Wafer Management

Release 1.0.0

Romain LAUP

Aug 04, 2023

CONTENTS

CHAPTER

ONE

VBD MODULE

VBD.**calculate_breakdown**(*X*, *Y*, *compliance*)

This functions calculates Voltage breakdown for two vectors given, and a compliance. Default value of the compliance is set to 1e-3

Parameters

- **X** (*<list>*) – Values of voltage. Will be converted to a np.array
- **Y** (*<list>*) – Values of current. Will be converted to a np.array and we will take the absolute value of the values
- **compliance** (*<float>*) – Value of compliance. By default is set to 1e-3

Return <float> Breakd_Volt
Voltage Breakdown. Can be Nan

Return <float> Breakd_Leak
Leakage Breakdown. Can be Nan

Return <bool> reached_compl
True if the current reached compliance, False otherwise. If it's False, Breakd_Volt will be Nan

Return <float> high_leak
Compliance if compliance is reached, highest value of the current otherwise

VBD.**create_wafer_map**(*wafer_id*, *session*, *structure_id*)

This function returns a plot converted to base64, so it can be sent to the User Interface. The plot shows the wafer map based on VBD. VBD are taken from the database. If a VBD is 'NaN', it's colored in black.

Parameters

- **wafer_id** (*<str>*) – the name of the wafer
- **session** (*<str>*) – Selected session
- **structure_id** (*<str>*) – the name of the structure

Return <list>
Plot converted to base64

VBD.**fig_to_base64**(*fig*)

Function used to convert a png to base64 to help communication between server and User. Used in ppt_matrix.

Parameters
fig (*<png>*) – a figure in png format.

Return <base64>
The converted figure

VBD.**get_all_x**(*wafer_id*, *session*, *structure_id*)

This function gets all coordinates x in a structure. Used to create the wafer map.

Parameters

- **wafer_id** (<*str*>) – the name of the wafer
- **session** (<*str*>) – name of the session
- **structure_id** (<*str*>) – the name of the structure

Return <list>
List of x in the structure

VBD.**get_all_y**(*wafer_id*, *session*, *structure_id*)

This function gets all coordinates y in a structure. Used to create the wafer map.

Parameters

- **wafer_id** (<*str*>) – the name of the wafer
- **session** (<*str*>) – Selected session
- **structure_id** (<*str*>) – the name of the structure

Return <list>
List of y in the structure

VBD.**get_compliance**(*wafer_id*, *session*)

This function finds the compliance from the specified session in the database Returns None if the structure has no compliance registered

Parameters

- **wafer_id** (<*str*>) – name of the wafer_id
- **session** (<*str*>) – name of the session

Return <str>
the compliance in the wafer

VBD.**get_vectors_in_matrix**(*wafer_id*, *session*, *structure_id*, *x*, *y*)

This function is used to get the values of voltages and current in a matrix. This function is always in parameters for calculate_breakdown

Parameters

- **wafer_id** (<*str*>) – the name of the wafer
- **session** (<*str*>) – Selected session
- **structure_id** (<*str*>) – the name of the structure
- **x** (<*str*>) – the horizontal coordinate of the matrix
- **y** (<*str*>) – the vertical coordinate of the matrix

Return <list> X
The values of voltage

Return <list> Y
The values of current

CHAPTER TWO

WAFERMAPS MODULE

`WaferMaps.`**`C_wafer_map`**`(`*wafer_id*, *session*, *structure_id*`)`

This function returns a plot converted to base64, so it can be sent to the User Interface. The plot shows the wafer map based on C. C are taken from the database.

Parameters

- **`wafer_id`** (<*`str`*>) – the name of the wafer
- **`session`** (<*`str`*>) – Selected session
- **`structure_id`** (<*`str`*>) – the name of the structure

Return <list>
Plot converted to base64

`WaferMaps.`**`Cmes_wafer_map`**`(`*wafer_id*, *session*, *structure_id*`)`

This function returns a plot converted to base64, so it can be sent to the User Interface. The plot shows the wafer map based on Cmes. Cmes are taken from the database.

