Pong Game

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

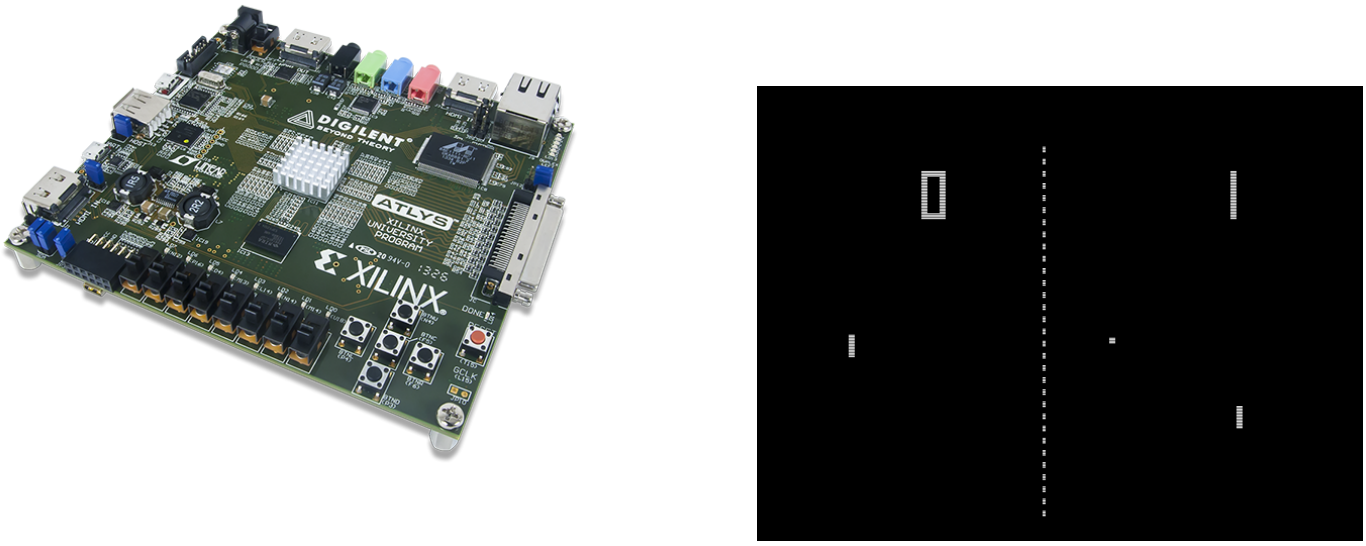
Contents

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.

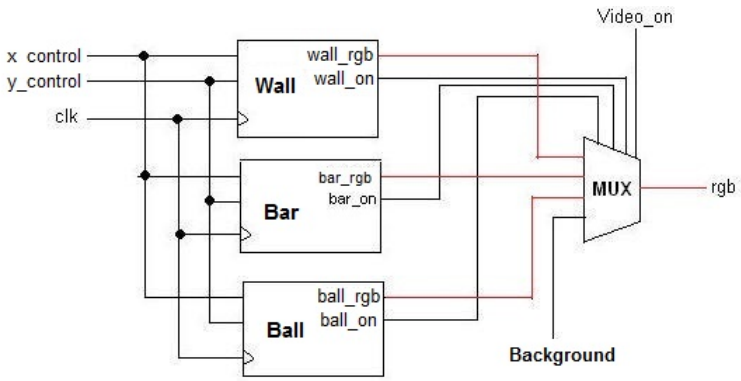


Figure 2: Block diagram of the Image Generator

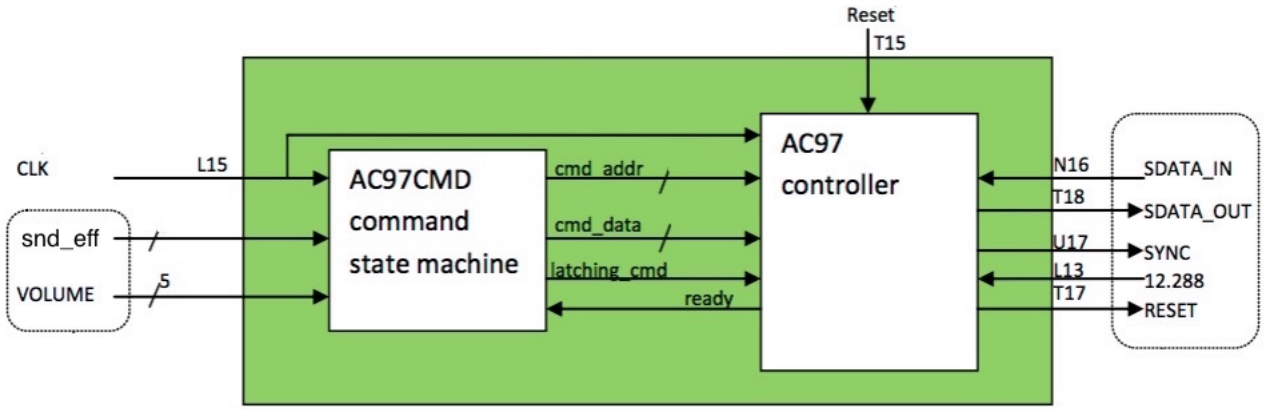


Figure 3: Block diagram of the Sound Generator

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

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.

2.5 Match Controller

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.

