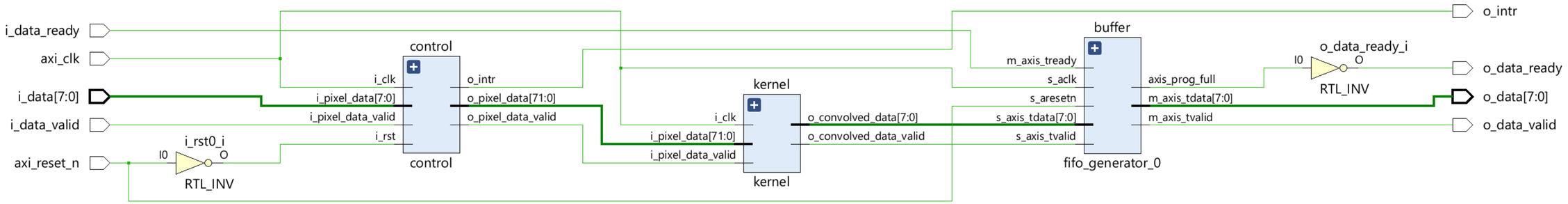
Image Processing with Pynq Z1

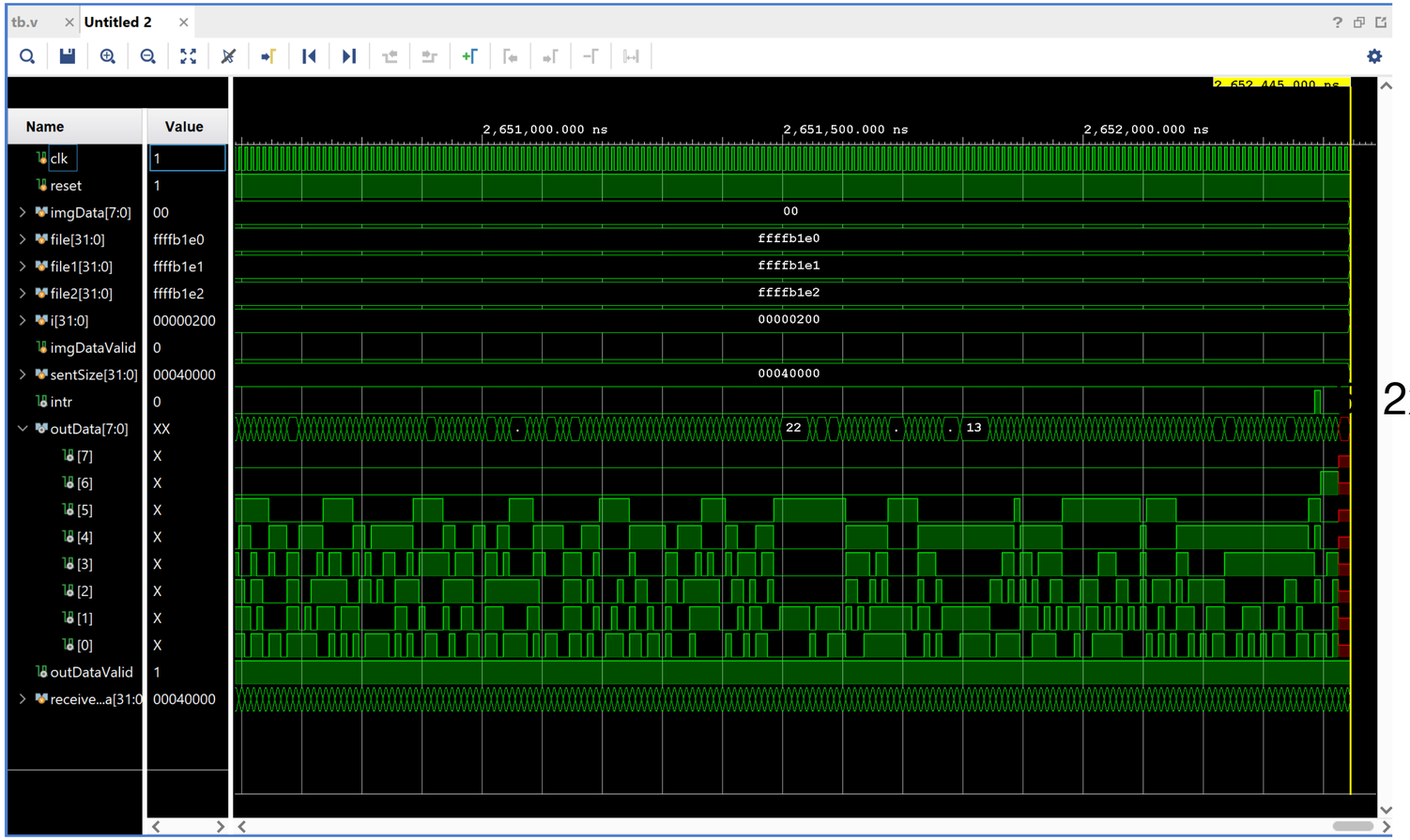
Rithani Priyanga Coimbatore Kannan, Vishnuvartthan Govindaraj

