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

Group 41 - Design Specification

Högskoleingenjörsutbildning i datateknik, 180 hp

Design Specification - September 29, 2015
System Design - Project, HT15
TSIU03, Linköpings universitet

Supervisor:
Petter Källström
Department of Electrical Engineering (ISY)

Contents

1. Vad dokumentet ska innehålla:

The entire Design Spec should be 5-7 pages

- Now you are the engineers who receive the requirement specification. What you have to do is to propose a solution that fulfills the requirements of the system.
- Contrary to the requirements specifications, now you have to think about HOW you are going to meet the requirements.
- The design specification must describe the entire system. Think about the main blocks that your system will have, the functionality of each block and the interaction between them, i.e., which information the have to send to each other. Explain the functionality of the blocks and their interaction from a signal processing point of view, i.e., how the audio, video, etc. are processed in each block and which infomation is transmitted between blocks. You can provide som equations to show the algorithms that are applied. Note that this is very different from providing the hardware interfaces between the blocks.
- Later, think about the difficulties that you will find in hardware and the hardware limitations (timing, bandwidth, word length, etc.) and check that your design is viable. Some calculations may be necessary. For instance, if a requirrement says that the system must be able to delay the audio signal one second, you will probably think of using a memory in order to meet the requirement. Then you should make some calculations to check how big the memory must be and if it fits in the FPGA or if you need to use an external memory.
- The design specification must be described from the system level. Please, avoid details that are not relevant at that level. Also, make sure that the person who reads the document can get a clear idea of the entire system.
- As a result, the design specification must be a technical proposal that shows that you have analyzed the problem and found the difficulties that you will face, and provides a first approach to the solution. A good approach for writing the design specification is to present a block diagram of the system, provide a high-level description (at signal processing level) about the functionality of each block and how the blocks interact, and show which requirements present challenges and how you will solve them in hardware.

1.1 Layoutstandarder

Läs README:n i ../Projektrapport/ först och främst. Jag har strukturerat det hela genom att ha varje chapter och section i en fil med motsvarande namn, och nya borde inte behövas. Referera till bilder, avsnitt etc. på adekvat sätt med`\ref` (ger kapitelnummer eller figurnummer) och `\pageref` (ger sidnummer). Hänvisningar inom parentes sätts ej kursivt, i löptext anges de enligt (ta gärna en titt i description.tex):

```
\emph{<Typ> \ref{ref:name}: Name}
```

Jag föreslår även att vi använder oss av `\verb+name+` för att markera namn på moduler, signaler, etc.

Förhoppningsvis har ni vid det här laget bekantat er med min LaTeX-guide, men har ni frågor är det bara att hojta. Använd gärna emacs för att redigera dokumenten då det har stöd för uppmärkning av LaTeX-syntax.

Får jag igång SSH:n till burken därhemma via skolans burkar ska jag försöka hålla autokompileringen vid liv under dagen, men jag garanterar inget.

2. Introduction

Project 41 is based around audio signal processing. The audio input and output both go through the WM8731 chip on a DE2 board. Meanwhile, the hardware settings are controlled from a PS/2 keyboard and displayed on a VGA screen. The hardware settings to be implemented are a volume control and a balance control. In addition, an interface consisting of the input and output power level along with appropriate indicators as stated in the requirement specification.

The WM8731 is a stereo codec, which in Project 41 is used as a bridge between the audio source and a class-D amplifier. The custom hardware controls the WM8731 as the analysis of the input controls and encoding the graphical output. The output sound sent to a Class-D amplifier is then allowed further amplification through another instance of pulse width modulation within the amplifier.

3. System Level Description

System Level Description (Block diagram + description of 1 to 2 pages).

(Make sure that your description justifies how the Requirements of your system are met, especially those which are not obvious.)

This chapter will describe the system main blocks, the functionality of each of them, and the interaction between each block and its adjacent modules. Presented below (Figure 3.1) is a graphical overview of the system and its first layer of modules. A high resolution version is also included in *Appendix A: System Overview*.

