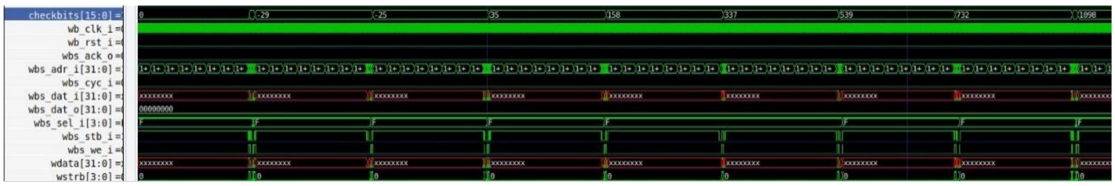
Soc Design Lab6 Report

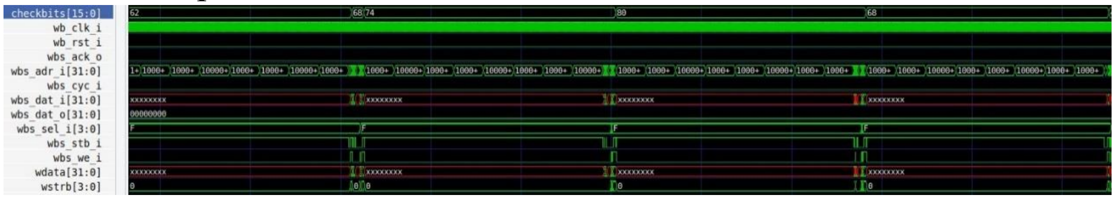
組員: 王證皓、丁緒翰、潘金生

•How do you verify your answer from notebook

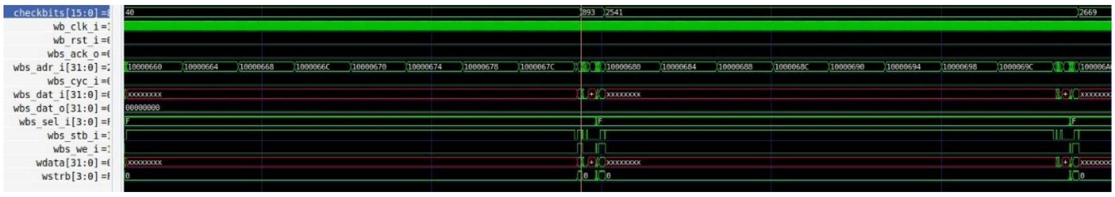
Fir:



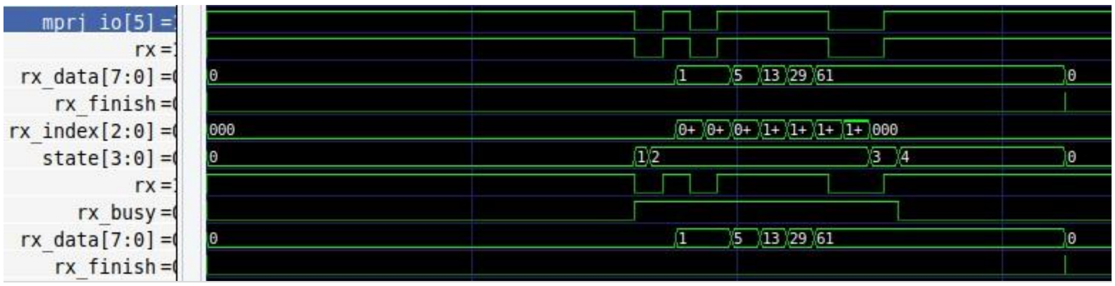
Matrix Multiplication:



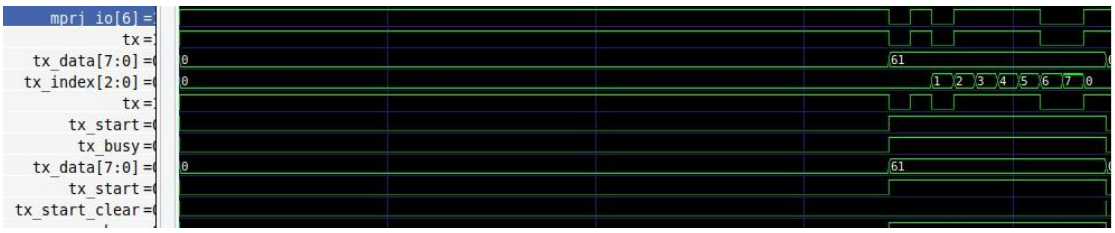
Quick Sort:



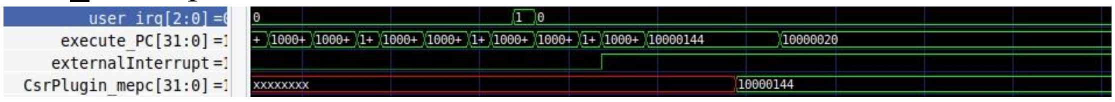
Uart_rx:



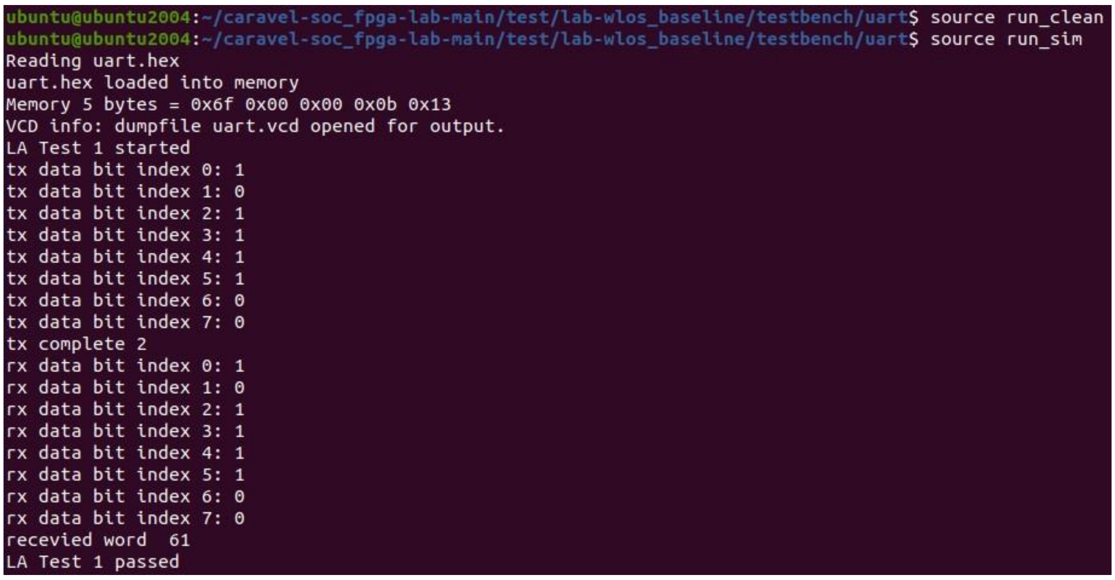
Uart_tx:



Uart_interrupt:



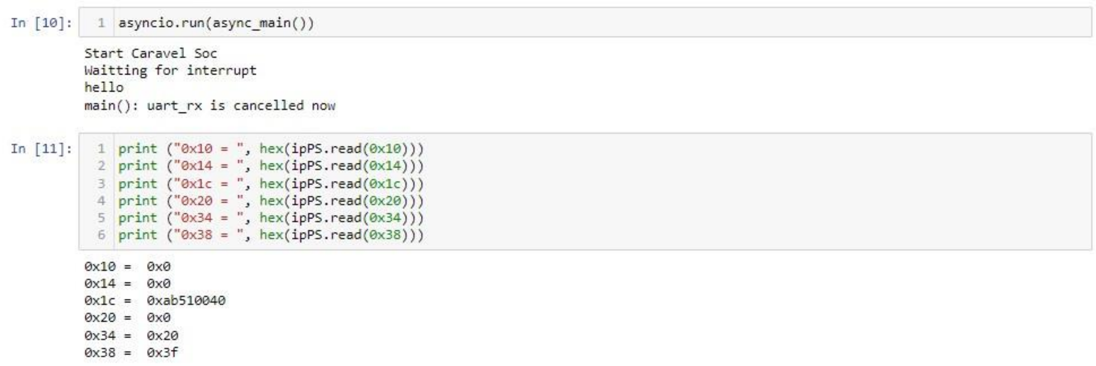
Uart_simulation:



```
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab-main/test/lab-wlos_baseline/testbench/uart$ source run_clean
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab-main/test/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
LA Test 1 passed
```

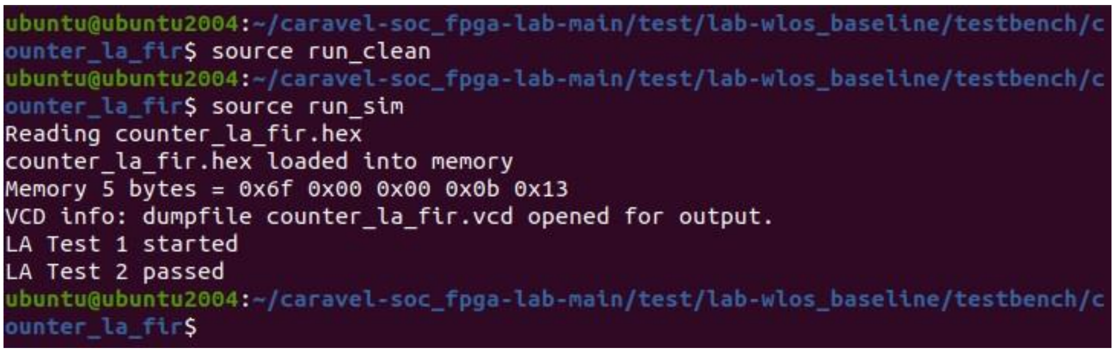
Notebook:

