A CNN Accelerator for Edge devices.

Name: Ayyappa Koppuravuri.
Roll no: IMT2020555.
Github link: https://github.com/Ayyappa1911/A-CNN-Accelerator-for-Edge-Devices.git

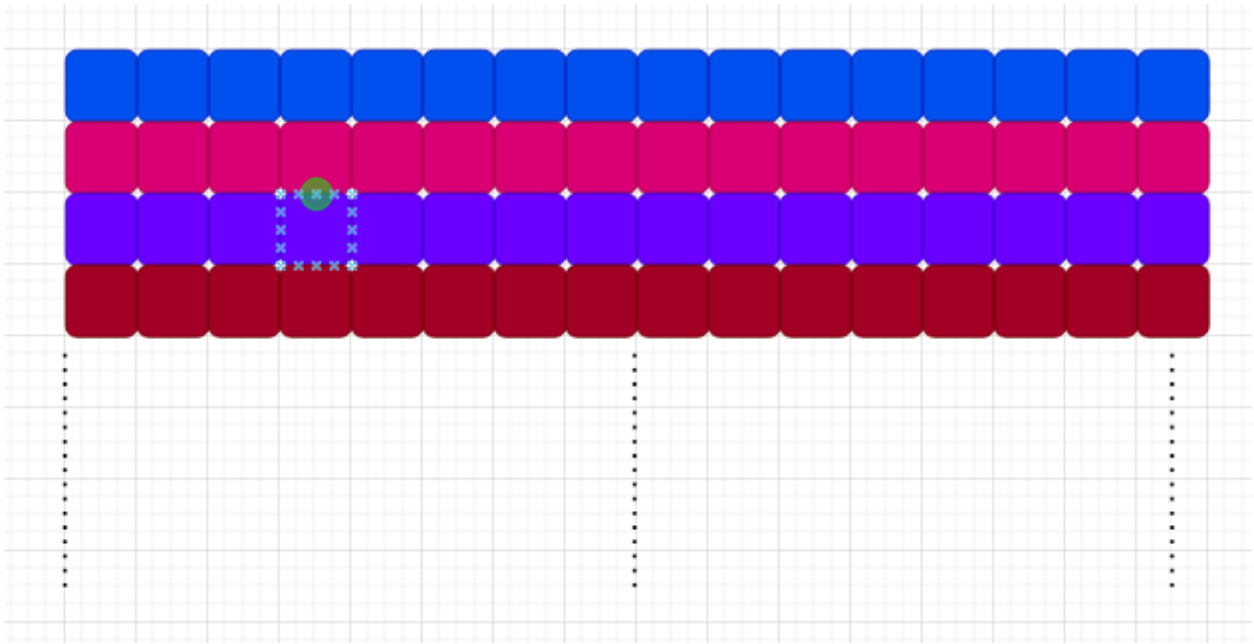
The Initial idea is to develop a system with PICORV_32 with AXI_interface and operate a Convolution operation. Later extending it to a single Layer of Convolutional Neural networks(preferably of Lenet-5)

I am designing a new architecture for Convolution using a systolic array for dot product which is compatible to operate through PICORV32 with AXI-lite interface.

AXI4-lite interface provides only a single pixel of image data per data transaction to the slave, and can only read one Pixel data of the image from the slave per transaction.

So we have to design a Convolution operation that gets 1 data per transaction using data_request(requesting for the data if needed) signals, and data_valid(whether the input data is valid or not) signals.

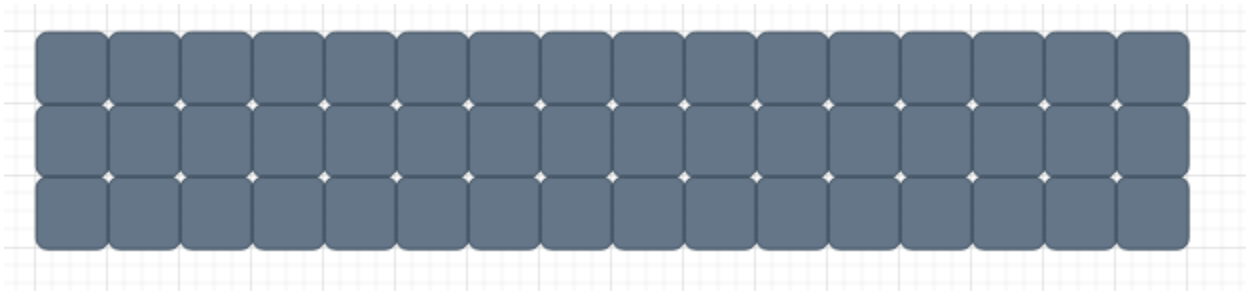
Image data: Let's input the following image data for conv.



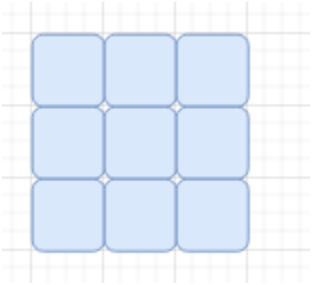
The data of the above image is transmitted via PICORV_32 using AXI4_lite interface pixel after pixel i.e., 1 unit of data per transaction.

The Colours for each pixel are to differentiate the rows(They don't have any particular significance to the values of the pixel).

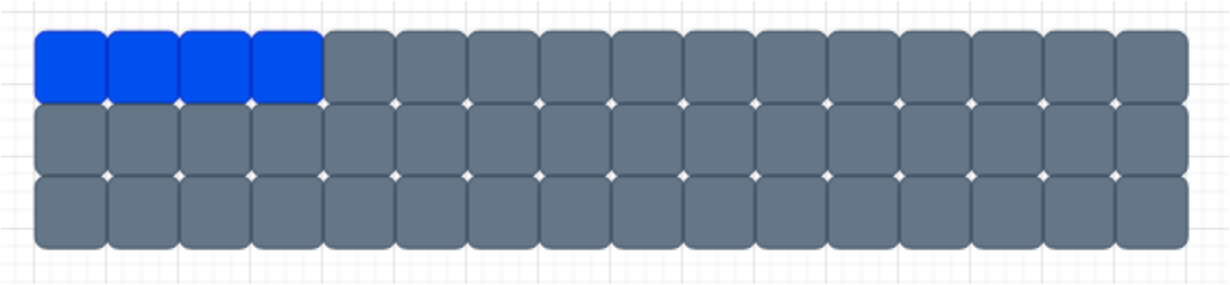
- We will maintain a 2-D array with dimensions as (img_length x kernel_length) in this convolution method. As shown in the below figure. The gray color of a pixel indicates an invalid value.



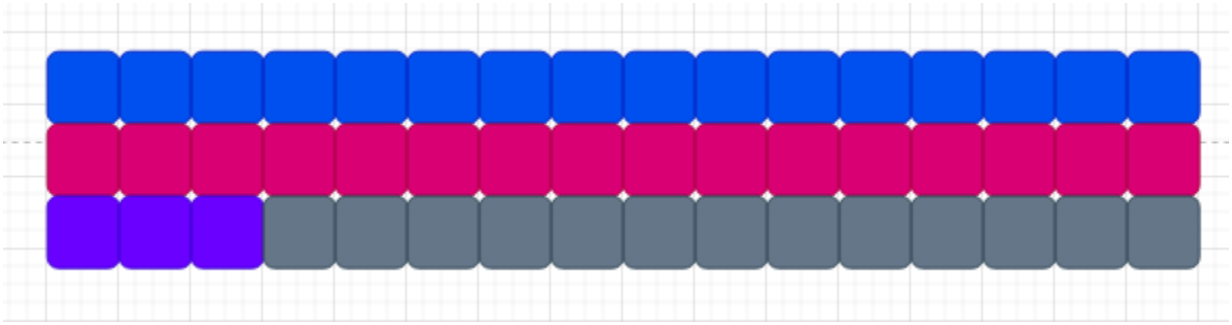
- In this explanation let us consider a 3x3 Kernel.



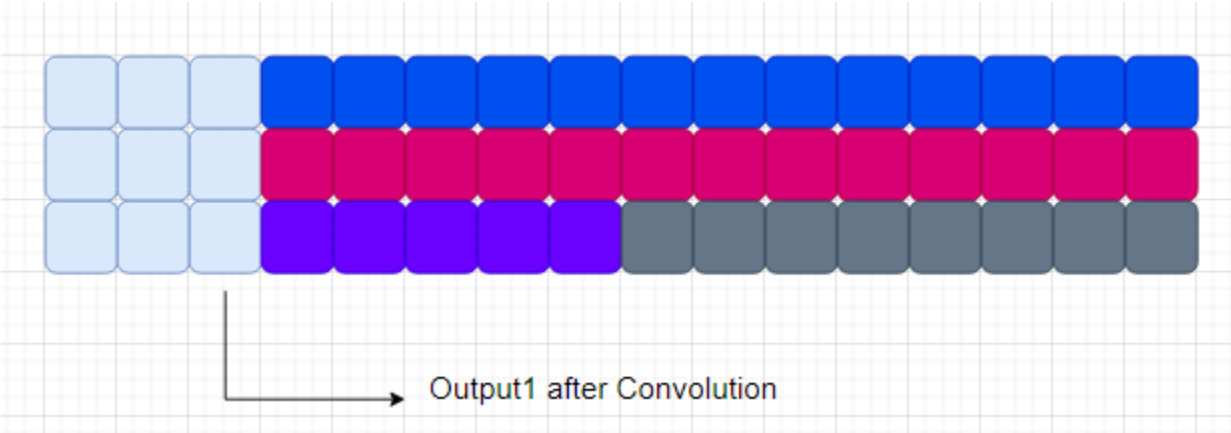
- Initially, data is filled in the 2-D array one pixel after another per transaction until enough data is sufficient for convolution. Using data_reuest and data_valid signals.



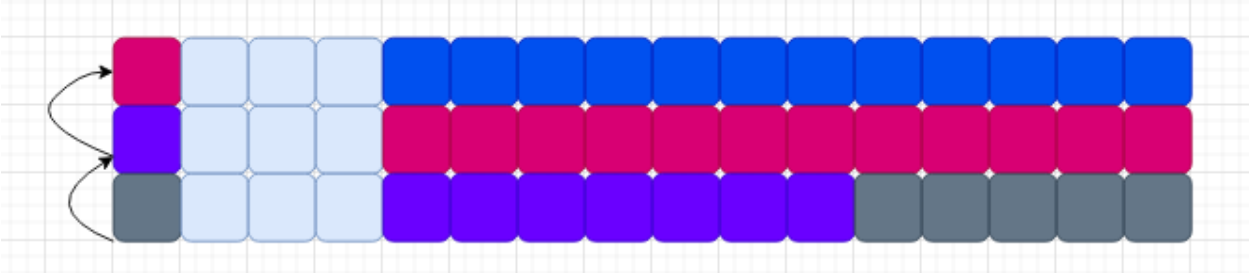
- After the data is sufficient for convolution.



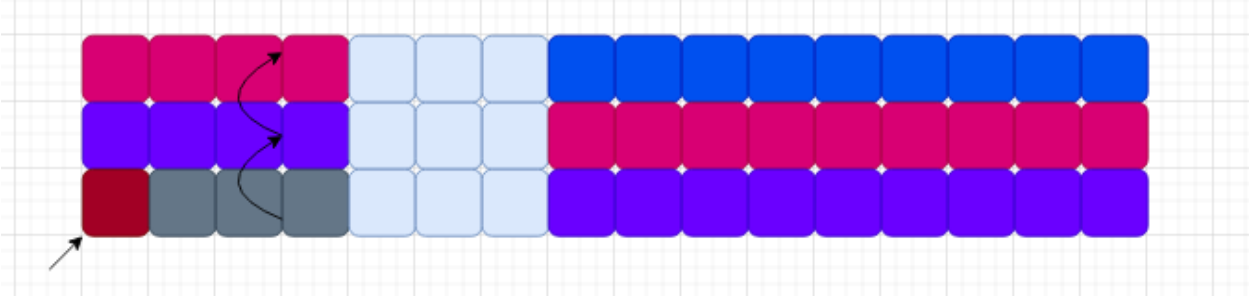
- Now the filter starts sliding on this 2-D array. The convolution operation is started. Now the remaining pixels of data are also filled parallelly.



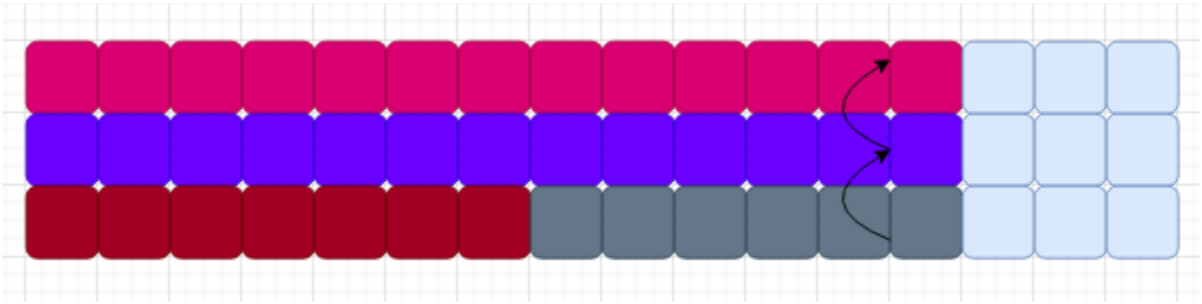
- After the convolution at the first pixel, the kernel will start moving horizontally. The previous values in the pixels involved in convolution are rearranged according to reuse them for further convolutions.



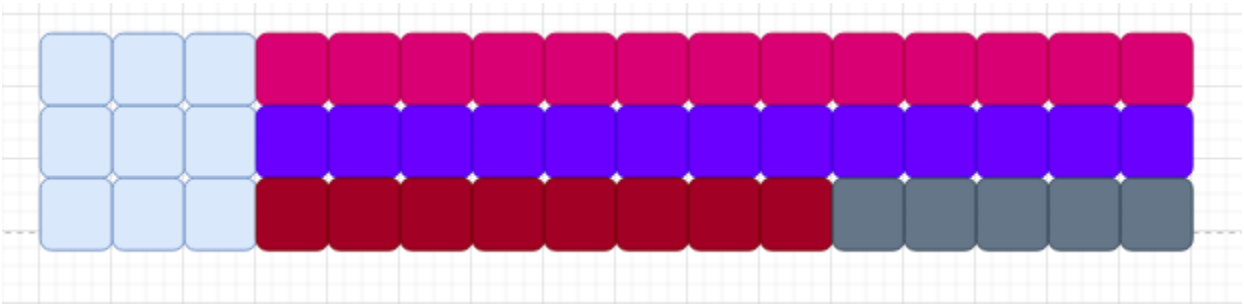
- This process goes on till the Last row is filled. After the last row is filled new row automatically will be filled with the 4th row of the Image.



- This step goes on until the kernel reaches the end of the array.



- Finally, the kernel will start from the beginning of the array, and the process repeats.



Synthesis Results for 32x32 image and the kernel size is 5x5.

Resource Utilization:

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	DSPs (90)	Bonded IOB (106)	BUFGCTRL (32)
full_conv	4718	5233	15	5	869	1
conv_ut (conv)	4360	2575	15	5	0	0
xyz (dotpdt)	2523	1873	0	5	0	0
sysarr (iiitb	16	864	0	5	0	0

Timing Analysis. (10ns clock)

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	-6.944 ns	Worst Hold Slack (WHS):	0.097 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	-2831.592 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	1102	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	9951	Total Number of Endpoints:	9951	Total Number of Endpoints:	5238

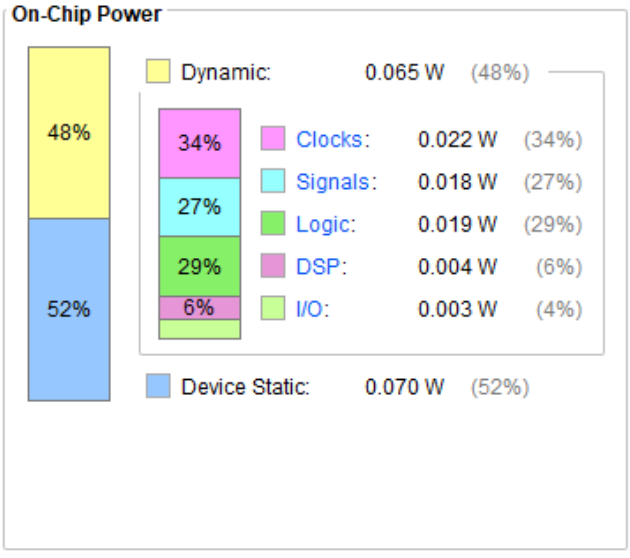
Timing constraints are not met.

Power:

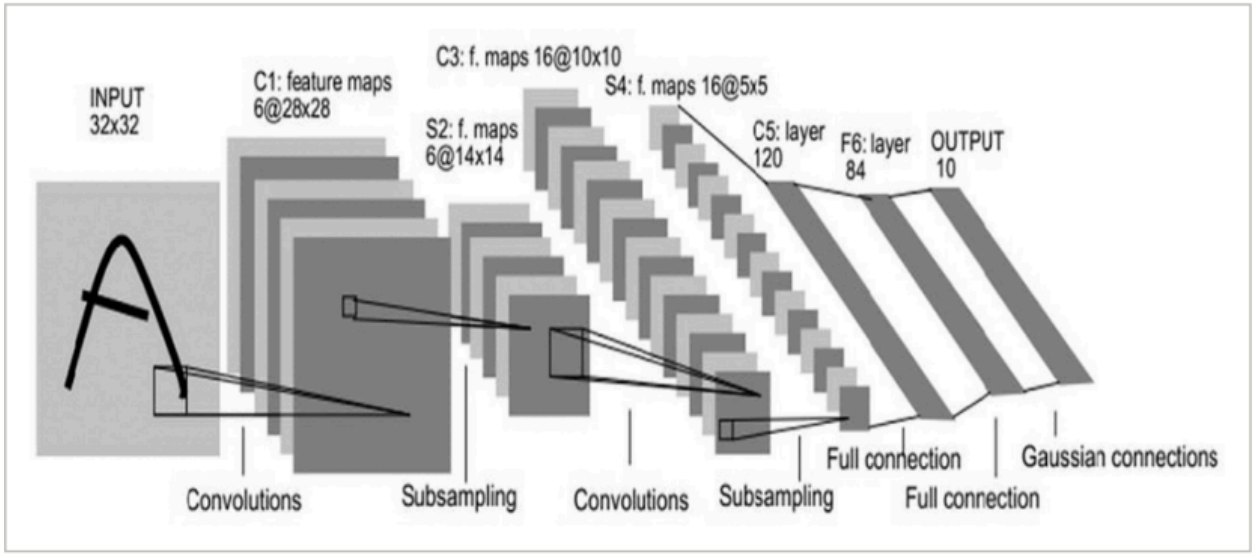
Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.136 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.7°C**
Thermal Margin:	59.3°C (11.8 W)
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

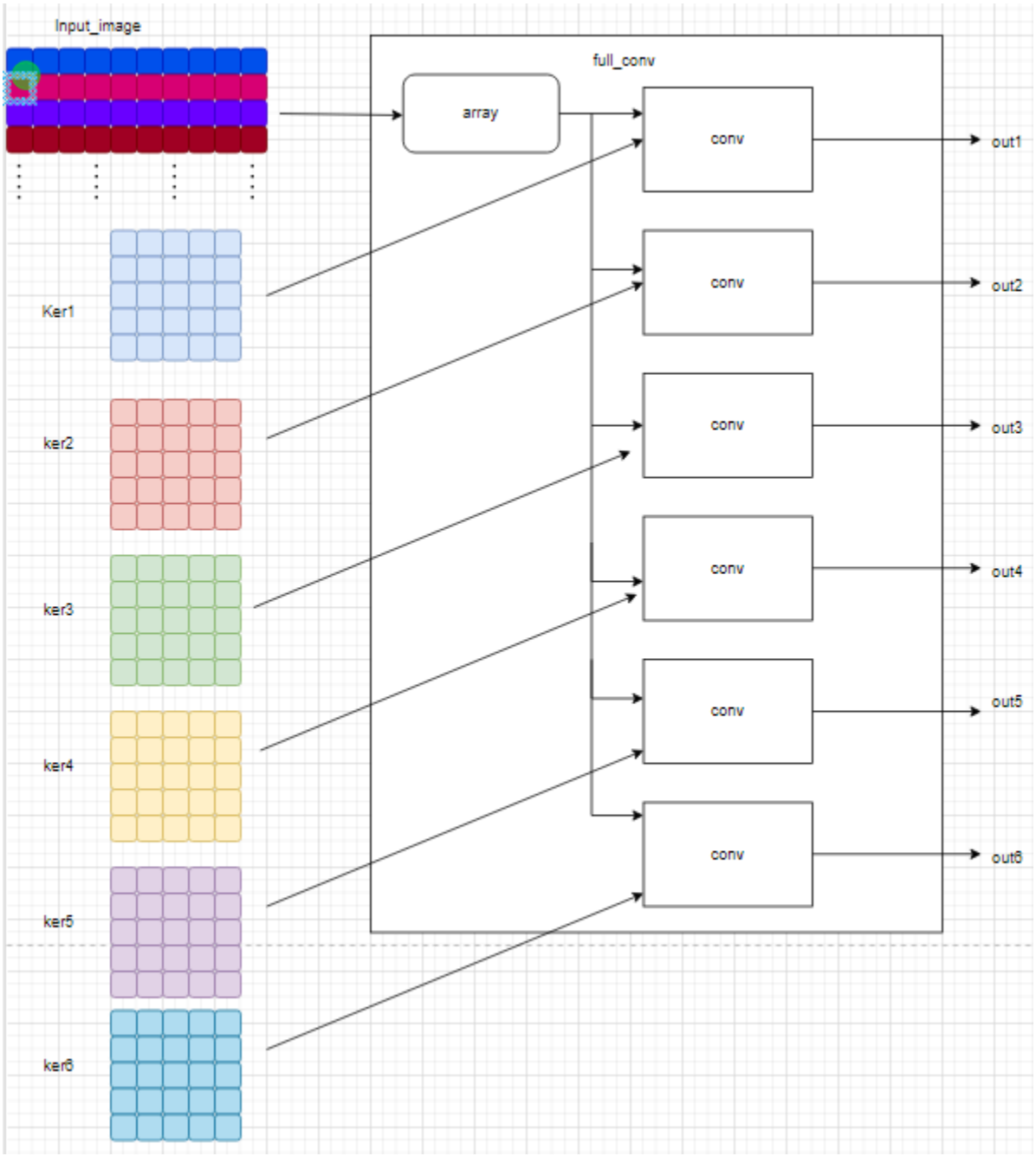


Extending the above convolution to LENET-5 CNN Layer-1



- Lenet-5 layer 1 takes an input of 32x32 size.
- The size of the kernels is 5x5.
- There will be 6 feature maps of size 28x28 after the first layer of convolution
- Max pooling with stride 2 is done after conv layer 1 which gives an output of 6 feature maps of size 14x14.

Block diagram for LENET-5 Layer 1 implementation.



We maintain separate copies of Conv for all 6 different kernels. As all the kernels are the same size they slide over the array in the full_conv module at the same point.

Synthesis Results of LENET-5 conv Layer 1.

Timing Analysis. (10ns clock)(60.27MHz)

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	-6.592 ns	Worst Hold Slack (WHS):	0.103 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	-16575.526 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	6576	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	35856	Total Number of Endpoints:	35856	Total Number of Endpoints:	18133

Timing constraints are not met.

Utilization:

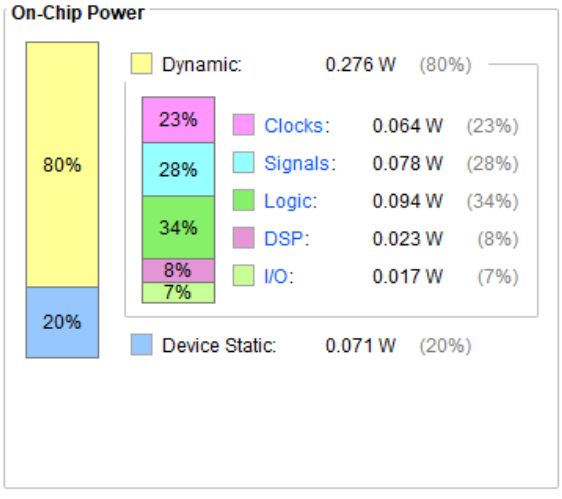
Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	DSPs (90)	Bonded IOB (106)	BUFGCTRL (32)
full_conv	22218	18108	90	30	5109	1
conv_ut1 (conv)	3452	2575	15	5	0	0
xyz (dotpdt_136)	1456	1873	0	5	0	0
sysarr (iiitb_sy	16	864	0	5	0	0
conv_ut2 (conv_0)	3452	2575	15	5	0	0
conv_ut3 (conv_1)	3452	2575	15	5	0	0
conv_ut4 (conv_2)	3452	2575	15	5	0	0
conv_ut5 (conv_3)	3454	2575	15	5	0	0
conv_ut6 (conv_4)	3452	2575	15	5	0	0

Power:

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.347 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**26.7°C**
Thermal Margin:	58.3°C (11.6 W)
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

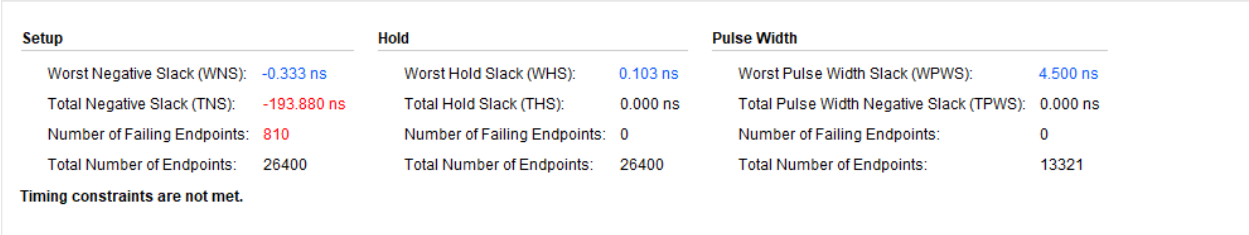
Launch Power Constraint Advisor to find and fix invalid switching activity



- **The above results for the CONV unit with a programmable dot product. (Dot product of larger-sized matrices can be done using smaller-sized systolic arrays). This is causing an overhead in terms of resources.**
- **As our idea is to develop a convolution unit for low-area devices, we are referring to smaller area-consuming designs. Therefore, we are removing this programmable functionality of the dot product.**

Synthesis Results for LENET-5 conv Layer 1. (without programmable functionality to the dot product.)

Timing Analysis. (10ns clock)(96.777MHz)



Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	-0.333 ns	Worst Hold Slack (WHS):	0.103 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	-193.880 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	810	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	26400	Total Number of Endpoints:	26400	Total Number of Endpoints:	13321

Timing constraints are not met.

