

Exercise 15 Verifying desiredDrive.sv via Randoms

- Back in HW2 (and Ex06) you produced the **desiredDrive.sv** for the 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 desiredDrive in two separate files that you can download from the website. **desiredDrive_stim.hex** & **desiredDrive_resp.hex**.
 - For **desiredDrive_stim.hex** the vector is 33-bits wide and is assigned as follows:

Stimulus Bit Range:	Signal Assignment:
stim[32:21]	avg_torque[11:0]
stim[20:16]	cadence_vec[4:0]
stim[15:2]	incline[13:0]
stim[1:0]	setting[1:0]

- For **desiredDrive_resp.hex** the 12-bit vector simply represents the expected response on **target_curr[11:0]**.

Exercise 15 Verifying desiredDrive.sv via Randoms

- 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 (**desiredDrive_rnd_tb.v**) that reads in **desiredDrive_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 #1 time unit and compare the DUT output (**target_curr**) to the corresponding response vector in **desiredDrive_resp.hex** (self check). Do all 1000 vectors match?
- Submit to the dropbox:
 - Your testbench **desiredDrive_rnd_tb.v**
 - Proof of some form that you ran the test bench and it passed.

(Some Hints Given on Next Slide)

Exercise 15 desiredDrive_rnd_tb.v Hints:

- Instantiate DUT (desiredDrive.sv) connecting all the inputs to the respective bits of a 32-bit wide vector of type **reg**.
- Declare a “memory” of type **reg** that is 33-bits wide and has 1000 entries. This is your stimulus memory
- Declare a “memory” of type **reg** that is 12-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 (**target_curr**) match the respective bits of the response vector?