Hello 是 UART 輸出的結果且 UART 結束位置是 510040



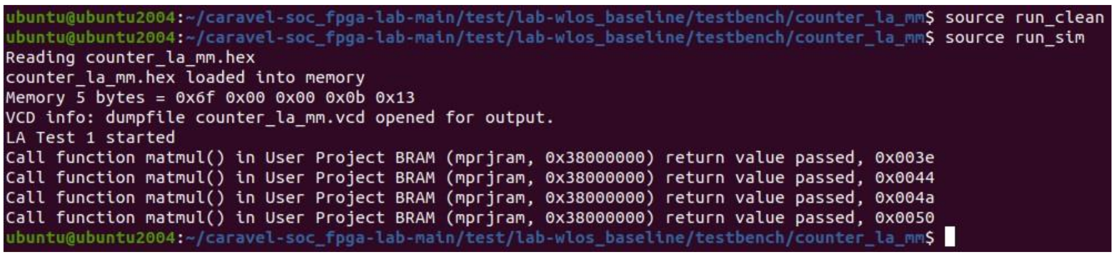
```
In [10]:  asyncio.run(async_main())

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

In [11]:  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
```

Fir_sim:



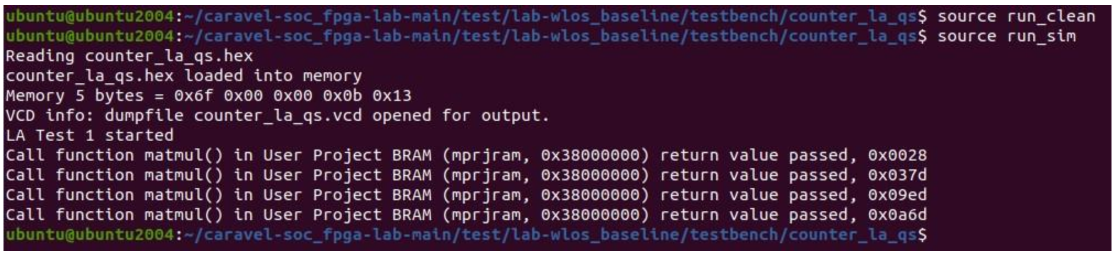
```
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab-main/test/lab-wlos_baseline/testbench/c
ounter_la_fir$ source run_clean
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab-main/test/lab-wlos_baseline/testbench/c
ounter_la_fir$ source run_sim
Reading counter_la_fir.hex
counter_la_fir.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
VCD info: dumpfile counter_la_fir.vcd opened for output.
LA Test 1 started
LA Test 2 passed
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab-main/test/lab-wlos_baseline/testbench/c
ounter_la_fir$
```

Matrix Multiplication_sim:



```
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab-main/test/lab-wlos_baseline/testbench/counter_la_mm$ source run_clean
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab-main/test/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
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab-main/test/lab-wlos_baseline/testbench/counter_la_mm$
```

Quick sort_sim:



```
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab-main/test/lab-wlos_baseline/testbench/counter_la_qs$ source run_clean
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab-main/test/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, 0x0028
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x037d
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x09ed
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0a6d
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab-main/test/lab-wlos_baseline/testbench/counter_la_qs$
```

以上皆為 6-1 之內容。

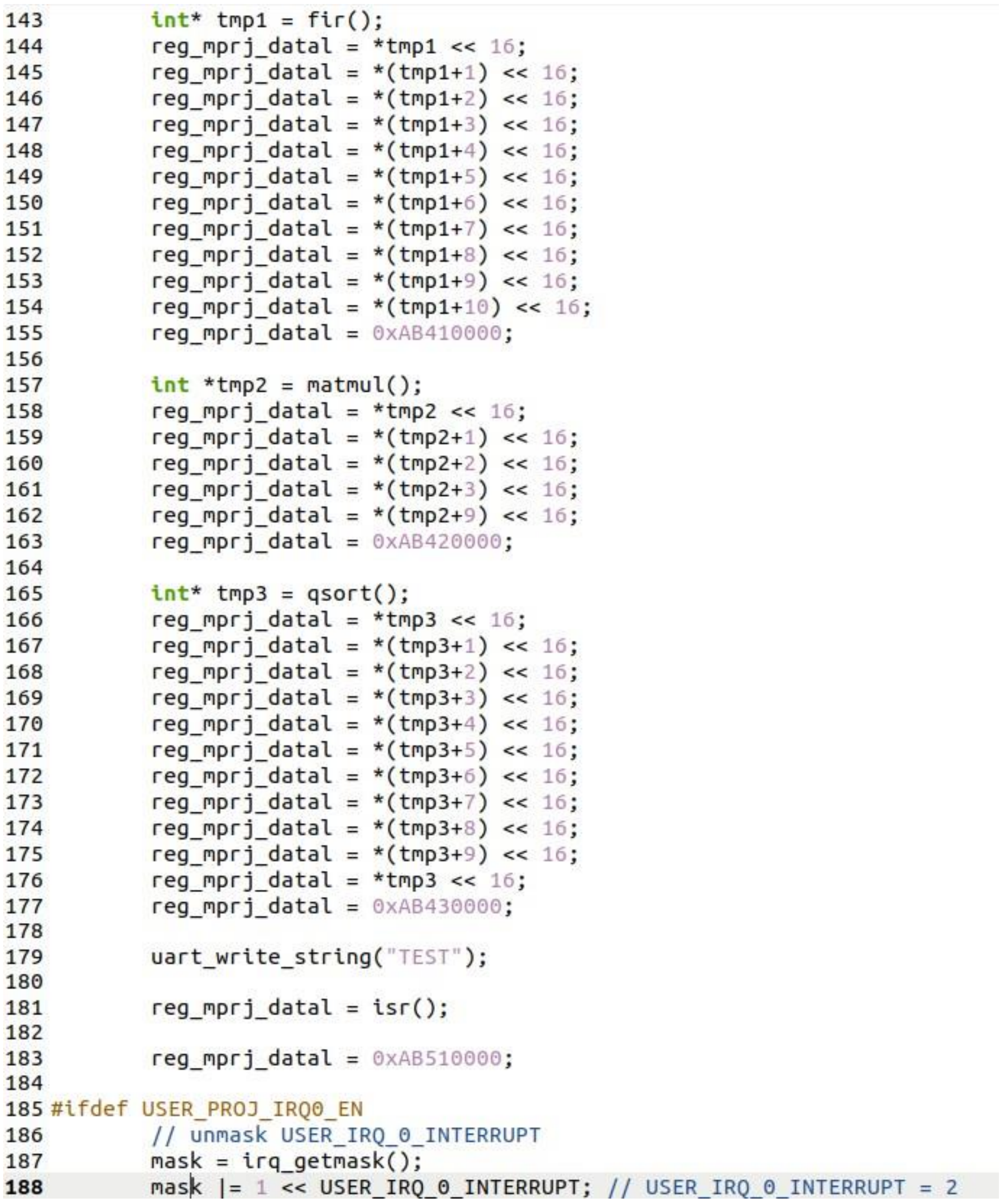
Firmware:

