A VLSI Implementation of a Two by Two Packet Router

Computation Structures Group Memo 197
July 1980

Paul S. Ries

S.B. thesis, Department of Electrical Engineering and Computer Science, MIT.

Abstract

A VLSI Implementation of a Two by Two Packet Router

Paul S. Ries

S. B. Thesis for Jack B. Dennis

This thesis concerns the implementation of a two by two packet router, which could be used to build large packet routing networks. The implementation that is considered here uses Metal-Oxide Semiconductor Field Effect Transistor logic, with Very Large Scale Integration. The goal is to implement a packet routing network for a large data-flow computer.

The packet router uses an eight-bit data byte-wide protocol. It has two inputs and two outputs. The output direction of a packet depends on an address bit that is carried within the packet. The protocol allows for arbitrary length packets. The two by two router would be used in an array of N/2 by $\log_2 N$ to implement an N-input, N-output routing network.

The implementation considered here makes use of layout cells that implement data-flow modules. A higher-level design using data-flow modules is used to assemble these various modules to form a packet router. The cells are sufficiently general-purpose that they could be used to implement other control logic systems. Furthermore, they are standardized in a manner that would allow automated placement and interconnection as part of an automated design system.

The logic used to implement the modules is fully-asynchronous, self-timed, dual-rail logic. In part, this thesis is a consideration of the feasibility of this particular approach to logic design. To that end, a test chip that implements a First-In First-Out Buffer using the modules was designed and is being fabricated.

Table of Contents

Table of Figures

Table of Tables

I. Introduction

One of the main problems of integrated circuit systems design has been finding something to design. It is often difficult to find applications suited to a VLSI design approach. To be economically attractive, a VLSI integrated circuit system must be either very general in its function (eg.: microprocessors) or simple enough that it can be widely used as a subcomponent (eg.: memory, programmable logic arrays). As VLSI systems have become less expensive to design and manufacture, custom design has become more feasible. Economics, however, is still a factor in custom design. The best approach, even when using custom design, is to design a system that is useful in fairly large quantities. This thesis examines a VLSI device that could be used in large arrays within a high performance computer. As many of these systems would be required in a single computer, and will, hopefully, result in a very high performance computer, this device is potentially useful and economically attractive.

The purpose of this thesis is to examine the implementation of a VLSI component to be used in a network for data packet routing between unsynchronized computer elements. Computer architectures consisting of elements communicating via an asynchronous routing network have been investigated by Dennis, Misunas and Leung [1].

The problem of designing a network capable of supporting the high data transfer rates inherent in such a computer is a fundamental one. An investigation of the general approaches to designing such networks was made by Boughton [2]. A more specific approach was taken by Leung [3]. Leung's approach will be investigated more thoroughly in this thesis. The particular area of investigation is that of implementing a component of a routing network called a 2 by 2 packet router.

The approach taken by this thesis is to design and lay out VLSI circuits corresponding to several basic modules specified by Leung. The VLSI circuits, furthermore, are designed using self-timed asynchronous logic circuits as described by Seitz [4]. This thesis will develop layouts, using these basic logic circuits, designed in a manner that would allow them to be assembled into a router system. Most of the modules are designed to allow flexibility in the byte width of the data passing through them, so that their use can be generalized. It should be noted that, as the design approach taken by Leung is a general one, the modules could easily be used in other systems than the router.

The research undertaken in this thesis is useful in three respects. The primary purpose is to investigate the feasibility of implementing Leung's packet router on a single VLSI chip. The second area of usefulness involves the design of generalized VLSI modules that are the components of the packet router. Finally, this thesis will help determine the usefulness of the self-timed logic elements used to implement the packet router.

As the packet router is the primary focus of this thesis, it is necessary to define its functionality before its implementation can be considered. The two by two router specified by Leung could be the basic element in a packet routing network with a large number of inputs and outputs. A two by two router has two inputs and two outputs. The inputs receive bytes of information in sequential strings of arbitrary length called packets. Its function is to take packets at either of its inputs and transfer them either to the upper or lower output depending on an address bit within the packet (see Figure 1.1). It is used in an array of $\log_2 N$ stages, each consisting of N/2 two by two packet routers, where N is the number of inputs and outputs to the array. Each stage would correspond with a particular bit of the address which specifies each output of the network. The correct interconnection of two by two routers will yield a network that allows a packet from any input to travel to any of the inputs to any of the outputs, as demonstrated by the eight-input, eight-output example in Figure 1.2. For an network having $N=2^n$ inputs and as many outputs, where n is an integer, the network can be defined in terms of two by two routers, and two 2^{n-1} by 2^{n-1} networks, as shown in Figure 1.3. The 2^{n-1} by 2^{n-1} networks can be further decomposed in a similar manner, allowing a complete implementation in terms of two by two routers. The outputs will each be specified by an n-bit address to be contained in each packet.

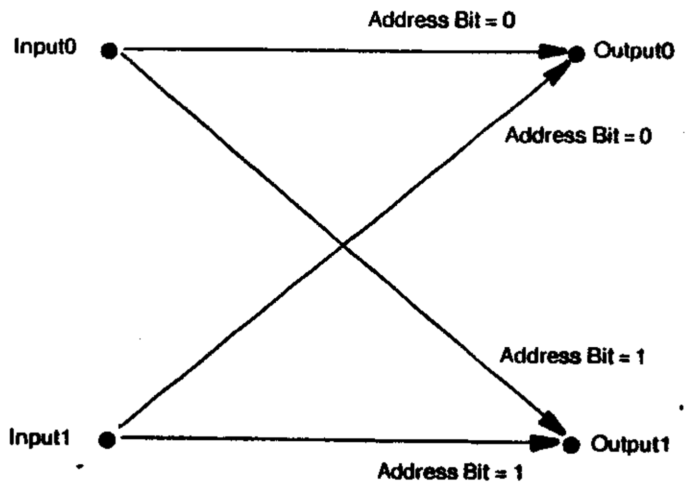


Figure 1.1 Function of a 2 by 2 Packet Router

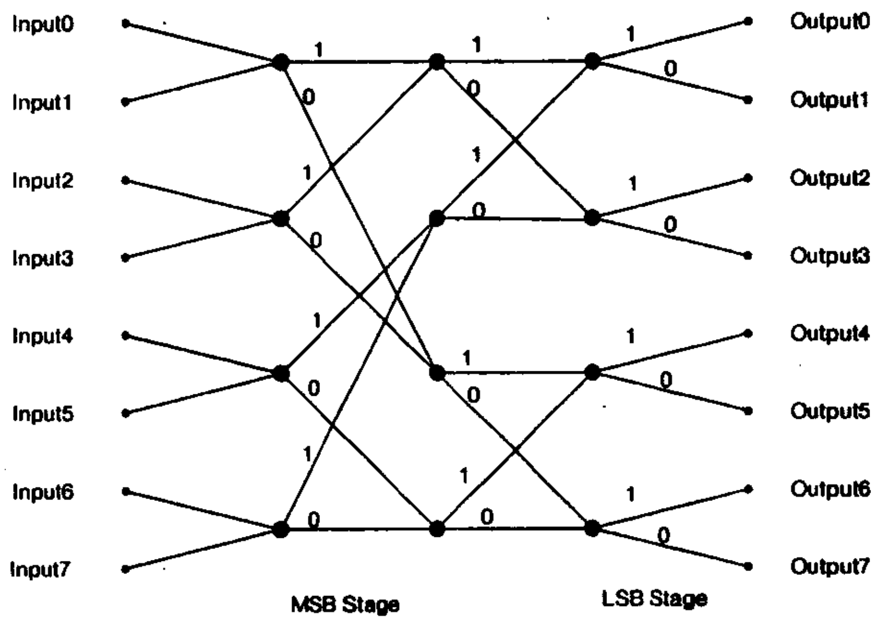


Figure 1.2 An 8 by 8 routing Network
(each node represents a 2 by 2 Router)

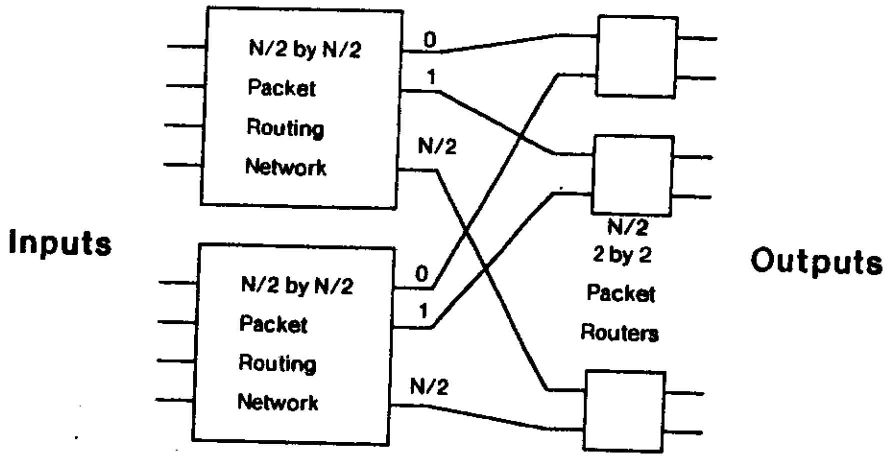


Figure 1.3 Recursive Definition of a N by N Routing Network

The router could be implemented in a number of ways. The necessity of high throughput must be primary in the consideration of alternative methods of implementation. An asynchronous self-timed circuit technology was chosen for this implementation in the hope that it would result in a high throughput.

There are a number of reasons for using asynchronous circuits for high performance. The simplest of these is that in synchronous systems, the clocking rate is limited by the worst case delay of the logic circuits between the clocked elements in the system. An asynchronous self-timed implementation, on the other hand, will execute in the minimum delay time of the logic elements. This advantage is bought at the cost of higher logic complexity and possibly a longer circuit delay time for some circuits. Pure circuit complexity is not as strongly related to cost when considering a VLSI implementation, as when considering an implementation using MSI and SSI parts, because interconnections take up a significantly larger proportion of silicon area in VLSI than in MSI and SSI. In VLSI, a well-structured system might take up less area than a less complex, but less structured system. It is the shift of emphasis in VLSI design that prompts us to reconsider implementations that were rejected years ago because of their complexity in terms of logic gates. It is hoped that this approach will give higher performance than a synchronous approach, while not requiring too much more real-estate on an integrated circuit.

In a routing network like the one described above, conflicts can occur between two packets from different inputs whose addresses cause them to contend for the same set of output wires. These conflicts can be resolved in two different ways. They can be synchronized to a common clock, and the conflict can be resolved by a simple sequential logic circuit. The self-timed approach uses an asynchronous arbiter circuit that will resolve the conflict without synchronization. Studies have shown that asynchronous arbitration and synchronization have similar delay properties. More specifically, it has been shown that neither arbitration nor synchronization can be accomplished with absolute success within a bounded time [5]. This research suggests that a synchronous approach must allot a relatively long time to synchronization to assure that it will have a high probability of success. The self-timed approach can accomplish arbitration in a shorter average time, as it will wait only as long the arbitration takes for each individual usage. There are two synchronous approaches that we can contrast the self-timed approach with. One approach consists of breaking the network into smaller synchronous systems, as we have proposed, and performing synchronization at the inputs to each of these smaller systems. The performance of this approach would be severely limited by the time required for synchronization. The importance of the information passing through the proposed routing network would require extremely low error occurrence. The synchronizer, therefore, would have to be allotted a period substantially longer than the average synchronization time. The self-timed approach would only require the exact amount of time required for each individual arbitration, because a self-timed system will respond to the arbiter's acknowledge signal as soon as it is asserted. The other synchronous approach consists of synchronizing the inputs to the network once, and treating the network as one large synchronous domain. One of the problem with this approach will be immediately evident to anyone who has built a synchronous system that even begins to approach the scale of a 256-input, 256-output routing network. The clock skew inherent in any practical approach to constructing a system of that size will markedly reduce the speed at which such a network will operate. A complex timing waveform (like the two-phase clock used in most synchronous MOS integrated circuits) might improve the performance by allotting more time for signal that have to travel farther, but this approach would have its own timing problems, simply because of the added complexity. Furthermore, this approach does not really avoid the synchronization problem. The synchronizer will be a bottle-neck to the through-put of the network whether it is used once, or many times throughout the network.

The second advantage of the self-timed approach is more closely linked to the VLSI implementation. It has been stated that for larger VLSI systems, clock skew within a chip, caused by the distributed resistive and capacitive nature of the wiring, becomes an important problem [4]. A two by two router is not an excessively large system, but the investigation of circuits that will be less sensitive to skew of critical timing signals is a worthwhile area of research in itself. Self-timed logic avoids timing skew, because timing signals are usually exchanged between circuits that are in close proximity to each other on the chip. Furthermore, even if timing signals are sent over a longer distance, there is less difficulty, because the data signal will accompany the timing signals from one stage to the next. The synchronous approach sends a single timing signal all over the chip, which requires the clock system to drive very large capacitive loads. The clock signal does not accompany the data signals, so that a synchronous system is much more likely to have clock skewing problems. Performance of self-timed logic, therefore, is not as likely to be limited by wire delays, as synchronous approaches may be.

The issues of comparing self-timed logic with synchronous logic are not the topic of this thesis. Several arguments are presented here in favor of the self-timed approach to give the reader a feeling for why one would wish to experiment with self-timed logic when the synchronous approach is predominant in current semiconductor logic. This thesis merely seeks to lay the groundwork for later evaluation of the design approaches taken here.

The first part of this thesis paper will describe the functional design developed by Leung, as used in this implementation of a packet router. The basic modules described by Leung are the starting point for this implementation.

As this thesis deals with implementation, the question of how the modules will be implemented is an important one. A portion of this thesis will examine a generalized VLSI technology that was available for this implementation. The details of this presentation follow Mead and Conway [6]. The technology is a particular form of N-channel MOS transistor logic whose design rules have been simplified by Mead and Conway to a level that makes this thesis a much simpler undertaking.

After presenting both a functional description of the router and a detailed description of the VLSI technology to be used, the actual module designs are be developed. The basic logic representations of the modules have been modified to fit the MOS VLSI technology to yield circuits for this implementation. These circuits have been laid out to form the modular layouts in a form that allows a fair amount of flexibility in combining them together.

Finally, some consideration will be given to combining these modules to form useful systems. A guideline for the layout of a full router module will be presented. Also, the design of a chip for testing the feasibility of using self-timed logic will be presented.

II. Functional Description of The Packet Router

As the main purpose of this thesis is to consider the implementation of a two by two packet router, it is essential to specify the functional characteristics of such a router before presenting the actual design approach. A functional description for a two by two packet router was developed by Leung [3] using data flow concepts previously developed by Dennis [7]. The functional description is presented by Leung in a top-down fashion. The network and protocols that specify the packet router network were specified initially. Next, data flow concepts were used to implement a router on a modular level, consistent with the specified network and protocols. Finally, a behavioral description of the various modules was presented. This chapter will present Leung's functional description of the router, as well as a behavioral description of the modules, so that the implementations of these various modules and of the router can be developed later on.

The highest level of a packet routing network is the network itself. A packet routing network can be defined as a system with a number of inputs and outputs, that allows packets of information to flow from any input to any output. As a subcomponent of such a network, the two by two router can be described as a two input, two output system, whose inputs and outputs are packets of information (as shown in Figure 1.1). Its basic function is to take packets at either input and send them to a particular output. In this manner a network allowing information flow from any of N inputs to any of N outputs can be constructed (see Figure 1.2 and Figure 1.3). The direction of output is controlled by an address bit in each packet, and each stage of the network examines a different bit. In the network of Figure 1.2, it can be seen that the bits examined at each stage of the network, when taken together characterize an distinct n-bit address for each of the N inputs, where $N = 2^n$.

Breaking the network into smaller, simpler, identical parts makes possible a VLSI approach to designing a packet routing network. To be economically feasible, a VLSI system should be one whose function is simple enough that it is required in large numbers. Clearly, this is the case for the two by two packet router, as $\log_2 n$ stages, each consisting of N/2 of these are required for a network of N inputs and N outputs. VLSI has limits to the complexity that can be attained on a single chip. These limits are due to both area limitations and pin limitations. It is an accepted fact that chip yield is related to e^{-NA}, where A is the area of the chip, and N is the number of flaws per unit area, which is a process-dependent quantity. Clearly, a more complex system will require more area, and therefore will be less feasible. As processing technology improves, this restriction will relax. Another important limit to complexity, the pin count, is due partly to packaging constraints, and partly to the fact that each output pad requires a certain amount of area and interface circuitry. The packaging constraints will also relax in time, but it will always be desirable to limit the number of outputs of a VLSI system. Given current technologies, the implementation of a large packet routing network with wide packet bytes, is not feasible as a single package. The critical limitation for VLSI technology in its present form,with respect to packet routing networks will always be pin count. The partitioning of a routing network into smaller elements is, therefore, also necessary for technological reasons.

The packets themselves must be structured to allow the address bit in each stage to be examined easily by a system that hopefully will not require specialized units at each stage. This is accomplished by splitting the packets into bytes and specifying the first byte of each packet as the address byte. The packets splitting will yield another a performance advantage, as the bits in each byte can be processed in parallel. In Leung's specification, the packets consist of eight-bit bytes, which allow 2^8 or 256 possible addresses. The eight bit byte also gives the system byte-wide compatibility with many forms of computing machinery. Furthermore, eight bit-wide packets make a VLSI two by two router implementation possible in a 48 pin package, which is a conventional package size for VLSI components.

The addressing approach considered above allows the same router chip to be used at any stage of a routing network as illustrated in Figure 2.1. The only bit of information of concern to the router is the address bit. Therefore the packet bytes can be bit-rotated from one stage to the next, so that the circuit that determines the direction to send the packet can be attached to a single bit in the router and still access a different bit at each stage.

Another mechanism is required to form a complete packet protocol. Some method of partitioning between one packet and the next is required. Since it is desirable to make the network transparent to the systems using it, the best approach, as determined by Leung, is to add an additional bit of information for this purpose. This bit will have a logic value of 1 if the byte it is associated with is the last byte of a packet. With this addition, a complete packet switching protocol has been established. The reader is reminded that this bit is added to the eight data bits, and is not be bit-rotated with the rest of the byte between stages.

The top-level functionality has been specified, but it is desirable to add one more feature to this specification. It is important to note that a packet networl that deals with unsynchronized systems may have to deal with more than one packet at a time. It would be an significant performance loss if we did not allow the router to process packets from separate inputs destined for separate outputs in parallel. It is necessary, then, to add a feature into the specification of a packet router to deal with multiple packets at a router inputs. The system must not allow conflicting packets to interfere with each other's content.

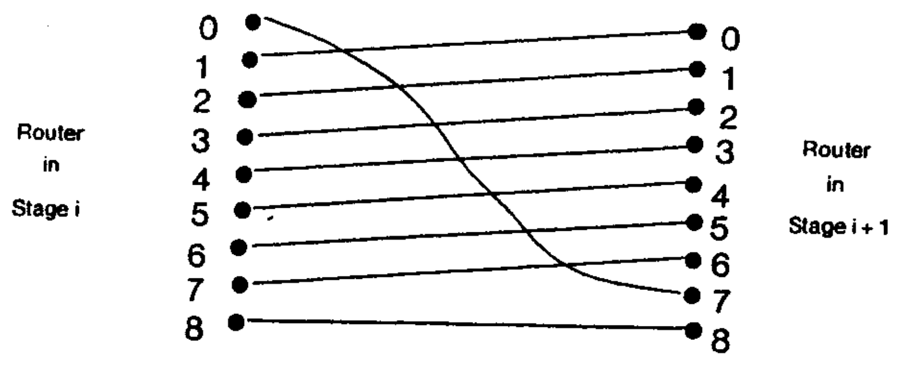


Figure 2.1 Bit Rotation in Router Interconnection

Conflict must be defined on the router-level so that our specification can include features to prevent this from occurring. Conflict occurs whenever two packets entering a router have the same address bit, so that they both are destined to use the same output. When such a conflict occurs, arbitration between the two packets must resolve the conflict in favor of one packet or the other. The packet that is not chosen can wait for the router to finish processing the chosen packet. This type of arbitration is easily implemented using Dennis' data flow concepts.

Some functional partitioning of the router is necessary to implement an arbitration scheme. The approach taken by Leung (see Figure 2.2) is to partition the router into four modules of two different types. One type of module will process inputs. It will wait for packets to arrive and will examine their address bits to determine their output direction. The other type of module will process outputs. It will perform the arbitration between conflicting packets. The Control Module (CM) will have one input from the previous router stage, or from the input to the network, and two outputs, one to each of the two Output Modules (OMs). The OM, in turn, will have two inputs, one from each of the CMs, and one output to the next router stage, or to the output of the network.

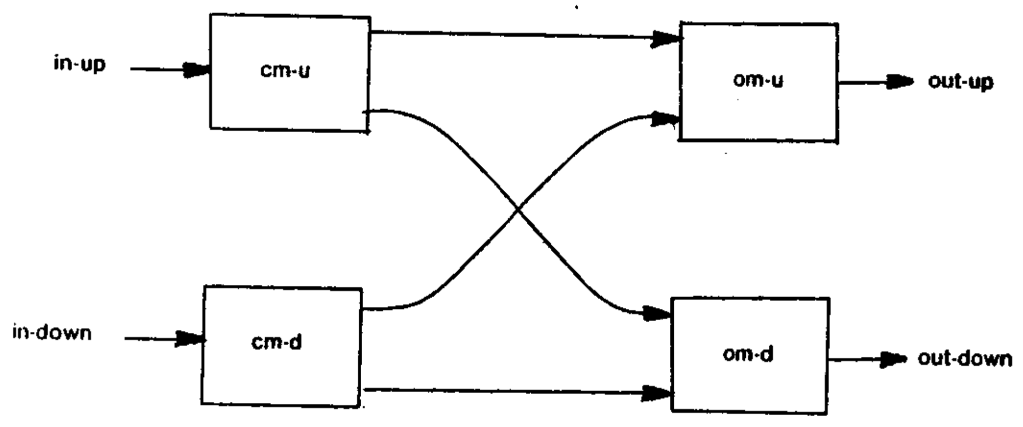


Figure 2.2 Functional Partitioning of Packet Router

This particular partitioning allows for parallel processing by the router if the incoming packets do not conflict, as demonstrated by Figure 2.3. Parallel processing on the router-level allows for a much higher network throughput rate that would be possible otherwise.

So far, several of the choices made in implementing a top-down design have been consistent with the high performance goal that originally prompted our choice of asynchronous self-timed logic. Both the byte-wide parallel processing and the router-level parallel processing of packets contribute to the performance of the router. In any design effort, such consistency on different levels of design is necessary to achieve a desired characteristic such as high performance.

The basic principles behind Leung's specification of the two by two router modules have been presented. The transition from our current level of design to a modular level of design is significant only in its results. A modular description of the CM and the OM will be presented in terms of data flow modules as described by Leung [3]. The modular level of design will be presented without further explanation of the functions of the modules until the design of the modules themselves is presented.

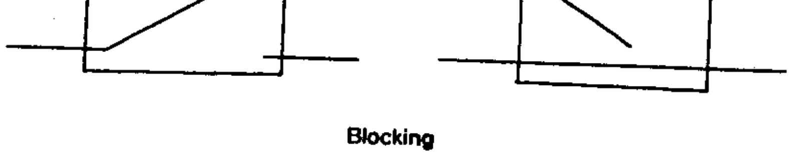


Figure 2.3 Parallel Processing in a 2 by 2 Router

A bit-level data-flow representation of the CM is presented in Figure 2.4. The variables In0-In8 comprise the byte input to the packet router. It should be noted that the bit In0 represents the address bit for the packet. It is loaded into the CM whenever a new packet is begun, when it becomes the new value of UpP, the current packet address bit. UpP is stored in a feedback loop until the next packet arrives. The bit In8 informs the CM whether or not the current byte is the last byte in a packet. This bit is stored until the next byte arrives, at which point it becomes the variable Pkt_Byte_Id which represents whether or not the current byte is the first byte of a new packet. The assumption is a basic one, that the byte following the last byte of a packet will be the first byte of a packet. This variable is used to control whether the input In0 will replace the old value of UpP. UpP is used to route the bytes on input line In0-In8 to either of the outputs comprising either Out_U0-Out_U8 or Out_D0-Out_D8,which are the inputs to either the upper or the lower OM respectively. Finally, Pkt_Byte_Id determines whether or not UpP is used to send a request signal to either the upper of lower OM whenever a new packet arrives. The output signals Req_U and Req_D are requests to the upper and lower OM respectively. The data-flow description of the CM in Figure 2.4 fully specifies the behavior of the CM, and hardware implementations of the data-flow modules will be used to implement the CM.

The OM is specified with data-flow modules in Figure 2.5. The inputs Reg_U and Req_D come from the upper and lower CM. A MERGE module is used to arbitrate between them, as the OM can only take input bytes from one CM at a time. The MERGE generates a single variable Idir which specifies which of the two inputs will be used. Idir will be stored in a feedback loop which is reinitialized from the first byte of each packet. Idir is used to choose one of the two byte inputs consisting of bits In_U0-In_U8 and bits In_D0-In_D8 from either the upper or lower CM. D0-D8 are the bits of resulting internal packet byte. D8 is the bit used to denote the last byte of a packet, and is used to allow initialization of the feedback loop holding Idir when the next byte arrives. The variable D0-D8 become the output variables Out0-Out8, which are the outputs of the entire packet router. The modules comprising the OM will also be implemented in hardware. The implementations of the CM and OM will

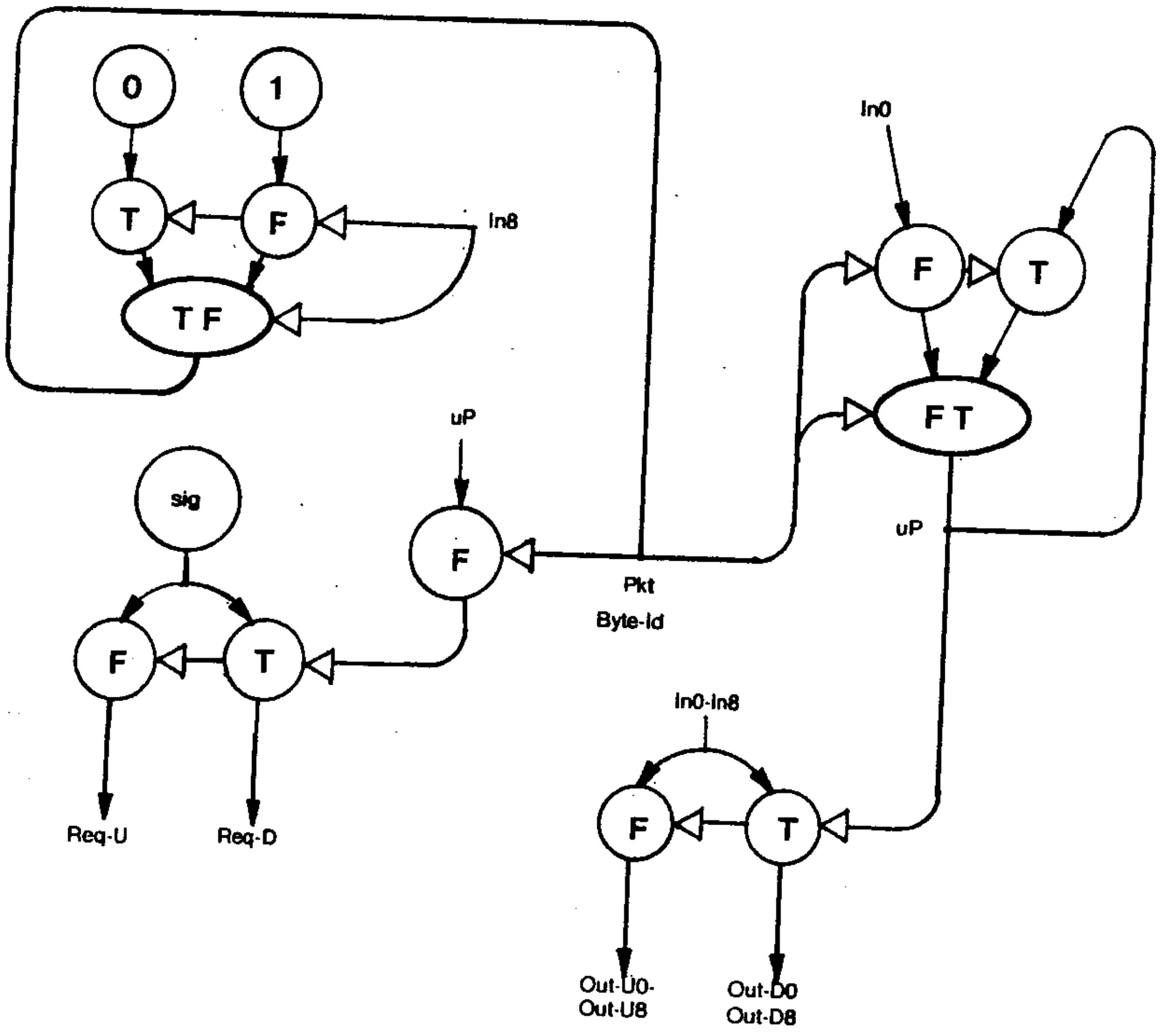


Figure 2.4 Bit-Level Data Flow Representation of the CM

be integrated into a single structure as was shown in Figure 2.2. Thus, a complete behavioral description of the packet router has been presented using data-flow modules.

In order to implement the packet router, the modules making up the router must be expressed in terms of simpler elements that represent logic circuits to be used. In addition to standard AND, OR, and NOT elements, two bistable elements are required. The two elements that will be used are the Muller C-element and the Arbiter. The Muller C-element dates back to early considerations of asynchronous logic [8]. Its behavior is shown in Figure 2.6. The C-element is a bistable circuit whose output changes whenever all of the inputs have the logical value that is the complement of the output logic value. In this implementation of the packet router, the C-element will be used for storage in feedback loops, and for switching and multiplexing logical variables. The Arbiter is a more common logic element. It behavior is demonstrated in Figure 2.7. It has two inputs and two outputs. Whenever one input has a logic-1 input, its corresponding output has a value of logic-1. If both of the inputs, have a logic one, only one of the corresponding outputs will have a value of logic-1. Ideally, the output corresponding to the input that rises first should have a value of logic-1. If both of the inputs have a value of logic-0, then both of the outputs shall have a value of logic-0. For the purposes of presenting the implementations of the various data-flow modules, the practical problems of implementing any of the above logic elements will not be discussed until later. For now, these elements will be treated as ideal.

