112061603 李柏叡

112061620 郭邑哲

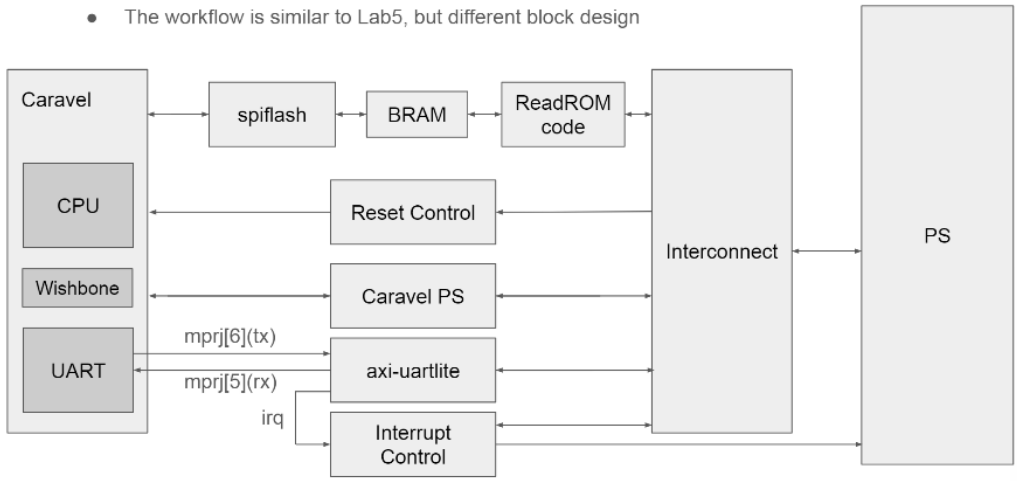
112061621 貢暐家

SOC Lab6

Group 7

Block Design:

PS 寫一筆資料，透過 Interconnect 送到 AXI-uartlite，然後 Caravel 會去接收這筆資料；如果 Caravel 送去資料的話， Uartlite 會去 trigger interrupt，interrupt 會送到 axi_interrupt_controller 再到 irq，讓 PS 知道 Uartlite 已經有資料讓它去讀。



- **Simulation on Matrix Multiplication, Quick Sort, FIR and UART**

對應 checkbits:

Mm:

```
Reading counter_la_mm.hex
counter_la_mm.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
VCD info: dumpfile counter_la_mm.vcd opened for output.
LA Test 1 started
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value p
assed, 0x003e
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value p
assed, 0x0044
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value p
assed, 0x004a
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value p
assed, 0x0050
LA Test 2 passed
```

波形圖:

Qs:

```
Reading counter_la_qs.hex
counter_la_qs.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
VCD info: dumpfile counter_la_qs.vcd opened for output.
LA Test 1 started
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value p
assed, 0x0028
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value p
assed, 0x037d
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value p
assed, 0x09ed
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value p
assed, 0x0a6d
LA Test 2 passed
```

波形圖:

Fir:

```
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
```

波形圖:

Uart:

```
Reading uart.hex
uart.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
VCD info: dumpfile uart.vcd opened for output.
LA Test 1 started
tx data bit index 0: 1
tx data bit index 1: 0
tx data bit index 2: 1
tx data bit index 3: 1
tx data bit index 4: 1
tx data bit index 5: 1
tx data bit index 6: 0
tx data bit index 7: 0
tx complete 2
rx data bit index 0: 1
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
```

波形圖:

Intergrated simulation result (with uart random delay = 81000cycle):

```
Reading all.hex
all.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
VCD info: dumpfile all.vcd opened for output.
mm started
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
qs started
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, 0x0a6d
UART started with random delay       81000 cycles
UART interrupt at              2024988
qs passed
fir started
tx data bit index 0: 1
tx data bit index 1: 0
tx data bit index 2: 1
tx data bit index 3: 1
tx data bit index 4: 1
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0000
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, 0xfff6
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, 0xffe3
tx data bit index 5: 1
Call function fir() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0393
fir passed
tx data bit index 6: 0
tx data bit index 7: 0
tx complete 2
rx data bit index 0: 1
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  61
uart passed
Whole system end at              4427055
```