Utilization:

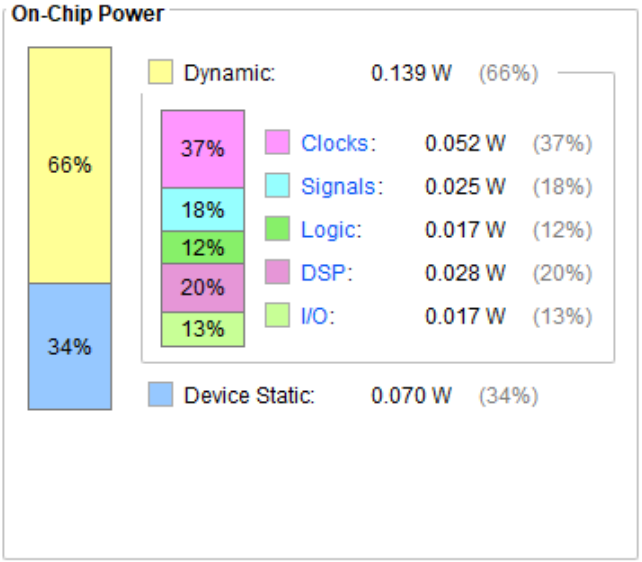
Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	DSPs (90)	Bonded IOB (106)	BUFGCTRL (32)
full_conv	12384	13296	90	30	5109	1
conv_ut1 (conv)	1854	1773	15	5	0	0
xyz (dotpdt_136)	81	1073	0	5	0	0
sysarr (iiitb_sy	16	864	0	5	0	0
conv_ut2 (conv_0)	1854	1773	15	5	0	0
conv_ut3 (conv_1)	1854	1773	15	5	0	0
conv_ut4 (conv_2)	1854	1773	15	5	0	0
conv_ut5 (conv_3)	1854	1773	15	5	0	0
conv_ut6 (conv_4)	1856	1773	15	5	0	0

Power:

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.209 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**26.0°C**
Thermal Margin:	59.0°C (11.7 W)
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

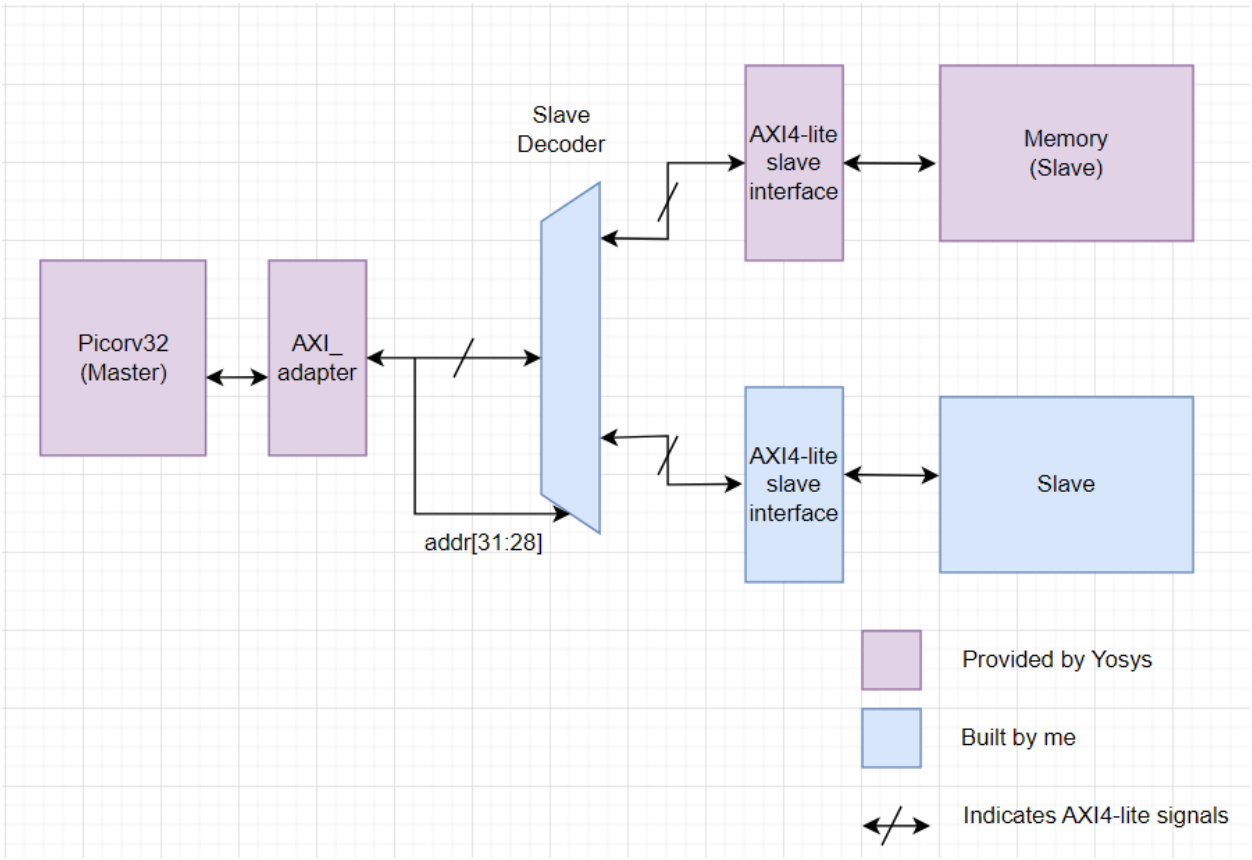


- The Frequency of operation increased from 60.27 MHz to 96.777 MHz). The resources and the power consumed also decreased significantly.

Design of the Entire system.

- The current Convolution design is usable as the frequency of operation is around 100MHz for MNIST CNN Convolution Layer 1. Further optimization can be done according to the requirements.
- Let's discuss the design of the Entire system(Convolution + PICORV32_AXI)

  Block diagram for the entire system.



- The convolution/CNN Layer 1 will be operated by the PICORV32 using AXI4_lite protocols. In the above block diagram, the AXI4-lite interfaces in purple are provided by the designers of picorv32.

- Here in the above diagram, we can observe that there are multiple slaves for a single master. Therefore, we have to build a slave decoder that supports AXI-4 lite signals.
    - The slave decoder is controlled by the 4 MSB bits of the address provided by the Master. (**This will allow the system to support up to 16 slaves**). The addresses in the particular range will indicate which slave is probed by the master.
    - The advantage of this design is that no changes are made to the original RISC-V core. This will enable the design compatible with the toolchains provided by the RISC-V in the future.
    - The disadvantage of this address split will decrease the maximum memory size.
    - If slave_dec == 4'b0000 the signals are diverted to AXi_memory.
    - If slave_dec == 4'b0001 then the signals are diverted to slave1.
- We have to build our own AXI-4 lite slave interface for the additional slaves in the design(to support AXI4-lite protocols).
- The data for operating the CNN will be read/stored in the memory module. Therefore there has to be a particular protocol for that to happen.

AXI4-Lite.

- Both the master and slave can control the data flow. However, the data operation can only be decided by the master. The operations are:
    - Read - The master reads the data from the slave.
    - Write - The master writes the data to the slave.

Procedure:

- The master uses the Load and the store instructions to read the data from the slave and write the data to the slave.
- If a single pixel-data(input of CNN) has to be written to the slave.
    - The master has to read the data from the memory using the load instruction with an appropriate address.
    - Then the master has to write the data into the slave using the store instruction with an appropriate address.
- If a single pixel data (output of CNN) has to be read from the slave.
    - The master has to read the data from the slave using the load instruction with an appropriate address.
    - Then the master has to write the data into the memory using the store instruction with an appropriate address.

Literature Surveys:

Variable Bit-Precision Vector Extension for RISC-V Based Processors.

- This paper changes RISC-V's ISA to support Multiply and add operations. The work is demonstrated on FPGAs using PICORV-32 for MAC operations.

A Generic On-Board Computer based on RISC-V Architecture Processor for Low Cost Nanosatellite Applications.

- Complete architecture for Nanosatellite applications using PICORV-32.

Classifying Computations on Multi-Tenant FPGAs

-

TIGRA: A Tightly Integrated Generic RISC-V Accelerator Interface.

- TIGRA is an interface that can Integrate accelerators like Posit arithmetic, AES, etc. by adding some extra hardware to the RISC-V core but without changing ISA. This work is demonstrated for PICORV-32.

OpenSpike: An OpenRAM SNN Accelerator.

- This paper demonstrates an SNN accelerator taped out using the Efabless, Google, and the Sky Water Foundry open-source program. The auxiliary controller in this chip is Picorv32.

CNN Specific ISA Extensions Based on RISC-V Processors.

Design and Implementation of CNN Custom Processor Based on RISC-V Architecture.

Extending a Soft-Core RISC-V Processor to Accelerate CNN Inference.

- ISA changes were made in this paper.

Implementation of CNN Heterogeneous Scheme Based on Domestic FPGA with RISC-V Soft Core CPU.

- .Designed an accelerator to support CNN acceleration on local FPGAs using a RISC-V core.

A Programmable CNN Accelerator with RISC-V Core in Real-Time Wearable Application

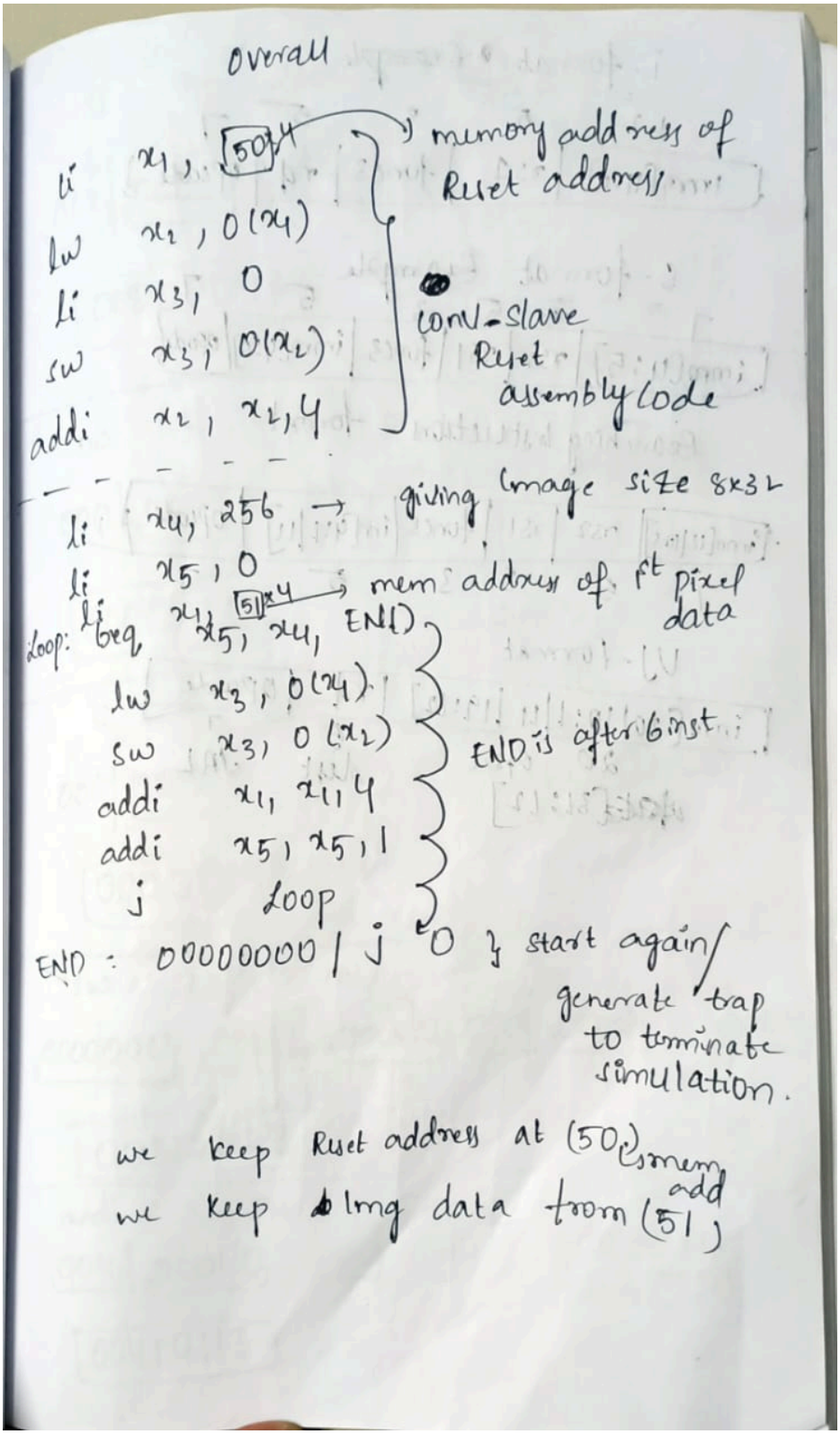
- They have implemented a network using a RISC-V core as a controller and an 8-layer CNN classifier. The architectural idea is entirely different.

Integrated design:

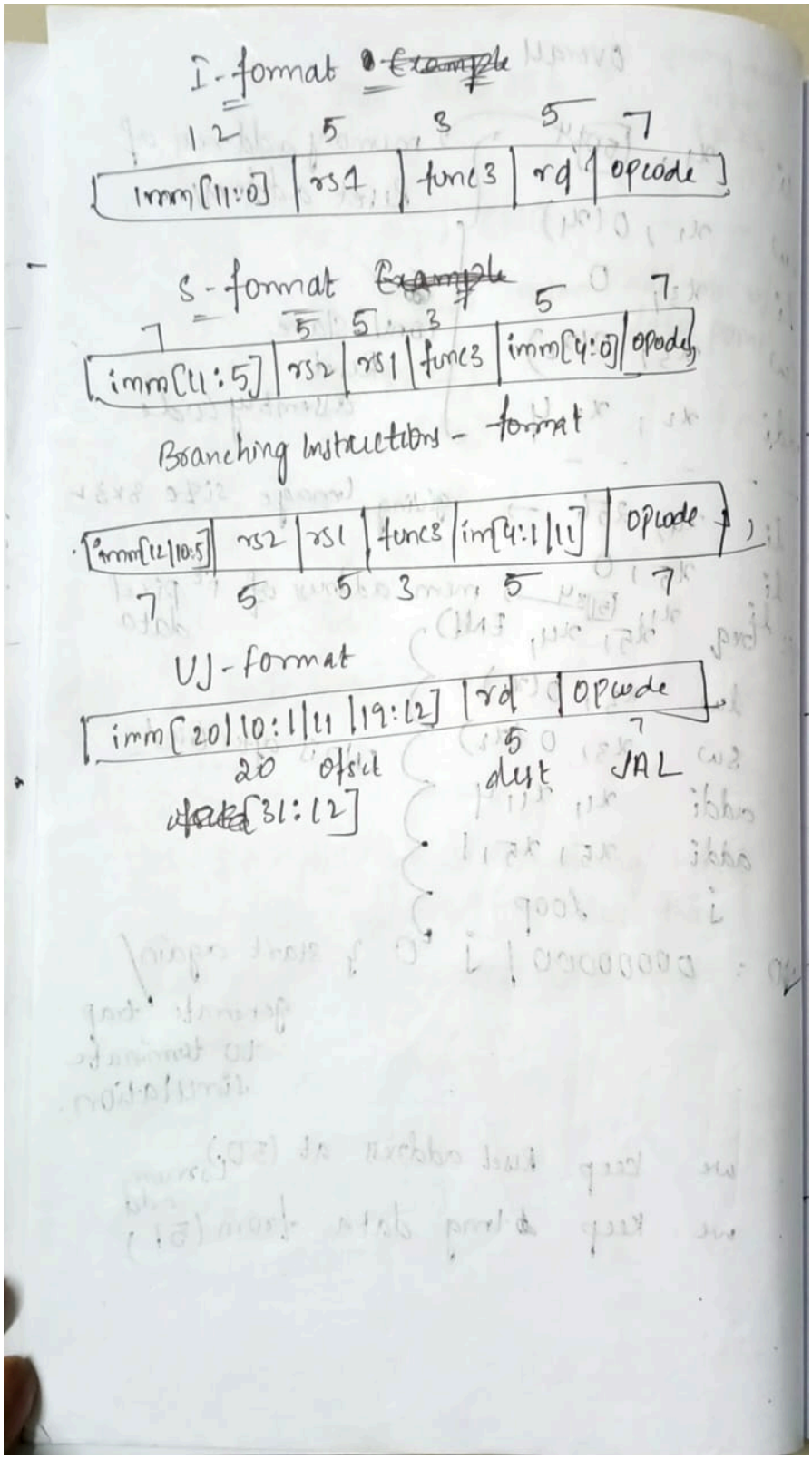
Let us start the integration of both the above modules, beginning with resetting the Convolution module.

- The integrated design was implemented by writing some random number to the address with the 28 LSBs zero.
- For the address 32’h1000_0004, the pixel data can be written into the convolution slave module.
- Here, we are only starting with the master writing the image data into the slave module. Later the Functionality of the master reading the output data from the slave module will be added.

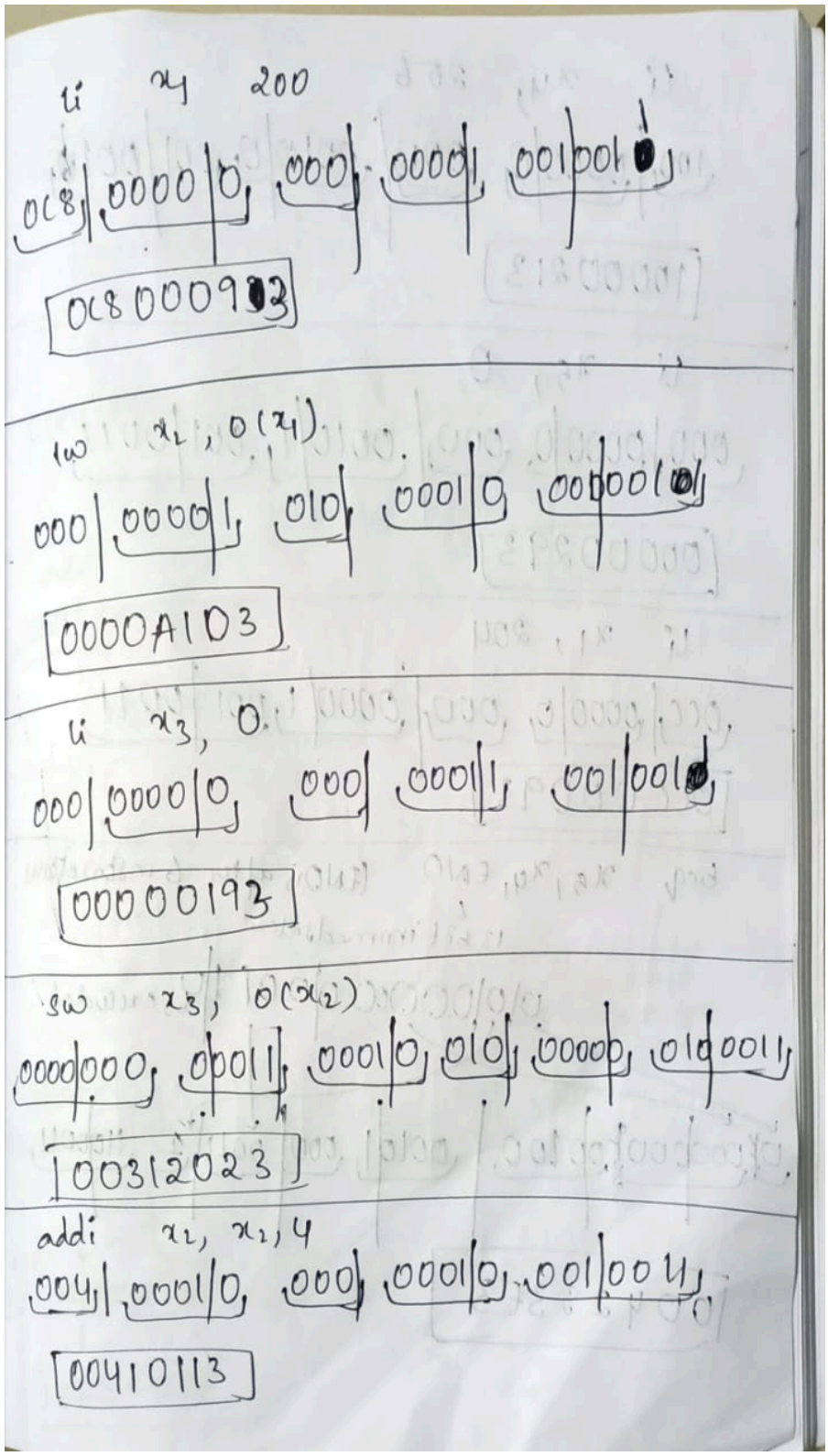
Assembly code for the above assumptions:



Some important RISC-V instruction formats:



Converting the above Assembly code into Hexcode:



li x4, 256

100 | 0000 0 | 000 | 0010 0 | 001 | 0010

10000213

li x5, 0

000 | 0000 0 | 000 | 0010 1 | 001 | 0011

00000293

li x1, 204

0CC | 0000 0 | 000 | 0000 1 | 001 | 0011

0CC00093

beq x5, x4, END (END is after 6 instructions)

13 bit immediate

0 | 0 | 000000 | 0001 | 1 ← excluded

0 | 000000 | 00100 | 00101 | 000 | 0011 | 1 | 110001

004283E3

lw $x_3, 0(x_1)$

000 | 0000 | 010 | 0001 | 1 | 000 | 0011

000DA183

sw $x_3, 0(x_2)$

0000 | 000 | 00011 | 00010 | 010 | 00000 | 010 | 0011

00312023

addi $x_1, x_1, 4$

004 | 0000 | 1 | 000 | 00001 | 001 | 0011

00408093

addi $x_5, x_5, 1$

001 | 0010 | 1 | 000 | 00101 | 001 | 0011

00128293

j loop → loop is 5 instructions back
we have to go
5 instructions Back as
PC stores
rd = 0 for j
Offset = −5×4
= −20

offset = 1111 1111 1111 1010 (0 not used)

1 | 111 | 1111 | 101 | 1 | 1111 | 1111 | 00000 | 110 | 1111

FFBFF06F

j loop → loop is 5 instructions back

Offset = $-5 \times 4 = -20$

↓

jump instruction is implemented using jal instruction with $rd = x_0$

as jal does the operation $PC = PC + Offset$

for = 20 binary (21 bit number)
= 16+4

0|0000|0000|0000|0001|0100

1's comp

1|1111|1111|1111|1110|1011

2's comp add 1

1|1111|1111|1111|1110|1100(0) ↳ discarded in jal

1|1111 1111 |1|111 1110 110|00

1, 111|1110|110, 1|1111|1111, |00000, 110|1111

FEDFF06F

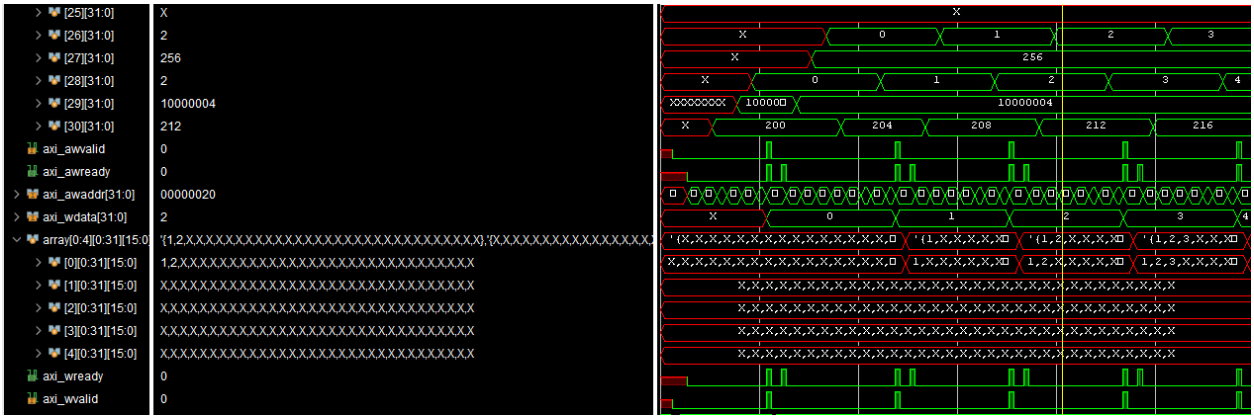
Currently, the above assembly code is stored in memory as follows:

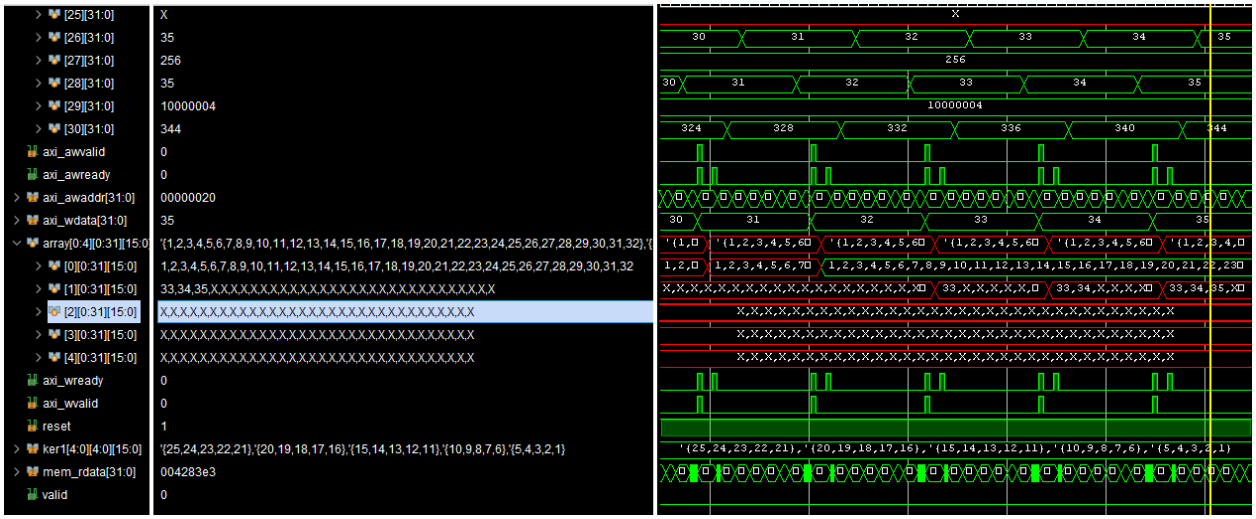
```
initial begin
    memory[0] = 32'h0c800093;
    memory[1] = 32'h0000a103;
    memory[2] = 32'h00000193;
    memory[3] = 32'h00312023;
    memory[4] = 32'h00410113;
    memory[5] = 32'h10000213;
    memory[6] = 32'h00000293;
    memory[7] = 32'h0cc00093;
    memory[8] = 32'h004283e3;
    memory[9] = 32'h0000a183;
    memory[10] = 32'h00312023;
    memory[11] = 32'h00408093;
    memory[12] = 32'h00128293;
    memory[13] = 32'hfedff06f;
    memory[14] = 0;
    memory[15] = 0;

    memory[50] = 32'h10000000; // Reset address of
    for(itt = 1; itt <= 256; itt = itt + 1) begin
        memory[50+itt] = itt;
    end

end
```

The Image data is stored from the memory location 51 till 51 + 256. I.e., testing on an 8x32 image. The kernel size is still 5x5.

