CMU-CS-84-101

Let's Design CMOS Circuits!

Part One

January 1984

Marco Annaratone
Department of Computer Science
Carnegie-Mellon University

The research was supported in part by the Defense Advanced Research Projects Agency (DoD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory under Contract F33615-81-K-1539.

Table of Contents

LIST OF FIGURES

List of Figures

Abstract

Designing CMOS circuits is quite different from designing nMOS circuits. This is especially true for CMOS Bulk. The following notes are not a course on CMOS design. They simply are a collection of information, basic concepts and methodologies aiming at simplifying the very first impact with complementary MOS design. A basic knowledge of nMOS design and nMOS CAD tools is assumed; no knowledge of CMOS is required. Therefore, this document does not give a very deep insight on each aspect of CMOS design but only a general overview of the problems this technology arises.

This brief survey on CMOS design is divided into two parts: part one introduces the basic concepts of CMOS technology, deals with some design methodologies and aims at giving all the necessary information to start designing CMOS chips. Part two, which will be available in 1984, will present experimental results and, hopefully, will be the complement and conclusion of part one.

1. Introduction

In this period (Fall 1983) MOSIS has started to regularly offer CMOS-Bulk runs (basically one each month). Time to design CMOS chips has come. As we shall see in this report, CMOS design is neither very easy nor free from drawbacks: why should we design CMOS chips, then? Because CMOS, whatever is its fabrication process (i.e. Bulk P-well, Bulk N-well, twin-tub, SOS, SOI) is one of the main technologies of the eighties. Actually, many commercial chips are already built in CMOS and an even greater number will be built in the future. CMOS is no more a technology good only for wrist watches, washing-machines or toys: floating point chips are built in CMOS, 32-bit microprocessors are built in CMOS, state-of-the-art A/D converters are built in CMOS. Time has come to design CMOS chips in the universities.

1.1. What is CMOS

CMOS means Complementary Metal Oxide Semiconductor. While in nMOS we have only one type of switching device, i.e. the n-channel (the depletion transistor acts as a resistor), in CMOS we have both the n-channel and the p-channel available. As we shall see later on, a problem in CMOS is not only to isolate similar devices among themselves, but also to isolate *different devices* among themselves (i.e. to isolate the n-channel's from the p-channel's); also for this reason, CMOS design is *significantly* different from nMOS design.

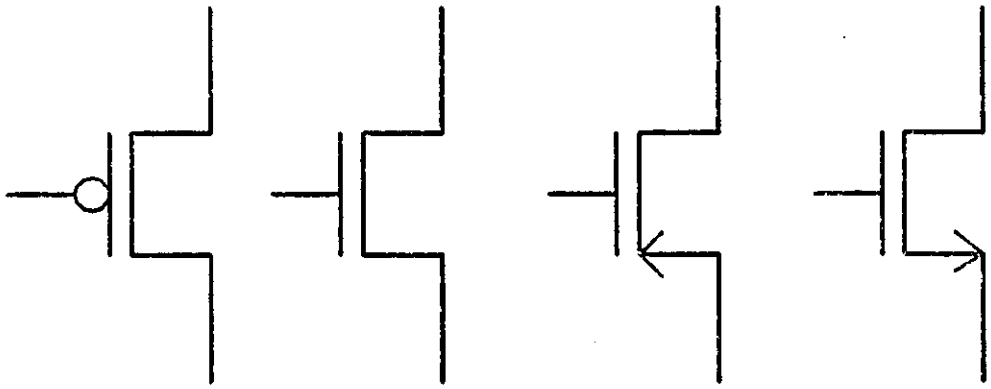


Figure 1-1: Two different symbols for p-channel and n-channel

In Fig. 1-1 two different representations of the n-channel and p-channel transistor are shown. The representation on the left will be used from now on. In Fig. 1-2 an nMOS inverter (left) and a CMOS inverter are shown. Note that the input signal, in the CMOS inverter, is connected also to the gate of the p-channel transistor, in a typical push-pull configuration. While in nMOS the pull-up is a depletion

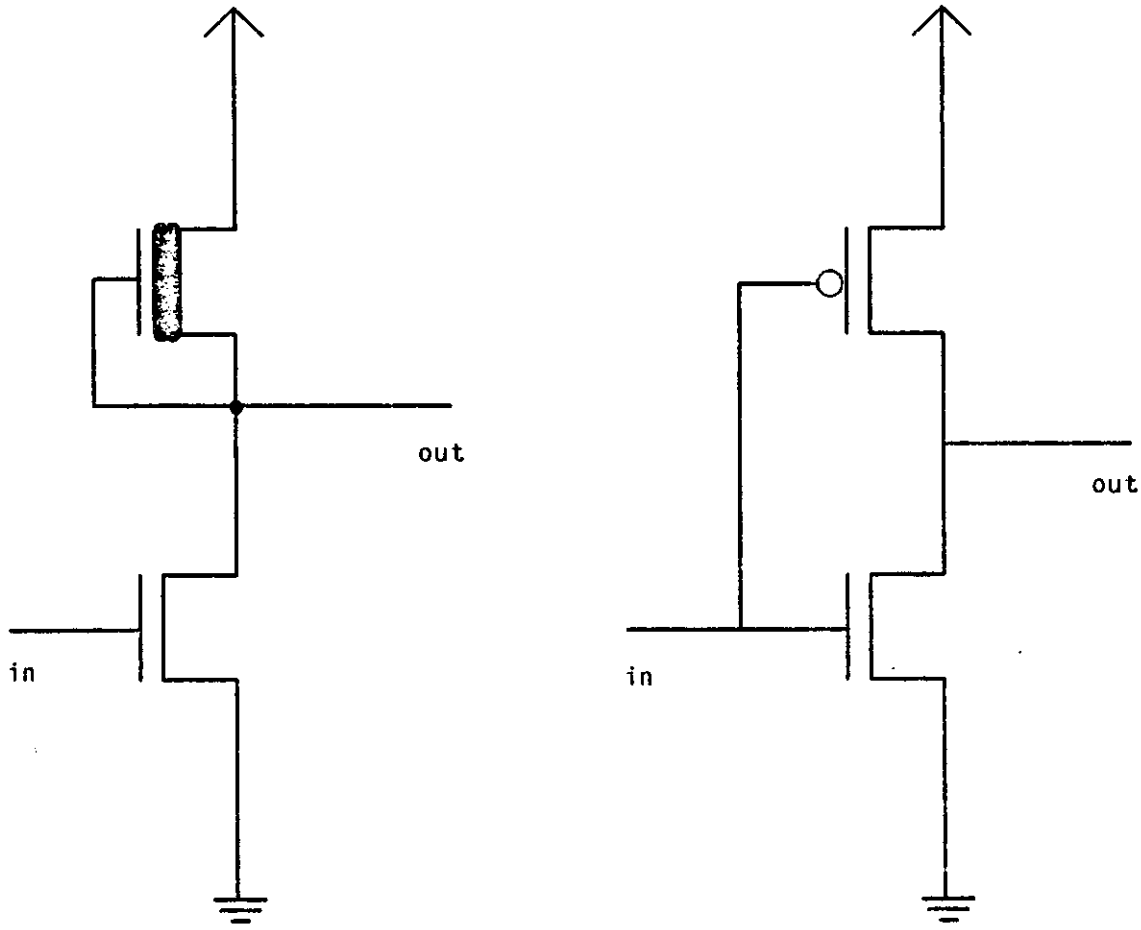


Figure 1-2: nMOS and CMOS inverters

device, basically a resistance and therefore acting as a current source, in CMOS we have an active pull-down (like in nMOS) *and an active pull-up:* the p-channel device.

This simmetry leads to the following results:

- Up-down and bottom-up transition delays are, theoretically, the same[1].
- One of the two transistors is always "on" and one transistor is always "off". In nMOS , when the pull-down is on (logic input = 1), its load is the depletion device that has a low resistance (several kOhm). Therefore, in this state, the nMOS inverter dissipates power, while the CMOS counterpart, being the pull-up off (several thousands of MOhm), does not dissipate

[1] Actually, the two delays depend on many parameters, e.g. the mobility of holes and electrons and the presence of the well. The former is a parameter of the process and can be very different for the two devices; the latter turns out to be a highly capacitive load (being a heavily doped region) and therefore up-down and bottom-up transitions see different loads.

power. A CMOS inverter dissipates only in the very short period (switching) when both devices are conducting.

- Unlike nMOS, the output of a CMOS gate makes a full excursion between Vdd and GND. This is important in order to increase the insensitivity to the noise ("noise margin"). It also allows design methodologies that uniquely characterize CMOS.

In the long range, CMOS should benefit from the scaling in feature size, while nMOS, basically for power consumption reasons, would create more problems. Some expert even claims that CMOS could be a good subnanosecond technology and 300 psec. gate delays have already been achieved. This is approximately one order of magnitude higher than a 0.7μ GaAs technology ($\tau \sim 60$ps).

While it seems still difficult to consider CMOS a "very fast" technology, even at very low feature size, its speed-power product will always be one of the most favourable, at least at room temperature.

1.2. Nand, Nor and Transmission Gate

In Fig. 1-3 a three-input NAND (left), three-input NOR (right) and a transmission gate are shown. The full complementarity between the pull-up and the pull-down stage and a redundancy in the structures are evident: the pull-up only (or the pull-down only) would be sufficient to implement the logic function. This observation leads to different, less redundant, implementations of the basic gates, as we shall see later on in this document. CMOS is a *ratioless* logic: this means that no special ratio between pull-up and pull-down is needed; the ratio influences only the dynamic behavior of the circuit. In other words, a wrong pull-up/pull-down ratio can still produce a working chip, presumably much slower than expected[2].

From Fig. 1-3 and Fig. 1-4 it is evident that CMOS gates are much more complex (and area consuming) than their nMOS counterparts; this is true if we use *static* gates only, i.e. one pull-up (p-channel) for each pull-down (n-channel). Luckily, there are other design methodologies overcoming this problem that, otherwise, would have made this technology useless in all the applications in which the noise margin and/or the power consumption were not a major constraint (see chapter 2).

In Fig. 1-4 a comparison between a CMOS inverter and an nMOS inverter (up) and between a CMOS nor gate and an nMOS nor gate is presented. nMOS is a 3μ feature size MOSIS process and CMOS is a 3μ feature size MOSIS process. Buried contact and P-well are depicted in the same way for sake of simplicity.

[2] With "wrong", it is meant "reasonably wrong", obviously!

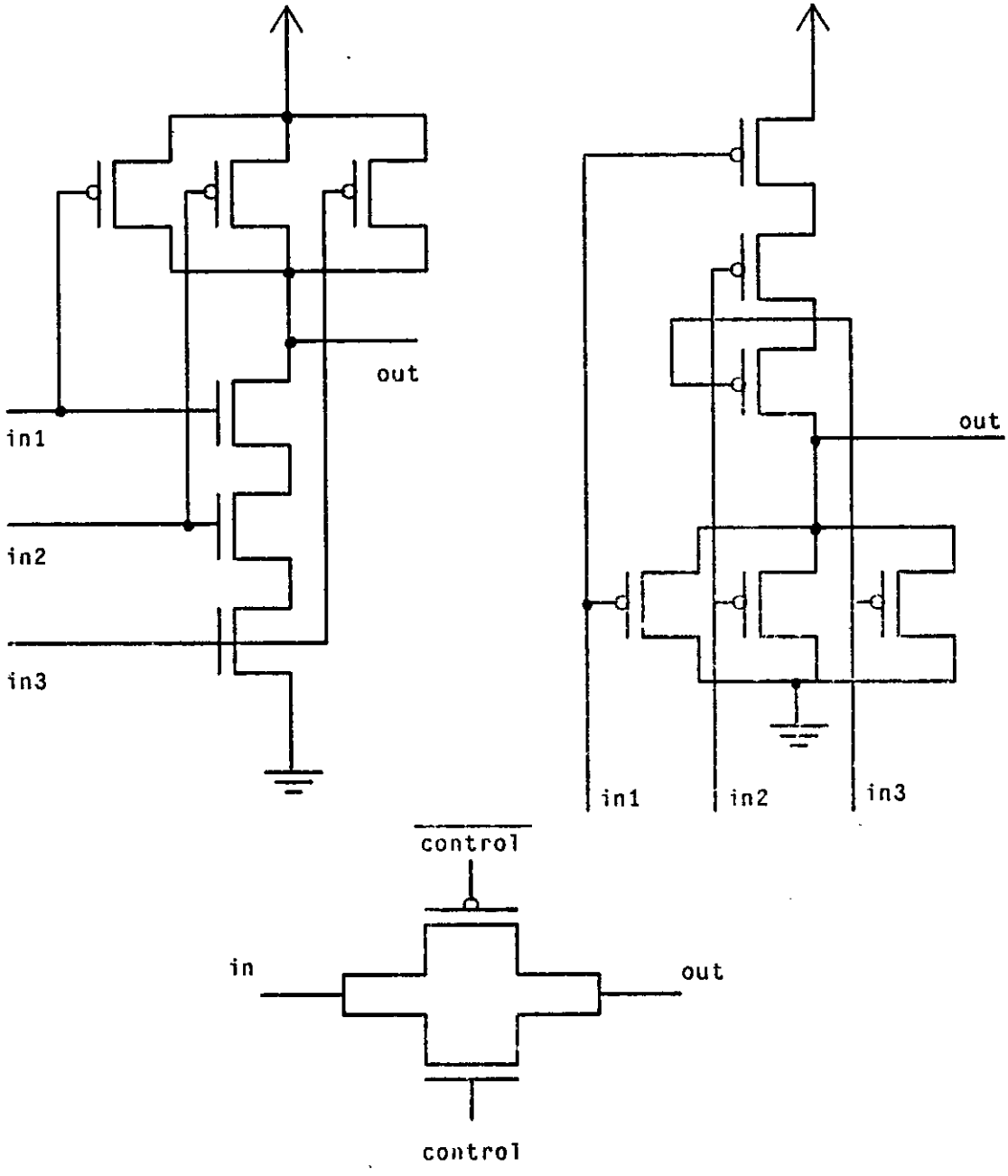
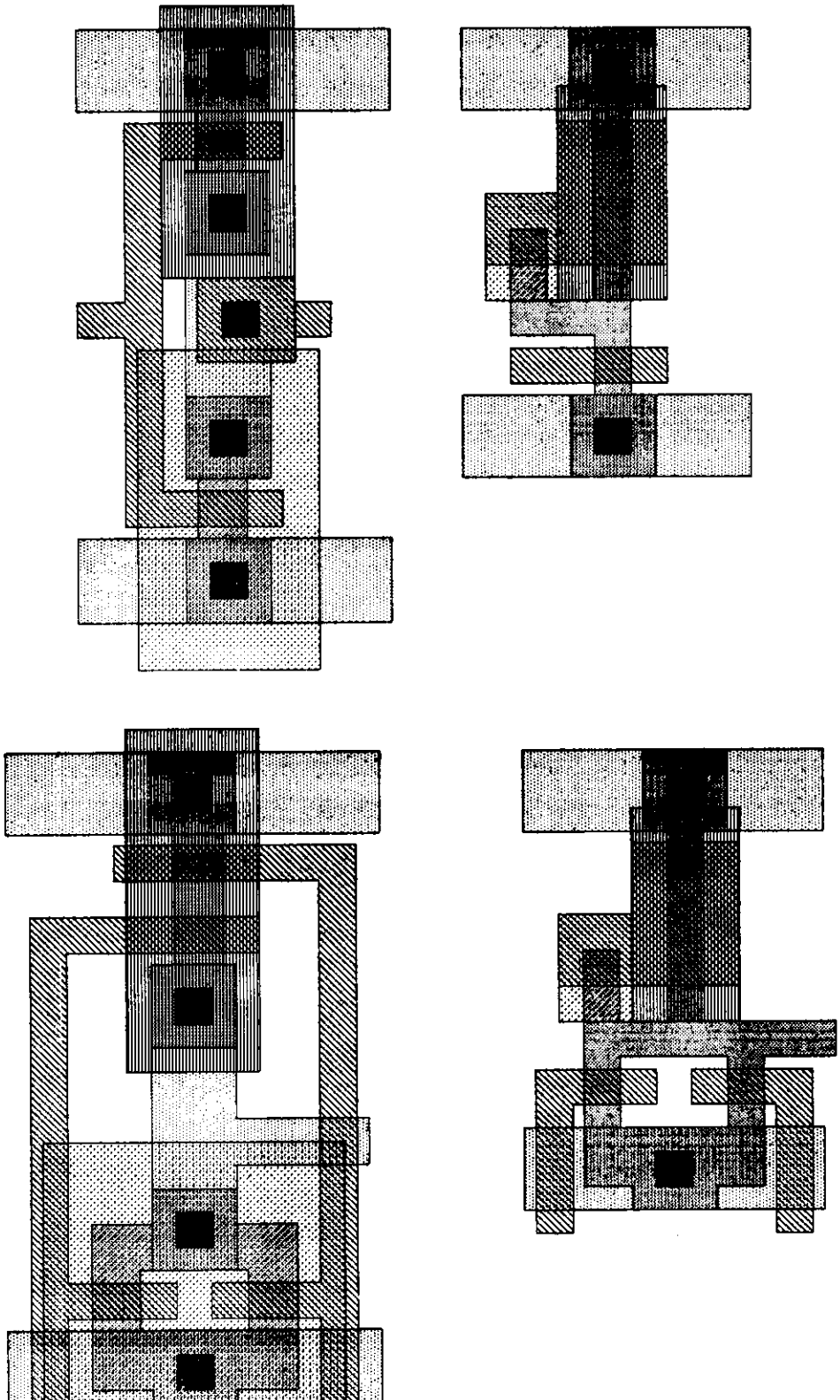


Figure 1-3: CMOS Gates

Figure 1-4: CMOS inverter and nMOS inverter (up); CMOS nor and nMOS nor



This comparison is normally unfair and is more qualitative than quantitative. The ratio between the area of the CMOS and nMOS inverter has almost nothing to do with the circuit density factor of the two technologies. The figure simply aims at showing that *static* CMOS is more area-consuming than *static* nMOS.

As far as the CMOS transmission gate, the parallelism with the nMOS pass-transistor is evident. There are two major differences, however:

1. The CMOS transmission gate, if "bilateral" as in Fig. 1-3, does not need a level-restoring logic circuitry (the well-known 8:1 ratio in nMOS). It is evident that the transmission gate can also be implemented "unilaterally" (i.e. with the n-channel device or the p-channel device only): in this case, a level-restoring gate is needed: this does *not* mean that a special ratio (e.g. 8:1) must be used in the restoring-level circuitry; it simply means that *the signal does not make a full excursion any more and a gate is needed to restore the full excursion.* However, the use of unilateral gates, unless carefully considered, can lead to serious mistakes: see section 2.4 for details.

2. The CMOS transmission gate needs *both* control signal polarities; this leads to a greater complexity in the control structure if compared to the simplicity of an nMOS pass transistor. Actually, many CMOS chips are implemented with unilateral transmission gates: although this approach partially affects the chip's performance, the saving in area can make this choice attractive.

It must be pointed out that, whatever strategy of implementation is chosen, nMOS circuit density is higher than CMOS's. More precisely, nMOS is denser than CMOS-SOS which is denser than CMOS-Bulk.

The delay of a minimum-size inverter loaded by another minimum-size inverter is:

$$\tau = 1/2\,(L_n/v_n + L_p/v_p) + 1/2\,(C_l V_{tn}/I_n + C_l V_{tp}/I_p)$$

where:

- L_n and L_p are the channel lengths of, respectively, the n-channel and the p-channel transistor;
- v_n and v_p are the carrier velocities;
- V_{tn} and V_{tp} are the threshold voltages;
- C_l is the load capacitance;
- I_n and I_p are the saturation drain currents.

Let $L_n = L_p = 3\mu$, $V_{tn} = V_{tp} = 1$ V, $C_l \sim 30$fF[3].

[3]The junction capacitance is 6fF and the parasitic capacitance is 18fF: 2 x 6 + 18 = 30.

With $W_n = W_p = 4\mu$, $C_{ox} = 5.7 * 10^4$ pf/cm^2, $I_n = 450\mu A$ @$V_{GS} = 5$ V and $I_p = 50\mu A$ @$V_{GS} = 2$ V, from:

$$v_n = I_n/(W_n C_{ox}(V_{GS} - V_{tn}))$$

and

$$v_p = I_p/(W_p C_{ox}(V_{GS} - V_{tp})),$$

we have:

$$v_n \sim 49000 \text{ m/s and } v_p \sim 21900 \text{ m/s},$$

and, finally, $\tau \sim 450$ ps (for CMOS-Bulk only).

For a more comprehensive analysis, the reader is referred either to [22] or to the introduction to CMOS-SOS by Charles Seitz[4].

[4]Available from MOSIS; it can be found also in /usr/cmos/sos.notes

2. Design Methodologies

This section does not aim at giving a complete survey on CMOS design methodologies; some simple examples will be presented and attention will be focused on aspects that are peculiar to CMOS. More precisely, the following topics will be dealt with:

- Static vs. dynamic design;
- miscellanea: xor, latches, full-adders, multipliers etc.;
- PLA's and RAM's;
- analog circuits.

As far as high-level design is concerned, a chip can be build out of a library of *validated* modules and with automatic (although constrained) placing and routing. This would allow to reduce the turn-around and experiment different architectural approaches. This is the approach pursued by the speech chip group (Anantharaman, Annaratone, Bisiani) (even if the technology is still nMOS).

Another very promising approach is the use of gate arrays (polycell). Although this methodology would indeed require at least a second metal layer, a number of interesting experiments could be carried on using the process presently offered by MOSIS (a second metal layer is in the future, however). The definition of the cell is not an easy task. The resistance of contacts, the sheet resistance of poly lines and diffused regions are usually critical parameters. A 6K-gate CMOS gate array is presented in [20]. Four layers are reserved for user personalization. Presently, 8K-gate chips are available. Generally speaking, the technology used to implement CMOS gate arrays resembles the fabrication process for RAM's.

What is common to both methodologies (i.e. library of cells and gate arrays) is a bad utilization of the chip area. On the other hand, there are many noticeable advantages: faster development, more reliable design, automatic (or computer-assisted) placement and routing.

Designing with CMOS gate arrays could be, at CMU, made much easier by the availability of the Daisy system (together with the usual VLSI tools). This idea is really worth being explored in the next future.

2.1. Static vs. dynamic design

Static-gate CMOS design suffers from three major drawbacks:

- it is area consuming;
- it can be slow: in a gate, the p-channel's are in parallel with the n-channels sharing the same gate: hence, also their input capacitance is in parallel;

- a static CMOS gate is intrinsically redundant, because it duplicates the functionality in both the pull-up and the pull-down section.

Different approaches can be pursued to reduce the area and the redundancy and to increase the speed:

1. Extensive use of transmission gates to build up logic functions (i.e. still a static-gate CMOS design);

2. nMOS-like style of design (again, static);

3. dynamic logic circuit design.

While all these approaches often reduce the area occupation and the redundancy, the factor of speed-up they can provide depends heavily on the application and the overall architecture of the chip; therefore, it is not true, for instance, that both transmission gate-intensive circuits and dynamic logic circuits are *always* faster than their static counterpart (this is especially true for dynamic logic design).

Using transmission gates means, moreover, that we always need both control signal polarities to drive a transmission gate. The number of wires can therefore be high and, if one metal layer is available, routing can be very-area consuming. There are many ways to avoid the use of static gates. The simplest one, which is still a static approach, is the use of an nMOS-like structure (see Fig. 2-1); this can be done at the expenses of power consumption, as we have a dc path to ground. On the other hand, the saving in area is considerable, redundancy is set to zero and the input capacitance decreases.

A more effective solution is to use dynamic logic. In Fig. 2-2 different "dynamic" implementation of a three-input NAND gate are shown. Among the different approaches presented, "domino logic" is maybe the most well-known. A good survey on dynamic logic design with an introduction to domino logic can be found in [8]. The circuit shown in Fig. 2-1 can be implemented, with domino logic, as depicted in Fig. 2-3. Basically, the circuit is the logic AND between the boolean function we want to implement and a "control" signal (clock). When the clock is low, precharging is performed; when the clock goes high, the evaluation takes place. This clock is common to all the blocks and, therefore, during the evaluation, the signals "ripple through" all the chip, as though the logic were purely static. If the final output is not consumed by off-the-chip circuitry, static latches are needed to temporarily store the information. The inverter is a static CMOS buffer. All the input nodes, during precharging, are low; during the evaluation they can make one transition only (i.e. bottom-up).

It must be pointed out that domino logic is not a complete general-purpose solution. The structure shown in Fig. 2-4 is not well-suited for domino logic and, if implemented via domino logic , it might be more area consuming than if it were implemented statically. The problem can be solved, at a different level of abstraction, by re-organizing the overall logic function; however, this might lead to a

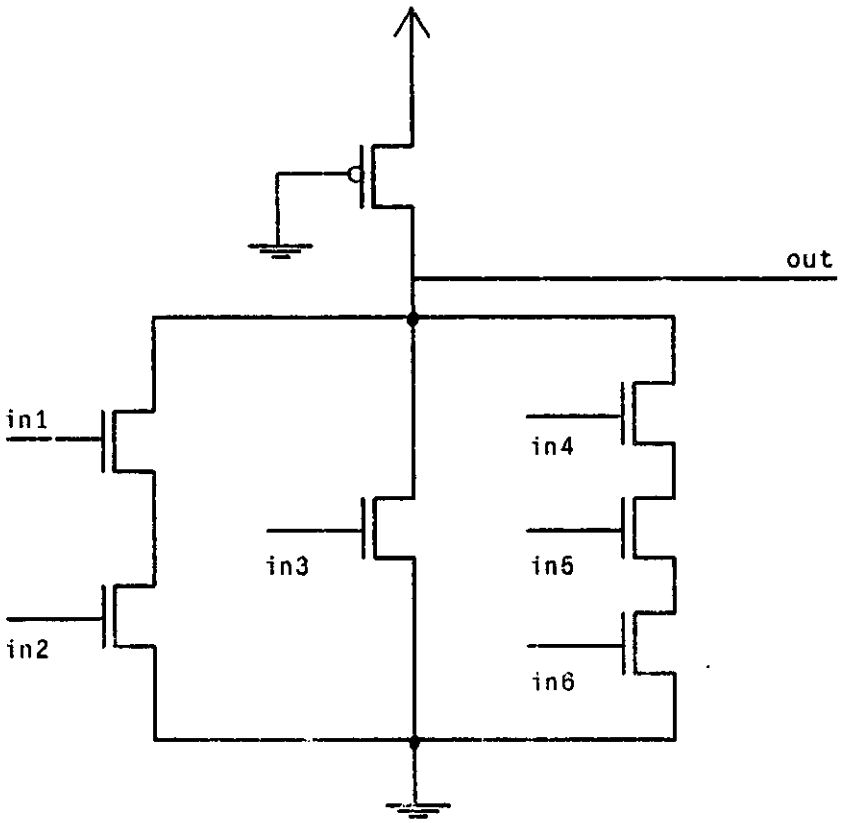


Figure 2-1: Static, nMOS-like, structure

slower execution.

2.2. Miscellanea

In Fig. 2-5 a straightforward topology for an exclusive-OR is shown (up). If inputs are available with both polarities *and SOS is used*, a more effective implementations is shown in Fig. 2-5 (down).

Latch design can be critical. The gain of the gates has to be carefully matched, otherwise the latch does not work. An example is shown in Fig. 2-6: all the units are in micron. If minimum-size inverters were used in the loop, the *latch could not work*. Before using a new latch in your chip, you should always simulate it with SPICE. Alternative topologies are shown in Fig. 2-7.

It is interesting to notice that it is very simple to design a muller element. Its scheme is shown in Fig. 2-8; in this scheme the element is dynamic: if a static muller element is needed, a static flip-flop can be used (e.g. cross-coupled inverters).