Parameters

- **`wafer_id`** (<*`str`*>) – the name of the wafer
- **`session`** (<*`str`*>) – Selected session
- **`structure_id`** (<*`str`*>) – the name of the structure

Return <list>
Plot converted to base64

`WaferMaps.`**`Leak_wafer_map`**`(`*wafer_id*, *session*, *structure_id*`)`

This function returns a plot converted to base64, so it can be sent to the User Interface. The plot shows the wafer map based on Leak. Leak are taken from the database.

Parameters

- **`wafer_id`** (<*`str`*>) – the name of the wafer
- **`session`** (<*`str`*>) – Selected session
- **`structure_id`** (<*`str`*>) – the name of the structure

Return <list>
Plot converted to base64

`WaferMaps.`**`R_wafer_map`**`(`*wafer_id*, *session*, *structure_id*`)`

This function returns a plot converted to base64, so it can be sent to the User Interface. The plot shows the wafer map based on R. R are taken from the database. If an R is a broken value, like 999999 or 999997, it's colored in black.

Parameters

- `wafer_id` (<*str*>) – the name of the wafer
- `session` (<*str*>) – Selected session
- `structure_id` (<*str*>) – the name of the structure

Return <list>
Plot converted to base64

CHAPTER

THREE

APP MODULE

app.**C_WM**(*waferId*, *session*, *structure*)

Used to plot the wafer map based on Capacitance values

app.**C_normal**(*waferId*, *sessions*, *structures*, *coords*)

Used for plotting the normal plots of C. We first refactor received information and then call the function

app.**Cmes_WM**(*waferId*, *session*, *structure*)

Used to plot the wafer map based on Measured Capacitance values

app.**Cmes_normal**(*waferId*, *sessions*, *structures*, *coords*)

Used for plotting the normal plots of Cmes. We first refactor received information and then call the function

app.**Leak_WM**(*waferId*, *session*, *structure*)

Used to plot the wafer map based on Leakage values

app.**Leak_normal**(*waferId*, *sessions*, *structures*, *coords*)

Used for plotting the normal plots of Leak. We first refactor received information and then call the function

app.**R_WM**(*waferId*, *session*, *structure*)

Used to plot the wafer map based on resistance values

app.**R_normal**(*waferId*, *sessions*, *structures*, *coords*)

Used for plotting the normal plots of R. We first refactor received information and then call the function

app.**VBD_normal**(*waferId*, *sessions*, *structures*, *coords*)

Used for plotting the normal plots of selected VBD. We first refactor received information and then call the function

app.**delete_wafer**(*wafer_id*)

Used for deleting a wafer

app.**excel_structure_route**(*waferId*, *sessions*, *structures*, *types*, *temps*, *files*, *coords*, *file_name*)

Used for creating an Excel with selected parameters. We first refactor received information and then call the function

app.**filter_by_Coords**(*wafer_id*, *selectedMeasurement*)

Used for displaying all structures that contain the selected coordinates inside the given wafer.

app.**filter_by_Filenames**(*wafer_id*, *selectedMeasurement*)

Used for displaying all structures that contain the selected filename inside the given wafer.

app.**filter_by_Meas**(*wafer_id*, *selectedMeasurement*)

Used for displaying all structures that contain the selected type of measure inside the given wafer.

`app.`**`filter_by_Session`**(*wafer_id*, *selectedMeasurement*)

Used for displaying all structures that contain the selected session inside the given wafer.

`app.`**`filter_by_Temps`**(*wafer_id*, *selectedMeasurement*)

Used for displaying all structures that contain the selected temperature inside the given wafer.

