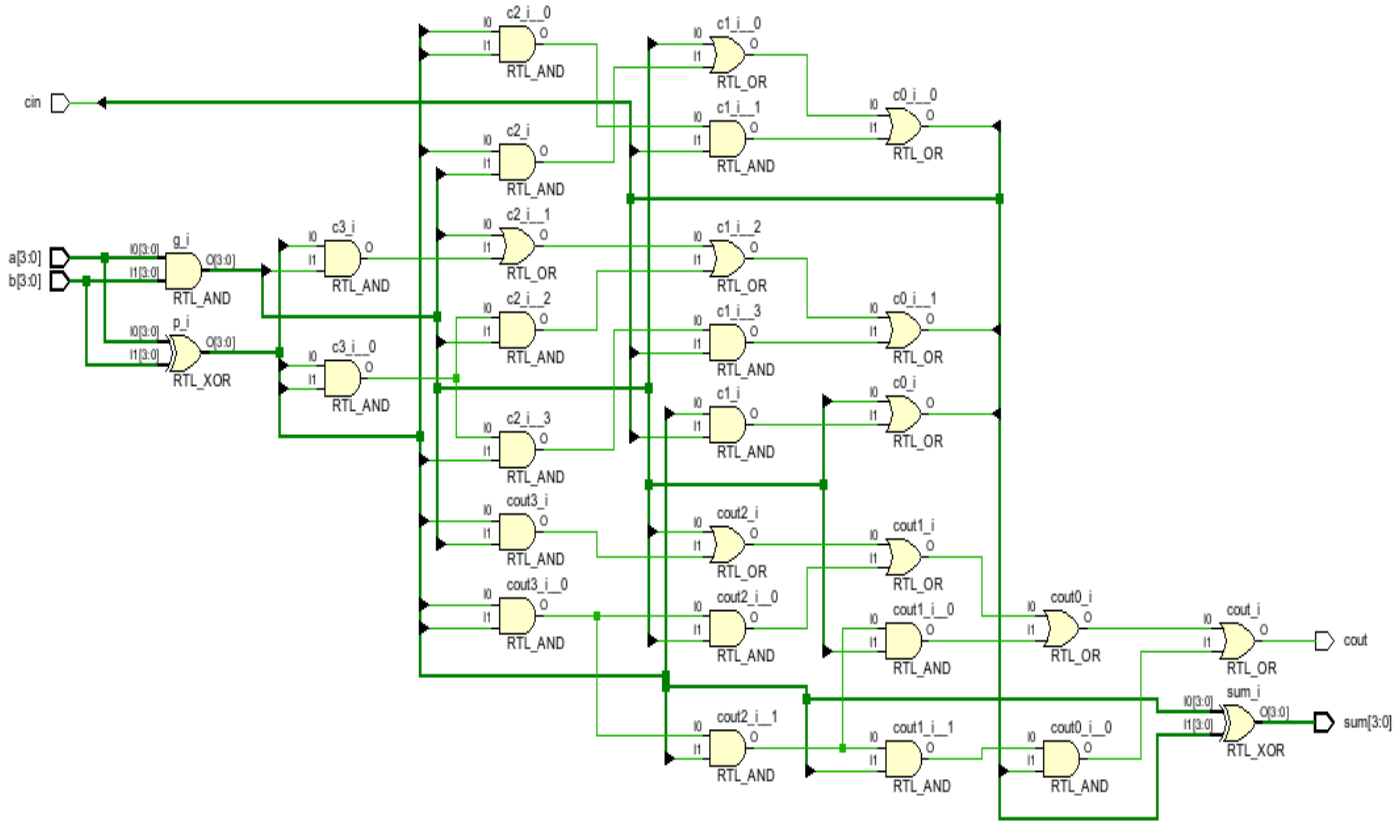
## 32-bit carry-look-ahead adder

The parallel adder is a ripple carry adder type in which the carry output of each full-adder stage is connected to the carry input of the next higher-order stage. Therefore, the sum and carry outputs of any stage cannot be produced until the input carry occurs;this leads to a time delay in the addition process.This delay is known as carry propagation delay.

One method of speeding up this process by eliminating inter stage carry delay is called look ahead-carry addition.

