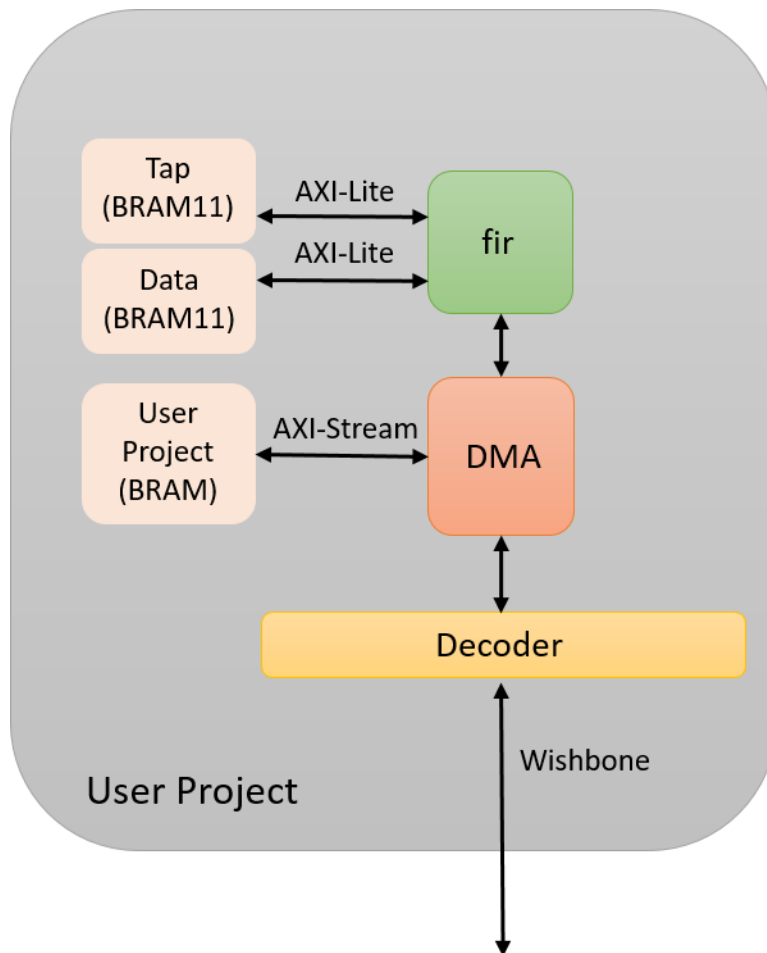


SOC Lab Final Project

交大電物碩 311651055 林柏宇

1. Block Diagram :



2. DMA :

(1) Transfer lots of Data without CPU

(2) Support 64 data, and each data is 32 bits

```
reg [31:0] rbuffer [0:63];  
reg [31:0] wbuffer [0:63];
```

(3) Used AXI-Stream to transfer datas

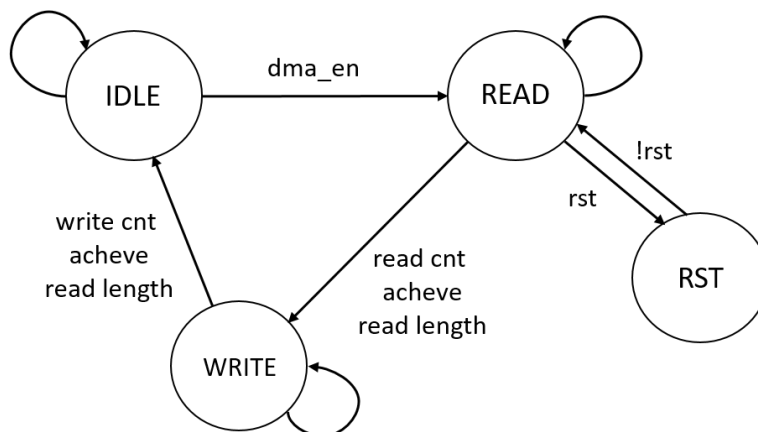
```
output reg          sm_tready,  
input  wire         sm_tvalid,  
input  wire [31:0] sm_tdata,  
input  wire         sm_tlast,  
  
output reg          ss_tlast,  
output reg [31:0] ss_tdata,  
output reg          ss_tvalid,  
input  wire         ss_tready,
```