`app.`**`getCSess`**(*waferId*)

Used to get all session that contain Capacitance values

`app.`**`getCStruct`**(*waferId*, *session*)

Used to get all structures that contain Capacitance values

`app.`**`getCmesSess`**(*waferId*)

Used to get all session that contain Measured Capacitance values

`app.`**`getCmesStruct`**(*waferId*, *session*)

Used to get all structures that contain Measured Capacitance values

`app.`**`getLeakSess`**(*waferId*)

Used to get all session that contain Leakage values

`app.`**`getLeakStruct`**(*waferId*, *session*)

Used to get all structures that contain Leakage values

`app.`**`getRSess`**(*waferId*)

Used to get all session that contain Resistance values

`app.`**`getRStruct`**(*waferId*, *session*)

Used to get all structures that contain Resistance values

`app.`**`get_all_coords`**(*wafer_id*)

Used for getting all coordinates inside a given wafer.

`app.`**`get_all_filenames`**(*wafer_id*)

Used for getting all filenames inside a given wafer.

`app.`**`get_all_structures`**(*wafer_id*)

Used for getting all structures in all sessions in a wafer.

`app.`**`get_all_temps`**(*wafer_id*)

Used for getting all temperatures inside a given wafer.

`app.`**`get_all_types`**(*wafer_id*)

Used for getting all types of measures inside a given wafer.

`app.`**`get_compl`**(*waferId*, *session*)

Used for getting the compliance of the selected session

`app.`**`get_map_sessions_server`**(*wafer_id*)

Used for getting all sessions that contain I-V measurements (for the wafer map)

`app.`**`get_map_structures_server`**(*wafer_id*, *session*)

Used for getting all structures that contain I-V measurements (for the wafer map)

`app.`**`get_matrices`**(*wafer_id*, *structure_id*)

Used for getting all dies in the structure selected app.**get_normal_values**(*waferId*)

Used for getting all extracted values in a wafer (Leak, R, C, Cmes and/or VBD)

app.**get_sessions_server**(*wafer_id*)

Used for getting all sessions inside a given wafer.

app.**get_structures_json**(*wafer_id*, *session*)

Used for getting all structures inside a given session in a wafer.

app.**index**()

Used for set up the app

app.**open**()

Used for displaying all registered wafers.

app.**options**(*checkbox_checked*)

Used for registering data in the database. We check if the user wants to register J-V measures. Then, we use the correct function for each type of file (txt, tbl or lim) tbl files are converted into txt. Finally, Upload Folder is cleared

app.**personal_wafer_map**(*waferId*, *session*, *structure*)

Used for plotting the wafer map of the selected structure

app.**plot_we_want**(*waferId*, *sessions*, *structures*, *types*, *temps*, *files*, *coords*)

Used for plotting the dies selected with selected parameters. We first refactor received information and then call the function

app.**ppt_structure_route**(*waferId*, *sessions*, *structures*, *types*, *temps*, *files*, *coords*, *file_name*)

Used for creating a Powerpoint with selected parameters. We first refactor received information and then call the function

app.**reg_excel_VBD**(*waferId*, *sessions*, *structures*, *temps*, *files*, *coords*, *file_name*)

Used for saving selected VBDs in an excel file. We first refactor received information and then call the function

app.**send_css**(*path*)

Used for set up the app

app.**send_js**(*path*)

Used for set up the app

app.**send_static_files**(*path*)

Used for set up the app

app.**serve**(*path*)

Used for set up the app

app.**set_compl**(*waferId*, *session*, *compliance*)

Used for setting the compliance of the selected session

app.**upload**()

Used for collect files dropped by user. We create an Upload Folder and then handle each file: Step 1: uncompress file if it is a compressed file Step 2: Manage lim files and file without extension

CHAPTER

FOUR

CONVERTER MODULE

`converter.`**`handle_file`**`(`*file_path*`)`

This function takes the path of a file and uncompress and convert it into a txt file. Files that can be handled: .tbl.Z, .tbl and .txt After conversion, the original file is deleted from the DataFiles folder

Parameters
`file_path` (<*`str`*>) – path of the file to be converted

Return <str>
path of the converted file

`converter.`**`tbl_to_txt`**`(`*path*`)`

Converts a tbl file into a txt file. The file is read, all information is converted and saved in a list and then we write all at once in an empty txt file.

Parameters
`path` – Path of the file to be converted

`converter.`**`traducer`**`(`*line*`)`

Used for converting tbl files into txt files. Parse a line and returns information into the format used in txt files. This function is called in tbl_to_txt.

