A METHODOLOGY FOR IMPROVED VERIFICATION OF VLSI DESIGNS WITHOUT LOSS OF AREA

Louis K. Scheffer
Departments of Electrical Engineering
and Computer Science
Stanford University
Stanford, California 94305

ABSTRACT

This paper describes an IC layout methodology based on arbitrary outline cells, prevention of overlap, and mixed programs and graphics. Advantages are: no loss in area over hand packing; incremental checking of design rules, component interconnection, and timing; reduction of visible complexity; and easy implementation. Disadvantages are: possible proliferation of cell types and poor handling of cells with contacts not on the boundary. An implementation that uses and enforces this methodology is discussed.

INTRODUCTION

IC design methodologies in use today have a number of serious defects with respect to design verification. They defer design rule checks (DRC) and electrical continuity tests until the end of the design cycle. This is necessary since the possibility of another item overlaying an item that has already been checked cannot be ruled out. However, at the end of the design cycle, errors that could have been fixed quite simply earlier may require extensive revision. Multiple errors may require multiple analyses of the entire chip to find all the errors. The classic example of this is the power ground short. If extracting a component list from the artwork reveals a power to ground short, then the list will be useless for other purposes such as simulation or error checking. It is necessary to fix the short, and rerun the component extraction program on the entire chip. Furthermore, DRC and component extraction require execution times ranging from NlogN to N**3/2, making these tests time consuming and expensive as chips increase in size.

In this paper, a methodology is introduced that depends on not allowing cells to overlap other cells or geometrical primitives, but allows cells to have arbitrary shapes. The advantages and disadvantages of this methodology

are discussed, and comparisons made between this methodology and others that might be adopted to solve the same problems. A discussion of how the problems of DRC and component extraction are treated in this system follows, along with a proof that shows that this methodology does not cost any area. Mext there is a discussion of the relationship between this methodology and schematic drawing systems, and a section on an implementation of an IC layout editor that uses and enforces this methodology. Finally, there is a section on possible future developments using this methodology.

BASIC IDEA OF THE, bIETKODOLOGY

The theme of this methodology is simply to avoid overlap. Towards this end, cells may not overlap each other, and cells may not overlap geometrical primitives, such as rectangles and polygons. The only objects that may overlap are geometrical primitives within the same cell. Cells, however, may be of arbitrary polygonal shape. To replace the capability lost when overlapping cells are outlawed, a mechanism is supplied to mix layouts and programs to define devices such as ROMs and PLAs. In addition there is a requirement that no active component area touch the boundary of a cell.

An analogy exists between this methodology and that of structured programming. In both cases, the intent is reduce the interaction of disparate parts of the system, so that each part may be designed, tested and understood by itself, independent of the remainder of the design. Both methodologies attempt to hide the details of the implementation; in programming the essential information is contained in the calling sequence, whereas in IC layout it is contained in the boundary of the cell. Not allowing overlap is similiar to not allowing GOTOs into or out of procedures. Not allowing components to touch the boundary of a cell is similiar to not allowing statements to be split over procedure boundaries.

This analogy explains why verification is so much easier if overlap is forbidden. Since there is no equivalent of global variables in IC design, each piece may be considered on its own, just as a procedure that has no global variables may be checked on its own. Furthermore, a designer (or a program) may use a cell without knowing anything about it except the behavior as expressed from the terminals.

One final analogy is that when GOTOs are eliminated, a richer variety of control structures is required if efficiency is not to be sacrificed. Similiarly, in IC design, if overlap is not allowed then cells must be allowed to assume arbitrary shapes.

ADVANTAGES AND DISADVANTAGES OF THIS METHODOLOGY

There are several advantages to this choice of methodology. First, it does not cost any area over a hand layout (proof follows). Second, it allows a completely hierarchical design, where only two adjacent levels of the hierarchy need to be examined at any time. This reduces the complexity as seen by the user and any analysis programs. Third, many checks on the validity of the data may be made incrementally, as each cell is defined and used. These include DRC, component extraction, and timing verification (if

the technique of assertions is used). Many errors can therefore be caught at an early stage, where correction is still easy, instead of at the end of the design process where correction is quite difficult and costly. Fourth, most processes that must be applied to the entire chip, such as the generation of new layers through logical operations on the existing layers, may be done by processing the chip on a cell by cell basis. This results in large savings when cells are used more than once.