**Step-1 :** Implementation of 4 bit carry-look ahead adder

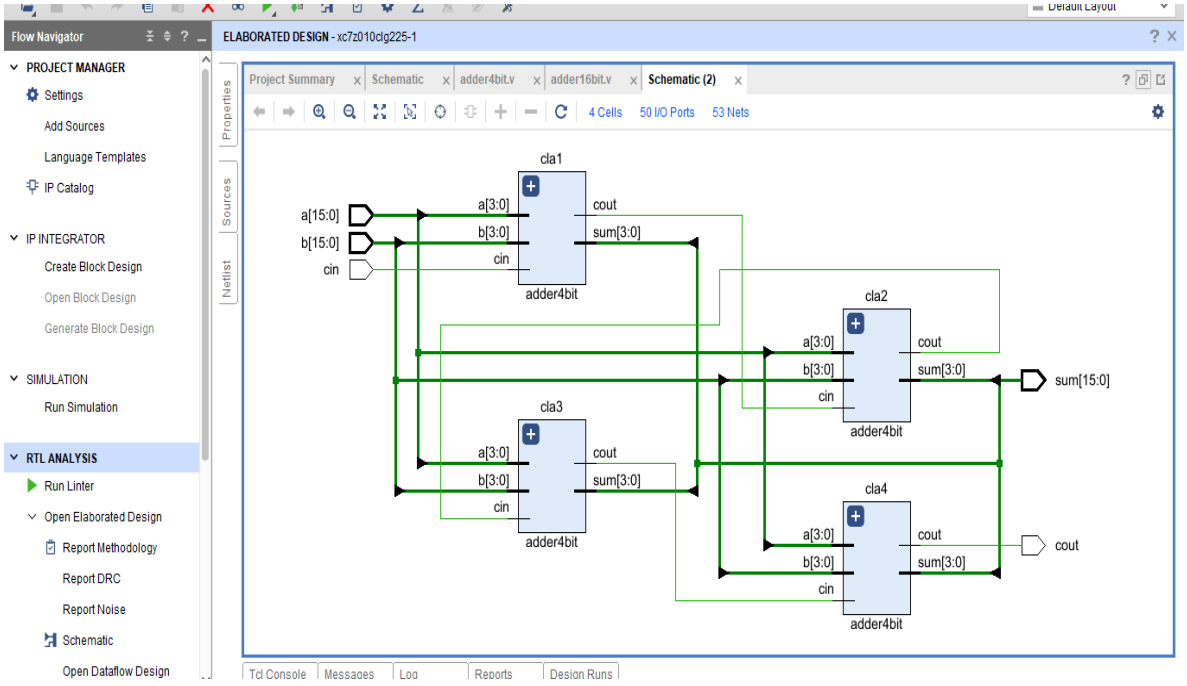
```
module adder4bit(a,b, cin, sum,cout);
input [3:0] a,b;
input cin;
output [3:0] sum;
output cout;
wire [3:0] p,g,c;
assign p=a^b;
assign g=a&b;
assign c[0]=cin;
assign c[1]= g[0]|(p[0]&c[0]);
assign c[2]= g[1] | (p[1]&g[0]) | p[1]&p[0]&c[0];
assign c[3]= g[2] | (p[2]&g[1]) | p[2]&p[1]&g[0] | p[2]&p[1]&p[0]&c[0];
assign cout= 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]&c[0];
assign sum=p^c;

endmodule
```



**Step-2 :** Implementation of 16 bit carry-look ahead adder

Instantiate of four 4bit carry-look ahead adder to design 16 bit carry-look ahead adder.

