Area Efficient Cells for LagerIV's DPP Library

Georges E. Smine and Vason P. Srini

Computer Science Division, EECS

University of California, Berkeley, CA 94720

ABSTRACT

Semi-custom design of high-performance VLSI processors has been demonstrated by the Berkeley VLSI-PLM chip using Mentor Graphics IDEA station, Cell station tools and NCR tools. To support semi-custom design using Berkeley VLSI tools such as LagerIV, we have developed a set of cells. These cells are designed with the goal of designing a high-performance VLSI Parallel Prolog Processor. They can be used in other designs such as DSP chips. Some of these cells complement those in LagerIV's DPP cell library. Others provide an area efficient replacement for the DPP cells.

1. Introduction

The LagerIV [Bro88] silicon assembly system supports semi-custom design of microprocessors using standard cells and macro cells. Semi-custom design eliminates many of the problems at the physical layout level at the expense of area and performance. Since the use of standard cells reduces the performance of processors and uses up larger die area, other alternatives are needed in order to satisfy the high-performance requirements of processors. This has motivated the development of larger macro cells. Standard macro cells allow a rapid semi-custom design, reduce chip area, and improve the speed of the processor. In designing a complex systems such as the Parallel Prolog Processor [DeS88], macro cells provide the designer with a basis for a rapid design without affecting the area and performance of the chip. In this report we describe the design of area efficient cells to design a high-performance processor such as the Prolog processor.

The Parallel Prolog Processor (PPP) [DeS88] is a high-performance processor that will support symbolic languages such as Prolog and LISP. Due to the complexity of the design of such a processor, especially when high-performance is an issue, a semi-custom VLSI design is chosen. Development is faster in a semi-custom design approach since we avoid many of the physical level design details inside the cells. Since the cells available in standard libraries did not meet our specific demands, we decided to add area efficient cells to an existing library, the LagerIV Datapath library [Bro88] (also referred to as the DPP library). The critical component in the datapath of the processor is the ALU. We selected the ALU as a basis for building a complete library of cells. Section 2 contains the first part of the design from logic gates to a layout. Section 3 describes the geometry and the rules of the layout design. A comparison of the newly created cells with their dpp equivalent is shown in section 4.

2. Description of Cells

The layout design followed the logic design of the arithmetic logic unit based on a carry bypass scheme. The design and simulation of the ALU at the gate level were done using Mentor Graphics' tools on Apollo workstations. Under best-case conditions, the ALU did a 32-bit add in 35ns; under worst-case conditions it did the addition in 50ns. These figures resulted from using NCR's 1.5 micron standard cell library. We needed a 40% improvement in speed using the proposed extented cell library.

About twelve logic gates were used in the design of the ALU. Most of these are basic gates such as 2 input AND, OR, XOR gates and 2 input multiplexer. Additional cells were needed in other parts of the datapath. In all, twenty-three cells were added to the Lager DPP library. Some of the cells are tri-stated demultiplexers with 3, 4, and 5 inputs, multiplexers with 3, 4, 5, and 6 inputs. A buffer/inverter and a D flip-flop with high fan-outs were also designed for other parts of the datapath of the PPP.

3. Geometry of Cells

Since the new cells are an extension of the LagerIV DPP cells, the DPP layout conventions and standards were followed in order to keep things compatible. First, **Magic** was the main graphics editor used for the development of the layout. Second, every cell was converted to its physical instance in the **OCT** environment because the router we were using was compatible with the **OCT** tools [Spi88]. Since Magic is a physical layout editor, one could not rely on a compactor to give the optimum area for a design. So, manual design was needed. In addition to the design rules and electrical functionality of the circuit, the height had to be approximately fifty lambda units. Each cell also has to be split into two regions: an *N-Well* region and a *P-Well* region separated by control, power, and ground lines crossing the cells vertically in *metal1*. Data lines were to cross the cells horizontally in *metal2* or *polysilicon*. The *polysilicon* lines were used for data lines only when these lines did not exceed 100 lambda units in length, due to its higher resistivity compared to *metal*. Horizontal feed lines in *metal2* were added whenever possible, allowing for bus lines to travel across the cells without complicating the routing. Figure 1 shows the general layout geometry. The approach to the layout was to draw the data buses horizontally and control lines vertically. After defining the outer geometry, transistors were layed out taking advantage of any shared contact or connection for the sole purpose of saving area. After the necessary circuitry was laid out, the feed lines were added wherever there was enough space to do so. One of the biggest challenges was the routing of some control signals and data lines inside the cells. Even though planning and systematic designs reduce the likelihood of such problems, internal routing became the main difficulty in keeping the cell's area size as it was planned. The solution to this frequent inconvenience was to use *poly*, *metal1*, and *metal2* interconnections as long as it did not violate the neighboring and underlying materials. The circuit diagram of the cells are in Figures 2-21, and the layouts are in Plots 2-21. The transistor sizes are in lambda units.

4. Comparison with DPP library

Table 1 contains the size of the every cell in terms of lambda squared. This is compared to the size of the equivalent cell built using the cells from the DPP library. The equivalent cells, however, do not represent the final area since the area used for routing interconnection was not included. Thus, the percentage decrease in area is a raw figure, and eventually the decrease would be bigger. The cells buf4 and buf5 used up more area than their DPP equivalent precisely because the area used for interconnecting the DPP cells were not considered. Routing used up a high percentage of the area due to obstacles between input and output lines.

Table 1

Comparison of new and old cells

New Cell	Area	dpp Cell	dpp Cell Area	dpp Total Area	% Decrease
2 inputs AND	1092	ANDNOR	4492	4492	75.69%
2 inputs NAND	1092	NANDNOR	3136	3136	65.17%
2 inputs OR	1092	NA	NA	NA	-
2 inputs NOR	1092	ANDNOR	4492	4492	75.69%
2 inputs XOR	2940	XORNOR	NA	NA	-
2 inputs XNOR	2698	XOR+INV	NA	NA	-
4 inputs AND	1558	3(ANDNOR)	4492	13476	88.43%
4 inputs NOR	1520	3(ANDNOR)	4492	13476	88.72%
3 inputs NAND	1184	2(NANDNOR)	3136	6272	81.12%
4 inputs NAND	1558	3(NANDNOR)	3136	9408	83.43%
5 inputs NAND	1911	4(NANDNOR)	3136	12544	84.76%
6 inputs NAND	2233	5(NANDNOR)	3136	15680	85.75%
3 inputs Buffer	5311	3(1 input Buffer)	1900	5700	6.82%
4 inputs Buffer	8208	4(1 input Buffer)	1900	7600	-8.00%
5 inputs Buffer	9792	5(1 input Buffer)	1900	9500	-3.07%
3 to 1 MUX	5082	2(2 to 1 MUX)	3450	6900	26.34%
4 to 1 MUX	6930	3(2 to 1 MUX)	3450	10350	33.04%
5 to 1 MUX	10626	4(2 to 1 MUX)	3450	13800	23.00%
6 to 1 MUX	13622	5(2 to 1 MUX)	3450	17250	21.03%
AB + CD	2747	3(NANDNOR)	3136	9408	70.80%
SUM CELL	5967	NA	NA	NA	-
32 Lines Driver	6380	NA	NA	NA	-
DFF	6897	NA	NA	NA	-

NA: Not Available.

Areas in lambda squared.

5. Conclusion

Since fully-custom design is time consuming and hard to debug, semi-custom design has emerged as an alternative for fast layout generation and simulation. However, the variety of cells that exists in certain libraries may not be enough for some designs. Thus, the need to develop an extended library is justified. We have developed twenty-three cells that follow the LagerIV DPP library guidelines and meet our needs for the layout generation of the arithmetic unit of the Parallel Prolog Processor. These new cells will allow us to rapidly generate a layout for the PPP while giving us a much better area than the layout using the regular DPP cells. In addition, the performance of the processor will be enhanced because of the use of these cells.

Acknowledgements

We would like to thank Samuel Sheng and Professor Robert Brodersen for the use of the LagerIV in our research. Our special thanks to Samuel for providing us his programs for a fast SPICE simulation, and Linda Bushnell for proof-reading this report. Finally, we thank all members of the Aquarius project for their help, comments, and suggestions.

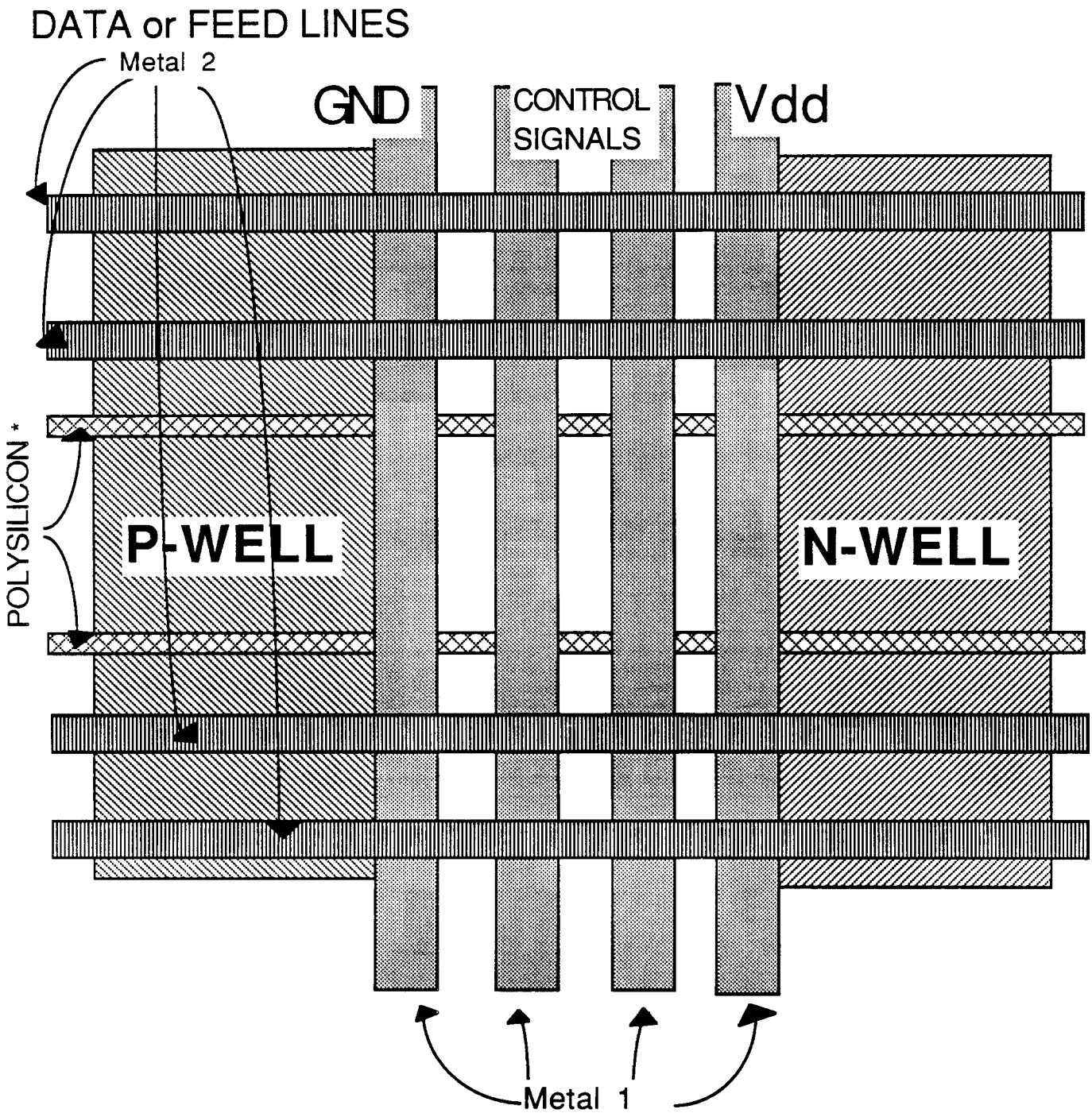
This research was partially funded by the Defense Advanced Research Projects Agency (DoD) and monitored by Office of Naval Research under Contract No. N00014-88-K-0579.

References

[Bro88] R. W. Brodersen, *LagerIV Distribution 1.0 Silicon Assembly System Manual*, Electronics Research Laboratory, University of California, Berkeley, CA, June 1988.

[DeS88] A. M. Despain and V. P. Srini, Multiprocessor Architecture Research for Prolog, *Proceedings of the State of California MICRO-1986 Report*, March 1988.

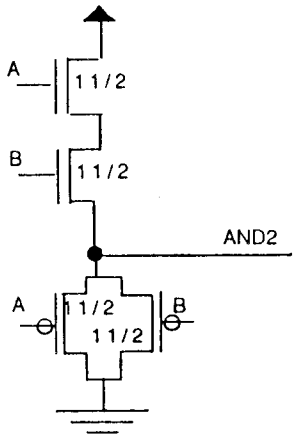
[Spi88] R. Spickelmier, *Oct Tools Distribution 2.1*, Electronics Research Laboratory, University of California, Berkeley, CA, March 1988.



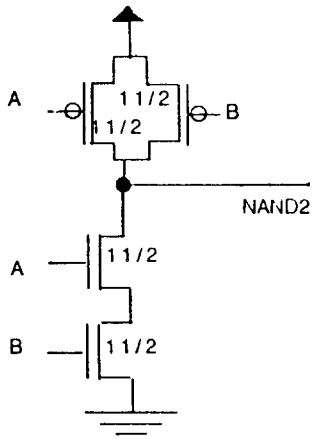
* Polysilicon lines are for data lines only when not exceeding 100 lambdas in length. In the latter case Metal2 is used for data lines.