Another advantage of this methodology is that it makes minimal demands on computer resources. This translates into low cost per station, either with a large computer timeshared between several stations or with a small computer per user. Since all processing is done one cell at a time, large amounts of memory are not required in the processor that does the DRC and component extraction. By using assertions, the work necessary to verify performance can also be vastly reduced.

Maturally, there are disadvantages as well. One serious objection is that designers are not used to designing with a total lack of overlaps. If the advantages described above are not attractive enough, the designers will be reticent to use a system that takes away some of their former freedom. Another objection is that this methodology is optimized for cells with their connections on the edges. If cells have terminals in the middle then this methodology will require a large number of nearly identical cells, with the only differences being the wiring required to bring the internal terminals to the edge. A third problem may appear if an operation such as oversizing or undersizing a mask is attempted. In this case the modifications within a cell may depend on the surroundings. Thus these modifications cannot be done on a strictly cell by cell basis, and it may be necessary to accept a proliferation of nearly identical cells in order to perform these operations and still keep the results in the strictly hierarchical format. These problems should not be serious for highly structured chips, but may present real barriers to using this methodology in other cases. For example, it is very ill-suited to masterslice type construction.

There are other problems that this methodology does not help, but does not hurt either. These are primarily calculations of a global nature that cannot be completed until the entire circuit is known. The general problem of calculation of the resistance of interconnections and the resulting time delays is one of these problems. Since the problem is inherently global, unlike capacitance (which is the sum of the local capacitances), the entire net must be known before the resistance or time delay may be calculated. Therefore this cannot be done at cell definition time, but must wait until the cell containing the entire net is constructed.

Penfield and Rubinstein [Pe81] have recently demonstrated a method which bounds the time delay for tree structured networks. This case can be solved on a local basis, and hence fits very well with the proposed methodology. The only test that would still have to be made globally is to insure that the interconnections are indeed tree structured, since this cannot be determined from local data.

COMPARISON WITH OTHER METHODOLOGIES

There are many other methodologies that may be used to create a chip. How does this methodology compare? Three strategies that have been considered are: allow any overlaps and do all checking at the end of the design process, allow overlaps but account for them correctly, and do not allow overlaps and use rectangular bounding boxes.

Currently, the strategy used by most artwork systems is to allow the designer to create any overlap desired. Design rule checks, and tests for logical correctness, are performed once the design has been completed and the entire design expanded out. This approach gives the designer the maximum amount of freedom, and makes editing very simple, since anything is allowed. However, it requires that the checking programs must run on huge amounts of data. In particular, repeated cells are checked as many times as they occur. As chips get bigger, the volume of data will only get larger. Furthermore, design errors are found at the end of the design cycle, when they are the most difficult to fix. Errors in repeated cells show up many times, making it hard to find errors that may only have occurred once. One error, such as shorting power and ground, may make it impossible to find others without repeating the entire cycle.

Another methodology, tried by Whitney [Wh80] for DRC, is to allow the designer to overlap items wherever desired. The resulting design is checked one cell at a time, but the design rule checking program detects and accounts for these overlaps in the checking process. This approach also puts no restrictions on the designer, and allows the traditional methods of programming PLAs and ROMs. Redundant errors are also reduced, since each cell is checked only once except where it overlaps others. The drawbacks are that checking must still be done on the whole design at the end of the cycle, and that accounting for the overlaps may take more computational effort than simply checking the entire design as one piece. Furthermore, this approach is not well suited to incremental checking, since there is no guarantee that any portion of the design is complete until the entire design is finished. This approach probably represents the best that can be done without imposing any restrictions whatsoever on the designer.

McGrath and Whitney [McG80] propose a methodology that allows cells to overlap, but requires that each cell be correct when checked by itself, and all active devices to be represented by specially marked cells. The requirement that each cell be correct by itself leads to many false errors in the cases where a half width feature on a boundary is mated with a similar feature on the other side of the boundary. The suggested solution is to include full width features, and avoid the loss of area by overlapping the two full width edges. This means that overlap must be permitted, which leads to the problems in DRC discussed under the last methodology. Component extraction must also take the overlap into account, at least for capacitance calculations. Isolating active devices in cells of their own is almost a necessity in bipolar technology, where a given geometry could be a resistor, a transistor, or a zener depending on how it is biased, but is a quite significant (and unnecessary) restriction on the designer in normal MOS design. The transistors may be recognized quite easily, and unintentional

transistors are caught when the layout is compared to the logic schematic.

