

CMORGEN : A CMOS REGULAR STRUCTURES GENERATOR

H. DJEMAH, H. HADJ MOUSSA, A. MAHDOUM

Microelectronics Laboratory, CDTA

128 Chemin Mohamed GACEM BP245 El Madania 16075 Algiers ALGERIA

Tel.: (213) (02) 27-68-68 Fax: 27-93-93 Email: mahdoum@usa.net

Abstract:

The VLSI system design is time and money expensive. Therefore, fast and error-free approaches are required. Today, standard cells and silicon compilation are the most promising methodologies for VLSI system design.

In this paper, we present CMORGEN, a tool that efficiently produces layouts of parameterized cells concerning shifter arrays, memory address decoders, and PLAs. This work is a twofold purpose. It is intended to either design library cells, rapidly, or to serve as a tool in a silicon compilation environment. Indeed, CMORGEN has been able to produce layouts for large circuits in a reasonable CPU time.

Key-words: VLSI System Design- Automatic Layout Generation- Parameterized Cells- Packing Tiles Technique- Regular CMOS Structures

I. INTRODUCTION

VLSI system design has involved three main approaches: full-custom, gate arrays and standard cells. Full-custom approach has the benefit to produce efficient integrated circuits, but at a high expense of time and money. Thus, only a high production of prototypes can render such an approach profitable. Gate array approach delivers pre-metallized circuit solution which allows the designer to connect his interested functions so that the circuit will perform the desired behavior. Although this way yields a fast circuit design, the main drawback is that one deal with an area consuming technique since the functions that do not concern the interested behavior are also present into the circuit. With standard cell approach, an integrated circuit is designed using library cells (leaf cells). Such an approach tends toward faster circuit design without an increasing amount of area, which make it more interesting than the two other ones. A most promising methodology for integrated system design has emerged since a few years. It is a top-down system design where many works are investigated at the system level for system synthesis purpose. This methodology is termed "silicon compilation" and addresses user-specified constraints concerning area, speed, and/or power dissipation. Both silicon compilation and standard cell approaches feature the use of already designed cells. For each of these cells there exist many instances presenting different electrical characteristics so that appropriate tools (e.g. mapping tools) can combine these cells in order to meet the user requirements. A structure generator may help us to either map directly the generated structure into the circuit under design, or to constitute a library cells, rapidly. Such cells can also serve to build macro-cells, which in turn become leaf cells. Unfortunately, one need complex techniques such as expert systems to produce efficient cells in terms of area, speed, and/or power consumption. However, there exist particular cells that can be efficiently produced using simpler techniques. These cells are termed "regular structures" and address implementations of PLAs, ROMs, RAMs, decoders, shifters,....

Techniques other than expert systems are also in use for producing layouts, automatically. They are procedural [3], graphical [4] and packing tiles techniques [5]. One seldom use the procedural technique because it involves a long code even for describing a simple circuit. The graphical one could be interesting for small cells when it becomes fastidious for large ones. Expert systems [6] have the benefit to cope with complex structures. However, simpler structures could be efficiently (in terms of area, speed, and/or power consumption) and easily produced with the packing tiles technique. With this technique, the leaf cells (tiles) are designed using the graphical technique whereas the procedural one is only used to place and route the leaf cells. Many works are still investigated in layout synthesis, targeting area minimization, fast circuits, and/or low power consuming circuits [9 -15].

Our leaf cells have been designed using a graphical editor, a design rule checker, and a circuit extractor, all of them incorporated in MAGIC [7]. Another tool, SPICE, has been used for the electrical simulation purpose at both transistor level and physical design level. CMORGEN is written in C language, and is currently able to produce **parameterized** cells concerning shifter arrays, memory-address decoders, and PLAs in an interesting CPU time. This paper is organized as follows. Section II stands for some electrical problems. In the same section, we give the solutions which have helped us to cope with such problems, and show how the leaf cells are chosen, then designed. Follows then section III in which an overview, the data structures, and the algorithm implementing CMORGEN are depicted. Experimental results for various instances of the three types of circuits are shown in section IV. Section V concludes the paper.

II. ELECTRICAL DESIGN LEVEL

2.1. I_{DS} in a MOS device transistor:

Let us briefly recall the current conduction in a MOS transistor channel. The drain-source current in an NMOS channel is given by:

$$I_{DS} = 0 ; \quad V_{gs} - V_t \leq 0 \quad (\text{cut-off}) \quad (2.1)$$

$$I_{DS} = \beta \left[(V_{gs} - V_t) * V_{DS} - \frac{V_{DS}^2}{2} \right] ; 0 < V_{DS} < V_{gs} - V_t \quad (\text{linear}) \quad (2.2)$$

$$I_{DS} = \frac{\beta}{2} (V_{gs} - V_t)^2 ; 0 < V_{gs} - V_t < V_{ds} \quad (\text{saturation}) \quad (2.3)$$

V_{gs} is the gate-source voltage

V_{ds} is the drain-source voltage

$$V_t = V_{t0} + \lambda V_{DS} ;$$

V_{t0} is the initial threshold voltage

λ is a parameter defined by the substrate effect

$$\beta = \mu C_{ox} W / L$$

μ is the electron mobility

W and L are the size of the transistor channel

C_{ox} is the oxide capacitance

2.2. Area, speed and power consumption in MOS circuits:

2.2.1. Area:

The area at the register transfer level is a function of the numbers of buses, registers, and functional units that constitute the data path entity. The control unit area depends on the numbers of primary input and output symbols, variable states, and product terms.