Parameters
`line` (<*`str`*>) – Line to be parsed

Returns
Line converted to the format used in the txt files

CHAPTER

FIVE

EXCEL MODULE

`excel.classify_row(row)`

Used to know in which sheet the corresponding row has to go

Parameters
`row` – Row of the DataFrame

Returns
A str : 'Positives', 'Negatives' or 'NaN'

`excel.excel_VBD(wafer_id, sessions, structures, Temps, Files, coords, file_name)`

An Excel file is created with all VBDs inside the wafer, following selected filters. 3 sheets are created: one for positive values, one for negatives and one for NaN

Parameters

- `wafer_id` – ID of the wafer
- `sessions` – Filtered sessions
- `structures` – Filtered structures
- `Temps` – Filtered temperatures
- `Files` – Filtered files
- `coords` – Filtered coordinates
- `file_name` – Name under which the file will be registered

`excel.wanted_excel(wafer_id, sessions, structures, types, Temps, Files, coords, file_name)`

This function creates an Excel file with given information and register it in a folder named following the concerned wafer. We first get all information from the database into a Pandas' DataFrame, and then we write the DataFrame into an Excel. One column is: [name of the session, name of the structure, Unit of the Measure, [Measures]] One sheet is created per die and one file is created per type of measure Size of columns are automatically adjusted for better readability.

Parameters

- `wafer_id` – ID of the Wafer
- `sessions` – All sessions that we want to write
- `structures` – All structures that we want to write
- `types` – All types that we want to write
- `Temps` – All temperatures that we want to write
- `Files` – All files that we want to write

- `coords` – All coordinates that we want to write
- `file_name` – Name under which the file will be registered

CHAPTER

SIX

FILTER MODULE

`filter.`**`filter_by_coord`**(*coords*, *wafer_id*)

This function browse the database to find all structures that have a matrix with the couple of coordinates specified.

Parameters

- **`coords`** (<`list`>) – List of str, contains all couples of coordinates that want to be matched
- **`wafer_id`** (<`str`>) – Name of the wafer_id

Return <list>

A list of structures that contains the specified coordinates

`filter.`**`filter_by_filename`**(*file=<class 'str'>*, *wafer_id=<class 'str'>*)

This function browse the database to find all structures that have measurements from thr file specified.

Parameters

- **`file`** (<`list`>) – List of str, contains all files that wants to be matched
- **`wafer_id`** (<`str`>) – Name of the wafer_id

Return <list>

A list of structures that contains the specified files

`filter.`**`filter_by_meas`**(*meas*, *wafer_id*)

This function browse the database to find all structures that have the types of measurements specified.

Parameters

- **`meas`** (<`list`>) – List of str, contains all Measurements that wants to be matched
- **`wafer_id`** (<`str`>) – Name of the wafer_id

Return <list>

A list of structures that contains the specified measurements

`filter.`**`filter_by_session`**(*session=<class 'str'>*, *wafer_id=<class 'str'>*)

This function browse the database to find all structures that have measurements from thr file specified.

Parameters

- **`session`** (<`list`>) – List of str, contains all files that wants to be matched
- **`wafer_id`** (<`str`>) – Name of the wafer_id

Return <list>

A list of structures that contains the specified session `filter.`**`filter_by_temp`**(*temps*, *wafer_id*)

This function browse the database to find all structures that have the temperature specified.

Parameters

- **`temps`** (*<list>*) – List of str, contains all temperatures that wants to be matched
- **`wafer_id`** (*<str>*) – Name of the wafer_id

Return <list>

A list of structures that contains the specified temperatures

GETTER MODULE

`getter.connexion()`

Used to connect to the database, so we can manipulate data. /! Only use to read data, not to write data in the DB /! :return: The collection.

