숭실대학교 Soongsil University

R E P O R T

[HW Based Convolution & Video Filter]

• 과 목 명	:	고급 Project
• 담 당 교 수	:	송인철
• 제 출 일	:	2025 년 12 월 14 일
• 학 과	:	전자공학과
• 학 번	:	20202840, 20201641 ,20222947
• 팀 명	:	5 조

숭실대학교 Soongsil University

목 차

1. Introduction

1.1. Overview

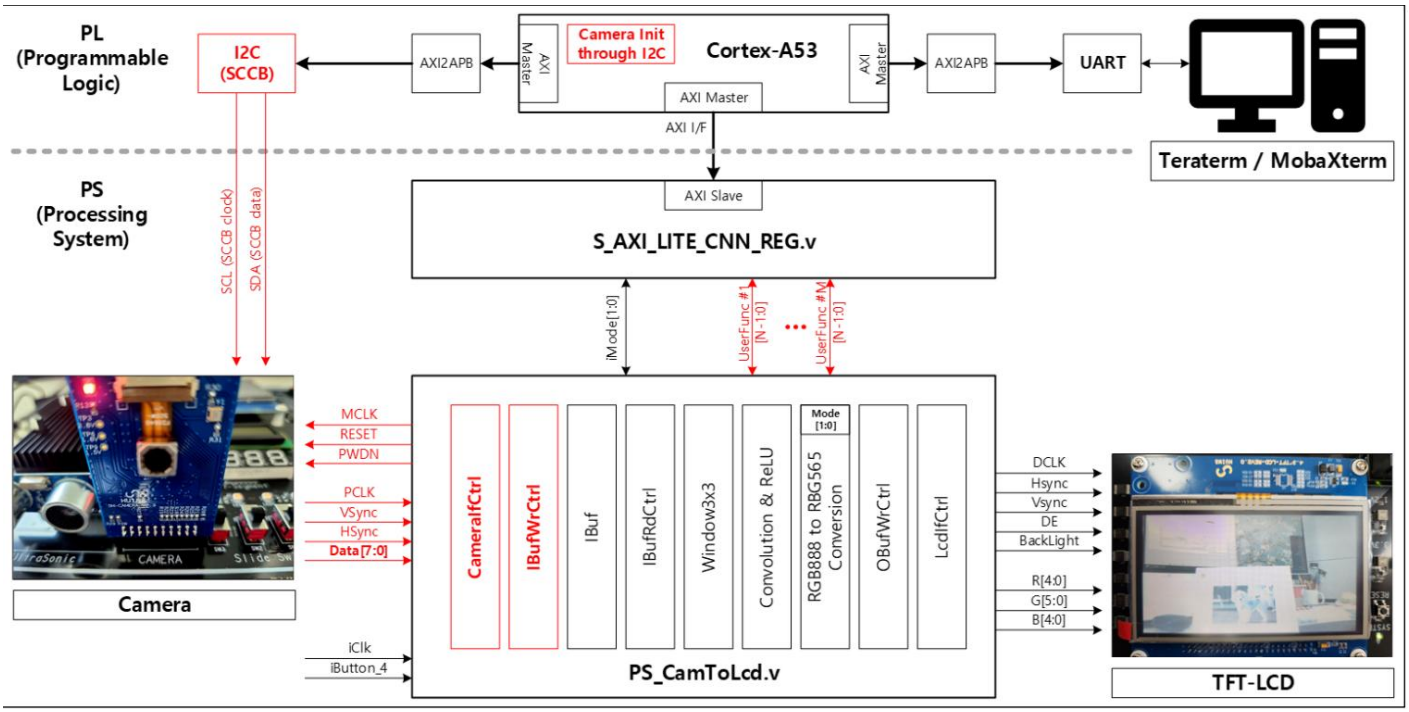
- 본 프로젝트는 FPGA 환경에서 OV5640 카메라를 통해 입력되는 영상을 실시간으로 처리하는 CNN 하드웨어를 구현하는 것입니다.

시스템은 카메라 센서로부터 들어오는 480x272 해상도의 RGB888 스트림 데이터를 캡처하여 입력 버퍼에 저장하는 것으로 시작됩니다. 연속적인 프레임 입력에 대응하기 위해, 저장된 데이터는 지연 없이 윈도우 생성 로직으로 전달되어 3x3 픽셀 단위로 정렬되고 Convolution 및 ReLU 연산 모듈을 거칩니다. 연산이 완료된 데이터는 즉시 RGB565 포맷으로 변환되어 출력 버퍼에 저장된 뒤, TFT-LCD 컨트롤러를 통해 디스플레이에 시각화됩니다

1.2. Design Plan

- 설계 전 예상 구조

*Overall Architecture



*Sub Block Architecture

- Clock Gen: 100MHz -> 12.5MHz Enable Generation
- Clk gen2 : 100MHz -> 25.MHz Enable Generation
- Camera Ctrl & Interface : I2C 설정, 픽셀 수신/패킹, RGB 변환
- Window Generator: Inbuf 에서 픽셀을 읽어와 3x3 윈도우 구성.
- Convolution & ReLU: 3x3 Conv & Relu 연산 [Kernerl : Sharpen, Edge Enhance, bypass].
- Pixel Conversion: MSB 추출 [24bit -> 16bit]
- Mem. To TFT-LCD: RGB565 포맷의 Data 를 LCD 데이터 시트에 맞게 설계 후 출력

- IO List:

  - Input: iClk (100MHz), iRst_n, iMode, Cam_PCLK, CAM_VSYNC, CAM_HSYNC, CAM_DATA, CAM_SCCB_SCL, CAM_SCCB_SDA
  - Output: oLcdClk, oLcdHSync, oLcdVSync, oLcdDe, oLcdR/G/B, CAM_MCLK, CAM_RESETn, CAM_PWDN

- Memory List & Size :

  - Input Buffer: 480x272 pixel , RGB888 (24bit). Vivado Simple Dual Port BRAM IP.
  - Output Buffer: 480x272 pixel, RGB565 (16bit). Vivado Simple Dual Port BRAM IP.

1.3 Job Parition

한정호 : Top Module, Pixel Conversion, Clock Generation, TestBench 설계, cam_i2c 통합

노영찬 : Window Function, Address Control Logic, Mem to TFT-LCD 설계

전민서 : Conv & Relu, SOC 구조 설계, SW Control 설계, Cam to Inbuf Interface 설계

1.4 Project Schedule

Week1 : 데이터 시트 학습 및 Sub Module 구현

Week 2 : CNN HW : Top Module 구현 및 SOC 구현

Week 3 : SW 제어 구현

Week 4 : 카메라 모듈 및 카메라 인터페이스 구현

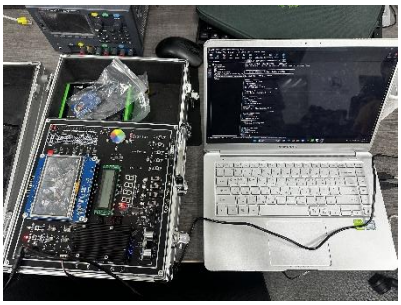
week 5 : 카메라 통합 Top Module, SOC, SW 제어 구현

2. Design

2.1 Development Environment

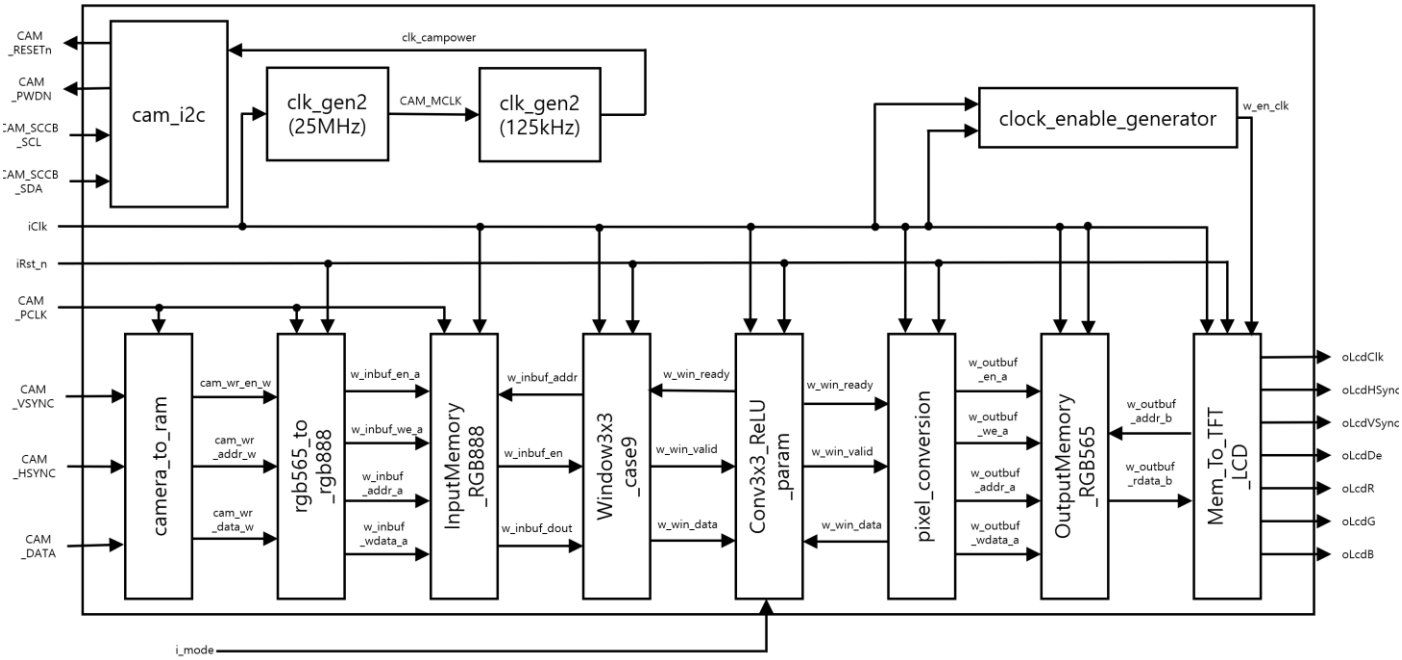
본 프로젝트를 수행하기 위해 사용된 하드웨어 및 소프트웨어 환경입니다.

