Lab 6 Report

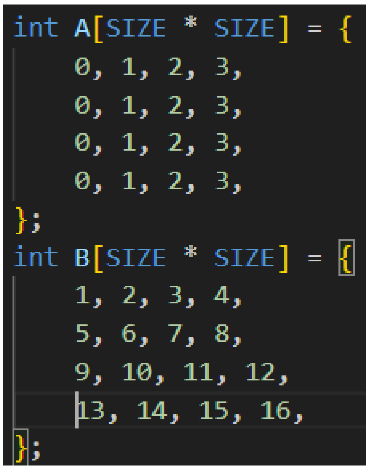
組別:第1組

姓名:游承緯 312580044 符顥瀚 312510154

Run simulation

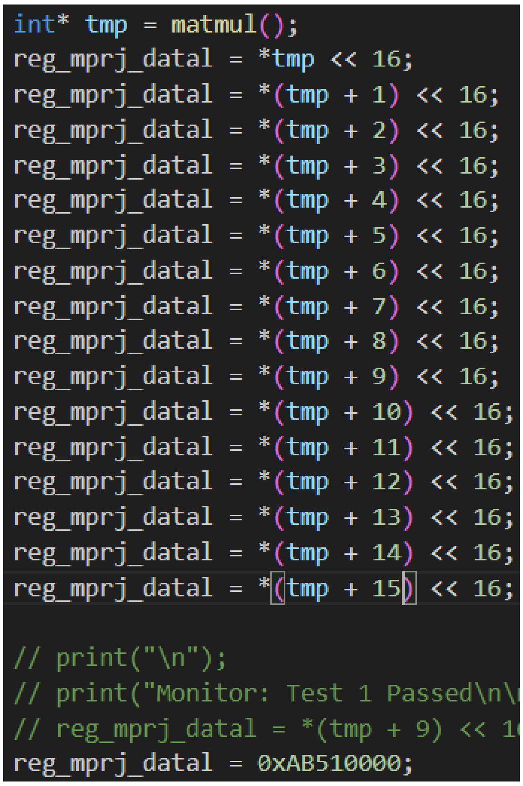
1. matrix multiplication

   a. matmul.h



```
int A[SIZE * SIZE] = {
    0, 1, 2, 3,
    0, 1, 2, 3,
    0, 1, 2, 3,
    0, 1, 2, 3,
};
int B[SIZE * SIZE] = {
    1, 2, 3, 4,
    5, 6, 7, 8,
    9, 10, 11, 12,
    13, 14, 15, 16,
};
```

   b. counter_la_mm.c



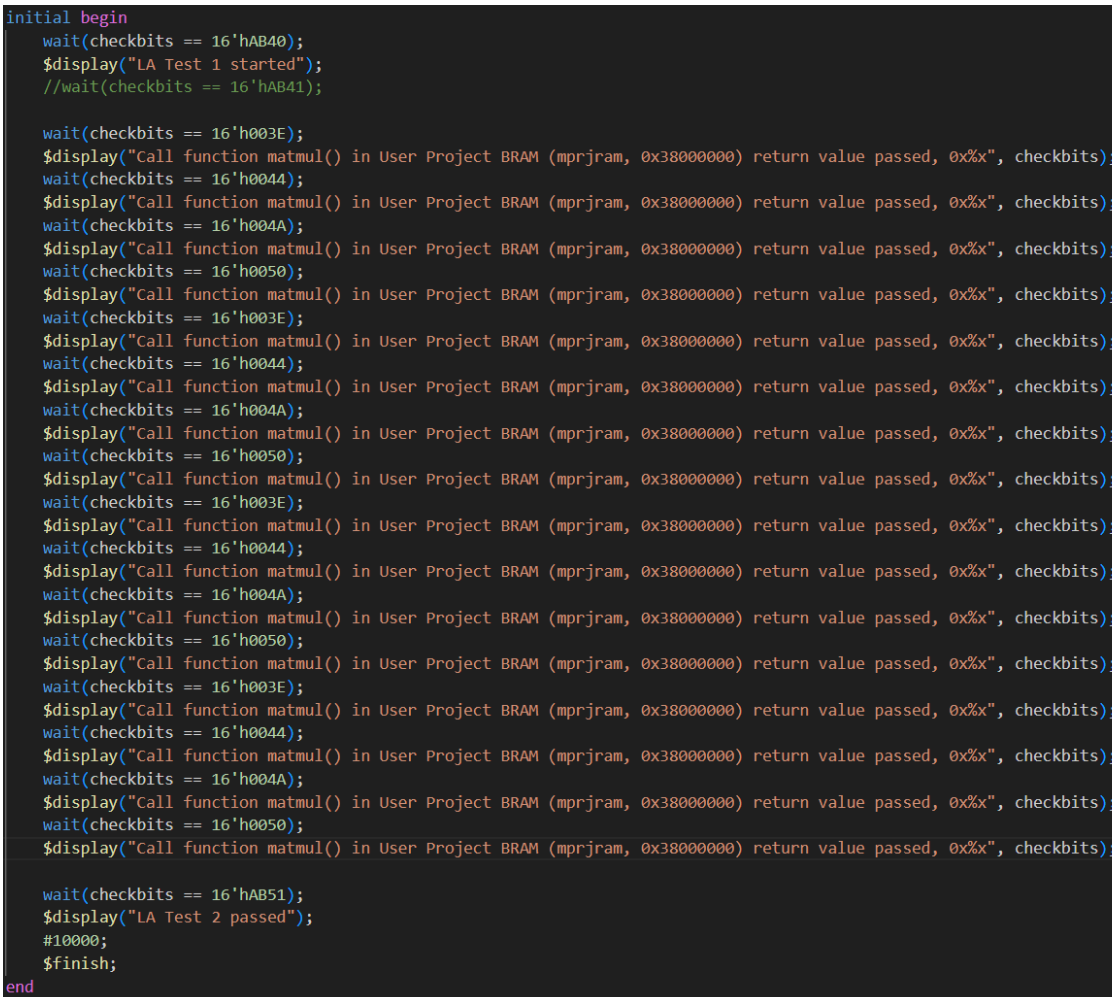
```
int* tmp = matmul();
reg_mprj_datal = *tmp << 16;
reg_mprj_datal = *(tmp + 1) << 16;
reg_mprj_datal = *(tmp + 2) << 16;
reg_mprj_datal = *(tmp + 3) << 16;
reg_mprj_datal = *(tmp + 4) << 16;
reg_mprj_datal = *(tmp + 5) << 16;
reg_mprj_datal = *(tmp + 6) << 16;
reg_mprj_datal = *(tmp + 7) << 16;
reg_mprj_datal = *(tmp + 8) << 16;
reg_mprj_datal = *(tmp + 9) << 16;
reg_mprj_datal = *(tmp + 10) << 16;
reg_mprj_datal = *(tmp + 11) << 16;
reg_mprj_datal = *(tmp + 12) << 16;
reg_mprj_datal = *(tmp + 13) << 16;
reg_mprj_datal = *(tmp + 14) << 16;
reg_mprj_datal = *(tmp + 15) << 16;

// print("\n");
// print("Monitor: Test 1 Passed\n\
// reg_mprj_datal = *(tmp + 9) << 1
reg_mprj_datal = 0xAB510000;
```