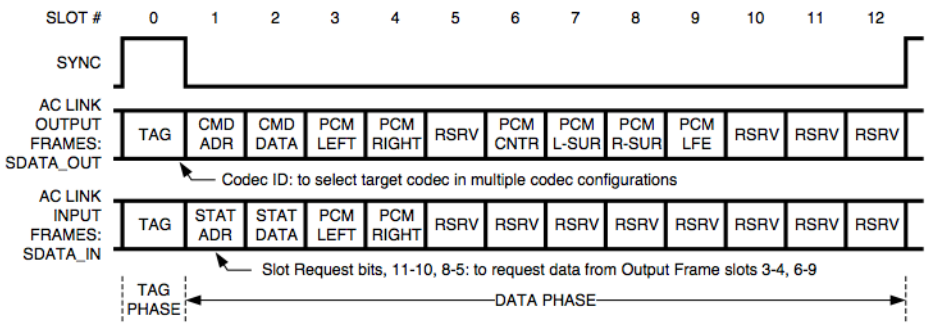
- **AC97 controller:** implements the AC Link serial interface protocol. Figure 4a shows an AC bidirectional audio frame, whereas figure 4b shows an AC output audio frame. In this project, we will be using the LM4550 chip for output only, however, the input audio frame (not shown here) has also been 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 4b, 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.

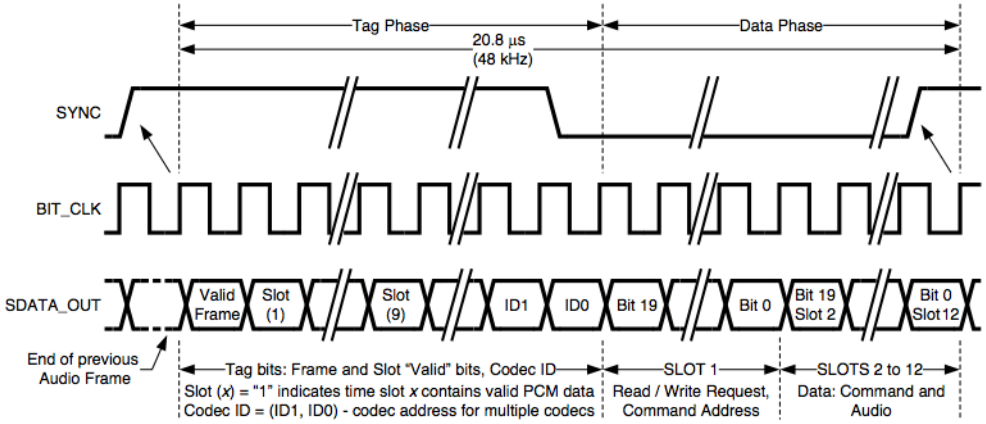
- **AC97CMD command state machine:** is a state machine that creates the control data based on the inputs `snd_eff` and `volume` and passes it to the AC97 Controller. This control data controls among others the volume of the output audio and the DAC sample rate. Figure 4c shows the path of the audio 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 4c `LINE_OUT` is muted by default.

  ...

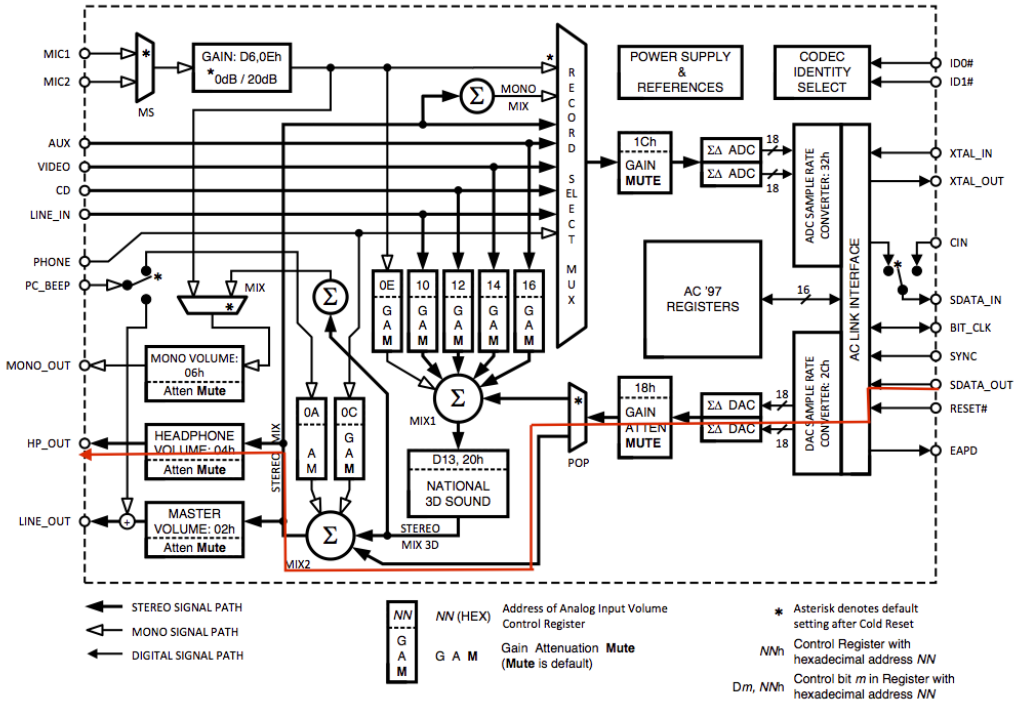
```
process (next_state, cur_state, atten)
    begin
        case cur_state is
            when S0 =>
            cmd <= X"02_8000";  -- master volume     0 0000->0dB atten,
                                                    1 1111->46.5dB atten
                next\_state <= S2;
            when S1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



(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 4: AC link serial interface protocol

...

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 VHDL implementation of the AC97 controller was based on the sample porject provided by Xilinx. See `http://www.xilinx.com/support/university/boards-portfolio/xup-boards/AtlysBoard.html#docs`
- LM4550 manual `http://www.ti.com/lit/ds/symlink/lm4550.pdf`