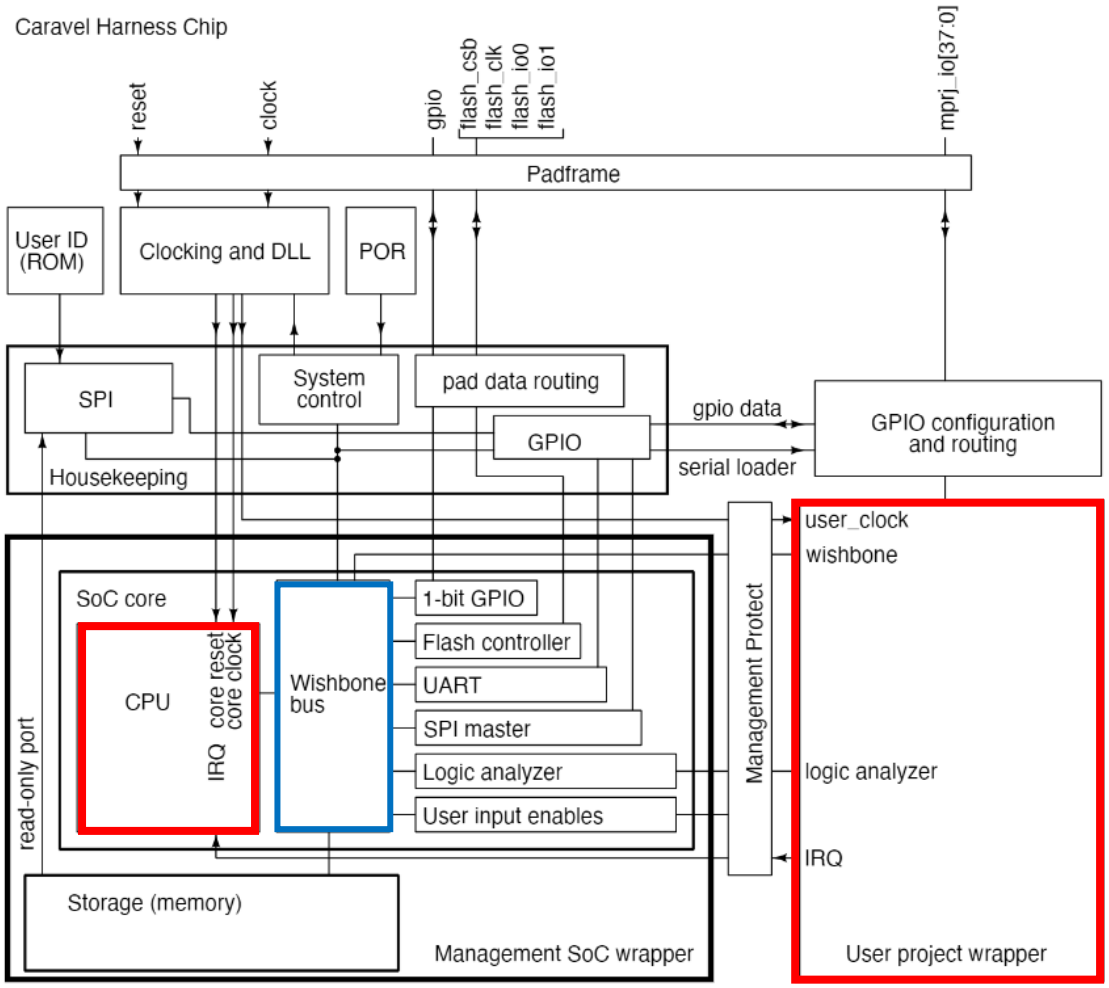
NYCU 電子研究所 系統晶片設計 LAB4-1

呂紹愷 311510187 江尹凡 311510207 廖智偉 311510216

- **Introduction**



此次的 LAB 目的為 CPU 以及 User project 間的溝通，我們要透過 wishbone bus 來把 CPU 上 compile 後的 fir.c machine code 透過 wishbone 傳給 User project 中的 BRAM 做儲存以及讀取，user 就可以依序讀出 bram 的這些 instruction 來執行，可以依據 user 用硬體或軟體來跑這些指令。

