

NRL Report 8750

# Digital Correlator Speed Improvement by Multiplexing

L. M. LEIBOWITZ

*Identification Systems Branch  
Radar Division*

September 30, 1983



**NAVAL RESEARCH LABORATORY**  
Washington, D.C.



20. ABSTRACT (Continued)

approaches to time division multiplexing of digital correlator devices, whereby  $k$  operations are overlapped in time, are developed in this report. Where synchronization with the input signal exists, the resulting multiplexed correlators provide  $k$ -times speed improvement over individual correlator devices. A completely synchronized correlator can be formed from  $k$  such multiplexed correlators by proper selection of outputs. A simplified realization is presented that only requires one set of  $k$   $N/k$ -bit signal and reference registers and  $k$  sets of  $N$ -bit comparator circuits. Laboratory experimentation and computer simulations to support these concepts are described.

## CONTENTS

INTRODUCTION .....	1
BACKGROUND .....	1
TIME DIVISION MULTIPLEXING .....	2
MULTIPLEXED DIGITAL CORRELATORS .....	3
Serial Load .....	3
Parallel Load .....	7
SYNCHRONIZATION .....	9
Serial Load .....	9
Parallel Load .....	12
SIMPLIFIED REALIZATION .....	16
SUMMARY AND CONCLUSIONS .....	18
ACKNOWLEDGMENTS .....	19
REFERENCES .....	19
APPENDIX A—Experimental Circuitry .....	20
(Coauthored by D. Garfinkel)	
APPENDIX B—Software Simulations .....	28
(Coauthored by D. Garfinkel)	

## DIGITAL CORRELATOR SPEED IMPROVEMENT BY MULTIPLEXING

### INTRODUCTION

A digital correlator is a device capable of detecting the presence of a replica of a finite length reference binary code sequence in a relatively long signal sequence of bits. These devices have many applications such as in signal processing, code synchronization/detection, and error correction coding. Correlation provides for the recovery of base band information in proposed applications of spread spectrum techniques to future target identification interrogation/reply systems. High rate pseudorandom binary codes may be utilized to spread transmission bandwidth and reduce spectral energy density in order to improve transmission security and resistance to interference and jamming [1,2]. The application of time division multiplexing to the speed improvement of digital correlator devices is considered here.

### BACKGROUND

Digital correlators have desirable properties with respect to reliability, maintainability, cost, and the application of high-scale integrated circuit techniques. They are capable of correlating very long reference sequences without degradation in signal levels. One of the factors limiting their application has been their low speed or bandwidth capability relative to other technologies such as surface acoustic wave (SAW) devices [3].

A simplified logic diagram of a digital correlator is shown in Fig. 1. An  $N$ -bit reference sequence is serially loaded into  $N$ -bit Reference Register R. A signal sequence is then clocked into  $N$ -bit signal Register S. Each pair of corresponding bits  $s_i$ ,  $r_i$  in the two registers is compared by means of an equivalence gate (exclusive-OR gate with inverter). A logical 1 output is produced only when the bits  $s_i$  and  $r_i$  are in exact agreement. The results of the comparisons of each bit position are summed in

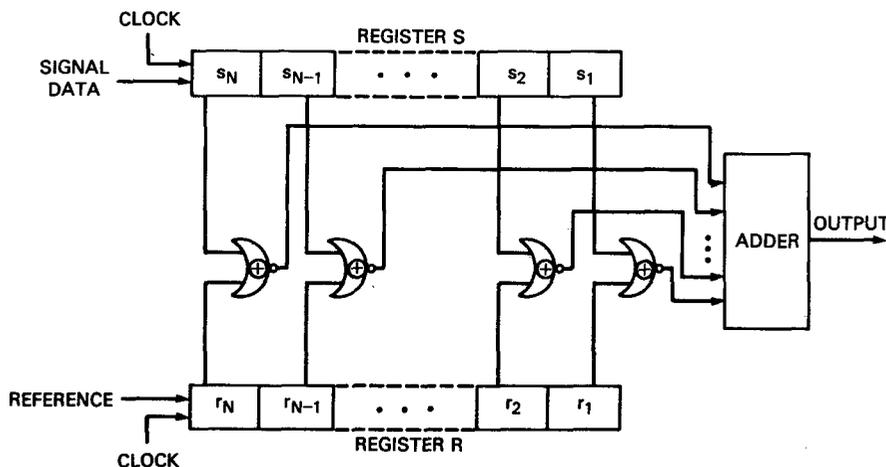


Fig. 1 — Digital correlator

analog or digital form to produce the correlator output. The output corresponds to the number of bit agreements  $A$  in the  $N$ -bit signal and reference sequences. In applications such as spread spectrum communications a more desirable measure of correlation is the difference ( $A - D$ ) between bit agreements ( $A$ ) and disagreements ( $D$ ) [1]. Since the length of the sequence is  $N$ , the number of bit disagreements is  $N - A$  and thus

$$\begin{aligned} A - D &= A - (N - A) \\ &= 2A - N. \end{aligned}$$

Therefore  $A - D$  can easily be determined from the output  $A$  of a digital output correlator device by simply shifting  $A$  one bit position to multiply by 2 and then subtracting  $N$ . By scaling and level shifting, a similar result can be obtained for analog output devices. In most instances the determination of  $A$  or  $A - D$  is not the important result. What is usually desired is an indication of the detection of the occurrence of a replica of the reference code  $R = r_N r_{N-1} \dots r_2 r_1$  in the signal sequence. This is accomplished by applying the correlator output, either  $A$  or  $A - D$ , to a threshold circuit (digital or analog as appropriate). The proper threshold setting is determined by probability of detection and false alarm considerations.

Representative of current digital correlator technology are two 64-bit digital correlator devices produced by TRW LSI Products [4,5]. The TDC 1004J produces an analog output representing the correlation sum and is capable of a 15 MHz rate. The TDC 1023J produces a seven-bit digital output and can operate up to 20 MHz. The TDC 1023J also contains a register that stores a new reference code for parallel loading to the reference register while correlation continues uninterrupted; it also includes a digital thresholding circuit. Both devices provide a mask register that allows the user to choose "no compare" bit positions. This can easily be used to effectively shorten the length of a correlation and permit, for example, the use of pseudorandom (PN) linear code generator reference sequences of length  $N = 2^n - 1$ .

