SOC-Design Lab1

EECS24 109020014 李承澔

This report discusses the execution of an HLS multiplication kernel on the PYNQ-Z2 board.

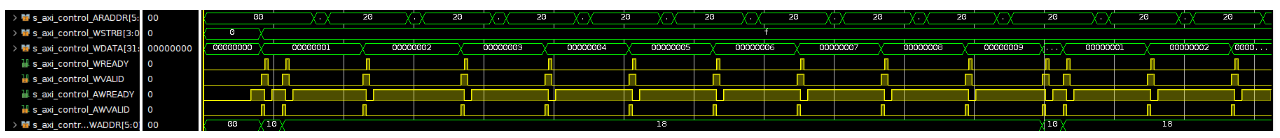
Brief introduction

In this lab, we delve into the FPGA High-Level Synthesis (HLS) flow.
Our objective is to design a multiplier with `2 inputs` and `1 output` , subsequently exporting it to RTL IP. This will then be deployed on the PYNQ-Z2 MPSoC PL side. The PYNQ Arm core will run a Python program that invokes the multiplier IP to compute the result.
We will then assess both its performance and utilization.

Observed & Learned

```
from __future__ import print_function

import sys, os

sys.path.append('/home/xilinx')
os.environ['XILINX_XRT'] = '/usr'
from pynq import Overlay

if __name__ == "__main__":
    print("Entry:", sys.argv[0])
    print("System argument(s):", len(sys.argv))

    print("Start of \"" + sys.argv[0] + "\"")

    ol = Overlay("/home/xilinx/jupyter_notebooks/Multip2Num.bit")
    regIP = ol.multip_2num_0

    for i in range(9):
        print("============================")
        for j in range(9):
            regIP.write(0x10, i + 1)
            regIP.write(0x18, j + 1)
            Res = regIP.read(0x20)
            print(str(i + 1) + " * " + str(j + 1) + " = " + str(Res))
    print("============================")
    print("Exit process")
```

In the `Multip2Num.ipynb` file, the Python program employs an API to invoke the kernel function, specifying addresses for input placement and result retrieval.
Precisely, the host writes two int values at addresses `0x10` and `0x18` , and fetches the result from `0x20` .
This behavior can be further confirmed by examining the waveform:

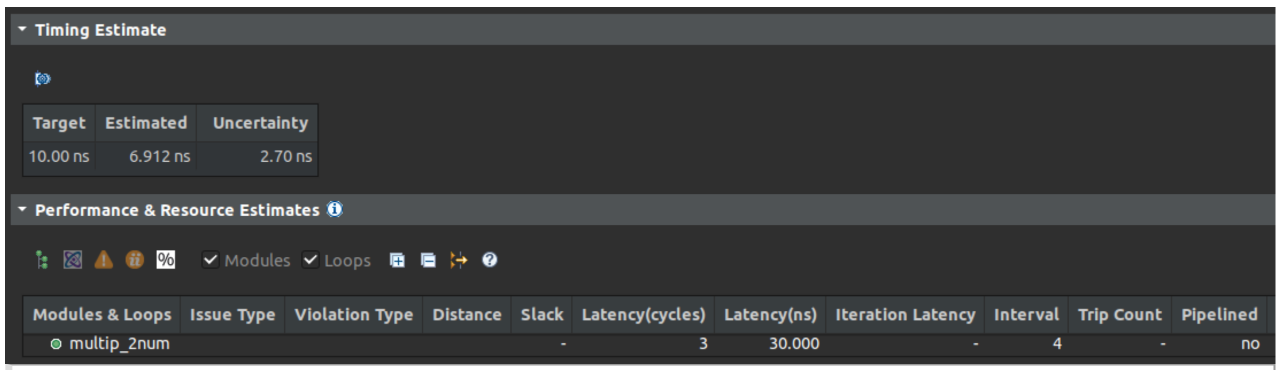


Here, the `s_axi_controlAWADDR` value alternates between `10` and `18`, while `s_axi_controlARADDR` is set at `20`.

From this lab, I’ve gained an insightful introduction to the FPGA HLS kernel design flow and learned to interpret the associated performance and utilization reports. Furthermore, I’ve deepened my understanding of the data transfer protocols at both block and port levels.

