Test vector generation in post silicon verification and validation process

Intro

David is an engineer working for Infineon, a semiconductor manufacturing company. His role involves verifying and validating semiconductors that are used in **safety** critical systems, such as those found in cars, airplanes, medical equipment, communication systems etc. David takes great pride in knowing that his work contributes to keeping people **safe** and **secure** in their everyday lives. He spends long hours running tests and analyzing data, ensuring that every chip meets the highest standards of quality and reliability. David's dedication to his work and its importance in ensuring **safety** make him an integral part of Infineon's team.

Samples

David's job involves testing multiple variants of chips (samples). Different chips are designed to meet different specifications and requirements depending on the specific use case they are intended for. For instance, chips designed for use in medical equipment may have different specifications than those designed for use in automotive systems. Additionally, chips may be delivered in different packages, such as a small outline package (SOP), a quad flat package (QFP), or a ball grid array (BGA). The package type can impact the chip's performance and can influence the test methods used by David and his team. Therefore, David must carefully consider the specifications of each chip sample and the package it is delivered in when developing his testing approach. To ensure that every chip meets the required specifications, David performs a range of tests on each sample, his attention to detail and expertise in testing ensure that Infineon's semiconductors are of the highest quality and reliability, regardless of the specific application or package they are designed for.

Let's stick to a single example where David is testing samples of the AURIX™ microcontroller family. David has been given the responsibility of conducting measurements for this project over the course of the next few months. The table below presents various chip variants that he will be working with, along with their corresponding sample code names.

Samples (Project.Samples)			
ID(.Id)	**Family name**(.FamilyName)	**Product name**(.ProducName)	**Name**(.Name)
1	AURIX™	TC3xx	POR1
2	AURIX™	TC2xx	SS1
3	AURIX™	TC2xx	POR1

Input conditions

As part of the testing process for each sample, David needs to test different input conditions that are specific to each chip's design and intended application. These input conditions can include variables such as temperature, humidity, input voltage etc. Testing under different input conditions is important because it can help identify potential issues that could arise under real-world operating conditions.

Input conditions(Project.InputConditions)				
ID(.Id)	Parameter(.Parameter)	Minimum(.Min)	Maximum(.Max)	Time Between Points(.TimeBetweenPoints)
1	Temperature	-40	150	5
2	Humidity	10	30	15
3	Input voltage	0	5	0
4	Frequency	5	50	0

Test point collections

While input conditions are specified on project level it could be that their boundaries are fully relevant for each sample. Each sample may be tested with different test point collections depending on its specific application and design. A semiconductor chip designed for use in an automotive application may need to be tested under extreme temperatures and varying conditions to ensure that it can operate reliably in the demanding environments of a car. In contrast, a chip designed for use in a home appliance may need to be tested under different humidity conditions to ensure it can perform reliably in a range of household environments.

While some input conditions may be specific to a particular sample, there may be some that can be shared across multiple samples. For example, testing under different input voltages may be relevant for all samples, regardless of their specific application. By sharing input conditions across multiple samples, David can optimize the testing process to ensure maximum efficiency while still maintaining the highest level of testing rigor.

Test point collections(Project.TestPointCollections)			
ID(.Id)	Input condition ID(.InputConditionId)	Sample ID(.SampleIds)	Test points(.TestPoints)
1	**1**	**1,2**	**[-40, -20, 0, 20, 40]**
2	**1**	**3**	**[0, 20, 40, 60, 80, 100, 120, 140]**
3	**2**	**1**	**[10, 20]**
4	**2**	**2,3**	**[20, 30]**
5	**3**	**1,3**	**[3.3]**
6	**3**	**2**	**[3.3, 5]**
7	**4**	**1**	**[5, 15]**
8	**4**	**2**	**[5, 25]**
9	**4**	**3**	**[5, 50]**

Test vectors

Test vectors are essentially sets of input values that are used to simulate the operating conditions of the chip and to verify its functionality. As part of the testing process, David needs to generate test vectors for each sample, based on the test point collections that are specific to each chip's design and intended application.

Test vectors for sample 1				
Index	**Temperature**	**Humidity**	**Input voltage**	**Frequency**
1	-40	10	3.3	5
2	-20	10	3.3	5
3	0	10	3.3	5
4	20	10	3.3	5
5	40	10	3.3	5
6	50	10	3.3	5
7	100	10	3.3	5
8	-40	20	3.3	5
9	-20	20	3.3	5
10	0	20	3.3	5
11	20	20	3.3	5
12	40	20	3.3	5
13	50	20	3.3	5
14	100	20	3.3	5
15	-40	10	5.0	5
16	-20	10	5.0	5
17	0	10	5.0	5
18	20	10	5.0	5
19	40	10	5.0	5
20	50	10	5.0	5
21	100	10	5.0	5
22	-40	20	5.0	5
23	-20	20	5.0	5
24	0	20	5.0	5
25	20	20	5.0	5
26	40	20	5.0	5
27	50	20	5.0	5
28	100	20	5.0	5

Engineer experience

While it's important for David to generate test vectors that accurately simulate the chip's operating conditions, not every possible combination of input variables needs to be tested. Some test vectors may not be possible or relevant for a given sample, and it's important to skip those test points to optimize the testing process.

