컴퓨터공학 기초 실험2

Lab #4

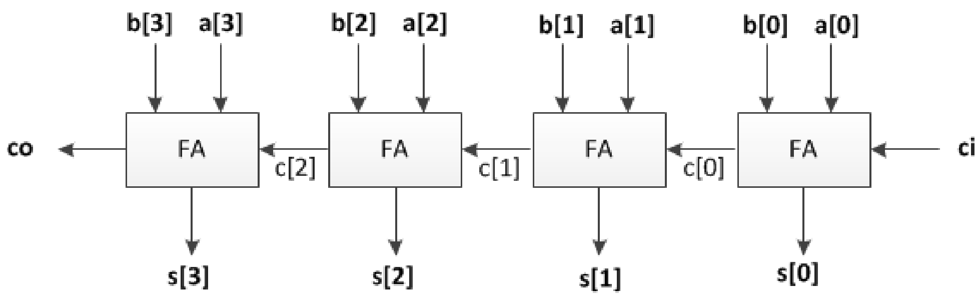
Carry Look-ahead Adder(CLA)

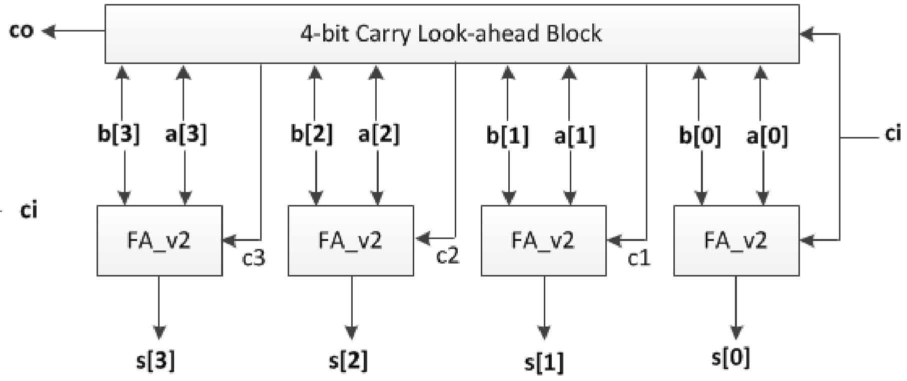
Kwangwoon University
Basic Computer Engineering Lab2

CARRY LOOK-AHEAD ADDER

Carry Look-ahead Adder

- Ripple Carry Adder(RCA)가 계산이 완료될 때까지의 시간이 많이 걸리는 단점을 보완하기 위해 입력 a, b 그리고 carry가 주어질 때, 모든 올림수가 동시에 구해져 계산시간을 단축시킨 가산기
  - ✓ Carry만을 계산해주는 별도의 carry look-ahead block이 존재





RCA **CLA**

Carry Look-ahead Adder(Cont.)

- CLA로 carry out 값을 미리 계산해주기 위해서는 generation signal(G_i), propagation signal(P_i)을 아래와 같이 정의한다.

$$G_i = A_iB_i$$
$$P_i = A_i + B_i$$

- 위 식을 full adder의 carry out에 적용하면 다음과 같다.

$$C_{i+1} = A_iB_i + (A_i + B_i)C_i = G_i + P_iC_i$$

- 이를 적용하여 4-bit adder의 carry를 미리 계산하면 다음과 같다.

$C_1 =$
$C_2 =$
$C_3 =$
$Cout =$

Boolean Equation

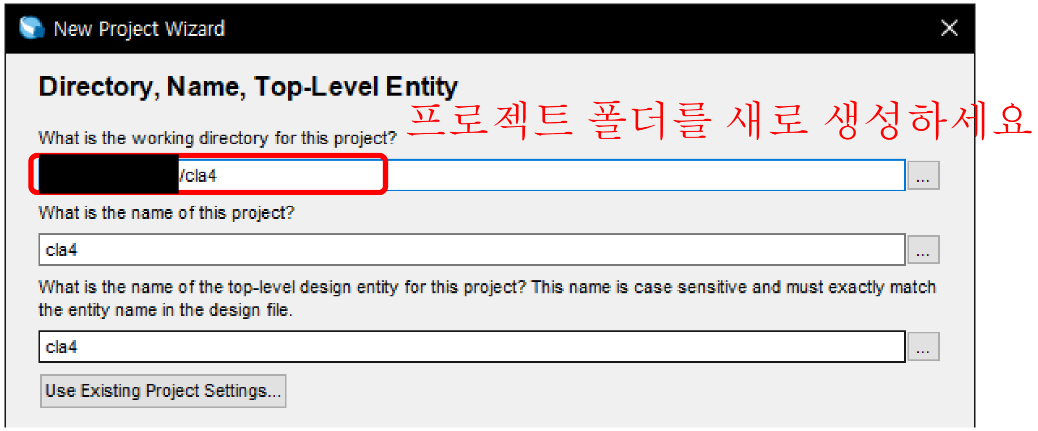
4-bit Carry Look-ahead Adder

PRACTICE I

Carry Look-ahead Adder

- New Project Wizard
  - ✓ Project name : cla4
  - ✓ Family & Device : Cyclone V 5CSXFC6D6F31C6 (밑에서 6번째)

- Verilog file
  - ✓ Add file : gates.v
  - ✓ New file : fa_v2.v, clb4.v, cla4.v, tb_cla4.v



Directory, Name, Top-Level Entity

프로젝트 폴더를 새로 생성하세요

What is the working directory for this project?

[redacted]/cla4

What is the name of this project?

cla4

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

cla4

Basic Logic Gates (1/2)

- Inverter

```
module _inv(a,y);
input a;
output y;
assign y=~a;
endmodule
```

- 2-to-1 nand gate

```
module _nand2(a,b,y);
input a,b;
output y;
assign y=~(a&b);
endmodule
```

- 2-to-1 and gate

```
module _and2(a,b,y);
input a,b;
output y;
assign y=a&b;
endmodule
```

- 2-to-1 or gate

```
module _or2(a,b,y);
input a,b;
output y;
assign y=a|b;
endmodule
```

- 2-to-1 xor gate

```
module _xor2(a,b,y);
input a, b;
output y;
wire inv_a, inv_b;
wire w0, w1;
_inv U0_inv(.a(a), .y(inv_a));
_inv U1_inv(.a(b), .y(inv_b));
_and2 U2_and2(.a(inv_a), .b(b), .y(w0));
_and2 U3_and2(.a(a),.b(inv_b), .y(w1));
_or2 U4_or2(.a(w0), .b(w1),.y(y));
endmodule
```

Basic Logic Gates (2/2)

- 3-to-1 and gate

```
module _and3(a,b,c,y);
input a,b,c;
output y;
assign y=a&b&c;
endmodule
```

- 4-to-1 and gate

```
module _and4(a,b,c,d,y);
input a,b,c,d;
output y;
assign y=a&b&c&d;
endmodule
```

- 5-to-1 and gate

```
module _and5(a,b,c,d,e,y);
input a,b,c,d,e;
output y;
assign y=a&b&c&d&e;
endmodule
```

- 3-to-1 or gate

```
module _or3(a,b,c,y);
input a,b,c;
output y;
assign y=a|b|c;
endmodule
```

- 4-to-1 or gate

```
module _or4(a,b,c,d,y);
input a,b,c,d;
output y;
assign y=a|b|c|d;
endmodule
```

- 5-to-1 or gate

```
module _or5(a,b,c,d,e,y);
input a,b,c,d,e;
output y;
assign y=a|b|c|d|e;
endmodule
```

(Lab 3에 작성한 gates.v에 추가)

Revision of Full Adder

- Full Adder
  - RCA에서 작성했던 full adder와 다르게 CLA에서는 full adder의 carry out이 필요하지 않다.
  - module name : fa_v2
  - Exclusive or gate 2개를 instance하여 작성한다.

Input			Output
Ci	a	b	s
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

```
module fa_v2(a,b,ci,s);
input a,b,ci;
output s;
wire w0;

Instances

endmodule
```

Carry Look-ahead Block (1/2)

- Module name : clb4

$G_i =$

$P_i =$

$C_1 =$

$C_2 =$

$C_3 =$

$Co =$

Boolean Equation

```
module clb4(a,b,ci,c1,c2,c3,co);
 input [3:0]a,b;
 input ci;
 output c1,c2,c3,co;
 wire [3:0] g,p;
 wire w0_c1;
 wire w0_c2, w1_c2;
 wire w0_c3, w1_c3, w2_c3;
 wire w0_co, w1_co, w2_co, w3_co;

  // Generate
```

Instances about generate signal (Gi)

```
   // Propagate
```

Instances about propagate signal (Pi)

Carry Look-ahead Block (2/2)

```
  // c1 = g[0] | (p[0] & ci);
```

Instances about c1

```
  // c2 = g[1] | (p[1] & g[0]) | (p[1] & p[0] & ci);
```

Instances about c2

```
  // c3 = g[2] | (p[2] & g[1]) | (p[2] & p[1] & g[0]) | (p[2] & p[1] & p[0] & ci);
```

Instances about c3

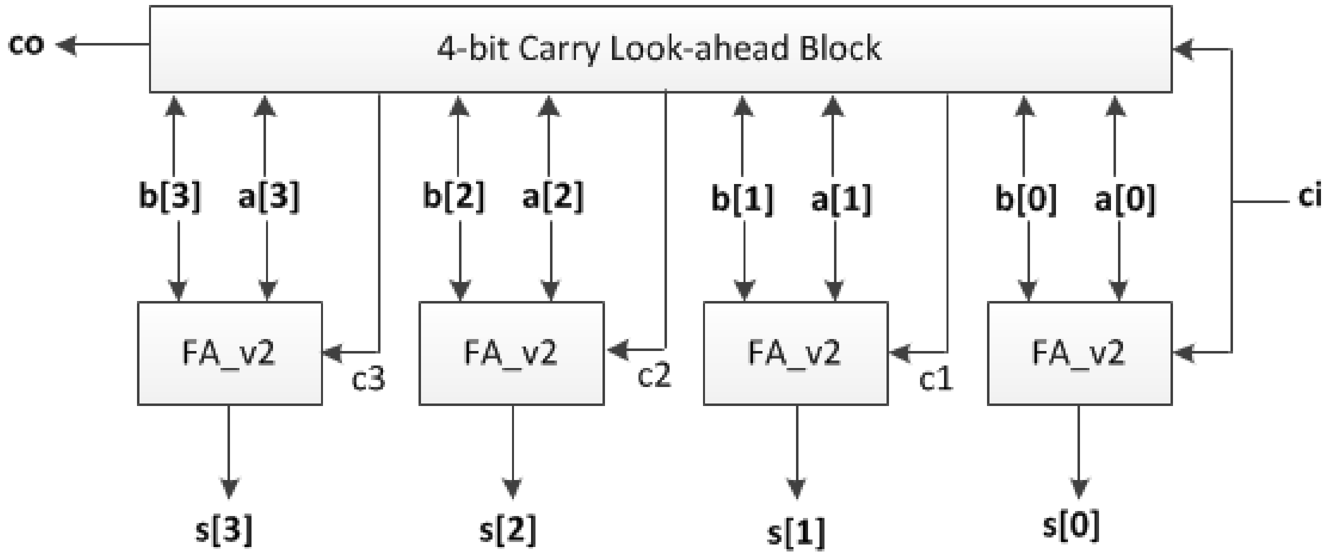
```
  // co = g[3]
  // | (p[3] & g[2])
  // | (p[3] & p[2] & g[1])
  // | (p[3] & p[2] & p[1] & g[0])
  // | (p[3] & p[2] & p[1] & p[0] & ci)
```

Instances about co

```
endmodule
```

Carry Look-ahead Adder

- cla4



```
module cla4(a, b, ci, s, co);
  input [3:0] a, b;
  input ci;
  output [3:0] s;
  output co;

  wire [3:0] c;

  fa_v2 U0_fa
  fa_v2 U1_fa
  fa_v2 U2_fa
  fa_v2 U3_fa
  clb4  U4_clb4
endmodule
```