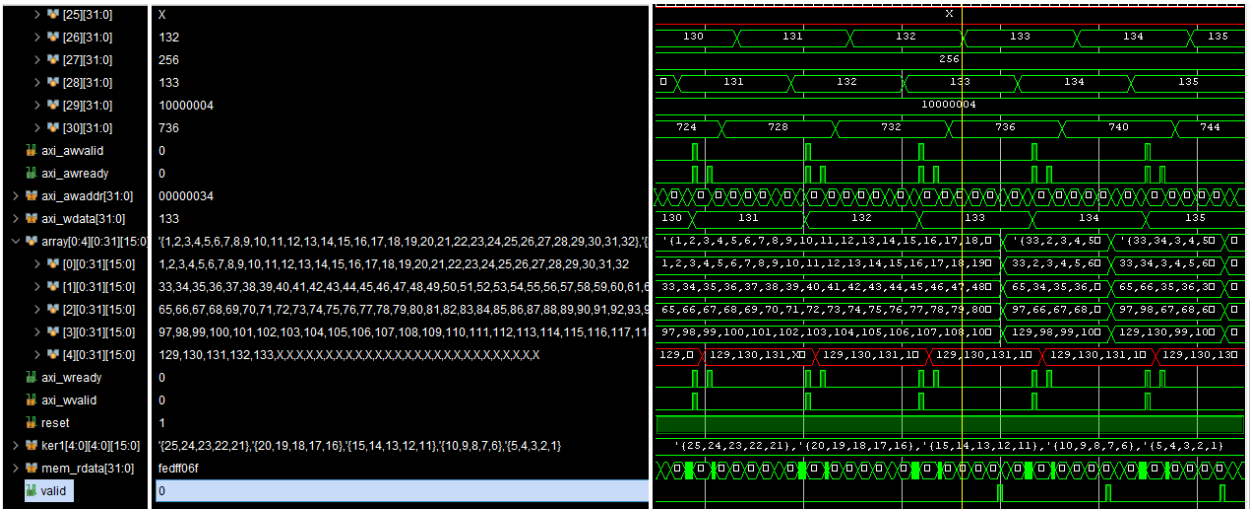
The image data is written into the buffer of the convolution module using the protocol mentioned before.

Starting

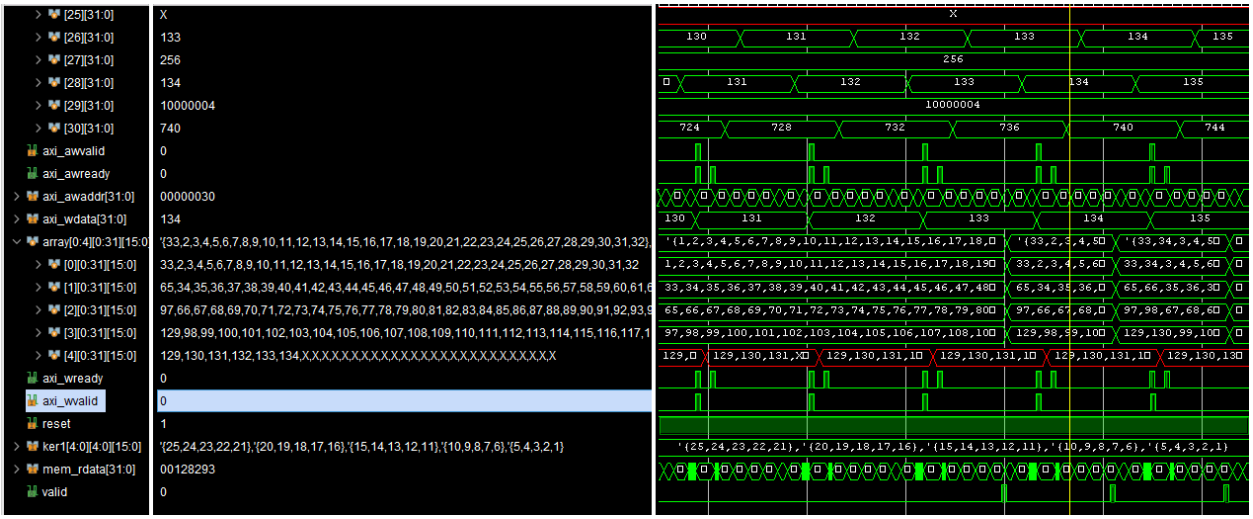




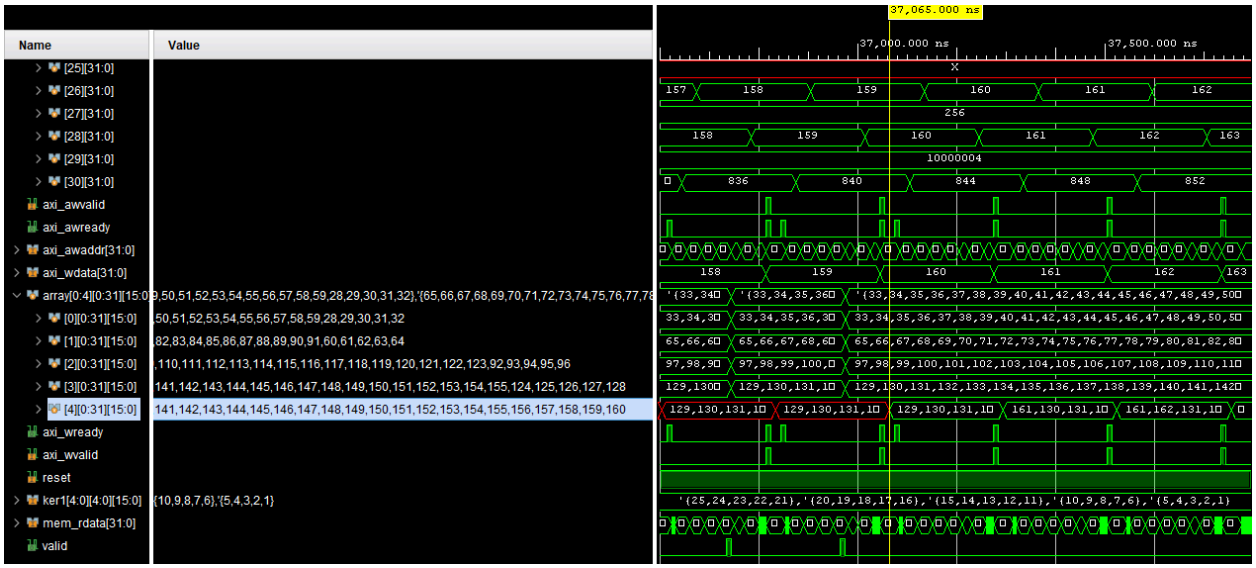
Convolution starts:



Here, it is observed that the convolution operation is faster than the data rate. Specific measures are taken, like holding the convolution operation(the horizontal movement of filters is stopped till the new data arrives).



For the next input data. The vertical movement of the used data also started appropriately.



But after filling the buffer for the first time. The convolution operation is terminating. But the new data is refilling the buffer.

Next Tasks:

1. Debug the Convolution operation in the Integrated design.
2. Implement the model by transmitting the data of 4 pixels per transaction(8x4 = 32 bits). (i.e., For utilizing the entire bandwidth).
3. Implement Lenet-5 in Python using the existing libraries and figure out what happens to the feature maps after pooling layer1 before passing through convolution layer2.
4. Extend the design with the Max/Avg pooling layer accordingly.
5. Make the changes to the Verilog code according to the above observations(Selecting the size of data for convolution module inputs and outputs).
6. Implement the protocols for the AXI-4 lite slave convolution interface supporting both write and read operations.
7. Test the integrated design with original image data.
8. Train the model implemented in Python using the MNIST data set and extract the weights in 8-bit data.
9. Infer and validate the Integrated design using the above weights.
10. Extract the synthesis results and further optimize.

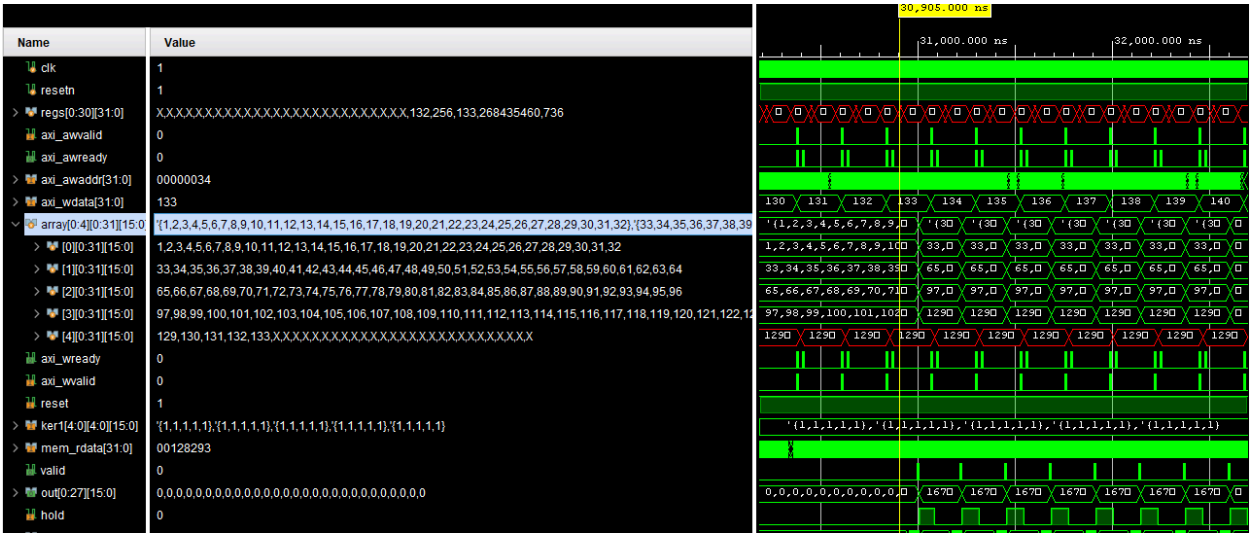
Task 1:

Convolution after connecting to the AXI-4 lite interface is Debugged.

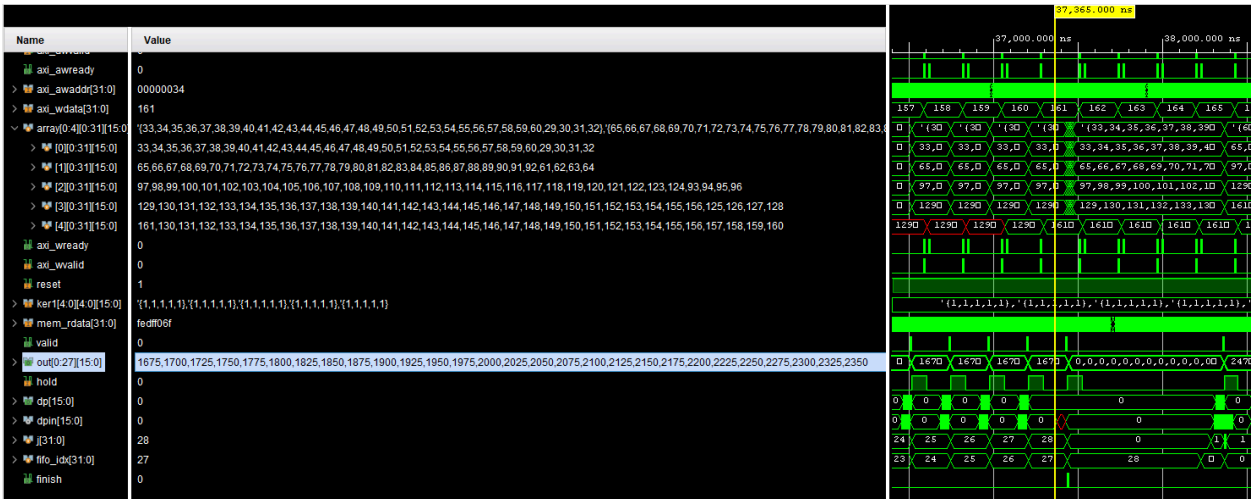
- The maximum input data rate supported by the convolution is 1-pixel data per clock cycle.
- This convolution design supports inputs with low input data rates such as 1-pixel data per transaction (4 instructions per transaction).
- The convolution is held in between the using a control signal named hold if the data for further convolution is not yet available because of low input data rates.

The convolution module is tested with a test image of size 8x32 with filled 1-256 numbers respectively. The kernel is a 5x5 matrix with each of its cells filled with 1. The size of the output image is 4x28 and it is given below.

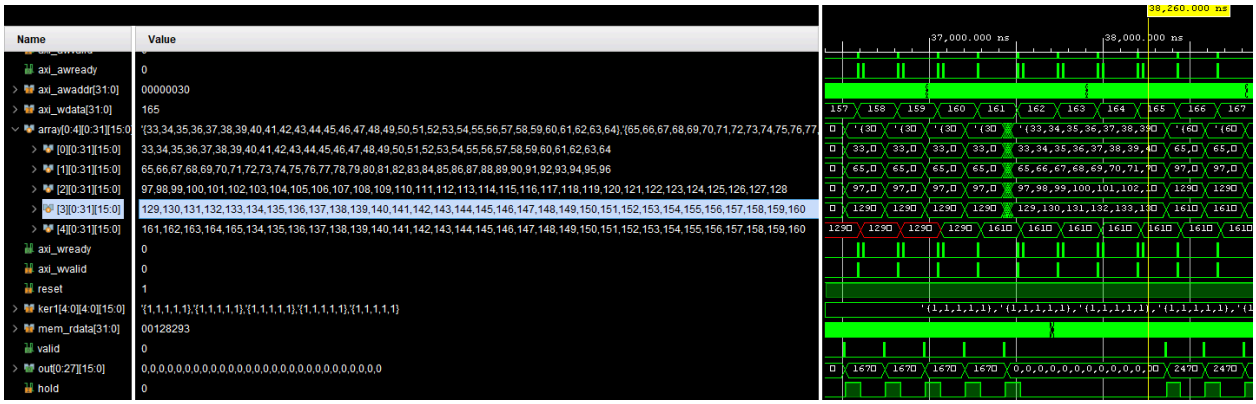
- The start of the convolution.



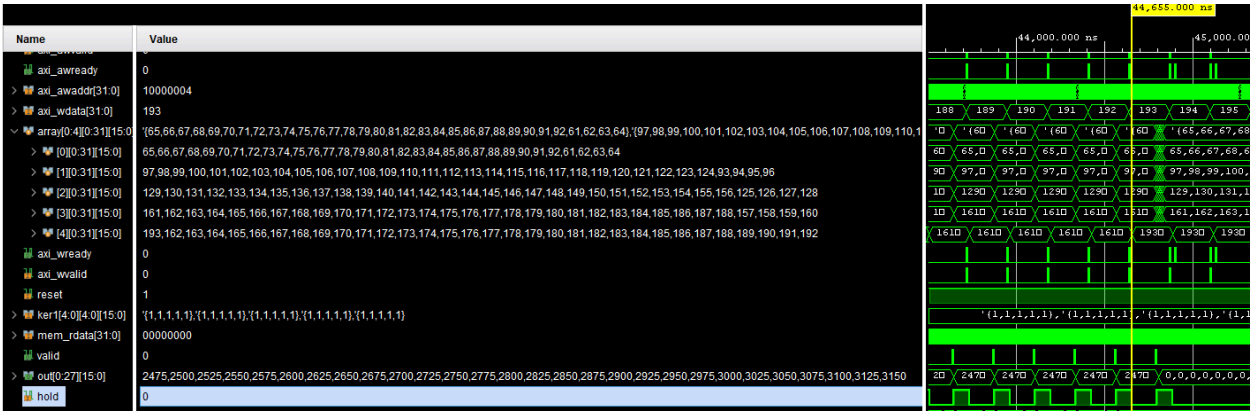
- **The first row of the output of the convolution operation.**



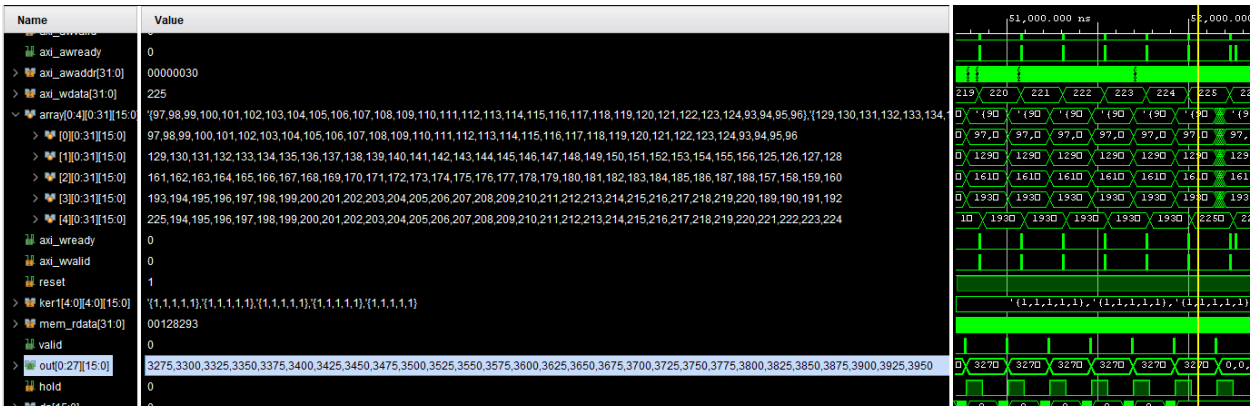
- The convolution is stopped until the data for the convolution of the next row of outputs is available. (This is not done using the hold signal).
- The convolution for the second row of outputs starts after filling the last row of the buffer with 165.



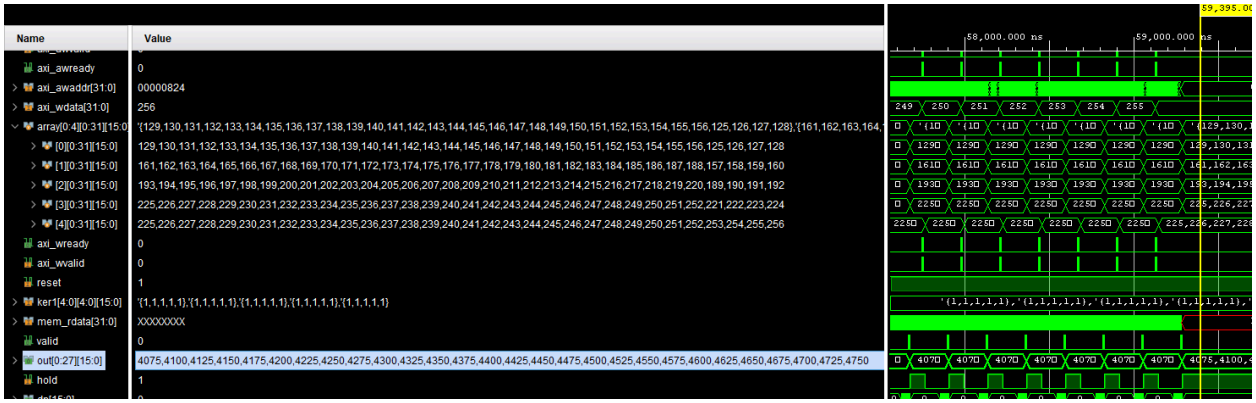
- **The second row of the outputs.**



- A similar process continues until the end of convolution.
- **The third row of the outputs.**



- **The fourth row of the outputs.**



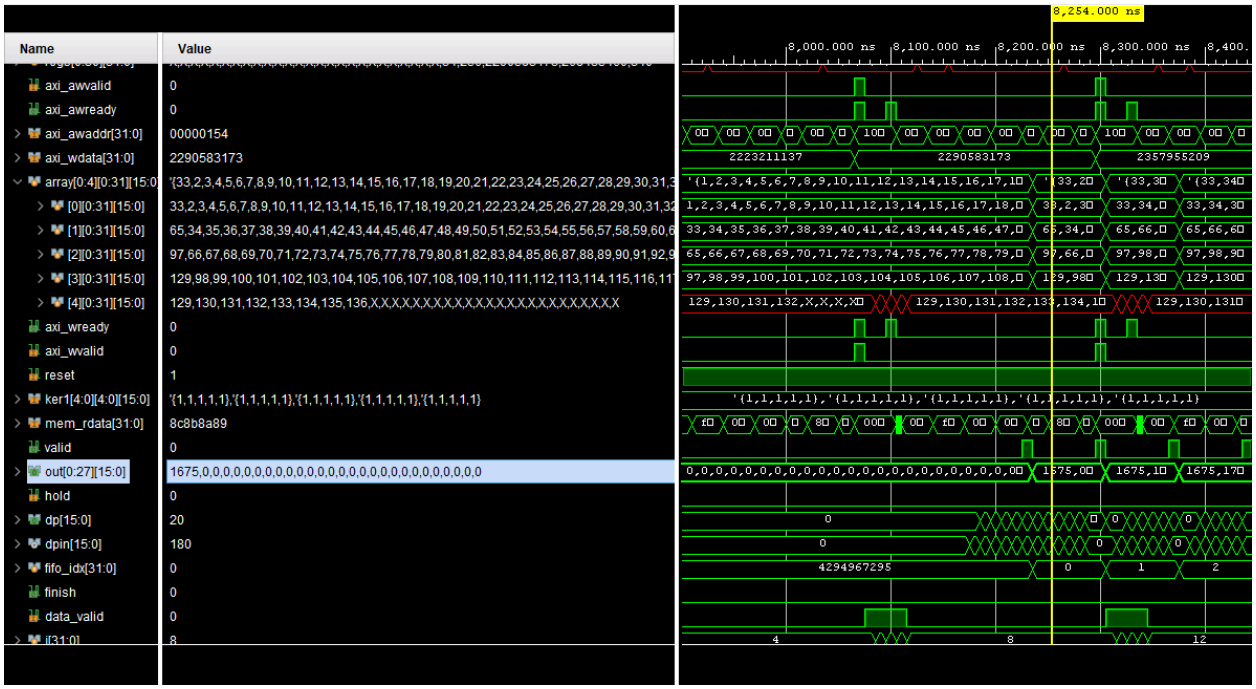
Task 2:

We transmit 4 pixels of data per transaction by exploiting the 32-bit width of the axi_write signal.

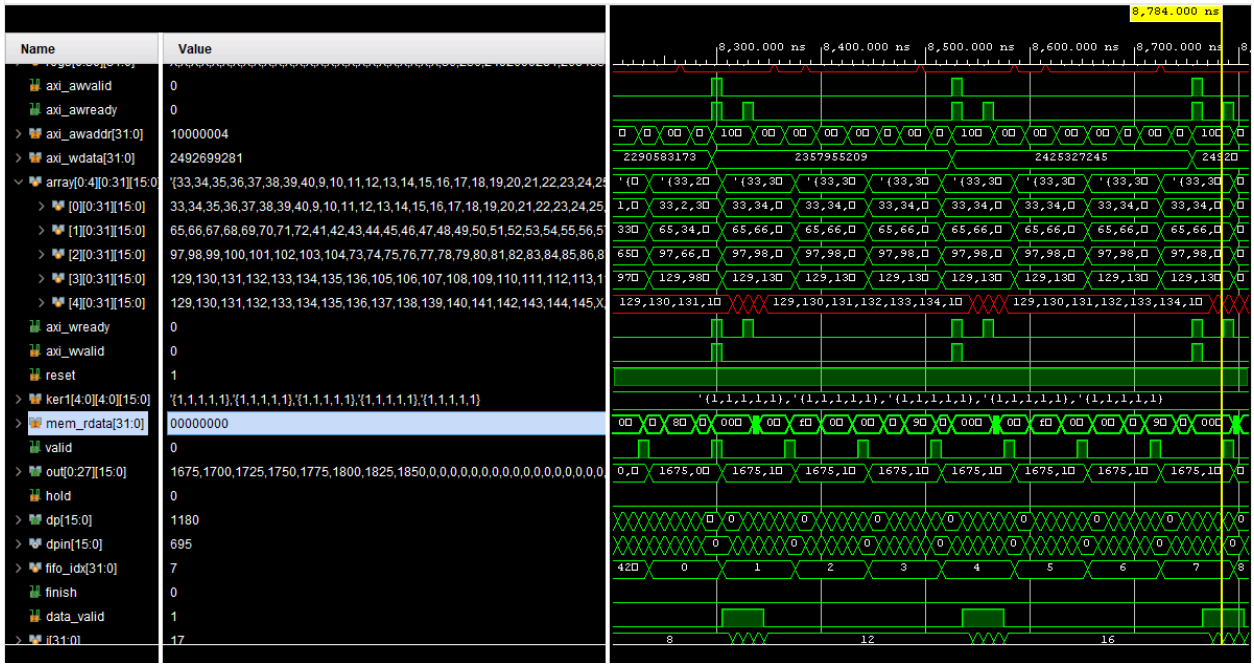
The convolution module is tested with a test image of size 32x32 with filled 1-256 numbers respectively. The kernel is a 5x5 matrix with each of its cells filled with 1. The size of the output image is 28x28 and it is given below.

