There are some subjects that seem to cause endless confusion. One of those is about the differences between emulation, acceleration, prototyping and simulation. Part of the confusion is caused by marketing departments, part because of changing technology, part because of changing users and part because there is no clear answer as to which one is right for a particular application until you get into the details. In this blog I will try to dispel some of the confusion, but we shall see if I succeed or not.

Before we get into the difference, let's talk about what is common between them. They all start by taking a model of all or part of a hardware system. The model can be subjected to stimulus and the results observed. They can all perform this task before availability of the actual hardware in silicon. I think that it is about it for the commonality. The only one of them that does not require special hardware is the simulator which is capable of running on a general purpose computer.

There is another primary distinction that can be made. Simulators and accelerators act on the model of the hardware, whereas emulation and prototyping create an actual implementation of the design, although not the same as the one that will ultimately appear in silicon. If we take a simulator as the baseline, emulation, acceleration and prototyping can all run faster than a simulator, but the speed-up is dependent on many factors.

As far as users are concerned, hardware developers are more likely to use simulation, emulation or acceleration. This is primarily because they provide a better hardware debug environment than a prototype. On the other hand, a software developer is most likely to use a prototype as this provides the highest performance. It gets most of this additional performance because it does not have the same level of debug capability.

Now, this is where we have to be very careful, because a prototype can be configured to have almost unlimited debug capabilities, and when it is used in this manner, it starts to look very much like an emulator. In addition, some emulators use field programmable gate arrays (FPGAs) which is the same hardware component used to build prototypes. Some emulators and accelerators use custom chips, but this is only a manifestation of the implementation, it does not mean it is an emulator or accelerator because it uses custom chips.
Also for the purposes of this discussion, there are few discernible differences between an emulator and an accelerator and so as far as the user is concerned, this is a marketing distinction. Don’t get me wrong, there are differences in their fundamental hardware, but again, this is an implementation distinction only.

In reality, the biggest difference between an emulator and a prototype is the way in which they are packaged as a product. A prototype tends to be a lot more do-it-yourself whereas an emulator attempts to make the whole process look as much like a software simulator as possible and that includes attributes such as flexibility, ease of use, connection to other environments and ease with which a design can be mapped. This extra packaging is expensive.

As an example, consider the process of mapping a design onto the hardware. All of the solutions are composed of an array of chips (could be custom, could be an FPGA), where a small part of the design can be mapped into each of these chips. Thus multiple chips are required. In an emulator, it is likely that the user has no idea how many chips exist in the array or how the design is mapped onto them. For a prototype, it is likely that the user will have manually or semi-automatically divided the design into pieces and decided what should go into each of the chips. The software will assist with stitching the pieces back together again to make the complete design.

This is clearly still scratching the surface of the differences and will be continued in another blog. If anything I have covered so far is not clear, please let me know. That way I can refine it before I move on.