A fourth methodology is followed by many automated layout programs such as CICLOPS [Pr79]. This is to allow no overlaps, and to require rectangular cell boundaries. This allows design and analysis to proceed on a cell by cell basis, but often wastes area since only rarely is it possible to combine several different sized rectangles into a resultant rectangle without any wasted area. Because of this obvious waste, human designers are reluctant to use this methodology, especially for high volume products.

NO LOSS OF AREA

It is straightforward to show that the proposed methodology result.; in no loss of area in any technology with only one active layer. Given any design, first expand out any hierarchy that may be present. At this point, the data is merely a large number of polygons with no included cells, and hence it meets all the requirements of the proposed methodology. Now iteratively perform the oll f procedure: cover the chip with two non-overlapping polygons in such a way that half of the devices are inside one polygon, and the other half of the devices are in the other. The polygon boundaries may not pass through any devices. Since the devices are topologically separate in a technology with one active layer, this division into two polygons can always be performed (although it may require using polygons with internal holes in pathological cases). Call each polygon that has been created this way a cell; then the chip is now represented as two cells, each of which meets the methodology requirements. These cells may be in turn subdivided further, until the entire chip is represented as a hierarchy, with each cell containing at most two other cells, or two devices.

It should be emphasized that although it is possible to automatically split a chip up this way for DRC purposes, it is far better to have people design hierarchically in the first place. If this is done, then no arbitrary splitting of any sort is necessary. Splitting a complete chip voids most of the benefits of the proposed methodology, and is only intended as a thought experiment to show that any design may be represented in this methodology without loss of area.

PERFORMING OPERATIONS IN THE PROPOSED METHODOLOGY

Design rule checking is done as follows in this methodology. Let MAXRULE be the maximum distance specified in any of the design rules. Define the OUTLINE of a cell as everything within MAXRULE of the boundary. Then, for each cell, substitute in the outlines of all the lower level cells, and perform all the DRC tests. Ignore all errors within the outline that could possibly be affected by whatever surrounds the cell, for it is not known whether these are errors until the cell is used. An error of this type is shown in figure 1, for the case of width violations. If the square C is a minimum size feature, then the rectangles A and B represent potential width violations. Rectangle A cannot be fixed by the addition of geometry outside the cell boundary, and hence represents a real error that must be reported. Rectangle B, however, could be fixed by the addition of another rectangle when the cell is used, and therefore it is not reported. If the additional

geometry is not added to a cell in which this cell is used, then the error will be reported at that time.

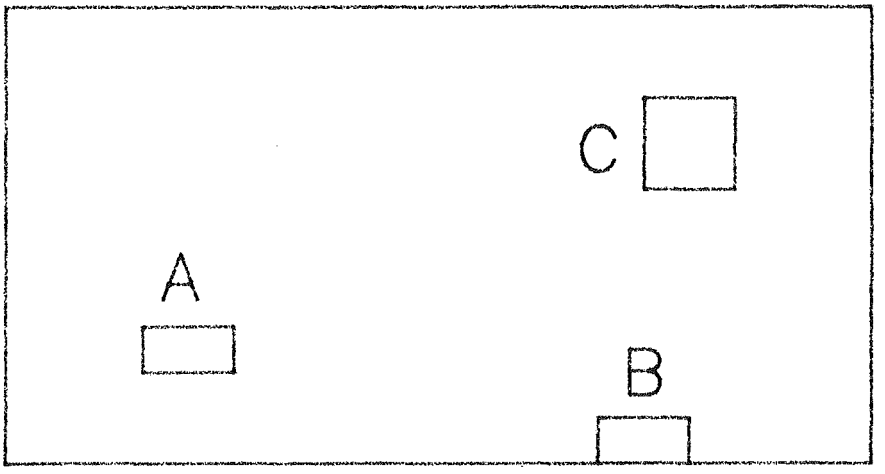

Figure 1: Real and potential errors

Similiarly, the design rule checking code must ignore all errors that could be fixed by something inside the inner boundary of an included cell. If the error is real, it would have been caught when the cell was checked. This step is quite important, since the process of saving everything within MAXRULE of the boundary may split internal items into pieces that do not meet the design rules. Figure 2 shows the portion of a cell that is retained and substituted in whereever the cell is used. Since only the geometry within MAXRULE of the edges is retained, the polygon at A has been split, creating a possible error. The design rule checker, however, should not report this error, since it is against an inner boundary, and if it was a real error then it would have been caught when the cell was checked.

