2025 AI 시스템 반도체 설계 2기

대한상공회의소
서울기술교육센터

FPGA Based Real-time Image Processing Photo Booth

Semicon 4 Cut

발표자 김민규

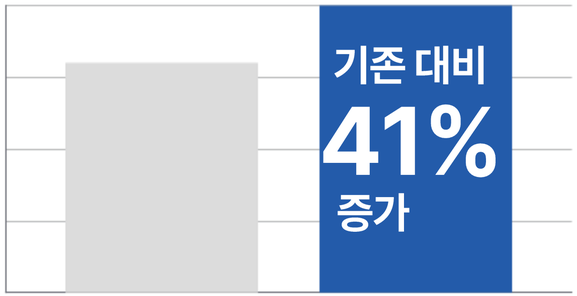
팀장 김민규

팀원 고종완 | 김종현 | 서윤철 | 임도엽 | 임재홍 | 최규리

목차-제본용

01 개요 | 프로젝트 목표

01 셀프 포토부스 시장 성장성



- 최근 5년간 셀프 촬영업 사업자 기준
  →셀프 포토 스튜디오 산업의 성장

02 K- 콘텐츠 문화로 자리잡은 인생네컷

BRAND K

- 포토이즘 브랜드 '브랜드K' 선정
  → K-콘텐츠 문화로 자리매김
- 국내 인생네컷 사업 글로벌 확산

03 프로젝트 목표 도출

11가지 필터

촬영 후 AI 분석

전력 소모 절감

문화

돈룩업, 에스파 새 싱글 발매 기념 콜라보레이션 선보여

동아경제 | 업데이트 2025-07-03 10:41

약 2년간 브랜드K 상표 사용 권한 부여, 다양한 해외 진출 지원

포토이즘, 브랜드K 선정…“K-콘텐츠 문화 리더 입증”

입력 2023.08.01 10:13 | 수정 2023.08.01 10:33 | 댓글

엘케이벤쳐스 '인생네컷', 2025 대한민국 소비자만족지수1위 대상 수상

홍보경 기자 2025.09.10 17:20

중국 MZ세대 사이에서 가장 핫한 트렌드, 셀프 포토 부스

트렌드 | 중국 | 광저우무역관 | 2024-10-10 | 출처 : KOTRA

지드래곤, 돈룩업과 콜라보 'POWER' 테마 프레임·포토부스 18일 공개

중앙일보 | 입력 2024.11.18 10:00

셀프 포토부스 시장 성장성

자리잡

문화로 자리매김

사업 글로벌 확산

11가지 필터

AI 분석

소모 절감

AI 기반 콘텐츠 생성 및 변환 기술

사용자 감정 및 포즈

촬영된 사진

개발 환경

Basys3

VGA to HDMI

HDMI Capture Board

OV7670

VIVADO

Python

System Verilog

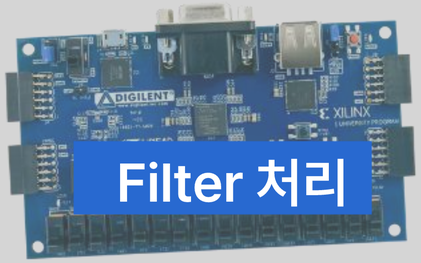
Flutter

System Flow

영상 촬영

▲ Camera

Filter 처리



▲ Basys3

HDMI 화면 송출

▲ VGA to HDMI

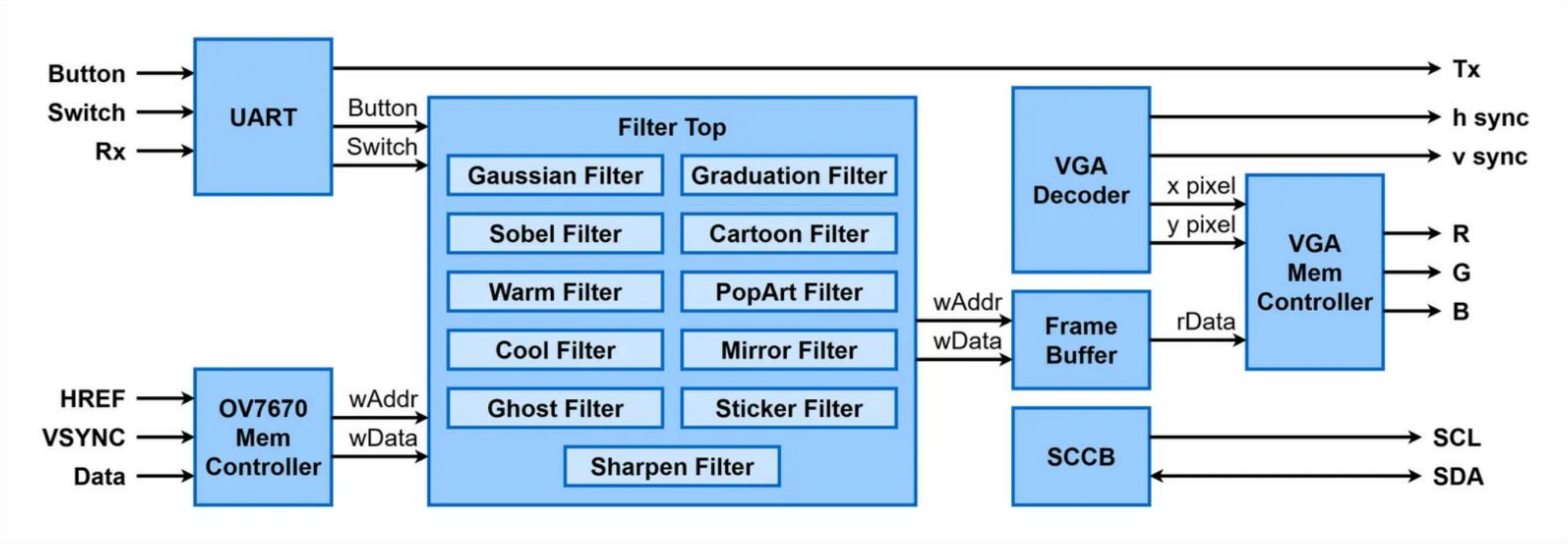
데이터 브릿지

▲ Local Server

프레임 시각화

▲ GUI

02 System Architecture



Filter

Filter Setting

Comic	Pop Art	mirror
Warm	Cool	Phantom
Soft Glow	Retro Frame	Edge Pop
Original		Sticker

Filter 적용

Simple Filters (Cool, Warm, Ghost Filter)

```
always_comb begin
    r_in   = wData_in[15:11];
    g_in   = wData_in[10:5];
    b_in   = wData_in[4:0];

    r_warm = (r_in + 2 > 31) ? 5'd31 : (r_in + 2);
    b_warm = (b_in > 2) ? (b_in - 2) : 5'd0;
    g_warm = g_in;
end
```

```
always_comb begin
    r_in   = wData_in[15:11];
    g_in   = wData_in[10:5];
    b_in   = wData_in[4:0];

    b_cool = (b_in + 2 > 31) ? 5'd31 : (b_in + 2);
    r_cool = (r_in > 2) ? (r_in - 2) : 5'd0;
    g_cool = g_in;
end
```

```
always_comb begin
    r_in  = wData_in[15:11];
    g_in  = wData_in[10:5];
    b_in  = wData_in[4:0];

    r_inv = ~r_in & 5'h1F;
    g_inv = ~g_in & 6'h3F;
    b_inv = ~b_in & 5'h1F;
end
```

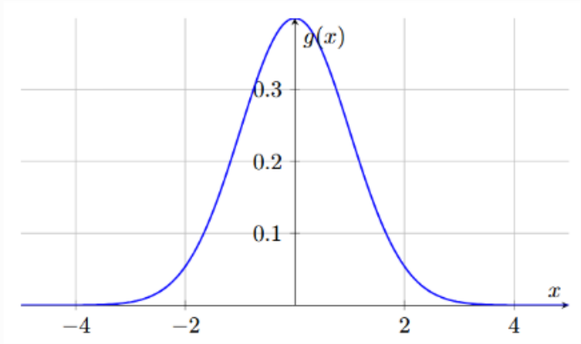
▲ Warm, Cool, Ghost Filter Code Snipet

Enhance needed colors (e.g. blue for cool, red for warm) while softening unnecessary tones
Ghost filter inverts colors to create a distinctive ghost-like effect.

Advanced Filter Design (Gaussian Filter)

$$g(x) = \sqrt{\frac{a}{\pi}}\, e^{-ax^2} \quad \longleftrightarrow \quad \hat{g}(f) = e^{-\pi^2 f^2 / a}$$

▲ Time Domain and Frequency Domain of Gaussian Filter



▲ 1D Gaussian curve with σ = 1

Gaussian Function is an infinite function on each end.
This feature gives us an advantage when designing the filter.

Advanced Filter Design (Gaussian Filter)

$$g(x) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{x^2}{2\sigma^2}} \quad \longleftrightarrow \quad \hat{g}(f) = e^{-\frac{f^2}{2\sigma_f^2}}$$

▲ General expression of Gaussian Function(mean = 0 and deviation = 0)

$$g(x) = \sqrt{\frac{a}{\pi}}\, e^{-ax^2} \qquad (1)$$

$$g(x) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{x^2}{2\sigma^2}} \qquad (3)$$

$$\longrightarrow \quad \sigma = \frac{1}{\sqrt{2a}}$$

▲ Relation between a and σ using 1 and 3

Using 1 and 3 equations we can get an relation between a and σ.

Advanced Filter Design (Gaussian Filter)

$$\sigma = \frac{1}{\sqrt{2a}} \tag{5}$$

$$\hat{g}(f) = e^{-\pi^2 f^2 (2\sigma^2)} = e^{-2\pi^2 \sigma^2 f^2} \tag{6}$$

$$\sigma\,\sigma_f = \frac{1}{2\pi}$$

▲ Relation between time domain σ and frequency domain σ using 5 and 6

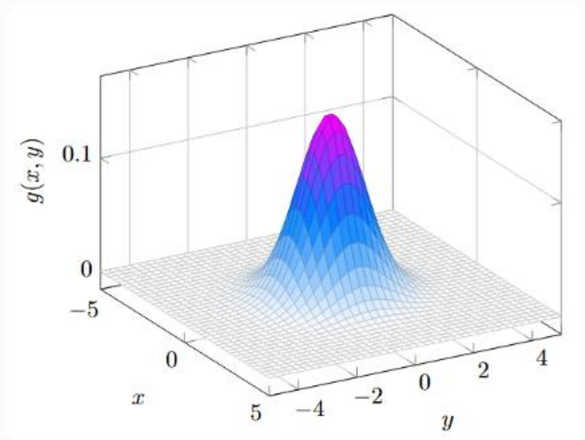
This means that the deviation relation between time
and
frequency domain are inversely proportional to each other.

Advanced Filter Design (Gaussian Filter)

$$g(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{8}$$

▲ 2 - D equation for the Gaussian Funciton

This relation could be also expanded to a 2 Dimensional equation.
Therefore in general, N dimensional equation.



▲ 2 - D Gaussian Funciton Graph

Advanced Filter Design (Gaussian Filter)

$$\hat{g}(f) = e^{-\pi^2 f^2 (2\sigma^2)} = e^{-2\pi^2 \sigma^2 f^2} \qquad (6)$$

▲ Frequency domain of Gaussian Function

Using equation 6, let us define f as the cutoff frequency and derive σ as a function of f, we can get a relation of σ and cutoff freqeuncy.

$$\sigma = \frac{\sqrt{\ln 2}}{2\pi f_c} \qquad \sigma\,\sigma_f = \frac{1}{2\pi}$$

In conclusion, we can make a low pass filter using a realtion of σ and f.
Leading us to make various applications like the "Gaussian Blur Filter".

Advanced Filter Design (Gaussian Filter)

$$\hat{g}(f) = e^{-\pi^2 f^2 (2\sigma^2)} = e^{-2\pi^2 \sigma^2 f^2} \tag{6}$$

▲ Frequency domain of Gaussian Function

Using a "mask" on filters.

1/16 x

1	2	1
2	4	2
1	2	1

3x3

1/273 x

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

5x5

1/1003 x

0	0	1	2	1	0	0
0	3	13	22	13	3	0
1	13	59	97	59	13	1
2	22	97	159	97	22	2
1	13	59	97	59	13	1
0	3	13	22	13	3	0
0	0	1	2	1	0	0

7x7

Larger filters use more resources when operating.

Advanced Filter Design (Gaussian Filter)

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

$$\frac{1}{16} \times \begin{array}{|c|c|c|}\hline 1 & 2 & 1\\ \hline 2 & 4 & 2\\ \hline 1 & 2 & 1\\ \hline\end{array}$$

▲ Example of a Filter Mask 3x3 Used on our projects

The mask value is divided by the sum of the matrix elements due to consider the density of the color features when multiplied.

Mask multiplication will be used with line buffers and frame buffers. This will be described further when explaining Gaussian Blur Filter.

