

# SOC Design

## Lab 4-1 Execute Code in User Memory

Group no: 5

Members:

M11207415 陳謝鎧

M11207002 陳泊佑

M11107426 廖千慧

M11207328 吳奕帆

## Prepare firmware code & RTL:

Generate data in header file – fir.h:

- Define taps parameters and inputsignal as lab3 in header file.

```
// file's name: fie.v

#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
```

C code – fir.c:

- Implement FIR function in c code.

```
// file's name : fir.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
    for (int i = 0; i < N; i++)
    {
        int data_get = inputsignal[i]; // get data from axi-stream
        inputbuffer[i] = data_get;     // store data to bram
        for (int j = 0; j <= i; j++)
        {
            outputsignal[i] += inputbuffer[j] * taps[i - j];
        }
    }
    return outputsignal;
}
```

## Firmware management in main(): (Already designed)

- In testbench/counter\_la\_fir.c, parameter reg\_mprj\_xfer will be initially to 1, and will not start fir until the external signal is given to 0.

```
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;

// Now, apply the configuration
reg_mprj_xfer = 1;
while (reg_mprj_xfer == 1);
```

## Linker for address arrangement: (Already designed)

- In firmware/section.ids, mpjram is our bram, it's original address is at 0x38000000, and it's size is 4 KB.

```
/* INCLUDE ../../generated/regions.ld */

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
    mpjram : ORIGIN = 0x38000000, LENGTH = 0x00400000
    hk : ORIGIN = 0x26000000, LENGTH = 0x00100000
    csr : ORIGIN = 0xf0000000, LENGTH = 0x00010000
}
```

## Design BRAM in user\_project

- Estimated the required size of RAM

```
// file's name : bram.v

module bram(
    CLK,
    WE0,
    EN0,
    Di0,
    Do0,
    A0
);

    input wire CLK;
    input wire [3:0] WE0;
    input wire EN0;
    input wire [31:0] Di0;
    output reg [31:0] Do0;
    input wire [31:0] A0;

    // 16KB (// 4 kB)
    // 32 bit = 4 byte -> 16KB = 4 byte * 2 ^ N
    parameter N = 12;
    (* 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
```

- Design the controller connected with wishbone bus and ack response need to after Delay (10 delays)

```
// file's name : user_proj_example.counter.v

`define MPRJ_IO_PADS_1 19 /* number of user GPIO pads on user1 side */
`define MPRJ_IO_PADS_2 19 /* number of user GPIO pads on user2 side */
`define MPRJ_IO_PADS (`MPRJ_IO_PADS_1 + `MPRJ_IO_PADS_2)

`default_nettype wire

module user_proj_example #(
    parameter BITS = 32,
    parameter DELAYS=10
)()
`ifdef USE_POWER_PINS
    inout vccd1, // User area 1 1.8V supply
    inout vssd1, // User area 1 digital ground
`endif

    // Wishbone Slave ports (WB MI A)
    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,

    // Logic Analyzer Signals
    input [127:0] la_data_in,
    output [127:0] la_data_out,
    input [127:0] la_oenb,

    // I/Os
    input [`MPRJ_IO_PADS-1:0] io_in,
    output [`MPRJ_IO_PADS-1:0] io_out,
    output [`MPRJ_IO_PADS-1:0] io_oeb,

    // IRQ
    output [2:0] irq
);
    wire clk;
    wire rst;
    assign clk = wb_clk_i;
    assign rst = wb_rst_i;

    //wire [`MPRJ_IO_PADS-1:0] io_in;
    //wire [`MPRJ_IO_PADS-1:0] io_out;
    //wire [`MPRJ_IO_PADS-1:0] io_oeb;

    // input to ram
    wire ram_en;
    wire [3:0] ram_we;
    wire [31:0] ram_adr;
    wire [31:0] ram_data;

    bram user_bram (
        .CLK(clk),
        .WE0(ram_we),
        .EN0(ram_en),
        .Di0(wbs_dat_i),
        .Do0(wbs_dat_o),
        .A0(wbs_adr_i)
    );

    // write data to on_chip ram only when request_sig assert
    wire request_sig;
    assign request_sig = wbs_cyc_i & wbs_stb_i;
    assign ram_adr = (request_sig==1'b1)? wbs_adr_i : 32'b0;
    assign ram_data = (request_sig==1'b1)? wbs_dat_i : 32'b0;
    assign ram_we = (request_sig==1'b1)? ({4{wbs_we_i}} & wbs_sel_i) : 4'b0;
    assign ram_en = (request_sig==1'b1)? (wbs_cyc_i & wbs_sel_i) : 1'b0;

    reg wbs_ack_o;
    reg [3:0] delay_cnt; // delay = 10 (DELAYS) < 2^4
    always @ (posedge clk) begin
        if (rst) begin
            wbs_ack_o <= 0;
            delay_cnt <= 0;
        end
        else if (request_sig == 1'b1) begin
            if (delay_cnt == DELAYS) begin
                wbs_ack_o <= 1'b1;
                delay_cnt <= 0;
            end
            else begin
                wbs_ack_o <= 1'b0;
                delay_cnt <= delay_cnt + 1 ;
            end
        end
        else
            wbs_ack_o <= 1'b0;
    end

endmodule

`default_nettype wire
```

