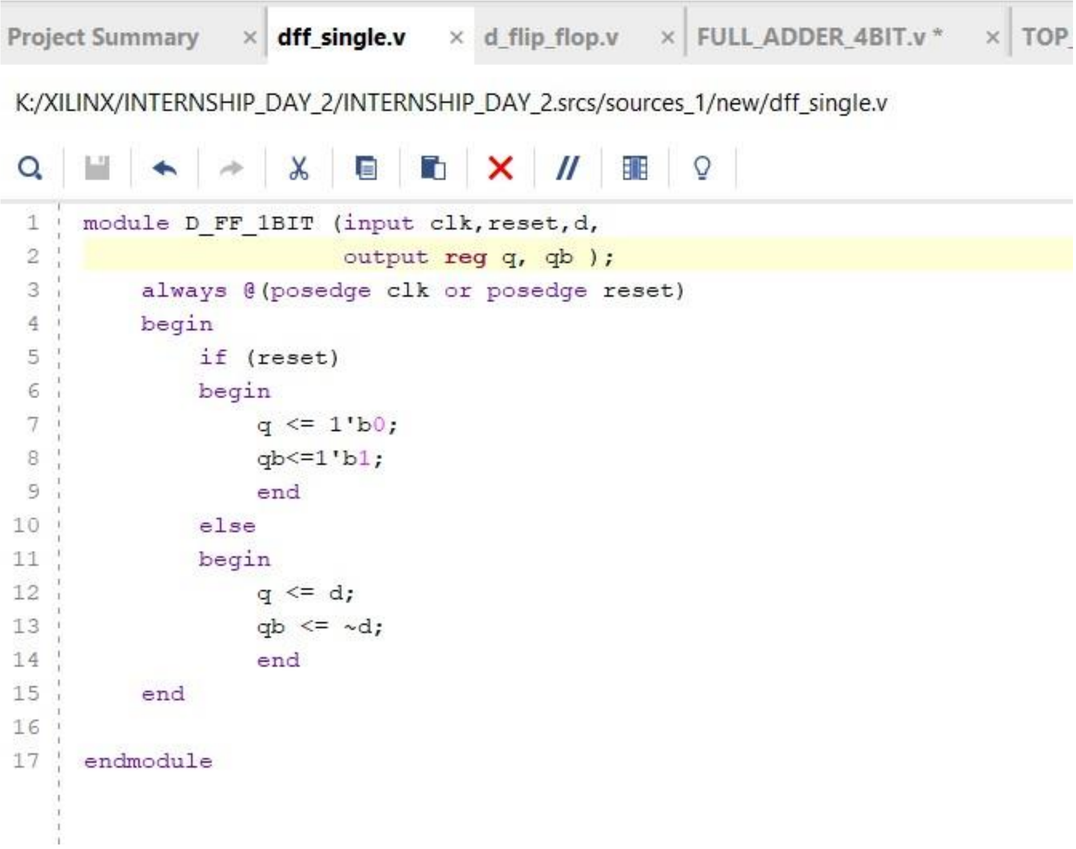
TESTBENCH:



K:/XILINX/INTERNSHIP_DAY_2/INTERNSHIP_DAY_2.srcs/sim_1/new/FULL_ADDER_4BIT_TB.v

```
`timescale 1ns / 1ps
module FULL_ADDER_4BIT_TB;
    reg [3:0] a, b;
    reg c_in;
    wire [3:0] sum;
    wire c_out;
    wire c1, c2, c3;

    FULL_ADDER_4BIT dut (.sum(sum),.c_out(c_out),.a(a),
        .b(b),.c_in(c_in),.c1(c1),.c2(c2),.c3(c3));
    initial begin
        a = 4'b1011; b = 4'b0100; c_in = 0; #10;
        a = 4'b0011; b = 4'b1000; c_in = 1; #10;
        $finish;
    end
endmodule
```

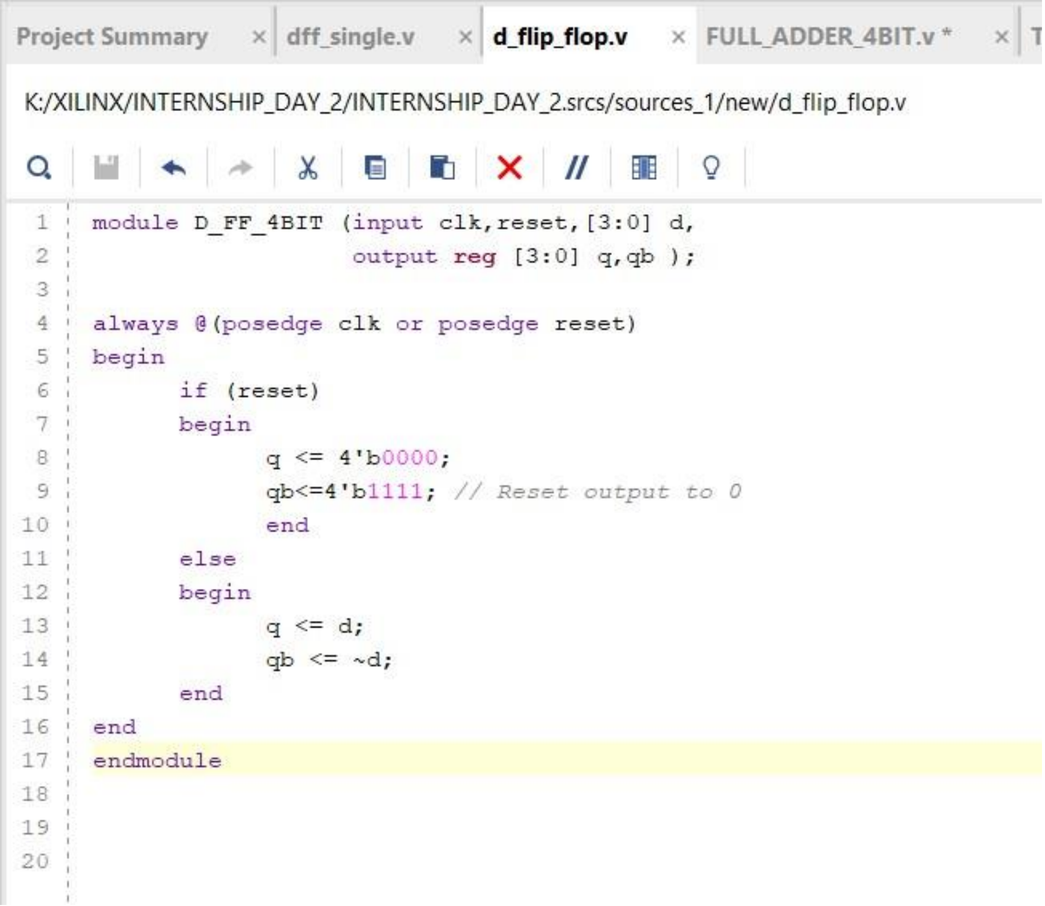
ONE BIT D FLIPFLOP:



K:/XILINX/INTERNSHIP_DAY_2/INTERNSHIP_DAY_2.srcs/sources_1/new/dff_single.v

```
module D_FF_1BIT (input clk,reset,d,
                  output reg q, qb );
    always @(posedge clk or posedge reset)
    begin
        if (reset)
        begin
            q <= 1'b0;
            qb<=1'b1;
            end
        else
        begin
            q <= d;
            qb <= ~d;
            end
    end

endmodule
```

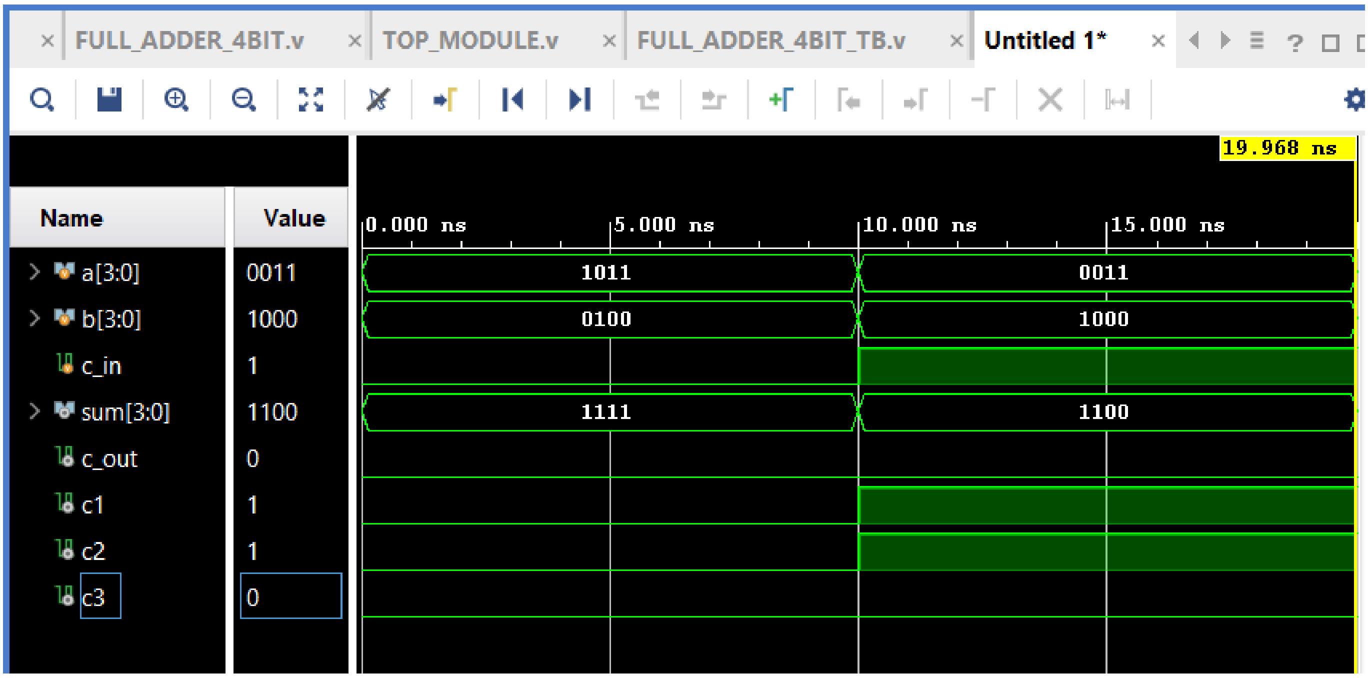
FOUR BIT D FLIPFLOP:



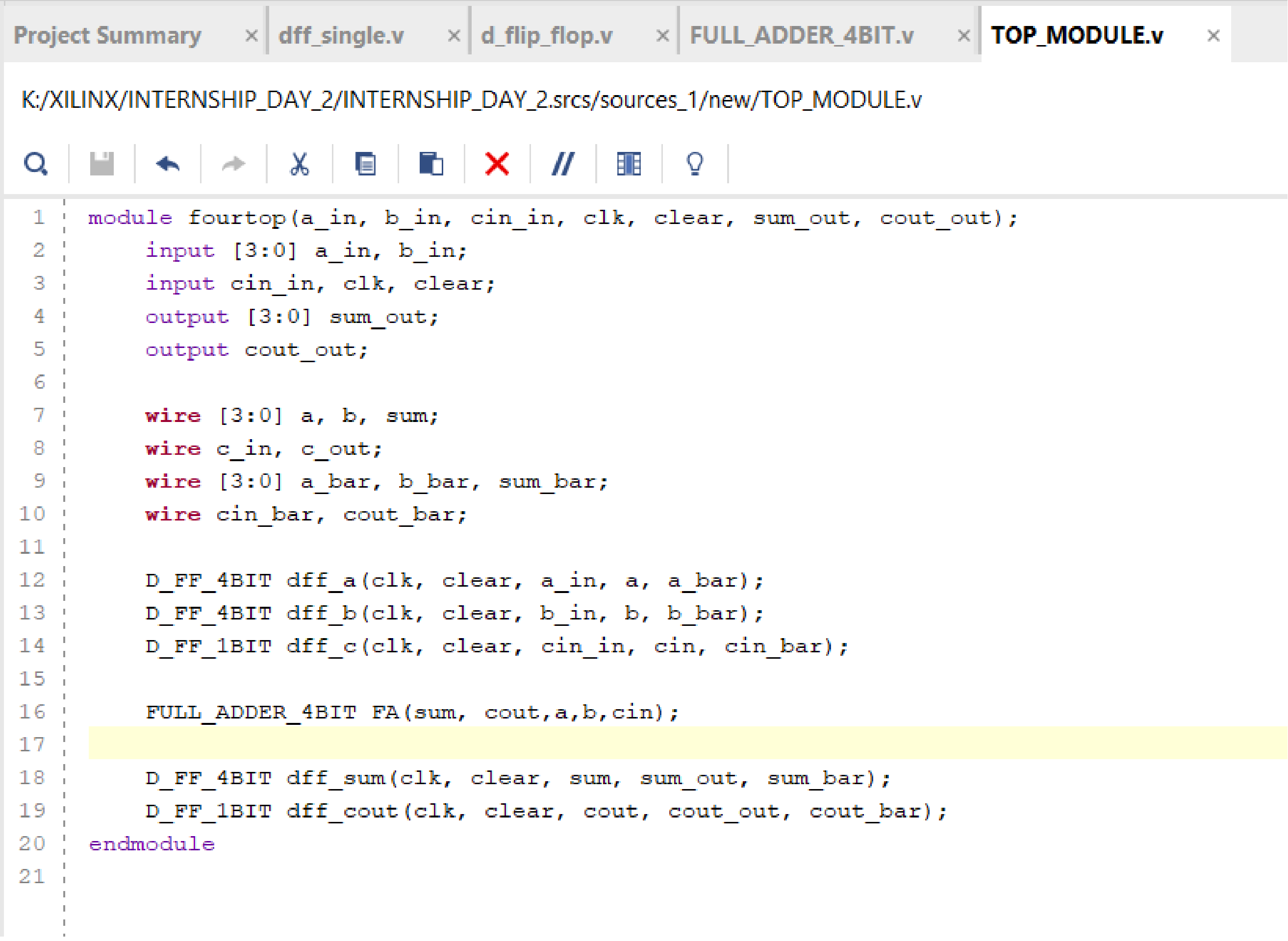
K:/XILINX/INTERNSHIP_DAY_2/INTERNSHIP_DAY_2.srcs/sources_1/new/d_flip_flop.v

```
module D_FF_4BIT (input clk,reset,[3:0] d,
                  output reg [3:0] q,qb );

always @(posedge clk or posedge reset)
begin
      if (reset)
      begin
            q <= 4'b0000;
            qb<=4'b1111; // Reset output to 0
            end
      else
      begin
            q <= d;
            qb <= ~d;
      end
end
endmodule
```

WAVEFORM:



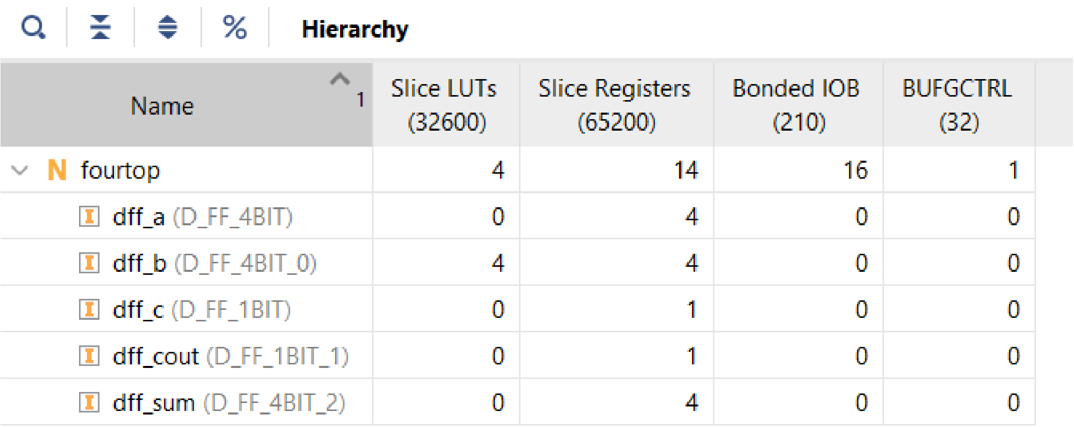
TOP MODULE :



K:/XILINX/INTERNSHIP_DAY_2/INTERNSHIP_DAY_2.srcs/sources_1/new/TOP_MODULE.v

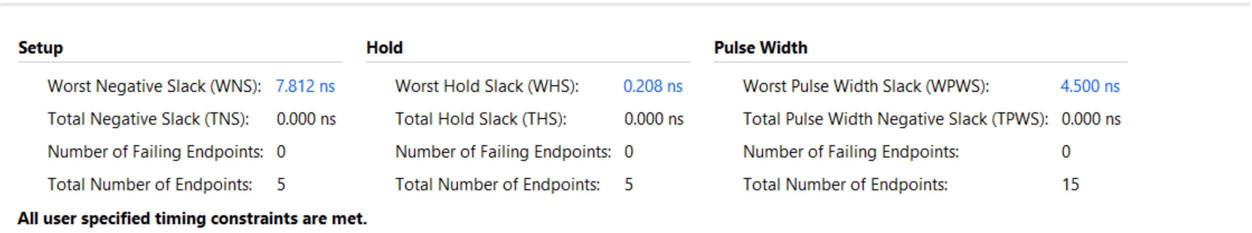
```
module fourtop(a_in, b_in, cin_in, clk, clear, sum_out, cout_out);
    input [3:0] a_in, b_in;
    input cin_in, clk, clear;
    output [3:0] sum_out;
    output cout_out;

    wire [3:0] a, b, sum;
    wire c_in, c_out;
    wire [3:0] a_bar, b_bar, sum_bar;
    wire cin_bar, cout_bar;

    D_FF_4BIT dff_a(clk, clear, a_in, a, a_bar);
    D_FF_4BIT dff_b(clk, clear, b_in, b, b_bar);
    D_FF_1BIT dff_c(clk, clear, cin_in, cin, cin_bar);

    FULL_ADDER_4BIT FA(sum, cout,a,b,cin);

    D_FF_4BIT dff_sum(clk, clear, sum, sum_out, sum_bar);
    D_FF_1BIT dff_cout(clk, clear, cout, cout_out, cout_bar);
endmodule
```

