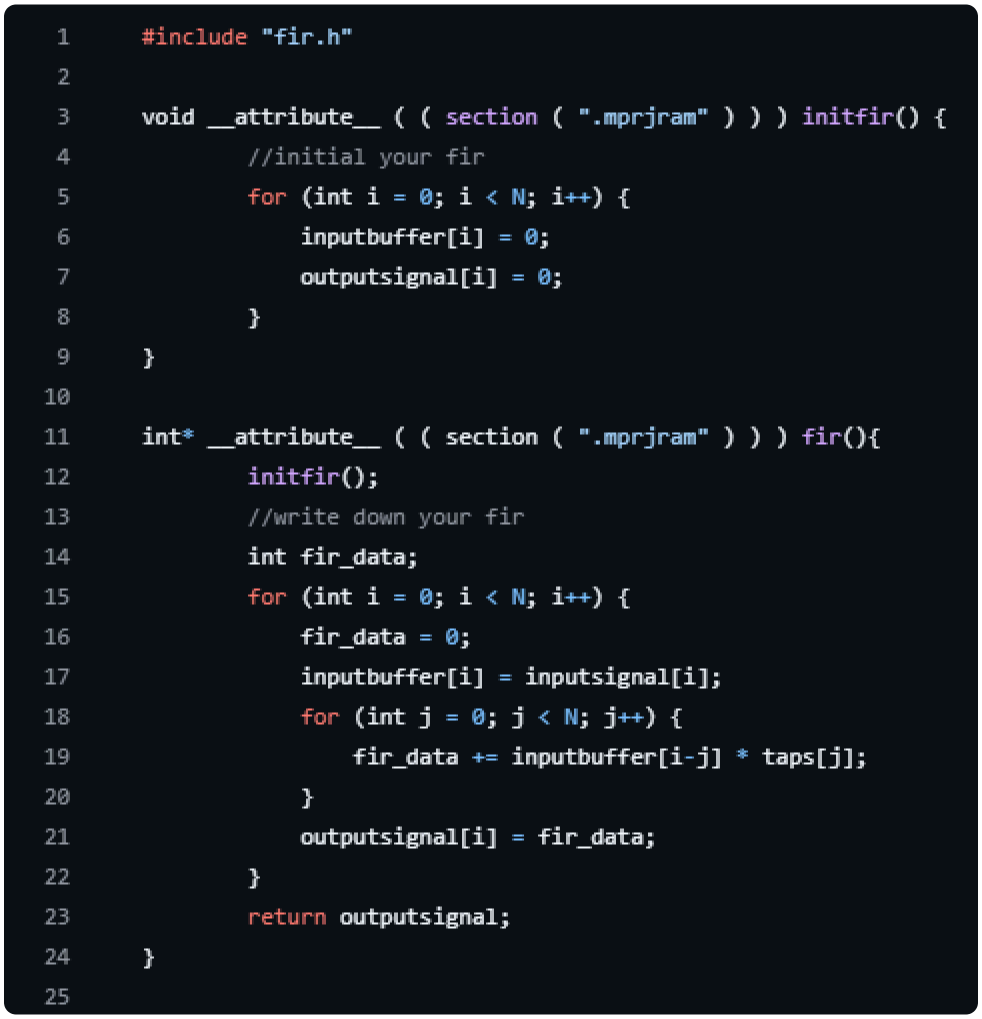
Execute Code in User Memory (lab_4-1)

Student ID: 20726557

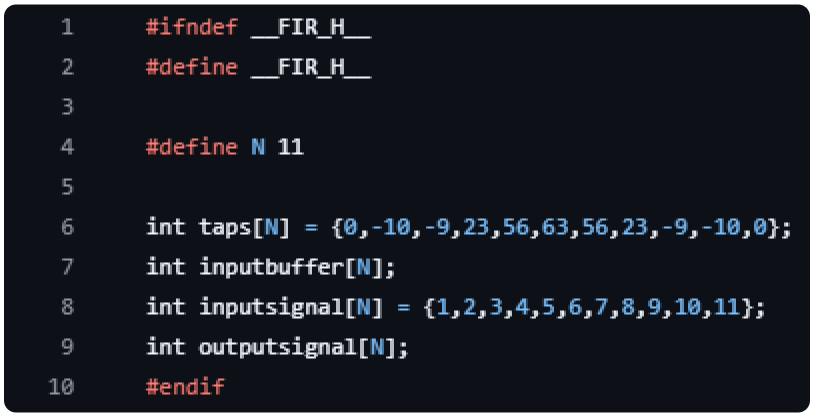
List relevant code changes from rtl/firmware

fir.c



```c
#include "fir.h"

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
	int fir_data;
	for (int i = 0; i < N; i++) {
	    fir_data = 0;
	    inputbuffer[i] = inputsignal[i];
	    for (int j = 0; j < N; j++) {
	        fir_data += inputbuffer[i-j] * taps[j];
	    }
	    outputsignal[i] = fir_data;
	}
	return outputsignal;
}

```

fir.h



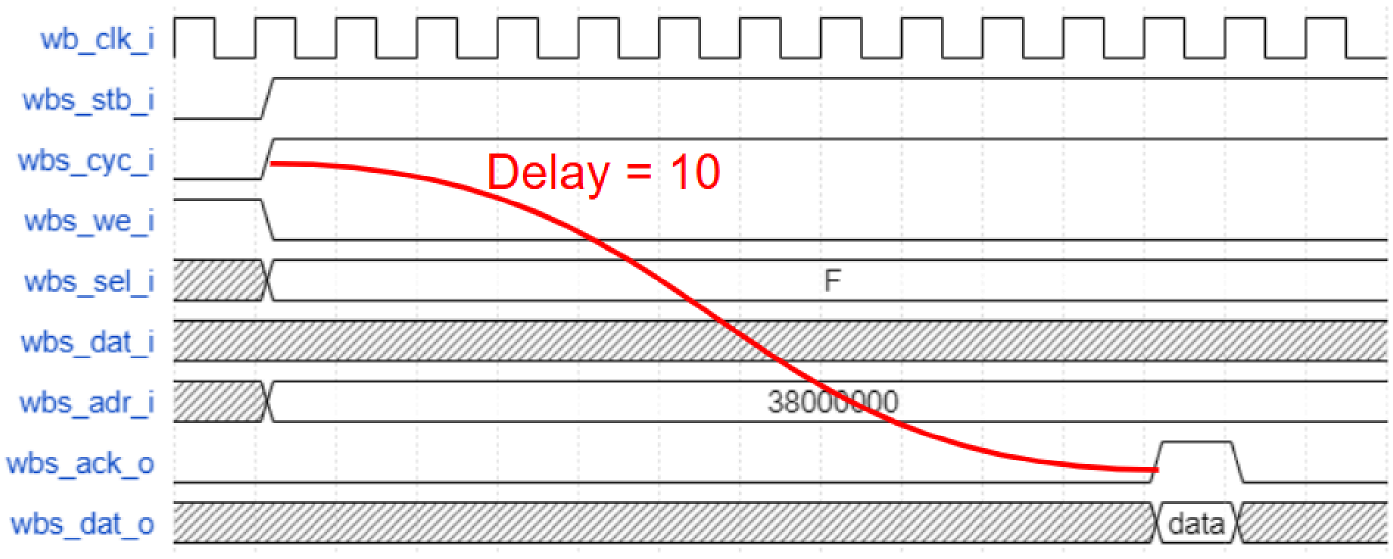
```c
#ifndef __FIR_H__
#define __FIR_H__

#define N 11

int taps[N] = {0,-10,-9,23,56,63,56,23,-9,-10,0};
int inputbuffer[N];
int inputsignal[N] = {1,2,3,4,5,6,7,8,9,10,11};
int outputsignal[N];
#endif
```

