論文 84-21-5-10

Design of CMOS PLA Using C Language

(C 언어를 이용한 CMOS PLA의 설계)

車 均 鉉*, 케빈·카플러스**

(Kyun Hyon Tchach and Kevin Karplus)

要 約

C 언어로 만든 VLSI 레이아웃 언어를 사용하여 CMOS PLA를 설계한다. PLA cell의 library를 만들고 protector 회로의 제어논리로 사용되는 PLA를 NCR 설계법칙을 이용하여 설계하고 레이아웃 프로그램을 만든다. 관련되는 프로그램 기법을 논의하고 레이아웃을 display할 수 있도록 한다.

Abstract

In this paper a custom design of CMOS PLA using procedual language, CHISEL is presented. Library of cells of PLA pieces are formed. A typical PLA is used as a control logic for the protector circuit. NCR's design rules are applied to program CMOS PLA using CHISEL which is a VILI layout language made by extending C language.

I. Introduction

A PLA (programmable logic array) is a regular structure for imlementing sequential logic functions. Because of the compactness and ease of design change and check, PLAs are used in the control logic of microprocessor chips and calculators. Also PLAs are used for code conversion, microprogram, address conversion, decesion tables, bus priority resolvers and memory overlay.

There are quite a few PLA generators implemented with bipolar, NMOS and CMOS. Among these CMOS had certain advantages, which are extremly low power consumption and immunity from power supply voltage fluctuatuation, noise and temperature change.

The PLA implemented with 4-micron CMOS cell library uses NAND logic in both AND and OR planes. This form of logic has been chosen as it is operated by 2 phase clock.

The challenge of VLSI is packing a lot circuitry into a small area, but designing large systems and automating the construction of chips from the specification are important as well. Many languages for VLSI layout can be roughly devided into four classes; interactive, symbolic, declarative, procedual.

Interactive systems are oriented toward manual layout techniques providing the user with an automated pencil.

Symbolic layout systems provide a semi-automatic approach where a designer draws objects that are abstracted somewhat from the rectangles that are actually laid out.

Declarative languages provide for description of chips with little or no explicit computation,

*正會員, 高麗大學校 工科大學 電子工學科
(Dept. of Electronics Eng., Korea Univ.)
**코넬대학교 전기과
(Dept. of Electrical Eng., Cornell Univ.)
接受日字 : 1984年 1月 16日

though some require considerable implicit computation.

Procedural languages are for writing programs that include explicit computation of sizes and locations of objects, usually by embeding geometric primitive in an existing high level languages.

CHISEL is a procedural language that allows to use cells created their own cells as well as cells created graphics by editors. CHISEL is made from verstile and flexible language C so that it considered generality and flexibllity more than terness and built-in algorithm. CHISEL has two files. One is CIF and the other is headerfile. Since CIF is a standard, there are many tools using as an input or output language (plotters, design rule checkers, circuit extracfors, PLA generators).

However, it is inadequate for representing much of the information needed to put the pieces of a design together. For example, it provides no mechanism for naming cells and connection points, nor does it have any way to specify replication distances.

On this paper, CIF is purely used for geometric information and an associated header file for all other information needed. Therefore, the CIF is only needed for making masks and checkplots, doing design rule ckecking and similar geometry-dependent tasks. The header file provide all the information such as size of the cell, connection information, etc, needed to use the cells to construct bigger cells and chips. In this paper programming with CHISEL for the CMOS PLA is also discussed.

II. Floor Plan and Layout

Fig. 1 shows the block diagram of a PLA macro block. The input for the AND array are generated automatically. The output of OR plane can also be generated automatically. A library of CMOS cells which forms the buiding blocks for the PLA consist of eleven piece of cells which are in the header file. Fig. 2 are the examples of CMOS PLA pieces in cell library. Cells in header file are used in Fig. 4. By designing inverter and transmission gate as shown in Fig. 2. Chip area can be minimized and design flexibility is obtained. Fig. 3 is the stick diagram for the inverter and transmission gate respectively.

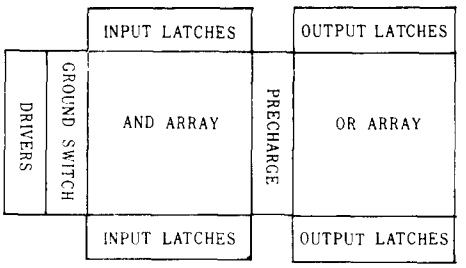

Fig. 1. The macro block of CMOS PLA.

