Pong Game

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

Contents

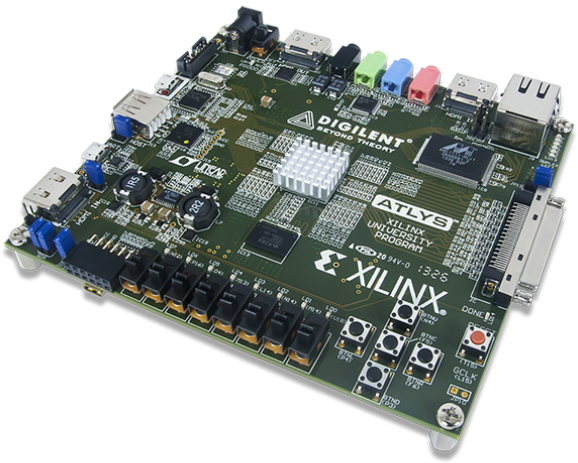
1 Topic

1.1 Brief Task Description

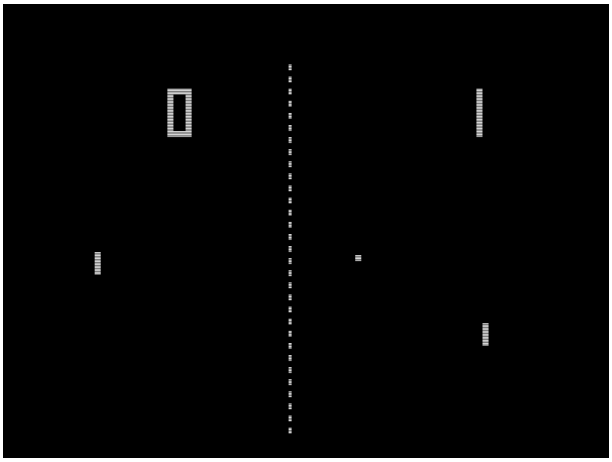
This project is about implementing the game Pong on the Atlys Spartan-6 FPGA board. Pong is a two dimensional multiplayer game that simulates table-tennis. Each of the two players controls an in game paddle by moving it vertically in order to hit a ball back and forth. A player scores a point when the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