(4) Used AXI-Stream to transfer datas

```
always@(posedge clk or negedge rst) begin  
    if(rst)  
        dma_or_wb <= 0;  
    else  
        begin  
            if(dma_busy == 1)  
                dma_or_wb <= 1;  
            else if(dma_busy == 0)  
                dma_or_wb <= 0;  
            else  
                dma_or_wb <= dma_or_wb;  
        end  
end
```

```
DMA dma_u0  
    .clk(clk),  
    .rst_n(!rst),  
    .dma_valid(dma_or_wb),  
    .dma_en(dma_en),  
    .r_start_addr(r_start_addr),  
    .w_start_addr(w_start_addr),  
    .read_len(read_len),
```

(5) State Machine



3. Configuration Register :

- Tap 存放在 0x31000040 到 0x31000068。
- Input Data 存放在 0x31000080 開始的位置。
- Output Data 存放在 0x31000084 開始的位置。
- Data Length 存放在 0x31000010。
- State (Start、Done、Idle)在 0x31000000 開始的位置。
- DMA start address：在 User_BRAM 中正確定位要讀取或寫入的數據

r_start_addr：讀取的起始位置，在 0x31000014 開始的位置。

W_start_addr：寫入的起始位置，在 0x31000018 開始的位置。

```
//input
#define inputsignal      (*(volatile uint32_t*)0x31000080)
//output
#define outputsignal     (*(volatile uint32_t*)0x31000084)
//tap
#define tap_1            (*(volatile uint32_t*)0x31000040)
#define tap_2            (*(volatile uint32_t*)0x31000044)
#define tap_3            (*(volatile uint32_t*)0x31000048)
#define tap_4            (*(volatile uint32_t*)0x3100004c)
#define tap_5            (*(volatile uint32_t*)0x31000050)
#define tap_6            (*(volatile uint32_t*)0x31000054)
#define tap_7            (*(volatile uint32_t*)0x31000058)
#define tap_8            (*(volatile uint32_t*)0x3100005c)
#define tap_9            (*(volatile uint32_t*)0x31000060)
#define tap_10           (*(volatile uint32_t*)0x31000064)
#define tap_11           (*(volatile uint32_t*)0x31000068)
//length
#define datalength       (*(volatile uint32_t*)0x31000010)
//status
#define status           (*(volatile uint32_t*)0x31000000)
//DMA Start
#define r_start_addr     (*(volatile uint32_t*)0x31000014)
#define w_start_addr     (*(volatile uint32_t*)0x31000018)
```

```
always@(*) begin
    wbs_dat_o = 0;
    wbs_ack_o = 0;
    if(wbs_cyc_i && wbs_stb_i) begin
        if(wbs_adr_i[31:24] == 'h38) begin
            wbs_dat_o = u_wbs_dat_o;
            wbs_ack_o = wbs_ack_o_user;
        end
        else if(wbs_adr_i[31:24] == 'h31) begin
            wbs_dat_o = wbs_dat_o_fir;
            wbs_ack_o = wbs_ack_o_fir;
        end
    end
end
```

4. Firmware Code :

- When fir start, configuration called fir start by 0x00000001
- fir status second bits change to 1, end

```
int* __attribute__((section(".mprjram"))) fir() {
    int current_status;
    int i = 0;
    initfir();

    current_status = status;
    while(!((current_status >> 1) & 1)) {
        current_status = status;
    }

    reg_mprj_data1 = 0xFF5A0000;

    return ans;
}

status = 0x00000001;
```

```
always@(*) begin
    case(state)
        IDLE: begin
            if(configuration[0] == 1) state_next = WAIT;
            else state_next = IDLE;
        end
        WAIT: begin
            if(ss_tvalid) state_next = WORK;
            else state_next = WAIT;
        end
        WORK: begin
            if(sm_tlast && sm_tready && sm_tvalid) state_next = IDLE;
            else state_next = WORK;
        end
        default: begin
            state_next = IDLE;
        end
    endcase
end
```

