SOC Design Lab4-2 Report

National Yang Ming Chiao Tung University
311511022 邱政岡
412010020 高振翔
311514037 蔡宇冠

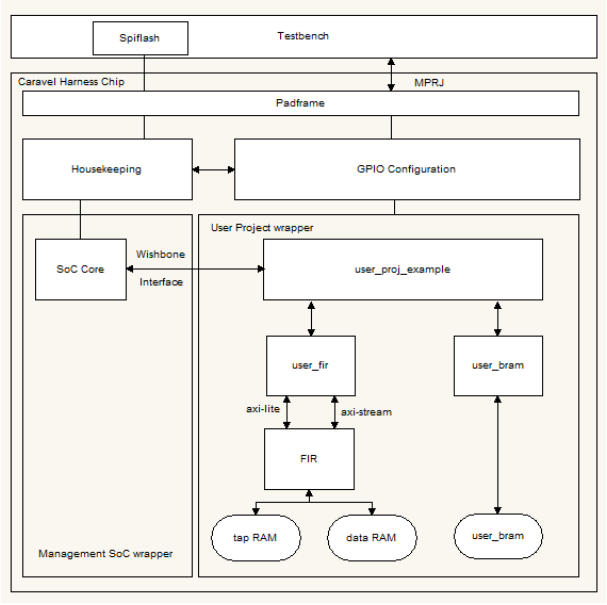
1. Introduction

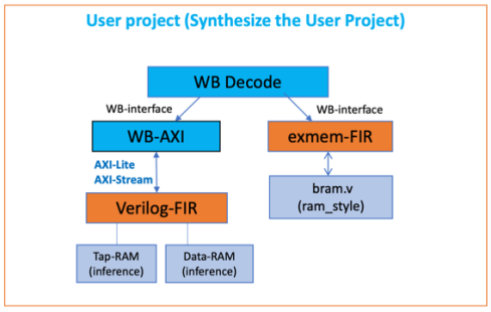
1.1 Objective

- Integrate the fir accelerator(lab3) into caraval user project area.
- Use the MMIO configuration to decode the destination between exmem and fir block.
- Interconnect the protocol wishbone and axi-lite/axi-stram.
- Program firmware code to set fir status and push in X data and pull Y data from fir engine.
- Program firmware code to set starMark and endMark to the testbench by mprj to calculate the latency.

1.2 Overview

In this lab, we integrate the lab3 fir accelerator into our user project. There are two parts to the design modified. The first part is that we integrate the tap ram to the fir design instead of the instance in the testbench. The other modified part adds two ap signals to indicate the X data is ready to receive and the Y data is ready to be read.





The testbench will instantiate a spiflash to supply the firmware code to RISC-V CPU, and receive data from Caravel SOC by mprj_io pin. The data transform between mprj_io, RISC-V CPU, and user project uses the wishbone protocol. The wishbone

interface is faster than the spi interface between RISC-V CPU and spiflash. So putting the firmware code to BRAM in user project has a smaller latency.

2. Firmware

We write the firmware code named fir_control.c. We merge the header file and c file and define the main function in the section(".mprjram).

```
#define MPRJ_RAM __attribute__ ( ( section ( ".mprjram" ) ) )
void MPRJ_RAM main() {
    ...
}
```

For the MMIO configuration of fir base address, we define these offset including ap status、data length、tap parameter、X[n] input、Y[n] output.

```
#define reg_fir_ctrl (*(volatile uint32_t*)0x30000000)
#define reg_fir_len  (*(volatile uint32_t*)0x30000010)
#define reg_fir_coef (*(volatile uint32_t*)0x30000040)
#define reg_fir_x    (*(volatile uint32_t*)0x30000080)
#define reg_fir_y    (*(volatile uint32_t*)0x30000084)
```

The same with lab3, it set the mprj_io initial config. Then we set the data length and tap the parameters. After this initial stage, it sets the start flag(startMark) to mprj_io. In the for loop of the code, it starts to wait x_ready signal to transfer X[n] and then wait for y_ready signal to transfer Y[n]. I should point out that in this design, we set the x_ready and y_ready always true in the FIR engine. Before the end of the for loop(data length=64), the firmware code will transfer Y[n] to testbench by mprj_io. At the end of the loop will send the end flag(endMark) to the testbench.

