SOC LAB# Test (final_project) Report

Group no: 4

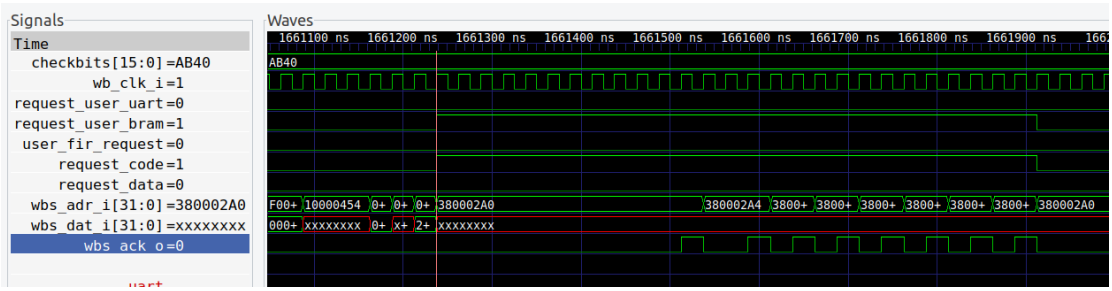
Members:

M11107410 羅善寬

M11107004 曹榮恩

M11107409 陳昱碩

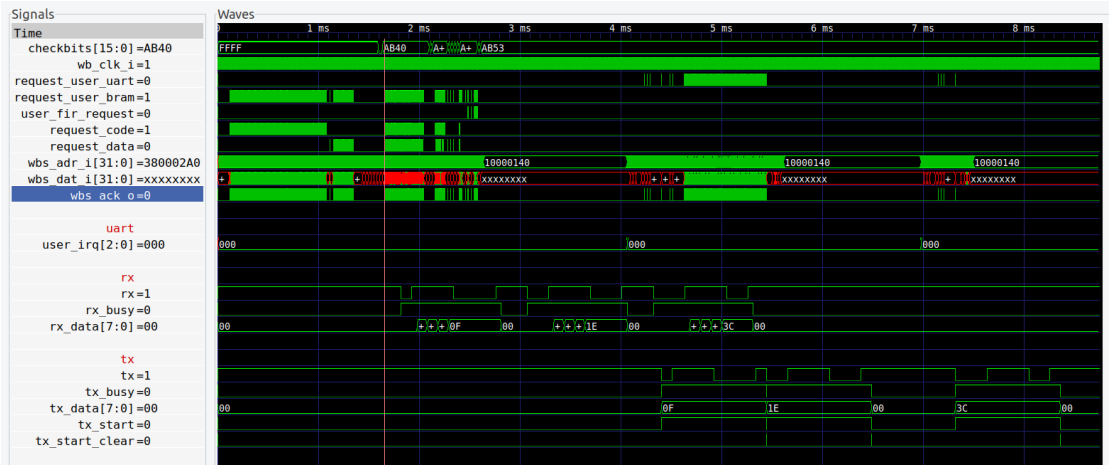
1. Bram



bram 輸入 addr 後 10T 取出資料。

使用 exmem_pipline 來控制，來達到 10T 可 prefetch 9 筆資料。之後即可直接輸出相對應資料。

2. UART



此波型 UART_rx 輸入三筆資料，在這裡 FIFO 設為兩筆資料後才 interrupt。第三筆資料經過一段時間後會自己產生 interrupt。

Interrupt 時 UART_tx 會回傳接收到的資料。

```
#define reg_rx_data          (*(volatile uint32_t*)0x30000000)
#define reg_tx_data          (*(volatile uint32_t*)0x30000004)
#define reg_uart_stat        (*(volatile uint32_t*)0x30000008)
#define reg_uart_clkdiv      (*(volatile uint32_t*)0x3000000c)
#define reg_rx_data_num      (*(volatile uint32_t*)0x30000010)
```

reg_rx_data_num	回傳中斷時 fifo 裡要回傳的 data 數量

3. Integration

把 mm、qsort、fir、UART 整合一起。

功能	使用 .c / .v 計算
mm (#element = 16)	.c
qsort (#data = 10)	.c
fir (#data = 64)	.v

```
#define reg_fir_control (*(volatile uint32_t *)0x36000000)
#define reg_fir_coef (*(volatile uint32_t *)0x36000040)

#define reg_fir_x (*(volatile uint32_t *)0x36000080)
#define reg_fir_y (*(volatile uint32_t *)0x36000084)

#define reg_tap (*(volatile uint32_t *)0x35000C00)
#define reg_data (*(volatile uint32_t *)0x35000C40)
```

