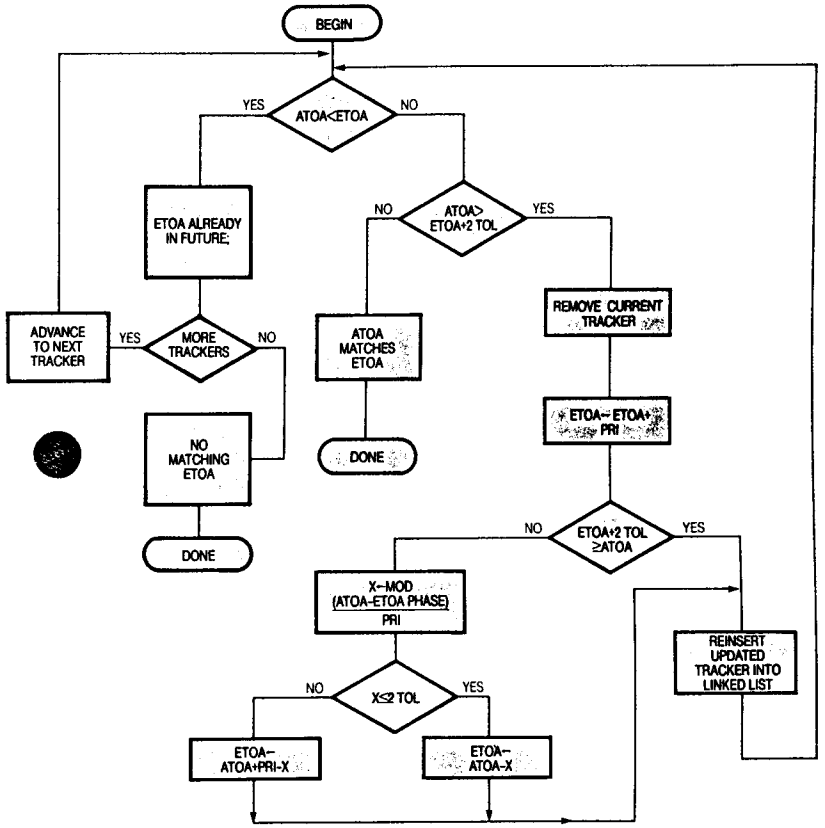
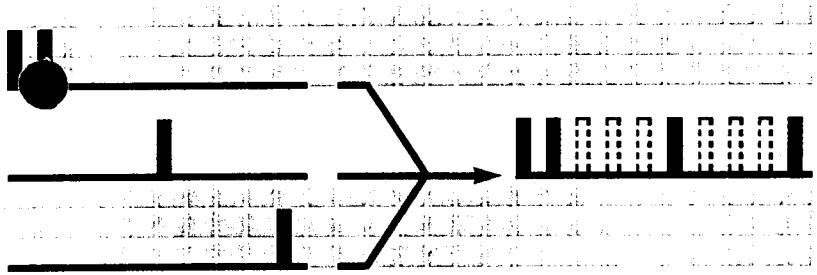
WATKINS-JOHNSON COMPANY
3333 HILLVIEW AVENUE
PALO ALTO, CA 94304
(415) 493-4141

VOL. 13 NO. 3 MAY/JUNE 1986

Time Tracking of High PRI Signals With Bit-Slice Microprocessors





Tech-notes

WATKINS-JOHNSON **Facility Locations**

United States

CALIFORNIA
Watkins-Johnson
3333 Hillview Avenue
Palo Alto, 94304
Telephone: (415) 493-4141

Watkins-Johnson
2525 North First Street
San Jose, 95131
Telephone: (408) 435-1400

Watkins-Johnson
440 Kings Village Road
Scotts Valley, 95066
Telephone: (408) 438-2100

MARYLAND
Watkins-Johnson
700 Quince Orchard Road
Gaithersburg, 20878
Telephone: (301) 948-7550

International

UNITED KINGDOM
Watkins-Johnson
Dedworth Road
Oakley Green
Windsor, Berkshire SL4 4LH
Telephone: (0753) 869241
Telex: 847578
Cable: WJUKW-WINDSOR