## TIME DIVISION MULTIPLEXING

It is well known that time division multiplexing techniques can be applied to digital systems in order to increase processing speed. Such multiplexing involves reformatting the hardware configuration so that  $k$  operations overlap in time with a resulting factor of  $k$  improvement in operating speed. These techniques have been used effectively with shift registers, analog to digital converters, and other devices [6,7]. As shown in Fig. 2(a), for example, a shift register of length  $N$  can be implemented with  $k$  shift registers each of length  $N/k$ . The normal register  $CK$  clock at rate  $f_{CK}$  is presented to a ring counter that distributes clock pulses sequentially on  $k$  outputs with the  $i$ th pulse of each sequence of  $k$  clock pulses appearing on the  $i$ th multiplexed clock output as  $CK_i$ . Thus successive clock pulses are presented to successive  $N/k$ -bit subregisters with  $CK_i$  providing the clock input to the  $i$ th subregister. As seen from the timing diagram in Fig. 2(b), data bit  $s_1$  is clocked into subregister 1 by  $CK_1$ , bit  $s_2$  is clocked into subregister 2 by  $CK_2$ , and so on with bit  $s_k$  clocked into subregister  $k$  by  $CK_k$ . The succeeding clock pulse will be  $CK_1$  and will clock  $s_{k+1}$  into subregister 1. Thus the shift registers are successively clocked and will each operate at a reduced rate of  $f_{CK}/k$ . As seen in Fig. 2(a) the composite register output can be formed by AND gating the output of each register with its associated multiplexed clock and combining these clocked outputs in an OR gate. Appropriate delays will generally be required in actual implementations in order to achieve satisfactory operation.

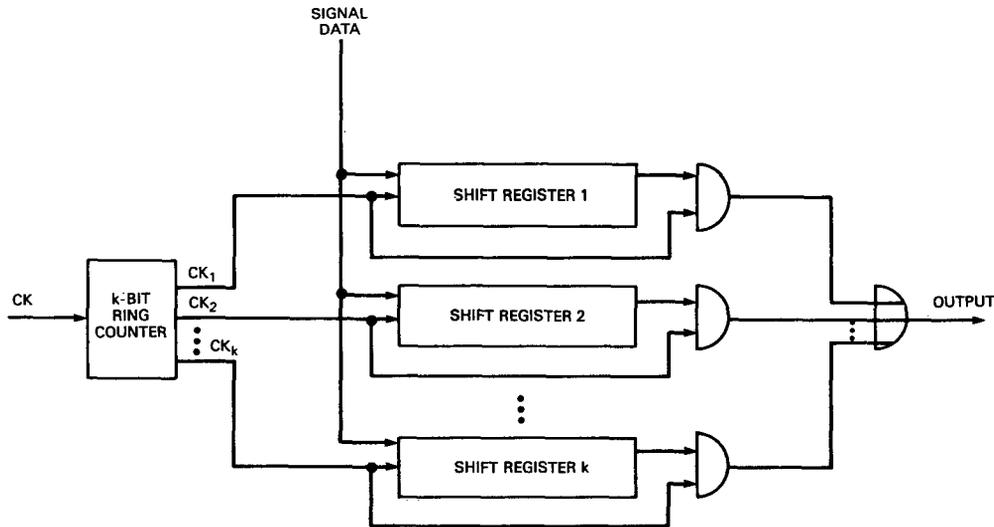


Fig. 2(a) -- Multiplexed shift register

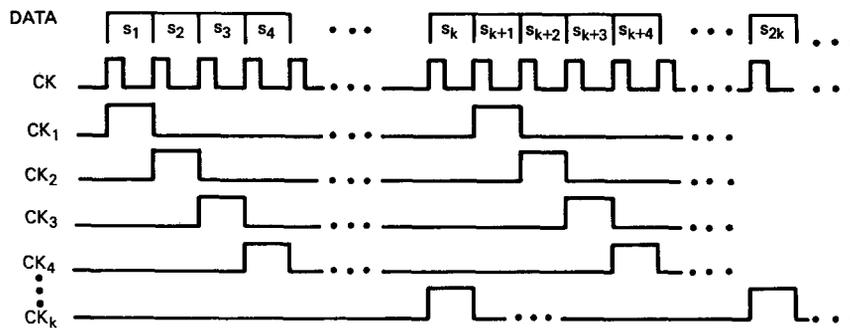


Fig. 2(b) -- Timing diagram for multiplexed shift register

**MULTIPLEXED DIGITAL CORRELATORS**

It is desired to apply such multiplexing techniques to digital correlation in order to accomplish increased operating speed. The TRW LSI Products 64-bit digital correlators discussed above are representative of presently available device technology. Taking a multiplexing approach similar to the shift register example, the resulting correlator would operate on bit sequences of length  $N = 64 \times k$ . To configure an  $N$ -bit digital correlator using  $k$ -times multiplexing involves the use of  $k$ ,  $N/k$  bit digital correlator devices designated here as "subcorrelators." The  $k$  subcorrelator outputs are summed to form the overall multiplexed correlator output. In practice,  $N/k$  will be a power of two. With  $k$  generally chosen for convenience as a power of two,  $N$  will also be some power ( $n$ ) of two. Two multiplexing arrangements, which are presented here, are designated as "serial load" and "parallel load." Although a resulting multiplexed correlator will require additional circuitry, where synchronization with the input signal exists, it can function like a normal  $N$ -bit digital correlator with  $k$ -times improvement in effective speed.

**Serial Load**

The serial load configuration shown in Fig. 3(a) will be discussed first. In this implementation the basic clock signal  $CK$  is presented to a  $k$ -bit ring counter which produces multiplexed clocks  $CK_1, CK_2,$

