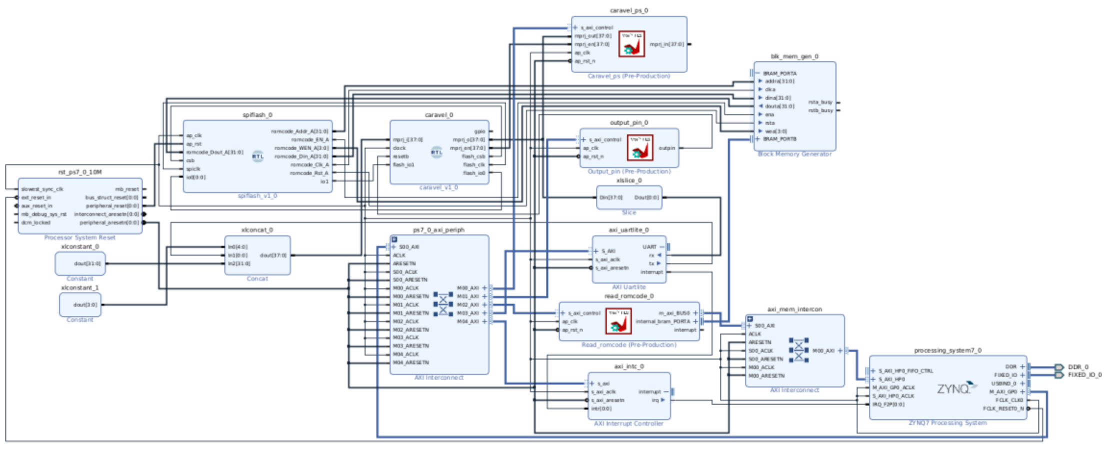
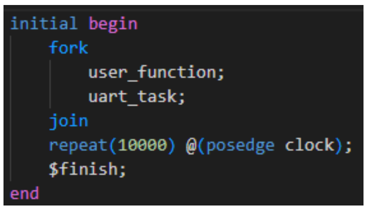
# Lab6 Report

## 1. Block design

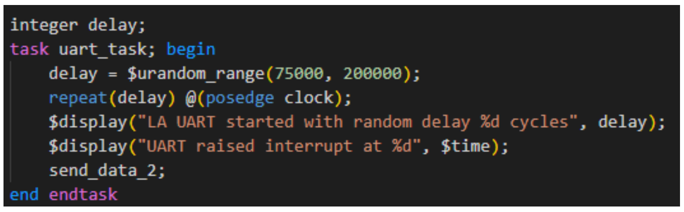


## 2. Concurrent execution in testbench and firmware code

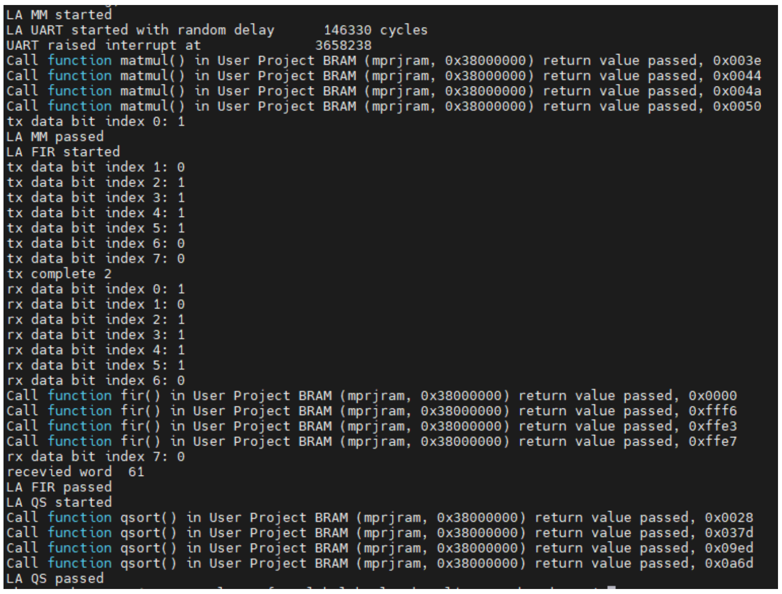
We define two tasks, one is user-function which is doing fir, quick sort and matrix multiplication tasks, the other is uart-task which is to sending interrupt signal while executing other three tasks in any time. Hence, we add a random delay to make this realize.



```
initial begin
    fork
        user_function;
        uart_task;
    join
    repeat(10000) @(posedge clock);
    $finish;
end
```



