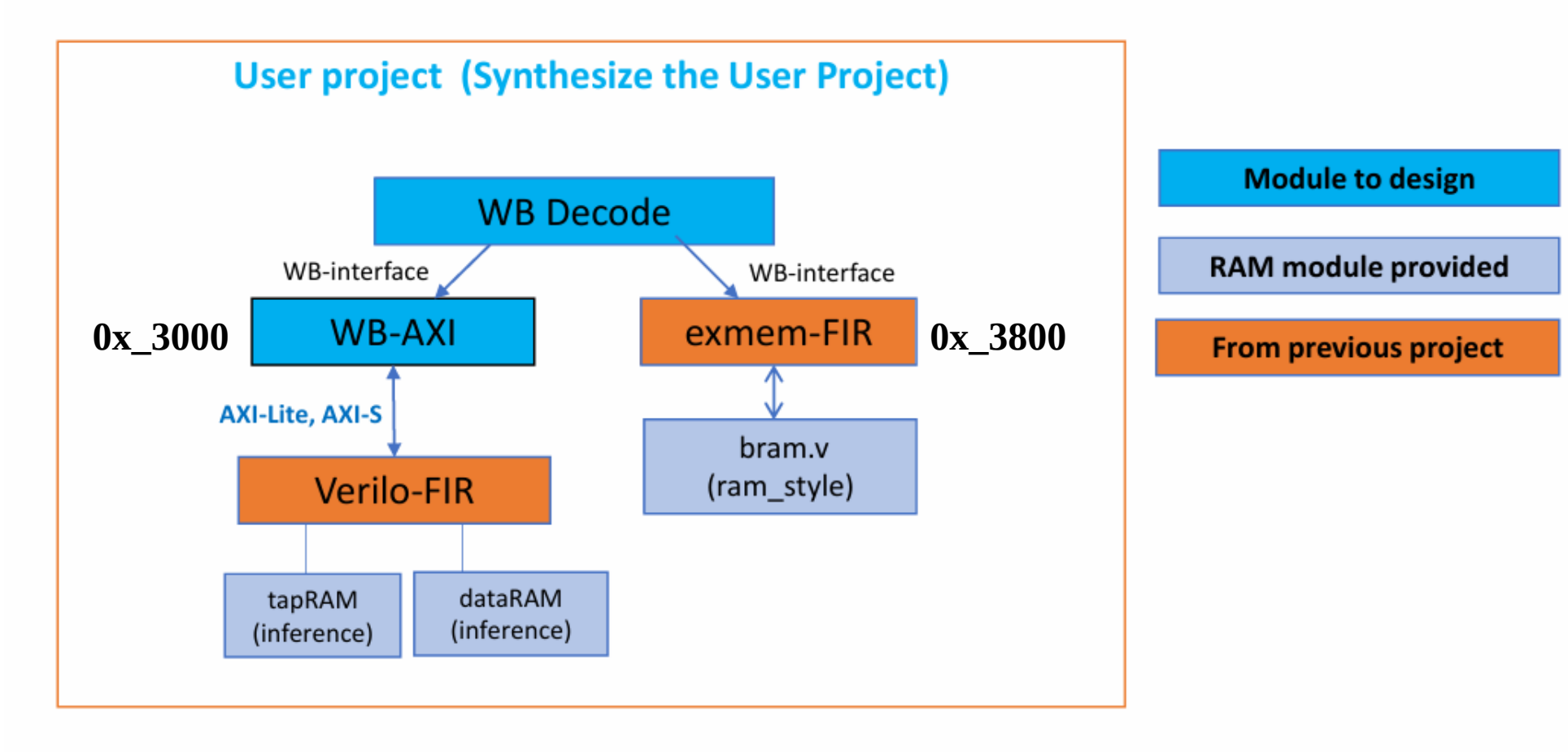
Lab4 Caravel-FIR Hardware-Software Codesign

陳瀚坪 姜維 吳鎧帆

Block Diagram

- Datapath



AXI read and write

```
assign arvalid = fir_addr && wbs_cyc_i && wbs_stb_i && (!wbs_we_i);
assign rready = fir_addr && wbs_cyc_i && wbs_stb_i && (!wbs_we_i);
assign araddr = wbs_adr_i;
//read ack
assign wbs_r_ack = rvalid && rready;
//read data,從output rdata中把data存到wbs_dat_o
assign wbs_r_dat = rdata;
```

```
assign awvalid = fir_addr && wbs_cyc_i && wbs_stb_i && wbs_we_i && (!stream_in) && (!stream_out);
assign wdata = wbs_dat_i;
assign wvalid = fir_addr && wbs_cyc_i && wbs_stb_i && wbs_we_i && (!stream_in) && (!stream_out);
assign awaddr = wbs_adr_i;
//write ack
assign wbs_w_ack = wready && wvalid;
//write data,把input wdata連到wbs_dat_i
assign wdata = wbs_dat_i;
```

stream in and out

```
//------------------------stream in-------------------------------

    assign ss_tvalid = stream_in && wbs_cyc_i && wbs_stb_i ;
    assign ss_tdata = wbs_dat_i;
    //assign ss_tlast = ;
    assign wbs_si_ack = ss_tready && ss_tvalid;
    //assign wbs_si_dat = ; dont care

//------------------------stream out-------------------------------

    assign sm_tready = stream_out && wbs_cyc_i && wbs_stb_i ;
    assign wbs_so_ack = sm_tvalid && sm_tready;
    assign wbs_so_dat = sm_tdata;
```