1.2 Block Diagrams

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

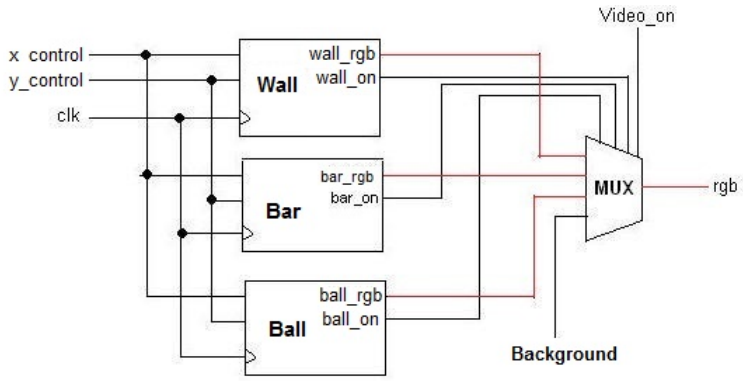


Figure 2: Block diagram of the Image Generator

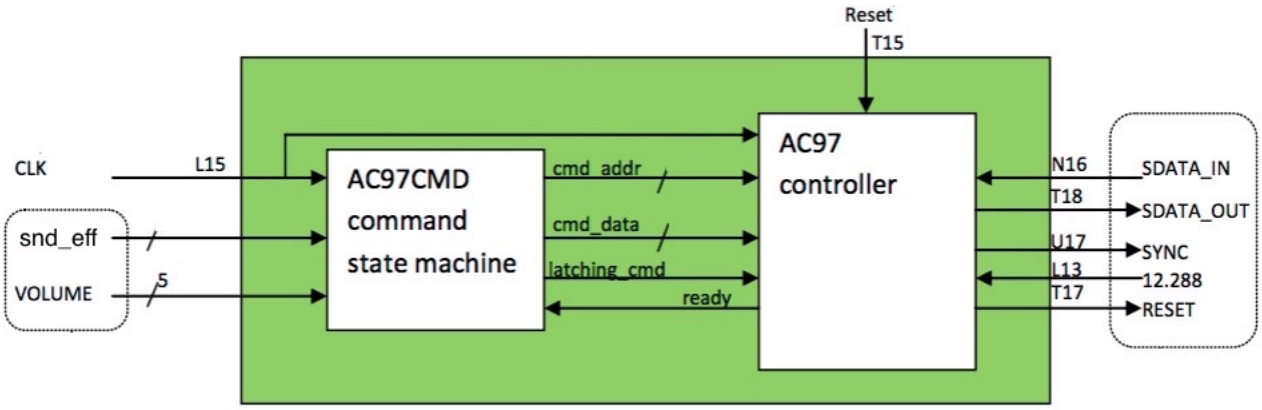


Figure 3: Block diagram of the Sound Generator

1.3 Functional Details

1.3.1 Image Generator

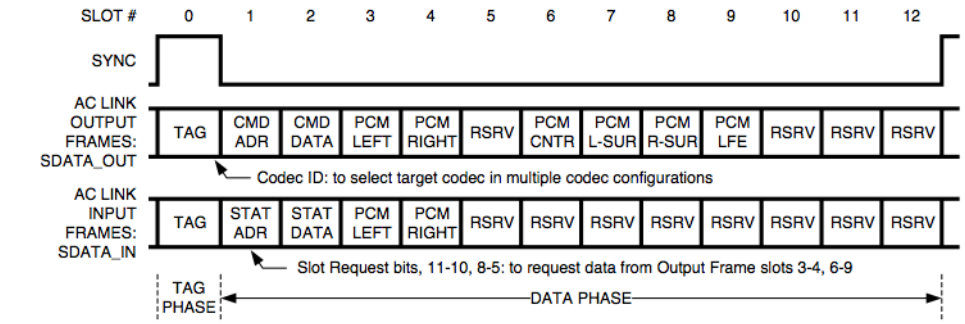
The Image Generator takes inputs from the players and outputs the video that can be displayed through the HDMI interface of the Atlys board. The panels shown in figure **??** are submodules of the module `image_generator_c`. This module calculates the movement of the ball and movement the two panels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

The movement of the ball is done by the `ball_c` module. (see next Section for more details on implementation). 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.

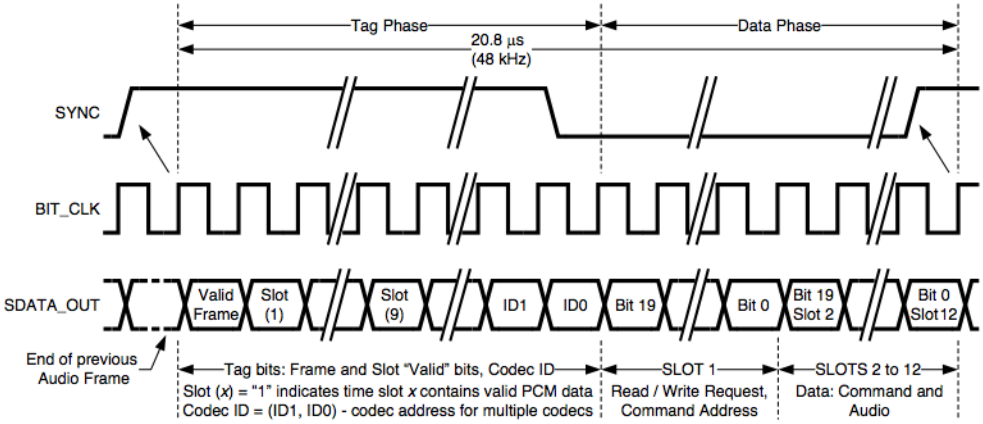
The module `panel_c` determines the y-coordinate of on panel based on the player input. For instance, pressing `btn_up` increments the `y_pos` signal if the panel did not reach the top edge already.

1.3.2 Sound Generator

In order to generate sound, we used the on board LM4550 chip. Figure 3 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



(a) AC link bidirectional frame



(b) AC link output audio frame

Figure 4: AC link serial interface protocol

to the LM4550 chip that in turn sends the sound effect to a connected speaker.

The input 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 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 configures the internal registers of the AC97 chip (i.e audio volume, gain etc...)

2 Implementation

2.1 Video Controller

2.1.1 VGA Controller

2.1.2 HDMI Controller

2.2 Image Generator

2.3 Match Controller

2.4 Sound Generator

3 Implementation

3.1 Modules

3.2 Results

3.2.1 Synthesis and Implementation results

3.3 Problems

4 Assessment

5 Summary

6 Attachment