* **Project Title:** Reg2Reg Synthesis in Synopsys Design Compiler
* **Name:** Thotla Umesh
* **Date:** 13-08-2025
* **Tool Used:** Synopsys Design Compiler (DC Shell)
* **Technology Node:** 28nm

**1. Introduction**

This project focuses on the synthesis of a **Register-to-Register (Reg2Reg)** design using Synopsys Design Compiler (DC Shell). In digital systems, a Reg2Reg path refers to a data path that starts from one register and ends at another register, possibly passing through combinational logic in between. Such designs are critical in synchronous circuits where timing closure plays a key role in achieving the desired clock performance.

The given RTL design, written in Verilog, consists of two main parts:

1. **Combinational Logic:** Implements basic logical operations between inputs (a, b, c, d) and an intermediate register value (qint).
2. **Sequential Logic:** Two registers (q and qint) store values on the rising edge of the clock (clk) or reset asynchronously on the falling edge of the reset signal (rst).

The synthesis process involves converting this RTL description into a gate-level netlist mapped to a standard cell library, while applying timing constraints to meet design performance goals.

The major steps performed in this synthesis flow include:

* **Setting target and link libraries** to define the technology and cell models.
* **Analyzing and elaborating the RTL code** to ensure it is syntactically correct and structurally complete.
* **Defining the clock** and applying **input/output delays** to model the real operating environment.
* **Setting driving cells** to represent upstream circuit driving strength.
* **Compiling the design** to generate an optimized gate-level netlist.
* **Generating reports** (report\_timing, report\_constraints, report\_qor) to verify timing, applied constraints, and overall quality of results.

## ****2. Design Description****

The given Verilog code describes a **Register-to-Register (Reg2Reg)** design that contains both **combinational** and **sequential** logic. It is structured as follows:

### **Module Declaration**

module reg2reg(output reg q, input a, b, c, d, clk, rst);

* **Output:** q (1-bit register) – final registered output.
* **Inputs:**
  + a, b, c, d → data inputs (1-bit each)
  + clk → clock signal
  + rst → asynchronous reset signal

### **Internal Signals**

reg qint;

wire combo\_out1, combo\_out2;

* **qint**: An internal register used as an intermediate storage between two stages.
* **combo\_out1 and combo\_out2**: Wires that hold outputs of combinational logic blocks.

### **Combinational Logic**

assign combo\_out1 = a & b;

assign combo\_out2 = c & d & qint;

* combo\_out1 is generated by performing a logical AND between inputs a and b.
* combo\_out2 is generated by performing a logical AND between inputs c, d, and the intermediate register value qint.

### **Sequential Logic**

always @(posedge clk or negedge rst)

begin

if (!rst)

begin

q <= 0;

qint <= 0;

end

else

begin

q <= combo\_out2;

qint <= combo\_out1;

end

end

* **Triggering:**
  + On **positive edge** of the clock (clk)
  + Or **negative edge** of the reset signal (rst)
* **Reset Behavior:**
  + When rst is low (0), both q and qint are reset to 0.
* **Normal Operation:**
  + On each rising clock edge, q stores the value of combo\_out2.
  + At the same time, qint stores the value of combo\_out1.

### **Functional Summary**

This design effectively models a **two-stage register pipeline**:

* **Stage 1:** combo\_out1 → stored in qint
* **Stage 2:** combo\_out2 (depends on qint) → stored in q

Such a structure is common in **pipelined digital circuits** to improve performance and meet timing requirements by breaking a long combinational path into shorter ones.