UTILIZATION AREA SUMMARY:



Name	Slice LUTs (32600)	Slice Registers (65200)	Bonded IOB (210)	BUFGCTRL (32)
N fourtop	4	14	16	1
dff_a (D_FF_4BIT)	0	4	0	0
dff_b (D_FF_4BIT_0)	4	4	0	0
dff_c (D_FF_1BIT)	0	1	0	0
dff_cout (D_FF_1BIT_1)	0	1	0	0
dff_sum (D_FF_4BIT_2)	0	4	0	0

DESIGN TIMING REPORT:



Setup

Worst Negative Slack (WNS):	7.812 ns
Total Negative Slack (TNS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	5

Hold

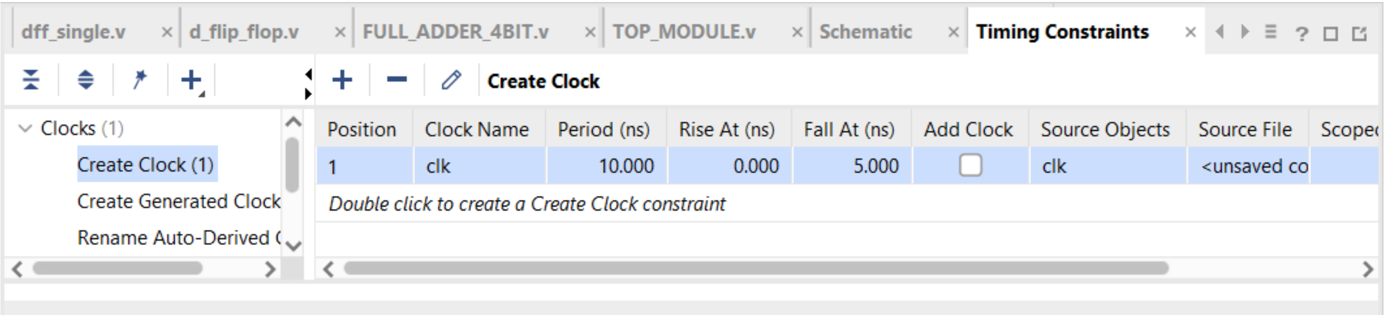
Worst Hold Slack (WHS):	0.208 ns
Total Hold Slack (THS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	5

Pulse Width

Worst Pulse Width Slack (WPWS):	4.500 ns
Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	15

All user specified timing constraints are met.

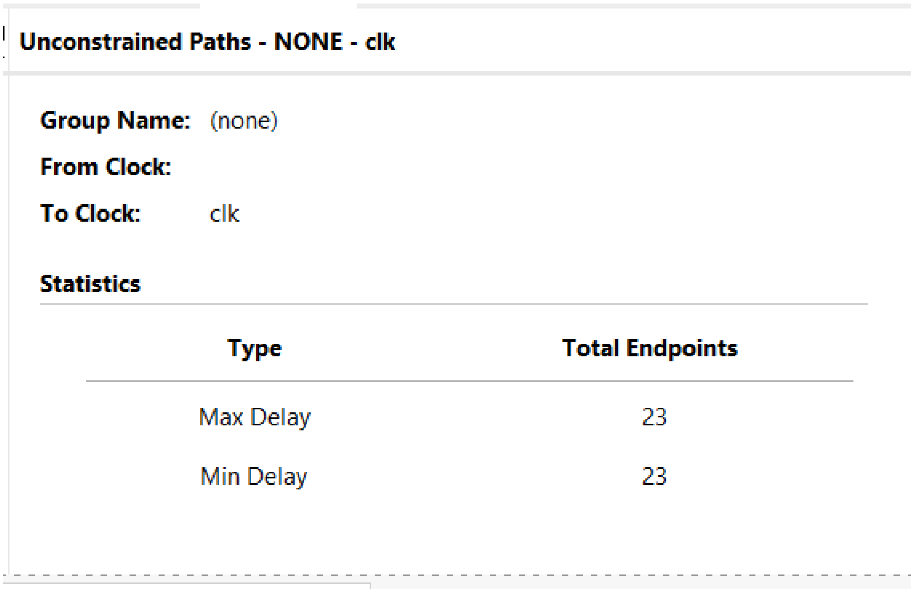
TIMING CONSTRAINTS:



Position	Clock Name	Period (ns)	Rise At (ns)	Fall At (ns)	Add Clock	Source Objects	Source File	Scope
1	clk	10.000	0.000	5.000	□	clk	<unsaved co	

Double click to create a Create Clock constraint

CHECK TIMING :



Unconstrained Paths - NONE - clk

Group Name: (none)

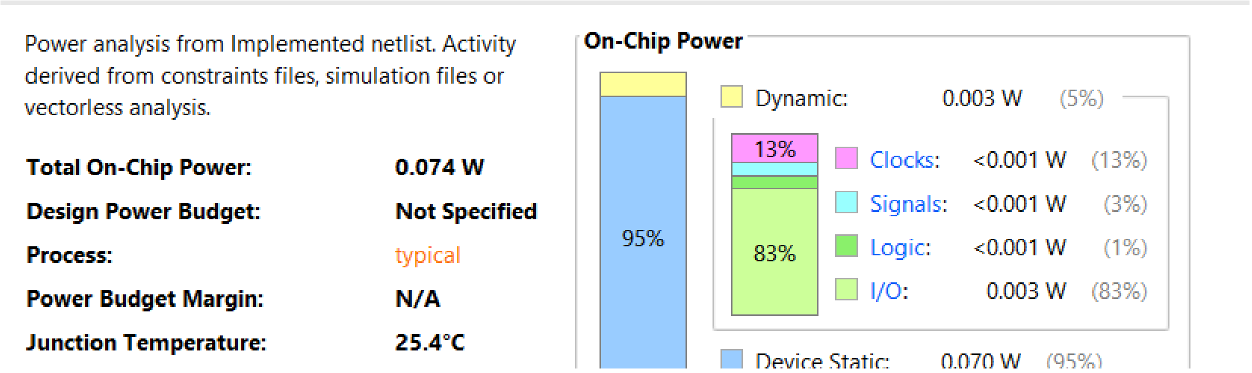
From Clock:

To Clock: clk

Statistics

Type	Total Endpoints
Max Delay	23
Min Delay	23

POWER REPORT:



Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.074 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**25.4°C**

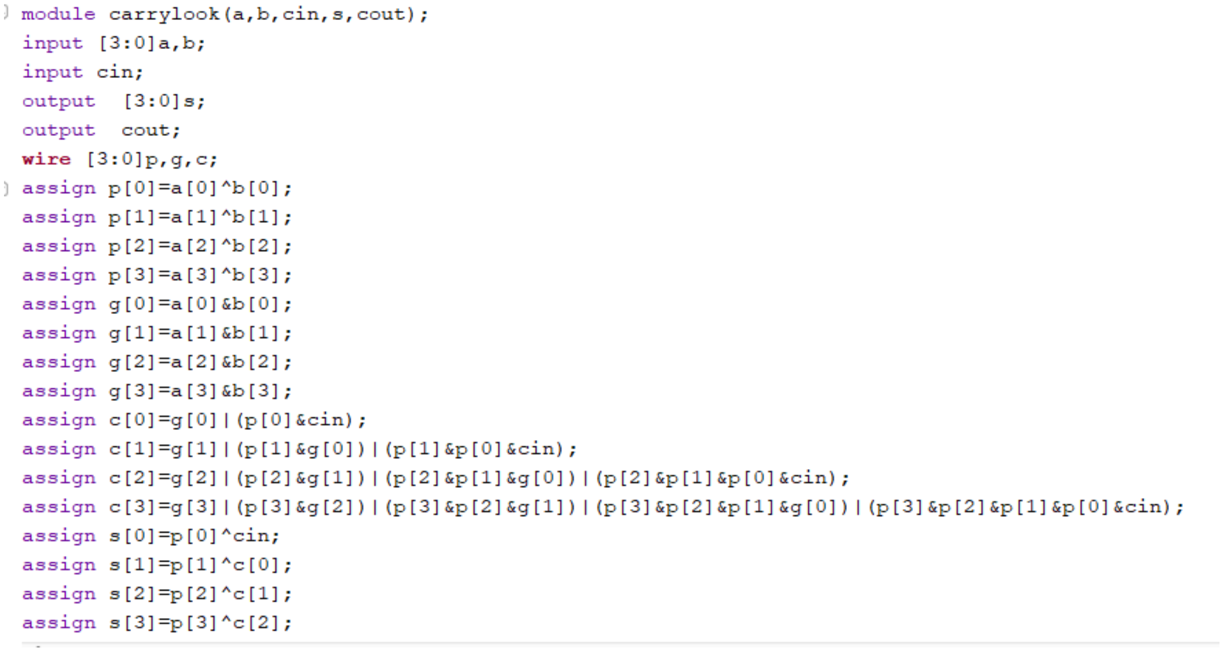
On-Chip Power

🟨	Dynamic:	0.003 W	(5%)
🟪	Clocks:	<0.001 W	(13%)
🟦	Signals:	<0.001 W	(3%)
🟩	Logic:	<0.001 W	(1%)
🟩	I/O:	0.003 W	(83%)
🟦	Device Static:	0.070 W	(95%)

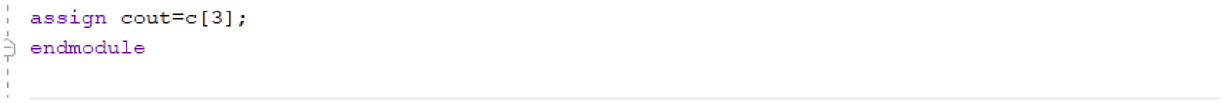
CARRY LOOK AHEAD ADDER REPORT

Submitted by SREENESH K.S
RAJAGIRI SCHOOL OF ENGINEERING & TECHNOLOGY [RSET]
MAIL ID : SREENESH K S RSET u2201199@rajagiri.edu.in

CODE

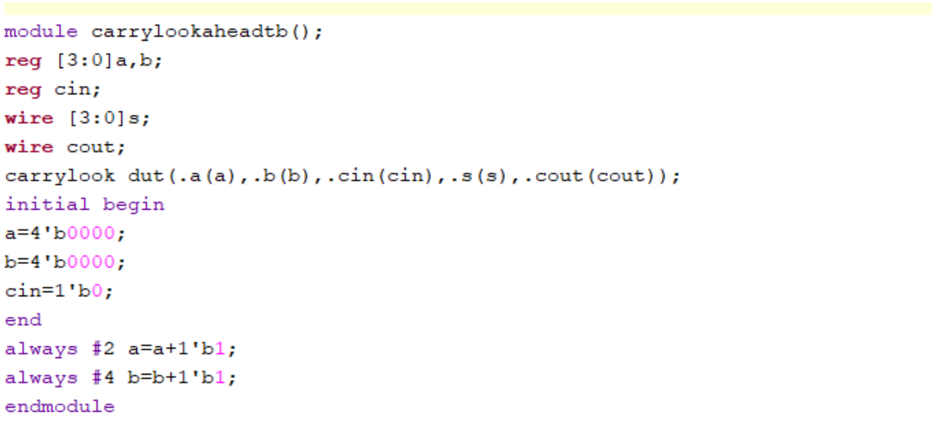


```
module carrylook(a,b,cin,s,cout);
input [3:0]a,b;
input cin;
output  [3:0]s;
output  cout;
wire [3:0]p,g,c;
assign p[0]=a[0]^b[0];
assign p[1]=a[1]^b[1];
assign p[2]=a[2]^b[2];
assign p[3]=a[3]^b[3];
assign g[0]=a[0]&b[0];
assign g[1]=a[1]&b[1];
assign g[2]=a[2]&b[2];
assign g[3]=a[3]&b[3];
assign c[0]=g[0]|(p[0]&cin);
assign c[1]=g[1]|(p[1]&g[0])|(p[1]&p[0]&cin);
assign c[2]=g[2]|(p[2]&g[1])|(p[2]&p[1]&g[0])|(p[2]&p[1]&p[0]&cin);
assign c[3]=g[3]|(p[3]&g[2])|(p[3]&p[2]&g[1])|(p[3]&p[2]&p[1]&g[0])|(p[3]&p[2]&p[1]&p[0]&cin);
assign s[0]=p[0]^cin;
assign s[1]=p[1]^c[0];
assign s[2]=p[2]^c[1];
assign s[3]=p[3]^c[2];
```



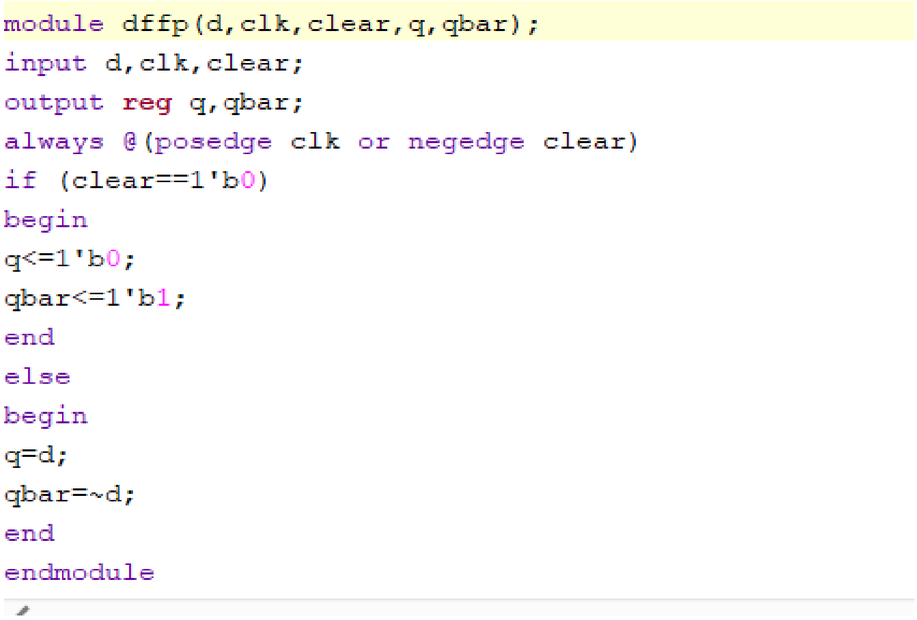
```
assign cout=c[3];
endmodule
```

TESTBENCH

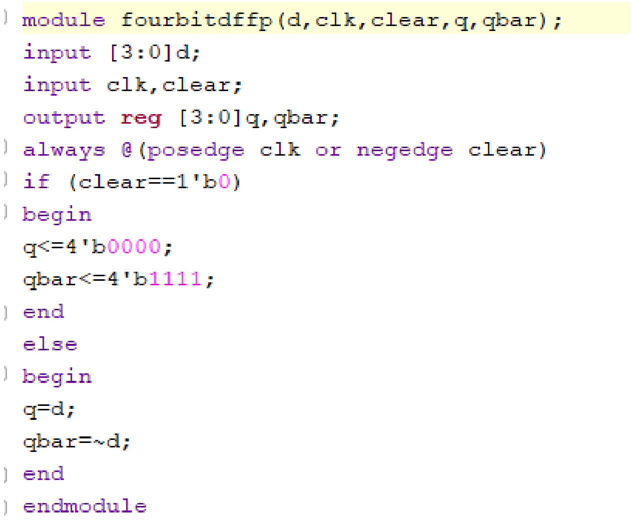


```
module carrylookaheadtb();
reg [3:0]a,b;
reg cin;
wire [3:0]s;
wire cout;
carrylook dut(.a(a),.b(b),.cin(cin),.s(s),.cout(cout));
initial begin
a=4'b0000;
b=4'b0000;
cin=1'b0;
end
always #2 a=a+1'b1;
always #4 b=b+1'b1;
endmodule
```

1 & 4 BIT D FLIP FLOP

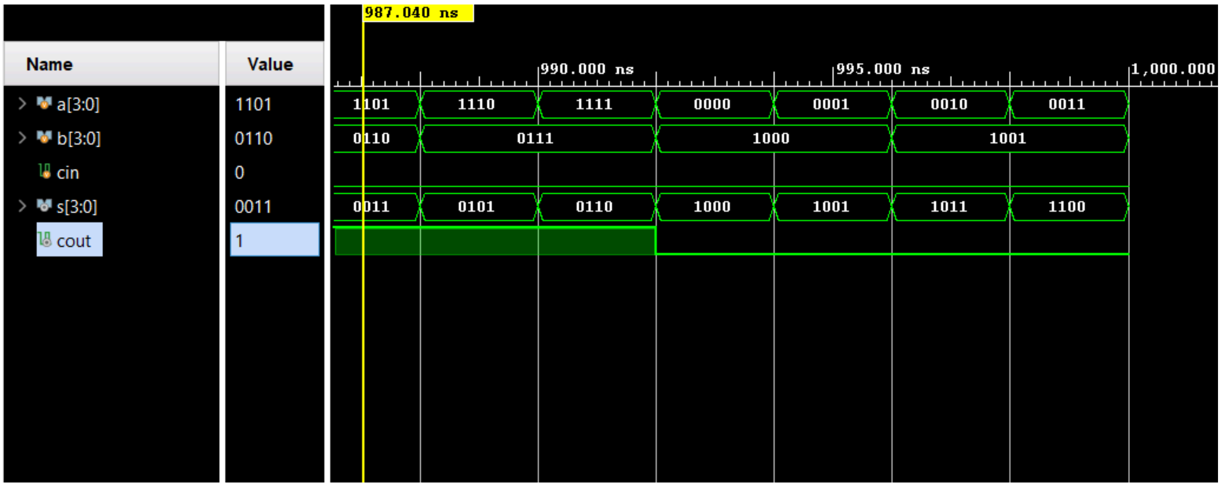


```
module dffp(d,clk,clear,q,qbar);
input d,clk,clear;
output reg q,qbar;
always @(posedge clk or negedge clear)
if (clear==1'b0)
begin
q<=1'b0;
qbar<=1'b1;
end
else
begin
q=d;
qbar=~d;
end
endmodule
```