At the physical level, the circuit area is function of the size of the transistor devices, interconnection area, power supply and ground lines, source and diffusion regions, and extra components (contacts, vias, ...).

2.2.2. Speed:

The above formulae of I_{DS} clearly show that the drain-source current is dependent on W and L , the wide and length of the transistor device, respectively. In other words, fast circuits need large transistors. Note, however, that the circuit speed also depends on parasitic resistances and capacitances (intrinsic, parallel and overlap capacitances), and on the style of logic design. Note also that for the same values of W and L , an NMOS transistor device is nearly three times faster than the PMOS one because of the values of μ_N and μ_P . Finally, the system speed is function of the number of functional units that perform simultaneously. Many works have been investigated for high speed VLSI system design. The interested reader may found advanced techniques developed in many papers. The low power system design is a more recent topic.

2.2.3. Switching power dissipation in CMOS circuits:

A power dissipation in CMOS circuits involves three main components. They are leakage currents, short circuits, and switching. The most relevant in CMOS circuits is the third one. The switching dissipation in those circuits is greatly dependent on the style of logic design, power supply V_{DD} , frequency, and load capacitances. Assuming that a switching power dissipation in a CMOS gate is given by:

$$P_g = 0.5 * V_{DD}^2 * f * C_g \quad (2.4)$$

the switching dissipation in a CMOS circuit is:

$$P = 0.5 * V_{DD}^2 * f * \max_{(V0, Vt) \in (B^n, B^n)} \sum_{i=1}^{\# \text{gates}} C_{gi} * N_{gi} \quad (2.5),$$

where N_{gi} is the number of transitions gate g_i could undergo. For a guidance of purpose, we give from [8] the maximal transition number a CMOS circuit could undergo for some styles of logic design:

- Pass transistors-free static circuits:

$$N_{\max} = \sum_{i=1}^S (S_i * Nb_i) \quad (2.6)$$

- Cascade stages interleaved by pass transistors:

$$N_{\max} = Nb_1 + 2 \sum_{i=2}^S Nb_i \quad (2.7)$$

- Static circuits whose end gates are precharged:

$$N_{\max} = N_S + 1 + \sum_{i=1}^{\# \text{gates}} \frac{Q_i}{q_i} \quad (2.8)$$

- 4-Phase clock dynamic circuits:

$$N_{\max} = 2|G| + \frac{1}{2} \sum_{i=1}^S Nb_i (S - i) \quad (2.9)$$

where

$S_i = 1$ in case no gate g_i feeds g_i

$S_i = \max(S_k) + 1$; g_{kl} feeds g_i

$Nb_i = |G_i|$; $G_i = \{\text{gates } g_k ; S_k = S_i ; k=1,2,\dots,|G_i|\}$

$N_S = |G|$; $G_i = \{\text{gates } g_k ; S_k = \max(S_i) ; i=1,2,\dots\}$

q_i is the propagation delay of gate g_i

q_{ik} is the propagation delay of gate g_j such that:

. $S_j = k$

. g_j belongs to the path including gates involved

in the determination of the output value of gate g_i

$$Q_i = \sum_{k=1}^{S_i} \max(q_{jk}) \quad ; 1 \leq j \leq |G_k|$$

$G = \{\text{gates } g \text{ belonging to the circuit}\}$

Note that although the above transition numbers are bound values, they could serve to select the style of logic design that matches the user-constraints in terms of speed, area and power consumption. Techniques are also developed in [8] to output near-optimal solutions for the NP_complete problem stated by eq.(2.5)

Finally, note that a circuit design subject to area, speed and power dissipation is a combinational problem and could be solved by either involving techniques of programming (e.g. integer linear programming) for small circuits (actual circuit designs are often computationally intensive), or by appropriate heuristics.

2.3. Leaf cell design:

In 2.2, we have made a brief survey of the main parameters that are involved in a VLSI system design. A promising way to produce efficient circuits is to design, for each cell, many instances with various values of speed, area, and power dissipation. Then, an appropriate tool (e.g. a mapping tool) can perform combinations among the different leaf cells so that the VLSI system will meet the user-requirements. Now, we will show how the leaf cells of the shifter array, memory address decoder and PLA have been designed. We will also discuss the styles of logic design for each of the three structures.