Hardware IP for 3x3 mean



Control (input buffers) -> 3x3 kernel -> output buffer

Hardware IP 3x3 Simulation

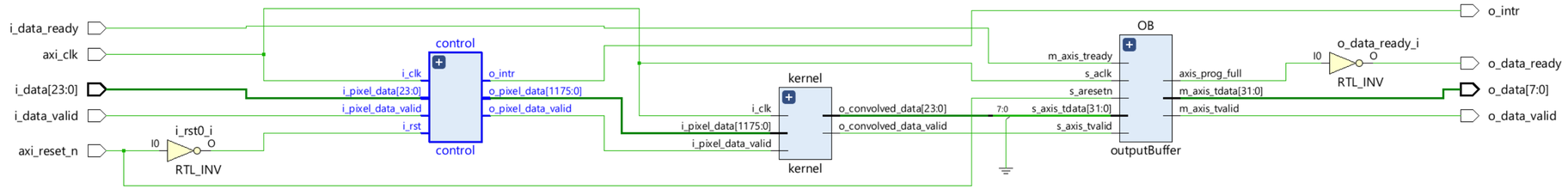


2x512 8bit

Output

Execution time: ~2.65 ms

Scaling up to 7x7 kernel



Schematics for 7x7 box filter IP

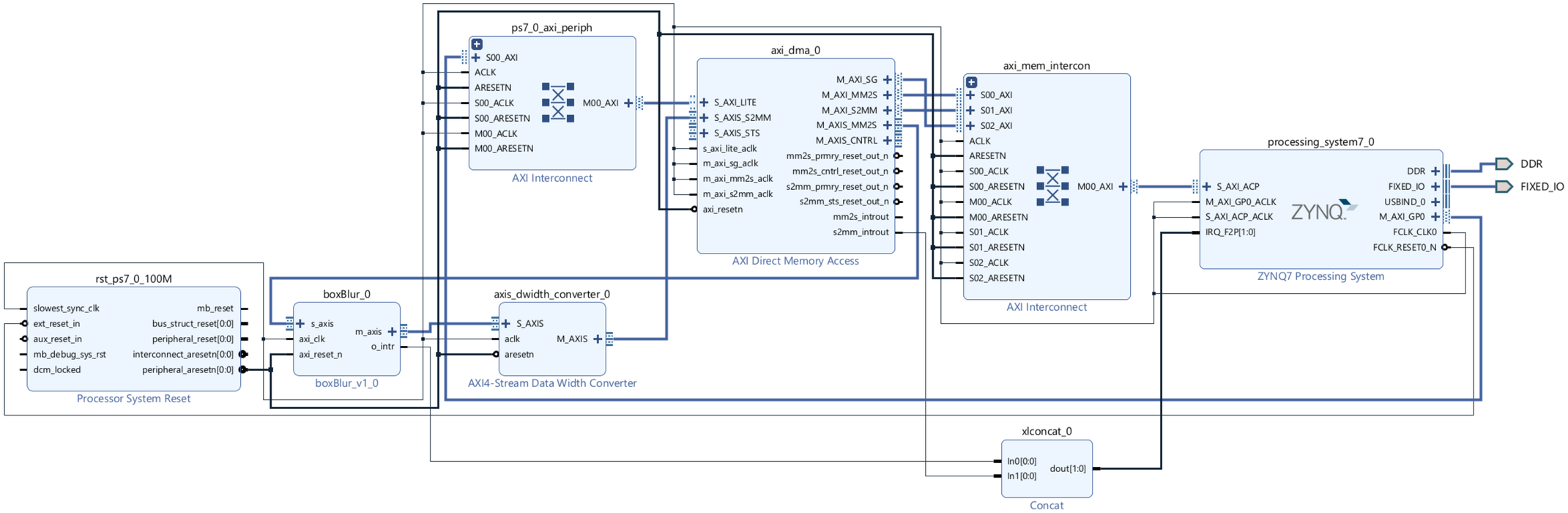
Name	Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	DSP
✓ **synth_1** (active)	**constrs_1**	**synth_design Complete!**									**7004**	**398**	**0**	**0**	**0**
✓ impl_1	constrs_1	route_design Complete!	NA	NA	NA	NA		NA	81.197	0	7067	488	1	0	0
Out-of-Context Module Runs															
✓ outputBuffer_synth_1	outputBuffer	synth_design Complete!									83	115	1	0	0

7x7, Utilization

Name	Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	LUT	FF	BRAM	URAM	DSP
✓ **synth_1** (active)	**constrs_1**	**synth_design Complete!**								**1660**	**210**	**0**	**0**	**0**
✓ impl_1	constrs_1	route_design Complete!	NA	NA	NA	NA		NA	18.942	1708	270	0.5	0	0
Out-of-Context Module Runs														
✓ fifo_generator_0_synth_1	fifo_generator_0	synth_design Complete!												

