YAL DESCRIPTION

Date: March 11, 1987

Modified: To accomodate rectilinear cells, power and ground nets with current and voltage restrictions, and the specification of critical net lengths.

The new YAL is not compatible with the old.

Jonathan Rose, November 18, 1987

I. General Comments

Comments are delimited by "/* " and " */". Comments MAY extend across physical line boundaries.

White space is the delimiter between tokens. White space is one or more of any combination of space, tab, line feed, or carriage return.

The data is free format. A semicolon, ";", is the logical line terminator. Any number (0 or more) of white space characters may separate a semicolon. A logical line can occupy more than one physical line (characters separated by carriage returns). Any number of logical lines can occupy a physical line.

"name"s are used for modules (cell definitions), cell instances (instatiations of modules), and signals (or nets). "name"s may contain any character except white space and semicolon. "name"s need not start with a letter. (e.g., "2" is a valid name). Case is significant; the name "A1" is different from the name "a1".

In the following descriptions, "|" means OR: choose one item from the list separated by the "|". Items between square brackets, "[" and "]", indicate optional fields on a logical line. Ellipsis, ". . .", (either horizontal or vertical) indicate that the preceding field or line should be repeated as necessary. Words between angle brackets, "<" and ">", are used as intermediate (non-terminal) definitions in the language. WORDS in upper case are keywords of the language.

"number"s are reals and are specified in microns.

Design rules are NOT specified in this language. Look in the text of circuit descriptions or cell libraries for design rules.

II. Module Definition

A module is a definition of a circuit being laid out or a constituent (or primitive) cell. The definition for each circuit or primitive cell begins with "MODULE <modulename>". The definition ends with "ENDMODULE". A template of a module follows.

```
        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;
                PLACEMENT
                        <instancename> <xlocation> <ylocation>
 [<reflection>] [<rotation>];
                        .
                        .
                        .
                        ENDPLACEMENT;
                CRITICALNETS;

                    <signalname>  <maximumlength>;
                                .
                                .
                                .
                        ENDCRITICALNETS;
                ENDMODULE;

        <modulename> ::= name
        <moduletype> ::= STANDARD | PAD | GENERAL | PARENT | FEEDTHROUGH
        <width> ::= number
        <height> ::= number
        <signalname> ::= name
        <terminaltype> ::= I | O | B | PI | PO | PB | F | PWR |  GND
        <side> ::= BOTTOM | RIGHT | TOP | LEFT
        <layer> ::= PDIFF | NDIFF | POLY | METAL1 | METAL2
        <xposition> ::= number
        <yposition> ::= number
        <position>  ::= number
        <instancename> ::= name
        <xlocation> ::= number
        <ylocation> ::= number
        <Xi> ::= number
        <Yi> ::= number
        <current> ::= number
        <voltage> ::= number
        <maximumlength> ::= number
        <reflection> ::= RFLNONE | RFLY
        <rotation> ::= ROT0 | ROT90 | ROT180 | ROT270
```

The TYPE line is required for all modules. The allowable types are:

STANDARD: an internal, primitive cell that must be placed on a row or column.
PAD: a driver (pad) cell that should be placed around the periphery of the chip.
GENERAL: an internal, primitive general cell or building block.
PARENT: a higher level module to be laid out.
FEEDTHROUGH: a cell used to connect a signal across a row.

Whether or not a line (e.g., DIMENSIONS) or a section (NETWORK or PLACEMENT) within a module is required depends on the TYPE of module being defined. A TYPE and an IOLIST are always required. A primitive cell (one to be used as a constituent of a layout: TYPE = STANDARD, PAD, GENERAL, or FEEDTHROUGH) is assumed to have been laid out previously and therefore has the following data: DIMENSIONS and IOLIST; <xposition> or <side>, <yposition> or <position>, <width> and <layer> are required for each of the IO <signalname>s within the IOLIST for primitive cells. NETWORK and PLACEMENT sections are ignored for primitive cells.

