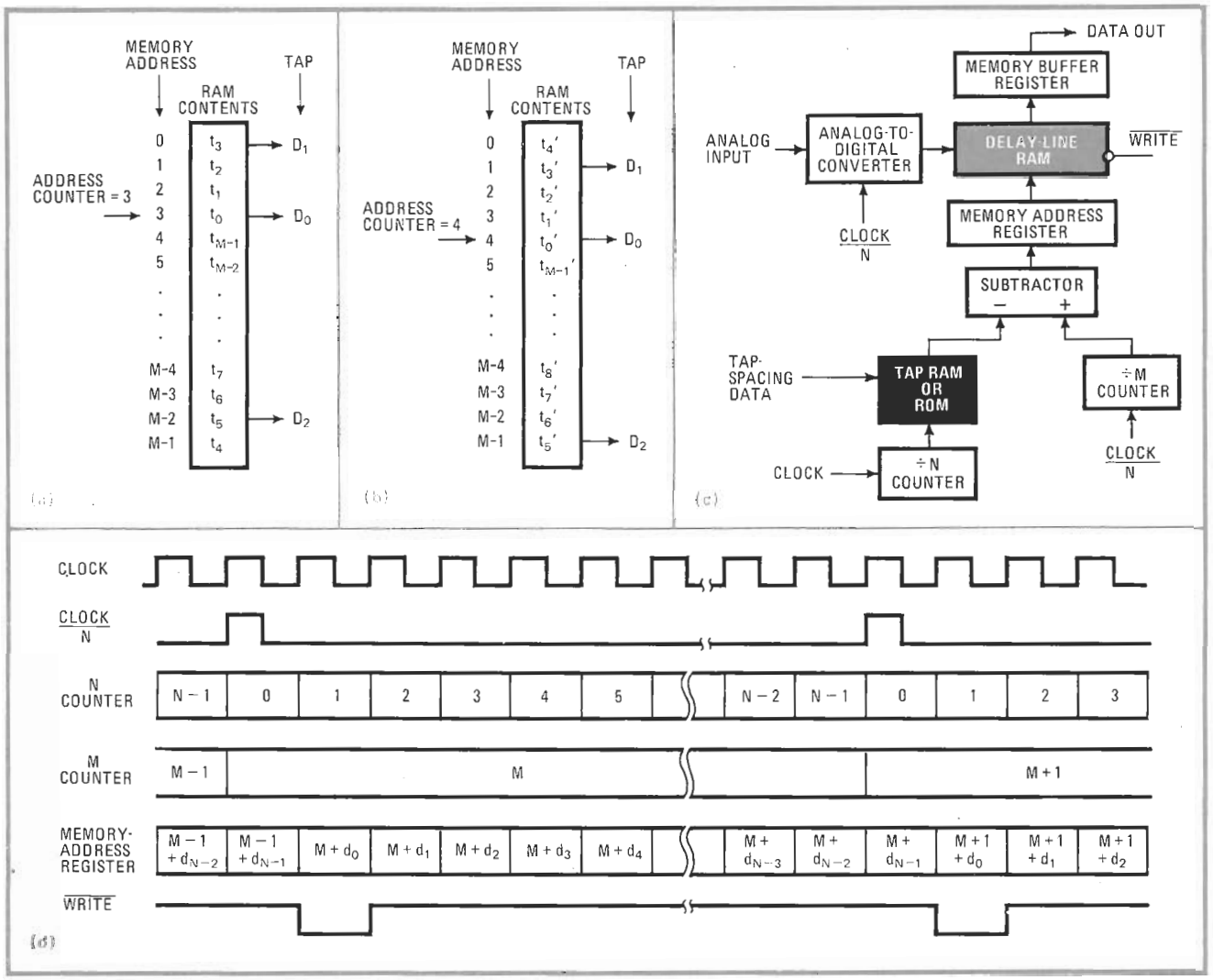
RAMs reduce chip count in programmable delay lines

by Scott M. Smith
University of Texas, Applied Research Laboratories, Austin, Texas

First-in, first-out buffers or variable-shift registers are most often used for the storage elements in digital programmable-tap delay lines (that is, one or more shift registers with multiple-output taps). But random-access memories can store a greater number of samples per integrated circuit and can therefore be used to reduce the total device count. A delay line that uses RAMs will cost much less than its FIFO or variable-length register counterparts if the total number of samples handled is fairly large.