ITALY
Watkins-Johnson S.p.A.
Piazza G. Marconi 25
00144 Roma-EUR
Telephone: 592 45 54
591 25 15
Telex: 612278
Cable: WJ ROM I

GERMANY, FEDERAL REPUBLIC OF
Watkins-Johnson
Keferloher Strasse 90
8000 Muenchen 40
Telephone: (089) 35 97 038
Telex: 509401
Cable: WJDBM-MUENCHEN

Watkins-Johnson
Deutschherrenstrasse 46
5300 Bonn 2
Telephone: (228) 33 20 91
Telex: (886) 9522
Cable: WJBN-BONN

The Watkins-Johnson *Tech-notes* is a bi-monthly periodical circulated to educational institutions, engineers, managers of companies or government agencies, and technicians. Individuals may receive issues of *Tech-notes* by sending their subscription request on company letterhead, stating position and nature of business to the Editor, *Tech-notes,* Palo Alto, California. Permission to reprint articles may also be obtained by writing the Editor.

Printed in U.S.A.

The modern radar environment is quite dense and includes intermittent signals from rotating emitters, multiposition staggers, and exotic signals. One useful way of characterizing this environment is by identifying constant PRI (Pulse Repetition Interval) emitters in real time. A method of signal processing known as *time tracking* can provide more accurate identification of constant PRI signals than other methods. With the appropriate hardware, over 200 signals in the millisecond PRI range can be time tracked simultaneously. Lower PRI signals can still be tracked by faster, more rudimentary methods in order to boost total system throughput. This article will explain the time tracking algorithm and discuss some of the details of its implementation with bit-slice microprocessors.

The Need For Time Tracking

In a high-density radar environment, it is often desirable to identify constant PRI signals to reduce the number of pulse reports sent to auxiliary signal processing hardware. Since it will not be bombarded with every pulse report, the auxiliary hardware will have time to perform an extended analysis on the exotic emitters. One method of eliminating the constant PRI trains is by estimating their carrier rfs and, subsequently, masking out all pulses detected within a narrow frequency tolerance of these carrier frequencies. Unfortunately, the environment sometimes consists of a high PRI and a low PRI signal with relatively close rfs.

In this instance, a tracking scheme based solely on carrier frequency would identify one constant PRI train instead of the two actually present. A multiposition stagger would also be reported as a single pulse train.

Frequency-only tracking also has limitations in dealing with signals from rotating emitters. These appear as a short burst of pulses, followed by a relatively lengthy period of inactivity. Based on carrier rf, it is not possible to determine whether these bursts of activity are independent signals, or if they all represent the same emitter. A more desirable tracking scheme would determine whether pulses from the current burst are a continuation of the constant PRI pulse train identified in the previous period of activity.

To more accurately identify the aforementioned signals, a method known as time tracking can be employed. Time tracking involves forming an estimate of the initial time occurence of a pulse, and predicting subsequent occurences

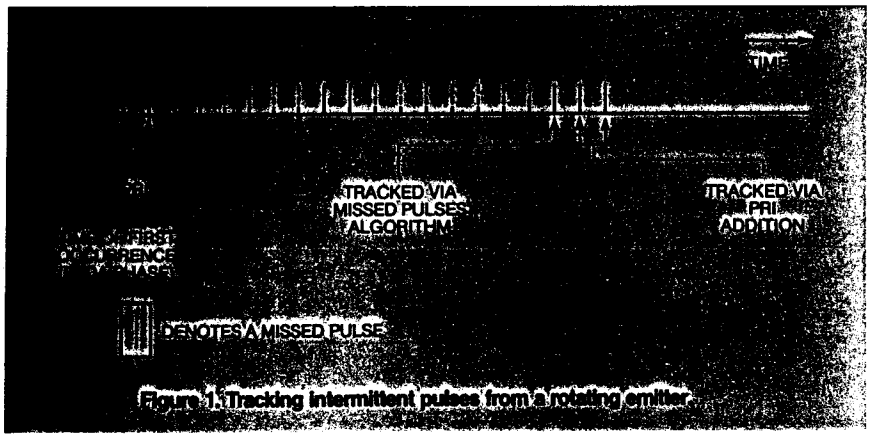


Figure 1. Tracking intermittent pulses from a rotating emitter.