c. counter_la_mm_tb.v

透過testbench的checkbits去檢查結果是否正確

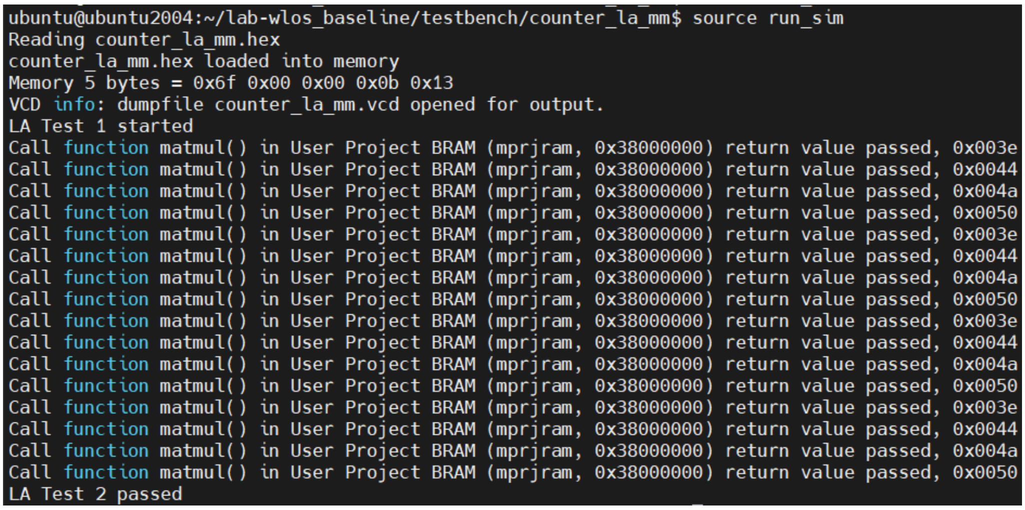
由以上的input可以算出，十進位的正確結果依序為

{62, 68, 74, 80, 62, 68, 74, 80, 62, 68, 74, 80, 62, 68, 74, 80}



```
initial begin
    wait(checkbits == 16'hAB40);
    $display("LA Test 1 started");
    //wait(checkbits == 16'hAB41);

    wait(checkbits == 16'h003E);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits)
    wait(checkbits == 16'h0044);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits)
    wait(checkbits == 16'h004A);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits)
    wait(checkbits == 16'h0050);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits)
    wait(checkbits == 16'h003E);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits)
    wait(checkbits == 16'h0044);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits)
    wait(checkbits == 16'h004A);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits)
    wait(checkbits == 16'h0050);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits)
    wait(checkbits == 16'h003E);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits)
    wait(checkbits == 16'h0044);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits)
    wait(checkbits == 16'h004A);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits)
    wait(checkbits == 16'h0050);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits)
    wait(checkbits == 16'h003E);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits)
    wait(checkbits == 16'h0044);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits)
    wait(checkbits == 16'h004A);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits)
    wait(checkbits == 16'h0050);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits)

    wait(checkbits == 16'hAB51);
    $display("LA Test 2 passed");
    #10000;
    $finish;
end
```

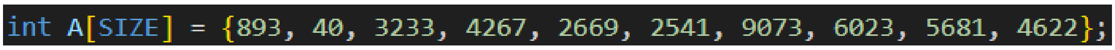
d. Result



```
ubuntu@ubuntu2004:~/lab-wlos_baseline/testbench/counter_la_mm$ source run_sim
Reading counter_la_mm.hex
counter_la_mm.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
VCD info: dumpfile counter_la_mm.vcd opened for output.
LA Test 1 started
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x003e
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0044
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x004a
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0050
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x003e
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0044
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x004a
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0050
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x003e
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0044
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x004a
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0050
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x003e
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0044
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x004a
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0050
LA Test 2 passed
```

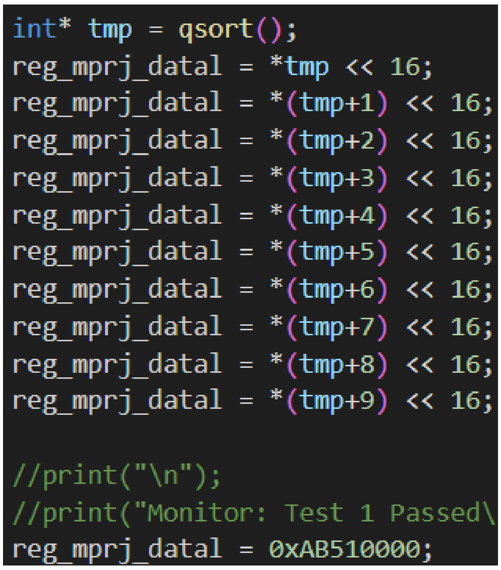
2. Quick Sort

a. qsort.h



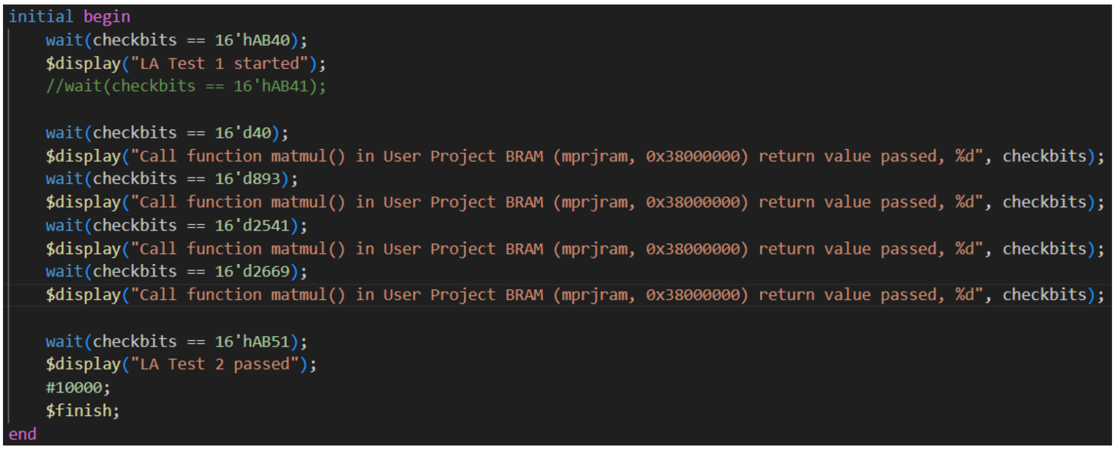
```
int A[SIZE] = {893, 40, 3233, 4267, 2669, 2541, 9073, 6023, 5681, 4622};
```

b. counter_la_qs.c



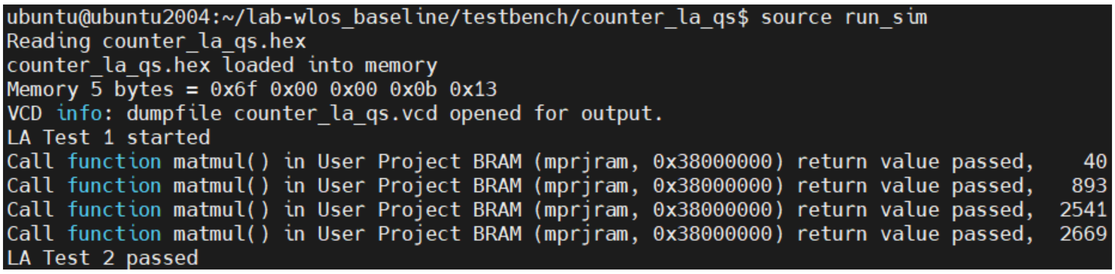
```
int* tmp = qsort();
reg_mprj_datal = *tmp << 16;
reg_mprj_datal = *(tmp+1) << 16;
reg_mprj_datal = *(tmp+2) << 16;
reg_mprj_datal = *(tmp+3) << 16;
reg_mprj_datal = *(tmp+4) << 16;
reg_mprj_datal = *(tmp+5) << 16;
reg_mprj_datal = *(tmp+6) << 16;
reg_mprj_datal = *(tmp+7) << 16;
reg_mprj_datal = *(tmp+8) << 16;
reg_mprj_datal = *(tmp+9) << 16;

//print("\n");
//print("Monitor: Test 1 Passed\
reg_mprj_datal = 0xAB510000;
```

c. counter_la_qs_tb.v



```
initial begin
    wait(checkbits == 16'hAB40);
    $display("LA Test 1 started");
    //wait(checkbits == 16'hAB41);

    wait(checkbits == 16'd40);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, %d", checkbits);
    wait(checkbits == 16'd893);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, %d", checkbits);
    wait(checkbits == 16'd2541);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, %d", checkbits);
    wait(checkbits == 16'd2669);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, %d", checkbits);

    wait(checkbits == 16'hAB51);
    $display("LA Test 2 passed");
    #10000;
    $finish;
end
```

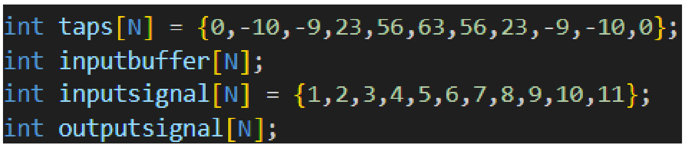
d. Result



```
ubuntu@ubuntu2004:~/lab-wlos_baseline/testbench/counter_la_qs$ source run_sim
Reading counter_la_qs.hex
counter_la_qs.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
VCD info: dumpfile counter_la_qs.vcd opened for output.
LA Test 1 started
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed,    40
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed,   893
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed,  2541
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed,  2669
LA Test 2 passed
```