Screen Dump

Performance



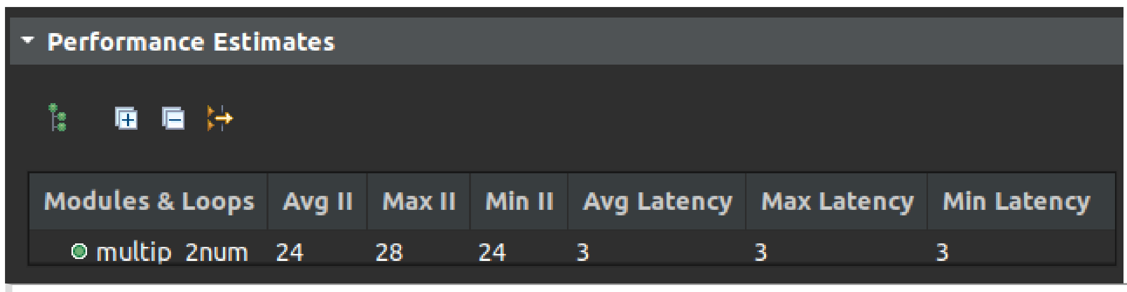
Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	6.912 ns	2.70 ns

Performance & Resource Estimates

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined
multip_2num				-	3	30.000	-	4	-	no

Synthesis Report

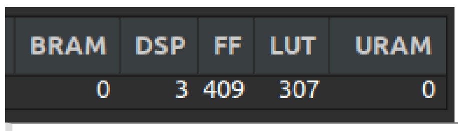


Performance Estimates

Modules & Loops	Avg II	Max II	Min II	Avg Latency	Max Latency	Min Latency
multip_2num	24	28	24	3	3	3

