COMPUTER SYSTEMS LABORATORY

DEPARTMENTS OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
STANFORD UNIVERSITY · STANFORD, CA 94305

YALE User's Guide: A SILT-Based VLSI Layout Editor

Tom Davis and Jim Clark

Technical Report No.. 233

. October **1982**

This research was supported by the Defense Advanced Research Projects Agency under contract # MDA903-79-C-0680.

YALE User's Guide:

A SILT-Based VLSI Layout Editor

Tom Davis and Jim Clark

Technical Report No. 233

October 1982

Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305

Abstract

YALE is a layout editor which runs on SUN workstations, and deals with ceils expressed in the SILT language. It provides graphical hooks into many features describable in SILT. YALE runs under the V Kernel, and makes use of a window manager that provides a multiple viewport capability.

Key Words and Phrases: VLSI, design tools, layout editors, relative geometry

Table of Contents

1. Introduction 1

1.1 System Overview 1
1.2 Starting Up 2
1.3 Implementation 2

2. User Interface 3

2.1 Initial Viewports 3
2.2 The Mouse 3
2.3 Pop Up Menus 4
2.4 Command Feedback 5
2.5 Typing in Information 5
2.6 The YALE Coordinate System 6

3. The Window Manager 7

3.1 Displays and Viewports 7
3.2 Entering and Exiting the Window Manager 8
3.3 Zooming In and Out 8
3.4 The Main Window Menu 9

4. Editing with YALE 11

4.1 Yale Overview 11
4.2 A Typical Editing Session 12
4.3 The Status Viewport 14

5. Adding Rectangles and Selection 17

5.1 Add Rectangle 17
5.2 Select Item 17

6. Create commands 19

6.1 Create Cell Definition 19
6.2 Create Cell Instance 19
6.3 Create Copy of Cell Definition 20
6.4 Create Array 21
6.5 Create Reference Point 21

7. Setting Defaults 23

7.1 Setting Default Rectangle Widths 23
7.2 Setting Default Expansion Depths 23

8. Select, Delete, and Move Commands 25

8.1 Selection 25
8.2 Deletion 26
8.3 Moving 26

9. Input/Output 29

9.1 Initialization 30
9.2 Close Connection 30
9.3 Read Library 30

9.4 User Name 30
9.5 Write 31
9.6 Write Backup 31

10. Miscellaneous YALE commands 33

10.1 Expand Cell 33
10.2 Re-reference 33
10.3 Window Manager 33
10.4 Show Cell Definition Names 34
10.5 Quit 34

11. **Errors** 35

11.1 The '%' Command 35
11.2 SILT Parsing Errors 35
11.3 Running out of space 35
11.4 Other Size Limitations 36
11.5 Bugs 36

I. Window Manager Command Outline 37

II. YALE Editor Commands 39

1. Introduction

The YALE (Yet Another Layout Editor) layout editor makes hierarchical cell layouts on a SUN workstation. It is meant to be used with the SILT translator. All the files produced and read by YALE are written in the SILT language (see [3]). The SILT program translates these into a CIF format for use with other programs such as design rule checkers, circuit simulators, and mask-making software.

1 .1 System Overview

The YALE layout editor runs under the V Kernel (see [2]) on a SUN workstation. The V Kernel is a message-based kernel supporting multiple processes of which YALE might be only one. Since the current implementation of YALE is large, it is unlikely that too much else will be running on the workstation at the same time as YALE.

To run YALE with any reasonably large layout at all requires more than the minimal 256K SUN configuration. The more memory is available on the SUN workstation, the better YALE will work.

Even if YALE is the only program running under the V Kernel, there are two processes with which you, as the user, must be familiar. One is the layout editor itself, and the other is the window manager. YALE itself has no idea where its viewports are presented on the screen, or even how many viewports there are. You are free to create more viewports opening on different parts of the cell being currently edited, and to move these viewports around and to adjust their sizes and magnifications. There is nothing special about YALE in this respect. Any process using the above-mentioned window manager can do the same thing.

At any time during a session, all the keystrokes and mouse-clicks are directed either to the window manager or to the YALE editor (assuming these are the only two processes running). Obviously, the input is interpreted differently by the two processes, so if something surprising happens, make sure that your input is going to the process you think it is.

It is easy to tell at a glance whether you are typing to (or “mousing” to) YALE or to the window manager. The shape of the cursor changes. In both cases, the cursor is an upward-pointing arrow, but for the window manager, it is shorter, and its lower half forms the letter "w".

There are really more than two processes running during a YALE session, but the editor and the window manager are the only two with which you need to deal. Other processes include one process that is busy watching the keyboard for input, and another that always watches the mouse.

1.2 Starting Up

To load YALE, you must use the V Kernel loader, called “Vload”. This is done by typing "n Vload” (note the capitalization here) to the monitor of the SUN workstation. After a while, this program will respond with something like:

```
V Kernel Loader - Version 3.1 - 28 June 1982

Program name :
```

Type in “Yale” (again, note the capitalization), and press <CR> (<CR> stands for the carriage return key). The system will respond by typing a few lines of exclamation marks, and finally with:

Type "c<CR>" ("c" stands for continue -- the reason that this step is necessary is that breakpoints could be put in here), and the program will start.

Three initial viewports will be painted on the screen whose functions are described in the next chapter, and you will be prompted for some startup information in the tty viewport (which is positioned initially in the lower-left hand corner of the screen).

1.3 Implementation

YALE is implemented entirely in the "C" programming language, and most of the internals of the YALE editor (almost everything save the display code) were thoroughly tested on a VAX before being ported to the SUN workstation. Since most of it has already been ported once, it should not be too difficult to port it again. All that needs to be written is another set of display routines.

Although YALE runs under the V Kernel, it does not make heavy use of its services. The V Kernel provides a process to track the cursor, and to read the mouse and keyboard. The window manager runs as a separate process, but this is not necessary, and in the initial implementation, it was part of the YALE editor.

A reasonably clean separation of the editor part from the window manager part has been made to simplify some experiments using YALE in a distributed mode -- YALE will run on a VAX, and the window manager and a display list interpreter will run on the SUN workstation.

In addition to the V Kernel, YALE makes use of the SUN rasterop (see [1]) package for the display of rectangles and stippfes on the screen, and of the leaf package (see [4]) for remote file access.

2. User Interface

This chapter describes the general features of the user interface for both YALE and the window manager. An attempt has been made to make the command interaction similar for both programs. Most of the conventions described below apply both to the layout editor and to the window manager.

2.1 Initial Viewports

When YALE starts up, it displays three viewports. One of these is called the **tty viewport,** and is used primarily for command feedback, error messages, and user type-in. It behaves exactly like **a** glass teletype in that each line is typed at the bottom, and lines typed earlier are scrolled **up.** Although the initial viewport is small, the tty window keeps 24 lines of text, and earlier commands can be examined by calling on the window manager and enlarging the viewport.

