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

« 11. Exercise 8 (/comp.ce.240/spring-2024... 12. Exercise 9 » (/comp.ce.240/spring-2024...

COMP.CE.240 (/comp.ce.240/spring-2024/) / 11. Exercise 8 (/comp.ce.240/spring-2024/E08/) / 11.1 Exercise 08: VHDL Test bench for audio codec controller (start early)

Assignment description (/comp.ce.240/spring-2024/E08/tb_audio_ctrl/) My submissions (3/1000)

This course has already ended.

Exercise 08: VHDL Test bench for audio codec controller (start early)

This exercise teaches the usage of finite state machines.

The task is to:

- create a model implementation of the DA7212 audio codec (`audio_codec_model.vhd`)
- make a test bench (`tb_audio_ctrl.vhd`) and
- verify the correct functionality of the audio controller made in the previous exercise (https://plus.tuni.fi/comp.ce.240/spring-2024/E07/audio_ctrl/).

You will also be using your wave generator block that you created in exercise 6 (https://plus.tuni.fi/comp.ce.240/spring-2024/E06/wave_gen/) for the test bench input generation.

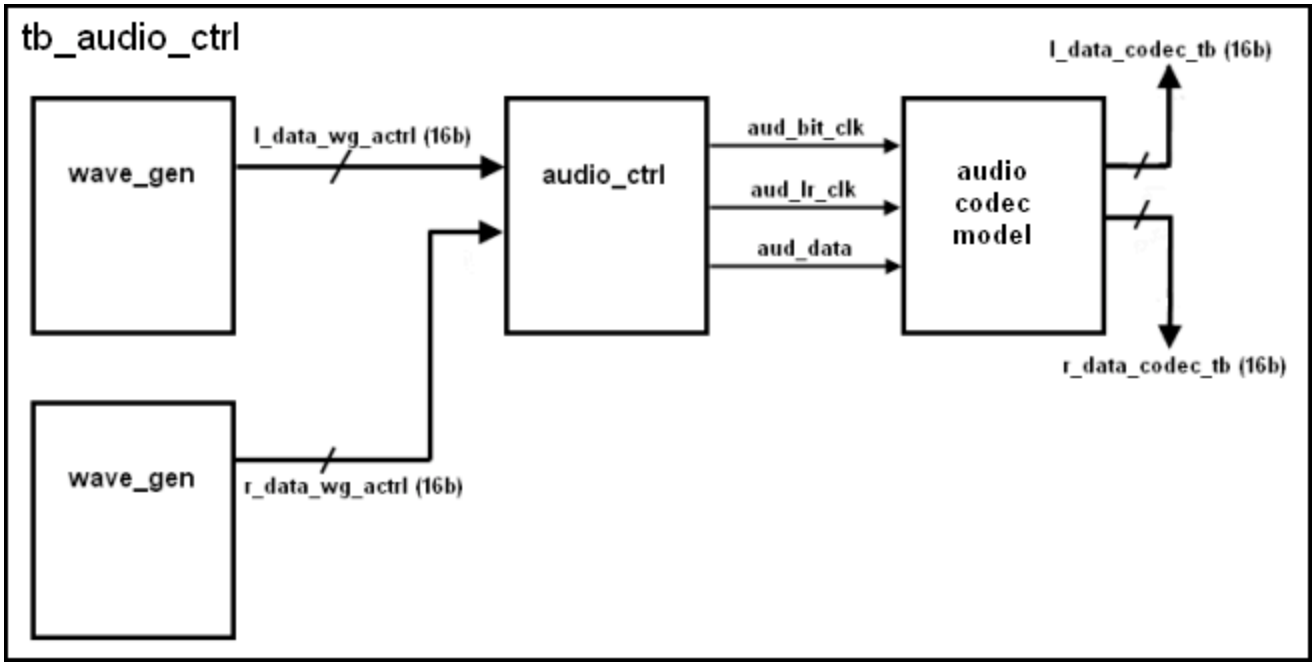
Test bench

Entity name: `tb_audio_ctrl`

- The test bench will be a top level block which connects two previously designed waveform generators (with different parameters) to the audio controller's left and right inputs
  - Hint: At first, you can drive a more simple "waveform" to the controller, e.g., plain zero to the left channel and plain one to the right
  - For instance, by creating signals in the test bench using `after`-keyword, it is possible to see easier what should be the output of the codec