initfir() initializes the inputbuffer and outputsignal arrays to 0. When fir() is called, the initfir() will be run and the FIR computation will store the data into the outputsignal.

user_project_example.counter.v

```verilog
    reg  [31:0] delays_count;

    wire [3:0]  bram_we;
    wire        bram_en;

    assign clk = wb_clk_i;
    assign rst = wb_rst_i;

    always @(posedge clk) begin
        if (rst) begin
            delays_count[31:0] <= 32'b0;
        end else begin
            if (bram_en) begin
                if (delays_count[31:0] != DELAYS) begin
                    delays_count[31:0] <= delays_count[31:0] + 32'b1;
                end else begin
                    delays_count[31:0] <= 32'b0;
                end
            end
        end
    end

    assign wbs_ack_o = delays_count[31:0] == DELAYS;

    assign bram_we[3:0] = {4{wbs_we_i}} & wbs_sel_i[3:0]                   ;
    assign bram_en      = wbs_stb_i & wbs_cyc_i & wbs_adr_i[31:24] == 8'h38;

    bram user_bram (
        .CLK(clk            ),
        .WE0(bram_we[3:0]   ),
        .EN0(bram_en        ),
        .Di0(wbs_dat_i[31:0]),
        .Do0(wbs_dat_o[31:0]),
        .A0 (wbs_adr_i[31:0])
    );

endmodule
```

The wishbone bus controller is designed with a response after the delay. The delay is configured by the DELAYS parameter. This is implemented by using a counter. When the counter reaches the value of DELAYS, wbs_acks_o will be asserted.



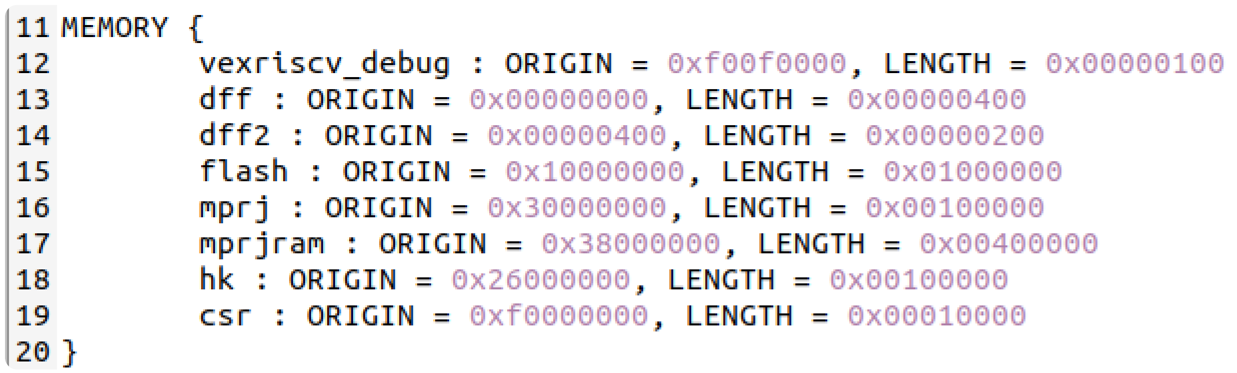
bram.v

```verilog

        // 16 kB
        parameter N = 11;
        (* ram_style = "block" *) reg [31:0] RAM[0:2**N-1];


        always @(posedge CLK)
            if(EN0) begin
                Do0 <= RAM[A0[N-1:0]];
                if(WE0[0]) RAM[A0[N-1:0]][7:0] <= Di0[7:0];
                if(WE0[1]) RAM[A0[N-1:0]][15:8] <= Di0[15:8];
                if(WE0[2]) RAM[A0[N-1:0]][23:16] <= Di0[23:16];
                if(WE0[3]) RAM[A0[N-1:0]][31:24] <= Di0[31:24];
            end
            else
                Do0 <= 32'b0;
    endmodule
```

To configure the suitable size of the BRAM, the BRAM will use the parameter N, which is the number of taps, to determine the size.

Memory map & linker (lds)

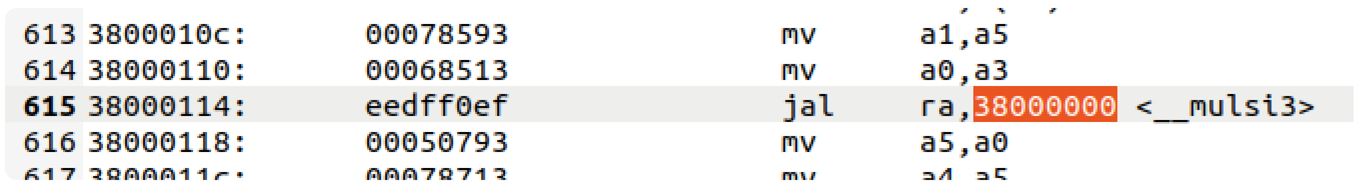
The following is the memory map of the project.



```
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
}
```