1. **For every transaction, 4 pixels of data are transmitted and they are filled in the buffer in the next four cycles.**

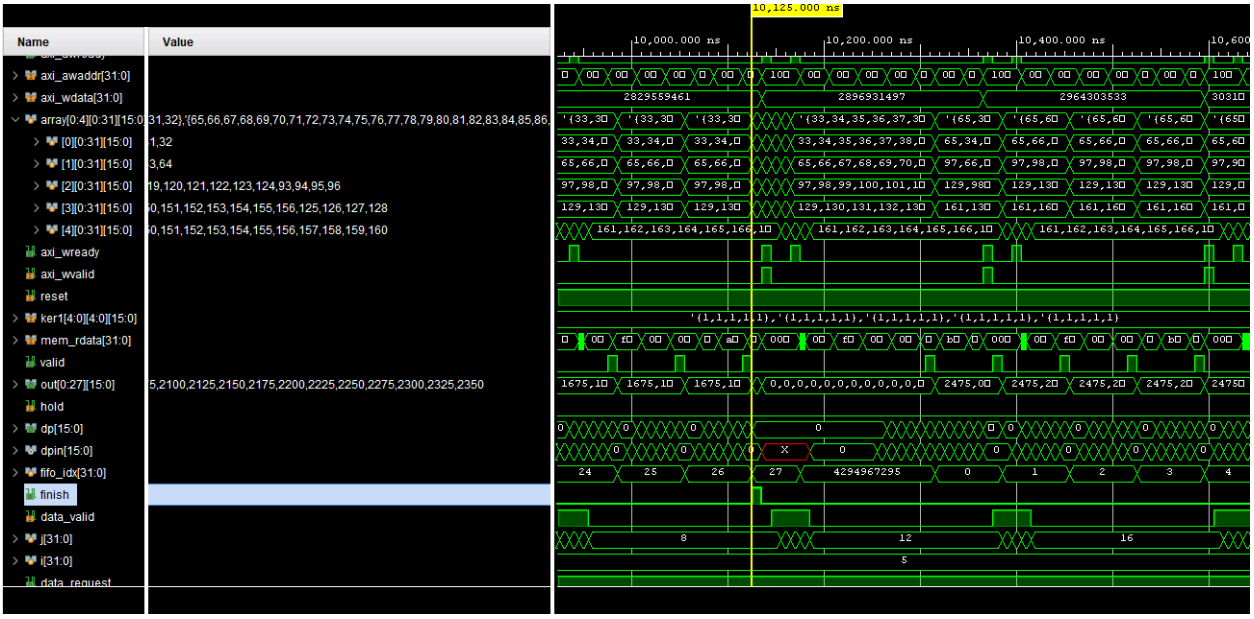
2. **The convolution starts after the 5 pixels of the last row of the conv buffer are filled.**



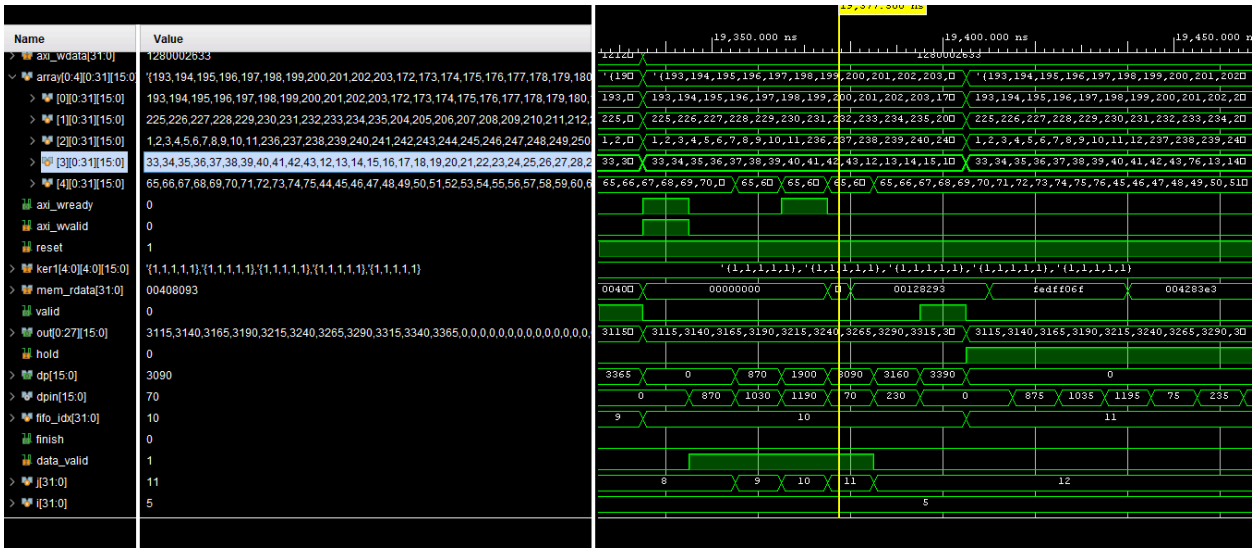
3. **Multiple convolution operations take place in a single transaction.**



4. **The first row of outputs.**



5. J is crossing Fifo_idx (input data rate is higher but still data_request is not off)



Task 3:

The observations made indicate that the 6 feature maps after the 1st pooling layer are independently used in the 2nd convolution layer. So they cannot be reduced and no changes can be made to the feature maps generated after the pooling layer.

https://medium.com/codex/lenet-5-complete-architecture-84c6d08215f9

C3 layer-convolutional layer:

- **Input:** all 6 or several feature map combinations in S2
- **Convolution kernel size:** 5 * 5
- **Convolution kernel type:** 16
- **Output featureMap size:** 10 * 10 (14–5 + 1) = 10
- Each feature map in C3 is connected to all 6 or several feature maps in S2, indicating that the feature map of this layer is a different combination of the feature maps extracted from the previous layer.
- One way is that the first 6 feature maps of C3 take 3 adjacent feature map subsets in S2 as input. The next 6 feature maps take 4 subsets of neighboring feature maps in S2 as input. The next three take the non-adjacent 4 feature map subsets as input. The last one takes all the feature maps in S2 as input.
- **The trainable parameters are:** 6 * (3 * 5 * 5 + 1) + 6 * (4 * 5 * 5 + 1) + 3 * (4 * 5 * 5 + 1) + 1 * (6 * 5 * 5 +1) = 1516
- **Number of connections:** 10 * 10 * 1516 = 151600

- The first 6 feature maps of C3 (corresponding to the 6th column of the first red box in the figure above) are connected to the 3 feature maps connected to the S2 layer (the first red box in the above figure), and the next 6 feature maps are connected to the S2 layer The 4 feature maps are connected (the second red box in the figure above), the next 3 feature maps are connected with the 4 feature maps that are not connected at the S2 layer, and the last is connected with all the feature maps at the S2 layer. The convolution kernel size is still 5 * 5, so there are 6 * (3 * 5 * 5 + 1) + 6 * (4 * 5 * 5 + 1) + 3 * (4 * 5 * 5 + 1) +1 * (6 * 5 * 5 + 1) = 1516 parameters. The image size is 10 * 10, so there are 151600 connections.

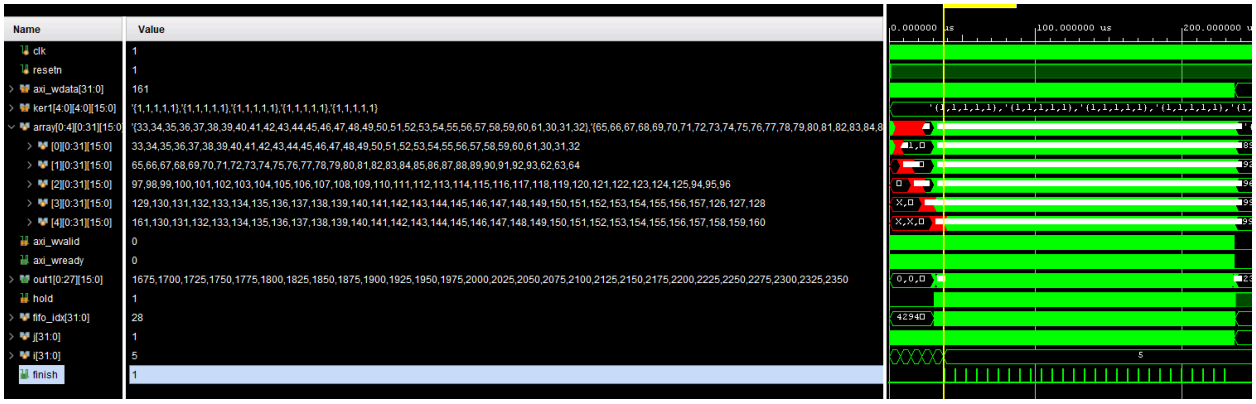
Task 7: Testing the implementation for 1 pixel per transaction for 32x32 input.

The memory assembly code is written for an image size of 8X32(256 pixels). Now it is extended to a 32x32(1024 pixels) image by the following changes in the memory assembly code.

The input image is a 32x32 matrix filled with 1-1024 numbers respectively. The kernel/filter is a 5x5 matrix filled with all 1's. The resultant output matrix is of size 28x28 matrix.

```
initial begin
    memory[0] = 32'h0c800093;
    memory[1] = 32'h0000a103;
    memory[2] = 32'h00000193;
    memory[3] = 32'h00312023;
    memory[4] = 32'h00410113;
//    memory[5] = 32'h10000213; // li x4, 256
    memory[5] = 32'h40000213;   // li x4, 1024
    memory[6] = 32'h00000293;
    memory[7] = 32'h0cc00093;
    memory[8] = 32'h004283e3;
    memory[9] = 32'h0000a183;
    memory[10] = 32'h00312023;
    memory[11] = 32'h00408093;
    memory[12] = 32'h00128293;
    memory[13] = 32'hfedff06f;
    memory[14] = 0;
    memory[15] = 0;

    memory[50] = 32'h10000000; // Reset address of conv slave.

//    $readmemb("seven.txt",A);
//    for(itt = 1; itt <= 1024; itt = itt + 1) begin
//        memory[50+itt] = A[itt-1];
//    end
    for(itt = 1; itt <= 1024; itt = itt + 1) begin
        memory[50+itt] = itt;
    end
```

The first row of outputs:

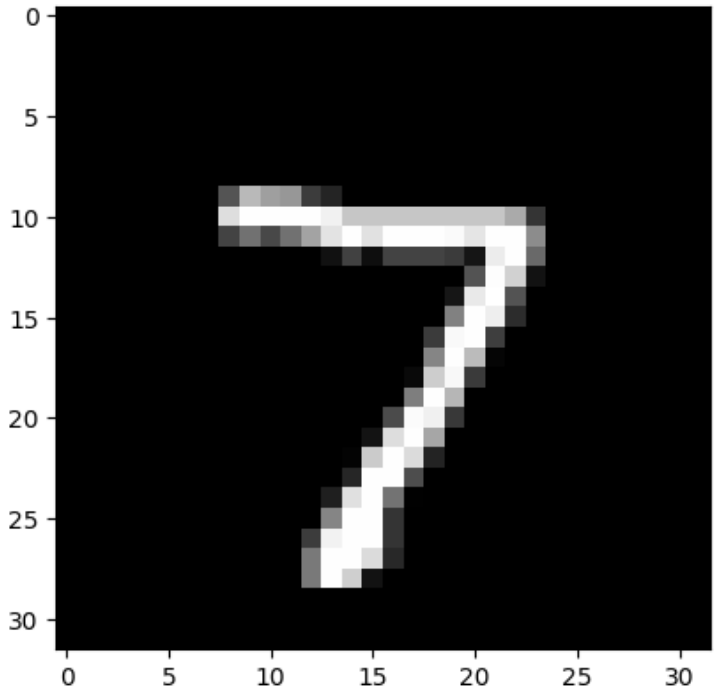


The last row of outputs:

All the rows of outputs are checked manually, and the resulting output matches the required values.

Testing the implementation for 1 pixel per transaction for 32x32 input with a test image.

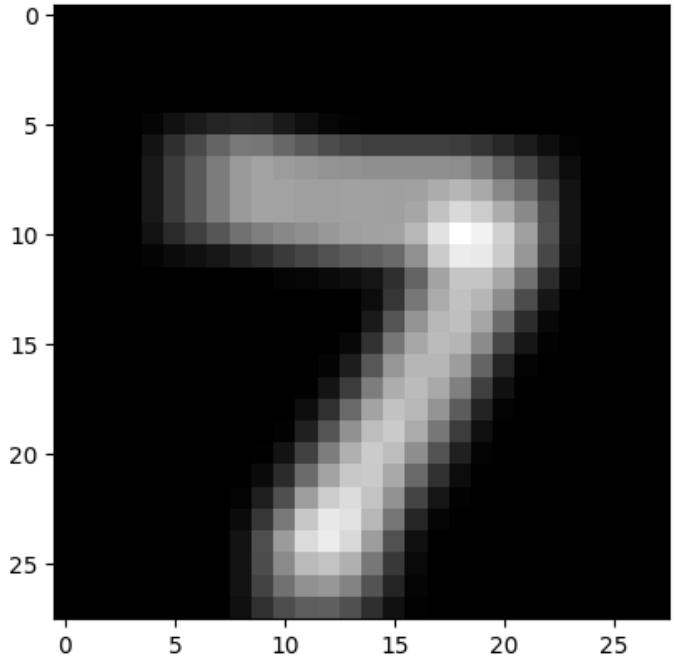
Input image: Mnist data set 7



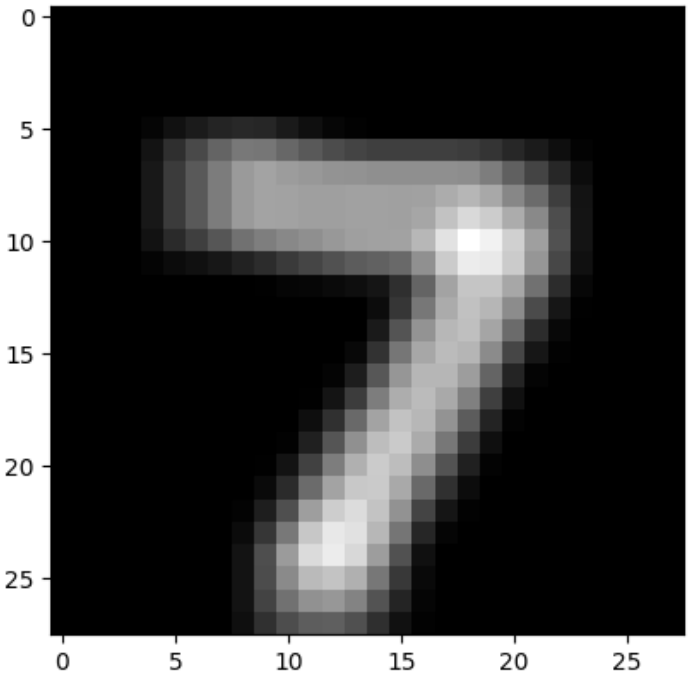
The image is extracted from the `mnist_test.csv` available in the Google Colab. The actual size of the image data is 28x28 but Lenet-5 takes an input of size 32x32. So we pad zeros to the original data to make it 32x32. (we won't interpolate it).

Kernel - 5x5 avg filter

Expected output:



Output extracted from Vivado:

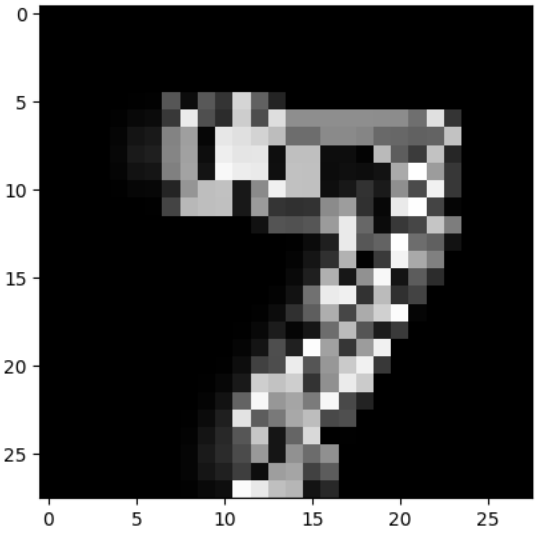


The similarity of the images is calculated using the ssim function.
For the above test case, the ssim value is 1.0.

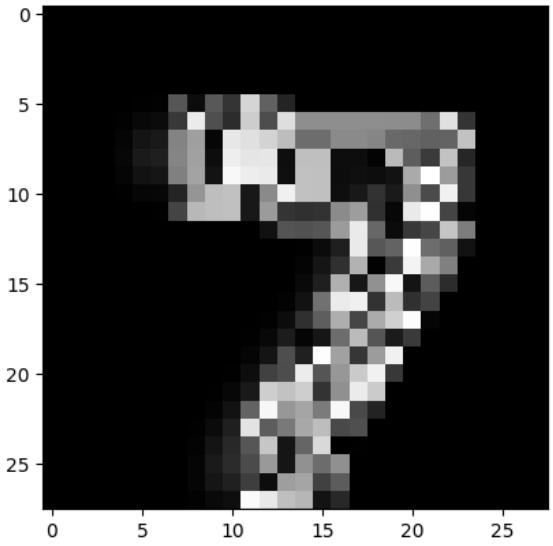
Kernel - Sobel left:

```
[-1, -2, 0, 2, 1]
[-4, -8, 0, 8, 4]
[-6, -12, 0, 12, 6]
[-4, -8, 0, 8, 4]
[-1, -2, 0, 2, 1]
```

Expected output:

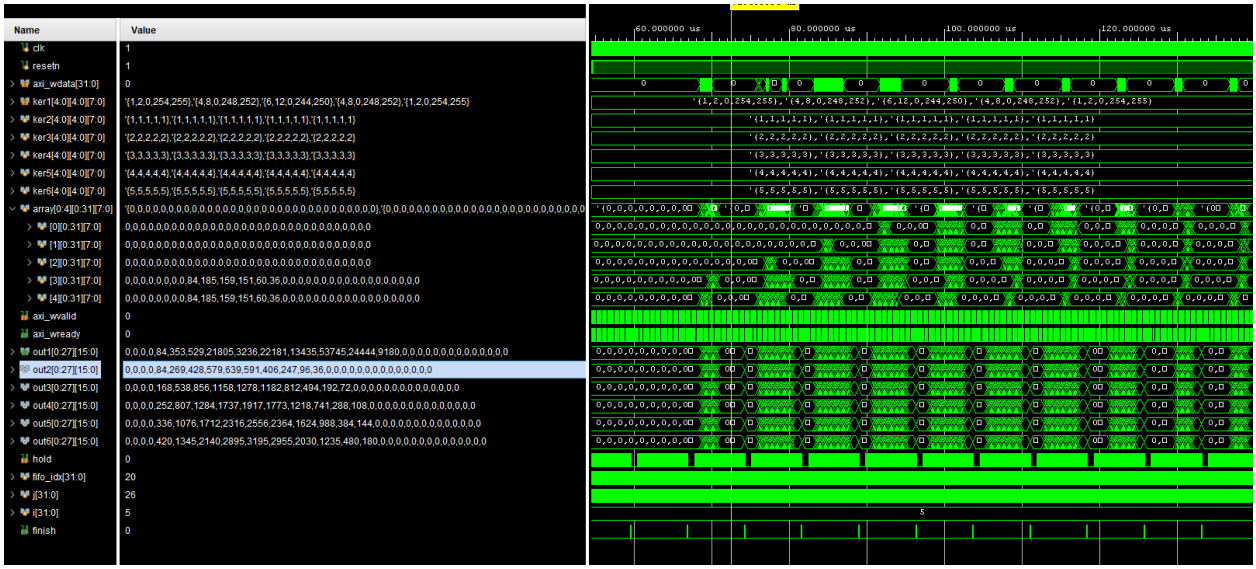


Output extracted from Vivado:



The similarity of the images was calculated using the ssim function. For the above test case, the ssim value is 1.0.

Results generated for the first Conv layer of Lenet-5.



Synthesis results for the Integrated design with memory:

Timing results:

There is a path from the positive edge of the clock cycle to the negative edge of the clock cycle between picorv32 and memory which is causing the half-cycle period to be 7.558ns. Therefore, the clock period is 7.558*2 ns = 15.116 ns

=> The frequency of operation = 1/clock period = 1/15.116ns = 66.15MHz.

Name	Slack ^1	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination C
Path 1	-2.558	7	8	141	top/slavedecod...waddr_reg__0/C	top/slavedecode..._reg[109][0]/CE	7.176	1.498	5.678	5.0	clk	clk
Path 2	-2.558	7	8	141	top/slavedecod...waddr_reg__0/C	top/slavedecode...eg[109][10]/CE	7.176	1.498	5.678	5.0	clk	clk
Path 3	-2.558	7	8	141	top/slavedecod...waddr_reg__0/C	top/slavedecode...eg[109][11]/CE	7.176	1.498	5.678	5.0	clk	clk
Path 4	-2.558	7	8	141	top/slavedecod...waddr_reg__0/C	top/slavedecode...eg[109][12]/CE	7.176	1.498	5.678	5.0	clk	clk
Path 5	-2.558	7	8	141	top/slavedecod...waddr_reg__0/C	top/slavedecode...eg[109][13]/CE	7.176	1.498	5.678	5.0	clk	clk
Path 6	-2.558	7	8	141	top/slavedecod...waddr_reg__0/C	top/slavedecode...eg[109][14]/CE	7.176	1.498	5.678	5.0	clk	clk
Path 7	-2.558	7	8	141	top/slavedecod...waddr_reg__0/C	top/slavedecode...eg[109][15]/CE	7.176	1.498	5.678	5.0	clk	clk
Path 8	-2.558	7	8	141	top/slavedecod...waddr_reg__0/C	top/slavedecode...eg[109][16]/CE	7.176	1.498	5.678	5.0	clk	clk
Path 9	-2.558	7	8	141	top/slavedecod...waddr_reg__0/C	top/slavedecode...eg[109][17]/CE	7.176	1.498	5.678	5.0	clk	clk
Path 10	-2.558	7	8	141	top/slavedecod...waddr_reg__0/C	top/slavedecode...eg[109][18]/CE	7.176	1.498	5.678	5.0	clk	clk

Utilization:

The following image shos the no of Luts consumed by the integrated design with memory.

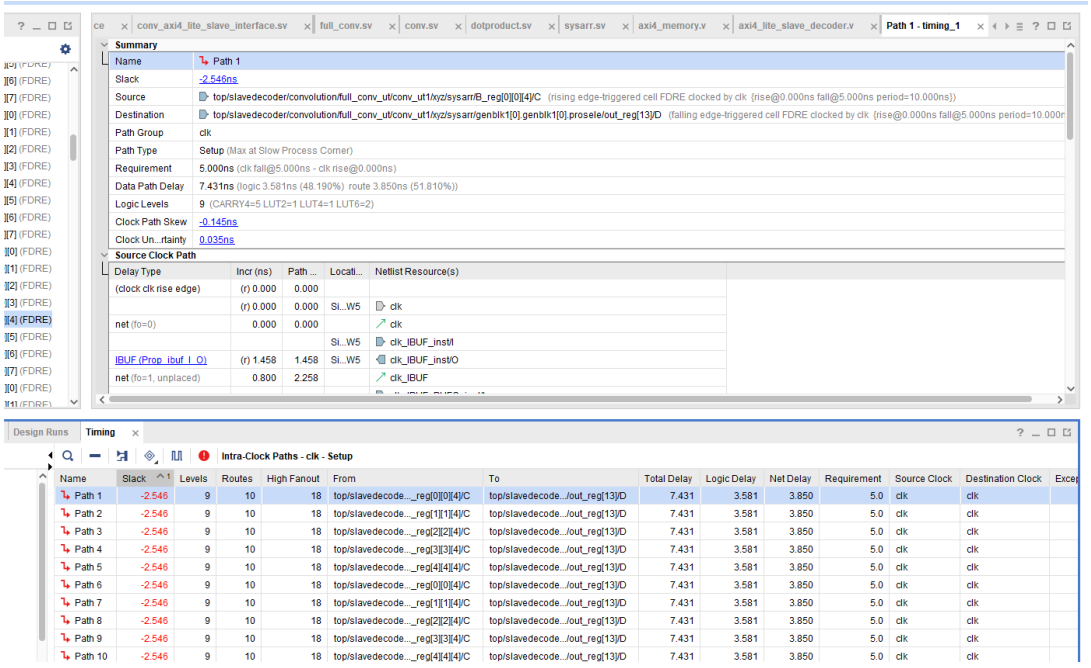
Name ^1	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Bonded IOB (106)	BUFGCTRL (32)
N picorv32_axi_top	81992	147225	19172	8842	34	2
top (picorv32_wrapper)	81992	147225	19172	8842	0	1
slavedecoder (axi4_lite	56955	146657	19172	8842	0	0
convolution (conv_a	21894	15223	1764	138	0	0
full_conv_ut (full_	21864	13966	1764	138	0	0
conv_ut1 (con	4778	1674	1190	63	0	0
xyz (dotpdt_	3550	1067	1120	48	0	0
conv_ut2 (con	2194	1662	38	15	0	0
xyz (dotpdt_	1210	1057	0	0	0	0
conv_ut3 (con	2194	1662	38	15	0	0
xyz (dotpdt_	1210	1057	0	0	0	0
conv_ut4 (con	2194	1662	38	15	0	0
conv_ut5 (con	2194	1662	38	15	0	0
conv_ut6 (con	2194	1662	38	15	0	0
mem (axi4_memor	35012	131397	17408	8704	0	0
uut (picorv32_axi)	25037	568	0	0	0	0
axi_adapter (picorv3	3	4	0	0	0	0
picorv32_core (pico	25034	564	0	0	0	0
cpuregs (picorv3	162	0	0	0	0	0

Synthesis results for the Integrated design without memory:

Timing results:

There is a path from the positive edge of the clock cycle to the negative edge of the clock cycle in the systolic array which is causing the half-cycle period to be 7.546ns. Therefore, the clock period is 7.546*2 ns = 15.092 ns

=> The frequency of operation = 1/clock period = 1/15.092ns = 66.26MHz.(we have to change this path from posedge to negegde to posedge to posedge).



Utilization:

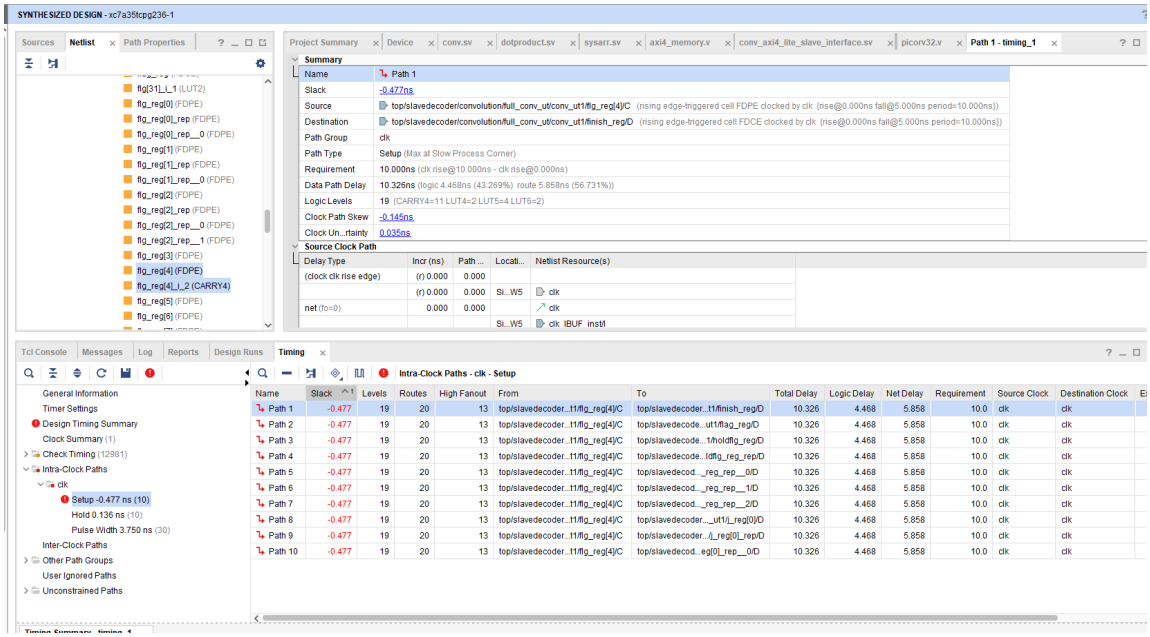
Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Bonded IOB (106)	BUFGCTRL (32)
picorv32_axi_top	24207	19705	1764	138	34	2
top (picorv32_wrapper)	24207	19705	1764	138	0	1
slavedecoder (axi4_lite	23215	19133	1764	138	0	0
convolution (conv_a	23168	19096	1764	138	0	0
full_conv_ut (full_	23138	17844	1764	138	0	0
conv_ut1 (con	2203	1671	38	15	0	0
conv_ut2 (con	2203	1671	38	15	0	0
conv_ut3 (con	2203	1671	38	15	0	0
conv_ut4 (con	2203	1671	38	15	0	0
conv_ut5 (con	2203	1671	38	15	0	0
conv_ut6 (con	2203	1671	38	15	0	0
mem (axi4_memor	0	0	0	0	0	0
uut (picorv32_axi)	992	572	0	0	0	0
axi_adapter (picorv	2	4	0	0	0	0
picorv32_core (pico	990	568	0	0	0	0
cpuregs (picorv3	147	0	0	0	0	0

Synthesis results for the Integrated design without memory(1st optimization):

Timing results:

After changing the path (from posedge to negedge -> posedge to posedge) the clock period = 10.477 ns

=> The frequency of operation = 1/clock period = 1/10.477ns = 95.447MHz. (30.57% increase)