by adding multiples of the PRI to the initial time estimate. An intermittent signal from a rotating emitter can be tracked by determining whether the first detected activity from the current burst is an integral number of multiples of PRI from the time occurence of a signal during the previous burst. Figure 1 depicts time tracking of an intermittent pulse train. Signals in a multiposition stagger can be de-interleaved by forming a different "time-of-first-occurrence" for each discrete pulse train, where all pulse trains contain the same PRI. In Figure 2, the de-interleaving process for a two-position stagger is shown.

The time required to perform sophisticated real-time arithmetic computations limits the use of time tracking for arbitrarily low PRI signals. Fortunately, many rotating emitters are characterized by a PRI large enough to be time tracked with state-of-the-art hardware.

Frequency-only tracking still proves useful for tracking pulse dopplers and other low PRI signals.

The Time Tracking Algorithm

An effective algorithm for implementing time tracking in a "real-world" environment entails much more than simply adding the PRI to the ETOA (Estimated Time of Arrival) of the tracker. Since the PRI of a signal often falls in between two quantized intervals of time, the PRI addition must be performed in an extended format to prevent a rapid divergence between the ETOA and the Actual Time of Arrival

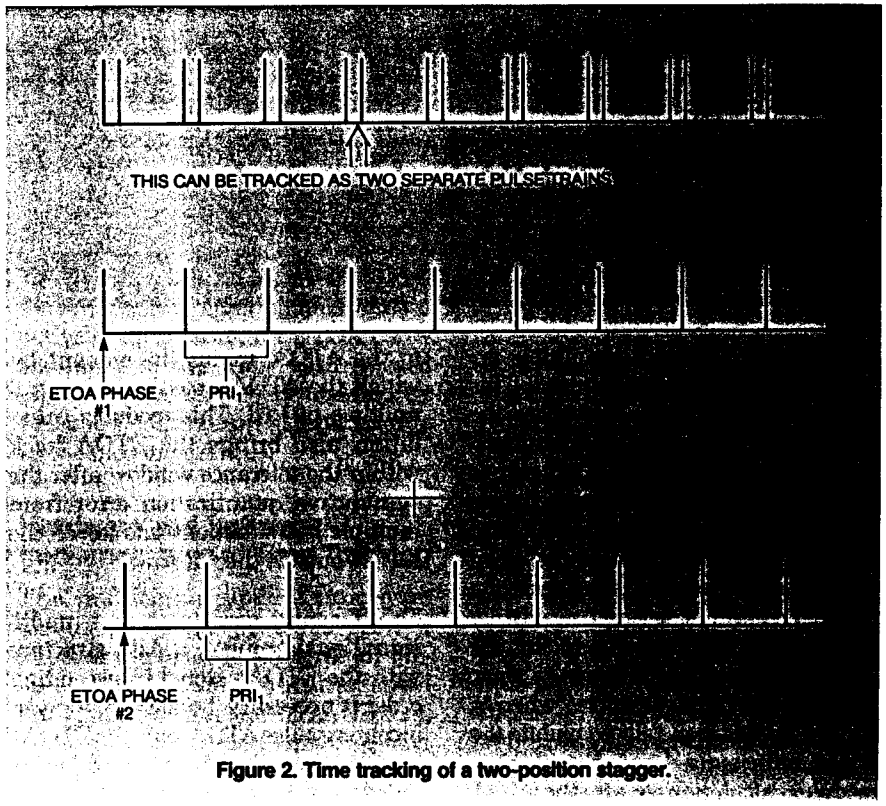


Figure 2. Time tracking of a two-position stagger.