fir 結束位置是 0XAB410000

matmul 結束位置是 0XAB420000

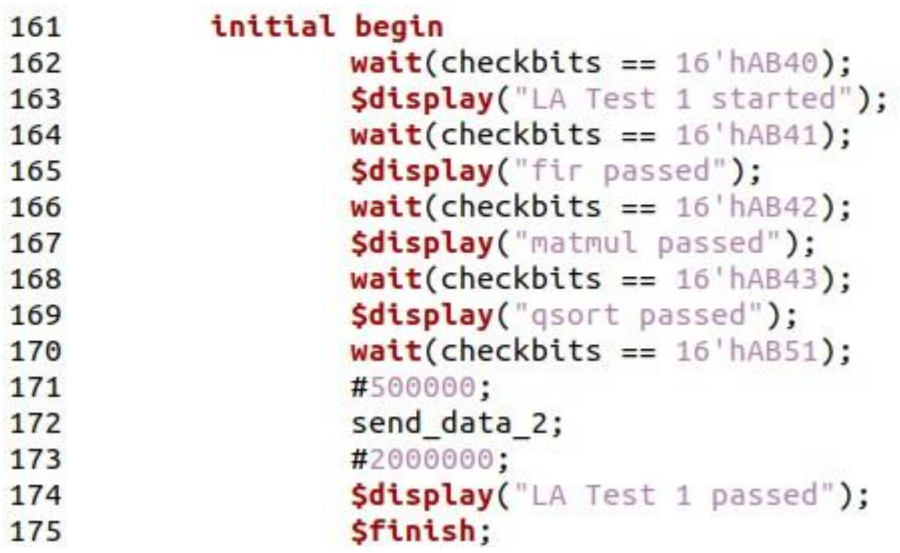
qsort 結束位置是 0XAB430000

UART 結束位置是 0XAB510000



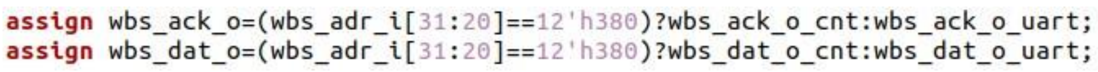
```
        int* tmp1 = fir();
        reg_mprj_datal = *tmp1 << 16;
        reg_mprj_datal = *(tmp1+1) << 16;
        reg_mprj_datal = *(tmp1+2) << 16;
        reg_mprj_datal = *(tmp1+3) << 16;
        reg_mprj_datal = *(tmp1+4) << 16;
        reg_mprj_datal = *(tmp1+5) << 16;
        reg_mprj_datal = *(tmp1+6) << 16;
        reg_mprj_datal = *(tmp1+7) << 16;
        reg_mprj_datal = *(tmp1+8) << 16;
        reg_mprj_datal = *(tmp1+9) << 16;
        reg_mprj_datal = *(tmp1+10) << 16;
        reg_mprj_datal = 0xAB410000;

        int *tmp2 = matmul();
        reg_mprj_datal = *tmp2 << 16;
        reg_mprj_datal = *(tmp2+1) << 16;
        reg_mprj_datal = *(tmp2+2) << 16;
        reg_mprj_datal = *(tmp2+3) << 16;
        reg_mprj_datal = *(tmp2+9) << 16;
        reg_mprj_datal = 0xAB420000;

        int* tmp3 = qsort();
        reg_mprj_datal = *tmp3 << 16;
        reg_mprj_datal = *(tmp3+1) << 16;
        reg_mprj_datal = *(tmp3+2) << 16;
        reg_mprj_datal = *(tmp3+3) << 16;
        reg_mprj_datal = *(tmp3+4) << 16;
        reg_mprj_datal = *(tmp3+5) << 16;
        reg_mprj_datal = *(tmp3+6) << 16;
        reg_mprj_datal = *(tmp3+7) << 16;
        reg_mprj_datal = *(tmp3+8) << 16;
        reg_mprj_datal = *(tmp3+9) << 16;
        reg_mprj_datal = *tmp3 << 16;
        reg_mprj_datal = 0xAB430000;

        uart_write_string("TEST");

        reg_mprj_datal = isr();

        reg_mprj_datal = 0xAB510000;

#ifdef USER_PROJ_IRQ0_EN
        // unmask USER_IRQ_0_INTERRUPT
        mask = irq_getmask();
        mask |= 1 << USER_IRQ_0_INTERRUPT; // USER_IRQ_0_INTERRUPT = 2
```