Advanced Filter Design (Gaussian Filter)

```
always_comb begin
    x = wAddr_in % IMG_WIDTH;
    y = wAddr_in / IMG_WIDTH;
end
```

▲ Calculate pixel position (x, y) with address

```
if (y[0] == 1'b0) begin
    mid_rd       <= line_even[x];
    top_rd       <= line_odd[x];
    line_even[x] <= wData_in;
end else begin
```

▲ Input first line to line buffer (y = 0, even)

```
if (x == 0) begin
    top_tap_0 <= '0;
    top_tap_1 <= '0;
    top_tap_2 <= top_rd;
    mid_tap_0 <= '0;
    mid_tap_1 <= '0;
    mid_tap_2 <= mid_rd;
    cur_tap_0 <= '0;
    cur_tap_1 <= '0;
    cur_tap_2 <= wData_in;
end else begin
```

▲ Input first line to line buffer (x = 0, even)

```
always_comb begin
    sumR = R5(top_tap_0) + (R5(top_tap_1) << 1) + R5(top_tap_2) +
        (R5(mid_tap_0) << 1) + (R5(mid_tap_1) << 2) + (R5(mid_tap_2) << 1) +
        R5(cur_tap_0) + (R5(cur_tap_1) << 1) + R5(cur_tap_2);
    sumG = G6(top_tap_0) + (G6(top_tap_1) << 1) + G6(top_tap_2) +
        (G6(mid_tap_0) << 1) + (G6(mid_tap_1) << 2) + (G6(mid_tap_2) << 1) +
        G6(cur_tap_0) + (G6(cur_tap_1) << 1) + G6(cur_tap_2);
    sumB = B5(top_tap_0) + (B5(top_tap_1) << 1) + B5(top_tap_2) +
        (B5(mid_tap_0) << 1) + (B5(mid_tap_1) << 2) + (B5(mid_tap_2) << 1) +
        B5(cur_tap_0) + (B5(cur_tap_1) << 1) + B5(cur_tap_2);

    outR = sumR >> 4;
    outG = sumG >> 4;
    outB = sumB >> 4;
end
```

▲ Calculating the average for each color and down scaling

Advanced Filter Design (Gaussian Filter)

```
always_comb begin
    if (we_in && win_valid) begin
        we_out_nxt = 1'b1;
        waddr_out_nxt = wAddr_in - (IMG_WIDTH + 17'd1);
        wdata_out_nxt = {outR, outG, outB};
    end else begin
        we_out_nxt    = 1'b0;
        waddr_out_nxt = '0;
        wdata_out_nxt = '0;
    end
end
```

▲ Calculate output pixel position (x - 1, y - 1) based on window center

```
always_ff @(posedge clk or posedge reset) begin
    if (reset) begin
        we_out <= 1'b0;
        wAddr_out <= '0;
        wData_out <= '0;
    end else begin
        we_out <= we_out_nxt;
        wAddr_out <= waddr_out_nxt;
        wData_out <= wdata_out_nxt;
    end
end
```

▲ Buffering each pixel

This process is when the filter starts at (x = 0, y = 0).

Please note that this is when we are at the edge of the window.

```
end else begin
   top_tap_0 <= top_tap_1;
   top_tap_1 <= top_tap_2;
   top_tap_2 <= top_rd;
   mid_tap_0 <= mid_tap_1;
   mid_tap_1 <= mid_tap_2;
   mid_tap_2 <= mid_rd;
   cur_tap_0 <= cur_tap_1;
   cur_tap_1 <= cur_tap_2;
   cur_tap_2 <= wData_in;
end
```

▲ In frame line

Advanced Filter Design (Gaussian Filter)

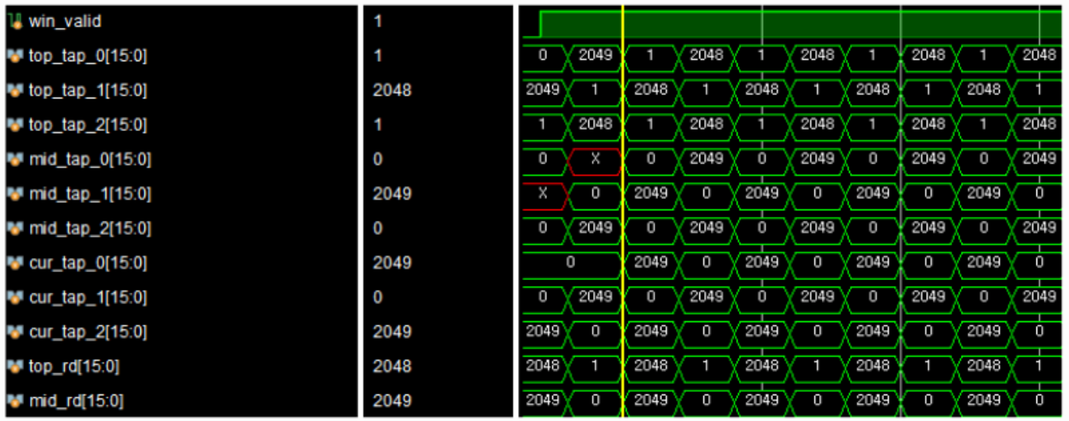
line_even[0:IMG_WIDTH-1] ($IMG\_WIDTH = 320$)					
line_even[0]	line_even[1]	line_even[2]	$\cdots$	line_even[318]	line_even[319]

line_odd[0:IMG_WIDTH-1] ($IMG\_WIDTH = 320$)					
line_odd[0]	line_odd[1]	line_odd[2]	$\cdots$	line_odd[318]	line_odd[319]

▲ Line Buffer Configuration

▲ Line Buffer Input

Advanced Filter Design (Gaussian Filter)



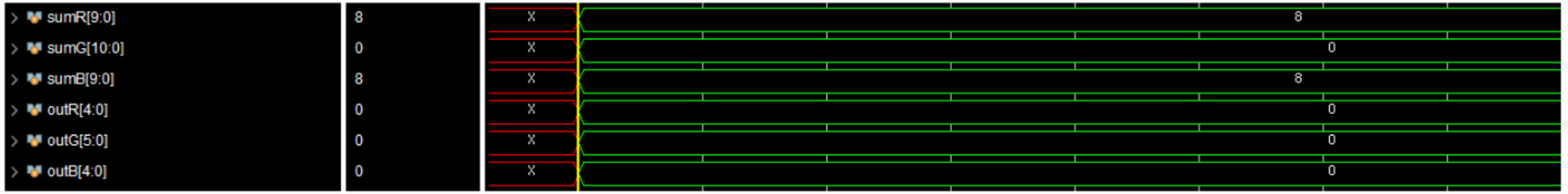
▲ Window configuration simulation

top_tap_0	top_tap_1	top_tap_2
mid_tap_0	mid_tap_1	mid_tap_2
cur_tap_0	cur_tap_1	cur_tap_2

▲ Window configuration

The pixel is valid only when the coordinate is (x > 1 and y > 1)

Advanced Filter Design (Gaussian Filter)



▲ Sum and Output Value

▲ Output Address and Data

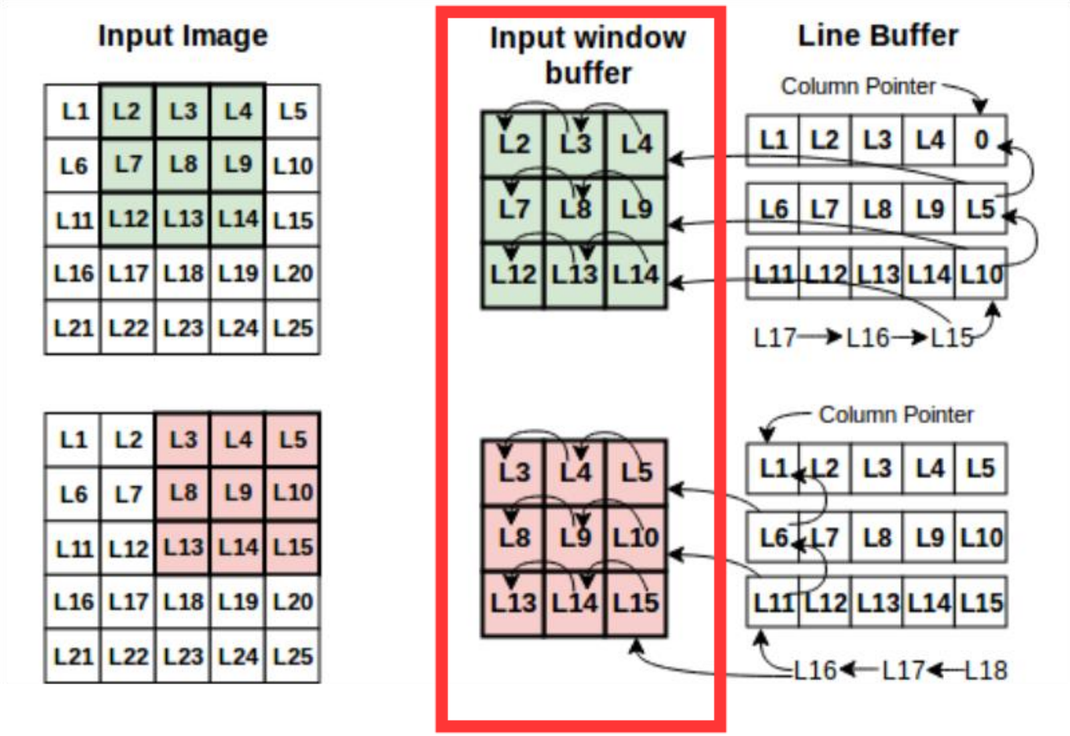
output address = (y coordinate * horizontal length) + x coordinate

```
run all

======================[RESULT]=======================
Checks    : 607
Mismatch  : 0
STATUS    : PASS
=====================================================
```

▲ Simulation Result

Advanced Filter Design (Sharpen Filter)

Line Buffer

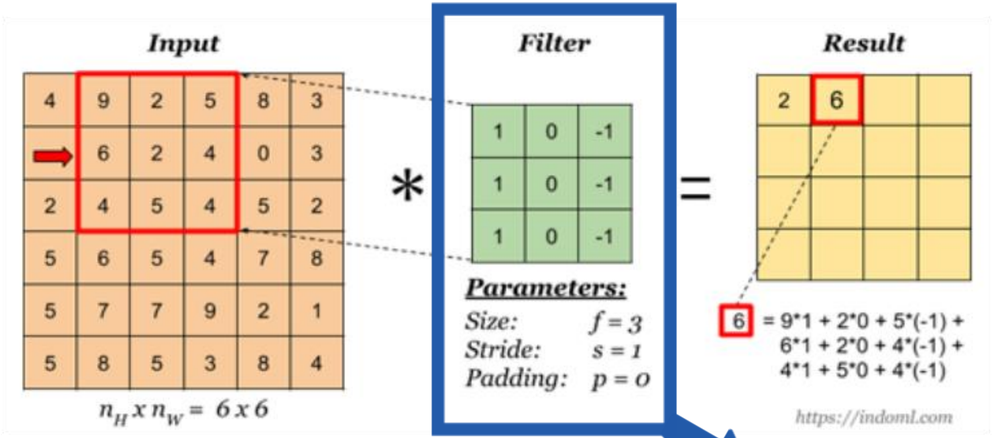



```
logic [4:0] line1_r[0:IMG_WIDTH-1], line2_r[0:IMG_WIDTH-1];
logic [5:0] line1_g[0:IMG_WIDTH-1], line2_g[0:IMG_WIDTH-1];
logic [4:0] line1_b[0:IMG_WIDTH-1], line2_b[0:IMG_WIDTH-1];

 if (we_in) begin
    r00 <= r01;
    r01 <= r02;
    r02 <= line2_r[col_cnt];
    r10 <= r11;
    r11 <= r12;
    r12 <= line1_r[col_cnt];
    r20 <= r21;
    r21 <= r22;
    r22 <= in_r5;

    line2_r[col_cnt] <= line1_r[col_cnt];
    line1_r[col_cnt] <= in_r5;
    // G, B 채널 동일
 end
```

실시간 처리, 메모리 사용량 감소

Advanced Filter Design (Sharpen Filter)

Convolution



$$h(x,y) = 5 \cdot f(x,y) - \Big(f(x-1,y) + f(x+1,y) + f(x,y-1) + f(x,y+1)\Big)$$

0	-1	0
-1	5	-1
0	-1	0

```
always_ff @(posedge clk) begin
    conv_r <= ($signed(
        {1'b0, r11}
    ) * 10'sd5) - ($signed(
        {1'b0, r01}
    ) + $signed(
        {1'b0, r10}
    ) + $signed(
        {1'b0, r12}
    ) + $signed(
        {1'b0, r21}
    ));
    // G, B 채널 동일
end
```

Advanced Filter Design (Sharpen Filter)

Before

After

Sharpening은 고주파 성분을 강조해 선명도를 높임

Advanced Filter Design (Sobel Filter)

Convolution

Gx, Gy Mask

Gx =

-1	0	1
-2	0	2
-1	0	1

수직 경계 검출

Gy =

-1	0	1
-2	0	2
-1	0	1

수평 경계 검출

```
always_ff @(posedge clk) begin
    if (reset) begin
        gx <= 0;
        gy <= 0;
        mag <= 0;
        sobel_val <= 0;
        edge_bin <= 0;
    end else if (we_in) begin
        gx <= (p02 + (p12 << 1) + p22) - (p00 + (p10 << 1) + p20);
        gy <= (p20 + (p21 << 1) + p22) - (p00 + (p01 << 1) + p02);
        mag <= (gx[11] ? -gx : gx) + (gy[11] ? -gy : gy);

        sobel_val <= (mag >> 3 > 8'hFF) ? 8'hFF : mag[10:3];
        edge_bin <= (sobel_val >= THRESHOLD) ? 8'hFF : 8'h00;
    end
end
```

Magnitude & Threshold

$$|G| = |G_x| + |G_y|, \quad Edge(x,y) = \begin{cases} 255, |G| \geq T \\ 0, |G| \geq T \end{cases}$$

Advanced Filter Design (Sobel Filter)