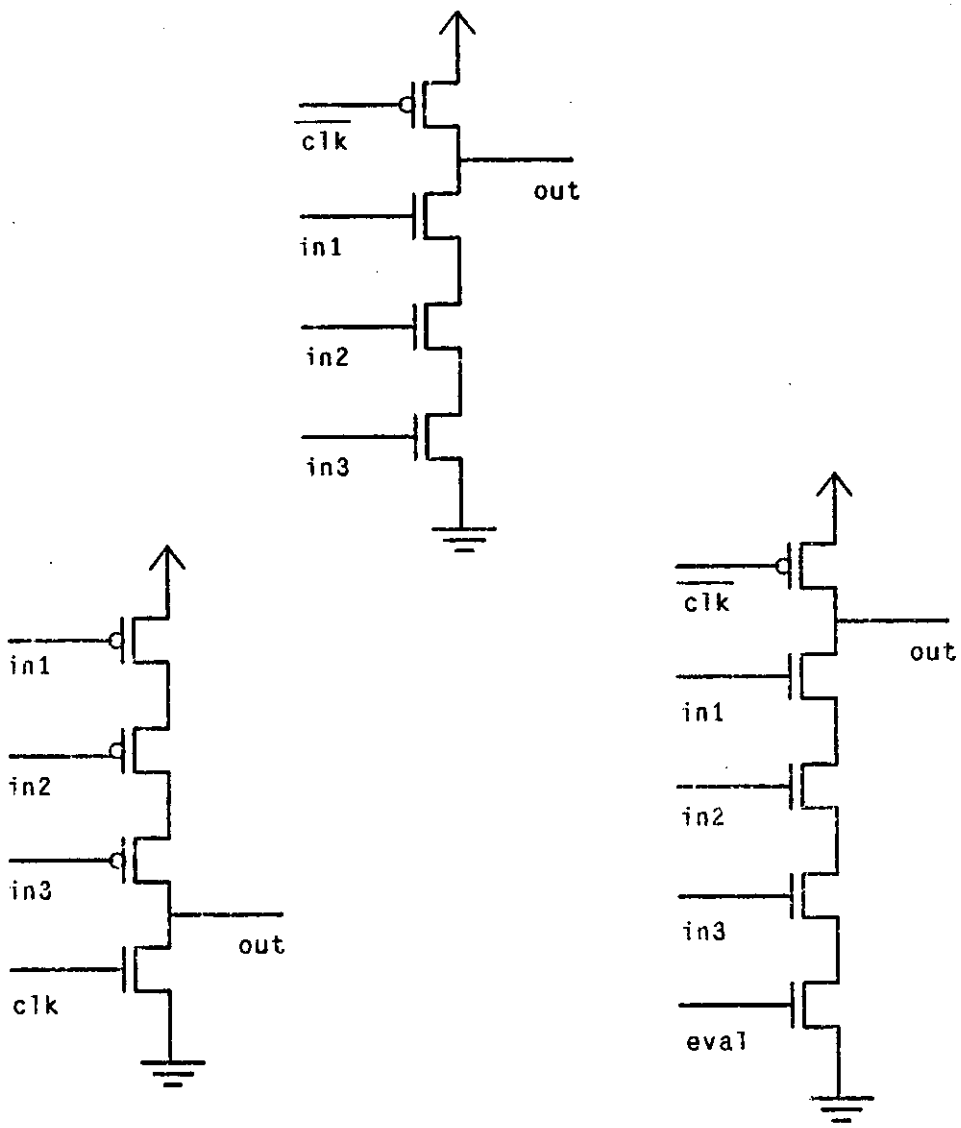


Figure 2-2: Different dynamic approaches to implement a 3-input NAND gate

Full-adders can be implemented in many different ways in CMOS. The circuit here presented is interesting for its structure, which is typical of CMOS. The full-adder scheme (taken from [10]) is shown in Fig. 2-10. It is worth noticing that two inverters in the second stage are powered by the output signal of a previous inverter. This is possible only in CMOS, because the signal makes a full excursion between Vdd and GND. This is a typical example of a transmission-gate intensive design. In Fig. 2-9 an nMOS full-adder (left) ($\lambda = 2\mu$) and a CMOS full-adder are shown. Although the nMOS layout was accurately studied, while the CMOS layout was more or less "experimental", the difference in size seems to be

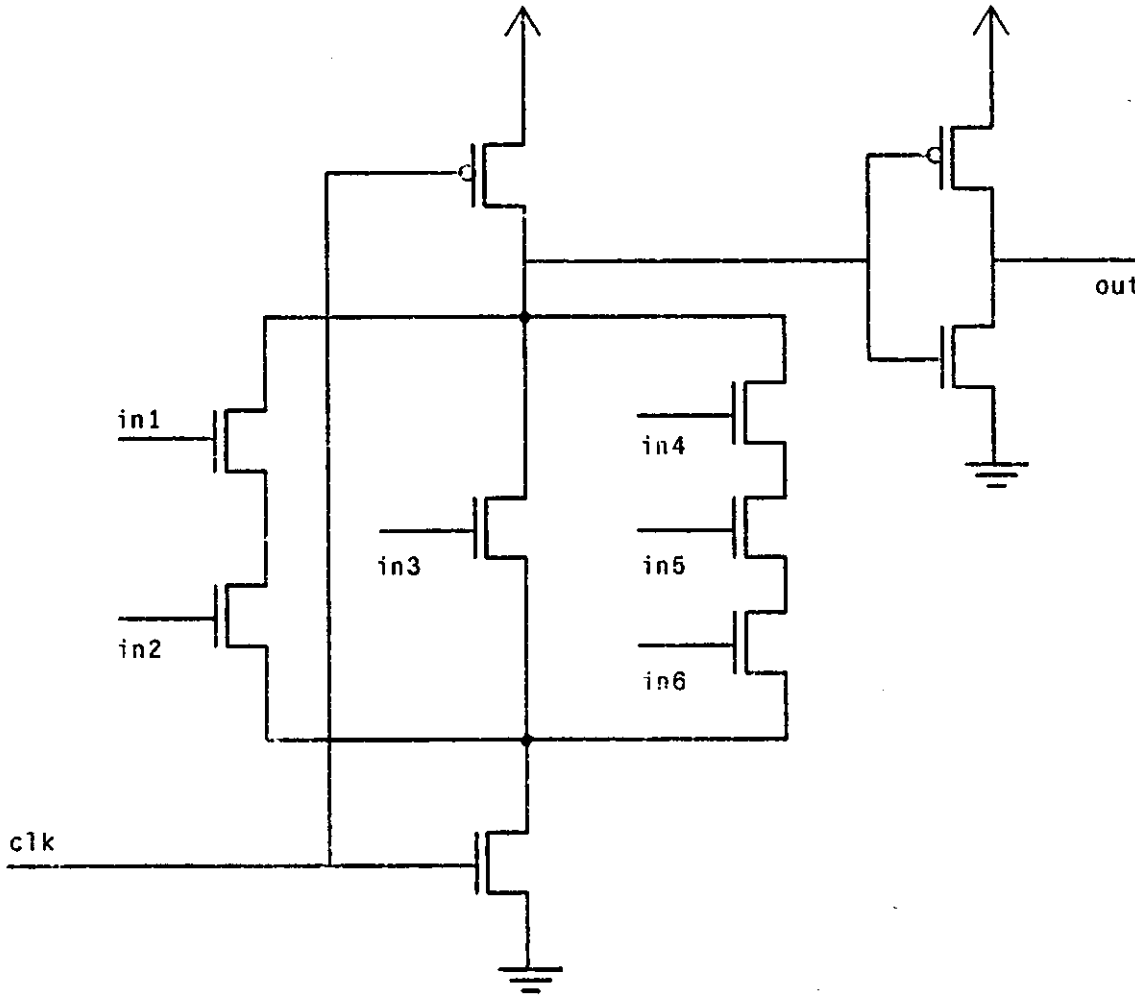


Figure 2-3: Domino logic

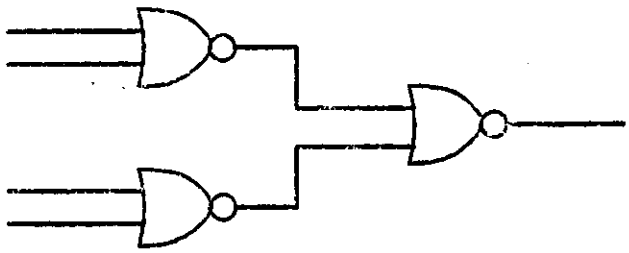


Figure 2-4: A structure difficult to implement with domino logic

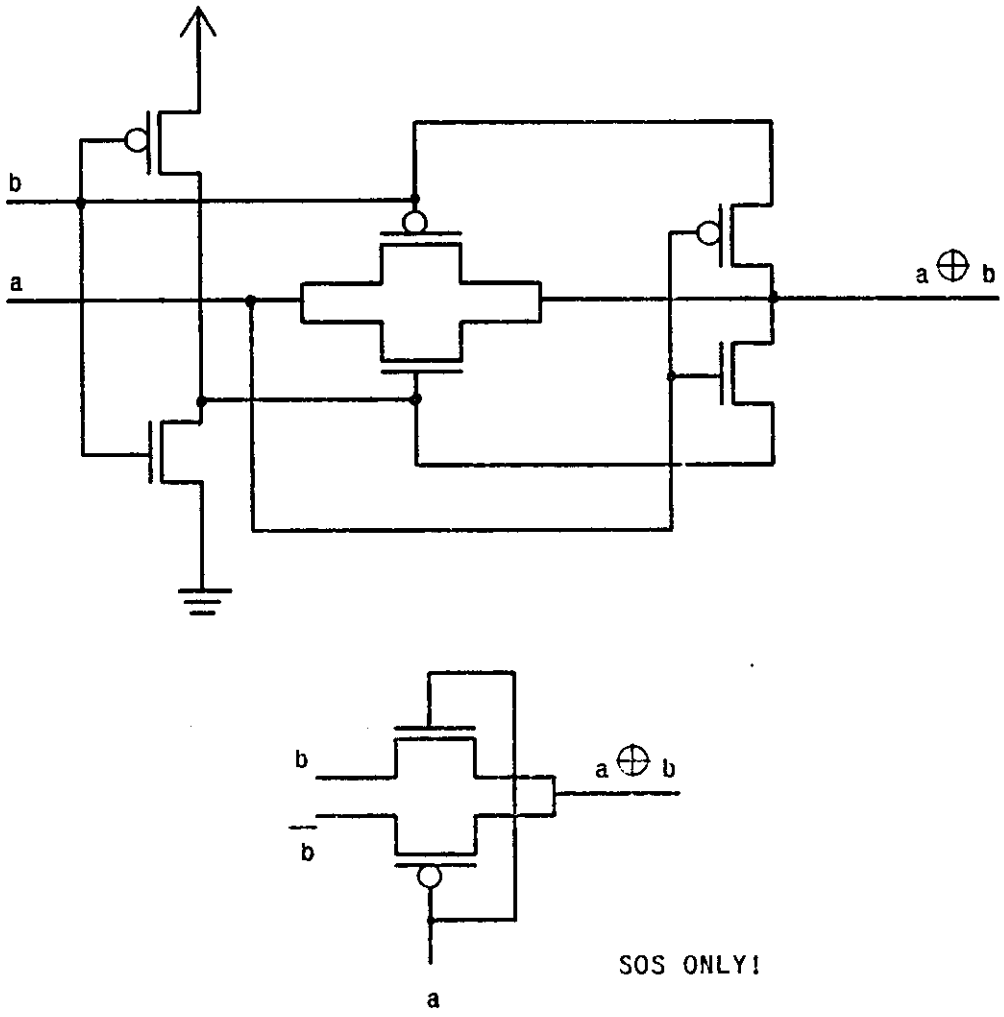


Figure 2-5: A CMOS exclusive-OR

noticeable.

CMOS did not change the situation as far as multiplier design is concerned. Still, the best structure is a Booth multiplier. Both Dadda and Wallace multipliers suffer from irregular wiring and do not speed up significantly the execution, for practical word-lengths. Moreover, the price to pay in terms of area is high enough to suggest different solutions, *at the architectural level*, able to speed up the overall system, rather than focusing on a single unit without considering the timing relationships among *all* the different modules.

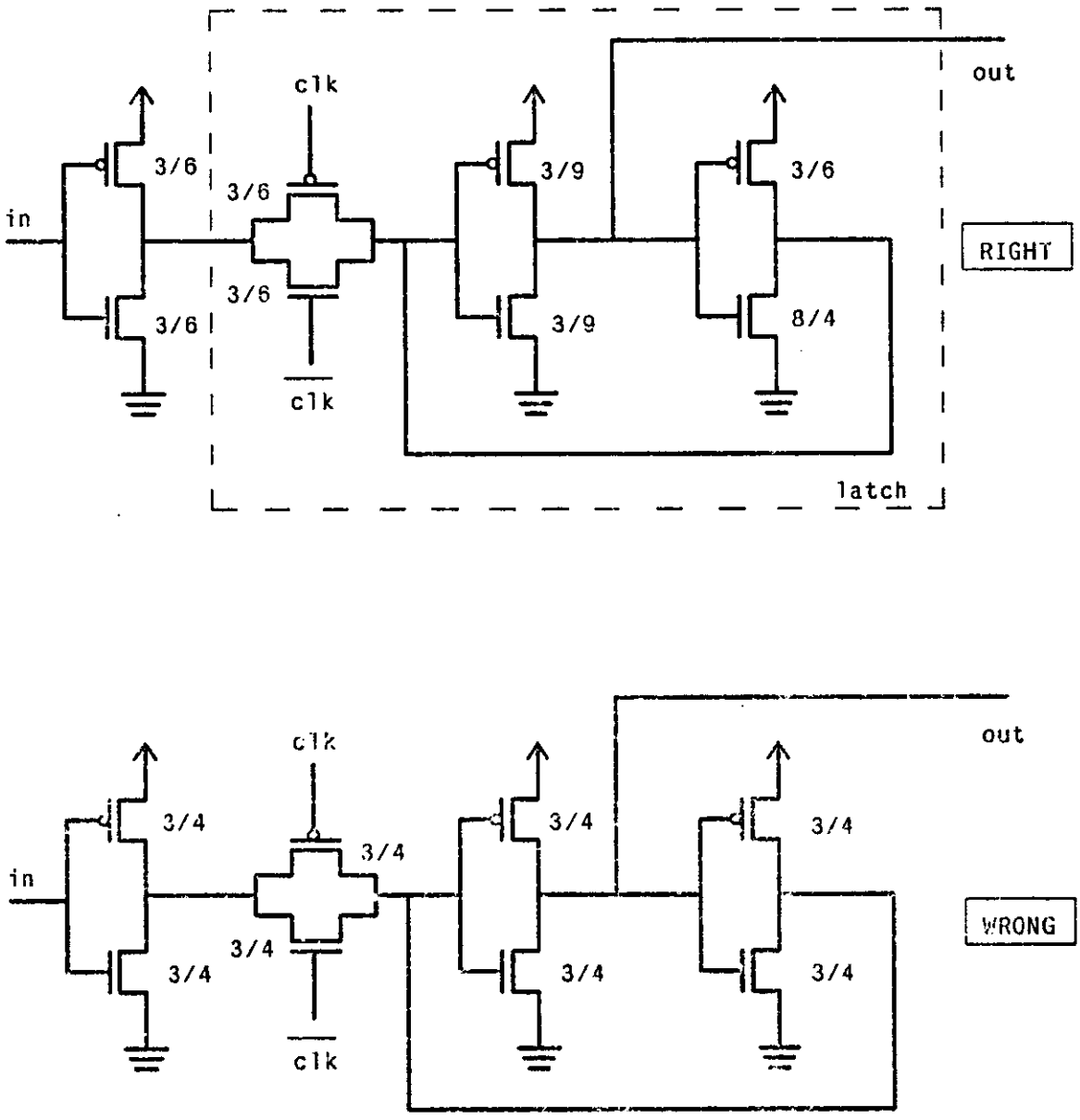


Figure 2-6: Latch design: right ratioes and wrong ratioes.

Finally, if the area is not a major concern, very large multipliers can be built with static CMOS gates. This was not possible in nMOS for power consumption problems.

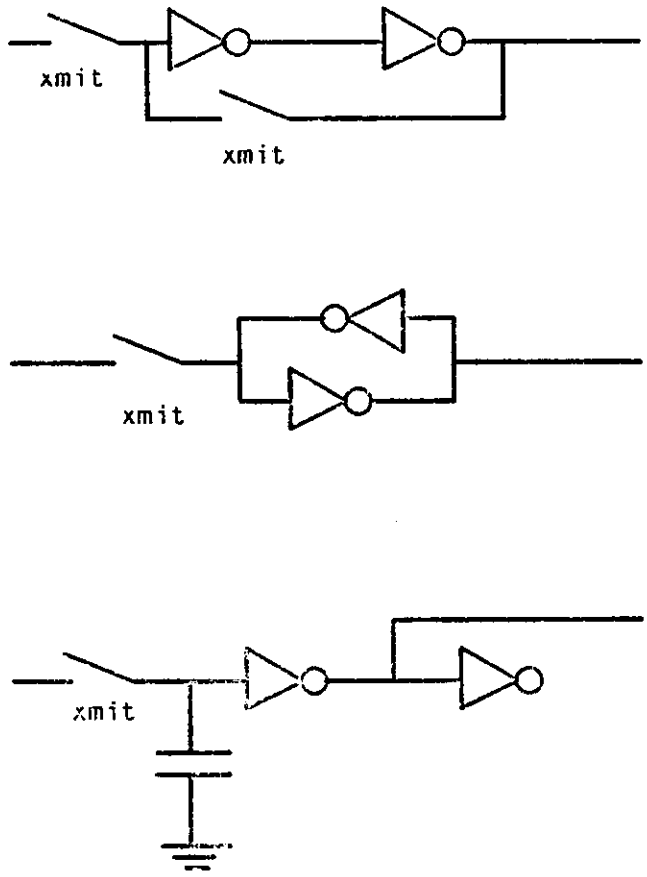


Figure 2-7: Latch design: some alternative topologies.

2.3. PLA's and RAM's

CMOS PLA's are significantly different from nMOS PLA's. While the AND-OR planes are usually implemented in nMOS by means of a NOR-NOR approach, this is not possible in CMOS for reasons of performance. The NAND structure is preferred.

Although a purely static PLA is feasible, dynamic PLA's are more suited to CMOS. The overall timing scheme of a dynamic PLA proceeds through these steps:

1. bottom-up clock transition: input data are latched;

2. clock high: evaluation;

3. up-down clock transition: output data are latched;

4. clock low: precharging.

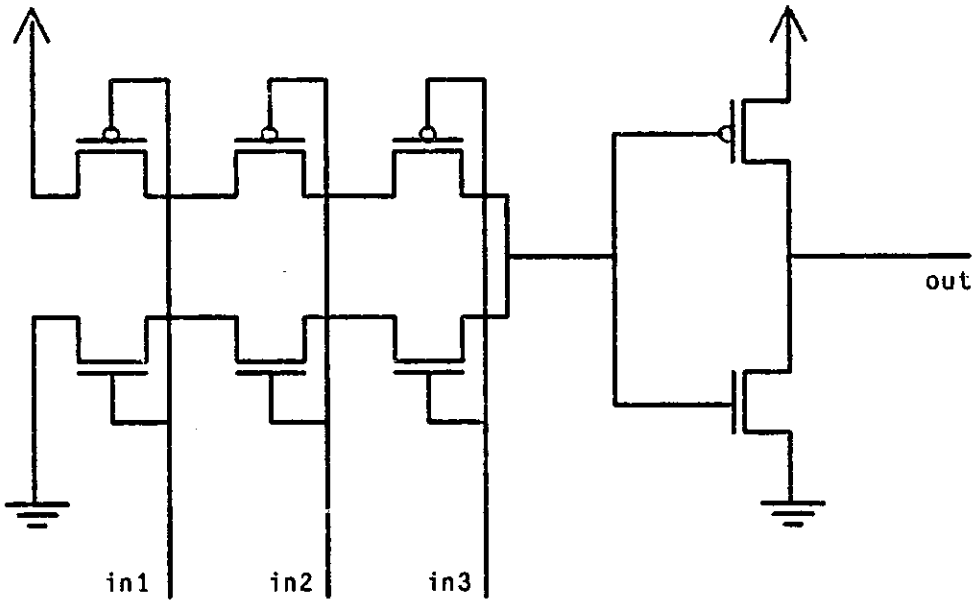
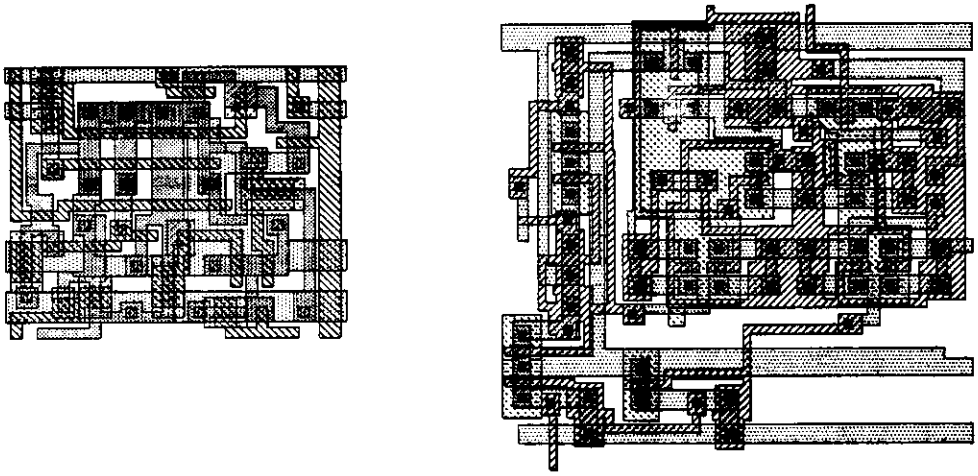


Figure 2-8: A "dynamic" muller element

Figure 2-9: nMOS full-adder (left) and CMOS full-adder

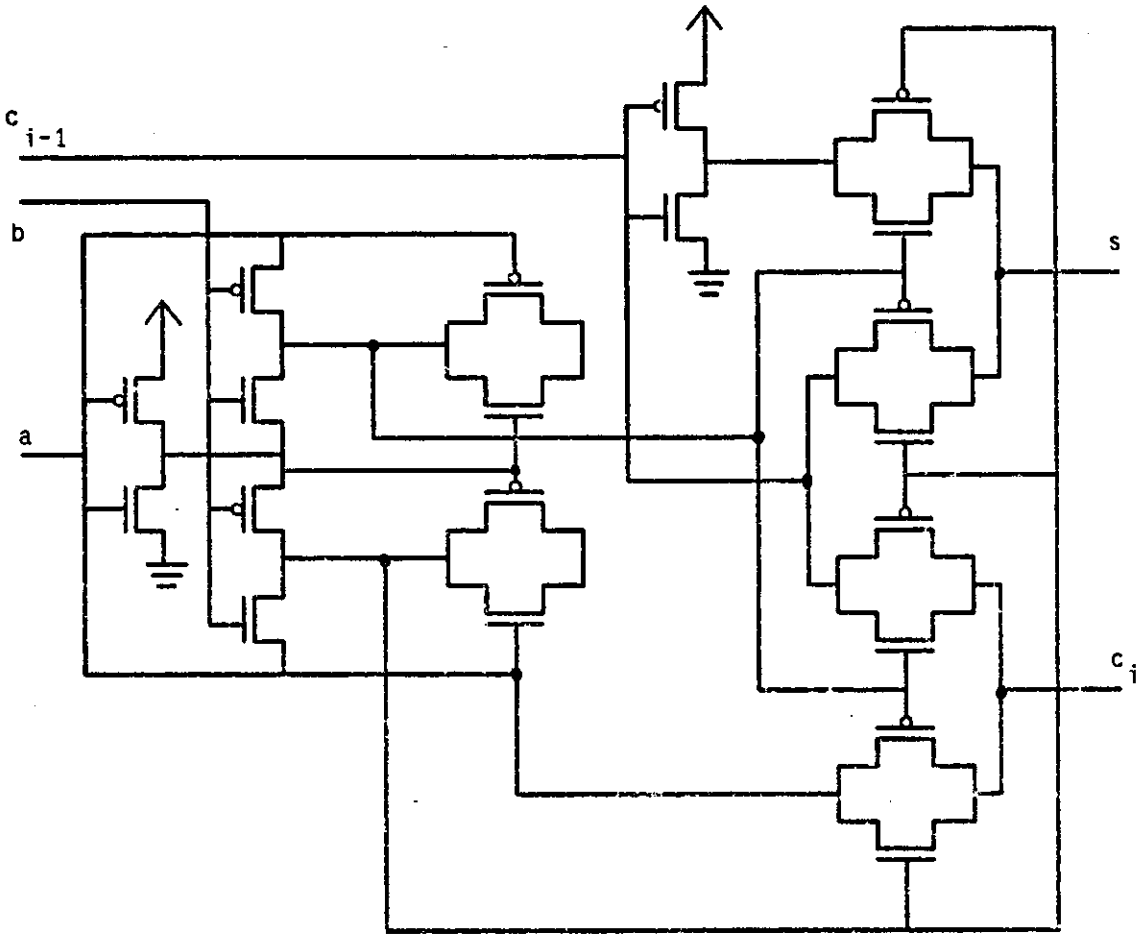


Much work has been done to define in a formal way CMOS PLA's. Interesting works are those of Glasser (MIT) and Seitz (Caltech). Some details on PLA's design can be found in [4].

CMOS RAM's feature:

- low standby power;

Figure 2-10: A full-adder scheme



- soft error immunity (alpha particles inducing errors are better handled by PMOS transistors);
- wide operational margin.

RAM's can be both static and dynamic (or pseudo-static). Clearly, static CMOS RAM's have the usual advantage over nMOS counterparts that power consumption is neglectable and therefore thermal problems are far less important. As usual in RAM design, the fabrication process plays a very important role: in CMOS, moreover, the only way to design state-of-the-art RAM's that are latchup-free (see section 3.1) is by means of sophisticated fabrication processes, because other solutions to the problem of latchup prevention cannot be applied in this case or would affect performance.

2.4. Watch out!

Although CMOS is more unconstrained than nMOS (ratioless the former, ratioed the latter), there can always be (more or less subtle) mistakes or even *wrong* topologies.

P-channel's do not conduct low voltages well; n-channel's do not conduct high voltages well (i.e. large voltage drop may occur). Therefore, the two configurations shown in Fig. 2-11 are *bugs*.

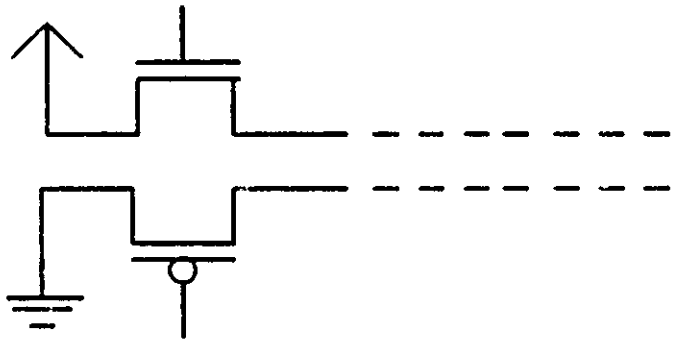


Figure 2-11: Avoid these configurations

Therefore, unilateral transmission gates can be used only when:

1. the input signal of the transmission gate is a logic zero: a unilateral, n-channel transistor, transmission gate must be used;

2. the input signal of the transmission gate is a logic one: a unilateral, p-channel transistor, transmission gate must be used;

3. CMOS-SOS is used: in this case it is less dangerous to transmit low voltages through a unilateral, p-channel, transmission gate (or, conversely, high voltages through a unilateral, n-channel, transmission gate).

If you are really interested in ultimate performance, an extensive use of buffered gates should be done. This because the input impedance in a CMOS gate depends on the input configuration and, therefore, very different input loads can be found. Input-output decoupling is therefore mandatory. A buffered gate is simply a gate followed by *two* inverters. Buffering allows to achieve a higher noise margin and faster speed.

It is evident how this methodology is area-consuming; therefore, "ultimate performance" should really mean *ultimate* performance. The use of buffered gates is therefore requested if and only if some nanoseconds make a difference, which will not be the case in ninety-five percent of our applications. A compromise is to put *some* decoupling logic, without buffering *all* the gates. A good design practice is,

anyway, to limit the number of inputs in the static gates. If this is either impossible or very difficult to achieve, dynamic logic design should be taken into serious consideration.

Latchup can be fired more easily by bootstrapped gates; if this technique is used, special care must be taken in the choice of the devices and in the layout.

2.5. Analog circuits

MOSIS is presently offering a process, called CBPE2, that features, for the first time, the availability of an integrated capacitor. This feature should not influence digital circuit design in a great extent (unless someone is interested in the implementation of either some kind of EPROM or non-standard storage technology).

On the other hand, this feature could be extremely useful in analog circuit design if it were accompanied by proper electrical parameters, which, unfortunately, is not the case. CBPE2 has a maximum voltage supply which is too low and threshold voltages which are, presumably, too high.

Nevertheless some experimentation can be made even at very low supply voltages. For sake of completeness, a brief overview of CMOS analog design has been included in this document, however. It is impossible to cover all the possible topics and only three of them will be dealt with:

- switched-capacitor technique;
- operational amplifier design;
- A/D conversion.

2.6. Switched-capacitor filter design

Accurate filters have always been a big headache. Accuracy mainly depends on the precision of the passive components (given an operational amplifier with high input impedance and low output impedance) that build up the filter. Resistances with .001% accuracy and neglectable thermal drift are available. Costs are astronomical, however.

As far as capacitors are concerned, things are much worse: 0.1% accuracy is a very good performance and, even using NPO capacitors, thermal drift is *not* neglectable and costs are even higher than resistances'.

The switched-capacitor technique is a clever solution to the problem of cost vs. accuracy. Capacitors are formed inside the chip and resistances are simulated *by switching-capacitors*. Finally, the

overall accuracy depends on the relative ratio among the various components and not on their absolute value, i.e. it is the geometrical accuracy of the fabrication process.

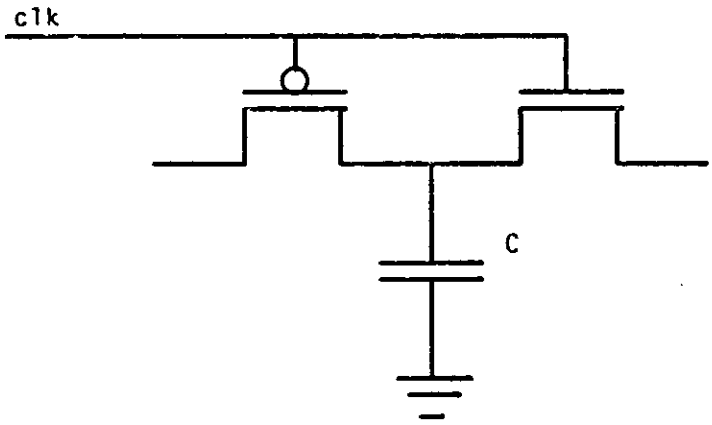


Figure 2-12: Switched-capacitor technique

In Fig. 2-12 the switched-capacitor technique is illustrated. The simulated resistance can be computed by the following formula:

$$R = 1/Cf_s,$$

where f_s is the switching frequency of the capacitor. Note that, the higher the resistance is, the smaller the capacitor (i.e. its area) should be. In Fig. 2-13 an integrator is shown.

The switched-capacitor technique allows to build any kind of filter and, obviously, it is especially powerful in those applications that require high accuracy; a typical example can be a high-order elliptic filter; tuning a high-order elliptic filter can be considered one of the most frustrating experience; this is not true any more with this technique.

Finally, it is interesting to notice that, as Butterworth, Bessel and Chebychev filters share the same topology, using switched-capacitors we can now *on the fly* change the type of filter. This was possible also before, but with additional circuitry and higher instability (e.g. using analog switches).

The literature on this topic is extensive. The reader is referred to [3], [1] for a survey on this topic. It is possible to find many applications in the IEEE Journal of Solid-state Circuit.

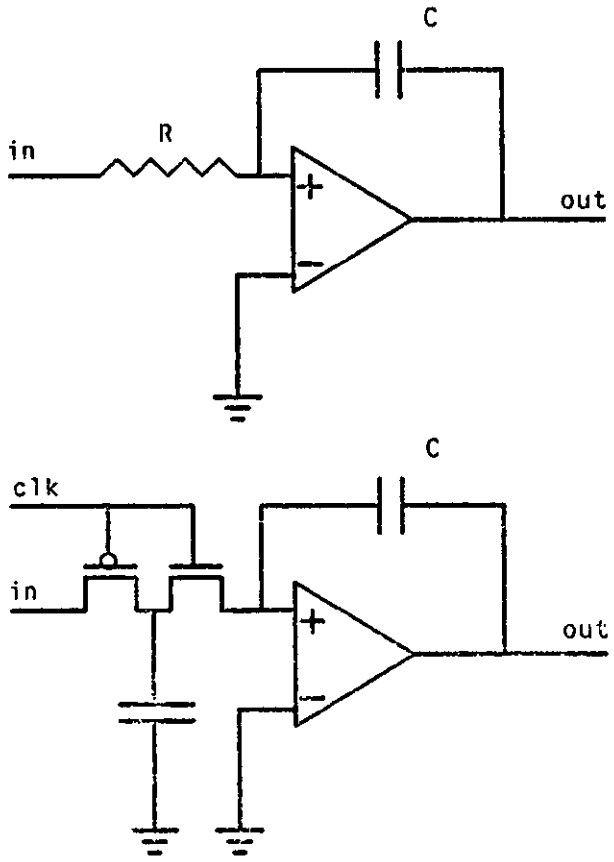


Figure 2-13: An integrator: scheme (up) and its implementation

2.7. Operational amplifier design

CMOS operational amplifier design very often exploits the same topologies used in bipolar operational amplifier design. A comparison between MOS and BJT transistor leads to the following results:

- It is difficult to achieve high voltage gains from CMOS op amp's (even though it is questionable, in some applications, whether a very high voltage gain is desirable[5]);
- the dc offset in a bipolar op amp (with differential input stage) is basically influenced by the

[5]Although a side-issue, let us spend some words on this topic. For example, a very high open-loop voltage gain is not desirable, unless it is not accompanied by a very wide open-loop frequency response, if a goal is to minimize all the possible causes of distorsion. TID (Transient Intermodulation Distorsion) and THD (Transient Harmonic Distorsion), for instance, are heavily dependent on the open-loop voltage gain. Most of the operational amplifiers that feature incredibly wide closed-loop frequency response, achieve this goal with extremely high open-loop voltage gains and poor open-loop frequency response and, therefore, very often suffer from the kind of second-order dynamic distorsions mentioned above.

topology of the first stage; in MOS, the first *and* the second stage contribute to the dc offset, because of the low gain achievable from MOS transistors;

- current driving capability is poor in MOS op amp's and off-chip load can heavily influence the overall performance;
- an MOS operational amplifier occupies less area than its bipolar counterpart and can more easily coexist with digital circuitry.