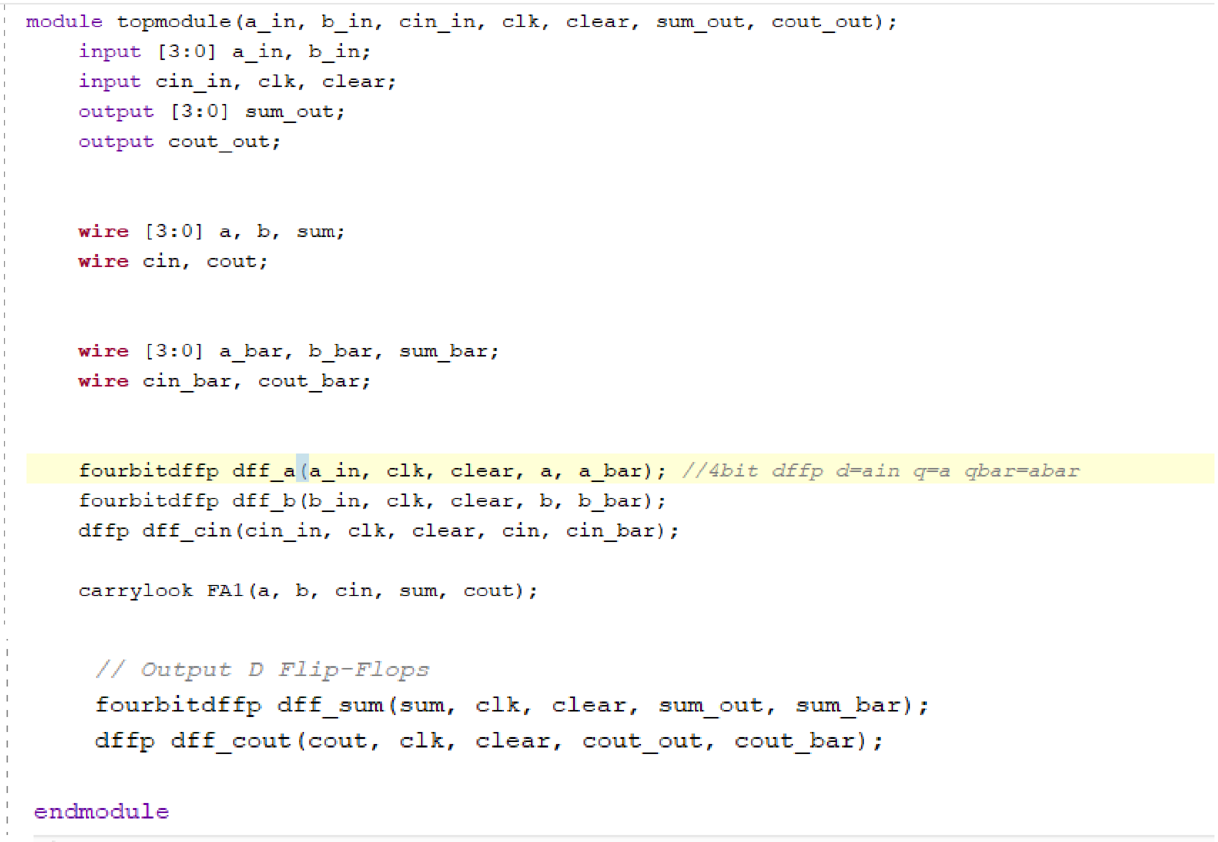


```
module fourbitdffp(d,clk,clear,q,qbar);
input [3:0]d;
input clk,clear;
output reg [3:0]q,qbar;
always @(posedge clk or negedge clear)
if (clear==1'b0)
begin
q<=4'b0000;
qbar<=4'b1111;
end
else
begin
q=d;
qbar=~d;
end
endmodule
```

SIMULATION RESULT

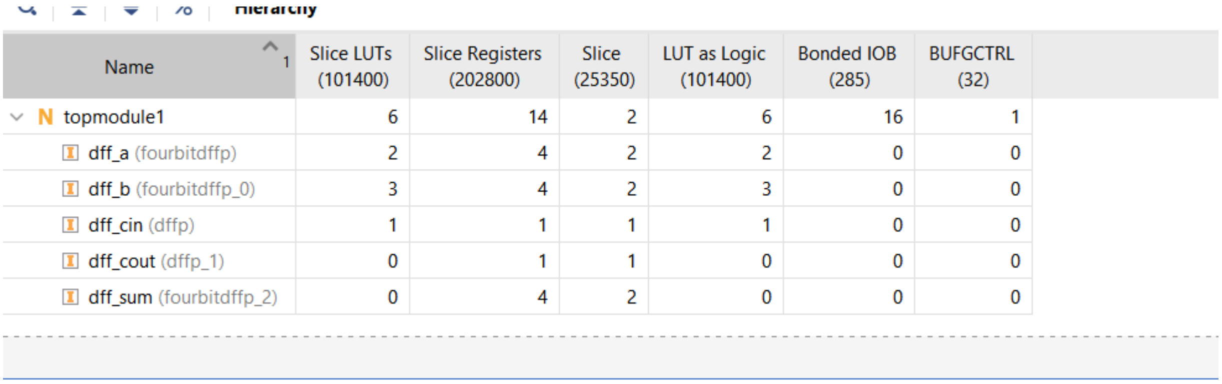


TOP MODULE



```
module topmodule(a_in, b_in, cin_in, clk, clear, sum_out, cout_out);
    input [3:0] a_in, b_in;
    input cin_in, clk, clear;
    output [3:0] sum_out;
    output cout_out;


    wire [3:0] a, b, sum;
    wire cin, cout;


    wire [3:0] a_bar, b_bar, sum_bar;
    wire cin_bar, cout_bar;


    fourbitdffp dff_a(a_in, clk, clear, a, a_bar); //4bit dffp d=ain q=a qbar=abar
    fourbitdffp dff_b(b_in, clk, clear, b, b_bar);
    dffp dff_cin(cin_in, clk, clear, cin, cin_bar);

    carrylook FA1(a, b, cin, sum, cout);


     // Output D Flip-Flops
     fourbitdffp dff_sum(sum, clk, clear, sum_out, sum_bar);
     dffp dff_cout(cout, clk, clear, cout_out, cout_bar);

 endmodule
```

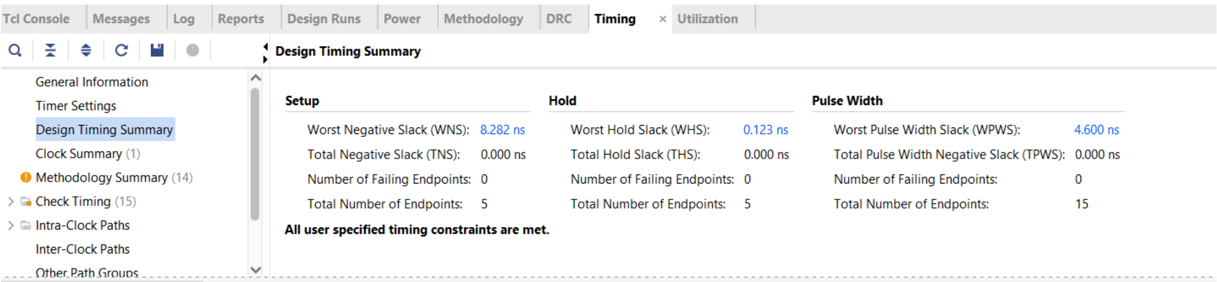
UTILIZATION AREA SUMMARY



Name	Slice LUTs (101400)	Slice Registers (202800)	Slice (25350)	LUT as Logic (101400)	Bonded IOB (285)	BUFGCTRL (32)
topmodule1	6	14	2	6	16	1
dff_a (fourbitdffp)	2	4	2	2	0	0
dff_b (fourbitdffp_0)	3	4	2	3	0	0
dff_cin (dffp)	1	1	1	1	0	0
dff_cout (dffp_1)	0	1	1	0	0	0
dff_sum (fourbitdffp_2)	0	4	2	0	0	0

TIMING REPORT

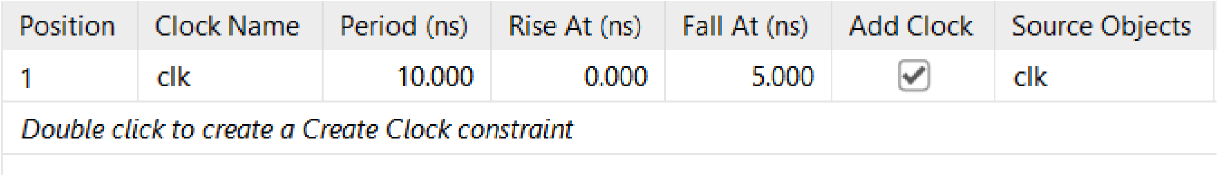
WORST NEGATIVE SLACK IS 8.282ns



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	8.282 ns	Worst Hold Slack (WHS):	0.123 ns	Worst Pulse Width Slack (WPWS):	4.600 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	5	Total Number of Endpoints:	5	Total Number of Endpoints:	15

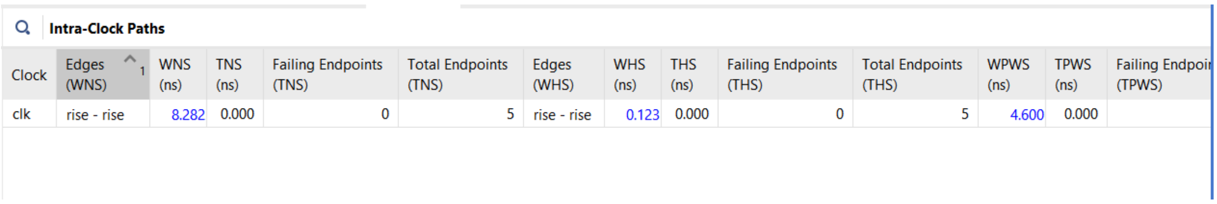
All user specified timing constraints are met.



Position	Clock Name	Period (ns)	Rise At (ns)	Fall At (ns)	Add Clock	Source Objects
1	clk	10.000	0.000	5.000	☑	clk

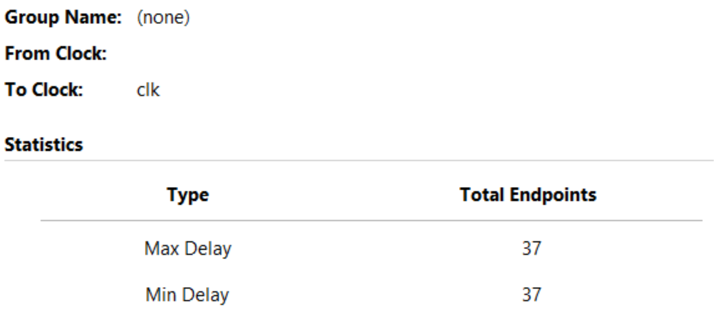
Double click to create a Create Clock constraint

Intra clock path



Intra-Clock Paths

Clock	Edges (WNS)	WNS (ns)	TNS (ns)	Failing Endpoints (TNS)	Total Endpoints (TNS)	Edges (WHS)	WHS (ns)	THS (ns)	Failing Endpoints (THS)	Total Endpoints (THS)	WPWS (ns)	TPWS (ns)	Failing Endpoi (TPWS)
clk	rise - rise	8.282	0.000	0	5	rise - rise	0.123	0.000	0	5	4.600	0.000	



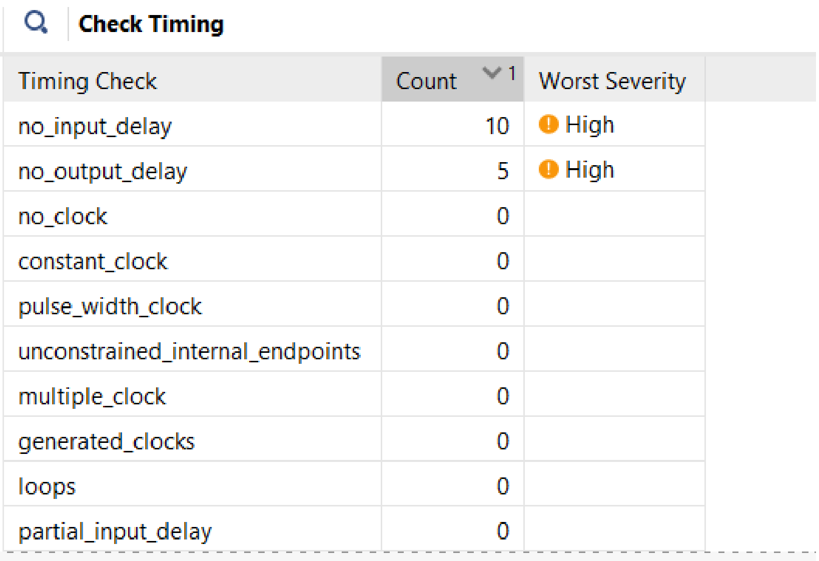
Group Name: (none)

From Clock:

To Clock: clk

Statistics

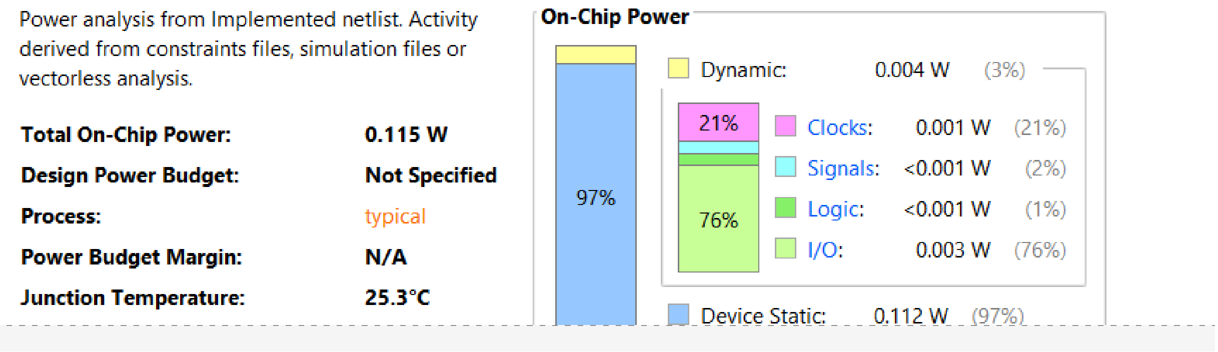
Type	Total Endpoints
Max Delay	37
Min Delay	37



Check Timing

Timing Check	Count	Worst Severity
no_input_delay	10	! High
no_output_delay	5	! High
no_clock	0	
constant_clock	0	
pulse_width_clock	0	
unconstrained_internal_endpoints	0	
multiple_clock	0	
generated_clocks	0	
loops	0	
partial_input_delay	0	

POWER REPORTs



Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.115 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**25.3°C**

On-Chip Power

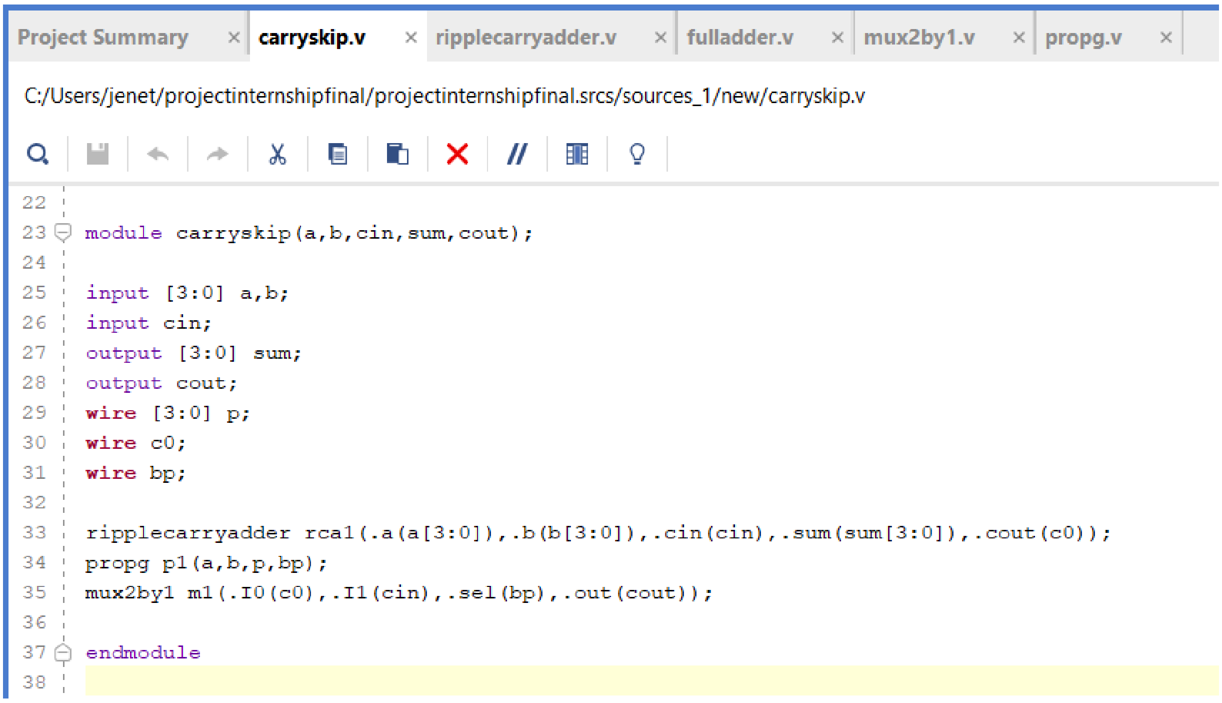
🟨	Dynamic:	0.004 W	(3%)
🟪	Clocks:	0.001 W	(21%)
🟦	Signals:	<0.001 W	(2%)
🟩	Logic:	<0.001 W	(1%)
🟩	I/O:	0.003 W	(76%)
🟦	Device Static:	0.112 W	(97%)

