

NCTV-EEIC LAB – FALL 2021

Lab08 Exercise

Design: Series Processing (SP)

◆ Data Preparation


1. Extract test data from TA's directory:
`% tar -xvf ~iclabta01/Lab08.tar`
2. The extracted LAB directory contains:
 - a. EXERCISE/
 - b. EXERCISE_wocg/
 - c. PRACTICE/
 - d. JG/

◆ Design Description

In this lab, you should do four things in your design, conversion between Excess-3(XS-3) and decimal, subtract the half of range, cumulation, and find three consecutive numbers which has largest sum. Moreover, you should **implement clock gating to save power**.

The Excess-3(XS-3) code, also known as Stibitz code, is a non-weighted and self-complementary BCD code used to represent the decimal numbers. Excess-3 code was used on some older computers as well as in cash registers and hand-held portable electronic calculators of the 1970s, among other uses. This code has a biased representation. This code plays an important role in arithmetic operations because it resolves deficiencies encountered when we use the 8421 BCD code for adding two decimal digits whose sum is greater than 9. The Excess-3 code can be obtained by adding 3(0011) to each decimal digit then it can be represented by using four bit binary number for each digit. The four-bit XS-3 code for any particular single base-10 digit is its representation in binary notation.

Table is following excess-3 codes for decimal digits :

| Decimal | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|--|------|------|------|------|------|------|------|------|------|------|
| BCD code | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 |
|  + 0011 | | | | | | | | | | |
| Excess-3 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 |

◆ Mode Description

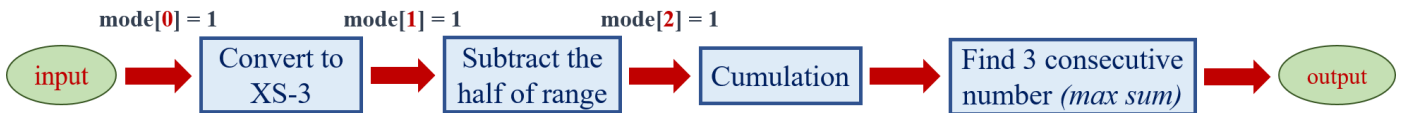
Input signal “in_mode” indicating input configuration and operation to execute.

| | |
|-------------|---|
| in_mode [0] | <p>“1” means input is XS-3 “0” means normal signed decimal number.</p> <p>If in_mode [0] = 1: e.g. 1: input = 0_1010_1100 represents 79 (decimal). e.g. 2: input = 1_0101_1000 represents -25 (decimal).</p> <p>If in_mode [0] = 0: e.g. 1: input = 0_1000_0001 represents 129 (decimal). e.g. 2: input = 1_0000_1101 represents -243 (decimal).</p> <p>◆ Notice :</p> <ul style="list-style-type: none"> Input has 9 bits. input [3:0] and input [7:4] won't exceed 9 if and only if in_mode[0] = 1. |
| in_mode [1] | <p>“1” indicates to subtract the half of range, which means $(\max + \min) / 2$ e.g. 1: original series: 12, 1, -4, 2, 5, -7 → max = 12, min = -7 → half of range = $(12 + (-7)) / 2 = 2$ → After subtract the half of range: 10, -1, -6, 0, 3, -9</p> <p>e.g. 2: original series: 7, 5, 4, 0, -2, -5, -12 → max = 7, min = -12 → half of range = $(7 + (-12)) / 2 = -2$ → After subtract the half of range: 9, 7, 6, 2, 0, -3, -10</p> |
| in_mode [2] | <p>“1” indicates to do cumulation with the rule like moving average. The ratio of old value to new value is 2:1. (round-down the answer if it is not integer)</p> <p>e.g. original series: 5, 3, -7, 0, -10</p> <p>1st number of new series: $(5 * 2 + 5 * 1) / 3 = 5$</p> <p>2nd number of new series: $(5 * 2 + 3 * 1) / 3 = 4$ (round down)</p> <p>3rd number of new series: $(4 * 2 + -7 * 1) / 3 = 0$ (round down)</p> <p>4th number of new series: $(0 * 2 + 0 * 1) / 3 = 0$</p> <p>5th number of new series: $(0 * 2 + -10 * 1) / 3 = -3$ (round down)</p> <p>→ After moving average: 5, 4, 0, 0, -3</p> |