## Compilation

- Run\_clean

```
// file's name: run_clean
rm -rf ./gdb.debug ./gdbwave.debug
rm -f *.vcd *.hex
rm -f *.s *.o *.i *.out *.map
```

## ➤ Run\_sim

```
// file's name: run_sim

rm -f counter_la_fir.hex

// Given script to compile
riscv32-unknown-elf-gcc -Wl,--no-warn-rwx-segments -g \
--save-temps \
-Xlinker -Map=output.map \
-I../firmware \
-march=rv32i -mabi=ilp32 -D_vexriscv__ \
-Wl,-Bstatic,-T,../firmware/sections.lds,--strip-discarded \
-ffreestanding -nostartfiles -o counter_la_fir.elf ../firmware/crt0_vex.S ../firmware/isr.c fir.c counter_la_fir.c

# -nostartfiles
riscv32-unknown-elf-objcopy -O verilog counter_la_fir.elf counter_la_fir.hex // Transfer .elf to .hex
riscv32-unknown-elf-objdump -D counter_la_fir.elf > counter_la_fir.out // Export assembly code for debugging

# to fix flash base address
sed -ie 's/@10/@00/g' counter_la_fir.hex

iverilog -Ttyp -DFUNCTIONAL -DSIM -DUNIT_DELAY=#1 \
-f./include.rtl.list -o counter_la_fir.vvp counter_la_fir.tb.v

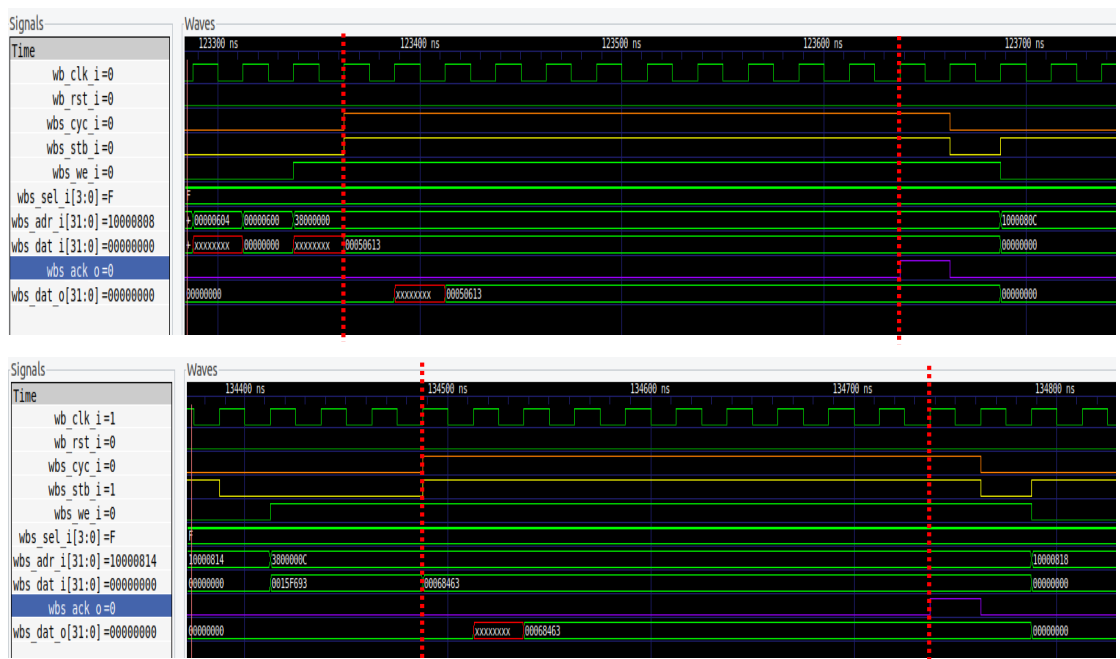
vvp counter_la_fir.vvp
rm -f counter_la_fir.vvp counter_la_fir.elf counter_la_fir.hex
```