2.3.1. Shifter array:

Shifter arrays are frequently used in many VLSI systems. In Fig.2.1 logical and transistor level of this structure are depicted. Electrical simulations are first made on the leaf cells Cell1 and Cell2, using SPICE tool. We resize the transistor devices until the desired propagate delays are obtained. Then, the leaf cells are physically designed, using a graphical editor and a design rule checker that are both incorporated in MAGIC. The physical circuit is then extracted and electrically simulated to make sure the circuit responses within the fixed delay. The MAGIC files corresponding to Cell1 and Cell2 are then used by CMORGEN to output the MAGIC file that stands to the whole circuit that is error-free design, and could be viewed using MAGIC graphical editor. Shifter arrays CMORGEN generates feature the following:

- a mirroring technique is used (see the second line in Fig.2.2) so that the clock, V_{SS} and V_{DD} lines could be shared. Without such a technique, the shifter array that is depicted in Fig.2.2 would have required 3 lines for each of ϕ_1 , ϕ_1 , ϕ_2 , ϕ_2 , V_{SS} and V_{DD} , namely 18 lines. In Fig.2.2, there are only 12.
- In general, more the size of the shifter array is high, more the gain in lines is.
- there are less contacts since the number of V_{SS} , V_{DD} and clock lines is reduced, which yields less area.
- because metal2 resistivity is better than the one of metal1, CMORGEN uses metal2 for interconnections whenever it is possible.

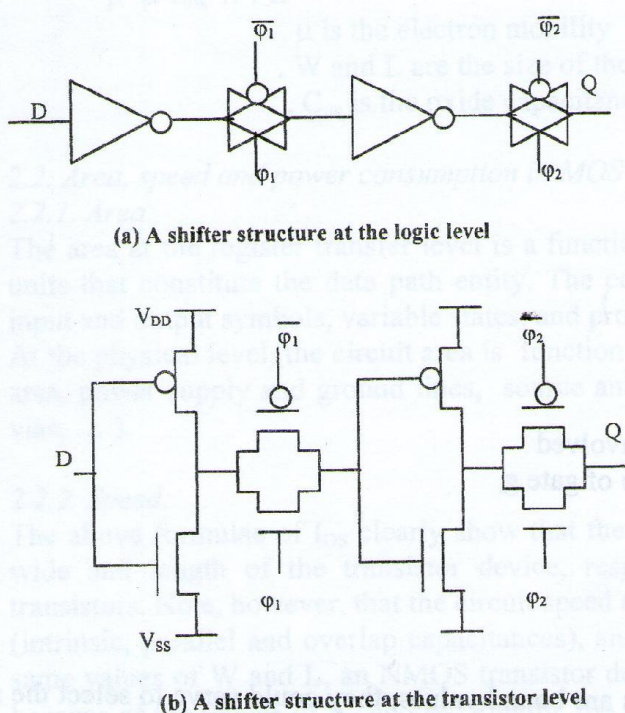


Fig.2.1. A 1*1 shifter array structure

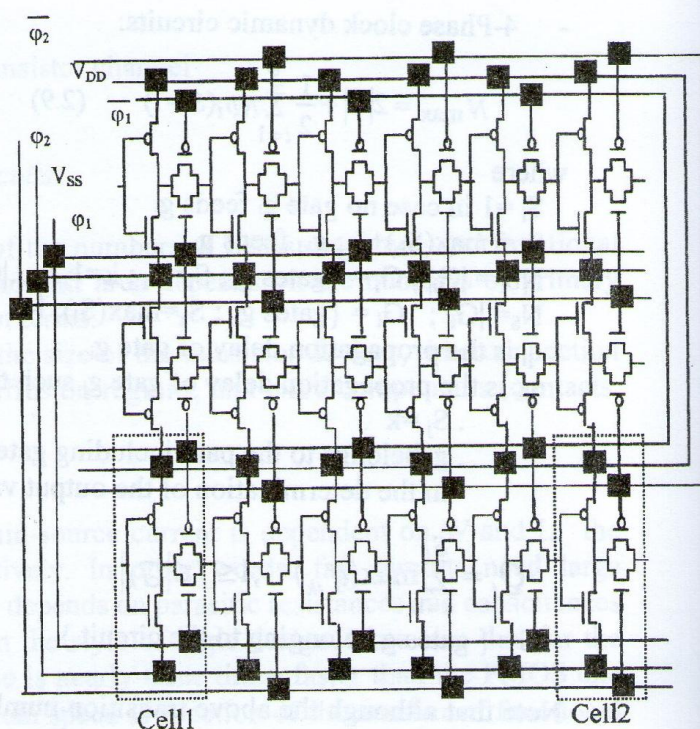


Fig.2.2. A 3*3 Shifter array

2.3.2. Memory address decoder:

The memory address decoder is also a regular structure. Its function consists to select one output line among 2^N ones for an N-input decoder, in order to select one memory cell for a read/write operation. Many styles of logic design could implement such a structure. We have studied the following:

- CMOS static gates: the drawback is that the area consumption is high. For an N-input decoder, $(2^N \cdot N + 2^N)$ transistor devices are required.
- Pseudo-NMOS: the static dissipation could be important (current flow from V_{DD} to V_{SS} in case all N-transistor devices conduct)
- CMOS dynamic gates: using a P-transistor device to load each of the 2^N load capacitances, and NOR gates to evaluate the output nodes, will discharge $2^N - 1$ load capacitances. Thus, an important switching power is dissipated for each read/write operation.

