Wiring Considerations in Analog VLSI Systems, with Application to Field-Programmable Networks

Thesis by
Massimo Antonio Sivilotti

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California USA
1991

(Defended July 17, 1990)

Acknowledgments

I am greatly indebted to Carver Mead, whose unfailing support and guidance through the long years made this work possible. His insight and encouragement were critical components in the process of my learning to do research. The environment that he created for his students was rich and fertile; in leading by example, Carver is a role model for all of us.

I would also like to thank my committee, Professors Yaser Abu-Mostafa, Al Barr, Sandeep Bhatt, Alain Martin, and Carver Mead, for their helpful comments and patient review of this thesis. Their office doors were always open to me, despite pressing commitments and demanding students of their own. It was an honor, and a privilege, to work with these gentlemen.

I owe much gratitude to my colleagues and fellow students in Carver's research group. In particular, the early work on implementations of neural networks was done in conjunction with Michael Emerling. The later work on self-timed retinal architectures was a collaboration with Misha Mahowald. In the intervening years, countless discussions with Steve DeWeerth, Michael, Tor Lande, Mary Ann Maher, Misha, John Tanner and John Wawryznek helped shape my research. Thanks also to the countless people who braved my design tools; even though I was not always as sympathetic as I should have been, their suggestions made it possible for all of us to get real chips done.

I owe what sanity I have today to a fabulous set of friends. Biking, hiking, camping, cooking, "Friday Lunching," WallyBall, indeed... Nan, Andy, SteveB, SteveD, Michael, JohnT, JohnW... what can I say? And what can I say about Ruth, other than, without her constant love and understanding, I would not be writing these words today?

Finally, I must acknowledge the moral support of my family, without which the long journey to Southern California would have been much more difficult.

This research was supported by the System Development Foundation and the Office of Naval Research, and by equipment grants from Hewlett-Packard. Integrated circuit fabrication was generously provided by MOSIS. I was personally supported by NSERC and IBM, for which I am grateful.

Abstract

This thesis develops a theoretical model for the wiring complexity of wide classes of systems, relating the degree of connectivity of a circuit to the dimensionality of its interconnect technology. This model is used to design an efficient, hierarchical interconnection network capable of accommodating large classes of circuits. Predesigned circuit elements can be incorporated into this hierarchy, permitting semi-customization for particular classes of systems (e.g., photoreceptors included on vision chips). A polynomial-time programming algorithm for embedding the desired circuit graph onto the prefabricated routing resources is presented, and is implemented as part of a general design tool for specifying, manipulating and comparing circuit netlists.

This thesis presents a system intended to facilitate analog circuit design. At its core is a VLSI chip that is electrically configured in the field by selectively connecting predesigned elements to form a desired circuit, which is then tested electrically. The system may be considered a hardware accelerator for simulation, and its large capacity permits testing system ideas, which is impractical using current means. A fast-turnaround simulator permitting rapid conception and evaluation of circuit ideas is an invaluable aid to developing an understanding of system design in a VLSI context.

We have constructed systems using both reconfigurable interconnection switches and laser-programmed interconnect. Prototypes capable of synthesizing circuits consisting of over 1000 transistors have been constructed. The flexibility of the system has been demonstrated, and data from parametric tests have proven the validity of the approach.

Finally, this thesis presents several new circuits that have become key components in many analog VLSI systems. Fast, dense and provably *safe* one-phase latches and hierarchical arbiters are presented, as are a low-noise analog switch, an isotropic novelty filter, a dense, active high-resistance element, and a subthreshold differential amplifier with a large linear input range.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The inexorable demands for greater functionality, higher performance, reduced silicon area, and lower power consumption are constantly driving advanced VLSI designs toward more aggressive technologies. Analog networks hold great promise, as their explicit parallelism is an obvious match for VLSI circuits. A clear advantage of such networks is a natural scalability intrinsic in their computational structure. Another advantage is that analog systems are often in a better position to exploit innovations in devices, processes and structures, because: (1) analog circuits rely on the rich physical behavior of their components to attain their compactness and great breadth of computational functionality, and (2) analog circuit designers are accustomed to paying great care in interfacing components (great flexibility is possible in the absence of rigid abstraction and composition rules). So-called *neural* networks are currently an attractive research topic. Their great potential is evidenced by the existence of functional biological systems, able to solve complex tasks necessary to survive in a dynamic, ill-structured, and often hostile environment. From a systems perspective, the chief virtue of neural networks is their departure from a reliance on *programming* in the conventional algorithmic sense; rather, such networks are commonly *taught* to solve a problem, generally by observation of a sequence of input data that has been previously classified by an external teacher. From an implementation perspective, these networks are intrinsically parallel, and typically comprise large numbers of relatively simple individual components; these characteristics make them attractive to VLSI implementation.

Because the connectivity of these networks is often explicitly determined by the neural computation being performed (and the neural training algorithm being applied), a careful consideration of the implications of this connectivity is both appropriate and required at an early point in the design process. Also, as neuromorphic systems continue to increase in sophistication, the necessity for hierarchical processing operating on evolving signal representations will become an increasingly important research issue. In this thesis, we present the design of a system capable of accommodating medium-scale networks by exploiting a hierarchical interconnect strategy; this system provides a prototyping platform for testing such hierarchical networks.

Aggressive designs require sophisticated and reliable design tools. Unfortunately, automated design aids, long considered barely adequate even for traditional digital VLSI applications, are woefully insufficient for large-scale analog systems. In particular, conventional software simulators are too slow, have limited capacities, are inaccurate (the device models are usually for discrete, and not integrated, devices and structures) and inextensible (it is usually difficult to improve the device models, or to add new ones) and, in general, constitute an unfriendly design environment. Consequently, such simulators are inappropriate for testing ideas relating to intermediate-scale systems. For example, analog networks, collective systems, and adaptive control systems all require larger number of components than can be reasonably accommodated in present circuit simulators. Also, large classes of systems exhibit highly data-dependent operation, and require simulation of large numbers of test cases. Even the advent of parallel circuit simulators will not solve the design bottleneck, as the issues of accurately modeling device behavior over wide ranges, and of modeling circuit parasitics, still remain.

The opportunity thus exists for a high-performance, high-capacity circuit simulator, with precision and flexibility as its principal objectives, and the capability to accommodate novel integrated structures. In this thesis, we propose such a system, built around a field-configurable network that interconnects actual physical devices.

1.1 Reader's road map

The first part of the thesis (Chapters 2, 3, and 4) is a discussion of the connectivity requirement of regular VLSI systems. We begin by describing an early system: the first VLSI implementation of a neural network. In particular, we discuss several implementations of Hopfield-style associative memories, and the consequences of their full interconnect with regard to the absolute size of viable implementations and the scalability of such networks.

We then consider one facet of the general graph embedding problem: by considering the dimensionality of the interconnection medium, we derive wiring limits that *determine* implementable systems. In particular, we propose a *requirement* for such systems: the components within the system should be approximately uniformly distributed spatially. From this requirement, we derive a wiring relation that is consistent with a commonly used empirical relation known as Rent's rule. Finally, we approach the wiring problem from the perspective of permissible wire length distributions; we use our wiring model to derive the average wire length for regularly arrayed cells, and relate this length to allowed physical area limits.

The second part of the thesis (Chapters 4 and 5) applies these connectivity constraints to designing an efficient, field-programmable interconnect structure intended for real-time prototyping of medium scale analog systems (a few thousand transistors). Various aspects of the design are discussed, including algorithms for embedding circuit graphs onto the hierarchical interconnect, various tradeoffs in the interconnect technology (e.g., volatility, area density, speed and ease of programming); examples of particular implementations are

presented. Test results quantifying the electrical performance of the system are presented.

The opportunity that exists for this system is to provide a fast, versatile environment for developing and testing analog circuits. To this end, in addition to considering the universality and functionality of the hierarchical interconnect network, we describe the software tools that provide utility functions such as netlist specification and manipulation, and provide a methodology for design validation in the form of testing for isomorphism between pairs of circuit graphs. These tools are available as part of the Caltech suite of VLSI CAD tools.

The third part of this thesis (Chapter 6) illustrates particular aspects of analog circuit design by presenting some previously unreported circuits that have become key components in several VLSI systems. Two of these, a single-clock-phase latch and a two-input asynchronous arbiter, are "digital" circuits whose correct operation is guaranteed by careful consideration of their internal analog components. Other circuits include a low-noise differentiator, built using a low clock-feedthrough analog switch, and an isotropic novelty filter that efficiently performs an $O(N^2)$ computation with only $O(N)$ components. System-level applications are also presented, including various asynchronous counters, and a self-timed retina design frame.

Chapter 2

A Novel Associative Memory Implemented Using Collective Computation[1]

Nullumst iam dictum quod non dictum sit prius.

Terrence (c. 190-159 B.C.), Eunuchus. Prolog. 41

2.1 Introduction

In nature, there exist many examples of systems consisting of large numbers of individually simple elements, which interact in simple ways, yet *collectively* yield highly complex behaviors. For example, soap molecules interact via local van der Waals forces, yet on a global scale produce strikingly robust *bubbles*, which minimize rather sophisticated energy functions due to surface tension. In biology, although neurons are not simple (and neither are the mechanisms of excitation/inhibition at synapses) a similar observation is made: the formidable macroscopic computing capability of a human brain must result from the highly interconnected nature of the neurons. In general, specific high-level functions cannot be attributed to a specific neuron, and furthermore it has been shown that the wiring "map" varies considerably between individuals of the same species [**?**].

[1]Portions of this chapter have been previously published. See [**?**] [**?**] [**?**].

Collective systems such as these generally display a number of intriguing properties. First, since no single element encapsulates the function of the system as a whole, they are very robust against the failure of a subset of elements. For example, in the human brain, about 0.1% of all cells die each year, and are not replaced (that's some 3 million per *day*); yet the functionality of the brain is not noticeably impaired. This fault tolerance is in direct contrast with our conventional design techniques for computers, where every element has a clear function, and failure of that element is, in general, catastrophic. For this reason, and for the obvious ability of collective systems to solve problems (e.g. vision, audition) that are difficult for conventional computational paradigms, the understanding of collective computations is desirable.

If computation is defined as the processing of information, there are two aspects which must be investigated: (1) the storage of information, with the associated issue of representation, and (2) retrieval of information, which may be considered a transformation between representations. We will describe a specific example of a collective computation, that addresses both of these aspects.

2.2 The Content-Addressable Memory

One of the simplest, and best understood, behaviors of collective systems is that of content-addressable memory (CAM), or associative memory. The model we use for this association is that of viewing the computation as an evolution in state space. The association consists of establishing an initial condition, then evolving to a final, stable state, which we call a *memory*. In some sense, this property can be considered *error correction*, as the initial partial information is increased until the state of the system (which represents some encoding of some piece of information) is fully determined.

A physical system can be used as a CAM if a specific set of states can be made to be stable, and if the dynamics of the system guarantee evolution into one of these states. As we shall see, an energy formulation shall prove useful to understanding the behavior of the system.

Conventional digital associative memories [Hayes, 1978] [Parhami, 1973] [Weems, 1982] have two characteristic shortcomings: (1) the association occurs on an a priori defined key, which must be supplied exactly, (2) resolution of multiple matches often demands an additional serial polling algorithm (slow). In addition, they only work on digital (restored) data, and thus are useless for recognition of patterns with grey scale. We have designed and tested a radically new genre of associative memory. In the ASSOCMEM, association takes place on the word as a whole; the result is the word stored in memory that is *closest* to the supplied word. There is no distinction made between *key* and *data*. The convergence of the association process is guaranteed by theory. The length of time required to perform the association depends, in some sense, on the *distance* between the input and the result.

This novel behavior is obtained through the use of analog circuit techniques to perform a "collective computation" [Hopfield, 1982]. Each bit of the word is represented by the state

of an active element (amplifier). These elements are interconnected in a way to make the desired memory contents stable states of the network. An association consists of setting the initial conditions of the network to a neutral state, then letting the network evolve to a fixed-point.

2.3 The Hopfield Model for Collective Systems

These concepts were formalized in Hopfield's fundamental work [**?**] [**?**]. He proposed an algorithm to model a collective system, and derived the dynamics of this model. His first model [**?**] provided for the stochastic update of binary processing elements, called "neurons" after the model of McCulloch and Pitts [**?**]. An extension to the model [**?**] allows continuous analog neurons, and generalizes to permit communication via "action potentials."

The stochastic model postulates N neurons; each neuron has 2 possible output states ($V_i = -1$ ("off") and $V_i = +1$ ("on")). The state of the system is characterized by $\vec{V} = (V_1, V_2, ..., V_N)$. The output of each neuron is fed back to the inputs of other neurons. The strength of coupling between the output of neuron j and the input of neuron i is given by T_{ij} – the matrix T is called the interconnect matrix.

The input to a neuron i is given by

$$v_i = \sum_j T_{ij} V_j.$$

The state change in the stochastic model is determined by the following algorithm: at random points in time, each neuron i compares its input with a threshold U_i (usually taken to be zero, by the symmetry of the scheme), and updates its output:

$$\left.\begin{array}{l} V_i \to +1 \\ V_i \to -1 \end{array}\right\} \quad \text{if} \quad \left\{\begin{array}{l} v_i > U_i \\ v_i < U_i \end{array}\right.$$

The justification for this algorithm becomes clear once the construction of the T_{ij} matrix is considered. A self-reinforcement argument leads to the outer product form [**?**]

$$T_{ij} = \sum_s V_i^s V_j^s, \text{ where } \{V^s, s = 1..M\} = \text{set of memories to be stored}$$

but with $T_{ii} = 0$. Hence,

$$\sum_j T_{ij} V_j = \sum_s V_i^s \left[\sum_j V_j V_j^s \right] \tag{2.1}$$

By statistical orthogonality between random states (a natural consequence of optimal encodings), the mean value of the bracketed term in Equation 2.1 is 0, unless V is near V^s,

in which case this term asymptotically approaches N, as $V \rightarrow V^s$. Thus, for a state near a memory $\sum_j T_{ij}V_j \approx N\,V_i^s$, which drives V_i toward V_i^s.

The stability of the system can be investigated by defining a function:

$$E = -\frac{1}{2}\sum_i \sum_j T_{ij}V_iV_j$$

For symmetric T ($T_{ij} = T_{ji}$), the ΔE due to ΔV_i is:

$$\Delta E = -\Delta V_i \sum_j T_{ij}V_j$$

Since, by the update algorithm, ΔV_i and $\sum T_{ij}V_j$ have the same sign, E is a monotonically decreasing function, the local minima of which constitute the memory states. By analogy with the Ising spin model, E is an energy function for the system where T_{ij} is the exchange coupling. Symmetric T_{ij}'s hence give rise to *spin glasses* [**?**], which are known to display many stable configurations.

2.3.1 A Continuous Model

Replacing the binary neurons with elements displaying graded transfer functions, and substituting continuous time constants for the asynchronous update schedule, yields a continuous system that displays similar properties [**?**]. Its analysis is facilitated by considering an electrical circuit model (Figure 2.1).

Application of Kirchoff's Current Law at each input node yields the dynamic equation:

$$C_i\frac{\partial v_i}{\partial t} = \sum_j G_{ij}(V_j - v_i) \tag{2.2}$$

where $V_i = g(v_i)$. The interconnection conductance G_{ij} plays the role of T_{ij}. A Liapunov energy function exists for the system:

$$E = -\frac{1}{2}\sum_i \sum_j T_{ij}V_iV_j + \sum_i \frac{1}{R_i}\int_0^{V_i} g_i^{-1}(V)dV \tag{2.3}$$

where R_i is the input impedance seen by the input of amplifier i ($\frac{1}{R_i} = \sum_j G_{ij}$). The time derivative of Equation 2.3 is (for symmetric T):

$$\frac{\partial E}{\partial t} = -\sum_i \frac{\partial V_i}{\partial t}\left(\sum_j T_{ij}V_j - \frac{v_i}{R_i}\right)$$

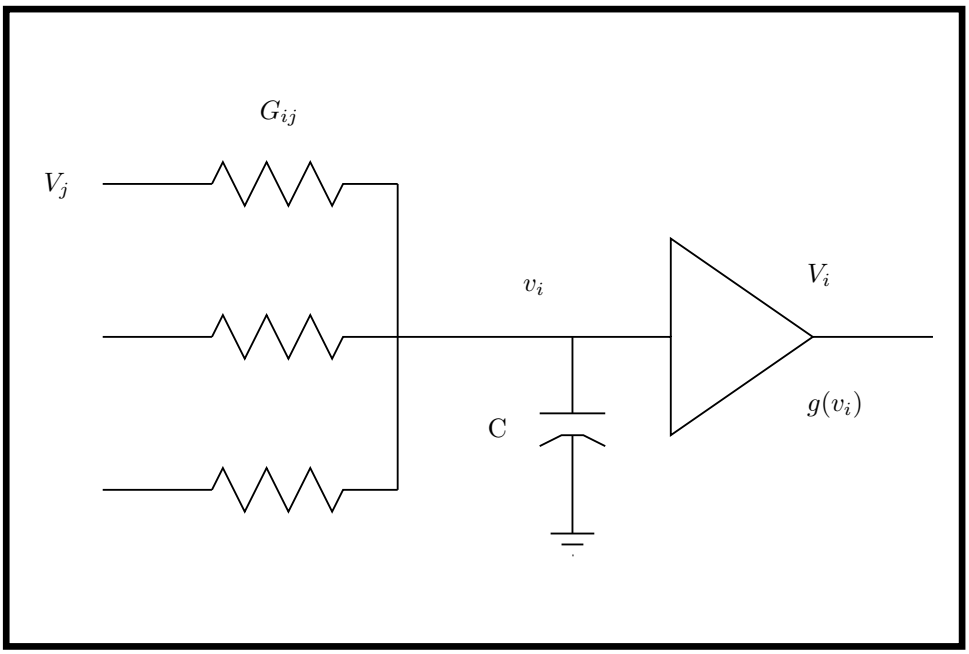


Figure 2.1: Hopfield's neural network model: interconnection of active elements (neurons).

Substituting Equation 2.2 for the parenthesized expression gives

$$\frac{\partial E}{\partial t} = -\sum_i C_i \frac{\partial V_i}{\partial t}\frac{\partial v_i}{\partial t} = -\sum_i C_i \left.\frac{\partial g^{-1}(V)}{\partial V}\right|_{V_i} \left(\frac{\partial V_i}{\partial t}\right)^2$$

If $g^{-1}(V)$ is monotonic and increasing,

$$\frac{\partial E}{\partial t} \leq 0 \quad \text{and} \quad \frac{\partial E}{\partial t} = 0 \;\Rightarrow\; \frac{\partial V_i}{\partial t} = 0, \forall i$$

Thus, the continuous model also describes a system which evolves towards stable states, corresponding to local minima of an energy function.

Such associative memories have several appealing features. First, it is possible to indicate *confidence* on a bit-by-bit basis, by setting the initial conditions appropriately, that is, by making certain bits weigh more heavily, if desired. This property allows the ASSOCMEM to accept analog inputs, thus permitting it to process image or sound data. Secondly, all the bits are treated uniformly, in that the network does not require the distinction of a search *key* from a *data* field in either the source or target words. The device may be thought of as an error-correcting machine, where randomly-occurring bit errors are corrected. Thirdly, the actual memories are *stored in the interconnect* in a highly redundant fashion, making this architecture naturally fault tolerant. Simulation results have indicated that 10% to 20% of all connections may be destroyed with practically no loss of functionality. If a small number of amplifiers are nonfunctional, they can still be "out-voted" by the rest of the network, and errors limited to those bits only.

2.4 VLSI Considerations

Since collective systems exhibit interesting global properties as a consequence of having large numbers of individually simple elements, implementation with very large scale integration (VLSI) circuit technology appears very suitable. There are, however, a number of technology-dependent limitations that are introduced by such a choice.

2.4.1 Cost

The principal cost measure in VLSI is *area*. Even with the use of die-stitching techniques, there exists a physical limit on the maximum area a circuit can occupy. Also, the off-chip environment is quite different from the internal circuit, for electrical reasons. This fact, coupled with the fundamental restriction on I/O pads, makes it desirable to integrate an *entire* system on a single chip.

Analog electronics, by exploiting the intrinsic physics of native devices, generally occupy less area per function than an implementation using a digital abstraction. For example, an analog differential-input multiplier may require as few as 8 transistors to perform the relatively complex calculation ($y = k(x_{1+} - x_{1-})(x_{2+} - x_{2-})$). Furthermore, there is none of the overhead associated with mapping what is essentially a continuous problem into a discrete-time (sampled digital) system.

2.4.2 Power

A common complaint about analog computing elements is that their power consumption is high, due to a desire for maximum linearity at high operating speeds, and because discrete (off-chip) components present relatively highly capacitive loads. In a VLSI context, power dissipation must be limited to a few Watts (for conventional packaging technologies). However, collective circuits implemented entirely on one die have no requirements to drive external loads, do not have to be particularly fast, and value symmetry much more highly than linearity.

It is important to note that the Hopfield circuit model exhibits nonzero power dissipation even after convergence is reached, and the computation is nominally terminated. For systems of several hundred amplifiers, it is not possible to build interconnect matrices of tens of thousands of resistors without explicitly limiting the power consumption of the amplifiers. A commonly suggested alternative, computation by current summing, is even more impractical, as the number of (power dissipating) current injectors that must be controlled scales with the number of synapses.

The approach we have taken to permit the implementation of large arrays is to limit the current consumption of the amplifiers, guaranteed by keeping most of the MOS devices in

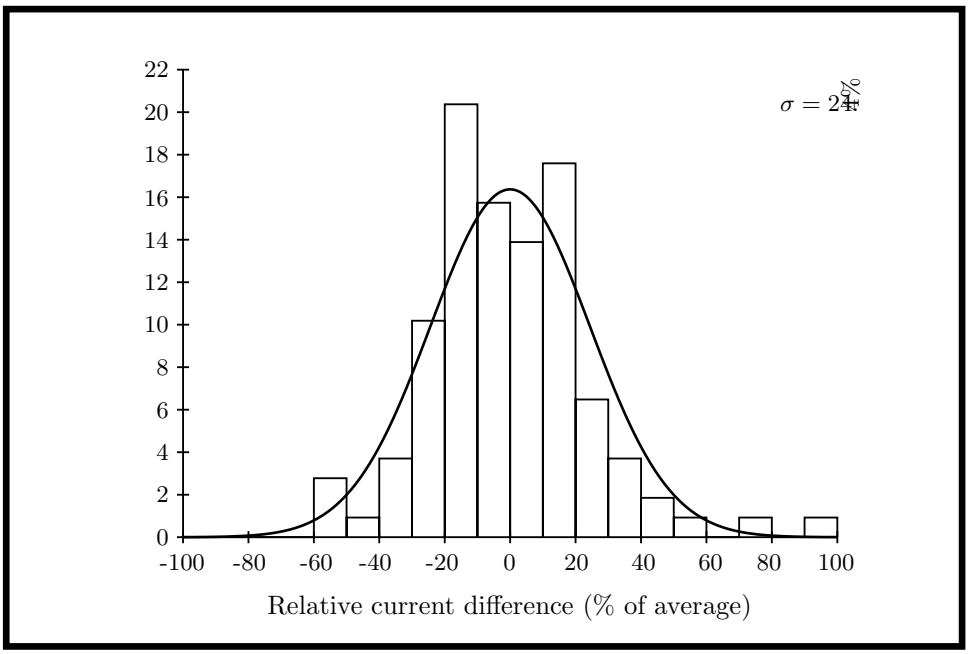


Figure 2.2: Full-wafer current variation in 3x3 micron MOS transistors.

a subthreshold regime of operation [**?**]. For sufficiently low gate voltages (less than the so-called "threshold voltage," below which the digital abstraction of transistor operation classifies the transistor as "off"), the drain current is exponential in the gate voltage. This behavior is exactly the same (and indeed, the physics are identical) as bipolar transistors exhibit throughout their operating range, with the additional benefit that MOS transistors draw no gate current.

2.4.3 Parameter Variation

An additional complication is introduced in the case of very small devices, where statistical or systematic doping variations can affect their transfer characteristics by significant amounts. These variations are particularly evident in the case of fabrication lines intended for digital chips (which are relatively insensitive to such variation). If an analog design methodology is used that requires currents to be precisely matched or subtracted, it is unlikely that sufficient accuracy can be obtained with single small transistors. The degree of variation to be expected is illustrated in Figure 2.2 [**?**], which shows the drain currents of identically biased MOS transistors from a typical MOSIS [**?**] digital process. When differences between *adjacent* transistor currents are taken (Figure 2.3), the variation in relative current differences is substantial, and indicates (in this case) a fairly random process, as opposed to some longer-range (die-scale) systematic variation. These variations can be minimized by using larger transistors (Figure 2.4), or by relying on statistical numbers of transistors to participate in a computation.

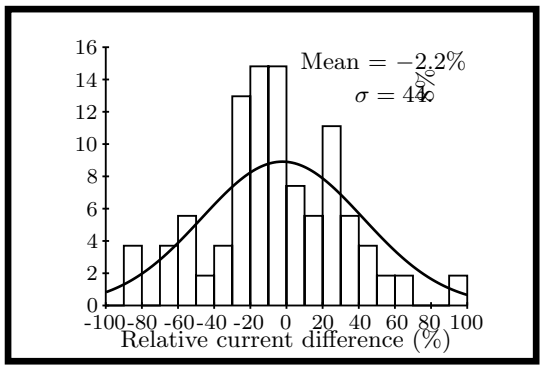


Figure 2.3: Drain current variation in adjacent 3x3μ MOS transistors.

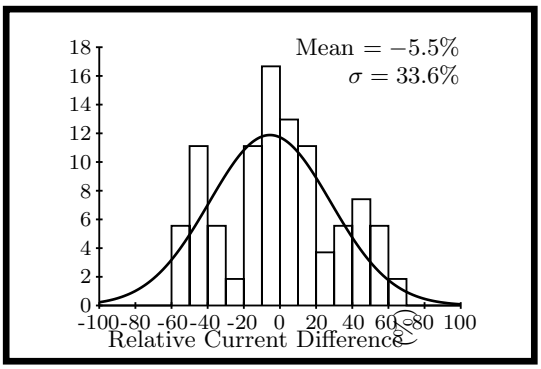


Figure 2.4: Drain current variation in adjacent 24x24μ MOS transistors.

Furthermore, it is clear that a computational scheme must be designed that is robust against such variation, and that displays a high tolerance to noise. Such claims are commonly made of collective systems; they must be carefully examined, however, in light of the actual implementation.

2.5 Design and Fabrication of a VLSI Implementation

The first version of the ASSOCMEM was designed in 4μm nMOS technology, and has a dynamically programmable full interconnect. Consequently, every amplifier must be connectable to every other amplifier, dictating a mesh topology with the amplifiers on the diagonal, and their input and output lines running orthogonal to each other. At each off-diagonal node, an interconnection "conductance," G_{ij}, is located. This topology (used in Figure 2.5) produces a two-dimensional embedding of the interconnect graph implied by Figure 2.1.

2.5.1 Design of the Active Elements

In order to achieve a general computation, both positive and negative signals and interconnect values must be representable. It is not possible to fabricate negative resistances, as would be implied by a negative matrix element. The solution was to design the amplifiers to take differential inputs and generate complementary outputs. These inputs may be thought of as excitatory and inhibitory (i.e., noninverting and inverting). This duality allows us to represent "negative" V_j's while using only positive signals. Also, it guarantees symmetry between "positive" and "negative" V_j's, since they are in fact the same voltage, merely on different lines (Figure 2.6). The matched parameters of VLSI devices ensure a high CMRR for this configuration.

This dual-rail voltage representation on both input and output to the active elements greatly

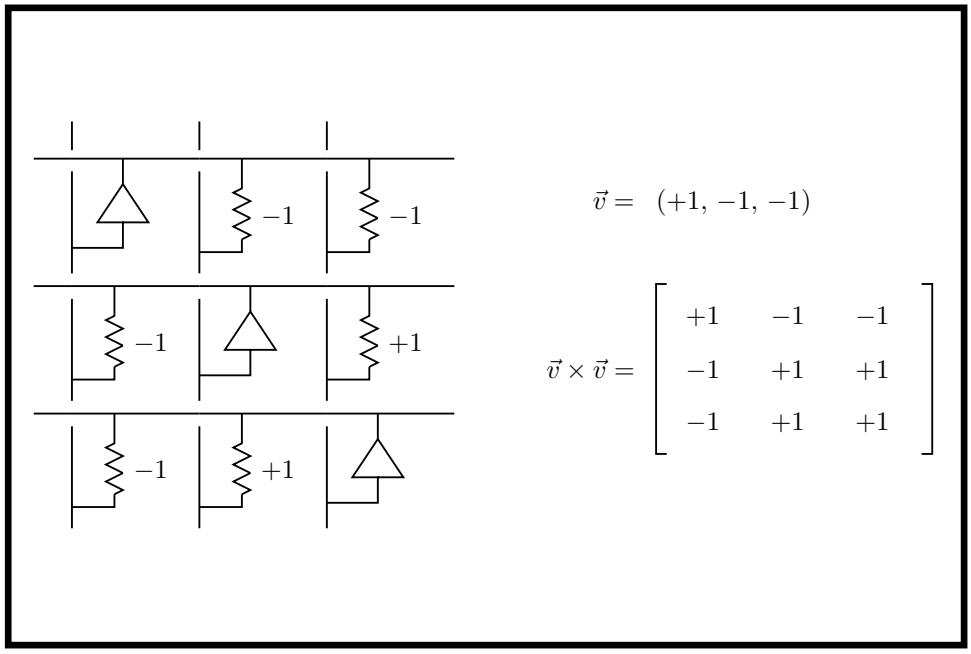


Figure 2.5: ASSOCMEM architecture, and reinforcement of stable state with interconnect matrix generated by outer product technique (guaranteeing the symmetry property $T_{ij} = T_{ji}$).

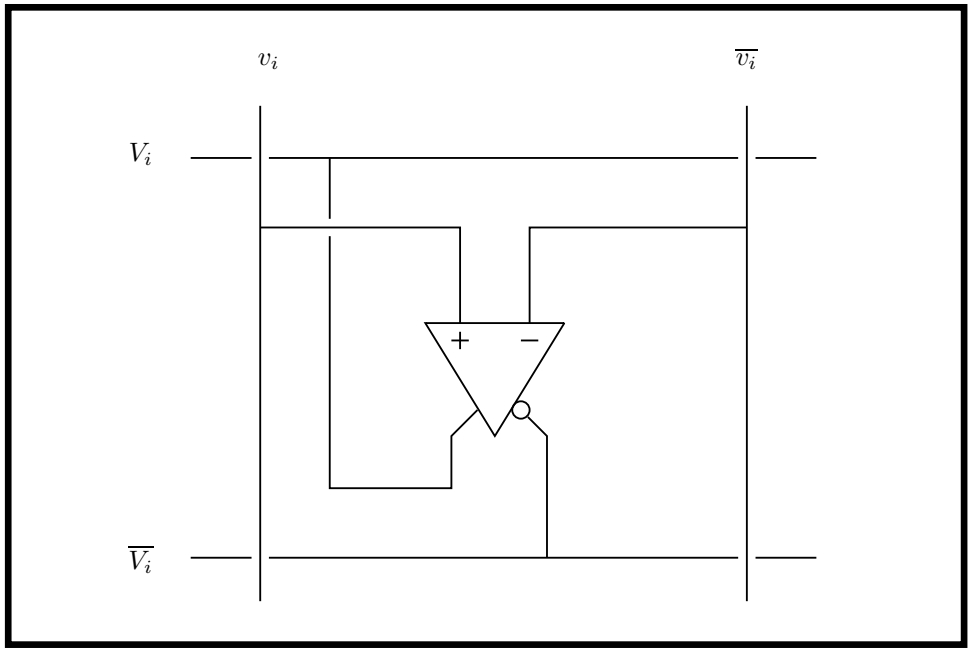


Figure 2.6: Dual-rail signal representation.

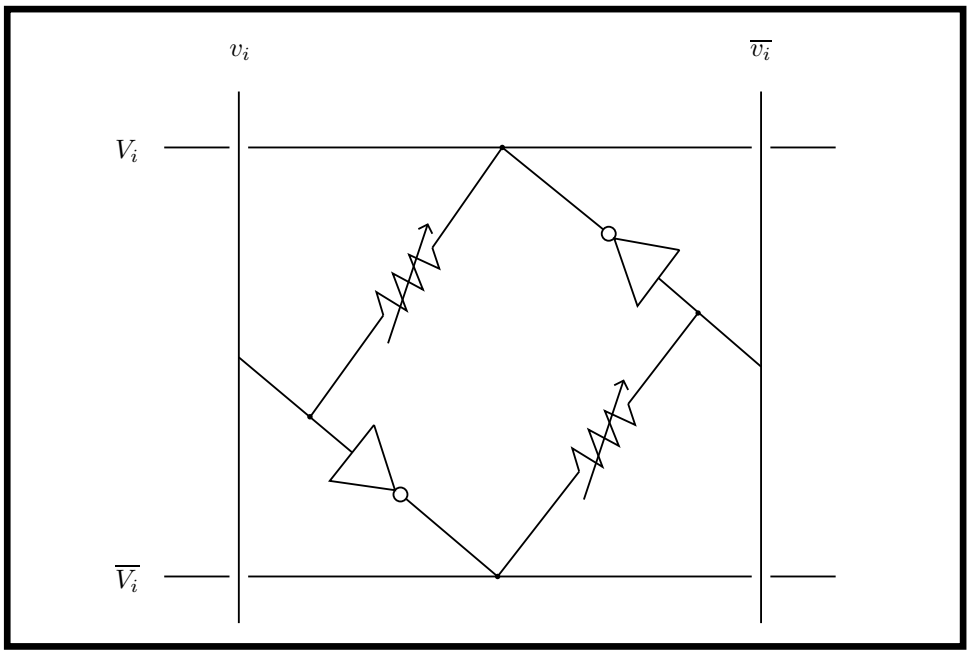


Figure 2.7: ASSOCMEM neuron element, with variable feedback in T_{ii}.

simplifies their design. The only constraint on the transfer function $g(v_j)$ is monotonicity (to guarantee convergence, by the energy theorem), and that the gain be sufficiently large. A loop gain of greater than 1 is required to create distinct stable states; a high gain guarantees that all outputs will be driven to the rails. We chose to implement the amplifiers with the simplest circuit element possible: an nMOS depletion-load inverter (Figure 2.7). Two inverters are required for each amplifier: one between the noninverting input and the complemented output, and one between the inverting input and the uncomplemented output. Note that feedback through the *matrix* is required to ensure that the amplifier outputs are indeed complementary. If the two inverters are replaced by a pair of cross-coupled NOR gates (i.e., a simple SR flip-flop), this complementarity can be enforced at each diagonal (T_{ii}) element.

The design that was implemented allowed for both modes of operation to be tested, by providing a variable strength cross-connection (CROSS_COUPLE) between the two inverters. The circuit diagram for the T_{ii} element is shown in Figure 2.8. Two pass transistors, gated on Φ_1, allow the inputs of the amplifier to be latched. This capability is required for programming the interconnection matrix, and is desirable to halt the operation of the memory, for test purposes.

2.5.2 Resistive Interconnect

The resistive coupling, representing the G_{ij} matrix, is provided by pass transistors, which constitute the functional part of the "G_{ij} Element" (Figure 2.9), located at every off-

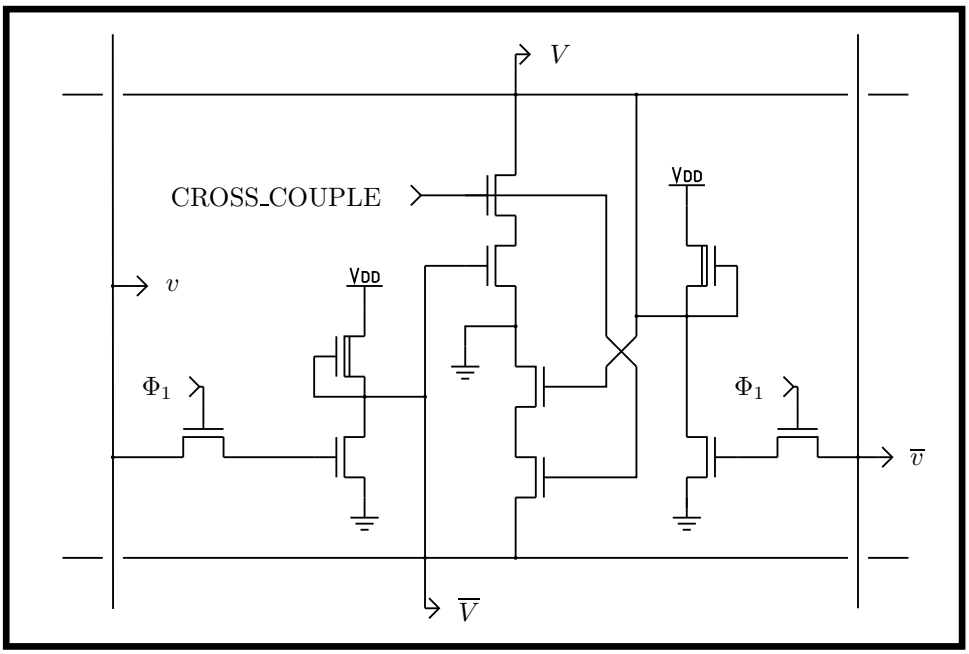


Figure 2.8: T_{ii} element schematic.

diagonal location of the chip. These pass transistors are controlled by the tri-flop T_{ij} cell, which provides three interconnection strengths (+1, 0, −1). When $T_{ij} = +1$, the + output of amplifier j is connected to the + input of amplifier i (excitation), and the − output is connected to the − input of the respective amplifiers (negative inhibition ⇒ excitation); the opposite is true for $T_{ij} = -1$. If $T_{ij} = 0$, all transistors are disconnected, and amplifier j has no *direct* influence on amplifier i. Each pass transistor is in series with an ENABLE transistor that allows the matrix to be selectively disengaged, providing a "single-step" operation.

2.5.3 Programming the T_{ij} Elements

The associative memory can be programmed to have a particular vector as a stable state by adding the outer product of that vector with itself to the existing T matrix. In the ASSOCMEM, this operation is a primitive of the hardware, and has been generalized to allow taking outer products of any two vectors. The components of the outer product are produced in place at the T_{ij} element where they are required, by placing one vector on the horizontal signal lines, and the other on the vertical lines. This calculation requires only a 1-bit by 1-bit multiplication (for vectors containing +1 and −1 exclusively), which can be implemented by an AND structure (Figure 2.10). The result of the correlation is added to the present contents of the T_{ij} cell, and the result is stored in the tri-flop T_{ij} latch.

The addition operation is truncated, and the behavior is described in Table 2.1. It should be noted that this operation is *not* associative (the final matrix depends on the order in which the memories were added); however, the symmetry of the final matrix is guaranteed

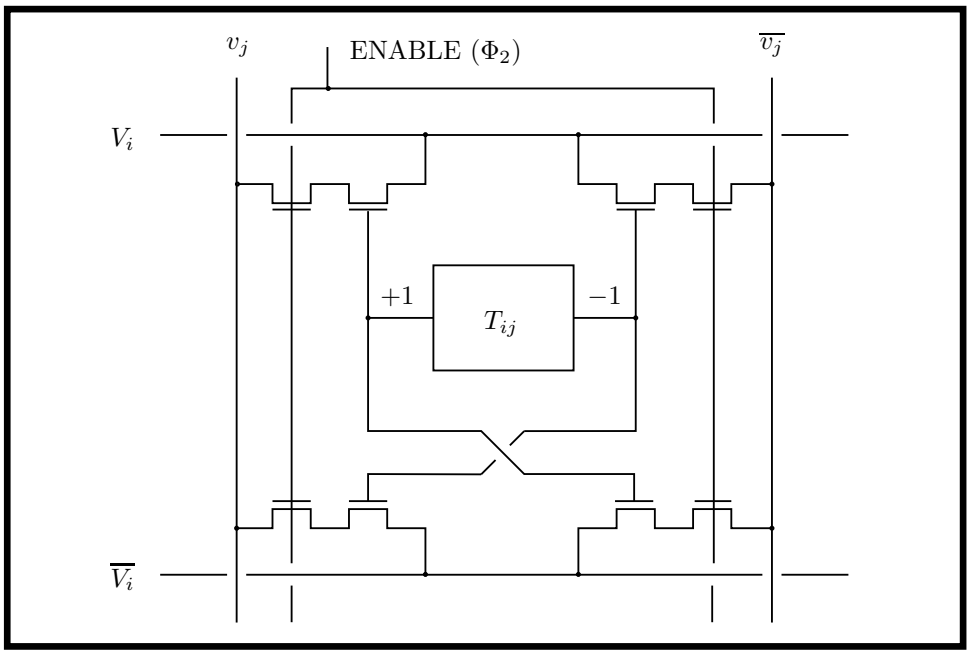


Figure 2.9: Resistive interconnect.

(since all intermediate sums comprised symmetric matrices, in the case of outer products taken of a vector with itself).

The effect of truncating the matrix in this fashion was one aspect of the theory we were interested in investigating. Software simulation of this scheme indicated that little functionality would be lost, in the case of uncorrelated memories (memories sufficiently far apart in Hamming space, so that no small number of bits become crucial in differentiating between them).

The complete schematic of the T_{ij} element, and the interconnection "resistors," is shown in Figure 2.11. It should be pointed out that all but four of the transistors are required just to implement the *programmability* of the ASSOCMEM. In many applications, the interconnect matrix could be determined a priori, and the chip fabricated to this specification. We estimate that 100 times as many active elements could be fabricated within the same die size, for a predetermined function.

2.6 Fabrication History

The first version of this chip was fabricated by MOSIS, the ARPA chip broker, on run M41U, in December 1983. It consisted of a 22 × 22 matrix, measuring 6700μm × 5700μm, and containing over 20,000 transistors (Figure 2.12). The ASSOCMEM required 53 pads, as both inputs of each amplifier were brought out to pads. This redundancy was incorporated to increase flexibility in testing the chip. Only one input per amplifier is required, as the

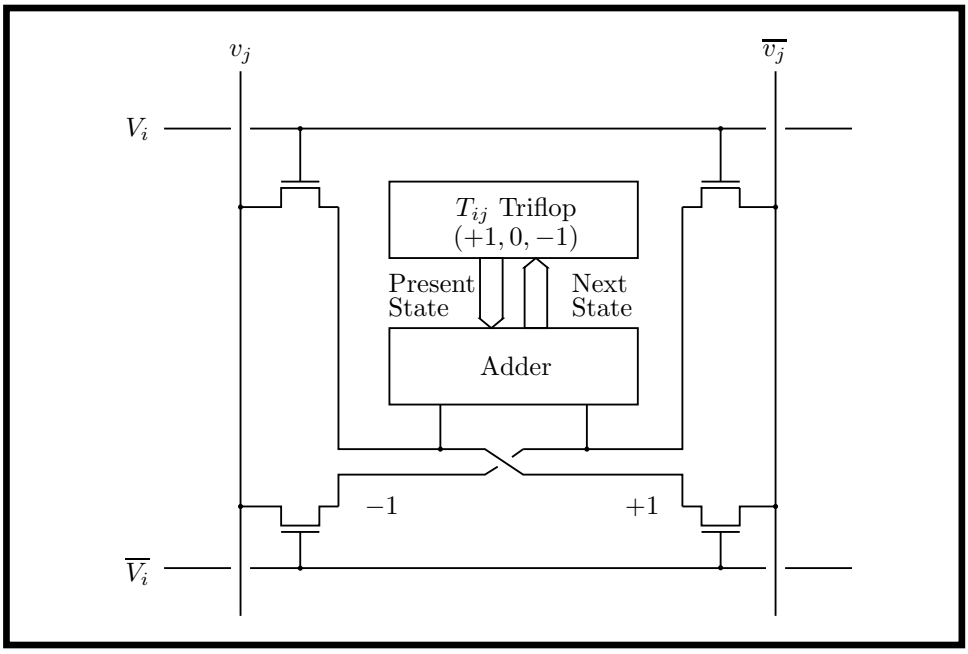


Figure 2.10: Programming the T_{ij} matrix.

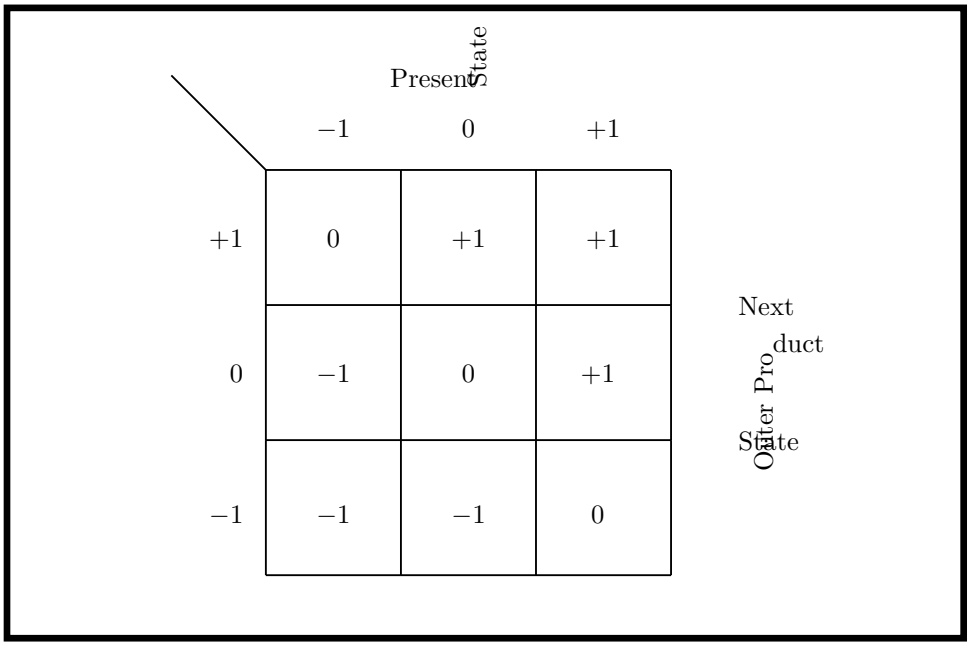


Table 2.1: T_{ij} update operation.

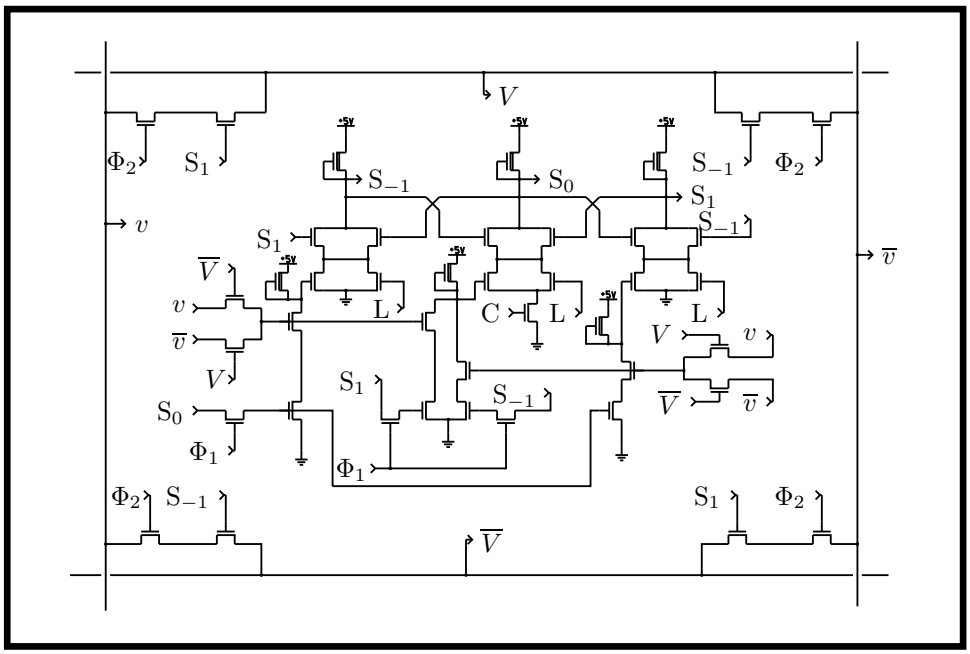


Figure 2.11: Schematic of T_{ij} element. S_1, S_0, and S_{-1} are the three state-holding nodes, C is the CLEAR line, and L is the LOAD line.

complement could be generated on-chip.