Before the implementation of the various modules comprising the packet router can be considered, the basic issues of representation of signals and variables must be considered. Furthermore, some form of Ready/Acknowledge protocol must be incorporated into the signalling representation to allow for the pipelining implied by the use of data-flow modules in the functional description. This issue was considered by Leung [3] and his approach will be followed. Variables are represented by a dual-rail code as shown in Figure 2.8. Dual-rail coding is not new to asynchronous circuit design. It was explored by Sims and Gray [9], and Gilchrist, Pomerine and Wong [10]. The basic principle of dual-rail coding is to

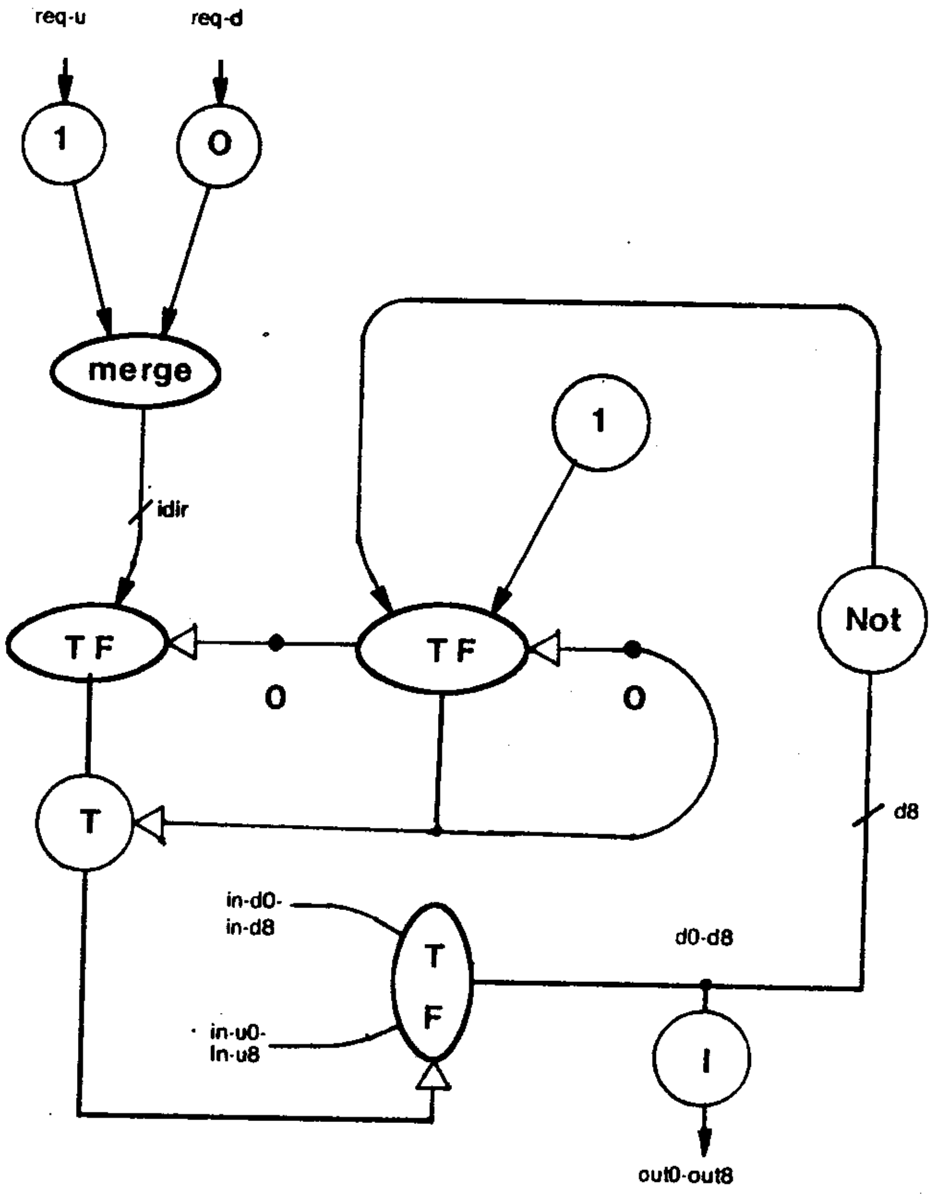


Figure 2.5 Bit-Level Data Flow Representation of the OM

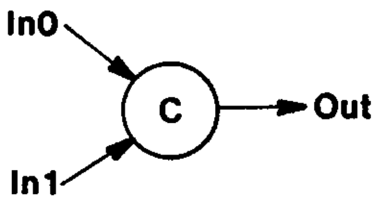


Figure 2.6a Muller C-Element

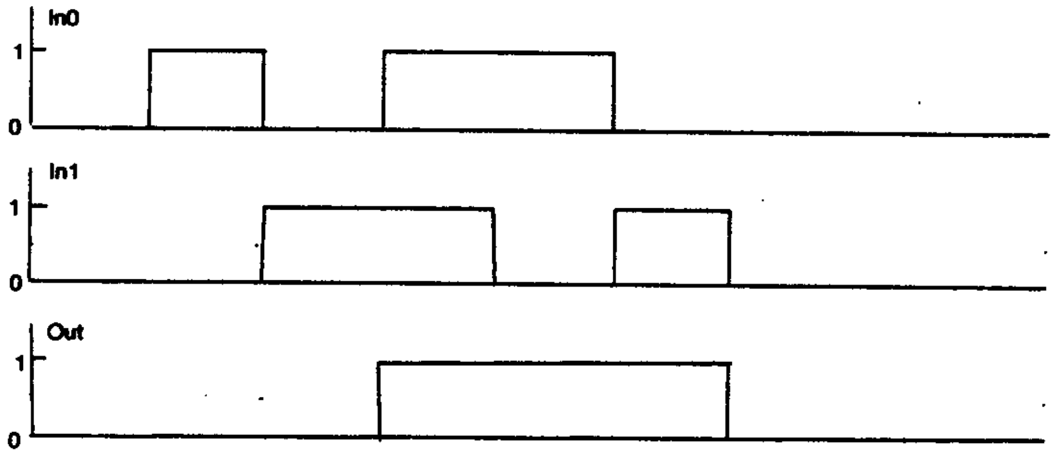


Figure 2.6b Behavior of C-Element

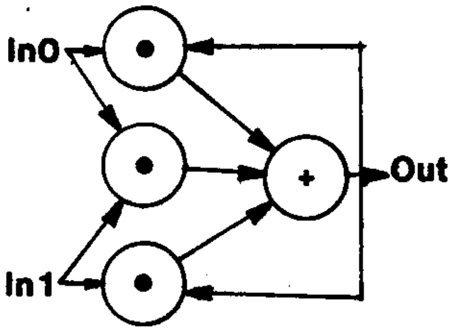


Figure 2.6c Logic Description of C-Element

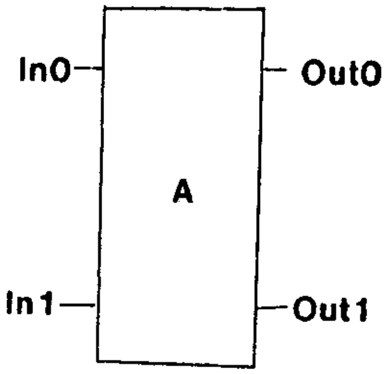


Figure 2.7a Arbiter

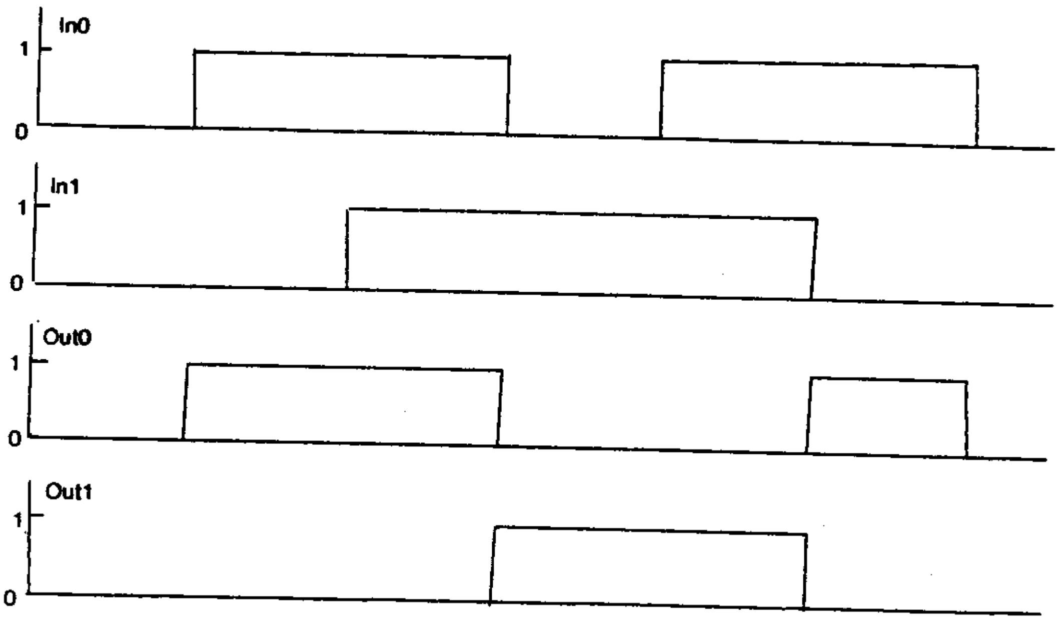


Figure 2.7b Behavior of Arbiter

embed the Ready signal in the data. Each data bit is represented by two wires: the One wire and the Zero wire. Logic value-1 and logic value-0 are represented by positive transitions on the One and Zero wires, respectively. A n-bit variable using 2n wires paired with a single wire that carries the Acknowledge signal, which is also denoted by a positive transition. This implementation also requires that only one of the two data wires for each bit has a logic value of 1. The Ready/Acknowledge signalling, then, is easily

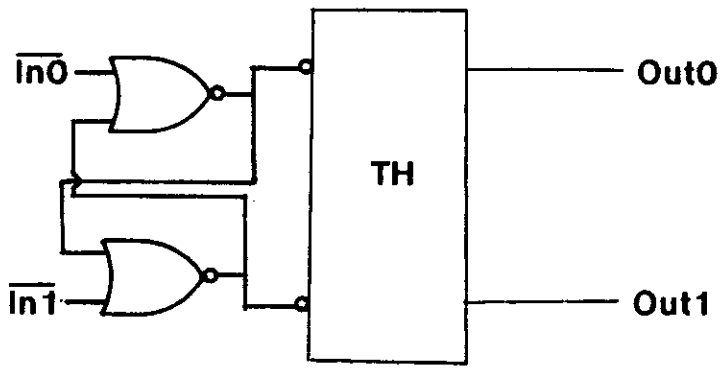


Figure 2.7c Logic Diagram of Arbiter
(TH = Threshold Element)

---

expressed. A Ready is a positive transition on one of the two wires of each bit of a variable. An Acknowledge is a positive transition on the Acknowledge wire. The data wires are allowed to fall again after the receipt of an Acknowledge signal. The Acknowledge wire is allowed to fall when all of the data wires have fallen to a logic-0 value (this is called a spacer state). We have specified a four-cycle Ready/Acknowledge scheme, with the Ready signal encoded in the data. For forward signals like Req_U and Req_D in the CM and OM, a single wire is used for Ready signalling, as no other data is required. The Ready/Acknowledge scheme used above is quite adequate, and is used for signal as shown in Figure 2.9.

We have have specified the elements that are used in the packet router, and we have specified the data and signal protocols, so we can now consider implementation of the various hardware modules that will comprise the packet router.

The data-flow representations for the CM and the OM (as shown in Figures 2.4 and 2.5) use T- and F-modules to perform switching and multiplexing functions. Leung [3] has specified separate modules for these functions, because the structures necessary to implement them are different. The modules that perform these functions are called the SWITCH and MUX modules, and are shown in their various

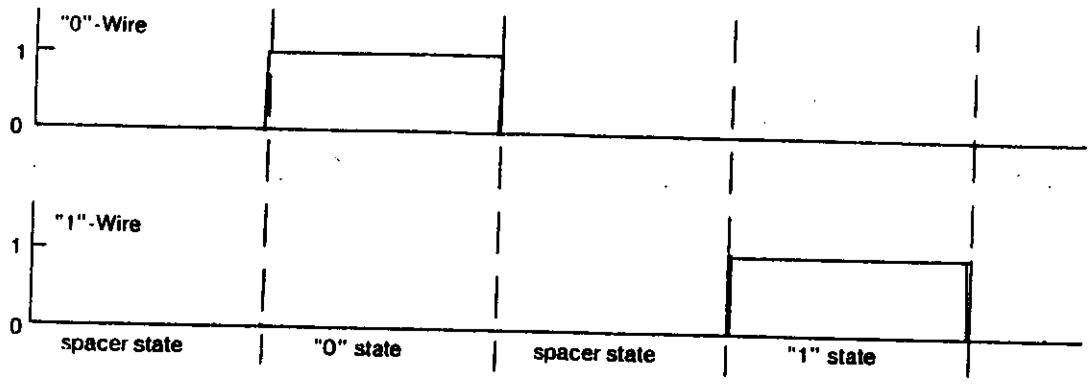


Figure 2.8 Dual-Rail Code

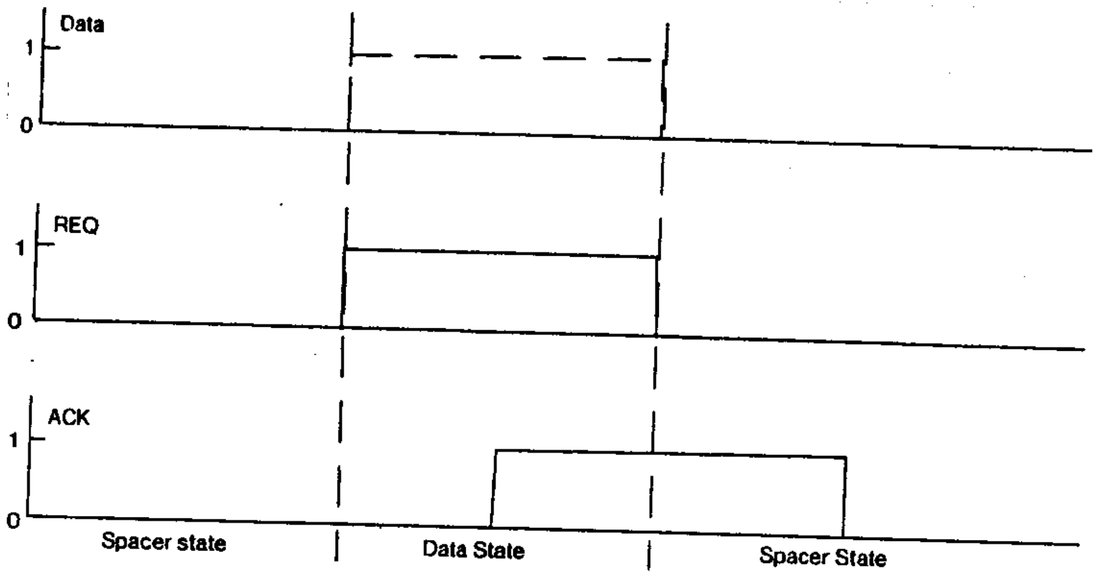


Figure 2.9 Ready/Acknowledge Protocol

representations in Figure 2.10.

The SWITCH module can be implemented using C-elements as shown in Figure 2.11. The SWITCH module requires a data input and a control input. The data input can be a signal or a variable, so the module is implemented for any number of signal wires, and a single Acknowledge wire. For our

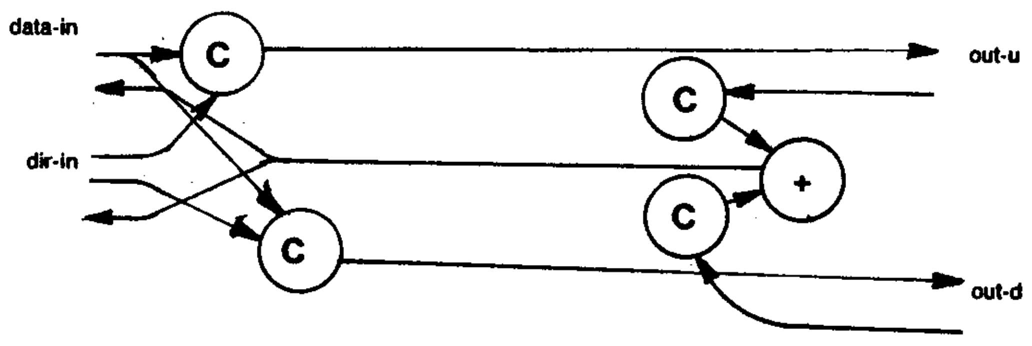


Figure 2.10 C-Element Implementation of a SWITCH Module

---

purposes, two outputs are adequate for each input, so the control input is a single bit variable and, therefore, is implemented as two input wires and an Acknowledge wire. Two C-elements are used for each data wire, one for each output wire. The appropriate control variable wire is connected to each C-element, so that the data is transmitted the output specified by the value of the control variable. The Acknowledge signal should only come from the selected output, so two C-elements are used to multiplex the Acknowledge signals together to give a single Acknowledge signal for the module previous to the SWITCH module. The control variable wires are used to decide which of the Acknowledge signals to take. The SWITCH module, then, requires two C-elements for each data wire, and two C-elements and an OR gate for the Acknowledge.

The MUX module can also be implemented using only C-elements, as shown in Figure 2.12. The MUX module has two data inputs and one output. It also has one Acknowledge input and two Acknowledge outputs. The C-elements, then, multiplex the data lines and switch the Acknowledge lines. The C-element implementation, as shown in Figure 2.12, uses the same configurations as the

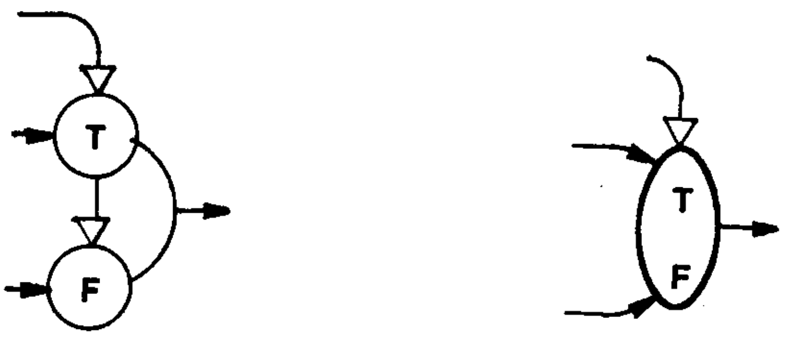


Figure 2.11a Data-Flow Representation of a SWITCH and a MUX

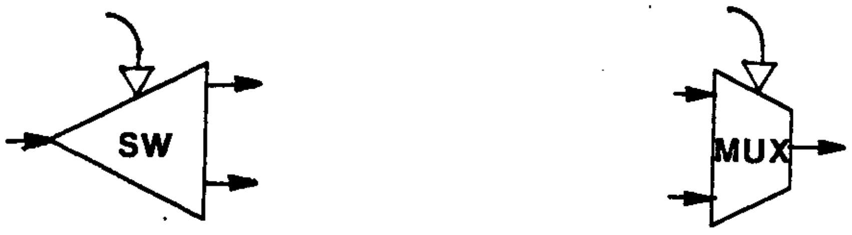


Figure 2.11b Module Representation of a SWITCH and a MUX

implementation of the SWITCH module, but in different places. Likewise, the MUX module requires two C-elements and an OR gate for each output bit and two C-elements for the Acknowledge line.

The feedback paths used in the CM and the OM require C-elements for storage. Unlike the SWITCH and MUX modules, a storage element must generate an Acknowledge signal, rather that just passing one along. To generate an Acknowledge signal, a storage element must be able to extract the Ready signal encoded in its own data output. The storage element, called a REG module, consists of two parts: a storage part consisting of a C-element for each bit, and a data detection part that generates an Acknowledge signal from the output of the storage part. The implementation of the REG module is shown in Figure 2.13. The storage section only needs to hold the input values at its output until an Acknowledge signal is received from the following module. Thus, the storage part consists solely of an inverter, and a C-element for each data input. The data detection part has to conform with the four-cycle Ready/Acknowledge protocol presented earlier. The output of the data detector should rise when data is

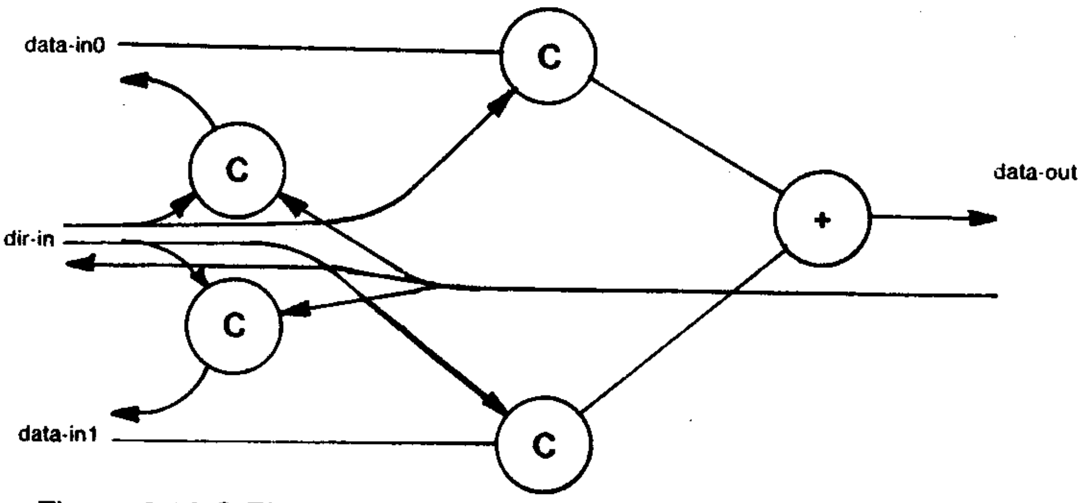


Figure 2.12 C-Element Implemenation of a MUX Module

detected and fall when the data has been removed (a spacer state). This can easily be implemented with a C-element with one input connected to the AND over all of the variable bits of the Exclusive-OR of the two wires representing the bit, and one input connected to the OR of all the data wires. When all the wires are in a data state, the resulting Acknowledge line goes high, and when all the wires are in a spacer state the Acknowledge line drops. Thus, the REG module can be implemented with two C-elements and an Exclusive-OR gate for each bit of the variable in addition to an Inverter, a C-element, and an AND gate for the Acknowledge signals.

The feedback paths in the CM and the OM can be implemented with REG modules as described above. The feedback paths requires storage of two variables simultaneously, one is the old value of the stored variable, and the new value that will replace the old value. Furthermore, it should be noted that two REG modules holding variables require a third REG module between them because of constraints applied by the Ready/Acknowledge protocol. A feedback path as implemented with three REG modules is shown in Figure 2.14.

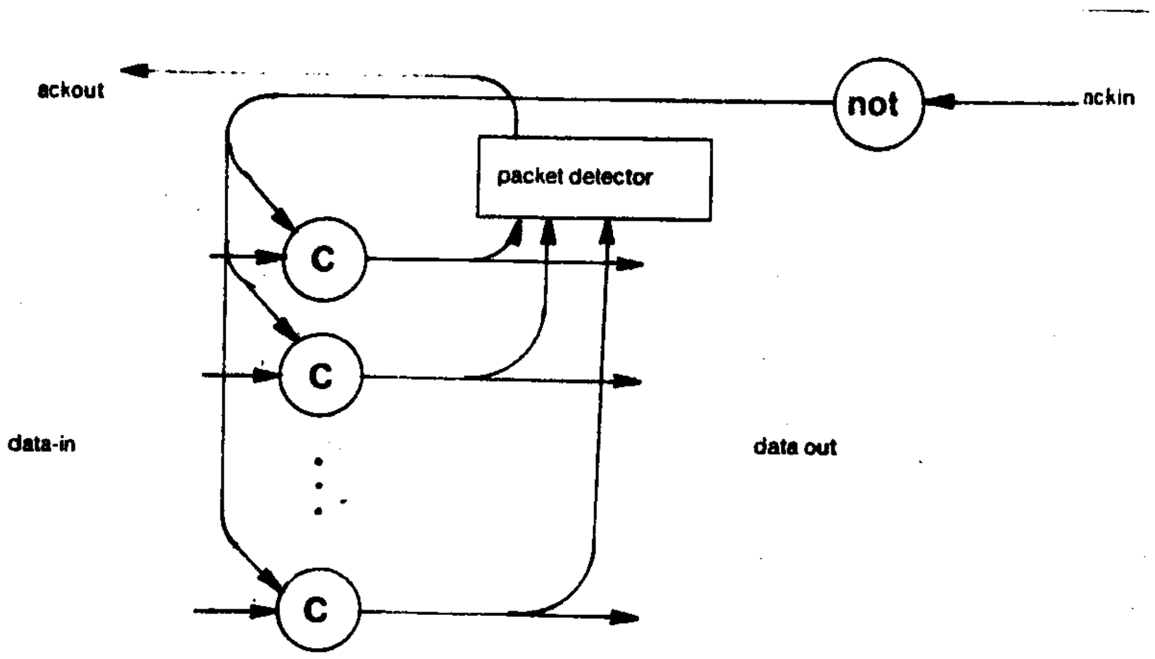


Figure 2.13a REG Module

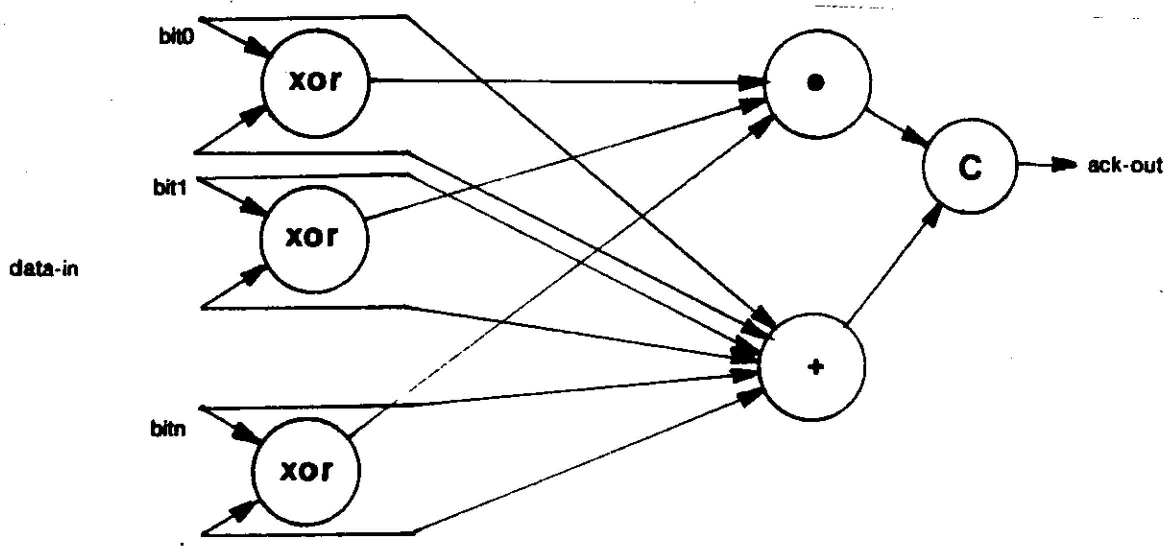


Figure 2.13b Packet Detector

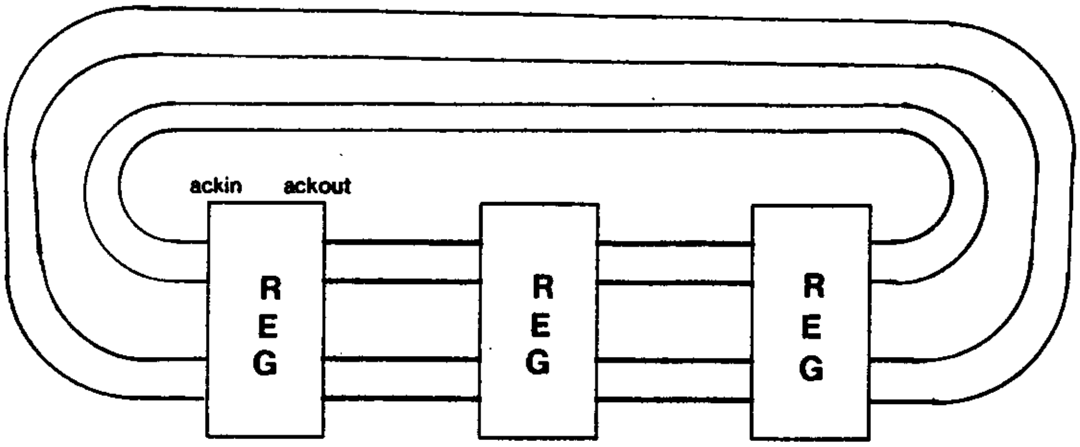


Figure 2.14 REG Module Feedback Path

The feedback paths used to implement the CM and the OM require an initialization feature. The circuits required to implement initialization of REG modules will not be presented here. The important issue is how to initialize a feedback loop, given a REG module that can be initialized. Initialization of the OM or the CM requires that every C-element that isn't part of an initialized feedback loop have a logic value of 0 on both inputs. This can be accomplished by setting the inputs of the packet router to zero, and initializing the output REG cell of the feedback path to a spacer state. We can then initialize the middle REG cell to the desired initialization value, and initialize the input REG module to a spacer state. The initialization of a feedback loop is illustrated in Figure 2.15.

It should also be noted that initialization is required for the use of REG modules, because it is possible for a pair of REG modules to hold an illegal state that is stable, and will block data flow. It is clear that a C-element will come up in either an on state or and off state. Suppose the first module of a pair of REG modules comes up in an illegal state (not a data state or a spacer state), and its packet detector C-element comes up in a zero state (no Acknowledge). Also suppose that the following REG

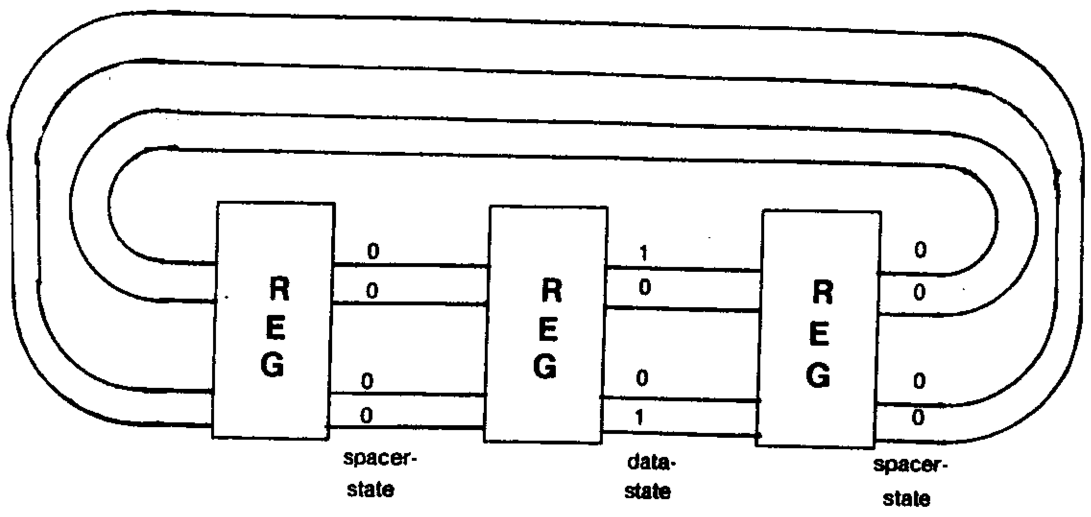


Figure 2.15 Initialization of REG Module Feedback Path

module comes up in a spacer state, which will also give no Acknowledge. The illegal state will pass on to the second module and will remain there, because its output (also an illegal value) will not generate an Acknowledge from the next stage. Any new input byte will get ORed with the illegal state, so that the state remains stable. Thus the REG module pair is in a dead-locked state, which is quite undesirable. For out implementation of the REG module, then, initialization is a requirement for a working system. A mechanism will be developed for this purpose, when the circuit implementation for the REG module is developed.

We recall that our chip input/output signals are in single-rail Ready/Acknowledge format, while our on-chip signals are in dual-rail format, with a single wire Acknowledge signal. Modules are needed to convert from one of these forms to another. Seitz [4] considers this problem, and presents two circuit designs shown in Figure 2.16. The first of these converts single-rail to dual-rail coding. The circuit is very simple: if the Ready signal is on, either the one or the zero wire is turned on, depending on the input logic value, which requires only three logic gates. The circuit that converts dual-rail to single-rail consists of two input OR gates for each bit of the variable, and a C-element with enough inputs to accommodate the

OR gates. The function of this module is identical to that of the packet detector, but it appears that this implementation might be more compact. The differences occur when the input is not a valid data or spacer state. This implementation will generate a Ready signal for an invalid input, as long as at least one of each dual-rail pair is on. The packet detector never generates a Ready signal for an invalid input, unless the input is first valid, and then changes to invalid. The Two-to-One Converter, as we shall call this module, does generate an Acknowledge for an invalid state, but it requires a valid spacer state afterwards to lower the Acknowledge line. It is desirable for a packet detection circuit to fail to detect invalid bytes, rather than passing passing them on, to simplify fault detection in a large system like the routing network these modules will be used to build.