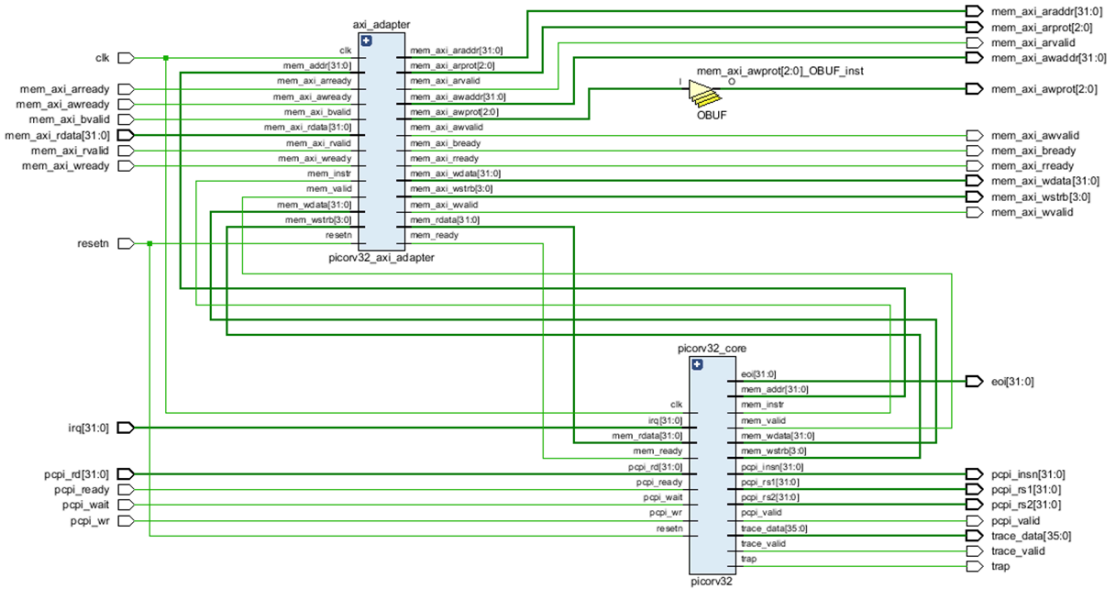


Utilization:

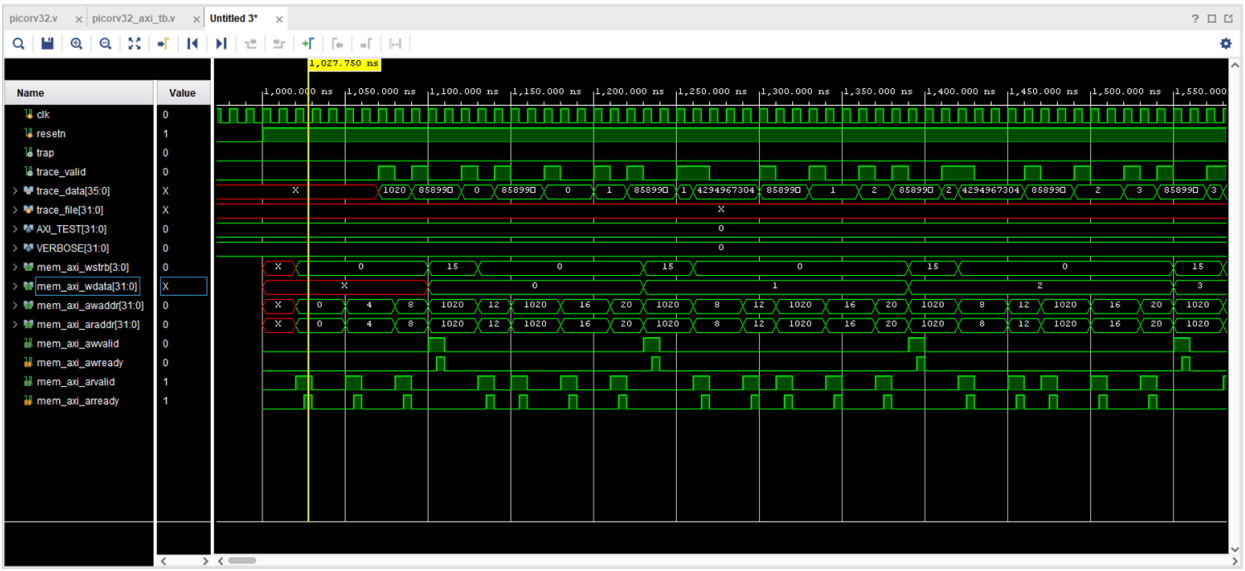
Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Bonded IOB (106)	BUFGCTRL (32)
picorv32_axi_top	22723	17661	1920	192	34	2
top (picorv32_wrapper)	22723	17661	1920	192	0	1
slavedecoder (axi4_lite	21731	17089	1920	192	0	0
convolution (conv_a	21684	17052	1920	192	0	0
full_conv_ut (full_	21654	15800	1920	192	0	0
conv_ut1 (cor	4515	1338	1216	72	0	0
xyz (dotpdt_	3382	729	1120	48	0	0
sysarr (	422	592	0	0	0	0
conv_ut2 (cor	1951	1328	64	24	0	0
conv_ut3 (cor	1951	1328	64	24	0	0
conv_ut4 (cor	1951	1328	64	24	0	0
conv_ut5 (cor	1951	1328	64	24	0	0
conv_ut6 (cor	1951	1328	64	24	0	0
mem (axi4_memor	0	0	0	0	0	0
uut (picorv32_axi)	992	572	0	0	0	0
axi_adapter (picorv	2	4	0	0	0	0
picorv32_core (pico	990	568	0	0	0	0

PICORV32_AXI:

The picorv32 core is connected to an axi_adapter module which provides an axi4-lite master interface.



Waveform for the Assembly code used in Picorv32 with AXi interface.



Synthesis results of PICORV32 with AXI interface.

Timing results:

The worst negative slack is 4.957 ns for the 10 ns clock. The clock period for this design is 10ns - 4.957ns = 5.403 ns.

=> The frequency of operation = 1/clock period = 185.08MHz

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	4.957 ns	Worst Hold Slack (WHS):	0.132 ns	Worst Pulse Width Slack (WPWS):	3.750 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	1646	Total Number of Endpoints:	1646	Total Number of Endpoints:	665

All user specified timing constraints are met.

Utilization:

Name	Slice LUTs (53200)	Slice Registers (106400)	Bonded IOB (200)	BUFGCTRL (32)
picorv32_axi	907	568	183	1
axi_adapter (picorv32_axi_adapter)	3	4	0	0
picorv32_core (picorv32)	904	564	0	0
cpuregs (picorv32_regs)	176	0	0	0

The number of clock cycles consumed for an input image of size NxN for kernel size of nxn is equal to NxN(4 instructions of picorv32) + (2*n + 2){for last pixel convolution}

= N*N (1 load instruction, 1 store instruction, 2 add immediate instructions) + (2*n + 2){for last pixel convolution}

For N = 32 and n = 5

No of clock cycles = 32*32*(1 load + 1 store + 2 addi) + 2*5 + 2
= 32*32*(1 load + 1 store + 2 addi) + 12
= 32*32*(16) + 12
= 1024*16 + 12
= 16396

Time period for the first convolution layer of LENET-5 to execute = no of clock cycles * clock period
= 16396 * 10.477 ns
= 171780.892 ns(for Basys-3 FPGA)
= 0.172 ms

D. Rongshi and T. Yongming, "Accelerator Implementation of Lenet-5 Convolution Neural Network Based on FPGA with HLS," 2019 3rd International Conference on Circuits, System and Simulation (ICCSS), Nanjing, China, 2019, pp. 64-67, doi: 10.1109/CIRSYSSIM.2019.8935599.

In the above paper for the same Lenet-5 CNN, the consumed for his implementation for the CNN layer1 is 2.72ms (ZYBO Z7 board).

Our implementation is 15.81 times faster than the above implementation.

The advantage of the integrated design:

- The convolution design is capable of taking the data without any rearrangement, decreasing the burden on the picorv32.
- The integrated design also supports multiple slaves.

The disadvantage of the integrated design:

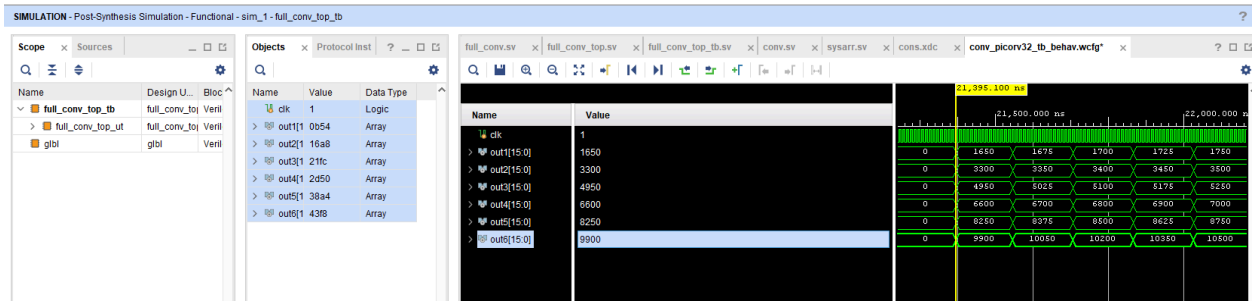
- The frequency of operation dropped from 185.08 ns to 95.447 ns. I.e. There is a 48.43% decrease in the frequency.

Future works:

- **Nullifying the disadvantage:** Building the axi4-lite slave interface so that the conv design and the picorv32 run on two different clocks.
- Extending the design for IEEE-754 single-precision floating point numbers, and half-precision floating point numbers.
- Extending the entire design to the multiple layers of Lenet-5 CNN.

Synthesis results with 6 conv in layer 1:

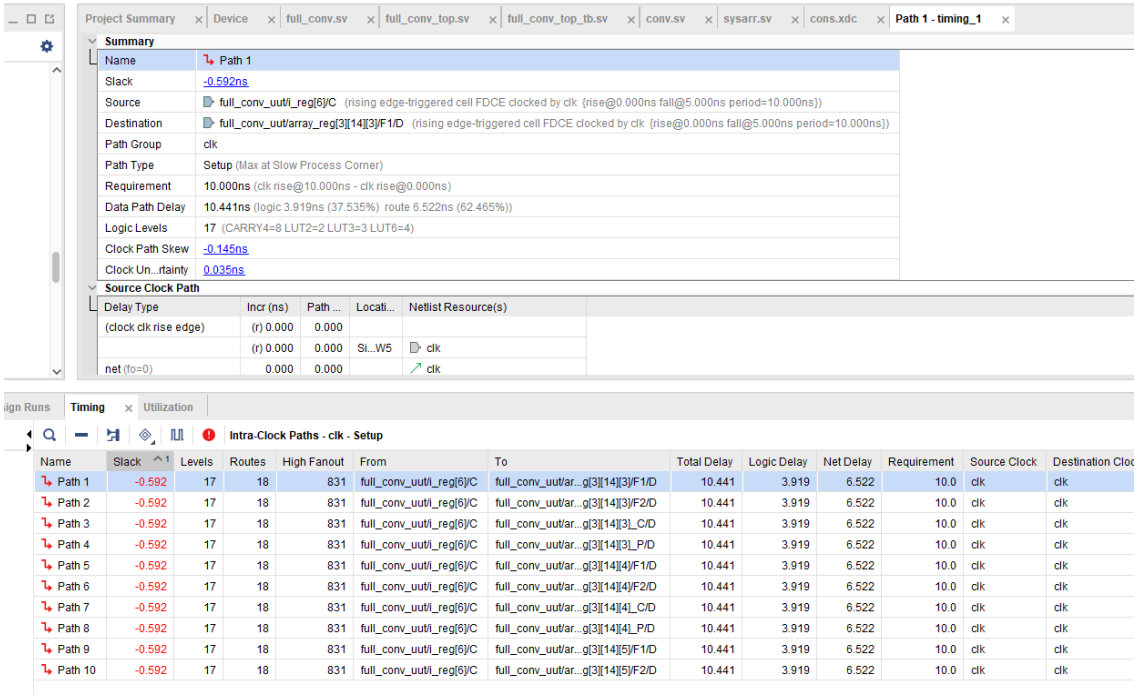
For the image data starting from 0 to 255. The kernels ker1, ker2, .. and ker6 are filled with all 1's to all 6's respectively. The corresponding outputs are from out1 to out6.



Utilization:

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Bonded IOB (106)	BUFGCTRL (32)
full_conv_top	18490	12713	1373	150	97	1
full_conv_uut (full_conv)	18477	12681	1373	150	0	0
conv_ut1 (conv)	1411	811	64	24	0	0
conv_ut2 (conv_0)	1410	810	64	24	0	0
conv_ut3 (conv_1)	1409	810	64	24	0	0
conv_ut4 (conv_2)	1411	811	64	24	0	0
conv_ut5 (conv_3)	1411	811	64	24	0	0
conv_ut6 (conv_4)	4040	811	909	30	0	0

Timing for 10ns clk:



The design is working on 94.41 MHz.

Testing on Basys-3 FPGA board:

Let's test the convolution Layer design on the Basys-3 FPGA board using ILA.
The ILA(Integrated Logic Analyser) takes all the 6 outputs out1, out2, .. and out6 as inputs.

Synthesis results with ILA.
Utilization

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
full_conv_top	19945	14878	1376	150	3	1	1
conv_ila (ila_1)	972	1684	3	0	3	0	0
dbg_hub (dbg_hub_CV)	0	0	0	0	0	0	0
full_conv_uut (full_conv)	18969	13171	1373	150	0	0	0
conv_ut1 (conv)	1496	893	64	24	0	0	0
conv_ut2 (conv_0)	1485	888	64	24	0	0	0
conv_ut3 (conv_1)	1489	891	64	24	0	0	0
conv_ut4 (conv_2)	1494	893	64	24	0	0	0
conv_ut5 (conv_3)	1494	893	64	24	0	0	0
conv_ut6 (conv_4)	4127	893	909	30	0	0	0

Timing results with ILA for a 10ns clock.

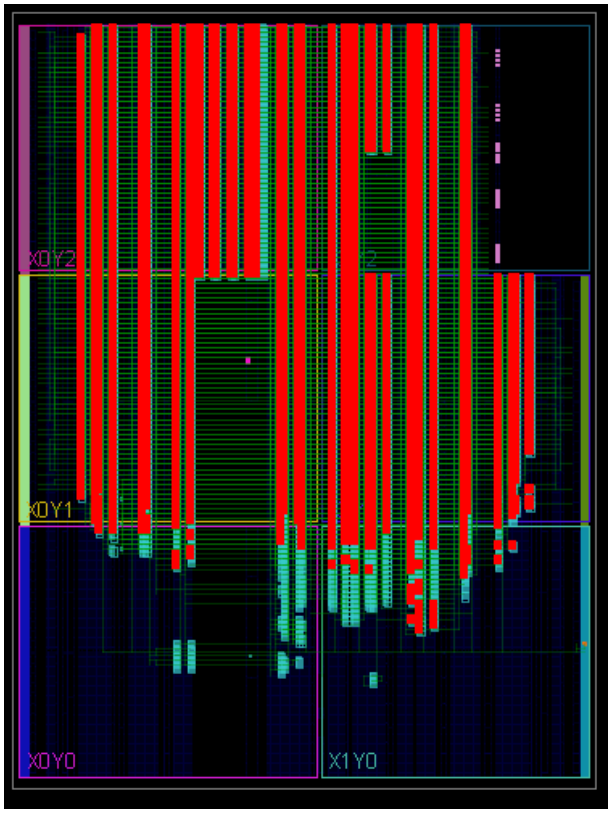
Runs | Timing | Utilization

Intra-Clock Paths - clk - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destin
Path 1	-0.665	17	18	4	full_conv_uut/conv_ut3/flg_reg[4]/C	full_conv_uut/conv_ut3/j_reg[0]/D	10.514	4.248	6.266	10.0	clk	clk
Path 2	-0.665	17	18	4	full_conv_uut/conv_ut3/flg_reg[4]/C	full_conv_uut/co.../j_reg[0]_rep/D	10.514	4.248	6.266	10.0	clk	clk
Path 3	-0.665	17	18	4	full_conv_uut/conv_ut3/flg_reg[4]/C	full_conv_uut/c...reg[0]_rep__0/D	10.514	4.248	6.266	10.0	clk	clk
Path 4	-0.593	17	18	831	full_conv_uut/i_reg[6]/C	full_conv_uut/ar...g[3][14][0]/F1/D	10.442	3.919	6.523	10.0	clk	clk
Path 5	-0.593	17	18	831	full_conv_uut/i_reg[6]/C	full_conv_uut/ar...g[3][14][0]/F2/D	10.442	3.919	6.523	10.0	clk	clk
Path 6	-0.593	17	18	831	full_conv_uut/i_reg[6]/C	full_conv_uut/ar...g[3][14][0]_C/D	10.442	3.919	6.523	10.0	clk	clk
Path 7	-0.593	17	18	831	full_conv_uut/i_reg[6]/C	full_conv_uut/ar...g[3][14][0]_P/D	10.442	3.919	6.523	10.0	clk	clk
Path 8	-0.593	17	18	831	full_conv_uut/i_reg[6]/C	full_conv_uut/ar...g[3][14][1]/F1/D	10.442	3.919	6.523	10.0	clk	clk
Path 9	-0.593	17	18	831	full_conv_uut/i_reg[6]/C	full_conv_uut/ar...g[3][14][1]/F2/D	10.442	3.919	6.523	10.0	clk	clk
Path 10	-0.593	17	18	831	full_conv_uut/i_reg[6]/C	full_conv_uut/ar...g[3][14][1]_C/D	10.442	3.919	6.523	10.0	clk	clk

The period of a clock is 10.665 ns. The frequency of operation is 93.76MHz.

Implementation results of the Design with ILA.



Implementation Timing Results with ILA for a 10ns clock

Project Summary | Device | full_conv.sv | full_conv_top.sv | full_conv_top_tb.sv | cons.xdc | Path 1 - full_conv_top_timing_summary_routed

Summary

Name	Path 1
Slack	-2.997ns
Source	full_conv_uut/j_reg[21]/C (rising edge-triggered cell FDCE clocked by clk {rise@0.000ns fall@5.000ns period=10.000ns})
Destination	full_conv_uut/array_reg[3][19][6]_P/D (rising edge-triggered cell FDPE clocked by clk {rise@0.000ns fall@5.000ns period=10.000ns})
Path Group	clk
Path Type	Setup (Max at Slow Process Corner)
Requirement	10.000ns (clk rise@10.000ns - clk rise@0.000ns)
Data Path Delay	12.618ns (logic 4.142ns (32.826%) route 8.476ns (67.174%))
Logic Levels	15 (CARRY4=8 LUT2=2 LUT3=2 LUT6=3)
Clock Path Skew	-0.299ns
Clock Un...rtainty	0.035ns

Source Clock Path

Delay Type	Incr (ns)	Path ...	Location	Netlist Resource(s)
(clock clk rise edge)	(r) 0.000	0.000		
	(r) 0.000	0.000	Site: W5	clk
net (fo=0)	0.000	0.000		clk

DRC | Methodology | Power | Timing

Intra-Clock Paths - clk - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Excep
Path 1	-2.997	15	850	full_conv_uut/j_reg[21]/C	full_conv_uut/ar...g[3][19][6]_P/D	12.618	4.142	8.476	10.0	clk	clk	
Path 2	-2.983	15	850	full_conv_uut/j_reg[21]/C	full_conv_uut/ar...g[3][17][2]_P/D	12.576	4.142	8.434	10.0	clk	clk	
Path 3	-2.983	15	850	full_conv_uut/j_reg[21]/C	full_conv_uut/ar...g[3][19][6]/F2/D	12.618	4.142	8.476	10.0	clk	clk	
Path 4	-2.949	15	850	full_conv_uut/j_reg[21]/C	full_conv_uut/ar...g[3][19][6]_C/D	12.527	4.142	8.385	10.0	clk	clk	

The period after implementation is found to be 12.997 ns. The frequency of operation of the design is 76.94 MHz.

Implementation Utilization for the design with ILA.

Hierarchy

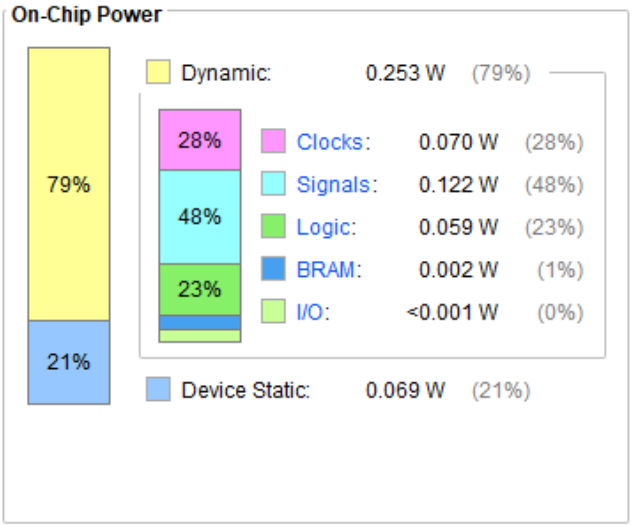
Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Slice (8150)	LUT as Logic (20800)	LUT as Memory (9600)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)	BSCANE2 (4)
full_conv_top	18549	13771	1376	150	5920	18368	181	3	1	2	1
conv_ila (ila_1)	895	1638	3	0	461	738	157	3	0	0	0
dbg_hub (dbg_hub)	443	727	0	0	229	419	24	0	0	1	1
full_conv_uut (full_conv)	17207	11360	1373	150	5365	17207	0	0	0	0	0
conv_ut1 (conv)	1406	583	64	24	739	1406	0	0	0	0	0
conv_ut2 (conv_0)	1399	578	64	24	763	1399	0	0	0	0	0
conv_ut3 (conv_1)	1404	593	64	24	773	1404	0	0	0	0	0
conv_ut4 (conv_2)	1406	583	64	24	745	1406	0	0	0	0	0
conv_ut5 (conv_3)	1408	593	64	24	714	1408	0	0	0	0	0
conv_ut6 (conv_4)	4244	593	909	30	2318	4244	0	0	0	0	0

Implementation Power results:

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

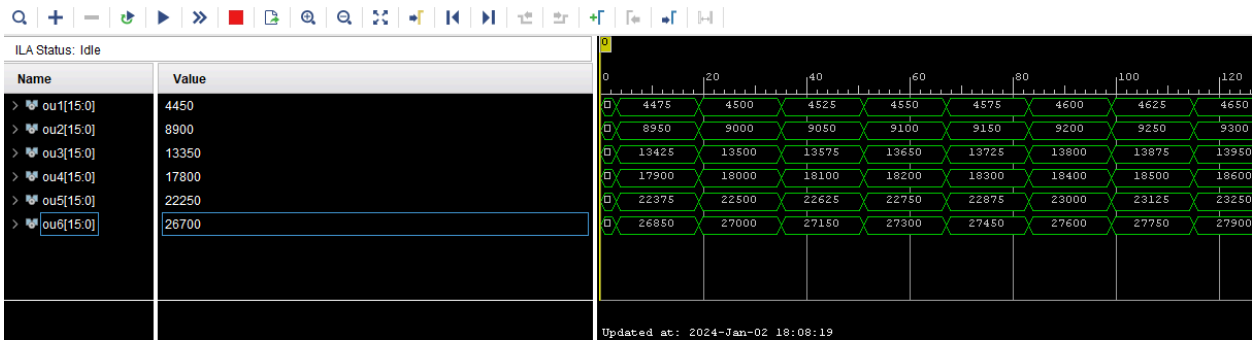
Total On-Chip Power:	**0.322 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**26.6°C**
Thermal Margin:	58.4°C (11.6 W)
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

Launch Power Constraint Advisor to find and fix invalid switching activity

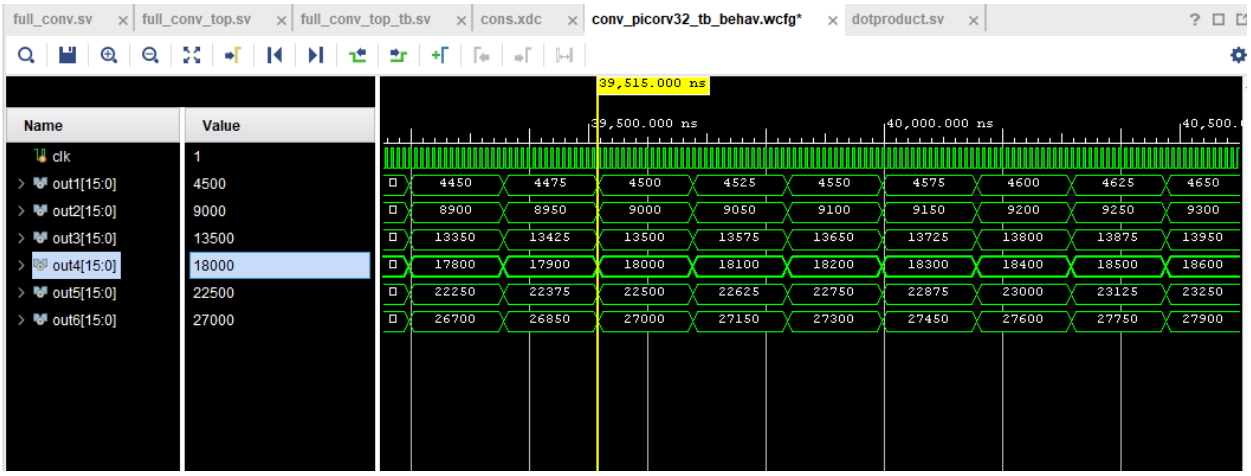


Results on ILA.

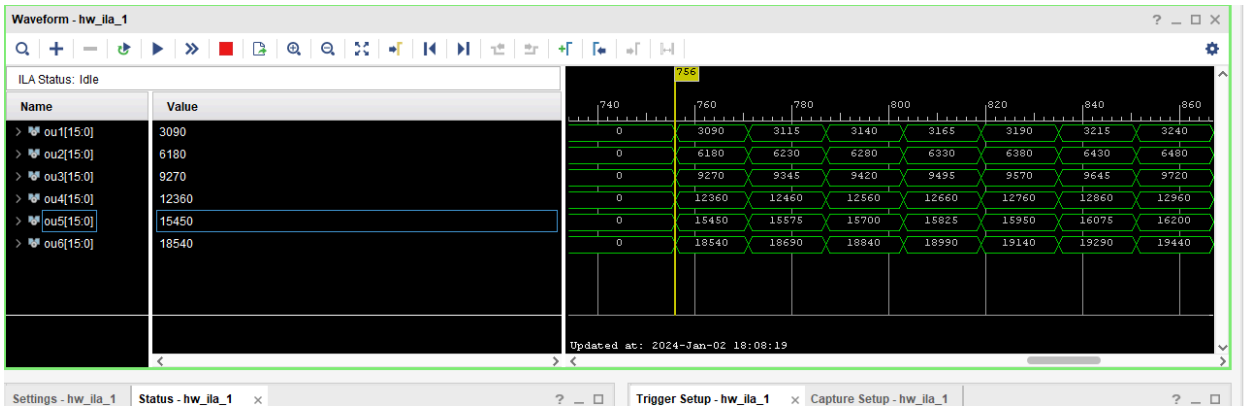
ILA extracts the results from the board on runtime. One of the extracted parts is given below.



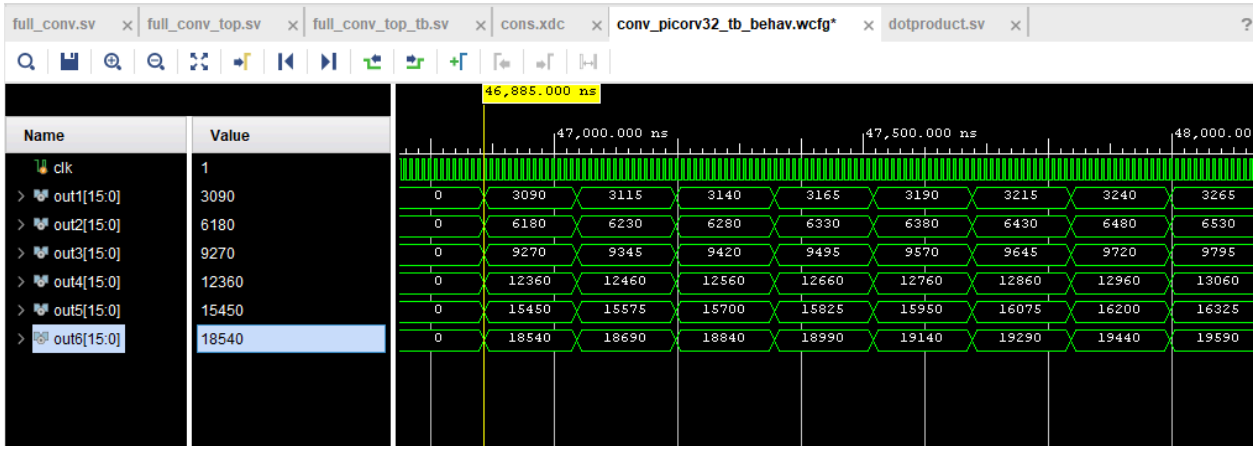
The above-extracted part of ILA can be matched with the Behavioural simulation part.