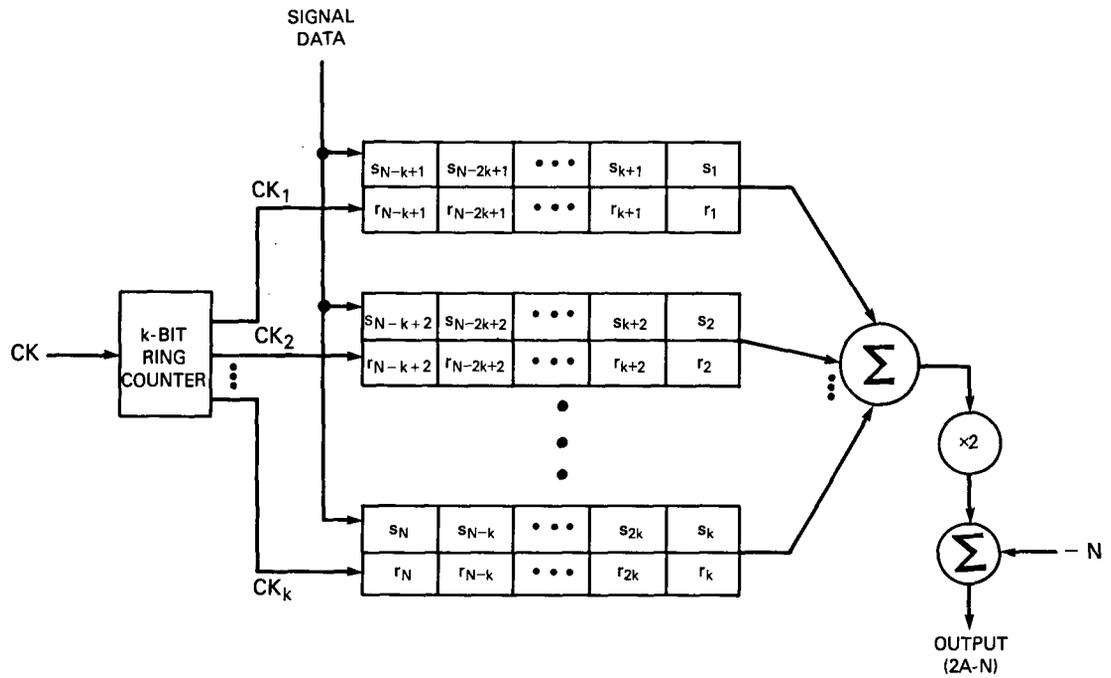


Fig. 3(a) — Serial load multiplexed correlator

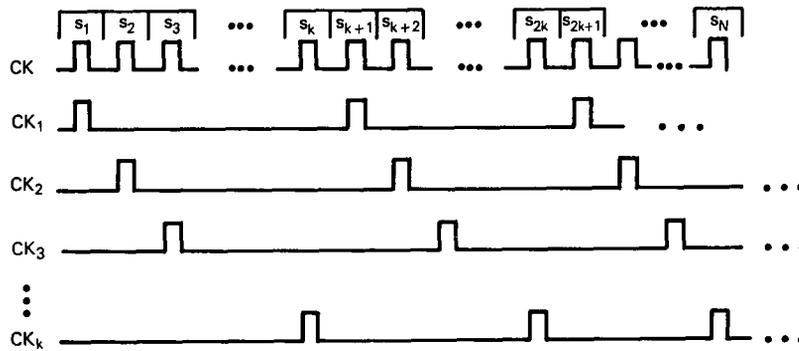


Fig. 3(b) — Timing diagram for serial load multiplexed correlator

... ,  $CK_k$  as shown in Fig. 3(b) with clock  $CK_i$  applied to the clock input of subcorrelator  $i$  ( $SC_i$ ). The signal data sequence is applied in common to the signal inputs of all  $N/k$ -bit subcorrelators. The first signal bit  $s_1$  is clocked into  $SC_1$  by  $CK_1$ . The second signal bit input  $s_2$  is clocked into  $SC_2$  by  $CK_2$  and so on with  $s_k$  clocked into  $SC_k$  by  $CK_k$ ,  $s_{k+1}$  clocked into  $SC_1$  by  $CK_1$ , and  $s_N$  eventually clocked into  $SC_k$  by  $CK_k$ . After  $N$  pulses of clock  $CK$  the  $k$  subcorrelators contain  $N$  bits of the signal sequence. The reference sequence is previously loaded in a corresponding fashion with  $r_{N-k+1}, r_{N-2k+1} \dots r_{k+1} r_1$  loaded into the  $R$  register of  $SC_1$ ,  $r_{N-k+2} r_{N-2k+2} \dots r_{k+2} r_2$  loaded into the  $R$  register of  $SC_2$ , etc., to form a  $k \times N/k$  matrix  $[Ro]$  of the contents of the reference registers of the  $k$  subcorrelators. A match will occur if the signal sequence is an exact replica of the reference sequence. In that case the output of each subcorrelator will be  $N/k$  and will produce an overall multiplexed correlator output sum of  $N = 2^n$ .\* It is important to carefully choose reference sequence codes that have good autocorrelation/crosscorrelation properties. Such codes will generally be of length  $2^n - 1$  and will require application of a mask register to account for the additional bit resulting from available power-of-2 register lengths.

The autocorrelation function  $\theta(m)$  is the number of bit agreements less disagreements ( $A - D$ ) as a function of  $m$ , which is the number of bit shifts between a reference code sequence and a replica of itself contained in an input signal sequence. At  $m = 0$  the reference code replica in the signal sequence contained in the  $S$  register agrees with the reference sequence, and the correlator output is  $N$  as described above. For  $|m| \leq k$  each positive or negative unit shift of the correlator about  $m = 0$  corresponds to a unit shift in a subcorrelator and ideally eliminates a match of  $N/k$  bits thus reducing the multiplexed correlator output sum by  $N/k$ . For  $|m| > k$  the correlator output is zero for a code with ideal autocorrelation properties. The ideal autocorrelation function of the serial load correlator can therefore be described as follows:

$$\theta(m) = 0, \quad |m| > k$$

$$\theta(m) = N \left[ 1 - \left| \frac{m}{k} \right| \right], \quad |m| \leq k.$$

An example of the serial load multiplexed digital correlator for  $N = 16$ ,  $k = 4$  is shown in Fig. 4(a) with the corresponding clock timing diagram indicated in Fig. 4(b). The ideal autocorrelation function for this case is shown in Fig. 4(c). For the example of Fig. 4(a) it can be seen that each of every four clock pulses is directed to a different subcorrelator which thus accepts every fourth signal bit. Although signal data are being presented to the correlator at the rate  $f_{CK}$ , each subcorrelator actually operates at a rate  $f_{CK}/4$ .

\*Using digital output correlator devices, subcorrelator outputs would be summed using  $k - 1$  binary adder stages. The output of  $SC_1$  would be added to that of  $SC_2$ , and the resulting sum added to  $SC_3$ , and so on with the sum of  $SC_1$  through  $SC_{k-1}$  added to  $SC_k$ . Since only the  $SC_k$  output changes in the clock interval subsequent to  $CK_k$ , the peak correlation sum can be formed with a single adder delay time. Adders must have sufficient bit capacity with a  $(1 + \log_2 N/k)$ -bit adder in the first stage and a  $\log_2 N$ -bit adder in the  $(k - 1)$ th stage.