## ➤ Compilation

```
ubuntu@ubuntu2004:~/lab-exmem_fir_Emma/testbench/counter_la_fir$ source run_clean
ubuntu@ubuntu2004:~/lab-exmem_fir_Emma/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:~/lab-exmem_fir_Emma/testbench/counter_la_fir$
```

## Synthesis & Verification

### ➤ Waveform – Wishbone's ack will have a 10cycle delay when writing.



## ➤ Timing report

Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.

```
-----
| Tool Version      : Vivado v.2022.1 (lin64) Build 3526262 Mon Apr 18 15:47:01 MDT 2022
| Date             : Wed Nov  8 12:25:34 2023
| Host             : ubuntu2004 running 64-bit Ubuntu 20.04.4 LTS
| Command          : report_timing_summary -delay_type min_max -report_unconstrained -check_timing_verbos -max_paths 10 -
input_pins -routable_nets -name timing_5 -file /home/ubuntu/Desktop/timing_rpt.txt
| Design           : user_proj_example
| Device           : xck26-sfvc784
| Speed File       : -2LV PRODUCTION 1.29 00-03-2020
| Temperature Grade : C
-----
```

### Timing Summary Report

#### | Timer Settings

```
-----
|
| Enable Multi Corner Analysis      : Yes
| Enable Pessimism Removal         : Yes
| Pessimism Removal Resolution     : Nearest Common Node
| Enable Input Delay Default Clock : No
| Enable Preset / Clear Arcs      : No
| Disable Flight Delays           : No
| Ignore I/O Paths                : No
| Timing Early Launch at Borrowing Latches : No
| Borrow Time for Max Delay Exceptions : Yes
| Merge Timing Exceptions         : Yes
|
-----
```

Corner Name	Analyze Max Paths	Analyze Min Paths
Slow	Yes	Yes
Fast	Yes	Yes

#### | Report Methodology

No report available as report\_methodology has not been run prior. Run report\_methodology on the current design for the summary of methodology violations.

### check\_timing report

#### Table of Contents

1. checking no\_clock (13)
2. checking constant\_clock (0)
3. checking pulse\_width\_clock (0)
4. checking unconstrained\_internal\_endpoints (154)
5. checking no\_input\_delay (52)
6. checking no\_output\_delay (33)
7. checking multiple\_clock (0)
8. checking generated\_clocks (0)
9. checking loops (0)
10. checking partial\_input\_delay (0)
11. checking partial\_output\_delay (0)
12. checking latch\_loops (0)

1. checking no\_clock (13)

There are 13 register/latch pins with no clock driven by root clock pin: wb\_clk\_i (HIGH)

#### | Timing Details

Path Group: (none)  
From Clock:  
To Clock:

Max Delay 187 Endpoints  
Min Delay 187 Endpoints

#### Max Delay Paths

```
Slack: inf
Source: user_bram/RAM_reg_2/CLKBWRCLK
        (rising edge-triggered cell RAMB36E2)
Destination: wbs_dat_o[17]
              (output port)
Path Group: (none)
Path Type: Max at Slow Process Corner
Data Path Delay: 5.782ns (logic 4.091ns (70.752%) route 1.691ns (29.248%))
Logic Levels: 2 (OBUF=1 RAMB36E2=1)

Location      Delay type      Incr(ns)  Path(ns)  Netlist Resource(s)
-----
RAMB36E2      RAMB36E2 (Prop_RAMB36E2_CLKBWRCLK_DOUTBDOUT[1])  0.000      0.000 r user_bram/RAM_reg_2/CLKBWRCLK
net (fo=1, unplaced)  1.387      1.387 r user_bram/RAM_reg_2/DOUTBDOUT[1]
net (fo=1, unplaced)  1.691      3.078 r wbs_dat_o_OBUF[17]
OBUF (Prop_OBUF_I_0)  2.704      5.782 r wbs_dat_o_OBUF[17]_inst/I
net (fo=0)         0.000      5.782 r wbs_dat_o[17]
net (fo=0)         0.000      5.782 r wbs_dat_o[17] (OUT)
```

```
Slack: inf
Source: user_bram/RAM_reg_0/CLKBWRCLK
        (rising edge-triggered cell RAMB36E2)
Destination: wbs_dat_o[1]
              (output port)
Path Group: (none)
Path Type: Max at Slow Process Corner
Data Path Delay: 5.782ns (logic 4.091ns (70.752%) route 1.691ns (29.248%))
Logic Levels: 2 (OBUF=1 RAMB36E2=1)

Location      Delay type      Incr(ns)  Path(ns)  Netlist Resource(s)
-----
RAMB36E2      RAMB36E2 (Prop_RAMB36E2_CLKBWRCLK_DOUTBDOUT[1])  0.000      0.000 r user_bram/RAM_reg_0/CLKBWRCLK
net (fo=1, unplaced)  1.387      1.387 r user_bram/RAM_reg_0/DOUTBDOUT[1]
net (fo=1, unplaced)  1.691      3.078 r wbs_dat_o_OBUF[1]
OBUF (Prop_OBUF_I_0)  2.704      5.782 r wbs_dat_o_OBUF[1]_inst/I
net (fo=0)         0.000      5.782 r wbs_dat_o[1]
net (fo=0)         0.000      5.782 r wbs_dat_o[1] (OUT)
```