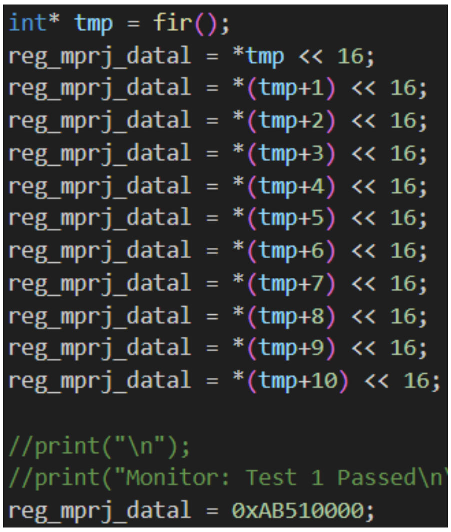
3. Fir

a. fir.h



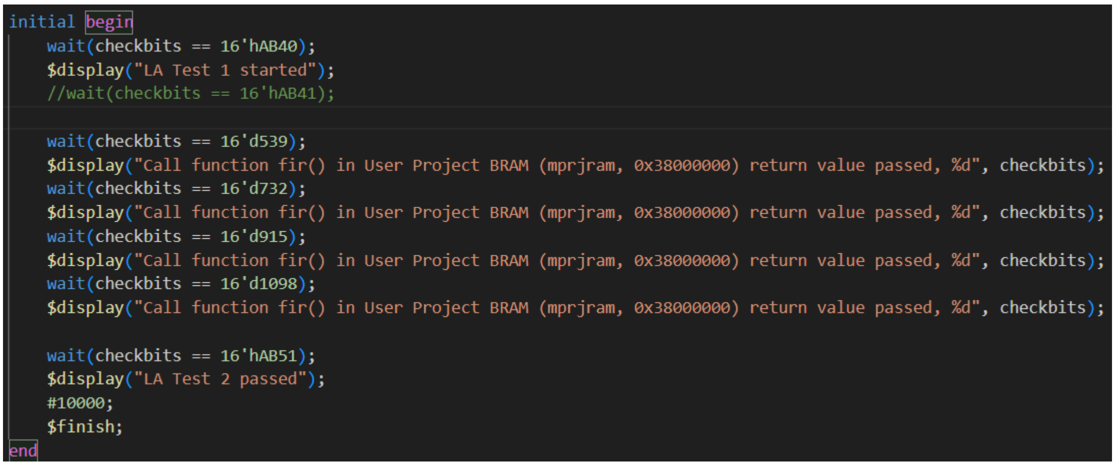
```
int taps[N] = {0,-10,-9,23,56,63,56,23,-9,-10,0};
int inputbuffer[N];
int inputsignal[N] = {1,2,3,4,5,6,7,8,9,10,11};
int outputsignal[N];
```

b. counter_la_fir.c



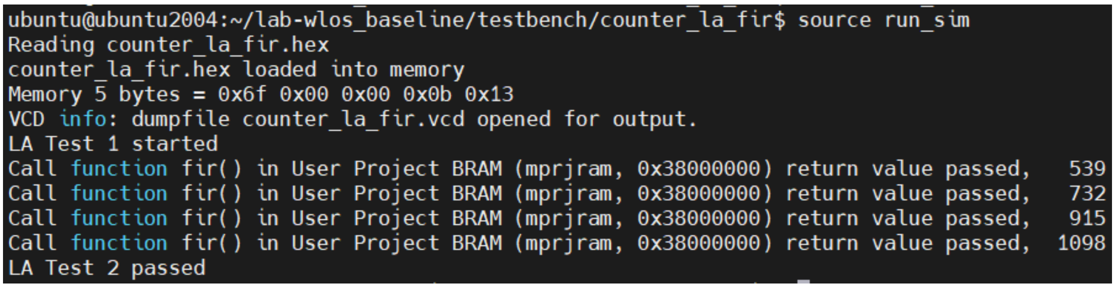
```
int* tmp = fir();
reg_mprj_datal = *tmp << 16;
reg_mprj_datal = *(tmp+1) << 16;
reg_mprj_datal = *(tmp+2) << 16;
reg_mprj_datal = *(tmp+3) << 16;
reg_mprj_datal = *(tmp+4) << 16;
reg_mprj_datal = *(tmp+5) << 16;
reg_mprj_datal = *(tmp+6) << 16;
reg_mprj_datal = *(tmp+7) << 16;
reg_mprj_datal = *(tmp+8) << 16;
reg_mprj_datal = *(tmp+9) << 16;
reg_mprj_datal = *(tmp+10) << 16;

//print("\n");
//print("Monitor: Test 1 Passed\n
reg_mprj_datal = 0xAB510000;
```

c. counter_la_fir_tb.v



```
initial begin
    wait(checkbits == 16'hAB40);
    $display("LA Test 1 started");
    //wait(checkbits == 16'hAB41);

    wait(checkbits == 16'd539);
    $display("Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, %d", checkbits);
    wait(checkbits == 16'd732);
    $display("Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, %d", checkbits);
    wait(checkbits == 16'd915);
    $display("Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, %d", checkbits);
    wait(checkbits == 16'd1098);
    $display("Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, %d", checkbits);

    wait(checkbits == 16'hAB51);
    $display("LA Test 2 passed");
    #10000;
    $finish;
end
```

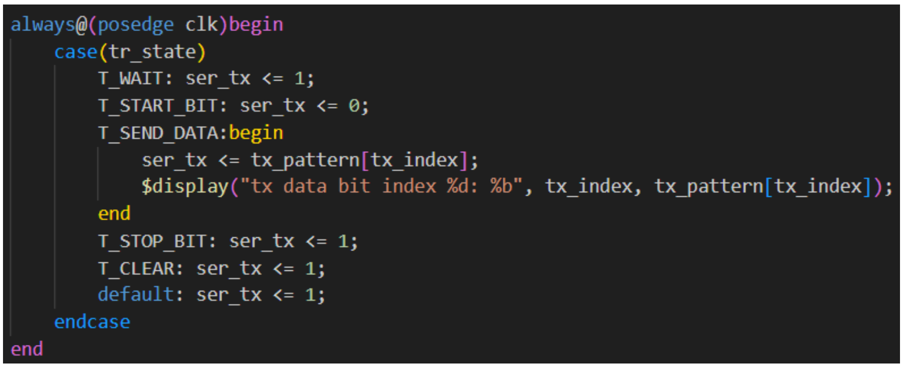
d. Result



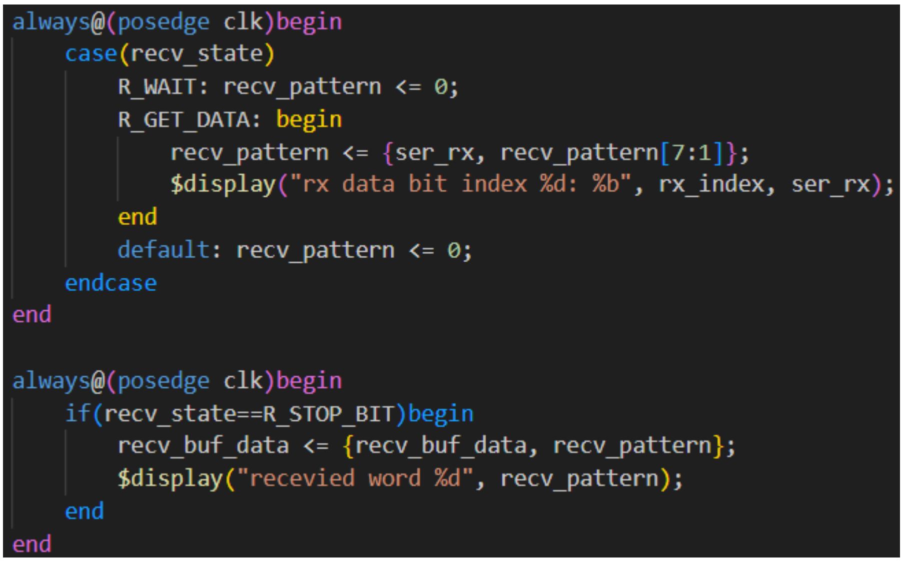
```
ubuntu@ubuntu2004:~/lab-wlos_baseline/testbench/counter_la_fir$ source run_sim
Reading counter_la_fir.hex
counter_la_fir.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
VCD info: dumpfile counter_la_fir.vcd opened for output.
LA Test 1 started
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,   539
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,   732
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,   915
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,  1098
LA Test 2 passed
```

