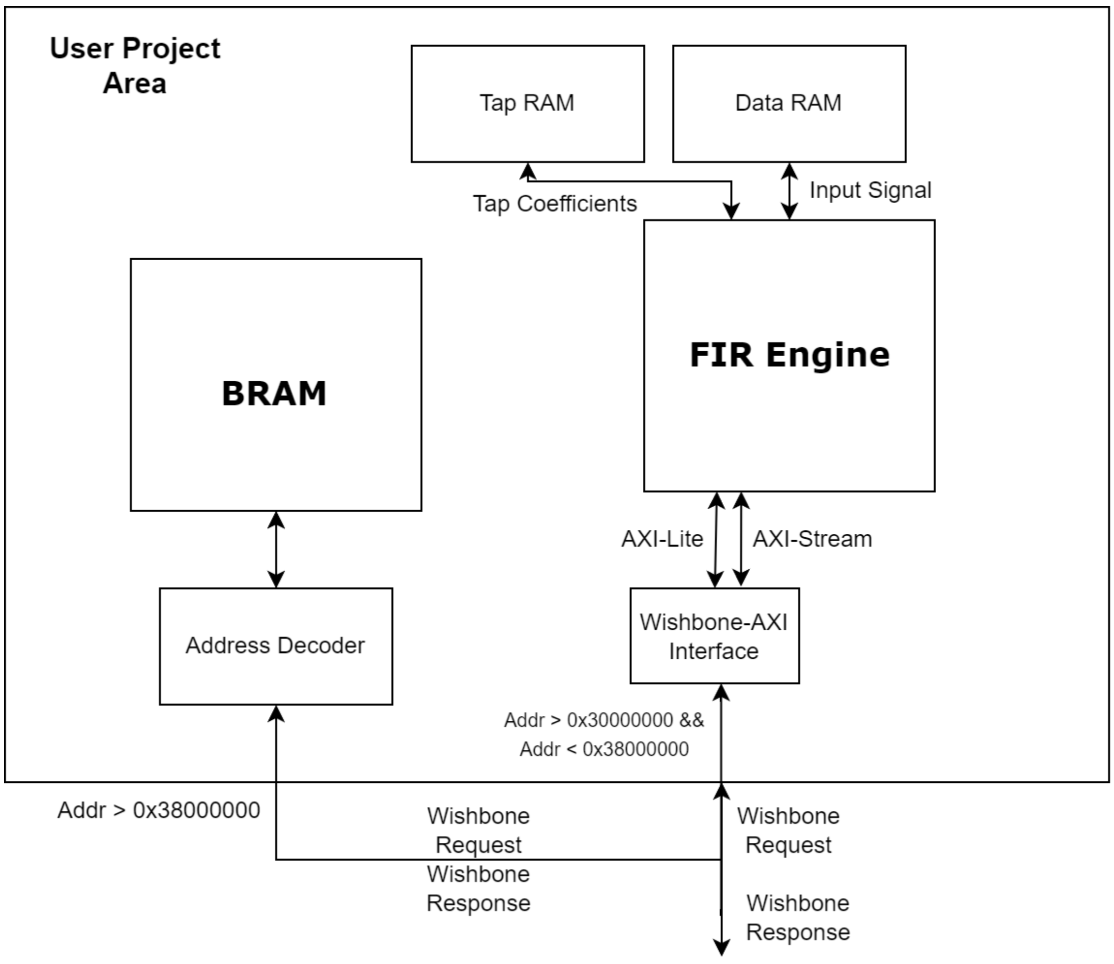
SOC lab4_1_report

r11921073 李丞峰

r11921075 江承祐

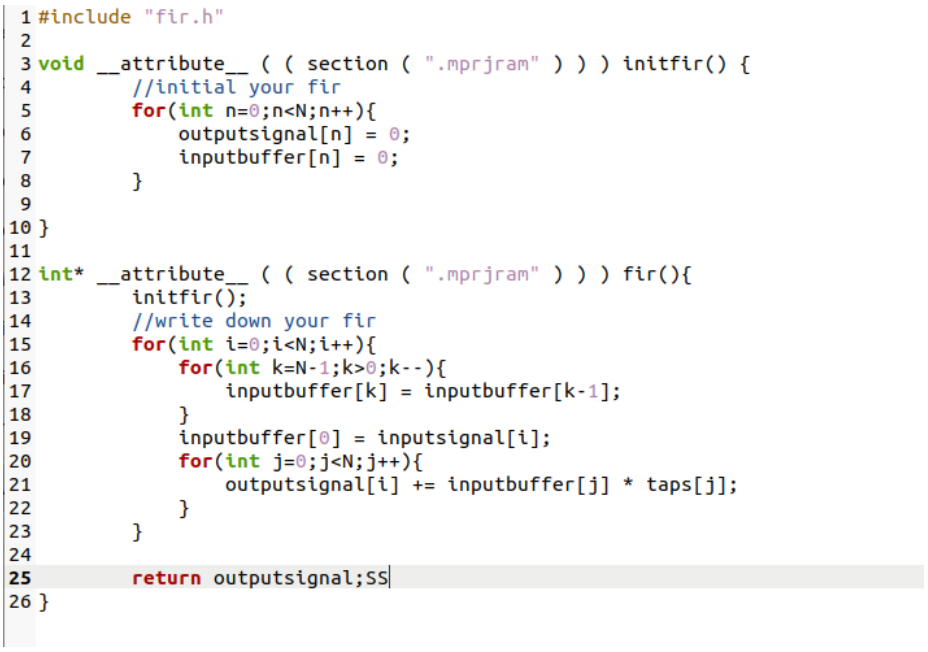
r12921006 黃芯柔

1. Block Diagram



2. The interface protocol between firmware, user project and testbench

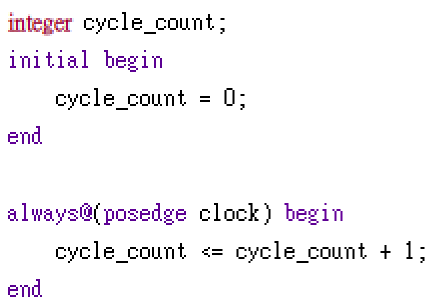
- **Firmware :**



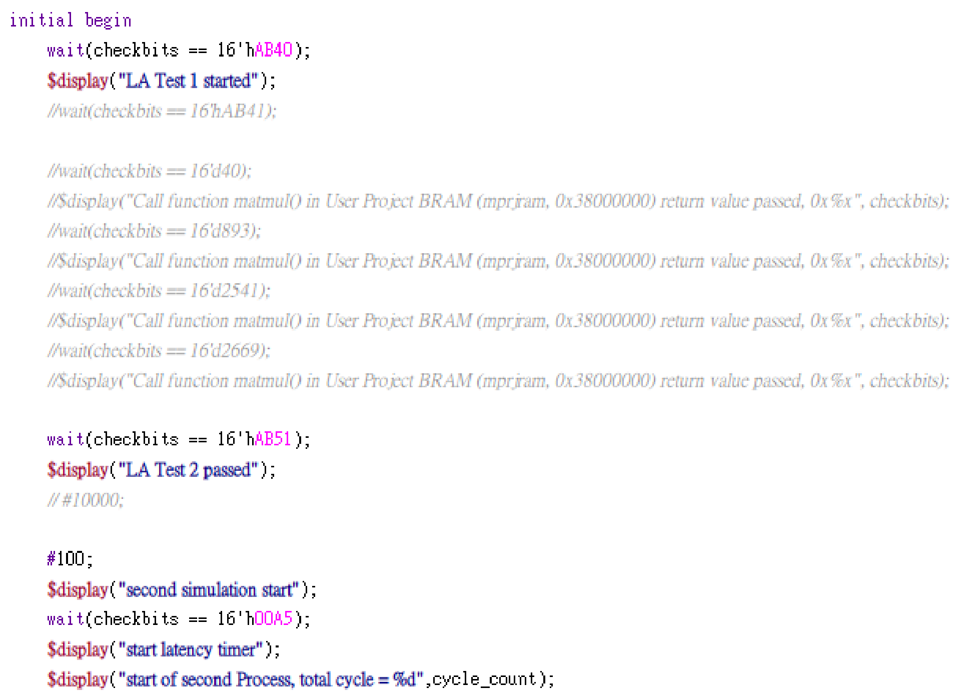
```c
#include "fir.h"

void __attribute__ ( ( section ( ".mprjram" ) ) ) initfir() {
        //initial your fir
        for(int n=0;n<N;n++){
            outputsignal[n] = 0;
            inputbuffer[n] = 0;
        }

}

int* __attribute__ ( ( section ( ".mprjram" ) ) ) fir(){
        initfir();
        //write down your fir
        for(int i=0;i<N;i++){
            for(int k=N-1;k>0;k--){
                inputbuffer[k] = inputbuffer[k-1];
            }
            inputbuffer[0] = inputsignal[i];
            for(int j=0;j<N;j++){
                outputsignal[i] += inputbuffer[j] * taps[j];
            }
        }

        return outputsignal;SS
}
```