GEOMETRY OF THE dpp CELLS

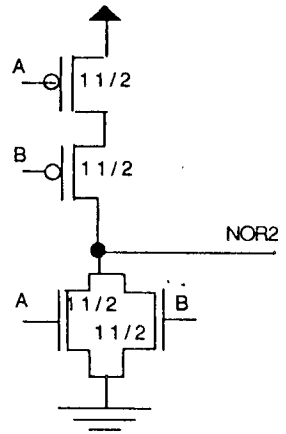
Figure 1



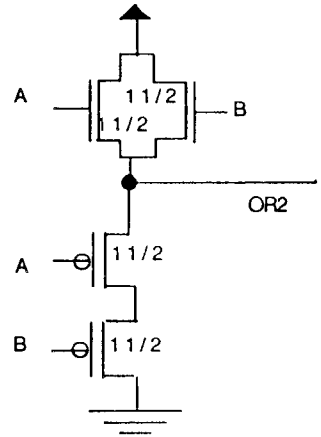
AND 2



NAND2

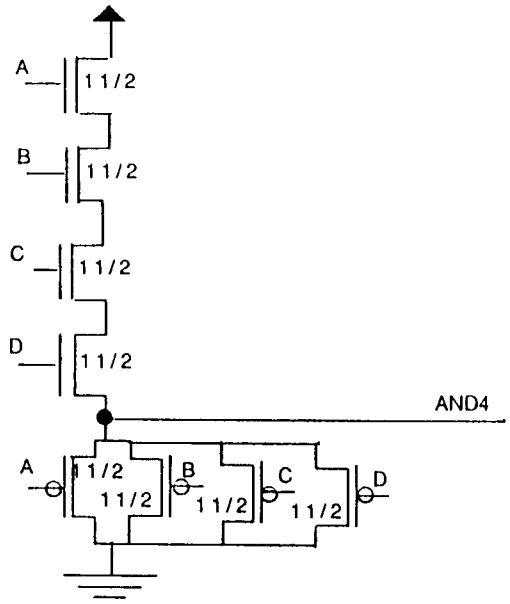


NOR2



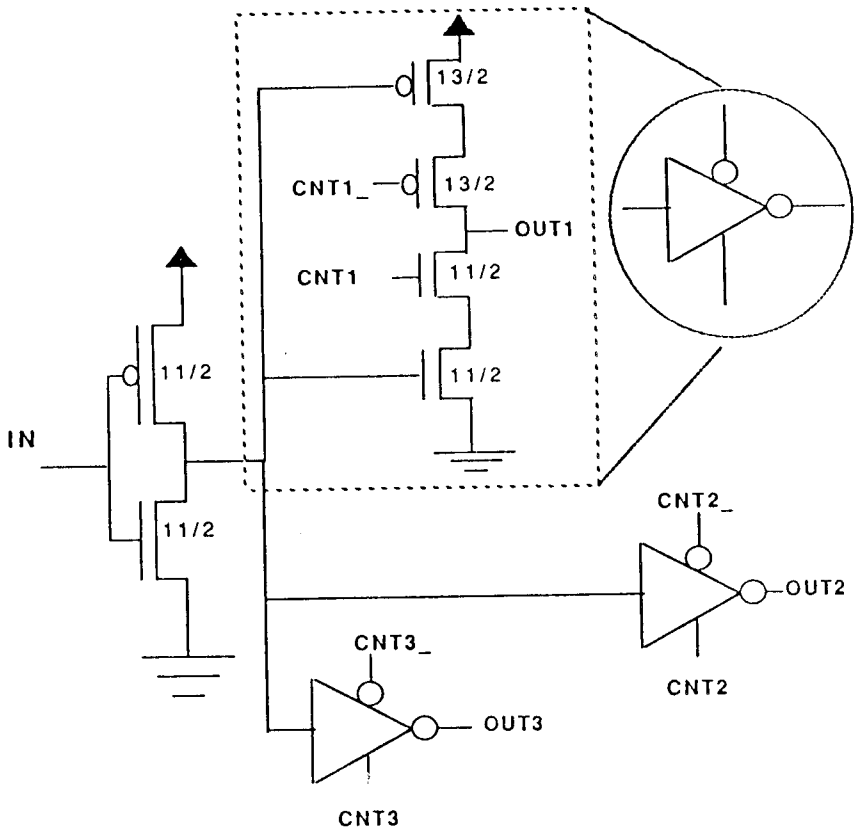
OR2

Figure 2



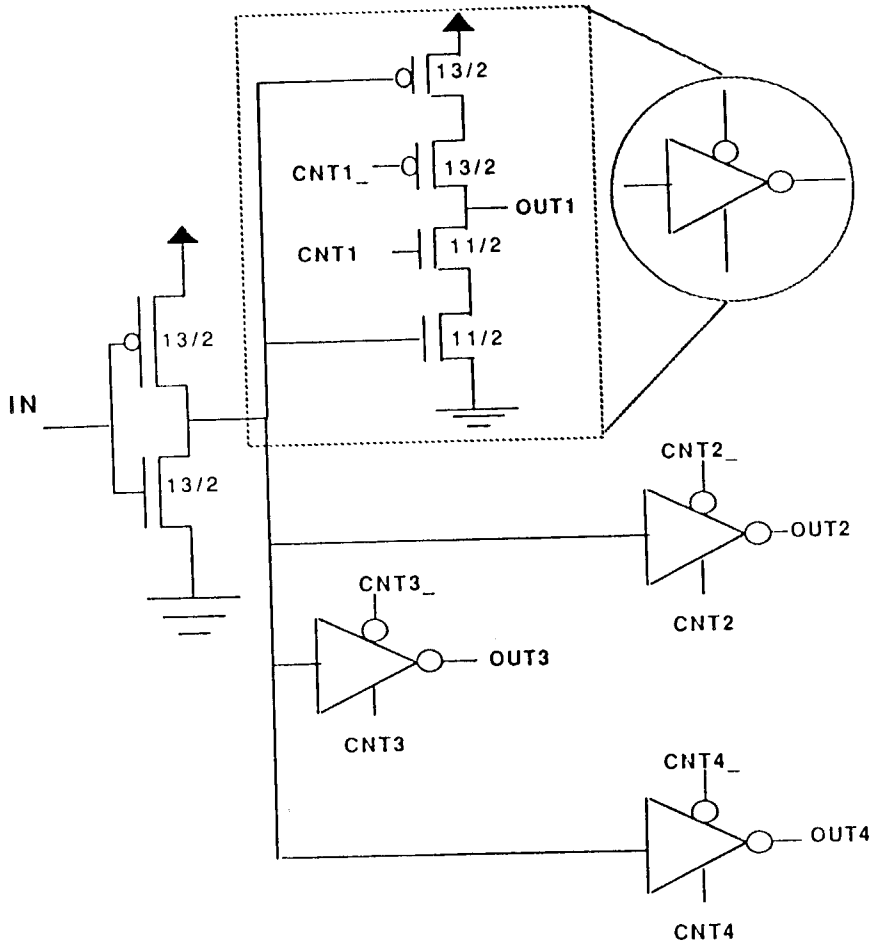
AND4

Figure 3



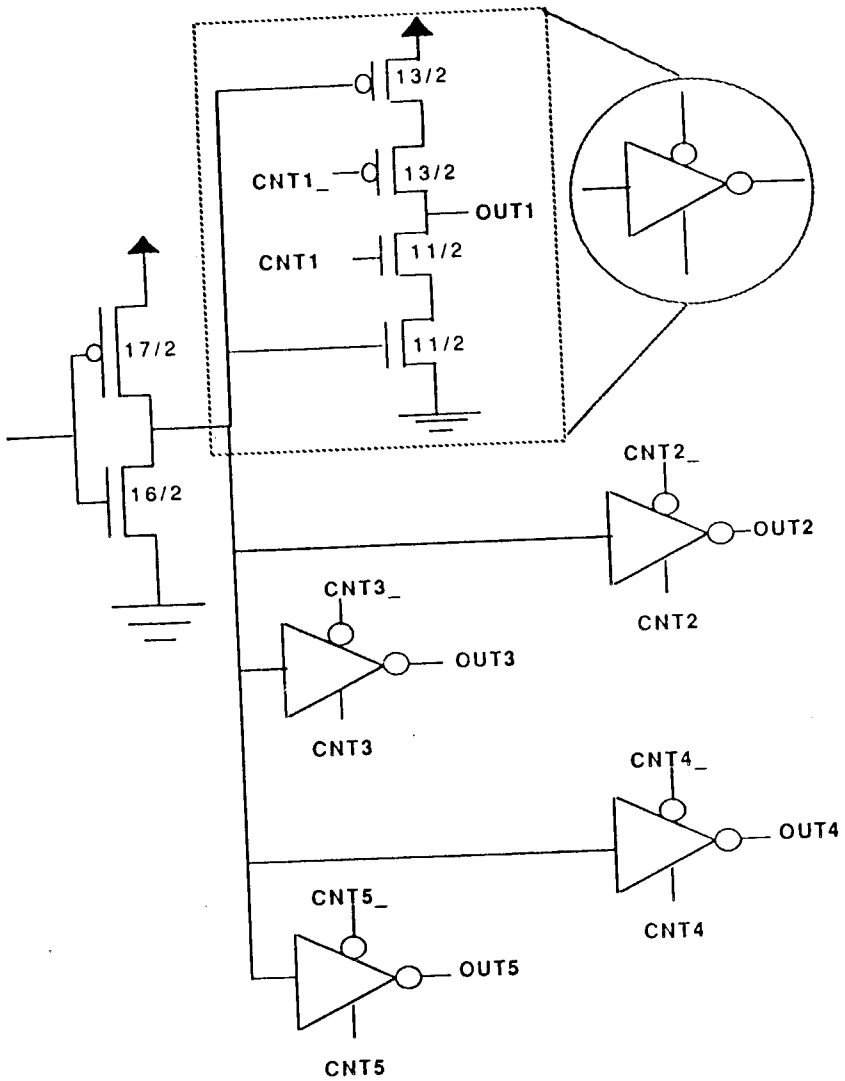
3 OUTPUTS TRI-STATE BUFFER

Figure 4



4 OUTPUTS TRI-STATE BUFFER

Figure 5



5 OUTPUTS TRI-STATE BUFFER

Figure 6

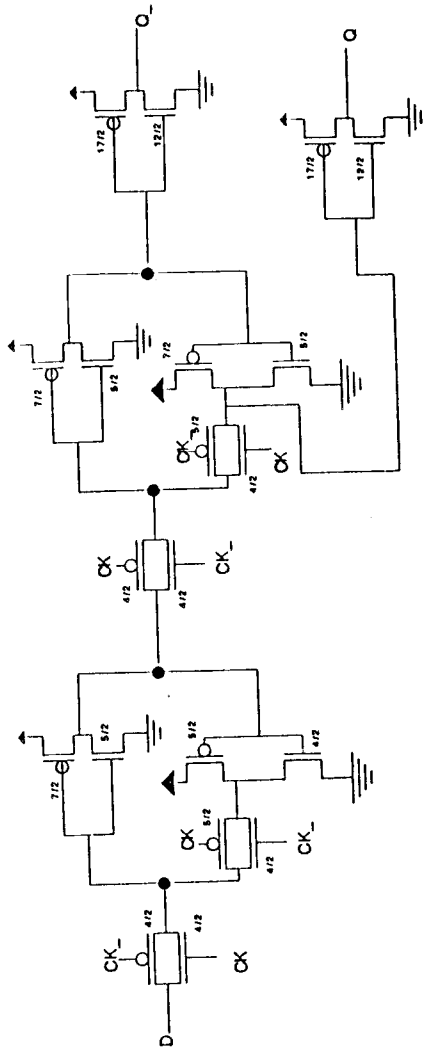
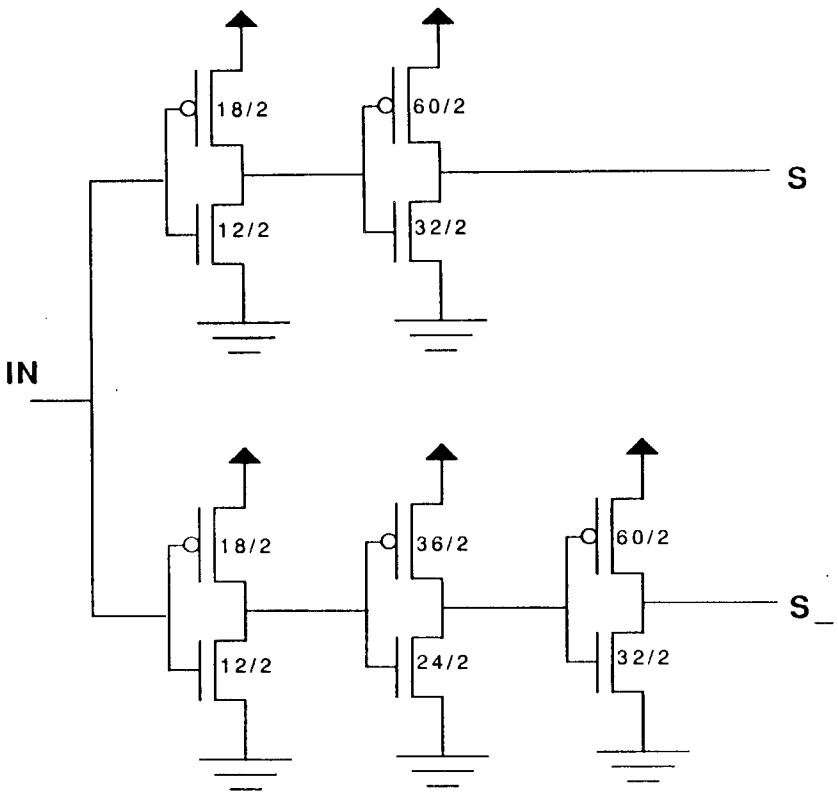