Before

After

영상의 기울기를 이용한 윤곽선 추출

Sticker Filter

Pipeline

색상 처리

▼

객체 분석

▼

데이터 관리

▼

이미지 합성

color space

```
// ============================================================
// RGB565 → 빨간색 검출
// ============================================================
logic [4:0] in_r, in_g, in_b;
logic is_red;

assign in_r = wData_in[15:11];  // 5비트 빨간색
assign in_g = wData_in[10:5];  // 6비트 초록색
assign in_b = wData_in[4:0];  // 5비트 파란색

always_ff @(posedge clk) begin
    if (reset) begin
        is_red <= 0;
    end else if (we_in) begin
        // 빨간색 검출: R이 높고, G와 B가 낮아야 함
        is_red <= (in_r > 14) && (in_g < 13) && (in_b < 13);
    end
end
```

2. connected Component & centroid

```
// ============================================================
// 연속된 빨간색 픽셀 6개 이상 찾기
// ============================================================
logic [7:0] red_sequence_count;  // 현재 연속 빨간색 픽셀 개수
logic [7:0] temp_red_col, temp_red_row;  // 검사 중인 좌표
logic temp_red_detected;  // 임시 검출 플래그
logic [7:0] red_center_col;
logic [7:0] red_center_row;
logic       red_detected;

// ============================================================
// 스티커 시스템 (5초간 표시)
// ============================================================
logic [7:0] sticker_center_col;  // 스티커 중심 열
logic [7:0] sticker_center_row;  // 스티커 중심 행
logic sticker_active;  // 스티커 활성화 플래그
logic [26:0] sticker_timer;                  // 5초 카운터 (25MHz 기준, 125,000,000 필요)
logic sticker_trigger;  // 스티커 트리거 (새로운 빨간 점 감지)

logic [$clog2(IMG_WIDTH)-1:0] col_cnt;
logic [$clog2(IMG_HEIGHT)-1:0] row_cnt;

// 주소 파이프라인 동기 (4단계 파이프라인)
logic [16:0] addr_d1, addr_d2, addr_d3, addr_d4;
logic we_d1, we_d2, we_d3, we_d4;

// 최종 출력 프레임 (filtered_skin과 동기화)
logic [$clog2(IMG_WIDTH)-1:0]
    col_cnt_d1, col_cnt_d2, col_cnt_d3, col_cnt_d4;
logic [$clog2(IMG_HEIGHT)-1:0]
    row_cnt_d1, row_cnt_d2, row_cnt_d3, row_cnt_d4;
```

Sticker Filter

3. image overlay

```
logic [15:0] dout_reg;
 logic centroid_valid;
 logic [7:0] centroid_x;
 logic [7:0] centroid_y;

 always_ff @(posedge clk) begin
     if (reset) dout_reg <= 16'h0000;
     else if (we_d4) begin  // 4단계 파이프라인된 we 사용
         if (sticker_active_d4) begin
             // 스티커 활성화시 무조건 그리기
             if (rom_data != 16'h0000) begin  // 투명이 아닌 경우
                 dout_reg <= rom_data;  // ROM에서 읽은 스티커 데이터
             end else begin
                 dout_reg <= wData_in_d4;  // 투명한 경우 원본 픽셀
             end
         end else begin
             // 스티커 비활성화시 원본 픽셀 출력
             dout_reg <= wData_in_d4;
         end
     end
 end
```

Cartoon Filter

Pipeline	Overview
Coordinate/Window	Line Buffer 2개 채널로 3×3 윈도우 생성
Gaussian Blur	Blur로 노이즈 감소
Posterize	Posterize로 색상 단계 축소
Sobel Edge	Sobel(|Gx|+|Gy|)로 윤곽 검출
Overlay	윤곽선 위치를 검정으로 덮어쓰기

Cartoon Filter

- 3x3 Gaussian blur

```
logic [11:0] sum_r, sum_g, sum_b;
logic [4:0]  blur_r5;
logic [5:0]  blur_g6;
logic [4:0]  blur_b5;

always_ff @(posedge clk) begin
    if (reset) begin
        sum_r   <= '0;
        sum_g   <= '0;
        sum_b   <= '0;
        blur_r5 <= '0;
        blur_g6 <= '0;
        blur_b5 <= '0;
    end else begin
        sum_r <= (r00 + (r01<<1) + r02)
               + ((r10<<1) + (r11<<2) + (r12<<1))
               + (r20 + (r21<<1) + r22);
        sum_g <= (g00 + (g01<<1) + g02)
               + ((g10<<1) + (g11<<2) + (g12<<1))
               + (g20 + (g21<<1) + g22);
        sum_b <= (b00 + (b01<<1) + b02)
               + ((b10<<1) + (b11<<2) + (b12<<1))
               + (b20 + (b21<<1) + b22);

        // >>4 with saturation
        blur_r5 <= (sum_r[11:4] > 12'd31) ? 5'd31 : sum_r[8:4];
        blur_g6 <= (sum_g[11:4] > 12'd63) ? 6'd63 : sum_g[9:4];
        blur_b5 <= (sum_b[11:4] > 12'd31) ? 5'd31 : sum_b[8:4];
    end
end
```

$$\mathrm{sum}_C = (c00 + 2c01 + c02) + (2c10 + 4c11 + 2c12) + (c20 + 2c21 + c22)$$

$$\mathrm{blur}_C = \mathrm{sat}\Big((\mathrm{sum}_C \gg 4), \mathrm{bits}_C\Big)$$

- Posterization

```
wire [4:0] post_r5_w;
wire [5:0] post_g6_w;
wire [4:0] post_b5_w;

// R/B: 5bit -> 상위 2비트만 유지, 하위 3비트 0
assign post_r5_w = {blur_r5[4:3], 3'b000};
assign post_b5_w = {blur_b5[4:3], 3'b000};

// G: 6bit -> 상위 2비트만 유지, 하위 4비트 0
assign post_g6_w = {blur_g6[5:4], 4'b0000};

// 파이프라인 레지스터 (동일)
logic [4:0] post_r5;
logic [5:0] post_g6;
logic [4:0] post_b5;

always_ff @(posedge clk) begin
    if (reset) begin
        post_r5 <= '0;
        post_g6 <= '0;
        post_b5 <= '0;
    end else begin
        post_r5 <= post_r5_w;
        post_g6 <= post_g6_w;
        post_b5 <= post_b5_w;
    end
end
```

$$\mathrm{post\_r5} = \{\,\mathrm{blur\_r5}[4:3],\ 000\,\}$$

Cartoon Filter

- Sobel edge on G

```
logic  signed [10:0] gx, gy;
logic         [10:0] abs_gx, abs_gy;
logic         [11:0] edge_mag;

always_ff @(posedge clk) begin
    if (reset) begin
        gx <= '0; gy <= '0;
        abs_gx <= '0; abs_gy <= '0;
        edge_mag <= '0;
    end else begin
        gx <= $signed({1'b0,g02})
            + $signed({1'b0,(g12<<1)})
            + $signed({1'b0,g22})
            - ( $signed({1'b0,g00})
            +   $signed({1'b0,(g10<<1)})
            +   $signed({1'b0,g20}) );

        gy <= $signed({1'b0,g20})
            + $signed({1'b0,(g21<<1)})
            + $signed({1'b0,g22})
            - ( $signed({1'b0,g00})
            +   $signed({1'b0,(g01<<1)})
            +   $signed({1'b0,g02}) );

        abs_gx   <= gx[10] ? (~gx + 11'd1) : gx;
        abs_gy   <= gy[10] ? (~gy + 11'd1) : gy;
        edge_mag <= abs_gx + abs_gy;
    end
end
```

$$G_x = (g02 + 2g12 + g22) - (g00 + 2g10 + g20), \quad G_y = (g20 + 2g21 + g22) - (g00 + 2g01 + g02)$$

$$\text{edge\_mag} = |G_x| + |G_y|$$

- Edge overlay

```
logic [4:0] out_r5;
logic [5:0] out_g6;
logic [4:0] out_b5;

always_ff @(posedge clk) begin
    if (reset) begin
        out_r5 <= '0;
        out_g6 <= '0;
        out_b5 <= '0;
    end else if (at_border_d3) begin
        // 경계: 원본 패스스루 (raw_pixel_d3와 정렬)
        out_r5 <= raw_pixel_d3[15:11];
        out_g6 <= raw_pixel_d3[10:5];
        out_b5 <= raw_pixel_d3[4:0];
    end else begin
        if (edge_mag >= EDGE_THR) begin
            out_r5 <= 5'd0;
            out_g6 <= 6'd0;
            out_b5 <= 5'd0;  // black edge
        end else begin
            out_r5 <= post_r5;
            out_g6 <= post_g6;
            out_b5 <= post_b5;  // flat color
        end
    end
end

assign wData_out = {out_r5, out_g6, out_b5};
```

$$\text{out} = \begin{cases} \text{orig} & \text{border} \\ (0,0,0) & \text{edge\_mag} \geq T \\ \text{post} & \text{otherwise} \end{cases}$$

Cartoon Filter

Before

After

평탄한 면 + 또렷한 경계 → 블러+양자화로 면 형성, 엣지로 경계선을 강조

Pop-Art Filter

Posterization

RGB 채널 상위 비트만 유지 → 색상 영역 단순화

Color Boost

원색 강조 (채도 극대화)

```
wire [4:0] post_r5;
wire [5:0] post_g6;
wire [4:0] post_b5;

assign post_r5 = {in_r5[4:3], 3'b000};
assign post_g6 = {in_g6[5:4], 4'b0000};
assign post_b5 = {in_b5[4:3], 3'b000};

logic [5:0] boost_g6;
logic [4:0] boost_b5;

always_comb begin
    boost_r5 = (post_r5 << 1 > 5'd31) ? 5'd31 : (post_r5 << 1);
    boost_g6 = (post_g6 << 1 > 6'd63) ? 6'd63 : (post_g6 << 1);
    boost_b5 = (post_b5 << 1 > 5'd31) ? 5'd31 : (post_b5 << 1);
end
```

Pop-Art Filter

```
    logic signed [7:0] gx, gy;
    logic [7:0] abs_gx, abs_gy;
    logic [8:0] edge_mag;

    always_ff @(posedge clk) begin
        if (reset) begin
            gx <= '0;
            gy <= '0;
            abs_gx <= '0;
            abs_gy <= '0;
            edge_mag <= '0;
        end else if (we_in) begin
            gx <= $signed(
                in_g6
            ) - $signed(
                in_r5
            );
            gy <= $signed(in_g6) - $signed(in_b5);

            abs_gx <= gx[7] ? (~gx + 8'd1) : gx;
            abs_gy <= gy[7] ? (~gy + 8'd1) : gy;
            edge_mag <= abs_gx + abs_gy;
        end
    end
```

Edge Detection(Sobel edge on G)

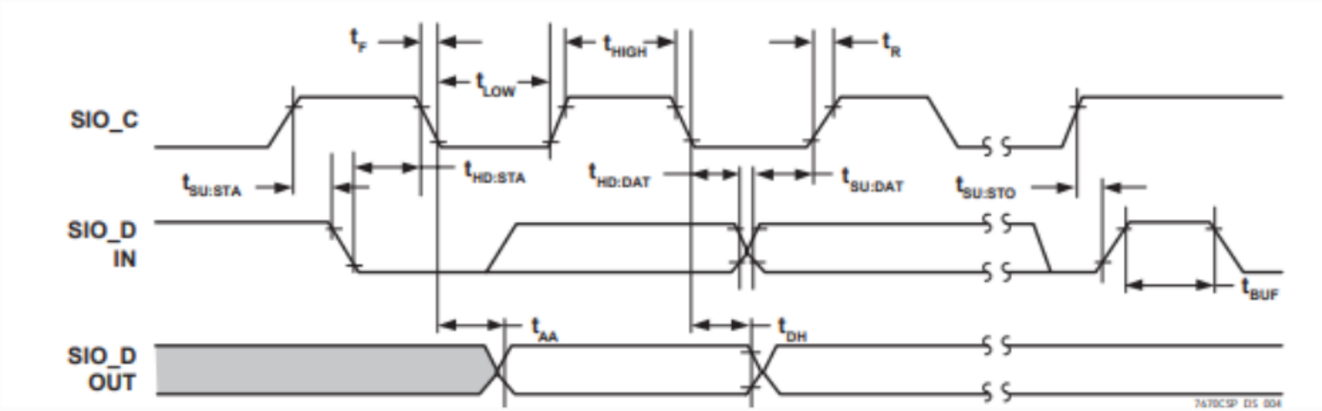
Sobel-like 연산으로 윤곽선 검출

```
    logic [4:0] out_r5;
    logic [5:0] out_g6;
    logic [4:0] out_b5;

    always_ff @(posedge clk) begin
        if (reset) begin
            out_r5 <= '0;
            out_g6 <= '0;
            out_b5 <= '0;
        end else if (we_in) begin
            if (edge_mag > EDGE_THR) begin
                out_r5 <= 5'd0;
                out_g6 <= 6'd0;
                out_b5 <= 5'd0;
            end else begin
                out_r5 <= (boost_r5 > 5'd0) ? boost_r5 : 5'd31;
                out_g6 <= (boost_g6 > 6'd0) ? boost_g6 : 6'd63;
                out_b5 <= (boost_b5 > 5'd0) ? boost_b5 : 5'd31;
            end
        end
    end

    assign wData_out = {out_r5, out_g6, out_b5};
}
```

