# YAL Language Specification for Netlists

Ted Manikas

Sept. 19, 1998

The YAL (Yet Another Language) language was developed by Bryan Preas and Ken Roberts at Xerox PARC (Palo Alto Research Center) in 1987. It is used to represent the current MCNC benchmark netlist circuits. Each module is described by a *template* as shown in Figure 1. The template syntax is the following:

关键字 **UPPERCASE** describes keywords of the YAL language.

<名称> < *anglebrackets* > enclose intermediate definitions of the language, such as module or signal names.

*option*1|*option*2 means either option1 *or* option2 is specified.

[**square brackets** ] enclose optional items.

. . . indicate that a preceeding field or line should be repeated as necessary.

Each module has a unique *module name* and a *module type*. The module types are:

**STANDARD** A leaf module (cell) that is placed according to the row layout style; i.e., a standard cell.

**PAD** A chip (I/O) pad.

**GENERAL** A leaf module that is placed according to the general layout style; i.e., a macro cell.

**PARENT** A soft module: one that contains a group of cells.

Each module also has an IOLIST, which describes the external net connections of the module. For each connection, the *signal name* identifies the net; each net has a unique signal name. The *terminal type* of the connection is one of the following:

**I** means that the connection is an input to the module.

**O** means that the connection is an output from the module.

MODULE < *modulename* >;
TYPE < *moduletype* >;
DIMENSIONS < $X1$ >< $Y1$ >< $X2$ >< $Y2$ > . . . < $XN$ >< $YN$ >
IOLIST;
< *signalname* >< *terminaltype* >
[< *Xposition* > | < *side* >
< *Yposition* > | < *position* > [< *width* >< *layer* >]]
[*CURRENT* < *current* >][*VOLTAGE* < *voltage* >];
.
.
.
ENDIOLIST;
NETWORK;
< *instancename* >< *modulename* >< *signalname* > . . . ;
.
.
.
ENDNETWORK;
ENDMODULE;

Figure 1: YAL template.

**B** means that the connection is bidirectional: input and/or output.

**PI** is a pad terminal input.

**PO** is a pad terminal output.

**PB** is a bidirectional pad terminal.

**F** is a feedthrough signal. This signal is not used by the module, but the connection is used to pass the signal through the module.

**PWR** is a power connection (VDD).

**GND** is a ground connection (VSS).

Each module description may contain additional information, depending on its classification. A module is classified as *hard* if its characteristics (e.g., dimensions, terminal locations) are defined. A module with undefined characteristics is *soft*.

# 1 Hard Modules (STANDARD, GENERAL, PAD)

The GENERAL and STANDARD type of modules are hard, so they have defined characteristics. For mixed macro and standard cell netlists, a macro cell is

a GENERAL type module and a STANDARD cell is a standard type module. Chip pads are represented as hard modules. The DIMENSIONS line gives the (x,y) coordinates of the corners of the module in counter-clockwise order. Each terminal of the IOLIST must contain an Xposition, Yposition, width, and layer. The units are in microns ($\mu m$). *Xposition* and *Yposition* are the (x,y) coordinates of the center of the terminal with respect to the module origin, *width* specifies the terminal width, and *layer* specifies the conductor layer (PDIFF, NDIFF, POLY, METAL1, METAL2).

## 2 Soft Modules (PARENT)

Parent modules are soft, so their dimensions and terminal locations are undefined. Since current netlists are flat, each netlist contains *one* parent module that contains all the cells of the chip. Each terminal of the IOLIST represents a connection to a chip pad, and may contain a *side* (TOP, BOTTOM, RIGHT, LEFT) and/or a *position* if the pad locations are pre-defined. A parent module contains a NETWORK that defines the sub-modules and interconnections to be placed and routed. The sub-modules are all pre-defined leaf cells. Each sub-module has a unique *module name*. The *instance name* of a sub-module matches the module name of a pre-defined hard module in the netlist. Each sub-module contains a sequence of signal names; these must correspond to matching signal names in the pre-defined hard module.