## ➤ Synthesis report

Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.

Tool Version : Vivado v.2022.1 (lin64) Build 3526262 Mon Apr 18 15:47:01 MDT 2022

Date : Wed Nov 8 12:26:34 2023

Host : ubuntu2004 running 64-bit Ubuntu 20.04.4 LTS

Command : report\_utilization -file /home/ubuntu/Desktop/utilization\_rpt.txt -name utilization\_2

Design : user\_proj\_example

Device : xck26-sfvc784-2LV-c

Speed File : -2LV

Design State : Synthesized

-----

Utilization Design Information

-----

Table of Contents

-----

1. CLB Logic

1.1 Summary of Registers by Type

2. BLOCKRAM

3. ARITHMETIC

4. I/O

5. CLOCK

6. ADVANCED

7. CONFIGURATION

8. Primitives

Site Type	Used	Fixed	Prohibited	Available	Util%
CLB LUTs*	7	0	0	117120	<0.01
LUT as Logic	7	0	0	117120	<0.01
LUT as Memory	0	0	0	57600	0.00
CLB Registers	5	0	0	234240	<0.01
Register as Flip Flop	5	0	0	234240	<0.01
Register as Latch	0	0	0	234240	0.00
CARRY8	0	0	0	14640	0.00
F7 Muxes	0	0	0	58560	0.00
F8 Muxes	0	0	0	29280	0.00
F9 Muxes	0	0	0	14640	0.00

\* Warning! The Final LUT count, after physical optimizations and full implementation, is typically lower. Run opt\_design after synthesis, if not already completed, for a more realistic count.

1.1 Summary of Registers by Type

Total	Clock Enable	Synchronous	Asynchronous
0	-	-	Set
0	-	-	Reset
0	-	Set	-
0	-	Reset	-
0	Yes	-	-
0	Yes	-	Set
0	Yes	-	Reset
0	Yes	Set	-
5	Yes	Reset	-

2. BLOCKRAM

Site Type	Used	Fixed	Prohibited	Available	Util%
Block RAM Tile	4	0	0	144	2.78
RAMB36/FIFO*	4	0	0	144	2.78
RAMB36E2 only	4	0	0	0	0.00
RAMB18	0	0	0	288	0.00
URAM	0	0	0	64	0.00

\* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E2 or one FIFO18E2. However, if a FIFO18E2 occupies a Block RAM Tile, that tile can still accommodate a RAMB18E2

3. ARITHMETIC

Site Type	Used	Fixed	Prohibited	Available	Util%
DSPs	0	0	0	1248	0.00

4. I/O

Site Type	Used	Fixed	Prohibited	Available	Util%
Bonded IOB	293	0	0	189	155.03

5. CLOCK

Site Type	Used	Fixed	Prohibited	Available	Util%
GLOBAL CLOCK BUFFERS	1	0	0	352	0.28
BUFGCE	1	0	0	112	0.09
BUFGCE_DIV	0	0	0	16	0.00
BUFG_GT	0	0	0	96	0.00
BUFG_PS	0	0	0	96	0.00
BUFGCTRL*	0	0	0	32	0.00
PLL	0	0	0	8	0.00
MCM	0	0	0	4	0.00

\* Note: Each used BUFGCTRL counts as two GLOBAL CLOCK BUFFERS. This table does not include global clocking resources, only buffer cell usage. See the Clock Utilization Report (report\_clock\_utilization) for detailed accounting of global clocking resource availability.

