6. When The Wafers are Delivered...

The return of the finished wafers is an exciting moment for the IC designer. Proper preparation for this day can help to prevent frustrating delays in wafer separation and packaging, thus bringing the wafers to the testing stage more quickly. This, in turn, promotes rapid feedback concerning the success of the designs. The following discussion of wafer separation and chip bonding techniques is primarily aimed at those in a research, rather than a production environment.

6.1 Wafer Separation

The wafers, as returned from the fabrication line, are 3" disks of silicon which must be broken into chips the size of a DIP cavity. Wafer separation is accomplished in one of two ways: *scribing* or *sawing*.

Scribing is a simple operation similar to glass cutting. The wafer is held on a vacuum table (which is an integral part of the scribing machine, or *scriber*) and a diamond-tipped scribing tool is dragged across the surface within the confines of the scribe lines. The tool, where it slides over bare silicon, induces stress cracks in the vertical direction under the diamond tip. The pressure that the scribing tool exerts on the silicon is critical, too little results in random breakage in the fracturing operation, while too much produces stress cracks in the horizontal direction. Such cracks cause splintering of the wafer radially from the scribe lines, probably into active circuit elements. After each scribe line in the grid has been "scratched" in this fashion, the wafer (at this point it is still in one piece) is removed from the scriber and broken into chips. This may be achieved in a number of ways, for example by sandwiching the wafer in some soft material (e.g. rubber sheeting, filter paper), supporting it on a foam rubber block, and rolling a cylindrical bar over it. If the wafer was properly scribed the flexing force is concentrated at the scribe lines and the wafer fractures cleanly along the scribe lines.

Sawing is an alternative to scribing. In this technique a thin saw blade with an edge containing diamond-dust is used to cut approximately half-way through the silicon wafer. Again, the wafer is removed from the saw in one piece and fractured by techniques similar to the one outlined above.

Sawing offers several advantages over scribing. The saw can slice anywhere on the wafer, thus no scribe lines are needed. This allows dense packing of projects on a multi-project chip; when the wafers are returned from fabrication each designer can have a wafer sliced up without regard to the location of other projects on the wafer (i.e. by sacrificing neighboring projects to the saw

blade). Sawing also leaves square edges after fracturing which makes manipulating the chips with tweezers an easy task.

Disadvantages include the necessity of removing the silicon dust (*slurry*) generated in the sawing process -- this means an extra cleaning step following wafer separation. More importantly, sawing equipment is complex and expensive ($12,000-$20,000) compared to scribers ($2,000-$5,000). There is also more maintainance required and more setup overhead involved. At this time it seems that scribing is a more economical, less finicky approach to wafer separation, especially in an environment where only a few wafers are handled each month.

6.2 Chip Bonding

Once the wafers are fractured into chips only *bonding* remains before they are ready to be tested. Bonding encompasses two different operations, *chip attachment* and *wire bonding*. In the first operation the chip is permanently affixed to the IC package; the second involves connecting the aluminum pads on the chip to posts surrounding the package cavity. These posts are connected through the package to the external pins.

Chip attachment is a straightforward process, especially in a low volume research environment. The chip must be solidly attached to the mounting pad (*header*) in the package cavity. The bond should exhibit low thermal resistance and make good electrical contact with the silicon substrate. Common means of attachment include *solder* bonding (both header and chip must be heated to the melting point of the solder used) and *eutectic* bonding (usually utilizing a gold-silicon alloy, see [Glaser 1977]). By far the most convenient for the researcher is *epoxy* bonding: the backside of the chip is dabbed with a commercially available gold/epoxy mixture and then pressed onto the header. Tweezers suffice for handling the chips. The header and chip are baked at a low temperature for a few hours to cure the epoxy and the assembly is ready for wire bonding.

Both of the manual wire bonding techniques in widespread use require considerable skill on the machine operator's part. *Thermocompression* bonding relies on pressure and heat to produce a strong bond. Typically a gold ball (on the end of a fine gold wire) is squashed against the aluminum bonding pad on the chip. The header and the capillary holding the wire are maintained at about 300 degrees centigrade and the bond is formed in a fraction of a second. As the capillary is withdrawn from the bonding pad, wire is automatically payed out; the operator maneuvers the capillary over the desired post and the wire is mashed against it, forming the second bond. As the capillary is backed away, the wire is cut by a gas flame, which simultaneously forms a gold ball for

the next bond.

Ultrasonic bonding utilizes aluminum wire and ultrasonic energy to make bonds. The aluminum wire is pressed against the bonding pad and a short burst of ultrasonic energy locally heats the wire/pad interface so that a bond is formed. Similarly, a second bond is made on a post, and the wire is cut, usually by mechanical means. The header may be heated to assist the bonding process.