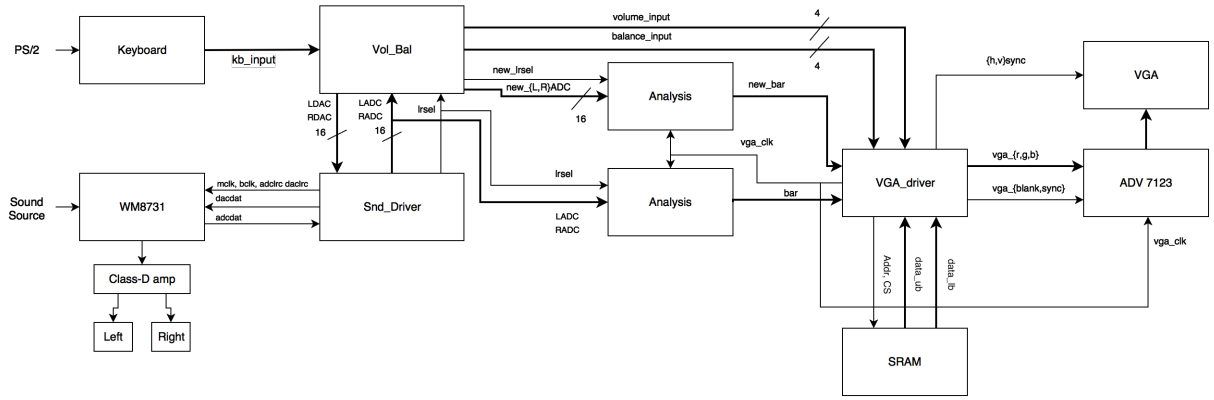


Figure 3.1: A graphical overview of the system's first module layer. Letters inside curly braces indicates multiple signals exlusively including each of the letters.

3.1 Keyboard

The user interacts with the system through a PS/2-connected keyboard. The keyboard is then handled by the module `Keyboard` which reads the scan codes, matches these against a *one hot encoded* preset which makes up the `kb_input` signal passed to `Vol_Bal`.

The module inputs are `PS2_DAT`, `PS2_CLK`, `clk` and `rstn` which are used to shift in the scan code and compare the result with the preset, resulting in `kb_input` — a 5-bit unsigned value indicating if either of the arrow keys have been released. `Vol_Bal` will then use this signal to adjust the volume and balance level. The Up/Down arrow keys controls the volume, and the Left/Right arrow keys controls the stereo channel balance.

The scan codes for the arrow keys consists of two (make code) or three (break code) bytes of information. These codes correspond to each other in the manner listed in figure 3.2.

The scan codes are shifted into a 26-bit shift register which is reset to all ones, and once the start bit (0) is shifted out, the third byte (bit 23...16) is NAND:ed with FF_{16}. A result of "1" then compared to the expected third byte of a released control key which on success sends a `kb_input` to `Vol_Bal`.

Figure 3.2: PS/2 Scan Codes used and corresponding `kb_input`

KEY	MAKE	BREAK	`kb_input`
U ARROW	E0,75	E0,F0,75	00001
L ARROW	E0,6B	E0,F0,6B	00010
D ARROW	E0,72	E0,F0,72	00100
R ARROW	E0,74	E0,F0,74	01000
END	E0,69	E0,F0,69	10000

3.2 Snd_Driver

The `Snd_Driver` module is an audio signal coder/decoder. It translates the signal between a parallel format and a bit serial format. The parallel format is sent to the `Vol_Bal` module which processes the sound and sends it back. The bit serial format is used by the WM8731 chip. `Snd_Driver` consists of two submodules: the `Ctrl_Block` and two instances of the submodule `Channel_Mod`. Depicted below (Figure 3.3), is a graphical representation of `Snd_Driver`.

