Group 41

TSIU03 - Final Presentation

Niklas Blomqvist, Philip Johansson, Matteus Laurent,
Johan Levinsson, Oscar Petersson, Erik Peyronson

November 10, 2015

Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

Introduction

Audio signal processing

Signal level indicator

Our own flavor:

- Background image
- Graphic scales for L/R input/output channels
- Class-D amplifier

Group 41

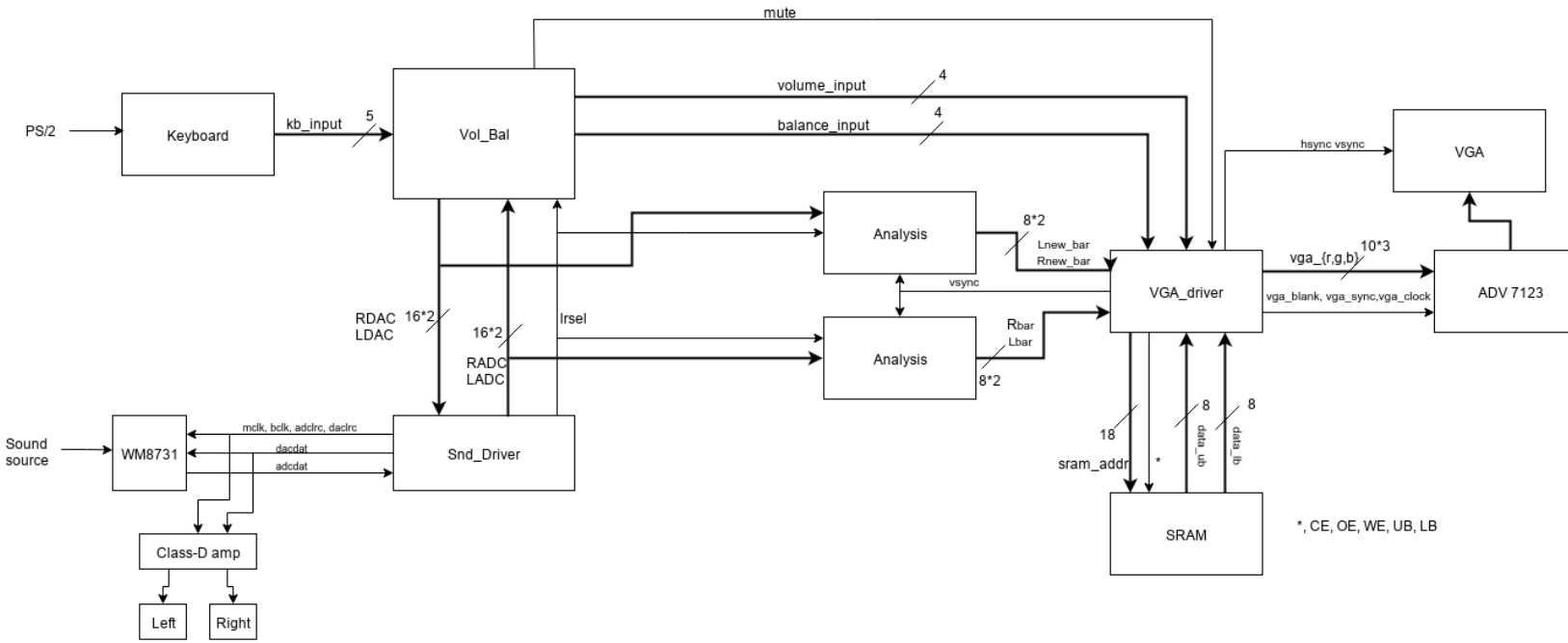
Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