As fault detection is an important quality for large systems, it is pertinent to our design description to examine in more detail the difference between the SINK module packet detector developed earlier, and the Two-to-One data detector we have just discussed. For certain static faults, like stuck-at faults (a wire is tied to Ground or V_{DD} instead of passing its intended signal), there is very little difference

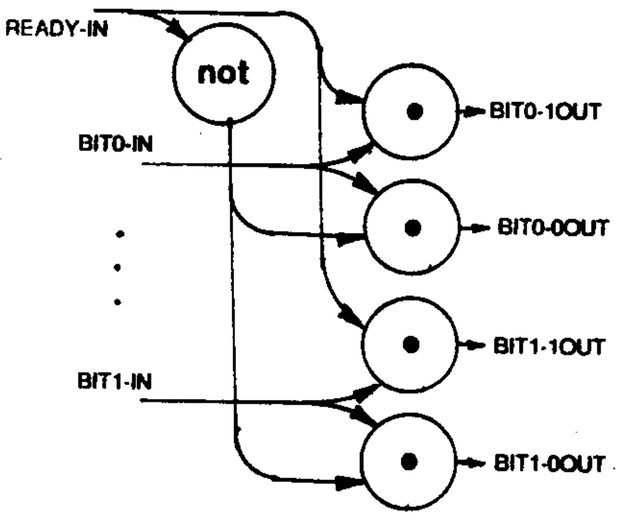


Figure 2.16a One-to-Two Converter

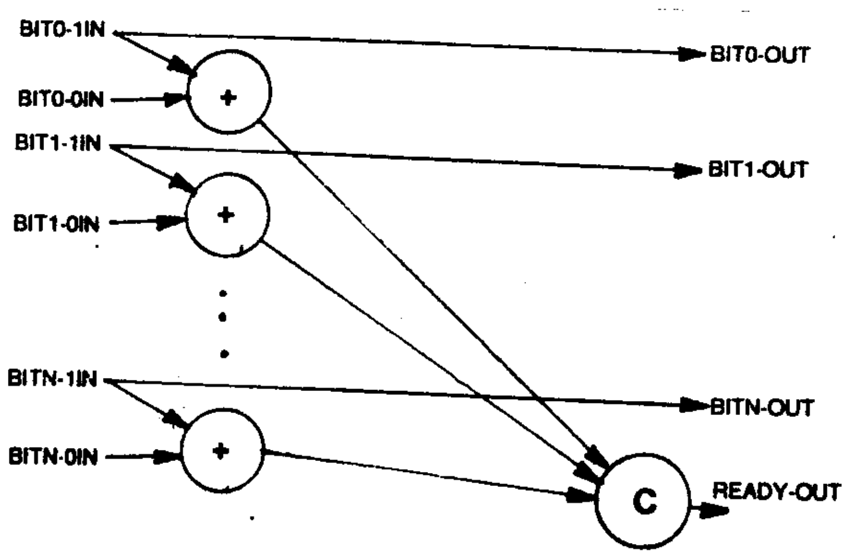


Figure 2.16b Two-to-One Converter

between the two. The SINK module will never acknowledge a byte that tries assert a stuck-at-Ground wire, or a byte that tries to lower a stuck-at-V_{DD} wire. The Two-to-One Converter will never acknowledge a byte that tries to assert a stuck-at-Ground wire. If the byte tries to ground a stuck-at-V_{DD} wire, the Acknowledge output will go high, but will not go low after the byte has been acknowledged, which is just as good for fault-detection purposes. For most types of shorts, the Two-to-One Converter does about as well as the SINK, because of the fact that nMOS logic pull-down transistors can sink significantly more current than the pull-up transistors, because of the ratios of their W/Ls. If an asserted wire is shorted to a low wire, the low wire will pull down the asserted wire, and the neither packet detector will work. In some cases, though, wires from a from two different sized logic gates, or from different types of logic gates (like a NAND and a NOR) may short such that the pull-up transistor of one is capable of defeating the pull-down transistor of the other. In this case, illegal states may get passed by the Two-to-One Converter, but would not be passed by the SINK module. If dynamic faults occur (glitches, etc.), the SINK may or may not have an advantage over the Two-to-One Converter, depending on the nature of the fault, but it seems likely that the SINK module will be better on a statistical basis, which may help fault detection some, as these kinds of faults are very difficult to find in any form of logic system. The choice between the SINK module and the Two-to-One Converter will be left to the designer. The trade-offs will be area, power, and speed versus better use of dual-rail coding as an error-detection coding.

Another group of modules that must be implemented are the basic logic functions AND, OR, and NOT for dual-rail variables. The NOT function is simple, all that is necessary is to reverse the wires corresponding to the bit to complemented, so that the wire denoting a logic value-1 becomes the wire denoting logic value-0, and vice-versa. This property of dual-rail logic can be used to reduce the problem of implementing the AND and OR modules to that of implementing only one of the two. DeMorgan's theorem can be used, switching the input and output wires to apply a logic NOT function to each, on a AND module to get an OR module. The AND, OR, and NOT functions can be implemented, and only one module design is necessary, as shown in Figure 2.17. The AND function can be implemented with one C-element for the logic-1 output wire, and three C-elements, with their outputs ORed for the logic-0 output. The Acknowledge signals for these logic signals is simply the Acknowledge signal from the next stage, so no special circuitry is required.

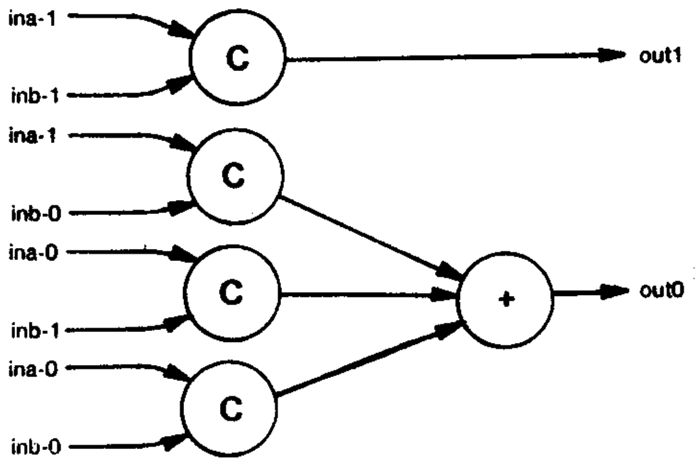


Figure 2.17 Dual-Rail AND Gate

A problem that occurs at this stage is how to fan in the Acknowledge signals if a fan out occurs at the output of a module. We want to send back an Acknowledge when all of the following stages have sent back Acknowledges, and we want the Acknowledge line to stay high until all of the Acknowledge lines from the following stages have dropped. This can easily be accomplished by an N-input C-element where N is the fan out of the variable at the output of the module, as illustrated in Figure 2.18. An N-input C-element can be implemented by cascading two-input C-elements as also shown in Figure 2.18.

The only remaining module is the MERGE module. The heart of a MERGE module is an Arbiter, as shown in Figure 2.19. We simply use the Arbiter to choose between the two data inputs using the outputs of two data-detectors, as designed for the REG module, as inputs. C-elements can be used to

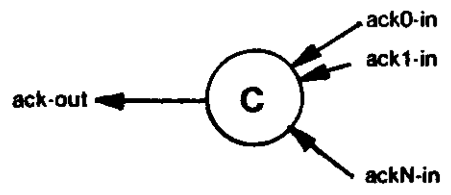


Figure 2.18a Combining N Acknowledge Signals

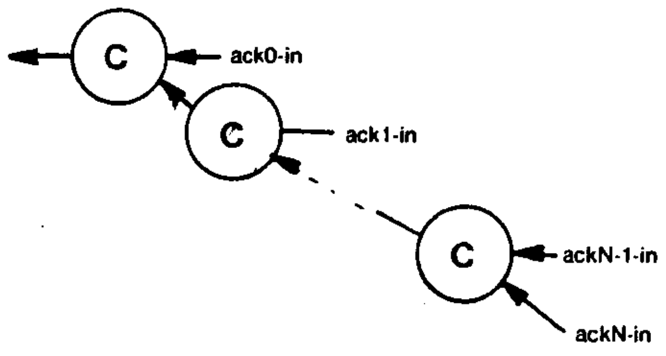


Figure 2.18b One Implementation of an N-Input C-Element

multiplex the two inputs together depending on the outputs of the arbiter. The Acknowledge lines are ORed with the outputs of the data-detectors, so that the Arbiter will remain engaged until the complete four-cycle Ready/Acknowledge transfer is complete. The outputs of the Arbiter are ANDed with the outputs of the data-detectors to multiplex the data lines and switch the Acknowledge lines, so that the Acknowledge lines won't drop until the Arbiter is disengaged.

Finally, two simple modules have to be developed to complete the set of necessary modules. We have not specified a module to replace the data-flow SOURCE modules used in Figures 2.4 and 2.5. A SOURCE module can be implemented by inverting the Acknowledge line from the next module and

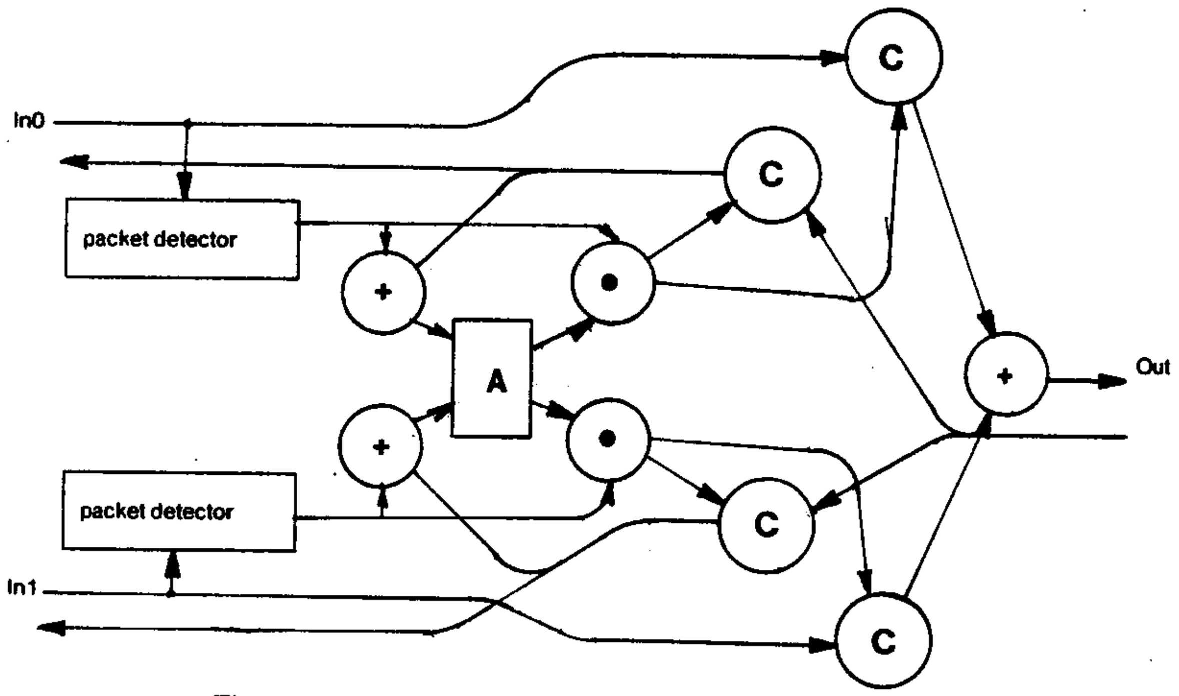


Figure 2.19 MERGE Module Implementation

using this value to drive the lines that represent the logic values that we wish the SOURCE module to transmit. The other lines can be tied low. Thus, when the SOURCE module, gets acknowledged by the next module, the data lines drop. When the Acknowledge line drops, the next SOURCE byte is sent. This implementation satisfies the four cycle Ready/Acknowledge protocol. Single bit logic-1 and logic-0 SOURCE modules are) illustrated in Figure 2.20. There are also cases when we need a SINK module. These cases occur in our data-flow representations whenever a variable enters a T-module that is not accompanied by an F-module, or vice versa. When a byte arrives that is not transmitted by the T-module, or in the opposite case, the F-module, the packet must be transmitted by a complementary module to a SINK module so that the byte will get acknowledged, and not cause a dead-lock. Each case of a single T-module or F-module in Figures 2.4 and 2.5 will get replaced by a MUX module with the output that does not correspond to the data-flow module going to a SINK module. The implementation of a SINK module has already been presented as the data-detector in the REG module, and will not be repeated here.

All the modules necessary for the implementation of the CM and the OM have been specified in terms of several basic logic elements. An implementation of the CM and the OM can be found by

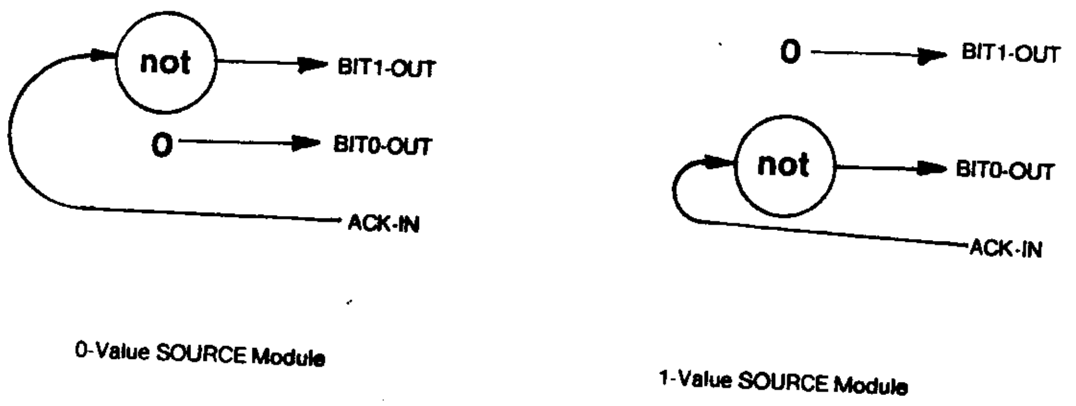


Figure 2.20 Source Modules

combining these elements as shown in Figure 2.21 and Figure 2.22. These implementations are based on implementations designed by Leung [3] using the same modules.

The structural decomposition of the basic functions that we specified for the packet router has been developed through three levels of hierarchy. First, the router was split into two basic functions implemented by the CM and the OM. Each of these was decomposed using data-flow modules. The data-flow representation was decomposed, in turn, into modules composed of several basic logic elements. The transition from the data-flow modules to the our basic hardware modules can be viewed as a transformation on a single hierarchical level. The goal of this thesis is to implement circuit designs and layouts for each of the hardware modules, using the circuit designs for the logic elements that were used to specify these modules. Unfortunately, the transition from our current level of design to a circuit and layout level of design is not as structured a transition as those we have made to our current modular level. The issues of developing minimal circuit designs for these modules, and develop layouts for the circuit designs have to be considered from the bottom-up as well as from the top-down. The basics of the integrated circuit technology used to implement the modules have to be considered before any form of circuit design or layout can be comprehensible.

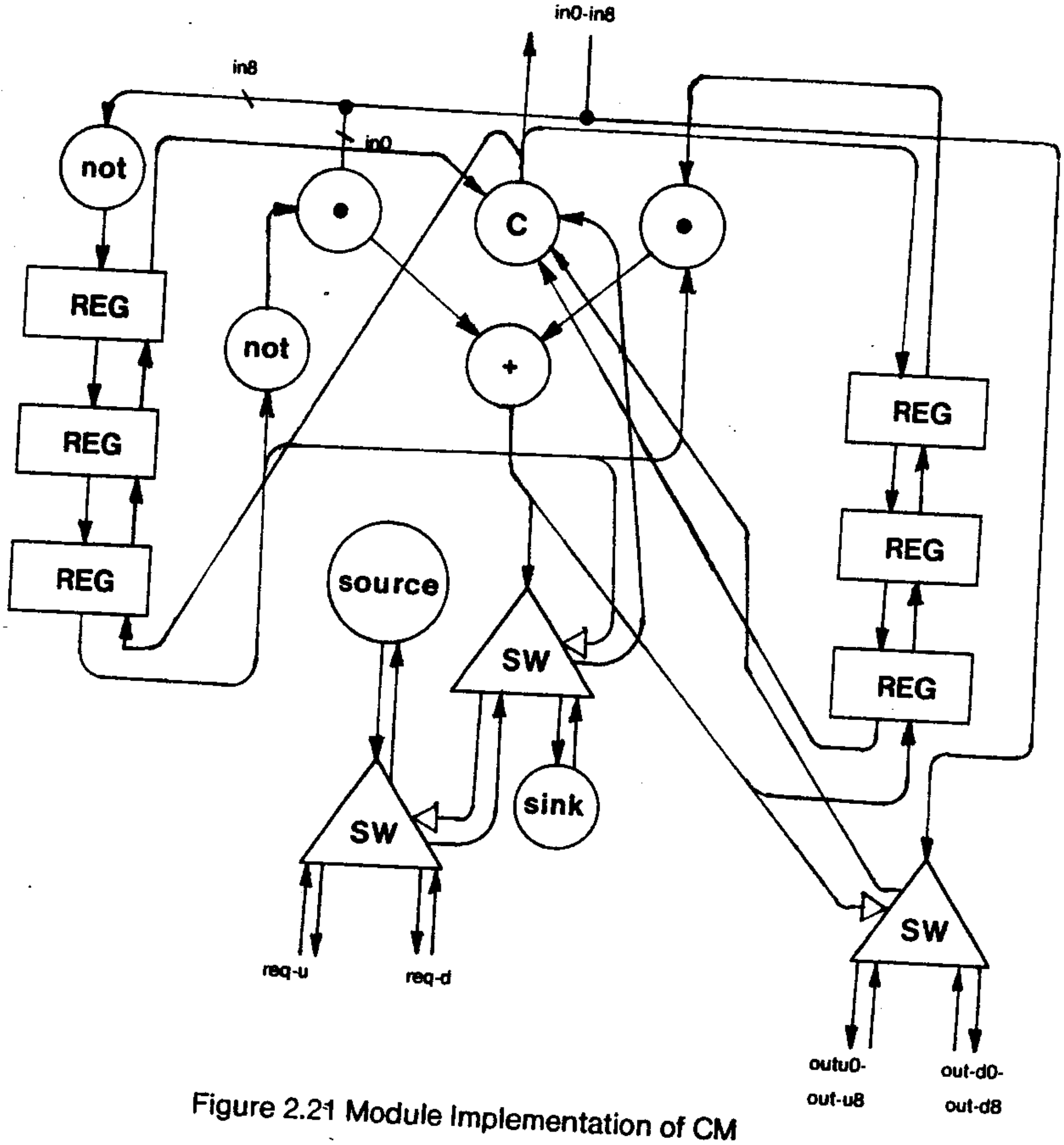


Figure 2.21 Module Implementation of CM

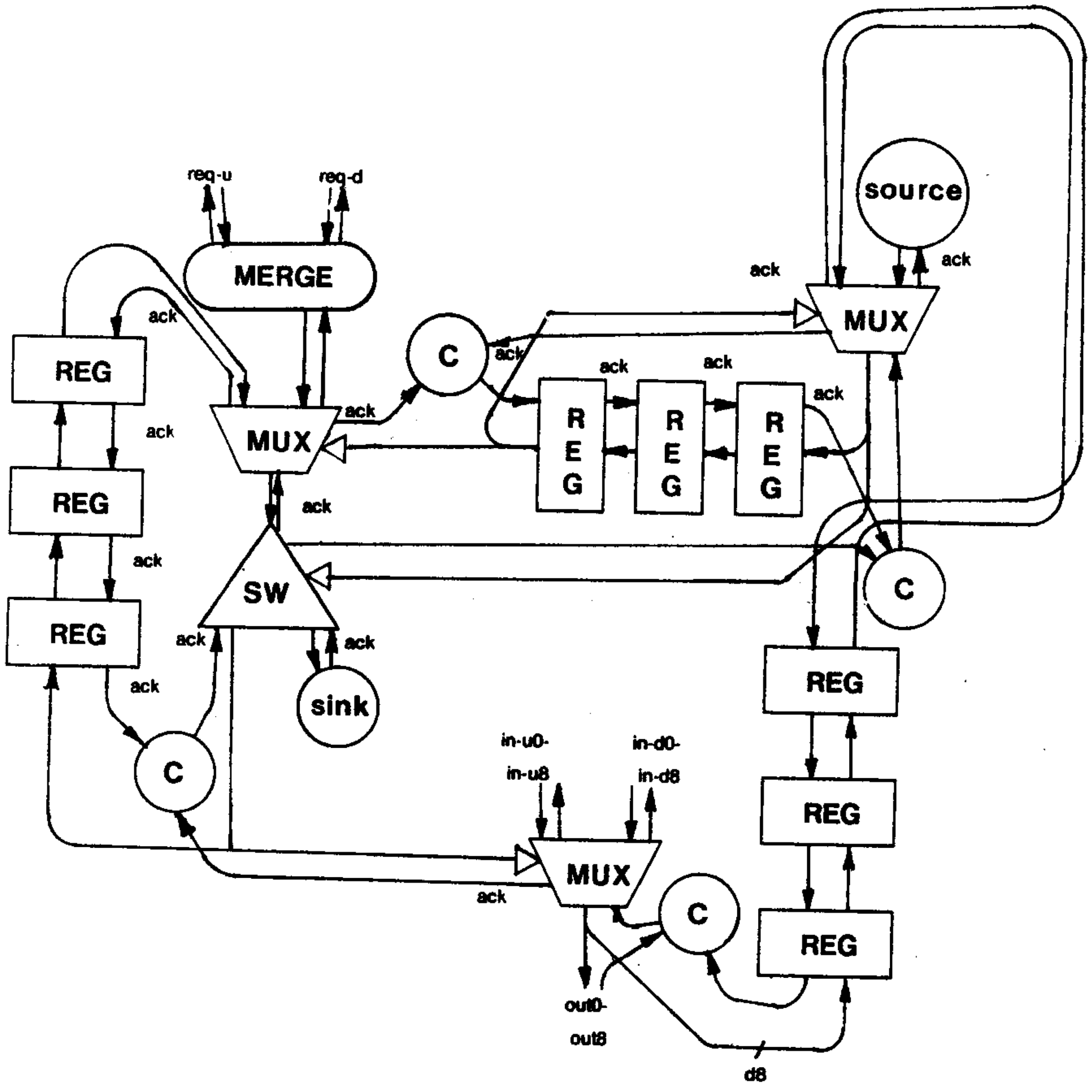


Figure 2.22 Module Implementation of OM

III. A DESCRIPTION OF MOS TECHNOLOGY

In order to make the layouts and circuit designs presented later in this thesis understandable, this chapter deals with the integrated circuit technology used to implement the router. There are three aspects of the circuit technology that are useful to gaining an understanding of the development of the layouts and circuits developed in this thesis. First, a general understanding of the processing steps required to implement the circuit technology should be gained. Next, the geometric design rules should be considered. Finally, the basic circuits that are used to implement the packet router, as well as the electrical design rules associated with them should be understood. These three areas characterize the process technology in sufficient depth to make this thesis understandable.

First, we should note that the circuit technology used for this thesis is an N-channel metal oxide semiconductor fi ld-effect transistor (MOSFET) technology. Furthermore, the logic has depletion loads, or in other words, the transistor that acts as a load has a negative threshold (depletion-mode). The pull-down transistor(s) has a positive threshold (enhancement-mode). A circuit diagram of an inverter using this technology is shown in Figure 3.1.

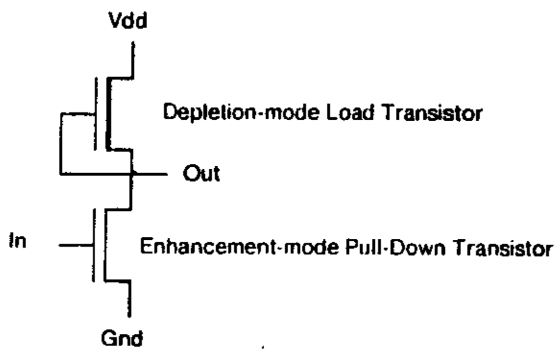

Vdd
Depletion-mode Load Transistor
Out
In
Enhancement-mode Pull-Down Transistor
Gnd

Figure 3.1 MOS Depletion-Load Inverter

Almost every N-channel MOS integrated circuit system is made by processing a thin wafer of lightly P-doped mono-crystalline silicon. The wafer is processed by introducing impurities beneath its surface, and by growing or depositing layers of oxide, metal or poly-crystalline silicon on the surface of the wafer. The logic circuits are produced by interactions between the various layers making up an integrated circuit. The layers are patterned according to the layouts to produce logic circuits. The patterns on various layers are achieved by a sequence of processing steps (see Figure 3.2) as described by Mead and Conway [6].

The basic processing sequence begins with a mono-crystalline, P-doped silicon wafer, on which a layer of silicon dioxide has been grown. Photographic lithography techniques are then used to cover a portion of the wafer with an organic substance that will act as a mask that resists etching. The portions of the silicon dioxide not covered by the masking material will be etched down to the silicon surface by

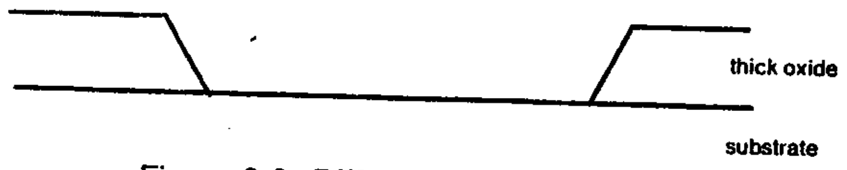


Figure 3.2a Diffusion Cut

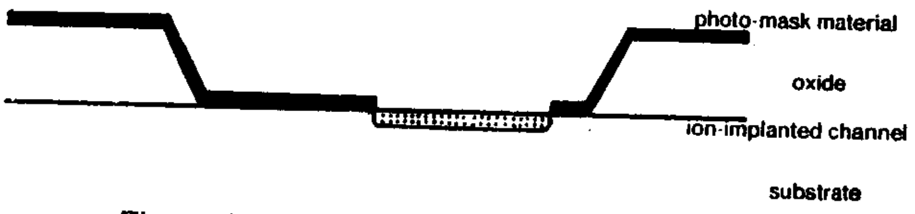


Figure 3.2b Ion-Implant Step

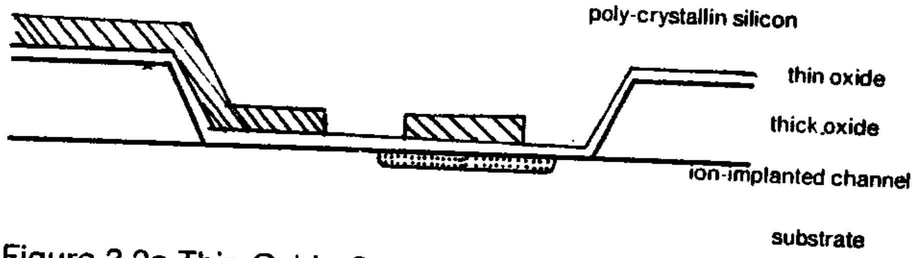


Figure 3.2c Thin Oxide Growth and Poly Deposition

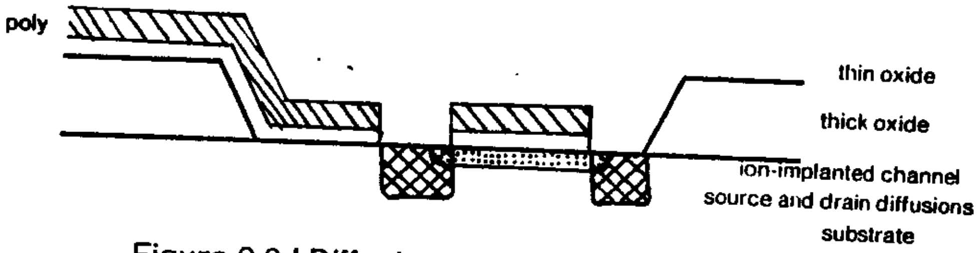


Figure 3.2d Diffusion of Source and Drain Regions

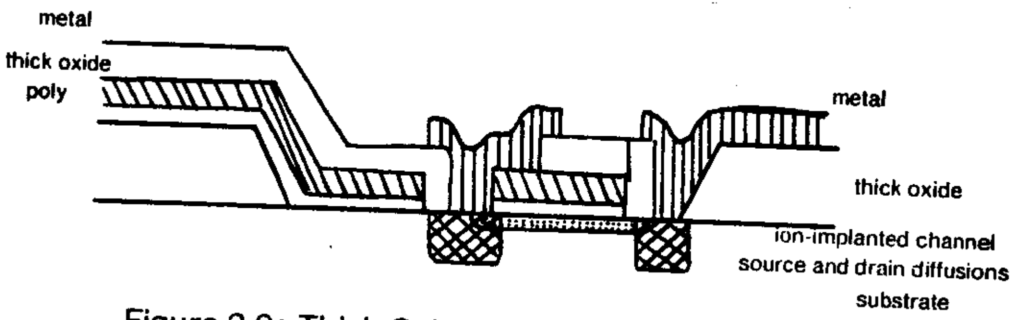


Figure 3.2e Thick Oxide Growth, Contact Cuts, and Metal Deposition

Figure 3.2 Processing Steps

exposing the wafer to hydorfluoric acid. The mask used for this processing step is known as the diffusion layer mask. The diffusion layer will cover the channel regions of all the MOSFETs, as well as the diffusion regions that act as the source and drain regions of the MOSFETs, and interconnect between these regions. The masking material is then dissolved, and a second photo-masking step is used to mask all areas on the exposed silicon surface that are not to be channel regions for depletion-mode transistors. The mask used for this is called the ion-implant layer mask. The remaining parts of the wafer are doped using ion-implantation, to produce depletion-mode channels wherever the ions cover a transistor channel region. The second photo-mask is dissolved, and a layer of thin silicon dioxide is grown. Next a conductive layer of poly-crystalline silicon (polysilicon or poly) is deposited on the wafer. A photo-masking and etching process will remove poly and thin oxide from all but the desired regions of the wafer. The mask used for this is called the poly layer mask. The poly layer defines the gates of all of the transistors, as well as poly wires used for interconnection. Next, the wafer is exposed to a source of N-type dopant, which is allowed to diffuse into the silicon wherever the silicon surface is exposed. The wafer is inherently masked from the diffusion wherever there is still thick oxide, or wherever poly was left from the previous step. This property of these processing steps taken in this order is called diffusion self-alignment, as the gate region aligns the source and drain diffusions by masking the channel region. Next a layer of thick oxide is grown. A photo-mask step, using the contact layer mask, and an etch step then opens contact cuts through the oxide to the silicon surface, or the poly surface. Next, a layer of aluminum is deposited, and a photomask step, using the metal layer mask, and an etch step removes undesired metal. Another oxide growth step, along with a photo-mask and etch step to open contact cuts for the input or output pads completes the processing. It should be noted that transistors occur wherever a poly layer crosses a diffusion layer, as the region below the poly does not get exposed to dopants that result in a conductive diffusion layer. If, however, the transistor area is ion-implanted, a depletion-mode transistor results. These processes result in what are called silicon-gate, diffusion self-aligned (DSA) transistors.

The process technology described above has certain geometric limitations. There are certain minimum values for various separations and line widths on the layout. These minimum values are determined by a variety of constraints on the process technology. The geometric constraints are, in general, limited by the resolution of the process (eg.: resolution of lithography), or by the circuit and device physics considerations (eg.: capacitance between adjacent wires). For the purpose of this thesis, the design rules developed by Mead and Conway [6] are used because of their simplicity. Mead and Conway present a generalized set of design rules that are valid irrespective of process variations by introducing a variable scaling factor called λ. For any set of processing constraints, λ can be changed so that none of the design rules violate the processing constraints. This approach may not yield the most area-efficient or speed-efficient designs, but it does simplify the task of the designer, and allows the use of any depletion-load, silicon gate, DSA MOSFET technology to be used in the implementation, changing only the value of λ to accommodate differences in the process technology. To give the reader some concept of scale for the geometric design rules, the λ used the fabricate the test chip presented later in this thesis was 2.5 microns.

The Mead and Conway design rules are briefly presented here for the sake of completeness. The design rules are summarized in Figure 3.3 The first group of design rules represent minimum line widths. The minimum widths of metal, poly and diffusion lines are, respectively, three λ, two λ and two λ, as can be seen in Figures 3.3a, 3.3b, and 3.3c. The next group of design rules specify minimum separations between lines on various layers. The minimum separation between two metal lines, two poly lines and two diffusion lines are, respectively, three λ, two λ, and two λ, as can be seen in Figures 3.3d, 3.3e, and 3.3f. Also, as there is an electrical interaction between the poly and diffusion layers, since an overlap of the two layers creates a transistor, a separation width must be specified for cases when a transistor is not desired. The minimum separation distance between a poly line and a diffusion line is one λ, as shown in Figure 3.3g. There is also a group of rules for the overlaps between the poly, ion-implant and diffusion layers, for cases when a transistor is desired. The minimum amount that a poly gate region must overlap a transistor (the diffusion layer that the poly region crosses, that is) is two λ, as shown in Fig. 3.3h. The minimum amount that the source and drain regions of a transistor must extend beyond a transistor without becoming narrower is two λ, and without becoming wider is one λ, as shown in Fig. 3.3i. The minimum width that an implanted region must overlap a transistor region (intersection of poly and diffusion) on all sides is one and a half λ, as shown in Fig. 3.3j. The minimum distance between an enhancement mode transistor, and an implanted region is one and a half λ, as shown in Fig. 3.3j. The last group of rules specifies the minimum dimensions for all contact cuts between layers, and the minimum dimensions for the overlap of the layers around a contact cut. A contact cut between metal and diffusion has a minimum width of two λ, and must be overlapped by metal and diffusion by one λ on all sides, as shown in Fig. 3.3k. A contact cut between metal and poly has a minimum width of two λ, and must be overlapped by metal and poly by one λ on all sides, as shown in Fig. 3.3l. The minimum spacing between contacts when connecting a large diffusion region with a large metal region is two λ, as shown in Fig. 3.3m. The mininum spacing between a contact cut in diffusion and a transistor is two λ, as shown if Figure 3.3n. Finally, a structure for connecting the poly layer to the diffusion layer without using two separate contact cuts is needed. Such a structure is shown in Fig. 3.3o, and is called a butting contact. The minimum size of the contact cut is four λ by two λ. The contact cut must be overlapped on all sides with either poly, or diffusion or both by at least one λ. The poly and diffusion layers must overlap by at least one λ. The contact cut must be overlapped on all sides with metal by at least one λ.

