Lab6 Workload optimized SOC – baseline

Group 6: 112061611 陳伯丞
112061524 葉又菘
110063553 張傑閔

1. How do you verify your answer from notebook

- Matrix Multiplication:

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} = \begin{bmatrix} 62 & 68 & 74 & 80 \\ 62 & 68 & 74 & 80 \\ 62 & 68 & 74 & 80 \\ 62 & 68 & 74 & 80 \end{bmatrix}$$

```
----------Test function matmul() Start----------------
Start Matmul Time: 9643988000
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
End Matmul Time: 10062238000
----------Test function matmul() Pass----------------
```

計算結果正確，答案以 16 進位顯示。

Calculation time in testbench: $10062238000 - 9643988000 = 0.418\text{ms}$

- Quick Sort

Golden pattern:

$$[40,\ 893,\ 2541,\ 2669,\ 3233,\ 4267,\ 4622,\ 5681,\ 6023,\ 9073]$$

共有 10 個答案，testbench 中我們只顯示後 4 個。

```
----------Test function qsort() Start----------------
Start QSort Time: 10062513000
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed,  4622
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed,  5681
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed,  6023
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed,  9073
End QSort Time: 10313763000
----------Test function qsort() Pass----------------
```

Calculation time in testbench: $10313763000 - 10062513000 = 0.251\text{ms}$

- FIR

```
----------Test function fir() Start----------------
Start FIR Time: 9364088000
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,   539
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,   732
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,   915
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,  1098
End FIR Time: 9643713000
----------Test function fir() Pass----------------
```

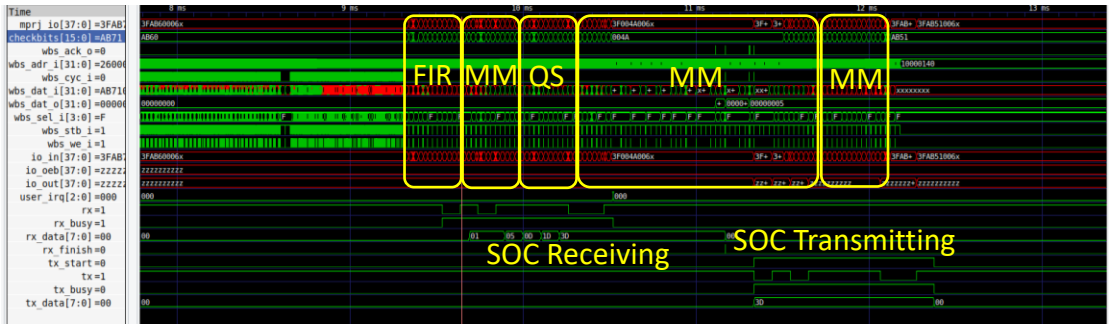
Calculation time in testbench: $9643713000 - 9364088000 = 0.280\text{ms}$

