Weekly Progress Report

Project: TinyML-Based Project on FPGA Board with RISC-V Core

Name: Aman Chauhan
Branch: Btech. Computer Science and Engineering (CSE Core)
Roll Number: 22BCE0476
College: Vellore Institute of Technology, Vellore
Mentor: Dr. Sudip Roy

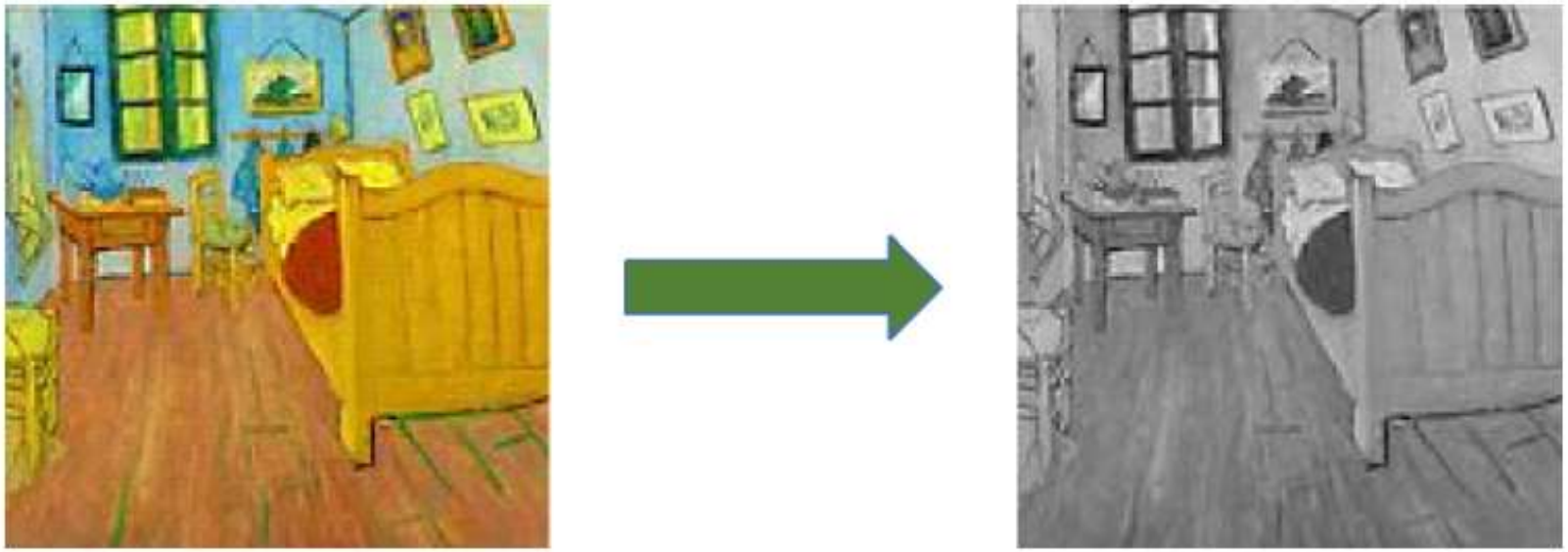
Week: *4th report*
Date: *13th May 2025*

Summary of Week 4 Activities

- Completed “**Mixing C assembly functions**" lesson from the Harvard edX RVfpga Course using SweRVolf on the **Nexys A7 Virtual Board.**
- Started working in **Xilinx Vivado for FPGA-based RISC-V** system design.
- Soft Core CPU RiscV in Xlinx.
- Initiated a TinyML-based soft computing **multi-classification model**.

Rvfpga Image Processing

Program that processes an RGB image (left side of the image below), and generates a grayscale version of that image (right side of the image below).