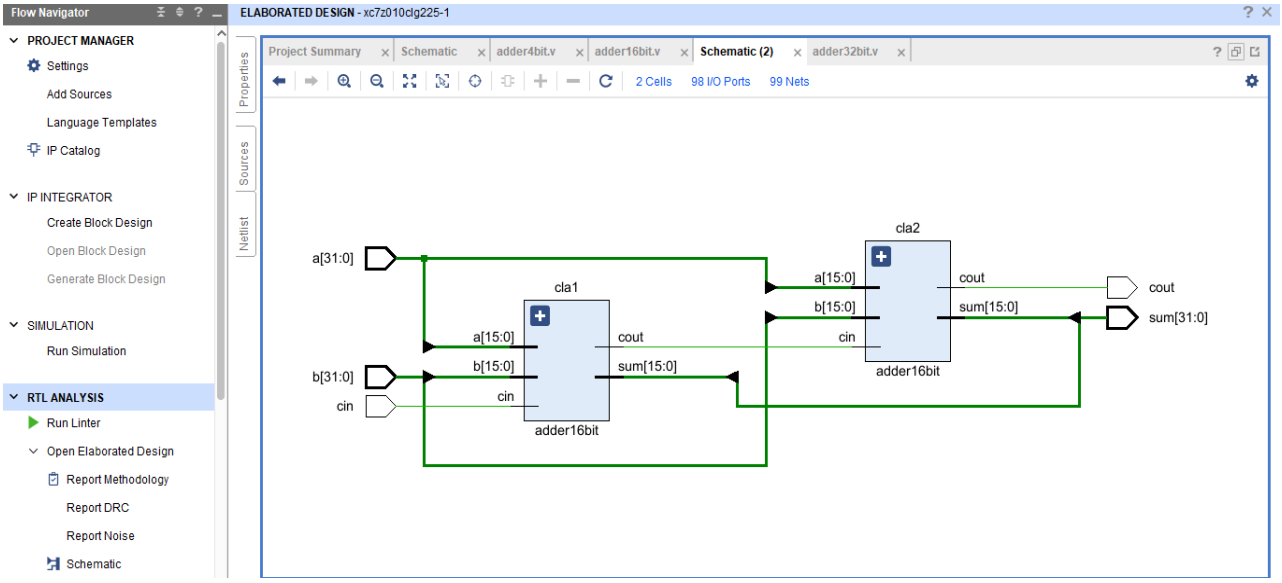
```
module adder16bit(a,b, cin, sum,cout);
        input[15:0]a,b;
        input cin;
        output [15:0] sum;
        output cout;
         wire c1,c2,c3;

adder4bit cla1 (.a(a[3:0]), .b(b[3:0]), .cin(cin), .sum(sum[3:0]), .cout(c1));

adder4bit cla2 (.a(a[7:4]), .b(b[7:4]), .cin(c1), .sum(sum[7:4]), .cout(c2));

adder4bit cla3(.a(a[11:8]), .b(b[11:8]), .cin(c2), .sum(sum[11:8]), .cout(c3));

adder4bit cla4(.a(a[15:12]), .b(b[15:12]), .cin(c3), .sum(sum[15:12]), .cout(cout));


endmodule
```




**Step-3 :** Implementation of 32 bit carry-look ahead adder

Instantiation of two 16bit carry-look ahead adder to design 32 bit carry-look ahead adder.

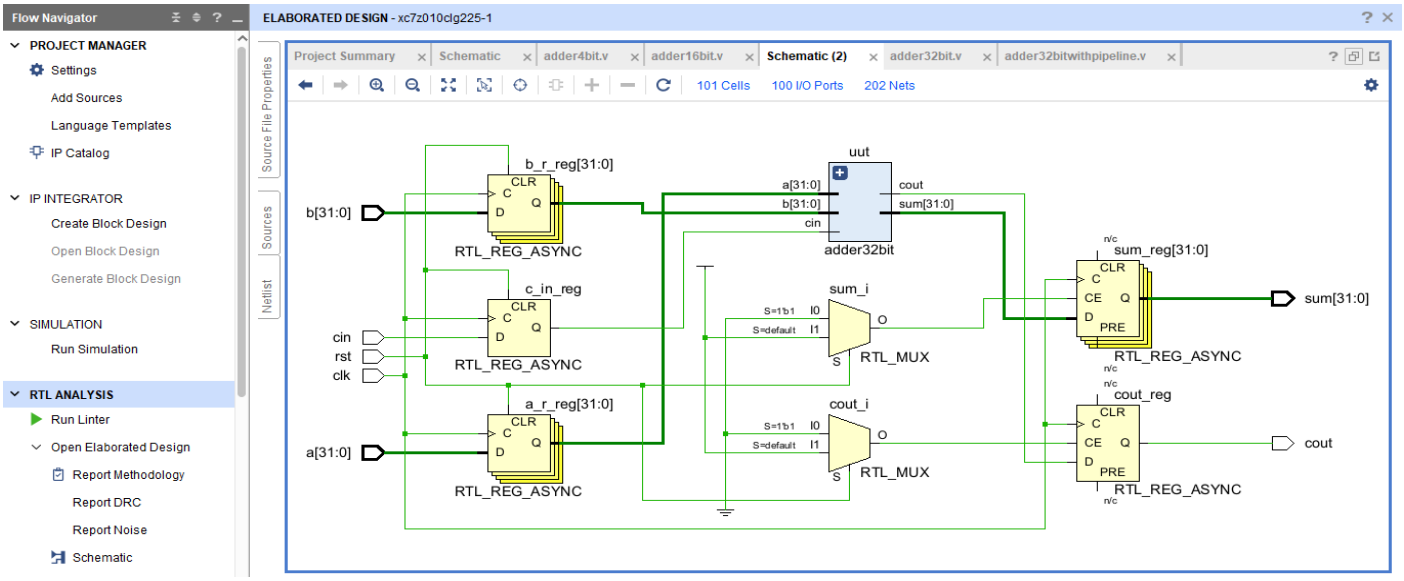
```
module adder32bit(a,b, cin, sum,cout);
input [31:0] a,b;
input cin;
output [31:0] sum;
output cout;
wire C1;
adder16bit cla1 (.a(a[15:0]), .b(b[15:0]), .cin(cin), .sum(sum[15:0]), .cout(C1));
Adder16bit cla2 (.a(a[31:16]), .b(b[31:16]), .cin(C1), .sum(sum[31:16]), .cout(cout));
endmodule
```