Having completed a description of the process used, and the circuit design rules that apply to the process, we can now consider the circuit design rules. There are two types of logic circuits that can be used with this process technology. The most common form of logic circuit is active logic, in which a logic gate has one pull-up device and one or more pull-down devices. The other form of logic is called steering logic. Steering logic consists of one or more transistors, called pass transistors, whose channels form a series connection from the output of one active logic gate to the input of another. Examples of both forms of logic are shown in Figure 3.4. When steering logic is being used for static logic circuits, it is

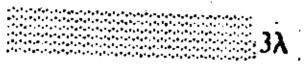


Figure 3.3a

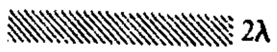


Figure3.3b

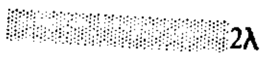


Figure 3.3c

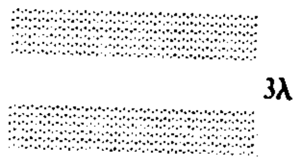


Figure 3.3d

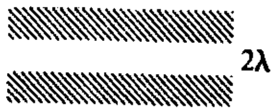


Figure 3.3e

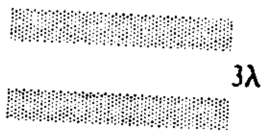


Figure 3.3f

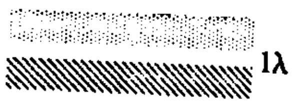


Figure 3.3g

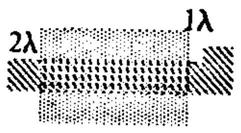


Figure 3.3h

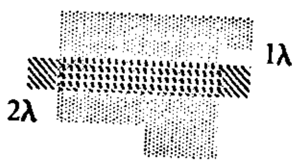


Figure 3.3i

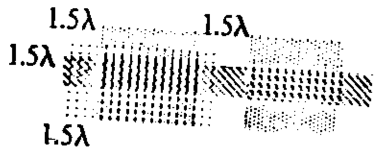


Figure 3.3j

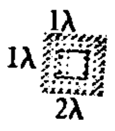


Figure 3.3k

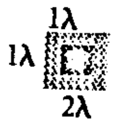


Figure 3.3l

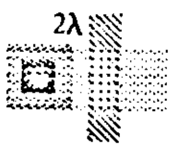


Figure 3.3m

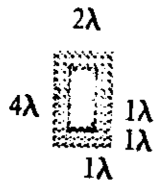


Figure 3.3n

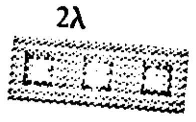


Figure 3.3o

poly ▨ ion-implant ░ metal ░ contact-cut ▩ diffusion ▒

Figure 3.3 MOS Layout Design Rules

important that every output node have a path of pass-transistors to some driven node, or else the output node will have an indeterminate value.

The active form of logic has one important circuit constraint. An active logic circuit has a property called a logic threshold voltage, which is defined as the input voltage that results in an identical output voltage. The constraint that concerns active logic is that the threshold be halfway between Vdd and ground. The purpose of this constraint is to maximize noise immunity. Since the voltages associated with the logic values 1 and 0 are close to Vdd and ground, respectively, noise immunity is maximized by putting the logic threshold at half of Vdd. This constraint can be combined with various circuit constraints and processing constraints, and can be optimized with respect to current driving capability of the pull-up transistor and the pull-down transistor in an inverter circuit [6]. The resulting optimal transistor sizes can be related in terms of the width to length ratio (W/L) of the two transistors. It is a fundamental property of planar MOSFETs that current driving capability is proportional to W/L. The optimal ratio of W/L for the pull-down transistor to W/L for the pull-up transistor is 4:1. The result can be extended to multi-transistor pull-down structures by treating turned-on pull-down transistors as resistors of size L/W and use resistor series-parallel combination rules to find the maximum equivalent

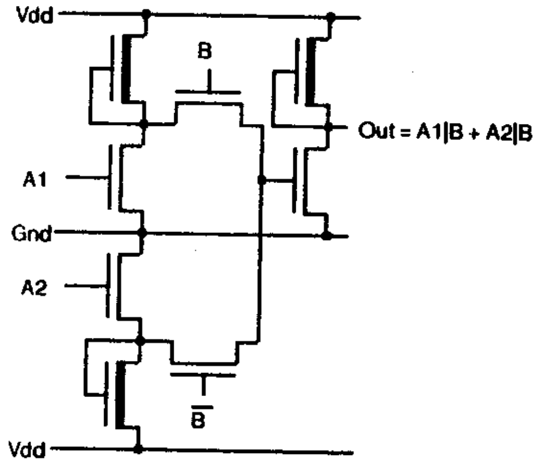


Figure 3.4 Steering Logic Example

W/L for any combination of inputs that should turn off the logic gate, to use as the effective pull-down W/L, which will specify the W/L for the pull-up transistor. Several examples are illustrated in Fig. 3.5. Thus, the electrical design rule for active logic has been specified.

Other electrical design rules concern the use of steering logic. The use of pass transistors has two basic problems. Whenever the input attached to the channel (in this case the drain) is higher than the voltage at the output of the channel, the pass transistor acts as a pull-up transistor. In this case, the pass transistor cannot pull the output any higher than one enhancement threshold below the channel input, so that the logic gate following a pass transistor receives an enhancement threshold less voltage drive than

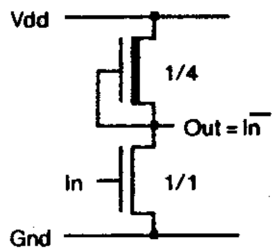


Figure 3.5a Inverter

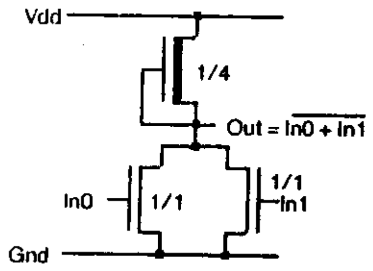


Figure 3.5b NOR Gate

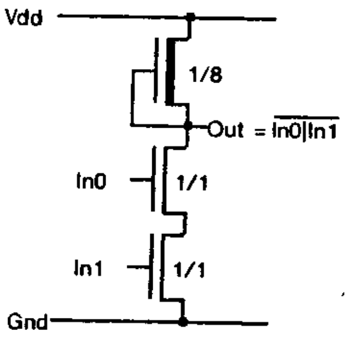


Figure 3.5c NAND Gate

Figure 3.5 W/L Examples

would a logic gate following another logic gate. To compensate for this problem, a higher pull-down to pull-up W/L is required so that the stage following a pass transistor can drive its output as hard as a logic gate driven directly by another logic gate. Mead and Conway [6] calculate the appropriate ratio of pull-down W/L to pull-up W/L to be 8:1, for any length chain of pass transistors. Another problem arises, however, when very long chains of pass transistors are used. The problem is related to the delay time of a long chain of pass transistors. A chain of pass transistors can be modeled as an R-C delay line, and therefore, the delay of a chain of n pass transistors becomes proportional to n^2 as n becomes larger. It becomes desirable, so that the delay doesn't become excessive, to break up a chain of pass transistors every so often, using an active logic gate, which restores the signal, and breaks up the diffusion nature of the signal propagation. Mead and Conway calculate the optimum number of pass transistors between active logic gates to be about four.

A major problem of ratioed logic concerns its inherent asymmetry. The enhancement-mode pull-down transistor have four times the current-driving capability of the depletion-mode load transistor in the basic inverter. This results in an rise-time that is roughly four times slower than the fall-time. It is desirable in some applications to apply a voltage to the pull-up transistor, rather than to connect it to the output node of the inverter, to increasing its current-driving capability during the first part of the pull-up transient. To accomplish inversion, the voltage must corresponding to the logical inverse of the voltage driving the pull-down transistor. A circuit with a driven depletion load, must therefore be preceded by an inverter. The resulting circuit is called a super-buffer, and can be constructed in an inverting or a non-inverting form, as shown in Figure 3.6. The super-buffer is most useful in driving large capacitive loads, since pull-up transient will be significantly shortened. In other applications, however, use of the super-buffer may not be warranted, because of the penalty of extra-size, or the extra delay inherent in using two gates instead of one.

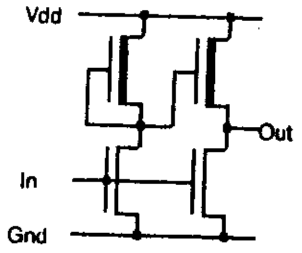


Figure 3.6a Inverting Super-Buffer

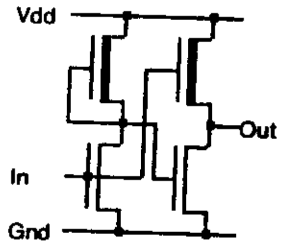


Figure 3.6b Non-Inverting Super-Buffer

Finally, we can develop the basic circuits that make up the packet router. The C-element and the arbiter form the basis for implementing the modules developed in the previous chapter. N-MOS implementations for both of the circuits are given by Seitz [4]. Circuit diagrams for the C-element and the arbiter are shown in Fig. 3.7. The C-element consists of two logic gates. The first logic gate is a multi-function logic gate, which is the NOR of the AND of the inputs with the AND of each of the inputs with the feedback value. The feedback value is the complement of the output of the multi-function logic gate. The C-element only changes its output when all of its inputs agree. The arbiter consists of a cross-coupled NOR-gate latch and a threshold gate. The output of a NOR must be low, and the output of the alternate NOR must be high to allow an output to be pulled low. It should be noted that the inputs and outputs of this circuit are negative-logic, and should be inverted to give a positive-logic implementation.

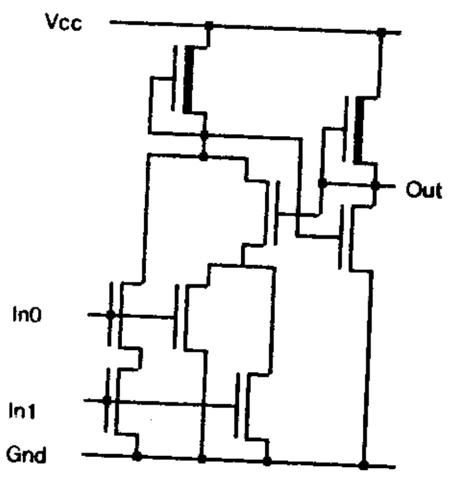


Figure 3.7a C-element Circuit Design

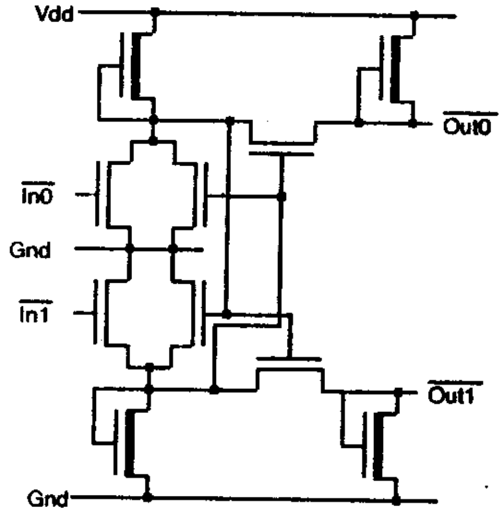


Figure 3.7b Arbiter Circuit Design

In this chapter, some relevant details of the process technology used for implementing the modules of the packet router were presented. The geometric and electrical design rules necessary to design logic using this process technology, were also presented. Finally, the basic circuits required to implement the basic modules from Chapter II were presented. We are now ready to consider the actual circuit designs and layouts used to implement the modules.

IV. DESIGN AND LAYOUT OF THE MODULES OF THE PACKET ROUTER

The fundamental ideas that precede the circuit design and layout of the various router modules have been presented in earlier chapters. A functional decomposition down to the modular level was presented in Chapter II. Furthermore, details of the process technology were presented, along with basic geometric and electrical design rules in Chapter III. Also, the circuit designs for the basic components required for our implementation of the packet router have been presented. All that now remains is to present the module designs and layouts, along with the systems that were designed with them.

We first need to consider the modules, their functions and their implementations. Each module will be developed individually from the functional descriptions given in Chapter II to a circuit-level description. The circuit designs will then be used to develop a layout for each of the modules. In some cases, it will prove advantageous to develop more than one circuit design and layout for the module in question. Additional layouts will generally be developed to give some flexibility to the system designer using the modules. Where applicable, a circuit and module design will be developed in a form to allow the function of the module to be extended to an arbitrary number of bits. For some of these modules, a separate circuit design and layout will developed whose function can be applied only to one bit. The single bit implementation will generally be faster and less space-consuming than the multi-bit implementation when applied to one bit. In other cases, only a multi-bit implementation, or on the other hand, a single bit implementation will prove necessary.

As the C-element and arbiter circuit designs have already been presented, and since they are the basic components of the other modules, their layouts will be presented first. The Packet Detector and the REG module will be presented next. The SWITCH and MUX modules will then be developed. Following that, the circuits and layouts for modules to convert signals from single-wire, Ready/Acknowledge signalling to the dual-rail signalling, and modules to convert from dual-rail signalling to single-rail Ready/Acknowledge signalling will be developed. The dual-rail versions of the

AND and OR gates will then be presented as modules. Finally, the MERGE module will be considered.

The C-element is used in almost every module developed in this thesis. Its circuit design is fairly simple as was demonstrated in Chapter 3 (Fig. 3.6a). As it is a basic element in other designs, a minimal-size layout is desired to minimize the size of the layouts of modules using the C-element. A layout using minimum-size transistors is given is Fig. 4.1a. There are cases that require C-elements with more that two inputs. Specifically, the design of the CM and the OM presented in Chapter II requires three-input C-elements for combining Acknowledge signals for logic gates whose outputs fan out three ways. There are two possible means of implementing a three-input C-element. The easiest way is to combine two two-input C-elements [4] as shown in Fig. 2.18b. The more time- and area-efficient implementation expands the circuit design for the two-input C-element to three elements, by modifying the input logic gate as shown in Fig. 4.1b. A layout was developed using this circuit design as shown in Fig. 4.1c.

The arbiter is also an important module in the OM, not because of the number of times it is used, but because of the importance of its function. The task of arbitration is a fundamental sub-task of the packet router. A circuit design for the arbiter was presented in Chapter III. Unfortunately, the circuit design given by Seitz [4], as shown in Fig. 3.6b, does not completely specify a circuit design for the arbiter. The threshold element that follows the cross-coupled NOR gates is not a simple ratioed logic gate that follows the electrical design rules presented in Chapter III. The cross-coupled NOR gates are standard logic circuits, and will, therefore, act like NOR gates if designed with a standard 4:1 ratio. The threshold gate, however, will not begin to turn on until there is a difference in the voltage levels at the outputs of the cross-coupled NOR gates of a least one MOS threshold. Furthermore, the pull-down transistors of the NOR gates has to sink the current flowing through the corresponding leg of the threshold element in addition to the current flowing through its own pull-up transistor. The threshold element in combination with the corresponding NOR gate pull-down transistors can be treated as a

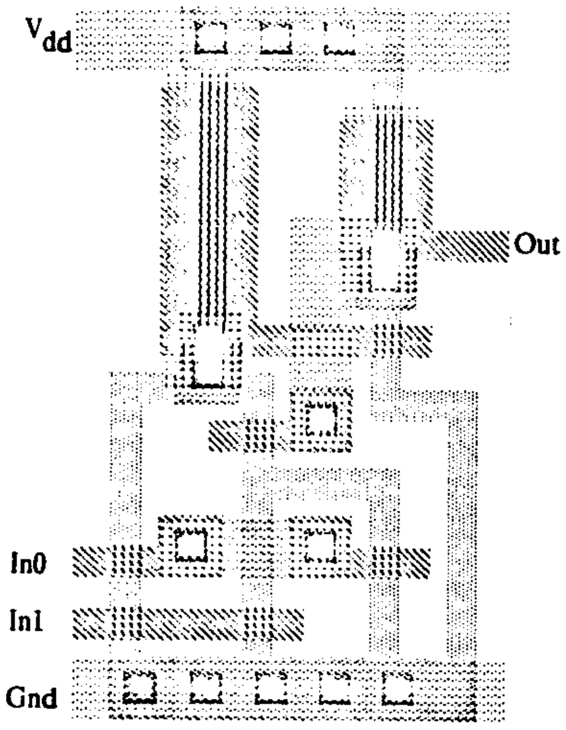


Figure 4.1a Two-Input C-Element Layout

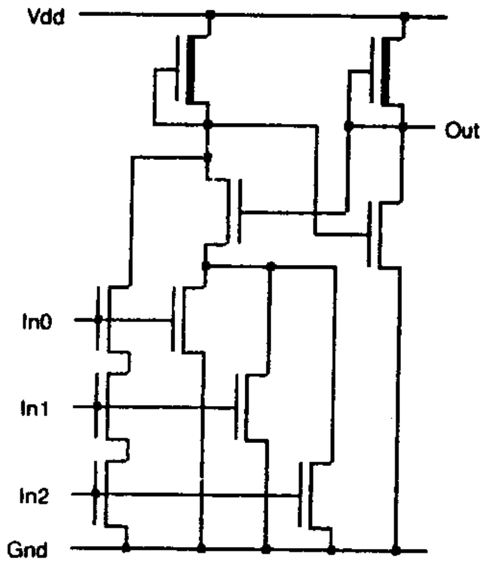


Figure 4.1b Three-Input C-Element

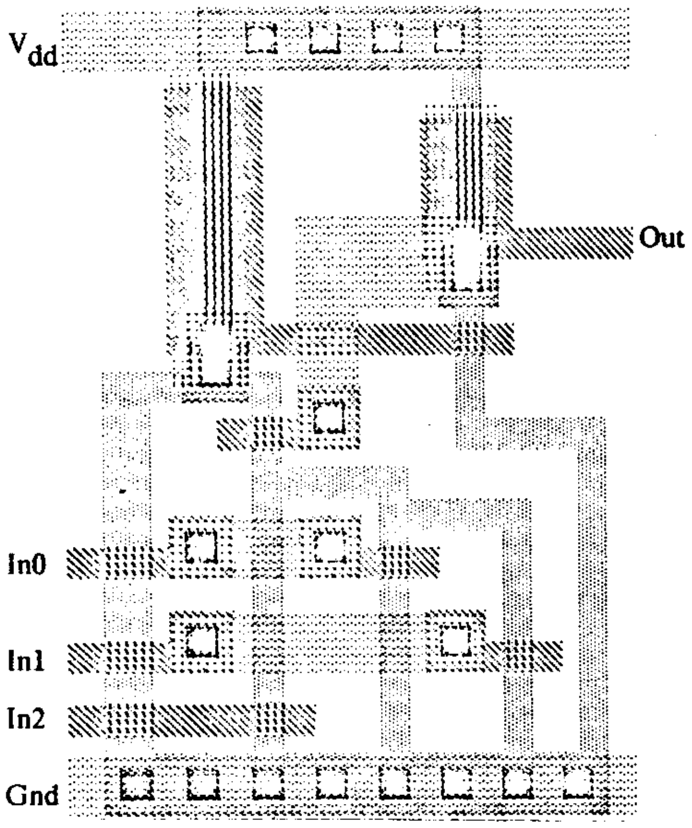


Figure 4.1c Three-Input C-Element Layout

standard NAND gate, so its pull-up transistor should have 1/8th the saturation current of each of its pull-down transistors. If the threshold pull-down transistor and the NOR gate pull-down transistors are minimum size (two λ by two λ), then the threshold pull-up transistor will be 8 times as long (two λ by sixteen λ). The pull-up transistor for the NOR gate would normally have 4 times the length of the pull-down transistors associated with it, but in this case, the pull-down transistors have to sink the saturation current of the threshold pull-up transistors, which is half of what they normally have to sink. The NOR gate pull-up transistors, then should be made 8 times the length of its pull-down transistors. A layout using these ratios is shown in Figure 4.2. It should be noted that the assumptions made for this design are extremely conservative, and a more complete analysis would probably yield a faster and more compact design. The fact that ratioed logic rules are based more on engineering approximations and experience than on solid theory makes it difficult to design non-standard logic elements like the arbiter. This is an area, however, in which further research would be of great benefit.

The REG module is one of the most important modules, as it is required to store the state variables required to implement the packet router, as well as to allow pipelining in the data paths of the packet router. We recall from our functional description of the REG module that it contains a packet detector for generating an Acknowledge signal. To help design the REG module in the most compact and

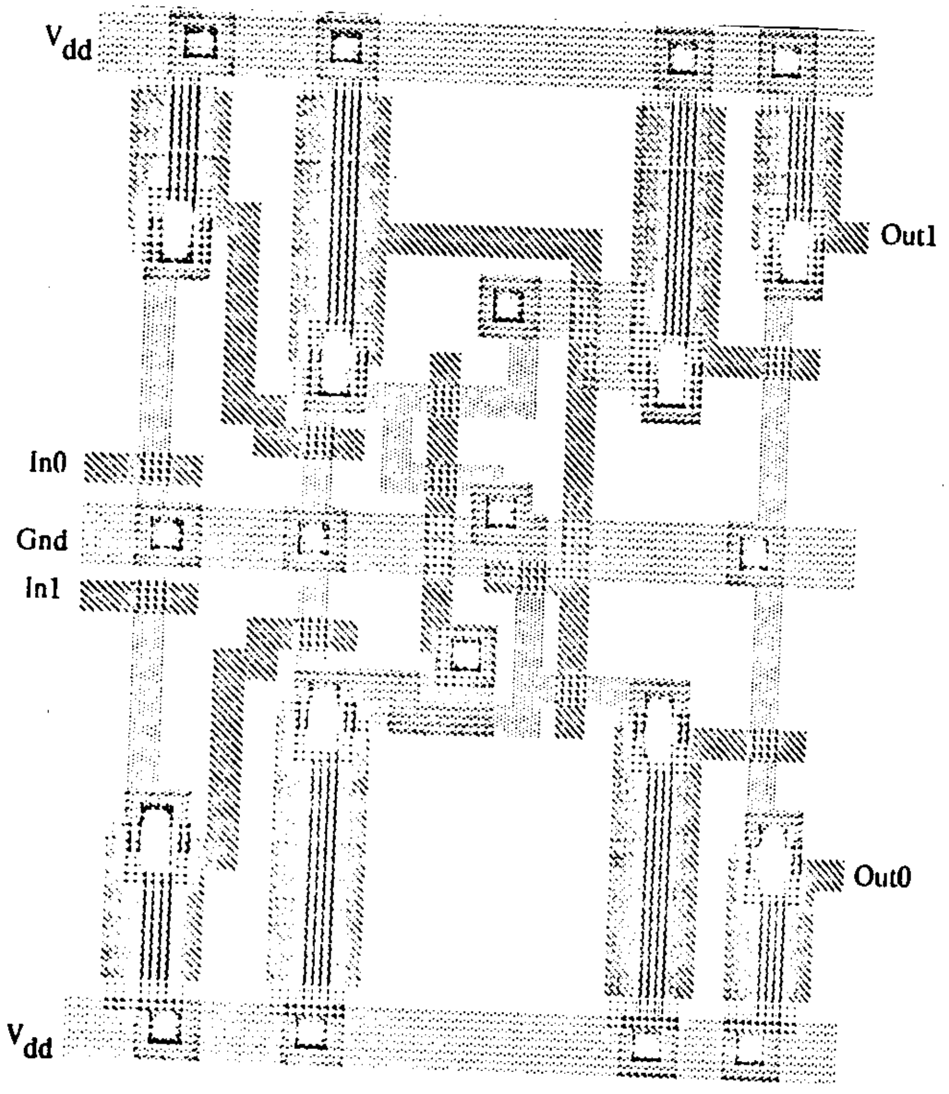


Figure 4.2 Arbiter Layout

efficient form, we will first develop a circuit design and layout for the packet detector, as a module in itself. We recall that the packet detector has some value of its own, as the SINK module specified in Chapter II, so for this purpose, we will develop a circuit design and layout for a packet detector for an arbitrary number of data bits.

An implementation of the SINK module for an arbitrary number of bits, can be divided into two separate parts. One part of the module will be repeated for each bit of input, and will be referred to as the bit cell. The other part is used once in each SINK module, and combines the outputs of the bit cell portions to give a single output. The second part will be referred to as the output cell. We recall from Chapter II, that a packet detector can be implemented as a two-input C-element with one input the OR of all the dual-rail input wires, and the other input the AND of the XORs of the wire pairs comprising all of the dual-rail bits. Our division between a bit cell and a output cell can be easily chosen. The XOR gates, and parts of the OR and AND gates should be implemented in the cell circuit, and the C-element and remaining parts of the AND and OR gates will be contained in the output circuit. The design constraints of ratioed-logic force us to use NOR gates to implement the OR over 2N bits, and the AND over N bits, for an N-bit variable, that are needed to implement the SINK module. The OR of all of the dual-rail wires is implemented simply by a NOR of all the bits, followed by an inverter. The NOR pull-down transistors will be included in the bit-cell portion of the layout, while the NOR pull-up transistor will be in the output cell layout. To implement an AND, we can use DeMorgan's theorem, by putting inverters before a NOR gate to give an AND gate. The inverters can be merged with the XORs, as we shall see. The easiest way to implement a XOR uses two pass transistors, and two inverters, as shown in Fig. 4.3a. If we exchange the gate connections to the two pass-transistors in the XOR circuit, as demonstrated in Fig. 4.3b, we can achieve an XOR with an inverted output, or an "If and Only If" logic gate. If we combine this with the pull-down transistor for the NOR gate to implement the AND, and the pull-down transistors for the OR structure, we get the circuit and bit cell layout shown in Fig. 4.4. The pull-ups for the OR structure, and the AND structure can be combined with an inverter for the OR structure and a C-element to give an output circuit and cell layout, as shown in Fig. 4.5. The pull-up for the AND structure was given a 1/8 W/L ratio, to give threshold restoration to compensate for the pass transistors in the bit cell. The SINK module is laid out by stacking bit cells, so that the wires combining the two NOR structures are linked together throughout the module. The output cell is placed on the top of the stack.

We recall from Chapter II that a REG module is simply a group of 2N C-elements, for an N-bit variable, whose inputs consist of the inputs to the REG cell, and the complement of the Acknowledge signal from the next stage, along with a packet detector to generate the Acknowledge signal for the previous stage. The circuit is simply an extension of the packet detector with C-elements added for the storage element. The REG module used in the packet router requires a mechanism for initialization, as the feedback paths in the CM and OM require initial values for the packet router to function properly.

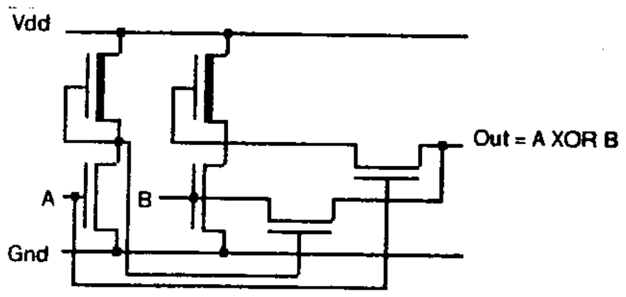


Figure 4.3a Exclusive-OR Gate

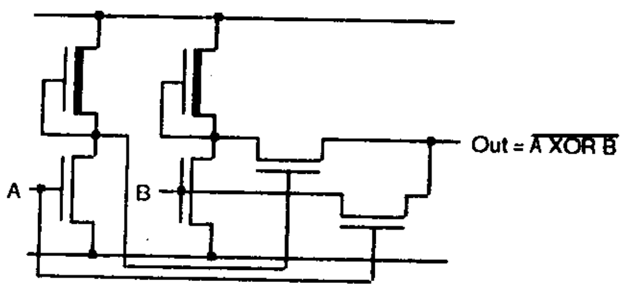


Figure 4.3b "If-And-Only-If" Gate

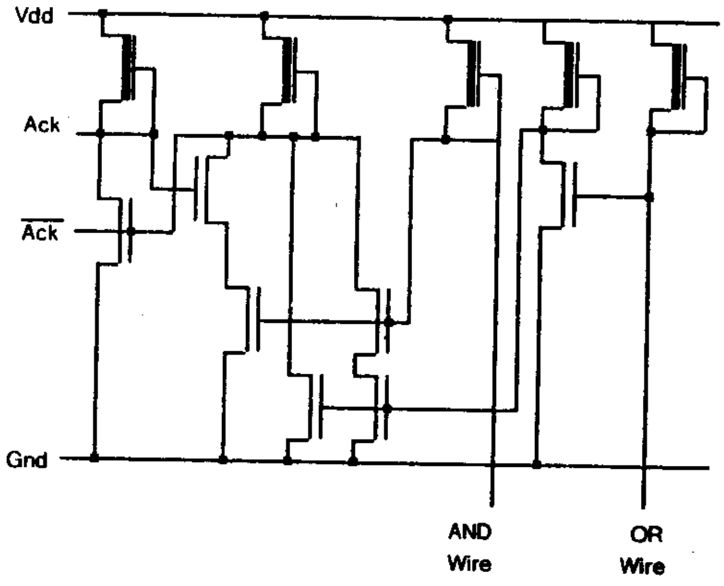


Figure 4.4a SINK Module Output Cell Circuit

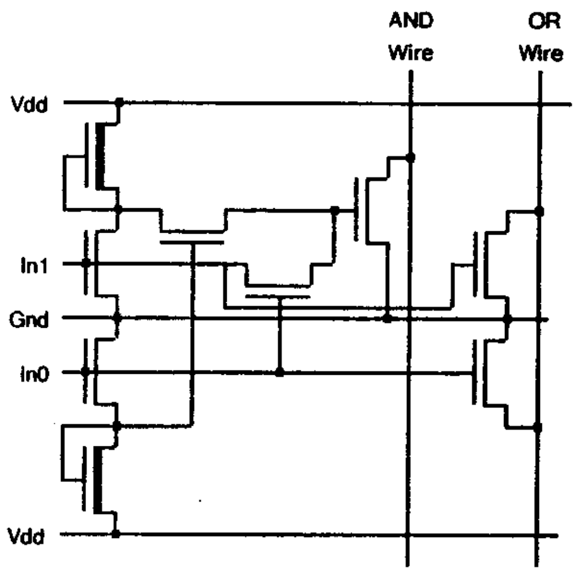


Figure 4.4b Sink Module Bit Cell

Also, the behavior of a string of REG modules that aren't initialized to have spacer states following and preceding every data state, and spacers everywhere else, is undependable, as was shown in Chapter 2. An extra transistor has been inserted into the C-element to accomplish initialization. This pull-down

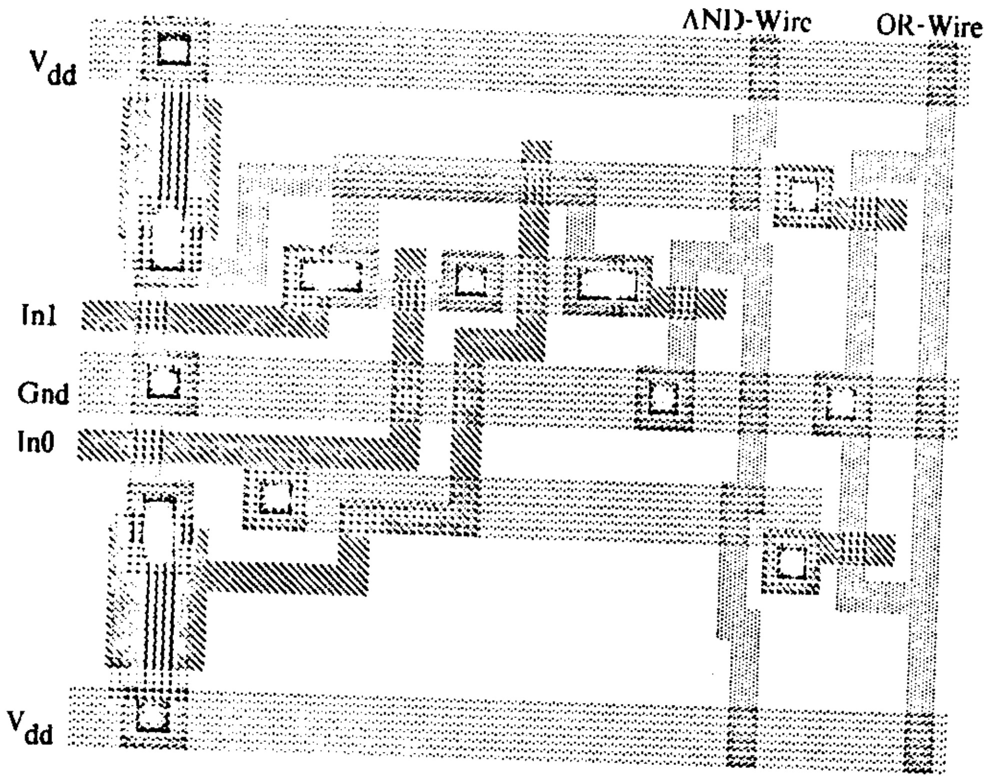


Figure 4.5a SINK Module Bit-Cell Layout

transistor can pull either the output node or the feedback node of its C-element to ground, and thereby initialize any output wire of the REG module to a zero or a one. This feature is programmed by the designer by inserting a piece of poly-silicon between the drain of the initialization transistor and the appropriate gate. If spacer states are alternated with data states, any group of REG modules can be initialized by asserting a wire attached to all of the gates of the initialization transistors in the group. The REG module, like the packet detector, can be divided into a bit cell and an output cell. The bit cell will consist of the two C-elements associated with each bit of a REG module, in addition to the logic for the bit cell of the packet detector, and an initialization transistor for each. The output cell will consist of the output cell of packet detector, and a super-buffer inverter to invert the acknowledge signal from the next stage. The circuit diagrams and the layouts for the two parts of the REG module are shown in Fig. 4.6, and Fig. 4.7. The possible connections for initialization are demonstrated in Figure 4.8. Figure 4.8a

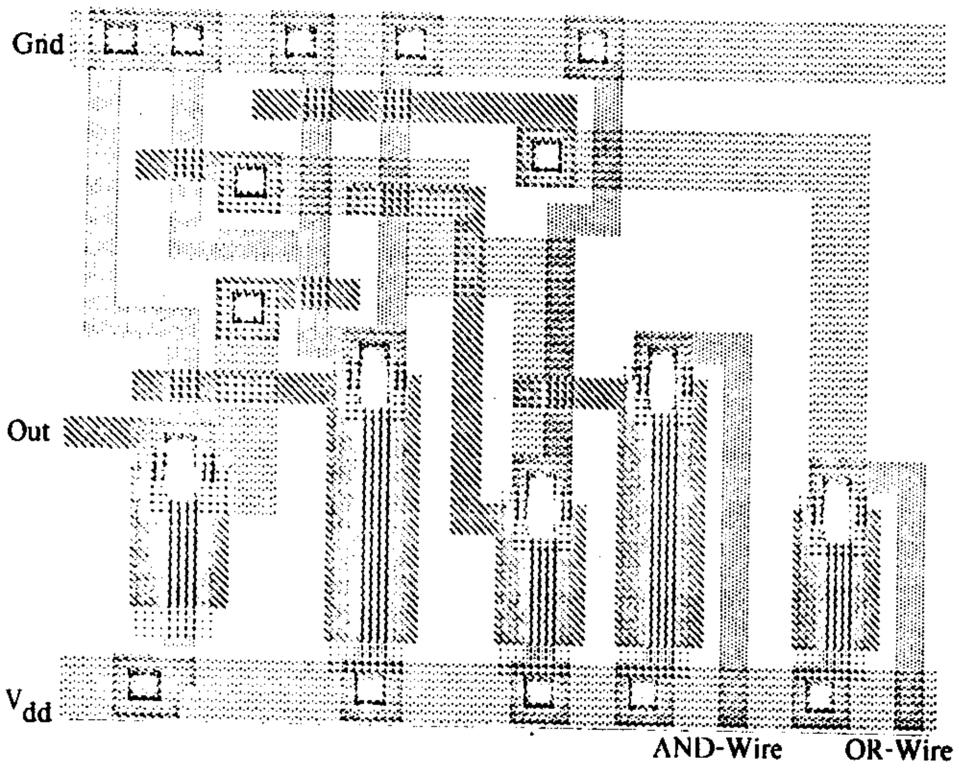


Figure 4.5b SINK Module Acknowledge Cell Layout

shows the connection for a REG cell initialized for a spacer state. Figure 4.8b shows the REG cell that will be initialized in a ZERO state. The REG cell in Figure 4.8c will be initialized in a ONE state. The REG module is assembled by stacking bit cells, so that the wires for the packet detector connect together. The output cell sits at the top of the module.