Co-Simulation Report

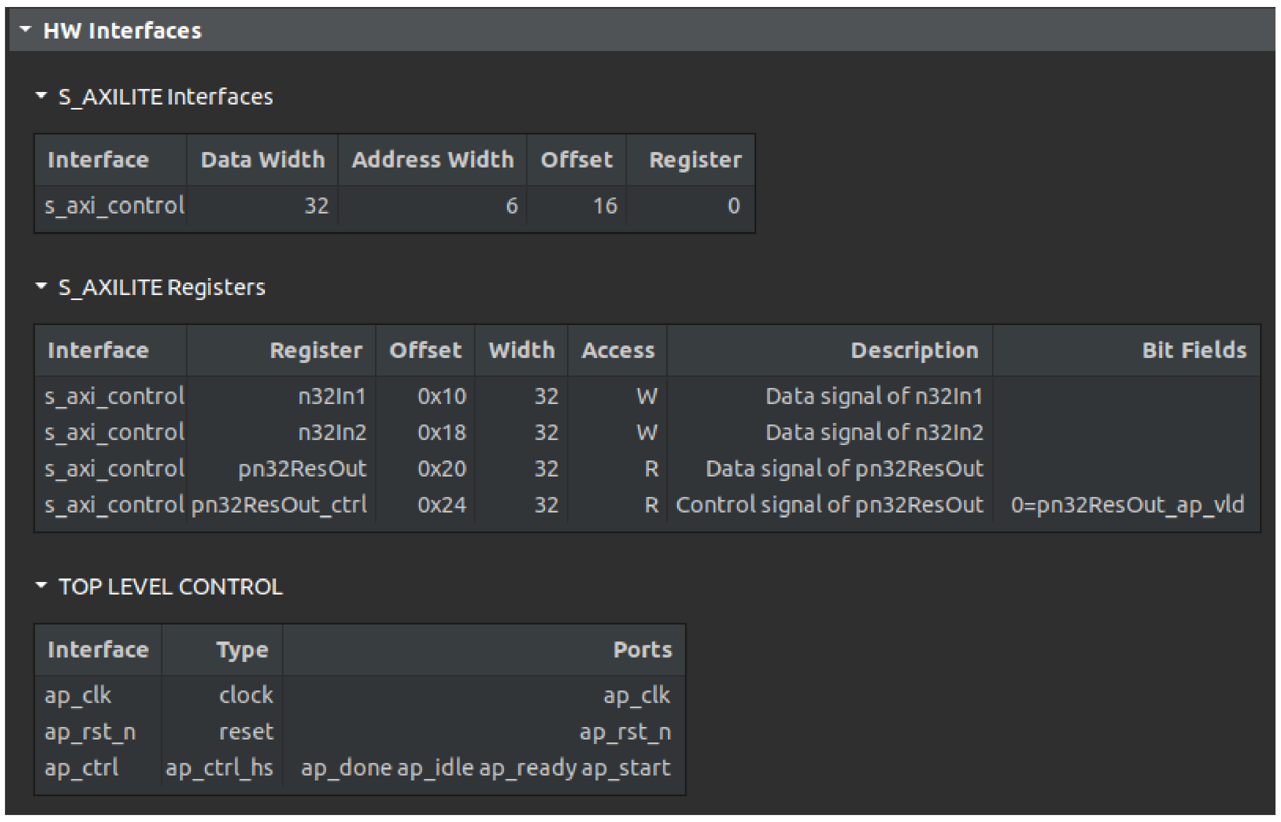
Utilization



BRAM	DSP	FF	LUT	URAM
0	3	409	307	0

Synthesis Report

Interface



HW Interfaces

S_AXILITE Interfaces

Interface	Data Width	Address Width	Offset	Register
s_axi_control	32	6	16	0

S_AXILITE Registers

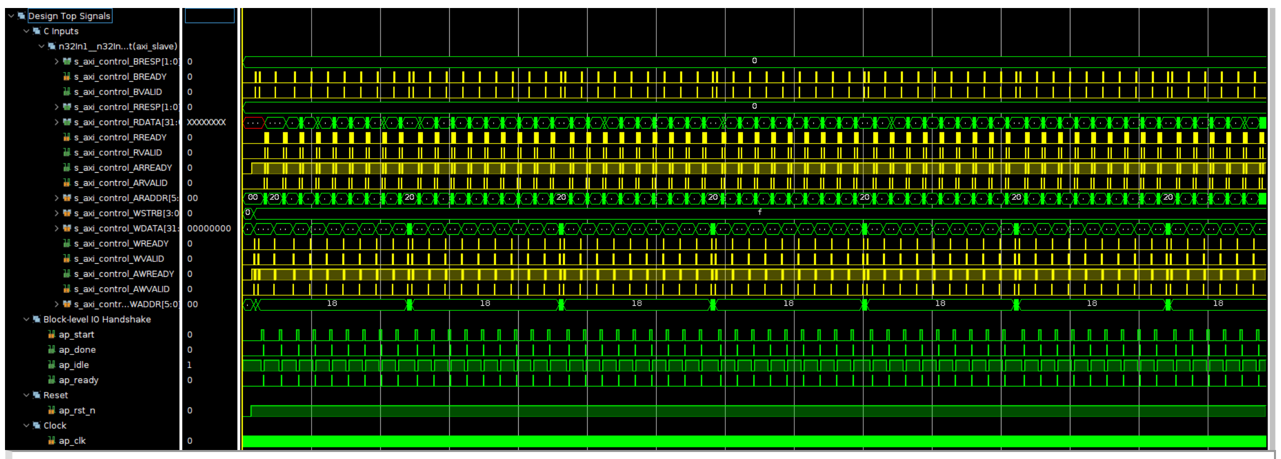
Interface	Register	Offset	Width	Access	Description	Bit Fields
s_axi_control	n32In1	0x10	32	W	Data signal of n32In1	
s_axi_control	n32In2	0x18	32	W	Data signal of n32In2	
s_axi_control	pn32ResOut	0x20	32	R	Data signal of pn32ResOut	
s_axi_control	pn32ResOut_ctrl	0x24	32	R	Control signal of pn32ResOut	0=pn32ResOut_ap_vld

TOP LEVEL CONTROL

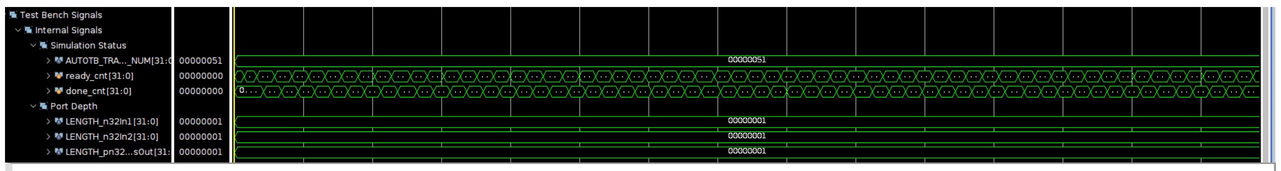
Interface	Type	Ports
ap_clk	clock	ap_clk
ap_rst_n	reset	ap_rst_n
ap_ctrl	ap_ctrl_hs	ap_done ap_idle ap_ready ap_start