A feedthrough cell (TYPE = FEEDTHROUGH) should not appear in a NETWORK but is inserted into a layout by the layout system as necessary.

A module that defines a circuit to be laid out (TYPE = PARENT) must have an IOLIST and a NETWORK section. DIMENSIONS are ignored.

The DIMENSIONS line gives the co-ordinates of the the corners of the rectilinear box, in counter-clockwise order.

The IOLIST section defines the external connections or terminals of the module; this section should be included in definitions of all modules. Each line in this section defines a segment of connection to a higher level of hierarchy for the module. Each line must contain at least a <signalname> of an IO terminal and its <terminaltype>. Within a module being laid out (TYPE = PARENT) a <signalname> on a terminal line in the IOLIST is in the same name space as the <signalname>s in the NETWORK of that same <modulename>. Each module has its own separate name space. Other fields may be required depending on the terminal and the module that contains it as described below.

If a primitive module (TYPE = STANDARD, PAD, GENERAL, or FEEDTHROUGH) is being defined, the lines within an IOLIST must contain the <xposition>, <yposition>, <width>, and <layer>. Terminals appear in the order in which they will be referenced in the NETWORK section of any higher level module. Electrically equivalent terminals (e.g., in a dual- ported primitive module) must have the same <signalname>s. (In this case binding between the terminal <signalname>s of a primitive module and the <signalname>s of the higher level module are performed in the order of unique signal names. Refer to the example in Section 3.) It should be assumed that electrically equivalent terminals can be used as "feedthroughs" for the signals connected to them. If additional over-cell routing is allowed, i.e., for signals not connected to the cell, terminal locations for these feedthoughs will appear at the end of the IO terminal list. Valid <terminaltype>s are

```
                "I" for input
                "O" for output
                "B" for bidirectional
                "PI", "PO", "PB" for a pad terminal on a primitive,
                        where I, O, B are input, output, and
                        birectional, as above
                "F" for an independent feedthrough location
                "PWR" for VDD power input
                "GND" for VSS ground input
```

Terminal locations are specified by <xposition> or <side>, <yposition> or <position>, <width> and <layer>, or LEFT. <xposition> and <yposition> give the coordinate of the center of the terminal in microns on the edge of the cell. To be compatible with the old YAL, terminals can also be specified as <side> and position. <side> may be BOTTOM, RIGHT, TOP or LEFT. <position> gives the coordinate of the center of the terminal in microns along the indicated <side> from the left or bottom of the module as appropriate. <width> specifies the dimension of the terminal along the indicated edge; it extends from <xposition> - <width>/2 to <xposition> + <width>/2 or <yposition> - <width>/2 to <yposition> + <width>/2 as appropriate. <layer> specifies the conductor layer of the terminal.

Fixed pad placement for higher level circuits (TYPE = PARENT) requires a slightly different interpretation of the fields and is specified in the following manner. In this case, the <side> and <position> specifiers must be used, rather than the <xposition> <yposition>. If the terminal is bound to the pad of an IO driver, then the <side> refers to the side of the chip and <position> refers to the position of the origin of that driver cell.

<width> and <layer> are ignored. If no <position> is specified, then the pad is restricted to the indicated <side> but not to a <position> on the <side>. If no <side> is specified for a terminal bound to an IO driver then that IO driver may be placed on any <side>.

The NETWORK section defines the internal connectivity for the module. If a NETWORK section is included in the definition of a primitive module (TYPE = STANDARD, PAD, GENERAL, or FEEDTHROUGH), it will be ignored. Each line defines an instance of a primitive module that is included in the circuit. Each instance definition has the following fields:

<instancename> is the name of the instance of a module. An instance has signal bindings (specified in this section) and a location and orientation to be determined by the layout system (specified in the PLACEMENT section).

<modulename> is the name of the module to which this instance is bound.

