LAB6

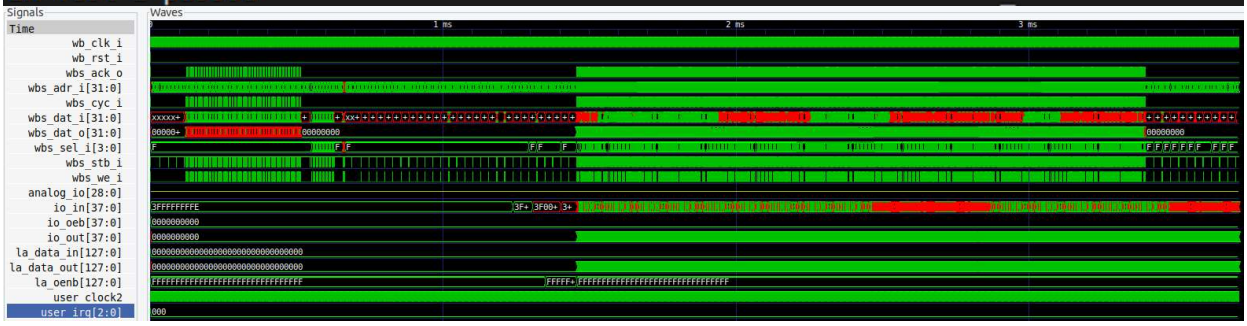
How do you verify your answer from notebook

Simulate first, uart/integrate validate on the FPGA board

1.simulation

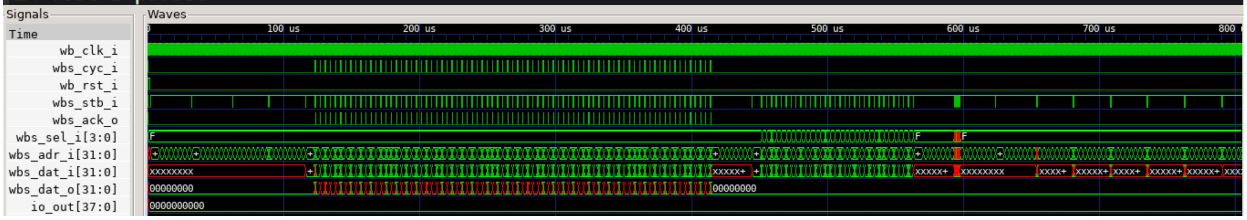
counter_la_fir:

```
ubuntu@ubuntu2004:~/lab-wlos_baseline/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
```



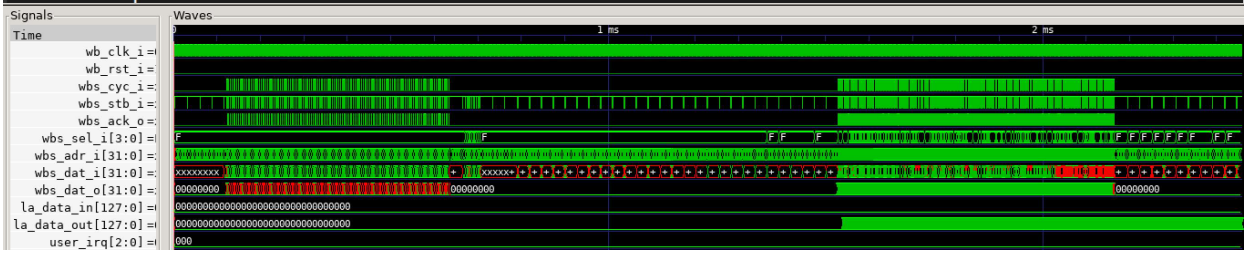
counter_la_mm:

```
ubuntu@ubuntu2004:~$ cd ~/caravel-soc_fpga-lab/lab-wlos_baseline/testbench/counter_la_mm
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab/lab-wlos_baseline/testbench/counter_la_mm$ source run_clean
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab/lab-wlos_baseline/testbench/counter_la_mm$ source run_sim
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 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
LA Test 2 passed
```



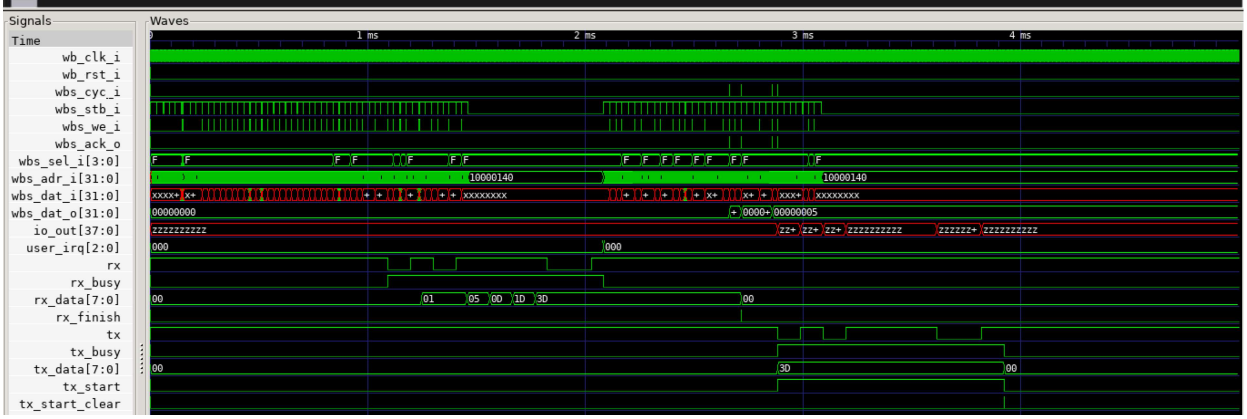
qsort:

```
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab/lab-wlos_baseline/testbench/counter_la_qs$ source run_clean
ubuntu@ubuntu2004:~/caravel-soc_fpga-lab/lab-wlos_baseline/testbench/counter_la_qs$ source run_sim
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 passed, 0x0028
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x037d
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x09ed
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0a6d
LA Test 2 passed
```



