4-bits Multiplier Design based on VHDL

Ziyu Tian

1: Parallel Multiplier

> The full codes of parallel multiplier could be viewed on https://edaplayground.com/x/KWM6.

1.1: Design process

At the RTL level design, I designed the components AND gate and XOR gate, and constructed AND and XOR component to a half-adder in structural level. Using the half-adder, the full_adder can be built shown as Fig.1.

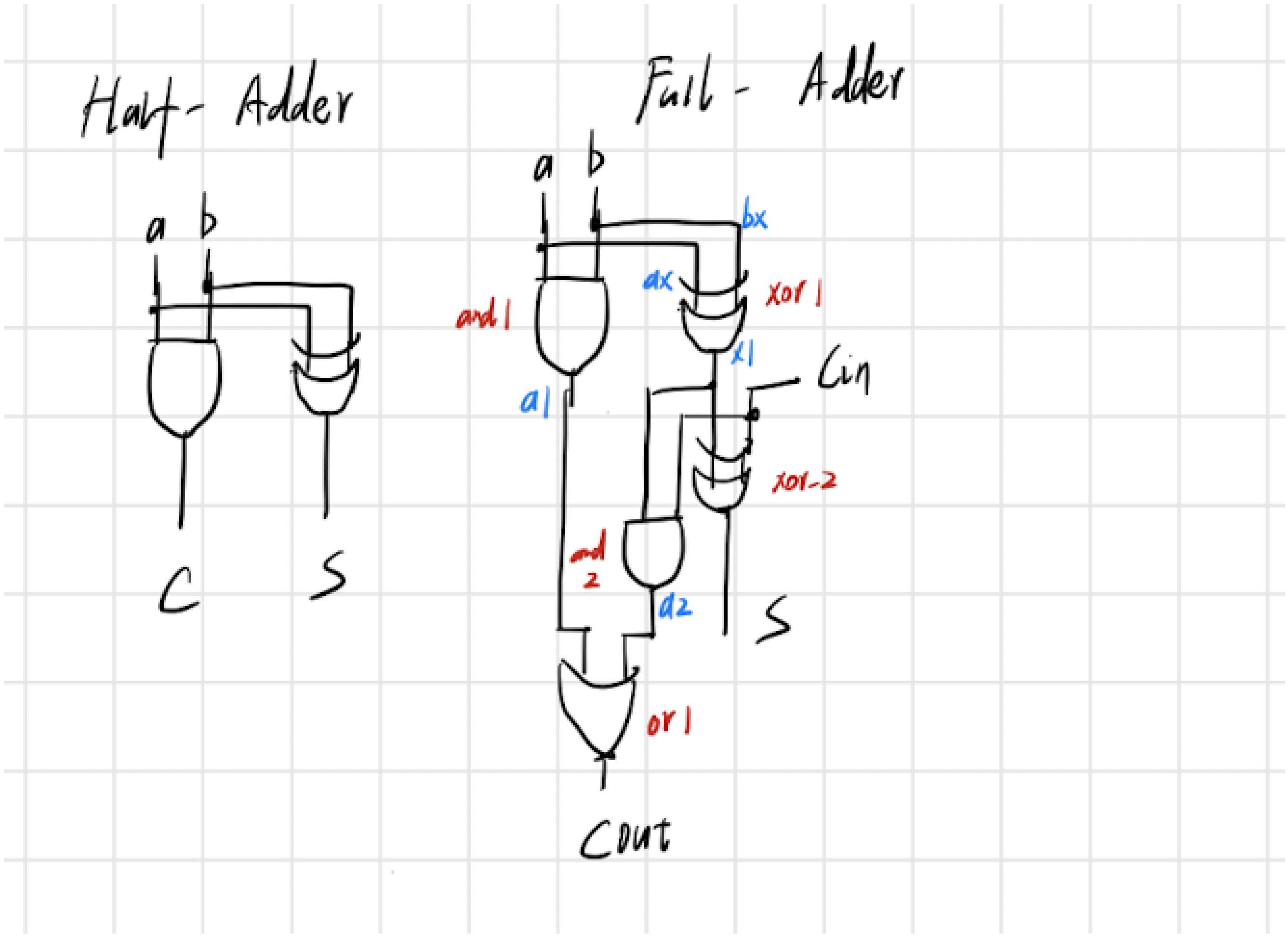
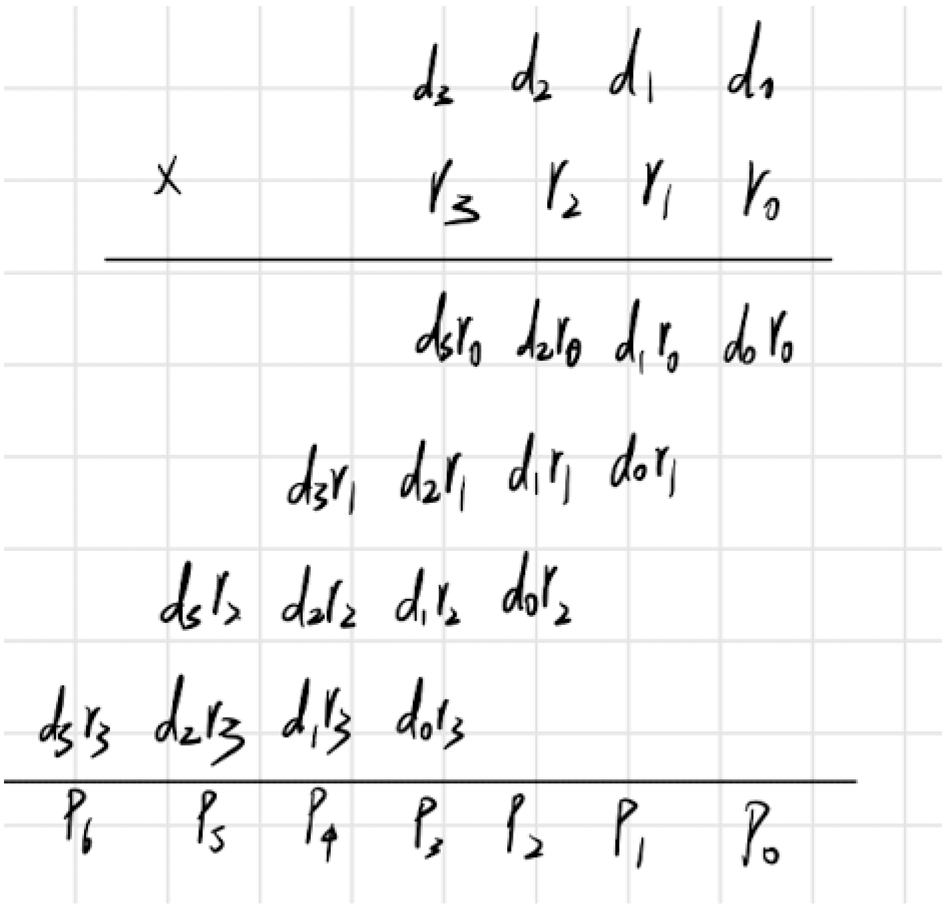