Testbench:



```
        initial begin
                wait(checkbits == 16'hAB40);
                $display("LA Test 1 started");
                wait(checkbits == 16'hAB41);
                $display("fir passed");
                wait(checkbits == 16'hAB42);
                $display("matmul passed");
                wait(checkbits == 16'hAB43);
                $display("qsort passed");
                wait(checkbits == 16'hAB51);
                #500000;
                send_data_2;
                #2000000;
                $display("LA Test 1 passed");
                $finish;
```

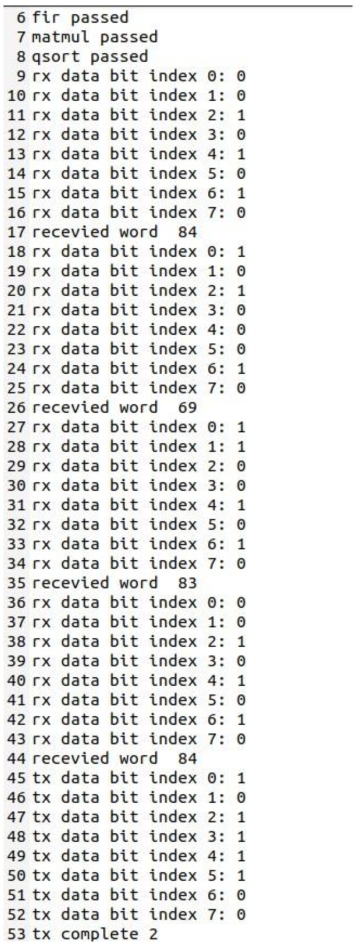
Hardware decode:



```
assign wbs_ack_o=(wbs_adr_i[31:20]==12'h380)?wbs_ack_o_cnt:wbs_ack_o_uart;
assign wbs_dat_o=(wbs_adr_i[31:20]==12'h380)?wbs_dat_o_cnt:wbs_dat_o_uart;
```

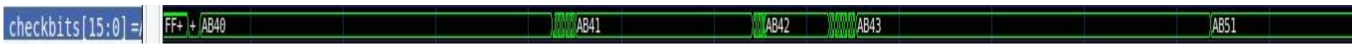
Intergrate_sim:

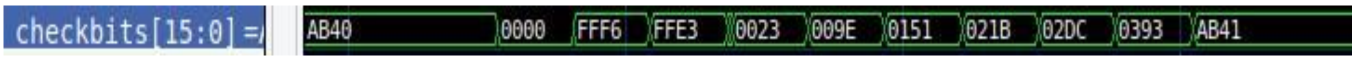
接收到數字 ASCII code: 84 代表英文字母 T, 69 是 E, 83 是 S。



