Implementation of a scalable and configurable program for automatic generation of memory circuit layout in nanometer technologies

Axel Andersson and John Gustavsson

Department of Electrical and Information Technology
Lund University

Right now, all across the world, production processes are successively getting more and more automated. Naturally this includes the memory industry as well, where the memory compiler is an important tool to improve effectiveness.

Memory compilers are designed for the purpose of automatically generating the layout of a memory's circuit. All memory vendors use this kind of tool to generate memories for their customers. Compilers are designed to support generation of memories with different attributes, e.g. speed, power consumption and various sizes. The tool may be very advanced and take a lot of time to develop. It is therefore necessary to make the implementation of the compiler configurable and scalable. A compiler has to be somewhat technology independent as well, e.g. being compatible for both 130 nm and 65 nm technologies, so that it is usable for a long period of time. This will make it efficient to use the compiler in the long run compared to a manual circuit design, where almost everything in the design process becomes obsolete after a technology switch.

During this thesis a compiler was implemented for an architecture developed at Lund University for Static Random-Access Memories (SRAM), a fast memory type used in for example computer systems as cache memory. The compiler is close to being solely dependant on the architecture of the memory, i.e. the logic function of each component. Therefore most of the components are interchangeable, given that they are compatible with it's surrounding blocks of course. Because of this, the compiler can be used to generate memories with different speed and bit over area density constraints. Support for configurable sizing of the memory was also achieved. Not only in the sense of total number of bits stored, but also how many addresses and how many bits there is per address. Having a compiler which is flexible is really what it is all about, since this means that the memory provider can support a wide range possible customer requests.

Sometimes there may be a discrepancy between what the customer needs and what the vendors can provide. Therefore one of the primary goals was to make an implementation with a general purpose approach. A compiler which is technology independent as well as the source code being structured in a convenient manner. This makes it possible for other designers to improve the compiler over time, even though new technologies are released, and there by keeping the compiler relevant and up to date.

For now the functionality of the compiler is limited to only working for one architecture. This is something for further work to improve. A lot of the source code can be used for other architectures, thanks to the general purpose approach, which should ease inclusion of more architectures and functionality.