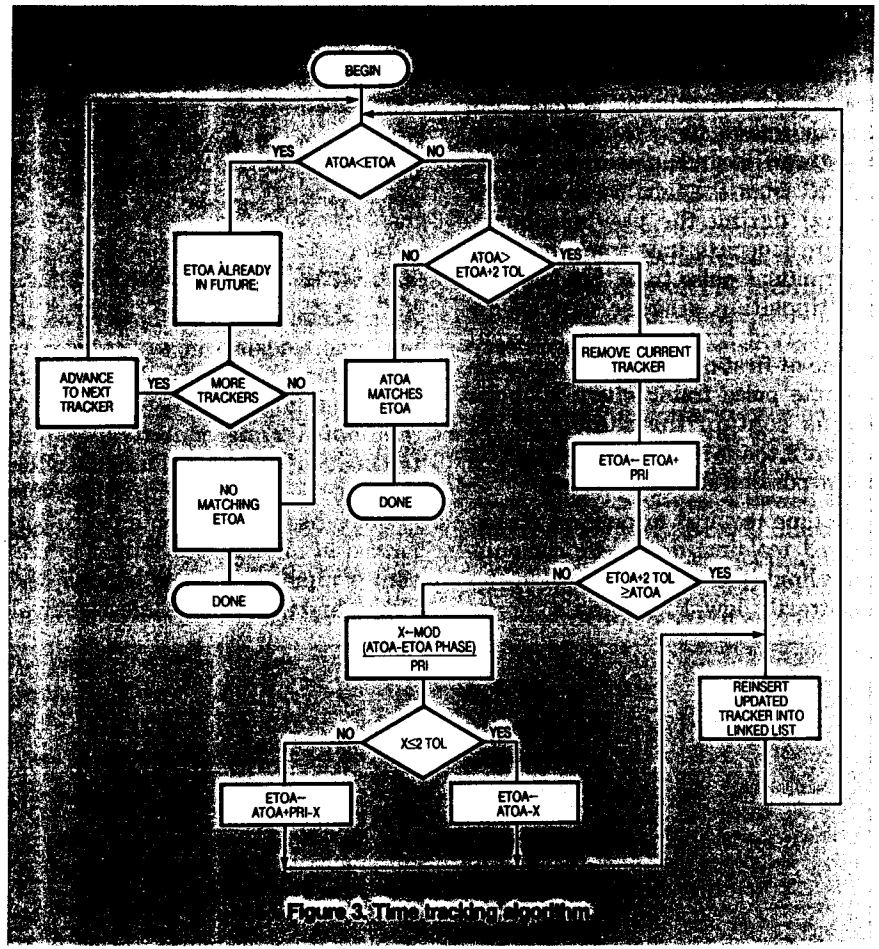


Figure 3. Time tracking algorithm

(ATOA) of a pulse after multiple updates have occurred (see Figure 3). This numerical inaccuracy, as well as the PRI jitter inherent in most emitters, necessitates the use of a tolerance "window" when determining whether the new ETOA matches the ATOA of the incoming pulse. Furthermore, when the first pulse from the latest scan of a rotating emitter is detected, repeatedly adding PRI to compensate for the "missed" pulses is prohibitively slow. To overcome this obstacle, when a single PRI addition fails to update the ETOA sufficiently close to the ATOA, a modulus calculation equates the ETOA to the ATOA, minus the amount by which the ATOA exceeds an integral multiple of PRI. This modulus calculation also brings the ETOA back within the tolerance window after the cumulative quantization error from multiple PRI additions has forced the ETOA outside the window.

To efficiently time track in a multisignal environment, a data structure with the ETOAs stored in ascending order is necessary, because for a given incoming pulse ATOA, only the ETOAs smaller than this ATOA need be updated for a potential match. The smallest ETOA must be removed from the data structure, updated, and reinserted in the appropriate location (possibly a different one). If this updated ETOA fails to match the ATOA, the next smallest ETOA must be removed, updated, and reinserted.

This procedure ceases if an ATOA/ETOA match is found; otherwise, it continues until all smaller ETOAs have been updated past the ATOA. The algorithm must also halt if all trackers have been updated without an ATOA/ETOA match occurring.

