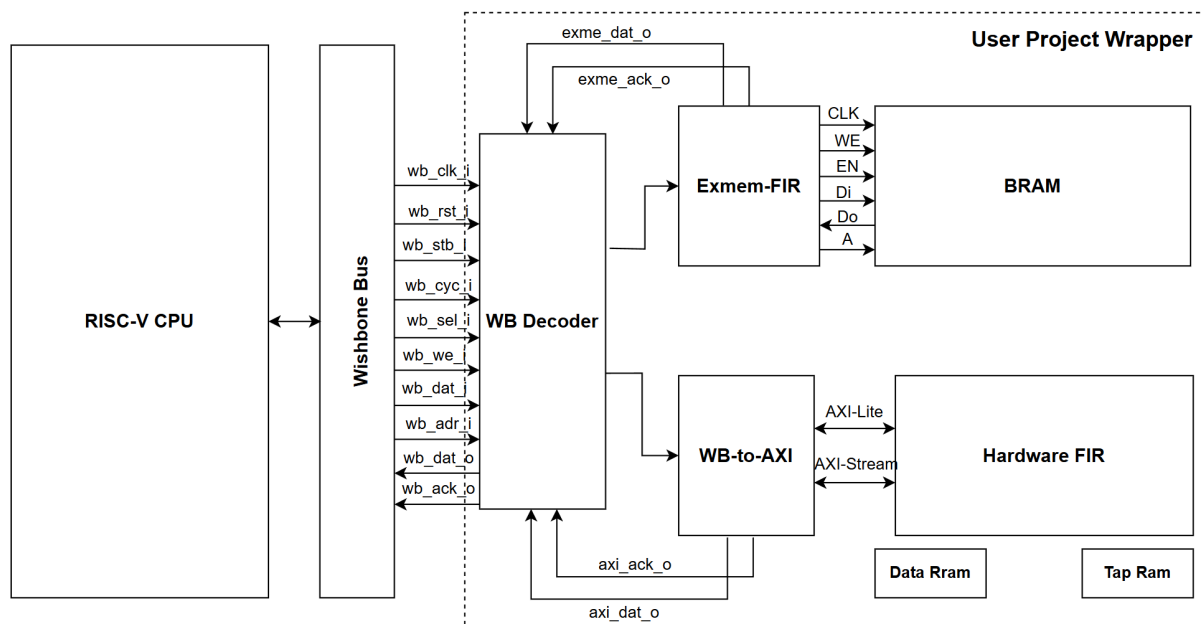


Name: Wu Chengkai
Student ID: 21144421

Laboratory 4-2 Caravel FIR

This experiment integrates Lab 3-FIR and Lab 4-1 exmem-FIR into the Caravel user project area. A Wishbone-to-AXI interface is designed to communicate with the FIR engine from Lab 3.

Design Block Diagram - Data Path and Control Path



Firmware, User Project, and Testbench Interface Protocols

Communication between firmware and the user project uses the Wishbone protocol. The RISC-V CPU executes firmware instructions to transmit data to the FIR engine and EXMEM via the Wishbone interface. The Wishbone decoder detects transactions based on addresses:

hardware FIR: Accessed via address `0x3000_XXXX`

exmem FIR: Accessed via address `0x3800_XXXX`

The firmware accesses the FIR engine in the user project through a Wishbone-to-AXI conversion layer, which translates Wishbone requests into AXI-Lite or AXI-Stream formats.

FIR Engine Register Address Mapping:

```
0x00:    AP Configuration Register (AXI-Lite)
0x10-13: Data Length Register (AXI-Lite)
0x40-7F: Tap Parameter Storage Area
0x80-43: X[n] Input  (AXI-Stream)
0x84-87: Y[n] Output (AXI-Stream)
```

Firmware and Testbench

Communication via mprj_io signals:

The testbench sets mprj_io[31:16] as a checksum to monitor project status.

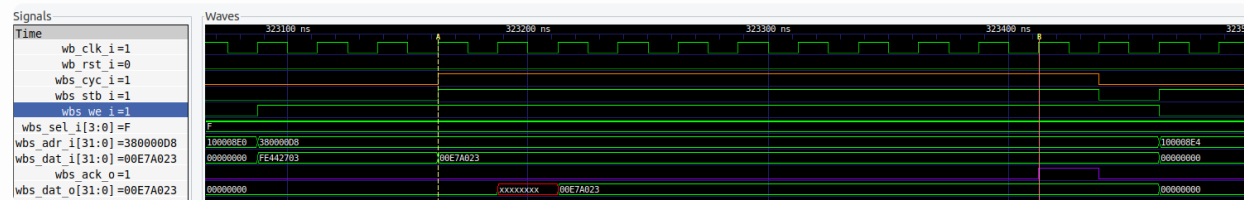
When the firmware loads the FIR engine, the checksum is set to 16'h00A5, signaling the testbench to begin checking FIR outputs and calculating latency.

After FIR processing completes, the firmware sets the checksum to 8'h5A, prompting the testbench to validate final Y[n] outputs against reference values and record total latency. This process repeats 3 times for statistical analysis.

