IC Design

Homework # 4

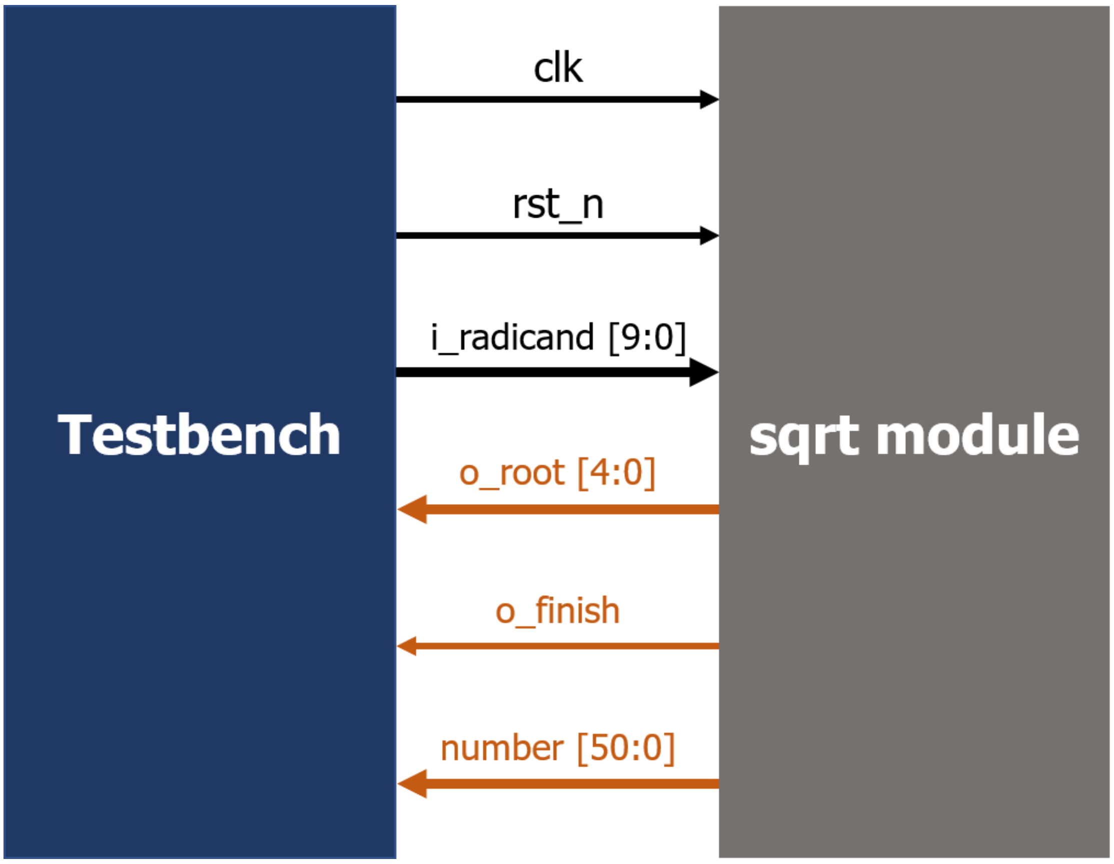
(Due on 2021/01/08, 13:20, Verilog code and report upload to CEIBA, no hardcopy)

✧ Plagiarism is not allowed. 10% penalty for each day of delay.

Problem Specification

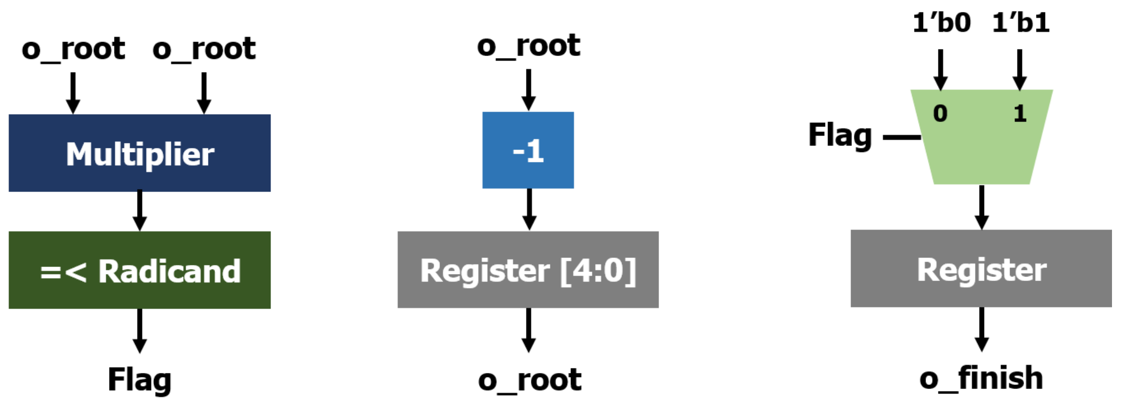
Design a circuit with reset that computes the **square root of an integer**. There is one input, i.e., i_radicand with 10 bits, and there is one output o_root with 5 bits. Note that both radicand and root are **unsigned integers(小數點後無條件捨去)**. The relation between inputs and outputs is

$$root = floor\left(abs(\sqrt{radicand})\right).$$



Note that the two output signals: "o_root [4:0]" and "o_finish" must be registered, i.e., they are outputs of DFFs **(use module FD2 (positive edge) in lib.v)**.