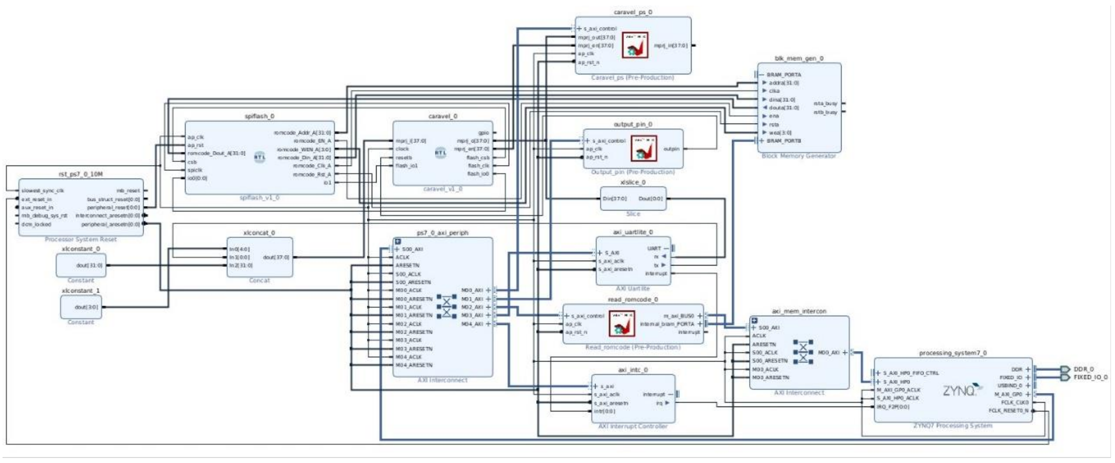
```
fir passed
matmul passed
qsort passed
rx data bit index 0: 0
rx data bit index 1: 0
rx data bit index 2: 1
rx data bit index 3: 0
rx data bit index 4: 1
rx data bit index 5: 0
rx data bit index 6: 1
rx data bit index 7: 0
recevied word  84
rx data bit index 0: 1
rx data bit index 1: 0
rx data bit index 2: 1
rx data bit index 3: 0
rx data bit index 4: 0
rx data bit index 5: 0
rx data bit index 6: 1
rx data bit index 7: 0
recevied word  69
rx data bit index 0: 1
rx data bit index 1: 1
rx data bit index 2: 0
rx data bit index 3: 0
rx data bit index 4: 1
rx data bit index 5: 0
rx data bit index 6: 1
rx data bit index 7: 0
recevied word  83
rx data bit index 0: 0
rx data bit index 1: 0
rx data bit index 2: 1
rx data bit index 3: 0
rx data bit index 4: 1
rx data bit index 5: 0
rx data bit index 6: 1
rx data bit index 7: 0
recevied word  84
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
```

Checkbits:


checkbits[15:0] =
FF+
+
AB40
AB41
AB42
AB43
AB51

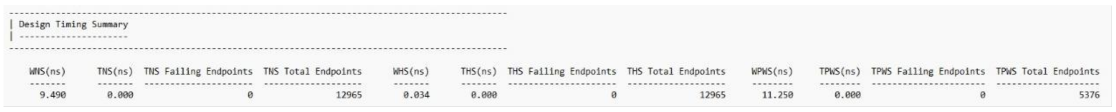

checkbits[15:0] =
AB40
0000
FFF6
FFE3
0023
009E
0151
021B
02DC
0393
AB41

•Block design



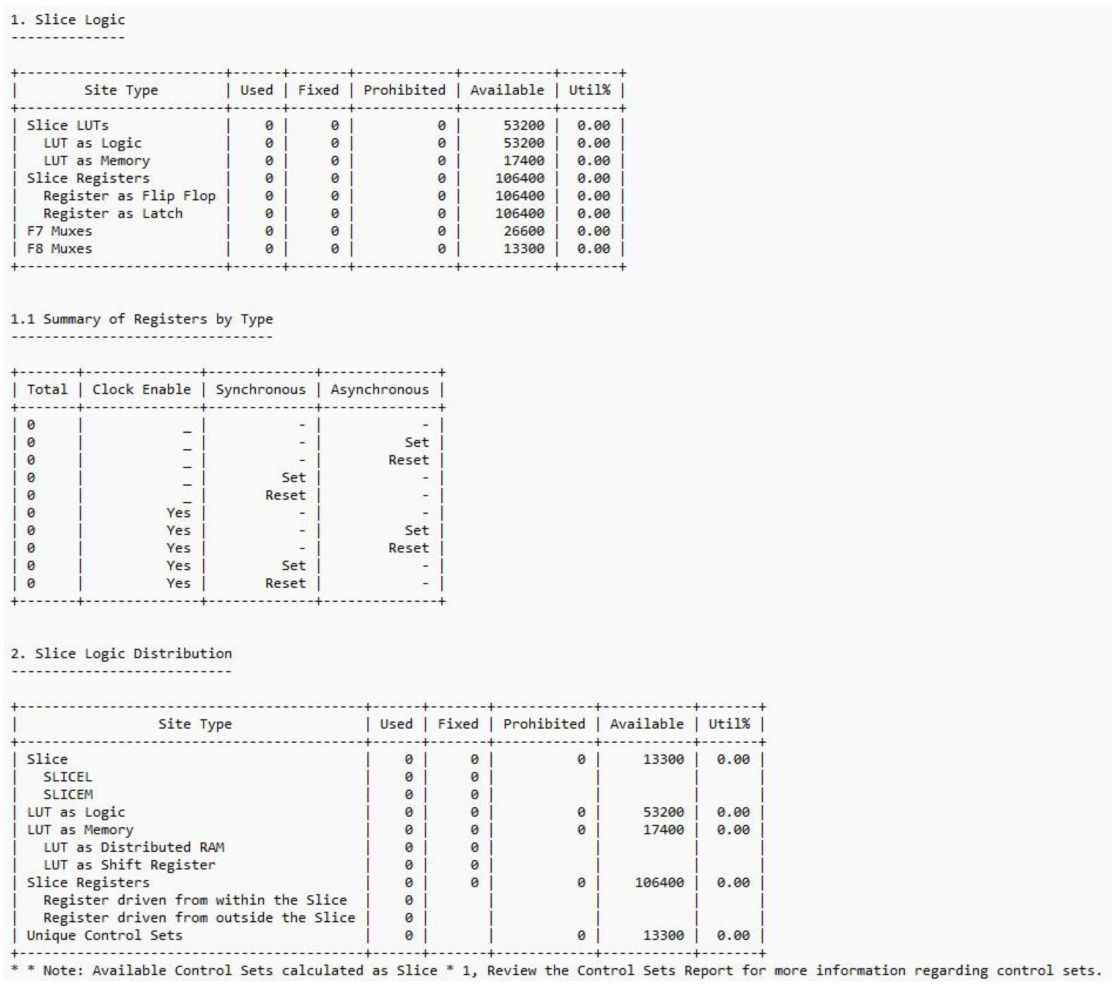
•Timing report/ resource report after synthesis