## ****3. Tool Flow Steps in Synopsys Design Compiler****

The synthesis of the Reg2Reg design was carried out in Synopsys Design Compiler (DC Shell) using the following steps:

### **3.1 Set Target Library**

Specifies the standard cell library used for synthesis.

set target\_library "<path\_to\_standard\_cell\_library>"

### **3.2 Set Link Library**

Defines the libraries used for linking and resolving references.

set link\_library "\* $target\_library"

### **3.3 Analyze RTL Code**

Reads the Verilog source file and checks for syntax errors.

analyze -format verilog reg2reg.v

### **3.4 Elaborate Design**

Converts the RTL into a generic design representation.

elaborate reg2reg

### **3.5 Create Clock**

Defines the clock with a given period.

create\_clock -period 0.2 [get\_ports clk]

### **3.6 Set Input Delays**

Models the delay from the external source to the input ports.

set\_input\_delay 0.04 -clock clk [all\_inputs]

### **3.7 Set Output Delays**

Models the delay from the output ports to the external destination.

set\_output\_delay 0.04 -clock clk [all\_outputs]

### **3.8 Set Driving Cell**

Specifies the driving strength for input ports.

set\_driving\_cell -lib\_cell BUFX2 [all\_inputs]

### **3.9 Compile Design**

Optimizes and maps the RTL design to standard cells.