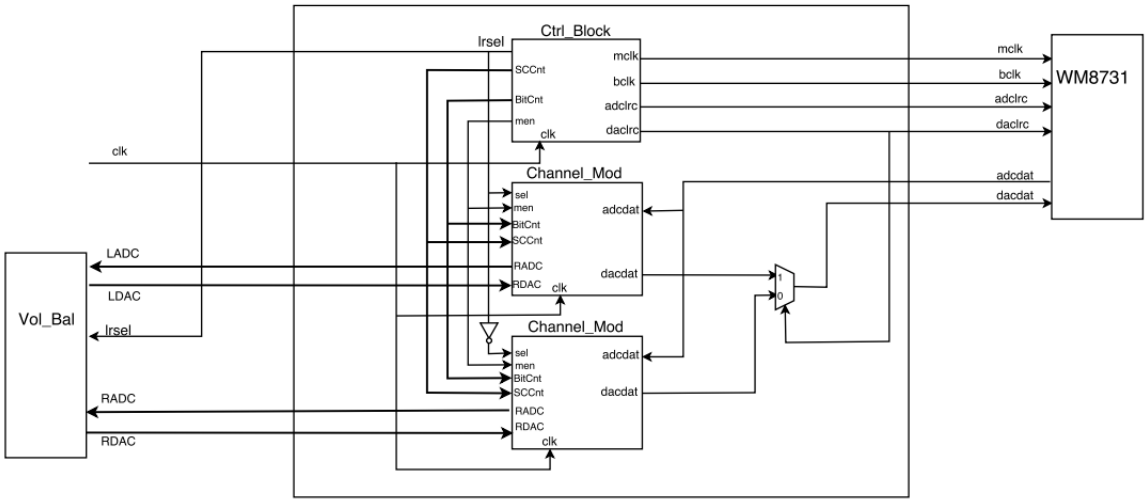


Figure 3.3: Graphical representation of Snd_Driver.

Figure 3.4: List of output signals

Name	Description
`LADC/RADC`	Left/right outgoing samples for the `Vol_Bal` module to process. 16-bits size.
`lrsel`	Select signal for usage of left/right sample.
`mclk/bclk`	`mclk` is WM8731's internal operation clock, bclk is a bit clock
`adclrc`	Left/right selector for `adcdat`. `adclrc`='1' for left.
`daclrc`	Left/right selector for `dacdat`. `daclrc`='1' for left.
`dacdat`	Serial bits to the DAC, one bit per `bclk` pulse.

Figure 3.5: List of input signals

Name	Description
LDAC/RDAC	Left/right incoming samples which shall be shifted out to the WM8731. 16-bits size.
adcdat	Serial bits from the ADC, one bit per bclk pulse.
clk	The system clock. (50 MHz)

3.2.1 Snd_Driver:Channel_Mod

Channel_Mod is a submodule instantiated twice, once for each bidirectional channel. One for the left and one for the right channel. The difference between the two is the sel, or select, signal. One instance receives the sel signal inverted. Channel_Mod gets the sel signal, which if active, indicates that the sound has been processed by Vol_Bal for the selected channel and is ready to be sent back to WM8731. Depending on Sel, Channel_Mod shifts in/out the bits from/to adcdat/dacdat.

3.2.2 Snd_Driver:Ctrl_Block

Ctrl_Block acts as the control block of the module. It is responsible for generating several control signals. The module is built upon a 10-bit counter, cntr. The control signals for the rest of the module is generated from different bits of cntr in the Ctrl_Block, essentially keeping track of what shall be done at which time.

3.3 Vol_Bal

3.3.1 Vol_Bal:current_vol_bal

The Volume/Balance module (Vol_Bal) acts as the hub for processing incoming digital audio signals, forwarded from WM8731 via the SndDriver module. As such, Vol_Bal also keeps internal registers that holds current volume and balance levels as signed 4-bit values (legal values range from -5 to 5 where 0 represents no adjustment). These registers update via the one-hot coded input signal kb_input applied by the Keyboard module, and the values they hold are used as signals (i_volume_lvl, i_balance_lvl) for the internal modules that process the LADC and RADC inputs. Furthermore, the signals are also directed as outputs connected to the VGA_Driver module so that they can be rendered on the screen.

The main function of the Volume/Balance module is to make requested adjustments to incoming values LADC and RADC, which represent a measured amplitude respectively at a certain time. They will first be adjusted for volume by a function $A_{new} = A_{old} * \sqrt{2}^{n}$, where A is the amplitude and n is the signed value in the volume level register. The new values are forwarded for balance adjustment jointly with a *ready* signal to inform that the adj_LADC (or RADC) should be read. Same processing is applied in the Bal_Adj module to produce the LDAC and RDAC outputs conveyed to SndDriver and Analysis.

As a result, the user can digitally adjust volume by -15/+15 dB and also decrease volume by another 15 dB on a single left/right audio channel.

(Lastly, this design will be simplified by combining volume and balance adjustment volumes)

3.4 Analysis

3.5 VGA_driver

The `vga_drive` module exists to handle the rendering of a 640x480 resolution image and the bar-graphs on the vga display. The image beeing rendered consists of a background image previously stored in the SRAM consisting of pre filled bars that within the module will be blanked out according to the input stimuli, which will give the appearance of bars being filled to different levels.