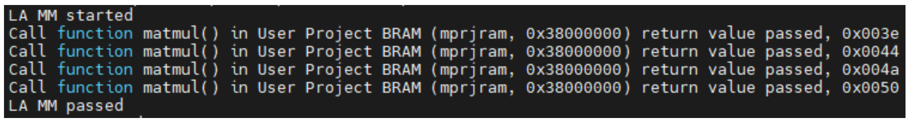
```
integer delay;
task uart_task; begin
    delay = $urandom_range(75000, 200000);
    repeat(delay) @(posedge clock);
    $display("LA UART started with random delay %d cycles", delay);
    $display("UART raised interrupt at %d", $time);
    send_data_2;
end endtask
```



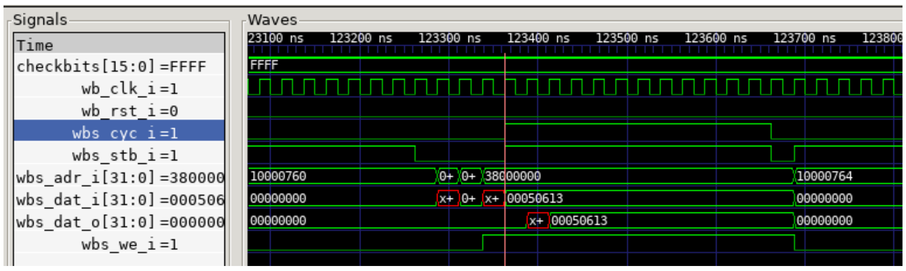
```
LA MM started
LA UART started with random delay      146330 cycles
UART raised interrupt at             3658238
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x003e
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0044
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
tx data bit index 0: 1
LA MM passed
LA FIR started
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
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0000
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, 0xfff6
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, 0xffe3
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, 0xffe7
rx data bit index 7: 0
recevied word   61
LA FIR passed
LA QS started
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0028
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x037d
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x09ed
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0a6d
LA QS passed
```

## 3. Firmware execution:

### 1. Matrix multiplication:

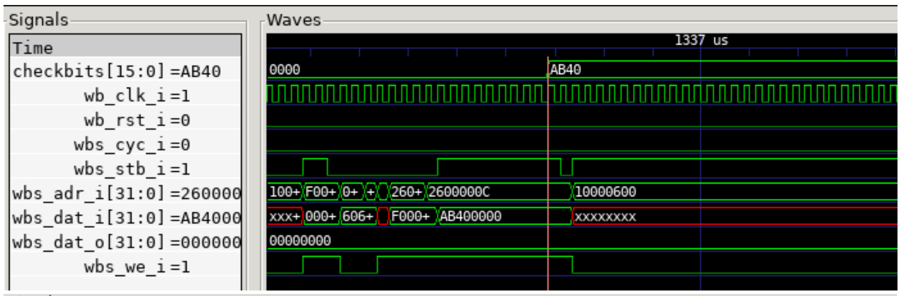


```
LA MM started
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x003e
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0044
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x004a
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0050
LA MM passed
```



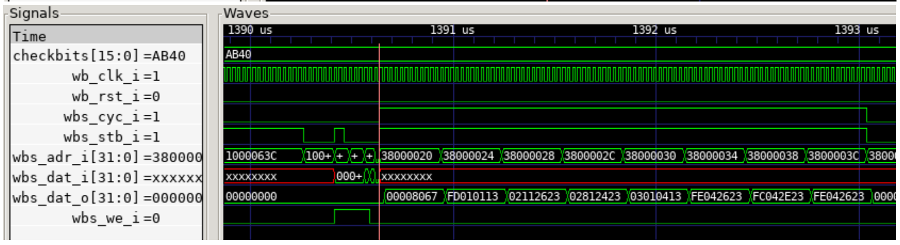
```
Time
checkbits[15:0] =FFFF
wb_clk_i =1
wb_rst_i =0
wbs_cyc_i =1
wbs_stb_i =1
wbs_adr_i[31:0] =380000
wbs_dat_i[31:0] =000506
wbs_dat_o[31:0] =000000
wbs_we_i =1
```