The second viewport is called the **status viewport,** and contains some YALE status information. Such things as the current file, the name of the cell being edited, x- and y- coordinates of the last mouse click, the currently selected layer, and the default widths of the layers are presented. The information here will be covered in more detail later. See.

The third viewport is the main YALE **graphics viewport.** This is where the currently open cell is presented. Most other YALE viewports will be graphics viewports, but views opening on different portions of the cell being edited.

2.2 The Mouse

YALE and the window manager both receive most of their commands from the mouse. When the mouse is held with its three buttons on top, the left-most mouse button is called number 1, the center is number 2, and the right-most button is numbered 3. In some of the prompting that appears on the screen, and in the documentation that follows, they are referred to as MB1 , MB2, and MB3.

Some YALE commands require that more than one mouse button be pressed at the same time. Since it is impossible to press the buttons at exactly the same time, the mouse input is interpreted by YALE as follows: The interpretation begins when the transition is made from all buttons up to at least one button down. It ends when all buttons are up again. All buttons depressed in the interim are recorded as part of the mouse event. Thus, as long as you have not released all the buttons, you can always press another button. Because of this, if the mouse event is not yet complete, it is always possible to press down all three buttons, and YALE ‘takes advantage of this by defining a three-button push as aborting the command. If you accidentally press the wrong button and notice it before it is released, simply press down the other two buttons and then release all three, and there will be no net effect.

In both YALE and the window manager, the general philosophy is to bind the most useful commands to the left and center mouse buttons (MB1 and MB2). All other commands are accessed by one or more pop up menus, described in the next section. For both programs, the pop up menu containing the rest of the commands is accessed by pressing and releasing the third mouse button.

The action of pointing to an object or screen position with the cursor, and choosing it with a mouse button click is often referred to here as “bugging”. One can thus “bug a rectangle” to select it, or identify a viewport to move by “bugging it”.

2.3 Pop Up Menus

Since there are only 7 mouse button combinations, even if they all were to be used, there are too many YALE commands to go around. The same thing is true of the window manager, so only the most useful commands are bound directly to mouse clicks, and the rest of the commands are invoked using pop up menus. For both the window manager and for general YALE layout editor commands, the pop up menu is gotten by using MB3 by itself. A menu containing a variable number of items appears under the cursor at that point. To select a command presented there, move the mouse until the tip of the cursor inside its box, and press any button. (In other words, “bug” the correct menu entry.) Sometimes menus are two-level, so a second menu will appear for the sub-command. If you invoke a menu by accident and do not really wish to select any of the commands in the menu, simply move the cursor completely outside the menu and press any button.

Important! To get a pop up menu for YALE, you must press the third button while the cursor is in one of YALE’s windows. In this way, it is possible to use the window manager with more than one process. If the third mouse button is pressed while it is outside any window, it will have no effect. On the other hand, when your input is directed to the window manager, the main pop up menu (for the window manager) can be pressed at any time.

2.4 Command Feedback

As each command is issued, whether it is clicked in with the mouse buttons, or accessed through a series of one or more pop up menus, an English sentence is gradually built up in the tty viewport both to show what command is being specified, and, where possible, the next input required of the user. If there is ever any confusion about which command is being specified, the last line in the tty viewport shows what is going on and what is expected next.

There are a few abbreviations that are commonly used in this feedback in addition to MB1, MB2, and MB3 for the mouse buttons. These include:

Abbreviation	**Meaning**
T:	Type in textual data, followed by <CR>.
B:	“Bug” an object or position on the screen by pointing to it with the cursor and pressing (usually) MB1.
M:	A menu selection of some sort is to be made. Move the cursor so that its tip is in the correct menu entry, and press a mouse button. Some menus are special, and have "typein" entry at the bottom. If this entry is selected, you will be asked to type in the correct information, followed by a <CR>. This mechanism is used when there is a set of common choices, but where you may wish to use some unusual choice from time to time.

2.5 Typing in Information

From time to time, certain of the commands require that you type in some textual information -- the name of a new cell definition, the name of a reference point, or the name of a file to be used for Input or output. When this happens, a prompt appears in the command feedback viewport, and you simply type in the text, followed by a carriage return. For new names, YALE obviously has no choice but to ask you directly for the information.

On the other hand, since the most commonly typed text will be the names of cell definitions that are already known (such as “expand cell named . ..", “create instance of cell named . .."), YALE keeps a list of a few of the most recently referenced cell names. Thus, when you need to specify a cell name, YALE puts up a Pop Up menu containing these most recently referenced cells. The last item in the menu is always “Typein”, and if that entry is selected, you will be requested to type in the name of the definition as you would for any other text string.

YALE automatically converts all typed input and all input from files to lower case (except, of course, for UNIX file names). You may type your names in using any case you wish, but YALE will print them back to you in lower case only.

Usually, a mechanism analogous to the three-mouse-button abort exists for textual input. When YALE requests type-in, it is not looking at the mouse, so it cannot interpret a three-button abort. When a name of some sort is required and you have committed to a typein, typing a <CR> with no text aborts the command. This mechanism is not so universal as is the one using three mouse buttons.

2.6 The YALE Coordinate System

YALE (and SILT) work in a standard mathematical right-handed Cartesian x-y coordinate system. SILT allows arbitrary real coordinates, but YALE restricts the coordinates to be integers or half-integers. When selecting a point on the screen, the nearest point with half-integer coordinates is chosen.

The grid marks that can appear on the screen are always placed at distances that are equal of a power of 2 times lambda. The exponent can be negative, however.

3. The Window Manager

3.1 Displays and Viewports

During any session, there may be many distinct “universes” whose contents you may wish to view. When YALE starts up, there are three -- the text shown in the tty viewport, the information shown in the YALE status viewport, and the geometry of the cell being edited in the graphics viewport. If the terminal is being used for other user processes, additional viewports may be required.

Each of these “universes” is called a **display.** A process may have zero or more displays associated with it. (In particular, the mouse and keyboard watcher processes have zero.) In addition, it may be useful to have more than one view of any given display. For VLSI editing, one ofter wants to have one large-scale view of the cell being edited, and another zoomed-in view of the small region being actively changed. For complex, long-distance wiring, it may be useful to have views of each of the areas on the chip where the wires bend.

It is sometimes useful to make a distinction among the terms “display”, “viewport”, and “window”, especially between the latter two. The display contains all the objects in the universe of interest, whether they are shown on the screen or not. The viewport is the physical area on the display screen upon which objects are displayed, and the window contains the same information as the viewport, but is described in terms of world (or YALE) coordinates instead of in screen coordinates. Thus, one might move a viewport on the display screen, and one would center a window coordinate in the current viewport.

Each individual view of a display is called a **viewport,** and occupies some physical space on the screen. The window manager can change both the part of the display being viewed, and that view’s magnification.