Instances of fa_v2 and clb4

Testbench

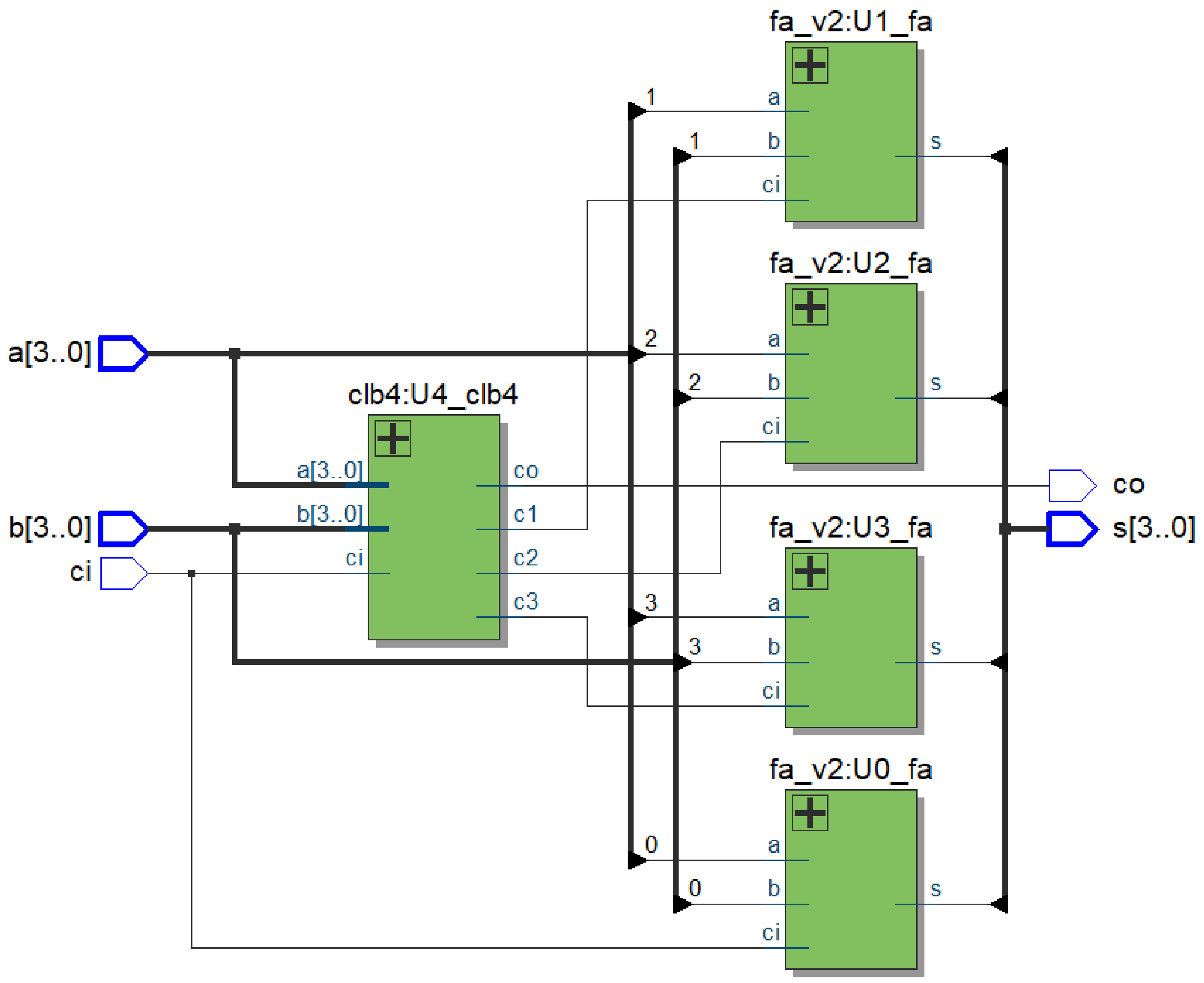
- Testbench
  - ✓ Functional simulation을 수행

```
`timescale 1ns/100ps

module tb_cla4;
  reg [3:0] tb_a, tb_b;
  reg tb_ci;
  wire [3:0] tb_s;
  wire tb_co;
  wire [4:0] tb_result;

  assign tb_result = {tb_co, tb_s};

  cla4 U0_cla4(.a(tb_a), .b(tb_b), .ci(tb_ci), .s(tb_s), .co(tb_co));

  initial
  begin
```

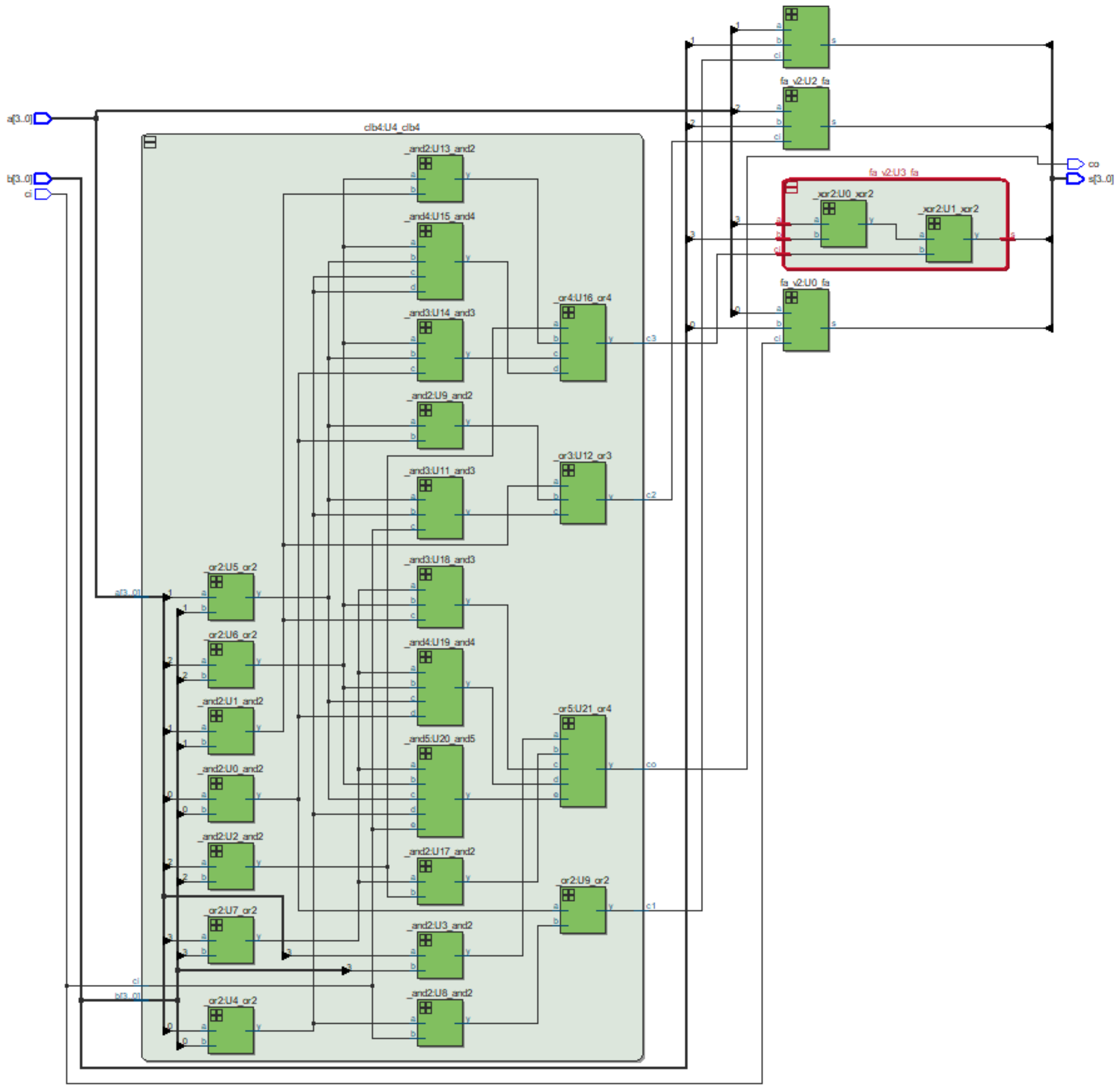
Testbench

```
  end
endmodule
```

RTL Viewer (1/2)

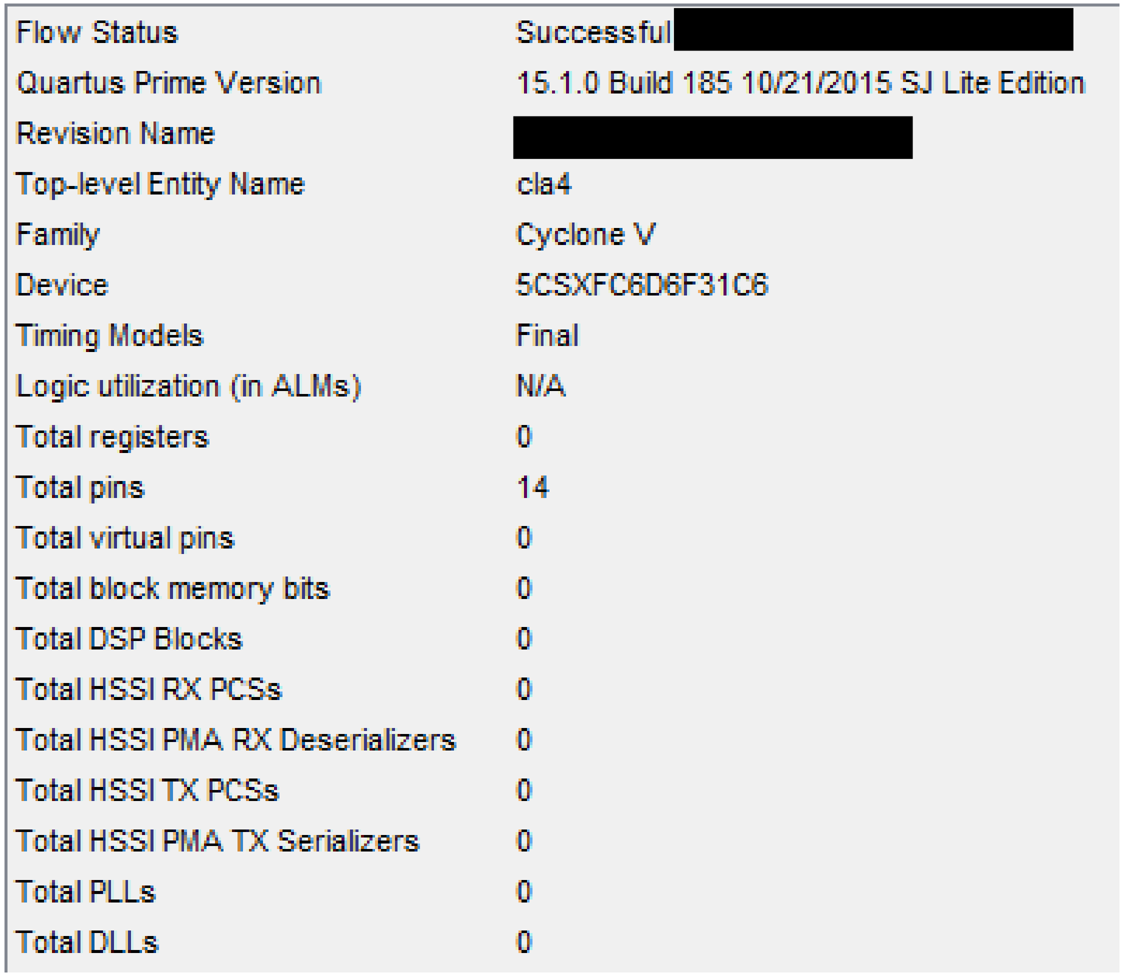


RTL Viewer (2/2)



Compilation Report

- RCA와 비교하여 본다.



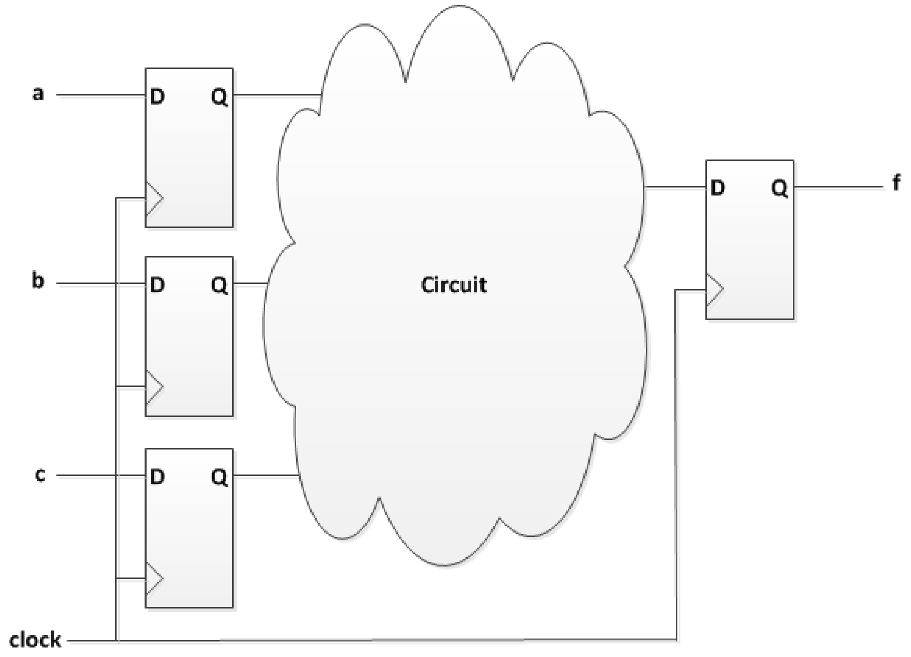
Flow Status	Successful [redacted]
Quartus Prime Version	15.1.0 Build 185 10/21/2015 SJ Lite Edition
Revision Name	[redacted]
Top-level Entity Name	cla4
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	0
Total pins	14
Total virtual pins	0
Total block memory bits	0
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0
Total DLLs	0

Timing Analysis of 32-bit CLA

PRACTICE II

Timing Analysis

- Timing analysis
  - ✓ **A process of analyzing delays** in a logic circuit to determine the conditions under which the circuit operates reliably
  - ✓ These conditions include, but are not limited to, the maximum clock frequency(f_{max}) for which the circuit will produce a correct output