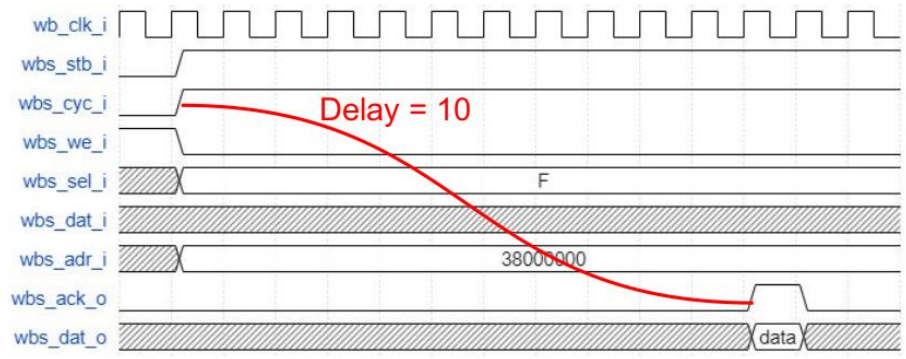
```
//----------------------------select output-------------
    always@(*) begin
        if(fir_addr) begin
            if(wbs_we_i) begin
                if(wbs_adr_i[7:0] == 8'h80) begin
                    temp1 = wbs_si_ack;
                    temp2 = 32'd0;
                end
                else begin
                    temp1 = wbs_w_ack;
                    temp2 = 32'd0;
                end
            end
            else begin
                if(wbs_adr_i[7:0] == 8'h84) begin
                    temp1 = wbs_so_ack;
                    temp2 = wbs_so_dat;
                end
                else begin
                    temp1 = wbs_r_ack;
                    temp2 = wbs_r_dat;
                end
            end
        end
        //user project
        else begin
            temp1 = usr1;
            temp2 = usr2;
        end
    end
```

Block Diagram

- Exmem FIR

```
80      reg  [ 3:0] delay_cnt;
81      wire [ 3:0] bram_we   = wbs_sel_i & {4{wbs_we_i}};
82      wire        bram_en   = wbs_cyc_i & wbs_stb_i & (wbs_adr_i[31:16] == 16'h3800);
83      wire [31:0] bram_di   = wbs_dat_i;
84      wire [31:0] bram_do;
85      wire [31:0] bram_adr  = wbs_adr_i;
86      assign      wbs_dat_o = bram_do;
87      assign      wbs_ack_o = (delay_cnt == 10)? 1 : 0;
88
89      always@(posedge wb_clk_i or posedge wb_rst_i) begin
90          if(wb_rst_i) delay_cnt <= 0;
91          else if(bram_en && delay_cnt == 10) delay_cnt <= 0;
92          else if(bram_en && delay_cnt < 10)  delay_cnt <= delay_cnt + 1;
93          else delay_cnt <= 0;
94      end
95
96      bram user_bram (
97          .CLK(wb_clk_i),
98          .WE0(bram_we),
99          .EN0(bram_en),
100         .Di0(bram_di),
101         .Do0(bram_do),
102         .A0(bram_adr)
103     );
```

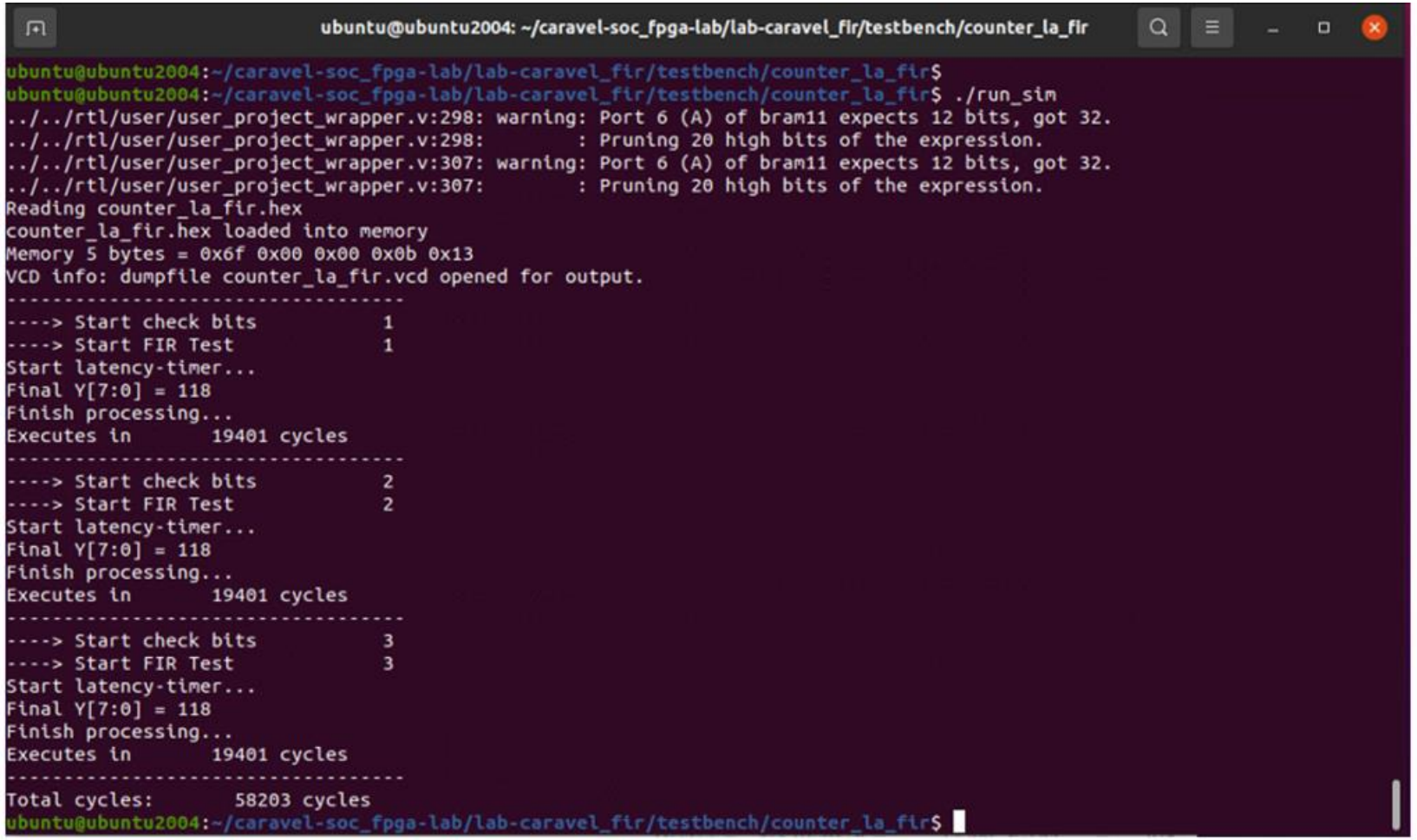
```
input wb_clk_i,
input wb_rst_i,
input wbs_stb_i,
input wbs_cyc_i,
input wbs_we_i,
input [3:0] wbs_sel_i,
input [31:0] wbs_dat_i,
input [31:0] wbs_adr_i,
output wbs_ack_o,
output [31:0] wbs_dat_o,
```