Timing report: Slack ≥ 0

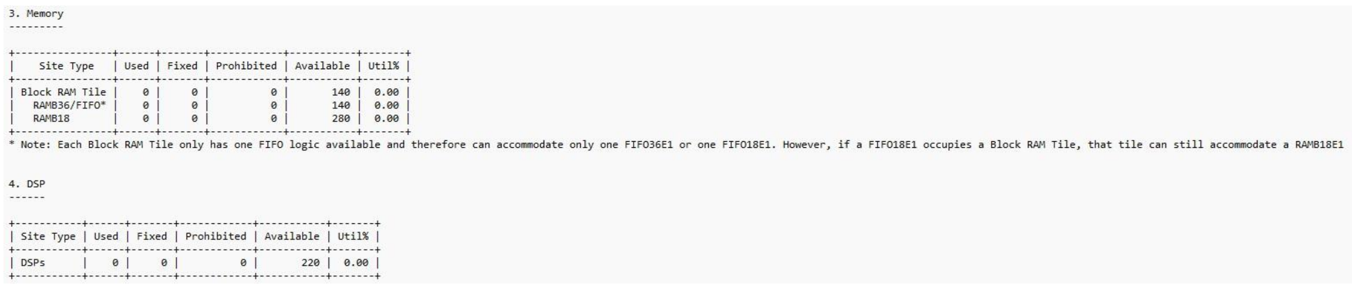


```
------------------------------------------------------------------------------------------------
| Design Timing Summary
| ---------------------
------------------------------------------------------------------------------------------------

    WNS(ns)      TNS(ns)  TNS Failing Endpoints  TNS Total Endpoints      WHS(ns)      THS(ns)  THS Failing Endpoints  THS Total Endpoints     WPWS(ns)     TPWS(ns)  TPWS Failing Endpoints  TPWS Total Endpoints
    -------      -------  ---------------------  -------------------      -------      -------  ---------------------  -------------------     --------     --------  ----------------------  --------------------
      9.490        0.000                      0                12965        0.034        0.000                      0                12965       11.250        0.000                       0                  5376
```

Utilization:



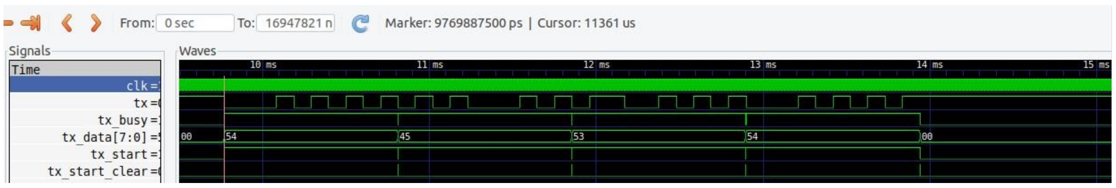
```
1. Slice Logic
--------------

+-------------------------+------+-------+------------+-----------+-------+
|        Site Type        | Used | Fixed | Prohibited | Available | Util% |
+-------------------------+------+-------+------------+-----------+-------+
| Slice LUTs              |    0 |     0 |          0 |     53200 |  0.00 |
|   LUT as Logic          |    0 |     0 |          0 |     53200 |  0.00 |
|   LUT as Memory         |    0 |     0 |          0 |     17400 |  0.00 |
| Slice Registers         |    0 |     0 |          0 |    106400 |  0.00 |
|   Register as Flip Flop |    0 |     0 |          0 |    106400 |  0.00 |
|   Register as Latch     |    0 |     0 |          0 |    106400 |  0.00 |
| F7 Muxes                |    0 |     0 |          0 |     26600 |  0.00 |
| F8 Muxes                |    0 |     0 |          0 |     13300 |  0.00 |
+-------------------------+------+-------+------------+-----------+-------+


1.1 Summary of Registers by Type
--------------------------------

+-------+--------------+-------------+--------------+
| Total | Clock Enable | Synchronous | Asynchronous |
+-------+--------------+-------------+--------------+
| 0     |            _ |           - |            - |
| 0     |            _ |           - |          Set |
| 0     |            _ |           - |        Reset |
| 0     |            _ |         Set |            - |
| 0     |            _ |       Reset |            - |
| 0     |          Yes |           - |            - |
| 0     |          Yes |           - |          Set |
| 0     |          Yes |           - |        Reset |
| 0     |          Yes |         Set |            - |
| 0     |          Yes |       Reset |            - |
+-------+--------------+-------------+--------------+


2. Slice Logic Distribution
---------------------------

+--------------------------------------------+------+-------+------------+-----------+-------+
|                  Site Type                 | Used | Fixed | Prohibited | Available | Util% |
+--------------------------------------------+------+-------+------------+-----------+-------+
| Slice                                      |    0 |     0 |          0 |     13300 |  0.00 |
|   SLICEL                                   |    0 |     0 |            |           |       |
|   SLICEM                                   |    0 |     0 |            |           |       |
| LUT as Logic                               |    0 |     0 |          0 |     53200 |  0.00 |
| LUT as Memory                              |    0 |     0 |          0 |     17400 |  0.00 |
|   LUT as Distributed RAM                   |    0 |     0 |            |           |       |
|   LUT as Shift Register                    |    0 |     0 |            |           |       |
| Slice Registers                            |    0 |     0 |          0 |    106400 |  0.00 |
|   Register driven from within the Slice    |    0 |       |            |           |       |
|   Register driven from outside the Slice   |    0 |       |            |           |       |
| Unique Control Sets                        |    0 |       |          0 |     13300 |  0.00 |
+--------------------------------------------+------+-------+------------+-----------+-------+
* * Note: Available Control Sets calculated as Slice * 1, Review the Control Sets Report for more information regarding control sets.
```

