Pong Game

Assignment date: 04.07.2016
Submission date: 24.07.2016
Group members: Sebastian Wittka, Felix Kaiser and Habib Gahbiche.

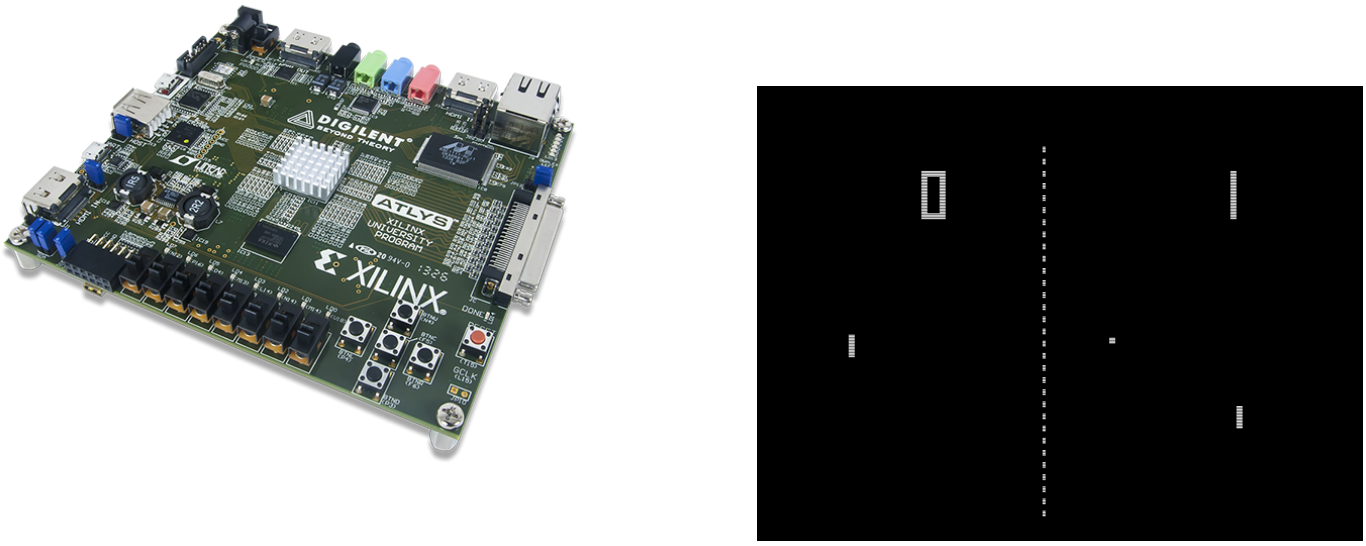
Contents

1 Introduction

This project is about implementing the game Pong on the Atlys Spartan-6 FPGA board. Pong is a two dimensional multiplayer game that simulates table-tennis. Each of the two players controls an in game paddle by moving it vertically in order to hit a ball back and forth. A player scores a point when the opponent fails to return the ball.

We also took advantage of the built-in HDMI port and the AC-97 Codec to produce a better image and audio quality output.

Figure 1 shows a picture of the used board, and a screenshot of the (yet to be) realized game.



(a) Atlys Spartan-6 board

(b) Screenshot of the game Pong

Figure 1: Used board and screenshot of the game

1.1 Block Diagrams

Figure 2 shows the block diagram of the module `image_generator_c`. Figure 3 shows a block diagram of the sound generator.