3. Hardware

3.1 Wishbone decoder

We implement the address decoder in the user_proj_example.v to distinguish the access destination from exmem to fir. The wishbone signal of cyc、ack、data_o are separate to two part according to the destination.

```
always @(*) begin
    sel[0] = (wbs_adr_i[31:24] == 'h30);    // FIR
    sel[1] = (wbs_adr_i[31:24] == 'h38);    // RAM
end
```

3.2 Interface between FIR and Wishbone

The interconnection between wishbone and axi bus is implemented in the fir_wb.v. The handshake separate to two part including axi-lite and axi-stream. Please

reference the handshake waveform in the report.

Four signals: *axi_w*, *axi_r, stream_w*, *stream_r* are used to determine which channel to access.

```
    assign valid = wbs_stb_i & wbs_cyc_i;
    assign sel = wbs_adr_i[7];  // 0->axi-lite, 1->axi-stream
    assign axi_w = ~sel & wbs_we_i;
    assign axi_r = ~sel & ~wbs_we_i;
    assign stream_w = sel & wbs_we_i;
    assign stream_r = sel & ~wbs_we_i;
```

3.3 Interface between BRAM and Wishbone

(Same as Lab4-1 and rename the module to bram_wb.)

4. Simulation

4.1 Testbench

In the testbench, the code will get startMark(0xA5) and calculate the latency between endMark(0x5A). We run three times of fir calculation. The latency of each round is about 1007325. After three times, the total latency is 3021975 ns.

4.2 Simulation result

```
ubuntu@ubuntu2004:~/SOC_Design/Lab4/Lab4-2/testbench/counter_la_fir$ ./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.
FIR Test 1 started
result=00
FIR Test 1 passed, Latency = 1007325 ns
result=76
FIR Test 2 started
result=00
FIR Test 2 passed, Latency = 1007325 ns
result=76
FIR Test 3 started
result=00
FIR Test 3 passed, Latency = 1007325 ns
Total Latency = 3021975 ns
result=76
```

4.3 Analysis

4.3.1 Throughput

The theoretical throughput is 1 output data every 11 cycles.

1 / 11 = 0.091 data/cycle

The actual measured throughput is 64 output data every 1007325 seconds.