5. Add -O3 in compiler :

```
rm -f final.hex

riscv32-unknown-elf-gcc -Wl,--no-warn-rwx-segments -g \
    -O3 \
    --save-temps \
    -Xlinker -Map=output.map \
    -I../firmware \
    -march=rv32i -mabi=ilp32 -D__vexriscv__ -DUSER_PROJ_IRQ0_EN \
    -Wl,-Bstatic,-T,../firmware/sections.lds,--strip-discarded \
    -ffreestanding -nostartfiles -o final.elf ../firmware/crt0_vex.S ../firmware/isr.c operate.c final.c
# -nostartfiles
riscv32-unknown-elf-objcopy -O verilog final.elf final.hex
riscv32-unknown-elf-objdump -D final.elf > final.out

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

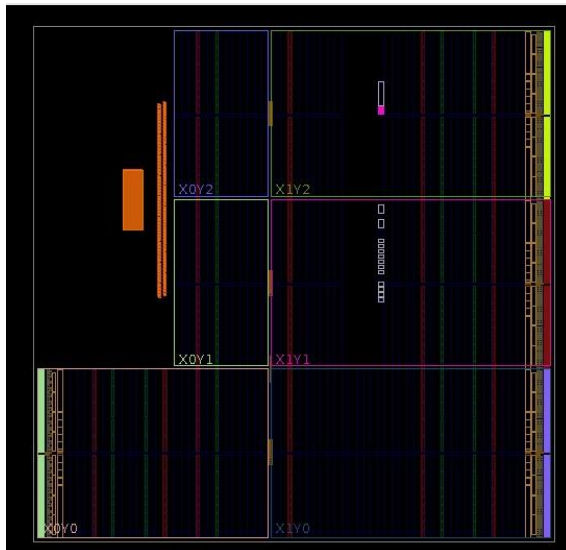
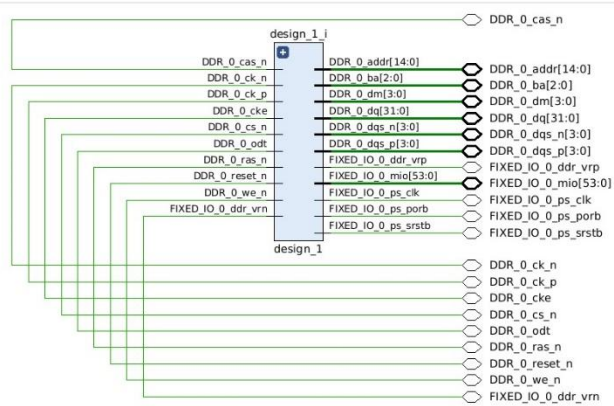
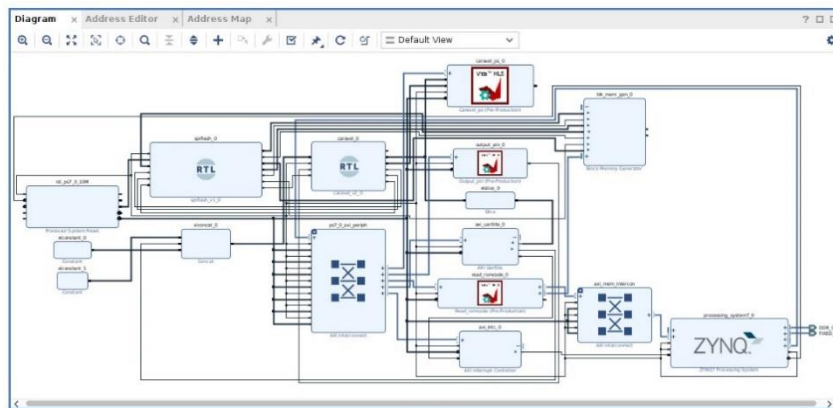
iverilog -Ttyp -DFUNCTIONAL -DSIM -DUNIT_DELAY=#1 \
    -f./include.rtl.list -o final.vvp final_tb.v

vvp -l simulation.log final.vvp
rm -f final.vvp final.elf final.hexe
```

6. Result

Fir use without -O3	839508 CLKs
Fir with -O3	325983 CLKs
Fir use DMA without -O3	173634 CLKs
Fir use DMA with -O3	31125 CLKs

7. Synthesis Report :



5. checking no_input_delay (0)

There are 0 input ports with no input delay specified.