CARRY SKIP ADDER

Submitted by: Jenet Joseph

Mail ID: jenetjoseph.2004@gmail.com

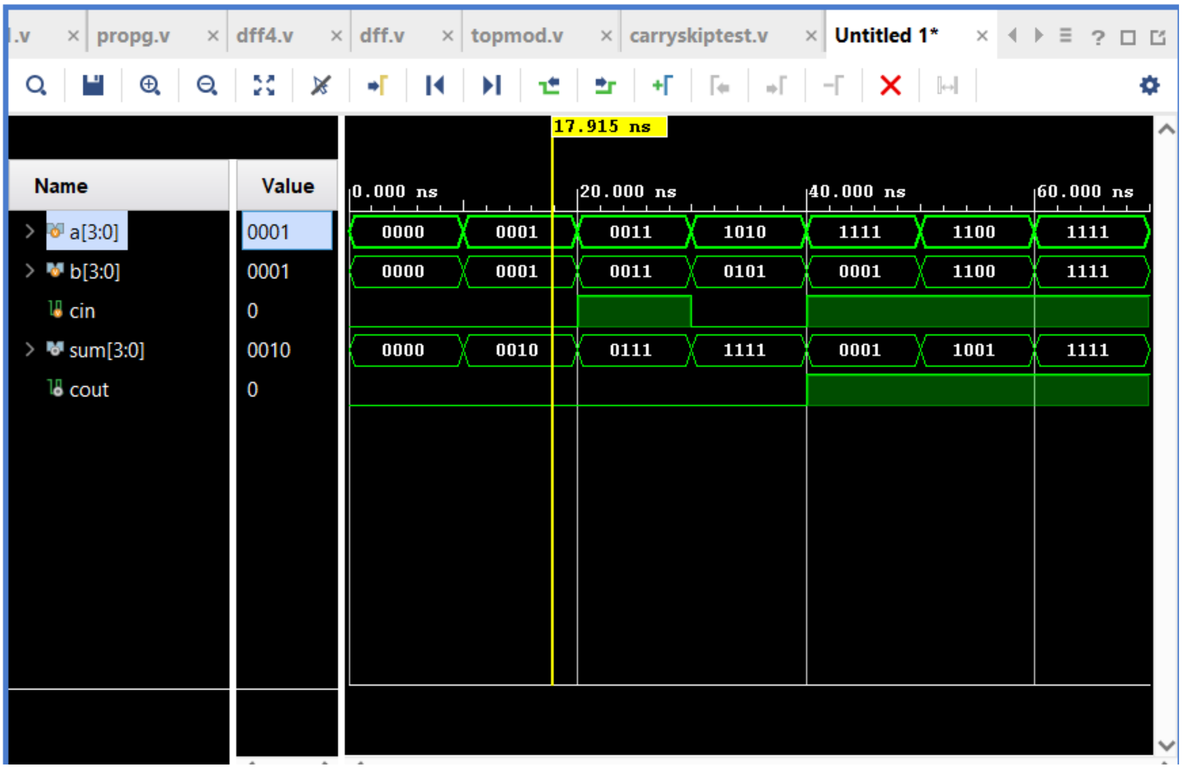
CARRY SKIP ADDER CODE:



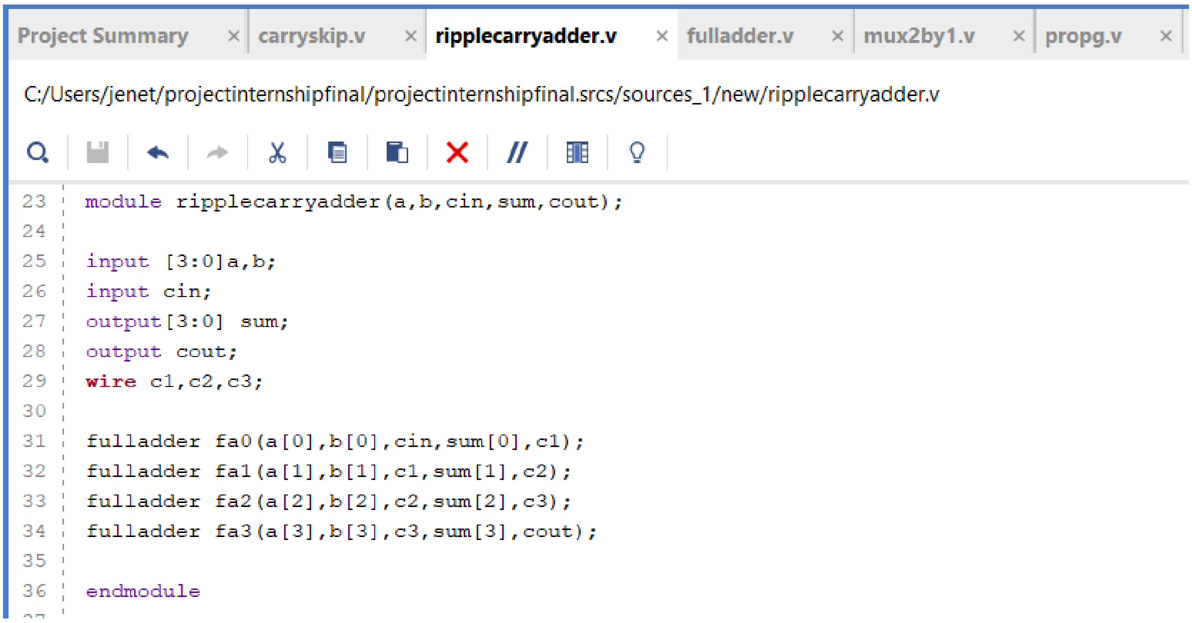
Project Summary
carryskip.v
ripplecarryadder.v
fulladder.v
mux2by1.v
propg.v
C:/Users/jenet/projectinternshipfinal/projectinternshipfinal.srcs/sources_1/new/carryskip.v

```
module carryskip(a,b,cin,sum,cout);

input [3:0] a,b;
input cin;
output [3:0] sum;
output cout;
wire [3:0] p;
wire c0;
wire bp;

ripplecarryadder rca1(.a(a[3:0]),.b(b[3:0]),.cin(cin),.sum(sum[3:0]),.cout(c0));
propg p1(a,b,p,bp);
mux2by1 m1(.I0(c0),.I1(cin),.sel(bp),.out(cout));

endmodule
```

WAVEFORM:

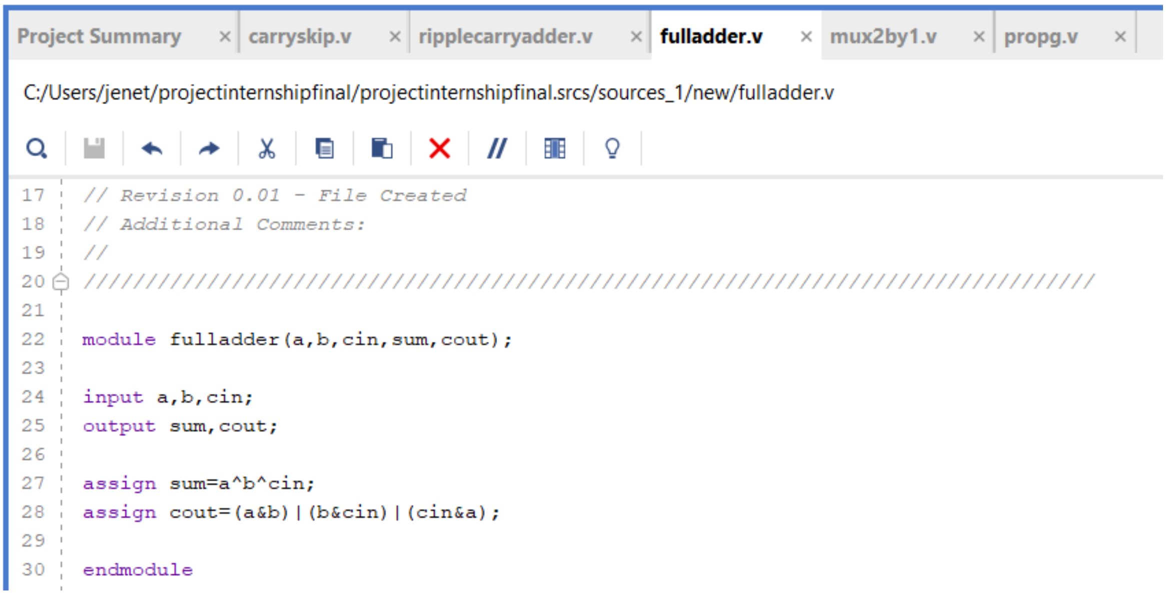


RIPPLE CARRY ADDER:


Project Summary
carryskip.v
ripplecarryadder.v
fulladder.v
mux2by1.v
propg.v
C:/Users/jenet/projectinternshipfinal/projectinternshipfinal.srcs/sources_1/new/ripplecarryadder.v

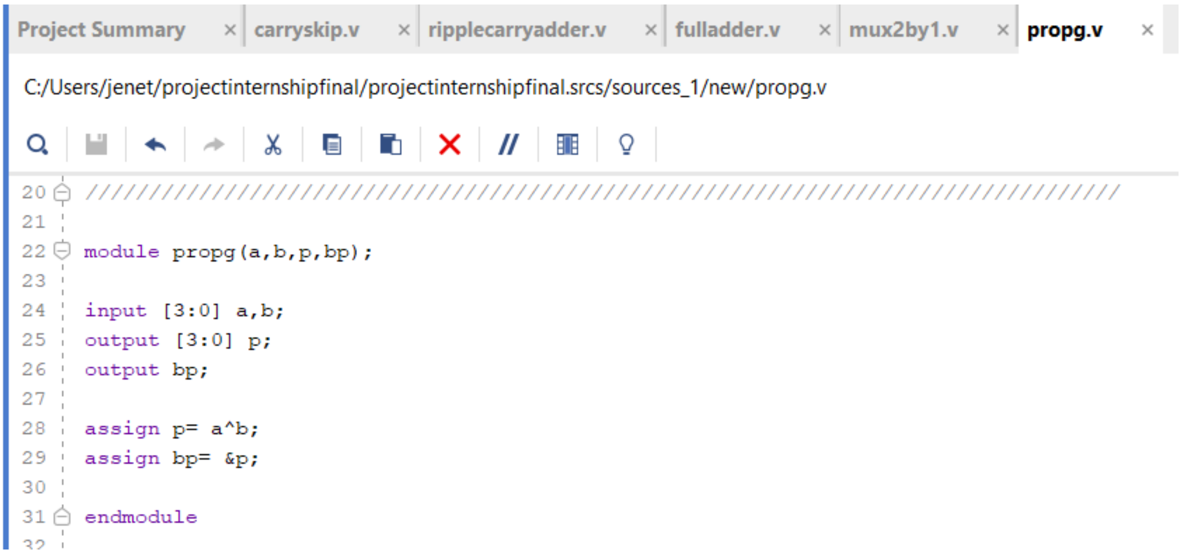
```
module ripplecarryadder(a,b,cin,sum,cout);

input [3:0]a,b;
input cin;
output[3:0] sum;
output cout;
wire c1,c2,c3;

fulladder fa0(a[0],b[0],cin,sum[0],c1);
fulladder fa1(a[1],b[1],c1,sum[1],c2);
fulladder fa2(a[2],b[2],c2,sum[2],c3);
fulladder fa3(a[3],b[3],c3,sum[3],cout);

endmodule
```

FULL ADDER:


Project Summary
carryskip.v
ripplecarryadder.v
fulladder.v
mux2by1.v
propg.v
C:/Users/jenet/projectinternshipfinal/projectinternshipfinal.srcs/sources_1/new/fulladder.v

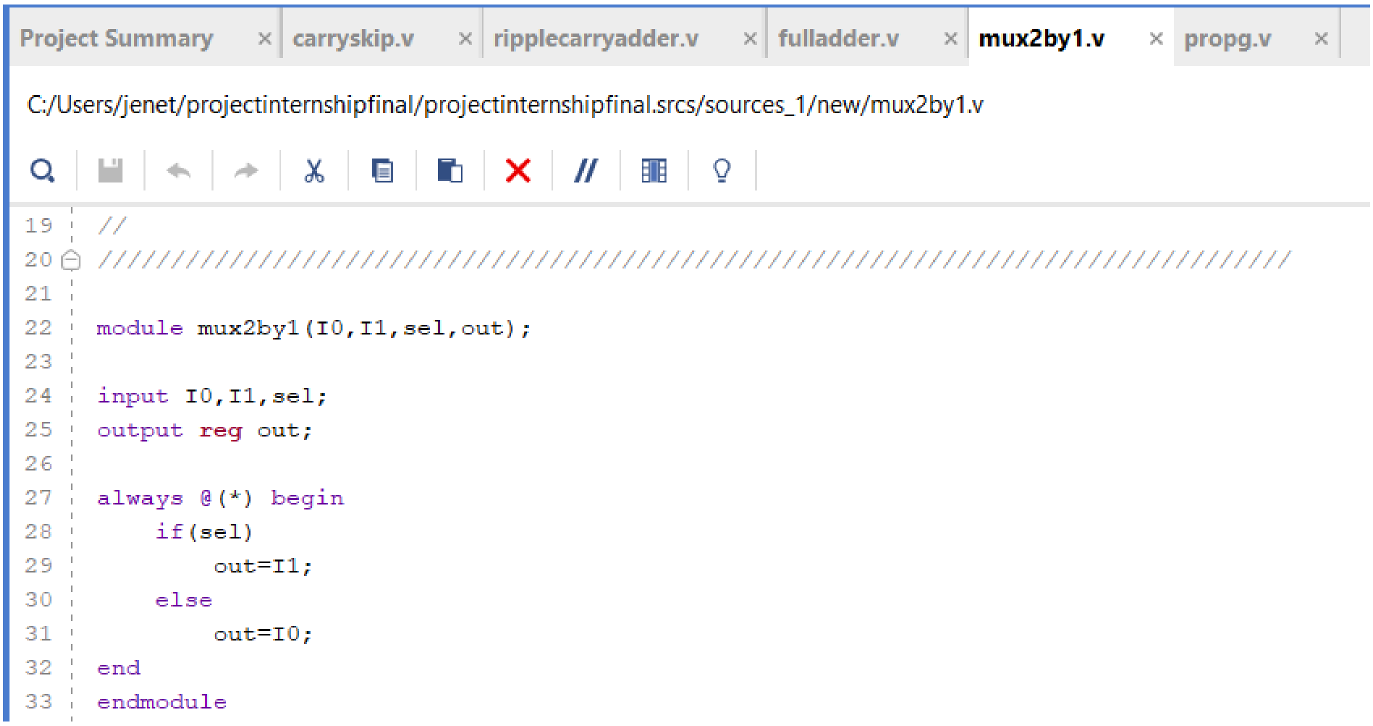
```
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module fulladder(a,b,cin,sum,cout);

input a,b,cin;
output sum,cout;

assign sum=a^b^cin;
assign cout=(a&b)|(b&cin)|(cin&a);

endmodule
```

PROPAGATE:


Project Summary
carryskip.v
ripplecarryadder.v
fulladder.v
mux2by1.v
propg.v
C:/Users/jenet/projectinternshipfinal/projectinternshipfinal.srcs/sources_1/new/propg.v

```
//////////////////////////////////////////////////////////////////////////////////

module propg(a,b,p,bp);

input [3:0] a,b;
output [3:0] p;
output bp;

assign p= a^b;
assign bp= &p;

endmodule
```