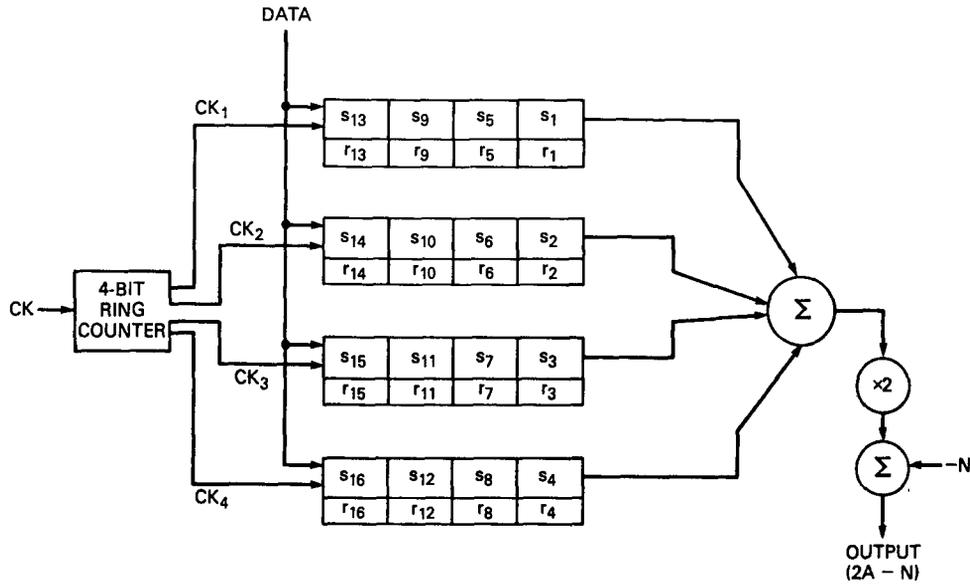


Fig. 4(a) — Serial load multiplexed correlator for  $N = 16, k = 4$

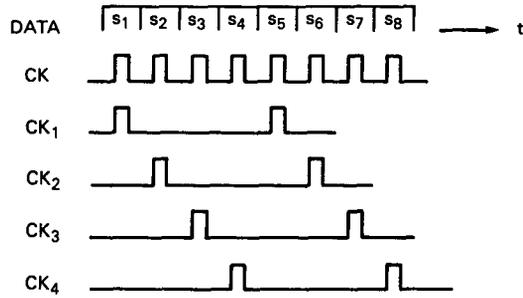


Fig. 4(b) — Timing diagram for serial load multiplexed correlator for  $N = 16, k = 4$

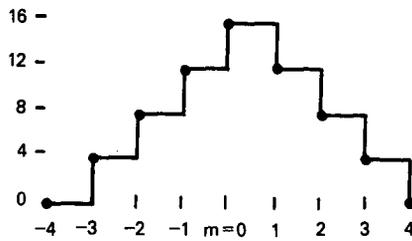


Fig. 4(c) — Ideal autocorrelation function for serial load multiplexed correlator for  $N = 16, k = 4$

**Parallel Load**

The parallel load approach to multiplexing digital correlators is shown in Fig. 5(a). In this case serial signal data at a rate  $f_{CK}$  is serially shifted into a  $k$ -bit shift register by clock signal  $CK$  at rate  $f_{CK}$ . The basic clock  $CK$  is divided in frequency by  $k$  to form  $CK'$  as shown in Fig. 5(b). The reduced rate clock  $CK'$  (with appropriate delay) is used to parallel load each of the last  $k$  signal data bits, serially shifted into the  $k$ -bit shift register, into the  $S$  register of each of the  $k$  subcorrelators. An  $N$ -bit signal sequence thus requires  $N/k$  reduced rate clock pulses to be loaded into the correlator. Thus each subcorrelator is shifted at the rate  $f_{CK}/k$  and receives every  $k$ th signal bit. With every  $N/k$ th reduced-rate clock pulse, another  $k$  bits of the signal sequence are loaded into the correlator to form a new correlation sum while the oldest  $k$  bits previously stored are shifted out.\* If a signal sequence containing a replica of a reference sequence  $R$  with ideal autocorrelation properties is shifted into the correlator, the output will be 0 until an exact match occurs at  $m = 0$ . The ideal correlator output will then be  $N$  and will remain at this level until  $m = k$  when the next  $k$  bits are shifted into the correlator and its output returns to the zero level. Thus the ideal autocorrelation function for the parallel load correlator is

$$\begin{aligned} \theta(m) &= 0, & m < 0 \\ &= N, & 0 \leq m < k \\ &= 0, & m \geq k. \end{aligned}$$