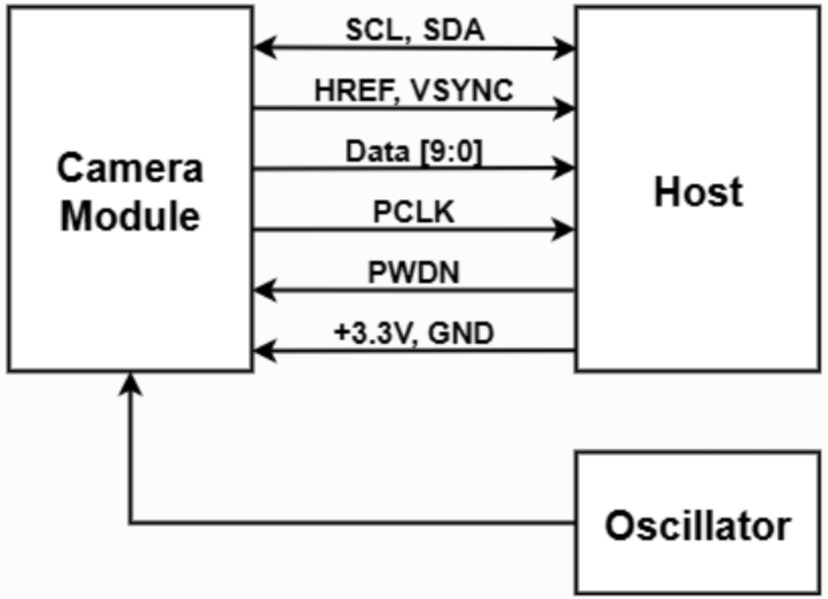
Output Mapping

밝은 배경 위에 원색 블록 + 블랙 윤곽선

SCCB

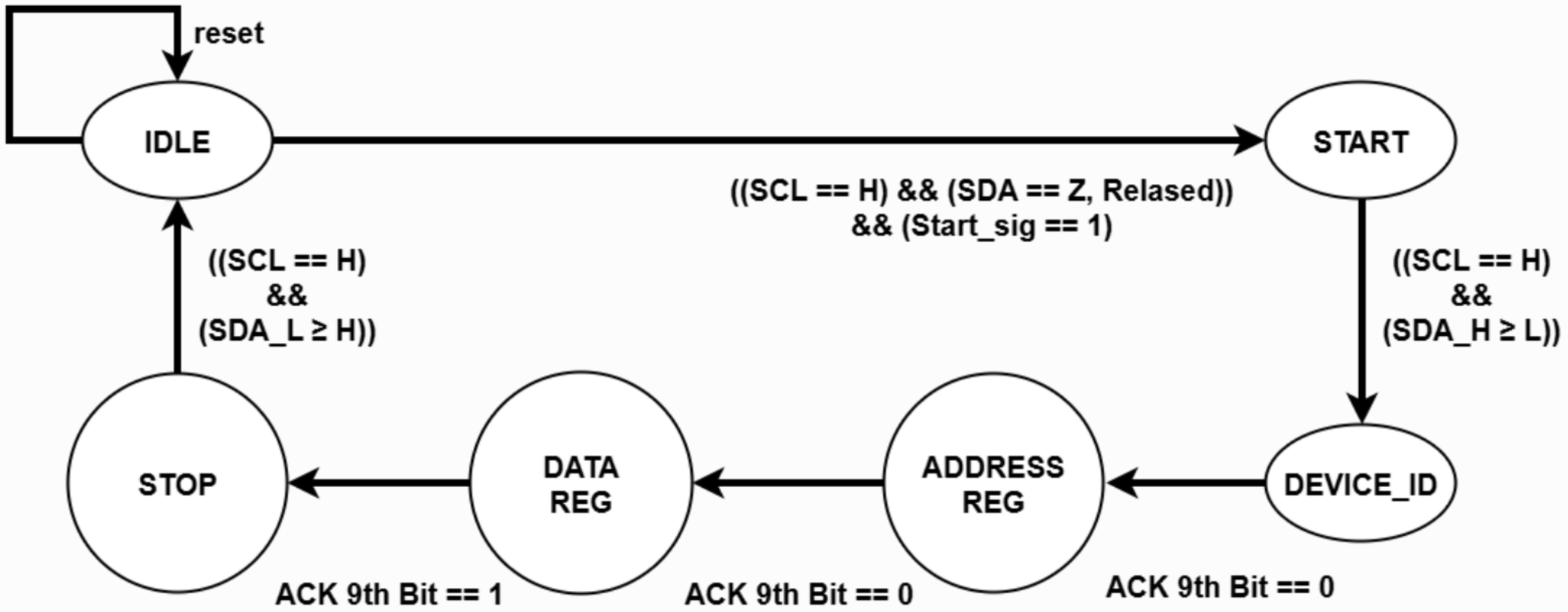


▲ SCCB Protocol Timing Diagram



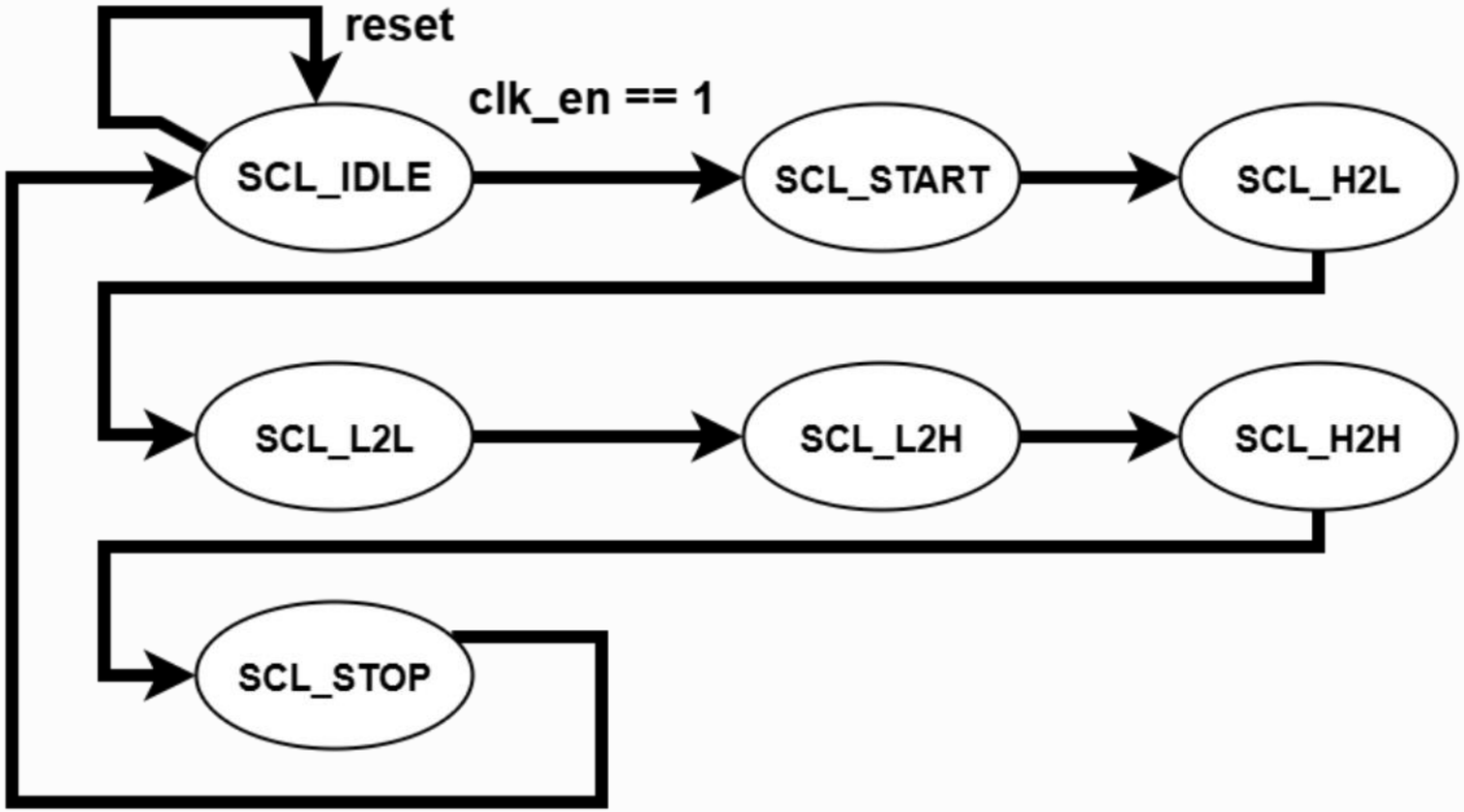
▲ SCCB Block Diagram

SCCB Transaction Level



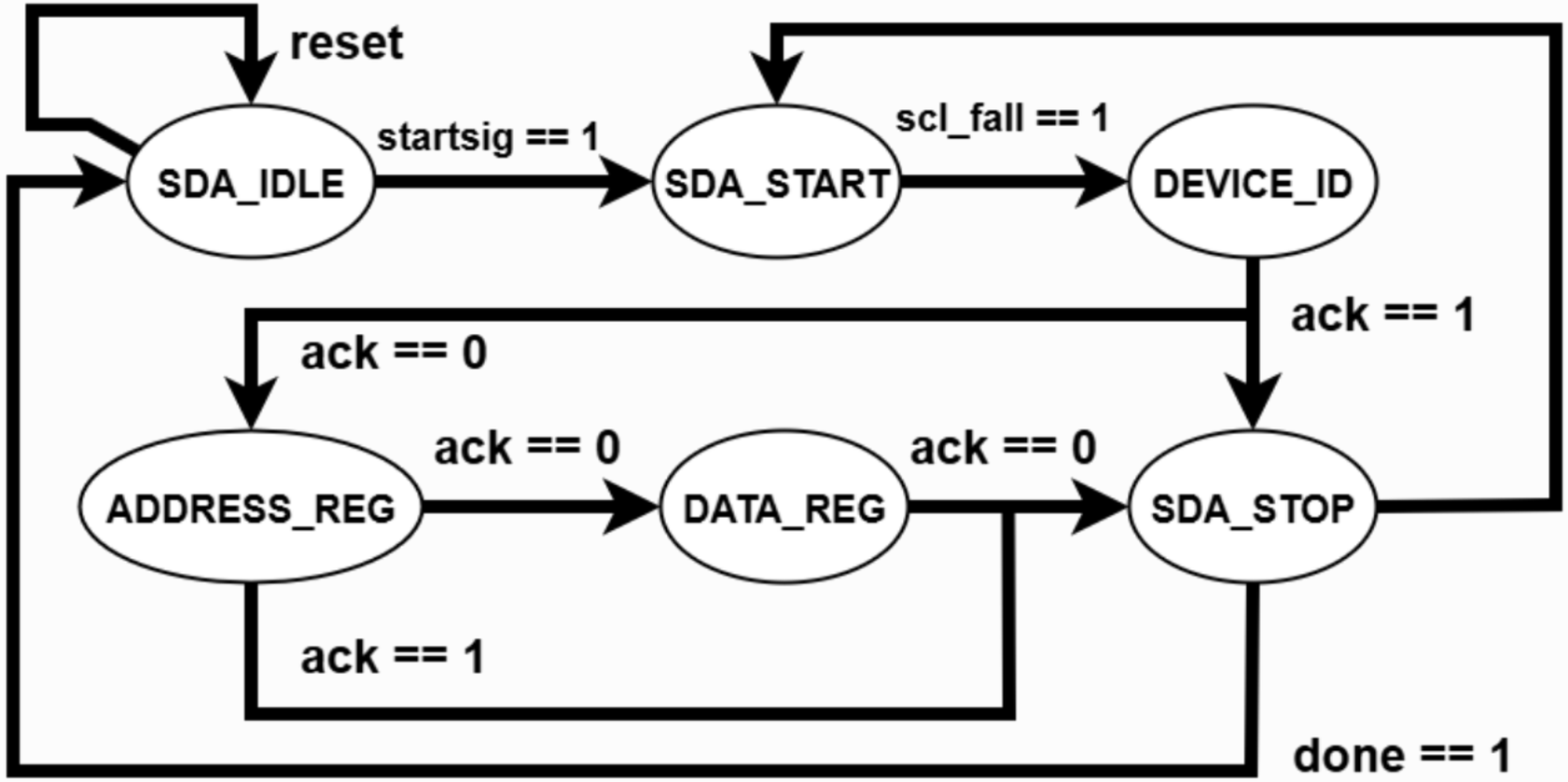
▲ SCCB Protocol Byte-level FSM

SCCB SCL FSM



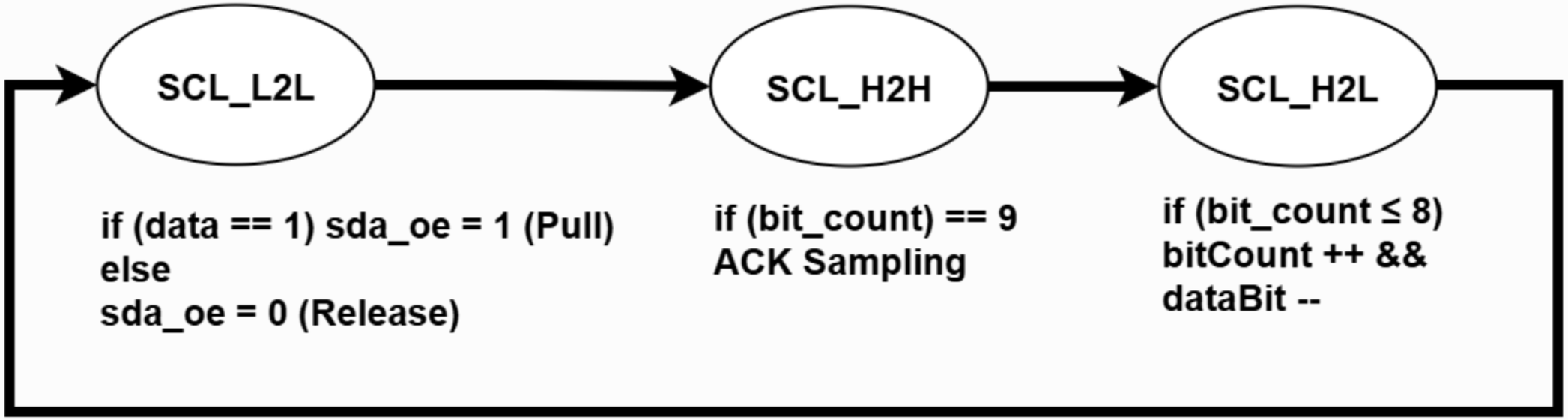
▲ SCCB Protocol Bit-level SCL FSM

SCCB SDA FSM



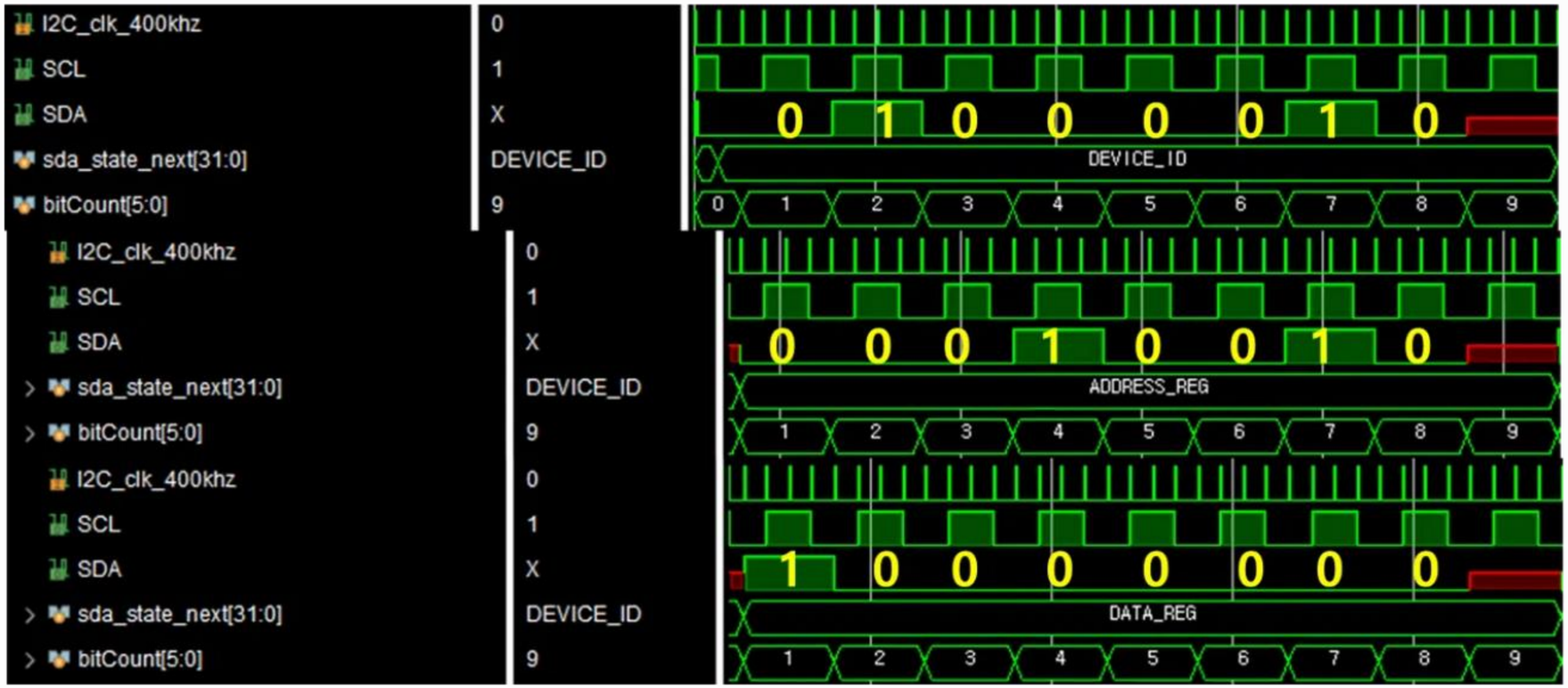
▲ SCCB Protocol Byte-level SDA FSM

SCCB SDA Bit Level FSM



