

A CMOS Standard-Cell Library for the PC-based LASI Layout System

Hao Chen and R. Jacob Baker
 Microelectronics Research Center
 The University of Idaho Center in Boise*
 800 Park Blvd., Boise, Idaho, 83712

haochen@ui Boise.engboi.uidaho.edu or jbaker@uidaho.edu

Abstract – A digital standard-cell library using the MOSIS scalable design rules, for use with the LASI layout system, is presented. Both the cell library and the layout system, running on a PC, are described. The cell library (the layouts of the cells) is a translated version of Mississippi State University's standard cell library for use with the Mentor Graphics set of design tools. Design verification using LASI is discussed.

I. Introduction

The LAYout System for Individuals, LASI (pronounced "LAZY") is a PC based integrated circuit design tool

[1,2]. LASI can be used for basic layout, design rule checking and design verification. This PC tool is becoming popular in Universities, especially those with outreach programs, giving students in VLSI design courses an option, other than a workstation, for the layout of integrated circuits. The Windows version LASI system, which comes with a more friendly user interface, is available now [3]. Figure 1 is a layout shown in the Windows LASI system.

A standard cell library [4], based on Mississippi State University's (MSU) standard cell library, for use with LASI that is based on the MOSIS scalable design rules [5], has been established. In this cell library, each

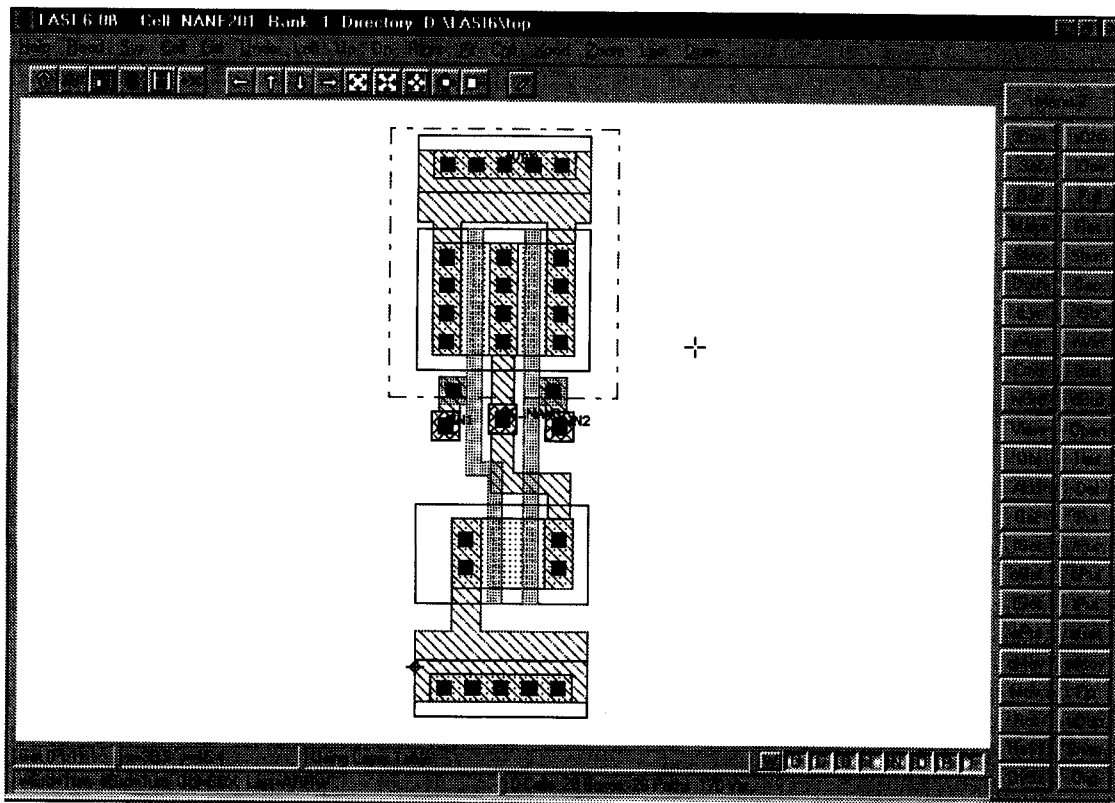


Figure 1 – The layout of a NAND gate shown in the Windows LASI

* This work was supported by Micron Technology

cell includes a schematic that can be used to generate a SPICE netlist and a layout translated from the MSU's standard cell library. These layouts can be very good examples for learning CMOS IC design. In addition, the cells from the cell library can be directly used as sub-circuits in VLSI design.

II. Cell hierarchy

In the LASI system, complex IC designs are made from simpler object cells. A cell might be a logic gate or an op-amp. Each cell is assigned a name and a rank. A cell with a rank of n can contain cells with ranks of $n-1$ or lower.