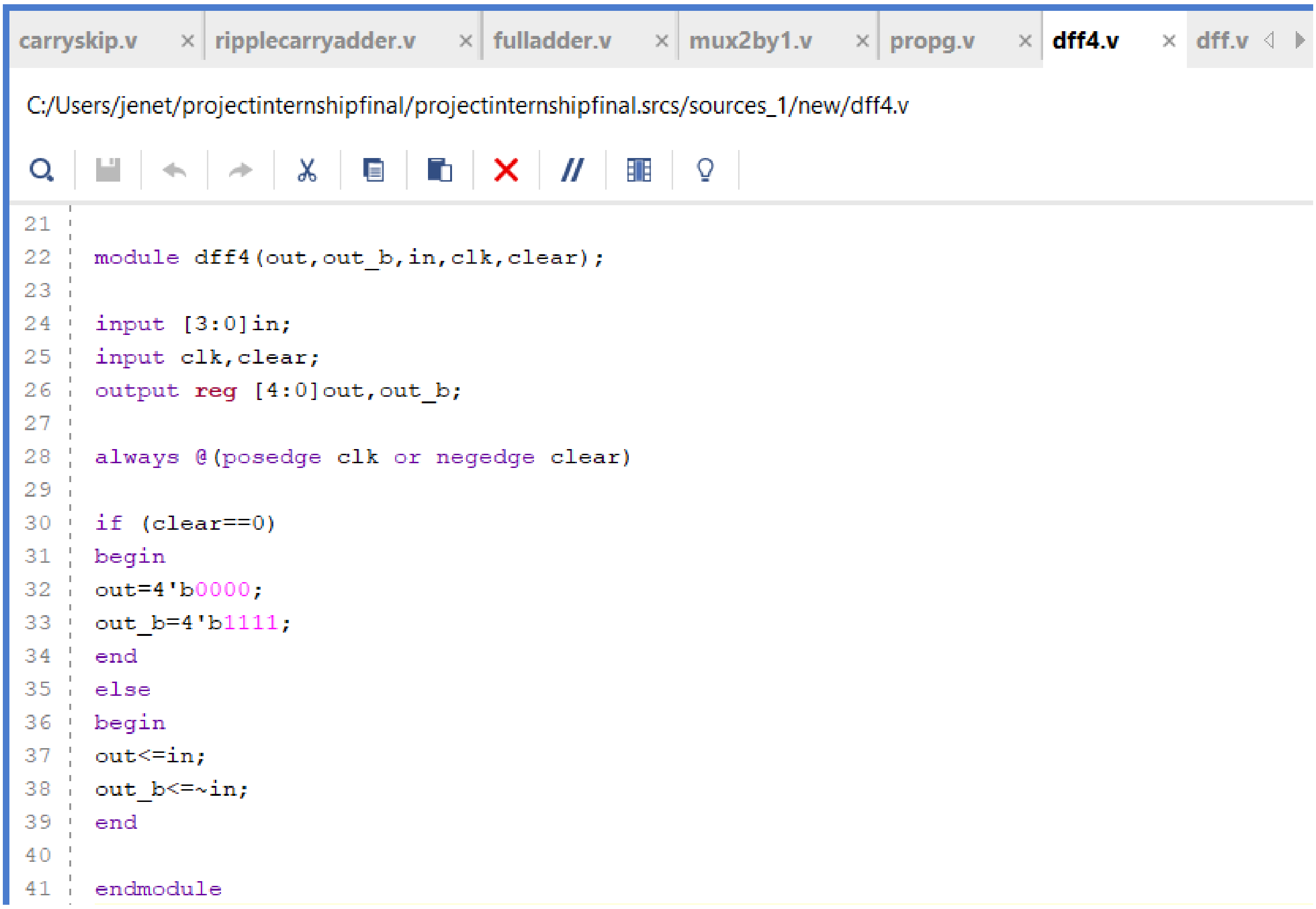
2X1 MUX:



C:/Users/jenet/projectinternshipfinal/projectinternshipfinal.srcs/sources_1/new/mux2by1.v

```
//
////////////////////////////////////////////////////////////////////////////////

module mux2by1(I0,I1,sel,out);

input I0,I1,sel;
output reg out;

always @(*) begin
    if(sel)
        out=I1;
    else
        out=I0;
end
endmodule
```

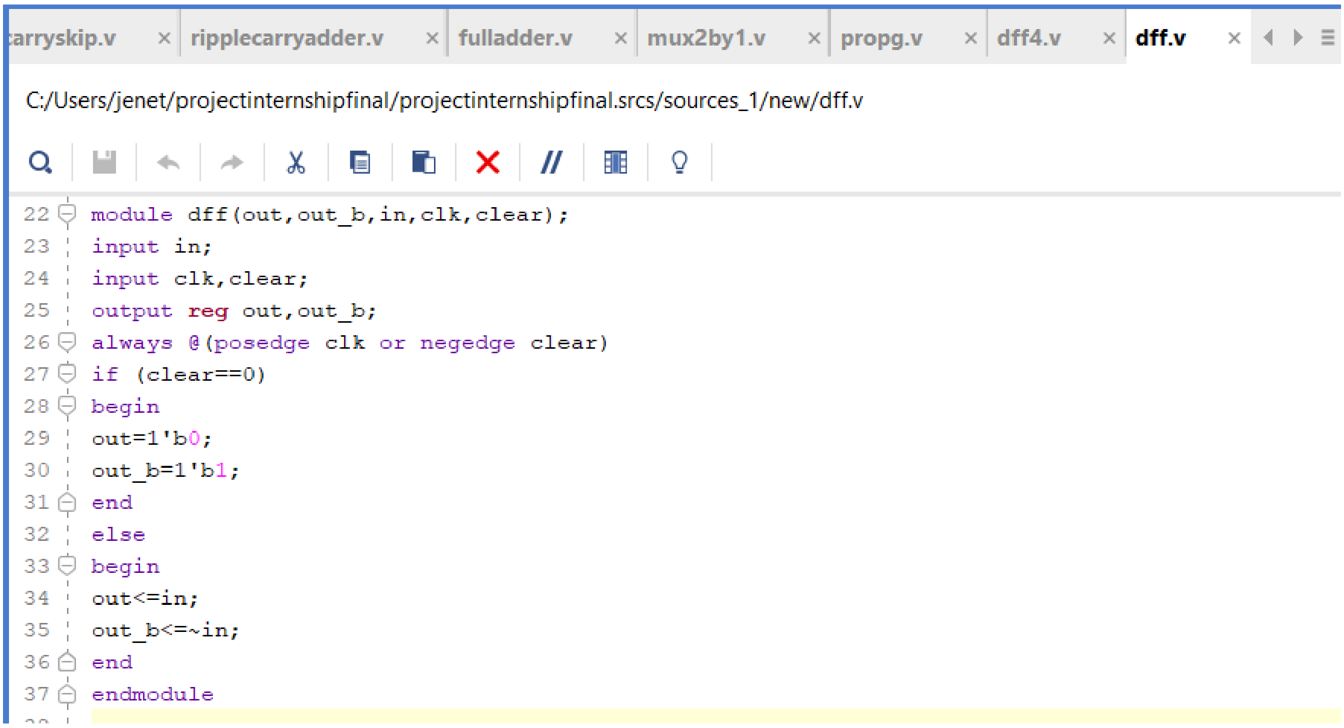
4 BIT D FLIP FLOP:



C:/Users/jenet/projectinternshipfinal/projectinternshipfinal.srcs/sources_1/new/dff4.v

```

module dff4(out,out_b,in,clk,clear);

input [3:0]in;
input clk,clear;
output reg [4:0]out,out_b;

always @(posedge clk or negedge clear)

if (clear==0)
begin
out=4'b0000;
out_b=4'b1111;
end
else
begin
out<=in;
out_b<=~in;
end

endmodule
```

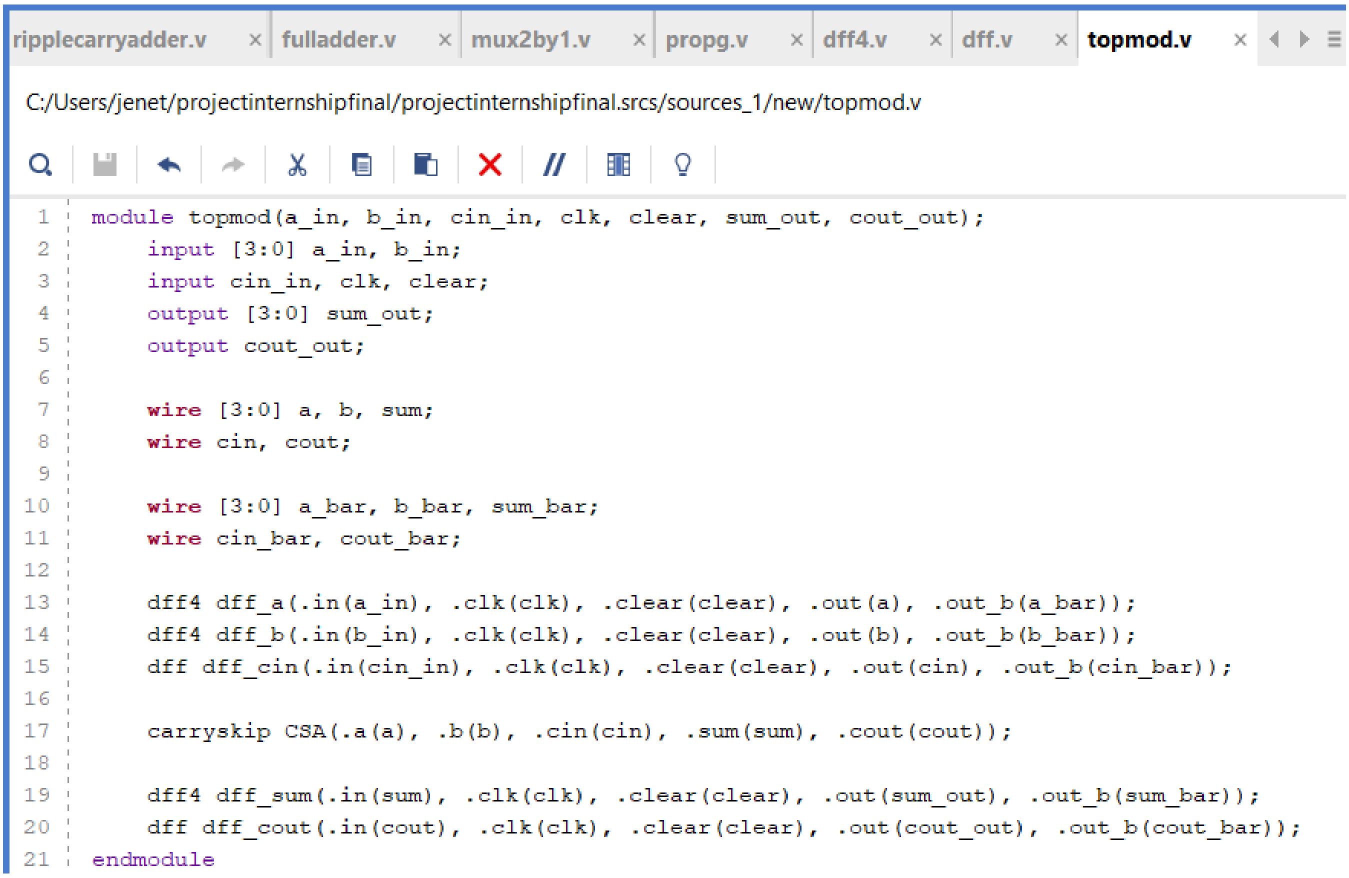
1 BIT D FLIP FLOP:



C:/Users/jenet/projectinternshipfinal/projectinternshipfinal.srcs/sources_1/new/dff.v

```
module dff(out,out_b,in,clk,clear);
input in;
input clk,clear;
output reg out,out_b;
always @(posedge clk or negedge clear)
if (clear==0)
begin
out=1'b0;
out_b=1'b1;
end
else
begin
out<=in;
out_b<=~in;
end
endmodule
```

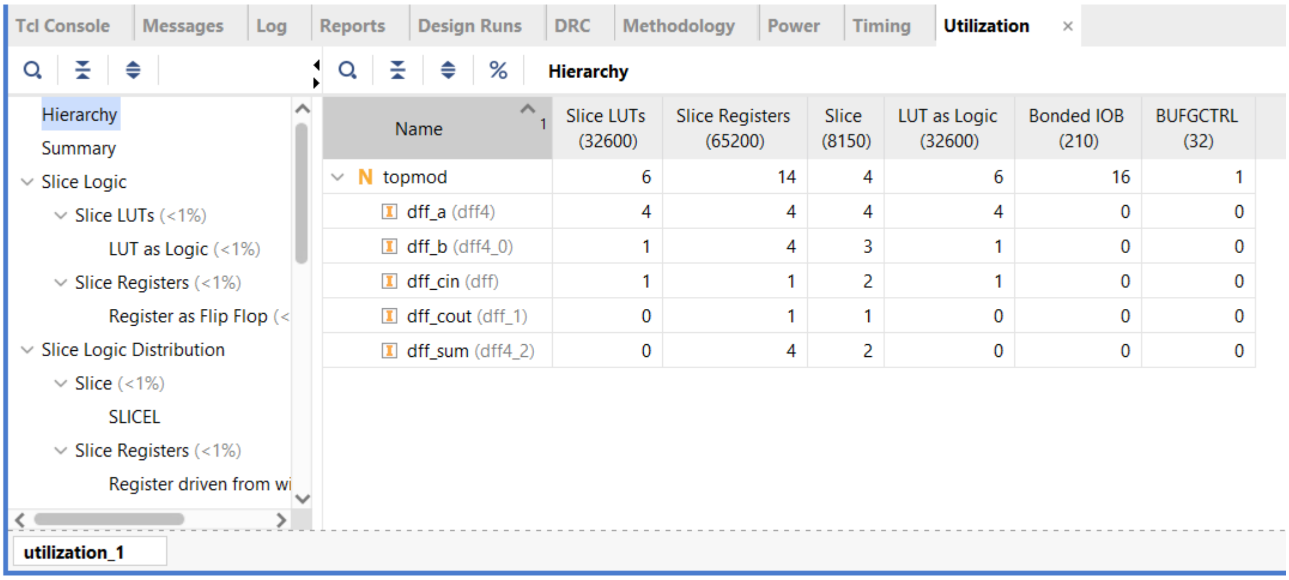
TOP MODULE:



C:/Users/jenet/projectinternshipfinal/projectinternshipfinal.srcs/sources_1/new/topmod.v

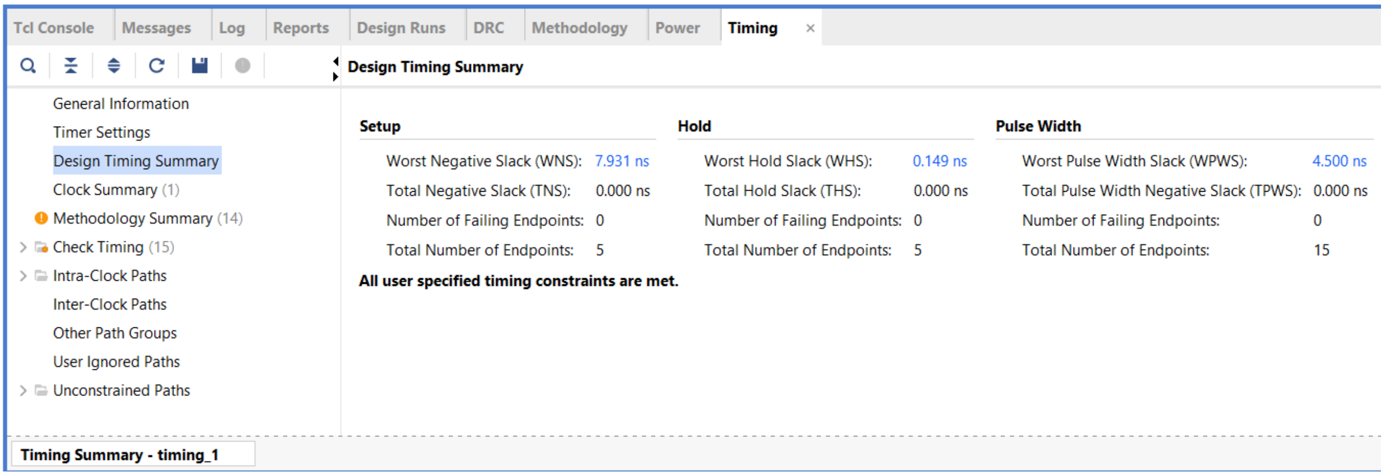
```
module topmod(a_in, b_in, cin_in, clk, clear, sum_out, cout_out);
    input [3:0] a_in, b_in;
    input cin_in, clk, clear;
    output [3:0] sum_out;
    output cout_out;

    wire [3:0] a, b, sum;
    wire cin, cout;

    wire [3:0] a_bar, b_bar, sum_bar;
    wire cin_bar, cout_bar;

    dff4 dff_a(.in(a_in), .clk(clk), .clear(clear), .out(a), .out_b(a_bar));
    dff4 dff_b(.in(b_in), .clk(clk), .clear(clear), .out(b), .out_b(b_bar));
    dff dff_cin(.in(cin_in), .clk(clk), .clear(clear), .out(cin), .out_b(cin_bar));

    carryskip CSA(.a(a), .b(b), .cin(cin), .sum(sum), .cout(cout));

    dff4 dff_sum(.in(sum), .clk(clk), .clear(clear), .out(sum_out), .out_b(sum_bar));
    dff dff_cout(.in(cout), .clk(clk), .clear(clear), .out(cout_out), .out_b(cout_bar));
endmodule
```

UTILIZATION AREA SUMMARY:



Name	Slice LUTs (32600)	Slice Registers (65200)	Slice (8150)	LUT as Logic (32600)	Bonded IOB (210)	BUFGCTRL (32)
N topmod	6	14	4	6	16	1
dff_a (dff4)	4	4	4	4	0	0
dff_b (dff4_0)	1	4	3	1	0	0
dff_cin (dff)	1	1	2	1	0	0
dff_cout (dff_1)	0	1	1	0	0	0
dff_sum (dff4_2)	0	4	2	0	0	0

TIMING REPORT:



Design Timing Summary

Setup

Worst Negative Slack (WNS): 7.931 ns
Total Negative Slack (TNS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 5

Hold

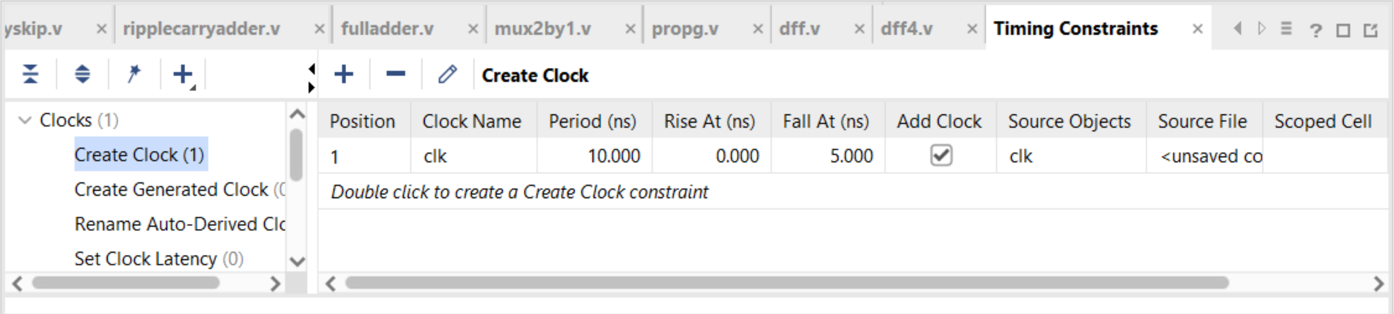
Worst Hold Slack (WHS): 0.149 ns
Total Hold Slack (THS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 5

Pulse Width

Worst Pulse Width Slack (WPWS): 4.500 ns
Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 15

All user specified timing constraints are met.

