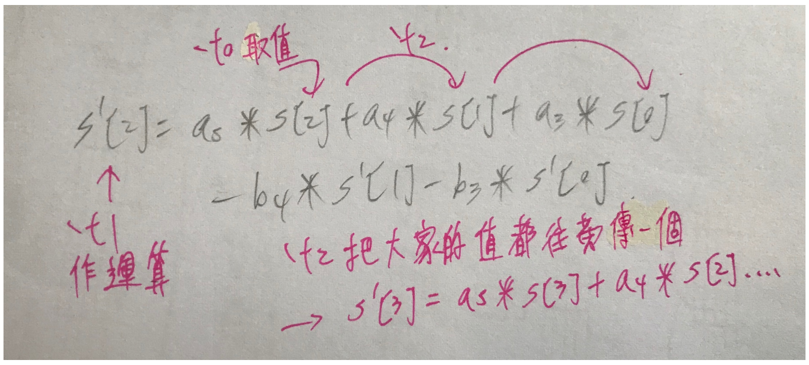
Lab5 Report

黃采瑩 106072237 Econ

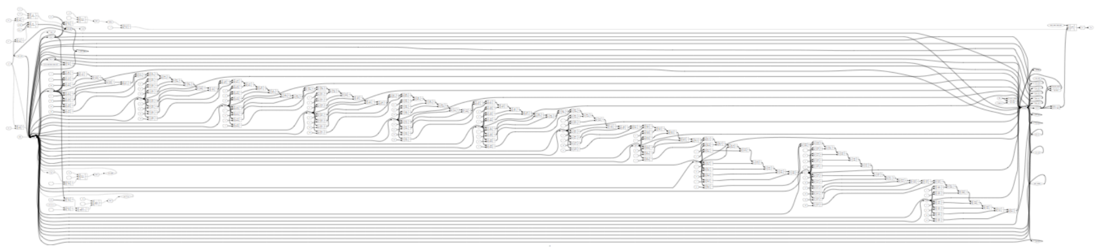
State Transition Graph


`t0 取值
`t2.
s'[2] = a5 * s[2] + a4 * s[1] + a3 * s[0]
- b4 * s'[1] - b3 * s'[0]
`t1 作運算
`t2 把大家的值都往前傳一個
→ s'[3] = a5 * s[3] + a4 * s[2]....

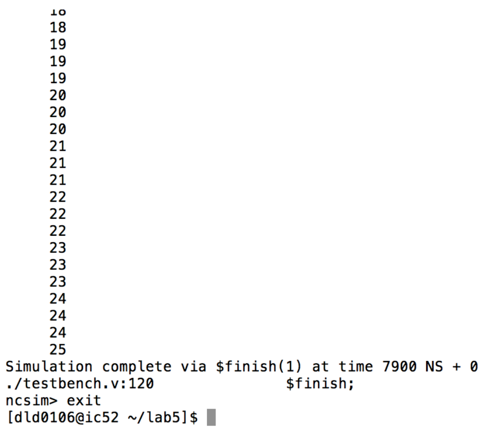
Explanation

I wrote for states, which are `IDLE, `t0, `t1, `t2. `IDLE receive rst and begin the calculation, `t0 receive the data input and do the signed extension, `t1 sums all the shift together, `t2 tranfer values to the regs, prepare for the next calculation. I wrote state transition and value calculation in different always block, in order to test if it is really more easily to debug, turns out the bugs weren't separately, so I didn't feel it easier to debug, I guess when the time I had to write longer and more complicate code, this coding style will benefits me a lot.

Block Diagram



Ncverilog Simulation Result

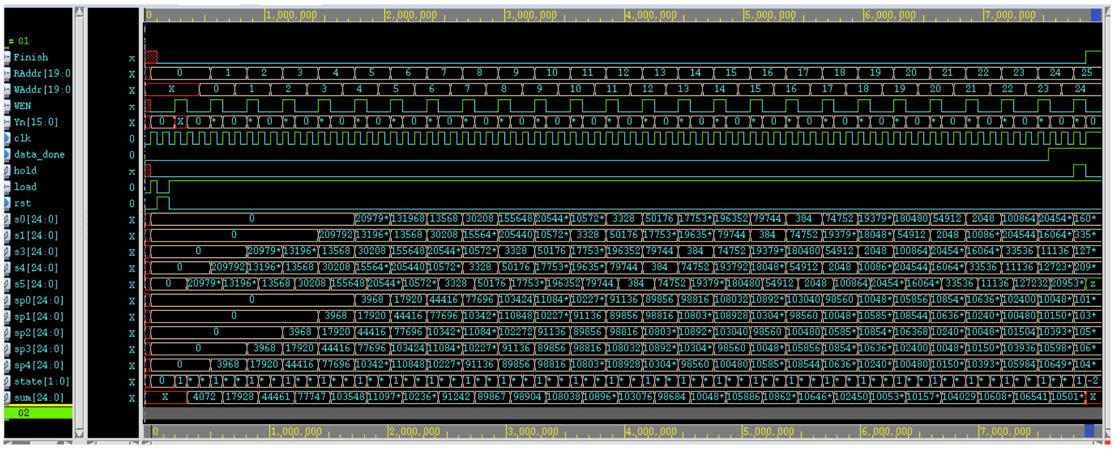


```
        18
        18
        19
        19
        19
        20
        20
        20
        21
        21
        21
        22
        22
        22
        23
        23
        23
        24
        24
        24
        25
Simulation complete via $finish(1) at time 7900 NS + 0
./testbench.v:120                $finish;
ncsim> exit
[dld0106@ic52 ~/lab5]$ 
```

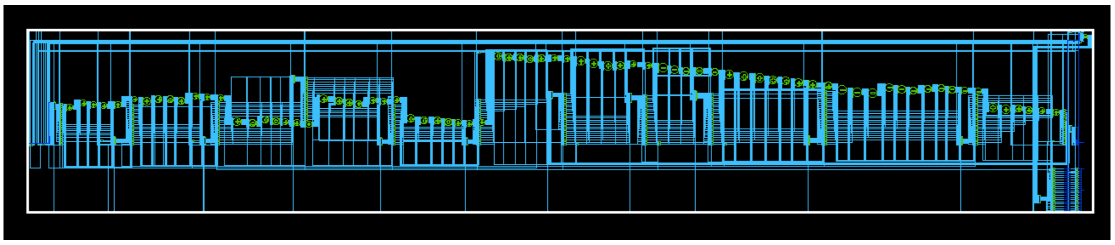
Ncverilog Synthesis Result

```
        18
        18
        19
        19
        19
        20
        20
        20
        21
        21
        21
        22
        22
        22
        23
        23
        23
        24
        24
        24
        25
Simulation complete via $finish(1) at time 7900 NS + 0
./testbench.v:120               $finish;
ncsim> exit
[dld0106@ic52 ~/lab5]$ 
```

nWave Graph



Schematic Graph



Make Syn Results

```
[dld0106@ic52 ~/lab5]$ python check.py 0
Congratulations
[dld0106@ic52 ~/lab5]$ python check.py 1
Congratulations
[dld0106@ic52 ~/lab5]$ python check.py 2
Congratulations
[dld0106@ic52 ~/lab5]$ python check.py 3
Congratulations
[dld0106@ic52 ~/lab5]$ 
```

Acquisition Through the Writing Processes

1. It is extremely important to understand the topic before start writing the code. I didn’t realize what the topic wants at the very beginning, so I encountered many difficulties when writing the code. After the discussion with my classmates, I understood what I had to do, thus made everything much more smoothly.
2. It’s fun to discuss Verilog with others, I’ve learned a lot from it.
3. I learned a cool way to type code with my dearest classmate. That is to press command on the mac keyboard and press D at the same time, so you can type same thing simultaneously on continually different lines of code. For example, to type signed after so many regs. This is truly a convenient function.

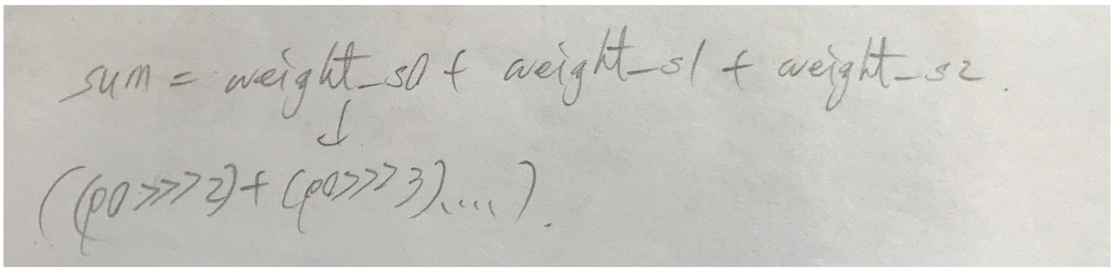
Problems Encountered and Solution

1. I wrote rst as reset, and made many other similar silly mistakes.
   A: To read the topic, input and output name more carefully.
2. I didn’t set the regs needed for calculation to be signed, so they couldn’t be implemented as arithmetic shift.
   A: I set all the regs needed for calculation to be signed.
3. The Yn isn’t the same with test.dat I got after make TEST, though nWave shows the right answers with the value of Yn.
   A: I found the problem is due to the code about Yn, I initially write it as Yn = {{sum[24]}, {sum[21:7]}, but after I change it to Yn = (WEN == 1’b1) ? {{sum[24]}, {sum[21:7]} : 16’b0, It printed most of the right value, but I also found the other problem, which is the one discussed below.
4. After I successively made the ans.dat almost the same with Yn, I found out that the first result didn’t write in it, which made me had a severe headache.
   A: After checking the nWave, me and my friend found the problem is due to WEN. Because I set WEN to be 1’b1 at `IDLE and hasn’t put it down ever since then, so the value would be written in every time the clock posedge, so I change the writing from WEN = (rst == 1’b0) ? 1’b1 : 1’b0 in the always block to assign WEN = (state == `s0) ? 1’b1 : 1’b0, and the value would be written only once after the calculation.
5. Finish rises too quick that it stops printing out values even the last value hadn’t be printed out yet.
   A: I wrote a hold, and set the hold to be finish when data_down == 1’b1, this delay finish a cycle to rise, so all of the values can be printed out successfully.

6. I don’t know how to save report_area and report_clk so I messed up the synthesis file and got wrong answers when making syn.
   A: I read the Hackmd again and made everything right.

Questions Desire for Answers

1. I initially write the sum separately, like the thing I write in the below graph, I set regs to store shift of every signal, and calculate them ultimately, but the answer is never right, so I cange it to the way it is now, I just wonder why the way I tried before doesn’t work.



sum = weight_s0 + weight_s1 + weight_s2.

↓

((p0>>>2)+(p0>>>3)....).