Script to generate testcases

AGENDA

- ❑. Overview
- ❑. Background
- ❑. Input file
- ❑. Python script
  - ❑Functions
- ❑. Running the script
- ❑. DEMO

OVERVIEW

- Script for test generation helps reduce time in creating multiple testcases
- Doesn’t require too much attention to the register specifics of design
- Makes the process of testcase generation less mechanical
- Python was used as it provides a lot of packages to make it programmer friendly

BACKGROUND

- The test case is divided into sections that need to be/ can be modified for a particular testcase. These sections are stored in separate text files.
- The input parameters of the testcase are given through a csv file which has the following format.
- The first column has the parameter names.
- The following columns represent the values of the parameters for each train.
- Each of the subsequent columns represent a train.
- The script is written in python 3.7 and uses a package “pandas” which is required to be installed to run either on Windows or Linux
- Note: This does not use regular expressions but just string operations for text manipulation

Name	Date modified	Type	Size
__pycache__	10/23/2018 11:33 ...	File folder	
inputs.csv	10/23/2018 12:07 ...	Microsoft Excel C...	1 KB
stdig_test_gen.py	10/24/2018 2:59 PM	Python File	20 KB
arb.txt	10/23/2018 12:07 ...	TXT File	5 KB
e_off_trim.txt	10/23/2018 12:07 ...	TXT File	1 KB
rot_map.txt	10/23/2018 12:07 ...	TXT File	1 KB
sti_map.txt	10/23/2018 12:07 ...	TXT File	1 KB
template.txt	10/23/2018 12:07 ...	TXT File	6 KB
trvn.txt	10/23/2018 12:07 ...	TXT File	3 KB

INPUT FILE

- It is recommended to use Microsoft Excel or Libre Calc for ease in providing inputs
- The parameters here corresponds to the most commonly used parameters that are provided in the stim_dig spec document.
- More parameters can be added as per your requirement.
- However the function that enables assignment in the testcase needs to be added in the script
- The particular parameters can explained during the demo

INPUT FILE

	A	B
1		tr_1
2	e0_sti	a0
3	e1_sti	
4	e2_sti	a7
5	e3_sti	
6	e4_ret	a1
7	e5_ret	
8	e6_ret	a3
9	e7_ret	
10	e0_rot	a1
11	e1_rot	a7
12	e2_rot	b3
13	e3_rot	a4
14	matching	1
15	pulse_type	3
16	rot_num	3
17	rot_ca	0
18	csw_gnd	0
19	csw_ret	1
20	on_dur	$urandom
21	off_dur	$urandom
22	ramp_dur	1
23	tslice_dur	1
24	secg_ext_dur	1
25	glbl_ext_dur	1
26	ramp_stp	3
27	ru_en	1
28	rd_en	1
29	glbl_ext_en	1
30	secg_ext_en	1
31	ap_bl	0
32	frc_bl_en	0
33	nf_bl	1
34	pbl_ext_en	1
35	ecap_bl_ext_	$urandom
36	tr_intrvl	800
37	sti_amp	$urandom
38	pulse_stim_d	$urandom_range(1,30)
39	frc_bl_ext_du	1
40	ecap_bl_ext_	$urandom_range(1,63)
41	ip_dly	$urandom_range(1,31)
42	acbl_ratio	1
43	n_pulse	4
44	pulse_dur	$urandom_range(1,63)
45	hi_res	$urandom
46	tr_mode	0
47	active_tr	1
48	priority_tr	0

PYTHON SCRIPT

- The script reads and stores data from 6 TEXT files, each is a section of the SV test code
- Functions are defined to read values from the CSV file and manipulate them to write to the register
  - For example, Consider Train1 config reg 0

Table-276: Train 1 configuration control register 0

Register: trv1_cfg_0
Address: 070

Bit	Type	Reset Value (hex)	Description
7:4	RES	0	**Reserved**
3:2	C	0	**current config setting** 00 = normal matching 01 = high matching 10 = reserved 11 = reserved
1:0	C	0	**train type setting** 00 = pulse 01 = pulse group 10 = arbitrary 11 = rotating electrodes

EXAMPLE FUNCTIONS

- This function defined in the script reads the value from the csv file and performs bit manipulation to program the register as shown below
- Here the function ifrandom checks if the parameter is random in the input file and generates a random value within the limits if true.

```
def trv_cfg_0():
        matching = ifrandom('matching',0,3)
        pulse_type = ifrandom('pulse_type',0,3)
        val = ((matching<<2) | pulse_type)
        val =format(val,'02X')
        val ='8\'h'+val
        return val
```

```
/*_____________CONFIG_____________*/
    register_prog(trv1_cfg_0_addr,8'h07);
```

EXAMPLE FUNCTIONS

- Another function that is important and can be used in functions for other registers is the get_value function, which retrieves the value from the csv file.
- This can be used to add more parameters in the future

```
def get_value(parameter):
        if os.path.exists(filename):
                with open(filename, 'r') as csvfile :
                        df = pd.read_csv(csvfile, index_col =0)
                        try:
                                val = df['tr_1'][parameter]
                        except:
                                print("Error in reading "+parameter
                                raise
        else:
                print("path does not exist")
                return
        if (val== None):
                return 0
        else:
                return val
```

```
def trv_on_dur():
        on_dur = get_value('on_dur')
        if (on_dur.find('$urandom') == -1):
                val =int(on_dur)
                val = format(val,'02X')
                val ='8\'h'+val
                return(val)
        else:
                return on_dur
```

RUNNING THE SCRIPT

- Currently the script allows the specification of the testname, path where the test is to be stored, number of trains, the input csv file.
- Provisions will be made to randomize a particular parameter based on the testcase
- Currently the script is incapable of generating testcases that have more than one train, this should be implemented soon.
- Therefore, sequential, interleave and multi train modes are done manually for now
- The command looks as follows in the linux environment
- $ python stdig_test_gen.py -option input
- Any other suggestions are welcome