Any remaining errors are valid, since nothing can be added within MAXRULE of them, and should be displayed. The final step of the design rule checking process saves the outline, which will be substituted for the cell whenever the cell is to be involved in DRC checks.

This approach handles split contacts and half-width lines on the edges of cells without special cases. Furthermore, all of the rules, including the complicated anti-reflection rules, may be checked in this manner. The designer therefore gets immediate feedback as each cell is created or modified. This is important since errors are much harder to correct later on. The ability to check all rules is in contrast to some interactive design rule checkers that check for validity while the editing is in progress. These give even more immediate feedback, but cannot check some of the more difficult rules. The proposed methodology will never create false errors, but errors in the outline of the cell may not be caught until the cell is used, and may appear as many times as the cell is used.

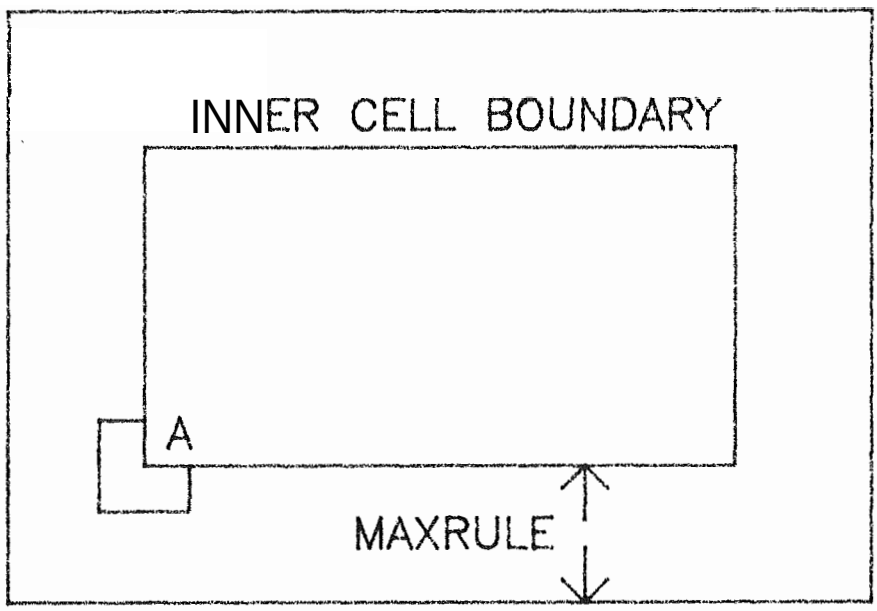

Figure 2– Possible internal error

Component extraction is made simple by the fact that no active areas may touch the boundary. This means each device is entirely contained within a cell, and only interconnections touch the edges. Implied devices, such as transistors formed where polysilicon overlaps diffusion, are not a problem since we know nothing will overlay the cell under consideration. Thus each cell can be extracted separately and reduced to a boundary that shows the connection points. Capacitance (as a function of area) may be easily calculated since the area of a node is the sum of the area in each of the cells that use that node (since there are no overlaps). Sidewall capacitance may also be computed, with the proviso that sidewall capacitance on the cell boundary cannot be computed until the cell is used. Similiarly, node to node capacitance may be computed, at least for nodes lying within MAXRULE of each other. (MAXRULE could be increased if this proves to be a restriction.)

Enforcement of this methodology is quite easy. Check to see that no cells overlap, check to see that no primitives overlap any cells, and check to see that no components extend past the boundary, if the boundary is specified by the user.

Finally, some mechanism is needed to allow the operations that formerly were done by overlapping cells, such as filling out the bits in a ROM or PLA. One approach to this problem is to allow programs access to the graphics primitives, so a program can write a bit pattern to be included in a ROM or PLA. The program does not get any special privileges such as the ability to overlap cells. The output is subjected to exactly the same tests as if it had been entered by a human. This is useful for catching errors in the programs.