compile

### **3.10 Generate Reports**

* **Timing Report:**

report\_timing

* **Constraints Report:**

report\_constraints

* **Quality of Results (QoR) Report:**

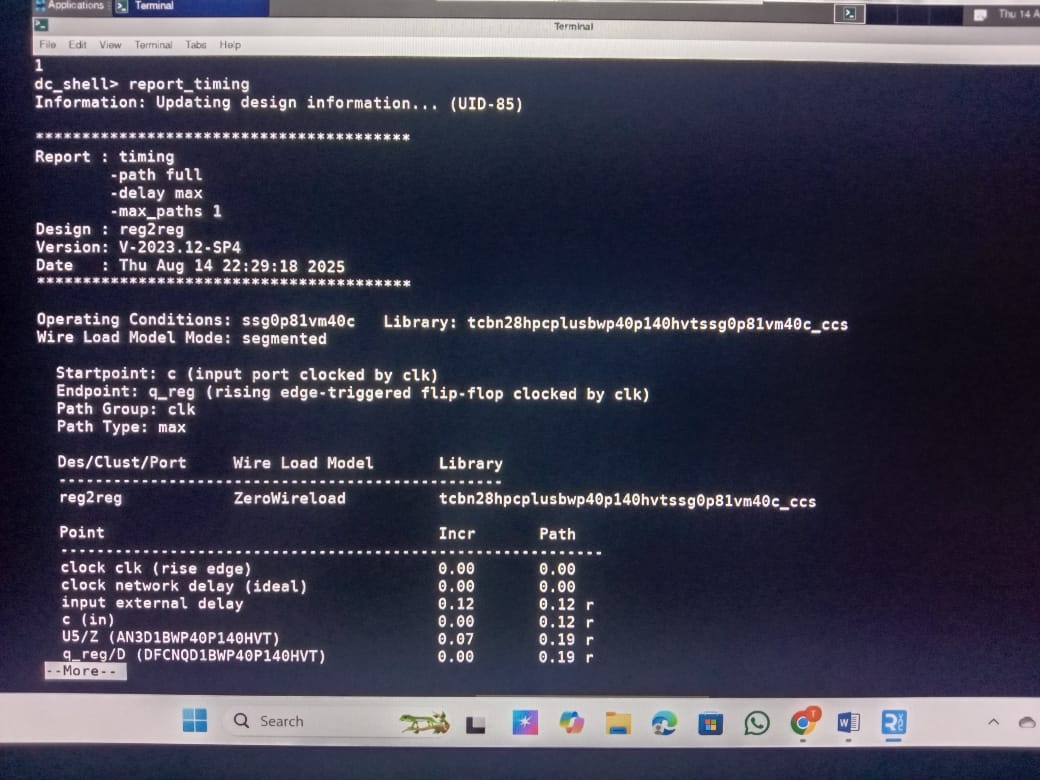
report\_qor

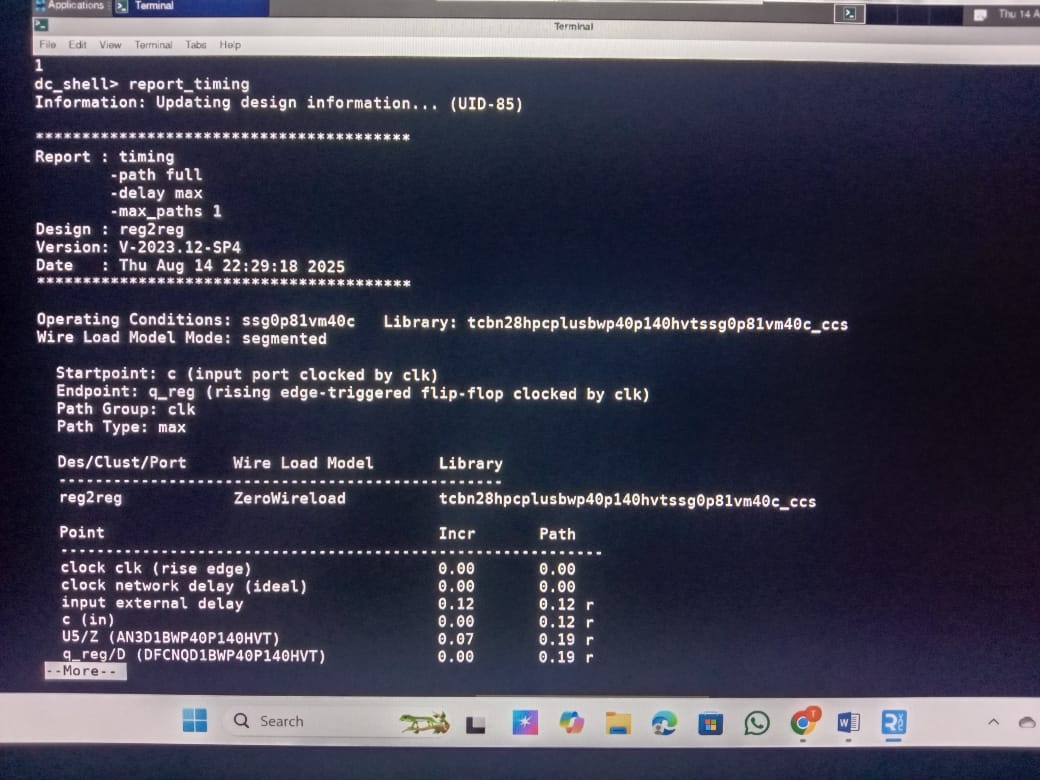
## ****4. Reports Generated****

Include **screenshots** of your DC Shell outputs for each report.

