WEEKLY REPORT 7 25.11.2005

This week, we continued researching the technologies we mentioned in our requirement analysis report and began defining the classes of our application.

Eren YILMAZ

This week, I worked on JHDL in detail. I first tried to understand the JHDL language structure, and read the User's Manual. While reading, I also tried to run some JHDL codes in its own simulator, “Dynamic Test Bench”. The whole system seems good, however, it does everything in backwards that we want to do, for example, the integral simulator takes a pre-compiled code of the circuit and then simulates and draws the circuit. That order confused me a lot. Then, I looked for a way to run the JHDL simulator from an external application, but there was a point that we missed: JHDL is not a library. It's a whole application, and seemingly it will not fit into an application easily. We may even take a radical decision and completely abandon using JHDL.

To the end of this week, we altogether began working on the class structure of our application.

Ilgın YARIMAĞAN

This week I continued working on the multithreading part of our project. I examined the methods and structure of the Thread and Runnable classes in javax.swing package. I considered, which classes of our project shall extend from these classes and which methods shall be overidden. Before doing that I tried to determine which processes in our program should form a thread class, in other words which processses should work concurrently with others.

After constructing the Thread classes in a general way, I focused on how these classes will interact with each other, which methods of the Thread class they shall call. For example I considered which one should call a *wait to* wait for another one, which one should call *notify* in inform the others not to wait anymore.

Finally I worked on generating the class diagrams of these Thread classes.

Mehtap Ayfer PARLAK

This week Ilgın, and me generally focused on the multithreading subject and how can we use the threads in our project. Multithreading is necessary for our project in order to use CPU efficiently. Thus, I searched the thread usage through internet and found some examples that we can use. Then I red the threads subject from our Operating Systems book. Finally, we determined roughly, which parts of our project must be a different thread. Namely, printing operations, file operations, GUI editing and simulation operations must be separate threads.

Besides searching threads, I also investigated the source code of a simulation tool; MicroDev in order to gain information about how to use and implement threads in a simulation program.

Emin ÖZCAN

This week, we have begun to prepare the initial design report. Firstly, i investigated JHDL deeply. I compiled a few JHDL codes,and by using Dynamic Test Bench program, i tried to run them. After running JHDL codes, we understood that this program does the reverse whatever we want. We want to draw a circuit and then simulate it, but this program does the reverse. In order to escape from this condition, we have researched bit coding, Java assembly and reflection in Java. According to our researches, we realized that reflection and bit coding in Java is very hard to work with. So we fell down in a dilemma, we could continue with JHDL or we could form our own classes instead of the classes in JHDL. Finally, we decided to solve this problem after the meeting with our project assistant on Friday. In addition to this, we discussed our classes and formed some basic classes. Also we discussed on the thread mechanism and we determined some basic threads.

M. Ergin SEYFE

This week I worked with the design of our project. Firstly, I tried to construct the class diagram of our project. Eren and Emin sent me a report about the class architecture of JHDL then I draw two different class diagrams and proposed them to group members. Also I looked at JHDL and tried to build a project using JHDL jar file. I faced with some problems. Then we organized a teem meeting. In this meeting our main problems was how we can use this library. There is a tool named as DTB(**dynamic test bench)** in this library. We need to give a compiled Java file(class file) to this simulator. I had an experience in constructing byte code (using BCEL Library) in run time. But this class has a dynamic constructor so it is very difficult to build a byte code in this situation. Also after doing this hard work we are not sure to use the outputs of DTB efficiently. Meanwhile, we can't find a way to use JHDL without DTB in our project. Therefore we decided to leave JHDL. Most probably we will use our class files instead of JHDL. Today we will meet again and give our final and important decision.