1 clock period is 25 ns. One round of fir is 1007325/25 = 40293 cycle

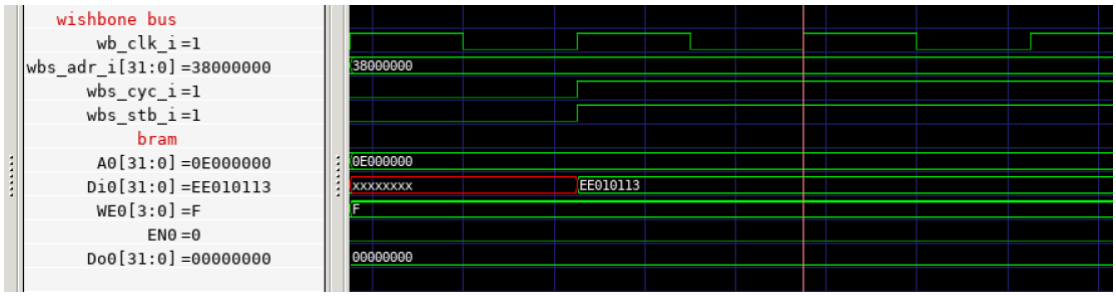
64 / 40293 = 0.0015884 data/cycle

5. Waveform

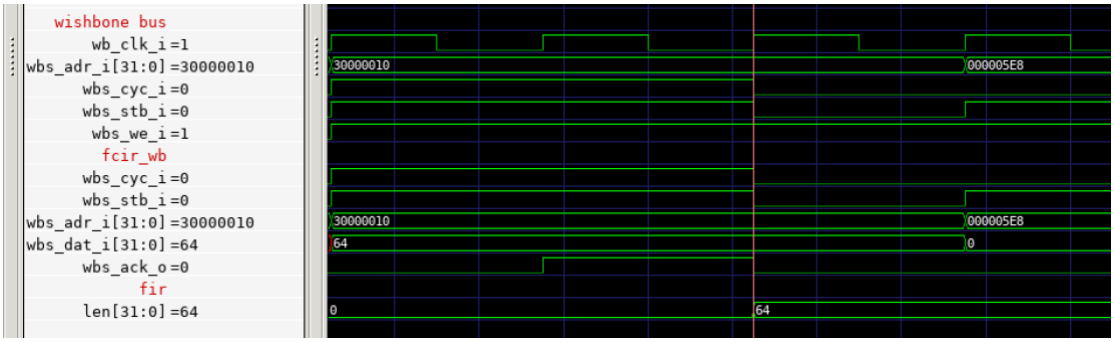
5.1 Step by firmware code

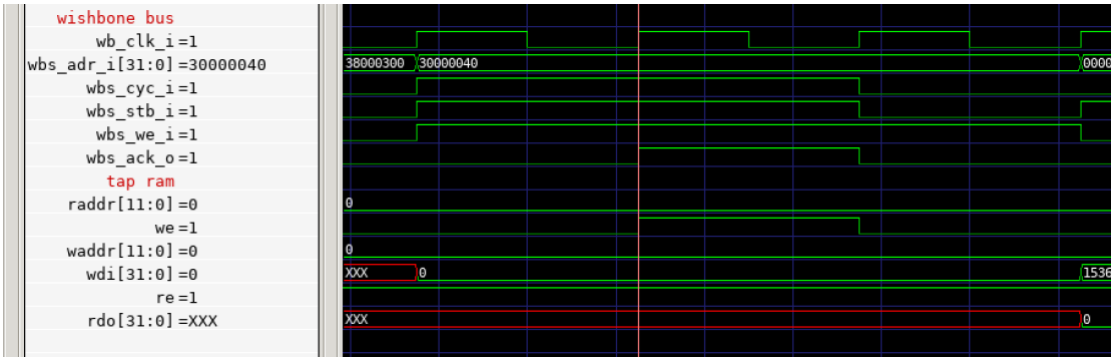
5.1.1 Allocate firmware code

The main function of the firmware code is written to BRAM at the beginning of the simulation.