A second version of the chip was fabricated in May 1984, correcting some defects that made sections of the first chip nonfunctional and nonobservable. This redesign yielded testable chips; however, the chips were returned unbonded, and only a few could be bonded locally and tested. The yield was low, and no single project was fully functional.

The desire for a larger sample size of chips prompted a third re-submission, which incorporated a set of minor modifications making it possible for MOSIS to package and bond the ASSOCMEM. This set of chips was returned in November 1984, and was subsequently tested.

2.7 Experimental Results

The ASSOCMEM was tested on a dedicated, workstation-based functional tester. The first sets of experiments were designed to measure the input-output characteristics of the active elements. The input of a single T_{ii} element was connected, (via T_{ij} elements) to the outputs of the other amplifiers. The output of this element was monitored for 220,000 randomly chosen input vectors (this operation corresponds to taking the inner product of the random input vector with a vector that is all 1s). The resulting output was plotted as a function of the arithmetic sum of the input bits (i.e., the number of +1's in the input vector minus the

Figure 2.12: The ASSOCMEM.

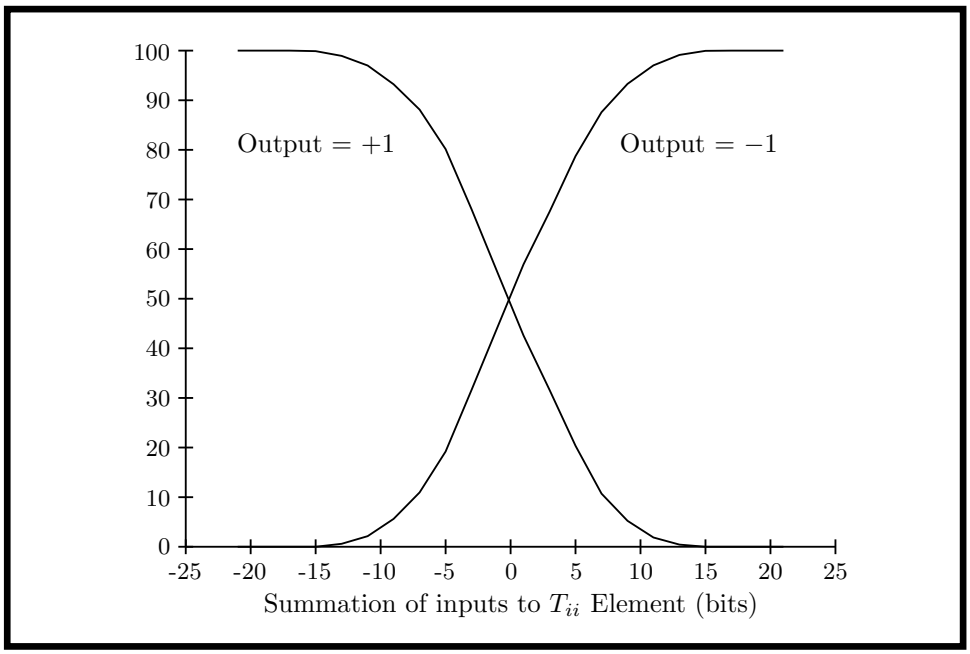


Figure 2.13: T_{ii} transfer characteristic.

number of -1's), as shown in Figure 2.13. The symmetry of the graph demonstrates that the output is unbiased and the input sum to the active element is balanced for an input vector containing equal numbers of $+1$'s and -1's.

The ability to arbitrarily set a single T_{ij} element confirmed the functionality of the adder elements and tri-flops within each T_{ij} element. Several ring oscillators were programmed into the matrix, ranging from 3 stages to 19 stages. A typical waveform is shown in Figure 2.14. The results in Figure 2.15 indicate a characteristic propagation delay of 510ns per stage. The ring oscillators required an asymmetric connection matrix, as oscillatory behavior is excluded for symmetric matrices.

The next experiment was to perform an actual association. Two random memories were programmed into the matrix, via the on-chip outer product and adder mechanisms, and were observed to be, in fact, fixed points of the network. Due to the fast T_{ii} elements, the convergence of the network, including the driving of the output pads, was generally completed within a few microseconds.

If the function of the ASSOCMEM is considered to be that of an error-correcting code, it is useful to consider the final stable state as a function of the bit-errors in the initial state. Typical experimental results demonstrating the distinct regions of convergence for a two memory system are shown in Figure 2.16. Effectively, this system always converges to the memory nearest in Hamming space to the initial condition.

By using Φ_1 and Φ_2 as nonoverlapping "clocks" to successively engage and disengage the connection matrix and T_{ii} elements, it is possible to "single-step" through the convergence

Figure 2.14: Ring oscillator.

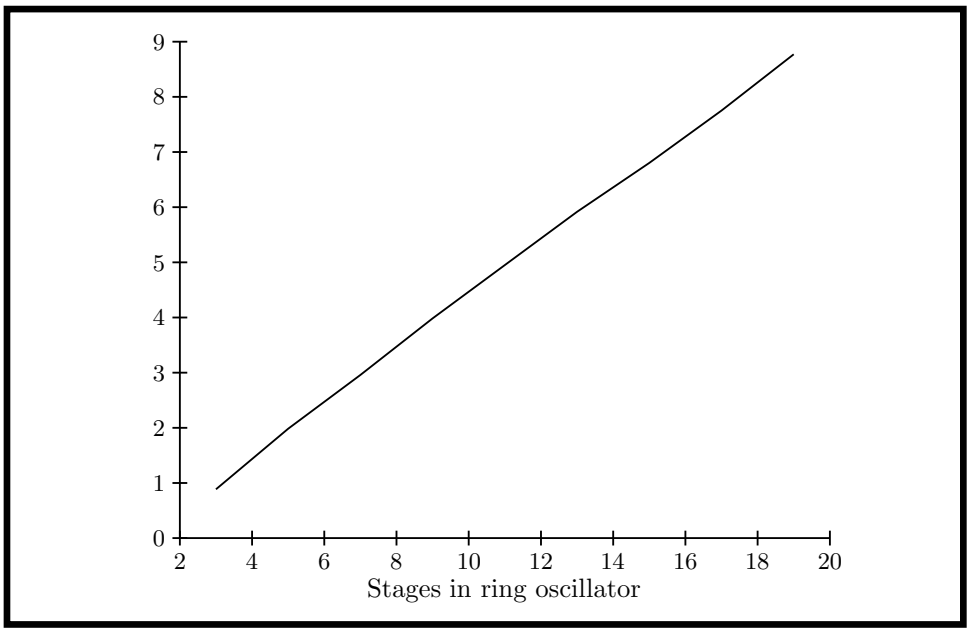


Figure 2.15: Ring oscillator periods.

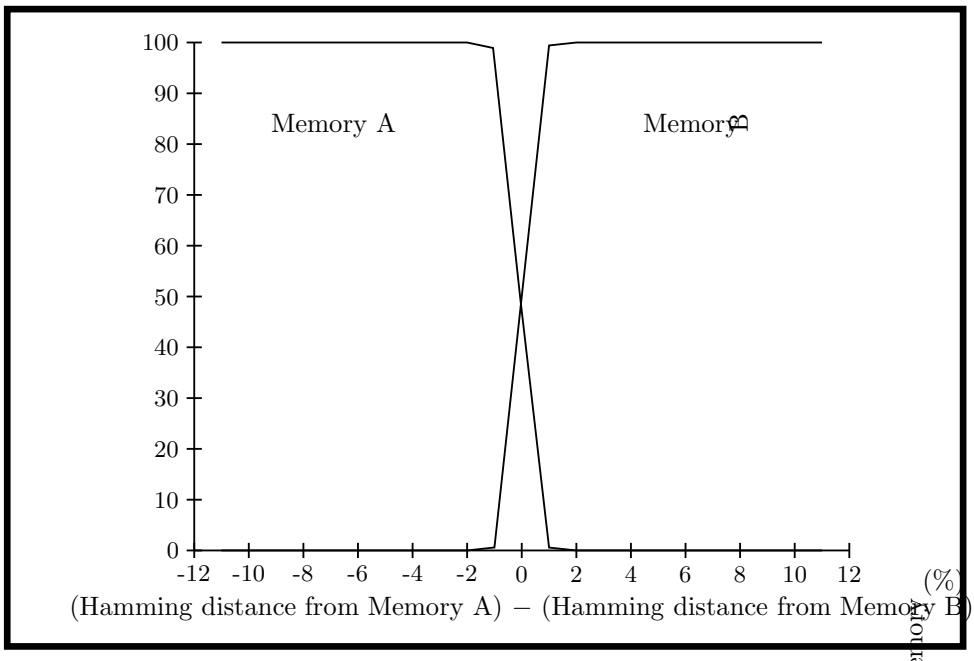


Figure 2.16: ASSOCMEM convergence properties.

operation. Each step corresponds to the network synchronously performing an inner product calculation to obtain the next state from the present state. It can be proven that a network which converges in synchronous-update mode will also do so asynchronously (i.e., Φ_1 and Φ_2 ON simultaneously), although the converse is *not* true. For networks that do converge in single-step mode, it is interesting to observe the process in action (see Figure 2.17). In this illustration, note that the intermediate results are restored approximations to the analog voltages within the ASSOCMEM, and that although two iterations appear identical, the convergence is not yet complete.

Simulation results [**?**] have suggested an empirical limit on the number of uncorrelated memories that can be programmed into a Hopfield network to be approximately 15% of the number of active elements. Beyond this, spurious stable states begin to appear, often displacing desired memories. Since the network is based on mutual inhibition and reinforcement, any correlation between memories causes some bits to be weighted more heavily than others. Statistical orthogonality between memories minimizes this effect.

A second limitation on the number of stable states arises due to the clipped nature of the T_{ij} matrix. Hopfield and Feinstein [Feinstein and Hopfield, 1985] showed that the "attraction" of stable states decays exponentially with the number of memories subsequently added to the matrix.

For the ASSOCMEM, it was found that up to three stable states could be reliably programmed, along with their complements (which appear due to the symmetry of the system). At all stages of the experimentation, consistency with simulation results was observed.

```
Incremental Association:

(stable states  (hex):  0F0F0F, 000FFF)
(vector written (hex):  00000F)

Iteration 0:  V = 00000000000000001111
Iteration 1:  V = 00000000000111100001111
Iteration 2:  V = 00000000000111100001111
Iteration 3:  V = 00010100000111100001111
Iteration 4:  V = 00111100000111100001111
```

Figure 2.17: Single-step association.

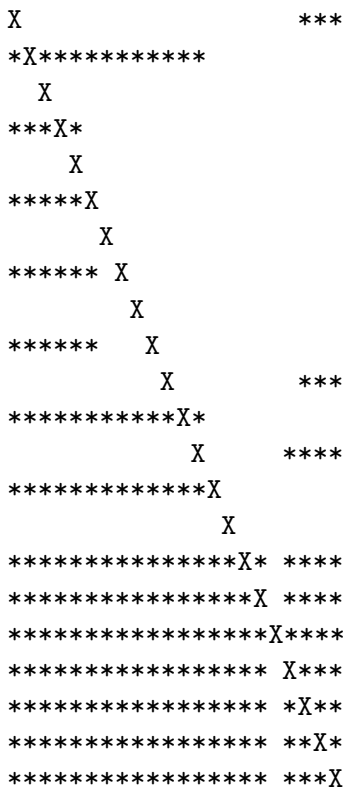


Figure 2.18: T_{ij} yield map.

Of the ten chips returned by MOSIS, all were found to be functional to the extent of permitting at least two stable states. A CIF defect made a particular T_{ij} element malfunction in all chips. On some chips, sizable numbers of T_{ij} elements were inoperative. A sample map, from a particularly bad chip, is shown in Figure 2.18 ("$*$" represent good T_{ij} elements). Since care was taken to keep the CLEAR circuit simple, and the matrices on all chips were observed to clear correctly (and totally), it is postulated that the primary failure mode was T_{ij} stuck-at-zero faults. Since null T_{ij}'s do not contribute *detrimentally* to the operation of the network, even the ASSOCMEM of Figure 2.18 was capable of memorizing two stable states, and associating to them.

2.8 High-Density Preprogrammed Associative Memories

In order to explore the behavior of large associative memories, we undertook the design and fabrication of a series of chips incorporating fixed (i.e., mask-programmable) interconnect matrices. These T_{ij} elements were considerably smaller than the programmable T_{ij} element described in Section 2.5.2. Because of pitch-matching difficulties between the interconnect conductances and the active amplifiers, a different topology was employed (see

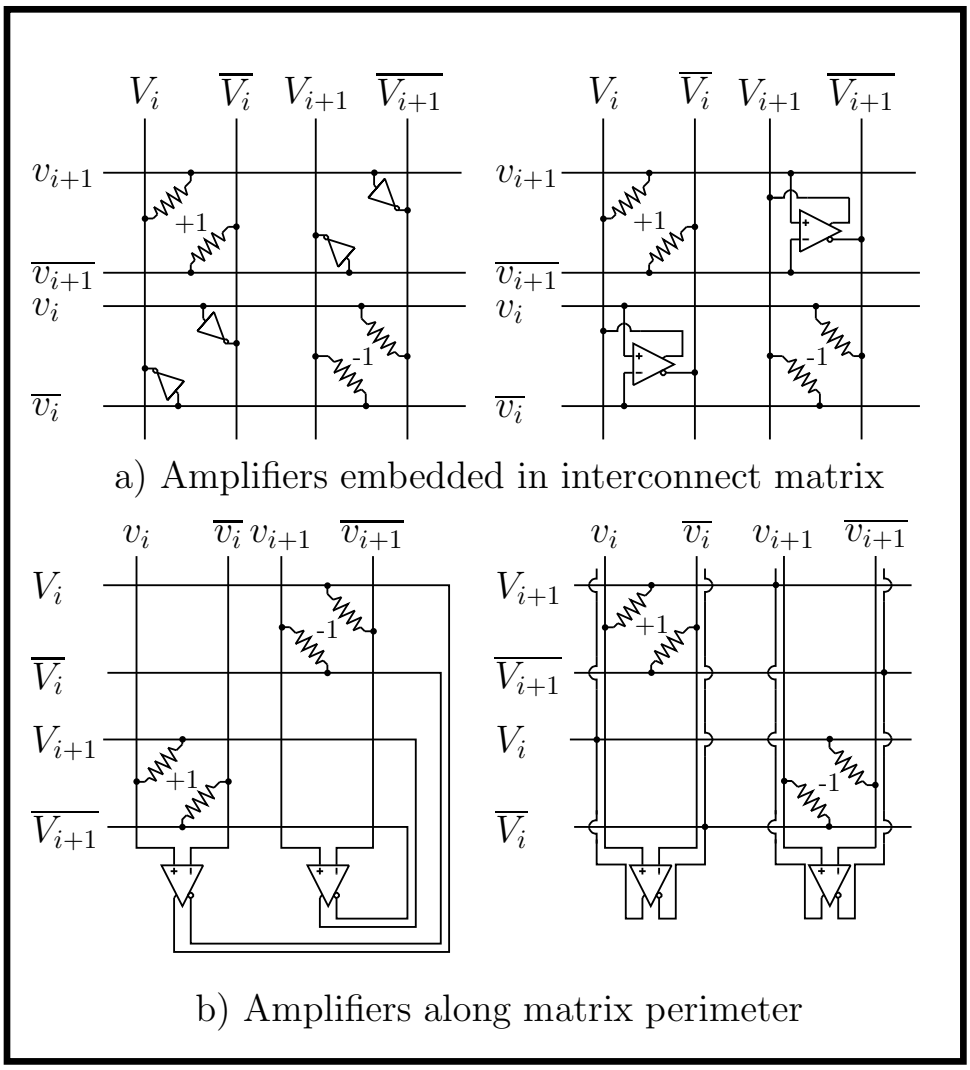


Figure 2.19: Fully-connected associative memory layout strategies.

Figure 2.19(b)); arranging the amplifiers along two sides of the array also facilitated I/O (the states of the neurons were read/loaded by means of a serial shift register).

At this time, a redesign in a 3μ CMOS technology was undertaken. The amplifiers, which had been implemented with inverters in nMOS, were dual-output transconductance amplifiers, with true differential inputs (see Figure 2.20). The interconnect conductances measured $16\times18\lambda$, and could be mask-programmed to the values $(-2,-1,0,+1,+2)$. The entire chip contained 290 neurons (and 167620 possible transistors in the interconnect matrix). Twenty memories were stored, corresponding to bit-patterns for a schematized alphanumeric character set sampled at 17×17 resolution.

The chip was fabricated on MOSIS run M65N in May, 1986. Upon testing, we found that all 20 memories were stable, and that the performance-limiting step was the serial readout

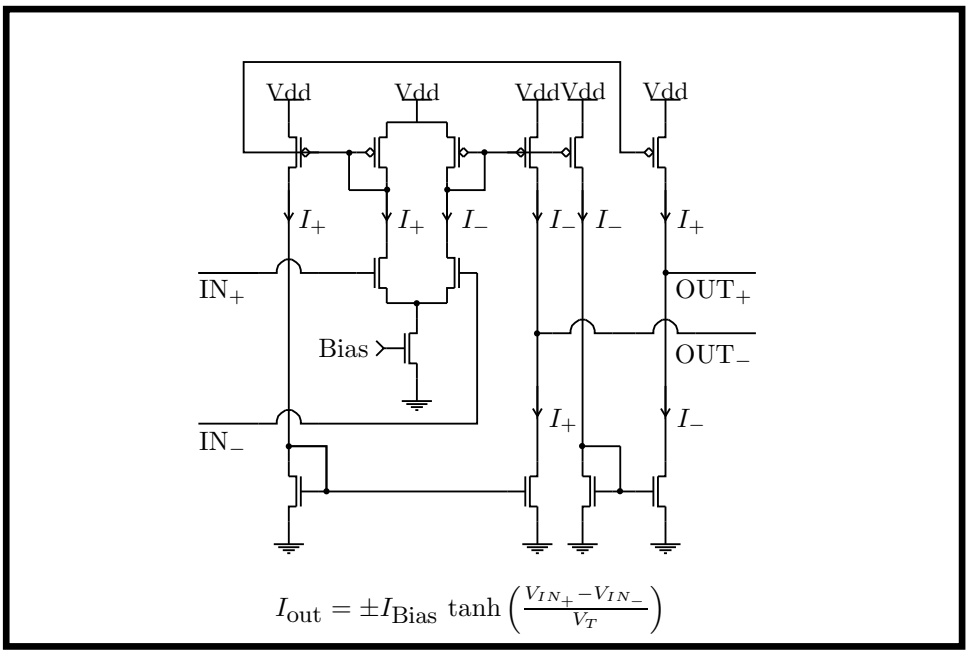


Figure 2.20: Subthreshold CMOS active element.

of the results. Nonetheless, 50,000 associations per second were attainable. Of course, if a higher data rate were required, word-parallel inputs to the shift register would accelerate the data transfer.

2.9 Summary

Collective systems exhibit many appealing properties, including robustness and fault-tolerance, an ability to deal with ill-posed problems and noisy data, which conventional digital architectures do not. The ASSOCMEM represents a novel computational structure for VLSI. By implementing a well-understood system, such as associative memories, with collective circuit design, we have produced a functional component demonstrating these properties. Collective circuits are ideally suited to VLSI, as they require very large numbers of individually simple elements.

However, the model, as presented in this chapter, scales very poorly with available silicon area. The main difficulty is due to the full interconnect of the general Hopfield model; every neuron has to connect to every other neuron. Thus, the number of neurons, which determines the length of the codewords stored in the memory, increases proportionately to the *square root* of the die area. A secondary difficulty arises in the practical implementation of the different components: the architectures we have discussed require pitch-matching of the "neuron" and "synapse" components. In the 22-neuron ASSOCMEM, the programmable synapses were much more complicated than the active gain elements; in the preprogrammed associative memory, the synapses consisted of merely two transistors, while

the CMOS amplifiers were much more complex.

Nonetheless, as feature sizes continue to shrink to the submicron range, the design of associative memories containing 1000 fully-connected neurons is feasible. Because of their ability to reach crisp decisions in the face of noisy or incomplete data, we expect that such networks will find application in the analysis of "fuzzy" sensory data, an application at which biological processes excel, while conventional computer architectures fail miserably.

Chapter 3

A Theoretical Model for Estimation of Circuit Wiring Density

Get your facts first, then you can distort them as you please.

Mark Twain (1835–1910)

3.1 Introduction

The need to consider complexity of wiring as an essential component of any algorithm has been a key contribution to computer science from VLSI design. Whereas in digital VLSI, wiring considerations give rise to an area–time tradeoff, in analog VLSI (in the absence of time multiplexing) wiring complexity determines the *scalability* of an architecture or algorithm. This constraint is amply illustrated by the full interconnect of the Hopfield associative memory presented in Chapter 2. A fully-connected architecture requires $\mathrm{O}(N^2)$ area, where N is the number of neurons in the collective system.

Much of VLSI complexity theory has concentrated on efficient embeddings of particular computational graphs, under particular models for interconnect [Thompson, 1980] [Ullman, 1984]. The basic procedure is to select a class of graphs, and derive an information-theoretic lower

bound for a separator (or partitioning) operator. This separator is then combined with a simple synchronous model for data communication to determine a minimum AT^2 constraint on the computation.

In this section, we take a different approach: we ignore the time cost of communication (we assume that our analog networks use dedicated wires for all signals), and consider only the connectivity of graphs that can be generated when we define a set of layout parameters that specify acceptable component placements. This procedure will define a necessary (but not *sufficient*) constraint on the connectivity of a circuit graph, in contrast to traditional approaches, which generate the worst-case solution for a class of graphs, not the best-case for a specific graph (i.e., solutions which are "*not* universally optimal, but existentially optimal" [Bhatt and Leighton, 1984]).

Under a connectivity model based on the intrinsic dimensionality of the interconnect technology, we will derive a wire partition function to bound the number of wires that can cross a closed perimeter. We will show how this function is consistent with a common empirical wiring relation known as Rent's rule. Finally, we will consider acceptable *distributions* of wire lengths within regular layouts, and derive a physical constraint on the average wire length predicted by the model.

3.2 Rent's Rule

A circuit may be considered as a graph, the vertices of which represent elements, linked by arcs corresponding to wires. Rent's rule [**?**] is an empirical relationship that relates the number of wires crossing an arbitrary boundary containing a group of elements:

$$P = P_0 N^b \tag{3.1}$$

where P is the number of wires that need to cross the border, N is the number of elements (each of which has P_0 pins) contained within the border, and b is an assumed constant. Analysis of several large designs has indicated that typical values lie in the range $b \in [0.5, 0.75]$ [**?**].

3.3 An Operational Definition of Optimal Design

In this section, we introduce an *aesthetic* evaluation metric by which we can measure the quality of a design's implementation. Unlike the information-theoretic lower bounds developed by classical VLSI complexity theory [Thompson, 1980] [Ullman, 1984], we are less interested in bounds on *classes* of graphs than in considering the optimality of the layout of a specific instance within the class. In particular, we are interested in specifying minimum acceptance levels, below which a layout is considered too inefficient, and hence not economically viable. Finally, we will show how our constraint conforms to an empirical wiring relation known as Rent's rule.

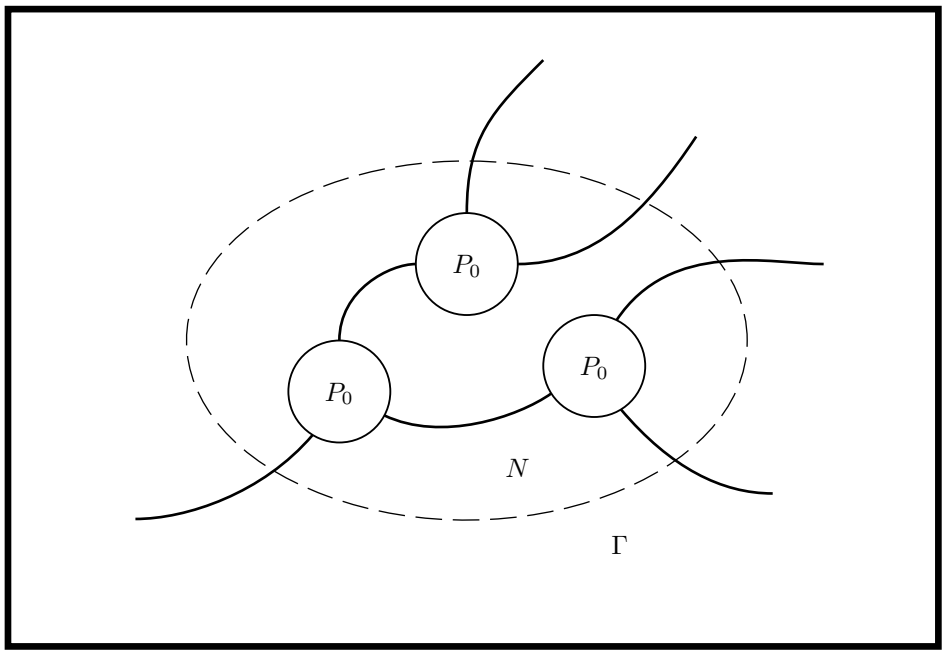


Figure 3.1: Estimating the number of wires crossing the perimeter.

We begin with a key observation regarding manufactured systems. We consider such systems to be examples of "optimal systems"—they have successfully satisfied the constraints imposed by the end-application (e.g., cost, size, power, weight, reliability, manufacturability).

Observation 3.1 In any *system* consisting of a collection of *components*, these components are distributed spatially in an approximately uniform manner.

The phrasing of this observation is intentionally general; we intend for the results of this section to be applicable to many different systems, ranging from integrated circuits to printed-circuit boards to entire racks and boxes. In this section, we combine this observation with the physical dimensionality of interconnect wiring to obtain an upper bound *partition function*. This function bounds the number of wires crossing a manifold containing a number of components of the system in question (see Figure 3.1). We will derive a particular function for systems that are intrinsically two-dimensional (e.g., integrated circuits and printed-circuit boards) and repeat the derivation for three-dimensional systems (e.g., computer back-planes and other aggressive interconnect technologies).

We begin by quantifying Observation 3.1: In a two-dimensional space, assume the density of components is uniform and equal to ρ components per unit area. Furthermore, assume that a fraction γ of the total area of the system is consumed by components. If we model the components as identical and square, of dimension $x \times x$, we have

$$x^2 = \frac{\gamma}{\rho}$$

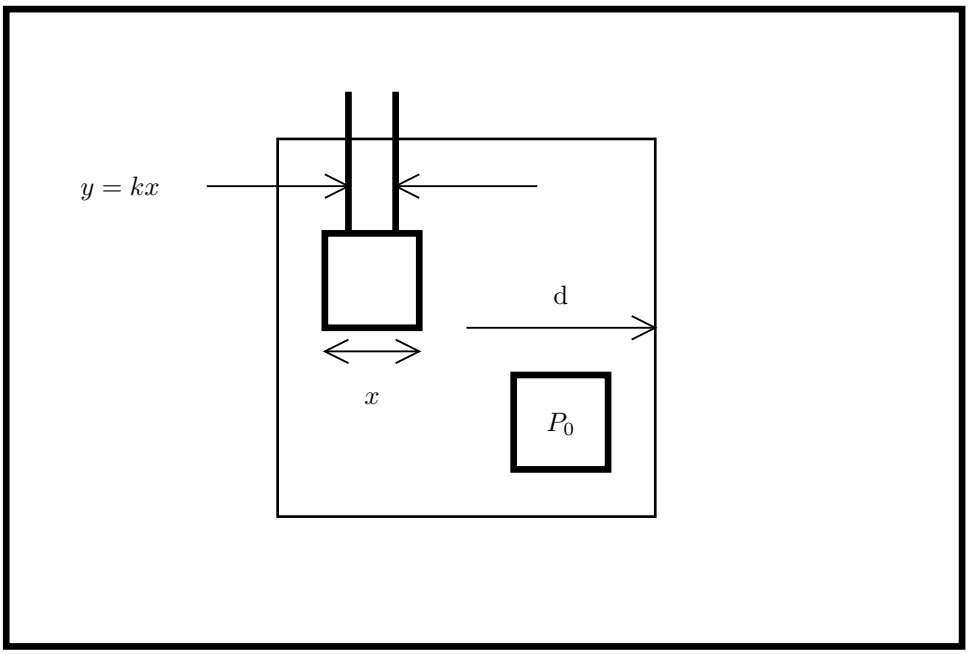


Figure 3.2: A model for circuit connectivity. The area in question measures $2d$ by $2d$, and has perimeter $8d$.

Proof: Consider an area A of the system. By definition, the number of components within this area is ρA, and the fraction of area occupied by these components is γA. The average area per component is then γ/ρ, and, because the components measure x by x, the result $x^2 = \gamma/\rho$ follows directly. □

We now apply this result to the system in Figure 3.2: a square area of dimensions $2d \times 2d$. We wish to compute the number of wiring tracks crossing the perimeter as a function of the number of components within the boundary. We begin by parameterizing the space occupied by these wiring tracks (i.e., the wiring *pitch*) in terms of the single dimension of the model: x, the size of the components.

Let the average wire spacing $y = kx$. In the two-dimensional model, k is a scalar, because x and y are simple linear dimensions.

The component count within the boundary is then

$$N = 4d^2\rho \tag{3.2}$$

The maximum pin count *per element* is limited by the number of wiring channels that can abut the perimeter of a single component:

$$P_0 = \frac{4x}{y} = \frac{4}{k} \tag{3.3}$$

3.4 A Model for Circuit Connectivity

Estimating the number of wiring channels that cross the perimeter of the region is somewhat more subtle. The analysis is complicated by the need to model the routing capacity through the perimeter Γ in regions where Γ lies *within* a components. We make the (rather pessimistic) assumption that *no routing* can occupy the same physical location as the components. This restriction is appropriate for integrated circuits, where programmable logic devices (PLDs), gate arrays, and standard-cell designs tend to have preallocated wiring channels between rows of cells, with severely limited (if any) routing capability through these cells. In particular, this restriction is appropriate for the architecture in Chapter 4. It is less appropriate for technologies such as multi-layer PCBs; however, as we shall see, this approximation only affects certain multiplicative constants—it does not affect the general form of our results.

For randomly placed components, the probability that any given point on the perimeter lies within a component is γ; consequently, the amount of perimeter available to wiring tracks is $8d(1-\gamma)$. The maximum number of wiring tracks is then

$$P = \frac{8d(1-\gamma)}{kx} \tag{3.4}$$

Combining Equations 3.2, 3.3, and 3.4, and recalling $x^2 = \gamma/\rho$, we obtain the partition function

$$P = \frac{1-\gamma}{\gamma^{1/2}} P_0 N^{1/2} \tag{3.5}$$

As a self-consistency check, we can compare this expression with the conventional statement of Rent's rule (Equation 3.1): $P = P_0 N^b$. Setting $b = 1/2$, and solving $\frac{1-\gamma}{\gamma^{1/2}} = 1$, we obtain equivalence between the two forms for $\gamma = 0.38$. This result corresponds to an area utilization of approximately 40 percent, a typical value for semiautomated VLSI designs (gate arrays, standard-cells, etc.).

We are now in a position to reconsider our assumption that area dedicated to components was unavailable to routing tracks. If all of the perimeter is available for routing, Equation 3.5 becomes $P = \gamma^{-1/2} P_0 N^{1/2}$, which would allow γ to reach values near 1. Of course, such high utilization of board area is precisely what we see for typical multi-layer printed-circuit boards.

Now let us consider a higher dimensionality system. In three dimensions, we set $y = kx^2$ (i.e., wires now have finite cross-sectional area), and observe that $x^3 = \gamma/\rho$.

The number of components within the volume is

$$N = \rho(2d)^3 \tag{3.6}$$

The maximum number of wires connecting each component is

$$P_0 = \frac{6x^2}{y} = \frac{6}{k} \tag{3.7}$$

The maximum number of wires crossing the surface Γ is

$$P = (1-\gamma)\frac{6(2d)^2}{y} \tag{3.8}$$

Again, combining Equations 3.6, 3.7, and 3.8, we obtain

$$P = \frac{1-\gamma}{\gamma^{2/3}} P_0 N^{2/3} \tag{3.9}$$

Setting $\frac{1-\gamma}{\gamma^{2/3}} = 1$ gives $\gamma = 0.43$.

In summary, this analysis yields wiring bounds that are consistent with Rent's rule in form, and that suggest that the typically observed values for Rent's rule exponents (typically in the range [0.5, 0.7] [**?**]) are due to dimensional constraints of the implementation technology.

3.5 Contact Density

An interesting consequence to the partition function is obtained by considering the number of *contacts* that must be placed in order to satisfy the wiring limit of Equation 3.5. Clearly, not all of the wires that originate within the perimeter of the region in Figure 3.2 can propagate through the boundary. Each of the other wires must terminate at a contact with at least one other internal wire. For the purposes of this analysis, we assume, for simplicity, that precisely two wires meet at a contact (this assumption does not affect the form of the final result).

The total number of wires originating within a region measuring $L \times L$ (or, $d = L/2$) is

$$P_{\text{total}} = P_0 L^2 \rho$$

The maximum number that can pass through the region's perimeter is given by Equation 3.4:

$$P = \frac{4L(1-\gamma)}{kx}$$

The number of required contacts is then

$$P_c = \frac{1}{2}\left(P_0 L^2 \rho - \frac{4L(1-\gamma)}{kx}\right)$$

In the large L limit,

$$P_c \to \frac{P_0 \rho L^2}{2}$$

and the average area per contact is

$$A_c = \frac{L^2}{P_c} = \frac{2(1/\rho)}{P_0}$$

We can interpret this expression by observing that $(1/\rho)$ is the average area per component, in which P_0 pins originate. Each pin must terminate at a contact—the factor of 2 indicates that each contact has two wires connected to it.

If we compute the area density of contacts as a function of d, we obtain (recall that $\rho = \gamma/x^2$, and that $P_0 = 4/k$)

$$\begin{aligned} \Delta P_c &= \frac{P_0}{x}\left(\frac{4d\gamma}{x} + \gamma - 1\right)\Delta d \\ \Delta A &= 8d\Delta d \\ \frac{\partial P_c}{\partial A} &= \frac{P_0}{x}\left(\frac{\gamma}{2x} + \frac{\gamma - 1}{8d}\right) \end{aligned} \tag{3.10}$$

Thus, the area density of contacts as a function of position d tends to a uniform limit. This result is consistent with our original assumption that components are uniformly distributed throughout the region.

3.6 Limits on Wire Length

Although the results of Section 3.3 place upper bounds on the *number* of available wiring channels, they tell us nothing about the macroscopic properties of the interconnect. In particular, it would be useful to consider the distribution of wire lengths predicted by the model, and to verify that this distribution satisfies the dimensional constraints of the implementation technology.

Before we can compute the average wire length, we must consider the rate at which wires from a given cell terminate as a function of distance. Consider the two-dimensional case illustrated in Figure 3.3; a circular partition is drawn a distance r from the cell in question. Applying the general form of Rent's rule, we have

$$\begin{aligned} P &= P_0 N^b \\ &= P_0 \gamma^b \left(\frac{r}{r_0}\right)^{2b} \end{aligned}$$

since $N \equiv \frac{\pi r^2 \gamma}{\pi r_0^2}$.

If we consider the region between r and $r + \Delta r$, we estimate the wires that cross the outer perimeter to be

$$P_{\text{outer}} = P \cdot \left(1 + \frac{2b\Delta r}{r}\right)$$

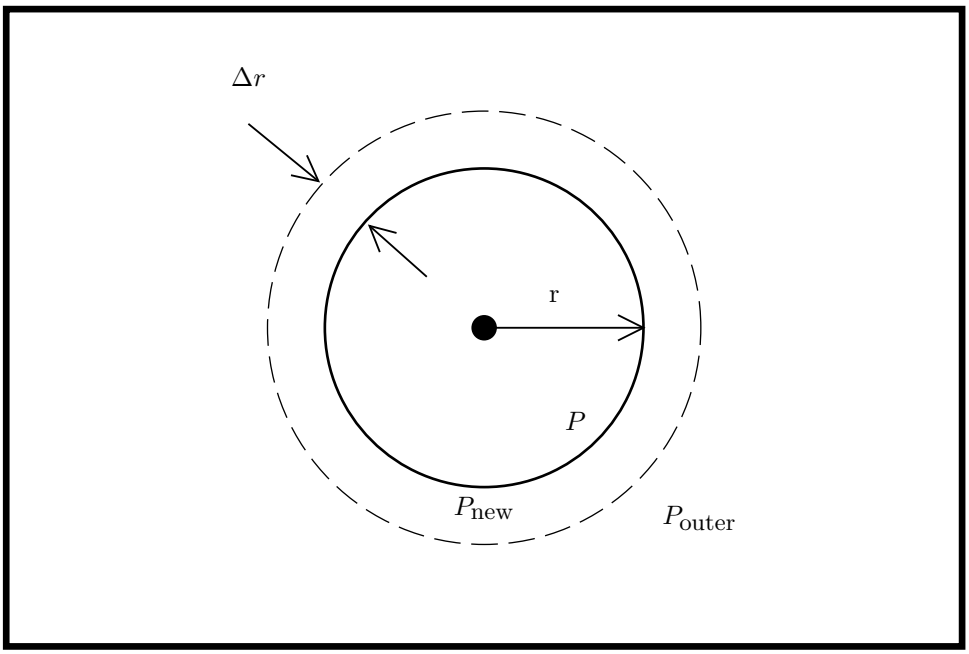


Figure 3.3: Model for predicting distribution of wire lengths. The system consists of a regular array of identical cells of area πr_c^2 each. P wires cross the perimeter at r, while P_{outer} cross the perimeter at $r + \Delta r$. The number of new pins appearing in the annulus is P_{new}.

The new pins that are created within the annulus are

$$P_{\text{new}} = \frac{2\gamma r \Delta r P_0}{r_0^2}$$

Lemma 3.1 We now compute the number of wires P_T crossing through the annulus.

$$P_T = \frac{1}{2}(P + P_{\text{outer}} - P_{\text{new}})$$

Proof: Let x equal the number of wires that enter the annulus though the inner perimeter and connect to new pins within the annulus. Similarly, let y equal the number of wires that originate on new pins within the annulus, and exit the region through the outer perimeter. Let P_T equal the number of wires that enter through the inner perimeter and exit through the outer perimeter. The following relations hold:

$$P = x + P_T \tag{3.11}$$

$$P_{\text{outer}} = y + P_T \tag{3.12}$$

$$P_{\text{new}} = x + y \tag{3.13}$$

Adding Equations 3.11 and 3.12, and subtracting Equation 3.13 yields

$$P + P_{\text{outer}} - P_{\text{new}} = 2P_T$$

from which the lemma follows directly. □

Substituting the values of P, P_{outer}, and P_{new} gives

$$P_T = P + \frac{Pb\Delta r}{r} - \frac{P_0 \gamma r \Delta r}{r_0^2}$$

The *fraction* of wires crossing though the annulus is then

$$F \equiv \frac{P_T}{P} = 1 + \frac{b\Delta r}{r} - \frac{\gamma^{1-b} r^{1-2b} \Delta r}{r_0^{2-2b}}$$

If $\eta(r)$ is the number of wires from the central element that are still in existence a distance r from that element, a recurrence relation for η exists:

$$\begin{aligned}
\eta(r+\Delta r) &= \eta(r)F \\
&= \eta(r)\left(1 + \left(br^{-1} - \frac{\gamma^{1-b} r^{1-2b}}{r_0^{2-2b}}\right)\Delta r\right) \\
&= \eta(r) + \eta(r)\left(br^{-1} - \frac{\gamma^{1-b} r^{1-2b}}{r_0^{2-2b}}\right)\Delta r
\end{aligned}$$

Comparing this expression to the Taylor series expansion of $\eta(r+\Delta r)$

$$\eta(r+\Delta r) = \eta(r) + \eta'(r)\Delta r + O(\Delta r^2)$$

yields the nonlinear differential equation

$$\eta'(r) = \eta(r)\left(br^{-1} - \frac{\gamma^{1-b} r^{1-2b}}{r_0^{2-2b}}\right)$$

This equation can be solved by noting that $\frac{\partial}{\partial r}\log\eta(r) = \frac{\eta'(r)}{\eta(r)}$. Hence,

$$\begin{aligned}
\int_{r_0}^{r} \frac{\partial}{\partial r}\log\eta(r)\,\partial r &= \int_{r_0}^{r}\left(br^{-1} - \frac{\gamma^{1-b} r^{1-2b}}{r_0^{2-2b}}\right)\partial r \\
\log\eta(r) - \log\eta(r_0) &= b\log r - \frac{\gamma^{1-b}}{(2-2b)r_0^{2-2b}} r^{2-2b}\Big|_{r_0}^{r}
\end{aligned}$$

The final solution is then

$$\eta(r) = A(r_0) r^b \exp\left(-\frac{\gamma^{1-b}}{2-2b}\left(\frac{r}{r_0}\right)^{2-2b}\right)$$

which we write, for the sake of normalized distance measures, as

$$\eta(r) = A_0\left(\frac{r}{r_0}\right)^b \exp\left(-\frac{\gamma^{1-b}}{2-2b}\left(\frac{r}{r_0}\right)^{2-2b}\right) \tag{3.14}$$

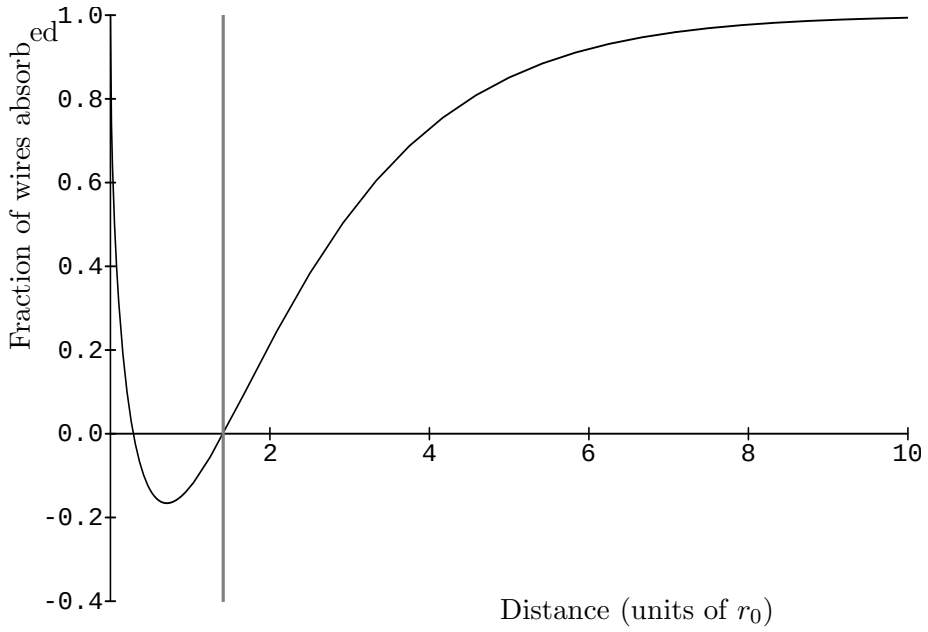


Figure 3.4: Predicted probability of wire absorption with distance. The vertical line marks the point $r_c = r_0/\sqrt{\gamma}$. The curve is for $\gamma = 0.5$, $b = 0.5$. The horizontal scale is in units of r_0.

The fraction of wires that continue a distance r from the original element is then

$$f(r) = 1 - \eta(r)$$

where $f(r_c) \equiv 1$, and r_c is the size of the "unit cell" of the tiled elements. In particular, for an area utilization γ and an element radius r_0, we have

$$\gamma \pi r_c^2 = \pi r_0^2 \quad \Rightarrow \quad r_c = \frac{r_0}{\sqrt{\gamma}}$$

Then,

$$f(r_c) = 1 \quad \Rightarrow \quad A_0 = \left(\frac{r_c}{r_0}\right)^{-b} \exp\left(\frac{\gamma^{1-b}}{2-2b}\left(\frac{r_c}{r_0}\right)^{2-2b}\right)$$

Figure 3.4 plots $f(r)$ for typical values of γ and b.

At this point, we can confirm that the model yields a plausible distribution function (i.e., one that is monotonic, equal to 0 at r_c and has a limiting value of 1 for large r). In fact, it is easy to show that f satisfies these requirements for all values of γ and b. The function f reaches a minimum at $r_{\min}$, given by

$$\frac{\partial f}{\partial r} = 0 \quad \Rightarrow \quad b\gamma^{b-1} = \left(\frac{r_{\min}}{r_0}\right)^{2-2b} \tag{3.15}$$

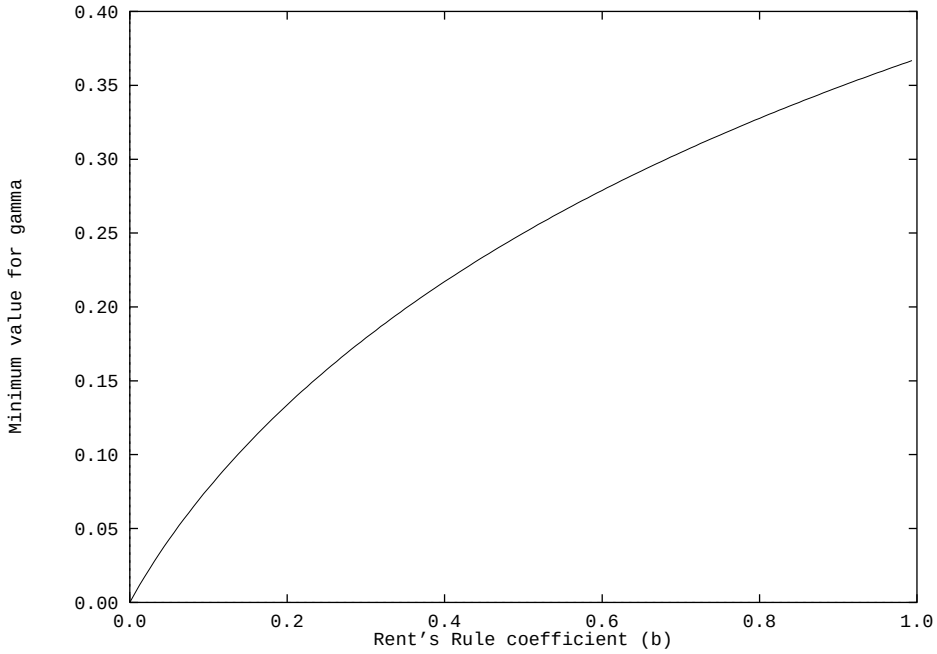


Figure 3.5: Range of application of restricted model. If we only permit wires to terminate outside the "unit cell," the restriction on γ disappears.