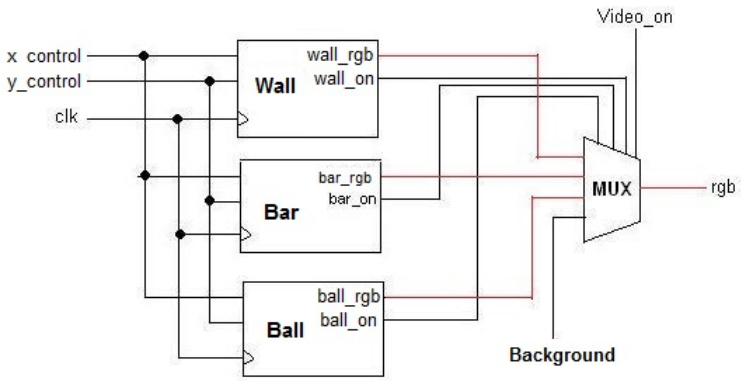


Figure 2: Block diagram of the Image Generator

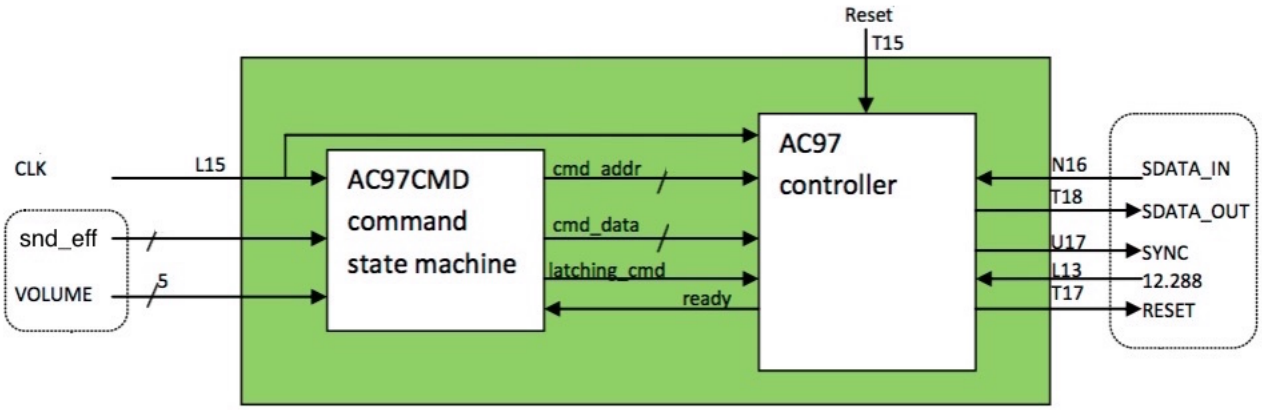


Figure 3: Block diagram of the Sound Generator

2 Implementation

The hardware description of the implemented game pong is divided into six modules. Firstly the video controller, which implements the VGA respectively HDMI interface, provides the functionality to display the video output on a monitor. Two modules generate the inputs for the movement of the paddles. The right paddle is controlled by the debounced up and down buttons of the board and the right paddle is controlled by the computer opponent. Additionally the image generator creates all required objects like the ball and paddles as well as their movement. Furthermore the match controller manages the interactions between the objects and the state of the match itself. Lastly the audio output is created by the sound generator module.

2.1 Video Controller

The video controller provides an interface for the image generator so objects can easily be displayed on a monitor. Thereby it makes the coordinates of the current pixel available and hides all additional requirements of the protocol like timing. The point of origin of the pixel matrix is located at the top left corner of the monitor. Because our group has access to the Atlys Spartan-6 board with HDMI connector as well as the Nexys 4 board with VGA connector, we decided to implement a video controller for both protocols, so the remaining modules could be implemented using both boards. The main focus was laid on the above mentioned uniform interface for the image generator so the remaining development is independent of the used video controller.

2.1.1 VGA Controller

2.1.2 HDMI Controller

For the first prototypes of the project VGA was used to stream the images to a screen. That needed to be changed because of two reasons: Firstly digital visual interfaces can get provide better resolution than analogue ones and secondly, and that is the actual main reason, the atlys board does not have a VGA port.

2.1.3 HDMI - Principle of Operation

HDMI(High Definition Multimedia Interface) is a digital interface which transport visual as well as audio data differential. The used encoding is the same as DVI called Transition-minimized differential signaling. TMDS is a kind of 8b/10b encoding but not the actual 8b/10b used by PCIE. It is optimized to create as less as possible transitions. Fast transitions are the main cause for electromagnetic interference which couples the twisted pair connection.
The TMDS encoder transforms the actual bits with the previous bits with a XOR or a XNOR function. It takes always the one which causes less transitions. This transformation happens for the 8 data bits while the ninth bit encodes which operation was used. Additionally the sustained average DC level should stay similar. Therefore the encoder can invert the data to even out the balance of ones and zeros. This is saved in the tenth bit.
Because of the fact that a HDMI connector just have 19 Pins, the data is transferred sequentially at a higher clockspeed. The first 12 Pins are for the data and the clock. For each color(RGB) and the clock there is a differential pair with one shield connection. That means that the bits have to be transferred with a 10 times faster clock than the clock which generates the pixel.

2.1.4 HDMI with Spartan6 FPGAs on Atlys Boards

The Atlys Board which is used in the project got 4 HDMI ports. The one which is called J2 is a usual Type A HDMI Connector. With this information it is possible to get the right constraints out of the following table(4).

