

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

# Group 41

## TSIU03 - Final Presentation

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

November 9, 2015

# Introduction

Audio signal processing

Signal level indicator

Our own flavor:

Background image

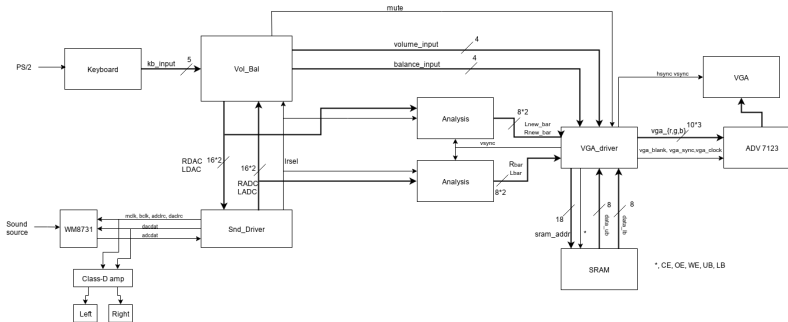
Graphic scales for L/R input/output channels

Class-D amplifier



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

# Overview



Keyboard, Vol\_Bal, Snd\_Driver, Analysis, VGA\_Driver

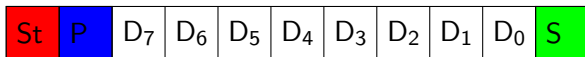
# Keyboard Decoding

## Scan Code Detection



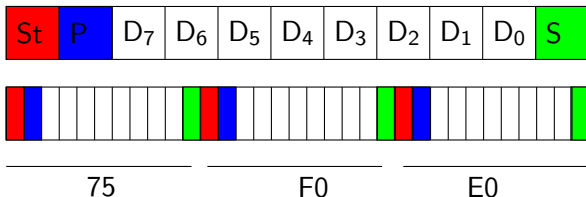
# Keyboard Decoding

## Scan Code Detection



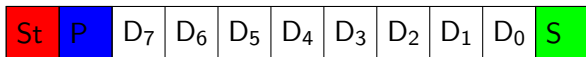
# Keyboard Decoding

## Scan Code Detection



# Keyboard Decoding

## Scan Code Detection



75

F0

E0



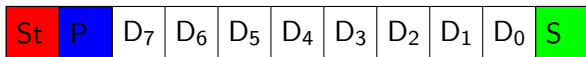
75

F0



# Keyboard Decoding

## Scan Code Detection



75

F0

E0



75

F0



75

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

# Keyboard Decoding

Scan Code Detection



0

75

F0

BREAKSET

# Keyboard Decoding

Scan Code Detection



0

75

F0

BREAKSET



1

75

BREAKSET

# Keyboard Decoding

Scan Code Detection



0

75

F0

BREAKSET



1

75

BREAKSET



0

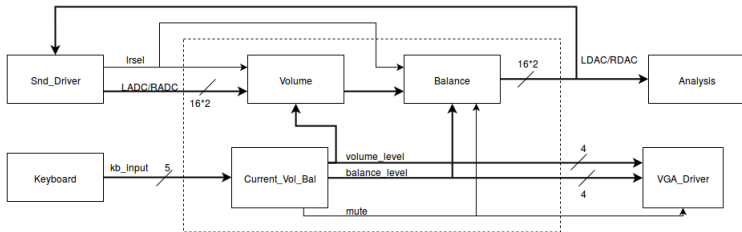
NN

BREAKSET

# Volume and Balance adjustment

Audio signal adjustment

Storage of system levels



# 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)

# 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

# Volume and Balance adjustment

## Balance\_Adjustment

### Linear scaling

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

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

Controlled by volume\_done and lrsel.



# Analysis

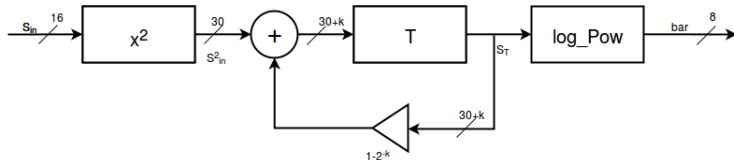
Analyzes incoming samples, low-pass filtering them.

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

Updates in sync with vsync.

Both left & right channel separately.

# Analysis



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

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

# VGA\_driver

## Intro

Based on Laboration 2.

# VGA\_driver

## Intro

Based on Laboration 2.

Two new submodules.

# VGA\_driver

## Intro

Based on Laboration 2.

