

# NM6008 Digital IC Design Test

Huanjia Liu

G2103932C

August 1, 2022

# 1 Introduction

This is the final project, and we are asked to design a character counter chip. There are 7 pins on this chip. 4 inputs: clock signal CLK, reset signal rst, 5 bits character signal chr and 1 bit input signal in. 3 output: count signal cnt, count finish signal oval, input ready signal irdy.

We will accept string through "in" 1 bit each clock. Every time we read 5 bits, comparing with character signal, if same add one to count signal, if not, just go on, until we read all the things.

## 2 Verilog module

Since verilog code is harder than brief flow chart, I use the flow chart below to describe my design. The detail verilog file character\_counter.v was uploaded to one drive folder, readers can access from there.

At the start state, the irdy will turn on to 1, told the user chip is ready to accept the input. The chip will accept 1 bit data each time and save the data into registers. When we saved 5 bits data, the data will be compared with 5 bits "chr" signal. If they are same, count signal "cnt" add one, clear the register and begin to read another 5 bits data. If they are different, just clear the register and begin to read another 5 bit data directly.

As for count finishing, there are three cases. First, the program ends naturally. When we finish reading strings, there is no input signal anymore, it will turn off to 0. Thus, after we get 0 in five clock, that means processing is finish. Then turn on the 'oval' and 'irdy' signal to 1. The second is when we count 15 times. Since the width of our "cnt" signal is only four bits. So if count equal to 15, we turn on the 'oval' and 'irdy' to 1. I have also set up a security measures here. When "oval" turning on, all user input data is ignored at this time. This prevents the user from entering some other data by mistake after the count is over to change the real count number. Users can simply reset to start a new count.

The third happens when we reset the chip. Be careful, at this time, only 'irdy' is 1, 'oval' is 0 because we haven't really finished counting, so the "cnt" is not available at this point.