Transformation of an RGB Image to a Grayscale Image

```c
int main(void) {
    // Create an NxM matrix using the input image
    initColourImage(ColourImage);

    // Transform Colour Image to Grey Image
    ColourToGrey(ColourImage,GreyImage);

    // Initialize Uart
    uartInit();
    // Print message on the serial output
    printfNexys("Created Grey Image");

    while(1);

    return 0;
}
```

```c
typedef struct {
    unsigned char R;
    unsigned char G;
    unsigned char B;
} RGB;
```

```c
unsigned char GreyImage[N][M];
```

2-Dimensional Array of Characters

```c
extern int ColourToGrey_Pixel(int R, int G, int B);

void ColourToGrey(RGB Colour[N][M], unsigned char Grey[N][M]) {
    int i,j;

    for (i=0;i<N;i++)
        for (j=0; j<M; j++)
            Grey[i][j] =  ColourToGrey_Pixel(Colour[i][j].R, Colour[i][j].G, Colour[i][j].B);
}
```

ColourToGrey Function and ColourToGrey_Pixel Subroutine

```
.globl ColourToGrey_Pixel

.text

ColourToGrey_Pixel:

    li x28, 306
    mul a0, a0, x28

    li x28, 601
    mul a1, a1, x28

    li x28, 117
    mul a2, a2, x28

    add a0, a0, a1
    add a0, a0, a2

    srl a0, a0, 10

    ret

.end
```

•**SweRV EH1** lacks floating-point support, so **RGB to Grayscale** conversion uses integer arithmetic.

•Standard weights (0.299, 0.587, 0.114) are scaled by 1024, giving integers: 306 (R), 601 (G), 117 (B).

•Final formula: **Grayscale = (306×R + 601×G + 117×B) >> 10** (equivalent to dividing by 1024).

•Ensures grayscale output stays within 0–255 range.

•Implemented using a C function (ColourToGrey) and an assembly subroutine (ColourToGrey_Pixel).

Xilinx Vivado for FPGA-based RISC-V

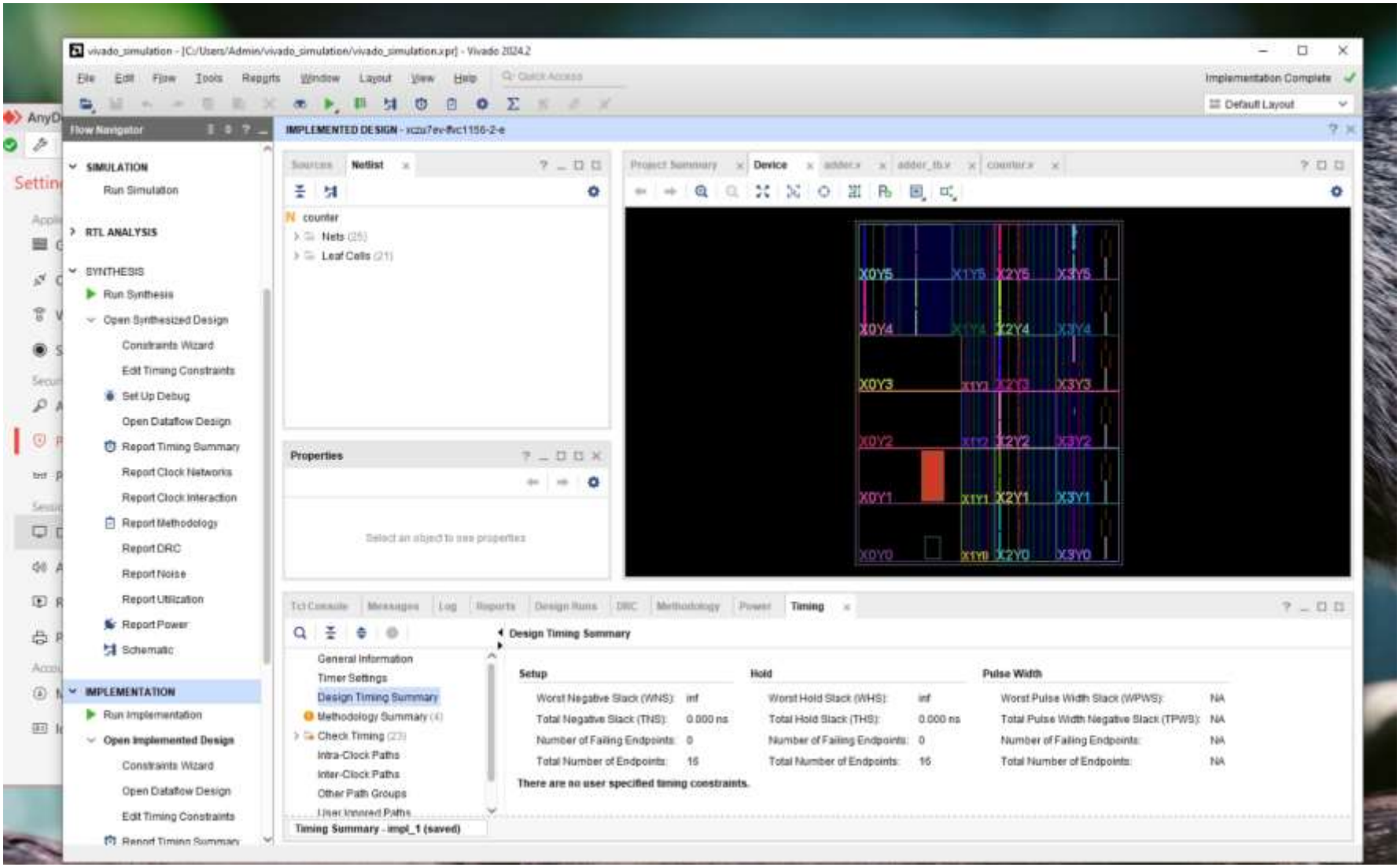
•Used **Xilinx Vivado Design Suite** to develop a **RISC-V based system** on an **industrial-grade FPGA board**.