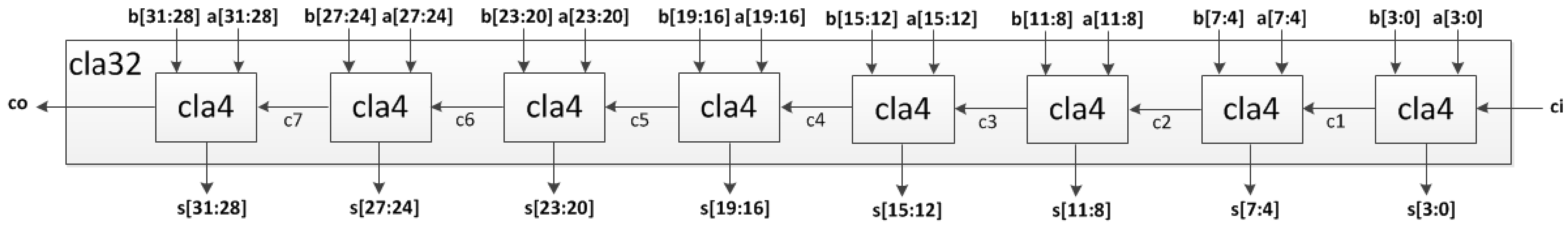
Timing Analysis of CLA

- New Project Wizard
  - ✓ Project name : cla_clk
  - ✓ Family & Device : Cyclone V 5CSXFC6D6F31C6 (밑에서 6번째)

- Verilog file
  - ✓ Add file: gates.v, fa_v2.v, clb4.v, cla4.v
  - ✓ New file : cla32.v, cla_clk.v, tb_cla_clk.v

32-bit CLA

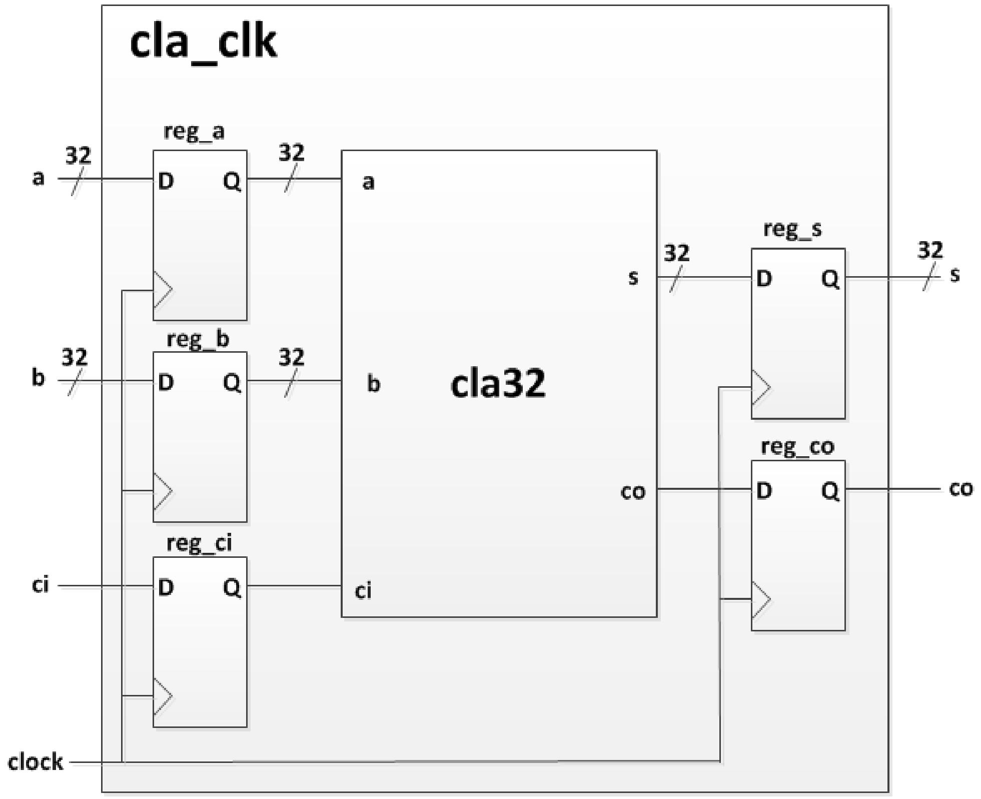
- 32-bit CLA
  - File name : cla32.v – Module name : cla32
  - 앞서 구현한 4-bit cla 8개를 instance하여 직렬로 연결하여 구현



```
module cla32(a, b, ci, s, co);
  input [31:0] a, b;
  input ci;
  output [31:0] s;
  output co;

  wire c1, c2, c3, c4, c5, c6, c7;

                    Instance of cla4

endmodule
```

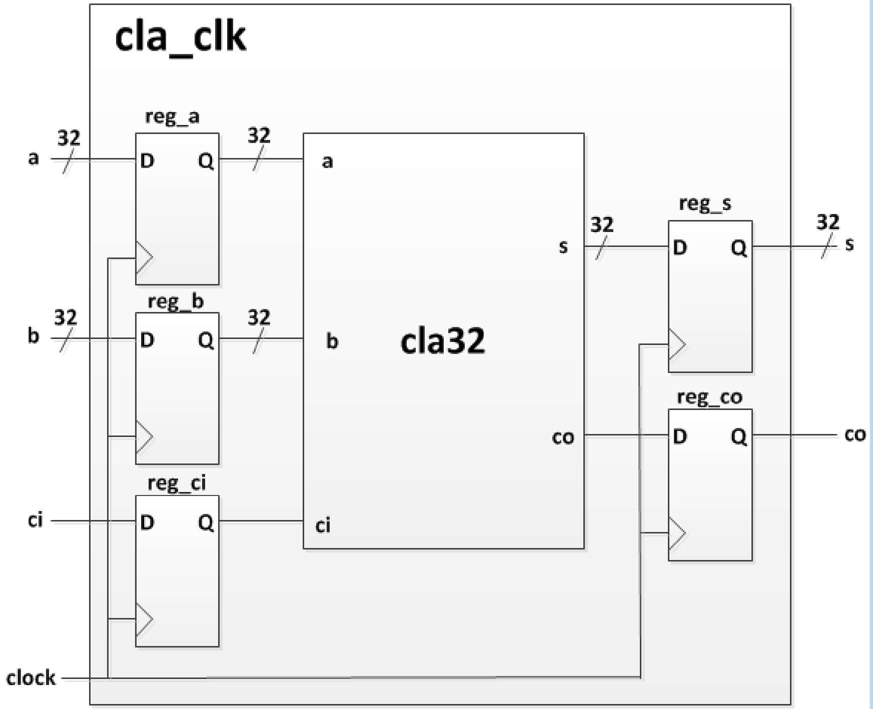
CLA with Register (1/2)

- cla_clk module
  - 32-bit cla에 앞 뒤로 flip-flop을 추가하여 clock과 연결하는 module
  - 해당 모듈에서 clock을 인가하여 valid한 결과가 나오는 데 필요한 clock period를 확인하는 게 목적



CLA with Register (2/2)

- cla_clk module



```
module cla_clk(clk, a, b, ci, s, co);
  input                    clk;
  input  [31:0]            a, b;
  input                    ci;
  output         [31:0]           s;
  output                          co;

  reg    [31:0]            reg_a, reg_b;
  reg                      reg_ci;
  reg    [31:0]            reg_s;
  reg                      reg_co;

  wire   [31:0]            wire_s;
  wire                     wire_co;

  always @ (posedge clk)
  begin
```

always

```
  end
```

Instance of cla32

assign

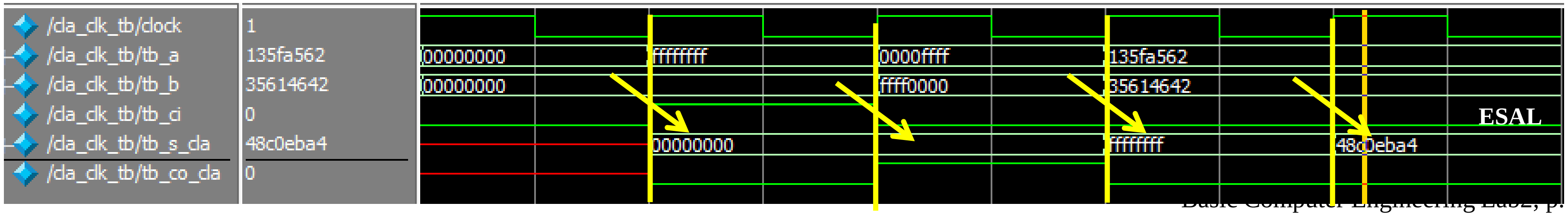
```
Endmodule
```

Testbench

```
`timescale 1ns/100ps

module tb_cla_clk;
  reg           clk;
  reg   [31:0]  tb_a, tb_b;
  reg           tb_ci;
  wire  [31:0]  tb_s_cla;
  wire          tb_co_cla;

  parameter STEP = 10;

  cla_clk U0_cla_clk(.clk(clk), .a(tb_a), .b(tb_b), .ci(tb_ci),
         .s_cla(tb_s_cla), .co_cla(tb_co_cla));

  always #(STEP/2) clk = ~clk;

  initial
  begin
```

Testbench

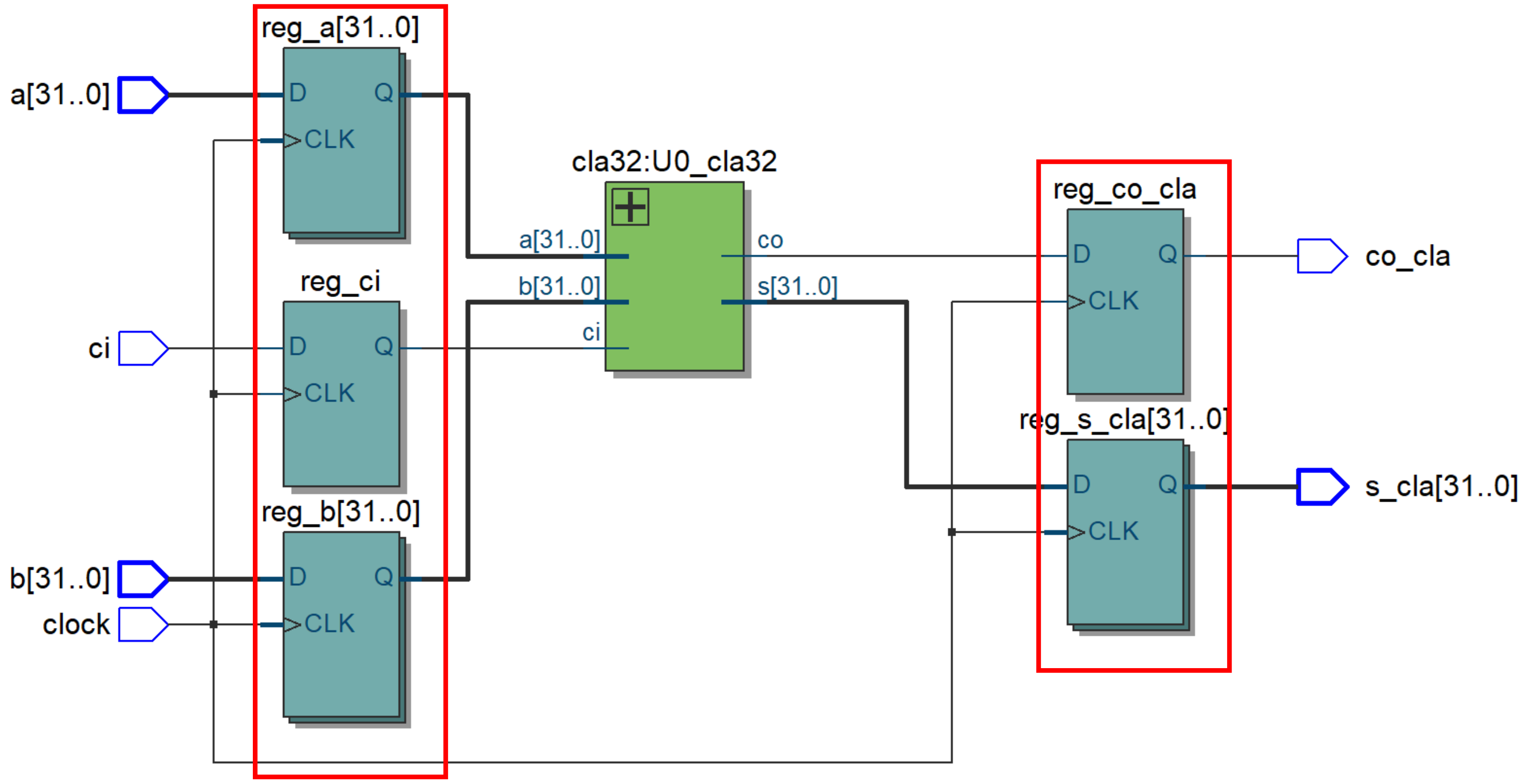
```
  end
endmodule
```



```
/cla_clk_tb/clock      1
/cla_clk_tb/tb_a       135fa562   00000000   ffffffff   0000ffff   135fa562
/cla_clk_tb/tb_b       35614642   00000000              ffff0000   35614642
/cla_clk_tb/tb_ci      0
/cla_clk_tb/tb_s_cla   48c0eba4              00000000              ffffffff   48c0eba4
/cla_clk_tb/tb_co_cla  0
```