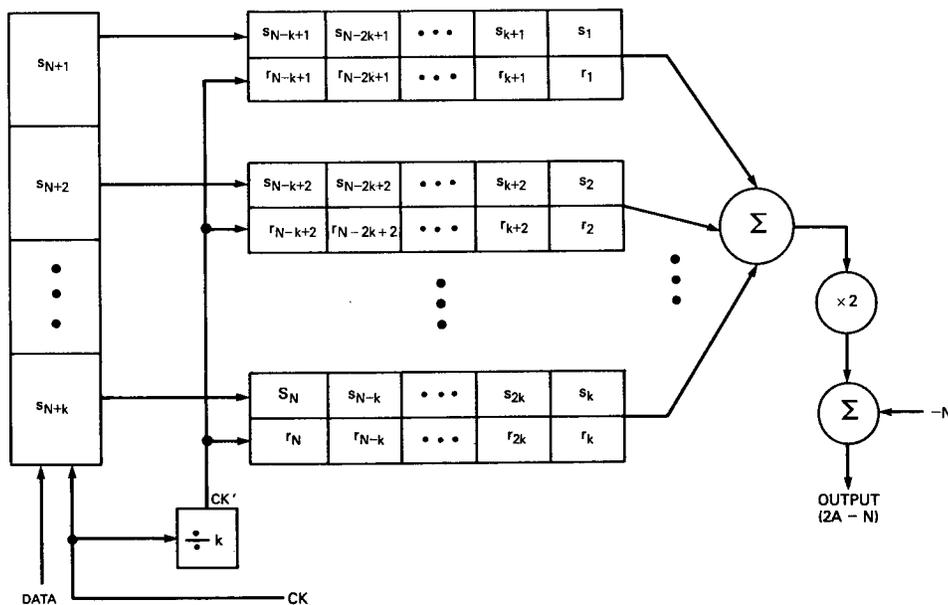


Fig. 5(a) — Parallel load multiplexed correlator

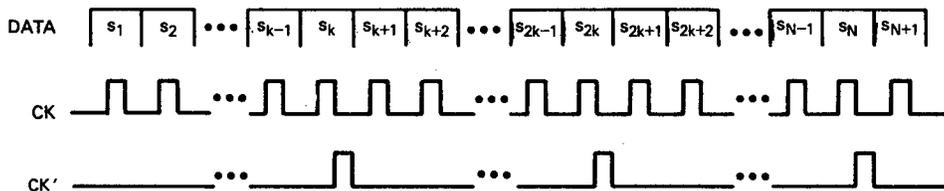


Fig. 5(b) — Timing diagram for parallel load multiplexed correlator

\*Using digital output correlator devices, the subcorrelator outputs would be summed using  $k - 1$  binary adders arranged in a binary tree of  $\log_2 k$  stages (adder delays). In the first stage,  $k/2 (1 + \log_2 N/k)$ -bit adders would operate on pairs of subcorrelator outputs with pairs of adder-outputs summed in subsequent stages until the final sum is formed. The final stage would consist of a single  $\log_2 N$ -bit adder.

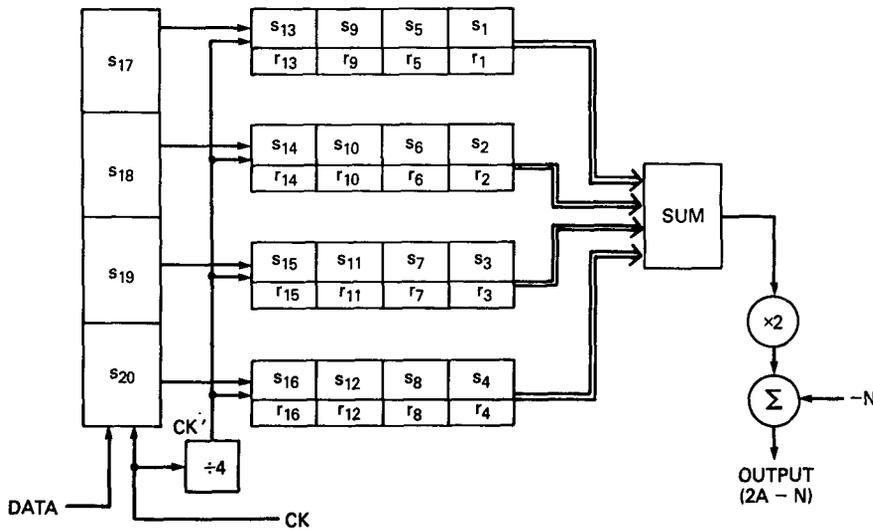


Fig. 6(a) — Parallel load multiplexed correlator for  $N = 16, k = 4$

Fig. 6(b) — Timing diagram for parallel load multiplexed correlator for  $N = 16, k = 4$

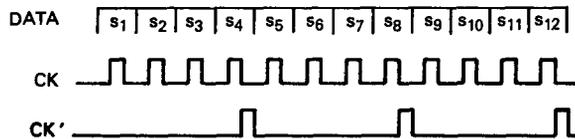
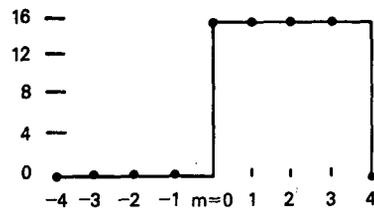


Fig. 6(c) — Ideal autocorrelation function for parallel load multiplexed correlator for  $N = 16, k = 4$



The parallel load multiplexed correlator configuration is shown in Fig. 6(a) for the example case  $N = 16, k = 4$ . The corresponding clock signals are shown in Fig. 6(b). Whenever the latest 16 signal bits shifted into the correlator agree with the reference sequence  $r_{16} \dots r_2 r_1$ , the correlator output is maximum ( $N = 16$ ). The ideal autocorrelation function for this example case is shown in Fig. 6(c).

From the above analysis of the serial load and parallel load multiplexed correlators, it can be seen that they are both suitable for locating a reference code replica contained in a digital signal sequence. They can both utilize digital correlator devices operating at a rate of  $f_{CK}/k$  while throughputting signal data at rate of  $f_{CK}$ . However, both multiplexing approaches as presented require exact synchronization of the signal sequence with the clock for proper positioning in the  $S$  registers in order for autocorrelation to occur. Another disadvantage is a reduction in time resolution compared to a correlator without multiplexing. For the serial load correlator, the correlation peak output occurs only over one clock interval ( $1/f_{CK}$ ) but the correlator output decreases linearly step-wise with  $m$  for  $k - 1$  clock intervals before and after the peak correlation interval. Ideally, the autocorrelation function of the parallel load correlator configuration is at the peak level  $N$  over the interval  $0 \leq m < k$  and zero elsewhere. The parallel load configuration has one advantage due to the step increase in the peak autocorrelation value ( $N$ ) at  $m = 0$  which is easier to threshold on than the smaller step increase ( $N/k$ ) at  $m = 0$  in the serial load configuration. For both the serial and parallel load cases, a multiplexed correlator produces a

correct correlation output, equivalent to that of an ordinary correlator, only every  $k$  intervals of the clock  $CK$ . It will be shown that by the use of  $k$ -such multiplexed correlators, the synchronization and resolution limitations can be eliminated.

## SYNCHRONIZATION

A serious limitation of both these multiplexed correlator techniques is a lack of inherent code synchronization. Thus far we have assumed synchronization with the signal code sequence always properly positioned in the  $S$  registers so that correlation with the  $N$ -bit reference code in the  $R$  registers can take place. This requires an exact relationship between the input signal sequence and subcorrelator clock timing which generally, will not be the case. In both the serial and parallel load techniques, the  $S$  register of each subcorrelator is shifted only once every  $k$  basic clock intervals. Since the clock signals run continuously, the correspondence between subcorrelator clocks and the data sequence will be arbitrary. Thus synchronization in the serial load case occurs only when the first bit of a reference code replica in the signal sequence corresponds to a clock pulse of subcorrelator 1. Synchronization in the parallel load case occurs only when the first bit of a reference code replica in the signal sequence corresponds to the first basic clock pulse following a subcorrelator shift pulse. An input signal sequence can, however, be delayed or advanced relative to the subcorrelator clocks and thus be out of synchronization by from 1 to  $k - 1$  basic clock intervals. A shift in synchronization by exactly  $k$  basic clock intervals is equivalent to synchronization. Therefore, synchronization is determined modulo  $k$  and thus only  $k$  synchronization possibilities need be considered. Being out of synchronization by  $k - d$  clock intervals (advanced) corresponds exactly to being out of synchronization by  $-d$  clock intervals (delayed  $d$  intervals).