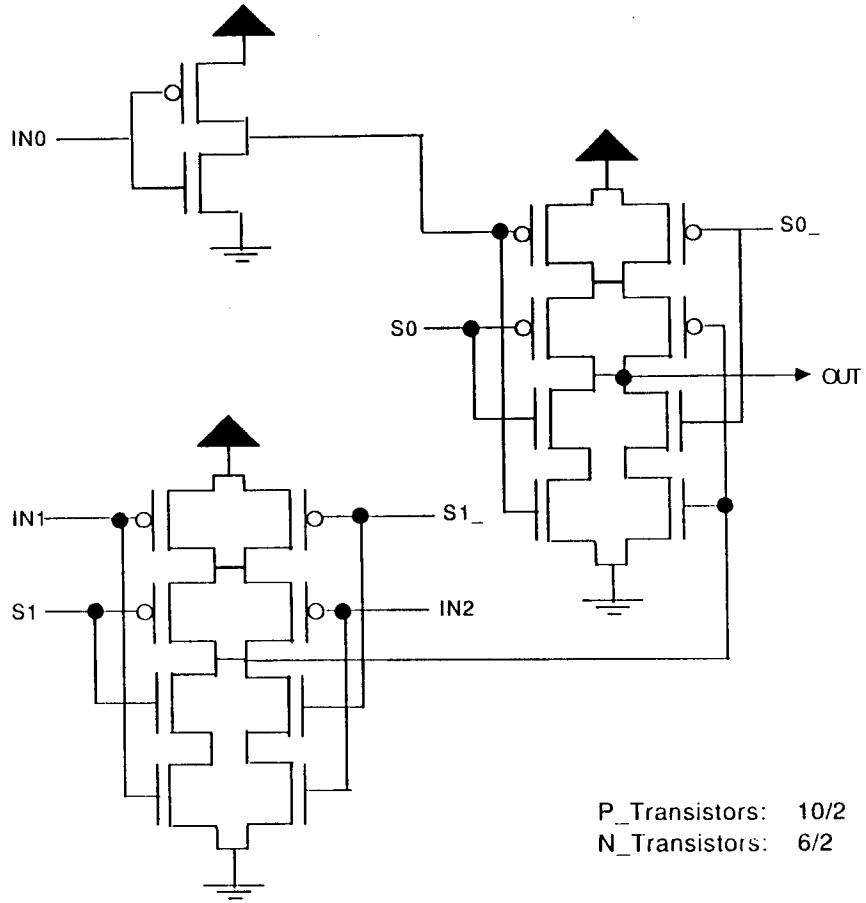
Figure 7

DFF



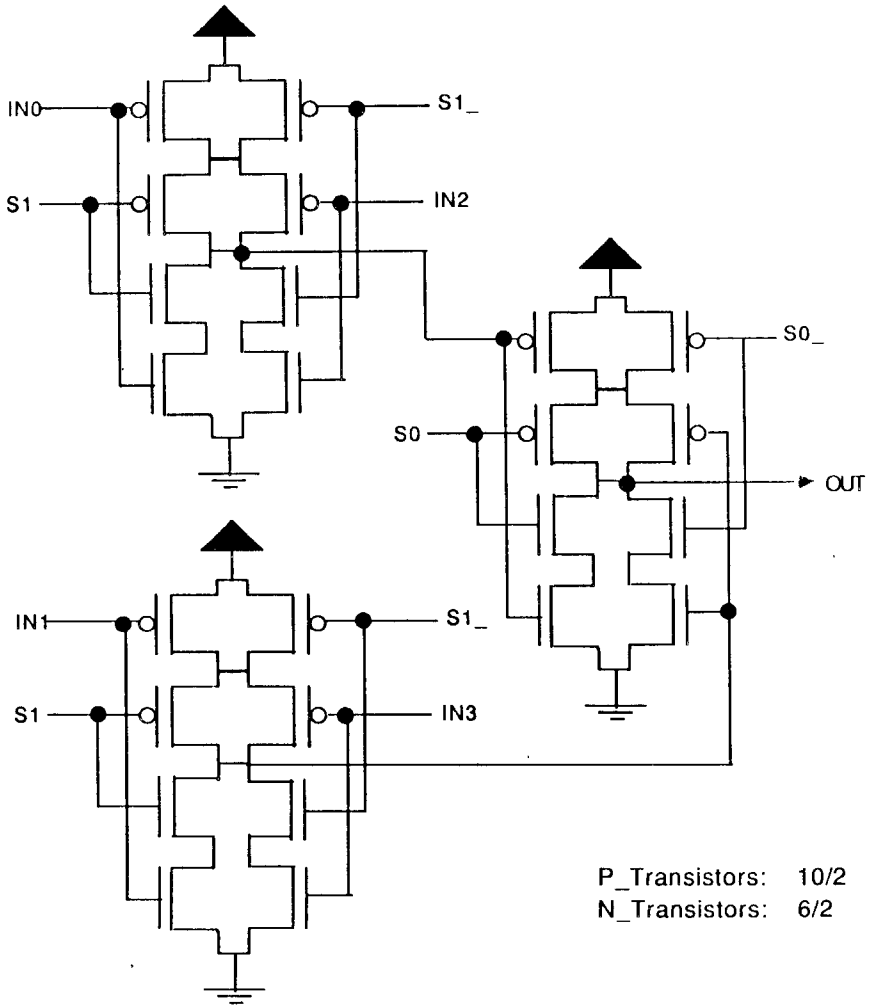
32 Lines Driver

Figure 8



3 to 1 MUX

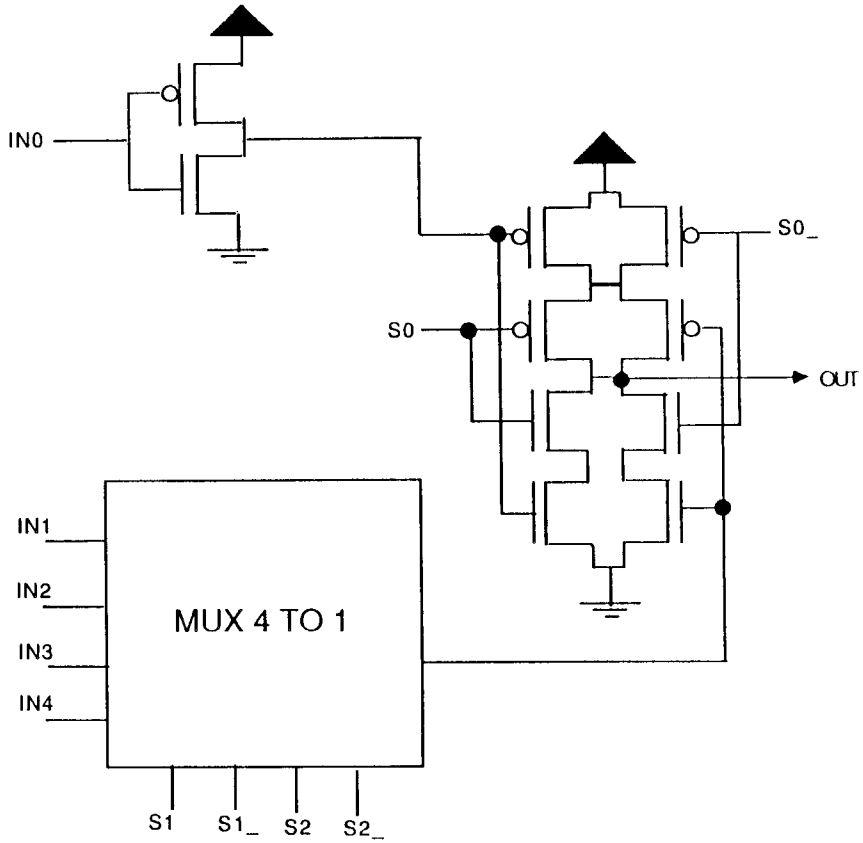
Figure 9



P_Transistors: 10/2
N_Transistors: 6/2

4 to 1 MUX

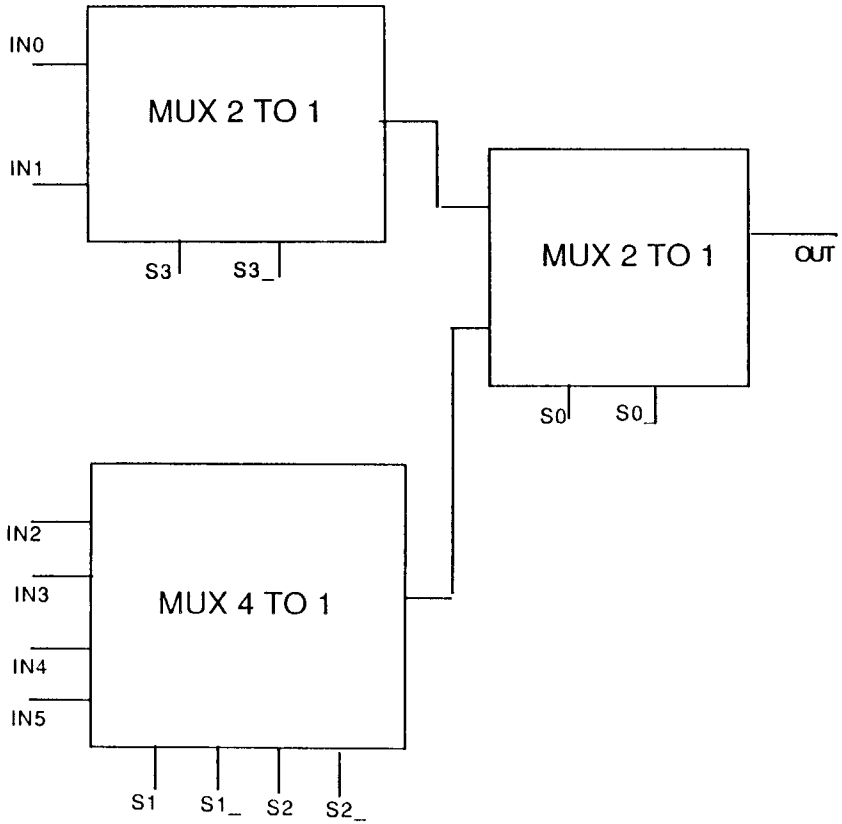
Figure 10



P_Transistors: 10/2
N_Transistors: 6/2

5 to 1 MUX

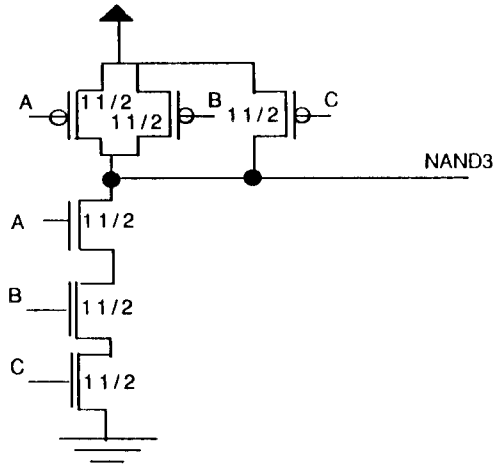
Figure 11



P_Transistors: 10/2
N_Transistors: 6/2

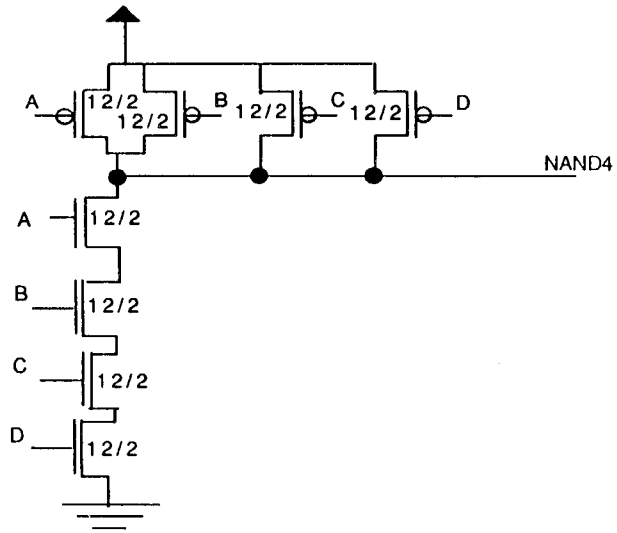