The actual design for the HDMI interface is shown in 5. This is a reference design by Xilinx itself. In the presented design the audio HDMI interface was not used, so ignore this part. Because of that there are just the video signals and the pixel clock there to mention. At first the RGB data is encoded by the TMDS module like it was described in the sections before. For a well synchronized transfer we need at the same time a perfectly timed pixel clock at a speed of 74.25 MHz. Because 74.25 MHz is hard to reach out of 100 MHz, 75 MHz were taken and it still worked fine. This clock was needed to be created by the clockwizard IP. In the next parts the TMDS encoded RGB data is converted and serialized into the differential TMDS signals. Because of the serialization we need a 10 times faster clock which is created out of the PLL shown in the schematic. Additionally to RGB data the clock is sent as well for synchronization.

2.2 Debouncer

2.3 Computer Opponent

2.4 Image Generator

The Image Generator takes inputs from the players and outputs the video that can be displayed through the HDMI interface of the Atlys board. The panels shown in figure **??** are submodules of the module `image_generator_c`. This module calculates the movement of the ball and movement the two panels that are controlled by the players.

The movement of the ball is done by the `ball_c` module. After a well determined time frame, the ball's movement direction is determined and the next `x_pos` and `y_pos` are either incremented or decremented.

The module `panel_c` determines the y-coordinate of on panel based on the player input. For instance, pressing `btn_up` increments the `y_pos` signal if the panel did not reach the top edge already.

HDMI Type A Connectors				HDMI Type D	
Pin/Signal	J1: IN	J2: Out	J3: IN	Pin/Signal	JA: BiDi
1: D2+	B12	B8	J16	1: HPD	JP3*
2: D2_S	GND	GND	GND	2: RES	VCCB2
3: D2-	A12	A8	J18	3: D2+	N5
4: D1+	B11	C7	L17	4: D2_S	GND
5: D1_S	GND	GND	GND	5: D2-	P6
6: D1-	A11	A7	L18	6: D1+	T4
7: D0+	G9	D8	K17	7: D1_S	GND
8: D0_S	GND	GND	GND	8: D1-	V4
9: D0-	F9	C8	K18	9: D0+	R3
10: Clk+	D11	B6	H17	10: D0_S	GND
11: Clk_S	GND	GND	GND	11: D0-	T3
12: Clk-	C11	A6	H18	12: Clk+	T9
13: CEC	NC	0K to Gnd	NC	13: Clk_S	GND
14: RES	NC	NC	NC	14: Clk-	V9
15: SCL	C13	D9	M16	15: CEC	VCCB2
16: SDA	A13	C9	M18	16: Gnd	GND
17: Gnd	GND	GND	GND	17: SCL	C13**
18: 5V	JP4*	5V	JP8*	18: SCA	A13**
19: HPD	1K to 5V	NC	1K to 5V	19: 5V	JP3

*jumper can disconnect Vdd **shared with J1 I2C signals via jumper JP2

EDK designs can use the xps_tft IP core (and its associated driver) to access the HDMI ports. The xps_tft core reads

Figure 4: Constraints for HDMI(out of the Atlys User Manual)

2.5 Match Controller

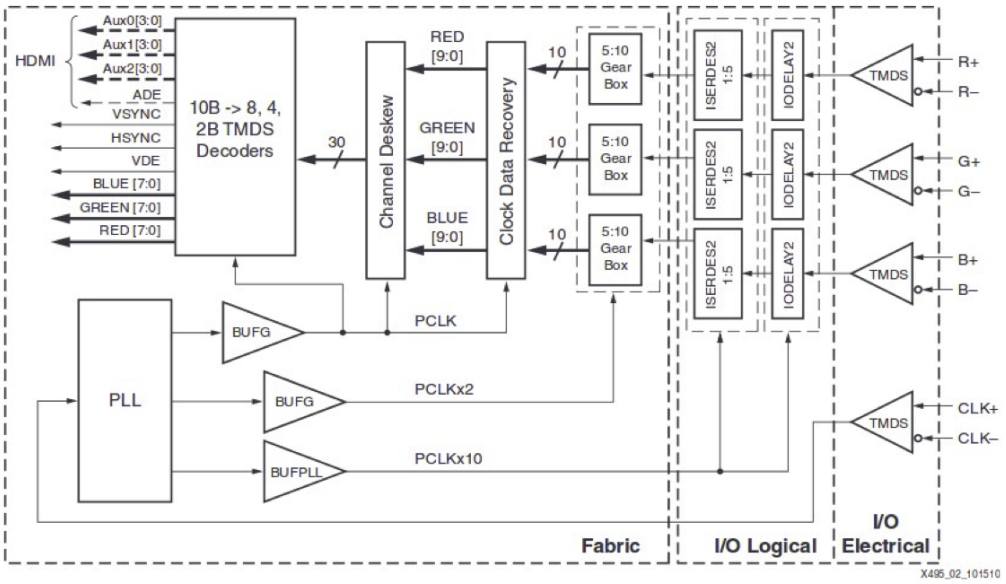


Figure 2: **TMDS Receiver Design**

Figure 5: Design how to build a HDMI interface(out of the TMDS User Interface Manual for Spartan6)