UI

Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

Overview



Keyboard, Vol_Bal, Snd_Driver, Analysis, VGA_Driver

Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

Keyboard Decoding

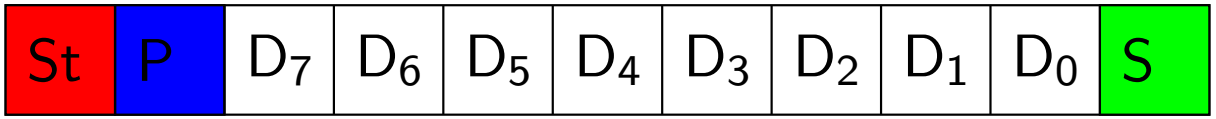
Scan Code Detection

Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

Keyboard Decoding

Scan Code Detection

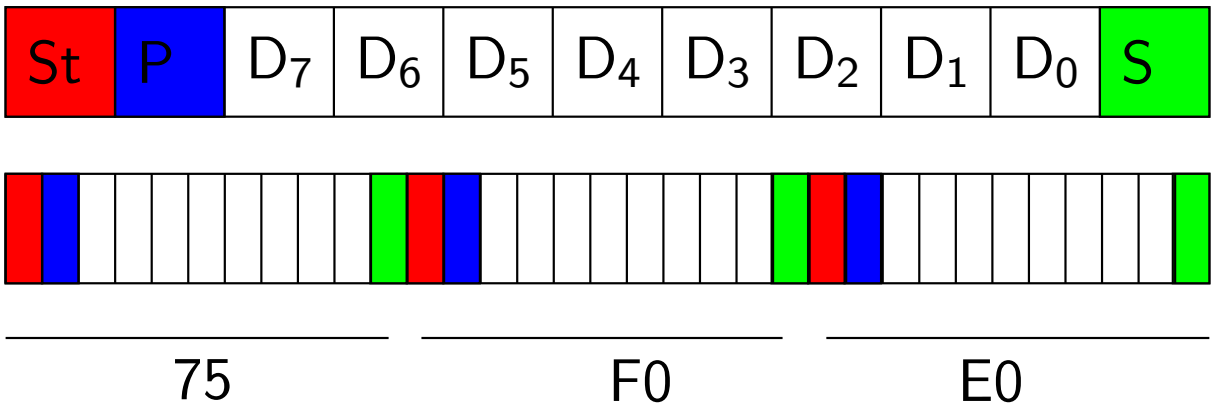


Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

Keyboard Decoding

Scan Code Detection

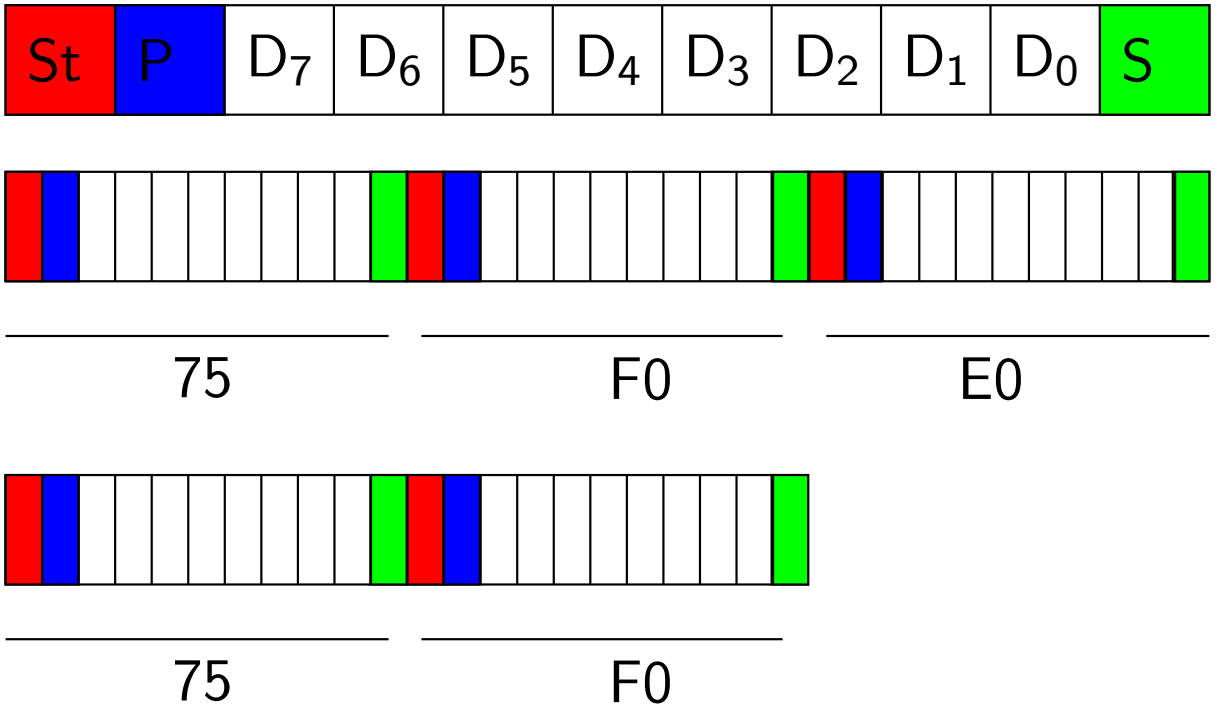


Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

Keyboard Decoding

Scan Code Detection

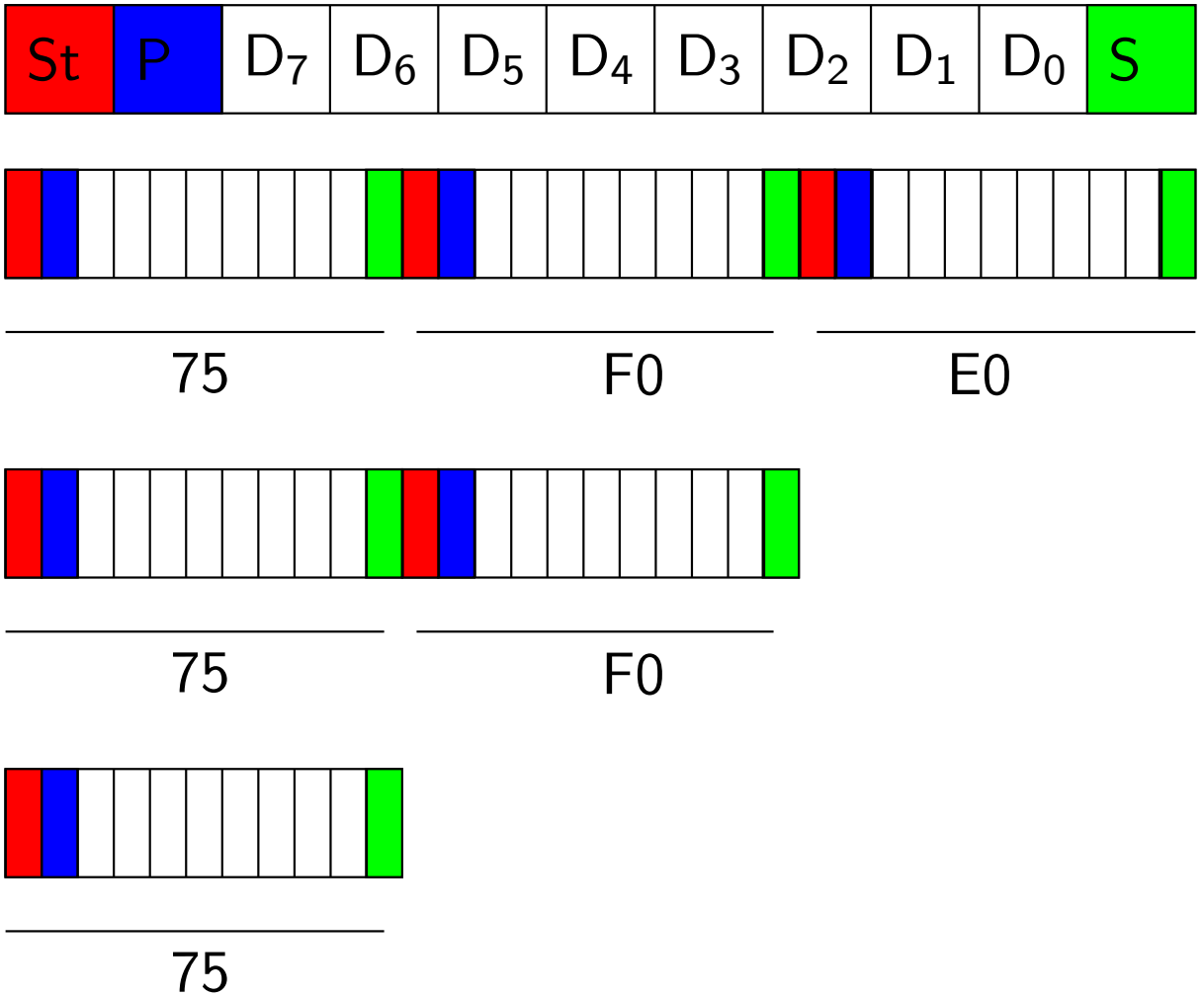


Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

Keyboard Decoding

Scan Code Detection

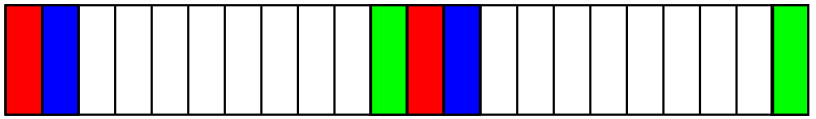


Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

Keyboard Decoding

Scan Code Detection



0

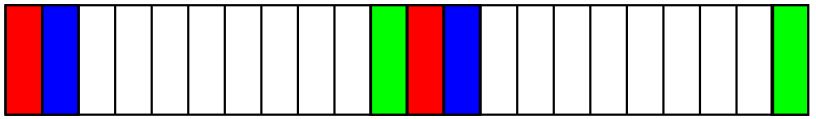
75 F0 BREAKSET

Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

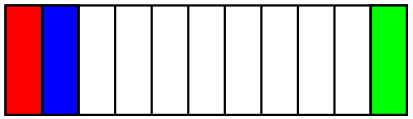
Keyboard Decoding

Scan Code Detection



0

75 F0 BREAKSET



1

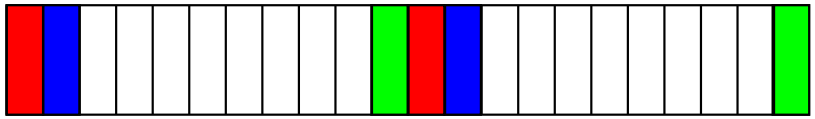
75 BREAKSET

Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

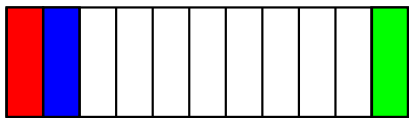
Keyboard Decoding

Scan Code Detection



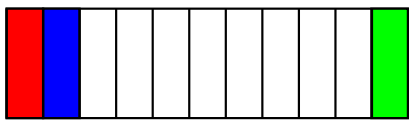
0

75 F0 BREAKSET



1

75 BREAKSET



0

NN BREAKSET

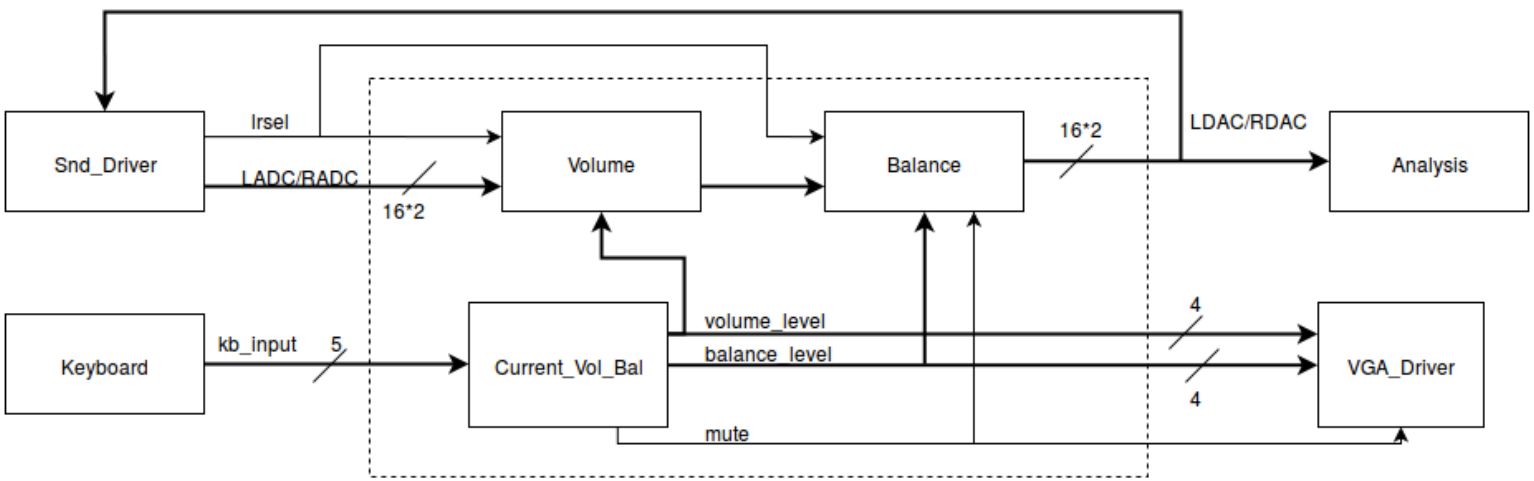
Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

Volume and Balance adjustment

Audio signal adjustment

Storage of system levels



Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

Volume and Balance adjustment

Current_Vol_Bal

Volume: 0 to 10

Balance: -8 to 8

Legality example:

`i_volume`:	1	0	1	0		(10 - lowest volume)
`kb_input`:	0	0	1	0	0	(lower volume)
`i_kb_input`:	0	0	0	0	0	(result: do nothing)

Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

Volume and Balance adjustment

Volume_Adjustment

Logarithmic scaling

$$A_{adj} = A_{in} \cdot (1/\sqrt{2})^n$$

Output range: -30 to 0 dB

Implemented using a state machine

Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

Volume and Balance adjustment

Balance_Adjustment

Linear scaling

$$A_{l\_out} = \frac{8 - m}{8} \cdot A_{l\_adj} \quad , \; A_{l\_out} = A_{l\_adj} \; for \; m < 0$$

$$A_{r\_out} = \frac{8 - |m|}{8} \cdot A_{r\_adj} \quad , \; A_{r\_out} = A_{r\_adj} \; for \; m > 0$$

Controlled by `volume_done` and `lrsel`.