As far as noise is concerned, it depends on many parameters (e.g layout, fabrication process, topology): among them, the minimum feature size plays an important role; it is difficult to forecast MOS op amp's designed with a minimum feature size smaller than 4μ.

The basic topology for a CMOS op amp input stage is still the differential amplifier with current mirror, which, unless additional circuitry is used, requires both +Vdd and -Vee power supply. Two typical configurations for a CMOS differential amplifier are shown in Fig. 2-14: on the left a dual-input, balanced output is shown; on the right a dual-input, unbalanced output is shown. If large voltage swing is required and common mode range is not a major constraint, a differential/cascode configuration can be used (as it it is done on bipolar op amp's: see Fig. 2-15). The differential/single-ended conversion is usually performed by straightforward topologies (e.g. common source). Dynamic biasing is sometime used [7]: this design methodology was and is already used in the design of bipolar power amplifiers.

In order to drive off-chip loads, output buffers are needed. Class AB is the usual choice; if power dissipation does not represent a problem, class A can be considered. Finally, if crossover distorsion does not represent a problem, a class B amplifier can be designed. The basic problem, however, is the poor voltage swing achievable, even using complementary push-pull configuration. This is especially true if the fabrication process was oriented to digital application (i.e. devices with significant voltage threshold and operating at low voltage).

As a conclusion, a comprehensive survey on MOS operational amplifier design can be found in [5]; an interesting CMOS operational amplifier is presented in [17].

2.8. A/D conversion

CMOS can play an important role in analog-to-digital conversion. Presently, apart from extremely sophisticated applications in which discrete devices are still needed (sampling frequency ranging in the order of hundreds of MHz), single-chip A/D converters are widely used.

The most used technique, for medium to high speed applications, works for "successive approximations": the overall scheme is basically a digital-to-analog converter inside a feedback loop.

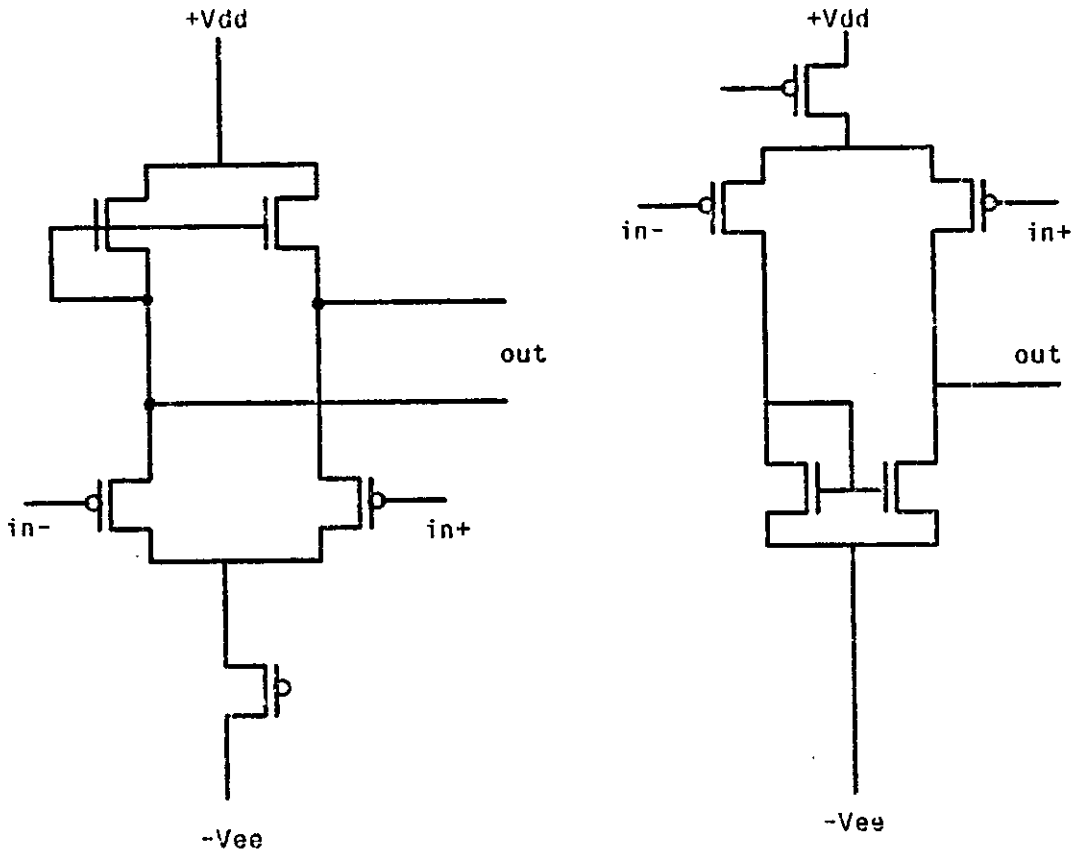


Figure 2-14: Differential amplifier with current-mirror: balanced (left) and unbalanced output

The conversion time takes n-cycles where n is the word-length in bit.

The ICL 7115 analog-to-digital converter, from Intersil, is a CMOS, successive-approximation, analog-to-digital converter with interesting features:

- conversion time: 50μsec.;
- resolution: 14 bit;
- on-chip PROM to achieve 14-bit linearity without laser-trimmered resistors.

"Flash" conversion is among the fastest techniques presently available. Its only drawback is that a flash converter needs as many operational amplifiers inside the chip as the number of quantization levels. Therefore, if we need an 8-bit A/D converter, we have to put 255 operational amplifiers inside the chip. While, until some year ago, the maximum word-length was 6 bit, 8-bit flash converters are presently available.

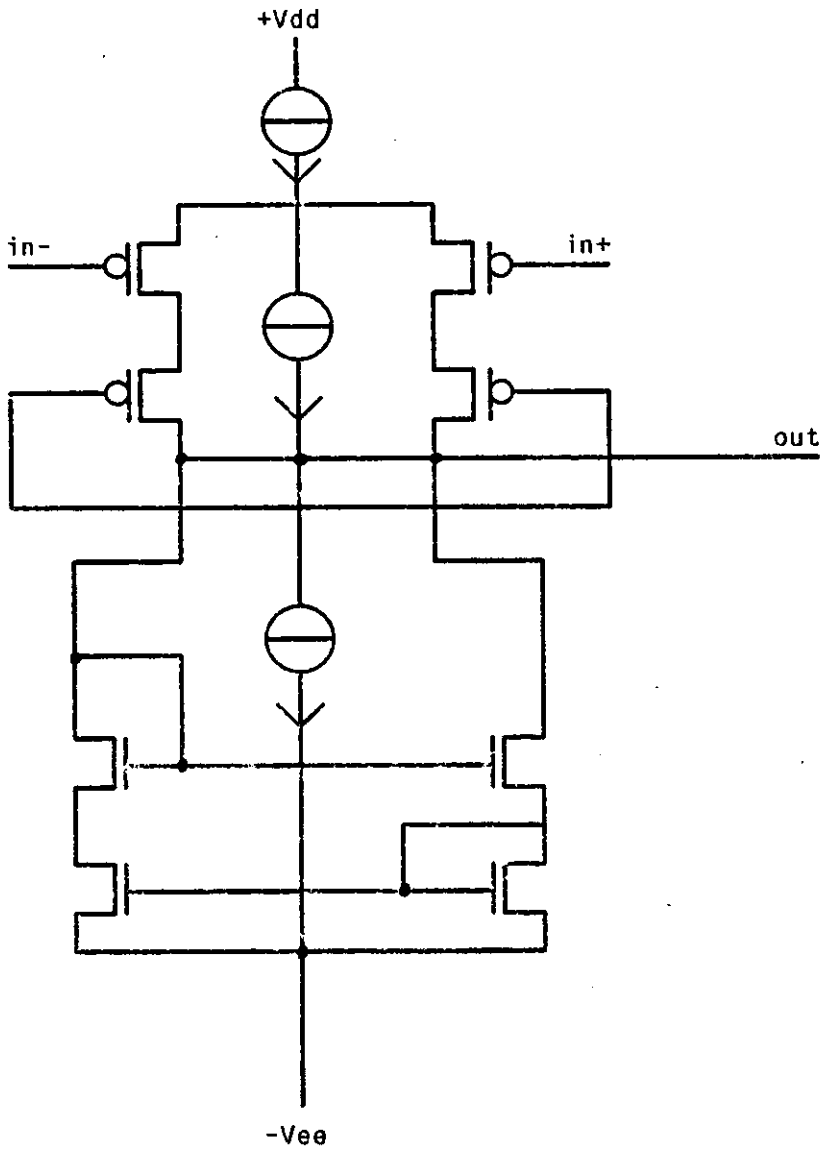


Figure 2-15: Cascode - differential input stage

Although CMOS cannot be considered a fast technology (but with dimensions scaling down this cannot be any more necessarily true), a CMOS flash converter is feasible because a CMOS operational amplifier occupies a small area and does not need area-consuming output buffers. The output of these operational amplifiers remains in fact confined inside the chip. An ultra-fast CMOS analog-to-digital converter is presented in [2].

A block scheme for a flash A/D converter is shown in Fig. 2-16. One problem in flash analog-to-

digital converters is that the input signal goes to all the operational amplifiers, i.e. for an 8-bit A/D we have an input capacitance which is 255 times the input capacitance of the single operational amplifier; it is therefore very important to use a technology that features low junction capacitance.

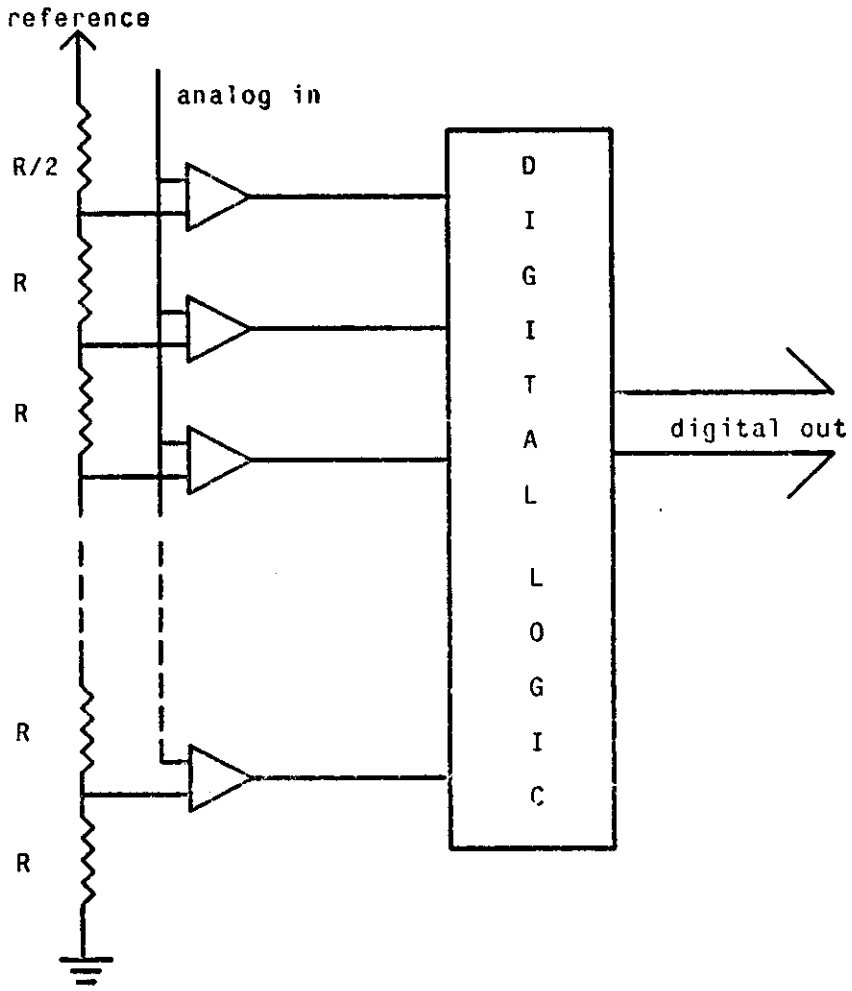


Figure 2-16: Block scheme of a flash A/D converter

3. CMOS-Bulk

One of the biggest issue in CMOS is how to isolate the two devices. This can be accomplished in different ways, either by using a proper isolating substrate (as in CMOS-SOS) or via a more complex fabrication process. In CMOS-Bulk this isolation is achieved by forming the n-channel (p-channel) transistor in a *P-well* (*N-well*). Therefore we can have either CMOS-Bulk P-well or CMOS-Bulk N-well. A third process, called twin-tub, provides both the n-channel and the p-channel with an isolating structure ("tub" and "well" are synonyms).

The well is the actual substrate (back-gate) on which one of the two transistor will be formed. Therefore, in CMOS P-well, we shall form the p-channel's on an n-type substrate and the n-channel's on a p-type substrate (the P-well). The well is surrounded by the n-type substrate.

Between P-well and N-well the winner is N-well. The reason why N-well is "better" than P-well is basically the fact that N-well is nMOS compatible, i.e. it allows the fabrication of nMOS/CMOS chips. As far as speed, it is more difficult to determine a clear superiority of one technology over the other; two different considerations can be found in the literature:

- N-well *could* be faster but the well influences so heavily the speed that there is no significant difference between P-well and N-well (see, for instance, [19] pp. 482-483);
- the well does not play such an important role, therefore N-well is faster (see, for instance, [10] pp.57-58).

As far as we are concerned, P-well and N-well are interchangeable; therefore, we shall use P-well to explain the fabrication process, as P-well is the technology presently offered by MOSIS.

The fabrication process is shown in Fig. 3-1. The fabrication process proceeds through the following steps:

1. the P-well is patterned;
2. the active area inside and outside P-well is established;
3. polysilicon is patterned;
4. the two implant masks are placed: the N+ mask is simply the *negative* (in the photographic sense) of the P+ mask[6];
5. contacts are placed;

[6]This is the way in which the process supported by MOSIS is carried on; obviously, other processes could need explicitely an N+ mask.

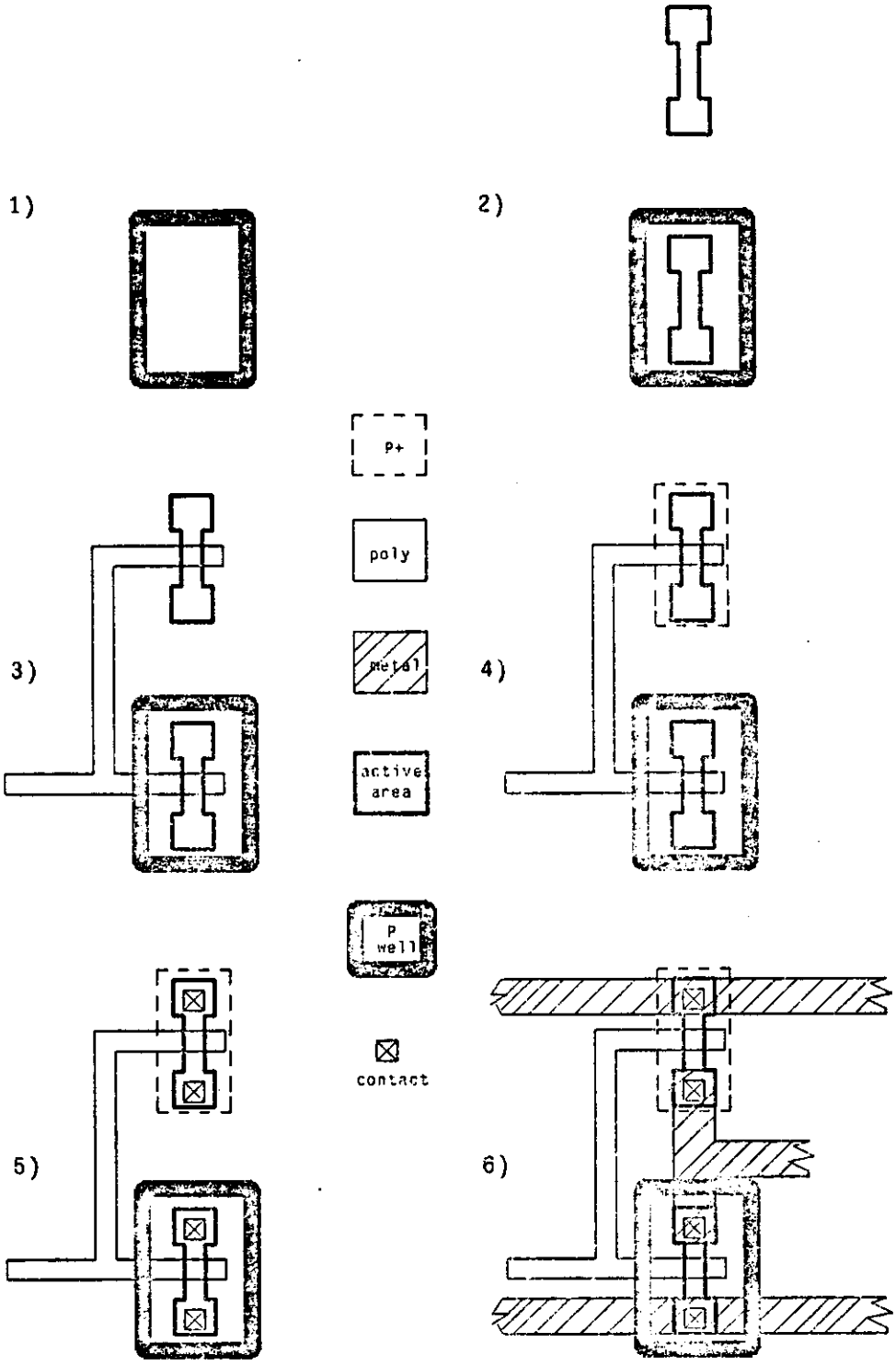


Figure 3-1: CMOS-Bulk P-well fabrication process

6. metal is placed.

The reader is referred, for more information on the fabrication process, to [19] pp.482-485 and, more generally, to all the literature dealing with semiconductor technology.

As far as the MOSIS fabrication process, it is important to remind that:

1. the polysilicon sheet resistivity is higher than in the comparable nMOS process and this contributes to make the design of CMOS chips harder than the design of nMOS chips;

2. the resistance of the contact is higher than in the usual 4μ nMOS process: although it does not yet represent a major problem (as far as 2μ it cannot definitely influence the overall design), it would be better to carefully study the routing scheme in order to minimize the number of contacts.

3.1. What is latchup and how to avoid it

Latchup can cause the complete destruction of the chip. Many researchers have tried to define in a formal way under which conditions latchup is fired; the task is extremely complex and the problem is still open. What makes a formalization of the phenomenon very difficult is, for instance, latchup strong dependency on the layout.

Being so dangerous, very much care has to be taken in order to avoid the occurrence of this phenomenon. Basically, latchup produces a "short circuit" between Vdd and GND. Latchup is fired when the output of the gate falls below GND or goes above Vdd (for noise spikes, electrostatic discharge and so on).

During the fabrication process, parasitic bipolar transistors are formed in the substrate (both P-well and n-substrate: see Fig. 3-2); the bases of these transistors are, respectively, the P-well (npn) and the n-type substrate (pnp). The parasitic transistors are connected in such a way that an SCR is generated (see Fig. 3-3). If the output of the gate goes below GND for an amount comparable to the threshold of the device, the emitter of the npn starts to inject current into the base (P-well); the electrons, from the collector, migrate to the Vdd node; if between Vdd and the source (p+) of the pull-up there is enough resistance, a voltage drop occurs and the potential of the n-substrate is lowered of the same amount the P-well potential was lowered; holes will start to migrate from the emitter and, through the pnp, they will reach the P-well and, if there is enough resistance between GND and the n+ source, a voltage drop will occur; therefore, the n+ source will start to inject electrons in the P-well. As evident, a positive feedback has been created. The only way to stop this destructive process is to disconnect either Vdd or GND.

For a more comprehensive survey on latchup, the reader is referred to [19] pp.481-482 (the very

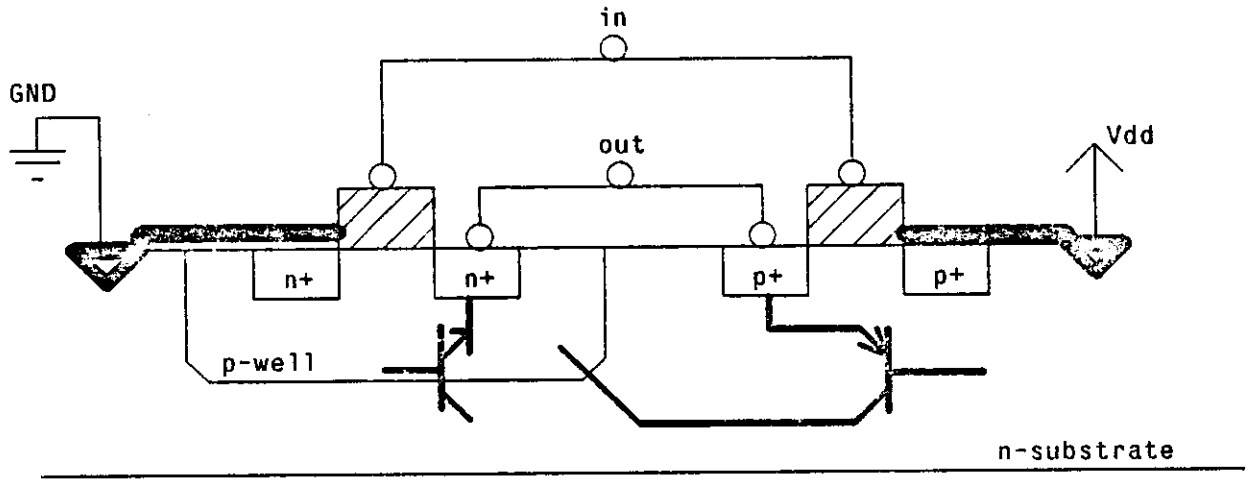
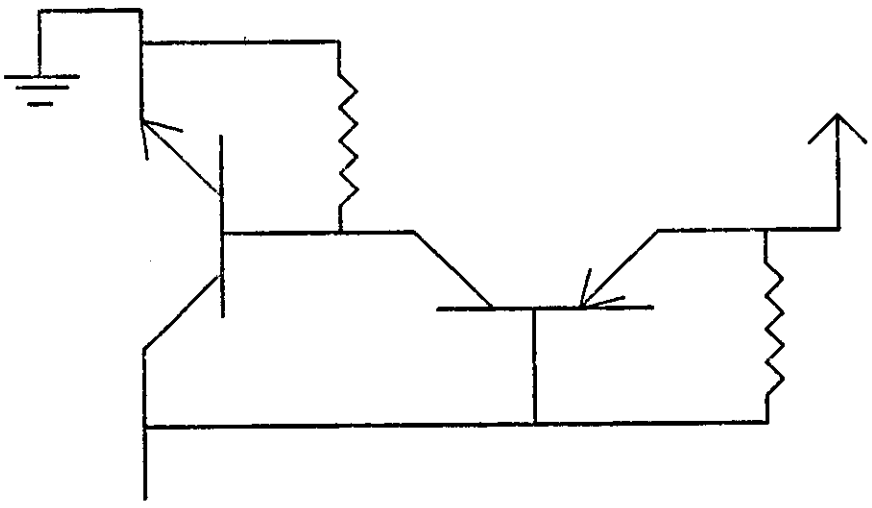


Figure 3-2: Parasitic bipolar transistors causing latchup

Figure 3-3: Parasitic SCR



first to be read), [21], [14] and [23].

How to avoid or limit the occurrence of latchup? One possibility is to drastically reduce the gain of the parasitic transistors (i.e. $h_{fe}(npn) \times h_{fe}(pnp) < 1$); this can be done through sophisticated doping procedures. Another solution, claimed to be "the" solution, aims at "destroying" the SCR; the approach is interesting and the strategy to pursue the goal is really peculiar [18].

A third solution is to use twin-tub, that significantly reduces the occurrence of latchup. However, as far as the chips fabricated by MOSIS are concerned, presently they are built with a technology (P-well) which is not intrinsically latchup-free.

In order to find some solution to this problem, let us summarize the basic information we have on latchup:

1. latchup takes place for an imperfect isolation between the n-channel's and the p-channel's (parasitic, *active* devices are present);

2. latchup takes place if the output signal of a gate goes below GND or above Vdd;

3. latchup is fired by a positive feedback; the amount of feedback is inversely proportional to the "resistance" between the n-channel and the p-channel (this formulation will disgust many people but is simple to understand and I am going to take the risk of using it).

Possible precautions to adopt are, therefore:

1. we can better isolate the n-channel from the p-channel: to this end, we can put a *"guard-ring"* around the P-well, in order to isolate the two devices. The guard-ring is simply a *grounded* P+ ring (MOSIS design rules deal with it) around the P-well. We can also put another guard-ring in the p-channel section, with n+ contacts to the substrate tied to Vdd.

2. the possibility that a signal goes below GND or above Vdd is greater in the I/O section (which is more sensitive to environmental influence): therefore a set of I/O pads which are latchup-free is a must. Moreover, during power up, the chip can be more easily affected by overvoltages that could in turn fire latchup; a good precaution is to put a capacitor (47nF ... 470nF) between *each* Vdd and GND pad.

3. we can have higher resistance between the n-channel and the p-channel simply keeping them far away: this seems to be an insane solution; nevertheless it is used and actually one big issue in CMOS-Bulk is: "how far?"

As far as the last question, it is obvious that the spacing depends on the feature size we are talking about; in some sense, MOSIS design rules take into account this problem and it does not seem useful to significantly increase the natural spacing that we can achieve by simply following the design rules.

Many papers deal with the problem of latchup prevention, from different points of view

(fabrication process, design rules, substrate insulation): see, for example, [15], [9] and [19] p. 490.

3.2. Design Rules

This section will not deal with JPL design rules; the new MOSIS design rules, referring to a 3μ feature size, are supposed to be "the" rules for the next future. A copy of MOSIS design rules is included at the end of the document.

Before considering these rules, it would be better to understand how the fabrication process works (in a *very* general sense: why we need a well, why P+ and N+ etc.); this can help in the future when some doubts will arise. Unlike nMOS, it is more difficult to build a CMOS chip with no knowledge of the process and its electrical features; in other words, the (nMOS) approach that sounds like: "green and red make a transistor ..."; "if I put yellow I make a depletion transistor ..." and so on, is very dangerous. A minimum level of knowledge about the meaning and the actual behavior of each layer is recommended. When a library of cells becomes available, people not deeply involved (and interested) in this kind of "electrical nightmares", will be able to build CMOS chips as though they were nMOS chips. Apart from that, some terminology is needed, anyway:

- *active area*: in this area a transistor (either n-channel or p-channel) will be formed; *caesar* displays it in green (according to the technology file and the colormap shown later in this report) and you can call it "diffusion";
- *P+*: if implanted in an active area, it generates two p-doped regions, i.e. the source and the drain of the p-channel. Its color is yellow;
- *N+*: it does not exist as an explicit layer. Even though specific design rules are given for the N+ mask, this mask is defined as the *negative* of the P+ mask, where "negative" is meant in the photographic sense. In other words, where there is no P+ mask, a N+ mask will be implanted automatically *during the fabrication process*. Saying that N+ does not exist means that you do not have to worry about it, the silicon foundry will take care of that. The reason why a set of design rules for N+ mask has been introduced is because this set of rules should be valid for either P-well *and* N-well (that might be available in the future). Another reason is to allow the designer to explicitly define an N+ layer with its own design rules : before submitting the chip, this layer must be removed from the cif file.

The most important design rules are graphically represented in Fig. 3-4, Fig. 3-5, Fig. 3-6, Fig. 3-7 and Fig. 3-8.

Some comments: the metal-poly contact has a different geometry from either the CMOS-JPL contact or the nMOS contact. Minimum metal width is smaller (scaling the process, obviously) than nMOS metal width. This is a very good feature, being CMOS metal-intensive (poly and metal are presently the only possible layers for interconnections).