As mentioned earlier, it sometimes is desirable to have a separate layout and circuit design for the one-bit implementation of a module when it is used a lot, and the reduction in layout area is significant. This happens to be the case for the REG module. All of the REG modules used for feedback in the CM and OM store single bit variables. The C-element of the packet detector is not required if the packet is only one bit wide. Thus, a simple OR gate is sufficient as a packet detector for the one bit case. The layouts of the multi-bit REG module have been repeated with a NOR gate and an inverter replacing the

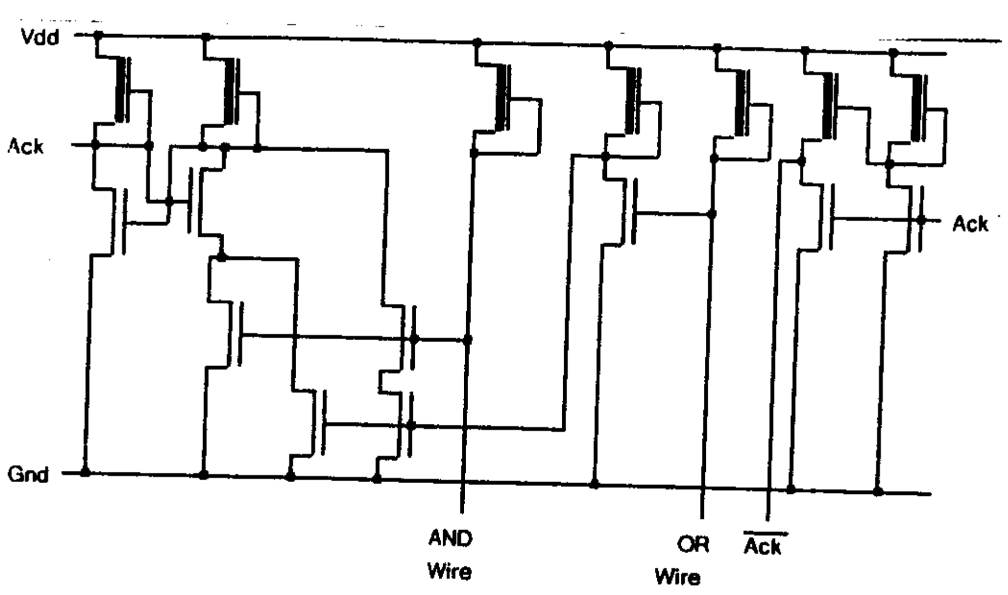


Figure 4.6a REG Module Output Cell Circuit

packet detector. The layout for the one-bit REG module are shown in Fig. 4.9. The initialization scheme is identical to that used in the multi-bit implementation, as discussed earlier, and are demonstrated in Figure 4.9. It can be seen that the layout for the single-bit implementation is significantly smaller than the multi-bit implementation applied in a single-bit application. Furthermore, its is reasonable to assume that the simpler logic will yield better speed performance.

The SWITCH and MUX modules can be implemented directly from their block descriptions presented in Chapter II (Figures 2.11 and 2.12). The OR gates required for combining the data signals in the MUX and for combining the Acknowledge signals in the SWITCH are implemented using the inverted C-element outputs, and a NAND gate. Like the modules we have already designed, we want an N-bit implementation, with bit cell layouts and output cell layouts like before. The SWITCH and MUX functions work on data or signals, so the basic bit cell will only perform its function on one wire, and it can be used 2N times for an N-bit variable. As these modules are mostly used for large variables, a single-bit layout will not be designed.

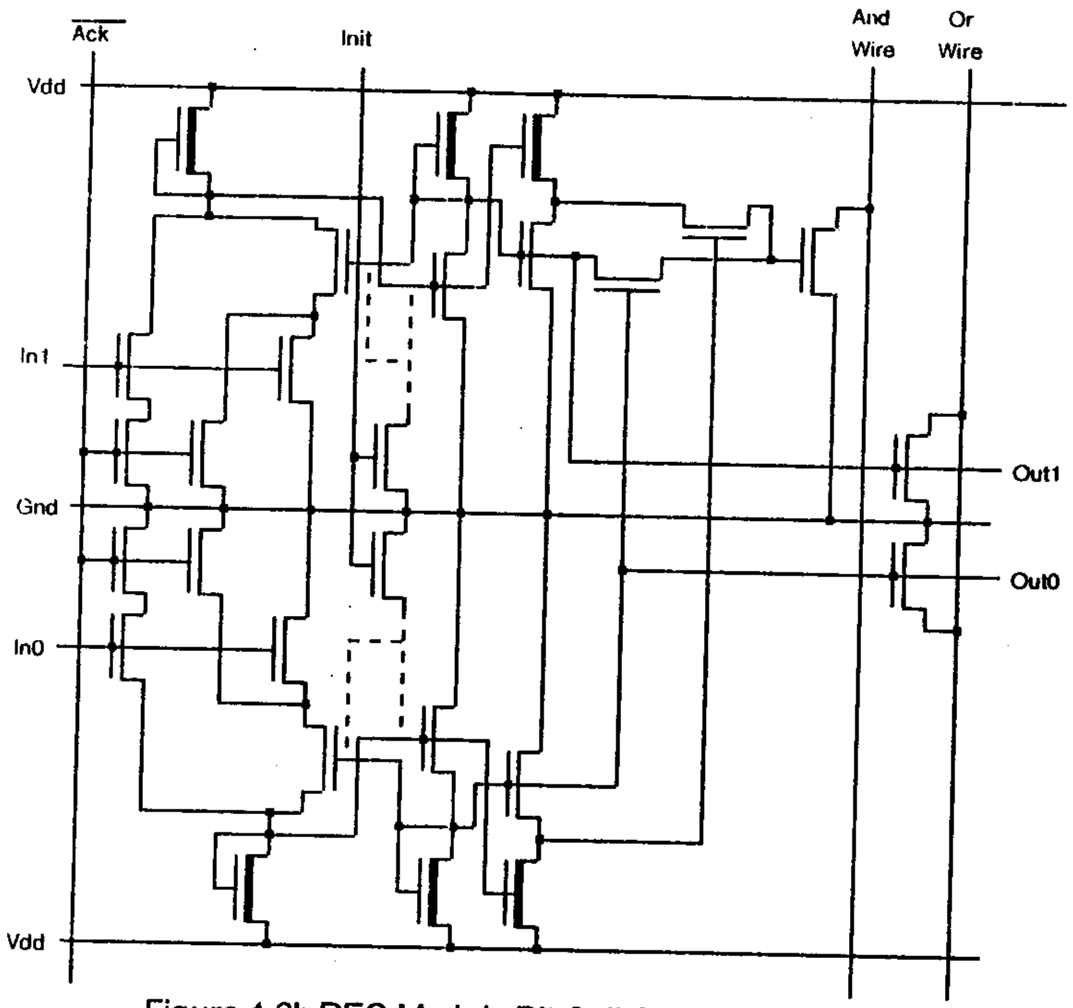


Figure 4.6b REG Module Bit Cell Circuit

First, lets consider the SWITCH module. The bit cell need only consist of two C-elements with the input wire common to both. Two wires run vertically, and determine the switching direction. They are driven by the output cell, and are therefore common to all the bit cells in a SWITCH module. The layout for the bit cell is shown in Figure 4.10. The output cell of the SWITCH module takes the two Acknowledge lines from the modules following the SWITCH and select the one coming from the output direction of the SWITCH. The resulting Acknowledge is sent to the modules generating the inputs to the SWITCH. Both the data byte and the switching variable are acknowledged, so the Acknowledge signal

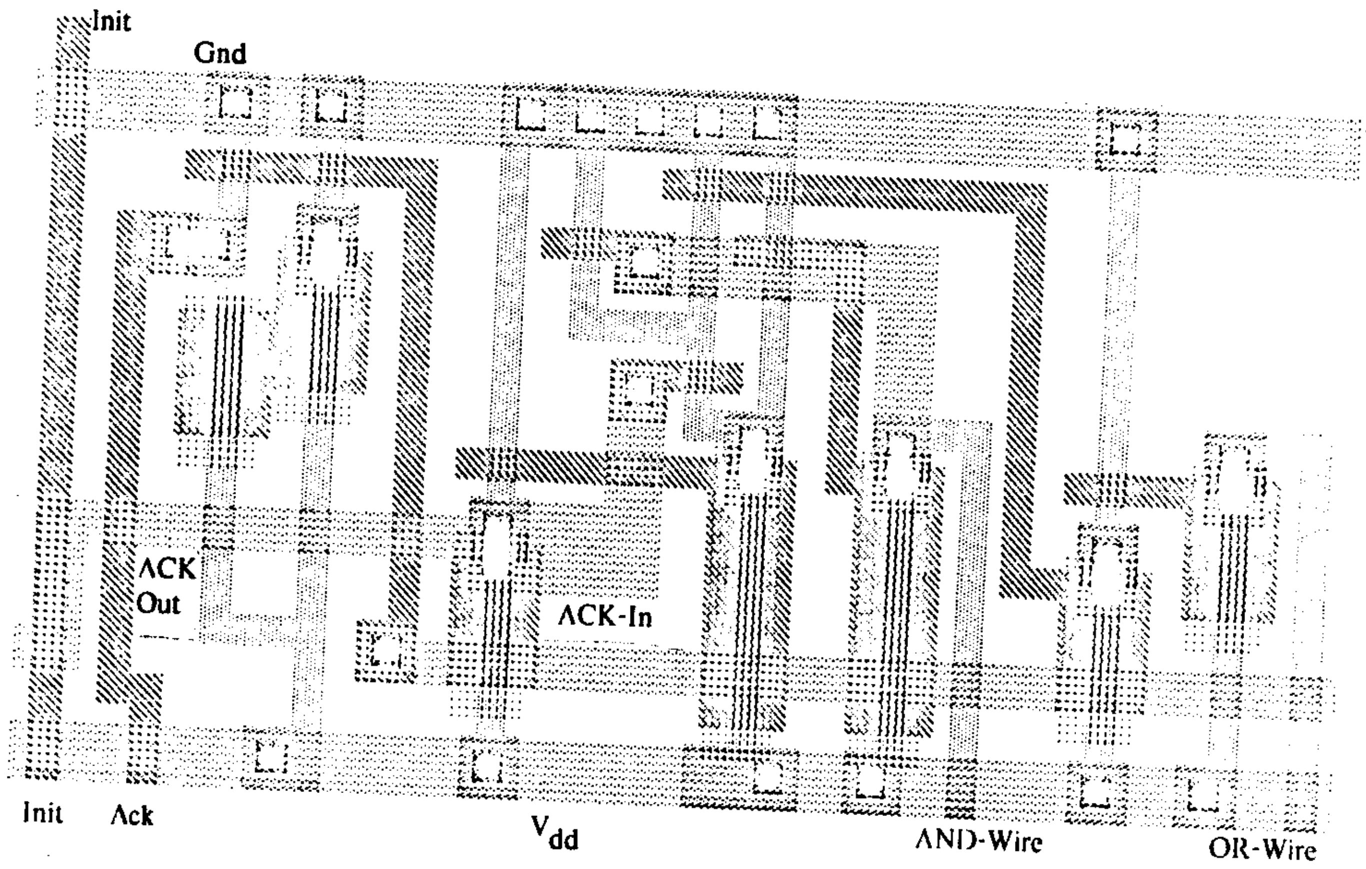


Figure 4.7a Extendable-REG Module Acknowledge-Cell Layout

fans out to both of the previous modules. The cell contains two C-elements which select the appropriate Acknowledge signal. The outputs of the two C-elements are ORed by NANDing the negative outputs of the C-elements, coming from the first stage of the C-elements. The use of the internal node as the output of the C-element in this module is not as dangerous as it would appear. The inverter contained in the C-element is only loaded by the feedback transistor, therefore the C-element should stabilize very quickly after the internal node, and there is very little danger of the output of two-input NAND gate stabilizing before the C-elements stabilize. The layout for the output cell is shown in Figure 4.11. The bit cells are stacked above the output cell, so that the direction wires link together throughout the module.

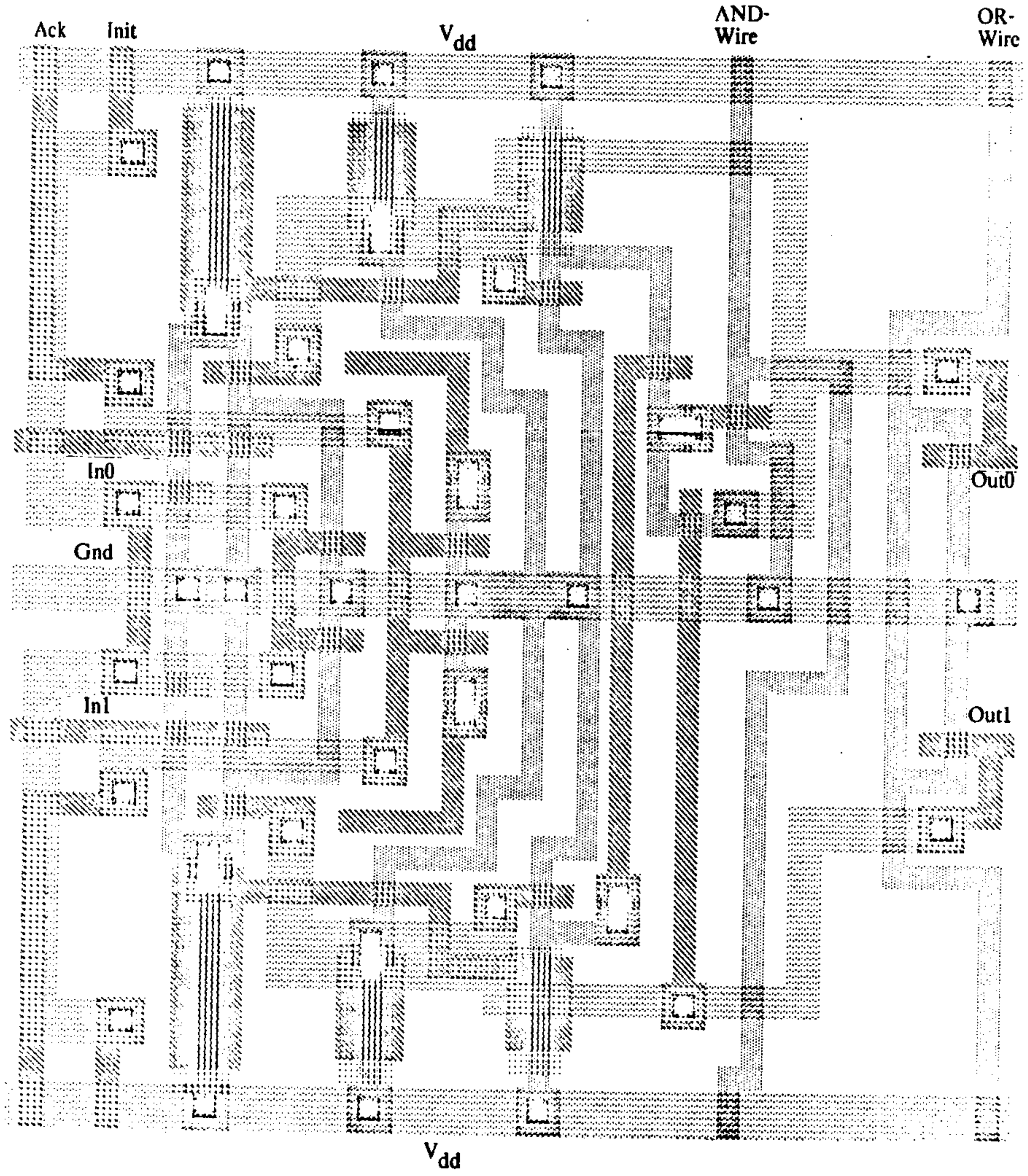


Figure 4.7b Extendable-REG Module Bit-Cell Layout

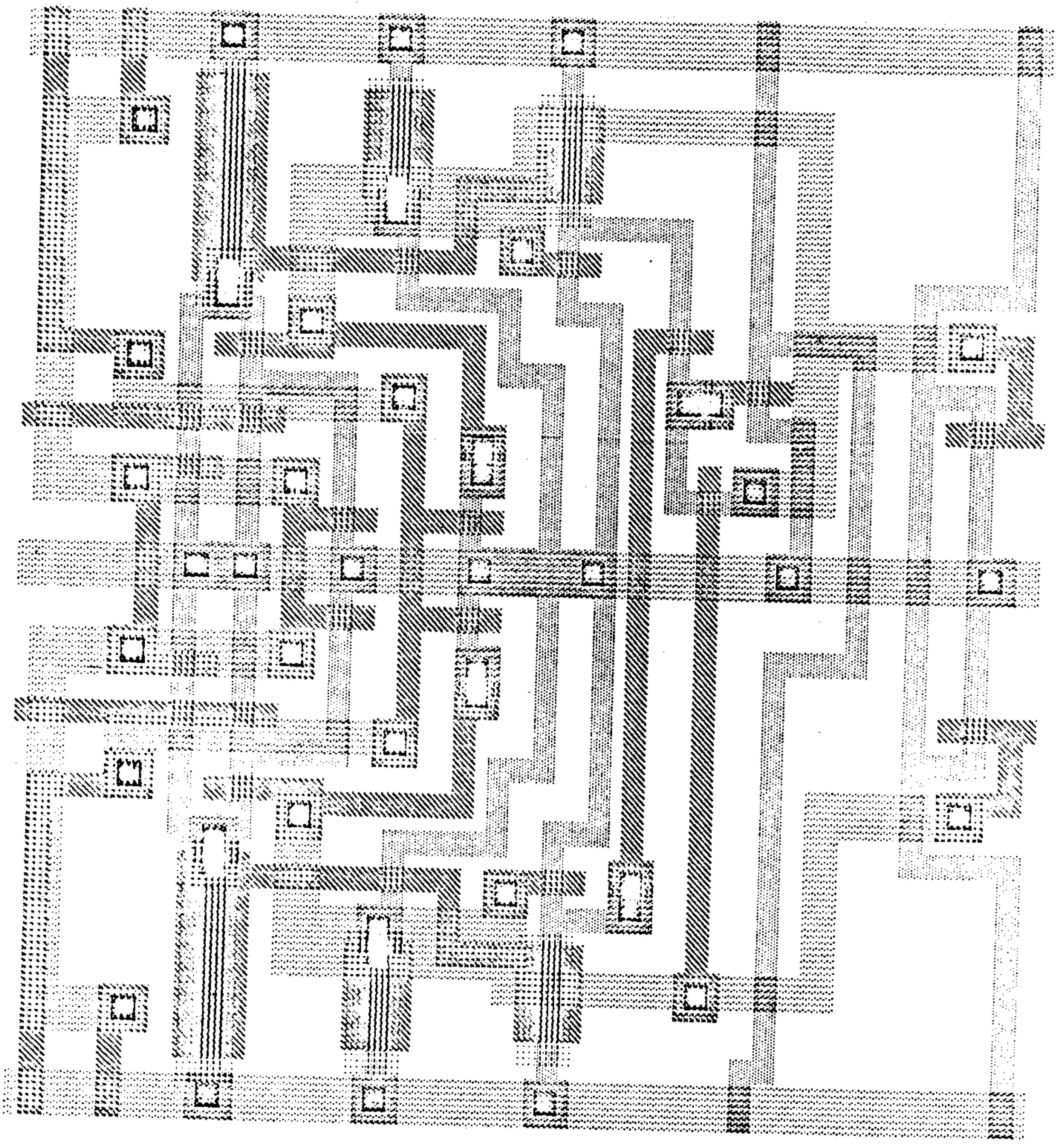


Figure 4.8a Extendable-REG Module Bit-Cell Initialized as SPACER

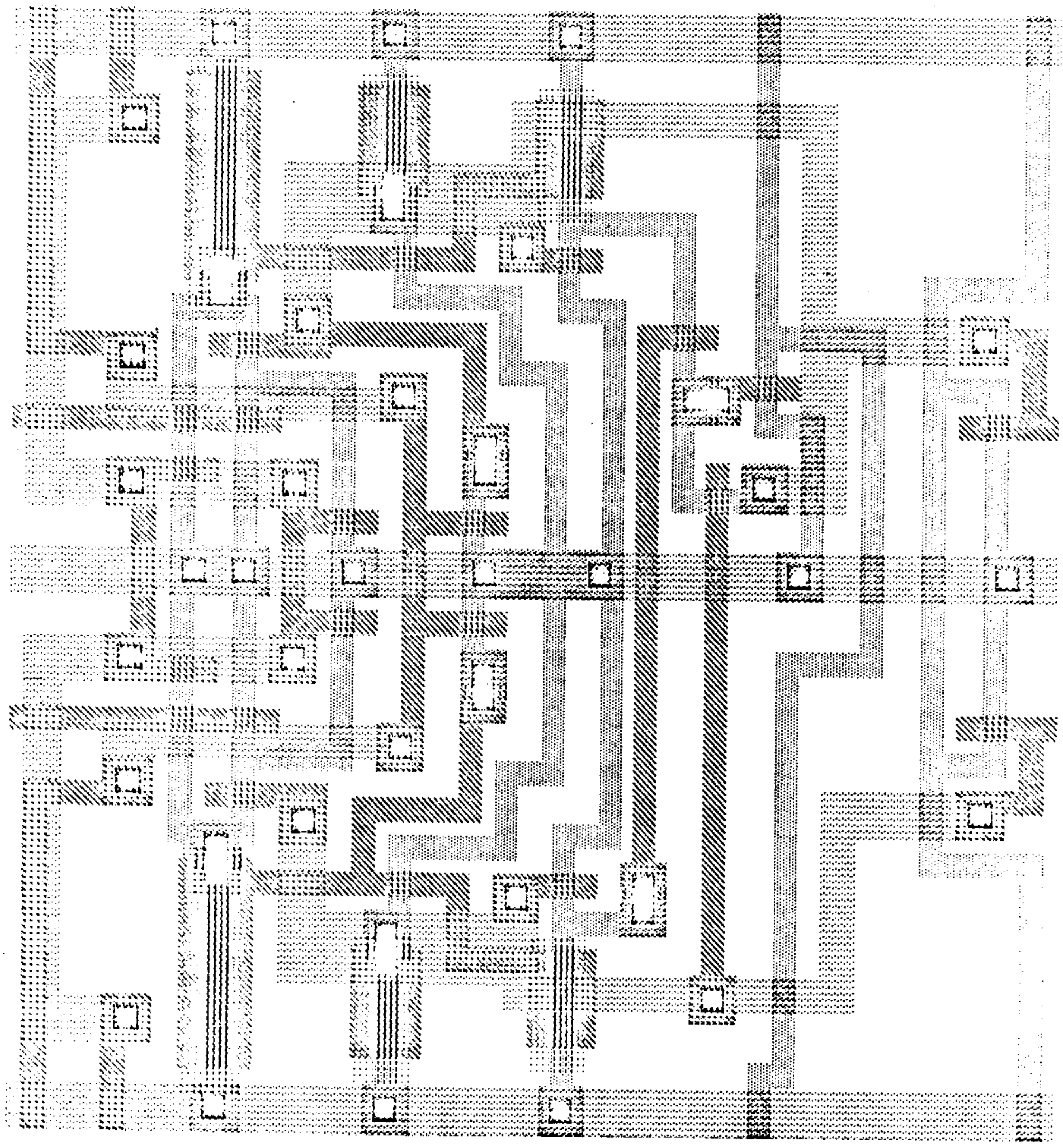


Figure 4.8b Extendable-REG Module Bit-Cell Initialized as ZERO

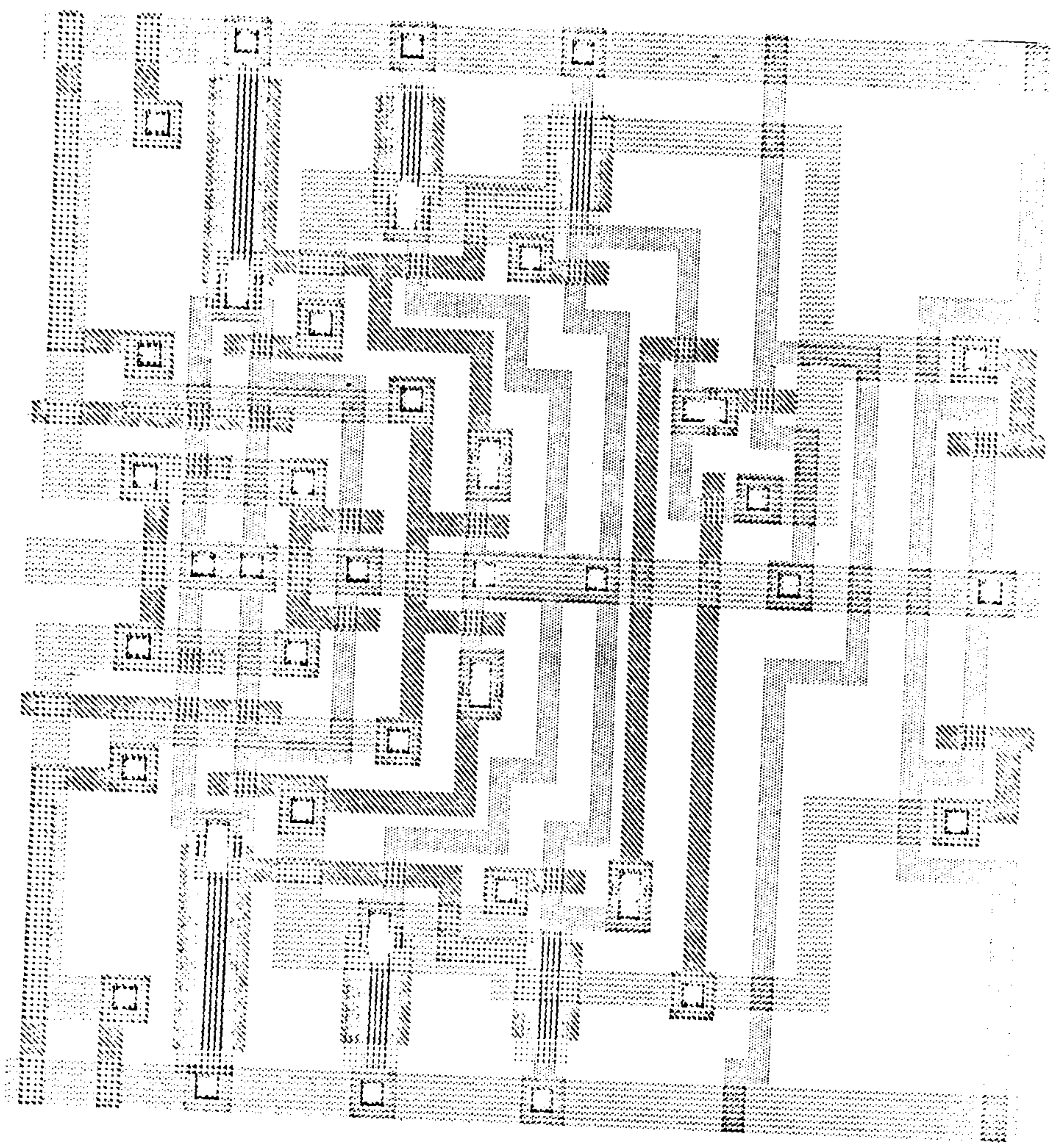


Figure 4.8c Extendable-REG Module Bit-Cell Initialized as ONE

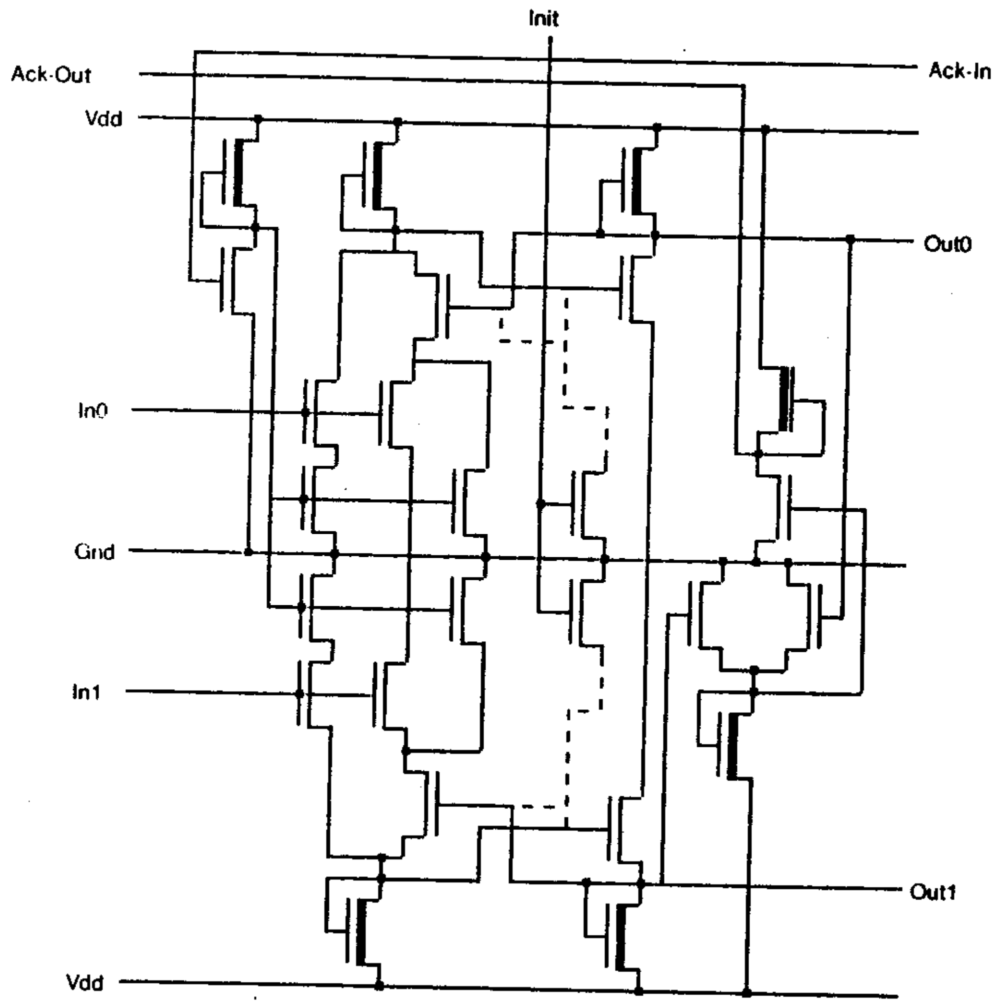


Figure 4.9a Single-Bit REG Module Circuit Diagram

The MUX module is practically the same as the SWITCH. The circuits are exactly the same, except the circuit for the bit cell of the MUX is the circuit for the output cell of the SWITCH. The two inputs get selected by two direction lines, using two C-elements, and the outputs of the C-elements are combined using a NAND gate to OR the logic values. The MUX bit cell is shown in Figure 4.12. Likewise, the MUX output cell uses the same circuit as the SWITCH bit cell. The MUX output cell switches the Acknowledge from the output to the input corresponding to the direction logic value. The Acknowledge input will be routed around the output cell to acknowledge the direction variable, since it