3x3, Utilization

Implementing the IP



Block design

Using xfOpenCV

3x3 kernel

Gaussian : np.array([[0.0625,0.125,0.0625],[0.125,0.25,0.125],[0.0625,0.125,0.0625]],np.float32)

Box : kernel = np.ones((3,3),np.float32)/9.0

Sobel : np.array([[1.0,0.0,-1.0],[2.0,0.0,-2.0],[1.0,0.0,-1.0]],np.float32)

Code Example

```
from pynq import Overlay, Xlnk
import pynq_cv.overlays.xv2Filter2DDilate as xv2
import cv2
import numpy as np
import matplotlib.pyplot as plt
import time
from pynq import allocate

ol = Overlay("/usr/local/lib/python3.6/dist-packages/pynq_cv/overlays/xv2Filter2DDilate.bit")
Xlnk.set_allocator_library("/usr/local/lib/python3.6/dist-packages/pynq_cv/overlays/xv2Filter2DDilate.so")

og_image = cv2.imread('bellatrix.jpg', cv2.IMREAD_GRAYSCALE)
s = 0.2
image = cv2.resize(og_image, (0, 0), fx=s, fy=s, interpolation=cv2.INTER_AREA)
height, width = image.shape

xFimg = allocate(shape=(height, width), dtype=np.uint8)
xFout = allocate(shape=(height, width), dtype=np.uint8)

xFimg[:] = image[:]
xFimg.flush()
kernel = np.ones((3,3),np.float32)/9.0

start_time = time.time()
xv2.filter2D(xFimg, -1, kernel, dst=xFout, borderType=cv2.BORDER_CONSTANT)

end_time = time.time()
elapsed_time = (end_time - start_time) * 1000

print(f"Image size       : {width} x {height} pixels")
print(f"Kernel size      : {kernel_size[0]} x {kernel_size[1]}")
print(f"Box blur (mean filter) execution time: {elapsed_time:.3f} ms")
```

Execution

Original Image

Sobel Edge Detection

Execution time on PS and PL

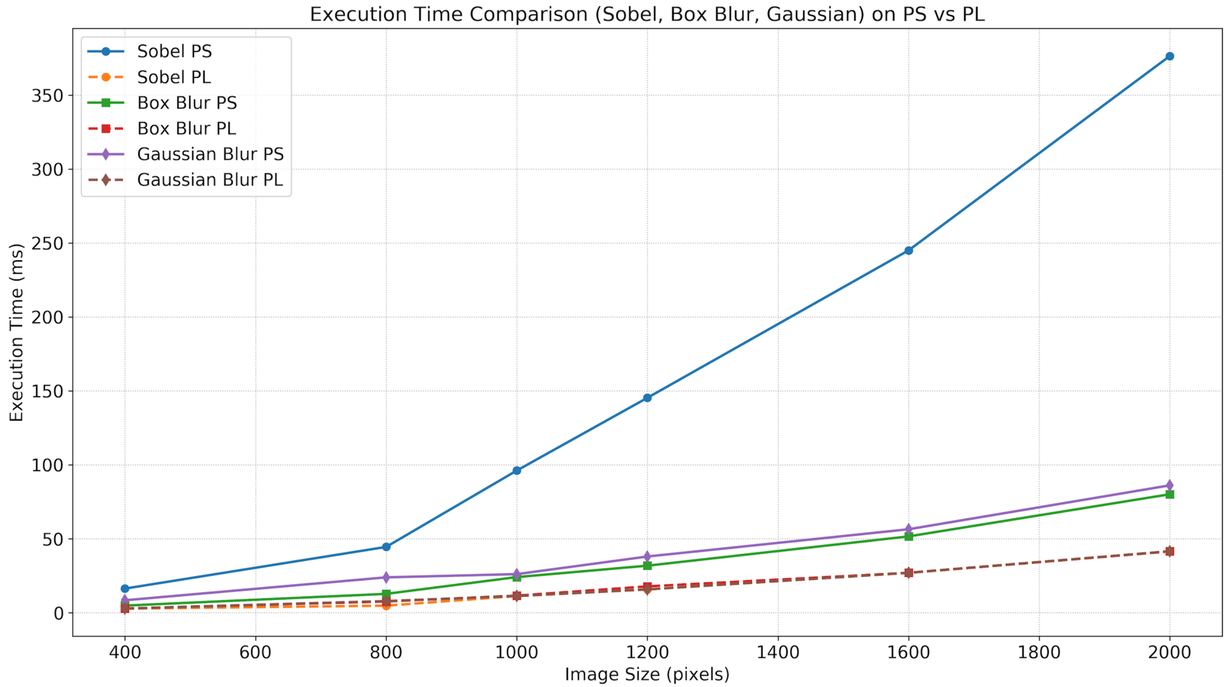


Image Size (px)	Sobel PS (ms)	Sobel PL (ms)	Box PS (ms)	Box PL (ms)	Gaussian PS (ms)	Gaussian PL (ms)
400	16.275	2.728	4.752	2.797	8.370	2.792
800	44.463	4.726	12.710	7.543	23.873	7.824
1000	96.104	11.239	24.054	11.502	26.055	11.245
1200	145.298	15.683	31.764	17.763	37.966	15.752
1600	245.032	26.946	51.576	26.929	56.404	26.953
2000	376.466	41.414	80.076	41.435	86.093	41.550

Thank You