To achieve autocorrelation with the reference code replica contained in the signal sequence, we must provide for all  $k$  synchronization possibilities. This requires  $k$  separate multiplexed correlators. We will show that at any time, one of these  $k$  correlators will have the correct correlation output. Furthermore, the overall synchronized correlator output can be formed by properly selecting the outputs from successive correlators at succeeding clock intervals.

### Serial Load

One configuration of a synchronized serial load multiplexed correlator is shown in Fig. 7. A single ring counter serves each of  $k$  individual multiplexed correlators with  $k$  multiplexed clock signals  $CK_1, CK_2, \dots, CK_k$ . Although no actual delay circuitry is used, there is an effective one-count delay between each application of the  $k$  multiplexed correlator clocks to succeeding correlators. This is accomplished by a simple wired circular shift of one position in the order of application of the ring counter outputs to succeeding correlator clock inputs. Thus while subcorrelators 1, 2,  $\dots$ ,  $k$  of correlator 1 are clocked by  $CK_1, CK_2, \dots, CK_k$ , respectively, correlator 2's subcorrelators are clocked by  $CK_2, CK_3, \dots, CK_k, CK_1$ , correlator 3's by  $CK_3, CK_4, \dots, CK_k, CK_1, CK_2$ , and so on with correlator  $K$ 's subcorrelators clocked by  $CK_k, CK_1, \dots, CK_{k-2}, CK_{k-1}$ , respectively. The relationship between subcorrelators and clock signals is shown for the case  $N = 16, k = 4$  in Fig. 8. The  $S$  registers of each correlator will always contain the last  $N$  bits of the input signal. With the one-count delay in the clocks of successive correlators, the  $S$  registers of correlators  $i$  and  $i + 1$  will always contain identical data but the latter will be circularly rotated bottom-up by one register location. Thus, for example, the contents of registers  $S_1, S_2, \dots, S_{k-1}, S_k$  of correlator 1 will always correspond exactly to registers  $S_k, S_1, \dots, S_{k-2}, S_{k-1}$  of correlator 2. Therefore, all  $k$  synchronization possibilities are available for comparison to  $[R_0]$ , a  $k \times N/k$  matrix of the contents of the reference code registers as indicated in Fig. 3(a). The following discussion describes how the outputs of the  $k$  correlators can be selected to form an overall output identical to that of an ordinary serial correlator operating at the overall clock rate,  $f_{CK}$ .

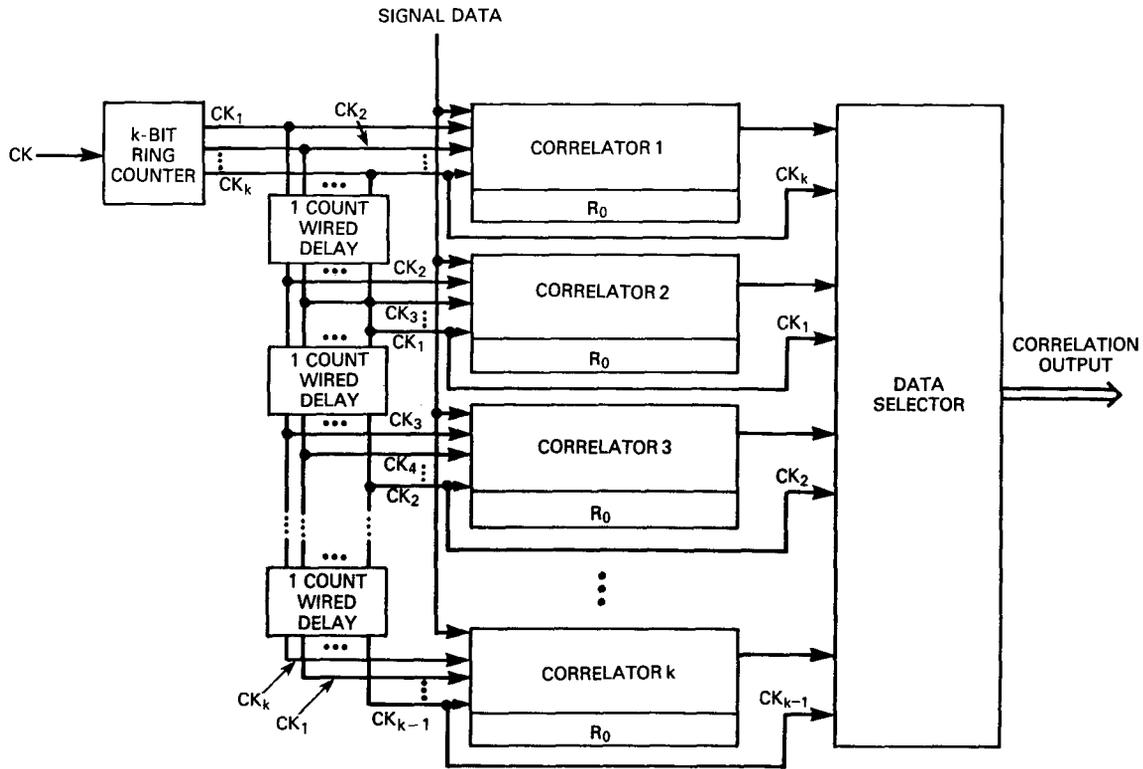


Fig. 7 — Synchronized serial load multiplexed correlator with fixed reference codes