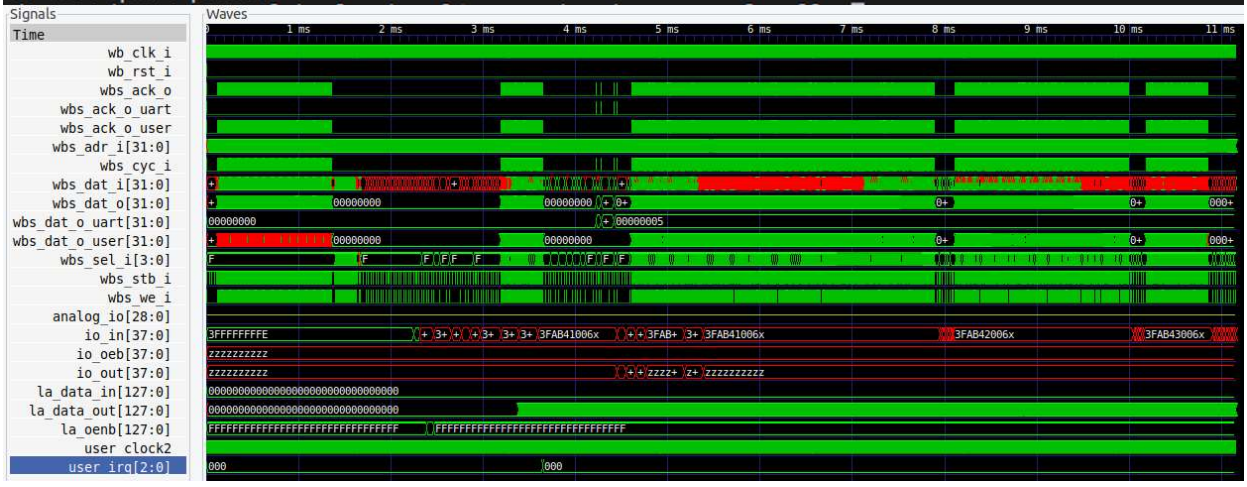
uart:

```
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
recevied word  61
```



integrate

```
ubuntu@ubuntu2004:~/lab-wlos_baseline/testbench/counter_la_all$ source run_sim
Reading counter_la_all.hex
counter_la_all.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
VCD info: dumpfile counter_la_all.vcd opened for output.
LA Test uart 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
LA Test fir started
tx data bit index 5: 1
tx data bit index 6: 0
tx data bit index 7: 0
tx complete 1
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
LA Test fir passed
LA Test matmul 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
LA Test matmul passed
LA Test qsort 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
LA Test qsort passed
```



2.FPGA

uart:

```
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
ol = Overlay("./design_1_wrapper.bit")
#ol.ip_dict
ipOUTPIN = ol.output_pin_0
ipPS = ol.caravel_ps_0
ipReadROMCODE = ol.read_romcode_0
ipUart = ol.axi_uartlite_0
ol.interrupt_pins

```

```
{'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'}}
```

```
# See what interrupts are in the system
#ol.interrupt_pins

# Each IP instances has a _interrupts dictionary which lists the names of the interru
#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
# 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)

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

```
# 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
```

```
# Initialize AXI UART
uart = UartAXI(ipUart.mmio.base_addr)

# Setup AXI UART register
uart.setupCtrlReg()

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

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

```
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
    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 buf == '\n':
            return

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

```
Start Caravel Soc
Waitting for interrupt
hello
```

```
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
```

integrate:

```
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
ol = Overlay("./design_1.bit")
#ol.ip_dict
ipOUTPIN = ol.output_pin_0
ipPS = ol.caravel_ps_0
ipReadROMCODE = ol.read_romcode_0
ipUart = ol.axi_uartlite_0
ol.interrupt_pins
```

```
{'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'}}
```

```
# See what interrupts are in the system
#ol.interrupt_pins

# Each IP instances has a _interrupts dictionary which lists the names of the interru
#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
# 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("counter_la_all21.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)