reg_tap	tap 儲存在 0x3500C00~0x3500C3F
reg_data	data 儲存在 0x3500C40~0x3500FFF

```
ubuntu@ubuntu2004:~/Desktop/lab-wlos_baseline_1/testbench/integrate$ source run_clean
ubuntu@ubuntu2004:~/Desktop/lab-wlos_baseline_1/testbench/integrate$ source run_sim
Reading integrate.hex
integrate.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
VCD info: dumpfile integrate.vcd opened for output.
LA Test 1 started
tx data bit index 0: 1
tx data bit index 1: 1
mm
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x003e
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0044
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x004a
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0050
mm passed
tx data bit index 2: 1
tx data bit index 3: 1
qsort
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0028
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x037d
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x09ed
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0ca1
qsort passed
tx data bit index 4: 0
tx data bit index 5: 0
tx data bit index 6: 0
fir
fir passed
LA Test 1 passed
tx data bit index 7: 0
tx complete 1
tx data bit index 0: 0
```

```
fir passed
LA Test 1 passed
tx data bit index 7: 0
tx complete 1
tx data bit index 0: 0
tx data bit index 1: 1
tx data bit index 2: 1
tx data bit index 3: 1
tx data bit index 4: 1
tx data bit index 5: 0
tx data bit index 6: 0
tx data bit index 7: 0
tx complete 2
tx data bit index 0: 0
rx data bit index 0: 1
tx data bit index 1: 0
tx data bit index 2: 1
rx data bit index 1: 1
rx data bit index 2: 1
tx data bit index 3: 1
tx data bit index 4: 1
rx data bit index 3: 1
rx data bit index 4: 0
tx data bit index 5: 1
tx data bit index 6: 0
rx data bit index 5: 0
rx data bit index 6: 0
tx data bit index 7: 0
rx data bit index 7: 0
tx complete 3
recevied word  15
rx data bit index 0: 0
```

```
recevied word  15
rx data bit index 0: 0
rx data bit index 1: 1
rx data bit index 2: 1
rx data bit index 3: 1
rx data bit index 4: 1
rx data bit index 5: 0
rx data bit index 6: 0
rx data bit index 7: 0
recevied word  30
rx data bit index 0: 0
rx data bit index 1: 0
rx data bit index 2: 1
rx data bit index 3: 1
rx data bit index 4: 1
rx data bit index 5: 1
rx data bit index 6: 0
rx data bit index 7: 0
recevied word  60

Monitor: Timeout, Test LA (RTL) Failed

ubuntu@ubuntu2004:~/Desktop/lab-wlos_baseline_1/testbench/integrate$
```

3. Dataset preloaded into user memory area.

	wbs_adr_i[31:20]	wbs_adr_i[11:10]
request_data	0x350	10
request_fir_data	0x350	11
request_code	0x380	0x

```
wire request_code;
assign request_code = (wbs_stb_i & wbs_cyc_i &(wbs_adr_i[31:20] == 12'h380) & (!wbs_adr_i[11])) ? 1 : 0;

wire request_data;
assign request_data = (wbs_stb_i & wbs_cyc_i &(wbs_adr_i[31:20] == 12'h350) & (wbs_adr_i[11] & !wbs_adr_i[10])) ? 1 : 0;

wire request_fir_data;
assign request_fir_data = (wbs_stb_i & wbs_cyc_i &(wbs_adr_i[31:20] == 12'h350) & (wbs_adr_i[11] & wbs_adr_i[10])) ? 1 : 0;
```

```
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
    code_data : ORIGIN = 0x35000800, LENGTH = 0x00100000
    mprjram : ORIGIN = 0x38000000, LENGTH = 0x00400000
    hk : ORIGIN = 0x26000000, LENGTH = 0x00100000
    csr : ORIGIN = 0xf0000000, LENGTH = 0x00010000
}
```

```
48      .data :
49      {
50          . = ALIGN(8);
51          _fdata = .;
52          *(.data .data.* .gnu.linkonce.d.*)
53          *(.data1)
54          _gp = ALIGN(16);
55          *(.sdata .sdata.* .gnu.linkonce.s.*)
56          . = ALIGN(8);
57          _edata = .;
58      } > code_data AT > flash
59
60      .bss :
61      {
62          . = ALIGN(8);
63          _fbss = .;
64          *(.dynsbss)
65          *(.sbss .sbss.* .gnu.linkonce.sb.*)
66          *(.scommon)
67          *(.dynbss)
68          *(.bss .bss.* .gnu.linkonce.b.*)
69          *(COMMON)
70          . = ALIGN(8);
71          _ebss = .;
72          _end = .;
73      } > dff AT > flash
```