波形圖:

由圖可知 uart 可與 mm、qsort、fir 平行執行

- **How do you verify your answer from notebook ?**

由於 notebook (PS 端)對於輸出 mprj_out 的掃描較慢，導致無法檢查值，因此

我們在 firmware 中加入 loop 延遲，方便我們在 notebook 檢查運算是否正確

在想要觀察的運算值後，加入以下迴圈，便可在 notebook 中觀測到

```
int volatile j;
j=0;
while(j<100)
    j++;
```

需要注意的是，由於我們為了符合本次實驗架構的 BRAM size: 8K Byte，因此採用-Os 優化策略減少.hex size，但這樣同時會使我們設計的 loop 被優化掉，因此要特別使用 volatile，防止優化以達到目的。

未使用 loop 的觀察結果：

```
Start Caravel Soc
Waitting for interrupt
0x8
0x48
0xab500040
0xab500000
0xab510000
0xab700040
hello
```

使用 loop 的觀察結果:

檢查順序依次為

1. mm[9]:68
2. mm passed(mprj_out=0xab510041)
3. qsort[9]:9073
4. qsort passed(mprj_out=0xab610041)
5. fir[10]:1098
6. qsort passed(mprj_out=0xab710041)

```
Start Caravel Soc
Waitting for interrupt

68 checked
0x440040
0x440040
0x440040
0x440040
0x440040
0x440040
0x440040
0x440040
mm passed
0xab510040
0xab510040
0xab510040
0xab510040
0xab510040
0xab510040
0xab510040
0xab510040
0xab510040
0xab510040
0xab510040
0xab510040
0xab510040
0xab510040
0xab510040
9073 checked
0x23710040
0x23710040
0x23710040
0x23710040
0x23710040
0x23710040
0x23710040
0x23710040
0x23710040
qsort passed
0x23710040
1098 checked
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
0x44a0040
fir passed
hello
```

Uart 完成傳輸後的 mprj:

```
0x10 =  0x0
0x14 =  0x0
0x1c =  0xab800040
0x20 =  0x0
0x34 =  0x20
0x38 =  0x3f
```

Python code:

```python
async def caravel_show():
    s = ""
    while(ipPS.read(0x1c) != 0x00440040):
        s = hex(ipPS.read(0x1c))
        print(s)
    print(s)
    print("68 checked")
    while(ipPS.read(0x1c) != 0xab510040):
        s = hex(ipPS.read(0x1c))
        print(s)
    print(s)
    print("mm passed")
    while(ipPS.read(0x1c) != 0x23710040):
        s = hex(ipPS.read(0x1c))
        print(s)
    print(s)
    print("9073 checked")
    while(ipPS.read(0x1c) != 0xab610040):
        s = hex(ipPS.read(0x1c))
        print(s)
    print(s)
    print("qsort passed")
    while(ipPS.read(0x1c) != 0x044a0040):
        s = hex(ipPS.read(0x1c))
        print(s)
    print(s)
    print("1098 checked")
    while(ipPS.read(0x1c) != 0xab710040):
        s = hex(ipPS.read(0x1c))
        print(s)
    print(s)
    print("fir passed")
```

- **Timing report/ resource report after synthesis ?**

Timing report

```
From Clock:  clk_fpga_0
  To Clock:  clk_fpga_0

Setup :            0  Failing Endpoints,  Worst Slack        9.268ns,  Total Violation        0.000ns
Hold  :            0  Failing Endpoints,  Worst Slack        0.020ns,  Total Violation        0.000ns
PW    :            0  Failing Endpoints,  Worst Slack       11.250ns,  Total Violation        0.000ns
```

Max delay path

Slack: 9.268ns

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             9.268ns  (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:        5.254ns  (logic 0.344ns (6.547%)  route 4.910ns (93.453%))
  Logic Levels:           3  (BUFG=1 LUT1=1 LUT6=1)
  Clock Path Skew:        2.658ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.658ns = ( 27.658 - 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
```

Utilization report

```
1. Slice Logic
--------------

+----------------------------+------+-------+------------+-----------+-------+
|          Site Type         | Used | Fixed | Prohibited | Available | Util% |
+----------------------------+------+-------+------------+-----------+-------+
| Slice LUTs                 | 5329 |     0 |          0 |     53200 | 10.02 |
|   LUT as Logic             | 5141 |     0 |          0 |     53200 |  9.66 |
|   LUT as Memory            |  188 |     0 |          0 |     17400 |  1.08 |
|     LUT as Distributed RAM |   18 |     0 |            |           |       |
|     LUT as Shift Register  |  170 |     0 |            |           |       |
| Slice Registers            | 6159 |     0 |          0 |    106400 |  5.79 |
|   Register as Flip Flop    | 6159 |     0 |          0 |    106400 |  5.79 |
|   Register as Latch        |    0 |     0 |          0 |    106400 |  0.00 |
| F7 Muxes                   |  169 |     0 |          0 |     26600 |  0.64 |
| F8 Muxes                   |   47 |     0 |          0 |     13300 |  0.35 |
+----------------------------+------+-------+------------+-----------+-------+
```

- **Latency for a character loop back using UART ?**

Notebook result:

```
Loop back Latency: 0.004352490107218425 per character
main(): uart_rx is cancelled now
```

Python code:

```
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
    global delay
    time1 = 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))
            time2 = time()
            delay += time2 - time1
            if i<len(tx_str):
                ipUart.write(TX_FIFO, ord(tx_str[i]))
                i=i+1
                time1 = time()
        print(buf, end='')
```

```
print('Loop back Latency: {} per character'.format(delay/6.0))
print('main(): uart_rx is cancelled now')
```

- **Suggestion for improving latency for UART loop back ?**

1. Directly loop back in hardware

直接將 rx 所接收到的資料經由 tx 傳送出去，而不是去中斷 cpu，然後 cpu 將接

收到的資料讀回去後送出來，這樣可以大幅降低 uart loop back 的延遲。

2. Using higher baud rate

使用更高的 baud rate 能降低資料傳送所需的時間，進而改善 loop back 的 latency

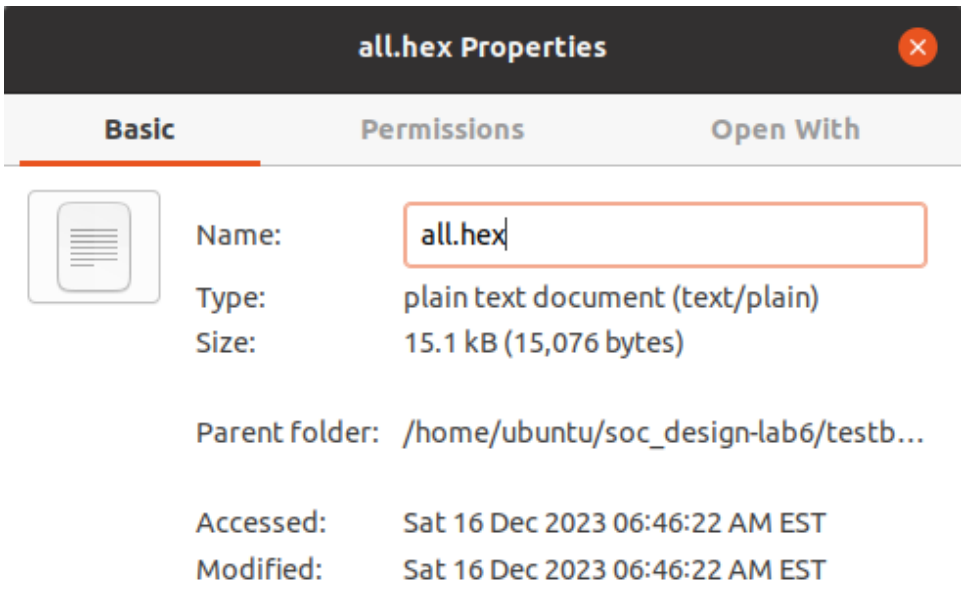
- **What else do you observe ?**

1.可使用指令使.hex 檔在不同策略下優化

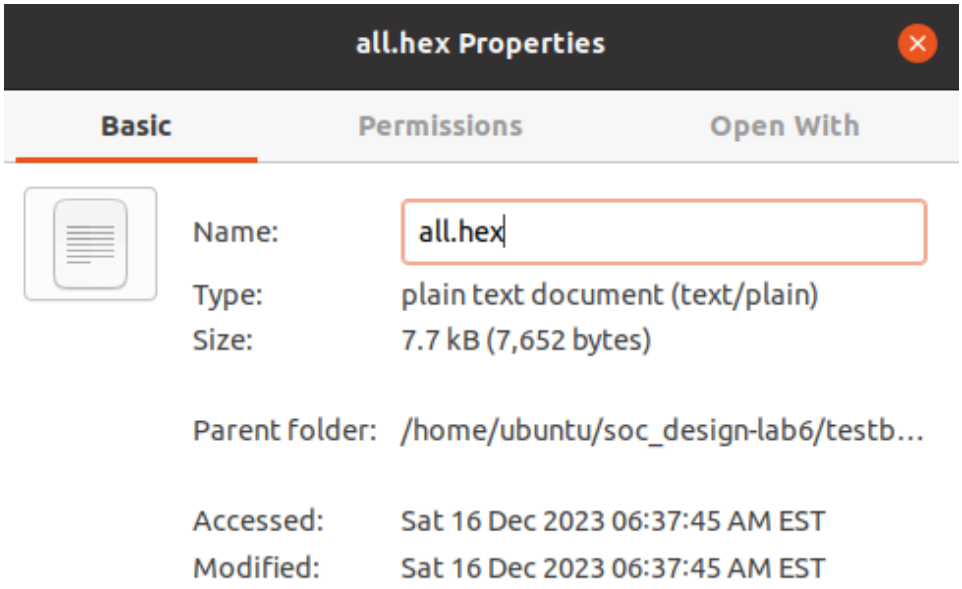
可在 run_sim 檔中修改以下指令使 compiler 在進行編譯時先進行優化，有效降低指令數量及提高運算速度

```
riscv32-unknown-elf-gcc -Os -Wl,--no-warn-rwx-segments -g \
```

未優化前檔案大小:



優化後檔案大小:



未優化前所需時間:

```
Whole system end at                6663813
```

優化後所需時間:

```
Whole system end at                4427055
```

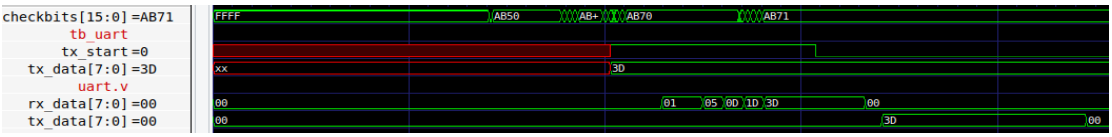
使用 Os 策略在指令檔大小方面減少了約 49%，而在速度方面則加快了整個系

統約 33.6%

2.可在 tcl 合成檔中加入如下指令，有效解決 hold time violation

```
update_compile_order -fileset sources_1
set_property STEPS.PHYS_OPT_DESIGN.ARGS.DIRECTIVE ExploreWithHoldFix [get_runs impl_1]
launch_runs impl_1 -to_step write_bitstream -job 2
wait_on_run impl_1
```

3. 執行各項 workload 所需的時間差異



從波形圖中可以看出執行 fir 和 mm 所需要的時間比較長，我們認為可以用硬體來加速此部分的計算，而且 fir 及 mm 都會用到乘加運算，因此我們打算在 final project 中實作一個 module 搭配 DMA，用來加速 fir 與 mm。

- **Github link:**

https://github.com/ruei7916/soc_design-lab6