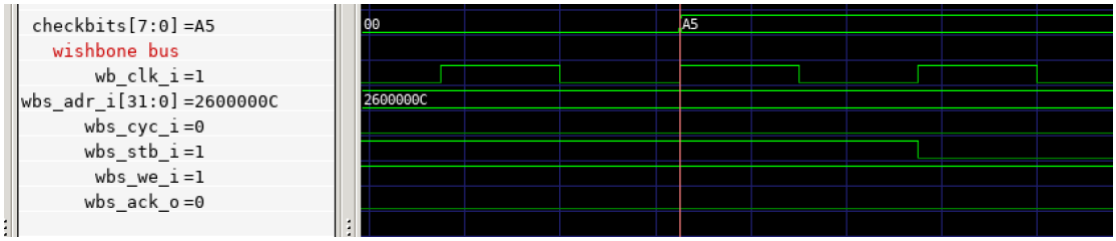


5.1.2 Program length and coefficient

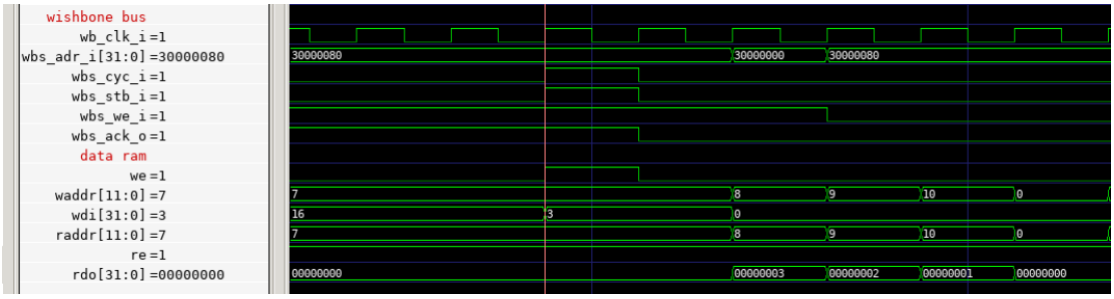




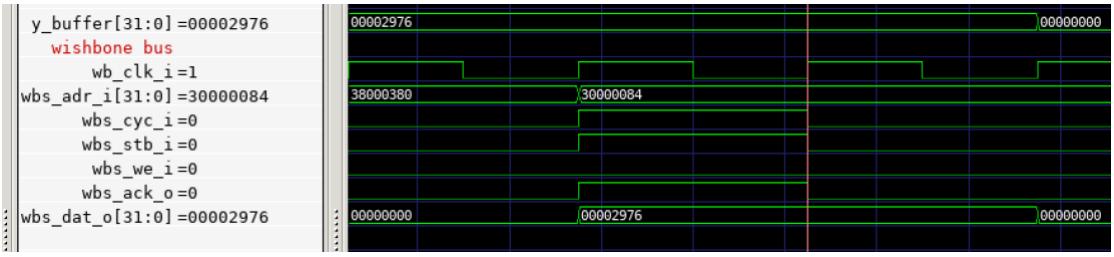
5.1.3 Set startMark



5.1.4 Send X



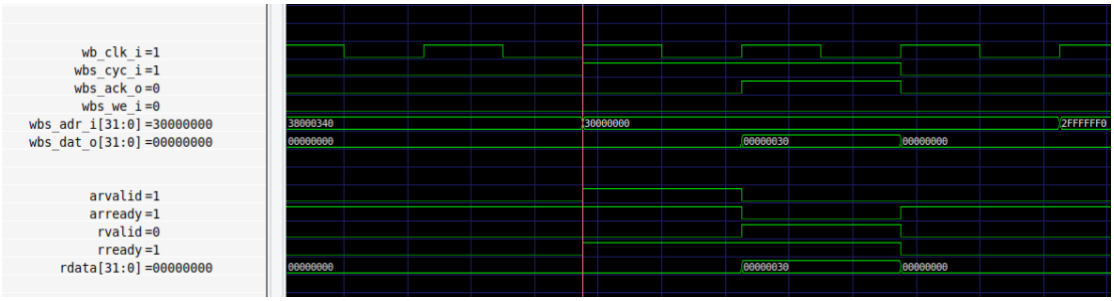
5.1.5 Receive Y



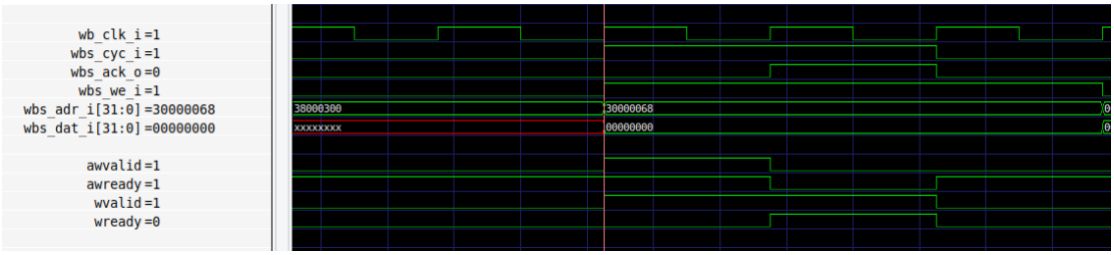
5.2 Handshake between different interface