We can substitute $r_0 = \sqrt{\gamma} r_c$ into Equation 3.15 to obtain

$$\left(\frac{r_{\min}}{r_c}\right)^{2-2b} = b \tag{3.16}$$

Since $b < 1$, we have $r_{\min} < r_c$ always, and the model is applicable for all values of γ and b.

If the model is modified to $f(r_0) = 1$ (i.e., we permit wires from a component to terminate *within* the "unit cell" surrounding the component), the monotonicity constraint requires the function f to reach a minimum at a point $r_{\min}$ that is less than r_0. Consequently, Equation 3.15 requires $\gamma > b^{\frac{1}{1-b}}$. This expression is plotted in Figure 3.5.

Armed with the function $f(r)$, we can now compute the probability density function $p_f(r)$ that specifies the probability that a wire terminates between a distance r and $r + \delta r$:

$$p_f(r) = \frac{\partial f(r)}{\partial r} \tag{3.17}$$

The average length of the wires connected to a component is then given by

$$\langle w \rangle = \int_{r=r_c}^{\infty} r\, p_f(r)\, dr \tag{3.18}$$

It is instructive to plot the integrand (Figure 3.6). We observe that integrating the function in the region $0 < r < r_c$ would contribute only a small error to $\langle w \rangle$. Figure 3.7 illustrates

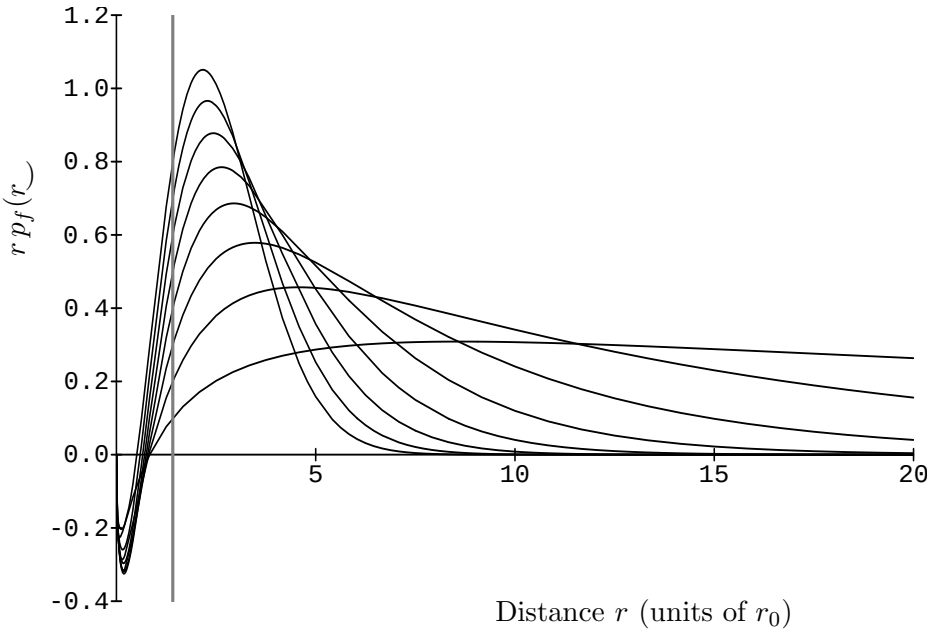


Figure 3.6: Integrand of $\int_{r=r_c}^{\infty} r\, p_f(r)\, dr$, plotted for $\gamma = 0.5$, and b ranging from 0.2 to 0.9 (in increments of 0.1). The vertical line represents the edge of the unit cell.

the relative error introduced by making the approximation

$$\langle \tilde{w} \rangle = \int_{r=0}^{\infty} r\, p_f(r)\, dr$$

One advantage of making this approximation is the existence of a closed form solution for $\langle \tilde{w} \rangle$, that we obtain from Equations 3.14 and 3.17 by applying ([Gradshteyn and Ryzhik, 1980] Equation 3.478.1)

$$\int_0^{\infty} x^{\nu-1} e^{-\mu x^p}\, dx = \frac{1}{p} \left(\mu^{-\frac{\nu}{p}}\right) \Gamma\left(\frac{\nu}{p}\right)$$

Giving

$$\begin{aligned} \langle \tilde{w} \rangle \quad = \quad & \frac{A_0}{2-2b} \left(\frac{\gamma^{1-b}}{r_0^{2-2b}} \left(\frac{\gamma^{1-b}}{(2-2b) r_0^{2-2b}} \right)^{-\frac{3-b}{2-2b}} \Gamma\left(\frac{3-b}{2-2b}\right) \right. \\ & \left. -b \left(\frac{\gamma^{1-b}}{(2-2b) r_0^{2-2b}} \right)^{-\frac{b+1}{2-2b}} \Gamma\left(\frac{b+1}{2-2b}\right) \right) \end{aligned} \tag{3.19}$$

This approximation is plotted in Figure 3.8 for several typical parameter values.

In addition to its intrinsic physical plausibility, the model that wires only terminate *outside* the "unit cell" has the advantages of providing an accurate closed form approximation to the average wire length, and of removing any constraints on the values of the model parameters γ and b.

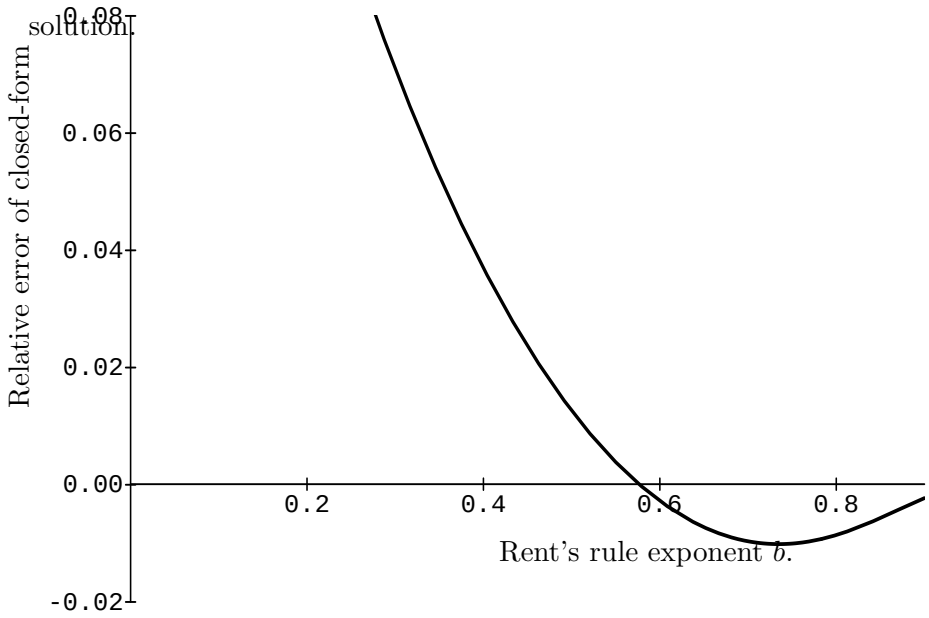


Figure 3.7: Relative error introduced by integrating model from $r = 0$. This curve is computed from numerical integration of Equation 3.18 and comparison to the closed form expression of Equation 3.19. This error is independent of γ.

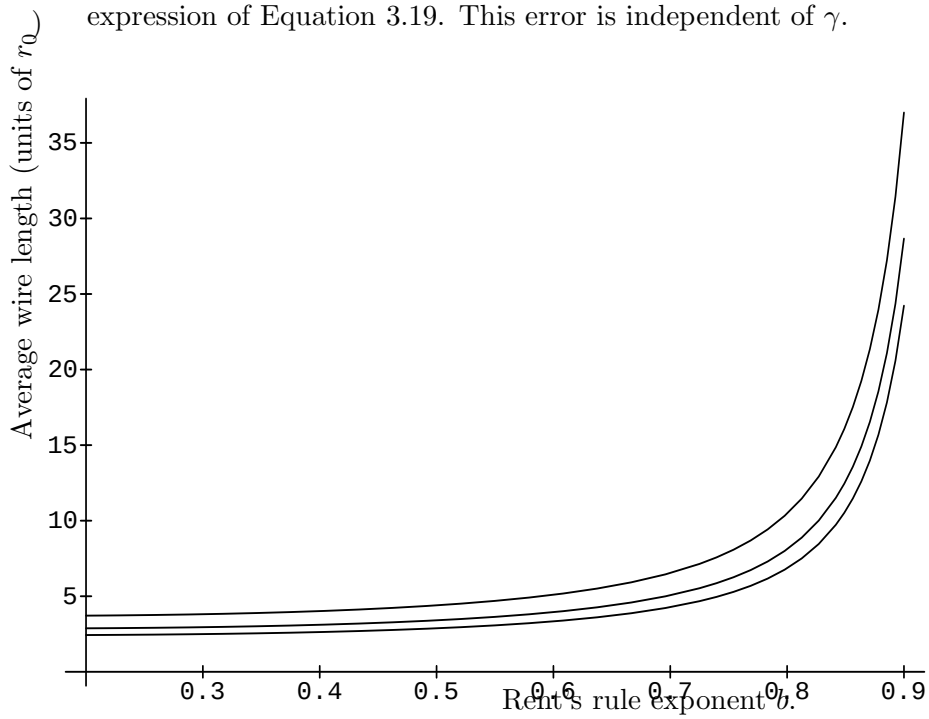


Figure 3.8: Average wire length predicted by model. The wire length is in units of r_0, and is plotted as a function of the Rent's rule coefficient $b.$ for $\gamma = 0.3$, 0.5, and 0.7.

3.6.1 Area Limits on Average Wire Length

For a regularly tiled array of cells, the area of wire connected to each cell must be *less* than the area of the cell itself. This result is obtained by "folding back" all wires from a particular cell onto the cell itself, and can be expressed as

$$\sum_{L=0}^{L_{\max}} W\, L\, n(L) = A$$

where W is the wire *pitch* (the minimum center-to-center distance between two wires), L is the length of a given wire, and $n(L)$ is the number of wires of that length [Mead, 1990]. In the continuous limit, the summation is replaced by an integral using the probability distribution function of Equation 3.17. If we attribute each wire equally to the two cells it connects, and we substitute $A = \pi r_c^2$, we obtain

$$\frac{W P_0 \langle w \rangle}{2} < \pi r_c^2$$

Giving

$$\langle w \rangle < \frac{2\pi r_0^2}{\gamma W P_0}$$

3.7 Summary

In this chapter, we have developed a simple wiring model based on an analysis of the intrinsic dimensionality of the interconnection medium. The essential assumption we made is that components are uniformly distributed throughout the spatial extent of the system. This postulate leads directly to the result that a power law relates available wiring density to component count. We have presented this result as a specific case of Rent's rule, a general empirical wire partition relation.

Based on Rent's rule, we have derived a closed-form approximation to the average wire length for cells in regular arrays. Finally, we have reintroduced physical wiring constraints to obtain yet another upper bound on the connectivity of cells.

Chapter 4

A Dynamically Configurable Architecture for Prototyping of Analog Circuits[1]

One of the symptoms of an approaching nervous breakdown is the belief that one's work is terribly important.

Bertrand Russell (1872–1970)

4.1 Introduction

Even with fast-turnaround fabrication, a considerable fraction of prototyping lead-time is consumed by the VLSI fabrication step. This delay is particularly expensive when we consider prototyping subcircuits, or exploring new circuit structures and logic forms as a prelude to a large design project. The fabrication step is beyond the control of the designer, raising technical issues of yield and parametric behavior, as well as the logistic issues of cost, time, and the need to interface with an external vendor.

This chapter presents the design of a VLSI prototyping platform, in which the PROTOCHIP, a field-reconfigurable integrated circuit, is used to physically wire up the circuit

[1]Portions of this chapter have been previously published. See [**?**].

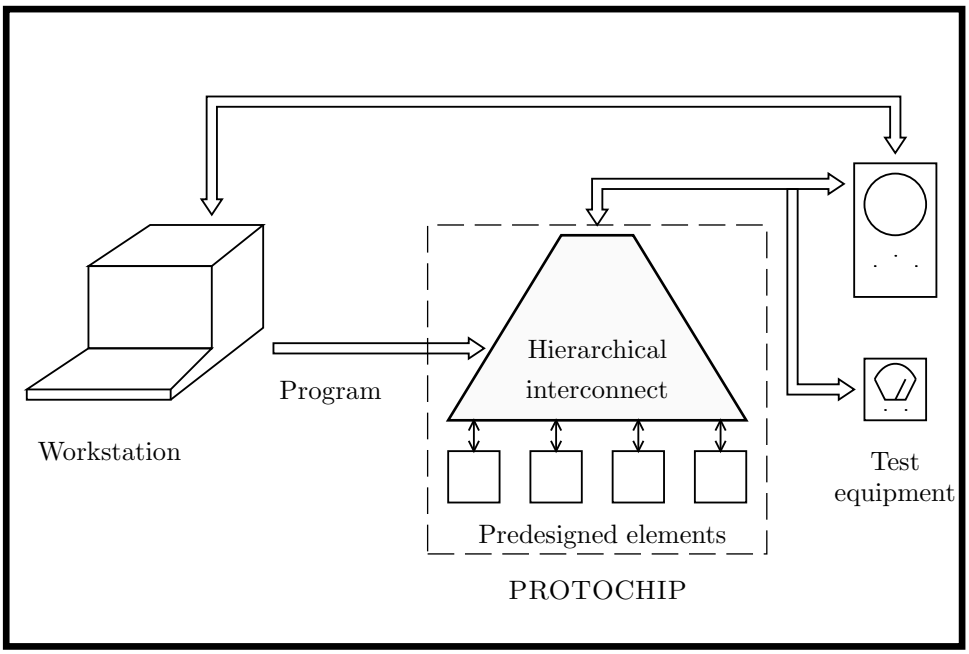


Figure 4.1: The PROTOCHIP environment. We program the PROTOCHIP by interconnecting the elements into a circuit that is then tested electrically.

to be tested. The PROTOCHIP is particularly intended for synthesis and testing of analog neural-network architectures [**?**], which generally require large numbers of active elements, but usually with relaxed precision requirements. These demands make neural networks a poor match for conventional software simulators. In addition to the obvious advantages of a hardware accelerator and simulator for fast prototyping, this system provides the first opportunity to attain a truly integrated design–fabricate–test environment.

The basic structure of the PROTOCHIP is a set of circuit elements arranged within a hierarchy of interconnect switches, facilitating the synthesis of the desired circuit. This programming process is controlled in the field by a workstation (Figure 4.1). Because this chip can effectively be considered an extension of a design-tool suite, similar performance criteria are applicable:

1. **Universality.** Given that the circuits to be prototyped generally are designed *after* the PROTOCHIP has been fabricated, can the user be reasonably assured the system is capable of testing a usefully wide range of circuits?

2. **Functionality and performance.** Do the prototyped circuits function correctly? How useful is the tool as a simulator?

3. **Programmability.** How directly can the PROTOCHIP be configured into the desired circuit for testing?

4. **Scalability.** Can the prototyping-chip architecture exploit the additional capacity of improving VLSI processes? In particular, can we exploit hierarchical architectures and

modular programming algorithms, in order to remain efficient as transistor densities and counts increase.

5. **Flexibility.** Can the design of the PROTOCHIP itself be usefully adapted for synthesizing particular classes of circuit networks? The system can be custom-tailored at *two* levels: the basic circuit elements and interconnect structure are bound at fabricate time, and the actual PROTOCHIP is configured electrically in the field at runtime.

This chapter concentrates on three aspects of the PROTOCHIP's design. First, an analysis of a general model for circuit connectivity is used to explore the requirements on the interconnect matrix (Section 4.2). The case for a hierarchical architecture is presented, based on a desire to minimize these wiring resources, as are some general observations regarding the organization of large systems. Second, two interconnection switch technologies are discussed, and their relative merits are compared (Section 4.5). Third, the structure and design of the fabricated PROTOCHIP is discussed, and test results are presented (Sections 4.6and 4.7). We defer discussion of algorithms for embedding circuits onto the PROTOCHIP to Chapter 5.

4.2 Connectivity of Circuits

The general form of the prototyping chip is a collection of functional elements interconnected by a programmable switch matrix. This matrix is responsible for both the great flexibility of the system *and* also is that system's most severe constraint. Because the design of the switch matrix is bound when the PROTOCHIP is fabricated, it needs to accommodate unanticipated circuits, while not incurring a prohibitive area penalty.

To this end, it is highly instructive to examine some general interconnect properties of circuits. This section uses Rent's rule as a connectivity model to investigate various wiring strategies. The ultimate objective is the development of a hierarchical interconnect matrix.

It is important to distinguish between topology and structure; the former deals exclusively with connectivity, the latter adds placement and layout constraints. The high cost of wiring in VLSI circuits makes consideration of structure imperative, as do several objectives for particular networks (e.g., an imaging array should be regularly placed); discussion of structure is, however, deferred to Section 4.6.2.

The least restrictive model of circuit connectivity requires every element to be connectable to any other. Defining N to be the number of elements permitted on the chip, and P_L to be the number of ports (pins) on each of these *leaf* elements, the required number of switches is

$$N_S = P_L^2 N^2$$

Of course, the requirement that any pin connect arbitrarily to any other pin (and possibly

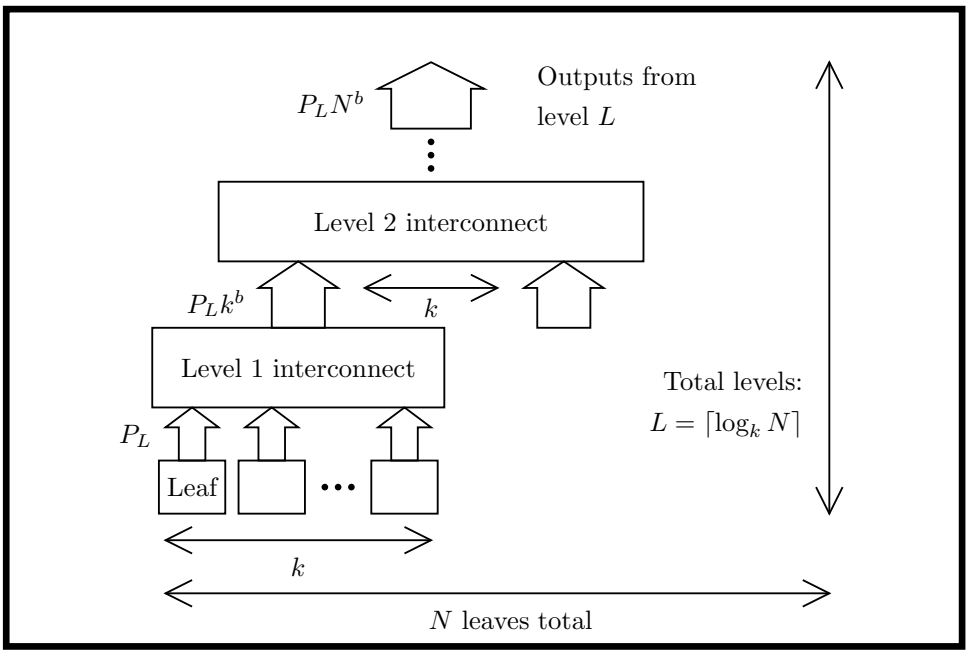


Figure 4.2: Hierarchical interconnect parameters.

to any number of other pins) requires the full generality of a crossbar switch, with its associated quadratic overhead. We define a useful measure of circuit-wiring complexity to be

$$\beta = \frac{N_S}{P_L^2 N}$$

which can be thought of as the number of other *elements* to which a particular element must fan out. For the crossbar,

$$\beta_{\text{crossbar}} = N$$

Now consider an interconnect chip with N elements. If the interconnect is flat (nonhierarchical), applying Rent's rule (Equation 3.1) once gives the following number of outputs:

$$P_{\text{out}} = P_L N^b$$

The interconnection matrix can then be realized by a crossbar with $P_L N$ inputs and P_{out} outputs. The total number of switches is then

$$N_S = P_L^2 N^{b+1} \;\Rightarrow\; \beta = N^b \tag{4.1}$$

which, for large N, represents a sizable improvement over the full-crossbar interconnect.

Another approach to connecting the N elements is incremental in nature: Suppose k elements are connected simultaneously with a crossbar switch; then, k of these meta-elements are connected. Repeating this procedure gives rise to the k-ary tree in Figure 4.2, of height $L = \log_k N$.

The number of switches required to implement the ith level of interconnect is then given by

$$\begin{aligned} N_{S_i} &= (\text{inputs to each matrix at level } i) \cdot \\ &\quad (\text{outputs from level } i) \cdot (\text{number of } i\text{th level matrices}) \\ \text{Level 1: } N_{S_1} &= (kP_L)(P_L k^b)\left(\frac{N}{k}\right) \\ \text{Level 2: } N_{S_2} &= \Big(k(P_L k^b)\Big)\Big((P_L k^b)k^b\Big)\left(\frac{N}{k^2}\right) \\ \text{Level } i\text{: } N_{S_i} &= kP_{i-1}^2 k^b \left(\frac{N}{k^i}\right) \end{aligned}$$

where

$$P_i = \text{outputs of matrix at level } i = \begin{cases} P_L k^b & \text{if } i = 1 \\ k^b P_{i-1} & \text{otherwise} \end{cases}$$

The solution to this recurrence is $P_i = P_L k^{bi}$, which gives the number of switches at level i to be

$$N_{S_i} = P_L^2 k^{b+1} k^{2b(i-1)} \frac{N}{k^i} = P_L^2 N k^{(2b-1)i} k^{1-b}$$

The total number of switches is the sum over all levels in the hierarchy:

$$\begin{aligned} N_S &= \sum_{i=1}^{\log_k N} P_L^2 N k^{1-b} k^{(2b-1)i} \\ &= P_L^2 \frac{k^b}{k^{2b-1}-1}(N^{2b} - N) \end{aligned}$$

Giving

$$\beta = \frac{k^b}{k^{2b-1}-1}(N^{2b-1} - 1)$$

Now $2b-1 \leq b$ for all $b \in [0,1]$, and, asymptotically, this expression gives us fewer switches than does Equation 4.1.

In addition, an interesting case results from $b = 0.5$, which yields the greatest savings

$$\beta = \frac{\sqrt{k}}{\ln k} \ln N$$

It is useful to consider some typical numbers. The number of switches for the hierarchical interconnect compared to the flat Rent's rule interconnect is

$$\frac{\beta_{\text{hier}}}{\beta_{\text{flat}}} = \frac{N^{b-1} - N^{-b}}{k^{b-1} - k^{-b}} = \frac{N_{S_{\text{hier}}}}{N_{S_{\text{flat}}}} \tag{4.2}$$

Equation 4.2 is plotted in Figure 4.3 for $k = 2$.

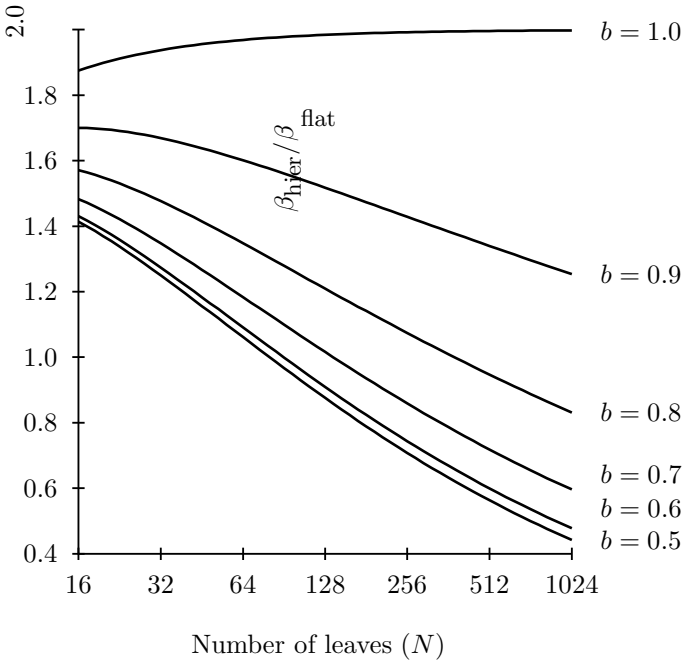


Figure 4.3: Number of switches in hierarchical interconnect relative to flat Rent's rule interconnect. Equation 4.2 is plotted for $k = 2$.

We can obtain further gains by interleaving processing elements with the interconnect hierarchy. In general, for a tree with branching factor k, the addition of one fixed fanout element at each interconnect node yields

$$\beta = \beta_{\text{hier}} \frac{k-1}{k}$$

where β_{hier} is the quantity computed previously (and N is the number of leaves at the *bottom* of the tree).

4.3 Hierarchy and Abstraction in System Design

In addition to assisting in the management of the basic issues of wiring complexity and length, the hierarchy in the PROTOCHIP can be used to take advantage of system-level organization in the network to be embedded. In particular, the top-down design style expounded in structured design methodologies [**?**] can be exploited; we can simplify greatly the mapping from design to silicon by using for interconnect the same hierarchical specification present in the functional description of the design. In short, most circuits that we wish to prototype are hierarchical, and therefore the general PROTOCHIP architecture uses hierarchy as well. Furthermore, such circuits generally are easily scalable to larger systems, as is the PROTOCHIP architecture itself. Several other issues concerning hierarchy in system synthesis are discussed in the literature [**?**].

We chose a hierarchical design for three other reasons. First, the electrical properties of the resulting circuits are improved, due to (1) lower capacitances, because the wires are shorter and there are fewer switches, and (2) the ability (and space, as will be shown in Section 4.6.2) to place buffer stages within the interconnect itself. This possibility of integrating processing with interconnect also is appealing for networks that are inherently specified recursively (as are, e.g., many neural-network architectures). Second, the partitioning of a circuit into a hierarchy permits (1) nonuniform hierarchies to be tailored a priori to specific architectures (e.g., neural-networks, processing surfaces, systolic arrays), (2) high-density special cells to be incorporated, and (3) subtle embeddings of subcircuits to be stored in libraries and easily integrated into subsequent designs. Third, hierarchy simplifies the embedding problem by providing a convenient interface with CAD tools, most of which encourage or enforce a hierarchical style. In addition, most interchange formats (CIF, NTK, EDIF, etc.) intrinsically support hierarchy.

4.4 Embedding a Circuit Graph

The ultimate workability of this prototyping system is determined by the ease with which the desired circuit graph can be embedded onto the interconnect matrix. Basically, there are four approaches to this problem:

1. Exploit the hierarchy present in the circuit specification as much as possible. When this approach results in an unroutable system, resort to one of the other approaches to solve the embedding of the problem spot.

2. Exploit structural information available from the design tool. For example, a schematic editor also identifies approximate relative placement of subcircuits; schematics usually are drawn with similar objectives (e.g., minimal global and random wiring) as required by the embedding algorithm.

3. Ignore any hierarchy in the specification, and attempt to synthesize a partition of the circuit graph that can be embedded directly.

4. Accept a user-provided interactive routing strategy.

We have developed an algorithm based on the third approach; it is described in detail in Chapter 5. The network-embedding compiler accepts a flat circuit graph, and generates a hierarchical decomposition, with the provision for optional human assistance. Alternately, the explicit hierarchy present in the circuit specification can be followed, and embeddings for duplicated subcells need only be computed once. The addition of a final global optimization pass would be a worthwhile improvement.

4.5 Interconnect Switches

By far the most expensive components of the PROTOCHIP, in terms of area and performance, are the switches in the interconnect matrix. In this section, we consider three degrees of freedom of field-reconfigurable systems:

1. Are we able to reconfigure the system? That is, can the same silicon die be reused for multiple applications? This flexibility is often achieved at the expense of requiring large switches or special processes.

2. How volatile are the switch settings? Does the device need to be reprogrammed after power interruption? Again, special processing may be required, leading to additional expense.

3. What is the density of interconnect switches? The size of the interconnect ultimately determines the scale of the system we can construct, but must be traded off against process cost and yield.

Each of these compromises has a direct effect on the ease of manufacture—and hence the yield efficiency and cost—of the system. As an example, a high-density nonvolatile reconfigurable system could be implemented using an "E-squared" process, at the additional expense of special processing steps to manufacture the tunneling oxides.

4.5.1 Reconfigurable Interconnect Switches

For the first PROTOCHIP, CMOS transmission gates were used as the active switch elements, controlled by a static RAM cell based on a modified CSRL design [**?**] (Figure 4.4). The modification permits the CSRL to remain powered when a read operation is being performed. Two global signal lines broadcast the Q and $\overline{Q}$ data signals along the perimeter of the chip. The vertical decoder connects these lines to a single horizontal row; the horizontal decoder enables the access pass transistors of an entire column. If a read operation is being performed, transistor M_W remains ON, and the single selected cell drives its contents onto the global $Q/\overline{Q}$ bus, which is buffered at pads when it comes off-chip. If a write operation is requested, the vertical row decoder ANDs its output with the $\overline{\mathrm{R}}/\mathrm{W}$ request, and disables transistor M_W in the row addressed. The other power-down transistor is disabled by the horizontal column decoder. The cross-coupled inverter is then in an undriven state, and is loaded with $Q/\overline{Q}$ from the bus. The sequence is reversed to reapply power and to make the RAM cell static.

The size of the prototype switch is $39 \times 49\lambda$ (including RAM and pass-gate using $10 \times 2\lambda$ transistors), for a total area of $4300\mu^2$ ($\lambda = 1.5\mu$).

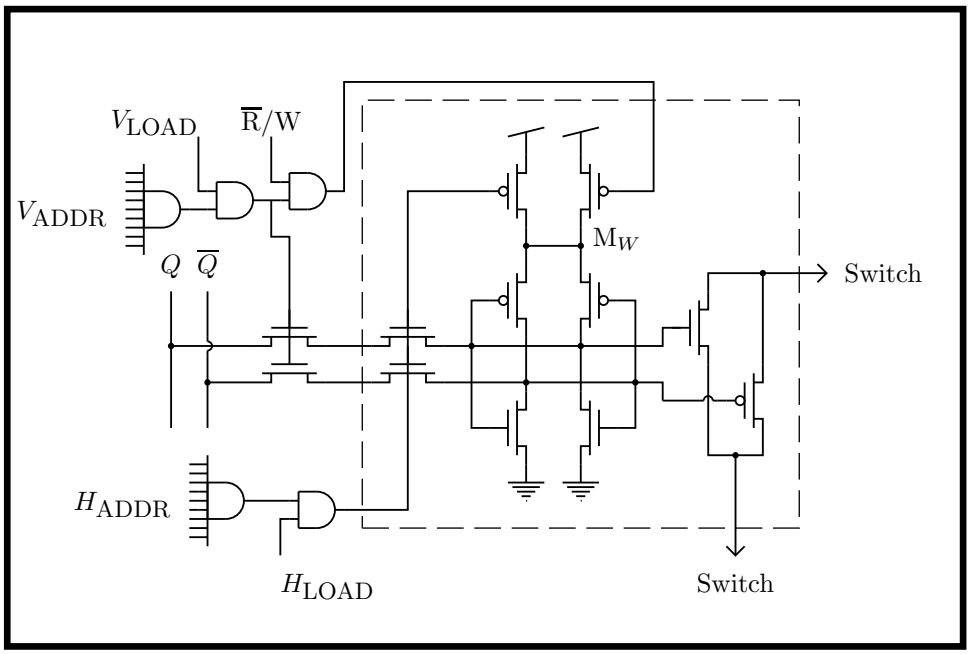


Figure 4.4: CSRL static RAM and transmission gate switch. The enclosed circuit is replicated at each interconnect site in Figure 4.10; the decoders are shared by the entire row or column.

4.5.2 Scaling of Transmission Gate Transistors for Maximum Linearity

In order to minimize the variation in interconnect resistance with operating point, it is important to scale the p- and n-channel MOS transistors appropriately within the transmission gate. To this end, it is useful to consider the resistance of the transmission gate for small differential voltages. For a differential voltage δ applied across a transmission gate with geometry W/L for the p-device, and $Z(W/L)$ for the n-device, with both devices operating in the above-threshold ohmic region, at a terminal voltage V, we have

$$I = Zk_n\left[(V_{\mathrm{dd}} - V - V_{T_n})\delta + \frac{\delta^2}{2}\right] + k_p\left[(V + \delta - V_{T_p})\delta - \frac{\delta^2}{2}\right]$$

The conductance is then

$$g \equiv \frac{\partial I}{\partial \delta} = Zk_nV_{\mathrm{dd}} - (Zk_nV_{T_n} + k_pV_{T_p}) + (k_p - Zk_n)(V + \delta) \tag{4.3}$$

To first order (ignoring the contribution of V to the threshold voltages V_{T_n} and V_{T_p}), setting $k_p - Zk_n = 0$ will remove the dependence of g on V. Note that this expression is valid for arbitrarily large values of δ, provided that the assumption of ohmic-regime operation is not violated. This large-signal effect will be evident in a later experiment (Figure 4.19), where the voltage drop across transmission gates in series is divided equally between those gates.

We can be even more effective, however, by incorporating a standard model for threshold dependence on substrate bias [Allen and Holberg, 1987]:

$$\begin{aligned} V_{T_n} &= V_{T_{n_0}} + \gamma_n(\sqrt{V+\phi_n} - \sqrt{\phi_n}) \\ V_{T_p} &= V_{T_{p_0}} + \gamma_p(\sqrt{V_{\rm dd} - V - \delta + \phi_n} - \sqrt{\phi_p}) \end{aligned}$$

where

$$\begin{aligned} \phi &= \frac{2kt}{q}\log\left(\frac{N_{\rm maj}}{n_i}\right) \\ \gamma &\quad \text{is in the range } [0.1, 1.0] \end{aligned}$$

Minimizing the effect of variations in V on Equation 4.3 gives

$$\frac{\partial}{\partial V}\left(\frac{1}{R}\right) = -Zk_n\gamma_n\frac{1}{2}(V+\phi_n)^{-1/2} - k_p\gamma_p\frac{1}{2}(V_{\rm dd} - V - \delta + \phi_p)^{-1/2} + k_p - Zk_n$$

Setting this derivative to 0, and solving for Z, gives

$$Z_{\rm min} = \frac{k_p}{k_n}\left(\frac{1 + \frac{\gamma_p}{2\sqrt{V_{\rm dd} - v - \delta + \phi_p}}}{1 + \frac{\gamma_n}{2\sqrt{V+\phi_n}}}\right)$$

Taking process parameters from a typical MOSIS process:

$$\begin{aligned} V_{T_{P_0}} &= 0.852 \text{ Volts} \\ V_{T_{N_0}} &= 0.938 \text{ Volts} \\ \phi_P &= 0.6 \text{ Volts} \\ \phi_N &= 0.6 \text{ Volts} \\ \gamma_P &= 0.5 \text{ V}^{1/2} \\ \gamma_N &= 1.109 \text{ V}^{1/2} \\ \kappa_N &= 2.15 \times 10^{-5} \text{ A/V}^2 \\ \kappa_P &= 9.22 \times 10^{-6} \text{ A/V}^2 \end{aligned}$$

and setting the derivative to 0 at $V = 2.5$ V, gives $Z_{\rm min} = 0.372$. The simple analysis, which ignored the back-gate effect, would set $Z_{\rm min} = k_p/k_n = 0.429$. The results are plotted in Figure 4.5.

4.5.3 A High-Density Nonvolatile Switch Technology

In keeping with our approach of fabricating using the MOSIS baseline CMOS process, we have pursued a different strategy for attaining the objectives of high switch density and low switch resistance: postfabrication laser configuration of a stock MOSIS die.

The work presented in this section is the result of a collaboration with Jack Raffel and Matthew Rhodes of the Restructurable VLSI group at M.I.T. Lincoln Laboratory. Their

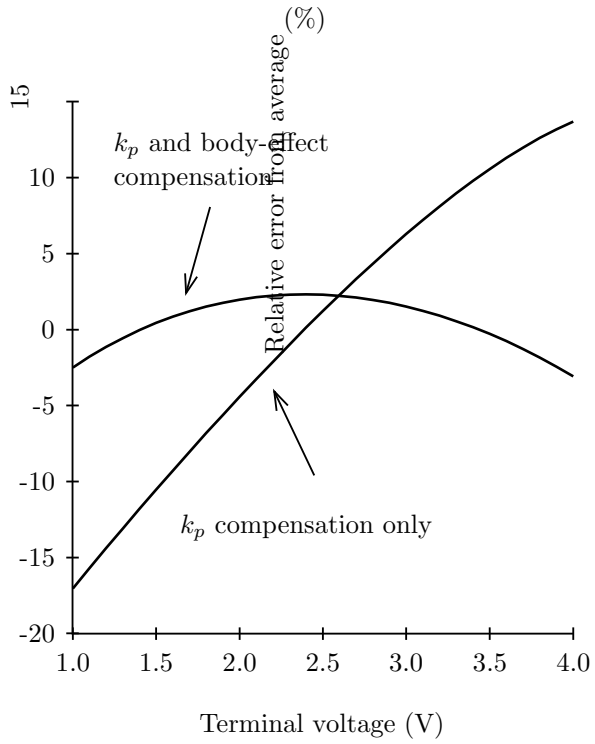


Figure 4.5: Transmission gate small-signal resistance variation. Zero-order compensation is obtained by scaling both devices inversely with their majority carrier mobilities (and hence k_p). A slightly different scaling can be used to minimize the variation caused by the first-order threshold voltage changes due to the body effect.

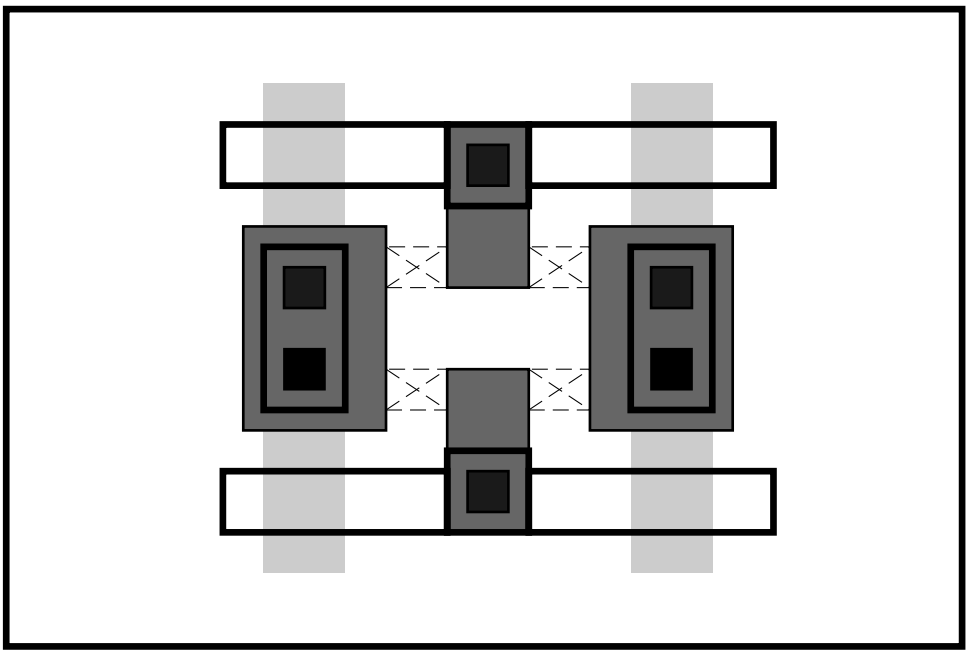


Figure 4.6: Layout of 4×4 laser-programmed switch matrix. The design rules are: laser-programming area = $3\mu \times 4\mu$, and spacing to other geometry = 2μ. The total area for the 4 switches is $18\mu \times 16\mu$.

process permits *cutting* of both first- and second-level metal lines, and *linking* of adjacent diffusion areas (separated by a minimum geometric design-rule spacing). The linking process consists of melting an area of the substrate, thereby permitting the dopant ions present in the degenerately doped implanted areas to diffuse together, producing a low-resistivity path (typically several hundred ohms per square). The attractive feature of this process is the ability to cut or link structures *without* the prior removal of the protective overglass from the die.

Because the PROTOCHIP architecture lends itself to programming by *closing* a few switches (rather than by opening a large number of previously closed switches—or shorted metal lines), and because the laser-programming fixture has a physical limit of approximately 10 reposition–zap cycles per second, the *link* approach was taken. Figure 4.6 illustrates the basic interconnect cell; four switches occupy an area of $18\lambda \times 16\lambda$, corresponding to $72\lambda^2$ per switch (compared to $1911\lambda^2$ per switch for the CSRL–transmission-gate switch of Section 4.5). The laser-link process has been tested down to $\lambda = 1\mu$. Such small switches could permit construction of PROTOCHIP systems with 4000 or more leaf cells on a standard MOSIS die.

To test these ideas, we fabricated a prototype analog leaf cell, and modified the netlist embedder to generate the laser-programming description data file [Garverick and Frankel, 1986]. Several circuits were tested; Figure 4.7 shows the results of programming the inverter of Figure 4.18. The transfer characteristic of the inverter is shown in Figure 4.8, and the

measured switch resistance is shown in Figure 4.9.

In summary, laser-configurable interconnect switches are attractive because they are non-volatile, and denser than reconfigurable RAM-based interconnect switches. Electrically, the performances of the two switch technologies are comparable, in terms of both resistance and capacitance. The primary disadvantages of the laser-programmed interconnect are the need for specialized programming equipment (laser, optical microscope, high-precision motor-driven stage, control software), the relatively slow rate of programming, and the "write-once" nature of the programming.

4.6 Compiling a PROTOCHIP

Three issues still need to be resolved: the specification of internal contents and organization of the leaf cells, the physical placement of leaves and interconnect on the silicon (the structure of the chip), and the implementation of a suitable interconnect element.

These issues have been deferred until now because they represent degrees of freedom (number and kind of leaves, matrix branching factor, and Rent's rule exponent) that are bound at *fabrication* time, whereas the network to be embedded is dynamically configured at runtime by the network compiler described in the previous section. This sequence opens the possibility of designing a metacompiler, with which a user fabricates a *custom* PROTOCHIP with parameters and components optimized for the end application. As specialized examples, leaf cells containing photoreceptors and simple analog circuits could be configured into image-processing surfaces [**?**, **?**], and leaf cells containing bit-serial adders and multipliers allow implementation of configurable data-flow architectures [**?**].

We have designed a simple metacompiler, using the WOL composition system and design tool [**?**]. The user provides a leaf cell, which must satisfy simple size constraints (both height and width must be integer multiples of certain constants), and hierarchical composition parameters (fanout at each level, which is more general than the Rent's rule model presented in Section 4.2). The system generates a PROTOCHIP ready for fabrication, and produces a programming file of topology parameters needed by the network-embedding compiler.

4.6.1 Leaf Cells

One approach to maximizing the versatility of the PROTOCHIP while maintaining a uniform hierarchy is to make the leaf cells themselves programmable. To illustrate this technique, we shall describe the design of a transistor-level leaf intended for prototyping low-level analog circuits and for designing new logic families, and consider a possible implementation of a leaf cell for constructing digital logic.

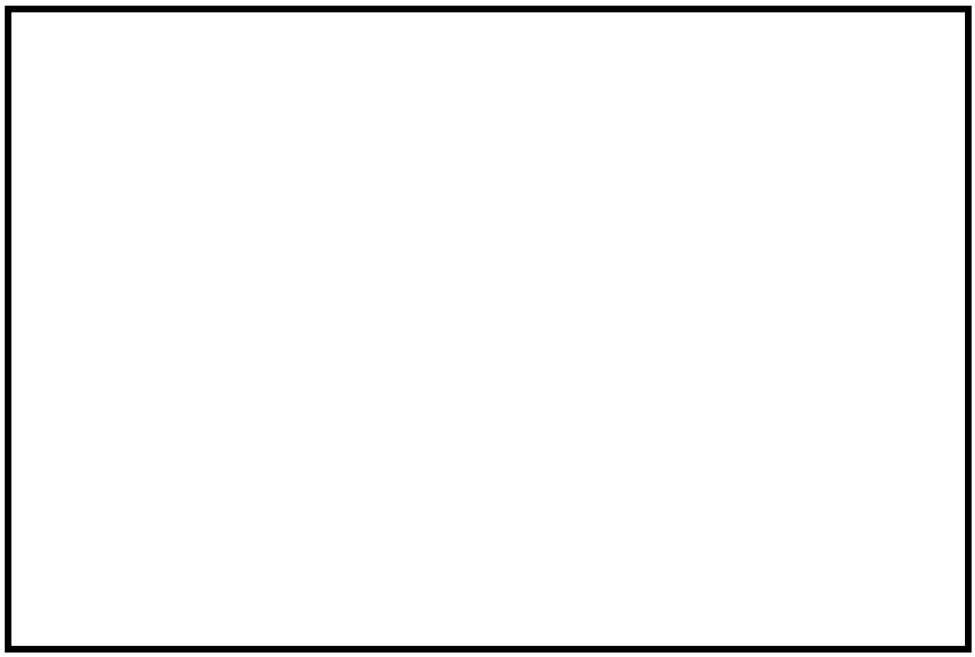


Figure 4.7: Photographs of laser-programmed interconnect matrix. The upper photograph shows the laser-controlled linking of adjacent diffusion areas; the lower photograph shows the same interconnect matrix, obtained by cutting metal2 lines.

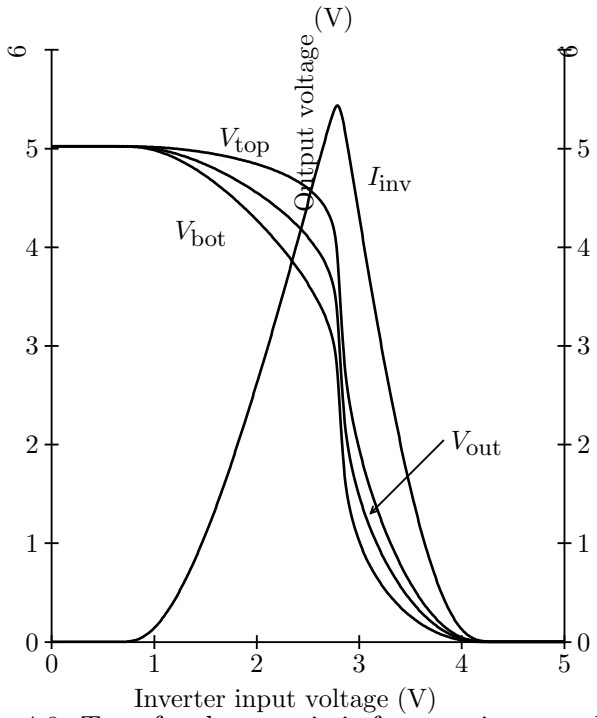


Figure 4.8: Transfer characteristic for poor inverter design of Figure 4.18. The inverter has two diffusion switches in series with high-current path.

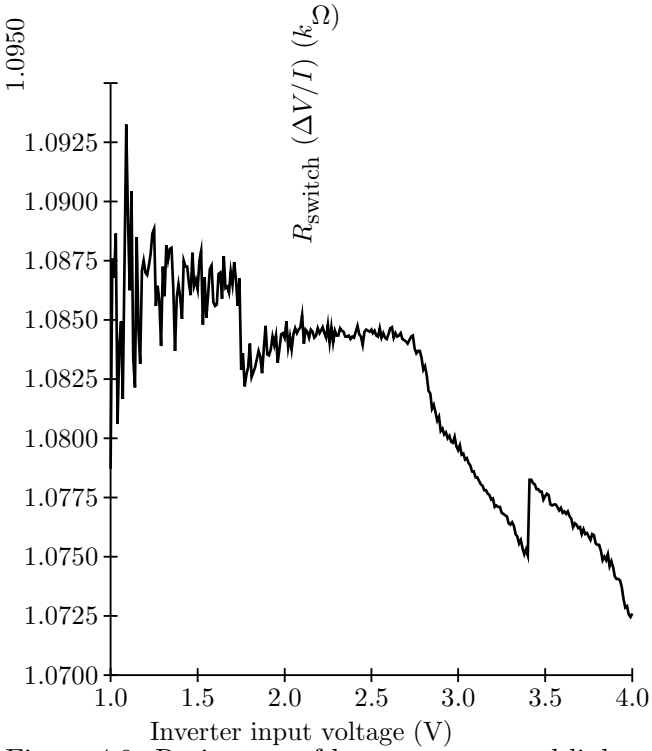


Figure 4.9: Resistance of laser-programmed link.