Read BRAM

Waveform

- Waveform (303T / per Y)
- 32’ d10614 = 32’ h2976 ($76_h = 118_d$)



Waveform (Instruction Reorder)

- Waveform (243T / per Y)
- 36cycle(SS) + 67cycle(Fir.v) + 140cycle(SM)

```
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab/lab-caravel_fir/testbench/counter_la_fir$ ./run_sim
../../rtl/user/user_project_wrapper.v:320: warning: Port 6 (A) of bram11 expects 12 bits, got 32.
../../rtl/user/user_project_wrapper.v:320:        : Pruning 20 high bits of the expression.
../../rtl/user/user_project_wrapper.v:331: warning: Port 6 (A) of bram11 expects 12 bits, got 32.
../../rtl/user/user_project_wrapper.v:331:        : Pruning 20 high bits of the expression.
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.
---------------------------------
----> Start check bits           1
----> Start FIR Test             1
Start latency-timer...
Final Y[7:0] = 118
Finish processing...
Executes in       16034 cycles
---------------------------------
----> Start check bits           2
----> Start FIR Test             2
Start latency-timer...
Final Y[7:0] = 118
Finish processing...
Executes in       16034 cycles
---------------------------------
----> Start check bits           3
----> Start FIR Test             3
Start latency-timer...
Final Y[7:0] = 118
Finish processing...
Executes in       16034 cycles
---------------------------------
Total cycles:       48102 cycles
```

Waveform (Instruction Reorder & -O2)

- Waveform (23T / per Y)
- 9cycle(SS) + 14cycle(Fir.v)

```
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab/lab-caravel_fir/testbench/counter_la_fir$ ./run_sim
../../rtl/user/user_project_wrapper.v:320: warning: Port 6 (A) of bram11 expects 12 bits, got 32.
../../rtl/user/user_project_wrapper.v:320:        : Pruning 20 high bits of the expression.
../../rtl/user/user_project_wrapper.v:331: warning: Port 6 (A) of bram11 expects 12 bits, got 32.
../../rtl/user/user_project_wrapper.v:331:        : Pruning 20 high bits of the expression.
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.
-----------------------------------
----> Start check bits           1
----> Start FIR Test             1
Start latency-timer...
Final Y[7:0] = 118
Finish processing...
Executes in        1700 cycles
-----------------------------------
----> Start check bits           2
----> Start FIR Test             2
Start latency-timer...
Final Y[7:0] = 118
Finish processing...
Executes in        1700 cycles
-----------------------------------
----> Start check bits           3
----> Start FIR Test             3
Start latency-timer...
Final Y[7:0] = 118
Finish processing...
Executes in        1700 cycles
-----------------------------------
Total cycles:        5100 cycles
```

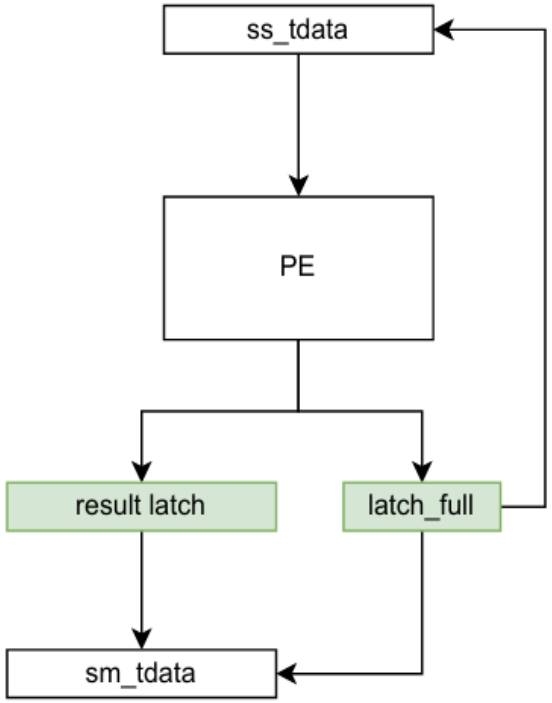
Waveform (Instruction Reorder 2 & -O2)