Ultrasonic bonding offers low materials cost but is less flexible than the thermocompression technique, which allows "daisy-chaining" of connection points. Thermocompression also gives more freedom to choose the angles at which wires leave the bonding pads, enabling some further flexibility which may be needed in a research environment.

In general, research chips need not be hermetically sealed in their packages, often a piece of tape over the cavity (or no cover at all) will prove adequate. Users should be aware, however, that MOS circuits exhibit very different device characteristics when operated in light.

6.3 IC Testing

[section contributed by Peter Dobrowolski, UC Berkeley]

Although IC's are quickly becoming cheaper and easier to use, their complexity, and the difficulty of testing them, is increasing quickly, too. It is important for both IC designers and users to consider testing in some detail.

The process of IC design may soon resemble the programming process. Programs are written, tested, rewritten and retested. This iteration continues until correct and perhaps optimized software is produced. Since it isn't absolutely necessary that a program work the first time, it can be written somewhat less conservatively, with an emphasis on creativity rather than cautious restraint. Once the procedure for producing IC's from symbolic logic diagrams is streamlined, IC designers will be able to apply the same principles that programmers use now. Quick and effective testing and debugging of prototype chips then becomes a crucial issue for this experimental approach, which is much different from today's industrial environments which must, of necessity, be more conservative, than research groups and individuals who quickly want to produce a few IC's to try out some novel ideas.

Testing involves such concepts as:

functionality	Does the IC work as it should?

if yes:

quality	How well does the IC perform?
reliability	Will the IC always work as it should?
testability	How easy is it to determine functionality, quality, and reliability?

if no:

reason for failure	Bad processing run
	Bad mask set
	Misplaced or misshaped features in the layout
	Logic errors at the circuit level

These questions should be kept in mind during all phases of the the design process, and during evaluation of the finished chips.

Assuming that the reader will participate in all the stages of IC design, this section provides some guidance for the testing process by dividing it into two parts: *defensive design* and *systematic testing.*

6.3.1 Before ICs are made - defensive design

In order to facilitate IC design, a number of simple rules should be followed:

Observe the design rules. The chip may work even if a design rule is broken, but the odds are against it.

Keep your chip size within reasonable bounds. The smaller the better. A typical maximum size (for 1978) should be about 6 millimeters on a side. Yield decreases sharply with increasing circuit size and with design rule violations.

Include test patterns on your wafer. Test patterns consist of simple structures such as single transistors or *ring oscillators*; examples are described in detail in section 7.2. Test patterns can be used to determine the process and circuit (device) parameters. For example, we can determine the basic inverter stage delay by taking measurements on a ring oscillator test structure.

Think about how you will test every module you design. When you design an LSI cell or subsystem, think through how you will test it, just as in writing a subroutine you should plan some tests to check its correctness before using it in a larger program.

Consider input/output. Use a *lightning arrestor* circuit to protect your input pads against damaging overvoltages from static discharges. Outputs should be buffered to enable them to drive capacitive loads of up to 50pf. Standard input and output structures are described in Appendix E.

Provide access to internal paths of the circuit. In the event of a malfunction it may be desirable to access internal nodes for debugging. Typically on experimental IC chips there is room for additional bonding pads. Not all of them have to be connected when the chip is mounted in a package. Several chips of the same IC can be mounted in different ways in separate dual in-line packages. Alternatively the test bonding pads can be accessed on the *wafer prober.*

Consider the testability of the overall system. It is very likely that the LSI module that you are designing is going to exist as a small part of a large system. Then it may not be sufficient that every module be testable by itself, but necessary that every module be *individually testable within a large system.* This is because modules, unlike subroutines, cannot be reproduced exactly. Certain flaws may depend critically on the working environment of the chip (loading factor, noise pickup) and can not be seen in a separate test setup. Providing the capability to detect and isolate, or even correct, defects in large systems is perhaps one of the most challenging tasks facing LSI system designers today.