首先 inifir()負責將原本位於 globle 記憶體的數值全部清為 0，接著每次大迴圈代表處裡一次 FIR 的數值，第一個小迴圈為讓 inputbuffer 的第 0 個 entry 空出來，並且讓原本的數值向右移一個 entry，接著將第 i 個 inputsignal 放到 inputbuffer 的第 0 個 entry。接著 ouputsignal 將會收取 inputbuffer 與 tap coefficient 乘加的結果。

- **Testbench:**



```verilog
integer cycle_count;
initial begin
    cycle_count = 0;
end

always@(posedge clock) begin
    cycle_count <= cycle_count + 1;
end
```

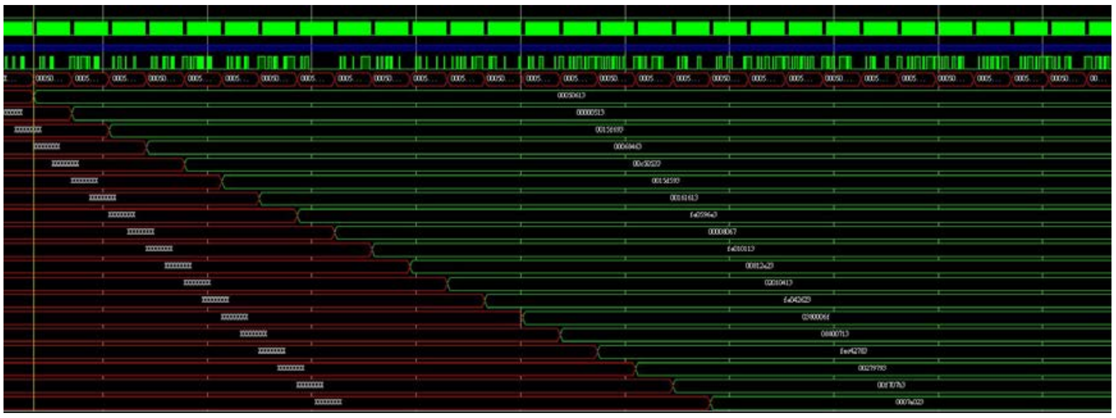


```
initial begin
    wait(checkbits == 16'hAB40);
    $display("LA Test 1 started");
    //wait(checkbits == 16'hAB41);

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

    #100;
    $display("second simulation start");
    wait(checkbits == 16'h00A5);
    $display("start latency timer");
    $display("start of second Process, total cycle = %d",cycle_count);
```

Testbench 的邏輯相對簡單，主要是設定 cycle count 來計時執行時間，以及透過 mprj 的輸出來檢驗 FIR 計算結果是否有誤，一開始會先進行 lab4-1 的模擬與測試，模擬內容與 lab4-1 的內容相同，都是先檢測 mprj = 16’hAB 作為開始，直到 mprj = 16’h51 時顯示模擬結束。