- Waveform (16T / per Y)
- 2cycle(SS) + 14cycle(Fir.v)

```
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab/lab-caravel_fir/testbench/counter_la_fir$ ./run_sim
../../rtl/user/user_project_wrapper.v:320: warning: Port 6 (A) of bram11 expects 12 bits, got 32.
../../rtl/user/user_project_wrapper.v:320:        : Pruning 20 high bits of the expression.
../../rtl/user/user_project_wrapper.v:331: warning: Port 6 (A) of bram11 expects 12 bits, got 32.
../../rtl/user/user_project_wrapper.v:331:        : Pruning 20 high bits of the expression.
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.
-----------------------------------
----> Start check bits           1
----> Start FIR Test             1
Start latency-timer...
Final Y[7:0] = 118
Finish processing...
Executes in        1370 cycles
-----------------------------------
----> Start check bits           2
----> Start FIR Test             2
Start latency-timer...
Final Y[7:0] = 118
Finish processing...
Executes in        1370 cycles
-----------------------------------
----> Start check bits           3
----> Start FIR Test             3
Start latency-timer...
Final Y[7:0] = 118
Finish processing...
Executes in        1370 cycles
-----------------------------------
Total cycles:        4110 cycles
```

Improve FIR

Improve FIR



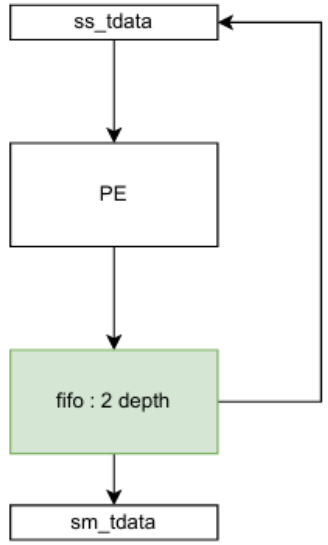
full - ss_tready

ss_tready = !latch_full

pe result latch , latch full <= 1

sm_tvalid &sm_tready & latch_full , latch full <= 0

next ss_tdata need to wait current sm_tvalid



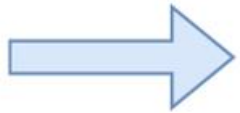
full - ss_tready

have 2 depth fifo can pass next ss_tdata ready before current sm_tdata valid.

So, it can do that, change the next x input before the current y output.

Fir.c

```
25      wb_write(reg_fir_x_in, t);
26      for (uint8_t t = 1; t < N; t++) {
27          temp = wb_read(reg_fir_y_out);
28          wb_write(reg_fir_x_in, t);
29          outputsignal[t - 1] = temp;
30      }
```



```
25      wb_write(reg_fir_x_in, t);
26      for (uint8_t t = 1; t < N; t++) {
27          wb_write(reg_fir_x_in, t);
28          temp = wb_read(reg_fir_y_out);
29          outputsignal[t - 1] = temp;
30      }
```

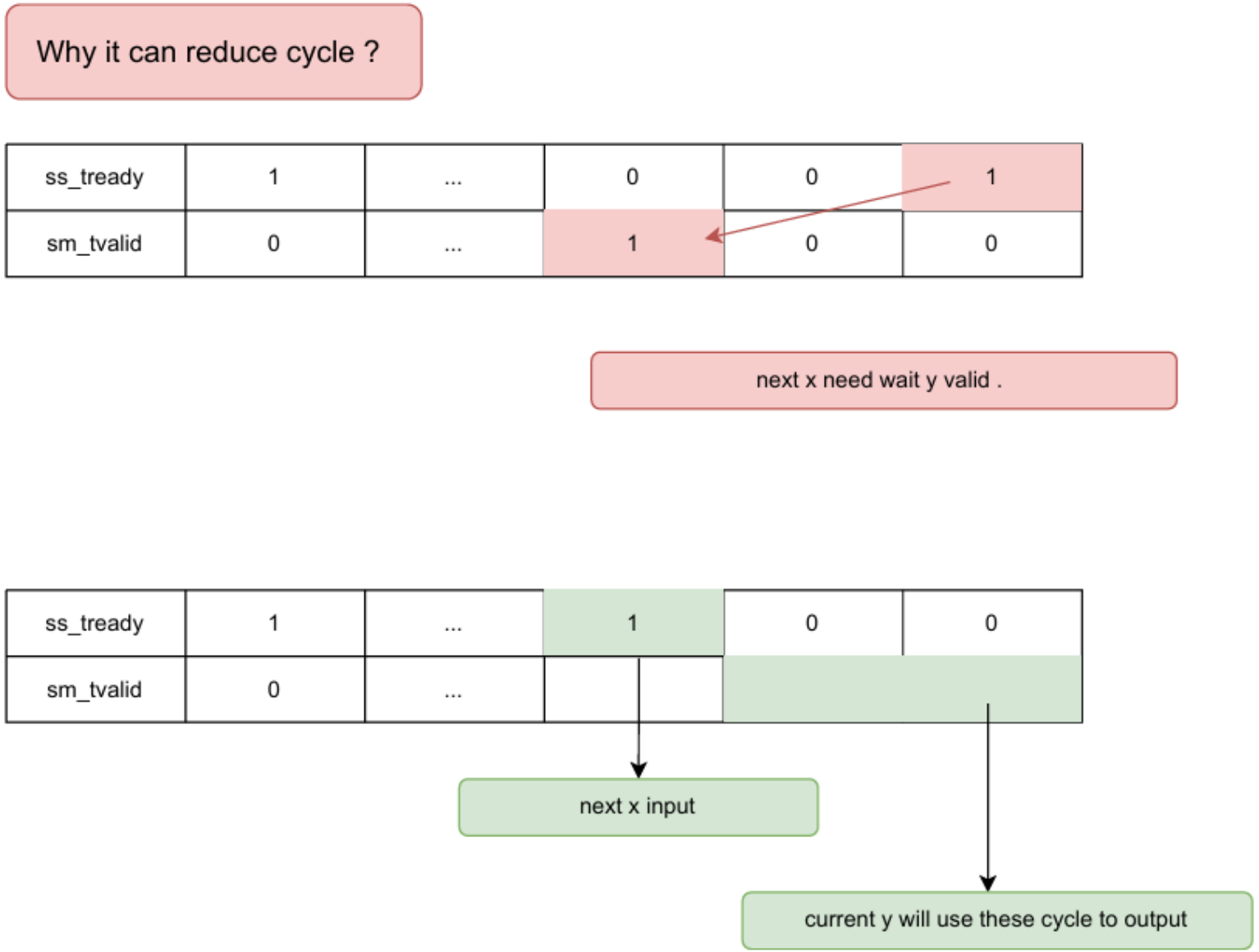
```
.L6:
    lw  a2,132(a3)
    sw  a5,128(a3)
    addi    a5,a5,1
    sw  a2,0(a4)    # store to array !!!
    addi    a4,a4,4
```

```
.L6:
    sw  a5,128(a3)
    addi    a5,a5,1
    sw  a2,0(a4)    # store to array !!!
    addi    a4,a4,4
    lw  a2,132(a3)
```