2.6 Sound Generator

In order to generate sound, we used the on board LM4550 chip. Figure 3 shows the block diagram of the sound generator. This module takes in the sound events from the Match Controller module (i.e when sound generation should take place), reads a sound effect from a ROM and sends it to the LM4550 chip that in turn outputs the sound effect to a connected speaker.

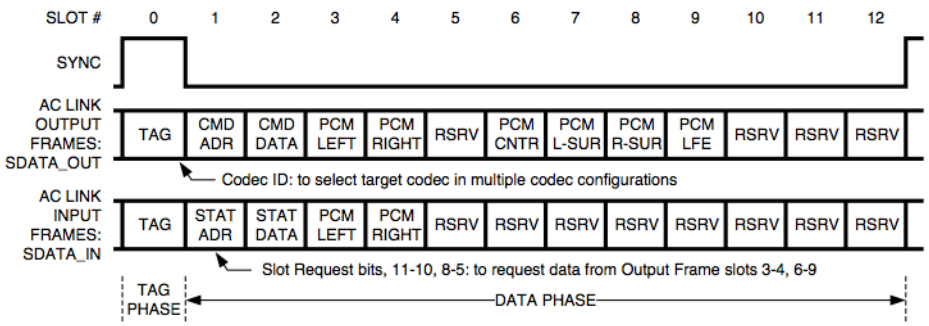
The inputs to the `snd_gen_c` include a `clk`, an active low reset, a serial data in line `sdata_in`, a 12.288 MHz bit clock from the AC97 chip, 3 bit `snd_eff` signal and 5 bit `volume` control (will be connected to the switches of the Atlys board).

The module's output include a `sync` signal, serial data output `sdata_out` and an AC97 active low reset that initializes the AC97 (LM4550) chip.

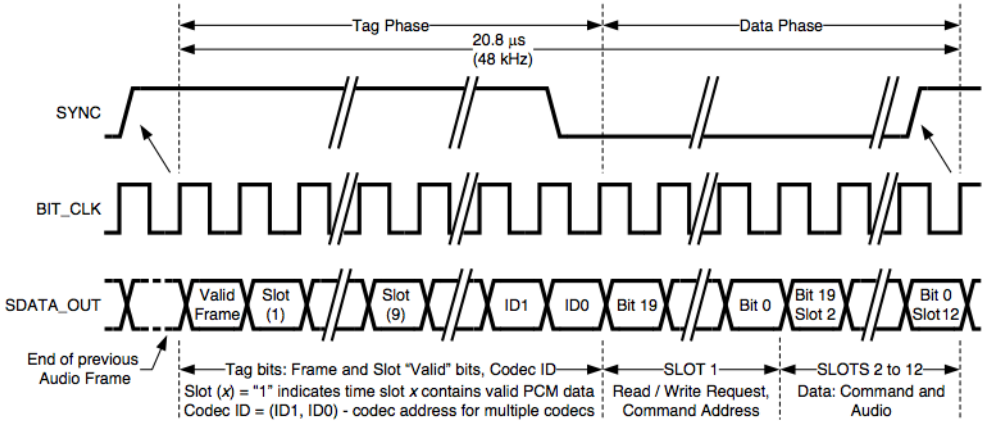
Internally, the `snd_gen_c` module contains an AC97 controller and an AC97CMD submodules.