**Step-4 :** Implementation of 32 bit pipe-lined carry-look ahead adder

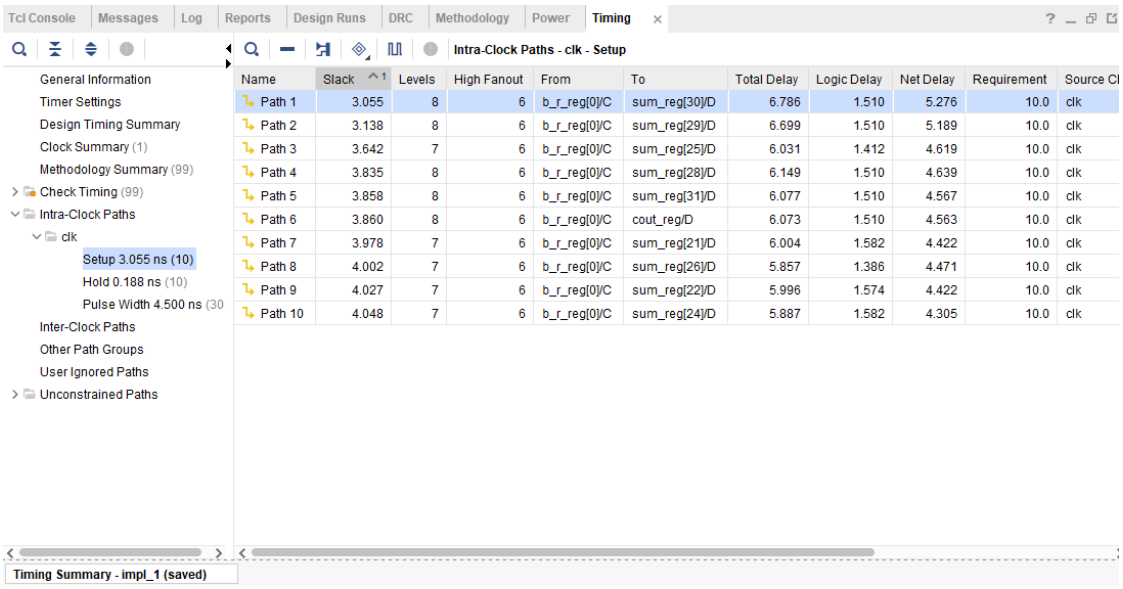
Pipe-lined carry-look ahead adder is nothing but the all inputs and outputs to the adder it's come from the register and get stored in registers.

```
module adder32bitwithpipeline(input clk,
input rst,
input  [31:0]a,
input  [31:0]b,
input cin,
output  reg[31:0]sum,
output  reg cout
    );
    reg [31:0] a_r,b_r;
    reg c_in;
    wire [31:0] s_r;
    wire c_r;
    Adder32bit uut(.a(a_r), .b(b_r), .cin(c_in), .sum(s_r), .cout(c_r));
    always @ (posedge clk or posedge rst) begin
    if(rst)
    begin
    a_r<=32'b00000000000000000000000000000000;
    b_r<=32'b00000000000000000000000000000000;
    c_in<=1'b0000000000000000000000000000000;
    end
    else
    begin
    a_r<=a;
    b_r<=b;
    c_in<=cin;
    sum<=s_r;
    cout<=c_r;
    end
    end
endmodule
```




## Timing Summary :

There are total 10 setup timing paths and 10 hold timing paths. For the designed shown above the clock is kept constant that is 10ns.