`getter.get_C_sessions(`*wafer_id*`)`

Used to get all sessions that contain C values (for wafer maps)

Parameters
`wafer_id` (<*str*>) – ID of the wafer

Return <list>
List of all sessions that contain C values

`getter.get_C_structures(`*wafer_id*, *session*`)`

Used to get all structures that contain C values (for wafer maps)

Parameters

- `wafer_id` (<*str*>) – ID of the wafer
- `session` (<*str*>) – Name of the session

Return <list>
List of all structures that contain C values

`getter.get_Cmes_sessions(`*wafer_id*`)`

Used to get all sessions that contain Cmes values (for wafer maps)

Parameters
`wafer_id` (<*str*>) – ID of the wafer

Return <list>
List of all sessions that contain Cmes values

`getter.get_Cmes_structures(`*wafer_id*, *session*`)`

Used to get all structures that contain Cmes values (for wafer maps)

Parameters

- `wafer_id` (<*str*>) – ID of the wafer
- `session` (<*str*>) – Name of the session

Return <list>
List of all structures that contain Cmes values

`getter.`**`get_Leak_sessions`**`(`*wafer_id*`)`

Used to get all sessions that contain Leak values (for wafer maps)

Parameters
: **`wafer_id`** (<*`str`*>) – ID of the wafer

Return <list>
: List of all sessions that contain Leak values

`getter.`**`get_Leak_structures`**`(`*wafer_id*, *session*`)`

Used to get all structures that contain Leak values (for wafer maps)

Parameters

- **`wafer_id`** (<*`str`*>) – ID of the wafer
- **`session`** (<*`str`*>) – Name of the session

Return <list>
: List of all structures that contain Leak values

`getter.`**`get_R_sessions`**`(`*wafer_id*`)`

Used to get all sessions that contain R values (for wafer maps)

Parameters
: **`wafer_id`** (<*`str`*>) – ID of the wafer

Return <list>
: List of all sessions that contain R values

`getter.`**`get_R_structures`**`(`*wafer_id*, *session*`)`

Used to get all structures that contain R values (for wafer maps)

Parameters

- **`wafer_id`** (<*`str`*>) – ID of the wafer
- **`session`** (<*`str`*>) – Name of the session

Return <list>
: List of all structures that contain R values

`getter.`**`get_compliance`**`(`*wafer_id*, *session*`)`

This function finds the compliance from the specified structure in the database Returns None if the structure has no compliance registered

Parameters

- **`wafer_id`** (<*`str`*>) – name of the wafer_id
- **`session`** (<*`str`*>) – name of the session

Return <str>
: the compliance in the wafer

`getter.`**`get_coords`**`(`*wafer_id*`)`

This function finds all the coordinates from the specified wafer in the database

Parameters
: **`wafer_id`** (<*`str`*>) – name of the wafer_id

Return <list>
: the list of coordinates in the wafer

getter.**get_db_name**(*db_name='New Wafers'*)

Used to know which database has to be opened. Default parameter can be changed manually, so a new database will be created if it doesn't exist yet.

Parameters
`db_name` (*<str>*) – Name of the database. Please change it to create a new database or switch to another existing

Returns
Name of the database

getter.**get_filenames**(*wafer_id*)

This function finds all the filenames in the specified wafer in the database

Parameters
`wafer_id` (*<str>*) – name of the wafer_id

Return <list>
the list of filenames in the wafer

getter.**get_map_sessions**(*wafer_id*)

Used to get all sessions that contain I-V measurements (for wafer maps) inside a wafer.

Parameters
`wafer_id` (*<str>*) – ID of the wafer

Return <list of str>
All sessions with I-V measurements inside the wafer

getter.**get_map_structures**(*wafer_id*, *session*)

Used to get all structures that contain I-V measurements (for wafer maps) inside the given session of a wafer.

Parameters

- `wafer_id` (*<str>*) – ID of the wafer
- `session` (*<str>*) – Selected session

Return <list of str>
All structures that contain I-V measurements inside the session

getter.**get_matrices_with_I**(*wafer_id*, *structure_id*)

This function finds all matrices that contain I-V measurements. Used to display buttons in the right place in the UI

Parameters

- `wafer_id` (*<str>*) – the name of the wafer
- `structure_id` (*<str>*) – the name of the structure

Return <list>
List of matrices that contains I-V measurements

getter.**get_sessions**(*wafer_id*)

Used to get all sessions inside a given wafer.