- OS: Windows 11
- EDA Tool: Xilinx Vivado 2024.1 / Xilinx Vitis 2024.1
- Use Tool : Mobaxterm
- Target Board: Avnet Ultra96-V2 (Xilinx Zynq UltraScale+ MPSoC ZU3EG)
- Language: Verilog, C



2.2 Overall architecture

[a] Block Diagram



강의자료 구조를 인용하여 탑 다이어그램을 설계하였습니다.

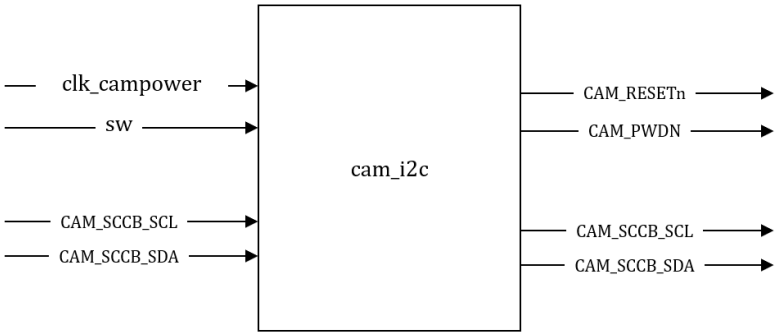
[b] I/O

Signal	Source	Description
iClk	External	100MHz System Clock
iRst_n	External	Active-Low System Reset
iMode	External [SW]	Filter Mode Select [00: Sharpen, 01: Edge, 10/11: Bypass]
oLcdClk	LCD Controller	LCD Pixel Clock [6.25MHz]
oLcdHSync	LCD Controller	Horizontal Sync Signal
oLcdVSync	LCD Controller	Vertical Sync Signal
oLcdDe	CNN_Top	Data Enable [Always High]
oLcdBacklight	CNN_Top	Backlight Control [Always High]
oLcdR/G/B	LCD Controller	LCD Data [RGB565]
CAM_PCLK	Camera	Camera Pixel Clock
CAM_VSYNC	Camera	Camera Vertical Sync Signal
CAM_HSYNC	Camera	Camera Horizontal Sync Signal
CAM_DATA	Camera	Camera Pixel Data Input
CAM_RESETn	Cam_i2c	Camera Reset Signal
CAM_PWDN	Cam_i2c	Camera Power Down Signal
CAM_SCCB_SCL	Cam_i2c	Camera I2C Clock (SCCB Interface)
CAM_SCCB_SDA	Cam_i2c	Camera I2C Data (SCCB Interface)

2.3 Sub-block diagram & Description

[1] Camera SCCB Controller (cam_i2c)

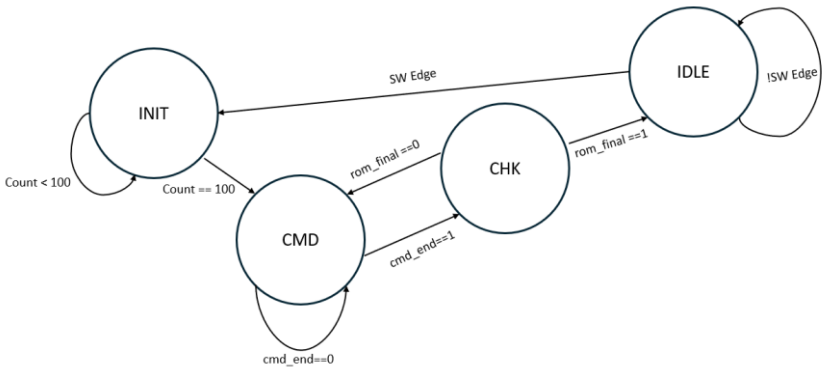
* Function: 카메라의 초기화 시퀀스를 제어하고, 내부 ROM 에 저장된 레지스터 설정값들을 SCCB(I2C) 프로토콜을 통해 순차적으로 전송하여 카메라의 해상도, 포맷, 컬러 매트릭스 등을 설정한다.



Signal	Direction	Description
clk_campower	Input	Clk gen2's Clock
sw	Input	SW control [Always High]
CAM_RESETn	Output	Camera Reset Signal (Active Low)
CAM_PWDN	Output	Camera Power Down Signal (Active High)
CAM_SCCB_SCL	Inout	SCCB Serial Clock
CAM_SCCB_SDA	Inout	SCCB Serial Data

Structure

- State Machine (FSM):



  - INIT: 카메라의 하드웨어 리셋(cam_rst_no)과 파워다운(cam_pwdn) 핀을 제어하여 센서 ON.
  - CMD (Command): ROM 에서 읽어온 데이터(레지스터 주소+값)를 cam_i2c_command 모듈에 전달하고 전송을 시작(sig_i2c_command_en)한다.
  - CHK (Check): 하나의 명령어 전송이 완료되면(sig_i2c_command_end), 다음 ROM 주소로 이동한다. 만약 마지막 데이터(sig_i2c_rom_final)라면 IDLE 로 이동한다.
  - IDLE: 설정이 완료된 상태. sw 입력의 변화(Edge)를 감지하면 다시 INIT 상태로 돌아가 재설정을 수행한다.

Sub-Submodules

2. I2C ROM (cam_i2c_rom)

- Function: 카메라 초기화에 필요한 레지스터 설정값 목록(Look-up Table)을 저장.
- Logic:
  - sw1_i = 1 입력 받아 OV5640 모드 설정.
  - 주소(i2c_rom_addr_i)를 입력 -> 32 비트 데이터(i2c_rom_data_o)를 출력.
  - 마지막 데이터 주소에 도달 -> i2c_rom_final_o = 1 출력.
  -

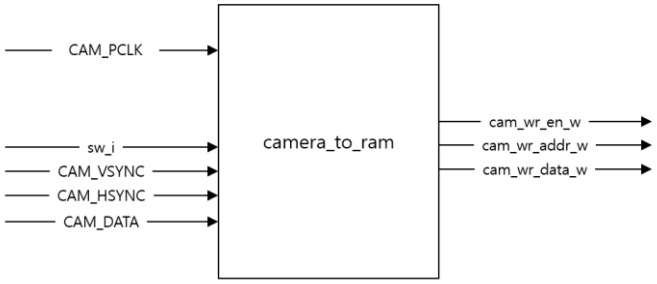
[31:24]	[23:16]	[15:8]	[7:0]
0x78	0x30~0x50	0x00~0xFF	0x00~0xFF
Fixed Value *(I2C Write)*	Register Address *(High Byte)*	Register Address *(Low Byte)*	Data Value *(8-bit)*

3. I2C Command Driver (cam_i2c_command)

- Function: 상위 모듈로부터 받은 데이터를 물리적인 SCCB(I2C) 프로토콜 타이밍에 맞춰 SCL 과 SDA 신호로 변환.
- Logic:
  - i2c_cmd_en_i = High -> 전송 시작.
  - sw1_i = 1 입력 받아 OV5640 모드 설정.
  - OV5640 Mode: (Device Addr -> Reg Addr High -> Reg Addr Low -> Data)을 수행하며, 카운터(sig_count) 0~113 동안 각 비트의 SCL/SDA 타이밍을 생성한다.
  - 전송 완료 -> i2c_cmd_end_o =1.

[2] Camera Interface (camera_to_ram)

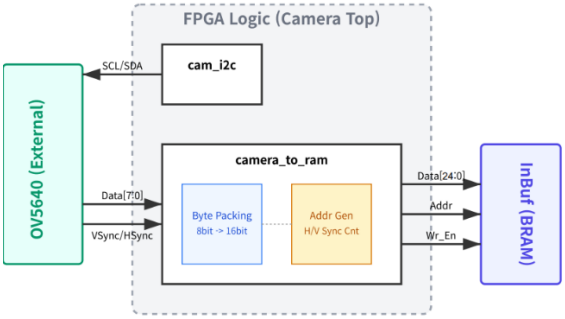
* Function: 카메라로부터 8 비트 병렬 데이터와 동기 신호(VSYNC, HSYNC)를 수신하여, 8 비트 데이터를 16 비트(RGB565) 포맷으로 패킹(Packing)하고 외부 RAM 에 저장하기 위한 주소(Address) 및 쓰기 제어 신호(Write Enable)를 생성한다



*I/O Description

Signal	Direction	Description
clk_i	Input	Camera Pixel Clock (PCLK)
sw_i	Input	Switch Input (Unused in logic)
cam_vsync_i	Input	Camera Vertical Sync (Frame Sync)
cam_hsync_i	Input	Camera Horizontal Sync (Line Sync)
cam_data_i	Input	Camera 8-bit Data Input
ram_wr_en_o	Output	RAM Write Enable
ram_wr_addr_o	Output	RAM Write Address (Linear)
ram_wr_data_o	Output	RAM Write Data (16-bit RGB565)

Structure



1. Input Synchronization:
    - 입력 신호(cam_vsync_i, cam_hsync_i, cam_data_i)를 안정적인 처리를 위해 clk_i 에 맞춰 레지스터링(Delay)한다.
2. Byte Packing Logic (8-bit to 16-bit):
    - sig_temp 레지스터를 사용하여 바이트 순서를 구분한다.
    - Phase 0 (sig_temp=0): 첫 번째 바이트(High Byte)를 수신하여 상위 8 비트([15:8])에 임시 저장한다.
    - Phase 1 (sig_temp=1): 두 번째 바이트(Low Byte)를 수신하여 하위 8 비트([7:0])에 병합하고, RAM 쓰기 신호(sig_ram_wr_en)를 활성화한다.