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

```
# 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
```

```
# Initialize AXI UART
uart = UartAXI(ipUart.mmio.base_addr)

# Setup AXI UART register
uart.setupCtrlReg()

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

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

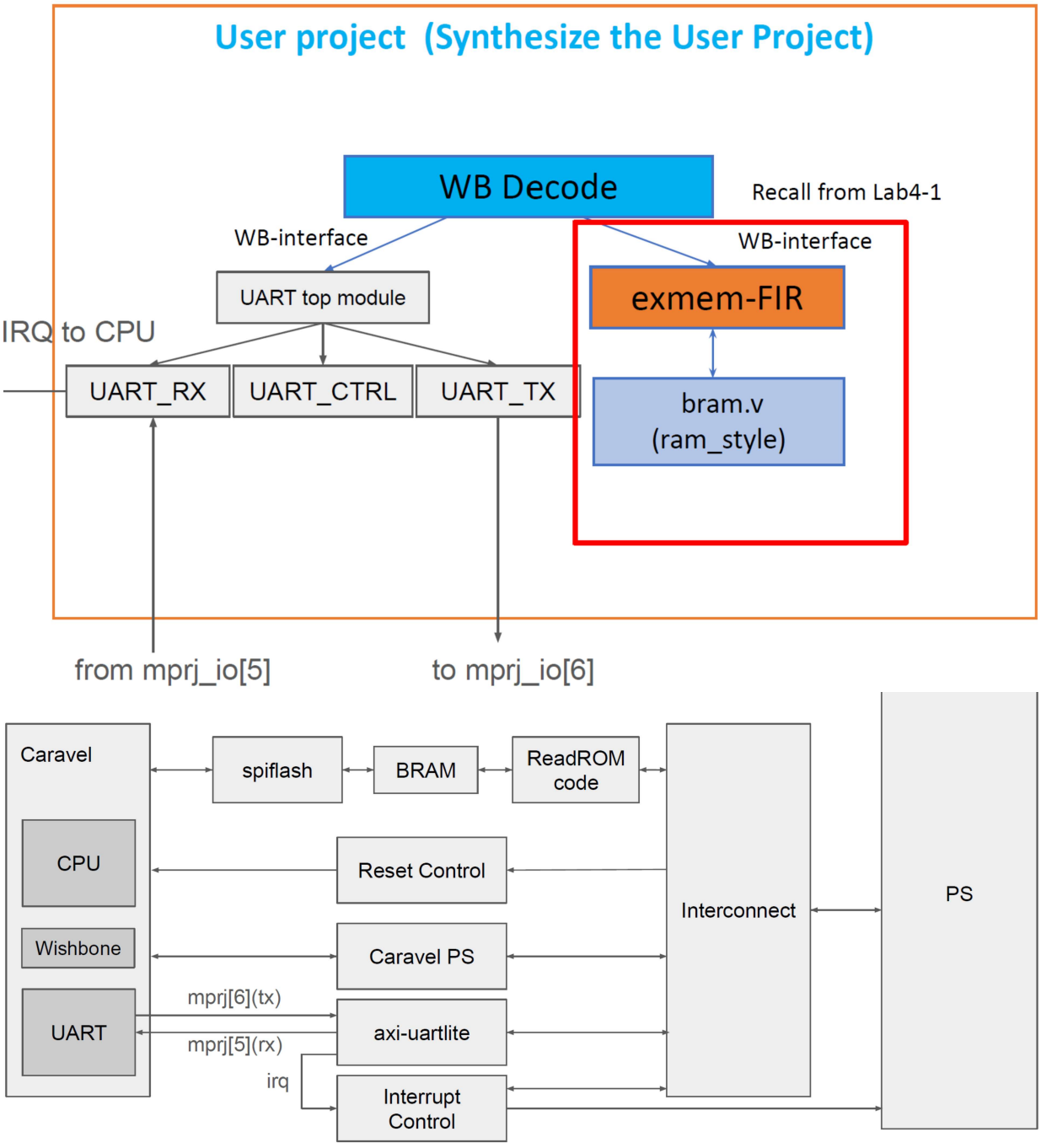
```
async def uart_rxtx():
    # Reset FIFOs, enable interrupts
    ipUart.write(CTRL_REG, 1<<RST_TX | 1<<RST_RX | 1<<INTR_EN)
    print("Waitting for interrupt")
    tx_str = "hello\n"
    ipUart.write(TX_FIFO, ord(tx_str[0]))
    i = 1
    while(True):
        await intUart.wait()
        buf = ""
        # 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 buf == '\n':
            return

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

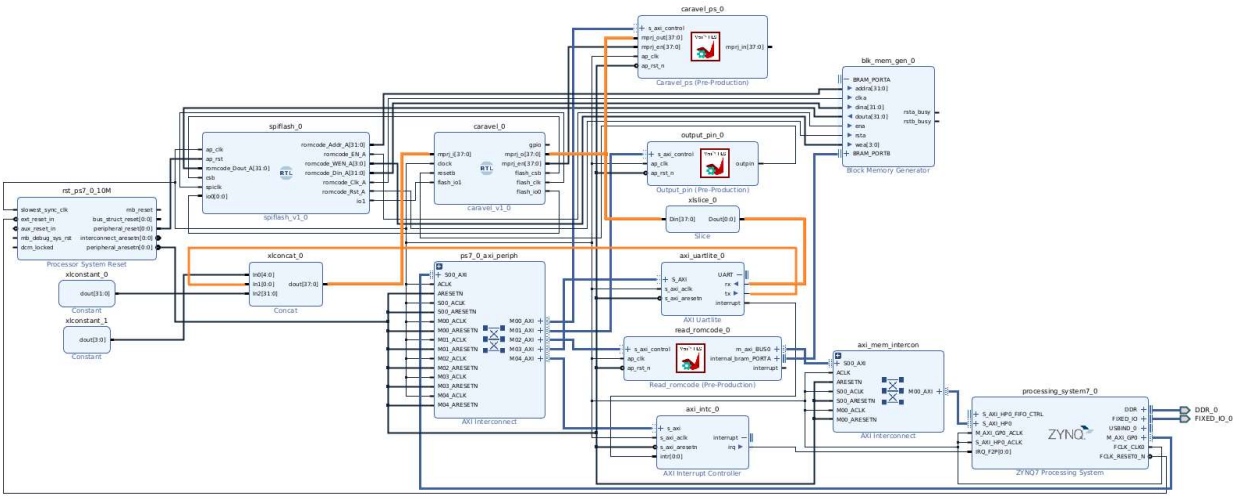
```
Start Caravel Soc
Waitting for interrupt
hello
```

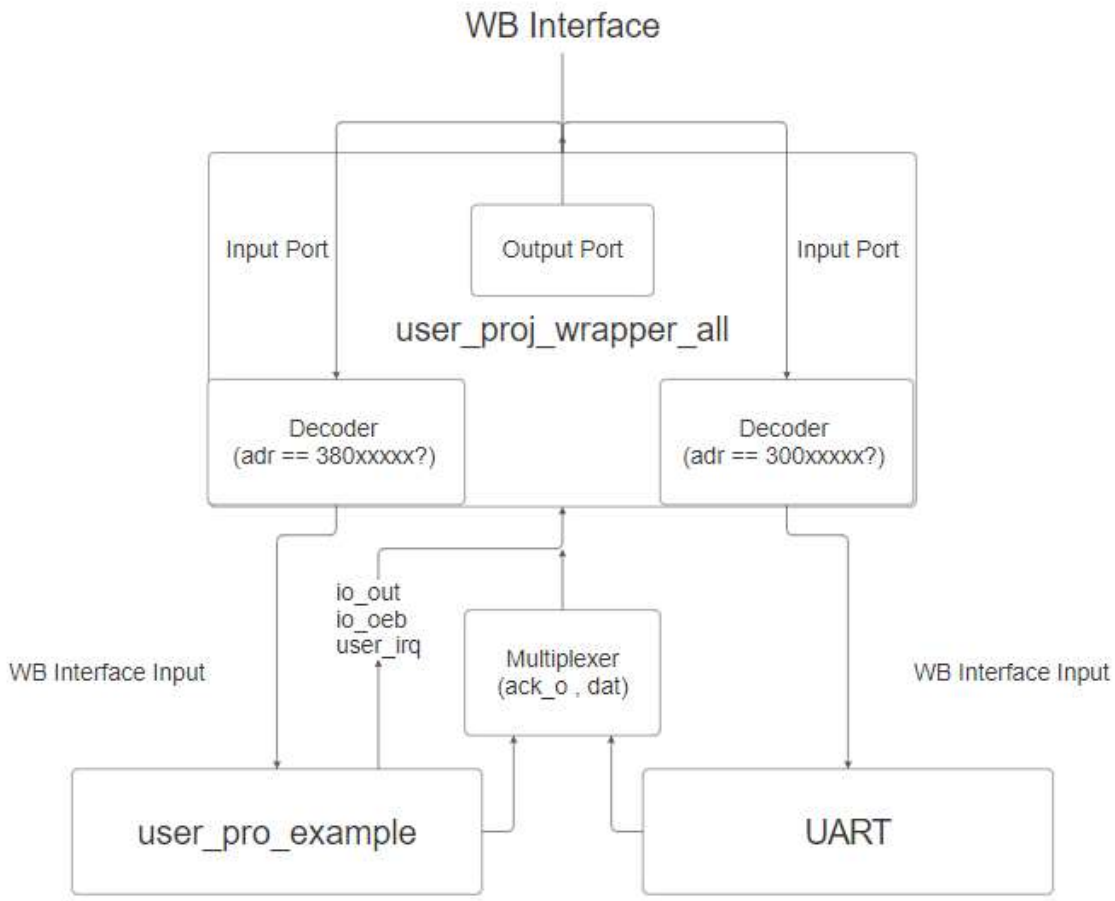
```
print ("0x10 = ", hex(ipPS.read(0x10)))
print ("0x14 = ", hex(ipPS.read(0x14)))
print ("0x1c = ", hex(ipPS.read(0x1c)))
print ("0x20 = ", hex(ipPS.read(0x20)))
print ("0x34 = ", hex(ipPS.read(0x34)))
print ("0x38 = ", hex(ipPS.read(0x38)))
```

```
0x10 =  0x0
0x14 =  0x0
0x1c =  0xab530040
0x20 =  0x0
0x34 =  0x20
0x38 =  0x3f
```

Block design







Timing report/ resource report after synthesis

uart:

Worst Negative Slack (WNS)

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             9.353ns  (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=
  Destination:            design_1_i/caravel_ps_0/inst/control_s_axi_U/int_ps_mprj_out_reg[
                            (rising edge-triggered cell FDRE clocked by clk_fpga_0  {rise@0
  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:        5.165ns  (logic 0.345ns (6.680%)  route 4.820ns (93.320%))
  Logic Levels:           3  (BUFG=1 LUT1=1 LUT6=1)
  Clock Path Skew:        2.655ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.655ns = ( 27.655 - 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

    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s
  -------------------------------------------------------------------    -------------------
                         (clock clk_fpga_0 fall edge)
                                                   12.500    12.500 f
    PS7_X0Y0             PS7                        0.000    12.500 f  design_1_i/process
                         net (fo=1, routed)         1.193    13.693    design_1_i/process
    BUFGCTRL_X0Y20                                                 f  design_1_i/process
    BUFGCTRL_X0Y20       BUFG (Prop_bufg_I_O)       0.101    13.794 f  design_1_i/process
                         net (fo=5262, routed)      2.330    16.124    design_1_i/caravel
    SLICE_X44Y90                                                   f  design_1_i/caravel
    SLICE_X44Y90         LUT6 (Prop_lut6_I3_O)      0.124    16.248 f  design_1_i/caravel
                         net (fo=1, routed)         0.610    16.857    design_1_i/caravel
    SLICE_X35Y90                                                   f  design_1_i/caravel
    SLICE_X35Y90         LUT1 (Prop_lut1_I0_O)      0.120    16.977 f  design_1_i/caravel
                         net (fo=1, routed)         0.687    17.665    design_1_i/caravel
    SLICE_X44Y90         FDRE                                      f  design_1_i/caravel
  -------------------------------------------------------------------    -------------------

                         (clock clk_fpga_0 rise edge)
                                                   25.000    25.000 r
    PS7_X0Y0             PS7                        0.000    25.000 r  design_1_i/process
                         net (fo=1, routed)         1.088    26.088    design_1_i/process
    BUFGCTRL_X0Y20                                                 r  design_1_i/process
    BUFGCTRL_X0Y20       BUFG (Prop_bufg_I_O)       0.091    26.179 r  design_1_i/process
                         net (fo=5262, routed)      1.476    27.655    design_1_i/caravel
    SLICE_X44Y90         FDRE                                      r  design_1_i/caravel
                         clock pessimism            0.000    27.655
                         clock uncertainty         -0.377    27.278
    SLICE_X44Y90         FDRE (Setup_fdre_C_D)     -0.261    27.017    design_1_i/caravel
  -------------------------------------------------------------------
                         required time                       27.017
                         arrival time                       -17.665
  -------------------------------------------------------------------
                         slack                               9.353
```

Worst Hold Slack (WHS)

```
Min Delay Paths
--------------------------------------------------------------------------------
Slack (MET) :             0.038ns  (arrival time - required time)
  Source:                 design_1_i/caravel_0/inst/soc/core/mgmtsoc_litespisdrphycore_stor
                            (rising edge-triggered cell FDRE clocked by clk_fpga_0  {rise@0
  Destination:            design_1_i/caravel_0/inst/soc/core/interface4_bank_bus_dat_r_reg[
                            (rising edge-triggered cell FDRE clocked by clk_fpga_0  {rise@0
  Path Group:             clk_fpga_0
  Path Type:              Hold (Min at Fast Process Corner)
  Requirement:            0.000ns  (clk_fpga_0 rise@0.000ns - clk_fpga_0 rise@0.000ns)
  Data Path Delay:        0.356ns  (logic 0.164ns (46.046%)  route 0.192ns (53.954%))
  Logic Levels:           0
  Clock Path Skew:        0.259ns (DCD - SCD - CPR)
    Destination Clock Delay (DCD):    1.192ns
    Source Clock Delay      (SCD):    0.898ns
    Clock Pessimism Removal (CPR):    0.035ns

    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s
  -------------------------------------------------------------------    -------------------
                         (clock clk_fpga_0 rise edge)
                                                      0.000     0.000 r
    PS7_X0Y0             PS7                          0.000     0.000 r  design_1_i/process
                         net (fo=1, routed)           0.310     0.310    design_1_i/process
    BUFGCTRL_X0Y20                                                  r  design_1_i/process
    BUFGCTRL_X0Y20       BUFG (Prop_bufg_I_O)         0.026     0.336 r  design_1_i/process
                         net (fo=5262, routed)        0.562     0.898    design_1_i/caravel
    SLICE_X46Y46         FDRE                                       r  design_1_i/caravel
  -------------------------------------------------------------------    -------------------
    SLICE_X46Y46         FDRE (Prop_fdre_C_Q)         0.164     1.062 r  design_1_i/caravel
                         net (fo=5, routed)           0.192     1.254    design_1_i/caravel
    SLICE_X50Y47         FDRE                                       r  design_1_i/caravel
  -------------------------------------------------------------------    -------------------

                         (clock clk_fpga_0 rise edge)
                                                      0.000     0.000 r
    PS7_X0Y0             PS7                          0.000     0.000 r  design_1_i/process
                         net (fo=1, routed)           0.337     0.337    design_1_i/process
    BUFGCTRL_X0Y20                                                  r  design_1_i/process
    BUFGCTRL_X0Y20       BUFG (Prop_bufg_I_O)         0.029     0.366 r  design_1_i/process
                         net (fo=5262, routed)        0.826     1.192    design_1_i/caravel
    SLICE_X50Y47         FDRE                                       r  design_1_i/caravel
                         clock pessimism             -0.035     1.157
    SLICE_X50Y47         FDRE (Hold_fdre_C_D)         0.059     1.216    design_1_i/caravel
  -------------------------------------------------------------------
                         required time                         -1.216
                         arrival time                           1.254
  -------------------------------------------------------------------
                         slack                                  0.038
```

integrate:

Worst Negative Slack (WNS)

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             9.353ns  (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=
  Destination:            design_1_i/caravel_ps_0/inst/control_s_axi_U/int_ps_mprj_out_reg[
                            (rising edge-triggered cell FDRE clocked by clk_fpga_0  {rise@0
  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:        5.165ns  (logic 0.345ns (6.680%)  route 4.820ns (93.320%))
  Logic Levels:           3  (BUFG=1 LUT1=1 LUT6=1)
  Clock Path Skew:        2.655ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.655ns = ( 27.655 - 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

    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s
  -------------------------------------------------------------------    -------------------
                         (clock clk_fpga_0 fall edge)
                                                   12.500    12.500 f
    PS7_X0Y0             PS7                        0.000    12.500 f  design_1_i/process
                         net (fo=1, routed)         1.193    13.693    design_1_i/process
    BUFGCTRL_X0Y20                                                 f  design_1_i/process
    BUFGCTRL_X0Y20       BUFG (Prop_bufg_I_O)       0.101    13.794 f  design_1_i/process
                         net (fo=5262, routed)      2.330    16.124    design_1_i/caravel
    SLICE_X44Y90                                                   f  design_1_i/caravel
    SLICE_X44Y90         LUT6 (Prop_lut6_I3_O)      0.124    16.248 f  design_1_i/caravel
                         net (fo=1, routed)         0.610    16.857    design_1_i/caravel
    SLICE_X35Y90                                                   f  design_1_i/caravel
    SLICE_X35Y90         LUT1 (Prop_lut1_I0_O)      0.120    16.977 f  design_1_i/caravel
                         net (fo=1, routed)         0.687    17.665    design_1_i/caravel
    SLICE_X44Y90         FDRE                                      f  design_1_i/caravel
  -------------------------------------------------------------------    -------------------

                         (clock clk_fpga_0 rise edge)
                                                   25.000    25.000 r
    PS7_X0Y0             PS7                        0.000    25.000 r  design_1_i/process
                         net (fo=1, routed)         1.088    26.088    design_1_i/process
    BUFGCTRL_X0Y20                                                 r  design_1_i/process
    BUFGCTRL_X0Y20       BUFG (Prop_bufg_I_O)       0.091    26.179 r  design_1_i/process
                         net (fo=5262, routed)      1.476    27.655    design_1_i/caravel
    SLICE_X44Y90         FDRE                                      r  design_1_i/caravel
                         clock pessimism            0.000    27.655
                         clock uncertainty         -0.377    27.278
    SLICE_X44Y90         FDRE (Setup_fdre_C_D)     -0.261    27.017    design_1_i/caravel
  -------------------------------------------------------------------
                         required time                       27.017
                         arrival time                       -17.665
  -------------------------------------------------------------------
                         slack                               9.353
```

Worst Hold Slack (WHS)

```
Min Delay Paths
--------------------------------------------------------------------------------
Slack (MET) :             0.038ns  (arrival time - required time)
  Source:                 design_1_i/caravel_0/inst/soc/core/mgmtsoc_litespisdrphycore_stor
                            (rising edge-triggered cell FDRE clocked by clk_fpga_0  {rise@0
  Destination:            design_1_i/caravel_0/inst/soc/core/interface4_bank_bus_dat_r_reg[
                            (rising edge-triggered cell FDRE clocked by clk_fpga_0  {rise@0
  Path Group:             clk_fpga_0
  Path Type:              Hold (Min at Fast Process Corner)
  Requirement:            0.000ns  (clk_fpga_0 rise@0.000ns - clk_fpga_0 rise@0.000ns)
  Data Path Delay:        0.356ns  (logic 0.164ns (46.046%)  route 0.192ns (53.954%))
  Logic Levels:           0
  Clock Path Skew:        0.259ns (DCD - SCD - CPR)
    Destination Clock Delay (DCD):    1.192ns
    Source Clock Delay      (SCD):    0.898ns
    Clock Pessimism Removal (CPR):    0.035ns

    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s
  -------------------------------------------------------------------    -------------------
                         (clock clk_fpga_0 rise edge)
                                                      0.000     0.000 r
    PS7_X0Y0             PS7                          0.000     0.000 r  design_1_i/process
                         net (fo=1, routed)           0.310     0.310    design_1_i/process
    BUFGCTRL_X0Y20                                                  r  design_1_i/process
    BUFGCTRL_X0Y20       BUFG (Prop_bufg_I_O)         0.026     0.336 r  design_1_i/process
                         net (fo=5262, routed)        0.562     0.898    design_1_i/caravel
    SLICE_X46Y46         FDRE                                       r  design_1_i/caravel
  -------------------------------------------------------------------    -------------------
    SLICE_X46Y46         FDRE (Prop_fdre_C_Q)         0.164     1.062 r  design_1_i/caravel
                         net (fo=5, routed)           0.192     1.254    design_1_i/caravel
    SLICE_X50Y47         FDRE                                       r  design_1_i/caravel
  -------------------------------------------------------------------    -------------------

                         (clock clk_fpga_0 rise edge)
                                                      0.000     0.000 r
    PS7_X0Y0             PS7                          0.000     0.000 r  design_1_i/process
                         net (fo=1, routed)           0.337     0.337    design_1_i/process
    BUFGCTRL_X0Y20                                                  r  design_1_i/process
    BUFGCTRL_X0Y20       BUFG (Prop_bufg_I_O)         0.029     0.366 r  design_1_i/process
                         net (fo=5262, routed)        0.826     1.192    design_1_i/caravel
    SLICE_X50Y47         FDRE                                       r  design_1_i/caravel
                         clock pessimism             -0.035     1.157
    SLICE_X50Y47         FDRE (Hold_fdre_C_D)         0.059     1.216    design_1_i/caravel
  -------------------------------------------------------------------
                         required time                         -1.216
                         arrival time                           1.254
  -------------------------------------------------------------------
                         slack                                  0.038
```

integrate utilization

Utilization Design Information

Table of Contents
----
1. Slice Logic
1.1 Summary of Registers by Type
2. Slice Logic Distribution
3. Memory
4. DSP
5. IO and GT Specific
6. Clocking
7. Specific Feature
8. Primitives
9. Black Boxes
10. Instantiated Netlists

1. Slice Logic

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs	5373	0	0	53200	10.10
LUT as Logic	5185	0	0	53200	9.75
LUT as Memory	188	0	0	17400	1.08
LUT as Distributed RAM	18	0			
LUT as Shift Register	170	0			
Slice Registers	6173	0	0	106400	5.80
Register as Flip Flop	6173	0	0	106400	5.80
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

1.1 Summary of Registers by Type

Total	Clock Enable	Synchronous	Asynchronous
0	_	-	-
0	_	-	Set
0	_	-	Reset
0	_	Set	-
0	_	Reset	-
0	Yes	-	-
283	Yes	-	Set
1031	Yes	-	Reset
130	Yes	Set	-
4729	Yes	Reset	-

2. Slice Logic Distribution

Site Type	Used	Fixed	Prohibited	Available	Util
Slice	2373	0	0	13300	17.8

SLICEL	1674	0		
SLICEM	699	0		
LUT as Logic	5185	0	0	53200	9.7
using O5 output only	0			
using O6 output only	4166			
using O5 and O6	1019			
LUT as Memory	188	0	0	17400	1.0
LUT as Distributed RAM	18	0		
using O5 output only	0			
using O6 output only	2			
using O5 and O6	16			
LUT as Shift Register	170	0		
using O5 output only	43			
using O6 output only	81			
using O5 and O6	46			
Slice Registers	6173	0	0	106400	5.8
Register driven from within the Slice	3058			
Register driven from outside the Slice	3115			
LUT in front of the register is unused	2073			
LUT in front of the register is used	1042			
Unique Control Sets	320		0	13300	2.4

* * Note: Available Control Sets calculated as Slice * 1, Review the Control Sets Report fo

3. Memory

Site Type	Used	Fixed	Prohibited	Available	Util%
Block RAM Tile	10	0	0	140	7.14
RAMB36/FIFO*	7	0	0	140	5.00
RAMB36E1 only	7				
RAMB18	6	0	0	280	2.14
RAMB18E1 only	6				

* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate

4. DSP

Site Type	Used	Fixed	Prohibited	Available	Util%
DSPs	0	0	0	220	0.00

5. IO and GT Specific

Site Type	Used	Fixed	Prohibited	Available	Util%
Bonded IOB	0	0	0	125	0.00
Bonded IPADs	0	0	0	2	0.00
Bonded IOPADs	130	130	0	130	100.00
PHY_CONTROL	0	0	0	4	0.00
PHASER_REF	0	0	0	4	0.00
OUT_FIFO	0	0	0	16	0.00

IN_FIFO	0	0	0	16	0.00
IDELAYCTRL	0	0	0	4	0.00
IBUFDS	0	0	0	121	0.00
PHASER_OUT/PHASER_OUT_PHY	0	0	0	16	0.00
PHASER_IN/PHASER_IN_PHY	0	0	0	16	0.00
IDELAYE2/IDELAYE2_FINEDELAY	0	0	0	200	0.00
ILOGIC	0	0	0	125	0.00
OLOGIC	0	0	0	125	0.00

6. Clocking

Site Type	Used	Fixed	Prohibited	Available	Util%
BUFGCTRL	5	0	0	32	15.63
BUFIO	0	0	0	16	0.00
MMCME2_ADV	0	0	0	4	0.00
PLLE2_ADV	0	0	0	4	0.00
BUFMRCE	0	0	0	8	0.00
BUFHCE	0	0	0	72	0.00
BUFR	0	0	0	16	0.00

7. Specific Feature

Site Type	Used	Fixed	Prohibited	Available	Util%
BSCANE2	0	0	0	4	0.00
CAPTUREE2	0	0	0	1	0.00
DNA_PORT	0	0	0	1	0.00
EFUSE_USR	0	0	0	1	0.00
FRAME_ECCE2	0	0	0	1	0.00
ICAPE2	0	0	0	2	0.00
STARTUPE2	0	0	0	1	0.00
XADC	0	0	0	1	0.00

8. Primitives

Ref Name	Used	Functional Category
FDRE	4729	Flop & Latch
LUT6	2120	LUT
LUT5	1178	LUT
LUT4	1042	LUT
FDCE	1031	Flop & Latch
LUT3	982	LUT
LUT2	666	LUT
FDPE	283	Flop & Latch
CARRY4	236	CarryLogic
LUT1	216	LUT
MUXF7	169	MuxFx
SRL16E	152	Distributed Memory

FDSE	130	Flop & Latch
BIBUF	130	IO
SRLC32E	64	Distributed Memory
MUXF8	47	MuxFx
RAMD32	26	Distributed Memory
RAMS32	8	Distributed Memory
RAMB36E1	7	Block Memory
RAMB18E1	6	Block Memory
BUFG	5	Clock
PS7	1	Specialized Resource

9. Black Boxes

Ref Name	Used

10. Instantiated Netlists

Ref Name	Used
design_1_xbar_0	1
design_1_spiflash_0_0	1
design_1_rst_ps7_0_10M_0	1
design_1_read_romcode_0_0	1
design_1_processing_system7_0_0	1
design_1_output_pin_0_0	1
design_1_caravel_ps_0_0	1
design_1_caravel_0_0	1
design_1_blk_mem_gen_0_0	1
design_1_axi_uartlite_0_0	1
design_1_axi_intc_0_0	1
design_1_auto_us_0	1
design_1_auto_pc_1	1
design_1_auto_pc_0	1

Latency for a character loop back using UART

```
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(" Latancy Time:",(end-start))
            if i<len(tx_str):
                ipUart.write(TX_FIFO, ord(tx_str[i]))
                i=i+1
        print(buf, end='')
        if buf == '\n':
            return

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

uart latency:

```
Start Caravel Soc
Waitting for interrupt
 Latancy Time: 0.004650592803955078
h Latancy Time: 0.010066032409667969
e Latancy Time: 0.015562057495117188
l Latancy Time: 0.022139549255371094
l Latancy Time: 0.027289628982543945
o Latancy Time: 0.033171653747558594
```

integrate latency:

```
Start Caravel Soc
Waitting for interrupt
 Latancy Time: 0.009405851364135742
h Latancy Time: 0.03203272819519043
e Latancy Time: 0.050551652908325195
l Latancy Time: 0.07190442085266113
l Latancy Time: 0.07689094543457031
o Latancy Time: 0.08246421813964844
```

Suggestion for improving latency for UART loop back

1.FIFO buffer：在UART收發線路上分別使用FIFO buffer，在讀取或寫入1 byte data時，UART可以同時接收或傳送另一byte data。

2.如果支援DMA：data傳輸過程中Direct Memory Access，減少路徑傳輸時間，降低latency

3.使用硬體加速器處理UART通信

What else do you observe

在合成時原先worst hold slack只有0.032ns，由於需要將線路放到FPGA板子上，實際情況與原先設置constrain可能並不相符，導致在灌入.hex時產生錯誤，使我們認為是記憶體不夠大或是bram問題。

中途嘗試過將bram或是在python端的rom增大但都無濟於事，也有嘗試使用vivado內部implementation的改變作調整，若是將phys_opt_design中的directive設置為AggressiveHoldFix能再將hold slack提高到0.034ns，但實際效用不大，從path上可以發現是與bram相關，因此我們在wrapper裡module連接的地方增加更多條件判斷(如：address decode使用更多位元等)以嘗試增加hold slack，使軟體能夠重新選擇cell以達到更多slack。

最後在module引入的輸入端加上enable(也就是address decode)將worst hold slack提高至0.038ns，並通過測試，因此我們認為有可能是在原先放置bram時，某些bram的timing較緊，導致在輸入hex時因實際電路產生negative hold slack，因此才會有討論區中用加法計算fir就不會有問題的情況，也就是用加法的話hex較短就不會用到timing較緊的那個bram。

但若認為是hold time slack造成上述問題，矛盾的點在於時序可以說是幾乎沒改變只多了0.006ns，對於電路的改善理應不大，因此目前只能推斷是放置時因為加入enable等其他邏輯，造成flash或是bram的相對位置改變，造成電路在時序上的時間改變(應該更遠才會使slack更多，而且還只有一點點)，因此實際的原因仍需更多實驗驗證。

(討論區連結 https://github.com/bol-edu/HLS-SOC-Discussions/discussions/168 (https://github.com/bol-edu/HLS-SOC-Discussions/discussions/168)) 。