ESAL

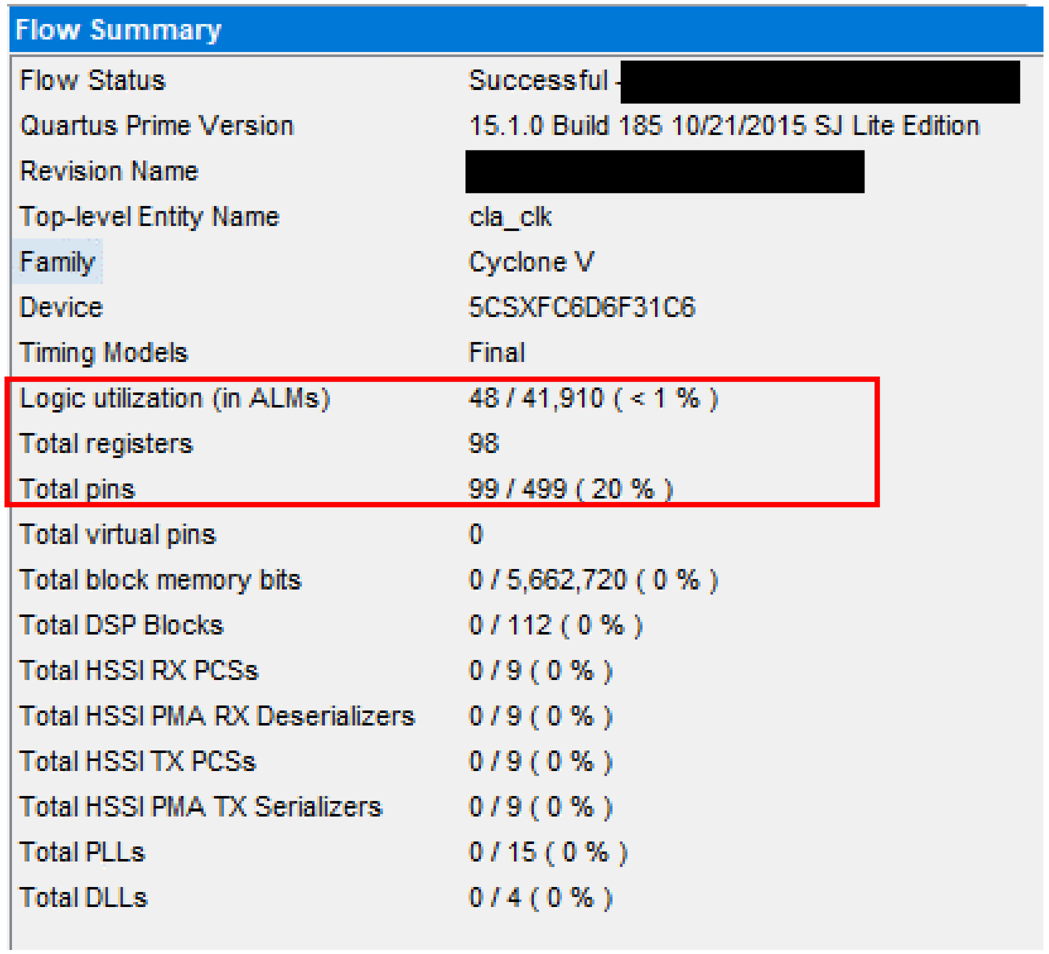
RTL Viewer



a[31..0]
ci
b[31..0]
clock
reg_a[31..0]
reg_ci
reg_b[31..0]
D
Q
CLK
cla32:U0_cla32
a[31..0]
b[31..0]
ci
co
s[31..0]
reg_co_cla
reg_s_cla[31..0]
co_cla
s_cla[31..0]

Compilation Report

- Flow Summary

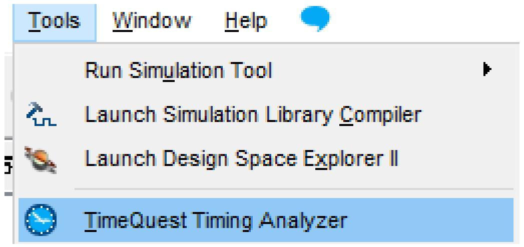


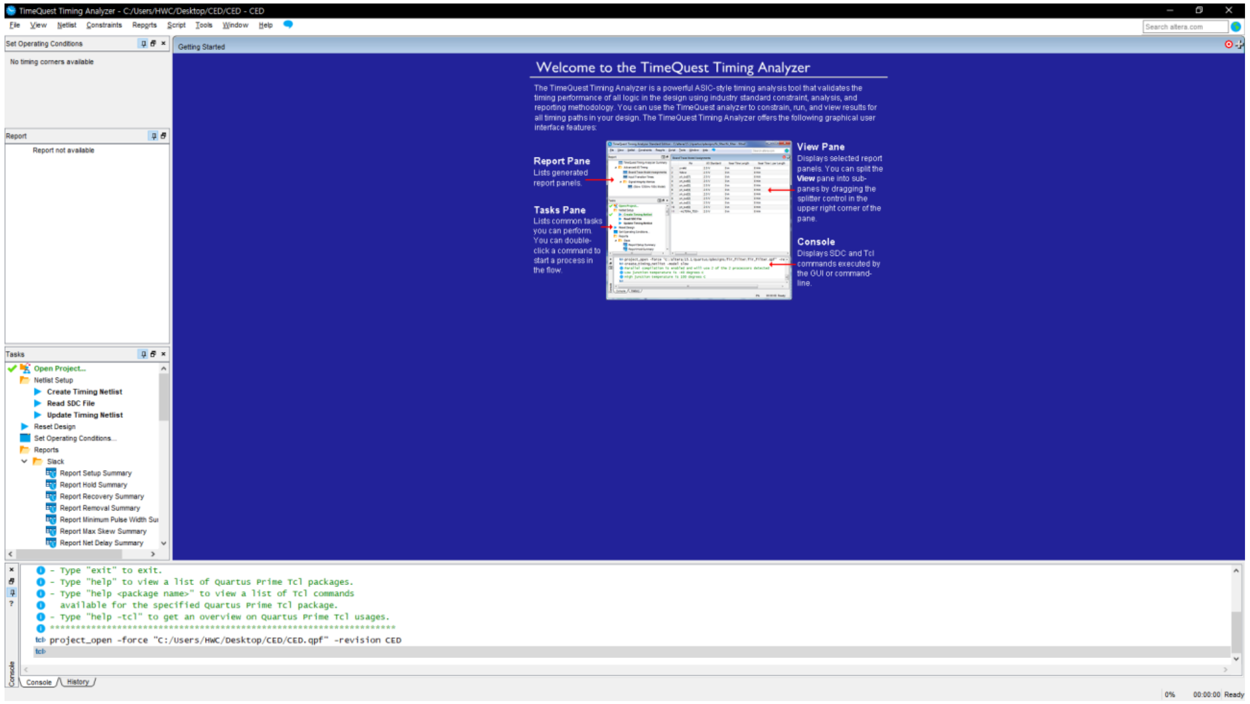
Flow Summary

Flow Status	Successful - [redacted]
Quartus Prime Version	15.1.0 Build 185 10/21/2015 SJ Lite Edition
Revision Name	[redacted]
Top-level Entity Name	cla_clk
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	48 / 41,910 (< 1 %)
Total registers	98
Total pins	99 / 499 (20 %)
Total virtual pins	0
Total block memory bits	0 / 5,662,720 (0 %)
Total DSP Blocks	0 / 112 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 15 (0 %)
Total DLLs	0 / 4 (0 %)

TimeQuest Timing Analyzer (1/2)

- Tools ➔ TimeQuest Timing Analyzer


Tools
Window
Help
Run Simulation Tool
Launch Simulation Library Compiler
Launch Design Space Explorer II
TimeQuest Timing Analyzer



Welcome to the TimeQuest Timing Analyzer

The TimeQuest Timing Analyzer is a powerful ASIC-style timing analysis tool that validates the timing performance of all logic in the design using industry standard constraint, analysis, and reporting methodology. You can use the TimeQuest analyzer to constrain, run, and view results for all timing paths in your design. The TimeQuest Timing Analyzer offers the following graphical user interface features:

Report Pane Lists generated report panels.

Tasks Pane Lists common tasks you can perform. You can double-click a command to start a process in the flow.

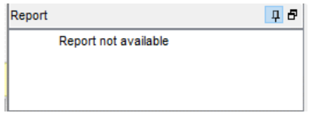
View Pane Displays selected report panels. You can split the **View** pane into sub-panes by dragging the splitter control in the upper right corner of the pane.

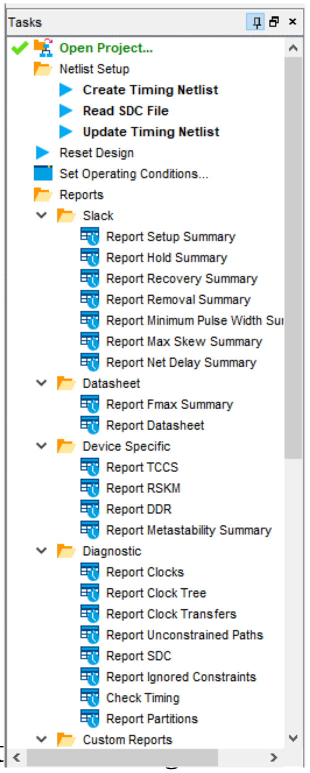
Console Displays SDC and Tcl commands executed by the GUI or command-line.

```
- Type "exit" to exit.
- Type "help" to view a list of Quartus Prime Tcl packages.
- Type "help <package name>" to view a list of Tcl commands
  available for the specified Quartus Prime Tcl package.
- Type "help -tcl" to get an overview on Quartus Prime Tcl usages.
******************************************************************
tcl> project_open -force "C:/Users/HWC/Desktop/CED/CED.qpf" -revision CED
tcl>
```

TimeQuest Timing Analyzer (2/2)

- TimeQuest
  - ✓ Report pane & Tasks Pane
  - ✓ Report pane
    - Contains any reports generated when using the tool
  - ✓ Tasks pane
    - Contains a sequence of actions that can be performed to obtain timing reports

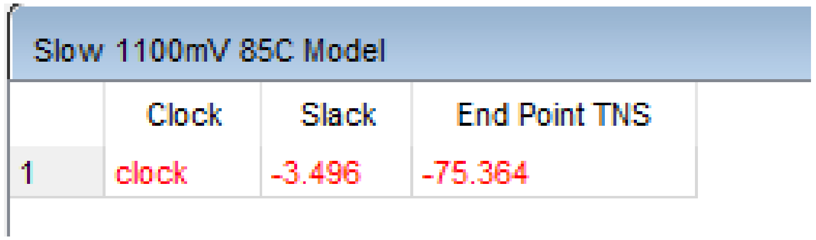

Report
Report not available


Tasks
- Open Project...
- Netlist Setup
  - Create Timing Netlist
  - Read SDC File
  - Update Timing Netlist
- Reset Design
- Set Operating Conditions...
- Reports
  - Slack
    - Report Setup Summary
    - Report Hold Summary
    - Report Recovery Summary
    - Report Removal Summary
    - Report Minimum Pulse Width Su
    - Report Max Skew Summary
    - Report Net Delay Summary
  - Datasheet
    - Report Fmax Summary
    - Report Datasheet
  - Device Specific
    - Report TCCS
    - Report RSKM
    - Report DDR
    - Report Metastability Summary
  - Diagnostic
    - Report Clocks
    - Report Clock Tree
    - Report Clock Transfers
    - Report Unconstrained Paths
    - Report SDC
    - Report Ignored Constraints
    - Check Timing
    - Report Partitions
  - Custom Reports

  - ✓ Double click
    1. **Create Timing Netlist**
       - To create a timing netlist, which will be used to perform the analysis.
    2. **Read SDC File**
       - To instruct the analyzer to read a Synopsys Design Contraints(SDC) file and apply the const analysis.
       - Specifying the constraints enables the analyzer to determine which parts of the design will operate correctly and which will not.
       - Initially, no constraints are specified and the default constraint of 1 GHz on the clock signal is applied automatically.
    3. **Update Timing Netlist**
       - To use the specified constraints to determine which parts of the circuit fail to meet them.

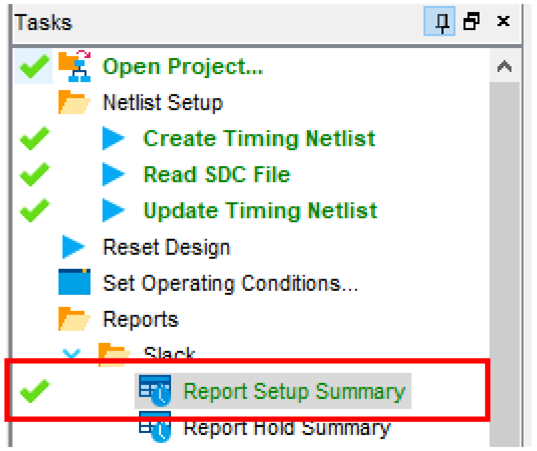
Timing Analysis of CLA (1/11)

- Double click
  - ✓ Report Setup Summary



Slow 1100mV 85C Model

	Clock	Slack	End Point TNS
1	clock	-3.496	-75.364



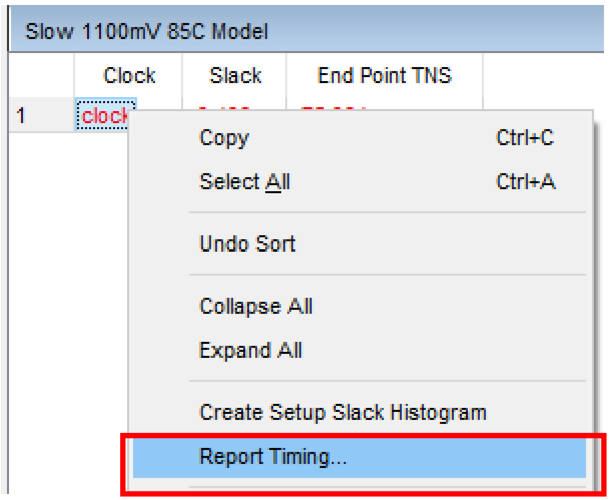
Tasks

- ✓ Open Project...
- Netlist Setup
  - ✓ Create Timing Netlist
  - ✓ Read SDC File
  - ✓ Update Timing Netlist
- Reset Design
- Set Operating Conditions...
- Reports
  - Slack
    - ✓ Report Setup Summary
    - Report Hold Summary