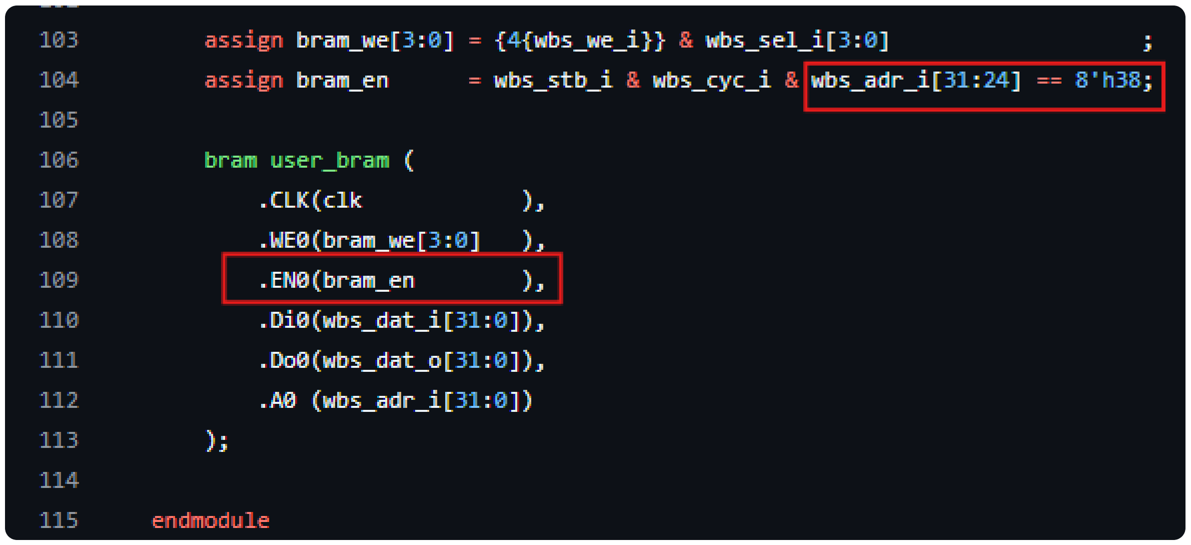
In this lab, the BRAM is used for FIR computation. The BRAM address is mapped to "mprjram" at 0x38000000.

To excute the FIR computation and store in BRAM, the address at 0x38000000 is accessed.



```
3800010c:       00078593                mv      a1,a5
38000110:       00068513                mv      a0,a3
38000114:       eedff0ef                jal     ra,38000000 <__mulsi3>
38000118:       00050793                mv      a5,a0
3800011c:       00078713                mv      a4,a5
```

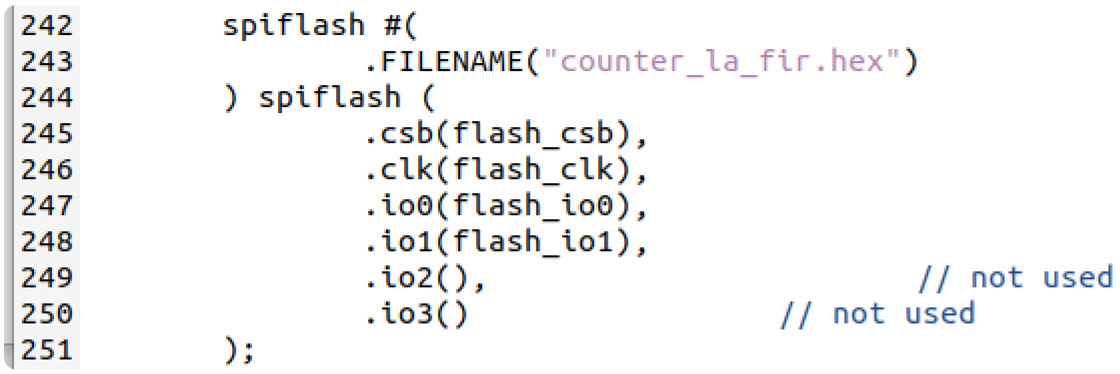
In the wishbone controller, the BRAM enable is qualified by the wishbone address of 0x38000000.



```
        assign bram_we[3:0] = {4{wbs_we_i}} & wbs_sel_i[3:0]                ;
        assign bram_en      = wbs_stb_i & wbs_cyc_i & wbs_adr_i[31:24] == 8'h38;

        bram user_bram (
            .CLK(clk              ),
            .WE0(bram_we[3:0]     ),
            .EN0(bram_en          ),
            .Di0(wbs_dat_i[31:0]),
            .Do0(wbs_dat_o[31:0]),
            .A0 (wbs_adr_i[31:0])
        );

    endmodule
```

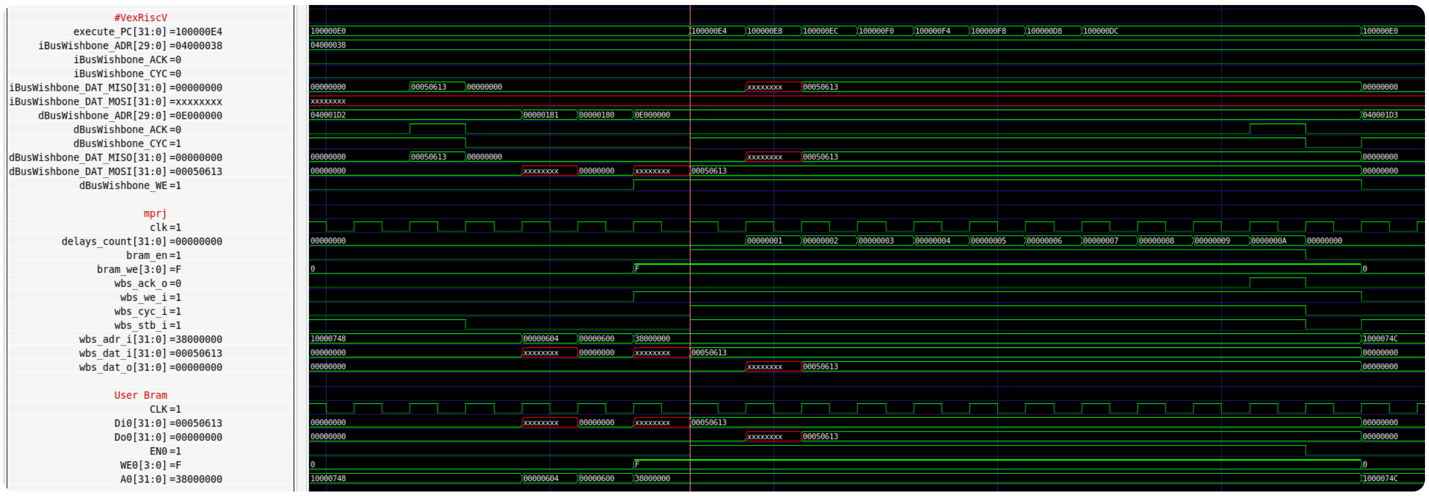
How to move code from spiflash to user project area memory

After compiling the code, it is stored in "counter_la_fir.hex". In the testbench, the .hex file is directly written as the spiflash.



```
        spiflash #(
                .FILENAME("counter_la_fir.hex")
        ) spiflash (
                .csb(flash_csb),
                .clk(flash_clk),
                .io0(flash_io0),
                .io1(flash_io1),
                .io2(),                         // not used
                .io3()                  // not used
        );
```