Required:

fir.c

fir.h

bram.v

user_proj_example.counter.v

Github link:

https://github.com/Ivan890129/Soc_lab04/tree/main/lab-exmem_fir

- **Explanation of your firmware code**

fir.c:

```c
#include "fir.h"

//#include <defs.h>
//#include <stub.c>
void __attribute__ ( ( section ( ".mprjram" ) ) ) initfir() {
    //initial your fir
    for(int i=0;i<N;i++){
        inputbuffer[i]=0;
        outputsignal[i]=0;
    }
}

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

此 C code 分為兩部分，第一部分為初始設定，將其初始值都歸零；第二部分則去實現 fir 的功能，input 總共有 N 個，依序將輸入訊號放到 inputbuffer 中，接著跑內部的 for 迴圈，去控制 taps 要怎麼做相乘並累加。

從 counter_la_fir.out 中可以看到 assembly code

```
38000024 <initfir>:
38000024: fe010113          addi  sp,sp,-32
38000028: 00812e23          sw  s0,28(sp)
3800002c: 02010413          addi  s0,sp,32
38000030: fe042623          sw  zero,-20(s0)
38000034: 0380006f          j 3800006c <initfir+0x48>
38000038: 05c00713          li  a4,92
3800003c: fec42783          lw  a5,-20(s0)
38000040: 00279793          slli  a5,a5,0x2
38000044: 00f707b3          add a5,a4,a5
38000048: 0007a023          sw  zero,0(a5)
3800004c: 08800713          li  a4,136
38000050: fec42783          lw  a5,-20(s0)
38000054: 00279793          slli  a5,a5,0x2
38000058: 00f707b3          add a5,a4,a5
3800005c: 0007a023          sw  zero,0(a5)
38000060: fec42783          lw  a5,-20(s0)
38000064: 00178793          addi  a5,a5,1
38000068: fef42623          sw  a5,-20(s0)
3800006c: fec42703          lw  a4,-20(s0)
38000070: 00a00793          li  a5,10
38000074: fce7d2e3          bge a5,a4,38000038 <initfir+0x14>
38000078: 00000013          nop
3800007c: 00000013          nop
38000080: 01c12403          lw  s0,28(sp)
38000084: 02010113          addi  sp,sp,32
38000088: 00008067          ret
```

上圖的部分對應到 fir.c 中的 initfir。

fir.c 中的 fir 部分，如下圖所示

```
3800008c <fir>:
3800008c: fe010113          addi  sp,sp,-32
38000090: 00112e23          sw  ra,28(sp)
38000094: 00812c23          sw  s0,24(sp)
38000098: 00912a23          sw  s1,20(sp)
3800009c: 02010413          addi  s0,sp,32
380000a0: f85ff0ef          jal ra,38000024 <initfir>
380000a4: fe042623          sw  zero,-20(s0)
380000a8: 0d40006f          j 3800017c <fir+0xf0>
380000ac: 02c00713          li  a4,44
380000b0: fec42783          lw  a5,-20(s0)
380000b4: 00279793          slli  a5,a5,0x2
380000b8: 00f707b3          add a5,a4,a5
380000bc: 0007a703          lw  a4,0(a5)
380000c0: 05c00693          li  a3,92
380000c4: fec42783          lw  a5,-20(s0)
380000c8: 00279793          slli  a5,a5,0x2
380000cc: 00f687b3          add a5,a3,a5
380000d0: 00e7a023          sw  a4,0(a5)
380000d4: fe042423          sw  zero,-24(s0)
380000d8: 08c0006f          j 38000164 <fir+0xd8>
380000dc: fec42703          lw  a4,-20(s0)
380000e0: fe842783          lw  a5,-24(s0)
380000e4: 40f707b3          sub a5,a4,a5
380000e8: 0607c863          bltz  a5,38000158 <fir+0xcc>
380000ec: 08800713          li  a4,136
380000f0: fec42783          lw  a5,-20(s0)
380000f4: 00279793          slli  a5,a5,0x2
380000f8: 00f707b3          add a5,a4,a5
380000fc: 0007a483          lw  s1,0(a5)
38000100: 00000713          li  a4,0
38000104: fe842783          lw  a5,-24(s0)
38000108: 00279793          slli  a5,a5,0x2
3800010c: 00f707b3          add a5,a4,a5
38000110: 0007a683          lw  a3,0(a5)
38000114: fec42703          lw  a4,-20(s0)
38000118: fe842783          lw  a5,-24(s0)
3800011c: 40f707b3          sub a5,a4,a5
38000120: 05c00713          li  a4,92
38000124: 00279793          slli  a5,a5,0x2
38000128: 00f707b3          add a5,a4,a5
3800012c: 0007a783          lw  a5,0(a5)
38000130: 00078593          mv  a1,a5
38000134: 00068513          mv  a0,a3
38000138: ec9ff0ef          jal ra,38000000 <__mulsi3>
3800013c: 00050793          mv  a5,a0
38000140: 00f48733          add a4,s1,a5
38000144: 08800693          li  a3,136
38000148: fec42783          lw  a5,-20(s0)
3800014c: 00279793          slli  a5,a5,0x2
38000150: 00f687b3          add a5,a3,a5
38000154: 00e7a023          sw  a4,0(a5)
38000158: fe842783          lw  a5,-24(s0)
3800015c: 00178793          addi  a5,a5,1
38000160: fef42423          sw  a5,-24(s0)
38000164: fe842703          lw  a4,-24(s0)
38000168: 00a00793          li  a5,10
3800016c: f6e7d8e3          bge a5,a4,380000dc <fir+0x50>
38000170: fec42783          lw  a5,-20(s0)
38000174: 00178793          addi  a5,a5,1
38000178: fef42623          sw  a5,-20(s0)
3800017c: fec42703          lw  a4,-20(s0)
38000180: 00a00793          li  a5,10
38000184: f2e7d4e3          bge a5,a4,380000ac <fir+0x20>
38000188: 08800793          li  a5,136
3800018c: 00078513          mv  a0,a5
38000190: 01c12083          lw  ra,28(sp)
38000194: 01812403          lw  s0,24(sp)
38000198: 01412483          lw  s1,20(sp)
3800019c: 02010113          addi  sp,sp,32
```

- **How does it execute a multiplication in assembly code?**
  使用__mulsi3 函數來做乘法運算