Fig.1 HA and FA design

If we assumed the multiplier and multiplicand expressed as Fig.2, the carry-save parallel multiplier structure is designed as Fig.3:



d_3 d_2 d_1 d_0
× r_3 r_2 r_1 r_0
d_3r_0 d_2r_0 d_1r_0 d_0r_0
d_3r_1 d_2r_1 d_1r_1 d_0r_1
d_3r_2 d_2r_2 d_1r_2 d_0r_2
d_3r_3 d_2r_3 d_1r_3 d_0r_3
P_6 P_5 P_4 P_3 P_2 P_1 P_0

Fig.2 Assumed Expression of multiplier and multiplicand

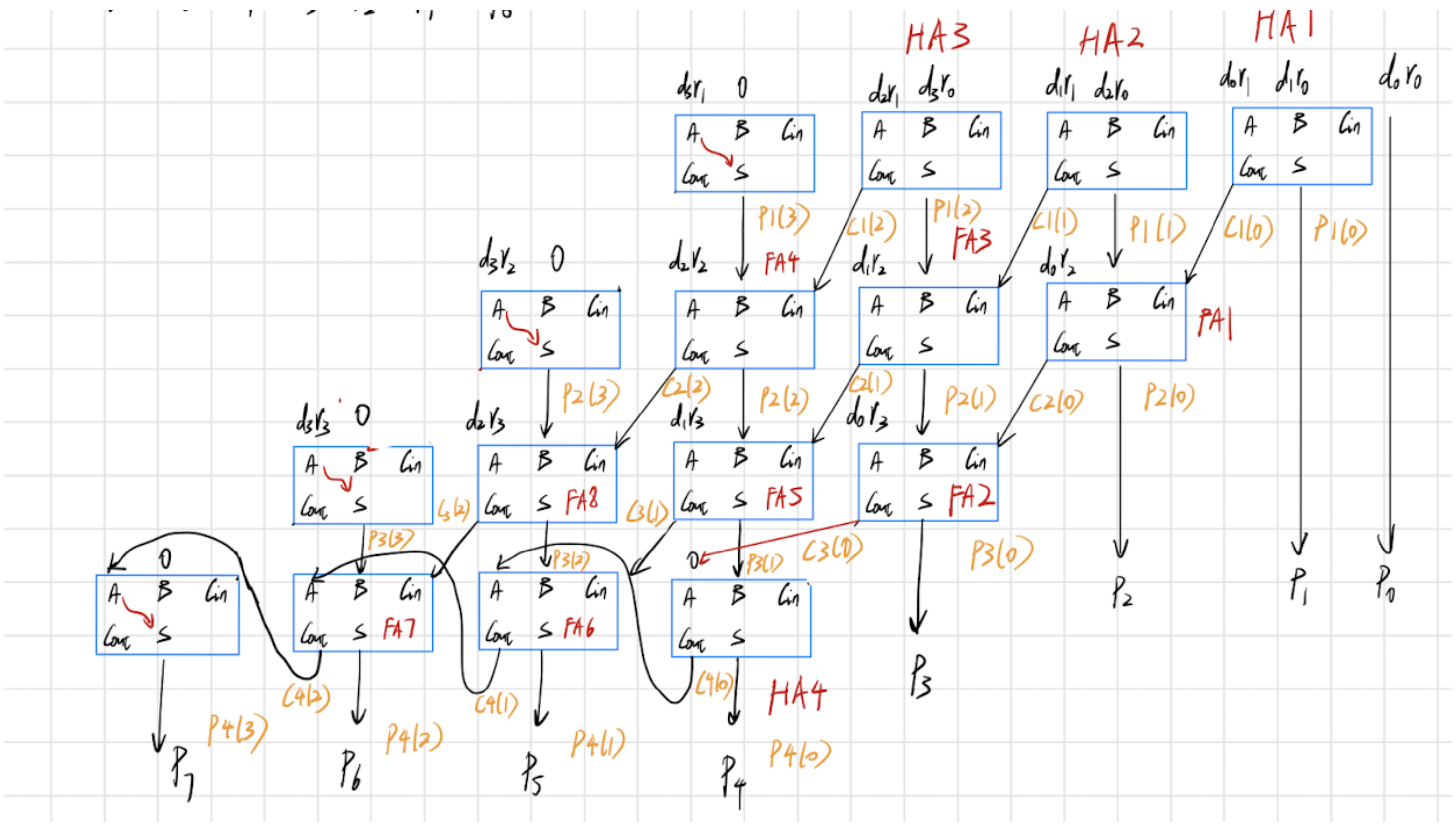


Fig.3 Parallel Multiplier Structure

1.2: Testbench design process

- In the parallelM_tb.vhd file, I tested the product of 0110 x 1100, 0010 x 0001, 0111 x 0101, 0100 x 1101, 1010 x 1110 and 1011 x 1011 with the delay of 100 ns.
- The result proved that the parallel multiplier worked properly.