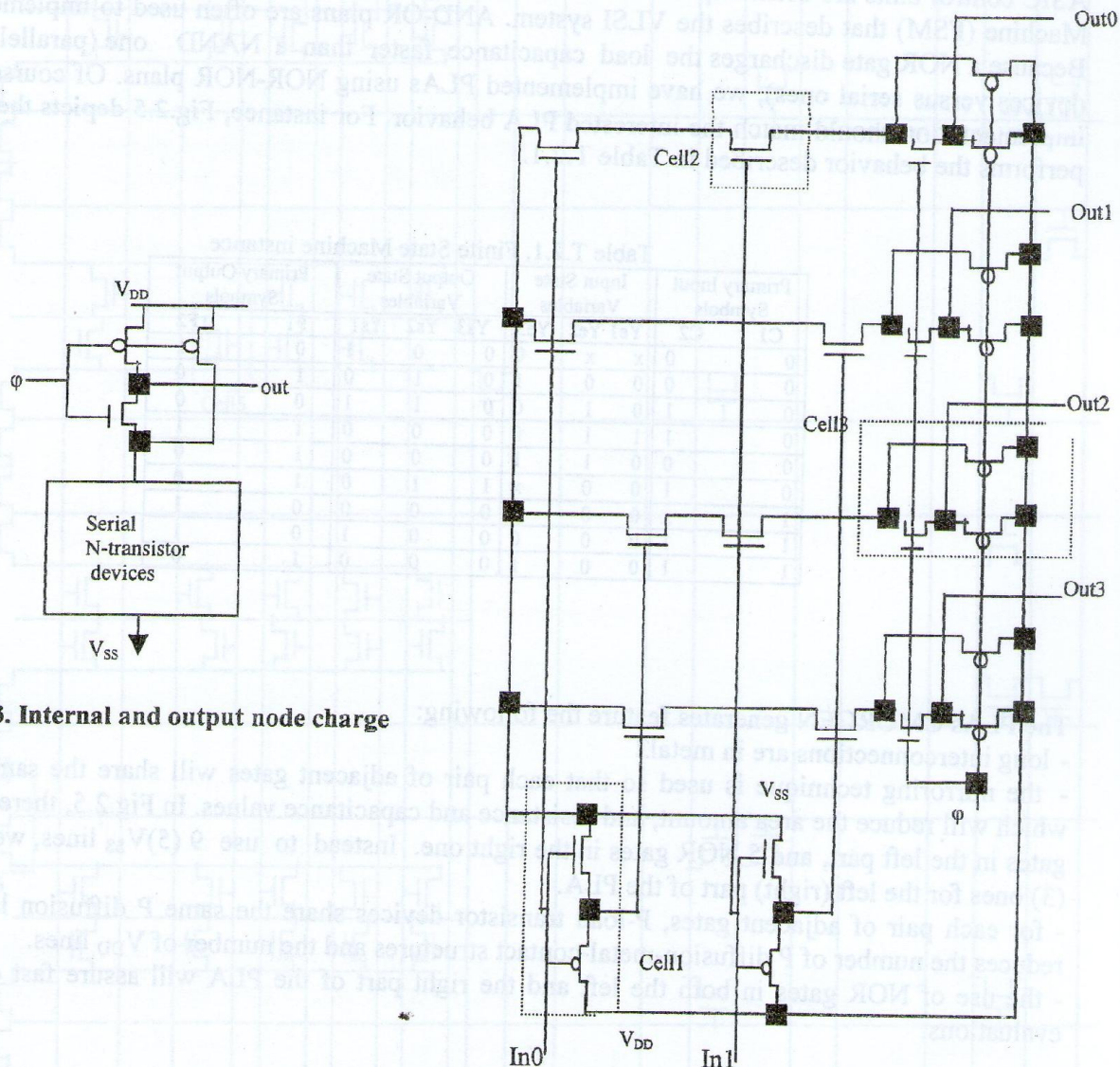


Fig.2.3. Internal and output node charge

Fig.2.4. A Memory address decoder

Because of the above drawbacks, we have implemented our decoder with CMOS dynamic NAND gates. This will assure that a single capacitance among 2^N ones will be discharged, which reduces the switching power dissipation, considerably. However, an electrical problem concerning the charge sharing holds. Assuming that the output node of any NAND gate is first precharged high, and that only the transistor device whose source V_{SS} is cut-off, the NAND gate must hold the output node high. Actually, this node is considerably discharged and the gate does not perform correctly. We have solved this problem using an extra-transistor device to precharge the source of the N-transistor device (let us call it T_{SO}) whose drain is the gate output. During the evaluation phase, T_{SO} becomes bi-directional: in case there exists at least one N-transistor device that is cut-off, the capacitance of the output node will be charged from the charge that is present in the source of T_{SO} . Otherwise, each of the N transistor devices, including T_{SO} , will assure a current flow from the gate output node to V_{SS} . In

Fig.2.3 we show how this charge sharing problem has been solved when Fig.2.4 depicts the decoder and its leaf cells. As for shifter arrays, electrical simulations are made for the decoder leaf cells at both transistor and physical level. The address decoders CMORGEN generates feature the following :

- long interconnections are in metal2,
- the switching power dissipation is considerably reduced since only a single load capacitance (the one that corresponds to the selected memory address) among 2^N ones, is discharged
- in order to reduce the contact resistance, diffusion-metal contacts are avoided when 2 adjacent transistors are too close so that these transistors share the same node (same diffusion layer). Otherwise, the diffusion-metal-contact structure is used.

