IC Design

Homework # 4

(Due 2021/12/15, 9:00, Verilog code and report upload to NTUCool, no hardcopy)

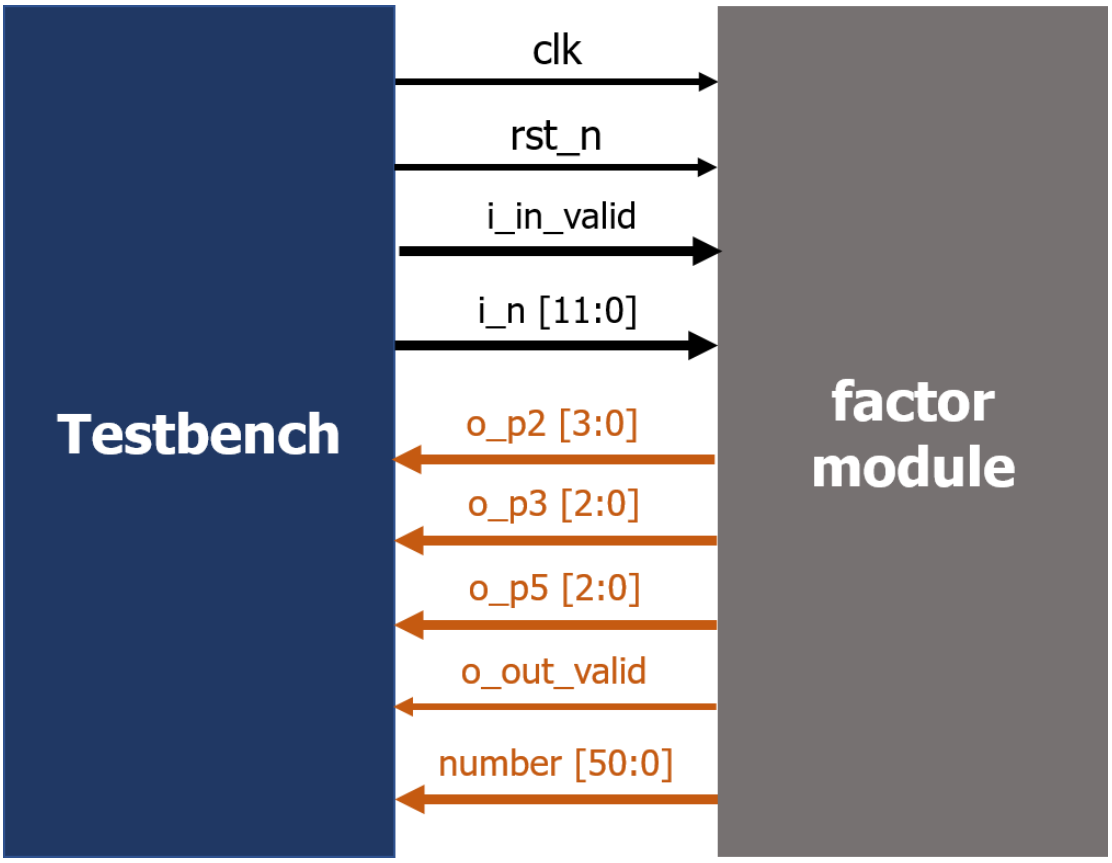
✧ Plagiarism is not allowed. 10% penalty for each day of delay.

Problem Specification

Design a prime factorization circuit with reset that computes the **powers of the prime factors 2, 3, 5 of the target number N, namely P2, P3, and P5**. There are two input signals for the circuit, i.e., i_n with 12 bits, and a 1-bit i_in_valid. The circuit contains four output signals, i.e., 4-bit o_p2 for the power of factor 2, 3-bit o_p3 for the power of factor 3, 3-bit o_p5 for the power of factor 5, and a 1-bit o_out_valid. Note that all the input and output signals are **unsigned integers.** The target N is guaranteed to be **non-zero**. N might contain other prime factors, just ignore them. The relation between the input and the output signals is