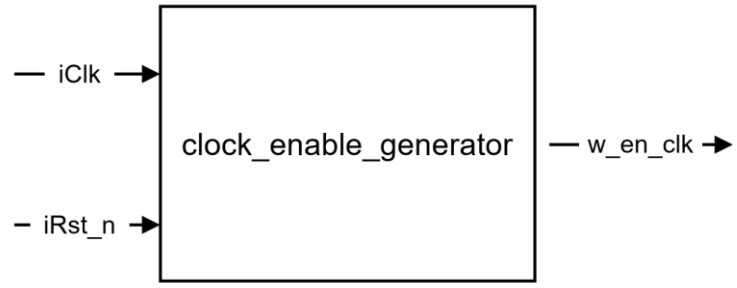
3. Coordinate & Address Generation:

   - Valid Area Check: sig_v_count(< 272)와 sig_h_count(< 480)가 유효 해상도 범위 내에 있을 때만 데이터 쓰기를 허용한다 (sig_en).
   - Address Counter: 유효 데이터가 써질 때마다(sig_ram_wr_en High) 주소(sig_addr_count)를 1 씩 증가시킨다.
   - Reset: cam_vsync_i 가 High(Frame Blanking)일 때 주소 카운터를 0 으로 초기화하여 다음 프레임을 준비한다.

[a] Clock Enable Generator (clock_enable_generator)

* Function: 100MHz 입력을 받아 8 분주하여 약 12.5MHz 의 Enable 펄스를 생성한다.

*Block Diagram



*I/O Description

Signal	Source	Description
iClk	External	100MHz System Clock
iRst_n	External	Active-Low System Reset
w_en_clk	Output	12.5MHz Clock Enable Pulse

* Structure: 3 비트 카운터를 사용하여 0~7 를 카운트, 7 일 때 1 사이클 펄스를 출력한다.

[b] Programmable Clock Generator (clk_gen2)

* Function: 입력 클럭(clk_i)을 받아 외부에서 설정한 16 비트 카운트 값(count_i)에 따라 분주된 클럭을 생성한다. 내부 운터가 설정값에 도달할 때마다 출력 신호를 반전(Toggle)시켜 50% 듀티비를 가진 분주 클럭을 출력한다.

*Block Diagram

*I/O Description

Signal	Source	Description
clk_i	External	100MHz System Clock
count_i	Top Module	Frequency Control Value (Divider Ratio Setting)
clk_o	Output	Divided Clock Output (Toggles when count match)

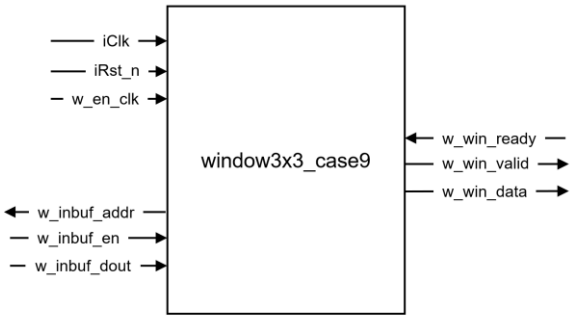
* Structure

1. Counter Logic: 16 비트 레지스터 sig_count 가 매 clk_i 상승 엣지마다 증가한다.
2. Compare & Reset: sig_count 가 입력받은 count_i 값과 같아지면 sig_count 는 0 으로 초기화된다.
3. Output Generation: 동시에 카운터가 초기화되는 시점에서 출력 레지스터 sig_clk_out 의 값을 반전(Invert)시켜 클럭을 생성한다.

[b] Window Generator (window3x3_case9)

* Function: 입력 이미지(BRAM)로부터 픽셀 데이터를 순차적으로 읽어와 Zero Padding 을 적용한 후, 3x3 크기의 슬라이딩 윈도우(Sliding Window) 데이터를 출력한다. 이미지 처리 블록에 9 개의 픽셀 데이터를 병렬로 공급하는 역할을 한다.

*Block Diagram

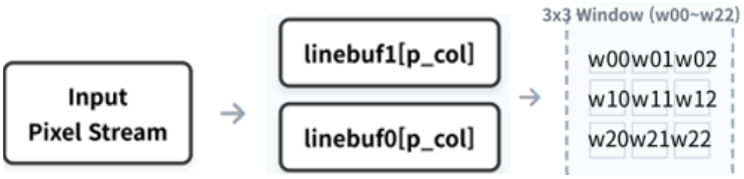


*I/O Description

Signal	Direction	Description
iClk / iRst_n	Input	System Clock & Reset
w_en_clk	Input	Clock Enable Signal
w_inbuf_addr	Output	Address to Input Memory (BRAM)
w_inbuf_en	Output	Read Enable to Input Memory
w_inbuf_dout	Input	Data from Input Memory (24-bit RGB)
w_win_ready	Input	Handshake Ready from Conv Module
w_win_valid	Output	Handshake Valid to Conv Module
w_win_data	Output	3x3 Window Pixel Data (24bit * 9)

* Structure

- Line Buffer

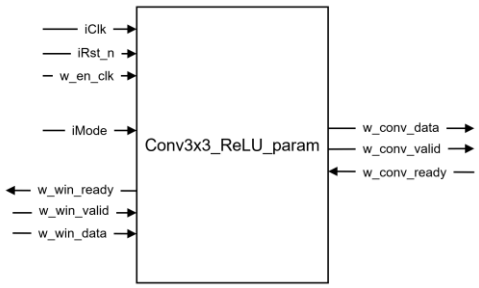


1. Stage 1: Read Request (Address Generation)
    - Padding Coordinates: 실제 이미지보다 가로/세로 2 픽셀씩 더 큰 영역(0~481, 0~273)을 스캔한다.
    - Address Control: 현재 좌표(r_col, r_row)가 유효 이미지 영역(1~480, 1~272) 내부일 때만 o_bram_en 을 활성화하고 BRAM 주소를 증가시킨다. 영역 밖(가장자리)은 패딩 구간으로 처리한다.
    - Frame Loop: 프레임 끝에 도달-> addr= 0 -> 다음 프레임을 시작한다.
2. Stage 2: Data Fetch & Window Shift
    - Line Buffers: 2 개의 라인 버퍼(linebuf0, linebuf1)를 사용하여 이전 행들의 데이터를 저장한다.
    - Shift Operation:
        - Input -> LineBuf1 -> LineBuf0 (수직 이동)
        - w22 <- w21 <- w20 (수평 이동, 레지스터 시프트)
    - Zero Padding: p_is_padding 신호에 따라 BRAM 데이터 대신 0(Black)을 주입한다.
    - Output Mapping: 3x3 레지스터(w00~w22)를 묶어 o_data 로 출력한다.

[c] Convolution & ReLU (Conv3x3_ReLU_param)

* Function: 3x3 윈도우 픽셀과 커널 계수에 대한 Convolution 연산 수행 후, ReLU 활성화 함수를 적용한다.

*Block Diagram

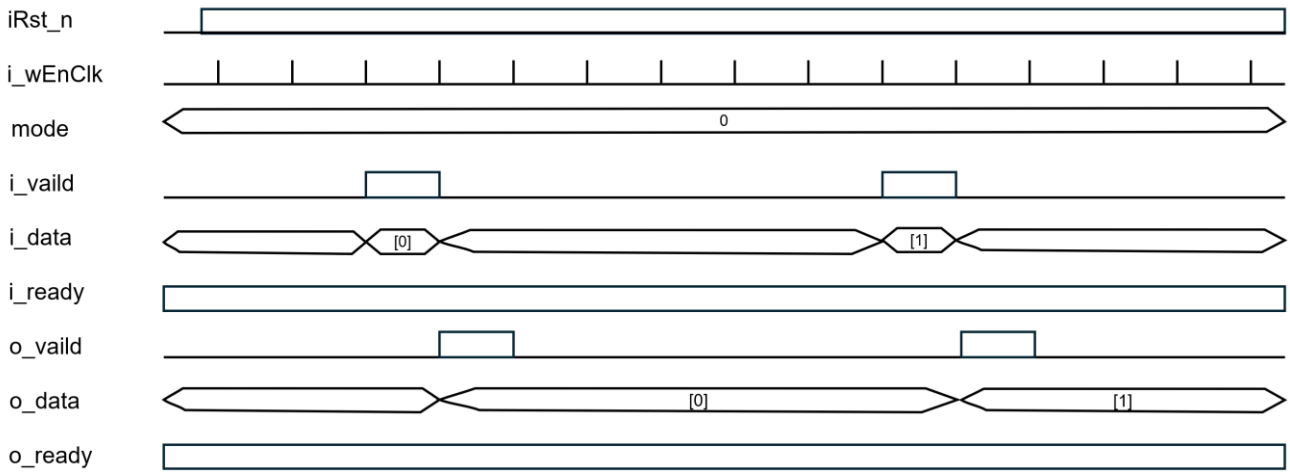


*I/O Description

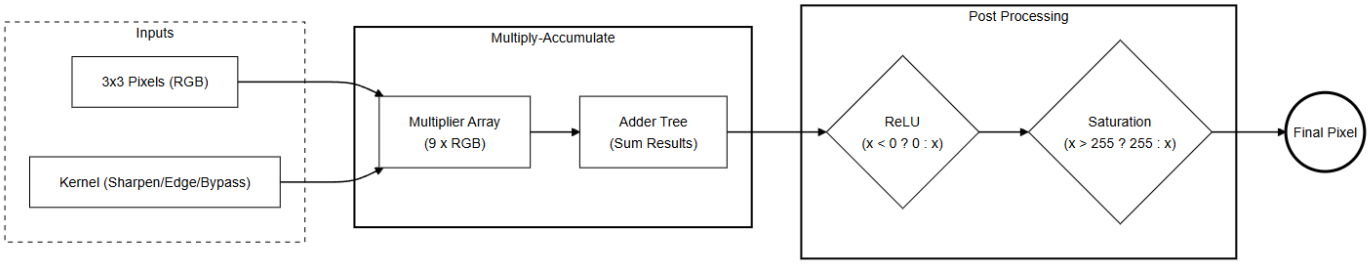
Signal	Direction	Description
iClk / iRst_n	Input	System Clock & Reset
w_en_clk	Input	Clock Enable Signal
iMode	Input	Filter Mode Select (Sharpen/Edge/Bypass)

w_win_data	Input	3x3 Window Data from Window Gen
w_win_valid	Input	Handshake Valid from Window Gen
w_win_ready	Output	Handshake Ready to Window Gen
w_conv_data	Output	Convolved Pixel Data (24-bit RGB)
w_conv_valid	Output	Handshake Valid to Pixel Conv
w_conv_ready	Input	Handshake Ready from Pixel Conv