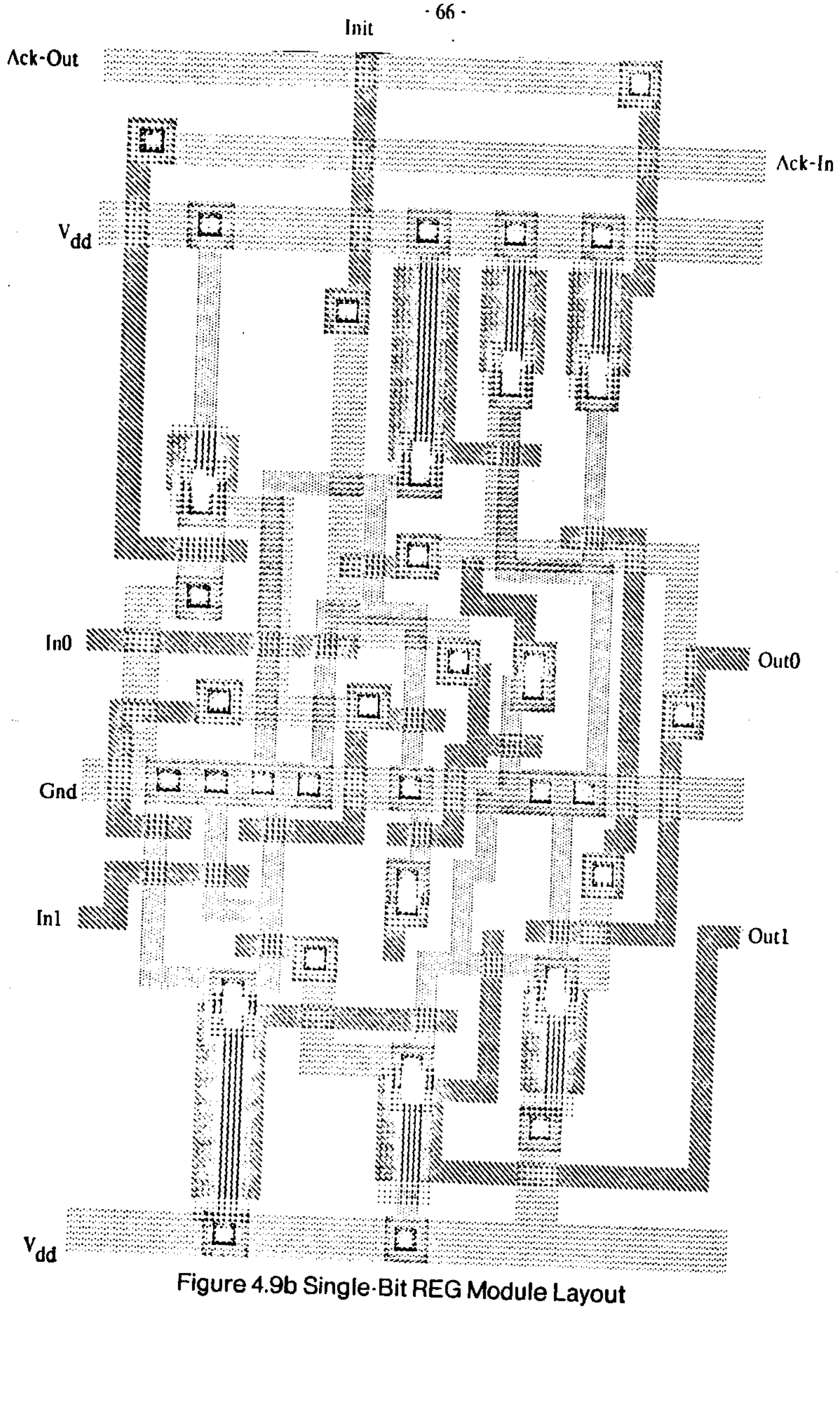


Figure 4.9b Single-Bit REG Module Layout

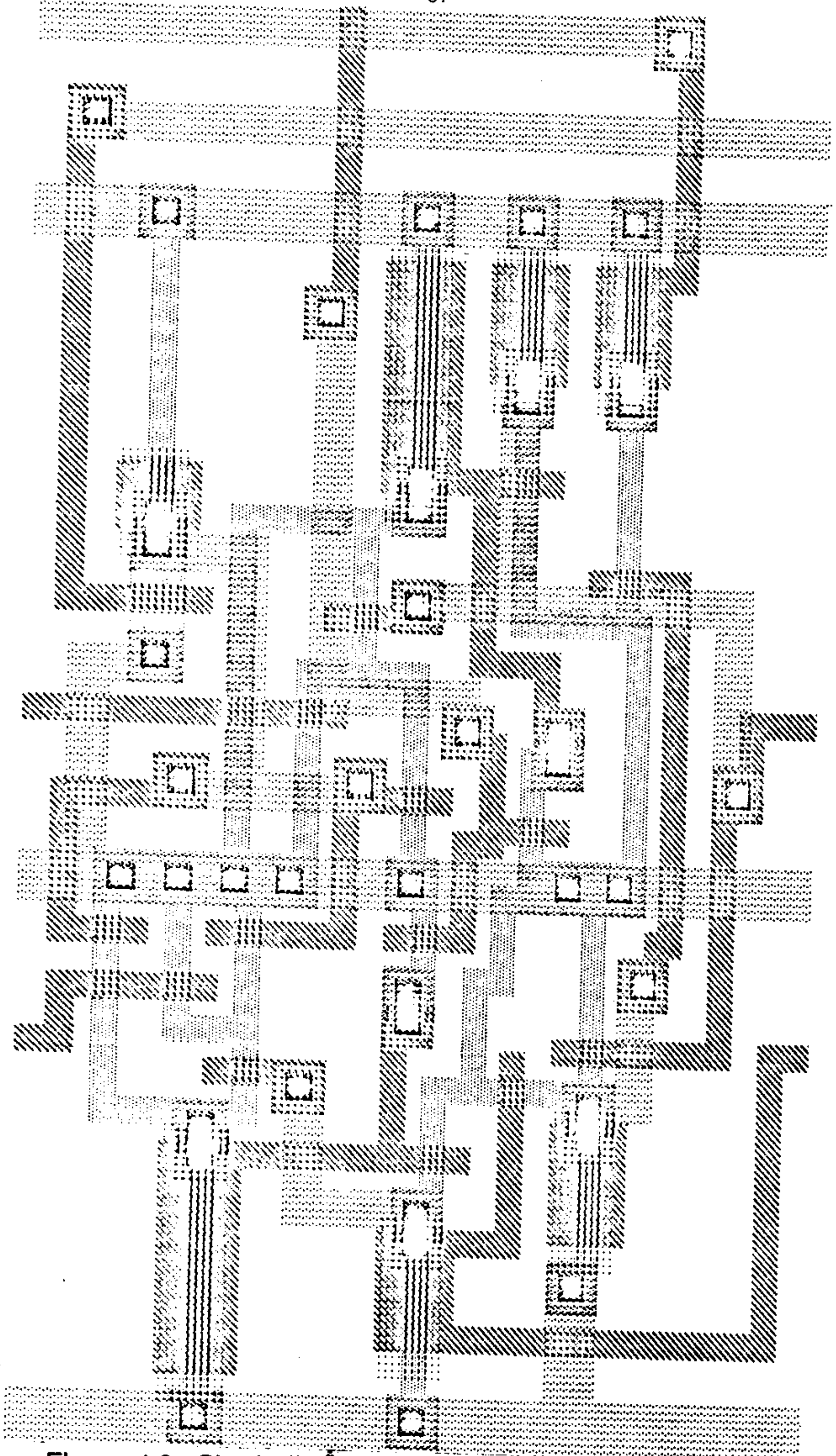


Figure 4.9c Single-Bit REG Module Initialized as SPACER

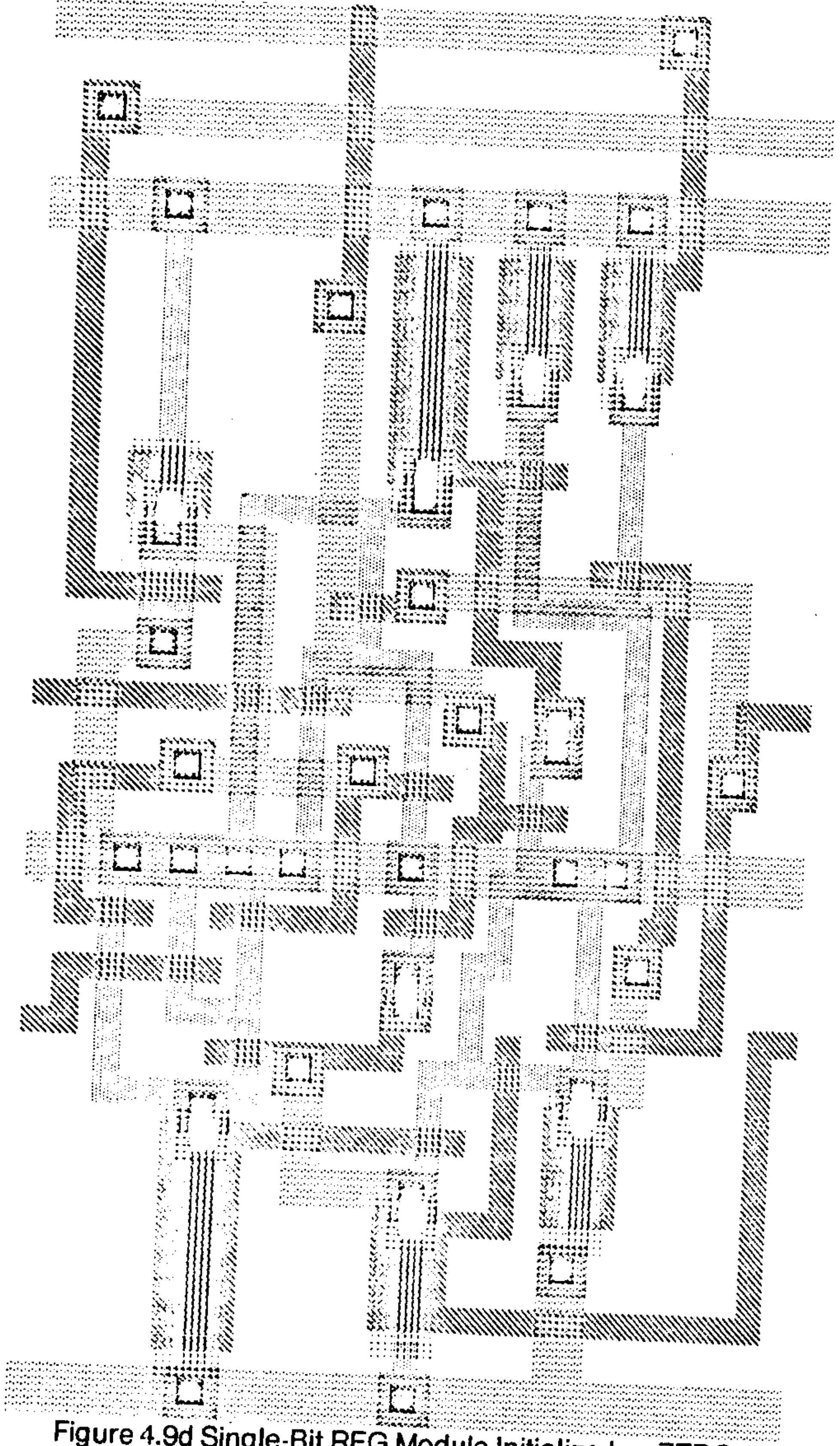


Figure 4.9d Single-Bit REG Module Initialized as ZERO

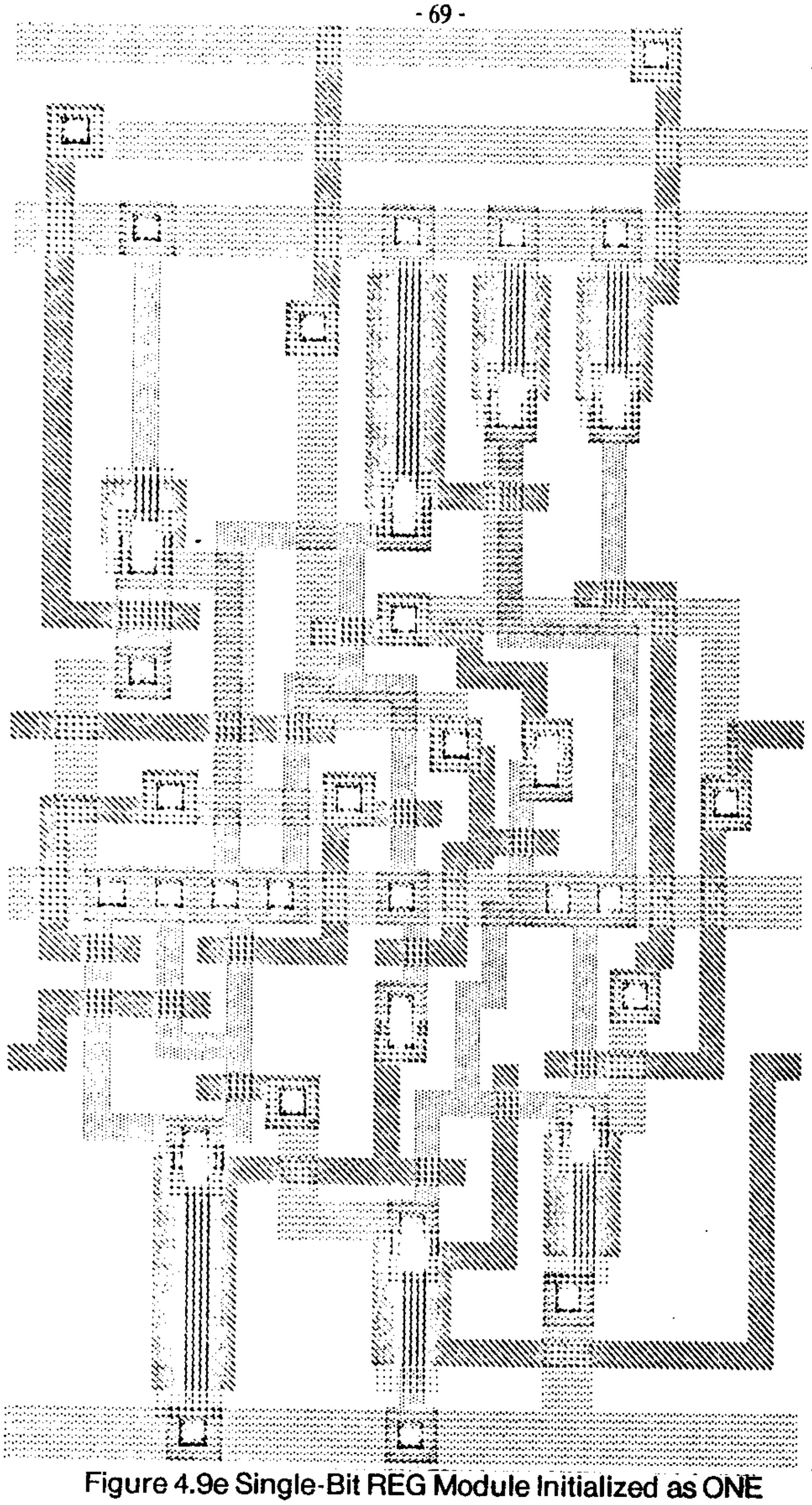


Figure 4.9e Single-Bit REG Module Initialized as ONE

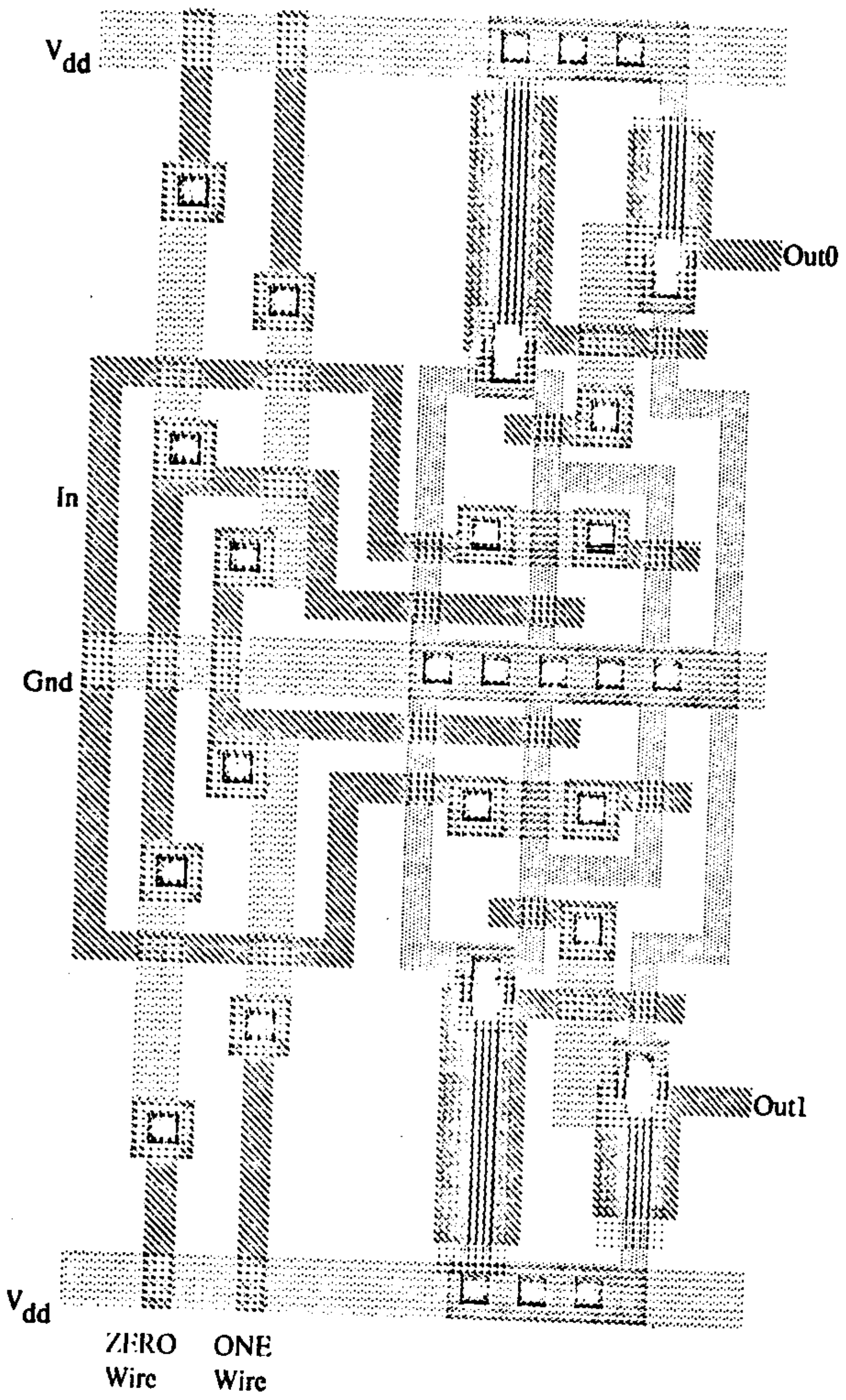


Figure 4.10 SWITCH Module Bit-Cell Layout

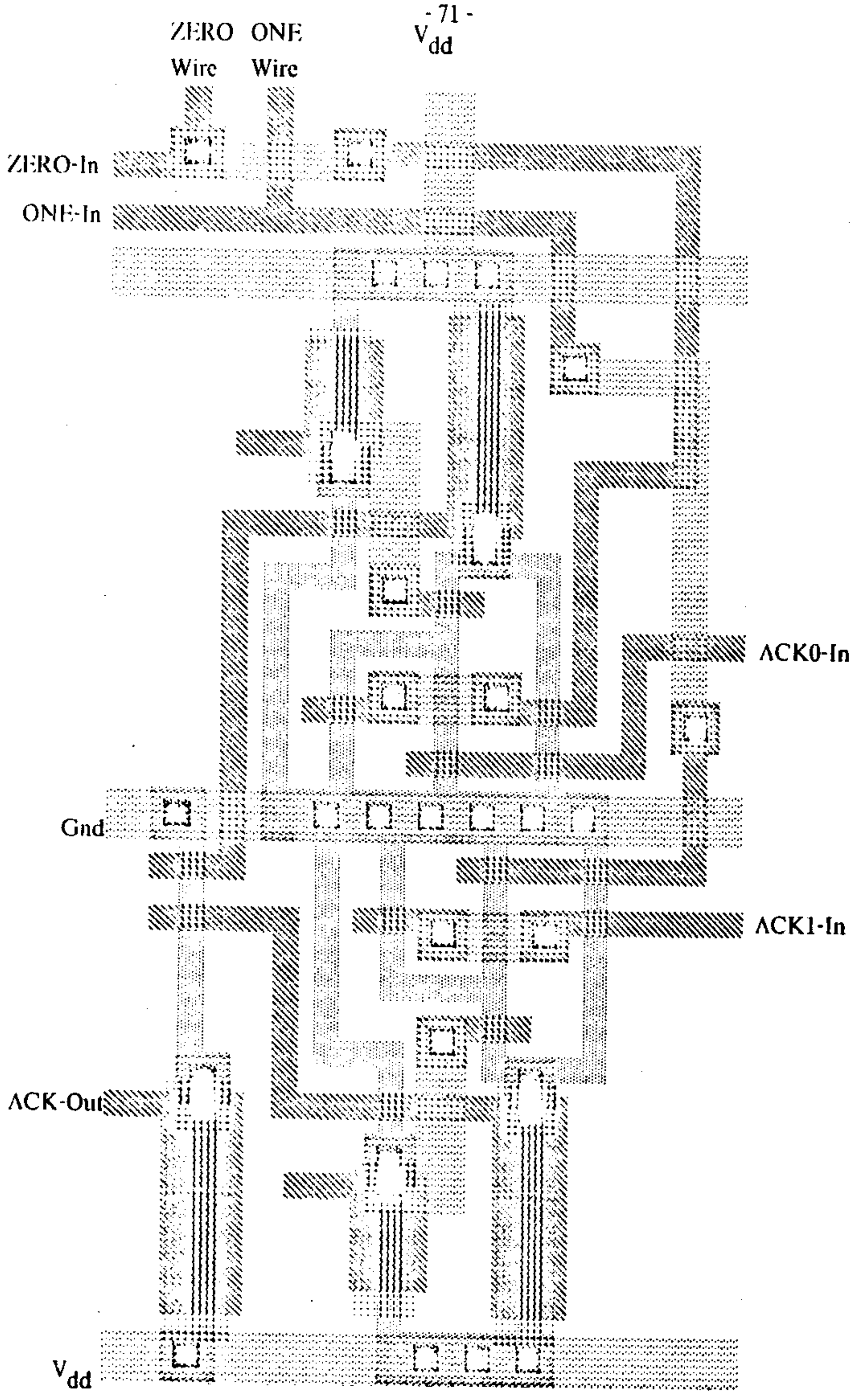


Figure 4.11 SWITCH Module Acknowledge-Cell Layout

must be acknowledged irrespective of its value (the direction). The output cell is simply two C-elements, and is shown in Figure 4.13. Just like the SWITCH, the MUX bit cells are stacked above the output cell, so that the direction variable passes throughout the module.

Next, we will consider modules designed to convert dual-rail signalling to single-rail Ready/Acknowledge signalling, and vice versa. These two tasks are a necessity for any integrated circuit implementation using the dual-rail modules discussed in this thesis, because of pin limitations. The number of pins an integrated circuit can have has a maximum value, because packaging technology has a considerable influence on cost, which means that integrated circuit manufacturers have to limit the complexity of their packaging technology. As dual-rail signalling requires almost twice as many wires as single-wire Ready/Acknowledge signalling, the latter is preferable for chip input/output. The circuitry required for conversion of one form of signalling to another is fairly simple. We considered the modules required to convert to and from dual-rail coding in Chapter 2. The circuit for a One-to-Two Converter was very simple. A cell-type layout for N bits will be developed here. It was noted that two approaches could be taken to converting dual-rail coding to single-rail coding. One scheme used a SINK module, which we have already designed, to generate the Acknowledge signal. The other scheme used a simplified version of the SINK module, designed by Seitz [4]. The area difference between the two does not really justify developing two different packet detectors, but, as discussed in Chapter 2, the differences between the two in terms of error detection are not entirely clear. The REG and SINK modules use the same kind of packet detection, while the Two-to-One converter uses a simpler method that could potentially reduce the area of these modules significantly. Laying out the Two-to-One Converter will help judge the area savings, and its implementation can be used to test the differences between the two packet detection schemes in terms of error-detection and speed.

The One-to-Two Converter is a very simple module (see Figure 2.16). The easiest MOS implementation uses two inverters and two 2-input NOR gates. The Ready signal is inverted, and is an

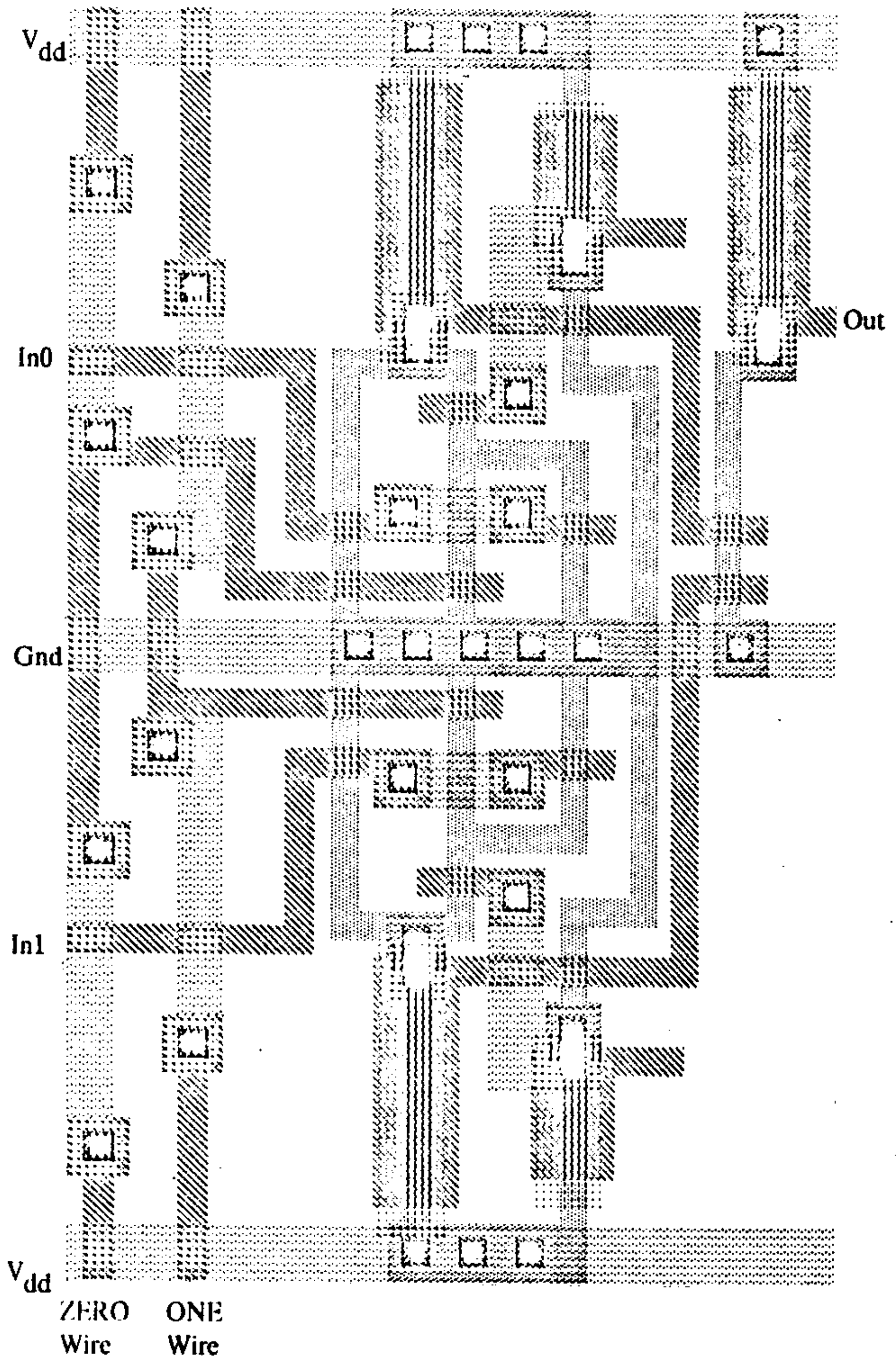


Figure 4.12 MUX Module Bit-Cell Layout

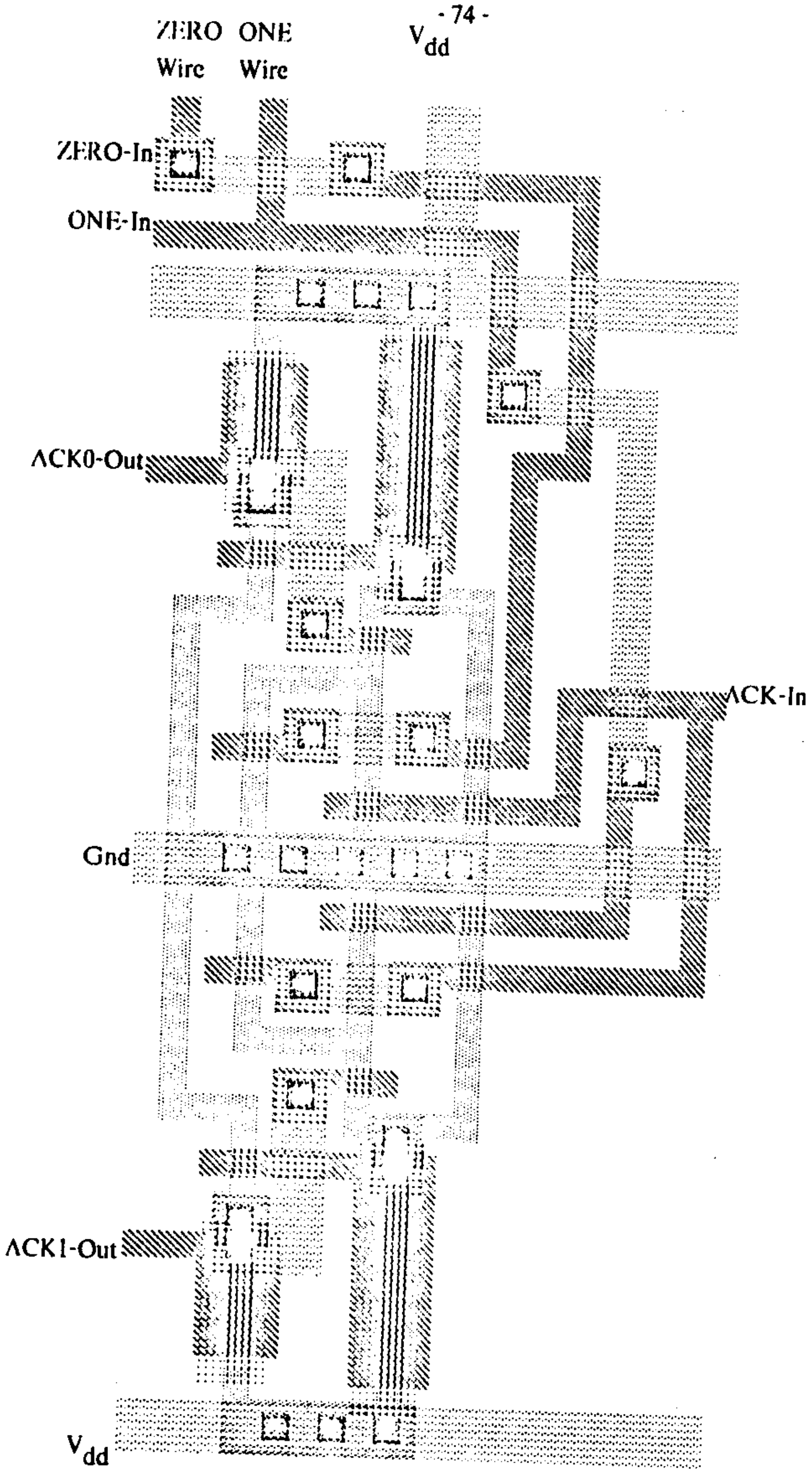


Figure 4.13 MUX Module Acknowledge Cell Layout

input to both NOR gates. The inverted input bit is the other input to the NOR which generates the ONE wire of the output bit. The input bit is the other input to the NOR which generates the ZERO wire of the output bit. The NOR gates can be considered equivalent to the AND gates of Figure 2.16 by applying DeMorgan's Theorem. This circuit design can easily be partitioned into a bit cell and an Ready cell. The bit cell contains the two NOR gates, and the inverter associated with the bit input. The Ready cell consists of the inverter for the Ready signal for each of the bit cells. The Ready cell will have to drive a large number of gates, so a super-buffer implementation is used. Furthermore, since we wish to output a spacer state when the system is being initialized (the signal Reset is high), we need to add some circuitry before the super-buffer. In this case, the simplest circuit used an inverter, a NOR gate and an inverting super-buffer. The circuit designs for the cells of the One-to-Two Converter are shown in Figure 4.14. The layouts are shown in Figure 4.15. Like all of the previous modules, the One-to-Two Converter cells are stacked to form the module, aligned so that the wires for the inverted Acknowledge signal runs throughout the module.