Using cell hierarchy to design complex ICs can keep the size of the design files from getting too large [2]. More important, cell hierarchy in LASI makes it possible to develop a standard cell library for using these cells in complex IC designs.

III. Schematic generation

The LASI system can be used for schematic generation, as shown in figure 2. Rank one cells, in the schematic of the inverter (Fig. 2(a)), are the basic building blocks, i.e., the NMOS (Fig. 2(b)) and PMOS symbols. More basic schematic cells can be found in [6].

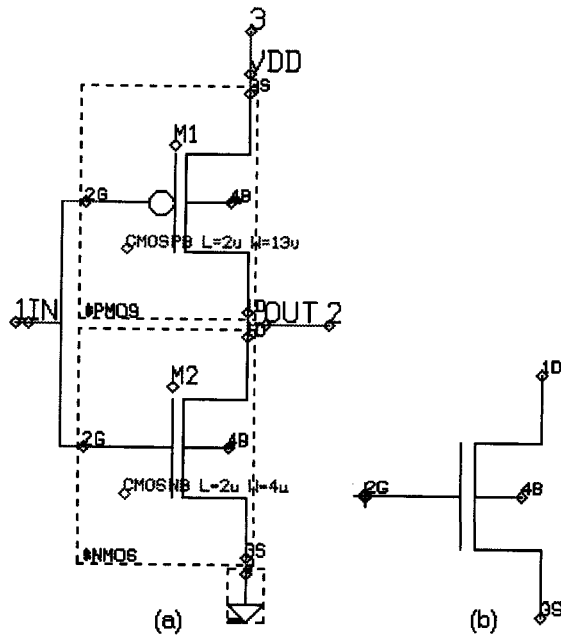


Figure 2 – (a) LASI Inverter schematic (b) \$NMOS

The actual symbol of these basic components was drawn on the schematic layer using a zero-width path or polygon. Rank two or higher cells may be inverters, logic gates or flip-flops to name a few cells. For example, the inverter in figure 2(a) is a rank two cell that is made up of one NMOS cell, one PMOS cell, one

ground cell and some wires (drawn on the MET1 layer using a zero-width path or polygon)[2]. To distinguish between a layout and a schematic in the LASI cell library, a \$ is used as the first letter of a cell name to indicate a schematic cell. The NMOS and PMOS cells in figure 2(a) are called \$NMOS and \$PMOS.

We have generated schematics for all cells in the MSU cell library using LASI, and all of these schematics have ranks of 2.

IV. SPICE netlist generation

From the schematic or layout drawings a LASI utility called LASICKT can be run to generate a SPICE netlist.

LASICKT works by labeling a drawing with text on specific layers at specific locations. In a LASI drawing, the circuit devices are simply cells interconnected with lines or areas. To mark how circuits are connected we use "Node name" and "Connector". A connector locates the "connection point" on a cell and passes that location to higher ranking cells where that particular cell is being connected as part of a larger circuit.

Figure 2(b) shows a basic schematic - \$NMOS. In this rank 1 cell, "1D, 2G, 3S and 4B" are connectors. In figure 2(a), "IN" is a node name for the input of the inverter, and it is connected to "2G" of the \$NMOS cell. Like "2G" in the \$NMOS, the connector "1" in the inverter is connected to the higher rank cell.

The SPICE code also requires a name and certain parameters for each device. A "device" must be a cell. The lowest ranking device is a rank 1 cell. To indicate device names and parameters, two other text layers – "Device Name" and "Device Parameter" – are used. In figure 2(a), "M2" is the device name of the NMOS transistor that is drawn on the DTXT layer, and "CMOSNB L=2u W=4u" is the parameter of the N-channel MOSFET that is drawn on the PTXT layer.

The netlist generated from the schematic shown in figure 2(a) is:

```
*MAIN CIRCUIT $INVF101
M2 OUT IN 0 0 CMOSNB L=2u W=4u
M1 OUT IN VDD VDD CMOSPBL=2u W=13u
.END
```

Notice that the text is literally passed on to the SPICE netlist.

After running LASICKT, a circuit file (*.CIR) will be generated, which includes the netlist and two other files – Header and Footer files. LASICKT can pass these two files directly into the circuit file without any change. The header file will be inserted at the beginning of the circuit file (*.CIR). This file generally contains information such as power supply voltages and input sources. The footer file will be appended to the end of the circuit file. This file generally contains the SPICE models and can be common to all cells in the drawing directory. These two files are completely arbitrary and may be omitted by simply putting in a name of a file

that doesn't exist or leaving the name blank by entering only a space character.

