
Project: CNN Acceleration System

EE470 – System-on-Chip (SoC) Design

Kyung Hee University
Electrical Engineering

Fall 2025

Seungkyu Choi (seungkc@khu.ac.kr)

Project Overview

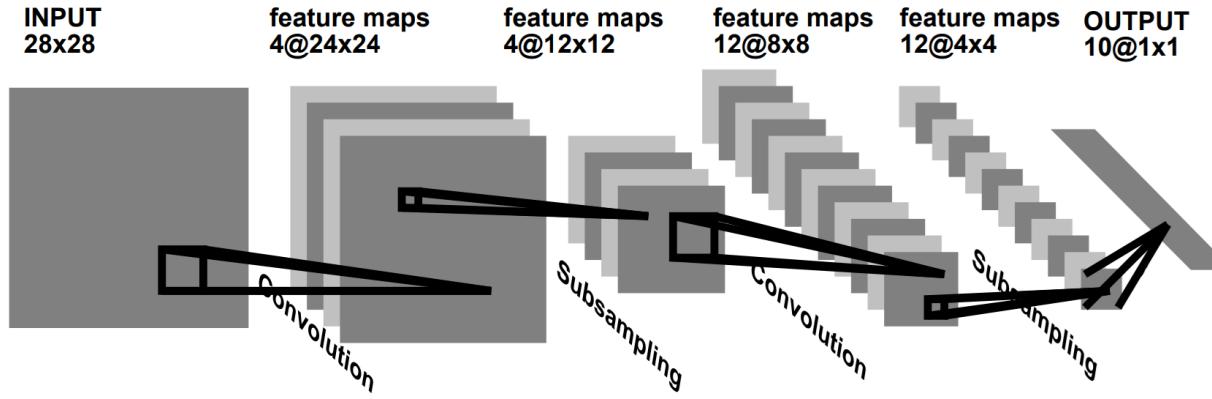
- **Goal:** Build an end-to-end CNN inference system on Zynq-7000
- **Model:** LeNet-1 (modified) on MNIST (28×28 grayscale)
- **Platform:** Arty Z7-20
- **Tools:** Vivado + Vitis (2024.2)
- **Implementation Scope:**

Students will design the PL accelerator, develop a PS-side application, and integrate them through an AXI-based interface.
- **Focus Areas:**

BRAM-based buffering, tiling, and sustained dataflow for efficient computation.
(Target Metrics: Latency / Accuracy / Hardware Utilization)

CNN Model

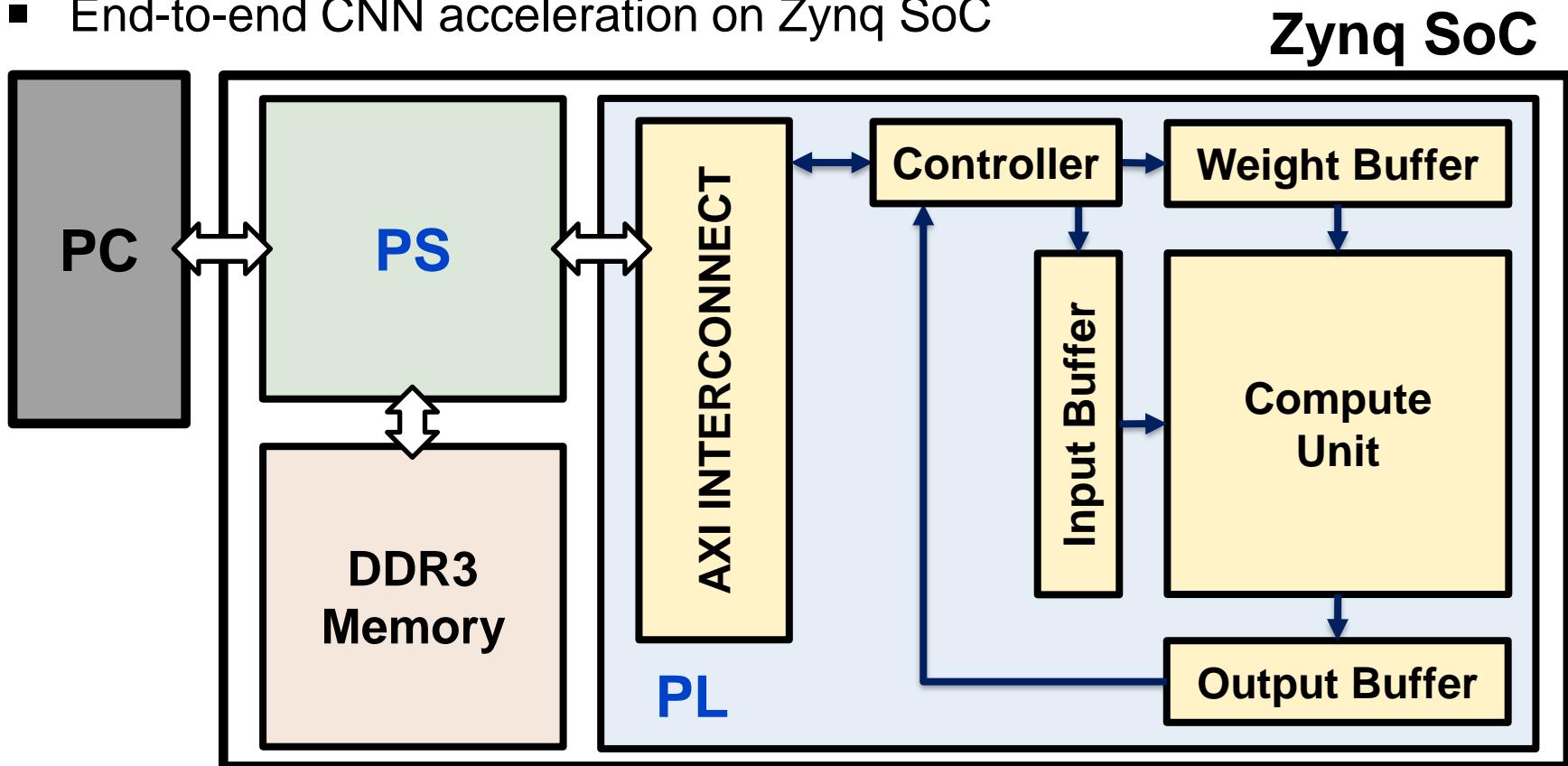
- LeNet-1 (modified) on MNIST(28×28 grayscale, uint8)



- **CONV1** : $4 \times (5 \times 5)$ kernel, stride=1, no padding or bias → **Output**: $4 \times 24 \times 24$
 - **POOL1** : 2×2 max-pool, stride=2 → **Output**: $4 \times 12 \times 12$
 - **CONV2** : $12 \times (5 \times 5)$ kernel, stride=1, no padding or bias → **Output**: $12 \times 8 \times 8$
 - **POOL2** : 2×2 max-pool, stride=2 → **Output**: $12 \times 4 \times 4$
 - **FC** : Flatten ($12 \times 4 \times 4 = 192$), no bias → **Output**: 10 classes
-
- ReLU is used as an activation function after each layer.
 - Quantized Weights(uint8) are provided in .h format for each layer.(CONV1, CONV2, FC)

System Architecture

- End-to-end CNN acceleration on Zynq SoC



- PS (ARM Cortex-A9): Control, data management
- PL (FPGA fabric): CNN computation (CONV/FC)
- Communication: AXI4-Lite

Design Requirements

- Model size must remain unchanged.
(Keep the same given LeNet-1 structure and feature map dimensions.)
- CONV & FC must execute in PL
- Implement BRAM-backed buffering
- Develop a PS-side Vitis application(`main.c`) to control and test the CNN accelerator.
(You are only allowed to add code in the permitted sections.)
- Complete Vivado synthesis/implementation and ensure timing closure.
(Check WNS/TNS.)

Flexibility & Advanced Options

■ Compression techniques & Optimizations

- Quantization
- Pruning
- Word Alignment
- Weight reordering (e.g., transpose, im2col)
- Finetune *(*Include before/after accuracy & training log in your report*)

→ You may **customize** your own weight file.

(Describe the change in file size and your optimization method in the report)

■ **Interface:** You may use AXI-DMA/Stream beyond just AXI4-Lite.

■ **Compute Unit:** The internal compute architecture is open to your design.
(e.g., systolic array, parallel MAC array)

Evaluation Criteria

- **Dataset:**
 - From the MNIST test set (10,000 images), a 1,000-image subset will be provided for development and debugging.
 - The grading subset (another 1,000-image subset) will **not** be provided.
- **Evaluation Metric:**
 - **Latency:** End-to-end latency measured from the first PS-PL transaction to result return.
 - **Accuracy:** Computed on PS using logits from the PL FC output.
Report Top-1 accuracy (%) on the test set.
 - **Hardware Utilization:** LUT / FF / BRAM / DSP usage
(post-implementation report)
- **Note:**

Overall trade-offs among latency, accuracy, and resource efficiency will be well considered.

Design Workflow

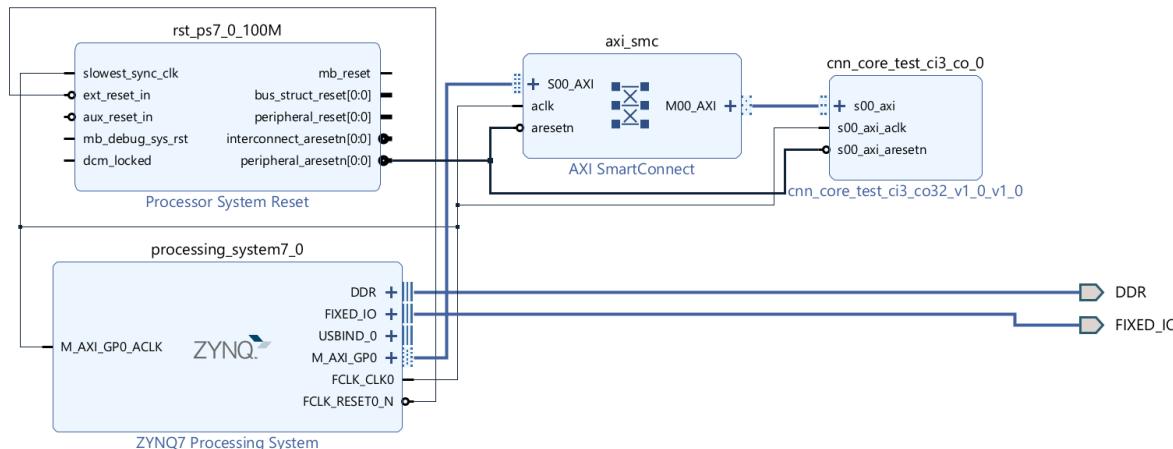
< Vivado: Hardware Design >

1. Create IPs from Verilog RTL modules.

- Develop your CNN accelerator modules (e.g., CONV, FC, Controller).
- Package them as custom IPs.

2. Build a System Block Design

- Instantiate your custom IP along with PS, BRAM, AXI Interconnect, etc.
- Configure address mapping and interface connections.



Design Workflow

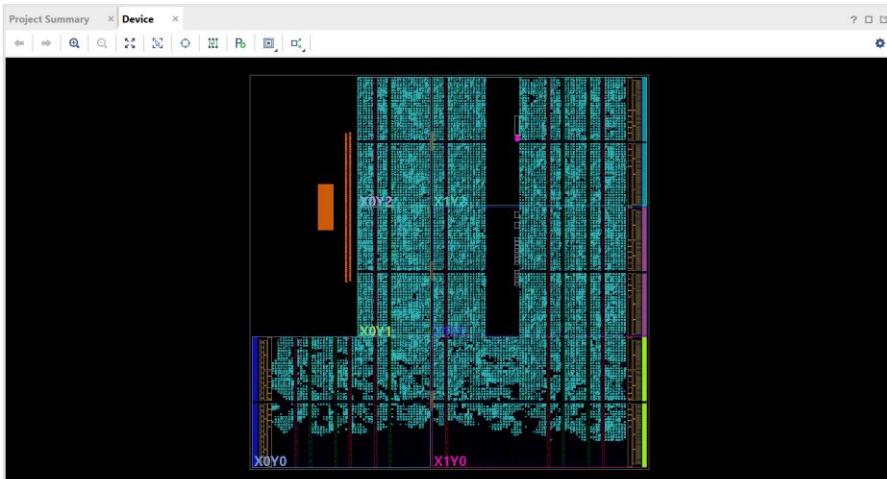
< Vivado: Hardware Design >

3. Generate Bitstream

- Run synthesis and implementation.
- Check **timing, power, and utilization reports**. (WNS/TNS, LUT/FF/DSP/BRAM)

4. Export Hardware (.xsa)

- Create a top-level wrapper and export the design as **.xsa** for Vitis.



[Device View]

이름	수정한 날짜	유형	크기
SoC_prj_a.cache	2025-10-28 오후 3:12	파일 폴더	
SoC_prj_a.gen	2025-10-28 오후 3:20	파일 폴더	
SoC_prj_a.hw	2025-10-28 오후 3:12	파일 폴더	
SoC_prj_a.ip_user_files	2025-10-28 오후 3:12	파일 폴더	
SoC_prj_a.runs	2025-10-28 오후 3:21	파일 폴더	
SoC_prj_a.sim	2025-10-28 오후 3:12	파일 폴더	
SoC_prj_a.srcs	2025-10-28 오후 3:15	파일 폴더	
design_1_wrapper.xsa	2025-10-28 오후 4:44	XSA 파일	1,385KB
SoC_prj_a	2025-10-28 오후 3:41	Vivado Project File	24KB

[.xsa file]

Design Workflow

< Vivado: Hardware Design >

Util.
Report

Power
Report

Timing
Report

The screenshot displays three main reports from the Vivado Design Suite:

- Utilization Report:** Shows resource usage across various logic blocks. The table includes columns for Name, Slice LUTs (53200), Slice Registers (106400), F7 Muxes (26600), F8 Muxes (13300), Slice (13300), LUT as Logic (53200), LUT as Memory (17400), Bonded IOPADS (130), and BUFCTRL (32). The hierarchy tree on the left shows "design_1_wrapper" and "design_1_i".
- Power Report:** Provides power analysis details. It shows Total On-Chip Power (1.567 W), Design Power Budget (Not Specified), Process (typical), and Power Budget Margin (N/A). The "On-Chip Power" section breaks down power consumption by category: Dynamic (91%), Clocks (3%), Signals (4%), Logic (5%), PS7 (88%), and Device Static (9%). Junction Temperature is listed as 43.1°C.
- Timing Report:** Shows the Design Timing Summary. It includes General Information, Timer Settings, and sections for Setup, Hold, and Pulse Width. Key timing values include Worst Negative Slack (WNS) at 0.102 ns, Worst Hold Slack (WHS) at 0.084 ns, and Worst Pulse Width Slack (WPWS) at 4.020 ns. The report also states "All user specified timing constraints are met."

Design Workflow

< Vitis: Software Integration & Testing >

1. Import Hardware Platform (.xsa)

- Create a new workspace with the exported hardware design as the base platform.

2. Develop Application Project

- Use the provided `main.c` skeleton to control your PL accelerator via AXI.
- Measure accuracy and latency, and compare them with the software implementation on PS.

3. Run on Hardware (Serial Output)

- Connect via UART/Serial Port to monitor execution.
- Display classification accuracy and total inference latency on terminal.

Design Workflow

< Vitis: Software Integration & Testing >

The screenshot shows the Vitis Explorer interface with a red box highlighting the project tree and another red box highlighting the terminal window.

VITIS EXPLORER

- SOC_PRJ_D_VITIS - VITIS COMPONENTS
 - A_SoC_prj_d [Application] (highlighted by a red box)
 - Settings
 - Includes
 - Sources
 - src
 - cnn_common.h
 - conv1_arr.h
 - conv2_arr.h
 - fc1_arr.h
 - Iscript.ld
 - main.c (highlighted by a green bar)
 - platform.c
 - platform.h
 - test_images_arr.h
 - test_labels_arr.h
 - Output
 - P_cnn_m [Platform] (highlighted by a red box)

Terminal Output:

```
Opened with baud rate: 115200
***** SoC CNN Acceleration System *****
Press '1' Start Test
Press '2' to exit
Selection:1

[PS vs PL E2E CNN] N_TEST=1000
>>> CNN Running in PS...
>>> Loading Weights on PL...
>>> CNN Running in PL...

==== Summary ====
PS Acc = 986/1000 = 98.60% | Total = 8665773.53 us | Avg/img = 8665.77 us
PL Acc = 986/1000 = 98.60% | Weight upload = 240.80 us
PL Inference only: Total = 1018604.20 us | Avg/img = 1018.60 us
PL Cold-start total (upload + infer) = 1018845.00 us | Avg/img (including upload) = 1018.85 us
Speedup (PS/PL, steady-state avg) = 8.51x
```

[Test Result]

*Shown for illustration only
(NOT actual result)

[Vitis Components]

Fine-tuning Guideline

■ MNIST Dataset Access for Fine-tuning (Option)

- **MNIST:** A dataset of 28x28 grayscale images of handwritten digits (0-9)
- **How to Download**

```
from torch vision import datasets  
Import torchvision.transforms  
  
train_ds = datasets.MNIST(  
    root="./my_root",  
    train=True,  
    transform=~~,  
    download=~~)
```

- **Root:** where to store MNIST locally
 - **Train:** True = training set, False = test set
 - **Transform:** preprocess per image (e.g. ToTensor(), PILToTensor())
 - **Download:** download if not already present
- ❖ **Reference:** MNIST docs
(<https://docs.pytorch.org/vision/main/generated/torchvision.datasets.MNIST.html#torchvision.datasets.MNIST>)

Submission Rule (Due: Dec 14 23:59)

- Submission file: Zip File *File Name: prj_xx.zip (xx = team number 01 ~ N)*
 - Complete SoC Project Package
 - Entire Vivado project folder (including .xpr)
 - Vitis **main.c** file
 - Weight files (.h) *(if modified)*
 - PPT & Report (.pdf)
 - Should describe overall system architecture, implementation details, and control flow.
 - Presentation time: 10 - 12 minutes.
 - Report length: under 10 pages. *(Korean allowed)*