The window manager can handle up to 8 different viewports. All must be rectangular and non-overlapping. It is possible to shrink little-used viewports, so it is not impossible to make use of all 8 viewports occasionally. It can also be used to create more or fewer viewports associated with the same display. Usually, however, 4 or 5 viewports should be all that are required. If the window manager is someday modified so that it can handle overlapping viewports, all this may change.

3.2 Entering and Exiting the Window Manager

When YALE starts running, all input is directed to the YALE layout editor. There are two ways to get the attention of the window manager, and to begin directing input to it.

The easiest method is via a selection from YALE's main pop up menu. Press MB3, move the cursor to point at the entry entitled "Window Manager", and bug it. The cursor will change from a simple arrow to a shorter arrow with the character "w" underneath.

The second method is to type the so-called "brain escape character", which is initially set to be "↑C" (control-C). The cursor will immediately change to its window manager form. The brain escape character can be reset to something besides "↑C" using a command to the window manager (see 3.4).

The window manager is exited in only one way -- this is via a selection from the window manager's main menu (again accessed using MB3). The menu entry is labeled "Return to YALE".

3.3 Zooming In and Out

The most useful commands that can be issued to the window manager are for zooming in and out on the view. MB1 is bound to "Zoom In" and MB2 is bound to "Zoom Out". These commands affect only the viewport in which the cursor is displayed when the button is pressed. It is therefore possible to have different viewports displaying the cell being edited with different magnifications.

When MB1 ("Zoom In") is pressed in a viewport, the magnification in that viewport is doubled, and the point at the tip of the cursor arrow is appears in the center of the new screen. MB2 ("Zoom Out") has just the opposite effect -- the magnification factor is cut in half, and point at the tip of the cursor arrow again appears in the center of the new view.

Notice that one can change the view without changing magnification simply by zooming out, and then zooming in to a new center, and can thus be accomplished using two mouse clicks. If the cell being displayed is complicated, however, it is probably easier to use the "Center Window" command (described below), since the display is re-created after each zooming action.

In some viewports, zooming in and zooming out have no effect on the material displayed. This is usually the case where the viewport is displaying some textual material. Neither the tty viewport nor the status viewport are affected by these commands, but all the YALE graphics viewports are.

3.4 The Main Window Menu

The main window manager menu is accessed using MB3. Each of the possible menu commands is described in a paragraph below:

Create viewport. This command creates another viewport on the screen. The next two mouse clicks are interpreted as opposite corners for the new viewport, and may be given in any order. Finally, an existing viewport is bugged to show which display is to be painted in the newly created viewport. If an attempt is made to create a new viewport that overlaps existing viewports, the error message “illegal stretch” will appear in the tty viewport.

Move **Edge.** To change the shape of a viewport, bug the “Move Edge” entry in the main window manager menu, move the cursor to an edge or corner of a viewport, and press any mouse button. Then move the cursor to a new screen position, and press the button again (remember that if all three buttons are presed at any time, the command can be aborted with no effect). The viewport is redrawn with its corner or edge moved to the second mouse position. Again, since viewports are not allowed to overlap, a common error is “illegal stretch”.

Move Viewport. To move a viewport rigidly on the screen so that it remains the same size and shape, and continues to view the same display, bug the “Move viewport” command in the main window manager menu. Next, choose a point of reference in the viewport to be moved, move the cursor to that point, and press a mouse button. When the cursor is rnoved to **a** new position on the screen, and a mouse button is pressed, the point of reference in the old viewport is moved to that position. Usually this command is easiest to control if the reference point is selected near one of its corners. The window manager makes sure that the viewport in its new position does not lie outside the screen boundaries, and that it does not overlap existing viewports. If an attempt is made to move a viewport off the visible screen, its position is adjusted to put it entirely on-screen. If an attempt is made to overlap another viewport, an error message is printed in the tty viewport, and nothing happens.

Delete Viewport. This command deletes the viewport in which the cursor sits when the next mouse button is pressed. *Don’t delete the last viewport on any given display, or you may have a hard time getting back the view.* This mis-feature should be corrected someday.

Center Window. This command allows one to change the view so that a new point in the display is moved to the center of the viewport. Simply press a mouse button with the cursor in a viewport, and the viewport will be redrawn with that point moved to the center of the new view. It was pointed out earlier that exactly the same change can be affected using a “Zoom Out” followed by a “Zoom In” command.

Red raw. From time to time, YALE or the window manager gets mixed up about exactly what is on the screen. This command simply causes the contents of all viewports of be redrawn from scratch. When all the bugs are removed, this command will be, too.

Brain Escape. This command resets the window manager escape character (which was initially set to be ↑C). A new escape character is typed, followed by a carriage return. The only restriction is that the character can not be "%" (See 11.1), but it is probably a bad idea to use printing characters that you may wish to use for typing information to YALE. Since YALE has a built-in command for accessing the window manager, this command probably has low utility. Other programs that will eventually run under the V Kernel may have no idea that there is anything like a window manager, and it is for them that this facility was included.

Toggle Grid. To aid in editing, it is sometimes useful to have a grid presented on the screen. The grid points are always the same distance apart on the screen, so at different magnifications, they correspond to different lambda-measures. After selecting the “Toggle Grid” option, you must then show which viewport is affected with another mouse click. If the grid is currently displayed in that viewport, it is turned off, and vice-versa. The grid may be toggled in any viewport, but it is probably not too useful to display a grid in any but a graphical viewport.

Return to Yale. This command diverts all further type-in and mouse clicks back to the YALE layout editor.

4. Editing with YALE

The next few chapters deal with the YALE layout editor itself. This first chapter gives an overview of the editing system, and the others describe in detail the commands that can be issued.

4.1 Yale Overview

Before going into a description of the meaning of each of the YALE editing commands, a short description will be given of the various sorts of things that the YALE editor manipulates. What is presented in the next few paragraphs is condensed, and an understanding of the SILT language would be extremely useful. SILT is a powerful language for describing cell layout, and YALE deals only with a small subset of the available SILT commands.

Because of this, it may be necessary to use a combination of YALE and hand-editing of the SILT format files produced by YALE to deal with a complicated layout. All the basic cell layout can be done with YALE, together with some hierarchical depth. If it is necessary to make use of SILT's river-routing facilities, or of other higher-level features, they will have to be specified by hand, in SILT, using the text editor of your choice.

Each layout is defined in terms of a hierarchical set of syrnbol calls'[1], with one symbol designated as the master layout symbol. A general SILT file could have many different commands in the main file body, but YALE restricts it to having a single symbol call in the main file body.

Each symbol is made up of rectangles, reference points, and calls on other symbols. See the SILT documentation for a complete description of reference points.

The rectangles on the screen are represented with stipple patterns, the symbol calls are normally expanded, and the reference points are displayed as labeled horizontal and vertical lines. In addition to the user-defined reference points, another pair (the x- and y- origins) are also shown. In some ways, these origins behave exactly as the other reference points, and in other ways they do not. The main difference is that it is not possible to move the origin reference points. The origin of the cell is the point where the origin reference points cross.

[1] In this documentation. the term "symbol" and "cell" are used almost interchangeably. SILT uses the term "symbol", but the-term "cell" is also widely used.

A symbol instance is placed with its origin relative to a horizontal and a vertical reference point. A rectangle has its top and bottom edge relative to horizontal reference points and its left and right edges relative to vertical reference points. For a symbol or rectangle edge to be placed “relative to a reference point” means that it remains a fixed distance from that reference point. If the reference point is moved, then all objects placed relative to it are moved as well. If both of a rectangle’s vertical edges are relative to the same vertical reference point, and the reference point is moved, then the rectangle will be moved rigidly. If the edges are relative to different reference points, then the rectangle will stretch or shrink if one of its reference points is moved without moving the other.

In SILT, there is really nothing analogous to the origin reference points, but they are required for a reasonable graphical interface. The SILT code for a rectangle produced relative to the reference points named “xref” and "yref" would look something like this:

```
place box( xref + 3, yref - 6, metal) to (xref + 11, yref + 1);
```

while a rectangle placed relative to the origin reference points would generate the following SILT code:

```
place box(3, -6, metal) to (11, 1);
```

If “xref” and “yref” are moved so that they are coincident with the respective origins, the two lines of SILT code would represent the same box.

4.2 A Typical Editing Session

Before going into detail about the commands available in the YALE editor, it is useful to give an idea of how a typical editing session would be carried out. Let us assume that you wish to design a new cell, and already have available a small library of SILT cells in a file on your VAX.

After loading the V Kernel and the YALE program, some initialization is required. You will be asked for the name of the VAX with your files, your user name and password, and the name of the file that contains (or will contain) the layout produced. After YALE has verified that the VAX is up, that the user name and password are valid, and that the file exists (if the name is a new one, it is created), you are left in a state where the input goes to the layout editor.

From now on, the procedure is to open one of the symbols for editing, to make changes to it, and then to close it and open another symbol. Only the cell currently open can be modified. Modifications take the form of adding or deleting rectangles, sub cells, and reference points. If a cell is closed, then one of its subcells opened and modified, and the original cell opened again, all instances of the subcell in the original cell will show the modifications.

Cells are opened either with the “Expand Cell” command, or by creating them with the “Create Cell instance” command. Cells are closed simply by opening another cell. Having a cell open is not really analogous to having a file open on a computer -- open cells are not particularly vulnerable. The fact that a cell is open simply means that it is the one displayed, and to which modifications will be made. There are not too many YALE operations that cannot easily be undone, except possibly deleting an entire cell definition, and deleting all the selected items, when many of them are selected.

When a new symbol is created, it contains nothing but an x-origin and a y-origin. These behave somewhat as default reference points. At any point during the editing, exactly one horizontal and one vertical reference point is selected, and the selected reference point is shown by being displayed with a bolder line. In a newly created cell, the x- and y-origins are the selected reference points. Any geometry (rectangles and symbol instances) which is entered is placed “relative to” the currently selected reference points -- that is to say, if the reference point is moved, the objects placed relative to it will also be rigidly moved.

Rectangles that are to remain rigid when the reference points are moved are easily inserted -- just make sure that the appropriate vertical and horizontal reference points are selected before inserting the rectangle. If it is required that a rectangle stretch or shrink when some reference points are moved (i. e. one edge is relative to one reference point, and the other edge is to be relative to another), the usual procedure is to make sure that one of the reference points is selected when the rectangle is inserted, and then to re-reference the other edge to a different reference point.

It is easy to see if all the internal symbols and rectangles are placed relative to the correct reference points -- simply move the reference points around a little, and make sure that the components move and stretch as they should. Any parts that do not can then be re-referenced.

As the editing proceeds, it is usual to make backup copies of the work from time to time. The first time the “Write Backup” command is invoked, you will be asked for a file name and a name of the master symbol. (The master symbol is usually the one corresponding to the entire chip. If there is no master symbol, any symbol name will do.) After this, making a backup is simple -- just issue the “Write Backup” command, and the old backup file will be over-written with the new version of the layout.

At the end of an editing session, the “Write Main File” command is usually given, followed by the “Quit” command. YALE is aborted, and you are returned to the SUN monitor.

4.3 The Status Viewport

Throughout a YALE editing session, one viewport is devoted to presenting the status of the session. The meaning of some of the information in this window is obvious, such as the name of the main input file and the name of the cell currently open for editing.

The most important feature of the status viewport is the set of seven small stipple pattern samples. These are the stipple patterns for the various layers. Above each sample of the pattern is an abbreviation for the name of the layer: "metl", "poly", "diff", "impl", "burd", "glas", and "cont" standing for “metal”, “polysilicon”, “diffusion”, “implant”, “buried”, “glass”, and “contact cut”, respectively. Underneath each of these rectangles is a number indicating the default width of a wire made of this material. If the default width for the metal layer is 4, this means that whenever the metal layer is selected, all rectangles placed in the currently open cell will be made of metal, and will have width 4.

One of the seven patterns is selected (outlined), and this is the layer that will be used by the “Add Rectangle” command. To select another layer, simply move the cursor so that it points to a new stipple in the status viewport, and depress a mouse button. The newly selected stipple should be selected (outlined), and the previously selected layer should have its outline removed.

The default widths associated with each layer can also be changed. See 7.1, below.

Also in the status viewport are four entries labeled "x:", "y:", "dx:", and "dy:". Every time a mouse button is clicked in a YALE graphics viewport, these values are updated. The "x:" and "y:" values give the absolute position of the click relative to the origin of the cell; the "dx:" and "dy:" values give the displacement from the last "x:" and "y:" values. These entries can be used to measure distances within a cell, and to find out about where you are if the view is zoomed in so far that there are no reference points visible.

Finally, there is an entry for the currently active repeat command. A few of the common commands in YALE have the feature that although they must be specified using a series of menus originally, they may then be easily repeated. The name of the most recently issued command of this sort appears here.

To cause that command to repeat, simply depress the two left-most mouse buttons (MB1 and MB2). Exactly what happens next is slightly dependent on the repeated command. The three repeatable commands are “Delete Selections”, “Create Cell Instance”, and “Add to Selections”. See 8.2, 6.2, and 8.1, respectively for details on how the repeat command will behave in each circumstance.

5. Adding Rectangles and Selection

The addition of rectangles to a layout, and the selection of objects in a layout to modify are the two most common operations in YALE. Thus, each is tied to a single mouse click. The exact action of these two important commands is described in the two sections that follow.

5.1 Add Rectangle

At any point during the editing, there is a default layer for rectangles selected, and a default width for that layer. Placing rectangles on the screen involves marking the two endpoints of the wire using MB1. YALE figures out whether the selected points are more vertical or more horizontal and inserts an appropriate rectangle. The first point selected is guaranteed to be the center of one end of the wire. For example, suppose that MB1 is first pressed at the point having coordinates (15, 32), and is next pressed at (43, 35). The change in the x-direction (28) is much greater than the change in the y-direction (3), so the rectangle is assumed to be a horizontal one. Since MB1 was first pressed with a y-coordinate of 32, this will be the y-coordinate for the center of the inserted wire.

If MB1 is pressed accidentally, placement of the rectangle can be aborted by pressing any but MB1 (in particular, all three buttons can be pressed, causing the usual abort to occur).

When a rectangle is inserted, all currently selected items ‘are de-selected, and the newly inserted rectangle is selected. This makes it easy to move it to the correct position or to delete it if it was placed incorrectly.

The left and right edges of the rectangle are placed relative to the currently selected vertical reference point, and the top and bottom edges are placed relative to the selected horizontal reference point. If there is no ceil currently open, an error message will be presented in the tty viewport.

5.2 Select Item

To select an item, point the cursor at it, and press MB2. If the item is a symbol instance or reference point, it is simply selected. If it is a rectangle, things are a bit more complicated. If the cursor is in the center of the rectangle, the entire rectangle (all four edges) is selected. If the cursor is pointing to an edge, just that edge is selected. If the cursor is.pointing to a corner, then both of the edges adjacent to that corner are selected. In every case, all other currently selected items are de-selected before the selection takes place. A selected symbol instance is displayed with a bold outline, and a selected rectangle edge is likewise highlighted.

Since objects may overlap each other in an arbitrary way, there must be some mechanism for disambiguating a selection that could be interpreted in more than one way. The disambiguation algorithm is described below.

1. If there is only one object under the cursor, then that object is selected.

2. If there are no objects under the cursor, then YALE looks within a few pixels of the cursor for nearby objects. If there is exactly one nearby object, then that is selected.

3. If there is more than one object under the cursor, or if there are no objects under the cursor, but there is more than one nearby object, then the areas of all possible objects are compared, and the object with the smallest area is selected. Reference points are judged to have essentially infinite areas.

4. If there is still ambiguity (i. e. there are still two or more items with exactly the same area and under the cursor),then the object with the smallest height-to-width ratio is selected.

5. Finally, if there is more than one object with exactly the same height and width of smallest area under the cursor, a series of possibilities is presented in the tty viewport. A short description of the object is given, and it can be selected by typing "y<CR>". Typing "n<CR>" causes the description of the next possible object to be presented. As soon as the user responds positively, that item is selected, and the selection is finished. If "n<CR>" is typed in response to every option, then nothing is selected.

Note that it is still possible to construct an example where it is impossible to select a certain piece of geometry. In particular, it will happen when a large object is completely covered by smaller objects, as is the metal layer in a butting contact. If this happens, the only way out is to delete or move the object(s) causing the conflict, select and deal with the object of interest, and then recreate or move back the other objects. This should not happen often.

Exactly the same disambiguation algorihm is used with the "Add Selection" command, described below.

The most commonly-used commands in YALE are the "Add Rectangle" and "Select" commands, accessed using MB1 and MB2. All the other YALE commands are invoked through a series of one or more pop up menus. The main pop up menu is accessed by pressing MB3. It presents 10 options that are described in the following chapters. Most of the items in this main menu are really just categories of commands, and simply bring up a sub-menu containing specific commands. A few of the entries, however, are bound directly to commonly-used commands.

6. Create commands

Selecting the “Create” entry from the main YALE pop up menu brings up the sub-menu of the create commands. There are five entries in the sub-menu, including “Create Array”, “Create Cell Definition”, “Create Cell Instance”, “Create Copy of Cell Definition” and “Create Reference Point”. Each is described individually in a section below.

6.1 Create Cell Definition

This command creates a brand-new symbol (cell) definition. You are asked to type in a name for the new symbol, and that symbol is created and opened. The newly created symbol will contain nothing but an x- and y- origin, and will be presented in all YALE’s graphical viewports with exactly the magnifications that were there previously. If this command is accidentally selected, and it is not desired to create a new cell definition, simply type a <CR> instead of a cell name. This aborts the command as if nothing happened.

6.2 Create Cell instance

This command inserts an instance of a previously defined cell within another. After selecting this command, you next show which symbol is to be inserted (via the most-recently-used-cell pop up). If you wish to place the symbol exactly as defined (no rotation or reflection), simply identify with MB3 exactly where you wish the origin to go. If the newly-inserted instance is to be placed with a rotation or reflection, again show where it is to go, but do so with MB1. You will then be presented with a pop up menu including such things as “flip lr", “flip ud", “rotate 3", “rotate 6", and “rotate 9”. "flip lr" and “flip ud" refer to mirrorings (through the origin) left-right and up-down, respectively. The numbers in the rotate command are described in terms of a standard clock face. Imagine that in your original symbol, the hour hand points straight up. “rotate 3” means to rotate that hour hand until it points to the "3", and so on. Any combination of these transformations can be specified, and they are done one after another to the symbol before it is placed. In practice, at most 2 are required. For example, a “rotate 3” followed by a “rotate 6” would be exactly equivalent to a “rotate 9” selection, but there is no easy way to specify the the combination of a “rotate 3” followed by a “flip lr" command.

When you are finished specifying the transform, choose the bottom selection from the menu. The symbol will then be placed where the original MB1 was pressed.

As was the case with rectangles, the symbol instance is selected as it is inserted. In this way, if it is placed incorrectly, it is easy to issue a “Move Selected Objects” command and adjust it to the correct position.

The new symbol instance’s origin (the intersection of the x- and y- origin reference points) is placed relative to the currently selected reference points in the currently open cell. An attempt to insert a symbol when no cell is open results in an error message in the tty viewport.

One often wishes to repeat this command over and over -- inserting a series of symbol definitions. Therefore, the “Create Cell Instance” command is repeatable. As soon as it is used, the "Rpt Cmd:" entry in the status viewport is updated to show that the currently repeatable command is “Create Instance”. To repeat the command, simply press MB1 and MB2 at the same time. You will immediately be presented with the pop up menu of recently touched symbol definitions, and from then on, the command proceeds in exactly the same way as if you had gotten there by the usual path of getting the main YALE pop up, selecting “Create”, and finallv selecting the sub-menu entry “Cell instance”.

The “Create Cell Instance” command remains the repeatable command until another repeatable command is issued. There are only two others -- “Delete Selections”, and “Add Selections”.

6.3 Create Copy of Cell Definition

General SILT allows cells to be called over and over again with different parameters. YALE will eventually allow this, but now, although each cell definition may allow fcr stretchability, it can only be called with one set of parameters. If you wish to use a cell definition in two places with different parameters, you must make a copy of the cell definition, with a different name, and use that new definition with the new parameter set.

To make a copy of a cell definition, use the “Create Copy of Cell Definition” command. You will be asked to identify first the name of the cell definition to copy using the menu of recently touched symbol names. Next, you must type in the new name for the copy. The command can be aborted in two ways. You can bug outside the menu of most recently used cells, or you can type a <CR> when a typein is requested.

After the copy is made, YALE assumes that you are going to want to do something with it, so the previously open cell is closed, and the newly-created cell is opened and displayed in YALE’s graphics viewports.

This command makes use of a scratch file on the remote host, so you must have a valid user name and password to use it. A scratch file called "_yale_scratch" is created there, and is only used during the execution of this command. The file can be deleted at any time, although it will be small in general, and can safely be left there.

6.4 Create Array

This command places a rectangular array of instances of the same symbol. At present, this command is a bit clumsy to use, but it is easier than putting in the symbols one at a time. When the command is issued, you are prompted in the tty viewport for the number of symbols to be placed in the x- and y-directions, and then for the x- and y- displacements. After this information is provided, you specify the position of the origin of the symbol in the lower left hand corner of the array. This is done in exactly the same way as it is for the simple "Insert Symbol" command, using MB3 to insert the members of the array in their standard orientation, and MB1 to specify a mirroring or rotation transformation, or some combination of the two.

When an array of symbols is inserted, all are selected, so that the array can be moved or deleted as a block if some error in spacing or placement was made.

If such an error is made, it is a good idea to correct it immediately, since The inserted array is not considered internally by YALE to be an array, but rather a series of individual symbol calls. Thus individual symbols in the array can be moved or deleted after the array is put in.

6.5 Create Reference Point

This command makes a new reference point. Reference points have a tree-like dependence, where the x- and y-origins serve as the the roots of the two reference point trees. When a reference point is moved, not only is all the geometry associated with it moved rigidly, but also all reference points that are relative to it. Thus, when a new reference point is inserted, it is placed relative to another reference point (which may be an origin).

After this command is issued, you are first prompted for the "parent" reference point (which is identified by bugging it with a mouse click). After this, its screen position is identified in the same way, and finally, the textual name for the new reference point must be typed in.

7. Setting Defaults

YALE has a few default values that can be set by the user that affect the actions of some of the editing and viewing commands. One of these, the default layer selected, has been described earlier. Every time an "Add rectangle" command is issued, the rectangle is made of material from the selected layer. To select a layer, simply point to the appropriate stipple sample in the status viewport, and bug it with a mouse click. The other two default settings are new, and are described in the sections that follow. All are accessed initially by bugging the "Defaults" entry in the main YALE menu.

7.1 Setting Default Rectangle Widths

Each of the layers (metal, polysilicon, etc.) has a default width associated with it. In other words, when the default layer is set to be metal, and you insert a rectangle, the rectangle will be in metal, its length will be determined by the placement of the mouse clicks, and its width will be determined by the default width for the metal layer.

The current default widths for the various layers can be determined by looking at the number printed under the stipple samples in the status window. They can be changed using this command.

The "Set Default Widths" command is a multi-level pop up menu command. After bugging "Defaults" in the main menu and the "Line Widths" entry in the sub-menu, another sub-sub-menu of layers is presented. When one'of these is selected, a fourth menu comes up that has as entries a set of typical small numbers and a "type in" option. If the width you want is in this menu, bug it, and you are done. If you need an unusual default width, bug the "type in" menu entry, and type in the number you want, followed by a <CR>. The change takes place immediately, and the information in the status window will be updated.

7.2 Setting Default Expansion Depths

In a complicated layout, when editing the top level symbol, you may not care about the internal detail? of the sub-cells. Filling in all the wires and transistors may even tend to clutter the screen too much. Therefore, a method is provided to control the expansion depth when viewing a symbol.

The cell that is currently open is at level zero, as are the rectangles contained within it. The subcells of the open cell are, together with their included rectangles, at level 1. The cells and their rectangles contained within these sub-cells are at level 2, and so on.

Thus, if the expansion depth is set to zero, only the rectangles in the currently open cell are visible. Any sub-cells are simply indicated by an outline on the screen enclosing their name. All details of their interiors are hidden. If the expansion depth is set to 1, the rectangles within these cells are visible, but not the interiors of their sub-cells, and so on.

When YALE starts up, the expansion depth is set to a very large number (32767), so that essentially everything is visible.

Changing the expansion depth is much like setting the default rectangle widths, described above. After bugging “Default”, and “Expansion Depth”, you will be presented with a menu containing a set of small numbers, together with an entry marked “all”, as well as the usual "typein" entry. The “all” entry sets the expansion depth to 32767, and the "typein" entry works just as it did in the “Set Default Widths” command -- just type in the required depth, followed by a <CR>. This command takes effect immediately, and the screen is redrawn with the new viewing parameters.

The default expansion depth affects all the YALE graphics viewports. Someday this should be changed to work on a viewport by viewport basis.

8. Select, Delete, and Move Commands

8.1 Selection

To move or delete objects in a symbol definition, it is necessary first to identify what those objects are. This is the purpose of the selection commands. The most commonly used selection command has already been described, and it selects a new reference point, syrnbol instance, or set of rectangle edges. If the selection is a symbol instance or set of rectangle edges, everything previously selected is first de-selected. If the selected object is a reference point the old selected reference point (with the same orientation -- vertical or horizontal) is first de-selected.

Add Selection. There can never be more than one reference point in each orientation (vertical or horizontal) selected, but it is often convenient to select many rectangles and symbol instances at the same time. This can be done with the "Add selections" entry of the sub-menu gotten from the "Selection" entry of the main YALE menu.

After bugging the "Add Selections" entry, the next mouse click is interpreted in the same way as MB2 is interpreted for a straight selection. The only difference is that previously selected items are not de-selected first.

Since it is common to use this command repeatedly to get many things selected at the same time, the command is repeatable in the same way as was "Create Cell Instance". If "Add Selections" is the current repeatable command, it is easy to use. Simply move the cursor to point to the object to be selected, and press **MB1** and MB2 at the same time. The object pointed to will immediately be added to the selection list.

Select Ref. Pt. Dependents. The command to "Select Reference Point Dependents" selects all the rectangle edges and symbol instances that are dependent on some reference point. Before this happens, all other selected items are first de-selected. This command is most often used simply to find out what the reference point dependencies are -- that the items are selected is merely a side effect. The selection is a real one, however, and those items can be moved, deleted, and so on.

After bugging the entry for "Ref. Pt. Dependents", you are asked to bug a reference point, and it is items relative to this reference point that are selected.

8.2 Deletion

Delete Selections. There are three commands under the "Delete" entry of the main YALE pop up menu. The most important is "Delete Selections", which does just that -- all completely selected rectangles (those rectangles having all four edges selected) and selected symbol instances are deleted from the currently open cell. The deletion is permanent -- the deleted items cannot be retrieved from something like a "yank buffer" that is present in some textual and graphical editors.

The "Delete Selections" command is repeatable, and you can use it to delete a series of objects by alternately selecting an object with MB2, and then deleting the object by pressing MB1 and MB2 simultaneously.

Delete Cell Definition. The second delete command deletes an entire cell definition. This can only be done if that cell is called by no other cell in the layout. If this is not the case, an error message appears in the tty viewport.

Delete **Reference Point.** Finally, there is a command to delete a reference point. After bugging the "Reference Point" entry in the sub-menu, you will be asked to identify a reference point by bugging it. If the reference point has any geometry dependent on it, the deletion will not be carried out, and an error message will be presented in the tty viewport. If this is the case, remember that there is a command ("Select Reference Point Dependents", see 8.1) which will show all the dependent geometry.

8.3 Moving

Move Reference Point. The main reason for having reference points in YALE is to move them around to stretch and shrink cell definitions. This is done with the "Move Reference Point" command.

To do it, first bug the "Move" entry in the main menu, then the "Reference Point" entry in the sub-menu. Next, identify the reference point to be moved by bugging it, and identify its new location by bugging the screen. The reference point, together with all associated geometry (symbol instances and rectangle edges) should be moved rigidly along with it.

Move **Selections.** This command moves all the selected rectangle edges and cell instances rigidly to some other position in the cell. The motion references al! items to the currently selected reference point, but the dimensions of the moved objects all remain fixed, even if they were previously dependent on (placed relative to) two different reference points.

To use this command, bug the “Move” and “Selections” entry in the main YALE menu and in the sub-menu, respectively. You will then be asked to show how one point will move (remember that everything else moves the same way) by bugging its initial and then final positions. A common thing to do is to bug one corner of a selected rectangle where it is, and then to bug the position to which that corner should move. If all selected items are thought of as rigidly attached to the first point, the move command is equivalent to translating that point with everything attached to the second moused position.

Moving a collection of rectangle edges and cell instances does not de-select them. Thus, if an error is made, and they are moved to the wrong place, it is easy to give the move command again, and adjust the position again.

When only one of a rectangle’s edges is selected, and the “Move Selections” command is issued, a somewhat surprising result can occur if the selected edge is moved so that it winds up on the other side of the unselected edge from which it was before. YALE simply remembers that, say, “the right edge is selected”, and when what was the right edge becomes the new left edge, YALE still remembers that the right edge is selected. This situation does not often arise with norrnal cell transformations.

9. Input/Output

The SUN workstations do not have any local permanent storage, so all the data must be read from and written to remote files. This is done using the ethernet, and the files are available from any other ethernet host that provides remote file access, and supports the "leaf" protocol.

At the beginning of a YALE editing session, some initialization is done, including the opening of a connection to such a host, the specification of a user name, password, and of a main file name. After the initialization, all other input/output is done using the "Input/Output" entry in the main YALE pop up menu.

There are three kinds of remote files used by YALE. The first is the main input/output file that contains the current version of the layout being edited. Generally, this file is opened during initialization, is read then, and is written at the end of the YALE editing session. There is only one main input file used during a YALE session.

The second kind of file is the backup file. Whenever you give the "Backup" input/output command, this backup file is over-written with the current contents of the YALE memory. The previous backup version is lost. There is only one backup file used during a YALE session.

Finally, YALE supports (in a primitive way) library files. A library of cell definitions can be read in and combined with the user-defined cell definitions. Once they are combined, a copy is permanently kept in the user's layout file. Because of this, the best way to implement libraries is probably as small files, each containing only one or two library symbols. YALE has no restrictions on how many library files may be read in.

Of the three file types mentioned above, the library files are opened in a read-only mode, and the other two are opened in read-write mode. Errors may occur if the user you logged in as does not have the appropriate access to the files in question.

In general, the input/output structure could be improved a lot. There are, however, enough commands to do almost anything, albeit somewhat clumsily. Some of the defects can be remedied by going in with your favorite text editor and modifying the SILT files on the machine providing remote file storage.

9.1 initialization

When you begin a YALE session, some initialization must be done. You will be asked for information about the main SILT file to be used for input and output. To find this out, YALE must connect to a host, log you in, and read the appropriate file. You will be prompted in the tty viewport (initially at the bottom of the screen) for a host (<CR> gives Shasta), a user name and password, and finally a file name. Type each of these followed by a <CR>. After this, you may issue any of the standard YALE commands.

If the file whose name you give in response to the request for the main file name does not exist, YALE assumes that you wish to create a new file by this name, and does so.

If you give a null file name (i. e. just type <CR>), YALE will not try to open a remote file. Thus, if you can experiment with YALE even if you do not have an account on a machine with an active leaf server. Just type <CR> in response to the user name and password requests. obviously, if you do this, you will have an empty layout, and will have to create everything from scratch. If you do this, and then decide that you really want to save your edits, you can always re-initialize the connection -- see the next paragraph.

The initialization commands are also available within an editing session as the menu entry "Initialize", found under the “Input/Output” entry in the main YALE pop up menu.

9.2 Close Connection

This command closes the currently open leaf connection.

9.3 Read Library

When this command is issued, you are asked for the name of a SILT library file, and that file is opened, and all the cell definitions contained therein are added to those in your layout.

9.4 User Name ,

The “User Name” command sends the remote system a new user name/password combination. At any point, YALE only keeps track of one such combination, so if you use this command to read in a library file, you had better use it again to connect back to the main directory if you wish to write out your main file after editing.

9.5 Write

The "Write" command writes out the contents of memory into the main file. If you have not yet specified it, you will be asked for the name of the main symbol to be used. The symbol name you select is added to the end of the SILT file produced so that it is the symbol expanded to create the whole layout. For example, if you select "foo" as your main symbol, a line of the form:

```
place foo() at (0, 0);
```

is put in as the only symbol call in the main begin-end block of the SILT file.

The reason that the main symbol name is requested during each YALE session instead of inferring it from the input file is that in this way it is possible to change it. YALE should be changed so that it is usually inferred, but can be changed with a specific "Default" or "Input/Output" command.

9.6 Write Backup

The "Write Backup" command is exactly like the "Write" command above, except that the information is written onto a backup file instead of the main YALE output file. The first time this command is issued, you will be asked for the name of the output fife, and if you have not yet specified it, the name of the master symbol. After this information has been provided once, you will never be asked for it again during that session.

10. Miscellaneous YALE commands

Although the chapter is entitled “Miscellaneous YALE commands”, this does not mean that they are not important. In fact, the commands documented here tend to be extremely important. All are accessed via only a single level of pop up menu.

10.1 Expand Cell

At any point in a YALE editing session, exactly one cell is open for editing. The “Expand Cell” command opens this cell. You show which cell is to be opened using the most-recently-used pop up menu mechanism. When a cell name is selected or typed in, the old cell is closed, and the new cell is presented in all the YALE graphics viewports. Everything in the cell (except, of course, for the vertical and horizontal origin reference points) is de-selected.

10.2 Re-reference

As each rectangle or symbol instance is added to an open cell, it is placed relative to the currently selected horizontal and vertical reference points. For rectangles, both edges are so placed, so without this command, there would be no way to make stretchable rectangles. The other common use for this command is when it is desired to make a cell flexible in some position where it was previously rigid. For example, suppose that it is desired to make a cell that was originally totally rigid so that it is stretchable in the x-direction. To do this, simply add a vertical reference point, and re-reference the items generally to the right to this new reference point. Move the new reference point around a little to see if all the stretching is done correctly, and then do some more re-referencing to correct errors.

The “Re-reference” command deals with the currently selected items, and leaves them in place, but re-references them to a new reference point that is identified by bugging it. It is effectively exactly the same as selecting the new reference point, and then “moving” all selected items by a distance of zero.

10.3 Window Manager

This command is available to allow an easy transition to the window manager from within YALE without requiring the user to take his hands of the mouse. It is exactly equivalent to typing the “brain escape character”’

10.4 Show Cell Definition Names

This command displays a complete list of all the cell definitions currently in the layout. The names of the cell definitions are printed in the tty viewport, six to a line. The initial size of the tty viewport is small, and if there are many names, or if there are long names, they will not all fit in the viewport. Remember, however, that the tty viewport is like any other viewport on the display, and its size and shape can be modified with commands to the window manager. Just because the lines of text are not visible does not mean that they are not there.

10.5 Quit

The “Quit” command exits from YALE. It closes all open files on the remote machine, and closes the connection. The user is then returned to the SUN monitor. No warning is given if the edits have not been saved. In almost all cases, the next-to-last command is to write out the main file (see the next chapter).

The “Quit” command *must* be confirmed with a "y<CR>" (or a "Y<CR>"). To abort the “Quit” command, simply type anything else -- a raw <CR>, "n<CR>", or anything else. The editing session will then continue as if nothing has happened.

11. Errors

11.1 The '%' Command

If YALE (or any other user process running with the window manager) gets totally hung for some reason, there is an emergency abort that almost always works. Simply type the percent character, and everything should halt, returning you to the SUN monitor. This is a last-ditch attempt, and everything about the editing session is lost (except, of course, for any backup files that were written).

It works because there is a separate process watching the keyboard input, and even if some other process is stuck in a loop, the keyboard process still gets run regularly.

Use this command instead of the usual break command, since this one shuts up the mouse. (On some of the SUN monitors, the mouse is turned on by sending a command to the keyboard, and it continues to transmit its coordinates until it is turned off. The break key does not turn off the mouse, and the terminal is left in a useless state until the keyboard is unplugged or the terminal is powered down.)

11.2 SILT Parsing Errors

If all your editing is done using YALE, you should never have any trouble with this. On the other hand, if you go in with an editor and edit the SILT files produced by YALE by hand, it is not too hard to introduce syntax errors. YALE tries to read the file containing errors as best it can, and to continue after the error(s), but it will obviously sometimes be unsuccessful. There are many messages related to SILT syntax errors, and they should be self-explanatory. All such errors will be written in the tty viewport.

If such errors occur, the best bet is to go back to the original files and edit them until the errors go away.

11.3 Running out of space

No matter how much storage is available, it is possible to make a layout that is too big for it. Since YALE can run out of storage at totally unpredictable times, it is impossible to predict what the consequences are. Some of the storage is not efficiently reclaimed, and thus after editing for a long time, you may run out of space even though the layout has not gotten much more complicated. Writing out the file, restarting YALE, and reading in the file again can sometimes help.

Some mechanism should be created in YALE to cache data on the remote host, but this has not yet been done.

If the layout is just too big to deal with, YALE can be used to put together parts of it, and all the parts can be combined by hand using SILT to wire together the pieces. The SILT translator, running on the VAX, can handle very large layouts.

A warning about low free storage is hard to give, since a single YALE command can use up a large amount of it. For example, creating a large array of symbol instances can chew up free storage in a hurry. When free storage is exhausted, the message "Out of free storage" is put up in the tty viewport, and YALE attempts to leave its data structures in a reasonable state, but this is not always possible. The best advice is to back up your work frequently when you think that you may be low on free storage.

11.4 Other Size Limitations

YALE has no absolute cell size -- a symbol can call as many rectangles as it wants. There is an implementation restriction on the number of symbol calls that a given symbol can make. A single cell must contain less than 300 symbol calls, but it is not as simple as that. If symbols are deeply nested, this number can be reduced. If, during an "Expand Symbol" command, the error "Too many symbol calls" occurs, try reducing the number of symbol calls in symbols with a large number of them.

Another scarce resource is something called "Sun Instance Numbers".' The number of these is a compiled-in constant, so can be changed by recompiling. Each cell definition, cell instance, reference point, and rectangle has a Sun Instance Number. When an object is deleted, its Sun Instance Number is returned to a pool of free numbers. Sun Instance Numbers are used to look up quickly information about items displayed on the screen.

11.5 Bugs

Some of the error messages are prefixed by the word "bug". These should never be printed out. If one is, please report it to the YALE maintainer, together with as much information about what was going on as possible.

Reports of other bugs are also welcome.

I. Window Manager Command Outline

This is a list of all the commands available in the window manager. Each command is followed by the page number on which it is more fully described.

Zoom In (MB1) 8
Zoom Out (MB2) 8

Brain Escape Character 10
Center Window 9
Create Viewport 9
Delete Viewport 9
Move Edge 9
Move Viewport 9
Redraw 10
Return to Yale 10
Toggle Grid 10

% (emergency abort command) 35

II. YALE Editor Commands

This is a list of all the commands available in the YALE layout editor. Each command is followed by the page number on which it is more fully described. Those commands followed by an asterisk (*)are repeatable.

Add Rectangle (MB1) 17
Select Item (MB2) 17

Create Array (of instances) 21
Create Cell Instance (*) 19
Create Cell Definition 19
Create Copy of Cell Definition 20
Create Reference Point 21

Default: Set Default Width 23
Default Expansion Depth 23

Delete Cell Definition 26
Delete Reference Point 26
Delete Selections (*) 26

Expand Cell 33

Input/Output: Initialize 30
Input/Output: Close Connection 30
Input/Output: Read Library 30
Input/Output: User Name 30
Input/Output: Write 31
Input/Output: Write Backup 31

Move Reference Point 26
Move Selections 26

Re-reference Selections 33

Select: Add (to) Selections (*) 25
Select Reference Point Dependents 25

Show Cell Definition Names 34

Quit 34

Window Manager 33

References

1. Brown, David J., and Nowicki, William I. A Package of Graphics Primitives for SUN. Stanford internal documentation of the Raster Operation routines.

2. Cheriton, David R., and Mann, Timothy P. The V Kernel: A Distributed Message-based Kernel. Stanford, 1982.

3. Davis, Tom, and Clark, Jim. SILT: A VLSI Design Language. Stanford, 1982.

4. Mogul, Jeffrey. Leaf and Sequin Protocols. Unpublished paper, Stanford University, March 12, 1981