⚠ You have previously submitted this assignment with David Rama Jimeno. Group can only change between different assignments.

This course has already ended.

« 10. Exercise 7 (/comp.ce.240/spring-2024... 11. Exercise 8 » (/comp.ce.240/spring-2024...

COMP.CE.240 (/comp.ce.240/spring-2024/) / 10. Exercise 7 (/comp.ce.240/spring-2024/E07/) / 10.1 Exercise 07: Audio codec controller (start early)

Assignment description (/comp.ce.240/spring-2024/E07/audio_ctrl/)

My submissions (1/1000)

Exercise 07: Audio codec controller (start early)

In this exercise you will learn

- how to read data sheets
- to generate signals with specific frequencies
- parallel-serial transformation

Your task is to create a controller for the DA7212 Audio codec. It is located in the ARD-AUDIO-DA7212 Audio Shield on top of the PYNQ board and accessible via GPIO pins.

- The codec datasheet (https://www.renesas.com/eu/en/document/dst/da7212-datasheet)
- User guide of the shield (https://www.renesas.com/eu/en/document/mat/ard-audio-da7212-audio-shield-user-guide)

In this synthesizer figure (https://plus.tuni.fi/graderA/static/compce240-f2021/project_work_fixed.png) the codec is the blue box. The audio controller that you create in this exercise can be seen in the lower right corner of the synthesizer block.

- Block interface:
  - Entity name: `audio_ctrl`
  - Integer type generics with default values:
    - `ref_clk_freq_g, 12 288 000` (Hz)
    - `sample_rate_g, 48 000` (Hz)
    - `data_width_g, 16` (bits)
  - Inputs:
    - `clk`
    - `rst_n`
    - `left_data_in` (width `data_width_g` bits)

        - `right_data_in` (width `data_width_g` bits)
    - Outputs:
        - `aud_bclk_out` (width 1 bit) (=Bit clock)
        - `aud_data_out` (width 1 bit) (=Data output)
        - `aud_lrclk_out` (width 1 bit) (=Left-right clock)
- Define yourself the needed constants, signals and processes
    - Described functionality:
        - Your controller takes **a snapshot** from `left_data_in` and `right_data_in` concurrently to a register at the beginning of each sample cycle
        - Then the controller feeds the content of the register to `aud_data_out` output according to the specification of Dialog's codec
        - For this purpose, the controller generates two oscillating signals (clock signals to DA7212 codec): `aud_bclk_out` and `aud_lrclk_out`
            - These kinds of signals oscillating slower than main clock can be implemented with counters which are clocked with the main clock `clk`
            - When a counter reaches its maximum value, it inverts the output signal, thereby making it to change between zero and one with preferred frequency
            - Thus, the maximum value of the counter defines half of the clock period length of the generated clock signal
            - Calculate the clock period lengths and the maximum values of the counters from the given frequencies
            - **Important:** The audio controller also has to work with other frequencies than 12.288 MHz, so make your code generic
            - Take care that `bclk` and `lr_clk` are synchronized together even if the sampling frequency or the clock frequency changes (the block is not tested using the default values).
        - The controller is synthesized to the FPGA and we want to make a synchronous, one-clock system so remember that **you cannot use `bclk`-signal to clock the flip-flops; i.e., make all the processes sensitive only to the main clock signal (`clk`)**

- See Dialog Semiconductors DA7212 codec specification (https://www.renesas.com/eu/en/document/dst/da7212-datasheet)
    - In audio codec block diagram on page 8, you can see three signals which your controller should drive:
        - Digital audio interface (DAI) data in (DATIN)
        - Digital audio interface (DAI) word clock (WCLK) (also known as left-right clock (LRCLK))
        - Bit clock (BCLK)
    - Note that left-right clock and bit clock can be either inputs or outputs (depends on if the codec is in master or slave mode). Our codec will be in slave mode so we will use them now in input mode (outputs of our own block).

    - Get familiar with the circuit's digital audio interface. Relevant information can be found on page 55 and few pages onward (especially "left justified slave mode" is important to understand). Timing is shown in Figure 30.
    - You will be using the left-justified mode as shown in the figure. There will be 16 bits going on each line, both left and right.
- The desired data feed mode is chosen using separate configuration interface with I2C in exercise 11

- Hints
    - Before writing the code, it is useful to make a preliminary test bench (see Exercise 8 (https://plus.tuni.fi/comp.ce.240/spring-2024/E08/tb_audio_ctrl/)). As very first, make a structural description of the test bench as well as clock and reset generation. This way, you can get right away to examine what your controller roughly does.
    - Whenever possible, it is practical to use equality and inequality operators (=, /=) instead of the other comparison operators (>, <, >=, <=) to save in the area of the hardware implementation

---

Return:

Obtain the files specified below and put them under `E07` folder in your Git repository. Remember to do a git push. Return the files also by file return in the bottom of this page.

- Files to return:
    - `answer07.txt`
- Write a small plan of your block in the return which contains at least:
    - Bit clock frequency
    - Limit values of the clock divider counters and how they were derived
    - A brief plain language summary of the controller's process/processes and what they do
- It is enough to answer just the questions now, as the code will be returned in Exercise 8 (https://plus.tuni.fi/comp.ce.240/spring-2024/E08/tb_audio_ctrl/)

answer07.txt

Choose File No file chosen

Submit with David Rama Jimeno Submit

Earned points

1 / 1

Exercise info

Assignment category
VHDL exercises

Your submissions
1 / 1000

Points required to pass
1

Deadline
Sunday, 10 March 2024, 23:59

Late submission deadline
Friday, 31 May 2024, 23:59

Group size
1-2

Total number of submitters
62

« 10. Exercise 7 (/comp.ce.240/spring-2024... 11. Exercise 8 » (/comp.ce.240/spring-2024...