ILA Extract 2:



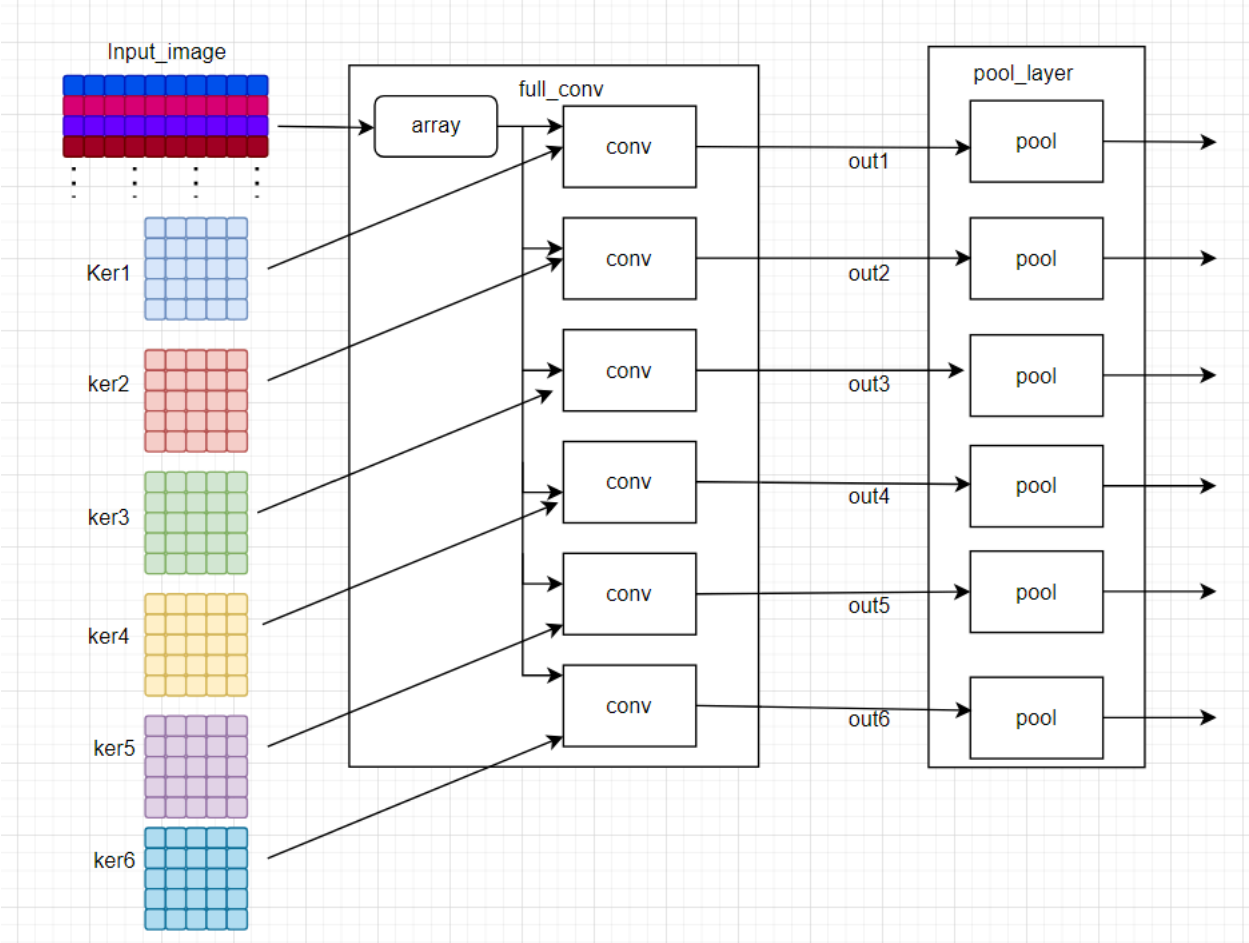
Behavioral simulation matching the above extract.



Pooling Layer:

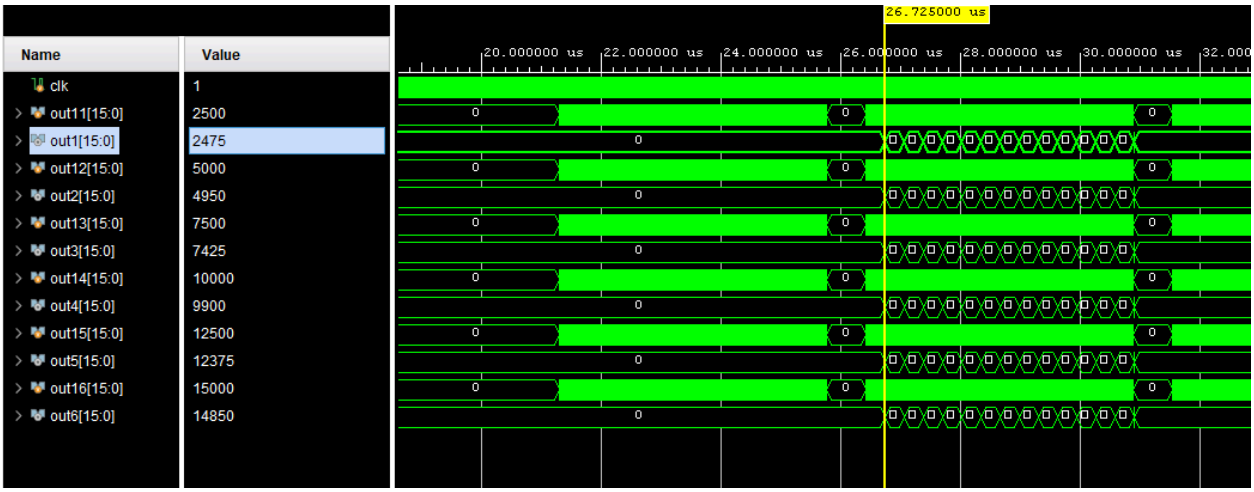
The pooling layer takes the input from the Convolutional Layer. The pooling is parametrized for various stride values. As this is an independent block that operates on outputs of the convolutional layer it will not add any overhead to the frequency of operation(lesser complex than the convolutional layer).

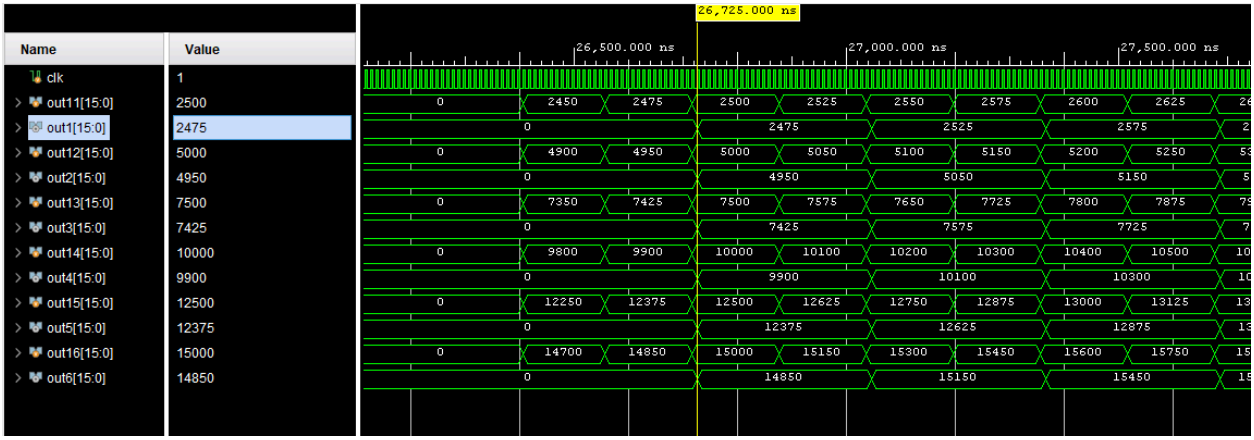
The number of inputs to the pooling layer is not parametrized. Max pooling is implemented. This design can be easily extendible to average pooling.



With stride 2:

Out11, out12, out13,.. Out16 are the outputs of the convolutional layer. Ou1,ou2,..ou6 are the outputs of the pooling layer. We can observe that the pooling layer gives outputs for every two rows of the feature maps generated.





The Pooling layer is parametrized for various values of strides.

Synthesis results for conv_layer + pool_layer design:

Utilization:

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Bonded IOB (106)	BUFGCTRL (32)
N full_conv_top	22266	13534	1352	144	97	1
full_conv_uut (full_conv)	20712	12009	1328	144	0	0
conv_ut1 (conv)	3606	831	64	24	0	0
conv_ut2 (conv_5)	1409	810	64	24	0	0
conv_ut3 (conv_6)	1410	810	64	24	0	0
conv_ut4 (conv_7)	1405	811	64	24	0	0
conv_ut5 (conv_8)	1406	811	64	24	0	0
conv_ut6 (conv_9)	4571	811	896	24	0	0
pool1 (pool_layer)	1543	1494	24	0	0	0
pool1 (pool)	258	249	4	0	0	0
pool2 (pool_0)	257	249	4	0	0	0
pool3 (pool_1)	257	249	4	0	0	0
pool4 (pool_2)	257	249	4	0	0	0
pool5 (pool_3)	257	249	4	0	0	0
pool6 (pool_4)	257	249	4	0	0	0

Timing results for a 10 ns clock:

uns | Timing | Utilization

Intra-Clock Paths - clk - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Cloc
Path 1	-2.845	20	1439	full_conv_uut/i_reg[6]/C	full_conv_uut/ar...g[3][20][0]/F1/D	12.694	4.297	8.397	10.000	clk	clk
Path 2	-2.845	20	1439	full_conv_uut/i_reg[6]/C	full_conv_uut/ar...g[3][20][0]/F2/D	12.694	4.297	8.397	10.000	clk	clk
Path 3	-2.845	20	1439	full_conv_uut/i_reg[6]/C	full_conv_uut/ar...g[3][20][0]_C/D	12.694	4.297	8.397	10.000	clk	clk
Path 4	-2.845	20	1439	full_conv_uut/i_reg[6]/C	full_conv_uut/ar...g[3][20][0]_P/D	12.694	4.297	8.397	10.000	clk	clk
Path 5	-2.830	20	1439	full_conv_uut/i_reg[6]/C	full_conv_uut/ar...reg[3][4][1]_C/D	12.679	4.297	8.382	10.000	clk	clk
Path 6	-2.830	20	1439	full_conv_uut/i_reg[6]/C	full_conv_uut/ar...reg[3][4][1]_P/D	12.679	4.297	8.382	10.000	clk	clk
Path 7	-2.830	20	1439	full_conv_uut/i_reg[6]/C	full_conv_uut/ar...reg[3][4][2]_C/D	12.679	4.297	8.382	10.000	clk	clk
Path 8	-2.830	20	1439	full_conv_uut/i_reg[6]/C	full_conv_uut/ar...reg[3][4][2]_P/D	12.679	4.297	8.382	10.000	clk	clk
Path 9	-2.830	20	1439	full_conv_uut/i_reg[6]/C	full_conv_uut/ar...reg[3][4][3]_C/D	12.679	4.297	8.382	10.000	clk	clk
Path 10	-2.830	20	1439	full_conv_uut/i_reg[6]/C	full_conv_uut/ar...reg[3][4][3]_P/D	12.679	4.297	8.382	10.000	clk	clk

The time_period is 12.845 ns. The frequency of operation is 77.85 MHz.

Optimized version of convolution Layer

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Bonded IOB (106)	BUFGCTRL (32)
full_conv_top	19831	10047	554	147	97	1
full_conv_uut (full_conv)	18284	8530	530	147	0	0
conv_ut1 (conv)	2833	789	64	24	0	0
conv_ut2 (conv_5)	1510	769	64	24	0	0
conv_ut3 (conv_6)	1509	769	64	24	0	0
conv_ut4 (conv_7)	1509	769	64	24	0	0
conv_ut5 (conv_8)	1509	769	64	24	0	0
conv_ut6 (conv_9)	4061	769	99	27	0	0
pool1 (pool_layer)	1543	1494	24	0	0	0
pool1 (pool)	258	249	4	0	0	0
pool2 (pool_0)	257	249	4	0	0	0
pool3 (pool_1)	257	249	4	0	0	0
pool4 (pool_2)	257	249	4	0	0	0
pool5 (pool_3)	257	249	4	0	0	0
pool6 (pool_4)	257	249	4	0	0	0

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Ex
Path 1	-0.430	11	9	full_conv_uut/conv_ut2/flg_reg[4]/C	full_conv_uut/co...t2/holdflg_reg/D	10.279	3.625	6.654	10.000	clk	clk	
Path 2	-0.430	11	9	full_conv_uut/conv_ut3/flg_reg[4]/C	full_conv_uut/co...t3/holdflg_reg/D	10.279	3.625	6.654	10.000	clk	clk	
Path 3	-0.430	11	9	full_conv_uut/conv_ut4/flg_reg[4]/C	full_conv_uut/co...t4/holdflg_reg/D	10.279	3.625	6.654	10.000	clk	clk	
Path 4	-0.430	11	9	full_conv_uut/conv_ut5/flg_reg[4]/C	full_conv_uut/co...t5/holdflg_reg/D	10.279	3.625	6.654	10.000	clk	clk	
Path 5	-0.430	11	9	full_conv_uut/conv_ut6/flg_reg[4]/C	full_conv_uut/co...t6/holdflg_reg/D	10.279	3.625	6.654	10.000	clk	clk	
Path 6	-0.091	13	33	full_conv_uut/conv_ut1/flg_reg[4]/C	full_conv_uut/co..._ut1/j_reg[13]/D	9.940	3.685	6.255	10.000	clk	clk	
Path 7	-0.091	13	33	full_conv_uut/conv_ut1/flg_reg[4]/C	full_conv_uut/co..._ut1/j_reg[14]/D	9.940	3.685	6.255	10.000	clk	clk	
Path 8	-0.091	13	33	full_conv_uut/conv_ut1/flg_reg[4]/C	full_conv_uut/co..._ut1/j_reg[15]/D	9.940	3.685	6.255	10.000	clk	clk	
Path 9	0.143	10	9	full_conv_uut/conv_ut2/flg_reg[4]/C	full_conv_uut/co...t2/finish_reg/D	9.706	3.501	6.205	10.000	clk	clk	
Path 10	0.143	10	9	full_conv_uut/conv_ut2/flg_reg[4]/C	full_conv_uut/conv_ut2/flag_reg/D	9.706	3.501	6.205	10.000	clk	clk	

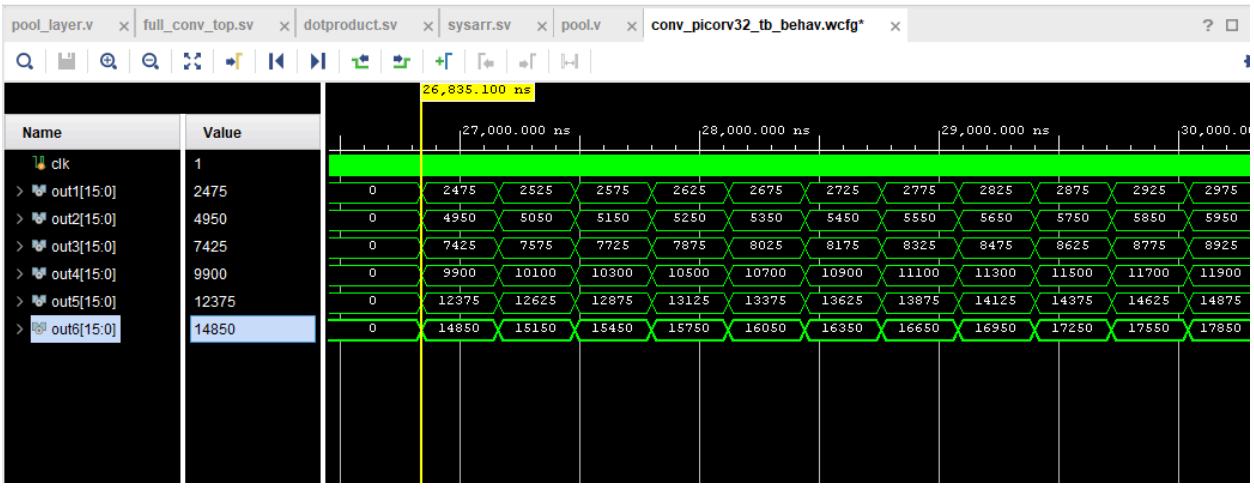
Freq of operation = 1/10.430ns = 95.877 MHz.

Synthesis results for stride = 3

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destinati
Path 1	-0.775	12	142	full_conv_uut/co...id_idx_reg[2]/C	pool1/pool1/out_reg[8]/D	10.624	2.476	8.148	10.000	clk	clk
Path 2	-0.770	12	142	full_conv_uut/co...id_idx_reg[2]/C	pool1/pool1/out_reg[0]/D	10.619	2.476	8.143	10.000	clk	clk
Path 3	-0.770	12	142	full_conv_uut/co...id_idx_reg[2]/C	pool1/pool1/out_reg[13]/D	10.619	2.476	8.143	10.000	clk	clk
Path 4	-0.770	12	142	full_conv_uut/co...id_idx_reg[2]/C	pool1/pool1/out_reg[14]/D	10.619	2.476	8.143	10.000	clk	clk
Path 5	-0.770	12	142	full_conv_uut/co...id_idx_reg[2]/C	pool1/pool1/out_reg[15]/D	10.619	2.476	8.143	10.000	clk	clk
Path 6	-0.770	12	142	full_conv_uut/co...id_idx_reg[2]/C	pool1/pool1/out_reg[1]/D	10.619	2.476	8.143	10.000	clk	clk
Path 7	-0.770	12	142	full_conv_uut/co...id_idx_reg[2]/C	pool1/pool1/out_reg[2]/D	10.619	2.476	8.143	10.000	clk	clk
Path 8	-0.770	12	142	full_conv_uut/co...id_idx_reg[2]/C	pool1/pool1/out_reg[3]/D	10.619	2.476	8.143	10.000	clk	clk
Path 9	-0.770	12	142	full_conv_uut/co...id_idx_reg[2]/C	pool1/pool1/out_reg[5]/D	10.619	2.476	8.143	10.000	clk	clk
Path 10	-0.770	12	142	full_conv_uut/co...id_idx_reg[2]/C	pool1/pool1/out_reg[6]/D	10.619	2.476	8.143	10.000	clk	clk

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Bonded IOB (106)	BUFGCTRL (32)
full_conv_top	19637	10056	690	227	97	1
full_conv_uut (full_conv)	18610	8519	530	147	0	0
conv_ut1 (conv)	3799	777	64	24	0	0
conv_ut2 (conv_5)	1392	769	64	24	0	0
conv_ut3 (conv_6)	1391	769	64	24	0	0
conv_ut4 (conv_7)	1391	769	64	24	0	0
conv_ut5 (conv_8)	1391	769	64	24	0	0
conv_ut6 (conv_9)	3885	769	99	27	0	0
pool1 (pool_layer)	1062	1514	160	80	0	0
pool1 (pool)	195	249	0	0	0	0
pool2 (pool_0)	175	253	32	16	0	0
pool3 (pool_1)	173	253	32	16	0	0
pool4 (pool_2)	173	253	32	16	0	0
pool5 (pool_3)	173	253	32	16	0	0
pool6 (pool_4)	173	253	32	16	0	0

Post- Synthesis functional simulation results for stride 2 for pooling



The results match with the behavioral simulation results

Testing on FPGA board using ILA:

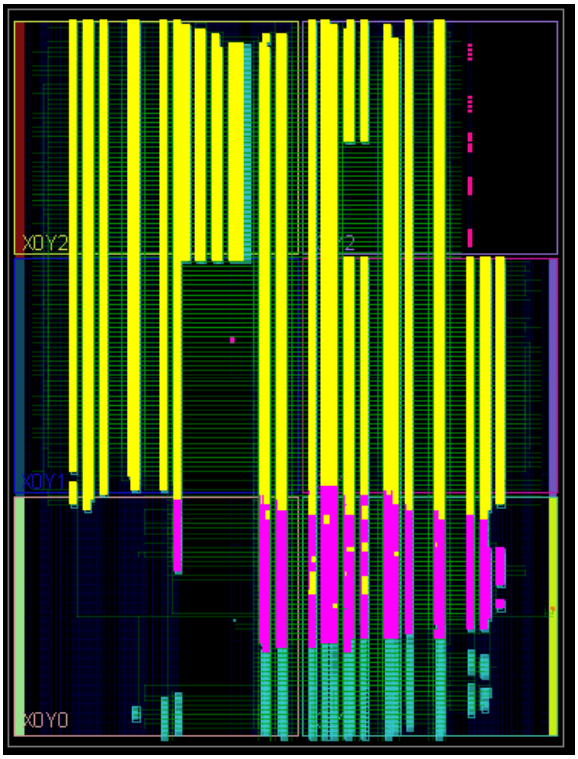
The conv_Layer + pool_layer design is integrated with ILA for testing on Basys-3 FPGA board.

Synthesis results with ILA

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
full_conv_top	20054	11719	568	146	3	1	1
conv_ila (ila_1)	972	1684	3	0	3	0	0
dbg_hub (dbg_hub_CV)	0	0	0	0	0	0	0
full_conv_uut (full_conv)	15633	8518	535	146	0	0	0
conv_ut1 (conv)	1375	777	64	24	0	0	0
conv_ut2 (conv_0)	1344	769	64	24	0	0	0
conv_ut3 (conv_1)	1343	769	64	24	0	0	0
conv_ut4 (conv_2)	1343	769	64	24	0	0	0
conv_ut5 (conv_3)	1343	769	64	24	0	0	0
conv_ut6 (conv_4)	3729	769	114	26	0	0	0
pool1 (pool_layer)	3445	1494	30	0	0	0	0
pool1 (pool)	574	249	5	0	0	0	0
pool2 (pool_163)	574	249	5	0	0	0	0
pool3 (pool_164)	574	249	5	0	0	0	0
pool4 (pool_165)	574	249	5	0	0	0	0
pool5 (pool_166)	574	249	5	0	0	0	0
pool6 (pool_167)	575	249	5	0	0	0	0

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Cl
Path 1	-0.430	11	9	full_conv_uut/conv_ut2/flg_reg[4]/C	full_conv_uut/co...t2/holdflg_reg/D	10.279	3.625	6.654	10.000	clk	clk
Path 2	-0.430	11	9	full_conv_uut/conv_ut3/flg_reg[4]/C	full_conv_uut/co...t3/holdflg_reg/D	10.279	3.625	6.654	10.000	clk	clk
Path 3	-0.430	11	9	full_conv_uut/conv_ut4/flg_reg[4]/C	full_conv_uut/co...t4/holdflg_reg/D	10.279	3.625	6.654	10.000	clk	clk
Path 4	-0.430	11	9	full_conv_uut/conv_ut5/flg_reg[4]/C	full_conv_uut/co...t5/holdflg_reg/D	10.279	3.625	6.654	10.000	clk	clk
Path 5	-0.430	11	9	full_conv_uut/conv_ut6/flg_reg[4]/C	full_conv_uut/co...t6/holdflg_reg/D	10.279	3.625	6.654	10.000	clk	clk
Path 6	-0.091	13	33	full_conv_uut/conv_ut1/flg_reg[4]/C	full_conv_uut/co..._ut1/j_reg[13]/D	9.940	3.685	6.255	10.000	clk	clk
Path 7	-0.091	13	33	full_conv_uut/conv_ut1/flg_reg[4]/C	full_conv_uut/co..._ut1/j_reg[14]/D	9.940	3.685	6.255	10.000	clk	clk
Path 8	-0.091	13	33	full_conv_uut/conv_ut1/flg_reg[4]/C	full_conv_uut/co..._ut1/j_reg[15]/D	9.940	3.685	6.255	10.000	clk	clk
Path 9	0.143	10	9	full_conv_uut/conv_ut2/flg_reg[4]/C	full_conv_uut/co...t2/finish_reg/D	9.706	3.501	6.205	10.000	clk	clk
Path 10	0.143	10	9	full_conv_uut/conv_ut2/flg_reg[4]/C	full_conv_uut/conv_ut2/flag_reg/D	9.706	3.501	6.205	10.000	clk	clk

Implementation results with ILA



The yellow color is the part of the conv layer and the pink is the part of the pooling layer

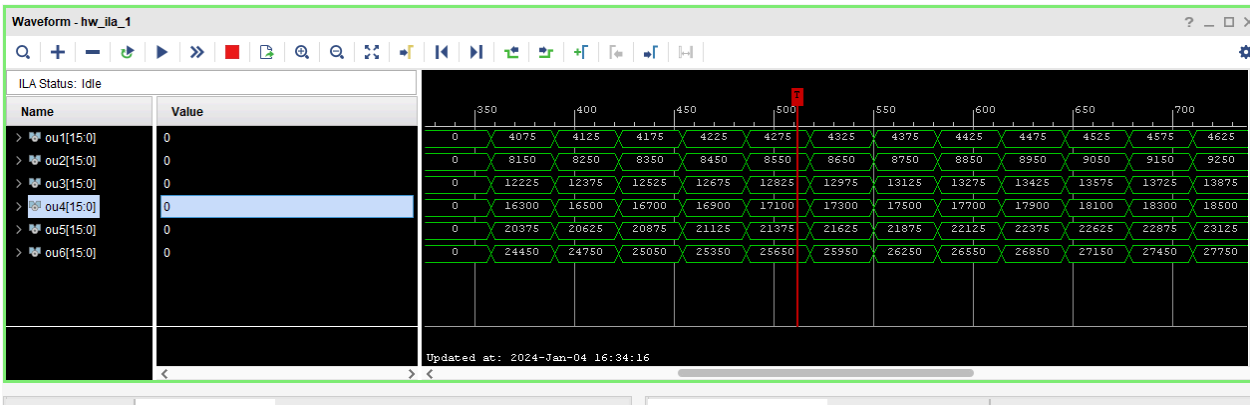
Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Slice (8150)	LUT as Logic (20800)	LUT as Memory (9600)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)	BSCANE2 (4)
full_conv_top	18643	10976	568	146	6893	18462	181	3	1	2	1
conv_ila (ila_1)	894	1638	3	0	448	737	157	3	0	0	0
dbg_hub (dbg_hub)	445	727	0	0	210	421	24	0	0	1	1
full_conv_uut (full_conv)	13848	7094	535	146	5342	13848	0	0	0	0	0
conv_ut1 (conv)	1354	553	64	24	681	1354	0	0	0	0	0
conv_ut2 (conv_0)	1329	529	64	24	623	1329	0	0	0	0	0
conv_ut3 (conv_1)	1316	529	64	24	604	1316	0	0	0	0	0
conv_ut4 (conv_2)	1319	529	64	24	668	1319	0	0	0	0	0
conv_ut5 (conv_3)	1314	529	64	24	612	1314	0	0	0	0	0
conv_ut6 (conv_4)	3620	529	114	26	1810	3620	0	0	0	0	0
pool1 (pool_layer)	3452	1494	30	0	955	3452	0	0	0	0	0
pool1 (pool)	573	249	5	0	164	573	0	0	0	0	0
pool2 (pool_163)	573	249	5	0	162	573	0	0	0	0	0
pool3 (pool_164)	577	249	5	0	161	577	0	0	0	0	0
pool4 (pool_165)	573	249	5	0	157	573	0	0	0	0	0
pool5 (pool_166)	577	249	5	0	164	577	0	0	0	0	0
pool6 (pool_167)	582	249	5	0	166	582	0	0	0	0	0

Timing

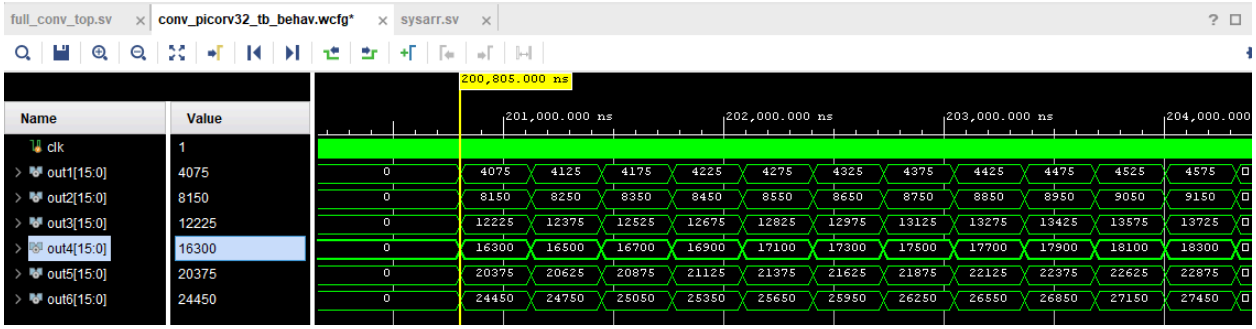
Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination
Path 1	-0.565	12	179	full_conv_uut/j_reg[5]/C	full_conv_uut/ar...g[0][30][3]_P/D	10.171	2.597	7.574	10.0	clk	clk
Path 2	-0.533	12	179	full_conv_uut/j_reg[5]/C	full_conv_uut/ar...g[0][30][3]_C/D	10.162	2.597	7.565	10.0	clk	clk
Path 3	-0.499	9	39	full_conv_uut/ar...g[3][22][1]_P/C	full_conv_uut/ar...g[1][29][1]_P/D	10.349	2.192	8.157	10.0	clk	clk
Path 4	-0.436	12	179	full_conv_uut/j_reg[5]/C	full_conv_uut/ar...g[1][29][6]_P/D	10.033	2.597	7.436	10.0	clk	clk
Path 5	-0.393	12	179	full_conv_uut/j_reg[5]/C	full_conv_uut/ar...g[2][29][7]_C/D	10.253	2.597	7.656	10.0	clk	clk
Path 6	-0.382	12	179	full_conv_uut/j_reg[5]/C	full_conv_uut/ar...g[1][29][6]_C/D	9.986	2.597	7.389	10.0	clk	clk
Path 7	-0.340	12	179	full_conv_uut/j_reg[5]/C	full_conv_uut/ar...g[2][29][6]_C/D	9.976	2.597	7.379	10.0	clk	clk
Path 8	-0.321	12	179	full_conv_uut/j_reg[5]/C	full_conv_uut/ar...g[1][31][5]_C/D	9.951	2.597	7.354	10.0	clk	clk
Path 9	-0.318	9	39	full_conv_uut/ar...g[3][22][1]_P/C	full_conv_uut/ar...g[1][29][1]_C/D	10.118	2.192	7.926	10.0	clk	clk
Path 10	-0.308	12	179	full_conv_uut/j_reg[5]/C	full_conv_uut/ar...g[1][31][6]_C/D	9.946	2.597	7.349	10.0	clk	clk

Frequency = 1/10.565 ns = 94.65 MHz.