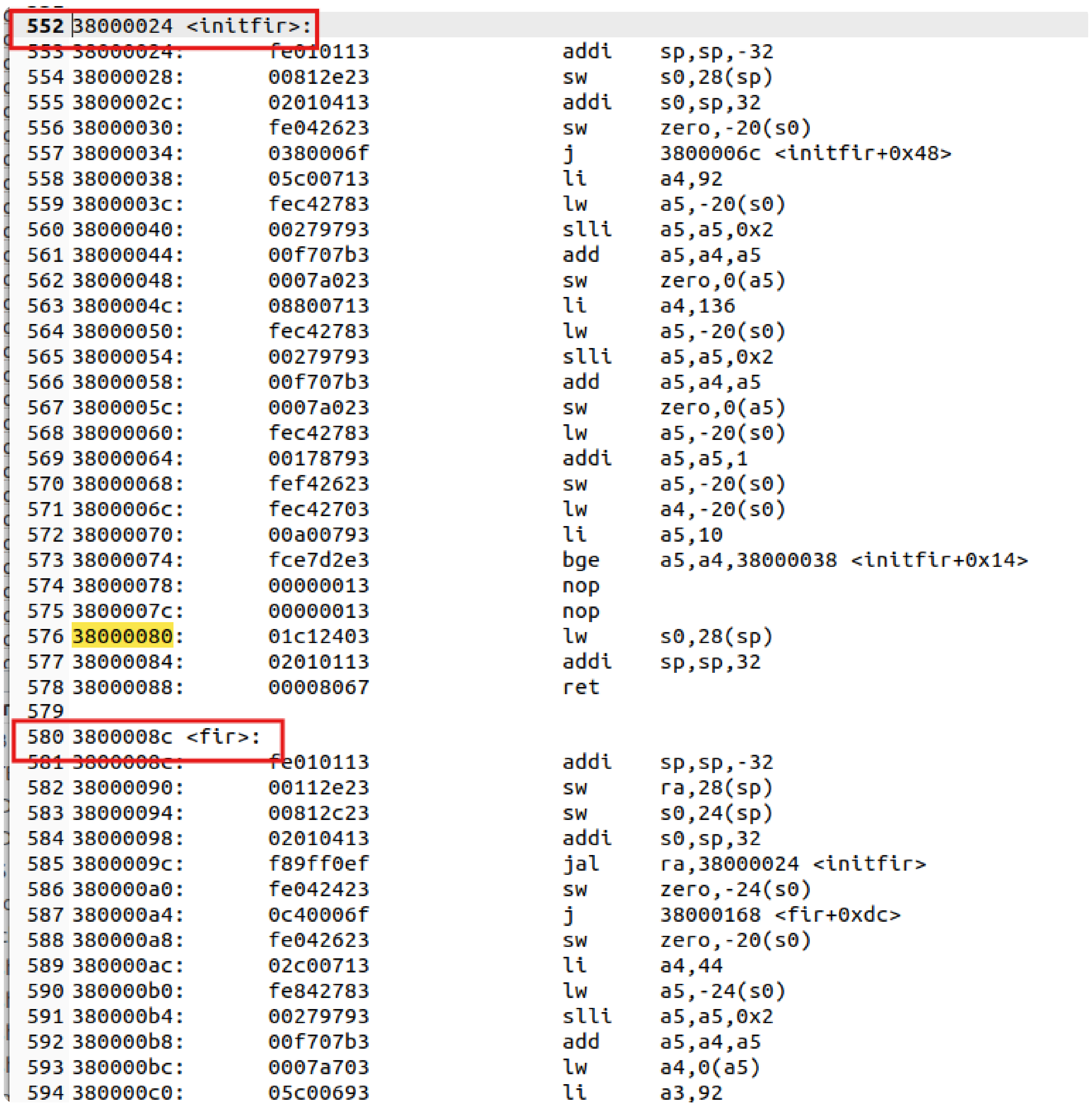
The code from the SPIFlash is written to the user project area memory through the Wishbone interface.



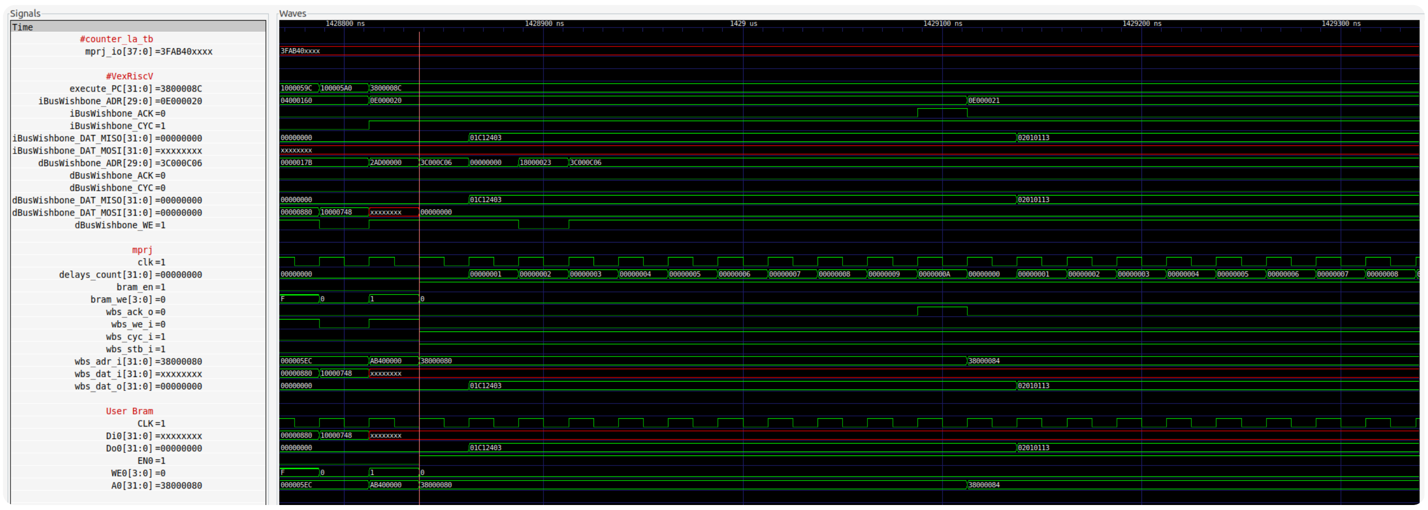
```
#VexRiscV
execute_PC[31:0] =100000E4
iBusWishbone_ADR[29:0] =04000038
iBusWishbone_ACK =0
iBusWishbone_CYC =0
iBusWishbone_DAT_MISO[31:0] =00000000
iBusWishbone_DAT_MOSI[31:0] =xxxxxxxx
dBusWishbone_ADR[29:0] =0E000000
dBusWishbone_ACK =0
dBusWishbone_CYC =1
dBusWishbone_DAT_MISO[31:0] =00000000
dBusWishbone_DAT_MOSI[31:0] =00050613
dBusWishbone_WE =1

mprj
clk =1
delays_count[31:0] =00000000
bram_en =1
bram_we[3:0] =F
wbs_ack_o =0
wbs_we_i =1
wbs_cyc_i =1
wbs_stb_i =1
wbs_adr_i[31:0] =38000000
wbs_dat_i[31:0] =00050613
wbs_dat_o[31:0] =00000000

User Bram
CLK =1
Di0[31:0] =00050613
Do0[31:0] =00000000
EN0 =1
WE0[3:0] =F
A0[31:0] =38000000
```