```
38000000 <__mulsi3>:
38000000: 00050613          mv  a2,a0
38000004: 00000513          li  a0,0
38000008: 0015f693          andi  a3,a1,1
3800000c: 00068463          beqz  a3,38000014 <__mulsi3+0x14>
38000010: 00c50533          add a0,a0,a2
38000014: 0015d593          srli  a1,a1,0x1
38000018: 00161613          slli  a2,a2,0x1
3800001c: fe0596e3          bnez  a1,38000008 <__mulsi3+0x8>
38000020: 00008067          ret
```

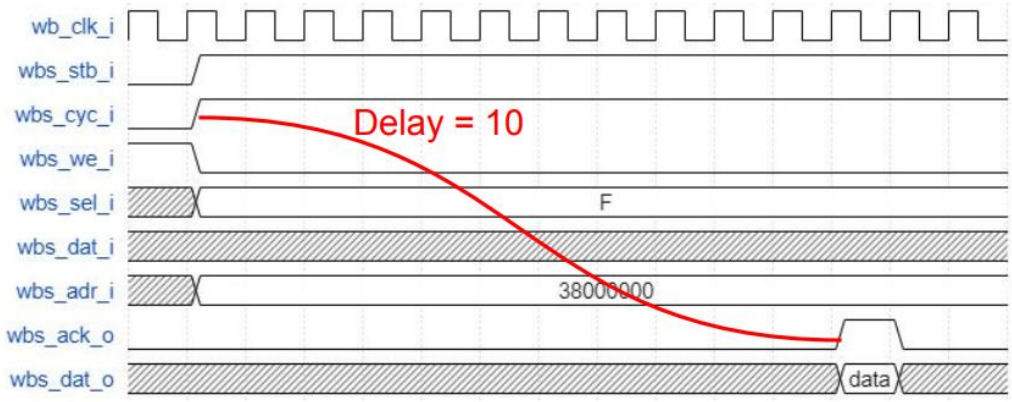
1. mv a2, a0：將暫存器 a0 的值複製到暫存器 a2。
2. li a0, 0：將暫存器 a0 的值設置為 0。
3. andi a3, a1, 1：將 a1 中的值和 1 進行按位與運算，結果存在 a3 中。
4. beqz a3, 38000014：如果 a3 的值為 0，則跳到記憶體地址 38000014 執行指令。
5. add a0, a0, a2：將 a2 加到 a0 上。
6. srli a1, a1, 0x1：將 a1 右移一位。
7. slli a2, a2, 0x1：將 a2 左移一位。
8. bnez a1, 38000008：如果 a1 不為零，則跳到記憶體地址 38000008 執行指令。
9. ret：返回。

- **What address allocate for user project and how many space is required to allocate to firmware code**

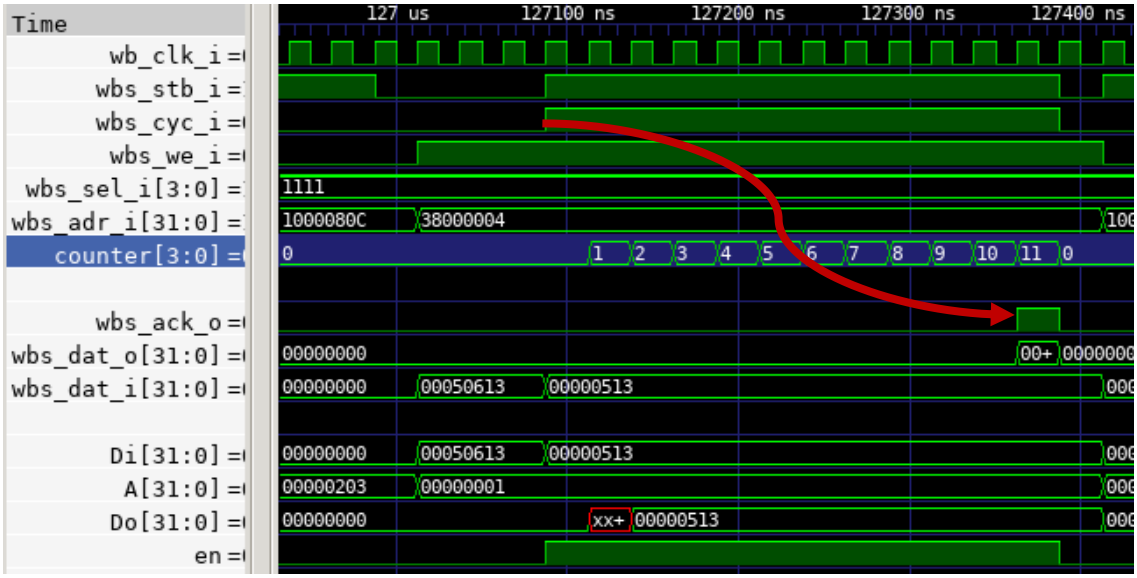
```
MEMORY {
    vexriscv_debug : ORIGIN = 0xf00f0000, LENGTH = 0x00000100