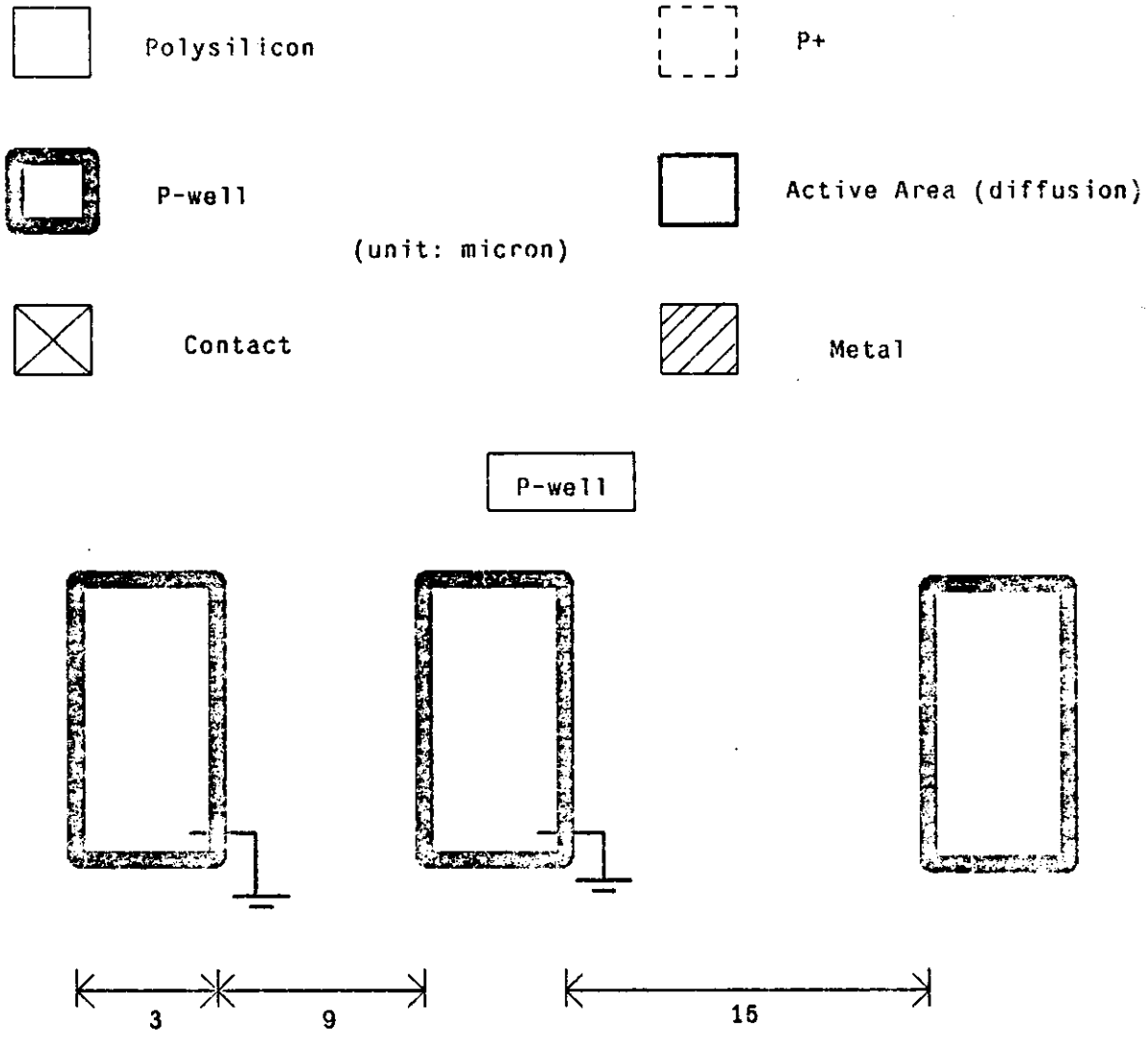


Figure 3-4: MOSIS CMOS-Bulk Design Rules (a)

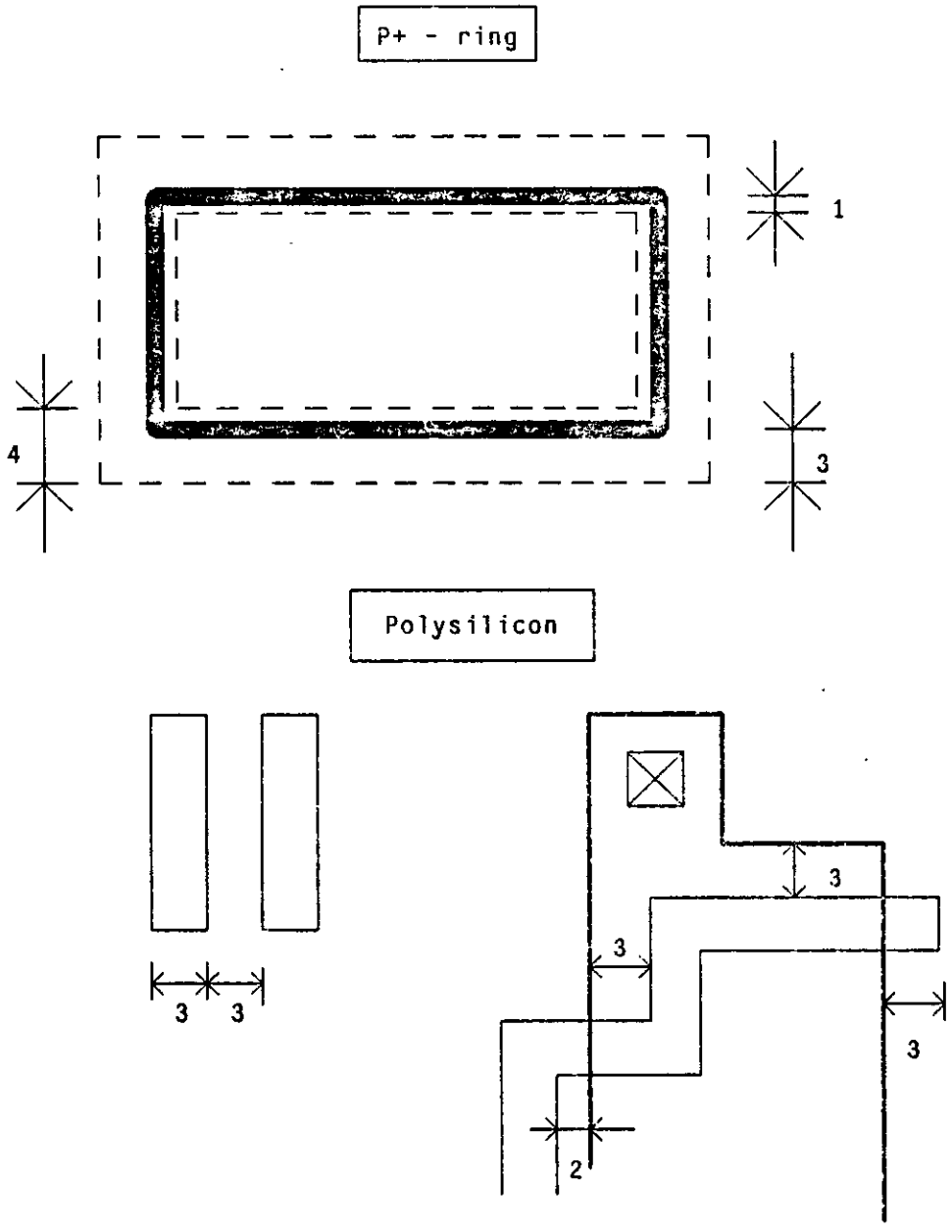


Figure 3-5: MOSIS CMOS-Bulk Design Rules (b)

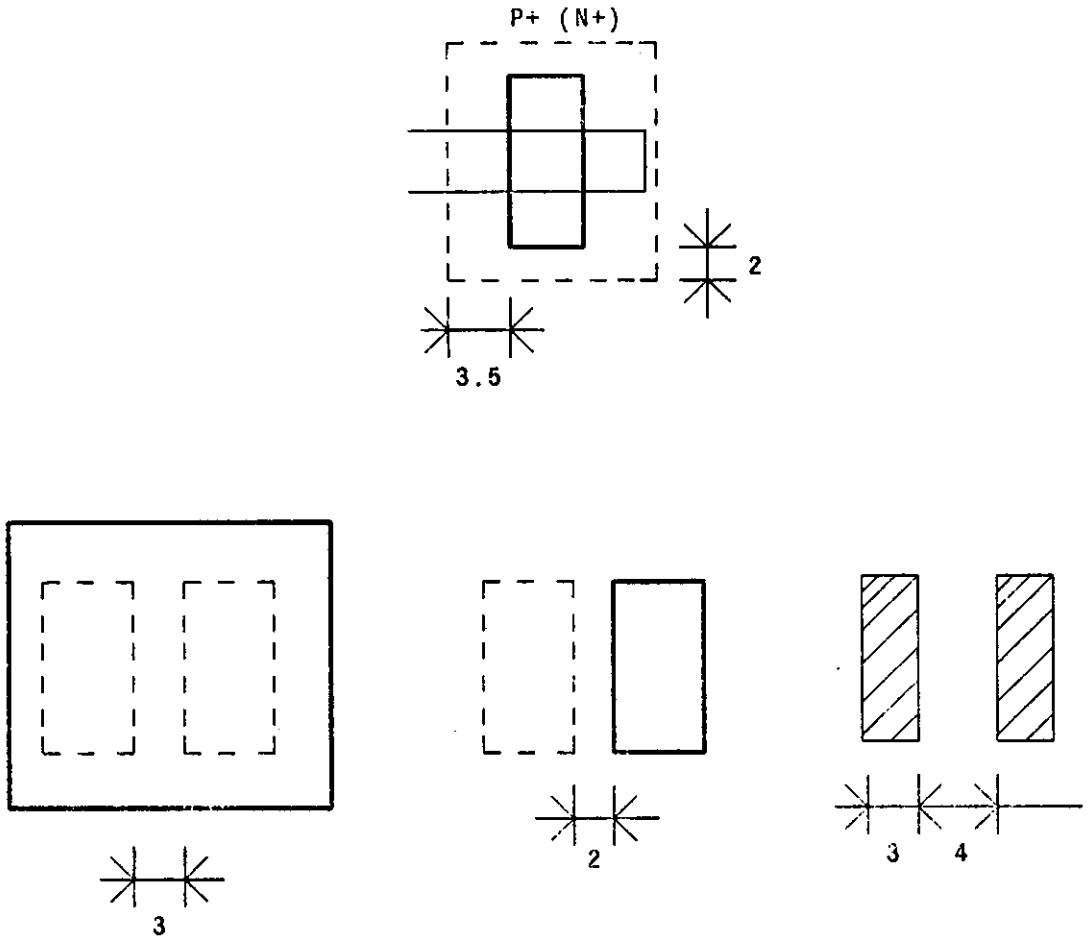


Figure 3-6: MOSIS CMOS-Bulk Design Rules (c)

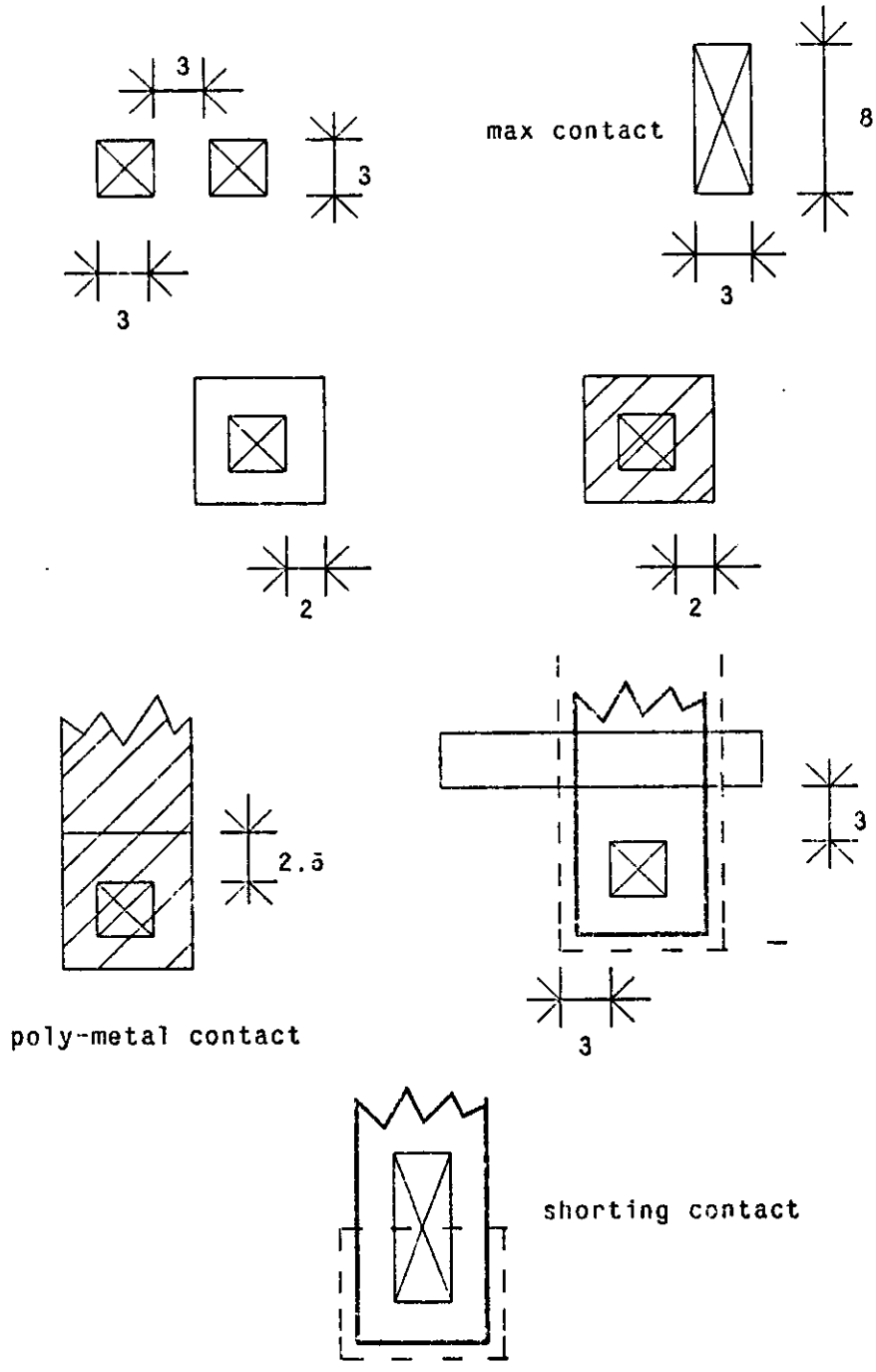


Figure 3-7: MOSIS CMOS-Bulk Design Rules (d)

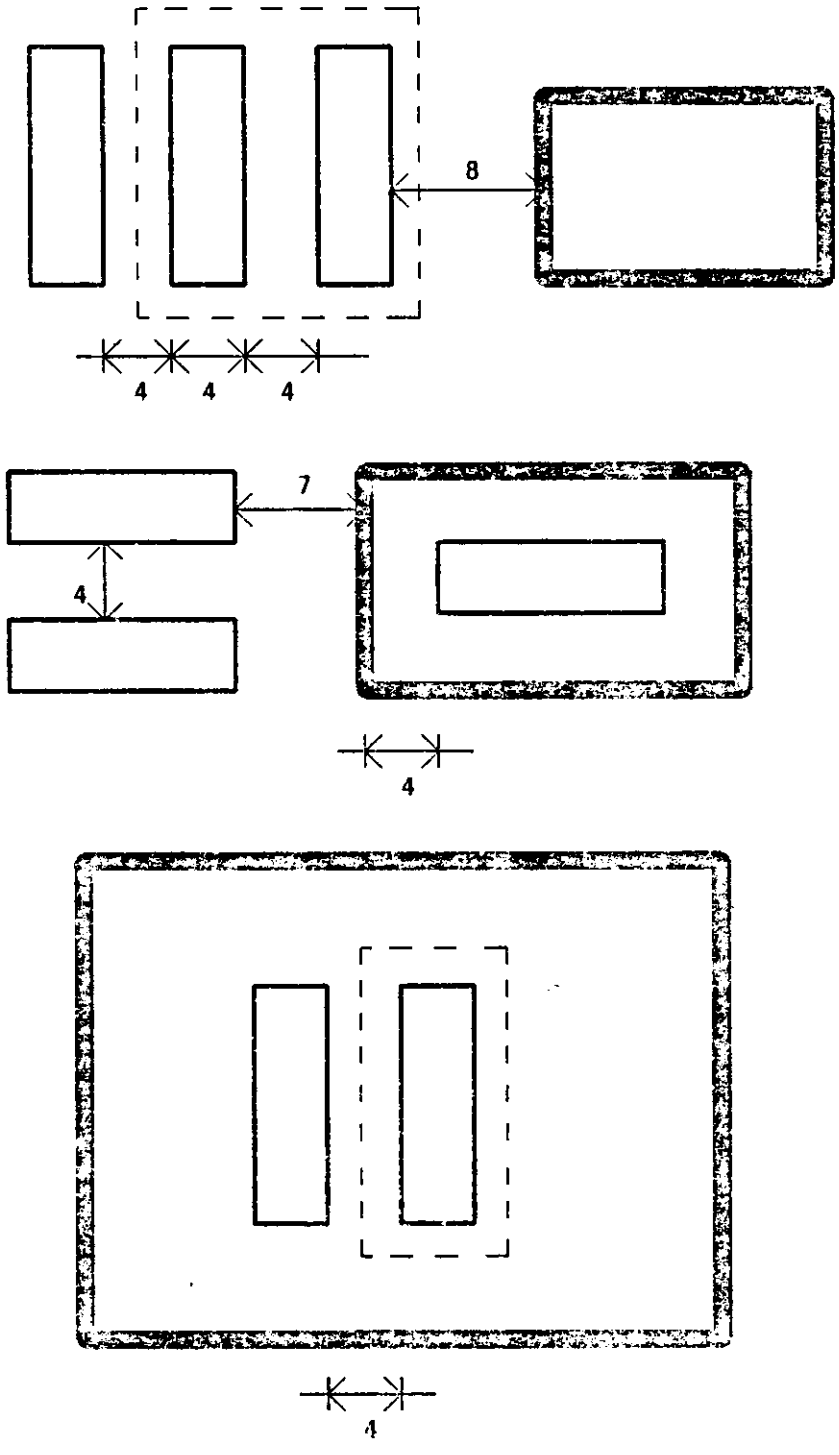


Figure 3-8: MOSIS CMOS-Bulk Design Rules (c)

Finally, looking at the minimum-dimension inverter and transmission gate (see, respectively, Fig. 3-9 and Fig. 3-10 and especially Fig. 1-4), it is evident how these units are larger than the corresponding nMOS units. Therefore, a decision was made that:

a caesar unit is 100 centimicrons

This allows us to better approximate the design rules and to reduce the dimensions of the gates. This choice has positively influenced, for instance, the absolute dimensions of the poly-metal contact, allowing the design of a 7 x 8 micron contact, instead of a 9 x 9 micron contact obtained with lambda-based design rules (i.e. $\lambda = 1.5\mu$). The fact that the design is no more structured "a la" Mead & Conway was not thought to be a drawback for the following reasons:

1. CMOS-Bulk chips designed in this environment in the next future will be either highly experimental or "aggressive", with no other target rather than to exploit the technology at its most;

2. it is hard to believe that a chip built in CMOS-Bulk can be, at least in the next future, "the first chip"; in other words, a designer interested in CMOS-Bulk is supposed to have already designed at least some nMOS chips and therefore will not be worried by using "unstructured" design rules.

3. as the process scales down, it is reasonable to assume that a very different set of design rules will be produced; differently from nMOS, that scaled down from 5μ, through 4μ, to 3μ keeping the relative ratio among dimensions almost unchanged, a chip built with MOSIS 3μ design rules will hardly satisfy , let's say, MOSIS 2μ design rules. This also because, at that time, CMOS-Bulk can be replaced by more powerful technologies (e.g. CMOS-SOI).

Finally, if you do not like this choice or you do not agree, you can still use a $\lambda = 1.5\mu$ (or even $\lambda = 2\mu$) and live in a structured environment. As you will see later on, the design rule checker ought to be modified, in this case.

3.3. Second metal layer and capacitor

Although a second metal layer is not presently fully supported by MOSIS, the design rules have been released. In Fig. 3-11 they are shown. On the contrary, the capacitor (or "second poly") is supported and its design rules are shown in Fig. 3-12.

Needless to say, a second metal layer would help us to solve the problem of interconnections that, having only two possible layers presently, is particularly annoying.

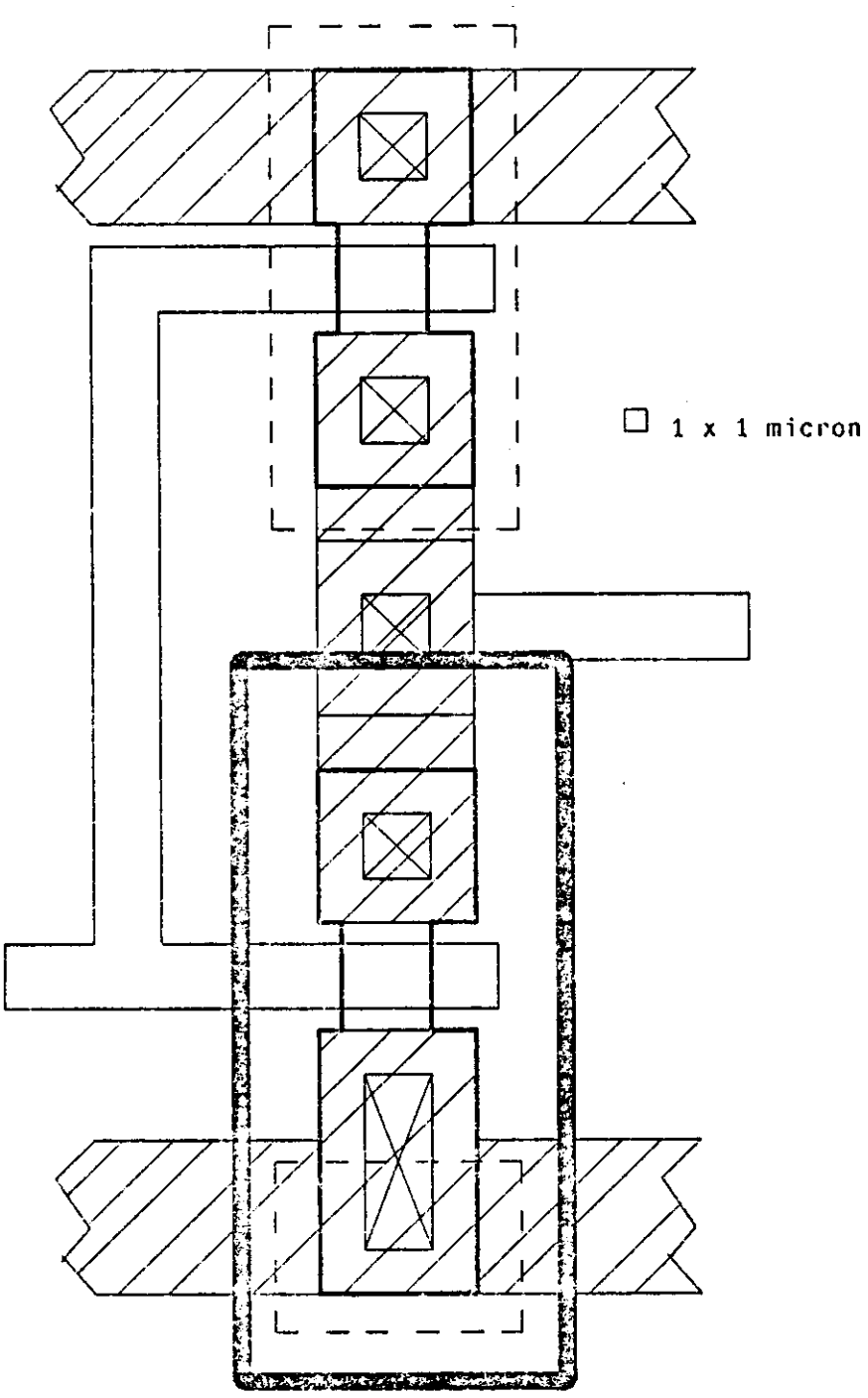


Figure 3-9: CMOS-Bulk, MOSIS design rules: minimum dimension inverter

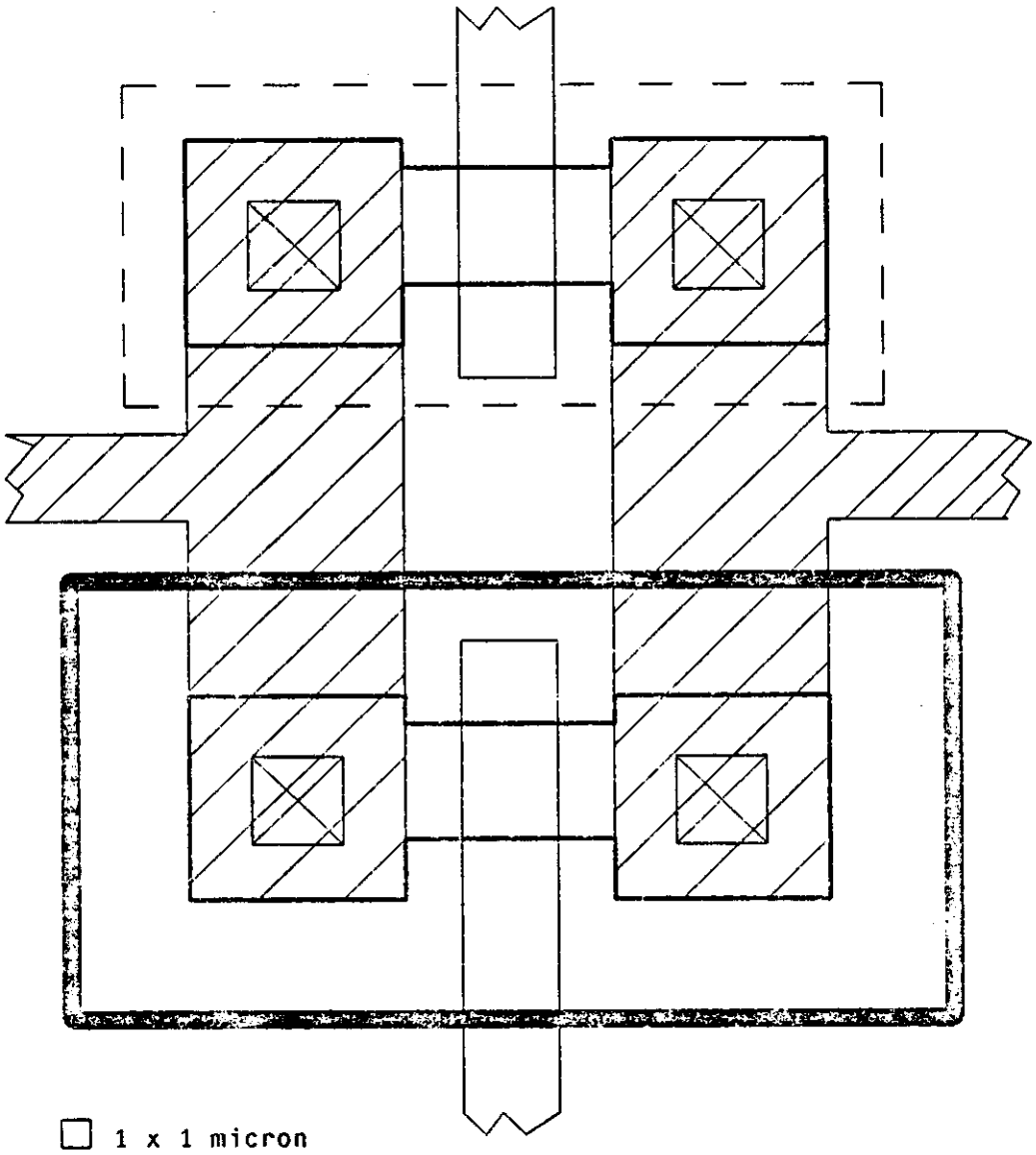


Figure 3-10: CMOS-Bulk, MOSIS design rules: minimum dimension transmission gate

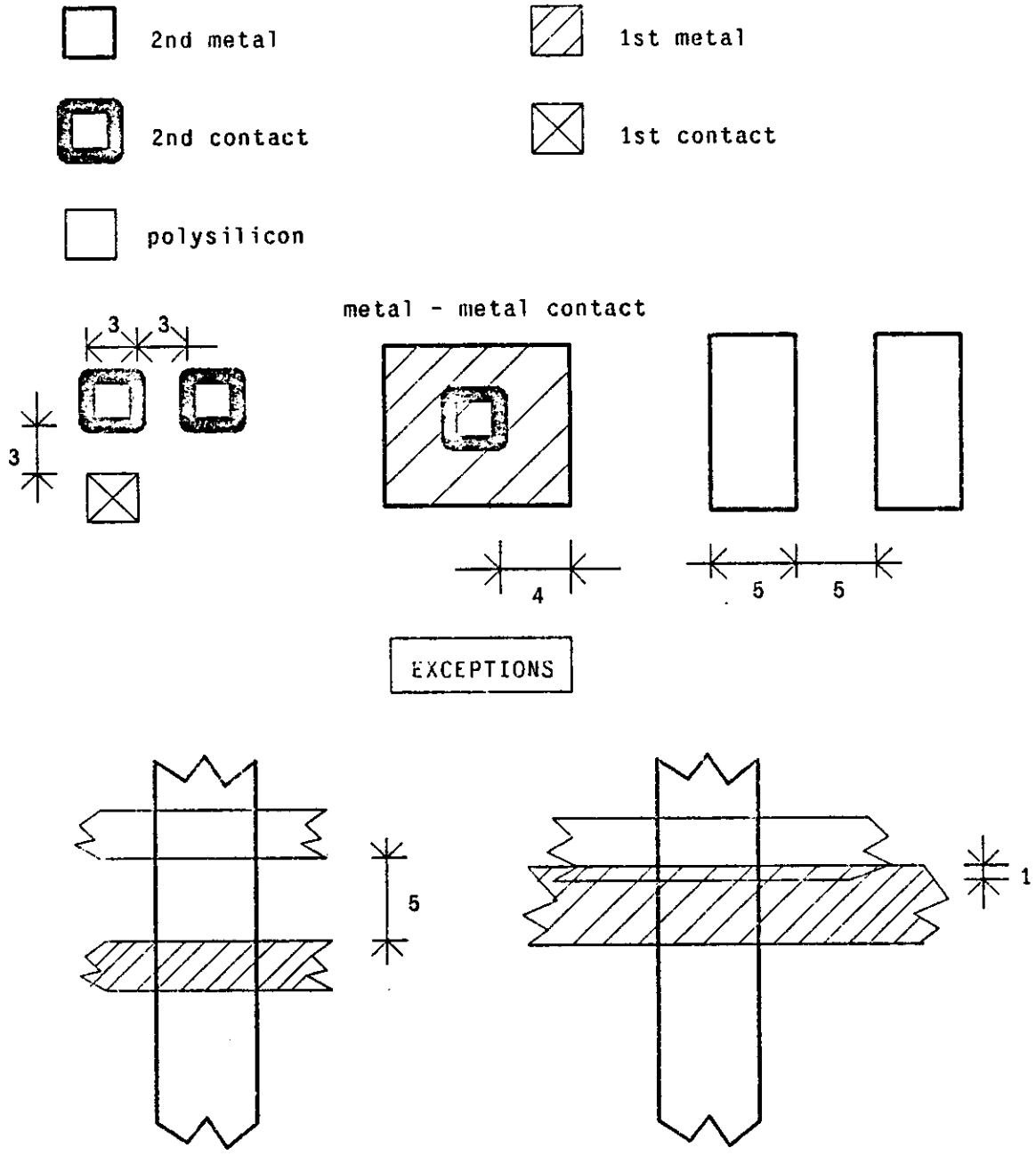


Figure 3-11: Design rules for a second metal layer

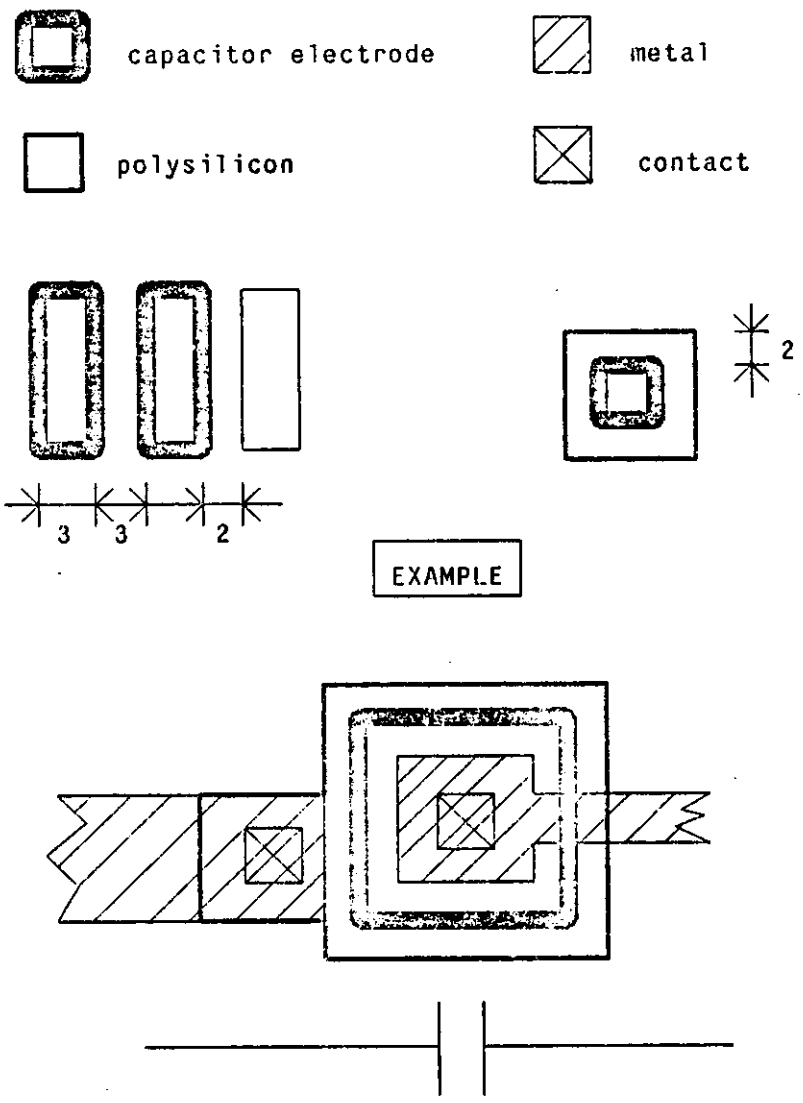


Figure 3-12: Design rules for capacitor

4. CMOS-SOS

CMOS-SOS is being offered experimentally by MOSIS with a $\lambda = 2\mu$ (4-micron feature size). Unlike Bulk, CMOS-SOS is not affected by latchup, because the sapphire substrate intrinsically isolates the n-channel from p-channel devices and viceversa. An excellent introduction to CMOS-SOS has been written by Charles Seitz and can be obtained either by sending a "Request: Information - Topic: SOS.INF" to MOSIS or in /usr/cmos/doc/sos.notes. Note that what was called "diffusion" in nMOS and "active area" in CMOS-Bulk, is called "island" in CMOS-SOS. CMOS-SOS is easier to design than CMOS-Bulk, because the well is not needed and latchup does not exist. It is possible to use lambda rules.