  - ✓ Slack and TNS(Total Negative Slack)
    - Indicate how well the design meets setup constraints for each clock domain
      - Slack shows the difference between the time a signal is required to arrive at a destination FF, as per the desired clock period, and the actual arrival time.
      - **When the slack is negative, the path takes too long to compute, and the timing is considered to violate the clock constraint.**

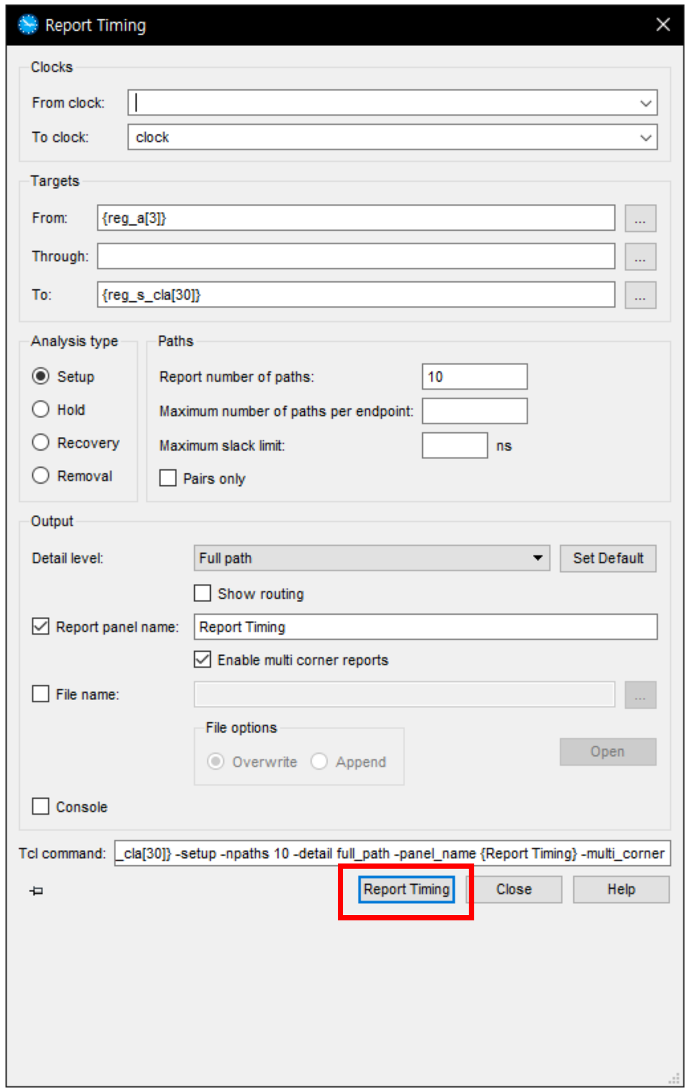
Timing Analysis of CLA (2/11)

➢ Report Timing



Slow 1100mV 85C Model

	Clock	Slack	End Point TNS
1	clock		



Report Timing

Clocks
From clock:
To clock: clock

Targets
From: {reg_a[3]}
Through:
To: {reg_s_cla[30]}

Analysis type: [✓ Setup] Hold Recovery Removal

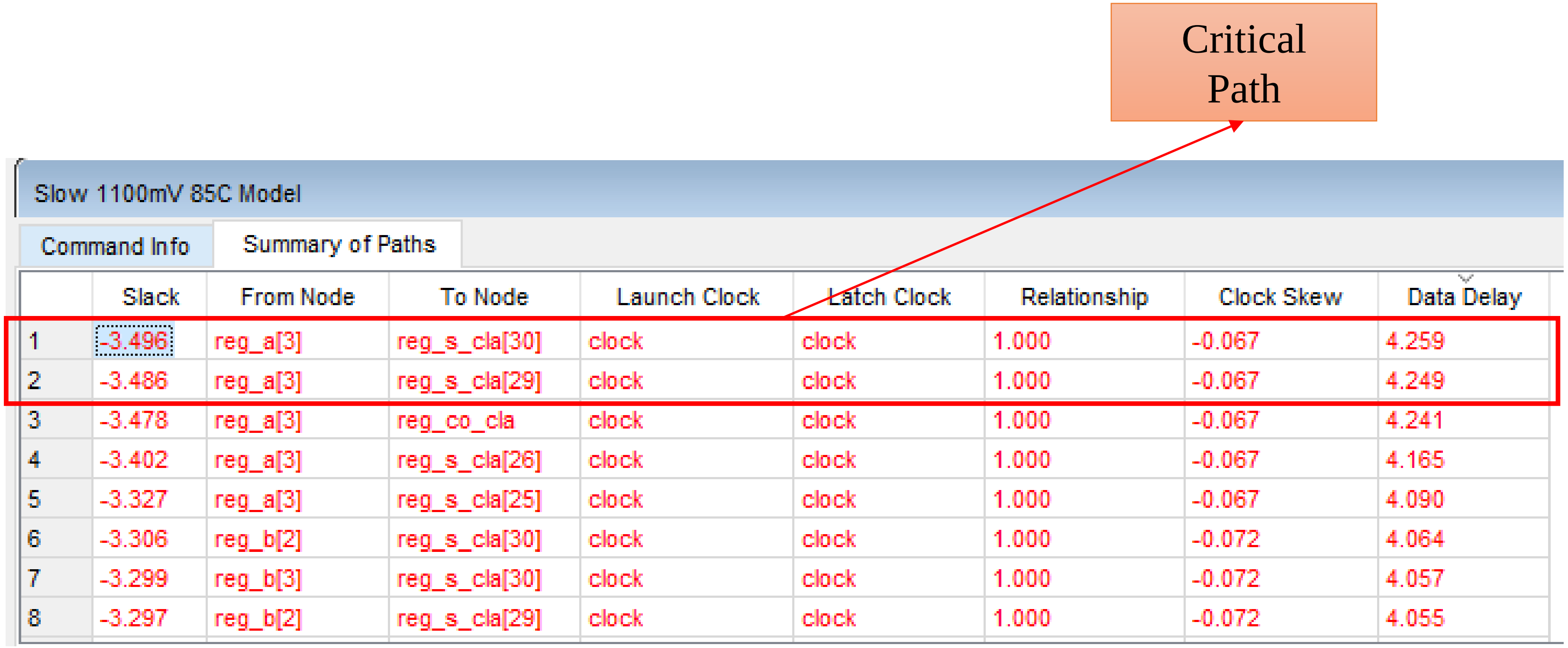
Paths
Report number of paths: 10
Maximum number of paths per endpoint:
Maximum slack limit: ns
Pairs only

Output
Detail level: Full path
Show routing
[✓] Report panel name: Report Timing
[✓] Enable multi corner reports
File name:
File options: Overwrite Append
Console

Tcl command: _cla[30]} -setup -npaths 10 -detail full_path -panel_name {Report Timing} -multi_corner

Timing Analysis of CLA (3/11)

- Setup: clock – Summary of Paths



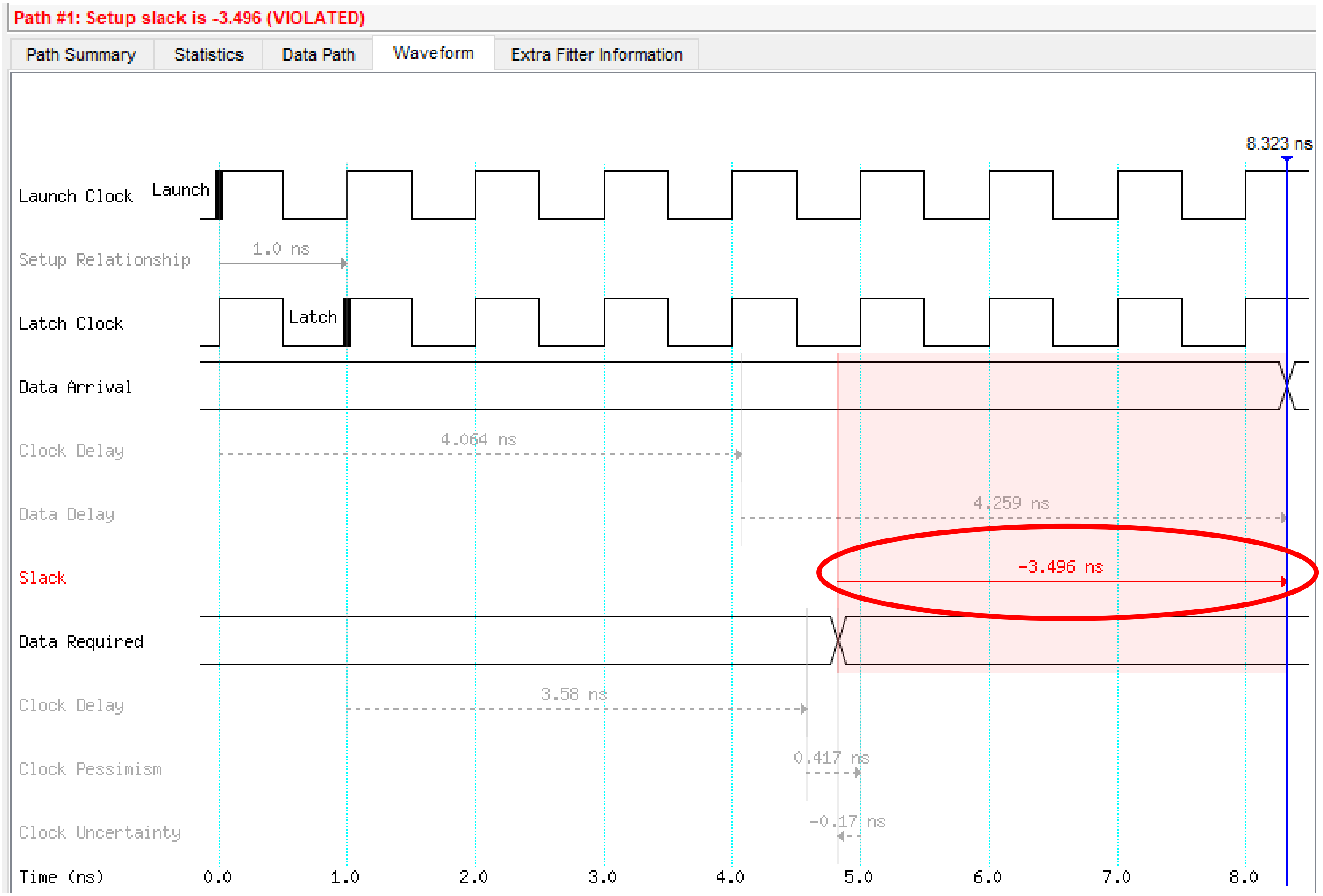
Slow 1100mV 85C Model

Command Info | Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-3.496	reg_a[3]	reg_s_cla[30]	clock	clock	1.000	-0.067	4.259
2	-3.486	reg_a[3]	reg_s_cla[29]	clock	clock	1.000	-0.067	4.249
3	-3.478	reg_a[3]	reg_co_cla	clock	clock	1.000	-0.067	4.241
4	-3.402	reg_a[3]	reg_s_cla[26]	clock	clock	1.000	-0.067	4.165
5	-3.327	reg_a[3]	reg_s_cla[25]	clock	clock	1.000	-0.067	4.090
6	-3.306	reg_b[2]	reg_s_cla[30]	clock	clock	1.000	-0.072	4.064
7	-3.299	reg_b[3]	reg_s_cla[30]	clock	clock	1.000	-0.072	4.057
8	-3.297	reg_b[2]	reg_s_cla[29]	clock	clock	1.000	-0.072	4.055

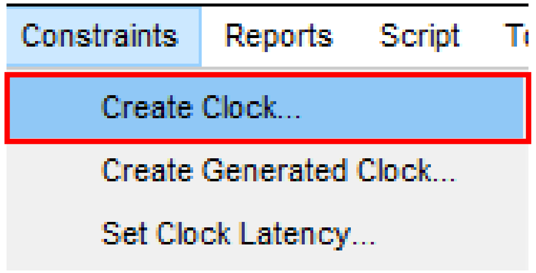
Timing Analysis of CLA (4/11)

- Setup: clock - waveform
  - ✓ Slack이 음수이기 때문에 clock을 조절할 필요가 발생

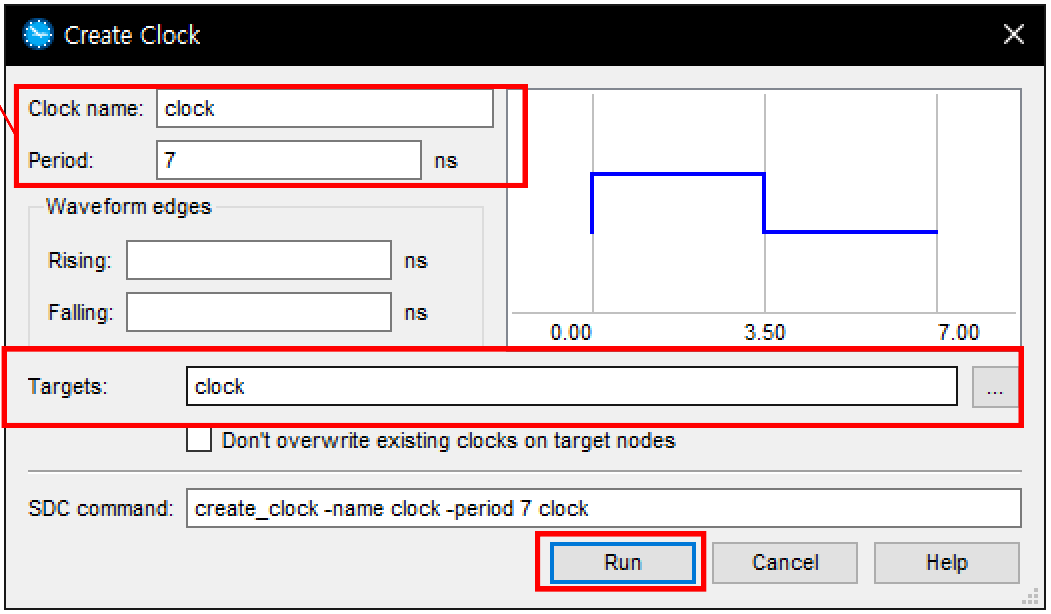


Timing Analysis of CLA (5/11)

- Constraints ➔ Create Clock



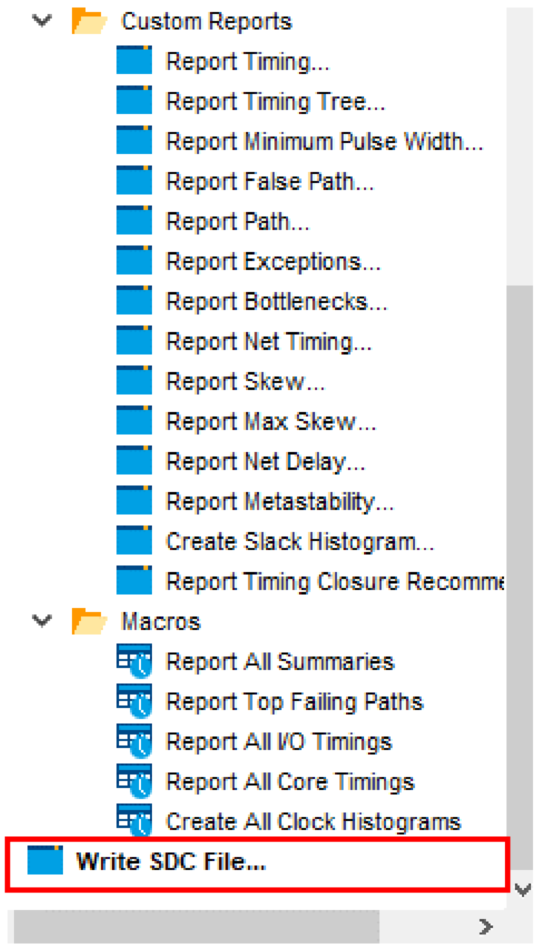
CLA module의 clock signal name을 적을 것

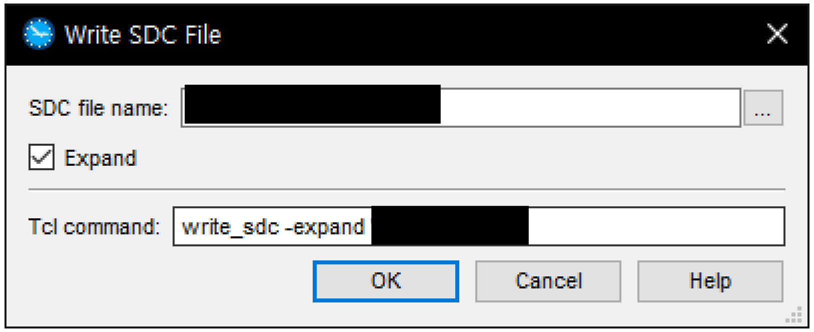


Clock name: clock
Period: 7 ns
Rising: ns
Falling: ns
Targets: clock
Don't overwrite existing clocks on target nodes
SDC command: create_clock -name clock -period 7 clock

Timing Analysis of CLA (6/11)

- Double clock
  - Write SDC File





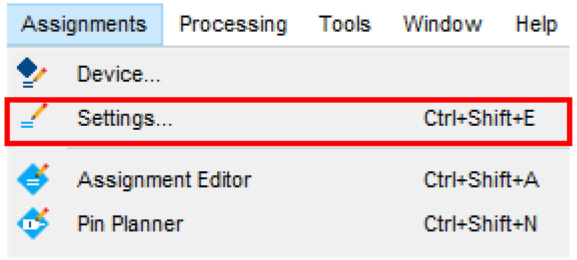
SDC file name: [redacted]

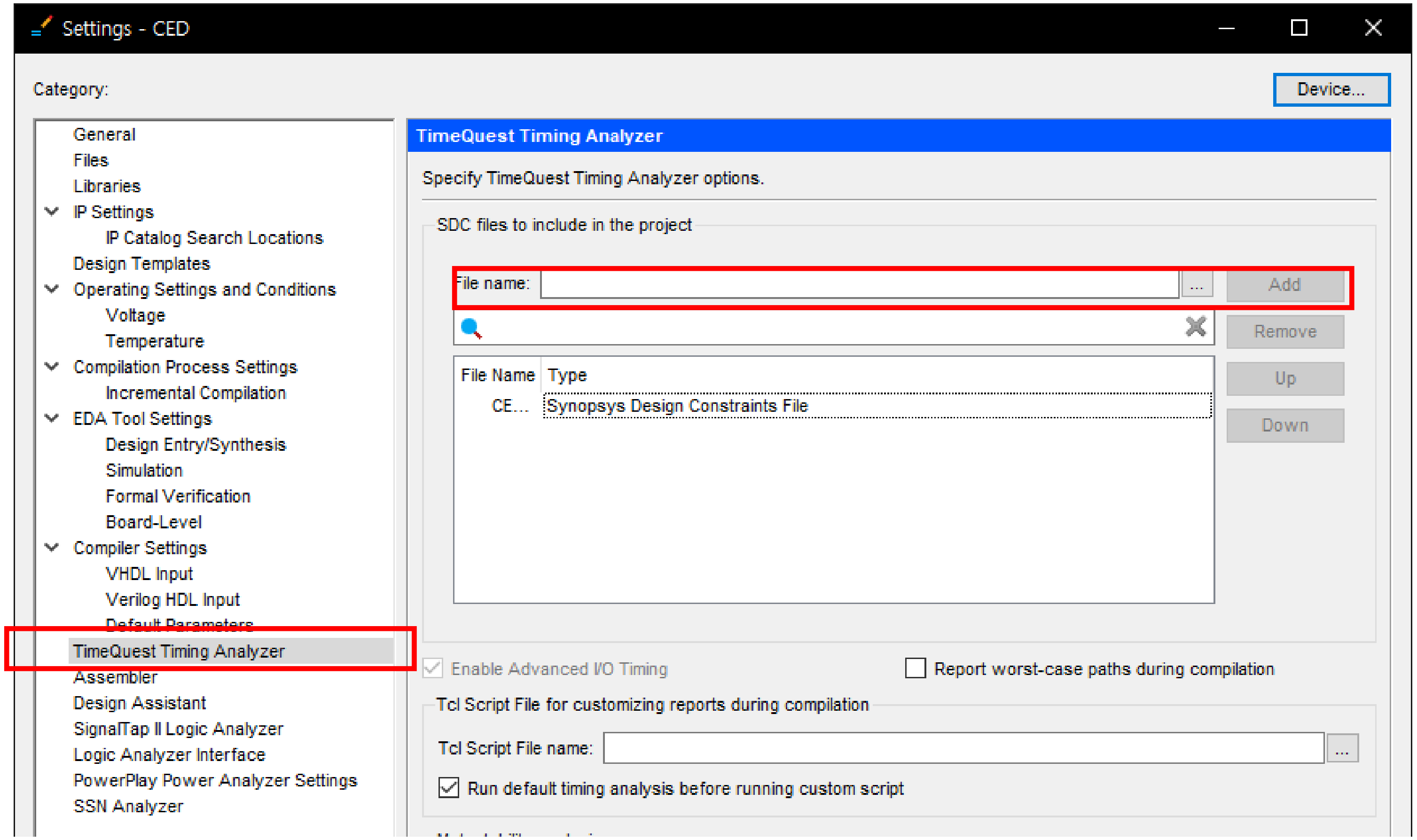
[✓] Expand

Tcl command: write_sdc -expand [redacted]

Timing Analysis of CLA (7/11)

- Assignments ➔ Settings
- TimeQuest Timing Analyzer → Add sdc file





Specify TimeQuest Timing Analyzer options.

SDC files to include in the project

File name:

File Name	Type
CE...	Synopsys Design Constraints File

☑ Enable Advanced I/O Timing ☐ Report worst-case paths during compilation

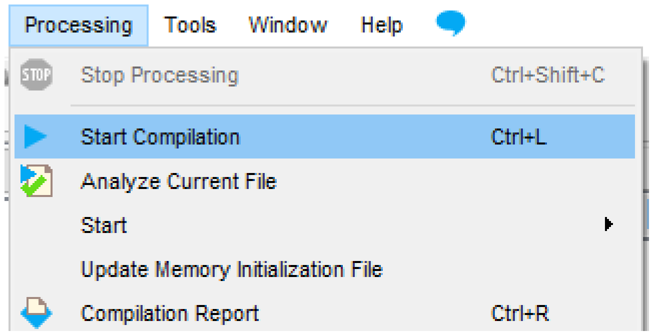
Tcl Script File for customizing reports during compilation

Tcl Script File name:

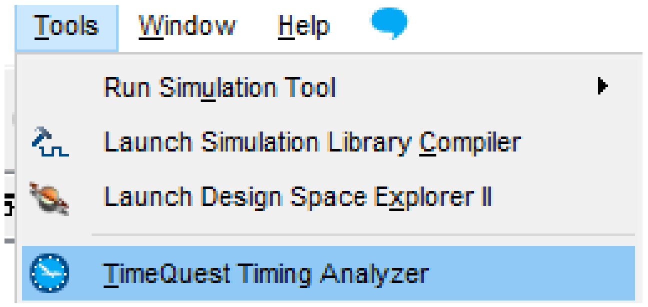
☑ Run default timing analysis before running custom script

Timing Analysis of CLA (8/11)

- Recompile

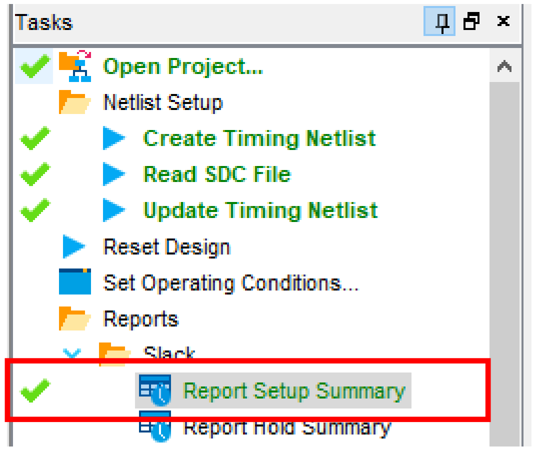

Processing
Tools
Window
Help
Stop Processing Ctrl+Shift+C
Start Compilation Ctrl+L
Analyze Current File
Start
Update Memory Initialization File
Compilation Report Ctrl+R

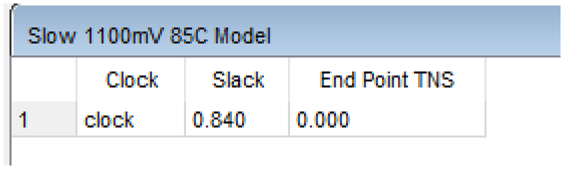
- Tools ➔ TimeQuest Timing Analyzer


Tools
Window
Help
Run Simulation Tool
Launch Simulation Library Compiler
Launch Design Space Explorer II
TimeQuest Timing Analyzer

Timing Analysis of CLA (9/11)

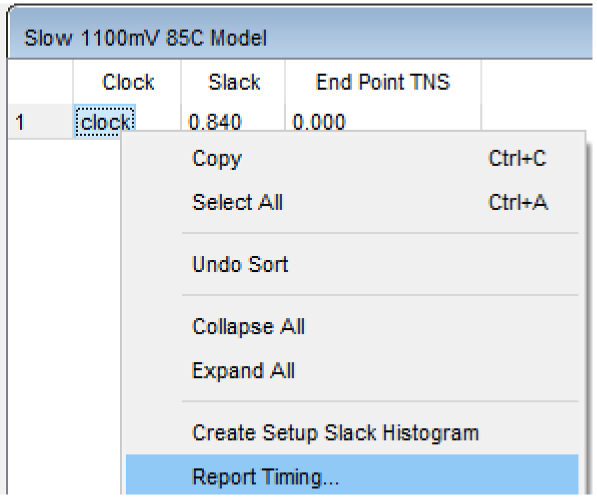
- Double click
  - ✓ Create Timing Netlist
  - ✓ Read SDC File
  - ✓ Update Timing Netlist
  - ✓ Report Setup Summary