Signals
Waves
23100 ns 123200 ns 123300 ns 123400 ns 123500 ns 123600 ns 123700 ns 123800
FFFF
10000760 0+ 0+ 38000000 10000764
00000000 x+ 0+ x+ 00050613 00000000
00000000 x+ 00050613 00000000



```
Time
checkbits[15:0] =AB40
wb_clk_i =1
wb_rst_i =0
wbs_cyc_i =0
wbs_stb_i =1
wbs_adr_i[31:0] =260000
wbs_dat_i[31:0] =AB4000
wbs_dat_o[31:0] =000000
wbs_we_i =1
```

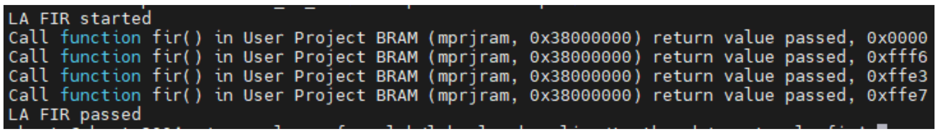
Signals
Waves
1337 us
0000 AB40
100+ F00+ 0+ + 260+ 2600000C 10000600
xxx+ 000+ 606+ F000+ AB400000 xxxxxxxx
00000000



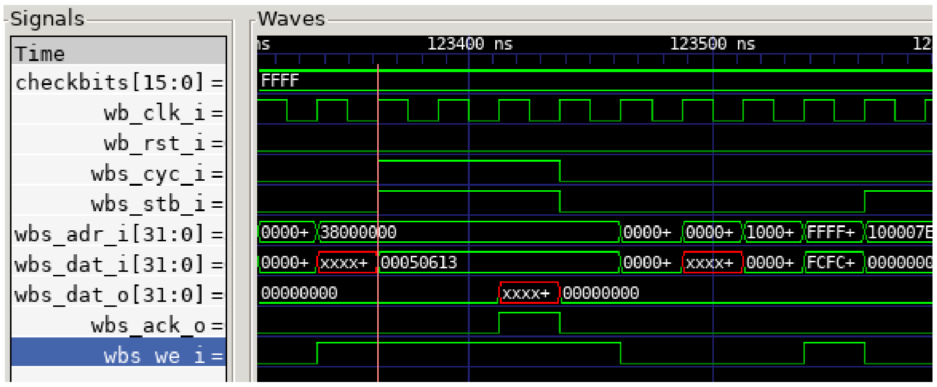
```
Time
checkbits[15:0] =AB40
wb_clk_i =1
wb_rst_i =0
wbs_cyc_i =1
wbs_stb_i =1
wbs_adr_i[31:0] =380000
wbs_dat_i[31:0] =xxxxxx
wbs_dat_o[31:0] =000000
wbs_we_i =0
```

Signals
Waves
1390 us 1391 us 1392 us 1393 us
AB40
1000063C 100+ + + + 38000020 38000024 38000028 3800002C 38000030 38000034 38000038 3800003C 3800
xxxxxxxx 000+ xxxxxxxx
00000000 00008067 FD010113 02112623 02812423 03010413 FE042623 FC042E23 FE042623 000

