Universitaet Kaiserslautern
Lehrstuhl für Rechnerstrukturen

Bau 12/4, Postfach 3049,
D–675 Kaiserslautern, F.R.G.

Description of the KARL System and Hardware Description Language

The KARL System is the implementation of the KARL hardware description language. It includes a compiler and a simulator. A number of other CAD tools for VLSI have interfaces to it. KARL–III is written in Standard PASCAL. It is available under VMS on VAX–11. Upon special request also versions can be made available, running under UNIX or UNIX–alike on VAX–11, Apollo Domain, PCS Cadmus, and SUN workstations.

KARL–III is a highly mnemomic PASCALish non–procedural register transfer (RT) language including primitives for hardware description at RT level, gate level, circuit level, and mixtures of these levels. At circuit level KARL uses a technology– independant notation suitable for modelling all types of buses, bus drivers, bus receivers etc (pull–up, pull–down, and three–state). A simple digital model is provided for all kinds of transistors. Also a model for dynamic circuitry, such as e. g. with pass transistors, is included. Also a simple timing model is provided by delay primitives for exact delays and delay ranges.

KARL is a structural, behavioural, and functional language. KARL structural features provide a simple **func** function module, as well as a VLSI–oriented **cell** module declaration mechanism distinguishing port locations at four different sides (front, back, left, right). For instantiation of cells and multi–level cell hierarchies a powerful **make** command provides **rotate** and **mirror** transforms along with horizontal and vertical abutment including automatic wiring with mismatch diagnostic.

Other structural features are KARL *wiring operators*, such as **shift** and **shuffle** operators at bit level and word level, which support concepts of direct implementation of algorithms onto silicon in a variety of application areas, such as e.g. sorting, signal processing, systolic arrays, multiprocessor architectures etc.

Functional primitives e. g. are supporting compact hardware specification by KARL description. KARL functional/behavioural primitives include: multiply, divide, add, substract, relational operators, logical operators, multiplexers, demultiplexers, increment, decrement, priority, parity, decode, encode, type check, clocks, registers (latch–mode, edge–triggered, master/slave, and mixed mode), read/write memories, and read–only registers and memories.

The KARL system supports conceptual and logic hardware design from a RT level KARL functional specification by stepwise refinement downto gate level and circuit level. The simulator also processes mixtures of all three levels of description, so that the logical correctness of the embedding of circuit descriptions within higher level descriptions may be checked. We call this *"logic simulation at circuit level"*.

The KARL system defines 3 different languages: (1) KARL itself, (2) SCIL, the simulator control and test description language, and (3) RTcode format, the executable object data structure produced by the KARL compiler. All 3 languages are interfaces to a number of other CAD tools provided by the CVT project, the EIS project, or by our own group, such as e. g. the ABLED interactive graphic RT level synthesis tool using a block diagram/floor plan notation.

The KARL simulator supports debugging by a restrictive modelling philosphy which has been implemented by a very careful application of the "?" response ("undefined" bit value). Conditional SCIL commands may be used to avoid excessive simulator listings. SCIL also has a macro definition mechanism. SCIL may be used interactively, or in batch mode.