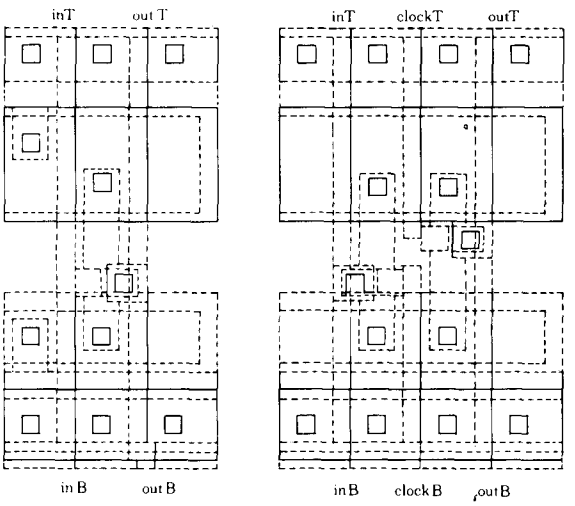

Fig. 2. Examples of cells in cell library.
(a) An inverter (b) A transmission gate.

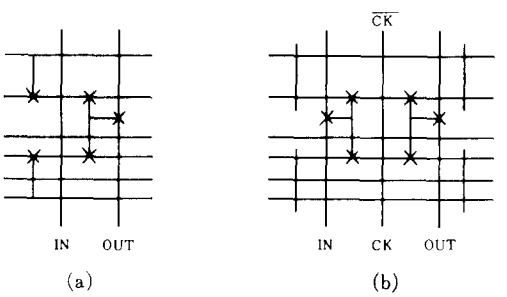

Fig. 3. The stick diagram.
(a) An inverter (b) A transmission gate.

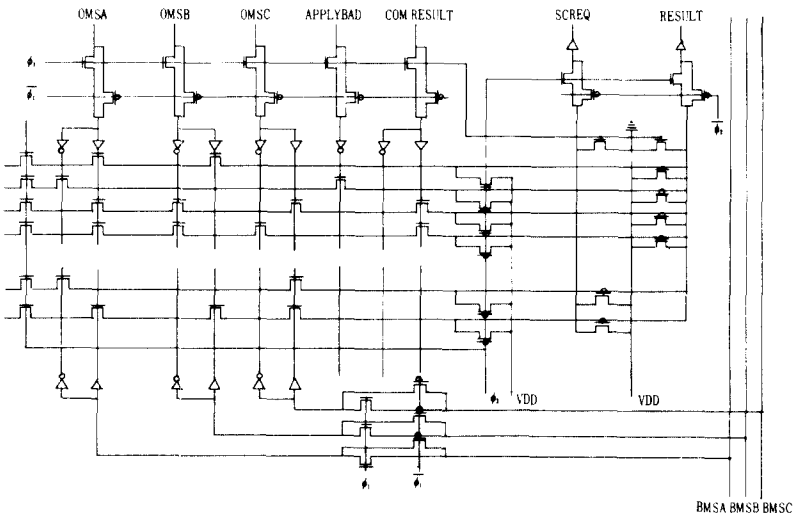

Fig. 4. A PLA circuit for the protector circuit.

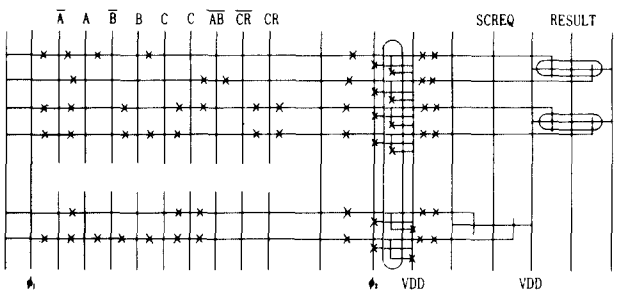

Fig. 5. The stick diagram of the circuit in Fig. 4.

The transmission gate is used to switch the signal. Fig. 4 is the PLA circuit to use in protector circuit and their stick diagram is shown in Fig. 5. Each cell placed so that chip area is as square as possible.

III. Software

Software consist of 2 files. One is for the CMOS PLA pieces and the other is for the layout of CMOS PLA. If input is given, the program automatically generate CMOS PLA.

The main program consist of 23 procedures. The essence of this program is to read input and arranging terms so that any arbitrary input can put into the AND or OR array without violating design rules. Table 1 shows an example of inputs for the PLA shown in Fig. 4 "1" means input in term, "0" means input bar in terms and any other character means input doesn't affect term. Thus, high and low input are expressed in a column. Each column of Table 1 separates into left column and right column. "1" goes to left column and "0" to right column. Since inputs come from both top and bottom, these two input are combined into single column to save chip area. In order to combine both input, there should be at least one empty space on the term not to violate design rule between transistor from the top input and that of from the bottom. The left and right inputs can be interchanged for input combination not to be violated design rules.

The following examples show how to combine. Since both "1" and "0" represent transister on the term, they are represented by "+" in the combined input. Connection between transistors on each input on the layout are made by poly line and "p" represent this ploy line.

Note that marking *on the 4th and 5th row of the combined input do violate design rule because there is no empty space between transistor from top and from bottom. After interchanging the left and right column of the bottom input, we can find that there is no design rule violations. However, this inter-

Example 1.

top input	bottom input	input separation		combined input	
		left	right	left	right
1	●	1	●	+	P
0	●	●	0	p	+
1	●	1	●	+	
●	●	●	●		
●	0	●	0		+
●	1	1	●	+	p

The combined inputs of example 1 do not violate design rule because there are empty spaces between transistors.

Example 2.

top input	bottom input	input seperation		combined input		after interchanging left and right column of the bottom input	
		left	right	left	right		
1	●	1	●	+	p	+	P
1	●	1	●	+	p	+	P
0	●	●	0	p	+	p	+
1	●	*1	●	*+		+	
●	1	*1	●	*+			+
●	0	●	0	p	+	+	P

change of the bottom input must be taken into account on the routing.

If there is the case which the top and bottom input can not be combined in any other means, the top input or bottom input need extra spaces on the layout as shown in Fig. 6.

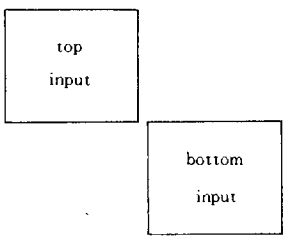

Fig. 6. The top input or the bottom input need extra space when input combination is not possible.

The over all algorithm to make new terms by combining inputs is as follow (Fig. 7).

These combination can be done by 4 procedures. FillCol ( ) fill column with "+" for transistor and with "p" for poly bar. The poly bar ends just before the last transistor from top or bottom. Procedure FindSpace ( ) finds the position of the space of after (in case of top input) or before (in case of bottom input) the last occurance of character and returns the subscript for position which must be empty in order for character to be in this column.

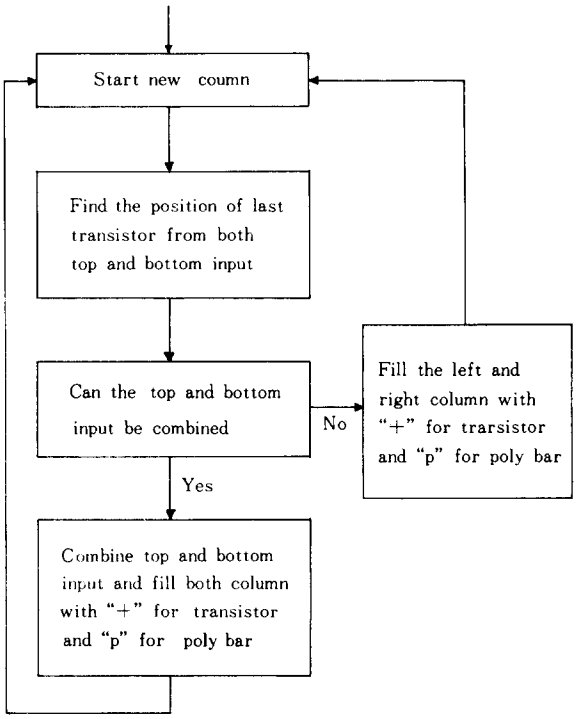

Fig. 7. Algorithm making new terms from the input.

Table 2 shows the new terms which is obtained by above algorithm, and is used to make "and cells" in AND plane. DrawLine ( ) draws line between two point so that metal line and poly line can be drawn using this prodecure. Since layout can be displayed on the CRT teminal, designer are able to debug them if there are any design rule violations.

The language used is CHISEL which is extending C to design VLSI. The sample program for one of a cell in a header file is as follow.