4. Uart

a. tbuart.v

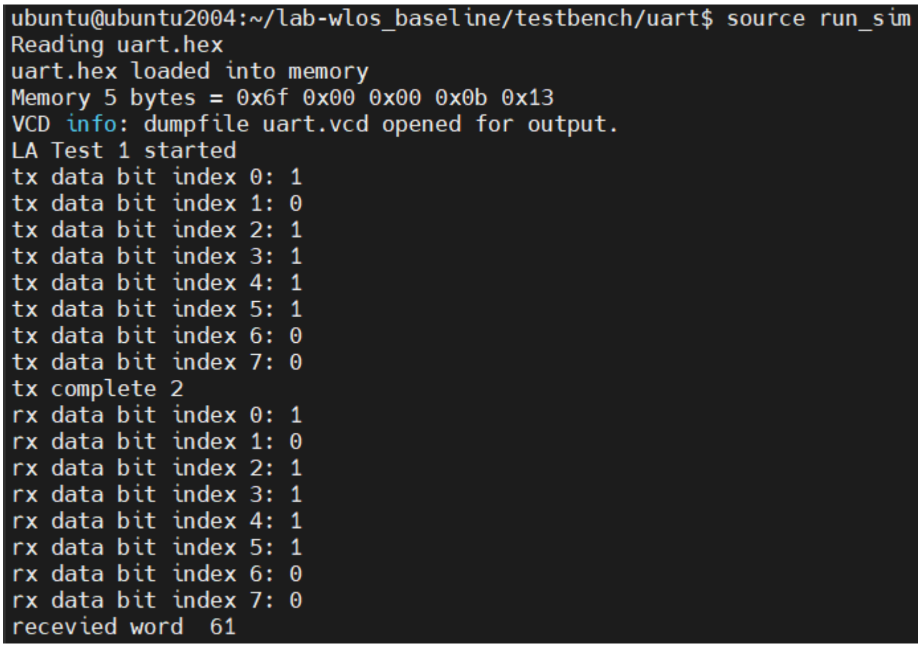


```
always@(posedge clk)begin
    case(tr_state)
        T_WAIT: ser_tx <= 1;
        T_START_BIT: ser_tx <= 0;
        T_SEND_DATA:begin
            ser_tx <= tx_pattern[tx_index];
            $display("tx data bit index %d: %b", tx_index, tx_pattern[tx_index]);
        end
        T_STOP_BIT: ser_tx <= 1;
        T_CLEAR: ser_tx <= 1;
        default: ser_tx <= 1;
    endcase
end
```



```
always@(posedge clk)begin
    case(recv_state)
        R_WAIT: recv_pattern <= 0;
        R_GET_DATA: begin
            recv_pattern <= {ser_rx, recv_pattern[7:1]};
            $display("rx data bit index %d: %b", rx_index, ser_rx);
        end
        default: recv_pattern <= 0;
    endcase
end

always@(posedge clk)begin
    if(recv_state==R_STOP_BIT)begin
        recv_buf_data <= {recv_buf_data, recv_pattern};
        $display("recevied word %d", recv_pattern);
    end
end
```

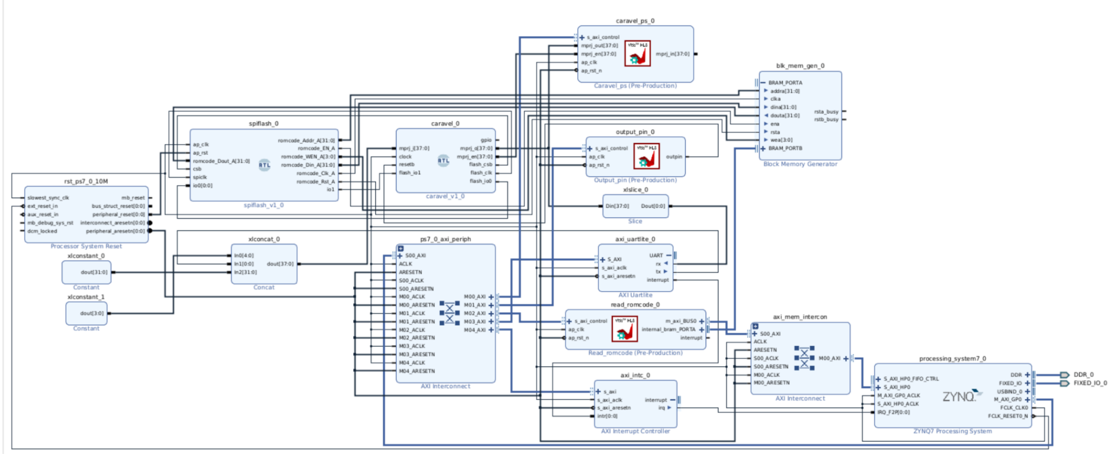
b. Result



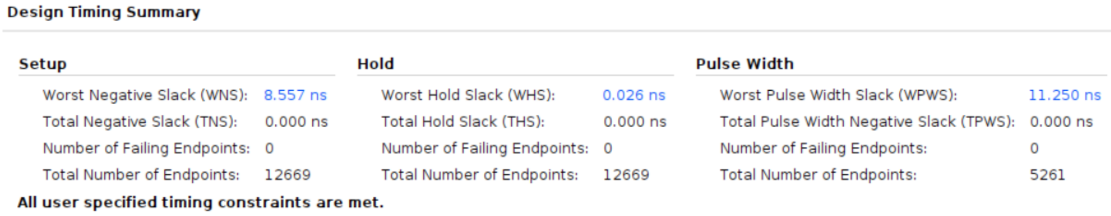
```
ubuntu@ubuntu2004:~/lab-wlos_baseline/testbench/uart$ source run_sim
Reading uart.hex
uart.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
VCD info: dumpfile uart.vcd opened for output.
LA Test 1 started
tx data bit index 0: 1
tx data bit index 1: 0
tx data bit index 2: 1
tx data bit index 3: 1
tx data bit index 4: 1
tx data bit index 5: 1
tx data bit index 6: 0
tx data bit index 7: 0
tx complete 2
rx data bit index 0: 1
rx data bit index 1: 0
rx data bit index 2: 1
rx data bit index 3: 1
rx data bit index 4: 1
rx data bit index 5: 1
rx data bit index 6: 0
rx data bit index 7: 0
recevied word  61
```

Run UART on FPGA

1. Block design



2. Time report

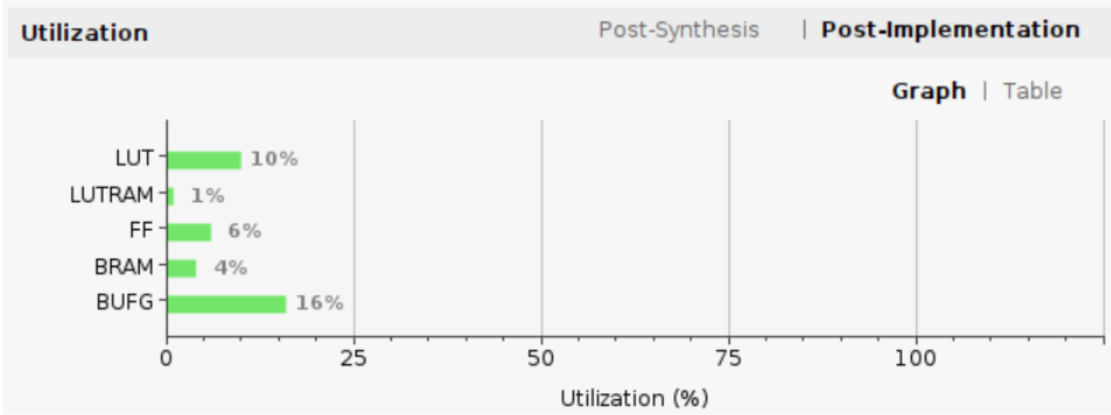


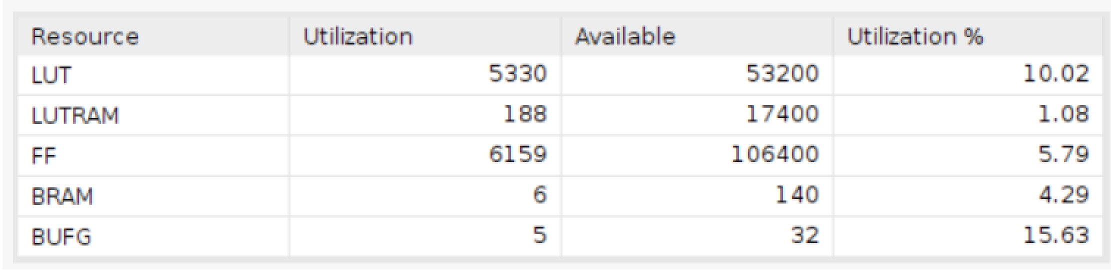
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	8.557 ns	Worst Hold Slack (WHS):	0.026 ns	Worst Pulse Width Slack (WPWS):	11.250 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	12669	Total Number of Endpoints:	12669	Total Number of Endpoints:	5261