2.3.3. PLAs:

ASIC control units are often implemented with PLAs. The PLA structure results from the Finite State Machine (FSM) that describes the VLSI system. AND-OR plans are often used to implement PLAs. Because a NOR gate discharges the load capacitance faster than a NAND one (parallel transistor devices versus serial ones), we have implemented PLAs using NOR-NOR plans. Of course, such an implementation should match the interested PLA behavior. For instance, Fig.2.5 depicts the PLA that performs the behavior described in Table T.2.1.

Table T.2.1. Finite State Machine instance

Primary Input Symbols		Input State Variables			Output State Variables			Primary Output Symbols	
C1	C2	Ye1	Ye2	Ye3	Ys3	Ys2	Ys1	F1	F2
0	0	x	x	0	0	0	1	0	1
0	0	0	0	1	0	1	0	1	0
0	1	0	1	0	0	1	1	0	0
0	1	1	1	0	0	0	0	1	1
0	0	0	1	1	0	0	0	1	0
0	1	0	0	x	1	1	0	1	0
1	x	0	0	0	0	0	0	0	1
1	1	0	0	0	0	0	1	0	1
1	1	0	0	1	0	0	0	1	0

The PLAs CMORGEN generates feature the following:

- long interconnections are in metal2
- the mirroring technique is used so that each pair of adjacent gates will share the same V_{SS} line, which will reduce the area amount, and resistance and capacitance values. In Fig.2.5, there are 9 NOR gates in the left part, and 5 NOR gates in the right one. Instead to use 9 (5) V_{SS} lines, we use only 5 (3) ones for the left (right) part of the PLA.
- for each pair of adjacent gates, P-load transistor devices share the same P diffusion layer, which reduces the number of P diffusion-metal-contact structures and the number of V_{DD} lines.
- the use of NOR gates in both the left and the right part of the PLA will assure fast output node evaluations.

III. PRESENTATION OF CMORGEN

3.1. Overview of CMORGEN:

CMORGEN is a modular tool so that we have no need to modify the files that contain the procedures that generate the previous regular circuits when designing a new regular structure. We have only to write the new procedure call in the main file, compile it, and then link all files. An overview of CMORGEN is depicted in Fig.3.1. MAIN is the main procedure that contains procedure calls to generate the regular structures. SHIFTER, DECODER, and PLA stand to the procedures that lay out the interested circuit.

File INCLUDE.H contains global variable declarations.

3.2. Data structure:

Our data structure aims at minimizing swapping, which yields faster processing. However, only a few data are simultaneously loaded into the main memory so that a memory allocation does not fail for large circuits. For this purpose, we use the following data structures. The former is useful to describe the layers of each leaf cell that is a tile for the interested circuit:

