viewer

November 8, 2019

[1]:
```
%matplotlib inline
import pymongo

import numpy as np
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

from viewing import *
from util import *
```

1 CONTROL E IMPLEMENTACIÓN DE UNA PUF EN UN MICROPROCESADOR

1.1 Sergio Vinagrero Gutiérrez

[2]:
```
# Configuration of the database
client = pymongo.MongoClient("mongodb://localhost:27017/")
database = client['thesis']
db_dumps = database['dumps']

# List of recurring data
boards_ids = list(set(i['board_id'] for i in db_dumps.find({},{"_id": 0,␣
 ↪"board_id": 1})))
```

[8]:
```
# Configuration of data frames
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
pd.set_option('display.max_colwidth', -1)

# Configuration of matplotlib and seaborn
plt.rcParams['xtick.labelsize'] = 12
plt.rcParams['ytick.labelsize']= 12
plt.rcParams['axes.titlesize']= 18
```

```
plt.rcParams['axes.labelsize']= 12
plt.rcParams['figure.titlesize']= 18
plt.rcParams['figure.figsize'] = [28, 14]  # Scale up plots
```

[9]:
```
# Cargamos los datos de los parametros de cada placa
extracted_ids_df = pd.read_csv('./extracted_ids.csv', index_col=None)
```

[10]:
```
dropdown_wafer = widgets.
 ↪Dropdown(options=create_dropdown_values(extracted_ids_df['Wafer']),
                                    description='Oblea:')

boards_ids_output = widgets.Output()
summary_boards_out = widgets.Output()

dropdown_boards_eh = generate_simple_eventhandler(boards_ids_output,
                                                  extracted_ids_df, 'Wafer')

dropdown_wafer.observe(dropdown_boards_eh, names='value')
lista_placas = widgets.VBox([dropdown_wafer, boards_ids_output])

with summary_boards_out:
    print("Numero de placas: " + str(len(extracted_ids_df['Board'].index)))
    print("Numero de obleas: " + str(len(set(extracted_ids_df['Wafer']))))
    print("Numero de lotes : " + str(len(set(extracted_ids_df['Lot']))))

tab_boards_ids = widgets.Tab([lista_placas, summary_boards_out])
tab_boards_ids.set_title(0, 'Lista de placas')
tab_boards_ids.set_title(1, 'Resumen')

display(tab_boards_ids)
```

```
Tab(children=(VBox(children=(Dropdown(description='Oblea:', options=('Ninguno', 'Todos', 30, 34
```

[11]:
```
final_df = pd.read_csv('./all_boards_individual.csv', index_col=None)
final_df.sort_values(['Region'], ascending=[True], inplace=True)

region_diffs_mean_df = pd.read_csv('./all_regions_mean.csv')
```

[12]:
```
dropdown_region = widgets.
 ↪Dropdown(options=create_dropdown_values(final_df['Region']),
                                    description='Region:')

regions_output = widgets.Output()
plot_regions_output = widgets.Output()
region_summary_output = widgets.Output()
```

```
blue_sns_palette = sns.color_palette("ch:3.5,-.3,dark=1")

with region_summary_output:
    display(region_diffs_mean_df)

def dropdown_region_eh(change):
        regions_output.clear_output()
        plot_regions_output.clear_output()

        if (change.new == 'Ninguno'):
            regions_output.clear_output()
            plot_regions_output.clear_output()

        elif (change.new == 'Todos'):
            with regions_output:
                display(final_df)
            fig, ax = plt.subplots()
            fig.set_size_inches(28, 180)
            ax = sns.violinplot(y="Region", x="Diff", data=final_df,␣
 ↪inner='point',
                                cut=0.0, orient='h', scale='count',␣
 ↪palette=blue_sns_palette)

            with plot_regions_output:
                display(plt.show())

        else:
            data_df = final_df[final_df['Region'] == change.new]
            with regions_output:
                display(data_df)

            fig, ax = plt.subplots()
            ax = sns.violinplot(y="Region", x="Diff", data=data_df, orient='h',
                                cut=0.0, palette=blue_sns_palette)
            with plot_regions_output:
                display(plt.show())


dropdown_region.observe(dropdown_region_eh, names='value')

tab_regions_all = widgets.Tab([regions_output, plot_regions_output,␣
 ↪region_summary_output])
tab_regions_all.set_title(0, 'Diffs de regiones')
tab_regions_all.set_title(1, 'Distribucion')
tab_regions_all.set_title(2, 'Resumen')

display(dropdown_region)
```