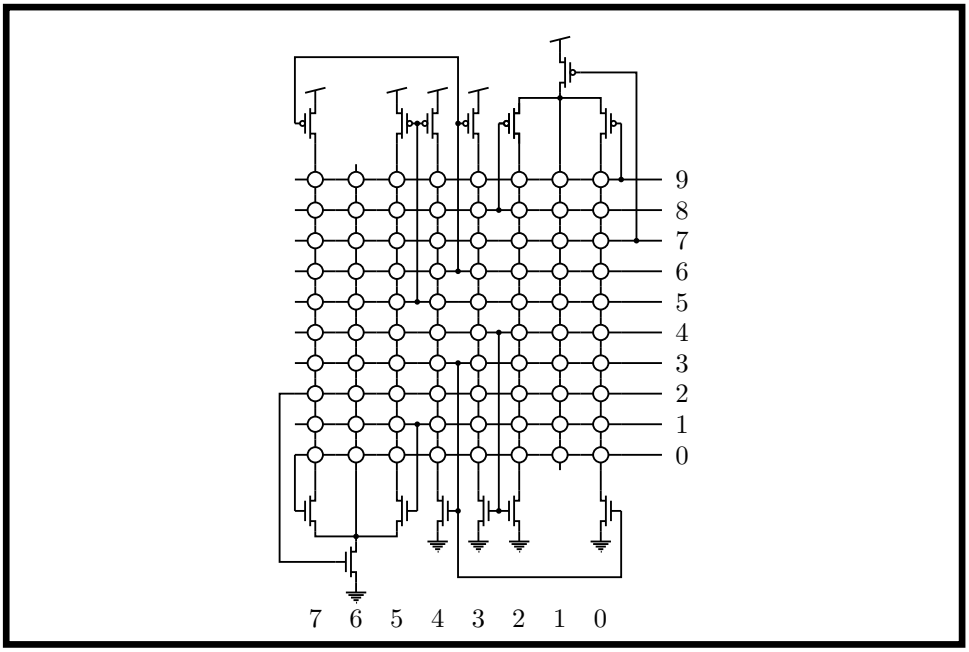


Figure 4.10: Programmable analog leaf cell. Each circle in the 10×8 matrix is a single programmable switch, which, when closed, connects the vertical and horizontal wires at that point.

Analog Leaf Cell

The analog leaf (Figure 4.10) consists of two blocks of partially preconnected n-transistors and p-transistors. Each programming switch connects a horizontal wire, which is connected to at least one transistor gate, to a vertical wire, which is connected to at least one of a p-drain and an n-drain. Commonly used analog building blocks, such as current mirrors and differential pairs, are prewired in an attempt to reduce the complexity and size of the programming matrix. If a full crossbar were used to interconnect seven three-terminal p-transistors to an equal number of n-devices, $(3 \times 7)^2 = 441$ switches would be required; with the merged devices, only 80 switches are used. Yet the compact design is sufficient to construct a rich set of analog circuits, including wide-range transconductance amplifiers (Figure 4.11), and horizontal resistors, as well as more conventional NAND/NOR and similar gates (Figure 4.12). In addition, a leaf cell is capable of configuring a fully independent floating three-terminal transistor.

If strictly digital combinational logic is desired, alternate approaches yield a higher density of gates per leaf. Our primary research interest has been the development of analog circuits, but the analog leaf can be usefully applied to the prototyping of new digital logic forms (for example, delay-insensitive asynchronous circuit elements or complementary set/reset logic (CSRL) [**?**]). In addition, the hierarchical architecture permits hybrid PROTOCHIPs, in which digital leaves containing gates or PLD-type devices are interleaved with analog elements.

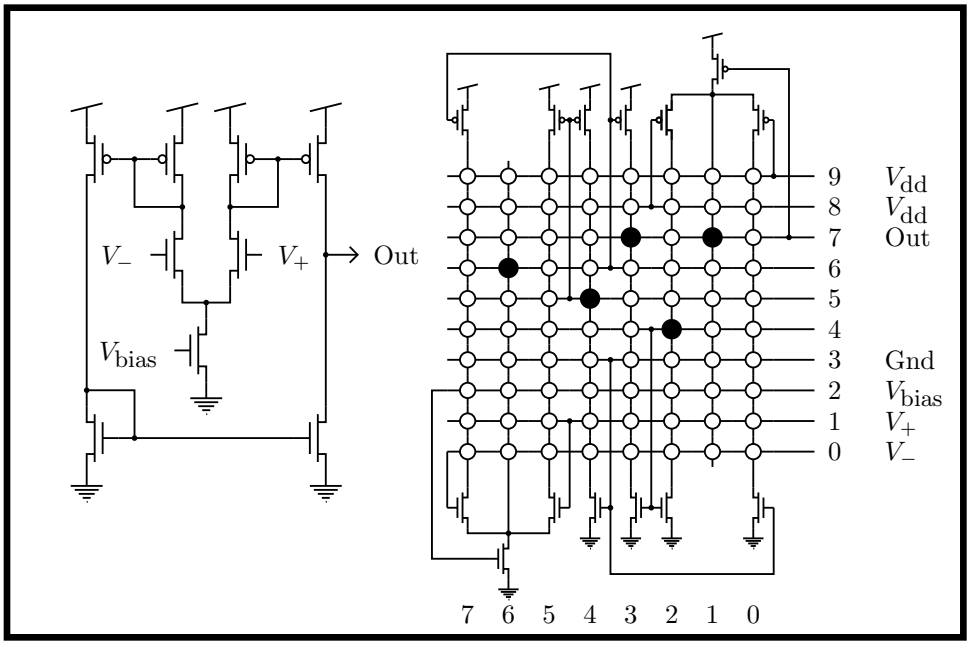


Figure 4.11: Wide-range transconductance amplifier. Applying the appropriate gate bias voltage disables the transistors that are not utilized.

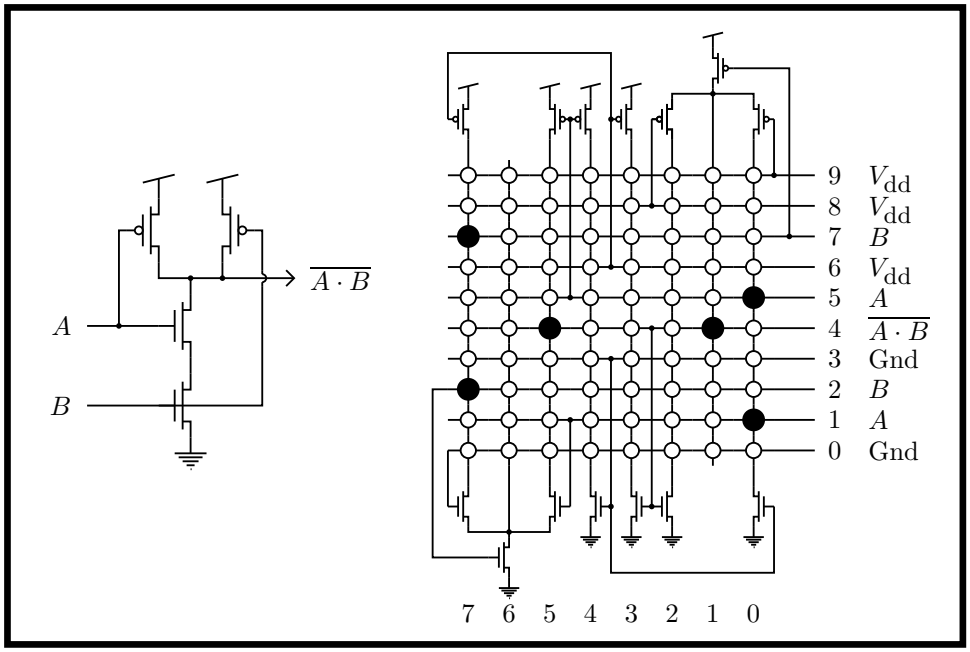


Figure 4.12: NAND gate implemented with analog leaf.

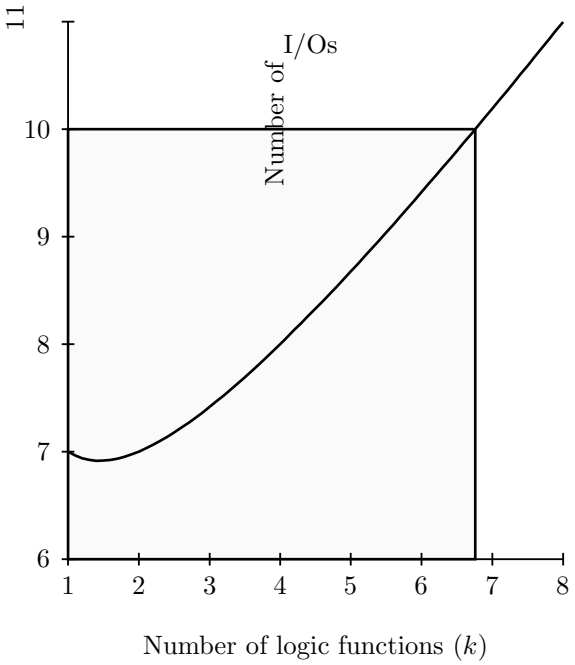


Figure 4.13: Tradeoff between inputs and outputs in digital leaf.

Digital Leaf Cell

In addition to the obvious technique of providing one (or more) hard-wired (nonconfigurable) gates (NAND, NOR, etc.) per cell, it is possible to exploit the programmable nature of the leaf cells. Instead of using the RAM array in the previous leaf to connect elements, we could use it for the explicit storage of Boolean truth tables. For a $H \times V$ array of storage (corresponding to $A = \log_2(HV)$ address bits or inputs to the logic gates), a simple tradeoff is possible between the number of Boolean functions (of the same inputs) that can be evaluated, and the number of inputs. Furthermore, we can make this tradeoff programmable (i.e., the N pins on the leaf cell could be dynamically divided between inputs and outputs), thereby making a wide variety of logic cells possible.

For k outputs, the maximum number of input bits is $A - \log_2 k$, and the total number of I/Os is then

$$P = A - \log_2 k + k$$

This expression is plotted in Figure 4.13 for a typical value ($A = 6$, corresponding to a 8×8 array of RAM); the flexibility of this approach is clearly demonstrated by the large variety of options that exists within the $P_L = 10$ pinout limit. This logic array has the additional advantage of being easy to lay out, for outputs numbering less than $\min(H, V)$, as it requires only a multiplexor to generate the combinations corresponding to the higher-order bits.

In addition to the obvious static logic applications of this cell (e.g., one cell can implement a one-bit full adder, equivalent to 9 three-input NOR gates), the possibility of external

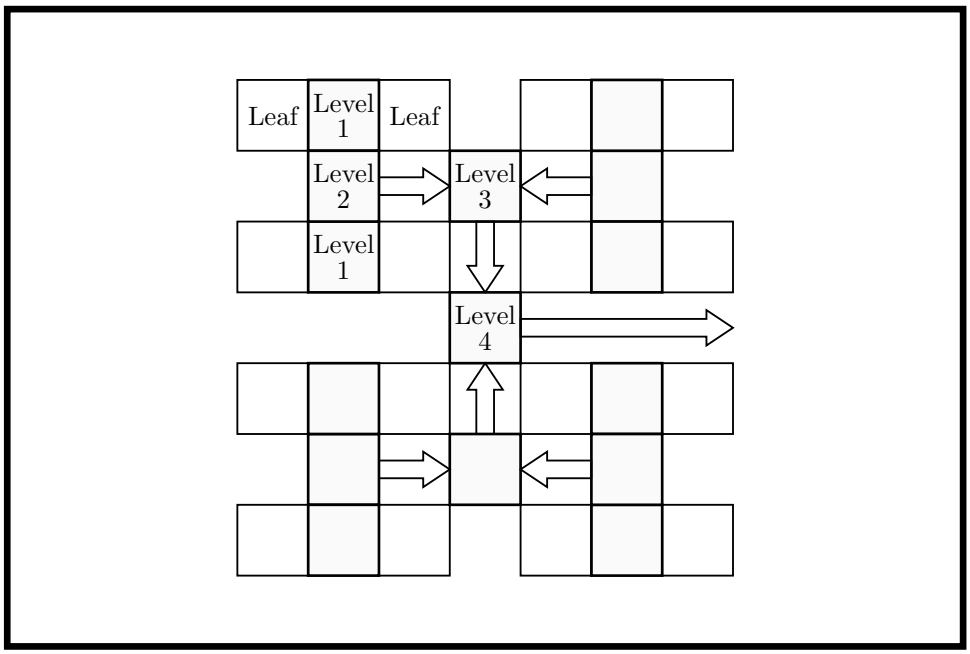


Figure 4.14: Binary H-tree hierarchical interconnect structure. The area of the silicon is divided between (1) leaf cells, (2) crossbar interconnect switches (shaded), (3) routing wires with optional buffer stages (arrows), and (4) space for processing elements interleaved in hierarchy.

feedback permits simple state machines (e.g., SR latches) to be constructed.

4.6.2 Physical Placement of Leaves and Interconnect

The first version of the PROTOCHIP contained a binary tree of interconnect laid out as an H-tree (Figure 4.14), in order to maximize the density of the switches and to simplify their programming. This structure permits the placement of the leaf cells on a regular grid. Address lines to set the switches are on a uniform mesh, with decoders located around the perimeter of the chip. We chose this approach over a hierarchical decoder because it is simpler. The mapping between hierarchical interconnect matrix coordinates and flat Cartesian coordinates is performed by the embedding compiler.

The first PROTOCHIP was fabricated on MOSIS run M75A, in 3μ CMOS; a photomicrograph is shown in Figure 4.15. Sixteen analog leaf cells (Figure 4.10) were included, with a leaf pinout of $P_L = 10$, and a Rent's rule exponent of $b = 0$. We chose this low fanout because the analog circuits of interest have a considerably smaller number of connections at the system level than do conventional digital circuits. The entire chip was generated recursively by the metacompiler.

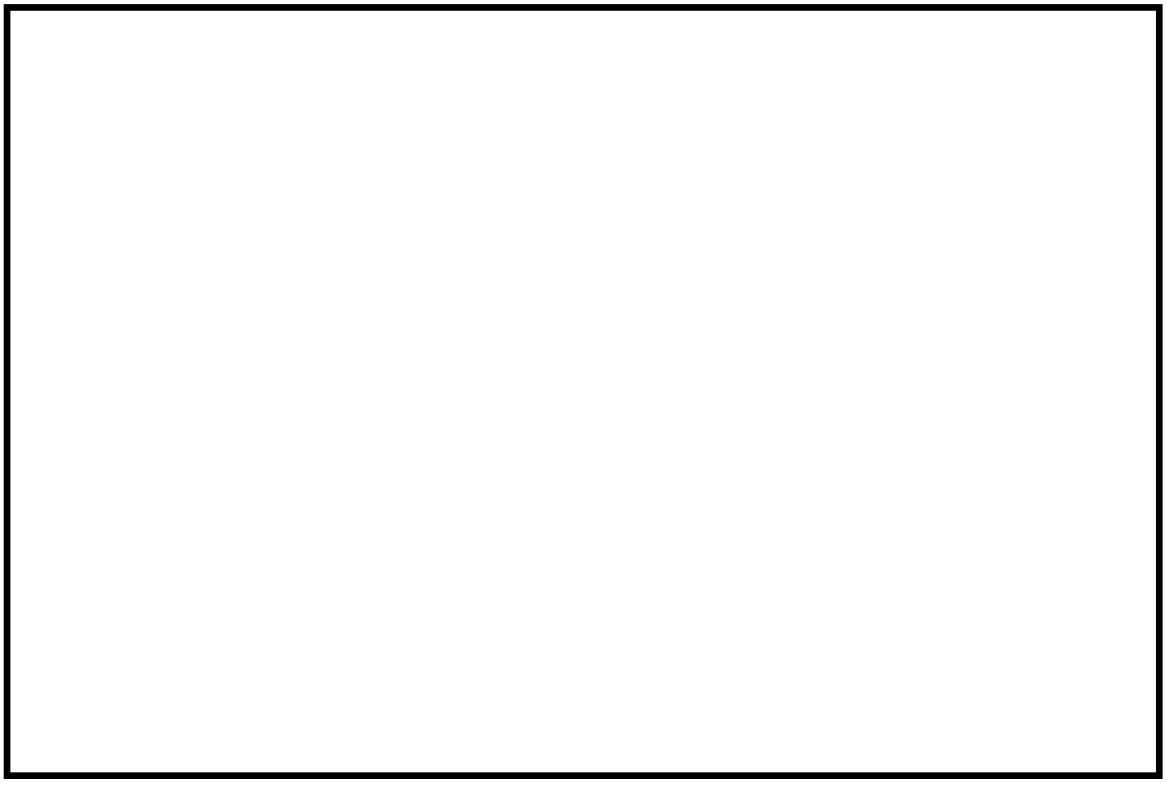


Figure 4.15: Photomicrograph of first PROTOCHIP.

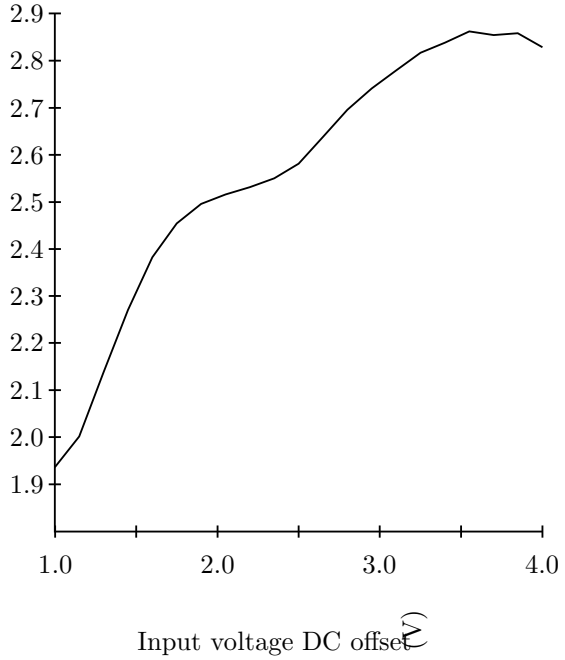


Figure 4.16: Transmission gate small-signal resistance. The experimental data describe the small-signal resistance of a scaled transmission gate. Unfortunately, we did not compensate for the shortening of the channel due to diffusion of the source–drain implants under the gate.

4.7 Experiments with the PROTOCHIP

Preliminary testing of the PROTOCHIP yielded encouraging results. To test the performance of the interconnect, we constructed a three-inverter ring oscillator; it ran at just over 2 MHz, with 10 pF of external capacitive load applied to all three internal electrical nodes. The experimentally measured capacitance of the interconnect was 500 fF/switch, due primarily to the diffusion sidewall capacitance of the transmission-gate drains. The ON resistance of the switches was 2.5 kΩ; the resistance as a function of terminal voltage is shown in Figure 4.16.

Because this version of the chip was intended for development of subthreshold analog circuits, the resistance of the switches was considered to be unimportant. The capacitance, however, was extremely critical, as it determined the effective time scale of the operation of the circuit. To try to minimize the effect of the capacitance, we scaled the transistors in the leaves such that the ratio of their total drain capacitance (diffusion capacitance *plus* leaf interconnect matrix capacitance) to transconductance was the same as that for an ordinary minimum-sized device (Figure 4.17). Compensation for capacitances in higher levels of the matrix could be provided by buffer stages inserted in the signal-routing channels (under the arrows in Figure 4.14).

The resistance of the switches becomes important for high-current digital networks, the

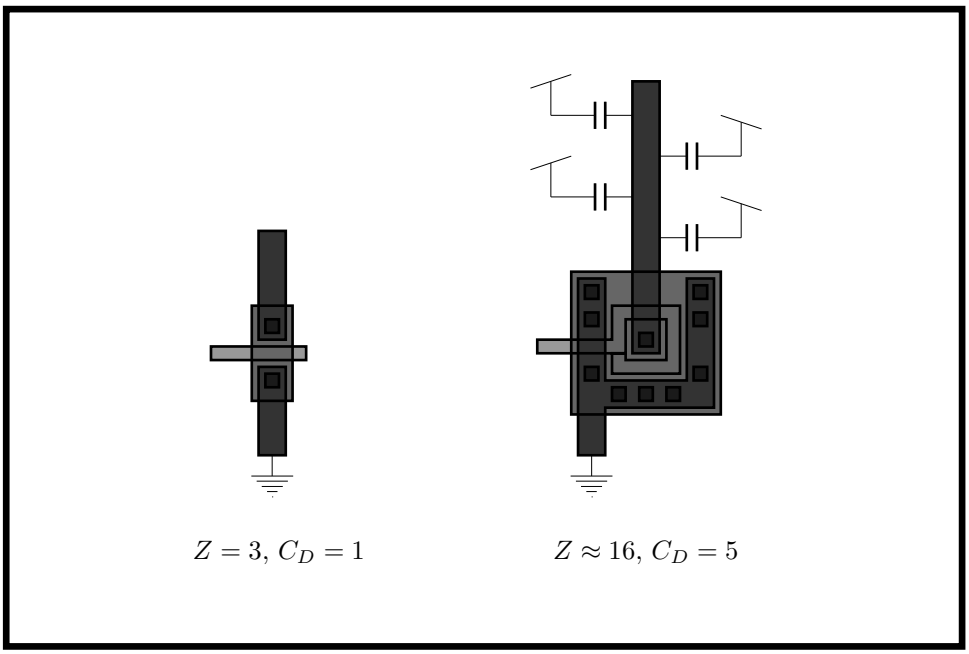


Figure 4.17: Ring transistors with small drain diffusion areas are used to maintain a constant ratio of current drive to capacitive load. The load in the case of the leaf cell comprises both diffusion (sidewall) capacitances of the transmission-gate switches, and wire capacitances.

transistors of which have relatively high drain conductances. As an artificially arranged example, a leaf cell configured as an inverter with two switches between n- and p-drains (Figure 4.18) exhibits the transfer function shown in Figure 4.19; the corresponding switch resistance is shown in Figure 4.20. For a correctly designed circuit, the transfer curve is more conventional (Figure 4.21).

Particularly for subtle leaf cells such as the analog leaf, sophisticated design may be required at this low level. Consequently, the network-embedding compiler searches a library of commonly used, predefined, and pretested leaves. The library currently contains simple and wide-range transconductance amplifiers, current mirrors, horizontal resistors, single transistors, two-input NAND and NOR gates, inverters, analog multipliers, and CSRL stages. It is being expanded constantly.

A test of the electrical performance of the hierarchical interconnect was performed by configuring a well-understood system: the follower–integrator analog delay line described in Chapter 9 of [**?**]. In theory, a delay line consisting of cascaded first-order sections (see Figure 4.22) will exhibit a transfer function

$$\frac{V_n}{V_{\text{in}}} = \frac{e^{-jn\omega\tau}}{1 + \frac{n}{2}(\omega\tau)^2}$$

where V_n is the output at the n^{th} tap, τ is the time constant of a single stage, and ω is the frequency component of the signal in question. Hence, at the cutoff frequency ω_c (the

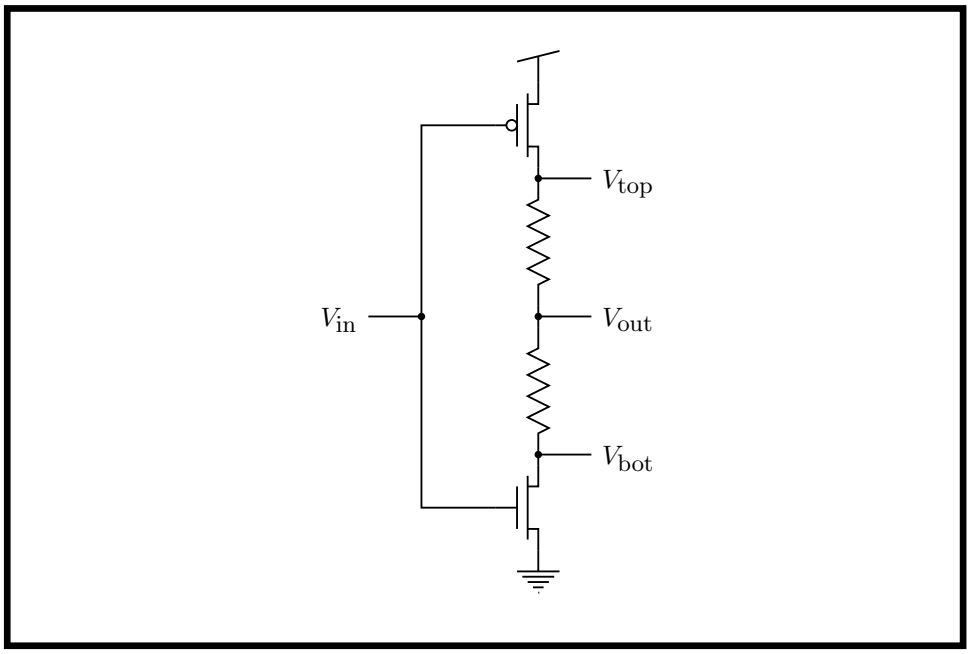


Figure 4.18: Inverter with transmission gates in series with high-current path.

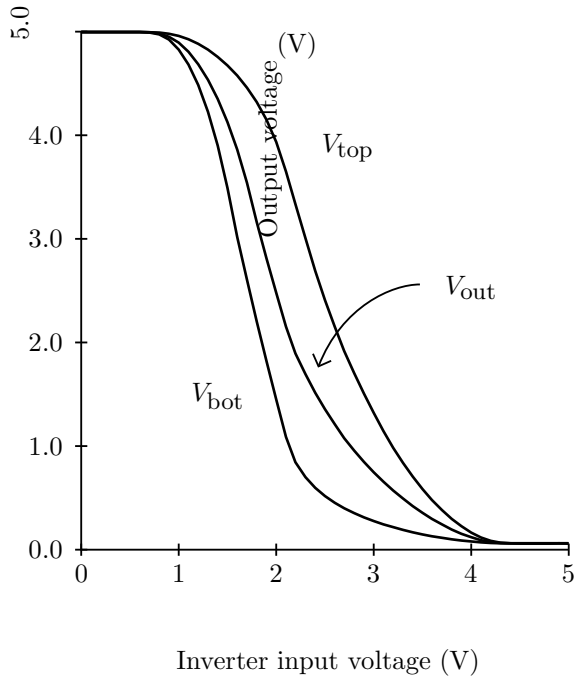


Figure 4.19: Transfer curve for inverter in Figure 4.18. A nonnegligible voltage drop across each transmission gate is clearly visible.

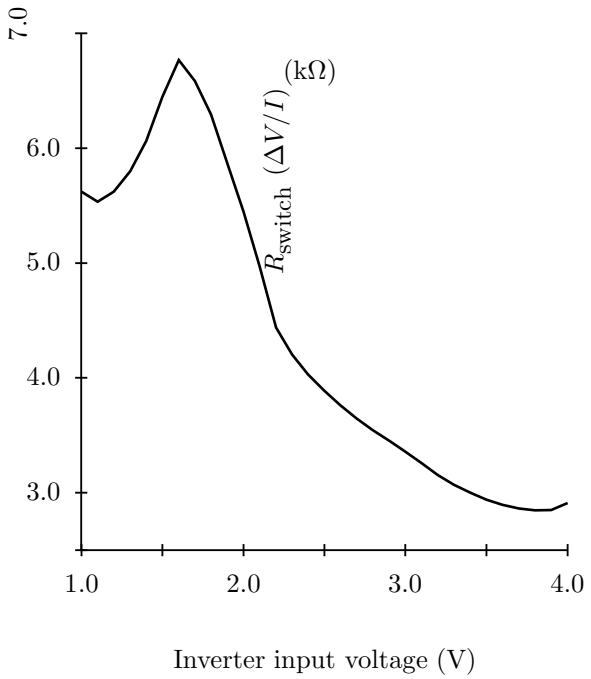


Figure 4.20: Effective large-signal resistance of transmission gates in Figure 4.18.

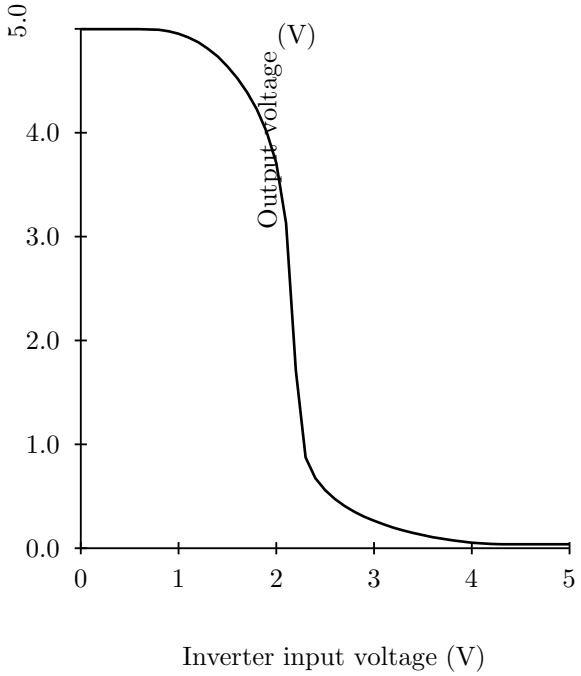


Figure 4.21: Transfer curve for an inverter designed with no transmission gates in series with high-current path.

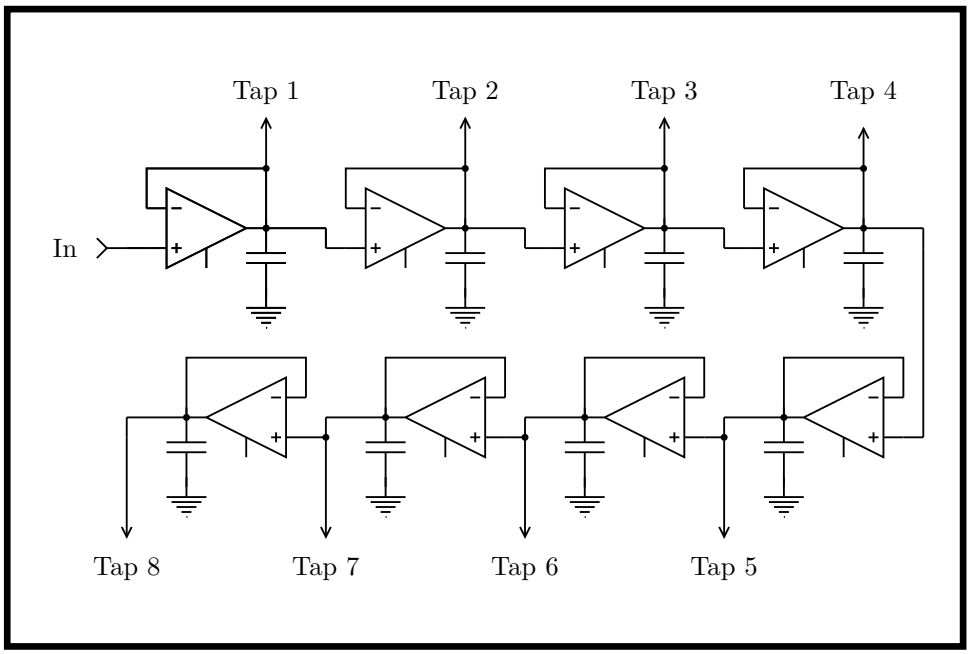


Figure 4.22: A follower–integrator delay line constructed of first-order sections (see [**?**]).

frequency at which the output amplitude is half the input amplitude), the delay-bandwidth product of the delay line is

$$\omega_c \tau = \sqrt{\frac{2}{n}}$$

Consequently, we expect the total phase delay to be a linear function of the distance along the delay line, and the rise-time of a step edge (t_r is approximately the reciprocal of the bandwidth ω_c) to be

$$t_r^2 = \frac{\tau^2 n}{2}$$

or, alternately, $t_r^2 \propto$ delay.

Experimental data are plotted in Figures 4.23 and 4.24, illustrating the correct electrical behavior of a follower-integrator delay line constructed using the PROTOCHIP.

4.8 Summary

The test PROTOCHIP has demonstrated the workability of a dynamically configurable array for prototyping analog circuits. The system described in this chapter is easy to use, provides fast turnaround (from schematic to ready-to-test chip in under 1 minute), and good performance and simulation accuracy, and is able to handle networks of interesting size.

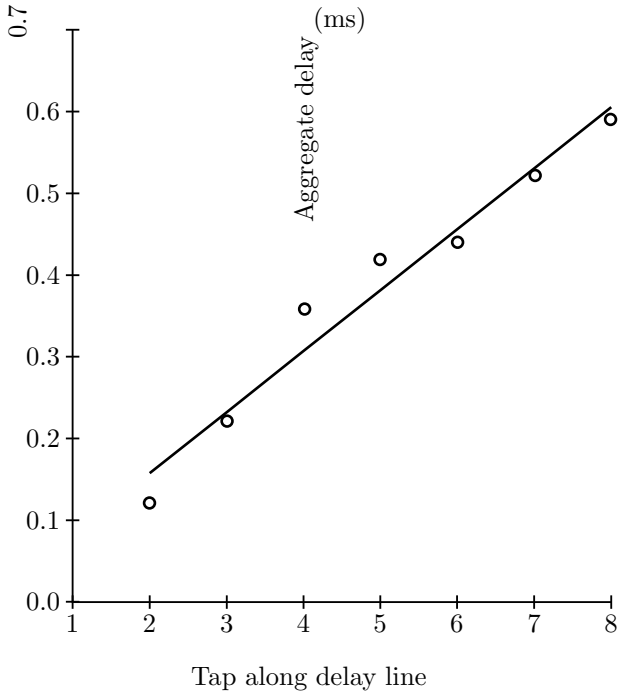


Figure 4.23: Waveform delay as a function of distance down the delay line of Figure 4.22.

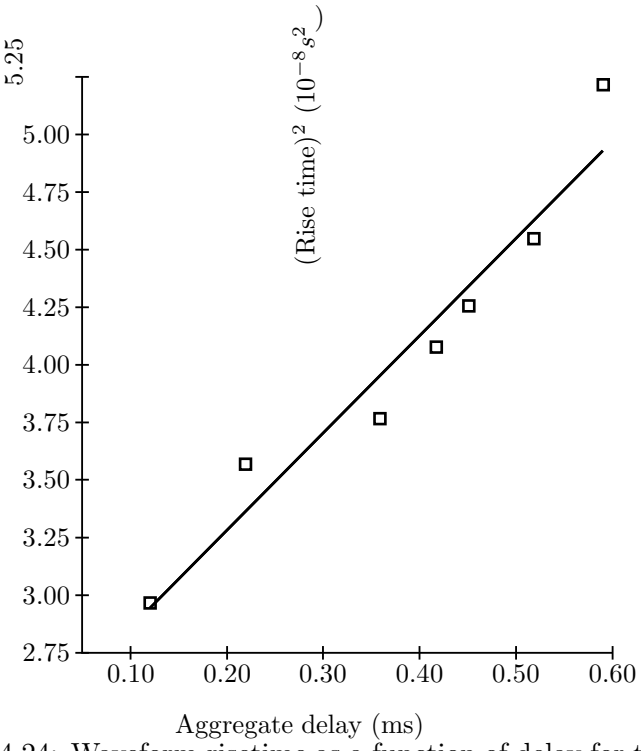


Figure 4.24: Waveform risetime as a function of delay for the follower–integrator delay line of Figure 4.22.

The ability to specify and instrument a circuit interactively and to perform a simulation is a great convenience to the user. We are currently integrating the network compiler and PROTOCHIP into a schematic capture/software simulator [**?**, **?**] environment. As the schematic editor is already part of our VLSI tool suite (for netlist generation and network comparison), this project is a step toward a truly unified design environment.

Future design iterations of the PROTOCHIP could provide a binary H-tree laid out as a hexagonally tesselated structure, intended for experiments on our retinal-model image-processing architectures. These designs would include dedicated local routing, and will intersperse processing elements in the interconnect hierarchy. Also, we are considering alternate interconnect technologies (in particular, a nonvolatile switch based on UV-programming of a floating gate in a standard CMOS process), driven by performance and density requirements.

Finally, we are exploring new applications of this methodology. For example, we are considering using the metacompiler to generate a system to model the analog computations that take place in the dendritic arborization of biological neurons, and to generate a system of interconnected neurons to facilitate experiments on models of biological central pattern generators.

Chapter 5

Specification of Netlists, Testing of Graph Isomorphism, and Embedding of Circuits onto the PROTOCHIP

Computers are useless. They only give you answers.

Pablo Picasso (1881-1973)

This chapter discusses three diverse subjects, tied together by their use of circuit netlists, and by their inclusion in a single design tool. In reverse order of treatment, these subjects are:

Embedding of circuits on a PROTOCHIP. We discuss several embedding algorithms, and reexamine the significance of Rent's rule in the context of hierarchical circuit graph partitioning. Then, we develop two algorithms to embed graphs. One is a dynamic-programming approach that operates bottom up; it is particularly suited to efficient embedding of small, difficult graphs. The other is a fast, greedy algorithm that operates top down; it is primarily useful for quickly embedding large, sparse systems. For completeness, these approaches are compared with a traditional simulated-annealing algorithm. All three approaches are supported within the PROTOCHIP software. Any combination of them can be used to embed subsections of a particular circuit.

Testing of graph isomorphism. A useful and often-underutilized VLSI design-validation approach is to compare a netlist extracted from layout to one specified independently, typically by circuit schematic. This technique is particularly valuable for validating automated layout generators. We have developed an efficient algorithm for netlist comparison; we discuss its implementation, and make general observations regarding its use over the last 5 years.

Specification of netlists. Because circuit netlists are essential to the two processes already mentioned, and because all popular netlist-specification formats have limitations or impose artificial restrictions on the user, the PROTOCHIP system employs a general mechanism to specify netlists. Built around a universal representation, NETGEN provides a versatile interface, allowing simple transformations between popular and commercial netlist formats.

5.1 NETGEN: A Netlist-Specification Language

Embedding a circuit onto a PROTOCHIP requires prior specification of the circuit. For generality, netlists are accepted from various sources, including LOG, anaLOG, WOL, and Magic. Many netlist formats exist in academia and industry. Four of the most popular are the following:

NTK: [Bryant et al., 1982] This format originated within the Caltech community. **NTK** is generated by **(ana)LOG** [**?**] and **WOL** [**?**], and is parsed by **netcmp**, **mossim** [Bryant et al., 1982], and **cosmos** [Beatty et al., 1989].

ext, sim: [Ousethout and et al., 1985] These formats are generated by the Berkeley design tools. **Ext** is the hierarchical extractor output format; **Sim** is a flat netlist format with limited electrical information, used primarily for switch-level simulation of digital circuits.

EDIF: [EIA88, 1988] This format is the emerging industrial standard. EDIF is a general purpose, LISP-like language capable of specifying schematics, layout, netlists, simulation vectors, and a host of other design-related data. Its versatility is also its greatest liability, as EDIF-based systems require embedded interpreters; this complexity has delayed EDIF's acceptance into the market.

SPICE: Spice "decks" are a ubiquitous netlist format, due to their use in a popular circuit simulator [**?**].

Unfortunately, all these formats suffer from shortcomings, generally related to their heritage as simulator inputs, extractor outputs, and so on. Furthermore, translation between one format and another has been inconvenient, and has spawned a large set of filters (ext2ntk, ext2sim, etc.). The opportunity thus exists for a simple, yet general, netlist-specification language.

NETGEN permits syntax-independent specification of connectivity by defining an internal representation (data structure) and a versatile interface to this representation, and by providing powerful primitive operators to manipulate this representation.

NETGEN is restricted to connectivity specification. That is, only the topology of a circuit is maintained; electrical properties of nodes or circuit elements are suppressed. However, many of these properties can be reintroduced as attributes of specific elements. For example, transistors of various sizes may be considered as different elements, instead of as variations within a single class. Other properties, such as node capacitance, may be reintroduced explicitly by lumped-element models. This flexibility allows the designer to identify the significant components of the circuit in question, and to abstract the rest.

5.1.1 Natural Support for Common Design Styles

It is imperative that any netlist tools be (1) convenient to use, and (2) useful for netlists of large scales. The degree of convenience is directly related to the speed with which the designer's intent can be captured.

It is now widely accepted that a structured design methodology is imperative to VLSI design [**?**]. An essential component of this methodology is hierarchical design; hierarchy is useful both as an abstraction and as an inheritance mechanism. A conservative and elegant design style in VLSI layout is that of a *separated hierarchy* in which a cell is either a layout cell or a composition of other cells [Rowson, 1980]. If we further restrict interconnectivity of cells to their perimeter (*composition by abutment*), it is then clear that the contents of each cell comprise its definition (i.e., there are no global wires that interact with the cell by being placed over it) [**?**].

On the other hand, strict hierarchy is often inconvenient when we are specifying large systems. Global signals, such as power buses and clock lines, are often suppressed for clarity. Signals of intermediate scope are often present (e.g., bit lines and word lines in a RAM).

NETGEN provides general hierarchical specification of netlists, with dynamic scoping rules that govern the visibility of objects. An optional topology-driven front end provides automatic connectivity between cells, according to a composition-by-abutment paradigm.

5.1.2 NETGEN's Internal Representation

The basic unit in defining a circuit is a *cell*. A cell, in turn, can contain any number of other cells, called *instances*. A cell has some special elements, called *ports*, that are the connections that can be made to the cell when that cell is instanced. After a cell is instanced, these ports are called *pins* within the context of the cell being constructed. The ports (of the current cell) and pins (of previously defined cells) may connect together directly, or through

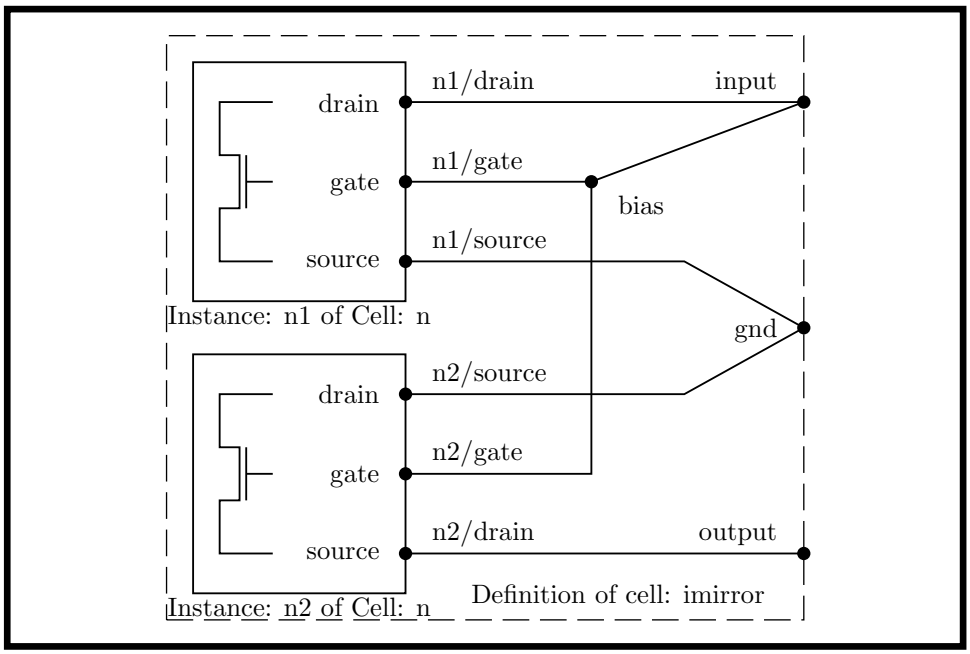


Figure 5.1: The components that comprise the NETGEN internal representation: cells, instances, ports, pins, and nodes. Ports actually come in several varieties: simple ports are shown in this figure, global ports and unique-global ports will be discussed later.

optional *nodes*; see Figure 5.1.

NETGEN has a uniform internal representation with a single operator: **connect**. Within a cell, the three kinds of elements (ports, nodes, and pins) are manipulated identically by **connect**. In defining a cell, the user **connect**s ports, nodes, and pins as required to construct the desired circuit. In particular, there is no direct equivalent of a "wire" within NETGEN; we can accommodate situations where named signal lines would be useful by using named internal nodes.

The **connect** operator is actually considerably more powerful, as it accepts as arguments two *lists*, whose components are then connected element by element. This behavior simplifies operations on structures such as buses. The list constructors can include general regular expressions; see Figure 5.2 for an example.

5.1.3 A Simple Model for Placement and Interconnect

NETGEN provides a simple model for VLSI cell placement: cells are rectangular and have ports along the perimeter on four sides (N,S,E,W). Composition is by abutment and can occur in either the horizontal or vertical direction. When two cells are placed such that they share an edge, the port lists along that edge of the respective cells are generated, and

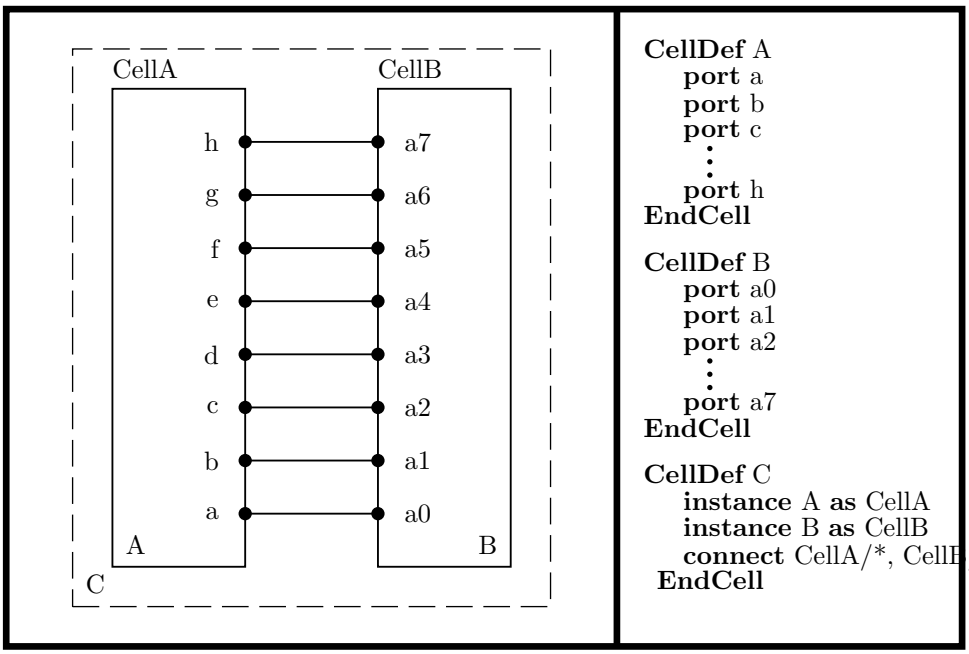


Figure 5.2: The connect operator can be applied to lists of elements. Regular expressions (and wildcards) can be used to generate these lists, providing a means to structure elements into buses, arrays, etc.

the **connect** operator is invoked on them. Ports along the orthogonal edges are propagated as ports of the cell being defined. The order of ports along all sides is preserved.