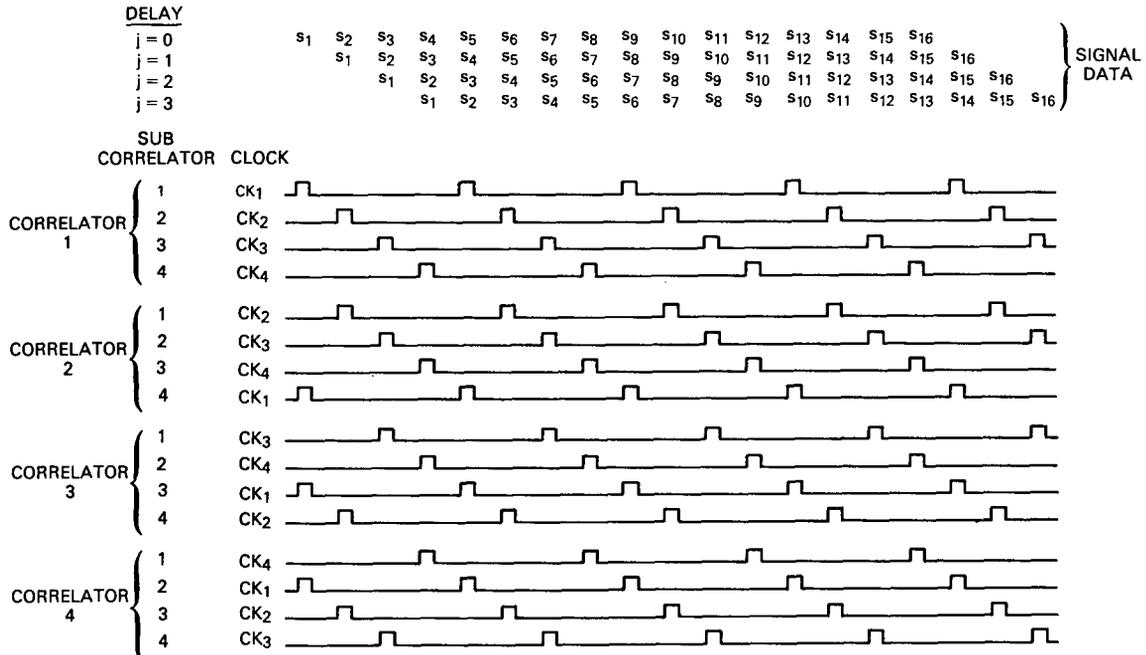


Fig. 8 — Subcorrelator timing diagram for synchronized serial load multiplexed correlator for  $N = 16$ ,  $k = 4$

To further describe the operation of the configuration of Fig. 7, we again use the case  $N = 16$  and  $k = 4$ . Figure 8 shows the required correlator timing. At the top of the diagram the  $k = 4$  synchronization possibilities are indicated as  $j = 0, 1, 2, 3$  where  $j$  is the delay in clock intervals in the signal sequence relative to the first clock pulse of correlator 1. The subcorrelator clock pulses for each of the  $k = 4$  correlators are indicated in Fig. 8. It can be seen that correlator  $i$  is always synchronized with the signal sequence when  $j = i - 1$ . Figure 9 (a) contains a table indicating the contents of the  $S$  registers of each of the four correlators after  $N = 16$  signal data bit entries for each of the synchronization possibilities. The rows of the table correspond to successive correlators  $i$  while the columns correspond to the various delays  $j$  in the signal sequence. Since  $k = 4$ ,  $i$  varies from 1 to 4 and  $j$  varies from 0 to  $k - 1 = 3$ . Each entry in the table is a  $(k \times N/k)$  matrix within which each successive row indicates the contents of a succeeding  $S$  register in each subcorrelator of the corresponding correlator. The  $j = 0$  condition corresponds to a synchronous condition in correlator 1 as indicated. Correlator 2 timing is delayed from that of correlator 1 by one basic clock interval and thus, for  $j = 1$ , correlator 2 will be synchronized; then autocorrelation with the reference register contents  $[R_0]$ , as indicated in Fig. 9 (b) for the case  $N = 16$  and  $k = 4$ , can take place. Continuing in this manner, the diagonal ( $i = j + 1$ ) entries each correspond to synchronization for the signal sequence delayed by  $j$  clock intervals. Thus, if the signal sequence is out of synchronization (delayed) relative to the clock of correlator 1 by  $j$  basic clock intervals, it will be in synchronization with the clock timing of correlator  $(j + 1)$ . Thus, for any synchronization condition, the last  $N$  bits of signal data will always be properly positioned in exactly one of the  $k$  multiplexed correlators for ordered comparison with  $[R_0]$ . The output of that multiplexed correlator at that time will correspond exactly to that of a normal  $N$ -bit serial correlator. Thus an autocorrelation peak will occur in one of the correlators when the latest  $N$  signal bits correspond exactly to

		DELAY			
		$j = 0$	$j = 1$	$j = 2$	$j = 3$
1		$s_{13} s_9 s_5 s_1$	$s_{16} s_{12} s_8 s_4$	$s_{15} s_{11} s_7 s_3$	$s_{14} s_{10} s_6 s_2$
		$s_{14} s_{10} s_6 s_2$	$s_{13} s_9 s_5 s_1$	$s_{16} s_{12} s_8 s_4$	$s_{15} s_{11} s_7 s_3$
		$s_{15} s_{11} s_7 s_3$	$s_{14} s_{10} s_6 s_2$	$s_{13} s_9 s_5 s_1$	$s_{16} s_{12} s_8 s_4$
		$s_{16} s_{12} s_8 s_4$	$s_{15} s_{11} s_7 s_3$	$s_{14} s_{10} s_6 s_2$	$s_{13} s_9 s_5 s_1$
2		$s_{14} s_{10} s_6 s_2$	$s_{13} s_9 s_5 s_1$	$s_{16} s_{12} s_8 s_4$	$s_{15} s_{11} s_7 s_3$
		$s_{15} s_{11} s_7 s_3$	$s_{14} s_{10} s_6 s_2$	$s_{13} s_9 s_5 s_1$	$s_{16} s_{12} s_8 s_4$
		$s_{16} s_{12} s_8 s_4$	$s_{15} s_{11} s_7 s_3$	$s_{14} s_{10} s_6 s_2$	$s_{13} s_9 s_5 s_1$
		$s_{13} s_9 s_5 s_1$	$s_{16} s_{12} s_8 s_4$	$s_{15} s_{11} s_7 s_3$	$s_{14} s_{10} s_6 s_2$
3		$s_{15} s_{11} s_7 s_3$	$s_{14} s_{10} s_6 s_2$	$s_{13} s_9 s_5 s_1$	$s_{16} s_{12} s_8 s_4$
		$s_{16} s_{12} s_8 s_4$	$s_{15} s_{11} s_7 s_3$	$s_{14} s_{10} s_6 s_2$	$s_{13} s_9 s_5 s_1$
		$s_{13} s_9 s_5 s_1$	$s_{16} s_{12} s_8 s_4$	$s_{15} s_{11} s_7 s_3$	$s_{14} s_{10} s_6 s_2$
		$s_{14} s_{10} s_6 s_2$	$s_{13} s_9 s_5 s_1$	$s_{16} s_{12} s_8 s_4$	$s_{15} s_{11} s_7 s_3$
4		$s_{16} s_{12} s_8 s_4$	$s_{15} s_{11} s_7 s_3$	$s_{14} s_{10} s_6 s_2$	$s_{13} s_9 s_5 s_1$
		$s_{13} s_9 s_5 s_1$	$s_{16} s_{12} s_8 s_4$	$s_{15} s_{11} s_7 s_3$	$s_{14} s_{10} s_6 s_2$
		$s_{14} s_{10} s_6 s_2$	$s_{13} s_9 s_5 s_1$	$s_{16} s_{12} s_8 s_4$	$s_{15} s_{11} s_7 s_3$
		$s_{15} s_{11} s_7 s_3$	$s_{14} s_{10} s_6 s_2$	$s_{13} s_9 s_5 s_1$	$s_{16} s_{12} s_8 s_4$

