Appendix 4

Simulator: equations.java

1. Introduction

An important part of this work is related to the analysis of the three proposed clock skew models. It was necessary to program a tool to make the simulations required to check the behaviour of each model. In this appendix, the operation of a JAVA program specifically developed to calculate clock skew estimations according to model equations is presented.

The programming language chosen is JAVA. The main reason is the characteristic of platform independence. It means that this program written in the Java language runs similarly on diverse hardware.

The operation can be summarized as a program that takes parameters (technology parameters) from an input file (with a specific format) or the keyboard and apply over them the model equations. The results are the clock skew estimations, which are presented in an output file and the computer screen.

2. Program operation

This program is executed from the command line. We need to have installed in the computer the Java Runtime Environment or JRE, which is the software required to run any application deployed on the Java Platform.

To start the program, we execute it in this way:

```
C:\Borland\Projects\equations>java equations
```

After this, a menu is showed on the screen:

```
Introduce what do you want to do:
      1. Utilize Kugelmass & Steiglitz Clock-Skew Model
      2. Utilize Zarkesh-Ha, Mule & Meindl Clock-Skew Model
      3. Utilize Jiang & Horiguchi Clock-Skew Model
      4. Utilize Kugelmass & Steiglitz Clock-Skew Model (without
optimal lenths)
      5. Utilize Jiang & Horiguchi Clock-Skew Model (without optimal
lenths)
```

Five options are possible. Options 1, 2 and 3 correspond with the three analysed clock skew model as they were described (buffered H-tree with buffers at the split points and tapered H-tree without intermediate buffers). Options 4 and 5 are the models 1 and 3 respectively adapted to the specific H-tree proposed to analyse the models in this work, where wires are partitioned with buffers to optimize their delay.

Next step is to choose the parameter input. It can be a file, which name is required to type through the command line, or the keyboard, where each parameter has to be individually introduced through the command line.

```
Choose input:
      1. File
      2. Keyboard
      3. Get input format file
```

There is a third option that allows us to get a file with the required input file format. This format is different for each model because they consider different parameter variations to calculate the clock skew. The format must be correct; otherwise the programs will show an error message. Next, an example of configuration file is shown. It is the case of the input file that model 3 requires in the 130 nm technology.

```
R_int=26.27e3
C_int=0.91e-10
R_0=3.94e3
C_0=0.77e-15
V_DD=1.2
V_t=0.19
D=17.32e-3
Dev_V_t=0.042
Dev_u=0.02
Dev_t_ox=0.013
Dev_W=0.05
Dev_L_eff=0.05
Dev_W_int=0.03
Dev_t_int=0.03
Dev_T_ild=0.03
```

Input File: Model 3, 130 nm parameters.

Next step in the program execution is to input the number of levels of the H-tree.

```
Number of levels in the H-tree:
```

After that, the name of the output file has to be introduced.

```
Type the name of the output file
```

After this step, results are displayed on the screen. Also, they can be read in the output file.

```
Total Clock-Skew: 1.6509922986937226E-9
```

Output example.

Model 2 has an important particularity. Authors propose the equations with 50 % of time delay in Sakurai expressions, but if we want that clock skew estimations be comparable with the estimations of models 1 and 3, equations must be calculated with 90 % of time delay. It is possible since during model 2 execution, we have to choose which time delay we want.

```
    Choose Sakurai's model:
       1. 50 % time delay
       2. 90 % time delay
```

To type all the options required for each simulation can be laborious if several simulations have to be realised. There is an alternative way to execute this program by introducing options and file names as additional parameters in the command line. It is shown in the following commands.

```
java equations <In file> <Out file> <Model> <No. levels> {<delay>}

java equations in.txt out.txt 1 8
java equations in.txt out.txt 3 8
java equations in.txt out.txt 4 8
java equations in.txt out.txt 5 8
java equations in.txt out.txt 2 8 2
```

3. Recommendations

When many simulations have to be realised, to execute the program lots of times can be laborious. It is recommendable to type a macro that helps us to make similar simulations. An example is when a model is analysed for different number of levels. The required macro is the following.

```
    java equations in.txt out.txt 4 1
    java equations in.txt out.txt 4 2
    java equations in.txt out.txt 4 3
    java equations in.txt out.txt 4 4
    java equations in.txt out.txt 4 5
    java equations in.txt out.txt 4 6
```

Other recommendation that makes our work easy is to redirect the output flow from the screen to a file thanks the parameter to redirect “>”.

```
Macro > results.txt
```

In this way, after executing the macro, all the results can be displayed in a single file facilitating their post-processing.

```
    C:\>java equations in.txt out.txt 4 1
       Clock-Skew: 5.418359660062558E-10

    C:\>java equations in.txt out.txt 4 2
       Clock-Skew: 1.7925917276162354E-9

    C:\>java equations in.txt out.txt 4 3
       Clock-Skew: 1.9237312208357593E-9

    C:\>java equations in.txt out.txt 4 4
       Clock-Skew: 3.427741470061498E-9

    C:\>java equations in.txt out.txt 4 5
       Clock-Skew: 2.774911991533342E-9

    C:\>java equations in.txt out.txt 4 6
       Clock-Skew: 4.255477662366857E-9
```

Macro output example.