Parameters
`wafer_id` (*<str>*) – ID of the wafer

Return <list of str>
All sessions inside the wafer

getter.**get_structures**(*wafer_id*, *session*)

Used to get all structures inside the given session of a wafer.

Parameters

- **wafer_id** (*<str>*) – ID of the wafer
- **session** (*<str>*) – Selected session

Return <list of str>
All structures inside the session

getter.**get_temps**(*wafer_id*)

This function finds all the temperatures from the specified wafer in the database

Parameters
wafer_id (*<str>*) – name of the wafer_id

Return <list>
the list of temperatures in the wafer

getter.**get_types**(*wafer_id*)

This function finds all the types of measurements from the specified wafer in the database

Parameters
wafer_id (*<str>*) – name of the wafer_id

Return <list>
the list of types in the wafer

getter.**get_wafer**(*wafer_id*)

This function finds the wafer specified in the database. /!Only use to read data, not to write data in the DB /! :param <str> wafer_id: name of the wafer_id

Return <dict>
the wafer

CHAPTER

EIGHT

WAFER-MANAGEMENT

8.1 new_manage_DB module

`new_manage_DB.`**`create_db`**`(`*path*, *is_JV*`)`

Used to get information from .txt files. This function creates a database or open it if it already exists, and fill it with measurement information We put all the file's information in a dictionary, and then we write all the dictionary in the database, so we just call the db once. This is much faster. Also, indexes are created if they don't already exist, so searching in the database in faster

Parameters

- **`path`** (*<str>*) – path of the file
- **`is_JV`** (*<bool>*) – True if the user wants to register J-V measurements, False otherwise

Return <list>
a list containing all wafers that have been registered

`new_manage_DB.`**`create_db_it`**`(`*path*`)`

Used to get information from .txt files that contain data for It measurements. This function creates a database or open it if it already exists, and fill it with measurement information We put all the file's information in a dictionary, and then we write all the dictionary in the database, so we just call the db once. This is much faster. Also, indexes are created if they don't already exist, so searching in the database in faster

Parameters
`path` (*<str>*) – path of the file

Return <list>
a list containing all wafers that have been registered

`new_manage_DB.`**`create_db_lim`**`(`*path*`)`

Used to get information from .lim files. First, we read the lim file and get all information from it in a list. Then, we read the file without extension and merge information from both files. Finally, we write it in the database.

Parameters
`path` (*<str>*) – path of the file

`new_manage_DB.`**`create_db_tbl`**`(`*path*, *is_JV*`)`

Used to get information from .tbl files. This function creates a database or open it if it already exists, and fill it with measurement information We put all the file's information in a dictionary, and then we write all the dictionary in the database, so we just call the db once. This is much faster. Also, indexes are created if they don't already exist, so searching in the database in faster

Parameters

- **`path`** (*<str>*) – path of the file

- **`is_JV`** (*<bool>*) – True if the user wants to register J-V measurements, False otherwise

Return <list>
a list containing all wafers that have been registered

`new_manage_DB.`**`setCompliance`**(*waferId*, *session*, *compliance*)

8.2 normal_plots module

`normal_plots.`**`C_normal_distrib_neg`**(*wafer_id*, *sessions*, *structures*, *dies*)

Used to plot the normal distribution of negatives values of C inside a wafer, following selected filters. We first get data inside the database and then plot it in a figure. We use a probability scale, and plot the reference line and the 90% confidence interval

Parameters

- **`wafer_id`** – ID of the wafer
- **`sessions`** – Selected sessions
- **`structures`** – Selected structures
- **`dies`** – Selected dies

Returns
The plot, converted into base64

`normal_plots.`**`C_normal_distrib_pos`**(*wafer_id*, *sessions*, *structures*, *dies*)

Used to plot the normal distribution of positive values of C inside a wafer, following selected filters. We first get data inside the database and then plot it in a figure. We use a probability scale, and plot the reference line and the 90% confidence interval