4.1. Design Rules

A "verbose" presentation of the design rules now follows. The design rules for the pads are not included and can be found in the document mentioned above.

- Island-poly minimum widths and spacings are 2λ. Minimum metal spacing and width is 3λ.
- As far as implant, it must overlap the p-island for 1.5λ and not come closer than 1.5λ to the n-island. Poly extension beyond island is 2λ(to make a transistor), while poly to island spacing is 2λ. Minimum spacing between poly over island (transistor) and contact is 2λ.
- The contact size is 2λ x 2λ with poly (or island) overlap of 1λ. Metal must overlap a contact for 1λ. Contact-to-contact spacing is 2λ.
- As in CMOS-Bulk, it is possible to short n-island and p-island. This shorting contact is 2λ by 4λ, with a 1λ surround, making it 4λ by 6λ overall, and is p-island on one end and n-island on the other end. The overall geometry is therefore identical to that used in CMOS-Bulk.
- If n-island and p-island are treated as layers, their minimum spacing is 3λ.

In Figs. 4-1 and 4-2 a graphic representation of the design rules is presented. In Fig. 4-3 the basic inverter is depicted.

Unfortunately, these design rules do not allow to "edit" a CMOS-Bulk cif file and automagically create a CMOS-SOS chip (e.g. there is a mismatch between the two poly-metal contacts). It could have been an interesting experiment.

4.2. Bulk or SOS?

The question cannot have a definite answer. Nevertheless, a comparison between the two technologies is interesting to be made. First of all, let us summarize the pro's and con's for each technology:

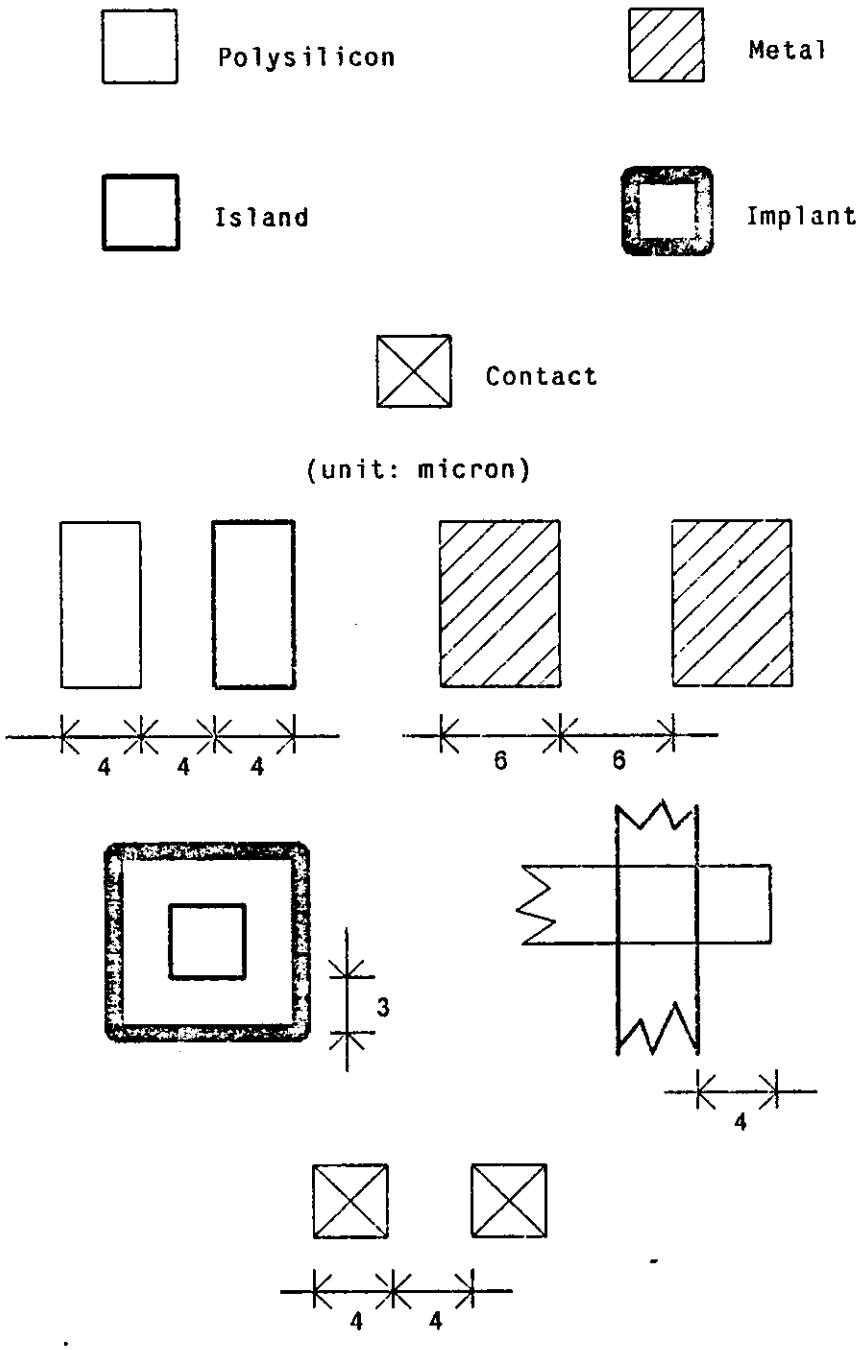


Figure 4-1: CMOS-SOS MOSIS Design Rules (a)

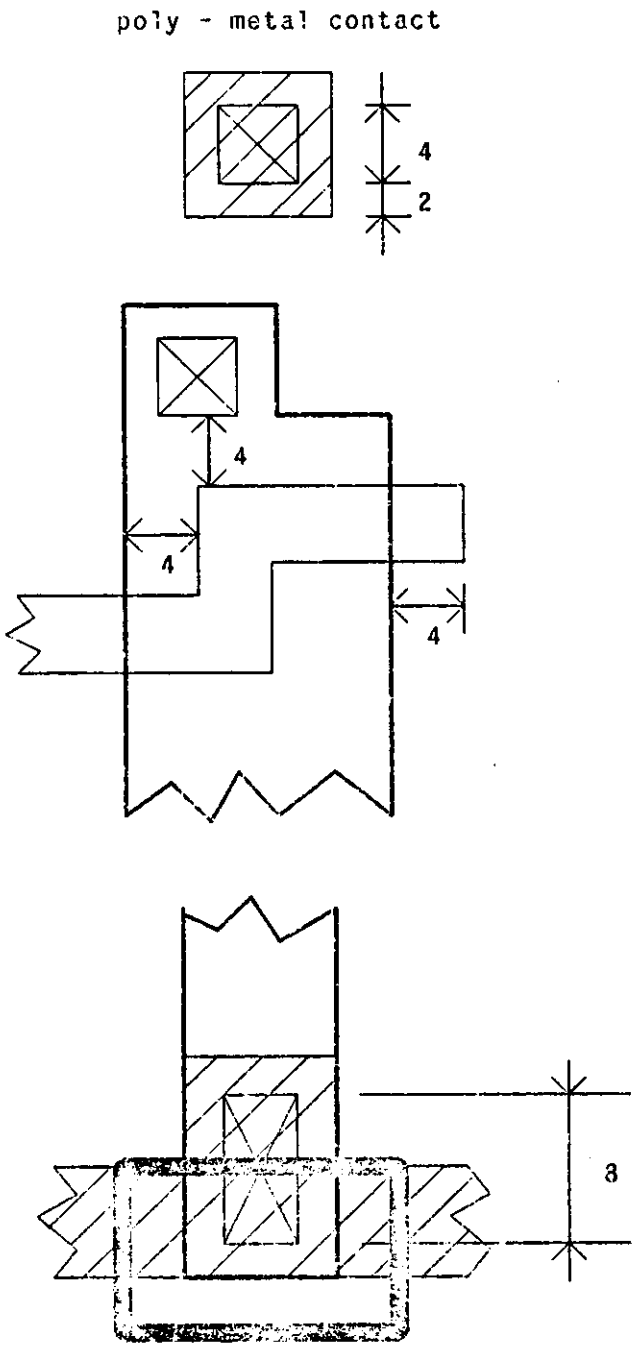


Figure 4-2: CMOS-SOS MOSIS Design Rules (b)

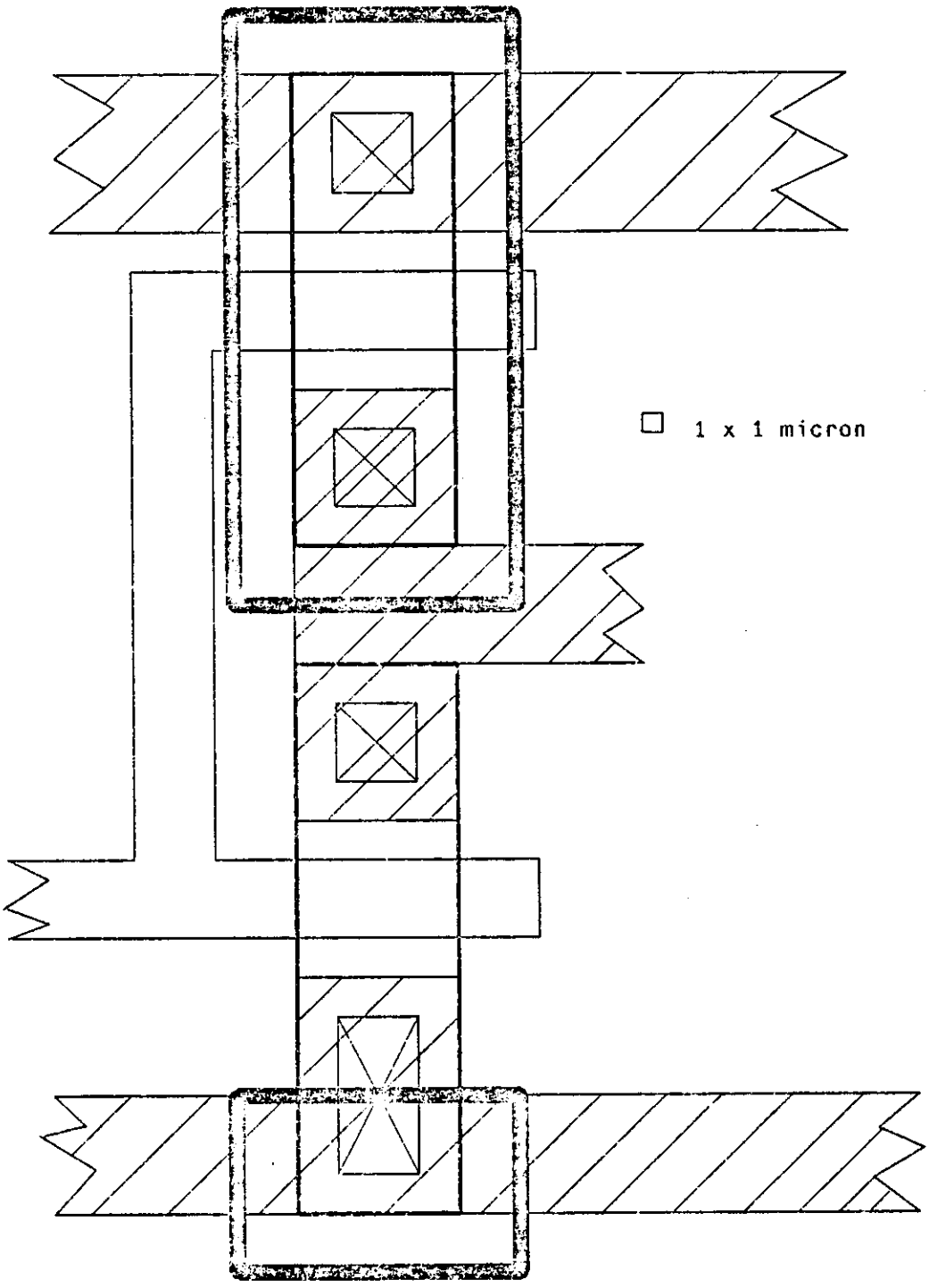


Figure 4-3: CMOS-SOS inverter

CMOS-Bulk: pro's

- very good noise margin;
- faster than nMOS;
- reliable and commercially viable fabrication process.

CMOS-Bulk: con's

- latchup (twin-tub and/or more complex fabrication processes can reduce the problem, but SOS behaves definitely much better);
- a guard-ring is needed to achieve radiation hardening;
- lower circuit density than nMOS and CMOS-SOS;
- design rules are more complex than either nMOS or CMOS-SOS: they also heavily depend on the fabrication process of the tub (both doping level and sinking depth).

CMOS-SOS: pro's

- much faster than nMOS; roughly twice as fast as a "comparable" CMOS-Bulk process;
- very good noise margin;
- intrinsically radiation hardened;
- allows high integration (but less than nMOS);
- no latchup;
- design rules are simpler than CMOS-Bulk.

CMOS-SOS: con's

- expensive fabrication process (due to the sapphire);
- sapphire variability;
- thermal mismatch between the sapphire substrate and silicon limits the carrier mobility;
- some experts claim that the fabrication process is critical and less reliable than the CMOS-Bulk process; however, the technology has improved lately and a number of defects have been eliminated (for an overview of these problems, see [19] pp.82-83 and references);
- back channel leakage: not a viable technology for dynamic storage.

Many experts agree that, while CMOS-SOS can hardly replace CMOS-Bulk, CMOS-SOI (Silicon-On-Insulator) is "the" technology of the future; the problem is to find an insulator without the drawbacks of sapphire. For more information about this topic, the reader is referred to [12], [22], [13] and [11].

5. How to start: Caesar, Lyra and other tools

Some modifications have to be made to some tools in order to make them capable to deal with CMOS. In this section the same tools will be considered for both CMOS-Bulk and CMOS-SOS.

CAESAR.

Caesar supports CMOS, both CMOS-Bulk and CMOS-SOS. It is up to you to decide your own colormap and technology files. A technology file for a CMOS-Bulk process is shown below. This technology file supports two metal layers, two different contacts and the capacitor electrode. The names of the layers are MOSIS compatible (/usr/cmos/cmos-pw.tech):

```
cmos-pw
/usr/cmos/cmos-pw.barco
polysilicon pr 0 solid 1
L CP
diffusion dg 0 solid 2
L CD
metal mb 0 solid 4
L CM
p-well w 377 stipple 10
L CW
210 42 210 42 210 42 210 42
p-plus Py 0 solid 20
L CS
contact c 377 cross 40
L CC
overglass o 0 solid 41
L CG
errors e 0 stipple 42
L CZ
377 0 377 0 377 0 377 0
metal-2nd-lvl a 377 ll-ur 43
L CM2
second-contact s 377 horizontal 44
L CC2
capacitor-electrode q 377 vertical 45
L CE
```

In order to use this technology file you have simply to put the cmos-pw.tech file and the cmos-pw.barco file (colormap) in your home directory and *in each other working directory* a .caesar file with the command:

technology /usr/cmos/cmos-pw.tech

The colormap is (/usr/cmos/cmos-pw.barco):

```
p 230 red 0 green 0 blue          ; poly
d 0 red 255 green 0 blue          ; active area (green)
m 0 red 0 green 220 blue          ; metal
w 150 red 100 green 0 blue        ; P-well
P 255 red 255 green 100 blue      ; P+
c 0 red 0 green 0 blue            ; contact
o 100 red 100 green 100 blue      ; cut in the overglass
e 255 red 255 green 255 blue      ; error
a 250 red 0 green 250 blue        ; second metal layer
s 0 red 0 green 0 blue            ; second contact
q 0 red 255 green 255 blue        ; capacitor electrode
```

Needless to say, you can choose your own colormap, simply using the three caesar commands:*color, cload, csave*. As far as the pattern of the layers, you have to modify the cmos-pw.tech file (see the caesar manual). As far as CMOS-SOS, you can simply use the nMOS technology file and rename. before either simulating or submitting the chip, all the layers according to the following CIF names[7]:

```
SIS or SE   SOS island,
SP          SOS polysilicon,
SIM or SI   SOS implant,
SM          SOS metal,
SC          SOS contact cut,
SG          SOS overglass opening.
```

Finally, it is impossible not to mention another graphic editor that could have been very useful, i.e. *electric* [16]. Electric has many useful features that make a structured design methodology easier:

- it is library-oriented;
- it handles the connectivity inside a cell and among cells;
- it has predefined symbols (e.g. p-channel, n-channel, poly-metal contact etc.).

The only serious drawbacks that electric has are the absence of a cif2electric program and the difficulty to embed a new technology. In a very short time a version of electric that can cope with the MOSIS-CMOS fabrication process should be available.

A third interesting IC layout system, presently under development at Berkeley, is *Caddy*. As far as we know, it should feature many of the interesting and useful characteristics electric has. It should also have a built-in channel router.

[7] There is also an explicit layer for pads to be bonded (XP); this is used when your circuit uses overglass openings elsewhere in the chip.

CIFPLOT.

Cifplot, in its original form, clearly does not work. CMOS-Bulk differs from nMOS for number of layers and CIF names (e.g. nMOS poly is NP, CMOS-Bulk poly is CP). However, it is sufficient to introduce a new bit-map with the proper layer names and use the option -P for cifplot.

A solution is to introduce, in your .cshrc file, the line:

alias cmosplot 'cifplot -P /usr/cmos/.plotcmos \!'*

The .plotcmos file could look like the following (/usr/cmos/.plotcmos):

```
"CP" 0x08080808  0x04040404  0x02020202  0x01010101
     0x80808080  0x40404040  0x20202020  0x10101010
"CD" 0x22222222 0x00000000 0x88888888 0x00000000
     0x22222222 0x00000000 0x88888888 0x00000000
"CM" 0x01010101 0x00000000 0x10101010 0x00000000
     0x01010101 0x00000000 0x10101010 0x00000000
"CW" 0x01010101 0x02020202 0x00000000 0x00000000
     0x10101010 0x20202020 0x00000000 0x00000000
"CS" 0x11111111 0x11111111 0x11111111 0x11111111
     0x11111111 0x11111111 0x11111111 0x11111111
"CC" 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
     0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
"CG" 0x1C1C1C1C 0x3E3E3E3E 0x36363636 0x3E3E3E3E
     0x1C1C1C1C 0x00000000 0x00000000 0x00000000
"CM2" 0x0C0C0C0C 0x1E1E1E1E 0x36363636 0x1E1E1E1E
      0x0C0C0C0C 0x10101010 0x10101010 0x10101010
"CC2" 0x03030303 0x07070707 0x07070707 0x07070707
      0x18181818 0x07070707 0x07070707 0x07070707
"CE" 0x22222222 0x28282828 0x88888888 0x282FF828
     0x222FF222 0x28282828 0x88888888 0x28282828
"CZ" 0xFFFFFFFF 0x00000000 0x00000000 0x00000000
     0x00000000 0x00000000 0x00000000 0x00000000
```

Giving the huge number of layers, a cifplot is hardly readable. However, if you are really going to decipher a cmos-plot, in Fig. 5-1 and Fig. 5-2 you can find the layers (basic and composite) as the bit-map shown above has produced.

As usual, if you do not like the way the layers are plotted, you can always change the bit-map. As far as CMOS-SOS is concerned, you can simply use the nMOS bit map.

Figure 5-1: Basic and composite layers (a)

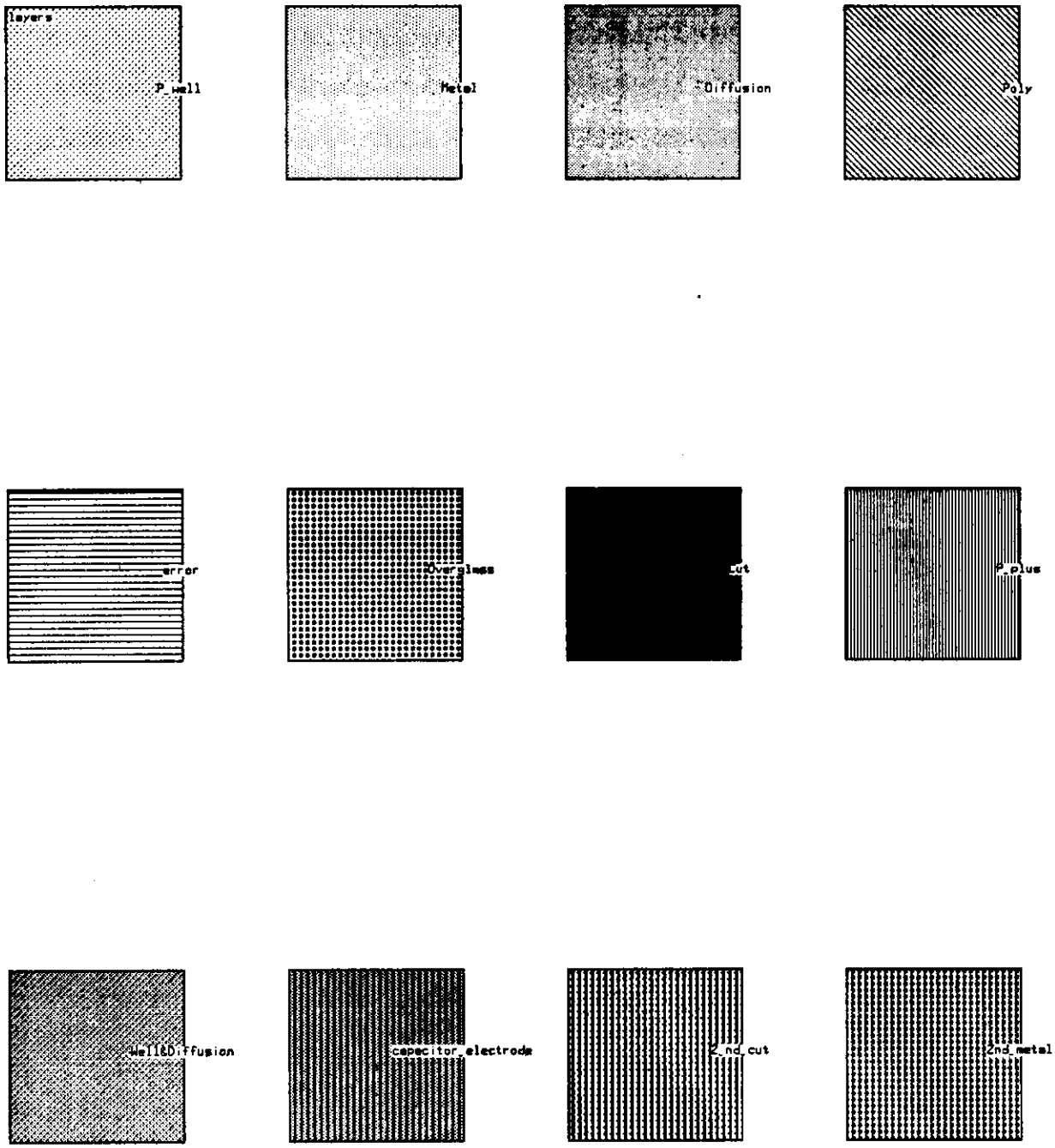
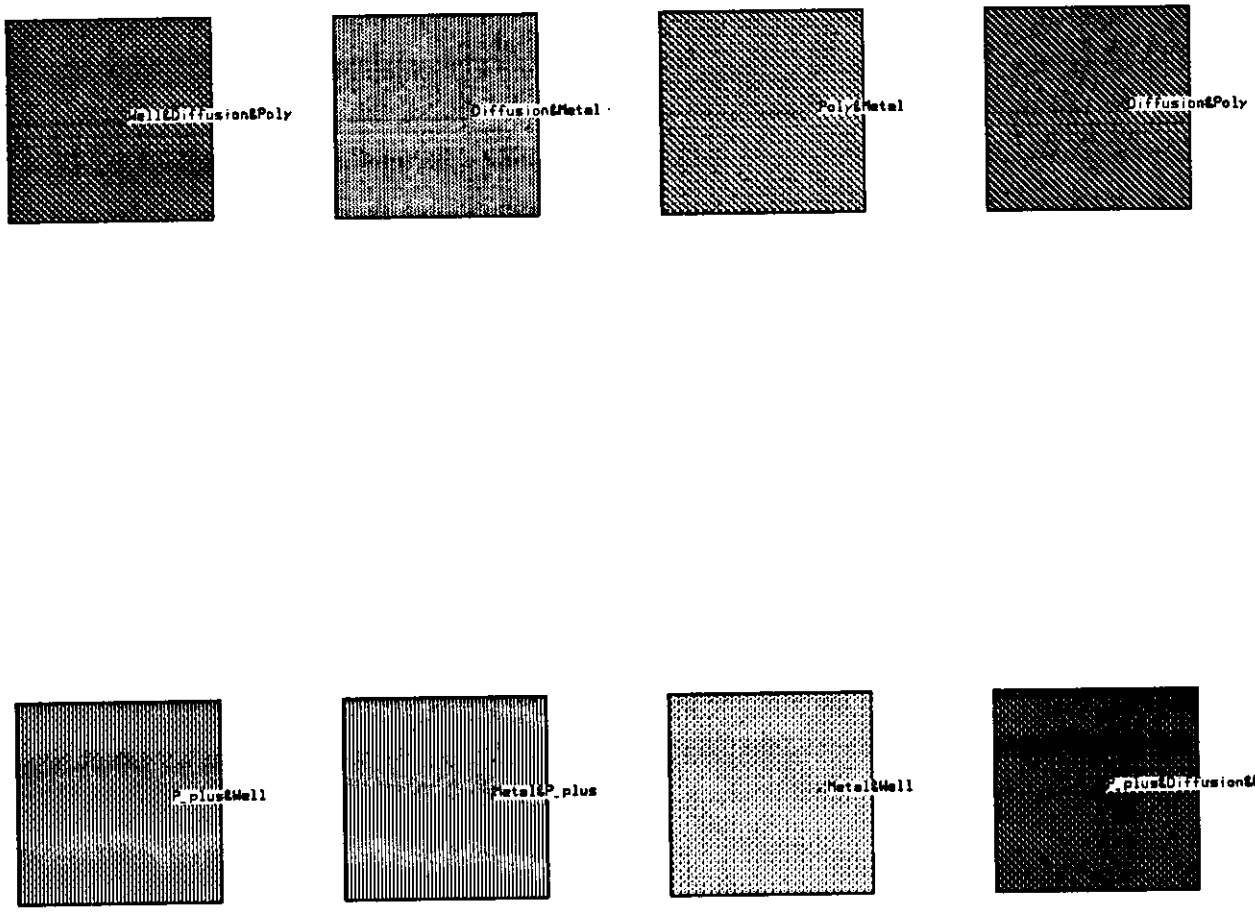


Figure 5-2: Basic and composite layers (b)



CIF2CA

No problems for CMOS-SOS: it is always possible to let cif2ca believe that we are designing an nMOS chip; for CMOS-Bulk cif2ca has already embedded the technology.

LYRA.

Lyra had already embedded the cmos-pwJPL design rules. Alan Sussman has embedded the CMOS-Bulk MOSIS design rules (for both the cbpe2 and the cbpem2 processes) and the CMOS-SOS design rules for a 2λ process. As far as CMOS-Bulk is concerned, lyra can work only if you consider one caesar unit equal to 100 centimicrons. A minimum-width poly line is therefore 3 caesar unit (i.e. 300 centimicrons). The lyra design rule checker for cbpe2 is called *plyra* while for cbpem2 (with second metal layer and second contact) is called *p2lyra.* See the "man" entry at the end of this document. In order to use the design rule checkers, simply put

alias plyra "lyra -r /usr/cmos/lyra/pwell/cmos-pwMOSIS"
alias p2lyra "lyra -r /usr/cmos/lyra/pwell/cmos-pwm2MOS"

in your .cshrc file.

Lyra does not check *maximum* dimensions. This is not a problem, apart from the contact. Therefore, it would be better to predefine a subcell containing a poly-metal contact and define a macro command in caesar to "get" the cell. Finally, lyra does not check design rules related to pads or scribe.

MEXTRA.

Mextra supports CMOS. Nevertheless some modifications were necessary. The version of mextra able to cope with both CMOS-Bulk P-well and CMOS-SOS is called *cmextra* and its "man" entry is at the end of this manual. Mextra did not know about CMOS-SOS and therefore it was necessary to embed this knowledge in the program. To run cmextra, simply add /usr/cmos/bin in your PATH.

Apart from that, some precautions are necessary during the labelling of the nodes. The correct procedure is here presented.

First of all, each Vdd node should be labelled Vdd!, while each ground node should be labelled

GND!. This allows other programs (e.g. sim2spice) to work better. What mextra does with nodes labelled with "!" is to consider them *global names* and therefore to *automatically* connect them together.

It is evident that, if you are looking for some missing connection in the Vdd or GND paths, this feature is a drawback. You can use the -g option to "disconnect" them.

SIM2SPICE

Sim2spice produces, from a .sim file, a .spice file. However, it is not possible to feed spice with this file for problems of formatting. A program, called *cformat* does the job (see the "man" entry at the end of this manual). To run cformat, simply add /usr/cmos/bin in your PATH.