This model is supported by the **place** operator; **place** may be considered a variation on the **instance** operator, with the side effect of "sealing" ports of adjacent cells. See Figure 5.3 for an example.

This placement system is layered upon the **instance** / **connect** paradigm discussed in Section 5.1.2. The choice of rectangular cells is arbitrary; triangular or hexagonal composition could be accommodated easily (the placement system is written in under 50 lines of C code, using the embedded-language interface to the NETGEN internal operators).

5.1.4 Scoping Rules for Identifiers

The requirement that we continually declare high-connectivity nodes (e.g., power rails, bit/word lines) explicitly as parameters to each cell often is inconvenient. Also, the number of ports of generated cells (see Figure 5.3) must be kept manageable.

The solution to these difficulties takes the form of scoping rules for element names; when a cell is instantiated within the context of a calling cell, particular elements within the *called* cell are implicitly connected to elements in the *parent* cell. This binding process is exactly

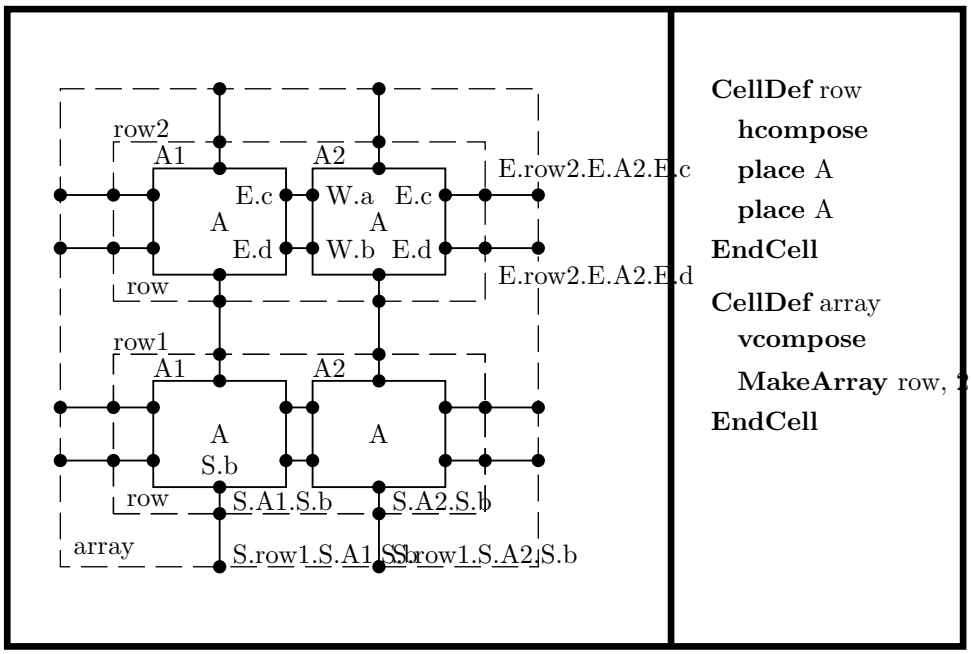


Figure 5.3: Using the NETGEN place operator to connect a row of cells. A direction of cell placement is specified by hcompose / vcompose. The MakeArray operator is simply a shorthand operator for multiple calls to place.

analogous to local/global variables within nested PASCAL procedure declarations, but the process is reversed: instead of declaring *local* variables, and assuming all others to have *global* visibility, particular ports are declared **global** and obey certain implicit connection rules.

Semantically, a global port A in cell B is a port that connects itself automatically when cell B is instanced. In particular, element A in cell B will automatically connect itself to any element named A that exists in the parent cell. If element A does not exist in the parent cell, instantiation of B will cause a **global** A to be defined in the parent. The result is that the unbound variable A is propagated to the next level of the hierarchy.

This dynamic scoping permits cell B to be declared *before* its parent cell (in contrast, this order is not possible with statically scoped languages such as PASCAL). An example of this feature is shown in Figure 5.4.

Naming Conventions for Ports

```
1) Ports, called pins in the context of the calling cell:
    <instance name> "/" <port name>

2) Ports - Oriented:
```

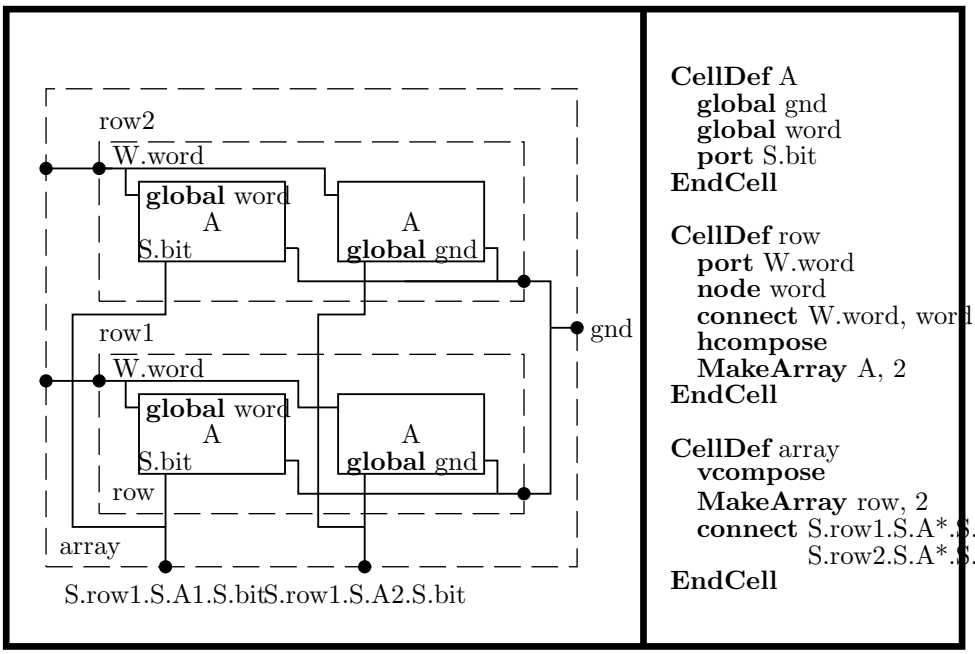


Figure 5.4: Dynamic scoping of identifiers.

```
    <side> {"." <instance name> "." <side>}* "." <port name>
    where side is one of (N, S, E, W)

3) Ports - Global:
    <port name> - name propagates unchanged, but is bound
       to locally-declared identifier of same name.

4) Ports - Unique Global:
    <my class name> "#" <instance name> "/" <port name>
```

5.1.5 Summary

NETGEN provides a versatile data structure for netlist representation, and a *policy-free* procedural interface. This interface allows simple interfacing to popular netlist formats (typically in either net-major or element-major order), and conversion between them.

Powerful netlist operators, including various flattening operators, permit convenient transformations required by various tools. Dynamic scoping of ports' visibility allows concise specification of large systems.

As a practical matter, NETGEN runs on a variety of platforms, and provides graphical (under the X Window System) and command-line interfaces, as well as an embedded-language interface in the form of a C language subroutine library.

Figure 5.5: A screen dump of the X Window System interface to NETGEN. User commands are entered via pull-down menus; arguments are specified in the text-entry subwindows.

5.2 Netlist Comparison for Validation of VLSI Systems

As VLSI systems continue to grow in scope and scale, increasing demands are placed on design tools. An effective but often underutilized design-validation technique is the comparison of the connectivity specified by a circuit netlist extracted from mask geometry to that of an independently specified netlist (typically obtained through a schematic-capture path). This technique has the intrinsic advantage of validating the entire set of design tools, as well as the design itself.

Furthermore, as designs increase in complexity, static validation techniques become more attractive than traditional dynamic techniques. Exhaustive switch- or even mixed-level simulation of a design is becoming increasingly impractical, and brings with it the well-known difficulty of designing a test-vector set with reasonable fault coverage. Indeed, as we shall see, the process of verification of circuit connectivity is applicable even for technologies and designs for which no corresponding simulation capability exists.

The objective of this section is to document an efficient algorithm for testing graph isomorphism, developed specifically for VLSI application, and to compare it to existing approaches. In addition, this section describes operational results of using this validation methodology over several years, in an academic environment.

5.2.1 Previous Work

The idea of connectivity verification has existed for some time. However, insufficient attention has been paid to development of algorithms for VLSI circuits. In general, most commercially available systems derive their origin from gate-level design of circuit boards, and are embedded within a suite of proprietary design tools. A notable exception is the WOMBAT system [**?**]. Like its predecessor LIVES [**?**], WOMBAT operates by assigning to each element a *signature* (ideally, a unique identifier based on an element's type, connected neighbors, fanin, fanout, and any other distinguishing characteristics the designer can identify). Element pairs (one from each netlist) with the same signature are *bound*; the algorithm proceeds with waves of elements bound by successive iterations. WOMBAT has demonstrated an ability to compare circuits comprising nearly 10,000 elements, although its internal data structures (and algorithms) are somewhat complex and memory-inefficient.

Our algorithm differs from this approach in five respects: (1) the concept of a signature is replaced by an equivalence class; (2) classes of indistinguishable elements are successively refined (we do not have to wait until we can bind a single pair of elements unambiguously); (3) the circuit is represented as a bipartite graph, with elements connecting to nodes; (4) a more efficient data structure permits substantial savings in *both* memory and execution time; (5) an efficient probabilistic algorithm has been developed to determine set equality.

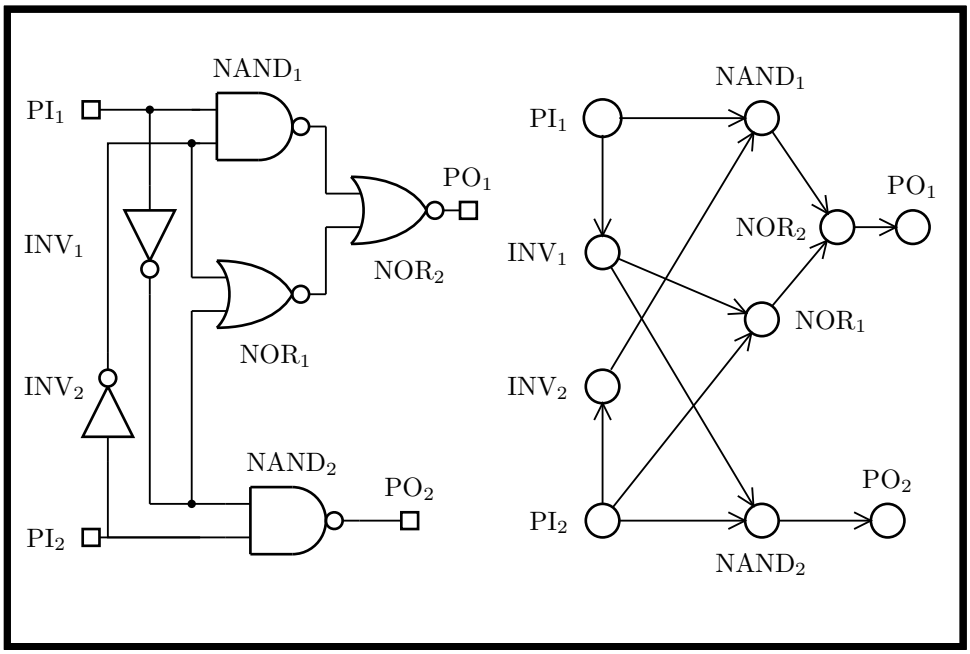


Figure 5.6: Transformation of a circuit into a graph. This representation is not unique unless there is only one driving pin per node (e.g., the output of an inverter), and that pin is known. In general, a node would have to be represented by a complete graph, which is wasteful of memory. Circuit and graph taken from [**?**].

5.2.2 Testing of Graph Isomorphism

A circuit can be represented as a *graph* (Figure 5.6), where elements (gates, transistors, etc.) are vertices, and edges imply connectivity. We immediately encounter two difficulties: (1) the fanout of circuit elements has structure (elements have distinct *pins*, some of which may be logically equivalent), and (2) electrical nodes are not uniquely represented by this transformation (all spanning trees are logically equivalent, but are not necessarily isomorphic). Both these problems are resolved with the graph shown in Figure 5.7; a bipartite graph of pins and nodes, with sets of pins grouped into elements (the problem of pin permutability is deferred until Section 5.2.3).

The problem of verifying connectivity is now cast as one of determining isomorphism between two graphs. That is, given two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$, we wish to know whether a permutation exists such that the one-to-one correspondence between vertices in the two graphs that it defines has the following property: For every edge between two vertices in the first graph, there is an edge that connects the two corresponding vertices in the second graph. Determination of graph isomorphism in the general case is possibly intractable (although it is not known to be NP-complete, either) [**?**]; intuitively, the difficulty is due to the need for complete enumeration of nearly identical parallel paths. In practice, such structures rarely arise in circuit design, and we report here an efficient algorithm that

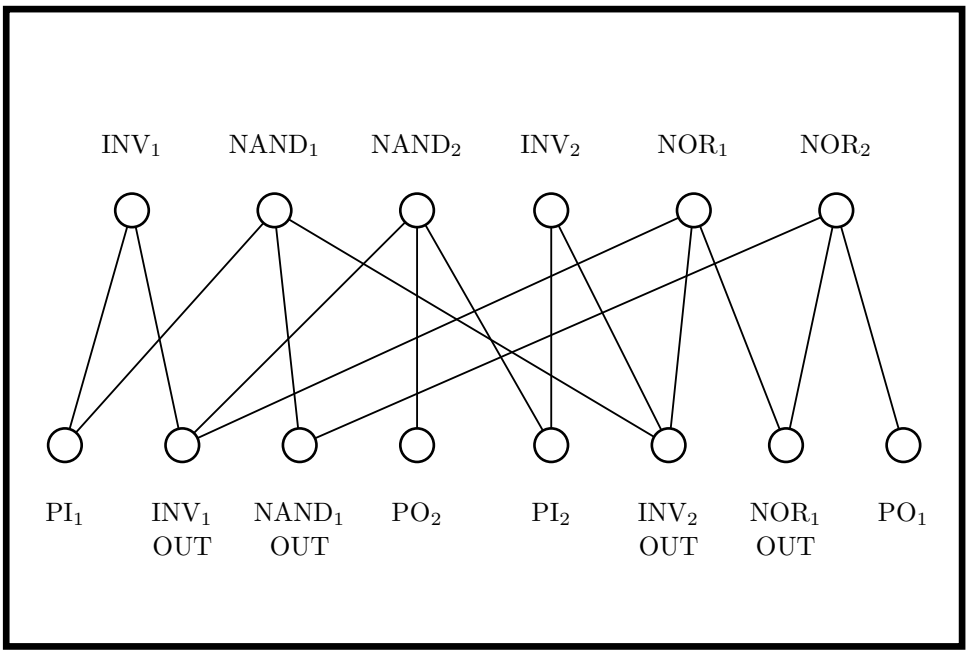


Figure 5.7: Representation of the circuit in Figure 5.6 as a bipartite graph of elements above and nodes below. This graph is unique for the circuit, and does not depend on knowledge about the elements and their pins.

has effectively linear running time in the number of elements in the graph.

Our algorithm is derived from the vertex-encoding procedure described in [**?**]. Initially, a graph is partitioned, using vertex invariants, into *equivalence classes* (sets of elements that cannot be distinguished, based on *current* information). We then refine these equivalence classes iteratively, by grouping members whose *neighbors* are indistinguishable. A simplified form of the algorithm is described in Figure 5.8.

The initial conditions are simple: All nodes are grouped into a single class, and all elements are partitioned according to their type (n-channel or p-channel MOSFET for typical CMOS processes). The first iteration then serves to fragment nodes according to their fanout.

In addition, we can incorporate into the initial conditions a user-specified explicit equivalence between a pair of nodes or elements, simply by creating an equivalence class containing only the two specified components.

The termination condition corresponds to three possible situations: (1) the graphs are isomorphic, and a unique correspondence has been found (i.e., the circuits are identical, and all elements and nodes in each have been labeled); (2) the graphs are different, and the difference has been identified; or (3) the algorithm was unable to determine that either case 1 or case 2 exists. It is instructive to consider the final equivalence classes in each of these cases.

```
repeat
       for each element class,
           for each element within the equivalence class,
               calculate a signature based on the node
               classes of the pins of this element
       group all elements with the same signature
           into new, separate equivalence classes

       for each node class,
           for each node within the equivalence class,
               calculate a signature based on the element
               classes of the pins contacting this node
       group all nodes with the same signature
           into new, separate equivalence classes

until no new equivalence classes were created
```

Figure 5.8: The simplified graph isomorphism algorithm. In practice, the loop is continued until any further refinement would create an illegal partition (an "equivalence class" with unequal numbers of components from each circuit). Termination is assured, since each iteration creates at least one new equivalence class, and classes are never re-combined.

In case 1, the algorithm terminates with E element equivalence classes and N node classes. Each class contains exactly two items, one from each graph, and encodes the correspondence explicitly. In case 2, the algorithm terminates with at least one equivalence class containing an odd number of elements. There can be no one-to-one pairing of elements within this class; hence the graphs are nonisomorphic.

Case 3 is the interesting situation, as the potential intractability of the problem is captured here. In its most benign form, this condition corresponds to automorphism in the parent graph (i.e., the graph is isomorphic to itself under some permutation of the vertices). In circuits, this situation arises most commonly due to parallel structures, such as parallel transistors in pad drivers. There is no unique labeling for such structures, but all labelings are equally good. In its pathological form, however, case 3 may conceal differences in the graphs. This result is most likely to occur in near-regular graphs (see Figure 5.9 and [**?**]); in practical terms, we observed this behavior only once in hundreds of test cases (documented in Section 5.2.4). Even so, the vertex-encoding algorithm makes a useful first pass, reducing the size of the subgraphs that can be subsequently test-labeled exhaustively. Furthermore, the number and kind of such potential automorphisms is logged, thereby giving the user a starting point for specifying explicit element equivalences.

At this point, it is useful to discuss several broad classes of common errors made in VLSI

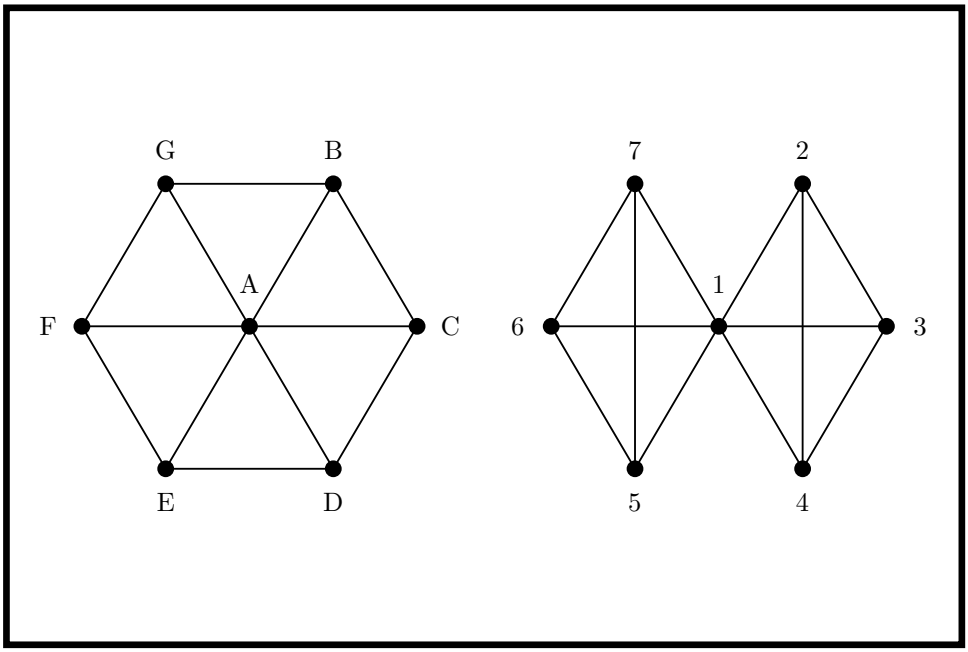


Figure 5.9: Two graphs that are different yet are incompletely partitioned. The algorithm terminates with the equivalence classes {A,1}, {B,C,D,E,F,G,2,3,4,5,6,7}. However, exhaustively trial-matching all elements of the second equivalence class yields no self-consistent partition, and hence the graphs are not isomorphic.

design and layout. The most common errors are shorts or opens; that is, a single electrical node in one graph corresponds to two (or more) nodes in the other graph. The algorithm easily identifies these errors, reporting upon convergence, a single "equivalence" class containing the nodes in question. Furthermore, a quick examination of the nodes' connectivities immediately indicates the problem (see Figure 5.10). The analogous error for elements is the omission or addition of a single element, usually in a long chain of identical structures. We can often determine the presence of a single defect of this type simply by comparing the total number of elements and nodes in the two circuits. Another easily identified error is the substitution of an incorrect gate type.

```
Parsing file:  csrltest.NTK
Found:  212 N Transistors, 212 P Transistors, 424 Total Transistors,
        164 Non-orphan Nodes, 2 Orphans,      166 Total Nodes.

Parsing file:  csrltcell.lntk
Found:  212 N Transistors, 212 P Transistors, 424 Total Transistors,
        163 Non-orphan Nodes, 0 Orphans,      163 Total Nodes.

********** results of fixed-point iteration **************
-----------------------------------------------
      (1)     Node:  _55                       fanout:    pg=1 ng=0 ps=1 ns=1
      (1)     Node:  _74                       fanout:    pg=0 ng=1 ps=0 ns=0
      (2)     Node:  CHIP/DELAY.3/TCAMP.5/IN   fanout:    pg=1 ng=1 ps=1 ns=1
-----------------------------------------------

1 difference found.
```

Figure 5.10: Typical output from NETCMP, illustrating an open node in the layout. The number of gates and source/drains connecting to node CHIP/DELAY.3/TCAMP.5/IN in circuit 2 is the sum of those connecting to nodes _55 and _74 in circuit 1.

A more subtle error occurs when symmetric elements are composed incorrectly (see Figure 5.11). These errors are typically difficult to isolate, since subgraphs are isomorphic, and only these subgraphs' connection through otherwise correct elements causes problems. The ability to equivalence particular nodes and elements explicitly is invaluable in tracking down this sort of problem. This procedure allows "waves" of binding to proceed outward from the near-symmetrical cell, rather than inward from the external environment.

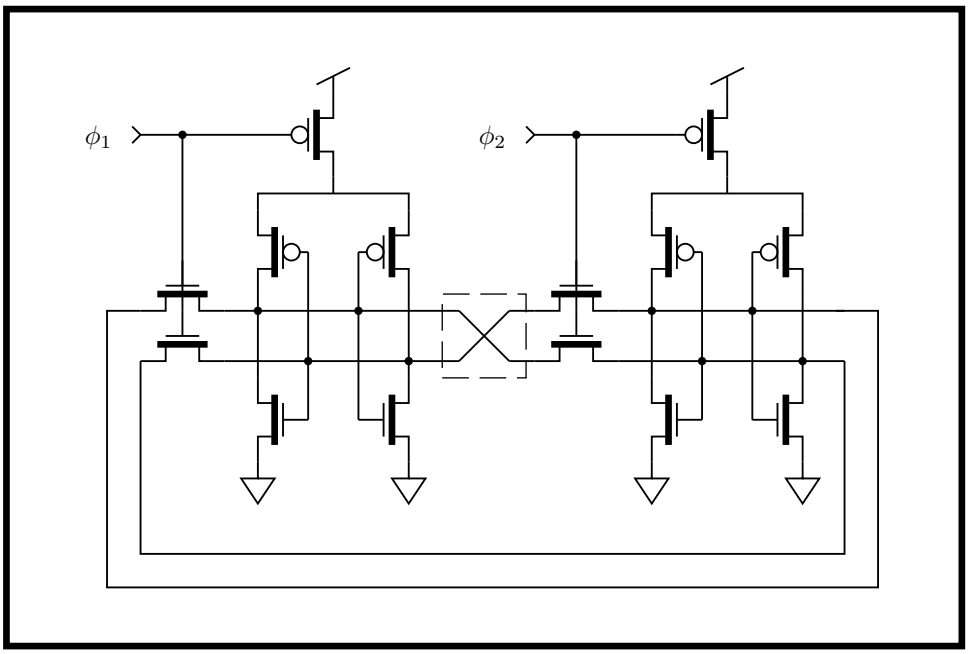


Figure 5.11: If the crossover indicated by the hatched line is absent, this CSRL [**?**] 1-bit counter becomes a latch! Unfortunately, the high degree of symmetry of this circuit make this difficult to detect. Typically, an error is reported at all four pass transistors, as well as their source / drain nodes.

5.2.3 Enhancements to the Algorithm

Computation of Set Equality

The most time-consuming part of the algorithm described in Figure 5.8 is the computation of set equivalences. In a computer, sets are represented by lists, and the *order* of elements within the list makes comparison of such sets difficult. For the purposes of this discussion, we define two lists L_1 and L_2, the elements of which are taken from a universe set $\mathcal{U}$. Furthermore, let S_1 be the set of elements in L_1, and S_2 be the set of elements in L_2.

In the simple case of no pin permutations, testing equality for element sets is simple: We traverse the pin lists of two elements, and determine whether the node classes of the contacted node are the same. Unfortunately, this procedure is inadequate for node classes, since the order of the pins is not guaranteed to be the same between the two graphs. Prior solutions include radix sorting, which requires $O(|\,\mathcal{U}\,|)$ memory for each element [**?**], and in-place sorting (implemented in an early version of NETCMP [**?**]), which requires $O(|\,S\,|\log_2|\,S\,|)$ time.

An alternative procedure to component-by-component comparison is the computation of an ensemble property of the list. We define a *hashing* function F such that

$$S_1 = S_2 \;\Rightarrow\; F(L_1) = F(L_2)$$

We can construct F by first defining a mapping $f : \mathcal{U} \to \mathcal{Z}_n$, where $\mathcal{Z}_n = \{0, 1, 2, \cdots, (n-1)\}$, and choosing $\oplus$ to be a commutative, associative operator over the elements of $\mathcal{Z}_n$. Then,

$$\begin{aligned} F(L) &= f(x_1) \oplus f(x_2) \oplus \cdots f(x_n) \\ L &= [x_1, x_2, \cdots, x_n] \end{aligned} \tag{5.1}$$

As a straightforward choice, assigning to each element of $\mathcal{U}$ some random integer (through the function f), and defining $\oplus$ as the addition operation modulo n, yields a mapping F with the desired behavior.

Consequently, in time $O(|\,S\,|)$, we can test the hypothesis $S_1 = S_2$, and possibly reject it. In fact, we can make this procedure necessary *and* sufficient by assigning unique integers to the members of $\mathcal{U}$, and iterating the process $\log_2|\,S\,|$ times (appropriately relabeling particular elements with each iteration). Of course, this complexity is equivalent to sorting the sets, then sequentially comparing them.

It is interesting to consider the probability of failure—that is, of incorrectly identifying two sets as equal when they are not. If n is a power of 2, and f assigns random *odd* integers from $\mathcal{Z}_n$, this probability is

$$P(F(S_1) = F(S_2) \mid S_1 \neq S_2) \;=\; \frac{1}{n}$$

Proof (sketch): Consider the difference sets S_1' and S_2' defined by:

$$\begin{aligned} C &= S_1 \cap S_2 \\ S_1' &= S_1 \setminus C \\ S_2' &= S_2 \setminus C \end{aligned}$$

By Equation 5.1,

$$\begin{aligned} F(S_1) &= F(S_1') \oplus F(C) \\ F(S_2) &= F(S_2') \oplus F(C) \end{aligned}$$

Since $\oplus$ is *invertible*, we can ignore the contribution due to $F(C)$, and observe

$$F(S_1) = F(S_2) \;\Leftrightarrow\; F(S_1') = F(S_2')$$

However, F(S) is a uniform random variable on $\mathcal{Z}_n$, which we have guaranteed by selecting the range of f to be those elements that are *generators* for the *cyclic group* $\mathcal{Z}_n$ under the operator $\oplus = +$, namely the odd integers in $\mathcal{Z}_n$ (proof omitted). Thus, the equality $F(S_1') = F(S_2')$ occurs with probability $\frac{1}{n}$. □

As a practical matter, 32-bit words offer more than adequate confidence ($\frac{1}{n} \approx 2.3 \times 10^{-10}$). For memory efficiency (see Section 5.2.3), even 16 bits are sufficient, especially in light of the possibility of testing for set equality by sorting and comparing, *after* the probabilistic algorithm has converged.

Pin Permutability

We are now ready to address the issue of pin permutability. Many VLSI structures provide terminals that are functionally equivalent, at some level of abstraction. A trivial example is the source–drain diffusions that contact a MOS transistor's channel; in a self-aligned gate process, these terminals are physically equivalent. A more complex example is an AOI gate (Figure 5.12), which has two pairs of logically equivalent inputs. In general, the problem of permitting pin permutability consists of allowing sets of terminals to have arbitrary order, while retaining the significance of other terminals' place in the pin list.

The solution to this problem is to extend the set equivalence metric described in the previous section to make *order* in the set significant. We can achieve this behavior by modifying the identifier associated with each set element based on its position in the list. The complete data structure for netlist comparison is shown in Figure 5.13; the set hash function for pin lists is then

$$F(\text{element}) = \sum_{i \in \text{PinList}[\,]} \mathit{place}(\text{PinList}_i) \text{ XOR } f(\text{class}[\text{node}[\text{PinList}_i]])$$

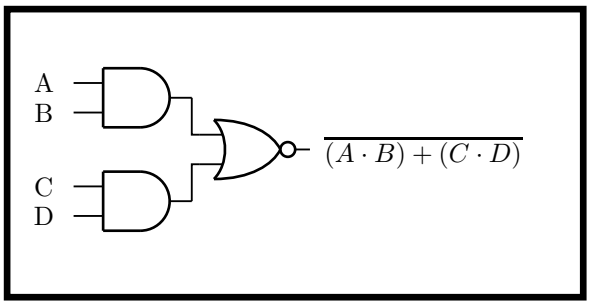


Figure 5.12: A typical and-or-invert (AOI) gate. Here, pins A and B are logically interchangeable, as are pins C and D.

Pins that are permutable are given identical *place* entries, in every instance of that element. (In this notation, $a[\ ]$ denotes dereferencing pointer "a", and $x(\ )$ denotes the value of an element in a structure.)

Similarly, the node hash function uses the PLACE field to ensure that the correct pin of an element has been contacted.

$$F(\text{pin}) = \sum_{i \in \text{ElementList}[\ ]} \mathit{place}(\text{pin}[\text{ElementList}_i]) \text{ XOR } f(\text{class}[\text{pin}[\text{ElementList}_i]])$$

Efficiency Considerations

We can obtain a further optimization by recognizing that the partitioning of node and element equivalence classes proceeds in waves: Only classes containing elements connected to nodes that are members of recently partitioned classes are themselves candidates for partitioning in the next iteration. These candidates are referred to as *activity regions*.

In addition, since the $\oplus$ operator is invertible, it is possible to compute the new hash function for a set *without* retraversing all of the set's elements:

$$\begin{aligned} F(S) \leftarrow F(S) \quad &\ominus \quad f_{\text{old}}(\text{class}[\text{node}[\text{PinList}_i]]) \\ &\oplus \quad f_{\text{new}}(\text{class}[\text{node}[\text{PinList}_i]]) \end{aligned}$$

where $\ominus$ is defined by $(x = z \ominus y) \equiv (z = x \oplus y)$.

Finally, to generate useful diagnostics in the case of nonisomorphic graphs, the partitioning of equivalence classes is continued until all remaining fractures would create an illegal partition ("equivalence" classes containing unequal numbers of components from each graph).

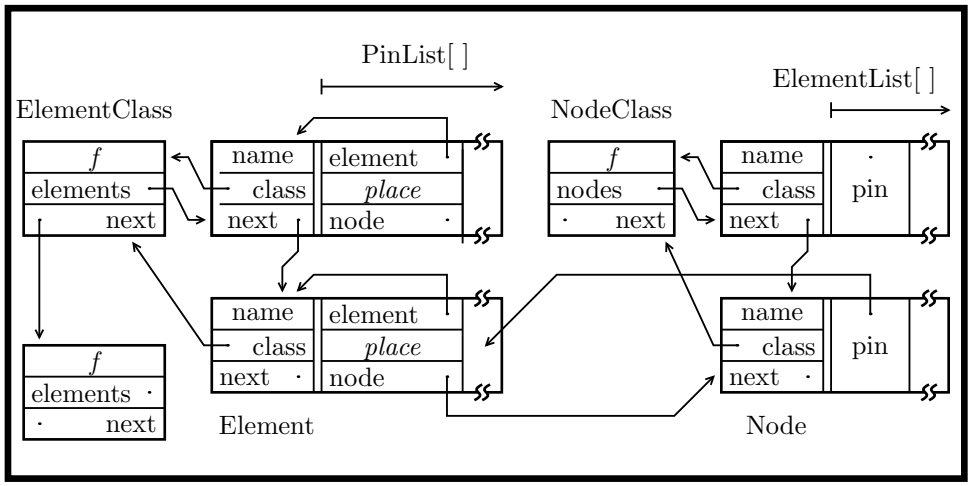


Figure 5.13: The datastructure for NETCMP, implementing a bipartite graph of the type shown in Figure 5.7. On the left, Elements are grouped into element equivalence classes. Each element contains a list of Pins, which point to Nodes. Similarly, on the right, Nodes contain lists of pointers back to pins. For clarity, not all pointers are shown. The "name" field is external to the algorithm, and is maintained for the user's convenience.

5.2.4 Implementation and Experience

The netlist-comparison procedure described in Section 5.2.2 has formed the core of our design-validation process for the past four years. Over that period, approximately 100 full-scale designs, and several hundred test-structure chips, have been verified, including all the systems described in [**?**]. Most of these projects contained analog components, making validation-by-simulation impossible. Validation by netlist comparison, coupled with geometric DRC, assured each project was at least testable on first silicon.

Extraction from the mask-level specification by the WOL layout editor [**?**], and a hierarchical schematic generated by LOG [**?**], provides the two independent netlists for comparison. In four years, over 5100 chip extractions were made, and the number of netlist comparisons is approximately the same. Unlike most previous algorithms, the NETCMP algorithm *requires* no explicit equivalencing of labels between the two circuits; this feature was most convenient for the user community, who were seldom willing to invest more than one person-day into the validation process.

A more recent version of NETCMP, implementing the probabilistic algorithm and features described in Section 5.2.3 has been incorporated into a system providing an embedded-language for netlist specification that supports a simple composition-by-abutment paradigm. Our objective is to integrate this netlist generator into the chip assembler tools, thereby producing a netlist simultaneously with the chip layout.

Performance of the algorithm is measured in terms of convergence time and memory usage. Typical performance data is shown in Table 5.1. Running time is approximately linear in circuit size; restricting partitioning to classes adjacent to recently partitioned classes provides a substantial speedup.

Circuit Elements	CPU Time (sec)	
	Without activity regions	With activity regions
8,400	21	9
51,100	140	52

Table 5.1: Performance of the NETCMP algorithm.

Comparison of Transistor Netlists

In the case where comparison of MOS transistor netlists is the only intended application, several simplifications to the algorithm are available to increase its performance. Transistor pins can be stored in fixed-length arrays, and the source/drain pin permutability can be hardwired into the algorithm, rather than the data structure. These simplifications reduce the structures in Figure 5.13 to use only $88T+34N$ bytes of memory to compare two circuits of T transistors and N nodes each. For most CMOS circuits, $N \approx T/2$ (see the next section), giving a total memory usage of 105 bytes per transistor. This result, in conjunction with the execution time performance of Table 5.1, demonstrates the feasibility of netlist comparison as a validation technique, as a 100,000 transistor chip fits comfortably in a (typical) 16Mb workstation. It is worth noting, however, that the performance of the algorithm degrades substantially when physical memory is exhausted, due to the effectively random references within the connectivity graph.

Node vs. transistor count for CMOS circuits It is interesting to note that for several common CMOS design styles, the number of electrical nodes N in a circuit is (almost exactly) one–half the number of transistors T. In practice, the actual number of nodes is likely to be lower, due to regular structures (such as PLAs and ROMs), and tri-state busses.

1. Restored logic gates with dual pull-up trees.

   Consider a n-input NOR gate, containing $T = 2n$ transistors.

	$3(2n)$	total pins
$-$	$2n$	gates are driven by other nodes
$-$	$2(n-1)$	pull-ups share output and power nodes
$-$	2	global power nodes
$-$	n	drains in the pull-down chain connect to sources
	n	nodes ($n-1$ internal, and 1 output)

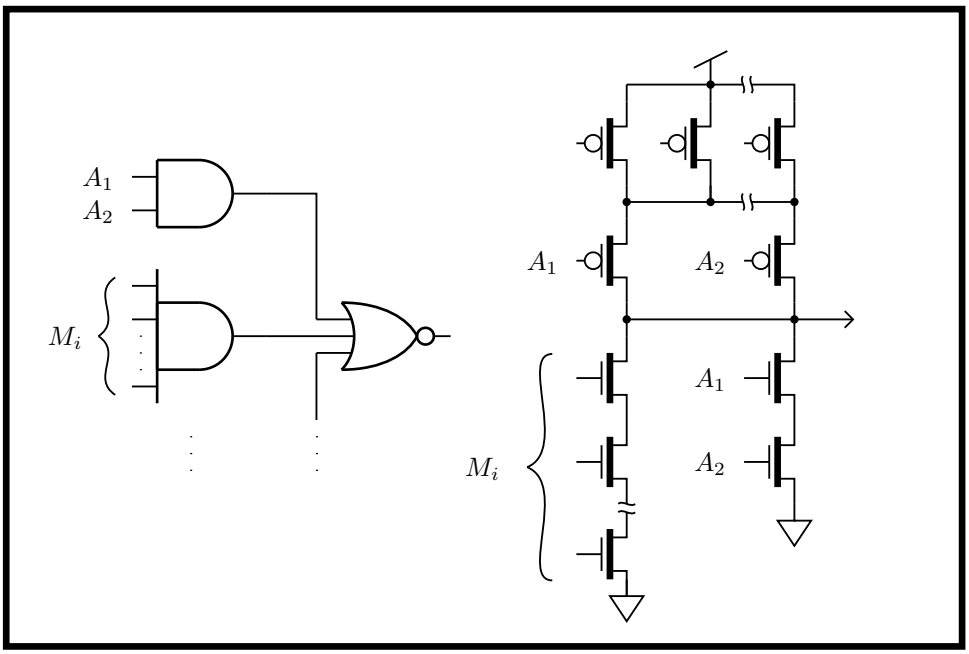


Figure 5.14: A general AOI gate. Each input gate has M_i inputs. The output NOR gate is actually a wired-NOR, and requires no additional transistors.

Of course, the same is true for n-input NAND gates, and for the 1-input degenerate form (the inverter). Consequently, any circuit comprised simply of such gates will have (counting external inputs and power nodes):

$$N = \frac{T}{2} + (\text{inputs}) + 2$$

2. Transmission gates, driven by restored logic.

   Each transmission gate adds two transistors and one node. Since their gates generally connect to global clock lines, at most a small (finite) number of global nodes is added to the system.

3. AOI gates.

   Consider the gate in Figure 5.14, with transistor count $T_i = 2M_i$. Each gate contributes $\left(\frac{T_i}{2} - 1\right)$ nodes to the pull-down chain, and 1 node to the pull-up chain. Hence,

$$N = \sum \frac{T_i}{2} = \frac{T}{2}$$

A Failure Case for the NETCMP Algorithm

In four years of use, the only observed failure of the algorithm to identify non-isomorphic circuits is illustrated in Figure 5.15. It is instructive to examine this case, to understand

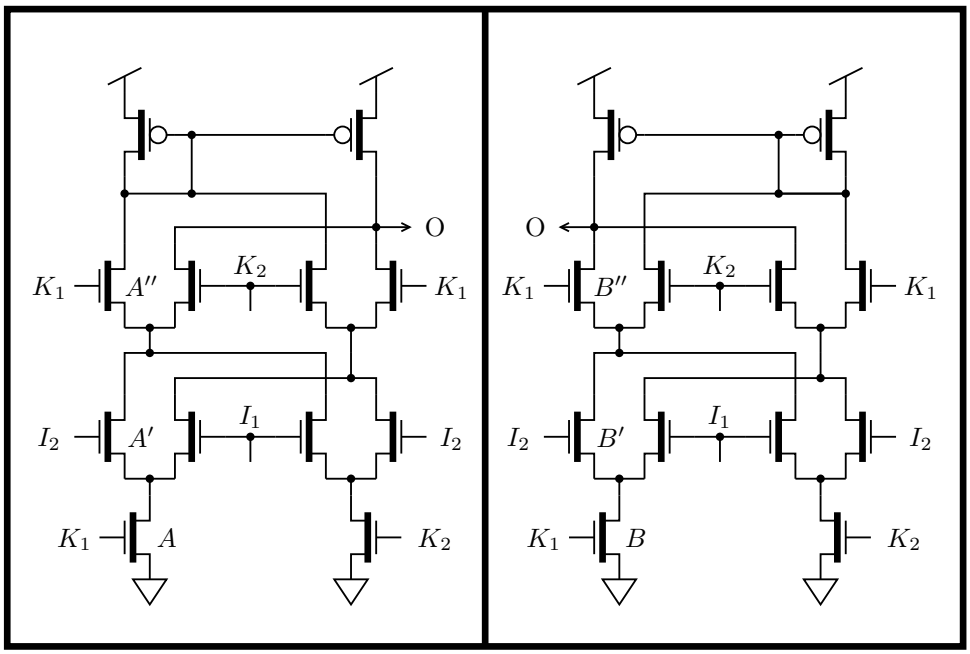


Figure 5.15: A real circuit for which the graph isomorphism algorithm fails. The circuit is *not* laterally symmetrical, because of the connection of the nodes marked K_1 and K_2. The algorithm groups the n-channel devices into 4 classes.

the limitations of the partitioning algorithm. Essentially, partitioning consists of applying a first-order predicate: do graph components A and B share the same set of connected elements? To correctly identify the difference between these two circuits, one must apply a second-order predicate: does A connect *above* to an element A′ which connects *above* to A″, while B connects eventually to B″ (clearly, A″ and B″ are not equivalent, and so neither are A and B). It should be noted that the high symmetry of the circuit could have been broken by a single labeling of a pair of inputs or transistors, which would have allowed the partitioning algorithm to correctly find the difference between the two circuits.

Automorphism Resolution by Trial Matching

To reduce the effect of the problem illustrated in Figure 5.15, we added a heuristic algorithm that is executed upon convergence of the NETCMP vertex-encoding algorithm. The heuristic consists of making a single trial match of two components within a single equivalence class, and then iterating the vertex-encoding algorithm to convergence. At this point, if no illegal "equivalence" classes have been detected, and if other automorphisms exist, the trial matching is repeated with another pair of components. This process is repeated until no automorphisms remain, or until an illegal partition of an equivalence class is made. In the former case, we report that the two original graphs were isomorphic, because we have obtained a self-consistent labeling of all their components. In the latter case, all we have shown is that the initial trial match was incorrect. Nonetheless, we report this as a dif-

ference between the graphs, primarily to encourage the user to examine the problem more closely (and perhaps consider trying some explicit specification of equivalent components, as an aid to the vertex-encoding algorithm).

5.2.5 Summary

Working with a large user community over a period of several years, we have observed netlist comparison to be an effective design validation procedure. It is an efficient *static* check that avoids the difficulties of dynamic validation (simulation is slow, and good test vectors hard to find). Also, netlist comparison provides a basic validation even for technologies that are not amenable to simulation (e.g., analog VLSI). As designs become more complex, the importance of fast validation increases.

The algorithm presented in this section is very fast, and, perhaps more importantly, is very efficient with regards to memory. The objective of operating on VLSI-scale circuits was designed-in from the start. The fundamental graph isomorphism algorithm was modified extensively by constraints of the VLSI environment (e.g., pin permutability, and the ability to iteratively compute the equivalence class hashing function F). Although the general graph isomorphism problem is hard, we have observed that the difficult cases rarely arise in real circuits.

It is important to note that validation techniques such as netlist comparison *supplement* automated layout systems (silicon compilers, module generators, etc.), and are not made obsolete by "correctness by construction." In fact, as silicon compilers are becoming an enabling technology for non-VLSI experts, the need to validate the compiler is actually more critical than that of validating individual designs.

Similarly, the value of mask-level extraction cannot be over-stated. In addition to validating the layout generator, mask–analysis tools are capable of identifying many common design errors, including floating nodes (e.g., CMOS wells that have not been connected to power rails), and yield-impact geometry (e.g., implant mask spacing rules that are missing from many DRCs).

In summary, netlist comparison provides a fast, *easy*, and efficient mechanism for identifying many catastrophic design errors.

5.3 A Bottom Up Graph-Embedding Algorithm

This section describes an algorithm to embed a flat circuit network on a $k = 2$ (binary) hierarchical interconnect network. For simplicity of description, all processing elements are considered to be at the leaves of the tree, and all leaves are assumed to be identical. In practice, we partially accommodate diversity among leaves by making the leaves programmable. In addition, the algorithm can be modified easily to accommodate a nonuniform hierarchy.

This graph-embedding problem is similar to the standard-cell place-and-route problem, with one significant difference: In the PROTOCHIP, the routing-channel resources are predefined and fixed, as is the structure of the interconnect. The basic approaches, however, remain the same. A top-down partitioning [**?**, **?**] attempts to divide a graph into two nearly equal parts, while approximately minimizing the number of wires crossing between them. A bottom-up technique considers candidate groups of elements for *merging*, based on some cost metric of amalgamating these cells into one [**?**, **?**]. Both of these approaches can be extended to incorporate the constraints of the hierarchical interconnect to prune the search tree of candidates for subdivision and merging, respectively. A third approach, based on simulated annealing [**?**], also is possible, but has not yet been explored in application to a hierarchical interconnect.

We selected a bottom-up approach employing dynamic programming for several reasons. First, a concise mathematical description of the algorithm provided a direct implementation, and facilitated an analysis of the problem complexity. More generally, the bottom-up approach can be applied easily to nonuniform hierarchies, and permits the use of cell libraries to take advantage of previously solved embeddings.

It is useful to view a circuit as a bipartite graph with two sets of vertices, corresponding to elements and electric nodes. Edges between these sets indicate the connectivity of the circuit. The nodal connectivity matrix C is defined by $C_{ij} = 1$ if circuit element E_j connects to E_i.

Two elements are candidates for merging if they share at least one common node. The element adjacency matrix A is computed from

$$\begin{aligned} A &= C^T C \\ A_{ij} &= \text{number of nodes shared by elements } i \text{ and } j \end{aligned}$$

A lower bound on the fanout of the element obtained by merging elements i and j is given by

$$F_{\min} = A_{ii} + A_{jj} - 2A_{ij}$$

If $F_{\min}$ is greater than the fanout of the hierarchical interconnect matrix, elements i and j cannot be merged, at least at this level.

An upper bound on the fanout is

$$F_{\max} = A_{ii} + A_{jj} - A_{ij}$$

Clearly, if $F_{\max}$ is less than the matrix fanout, elements i and j can be merged. Any intermediate cases must be judged on the basis of the actual fanout of the metaelement and the interconnect of the matrix.