Provide self-testability on very complex structures. Microprocessors and other related devices should contain testing algorithms in their *microcode store.*

6.3.2 After ICs are made - systematic testing

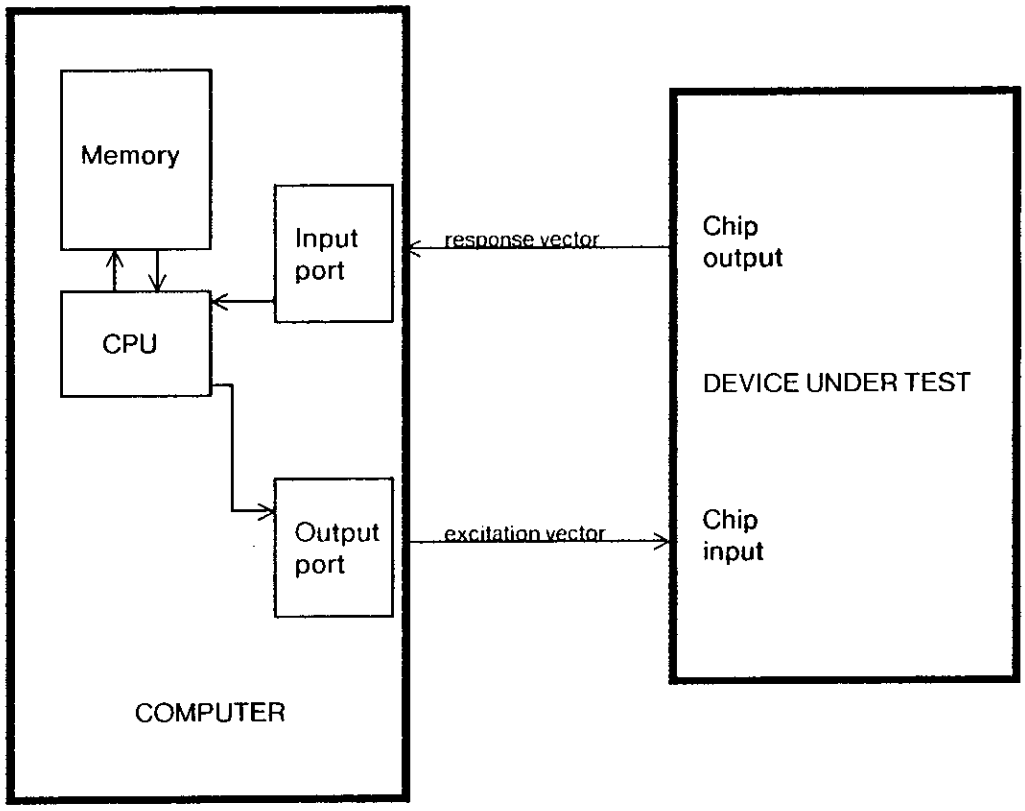
The first and most burning question is: *does the chip work?* When it doesn't, it is often possible to find the problem by looking at the chip under a microscope or by studying photomicrographs of it. If the chip is functional, one should test its performance and reliability (speed and power dissipation, for example).

Today's IC's are rarely simple enough to allow manual testing. Circuit complexity, the large number of inputs and outputs, and the multitude of possible states make manual testing prohibitively time-consuming. A better solution is to have a computer perform the tests (see figure 6.3.1). The IC is exercised by applying a properly defined *excitation vector,* provided by the computer's *output port,* to the inputs of the circuit under test. The *response vector* of the device under test can then be read by the computer's *input port* and compared to a stored correct response. The computer could be programmed to respond with an error message upon detecting deviation from the expected pattern. The test patterns must be carefully selected if they are to supply any information about the nature of the error. A more intelligent program may even do suitable branching dependent on the outcome of a few preliminary tests.

The excitation vector may need to be wider than the typical 8 or 16 bits in a word of the host computer; if so, use *multiplexing* to assemble successive words end-to-end in a vector of latches. Those latches should be contained on an interface board together with drivers and connectors.

The method just described requires a minimum of hardware in addition to the computer. Such a software-based testing system, however, might not be fast enough to capture some quickly

Figure 6.3.1
Software approach to IC testing. A computer is dedicated to exciting the device under test (DUT) and collecting the response. The disadvantage of this method is that it is slow and wasteful of computer time. Note that the computer is directly connected to the device under test.



changing response vectors, or it may have trouble exercising a complex device such as a microprocessor which may require some minimum data/clock rates for proper operation. Testing the speed performance of an IC also requires vector speeds which often can not be provided by a simple software system.

Higher I/O vector speeds can generally be obtained by moving more functions from software into hardware. Such a tester can use a semiconductor memory to store all excitation vectors. A counter is used to sequence the memory through the required words. The response vectors are simultaneously captured by another memory, and can later be analyzed at a slower rate. The tester is now a *peripheral* device to the host computer (see figure 6.3.2). The advantage of this method is two-fold; it allows higher testing speeds and frees the computer to perform other tasks while testing is in progress.