2: Serial Multiplier

2.1: Design process

> The full codes of parallel multiplier could be viewed on https://edaplayground.com/x/JawR

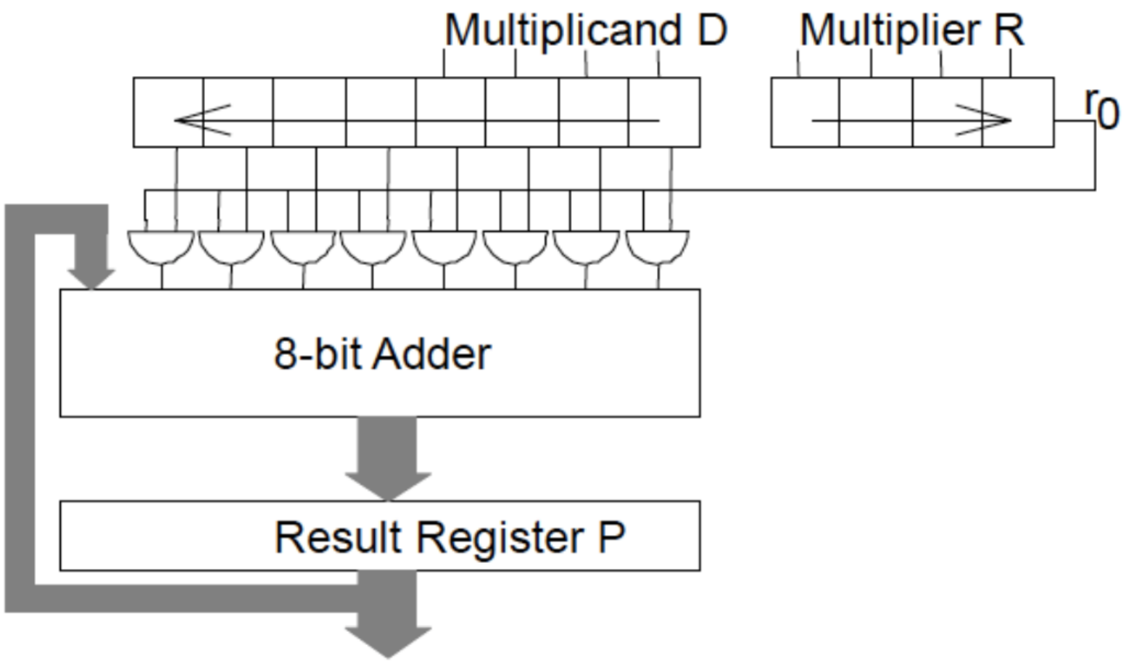


Fig.4.1. Serial Multiplier Structure

To design a 4-bits serial multiplier shown as Fig.4.1, the full-structure should be:

- Reg_1: the entity of the multiplier R register (Fig.4.2). CLK can be used to generate the clock signal and LOAD is used to loading (initialize) the multiplier value. r_1 is the 4-bits input with one output to pass the value that been shifted.