To render an image on the vga screen five main signals is needed. Three analog color channels (red, green and blue) and two signals for synchronization hsync and vsync. The image is rendered pixel by pixel line by line using a horizontal sweep pattern which is reset by the two sync signals. If a color is set when the sweep resets arbitrary patterns can occur and therefore the signal has to be blanked during the reset phase.

The module `vga_drive` has four input signals described in 3.7 and five output signals described in 3.8 and consists of 11 sub modules which can be overviewed in 3.6 and will be described below

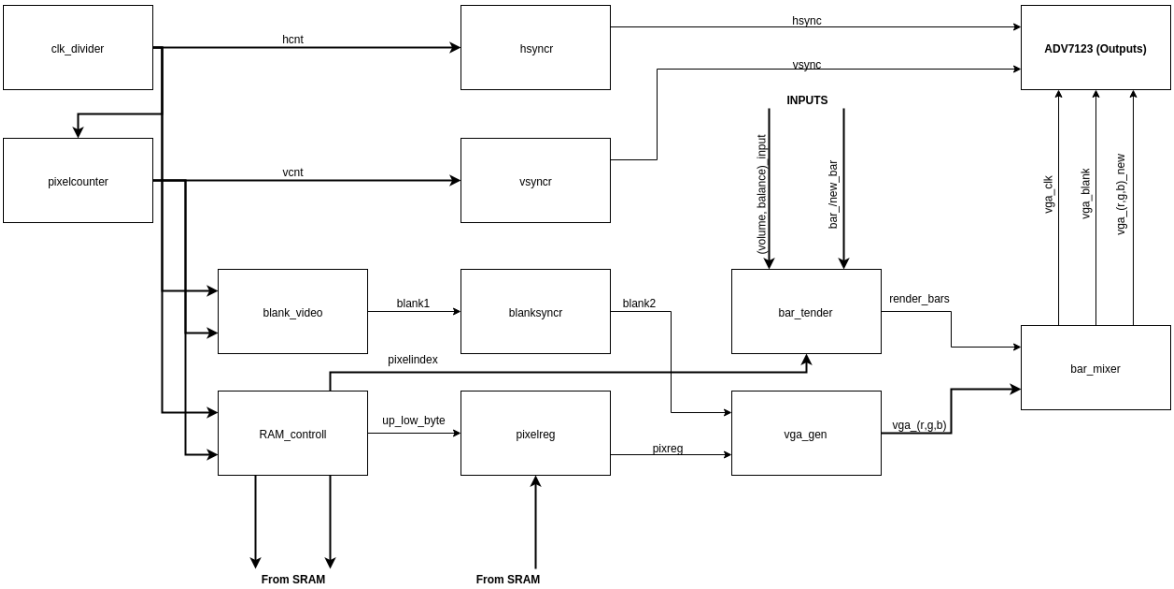


Figure 3.6: Block diagram of vga_drive

Figure 3.7: List of input signals

Input signals	
Name	Description
volume_input	a 4 bit input containing volume information
balance_input	a 4 bit input containing balance information
bar	a n bit input containing signal sound input signal level
new_bar	a n bit input containing manipulated input signal level

3.5.1 VGA_driver:Pipelining

Since there is a time delay for the system to calculate ram adress, generate pixel_index the system is pipelined. The four pipe-line stages is illustrated with dashed lines in figure 3.6. This means that there is a time delay of 3 clock cycles between which pixel is handled and which is drawn.

Figure 3.8: List of output signals

output signals	
Name	Description
vga_clk	clock signal needed for scanning
vga_blank	a blanking signal for blanking when resetting scan
vga_(r,g,b)	three signals containing color information

3.5.2 VGA_driver:pixelcounter

The sub module `pixelcounter` is a pixel generating the internal 10 bit signal hcount. hcount functions as a `pixelcounter` for each row and will count from 0 to 797 which represent the pixels needed for each row in the scanning.

3.5.3 VGA_driver:linecounter

`linecounter` works almost the same as pixelcounter and generates the signal vcount. vcount is incremented with one every time hcount resets and resets after reaching 525. This makes hcount and vcount together act as coordinates to each pixel on the screen during the scanning.

3.5.4 VGA_driver:clock_divider

`Clock\_divider` is just as it's name suggests a clock divider which divides the system clock of 50 MHz to 25 MHz. This is needed because the timings in the vga-interface for the resolution used requires a clock of 25 MHz

3.5.5 VGA_driver:blank_video

Due to the nature of the sweep in the vga-interface the signal needs to be blanked just before, during and after the syncronization signals. This is done using the blank_video sub module, the 1bit blanking signal is active low and should therefore be zero while outside of the visible image and and one inside. This is done by setting blank high when hsync is between 0 and 639 and vsync is between 0 and 479.