RELATION TO SCHEMATIC DRAWING SYSTEMS

In many ways, the proposed methodology resembles the methodology used in schematic drawing systems. This is not a coincidence, since every attempt was made to incorporate the features that have made hierarchical schematic drawing systems useful. In a schematic, the user never views more than two levels of the hierarchy at a time. Any arbitrary structure can be represented, although some fit more naturally into the hierarchical structure than others. Individual pieces nay be checked with the assurance that the rest of the design will not affect the piece that is being checked, except through the terminals of the device. These properties all carry over to layout, if it is done with this methodology.

In particular, two forms of checking originally designed for schematics are now possible for layouts. One is the timing check by means of assertions, as used by McWilliams [McW80]. In this technique the designer supplies statements about the signals which are believed to be true (the assertions). For example, consider the case of a simple cell with one input and one output. One assertion might state that the input must be true 50 NS after the clock, and another assertion might state that the output will be stable and correct by 90 NS after the clock. With the aid of these assertions, a cell may be checked without looking at the contents of any other cells.

In order to check a cell by this technique, the timing verifier assumes that the input assertions are correct, and attempts to prove the output assertions from the inputs and the properties of the devices contained in the cell. If any other cells are included in the cell under consideration, the verifier checks to be sure that their input assertions are net, and assumes that their output assertions true. By this means, an entire design may be verified one cell at a time, and timing verification may he done on the pieces long before the design is complete.

A second check that is easy to make using the proposed methodology is a conparison between the logic schematic of a cell and its layout. In the simplest form, the checking program could require that the hierarchy be identical in the schematic and layout representations. Then checking is quite simple. In more complicated forms, a theorem prover may be called into play when the correspondence is not exact, to see if the two different representations perform the same function. In either case, the check may be performed on a cell by cell basis.

There are still fundamental differences between a schematics and layouts, however. In a schematic, the inside and outside views are distinct. An op-amp, for example, is composed of a large number of transistors, but the configuration of these transistors bears no relation to the triangular symbol that represents the op-amp. Furthermore, the size and shape of the symbol do not depend on the actual complexity of the opamp, so the designer can design and wire an active filter, for example, without knowing the gain bandwidth products of the op-amps. These may be filled in later without disturbing the design.

In a layout system, such freedom does not exist. The outside view of a device must resemble the internal view in its size and the location of connections. The designer cannot wire up an inverter, for example, without knowing the load the inverter is to drive, and hence its size. If at some later date a larger, more powerful inverter must be substituted, then the wiring must be changed. This additional complexity does not plague the designer who works with schematics.

Schematics also allow several shortcuts for indicating connectivity that have no equivalent in the world of IC design. Global signals may be defined whose values are known to all cells. Within a diagram, pins may be connected together simply by giving them the same name. In a layout, on the other hand, all connections of any type must be shown explicitly.

AN IMPLEMENTATION

A program has been developed to test this methodology. Component extraction, boundaries, assertions, methodology enforcement, and programmatic access have been implemented, but DRC has not. The system is written in PASCAL and runs on a DEC-20 with an HP-2648 graphics terminal and a Summagraphics BitPad One graphics tablet. PASCAL is also used as the imbedded language in which the user manipulates graphic constructs.

The program consists of two major sections, the editor and the compiler. The editor is the user interface, and allows the user to create, modify, and purge cells. Commands are similar to many other graphics editors, allowing the user to add components, interconnect them, move them and so forth. A minimal amount of checking is done at entry time for obvious errors, such as illegal signal names and lines at angles that the rest of the process cannot accept.

The editor and the compiler are adapted to various processes by means of a "process file". This file contains data on all the layers to be used in the compilation process. It is similar to the file normally used to describe DRC operations, with information about each layer, such as minimum spacing and width, capacitance per unit area, and the logical operations by which it is computed if it is not entered by the user.

The compiler is called whenever the user writes a cell out to permanent storage, thus making the cell available for use elsewhere. The compiler takes the user specifications of the geometry, fills out lines to their minimum width, and converts all input to polygons. The new layers are then computed, if necessary, and a connectivity analysis made. The DRC test is done at this point, so it can take advantage of the continuity information if it so desires. Then the boundary is computed (or the user specified boundary is checked) and every signal that touches the boundary is noted. The outline is then saved away so that it may be used by other cells. The date when the last change was made is recorded, so that when a cell is changed the modifications will propagate correctly.