- Integrate the above tasks and UART

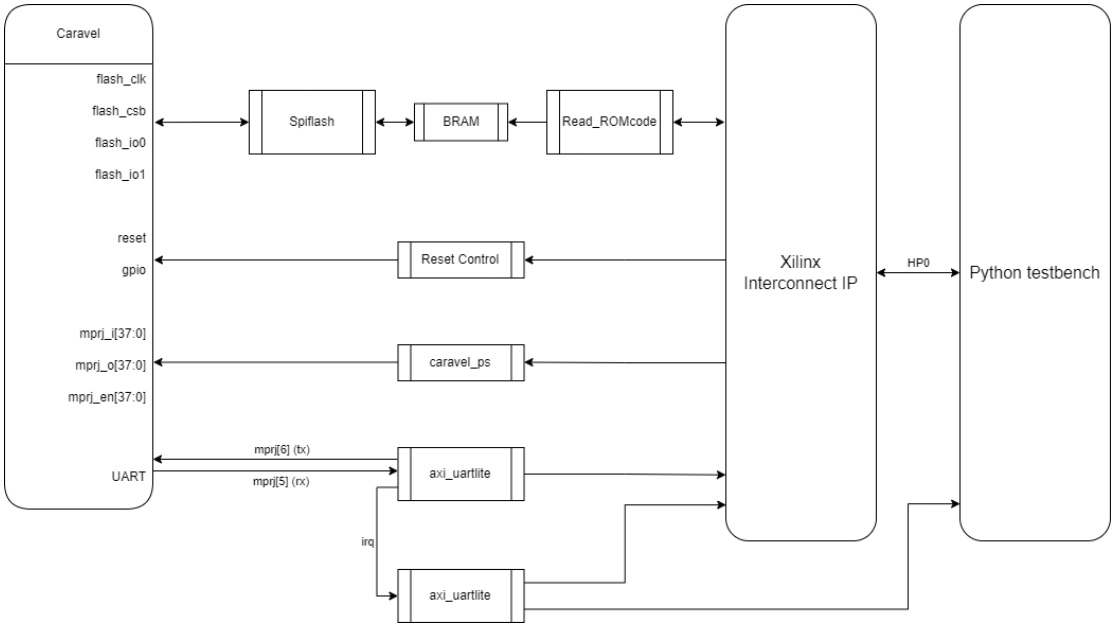
```
----------Test function fir() Start----------------
Start FIR Time: 9364088000
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,   539
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,   732
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,   915
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,  1098
tx data bit index 0: 1
End FIR Time: 9643713000
----------Test function fir() Pass-----------------

----------Test function matmul() Start----------------
Start Matmul Time: 9643988000
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
tx data bit index 1: 0
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0050
tx data bit index 2: 1
tx data bit index 3: 1
tx data bit index 4: 1
End Matmul Time: 10062238000
----------Test function matmul() Pass----------------

----------Test function qsort() Start----------------
Start QSort Time: 10062513000
tx data bit index 5: 1
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed,  4622
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed,  5681
tx data bit index 6: 0
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed,  6023
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed,  9073
End QSort Time: 10313763000
----------Test function qsort() Pass----------------

----------Test function matmul() 2nd Start----------------
Start Matmul 2nd Time: 10340713000
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x003e
tx data bit index 7: 0
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
tx complete 2
rx data bit index 0: 1
rx data bit index 1: 0
End Matmul 2nd Time: 11704113000
----------Test function matmul() 2nd Pass----------------
rx data bit index 2: 1
```

UART interrupts tasks

```
----------Test function matmul() 3rd Start----------------
Start Matmul 3rd Time: 11730938000
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
rx data bit index 3: 1
rx data bit index 4: 1
rx data bit index 5: 1
End Matmul 3rd Time: 12122913000
----------Test function matmul() 3rd Pass----------------
rx data bit index 6: 0
rx data bit index 7: 0
recevied word  61
```



2. Block design



3. Timing report/ resource report after synthesis

1. Slice Logic

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs	5336	0	0	53200	10.03
LUT as Logic	5148	0	0	53200	9.68
LUT as Memory	188	0	0	17400	1.08
LUT as Distributed RAM	18	0			
LUT as Shift Register	170	0			
Slice Registers	6175	0	0	106400	5.80
Register as Flip Flop	6175	0	0	106400	5.80
Register as Latch	0	0	0	106400	0.00
F7 Muxes	170	0	0	26600	0.64
F8 Muxes	47	0	0	13300	0.35

3. Memory

Site Type	Used	Fixed	Prohibited	Available	Util%
Block RAM Tile	7	0	0	140	5.00
RAMB36/FIFO*	4	0	0	140	2.86
RAMB36E1 only	4				
RAMB18	6	0	0	280	2.14
RAMB18E1 only	6				

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             8.982ns  (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:        5.700ns  (logic 0.374ns (6.562%)  route 5.326ns (93.438%))
  Logic Levels:           3  (BUFG=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

Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             17.122ns  (required time - arrival time)
  Source:                 design_1_i/output_pin_0/inst/control_s_axi_U/int_outpin_ctrl_reg[0]/C
                            (rising edge-triggered cell FDRE clocked by clk_fpga_0  {rise@0.000ns fall@12.500ns period=25.000ns})
  Destination:            design_1_i/caravel_0/inst/housekeeping/serial_data_staging_1_reg[0]/CLR
                            (recovery check against rising-edge clock clk_fpga_0  {rise@0.000ns fall@12.500ns period=25.000ns})
  Path Group:             **async_default**
  Path Type:              Recovery (Max at Slow Process Corner)
  Requirement:            25.000ns  (clk_fpga_0 rise@25.000ns - clk_fpga_0 rise@0.000ns)
  Data Path Delay:        7.105ns  (logic 0.642ns (9.036%)  route 6.463ns (90.964%))
  Logic Levels:           1  (LUT1=1)
  Clock Path Skew:        0.009ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.818ns = ( 27.818 - 25.000 )
    Source Clock Delay      (SCD):    2.938ns
    Clock Pessimism Removal (CPR):    0.129ns
  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
```

4. Latency for a character loop back using UART

```
    start = time.time()
    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))
            if i<len(tx_str):
                ipUart.write(TX_FIFO, ord(tx_str[i]))
                i=i+1
        print(buf, end='')
        if i == len(tx_str):
            end = time.time()
            print("\nTime:", end - start, "seconds")
            break
```

In [10]:

```
asyncio.run(async_main())
```

```
Start Caravel Soc
Waitting for interrupt
hello
Time: 0.024158954620361328 seconds
```

5. Suggestion for improving latency or UART loop back

```
----------Test function fir() Start----------------
Start FIR Time: 9364088000
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,   539
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,   732
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,   915
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed,  1098
tx data bit index 0: 1
End FIR Time: 9643713000
----------Test function fir() Pass----------------

----------Test function matmul() Start----------------
Start Matmul Time: 9643988000
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
tx data bit index 1: 0
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0050
tx data bit index 2: 1
tx data bit index 3: 1
tx data bit index 4: 1
End Matmul Time: 10062238000
----------Test function matmul() Pass----------------

----------Test function qsort() Start----------------
Start QSort Time: 10062513000
tx data bit index 5: 1
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed,  4622
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed,  5681
tx data bit index 6: 0
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed,  6023
Call function qsort() in User Project BRAM (mprjram, 0x38000000) return value passed,  9073
End QSort Time: 10313763000
----------Test function qsort() Pass----------------

----------Test function matmul() 2nd Start----------------
Start Matmul 2nd Time: 10340713000
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x003e
tx data bit index 7: 0
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
tx complete 2
```

Testbench 的結果中我們可以明顯看到每次 interrupt 之間間隔相當久，所以可以使用 FIFO 增加一次傳輸的資料數量來達到增加 throughput 的效果。

6. FPGA Result (UART)

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 time
import asyncio

ROM_SIZE = 0x2000 #8K
```

In [2]:
```
ol = Overlay("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("uart.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
        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)
```

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]:

```
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]))
    i = 1
    start = time.time()
    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))
            if i<len(tx_str):
                ipUart.write(TX_FIFO, ord(tx_str[i]))
                i=i+1
        print(buf, end='')
        if i == len(tx_str):
            end = time.time()
            print("\nTime:", end - start, "seconds")
            break

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

# Python 3.5+
#tasks = [ # Create a task list
#    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 [10]:
```
asyncio.run(async_main())
```

```
Start Caravel Soc
Waitting for interrupt
hello
Time: 0.024158954620361328 seconds
```

In [11]:
```
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 =  0xab510040
0x20 =  0x0
0x34 =  0x20
0x38 =  0x3f
```

7. GitHub link

https://github.com/yousungyeh/course-lab_6