How to execute code from user project memory

In this lab, the target function is the computer FIR. The code was first pre-loaded into the User Project Memory. The code is executed from the user project memory when the target function is called.



```
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

3800008c <fir>:
3800008c:       fe010113                addi    sp,sp,-32
38000090:       00112e23                sw      ra,28(sp)
38000094:       00812c23                sw      s0,24(sp)
38000098:       02010413                addi    s0,sp,32
3800009c:       f89ff0ef                jal     ra,38000024 <initfir>
380000a0:       fe042423                sw      zero,-24(s0)
380000a4:       0c40006f                j       38000168 <fir+0xdc>
380000a8:       fe042623                sw      zero,-20(s0)
380000ac:       02c00713                li      a4,44
380000b0:       fe842783                lw      a5,-24(s0)
380000b4:       00279793                slli    a5,a5,0x2
380000b8:       00f707b3                add     a5,a4,a5
380000bc:       0007a703                lw      a4,0(a5)
380000c0:       05c00693                li      a3,92
```

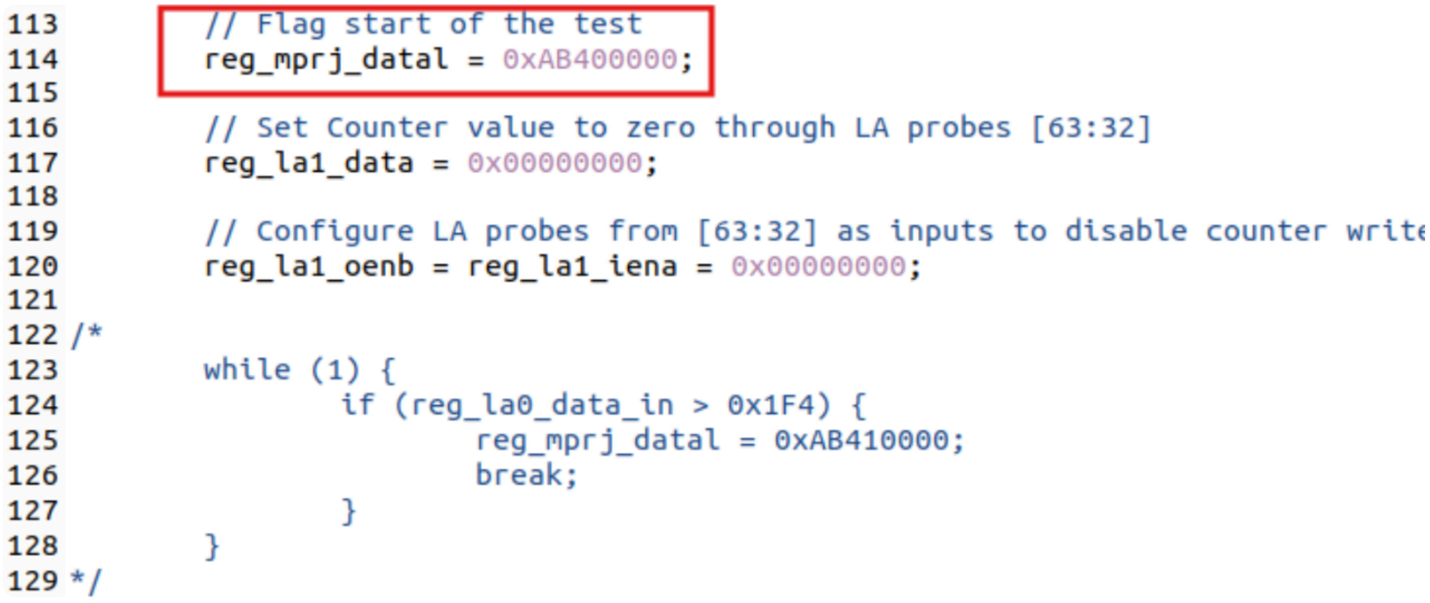


```
Signals
Time
              #counter_la_tb
              mprj_io[37:0] =3FAB40xxxx

                   #VexRiscV
          execute_PC[31:0] =3800008C
     iBusWishbone_ADR[29:0] =0E000020
          iBusWishbone_ACK =0
          iBusWishbone_CYC =1
iBusWishbone_DAT_MISO[31:0] =00000000
iBusWishbone_DAT_MOSI[31:0] =xxxxxxxx
     dBusWishbone_ADR[29:0] =3C000C06
          dBusWishbone_ACK =0
          dBusWishbone_CYC =0
dBusWishbone_DAT_MISO[31:0] =00000000
dBusWishbone_DAT_MOSI[31:0] =00000000
           dBusWishbone_WE =1

                      mprj
                       clk =1
        delays_count[31:0] =00000000
                   bram_en =1
             bram_we[3:0] =0
                 wbs_ack_o =0
                  wbs_we_i =0
                 wbs_cyc_i =1
                 wbs_stb_i =1
          wbs_adr_i[31:0] =38000080
          wbs_dat_i[31:0] =xxxxxxxx
          wbs_dat_o[31:0] =00000000

                 User Bram
                       CLK =1
                Di0[31:0] =xxxxxxxx
                Do0[31:0] =00000000
                       EN0 =1
                 WE0[3:0] =0
                 A0[31:0] =38000080
```