Table 1. An example of input arrays.

input / term	A1	B1	A2	B2	A3	B3	A4	A5
	1	•	1	•	•	•	•	•
	0	•	•	•	•	•	0	•
	1	•	0	•	1	•	•	1
	1	•	0	•	0	•	•	1
	•	0	•	•	•	1	•	•
	•	1	•	1	•	1	•	•

A'S are inputs from top and B'S are those from bottom

Table 2. New terms made from table 1.

column / term	L	R	L	R	L	R	R	L
	+		+	p	p	p	p	p
	p			p	p	p	+	p
	+			+	+	p		+
	+			+		+		+
		+			+			
	+	p	+		+			

```
Cell "nmos" ;
  Layer (SILI CON);
    Rect I (-14, -6, 14, 6);
  Layer (POLY); WireWidth=4;
    DefinePort "Pot" 0, 10    dir +y ;
    Cwire (At, X, -10) ;
    DefinePort "Pob" dir -y ;
Put "mctap A" @ [[-14, 0]]    rot Rot 0;
Put "mctap A" @ [[14, 0]]     rot Mirror X ;
WireWidth =8 ;
DefinePort "mtL" @ [[-14, 0]] dir -x layer
                              METAL;
DefinePort "mtR" @ [[14, 0]]  dir +x layer
                              METAL ;
REPLBOUNDS (OpenCell)=Box(-14, -10, 14, 10);
EndCell "nmos" ;
```

The description of procedures used in the program for CMOS PLA generators is in appendix.

IV. Conclusion

1) The CMOS PLA is designed without using graphic terminal.
2) This design method greatly enhance the creative ability of the disigner, and designer can throughly understand the VLSI layout.

Appendix

Name of procedures and their functions.

build-PLA (name) Build PLA with And-Plane ( ) and Or-Plane ( ) reding inputs, terms and outputs.

And-Plane ( ) Build the and plane (input, cells)

Or-Plane ( ) Build the or plane

read-input ( ), read-outputs ( ), read-terms ( ), inputs ( )

and-cells ( ) Create cells in And-Plane

Draw Line (Ptend, Ptbegin, i), Fill Coll (incol, outcol, LorR, LorR, Ch)

FillBoth (incol, outcol, Ch1, Ch2)

New-terms ( ), RGrowCell (b)

Numcat (S,N) Return F (string with S followed by the decimal representation of n)

ModRepl B ( ) Modify the REPLBOUNDS of OpenCell to grow as required by the most recent "put" command

PorttoLine (Pt, lcoord) Draw wries from the ports Pt, to the line with one coordinate lcoord

Xvalue (Pt, Cl, Sub) Find the sub'th Port name Pt in cell Cl, and returns its x coordinate

Yvalue (Pt, Cl, Sub) Find the sub'th Port named Pt in cell Cl, and return to y coordinate

ParseOptions (word)

PlaceNmos (LorR, Xat, in, t, npot, npot, trcount)

PlaceTgate (TorBIn, TorBCl, TorBCl, TorBL-Invt, inport, vddO, gndO, count)

PlaceInvt (TorBIn, TorBLInvt, TorBCl, inport, vddO, gndO, count)

The procedures which are missing description are explained in detail on the program.

References

[1] Brian W. Kernighan, Kennis M. Ritchie, *The C Programming Language*, Prentice Hall, Inc., 1978.

[2] Kevin Karplus, *CHISEL An Extension to the Programming Langueage C for VLSI Layout.* Ph.D. thesis, Department of Computer Science, Stanford University.

[3] Carver Mead and Lynn Conway, *Introduction to VLSI Systems.* Addision Wesley, Inc., 1980.

[4] Saburo Muroga, *VLSI System Design. John Wiley & Sons, Inc., 1982.*

[5] American Micro System, Inc., *MOS Integrated Circuits.* Van Nostrand Reinhold Company, 1972.

[6] H-F.S. Law and M. Shoji, *PLA Design for the BELIMAC-32A Micro-processor.* Proceedings of the 1982 International Conference on Circuit and Computers.

[7] Richard J. Lipton, et al., *ALI: a Procedural to Describe VLSI Layouts.* 19th Design Automation Conference, 1982.

[8] Richard F. Lyon, "Simplified design rules for VSLI layouts", *Lamda (Now VSLI Design)*, vol. 2, no. 1, First Quarter, 1981.

[9] Thomas W. Griswold, *Portable Design Rules for Bulk CMOS VLSI Design.* Sept. Oct. 1982.