3.5.6 VGA_driver:RAM_control

The image used as a background is pre-stored in the SRAM and to acess the image data we need to generate some signals to the SRAM. First is the control signals CE, OE, WE, UB, LB which is set constant to 0, 0, 1, 0, 0. Which basically says that SRAM should be enabled read-only and access both upper and lower byte.

Each 16 bit row in the SRAM contains color information about two pixels, one in the lower and one in the upper byte. Therefore we must generate two signals sram_addr which provides the SRAM with the correct adress and up_lo_byte that provides the sub module pixelreg with the information about which of the bytes should be read . It also generates the output pixelindex which is a counter that counts each pixel in the visible area used by bar_tender (more in section 3.5.11)

3.5.7 VGA_driver:hsyncr, vsyncr

The sub modules `hsyncr` and `vsyncr` is responsible for generating the hsync and vsync signals used for reseting the sweep. In the resolution used in this application this means that hsync should be active(low) during hcount values between 490 and 493 and vsync between vcount values of 655 and 750.

3.5.8 blank_syncr

blank_syncr is simply a d-flip flop needed for pipelining.

3.5.9 VGA_driver:pixel_reg

Pixelregister is responsible for reading the image information from SRAM, the image information is avaliable on the SRAM-bus and the module simply needs to read the correct byte (using up_lo_byte) and put it ine a pipeline register.

3.5.10 VGA_driver:vga_gen

`vga\_gen` will take the pix_reg register and supply it to the three color channel outputs vga_r, vga_g and vga_b. Due to the fact that the AVD7123 chip expects 10 bit values and the stored image containing 3bit color values the values need to be scaled up before provided to `bar\_mixer`

3.5.11 VGA_driver:bartender and barmixer

`bar\_tender` is the submodule responsible for rendering the bar-graphs displaying volume, balance and signal strength before and after signal manipulation. The background image already has the bars drawed filled and to give the apperence of them beeing filled to different levels pixels will be blanked out from the top down. Using the volume, balance, bar, new_bar and pixelindex `bar\_tender` will calculate which pixels should be blanked and set the signal render_bar high.

`bar\_mixer` works as a multiplexer blanking out the bars. The color information is passed through if render_bar signal is low and blanks out the pixel if high which gives the effect of bargraphs beeing filled.

4. Challenges in the Design and Proposed Approach

Challenges in the Design and Proposed Approach (Main 2 or 3 challenges and proposed solutions, 1/2 to 1 page per challenge).

The immediate challenge the project is facing is the `Analysis` module. Since the update rate of the VGA screen is far much lower than the polling rate of the `Snd_Driver`, the analysis have to extract average of a set of samples.

Further, there might be a need for the user interface to be further evened out over a longer period of time for a more fluent visual experience.

5. User Interface

The user interface will be able to display all the manageable settings on a VGA screen. There will in total be four bars. One to indicate the left incomming power, one to indicate the right incoming power, one to indicate the modified left power and one to indicate the modified right power.

The user interface will also display the current volume graduated in dB. The scale goes from -15 dB to +15 dB. This is controlled by the arrow keys, up and down. The balance indicator appears at the bottom of the interface. The balance indicator works like 0 is equal to the same amout of power from both left and right, if you press the right arrow at the keyboard, the balance indicator will step up the right side of the "0".

There will also be a mute figure to show if the mute button is activated.