lw can't after sw x and sw array

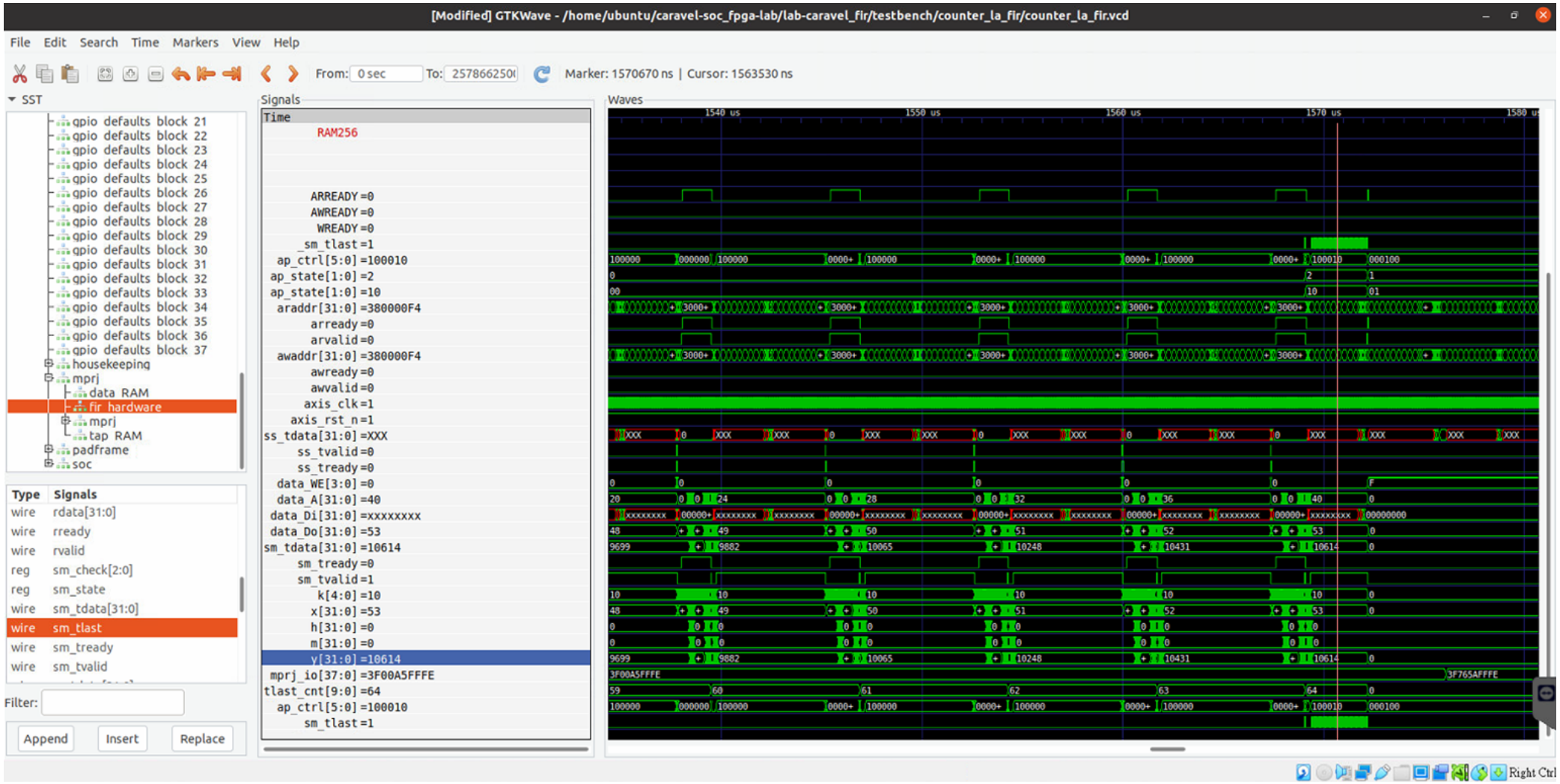
it will use x operation cycles(11 cycles) to do these thing

Why it can reduce cycle ?