The Two-to-One Converter circuitry is slightly more complicated. The implementation discussed in Chapter 2 requires and N-input C-element to generate the Acknowledge signal. If wish to make an implementation that can be divided into cells like the modules we have designed up to this point, we have

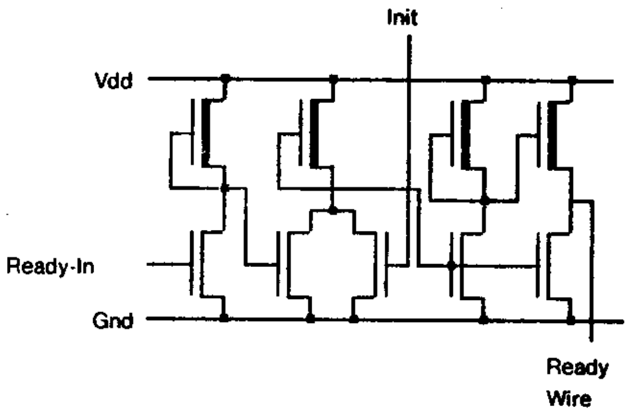


Figure 4.14a One-to-Two Converter Input Cell

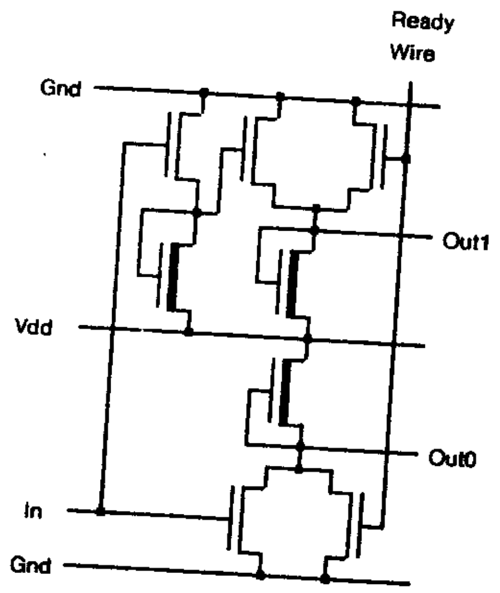


Figure 4.14b One-to-Two Converter Bit Cell

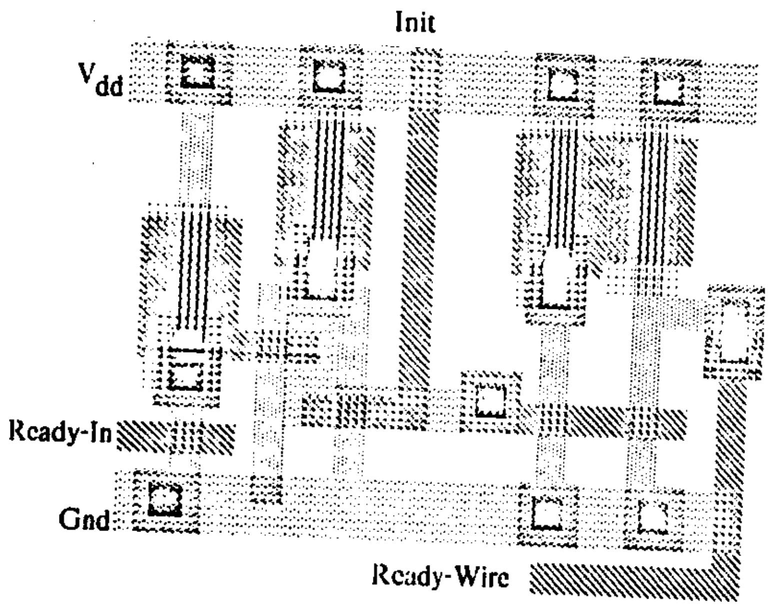


Figure 4.15a One-to-Two Converter Input Cell Layout

to modify the C-element designs we have already considered. An N-input cascading C-element could be designed using a three input C-element for each stage, but this approach will be very slow. If expand a

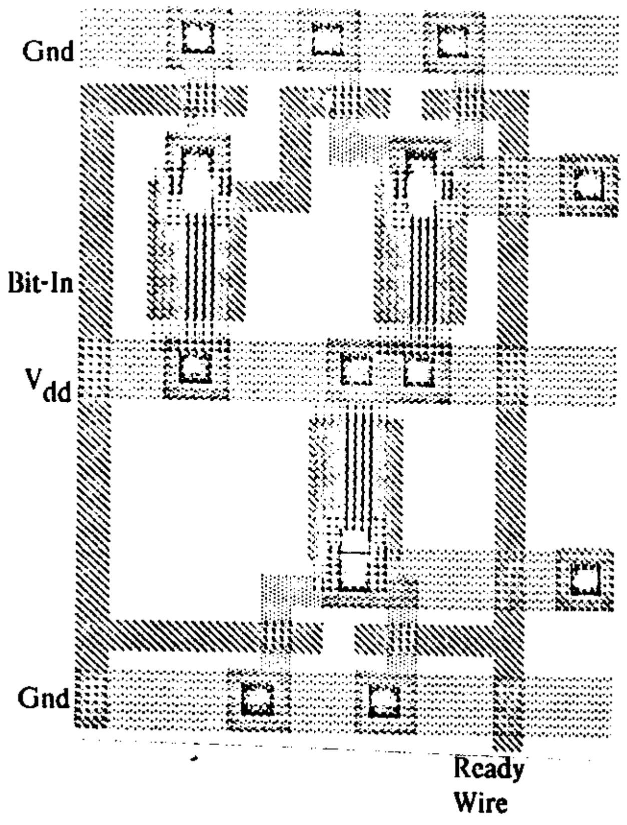


Figure 4.15b One-to Two Converter Bit Cell Layout

single C-element, as we did for the three input C-element, to N-inputs, as suggested by Seitz [4], we run into other problems. We recall that the first stage of the C-element contains a NAND-like structure, which ANDs all of the input wires together. Our MOS logic rules require that the pull-up transistor for this logic gate have a W/L one fourth the size of the effective W/L of all of the pull-down transistors. The pull-up transistor, then, has a length proportional to 2N (assuming minimum width). This is a situation which is not acceptable for a generalized N-bit implementation, as the layout of the C-element would have to change every time a different value of N was chosen. Instead, we will incorporate the functionality of the NAND-like part of the first logic gate of the C-element into a separate NOR gate, like we have in previous designs. We can still implement the NOR-like branch of the C-element as part of the first logic gate of the structure, since it doesn't change the size of the pull-up transistor for different values of N. The circuit then uses a NOR gate for each bit. The NOR outputs are NORed together over all the bits, to give the logic value to replace the NAND-like structure in the first logic gate of the C-element (effectively a AND). The outputs of the bit-level NOR gates is inverted to give the OR of the input variables, which are used to drive the NOR-like part of the first gate of the C-element. The circuits of the two cells are illustrated in Figure 4.16. The layout is partitioned by putting the bit-level NOR and the inverter that follows it, along with the pull-downs for the N-bit NOR-gate and the first logic gate of the C-element, in the bit cell. The output cell contains the pull-up transistors for the C-element and the N-bit NOR gate, and the inverter associated with the feedback variable in the C-element. The layouts for these cells are shown in Figure 4.17. The cells are stacked so that the wires connecting the pull-downs in the bit cells to the output cell circuitry flow throughout the module.

We can now consider the dual-rail implementations of the AND and OR gates, which we will call the AND module and the OR module. Unlike many of the modules we have presented so far, these modules will only be developed for two inputs, as N-bit AND and OR modules are not required for the packet router, and if required can be constructed by cascading the single-bit version. It is fairly clear that N-bit implementations are possible, but is it is not clear how a structure could be generalized for an arbitrary number of inputs. We recall from Chapter 2 that a 2 input AND module can implemented with four C-elements, and a three input AND gate. The ZERO wire output is the output of a C-element

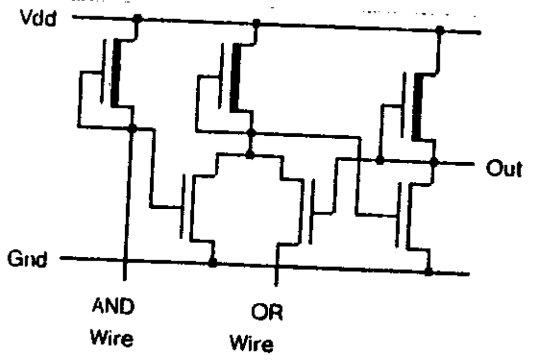


Figure 4.16a Two-to-One Converter Output Cell

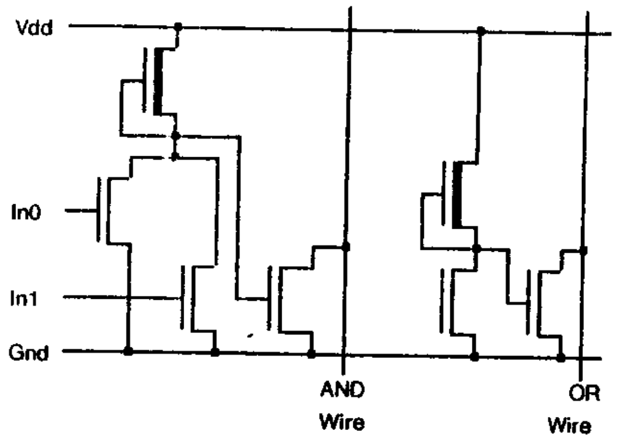


Figure 4.16b Two-to-One Converter Bit Cell

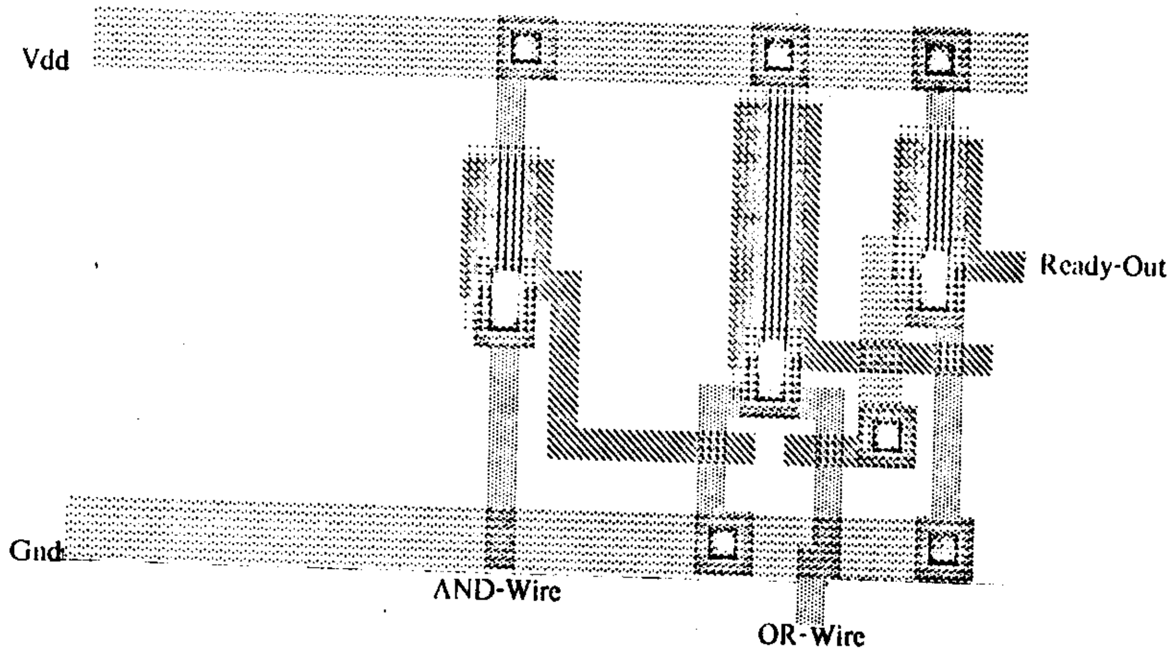


Figure 4.17a Two-to-One Converter Output Cell Layout

whose inputs are the ZERO wires of the two input variables. The other output is the AND of the three C-element outputs whose inputs are the other three possible combinations of the ONE and ZERO wires of the inputs (each C-element has one input from each variable, see Figure 2.17. The three input AND gate can be implemented by using the feedback variable, which is the negative of the normal output, and

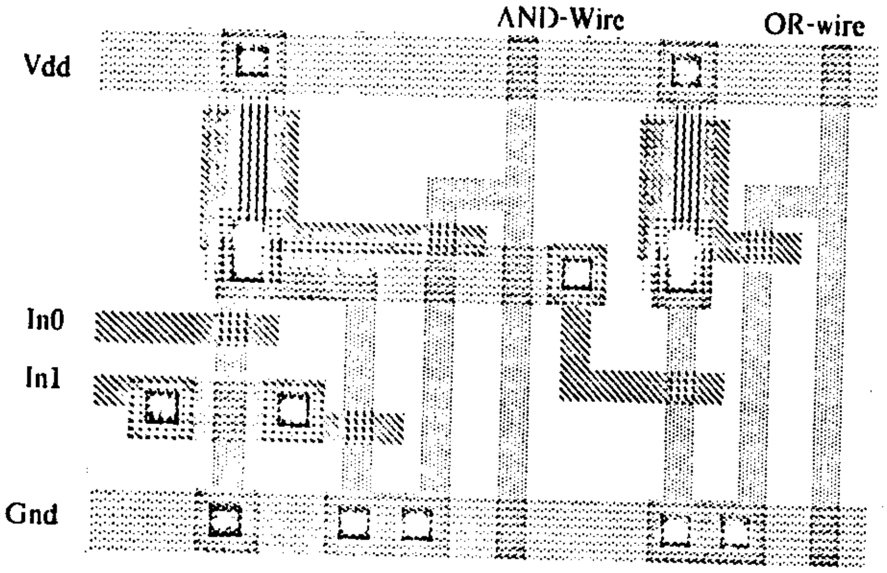


Figure 4.17b Two-to-One Converter Bit Cell Layout

a three input NOR gate. The layout of the AND module is fairly simple, as shown in Figure 4.18. The layout of the OR module is even easier. The dual rail approach allows inversion by exchanging the ONE wire with the ZERO wire. The OR module can be achieved by using this property to invert the inputs and the output. Dual-rail NAND and NOR gates can be implemented by using the same property to invert the outputs of the AND module and the OR module.

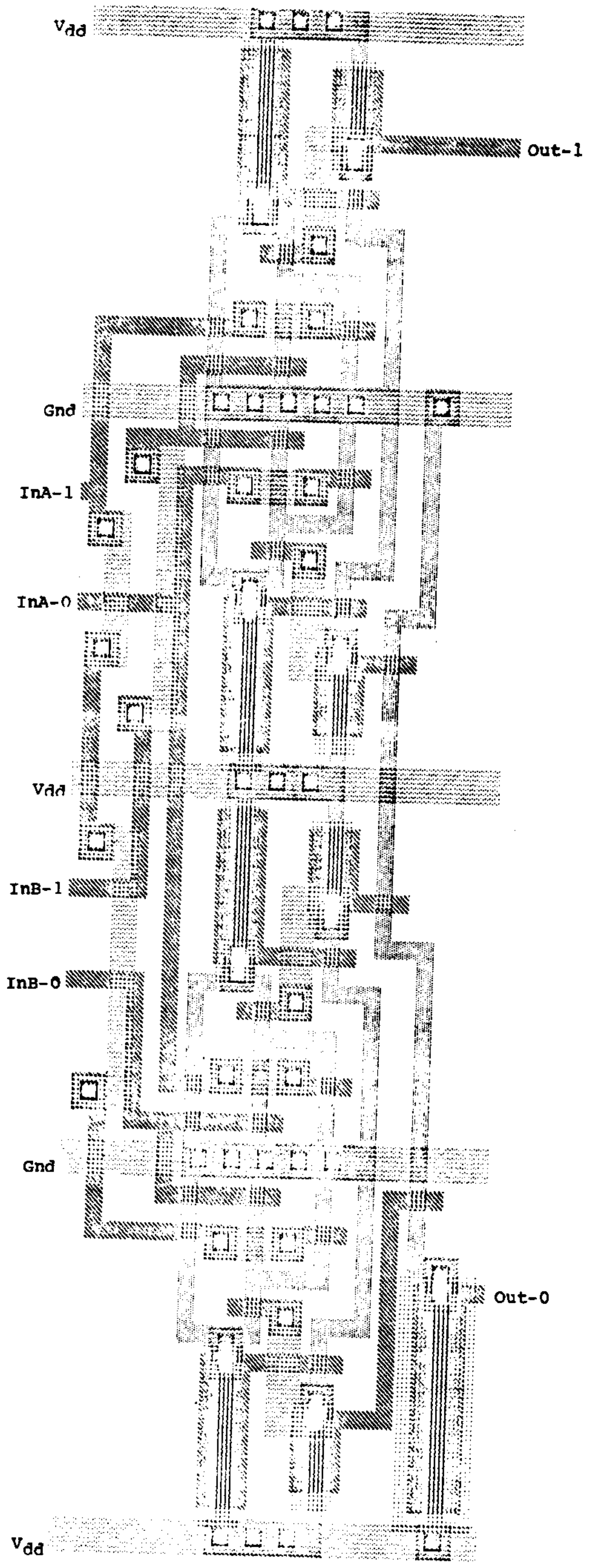
The only remaining module to be considered is the MERGE module. If we examine the MERGE module more closely, (see Figure 2.19) we notice that parts of it have already been designed in the form of other modules. We have already implemented packet detectors in more than one form. We have implemented the Arbiter. Finally, if we look at the four C-elements, along with the OR gate at the data output, we recognize a MUX module. All that remains is the various OR and AND gates around the Arbiter. We recall that the Arbiter module has inverters preceding and following the actual Arbiter circuit, which can be merged with the gates that we use to implement the two OR gates and the two AND gates. The OR gates can be merged with the inverters at the input of the Arbiter module, so that only two NOR gates are required. Likewise, at the output, the inverters can be merged with the AND gates so that

Figure on Next Page

Figure 4.18 AND Module Layout

---

only NOR gates are necessary. We will, however, have to add inverters for the other inputs to the AND gate. We can merge these inverters the output cells of the two packet detectors. We will lay out the Arbiter circuit with the four NOR gates and call the layout a MERGE module, although two packet detectors and a MUX module are required to complete the MERGE module (see Figure 4.19a). The interconnection of the various modular components to form a MERGE can be performed by the system designer, as any layout done here would be rather irregular in its shape, and the system designer may better be able to fit the parts together in a more space-economical form than could be done here. The


Vdd
Out-1
Gnd
InA-1
InA-0
Vdd
InB-1
InB-0
Gnd
Out-0
Vdd

MERGE module layout, as described above, is shown in Figure 4.19b.

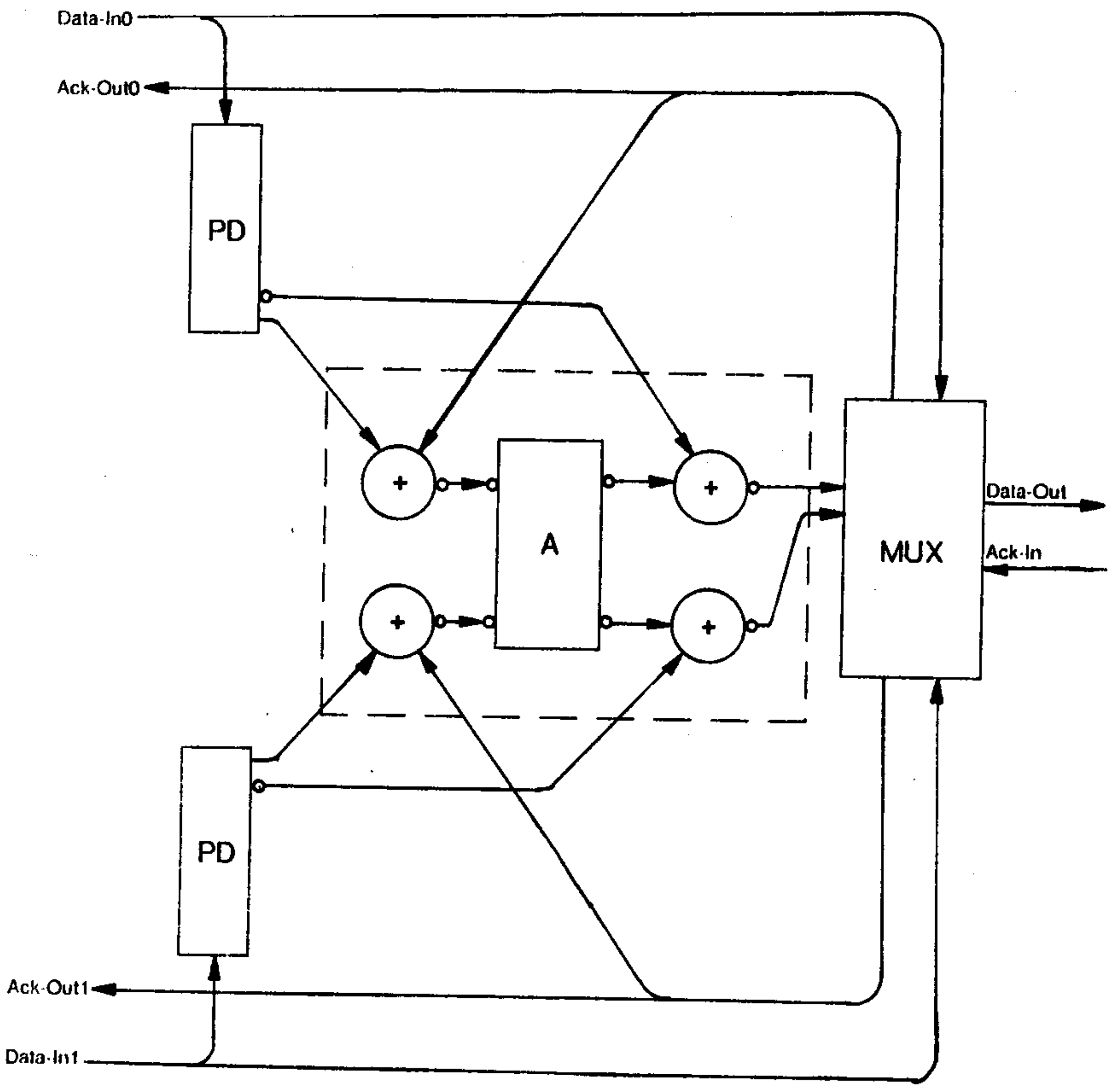


Figure 4.19a MERGE Module Partitioning

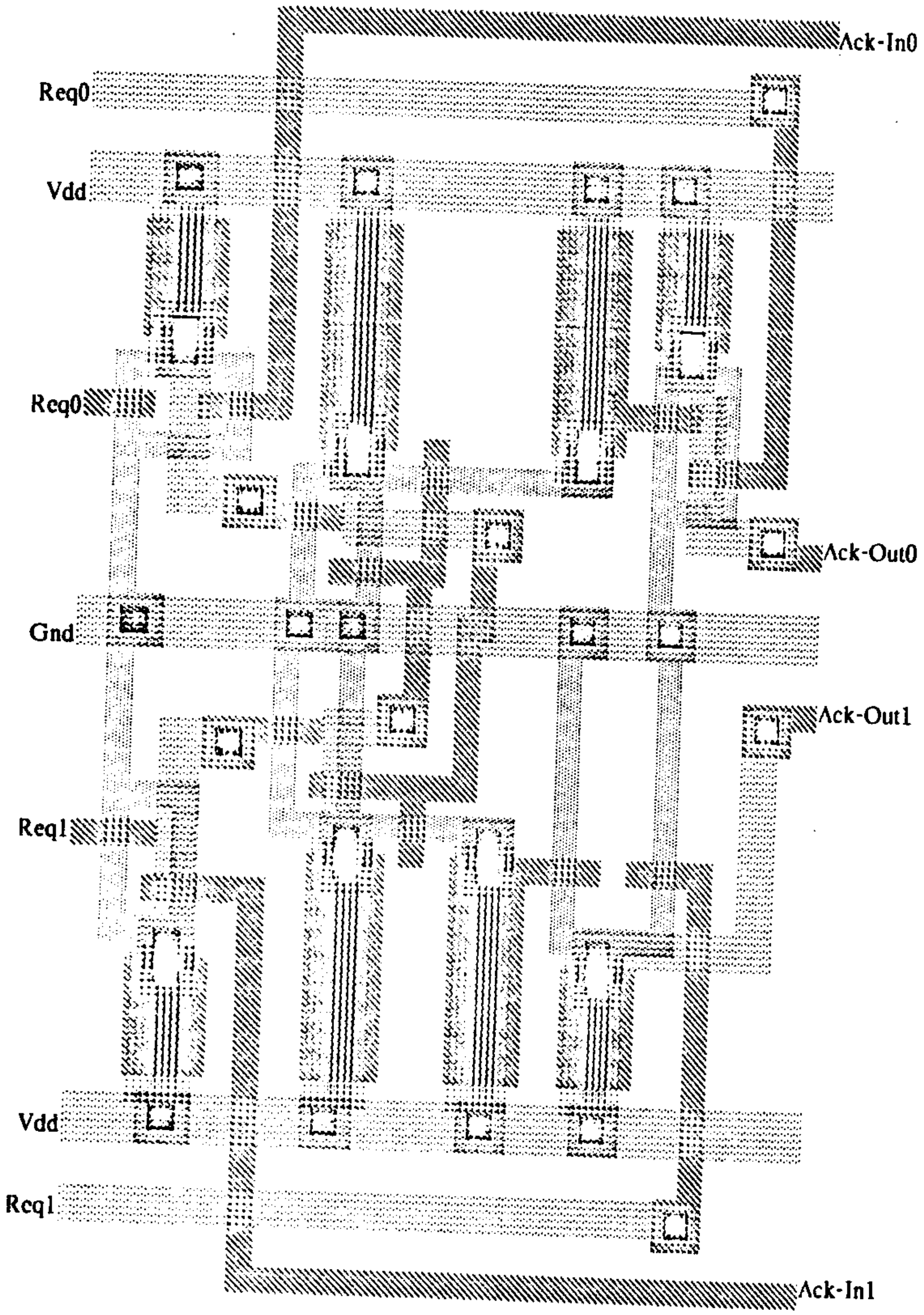


Figure 4.19b MERGE Module Layout

Now that the circuit designs and layouts for all of the modules have been presented, the properties of modules in their final forms can be summarized. The various dimensions of all the module layouts are given in Table 4.1. Linear dimensions are given in λ, while are was calculated in terms of square microns, assuming λ has the 1979 value of 2.5 microns. In Table 4.2, the power dissipation for each module and sub-module is calculated, assuming all of the transistors are turned on, and that every turned-on transistor has a resistance of 10^4 ohms/square. This approximation is an attempt to set an upper-bound for the power limitation, as the exact value would depend on the logical state of the circuit, and whether or not the circuits are undergoing a transition at the time of measurement. Both of the assumptions made in this approximation should tend to over-estimate the power consumption, as the transistors are not always in their resistive regions, and since there will rarely be more than half of the logic gates turned-on at one time.

module	cell	W(λ)	L(λ)	area(μ^2)
AND	-	50	168	52500
Arbiter	-	50	66	20625
Extendable REG	bit	86	90	48375
Extendable REG	ack	86	48	25800
MERGE	-	62	96	37200
MUX	bit	53	86	28488
MUX	ack	43	98	26338
One-to-Two Converter	bit	34	47	9988
One-to-Two Converter	out	41	34	8713
Single-Bit REG	-	59	104	38350
SINK	bit	57	46	16388
SINK	ack	57	45	16031
SWITCH	bit	46	86	24725
SWITCH	ack	48	98	29400
Three-Input C-Element	-	36	51	11475
Two-Input C-Element	-	27	45	7594
Two-to-One Converter	bit	50	33	10313
Two-to-One Converter	ack	62	38	14725

Table 4.1 Module Dimensions

module	cell	Power Dissipation (mW)
AND	-	3.29
Arbiter	-	3.05
Extendable REG	bit	2.62
Extendable REG	ack	2.72
MERGE	-	5.27
MUX	bit	1.81
MUX	ack	1.56
One-to-Two Converter	bit	1.61
One-to-Two Converter	out	2.05
Single-Bit REG	-	3.62
SINK	bit	1.11
SINK	ack	2.22
SWITCH	bit	1.56
SWITCH	ack	1.81
Three-Input C-Element	-	.69
Two-Input C-Element	-	.78
Two-to-One Converter	bit	1.05
Two-to-One Converter	ack	1.42

Table 4.2 Module Power Dissipation

We have designed all of the modules necessary to construct a packet router, but have not considered the timing constraints of the various modules, so that we cannot be entirely sure that a system built by combining these modules will work as desired. A brief analysis of the timing constraints in terms of a simple transient module for the MOS transistor will be presented here to help convince the reader of the practicality of the approaches taken in the design of the various modules. Furthermore, a test chip design will be presented that will help to prove the viability of the module designs presented here. It is suggested, through, that more accurate computer transient analyses would help verify the module designs even further.

