Exercise 15 Verifying band_scale.v via Randoms *(HW4 Problem 4)*

- Back in HW2 (and Ex06) you produced the **band_scale.v** for your project. You did the best you could validating it, but it was a difficult block to validate completely and you might have bugs.

- In this problem I am providing stimulus, and expected response for band_scale in two separate files that you can download from the website. **band_scale_stim.hex** & **band_scale_resp.hex**.

  - For **band_scale_stim.hex** the vector is 28-bits wide and is assigned as follows:

Stimulus Bit Range:	Signal Assignment:
Stim[27:16]	POT[11:0]
Stim[15:0]	audio[15:0]

  - For **band_scale_resp.hex** the 16-bit vector simply represents the expected response on **scaled[15:0].**

Exercise 15 Verifying band_scale.v via Randoms *(HW4 Problem 4)*

• There are 1000 vectors of stimulus and response that were generated from random vectors run on a known good implementation. Read each file into a separate memory using **$readmemh**.

• Produce a self checking testbench (**band_scale_rnd_tb.v**) that reads in **band_scale_stim.hex** and applies the stimulus to the appropriate signals

• Loop through the 1000 vectors and apply the stimulus vectors to the inputs as specified. Then wait till #1 time unit and compare the DUT output to the response vector (self check). Do all 1000 vectors match?

• Submit to the dropbox:
  - Your testbench **band_scale_rnd_tb.v**
  - Proof of some form that you ran the test bench and it passed.

(Some Hints Given on Next Slide)

Exercise 15 band_scale_rnd_tb.v Hints:

- Instantiate DUT (band_scale.v) connecting all the inputs to the respective bits of a 28-bit wide vector of type **reg**.
- Declare a “memory” of type **reg** that is 28-bits wide and has 1000 entries. This is your stimulus memory
- Declare a “memory” of type reg that is 16-bits wide and has 1000 entries. This is your expected response memory
- Inside the main “initial” block of your testbench do a **$readmemh** of the provided .hex files into the respective “memories”
- In a **for** loop (located later in the **initial** block) loop over 1000 entries assign an entry of the stim memory to the stim vector that drives the DUT inputs
- Wait for wait for #1 time unit. Now check, does the DUT output (***scaled***) match the respective bits of the response vector?