6 to 1 MUX

Figure 12



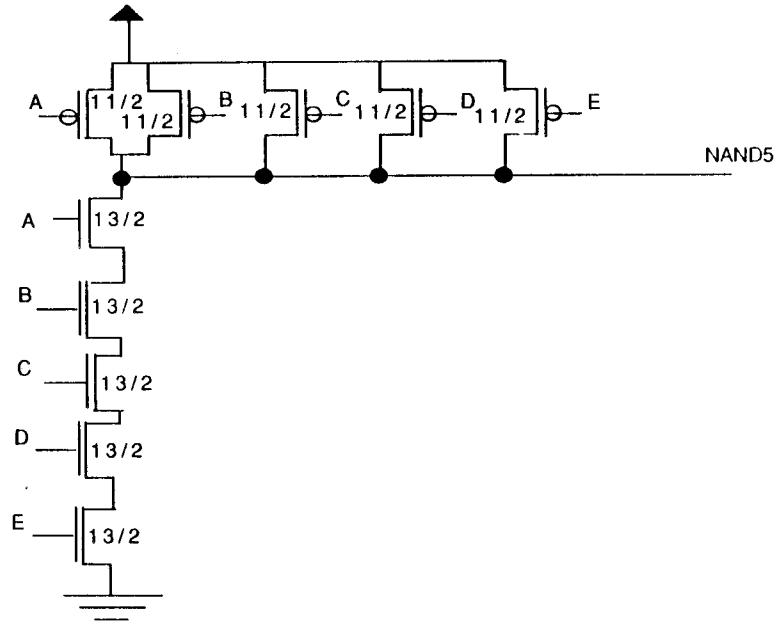
NAND3

Figure 13



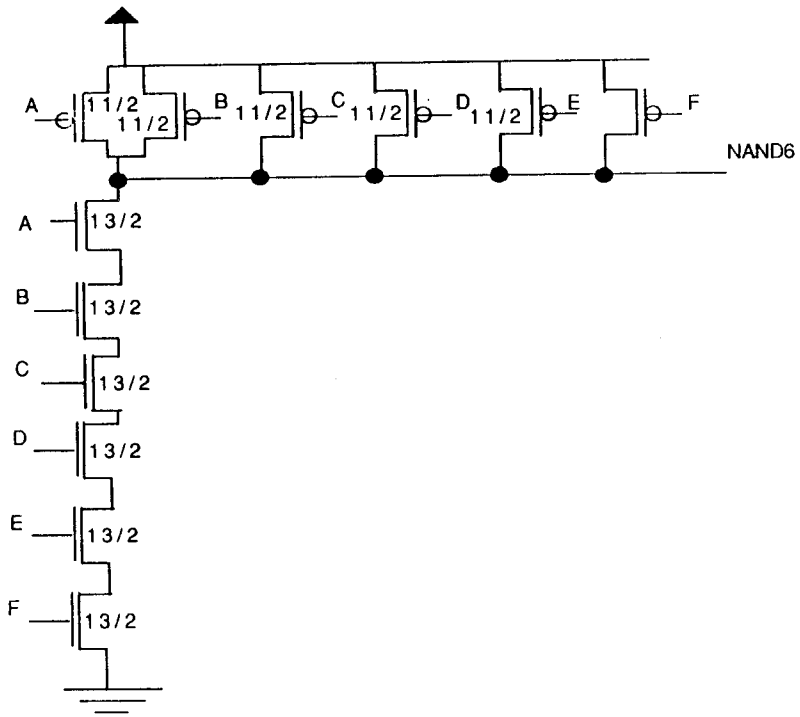
NAND4

Figure 14



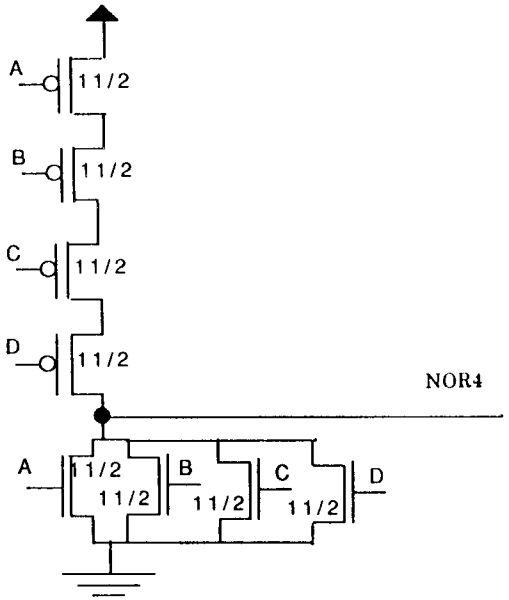
NAND5

Figure 15



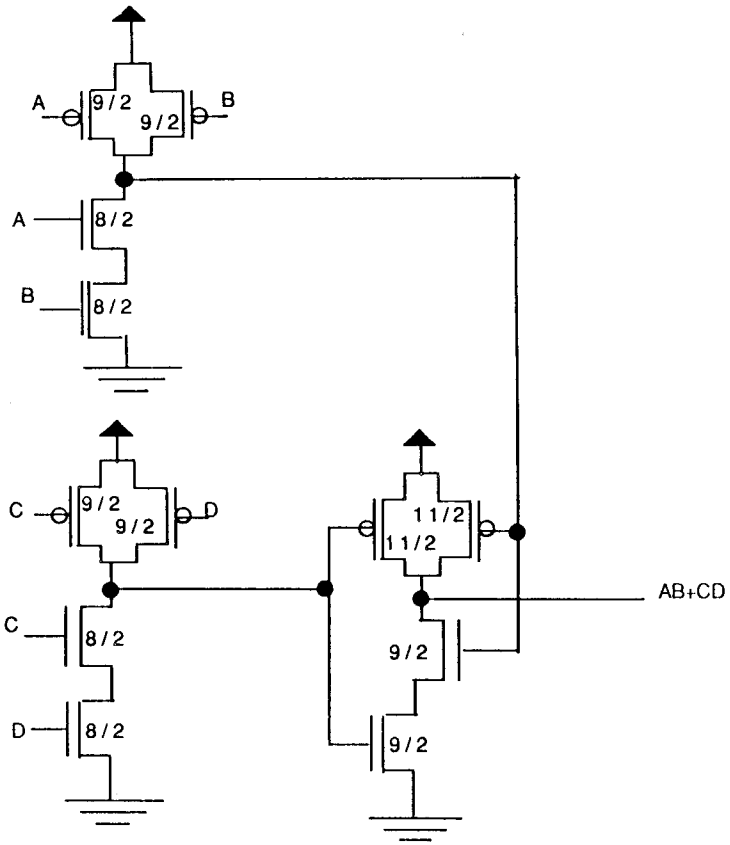
NAND6

Figure 16



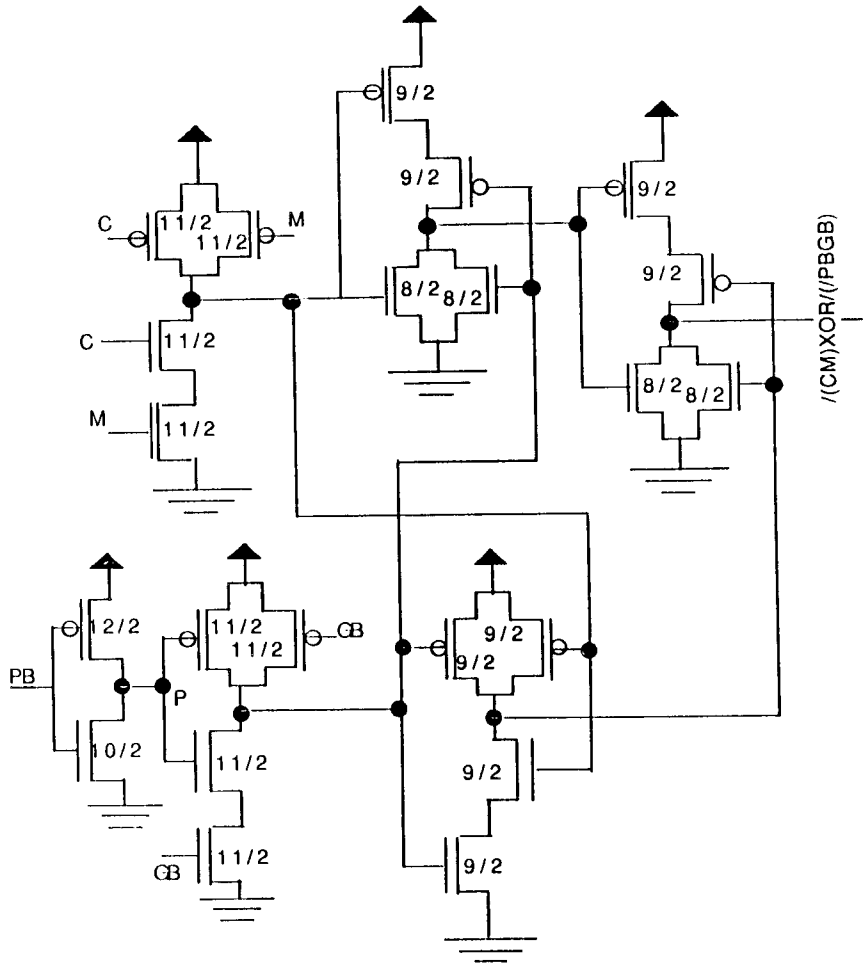
NOR4

Figure 17



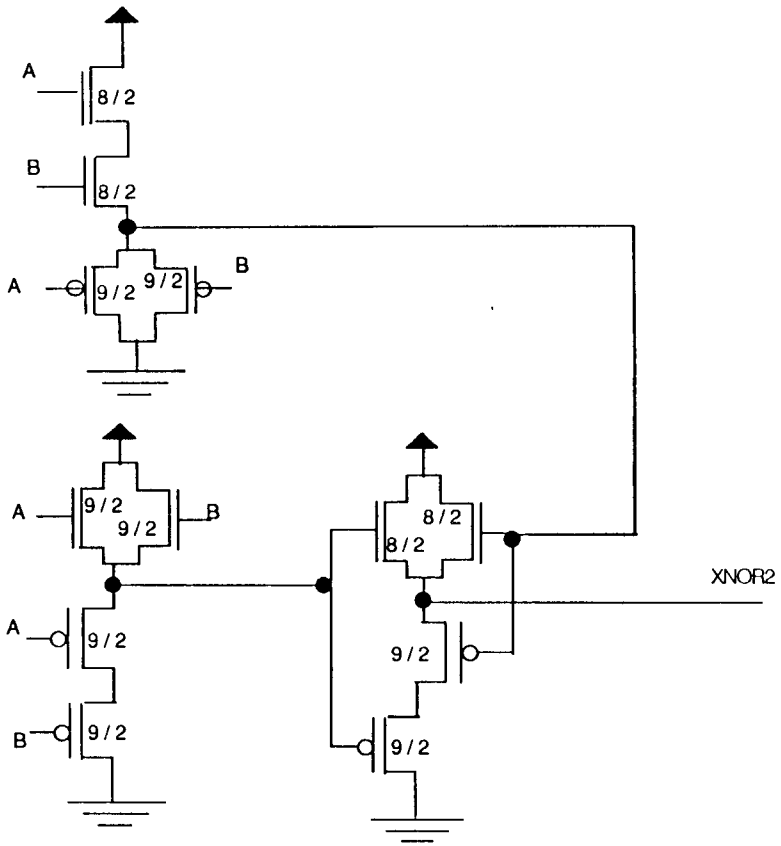
AB + CD

Figure 18



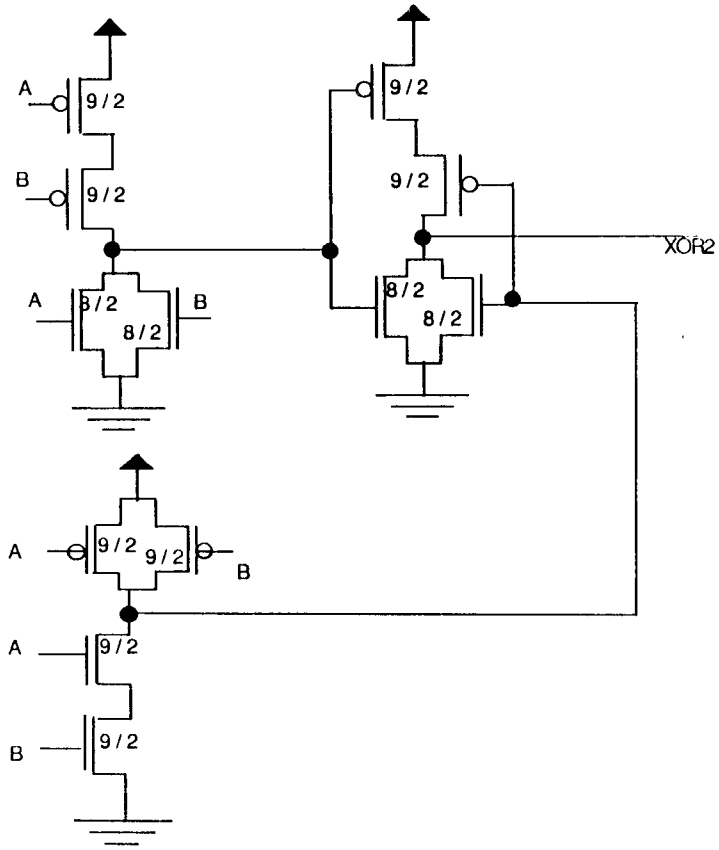
SUM CELL /(CM) XOR /(/PBGB)