*Timing Diagram



* Structure: 3 개의 채널(R, G, B)에 대해 병렬 연산을 수행한다. relu_sat8_long 함수를 통해 0~255 범위 제어



* Kernel

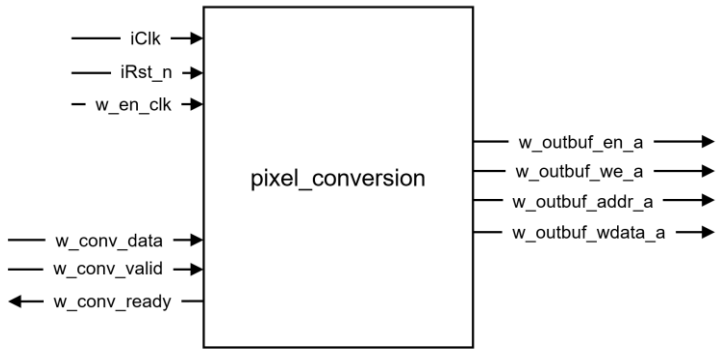
Sharpen	Edge Enhance	Bypass
0 −1 0 −1 5 −1 0 −1 0	−1 −1 −1 −1 9 −1 −1 −1 −1	0 0 0 0 1 0 0 0 0

* Relu : $\mathrm{ReLU}(x) = \max(0, x)$

[d] Pixel Conversion (pixel_conversion)

* Function: RGB888(24bit) 데이터를 RGB565(16bit)로 변환하고, 출력 메모리 주소를 생성한다.

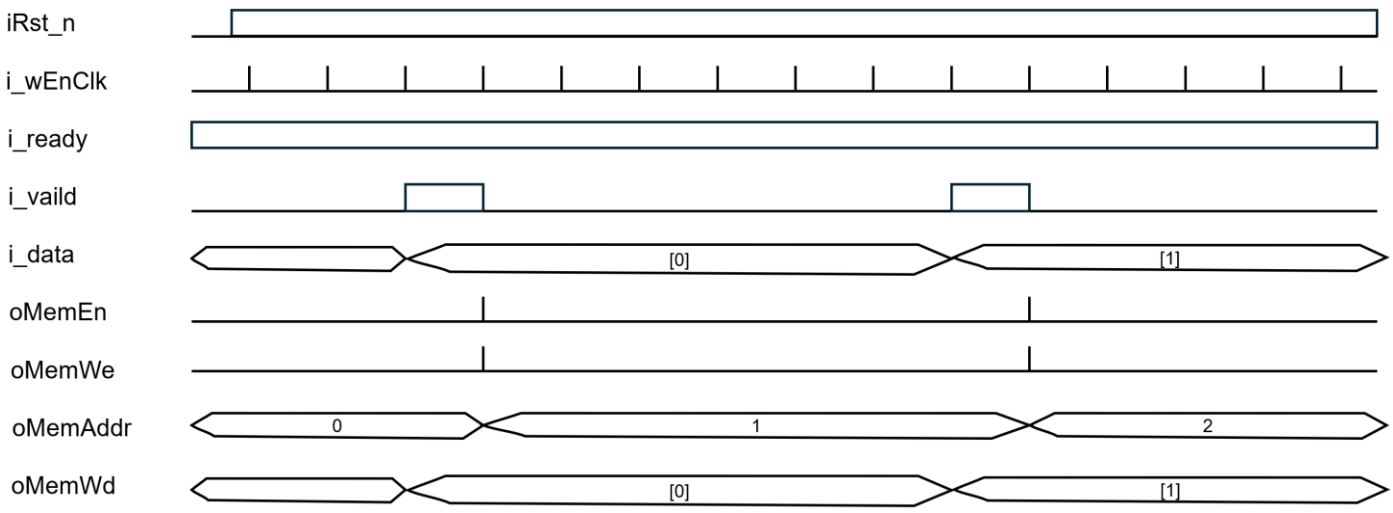
*Block Diagram



*I/O Description

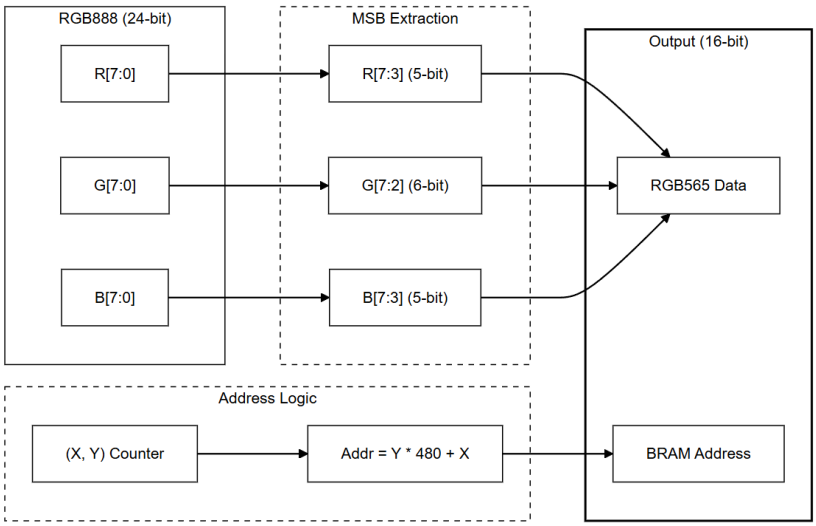
Signal	Direction	Description
iClk / iRst_n	Input	System Clock & Reset
w_en_clk	Input	Clock Enable Signal
w_conv_data	Input	RGB888 Data from Conv Module
w_conv_valid	Input	Handshake Valid from Conv Module
w_conv_ready	Output	Handshake Ready to Conv Module
w_outbuf_en_a	Output	Enable Signal to Output Memory
w_outbuf_we_a	Output	Write Enable to Output Memory
w_outbuf_addr_a	Output	Write Address to Output Memory
w_outbuf_wdata_a	Output	Write Data (RGB565) to Output Memory

*Timing Diagram



* Struture

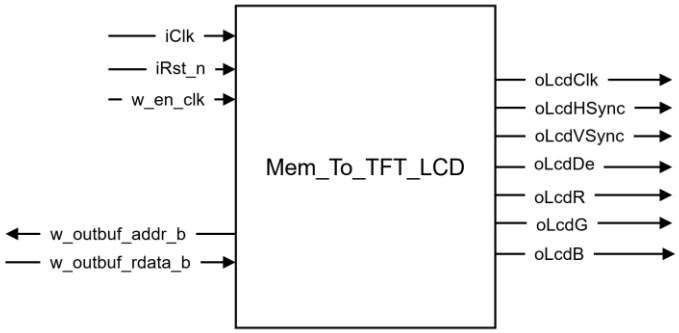
- MSB 추출, Outbuf 에 Address 전달.



[e] LCD Controller (Mem_To_TFT_LCD)

* Function: 출력 메모리에서 픽셀을 읽어 TFT-LCD 타이밍 신호와 함께 출력한다.