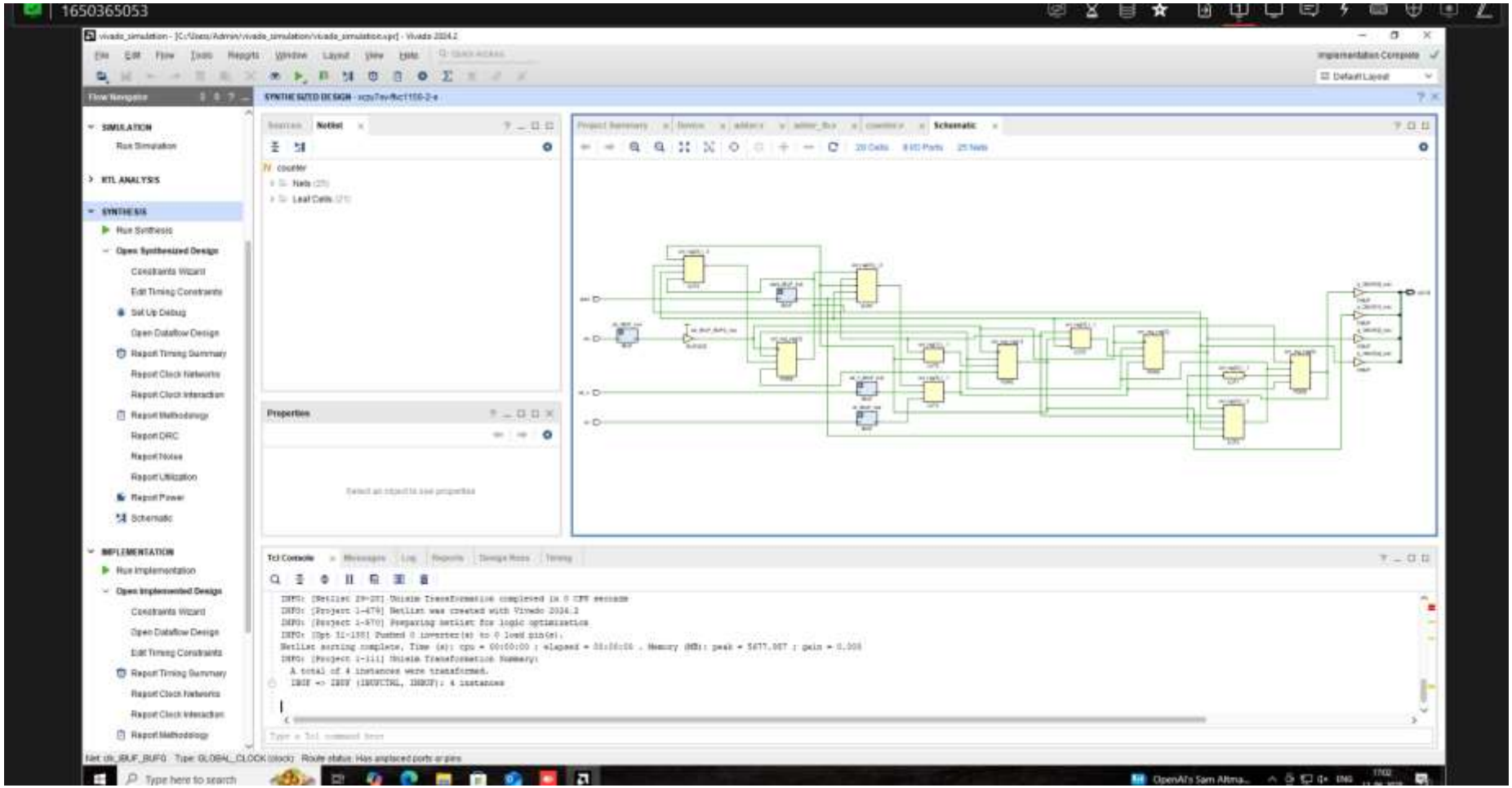
•Created a **custom block design** using Vivado IP Integrator, integrating RISC-V core with AXI interconnects.

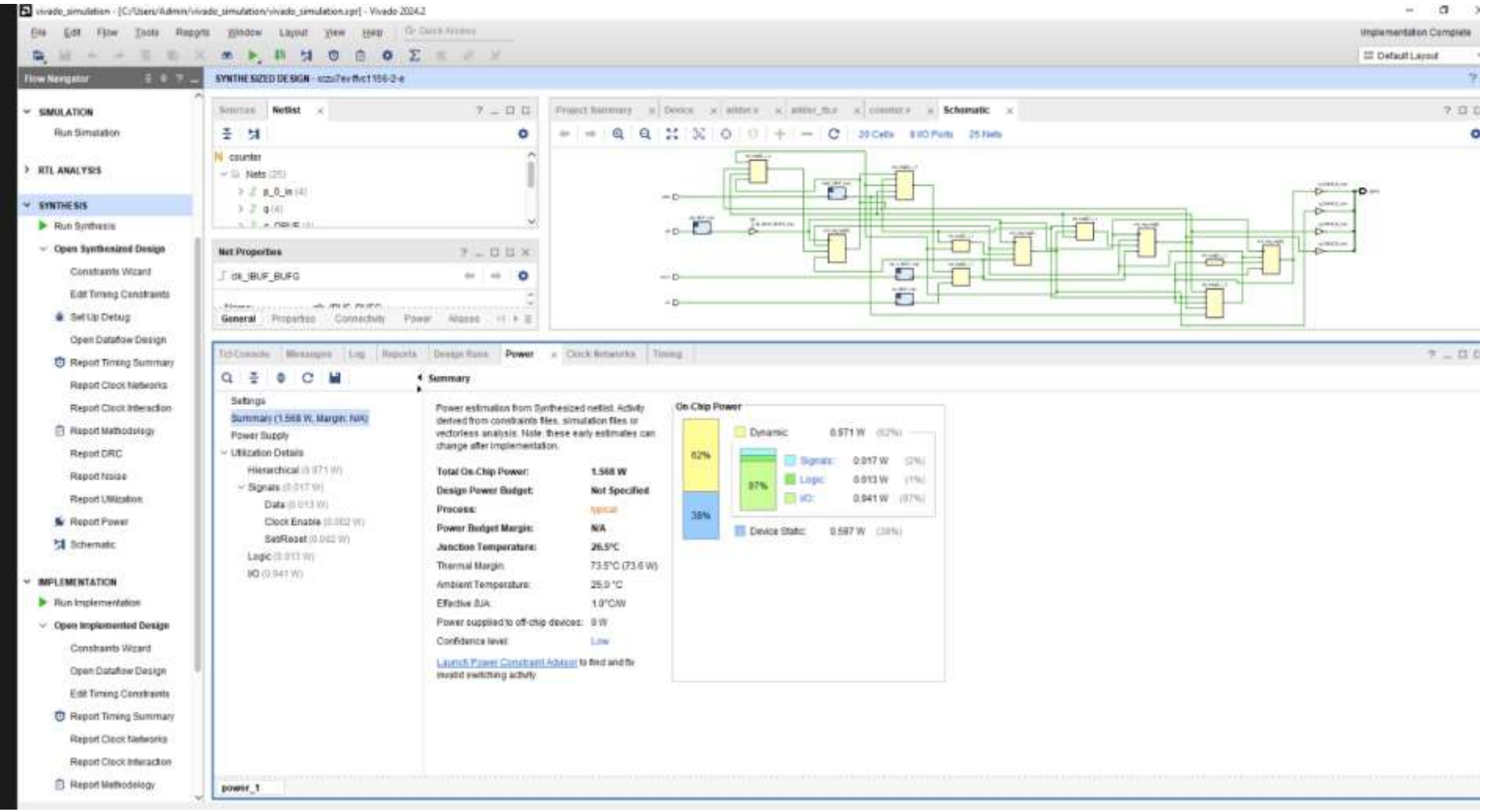
•Configured **memory controllers and peripheral interfaces** suitable for SoC-style architecture.

•Synthesized and implemented the design, followed by **bitstream generation** and **hardware debugging**.

•Focused on real-world deployment, gaining insights into **timing analysis**, **resource utilization**, and **hardware constraints**.


vivado_simulation - [C:/Users/Admin/vivado_simulation/vivado_simulation.xpr] - Vivado 2024.2
File Edit Flow Tools Reports Window Layout View Help
Implementation Complete
Default Layout
Flow Navigator
IMPLEMENTED DESIGN - xczu7ev-ffvc1156-2-e
SIMULATION
Run Simulation
RTL ANALYSIS
SYNTHESIS
Run Synthesis
Open Synthesized Design
Constraints Wizard
Edit Timing Constraints
Set Up Debug
Open Dataflow Design
Report Timing Summary
Report Clock Networks
Report Clock Interaction
Report Methodology
Report DRC
Report Noise
Report Utilization
Report Power
Schematic
IMPLEMENTATION
Run Implementation
Open Implemented Design
Constraints Wizard
Open Dataflow Design
Edit Timing Constraints
Sources
Netlist
counter
Nets (25)
Leaf Cells (21)
Properties
Select an object to see properties
Project Summary
Device
adder.v
adder_tb.v
counter.v
X0Y5
X1Y5
X2Y5
X3Y5
X0Y4
X1Y4
X2Y4
X3Y4
X0Y3
X1Y3
X2Y3
X3Y3
X0Y2
X1Y2
X2Y2
X3Y2
X0Y1
X1Y1
X2Y1
X3Y1
X0Y0
X1Y0
X2Y0
X3Y0
Tcl Console
Messages
Log
Reports
Design Runs
DRC
Methodology
Power
Timing
General Information
Timer Settings
Design Timing Summary
Methodology Summary (4)
Check Timing (23)
Intra-Clock Paths
Inter-Clock Paths
Other Path Groups
Design Timing Summary
Setup
Worst Negative Slack (WNS): inf
Total Negative Slack (TNS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 16
Hold
Worst Hold Slack (WHS): inf
Total Hold Slack (THS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 16
Pulse Width
Worst Pulse Width Slack (WPWS): NA
Total Pulse Width Negative Slack (TPWS): NA
Number of Failing Endpoints: NA
Total Number of Endpoints: NA
There are no user specified timing constraints.
Timing Summary - impl_1 (saved)


1650365053
SIMULATION
Run Simulation
RTL ANALYSIS
SYNTHESIS
Run Synthesis
Open Synthesized Design
Constraints Wizard
Edit Timing Constraints
Set Up Debug
Open Dataflow Design
Report Timing Summary
Report Clock Networks
Report Clock Interaction
Report Methodology
Report DRC
Report Noise
Report Utilization
Report Power
Schematic
IMPLEMENTATION
Run Implementation
Open Implemented Design
Constraints Wizard
Open Dataflow Design
Edit Timing Constraints
Report Timing Summary
Report Clock Networks
Report Clock Interaction
Report Methodology
Netlist
Schematic
Properties
Tcl Console
Type here to search


vivado_simulation - [C:/Users/Admin/vivado_simulation/vivado_simulation.xpr] - Vivado 2024.2
Flow Navigator
SIMULATION
Run Simulation
RTL ANALYSIS
SYNTHESIS
Run Synthesis
Open Synthesized Design
Constraints Wizard
Edit Timing Constraints
Set Up Debug
Open Dataflow Design
Report Timing Summary
Report Clock Networks
Report Clock Interaction
Report Methodology
Report DRC
Report Noise
Report Utilization
Report Power
Schematic
IMPLEMENTATION
Run Implementation
Open Implemented Design
Constraints Wizard
Open Dataflow Design
Edit Timing Constraints
Report Timing Summary
Report Clock Networks
Report Clock Interaction
Report Methodology
Implementation Complete
Default Layout
SYNTHESIZED DESIGN - xczu7ev-ffvc1156-2-e
Sources
Netlist
counter
Nets (25)
a_0_in (4)
q (4)
Net Properties
clk_IBUF_BUFG
General
Properties
Connectivity
Power
Aliases
Project Summary
Device
adder.v
adder_tb.v
counter.v
Schematic
20 Cells
8 I/O Ports
25 Nets
Tcl Console
Messages
Log
Reports
Design Runs
Power
Clock Networks
Timing
Summary
Settings
Summary (1.568 W, Margin: N/A)
Power Supply
Utilization Details
Hierarchical (0.971 W)
Signals (0.017 W)
Data (0.013 W)
Clock Enable (0.002 W)
Set/Reset (0.002 W)
Logic (0.013 W)
I/O (0.941 W)
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: 1.568 W
Design Power Budget: Not Specified
Process: typical
Power Budget Margin: N/A
Junction Temperature: 26.5°C
Thermal Margin: 73.5°C (73.6 W)
Ambient Temperature: 25.0 °C
Effective ΘJA: 1.0°C/W
Power supplied to off-chip devices: 0 W
Confidence level: Low
Launch Power Constraint Advisor to find and fix invalid switching activity
On-Chip Power
62%
38%
97%
Dynamic 0.971 W (62%)
Signals: 0.017 W (2%)
Logic: 0.013 W (1%)
I/O: 0.941 W (97%)
Device Static: 0.597 W (38%)
power_1

Multi-Classification Model

Steps Involved:-

Libraries Installation:-

Oceans cover two-thirds of the planet. In this assignment, you will build a classifier to tell several types of creatures apart.

```
import os

from collections import Counter

import matplotlib.pyplot as plt
import torch
import torch.nn as nn
import torch.optim as optim
from PIL import Image
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix
from torch.utils.data import DataLoader, random_split
from torchinfo import summary
from torchvision import datasets, transforms
from tqdm.notebook import tqdm

torch.backends.cudnn.deterministic = True
```

Environment Setup:-

```
if torch.cuda.is_available():
    device = "cuda"
```

```
os.listdir("sea_creatures")
```

```
['test', 'train']
```

```
train_dir = "sea_creatures/train"

# Get the list of class names (each folder is a class)
classes = os.listdir(train_dir)

# Print the class names
print(classes)
```

```
['Puffers', 'Sea Urchins', 'Turtle_Tortoise', 'Whale', 'Jelly Fish', 'Sharks', 'Octopus', 'Sea Rays', 'Dolphin']
```

Transform Pipeline:-

```
height = 224
width = 224
class ConvertToRGB:
    def __call__(self, img):
        if img.mode != "RGB":
            img = img.convert("RGB")
        return img
transform = transforms.Compose([
    ConvertToRGB(),
    transforms.Resize((224, 224)),
    transforms.ToTensor()
])
print(transform)
```

```
Compose(
    <__main__.ConvertToRGB object at 0x7fa102cbbb10>
    Resize(size=(224, 224), interpolation=bilinear, max_size=None, antialias=True)
    ToTensor()
)
```

```
sample_file = "sea_creatures/train/Dolphin/10004986625_0f786ab86b_b.jpg"

image = Image.open(sample_file)

transformed_image = transform(image)
print(transformed_image.shape)
```

```
torch.Size([3, 224, 224])
```

```
dataset = ImageFolder("sea_creatures/train", transform=transform)
print("Image size", dataset[0][0].shape)
print("Label", dataset[0][1])
```

```
Image size torch.Size([3, 224, 224])
Label 0
```

Data loader(batch size-30)

```
batch_size = 32
dataset_loader = DataLoader(dataset, batch_size=batch_size)
# Get one batch
first_batch = next(iter(dataset_loader))
print(f"Shape of one batch: {first_batch[0].shape}")
print(f"Shape of labels: {first_batch[1].shape}")
```

```
Shape of one batch: torch.Size([32, 3, 224, 224])
Shape of labels: torch.Size([32])
```

Transform Normalize:-

```
transform_norm =transforms.Compose(
    [
        ConvertToRGB(),
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize(mean=mean, std=std),
    ]
)
print(transform_norm)
```

```
Compose(
    <__main__.ConvertToRGB object at 0x7fa100bbda50>
    Resize(size=(224, 224), interpolation=bilinear, max_size=None, antial
    ToTensor()
    Normalize(mean=tensor([0.2992, 0.4125, 0.4588]), std=tensor([0.2697,
)
```

```
norm_dataset = datasets.ImageFolder(root=train_dir,transform=transform_no
print("Image size", norm_dataset[0][0].shape)
print("Label", norm_dataset[0][1])
```

```
Image size torch.Size([3, 224, 224])
Label 0
```

Set up data loaders for both the training and validation data sets. Use the same batch size as before. Remember to set `shuffle=True` on the training loader.

```
train_loader = DataLoader(train_dataset,batch_size=32,shuffle=True)

val_loader = DataLoader(val_dataset,batch_size=batch_size)
```

- Build Model:-

```
model.append(torch.nn.Dropout(p=0.5))
model.append(torch.nn.Linear(in_features=576, out_features=500))
model.append(torch.nn.ReLU())
model.append(torch.nn.Dropout())
model.append(torch.nn.Linear(500, 9))  # 9 output classes

summary(model, input_size=(batch_size, 3, height, width))
```

```
==========================================================================================
Layer (type:depth-idx)                             Output Shape              Param #
==========================================================================================
Sequential                                         [32, 9]                   --
├─Conv2d: 1-1                                      [32, 16, 224, 224]        448
├─ReLU: 1-2                                        [32, 16, 224, 224]        --
├─MaxPool2d: 1-3                                   [32, 16, 56, 56]          --
├─Conv2d: 1-4                                      [32, 32, 56, 56]          4,640
├─ReLU: 1-5                                        [32, 32, 56, 56]          --
├─MaxPool2d: 1-6                                   [32, 32, 14, 14]          --
├─Conv2d: 1-7                                      [32, 64, 14, 14]          18,496
├─ReLU: 1-8                                        [32, 64, 14, 14]          --
├─MaxPool2d: 1-9                                   [32, 64, 3, 3]            --
├─Flatten: 1-10                                    [32, 576]                 --
├─Dropout: 1-11                                    [32, 576]                 --
├─Linear: 1-12                                     [32, 500]                 288,500
├─ReLU: 1-13                                       [32, 500]                 --
├─Dropout: 1-14                                    [32, 500]                 --
├─Linear: 1-15                                     [32, 9]                   4,509
==========================================================================================
Total params: 316,593
Trainable params: 316,593
Non-trainable params: 0
```

Cross Entropy Loss:-

```
loss_fn = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
model.to(device)
# Send the model to the GPU
```

```
Sequential(
  (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU()
  (2): MaxPool2d(kernel_size=4, stride=4, padding=0, dilation=1, ceil_mode=False)
  (3): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (4): ReLU()
  (5): MaxPool2d(kernel_size=4, stride=4, padding=0, dilation=1, ceil_mode=False)
  (6): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (7): ReLU()
  (8): MaxPool2d(kernel_size=4, stride=4, padding=0, dilation=1, ceil_mode=False)
  (9): Flatten(start_dim=1, end_dim=-1)
  (10): Dropout(p=0.5, inplace=False)
  (11): Linear(in_features=576, out_features=500, bias=True)
  (12): ReLU()
  (13): Dropout(p=0.5, inplace=False)
  (14): Linear(in_features=500, out_features=9, bias=True)
)
```

Training Of Dataset

```
# Import the train and predict functions from `training.py`, instead of typing them out!
from training import train, predict
epochs = 10
train(model,optimizer,loss_fn,train_loader,val_loader,epochs=10,device=device)
# Train the model for 10 epochs
```

```
Training:   0%|          | 0/155 [00:00<?, ?it/s]
Scoring:   0%|          | 0/39 [00:00<?, ?it/s]
Epoch: 1, Training Loss: 1.69, Validation Loss: 1.50, Validation accuracy = 0.47
Training:   0%|          | 0/155 [00:00<?, ?it/s]
Scoring:   0%|          | 0/39 [00:00<?, ?it/s]
Epoch: 2, Training Loss: 1.46, Validation Loss: 1.40, Validation accuracy = 0.51
Training:   0%|          | 0/155 [00:00<?, ?it/s]
Scoring:   0%|          | 0/39 [00:00<?, ?it/s]
```

Evaluate Model Performance:-

```
# Compute the probabilities for each validation image
probabilities = predict(model,val_loader,device)
# Get the index associated with the largest probability for each
predictions = torch.argmax(probabilities,dim=1)

print("Number of predictions:", predictions.shape)
```

```
Predicting:   0%|          | 0/39 [00:00<?, ?it/s]
Number of predictions: torch.Size([1236])
```

Testing of dataset:-

```
test_dir = "sea_creatures/test"
test_transforms = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor()
])

test_dataset = test_dataset = datasets.ImageFolder(root=test_dir, transform=test_transforms)

print("Number of test images:", len(test_dataset))

test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)
```

```
Number of test images: 699
```

```
# Predict the probabilities for each test image
test_probabilities = predict(model, test_loader, device=device)

# Get the index associated with the largest probability for each test image
test_predictions = torch.argmax(test_probabilities, dim=1)

print("Number of predictions:", test_predictions.shape)
```

```
Predicting:   0%|          | 0/22 [00:00<?, ?it/s]
Number of predictions: torch.Size([699])
```

Task 1.5.22: Convert the class index to the class name for each test image.

```
test_classes = [classes[i] for i in test_predictions]

print("Number of class predictions:", len(test_classes))
```

Predictions through the validation dataset

```python
import matplotlib.pyplot as plt
import random

# Sample 12 random indices from the test dataset
sample_indices = random.sample(range(len(test_loader.dataset.samples)), 12)

# Create a grid of 4x3 subplots
fig, axes = plt.subplots(4, 3, figsize=(20, 10))

# Iterate over the sampled indices and plot the corresponding images
for ax, idx in zip(axes.flatten(), sample_indices):
    image_path = test_loader.dataset.samples[idx][0]
    img = Image.open(image_path)

    # Display the image on the axis
    ax.imshow(img)
    ax.axis('off')

    # Get the predicted class for this image
    predicted_class = test_classes[idx]

    # Set the title of the subplot to the predicted class
    ax.set_title(f"Predicted: {predicted_class}", fontsize=14)

plt.tight_layout()
```

References

- **RVfpga HarvardX edX** and **RVfpga source code**
- TinyML foundational courses and **TensorFlow Lite Micro documentation.**
- **WorldQuant University** Applied AI Lab content
- RVfpga simulator **Vibodo,Rvfpga Nexys Board,Piplines,Whisper**(Debugging and simulation) and Tracer (**GTK wave** usage), then **whisper for c program files** compilation.
- **Report** and **internal study materials.**