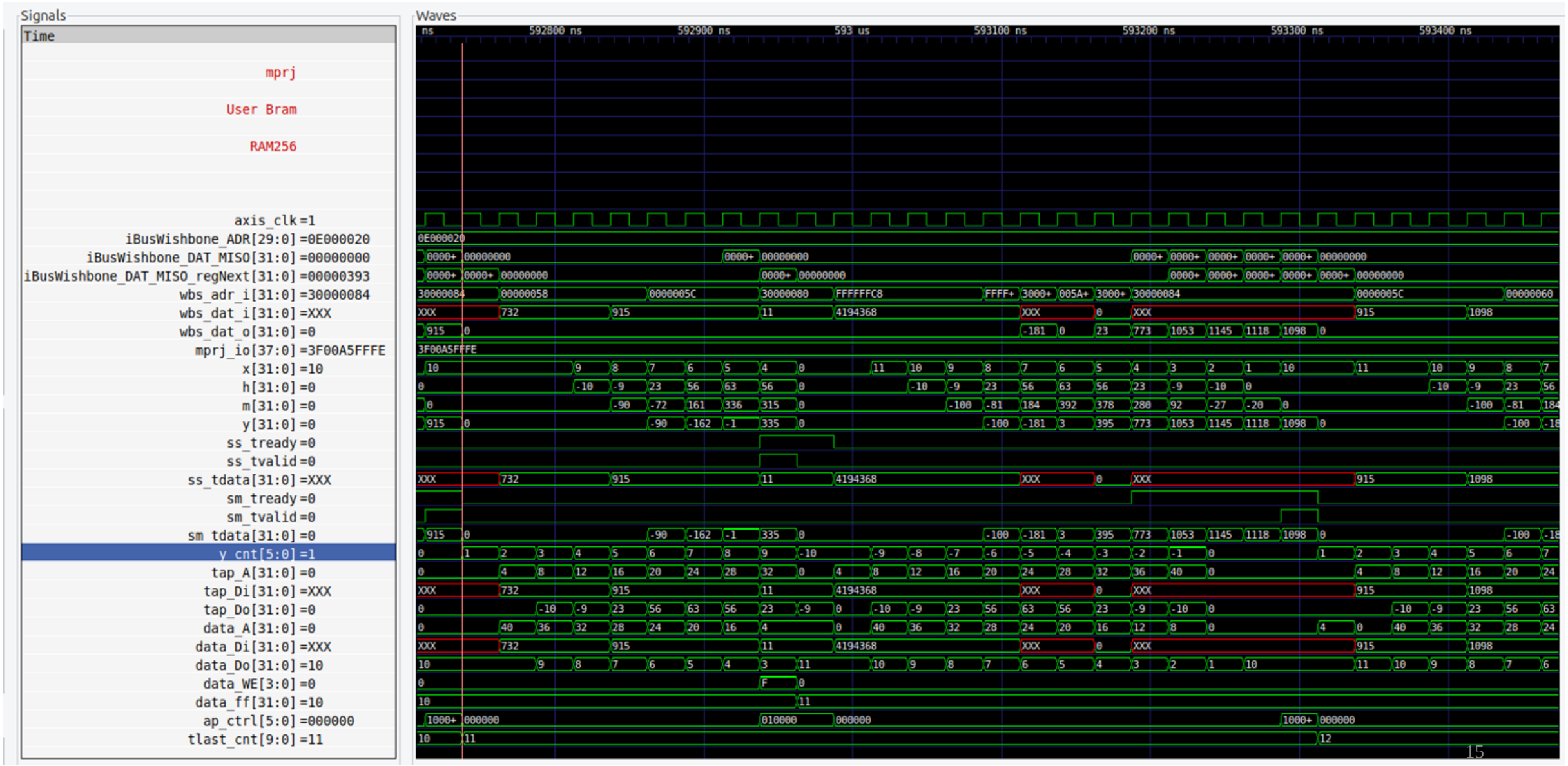


Waveform

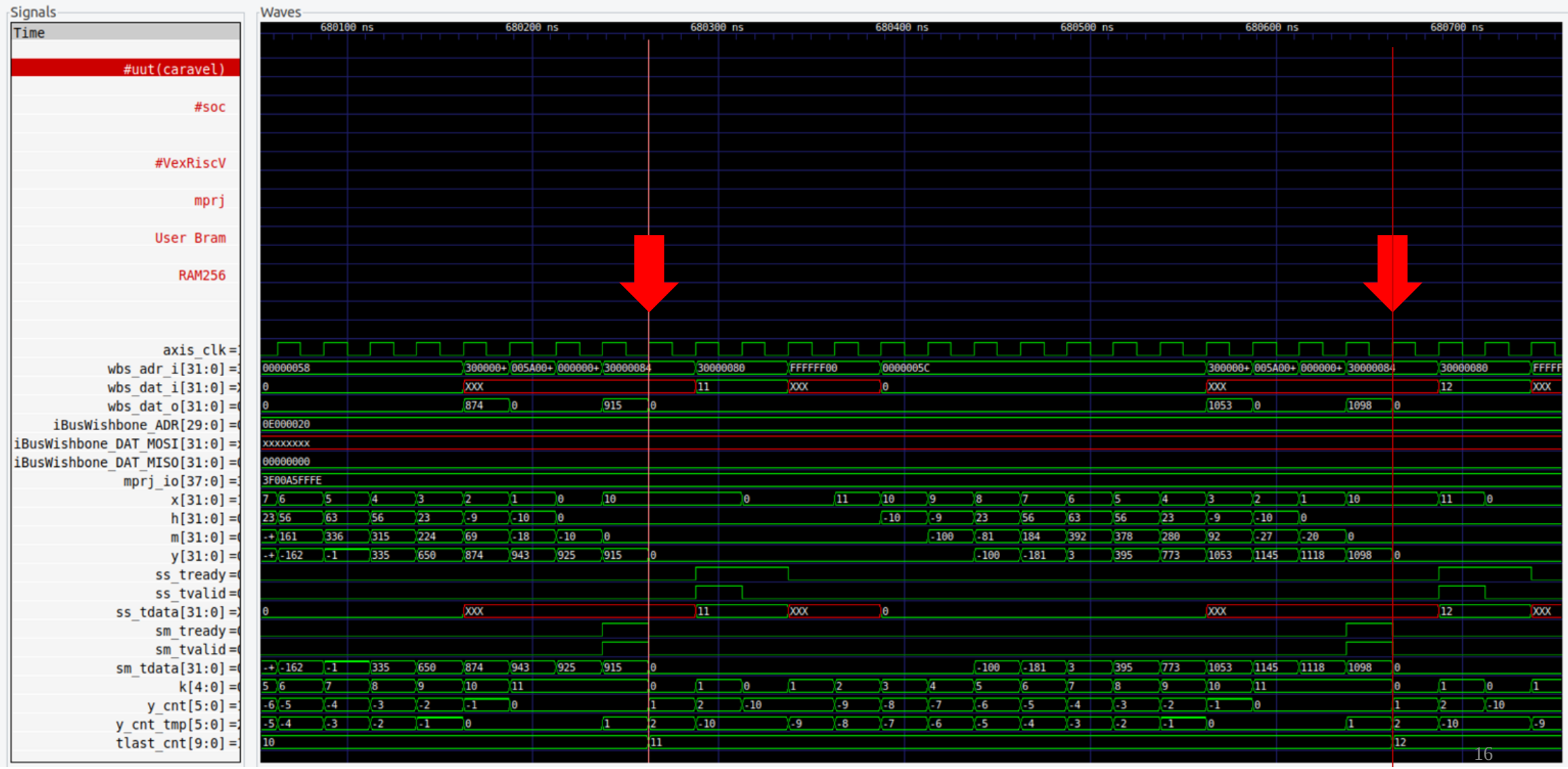
- Waveform



Waveform (Optimization)



Waveform (Optimization 2)



Fir.c (Optimization)

```c
#include "fir.h"
#include <stdint.h>

void __attribute__ ( ( section ( ".mprjram" ) ) ) initfir() {

        // program data length
        reg_fir_len = data_length;

        // program coefficient
        for (uint32_t i = 0; i < 11; i++) {
                wb_write(reg_fir_coeff + 4*i , taps[i]);
        }
}

int* __attribute__ ( ( section ( ".mprjram" ) ) ) fir_excute() {

        // StartMark
        reg_mprj_datal = 0x00A50000;

        // ap_start
        reg_fir_ap_ctrl = 1;

        uint8_t register t = 0;

        while (t < data_length) {
                reg_fir_x_in = t;
                outputsignal[t] =  reg_fir_y_out;
                t = t + 1;
        }

        // check the final Y by using    MPRJ[31:24]
        // send the EndMark 5A signal at MPRJ[23:16]
        reg_fir_ap_ctrl; // check ap_done
        checkbits = outputsignal[N-1] << 24 | 0x005A0000;

        return outputsignal;
}
```

```c
#include "fir.h"
#include <stdint.h>

void __attribute__ ( ( section ( ".mprjram" ) ) ) initfir() {

        // program data length
        reg_fir_len = data_length;

        // program coefficient
        for (uint32_t i = 0; i < 11; i++) {
                wb_write(reg_fir_coeff + 4*i , taps[i]);
        }
}

int* __attribute__ ( ( section ( ".mprjram" ) ) ) fir_excute() {

        // StartMark
        reg_mprj_datal = 0x00A50000;

        // ap_start
        reg_fir_ap_ctrl = 1;

        uint8_t register t = 0;

        //while (t < data_length) {
        //       reg_fir_x_in = t;