Updating several trackers to attempt to match a single incoming ATOA may seem inefficient, but the updates for the non-matching trackers are usually not wasted. If four trackers are updated due to an incoming ATOA before a match is found, the first three trackers will match the next periodic occurence of their respective pulse trains without further computation. However, if the next pulse in the constant PRI train is not detected and assigned an ATOA, the corresponding tracker will require at least one additional update. Exactly one extra update, via the modulus calculation, will occur if the next detected ATOA belongs to the same train as the missed pulse. If other signals remain active in the interim, the ETOA corresponding to the train with missed pulses will be repeatedly updated by PRI additions to keep it at least as large as the incoming ATOA.

Hardware and Software Implementation of Time Tracking

The multisignal update procedure discussed in the previous section imposes the following criteria on prospective ETOA data structures: speed and simplicity for removing an element, reinserting it and accessing the next higher ETOA. Two data structures satisfying these requirements are binary trees and linked lists. For inserting an element into an arbitrarily large list of N elements, the binary tree is more efficient; on the average, the correct insertion point can be found by examining ($\log_2 N$) elements versus (N/2) comparisons in a linked list.

However, the linked list is easier to implement in real-time hardware, and requires less time for changing pointers. In a computer simulation of a typical radar environment, the linked list provided a smaller average reinsertion time when fewer than 60 signals were being time tracked.

In the hardware implementation of the linked list illustrated in Figure 4, a dedicated memory board stores "tracking blocks" in an arbitrary order. These blocks contain the most recent ETOA, the PRI, the initial time estimate (ETOA) phase), and information useful for frequency tracking. At the address of the ETOA currently being updated or compared, a separate pointer memory contains the address of the next higher ETOA. A zero end-of-list marker is written into the pointer RAM at the address of the largest ETOA. An address comparator forces the ETOA comparison procedure to cease if this marker is encountered.

Achieving the signal processing rate discussed in the introduction also requires a high-speed ALU controlled by a hardware-state machine or a microprocessor. The microprocessor-based approach simplifies the process of making the numerous control logic changes normally associated with a new algorithm. During the prototype state, a development station can be used to efficiently test sections of the time-tracking procedure. In addition, the use of a microprocessor simplifies implementation of the various signed and unsigned arithmetic comparisons used in the ETOA computation. The

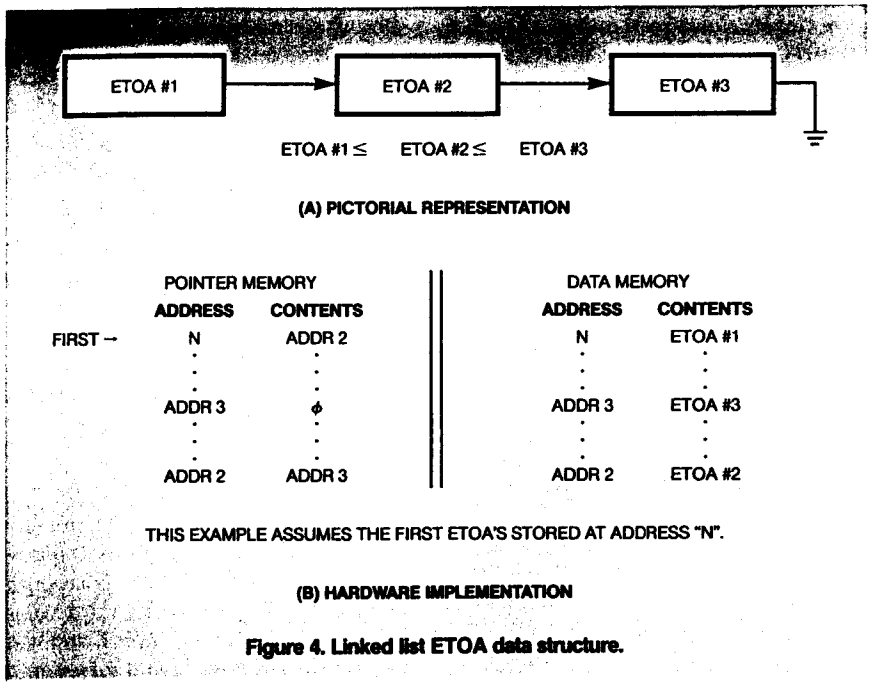


Figure 4. Linked list ETOA data structure.