With his experience and expertise, David can identify the most critical input conditions to test for each sample and generate test vectors that prioritize those variables. He can also recognize when certain combinations of input variables are not relevant or practical for a given sample, and skip those test points to save time and resources.

Another important consideration when generating test vectors is the order in which the input conditions are tested. Some input conditions may be easier or quicker to change than others, and it's important to take these factors into account when designing the testing process. For example, it may be easier and quicker to change input voltage than to change temperature, so it may make sense to frequently change voltage instead of temperature.

Properly sorting and prioritizing the input conditions for each sample can help optimize the testing process and ensure that the chip is thoroughly tested under a range of critical variables. By leveraging his experience and expertise, David can design test vectors that effectively simulate the chip's real-world operating conditions, without unnecessarily testing every possible combination of inputs. In a table you can find test vectors edited by David.

Filtered test vectors for sample 1					
Index	**Is used**	**Temperature**	**Humidity**	**Input voltage**	**Frequency**
1	TRUE	-40	10	3.3	5
2	TRUE	-40	10	5.0	5
3	TRUE	-40	20	3.3	5
4	TRUE	-40	20	5.0	5
5	TRUE	-20	10	3.3	5
6	TRUE	-20	10	5.0	5
7	TRUE	-20	20	3.3	5
8	TRUE	-20	20	5.0	5
9	TRUE	0	10	3.3	5
10	TRUE	0	10	5.0	5
11	TRUE	0	20	3.3	5
12	TRUE	0	20	5.0	5
13	TRUE	20	10	3.3	5
14	TRUE	20	10	5.0	5
15	TRUE	20	20	3.3	5
16	TRUE	20	20	5.0	5
17	TRUE	40	10	3.3	5
18	TRUE	40	10	5.0	5
19	TRUE	40	20	3.3	5
20	TRUE	40	20	5.0	5
21	TRUE	50	10	3.3	5
22	TRUE	50	10	5.0	5
23	TRUE	50	20	3.3	5
24	TRUE	50	20	5.0	5
25	TRUE	100	10	3.3	5
26	TRUE	100	10	5.0	5
27	FALSE	100	20	3.3	5
28	FALSE	100	20	5.0	5

Task

Assist David in his daily tasks of generating test vectors by developing an application that takes test point collections as input and provides him with a user-friendly GUI. The GUI should allow David to prioritize input conditions and exclude specific test vectors or combinations and generate a .csv file with the target test vectors as the end result. It is essential to optimize the application's performance as the list of test vectors can be extensive, to prevent David from wasting too much time. The application can be developed using any technology, and the inputs will be provided through a set of JSON files. Solutions will be evaluated both on performance and user experience.

Bonus tasks

Updatable test points

Test points within the collections can be updated, so the application should minimize repetitive work when David needs to modify the test vectors. Make sure that whenever a test point is updated the test vector containing that test point is also updated.

Input time constraints

By default all our inputs are treated the same. David decides what inputs should be of higher priority, meaning less changes on the environment for that input. Notice that our data set contains the average duration for changing the value within the environment, e.g. changing the temperature cannot be done immediately, while changing voltage is basically instant. Can you assist David in automatically ordering the inputs based on their switching duration? Note that it still should be possible to manually arrange the inputs afterwards.

Multiple output tables

Remember the sample names before? As the output table is valid for one sample only, David needs to execute the same procedure for multiple samples. Can you make it so that David only has to do it once, but receives multiple tables for his different samples?

Test vector editor

So we took the test points as out input, now it is time to work the other way around. Can you visualize the generated test vector in a table and make David be able to edit the test points within the test vector.

Inputs

All inputs you would need for the task can be found on the repository here:

https://github.com/kruljacInfineon/STEMGames2023

Please note that the repository will not become active before start of the assignment on Friday the 12th of May 2023. In the repository you will find multiple JSON files that represents the inputs described in the text above. These invaluable

resources can be utilized during both the development and presentation phases of your solution. Note that the blue text in the headers of each table above represent the JSON keys within the input documents.

Scoring

The task will be evaluated based on both performance and user experience. The following distribution of points will apply:

- Main Task (50 points)
- Updatable test points (15 points)
- Input time constraints (15 points)
- Multiple output tables (10 points)
- Test vector editor (10 points)

Note that in general a maximum of 100 points can be achieved, however bonus points are given for creativity which makes it possible to exceed 100 points. One example: you think of a very good additional feature that could be useful. If the jury finds the feature useful as well and if it was executed properly, you will be awarded 3. There is a maximum of 15 additional points.

For the main task, three major components are graded. Each component has several sub-components that are important.

Methodology (10 points)

- Version Control usage (e.g. Git)
- Task management (e.g. Jira, Trello, GitHub Project)
- Team work (e.g. good discussions, not one person did all commits)

User Experience (20 points)

- Style factor
- Intuitive, informative, proper use of language (e.g. grammar) and not cluttered.
- Clicks to action
- Accessibility Features (e.g. vision deficiencies)
- No exposed bugs or misbehaviors

Performance (20 points)

- Availability of benchmarks
- Speed compared to other teams
- Consistency
- Fluent UI (no spikes/freezes)
- Responsive (e.g. visual feedback)

For the bonus tasks, it is important to provide the described functionality. Distribution of the points for each bonus tasks depends on the completeness of your implementations.