        //        outputsignal[t] =  reg_fir_y_out;
        //        t = t + 1;
        //}

        reg_fir_x_in = t;
        while (t < data_length - 1) {
                outputsignal[t] =  reg_fir_y_out;
                t = t + 1;
                reg_fir_x_in = t;
        }
        outputsignal[t] =  reg_fir_y_out;
        // check the final Y by using    MPRJ[31:24]
        // send the EndMark 5A signal at MPRJ[23:16]
        reg_fir_ap_ctrl; // check ap_done
        checkbits = outputsignal[N-1] << 24 | 0x005A0000;

        return outputsignal;
}
```

```c
int* __attribute__ ( ( section ( ".mprjram" ) ) ) fir_excute() {

        // StartMark
        reg_mprj_datal = 0x00A50000;

        // ap_start
        reg_fir_ap_ctrl = 1;

        uint8_t register t = 0;
        uint8_t register tmp = 0;

        //while (t < data_length) {
        //       reg_fir_x_in = t;
        //        outputsignal[t] =  reg_fir_y_out;
        //        t = t + 1;
        //}

        //reg_fir_x_in = t;
        //while (t < data_length - 1) {
        //        outputsignal[t] =  reg_fir_y_out;
        //        t = t + 1;
        //        reg_fir_x_in = t;
        //}
        //outputsignal[t] =  reg_fir_y_out;

        reg_fir_x_in = t;
        while (t < data_length - 1) {
                tmp =  reg_fir_y_out;
                t = t + 1;
                reg_fir_x_in = t;
                outputsignal[t - 1] =  tmp;

        }
        outputsignal[t] = reg_fir_y_out;

        // check the final Y by using    MPRJ[31:24]
        // send the EndMark 5A signal at MPRJ[23:16]
        reg_fir_ap_ctrl; // check ap_done
        checkbits = outputsignal[N-1] << 24 | 0x005A0000;

        return outputsignal;
}
```

