

CONFUSION

We are now seeing the mass movement into the ESL design methodology. The Early Adopters have paved the way and although there is no questioning the advanced skill level of this next group of SoC designer, there is some confusion in the design community. This Industry Note and a following Industry Note will focus on the methodology and some of the dos and don'ts of the methodology.

ARCHITECTS, ESL DESIGNERS AND SOFTWARE ENGINEERS

Firmware Engineers are the guys that write the drivers, kernels and if need be the embedded OSs. They tend to be EEs. Embedded "Programmers" write the applications and they can be either EEs or Computer Science grads. Software Engineer is a term that has just become popular in the last five years. It is being used to differentiate the Firmware, Embedded and Software Architects that are trying to deal with concurrent software issues. They call themselves Software Engineers to differentiate themselves from the programmers that just deal with straight sequential programming. Therefore there is really not much difference between a Firmware and an Embedded Software Engineer.

In 1997 we did a survey and came up with around 400 architects. We projected 1,970 by the year 2000. That obviously makes them a really small market. What we didn't understand then was that the architect is really not the ESL market at all. John Darringer at IBM calls the System Architect the artist. The problem today is that systems are so complex that the System Architect has lost the ability to predictably define a system that is actually achievable, particularly when you throw in the power problem. This has spread to even fairly simple embedded board designs. Today an Embedded Board Designer, who is the default System Architect for simple systems, can't just swap in a more powerful microcontroller to solve his performance problem, because of the power issue.

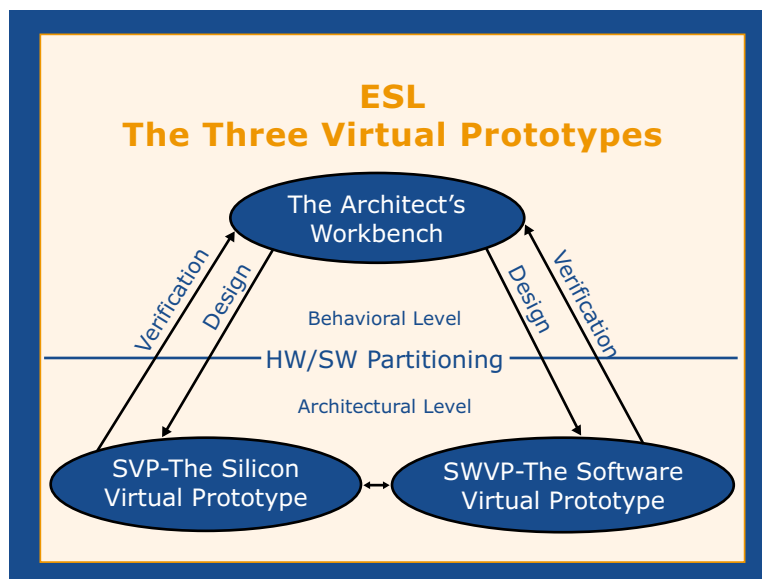
What companies like IBM are doing is putting a team of ESL Designers as close to the System Architect as possible, even physically close. Their job is to do the modeling of the proposed system and then extensive what-if analysis to come up with the optimal design. So today system partitioning is done by a multidiscipline team under the Architect's direction. This is why John desperately needs a Behavioral SystemC standard. In one talk he stated that not too long ago IBM had 44 models for a cache memory, none of which were compatible with each other nor compatible with any other of their Behavioral C models. That of course made simulation and therefore what-if analysis impossible. In the European Cell Phone companies they have gone to a System Architect, and an ESL Design Group, for each application which reports into a Chief System Architect, and an ESL Design Group for the complete phone.

WARNING – DO NOT CONSIDER BEHAVIORAL ESL DESIGN SOFTWARE, IT'S MODELING

This is usually the first mistake designers make when entering the world of ESL design. It's easy to do as many ESL Designers use C/C++. The other predominant language is MathWorks' "M", which is a modeling language. That's exactly what the ESL Designers are doing with C/C++; using it to build models.

What you are seeing is the ESL Designers are using the **Architect's Workbench** to model the system. This way they can do the what-if-analysis needed to prove out and optimize the system design. The Architect's Workbench is the first of the three Virtual Prototypes required for ES Level design (See Figure 1).

FIGURE 1: THE THREE VIRTUAL PROTOTYPES



(Source: Gary Smith EDA, February 2011)

Once the modeling, Behavioral Simulation and Hardware/Software partitioning is completed the portion that will become software is handed down to the Software Virtual Prototype (SWVP). That which will become hardware is handed down to the Silicon Virtual Prototype (SVP). There the design is continued to be optimized, simulated and checked to insure compliance with the specification. During these procedures the SVP and the SWVP continue to communicate and often further HW/SW partitioning is necessary. When completed the SW is compiled into finished C/C++ code and the HW is synthesized into RTL code for Implementation. The design phase is completed.