There are 0 input ports with no input delay but user has a false path constraint.

6. checking no_output_delay (0)

There are 0 ports with no output delay specified.

There are 0 ports with no output delay but user has a false path constraint

There are 0 ports with no output delay but with a timing clock defined on it or propagating through it

7. checking multiple_clock (0)

There are 0 register/latch pins with multiple clocks.

8. checking generated_clocks (0)

There are 0 generated clocks that are not connected to a clock source.

9. checking loops (0)

There are 0 combinational loops in the design.

10. checking partial_input_delay (0)

There are 0 input ports with partial input delay specified.

11. checking partial_output_delay (0)

There are 0 ports with partial output delay specified.

12. checking latch_loops (0)

There are 0 combinational latch loops in the design through latch input

| Timing Details | -----

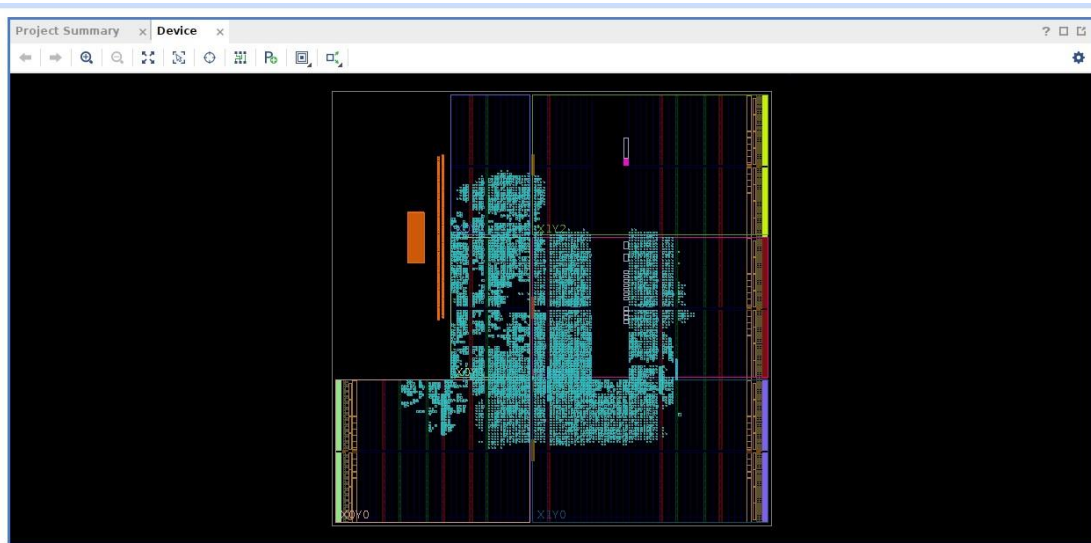
From Clock: clk_fpga_0
To Clock: clk_fpga_0

Setup :	0	Failing Endpoints, Worst Slack	10.916ns, Total Violation	0.000ns
Hold :	2	Failing Endpoints, Worst Slack	-0.762ns, Total Violation	-1.523ns
PW :	0	Failing Endpoints, Worst Slack	11.250ns, Total Violation	0.000ns

Max Delay Paths

Slack (MET) : 10.916ns (required time - arrival time)
Source: design_1_i/caravel_0/inst/mprij/fir_1/data_length_reg[1]/C
(rising edge-triggered cell FDCE clocked by clk_fpga_0 (rise@0.000ns fall@12.500ns period=25.000ns))
Destination: design_1_i/caravel_0/inst/mprij/fir_1/accumulated_result_reg[29]/D
(rising edge-triggered cell FDCE clocked by clk_fpga_0 (rise@0.000ns fall@12.500ns period=25.000ns))
Path Group: clk_fpga_0
Path Type: Setup (Max at Slow Process Corner)
Requirement: 25.000ns (clk_fpga_0 rise@25.000ns - clk_fpga_0 rise@0.000ns)
Data Path Delay: 13.638ns (logic 9.575ns (70.206%) route 4.063ns (29.794%))
Logic Levels: 15 (CARRY4=9 DSP48E1=2 LUT2=1 LUT3=1 LUT4=1 LUT5=1)
Clock Path Skew: -0.145ns (DCD - SCD + CPR)
Destination Clock Delay (DCD): 1.505ns (= 26.505 - 25.000)
Source Clock Delay (SCD): 1.700ns
Clock Pessimism Removal (CPR): 0.050ns
Clock Uncertainty: 0.377ns ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
Total System Jitter (TSJ): 0.071ns
Total Input Jitter (TIJ): 0.750ns
Discrete Jitter (DJ): 0.000ns
Phase Error (PE): 0.000ns

8. Implement Report :



5. checking no_input_delay (0)

There are 0 input ports with no input delay specified.