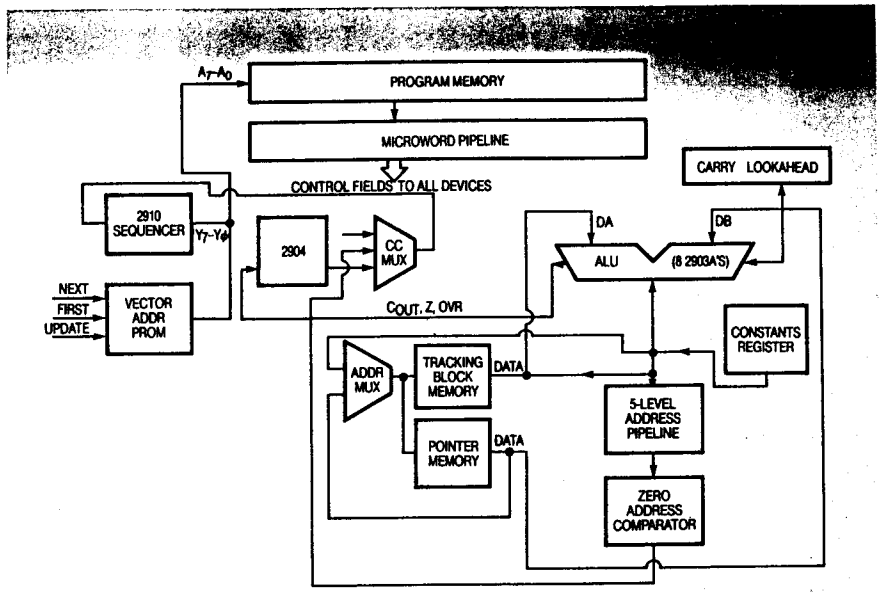


Figure 5. Time tracking processor block diagram.

only caveat is that the chosen processor must posses a short enough instruction cycle and sufficient parallel-processing capability to match the speed of a hardware-state machine.

The 2900 Bit-Slice Microprocessor family provides the best combination of sophisticated arithmetic processing capability, high-speed interfacing to the ETOA data structure, and short instruction cycle time. As shown in Figure 5, this chip set can be used to construct a 32-bit processor which provides 100-nanosecond ETOA resolution. The heart of this processor consists of eight 2903A Arithmetic Logic Unit slices cascaded together. The 2903A's contain a nonrestoring division instruction, which can be used to efficiently implement the modulus calculation for intermittent pulses. As noted previously, this calculation is the key to accurate time tracking. Furthermore, these ALUs contain three independent data busses for interfacing to the ETOA tracking block structure. On a single processor cycle, ETOA data can be read in on one bus (the DA bus), and used as an operand for a result which will appear on the main system Y bus. The third bus (DB) can be used to read in tracker addresses from the linked list without interfering with data on the main Y bus.

The time tracking software is controlled by the 2910 Program Sequencer whose opcodes include the vector jump instruction. With this feature, all requests for linked-list manipulations can be priority-encoded and tested on a single processor cycle. The active request with highest priority determines which jump address is fetched from an auxiliary PROM and loaded into the sequencer. Thus, auxiliary frequency-tracking hardware can efficiently obtain the first or next tracker address without interfering with the critical path operation, which is performance of an ETOA update.

The flexible software structure of the bit-slice family also lends itself to efficient control documentation of the time tracking algorithm and the linked-list hardware.

Instead of using a predefined assembly language, the designer creates a dictionary of arithmetic operations, jump-address conditions, data sources, and linked-list pipeline controls. The linked-list definitions include fields for latch enables, multiplexer selects for the address pipeline, and read/write selects for the tracking block and pointer memories. Fields from this dictionary are subsequently combined to form custom microcode instructions, and comments indicate the intended function of the specialized hardware controlled by these fields. Figure 6 enumerates the microcode fields used in the time-tracking processor and shows a sample instruction.