Quite unlike a standard shift register, in which input data is introduced at its standard-input port (first location) and then shifted through, a RAM must have its input data introduced at each individual location. The reason is obvious: the contents of the RAM cannot be shifted, but merely accessed by the system's address counter. Therefore, input data must be entered into the particular RAM location that corresponds to the present location of the address counter.

The memory map in Fig. 1a shows how a delay line is mapped onto a RAM having a length of M words and yields an insight into the factors involved in designing a practical circuit. Three output taps, D_0–D_2, are desired in this example. D_0 represents the zero-delay tap. The RAM address counter points to location 3, which contains



1. Super-long tapped delay. Memory map shows how an N-tap delay line is mapped onto an M-word RAM (a). Input data may be introduced into RAM by incrementing counter and placing sample there. Oldest data sample is destroyed and existing samples are redefined (b). Block diagram of system outlines procedure used to write data, examine output taps (c). Waveform diagram details timing constraints (d).

bers the statement line, thus simplifying the correction of errors.

Line-by-line translation also provides translation into compact code and immediate execution for test-program statements entered without line numbers. Thus, the operator can program power-supply voltages, and change the states of driver/sensor pins and of the test pin directly from the keyboard as if he or she were adjusting potentiometers and toggling switches on a manual test fixture. This feature is particularly handy when the operator is performing test-program debugging.

The most frequently used PSP Basic keywords are assigned to individual keys on the operator's keyboard. Pressing a key when the CMD key is depressed enters the complete keyword with one stroke. This greatly reduces typing errors and provides the operator with a convenient reference of the available commands.

Reducing the data base

The PSP uses the guided-probe fault isolation technique, in which a series of probing instructions lead the operator back from a failing output pin to the faulty node (IC and pin number). In addition to the test program, software requires two types of information.

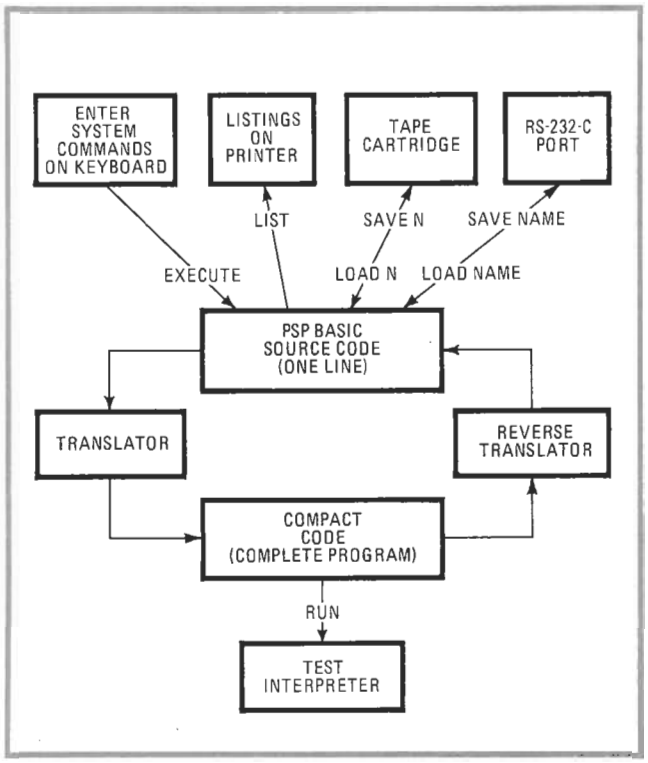
One is a circuit "image": a wire-list description of the interconnections of the components on the board under test. It can be prepared by clerks working from a schematic of the board, or it can be translated from an existing image.

The other necessary data is a table of the expected response at each node. The data base typically used on large production test systems consists of the logic-state data at all tests. Since a large board can have 1,000 or more nodes, and a large test program can contain 1,000 or more tests, the required data base can exceed a million bits and require a disk memory for backup storage of the expected responses. Obviously, this was impractical in the PSP, so some means for compacting the data had to be used.

Techniques for reducing a bit stream of 1s and 0s have been used in board testers for more than six years. One of the most widely used is transition counting, in which the number of logic-state transitions at a node is counted and stored as a signature that represents the correct response at that node. Other widely used techniques measure signatures consisting of 1s-counts, time-weighted transition counts, cyclic-redundancy-check codes, and other codes based upon shift-register sequences.