There are 0 input ports with no input delay but user has a false path constraint.

6. checking no_output_delay (0)

There are 0 ports with no output delay specified.

There are 0 ports with no output delay but user has a false path constraint

There are 0 ports with no output delay but with a timing clock defined on it or propagating through it

7. checking multiple_clock (0)

There are 0 register/latch pins with multiple clocks.

8. checking generated_clocks (0)

There are 0 generated clocks that are not connected to a clock source.

9. checking loops (0)

There are 0 combinational loops in the design.

10. checking partial_input_delay (0)

There are 0 input ports with partial input delay specified.

11. checking partial_output_delay (0)

There are 0 ports with partial output delay specified.

12. checking latch_loops (0)

There are 0 combinational latch loops in the design through latch input

Timing Details

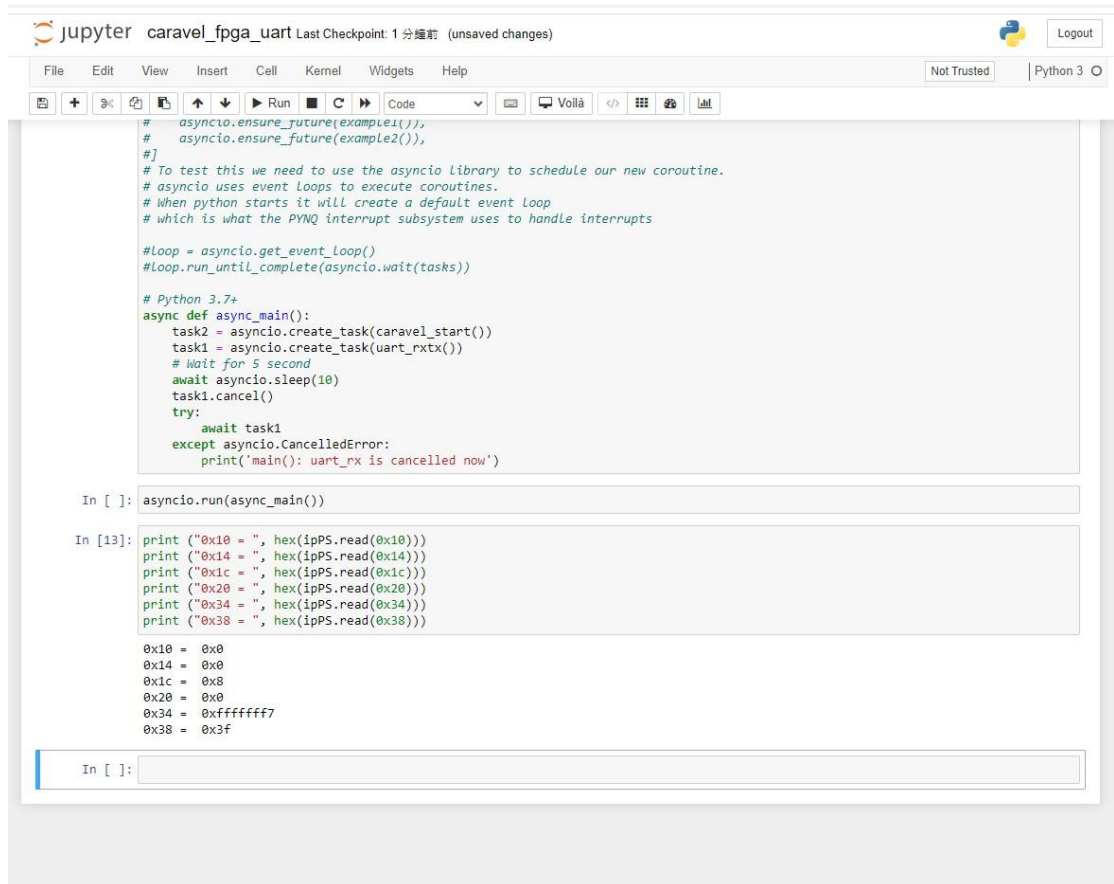
From Clock: clk_fpga_0
To Clock: clk_fpga_0