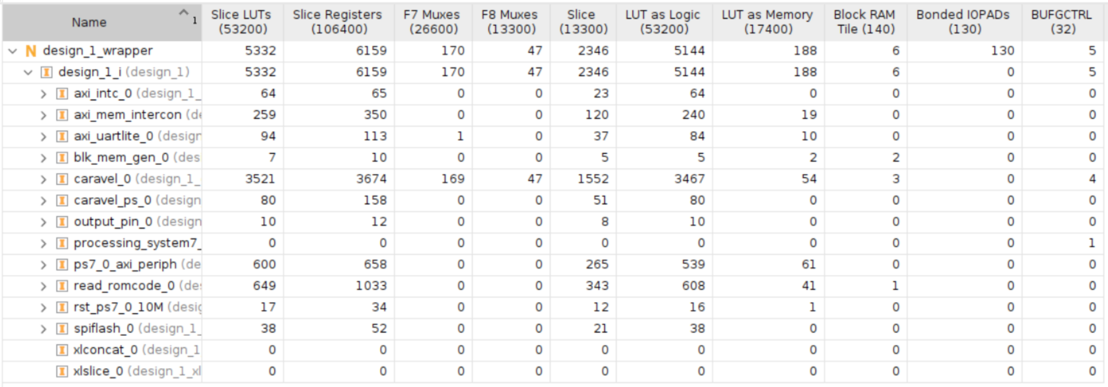
All user specified timing constraints are met.

3. Resource report



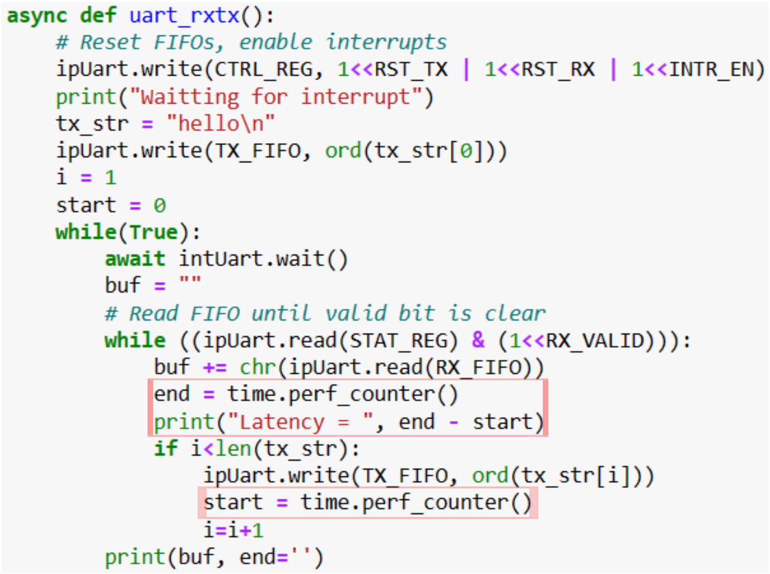


Resource	Utilization	Available	Utilization %
LUT	5330	53200	10.02
LUTRAM	188	17400	1.08
FF	6159	106400	5.79
BRAM	6	140	4.29
BUFG	5	32	15.63

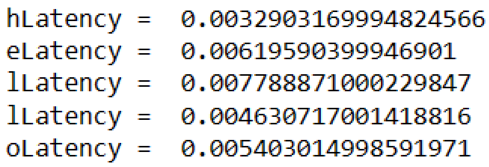


Name	Slice LUTs (53200)	Slice Registers (106400)	F7 Muxes (26600)	F8 Muxes (13300)	Slice (13300)	LUT as Logic (53200)	LUT as Memory (17400)	Block RAM Tile (140)	Bonded IOPADs (130)	BUFGCTRL (32)
design_1_wrapper	5332	6159	170	47	2346	5144	188	6	130	5
design_1_i (design_1)	5332	6159	170	47	2346	5144	188	6	0	5
axi_intc_0 (design_1_	64	65	0	0	23	64	0	0	0	0
axi_mem_intercon (de	259	350	0	0	120	240	19	0	0	0
axi_uartlite_0 (design	94	113	1	0	37	84	10	0	0	0
blk_mem_gen_0 (des	7	10	0	0	5	5	2	2	0	0
caravel_0 (design_1_	3521	3674	169	47	1552	3467	54	3	0	4
caravel_ps_0 (design	80	158	0	0	51	80	0	0	0	0
output_pin_0 (design	10	12	0	0	8	10	0	0	0	0
processing_system7_	0	0	0	0	0	0	0	0	0	1
ps7_0_axi_periph (de	600	658	0	0	265	539	61	0	0	0
read_romcode_0 (de	649	1033	0	0	343	608	41	1	0	0
rst_ps7_0_10M (desig	17	34	0	0	12	16	1	0	0	0
spiflash_0 (design_1_	38	52	0	0	21	38	0	0	0	0
xlconcat_0 (design_1	0	0	0	0	0	0	0	0	0	0
xlslice_0 (design_1_xl	0	0	0	0	0	0	0	0	0	0

4. Latency for a character loop back using UART



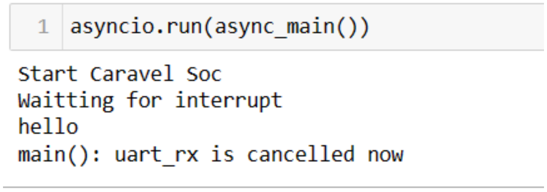
```
async def uart_rxtx():
    # Reset FIFOs, enable interrupts
    ipUart.write(CTRL_REG, 1<<RST_TX | 1<<RST_RX | 1<<INTR_EN)
    print("Waitting for interrupt")
    tx_str = "hello\n"
    ipUart.write(TX_FIFO, ord(tx_str[0]))
    i = 1
    start = 0
    while(True):
        await intUart.wait()
        buf = ""
        # Read FIFO until valid bit is clear
        while ((ipUart.read(STAT_REG) & (1<<RX_VALID))):
            buf += chr(ipUart.read(RX_FIFO))
            end = time.perf_counter()
            print("Latency = ", end - start)
            if i<len(tx_str):
                ipUart.write(TX_FIFO, ord(tx_str[i]))
                start = time.perf_counter()
                i=i+1
        print(buf, end='')
```



```
hLatency =  0.0032903169994824566
eLatency =  0.00619590399946901
lLatency =  0.007788871000229847
lLatency =  0.004630717001418816
oLatency =  0.005403014998591971
```

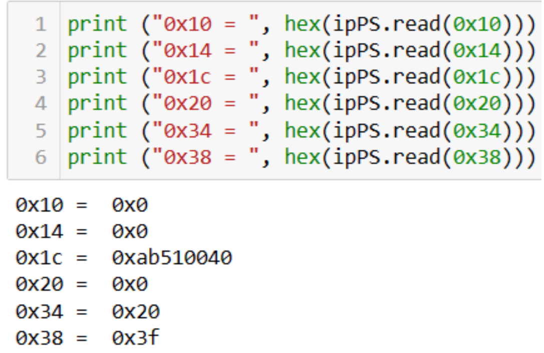
每個 character loop back 的 latency time 大概介於0.003~0.008之間

5. Result



```
asyncio.run(async_main())
```

```
Start Caravel Soc
Waitting for interrupt
hello
main(): uart_rx is cancelled now
```



```
print ("0x10 = ", hex(ipPS.read(0x10)))
print ("0x14 = ", hex(ipPS.read(0x14)))
print ("0x1c = ", hex(ipPS.read(0x1c)))
print ("0x20 = ", hex(ipPS.read(0x20)))
print ("0x34 = ", hex(ipPS.read(0x34)))
print ("0x38 = ", hex(ipPS.read(0x38)))
```