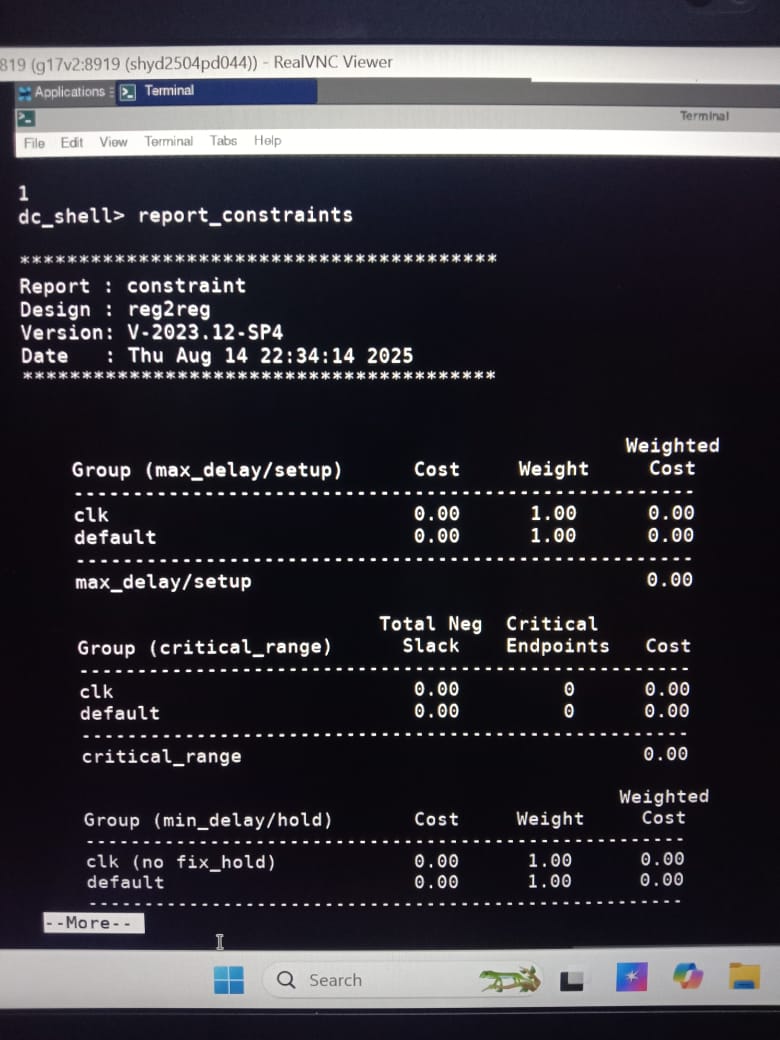
### 4.1 report\_timing

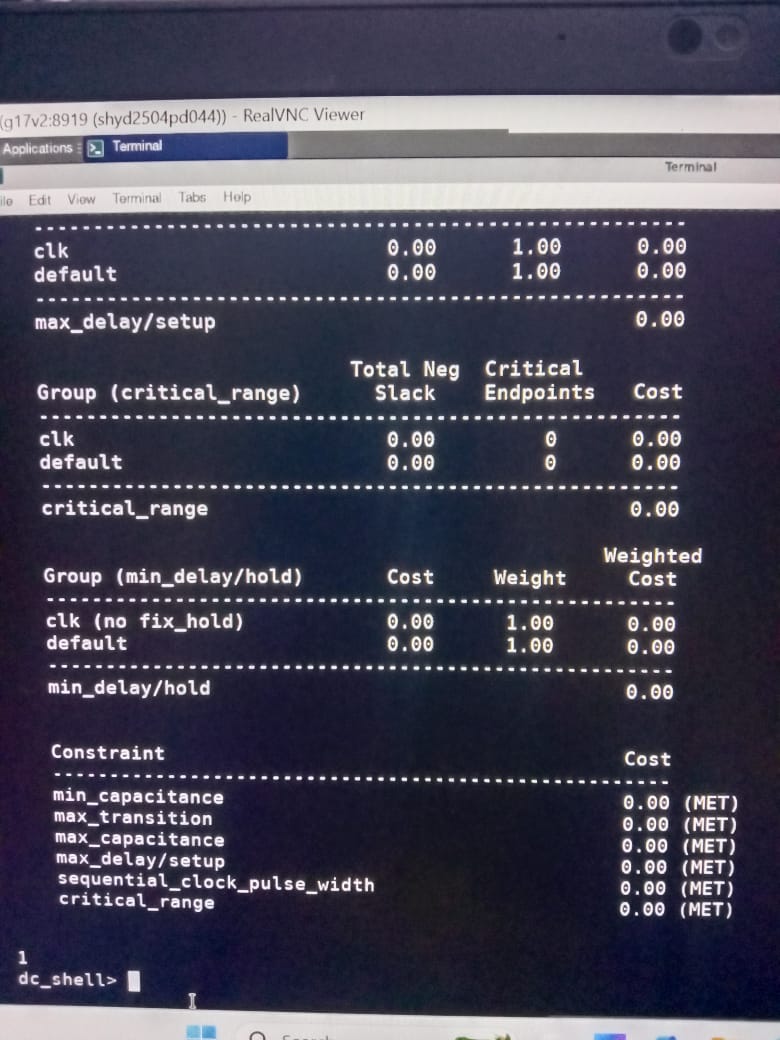
Purpose: Shows path delays, slack, and timing violations.