    dff : ORIGIN = 0x00000000, LENGTH = 0x00000400
    dff2 : ORIGIN = 0x00000400, LENGTH = 0x00000200
    flash : ORIGIN = 0x10000000, LENGTH = 0x01000000
    mprj : ORIGIN = 0x30000000, LENGTH = 0x00100000
    mprjram : ORIGIN = 0x38000000, LENGTH = 0x00400000
    hk : ORIGIN = 0x26000000, LENGTH = 0x00100000
    csr : ORIGIN = 0xf0000000, LENGTH = 0x00010000
}
```

BRAM 的範圍為 0x38000000 開始，而從 counter_la_fir.out 中可以觀察到一開始先處理<__mulsi3>從 0x38000000，接著是<initfir>，最後是<fir>，因此範圍是從 0x38000000~0x380001a0。

- **Interface between BRAM and wishbone**



Read BRAM




```
bram user_bram (
    .CLK(wb_clk_i),
    .WE0(wen),
    .EN0(en),
    .Di0(Di),
    .Do0(Do),
    .A0(A)
);
```

Wishbond 跟 Bram 的接口，wbs_stb_i、wbs_cyc_i 同時拉 1 且 wbs_adr_i 為 32'h38 開頭時，bram 的 en 拉 1，表示 bram 可以工作，同時 counter 也從 0 開始數，數到 10 (預設的 delay)，就把 wbs_ack 拉 1 同時輸出 wbs_dat_o。

Bram:

A ={10'd0, wbs_adr_i [23:2]}，把 wbs_adr_i 轉成對應的 bram 位置 (1 個 word 4bytes 所以[1:0]不用取最一開始的 8'h38 是 tag 也不用取)。

Di：直接對接 wbs_dat_i。

Do：直接對接 wbs_dat_o。

En : wbs_stb_i && wbs_cyc_i &&(wbs_adr_i[31:24]==8'h38)。