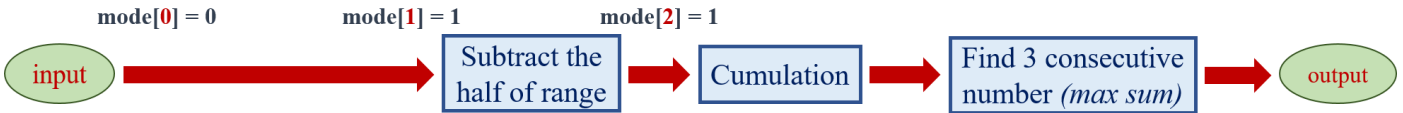
After doing these processes, you have to find the three consecutive number set such that the sum of these three numbers is the maximum among all possible sets, and output these three numbers. If there're more than one possible answer, please output the earlier one. For example: if the sequence is 10, 20, 30, 10, 20, 30, 10, 20, 30, please output the first three consecutive number set, which are 10, 20, 30.

Following are dataflow of some examples.

e.g. **in_mode = 3'b111**



e.g. **in_mode = 3'b110**



e.g. **in_mode = 3'b001**



■ Example :

At the beginning, you will obtain the **in_mode** and **in_data** in series.

in_mode = 3'b111

in_data = 0_1010_1010 → 1_1010_1010 → 1_0101_0100 → 1_1011_1010 →
1_0101_1100 → 0_1010_0101 → 1_1000_0111 → 1_1001_0011 → 0_0110_0111

↓ **in_mode[0] = 1**

Data series after converting to XS-3 : 77, -77, -21, -87, -29, 72, -54, -60, 34

↓ **in_mode[1] = 1**

max = 77, min = -87, half of range = $(77 + (-87)) / 2 = -5$

Data series after doing subtraction : 82, -72, -16, -82, -24, 77, -49, -55, 39

↓ **in_mode[2] = 1**

Data series after doing cumulation : 82, 30, 14, -18, -20, 12, -8, -23, -2



Due to $82+30+14=126$ is the maximum value of all three consecutive sets, output these three numbers.

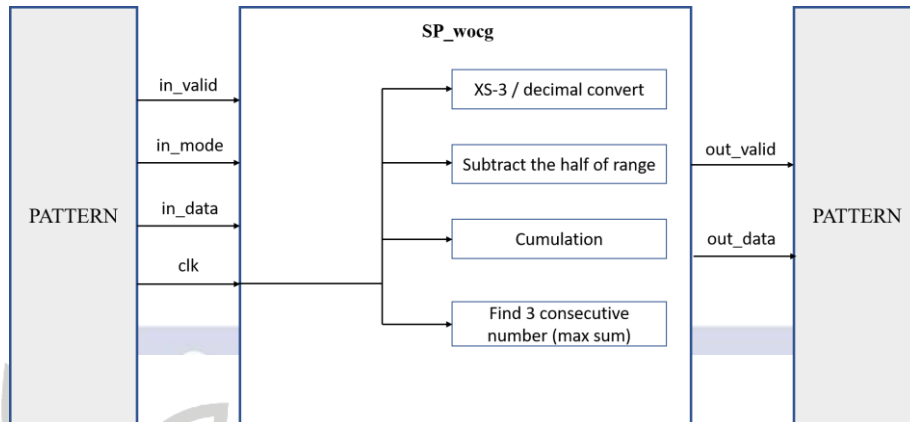
out_data : 82 → 30 → 14

◆ Lab Hint

In this lab, you need to design two version of module.

● Stage 1:

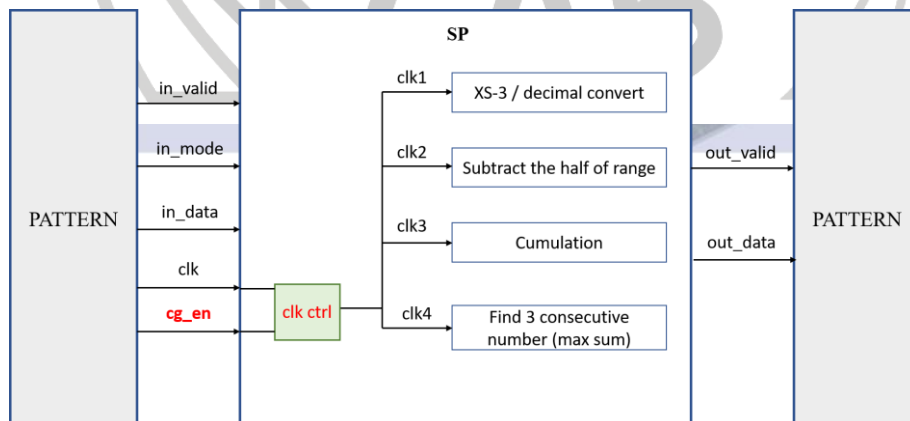
At EXERCISE_wocg/ **SP_wocg.v**, which means SP without clock gating, you should design a SP module as below figure.