▲ SCCB Protocol Bit-level SDA FSM

SCCB Verification (Device Address 0x42)



OV7670_config_rom=> 0: dout <= 16'h12_80;

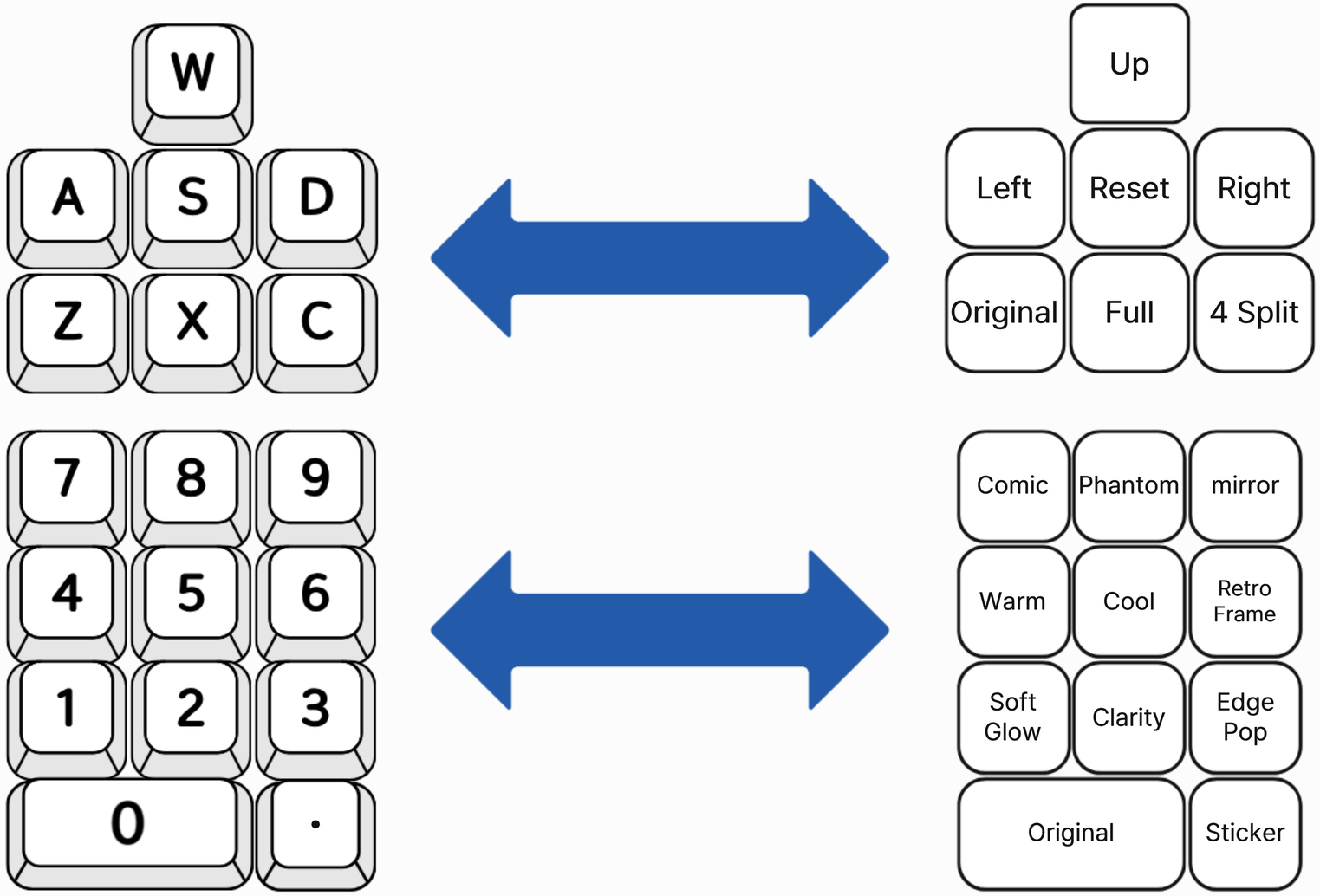
▲ SCCB Protocol Transaction Level Timing

SCCB Verification (Device Address 0x42)

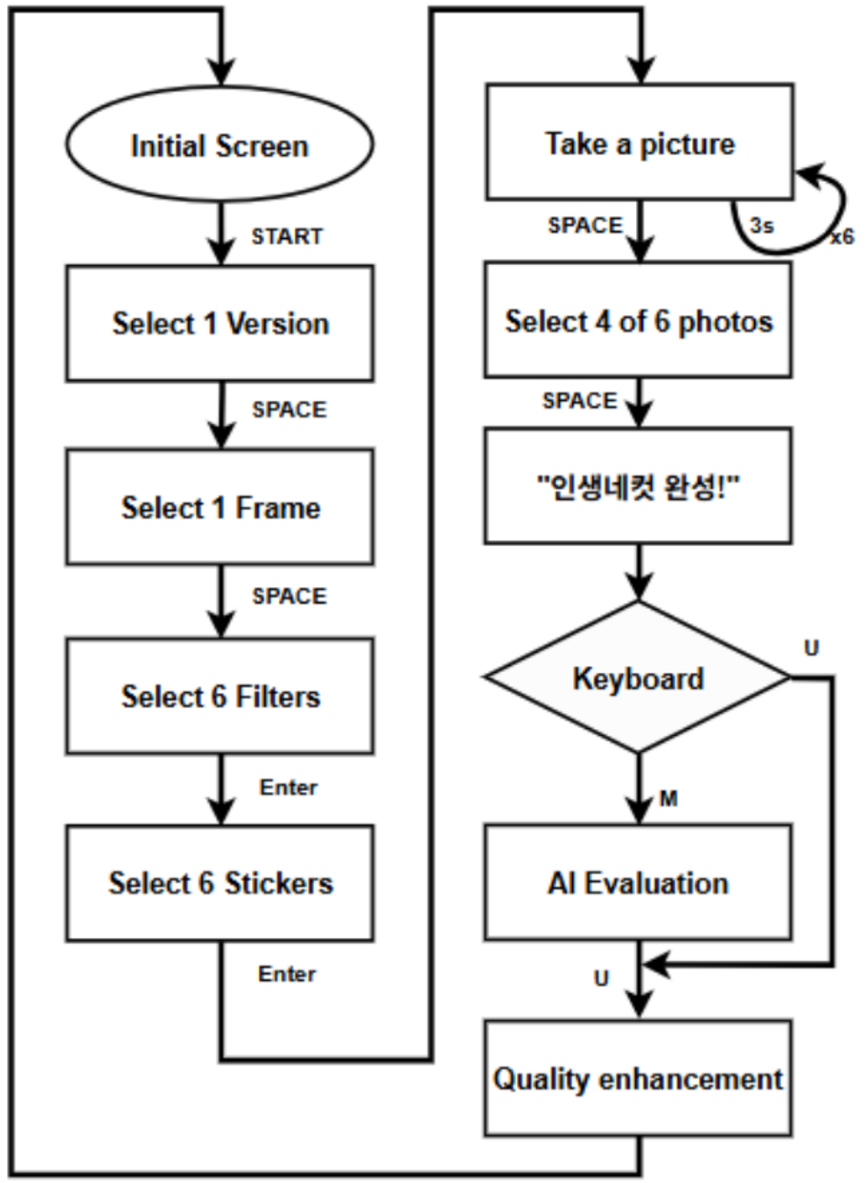
▲ SCCB Protocol Bit Level Timing (1's and 0's)

▲ SCCB Protocol Bit Level Timing (9th Bit Release)

UART Signal


W
A
S
D
Z
X
C
7
8
9
4
5
6
1
2
3
0
.
Up
Left
Reset
Right
Original
Full
4 Split
Comic
Phantom
mirror
Warm
Cool
Retro Frame
Soft Glow
Clarity
Edge Pop
Original
Sticker

