Interconnect and Geometric Layout in Hydra

John T. O'Donnell

Hydra is a functional computer hardware description language that allows circuit designs to be specified either with or without information about the geometric layout. Portions of a design may be specified at different levels of abstraction, and some may have geometric layouts while others do not. This talk describes how geometric information is incorporated in Hydra circuit specifications, and discusses its interaction with equational reasoning about the behaviour of circuits.

Hydra models every circuit as a function from inputs to outputs. A circuit may be defined directly as a function from signals to signals, or indirectly through the use of combinators that define families of related circuits. There are also several domain-specific sublanguages, such as a language for designing control algorithms, from which control circuits can be synthesised.

A number of alternative circuit semantics are provided. Some of them are concerned only with behaviour, so that the application of a circuit to suitable inputs will perform a simulation. Others are concerned with structure, so that execution of the same circuit function will generate a netlist. The netlist semantics is based on an algebraic data type for representing the abstract structure of the circuit.

A non-geometric circuit specification can use an arbitrary organisation of the input and output ports; they are simply function parameters. A geometric specification, however, treats a circuit as a rectilinear box with a sequence of ports on each of the four edges. Each port is identified as either input or output (there is also an experimental bidirectional port type). The circuit function uses only input ports as parameters (with a "wildcard pattern" in the position of outputs), and produces results only for the output ports.

Hydra is a functional CHDL, not a relational one, so adjacent circuits must in general be connected, not simply composed. This is performed by a combinator that takes two circuits with geometric layout, and defines a new circuit with the connections completed on their common edge. The connection is implemented via a program transformation. In earlier versions of Hydra (the combined geometric and functional specification was first implemented in the 1980s) the program transformation was performed manually. A new experimental prototype implementation uses Template Haskell to perform the program transformations automatically, making Hydra's geometric combinators far more usable.

Hydra is similar in many ways to Lava+Wired. There are also some significant differences. Wired is relational while Hydra is functional. The mechanisms used for making connections along the edges are also somewhat different. However, the motivations behind Hydra and Lava+Wired appear to be identical. The talk will compare the two languages in more detail.