Figure 19



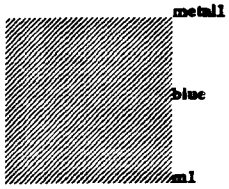
XNOR2

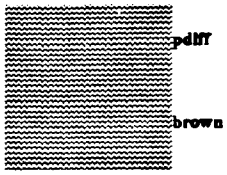
Figure 20

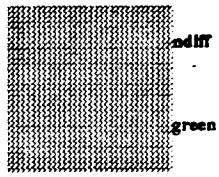


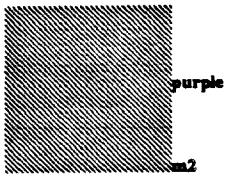
XOR 2

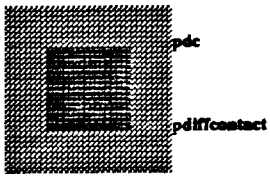
Figure 21

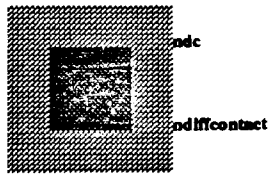


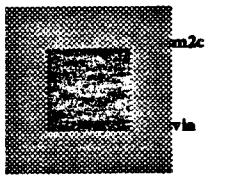


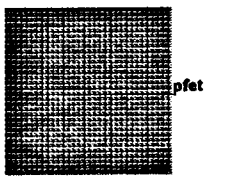


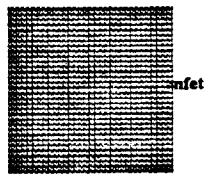


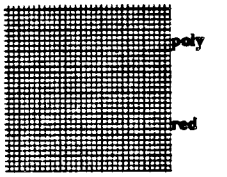


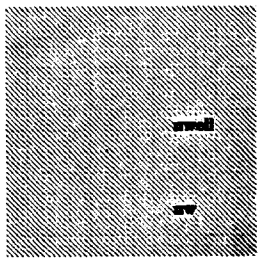


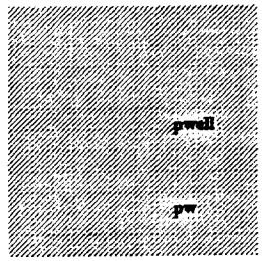