However, all of these techniques have a common disadvantage when used on complex boards containing feedback loops. The effect of the fault propagates around such loops so that all of the nodes in the loop appear faulty, and fault isolation to a single component is impossible.

The PSP had to be able to isolate faults to a single component even in complex feedback loops, so the simple signature techniques could not be used. Instead, a proprietary data-compacting technique was developed that retains sufficient information about the nodal responses in a form compact enough to fit in the memory. A patent on the technique is pending.



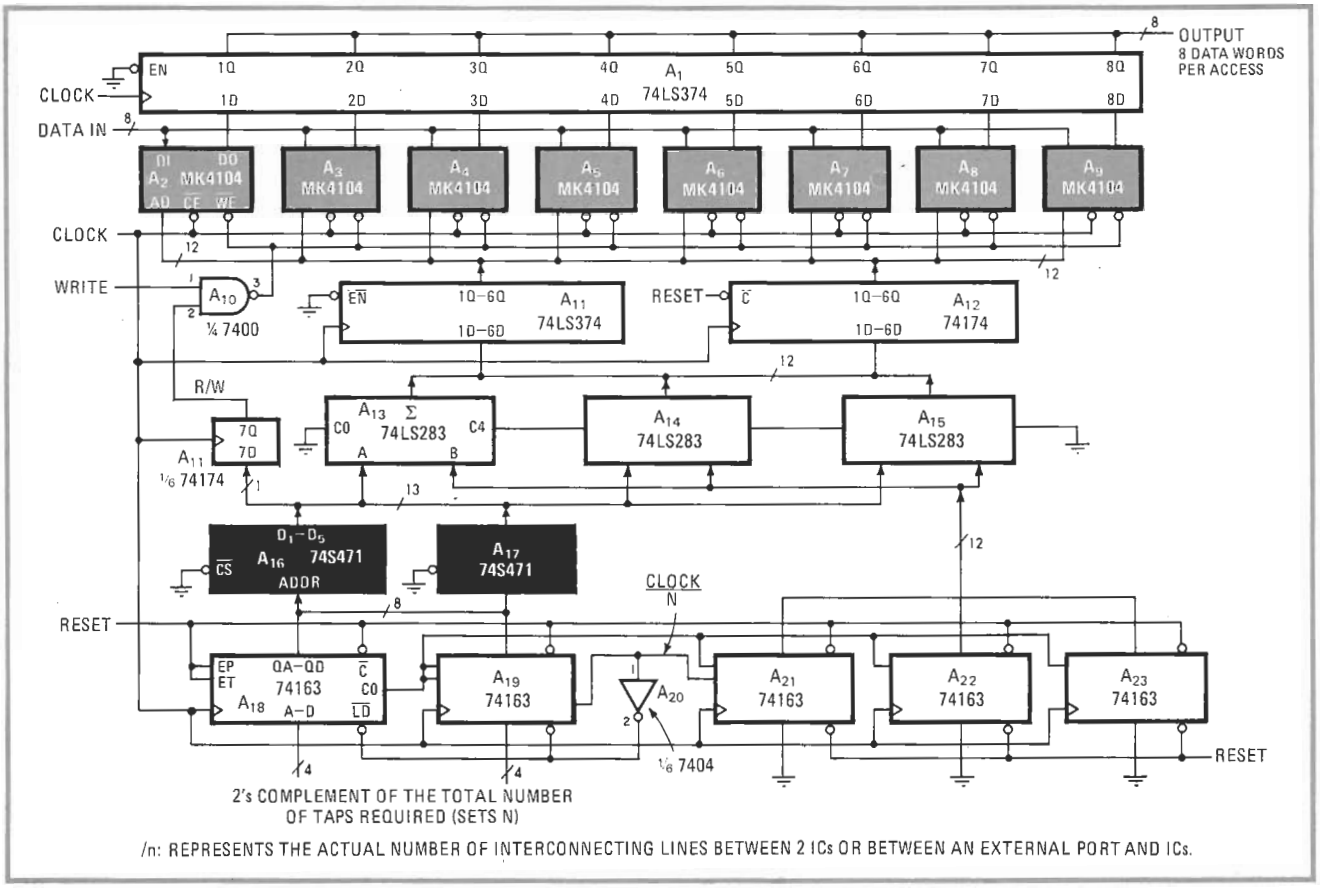
5. Architecture Software is written in microcode, macrocode, and higher-level languages to provide the optimum tradeoffs between execution speed, memory requirements, and user flexibility. Line-by-line translation, compacting, execution, and reversal are performed.