Setup :	0	Failing Endpoints,	Worst Slack	8.754ns,	Total Violation	0.000ns
Hold :	0	Failing Endpoints,	Worst Slack	0.010ns,	Total Violation	0.000ns
PW :	0	Failing Endpoints,	Worst Slack	11.250ns,	Total Violation	0.000ns

Max Delay Paths

Slack (MET) : 8.754ns (required time - arrival time)
Source: design_1_i/processing_system7_0/inst/PS7_i/FCLKCLK[0]
(clock source 'clk_fpga_0' (rise@0.000ns fall@12.500ns period=25.000ns))
Destination: design_1_i/caravel_ps_0/inst/control_s_axi_U/int_ps_mprj_out_reg[14]/D
(rising edge-triggered cell FDRE clocked by clk_fpga_0 (rise@0.000ns fall@12.500ns period=25.000ns))
Path Group: clk_fpga_0
Path Type: Setup (Max at Slow Process Corner)
Requirement: 12.500ns (clk_fpga_0 rise@25.000ns - clk_fpga_0 fall@12.500ns)
Data Path Delay: 5.919ns (logic 0.343ns (5.795%) route 5.576ns (94.205%))
Logic Levels: 3 (BUF@=1 LUT1=1 LUT6=1)
Clock Path Skew: 2.833ns (DCD - SCD + CPR)
Destination Clock Delay (DCD): 2.833ns = (27.833 - 25.000)
Source Clock Delay (SCD): 0.000ns = (12.500 - 12.500)
Clock Pessimism Removal (CPR): 0.000ns
Clock Uncertainty: 0.377ns ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
Total System Jitter (TSJ): 0.071ns
Total Input Jitter (TIJ): 0.750ns
Discrete Jitter (DJ): 0.000ns
Phase Error (PE): 0.000ns

9. On FPGA :



The image shows a Jupyter Notebook interface with the title "caravel_fpga_uart". The notebook contains a code cell with Python code using the `asyncio` library to test a UART interface on an FPGA. The code defines an `async_main` function that creates two tasks: `caravel_start` and `uart_rxtx`. It then runs these tasks concurrently using `asyncio.run`. Below the code cell, there are two input/output (In/Out) pairs. The first input is `asyncio.run(async_main())`, and the output shows the results of the UART reads at various memory addresses.

```
# asyncio.ensure_future(example1()),
# asyncio.ensure_future(example2()),
#]
# To test this we need to use the asyncio library to schedule our new coroutine.
# asyncio uses event loops to execute coroutines.
# When python starts it will create a default event loop
# which is what the PYNQ interrupt subsystem uses to handle interrupts

#loop = asyncio.get_event_loop()
#loop.run_until_complete(asyncio.wait(tasks))

# Python 3.7+
async def async_main():
    task2 = asyncio.create_task(caravel_start())
    task1 = asyncio.create_task(uart_rxtx())
    # Wait for 5 second
    await asyncio.sleep(10)
    task1.cancel()
    try:
        await task1
    except asyncio.CancelledError:
        print('main(): uart_rx is cancelled now')
```

In []: `asyncio.run(async_main())`

In [13]: `print ("0x10 = ", hex(ipPS.read(0x10)))`
`print ("0x14 = ", hex(ipPS.read(0x14)))`
`print ("0x1c = ", hex(ipPS.read(0x1c)))`
`print ("0x20 = ", hex(ipPS.read(0x20)))`
`print ("0x34 = ", hex(ipPS.read(0x34)))`
`print ("0x38 = ", hex(ipPS.read(0x38)))`

0x10 = 0x0
0x14 = 0x0
0x1c = 0x8
0x20 = 0x0
0x34 = 0xffffffff7
0x38 = 0x3f

In []: