**King Saud University**

College of Engineering

### Electrical Engineering Department

#### EE 406

#### VLSI Circuit Design Laboratory

#### First Semester 1435/1436

**By**

Prof. Shuja A. Abbasi

Prof. A. R. M. Alamoud

**King Saud University**

College of Engineering

### Electrical Engineering Department

#### EE 418: VLSI Circuit Design Laboratory

#### First Semester 1435/1436

##### Instructor: Prof. Shuja A. Abbasi; Office - Room 2C94, Tel. - 46-76729

http://faculty.ksu.edu.sa/Abbasi

**Books:** 1. Yalamanchili, “Introductory VHDL”, Prentice Hall, 2001.

2. Uyemura, “Physical Design of Integrated Circuits using L - EditTM”, PWS Pub. Co, 1995.

***The Schedule***

|  |  |
| --- | --- |
| Week # 1 | Lecture: VLSI Circuit Design |
| Week # 2 | Lecture: VHDL/ASIC |
| Week # 3 | Lecture: VHDL/ASIC |
| Week # 4 | Design and Implementation of simple circuits |
| Week # 5 |
| Week # 6 | Design and Implementation of a 2–bit Multiplier |
| Week # 7 |
| Week # 8 | Design and Implementation of 4–bit Multiplier |
| Week # 9 |
| Week # 10 | **Mid – term Examination** |
| Week # 11 | Laboratory Project |
| Week # 12 |
| Week # 13 |
| Week # 14 | Project Presentation |
| **Week # 15** | **Final Examination** |

***Grading:***

Lab. reports: 10%

Quizzes: 10%

Lab. Project: 20%

Mid-term Examination: 30%

Final Examination: 30%

**VHDL**

* VHDL is the VHSIC Hardware Description Language.
* VHDL is a language for designing digital electronic systems.
* Soon, every design engineer in the electronic industry should learn and use a hardware description language to keep pace with the productivity of competitors.
* VHDL is designed to fill a number of needs in the design process, e.g.:
* It allows description of the structure of a system.
* It allows the specification of the function of a system using familiar programming language forms.
* It allows the design of a system to be simulated before being manufactured.
* It allows the detailed structure of a design to be synthesized from mere abstract specifications, allowing designers to concentrate on more strategic design decisions and reducing time to market.
* In addition, VHDL provides the following capabilities:
* Power and flexibility.
* Device/Technology independent design.
* Portability.
* Benchmarking capabilities.
* ASIC migration.
* Quick time-to-market and low cost.
* Non-reverse engineering design.

**Main Features of VHDL**

* Entities and Architectures
* Concurrent Assignments and Signals
* Components and Port Maps
* Std\_logic and std\_logic-vector
* Test Benches

**Design Entity**

* It represents a block of hardware with well defined inputs and outputs and a well defined function.
* A design entity is split into two parts:

1. **The *entity declaration***: it represents the external interface to the design entity.

**Library IEEE;**

**use IEEE.STD\_LOGIC\_1164.all;**

**entity** Half\_Adder **is**

**port** (A, B: **in** std\_logic**;**

Sum, Cout: **out** std\_logic);

**end** Half\_Adder;

|  |
| --- |
| **Sum**  **Cout**  **B**  **A**  **Half Adder** |

1. **The *architecture body*:** it represents the internal description of the design entity.

**architecture** half **of** Half\_Adder **is**

**begin**

Sum **<=**A **xor** B;

Cout <=A **and** B;

**end** half;

* **It can be built hierarchically from other design entities.**

design entity

design entity

**Another Example**

**AND OR Invert (AOI) Gate**



**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.all;**

**entity** AOI **is**

**port** (A, B, C, D: **in** std\_logic**;**

F : **out** std\_logic);

**end** AOI;

**-- This is the internal description of my circuit(AOI);**

**architecture** Arch1 **of** AOI **is**

**begin**

**F<= not ( (A and B) or (C and D));**

**end** Arch1**;**

**[ ] { }**

**A.B + C.DInternal Signals**

A

C

O

B

D

AB

CD

F

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.all;**

**entity** AOI **is**

**port** (A, B, C, D: **in** std\_logic**;**

F : **out** std\_logic);

**end** AOI;

**architecture** Arch2 **of** AOI **is**

**signal AB, CD, O: STD\_LOGIC;**

**begin**

**AB <= A and B after 2 NS;**

**CD <= C and D after 2 NS;**

**O <= AB or CD after 2 NS;**

**F <= not O after 1 NS;**

**end** Arch2**;**

**Components**

|  |
| --- |
| **FB**  **F**  **B**  **SELB**  **AOI**  **A**  **SEL** |

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.all;**

**entity** INV **is**

**port ( A: in STD\_LOGIC; F: out STD\_LOGIC);**

**end** INV;

**architecture** INV **of** INV **is**

**begin**

**F<= not A;**

**end** INV**;**

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.all;**

**entity** MUX2 **is**

**port** (SEL, A, B,: **in** std\_logic**;** F : **out** std\_logic);

**end** MUX2;

**architecture** STRUCTURE **of** MUX2 **is**

**component INV**

**port ( A: in STD\_LOGIC; F: out STD\_LOGIC);**

**end component;**

**component AOI**

**port ( A, B, C, D: in STD\_LOGIC; F: out STD\_LOGIC);**

**end component;**

**signal SELB, FB: STD\_LOGIC;**

**begin**

**G1 : INV port map (SEL, SELB);**

**G2 : AOI port map (SELB, A, SEL, B, FB);**

**G3 : INV port map (FB, F);**

**end** STRUCTURE **;**

**Vector Ports**

|  |
| --- |
| MUX4 FDCA SEL B |

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.all;**

**entity** MUX4 **is**

**port** (**SEL : in** std\_logic\_VECTOR (1 downto 0)**; A, B, C, D: in** std\_logic;

**F: out** std\_logic);

**end** MUX4;

**std\_logic & std\_logic\_VECTOR**

* **std\_logic represents one digital logic value –**

**signal B: std\_logic;**

**B <= ‘0’; -- values ‘U’ ‘X’ ‘0’ ‘1’ ‘Z’**

* **std\_logic\_VECTOR represents a vector or bus**

**signal V: std\_logic\_Vector (7 downto 0);**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **V:** | **0** | **1** | **1** | **0** | **1** | **X** | **X** | **X** |
|  | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |

**V <= “01101XXX”;**

**V(7) <= ‘0’;**

**V(6) <= B and V(0);**

**Processes**

***Process*:** An important feature of VHDL is the ability to express the function of the hardware block by writing in the style of a software programming language. This is achieved by using the *Process Statement*.

***Concurrency*:** All process statements can execute concurrently with respect to each other. Statements inside a process are executed in sequence until the process suspends.

*Network Model*: In VHDL, ckts are modeled as a network of processes, connected by signals. Processes represent system or hardware blocks that transform data, signals represent wires, busses or data channels that allow data to pass between those blocks.

*Communication*: Processes communicate with each other via signals.

**Examples:**

* **With sensitivity list**

|  |  |
| --- | --- |
| library IEEE;  use IEEE.STD\_LOGIC\_1164.all;  entity MUXX is  port (SEL, A, B, C: in std\_logic;  OP: out std\_logic);  end MUXX;  architectureMUXX **of** MUXX **is**  begin  process (SEL, A ,B, C)  begin  if SEL = ‘1’ then  OP <= A and B;  else  OP <= C;  end if;  end process;  end MUXX; | **SEL** A  C  ‘1’  ‘0’  B **MUX****OP** |

**With wait statements**

**STIMULUS: process**

**begin**

**Reset <= ‘0’;**

**wait for 50 ns;**

**Reset <=’1’;**

**wait;**

**end process STIMULUS;**

* **Clocked Process**

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity FFP is

port (Clock, D: in std\_logic; Q: out std\_logic);

end FFP;

architectureFFP **of** FFP **is**

begin

**FLIPFLOP: process**

**begin**

**wait until Clock =’1’;**

**Q <=D;**

**end process FLIPFLOP;**

end FFP;

**Signal assignments**

**process (A, B)**

**begin**

**B <= A after 10 NS;**

**C <= B after 5 NS;**

**end process;**

## 5 NS

## 10 NS

C

B

A

**VHDL Code of a Two – bit Multiplier**

0

P(3)

P(2)

P(1)

P(0)

B(1)

B(0)

A(1)

A(0)

C Sum

Cin Y X

C Sum

Cin Y X

a2

a1

Temp(1)

A4

a3

Temp(2)

Temp(3)

Full Adder

Temp(4)

0

Full Adder

**Figure: A 2-Bit Multiplier**

**The VHDL code may be written as:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity AND2 is

Port ( in1 : in std\_logic; in2 : in std\_logic; out1 : out std\_logic);

end AND2;

architecture AND2 of AND2 is

begin

out1 <= in1 and in2;

end AND2;

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity FullAdd is

Port(X: in STD\_LOGIC; Y: in STD\_LOGIC; Cin: in STD\_LOGIC;

SUM: out STD\_LOGIC; C: out STD\_LOGIC);

end FullAdd;

architecture Fadd of FullAdd is

begin

SUM <= (X xor Y xor Cin);

C <= ((X and Y) or ((X xor Y) and Cin));

end Fadd;

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Mul is

Port ( A : in std\_logic\_vector(1 downto 0);

B : in std\_logic\_vector(1 downto 0);

P : out std\_logic\_vector(3 downto 0));

end Mul;

architecture Mul of Mul is

component AND2 Port ( in1 : in std\_logic; in2 : in std\_logic;

out1 : out std\_logic);

end component;

component FullAdd Port(X: in STD\_LOGIC; Y: in STD\_LOGIC;

Cin: in STD\_LOGIC; SUM: out STD\_LOGIC;

C: out STD\_LOGIC);

end component;

signal temp: STD\_LOGIC\_VECTOR(1 to 4);

begin

G1:and2 port map (A(0),B(0),P(0));

a2:and2 port map (A(1),B(0),temp(1));

a3:and2 port map (A(0),B(1),temp(2));

a4:and2 port map (A(1),B(1),temp(3));

f1:FullAdd port map (temp(2), '0', temp(1), P(1), temp(4));

f2:FullAdd port map ('0', temp(4),temp(3), P(2), P(3));

end Mul;

**The Application Specific Integrated Circuits (ASICs)**

* A wide variety of IC chips is available in the market as off – the – shelf items.
* An obvious method for designing and making (implementing) a circuit for an application is to use standard off – the – shelf ICs and solder them on a PCB.
* A relatively newer method is to design and fabricate the entire circuit on a single chip i.e. a circuit customized to a particular application.
* This is called an ASIC(Application Specific Integrated Circuit) or Custom IC.
* Special methods are required for the design and fabrication of ASICs since the economics of IC tech. is heavily dependent on volume of production.
* These methods are aimed at

(i) **reducing cost to acceptable limits at low volumes of production and**

(ii) **reducing turn around time to acceptable limits.**

* **The ASIC technology offers many advantages like –**

1. **cost reduction.**
2. **decreased size.**
3. **Increased performance ( reducing interconnections, excess area penalty, reducing parasitic capacitances) .**
4. **Low power consumption.**
5. **Special functions.**
6. **Improved service .**
7. **Copy protection (reverse engineering).**
8. **Short time to market.**
9. **More reliability (low soldered joints).**

**Types of ASIC:**

## **ASIC**

## **Full Custom ASIC**

## **Programmable ASIC**

## **Semi-Custom ASIC**

## **PLDs**

## **FPGAs**

## **Standard-Cell based**

## **Gate -Array based**

## **Channelless Gate -Array**

## **Channeled Gate -Array**

## **Structured Gate -Array**

**Programmable Logic Devices**

* **Programmable logic devices (PLD’s) are standard IC’s that are available in standard configurations from a catalog of parts and sold in a very high volume to many different customers.**
* **All PLD’s have the following important features in common:**

1. **No customized mask layers or logic cells.**
2. **Fast design turnaround.**
3. **A single large block of programmable interconnects.**
4. **A matrix of logic macrocells that** usually consist of programmable array logic followed by a flip-flop or latch.

**Field-Programmable Gate Arrays (FPGA’s)**

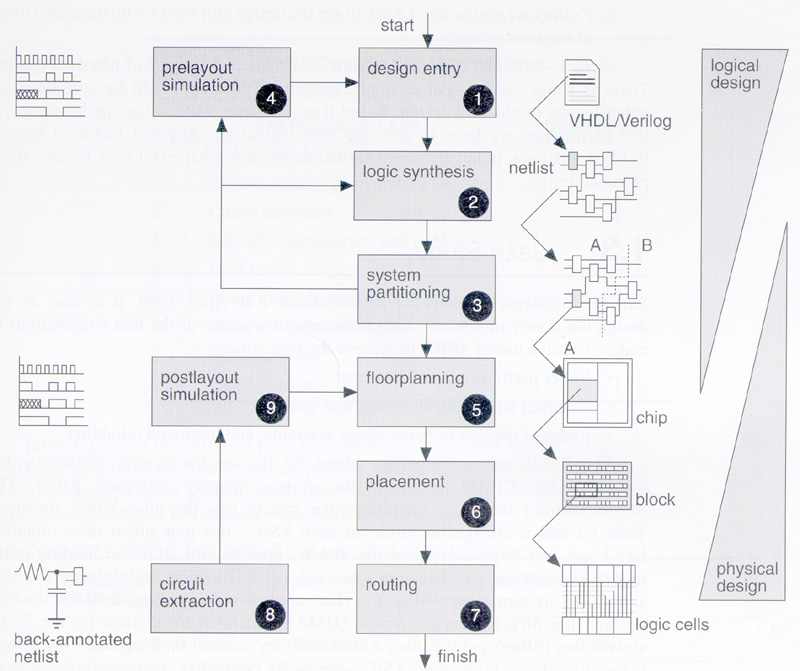
* **A step above the PLD complexity is the field-programmable gate arrays (FPGA).**
* There is very little difference between an FPGA and a PLD – an FPGA is usually just larger and more complex than a PLD.
* One of the most important features of the FPGAs is that they are **reconfigurable**.
* An FPGA is a chip that a system designer can program.
* An IC foundry produces FPGAs with some connections missing.
* The designer performs the design entry and simulation.
* Next, special software creates a string of bits describing the extra connections required to make the design – the configuration file.
* The designer then programs the chip to make the necessary connections according to the config. file.
* There is no customization of any mask level for an FPGA, allowing the FPGA to be manufactured as a standard part in high volume.
* FPGAs are ideal for prototyping systems or for low-volume production.
* All FPGAs have certain key elements in common.
* All FPGAs have a regular array of basic logic cells that are configured using programming technology.
* The chip inputs and outputs are special I/O logic cells that are different from the basic logic cells.
* The programming technology may or may not be permanent.
* You cannot undo the permanent programming in one-time programmable (OTP) FPGAs.
* Reprogrammable or erasable devices may be reused many times.
* The essential characteristics of an FPGA are:

1. None of the mask layers are customized.
2. A method for programming the basic logic cells and the interconnect.
3. The core is a regular array of programmable basic logic cells that can implement combinational as well as sequential logic cells.
4. A matrix of programmable interconnects surrounds the basic logic cells.
5. Programmable I/O cells around the core.
6. Design turnaround is a few hours.

**DESIGN FLOW**

* Design flow is the sequence of steps to design an ASIC. The typical steps are:

1. ***Design entry:* enter the design into an ASIC design system, either using schematic entry or using a hardware description language (HDL).**
2. ***Logic synthesis:* use an HDL (VHDL or Verilog) and a logic synthesis tool to produce a netlist – a description of the logic cells and their connections.**
3. ***System partitioning*: divide a large system into ASIC – sized pieces.**
4. ***Prelayout simulation:* check to see if the design functions correctly.**
5. ***Floorplanning:* arrange the blocks of the netlist on the chip.**
6. ***Placement:* decide the locations of cells in a block.**
7. ***Routing:* make the connections between cells and blocks.**
8. ***Extraction:* determine the resistance and capacitance of the interconnect.**
9. ***Postlayout simulation:* check to see the design still works with the added loads of the interconnect**.
10. ***Configure the FPGA***

**PROGRAMMABLE ASIC DESIGN SOFTWARE**

* For the design of any ASIC, a designer needs:
* design -entry software,
* a cell library, and
* physical-design software.
* Each of the FPGA vendors sells design kits that include all the software and hardware that a designer needs.
* Many of these kits use design-entry software produced by a different company.
* Design entry uses cell libraries that are unique to each FPGA vendor.
* After completing design entry and generating a netlist, the next step is simulation.
* Two types of simulators are normally used for FPGA design.
* The first is a logic simulator for behavioral, functional, and timing simulation.
* This tool can catch any design errors.
* The designer provides input waveforms to the simulator and checks to see that the outputs are as expected.
* The wiring delays will be known after physical design (place-and route) is completed.
* The second type of simulators used in FPGA design is a timing-analysis tool.
* A timing analyzer is a static simulator and removes a need for input waveforms.
* Instead the timing analyzer checks for critical paths that limit the speed of operation.
* Designers can set a certain delay restriction on a net or path as a timing constraint; if the actual delay is longer, this is a timing violation.
* In most design systems, designers can return to design entry and tag critical paths with attributes before completing the place-and-route step again.
* In the floor planning step, the blocks of the netlist are arranged on the chip.
* The input of the floor planning tool is a hierarchical netlist that describes the interconnection of the blocks; the logic cells within the blocks; and the logic cell connectors.
* The netlist is a logical description of the ASIC; the floorplan is a physical description of an ASIC.
* Floor planning is thus a mapping between the logical description (the netlist) and the physical description (floorplan).
* The goals of the floor planning are to:
* Arrange the blocks on a chip.
* Decide the location of the I/O pads.
* Decide the location and number of the power pads.
* Decide the type of power distribution.
* Decide the location and type of clock distribution.
* The objectives of floor planning are to minimize the chip area and minimize delay
* Next step is placement which defines the locations of the logic cells within the blocks and sets aside space for interconnect to each logic cell.
* Placement assigns each logic cell to a position in a row.
* Floor planning and placement are closely related and sometimes combined in a single CAD tools.
* The goal of a placement tool is to arrange all the logic cells within the blocks on a chip. Ideally, the objectives of the placement step are to:
* Guarantee that the router can complete the routing step.
* Minimize all the critical net delays.
* Make the chip as dense as possible.
* Minimize power dissipation.
* Minimize cross talk between signals.
* Once the designer has floor planned a chip and the logic cells within the flexible blocks have been placed, it is time to make the connections by routing the chip.
* After detailed routing is complete, the exact length and position of each interconnect for every net is known.
* The parasitic capacitance and resistance associated with each interconnect, via, and contact can be calculated.
* This data is generated by a circuit extraction tool.
* Postlayout simulation checks are performed to see that the design still works with the added loads of the interconnects.

|  |
| --- |
| **Braun**  **A 4-bit Braun multiplier**. |

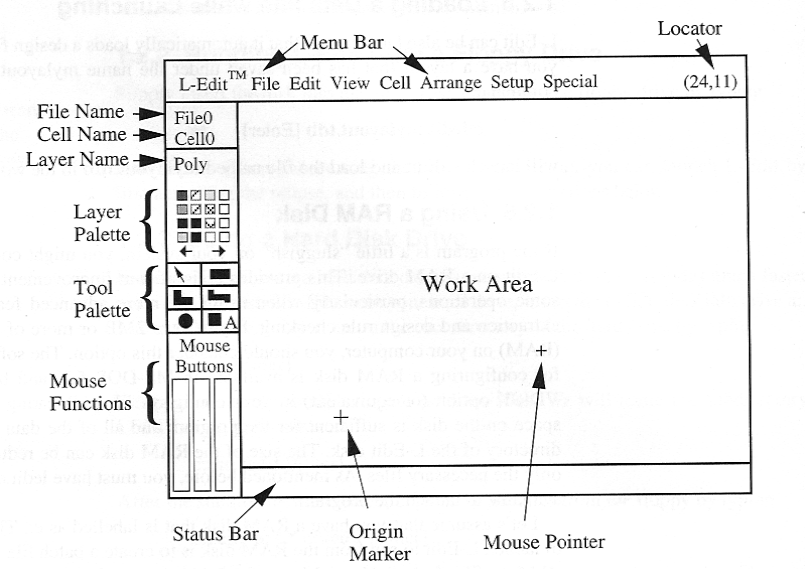
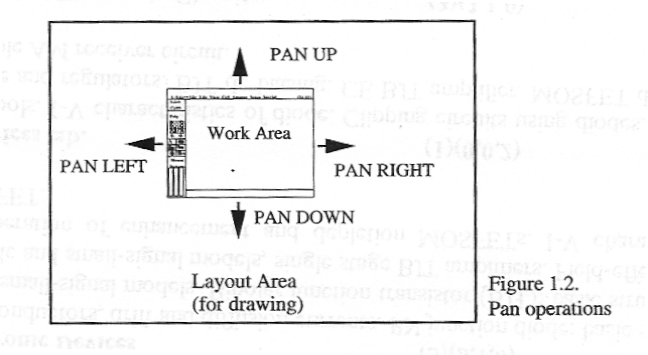
* The **L-Edit** is a very powerful layout editor and is quite easy to learn.
* The L-Edit screen, shown below, consists of several distinct areas. This includes the menu bar, the drawing area, the layer selections, and the mouse functions.

Figure 1.1: The L-Edit screen

* **The Mouse Buttons:** The mouse is used as the primary drawing and control device. L-Edit provides Mouse buttons visual indicator of the mouse button functions in the lower left side of the screen. This is convenient to remember, as the functions vary with the operation.
* **The Work Area:**  It is the portion of the entire drawing that falls within the L-Edit screen. The work area that appears in the L-Edit screen can be adjusted using the Zoom Out and Zoom **In** commands under the View heading; the "-" and "+" keys provide the same function from the keyboard. Device and circuit layouts are usually performed using the Zoom **In** to "magnify" the drawings, while larger sections of the chip can be viewed in a single screen by stepping the zoom out.
* **The Locator:**  Points on the drawing area are defined by a pair of grid coordinates (x,y); the current position of the mouse pointer is shown in the upper right-hand comer within the menu bar. Under normal operation, all objects in L-Edit are referenced to this coordinate system. The origin (0,0) is marked with a cross-hair, but it may not be within the viewing area of the screen.
* **The File Name:** The file name indicator lists the name of the current file that is loaded into L-Edit. When you save your work, the information will be stored on disk using the file name that is shown here.
* **The Cell Name:**  A cell is a basic building block for a VLSI design. A file can contain many cells. The cell name shows the name of the current block that you are editing. Cells can be stored and replicated within the layout, making them very useful for the design of large networks.
* **The Layer Name:** The layer name tells the current layer that has been selected for drawing. All layers can be accessed through the Layer Palette on the left side of the screen.
* **The layer Palette: A** graphical menu of the available layers is given on the left side of the screen. To specify a layer, simply use the mouse to point and click at the appropriate box in the palette. All objects will assume the chosen color until the selection is changed. The name of the chosen layer will appear in the Layer Name line, and also in the status bar at the bottom of the screen.
* **Drawing Tools:**  Predefined tools for drawing polygons and other shapes are shown on the left-hand side of the screen. Point to the desired shape, click to highlight it, then move the pointer to the desired position on the layout area. Pressing the mouse button and holding it down while moving the mouse (dragging the mouse) will draw the object; the size of the object is changed by moving the mouse. Releasing the button sets the final size of the object.
* There are six types of drawing tools in the Tool Palette:

1. Arrow: For pointing and selecting objects
2. Rectangle: Draws rectangular objects
3. Polygon: Allows you to construct general polygons
4. Wire: Acts like a wiring tool with a predefined width
5. Circle: Draws circular objects
6. Port: The port tool is used to define signal entry points, or to apply text labels
   * **The Layout Area:**  All drawing is done on the Layout Area. This can be visualized as a large x-y plane. Points on the Layout Area are defined by the coordinate pair *(x,y).* All geometrical objects created in the Layout Area have vertices defined by sets of coordinates.
   * The Layout Area provides grid points that are useful for drawing objects of a particular size. The grid can be turned on or off using the appropriate command in the View grouping of the Menu Bar (Show Grid or Hide Grid).
   * **Basic** **Navigation:**  The view seen in the Work Area of the L-Edit screen is controlled by two groups of commands, **Pan** and **Zoom:**

- **Pan:** The Pan commands allow you to move the viewing area to the desired location. The basic movements are vertical and horizontal, and are controlled by the arrow keys.

**- Zoom:** The zoom commands provide a method for obtaining "up-close" and "far-away" views of the drawing area. These functions can be accessed from the View group of the menu bar. Alternately, keyboard commands can be used: the " + " key is equivalent to Zoom In (for increased magnification), while the ,,-" key gives Zoom Out action.