One would like a tester to be flexible enough to test any conceivable digital IC. If we imagine the excitation vector for the DUT to be a set of *control words* emanating from a computer's *control unit*, and the result vector out of the DUT to act as a *condition vector* to this control unit, we can construct a tester based on the principle of a *microprogrammed* controller. This kind of a tester could be easily adapted to any task by simply changing the *microcode* (see figure 6.3.3). A tester based on this principle can exercise very complex devices due to its inherent ability to make logical decisions based on some of the results. When the test is concluded, relevant results are as before stored in a result RAM, and the host computer is signaled to fetch them.

Almost any computer or microcomputer can be used as the host. The only requirement is that it have an accessible I/O port. The control unit for the tester could be built using one of the fast bipolar *bit-slice* microprocessors.

It should be emphasized that preparing for the day when the wafers come back from the fab line may be as large and complicated a task as the original design. The proper custom made interface board between the DUT and the test system has to be built and the test routines have to be written. In preparing these routines the designer should keep in mind the possibility that the chip does not work at all and plan a strategy to deal with this case.

In summary, testing is the responsibility of the designer and should be kept in mind from the early stages of the design process to the day of the delivery of a finished product to a customer. The availability of quick turnaround IC implementation permits large integrated systems to be designed modularly and hierarchically, somewhat like programmers now design large software systems. It is

Figure 6.3.2
Hardware approach to IC testing. The computer initializes an IC Tester which is connected as a peripheral device. The sequencer (counter) steps both RAMs, sending an axcitation vector and collecting the result vector. When the test is over, the sequencer interrupts the computer. This method is fast and requires little CPU time. Note that the computer is no longer directly connected to the DUT.