### 2. FIR:

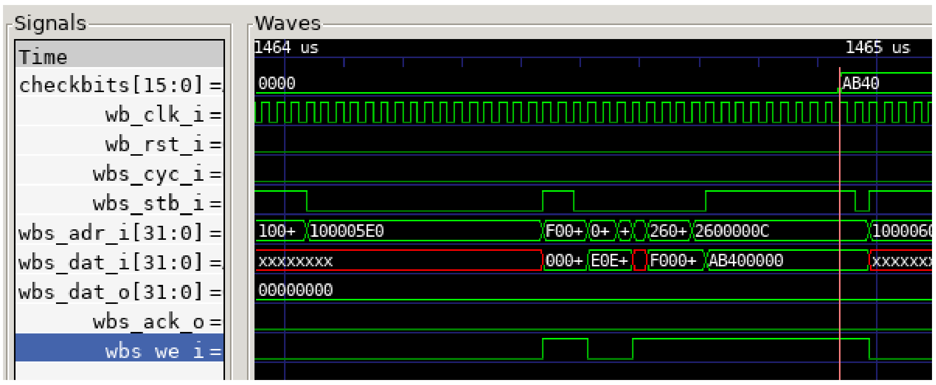


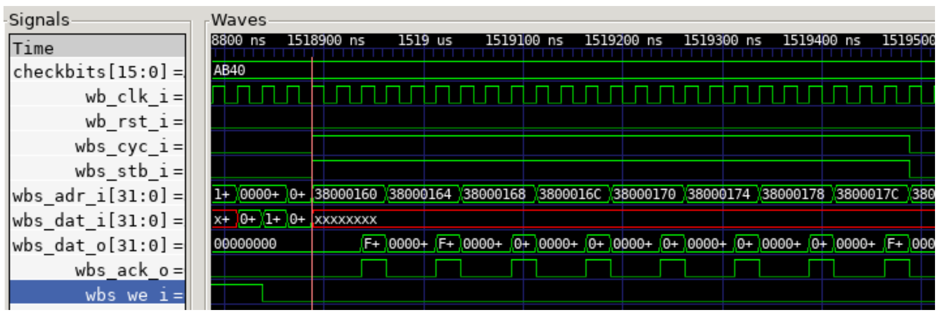
```
LA FIR started
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0000
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, 0xfff6
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, 0xffe3
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, 0xffe7
LA FIR passed
```



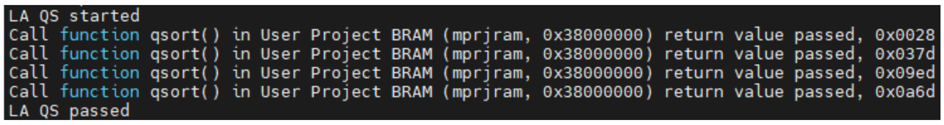
```
Time
checkbits[15:0] =
wb_clk_i =
wb_rst_i =
wbs_cyc_i =
wbs_stb_i =
wbs_adr_i[31:0] =
wbs_dat_i[31:0] =
wbs_dat_o[31:0] =
wbs_ack_o =
wbs_we_i =
```

Signals
Waves
123400 ns 123500 ns
FFFF
0000+ 38000000 0000+ 0000+ 1000+ FFFF+ 100007E
0000+ xxxx+ 00050613 0000+ xxxx+ 0000+ FCFC+ 0000000
00000000 xxxx+ 00000000