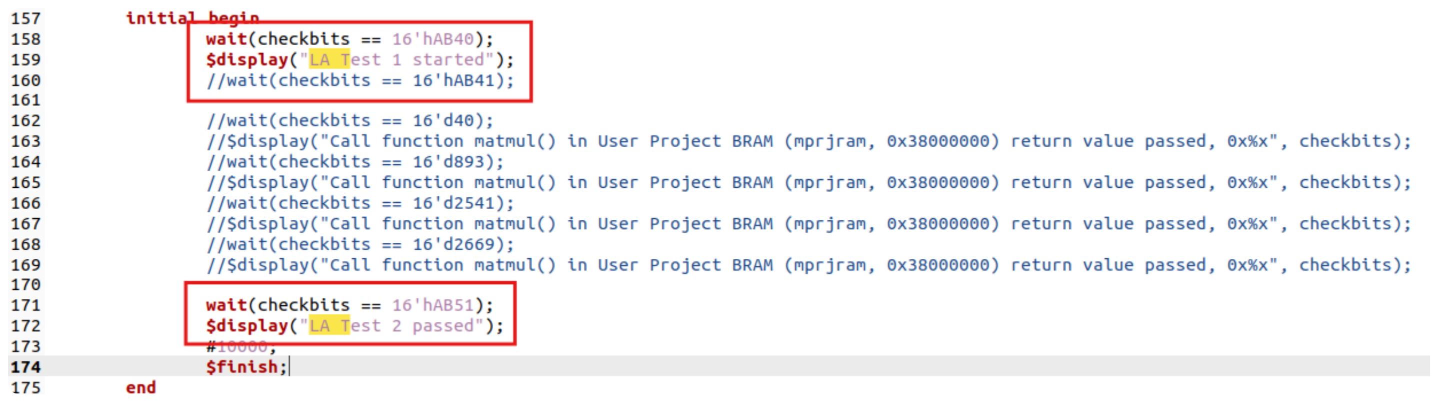
The code is read from the user project memory to fetch the corresponding instruction to execute.

Show the Operation sequence and its waveform

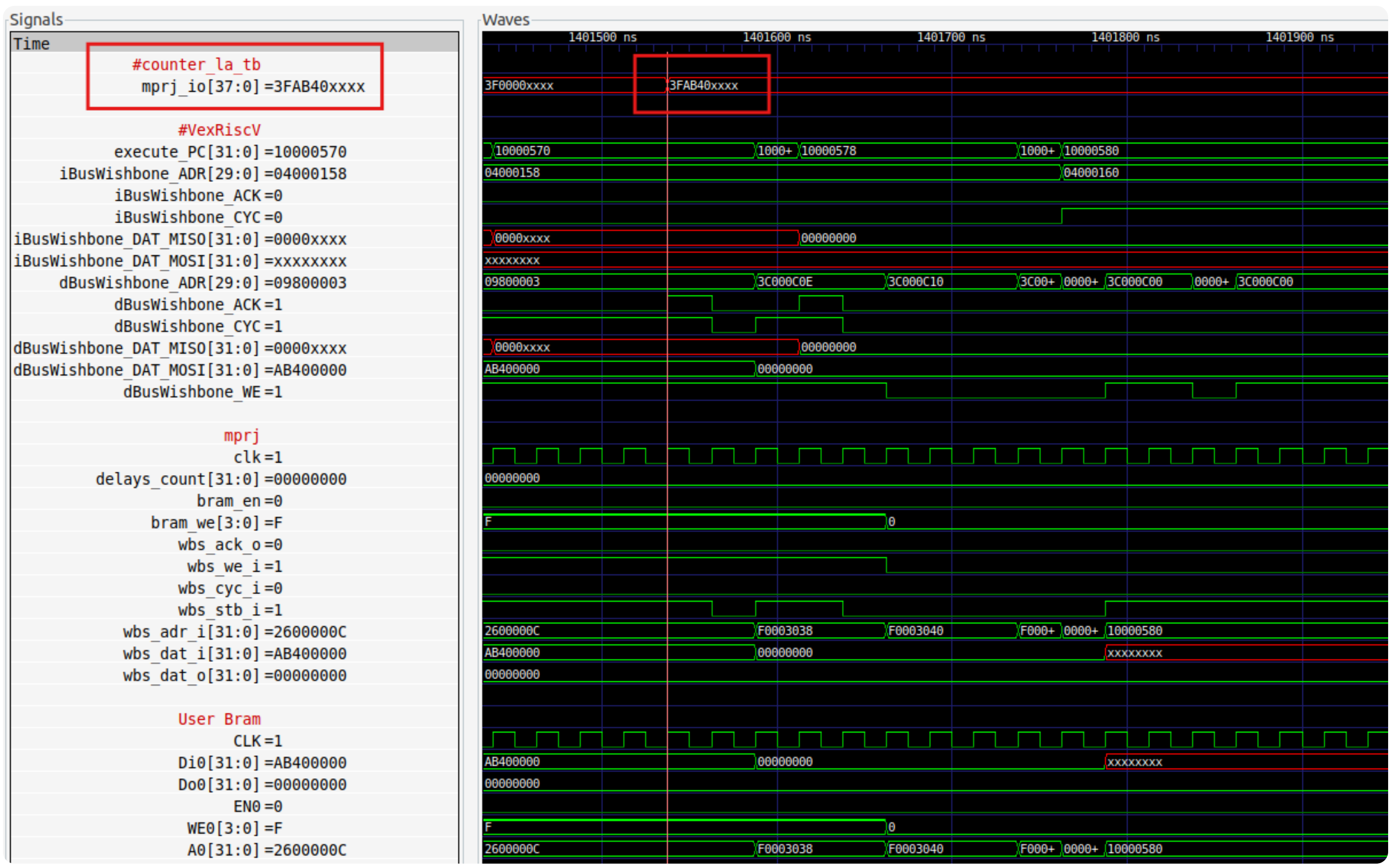
The FIR computation is started when mprj_io is set to 0xAB40.



```
        // Flag start of the test
        reg_mprj_datal = 0xAB400000;

        // Set Counter value to zero through LA probes [63:32]
        reg_la1_data = 0x00000000;

        // Configure LA probes from [63:32] as inputs to disable counter write
        reg_la1_oenb = reg_la1_iena = 0x00000000;

/*
        while (1) {
                if (reg_la0_data_in > 0x1F4) {
                        reg_mprj_datal = 0xAB410000;
                        break;
                }
        }
*/
```