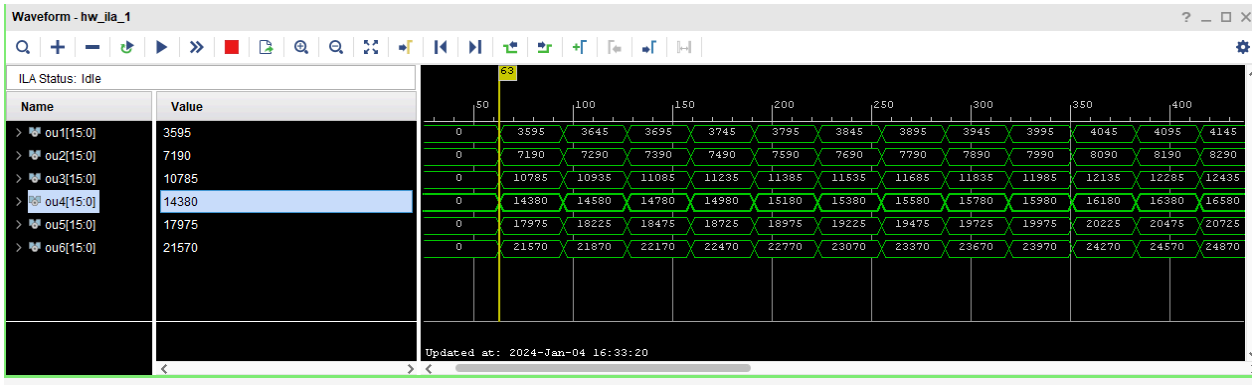
Simulation extracted from ILA



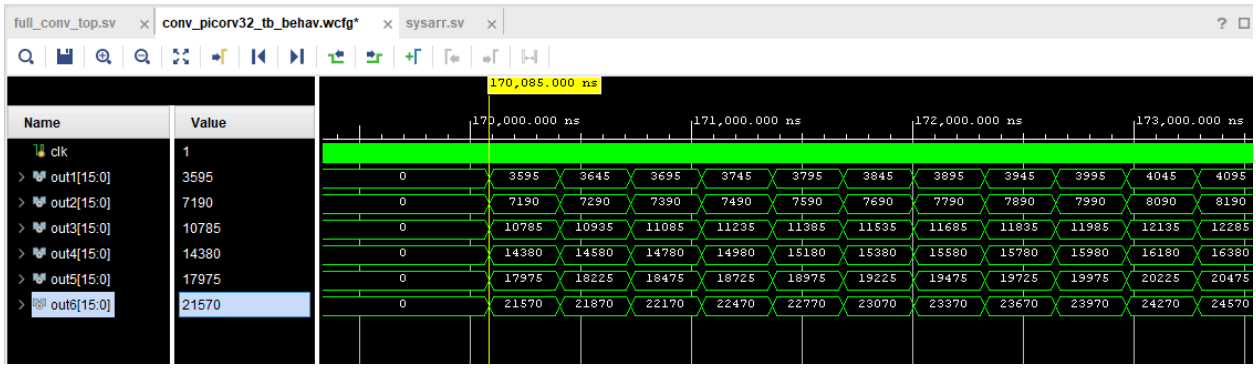
Behavioural sim



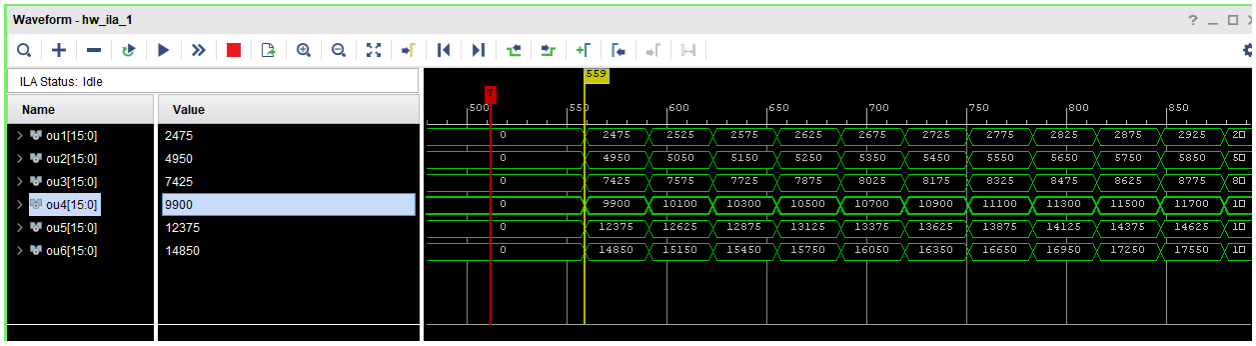
Simulation extracted from ILA



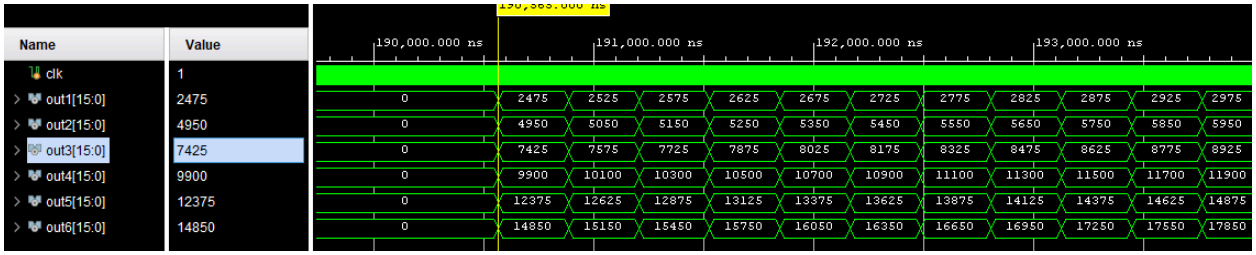
Behavioural sim



Simulation extracted from ILA

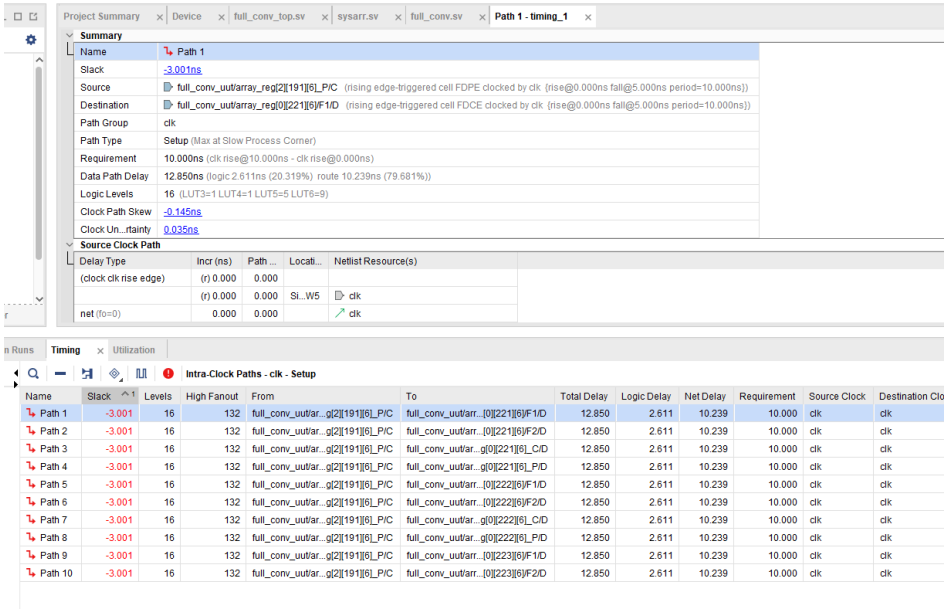


Behavioural sim



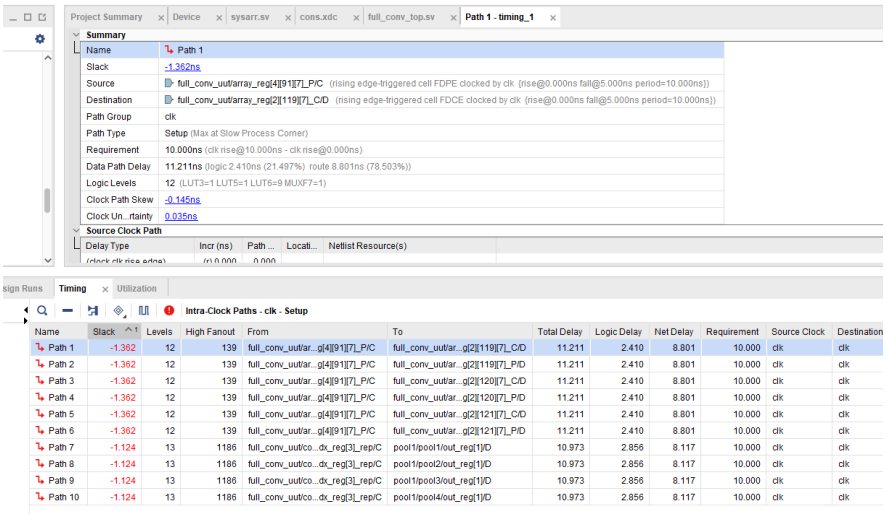
For 224x224 input size kernel size 5x5

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Bonded IOB (106)	BUFGCTRL (32)
full_conv_top	111520	57576	6345	1365	97	1
full_conv_uut (full_conv)	105548	46645	4905	1365	0	0
conv_ut1 (conv)	8197	894	664	176	0	0
conv_ut2 (conv_5)	7854	819	664	176	0	0
conv_ut3 (conv_6)	11584	823	664	176	0	0
conv_ut4 (conv_7)	18601	819	892	227	0	0
conv_ut5 (conv_8)	7851	816	664	176	0	0
conv_ut6 (conv_9)	10360	822	991	288	0	0
pool1 (pool_layer)	5962	10800	1440	0	0	0
pool1 (pool)	997	1800	240	0	0	0
pool2 (pool_0)	993	1800	240	0	0	0
pool3 (pool_1)	993	1800	240	0	0	0
pool4 (pool_2)	993	1800	240	0	0	0
pool5 (pool_3)	993	1800	240	0	0	0
pool6 (pool_4)	993	1800	240	0	0	0



For 122x122 and ker 5x5

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Bonded IOB (106)	BUFGCTRL (32)
full_conv_top	66462	29030	2863	319	97	1
full_conv_uut (full_conv)	60054	23039	2461	319	0	0
conv_ut1 (conv)	12451	816	665	225	0	0
conv_ut2 (conv_5)	4293	796	277	0	0	0
conv_ut3 (conv_6)	7236	796	422	0	0	0
conv_ut4 (conv_7)	4289	794	277	0	0	0
conv_ut5 (conv_8)	4297	799	277	0	0	0
conv_ut6 (conv_9)	5803	794	402	44	0	0
pool1 (pool_layer)	6448	5916	402	0	0	0
pool1 (pool)	1078	986	67	0	0	0
pool2 (pool_0)	1074	986	67	0	0	0
pool3 (pool_1)	1074	986	67	0	0	0
pool4 (pool_2)	1074	986	67	0	0	0
pool5 (pool_3)	1074	986	67	0	0	0
pool6 (pool_4)	1074	986	67	0	0	0



Project Summary | Device | sysarr.sv | cons.xdc | full_conv_top.sv | Path 1 - timing_1

Summary

Name	Path 1
Slack	-1.362ns
Source	full_conv_uut/array_reg[4][91][7]_P/C (rising edge-triggered cell FDPE clocked by clk {rise@0.000ns fall@5.000ns period=10.000ns})
Destination	full_conv_uut/array_reg[2][119][7]_C/D (rising edge-triggered cell FDCE clocked by clk {rise@0.000ns fall@5.000ns period=10.000ns})
Path Group	clk
Path Type	Setup (Max at Slow Process Corner)
Requirement	10.000ns (clk rise@10.000ns - clk rise@0.000ns)
Data Path Delay	11.211ns (logic 2.410ns (21.497%) route 8.801ns (78.503%))
Logic Levels	12 (LUT3=1 LUT5=1 LUT6=9 MUXF7=1)
Clock Path Skew	-0.145ns
Clock Un...rtainty	0.035ns

Source Clock Path

Delay Type	Incr (ns)	Path ...	Locati...	Netlist Resource(s)
(clock clk rise edge)	(r) 0.000	0.000		

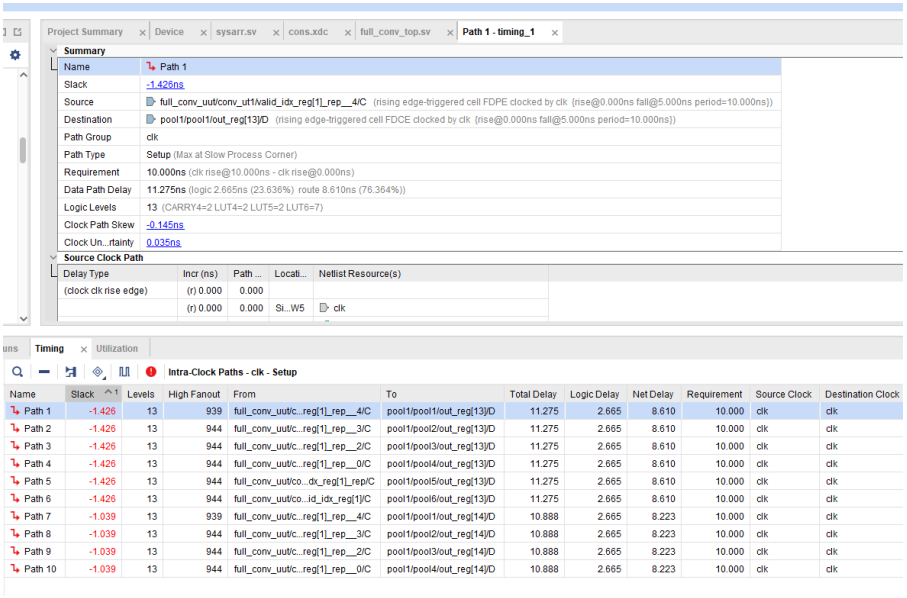
Timing | Utilization

Intra-Clock Paths - clk - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination
Path 1	-1.362	12	139	full_conv_uut/ar...g[4][91][7]_P/C	full_conv_uut/ar...g[2][119][7]_C/D	11.211	2.410	8.801	10.000	clk	clk
Path 2	-1.362	12	139	full_conv_uut/ar...g[4][91][7]_P/C	full_conv_uut/ar...g[2][119][7]_P/D	11.211	2.410	8.801	10.000	clk	clk
Path 3	-1.362	12	139	full_conv_uut/ar...g[4][91][7]_P/C	full_conv_uut/ar...g[2][120][7]_C/D	11.211	2.410	8.801	10.000	clk	clk
Path 4	-1.362	12	139	full_conv_uut/ar...g[4][91][7]_P/C	full_conv_uut/ar...g[2][120][7]_P/D	11.211	2.410	8.801	10.000	clk	clk
Path 5	-1.362	12	139	full_conv_uut/ar...g[4][91][7]_P/C	full_conv_uut/ar...g[2][121][7]_C/D	11.211	2.410	8.801	10.000	clk	clk
Path 6	-1.362	12	139	full_conv_uut/ar...g[4][91][7]_P/C	full_conv_uut/ar...g[2][121][7]_P/D	11.211	2.410	8.801	10.000	clk	clk
Path 7	-1.124	13	1186	full_conv_uut/co...dx_reg[3]_rep/C	pool1/pool1/out_reg[1]/D	10.973	2.856	8.117	10.000	clk	clk
Path 8	-1.124	13	1186	full_conv_uut/co...dx_reg[3]_rep/C	pool1/pool2/out_reg[1]/D	10.973	2.856	8.117	10.000	clk	clk
Path 9	-1.124	13	1186	full_conv_uut/co...dx_reg[3]_rep/C	pool1/pool3/out_reg[1]/D	10.973	2.856	8.117	10.000	clk	clk
Path 10	-1.124	13	1186	full_conv_uut/co...dx_reg[3]_rep/C	pool1/pool4/out_reg[1]/D	10.973	2.856	8.117	10.000	clk	clk

For 122x122 ker 3

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Bonded IOB (106)	BUFGCTRL (32)
full_conv_top	45745	26279	2161	464	97	1
full_conv_uut (full_conv)	39100	20242	1837	464	0	0
conv_ut1 (conv)	7417	486	520	176	0	0
conv_ut2 (conv_5)	2938	424	256	56	0	0
conv_ut3 (conv_6)	2936	423	256	56	0	0
conv_ut4 (conv_7)	5351	424	273	58	0	0
conv_ut5 (conv_8)	2935	423	256	56	0	0
conv_ut6 (conv_9)	3019	424	258	56	0	0
pool1 (pool_layer)	6687	6006	324	0	0	0
pool1 (pool)	1119	1001	54	0	0	0
pool2 (pool_0)	1113	1001	54	0	0	0
pool3 (pool_1)	1114	1001	54	0	0	0
pool4 (pool_2)	1114	1001	54	0	0	0
pool5 (pool_3)	1114	1001	54	0	0	0
pool6 (pool_4)	1113	1001	54	0	0	0



Project Summary | Device | sysarr.sv | cons.xdc | full_conv_top.sv | Path 1 - timing_1

Summary

Name	Path 1
Slack	-1.426ns
Source	full_conv_uut/conv_ut1/valid_idx_reg[1]_rep__4/C (rising edge-triggered cell FDPE clocked by clk {rise@0.000ns fall@5.000ns period=10.000ns})
Destination	pool1/pool1/out_reg[13]/D (rising edge-triggered cell FDCE clocked by clk {rise@0.000ns fall@5.000ns period=10.000ns})
Path Group	clk
Path Type	Setup (Max at Slow Process Corner)
Requirement	10.000ns (clk rise@10.000ns - clk rise@0.000ns)
Data Path Delay	11.275ns (logic 2.665ns (23.636%) route 8.610ns (76.364%))
Logic Levels	13 (CARRY4=2 LUT4=2 LUT5=2 LUT6=7)
Clock Path Skew	-0.145ns
Clock Un...rtainty	0.035ns

Source Clock Path

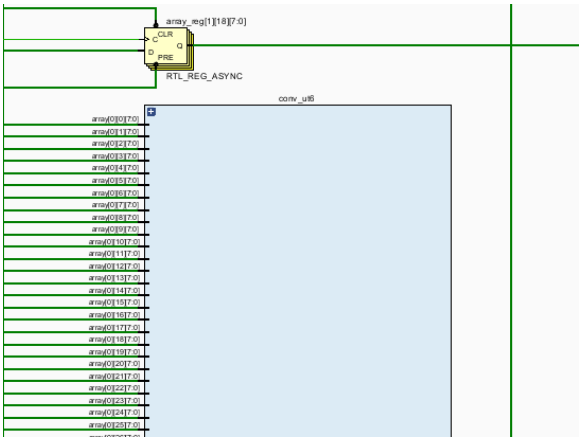
Delay Type	Incr (ns)	Path ...	Locati...	Netlist Resource(s)
(clock clk rise edge)	(r) 0.000	0.000		
	(r) 0.000	0.000	Si...W5	clk

Timing | Utilization

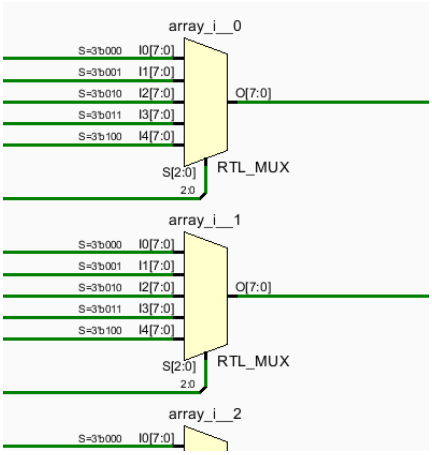
Intra-Clock Paths - clk - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock
Path 1	-1.426	13	939	full_conv_uut/c...reg[1]_rep__4/C	pool1/pool1/out_reg[13]/D	11.275	2.665	8.610	10.000	clk	clk
Path 2	-1.426	13	944	full_conv_uut/c...reg[1]_rep__3/C	pool1/pool2/out_reg[13]/D	11.275	2.665	8.610	10.000	clk	clk
Path 3	-1.426	13	944	full_conv_uut/c...reg[1]_rep__2/C	pool1/pool3/out_reg[13]/D	11.275	2.665	8.610	10.000	clk	clk
Path 4	-1.426	13	944	full_conv_uut/c...reg[1]_rep__0/C	pool1/pool4/out_reg[13]/D	11.275	2.665	8.610	10.000	clk	clk
Path 5	-1.426	13	944	full_conv_uut/co...dx_reg[1]_rep/C	pool1/pool5/out_reg[13]/D	11.275	2.665	8.610	10.000	clk	clk
Path 6	-1.426	13	944	full_conv_uut/co...id_idx_reg[1]/C	pool1/pool6/out_reg[13]/D	11.275	2.665	8.610	10.000	clk	clk
Path 7	-1.039	13	939	full_conv_uut/c...reg[1]_rep__4/C	pool1/pool1/out_reg[14]/D	10.888	2.665	8.223	10.000	clk	clk
Path 8	-1.039	13	944	full_conv_uut/c...reg[1]_rep__3/C	pool1/pool2/out_reg[14]/D	10.888	2.665	8.223	10.000	clk	clk
Path 9	-1.039	13	944	full_conv_uut/c...reg[1]_rep__2/C	pool1/pool3/out_reg[14]/D	10.888	2.665	8.223	10.000	clk	clk
Path 10	-1.039	13	944	full_conv_uut/c...reg[1]_rep__0/C	pool1/pool4/out_reg[14]/D	10.888	2.665	8.223	10.000	clk	clk

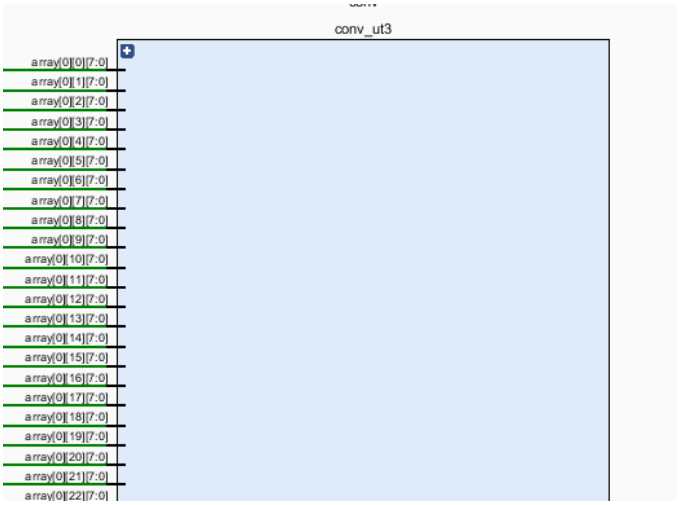
- In the above three cases, we can observe that the accelerator is size-dependent. It’s better to build an accelerator that is defined for a maximum image size and works for all the image sizes less than the maximum.
- Each convolution unit also increases with the increase in the size of the input image, As most operations in the conv module are the same in all the instantiations we can better optimize it here as we have many instantiations. If all the common operations are done once. There will be a significant decrease in resources and probably could increase the frequency of operation.