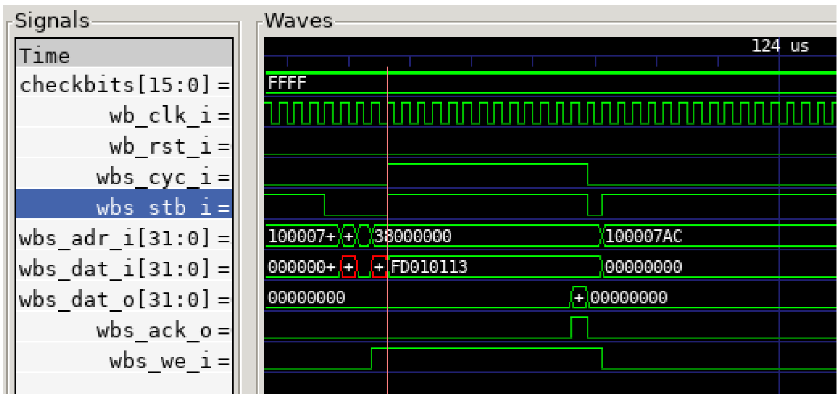


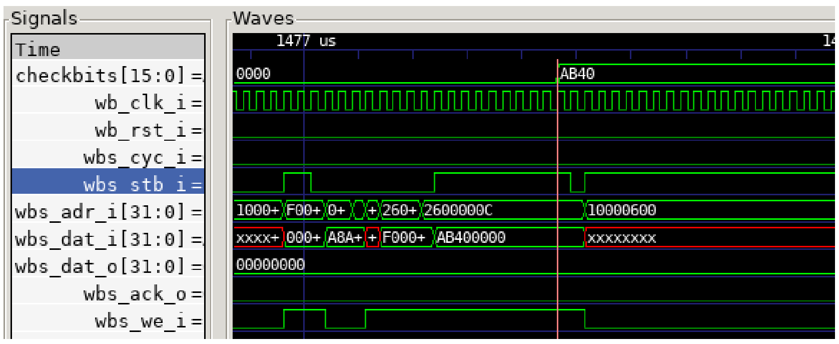


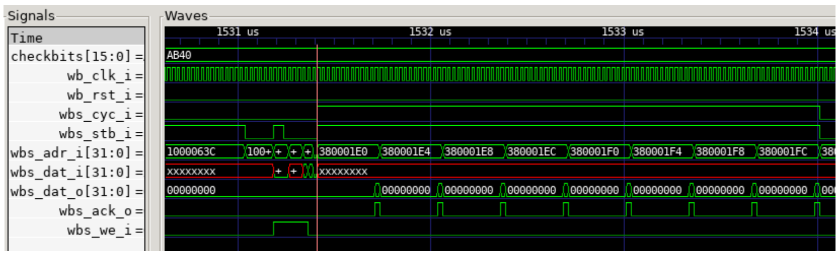
## 3. Quick sort:



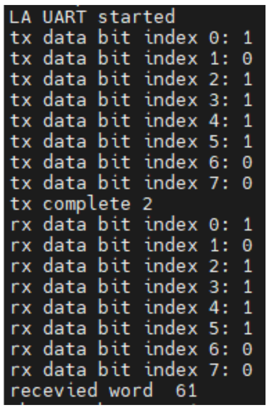
```
LA QS started
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0028
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x037d
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x09ed
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0a6d
LA QS passed
```



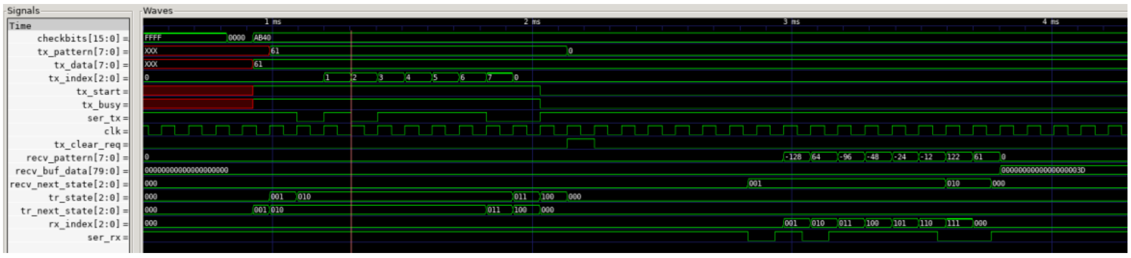




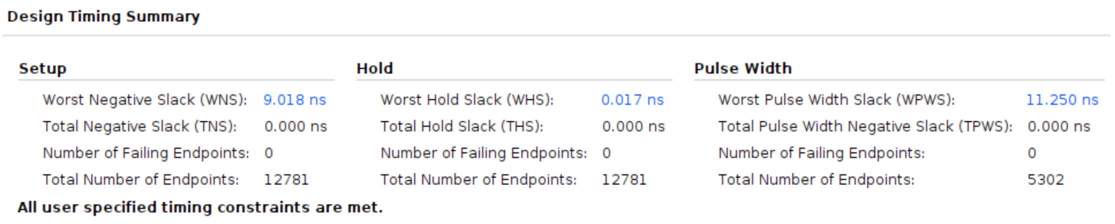
## 4. UART:



```
LA UART 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
```