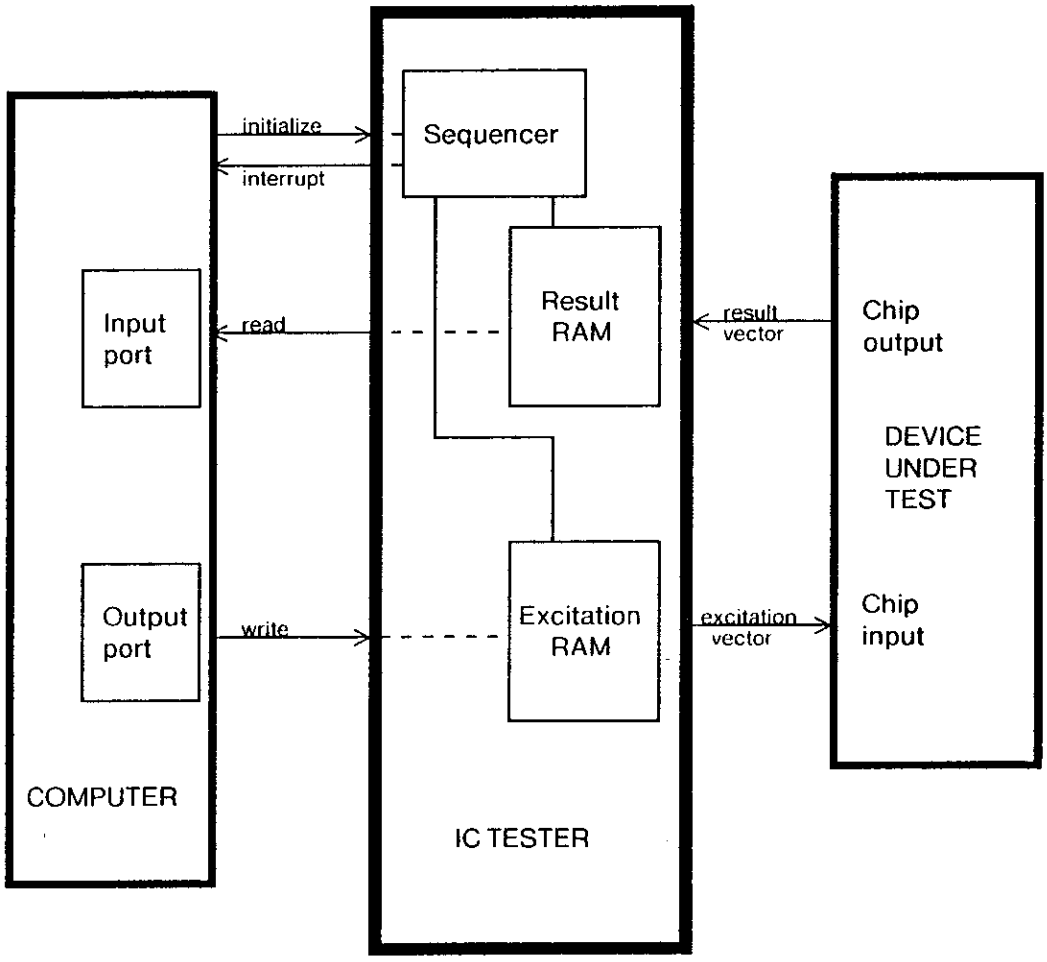
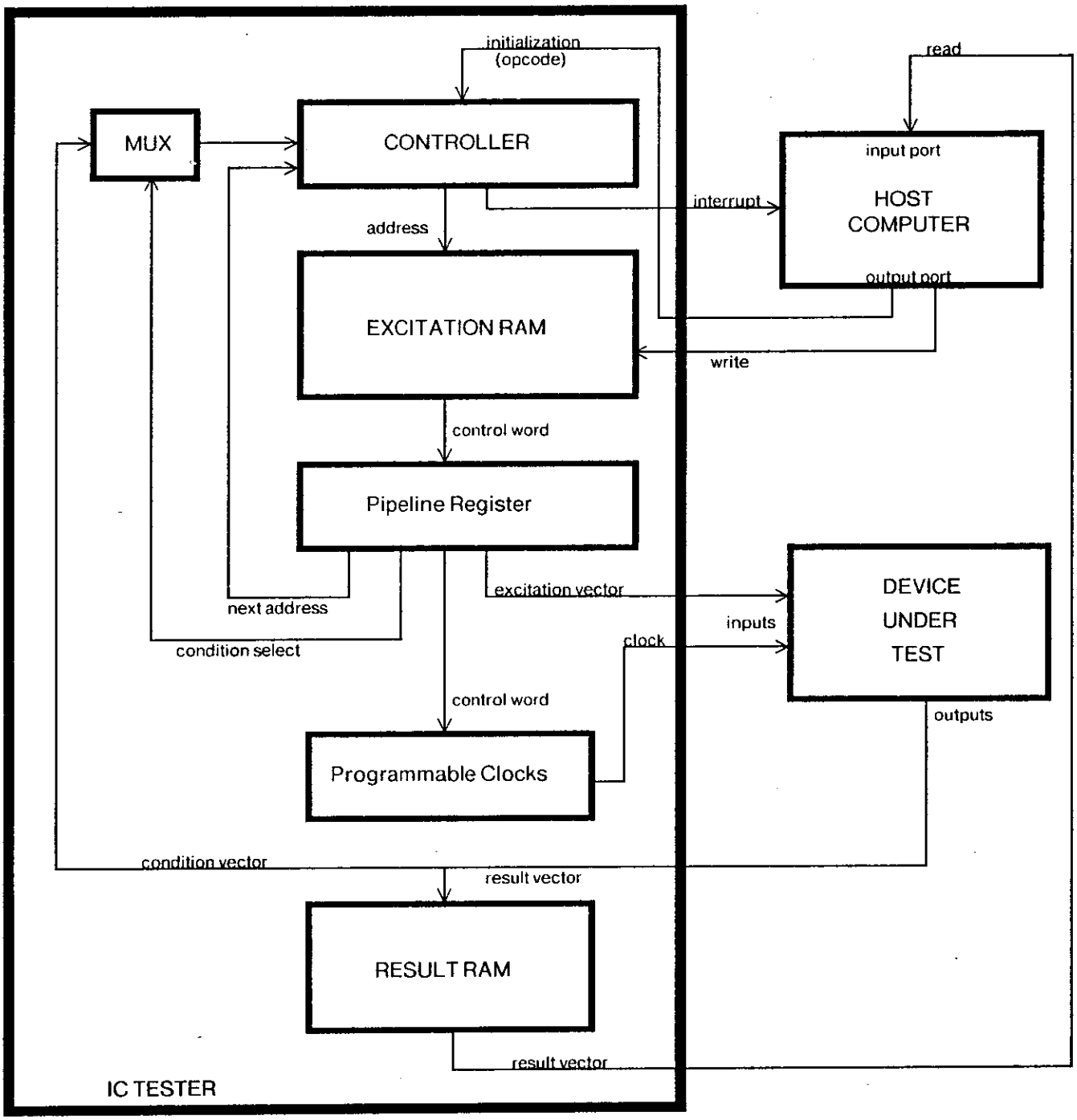


Figure 6.3.3
Block diagram of a Microprogrammed IC Tester. This approach allows maximum flexibility by being programmable. The computer initializes the controller which outputs the address of the first microinstruction to the microprogram RAM. The microprogram RAM supplies the next address and enables the controller to sequence through various testing microsubroutines.



clear that this iterative design loop will be closed when IC designers possess the necessary tools and practical knowledge for effectively testing the LSI modules they design.