Slow 1100mV 85C Model

	Clock	Slack	End Point TNS
1	clock	0.840	0.000

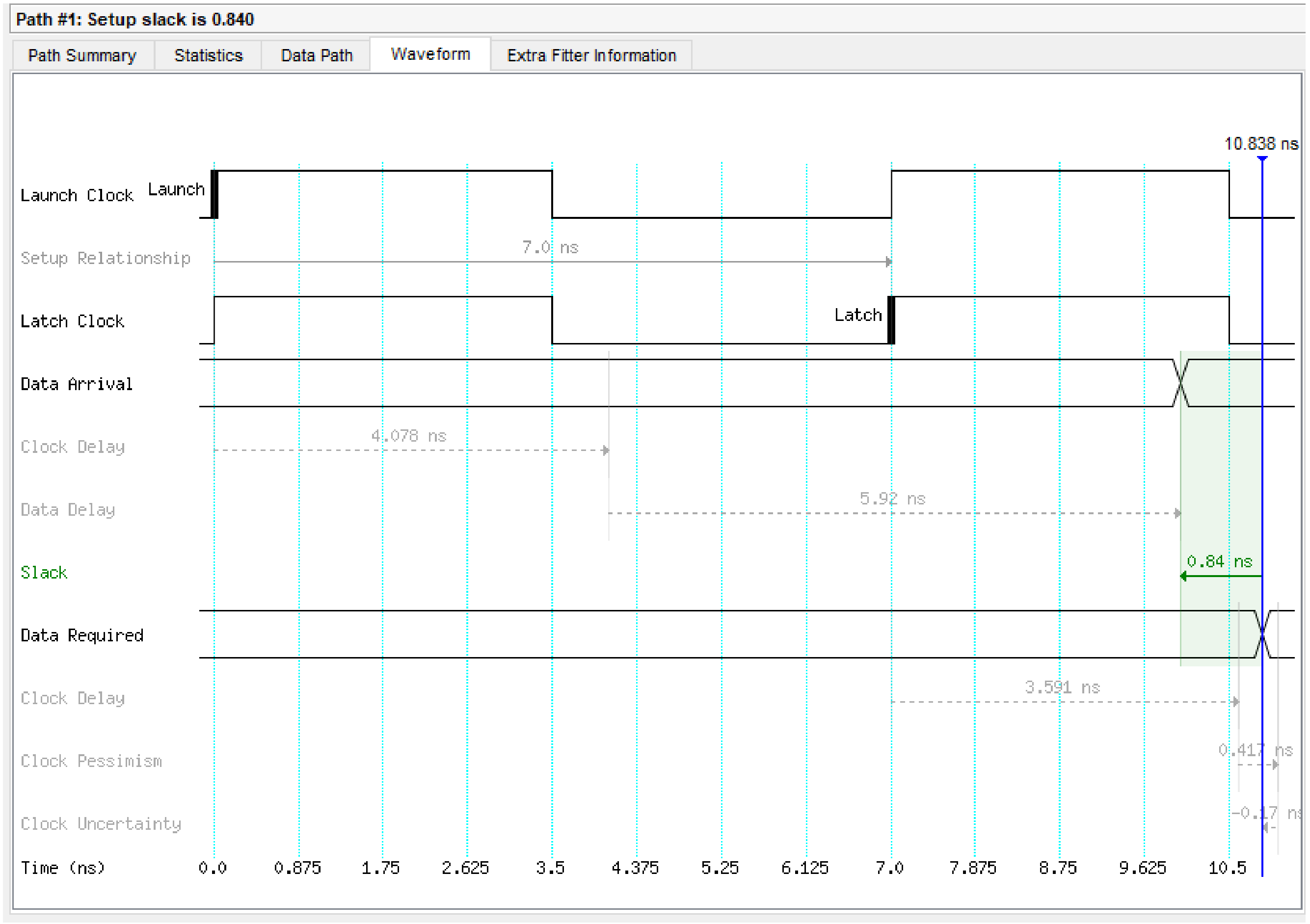


Slow 1100mV 85C Model

	Clock	Slack	End Point TNS
1	clock	0.840	0.000

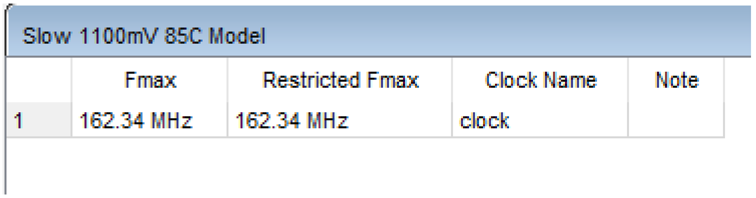
Timing Analysis of CLA (10/11)

- Setup: clock
  - ✓ Waveform에서 slack이 양수이기 때문에 violation이 발생하지 않음을 확인


Path #1: Setup slack is 0.840

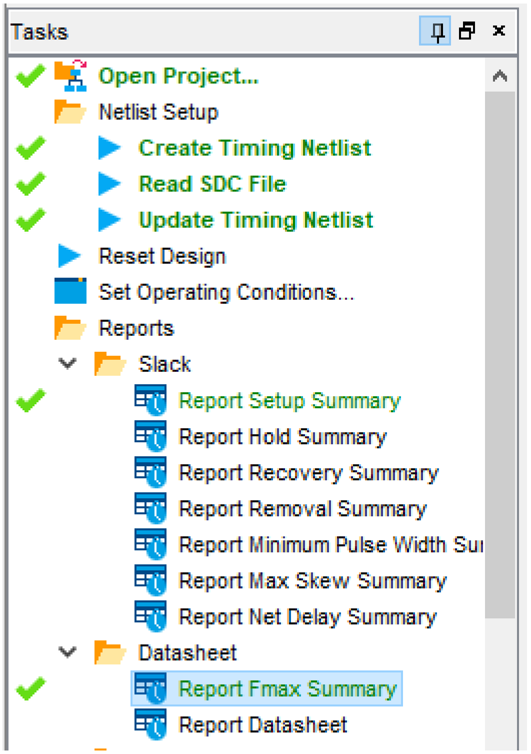
Timing Analysis of CLA (11/11)

- Double click
  - Report Fmax Summary



Slow 1100mV 85C Model

	Fmax	Restricted Fmax	Clock Name	Note
1	162.34 MHz	162.34 MHz	clock	



Tasks

- ✓ Open Project...
- Netlist Setup
  - ✓ Create Timing Netlist
  - ✓ Read SDC File
  - ✓ Update Timing Netlist
- Reset Design
- Set Operating Conditions...
- Reports
  - Slack
    - ✓ Report Setup Summary
    - Report Hold Summary
    - Report Recovery Summary
    - Report Removal Summary
    - Report Minimum Pulse Width Sur
    - Report Max Skew Summary
    - Report Net Delay Summary
  - Datasheet
    - ✓ Report Fmax Summary
    - Report Datasheet

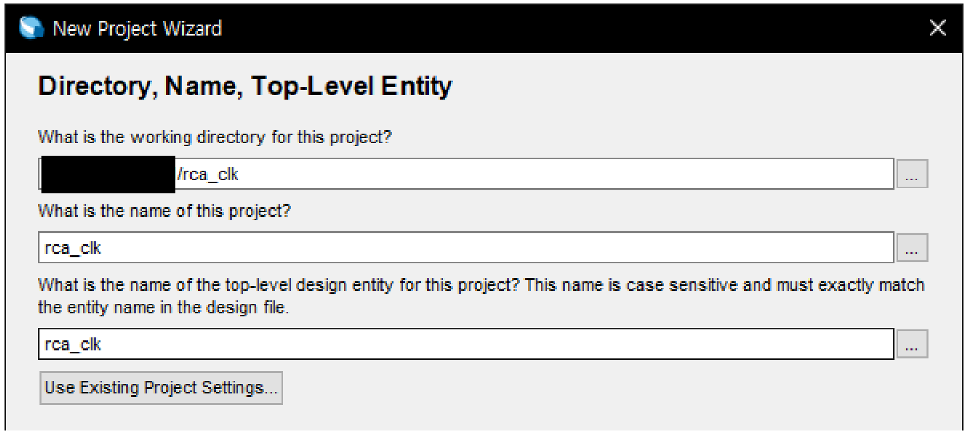
  - 해당 모듈(cla_clk)에서 violation이 발생하지 않는 최대의 frequency는 162.34 MHz인 것을 확인

Timing Analysis of 32-bit RCA

PRACTICE III

Timing Analysis of RCA

- New Project Wizard
  - ✓ Project name : rca_clk
  - ✓ Family & Device : Cyclone V 5CSXFC6D6F31C6 (밑에서 6번째)
- Verilog file
  - ✓ Add file: gates.v, ha.v, fa.v,
  - ✓ New file : rca4.v, rca32.v, rca_clk.v



Directory, Name, Top-Level Entity

What is the working directory for this project?

[redacted]/rca_clk

What is the name of this project?

rca_clk

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

rca_clk

4-bits Ripple Carry Adder

- 4-bits RCA

```
module rca4(a, b, ci, s, co);
  input [3:0] a,b;
  input ci;
  output [3:0] s;
  output co;

  wire [2:0] c;

  fa U0_fa(.a(a[0]), .b(b[0]), .ci(ci),.s(s[0]), .co(c[0]));
  fa U1_fa(.a(a[1]), .b(b[1]), .ci(c[0]),.s(s[1]), .co(c[1]));
  fa U2_fa(.a(a[2]), .b(b[2]), .ci(c[1]),.s(s[2]), .co(c[2]));
  fa U3_fa(.a(a[3]), .b(b[3]), .ci(c[2]),.s(s[3]), .co(co));

endmodule
```

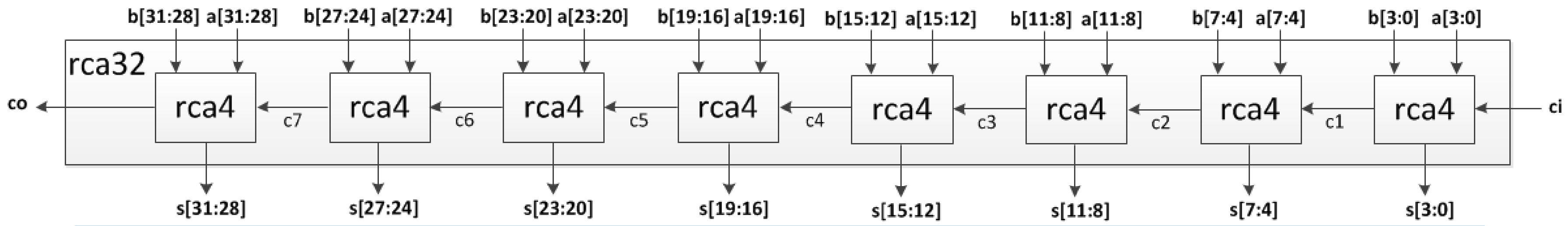
```
module fa(a,b, ci, s, co);
  input a, b, ci;
  output s, co;

  wire [2:0]w;

  ha U0_ha(.a(b), .b(ci), .s(w[0]), .co(w[1]));
  ha U1_ha(.a(a), .b(w[0]), .s(s), .co(w[2]));
  _or2 U2_or(.a(w[1]),.b(w[2]), .y(co));
endmodule
```

```
module ha(a, b, s, co);
  input a, b;
  output s, co;
  assign s = a ^ b;
  assign co = a & b;
endmodule
```

ha의 경우, 위와 같이 assign 구문을 사용하는 방법으로 변경한다.

32-bits Ripple Carry Adder

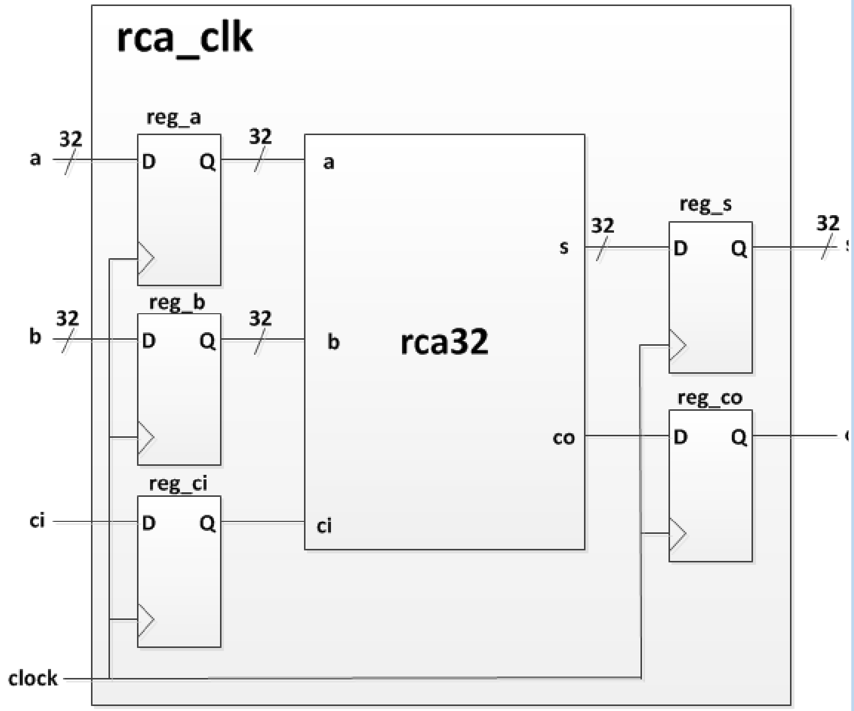
- 32-bits RCA
  - ✓ File name : rca32.v – Module name : rca32
  - ✓ 앞서 구현한 4-bit rca 8개를 instance하여 직렬로 연결하여 구현



```
module rca32(a,b,ci,s,co);
  input [31:0] a,b;
  input ci;
  output [31:0] s;
  output co;

  wire c1,c2,c3,c4,c5,c6,c7;

                    Instance of rca4

endmodule
```