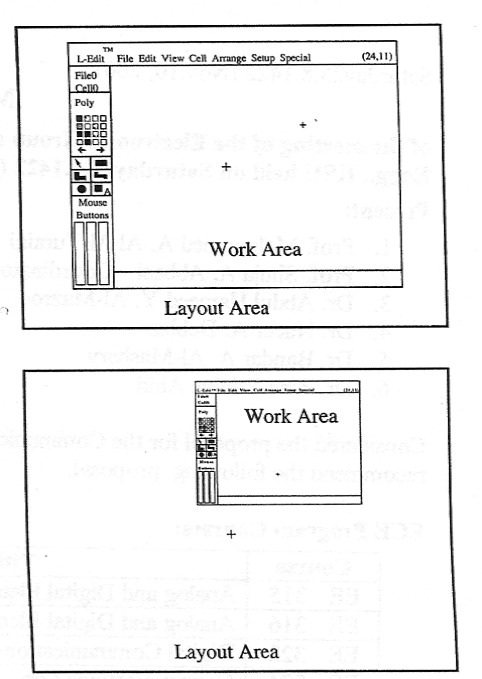


Figure 1.3. The ZOOM operations

* **Non-Manhattan Geometries :** L-Edit also allows you to construct a drawing with nonorthogonal polygons. To access this mode, double-click on the polygon or wire box. The Tool Palette will change icons. The first set that appears are for 45° angles.
* If the tool is clicked another time, L-Edit enters the general polygon drawing mode.
* A third click returns you to the Manhattan (900) geometry mode. Caution must be exercised if non-orthogonal polygons are used in an integrated circuit layout, since the fabrication process may not allow for these types of shapes.
* **The** **Object Editing:** L-Edit has a simple user interface to make object drawing intuitive and easy. Once the objects have been created, the program provides for several editing operations that can be used to modify the properties of objects.
* **Object Selection:**  A single object is selected by the point and click operation. The Mouse Button indicators always tell you the functions that are permitted in the current mode.
* If you point to the interior of an object and click, it will be selected. Or, if the pointer is outside of a group ot:objects, the nearest one win be selected.
* The selected object is identified by having a darker outline than the unselected objects. A group of objects is selected by using the Arrow Tool to drag a box around all of the objects. Every object must be contained within the box to be included with the group; those on the boundary will not be selected.
* Once a group of objects has been selected, you may add to the group by pressing Shift on the keyboard and dragging a box around the new objects. This operation is called Extend Select, and allows you to preserve an existing set while adding new objects to it.
* The Unselect operation is used to remove objects from the group. To effect an Unselect, press the Alt key while dragging a box around the objects that you wish to remove from the group; note that the right mouse button specifies the unselect operation.
* Using the keyboard command /\A selects all objects in the layout. This may also be accomplished using the command in the Edit window of the Menu Bar. The operation **Unselect All** can be performed using either Alt-A from the keyboard, or the Edit Menu entry.
* **Layer Hiding:**  Layers can be hidden using the Hide command.
* Hidden layers are not shown on the screen, and are not included in the select process.
* To hide a layer, point to the layer in the Technology Palette and click the Hide button; all objects on the hidden layer are not shown on the screen, but the information remains stored in the edit buffer.
* If a Select operation is performed when a layer is hidden, the objects will not be contained in the selected group after the layer is made visible again.
* **Moving Objects:** The location of an object may be changed by depressing the Move/Edit button while dragging the mouse pointer to the new location. This places the object at a new set of coordinates.
* To move a single object, place the pointer inside of the object, click the Move/ Edit button, and drag the object to the desired location.
* If you wish to move a group of objects, then you must first perform the Extend Select operation.
* **Changing the Size and Shape:** An object can be resized or reshaped using the edit operations. Simply place the mouse pointer on an edge or a comer, and use the Move/Edit button while dragging the mouse. Releasing the button sets the new size or shape.
* If you wish to add a new vertex to a polygon or a wire, select the tool and then switch it into the **All-angle** mode. Next, position the mouse pointer over the edge of the object where the vertex is to be added.
* **Cutting Objects:** An object (or group of objects) may be cut from the diagram and removed using this command in the Edit window. Just select the object(s) and choose **Cut** from the Edit window, or AX from the keyboard.
* **Copying Objects:**  An object (or group of objects) can be copied using the **Copy** command. You must be in the Draw mode (with a tool chosen) to activate this option. Select the object and then invoke the copy function using **Control** and the **Draw** button on the mouse; or, type /\C from the keyboard.
* **Arrange Commands:**  The Arrange window in the Menu Bar provides several useful commands that change the orientation of an object.
* **Rotate:**  This command rotates the selected object by 90 degrees in a counterclockwise sense about its center point. If this is applied to a group of objects, the rotation is about the center of the group.
* **Flip Horizontal/Flip Vertical:** The Flip commands perform a flip of the selected object about an axis that passes through the center.
* **Cut Vertical/Cut Horizontal:** A Cut command divides an object through an axis that is selected by the mouse pointer.
* **Merge Selections:** This command merges intersecting objects on the same layer into a single object. Just select a group of objects and execute the command. All objects on the same layer that intersect will be merged into single objects.
* **Advanced Features:** The L-Edit includes several advanced functions that place you into a realistic CMOS design environment. These are contained in the Special entry of the menu bar, and are accessed in the standard way.
* **Design Rule Checker:**  Design rules are a set of minimum size and minimum spacing values that should be obeyed to ensure maximum yield in the chip fabrication sequence. The design rule checker (DRC) included with L-Edit will analyze your layout, determine if any design rules have been violated, and report on violations that have been found. Violations can be specified graphically or in a text file format.
* **Circuit Extractor:**  L-Edit provides a circuit extraction algorithm that translates your layout into a SPICE-compatible text file that can be used to simulate the network.
* Elements such as MOSFETs are described by their node connections and geometries using a set of predefined rules for the technology. This allows for direct verification of the layout.
* The circuit extractor can be programmed to include parasitic resistance and capacitance in the devices and interconnects, so that the resulting simulation is directly based on the layout geometry.
* **Cross-Section Viewer:**  This unique feature of L-Edit allows you to see the "side view" of your layout design.
* Cross-sectional views for devices such as MOSFET are drawn according to the "top view" patterns that are drawn in the work area.
* The viewer lets you watch each layer being grown and patterned, and clearly illustrates how the layers stack to form a 3-dimensional integrated circuit structure.