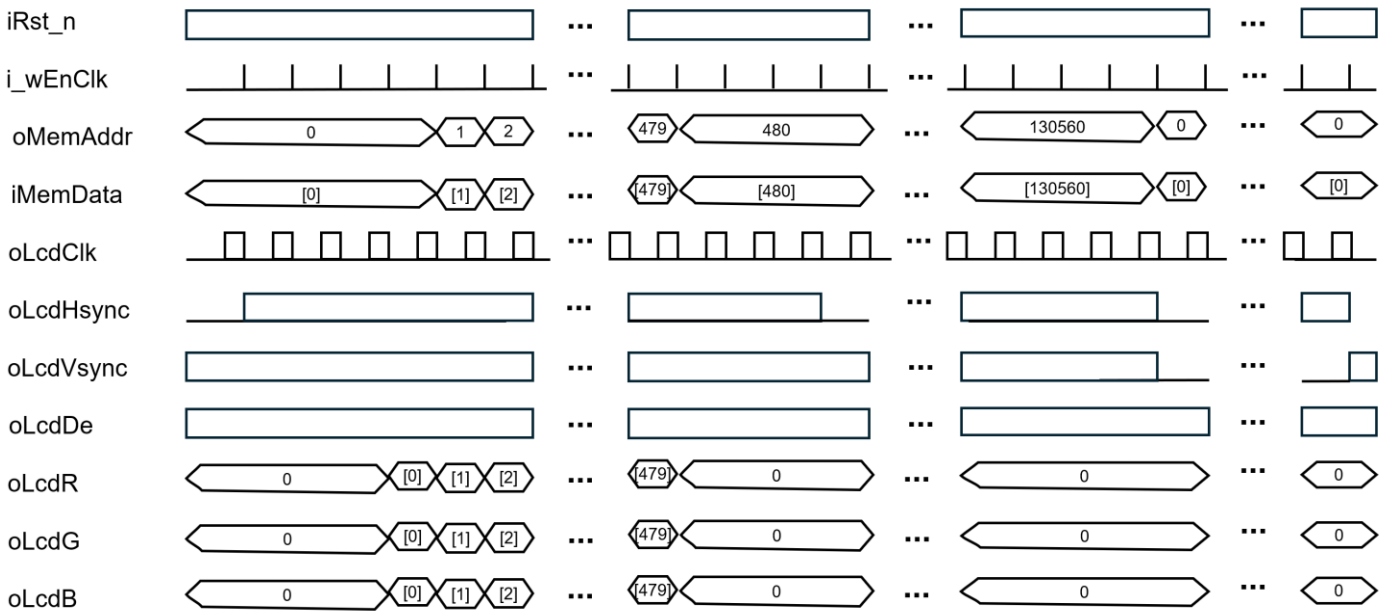
*Block Diagram



*I/O Description

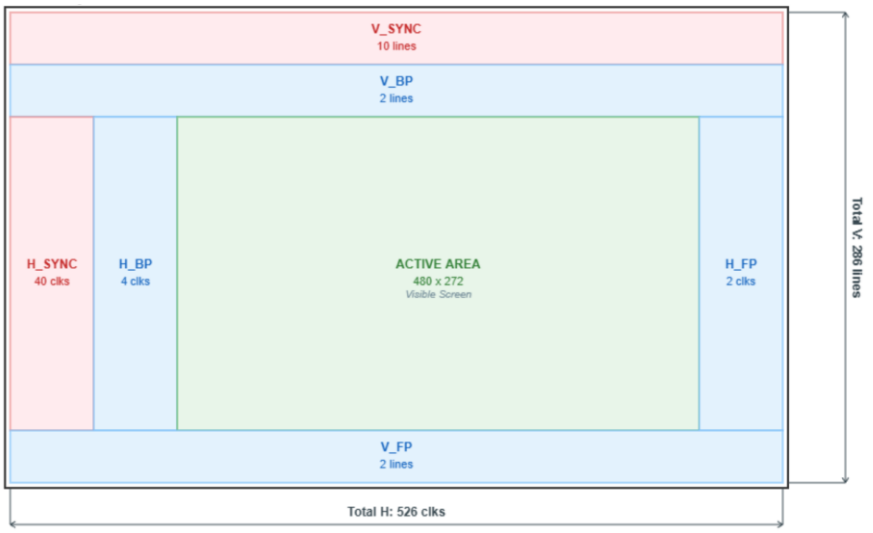
Signal	Direction	Description
iClk / iRst_n	Input	System Clock & Reset
w_en_clk	Input	Clock Enable Signal
w_outbuf_addr_b	Output	Read Address to Output Memory
w_outbuf_rdata_b	Input	RGB565 Data from Output Memory
oLcdClk	Output	LCD Pixel Clock
oLcdHSync	Output	Horizontal Sync
oLcdVSync	Output	Vertical Sync
oLcdDe	Output	Data Enable
oLcdR, oLcdG, oLcdB	Output	LCD Pixel Data

*Timing Diagram

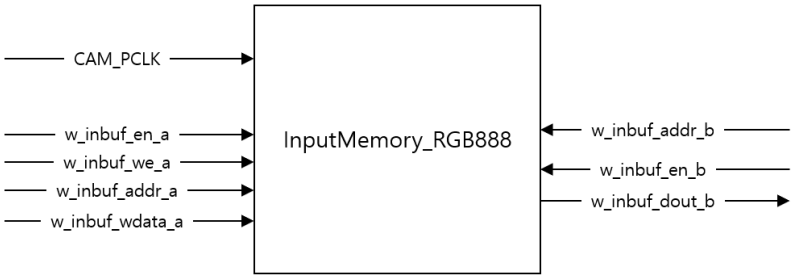


*Structure

- Data sheet, 코드 기반으로 구현한 LCD area Diagram 이다.

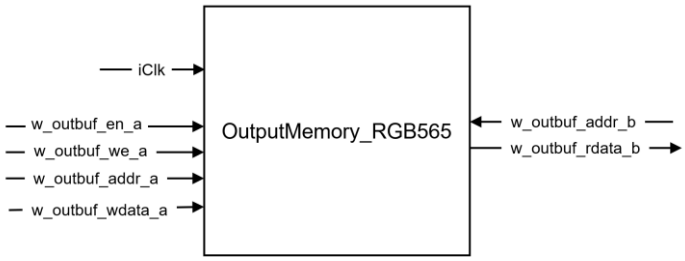


2.4 Memory List & Size



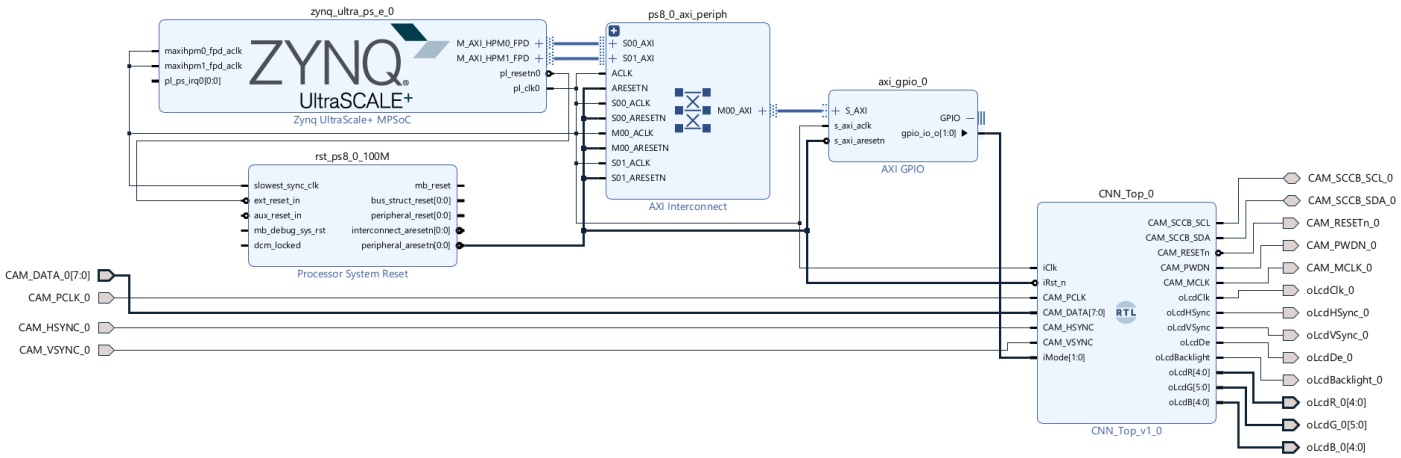
1. Input Memory (InBuf)
    - Usage: Store original image (RGB888)

- Size: 480 * 272 pixel (Address Width: 17bit)
- Data Width: 24 bit
- Type: Simple Dual Port RAM (Port A: Write from CAMERA, Port B: Read to Window)



2. Output Memory (OutBuf)
   - Usage: Store processed result image (RGB565)
   - Size: 480 * 272 pixel (Address Width: 17bit)
   - Data Width: 16 bit
   - Type: Simple Dual Port RAM (Port A: Write from CNN, Port B: Read to LCD)

2.5 SOC



1. Zynq UltraScale+ MPSoC (zynq_ultra_ps_e_0) : 시스템의 메인 프로세서, PL 영역에 클럭, 리셋 신호 공급.
   - pl_clk0: CNN_Top_0 및 주변 IP 구동을 위한 기준 클럭(100MHz)을 생성하여 공급한다.
   - pl_resetn0: 시스템 전체의 Active-Low 리셋 소스로 사용되며, proc_sys_reset_0 블록을 제어한다.
   - M_AXI_HPM0_FPD: AXI Interconnect 를 통해 PL 영역의 AXI Slave 주변 장치(axi_gpio_0)를 제어한다.
2. Control Path (axi_gpio_0 & ps8_0_axi_periph)
   - AXI Interconnect (ps8_0_axi_periph): Zynq PS 의 AXI Master <-> GPIO IP 의 AXI Slave 연결.

- AXI GPIO (axi_gpio_0): PS 에서 GPIO 레지스터 값을 설정, gpio_io_o[1:0] 를 통해 2bit 신호가 출력.
- Mode Control: axi_gpio_0 의 출력 <-> CNN_Top_0 의 iMode 연결.

3. Processor System Reset (proc_sys_reset_0)
   - PS 의 리셋 신호를 시스템 클럭에 동기화하여 안정적인 리셋(peripheral_aresetn) 신호를 주변 모듈에 분배.
4. CNN Accelerator (CNN_Top_0)
   - 설계한 Verilog RTL 을 IP 형태로 패키징하여 배치.
   - PS 로부터 클럭과 리셋을 받고, GPIO 로부터 모드 제어 신호를 받아 동작한다.
   - Input: PS 로부터 시스템 클럭(pl_clk0)과 리셋(iRst_n)을 받고, 외부 포트로부터 카메라 신호(CAM_PCLK, CAM_DATA 등)를 입력받으며, GPIO 로부터 모드 제어 신호(iMode[1:0])를 받아 동작한다.
   - Output: 처리된 영상 데이터와 제어 신호(oLcdClk, oLcdHSync, oLcdVSync, oLcdR/G/B 등)는 외부 포트를 통해 TFT-LCD 패널과 물리적으로 연결된다.