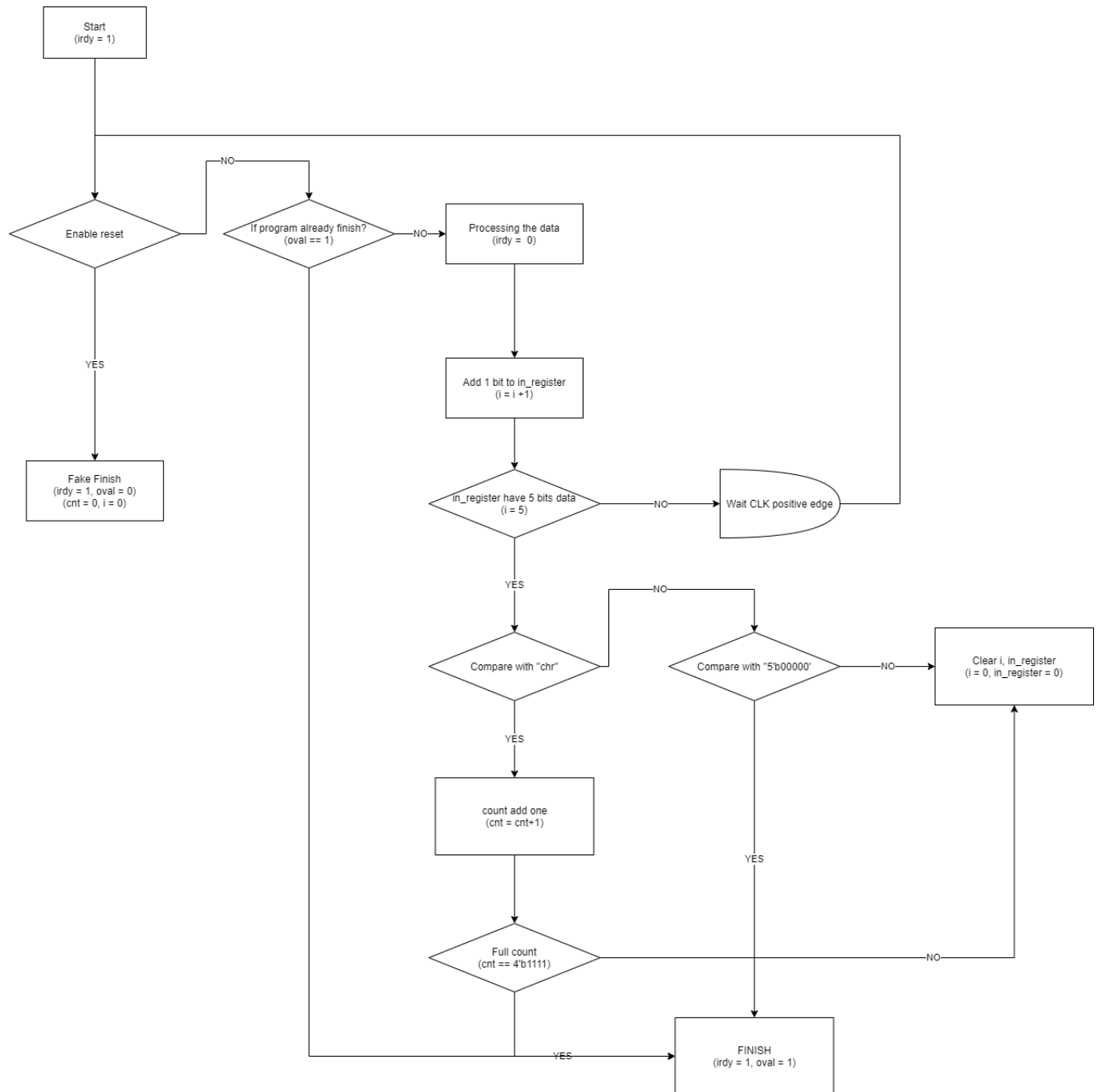


Figure 1: Flow chart

### 3 Test bench and simulation wave

In test bench I input my string of my name (LIU\_HUAN\_JIA) into chip and count "U". The input data signal will ready 1/4 period before positive edge of clock signal and keep 1 period. The detail code called 'tb.v' was uploaded to one drive folder, readers can access from there.

As for the character coding. I set A to "00010" (2) order to Z "11010" (27). "\_" and "." are set as "11100"(28) and "11101"(29) separately. Thus my name can be encoded into "01100|01001|10101|11011|01000|10101|00001|01110|11011|01010|01001|00001" (LIU\_HUAN\_JIA), the character "U" can be encoded into "10101". The figure below is my simulation wave.

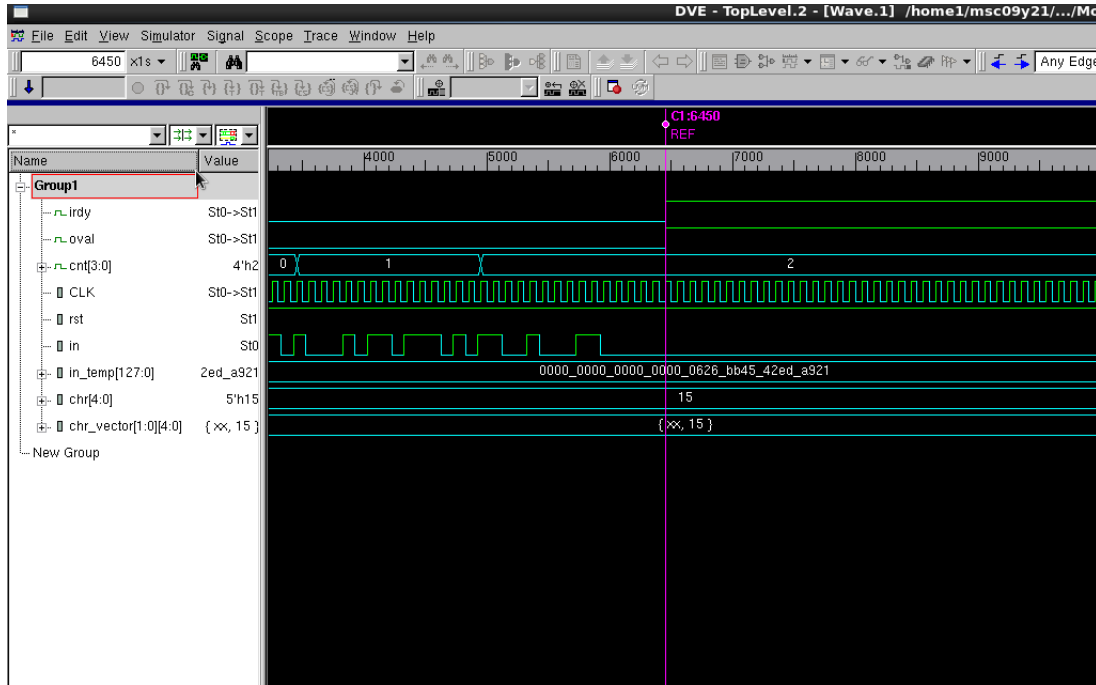
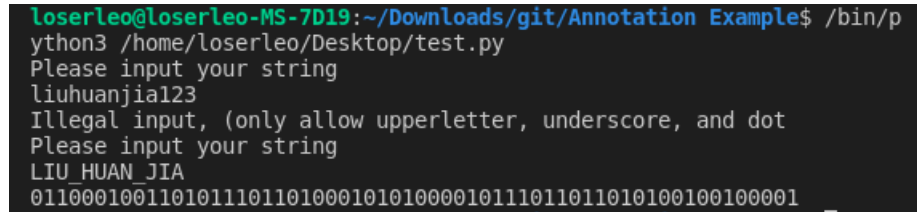