4. Verification on Jupyter notebook

mm、qs、fir 有通過，但是 Uart 部分沒有成功

In [1]:
```
from __future__ import print_function

import sys
import numpy as np
from time import time
import matplotlib.pyplot as plt

sys.path.append('/home/xilinx')
from pynq import Overlay
from pynq import allocate

from uartlite import *

import multiprocessing

# For sharing string variable
from multiprocessing import Process,Manager,Value
from ctypes import c_char_p

import asyncio

ROM_SIZE = 0x2000 #8K
```

In [2]:
```
ol = Overlay("/home/xilinx/jupyter_notebooks/caravel_fpga.bit")
#ol.ip_dict
```

In [3]:
```
ipOUTPIN = ol.output_pin_0
ipPS = ol.caravel_ps_0
ipReadROMCODE = ol.read_romcode_0
ipUart = ol.axi_uartlite_0
```

In [4]:
```
ol.interrupt_pins
```

Out[4]:
```
{'axi_intc_0/intr': {'controller': 'axi_intc_0',
  'index': 0,
  'fullpath': 'axi_intc_0/intr'},
 'axi_uartlite_0/interrupt': {'controller': 'axi_intc_0',
  'index': 0,
  'fullpath': 'axi_uartlite_0/interrupt'}}
```

In [5]:
```
# See what interrupts are in the system
#ol.interrupt_pins

# Each IP instances has a _interrupts dictionary which lists the names of the interrupts
#ipUart._interrupts

# The interrupts object can then be accessed by its name
# The Interrupt class provides a single function wait
# which is an asyncio coroutine that returns when the interrupt is signalled.
intUart = ipUart.interrupt
```

In [6]:
```
# Create np with 8K/4 (4 bytes per index) size and be initiled to 0
rom_size_final = 0

npROM = np.zeros(ROM_SIZE >> 2, dtype=np.uint32)
npROM_index = 0
npROM_offset = 0
fiROM = open("integrate.hex", "r+")
#fiROM = open("counter_wb.hex", "r+")

for line in fiROM:
    # offset header
    if line.startswith('@'):
        # Ignore first char @
        npROM_offset = int(line[1:].strip(b'\x00'.decode()), base = 16)
        npROM_offset = npROM_offset >> 2 # 4byte per offset
        #print (npROM_offset)
        npROM_index = 0
        continue
    #print (line)

    # We suppose the data must be 32bit alignment
    buffer = 0
    bytecount = 0
    for line_byte in line.strip(b'\x00'.decode()).split():
        buffer += int(line_byte, base = 16) << (8 * bytecount)
        bytecount += 1
        # Collect 4 bytes, write to npROM
        # Collect 4 bytes, write to npROM
        if(bytecount == 4):
            npROM[npROM_offset + npROM_index] = buffer
            # Clear buffer and bytecount
            buffer = 0
            bytecount = 0
            npROM_index += 1
            #print (npROM_index)
            continue
    # Fill rest data if not alignment 4 bytes
    if (bytecount != 0):
        npROM[npROM_offset + npROM_index] = buffer
        npROM_index += 1

fiROM.close()

rom_size_final = npROM_offset + npROM_index
#print (rom_size_final)

#for data in npROM:
#    print (hex(data))
```

In [7]:
```
# Allocate dram buffer will assign physical address to ip ipReadROMCODE

#rom_buffer = allocate(shape=(ROM_SIZE >> 2,), dtype=np.uint32)
rom_buffer = allocate(shape=(rom_size_final,), dtype=np.uint32)

# Initial it by npROM
#for index in range (ROM_SIZE >> 2):
for index in range (rom_size_final):
    rom_buffer[index] = npROM[index]

#for index in range (ROM_SIZE >> 2):
#    print ("0x{0:08x}".format(rom_buffer[index]))

# Program physical address for the romcode base address


# 0x00 : Control signals
#        bit 0  - ap_start (Read/Write/COH)
#        bit 1  - ap_done (Read/COR)
#        bit 2  - ap_idle (Read)
#        bit 3  - ap_ready (Read)
#        bit 7  - auto_restart (Read/Write)
#        others - reserved
# 0x10 : Data signal of romcode
#        bit 31~0 - romcode[31:0] (Read/Write)
```

```
# 0x14 : Data signal of romcode
#        bit 31~0 - romcode[63:32] (Read/Write)
# 0x1c : Data signal of length_r
#        bit 31~0 - length_r[31:0] (Read/Write)

ipReadROMCODE.write(0x10, rom_buffer.device_address)
ipReadROMCODE.write(0x1C, rom_size_final)

ipReadROMCODE.write(0x14, 0)

# ipReadROMCODE start to move the data from rom_buffer to bram
ipReadROMCODE.write(0x00, 1) # IP Start
while (ipReadROMCODE.read(0x00) & 0x04) == 0x00: # wait for done
    continue

print("Write to bram done")
```

```
Write to bram done
```

In [8]:
```
# Initialize AXI UART
uart = UartAXI(ipUart.mmio.base_addr)

# Setup AXI UART register
uart.setupCtrlReg()

# Get current UART status
uart.currentStatus()
```

Out[8]:
```
{'RX_VALID': 0,
 'RX_FULL': 0,
 'TX_EMPTY': 1,
 'TX_FULL': 0,
 'IS_INTR': 0,
 'OVERRUN_ERR': 0,
 'FRAME_ERR': 0,
 'PARITY_ERR': 0}
```

In [9]:
```
import time
async def uart_rxtx():
    # Reset FIFOs, enable interrupts
    ipUart.write(CTRL_REG, 1<<RST_TX | 1<<RST_RX | 1<<INTR_EN)
    print("Waitting for interrupt")
    tx_str = "hello\n"
    ipUart.write(TX_FIFO, ord(tx_str[0]))
    start = time.time()
    i = 1
    while(True):
        await intUart.wait()
        buf = ""
        # Read FIFO until valid bit is clear
        while ((ipUart.read(STAT_REG) & (1<<RX_VALID))):
            buf += chr(ipUart.read(RX_FIFO))
            end = time.time()
            #print("latency time:",(end - start))
            if i<len(tx_str):
                ipUart.write(TX_FIFO, ord(tx_str[i]))
                i=i+1
        print(buf, end='')

async def caravel_start():
    ipOUTPIN.write(0x10, 0)
    print("Start Caravel Soc")
    ipOUTPIN.write(0x10, 1)
```

```
29 async def check():
30     #x = hex(ipPS.read(0x1c)[:16])
31     while(((ipPS.read(0x1c)) & 0xffff0000) == 0xab510000):
32         continue
33     print("mm passed")
34     while(((ipPS.read(0x1c)) & 0xffff0000) == 0xab520000):
35         continue
36     print("qs passed")
37     while(((ipPS.read(0x1c)) & 0xffff0000) == 0xab530000):
38         continue
39     print("fir passed")
40
```

```python
# Python 3.7+
async def async_main():
    task2 = asyncio.create_task(caravel_start())
    task1 = asyncio.create_task(uart_rxtx())
    task0 = asyncio.create_task(check())
    # Wait for 5 second
    await asyncio.sleep(10)
    task1.cancel()
    try:
        await task1
    except asyncio.CancelledError:
        print('main(): uart_rx is cancelled now')
```

In [10]:
```python
asyncio.run(async_main())
```

```
Start Caravel Soc
Waitting for interrupt
mm passed
qs passed
fir passed
main(): uart_rx is cancelled now
```

In [11]:
```python
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 =  0xab530040
0x20 =  0x0
0x34 =  0x20
0x38 =  0x3f
```

5. Quality of Result

功能	# of clock
mm	(2044,537.5ns – 1661,237.5ns) / 25ns = 15332
qsort	(2252,687.5ns – 2158,237.5ns) / 25ns = 3778
fir	(2578,312.5ns – 2542,287.5ns) / 25ns = 1441

功能	Metrics = Latency – 500 * (1/baud_rate)
UART	Metrics = 680,340 us – 500 * (1/9600) (s /#bit) = 0.628s

因為 jupyter_notebook 沒有跑出來，所以用 simulation 波形估算的。

#data	512
fifo_length	32
Rx/tx transfer time per data	1250,000ns
Cpu intr to isr time	340,000ns
Latency	(transfer time per data * #data) + Cpu intr to isr time + (transfer time per data * fifo_length) = 680,340 us