- It is useful to make at least primitive mutation testing (https://en.wikipedia.org/wiki/Mutation_testing) for the test bench.
  - So, when the tested block (DUV) works at least partially, break it on purpose and see if the test bench notices it.
  - You can for instance comment out some assignment clause, negate a condition of an if-clause, change the value in comparison +/- 1 etc.
  - Fix the test bench so that it notices the errors.

The test block diagram will look like the following:



Inside the test bench you need to:

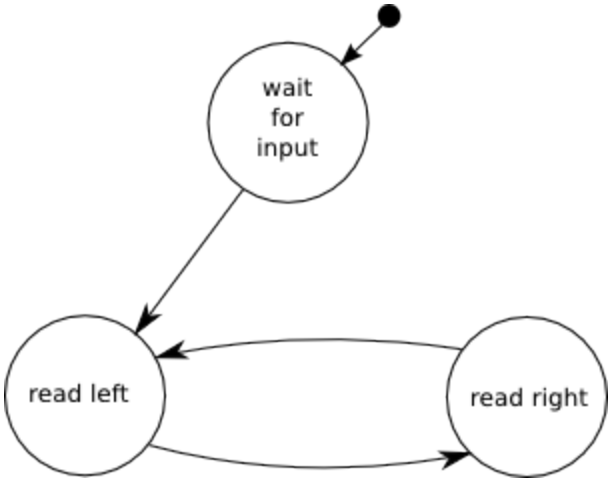
- Connect the blocks according the figure (2 wave_gens, audio_ctrl and audio_codec_model)
- Generate clock and reset signals
- **Generate sync_clear so that wave generators are reset at least once!** Do this before 10 ms in the simulation.
- Suitable (generic parameter) values are:
  - For the clock: 12.288 MHz frequency (but you can use 50 ns cycle for clock generation in simulation -> 20 Mhz frequency)
  - step-values two and ten
  - 48 kHz sampling frequency
- **Attention!** Audio controller also has to work with other frequencies than 12.288 MHz. You must modify the `ref_clk_freq_g` generic of the `audio_ctrl` component to do this. It is not enough to change the clock period alone!

Audio codec model

Since we can't obviously use the real hardware audio codec board in our simulation, we need to create a model implementation for the codec. It doesn't need to be synthesizable.

- Model interface:

    - Entity name: `audio_codec_model`
    - Generic integer type parameter:
        - `data_width_g, 16` (bits)
    - Inputs:
        - `rst_n`
        - `aud_data_in`
        - `aud_bclk_in`
        - `aud_lrclk_in`
    - Outputs:
        - Last fed left channel value `value_left_out` (width `data_width_g`)
        - Last fed right channel value `value_right_out` (width `data_width_g`)
- The state machine described below is required to be used in the model
    - Lecture slides help to implement the state machine
        - Three states:
            - `wait_for_input`
            - `read_left`
            - `read_right`
        - Move into the state `read_left` with the same condition from both the states `wait_for_input` and `read_right`.
            - Note that changing the channel requires caution:
              Remember that state transition takes 1 clock cycle and try not to lose the bit incoming during the transition.
            - If you use edge detection (e.g., for `lr_clk`), make it with logic, not with 'event-structure.
        - Define a new signal type which defines the previously mentioned states
        - Create a signal with this type for storing the registers of the state machine