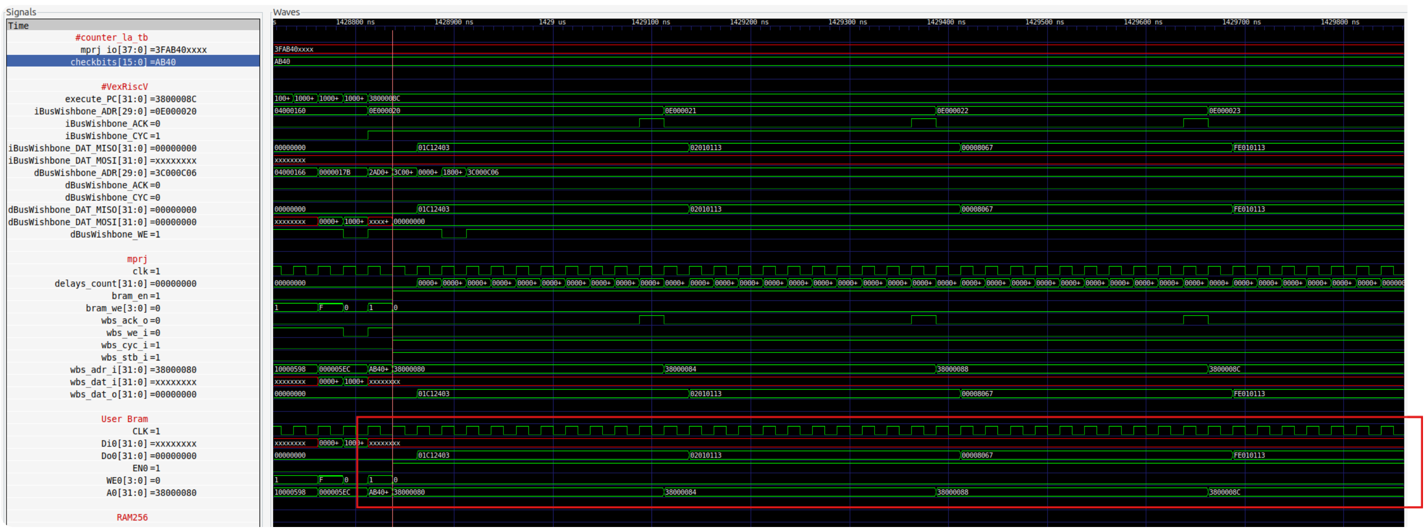


```
        initial begin
            wait(checkbits == 16'hAB40);
            $display("LA Test 1 started");
            //wait(checkbits == 16'hAB41);

            //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;
            $finish;
        end
```

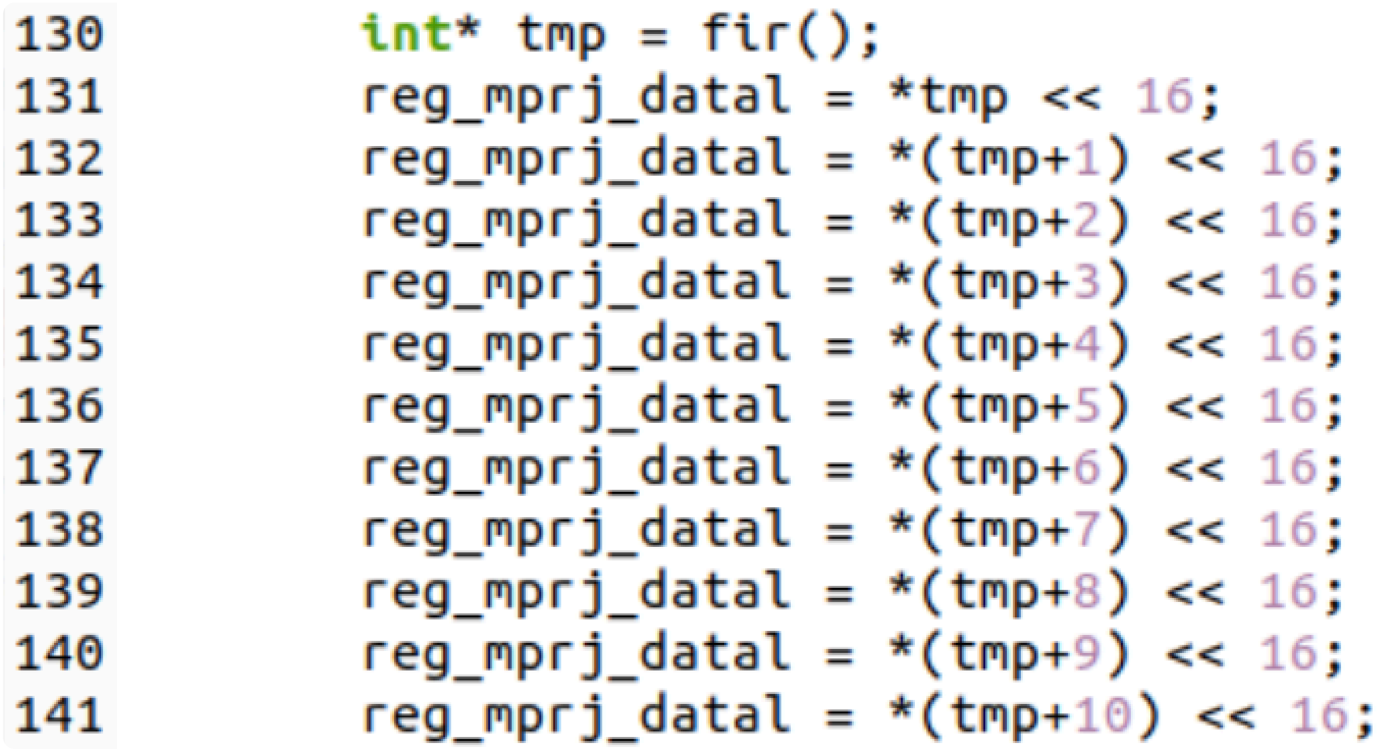


```
Signals
Time
#counter_la_tb
  mprj_io[37:0] =3FAB40xxxx
#VexRiscV
  execute_PC[31:0] =10000570
  iBusWishbone_ADR[29:0] =04000158
  iBusWishbone_ACK =0
  iBusWishbone_CYC =0
  iBusWishbone_DAT_MISO[31:0] =0000xxxx
  iBusWishbone_DAT_MOSI[31:0] =xxxxxxxx
  dBusWishbone_ADR[29:0] =09800003
  dBusWishbone_ACK =1
  dBusWishbone_CYC =1
  dBusWishbone_DAT_MISO[31:0] =0000xxxx
  dBusWishbone_DAT_MOSI[31:0] =AB400000
  dBusWishbone_WE =1
mprj
  clk =1
  delays_count[31:0] =00000000
  bram_en =0
  bram_we[3:0] =F
  wbs_ack_o =0
  wbs_we_i =1
  wbs_cyc_i =0
  wbs_stb_i =1
  wbs_adr_i[31:0] =2600000C
  wbs_dat_i[31:0] =AB400000
  wbs_dat_o[31:0] =00000000
User Bram
  CLK =1
  Di0[31:0] =AB400000
  Do0[31:0] =00000000
  EN0 =0
  WE0[3:0] =F
  A0[31:0] =2600000C
```

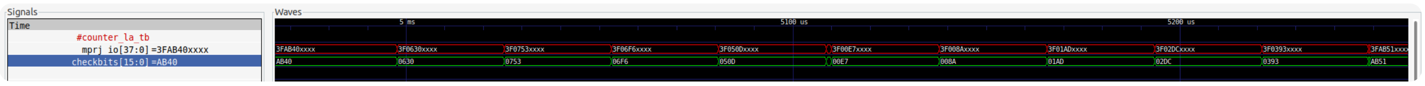
The FIR code is fetched from the user project memory to execute the "initfir" and "fir" function.