6. ADVANCED

Site Type	Used	Fixed	Prohibited	Available	Util%
GTHE4_CHANNEL	0	0	0	4	0.00
GTHE4_COMMON	0	0	0	1	0.00
OBUFDS_GTE4	0	0	0	2	0.00
OBUFDS_GTE4_ADV	0	0	0	2	0.00
PCIE4BE4	0	0	0	2	0.00
PS8	0	0	0	1	0.00
SYSMONE4	0	0	0	1	0.00
VCU	0	0	0	1	0.00

7. CONFIGURATION

Site Type	Used	Fixed	Prohibited	Available	Util%
BSCAN62	0	0	0	4	0.00
DNA_PORT62	0	0	0	1	0.00
EFUSE_USR	0	0	0	1	0.00
FRAME_ECCE4	0	0	0	1	0.00
ICAPE3	0	0	0	2	0.00
MASTER_JTAG	0	0	0	1	0.00
STARTUP63	0	0	0	1	0.00

8. Primitives

Ref Name	Used	Functional Category
OBUFF	287	I/O
INBUF	53	I/O
IBUFCTRL	53	Others
OBUF	33	I/O
LUT4	7	CLB
PDRE	5	Register
RAMB36E2	4	BLOCKRAM
LUT3	3	CLB
LUT6	1	CLB
LUT2	1	CLB
BUFGCE	1	Clock



## ➤ Timing Report

Intelligent Design Runs

Design Runs

Timing

Design Timing Summary

Setup

Worst Negative Slack (WNS): 9.366 ns

Total Negative Slack (TNS): 0.000 ns

Number of Failing Endpoints: 0

Total Number of Endpoints: 5

Hold

Worst Hold Slack (WHS): 0.000 ns

Total Hold Slack (THS): 0.000 ns

Number of Failing Endpoints: 0

Total Number of Endpoints: 5

Pulse Width

Worst Pulse Width Slack (WPWS): 4.423 ns

Total Pulse Width Negative Slack (TPWS): 0.000 ns

Number of Failing Endpoints: 0

Total Number of Endpoints: 14

All user specified timing constraints are met.

Intelligent Design Runs

Design Runs

Timing

Intra-Clock Paths - wb\_clk\_i - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock
Path 1	9.366	1	2	5	delay_cnt_reg[2]/C	delay_cnt_reg[3]/D	0.498	0.306	0.192	10.0	wb_clk_i	wb_clk_i
Path 2	9.405	1	2	5	delay_cnt_reg[2]/C	delay_cnt_reg[2]/D	0.459	0.268	0.191	10.0	wb_clk_i	wb_clk_i
Path 3	9.406	1	2	5	delay_cnt_reg[2]/C	delay_cnt_reg[0]/D	0.458	0.266	0.192	10.0	wb_clk_i	wb_clk_i
Path 4	9.406	1	2	5	delay_cnt_reg[2]/C	delay_cnt_reg[1]/D	0.458	0.266	0.192	10.0	wb_clk_i	wb_clk_i
Path 5	9.406	1	2	5	delay_cnt_reg[1]/C	wbs_ack_o_reg/D	0.458	0.266	0.192	10.0	wb_clk_i	wb_clk_i

心得:

在 Lab4-0 中我們練習了 Caravel SOC simulation，在 Lab4-1 中學習設計能夠在 Caravel RISC-V core 中跑的 firmware code (exemem-fir)，以及在 user project 中整合 RAM，使其能和 firmware 及 testbench 交互作用。

在設計過程中的 compilation 時曾發生過 ”Time out, Test LA failed (RTL)”，在 github 上也看到有人發問同樣的問題，我參考了底下留言的建議，去檢查確實我們的 Bram Module 接腳是否有正確的與 Wishbone 連接，確實沒有，因此我們很快找出問題點並將其修正。

之後在 vivado 中跑 bram.v 和 user\_proj\_example.counter.v 的合成驗證時也出現了 error，error 內容為 ”use of undefined macro ‘MPRJ\_IO\_PADS’ in user\_proj\_example.counter.v.” 以及 ”net type must be explicitly specified for ‘wb\_clk\_i’ when default\_nettype is none.” 同樣的在 github 上已經有同學遇到了此問題，也有同學很熱心且詳細的在底下解答，由於 MPRJ\_IO\_PADS 這個參數是定義於 rtl/header/defines.v 裡面，但我們在合成時只使用了 bram.v 和 user\_proj\_example.counter.v，因此我們需自行定義 MPRJ\_IO\_PADS 參數於 user\_proj\_example.counter.v 中。而第二個 error 是由於 ‘default\_nettype none 造成的，因此我們將 none 改為 wire 即可順利的合成。