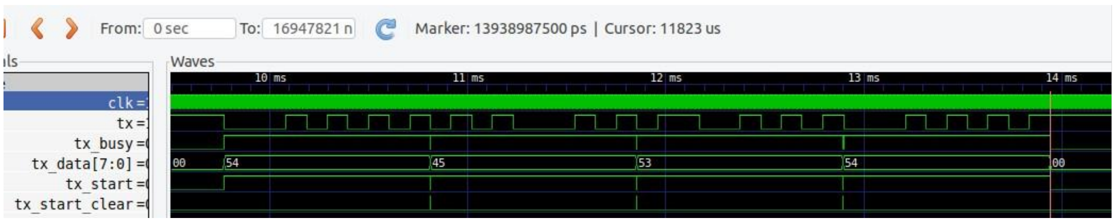


```
3. Memory
---------

+----------------+------+-------+------------+-----------+-------+
|    Site Type   | Used | Fixed | Prohibited | Available | Util% |
+----------------+------+-------+------------+-----------+-------+
| Block RAM Tile |    0 |     0 |          0 |       140 |  0.00 |
|   RAMB36/FIFO* |    0 |     0 |          0 |       140 |  0.00 |
|   RAMB18       |    0 |     0 |          0 |       280 |  0.00 |
+----------------+------+-------+------------+-----------+-------+
* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM Tile, that tile can still accommodate a RAMB18E1


4. DSP
------

+-----------+------+-------+------------+-----------+-------+
| Site Type | Used | Fixed | Prohibited | Available | Util% |
+-----------+------+-------+------------+-----------+-------+
| DSPs      |    0 |     0 |          0 |       220 |  0.00 |
+-----------+------+-------+------------+-----------+-------+
```

•Latency for a character loop back using UART

Latency=13938987500-9769887500(ps)= 4169100000(ps)=4.1691(ms)


From: 0 sec To: 16947821 n Marker: 9769887500 ps | Cursor: 11361 us


From: 0 sec To: 16947821 n Marker: 13938987500 ps | Cursor: 11823 us

•Suggestion for improving latency for UART loop back

要改善 latency 必須優化發射端與接收端的通訊過程，方法如下:

1.Baud Rate Optimization:

較高的波特率通常能夠降低傳輸一個字符所需的時間。

2. Hardware FIFO Buffers:

FIFO 緩衝區允許在單次操作中發送或接收多個字符，從而減少起始位和停止位的開銷。

3.DMA:

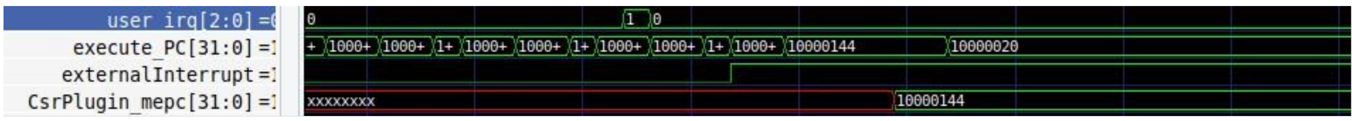
DMA 允許在內存和 UART 之間進行數據傳輸，無需 CPU 介入，從而減少延遲。

4. Interrupt-Driven Communication:

使用 Interrupt-Driven Communication 而不是使用 polling(輪詢)的方式，可以減少 CPU 不斷確認 UART 狀態所花的資源，同時也可以讓 CPU 可以在等待資料時去處理其他資料。

•What else do you observe

Uart_interrupt: 中斷的時候會記錄 mepc



Github Link:

https://github.com/shengxsheng/SOC-
Design/tree/1dcf826c4f4955e0be29050c57953793c7a8dd39/112061533_lab6