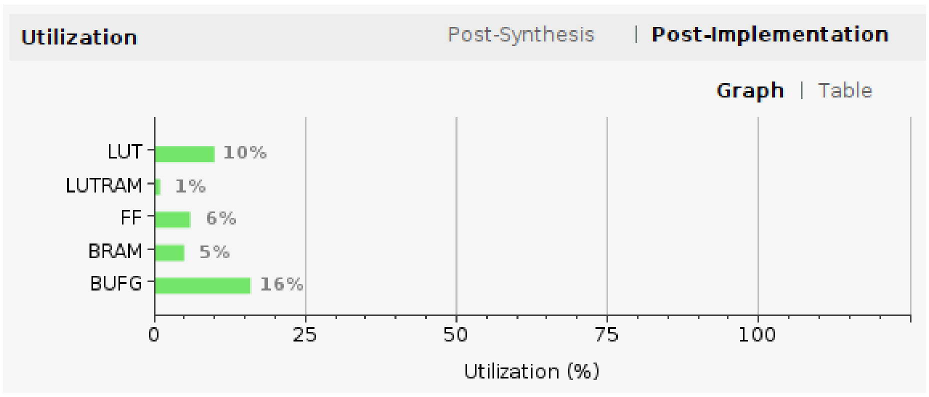
## 4. Timing report/ resource report after synthesis



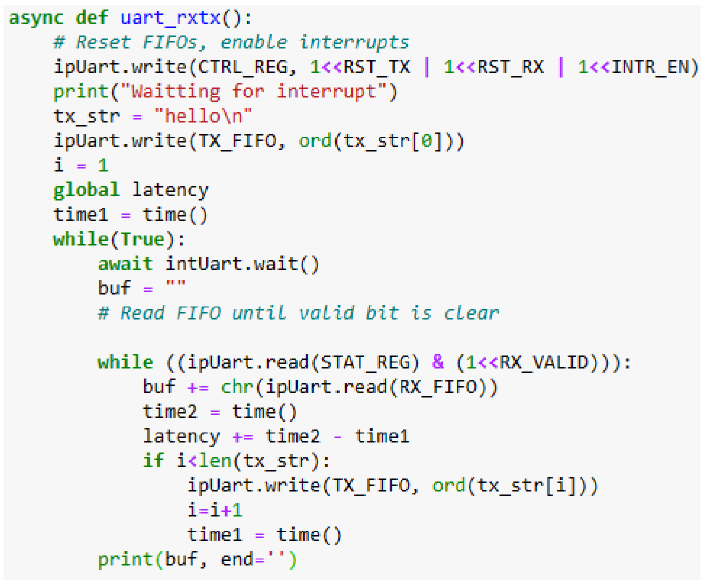
**Design Timing Summary**

| Setup | | Hold | | Pulse Width | |
|---|---|---|---|---|---|
| Worst Negative Slack (WNS): | 9.018 ns | Worst Hold Slack (WHS): | 0.017 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: | 12781 | Total Number of Endpoints: | 12781 | Total Number of Endpoints: | 5302 |

**All user specified timing constraints are met.**

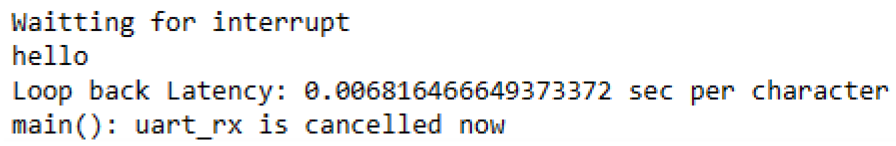


## 5. Latency for a character loop back using UART



```python
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
    global latency
    time1 = time()
    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))
            time2 = time()
            latency += time2 - time1
            if i<len(tx_str):
                ipUart.write(TX_FIFO, ord(tx_str[i]))
                i=i+1
                time1 = time()
        print(buf, end='')
```



```
Waitting for interrupt
hello
Loop back Latency: 0.006816466649373372 sec per character
main(): uart_rx is cancelled now
```

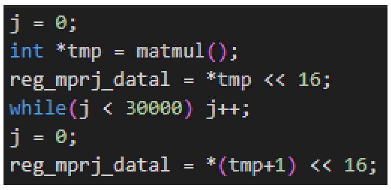
## 6. Suggestion for improving latency for UART loop back

Method1: We can add FIFOs as buffer to UART tasks. For example, if we add FIFO in the receive part of UART in caravel SoC, the hardware can support multiple tx transmission before rx reception. If we add FIFO into uartlite block, we can delay the interruption to PS until receiving multiple set of UART data. Both of these can reduce the overhead of UART transmission.

Method2: We can increase the Baud-Rate to improve the latency.