A 2904 Status and Shift Control Register connects the most significant ALU slice and the least significant slice under control of a 5-bit opcode. When the modulus is converted from a floating-point to a fixed-point format via a double-precision upshift, the 2904 routes the shifted-out MSB of the floating-point number to the LSB of the fixed-point version. The extended precision PRI addition requires that the carry-out of the most significant slice of the fractional part be routed to the carry-in of the least significant slice of the integer PRI portion.

The 2904 can also be programmed with a four-bit opcode to select the appropriate combination of carry and sign bits for both signed and unsigned arithmetic comparisons. The unsigned comparison is useful when comparing the updated ETOA against the elements of the linked list. Signed comparisons are useful in determining when the fixed-to-floating point conversion of ATOA minus first-predicted ETOA is complete, and testing whether

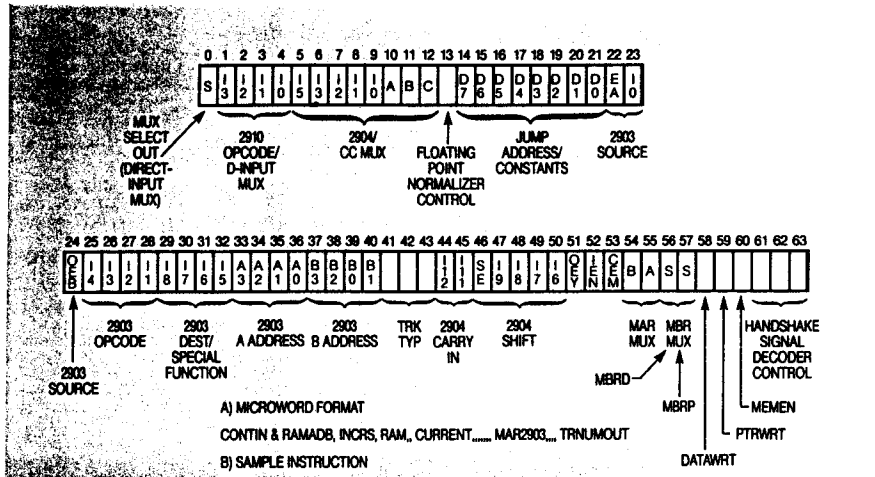


Figure 6. Microcode.

an extra subtraction has occurred at the end of the nonrestoring division procedure.

The use of a full 32-bit carry-ahead generator (54AS882) mitigates the limiting factor in processor operating speed, which is the time required to compute the carry-in for the most significant slice during the non-restoring division instruction. The carry look-ahead generator permits each instruction to be completed in a single 200-nanosecond clock cycle. This cycle time permits an ETOA update via PRI addition, and subsequent reinsertion, to occur in 6 microseconds. When the modulus calculation is performed, the complete process takes 10 microseconds.

The physical separation of the arithmetic functions from the program control circuitry permits a conditional branch instruction to be executed during the same clock cycle as an arithmetic operation. When an updated ETOA is reinserted into the linked list, a conditional jump is permitted, based on whether the ETOA belongs between the N-1st element and the Nth element.

On the same processor clock cycle, an arithmetic operation is performed to ascertain whether the ETOA should be reinstated between the Nth and N+1st elements. A conventional microprocessor would require separate instruction cycles for the conditional jump and the subtraction for the arithmetic comparison.

Additional parallelism (with resultant speed improvement) can be achieved in this time-tracking processor by connecting a five-level hardware tracker address pipeline to the linked-list pointer memory. During the ETOA comparison procedure, to find the correction reinsertion point in the linked list, this five-level pipeline permits access of consecutive ETOAs on successive processor cycles without intervention by the 2903A ALUs. This is accomplished by using the data read from the pointer memory during the current cycle as an address for the next cycle.

In addition, this pipeline enables writing of the new pointer addresses to begin on the first cycle after the correct reinsertion point for the updated ETOA is determined.