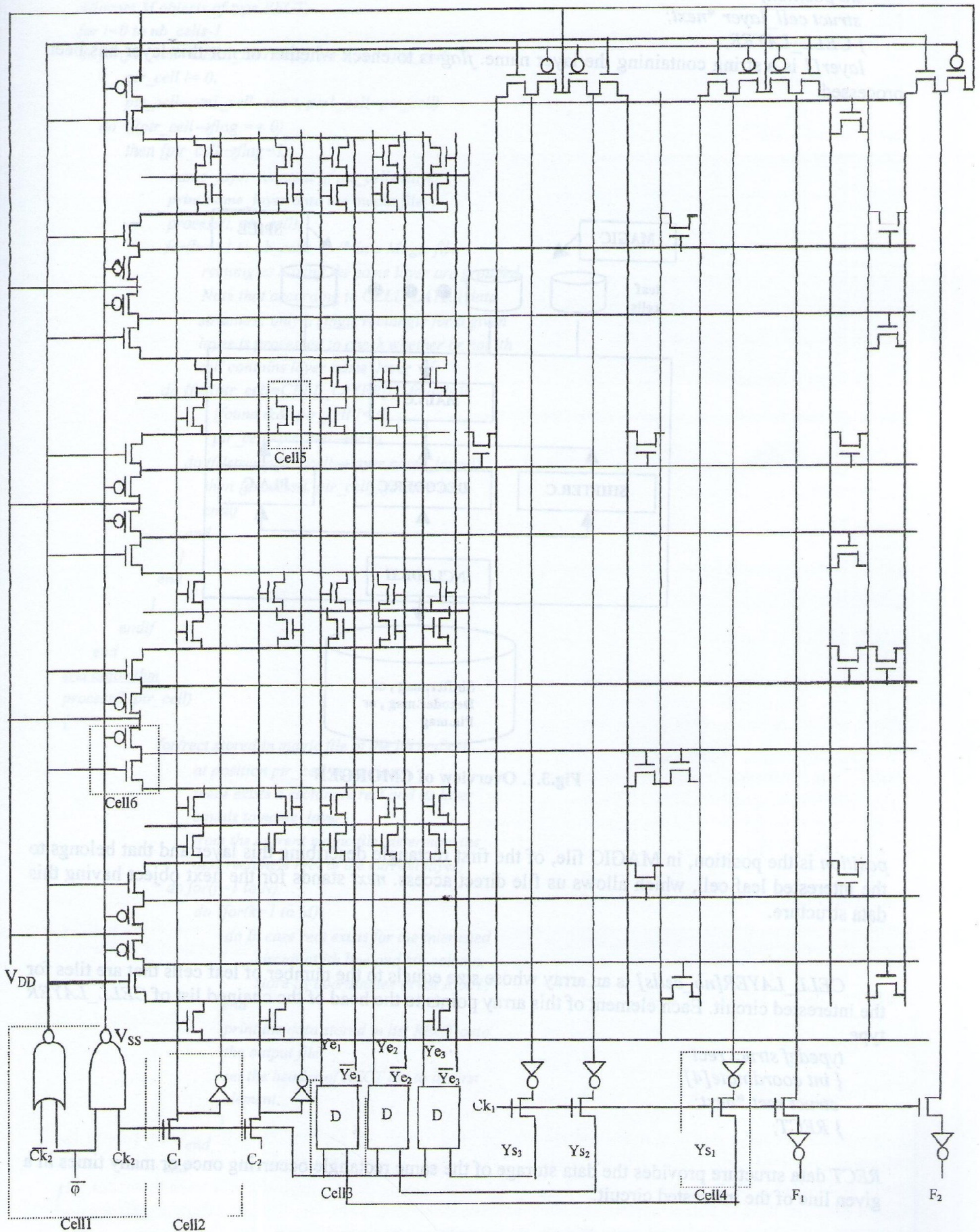


Fig.2.5. PLA implementing a control unit

```

typedef struct cell_layer
{ char layer[20];
  int flag;
  int position;
  struct cell_layer *next;
} CELL_LAYER;

```

layer[] is a string containing the layer name. *flag* is to check whether or not this layer has been processed.

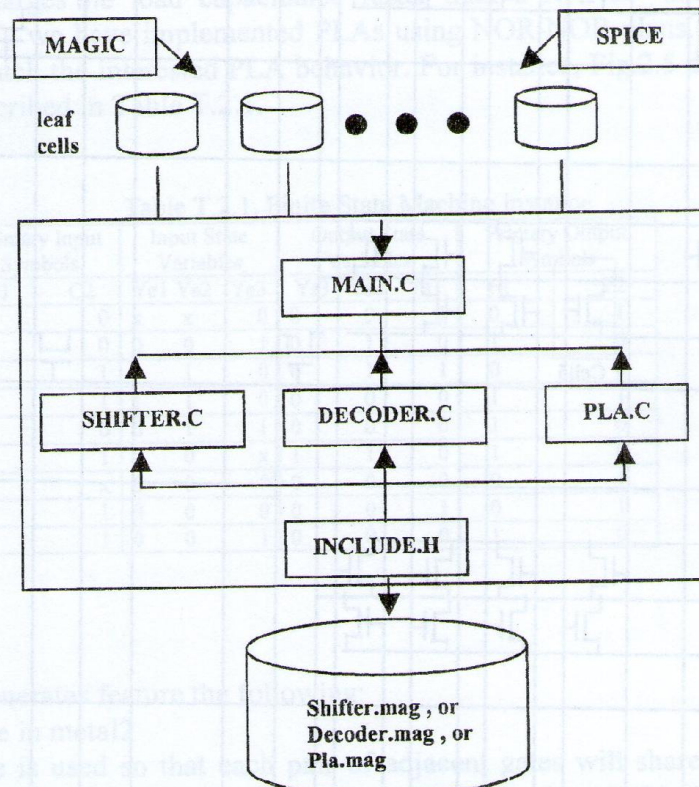


Fig.3.1. Overview of CMORGEN

position is the position, in MAGIC file, of the first rectangle describing this layer and that belongs to the interested leaf cell, which allows us file direct access. *next* stands for the next object having this data structure.

CELL_LAYER[nb_cells] is an array whose size equals to the number of leaf cells that are tiles for the interested circuit. Each element of this array points to the head of the chained list of *CELL_LAYER* type.

```

typedef struct rect
{ int coordinate[4];
  struct rect *next;
} RECT;

```

RECT data structure provides the data storage of the same rectangle occurring once or many times in a given line of the interested circuit.