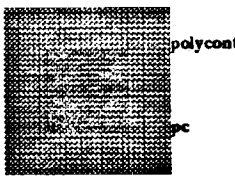


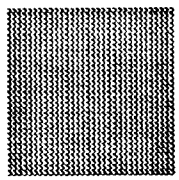


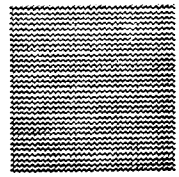




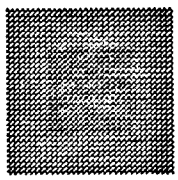


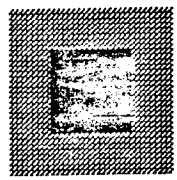


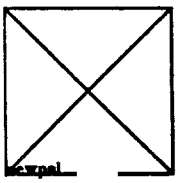




pad

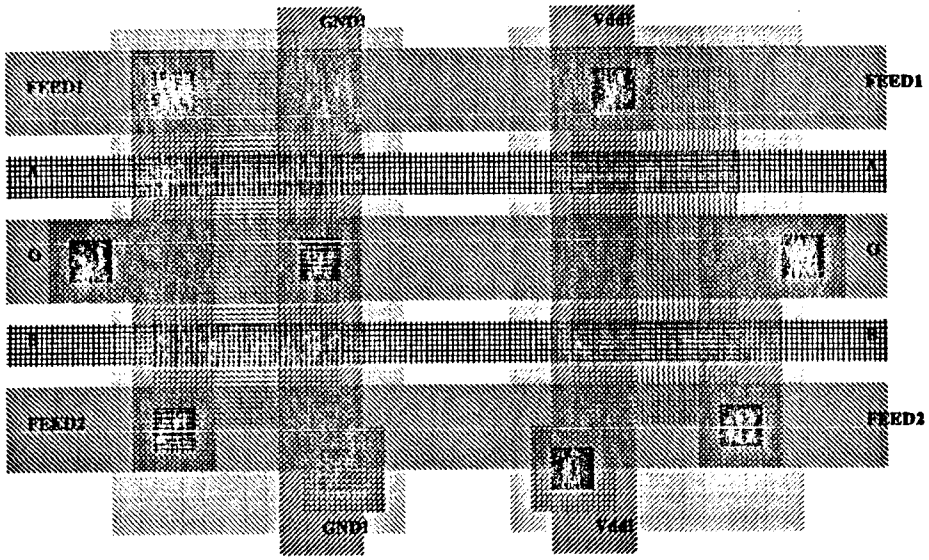




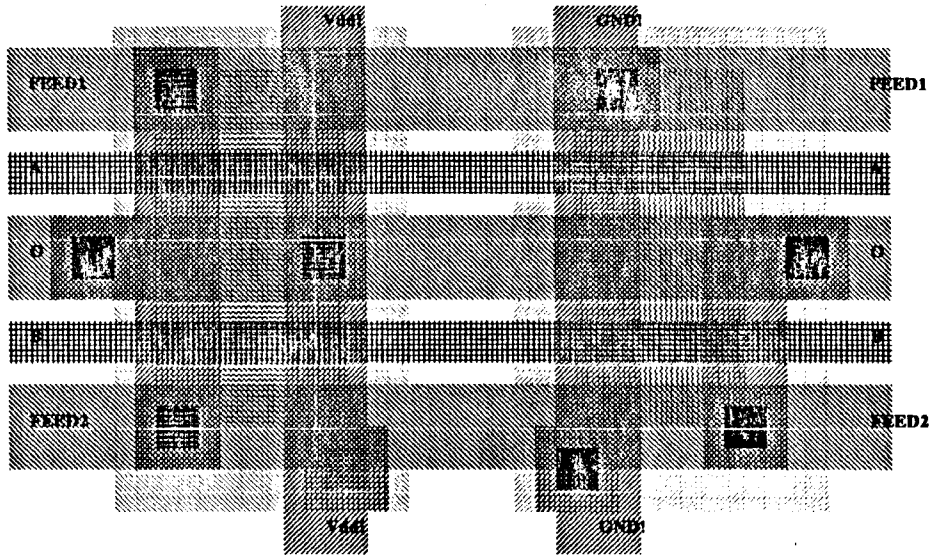


Palette

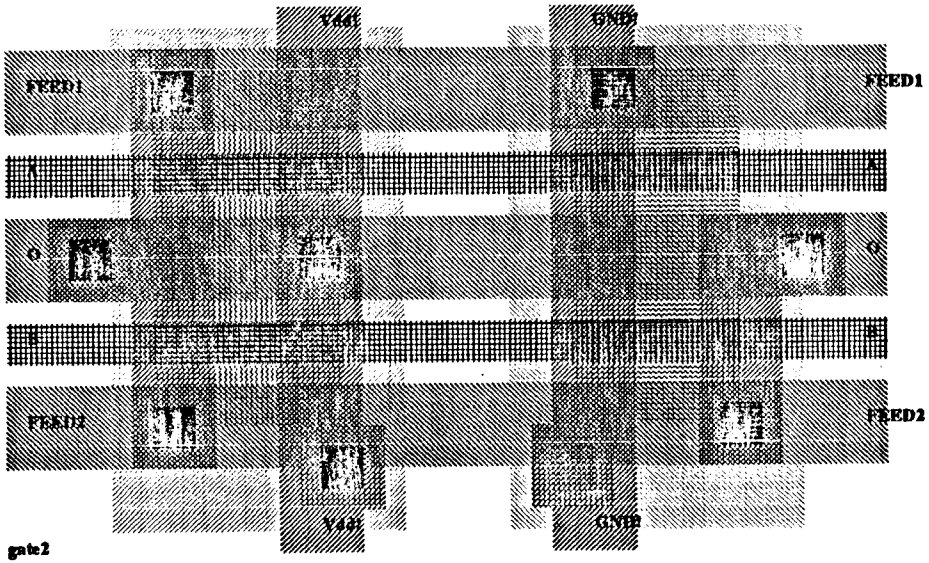
Plot 1



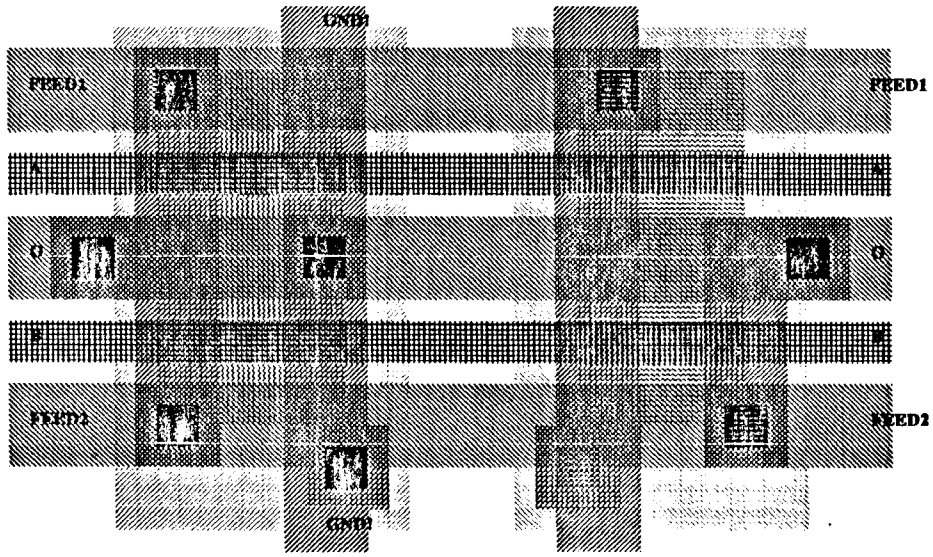
AND2



NAND2

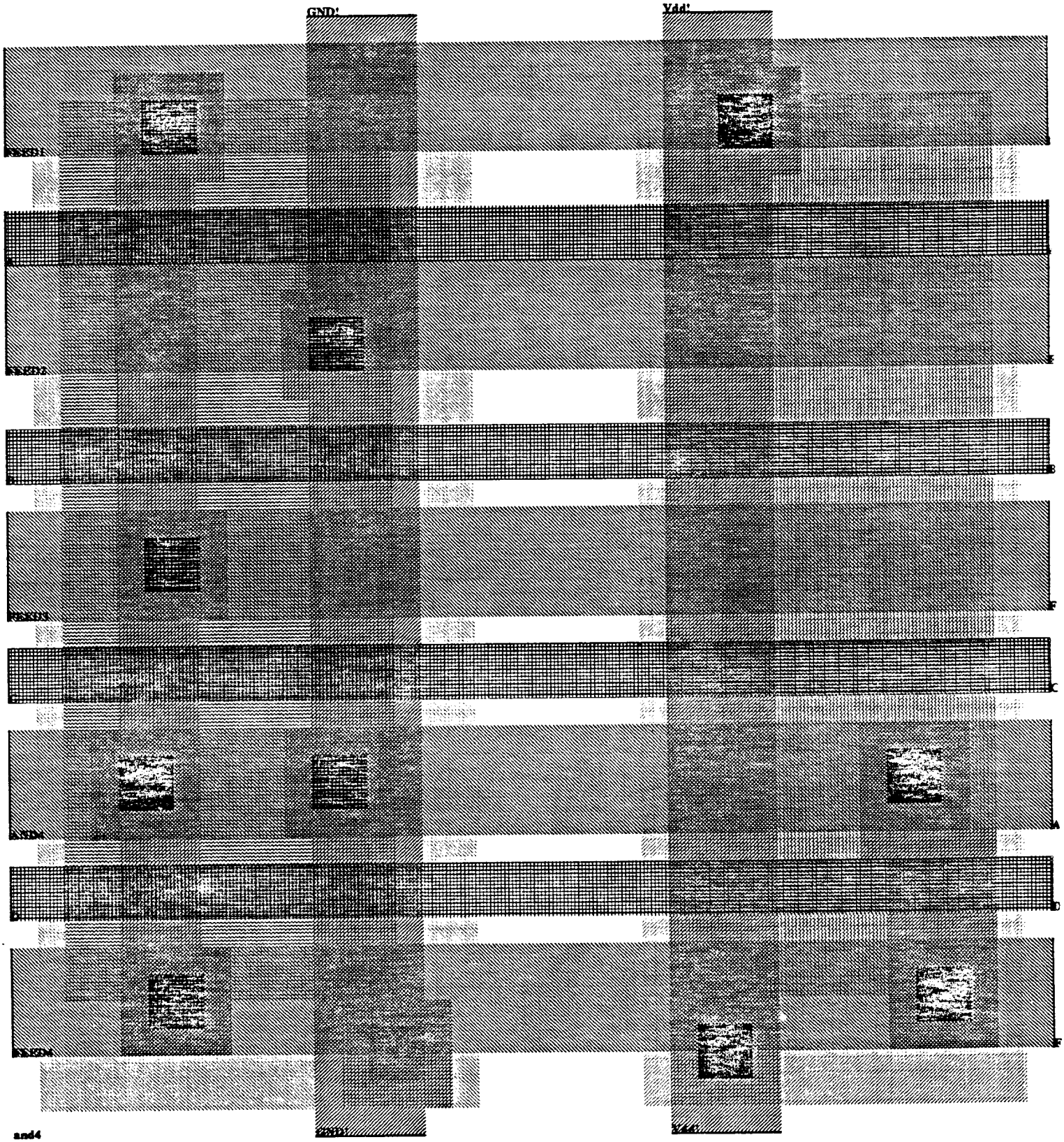


OR2



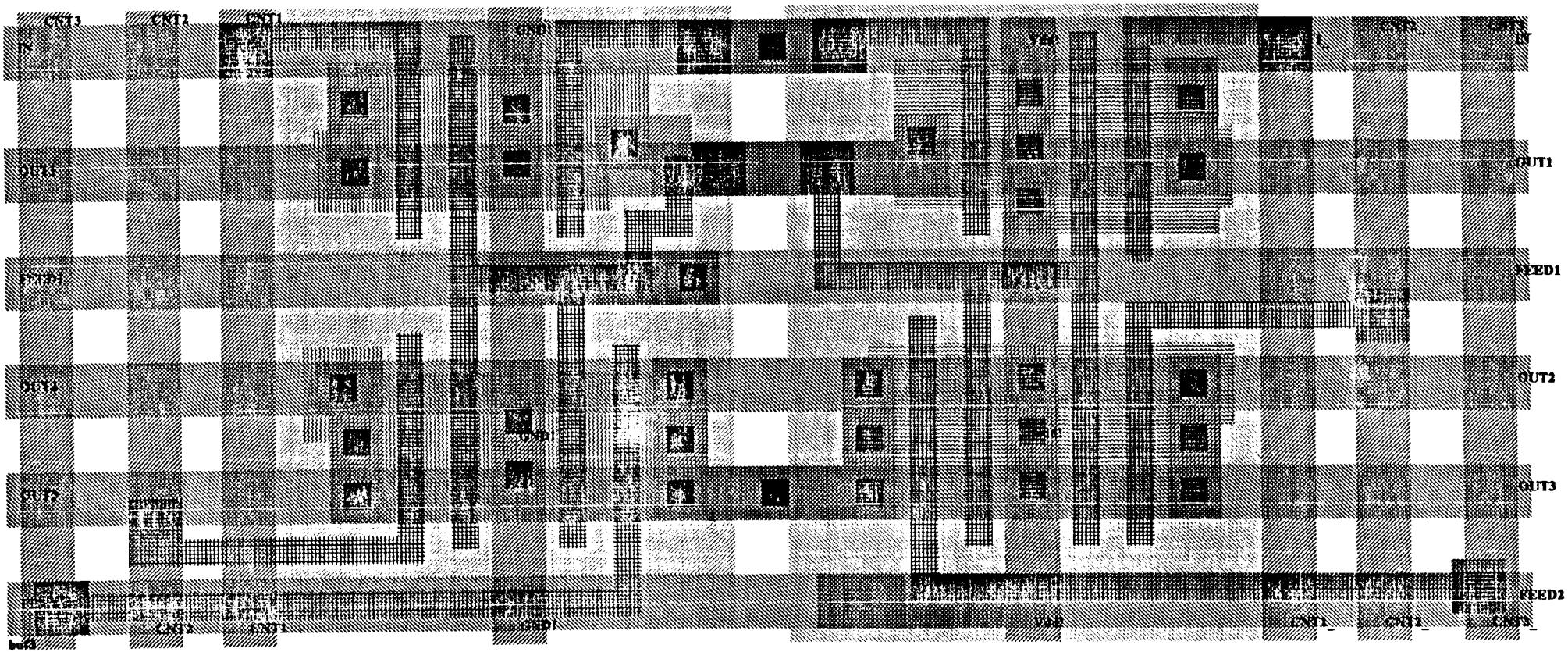