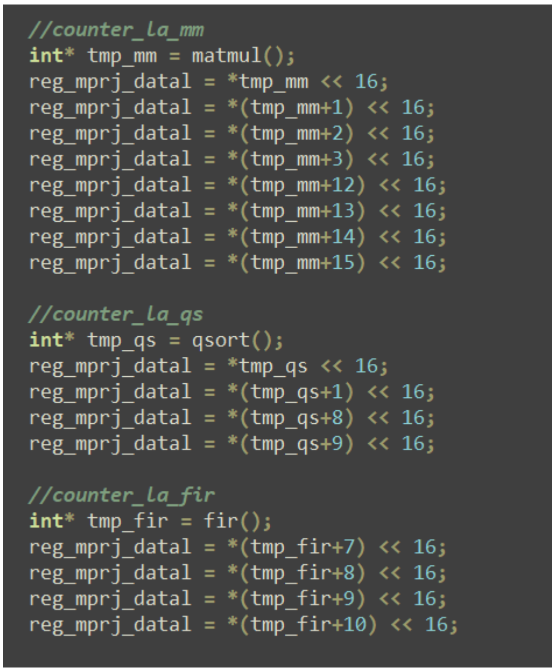
```
0x10 =  0x0
0x14 =  0x0
0x1c =  0xab510040
0x20 =  0x0
0x34 =  0x20
0x38 =  0x3f
```

6. Suggestion for improving latency for UART loop back

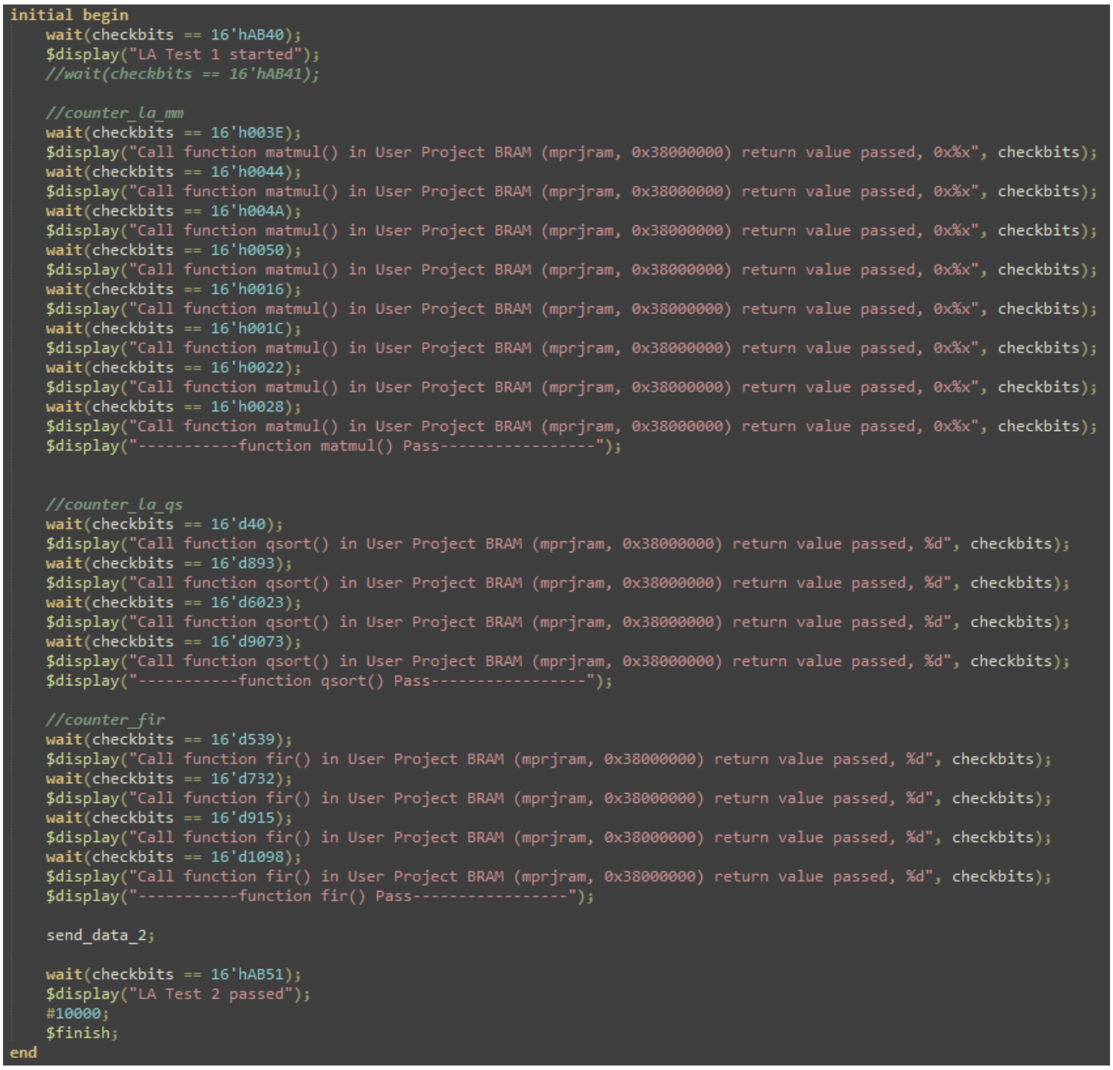
因為design中可能會有許多 character 的傳出都會需要觸發 interrupt，加入 FIFO 後可以用來暫存 傳入和傳出的 data，應該可以有效降低 latency

Merge

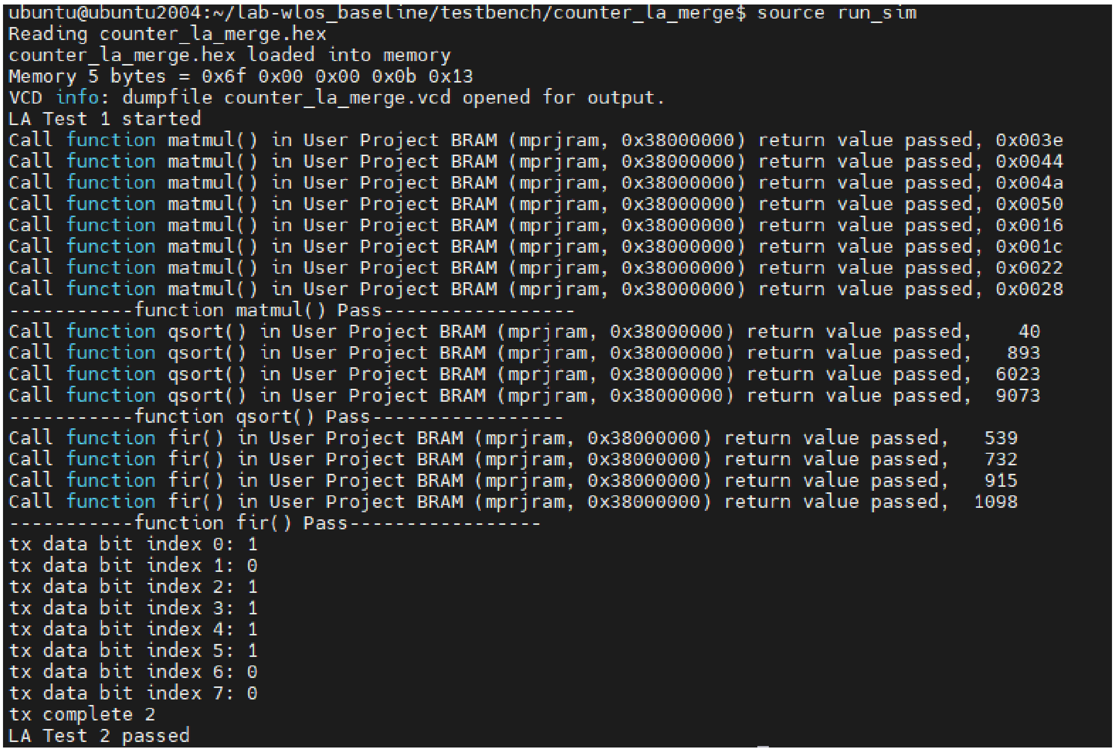
把前面的firmware code合併起來，並依序做matmul、qsort、fir、uart的檢查。



```c
//counter_la_mm
int* tmp_mm = matmul();
reg_mprj_datal = *tmp_mm << 16;
reg_mprj_datal = *(tmp_mm+1) << 16;
reg_mprj_datal = *(tmp_mm+2) << 16;
reg_mprj_datal = *(tmp_mm+3) << 16;
reg_mprj_datal = *(tmp_mm+12) << 16;
reg_mprj_datal = *(tmp_mm+13) << 16;
reg_mprj_datal = *(tmp_mm+14) << 16;
reg_mprj_datal = *(tmp_mm+15) << 16;

//counter_la_qs
int* tmp_qs = qsort();
reg_mprj_datal = *tmp_qs << 16;
reg_mprj_datal = *(tmp_qs+1) << 16;
reg_mprj_datal = *(tmp_qs+8) << 16;
reg_mprj_datal = *(tmp_qs+9) << 16;

//counter_la_fir
int* tmp_fir = fir();
reg_mprj_datal = *(tmp_fir+7) << 16;
reg_mprj_datal = *(tmp_fir+8) << 16;
reg_mprj_datal = *(tmp_fir+9) << 16;
reg_mprj_datal = *(tmp_fir+10) << 16;
```