3.3. Algorithm:

We now give the algorithm that lays out one of the three regular structures having N lines and M columns:

begin algorithm

Allocate M objects of type RECT;

for i=0 to nb_cells-1

do for(ptr_cell=CELL_LAYER[i], ptr1_cell=ptr_cell;

ptr_cell != 0;

ptr_cell=ptr1_cell→next, ptr1_cell=ptr_cell)

do if(ptr_cell→flag == 0)

then {ptr_cell→flag=1;

name_layer=content of ptr_cell→layer;

print name_layer onto the output file;

process(i, ptr_cell);

for(l=i+1 to nb_cells-1) /* In a Magic file,

rectangles having the same layer are grouped.

Note that according to CELL_LAYER data

structure, only a single rectangle for a given

layer is processed to check whether or not lth

cell contains layer name_layer */

do {for(ptr_cell=CELL_LAYER[l], found=0;

!found && ptr_cell != 0;

ptr_cell=ptr_cell→next)

do if(!strcmp(ptr_cell→layer, name_layer))

then {process(l, ptr_cell); found=1; }

endif

end

}

end

}

endif

end

end algorithm

process(l, ptr_cell)

{

for(rect stored in magic file of the lth leaf cell

at position ptr_cell→position;

there exists a rectangle rect and its layer

equals to name_layer;

from the current magic file determine the

next rectangle rect of the current layer)

do for(j=1 to N)

do {for(k=1 to M)

do In case rect exists for the interested

circuit at jth line and kth column,

store its coordinates in list RECT;

end

print the data stored in list RECT onto

the output file;

set the header of RECT list to its first

element;

}

end

end

}

IV.RESULTS

The following tables give the CPU time for various sizes for each of shifter array, decoder, and PLA structure.

4.1. Shifter array:

Table4.1. CPU time for various sizes of shifter array structures

Lines * Columns (# rectangles)	CPU Time (seconds)	
	SUN Station(50 MHz)	PENTIUM (100 MHz)
3 * 3 (2400)	1	0
10 * 4 (11073)	3	0
10 * 10 (35193)	8	2
20 * 10 (70373)	16	4
20 * 20 (192372)	39	11

4.2. Memory address decoder:

Table4.2. CPU time for various sizes of memory address decoder structures

# inputs (# rectangles)	CPU Time (seconds)	
	SUN Station(50 MHz)	PENTIUM (100 MHz)
4 (8409)	1	0
7 (104857)	13	3
9 (523439)	67	20
10 (1151738)	149	45

4.3. PLA:

Table4.3. CPU time for various sizes of PLA structure

PLA	#primary input symbols	#state variables	#primary output symbols	#product terms	#rect.	CPU time(s) Sun/Pent.
PLA1	2	3	2	9	65558	2 / 0
PLA2	4	16	8	40	83572	14 / 3
PLA3	4	16	8	90	176682	35 / 8

The above CPU times show that CMORGEN is able to cope with large circuits efficiently. For instance, Table4.2 shows that more than 1 Mega rectangles are processed in 45 seconds on a Pentium-based processor running at 100MHz. PLA3 which could implement an actual control unit is processed in only 8 seconds.

Finally, Fig.4.1. shows a layout for a 4*5 shifter array. A 4-input memory address decoder CMORGEN has layed out is given in Fig.4.2. The layout that corresponds to PLA3 in Table4.3 is given in Fig.4.3.

V.CONCLUSION

In this paper we have described some electrical problems and shown how they have been solved for designing our leaf cells. These cells have been designed using SPICE simulator along with tools (graphical editor, Design Rule Checker, Circuit Extractor) of MAGIC. CMORGEN uses these cells as tiles in order to lay out shifter arrays, memory address decoders, or PLAs. Efficient data structures and algorithm have been developed so that swapping operations are minimized while maintaining a free-

error main memory allocation. This feature yields CMORGEN to cope efficiently with large circuits as one could see in Tables 4.1, 4.2, and 4.3. Finally, in extent to this work, other leaf cells will be designed in order to generate other regular structures such as ROMs and RAMs.