<signalname> . . . is a list of <signalname>s connected to the module, in the order determined by the module definition. For example, the nth <signalname> is bound to the nth UNIQUE terminal <signalname> of the primitive module definition. Connections to the feedthrough terminals are not included in this list; they should be determined by the layout system. Single terminal <signalname>s are used to specify an unconnected terminal. An unconnected terminal may also result if the list of <signalname>s on an instance definition is shorter than the list of unique <signalname>s in the IOLIST of the primitive module being bound.

The PLACEMENT section is used to report placement results, one logical line per instance. If a PLACEMENT section is included in the definition of a primitive module (TYPE = STANDARD, PAD, GENERAL, or FEEDTHROUGH), it will be ignored.

<instancename> is the cell instance, and must match an <instancename> in the NETWORK section. <xlocation> and <ylocation> specify the location of the (0, 0) point (in microns) of the module after any orientation change of the instance. <reflection> and <rotation> are optional; the default for <reflection> is RFLNONE (the normal or "defined" reflection); RFLY means reflected about the Y axis. The default for <rotation> is ROT0 (no rotation); ROT90, ROT180, ROT270 indicate a counterclockwise rotation for the appropriate number of degrees. Reflection is applied before rotation.

The CRITICALNETS section is used to specify which nets are performance-critical, and their maximum length. <signalname> is the critical net, and must match a <signalname> in the NETWORK section. <maximumlength> is a number in microns specifing how long the signal wire can be restricted to. Any placement program interested in performance optimization tries to meet this limit.

III. Example 1: A Simple, Primitive Cell

This is a simple standard cell inverter. It has 1 input and 1 output. It is constructed on a 10-microns routing grid.

```
/*                                                         */
/* Module name = INV                                       */
/* Width = 20 microns, Height = 160 microns                */
/*                                                         */
MODULE INV;
 TYPE STANDARD;
 DIMENSIONS 0 0 20 0 20 160 0 160;
/*                                                         */
/* Define the external connections: I1 and O1              */
/*                                                         */
 IOLIST;
/* <signalname><terminaltype><xposition> <yposition><width><layer> */
  I1            I             5.0       160.0      3.0    METAL2;
  I1            I             5.0         0.0      3.0    METAL2;
  O1            O            15.0       160.0      3.0    METAL2;
  O1            O            15.0         0.0      3.0    METAL2;
 ENDIOLIST;
 ENDMODULE;


                 I1         O1
           ------x----------x------
           |    ---        ---    |
           |     |          |     |
           |     |          |     |
           |     |          |     |
           |     |          |     | 160 high
           |     |          |     | x 20 wide
           |     |          |     |
           |     |          |     |
           |     |          |     |
           |    ---        ---    |
           ------x----------x------
                 I1         O1

              3 wide     3 wide
              at 5.0     at 15.0
              on METAL2  on METAL2
```

This MODULE would be referenced in a NETLIST section as follows:

```
/*  <instancename> <modulename><signalname><signalname> */
  CELLX          INV         SIGIN       SIGOUT;
```

Terminal I1 on <modulename> INV is bound to <signalname> SIGIN
and terminal O1 is bound to <signalname> SIGOUT.

IV. Example 2: A Feedthrough Cell

This is a feedthrough standard cell. It provides a path from one
routing channel to another. It is constructed for a 10-microns
routing grid. This MODULE should not be referenced in a NETLIST
section.

```
/*                                                         */
/* Module name = FEEDER                                    */
/* Width = 10 microns, Height = 160 microns                */
/*                                                         */
MODULE FEEDER;
 TYPE FEEDTHROUGH;
 DIMENSIONS 0 0 10 0 10 160 0 160;
/*                                                         */
/* Define the external connection: F1                      */
/*                                                         */
 IOLIST;
/* <signalname><terminaltype><xposition> <yposition><width><layer> */
  F1           F             5.0          160.0      3.0    METAL2;
  F1           F             5.0            0.0      3.0    METAL2;
 ENDIOLIST;
 ENDMODULE;


                 F1
           -------------
           |    ---    |
           |     |     |
           |     |     |
           |     |     |
           |     |     | 160 high
           |     |     | x 10 wide
           |     |     |
           |     |     |
           |     |     |
           |    ---    |
           -------------
                 F1

              3 wide
              at 5
              on METAL2
```