5.2.1 Wishbone to AXI-Lite

The handshake of wishbone and axi-lite read operation

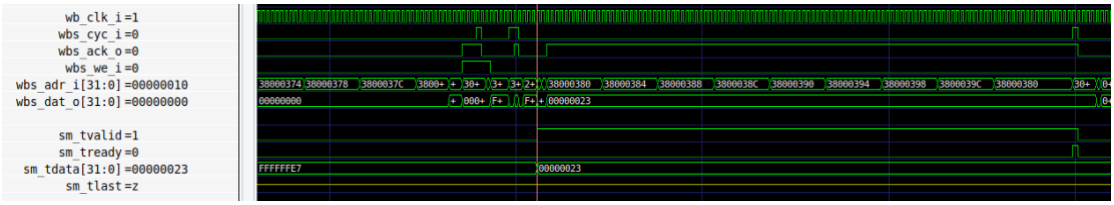


The handshake of wishbone and axi-lite write operation

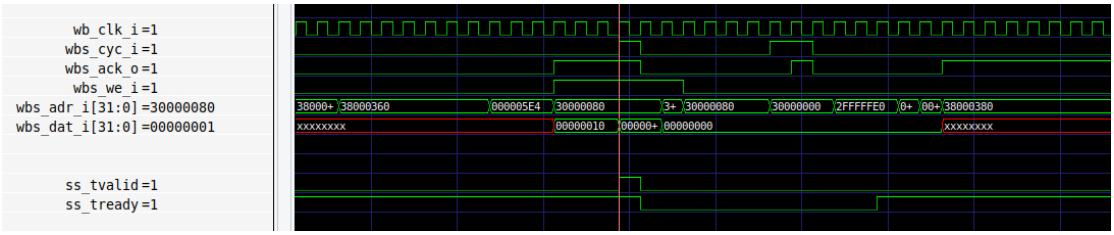


5.2.2 Wishbone to AXI-Stream

The handshake of wishbone and axi-stream read operation



The handshake of wishbone and axi-stream write operation



5.3 Analysis

5.3.1 Latency of firmware to feed data

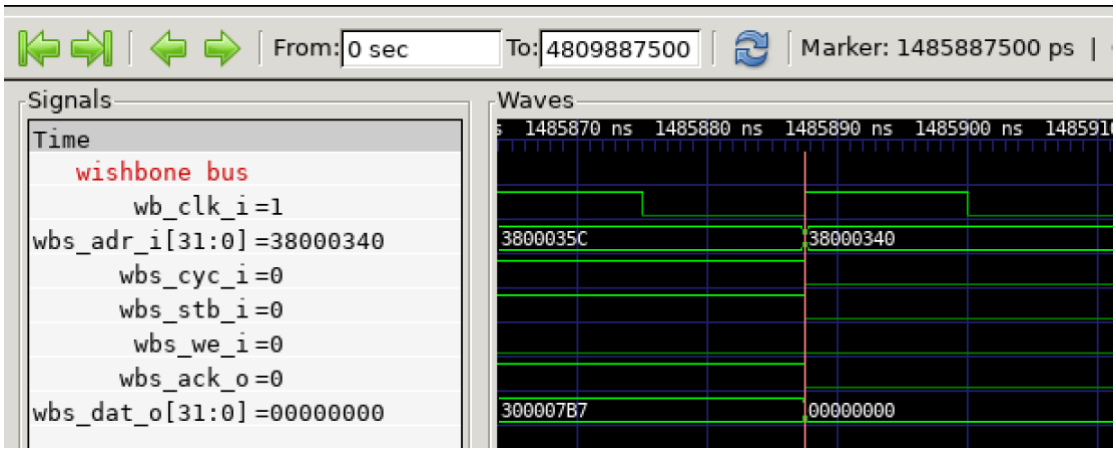
The instruction in firmware to feed data X to FIR engine.

```
    reg_fir_x = i;
```

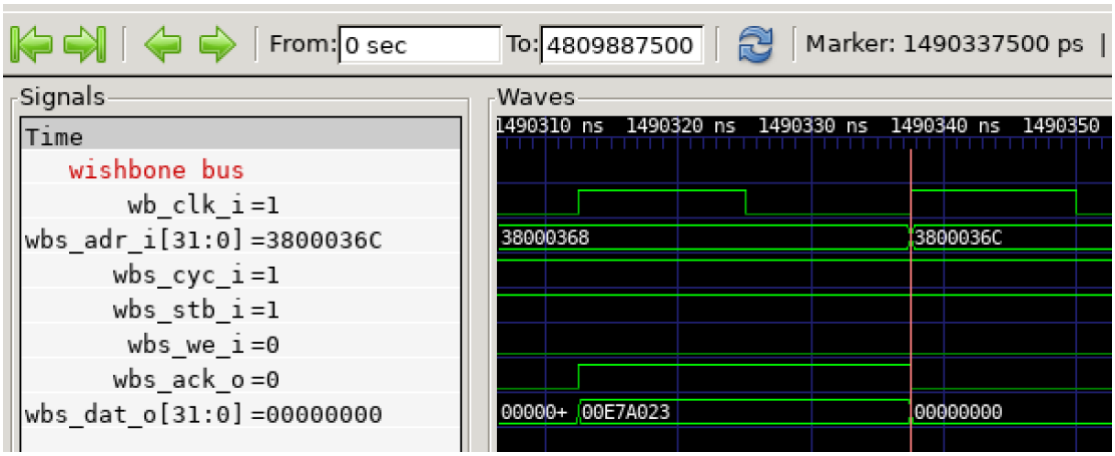
After compiling, we can the assembly code and the memory address.

```
    3800035c:    300007b7                lui     a5,0x30000
    38000360:    08078793                addi    a5,a5,128 # 30000080
<_esram_rom+0x1ffffd60>
    38000364:    fe442703                lw      a4,-28(s0)
    38000368:    00e7a023                sw      a4,0(a5)
```