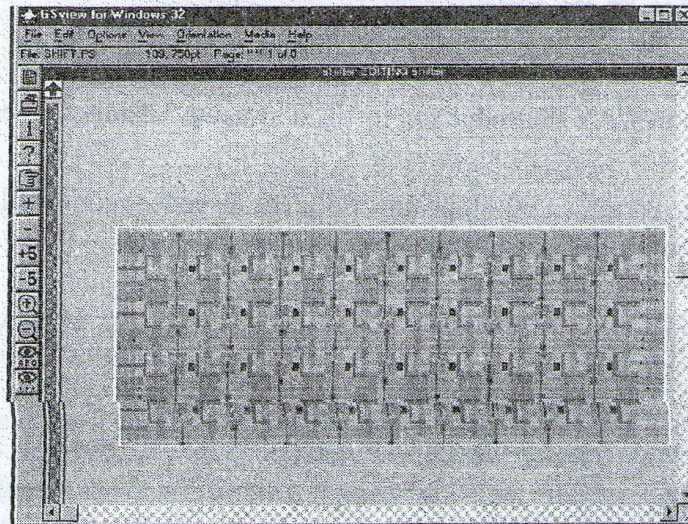


Fig.4.1. A 4*5 shifter array

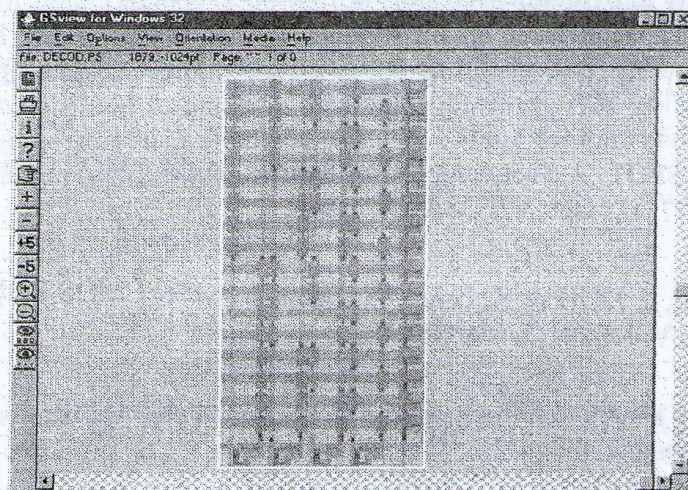


Fig.4.2. A 4-input memory address decoder.

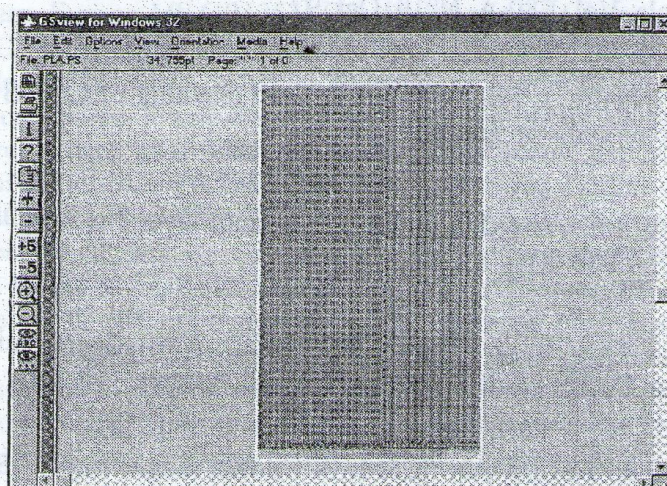


Fig.4.3. A control unit implementing PLA3.

REFERENCES

- [1] N.W.H.E. WESTE & K. ESHRAGHIAN "Principles of CMOS VLSI design: A systems perspective" Reading, MA Addison Wesley, 93.
- [2] C. MEAD & L. CONWAY "Introduction to VLSI systems " Addison Wesley
- [3] E. SICARD "Contribution au développement d'idées de conception intelligente de circuits intégrés VLSI", Thèse de Doctorat, 1987, Université de Toulouse, France.
- [4] R.N. MAYO "MONA CHIP: A System for Graphical Design of VLSI Module Generators" IEEE Int. Conf. On CAD, 1986.
- [5] R.N. MAYO & J.K. OUSTERHOUT "Pictures with parentheses: combining graphics and procedures in a VLSI layout tool" IEEE 20th Design Automation Conference, 1983.
- [6] P.W. KOLLARITSCH & N.H.E. WESTE "TOPOLOGIZER: An Expert System Translator of Transistor Connectivity to Symbolic Cell Layout", IEEE Journal of SSC, Vol.20, N°3, June 1985.
- [7] J.K. OUSTERHOUT & all "MAGIC: A Set of tools for VLSI circuit design" UC Berkeley, USA.
- [8] A. MAHDOUM "SPOT: A Switching Power Estimation for CMOS Circuits and Data Paths Tool" accepted in SASIMI'97, 1-2 Dec. 97, OSAKA, Japan.
- [9] J.C. TOU, P. GEE, J. DUH & R. Esley "A Submicrometer CMOS Embedded SRAM Compiler" IEEE Journal of SSC, Vol.27, N°3, March 92.
- [10] J.S. CARAVELLA "A Low Voltage SRAM for Embedded Applications" IEEE Journal of SSC, Vol.32, N°3, March 97.
- [11] S. SAIKA, M. FUKUI, N. SHINOMIYA, T. AKINO, S. KUNINOBU "A Two-Dimensional Transistor Placement Algorithm for Cell Synthesis and its Application to Standard Cells" IEICE Trans. On Fundamentals, Vol. E80-A, N°10, Oct. 97
- [12] S. WIMER, R. PINTER, J. FELDMAN "Optimal chaining of CMOS transistors in a functional cell" IEEE Trans. On CAD, Vol.6, sept.87
- [13] T. UEHARA & W.M. VAN CLEEMPUT "Optimal layout of CMOS functional arrays" IEEE Trans. On Computers, Vol.30, May 81.
- [14] V. CHILURURI, I. KOREN, J. BURNS "The effect of wire length minimization on yield" IEEE International Workshop Defect Fault Tolerance in VLSI systems, Oct. 94
- [15] J.L. BURNS & J.A. FELDMAN "C5M: A Control Logic Layout Synthesis System for High Performance Microprocessors" IEEE Trans. On CAD Vol.17, N°1, Jan. 98