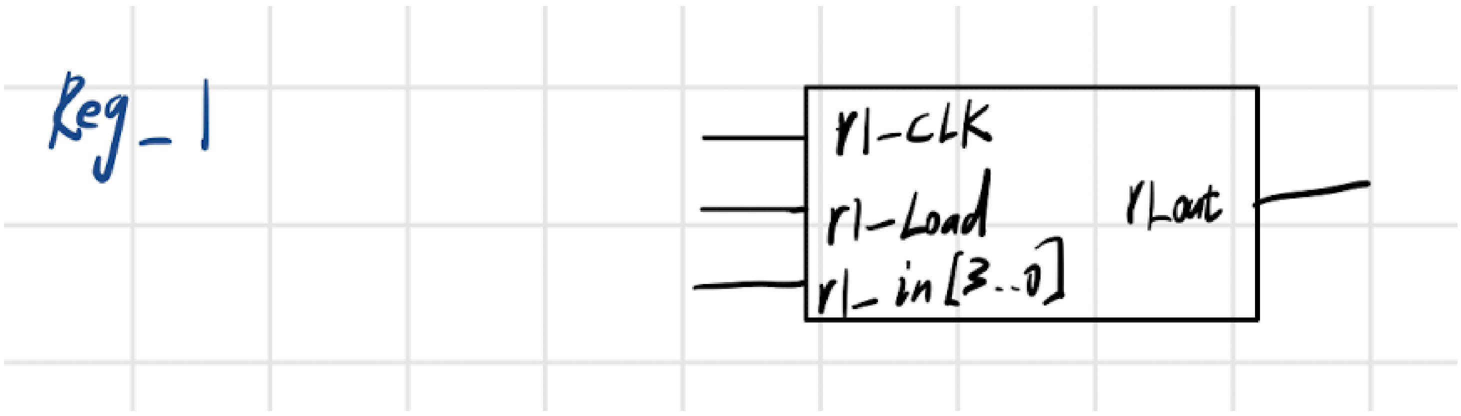


Fig.4.2. Reg_1

- singleMultiplier: the entity doing 1 bit x 4 bits multiplication. m1X receive one bit input of r and the multiplicand D is received by m1Y. The output is the result of the multiplication.

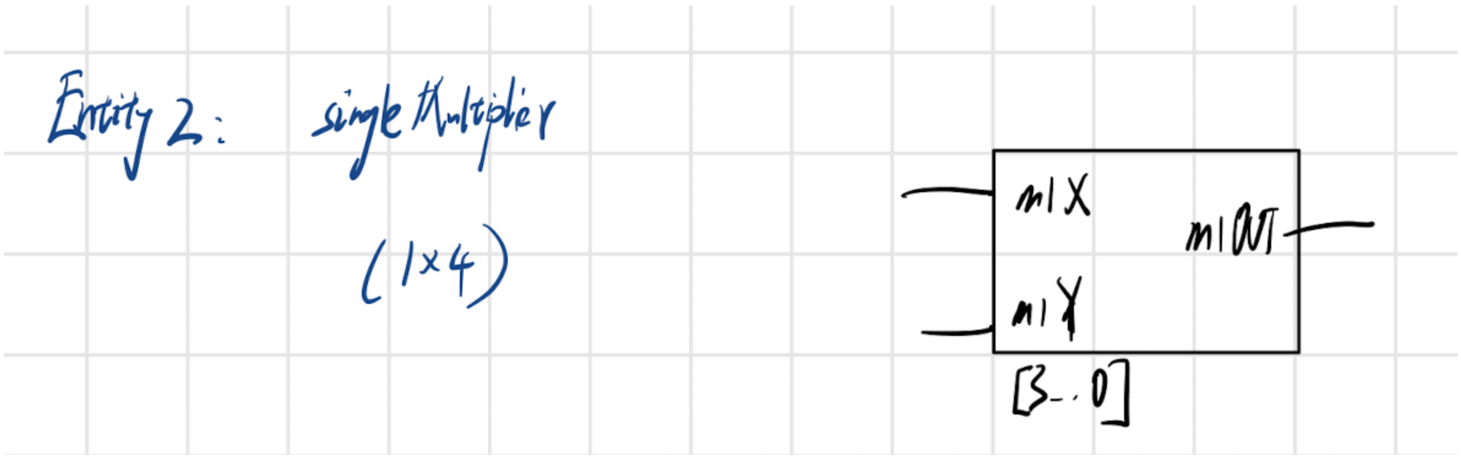


Fig.4.3. singleMultiplier

- Adder4 : the 4-bits adder composed of 4 full_adders, accept two 4-bits number and output the sum.