The component extraction routines check for the obvious problems of multiply named signals, and signals of the same name that are not connected together. Running capacitance calculations are available for applications where performance is crucial. The user can ask for continuity to be shown; the system can take a signal hy name or location and display on the screen where the signal is connected. All of the masks that have been generated as part of the DRC or component extraction process may be displayed.

Poundaries for cells may be specified in one of three ways. If nothing else is specified, then the boundary is computed as the logical OR of all the layers of the cell.. Active areas are expanded by one minimum unit before the OR to ensure that no active area hits the boundary. This gives the minimun possible size for the cell, but the boundary is often far more complicated than necessary, and presents a cluttered appearance. It is never wrong, however, and is the default when data is transferred from another system that does not use the same methodology.

Boundaries may also be the minimum bounding boxes. This is consistent with several other artwork systems, and works well when the resulting cells are to he fed to an auto-router (since current allto-routers cannot handle arbitrary shape cells well). Finally, the user can specify the boundary by entering it as a polygon on a reserved mask. This usuallyyields the best results, but requires the most work.

Assertions appear to the user simply as notes with a reserved first character. Like signal names, they attach to the closest object. The compiler makes no attempt to process the assertions, but records them, along with the signals they are attached to, in the dictionary of signal names. This dictionary is available as a file, and any checking program that desires to do so may check it, and use the assertions as it sees fit. This is desirable since people will doubtless think up new ways to use assertions. Tn MOS design, for example, it may be necessary to include assertions about capacitances to be driven and voltage levels of various signals, as well as assertions about the signal. timing.

The user tells the compiler to include a program along with a diagram by dividing the diagrams into FRAMES. Each frame has a name, and may contain primitives or text representing a program. If frames are present, and one is labelled PPOGRAM, then the program in that frame is executed before any other framesare evaluated. The program may direct the evaluation of other frames, add graphics primitives to a diagram, and make decisions about the parameters of diagrams.

There are several restrictions imposed by this implementation. Although component extraction has been implemented for MOS transistors and parasitic capacitances, no DRC has been implemented yet. This should be a relatively simple project, since it only involves running a nearly conventional DRC on portions of the artwork that are selected by the compiler. Lines at angles other that *0* and 90 degrees are not supported, but this should be a straightforward extension. Although the simple logical operations such as AND, OR, and ANDNOT are implemented, the more complicated operations of OVERSIZE and UMDERSIZE are only implemented in a form that is not always

correct at the cell boundaries. If a cell is used in different surroundings, the user might prefer to crsate additional cells in order to do the geometrical operations properly and keep the results in the strictly no overlap format. At present this option is not available.

A POSSIBLE EXTENSION

An extension that allows more designer freedom at the cost of additional complexity has been proposed by Horowitz [Ho80]. This is to store a boundary for each layer instead of one boundary for the whole cell. This allows cells with internal terminals to be routed, and routing to be run over cells that. were not specifically designed for this type of routing, if there is room. However, care must be exercised or else unintentional devices may be formed. Capacitance calculations and DRC checking also become more difficult, hut the basic hierarchical structure of the solutions remain unchanged.

ACKNOWLEDGEMENTS

I would like to thank the design aids group at Hewlett-Packard for supporting this work, and my colleagues at Stanford University for many fruitful discussions. Martin Newell suggested including the analogy to structured programming.

REFERENCES

[Pr79] Preas, Bryan T., "Placement and Routing Algorithms for Hierarchical Integrated Circuit Design", Technical Report #180, Computer Systems Lab, Stanford University, Stanford, CA. (August 1979)

[McG80] McGrath, Edward J, and Whitney, Telle, "Design Integrity and Immunity Checking", Proceedings of the Seventeenth Design Automation Conference, Minneapolis, Minnesota, June 1980

[Wh80] Whitney, Telle, "Description of the Hierarchical Design Rule Filter", SSP File #4027, Silicon Structures Project, California Institute of Technology, Pasadena CA. (October 1980)

[McW80] McWilliams, Thomas M., "Verification of Timing Constraints on Large Digital Systems", Proceedings of the Seventeenth Design Automation Conference, Minneapolis, Minnesota, June 1980

[Ho80] Horowitz, Mark. Private communication

[Pe81] Penfield, Paul, and Rubinstein, Jorge, "Signal Delay in RC Networks", Caltech Conference on Very Large Scale Integration, January 1981.