To implement the modulus calculation, the difference between the ATOA and the initial predicted time occurrence (ETOA phase), as well as the PRI, must first be expressed in floating-point notation. The difference in exponents of these two parameters determines the number of consecutive cycles of the non-restoring division instruction to be performed. During the first cycle of this division algorithm, the divisor is subtracted from the dividend, and the result is logically upshifted one bit. On subsequent cycles, the divisor is again subtracted from the remaining "partial dividend" if the previous cycle produced a '1' quotient bit; otherwise, the divisor is added back to the partial dividend after shifting. This procedure eliminates the need to conditionally add the divisor back to the partial dividend on a cycle that produced a '0' quotient bit (divisor > partial dividend). The mathematical justification for this procedure is that the partial dividend on the next cycle after a '0' result will be:

$$2 * (\text{dividend} - \text{divisor}) + \text{divisor} = 2 * \text{dividend} - \text{divisor}.$$

In the slower, restoring procedure, the divisor would have to be added back to the partial dividend prior to shifting, yielding:

$$2 * ((\text{dividend} - \text{divisor}) + \text{divisor})) - \text{divisor} = 2 * \text{dividend} - \text{divisor}.$$

On the final cycle of the non-restoring division algorithm, the logical upshift is not performed. If this cycle produces a '1' quotient bit, the partial dividend must be restored by adding the divisor back to it. The partial dividend now contains the modulus in floating-point notation. To convert this to a fixed-point answer, the floating-point number is first moved to the double-precision Q register of the 2903's.

One of the 16 general-purpose registers is selected to receive the carry-out from the Q-register as its carry-in. The divisor's exponent determines the number of logical upshifts performed on the Q register. After these shifts, the aforementioned general purpose register contains the integer part of the modulus, and the Q register contains the fractional part. Note that the fractional PRI divisor required to prevent a rapid divergence between the ATOA and the predicted ETOA's can produce a fractional modulus.

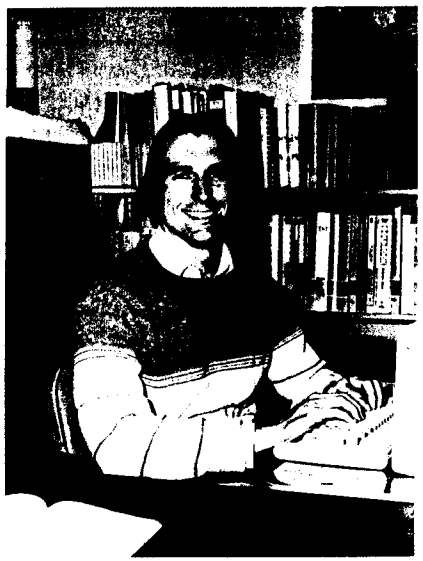
Conclusion

In summary, time tracking can be a useful means of identifying constant PRI signals without masking other signal activity in the same rf range. The bit-slice microprocessors contain the speed and arithmetic complexity necessary to implement time tracking. At the same time, they provide simpler algorithm documentation and data structure control than would be possible in a hardware-state machine implementation.

Glossary of Terms

1. **ATOA (Actual Time of Arrival)** — The time assigned to the leading edge of a detected radar pulse by a free-running clock.
2. **ETOA (Estimated Time of Arrival)** — The predicted time that the next pulse of a periodic pulse train will occur.
3. **ETOA Phase** — The first occurrence of a predicted pulse train. Used as a reference point for predicting the next ETOA of an intermittent train.
4. **PRI (Pulse Repetition Interval)** — The time interval between the leading edge of successive pulses in a periodic pulse train.
5. **Tracker** — The set of parameters characterizing a predicted pulse train. These parameters include the ETOA, ETOA phase, PRI, and carrier rf of the train.

Author:



Douglas Seter

Mr. Seter is a member of the technical staff, ESM Division. He is currently involved in the specification of an architecture for self-testing capabilities in a wide-band receiver system. His responsibilities include hardware design, firmware development, and production management. Other current tasks include the development of control architectures and signal identification algorithms for a new receiver system.

During the past four years, Mr. Seter's primary responsibility has been bit-slice microprocessor design and microcode development. He has also participated in digital hardware and software design necessary to integrate the units of a large receiver system.

Mr. Seter received his B.S.E.E. from Northwestern University and his M.S.E.E. from Stanford University. He is a member of IEEE, Eta Kappa Nu, Tau Beta Pi, and was the 1980 recipient of the Honeywell Award at Northwestern University.