SPICE.

Obviously, spice has no problem with CMOS. If you have labelled each Vdd node with Vdd! and each GND node with GND!, sim2spice will automatically assign node #1 to each node connected to Vdd and node #0 to each node connected to ground. Moreover, as far as the substrate is concerned, it will assign two different numbers (2 and 3) for the p-channel substrate and the n-channel substrate. You can either ignore these two numbers or set them to Vdd (1) and GND (0) respectively. In order to know the number of each node you are interested in, simply look at the .nodes file generated by cmextra.

The usual problem is the definition of the model with the .model cards. A "fast" model and a "slow" model are included. However, please, keep in mind that:

1. We are "too far" from the site of fabrication to be able to have a precise control on the parameters of the fabrication process;
2. the variance of the parameters from one process to another is even greater than in the nMOS process;
3. the model is largely incomplete;
4. the designer will use these models at his/her own risk.

```
                FASTMODEL
.model n nMOS vto=0.4 tox=0.7e-7 lambda=1e-7 ld=1.0e-6
+xj=1.1e-6 gamma=.3 uo=500 cbd=5e-4 cbs=5e-4

.model p pmos vto=-0.4 tox=0.7e-7 lambda=1e-7 ld=1.0e-6
+xj=1.1e-6 gamma=.3 uo=300 cbd=3.5e-4 cbs=3.5e-4

                SLOWMODEL
.model n nMOS vto=1.0 tox=0.8e-7 lambda=1e-7 ld=.5e-6
+xj=.6e-6 gamma=1.3 uo=400 cbd=6e-4 cbs=6e-4

.model p pmos vto=-1.0 tox=0.8e-7 lambda=1e-7 ld=.5e-6
+xj=.6e-6 gamma=.9 uo=200 cbd=4.1e-4 cbs=4.1e-4
```

Remember that:

- It is possible to use, if necessary, a slow n-channel model and a fast p-channel model (or viceversa);
- the "fast" model is probably too optimistic and the "slow" model is too pessimistic;
- the difference in performance between the fast and the slow model is considerable (see the results referred in appendix A);
- the holes and electrons mobility (uo) is generally *different* (the electrons mobility is from one and a half to twice the holes mobility) and this would change the pull-down/pull-up ratio: a discussion of this problem is presented in the introduction to CMOS-SOS by Charles Seitz; the author claims that a 1:1 ratio has more advantages than drawbacks while other communities, MIT for instance [4], commonly use a 2:1 ratio between pull-down and pull-up.

CRYSTAL

The first release of crystal could not be modified in order to deal with CMOS, basically because it only had built-in information on the transistor types. The second version is table-driven and therefore it is possible to make it capable to handle CMOS circuits. This work is presently in progress.

NET/NL/RNL

NET, NL and RNL can handle CMOS circuits. A slightly different version of NET has been produced; it is called *cnet* and has embedded the concept of inverter, nand, nor and transmission gate for CMOS. Therefore you do not have to define a macro but just use these predefined elements as you were designing nMOS circuits. See the "man" entry of cnet at the end of the manual. To run cnet, include

/usr/cmos/bin in your PATH.

RNL and NL work fine with CMOS. The only problem arises in signal-powered gates (as in Fig. 2-10). It seems that both RNL and NL do not understand this topology. It is difficult to claim that this behavior is typical, as few circuits have been checked. If you are using this design methodology, you should better watch out when you use RNL/NL.

What RNL and NL do not like is the syntax of the .sim file as produced by either cmextra or mextra. A program, called *cformat*, will take care of generating a .sim file (from a .sim file produced by mextra) which is RNL/NL compatible. See the "man" entry of *cformat* at the end of this report.

PLA GENERATOR

Presently we do not have any PLA generator either for CMOS-Bulk or CMOS-SOS. There is a CMOS-SOS PLA generator at Caltech and various CMOS-Bulk PLA generators at MIT, however. A CMOS PLA generator is not easy to build, as already pointed out in chapter 2. One possible solution is to design the appropriate "tiles" that, used by *tpla*, will generate a CMOS PLA.

What could be done more easily would be a ROM generator. Although ROM generators are less common (and useful) than PLA generators, two considerations can be made:

- Small PLA's very often use all the minterms and therefore could be implemented more efficiently with a ROM;
- sense amplifiers can be more easily designed in CMOS; this would allow the implementation of very large ROM's (especially with a second metal layer available).

The idea of building a ROM generator can be considered feasible and useful, therefore. In the next future we shall try to find out a solution to this problem.

5.1. Future trends

CMOS design in the universities is presently (January 1984) a fairly intricate and unclear business. Therefore, some "practical" information that was previously given can become obsolete in few months. What follows is a possible trend of the CMOS fabrication process, as offered by MOSIS.

- *3 μ fabrication process.* A 3 μ fabrication process will be available for at least a decay. What can change in the near future are the design rules, because different vendors, other than the two presently used, will be present.
- *Second metal layer.* On January 19th 1984 the first official run for a 3μ P-well CMOS

fabrication process featuring a second metal layer has taken place.

- *turn-around.* Starting from the next year, the turn-around is expected to be about two months.

- *1.25 μ CMOS.* In 1984 a 1.25 μ process will be offered. This process will have a slow turn-around, presumably. The design rules of this process *cannot* be scaled down from the 3 μ process. A second metal layer will be available. Four vendors are available.

As far as design tools are concerned, it is reasonable to forecast that, in the near future, caesar will be abandoned; either caddy, electric or something with the same philosophy will be used (symbol-oriented, built-in router, library-oriented, built-in design rule checker). In the near future, *TV* (the Stanford timing analyzer) will be available; some modifications will be necessary because it cannot handle CMOS. SPLICE will also substitute SPICE in many applications.

Appendix A
CMOS-Bulk: I/O Pads

When this document was written, no pads were available from MOSIS. When they will be available, the best thing is to use them. In the meanwhile it is possible to use either "very experimental" pads designed at MIT or "extremely experimental" pads that have been designed here, at CMU.
There are important differences between the two sets of pads (CMU pads - MIT pads):

- CMU pads are far less conservative, as far as design rules, and therefore could be more affected by latchup;
- MIT pads, on the other hand, are very large;
- MIT pads were already partially tested; CMU pads have not been tested yet[8].

A.1. CMU I/O Pads

The[9] scope of this section is to present some pads that were being used while MOSIS pads were not available yet. Their spice simulation is presented. The complete set of pads can be found in /usr/cmos/cbpe2/pads.
The pads are:

- *padin*;
- *padout*;
- *padvdd*;
- *padgnd*.

PADIN
The protection is achieved via a guard-ring and a resistance-diode structure. The resistance is simply a long poly line.
PADOUT
The output pad features guard-rings for protection. Its design is still far from a stable situation.
An I/O pad will be available when the "final" version of the output pad has been designed.

The output pad was simulated with SPICE using the two different .model cards shown in this document. The load was the TTL load used in [6] to simulate the nMOS output pad. The circuit is simply four inverters scaled by approximatively a factor of 3 (see Fig. A-1). The spice deck is shown in

[8] A comprehensive analysis of I/O pads will be available in the second part of this report, available during 1984.

[9] Remember that these pads were designed with $\lambda = 1\mu$ and therefore, if you are using $\lambda = 1.5\mu$, they need to be modified.

Fig. A-2. The slow-model and fast-model output responses are shown, respectively, in Fig. A-3 and Fig. A-4.

A.2. MIT Pads

The complete set of MIT pads can be found in /usr/cmos/pads/MIT. Basically, a resistance-diode structure is used for protection.

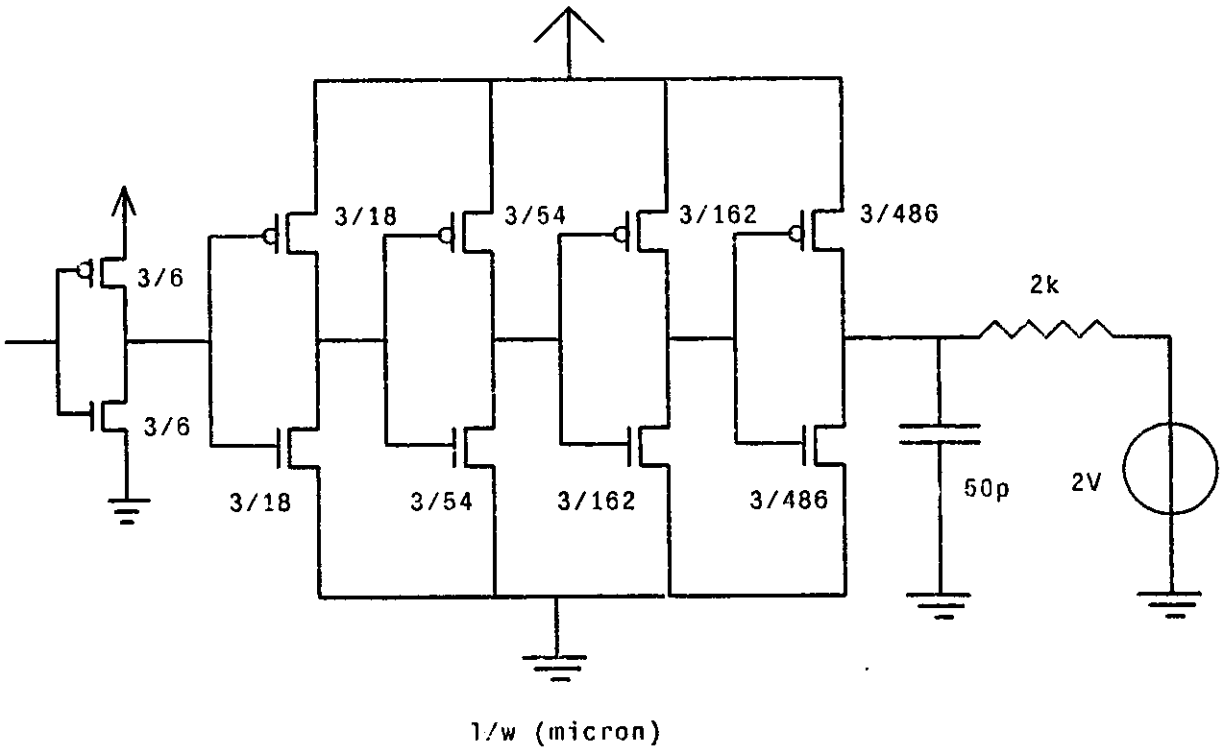


Figure A-1: Output pad: input gate, pad circuitry and load

Figure A-2: Circuit description for Spice

```
.option nonode nopage noacct nolist nomod
.width in = 80 out = 80

vdd 1 0 5
vin 5 0 pwl(0ns 0 2ns 5 52ns 5 54ns 0 100ns 0)
.tran 1ns 100ns
.plot tran v(10) v(6) (0,5)

m1 1 5 6 3 p l = 3.0u w = 6.0u
m2 1 6 7 3 p l = 3.0u w = 18.0u
m3 1 7 8 3 p l = 3.0u w = 54.0u
m4 1 8 9 3 p l = 3.0u w = 162.0u
m5 1 9 10 3 p l = 3.0u w = 486.0u

m6 0 5 6 2 n l = 3.0u w = 6.0u
m7 0 6 7 2 n l = 3.0u w = 18.0u
m8 0 7 8 2 n l = 3.0u w = 54.0u
m9 0 8 9 2 n l = 3.0u w = 162.0u
m10 0 9 10 2 n l = 3.0u w = 486.0u

c1 10 0 50p
r1 10 11 2k
vtl 11 0 2

.model n nmos vto = 1.0 tox = 0.8e-7 lambda = 1e-7 ld = .5e-6
+ xj = .6e-6 gamma = 1.3 uo = 400 cbd = 6e-4 cbs = 6e-4

.model p pmos vto = -1.0 tox = 0.8e-7 lambda = 1e-7 ld = .5e-6
+ xj = .6e-6 gamma = .9 uo = 200 cbd = 4.1e-4 cbs = 4.1e-4

.end
```

Figure A-3: Spice output with a slow-model

```
0****    transient analysis          temperature =  25.000 deg c
0*************************************************************************

0legend:

 *: v(10)
 +: v(6)
x
   time     v(10)

x( * + )---------- 0.000e+00   1.250e+00   2.500e+00   3.750e+00 5.000e+00
                   ---------------------------------
 0.000e+00 4.833e+00.                                          * +
 1.000e-09 4.833e+00.                                        + * .
 2.000e-09 4.833e+00.    +                                     * .
 3.000e-09 4.836e+00. +                                        * .
 4.000e-09 4.611e+00. +                                      *   .
 5.000e-09 3.988e+00 +                                     *     .
 6.000e-09 3.378e+00 +                                   *       .
 7.000e-09 2.789e+00 +                                 *         .
 8.000e-09 2.233e+00 +                               *           .
 9.000e-09 1.740e+00 +                             *             .
 1.000e-08 1.328e+00 +                           *               .
 1.100e-08 9.978e-01 +                         *                 .
 1.200e-08 7.435e-01 +                       *                   .
 1.300e-08 5.525e-01 +                     *                     .
 1.400e-08 4.120e-01 +                   *                       .
 1.500e-08 3.099e-01 +                 *                         .
 1.600e-08 2.364e-01 +                 *                         .
 1.700e-08 1.839e-01 +               *                           .
 1.800e-08 1.466e-01 +               *                           .
 1.900e-08 1.201e-01 +               *                           .
 2.000e-08 1.014e-01 +               *                           .
 2.100e-08 8.825e-02 +               *                           .
 2.200e-08 7.895e-02 +               *                           .
 2.300e-08 7.239e-02 +               *                           .
 2.400e-08 6.778e-02 +               *                           .
 2.500e-08 6.453e-02 +               *                           .
 2.600e-08 6.225e-02 +               *                           .
 2.700e-08 6.064e-02 +               *                           .
 2.800e-08 5.951e-02 +               *                           .
 2.900e-08 5.871e-02 +               *                           .
 3.000e-08 5.816e-02 +               *                           .
 3.100e-08 5.776e-02 +               *                           .
 3.200e-08 5.749e-02 +               *                           .
 3.300e-08 5.729e-02 +               *                           .
 3.400e-08 5.716e-02 +               *                           .
 3.500e-08 5.706e-02 +               *                           .
 3.600e-08 5.699e-02 +               *                           .
 3.700e-08 5.694e-02 +               *                           .
 3.800e-08 5.691e-02 +               *                           .
 3.900e-08 5.689e-02 +               *                           .
 4.000e-08 5.687e-02 +               *                           .
 4.100e-08 5.686e-02 +               *                           .
 4.200e-08 5.685e-02 +               *                           .
 4.300e-08 5.685e-02 +               *                           .
 4.400e-08 5.684e-02 +               *                           .
 4.500e-08 5.684e-02 +               *                           .
 4.600e-08 5.684e-02 +               *                           .
 4.700e-08 5.684e-02 +               *                           .
 4.800e-08 5.683e-02 +               *                           .
 4.900e-08 5.683e-02 +               *                           .
 5.000e-08 5.683e-02 +               *                           .
 5.100e-08 5.683e-02 +               *                           .
 5.200e-08 5.683e-02 +               *                           .
 5.300e-08 5.685e-02. *  +                                       .
 5.400e-08 5.681e-02. *                 +                        .
 5.500e-08 5.684e-02. *                                        + .
 5.600e-08 1.285e-01. *                                        + .
 5.700e-08 4.728e-01.   *                                      +
 5.800e-08 8.093e-01.     *                                    +
 5.900e-08 1.139e+00.       *                                  +
 6.000e-08 1.461e+00.         *                                +
 6.100e-08 1.775e+00.           *                              +
 6.200e-08 2.081e+00.             *                            +
 6.300e-08 2.376e+00.               *                          +
 6.400e-08 2.656e+00.                 *                        +
 6.500e-08 2.916e+00.                   *                      +
 6.600e-08 3.156e+00.                     *                    +
 6.700e-08 3.373e+00.                       *                  +
 6.800e-08 3.568e+00.                         *                +
 6.900e-08 3.742e+00.                           *              +
 7.000e-08 3.895e+00.                             *            +
```

Figure A-4: Spice output with a fast-model

```
0****   transient analysis        temperature =  25.000 deg c
0*************************************************************

0legend:

 *: v(10)
 +: v(6)
x
   time    v(10)

x(* +)---------- 0.000e+00   1.250e+00   2.500e+00   3.750e+00  5.000e+00
                 ---------------------------
 0.000e+00 4.956e+00.                                        x
 1.000e-09 4.956e+00.                                        *
 2.000e-09 2.775e+00 +                                       .
 3.000e-09 9.953e-01 +                                       .
 4.000e-09 3.379e-01 +                                       .
 5.000e-09 1.214e-01 +                                       .
 6.000e-09 4.864e-02 +                                       .
 7.000e-09 2.673e-02x                                        .
 8.000e-09 2.047e-02x                                        .
 9.000e-09 1.868e-02x                                        .
 1.000e-08 1.817e-02x                                        .
 1.100e-08 1.802e-02x                                        .
 1.200e-08 1.798e-02x                                        .
 1.300e-08 1.797e-02x                                        .
 1.400e-08 1.796e-02x                                        .
 1.500e-08 1.796e-02x                                        .
 1.600e-08 1.796e-02x                                        .
 1.700e-08 1.796e-02x                                        .
 1.800e-08 1.796e-02x                                        .
 1.900e-08 1.796e-02x                                        .
 2.000e-08 1.796e-02x                                        .
 2.100e-08 1.796e-02x                                        .
 2.200e-08 1.796e-02x                                        .
 2.300e-08 1.796e-02x                                        .
 2.400e-08 1.796e-02x                                        .
 2.500e-08 1.796e-02x                                        .
 2.600e-08 1.796e-02x                                        .
 2.700e-08 1.796e-02x                                        .
 2.800e-08 1.796e-02x                                        .
 2.900e-08 1.796e-02x                                        .
 3.000e-08 1.796e-02x                                        .
 3.100e-08 1.796e-02x                                        .
 3.200e-08 1.796e-02x                                        .
 3.300e-08 1.796e-02x                                        .
 3.400e-08 1.796e-02x                                        .
 3.500e-08 1.796e-02x                                        .
 3.600e-08 1.796e-02x                                        .
 3.700e-08 1.796e-02x                                        .
 3.800e-08 1.796e-02x                                        .
 3.900e-08 1.796e-02x                                        .
 4.000e-08 1.796e-02x                                        .
 4.100e-08 1.796e-02x                                        .
 4.200e-08 1.796e-02x                                        .
 4.300e-08 1.796e-02x                                        .
 4.400e-08 1.796e-02x                                        .
 4.500e-08 1.796e-02x                                        .
 4.600e-08 1.796e-02x                                        .
 4.700e-08 1.796e-02x                                        .
 4.800e-08 1.796e-02x                                        .
 4.900e-08 1.796e-02x                                        .
 5.000e-08 1.796e-02x                                        .
 5.100e-08 1.796e-02x                                        .
 5.200e-08 1.796e-02x                                        .
 5.300e-08 1.796e-02*                                        .
 5.400e-08 1.015e+00.                                        +
 5.500e-08 2.656e+00.                                        +
 5.600e-08 3.701e+00.                                        +
 5.700e-08 4.309e+00.                                        +
 5.800e-08 4.628e+00.                                        +
 5.900e-08 4.791e+00.                                        +
 6.000e-08 4.873e+00.                                        +
 6.100e-08 4.914e+00.                                        +
 6.200e-08 4.935e+00.                                        +
 6.300e-08 4.945e+00.                                        +
 6.400e-08 4.950e+00.                                        +
 6.500e-08 4.953e+00.                                        +
 6.600e-08 4.954e+00.                                        +
 6.700e-08 4.955e+00.                                        +
 6.800e-08 4.955e+00.                                        +
 6.900e-08 4.955e+00.                                        +
 7.000e-08 4.955e+00.                                        +
```