An examination of the modules and they way they would be connected to form larger systems leads one the conclude that the timing characteristics of four of the modules is critical to guaranteeing a properly functioning system. The circuits that convert to and from dual-rail code, of course, are key elements, and must work properly for any dual-rail logic system to work properly. The REG module is also crucial, because it generates Acknowledge signals that will eventually remove the data from its own inputs. If the REG module has not attained a stable state before the inputs are removed, the data involved could be lost. Finally, the Arbiter, which is the heart of the packet router, must work properly to prevent packets from being superimposed on each other. We will attempt to show that the modules mentioned above will perform as desired, using a simple model for the MOSFETs and some simple reasoning. We will assume, while considering each module, that the rest of the system obeys the dual-rail/Acknowledge and Ready/Acknowledge protocols presented in Chapter 2.

Our very simple first-order transient model for the MOSFET uses a simple RC time constant that is often used to express the speed of a given MOS technology. The parameter, called tau, is the RC time constant for a square transistor, treated as a resistor, charging another transistor with the same gate capacitance. If we scale this parameter for different shaped transistors, we can estimate the rise times and fall times of any node. We remember that the effective resistance of a transistor is proportional to the the inverse of the W/L ratio. We can use this model to consider the operation of our various modules when connected together in a system.

First, we can consider the modules that convert to and from our dual-rail code. The circuit to convert from Ready/Acknowledge to dual-rail code is very simple. The One-to-Two Converter will fail only if the Ready signal causes a packet to be sent before the data inputs have stabilized at the input of the Converter. If we examine the design in more detail (Figure 4.14), we see that the Ready signal passes through a super-buffer that has to charge 2N minimum-sized gates. The delay inherent in achieving this will guarantee that if the Ready signal occurs after the data inputs have settled, that no glitches will occur in the output wires of the module. Furthermore, when the Ready signal is generated in another system by a packet detector, there will be additional delay in between the data signals and the Ready signal.

The Two-to-One Converter can only fail if the C-element in the output cell of the Converter (see Figure 4.16 and Figure 4.17) fails. The C-element will fail if the inputs that put it in a particular state are allowed to change before the transistor of the first stage of the C-element associated with the state variable is turned on. This is close to impossible, because this transistor will be turning on (or off) at the same time the output node is being turned on (or off). The added delay from the NOR structures, and the associated C-element will delay the acknowledge signal significantly,so that it will should stabilize well after the C-elements stabilize. Furthermore, the Two-to-One Converter is generally used at the output of a system, the delay involved in sending the Ready signal (the output of the Two-to-One Converter) off-chip will also help prevent any problems with the Two-to-One Converter module.

The SINK module is very much like the Two-to-One Converter. The inputs must remain stable until the C-element have stabilized. The inputs will not change, however, until fairly long after the the output of the C-element has stabilized. As soon as the output of the C-element has stabilized, however, the feedback transistor of the C-element will be be charged to the correct value that will keep the C-element in its new state. The wire delays involved are insignificant compared to the contribution of the gate and parasitic capacitances inherent in the gates of the various transistors connected to the output node of the C-element. Furthermore, there must be some additional circuitry between the output of the SINK module and its inputs, so the delay inherent in this circuitry will assure proper operation of the SINK module.

The REG module has two types of C-elements. The C-element associated with the packet detector should have no problem, for the same reasons as the SINK module, since the packet detector in a REG module is a SINK module with the added delay of the other C-elements in the REG module between it and the previous stage. The other C-elements hold the data values in the REG module and are, therefore, critical to the operation of the REG module. They can fail if the Acknowledge signal from the next stage reaches them before their state variables can stabilize. Much like the SINK Module, the transistor that must be turned on is connected to the data outputs, and will be turned on at the same time as the transistors associated with the next stage. The Acknowledge signal, then, would have to be generated instantaneously, because the state variable is stable as soon as the transistor is on. The REG module, therefore, should have no timing problems. This statement holds true for both implementations of the REG module, although the multi-bit implementation would have more delay in the acknowledge loop because of the capacitance in the packet detector, which includes the diffusion wires that join up all of the bit cells.

Finally, we can examine the Arbiter. The Arbiter must only acknowledge one input. The threshold element is the critical part of the Arbiter, as it is designed to prevent the Arbiter from acknowledging more than one input. If we examine the threshold element, (see Figure 3.7b) it can be seen that one output of the cross-coupled NOR gates must exceed the other by one MOS threshold to give an output. Clearly, as long as the poly wires connecting the outputs of the NOR gates to the threshold element act like perfect wires (one voltage everywhere) then one voltage cannot be greater than, and less than another voltage by one MOS threshold, at the same time. The main assumption here is that, once the Arbiter has a significant voltage difference in its outputs, that it will remain in its states. This assumption requires a low noise-level. Much experimentation will be required to determine if a particular implementation has a low-enough error rate for a particular application. For the purpose of this thesis, however, we will assume that the any such problems can be dealt with using methods that have been developed for making reliable synchronizers in more conventional computing machinery.

Although we have argued that the modules have no timing problems, the design techniques used in this thesis have no known precedent in MOS logic. It is desirable, therefore, to test out the the modules by having a test chip fabricated. It was determined that the REG module is the most important module, since its timing has been determined to be of critical importance, and because it would be fairly easy to test. After deciding that it was important to test the REG module, it was necessary to determine the best way to test the REG module. If a FIFO consisting of a string of REG modules could pass a string of data values through at their maximum rate, then one could conclude that the REG modules on the test chip worked properly. Unfortunately, it is impossible to enter a data string into a string of REG modules at the maximum possible rate, because of the delay associated with sending the first Acknowledge signal off-chip. By the time the Acknowledge signal had stabilized on the output pin of the chip, the byte that generated the Acknowledge would be transmitted through the string until it reached a point where the next stage already had a packet. A FIFO using REG modules on a chip would only allow the testing of the static storage capabilities of the REG module. To test the dynamic capabilities, it is necessary to split the FIFO at some point and AND the data values with some variable called Run. As long as the Run variable was low, the REG modules before the AND gates could be filled. After the REG modules had been filled, the Run variable could be raised, and the packets would proceed through second half of the FIFO at very close the maximum rate possible. The accuracy of the test would depend on the delay of the circuits used to break the FIFO into halves.

It was decided that the test described above would accurately test the REG module. The FIFO was made two bits wide, since the delay of the Acknowledge loop (the factor that could cause problems) is smaller with fewer bits. A single bit implementation was rejected for two reasons. First, the multi-bit REG modules will never be used for single-bit variables, as a different REG module was designed for this purpose. Second, a multi-bit variable was desired to test the Two-to-One and One-to-Two Converter circuits in a meaningful fashion. It is suggested that another test chip be designed for the single-bit implementation of the REG module, as the timing of this circuit is also quite important.

The layout of the FIFO test chip was fairly straight-forward (see Figure 4.20). The AND gate required for the Run circuitry should be as fast as possible. A super-buffer NOR circuit used for this purpose (see Figure 4.21). The circuit is basically a non-inverting super-buffer, with a pull-down transistor to disable the output with the Run signal is high. The pull-down transistor must be able to hold the output low even when the pull-up transistor of the super-buffer is on. Given Mead and Conway's [6] process parameters for the threshold voltages of the depletion and enhancement transistors, the pull-up of a normal inverter is barely saturated when the output is low, so the drain current can be approximated as $I_{ds}=(k/2)(V_{gs}-V_{th})^2$, or $I_{ds}=(k/2)(V_{th})^2$. If the input to the super-buffer is high, the pull-up will not be saturated, and the drain current can be approximated as $I_{ds}=k(V_{gs}-V_{th})V_{ds}=k(-V_{th})V_{ds}$. We know from the process parameters that $-V_{th}=.8V_{dd}$, and the drain to source voltage should be about $.8V_{dd}$ to prevent the next stage from turning on. For a normal inverter load $I_{ds}=.32k(V_{dd})^2$, while for our turned-on super-buffer load $I_{ds}=.64k(V_{dd})^2$. The current is double that of the normal pull-up transistor, so the pull-down transistor associated with the Run signal must be doubled in width to compensate. The circuit diagram and layout of the circuit used are shown in Figure 4.21. The initialization feature of the REG module was also tested by setting-up a sequence of data and spacer states in the FIFO that can be read out sequentially from the FIFO. The chip will be fabricated in June of 1980.

We have developed all of the modules necessary to implement the packet router. We have tried to show that they will work by simple reasoning, and by designing a test chip. Simulations of the modules could be a better approach for determining how well the modules function. Assuming that testing shows that the modules work properly, we can consider the layout of the packet router.

The full layout of the packet router is a matter of interconnecting a large number of modules together. Furthermore, it may be desirable to add some pipelining (using REG modules) to each two by two router. Considerations of these areas are suggested for further research. A general block layout of the various modules will be sketched here just to show that it can be done on a reasonably small piece of silicon.

Before we can do the layout, we must re-examine the block diagram of the packet router. We recall that the router is divided into two different parts, each repeated twice. Figure 2.21 and Figure 2.22 show

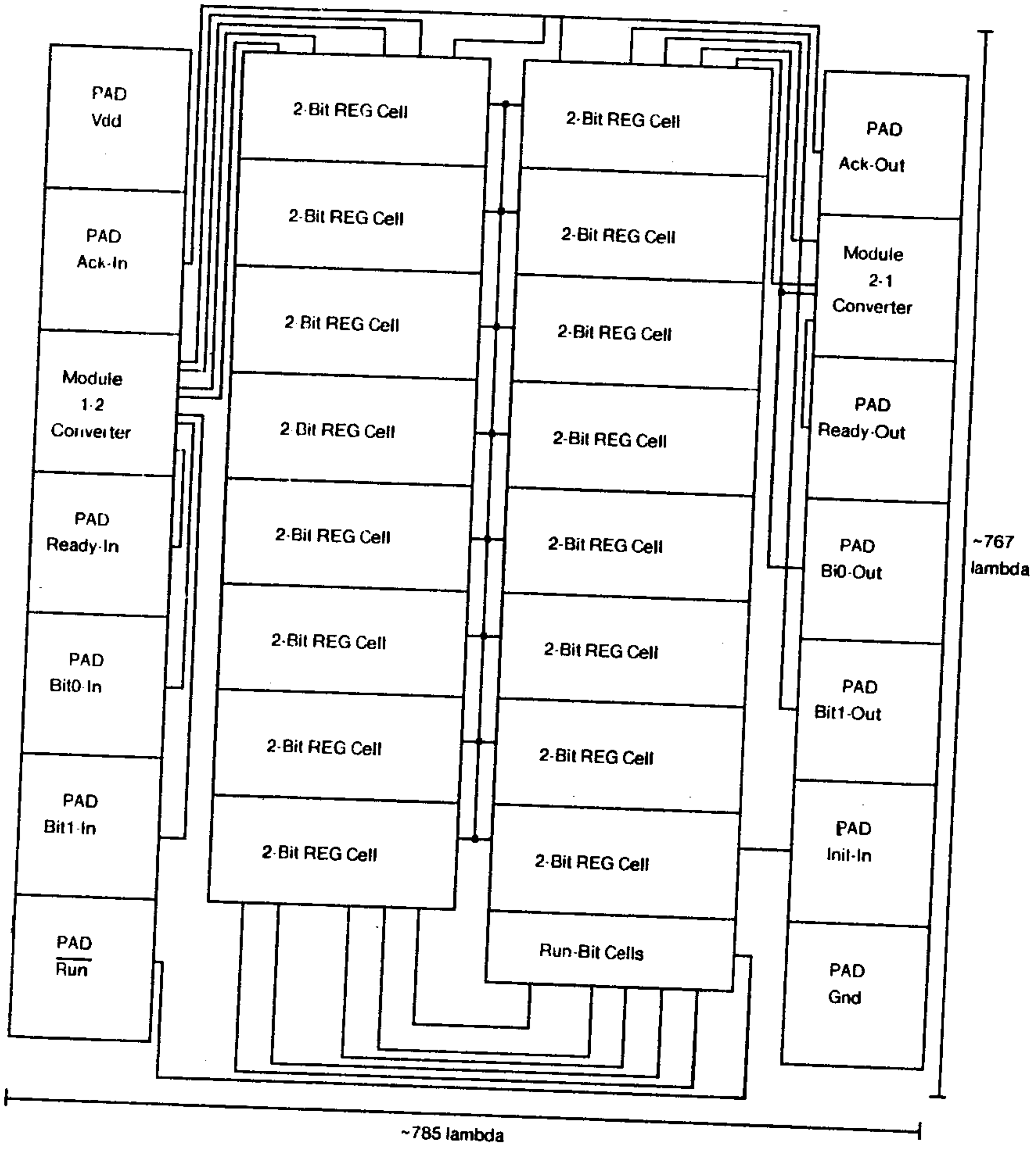


Figure 4.20 FIFO Test Chip Layout

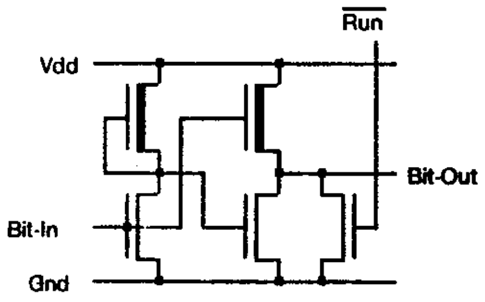


Figure 4.21a Run-Bit Cell

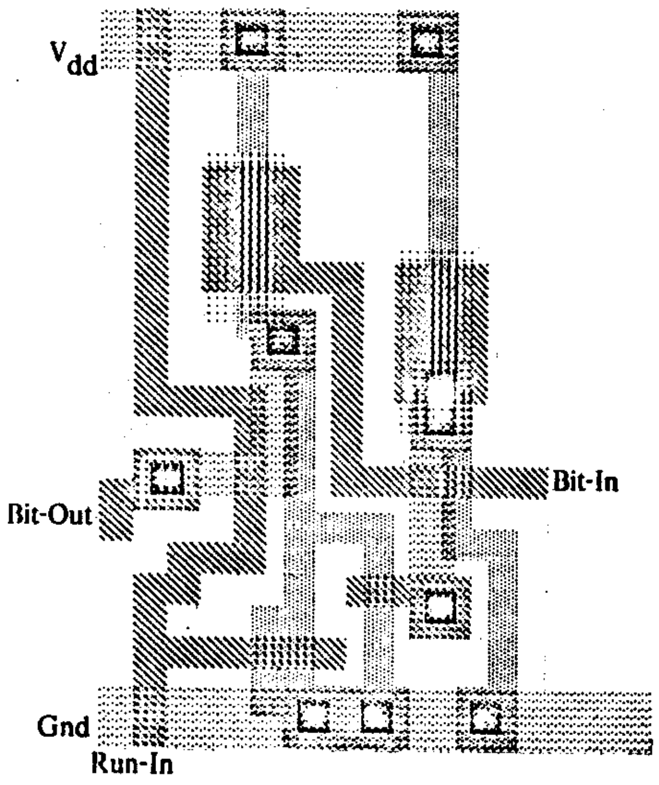


Figure 4.21b Run-Bit Cell Layout

the CM and the OM, complete with Acknowledge wires included. In any large integrated circuit, the area required for interconnection between various parts of the system is a major part of the total chip area. In an effort to minimize this excess area, we will try to determine which parts of the packet router are likely to require large amounts of interconnect area. It is easy to see that the byte data path, especially the area where the wires from the upper CM and the wires from the lower CM cross each other. Since 18 wires (8 dual-rail bits and two signal wires) from each CM must cross, we can see that the interconnect area will be substantial, unless it is restricted to one joint area in the middle of the chip. The additional circuitry of the two CMs and OMs can be put above and below this, and the SWITCHes, the MUXs, the conversion modules, and the input and output pads can be put on either side of this area. Even with this, the interconnect area will span the height of the SWITCH and MUX modules, and its width will be 7λ for each crossing wire (4λ for contacts, and 3λ for metal-metal spacing), which yields A252λ for 36 crossing wires. A full layout of the lower half of a packet router based on the principles discussed above is shown in Figure 4.22a. The modules that act as the control logic in the CM and the OM are shown separately in Figure 4.22b and Figure 4.22c. As can be seen a packet router layout can easily be accomplished in 800λ by 3690λ. In addition, FIFO buffering can be added to the input of the CM or the output of the OM by using two REG modules per bit of storage. The buffering will contribute 172λ per bit of storage. Assuming that $\lambda = 2.5\mu$, the basic packet router will be 2000μ by 9225λ, with the width increasing 430μ per bit of storage. A chip this size is certainly not excessively large using current process technology.

It should be noted at this point, that one important area of concern has not be touched. The size of some of the structures used in the modules discussed throughout this thesis makes any accurate estimation of the performance of a packet router almost impossible. Computer simulation is the most, and in most case the only, accurate means of estimating circuit performance. Unfortunately, the adequate computer tools to give a reasonably good guess at the performance of the router were not available at the time this thesis work was done. Some computer simulations that were performed indicated that the basic C-element had a delay time of 6 nanoseconds (rising), or 13.5 nanoseconds (falling) with no load on the output. The suggests that a byte could pass through the forward path of the router in 12 nanoseconds. This approximation does not account for the loading of the outputs of the SWITCH and MUX modules.

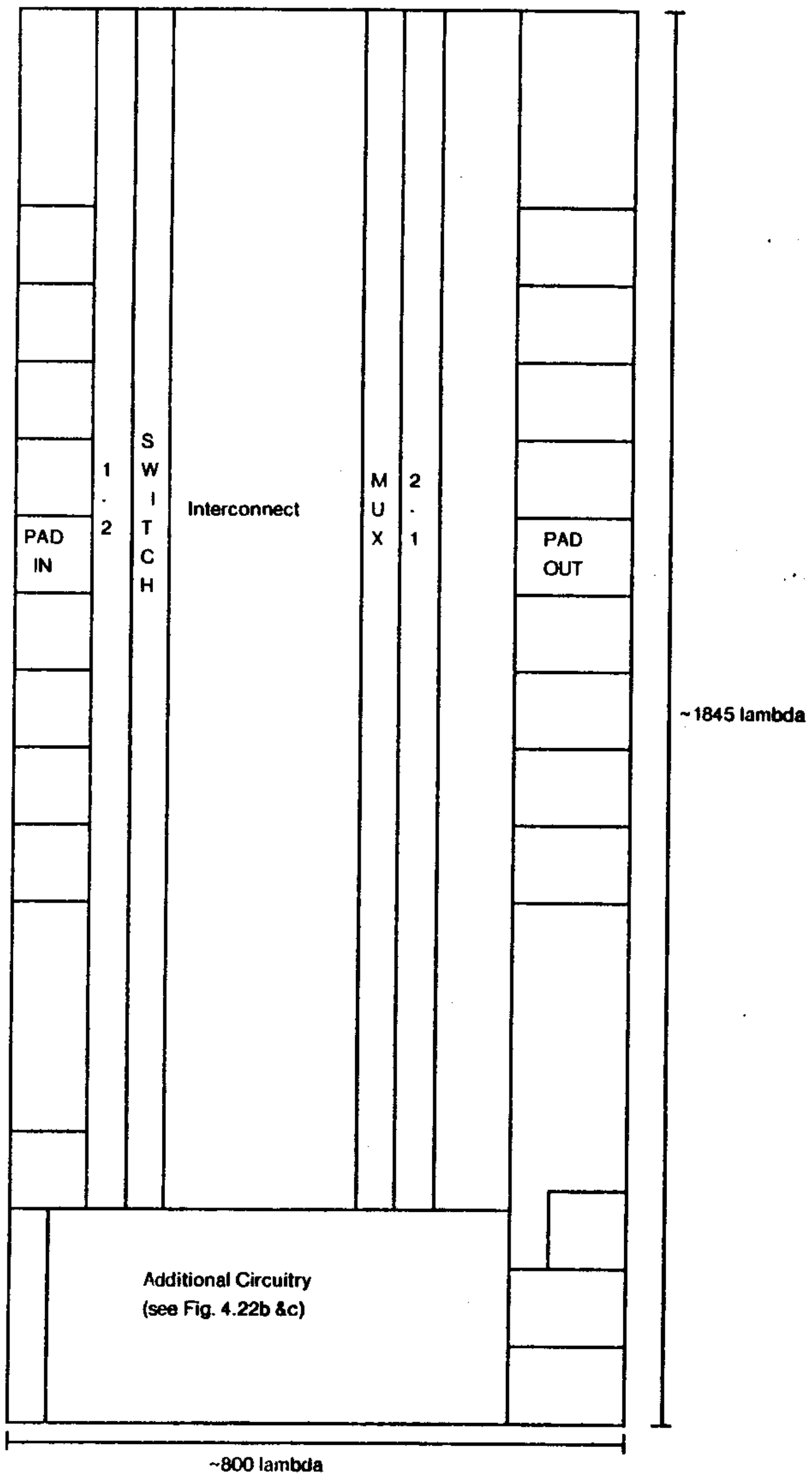


Figure 4.22a Layout For Packet Router (Lower -Half)

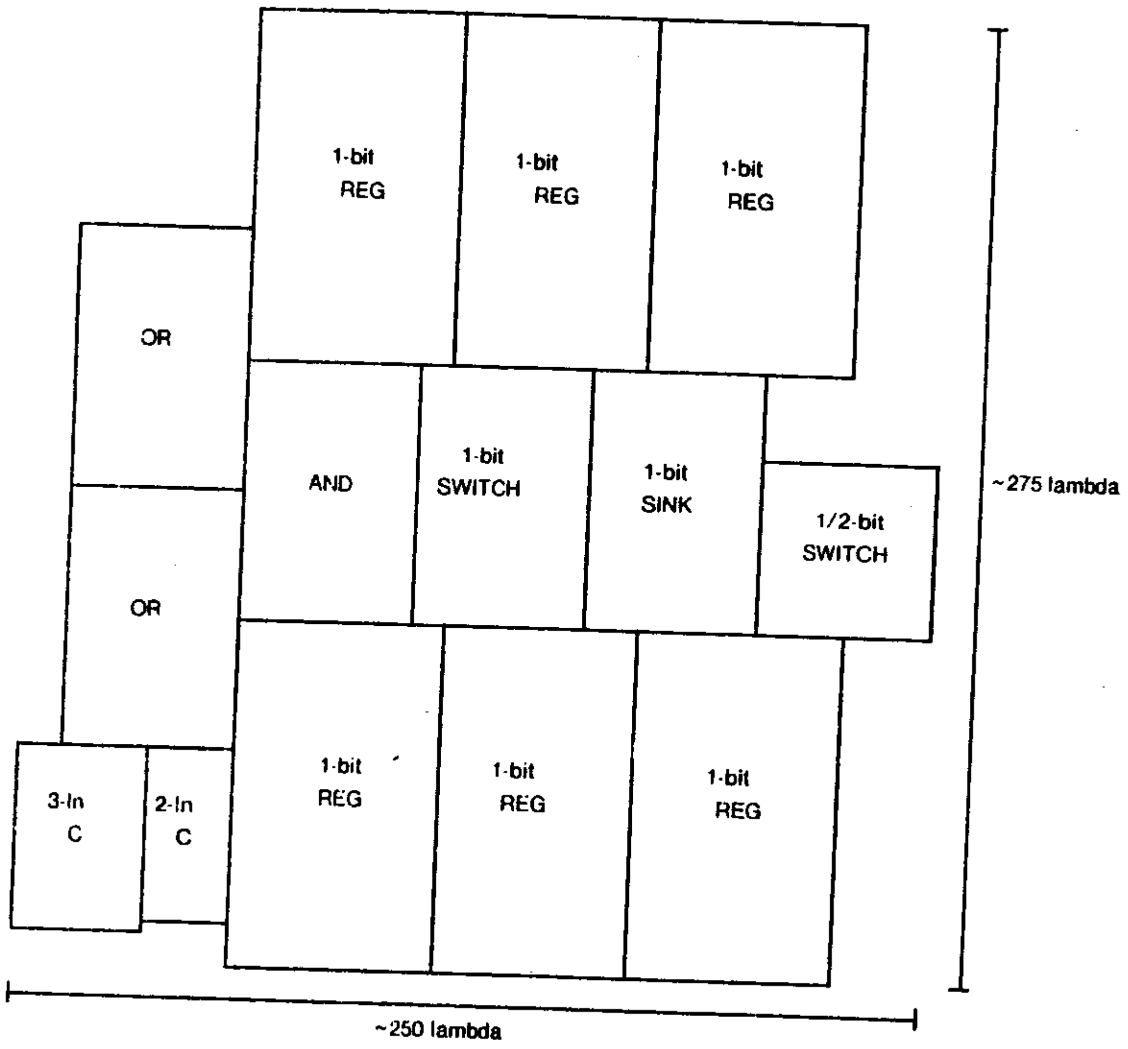


Figure 4.22b CM Layout

It does not account for the time required to generate the ready output signal, nor does it account for the pad delays. Furthermore, the acknowledge signal could take considerably longer to generate, so that a single byte might tie up the packet router for considerably longer than the time for the data to pass through the router. The simulation of the C-element, then, does little more than give us an order-of-magnitude guess at the performance of the packet router.

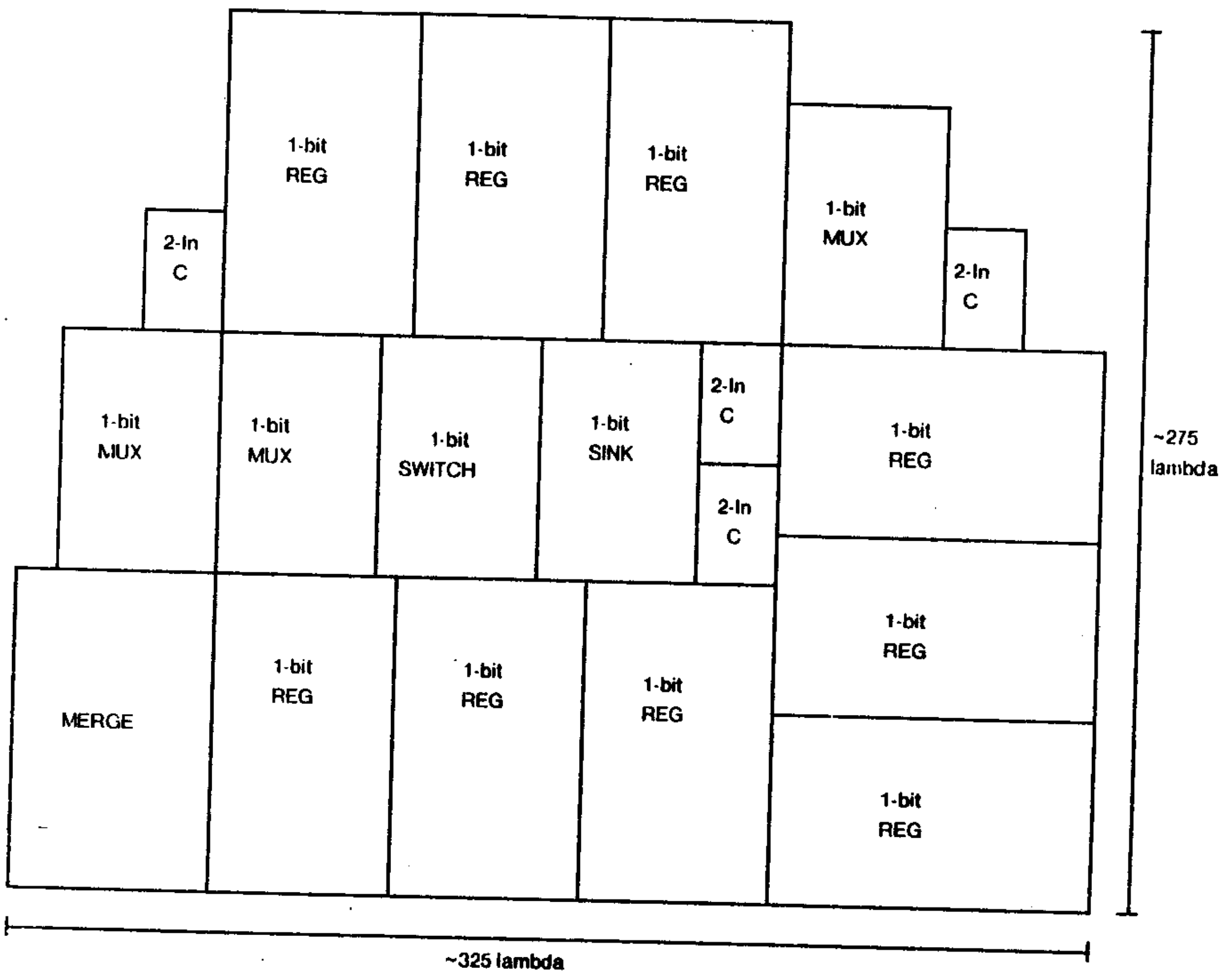


Figure 4.22c OM Layout

In this thesis, we have shown that a two by two packet router can be implemented on a reasonably small piece of silicon. We have not been able to show that the performance of such an implementation would be adequate for the intended application. This is a question that could be answered by further research in the area, or by implementation of the two by two router using the modules developed here. The main accomplishment is that a set of general-purpose control modules have been developed, and they can be used to implement a number of functions similar to that of the packet router. The other main question of this thesis is closely related to the first. Dual-rail logic has been shown to be implementable, but not necessarily practical. This question will be answered in part when the test chip becomes available. Implementation of the packet router, however, would give much stronger evidence concerning practicality, as it uses all of the various modules developed here.

Perhaps the most important idea developed in this thesis is the approach taken to designing the packet router. Instead of developing special-purpose modules to build the packet router from the bottom-up, the modules were developed from well-understood general-purpose data-flow modules. If the modules developed here are implementable and practical, designs can be developed directly from data-flow descriptions. Modules could be interconnected using a wire-routing algorithm, like that of Preas and Gwyn [11]. Furthermore, the placement of the modules could also be done automatically as described in Preas and VanCleemput [12]. This would greatly ease the design effort necessary to implement an VLSI circuit. The user of such a set of design tools would not need to understand anything about the technology, or about the tools themselves. The user's data-flow description could be implemented with no effort on his part. This ability is certainly an attractive one when one considers the amount of time and effort required to implement a VLSI circuit using contemporary design tools.

In sum, this thesis has been a moderate success. The goal of designing the modules necessary to implement a two by two packet router was achieved. Although they have not been tested to the extreme, we have shown that they are almost sure to work. Several areas were further work would be useful have been pointed out. It is hoped that the modules do turn out to be practical and valuable. Even if they are not practical, the experience gained from their design will be valuable, either by helping to correct the problems that make this particular approach impractical, or by giving the researchers some insight into what sorts of approaches are practical when one attempts to implement a VLSI logic circuit.

[1] Jack B. Dennis, David P. Misunas & Clement K. Leung, A Highly Parallel Processor Using a Data Flow Machine Language, Computation Structures Group Memo 134, Lab. for Computer Science, M.I.T., January 1977.

[2] G. B. Boughton, Routing Networks in Packet Communication Archetictures, Department of Electrical Engineering and Computer Science, M.I.I., SM Thesis, June, 1978.

[3] Clement K. Leung, On a Design Methodology for Packet Communication Architectures Based on a Hardware Design Language, Computation Structures Group Memo, Lab. for Computer Science, M.I.T.

[4] Charles L. Seitz, "System Timing", Chapter 7 of Introduction to VLSI Systems by C. A. Mead & L. A. Conway, Addison Wesley Press, 1980.

[5] Suhas S. Patil, Bounded and Unbounded Delay Synchronizers and Arbiters, Computation Structures Group Memo 103, Lab. for Computer Science, M.I.T., June 1974.

[6] Carver Mead & Lynn Conway, Introduction to VLSI Systems, Addison Wesley Press, 1980.

[7] Jack B. Dennis, "Modular, Asynchronous Control Structures for a High Performance Processor", Record of the Project MAC Conference on Concurrent Systems and Parallel Computation, Association for Computing Machinery, New York, 1970.

[8] D. E. Muller, "Asynchronous Logics and Application to Information Technology', in Proceedings of a Symposium on the Application of Switching Theory in Space Technology, edited by Aiken & Main, Stanford University Press, 1963.

[9] J. C. Sims and H. J. Gray, "Design Criteria for Autosynchronous Circuits", Proceedings of the Eastern Joint Computer Conference, pp 94-99, December 3-5, 1958.

[10] B. Gilchrist, J. H. Pomerine, & S. Y. Wong, "Fast Carry Logic for Digital Computers", IRE Transactions on IElectronic Computers, Vol. EC-4, December, 1955.

[11] B. T. Preas & C. W. Gwyn, "General Hierarchical Automatic Layout of Custom VLSI Circuit Masks", Journal of Design Automation and Fault Tolerant Computing 3, 1, pp 41-58, January 1979.

[12] B. T. Preas & W. M. VanCleemput, "Placement Algorithms for Arbitrarily Shaped Blocks", Proceedings of the 16th Design Automation Conference, pp 474-480, June 1979.