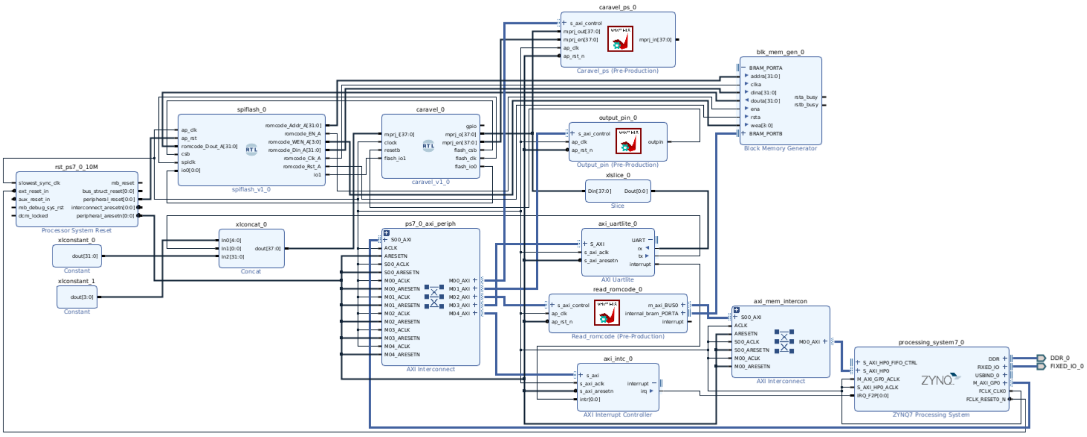
```
initial begin
    wait(checkbits == 16'hAB40);
    $display("LA Test 1 started");
    //wait(checkbits == 16'hAB41);

    //counter_la_mm
    wait(checkbits == 16'h003E);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits);
    wait(checkbits == 16'h0044);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits);
    wait(checkbits == 16'h004A);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits);
    wait(checkbits == 16'h0050);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits);
    wait(checkbits == 16'h0016);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits);
    wait(checkbits == 16'h001C);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits);
    wait(checkbits == 16'h0022);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits);
    wait(checkbits == 16'h0028);
    $display("Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x%x", checkbits);
    $display("----------function matmul() Pass-----------------");


    //counter_la_qs
    wait(checkbits == 16'd40);
    $display("Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed, %d", checkbits);
    wait(checkbits == 16'd893);
    $display("Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed, %d", checkbits);
    wait(checkbits == 16'd6023);
    $display("Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed, %d", checkbits);
    wait(checkbits == 16'd9073);
    $display("Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed, %d", checkbits);
    $display("----------function qsort() Pass-----------------");

    //counter_fir
    wait(checkbits == 16'd539);
    $display("Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, %d", checkbits);
    wait(checkbits == 16'd732);
    $display("Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, %d", checkbits);
    wait(checkbits == 16'd915);
    $display("Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, %d", checkbits);
    wait(checkbits == 16'd1098);
    $display("Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, %d", checkbits);
    $display("----------function fir() Pass-----------------");

    send_data_2;

    wait(checkbits == 16'hAB51);
    $display("LA Test 2 passed");
    #10000;
    $finish;
end
```



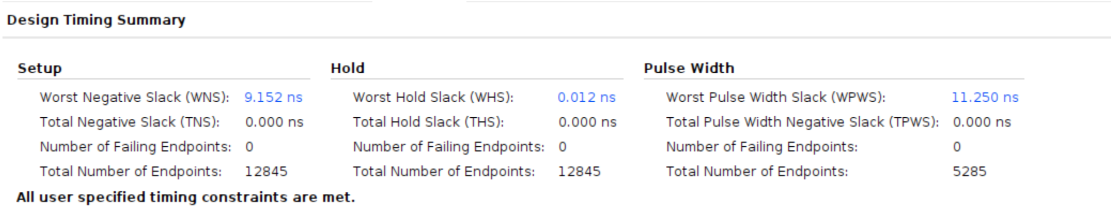
```
ubuntu@ubuntu2004:~/lab-wlos_baseline/testbench/counter_la_merge$ source run_sim
Reading counter_la_merge.hex
counter_la_merge.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
VCD info: dumpfile counter_la_merge.vcd opened for output.
LA Test 1 started
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x003e
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0044
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x004a
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0050
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0016
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x001c
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0022
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0028
----------function matmul() Pass-----------------
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed,    40
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed,   893
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed,  6023
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed,  9073
----------function qsort() Pass-----------------
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,   539
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,   732
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,   915
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,  1098
----------function fir() Pass-----------------
tx data bit index 0: 1
tx data bit index 1: 0
tx data bit index 2: 1
tx data bit index 3: 1
tx data bit index 4: 1
tx data bit index 5: 1
tx data bit index 6: 0
tx data bit index 7: 0
tx complete 2
LA Test 2 passed
```

Run Merge on Vivado

1. Block design



2. Timing report

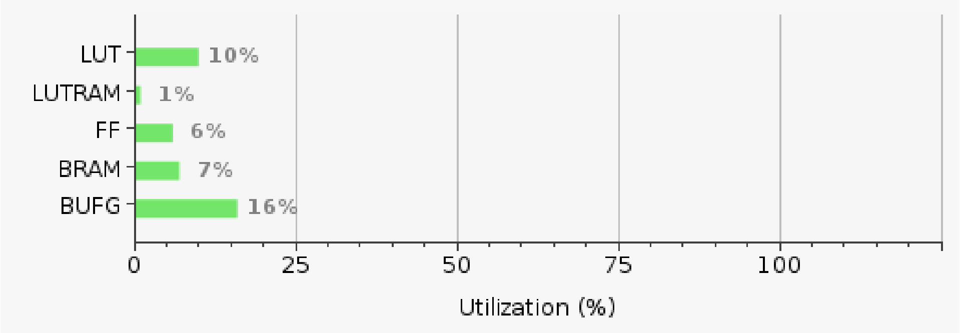


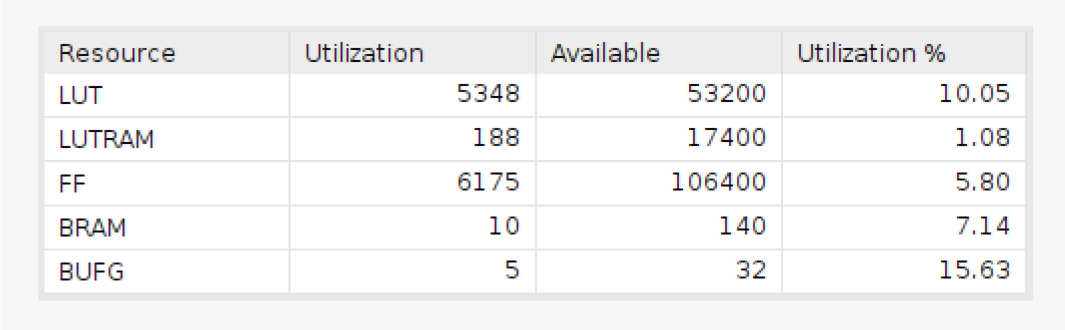
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	9.152 ns	Worst Hold Slack (WHS):	0.012 ns	Worst Pulse Width Slack (WPWS):	11.250 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	12845	Total Number of Endpoints:	12845	Total Number of Endpoints:	5285

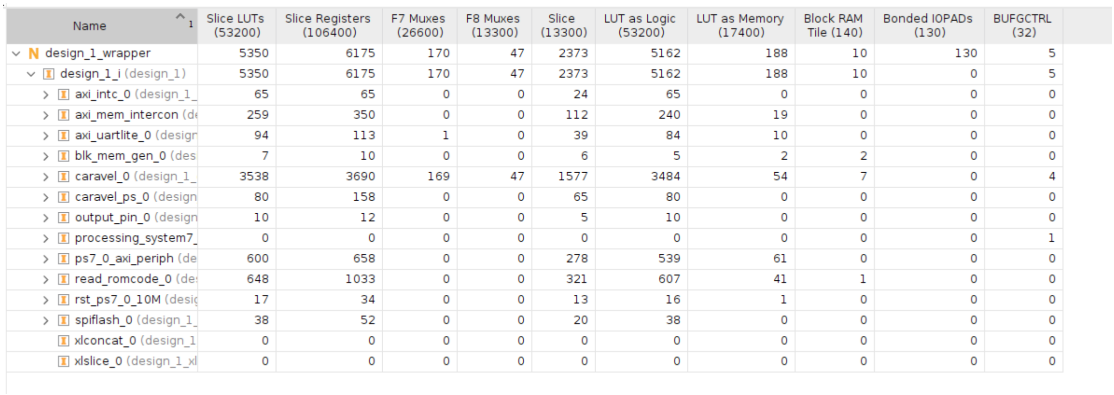
All user specified timing constraints are met.