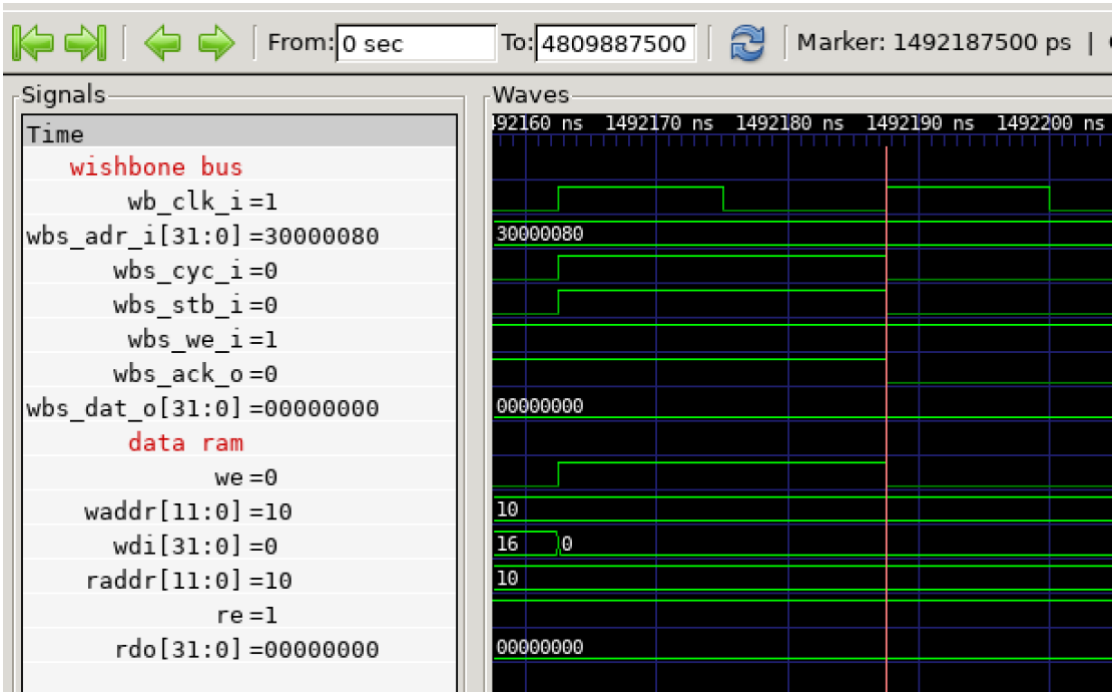
Find the waveform that RISC-V successfully read the instruction “0x300007b7” at address 0x3800035c. The simulation time is 1485887500 ps.



Find the waveform that RISC-V successfully read the store instruction “0x00e7a023” at address 0x38000368. The simulation time is 1490337500 ps.



Find the waveform that X data is successfully written to the data RAM in FIR engine. The simulation time is 1492187500 ps.



The latency of the firmware instruction:

1492187500 – 1485887500 = 6300000 ps = 6300 ns = 252 cycles

The latency of the store assembly instruction:

1492187500 – 1490337500 = 1850000 ps = 1850 ns = 74 cycles