A possible architecture is as follows (not the best design):

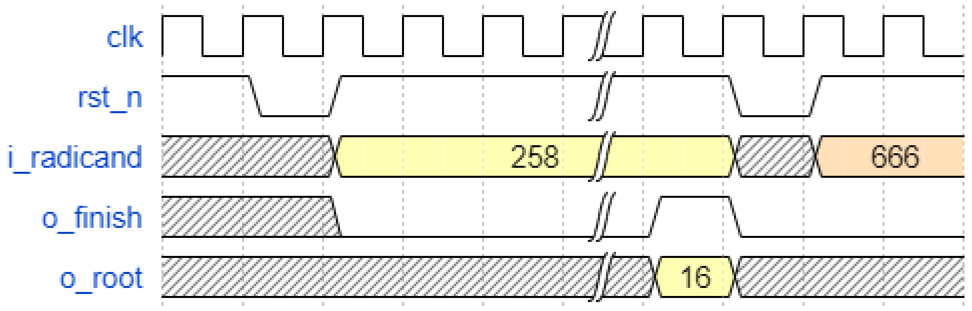


Timing Diagram

Since a design can either be recursive or pipelined, the testbench provides two input strategies, i.e., recursive and pipeline.

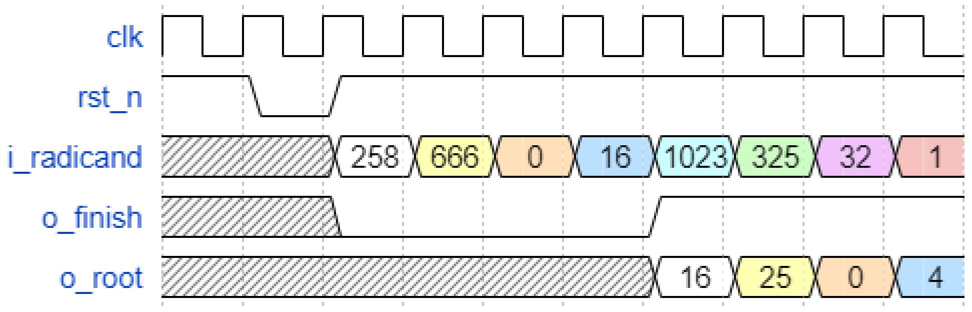
1. *Recursive*

In the recursive input strategy, **the circuit would be reset before every new radicand is input**. Note that the radicand would be input at the next cycle right after reset.



2. *Pipeline*

In the pipeline input strategy, the circuit would only be reset once. After reset, **a new radicand would be input every cycle**. Also, **o_finsh should maintain high once it was pulled up**, and the order of input radicands and their corresponding roots should not be changed (First in, first out.).



Signals Description

Signal name	I/O	Width	Simple description
clk	Input	1	Clock signal.
rst_n	Input	1	Active low asynchronous reset.
i_radicand	Input	10	Radicand number.
o_root	Output	5	The square root of the radicand.
o_finsih	Output	1	Indicate that the calculation was finished.
number	Output	51	The number of transistors.

Design Rules

Those who do not design according to the following rules will not be graded.

- **LUT-based designs are not allowed.**
- There should be a **reset signal** for the register.
- You are free to add pipeline registers.
- You can loosen your simulation timing first, (i.e., **`define CYCLE XXXX** in the testbench.v), then shorten the clock period to find your critical path.
- Your design should be based on the **standard cells in the lib.v**. All logic operations in your design **MUST consist of the standard cells** instead of using the operands such as "+", "-", "&", "|", ">", and "<".
- Design your homework in the given "sqrt.v" file. **You are NOT ALLOWED to change the filename and the header of the top module (i.e. the module name and the I/O ports)**.
- If your design contains more than one module, **don't create a new file for them**, just put those modules in "sqrt.v."

Grading Policy

1. *Gate-level design using Verilog (70%)*

Your score will depend on both the correctness and performance of your design.

(a) Correctness Score (40%)

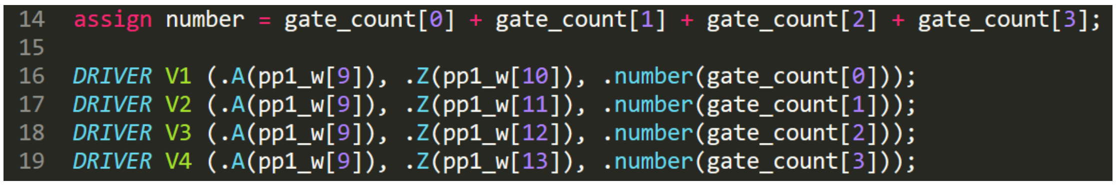
At this stage, we will only evaluate whether the function of the sqrt module is correct. Time and area are not considered. We provide a testbench with 1000 patterns which automatically grades your design. Your score in this part will be

$$40 \times \frac{correct\ number}{1000}.$$

(b) Performance Score (30%)

At this stage, you need to **add up the number of transistors of all used cells in the sqrt module and connect it to number [50:0]**.

Only in this section, you are allowed to use “assign” and “+” to help with calculations.



```
assign number = gate_count[0] + gate_count[1] + gate_count[2] + gate_count[3];

DRIVER V1 (.A(pp1_w[9]), .Z(pp1_w[10]), .number(gate_count[0]));
DRIVER V2 (.A(pp1_w[9]), .Z(pp1_w[11]), .number(gate_count[1]));
DRIVER V3 (.A(pp1_w[9]), .Z(pp1_w[12]), .number(gate_count[2]));
DRIVER V4 (.A(pp1_w[9]), .Z(pp1_w[13]), .number(gate_count[3]));
```

We will rank all students who pass ***(a)*** and have **no connection errors on number [50:0] port**. There will be a ranking according to **A*T**, where A represents the **number of transistors** and T represents the **total execution time**. Your performance score will be according to your ranking as the table below.

Percentage of passing students	**Performance Score**
If your ranking > 90 %	30
80% ~ 90%	27
70% ~ 80%	24
60% ~ 70%	21
50% ~ 60%	18
40% ~ 50%	15
30% ~ 40%	12
20% ~ 30%	9
10% ~ 20%	6
0% ~ 10%	3
Using operands, not standard cell logic	0
Correctness failed	0
Plagiarism	0

2. *Report (30%)*

(a) Simulation (0%)

Write down your **minimum cycle time** and **which strategy you used**. If you do not provide these information, “1. Gate-level design using Verilog (70%)” **will not get any score. This minimum cycle time would be verified by TAs.** Also, put the screenshot of the summary provided by the testbench in the report.

(b) Circuit diagram (10%)

You are encouraged to use software to help draw architecture **instead of hand drawing**. Plot it **hierarchically** so that readers can understand your design easily. **All of the above will improve your report score.**

(5%) Plot the gate-level circuit diagram of your design.

(5%) Plot critical path on the diagram above.

(c) Discussion (20%)

Discuss your design.

- ➢ (3%) Introduce your design.
- ➢ (2%) How do you cut your pipeline?
- ➢ (5%) How do you improve your critical path and the number of transistors?
- ➢ (5%) How do you trade-off between area and speed?
- ➢ (5%) Compare with other architectures you have designed (if any).

Notification

Following are the files you will need (available on the class website)

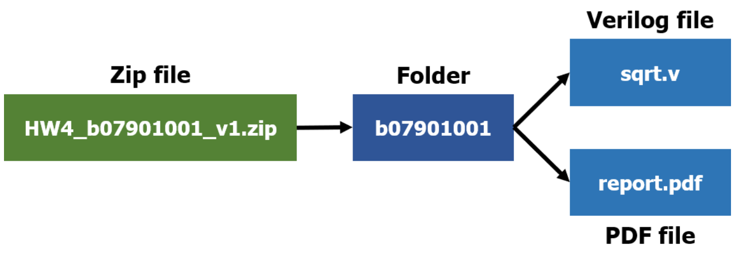
HW4.zip includes

- ■ **HW4_2020.pdf:** This document.
- ■ **HW4_tutorial_2020.pdf:** Tutorial in class.
- ■ **sqrt.v:**
  Dummy design file. Program the design in this file.
  The header of the top module and the declaration of the I/O ports are predefined in this file and you are not allowed to change them.
- ■ **lib.v:** Standard cells.
- ■ **tb.v:**
  The testbench for your design.
- ■ **pattern/radicand.dat:**
  Input patterns for the testbench. Please keep the hierarchy when simulation.
- ■ **pattern/golden.dat:**
  Patterns of correct answers for the testbench. Please keep the hierarchy when simulation.