Signals
Time
#counter_la_tb
mprj_io[37:0] =3FAB40xxxx
checkbits[15:0] =AB40
#VexRiscV
execute_PC[31:0] =3800008C
iBusWishbone_ADR[29:0] =0E000020
iBusWishbone_ACK =0
iBusWishbone_CYC =1
iBusWishbone_DAT_MISO[31:0] =00000000
iBusWishbone_DAT_MOSI[31:0] =xxxxxxxx
dBusWishbone_ADR[29:0] =3C000C06
dBusWishbone_ACK =0
dBusWishbone_CYC =0
dBusWishbone_DAT_MISO[31:0] =00000000
dBusWishbone_DAT_MOSI[31:0] =00000000
dBusWishbone_WE =1
mprj
clk =1
delays_count[31:0] =00000000
bram_en =1
bram_we[3:0] =0
wbs_ack_o =0
wbs_we_i =0
wbs_cyc_i =1
wbs_stb_i =1
wbs_adr_i[31:0] =38000080
wbs_dat_i[31:0] =xxxxxxxx
wbs_dat_o[31:0] =00000000
User Bram
CLK =1
Di0[31:0] =xxxxxxxx
Do0[31:0] =00000000
EN0 =1
WE0[3:0] =0
A0[31:0] =38000080
RAM256
Waves

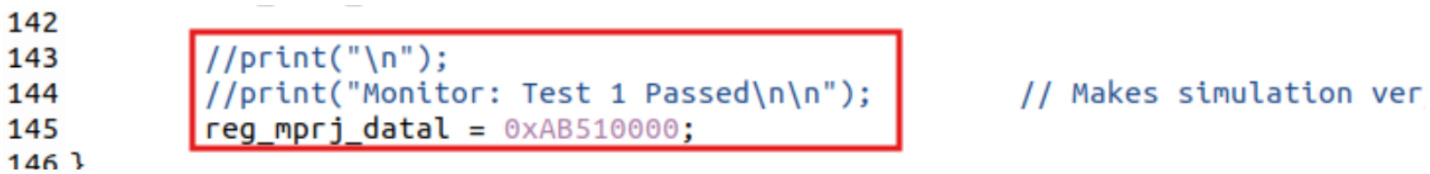
In the code, the "fir" function is computed 11 times and sets check bits 11 times.



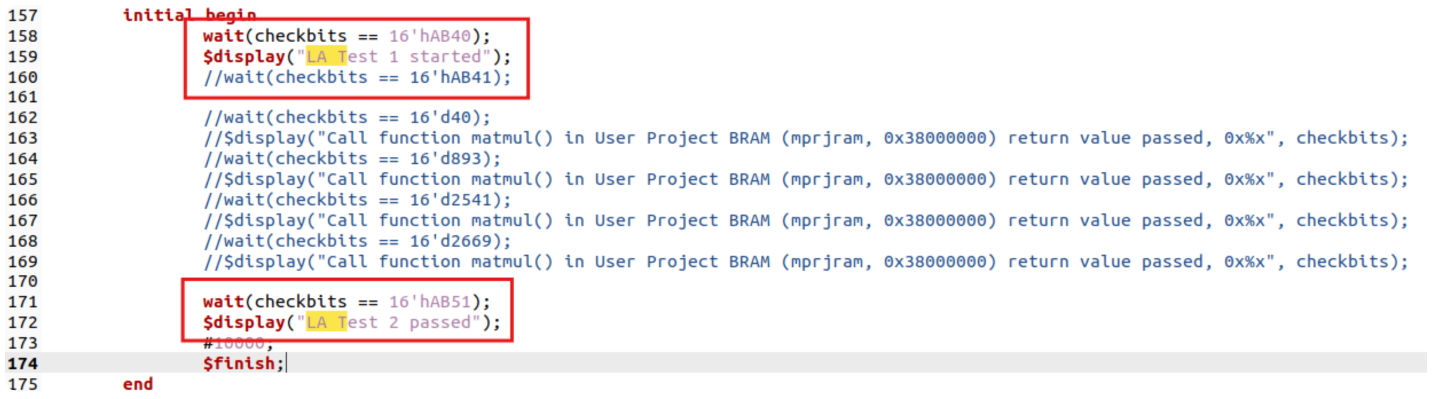
```c
int* tmp = fir();
reg_mprj_datal = *tmp << 16;
reg_mprj_datal = *(tmp+1) << 16;
reg_mprj_datal = *(tmp+2) << 16;
reg_mprj_datal = *(tmp+3) << 16;
reg_mprj_datal = *(tmp+4) << 16;
reg_mprj_datal = *(tmp+5) << 16;
reg_mprj_datal = *(tmp+6) << 16;
reg_mprj_datal = *(tmp+7) << 16;
reg_mprj_datal = *(tmp+8) << 16;
reg_mprj_datal = *(tmp+9) << 16;
reg_mprj_datal = *(tmp+10) << 16;
```



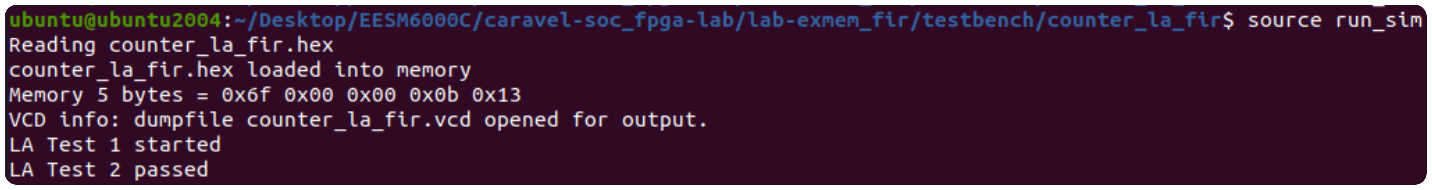
When check_bits is set to 0xAB51, the FIR computation is finished.



```c
        //print("\n");
        //print("Monitor: Test 1 Passed\n\n");        // Makes simulation ver
        reg_mprj_datal = 0xAB510000;
}
```



```
	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;
		$finish;
	end
```



```
ubuntu@ubuntu2004:~/Desktop/EESM6000C/caravel-soc_fpga-lab/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
```

GitHub

https://github.com/AnthonyGithub/EESM6000C-Lab-4/tree/main/Lab 4-1

(https://github.com/AnthonyGithub/EESM6000C-Lab-4/tree/main/Lab%204-1)