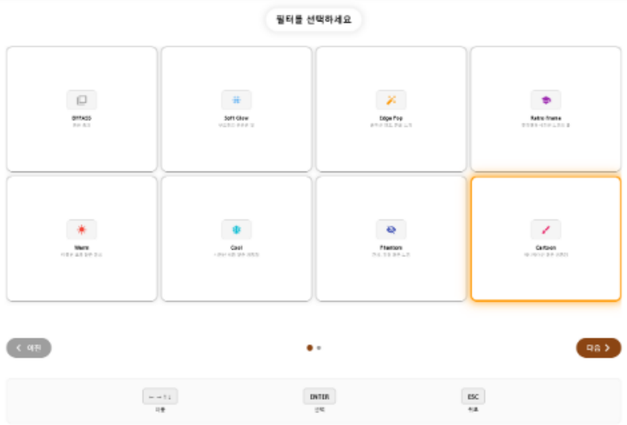
Flow Chart

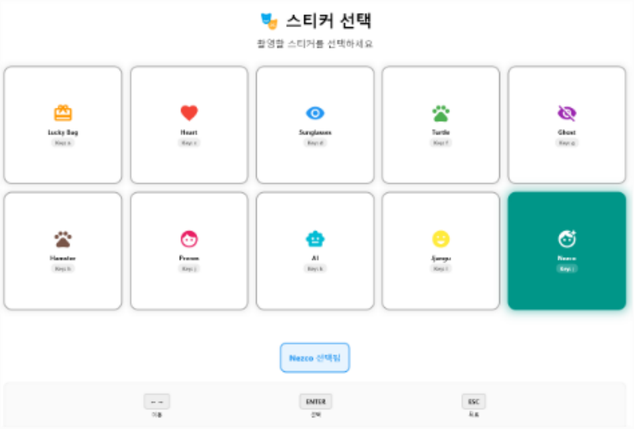

Initial Screen
START
Select 1 Version
SPACE
Select 1 Frame
SPACE
Select 6 Filters
Enter
Select 6 Stickers
Enter
Take a picture
SPACE
3s
x6
Select 4 of 6 photos
SPACE
"인생네컷 완성!"
Keyboard
U
M
AI Evaluation
U
Quality enhancement

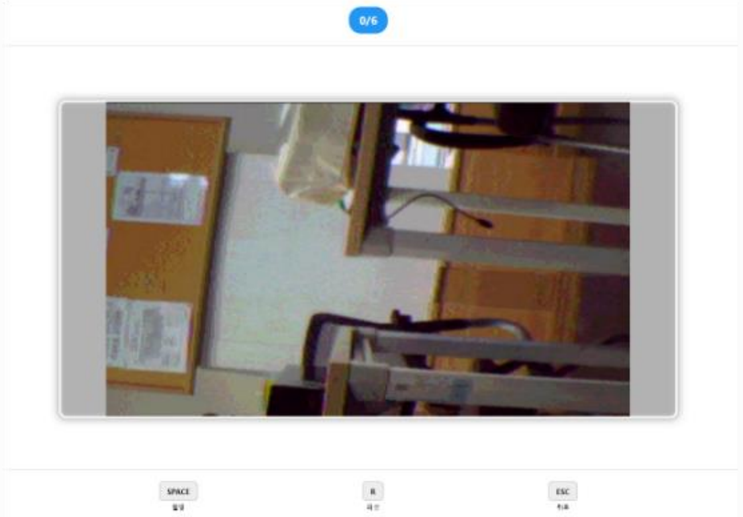
GUI


프레임 규격 선택
가로 4컷
세로 4컷
ENTER
ESC


프레임을 선택하세요
스페이스바로 선택
SPACE
ESC

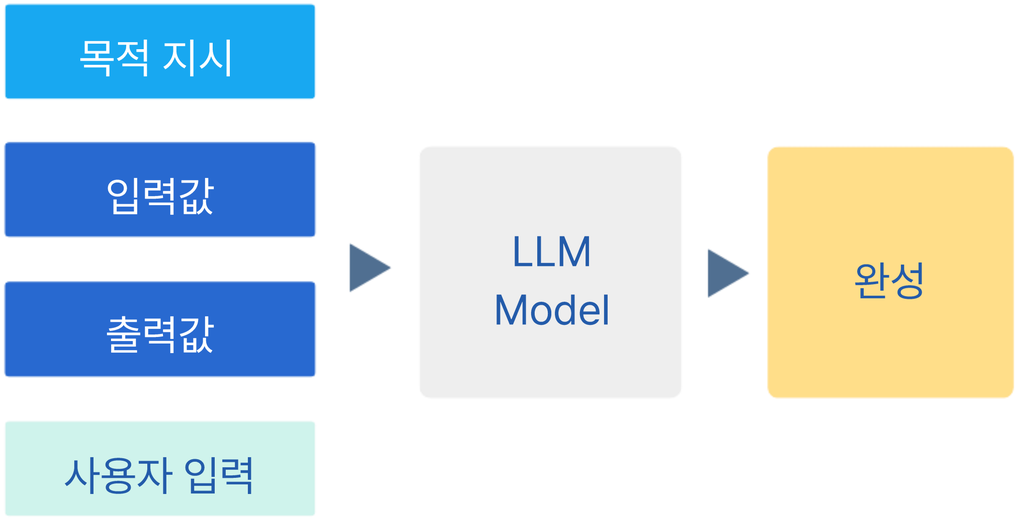

필터를 선택하세요
ENTER
ESC


스티커 선택
Lucky Bag
Heart
Sunglasses
Turtle
Ghost
Hamster
Frozen
AI
Nezco
Nezco 선택됨
ENTER
ESC


0/6
SPACE
R
ESC


인생네컷 완성!
지장 원료: LifeFourCut_frame1_20250922_150944.png
스페이스바를 눌러 완료!
M키: AI 감성평 보기
U키: AI 화신 개선
ESC: 취소하기

Third – Party API from Server

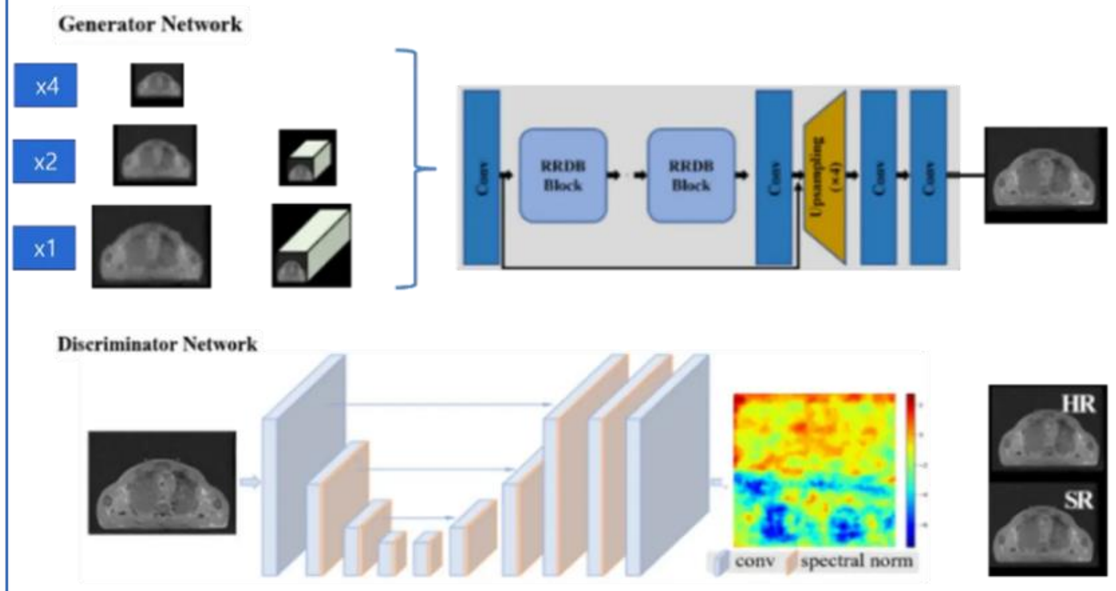


사용 모델

Gemini

이미지 분석	촬영물 코멘트 생성	Google 생성형 AI	gemini 1.5-flash
기능	용도	API	모델

Third – Party API from Server

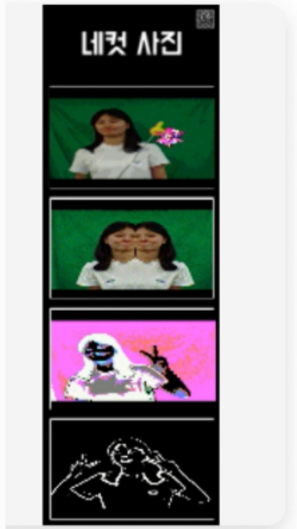


사용 모델

Real-ESRGAN
PRACTICAL RESTORATION

기능	용도	API
이미지 업스케일링	4X 스케일링	WaveSpeed RealESRGAN

AI Result



$1

• $1 자연스러운 포즈와 표정이 좋습니다. 하지만 배경이 너무 단색이라 조금 심심해요. 소품 활용은 좋았습니다. 녹색 배경과 옷 색깔의 조화는 조금 더 신경 써볼만 합니다.
• $1 좌우 대칭 효과는 독특하고 재밌는 시도입니다. 하지만 약간 어색한 부분이 있어요. 얼굴이 완벽하게 대칭되지 않아서 오히려 인위적으로 느껴질 수 있습니다.
• $1 핑크색 배경과 필터 효과는 개성이 강하지만, 사진이 너무 깨져서 디테일이 많이 손실되었어요. 필터 효과의 강도를 조절해야 합니다.
• $1 흑백 실루엣 사진은 분위기가 있지만, 역시 디테일이 부족하고, 피사체의 윤곽이 명확하지 않아서 아쉬움이 남습니다. 조명 조절을 통해 더욱 선명하고 드라마틱한 효과를 낼 수 있었을 거예요.

$1

• $1 단색 배경 대신 다양한 패턴이나 소품을 활용하여 더욱 풍성하고 시각적으로 즐거운 사진을 만들어 보세요.
• $1 과도한 필터 효과는 오히려 사진의 질을 떨어뜨릴 수 있습니다. 필터 효과의 강도를 조절하거나, 다른 필터를 시도해보는 것을 추천합니다.
• $1 다양한 포즈와 표정을 연습하여 사진에 생동감을 더해보세요. 자신에게 가장 잘 어울리는 표정과 포즈를 찾는 것이 중요합니다.
• $1 조명의 위치와 강도에 따라 사진의 분위기가 크게 달라집니다. 다양한 조명을 활용하여 사진의 퀄리티를 높여보세요.
• $1 사진의 해상도를 높여 선명하고 깨끗한 사진을 얻도록 노력해주세요.