Submission

All students who do not submit files according to the rules **will get a 20% penalty.**

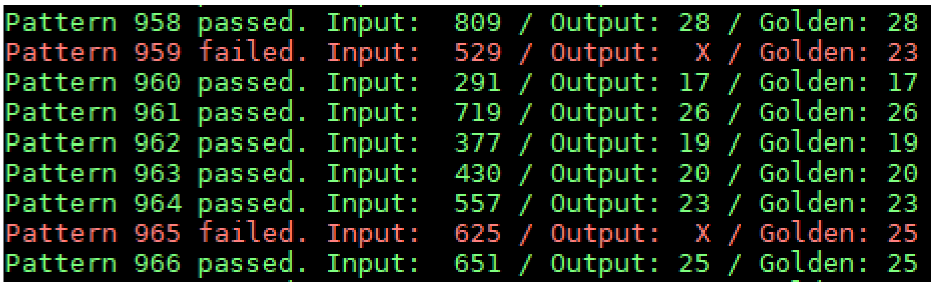
- ➢ You should upload a **zip file** to **CEIBA**, the file name is "HW4_Student ID_vx", vx represents the version you submitted. Ex: HW4_b07901001_v1.zip
- ➢ Your file must conform to the following structure.



Testbench

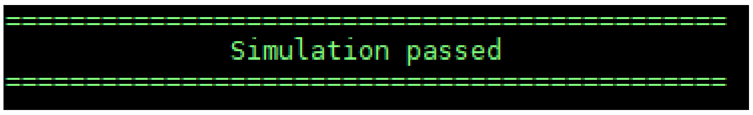
1. *Description*

- The output waveform will be dumped to file "sqrt.fsdb", you can use nWave to examine it.
- You can change the number of test data to debug, but the final score will still test 1000 data. (**`define PATTERN 1000**)
- You can enable the debug function, which will display the data sent and received.



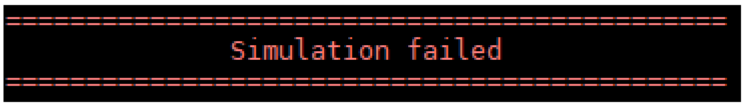
```
Pattern 958 passed. Input:  809 / Output: 28 / Golden: 28
Pattern 959 failed. Input:  529 / Output:  X / Golden: 23
Pattern 960 passed. Input:  291 / Output: 17 / Golden: 17
Pattern 961 passed. Input:  719 / Output: 26 / Golden: 26
Pattern 962 passed. Input:  377 / Output: 19 / Golden: 19
Pattern 963 passed. Input:  430 / Output: 20 / Golden: 20
Pattern 964 passed. Input:  557 / Output: 23 / Golden: 23
Pattern 965 failed. Input:  625 / Output:  X / Golden: 25
Pattern 966 passed. Input:  651 / Output: 25 / Golden: 25
```

- If you passed the simulation, you should see:



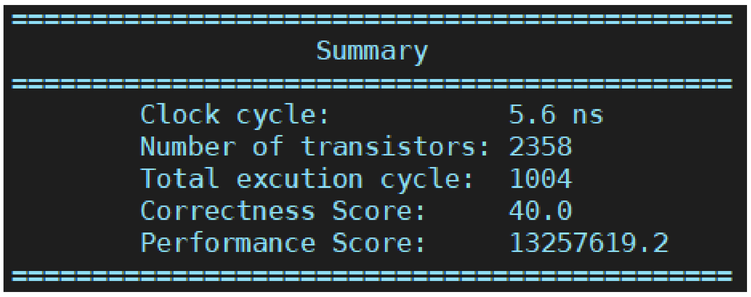
```
============================================
              Simulation passed
============================================
```

Otherwise, you would see:



```
============================================
              Simulation failed
============================================
```

You would also see the summary:



```
============================================
                  Summary
============================================
        Clock cycle:              5.6 ns
        Number of transistors:  2358
        Total excution cycle:   1004
        Correctness Score:      40.0
        Performance Score:      13257619.2
============================================
```

Note that the performance score is the A*T value that would be used in ranking.

2. *Simulation command*

(a) Recursive

> ncverilog tb.v sqrt.v lib.v +access+r

> ncverilog tb.v sqrt.v lib.v +access+r +define+DEBUG

(b) Pipeline

> ncverilog tb.v sqrt.v lib.v +access+r +define+PIPELINE

> ncverilog tb.v sqrt.v lib.v +access+r +define+DEBUG+PIPELINE

TA email: r08943018@ntu.edu.tw, EE2-329

HW4 Office hours: 2021/01/04 (Mon) 19:00~21:00 @BL213
2021/01/07 (Thur) 12:00~14:00 @BL215

If you have no time during office hours, you can email TA to discuss another time for the appointment.