Two new submodules.

`bar_tender.`

# VGA\_driver

## Intro

Based on Laboration 2.

Two new submodules.

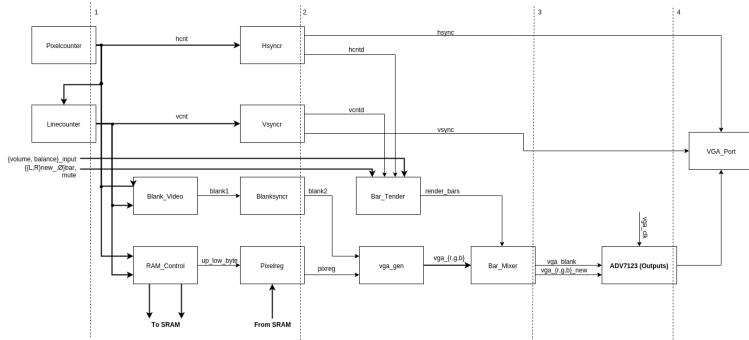
bar\_tender.

bar\_mixer.

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

# VGA\_driver

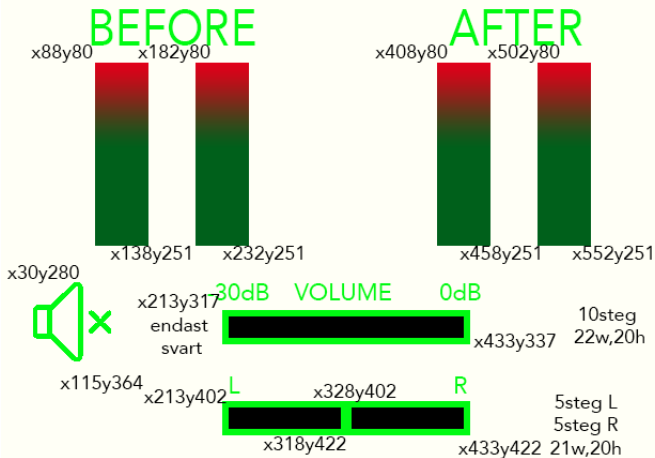
## Vga\_driver overview



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

## VGA\_driver

Concept UI





## VGA\_driver

## Bar\_tender

BEFORE



AFTER



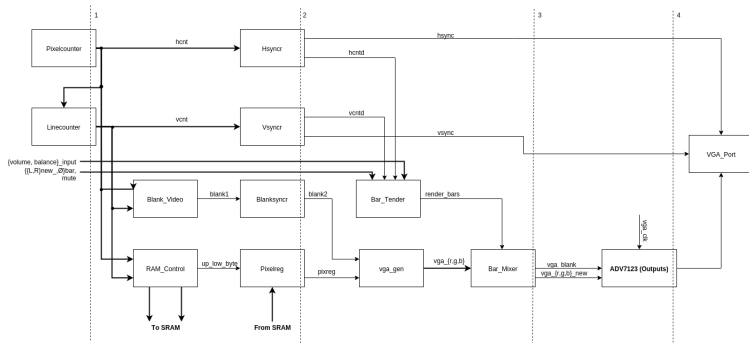
```

1 process(clk) -- Register that checks if vcnt and hcnt is within a bar and sets render bars accordingly
2 begin
3   if(rising_edge(clk)) then
4     render_barst <= '0';
5     render_peakl <= '0';
6
7     -- Left bar
8     if((hcnt >= l_bar_x and hcnt < (l_bar_x + bar_width)) ) then -- X boundaries
9       if(vcnt >= l_bar_y and vcnt <= l_bar_y + 171 - l_bar) then -- Y boundaries
10         if(vcnt >= l_bar_y + 171 - max_peak_l - peak_thickness and vcnt <= l_bar_y + 17171 - max_peak_l) then -- Peak_level
11           render_peakl <= '1'; --Inside peak level indicator
12         else
13           render_barst <= '1'; --Inside bar
14         end if;
15       end if;
16     end if;
17   end if;
18 end process;

```

# VGA\_driver

## Vga\_driver overview



# VGA\_driver

## Bar\_mixer

```
1 architecture rtl of bar_mixer is
2 begin
3   process(renderBars, renderPeak)
4   begin
5     if(renderBars = '1') then
6       vga_r_new <= (others => '0');
7       vga_b_new <= (others => '0');
8       vga_g_new <= (others => '0');
9     elsif (renderPeak = '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;
```