Synthesis Report

Co-simulation transcript/waveform

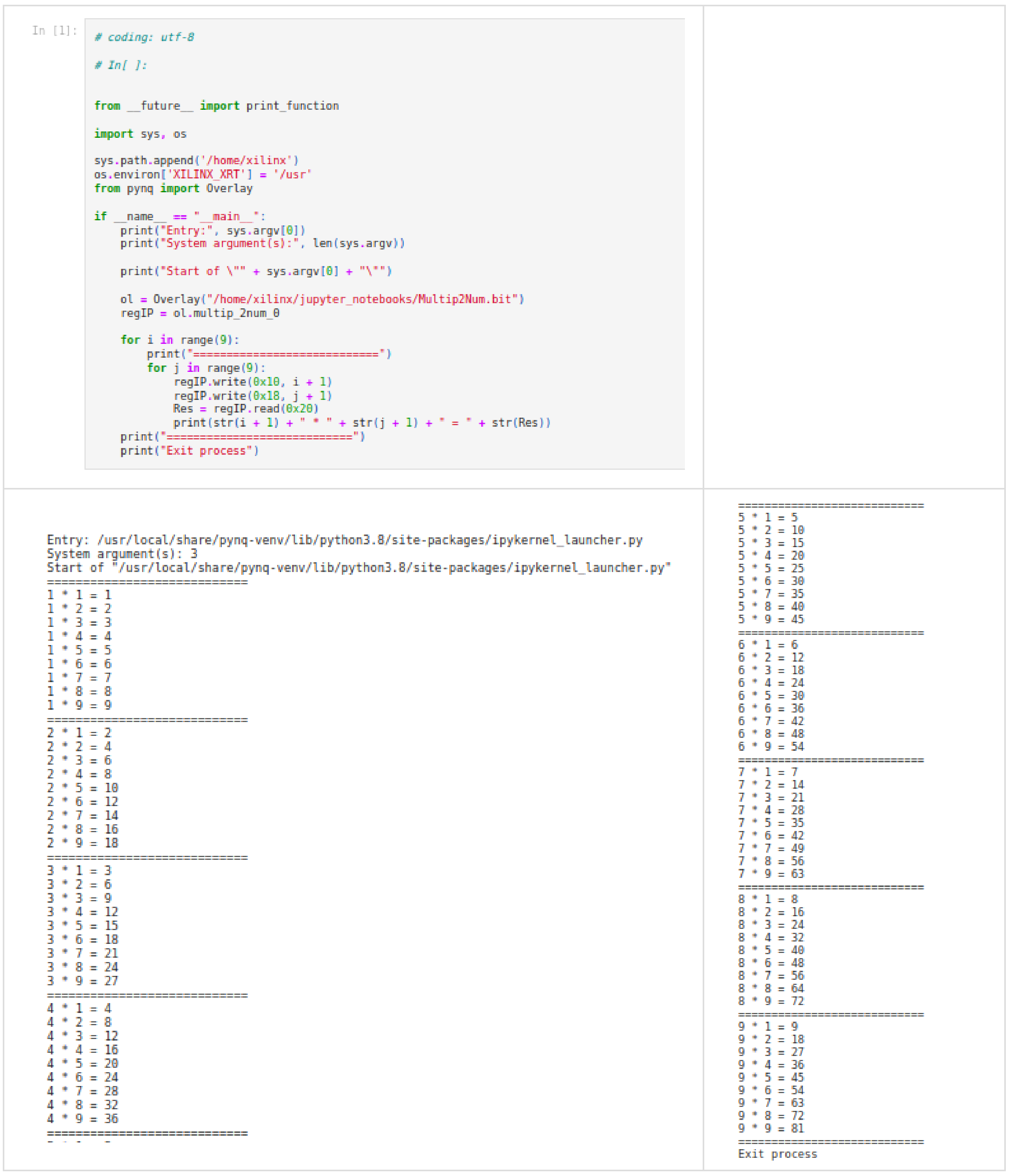


Design Waveform: Block and Port level signal


Test Bench Signals
Internal Signals
Simulation Status
AUTOTB_TRA..._NUM[31:0 00000051
ready_cnt[31:0] 00000000
done_cnt[31:0] 00000000
Port Depth
LENGTH_n32In1[31:0] 00000001
LENGTH_n32In2[31:0] 00000001
LENGTH_pn32...sOut[31: 00000001

Testbench Waveform

Jupyter Notebook execution results



In [1]:

```python
# coding: utf-8

# In[ ]:


from __future__ import print_function

import sys, os

sys.path.append('/home/xilinx')
os.environ['XILINX_XRT'] = '/usr'
from pynq import Overlay

if __name__ == "__main__":
    print("Entry:", sys.argv[0])
    print("System argument(s):", len(sys.argv))

    print("Start of \"" + sys.argv[0] + "\"")

    ol = Overlay("/home/xilinx/jupyter_notebooks/Multip2Num.bit")
    regIP = ol.multip_2num_0

    for i in range(9):
        print("============================")
        for j in range(9):
            regIP.write(0x10, i + 1)
            regIP.write(0x18, j + 1)
            Res = regIP.read(0x20)
            print(str(i + 1) + " * " + str(j + 1) + " = " + str(Res))
    print("============================")
    print("Exit process")
```

```
Entry: /usr/local/share/pynq-venv/lib/python3.8/site-packages/ipykernel_launcher.py
System argument(s): 3
Start of "/usr/local/share/pynq-venv/lib/python3.8/site-packages/ipykernel_launcher.py"
============================
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
============================
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
============================
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
============================
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
============================
============================
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
============================
6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
6 * 9 = 54
============================
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
============================
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
============================
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
============================
Exit process
```