## 7. How do you verify your answer from notebook?

The speed for caravel FPGA executing the firmware code is too fast for PS to read back the output data through mprj_out[31:0] (0x1c). When PS want to check the answers, the values have been overwritten by the results of later operations. Hence, we manually slow down the output transmission by adding the while loop after the output is placed onto the mprj_out ports like this:



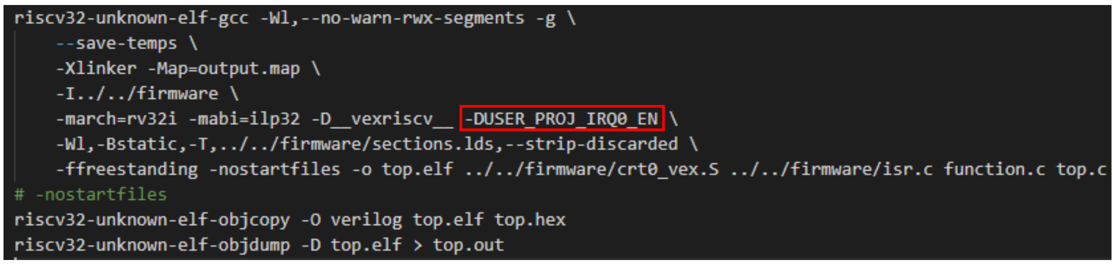
```c
j = 0;
int *tmp = matmul();
reg_mprj_datal = *tmp << 16;
while(j < 30000) j++;
j = 0;
reg_mprj_datal = *(tmp+1) << 16;
```

By doing so, PS has more time to access the data at the mprj_out ports to check the results.

## 8. What else do you observe

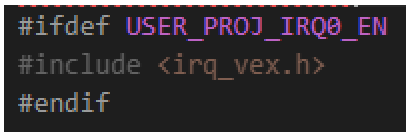
We notice that the mprj_io[37:32] ports are not used. If we can config those port as input ports to caravel SOC, we can use them as the control signals (acting like the ACK in Wishbone) that PS has received the result, which means we don't have to slow down the execution of the firmware code using while loop.

## 9. Problems we encountered

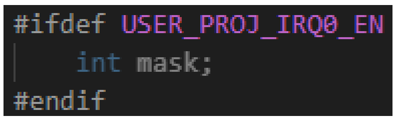


```
riscv32-unknown-elf-gcc -Wl,--no-warn-rwx-segments -g \
    --save-temps \
    -Xlinker -Map=output.map \
    -I../../firmware \
    -march=rv32i -mabi=ilp32 -D__vexriscv__ -DUSER_PROJ_IRQ0_EN \
    -Wl,-Bstatic,-T,../../firmware/sections.lds,--strip-discarded \
    -ffreestanding -nostartfiles -o top.elf ../../firmware/crt0_vex.S ../../firmware/isr.c function.c top.c
# -nostartfiles
riscv32-unknown-elf-objcopy -O verilog top.elf top.hex
riscv32-unknown-elf-objdump -D top.elf > top.out
```

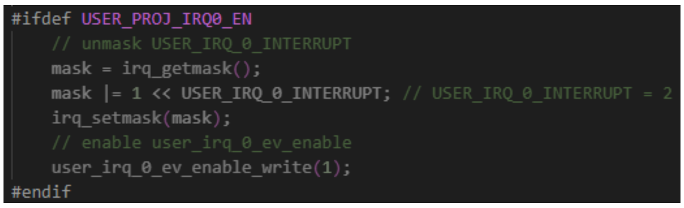
We notice that the UART task cannot act normally without defining this enable flag.



```
#ifdef USER_PROJ_IRQ0_EN
#include <irq_vex.h>
#endif
```



```
#ifdef USER_PROJ_IRQ0_EN
    int mask;
#endif
```



```
#ifdef USER_PROJ_IRQ0_EN
    // unmask USER_IRQ_0_INTERRUPT
    mask = irq_getmask();
    mask |= 1 << USER_IRQ_0_INTERRUPT; // USER_IRQ_0_INTERRUPT = 2
    irq_setmask(mask);
    // enable user_irq_0_ev_enable
    user_irq_0_ev_enable_write(1);
#endif
```

These instruction above serves as the important parts to run the UART task.