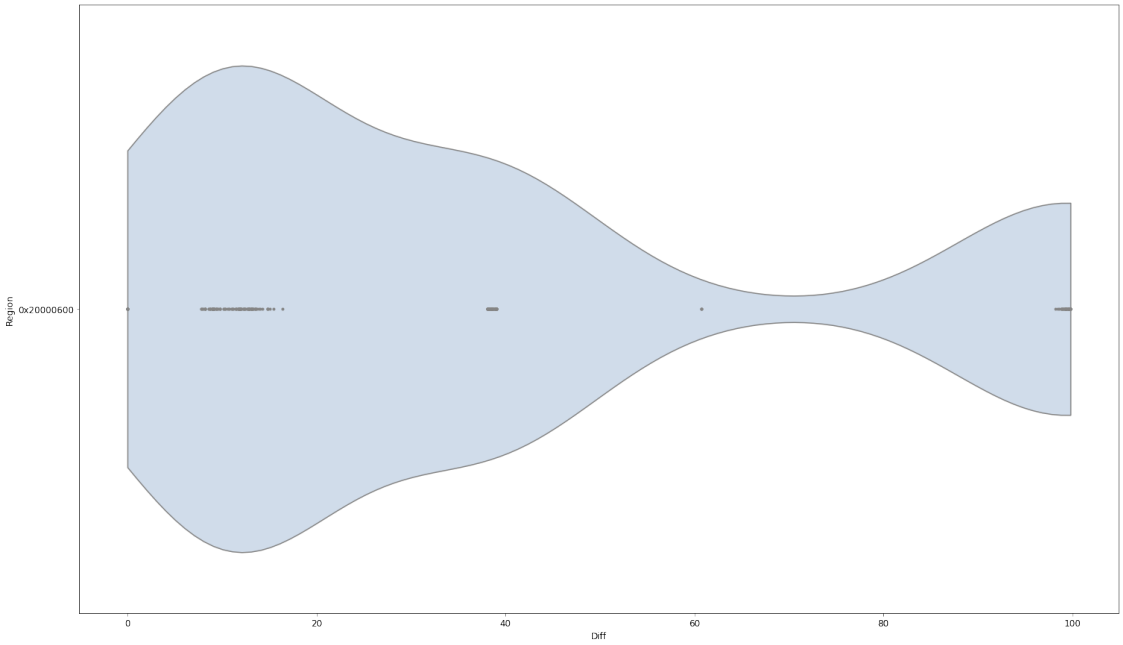
```
display(tab_regions_all)
```

```
Dropdown(description='Region:', options=('Ninguno', 'Todos', '0x20000000', '0x20000200', '0x200
```

```
Tab(children=(Output(), Output(), Output()), _titles={'0': 'Diffs de regiones', '1': 'Distribuc
```

[9]:
```
data_df = final_df[final_df['Region'] == '0x20000600']
fig, ax = plt.subplots()
fig.set_size_inches(25, 15)

ax = sns.violinplot(y="Region", x="Diff", data=data_df, cut=0.0, orient='h',
                    inner='point', palette=sns.color_palette("ch:3.5,-.
 ↪3,dark=1"))
plt.show()
```



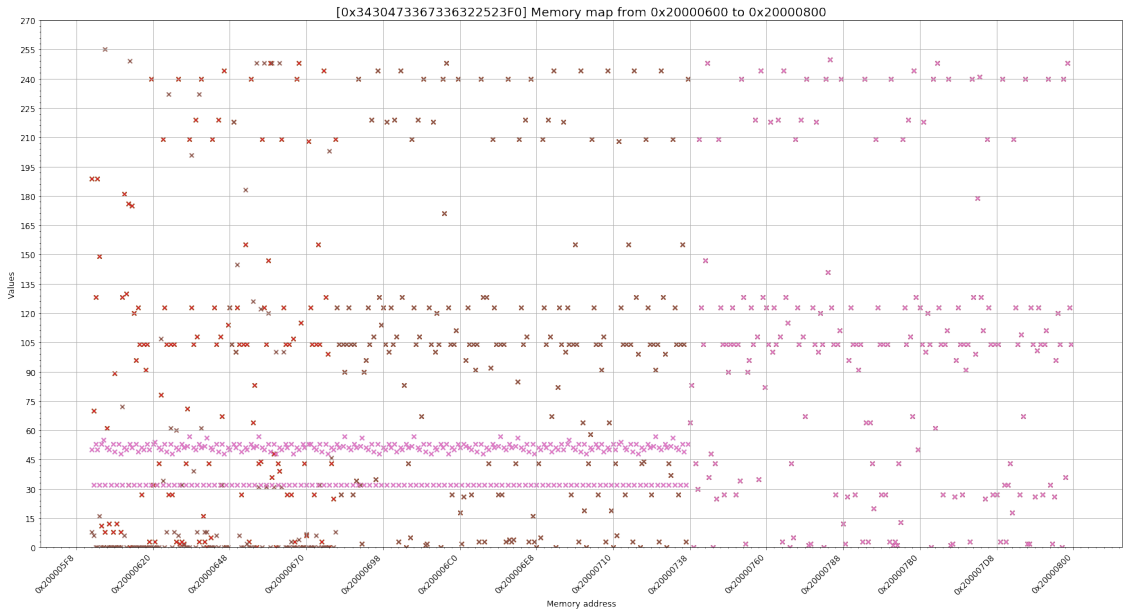
[12]:
```
from bokeh.plotting import figure
from bokeh.io import output_notebook, show

output_notebook()
```

[ ]:
```
```

[13]:
```
plot_memory_regions_bokeh(boards_ids[3], 3, 4, show_every=True)
plot_memory_regions_bokeh(boards_ids[2], 3, 4, show_every=True)
```

[14]:
```
for bid in boards_ids[0:7]:
    plot_memory_regions(bid, 3, 4, show_every=False)
```



[26]:
```
# Compare the same memory region in different boards

# memory_regions_uut = list(set([mem_pos['mem_pos'] for mem_pos in␣
 ↪board_dumps]))
# final_column_names = ["Board_1", "Board_2", "Region", "Id_1", "Id_2", "Diff"]

# fdata_df = pd.DataFrame(columns=final_column_names)

# for region_uut in memory_regions_uut:
#     dumps_board_0 = list(db_dumps.find({'mem_pos': region_uut, 'board_id':␣
 ↪boards_ids[0]}))
#     dumps_board_1 = list(db_dumps.find({'mem_pos': region_uut, 'board_id':␣
 ↪boards_ids[1]}))

#     num_samples = min(len(dumps_board_0), len(dumps_board_1))

#     for sample in range(num_samples):
#         diff = calculate_diff_one_board(dumps_board_0[sample],␣
 ↪dumps_board_1[sample])

#         data = {'Board_1': dumps_board_0[sample]['board_id'],
#                 'Board_2': dumps_board_1[sample]['board_id'],
#                 'Region': region_uut,
#                 'Id_1': str(dumps_board_0[sample]['_id']),
```

```
#                 'Id_2': str(dumps_board_1[sample]['_id']),
#                 'Diff': diff}

#         fdata_df = fdata_df.append(data, ignore_index=True)

# final_is_valid = fdata_df['Diff'] < 70
# fdata_df = fdata_df[final_is_valid]

# fdata_df.sort_values(['Diff'], ascending=[True], inplace=True)
# fdata_df
```

[16]:
```
# Compare all memory regions in all of the boards
# Filter only the ones with a low difference
#
# boards_dumps_df holds all the dump differences of the same
# board which are in the good range

column_names = ["Board", "Region", "Id_1", "Id_2", "Diff"]
boards_dumps_df = pd.DataFrame(columns=column_names)

output = parallelize_call_ma(compare_dumps_one_board, boards_ids[2:])

for o_df in output[0]:
    o_df = thresh_results_df(o_df, 5, 21)
    boards_dumps_df = boards_dumps_df.append(o_df, ignore_index=True)
```

[17]:
```
print(f'Number of results: {len(boards_dumps_df.index)}')
boards_dumps_df.head(10)
```

```
Number of results: 369
```

[17]:

	Board	Region	Id_1	Id_2	Diff
0	0x30314710303537323E0372	0x20001400	5db99afaddc2ad5d58d2040d	5db99baeddc2ad5d58d204ed	20.703125
1	0x30314710303537323E0372	0x20001400	5db99afaddc2ad5d58d2040d	5db99d3eddc2ad5d58d2050d	20.703125
2	0x30314710303537323E0372	0x20001400	5db99b47ddc2ad5d58d2042d	5db99baeddc2ad5d58d204ed	20.703125
3	0x30314710303537323E0372	0x20001400	5db99b47ddc2ad5d58d2042d	5db99d3eddc2ad5d58d2050d	20.703125
4	0x30314710303537323E0372	0x20001800	5db99afaddc2ad5d58d2040f	5db99baeddc2ad5d58d204ed	20.703125
5	0x30314710303537323E0372	0x20001800	5db99b47ddc2ad5d58d2042f	5db99baeddc2ad5d58d204ed	20.703125
6	0x30314710303537323E0372	0x20003800	5db99afcddc2ad5d58d2041f	5db99baeddc2ad5d58d204ed	20.703125
7	0x30314710303537323E0372	0x20003800	5db99afcddc2ad5d58d2041f	5db99d3eddc2ad5d58d2050d	20.703125
8	0x30314710303537323E0372	0x20003800	5db99afcddc2ad5d58d2041f	5db99baeddc2ad5d58d204ed	20.703125
9	0x30314710303537323E0372	0x20003800	5db99b49ddc2ad5d58d2043f	5db99baeddc2ad5d58d204ed	20.703125

```
3  5db99d3eddc2ad5d58d2050d  20.703125
4  5db99b73ddc2ad5d58d2048f  19.140625
5  5db99b73ddc2ad5d58d2048f  19.140625
6  5db99b80ddc2ad5d58d204bf  20.898438
7  5db99bb1ddc2ad5d58d204ff  20.117188
8  5db99d41ddc2ad5d58d2051f  20.117188
9  5db99b80ddc2ad5d58d204bf  20.898438
```

[43]:
```
# Calculate the mean and std_deviation of the diff from
# the data frame with the data in the range we desire

boards_umt_df = boards_dumps_df

# Not all boards are within the good range we want
good_boards = list(set(boards_umt_df['Board']))
good_regions = list(set(boards_umt_df['Region']))

mean_df = pd.DataFrame(columns=['Board', 'Region', 'Mean_Diff', 'Median',␣
↪'Std_Dev', 'Num_Samples'])

# Iterate over all the good boards
for gb in good_boards:

    board_selector = boards_umt_df['Board'] == gb
    board_ut_df = boards_umt_df[board_selector]
    mem_regions = list(set(board_ut_df['Region']))

    # Iterate over the good regions of the board
    for region in mem_regions:
        region_selector = board_ut_df['Region'] == region
        board_region_df = board_ut_df[region_selector]

        data = {'Board': gb,
                'Region': region,
                'Mean_Diff': board_region_df['Diff'].mean(),
                'Median': board_region_df['Diff'].median(),
                'Std_Dev': board_region_df['Diff'].std(),
                'Num_Samples': len(board_region_df.index)
               }

        mean_df = mean_df.append(data, ignore_index=True)
```

```
      ␣
↪---------------------------------------------------------------------------
```

```
NameError                                 Traceback (most recent call last)

<ipython-input-43-43082b2dc41f> in <module>
  2 # the data frame with the data in the range we desire
  3
----> 4 boards_umt_df = boards_dumps_df
  5
  6 # Not all boards are within the good range we want


NameError: name 'boards_dumps_df' is not defined
```

[226]:
```
mean_df.sort_values(['Board', 'Region'], ascending=[True, True], inplace=True)
print(f'Number of results: {len(mean_df.index)}')
mean_df
```

```
Number of results: 51
```

[226]:

	Board	Region	Num_Samples	Mean_Diff	Deviation
6	0x30314710303537323E0372	0x20000600	34	8.978631	0.660126
17	0x30314710303537323E0372	0x20000800	25	19.835938	0.936822
18	0x30314710303537323E0372	0x20000a00	10	20.039062	0.505973
22	0x30314710303537323E0372	0x20000c00	28	19.294085	1.518809
14	0x30314710303537323E0372	0x20000e00	17	19.772518	0.712721
9	0x30314710303537323E0372	0x20001000	10	19.804688	0.480185
7	0x30314710303537323E0372	0x20001200	13	20.177284	0.512949
25	0x30314710303537323E0372	0x20001400	4	20.703125	0.000000
8	0x30314710303537323E0372	0x20001600	3	20.312500	0.000000
3	0x30314710303537323E0372	0x20001800	2	19.140625	0.000000
4	0x30314710303537323E0372	0x20001a00	2	20.312500	0.000000
15	0x30314710303537323E0372	0x20001c00	1	20.898438	NaN
21	0x30314710303537323E0372	0x20001e00	10	20.468750	0.082351
24	0x30314710303537323E0372	0x20002000	7	19.196429	0.461013
23	0x30314710303537323E0372	0x20002800	1	20.703125	NaN
5	0x30314710303537323E0372	0x20002a00	1	20.703125	NaN
26	0x30314710303537323E0372	0x20002c00	15	19.921875	0.676582
20	0x30314710303537323E0372	0x20002e00	3	19.531250	1.014874
12	0x30314710303537323E0372	0x20003000	13	20.282452	0.743070
13	0x30314710303537323E0372	0x20003200	17	19.990809	0.816879
11	0x30314710303537323E0372	0x20003400	4	20.605469	0.112764
10	0x30314710303537323E0372	0x20003800	11	20.081676	0.476964
19	0x30314710303537323E0372	0x20003a00	11	20.205966	0.342367
27	0x30314710303537323E0372	0x20003c00	20	18.291016	2.886034
16	0x30314710303537323E0372	0x20003e00	7	19.280134	0.878648
42	0x30314710303537323E03A0	0x20000600	1	8.789062	NaN

38	0x30314710303537323E03A0	0x20000a00	1	18.554688	NaN
48	0x30314710303537323E03A0	0x20000e00	1	19.140625	NaN
39	0x30314710303537323E03A0	0x20001000	1	16.601562	NaN
44	0x30314710303537323E03A0	0x20001200	1	19.140625	NaN
40	0x30314710303537323E03A0	0x20001400	1	14.453125	NaN
46	0x30314710303537323E03A0	0x20001600	1	20.898438	NaN
47	0x30314710303537323E03A0	0x20002c00	1	20.703125	NaN
45	0x30314710303537323E03A0	0x20002e00	1	20.507812	NaN
43	0x30314710303537323E03A0	0x20003000	1	20.312500	NaN
41	0x30314710303537323E03A0	0x20003400	1	20.312500	NaN
49	0x30314710303537323E03A0	0x20003c00	2	15.429688	4.419417
50	0x30314710303537323E03A0	0x20003e00	2	19.335938	1.933495
2	0x3031471737343534300 3E0	0x20000600	66	12.556226	1.240369
1	0x3031471737343534300 3E0	0x20003e00	4	19.677734	0.907380
29	0x3031471737343534300460	0x20000600	1	11.718750	NaN
37	0x343047183673363222090	0x20000600	1	11.132812	NaN
36	0x343047183673363222090	0x20003e00	2	20.800781	0.138107
33	0x34304718367336323B0140	0x20000600	1	12.890625	NaN
0	0x343047183673363244043 0	0x20003c00	1	12.304688	NaN
32	0x343047336733632252 3F0	0x20003e00	2	20.410156	0.138107
28	0x34304733673363 23A03F0	0x20003e00	1	18.359375	NaN
34	0x343047A367336322440380	0x20003c00	1	12.304688	NaN
35	0x343047A367336322440380	0x20003e00	1	18.554688	NaN
31	0x343047A3673363244 02D0	0x20000600	1	14.257812	NaN
30	0x343047A3673363244 02D0	0x20003e00	1	19.921875	NaN

[ ]:
```
# TODO: Make a heatmap with the mean and deviation of the memory of a board to␣
↪show how it changes
```

[ ]:
```

```