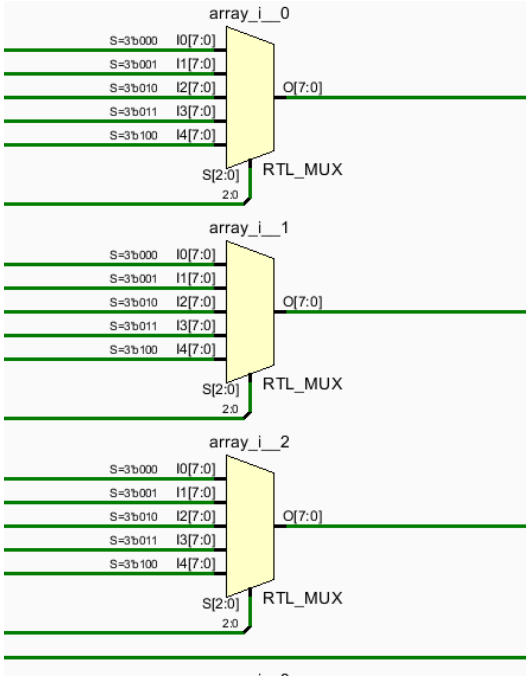
row decoders in the conv _ut6 module

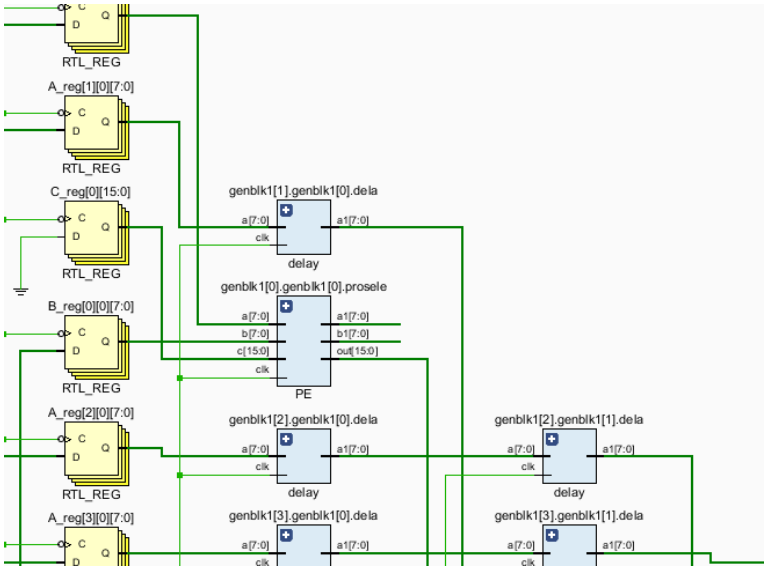


similarly,

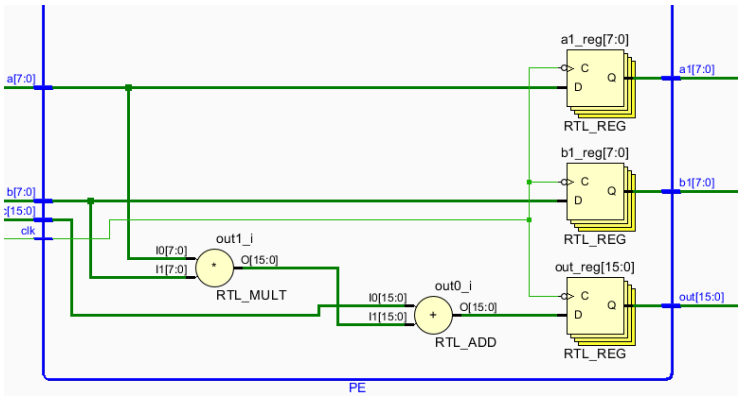


Row decoders of the conv_ut3 module





Even the systolic array doesn’t have to pass the data.



Results for the implementation with a maximum image size.

Results are generated for the Artix-7 Basys-3 Board.

Max Image size = 32x32

# CC per conv	Img Size	Ker Size	# CC	#LUT	#FF	Time Period(ns)	Freq (MHz)	Power (W)
5.56	32x32	5x5	4361	20936	10722	12.921	77.4	0.229
6.4	16x16	5x5	921					
45	5x5	5x5	45					

Max Image size = 224x224

# CC per conv	Img Size	Ker Size	# CC	#LUT	#FF	Time Period(ns)	Freq (MHz)	Power (W)
5.06	224x224	5x5	245321	106570	33748	13.681	73.1	0.852
5.14	112x112	5x5	59961					
5.56	32x32	5x5	4361					
45	5x5	5x5	45					

Results for the implementation with a given maximum image size, and a maximum kernel size

Results are generated for the Artix-7 Basys-3 Board.

Max Image size = 32x32
Max Kernel size = 5x5

# CC per conv	Img Size	Ker Size	# CC	#LUT	#FF	Time Period(ns)	Freq (MHz)	Power (W)
5.77	32x32	5x5	4528	19407	10728	13.066	76.5	
3.37	32x32	3x3	3037					
6.88	16x16	5x5	992					
3.83	16x16	3x3	751					
50	5x5	5x5	50					

Max Image size = 224x224
Max Kernel size = 5x5

# CC per conv	Img Size	Ker Size	# CC	#LUT	#FF	Time Period(ns)	Freq (MHz)	Power (W)
5.095	224x224	5x5	246640	102722	40887	13.918	71.84	
3.049	224x224	3x3	150301					
5.196	112x112	5x5	60608					
3.10	112x112	3x3	37517					
5.77	32x32	5x5	4528					
3.37	32x32	3x3	3037					

Results after optimizing Img read by removing multiple read (row and column) decoders in conv module:

Results are generated for the Artix-7 Basys-3 Board.

Max Image size = 32x32
Max Kernel size = 5x5

# CC per conv	Img Size	Ker Size	# CC	#LUT	#FF	Time Period(ns)	Freq (MHz)	Power (W)
	32x32	5x5		16138	10427	13.46	74.3	
	32x32	3x3						
	16x16	5x5						
	16x16	3x3						
	5x5	5x5						

Max Image size = 224x224
Max Kernel size = 5x5

# CC per conv	Img Size	Ker Size	# CC	#LUT	#FF	Time Period(ns)	Freq (MHz)	Power (W)
	224x224	5x5		74302	39126	14.53	68.8	
	224x224	3x3						
	112x112	5x5						
	112x112	3x3						
	32x32	5x5						
	32x32	3x3						

Parametrizing the number of Kernels

The parameterization is done by converting all the K input kernels of size nxn to a single 2D array of size Kx(n square).

Where each row is a Kernel. All the rows of a Kernel are serialized into a single row of the new 2D array.

Max Image size = 32x32
Max Kernel size = 5x5

# CC per conv	Img Size	Ker Size	# CC	#LUT	#FF	Time Period(ns)	Freq (MHz)	No of kernels
5.74	32x32	5x5	4501	16027	10385	13.926	71.8	6
5.74	32x32	5x5	4501	15502	9990	13.365	74.8	5
5.74	32x32	5x5	4501	14621	9598	12.636	79.138	4
5.74	32x32	5x5	4501	13826	9205	12.628	79.189	3
5.74	32x32	5x5	4501	13138	8818	12.544	79.719	2
5.74	32x32	5x5	4501	10965	4464	13.087	76.412	1

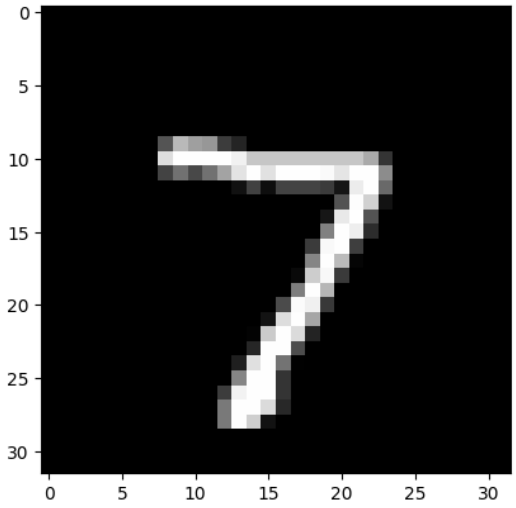
No of kernels = 6

# CC per conv	Img Size	Ker Size	# CC	#LUT	#FF	Time Period(ns)	Freq (MHz)	Power (W)
	32x32	5x5						
	32x32	3x3						
	16x16	5x5						
	16x16	3x3						
	5x5	5x5						

Next Tasks:

1. Debug the case for Ker_size < Max_size.
2. Test the circuit for all the cases
    a. Ker_size = Max_ker_size, Img_size = Max_img_size, nok>1
    b. Ker_size < Max_ker_size, Img_size = Max_img_size, nok>1
    c. Ker_size = Max_ker_size, Img_size < Max_img_size,nok>1
    d. Ker_size < Max_ker_size, Img_size < Max_img_size
    e. Img_size = Ker_size < Max_img_size, nok > 1
    f. See the results while decreasing the nok(number of kernels)
3. The Test cases from the Mnist data set and Imagenet data set. And generate the outputs via simulations and find ssim with matlab/python generated outputs.
4. Generate the hardware results considering all the cases.
5. Benchmark with previous papers.
6. Try to generate results using yosys for 1 or more cases. Using various Technology libraries.

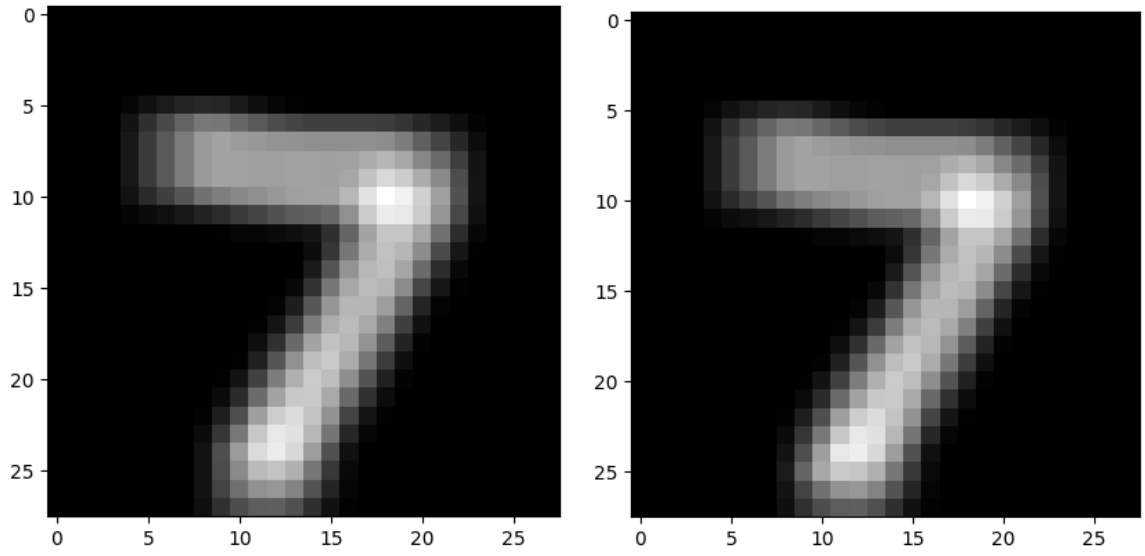
For image seven.png



For the kernel

[1,1,1,1,1;
  1,1,1,1,1;
  1,1,1,1,1;
  1,1,1,1,1;
  1,1,1,1,1;]

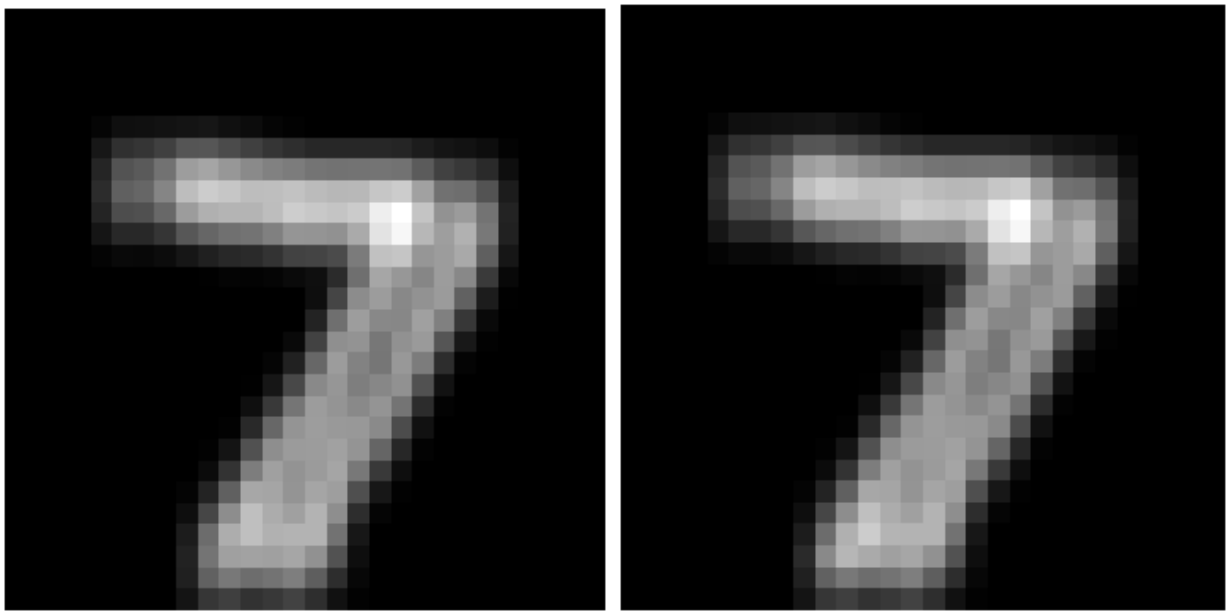
The output generated in Matlab/Python and Output from conv unit are



The ssim for this case is 0.9997. (13 pixels are diff)

For the kernel
[
   2, 1, 0, 1, 2;
   3, 2, 0, 2, 3;
   4, 3, 0, 3, 4;
   3, 2, 0, 2, 3;
   2, 1, 0, 1, 2;
];

The output generated in Matlab/Python and Output from the conv unit are:

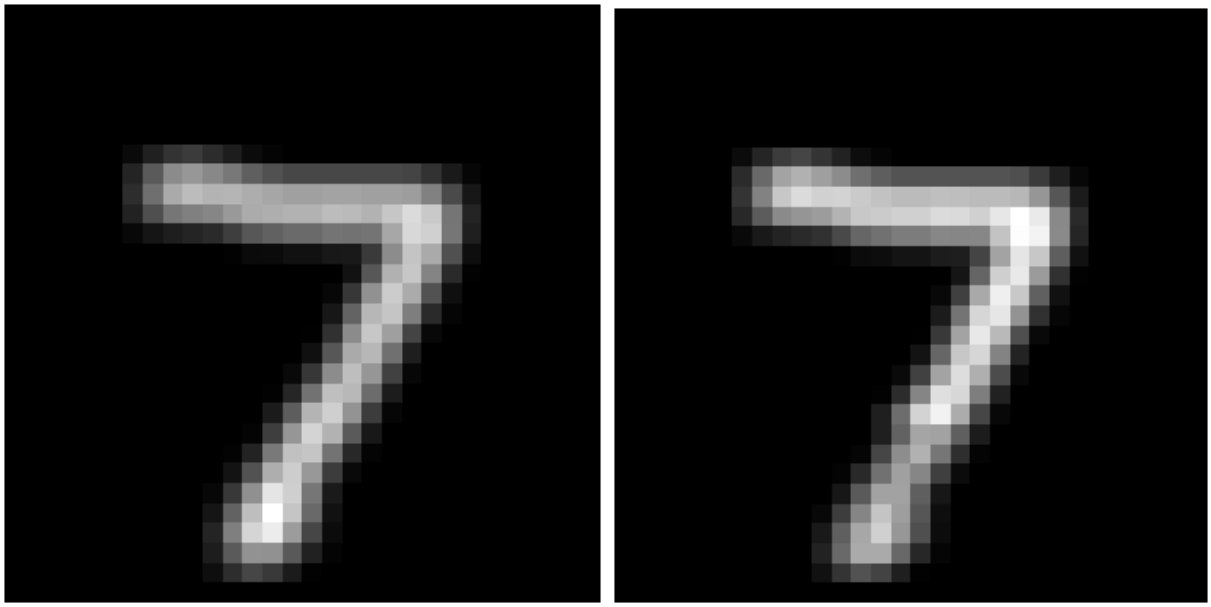


The ssim for this case is 0.9997. (13 pixels are diff)

For the kernel

```
[
   1,1,1;
   1,1,1;
   1,1,1;
];
```

The output generated in Matlab/Python and Output from the conv unit are:

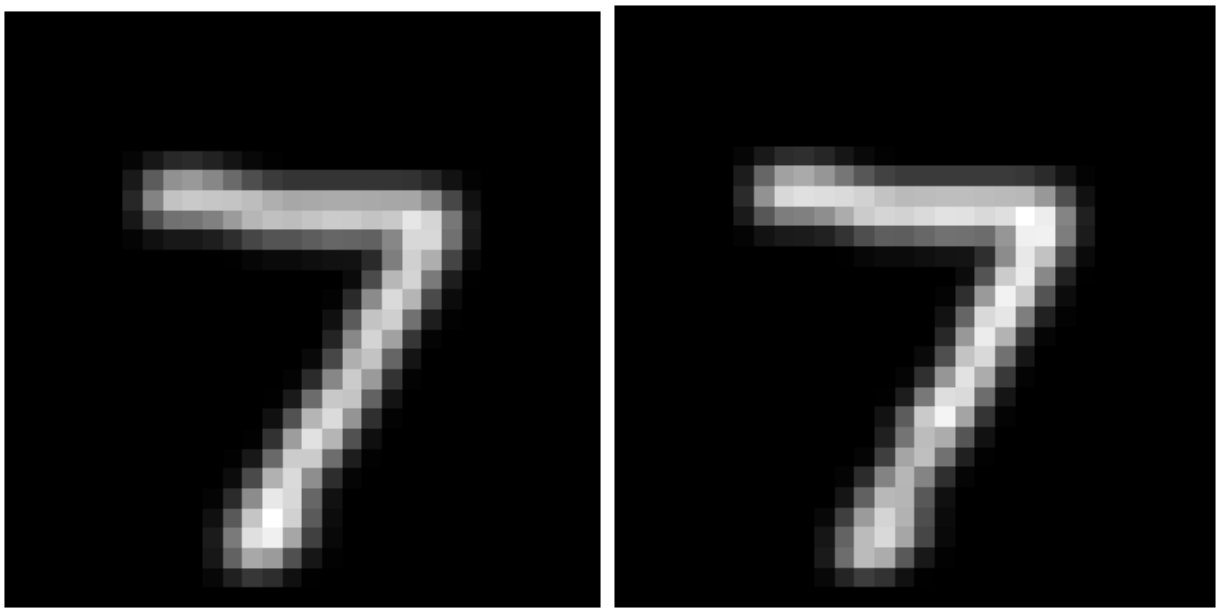


The ssim for this case is 0.9820. (39 pixels are diff)

For the kernel

```
[
   1,2,1;
   2,4,2;
   1,2,1;
];
```

The output generated in Matlab/Python and Output from the conv unit are:



The ssim for this case is 0.9913 (39 pixels are different)

Max Image size = 32x32
Max Kernel size = 5x5

# CC per conv	Img Size	Ker Size	# CC	#LUT	#FF	Time Period(ns)	Freq (MHz)	No of kernels
5.52	32x32	5x5	4335	21380	11529	12.397	80.664	8
5.52	32x32	5x5	4335	19707	11083	12.4	80.645	7
5.52	32x32	5x5	4335	18989	10649	12.991	76.976	6
5.52	32x32	5x5	4335	17991	10213	12.959	77.166	5
5.52	32x32	5x5	4335	16254	9760	12.332	81.089	4
5.52	32x32	5x5	4335	15727	9350	12.490	80.064	3
5.52	32x32	5x5	4335	13917	8910	12.452	80.308	2
5.52	32x32	5x5	4335	13905	8472	12.460	80.256	1

No of kernels = 6

# CC per conv	Img Size	Ker Size	# CC	#LUT	#FF	Time Period(ns)	Freq (MHz)
5.52	32x32	5x5	4335	18989	10649	12.991	76.976
3.34	32x32	3x3	3009	18989	10649	12.991	76.976
6.31	16x16	5x5	910	18989	10649	12.991	76.976
3.76	16x16	3x3	737	18989	10649	12.991	76.976
44	5x5	5x5	44	18989	10649	12.991	76.976

Max Image size = 224x224
Max Kernel size = 5x5

# CC per conv	Img Size	Ker Size	# CC	#LUT	#FF	Time Period(ns)	Freq (MHz)	No of kernels
5.06	224x224	5x5	245103	75980	38446	14.130	70.771	6
5.06	224x224	5x5	245103	76617	34866	13.563	73.730	5
5.06	224x224	5x5	245103	74938	35858	13.415	74.543	4
5.06	224x224	5x5	245103	73133	33995	13.047	76.645	3
5.06	224x224	5x5	245103	74182	41385	13.205	75.728	2
5.06	224x224	5x5	245103	73476	42843	12.625	79.207	1

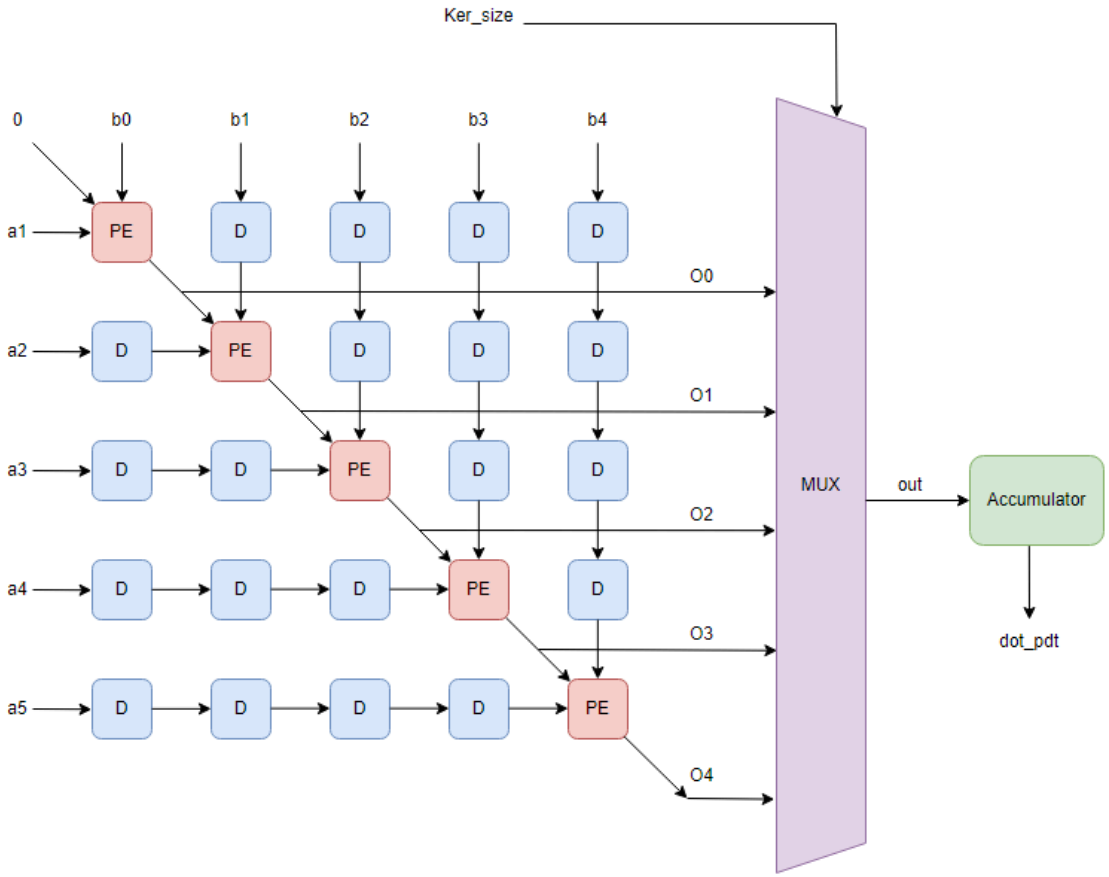
No of kernels = 6

# CC per conv	Img Size	Ker Size	# CC	#LUT	#FF	Time Period(ns)	Freq (MHz)
5.06	224x224	5x5	245103	75980	38446	13.341	74.956
3.045	224x224	3x3	150081	75980	38446	13.341	74.956
5.13	112x112	5x5	59855	75980	38446	13.341	74.956
3.09	112x112	3x3	37409	75980	38446	13.341	74.956
5.52	32x32	5x5	4335	75980	38446	13.341	74.956
3.34	32x32	3x3	3009	75980	38446	13.341	74.956

The architecture of the Programmable Systolic array.

The following image has the architecture for the Programmable systolic array used. This is implemented in System Verilog, the maximum size of the systolic array is parametrizable. After fixing a maximum size, the systolic array can be programmable for the given ker_size that is less than or equal to the maximum.

Here PE is the Processing element which involves the mac operation out = c + a*b. D is the Delay block that consists of a register, to hold the data for a clock cycle.



The outputs after each of the Processing elements is passed to the Mux and the output of the Mux is given to the Accumulator. (This is the part to make the systolic array programmable.

Benchmarking the design with "An FPGA based Tiled Systolic Array Generator to Accelerate CNNs"

The results in this paper are generated for the Zynq series Zedboard.
The latency and resources for one image convolution with a 3x3 kernel are given in the following table

Design	Latency for 28x28 in us	Latency for 224x224 in us	LUTs
MMDA(DSP)	3.851	243.21	430
MMDA(no DSP)	5.638	356.121	1335
EWMA	10.469	711.260	469
TSA	0.557	29.226	44774
EWMA(hls no pipelining)	280	21689	221
EWMA(hls pipelining)	11	755	32575

Our design has more than 1 kernel. The following row indicates the values for 6 kernels

3.34	28x28	3x3	2297	18989	10649	12.991	76.976

The Latency = 12.991 n * 2297 (clock period * no of clock cycles)
= 29840.327 ns = 29.84 us

3.045	224x224	3x3	150081	75980	38446	13.341	74.956

The Latency = 13.341 n * 150081 (clock period * no of clock cycles)
= 2002230.621 ns
= 2002.23 us

The comparison will be done based on Area*Delay product this is done with no of LUTS*Latency. The following table has the Lut*latency product. 'noc' indicates the number of convolutions

Design	**LDP for 28x28 * noc**	**LDP for 224x224 * noc**
MMDA(DSP)	9935.58 u	627481.8 u
MMDA(no DSP)	45172.38 u	2852529 u
EWMA	29459.76 u	2001485.64 u
TSA	149634.6 u	7851389.52 u
EWMA(hls no pipelining)	371280 u	28759614 u
EWMA(hls pipelining)	2149950 u	147564750 u
Our results with noc = 6 With max image len as 32 for 28x28, with max img len as 224 for 224x224, and max ker as 5x5	566631.76 u	152127156 u

For the 28x28 the LDP for our design is better than that of pipelined implementation of hls. When compared to other designs the proposed architecture is not better because of various reasons such as limited input throughput. But other designs are made with maximum throughput that is not the case in Edge devices where the data with high throughputs are not available. And also the delay is calculated for the basys-3 board in this design. The design would be faster on the Zed board when compared to the basys board. The results of the other designs are fixed for a single kernel.

The design can be improved by adding more number of kernels that would increase the throughput without increasing the no of LUTS. By running the systolic array and the convolution module asynchronously we can reduce the duration by 0.5 times.

This design has the advantage of a programmable kernel and image lengths. This is designed for areas where throughput is less. (less than 1-pixel data per clock cycle).