$$\mathrm{N} = 2^{P2} \times 3^{P3} \times 5^{P5} \times K.$$

Where K is an integer that satisfies gcd(K,30)=1



Pull up "o_out_valid" and output your answer to "o_p2[3:0]", "o_p3[2:0]," and

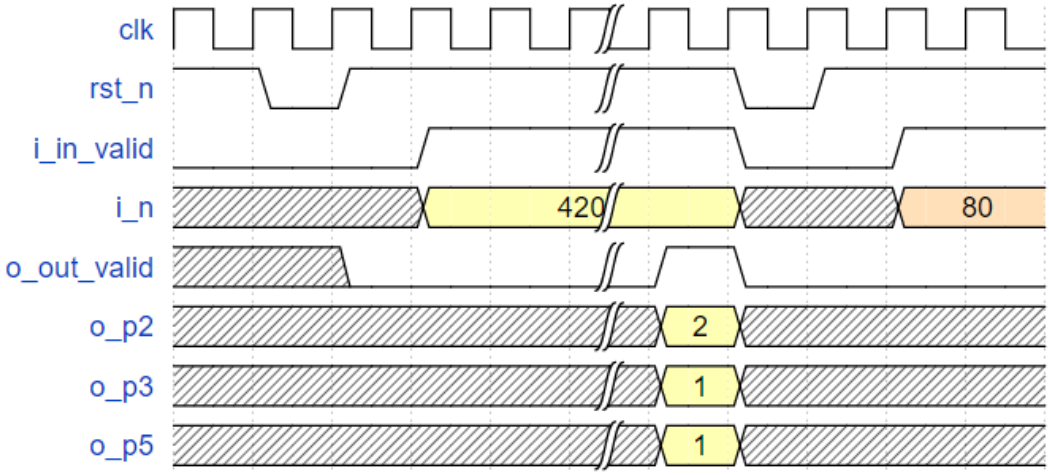
"o_p5[2:0]" once your circuit finishes the calculation, the testbench will then check your answers. Note that the output signals: "o_p2[3:0]", "o_p3[2:0]", "o_p5[2:0]," and "o_out_valid" must be registered, i.e., they are outputs of DFFs **(use module FD2 (positive edge) in lib.v)**.

Timing Diagram

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

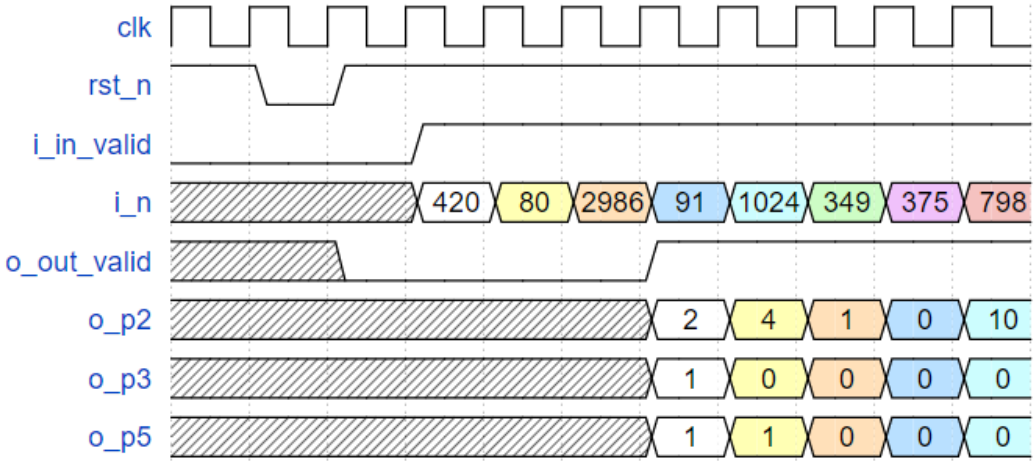
1. Recursive

In the recursive input strategy, **the circuit would be reset before every new input**.



2. Pipeline

In the pipeline input strategy, the circuit would only be reset once. After reset, **new values of i_n would be input at every cycle**. Besides, **o_out_valid should maintain high once it is pulled up**, and the order of input n and their corresponding powers of prime factors 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_in_valid	Input	1	Indicate that the input is valid
i_n	Input	12	Target for Factorization
o_p2	Output	4	Power of 2
o_p3	Output	3	Power of 3
o_p5	Output	3	Power of 5
o_out_valid	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 tb.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 "<".
- Using "assign" to concatenate signals or specify constant values is allowed.
- Design your homework in the given "factor.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 "factor.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 divider 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 div module and connect it to number [50:0]**.

Only in this section, you are allowed to use "+" to help with calculations.

```
14  assign number = gate_count[0] + gate_count[1] + gate_count[2] + gate_count[3];
15
16  DRIVER V1 (.A(pp1_w[9]), .Z(pp1_w[10]), .number(gate_count[0]));
17  DRIVER V2 (.A(pp1_w[9]), .Z(pp1_w[11]), .number(gate_count[1]));
18  DRIVER V3 (.A(pp1_w[9]), .Z(pp1_w[12]), .number(gate_count[2]));
19  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 graded by your ranking according to 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
Incorrect number [50:0] connection	0
Correctness failed	0
Plagiarism	0

2. Report (30%)

(a) Simulation (0%)

Specify your **minimum cycle time** and **which strategy you used**. If you do not provide this information, **You will not get any score** for part 1(70%)”**. This minimum cycle time would be verified by TAs.** Also, please put the screenshot of the summary provided by the testbench in the report.

(b) Circuit diagram (10%)

You are encouraged to use software to draw the 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 pipelined or recursive design?
- (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)

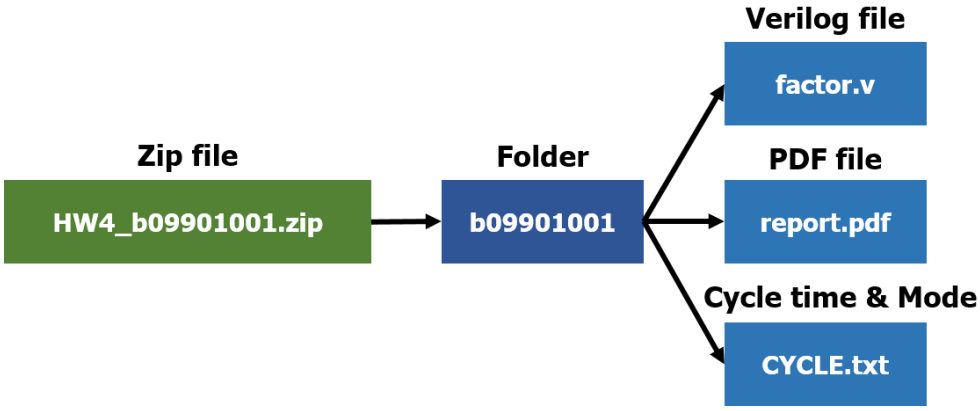
HW4.zip includes

- **HW4_2022.pdf:** This document.
- **HW4_tutorial_2022.pdf:** Tutorial in class.
- **factor.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/Inn.dat:**
  Input patterns for the testbench. Please keep the hierarchy when simulation.
- **pattern/Gol2.dat & Gol3.dat & Gol5.dat:**
  Patterns of the correct answers for the testbench. Please keep the hierarchy when simulation.

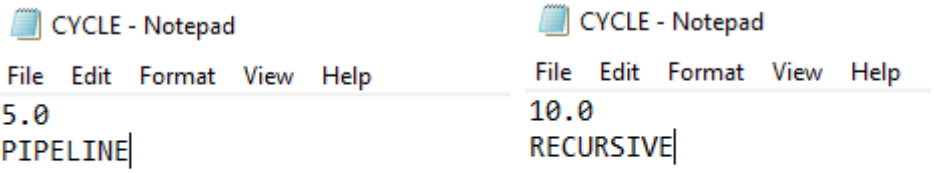
Submission

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

- You should upload a **zip file** to **NTUCool**, the file name is "HW4_Student ID", e.g., HW4_b09901001.zip
- Your file must conform to the following structure.



- Specify your cycle time and which input mode you prefer in CYCLE.txt, all letters are in uppercase, e.g.,



Testbench

1. *Description*

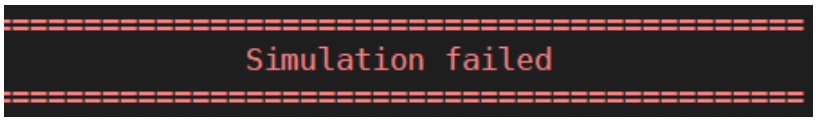
- The output waveform will be dumped to file "factor.fsdb", you can use nWave to examine it.
- You can change the number of test data to debug, but the final score will still test 1000 data. (`**define PATTERN 1000**)
- You can enable the debug function, which will display the data sent and received.

```
Pattern   0 passed. / Input N:  39 / Output P2: 0 / Output P3: 1 / Output P5: 0 / Golden P2: 0 / Golden P3: 1/ Golden P5: 0
Pattern   1 passed. / Input N:2044 / Output P2: 2 / Output P3: 0 / Output P5: 0 / Golden P2: 2 / Golden P3: 0/ Golden P5: 0
Pattern   2 failed. / Input N: 465 / Output P2: 0 / Output P3: 2 / Output P5: 1 / Golden P2: 0 / Golden P3: 1/ Golden P5: 1
Pattern   3 passed. / Input N: 205 / Output P2: 0 / Output P3: 0 / Output P5: 1 / Golden P2: 0 / Golden P3: 0/ Golden P5: 1
Pattern   4 passed. / Input N:2807 / Output P2: 0 / Output P3: 0 / Output P5: 0 / Golden P2: 0 / Golden P3: 0/ Golden P5: 0
Pattern   5 failed. / Input N:1995 / Output P2: 0 / Output P3: 2 / Output P5: 1 / Golden P2: 0 / Golden P3: 1/ Golden P5: 1
Pattern   6 passed. / Input N:3676 / Output P2: 2 / Output P3: 0 / Output P5: 0 / Golden P2: 2 / Golden P3: 0/ Golden P5: 0
Pattern   7 passed. / Input N:2652 / Output P2: 2 / Output P3: 1 / Output P5: 0 / Golden P2: 2 / Golden P3: 1/ Golden P5: 0
Pattern   8 passed. / Input N:3674 / Output P2: 1 / Output P3: 0 / Output P5: 0 / Golden P2: 1 / Golden P3: 0/ Golden P5: 0
Pattern   9 passed. / Input N:2954 / Output P2: 1 / Output P3: 0 / Output P5: 0 / Golden P2: 1 / Golden P3: 0/ Golden P5: 0
Pattern  10 failed. / Input N:3405 / Output P2: 0 / Output P3: 2 / Output P5: 1 / Golden P2: 0 / Golden P3: 1/ Golden P5: 1
Pattern  11 passed. / Input N:3389 / Output P2: 0 / Output P3: 0 / Output P5: 0 / Golden P2: 0 / Golden P3: 0/ Golden P5: 0
Pattern  12 passed. / Input N:3414 / Output P2: 1 / Output P3: 1 / Output P5: 0 / Golden P2: 1 / Golden P3: 1/ Golden P5: 0
```

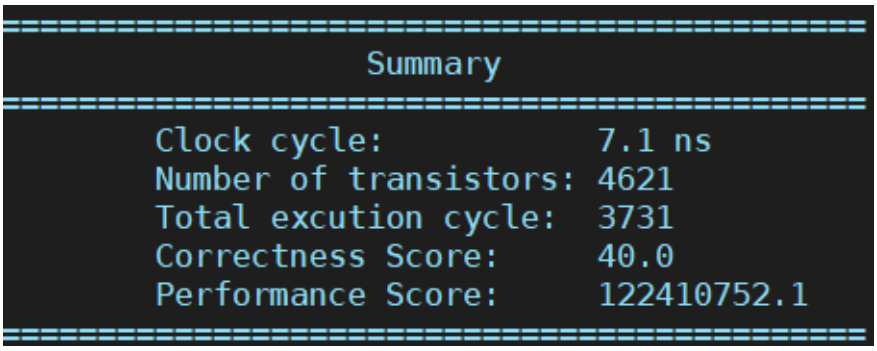
- If you pass the simulation, you should see:

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

Otherwise, you would see:



You would also see the summary:



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

2. Simulation command

(a) Recursive

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

(b) Pipeline

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

TA email: tp62u4m3@gmail.com, EE2-329
HW4 Office hours: 12/13 Tue 14:00~16:00 @EE2-329
12/14 Wed 14:00~16:00 @EE2-329
If you are not convinient during office hours, you can email TA to make an alternative appointment.