Figure 2: Final simulation wave

We got two "U" here, and after 6450 we got 0 in five clock signal, thus we finished program correctly and turn on the "oval" and "iridy" signal to 1. Everything is as expected. I also attached the detail figure in one drive folder.

## 4 Encode script

For convenient, I write a simple python script here to help me encode my string, just follow the instruction inside, and we can get the a encoded string and encoded character. The detail code is also in one drive folder.



```
loserleo@loserleo-MS-7D19:~/Downloads/git/Annotation Example$ /bin/python3 /home/loserleo/Desktop/test.py
Please input your string
liuhuanjial23
Illegal input, (only allow upperletter, underscore, and dot)
Please input your string
LIU_HUAN_JIA
011000100110101110110100010101000010111011011010100100100001
```

Figure 3: Python script

---

```
final_list = []

judge = 1

while(judge == 1):

    print("Please input your string")

    my_string = input()

    for i in my_string:

        number = ord(i) - 64

        if(number >=1 and number <= 26): #uppercase letter

            final_list.append(number)

        elif(number == (ord("_") - 64)): #underscore

            final_list.append(27)

        elif(number == (ord(".") - 64)): #dot

            final_list.append(28)

        else:

            final_list = []

            break

    if(len(final_list) == 0):
```

```

print("Illegal input, (only allow upperletter, underscore, and dot")

continue

else:

    final_string = ''

    for i in final_list:

        final_string += format(i,'05b') #5 bits output

    print(final_string)

    judge = 0

```

---

## 5 Synthesis

After running the Synthesis simulation, here are information we get from report.rpt.

As for registers, there are 43 ungated registers. The total power consumption is 0.1192 mW. Including internal power 2.8381e-2 mW, switching power 6.1212e-3 mW, and leakage power 8.4681e-2 mW. What's more, my design uses around 4923.359843 cell area.

In my design, there are 20 different kinds of gates from library CORE65GPSVT, I list the name and count below.

Name	Number	Name	Number
HS65_GSS_XOR2X3	1	HS65_GS_NAND4ABX3	6
HS65_GSS_XOR2X6	8	HS65_GS_NOR2AX3	1
HS65_GS_AND2X4	1	HS65_GS_NOR2X6	3
HS65_GS_AO22X4	28	HS65_GS_NOR3X4	5
HS65_GS_AOI13X2	1	HS65_GS_NOR4ABX2	4
HS65_GS_AOI22X1	2	HS65_GS_NOR4X4	2
HS65_GS_BFX2	1	HS65_GS_OAI12X2	3
HS65_GS_BFX9	1921	HS65_GS_OAI21X2	1
HS65_GS_DFPQNX9	3	HS65_GS_OAI21X3	2
HS65_GS_DFPQX9	2	HS65_GS_OAI21X18	1
HS65_GS_DFPQNX9	7	HS65_GS_OAI22X1	2
HS65_GS_DFPQX9	31	HS65_GS_OAI22X6	10
HS65_GS_HA1X4	30	HS65_GS_OAI31X5	1
HS65_GS_IVX9	27	HS65_GS_OAI212X5	1
HS65_GS_NAND2X7	4	HS65_GS_OR3X9	1
HS65_GS_NAND3X2	1	HS65_GS_OR4X4	7
HS65_GS_NAND3X5	5		