If the elements can be joined, the new metaelement's fanout is computed, and the new element is added (as a new column) to the C matrix. The process is iterated until a metaelement is formed that contains all the elements that were present at the beginning of execution of the algorithm. To provide a trace of this process, from which a hierarchical embedding can be determined, an ownership matrix M is maintained, where

$$M_{ij} = \begin{cases} 1 & \text{if element } j \text{ contains element } i \text{ as one of its children} \\ 0 & \text{otherwise} \end{cases}$$

The termination condition for the algorithm can be stated in terms of M^*, the transitive closure of M, which may be interpreted as $M^*_{ij} = 1$ if element i appears in the tree rooted by j. This transitive closure can be computed directly from M, because the columns of M are topologically sorted.

We can improve the algorithm by rejecting as candidates for merger any two cells that have children in common. A necessary and sufficient condition for such independence is

$$(M^*\hat{u}_i)^T(M^*\hat{u}_j) = \vec{0} \;\Rightarrow\; ({M^*}^T M^*)_{ij} = 0$$

The final criterion for candidates for merger is that they be previously untested. To this end, two sets of elements are maintained:

$$\begin{aligned} N &= \{\text{all elements currently defined}\} \\ N' &= \{\text{elements added in the last iteration of the algorithm}\} \end{aligned}$$

Clearly, $N' \subseteq N$, and we satisfy the novelty criterion by considering for merging only pairs in the sets $N \times N'$ and $N' \times N'$.

The operation of this algorithm on the circuit in Figure 5.16 is described in Figure 5.17.

Heuristic modifications can accelerate this algorithm tremendously. In particular, much time is spent considering nodes of large fanout; dividing elements into equivalence classes based on their connectivity to global signals frees the algorithm to consider primarily the short local interconnections, which correspond to nodes that are swallowed when two elements are merged. Also, in situations where the interconnect has substantial excess routing resources, it often is adequate to eliminate an element from further consideration after it has been amalgamated into one (or more) higher-level elements.

Empirically, this algorithm works well for sparsely connected circuits, or for those that are difficult to embed (in the sense that only a small number of embeddings are possible, and these usually are not obvious to the human operator). It still works, but becomes computationally inefficient, for highly connected networks, because the number of elements

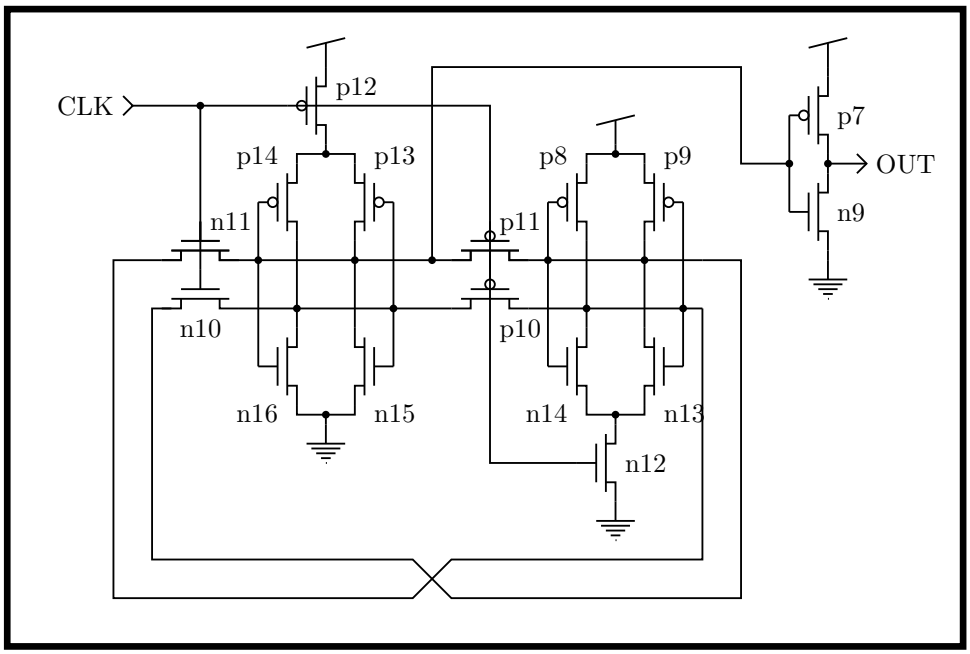


Figure 5.16: Embedding of a one-phase CSRL toggle flip-flop. The output from the graph-embedding algorithm is presented in Figure 5.17.

created becomes very large (in the absence of the simplifying heuristics). In general, these inefficiencies are a result of the breadth-first search pattern of the algorithm, and occur in cases when just about any trial embedding will work. These networks usually are easily embedded with simple operator intervention to the algorithm.

We can gain a rough understanding of the performance characteristics by examining the complexity of the embedding problem. Consider a simplified network of the form shown in Figure 5.18. For the first iteration of the algorithm, there are k equivalent neighbors of the center element to consider for merging. For the second iteration, there are $2(k-1)$ neighbors, but each of these is $O(k)$ organized from possible first-iteration mergers. Thus, there are $O(k^2)$ second-iteration merges. By induction, we can show that the number of neighbors at merge iteration n is

$$d_n = 2d_{n-1} - 2 = 2^n \left(\frac{k}{2} - 1\right) + 2$$

Each of these neighbors is organized $O(k^n)$.

For a graph of height $\log_2 N$, the total number of possible merges, adding the N possible starting points, is

$$M = N2^{\log_2 N}\left(\frac{k}{2} - 1\right)k^{\log_2 N} = \left(\frac{k}{2} - 1\right)N^{2+\log_2 k}$$

Note that this approximation is a gross overestimate, as the number of eligible neighbors

```
Embedding for csrlntk (level 4):
(
      (((n9 p7) (p11 n11)) ((n15 n16) (p14 p13)))
      (((n13 n14) (p12 p10)) ((p8 p9) (n12 n10)))
)
```

Figure 5.17: The embedding of the circuit in Figure 5.16 found by the bottom-up embedding algorithm. This circuit has a bisection width of eight wires, which happened to coincide exactly with the fabrication specification of an early PROTOCHIP prototype.

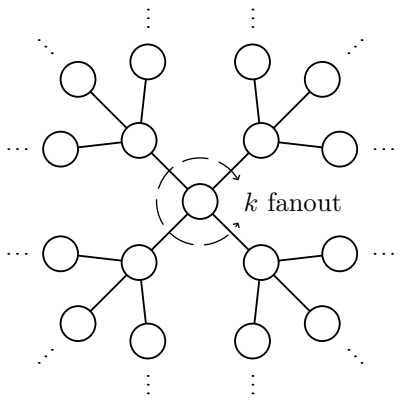


Figure 5.18: Network model for complexity analysis. Each of N elements has a fanout of k; for this analysis, wires run between pairs of elements only.

in the real (i.e., finite) circuit approaches 1 as $n \rightarrow \log_2 N$. Even so, the algorithm has a polynomial time complexity, due to the finite fanout of the individual elements.

The case of fanout to multiple elements can be handled by the transformation shown in Figure 5.19, where multiply connected nodes become pseudoelements. For nodes of fanout less than or equal to k, the previous analysis is unaffected. Nodes with fanout greater than k are best considered global signals, and are best accommodated explicitly in the allocation of routing resources. In other words, it is the consideration of elements that connect to large-fanout nodes (such as power rails and clocks) that makes the simple algorithm inefficient.

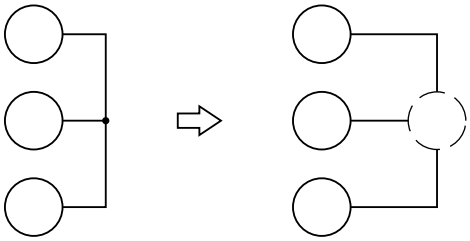


Figure 5.19: Transformation from shared node to pseudoelement(s) with bounded fanout.

5.4 A Fast Greedy Algorithm for Graph Embedding

In keeping with the objective of providing a rapid turn-around prototyping tool, it is often desirable to provide a fast, albeit suboptimal, graph-embedding mechanism. Such an algorithm would supplement the bottom-up algorithm described in Section 5.3, and would be particularly useful for the large number of "easy" circuits (i.e., those circuits that are easily accommodated by the wiring resources predesigned into any specific PROTOCHIP).

In this section, we describe a simple greedy algorithm for graph partitioning, and compare its performance to a graph-bisection algorithm based on simulated annealing. A theoretical model for partitioning random graphs is developed, and experimental results are compared.

5.4.1 Greedy Graph Partitioning

We take a direct approach to graph partitioning by performing a breadth-first traversal of the graph, terminating when one-half of the vertices of the graph have been visited. Breadth-first search is described in Figure 5.20.

```
S = ∅
Q = single vertex chosen at random
while ||S|| < N/2 do
      assign the element at the front of Q to V
      add V to S
      for each vertex P adjacent to V
          if not(P ∈ S or P ∈ Q)
             add P to the end of Q
```

Figure 5.20: Breadth-first traversal of the vertices in a graph. All vertices adjacent to a particular vertex are traversed in sequence. Breadth-first searching requires a queue of elements Q.

In practice, for real circuit graphs, two simplifications increase the effectiveness of this approach tremendously. First, high-connectivity nodes are not traversed. Empirically, for N element circuits, nodes with fanout greater than $\log(N)$ are best considered global nodes, and thus are not subject to traversal.

Second, simple optimization by gradient descent is usefully applied after the termination of the breadth-first search. This procedure is quite fast, as each element is assigned a score equal to the additional number of wires that would be cut if the element were moved across the partition. A negative score implies an improvement in the bisection width. Elements from each side with negative scores are selected and exchanged (a minor correction needs to be made to the score if the elements share common nodes, to prevent limit cycles). This

iterative improvement is $O(N)$, and typically reduces the bisection width by 20 percent.

Performance of this greedy algorithm is detailed in the next section.

5.4.2 Analysis of the Greedy Algorithm for Random Graphs

This section examines the performance of the greedy bisection algorithm on random graphs of fixed degree. Such graphs are intended not to model real circuit graphs faithfully, but rather to derive an upper bound for the performance degradation caused by simplifying transformations of the type made in Section 5.3, where high-connectivity nodes can be (randomly) broken up into a regular tree of pseudoelements.

We encounter two problems in approximating the bisection width of a random graph obtained by the greedy algorithm:

1. We need to compute the expected number of wires that will cross a given partition, under the random connectivity assumption.

2. We need to estimate the number of elements that these wires will contact.

Expression of the problem in graph terminology is helpful. Our random circuit is expressed as a regular graph (all vertices have the same degree (or fanout)), where each element corresponds to a single vertex, and each wire to a single edge. Because an element is allowed to contact another pin on itself, *loops* are permitted, and the graph is not restricted to be simple (simple graphs have all edges connected to *different* vertices).

In standard graph terminology, the number of vertices V in the graph is equal to the number of elements N in the circuit, and the degree of the vertices d is equal to the number of pins P.

Let us consider first the worst-case performance of the greedy algorithm. This case occurs in the tree structure (Figure 5.21); no children are multiply connected to the root, and consequently we maximize the number of wires that must be cut after level I.

There are P subtrees, each with branching factor $P-1$. At level i, each subtree contains $(P-1)^i$ elements. Consequently, the total number of elements in the tree, up to level I, is

$$\begin{aligned}\eta &= 1 + P\left(\sum_{i=0}^{I}(P-1)^i\right) \\ &= 1 + \frac{P}{P-2}\left((P-1)^{I+1}-1\right) \\ &= \frac{P(P-1)^{I+1}-2}{P-2}\end{aligned}$$

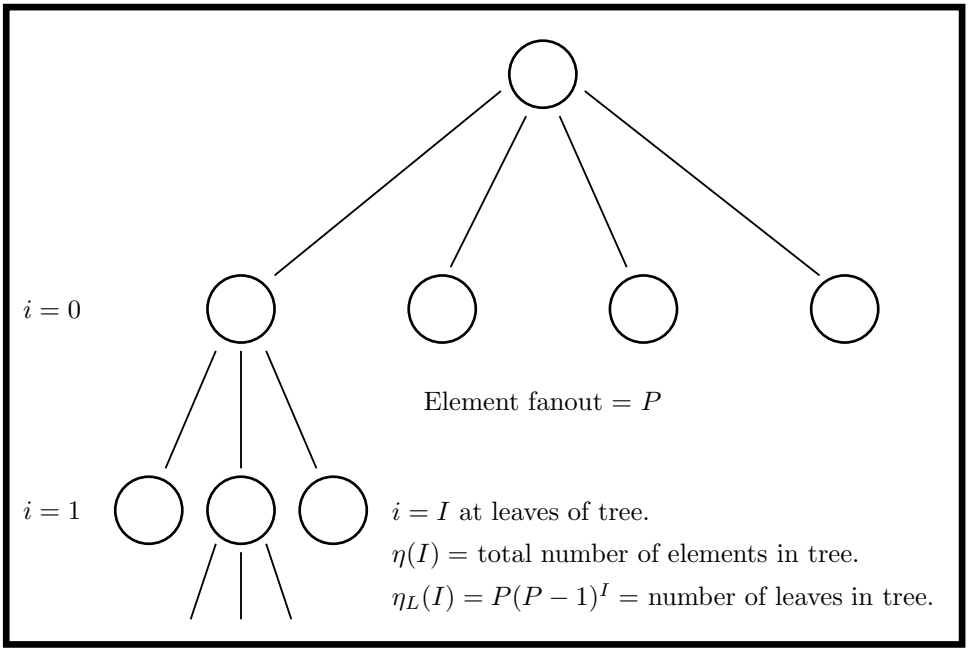


Figure 5.21: Worst-case analysis of greedy algorithm. Each element has only singly connected children, so a tree structure exists.

Now, we want to choose I such that $\eta = \frac{N}{2}$. Substituting into the previous expression gives us

$$(P-1)^{I+1} = \frac{\frac{N}{2}(P-2)+2}{P} \tag{5.2}$$

We can solve for I directly, or simply substitute Equation 5.2 into our first expression for the number of leaves, to obtain

$$\eta_L \equiv P(P-1)^I = \frac{\frac{N}{2}(P-2)+2}{P-1}$$

Each of these leaves has $P-1$ pins still to be connected. Now, we need to consider how many of these pins connect to each other, and how many connect to the other side. In the worst case, all these pins connect to the other side, resulting in a bisection width

$$w = \frac{N}{2}(P-2)+2$$

Recall that, for a purely random cut through the graph, the expected number of cut wires is $\frac{NP}{4}$. Consequently, the greedy algorithm is guaranteed to be superior to the random-cut algorithm for all cases where $P \leq \frac{4(N+2)}{N}$.

Suppose, however, that we remove the restriction that all remaining pins of the leaf elements connect to pins of elements on the other side, and consider the possibility that the pins of

some of these elements connect to pins of *other* leaf elements. Every such connection reduces the bisection width by two wires.

Connecting Pairs of Elements

This task of estimating the number of wires crossing a partition when the number of free terminals on each side is known is precisely problem 1 in Section 5.4.2. In this section, we present a solution, assuming that the connectivity of terminals is random, and that all terminals are unique. Note the subtle departure from conventional graph theory: The pins of an element are presumed unique (i.e., not permutable), whereas the different out-edges of a vertex are differentiated by only their other end. As a trivial example, there is only one way for a vertex of degree 3 to have one self-connected edge and one free terminal, but there are three ways for a transistor to have two terminals connected (assuming source and drain are distinguishable).

Lemma 5.1 Given A objects of type 1 and B objects of type 2, with all objects unique, and given an experiment in which pairs of objects are selected until the initial pool of objects is exhausted, the expected number of pairs that contain elements of *different* types is

$$\langle w \rangle = \frac{AB}{A+B-1} \tag{5.3}$$

Proof 1: Without loss of generality, consider the elements of type 1. Since the pairing process is random and the selection statistics are stationary, the probability of a *particular* element of type 1 being paired with an element of type 2 is

$$P\{\text{type 1 paired with type 2}\} = \frac{B}{A+B-1}$$

Note that the denominator precludes an element being paired with itself. Consequently, when this process is repeated for all A objects, the expected number of cross-type pairings is simply $\frac{AB}{A+B-1}$. □

Proof 2: Consider the number of pairs containing one element of type 1 and one element of type 2. Clearly, there exist AB such pairs. The total number of possible pairs is $\binom{A+B}{2}$. The probability that a *particular* pair will contain two elements of different types is then

$$P\{\text{pair is cross-typed}\} = \frac{AB}{\binom{A+B}{2}}$$

Our experiment consists of selecting $\frac{A+B}{2}$ pairs, in all. The expected number of cross-type pairings is then

$$\begin{aligned} \langle w \rangle = \left(\frac{A+B}{2}\right) \cdot P\{\text{pair is cross-typed}\} &= \frac{A+B}{2} \cdot \frac{AB}{\binom{A+B}{2}} \\ &= \frac{AB}{A+B-1} \end{aligned}$$

□

We now return to the analysis of the expected performance of the greedy algorithm, and consider the expected fanout of the tree of elements. Recall that

$$\eta_L \equiv P(P-1)^I = \frac{\frac{N}{2}(P-2)+2}{P-1}$$

The number of free pins on the tree side is $A = \eta_L(P-1)$; on the other side of the partition, all pins are available, and all are equally likely to be contacted, so

$$\begin{aligned} A &= \frac{N}{2}(P-2)+2 \\ B &= \frac{PN}{2} \end{aligned}$$

Applying Lemma 5.1 gives

$$\langle w \rangle = \frac{AB}{A+B-1} = \left(\frac{PN-2N+4}{PN-N+1}\right)\frac{PN}{4}$$

In the large N limit, we now have $\langle w \rangle \approx \frac{P-2}{P-1}\frac{PN}{4}$, yielding an advantage over the random-cut algorithm for *all* values of P. In particular, the interesting case of $P = 3$ (i.e., transistor-level schematics), yields a 50 percent improvement.

Determining the Incidence of Cut Wires

The obvious next step is to relax the worst-case assumption of a tree of elements, and to apply the wiring estimator of the previous section after each iteration of the greedy algorithm. Unfortunately, this approach requires the solution of another estimator problem: How do we estimate the number of elements these w wires will contact?

This problem, the second described in Section 5.4.2, is very similar to a problem encountered in analyzing the performance of a computer hash table. In particular, given w probes into a table, using a hash function that randomly returns r values, the expected number of occupied table entries is

$$n = r\left(1 - \left(\frac{r-1}{r}\right)^w\right)$$

This expression is nearly the solution to our problem, except that a hash table of this sort (commonly referred to as *hashing with chaining*) permits an unlimited number of table entries to occupy a given bin, whereas in our graph problem the circuit elements have a finite, bound fanout. Also, within each bin all entries are equivalent and the hash table fills up in order, whereas all the pins of the elements are assumed to be unique.

Lemma 5.2 If a rectangular matrix with N columns and P rows randomly contains w nonzero elements (with the remaining $NP-w$ elements being zero), the number of *columns*

of the matrix that contain *at least one* nonzero element is probably

$$\langle \text{number of columns that are not all zeros} \rangle = N\left(1 - \frac{\mathcal{P}_P^{NP-w}}{\mathcal{P}_P^{NP}}\right) \tag{5.4}$$

This expression is valid for $NP - w \geq P \Rightarrow w \leq P(N-1)$. If $w > P(N-1)$, it is trivial to show that all N columns must contain at least one nonzero element.

Proof: Again, the statistics are stationary, so consider the probability of a single column containing all zeros. In all, there are $NP - w$ zero entries to place, so the probability that the first entry in that column will be zero is $\frac{NP-w}{NP}$. The probability that the second element is also zero is $\frac{NP-w-1}{NP-1}$. The probability that all P entries in that column are zero is then

$$\begin{aligned} P\{\text{single column is all zeros}\} &= \frac{NP-w}{NP} \cdot \frac{NP-w-1}{NP-1} \cdots \frac{NP-w-P+1}{NP-P+1} \\ &= \frac{\mathcal{P}_P^{NP-w}}{\mathcal{P}_P^{NP}} \end{aligned}$$

The expected number of all-zero columns is then $N \cdot P\{\text{single column is all zeros}\}$, from which the lemma follows. □

We are now in a position to model the behavior of the greedy algorithm. By starting with the initial condition of a single contained element with fanout P, and iterating the application of Lemma 5.1 and Lemma 5.2 to find the number of cut wires and contacted elements, respectively, we obtain the data tabulated in Table 5.2. These data are compared to observed experimental data in Figure 5.22.

i	Inside elements	Inside terminals	Outside elements	Outside terminals	Cut wires	Contacted elements
	1	3	999	2997	3	3
0	4	6	996	2988	6	6
1	10	12	990	2970	12	12
2	22	24	978	2934	24	24
3	46	48	954	2862	47	46
4	92	91	908	2724	88	85
5	177	167	823	2469	156	146
6	323	282	677	2031	248	219
7	542	409	458	1374	315	248
8	790	429	210	630	255	166
9	956	243	44	132	86	42
10	998	40	2	6	6	2

Table 5.2: Iteration of the greedy algorithm model equations for a random graph of 1000 elements of fanout 3.

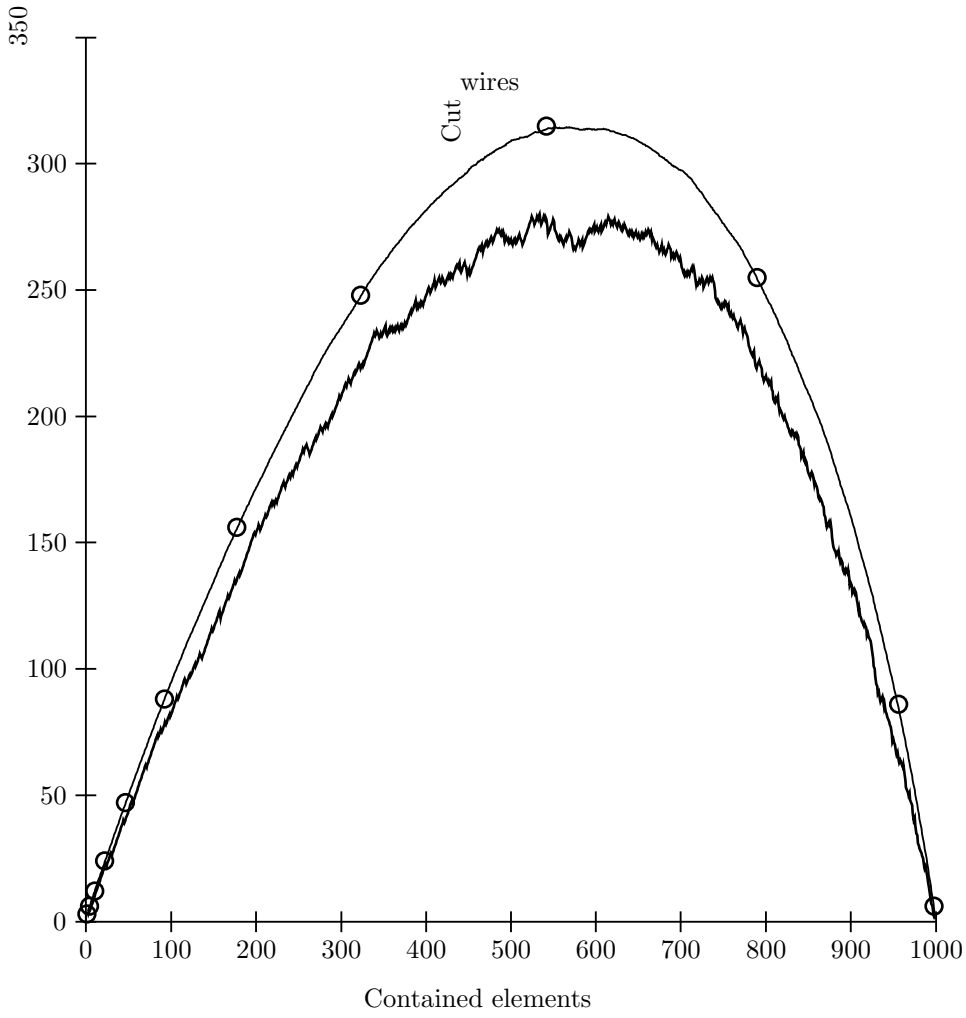


Figure 5.22: Graph bisection width found by greedy algorithm, as a function of number of contained elements. The graph contained 1000 elements of degree 3, with random connectivity. The curves represent the average bisection width (top curve), the *best* bisection width found (bottom curve—given 100 iterations with random initial element selection), and the predictions of the model (open circles represent data taken from Table 5.2).

5.4.3 Summary

The greedy algorithm for graph partitioning presented in this section is very fast and reasonably efficient. It has successfully embedded many networks, including large networks of several hundred components[1].

For comparison, experimental data for the performance of graph partitioning by simulated annealing [**?**] [Kernighan and Lin, 1970] [Dunlop and Kernighan, 1985] are presented in Figure 5.23. Although the final result is superior to that obtained by the greedy algorithm, the long running time required can make simulated annealing inappropriate for a fast-turnaround prototyping system. The PROTOCHIP run-time environment supports both approaches, thereby providing both fast embedding of easy circuits, and efficient embedding of more complex circuits.

[1]Thanks to D. Johannsen of Silicon Compilers for generously supplying several netlists generated by a high-level logic synthesis tool.

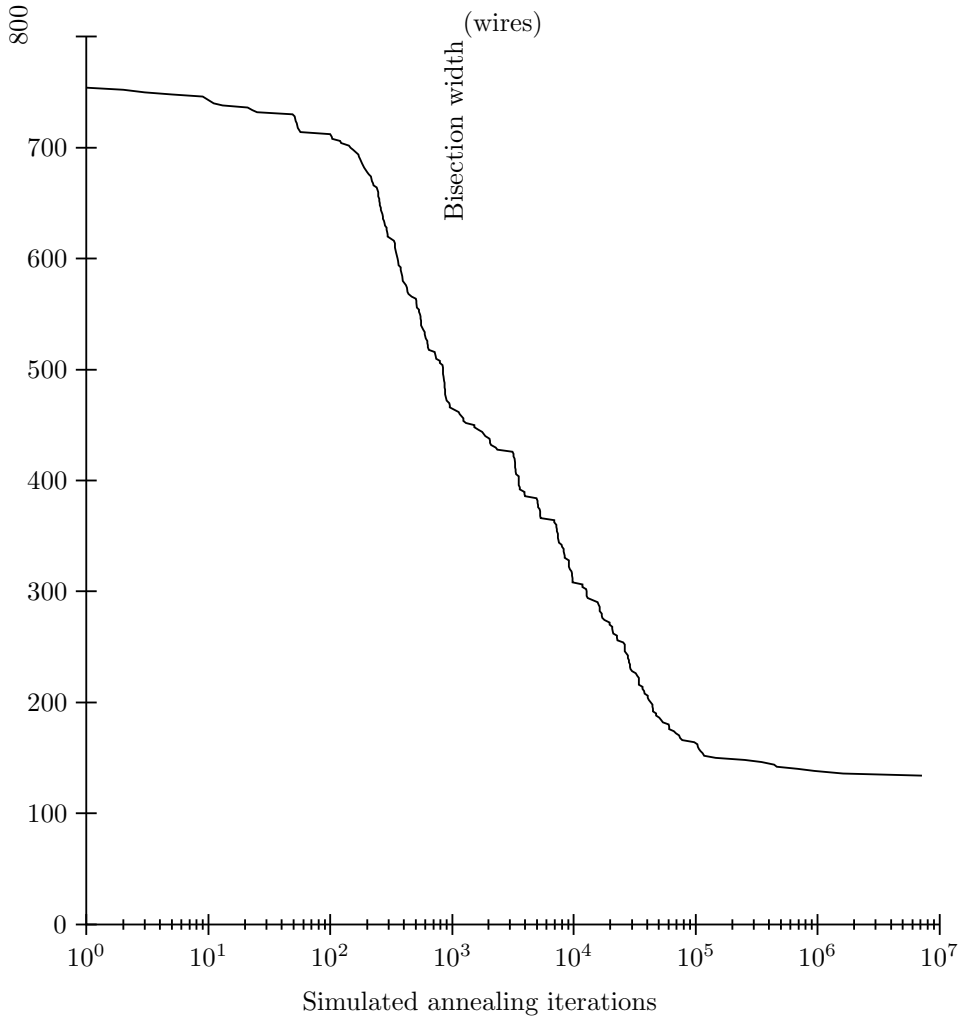


Figure 5.23: Minimum graph-bisection width found by simulated annealing, as a function of annealing algorithm iterations. The graph contained 1000 elements of degree 3, with random connectivity.

Chapter 6

Novel Circuits

A little inaccuracy sometimes saves tons of explanation.

H.H. Munro (Saki) (1870–1916)

The evolution of a new design discipline requires two components: the characterization of bottom-level components (i.e., the existence of a toolkit or library), and the specification of composition rules to interconnect these components. In digital systems, composition rules typically take the form of clocking disciplines and some electrical interface rules (e.g., buffered drivers matched to load capacitances).

In neural analog designs, the composition rules tend to be primarily topological, reflecting both the connectivity limits inherent in a $(2 + \epsilon)$-dimensional implementation technology, and the geometry of the problem in question (e.g., processing of two-dimensional visual images). Certain tasks, such as spatial averaging, lend themselves to solution in the voltage domain, via resistive networks. Other aggregating systems employ the natural summing property of currents [DeWeerth and Mead, 1988, Umminger and DeWeerth, 1988]. Still others utilize statistical properties of "neural" pulse trains for various computations [Tomlinson et al., 1990]. A comprehensive description of several system-level architectures is presented in [**?**].

In parallel with the development of such systems has been the development of component circuits. This section describes several such elements, and discusses potential applications. Descriptions of certain of these circuits have been published previously, and indeed the

circuits have been used as core components in several systems [**?**]. Other circuits are still relatively new and have not been explored fully.

The circuits discussed in this chapter are

1. A novelty filter that performs $O(N^2)$ multiplications with only $O(N)$ transistors
2. A low-noise sampling switch utilizing gated transconductance amplifiers, suitable for high-gain discrete-time differentiators
3. A digital logic family operating on a single-phase clock; we describe a toggle flip-flop cell, and its application to a fully asynchronous up–down counter suitable for integration of bipolar neural pulse trains
4. A composable two-input arbiter, applied to a self-timed retina design frame

6.1 An Isotropic Novelty Filter

An essential component of the computation occurring in biological early vision systems is the *compression* of information. This compression is required to reduce the bandwidth needed to transmit retinotopic data to higher visual centers. Note that this requirement is independent of the parallel need to *encode* visual data, such that spatiotemporal correlations can be reconstructed later.

In this section, we describe a circuit that functions as a "novelty filter" in that it generates a response based on activations within its "receptive field." When the stimulus is uniform (i.e., all inputs are equal), a minimum response is produced. When the activation deviates (in any way) from uniform, the circuit produces a response.

This circuit is interesting on two counts. First, it exhibits a spatially isotropic, or nondirectional, response. Second, it implements an $O(N^2)$ operation with only $O(N)$ devices (where N is the number of inputs in the receptive field).

Spatial isotropy is typically not observed in biological systems, which tend to be organized into antagonistic center–surround receptive fields at the retinal level, and these fields are combined into directionally selective receptive fields (edge detectors) in visual cortex. There is considerable evidence to suggest that these structures arise developmentally [Linsker, 1986b] [Linsker, 1986a]. as a result of local wiring considerations. It is interesting to note that, without raising the dimensionality of this interconnect, we can perform a much more subtle computation (at the expense, however, of increased specificity of wiring).

6.1.1 Series MOS Transistors with Back-Gate Effect

The heart of the novelty detector is a series chain of MOS transistors, the gates of which are independently biased by current mirror inputs (Figure 6.1). In the subthreshold regime [**?**], an MOS transistor can be modeled by:

$$I_d = e^{\frac{\kappa V_g}{V_T}} \left(e^{-\frac{V_s}{V_T}} - e^{-\frac{V_d}{V_T}} \right)$$

where all voltages are referred to the substrate (bulk), and $V_T = kT/q$ is the thermal-equivalent voltage of the environment. The back-gate effect is captured by κ; namely, variations in the gate potentials do not appear in their entirety in the channel potential.

Applying this model to Figure 6.1, we obtain

$$\begin{aligned} I_1 &= e^{\frac{\kappa V_1}{V_T}} \\ I_2 &= e^{\frac{\kappa V_2}{V_T}} \\ I_{\text{out}} &= e^{\frac{\kappa V_1}{V_T}} \left(1 - e^{-\frac{V_x}{V_T}} \right) = e^{\frac{\kappa V_2}{V_T}} e^{-\frac{V_x}{V_T}} \end{aligned}$$

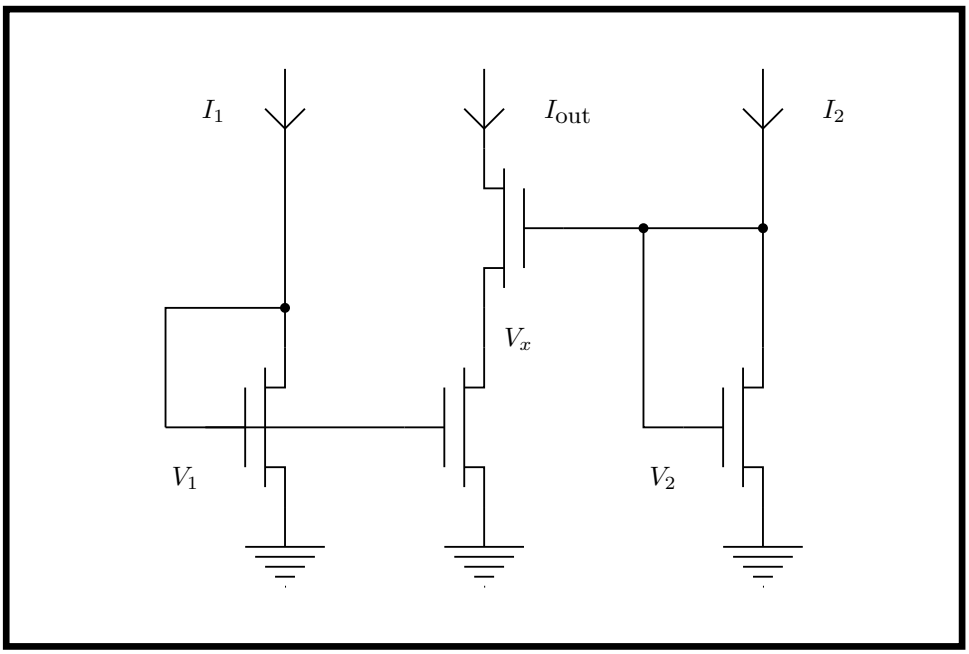


Figure 6.1: Analysis of a series chain of MOS transistors, used in the novelty filter. A subthreshold MOS model incorporating the back-gate effect will be used.

Eliminating V_x, and solving for I_{out} gives

$$I_{\text{out}} = \frac{I_1 I_2}{I_1 + I_2}$$

In the general case of N transistors in series, an identical analysis yields the result

$$\frac{1}{I_{\text{out}}} = \sum_{i=1}^{N} \frac{1}{I_i}$$

6.1.2 Multiple Differential Pair Circuit

The novelty filter is obtained when this circuit is combined with a *multiple differential pair* circuit. A general novelty filter circuit is illustrated in Figure 6.2. The analysis of this circuit is based on the observation that the current bias I_0 is partitioned between the various branches:

$$I_0 = \sum_{i=1}^{N} I_i = \sum_{i=1}^{N} e^{\frac{\kappa V_i}{V_T}} \left(e^{-\frac{V}{V_T}} \right)$$

This expression may be written as

$$\frac{I_i}{I_0} = \frac{e^{\frac{\kappa V_i}{V_T}}}{\sum_{i=1}^{N} e^{\frac{\kappa V_i}{V_T}}}$$

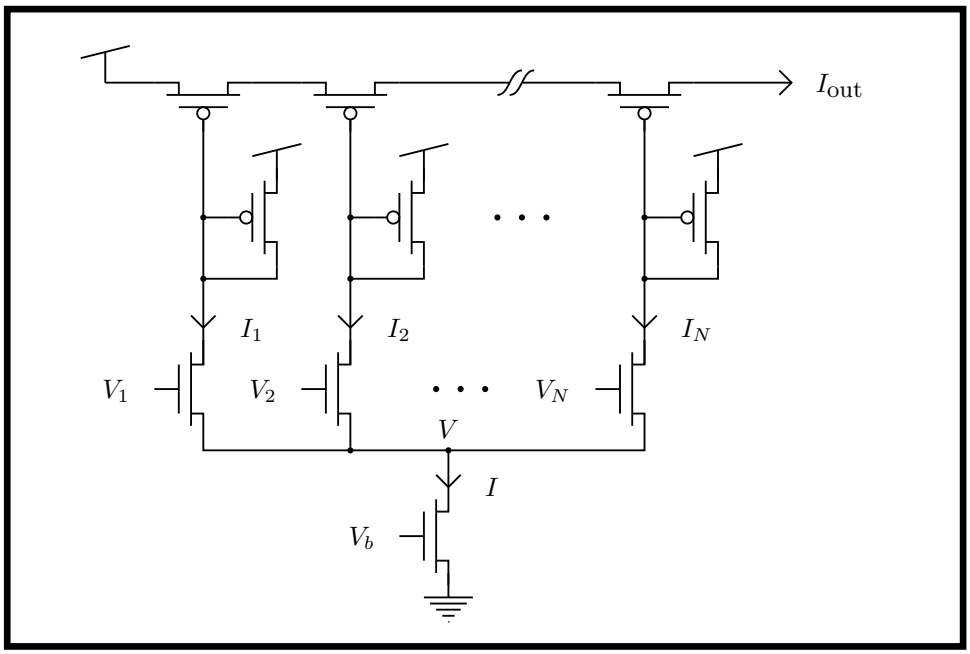


Figure 6.2: Novelty filter circuit. Combining a multiple input differential pair circuit and the series-MOS coincidence circuit described in Section 6.1.1 results in a novelty filter scalable to arbitrary input dimensions.

Connecting the multiple differential pair to the coincidence detector of Figure 6.1 yields

$$\frac{1}{I_{\text{out}}} = \sum_{i=1}^{N} \frac{1}{I_i} = e^{\frac{V}{V_T}} \sum_{i=1}^{N} \frac{1}{e^{\frac{\kappa V_i}{V_T}}} = \frac{1}{I} \sum_{i=1}^{N} e^{\frac{\kappa V_i}{V_T}} \sum_{i=1}^{N} e^{-\frac{\kappa V_i}{V_T}}$$

We can express this equation in terms of *differences* between pairs of input voltages:

$$\frac{1}{I_{\text{out}}} = \frac{1}{I} \sum_{i=1}^{N} \sum_{j=1}^{N} e^{\frac{\kappa (V_i - V_j)}{V_T}}$$

It is useful to rewrite this expression in terms of the symmetric hyperbolic sine function (sinh), using the identity $\sinh^2 x = \frac{1}{4}(e^{2x} - 2 + e^{-2x})$:

$$\frac{1}{I_{\text{out}}} = \frac{1}{I} \left(\sum_{i=1}^{N} \sum_{j=i+1}^{N} 4 \sinh^2 \frac{\kappa \Delta V_{ij}}{2V_T} + 2\binom{N}{2} + N \right)$$

Observing that $2\binom{N}{2} + N \equiv N^2$, we have

$$\frac{1}{I_{\text{out}}} = \frac{1}{I} \left(\sum_{i=1}^{N} \sum_{j=i+1}^{N} 4 \sinh^2 \frac{\kappa \Delta V_{ij}}{2V_T} + N^2 \right)$$

For the case $N = 2$ (reported by Delbrück, [Delbrück, 1990]), this expression reduces to $\frac{1}{I_{\text{out}}} = \frac{4}{I} \cosh^2 \frac{\kappa \Delta V}{2V_T}$.

6.1.3 Summary

A novelty filter has direct applications in systems that apply an energy-based metric for evaluation of image complexities, such as Delbrück's autofocus system [Delbrück, 1989]. It is also likely to be useful in systems such as the retina [**?**], where some evaluation of regional activity can usefully precede image transfer down a bandwidth-limited channel.

In this section, we have ignored the effect of interdevice variations. It is clear that systems employing this circuit will have to be self-calibrating to the extent of "learning" the response of the circuit to various input stimuli. Such a design strategy is common in large-scale neuromorphic systems, and is especially important in the case of reduced dimensionality circuits (e.g., the winner-take-all circuit and this novelty filter), because of the increased importance of individual devices.

Nonetheless, a circuit that performs the equivalent of $O(N^2)$ multiplications with approximately $3N$ devices (and, more important, lays out in $O(N)$ area) is of considerable interest, and merits careful consideration in many diverse applications.

6.2 A Low-Noise Sampling Circuit

Hybrid analog-digital systems have become increasingly popular in recent years. They combine the speed and area efficiency of analog circuits with the ease of design and interface typical of digital systems. Furthermore, they lend themselves to particular design *disciplines* that permit general principles to be applied to different applications of widely varying scale.

In this section, we shall discuss one small aspect of one such discipline, and shall consider a component useful to *clocked* analog designs. Such systems are characterized by analog processing elements (i.e., elements whose inputs and outputs are continuous state variables, such as current or voltage) operating on data that are sampled in time. Examples include sample-and-hold circuits, switched-capacitor circuits, and, in our case, a discrete-time differentiator circuit.

Discrete-time circuits are also used at the component level. Primary applications include frequency compensation of amplifiers by "chopper" stabilization [Hsieh et al., 1981], offset voltage nulling of differential amplifiers [**?**] [**?**], and circuits for interfacing to a time-sampled (and perhaps digital) environment (such as sample-and-hold circuits).

6.2.1 Estimation of Time Derivatives Using Discrete Differences

An example of a circuit employing time-sampled continuous variables is the Caltech RET20 *retina* [**?**]. A focal-plane parallel analog processing surface calculates spatial and temporal derivatives of an image. One particular application for these systems required maximum sensitivity to small signals on a spatially varying (but locally constant) background. Because an external synchronization signal was available, the natural solution involved computing a derivative approximated by the difference between the present value of a signal, and a previously sampled value (see Figure 6.3).

For a transconductance amplifier ($I_{\text{out}} \propto \Delta V_{\text{in}}$), the output signal from this circuit is

$$\begin{aligned} V_{\text{out}} &= \frac{1}{C}\int_0^T i\,dt + V_{\text{hold}} \\ &= \frac{1}{C}\int_0^T I_0(v - V_{\text{hold}})\,dt + V_{\text{hold}} \\ &= \frac{I_0 T}{C}(\langle v\rangle - V_{\text{hold}}) + V_{\text{hold}} \end{aligned}$$

If the output were sampled at the end of time T, the change in output voltage ($V_{\text{out}} - V_{\text{hold}}$) would approximate the derivative of the input.

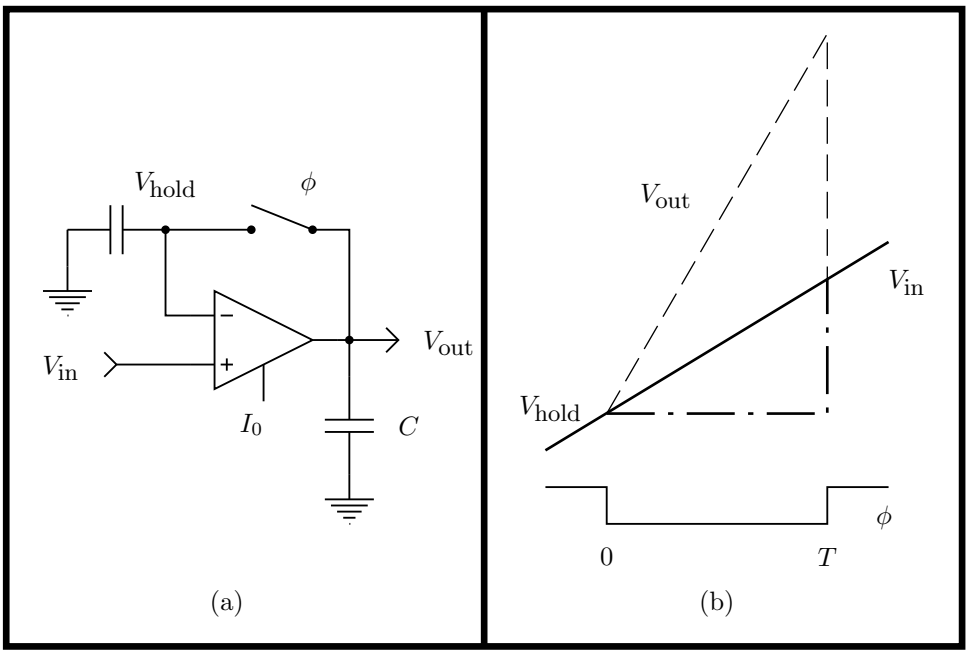


Figure 6.3: Discrete-time derivatives. We obtain an approximation to the derivative of a waveform by comparison of a sampled value with the current value. Typically, we assume that the derivative is constant during the window of integration.

6.2.2 Established Circuits for Sampling of Voltages

Considerable work has been done in the area of bilateral analog switches, primarily for switched-capacitor applications [McCharles and Hodges, 1978] [Allen and Sanches-Sinencio, 1984]. The basic operational principle is that gating charge on a capacitor between nodes at different potentials produces a current that is proportional to the potential difference (i.e., the system emulates a resistor).

A popular analog switch is a single MOS transistor. Although this system has the virtue of simplicity, it has a considerable drawback: The sampled voltage is in error, due to injection of channel charge onto the holding node. A simple model for this channel charge (in the above-threshold region in which such devices typically operate) is

$$Q = C_{\text{ox}}(V_{\text{G}} - V_{\text{TE}})$$

A reasonably accurate approximation [Maher, 1989] is that this channel charge is equally divided between the source and drain, when the gate clock goes low. This injected charge gives rise to a voltage error that is multiplied by the open-loop gain of the amplifier in Figure 6.3.

Many attempts have been made to reduce this injected charge. Several approaches are illustrated in Figure 6.4. The primary technique is to use MOS devices of opposing polarity

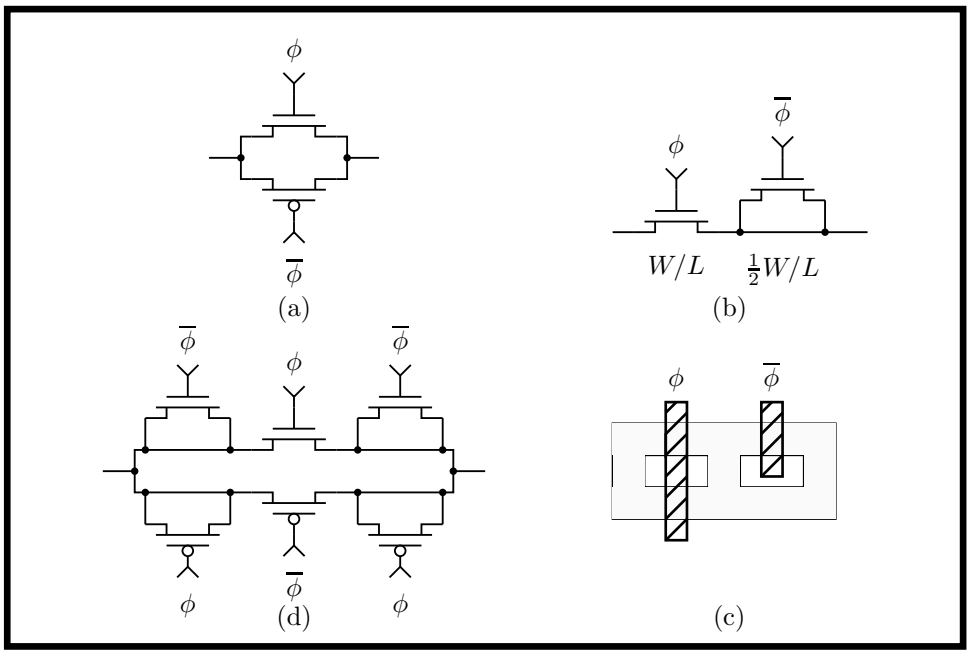


Figure 6.4: Four approaches to reducing injected charge noise. (a) A device of the opposite polarity is clocked with an inverted clock signal, in an attempt to annihilate the noise charge. (b) Compensation with a device of the same polarity, based on the assumption of equipartition of channel charge. (c) A layout for the circuit in (b), showing matching of overlap capacitances. (d) Combination of all of these approaches. In addition to occupying large area and requiring complementary clocks, such systems suffer the disadvantage of limited input ranges of operation, and sensitivity to clock skew.

to cancel the channel charge. Because such devices are poorly matched, geometric effects must also be used to minimize the injection of charge onto the sensitive node; dummy transistors with matched overlap capacitances are used.