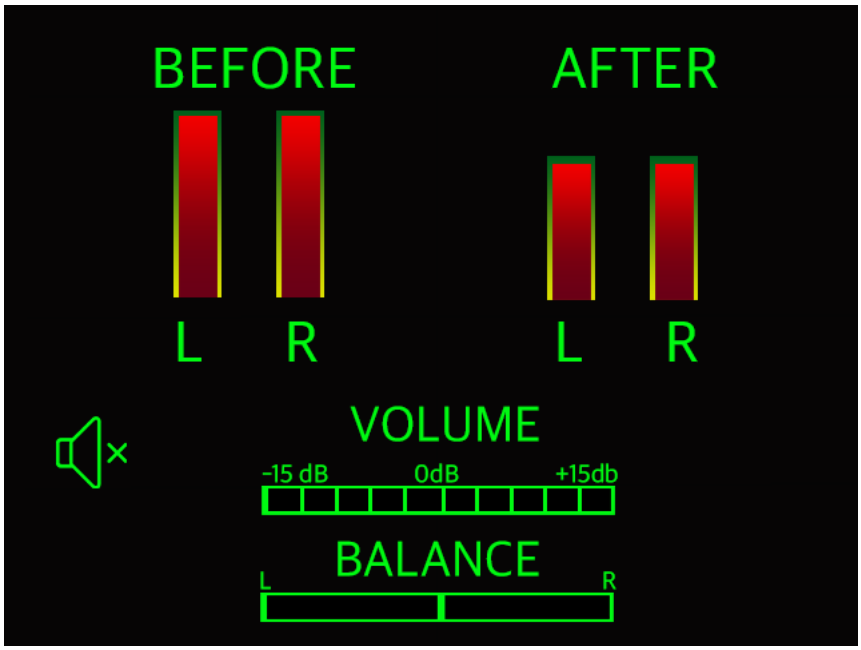
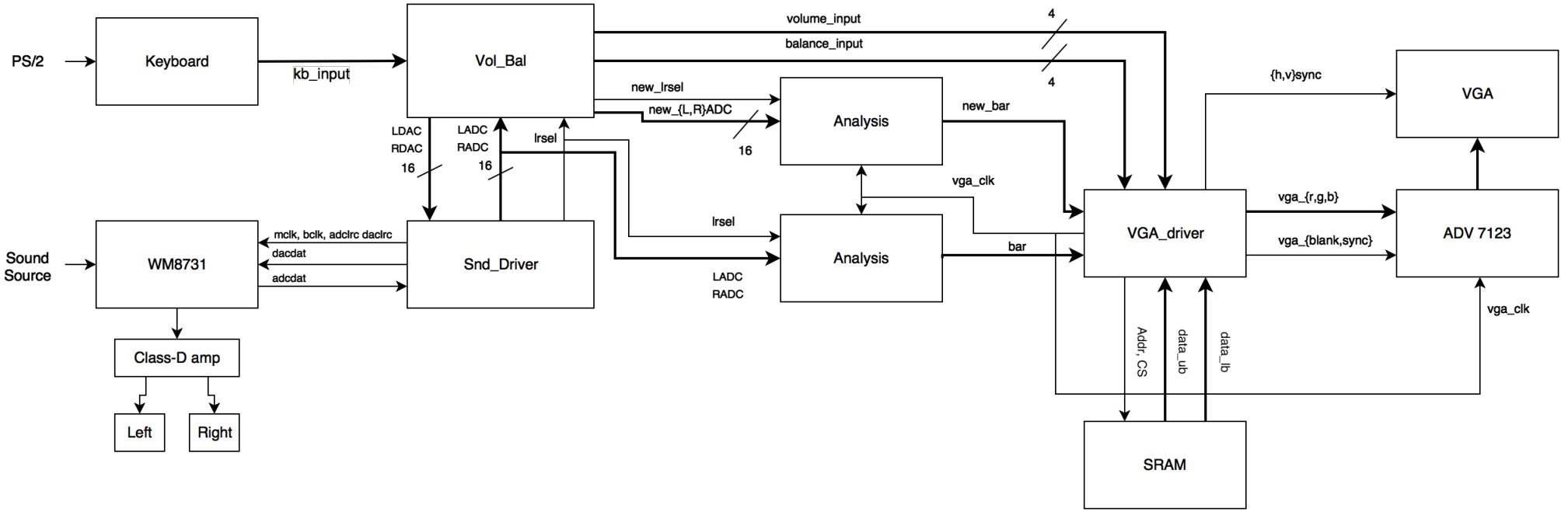


Figure 5.1: User interface

A. Appendix: System Overview


PS/2
Keyboard
kb_input
Vol_Bal
volume_input
balance_input
4
4
new_lrsel
new_{L,R}ADC
16
Analysis
new_bar
{h,v}sync
VGA
LDAC
RDAC
16
LADC
RADC
16
lrsel
vga_clk
vga_{r,g,b}
vga_{blank,sync}
ADV 7123
Sound Source
WM8731
mclk, bclk, adclrc daclrc
dacdat
adcdat
Snd_Driver
lrsel
LADC
RADC
Analysis
bar
VGA_driver
vga_clk
Class-D amp
Left
Right
Addr, CS
data_ub
data_lb
SRAM