Fir.h

```
#ifndef __FIR_H__
#define __FIR_H__

#define reg_fir_ap_ctrl  (*(volatile uint32_t*)0x30000000) // ap_control
#define reg_fir_ss_ready (*(volatile uint32_t*)0x30000004)
#define reg_fir_sm_valid (*(volatile uint32_t*)0x30000008)

#define reg_fir_len      (*(volatile uint32_t*)0x30000010) // data length
#define reg_fir_coeff    0x30000040                        // Load   into TapRAM
#define reg_fir_x_in     (*(volatile uint32_t*)0x30000080) // Load X into DataRAM
#define reg_fir_y_out    (*(volatile uint32_t*)0x30000084) // Take the output Y

#define checkbits        (*(volatile uint32_t*)0x2600000C) // MPRJ I/O
#define reg_mprj_xfer    (*(volatile uint32_t*)0x26000000)
#define reg_mprj_datal   (*(volatile uint32_t*)0x2600000c)

#define N 64
#define data_length N

int taps[11] = {0,-10,-9,23,56,63,56,23,-9,-10,0};
int outputsignal[N];

#define adr_ofst(target, offset) (target + offset)
#define wb_write(target, data)   (*(volatile uint32_t*)(target)) = data // wishbone write
#define wb_read(target)          (*(volatile uint32_t*)(target))       // wishbone read


#endif
```

Fir_control.c

```c
#include <defs.h>
#include <stub.c>

extern int* initfir();
extern int* fir_excute();

void __attribute__ ( ( section ( ".mprjram" ) ) ) main()
{

        // Step 1. Initialization code
        reg_mprj_io_31 = GPIO_MODE_MGMT_STD_OUTPUT;
        reg_mprj_io_30 = GPIO_MODE_MGMT_STD_OUTPUT;
        reg_mprj_io_29 = GPIO_MODE_MGMT_STD_OUTPUT;
        reg_mprj_io_28 = GPIO_MODE_MGMT_STD_OUTPUT;
        reg_mprj_io_27 = GPIO_MODE_MGMT_STD_OUTPUT;
        reg_mprj_io_26 = GPIO_MODE_MGMT_STD_OUTPUT;
        reg_mprj_io_25 = GPIO_MODE_MGMT_STD_OUTPUT;
        reg_mprj_io_24 = GPIO_MODE_MGMT_STD_OUTPUT;
        reg_mprj_io_23 = GPIO_MODE_MGMT_STD_OUTPUT;
        reg_mprj_io_22 = GPIO_MODE_MGMT_STD_OUTPUT;
        reg_mprj_io_21 = GPIO_MODE_MGMT_STD_OUTPUT;
        reg_mprj_io_20 = GPIO_MODE_MGMT_STD_OUTPUT;
        reg_mprj_io_19 = GPIO_MODE_MGMT_STD_OUTPUT;
        reg_mprj_io_18 = GPIO_MODE_MGMT_STD_OUTPUT;
        reg_mprj_io_17 = GPIO_MODE_MGMT_STD_OUTPUT;
        reg_mprj_io_16 = GPIO_MODE_MGMT_STD_OUTPUT;

        // After setting MPRJ I/O, apply the configuration
        reg_mprj_xfer = 1;
        while (reg_mprj_xfer == 1);

        initfir();

        for (int i = 0; i < 3; i++) {
                fir_excute();
        }

}
```

Open ▾ | **defs.h** ~/caravel-soc_fpga-lab/lab-caravel_fir/firmware

```c
#ifndef _DEF_H_
#define _DEF_H_

#include <stdint.h>
#include <stdbool.h>

#include <csr.h>
#include <caravel.h>

// a pointer to this is a null pointer, but the compiler does not
// know that because "sram" is a linker symbol from sections.lds.
extern uint32_t sram;

// Pointer to firmware flash routines
extern uint32_t flashio_worker_begin;
extern uint32_t flashio_worker_end;

// Storage area (MGMT: 0x0100_0000, User: 0x0200_0000)
#define reg_rw_block0  (*(volatile uint32_t*)0x01000000)
#define reg_rw_block1  (*(volatile uint32_t*)0x01100000)
#define reg_ro_block0  (*(volatile uint32_t*)0x02000000)

// UART (0x2000_0000)
//#define reg_uart_clkdiv (*(volatile uint32_t*)0x20000000)
#define reg_uart_data   (*(volatile uint32_t*) CSR_UART_RXTX_ADDR)
#define reg_uart_txfull   (*(volatile uint32_t*) CSR_UART_TXFULL_ADDR)
#define reg_uart_enable (*(volatile uint32_t*) CSR_UART_ENABLED_OUT_ADDR)
#define reg_uart_irq_en    (*(volatile uint32_t*) CSR_UART_EV_ENABLE_ADDR)

// DEBUG (0x2000_0000)
//#define reg_uart_clkdiv (*(volatile uint32_t*)0x20000000)
#define reg_reset   (*(volatile uint32_t*) CSR_CTRL_RESET_ADDR)
#define reg_debug_data   (*(volatile uint32_t*) CSR_DEBUG_RXTX_ADDR)
#define reg_debug_txfull   (*(volatile uint32_t*) CSR_DEBUG_TXFULL_ADDR)
#define reg_debug_irq_en   (*(volatile uint32_t*) CSR_USER_IRQ_3_EV_ENABLE_ADDR)
//#define reg_debug_enable (*(volatile uint32_t*) CSR_UART_ENABLED_OUT_ADDR)
```

Open ▾ | **stub.c** ~/caravel-soc_fpga-lab/lab-caravel_fir/firmware

```c
/*
 * SPDX-FileCopyrightText: 2020 Efabless Corporation
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 * SPDX-License-Identifier: Apache-2.0
 */

void putchar(char c)
{
        if (c == '\n')
                putchar('\r');
    while (reg_uart_txfull == 1);
        reg_uart_data = c;
}

void print(const char *p)
{
        while (*p)
                putchar(*(p++));
}
```