A careful analysis [**?**] reveals that the behavior of the system is complicated by the need to consider the capacitance on *both* sides of the switch (see Figure 6.5). The geometric charge-compensation strategies of Figure 6.4 are, in general, accurate only when C_i and C have the same value. Otherwise, the details of the wave shape of the clock signal become important, as does the operating voltage V. Of course, a solution that requires a large C_i is unattractive because of its inefficient use of silicon area.

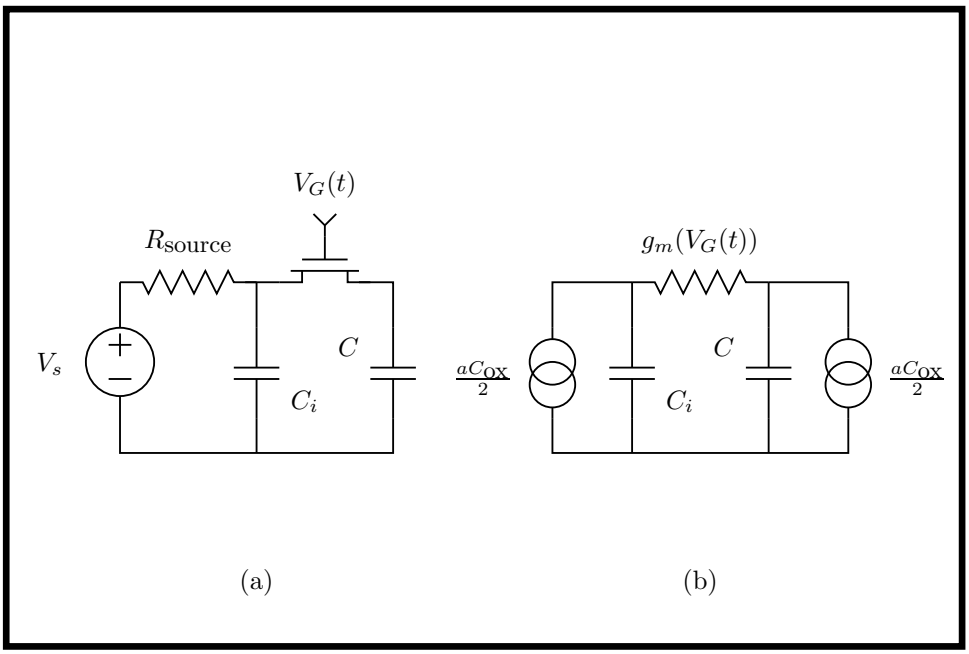


Figure 6.5: A model for MOS channel-charge injection [**?**].

6.2.3 A Circuit for Reduced Charge Injection

A traditional approach to increasing noise immunity is the use of differential input circuits. In fact, most switched-capacitor filters use single transistors as switches for *both* inputs, thereby using the virtual ground at the input of an operational amplifier to compensate for any injected noise charge.

The approach we have taken retains the differential component, but uses it to partition the injected noise charge equally between the two sides of the switch. The circuit is illustrated in Figure 6.6; it is simply a transconductance amplifier [**?**] with a clock signal applied to the bias device. When this bias current is reduced to zero, the output capacitor is effectively isolated from the input. To first order, no noise charge is injected onto C, because the channel charge in Q_1 is symmetrically divided between both branches of the differential pair (which have identical operating points in the follower configuration).

In addition, the clock does not need be rail to rail, thereby decreasing $\frac{dV}{dt}$ even further. Also, a single-phase, unipolar clock is attractive from the system interface perspective (i.e., there is no need to generate nonoverlapping phases, and no need to worry about clock skew in distributing these multiple phases).

Also, any capacitive clock feedthrough is decreased by the cascode connection of the differential pair. As a worst-case approximation that ignores the differential nature of the circuit, we can consider the capacitive feedthrough due to overlap capacitances from clock gate to

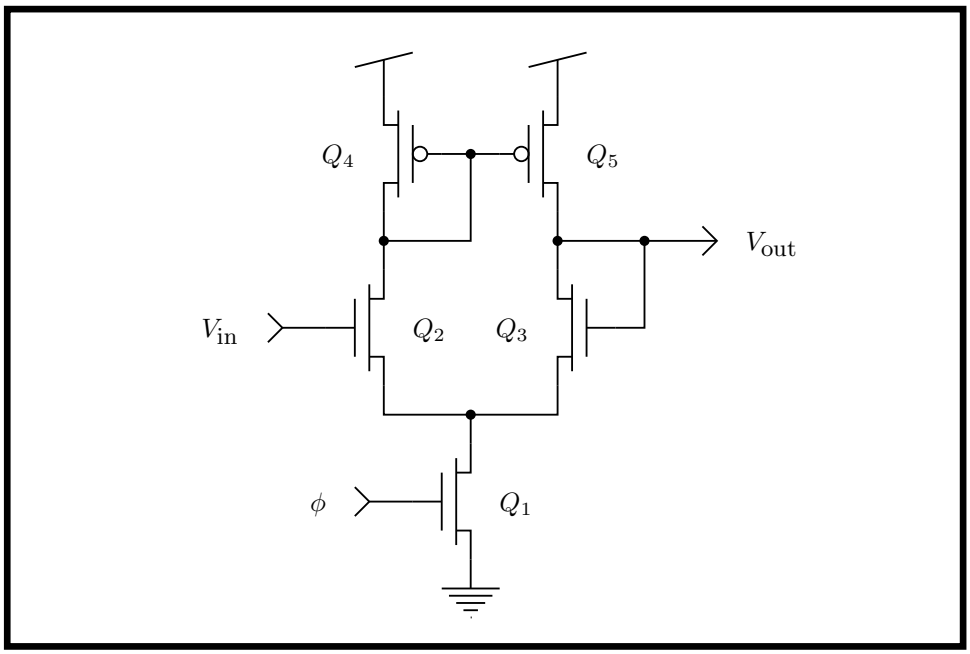


Figure 6.6: Low-noise sample-and-hold circuit.

output capacitor (see Figure 6.7). It is easy to show that

$$\begin{aligned}\Delta V_C &= \Delta V_\phi \left(\frac{C_{\text{ov}}^2}{CC_1 + 2CC_{\text{ov}} + C_1C_{\text{ov}} + C_{\text{ov}}^2} \right) \\ &< \Delta V_\phi \left(\frac{C_{\text{ov}}^2}{CC_1} \right)\end{aligned}$$

where C is the sampling capacitor, C_1 is the capacitance of the common node of the differential pair, and C_{ov} is the transistor gate–drain overlap capacitance. For typical values of $C = 0.2$ pF, $C_1 = 0.015$ pF, $C_{\text{ov}} = 0.0005$ pF, and a clock voltage change of $\Delta V_\phi = 0.5$ V, we obtain a coupled-noise voltage change of 42 μV. Assuming that C is the input to an open-loop stage of 50 dB gain, this capacitive-coupling noise will contribute a mere 13 mV error *at the output*.

The principal disadvantage of this circuit is that we no longer have a bilateral switch, but rather have a unidirectional one. Also, this "switch" is actually an active system, not merely a passive charge-transfer system as required by switched-capacitor systems (to emulate resistances). It is, however, ideally suited for sample-and-hold systems of the sort required for a RET20-style discrete-time differentiator [**?**]. It is also appropriate for video-speed amplifiers and multiplexors, of the sort found in the retina chips' scan circuitry. Finally, it could be incorporated into Vittoz's dynamic current mirror [Wegmann and Vittoz, 1989] to eliminate the single largest remaining source of error in an otherwise high-precision current mirror design.

A second limitation is that the input offset voltage of the differential pair within the switch is propagated to the output (after all, we are using a transconductance amplifier in a unity-

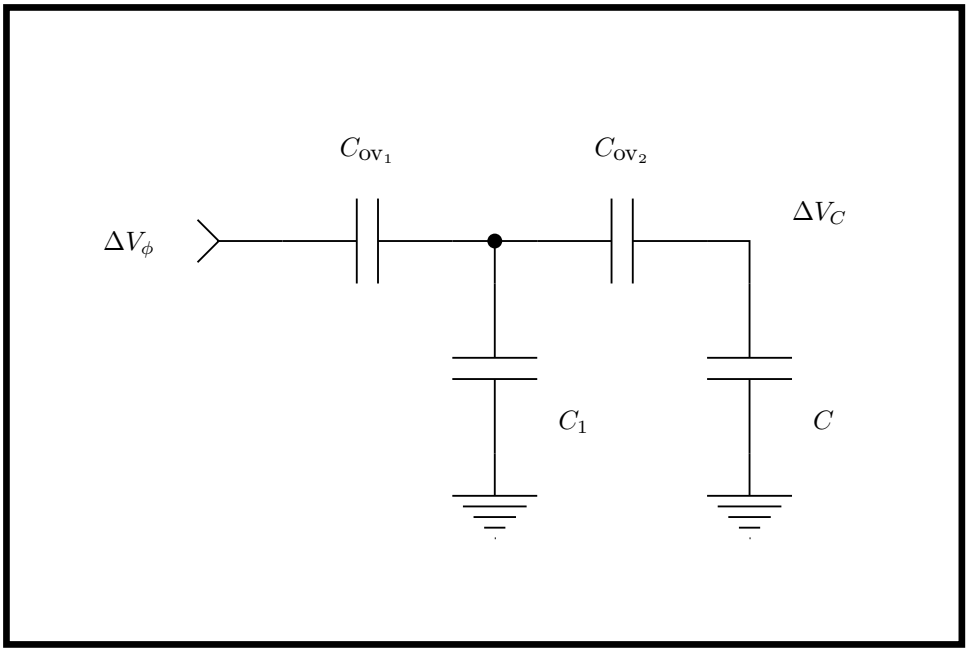


Figure 6.7: Analysis of clock feedthrough in the low-noise sample-and-hold circuit of Figure 6.6. Because Q_2 and Q_3 are identically biased, any noise charge injected by the bias device Q_1 will be equally divided between the two branches, resulting in no net charge at the output (due to current mirror Q_4–Q_5). In this figure, C_{ov_1} corresponds to the gate–drain overlap capacitance of Q_1, C_{ov_2} is the gate–source overlap capacitance of Q_3, C_1 is the diffusion capacitance of the common node in the differential pair, and C is the capacitance of the output node.

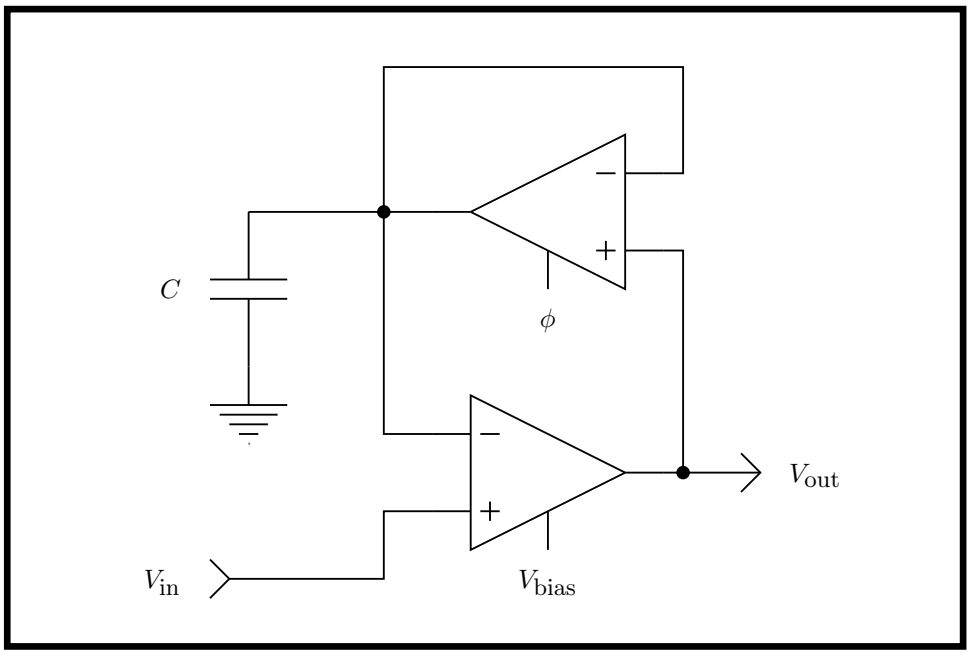


Figure 6.8: The discrete-time differentiator.

gain configuration). This behavior is not a problem in our discrete-time differentiator, because this offset appears inside a negative feedback loop with reasonably high gain. In an open-loop sample-and-hold application, this offset would need to be corrected, perhaps through sampled-offset techniques, by laser-trimming, or by programming floating nodes with ultraviolet light.

6.2.4 A Low-Noise Discrete-Time Differentiator

If we apply the low-noise sample-and-hold switch to the discrete-time differentiator of Figure 6.3, we obtain the circuit shown in Figure 6.8. It is interesting to note that the circuit is *identical* to the continuous-time differentiator described in Chapter 10 of [**?**] (the "diff2" circuit).

In the diff2 circuit, a constant bias voltage is applied to the ϕ input, and the "switch" transconductance amplifier acts as a low-pass filter, effectively storing a smoothed, delayed version of the input. In the discrete-time mode of operation, we sacrifice this smoothing for better control over the delay. The discrete-time circuit also avoids the potential second-order instability of the diff2, that is due to the additional time constant introduced by the capacitance of the V_{out} node. Consequently, the storage capacitor C of the discrete-time differentiator can be quite small.

Experimental data from the discrete-time differentiator are presented in Figure 6.9. We have tested circuits with over 70 dB signal-to-noise ratio.

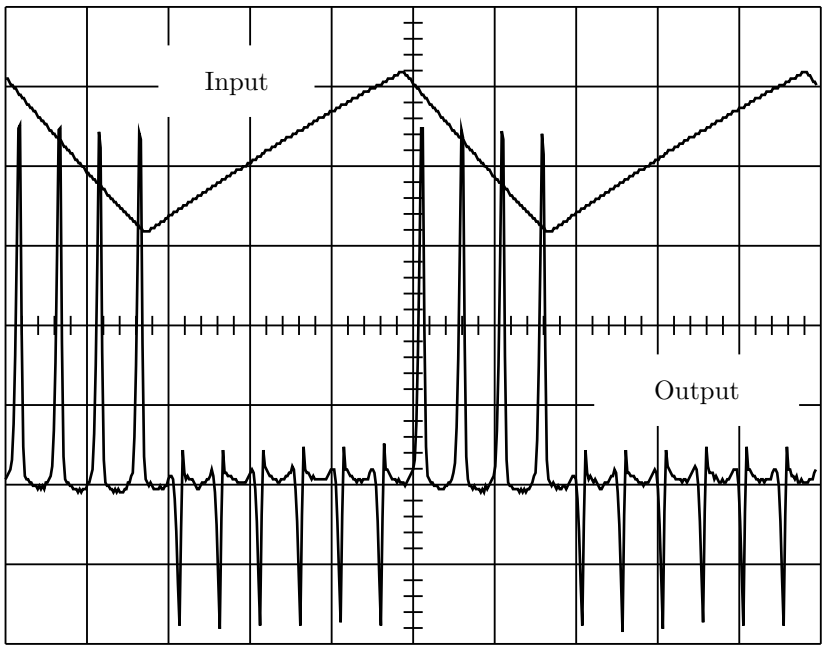


Figure 6.9: Operation of the low-noise discrete-time differentiator. A 1 kHz 1 $V_{p\text{-}p}$ sine wave was applied to the ϕ input of the discrete-time differentiator. An asymmetric 100 Hz 40 $mV_{p\text{-}p}$ triangle wave was applied to the input; the output waveform scale is 0.5 V/division. In practice, systems with a signal-to-noise ratio of over 70 dB have been implemented using this technique [**?**].

In summary, the low-noise switch described in this section is a valuable addition to the system designer's toolkit. The circuit provides a useful interface between continuous- and discrete-time subsystems, and can be used to construct compact, high-performance components, such as differentiators and dynamic current mirrors.

6.3 A Single-Phase Latch

An essential component of traditional sequential logic design is the synchronous latch, a storage element whose state changes only at clock *transitions*. Typically, such latches are placed between logic elements that are strictly combinational; clocking the latches with a single-phase clock provides a hazard-free design discipline.

Unfortunately, at the transistor level, devices are *level* sensitive, not edge sensitive. In conventional digital VLSI design, this constraint has given rise to design disciplines involving multiple clock phases. Typically, each clock phase enables resynchronization registers. In practice, these registers are often as simple as dynamic state-storage nodes gated by pass transistors. This explicit *pipelining* gives rise to constraints on each of the clock phases; this procedure often places severe demands on the design tools (since the clocking discipline becomes an integral part of the specification of the interface between circuit subsystems). Also, there are serious practical difficulties involved in distributing multiple phases (i.e., limiting clock skew), and additional circuit overhead involved in computing derived clocks.

Thus, although level-sensitive clock disciplines are very flexible, and ultimately hold the promise of maximum performance, they are needlessly complicated for many applications. In particular, novice designers would find extremely attractive the ability to apply familiar design techniques involving edge-triggered latches.

In Sections 6.3.1 to 6.3.2, we describe a new single-phase latch design. It is relatively small, comprising 14 transistors, yet can be designed to be risk-free. Although similar to master–slave designs, it is a true edge-sensitive implementation: The output becomes available immediately, and is not delayed until the complementary clock transition, as in traditional master–slave designs. This cell can be employed as either a D flip-flop or a T flip-flop. It has been used successfully in various systems, including scanning and self-timed retina design frames, and an asynchronous up–down counter for neural integrators.

6.3.1 A Level-Sensitive Latch

The edge-sensitive latch is based on complementary set–reset logic (CSRL) (see Figure 6.10 and [**?**]). A CSRL stage consists of two cross-coupled inverters, with a power-up transistor Q_3, and two access devices Q_1 and Q_2. In the following discussion, complementary inputs are assumed. When the clock ϕ is high, the flip-flop is disconnected from the V_{dd} rail, and the internal state is imposed by the environment through the access devices. In this condition, that the inverter pair is able to restore *one* of its internal state nodes, because the Q_6 and Q_7 devices are always connected to ground, and hence can pull down *one* of nodes Q and $\overline{Q}$. This ability plays an important role in our discussion of the operation of the edge-triggered latch.

When the clock goes low, the access devices are disabled, and the power-up device Q_3 turns ON, making the latch fully static. The conventional application of CSRL [**?**, Wawrzynek, 1987]

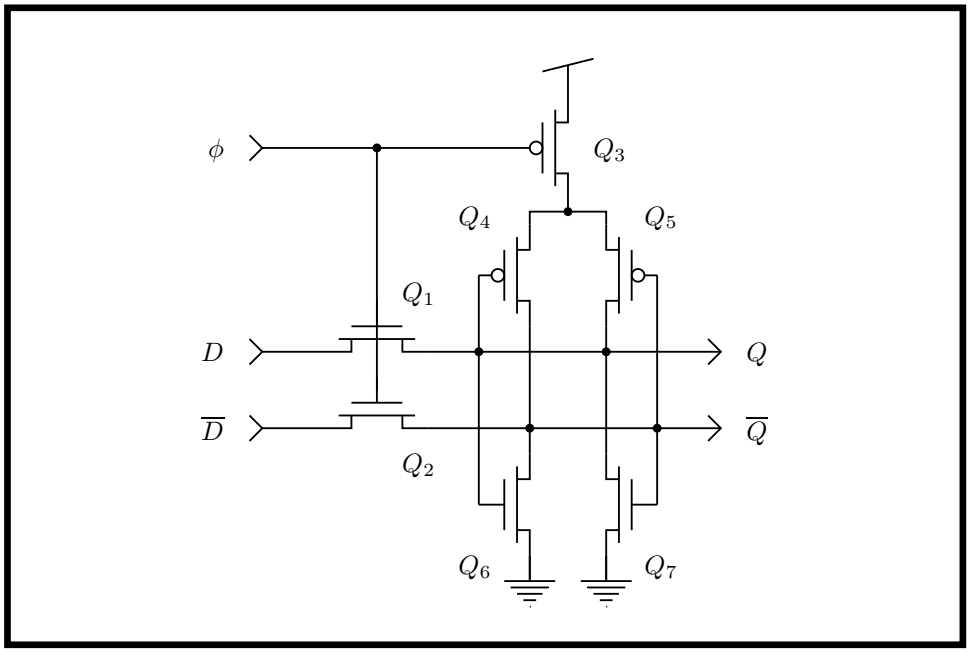


Figure 6.10: A single CSRL stage. When power is removed from the cross-coupled inverters Q_4/Q_6 and Q_5/Q_7, the access devices Q_1 and Q_2 are enabled, and can drive the external state $D/\overline{D}$ into the latch. When the clock goes low, the drive from the access devices is removed, and Q_3 turns ON, making the latch fully static.

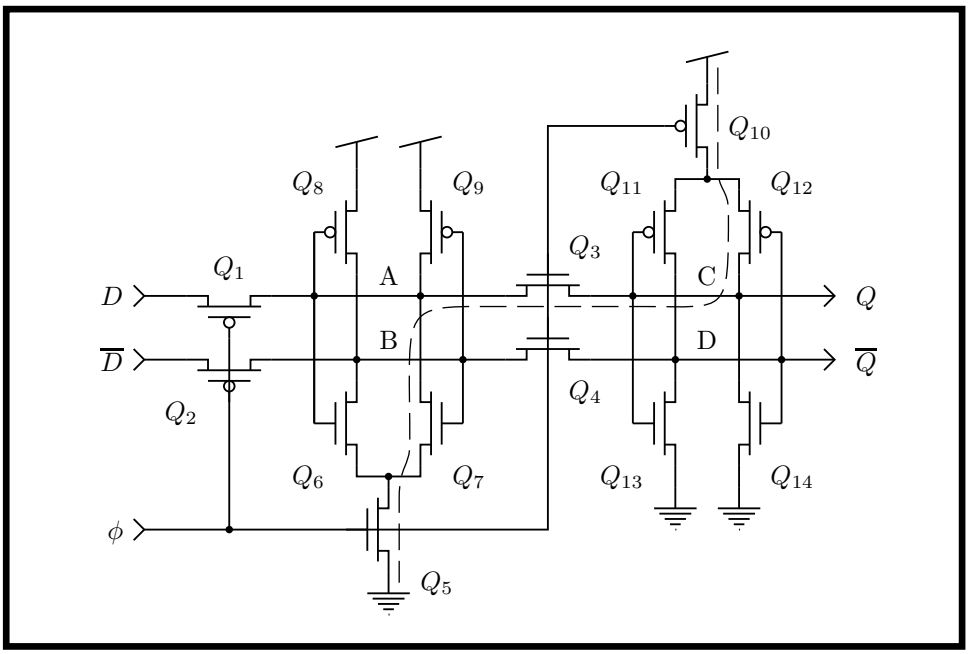


Figure 6.11: An edge-sensitive CSRL latch.

employs pipeline stages of CSRL latches, clocked by a two-phase nonoverlapping clock. The principal virtues of this approach are (1) only two clock phases are required (no locally generated complements are needed), (2) operation is fully restored and fully static, and (3) dense computational primitives are available that exploit the dual-rail signal inputs (and a general switching form described in [**?**]).

6.3.2 An Edge-Sensitive Latch

Instead of the use of two identical CSRL stages clocked by two nonoverlapping clock phases, we propose the use of two *complementary* stages clocked by the same clock (see Figure 6.11). This strategy is similar to the π-latch–μ-latch approach proposed by Denyer [McGregor et al., 1987]; the key difference is that when we employ CSRL stages we require only 14 transistors for fully static operation, instead of the 32 Denyer's design uses. In fact, our transistor count is quite competitive with the 10 transistors used in their *dynamic* design, especially considering the concomitant advantages of dual-rail signal representations.

The operation of the latch is conceptually simple: When the clock is low, the right-hand stage is powered, and is isolated from the left-hand stage. The left-hand stage, on the other hand, is driven by its inputs. When the clock goes high, the influence of the inputs is removed, and the left-hand stage latches its state. Furthermore, pass transistors Q_3 and Q_4 permit the new state to be propagated to the right-hand latch (and to the outputs); when the clock goes low, the outputs are restored to the rails, and the left-hand stage becomes sensitive to its inputs again.

6.3.3 Analysis of the Edge-Sensitive Latch

A more careful analysis is required to guarantee the direction of data transfer described in Section 6.3.2. A potential difficulty arises because pass transistors are inherently bidirectional elements, and the current state latched in the right-hand latch could corrupt the "next-state" data to be latched from the left, during the conduction overlap period as the clock rises with a finite rise time. The crucial observation that facilitates the analysis of the system is that, although CSRL stages can be thought of being powered down, they are, in fact, always capable of full current drive toward *one* of the power rails.

Referring again to Figure 6.11, we observe that, as ϕ rises, Q_3 and Q_4 begin to turn ON. There are four state variables we must consider; without loss of generality, suppose the initial conditions are $V_C = 5$ V, $V_D = 0$ V, $V_A \approx V_T$, and $V_B = 5$ V. Consequently, Q_8 drives node V_B high, just as Q_{13} pulls node V_D low. Thus, we can limit our consideration to nodes V_A and V_C, whose drives are a function of the clock voltage. Initially, V_C is driven high by Q_{10}–Q_{12}. Node V_A is the critical node: It is pulled up by Q_3, and pulled down by Q_7–Q_5 (it is also pulled down by Q_1, but, to be conservative, we ignore this current).

Q_3 is always saturated, so above threshold we have [Vittoz, 1989]

$$I_{Q_3} = \left(\frac{W}{L}\right)_{Q_3} k_n(\phi - V_{T_n} - nV_A)^2$$

Assuming Q_7 is ON and is ohmic, the current out of node V_A is limited by the current through Q_5:

$$I_{Q_5} = \left(\frac{W}{L}\right)_{Q_5} k_n\left(2(\phi - V_{T_n})V_A - V_A^2\right)$$

Clearly, a conservative *sufficient* condition for the correct operation of the latch is

$$\begin{aligned} I_{Q_5} &> I_{Q_3} \\ \Rightarrow \quad \frac{(W/L)_{Q_5}}{(W/L)_{Q_3}} &> \frac{(\phi - V_{T_n} - nV_A)^2}{2(\phi - V_{T_n})V_A - V_A^2} \end{aligned} \tag{6.1}$$

We select a noise margin for safe operation (e.g., we allow V_A to rise to 1.5 V), and we consider the range of possible voltages for ϕ. For a typical 2μ CMOS process, Figure 6.12 indicates safe values for the access-transistor geometry ratio.

Clearly, this analysis is conservative in several respects. First, no provision is made for decreased current drive from the right-hand stage as the clock rises (and the drive of Q_{10} decreases). Second, a conservative noise margin of approximately 1 V has been chosen for V_A (in practice, V_A could be allowed to rise to about 2.5 V before failure; only when V_A and V_B cross over does actual failure occur). Third, no account was taken of the additional current drive through access device Q_1, which acts to decrease the likelihood of failure.

This analysis also presents an attractive *intuitive* explanation of why the single-phase latch works: The conflict is between transistors *of the same type*. Thus, we can guarantee safe

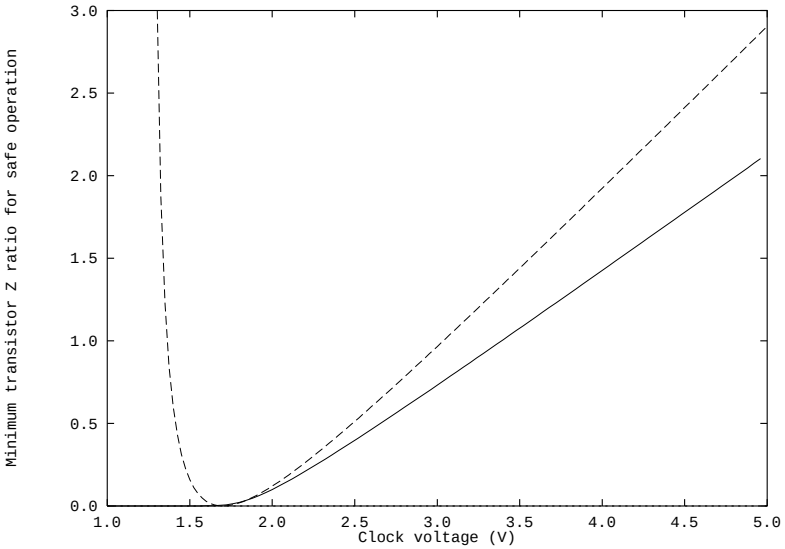


Figure 6.12: Minimum geometry factor for safe operation of the edge-sensitive latch. These values assume a typical CMOS process with $V_{T_n} = 1$ V, and $n = 1.4$. The dashed line is for Equation 6.1; the behavior at low voltages is an artifact, as the assumption of above-threshold saturation operation is violated. The solid line is obtained by applying a continuous model (above and below threshold, ohmic and saturation regimes) developed in [Vittoz, 1989], and repeating the analysis of Section 6.3.3. No account is taken of capacitive coupling between the access devices' gate–source overlap capacitance with the diffusion capacitance of node V_A (typical geometries reveal less than 0.2 V coupling, however).

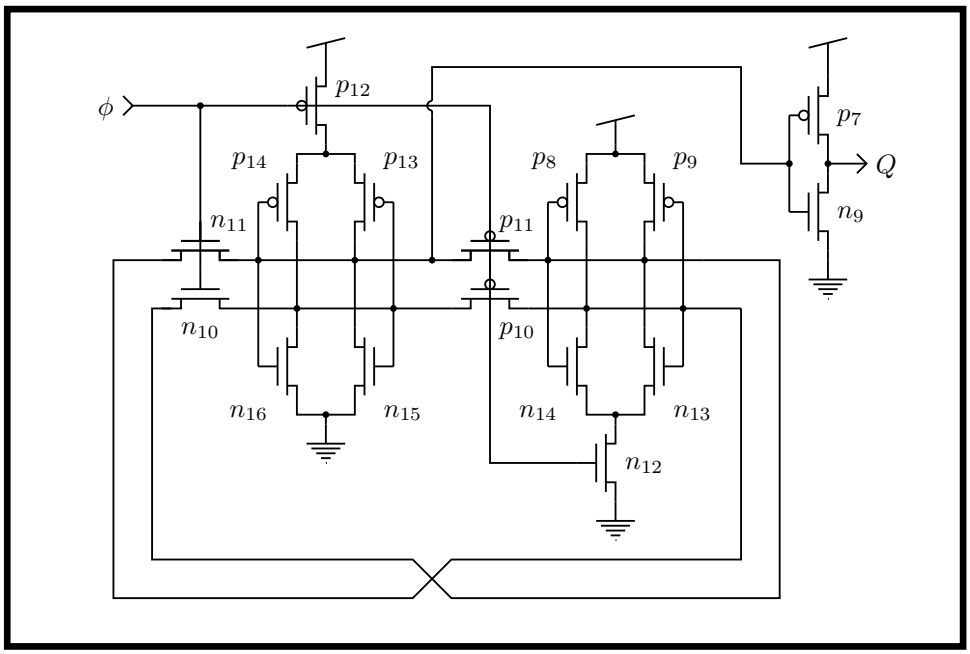


Figure 6.13: A toggle flip-flop cell using the single-clock CSRL discipline. The output inverter is included for increased drive, sharpening the output signal, and isolating the electrical environment within the cell. A typical T-flop would have two such buffered outputs (Q and $\overline{Q}$).

operation, effectively independent of fine details of process parameters and variations, simply by elongating the access pass transistors (a ratio of $(W/L)_{Q_5}/(W/L)_{Q_3} = 4$ is typically chosen, and has been observed experimentally to be safe over the current range of MOSIS processes). Finally, because only transistors of the same type are ever in conflict, it is safe to alternate n-CSRL with p-CSRL stages (e.g., to construct shift registers).

6.3.4 Single Phase Toggle Flip-Flops

We can exploit this closure property, which allows composition of alternate n- and p-CSRL stages, in a more direct way: By connecting the D and Q (and $\overline{D}$ and $\overline{Q}$) nodes of a *single* stage, we obtain a toggle flip-flop (see Figure 6.13). To maintain the electrical symmetry of the nodes within the T-flop, we buffer the outputs of the cell with inverters. The analysis for correct operation is essentially unchanged from that in the previous section. We have observed experimentally that, with the addition of the buffer inverters, a T-flop stage output can reliably drive the toggle-clock input of a subsequent stage (see Figure 6.14). In fact, the operation of the toggle cell is so robust that even very small clock signals can be used to toggle the cell (see Figure 6.15).

CH1 DC 2V 1ms AVG; CH2 DC 2V 1ms AVG;

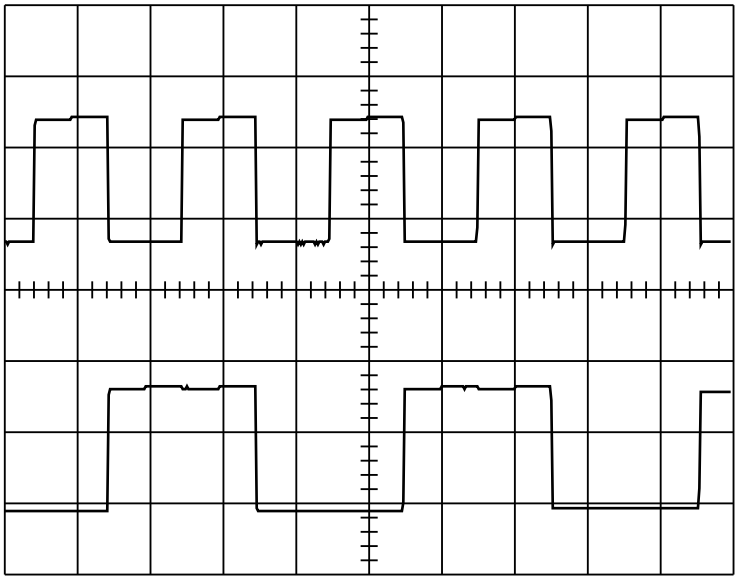


Figure 6.14: Two-bit binary counter implemented with toggle flip-flop cells. These experimental data are the outputs of the first two stages of a binary ripple counter.

CH1 DC 2V 5us AVG; CH2 DC 2V 5us AVG;

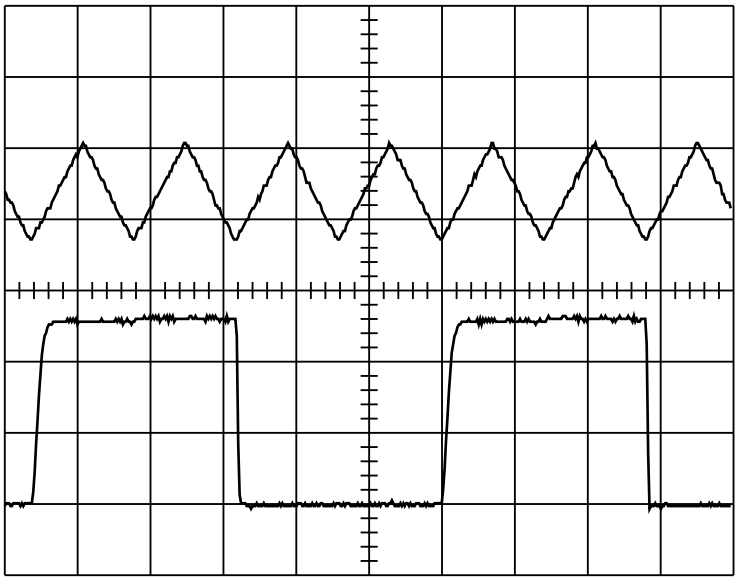


Figure 6.15: Nonrestored clock applied to toggle flip-flop cell. These experimental data illustrate the robustness of the toggle flip-flop cell against variations in clock rise- and fall-time, and in amplitude of clock waveform. The output is from the *second* stage of a binary ripple counter.

6.3.5 Asynchronous Up–Down Counters

A direct application of the edge-triggered toggle flip-flop cell is a binary ripple counter. Either an up-counter or a down-counter can be constructed, by selecting the $\overline{Q}$ or Q output (respectively) of a stage to drive the clock input of the next stage. In this section, we describe two interesting extensions to this procedure, to construct a universal cell for Gray code counters, and for combining two binary up-counters to provide an asynchronous up–down counter.

A Scalable Gray-Code Counter

A well-known disadvantage of binary ripple counters is their passage through spurious states as the carry signal propagates down the chain. For example, in going from the 0111 state to the 1000 state, a ripple counter will (probably) pass through the states 0110, 0100, and 0000. This behavior is unattractive for asynchronous systems (e.g., D/A converters) that are always sensitive to transitions in their inputs. Note that this behavior is independent of the counting *rate,* and is fundamentally different that the high count-rate dynamic instability of multiple carries propagating simultaneously down the chain.

Systems that depend on noise-free transitions between successive states, such as shaft-encoders, rely on Gray-code encodings, where adjacent states differ by exactly one bit. Figure 6.16(a) illustrates this code; Gray code can be converted to binary by exploiting the reflected-binary nature of the code (Figure 6.16(b)).

We can construct a Gray-code counter by observing the following property of the Gray-code sequence: At every clock transition, we flip the least-significant bit that takes us to a state that has not been already visited. Figure 6.17 shows an even more direct interpretation: The bit that changes is the one that describes the extent of carry propagation in the corresponding binary ripple counter. Figure 6.17b shows a direct implementation of this architecture; the result is an arbitrarily scalable Gray code counter, in which each stage is identical to the preceding stage (except for the last stage, which generates the most significant bit (MSB)). Sample output from a fabricated test structure is shown in Figure 6.18.

An Asynchronous Up–Down Binary Counter

We can synthesize an asynchronous $N-1$-bit up–down counter from two $(N-1)$-bit up-counters. Each $(N-1)$-bit counter can count in the range $[-M-1, M]$, where $M = 2^{N-2}-1$. The problem we must solve is how to handle the wrap-around that occurs when each of these counters overflows.

Let U be the state of the up-counter, and $u = U_{N-2}$ be the value of the MSB of the up-counter. Similarly, D and d refer to the down-counter's state and MSB value. Let X

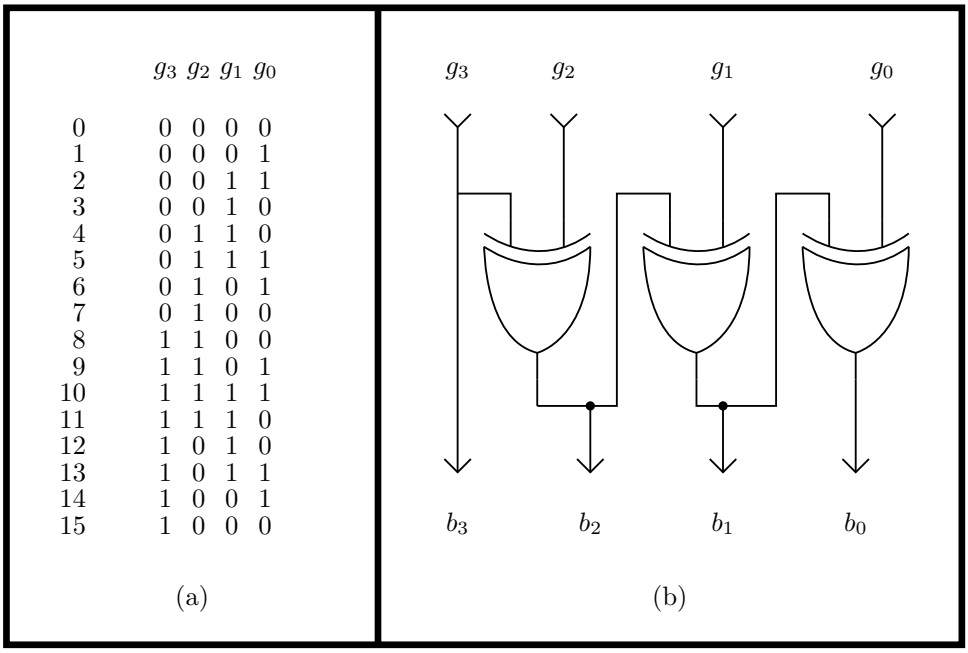


Figure 6.16: The Gray code. (a) Adjacent codewords differ by exactly one bit, thereby preventing spurious states between adjacent codewords. (b) Conversion of Gray code to binary, exploiting a recursive interpretation of the Gray code: If the most significant bit (MSB) of a subword is set, we must invert the remaining bits in the subword before interpreting the result.

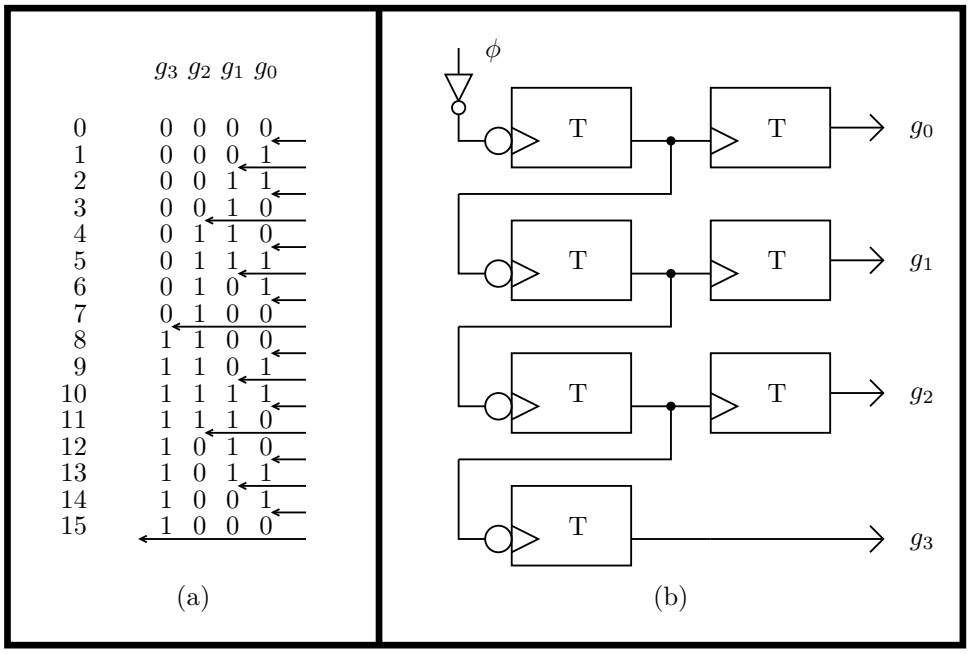


Figure 6.17: A scalable Gray-code counter. (a) A general algorithm for selecting which bit to toggle, when counting in a Gray code. (b)Use of a binary ripple up-counter to generate the toggle signal for that particular bit.

CH1 DC 2V 2ms NORMAL; CH2 DC 2V 2ms NORMAL;

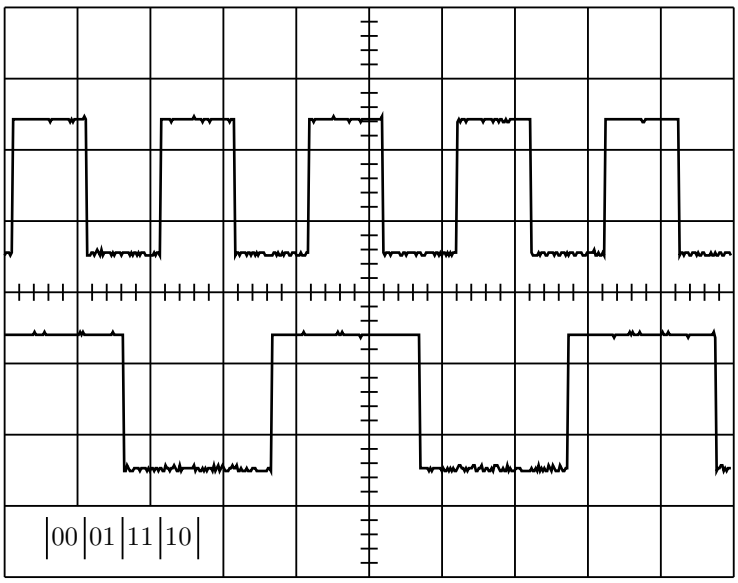


Figure 6.18: Sample output from a Gray-code counter. Both signals are logically inverted, and g_0 is displayed above g_1.

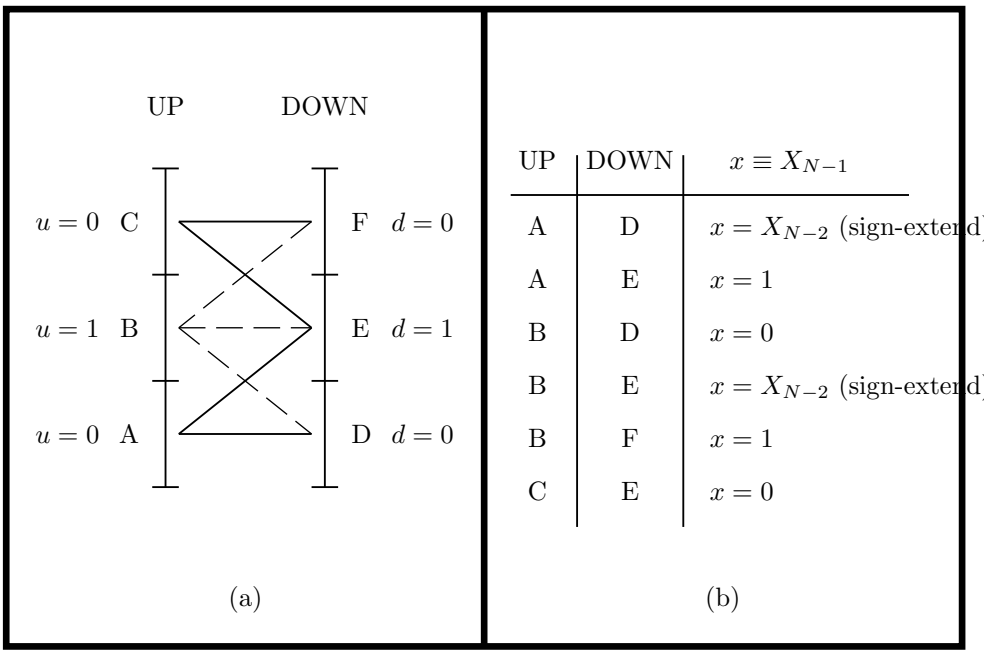


Figure 6.19: Synthesis of an up–down counter from two up-counters. We can interpret the difference between two up-counters uniquely, if we also keep track of in which subrange ($u = \{0, 1\}, d = \{0, 1\}$) the two up-counters are operating. In the worst case, the dynamic range of the synthesized difference output is $[-M, M]$, corresponding to $(N - 1)$ bits of precision.

be the difference between U and D, computed by a standard two's-complement $(N - 1)$-bit subtraction (defined by $X = U + \tilde{} D + 1$, where "~" is the bit-wise logical negation operator).