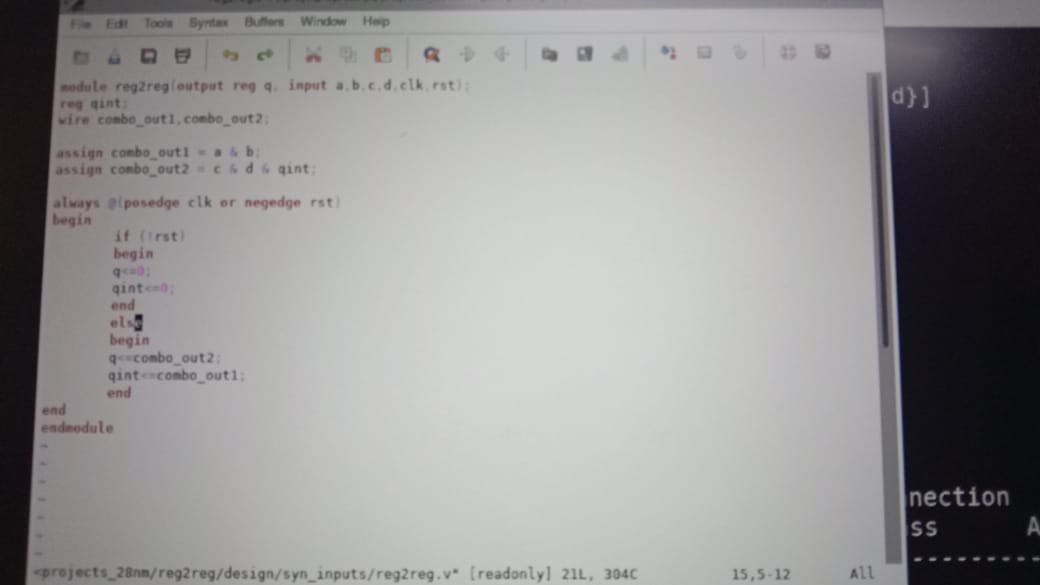


**4.2 report\_constraints**

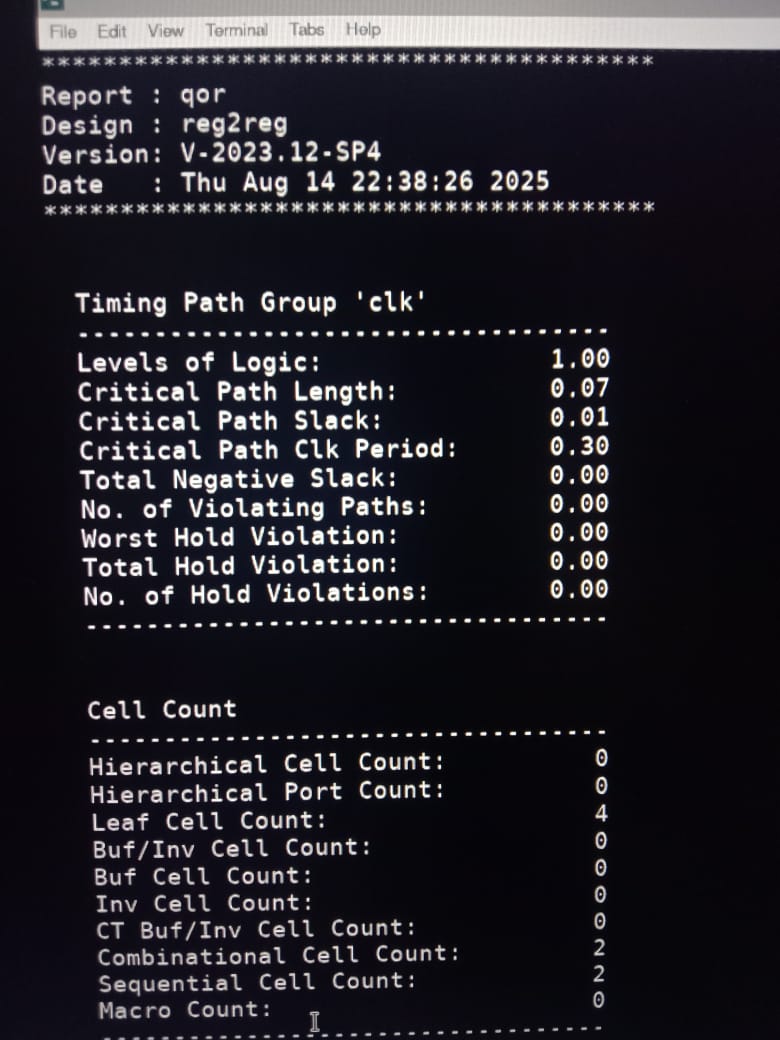


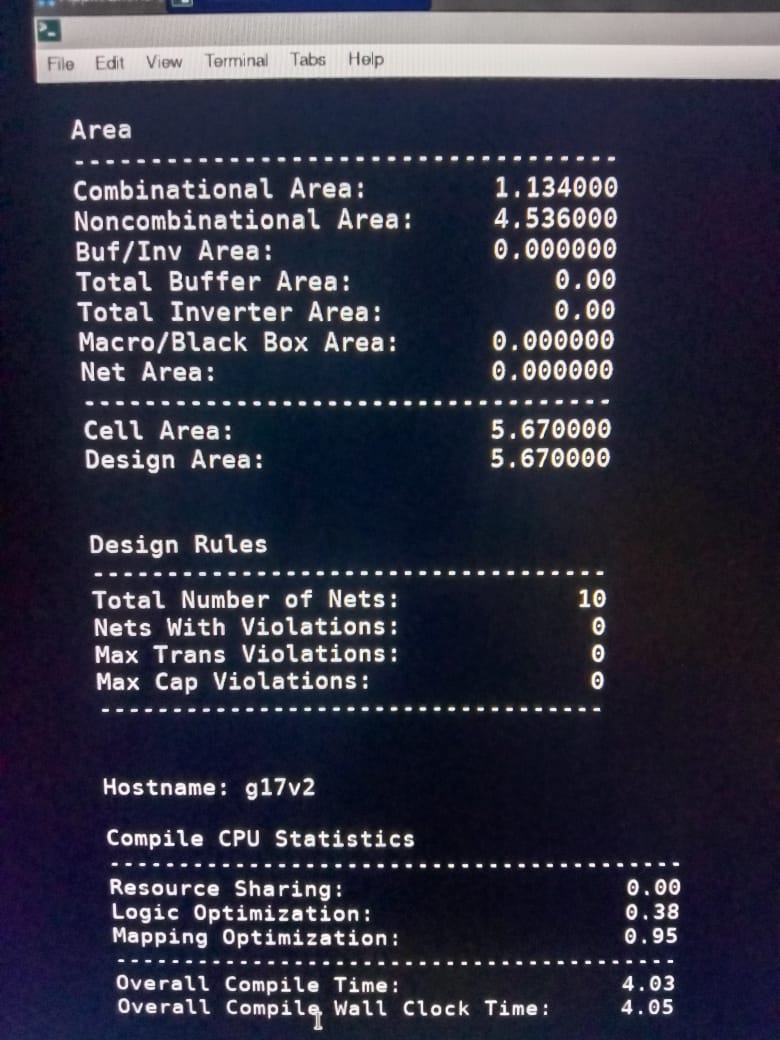


**My Verilog file: Module name:reg2reg::**

****

**Report\_qor(quality of results):**

****

****

## ****6. Conclusion****

In this project, a Register-to-Register (Reg2Reg) design was synthesized using Synopsys Design Compiler. The synthesis flow involved setting up technology libraries, analyzing and elaborating the RTL code, defining clock and timing constraints, applying driving strengths, and compiling the design to generate an optimized gate-level netlist.

The applied constraints ensured that the design was analyzed in a realistic operating environment, enabling accurate timing and performance evaluation. The generated reports — report\_timing, report\_constraints, and report\_qor — provided insights into timing closure, constraint validation, and overall design quality.

Through this process, the importance of proper constraint definition, clock setup, and cell selection in meeting timing requirements was demonstrated. This mini-project builds a strong foundation for understanding synthesis flows in ASIC design and serves as a stepping stone toward more complex designs in the VLSI Physical Design domain.