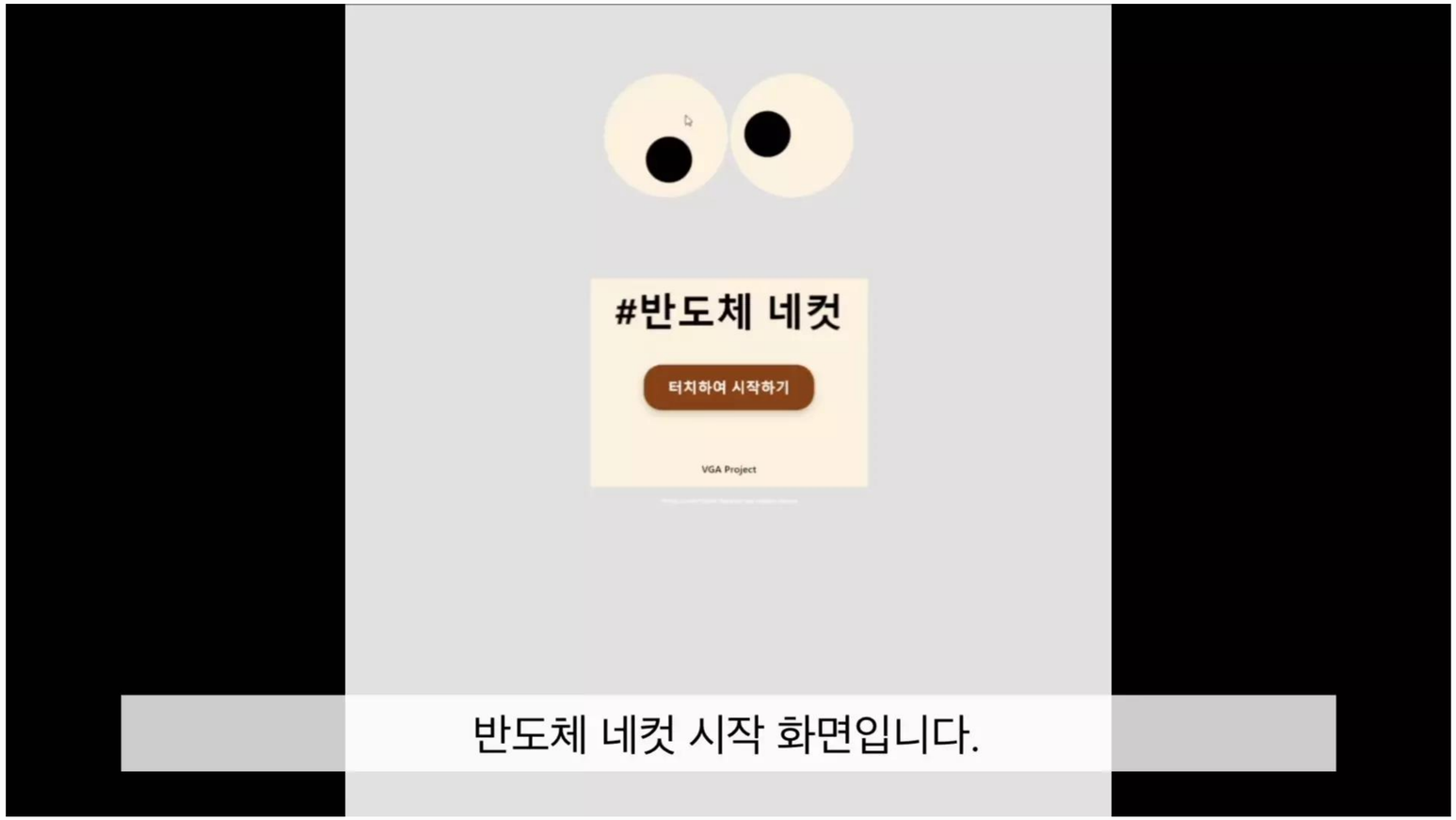
▲ Gemini 1.5 Flash – 멀티모달 이미지 분석 모델

▲ Real-ESRGAN – 딥러닝 기반 모델

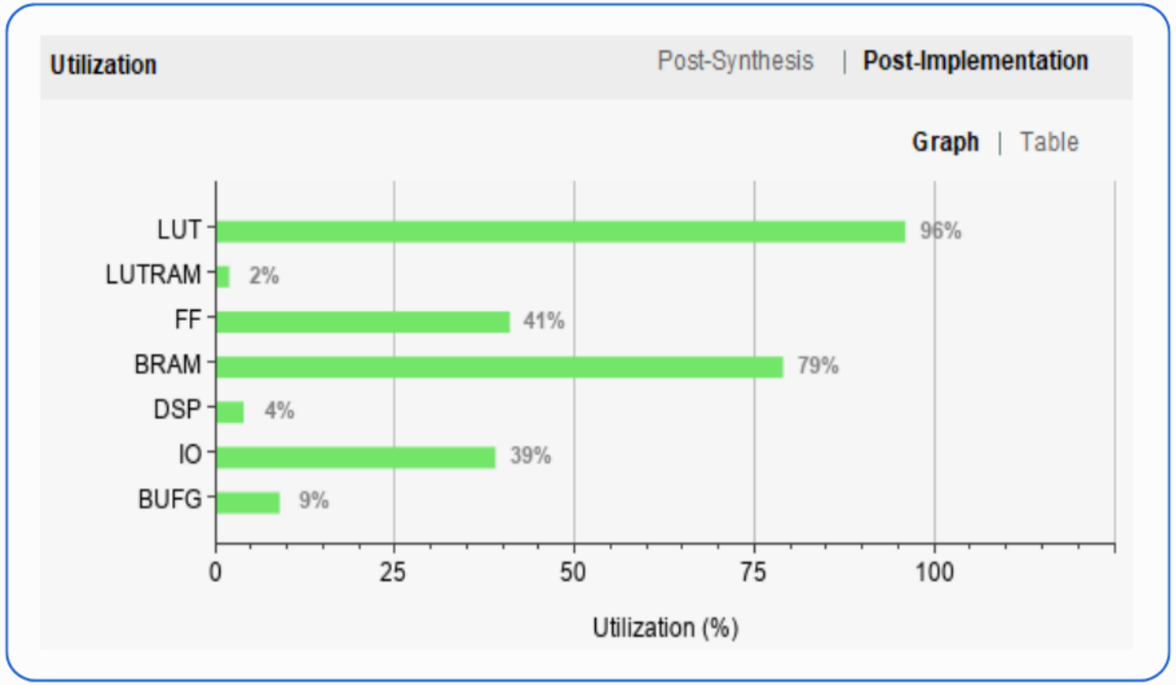
사진 코멘트 생성을 위한 Google Gen AI API 활용

고해상도 이미지 업스케일링

데모 영상

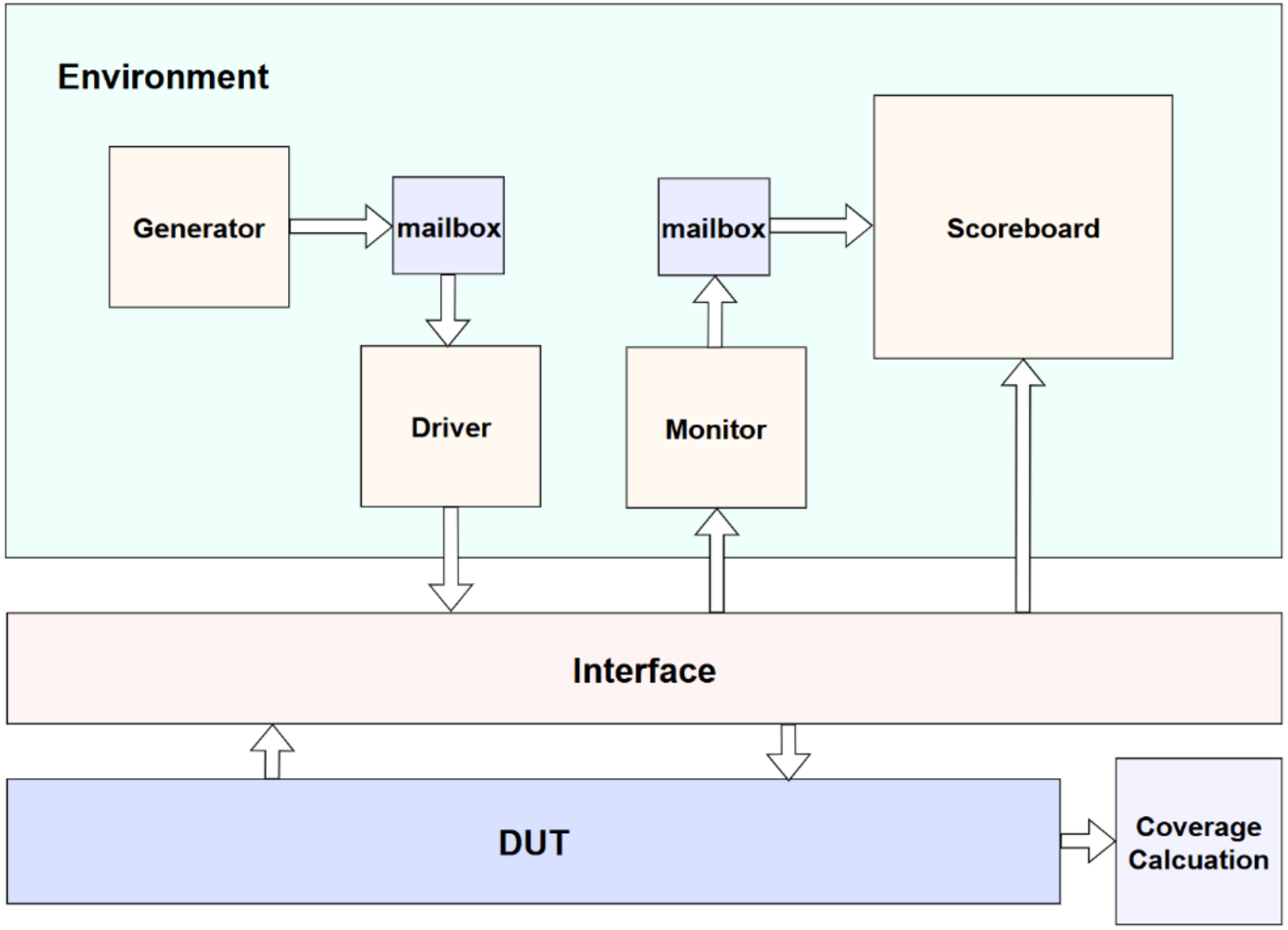


자원 활용



LUT	필터 및 멀티미디어 처리 연산 → 논리 자원 대부분 활용
FF	파이프라인 및 제어 로직에 사용
BRAM	Frame Buffer 저장
DSP	일부 연산(필터 연산 등) 사용
IO	VGA, 카메라 인터페이스 핀 점유
BUFG	클럭 분배에 사용

SystemVerilog 검증 구조



Block Diagram

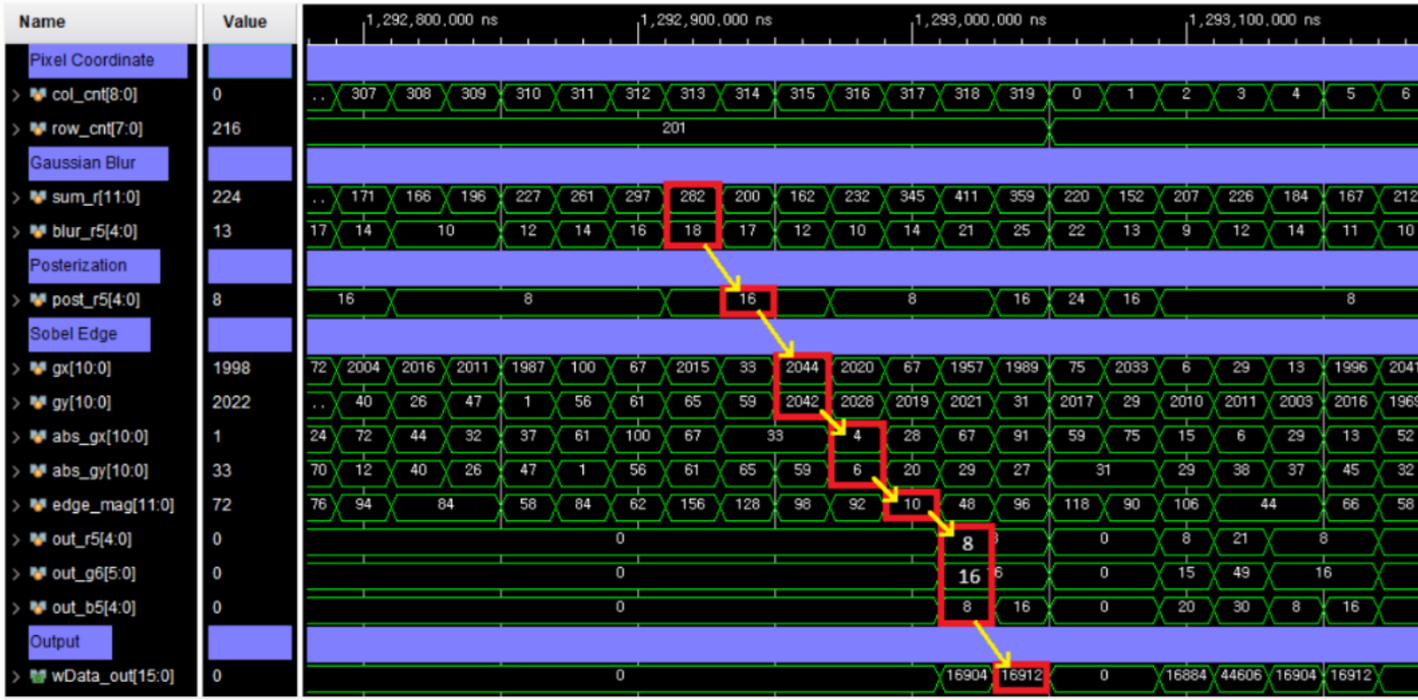
SystemVerilog (Cartoon Filter 검증)

Input

10	4	27
30	13	2
11	14	0

Weight

1	2	1
2	4	2
1	2	1




```
========================[COVERAGE]========================
Simulation finished at 4000230000
Border         : 100.00%
Edge Magnitude: 83.33%   ← Edge 여부 판정 (EDGE_THR = 50)
Orientation    : 100.00%
==========================================================
```

Trouble shooting : SCCB 검증

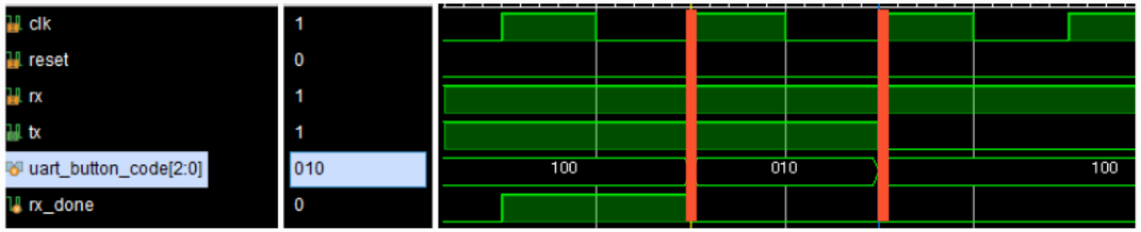
```
if (byte_ctr + 1 == 3) begin
    tx_count <= tx_count + 1;
    $display(
        "[%0t ns] TX%0d DEV=0x%02h REG=0x%02h DAT=0x%02h",
            $time, tx_count, dev_id, reg_addr,
        reg_data);
    byte_ctr <= 0;
end
```