We use an $(N-1)$-bit subtracter, and synthesized the MSB by a finite-state machine with u and d as inputs. For any given value in the up-counter, we can identify uniquely the correct difference if the down-counter's value lies in the range illustrated in Figure 6.19. If the down-counter's value lies outside this range, additional bits of state information are required—it would be simpler to extend the counters. Figure 6.19 also describes the interpretation that must be assigned to the $(N - 1)$-bit difference X (i.e., the value of the MSB ($x = X_{N-1}$) required for the correct interpretation of the difference, in two's-complement notation).

We can reduce the state-transition diagram (Figure 6.20) to a Huffman flow table (Figure 6.21). If we ignore the possibility of overflow errors, and assume that only fundamental-mode operation is allowed (i.e., only single transitions of u, d, or x are permitted), this flow table collapses to that shown in Figure 6.22, where A is the union of states 1, 2, and 3, and B is the union of states 3, 4, and 5.

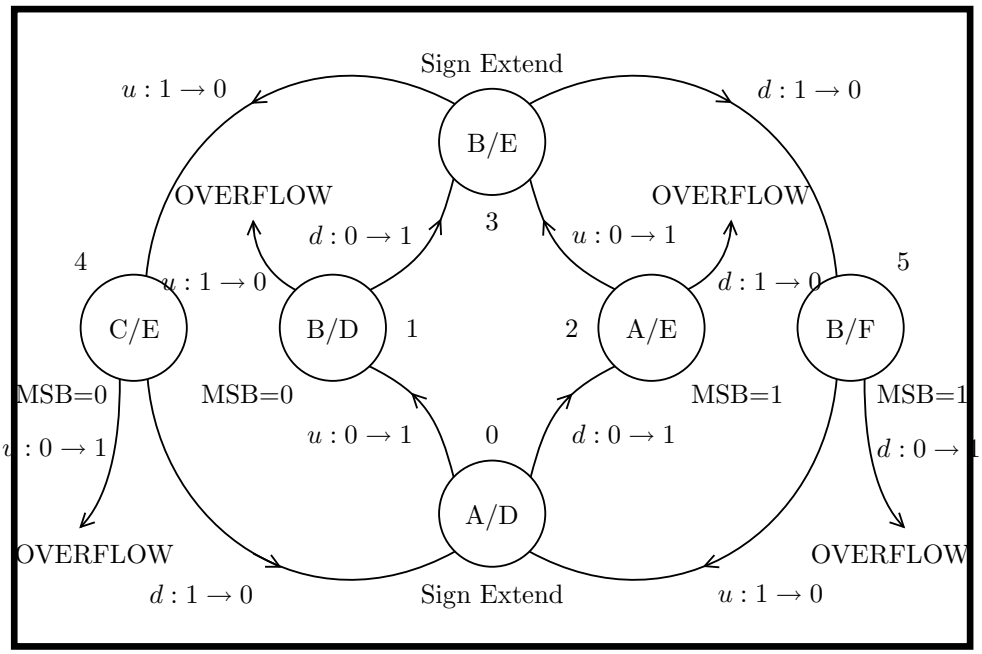


Figure 6.20: State-transition diagram for up–down counter. Transitions in the most significant bits (u and d) of the two up-counters are monitored; in conjunction with the current MSB (x), they determine the next state.

At this point, state encodings can be assigned, and Boolean logic minimization can be applied to obtain the following finite-state machine. For $S = \{A, B\} = \{0, 1\}$,

$$\begin{aligned} s^{\text{next}} &= u \cdot d \;+\; u \cdot s^{\text{present}} \;+\; d \cdot s^{\text{present}} \\ x^{\text{next}} &= \overline{u} \cdot \overline{d} \cdot x^{\text{present}} \;+\; \overline{u} \cdot d \cdot \overline{s} \;+\; u \cdot d \cdot x^{\text{present}} \;+\; u \cdot \overline{d} \cdot s \end{aligned}$$

If we want to detect overflow, the state flow table can be collapsed only to four states, and an additional state variable (w) is required:

$$\begin{aligned} s^{\text{next}} &= u \cdot d \;+\; u \cdot s^{\text{present}} \;+\; d \cdot s^{\text{present}} \\ w^{\text{next}} &= d \cdot \overline{s} \;+\; u \cdot d \;+\; u \cdot \overline{s} \\ x^{\text{next}} &= \overline{u} \cdot \overline{d} \cdot x^{\text{present}} \;+\; u \cdot d \cdot x^{\text{present}} \;+\; \overline{u} \cdot d \cdot \overline{s} \;+\; u \cdot \overline{d} \cdot s \\ \text{OVERFLOW} &= \overline{u} \cdot \overline{d} \cdot w \;+\; u \cdot d \cdot \overline{w} \end{aligned}$$

Thus, we have constructed an $N - 1$ bit up–down counter with a guaranteed range of $[-M, M]$ by subtracting two $(N - 1)$-bit counters. Because two independent up-counters are used, the resulting up–down counter is free to accept asynchronous UP and DOWN inputs. This property makes this up–down counter attractive for neural integrators of the sort used in bidirectional servomotor control [DeWeerth and Mead, 1988].

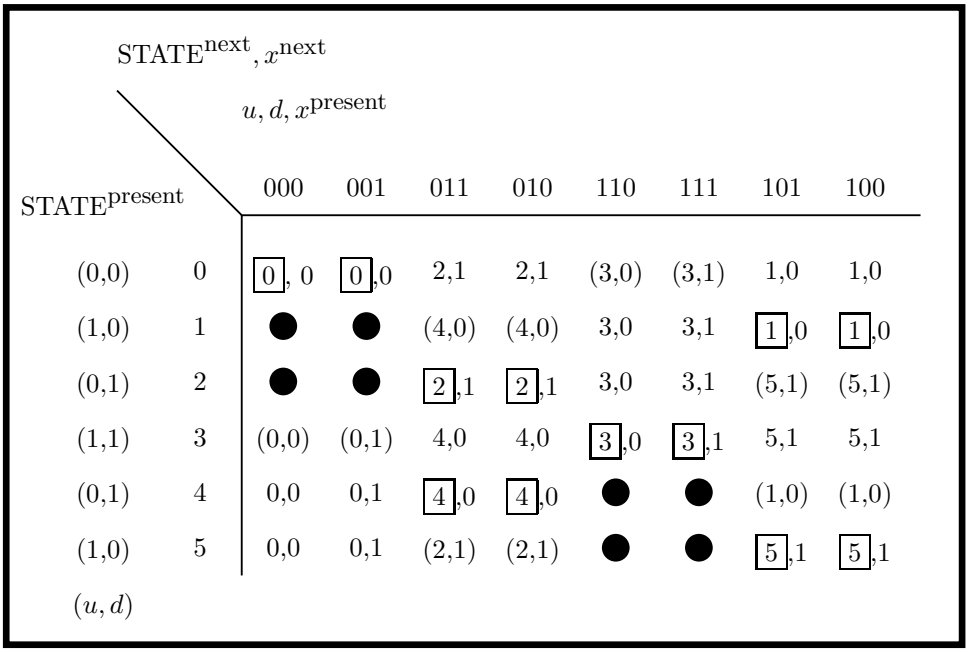


Figure 6.21: Huffman flow table for up–down counter. This table captures all the transitions of Figure 6.20, as a function of the inputs u, d, and x. Transitions in boxes represent fixed points, whereas transitions in brackets constitute violations of the fundamental-mode assumption (i.e., only one input is permitted to change at a time). The dark circles correspond to the detection of an overflow condition (i.e., a state that cannot be represented with only a single bit of additional information (x)).

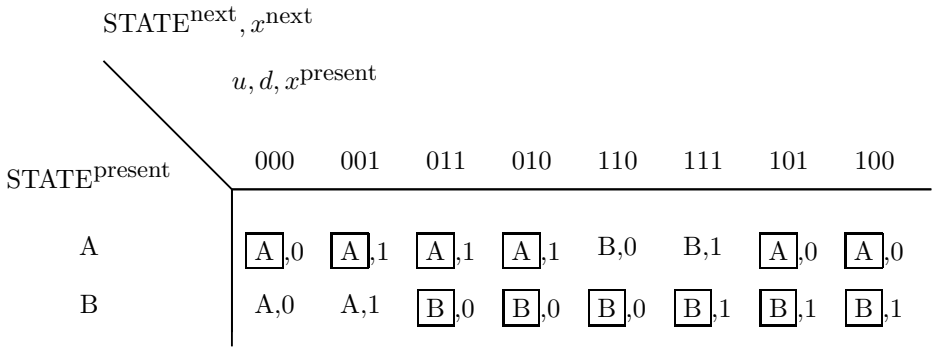


Figure 6.22: Simplified Huffman flow-table for up–down counter. We no longer attempt to detect the overflow condition.

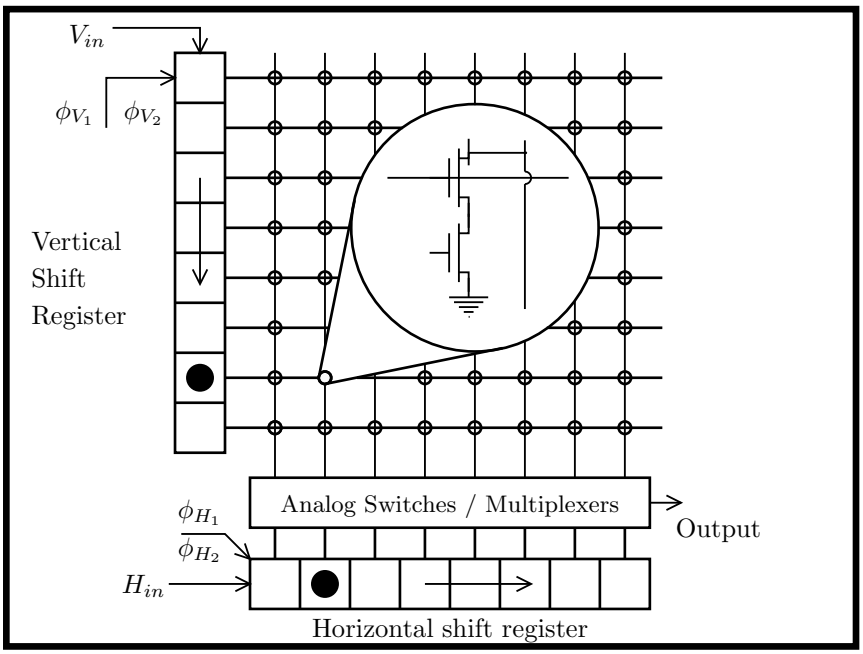


Figure 6.23: First-generation retina design frame.

6.4 A Self-Timed Retina Design Frame

An objective of the *retina* (focal-plane image processing array) research project has been the separation of the design responsibilities for the core processing cells and the external video interface. To facilitate experimentation with a wide variety of processing core cells, we adopted a *design frame* approach, with a standard peripheral frame providing all the communication support functions, as well as the data-sampling mechanism.

The first generation retina chips [**?**] employed the scanning design frame illustrated in Figure 6.23. Later generations of retinas incorporated the one-phase shift register described in Section 6.3, and were self-contained to the extent of requiring only an external crystal in order to generate NTSC video. Unfortunately, direct video requires pixel rates in excess of 10^6 pixels per second for moderate sized imaging arrays (typically less than 100 by 100 pixels). Such high data rates present a challenge for both the on- and off-chip analog electronics.

More importantly, serially scanned images, although very convenient for initial testing of image processing surfaces, are ultimately unsatisfying: They fail to reduce the *dimensionality* of the data. Such extraction of higher-order correlations (ultimately constructs such as edges, velocities, and objects) was one of the principal attractions of multiple level biologically-inspired visual systems. The eventual objective of such a vision system would be to reduce the quantity and rate of data present in a time-varying scene, condensing these data into abstractions acceptable to other systems that must interpret the information.

In this section, we describe an approach to overcome the practical limitation of excessive

pixel-data rates, as well as address the aesthetic concern regarding fully scanned images. The fundamental observation we make is that transmitting reduced-dimensionality data requires transmission of smaller quantities of spatially *nonlocal* information. For example, tracking a moving object probably requires monitoring its imaged perimeter as a function of time; however, even if the whole extent of the object has to be transmitted, less bandwidth is required than to transmit the entire visual scene.

We retain the concept of a design frame for imaging arrays, but replace the sequential scanning protocol with a *data-driven* process: Individual pixel communicate their data after first requesting permission to transmit, and waiting for that permission to be granted.

6.4.1 Designing An Efficient Arbiter

The heart of this data-driven communication channel is the arbitration between multiple requesters. For maximum scalability and flexibility, we have chosen a hierarchical arbitration strategy: Requests from two sources are resolved, and a single request is propagated. Likewise, a single *acknowledge* signal (representing the granting of the shared communication channel) is partitioned to the appropriate requester.

Two measures of efficiency of implementation are appropriate here. One is the actual size and performance of the arbiter cell. The second, and arguably most important, metric is the additional overhead required at the pixel cell to accommodate the asynchronous request–acknowledge protocol. To minimize this overhead, a simplified four-wire protocol was chosen (see Figure 6.24); because only single transitions are significant, the logic to decode the current state is extremely simple. Also, by specifying the removal of all requests after the granting of a single acknowledge, issues of fairness are eliminated without the introduction of additional state information (in practice, an external agent intervenes to temporarily withdraw all the requests—the individual pixels do not need to honor this requirement themselves).

The design frame takes the form of Figure 6.25. A transmitting cell makes a request in the horizontal direction. When an acknowledge signal comes back from the horizontal arbiter, a second request is made, this time in the vertical direction. Only the cell that receives *both* acknowledges is permitted to transfer its data.

In the general case, the addressed cell would now assume control of the shared resource. When it is finished using the resource, the cell communicates this fact by driving the global D line high, at which point an external agent drives the global reset line Z. The Z line controls two operations. First, the request signals are withdrawn at the perimeter of the array, permitting the resetting of the arbiter state. Second, the internal state of the addressed cell is reset, reflecting the fact that its data has been communicated.

In actual practice, an abbreviated form of the handshaking protocol is possible, in which there exists no global resource, but rather the very act of obtaining a pair of acknowledge signals constitutes the communication of information. As a concrete example, consider the

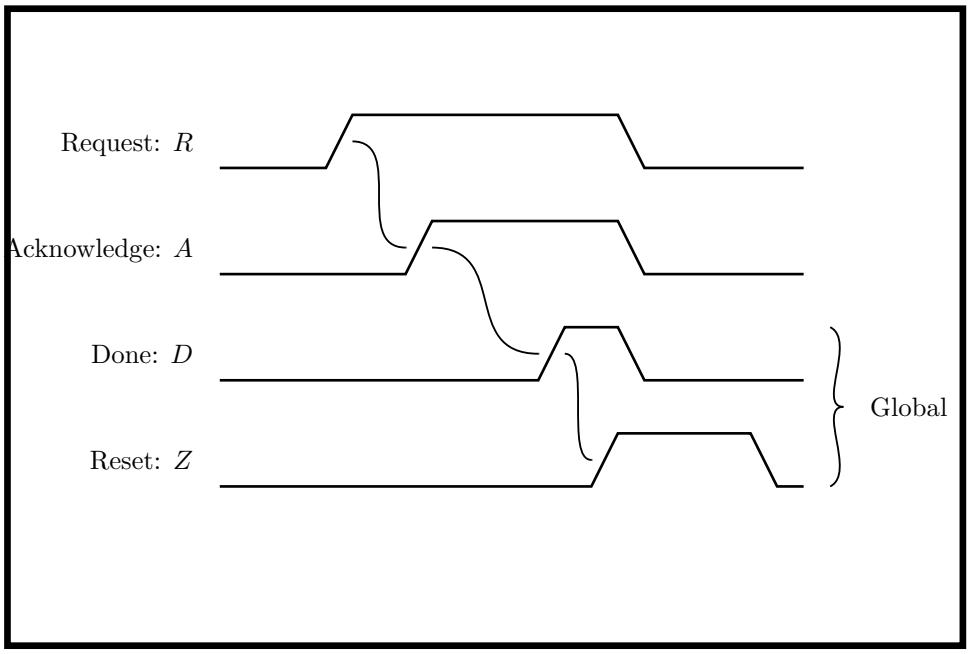


Figure 6.24: The self-timed retina asynchronous communication protocol.

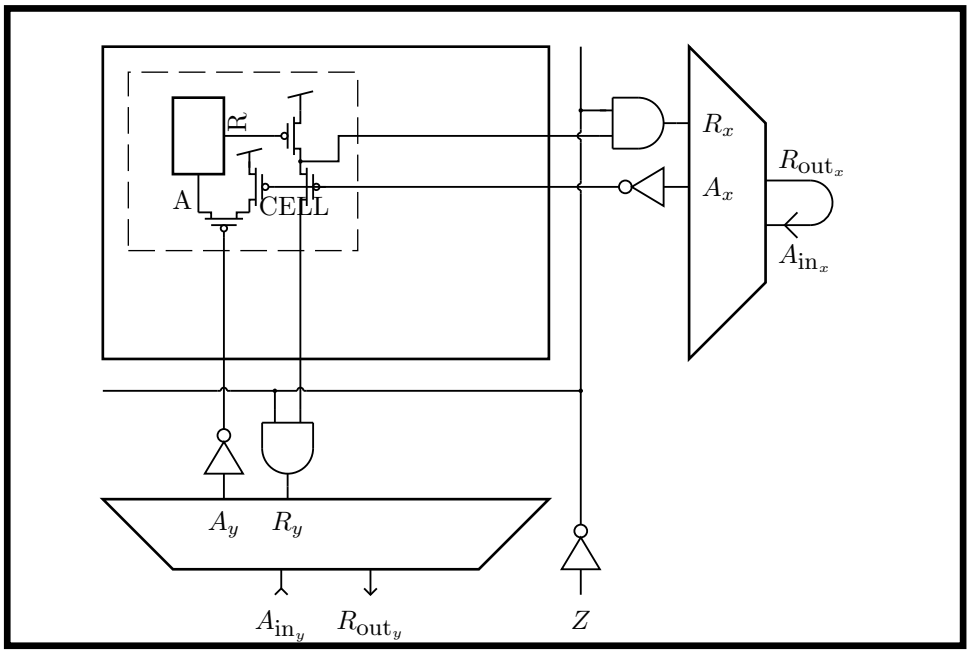


Figure 6.25: Self-timed retina design frame.

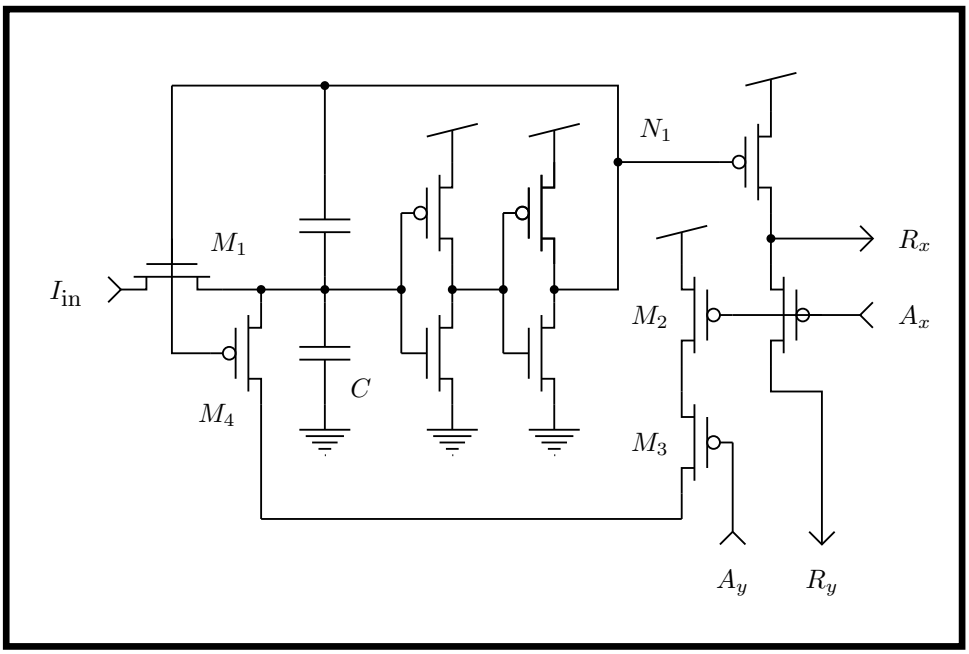


Figure 6.26: Delta-modulation cell.

delta-modulation cell in Figure 6.26: The "neuron" [**?**] cell integrates charge on C_1 until the inverter threshold voltage is reached, and node N_1 goes low. At this point, a horizontal request is made, *and* the integration of input charge is stopped by M_1. When the vertical (second) acknowledge signal arrives, the internal state of the neuron is reset completely (to V_{dd}) by M_2 and M_3. Series transistor M_4 prevents draining any input charge *after* the neuron changes state. Thus, this circuit generates a single request–acknowledge cycle for a particular fixed quantity of input charge.

A Compact Arbiter Cell

In addition to an efficient handshaking protocol, the self-timed retina design frame requires a small, fast arbiter. To maximize ease of scalability, we have chosen to construct a binary tree of 2-input arbiter cells (see Figure 6.27). Such a tree can be constructed of $O(N)$ width (for an $N \times N$ array), if $O(\log N)$ horizontal wiring channels are permitted. Mahowald [Mahowald, 1990] has implemented a WOLCOMP [**?**] module generator to construct arbiters to interface with arbitrary arrays.

The design of the arbiter cell itself is shown in Figure 6.28. In its minimum form, the cell consists of 14 transistors. Transistors M_1 through M_6 comprise a mutual-exclusion element: two cross-coupled inverters that can only drive to ground when at least one input request is present. The output corresponding to the nonrequesting input is kept high by one of M_7 and M_8. The output request signal is generated by the NOR of the two input requests. In fact, this signal is available as the node R_{out} in the mutual-exclusion element.

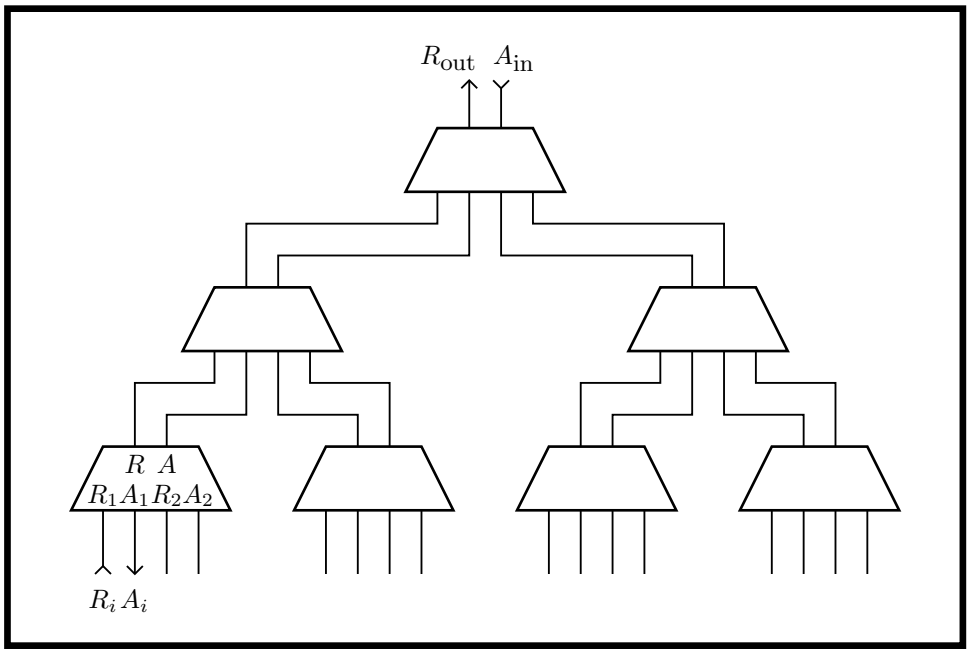
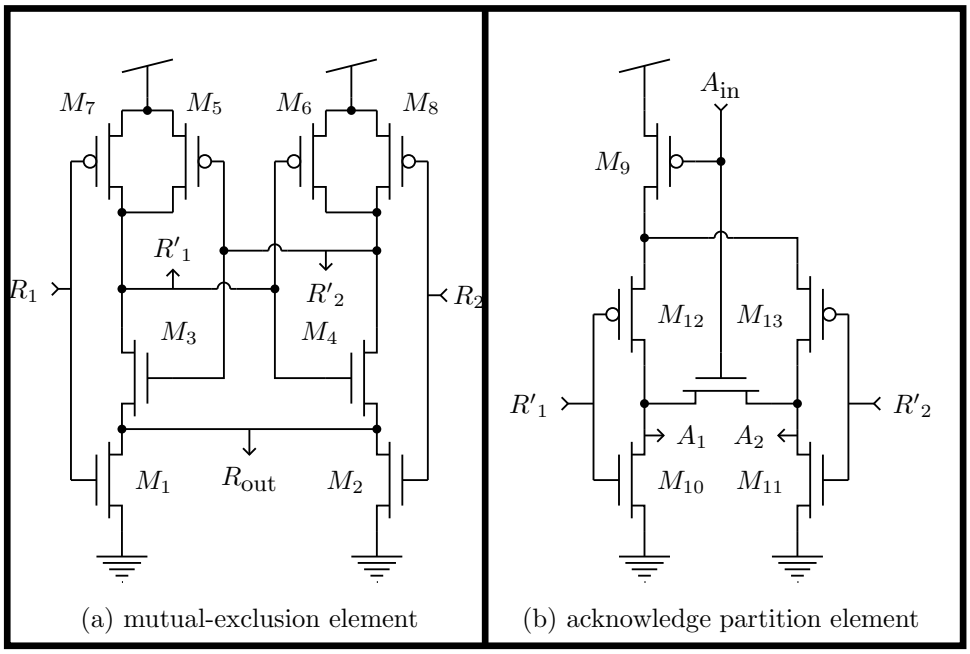


Figure 6.27: Binary tree of 2-input arbiters.



(a) mutual-exclusion element (b) acknowledge partition element

Figure 6.28: The 2-input arbiter cell.

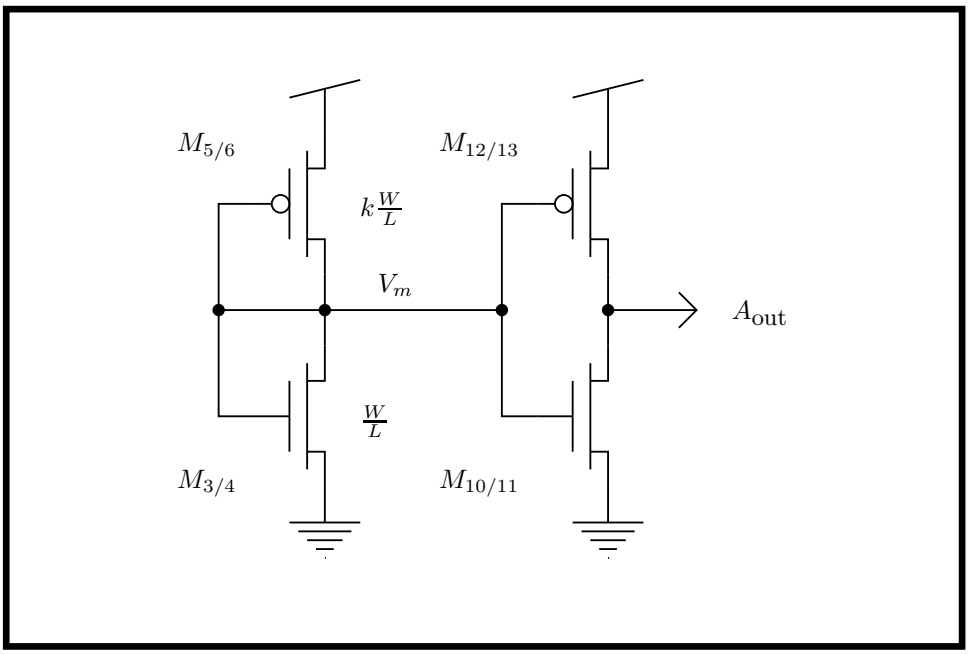


Figure 6.29: Equivalent circuit for arbiter in metastable state.

The second function performed by the arbiter is the partitioning of the incoming acknowledge signal to the acknowledge line corresponding to the requesting input (i.e., the input to which the mutual-exclusion element has awarded the request). This operation is performed by the circuit in Figure 6.28(b). This circuit is fundamentally equivalent to two AND gates generating $A_1 = \overline{R'_1} \cdot A_{\text{in}}$ and $A_2 = \overline{R'_2} \cdot A_{\text{in}}$, but shares components to reduce the transistor count to 6 devices. In addition, this circuit guarantees that *no* acknowledge signal propagates backward until any metastability in the mutual-exclusion element has been resolved. This behavior makes it safe to generate the request signal R_{out} *immediately* upon receipt of any input request, thereby "pipelining" the request and accelerating the operation of the system.

To determine the constraints required to obtain this safe behavior, we consider Figure 6.29, which depicts the equivalent circuit to the mutual-exclusion element and acknowledge-partition circuit in the case of simultaneous input requests. In the absence of noise or inter-device variation, this worst-case scenario produces unbounded metastability.

By symmetry, the internal nodes of the mutual-exclusion element are at identical voltages ($R'_1 = R'_2$); furthermore, this voltage is precisely the inverter threshold voltage (defined to be the voltage where $V_{\text{in}} = V_{\text{out}}$). Typically, VLSI processes adjust transistor thresholds to produce inverter threshold voltages near $V_{\text{dd}}/2$ for n- and p-channel devices of identical geometry. Since both devices are in saturation, this inverter threshold voltage (V_m) is given by:

$$Z_p(V_{\text{dd}} - V_m - V_{T_p})^2 = Z_n(V_m - V_{T_n})^2 \tag{6.2}$$

Suppose that we scale the p-channel device by k in order to shift the inverter threshold

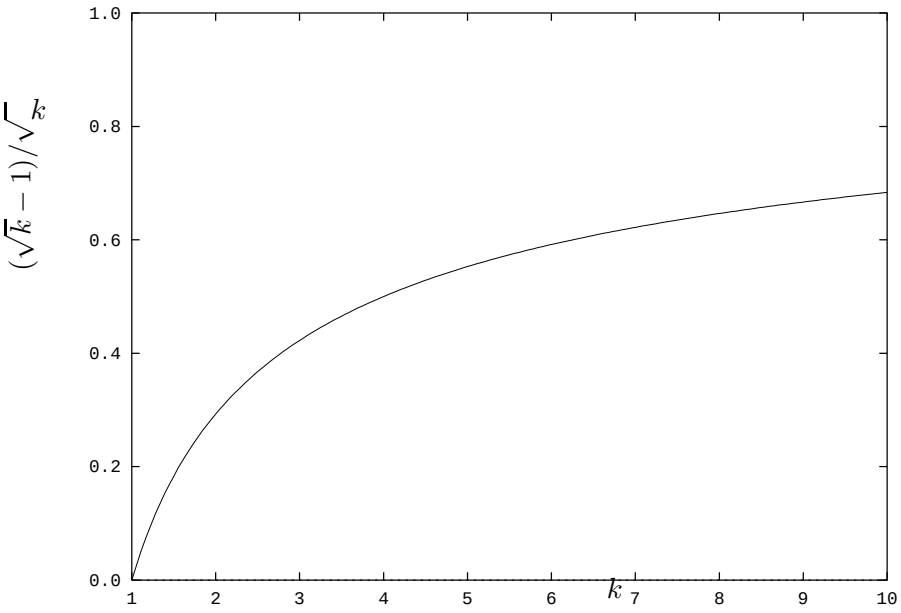


Figure 6.30: Guaranteeing safe operation of arbiter by scaling transistors. The plot describes the function $(\sqrt{k}-1)/\sqrt{k}$; the second term in Equation 6.5 is $\frac{\sqrt{Z_n}}{\sqrt{Z_n}+\sqrt{Z_p}}(V_m - V_{T_n}) \approx \frac{1}{2}\left(\frac{V_{\rm dd}}{2} - 1\text{ V}\right) \approx 0.8$ V. Consequently, we see that only modest geometry factors ($k = 6$) are sufficient to shift the inverter threshold voltage by half a volt, an amount that is more than adequate to keep the output inverter in Figure 6.29 from propagating the acknowledge signal.

voltage of the mutual-exclusion element by δ, thereby disabling the gate that partitions the returning acknowledge signal. Then we have (taking the square root of both sides)

$$\sqrt{kZ_p}(V_{\rm dd} - V_m - \delta - V_{T_p}) = \sqrt{Z_n}(V_m + \delta - V_{T_n}) \tag{6.3}$$

We can bound the effect of k on δ by observing that, for $k > 1$,

$$(V_m + k\delta - V_{T_n}) > (V_m + \delta - V_{T_n})$$

Substituting into Equation 6.3, we obtain

$$\sqrt{kZ_p}(V_{\rm dd} - V_m - \delta - V_{T_p}) < \sqrt{Z_n}(V_m + k\delta - V_{T_n}) \tag{6.4}$$

Solving for k, and substituting the square root of Equation 6.2, we obtain the bound

$$\delta > \frac{\sqrt{k}-1}{\sqrt{k}}\left(\frac{\sqrt{Z_n}}{\sqrt{Z_n}+\sqrt{Z_p}}(V_m - V_{T_n})\right) \tag{6.5}$$

We plot typical values of Equation 6.5 in Figure 6.30.

It is important to note that this analysis is conservative on two accounts. First, the probability that the mutual-exclusion element is still "hung" after the request has propagated all the way to the external environment and has returned is exceedingly slim. Secondly, the additional transistors in the actual circuit (transistors M_1 and M_2 in the mutual-exclusion element and device M_9 in the acknowledge partition circuit) both act in a manner to increase the inverter threshold offset; these transistors are ignored in the analysis of the arbiter safety margin. Consequently, we observe, in practice, that the circuit functions reliably without any explicit scaling (i.e., $k = 1$). This observation is the result of both detailed circuit simulation and measurements made on a working circuit.

6.5 Summary

The hierarchical arbiter is a part of the second-generation retina project currently under way at Caltech. A prototype chip with a 98×74 array of delta-modulation pixels has been fabricated, and is currently being tested. The arbiter occupies a region 500λ wide at the perimeter of the array. Preliminary tests indicate that data rates of 2×10^6 pixels/sec are attainable.

Chapter 7

Conclusions

This thesis has presented a topology-based wiring partition function, obtained by considering the intrinsic dimensionality of the interconnection medium. This result is consistent with an empirical wiring partition function known as Rent's rule. In addition, it is possible to take the general form of Rent's rule, and compute from it the expected wire length probability density function. Combining this result with several physical parameters that characterize the interconnect technology, we have derived another physical limit on the connectivity of regular VLSI systems.

We have seen the importance of taking wiring connectivity into account. This concern is particularly valid for analog VLSI, where neurally-inspired architectures require large component counts (and hence scalable designs), and the communication topology is dictated by the algorithm.

As implementation considerations become more important, many networks, such as the fully-connected Hopfield net and the fully-connected (within layers) feed-forward net (typically trained by back-propagation) become impractical. Rather, locally-connected networks, and hierarchically organized limited-connectivity networks will be required. Since only minimal work has been done in developing training algorithms for such networks, this endeavor must become a high priority if neural networks are to gain practical viability. To achieve this goal, we will probably need to attain a deeper understanding of, and attach a greater importance to, intermediate representations within neural networks. The currently popular model of considering a neural network as a "black box" that is presented with inputs and "taught" the corresponding output is far too simplistic, and does not transfer well to multi-layer networks, or networks intended for complex applications. This inadequacy argues strongly in favor of simultaneous research into established (i.e., biological) neural systems, in order to understand the representation (and, for that matter, connectivity and communication) issues relating to neural computations.

Essentially all of the hardware implementations of analog networks to date have been either of the fully-interconnected type (e.g., the Hopfield-style network of Chapter 2), or of the

local-connectivity type (e.g., retinas, cochleas, etc.). These endeavors have been fruitful, having identified tasks that are amenable to solution by such networks, or by identifying reasonable practical limits on the sizes of such networks.

However, it is now time to explore intermediate-connectivity networks. It is for this task that the hierarchically-interconnected PROTOCHIP architecture was developed.

Engineering has traditionally comprised two disciplines: analysis and synthesis. Unfortunately, analog circuit design is a particularly difficult endeavor, in large measure because it depends on the interaction of complex, non-linear elements, and the behavior of the resulting circuit often depends critically on higher-order effects or even parasitic components within the circuit. Consequently, the most common analysis technique (software circuit simulation) is becoming progressively less adequate to the task. Novel circuit structures (e.g., floating gate structures), wide operating regimes (e.g., subthreshold CMOS), and hybrid circuits (e.g., circuits containing both analog and digital subsystems) place extreme demands on the development environment. These *accuracy* requirements, in conjunction with the *performance* penalty associated with software simulation, are limiting the development of analog design innovation.

A hardware accelerator, such as the PROTOCHIP, avoids these difficulties by directly exercising the actual components, thereby bypassing the issue of modeling their behavior. Also, this approach is intrinsically parallel, and high performance can be obtained. Another advantage of this approach is the direct electrical interface to the circuit under tests, thus permitting the circuit to be exercised in an environment similar to that of its ultimate application, and, in the process, of simplifying the *specification* of the system's inputs. For example, the large-signal instability behavior of the second-order section [**?**] is not revealed by traditional small-signal Fourier analysis, and would require extensive exploration of the input signal space to discover. Many other systems, intended to accept real-world signals, are difficult to simulate, because of the difficulty of capturing and specifying these input signals.

With the removal of the execution performance bottleneck of conventional simulators, we must rate field-programmable networks instead on their *ease of use,* and their *universality.* The universality issue has been addressed by the connectivity analysis, and the selection of a hierarchical interconnect structure with appropriate fanout. The ease-of-use requirement has been satisfied by providing a versatile netlist-manipulation tool that includes several different graph-embedding algorithms. These algorithms, which include a bottom-up algorithm (most efficient from the perspective of efficient use of routing resources), a simulated-annealing graph-partition algorithm (a compromise between time- and resource-efficiency), and a fast, greedy routing algorithms that has proven itself sufficiently efficient to satisfy a majority of circuit embedding applications. Of course, a variety of these techniques can be combined within a single design; the embedding tool permits individual embeddings of different sub-cells within a hierarchical design, as well as providing utility functions for manipulating the hierarchy itself. Finally, the ability to efficiently specify netlists, and to verify graph isomorphism, is a useful technique to validate the final design of a system (typically a full-custom layout).

These efficient graph-embedding algorithms, in conjunction with an external schematic-capture graphical interface, contribute to creating an efficient design environment. It is envisioned that one of the principal short-term contributions of the field-programmable network approach is pedagogical: researchers will be able to quickly develop a familiarity with analog design techniques. This familiarity will be gained by design and experiment with actual circuits; at the same time, an understanding of integrated circuit handling and test procedures is obtained.

A longer-term contribution of the design approach presented in this thesis is the opportunity for researchers to custom-tailor PROTOCHIPs, by combining their own bottom-level leaf circuit cells with an automatically-generated hierarchical interconnect network. This approach permits direct utilization of the graph-embedding tools, and chip-programming interface, thus combining the advantages of programming speed and versatility and the possibility of chip re-use (and consequent savings in time and manufacturing costs (also due, in part, to the possibility of higher fabrication volumes)) with the specificity of interconnecting classes of circuits of particular interest.

A design discipline is the combination of specific design instances of fundamental components, and composition rules for their combination into systems with predictable behaviors. At the present time, analog VLSI is primarily in the exploratory stage of considering viable components, and characterizing their behavior. We have begun to see their application in large-scale systems, and have added large-system considerations (e.g., inter-component variations due to process uncertainties, and the effect and compensation of systematic offsets) to the collection of requirements for effective base components.

The composition rules for analog VLSI are still under development. System development has given us computational structures such as resistive sheets, and winner-take-all networks. With time, other powerful and composable analog structures will be developed.

Ultimately, all electronic circuits are analog in their operation. Digital behavior is a convenient *abstraction*, permitting reliable systems to be designed by analog non-experts. However, both higher performance and more dense circuits can be obtained, if one has a deeper understanding of the underlying functionality. Note that pursuit of these objectives does not necessarily change the level of *risk* associated with the design; we have seen several examples in this thesis (e.g., CSRL single-phase latch, two-input arbiter) of circuits that are safe, and rely on analog operation to provide low transistor-count implementations of sophisticated functions.

The prototyping system is now ready for the next phase: the transfer of the chip compiler and circuit embedder software to a user community that is interested in designing semi-custom applications. For example, a researcher concerned with early vision could produce a chip containing photoreceptors and dedicated local interconnect, yet defer issues related to inter-layer communication and post–image-plane processing structures until trial configurations are made in the lab.

One potentially fruitful area of future research would be the development of more sophisti-

cated graph-embedding procedures. Also, extraction of structure from circuit graphs could be used to simplify circuits, and provide technology-independent circuit representations.

An additional area of future research is the application of advanced interconnect technologies, permitting higher performance and densities. Nonvolatile charge storage, in conjunction with ultraviolet-light programming, could yield a commercially viable reconfigurable architecture. Commercially available "antifuse" electrically programmable switch technology could also be applied, but would require a special fabrication process. Finally, new high-speed laser-mediated interconnect processes would permit once-configurable systems of over 100,000 devices, while retaining the advantage of fast-turnaround fabrication time.

References

[Allen and Holberg, 1987] Allen, P. E. and Holberg, D. R. (1987). *CMOS Analog Circuit Design*. Holt, Rinehart and Winston.

[Allen and Sanches-Sinencio, 1984] Allen, P. E. and Sanches-Sinencio, E. (1984). *Switched Capacitor Circuits*. Van Nostrand Reinhold.

[Beatty et al., 1989] Beatty, D., Brace, K., Bryant, R. E., Cho, K., and Huang, L. (1989). *User's Guide to COSMOS: A Compiled Simulator for MOS Circuits*. Carnegie-Mellon University.

[Bhatt and Leighton, 1984] Bhatt, S. N. and Leighton, F. T. (1984). A framework for solving VLSI graph layout problems. *Journal of Computer and System Sciences*, 28:300–343.

[Bryant et al., 1982] Bryant, R., Shuster, M., and Whiting, D. (1982). Mossim II: A switch-level simulator for MOS LSI, user's manual. Technical Report TR #5033, California Institute of Technology.

[Delbrück, 1989] Delbrück, T. (1989). A chip that focuses an image on itself. In Mead, C. and Ismail, M., editors, *Analog VLSI Implementation of Neural Systems*, pages 171–188. Kluwer Academic Publishers.

[Delbrück, 1990] Delbrück, T. (1990). private communication.

[DeWeerth and Mead, 1988] DeWeerth, S. P. and Mead, C. A. (1988). A two-dimensional visual tracking array. In *1988 MIT Conference on Very Large Scale Integration*, pages 259–275, Cambridge, MA. MIT Press.

[Dunlop and Kernighan, 1985] Dunlop, A. E. and Kernighan, B. W. (1985). A procedure for placement of standard-cell VLSI circuits. *IEEE Transactions on Computer-Aided Design*, CAD-4(1):92–98.

[EIA88, 1988] EIA88 (1988). *Electronic Design Interchange Format Version 2 0 0, Recomended Standard EIA–548*. Electronic Industries Association.

[Feinstein and Hopfield, 1985] Feinstein, D. and Hopfield, J. J. (1985). private communication.

[Garverick and Frankel, 1986] Garverick, S. L. and Frankel, B. (1986). *LSH User Manual and Documentation*. M.I.T. Lincoln Laboratory.

[Gradshteyn and Ryzhik, 1980] Gradshteyn, I. S. and Ryzhik, I. M. (1980). *Table of Integrals, Series, and Products*. Academic Press.

[Hayes, 1978] Hayes, J. P. (1978). *Computer Architecture and Organization*. McGraw-Hill. Sec. 5.3.3.

[Hopfield, 1982] Hopfield, J. J. (1982). Collective processing and neural states. *Modelling and Analysis in Biomedicine*.

[Hsieh et al., 1981] Hsieh, K. C., Gray, P. R., Senderowicz, D., and Messerschmitt, D. G. (1981). A low-noise chopper-stabilized switched-capacitor filtering technique. *IEEE Journal of Solid-State Circuits*, SC-16(6):708–715.

[Kernighan and Lin, 1970] Kernighan, B. W. and Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, pages 291–307.

[Linsker, 1986a] Linsker, R. (1986a). From basic network principles to neural architecture: Emergence of orientation-selective cells. *Proc. Natl. Acad. Sci. USA*, 83:8390–8394.

[Linsker, 1986b] Linsker, R. (1986b). From basic network principles to neural architecture: Emergence of spatial-opponent cells. *Proc. Natl. Acad. Sci. USA*, 83:7508–7512.

[Maher, 1989] Maher, M. A. (1989). *A Charge-Controlled Model for MOS Transistors*. PhD thesis, California Institute of Technology.

[Mahowald, 1990] Mahowald, M. A. (1990). private communication.

[McCharles and Hodges, 1978] McCharles, R. H. and Hodges, D. A. (1978). Charge circuits for analog LSI. *IEEE Transactions on Circuits and Systems*, CAS-25:490–497.

[McGregor et al., 1987] McGregor, M. S., Denyer, P. B., and Murray, A. F. (1987). A single-phase clocking scheme for CMOS VLSI. In *1987 Stanford Conference on Very Large Scale Integration*, pages 257–271, Cambridge, MA. MIT Press.

[Mead, 1990] Mead, C. A. (1990). Transactions of IEEE, Special Issue on Neural Networks (in press).

[Ousethout and et al., 1985] Ousethout, J. K. and et al. (1985). The Magic VLSI layout system. *IEEE Design and Test of Computers*, 2(1):19–30.

[Parhami, 1973] Parhami, B. (1973). Associative memories and processors: An overview and selected bibliography. *Proc. IEEE*, 61:722–723.

[Rowson, 1980] Rowson, J. (1980). *Understanding Hierarchical Design*. PhD thesis, California Institute of Technology.

[Thompson, 1980] Thompson, C. D. (1980). *A Complexity Theory for VLSI*. PhD thesis, Carnegie-Mellon University. Technical Report CMU-CS-80-140.

[Tomlinson et al., 1990] Tomlinson, M. S., Walker, D. J., and Sivilotti, M. A. (1990). A digital neural network architecture for VLSI. In *Proceedings of IJCNN-90*.

[Ullman, 1984] Ullman, J. D. (1984). *Computational Aspects of VLSI*. Computer Science Press.

[Umminger and DeWeerth, 1988] Umminger, C. B. and DeWeerth, S. P. (1988). Implementing gradient following in analog VLSI. In *Advanced Research in VLSI: Proceedings of the Decennial Caltech Conference on VLSI, March 1989*, pages 195–208, Cambridge, MA. MIT Press.

[Vittoz, 1989] Vittoz, E. (1989). CMOS VLSI design: Analog & digital. Intensive Summer Course, Course notes, EPFL, Lausanne, Switzerland.

[Wawrzynek, 1987] Wawrzynek, J. R. (1987). *A VLSI Architecture for Sound Synthesis*. PhD thesis, California Institute of Technology.

[Weems, 1982] Weems, C. (1982). TITANIC: A VLSI based content addressable parallel array processor. *Proc. IEEE ICCC*, pages 236–239.

[Wegmann and Vittoz, 1989] Wegmann, G. and Vittoz, E. (1989). Very accurate dynamic current mirror. *IEEE Electron Letters*, 25:644–646.