Parameters

- **`wafer_id`** – ID of the wafer
- **`sessions`** – Selected sessions
- **`structures`** – Selected structures
- **`dies`** – Selected dies

Returns
The plot, converted into base64

`normal_plots.`**`Cmes_normal_distrib_neg`**(*wafer_id*, *sessions*, *structures*, *dies*)

Used to plot the normal distribution of negative values of Cmes inside a wafer, following selected filters. We first get data inside the database and then plot it in a figure. We use a probability scale, and plot the reference line and the 90% confidence interval

Parameters

- **`wafer_id`** – ID of the wafer
- **`sessions`** – Selected sessions
- **`structures`** – Selected structures
- **`dies`** – Selected dies

Returns
The plot, converted into base64

`normal_plots.`**`Cmes_normal_distrib_pos`**`(`*wafer_id*, *sessions*, *structures*, *dies*`)`

Used to plot the normal distribution of positive values of Cmes inside a wafer, following selected filters. We first get data inside the database and then plot it in a figure. We use a probability scale, and plot the reference line and the 90% confidence interval

Parameters

- **`wafer_id`** – ID of the wafer
- **`sessions`** – Selected sessions
- **`structures`** – Selected structures
- **`dies`** – Selected dies

Returns

The plot, converted into base64

`normal_plots.`**`Leakage_normal_distrib_neg`**`(`*wafer_id*, *sessions*, *structures*, *dies*`)`

Used to plot the normal distribution of negative values of Leakage inside a wafer, following selected filters. We first get data inside the database and then plot it in a figure. We use a probability scale, and plot the reference line and the 90% confidence interval

Parameters

- **`wafer_id`** – ID of the wafer
- **`sessions`** – Selected sessions
- **`structures`** – Selected structures
- **`dies`** – Selected dies

Returns

The plot, converted into base64

`normal_plots.`**`Leakage_normal_distrib_pos`**`(`*wafer_id*, *sessions*, *structures*, *dies*`)`

Used to plot the normal distribution of positive values of Leakage inside a wafer, following selected filters. We first get data inside the database and then plot it in a figure. We use a probability scale, and plot the reference line and the 90% confidence interval

Parameters

- **`wafer_id`** – ID of the wafer
- **`sessions`** – Selected sessions
- **`structures`** – Selected structures
- **`dies`** – Selected dies

Returns

The plot, converted into base64

`normal_plots.`**`R_normal_distrib_neg`**`(`*wafer_id*, *sessions*, *structures*, *dies*`)`

Used to plot the normal distribution of negative values of R inside a wafer, following selected filters. We first get data inside the database and then plot it in a figure. We use a probability scale, and plot the reference line and the 90% confidence interval

Parameters

- **`wafer_id`** – ID of the wafer
- **`sessions`** – Selected sessions

- **structures** – Selected structures
- **dies** – Selected dies

Returns
The plot, converted into base64

normal_plots.**R_normal_distrib_pos**(*wafer_id*, *sessions*, *structures*, *dies*)

Used to plot the normal distribution of positive values of R inside a wafer, following selected filters. We first get data inside the database and then plot it in a figure. We use a probability scale, and plot the reference line and the 90% confidence interval

Parameters

- **wafer_id** – ID of the wafer
- **sessions** – Selected sessions
- **structures** – Selected structures
- **dies** – Selected dies

Returns
The plot, converted into base64

normal_plots.**VBD_normal_distrib_neg**(*wafer_id*, *sessions*, *structures*, *dies*)

Used to plot the normal distribution of negative values of VBD inside a wafer, following selected filters. We first get data inside the database and then plot it in a figure. We use a probability scale, and plot the reference line and the 90% confidence interval

Parameters

- **wafer_id** – ID of the wafer
- **sessions** – Selected sessions
- **structures** – Selected structures
- **dies** – Selected dies

Returns
The plot, converted into base64

normal_plots.**VBD_normal_distrib_pos**(*wafer_id*, *sessions*, *structures*, *dies*)