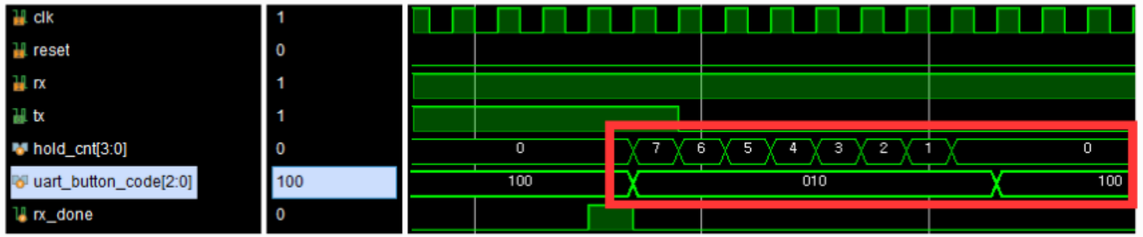
```
if (byte_ctr + 1 == 3) begin
    tx_count <= tx_count + 1;
    $strobe(
        "[%0t ns] TX%0d DEV=0x%02h REG=0x%02h DAT=0x%02h",
        $time, tx_count, dev_id, reg_addr,
        reg_data);
    byte_ctr <= 0;
end
```

```
[268815000 ns] TX0 DEV=0x42 REG=0x12 DAT=0x00
[836075000 ns] TX2 DEV=0x42 REG=0x12 DAT=0xf0
[1119705000 ns] TX3 DEV=0x42 REG=0x11 DAT=0x14
[1403335000 ns] TX4 DEV=0x42 REG=0x0c DAT=0x80
[1686965000 ns] TX5 DEV=0x42 REG=0x3e DAT=0x04
[1970595000 ns] TX6 DEV=0x42 REG=0x04 DAT=0x19
[2254225000 ns] TX7 DEV=0x42 REG=0x40 DAT=0x00
[2537855000 ns] TX8 DEV=0x42 REG=0x3a DAT=0xd0
[2821485000 ns] TX9 DEV=0x42 REG=0x14 DAT=0x04
[3105115000 ns] TX10 DEV=0x42 REG=0x4f DAT=0x18
[3388745000 ns] TX11 DEV=0x42 REG=0x50 DAT=0xb3
[3672375000 ns] TX12 DEV=0x42 REG=0x51 DAT=0xb3
[3956005000 ns] TX13 DEV=0x42 REG=0x52 DAT=0x00
[4239635000 ns] TX14 DEV=0x42 REG=0x53 DAT=0x3d
[4523265000 ns] TX15 DEV=0x42 REG=0x54 DAT=0xa7
[4806895000 ns] TX16 DEV=0x42 REG=0x58 DAT=0xe4
[5000000000 ns] TIMEOUT. tx_count=17
```

```
[268815000 ns] TX1 DEV=0x42 REG=0x12 DAT=0x80
[836075000 ns] TX3 DEV=0x42 REG=0x12 DAT=0x14
[1119705000 ns] TX4 DEV=0x42 REG=0x11 DAT=0x80
[1403335000 ns] TX5 DEV=0x42 REG=0x0c DAT=0x04
[1686965000 ns] TX6 DEV=0x42 REG=0x3e DAT=0x19
[1970595000 ns] TX7 DEV=0x42 REG=0x04 DAT=0x00
[2254225000 ns] TX8 DEV=0x42 REG=0x40 DAT=0xd0
[2537855000 ns] TX9 DEV=0x42 REG=0x3a DAT=0x04
[2821485000 ns] TX10 DEV=0x42 REG=0x14 DAT=0x18
[3105115000 ns] TX11 DEV=0x42 REG=0x4f DAT=0xb3
[3388745000 ns] TX12 DEV=0x42 REG=0x50 DAT=0xb3
[3672375000 ns] TX13 DEV=0x42 REG=0x51 DAT=0x00
[3956005000 ns] TX14 DEV=0x42 REG=0x52 DAT=0x3d
[4239635000 ns] TX15 DEV=0x42 REG=0x53 DAT=0xa7
[4523265000 ns] TX16 DEV=0x42 REG=0x54 DAT=0xe4
[4806895000 ns] TX17 DEV=0x42 REG=0x58 DAT=0x9e
[5000000000 ns] TIMEOUT. tx_count=17
```

Trouble shooting : UART





기존 코드

- 보드의 물리 버튼 입력(btn_left, btn_right 등)을 UART 입력으로 대체
- uart_button_code가 rx_done 시점에만 업데이트되고
- 1 클럭만 유지되는 원샷 펄스 형태
- 반면 물리 버튼은 누르는 동안 계속 High가 유지되므로 정상적으로 인식
- 버튼 신호를 샘플링하는 타이밍에 따라 UART 입력이 무시되는 문제 발생

수정

- 단일 클럭 이벤트 대신 8클럭 동안 유지하도록 수정
- 특정 입력이 들어왔을 때 상태를 일정 기간 유지

  타임아웃 시점에만 idle로 복귀하도록 로직 변경
- hold_cnt 카운터를 추가해 입력이 들어오면 상태를 8클럭 동안 유지

기대 효과

1	차별화된 사용자 경험	• AI 분석 기반 맞춤형 체험 제공 • 11가지 필터 제공으로 기존 인생네컷 대비 차별화
2	고효율 시스템	• FPGA 기반 디지털 촬영으로 GPU 보다 전력 소모 절감
3	지능형 콘텐츠 창출	• 실시간 감정 분석, 포즈 제안 기능 구현
4	문화 체험 공간으로의 확장성	• 포토 체험 공간으로 확장 가능성

역할 분배 | Filter

고종완

- TOP Module 구조화 및 통합
- 영상 처리 필터 설계 및 구현

(Gaussian, Ghost, Graduation)

- SCCB Interface 설계 및 검증

임도엽

- 영상 처리 필터 설계 및 구현

(Sharpen, Sobel)

- 라인버퍼 구조 설계 및 파이프라인 통합
- 모듈 인터페이스 및 타이밍 동기화

김종현

- 영상 처리 필터 설계 및 구현

(Gaussian, Pop Art)

- UART 설계 및 통합
- GUI 구현 및 HW 제작

임재홍

- Filter 설계 및 검증 (Cartoon)
- Top-level 설계
- HW 제작

역할 분배 | GUI

김민규

- 영상 처리 필터 설계 및 구현

(Sobel, Color Detection)

- GUI 구현 및 HW 제작
- 서버 및 API 기능 통합

최규리

- 영상 처리 필터 설계 및 구현

(Emboss, Pixelation, Laplacian)

- GUI 구현 및 HW 제작

서윤철

- 리소스 확보 연구 및 RTL 설계 보조
- SystemVerilog 검증
- GUI 구현 및 HW 제작

개인 소감

김민규

영상 처리 알고리즘을 사전 분석하고 이를 verilog 코드로 구현하는 과정에서 설계 흐름을 이해할 수 있었습니다. 곱셈 자원이 FPGA에 생기는 부담을 해결하기 위해 convolution 연산을 shift나 threshold로 해결하는 등 최적화에 집중할 수 있었습니다.
픽셀 스트림 처리 과정에서 생기는 CDC 문제의 해결을 위해 dual port RAM이나 async fifo 를 사용하는 다양한 테크닉에 대해서도 자신이 생겼습니다.
무엇보다 팀원들과의 유기적인 협업을 통해 혼자서는 어려운 과제들을 함께 해결해나갈 수 있다는 자신감이 생겼습니다.

고종완

이번 AI 시스템반도체설계 수업을 통해 기존에 알고 있던 내용을 체계적으로 정리하고, 부족했던 부분을 보완할 수 있었습니다.
또한 팀원들과 아이디어를 공유하고 피드백을 주고받으며 협업하는 자세를 익혔습니다.
이러한 경험을 바탕으로 앞으로 현업에서도 원활한 소통과 협업을 통해 팀과 조직에 기여할 수 있는 엔지니어로 성장하고자 합니다.

개인 소감

김종현

영상 처리와 필터 설계 역량을 키울 수 있었고 구현 과정에서 예상치 못한 문제를 해결하기 위해 서로의 아이디어를 나누고 조율하는 경험을 통해 협업의 중요성을 느낄 수 있었습니다.
팀과 함께 성장하고 기여할 수 있는 엔지니어가 되기 위해 꾸준히 노력하겠습니다.

서윤철

개별 모듈을 완성하는 데서 나아가, 하나의 시스템을 구축하는 과정에서 협업의 중요성을 다시 한번 느낄 수 있었습니다.
팀원들과의 소통을 통해 더 많은 것을 배웠고, 앞으로도 함께 일하고 싶은 동료이자 성장하는 엔지니어가 되기 위해 노력하겠습니다.

개인 소감

임도엽

영상 처리 필터를 하드웨어로 구현하는 과정에서 라인버퍼 설계, 타이밍 동기화, 자원 최적화 등 여러 부분을 고민하며 많이 성장할 수 있었습니다.
큰 규모의 프로젝트를 진행하면서 팀원들과 협업해 문제를 해결하는 과정에서 소통의 중요성을 다시 느꼈고, 초기 구조를 함께 정하고 역할을 분담하니 효율적으로 개발을 진행할 수 있었습니다.
이번 경험을 통해 FPGA 설계에 대한 자신감과 앞으로 스스로 구조를 설계·개선할 수 있다는 동기부여를 얻었습니다.

임재홍

필터 구현에 필요한 알고리즘을 하드웨어 환경에 맞게 설계하는 과정을 통해 자원 효율성과 연산 속도를 고려한 설계 능력을 기를 수 있었습니다.
통합 과정에서 발생한 타이밍 불일치 문제를 직접 분석하고 수정하면서, 실제 상황에서 문제를 해결하는 역량을 강화할 수 있었습니다.
개인의 능력도 중요하지만 양질의 결과물을 위해서는 팀원과의 원활한 소통과 협업이 필수적임을 깨달았습니다.

개인 소감

최규리

다양한 영상 처리 필터 설계를 통해 영상 처리 기술 역량을 향상 했습니다. 또한, GUI를 설계할 때 사용자 테스트를 진행하면서 사용자 중심의 인터페이스를 고민했던 경험이었습니다. 프로젝트를 진행하는 과정에서 팀원들과 함께 아이디어를 제시하고 피드백을 공유하면서 협업 하는 자세를 배울 수 있었습니다.

Q & A

THANK YOU

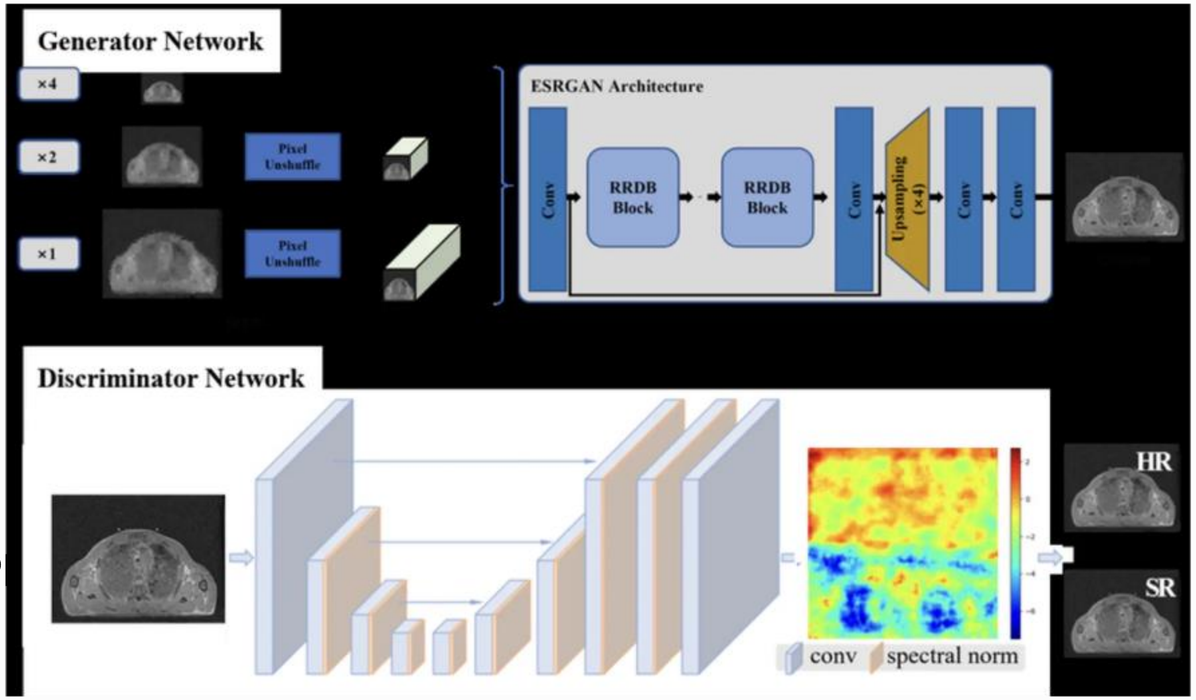
2025 AI 시스템 반도체 설계 2기

AI

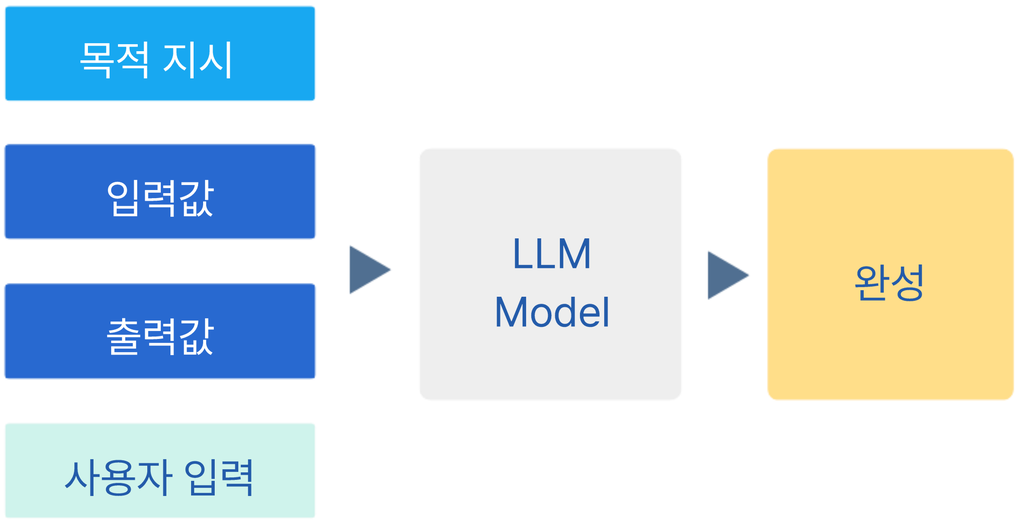
Real-ESRGAN
PRACTICAL RESTORATION

모델: wave_form_ai RealESRGAN
기능: 이미지 업스케일링
(Super-Resolution)
용도: 4X 스케일링
900×540 → 3600×2160

API: WaveSpeed.ai RealESRGAN AP



AI



사용 모델

Gemini

이미지 분석	촬영물 코멘트 생성	Google Generative
기능	용도	API