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

Analysis

Analyzes incoming samples, low-pass filtering them.

Output in form of a natural number, which determines the height of the bars.

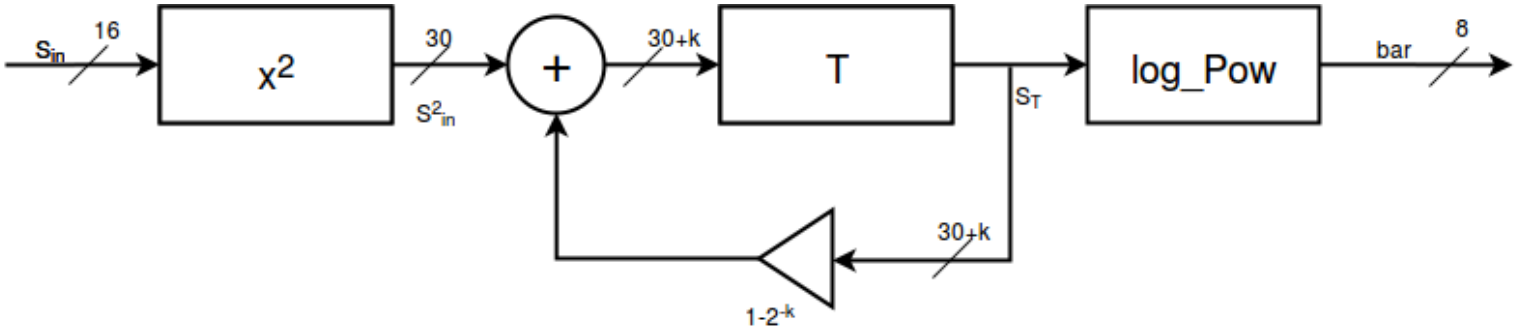
Updates in sync with vsync.

Both left & right channel separately.

Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

Analysis



The low-pass filter. k is chosen by the approximation

$$\frac{1}{10}\ \mathrm{s} = 2^k \cdot \frac{1}{48800} \Rightarrow 2^k = 4880 \approx 2^{12} \Rightarrow k = 12$$

Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

VGA_driver

Intro

Based on Laboration 3.

Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

VGA_driver

Intro

Based on Laboration 3.

Two new submodules.

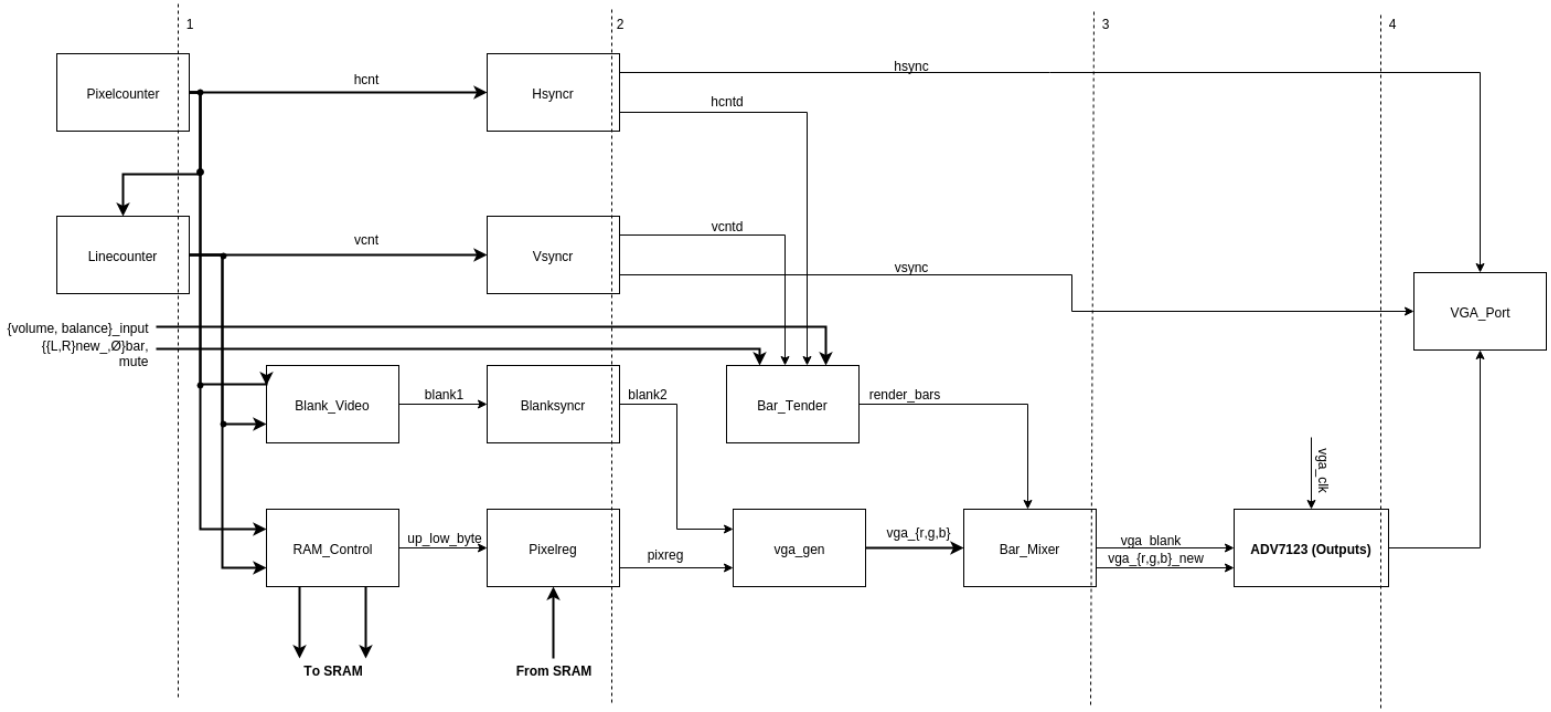
- bar_tender.
- bar_mixer.

Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

VGA_driver

Vga_driver overview

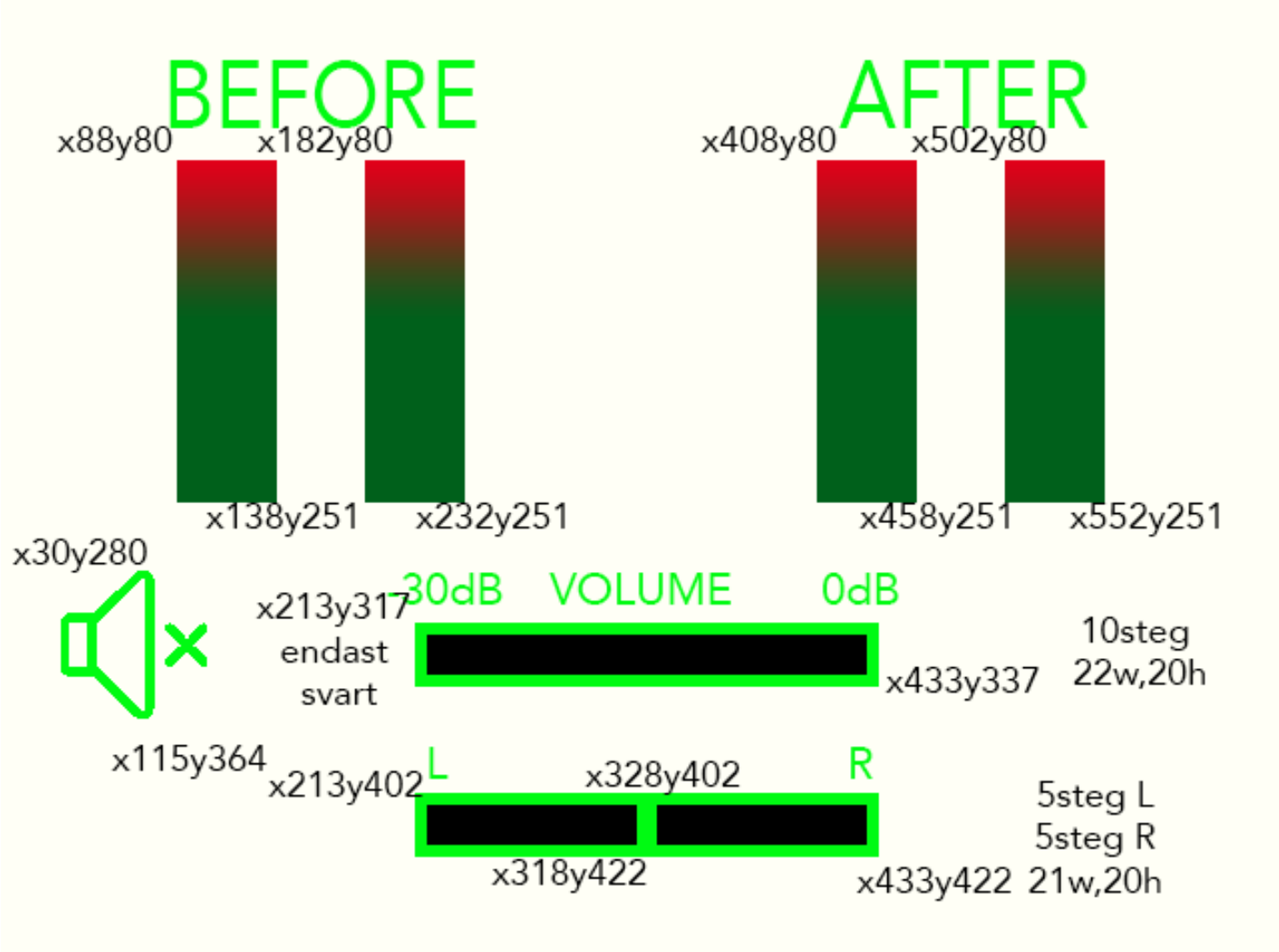


Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

VGA_driver

Concept UI

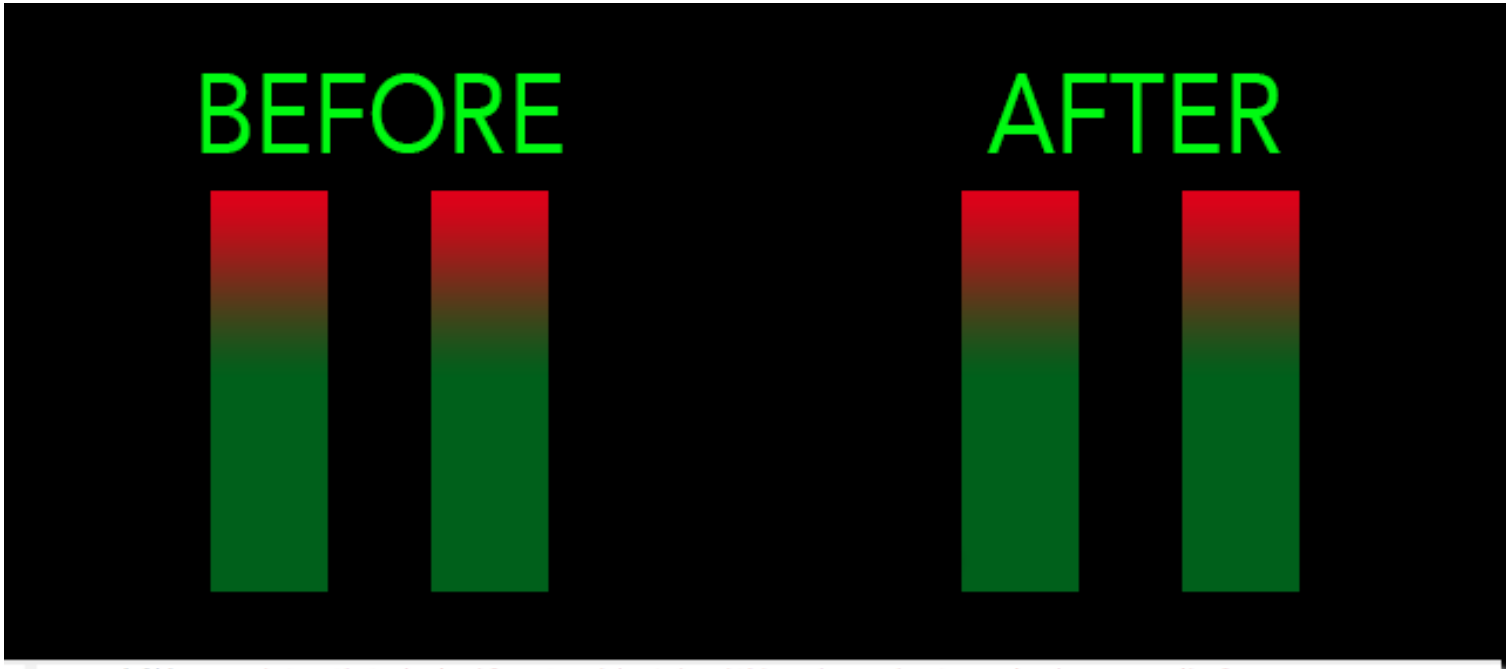


Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

VGA_driver

Bar_tender



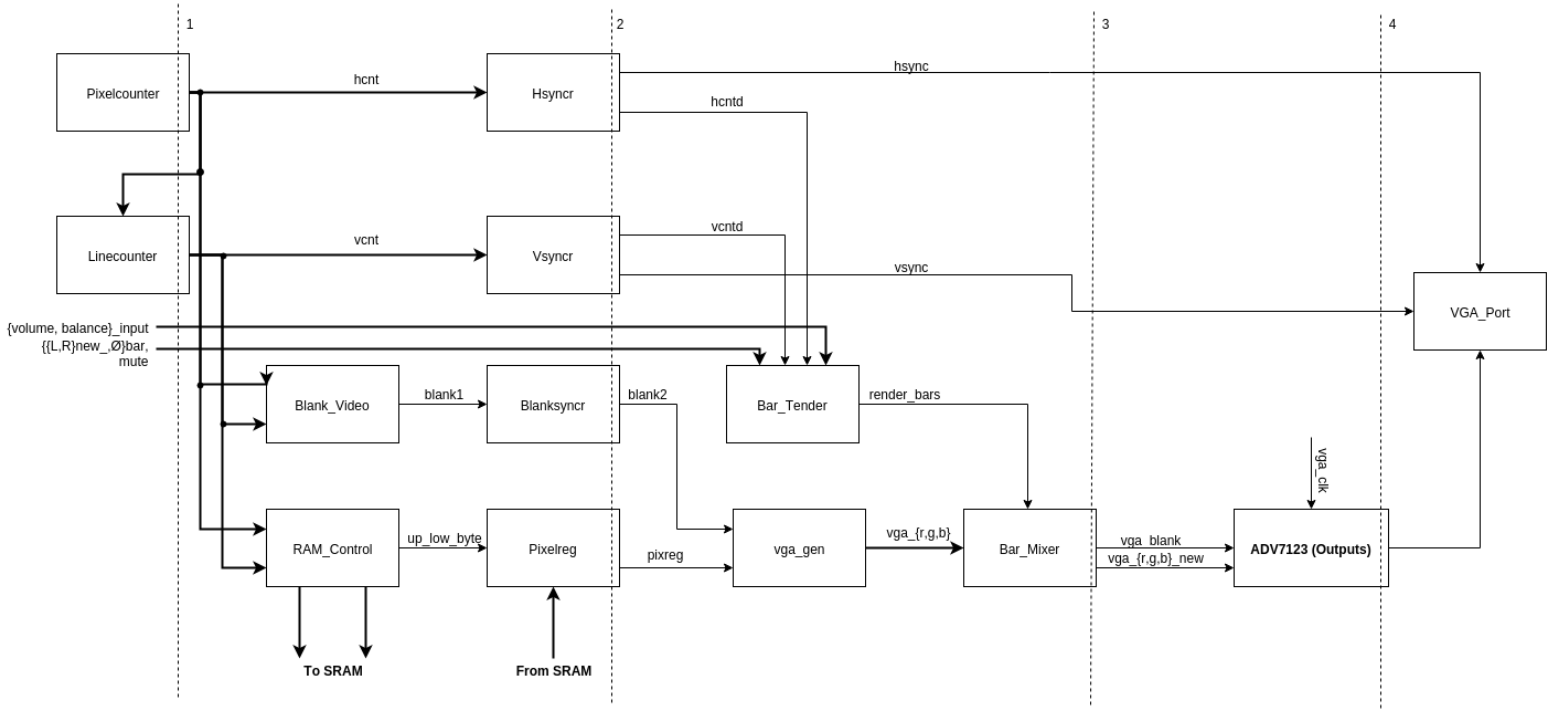

```
process(clk) -- Register that checks if vcnt and hcnt is within a bar and sets render bars accordingly
begin
  if(rising_edge(clk)) then
    render_barsi <= '0';
    render_peaki <= '0';

    -- Left bar