- **INPUT** : Receive **number series & mode** from PATTERN
 - ➔ The **in_data [8:0]** will be given in **9 cycles**.
- **SP** : Design module with some submodules perform series processing described as above.
 - ➔ This module should be design without clock gating.
- **OUTPUT** : Output resulting number series

● Stage 2:

At EXERCISE /**SP.v**, you should design a **SP module with clock gating cell** as below figure. Main different part is **clock gating cell** and **cg_en input signal**.



- **INPUT** : Receive **number series, mode and cg_en** from PATTERN
 - ➔ The **in_data [8:0]** will be given in **9 cycles**.
- **SP** : Design module. You should **add clock gating cell** in the design module.
 - If **cg_en** is high, four series processing blocks can perform clock gating; otherwise, if **cg_en** is low, four series processing blocks follow **clk**.
- **OUTPUT** : Output resulting number series

◆ Inputs

| I/O | Signal name | Bit Width | Description |
|-------|-----------------|-----------|---|
| Input | clk | 1 | Clock |
| Input | rst_n | 1 | Asynchronous active low reset |
| Input | cg_en | 1 | If cg_en is high, the series processing blocks should execute clock gating. Otherwise, if cg_en is low, the image processing blocks follow clk . |
| Input | in_valid | 1 | High when input signals are valid. |
| Input | in_data | 9 | in_data is valid when in_valid is high The in_data will be given in 9 cycles. |
| Input | in_mode | 3 | in_mode valid when in_valid is high. in_mode [0] = 1 indicates in_data is XS-3, otherwise signed number. in_mode [1] = 1 indicates subtract the half of range. in_mode [2] = 1 indicates doing cumulation. |

◆ Outputs

| I/O | Signal name | Bit Width | Description |
|--------|------------------|-----------|---|
| output | out_valid | 1 | Should be set to low after reset. Should set to high when your out_data is ready. |
| output | out_data | 10 | Output the resulting number series. The out_data should be given in 3 cycles. |

◆ Specifications

- Top module name : SP (File name: SP.v)
- Input pins : **clk, rst_n, cg_en, in_valid, in_data[8:0], in_mode [2:0]**
Output pins : **out_valid, out_data[9:0]**
- Use **asynchronous** reset active low architecture.
- All your output register should be set zero after reset.
- Changing clock period is prohibited (**fixed at 12ns**).
- The instance name of the clock gating cell (GATED_OR) should be **GATED_XXX** (e.g., GATED_cnt).

```

GATED_OR GATED_out (
    .CLOCK(clk), .SLEEP_CTRL(G_sleep_out),
    .RST_N(rst_n), .CLOCK_GATED(G_clock_out)
);

```
- After synthesis, check the “SP.area” and “SP.timing” in the folder “Report”. The area report is valid only when the slack in the end of “SP.timing” is **non-negative** and the result should be **MET**.

8. The next input will come in **2~5** cycles after your **out_valid** is pulled down.
9. The synthesis result **cannot** contain any **LATCH except for clocking gating latch**.
10. The synthesis result **cannot** contain any error.
11. The output loading is set to 0.05.
12. Input delay and output delay are 0.5*Clock Period.
13. You can't have timing violation in gate-level simulation
14. Your design **can't use memory** in this lab.
15. Your design **can't use DesignWare IP** in this lab.
16. The execution latency is limited in **1000 cycles**.
17. **The total cell area should not be larger than 150,000 μm^2 .**
18. The **out_valid** cannot overlap with **in_valid**.
19. **The redundant cycle is forbidden in this lab.**
20. Your design should have **at least 25% power reduction** from **cg_en-off to cg_en-**

on, otherwise it will be treated as failed. $\frac{P_{cg_enoff} - P_{cg_enon}}{P_{cg_enoff}} \geq 25\%$