- **Waveform and analysis of the hardware/software behavior.**

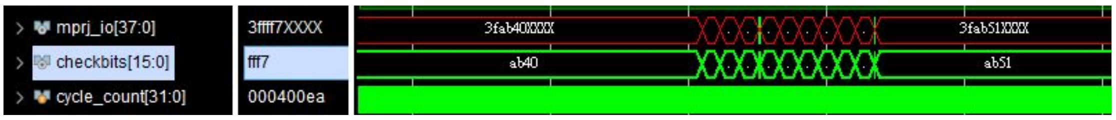
Spi flash to bram



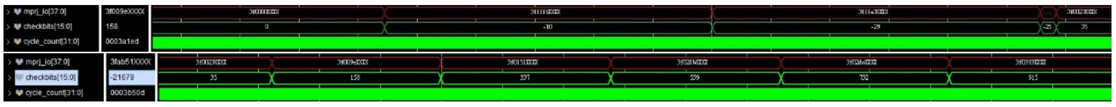
將 hex 從 SPI flash 讀入 BRAM。

Lab4-1

ab40 start、ab51 end

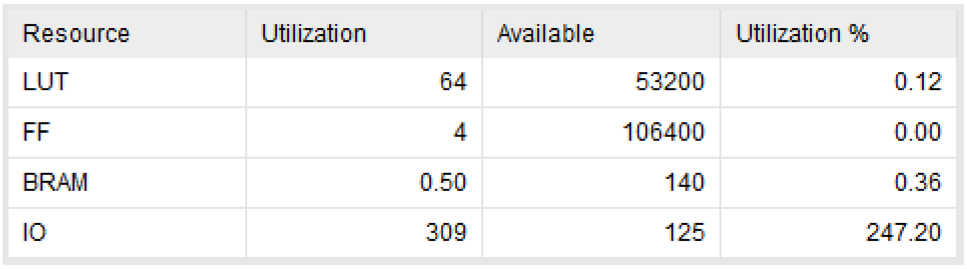


Mprj output fir.c



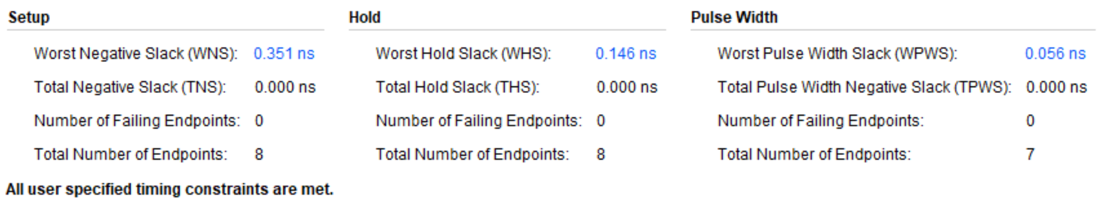
Testbench 分別偵測 checkbits 的輸出為 ab40 和 ab51 作為開始與結束。

3. Resource usage



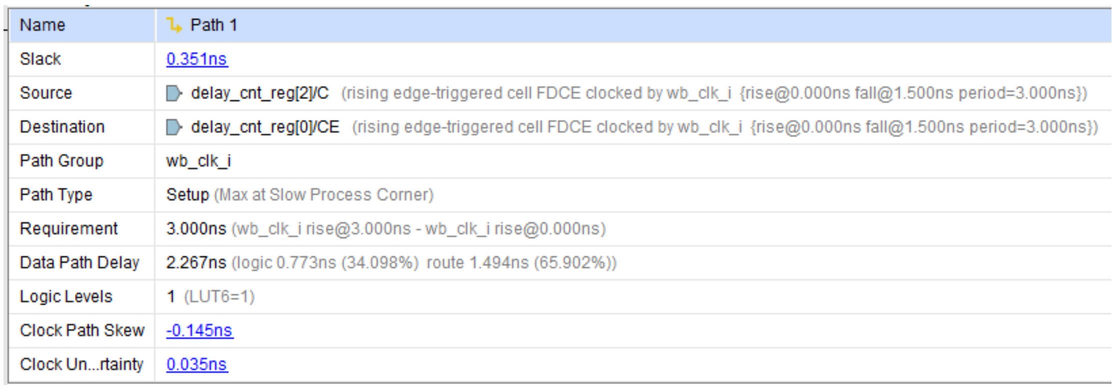
Resource	Utilization	Available	Utilization %
LUT	64	53200	0.12
FF	4	106400	0.00
BRAM	0.50	140	0.36
IO	309	125	247.20

4. Timing Report



Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.351 ns	Worst Hold Slack (WHS):	0.146 ns	Worst Pulse Width Slack (WPWS):	0.056 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:	8	Total Number of Endpoints:	8	Total Number of Endpoints:	7

All user specified timing constraints are met.



Name	Path 1
Slack	0.351ns
Source	delay_cnt_reg[2]/C (rising edge-triggered cell FDCE clocked by wb_clk_i {rise@0.000ns fall@1.500ns period=3.000ns})
Destination	delay_cnt_reg[0]/CE (rising edge-triggered cell FDCE clocked by wb_clk_i {rise@0.000ns fall@1.500ns period=3.000ns})
Path Group	wb_clk_i
Path Type	Setup (Max at Slow Process Corner)
Requirement	3.000ns (wb_clk_i rise@3.000ns - wb_clk_i rise@0.000ns)
Data Path Delay	2.267ns (logic 0.773ns (34.098%) route 1.494ns (65.902%))
Logic Levels	1 (LUT6=1)
Clock Path Skew	-0.145ns
Clock Un...rtainty	0.035ns