Used to plot the normal distribution of positive values of VBD inside a wafer, following selected filters. We first get data inside the database and then plot it in a figure. We use a probability scale, and plot the reference line and the 90% confidence interval

Parameters

- **wafer_id** – ID of the wafer
- **sessions** – Selected sessions
- **structures** – Selected structures
- **dies** – Selected dies

Returns
The plot, converted into base64

normal_plots.**get_values**(*wafer_id*)

Used to know which extracted value is inside a given wafer, so we display only available options to the user

Parameters
wafer_id – ID of the wafer

Returns
list of all Extracted values inside the wafer

8.3 plot_and_powerpoint module

plot_and_powerpoint.**fig_to_base64**(*fig*)

Function used to convert a png to base64 to help communication between server and User. Used in ppt_matrix.

Parameters
fig (*<png>*) – a figure in png format

Return <base64>
The converted figure

plot_and_powerpoint.**get_wafer**(*wafer_id*)

This function finds the wafer specified in the database

Parameters
wafer_id (*<str>*) – name of the wafer_id

Return <dict>
the wafer

plot_and_powerpoint.**plot_wanted_matrices**(*wafer_id*, *sessions*, *structures*, *types*, *Temps*, *Files*, *coords*)

Used to plot only selected dies following selected filters. These plots won't be registered but will be displayed for the user. One plot is created per type of Measurement. A single plot contain all session, all dies and all structures selected. A list of 15 colors is generated, so up to 15 differents lines can be differenciated. You can add a color in this list if you want (Line 44 to 60) but don't forget to change the number in line 109 (color_index % <this number>)

Parameters

- **wafer_id** – ID of the wafer
- **sessions** – Selected sessions
- **structures** – Selected structures
- **types** – Selected types
- **Temps** – Selected temperatures
- **Files** – Selected files
- **coords** – Selected coordinates

Returns
One plot per type of measurements, converted to base64

plot_and_powerpoint.**wanted_ppt**(*wafer_id*, *sessions*, *structures*, *types*, *Temps*, *Files*, *coords*, *file_name*)

Used to create a PowerPoint (registered in a folder named after the concerned wafer) with only selected dies following selected filters. These plots will be registered in a folder named 'plots'. One plot is created per type of Measurement. A single plot contain all session, all dies and all structures selected. A list of 15 colors is generated, so up to 15 differents lines can be differenciated on the plot. You can add a color in this list if you want (Line 160 to 176) but don't forget to change the number in line 227 (color_index % <this number>)

Parameters

- **wafer_id** – ID of the wafer
- **sessions** – Selected sessions

- **structures** – Selected structures
- **types** – Selected types
- **Temps** – Selected temperatures
- **Files** – Selected files
- **coords** – Selected coordinates
- **file_name** – Name given to the file

Returns
One plot per type of measurements, converted to base64

8.4 split_data module

split_data.**C_spliter**(*line*)

This function takes a line of datas from a .txt file containing C-V measurements in argument and returns a list with voltage in first position, RS in second position and CS in third position

Parameters
line (*<str>*) – The line you want to extract information

Returns
information needed

Return type
list of str

split_data.**converter_split**(*line*)

Used to handle lines when getting data from a tbl file

Parameters
line (*<str>*) – Line to be handled

Returns
Line handled

split_data.**converter_split_session**(*line*)

Used to get the name of the session when getting data from a tbl file.

Parameters
line – Line to be handled

Returns
Line handled

split_data.**dataSpliter**(*line*)

This function takes a line of datas from a .txt file in argument and returns a list with voltage in first position and current in second position

Parameters
line (*<str>*) – The line you want to extract the information

Returns
the information needed

Return type
list of str

`split_data.`**`spliter`**`(`*line*`)`

This function takes a line of a header from a .txt file in argument and returns the relevant information in this line

Parameters
: **`line`** (<*`str`*>) – The line you want to extract information

Returns
: Information needed

Return type
: str

8.5 test_DataExtraction module

`test_DataExtraction.`**`test_filter`**`()`

CHAPTER

NINE

TEST MODULE

CHAPTER

TEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX