Loading Corners and Report Generation

Corner Setup

- Importing Corners to Adexl from csv file will save us time instead of loading corners one by one.
- Csv file that needs to be loaded should be in a specific format cadence can accept.
- Variables which are not in format will be ignored by adexl while loading.

Perl code

csvgenerate.pl

```perl
use strict;
use warnings;
# Loading  the Text::CSV_XS and Spreadsheet::ParseXLSX modules
use Text::CSV_XS;
use Spreadsheet::ParseXLSX;
# Creating a new csv file
my $csv = Text::CSV_XS->new({  });
# Creating a new Spreadsheet
my $parser = Spreadsheet::ParseXLSX->new();
#giving name of the input that need to be read
my $excel_file = 'input.xlsx';
#converting file into readable by perl(parsing)
my $excel_workbook = $parser->parse($excel_file);

if (!defined $excel_workbook) {
    die $parser->error(), ".\n";
}
#getting list of work sheets
my @worksheets = $excel_workbook->worksheets();
# Initializing array
my @data;
#this loop will read all data in excel and arranges in data variable in same matrix form
for my $worksheet (@worksheets) {
    my ($row_min, $row_max) = $worksheet->row_range();
    my ($col_min, $col_max) = $worksheet->col_range();
    #for each row and column(element by element)
    for my $col ($col_min..$col_max) {
        my @column_data;
        for my $row ($row_min..$row_max) {
            my $cell = $worksheet->get_cell($row, $col);
        #pushes only if value is there
            if ($cell) {
                push @column_data, $cell->value();
            }
        }
        push @data, \@column_data;#storing in data variable
    }
}
```

Perl code

```perl
#declaring corner variables
my @corner;
my @enable;
my @moduleName;
my @vdd12;
my @temperature;
#pushing strings
push @corner, "Corner";
push @enable, "Enable";
push @moduleName, $data[3][0];
push @vdd12, "vdK";
push @temperature, "Temperature";
#there three loops as we require combinations of three variable values
for my $l (0..$#{$data[0]}) {
    for my $m (0..$#{$data[1]}) {
        for my $n (0..$#{$data[2]}) {
            next if (!$data[0][$l] ||  !$data[2][$m] || !$data[2][$n]);
            #checking whether value exists for each variable
            my $modName = "$data[0][$l]";
            my $vdd = "$data[1][$m]";
            my $tmp = "$data[2][$n]";
            #pushing in series like vector
            push @moduleName, $modName;
            push @vdd12, $vdd;
            push @temperature, $tmp;
        }
    }
}
#assigning corner name and mentioning true
for my $i (0..$#temperature) {
    push @corner, "C$i";
    push @enable, "t";
}
#generating csv file
my $csv_file = 'OUT.csv';
# opening the output file for writing
open(my $fh, '>', $csv_file) or die "Could not open file '$csv_file' $!";
```

Perl code

```
75  open(my $fh, '>', $csv_file) or die "Could not open file '$csv_file' $!";
76  print $fh join(",", @corner) . "\n";#appending each vector elements in the format
77  print $fh join(",", @enable) . "\n";
78  print $fh join(",", @vdd12) . "\n";
79  print $fh join(",", @temperature) . "\n";
80  print $fh join(",", @moduleName) . "\n";
81
82
83  close $fh;
84
```

Input file Format

input.xlsx

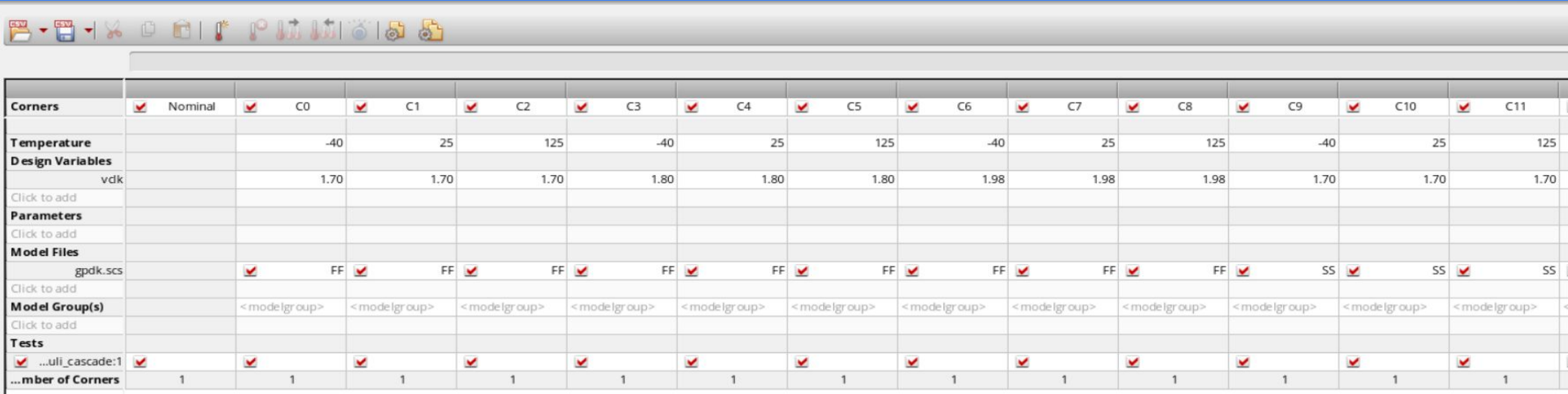
	A	B	C	D
1	t FF	1.70	-40	Modelfile::/cad/C
2	t SS	1.80	25	
3	t NN	1.98	125	
4	t FS			
5	t SF			

Output file Format

OUT.csv

```
Corner,C0,C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13,C14,C15,C16,C17,C18,C19,C20,C21,C22,C23,C
Enable,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t
vdK,1.70,1.70,1.70,1.80,1.80,1.80,1.98,1.98,1.98,1.70,1.70,1.70,1.80,1.80,1.80,1.98,1.98,1.98,
Temperature,-40,25,125,-40,25,125,-40,25,125,-40,25,125,-40,25,125,-40,25,125,-40,25,125,-40,2
Modelfile::/cad/Cadence6/FOUNDRY/analog/180nm/models/spectre/gpdk.scs,t FF,t FF,t FF,t FF,t FF
```

Loaded corner



Corners	Nominal	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
Temperature		-40	25	125	-40	25	125	-40	25	125	-40	25	125
Design Variables													
vdk		1.70	1.70	1.70	1.80	1.80	1.80	1.98	1.98	1.98	1.70	1.70	1.70
Click to add													
Parameters													
Click to add													
Model Files													
gpdk.scs		FF	FF	FF	FF	FF	FF	FF	FF	FF	SS	SS	SS
Click to add													
Model Group(s)		<modelgroup>	<modelgroup>	<modelgroup>	<modelgroup>	<modelgroup>	<modelgroup>	<modelgroup>	<modelgroup>	<modelgroup>	<modelgroup>	<modelgroup>	<modelgroup>
Click to add													
Tests													
...uli_cascade:1													
...mber of Corners	1	1	1	1	1	1	1	1	1	1	1	1	1

Report Generation

rd.py > ...

```python
import pyskill
import pandas as pd
import time

# Connect to the Cadence session
cadence = pyskill.Cadence()

# Get the current schematic
schematic = cadence.get_schematic()

# Get all the MOSFETs in the schematic
mosfets = schematic.get_instances('M')

# Create an empty DataFrame to store the MOSFET data
data = pd.DataFrame(columns=['Time', 'Instance name', 'Vgs value', 'Vds value', 'Operating region'])

# Loop over time from 0ns to 500ns with 50ns intervals
for t in range(0, 501, 50):
    # Wait for 50ns
    time.sleep(0.05)

    # Loop over all the MOSFETs and add their data to the DataFrame
    for mosfet in mosfets:
        name = mosfet.name
        vgs = mosfet.get_property_value('VGS')
        vds = mosfet.get_property_value('VDS')
        vth = mosfet.get_property_value('VTH')

        if vgs < vth:
            region = 'Cutoff'
        elif vds >= vgs - vth:
            region = 'Saturation'
        else:
            region = 'Linear'
```

```
            region = 'Linear'

        data = data.append({'Time': t, 'Instance name': name, 'Vgs value': vgs, 'Vds value': vds, 'Operating region': region},

    # Write the DataFrame to the Excel file
    writer = pd.ExcelWriter('mosfet_data.xlsx')
    data.to_excel(writer, index=False)
    writer.save()

print('Data written to mosfet_data.xlsx')
```

/home/Ganga/simulation/mouli_cascade/spectre/schematic/psf/spectre.out@mayur.phd

File Edit View Help

cadenc

```
Reading file:  /cad/Cadence6/FOUNDRY/analog/180nm/models/spectre/gpdk.scs
Reading file:  /cad/Cadence6/FOUNDRY/analog/180nm/models/spectre/mos25gen.scs
Reading file:  /cad/Cadence6/FOUNDRY/analog/180nm/models/spectre/nmos1.scs
Reading file:  /cad/Cadence6/FOUNDRY/analog/180nm/models/spectre/pmos1.scs
Reading file:  /cad/Cadence6/FOUNDRY/analog/180nm/models/spectre/resistor.scs
Reading file:  /cad/Cadence6/FOUNDRY/analog/180nm/models/spectre/capacitor.scs
Reading file:  /cad/Cadence6/FOUNDRY/analog/180nm/models/spectre/diode.scs
Reading file:  /cad/Cadence6/FOUNDRY/analog/180nm/models/spectre/bipolar.scs
Reading file:  /cad/Cadence6/FOUNDRY/analog/180nm/models/spectre/rfmos.scs
Reading file:  /cad/Cadence6/FOUNDRY/analog/180nm/models/spectre/xjvar_nf36.scs
Reading file:  /cad/Cadence6/FOUNDRY/analog/180nm/models/spectre/mcxjvar_w40.scs
Reading file:  /cad/Cadence6/FOUNDRY/analog/180nm/models/spectre/snacapacitor.scs
Reading file:  /cad/Cadence6/FOUNDRY/analog/180nm/models/spectre/cmodel.scs
Reading file:  /home/Ganga/Mouli_200020027_r#26d/rd.py

Notice from spectre during Digital Vector read-in.
    Process Vector Files.
    File read: /home/Ganga/Mouli_200020027_r#26d/rd.py
Error found by spectre during Digital Vector read-in.
    ERROR (USIMPRS-17865): Vector file `/home/Ganga/Mouli_200020027_r#26d/rd.py' does not contain the data section.

Time for NDB Parsing: CPU = 117.544 ms, elapsed = 688.483 ms.
Time accumulated: CPU = 137.469 ms, elapsed = 688.489 ms.
Peak resident memory used = 41.4 Mbytes.

Time for parsing: CPU = 111 us, elapsed = 111.103 us.
Time accumulated: CPU = 138.231 ms, elapsed = 689.251 ms.
Peak resident memory used = 41.4 Mbytes.

~~~~~~~~~~~~~~~~~~~~
Pre-Simulation Summary
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~

Aggregate audit (11:00:53 AM, Mon Apr 24, 2023):
Time used: CPU = 145 ms, elapsed = 696 ms, util. = 20.9%.
Time spent in licensing: elapsed = 13 ms.
Peak memory used = 42 Mbytes.
Simulation started at: 11:00:52 AM, Mon Apr 24, 2023, ended at: 11:00:53 AM, Mon Apr 24, 2023, with elapsed time (wall clock): 696 ms.
spectre completes with 1 error, 0 warnings, and 2 notices.
spectre terminated prematurely due to fatal error.
```

Thank You