2.6 XDC

```
# ============================================================================
# 0. FPGA Configuration Settings
# ============================================================================
set_property CONFIG_VOLTAGE 1.8 [current_design]
set_property BITSTREAM.CONFIG.UNUSEDPIN PULLNONE [current_design]

# ============================================================================
# 1. System Clock & Reset (Internal Zynq Connection - Commented Out)
# ============================================================================
# iClk, iRst_n 은 Zynq 내부에서 연결되므로 주석 유지
#set_property PACKAGE_PIN D7 [get_ports iClk]
#set_property IOSTANDARD LVCMOS18 [get_ports iClk]
#set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets iClk]

#set_property PACKAGE_PIN F8 [get_ports iRst_n]
#set_property IOSTANDARD LVCMOS12 [get_ports iRst_n]


# ============================================================================
# 5. Mode Selection Input (Internal AXI GPIO - Commented Out)
# ============================================================================
# iMode 는 AXI GPIO 를 통해 제어되므로 주석 유지
#set_property PACKAGE_PIN E6 [get_ports {iMode[0]}]
#set_property IOSTANDARD LVCMOS12 [get_ports {iMode[0]}]

#set_property PACKAGE_PIN G6 [get_ports {iMode[1]}]
#set_property IOSTANDARD LVCMOS12 [get_ports {iMode[1]}]


# ============================================================================
# 2. TFT LCD Control Signals
# ============================================================================
# [수정됨] Wrapper 포트명에 맞춰 뒤에 '_0' 추가
# ----------------------------------------------------------------------------
set_property PACKAGE_PIN G1 [get_ports oLcdClk_0]
set_property PACKAGE_PIN E4 [get_ports oLcdHSync_0]
set_property PACKAGE_PIN F1 [get_ports oLcdVSync_0]
set_property PACKAGE_PIN E3 [get_ports oLcdDe_0]
set_property PACKAGE_PIN E1 [get_ports oLcdBacklight_0]

# 전압 설정 유지 (LVCMOS12)
set_property IOSTANDARD LVCMOS12 [get_ports oLcdClk_0]
set_property IOSTANDARD LVCMOS12 [get_ports oLcdHSync_0]
set_property IOSTANDARD LVCMOS12 [get_ports oLcdVSync_0]
set_property IOSTANDARD LVCMOS12 [get_ports oLcdDe_0]
set_property IOSTANDARD LVCMOS12 [get_ports oLcdBacklight_0]
```

```
# ============================================================================
# 3. TFT LCD RGB Data Signals (RGB565)
# ============================================================================
# [수정됨] Wrapper 포트명에 맞춰 뒤에 '_0' 추가
# ----------------------------------------------------------------------------

# --- Red Channel (5 bits) ---
set_property PACKAGE_PIN R3 [get_ports {oLcdR_0[4]}]
set_property PACKAGE_PIN U2 [get_ports {oLcdR_0[3]}]
set_property PACKAGE_PIN U1 [get_ports {oLcdR_0[2]}]
set_property PACKAGE_PIN T3 [get_ports {oLcdR_0[1]}]
set_property PACKAGE_PIN T2 [get_ports {oLcdR_0[0]}]

# --- Green Channel (6 bits) ---
set_property PACKAGE_PIN M1 [get_ports {oLcdG_0[5]}]
set_property PACKAGE_PIN M5 [get_ports {oLcdG_0[4]}]
set_property PACKAGE_PIN M4 [get_ports {oLcdG_0[3]}]
set_property PACKAGE_PIN L2 [get_ports {oLcdG_0[2]}]
set_property PACKAGE_PIN L1 [get_ports {oLcdG_0[1]}]
set_property PACKAGE_PIN P3 [get_ports {oLcdG_0[0]}]

# --- Blue Channel (5 bits) ---
set_property PACKAGE_PIN N2 [get_ports {oLcdB_0[4]}]
set_property PACKAGE_PIN P1 [get_ports {oLcdB_0[3]}]
set_property PACKAGE_PIN N5 [get_ports {oLcdB_0[2]}]
set_property PACKAGE_PIN N4 [get_ports {oLcdB_0[1]}]
set_property PACKAGE_PIN M2 [get_ports {oLcdB_0[0]}]

###### OV5640 CIS Camera Pins
set_property PACKAGE_PIN G5 [get_ports CAM_PCLK_0]
#set_property PACKAGE_PIN D7 [get_ports CAM_PCLK]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets CAM_PCLK_0]

set_property PACKAGE_PIN A7 [get_ports CAM_PWDN_0]
set_property PACKAGE_PIN A6 [get_ports CAM_RESETn_0]
set_property PACKAGE_PIN E6 [get_ports CAM_SCCB_SCL_0]
set_property PACKAGE_PIN G6 [get_ports CAM_SCCB_SDA_0]
set_property PACKAGE_PIN F7 [get_ports CAM_HSYNC_0]
set_property PACKAGE_PIN G7 [get_ports CAM_VSYNC_0]
set_property PACKAGE_PIN G5 [get_ports CAM_PCLK_0]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets CAM_PCLK_0]
set_property PACKAGE_PIN F6 [get_ports CAM_MCLK_0]
set_property PACKAGE_PIN E5 [get_ports {CAM_DATA_0[0]}]
set_property PACKAGE_PIN D6 [get_ports {CAM_DATA_0[1]}]
set_property PACKAGE_PIN D5 [get_ports {CAM_DATA_0[2]}]
set_property PACKAGE_PIN C7 [get_ports {CAM_DATA_0[3]}]
set_property PACKAGE_PIN B6 [get_ports {CAM_DATA_0[4]}]
set_property PACKAGE_PIN C5 [get_ports {CAM_DATA_0[5]}]
set_property PACKAGE_PIN E8 [get_ports {CAM_DATA_0[6]}]
set_property PACKAGE_PIN D8 [get_ports {CAM_DATA_0[7]}]
#set_property IOSTANDARD LVCMOS18 [get_ports CAM_PCLK]
#set_property IOSTANDARD LVCMOS18 [get_ports CAM_PWDN]
#set_property IOSTANDARD LVCMOS18 [get_ports CAM_RESETn]
#set_property IOSTANDARD LVCMOS18 [get_ports CAM_SCCB_SCL]
#set_property IOSTANDARD LVCMOS18 [get_ports CAM_SCCB_SDA]
#set_property IOSTANDARD LVCMOS18 [get_ports {CAM_DATA[*]}]
#set_property IOSTANDARD LVCMOS18 [get_ports CAM_HSYNC]
#set_property IOSTANDARD LVCMOS18 [get_ports CAM_VSYNC]
#set_property IOSTANDARD LVCMOS18 [get_ports CAM_MCLK]

# ============================================================================
# 4. Bank Voltage / IOSTANDARD 설정
# ============================================================================
# Bank 26 : 1.8V (주요 데이터 핀들)
set_property IOSTANDARD LVCMOS18 [get_ports -of_objects [get_iobanks 26]]

# Bank 65 : 1.2V (클럭/HS/VS/DE/Reset 등 일부 제어핀)
set_property IOSTANDARD LVCMOS12 [get_ports -of_objects [get_iobanks 65]]
```

2.7 SW Control

*Code Overview

```
/*****************************************************************************
* CNN SW control
* Target: Zynq UltraScale+ MPSoC
* Interface: AXI GPIO (Base Address: 0xA0000000)
*****************************************************************************/
#include <stdio.h>
#include "xil_printf.h"
#include "xil_io.h"        // Xil_Out32 사용
#include "xparameters.h"   // 하드웨어 파라미터

// ============================================================================
// 1. Address & Mode Definitions
```

```
// ========================================================================
#define GPIO_BASE_ADDR      0xA0000000U : AXI GPIO IP Base Address [Vivado Address Editor]

#define MODE_SHARPEN        0   // RTL: 2'b00
#define MODE_EDGE_ENHANCE   1   // RTL: 2'b01
#define MODE_BYPASS         2   // RTL: default (2'b10)

// ========================================================================
// 2. Main Function
// ========================================================================
int main()
{
    char input_char;
    xil_printf("\n\r=================================================");
    xil_printf("\n\r   CNN Accelerator Control System Started");
    xil_printf("\n\r=================================================\n\r");
    // 초기 상태 설정 (Bypass)
    Xil_Out32(GPIO_BASE_ADDR, MODE_BYPASS);
    xil_printf("Default Mode Set: [2] Bypass\n\r");
    while (1) {
        // 메뉴 출력
        xil_printf("\n\r-----------------------------------");
        xil_printf("\n\r Select Filter Mode:");
        xil_printf("\n\r [0] Sharpen");
        xil_printf("\n\r [1] Edge Enhance");
        xil_printf("\n\r [2] Bypass (Original)");
        xil_printf("\n\r-----------------------------------");
        xil_printf("\n\r Command > ");
        // 1. 키보드 입력 대기 (Blocking)
        input_char = inbyte();
        // 2. 입력된 문자 화면에 표시 (Echo)
        outbyte(input_char);
        xil_printf("\n\r");
        // 3. 모드 변경 로직
        switch (input_char) {
            case '0':
                xil_printf(" -> Applying [Sharpen] Filter...\n\r");
                Xil_Out32(GPIO_BASE_ADDR, MODE_SHARPEN);
                break;
            case '1':
                xil_printf(" -> Applying [Edge Enhance] Filter...\n\r");
                Xil_Out32(GPIO_BASE_ADDR, MODE_EDGE_ENHANCE);
                break;
            case '2':
                xil_printf(" -> Applying [Bypass] Mode...\n\r");
                Xil_Out32(GPIO_BASE_ADDR, MODE_BYPASS);
                break;
            default:
                xil_printf(" -> Invalid Command. Please press 0, 1, or 2.\n\r");
                break;
        }
    }

    return 0;
}
```

* 메인 함수 동작 흐름

(1) 초기화 및 시작 메시지 출력 - 프로그램 실행 시 터미널에 시스템 시작 메시지를 출력하여 동작 여부를 알립니다.

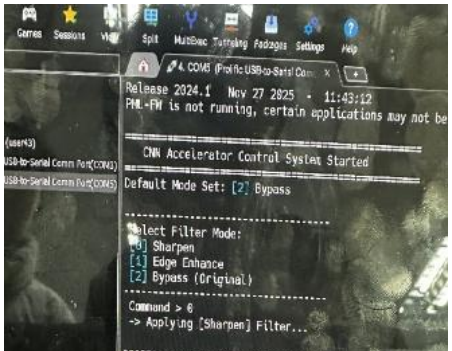
(2) 초기 상태 설정 (Default Set) - Xil_Out32(GPIO_BASE_ADDR, MODE_BYPASS); - 시스템 on, 즉시 'Bypass' 모드(2)

(3) 무한 루프 (While Loop) - 폴링 방식 제어

① 메뉴 출력: 사용자에게 선택 가능한 필터 옵션(0, 1, 2)을 보여줍니다.

② 입력 대기 (Blocking):

`input_char = inbyte();` 함수는 사용자가 키보드를 누를 때까지



CPU 를 대기 상태로 유지합니다.

③ 입력 확인 (Echo):

`outbyte(input_char);`를 통해 사용자가 누른 키를 터미널에 다시 보여줍니다.

④ 모드 변경 (Switch-Case):

- 입력된 문자('0', '1', '2')에 따라 분기합니다.

- 각 케이스마다 해당 모드에 맞는 값을 GPIO 레지스터에 씁니다.

예) '0' 입력 시 -> Xil_Out32(0xA0000000, 0); -> RTL 의 iMode 포트에 00 전달

- 유효하지 않은 키 입력 시 에러 메시지를 출력합니다.

* Interaction With HW

- 통신 방식: MMIO (Memory Mapped I/O)
- 데이터 흐름:[사용자 입력 (UART)] -> [PS (C 코드)] -> [AXI Interconnect] -> [AXI GPIO] -> [PL (CNN_Top 모듈의 iMode 포트)]
- 동작: SW 에서 Xil_Out32 함수가 실행 ->, FPGA 내부 iMode 신호 변경 -> 다음 클럭 사이클부터 CNN Module 는 변경된 필터 계수(Kernel Coefficient)를 사용하여 연산을 수행합니다.

3. 검증

3. 1 Testbench

[a] Code Overview

tb_CNN_Top Testbench 는 최상위 모듈인 CNN_Top 의 기능적 검증을 위해 설계되었다.
카메라 입력 스트림(CAM_PCLK, CAM_VSYNC, CAM_HSYNC, CAM_DATA) 을 실제 프레임 형태(480×272)로 생성하여 DUT 에 인가하고, CNN_Top 이 지정된 모드(iMode) 에 따라 입력 영상을 처리한 뒤 TFT-LCD 출력 포트(oLcdR/G/B, oLcdClk, oLcdHSync, oLcdVSync, oLcdDe) 로 유효한 픽셀 데이터와 동기 신호를 정상적으로 출력하는지를 확인하는 것이다.
LCD 출력 신호를 관찰하여 파이프라인이 정상적으로 동작하고 픽셀 데이터가 출력단까지 도달하는지를 검증하도록 구성하였다.

```
`timescale 1ns / 1ps

//============================================================
// Testbench for CNN_Top
//  - Simulates a single 480x272 RGB565 frame coming from camera
//  - UUT converts camera stream -> BRAM -> Conv3x3 -> LCD RGB565
//============================================================
module tb_CNN_Top;

    // ------------------------------------------------------
    // Parameters (camera frame size)
    // ------------------------------------------------------
    localparam IMG_W = 480;
    localparam IMG_H = 272;

    // ------------------------------------------------------
    // DUT I/O
    // ------------------------------------------------------
    reg         iClk;
    reg         iRst_n;

    wire        CAM_SCCB_SCL;
```

```
    wire        CAM_SCCB_SDA;
    reg         CAM_PCLK;
    reg  [7:0]  CAM_DATA;
    wire        CAM_RESETn;
    reg         CAM_HSYNC;
    reg         CAM_VSYNC;
    wire        CAM_PWDN;
    wire        CAM_MCLK;

    reg  [1:0]  iMode;

    wire        oLcdClk;
    wire        oLcdHSync;
    wire        oLcdVSync;
    wire        oLcdDe;
    wire        oLcdBacklight;
    wire [4:0]  oLcdR;
    wire [5:0]  oLcdG;
    wire [4:0]  oLcdB;

    // ------------------------------------------------------
    // DUT Instance
    // ------------------------------------------------------
    CNN_Top uut (
        // System
        .iClk          (iClk),
        .iRst_n        (iRst_n),

        // Camera side
        .CAM_SCCB_SCL  (CAM_SCCB_SCL),
        .CAM_SCCB_SDA  (CAM_SCCB_SDA),
        .CAM_PCLK      (CAM_PCLK),
        .CAM_DATA      (CAM_DATA),
        .CAM_RESETn    (CAM_RESETn),
        .CAM_HSYNC     (CAM_HSYNC),
        .CAM_VSYNC     (CAM_VSYNC),
        .CAM_PWDN      (CAM_PWDN),
        .CAM_MCLK      (CAM_MCLK),

        // Mode (00: Sharpen, 01: Edge Enhance, 10/11: Bypass)
        .iMode         (iMode),

        // LCD side
        .oLcdClk       (oLcdClk),
        .oLcdHSync     (oLcdHSync),
        .oLcdVSync     (oLcdVSync),
        .oLcdDe        (oLcdDe),
        .oLcdBacklight (oLcdBacklight),
        .oLcdR         (oLcdR),
        .oLcdG         (oLcdG),
        .oLcdB         (oLcdB)
    );

    // ------------------------------------------------------
    // Clock generation
    //  - iClk : 100MHz (10ns period)
    //  - CAM_PCLK : ~25MHz (40ns period, just for simulation)
    // ------------------------------------------------------
    initial begin
        iClk = 1'b0;
        forever #5 iClk = ~iClk;   // 100 MHz
    end

    initial begin
        CAM_PCLK = 1'b0;
        forever #20 CAM_PCLK = ~CAM_PCLK;  // 25 MHz
    end

    // ------------------------------------------------------
    // Reset & top-level stimulus
    // ------------------------------------------------------
    initial begin
        // 초기값
        iRst_n    = 1'b0;
        CAM_VSYNC = 1'b1;   // Frame inactive (idle)
        CAM_HSYNC = 1'b0;
        CAM_DATA  = 8'h00;
        iMode     = 2'b10;  // 기본: Bypass 모드 (Conv 결과가 입력과 비슷하게 나오도록)

        // 파형 덤프 (필요시 사용)
        // $dumpfile("cnn_top_tb.vcd");
        // $dumpvars(0, tb_CNN_Top);

        // Reset 시간
        #200;
        iRst_n = 1'b1;

        // (옵션) 카메라 I2C 설정이 도는 시간을 약간 기다려도 됨
        #10;  // 100us 정도 대기 (필요 없다면 줄여도 됨)

        // 단일 프레임 전송
        $display("[%0t] Start sending camera frame", $time);
        send_frame();
        $display("[%0t] Camera frame done. Waiting for LCD output...", $time);

        // LCD 쪽이 프레임을 다 소모할 시간을 여유 있게 줌
```

```
        #50_000_000;  // 50ms 정도 (필요시 조정)

        $display("[%0t] Simulation finished.", $time);
        $finish;
    end
    // ----------------------------------------------------
    // Task: 한 픽셀(RGB565)을 카메라 쪽으로 전송
    //  - camera_to_ram 모듈은 8bit 씩 2 번(PCLK 2 주기)에 걸쳐
    //    High byte, Low byte 순서로 받도록 설계되어 있음.
    //  - camera_to_ram 는 posedge CAM_PCLK 에서 샘플링하므로
    //    testbench 에서는 negedge 에서 데이터를 변경해줌.
    // ----------------------------------------------------
    task send_pixel(input [15:0] pixel);
    begin
        // High byte 전송
        @(negedge CAM_PCLK);
        CAM_DATA = pixel[15:8];

        // Low byte 전송
        @(negedge CAM_PCLK);
        CAM_DATA = pixel[7:0];
    end
    endtask
    // ----------------------------------------------------
    // Task: 한 라인(가로 480 픽셀)을 전송
    //  - HSYNC = 1 동안 데이터 유효
    //  - 라인 끝에서 HSYNC 를 0 으로 떨어뜨려서
    //    camera_to_ram 내부의 v_count 증가 조건을 만족시킴
    // ----------------------------------------------------
    task send_line(input integer line_idx);
        integer x;
        reg [15:0] pix;
    begin
        // 라인 시작
        CAM_HSYNC = 1'b1;
        for (x = 0; x < IMG_W; x = x + 1) begin
            // 간단한 패턴: y 방향으로 Red, x 방향으로 Green, (x+y)로 Blue
            pix[15:11] = line_idx[4:0];         // R (5bit)
            pix[10:5]  = x[5:0];                // G (6bit)
            pix[4:0]   = x + line_idx[4:0];   // B (5bit)

            send_pixel(pix);
        end
        // 라인 끝: HSYNC low + 약간의 H-blank
        @(negedge CAM_PCLK);
        CAM_HSYNC = 1'b0;
        CAM_DATA  = 8'h00;

        // H-blank 조금 더
        repeat (10) @(negedge CAM_PCLK);
    end
    endtask
    // ----------------------------------------------------
    // Task: 전체 프레임(480x272)을 전송
    //  - VSYNC = 0 동안 한 프레임
    //  - 각 라인 사이에 HSYNC falling edge 가 나오므로
    //    camera_to_ram 내부 v_count 가 증가함
    // ----------------------------------------------------
    task send_frame;
        integer y;
    begin
        // Frame 시작: VSYNC active-low
        @(negedge CAM_PCLK);
        CAM_VSYNC = 1'b0;
        // 약간의 vertical front porch (옵션)
        repeat (10) @(negedge CAM_PCLK);
        // 모든 라인 전송
        for (y = 0; y < IMG_H; y = y + 1) begin
            send_line(y);
        end
        // Frame 끝: VSYNC high (idle)
        @(negedge CAM_PCLK);
        CAM_VSYNC = 1'b1;
        CAM_HSYNC = 1'b0;
        CAM_DATA  = 8'h00;
        // Vertical blanking
        repeat (100) @(negedge CAM_PCLK);
    end
    endtask
    integer lcd_pixel_cnt;
    always @(posedge oLcdClk or negedge iRst_n) begin
        if (!iRst_n) begin
            lcd_pixel_cnt <= 0;
        end else begin
            if (oLcdHSync && oLcdVSync &&
                (oLcdR !== 5'd0 || oLcdG !== 6'd0 || oLcdB !== 5'd0)) begin
                lcd_pixel_cnt <= lcd_pixel_cnt + 1;
                if (lcd_pixel_cnt < 10) begin
                    $display("[%0t] LCD Pixel %0d : R=%0d G=%0d B=%0d",
                             $time, lcd_pixel_cnt, oLcdR, oLcdG, oLcdB);
                end
            end
        end
    end
endmodule
```

[b] 파라미터 정의 및 DUT 인스턴스화

• 파라미터 정의: IMG_W = 480, IMG_H = 272 -> 이미지 크기 정의.
해당 값은 send_line, send_frame 태스크의 반복 루프 조건으로 사용, 480×272 픽셀을 전송하도록 하였다.