- **AC97 controller:** implements the AC Link serial interface protocol. Figure 6a shows an AC bidirectional audio frame, whereas figure 6b shows an AC output audio frame. In this project, we will be using the LM4550 chip for output only, however, the input audio frame (not shown here) has also been implemented for testing reasons. The next paragraph is a brief description of the AC link interface protocol. For more details about the AC97 link serial interface protocol, see `http://www.ti.com/lit/ds/symlink/lm4550.pdf`

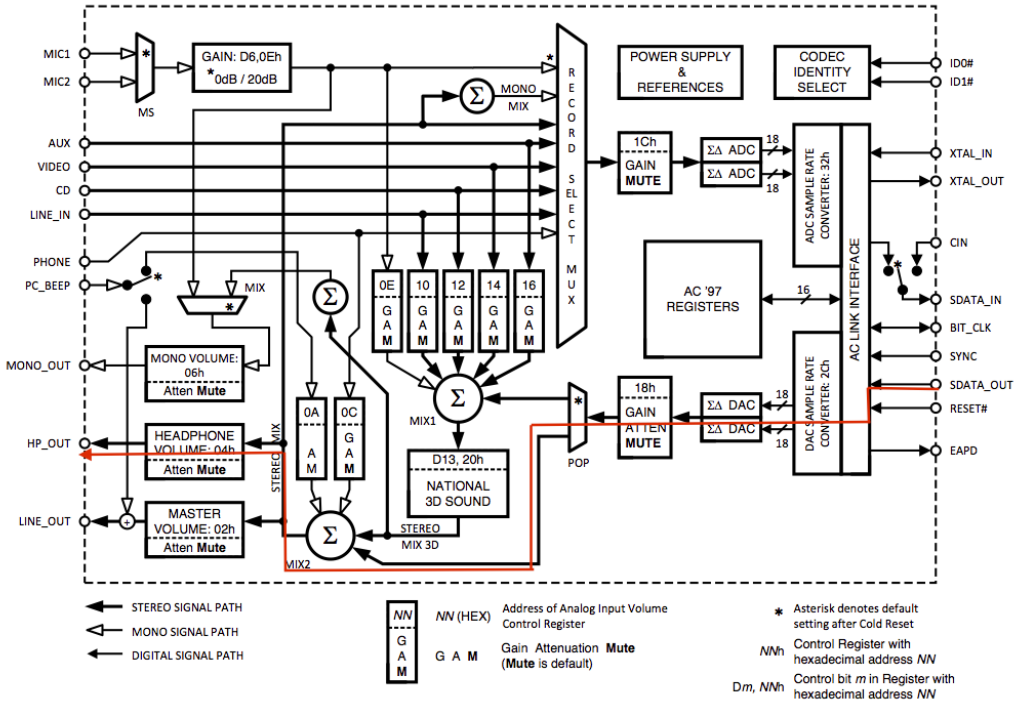
  The AC Link Output Frame carries control and PCM data to the LM4550 control registers and stereo DAC. Output Frames are carried on the `sdata_out` signal which is an output from the AC97 Controller and an input to the LM4550 codec. As shown in Figure 6b, Output Frames are constructed from thirteen time slots: one Tag Slot followed by twelve Data Slots. Each Frame consists of 256 bits with each of the twelve Data Slots containing 20 bits. Input and Output Frames are aligned to the same SYNC transition.



(a) AC link bidirectional frame



(b) AC link output audio frame



(c) Block diagram of LM4550 chip. Red line is the path from FPGA output to `HP_OUT`. All registers along this path need to be configured.

Figure 6: AC link serial interface protocol

- **AC97CMD command state machine:** is a state machine that creates the control data based on the inputs `snd_eff` and `volume` and passes it to the AC97 Controller. This control data controls among others the volume of the output audio and the DAC sample rate. Figure 6c shows the path of the audio data that gets transferred within the LM4550 chip and the registers that need to be configured along this path. The following code was extracted from the file `a97cmd_fsm.vhd` that implements the state machine necessary to configure the registers of the LM4550 chip. It shows an example of how to activate the output `HP_OUT`. As shown in figure 6c `LINE_OUT` is muted by default.

  ...

```
process (next_state, cur_state, atten)
    begin
        case cur_state is
            when S0 =>
            cmd <= X"02_8000";  -- master volume     0 0000->0dB atten,
                                                    1 1111->46.5dB atten
                next\_state <= S2;
            when S1 =>
                                                    -- HP_OUT volume
                cmd <= X"04" & "000" & atten & "000" & atten;
                next\_state <= S4;
            when S2 =>
```

  ...

  The AC97CMD module also reads the audio sound effects from a ROM that gets initialized when the FPGA is programmed. The right sound effect is chosen according to the signal `snd_eff`.

3 Assessment

4 Summary

5 references

- The sound generator module is based on the sample project provided by Xilinx. http://www.xilinx.com/support/university/boards-portfolio/xup-boards/AtlysBoard.html#docs
- LM4550 http://www.ti.com/lit/ds/symlink/lm4550.pdf
- HDMI reference https://reference.digilentinc.com/_media/atlys:atlys:atlys_rm.pdf