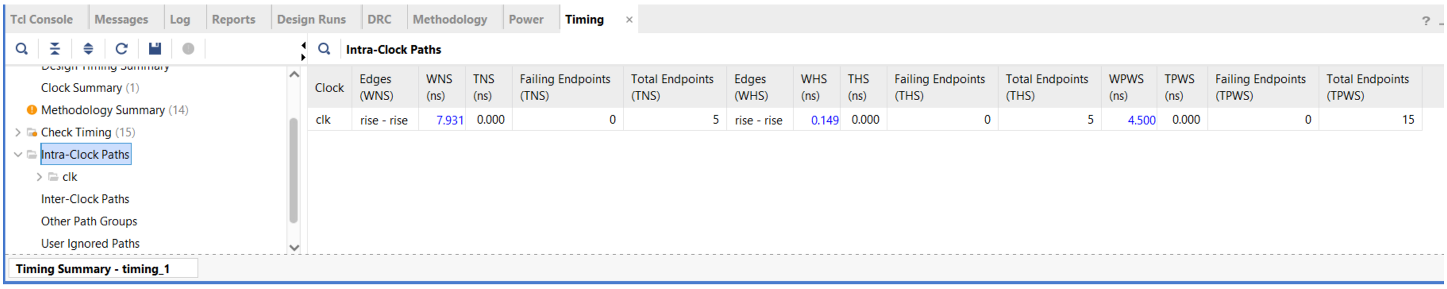
TIMING CONSTRAINTS:



Position	Clock Name	Period (ns)	Rise At (ns)	Fall At (ns)	Add Clock	Source Objects	Source File	Scoped Cell
1	clk	10.000	0.000	5.000	☑	clk	<unsaved co	

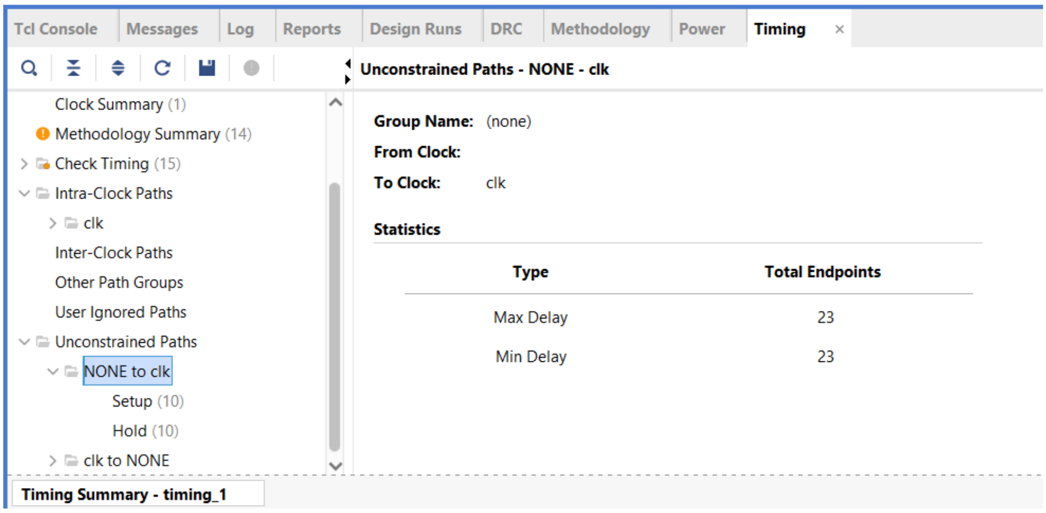
Double click to create a Create Clock constraint

INTRA CLOCK PATHS:



Clock	Edges (WNS)	WNS (ns)	TNS (ns)	Failing Endpoints (TNS)	Total Endpoints (TNS)	Edges (WHS)	WHS (ns)	THS (ns)	Failing Endpoints (THS)	Total Endpoints (THS)	WPWS (ns)	TPWS (ns)	Failing Endpoints (TPWS)	Total Endpoints (TPWS)
clk	rise - rise	7.931	0.000	0	5	rise - rise	0.149	0.000	0	5	4.500	0.000	0	15

UNCONSTRAINED PATHS:



Unconstrained Paths - NONE - clk

- Clock Summary (1)
- Methodology Summary (14)
- Check Timing (15)
- Intra-Clock Paths
  - clk
- Inter-Clock Paths
- Other Path Groups
- User Ignored Paths
- Unconstrained Paths
  - NONE to clk
    - Setup (10)
    - Hold (10)
  - clk to NONE

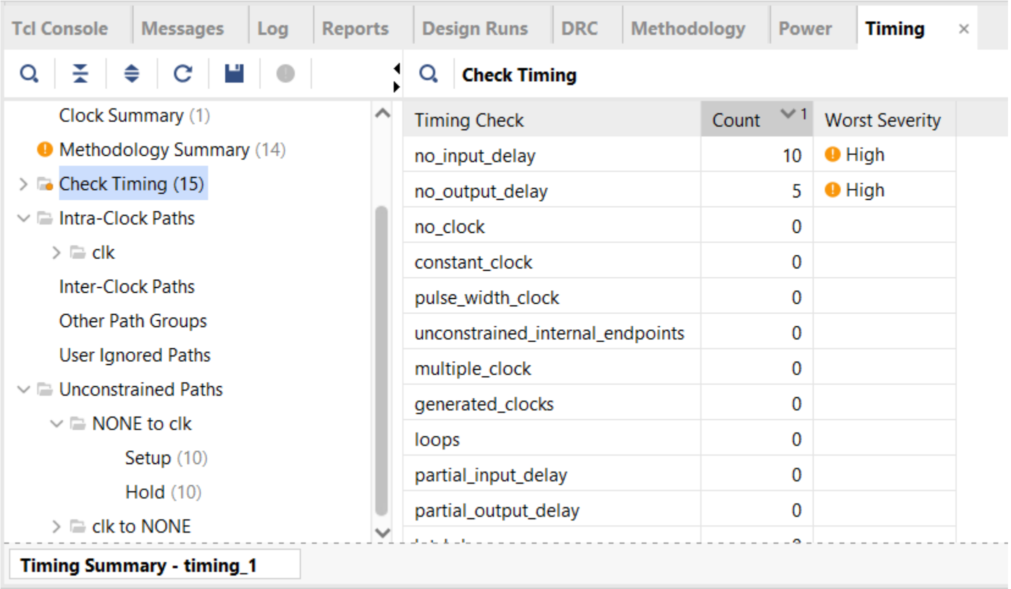
Group Name: (none)
From Clock:
To Clock: clk

Statistics

Type	Total Endpoints
Max Delay	23
Min Delay	23

Timing Summary - timing_1

DELAYS:



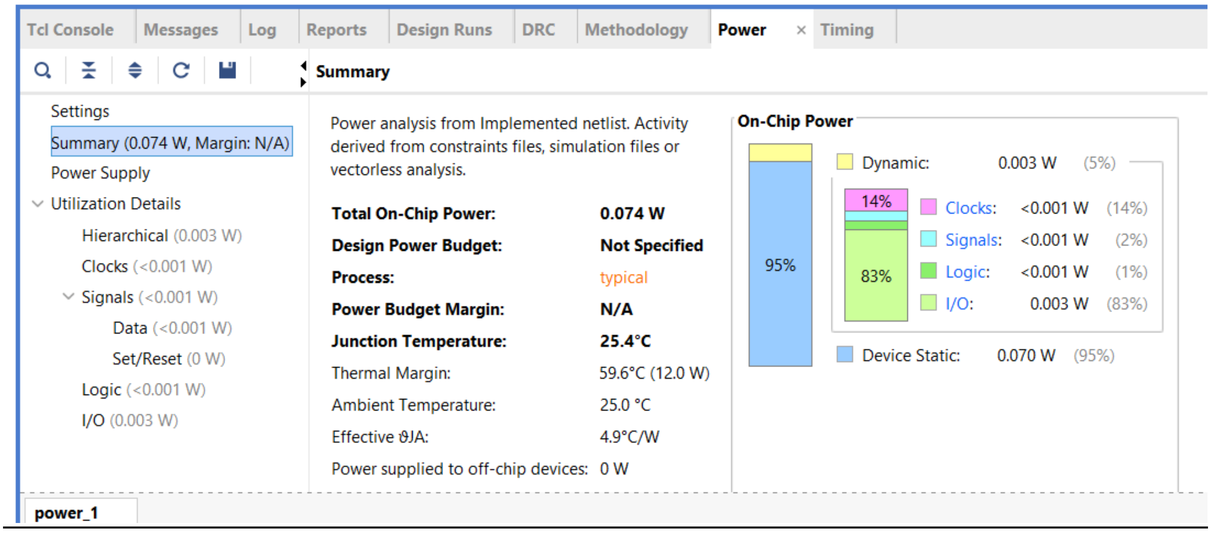
Check Timing

- Clock Summary (1)
- Methodology Summary (14)
- Check Timing (15)
- Intra-Clock Paths
  - clk
- Inter-Clock Paths
- Other Path Groups
- User Ignored Paths
- Unconstrained Paths
  - NONE to clk
    - Setup (10)
    - Hold (10)
  - clk to NONE

Timing Check	Count	Worst Severity
no_input_delay	10	High
no_output_delay	5	High
no_clock	0	
constant_clock	0	
pulse_width_clock	0	
unconstrained_internal_endpoints	0	
multiple_clock	0	
generated_clocks	0	
loops	0	
partial_input_delay	0	
partial_output_delay	0	

Timing Summary - timing_1

POWER REPORT:



Summary

- Settings
- Summary (0.074 W, Margin: N/A)
- Power Supply
- Utilization Details
  - Hierarchical (0.003 W)
  - Clocks (<0.001 W)
  - Signals (<0.001 W)
    - Data (<0.001 W)
    - Set/Reset (0 W)
  - Logic (<0.001 W)
  - I/O (0.003 W)

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.074 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**25.4°C**
Thermal Margin:	59.6°C (12.0 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	4.9°C/W
Power supplied to off-chip devices:	0 W

On-Chip Power

Dynamic:	0.003 W	(5%)
Clocks:	<0.001 W	(14%)
Signals:	<0.001 W	(2%)
Logic:	<0.001 W	(1%)
I/O:	0.003 W	(83%)
Device Static:	0.070 W	(95%)

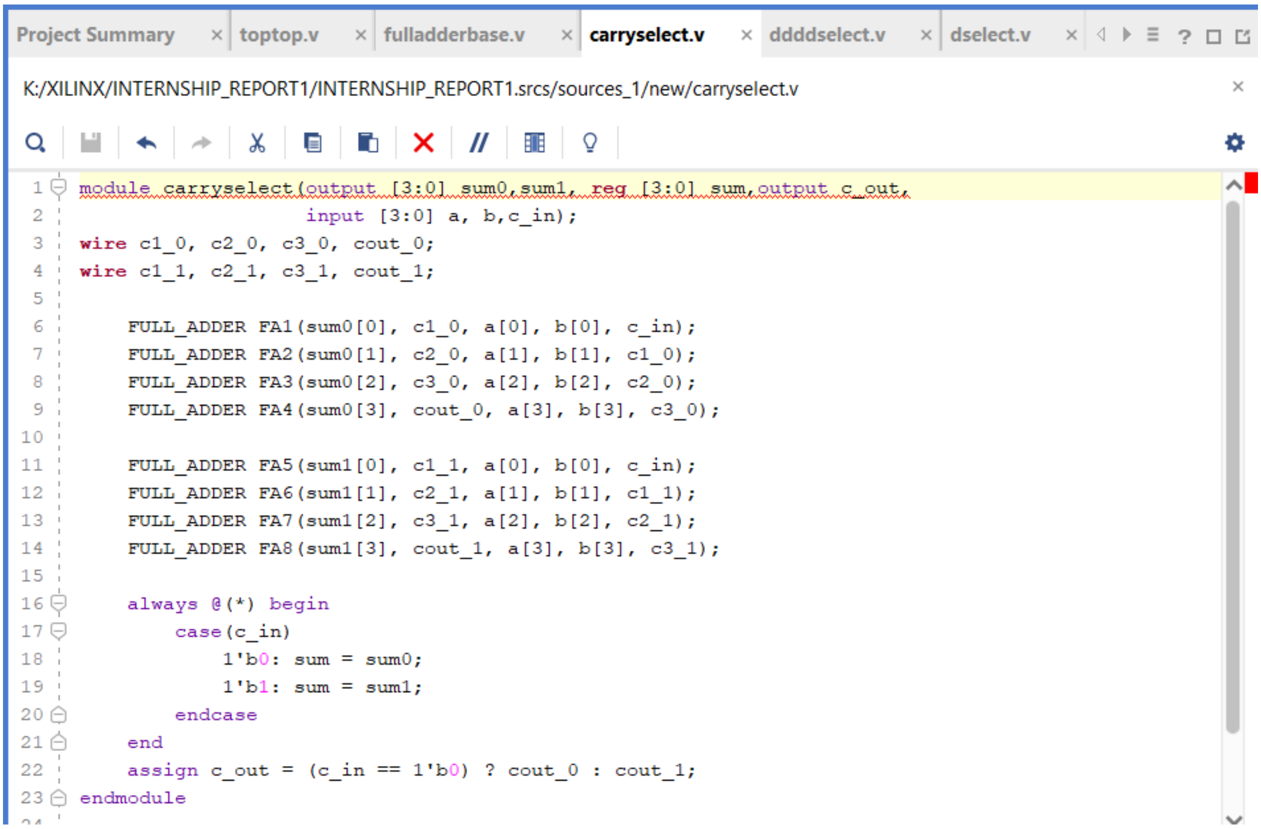
power_1

CARRY SELECT ADDER REPORT

NAME: DHARSHAN S

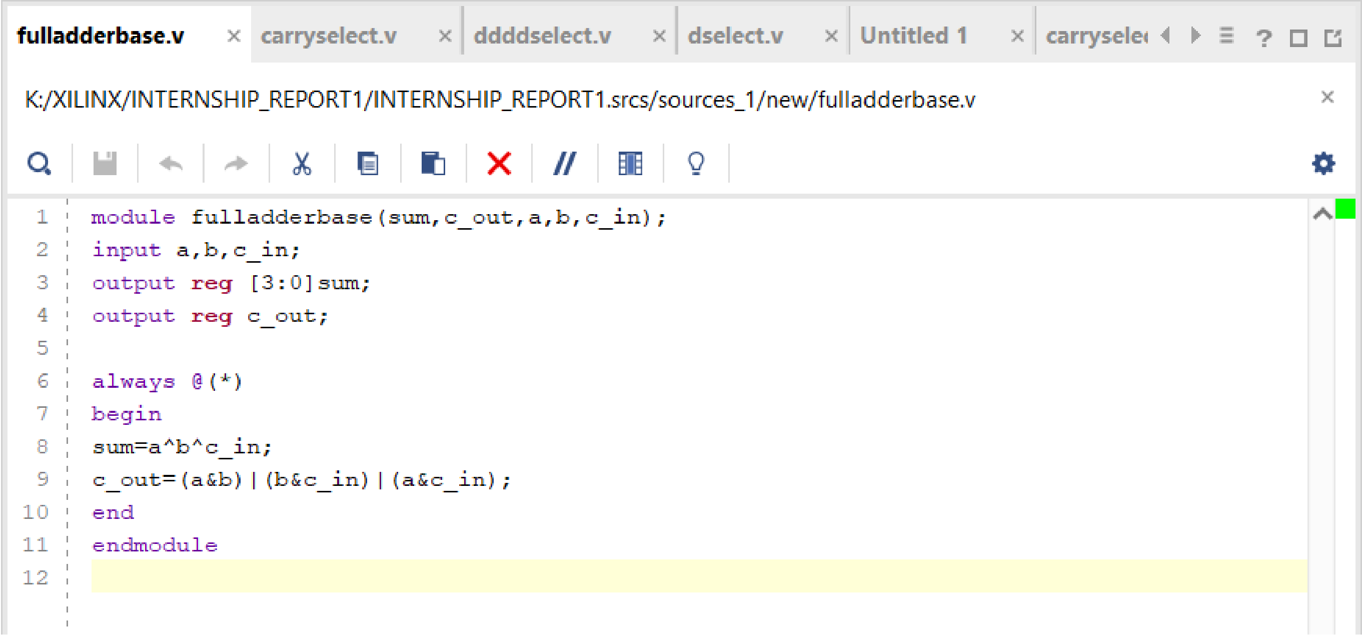
GMAIL: dharshans640@gmail.com

CARRY SELECT ADDER CODE:


Project Summary
toptop.v
fulladderbase.v
carryselect.v
ddddselect.v
dselect.v
K:/XILINX/INTERNSHIP_REPORT1/INTERNSHIP_REPORT1.srcs/sources_1/new/carryselect.v

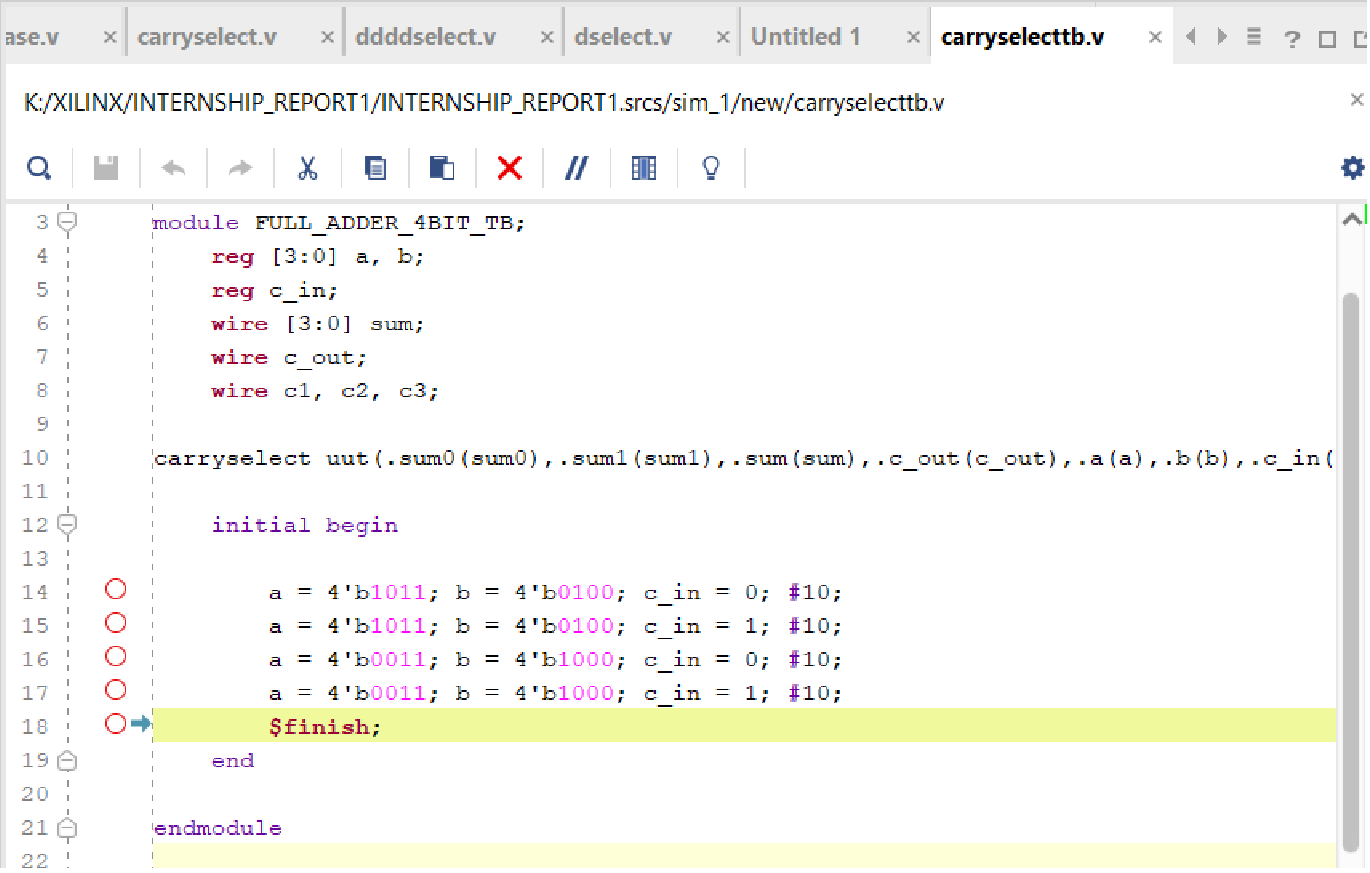
```
module carryselect(output [3:0] sum0,sum1, reg [3:0] sum,output c_out,
                   input [3:0] a, b,c_in);
wire c1_0, c2_0, c3_0, cout_0;
wire c1_1, c2_1, c3_1, cout_1;

    FULL_ADDER FA1(sum0[0], c1_0, a[0], b[0], c_in);
    FULL_ADDER FA2(sum0[1], c2_0, a[1], b[1], c1_0);
    FULL_ADDER FA3(sum0[2], c3_0, a[2], b[2], c2_0);
    FULL_ADDER FA4(sum0[3], cout_0, a[3], b[3], c3_0);

    FULL_ADDER FA5(sum1[0], c1_1, a[0], b[0], c_in);
    FULL_ADDER FA6(sum1[1], c2_1, a[1], b[1], c1_1);
    FULL_ADDER FA7(sum1[2], c3_1, a[2], b[2], c2_1);
    FULL_ADDER FA8(sum1[3], cout_1, a[3], b[3], c3_1);

    always @(*) begin
        case(c_in)
            1'b0: sum = sum0;
            1'b1: sum = sum1;
        endcase
    end
    assign c_out = (c_in == 1'b0) ? cout_0 : cout_1;
endmodule
```

ONE BIT ADDER:


fulladderbase.v
carryselect.v
ddddselect.v
dselect.v
Untitled 1
carrysele
K:/XILINX/INTERNSHIP_REPORT1/INTERNSHIP_REPORT1.srcs/sources_1/new/fulladderbase.v

```
module fulladderbase(sum,c_out,a,b,c_in);
input a,b,c_in;
output reg [3:0]sum;
output reg c_out;

always @(*)
begin
sum=a^b^c_in;
c_out=(a&b)|(b&c_in)|(a&c_in);
end
endmodule
```

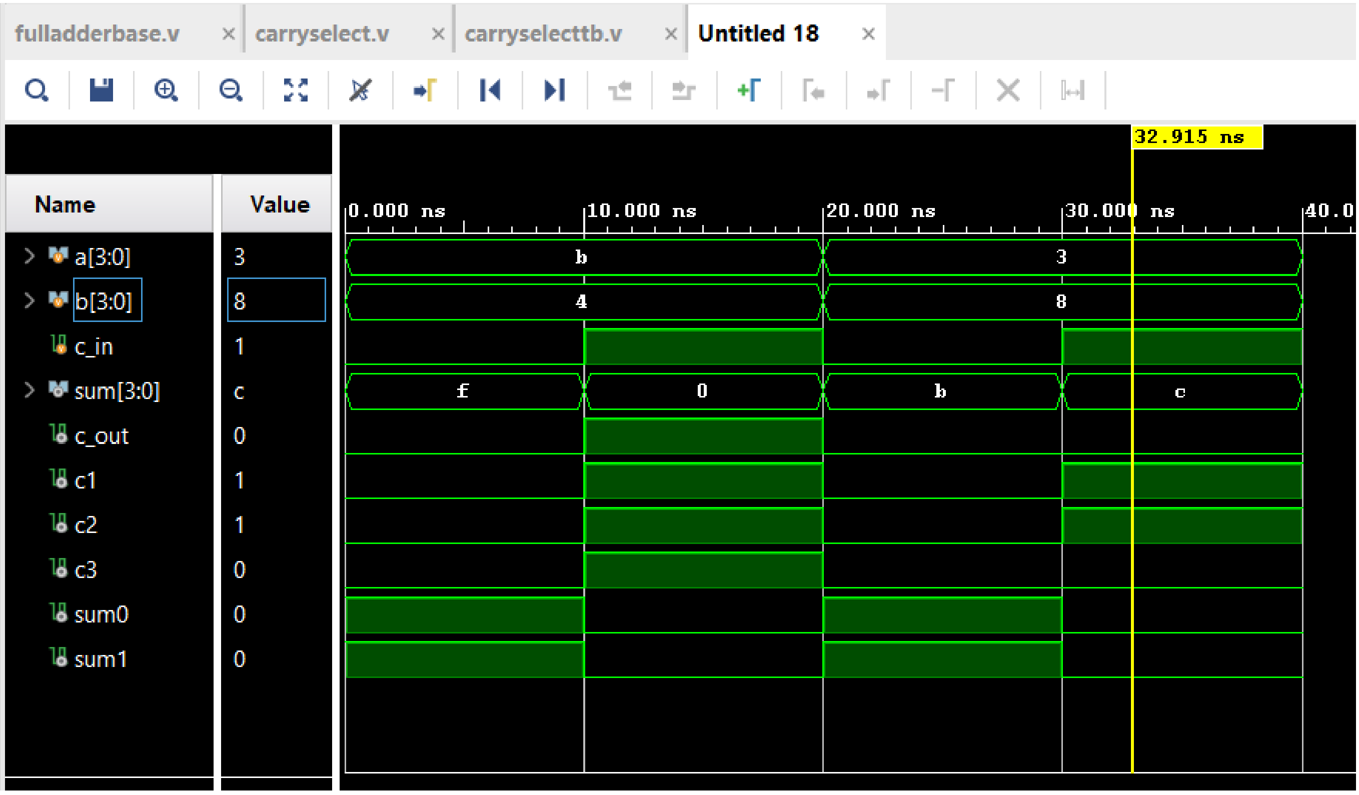
TEST BENCH:



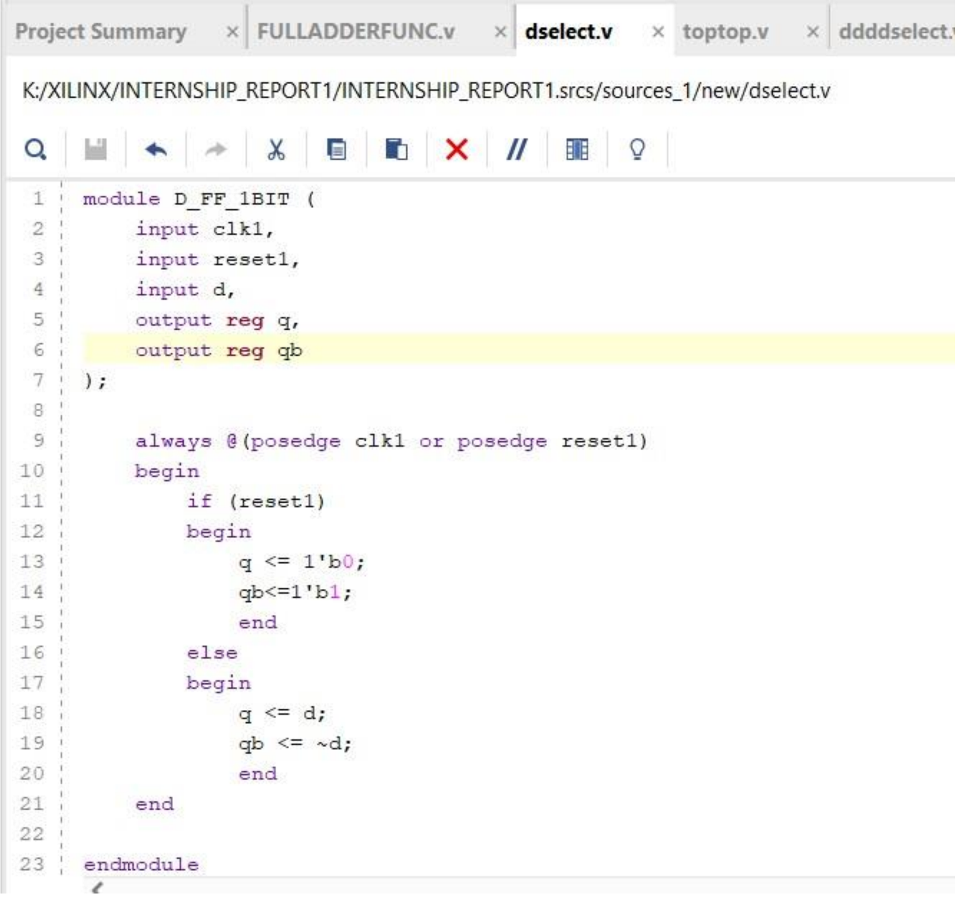
ase.v
carryselect.v
ddddselect.v
dselect.v
Untitled 1
carryselecttb.v
K:/XILINX/INTERNSHIP_REPORT1/INTERNSHIP_REPORT1.srcs/sim_1/new/carryselecttb.v

```
module FULL_ADDER_4BIT_TB;
    reg [3:0] a, b;
    reg c_in;
    wire [3:0] sum;
    wire c_out;
    wire c1, c2, c3;

carryselect uut(.sum0(sum0),.sum1(sum1),.sum(sum),.c_out(c_out),.a(a),.b(b),.c_in(

    initial begin

        a = 4'b1011; b = 4'b0100; c_in = 0; #10;
        a = 4'b1011; b = 4'b0100; c_in = 1; #10;
        a = 4'b0011; b = 4'b1000; c_in = 0; #10;
        a = 4'b0011; b = 4'b1000; c_in = 1; #10;
        $finish;
    end

endmodule
```

WAVEFORM:

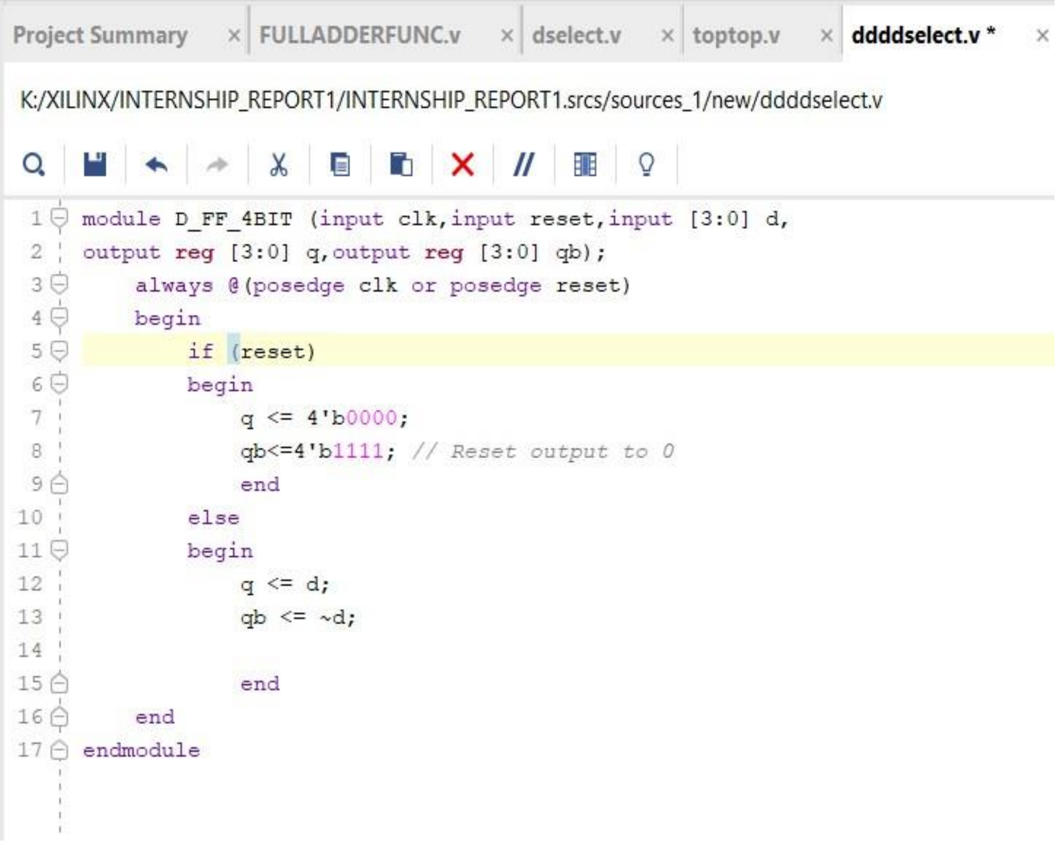


ONE BIT D FLIPFLOP:


Project Summary
FULLADDERFUNC.v
dselect.v
toptop.v
ddddselect.
K:/XILINX/INTERNSHIP_REPORT1/INTERNSHIP_REPORT1.srcs/sources_1/new/dselect.v