| Name | Slack | Levels | High Fanout | From | To | Total Delay | Logic Delay | Net Delay | Requirement | Source Cl |
|---|---|---|---|---|---|---|---|---|---|---|
| Path 1 | 3.055 | 8 | 6 | b_r_reg[0]/C | sum_reg[30]/D | 6.786 | 1.510 | 5.276 | 10.0 | clk |
| Path 2 | 3.138 | 8 | 6 | b_r_reg[0]/C | sum_reg[29]/D | 6.699 | 1.510 | 5.189 | 10.0 | clk |
| Path 3 | 3.642 | 7 | 6 | b_r_reg[0]/C | sum_reg[25]/D | 6.031 | 1.412 | 4.619 | 10.0 | clk |
| Path 4 | 3.835 | 8 | 6 | b_r_reg[0]/C | sum_reg[28]/D | 6.149 | 1.510 | 4.639 | 10.0 | clk |
| Path 5 | 3.858 | 8 | 6 | b_r_reg[0]/C | sum_reg[31]/D | 6.077 | 1.510 | 4.567 | 10.0 | clk |
| Path 6 | 3.860 | 8 | 6 | b_r_reg[0]/C | cout_reg/D | 6.073 | 1.510 | 4.563 | 10.0 | clk |
| Path 7 | 3.978 | 7 | 6 | b_r_reg[0]/C | sum_reg[21]/D | 6.004 | 1.582 | 4.422 | 10.0 | clk |
| Path 8 | 4.002 | 7 | 6 | b_r_reg[0]/C | sum_reg[26]/D | 5.857 | 1.386 | 4.471 | 10.0 | clk |
| Path 9 | 4.027 | 7 | 6 | b_r_reg[0]/C | sum_reg[22]/D | 5.996 | 1.574 | 4.422 | 10.0 | clk |
| Path 10 | 4.048 | 7 | 6 | b_r_reg[0]/C | sum_reg[24]/D | 5.887 | 1.582 | 4.305 | 10.0 | clk |

### 1. Path-1 :

Source(Launch FF): b_r_reg[0]/C
Destination (Capture FF): sum_reg[30]/D
Source clock path: clock clk rise edge + net + buffer + FDCE (Prop_fdce_C) = 5.228ns
Data path : FDCE (Prop_fdce_C_Q) + net + buffer + LUT + SU = 6.786ns.
Arrival time : 5.228 + 6.786 = **12.014ns.**
Required time : clock clk rise edge + net + buff + cp + cu = **15.069ns.**

**Slack** = Required time - Arrival time
= 15.069 – 12.01 = **3.055ns**

**Destination capture delay=**Required time - clock period
=15.069 - 10 = 5**.069ns**

2. **Path-2 :**

Source(Launch FF): b_r_reg[0]/C
Destination (Capture FF): sum_reg[29]/D
Source clock path: clock clk rise edge + net + buffer + FDCE (Prop_fdce_C) = 5.228ns
Data path : FDCE (Prop_fdce_C_Q) + net + buffer + LUT + SU = 6.699ns.
Arrival time : 5.228 + 6.786 = **11.926ns.**
Required time : clock clk rise edge + net + buff + cp + cu = **15.064ns.**

**Slack** = Required time - Arrival time
= 15.064 – 11.926
= **3.138ns**
**Destination capture delay=**Required time - clock period
=15.064 - 10 = 5**.064ns**

3. **Path-3 :**

Source(Launch FF): b_r_reg[0]/C
Destination (Capture FF): sum_reg[25]/D
Source clock path: clock clk rise edge + net + buffer + FDCE (Prop_fdce_C) = 5.228ns
Data path : FDCE (Prop_fdce_C_Q) + net + buffer + LUT + SU = 6.031ns.
Arrival time : 5.228 + 6.031 = **11.259ns.**
Required time : clock clk rise edge + net + buff + cp + cu = **14.901ns.**

**Slack** = Required time - Arrival time
= 14.901 – 11.259
= **3.642ns**
**Destination capture delay=**Required time - clock period
=14.901 - 10 = 4**.901ns**

4. **Path-4** :

Source(Launch FF): b_r_reg[0]/C
Destination (Capture FF): sum_reg[28]/D
Source clock path: clock clk rise edge + net + buffer + FDCE (Prop_fdce_C) = 5.228ns
Data path : FDCE (Prop_fdce_C_Q) + net + buffer + LUT + SU = 6.149ns.
Arrival time : 5.228 + 6.149 = **11.377ns.**
Required time : clock clk rise edge + net + buff + cp + cu = **15.212ns.**