• DUT 인스턴스화: CNN_Top 모듈을 uut 로 인스턴스화

[c] 클럭 및 리셋 시퀀스

- 클럭 생성 (iClk) : initial 블록에서 5ns 마다 신호를 반전시켜 100MHz 의 시스템 클럭을 생성한다.
- 카메라 픽셀 클럭 생성(CAM_PCLK) : #20 마다 토글하여 25MHz(주기 40ns) 의 픽셀 클럭을 생성한다.
- 리셋 :iRst_n 은 Active-Low 로, 초기 200ns 동안 Low 로 유지하여 DUT 를 초기화한 뒤 High 로 해제하였다.
- 카메라 입력은 초기 idle 상태로 두기 위해 CAM_VSYNC=1, CAM_HSYNC=0, CAM_DATA=0 으로 설정하였다.
- iMode = 2'b10(Bypass) 로 설정하여, 우선 출력 스트림이 정상적으로 형성되는지를 확인하도록 하였다.

[d] 카메라 프레임 입력 생성(Stimulus) 태스크

Testbench 는 카메라 입력을 모사하기 위해 픽셀/라인/프레임 단위의 태스크를 정의하였다.

• send_pixel(pixel): RGB565 1 픽셀 전송
  - 카메라 입력 = 8bit 데이터, RGB565(16bit) 픽셀을 상위 바이트 → 하위 바이트 순서로 2 번에 걸쳐 전송하도록 구성하였다.
  - DUT 가(posedge CAM_PCLK)에서 데이터를 샘플링하는 구조를 가정하여, Testbench 는 데이터 안정 시간을 확보하기 위해 (negedge CAM_PCLK)에서 CAM_DATA 를 갱신하도록 하였다.

• send_line(line_idx): 480 픽셀 한 라인 전송
  - 라인 구간 동안 CAM_HSYNC=1 로 설정하여 데이터 유효 구간을 나타내고, 480 개의 픽셀을 연속 전송하였다.
  - 픽셀 패턴은 검증을 쉽게 하기 위해 간단한 규칙 기반으로 생성하였다.
  - R(5bit) = line_idx[4:0] (y 방향 변화)
  - G(6bit) = x[5:0] (x 방향 변화)
  - B(5bit) = x + line_idx[4:0] (x+y 결합 변화)
  - 라인 종료 시 CAM_HSYNC=0 으로 내린 뒤, repeat(10) 으로 H-blank 구간을 추가하였다.

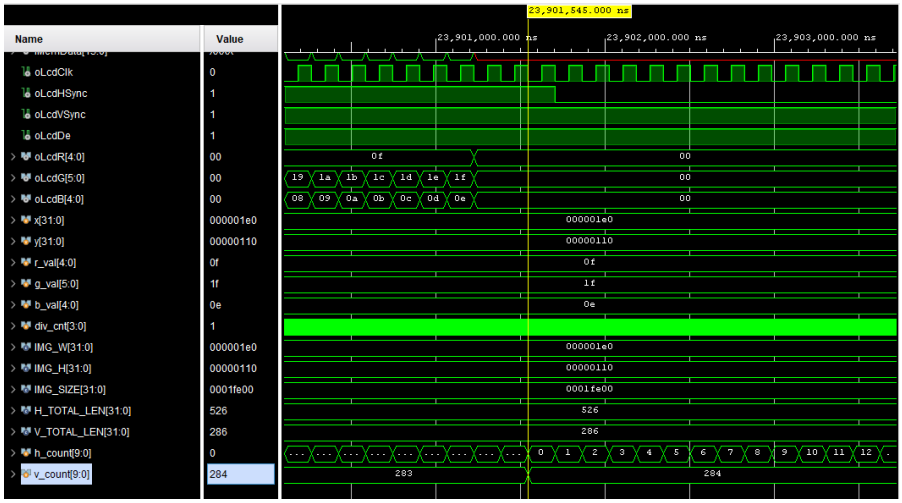
• send_frame: 480×272 전체 프레임 전송
  - 프레임 구간은 CAM_VSYNC=0(active-low) 동안으로 정의하였다.
  - Vertical porch 로 repeat(10) 대기 후, 272 개의 라인을 순차적으로 전송하였다.
  - 전송 완료 후 CAM_VSYNC=1 로 복귀시키고 repeat(100) 으로 vertical blanking 구간을 부여하였다.

[e] 시뮬레이션 흐름 및 종료

• 리셋 해제 후 send_frame() 을 호출하여 단일 프레임 입력을 전송하였다.

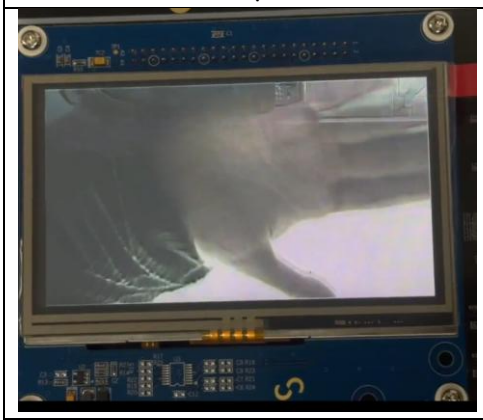
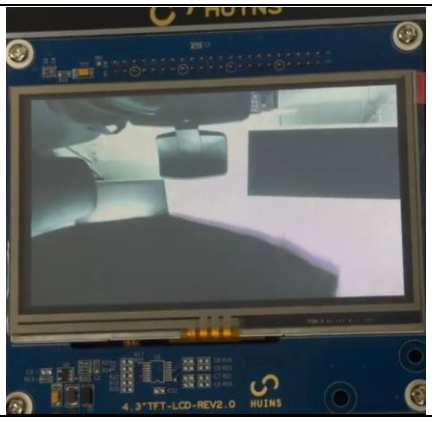
• 프레임 전송 완료 뒤 DUT 내부 파이프라인 처리 및 LCD 출력이 충분히 진행되도록 #50_000_000 만큼 추가 대기한 후 시뮬레이션을 종료하였다.

3.2 Wave Form



3.3 Operation Board Video Image

* Result Video Image

Sharpen	EdgeEnhance	Bypass
		

4. Conclusion

4.1 Further Work

[a] 가속기 고도화 개선

- 현재 문제점: 단순한 연산 방식 적용. 향후 파라미터 및 연산량 많은 작업 요구 시 병목 발생 가능성 존재함.
- 개선 방안: 병렬 연산 어레이(Systolic Array) 및 AXI DMA 기반 고속 인터페이스 적용

[b] pixel conversion logic 개선 : MSB -> Complex Logic

- 현재 문제점: RGB888 에서 RGB565 로 변환 시, 단순히 하위 비트를 버림처리하여 표현이 거칠어지거나 색상 밴딩현상이 발생할 수 있다.
- 개선 방안: 반올림 로직, 오차 확산 디더링 알고리즘을 적용하여, 줄어드는 비트 수로 인한 화질 열화를 최소화한다.

4.2 Reflections

[a] 한정호

본 프로젝트를 통해 하드웨어 기반의 이미지 처리 파이프라인 전체를 설계해보는 귀중한 경험을 했다.특히 Vivado BRAM IP 의 활용법과 클락 도메인이 다른 시스템간의 동기화 이슈를 Clock Enable 신호와 위상 정렬을 통해 해결하는 과정에서 타이밍 제약과 준안전성에 대한 이해를 높일 수 있었다. 또한 LCD 출력을 위해 타이밍 문제를 픽셀 하나하나 조절해가며 해결하면서 타이밍의 중요성을 깨달을 수 있었다.

[b] 노영찬

본 프로젝트를 통해 병목 구간 분석을 기반으로 window3x3 모듈을 개선하면서, 단순히 연산 기능을 구현하는 것과 실제로 높은 성능을 달성하는 것 사이에는 큰 차이가 있음을 깨달았다. 특히 데이터가 어떻게 이동하고 재사용되는지가 전체 시스템 성능을 좌우한다는 점을 명확히 이해하게 되었다. window3x3 모듈 개선 과정은 작은 구조 변경이 frame/s 에 직접적인 영향을 미칠 수 있음을 보여주었고, 이를 통해 하드웨어 설계에서 분석 → 개선 → 검증의 반복 과정이 매우 중요하다는 것을 배울 수 있었다..

[c] 전민서

Zynq UltraScale+ MPSoC 기반으로 SoC 구조를 직접 구성하고 운용하면서, PS 와 PL 간 제어·데이터 흐름이 어떻게 맞물려 동작하는지 체계적으로 이해할 수 있었다. 특히 AXI 인터페이스를 이용한 IP 제어와 Vitis 환경에서의 통합 검증을 수행하며, 단순히 연산 블록을 구현하는 수준을 넘어 임베디드 시스템 전체를 하나의 관점에서 설계·디버깅하는 역량이 강화되었다. 또한 카메라 모듈을 추가하여 실시간 영상 데이터를 InBuf 에 적재하는 입력단 모듈을 구현하면서 실제 동작 가능한 시스템으로 확장하는 경험을 할 수 있었다. 이 과정에서 픽셀 데이터 수집, 버퍼링, 동기 신호 처리 등 입력단의 병목과 안정성이 전체 처리 성능과 직결됨을 체감했고, HW 가속 로직과 SW 제어를 함께 고려하는 Codesign 관점의 중요성을 다시 한 번 깨달았다.

4.3 Reference

[1] 송인철, "고급 Project," 숭실대학교 강의 자료, 2025.

[2] Xilinx, "Zynq UltraScale+ Device Technical Reference Manual (UG1085)," v2.2, 2020.

[3] Tianma Micro-electronics, "Model No. TM043NBH02 Final Product Specification," Ver. 2.6, 2018