Most circuit boards are tested on automatic production test systems. However, all test systems have unique languages in which their test programs must be written. The PSP's return on investment in field engineering obviously would be reduced if all test programs had to be regenerated for the field tester. Therefore, a series of translator software packages were developed to convert the factory test programs and circuit images into the PSP Basic language.

Translating other languages

These translators, written in Fortran for transportability to whatever computer facilities are most convenient for each company, are presently available to convert the test languages of Computer Automation, GenRad, Micro Systems, and Teradyne. Under development are translators that will handle the test languages of several other commercial and in-house test systems. These test program translations are one of the major activities of the support effort necessary to deploy the PSP in the field.

Pseudorandom test patterns have been used to generate programs for simple and moderately complex boards for many years. Although such patterns are not directly translatable, the PSP does include a pseudorandom-pattern-generation capability that closely emulates the types of patterns generated by the most popular pseudorandom testers. Thus, the information about which types of patterns are to be applied to which pins is sufficient to convert existing pseudorandom test programs for use with the PSP. □



2. Great capacity. Eight-bit-wide programmable-tapped delay line is implemented with Mostek MK4104 random-access memories as shown. Each delay line is the equivalent of a 4,096-bit shift register. User may specify a total of 255 output taps with any desired spacing.

t_0, the most recent sample in the delay line. The next most recent sample is t_1, with t_{M-1} being the oldest sample. D_1 and D_2 are taps delayed three samples and five samples, respectively, with regard to D_0.

The value corresponding to memory address 3 (t_0 = logic 1 or logic 0) would appear at D_0 if that tap were requested. Similarly, the sample at memory address 0 would be fetched if D_1 were to be requested, and the sample at address $M-2$ would be fetched if D_2 were requested.

Figure 1b shows how a new sample would be inserted into the delay line. The counter would be incremented, pointing to location 4 as shown, and the new sample would be written into the location, thus shifting the oldest data sample (t_{M-1}) out of RAM. The memory contents of RAM would otherwise be unchanged; each memory address would be simply redefined as being one sample older. If D_1 were queried, the sample at memory address 1 would be fetched; when D_2 were requested, the sample at location $(M-1)$ would be fetched.

The block diagram shown in Fig. 1c more clearly explains how data is written, and taps are specified and read. A divide-by M counter driven by a clock running at $1/N$ times the system clock frequency is required for pointing to the most recent sample (D_0). Also required is a tap RAM or ROM, which is programmed so that its output is equal to the distance in time (that is, the number of samples) each user-specified output tap is from the most recent sample, t_0. Thus, the spacing between taps is specified. A divide-by-N counter (where N is the number of taps) is needed to address each tap in a sequence that is selected by the user. Note that the N counter must move through one complete cycle for each increment in the M count. The subtractor determines the numerical difference between the tap distance and the zero-delay location and stores the result in the memory address register in order to access the memory address desired. The delay-line RAM is then accessed to obtain the data sample corresponding to the tap selected, or to write in a new data sample. Then the sample that has been read is stored in the memory buffer register, to be shifted out in serial form.

The timing considerations for the circuit are shown in Fig. 1d. As may be observed, provision should be made to ensure that the M counter advances before any new data (write) is stored in RAM, if necessary, to allow the oldest sample to be read before it is overwritten. There are no other major considerations. The taps may be accessed in any order and are selected by appropriate programming of the tap delay memory (tap ROM). The maximum shift rate (the frequency with which new samples are placed in memory) is $f_{s\,max} = 1/Nt_{c\,min}$, where $t_{c\,min}$ is the minimum cycle time of the system.

Figure 2 shows a design example that uses an 8-bit-wide programmable tapped-delay line. The RAM memories, each holding 4,096 1-bit words, form a 4,096-word-by-8-bit array. A_1 is the memory buffer register, A_2–A_9 is the RAM delay line, the memory address register is

A_{11}–A_{12}, and A_{13}–A_{15} is the subtractor. The tap ROMs are implemented by A_{16}–A_{17}. The divide-by-N counter is implemented by A_{18}–A_{20}. Two hundred and fifty-five output taps may be specified. A_{21}–A_{23} is the divide-by-M counter. Note that bit 7 of A_{11} buffers the read/write definition bit from ROM. □