Figure A-5: CMU Input Pad

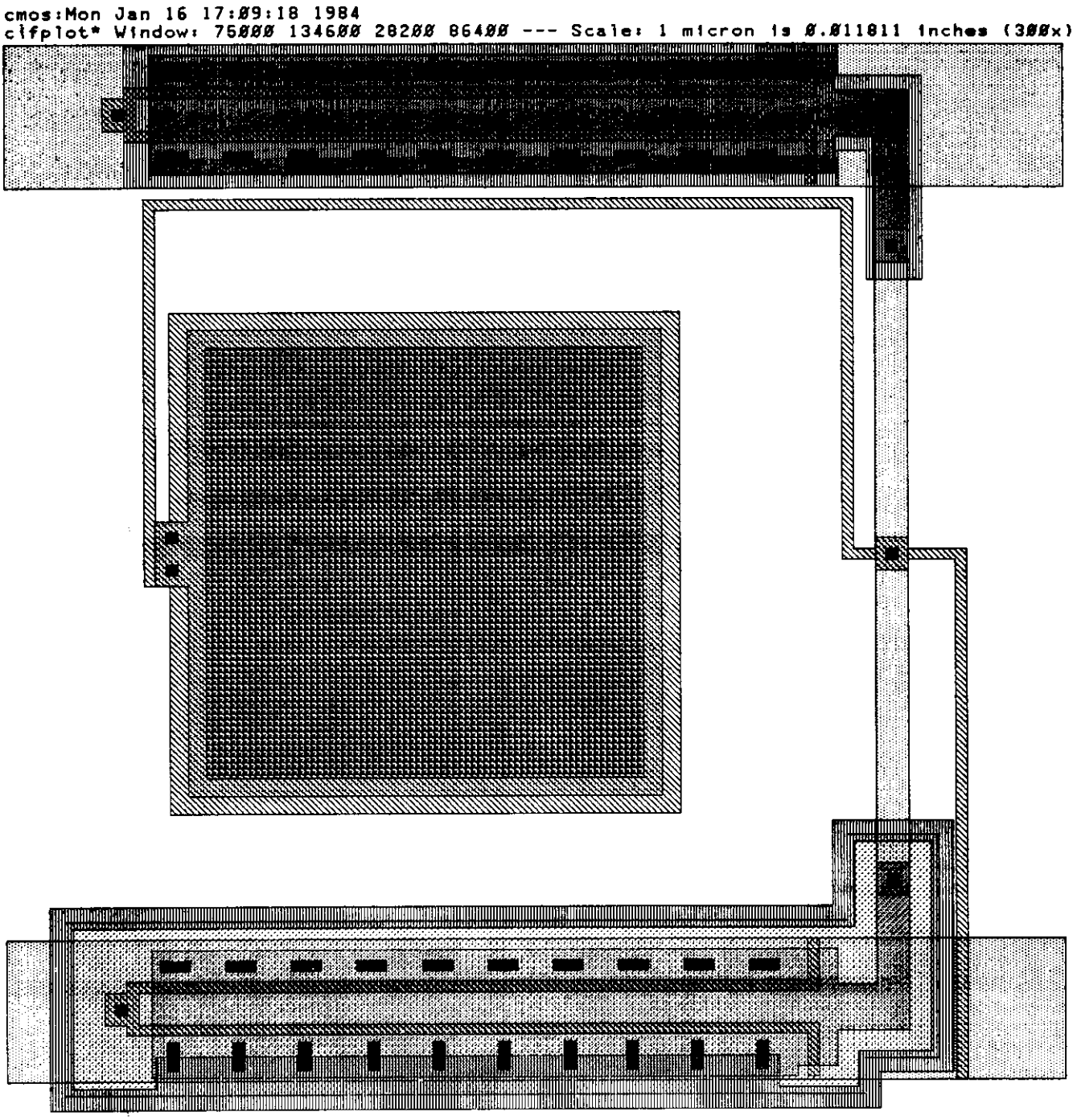


Figure A-6: CMU Output Pad

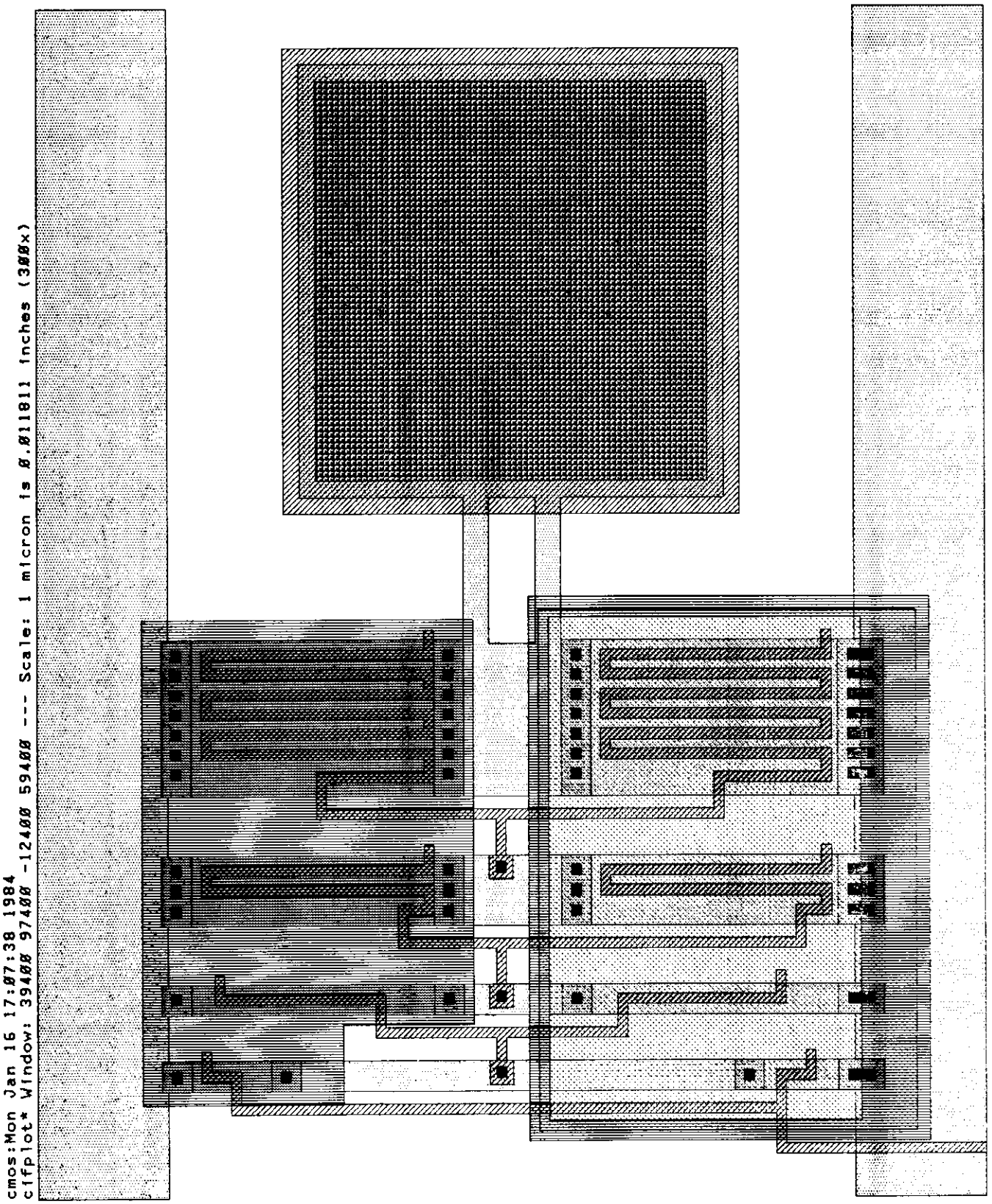


Figure A-7: MIT Input Pad

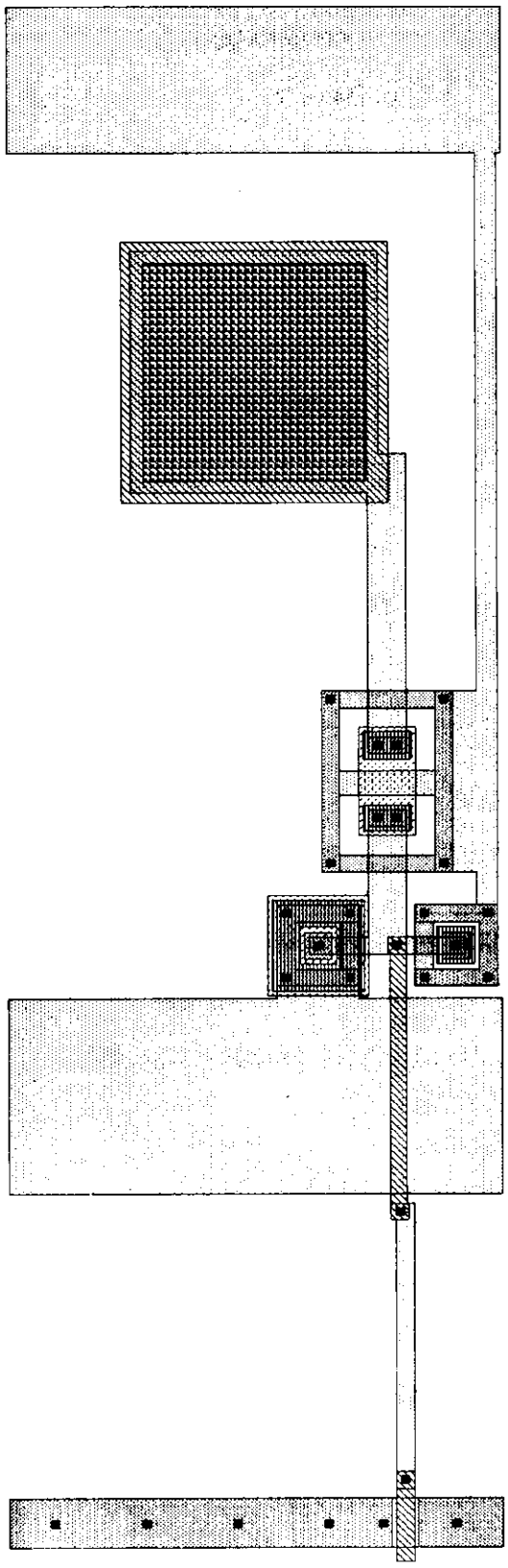


Figure A-8: MIT Output Pad

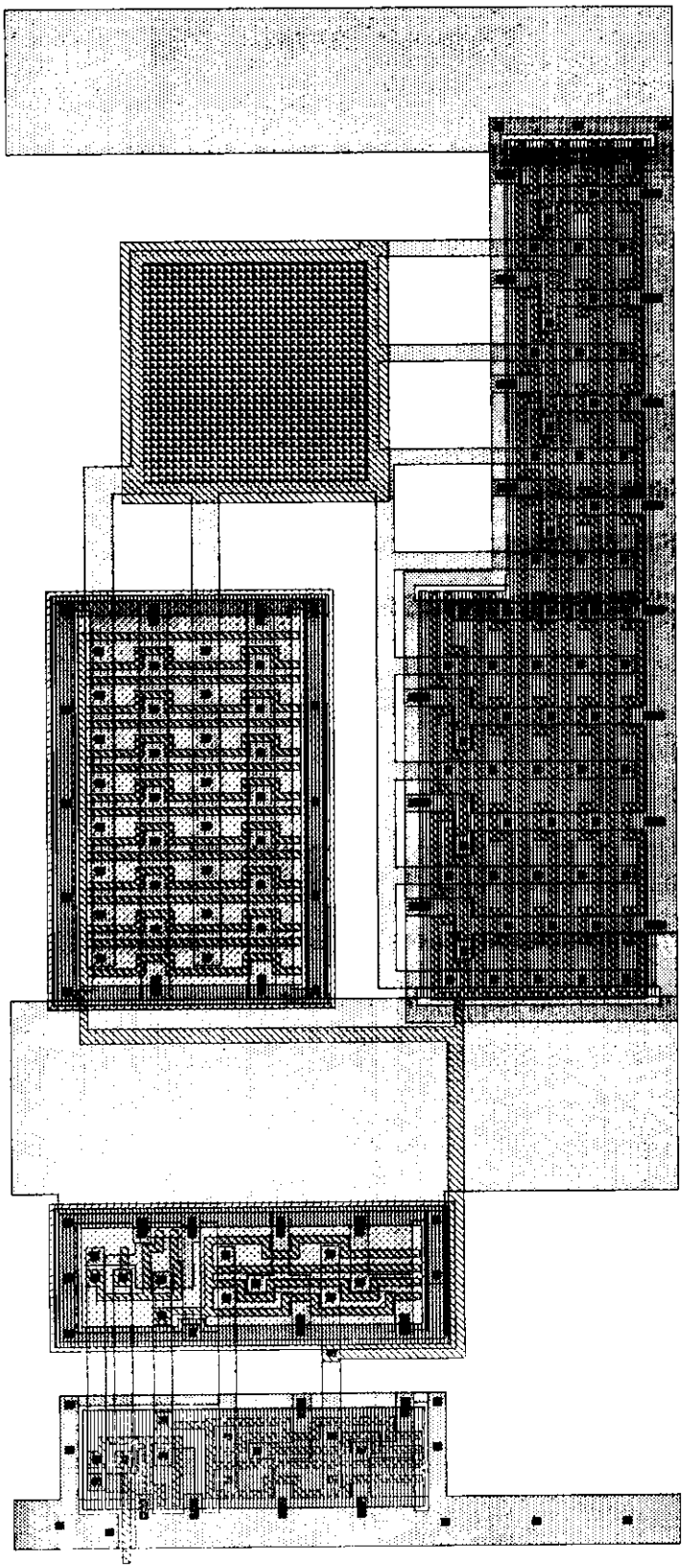


Figure A-9: MIT Vdd Pad

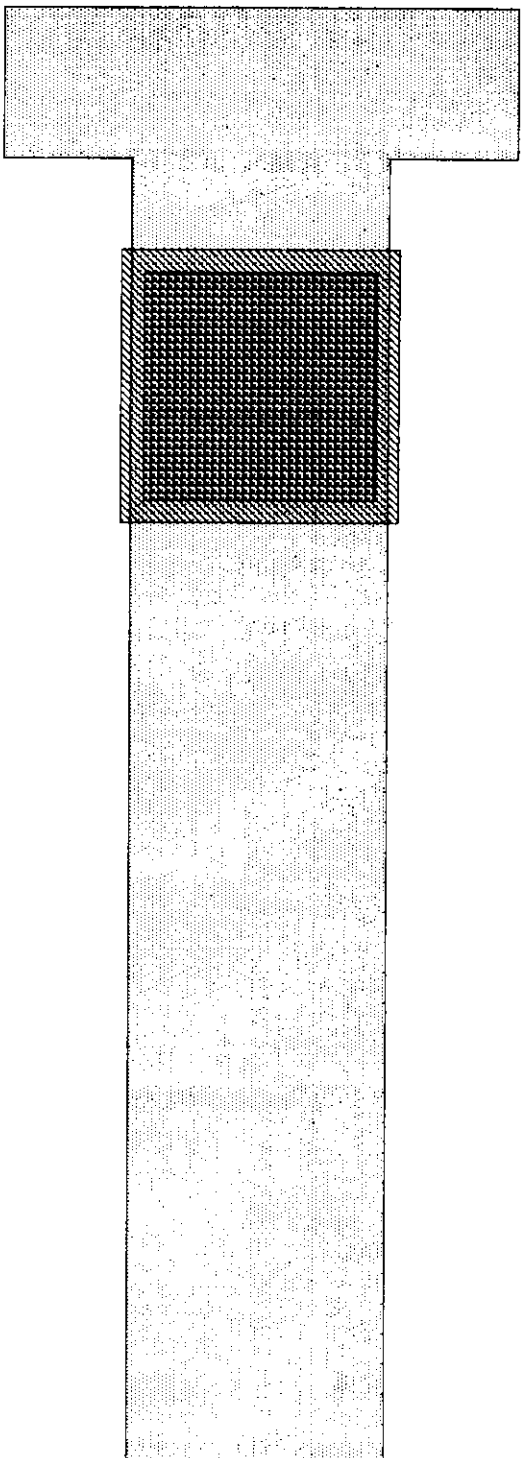


Figure A-10: MIT GND Pad

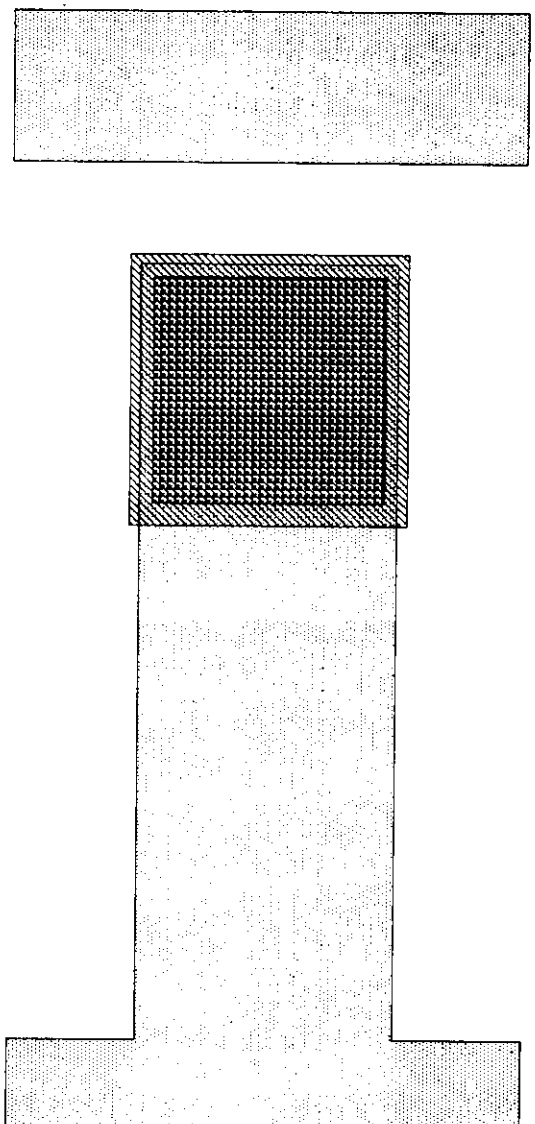
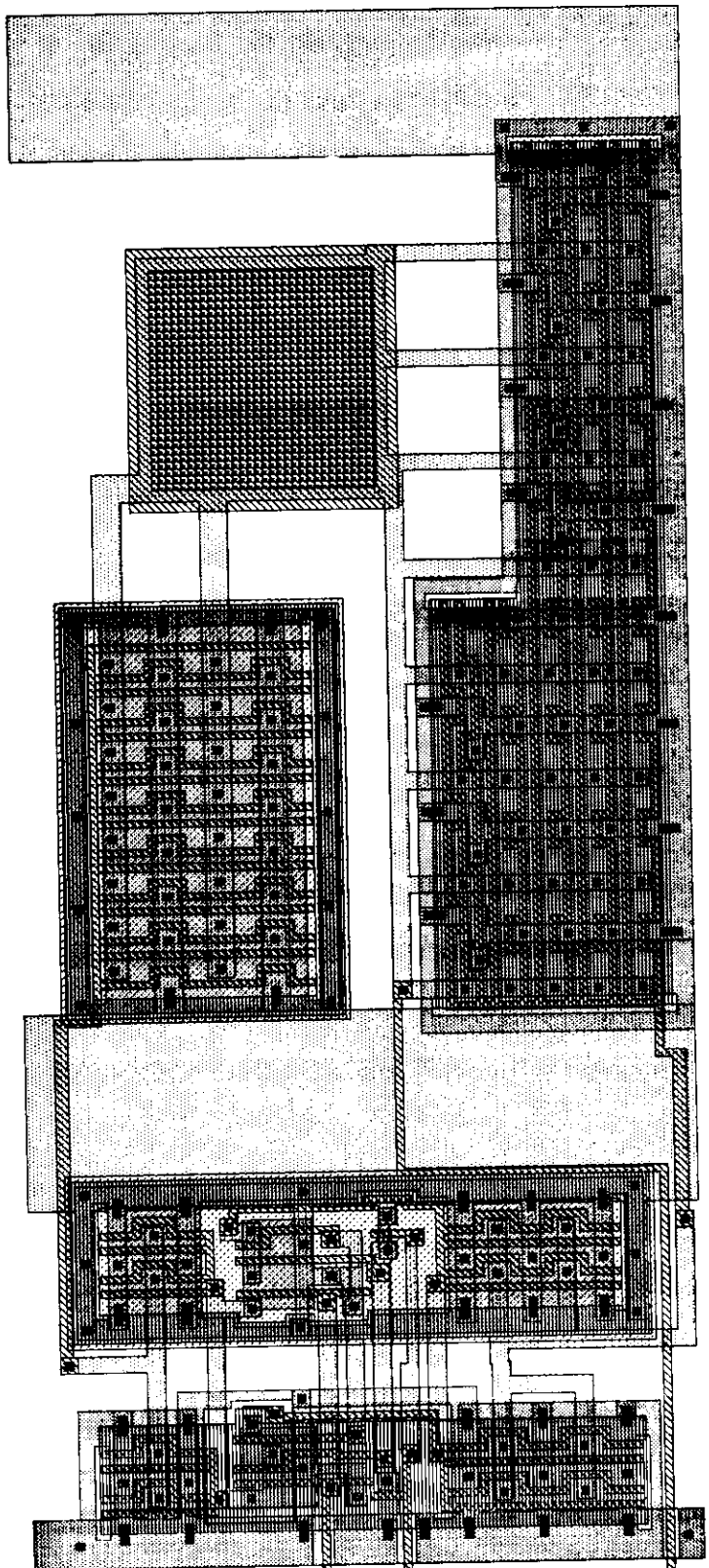


Figure A-11: MIT I/O Pad



Appendix B
CMOS-SOS: I/O Pads

These are the CMOS-SOS pads used at Caltech in the CMOS-SOS design course held by Charles Seitz. The following is the document, as supplied by MOSIS, on the SOS pads.

***

CMOS SOS PAD LIBRARY DESCRIPTION

The library consists of a set of CMOS SOS pads designed for a 2.5μ lambda process. All pads have 48 lambda square overglass holes and are designed to be packed (in any order) with a 100 lambda pitch. Since pads are all 104 lambda wide, they are expected to overlap 4 lambda when packed densely.

All pads have the geometry in padblank in common. This includes the pad per se, and 8-lambda-wide power and ground wires. Their center lines are:

Vdd: (-2,-4) to (102,-4)
Ground: (-2,-100) to (102,-100)

Note that except for these, the connection points to all pads are on the lower boundary of the cell.

THESE PADS ARE UNTESTED!!!

	MBB			
CIF SYMBOL	LLX	LLY	URX	URY
padblank	-2	-104	102	0
padground	-2	-104	102	0
padvdd	-2	-104	102	0
padin	-2	-104	102	0
meanpadin	-2	-104	102	0
padout	-2	-135	102	0
tripadout	-2	-163	102	0

PADBLANK

Geometry common to all pads. Contains metal pad, overglass cut, vdd wire at top and ground wire at bottom.

PADGROUND

Padblank with a connection to the ground wire.

PADVDD

Padblank with a connection to the vdd wire.

PADIN

Input pad with lightning arrester consisting of a 1K ohm resistor and diodes to vdd and ground. Connection point: padin – out (94,-103) on island. See Fig. B-1.

MEANPADIN

Same as padin except that the 1K ohm resistor is replaced with a metal short. To be used when driving large on-chip loads, when the resistor would introduce significant delay. Meanpadin provides less static protection than does padin as a result. Connection point: padin – out (94,-103) on island. See Fig. B-2.

PADOUT

Drives the pad with the signal on padout – in amplified through 4 stages of inverter. Pad is driven to ground with approx. 75 ohms or to vdd with approx. 200 ohms. Connection point: padout – in (64,-134) on poly. See Fig. B-3.

TRIPADOUT

Tristate output pad. When the level on tripad – ena is HI, the pad is driven with the signal on tripad – in amplified through 4 stages of inverter. When the level on tripad – ena is LO, the pad is not driven at all. The level on the pad itself appears at the connection point tripad – out. See Fig. B-4.

```
Connection points: tripadena   (30,-162)  on poly
                   tripadin    (73,-162)  on poly
                   tripad-out   (96.5,-162) on metal
```

***

The complete set of SOS pads can be found in /usr/cmos/sos/pads.

Figure B-1: CMOS-SOS input pad

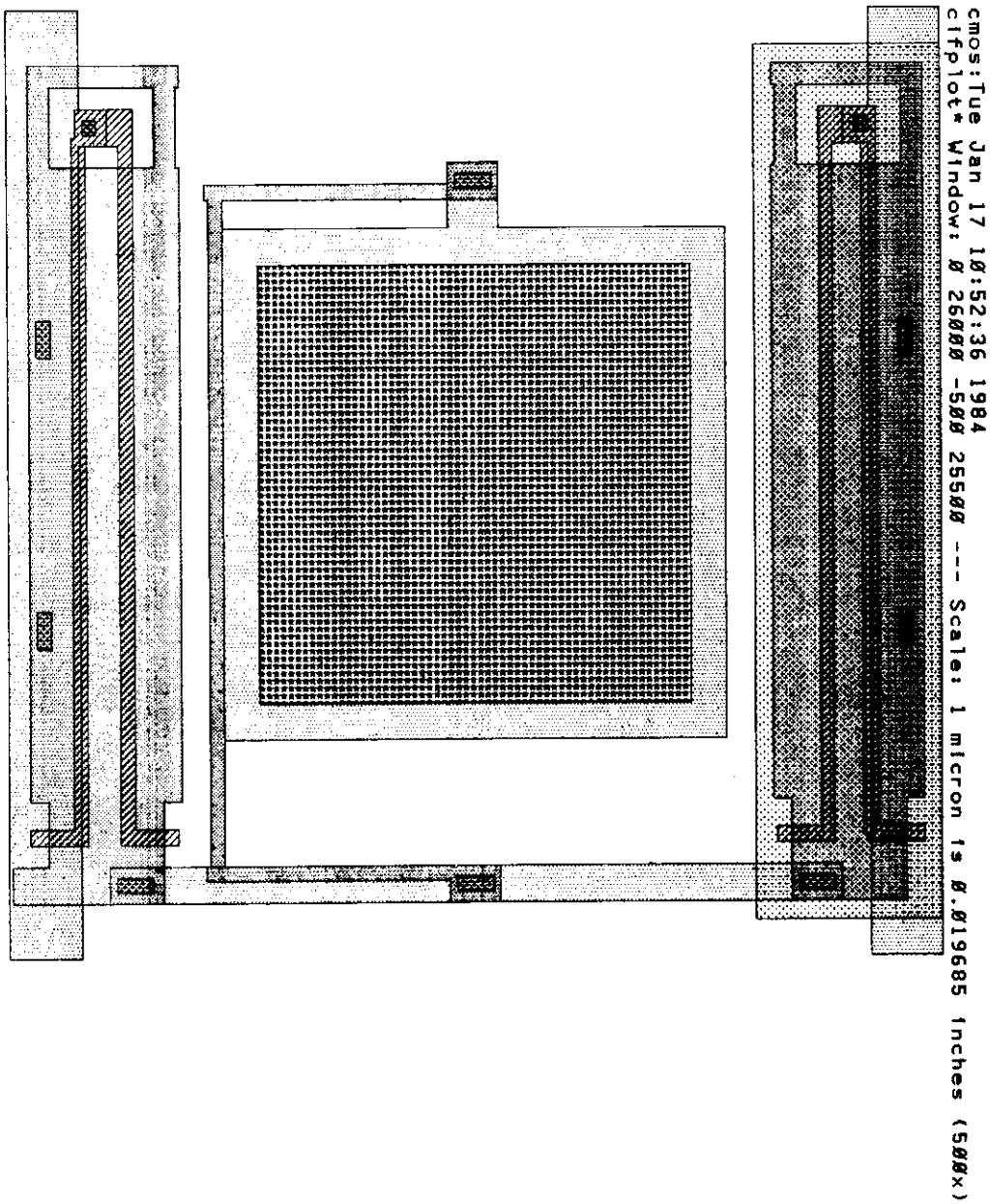


Figure B-2: SOS input pad without filtering resistance

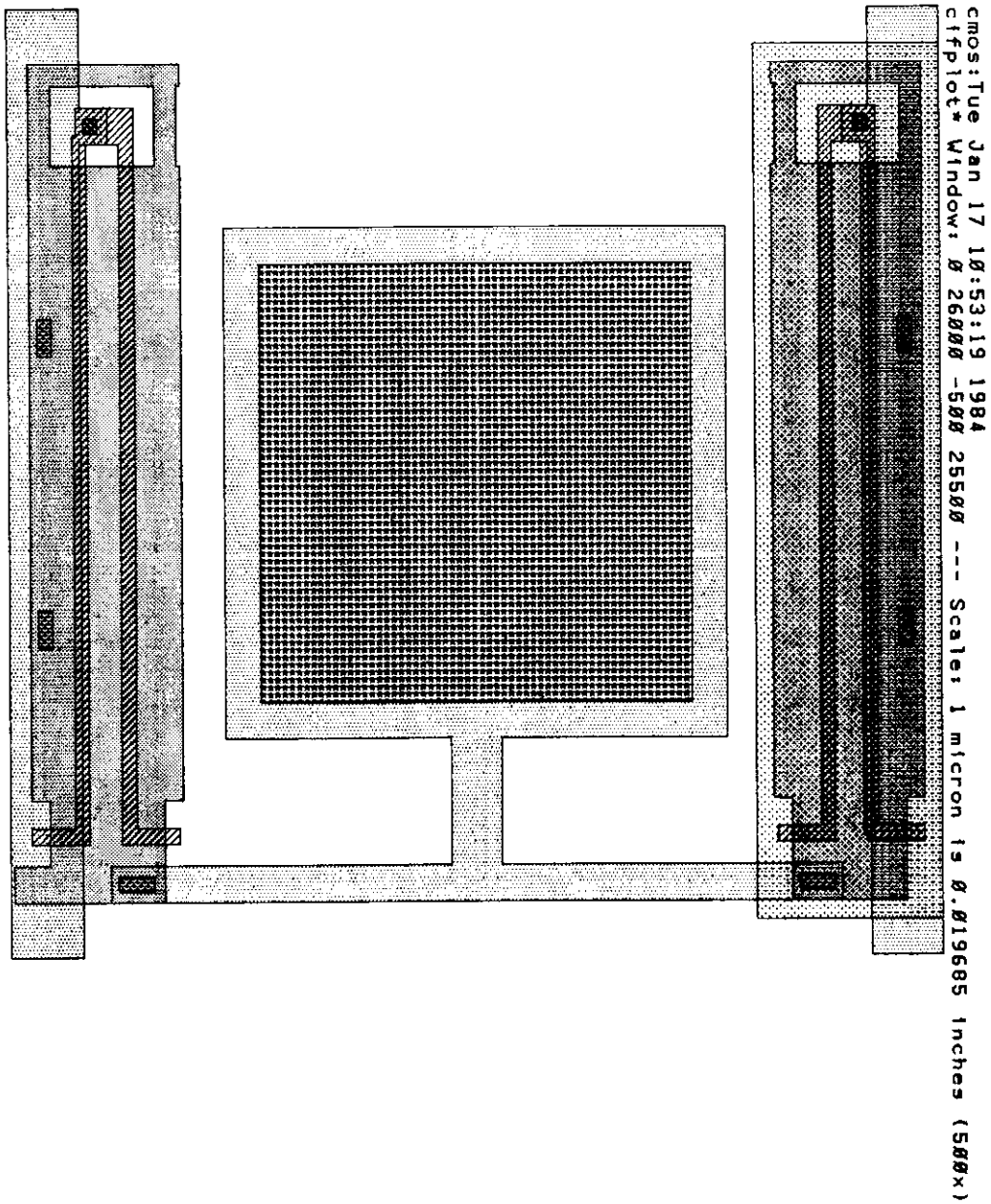


Figure B-3: CMOS-SOS output pad

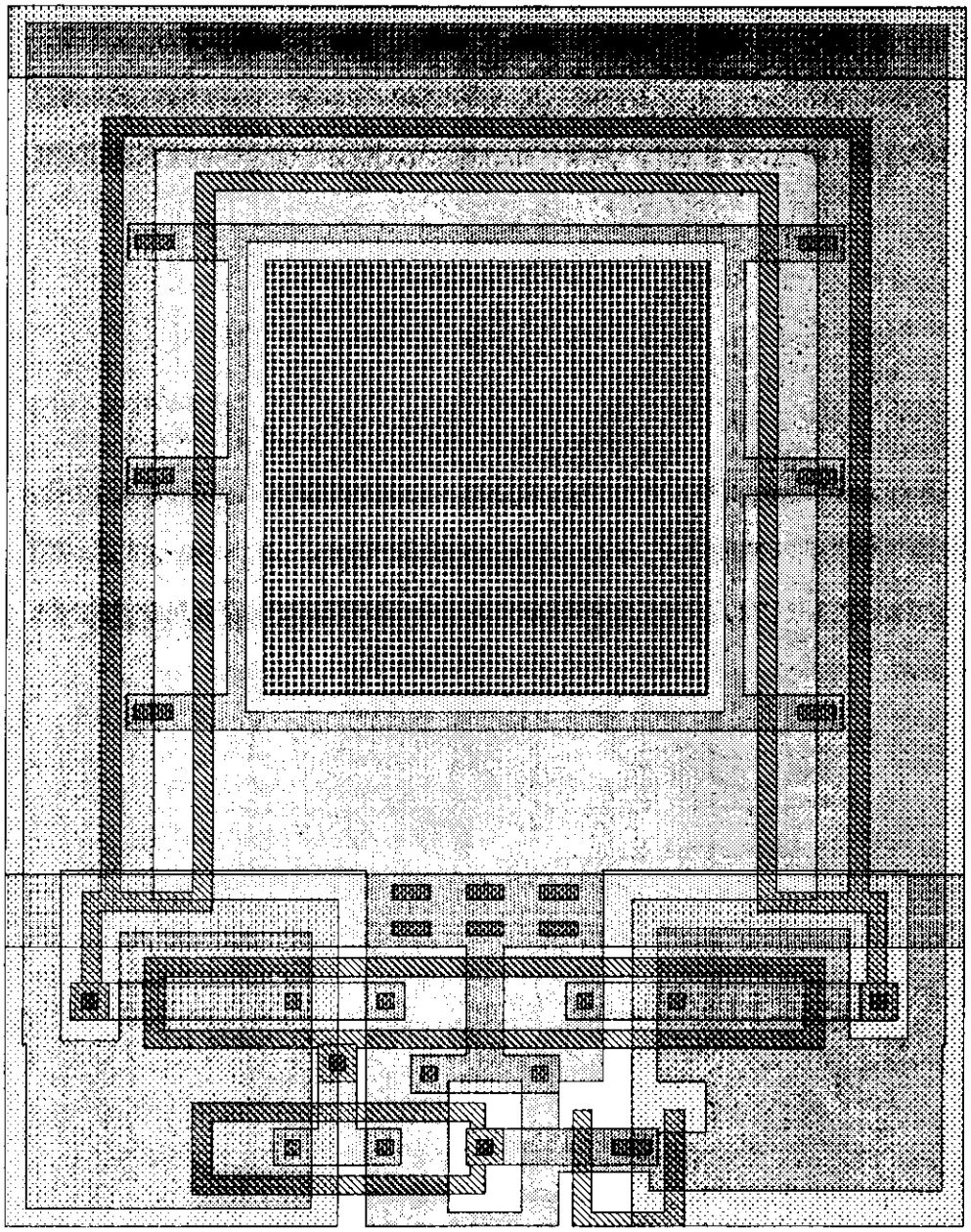
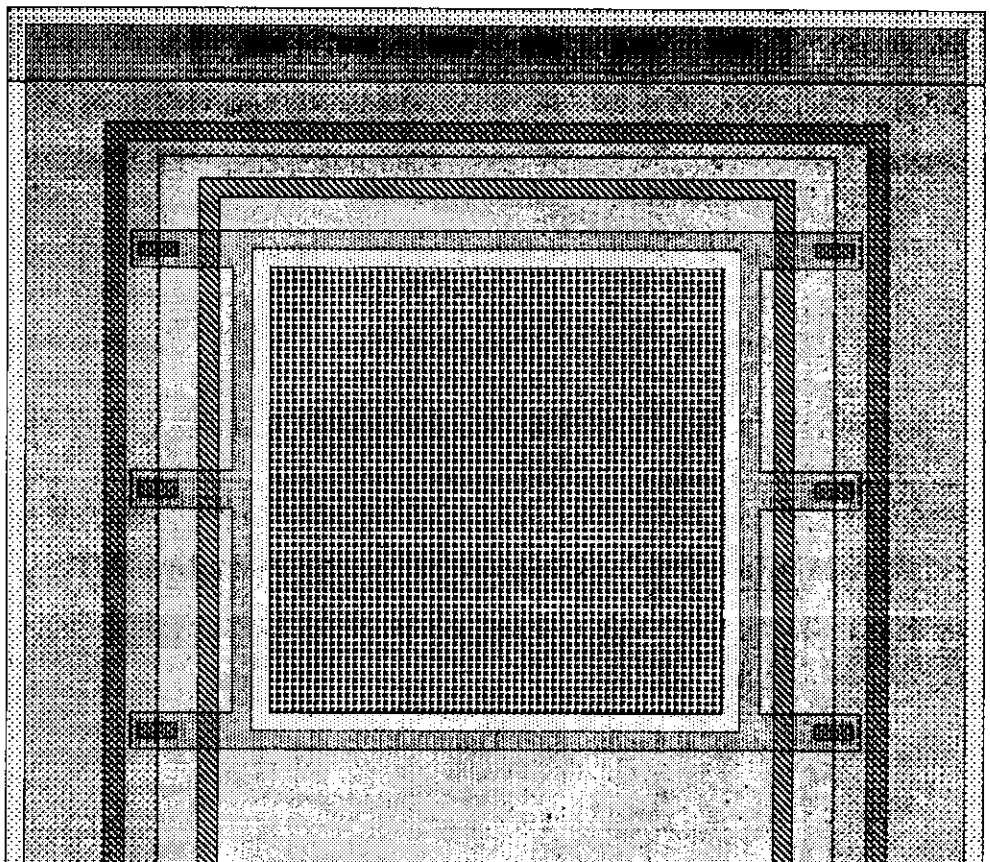


Figure B-4: CMOS-SOS I/O pad



References

[1] Allstot, D.J., Brodersen, R.W. and Gray, P.R.
MOS Switched capacitor ladder filters.
IEEE Journal Solid-state Circuits SC-13:806-814, December, 1978.

[2] Black, W.C. and Hodges, D.A.
Time Interleaved Converter Arrays.
IEEE Journal of Solid-state Circuits SC-15, 1980.

[3] Brodersen, R.W., Gray, P.R. and Hodges, D.A.
MOS switched capacitor filters.
Proc. IEEE SC-13:61-75, January, 1979.

[4] Glasser, L.A. and Song, W.S.
Introductory CMOS Techniques.
1982.
VLSI Memo No. 82-117, revised February 1983, unpublished.

[5] Gray, P.R. and Meyer, R.G.
MOS Operational Amplifier Design - A Tutorial Overview.
IEEE Journal of Solid-state Circuits SC-17(6):969-982, December, 1982.

[6] Hon, R.W. and Sequin, C.H.
A Guide to LSI Implementation.
Research Report SSL-79-7, XEROX-Parc, January, 1980.

[7] Hosticka, B.J.
Dynamic CMOS Amplifiers.
IEEE Journal of Solid State Circuits SC-15:887-894, 1980.

[8] Krambeck, R.H., Lee, C.M. and Law, H.S.
High-speed Compact Circuits with CMOS.
IEEE Journal of Solid State Circuits SC-17(3):614-619, June, 1982.

[9] Manoliu, J., Tseng, F.H., Woo, B.J. and Meier, T.J.
High-Density and Reduced Latchup Susceptibility CMOS Technology for VLSI.
IEEE Electron Device Letters EDL-4(7):233-235, July, 1983.

[10] Mavor, J., Jack, M.A. and Denyer, P.B.
Microelectronics Systems Design: Introduction to MOS LSI Design.
Addison-Wesley Publishers Ltd., 1983.

[11] Nishizawa, J. (ed.).
Japan Annual Reviews in Electronics, Computers and Telecommunications. : Semiconductor Technologies.
North-Holland Publishing Co., 1982, pages 296-307chapter 22.

[12] Okuto, Y., Fukuma, M. and Ohno, Y.
SOS/CMOS as a High-Performance LSI Device.
IEEE Trans. on Electron Devices ED-29(4):574-577, April, 1982.

[13] Oldham, H.F. and Partridge, S.L.
A Comparative Study of CMOS Processes for VLSI Applications.
IEEE Trans. on Electron Devices ED-29(10):1593-1598, October, 1982.

[14] Pattanayak, D.N., Kinoshita, G., Nelson, J.H. and Wong, Y.
Switching Conditions for CMOS Latch-up Path with Shunt Resistances.
IEEE Electron Device Letters EDL-4(4):116-119, April, 1983.

[15] Payne, R.S., Grant, W.N. and Bertram, W.J.
Elimination of Latch Up in Bulk CMOS.
In *Proc. of the International Electron Devices Meeting*, pages 248-251. IEEE, 1980.

[16] Rubin, S.
An Integrated Aid for Top-down Electrical Design.
In *Proc. ICCAD*, pages 111-112. September, 1983.

[17] Saari, V.R.
Low-power High-drive CMOS Operational Amplifiers.
IEEE Journal of Solid State Circuits SC-18(1):121-127, February, 1983.

[18] Sugino, M., Akers, L.A. and Rebeschini, M.E.
Latchup-free Schottky-barrier CMOS.
IEEE Transaction on Electron Devices, 1983.

[19] Sze, S.M. (ed.).
VLSI Technology.
McGraw-Hill Book Co., 1983.

[20] Tago, H., Kobayashi, T. Kobayashi, M. Moriya, T. and Yamamoto, S.
A 6K-gate CMOS Gate Array.
IEEE Journal of Solid-state Circuits SC-17(5):907-912, October, 1982.

[21] Troutman, R.R. and Zappe, H.P.
A Transient Analysis of Latchup in Bulk CMOS.
IEEE Trans. on Electron Devices ED-30(2):170-179, February, 1983.