NOR2

Plot 2



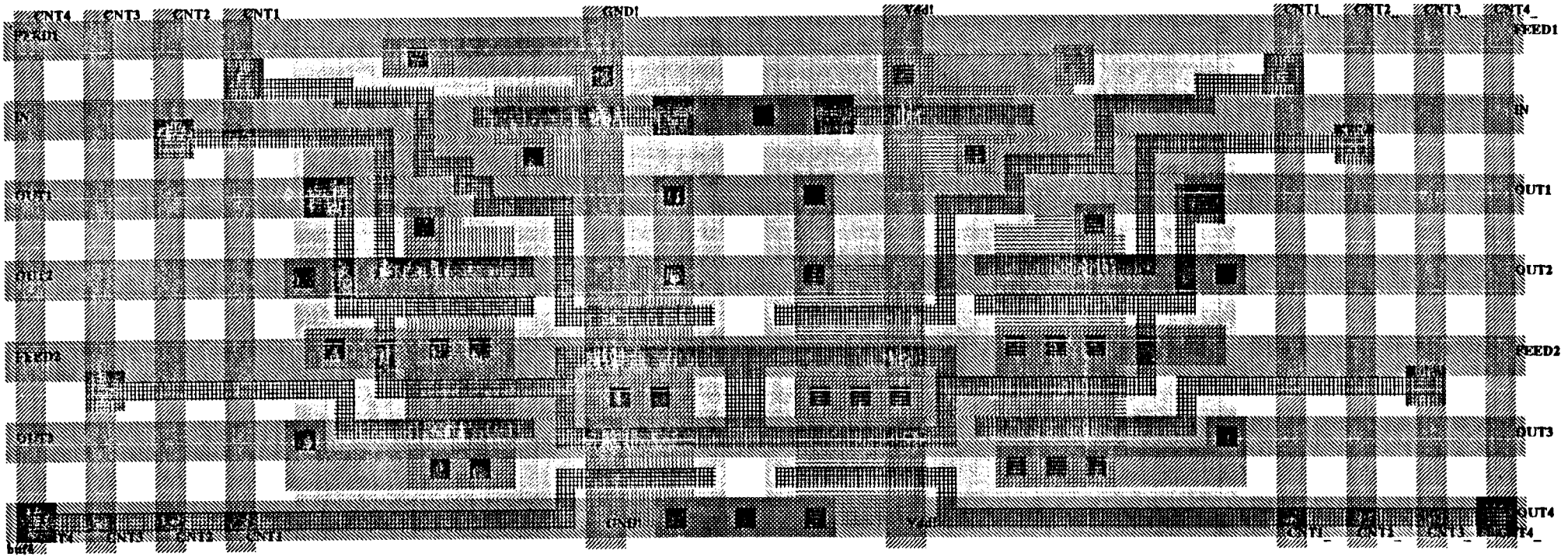
AND4

Plot 3



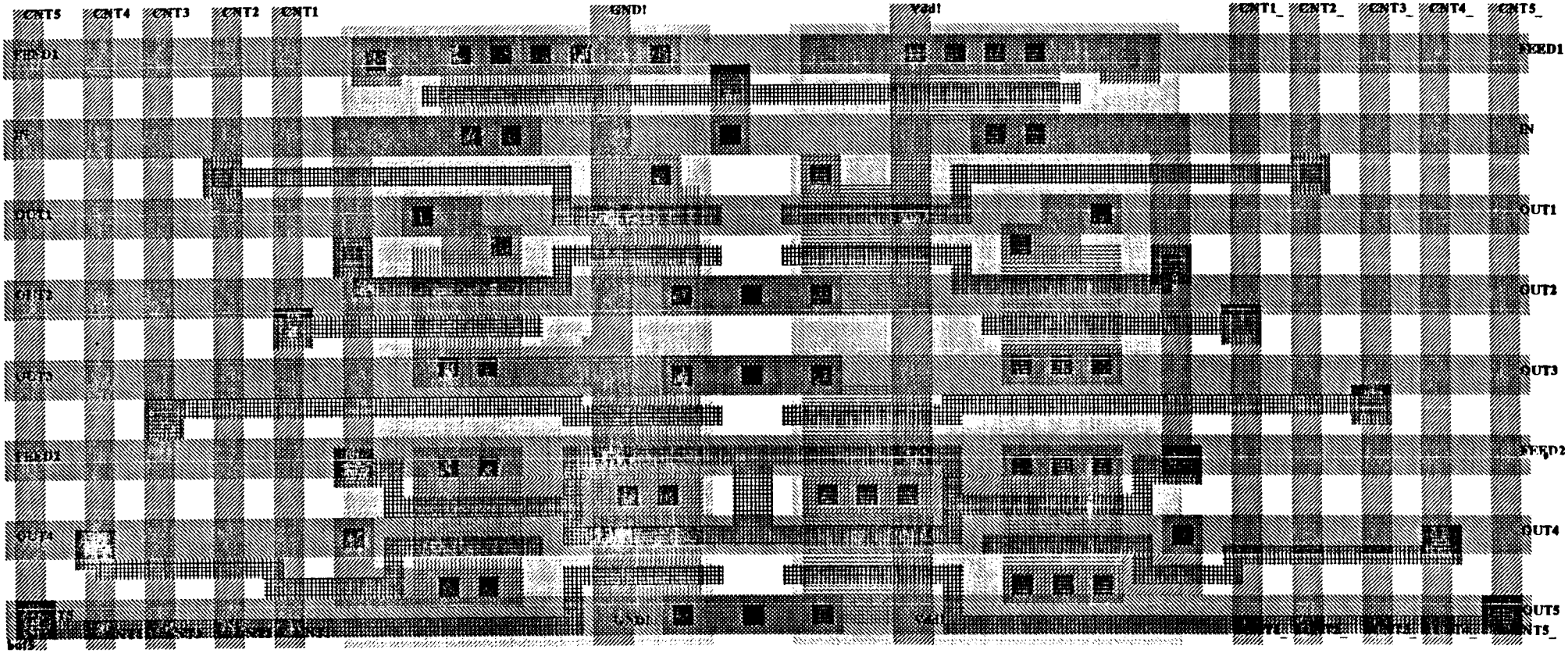
3 Inputs buffer

Plot 4



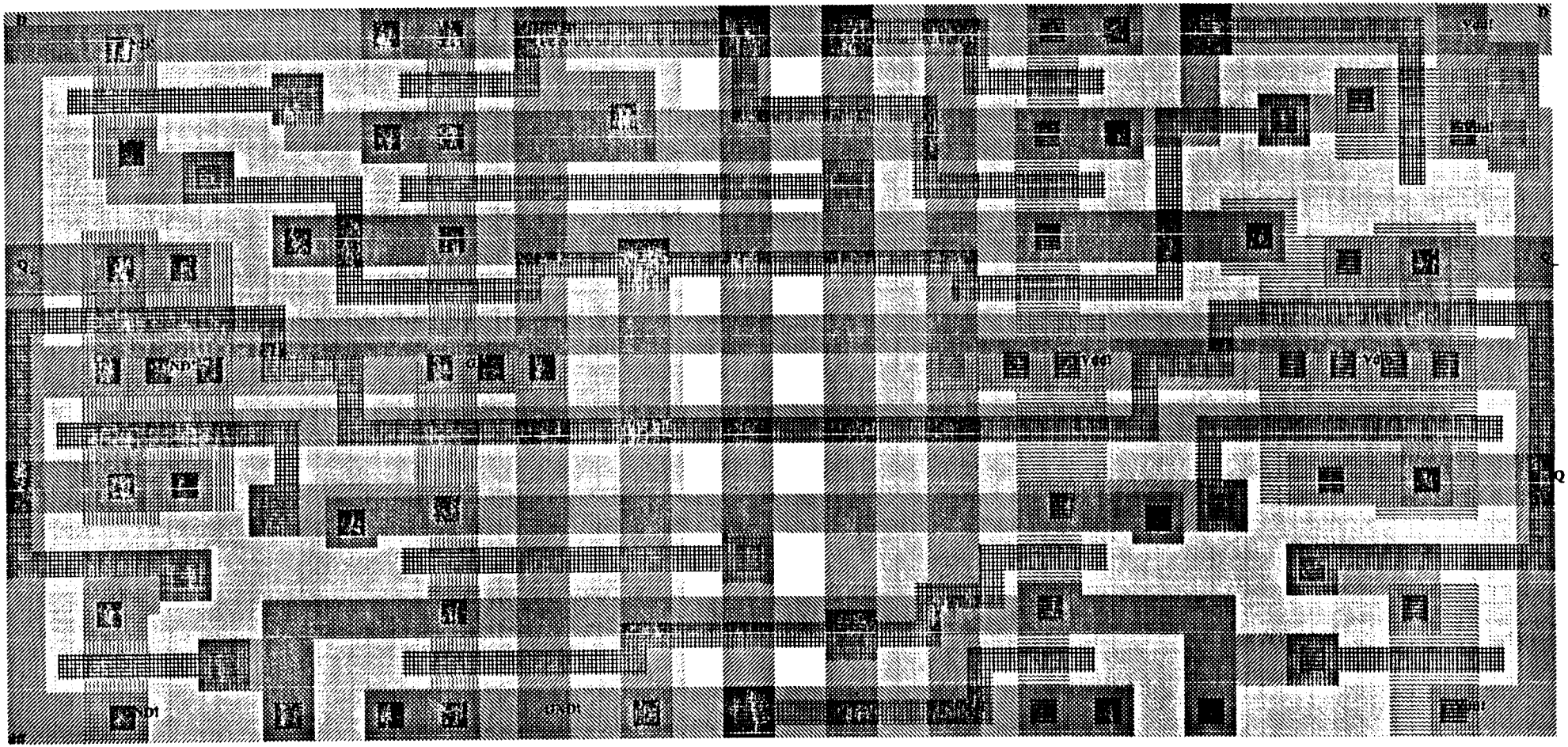
4 Inputs buffer

Plot 5



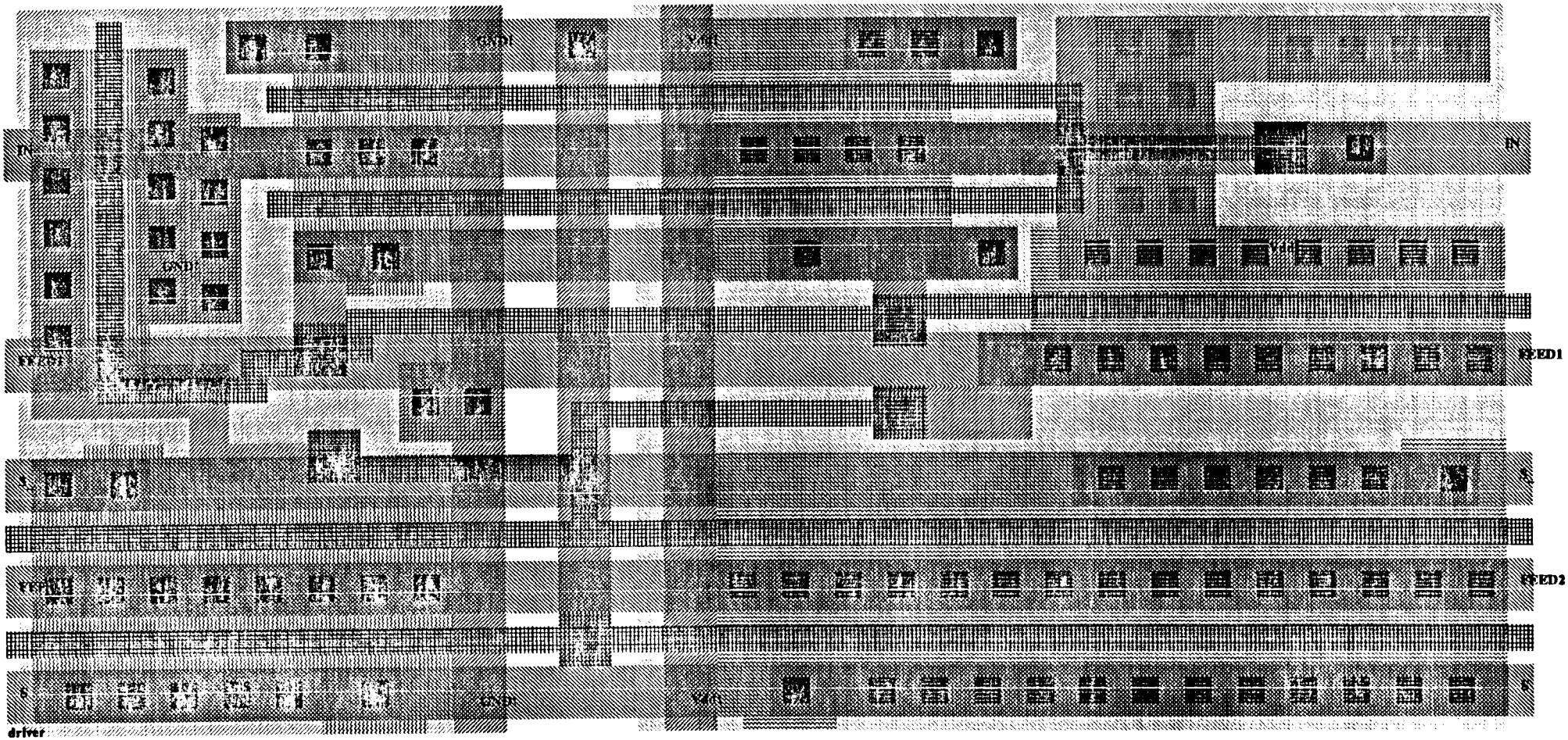
5 Inputs buffer

Plot 6



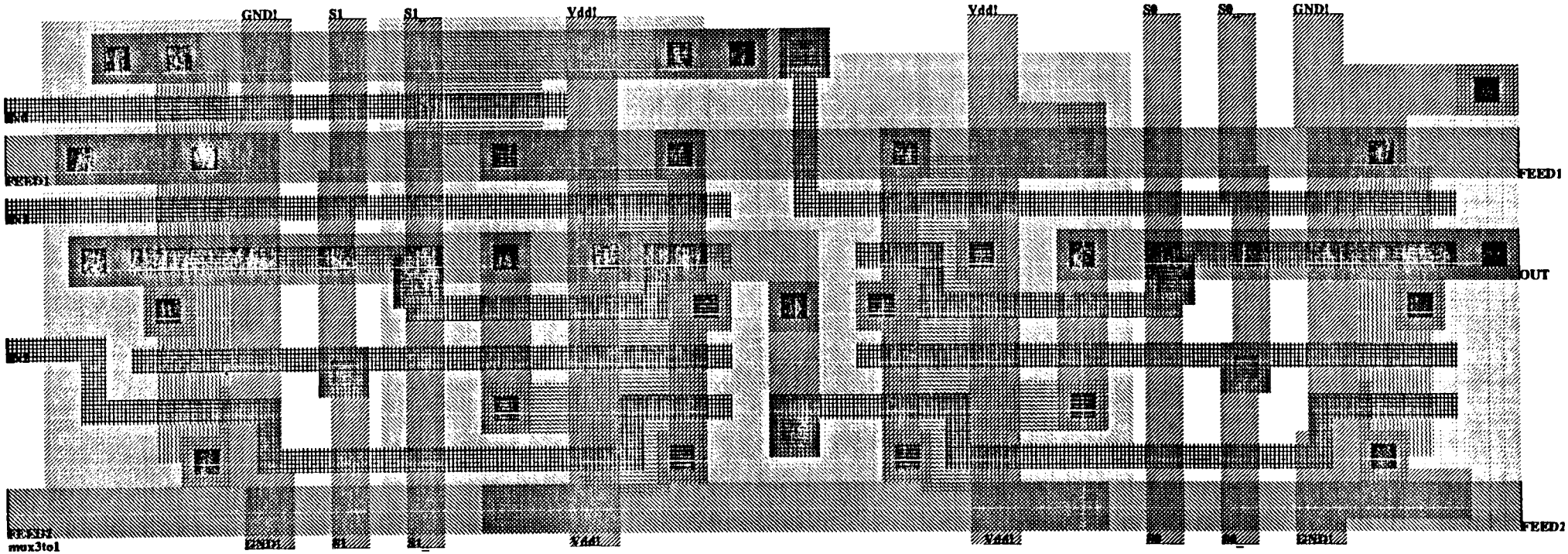
DFF

Plot 7



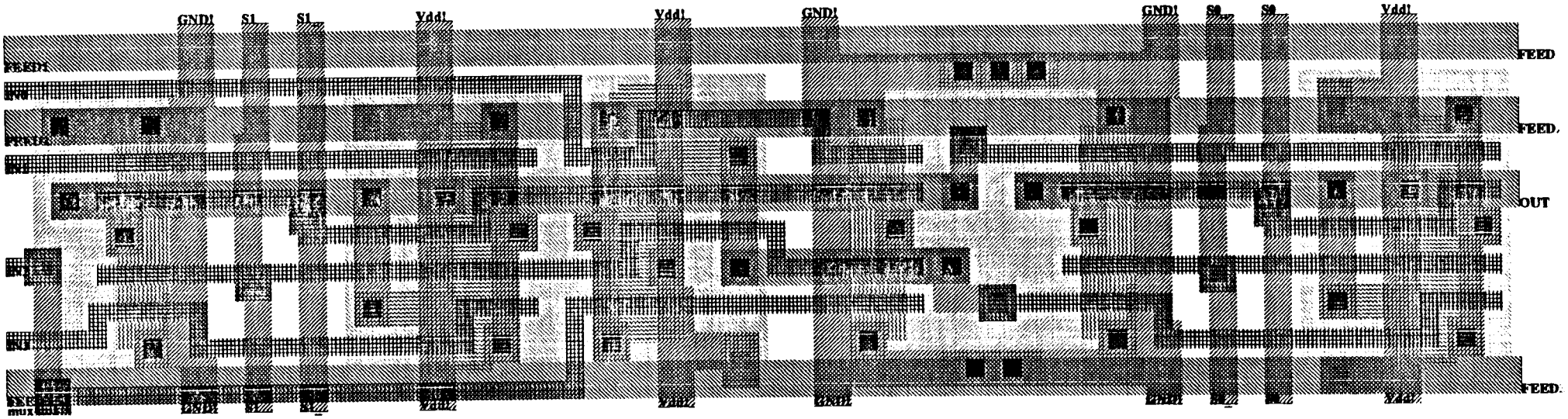
32 Lines Driver

Plot 8



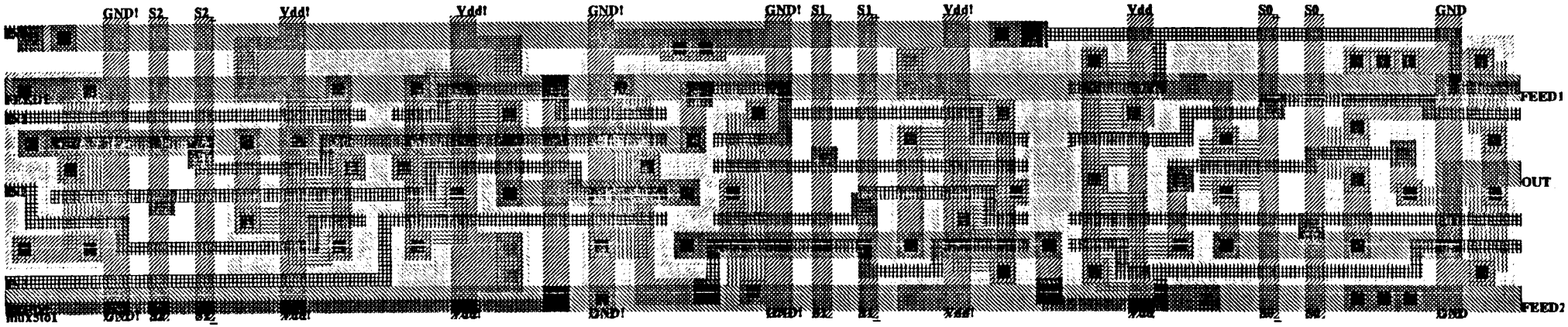
3 to 1 MUX

Plot 9



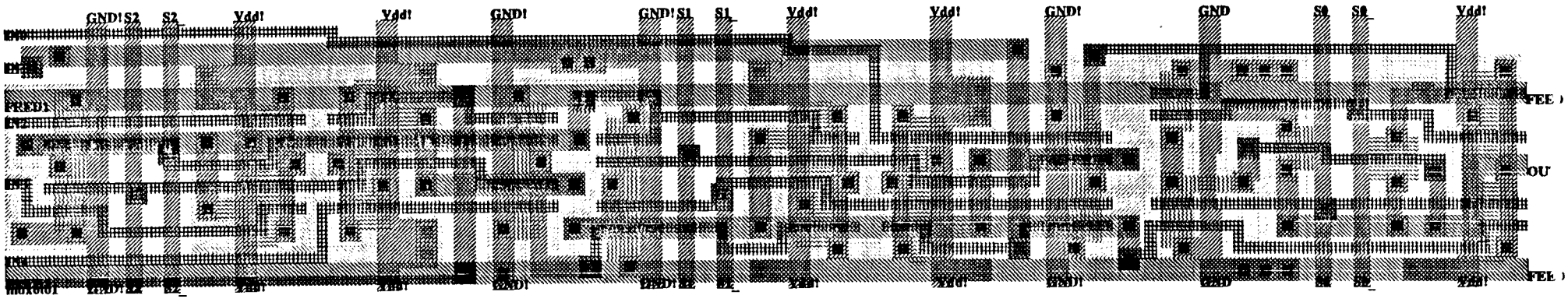
4 to 1 MUX

Plot 10



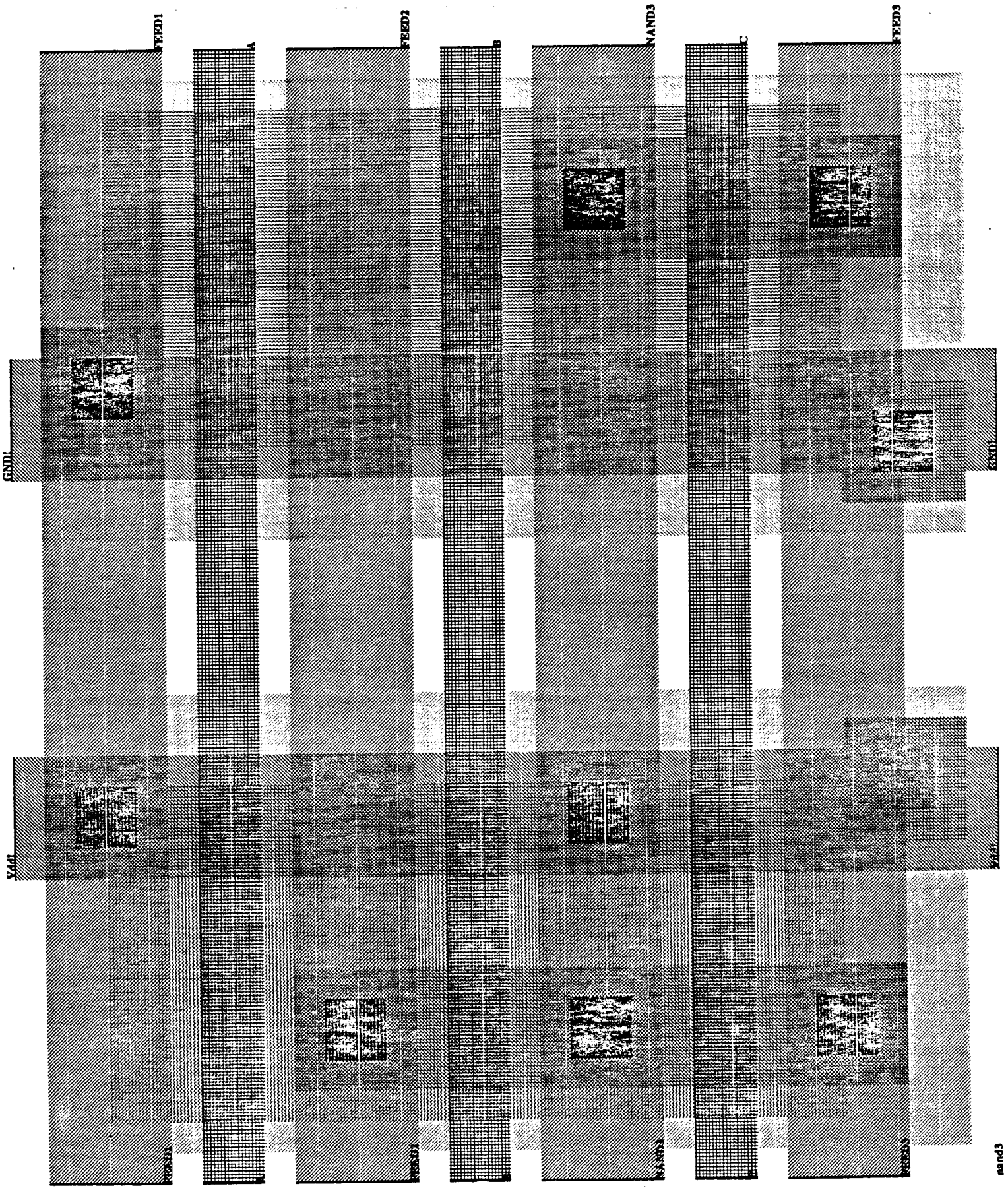