**Slack** = Required time - Arrival time
= 15.212 – 11.377
= **3.835ns**
**Destination capture delay=**Required time - clock period
=15.212 - 10 = 5**.212ns**

5. **Path-5** :

Source(Launch FF): b_r_reg[0]/C
Destination (Capture FF): sum_reg[31]/D
Source clock path: clock clk rise edge + net + buffer + FDCE (Prop_fdce_C) = 5.228ns
Data path : FDCE (Prop_fdce_C_Q) + net + buffer + LUT + SU = 6.077ns.
Arrival time : 5.228 + 6.077 = **11.304ns.**
Required time : clock clk rise edge + net + buff + cp + cu = **15.162ns.**

**Slack** = Required time - Arrival time
= 15.162 – 11.304
= **3.858ns**

**Destination capture delay=**Required time - clock period
=15.162 - 10 = 5.**162ns**

6. **Path-6** :

Source(Launch FF): b_r_reg[0]/C
Destination (Capture FF): cout_reg/D
Source clock path: clock clk rise edge + net + buffer + FDCE (Prop_fdce_C) = 5.228ns
Data path : FDCE (Prop_fdce_C_Q) + net + buffer + LUT + SU = 6.073ns.
Arrival time : 5.228 + 6.073 = **11.300ns.**
Required time : clock clk rise edge + net + buff + cp + cu = **15.160ns.**

**Slack** = Required time - Arrival time
= 15.160 – 11.300
= **3.860ns**

**Destination capture delay=**Required time - clock period
=15.160 - 10 = 5.**160ns**

7. **Path-7** :

Source(Launch FF): b_r_reg[0]/C
Destination (Capture FF): sum_reg[21]/D
Source clock path: clock clk rise edge + net + buffer + FDCE (Prop_fdce_C) = 5.228ns
Data path : FDCE (Prop_fdce_C_Q) + net + buffer + LUT + SU = 6.004ns.
Arrival time : 5.228 + 6.004 = **11.232ns.**
Required time : clock clk rise edge + net + buff + cp + cu = **15.210ns.**

**Slack** = Required time - Arrival time
= 15.210 – 11.232
= **3.978ns**

**Destination capture delay=**Required time - clock period
=15.210 - 10 = 5.**210ns**

8. **Path-8** :

Source(Launch FF): b_r_reg[0]/C
Destination (Capture FF): sum_reg[26]/D
Source clock path: clock clk rise edge + net + buffer + FDCE (Prop_fdce_C) = 5.228ns
Data path : FDCE (Prop_fdce_C_Q) + net + buffer + LUT + SU = 5.857ns.
Arrival time : 5.228 + 5.857 = **11.084ns.**
Required time : clock clk rise edge + net + buff + cp + cu = **15.086ns.**

**Slack** = Required time - Arrival time
= 15.086 – 11.084
= **4.002ns**

**Destination capture delay=**Required time - clock period
=15.086 - 10 = 5.**086ns**

9. **Path-9** :

Source(Launch FF): b_r_reg[0]/C
Destination (Capture FF): sum_reg[22]/D
Source clock path: clock clk rise edge + net + buffer + FDCE (Prop_fdce_C) = 5.228ns
Data path : FDCE (Prop_fdce_C_Q) + net + buffer + LUT + SU = 5.996ns.
Arrival time : 5.228 + 5.996 = **11.224ns.**

Required time : clock clk rise edge + net + buff + cp + cu = **15.251ns.**

**Slack** = Required time - Arrival time

= 15.251 – 11.224

= **4.027ns**

**Destination capture delay**=Required time - clock period

=15.251 - 10 = 5**.251ns**

10. **Path-10** :

Source(Launch FF): b_r_reg[0]/C

Destination (Capture FF): sum_reg[24]/D

Source clock path: clock clk rise edge + net + buffer + FDCE (Prop_fdce_C) = 5.228ns

Data path : FDCE (Prop_fdce_C_Q) + net + buffer + LUT + SU = 5.887ns.

Arrival time : 5.228 + 5.887 = **11.115ns.**

Required time : clock clk rise edge + net + buff + cp + cu = **15.162ns.**

**Slack** = Required time - Arrival time

= 15.162 – 11.115

= **4.048ns**

**Destination capture delay**=Required time - clock period

=15.162- 10 = 5**.162ns**

**Waveform:**