[22] White, M.H.
Characterization of CMOS Devices for VLSI.
IEEE Trans. on Electron Devices ED-29(4):578-584, April, 1982.

[23] Wieder, A.W., Werner, C. and Harter, J.
Design Model for Bulk CMOS Scaling Enabling Accurate Latchup Prediction.
IEEE Trans. on Electron Devices ED-30(3):240-245, March, 1983.

INFORMATION SCIENCES INSTITUTE
ERSITY OF SOUTHERN CALIFORNIA
4676 Admiralty Way/Marina del Rey/California 90291
(213) 822-1511

MOSIS

New Standard CMOS/bulk 3 μm Design-Rules Aug-82

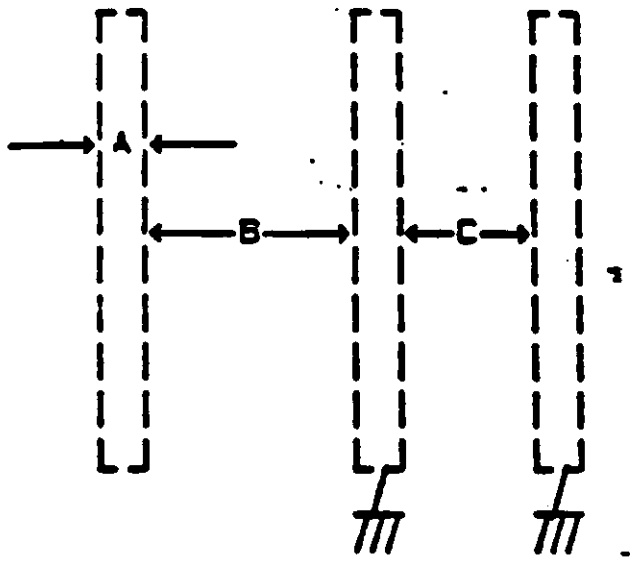
2.0 Geometric Line Key: The following key provides the symbols used to describe the Topological Layout Rules.

LEVEL		ALIGNS TO	LINE KEY
2.1	P-WELL	WAFER FLAT	
2.2	P+ RING	P-WELL	
2.3	ACTIVE AREA	P-WELL	
2.4	POLY	ACTIVE AREA	
2.5	P+	ACTIVE AREA	
2.6	N+	ACTIVE AREA	
2.7	CONTACT	POLY	
2.8	METAL	CONTACT	
2.9	PASSIVATION	METAL	

3. Topological Layout Rules: (Revised Layout Rules)

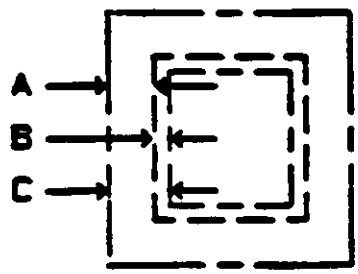
3.1 P-well

a. P-well width	3μ	____
b. P-well to P-well spacing (different potentials)	15	____
c. P-well to P-well spacing (same potential)	9	____



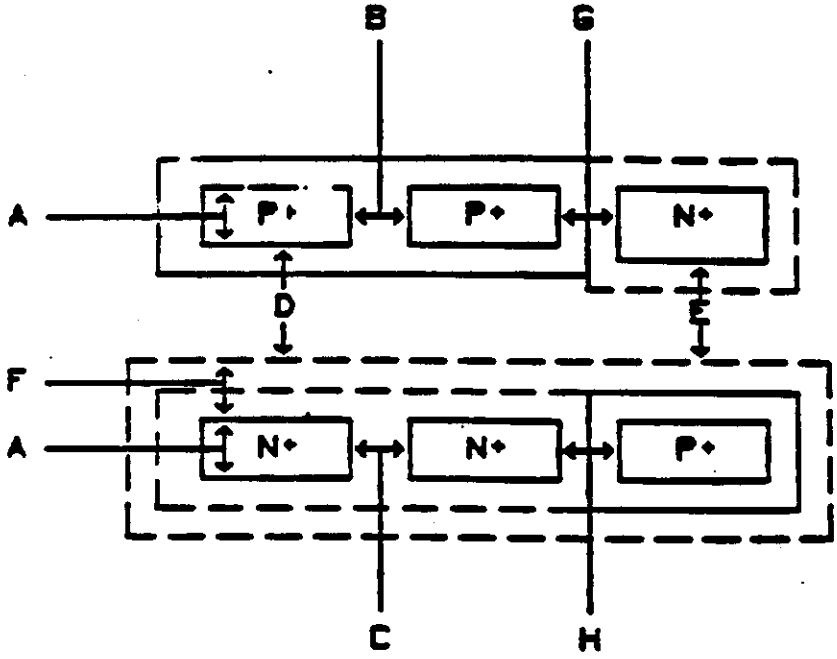
3.2 P+ Ring: (Optional - for radiation hardened applications only)

a. P+ ring overlap of P-well outside P-well	3	____
b. P+ ring overlap of P-well inside P-well	1	____
c. P+ ring width	4	____



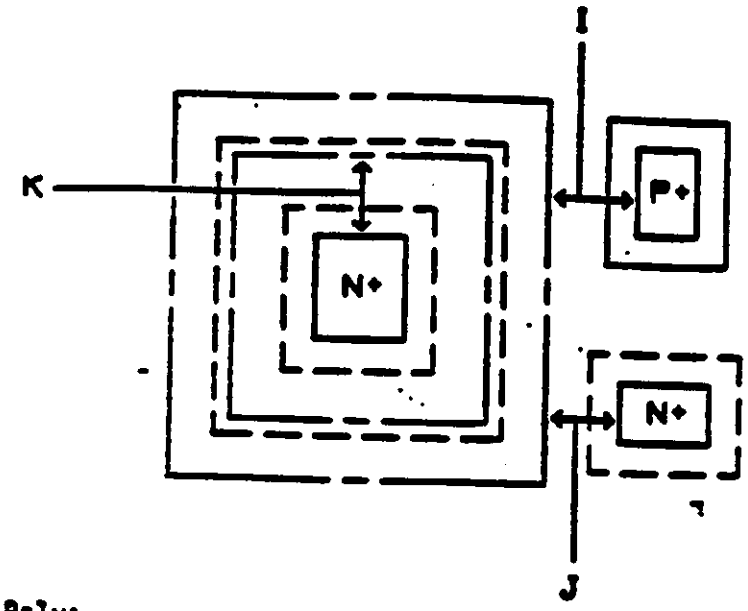
3.3 Active Area:

		Without P+ Ring	With P+ Ring
a.	Active area opening	4	(4)
b.	P+ active area to P+ active area spacing	4	(4)
c.	N+ active area to N+ active area spacing	4	(4)
d.	P+ active area in N-substrate to P-well edge spacing	8	(9)
e.	N+ active area in N-substrate to P-well edge spacing	7	(8)
f.	N+ active area in P-well to P-well edge spacing	4	(6.5)
g.	N+ active area to P+ active area spacing outside P-well	4	(4)
h.	N+ active area to P+ active area spacing inside well	4	(4)



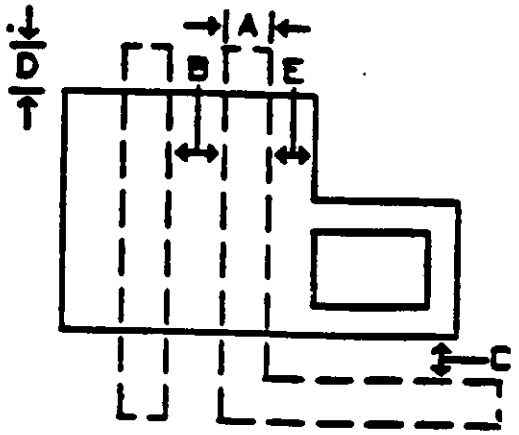
3.3 Active Area: (continued)

i.	P+ active area in N-substrate to P+ ring spacing	NA	(6)	
j.	N+ active area in N-substrate P+ ring spacing	NA	(5)	
k.	N+ active area in P-well to inside of P+ ring	NA	(5.5)	



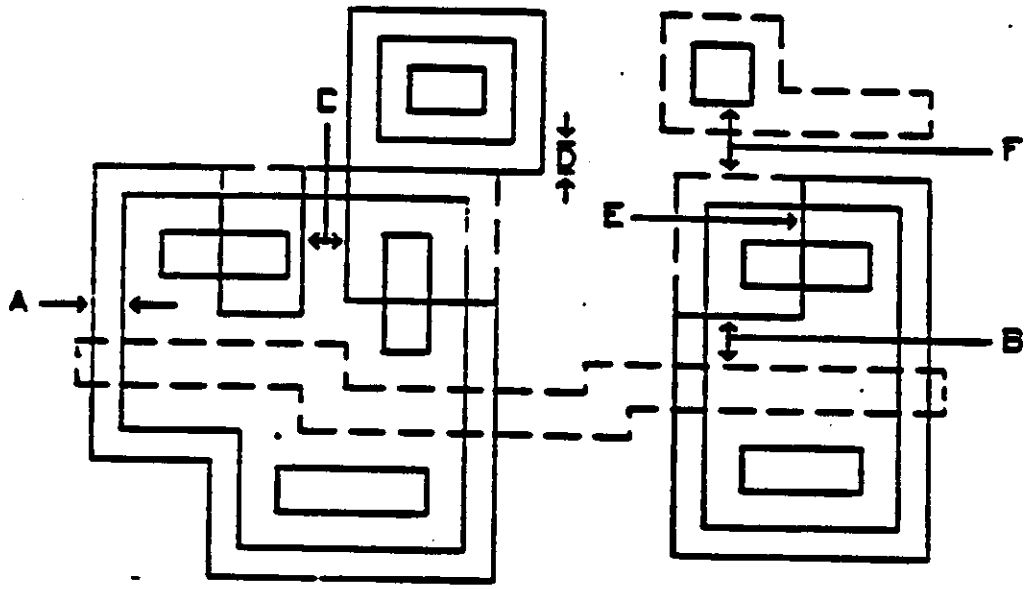
3.4 Poly:

a.	Poly width	3	
b.	Poly to poly spacing	3.0	
c.	Field poly to active area spacing	2.0	
d.	Poly gate extension over field	3	
e.	Gate poly to active area spacing	3	



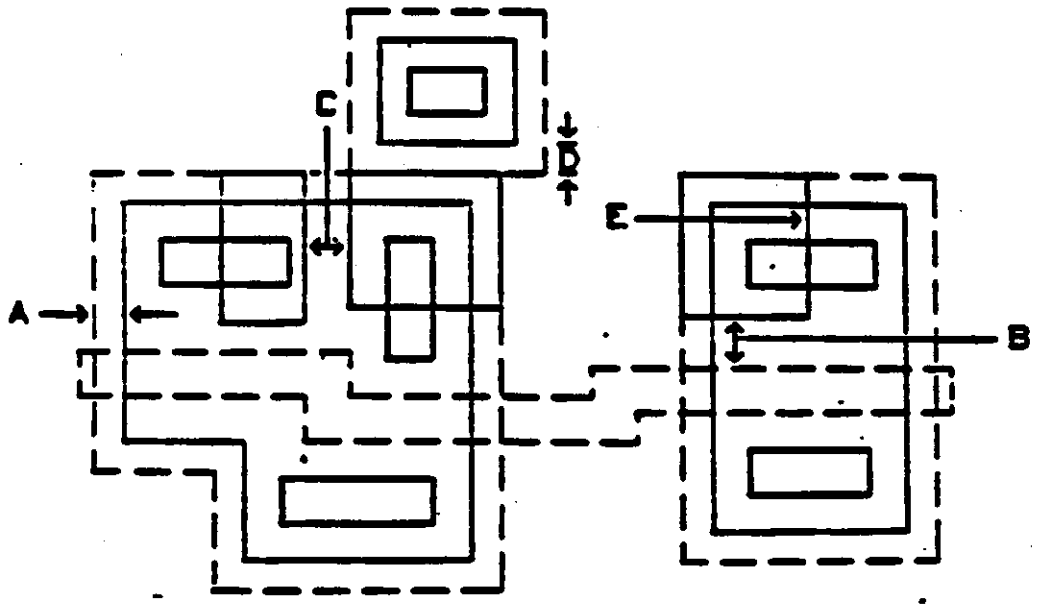
3.5 P+:

a.	P+ mask overlap of active area	2
b.	P+ mask overlap of poly in active area	3.5
c.	P+ mask to P+ mask spacing in active area	3
d.	P+ mask to N+ active area spacing (if P+ mask and N+ mask are coincident)	2
e.	P+ mask overlap of N+ mask to achieve shorting contact	0



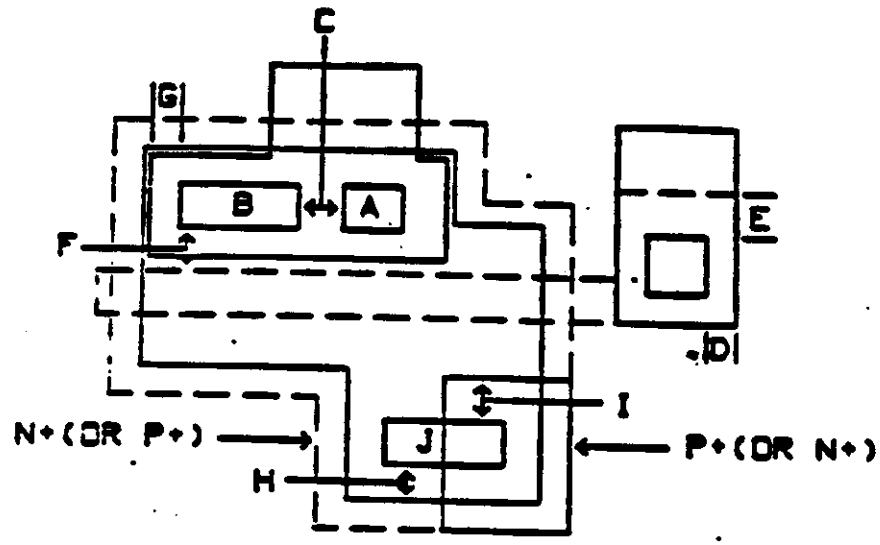
3.6 N+:

a.	N+ mask overlap of active area	2	
b.	N+ mask overlap of poly in active area	3.5	
c.	N+ mask to N+ mask spacing in active area	3	
d.	N+ mask to P+ active area spacing (if N+ mask and P+ mask are coincident)	2	
e.	N+ mask overlap of P+ mask to achieve shorting contact	0	



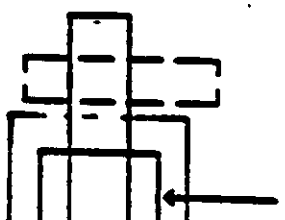
3.7 Contact:

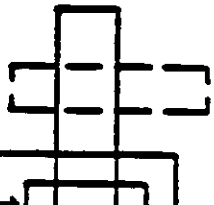
a.	Contact size	3 x 3
b.	Maximum contact size	3 x 8
c.	Contact to contact spacing	3
d.	Poly overlap of contact	2
e.	Poly overlap of contact in direction of metal	2.5
f.	Contact to poly channel spacing	3
g.	Metal overlap of contact	2
h.	Contact to active area spacing	2.0
i.	Contact to P+ and N+ mask spacing	3
j.	N+/P+ shorting contact size	3 x 8



If contact overlap of active area is permitted:

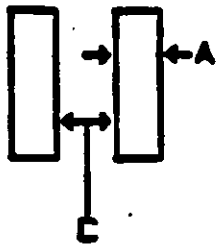
k.	Contact size to guarantee a 3 x 3μ contact area when contact edge is coincident with active area edge.	5.5 x 8
l.	Contact size to guarantee a 3 x 3μ contact area when contact edge is 2.5μ from active area edge.	3 x 8





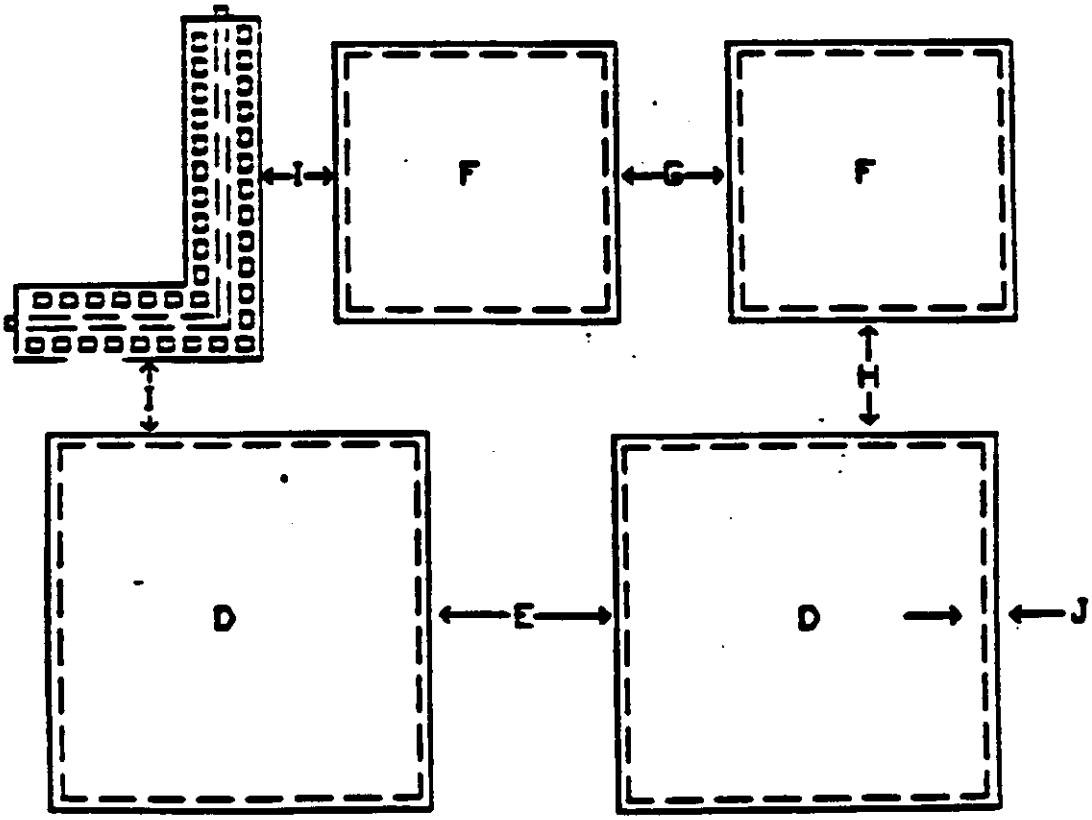
3.8 Metal:

a.	Metal width (interconnect)	3
b.	Buss metal current density (max)	0.7m A/μ
c.	Metal to metal spacing	4



SCALE: 2000X

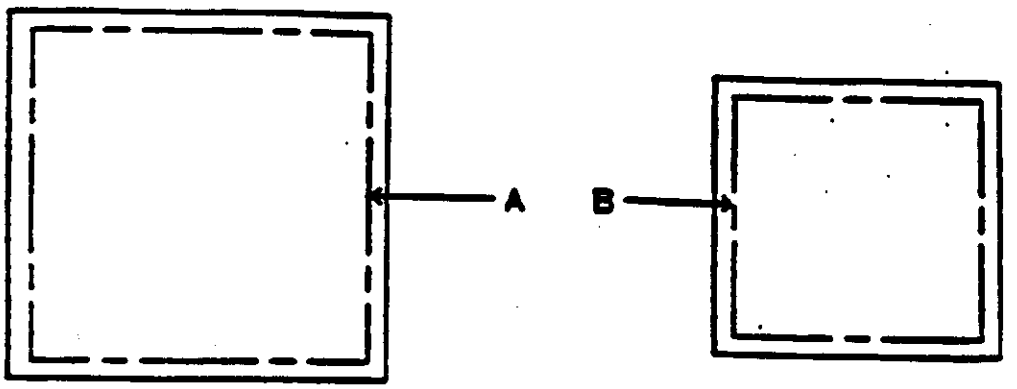
d.	Bonding pad metal area	100 x 100
e.	Bonding pad spacing (pad metal to pad metal)	100
f.	Probing pad metal area	75 x 75
g.	Probing pad spacing (pad metal to pad metal)	30
h.	Bonding pad to probing pad spacing	30
i.	Pad metal to circuit metal, active area, poly spacing	40
j.	Pad metal overlap of P-well	3
k.	Pad metal to scribe	50



SCALE: 500X

3.9 Passivation:

a. Bonding pad opening	90 x 90
b. Probing pad opening	65 x 65



SCALE: 500X

3.10 Please specify any additional design rules you require:

4. Electrical Parameters: In order to model device performance in a computer simulation program, it is necessary to know the electrical parameters of your silicon gate bulk CMOS process(es). In particular, we would like to know what your typical process spread is (i.e., what you would be willing to process to) for the parameters listed below. Please describe your test methods if they differ from those presented here. Also, if you have any internally generated documents on device modeling, we would like to review them.

To clarify how we measure threshold voltage and process constant, the following discussion is presented. Threshold voltage (V_t) and the process constant (K') are obtained from a low drain voltage ($V_{ds} \leq 50mV$) conductivity curve of drain current (I_{ds}) versus gate voltage (V_{gs}). In the linear portion of the curve (i.e., where V_{gs} has not yet started to affect mobility), the equation presented below describes the transfer characteristic of the curve.

$$I_{ds} = 2K' \frac{W_{eff}}{L_{eff}} ((V_{gs}-V_t) \cdot V_{ds})^*$$

Note that V_t is the extrapolated x-intercept ($I_{ds}=0$) and K' is determined from the slope of the curve.

*Reference - A. S. Grove, Physics and Technology of Semiconductor Devices, (New York: Wiley and Sons, 1967), p. 324.

4.1 The following is a list of worst case electrical parameters used in design specified at 25°C. (Revised electrical parameters)

a.	P-channel threshold voltage - max	-1.1	_____	volts
	- min	-.5	_____	volts
b.	N-channel threshold voltage - max	1.1	_____	volts
	- min	.5	_____	volts
c.	P-channel process constant - min ($K'p = \mu Co/2$)	6	_____	$\mu A/v^2$
	- max	12	_____	$\mu A/v^2$
d.	N-channel process constant - min ($K'n = \mu Co/2$)	15	_____	$\mu A/v^2$
	- max	30	_____	$\mu A/v^2$
e.	Gate oxide capacitance (600 Å)	5.7E4	_____	pf/cm^2
f.	Metal over substrate capacitance	5.2E3	_____	pf/cm^2
g.	Field poly over substrate capacitance	6.5E3	_____	pf/cm^2
h.	Metal over field poly capacitance	1.22E4	_____	pf/cm^2
i.	N+/P- junction capacitance	6.0E4	_____	pf/cm^2
j.	N-/P+ junction capacitance	4.1E4	_____	pf/cm^2
k.	Lateral diffusion (source/drain)	.4 .25 (simulations)	_____	µm
l.	Maximum operating voltage	7	_____	volts
m.	P+ Sheet Rho		_____	Ω/
n.	N+ Sheet Rho		_____	Ω/
o.	N+ Poly Sheet Rho	30	_____	Ω/
p.	N- Poly Sheet Rho	35	_____	Ω/
q.	P- Sheet Rho		_____	Ω/

5. The following tolerances, doping concentrations, and junction depths are assumed.

a. Photolithographic Dimensional Tolerances

1) Photomask (Except contact and Metal levels)	$\Delta P = \pm .5\mu m$ $\Delta P = \pm .25\mu m$	____ ____
2) Photoresist (Includes effects of over/under exposure and development)	$\Delta PR = \pm .25\mu m$	____
3) Alignment accuracy between 2 levels	$\Delta A = \pm .5\mu m$	____
4) Positive photoresist shall be used		

b. Etching Dimensional Tolerances (ΔE)

1) P-well level	$\Delta E = \pm 1\mu m$	____
2) P+ ring level, outside well inside well	$\Delta E = \pm 1\mu m$ $\Delta E = \pm 2\mu m$	____ ____
3) Active area level	$\Delta E = 0$	____
4) Poly level	$\Delta E = \pm .5\mu$	____
5) P+ level	$\Delta E = 0$	____
6) N+ level	$\Delta E = 0$	____
7) Contact level	$\Delta E = \pm .5\mu$	____
8) Metal level	$\Delta E = 0$	____
9) Passivation level	$\Delta E = \pm .5\mu$	____

c. Doping Concentrations

1) Starting material	0.60cm(1×10^{16} cm^{-3})N-type	____
2) P-well concentration	5×10^{16} cm^{-3}	____

d. Junction Depths

1) P-well	$X_j = 3\mu$	____
2) Source/drain	$X_j = .5\mu$	____

NAME

cformat – change format for different programs

SYNOPSIS

cformat [-s] [-m] infile outfile

DESCRIPTION

With the -s option cformat generates, from a .spice file produced by sim2spice, a second file which is spice compatible (believe it or not).

With the -m option cformat generates, from a .sim file produced by mextra (cmextra), a .sim file which is nl/rnl compatible (after presim).

FILES

/usr/cmos/src/cformat.c
/usr/cmos/bin/cformat

EXAMPLE

cformat -s myfile.spice myfile.sp

cformat -m myfile.sim myfile.simnl

BUGS

cformat does not work with .sim files generated by mextra with the -o option.

HISTORY

14-Oct-83 CMOS (cmos) at Carnegie-Mellon University
Created.

NAME

cnet – like "net" but with CMOS built-in capabilities.

SYNOPSIS

cnet -or-
cnet infile -or-
cnet infile outfile

DESCRIPTION

cnet is a net program that can directly handle cmos static gates.
The following circuits are available:

- cinvert : cmos inverter
- cnand : cmos nand gate
- cnor : cmos nor gate
- cxmit : cmos transmission gate
- bcnand : buffered cmos nand gate
- bcnor : buffered cmos nor gate

Format.
w is transistor width and l is transistor length. Both w and l are always optional; default is always w = 4.00, l = 3.00 (i.e. the minimum feature size of the MOSIS-CMOS technology).

(cinvert out (in w l))
(cnand out (in1 w1 l1) (in2 w2 l2) ... (in_nth w_nth l_nth))
(cnor out (in1 w1 l1) (in2 w2 l2) ... (in_nth w_nth l_nth))
(cxmit out in (e_gate w_e l_e) (p_gate w_p l_p))
(bcnand out (A wA lA) (B wB lB) (in1 w1 l1) ... (in_nth w_nth l_nth))
(bcnor out (AwA lA) (B wB lB) (in1 w1 l1) ... (in_nth w_nth l_nth))

Where, in cxmit:
out = the output of the transmission gate
in = the input of the transmission gate
e_gate = the n-channel transistor gate node
p_gate = the p-channel transistor gate node

Where, in bcnand, bcnor:
out = the final output of the buffered gate
B = output of the nand/nor and input of the first inverter
A = output of the first inverter and input of the second inverter
in1 ... in_nth = the actual inputs of the buffered gate.

FILES

/usr/cmos/src/cnet.c
(which substitutes /usr/vlsi/mitsim/net.c)
/usr/mxa/cmos/bin/cnet

ENVIRONMENT

PATH: /usr/mxa/cmos/bin
MPATH: /usr/mxa/cmos/man

SEE ALSO

User's Guide to NET, PRESIM, and RNL/NL by C.J. Terman.
MIT VLSI Memo No. 82-112 - July 1982

EXAMPLE

Three-input buffered nand gate. in1, in2, in3 are the three inputs. out_nand is the output of the nand. first_not_out is the output of the first inverter. final_out is the output of the buffered gate. The nand gate has w = 4 and l = 2 transistors. The first inverter has a w = 8 and l = 2 transistor. The second and last inverter has a w = 16, l = 2 transistor.

```
(bcnand final_out (first_not_out 16 2) (out_nand 8 2)          (in1 4 2) (in2 4 2)
(in3 4 2))
```

DIAGNOSTICS

Same as in net

BUGS

If dimensions are assigned to nodes that are supposed to be dimensionless, the program enters an endless loop.

Example: (cinvert (out 2 4) in) WRONG!
(out in cinvert is dimensionless. See DESCRIPTION: Format)

P-channel transistors always have the dimension of the corresponding N-channel. Presently, the fabrication process is extremely unstable and it is impossible to decide a fix holes/electrons mobility ratio. Therefore, if the ratio is different from 1, you'd better define your own macro.

The modifications have not been thought for analog applications. It does not deal either with clocked gates or domino logic (meaning: you have to build your own macro).

If you use cnet for nMOS, keep in mind that the default dimension of the enhancement transistor is 4 x 3.

HISTORY

16-Jan-84 CMOS (cmos) at Carnegie-Mellon University
Modified the default minimum feature size. Now it is 4 x 3 and is MOSIS-CMOS compatible, therefore.

06-Oct-83 CMOS (cmos) at Carnegie-Mellon University
Created.

NAME

cmextra -- mextra for CMOS (both Bulk and SOS)

SYNOPSIS

Same as in mextra.

DESCRIPTION

cmextra handles both CMOS Bulk and CMOS-SOS. cmextra can be used in the same way mextra is used.

As far as CMOS-SOS, keep attention to the names of the layers, that must be compatible with the MOSIS naming scheme.

cmextra does not support, for CMOS-SOS, both P+ and N+ masks, but P+ mask only. This is standard MOSIS.

cmextra decides the technology accordingly to the .cadrc file in your home directory. To use CMOS P-well, just put in the .cadrc file:

tech cmos-pw

or, for CMOS-SOS:

tech cmos-sos

Default is NMOS.

FILES

/usr/cmos/include/archiv/*.h
/usr/cmos/include/cmextra/*.h

/usr/cmos/lib/cmextra/*.lib

/usr/cmos/src/cmextra/*.c
/usr/cmos/src/archiv/*.c

/usr/cmos/bin/cmextra
/usr/cmos/bin/extname

SEE ALSO

The mextra man entry

DIAGNOSTICS

As in mextra.

BUGS

CMOS-SOS not extensively tested yet

HISTORY

18-Oct-83 CMOS (cmos) at Carnegie-Mellon University
Created.

NAME

plyra – p2lyra : design rule checkers for cbpe2 and cbpem2 processes

SYNOPSIS

plyra filename.ca

p2lyra filename.ca

DESCRIPTION

plyra and p2lyra are design rule checkers that handle the two CMOS-Bulk fabrication processes presently offered by MOSIS.

plyra copes with cbpe2 (CMOS-Bulk, capacitor electrode).

p2lyra copes with cbpem2 (CMOS-Bulk, capacitor electrode, two metal layers, two contacts).

Both plyra and p2lyra are lyra programs with embedded design rules for the proper process.

FILES

The design rules for plyra are in:

/usr/cmos/lyra/pwell/cmos-pwMOSIS.r

The design rules for p2lyra are in:

/usr/cmos/lyra/pwell/cmos-pwm2MOS.r

BUGS

They do not check design rules related to pad or scribe.

They do not check MAXIMUM dimensions. Therefore, they cannot check the maximum dimension of a contact-cut.

HISTORY

16-Jan-84 CMOS (cmos) at Carnegie-Mellon University
Created.