    if((hcnt >= l_bar_x) and (hcnt < (l_bar_x + bar_width)) ) then -- X boundries
      if(vcnt >= l_bar_y and vcnt <= l_bar_y + 171 - L_bar) then    -- Y boundries
        if(vcnt >= l_bar_y + 171 - max_peak_l - peak_thickness and vcnt <= l_bar_y + 17171 - max_peak_l) then -- Peak_level
          render_peaki <= '1';  --Inside peak level indicator
        else
          render_barsi <= '1';  --Inside bar
        end if;
      end if;
    end if;
  end if;
end process;
```

Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

VGA_driver

Vga_driver overview



Group 41

Niklas Blomqvist,
Philip Johansson,
Matteus Laurent,
Johan Levinsson,
Oscar Petersson,
Erik Peyronson

VGA_driver

Bar_mixer

```
1  architecture rtl of bar_mixer is
2  begin
3  process(render_bars, render_peak)
4  begin
5    if(render_bars = '1') then
6      vga_r_new <= (others => '0');
7      vga_b_new <= (others => '0');
8      vga_g_new <= (others => '0');
9    elsif (render_peak = '1') then
10     vga_r_new <= (others => '1');
11     vga_g_new <= (others => '1');
12     vga_b_new <= (others => '1');
13   else
14     vga_r_new <= vga_r;
15     vga_g_new <= vga_g;
16     vga_b_new <= vga_b;
17   end if;
18 end process;
```