V. Example 3: A Simple Chip.

This module has two internal cells and three pads.   No IO pre-
placement is specified.   One internal cell is specified above
(the INV cell); the other cell and the pads are defined here.

```
/*                                                         */
/* Module name = NAND                                      */
/* Width = 30 microns, Height = 160 microns                */
/*                                                         */
 MODULE NAND;
 TYPE STANDARD;
 DIMENSIONS 0 0 30 0 30 160 0 160;
/*                                                         */
/* Define the external connections: I1, I2 and OUT         */
/*                                                         */
 IOLIST;
/* <signalname><terminaltype><xposition> <yposition><width><layer> */
  I1          I             5.0           160.0      3.0    METAL2;
  I1          I             5.0             0.0      3.0    METAL2;
  I2          I             15.0          160.0      3.0    METAL2;
  I2          I             15.0            0.0      3.0    METAL2;
  OUT         O             25.0          160.0      3.0    METAL2;
  OUT         O             25.0            0.0      3.0    METAL2;
 ENDIOLIST;
 ENDMODULE;

/*                                                         */
/*            I1         I2        OUT                     */
/*      ------x----------x----------x------                */
/*      |    ---        ---        ---    |                */
/*      |     |          |          |     |                */
/*      |     |          |          |     |                */
/*      |     |          |          |     |                */
/*      |     |          |          |     | 160 high       */
/*      |     |          |          |     | x 30 wide      */
/*      |     |          |          |     |                */
/*      |     |          |          |     |                */
/*      |     |          |          |     |                */
/*      |    ---        ---        ---    |                */
/*      ------x----------x----------x------                */
/*            I1         O1         OUT                    */
/*                                                         */
/*         3 wide     3 wide     3 wide                    */
/*         at 5       at 15      at 25                     */
/*         on METAL2  on METAL2  on METAL2                 */
/*                                                         */
/* Module name = INPUTPAD                                  */
/* Width = 200 microns, Height = 200 microns               */
/*                                                         */
MODULE INPUTPAD;
 TYPE PAD;
 DIMENSIONS 0 0 200 0 200 200 0 200;
/*                                                         */
/* Define the external connections: OUT and INPAD          */
/*                                                         */
```

```
 IOLIST;
/* <signalname><terminaltype><xposition> <yposition><width><layer> */
  OUT         O             100.0          0.0       3.0    METAL2;
  INPAD       PI            100.0        200.0       50.0   METAL2;
 ENDIOLIST;
 ENDMODULE;
/*                                                           */
/*                     INPAD                                 */
/*      -----------------x-----------------                  */
/*      |           -----------           |                  */
/*      |           |         |           |                  */
/*      |           |         |           |                  */
/*      |           |         |           |                  */
/*      |           |         |           | 200 high         */
/*      |           |         |           | x 200 wide       */
/*      |           -----------           |                  */
/*      |                |                |                  */
/*      |                |                |                  */
/*      |                |                |                  */
/*      |                |                |                  */
/*      |                |                |                  */
/*      |                |                |                  */
/*      |                |                |                  */
/*      |                |                |                  */
/*      -----------------x-----------------                  */
/*                      OUT                                  */
/*                                                           */
/*                    3 wide                                 */
/*                    at 100.0                               */
/*                    on METAL2                              */
/*                                                           */
/*                                                           */
/* Module name = OUTPUTPAD                                   */
/* Width = 200 microns, Height = 200 microns                 */
/*                                                           */
MODULE OUTPUTPAD;
 TYPE PAD;
 DIMENSIONS 0 0 200 0 200 200 0 200;
/*                                                           */
/* Define the external connections: IN and OUTPAD            */
/*                                                           */

IOLIST;

/* <signalname><terminaltype><xposition> <yposition><width><layer> */
  IN          I             100.0          0.0      3.0    METAL2;
  OUTPAD      PO            100.0        200.0      50.0   METAL2;
 ENDIOLIST;
 ENDMODULE;
/*                                                           */
/*                    OUTPAD                                 */
/*      ----------------x------------------                  */
/*      |           -----------           |                  */
/*      |           |         |           |                  */
/*      |           |         |           |                  */
/*      |           |         |           |                  */
/*      |           |         |           | 200 high         */
/*      |           |         |           | x 200 wide       */
/*      |           -----------           |                  */
/*      |                |                |                  */
/*      |                |                |                  */
/*      |                |                |                  */
/*      |                |                |                  */
/*      |                |                |                  */
/*      |                |                |                  */
/*      |                |                |                  */
/*      |                |                |                  */
/*      -----------------x-----------------                  */
/*                      IN                                   */
/*                                                          */
/*                    3 wide                                */
/*                    at 100.0                              */
/*                    on METAL2                             */
/*                                                          */
/* Module to be laid out: name = AND                        */
/*                                                          */
MODULE AND;
 TYPE PARENT;
/*                                                          */
/* Define the external connections: I1, I2 and OUT          */
/* Pad positions are not fixed                              */
/*                                                          */
 IOLIST;
/* <signalname> <terminaltype><side><position><width><layer> */
  I1           PI             LEFT    1.0;
  I2           PI             RIGHT   2.0;
  OUT          PO             LEFT    3.0;
 ENDIOLIST;
/*                                                          */
/* Define the instances of primitive modules and the        */
/*   connectivity                                           */
/*                                                          */
 NETWORK;
/* <instancename> <modulename> <signalname> <signalname>    */
  I1             INPUTPAD     i1;
  I2             INPUTPAD     i2;
  Out            OUTPUTPAD    out;
  Inv            INV          nout          out;
  Nand           NAND         i1            i2          nout;
 ENDNETWORK;
 ENDMODULE;
```

VI. Example 4: A General Cell Example

```
MODULE A;
 TYPE GENERAL;
 DIMENSIONS 0 0 30 0 30 50 0 50;
 IOLIST;
  P_0 PWR 10 0 1 METAL2 CURRENT 0.010 VOLTAGE 0.080;
  P_1 B 30 10 1 METAL2;
  P_2 B 30 20 1 METAL2;
  P_3 B 30 30 1 METAL2;
  P_4 B 30 40 1 METAL2;
  P_5 PWR 20 50 1 METAL2 CURRENT 0.010 VOLTAGE 0.080;
  P_6 B 0 10 1 METAL2;
 ENDIOLIST;
ENDMODULE;
MODULE B;
 TYPE GENERAL;
 DIMENSIONS 0 0 30 0 30 70 0 70;
 IOLIST;
  P_0 PWR 10 0 1 METAL2 CURRENT 0.010 VOLTAGE 0.050;
  P_1 B 30 30 1 METAL2;
  P_2 PWR 20 70 1 METAL2 CURRENT 0.010 VOLTAGE 0.050;
  P_3 B 0 60 1 METAL2;
  P_4 B 0 50 1 METAL2;
  P_5 B 0 30 1 METAL2;
  P_6 B 0 20 1 METAL2;
 ENDIOLIST;
ENDMODULE;
MODULE bound;
 TYPE PARENT;
 DIMENSIONS 0 0 125 0 125 110 0 110;
 IOLIST;
  P PWR 55 0 1 METAL2 CURRENT 0.030 VOLTAGE100.000;
  P PWR 0 60 1 METAL2 CURRENT 0.030 VOLTAGE 100.000;
  G PWR 125 70 1 METAL2 CURRENT 0.030 VOLTAGE 100.000;
  G PWR 0 70 1 METAL2 CURRENT 0.030 VOLTAGE 100.000;
```

```
   ENDIOLIST;
   NETWORK;
    C_0 A P 2 3 4 5 G 1;
    C_1 B P 1 G 4 5 3 2;
   ENDNETWORK;
  ENDMODULE;
```