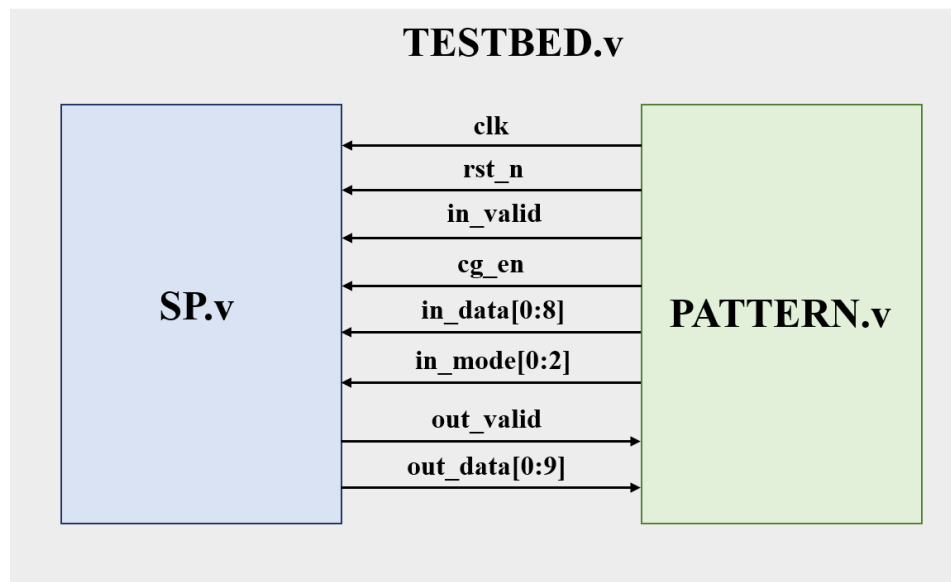
21. **Power report position: 04_PTPX/Report/SP_POWER or SP_CG_POWER**

Report Total Power Example:

| SP_POWER (w/o clock gating) | SP_CG_POWER (with clock gating) |
|---|---|
| <pre> Net Switching Power = 1.135e-03 (22.33%) Cell Internal Power = 3.933e-03 (77.40%) Cell Leakage Power = 1.348e-05 (0.27%) Intrinsic Leakage = 1.348e-05 Gate Leakage = 0.0000 Total Power = 5.082e-03 (100.00%) X Transition Power = 1.015e-05 Glitching Power = 0.0000 Peak Power = 0.0428 Peak Time = 16680 </pre> | <pre> Net Switching Power = 1.013e-03 (28.28%) Cell Internal Power = 2.556e-03 (71.35%) Cell Leakage Power = 1.348e-05 (0.38%) Intrinsic Leakage = 1.348e-05 Gate Leakage = 0.0000 Total Power = 3.583e-03 (100.00%) X Transition Power = 1.015e-05 Glitching Power = 0.0000 Peak Power = 0.0373 Peak Time = 16680 </pre> |

22. The gate level simulation cannot include any timing violations without the *notimingcheck* command
23. Don't use any wire/reg/submodule/parameter name called **error**, **congratulation**, **latch** or **fail** otherwise you will fail the lab. Note: *** means any char in front of or behind the word. e.g: error_note is forbidden.
24. Don't write Chinese comments or other language comments in the file you turned in.
25. Verilog commands *//synopsys dc_script_begin*, *//synopsys dc_script_end* *//synopsys translate_off*, *//synopsys translate_on* are only allowed during the usage of including and setting designware IPs, other design compiler optimizations are forbidden.
26. Using the above commands are allowed, however any error messages during synthesis and simulation, regardless of the result will lead to failure in this lab.

◆ Block Diagram



◆ Grading Policy

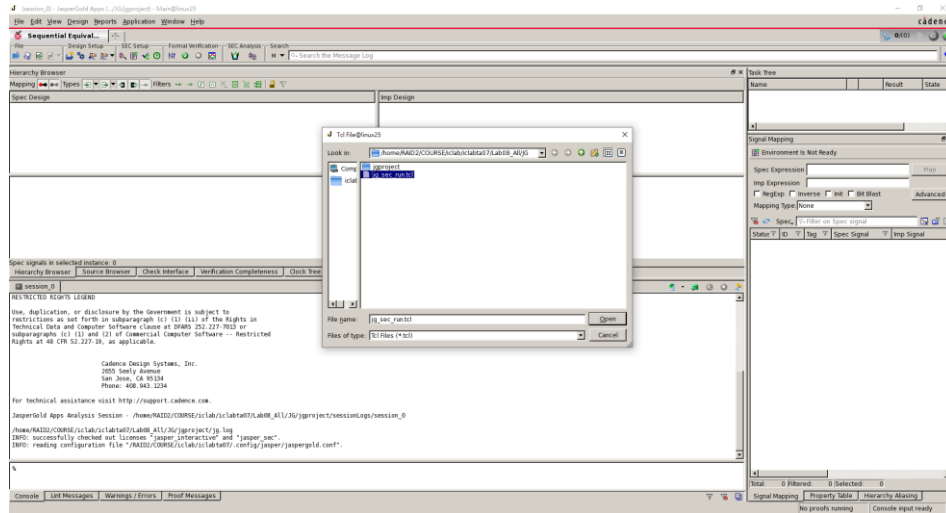
- **Functionality (75%) :**
 - SP_wocg and SP : **70%**
(You can only get these points if you pass the gate level simulation **both**)
 - JasperGold SEC check (5%) : **Run1 (2.5%) , Run2 (2.5%)**
(You can only get these points if you pass the gate level simulation. **No 2nd demo chance.**)
- **Performance (25%) :**
(Total Latency * Total Power (gated with CG)) * Area

◆ Note

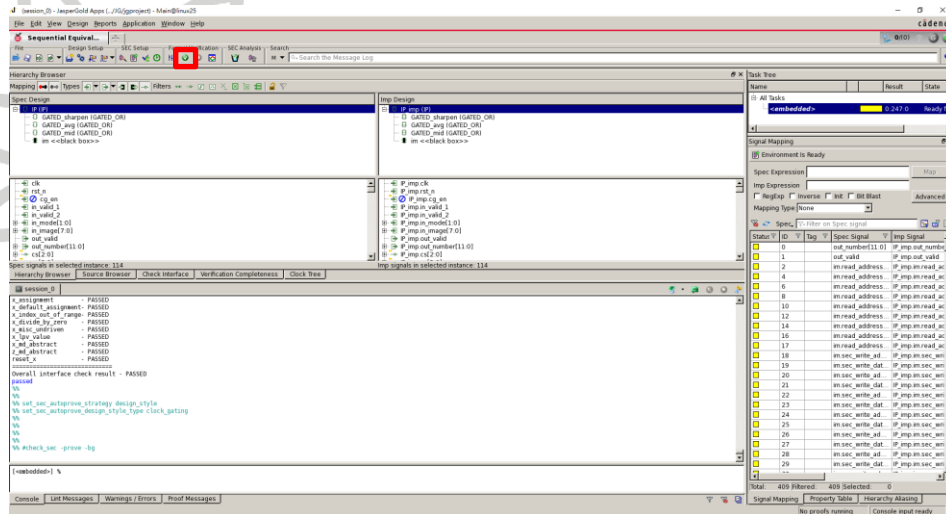
1. Please upload the following file on NewE3 platform before **12:00 at noon** on **Nov. 22**
RTL design : **SP_wocg_iclabXXX.v, SP_iclabXXX.v** (**XXX** is your account no.)
 2. Template folders and reference commands:
 - 01_RTL/** (RTL simulation)
"./01_run", ".02_run_cg"(only in EXERCISE folder)
 - 02_SYN/** (Synthesis)
"./01_run_dc"
(Check the design which contains **latch and error** or not in **syn.log**)
 - 03_GATE_SIM/** (Gate Level simulation)
"./01_run", ".02_run_cg"(only in EXERCISE folder)
 - 04_PTPX/**
"./01_run_ptpx" & ".02_run_cg_ptpx"to get the power of your design.
- You can key in **./09_clean_up** to clear all log files and dump files in each folder

3. JasperGold SEC execution steps: In folder JG/

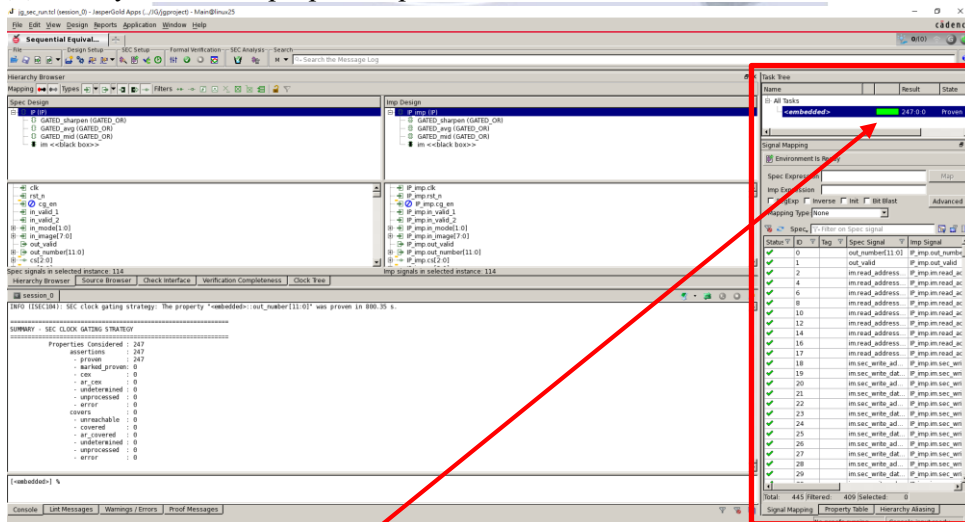
(1) **"./01_run"**、**"./02_run"** or **"jg -sec &"** and click the "source" button.



(2) Then, click the "Prove the SEC Properties"



(3) Finally, check all properties pass.



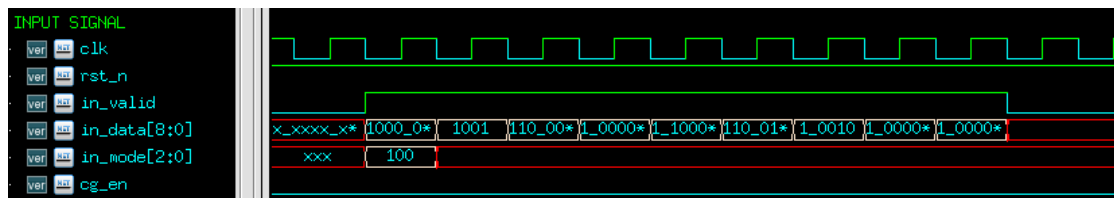
You should pass all tasks in both **"./01_run"**、**"./02_run"**.

◆ Waveform Example

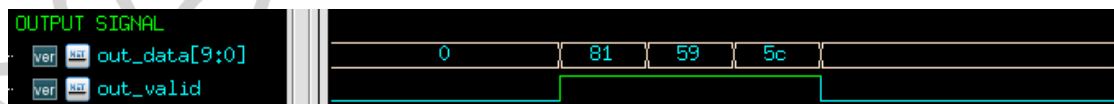
1. Reset Signal



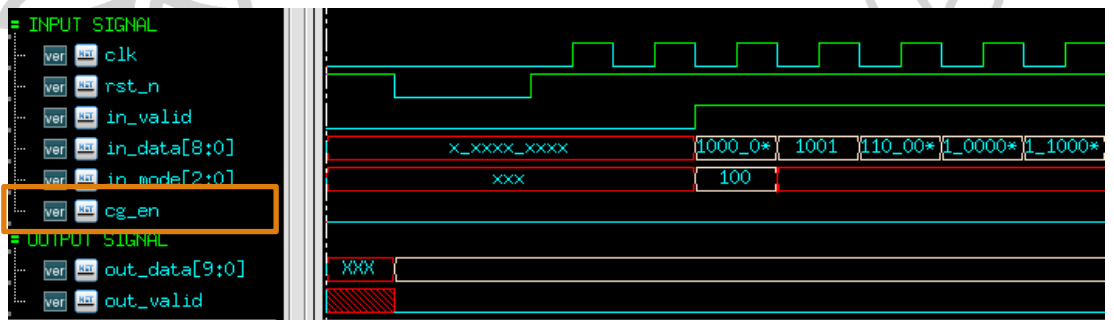
2. Input of in_mode, in_data and cg_en



3. Output



4. For PATTERN.v, you need to set **cg_en = 0** :



5. For PATTERN_CG.v, you need to set **cg_en = 1** :