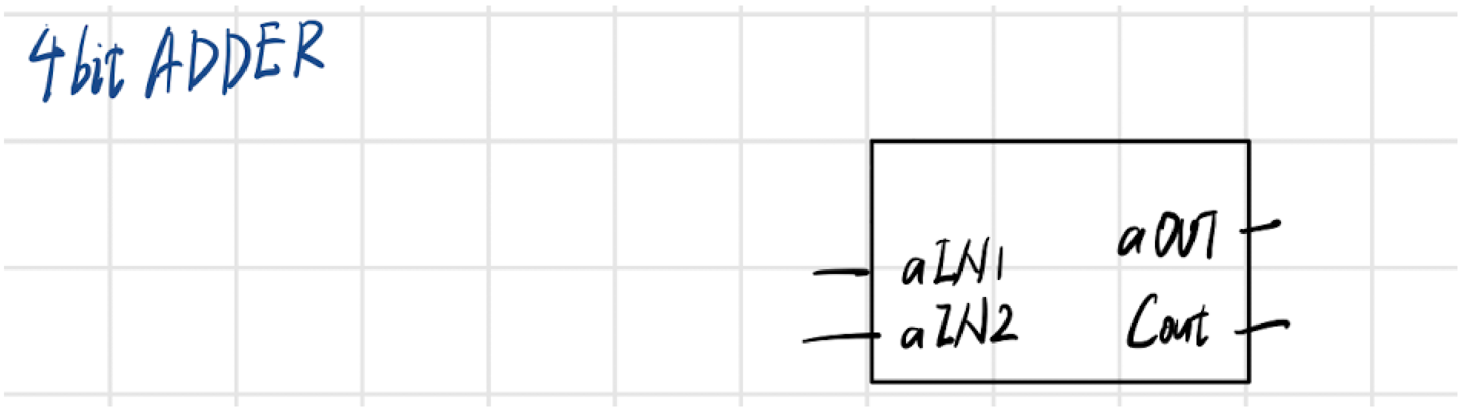


Fig.4.4. 4-bits adder

- Reg_2: receive a CLK signal to control the sequence, a CLR signal to clear the initial signal to '0000'. The output is a 8-bits number, which is the final result.

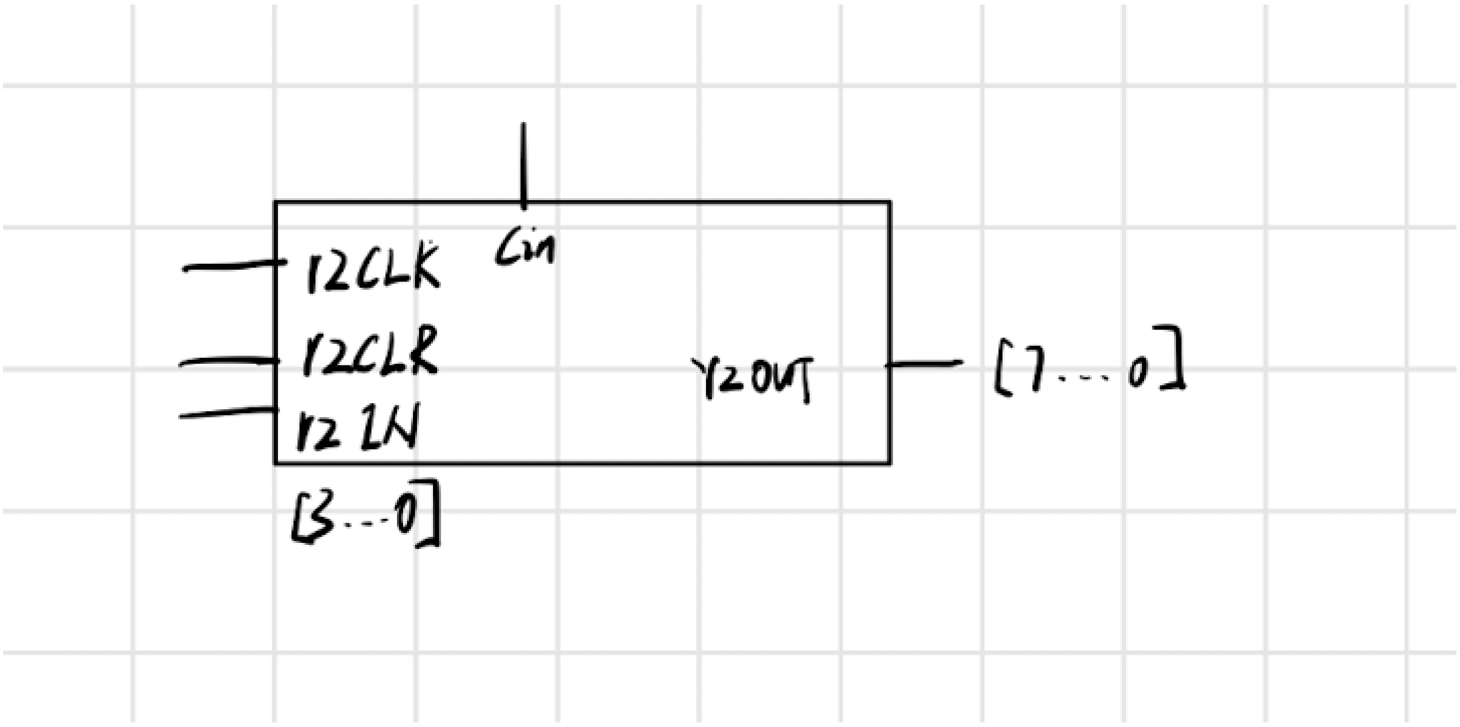


Fig.4.5. Reg_2

Using the components above, the full structure is shown below:

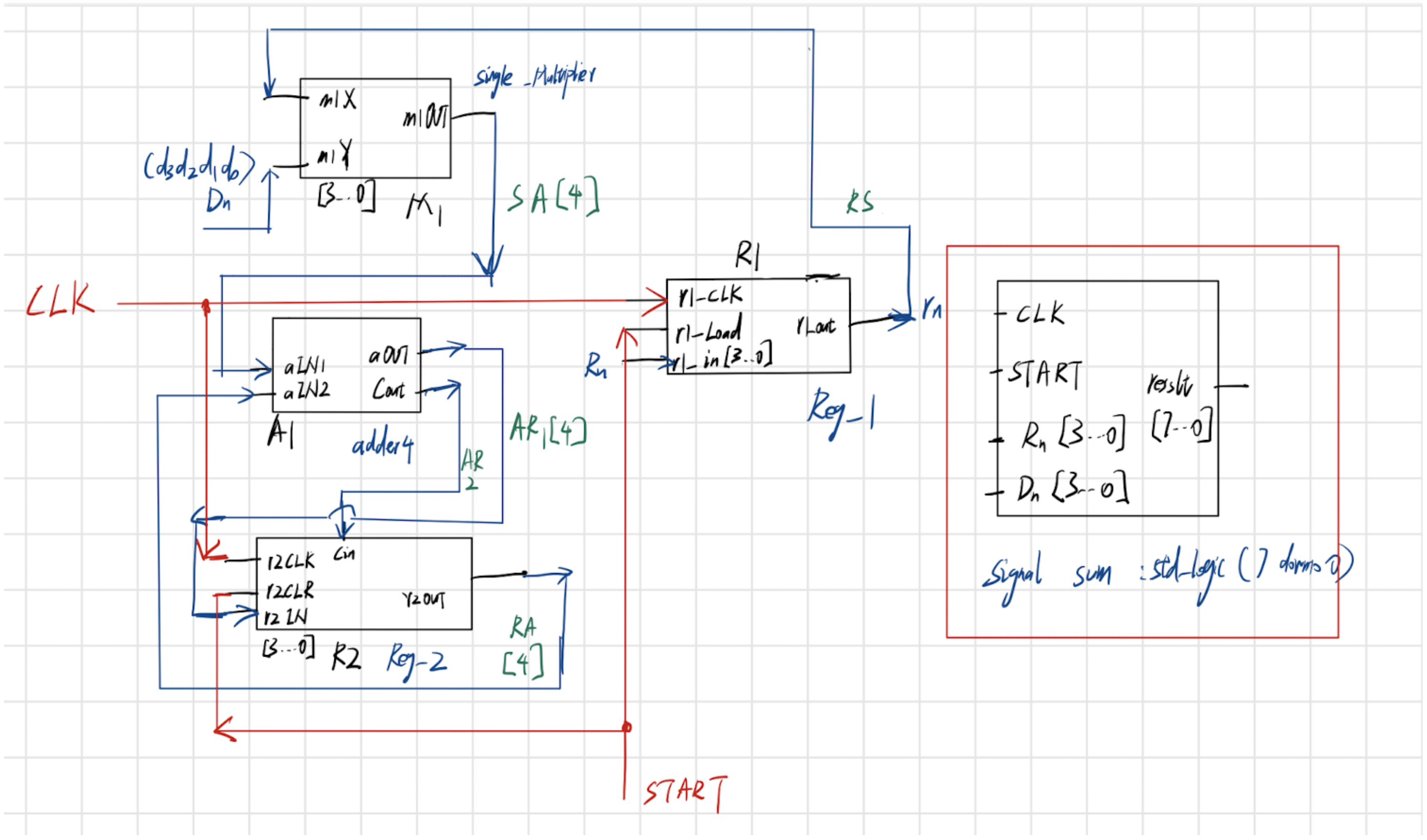


Fig.4.6. Full structure of Serial Multiplier

2.2: Testbench design process

- In the file serialM_tb.vhd file, I tested the input of 1111 x 1111, 1001 x 1101, 1100 x 1110 in 5 x 3 clk_cycles.
- In the first cycle, the START signal would be '1' to load and reset the register, in the next 4 cycles, START signal falls to '0' and the calculation happens at the rise-edge of CLK.

3: Delay Comparison of parallel and serial multiplier

3.0: Delay of the HA and FA

- In HA, delay from A/B to C is 1 τ.
- In FA, delay from A/B to C is 3 τ.

3.1: Parallel Multiplier Delay

For parallel multiplier, the value of $d_n r_n$ need one τ AND gate to get result at the same time.

The first line in Fig.3 using HA and cables, so the maximum delay from A/B to P1 is 1 τ. To form P2 and P3, several parallel FAs are applied, so the delay is 3 τ for each level.

The final level of parallel multiplier is a ripple-carry adder with one HA, 2 FA and one cable, so the delay is 2 x 3 τ for each FA and 1 τ for one HA. There is no delay for cables.

Finally, the delay from D/R to the result would be:

$$\tau_{total} = (1+1+3+3+1+3+3)\tau = 15\tau$$

3.2: Serial Multiplier

For serial multiplier, the full calculation need 1 clk_cycle to initialize value (without delay) and 4 clk_cycle to do the 4 partial_product calculation.

For each calculation clk_cycle, a 1x4 bits multiplier need one AND gate to form the answer in Fig.4.3, with the delay of 1 τ. To do the 4 bits adding calculation using entity Adder4 shown in Fig.4.4, which is composed of 4 full_adders using ripple carry adder. Its delay from input to output is 4 x 3τ. So the total delay of one calculation cycle is 13 τ.

The 4x4 bits multiplier have 4 calculation cycle, so the total delay to form the output is:

$$\tau_{total} = 13 \times 4\tau = 52\tau$$