5 to 1 MUX

Plot 11



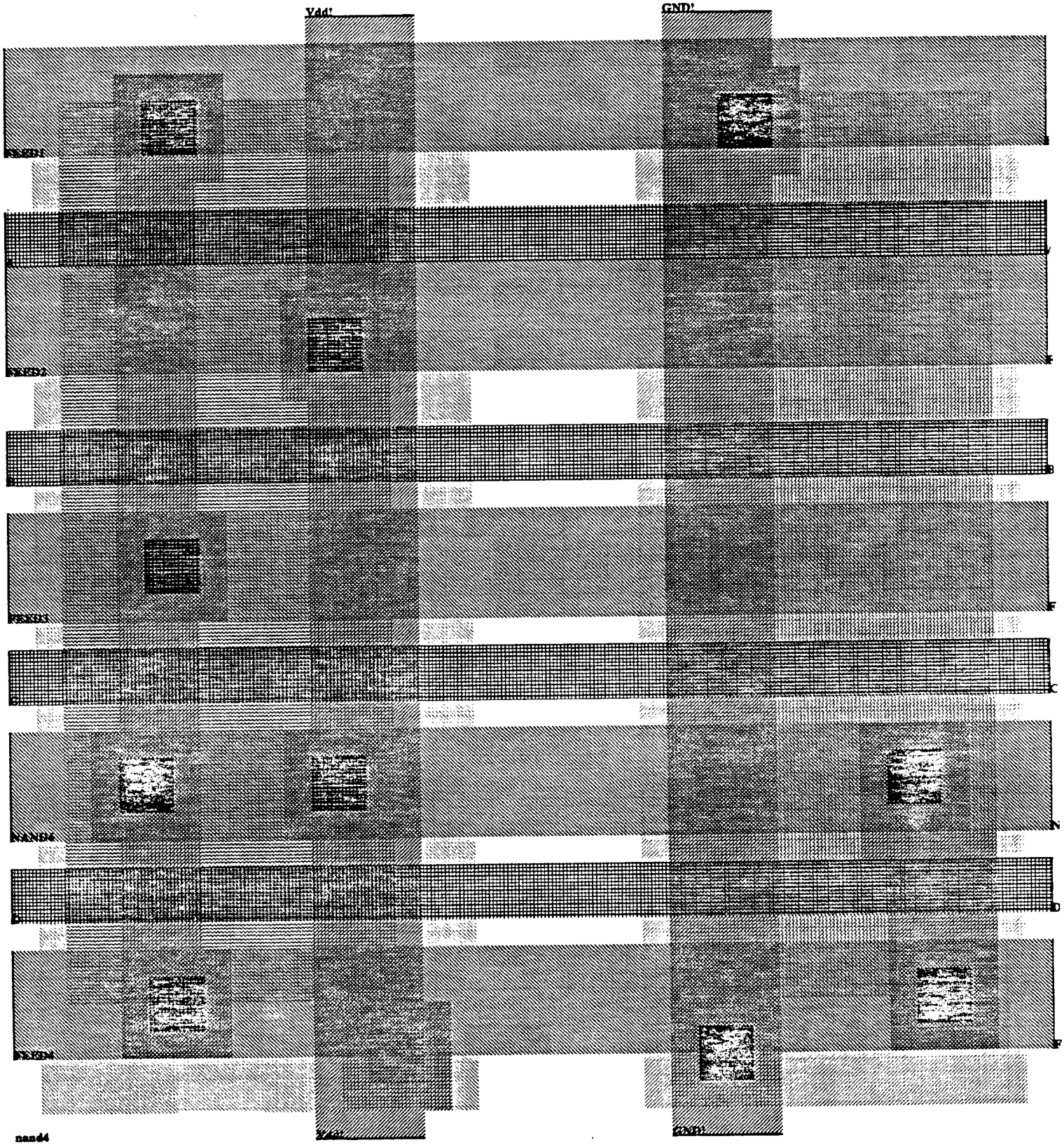
6 to 1 MUX

Plot 12



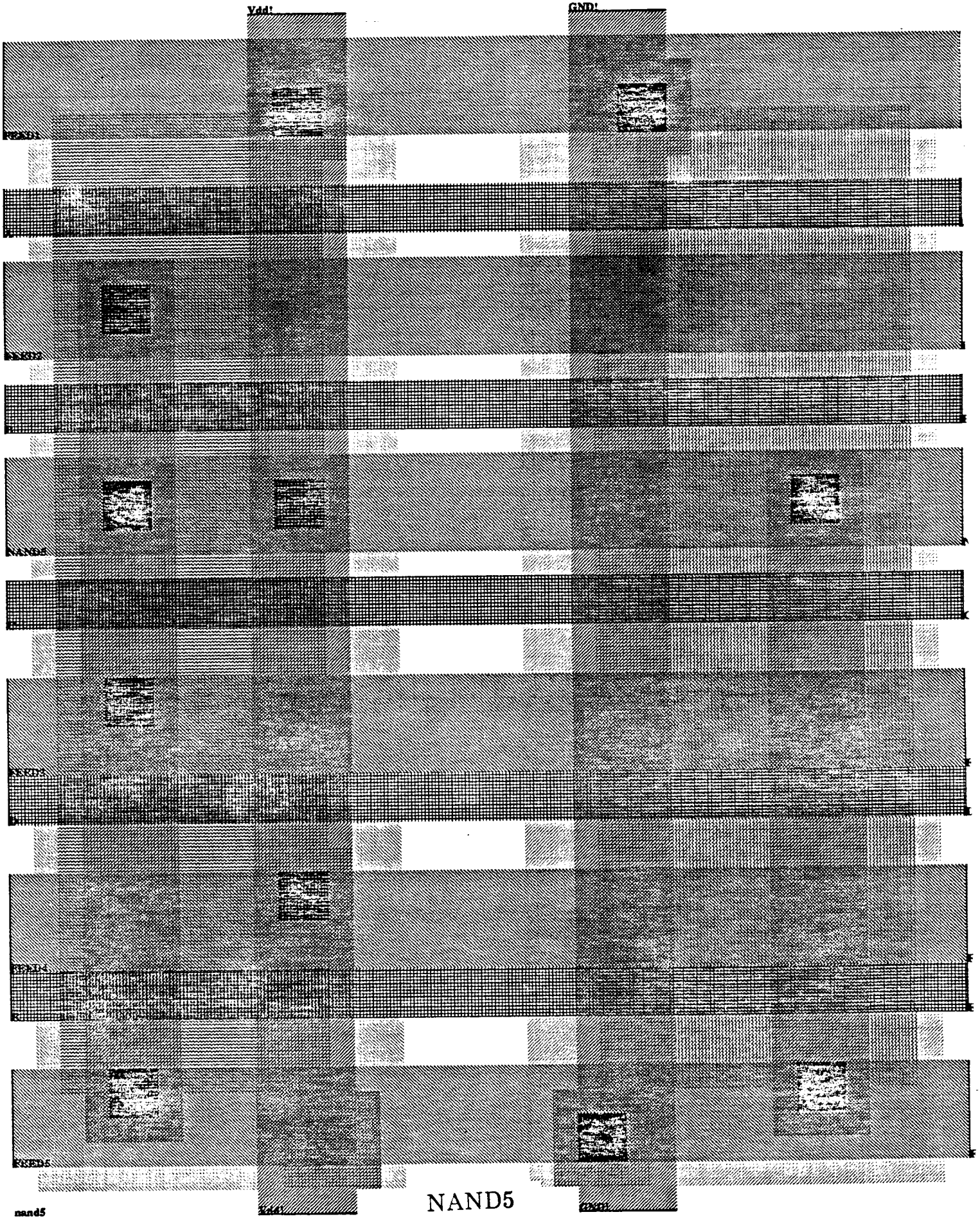
NAND3

Plot 13



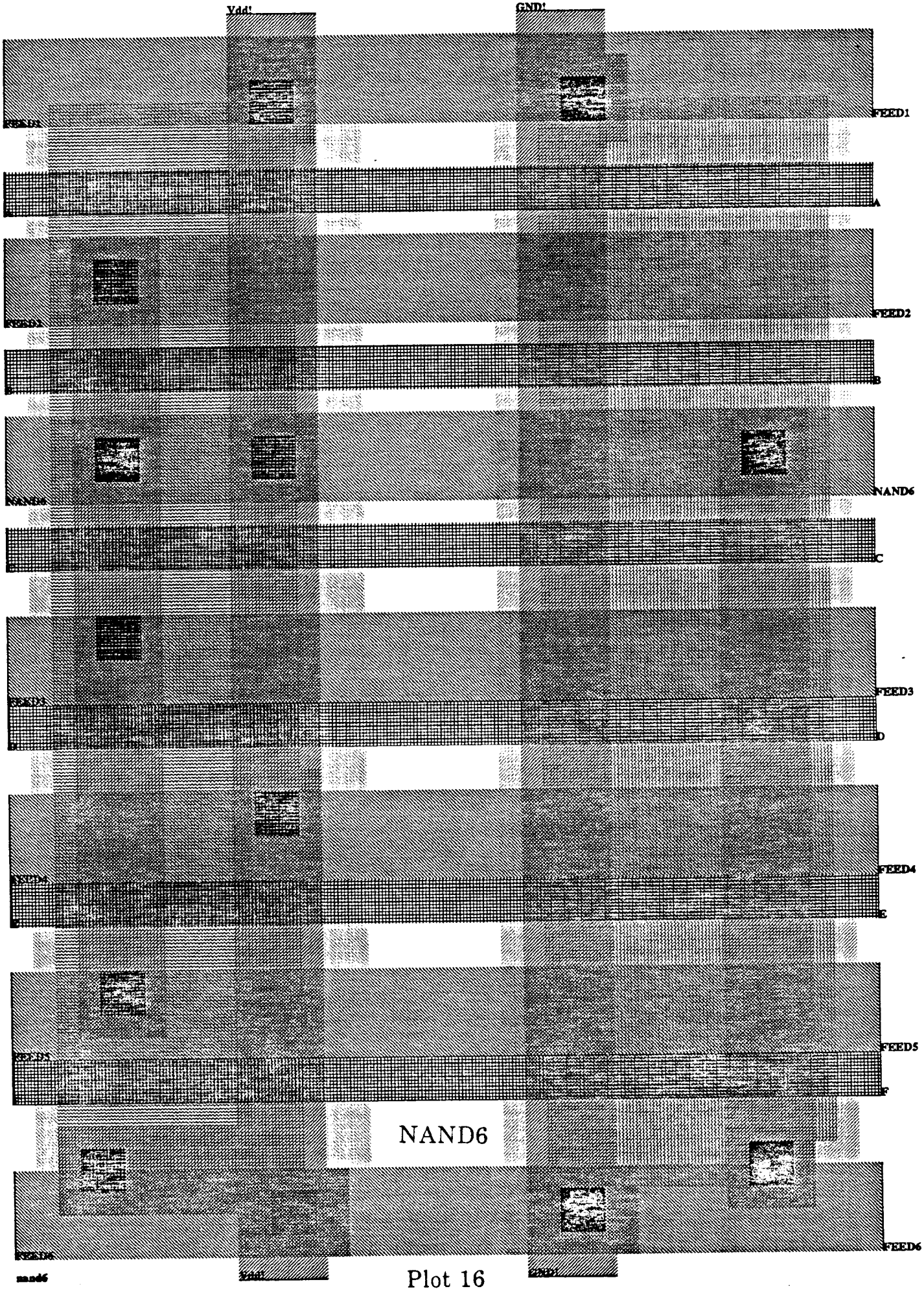
NAND4

Plot 14



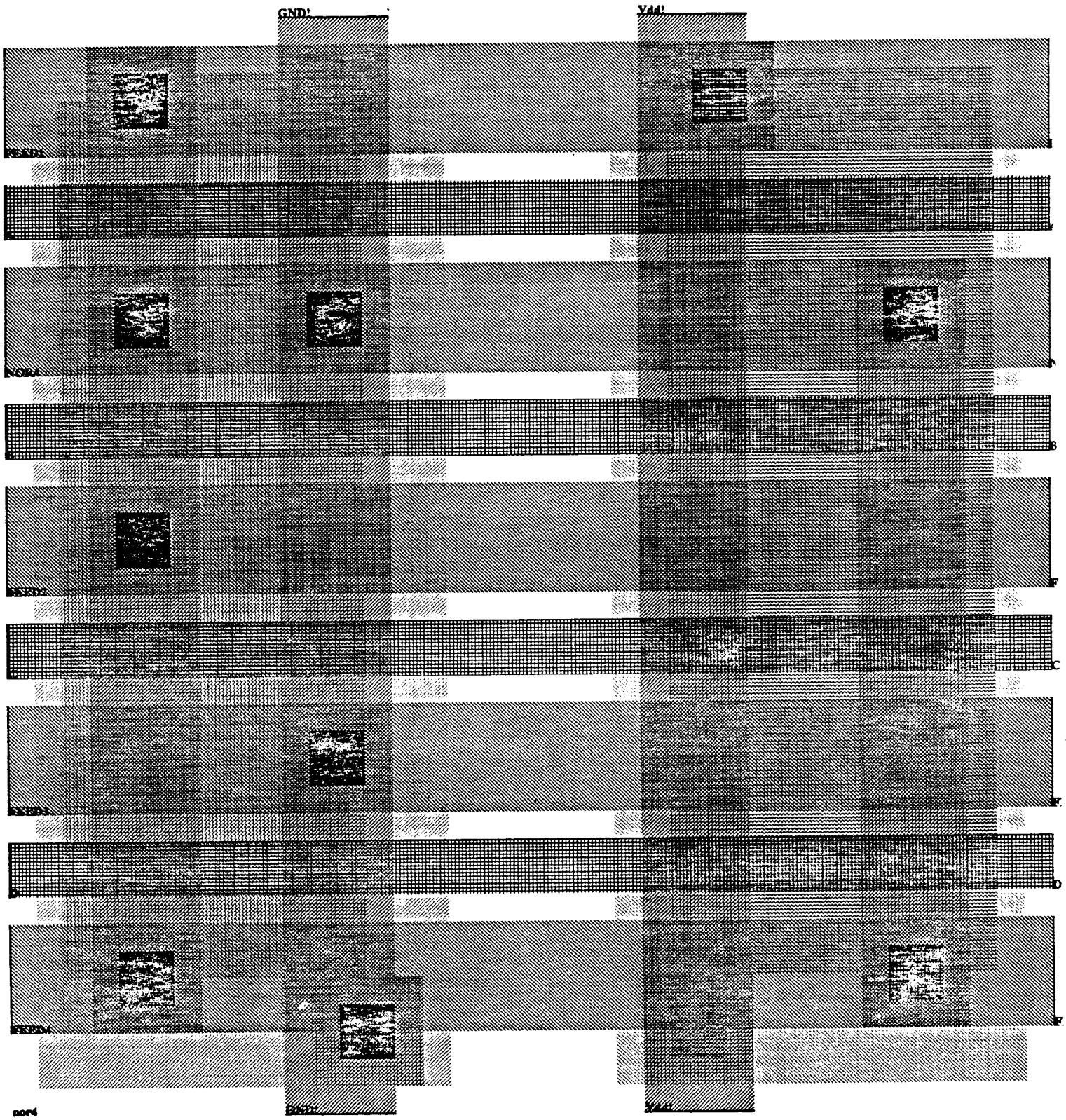
NAND5

Plot 15



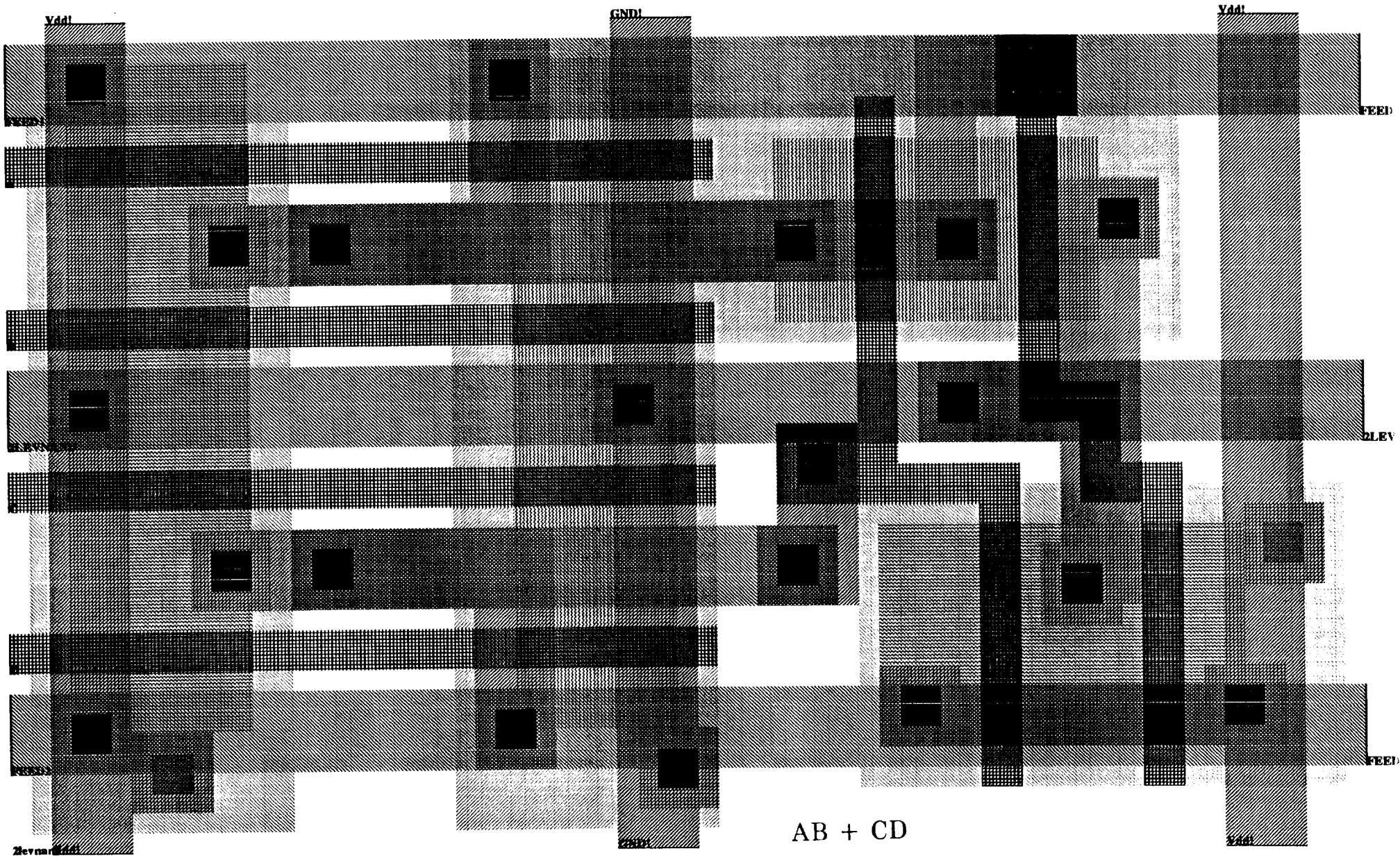
NAND6

Plot 16



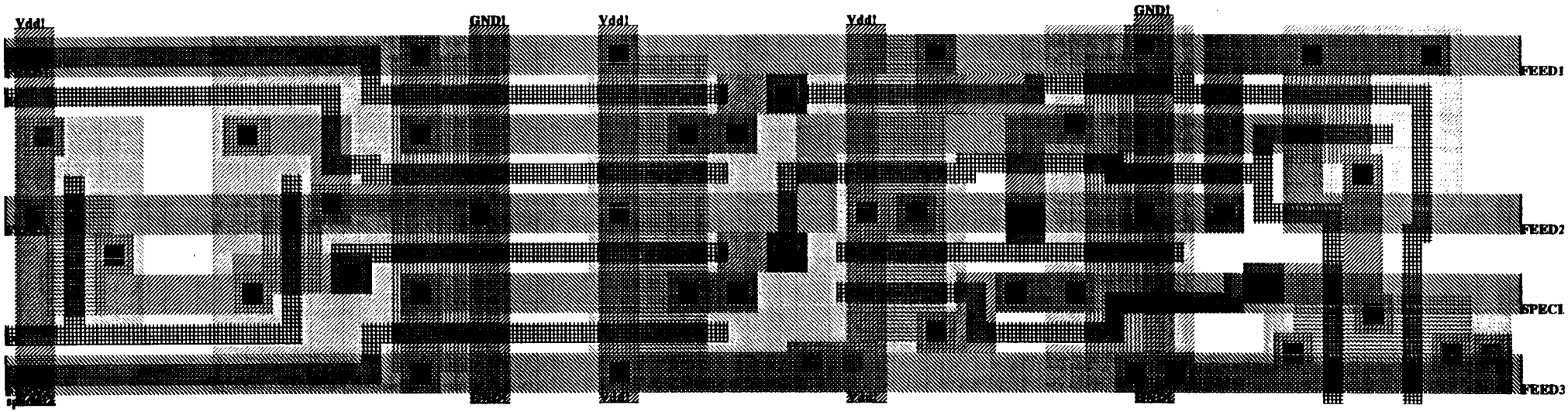
NOR4

Plot 17



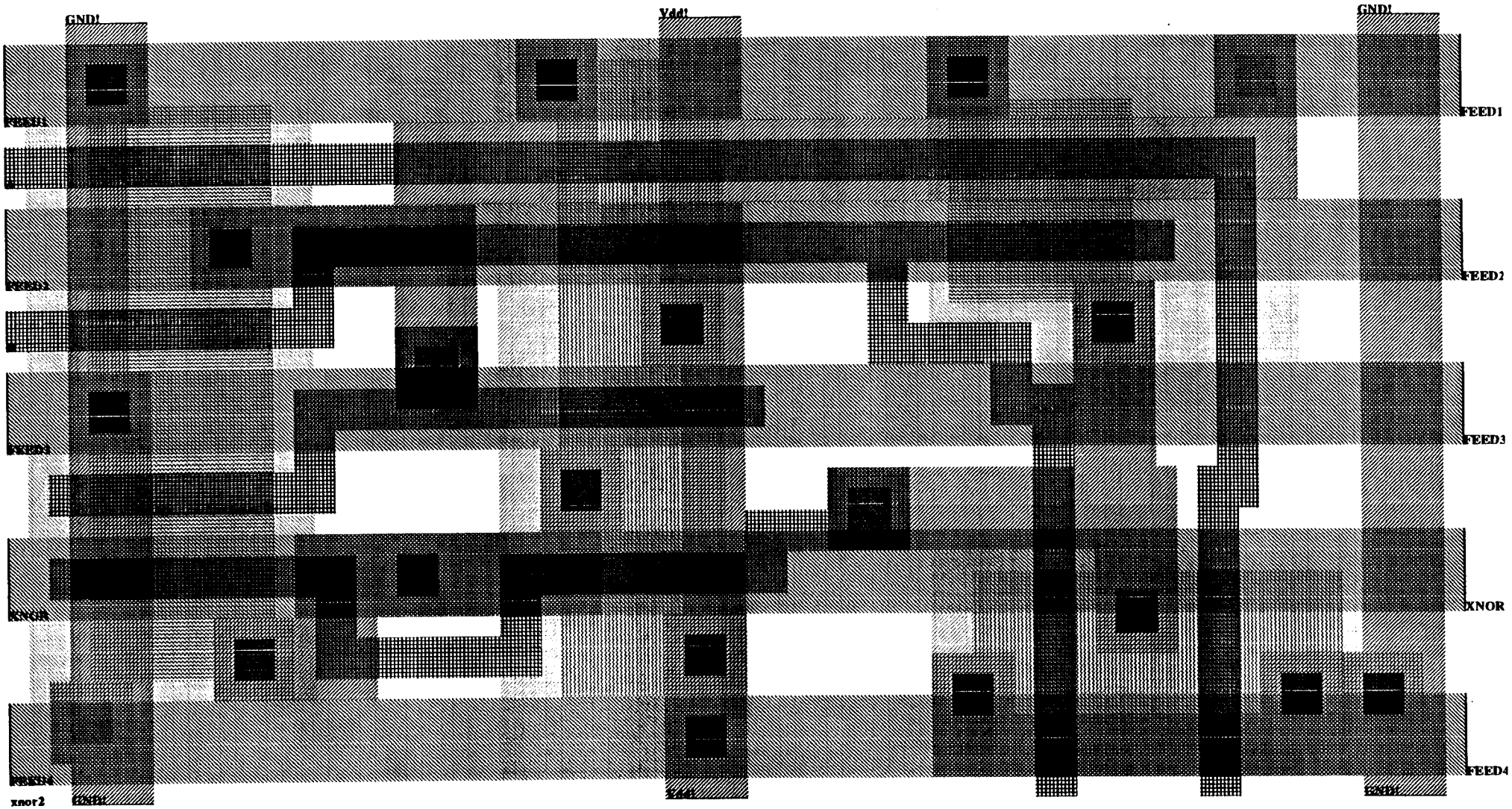
AB + CD

Plot 18



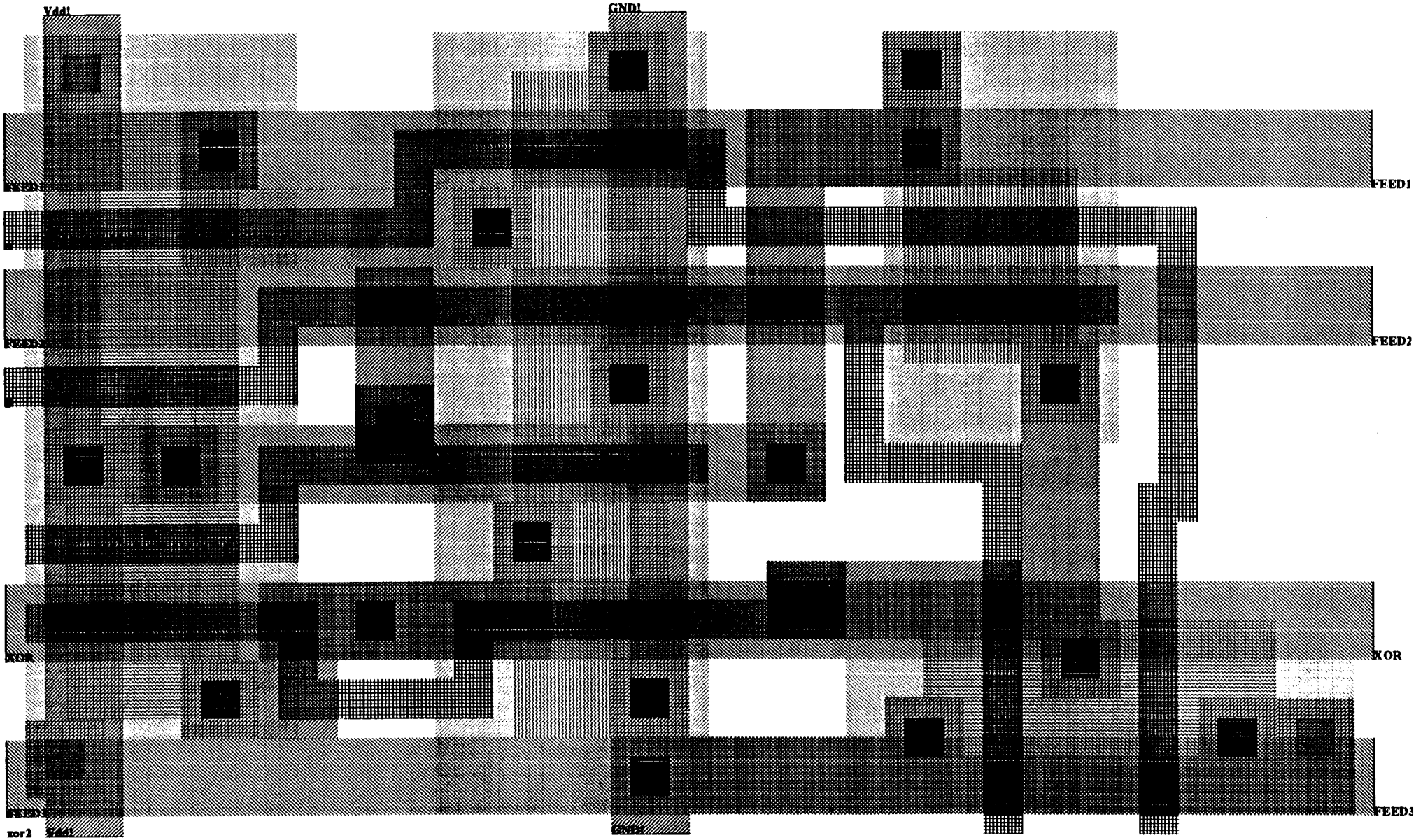
(C NAND M) XOR (/PB NAND GB)

Plot 19



XNOR2

Plot 20



XOR2

Plot 21