RCA with Register

- rca_clk module



```
module rca_clk(clk,a,b,ci,s,co);
   input               clk;
   input  [31:0]       a,b;
   input               ci;
   output [31:0]       s;
   output              co;

   reg [31:0]    reg_a, reg_b;
   reg           reg_ci;
   reg [31:0]    reg_s;
   reg           reg_co;

   wire  [31:0] wire_s;
   wire wire_co;

   always@(posedge clk)
   begin
```

always

```
   end
```

Instance of rca32

assign

```
endmodule
```

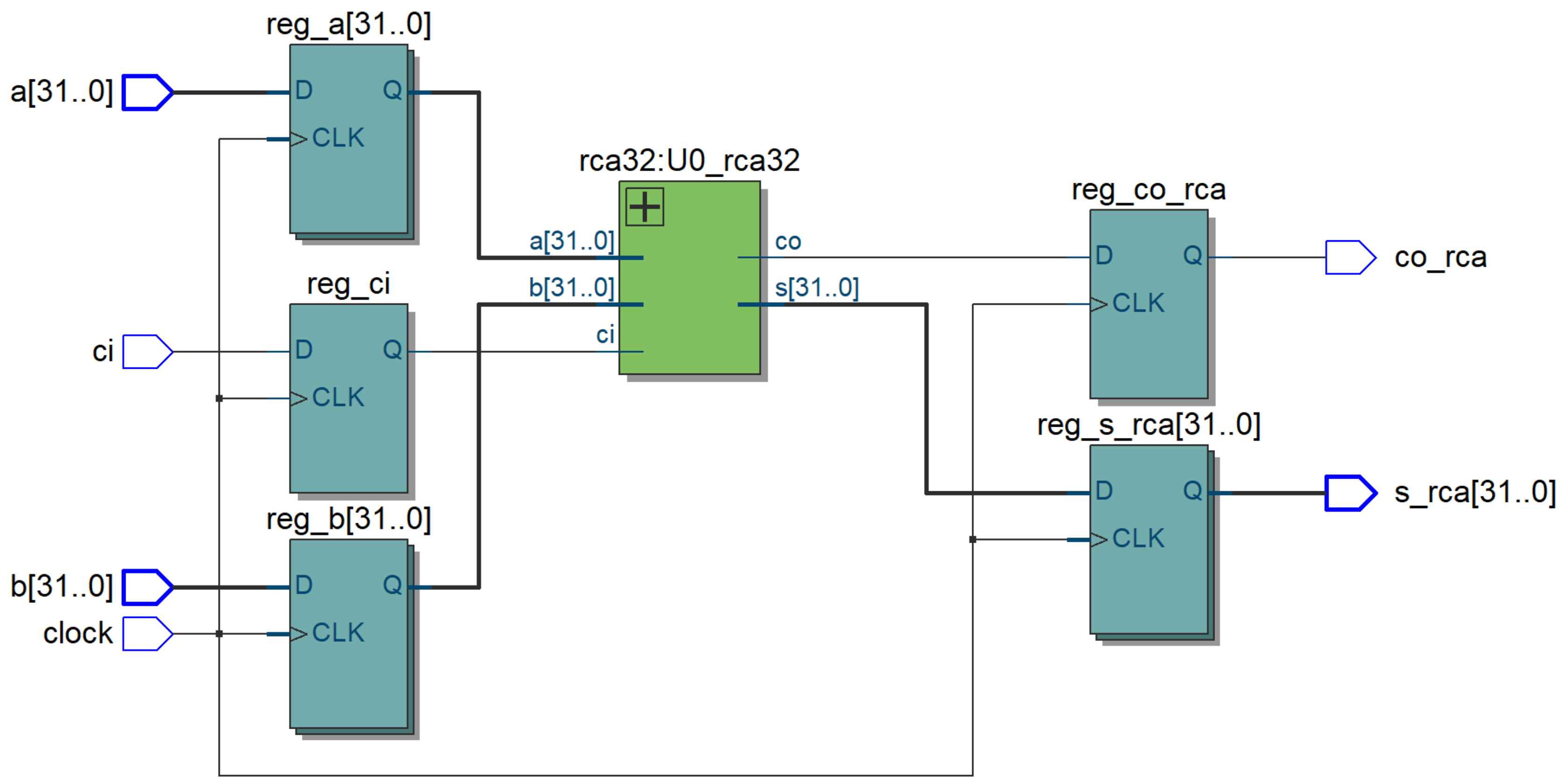
Testbench

```
`timescale 1ns/100ps

module tb_rca_clk;
  reg           clk;
  reg   [31:0]  tb_a, tb_b;
  reg           tb_ci;
  wire  [31:0]  tb_s;
  wire          tb_co;

  parameter STEP = 10;

  rca_clk U0_rca_clk(.clk(clk), .a(tb_a), .b(tb_b), .ci(tb_ci),
        .s(tb_s), .co_rca(tb_co));

  always # (STEP/2) clk = ~clk;

  initial
  begin
```

Testbench

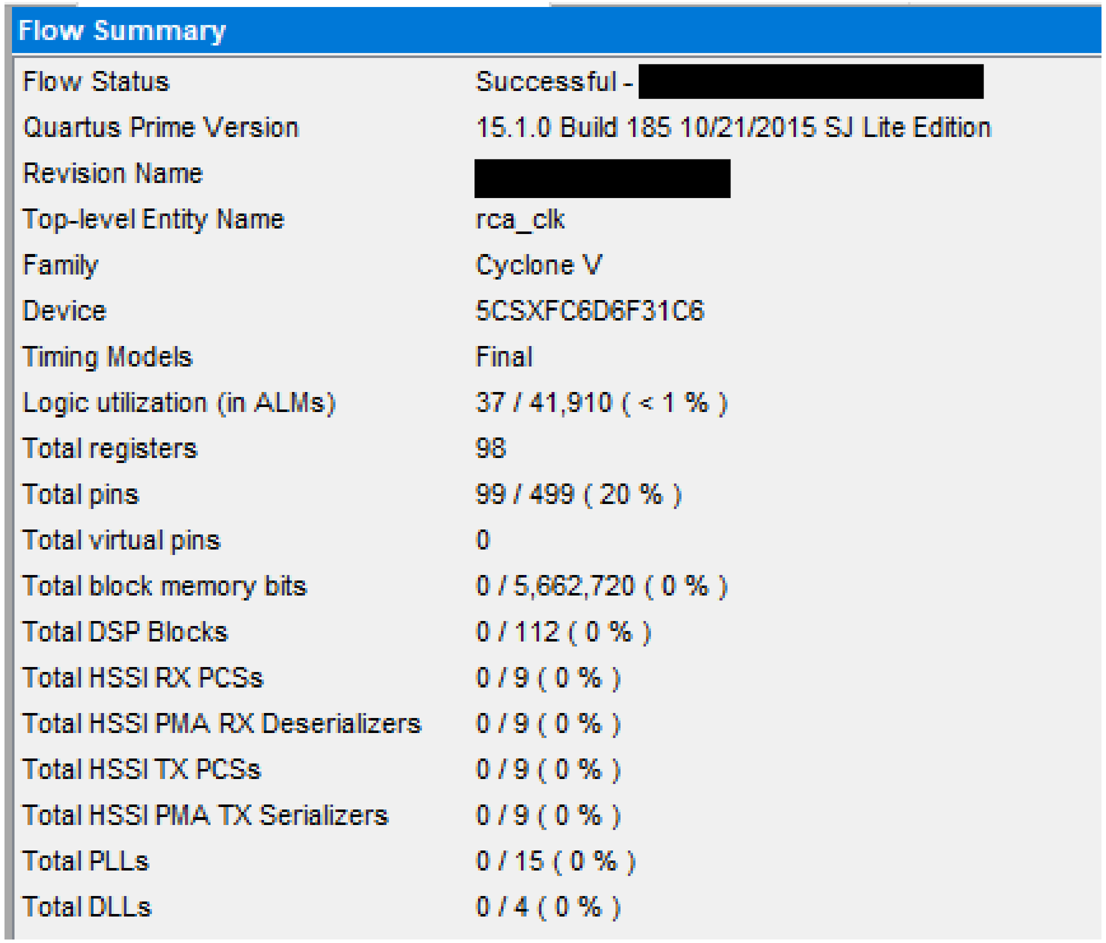
```
  end
endmodule
```

RTL Viewer


reg_a[31..0]
a[31..0]
D
Q
CLK
rca32:U0_rca32
reg_co_rca
a[31..0]
co
D
Q
co_rca
reg_ci
b[31..0]
s[31..0]
CLK
ci
D
Q
ci
CLK
reg_s_rca[31..0]
D
Q
s_rca[31..0]
reg_b[31..0]
CLK
b[31..0]
D
Q
clock
CLK

Compilation Report

- Flow Summary
  - cla_clk module과 크기를 비교하여 본다.

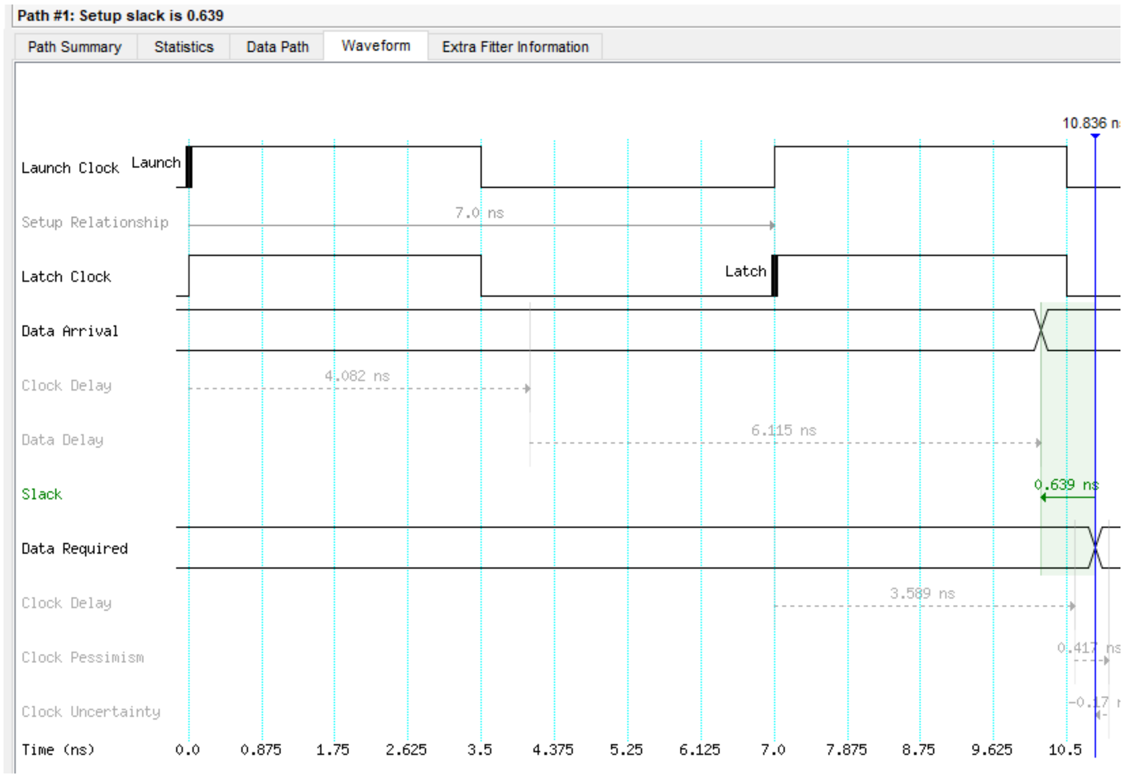


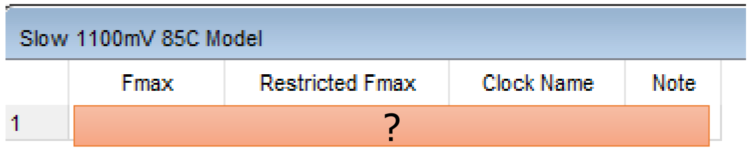
Flow Summary

Flow Status	Successful - [redacted]
Quartus Prime Version	15.1.0 Build 185 10/21/2015 SJ Lite Edition
Revision Name	[redacted]
Top-level Entity Name	rca_clk
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	37 / 41,910 (< 1 %)
Total registers	98
Total pins	99 / 499 (20 %)
Total virtual pins	0
Total block memory bits	0 / 5,662,720 (0 %)
Total DSP Blocks	0 / 112 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 15 (0 %)
Total DLLs	0 / 4 (0 %)

Timing Analysis of RCA(Cont.)

- TimeQuest
  - CLA에서 했던 방법처럼 TimeQuest를 사용하여 timing을 분석한다.


Path #1: Setup slack is 0.639


Slow 1100mV 85C Model

	Fmax	Restricted Fmax	Clock Name	Note
1	?			

Assignment 3

- Report
  - ✓ 자세한 사항은 lab document 참고

- Submission
  - ✓ 과제 기한은 공지 참고
  - ✓ 늦은 숙제는 제출 이틀 후 까지만 받음(20% 감점)

채점기준

세부사항		점수	최상	상	중	하	최하
소스코드	Source code가 잘 작성 되었는가? (Structural design으로 작성되었는가?)	10	10	8	5	3	0
	주석을 적절히 달았는가? (반드시 영어로 주석 작성)	20	20	15	10	5	0
설계검증 (보고서)	보고서를 성실히 작성하였는가? (보고서 형식에 맞추어 작성)	30	30	20	10	5	0
	합성결과를 설명하였는가?	10	10	8	5	3	0
	검증을 제대로 수행하였는가? (모든 입력 조합, waveform 설명)	30	30	20	10	5	0
총점		100					

References

- Altera Co., www.altera.com/
- 이준환, 디지털논리회로2 강의자료, 광운대학교, 컴퓨터 공학과, 2021

Q&A

THANK YOU