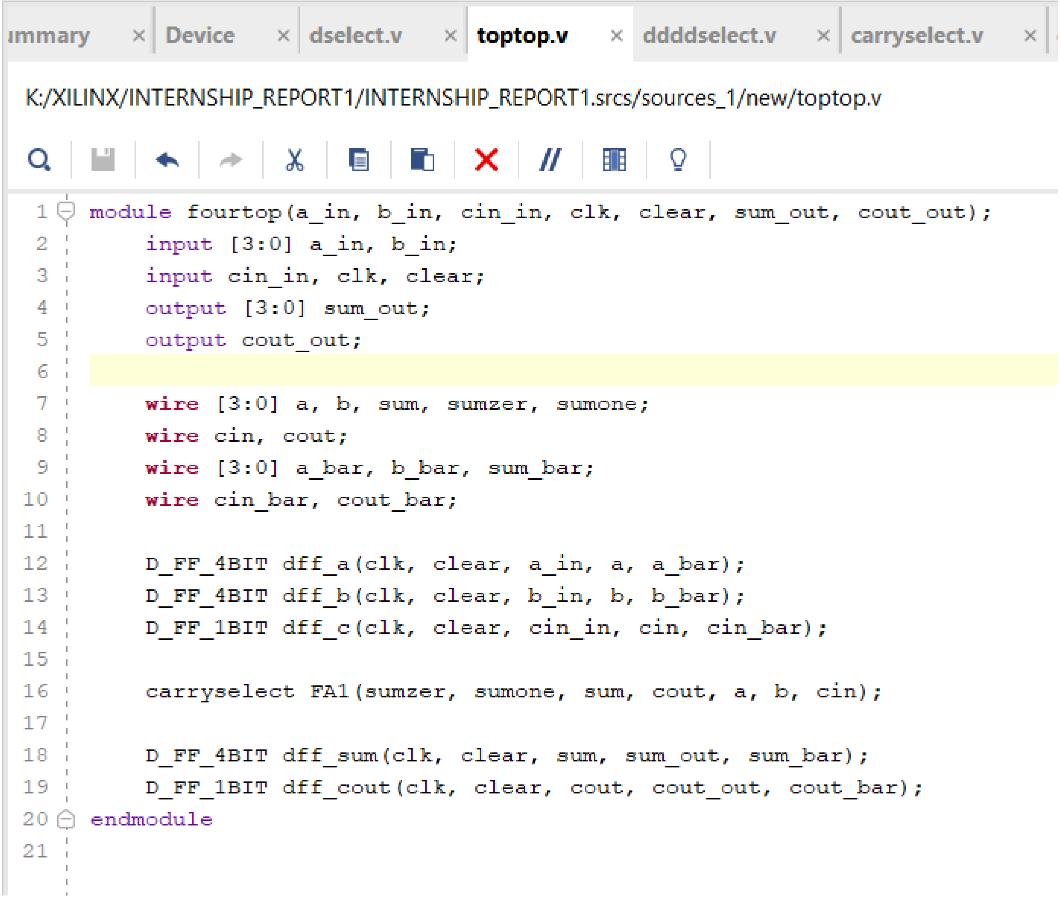
```
module D_FF_1BIT (
    input clk1,
    input reset1,
    input d,
    output reg q,
    output reg qb
);

    always @(posedge clk1 or posedge reset1)
    begin
        if (reset1)
        begin
            q <= 1'b0;
            qb<=1'b1;
            end
        else
        begin
            q <= d;
            qb <= ~d;
            end
    end

endmodule
```

FOUR BIT D FLIPFLOP:


Project Summary
FULLADDERFUNC.v
dselect.v
toptop.v
ddddselect.v *
K:/XILINX/INTERNSHIP_REPORT1/INTERNSHIP_REPORT1.srcs/sources_1/new/ddddselect.v

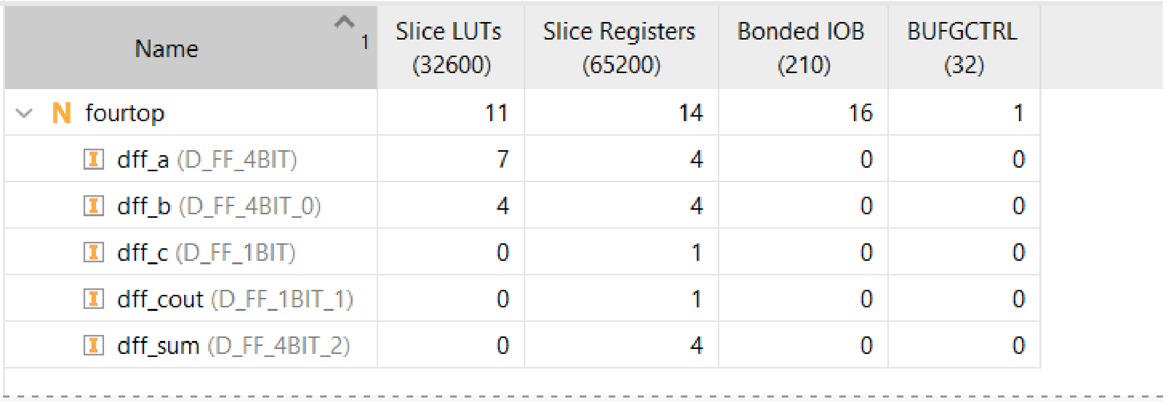
```
module D_FF_4BIT (input clk,input reset,input [3:0] d,
output reg [3:0] q,output reg [3:0] qb);
    always @(posedge clk or posedge reset)
    begin
        if (reset)
        begin
            q <= 4'b0000;
            qb<=4'b1111; // Reset output to 0
            end
        else
        begin
            q <= d;
            qb <= ~d;

            end
    end
endmodule
```

TOPMODULE:


ımmary
Device
dselect.v
toptop.v
ddddselect.v
carryselect.v
K:/XILINX/INTERNSHIP_REPORT1/INTERNSHIP_REPORT1.srcs/sources_1/new/toptop.v

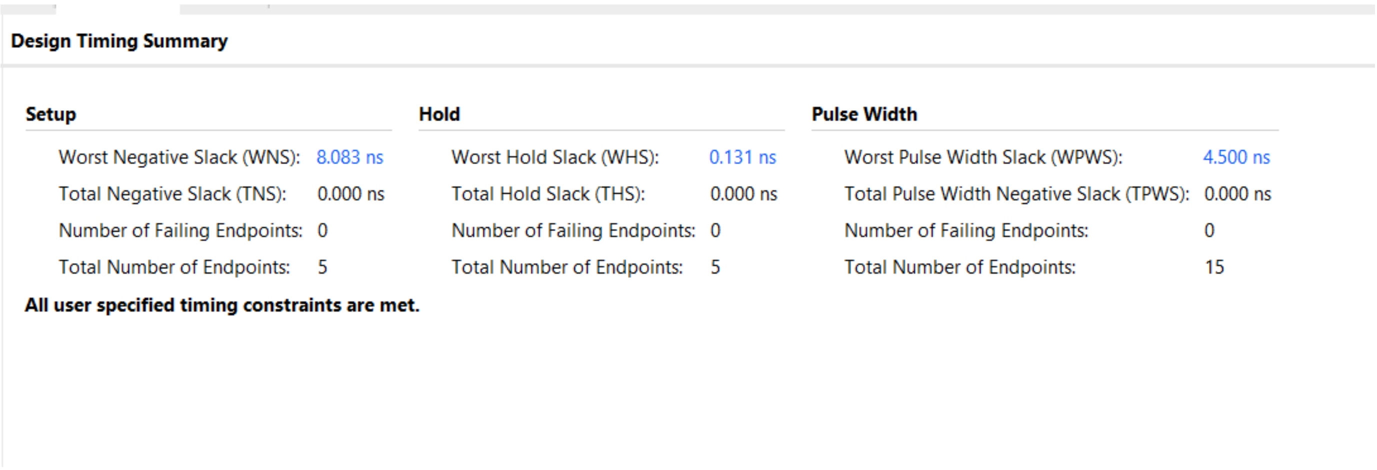
```verilog
module fourtop(a_in, b_in, cin_in, clk, clear, sum_out, cout_out);
    input [3:0] a_in, b_in;
    input cin_in, clk, clear;
    output [3:0] sum_out;
    output cout_out;

    wire [3:0] a, b, sum, sumzer, sumone;
    wire cin, cout;
    wire [3:0] a_bar, b_bar, sum_bar;
    wire cin_bar, cout_bar;

    D_FF_4BIT dff_a(clk, clear, a_in, a, a_bar);
    D_FF_4BIT dff_b(clk, clear, b_in, b, b_bar);
    D_FF_1BIT dff_c(clk, clear, cin_in, cin, cin_bar);

    carryselect FA1(sumzer, sumone, sum, cout, a, b, cin);

    D_FF_4BIT dff_sum(clk, clear, sum, sum_out, sum_bar);
    D_FF_1BIT dff_cout(clk, clear, cout, cout_out, cout_bar);
endmodule
```

UTILIZATION AREA SUMMARY:



Name	Slice LUTs (32600)	Slice Registers (65200)	Bonded IOB (210)	BUFGCTRL (32)
fourtop	11	14	16	1
dff_a (D_FF_4BIT)	7	4	0	0
dff_b (D_FF_4BIT_0)	4	4	0	0
dff_c (D_FF_1BIT)	0	1	0	0
dff_cout (D_FF_1BIT_1)	0	1	0	0
dff_sum (D_FF_4BIT_2)	0	4	0	0

TIMING REPORTS :



Design Timing Summary

Setup

Worst Negative Slack (WNS):	8.083 ns
Total Negative Slack (TNS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	5

Hold

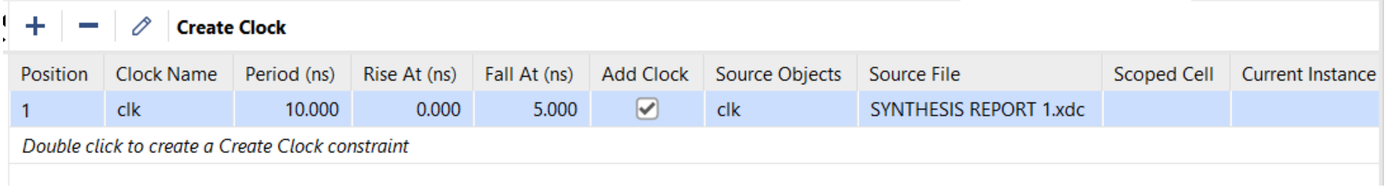
Worst Hold Slack (WHS):	0.131 ns
Total Hold Slack (THS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	5

Pulse Width

Worst Pulse Width Slack (WPWS):	4.500 ns
Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	15

All user specified timing constraints are met.

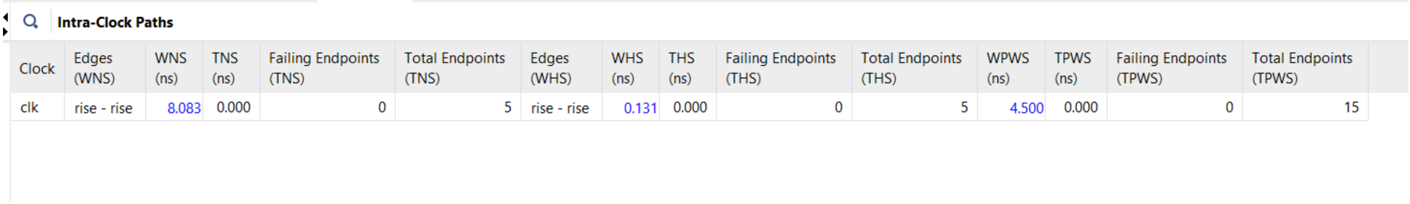
TIMING CONSTRAINTS:



Position	Clock Name	Period (ns)	Rise At (ns)	Fall At (ns)	Add Clock	Source Objects	Source File	Scoped Cell	Current Instance
1	clk	10.000	0.000	5.000	☑	clk	SYNTHESIS REPORT 1.xdc		

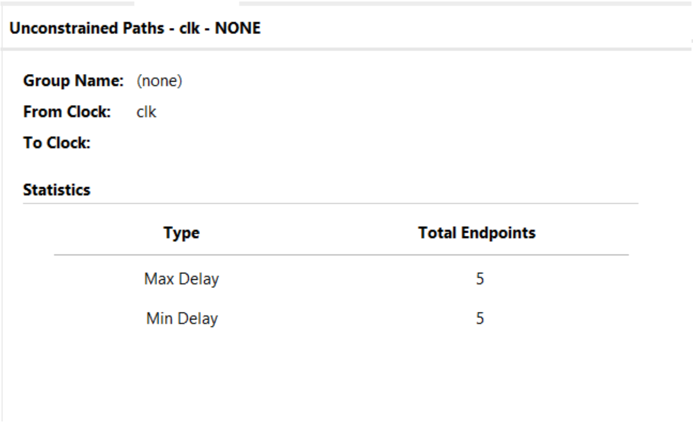
Double click to create a Create Clock constraint

INTRA CLOCK PATHS:



Clock	Edges (WNS)	WNS (ns)	TNS (ns)	Failing Endpoints (TNS)	Total Endpoints (TNS)	Edges (WHS)	WHS (ns)	THS (ns)	Failing Endpoints (THS)	Total Endpoints (THS)	WPWS (ns)	TPWS (ns)	Failing Endpoints (TPWS)	Total Endpoints (TPWS)
clk	rise - rise	8.083	0.000	0	5	rise - rise	0.131	0.000	0	5	4.500	0.000	0	15

UNCONSTRAINED PATHS:

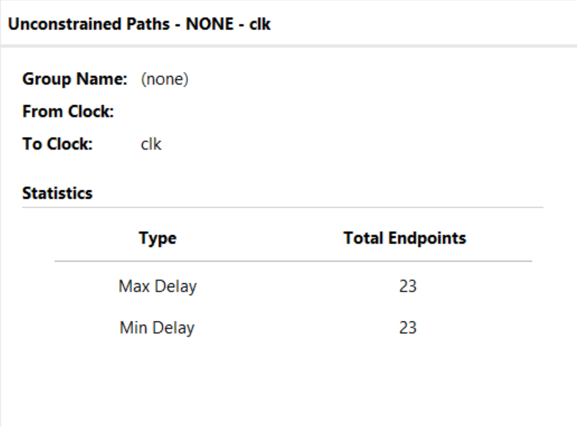


Unconstrained Paths - clk - NONE

Group Name: (none)
From Clock: clk
To Clock:

Statistics

Type	Total Endpoints
Max Delay	5
Min Delay	5



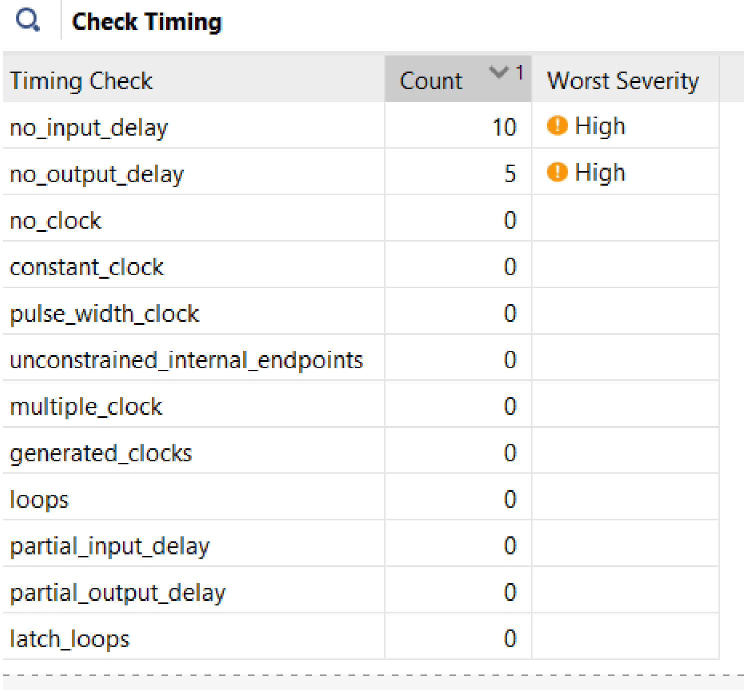
Unconstrained Paths - NONE - clk

Group Name: (none)
From Clock:
To Clock: clk

Statistics

Type	Total Endpoints
Max Delay	23
Min Delay	23

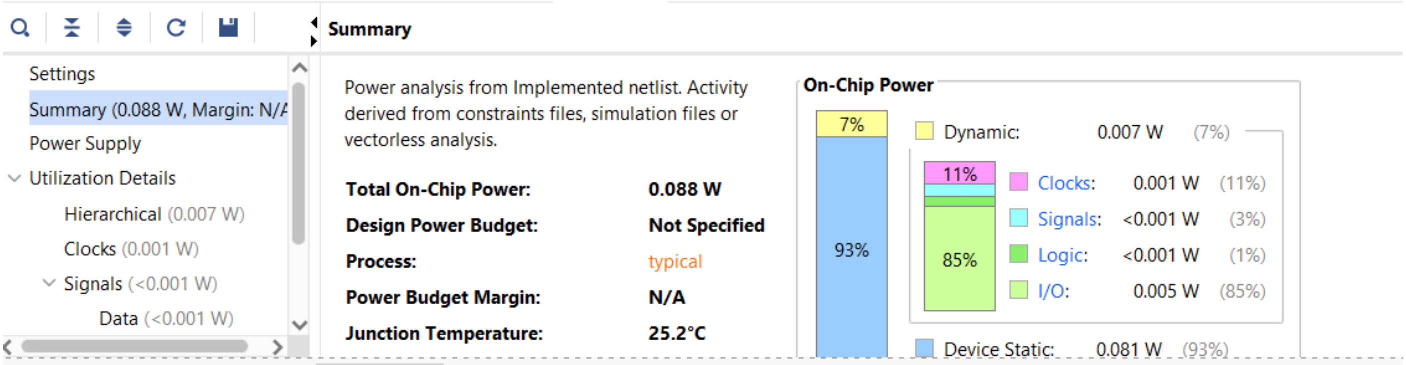
DELAY :



Check Timing

Timing Check	Count	Worst Severity
no_input_delay	10	! High
no_output_delay	5	! High
no_clock	0	
constant_clock	0	
pulse_width_clock	0	
unconstrained_internal_endpoints	0	
multiple_clock	0	
generated_clocks	0	
loops	0	
partial_input_delay	0	
partial_output_delay	0	
latch_loops	0	

POWER REPORT:



Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.088 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**25.2°C**

Dynamic:		0.007 W	(7%)
	Clocks:	0.001 W	(11%)
	Signals:	<0.001 W	(3%)
	Logic:	<0.001 W	(1%)
	I/O:	0.005 W	(85%)
Device Static:		0.081 W	(93%)

8 Conclusion

This project successfully compared four different adder architectures based on real synthesis reports.