The procedure to generate a SPICE netlist from LASI layouts is nearly the same as that of the schematic. The only difference is the fill of the interconnect layers in the setup menu of the LASICKT program. More information about how to generate a SPICE netlist can be found in [2].

In the standard cell library, every schematic has been properly labeled with text, every circuit file generated from them has been simulated, and the simulation results are correct. One can generate netlists directly from these cells or connect these cells to form a hierarchical design.

V. Layout of the standard cell library

Another part of the cell library is the layouts that are streamed from the layouts in the MSU cell library and converted into internal LASI Transportable LASI Cells (files with *.TLC extension). In addition, we have flattened the layouts of all cells and made them rank 1 to keep the number of cells in the LASI cell library small.

LASI drawings are not very useful if they cannot be converted to a commonly used CAD format for mask generation or for transferring into another CAD system. The most common format is formally called Calma Stream Format, which is also known as GDSII (Graphic Design System II), or just GDS. LASI has a drawing structure that is very similar to GDS, with the exception that LASI has box objects and is more limited in its data capacity and cell nesting depth. Since there can be many TLC files, a GDS file with almost any number of structures can be converted to TLC files. On the other hand, LASI can be converted to GDS format exactly.

When we converted the GDS file to the LASI TLC

file from the MSU cell library, too many different cells were generated. Most of these cells are basic drawings such as a standard contact connection between metal 1 and poly 1, and then there is no device correspondence between the schematic cells (i.e., a MOSFET) and the layout cells (i.e., a contact). No additional benefits gained by this lack of correspondence, so we have smashed all of the layouts to make them rank 1. We also redrew part of the objects in the layouts to make these layouts clear. Figure 3 shows one of the layouts from the cell library, which is a D flip-flop with Set and Reset.

One big difference between the layouts of the cell library from the schematics is that all of the layouts have minimum device ranks (rank 1) rather than rank 2. The MOSFETs contained in the layout are not "devices", and we cannot label them with device names and device parameters. So, at the present time, we cannot generate effective netlists directly from the layouts of the standard cell library.

VI. Design verification

An important step in the design process is verifying that the layout of an integrated circuit matches the schematic of the IC. As mentioned before, if an IC is designed by using cell hierarchy, from both the layout and the schematic, we should be able to generate a SPICE netlist and simulate the operation of the IC. A comparison between these two netlists as well as the resulting SPICE outputs can be used to verify the schematic and layout match.

In our cell library, we cannot make the comparison between the schematic and the layout, because we cannot generate netlists directly from the layouts. This does not lower the functionality of the cell library, because we can still make the comparison between the

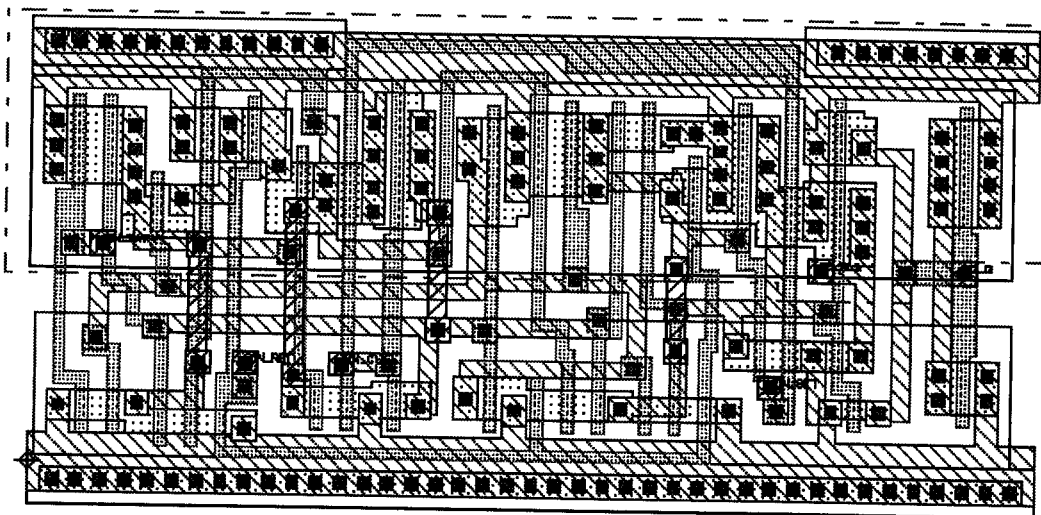


Figure 3 – A DFF

schematic and the layout of a hierarchical design that uses these cells.

In the cell library, every layout has been checked very carefully. Many of them have been fabricated to show they function correctly. If we generate a netlist from a layout in the cell library, this netlist should be the same as the netlist that was generated from the corresponding schematic in the cell library. Then we can insert the netlist generated from the schematic of a cell into the netlist of a complex circuit layout containing the layout of the cell, and make the comparison between the netlists generated from the schematic and the layout of the complex IC design.