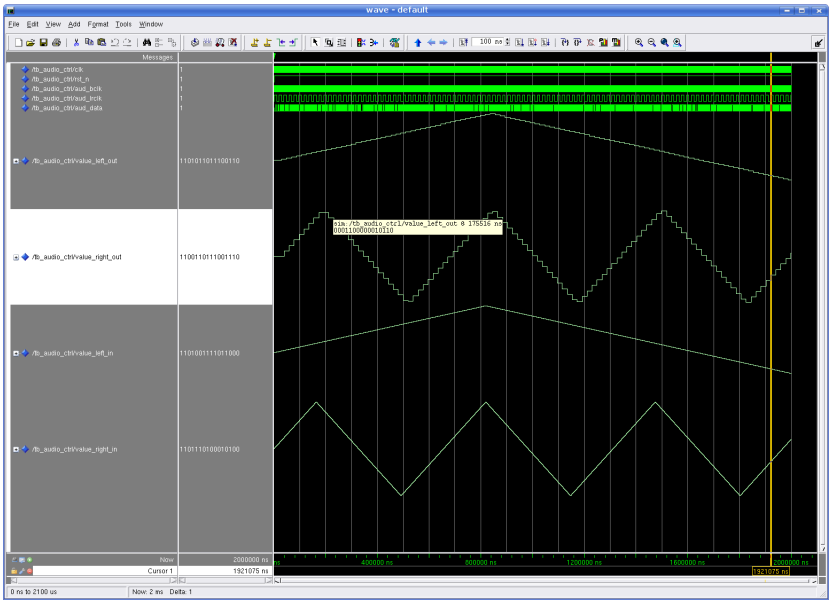
- The model shall save the inputs to an internal register and update the value to output as soon as possible
    - Hint: Be careful when LR-clock changes. Remember that state transition takes 1 clock cycle.

- As this audio codec model is only used for simulation and NOT synthesized to FPGA, **you can use aud_bclk_in as the clock for a synchronous process to create registers.**

Verification

Verify the functionality of the audio controller using the described test bench and monitor the generated waveforms

- The waveform produced by the `wave_gen`-block should be quantized (looks like stairs like below) because of the sample rate



(https://plus.tuni.fi/graderA/static/compce240-f2021/E08/reference_waveforms.png)
  - The upper waves are quantized and the lower are the original
  - The lower frequency step value is 2 and the other is 10
- Things to pay attention to in the waveform:
  - Does the synchronic reset function correctly?
  - Are the bit clock, LR-clock and audio data synchronized as in the audio codec data sheet's left-justified mode? **Errors in this are the leading reason for boomerangs!**
  - Is there the correct amount of audio data bits during one half-period of the LR-clock?
- Check the audio codec configuration from the previous exercise (https://plus.tuni.fi/comp.ce.240/spring-2024/E07/audio_ctrl/)
- Prefer equality evaluation (=) over other comparisons (>, <, >=, <=)

To earn a BONUS (https://plus.tuni.fi/comp.ce.240/spring-2024/bonus/self_checking_tb/) point, the test bench has to be modified to automatically recognize the correct functionality from wrong

---

Return:

- Put your returned files under `E08` folder in your Git repository
  - Return files:
    - `audio_codec_model.vhd`

    - `audio_ctrl.vhd`
    - `tb_audio_ctrl.vhd`
    - A screen capture of a wave form output from your audio codec model when run with your controller at 20 MHz clock. The wave forms should show several waves and look at least superficially correct before submission!
- Check that the files' header comments are valid, made according to instructions, and you have followed the coding rules
- Push the changes to your repository and submit (with your partner if you have a group!)
    - Use the **ssh variant** of the repository url in the submission. Otherwise the tests will fail 100% even with working design.
    - The url looks like *git@course-gitlab.tuni.fi:compce240-spring2024/<your_group_number>.git*.

Enter your Git repository address for grading

Did you remember git add - git commit - git push?

Submit with David Rama Jimeno

Submit

Earned points

10 / 10

Exercise info

Assignment category
VHDL exercises

Your submissions
3 / 1000

Points required to pass
10

Deadline
Sunday, 17 March 2024, 23:59

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

Group size

1-2

Total number of submitters

62

« 11. Exercise 8 (/comp.ce.240/spring-2024… 12. Exercise 9 » (/comp.ce.240/spring-2024…