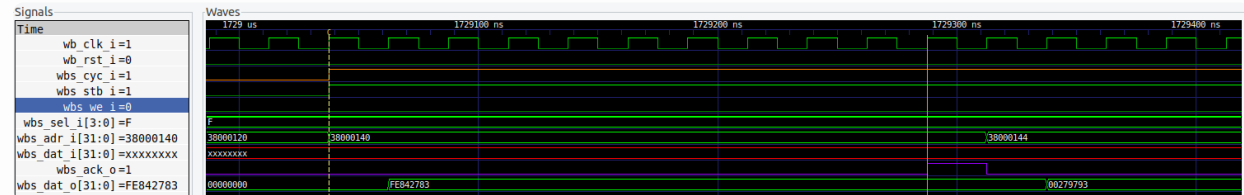
Simulation Results

Wb to Exmem (Lab4-1)

Write 10 cycles delay



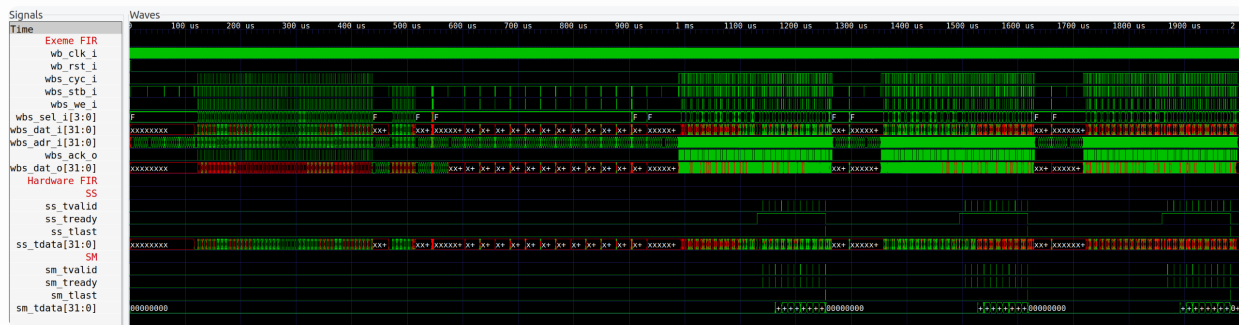
Read 10 cycles delay



Testbench results

```
ubuntu@ubuntu2004: ~/Documents/Lab4/Lab4-1/lab-exmem_fir/testbench/counter_la_fir
ubuntu@ubuntu2004:~/Documents/Lab4/Lab4-1/lab-exmem_fir/testbench/counter_la_fir$ source run_clean
ubuntu@ubuntu2004:~/Documents/Lab4/Lab4-1/lab-exmem_fir/testbench/counter_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:~/Documents/Lab4/Lab4-1/lab-exmem_fir/testbench/counter_la_fir$
```

Wb to FIR (Lab4-2)



Theoretical Throughput

For FIR engine, ideally needs 12 cycles to output Y (32bit)

$$\text{theoretical throughput} = 32 \text{ bits} / 12 \text{ cycle} = 0.1067 \text{ bits/ns}$$

Actually measured throughput

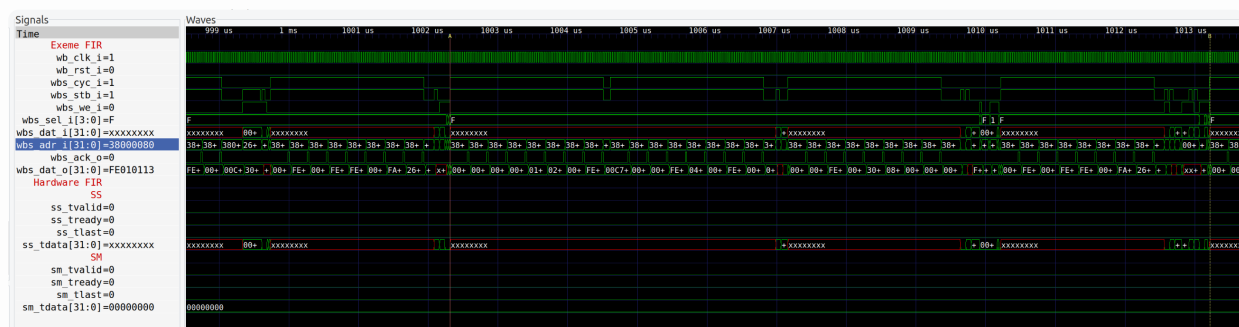
```
ubuntu@ubuntu2004: ~/Documents/Lab4/Lab4-2/lab-caravel_fir/testbench/counter_la_fir
ubuntu@ubuntu2004:~/Documents/Lab4/Lab4-2/lab-caravel_fir/testbench/counter_la_fir$ source run_cleuu
buubuntubuubuntu@ubuntu2004:~/Documents/Lab4/Lab4-2/lab-caravel_fir/testbench/counter_la_fir$ source
ubuntu@ubuntu2004:~/Documents/Lab4/Lab4-2/lab-caravel_fir/testbench/counter_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.
data length finished...
coefficient finished...
===== Start FIR Test 1 =====
Final Y value correct - golden: 0x93, final Y: 0x93
Executes in 5327 cycles
===== END FIR Test 1 =====
===== Start FIR Test 2 =====
Final Y value correct - golden: 0x93, final Y: 0x93
Executes in 5327 cycles
===== END FIR Test 2 =====
===== Start FIR Test 3 =====
Final Y value correct - golden: 0x93, final Y: 0x93
Executes in 5327 cycles
===== END FIR Test 3 =====
Total cycles: 15981
ubuntu@ubuntu2004:~/Documents/Lab4/Lab4-2/lab-caravel_fir/testbench/counter_la_fir$
```

$$5327 / 11 = 484 \text{ cycles}$$

$$32 \text{ bits} / 484 = 0.00264 \text{ bits/ns}$$

Latency Calculation

cycles between each $x[N]$ input at addr = 0x3000080



1013215ns - 1002312ns / 25 ns = 436 cycles

Throughput Optimization Techniques

Parallelize multiply-accumulate operations, adding additional Adder and multiplier.
Decouple input/output to preload $X[n]$ and reduce wait time for $Y[n]$.

GitHub Repository

<https://github.com/edwu186/System-on-Chip-SoC-Laboratory/tree/main/Lab4-2>