Figures 4 & 5 show the schematic and layout of a one input switch with active low enable. This switch uses the \$INVF101 cell shown in Fig. 1. In figure 4, the rank 2 cell \$INVF101 includes two rank 1 cells M1 and M2. The netlist generated from this schematic is:

```
.SUBCKT $INVF101 IN OUT VDD
M1 OUT IN VDD VDD CMOSPB L=2u W=13u
M2 OUT IN 0 0 CMOSNB L=2u W=4u
.ENDS
*MAIN CIRCUIT $SWITCH
X$INVF101 IN_CTL VN1 VDD $INVF101
M1 IN_DATA IN_CTL OUT VDD CMOSPB L=2u W=14u
M2 IN_DATA VN1 OUT 0 CMOSNB L=2u W=8u
.END
```

In figure 5, the INVF101 is a rank 1 cell that can be labeled with text, and the MOSFETs in the inverter cell are not cells and cannot be labeled. Cell INVF101 is labeled with device name "XINVF101" and device parameter "\$INVF101" (same as that of the schematic). The netlist generated from this layout is:

```
*MAIN CIRCUIT SWITCH
XINVF101 IN_CTL VN1 VDD $INVF101
M1 IN_DATA IN_CTL OUT VDD CMOSPB L=2u W=14u
M2 IN_DATA VN1 OUT 0 CMOSNB L=2u W=8u
.END
```

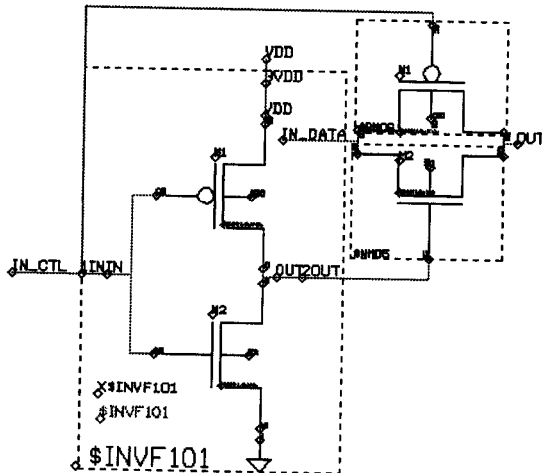


Figure 4 – The schematic of a switch

The only thing missing in this netlist is the sub-circuit description of the \$INVF101. By simply copying

and pasting the sub-circuit description into the header or footer file of the LASICKT setup menu when the netlist is generated from the layout, the sub-circuit description can be added into the circuit file. Then these two circuit files, as well as the SPICE simulation results, can be compared.

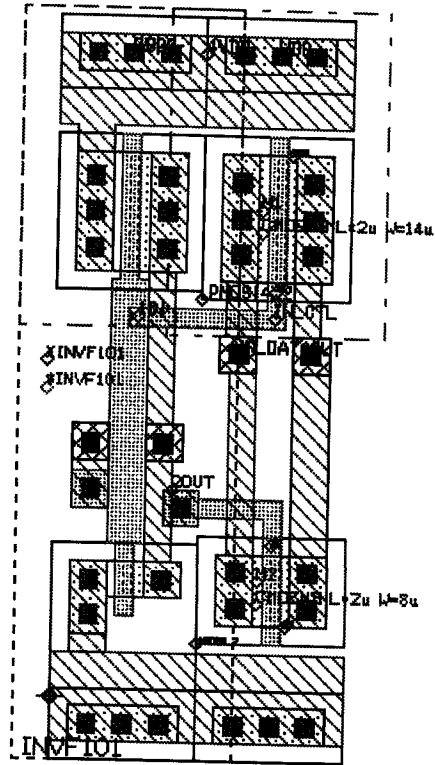


Figure 5 – The layout of the switch

VII. Conclusion

We have discussed a standard cell library for use with the LASI layout system. The combination of LASI with the cell library can be very effective tools in teaching custom IC design.

References

- [1] Boyce, D.E., and Baker, R.J. "A Complete Layout System for the PC", Proceedings of the 40th Midwest Symposium on Circuits and Systems, pp. 1091-1094.
- [2] Baker, R.J., Li, H.W., and Boyce, D.E. *CMOS: Circuit Design, Layout and Simulation*, IEEE Press 1998.
- [3] <http://www.engboi.uidaho.edu/lan-group/jbaker/wwwbook/winlasi/winlasi.htm>
- [4] <http://www.engboi.uidaho.edu/uioboise/mrcboise/clibrary/cindex.htm>
- [5] <http://www.mosis.org>
- [6] <http://www.engboi.uidaho.edu/uioboise/mrcboise/projects/scells.htm>