3. Resource report





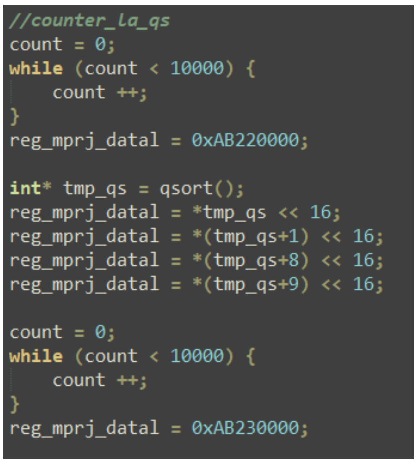
Resource	Utilization	Available	Utilization %
LUT	5348	53200	10.05
LUTRAM	188	17400	1.08
FF	6175	106400	5.80
BRAM	10	140	7.14
BUFG	5	32	15.63



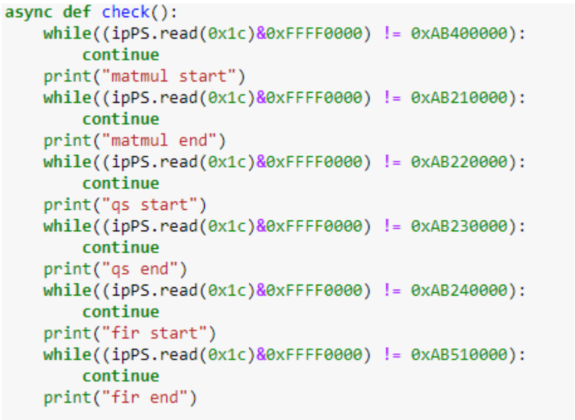
Name	Slice LUTs (53200)	Slice Registers (106400)	F7 Muxes (26600)	F8 Muxes (13300)	Slice (13300)	LUT as Logic (53200)	LUT as Memory (17400)	Block RAM Tile (140)	Bonded IOPADs (130)	BUFGCTRL (32)
design_1_wrapper	5350	6175	170	47	2373	5162	188	10	130	5
design_1_i (design_1)	5350	6175	170	47	2373	5162	188	10	0	5
axi_intc_0 (design_1_	65	65	0	0	24	65	0	0	0	0
axi_mem_intercon (de	259	350	0	0	112	240	19	0	0	0
axi_uartlite_0 (design	94	113	1	0	39	84	10	0	0	0
blk_mem_gen_0 (des	7	10	0	0	6	5	2	2	0	0
caravel_0 (design_1_	3538	3690	169	47	1577	3484	54	7	0	4
caravel_ps_0 (design	80	158	0	0	65	80	0	0	0	0
output_pin_0 (design	10	12	0	0	5	10	0	0	0	0
processing_system7_	0	0	0	0	0	0	0	0	0	1
ps7_0_axi_periph (de	600	658	0	0	278	539	61	0	0	0
read_romcode_0 (de	648	1033	0	0	321	607	41	1	0	0
rst_ps7_0_10M (desig	17	34	0	0	13	16	1	0	0	0
spiflash_0 (design_1_	38	52	0	0	20	38	0	0	0	0
xlconcat_0 (design_1	0	0	0	0	0	0	0	0	0	0
xlslice_0 (design_1_xl	0	0	0	0	0	0	0	0	0	0

4. Verify our answer from notebook & Result

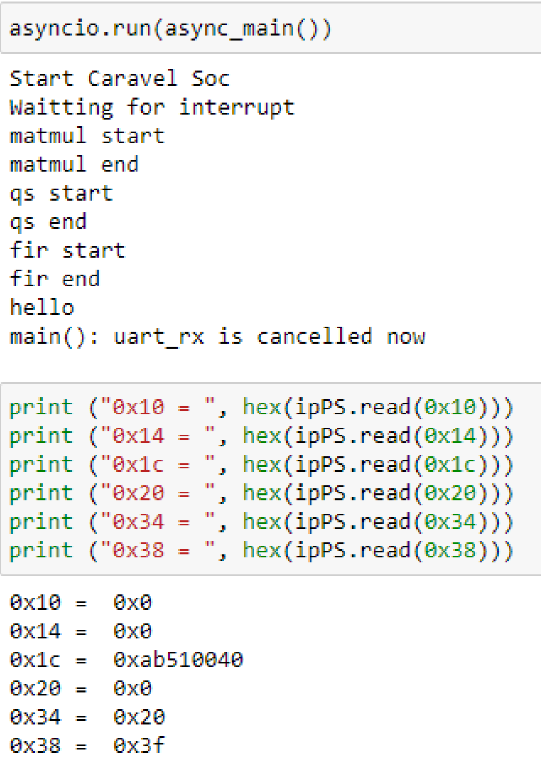
一開始驗證的時候，也都ipPS.read()不到想要read的值，於是上了討論區看大家的討論，跟我們推論的差不多，就是mprj端的變化太快了，notebook這邊沒辦法檢查到想要的值，所以就照了大家的方法，塞了一些while的loop，拖慢他變化的速度，於是就有以下的結果，有檢查到所有workload的開始與結束。



```
//counter_la_qs
count = 0;
while (count < 10000) {
    count ++;
}
reg_mprj_datal = 0xAB220000;

int* tmp_qs = qsort();
reg_mprj_datal = *tmp_qs << 16;
reg_mprj_datal = *(tmp_qs+1) << 16;
reg_mprj_datal = *(tmp_qs+8) << 16;
reg_mprj_datal = *(tmp_qs+9) << 16;

count = 0;
while (count < 10000) {
    count ++;
}
reg_mprj_datal = 0xAB230000;
```



```
async def check():
    while((ipPS.read(0x1c)&0xFFFF0000) != 0xAB400000):
        continue
    print("matmul start")
    while((ipPS.read(0x1c)&0xFFFF0000) != 0xAB210000):
        continue
    print("matmul end")
    while((ipPS.read(0x1c)&0xFFFF0000) != 0xAB220000):
        continue
    print("qs start")
    while((ipPS.read(0x1c)&0xFFFF0000) != 0xAB230000):
        continue
    print("qs end")
    while((ipPS.read(0x1c)&0xFFFF0000) != 0xAB240000):
        continue
    print("fir start")
    while((ipPS.read(0x1c)&0xFFFF0000) != 0xAB510000):
        continue
    print("fir end")
```



```
asyncio.run(async_main())
```

```
Start Caravel Soc
Waitting for interrupt
matmul start
matmul end
qs start
qs end
fir start
fir end
hello
main(): uart_rx is cancelled now
```

```
print ("0x10 = ", hex(ipPS.read(0x10)))
print ("0x14 = ", hex(ipPS.read(0x14)))
print ("0x1c = ", hex(ipPS.read(0x1c)))
print ("0x20 = ", hex(ipPS.read(0x20)))
print ("0x34 = ", hex(ipPS.read(0x34)))
print ("0x38 = ", hex(ipPS.read(0x38)))
```

```
0x10 =  0x0
0x14 =  0x0
0x1c =  0xab510040
0x20 =  0x0
0x34 =  0x20
0x38 =  0x3f
```

What else do you observe

這次大家的問題大部分都卡在firmware執行的速度比起PS端的快，這次討論出來的做法是把firmware放慢以配合PS端，但這樣大大的降低了模擬效率，如果可以控制firmware跟PS端連接的port就不用刻意把firmware放慢，hex檔就不需要那麼多instruction。