Fig. 9(a) — Signal register contents of synchronized serial load correlator after  $N = 16$  signal bit inputs

Fig. 9(b) — Corresponding contents of reference registers

$[R_0]$	$r_{13}$	$r_9$	$r_5$	$r_1$
	$r_{14}$	$r_{10}$	$r_6$	$r_2$
	$r_{15}$	$r_{11}$	$r_7$	$r_3$
	$r_{16}$	$r_{12}$	$r_8$	$r_4$

$[R_0]$ . Further investigation reveals that each correlator produces a correct sample of the overall correlation output when the most recent signal bit is clocked into its  $k$ th subcorrelator. Thus an equivalent correlator output can be formed by applying the outputs of the  $k$  correlators to a data selector as shown in Fig. 7 with the  $i$ th correlator output selected by clock  $CK_k$  for  $i = 1$  and clocks  $CK_{i-1}$  for  $2 \leq i \leq k$ .

The top row of Fig. 9 (a) indicates the contents of the  $S$  registers of correlator 1 after input of  $N = 16$  signal bits for each of the  $k = 4$  synchronization possibilities for the input signal sequence. This suggests an alternate means of accomplishing serial load synchronized correlation, using  $k$  correlators, with the identical data and timing inputs of correlator 1, but with  $k$  different reference code configurations. Let us designate each of the  $k$  signal-register matrix ( $SM$ ) possibilities for correlator 1 by  $[SM_j]$  where  $j$  is the number of basic clock intervals by which the signal sequence is out of synchronization with the subcorrelator 1 clock. Thus  $[SM_0]$  corresponds to the synchronized case  $j = 0$ . Then, any  $[SM_j]$  of correlator 1 represents a top-down circular rotation of the positions of the  $S$  registers of  $[SM_0]$ . We can accomplish autocorrelation by rearranging  $[R_0]$  to correspond to each of the  $k$  possible signal register matrix patterns. The resulting reference code arrangements are designated  $[R_j]$  and are also obtained by a circular top-down rotation of the rows ( $R$  registers) of  $[R_0]$ . The matrices  $[SM_j]$  and corresponding  $[R_j]$  are shown in Fig. 10 for  $N = 16$  and  $k = 4$ . A generalized block diagram of this alternative serial load synchronized multiplexed correlator appears in Fig. 11. The signal data and timing presented to each correlator are identical. The  $k$  correlator operations differ only in that the reference code used in correlator  $i$  is  $[R_{i-1}]$ . The relationship between the correlators, overall correlation output, and the clock timing is the same as before. Thus the output of correlator  $i$  is a sample of the overall correlator output whenever the latest signal bit has been clocked into correlator  $i$  by clock  $CK_k$  for  $i = 1$  or clock  $CK_{i-1}$  for  $2 \leq i \leq k$ . When the  $N$ th bit of a reference code replica is clocked into one of the correlators, the autocorrelation peak will occur in the overall correlation output.

### Parallel Load

A parallel load version of a synchronized multiplexed correlator is shown in Fig. 12. In this configuration, the data are clocked into a  $k$ -bit serial register by  $CK$  at the basic clock rate  $f_{CK}$ . The  $k$ -bit parallel output from the serial register is applied to the  $k$  signal registers of each of the  $k$  correlators. The basic clock  $CK$  is inverted and applied to a ring counter with each output  $CK_i$  applied to correlator  $i$ . Thus each correlator is clocked every  $k$  clock intervals with a one basic clock interval delay between successive correlators. Inverting  $CK$  before application to the ring counter, causes  $CK_i$  to be generated on the trailing edge of a  $CK$  pulse. This allows one clock width interval of  $CK$  for the latest signal bit to be accepted by the  $k$ -bit serial shift register before being shifted into correlator  $i$ .

A timing diagram showing the relationship between  $CK$  and each of the  $CK_i$  for  $k = 4$  is shown in Fig. 13. At the top of the diagram the  $k = 4$  synchronization possibilities are indicated as  $j = 0, 1, 2, 3$  where  $j$  is the delay in terms of the number of clock intervals of  $CK$  in the signal sequence relative to the first clock pulse of  $CK$  following a clock pulse  $CK_1$  of correlator 1. Figure 14 contains a table showing the contents of the  $S$  registers of succeeding subcorrelators for each of the four parallel multiplexed correlators after  $N = 16$  signal bit entries, for each of the four synchronization possibilities. The table consists of  $k \times N/k$  matrices of the contents of the  $S$  registers in the form of that presented in Fig. 9 (a) for the serial case. It can be seen that when synchronization exists ( $j = 0$ ), following each  $CK_1$  pulse, the latest  $N = 16$  signal bits will be in proper order in correlator 1 for comparison to  $[R_0]$  (indicated in Fig. 9 (b)). The output from correlator 1 at that time will be identical to that from a normal correlator for the same latest  $N = 16$  signal bits. If these  $N = 16$  signal bits match exactly with  $[R_0]$ , that output will indicate a correlation peak. When  $j = 1$ , following each  $CK_2$  pulse, correlator 2 will contain the proper overall correlation output. In general, correlator  $i$  will produce the proper contribution to the overall correlation at each  $CK_i$ . From Fig. 12, we see that the overall correlation output from the synchronized parallel multiplexed correlator is formed by means of a data selector with the output of correlator  $i$  selected by its clock  $CK_i$  (with appropriate delay).

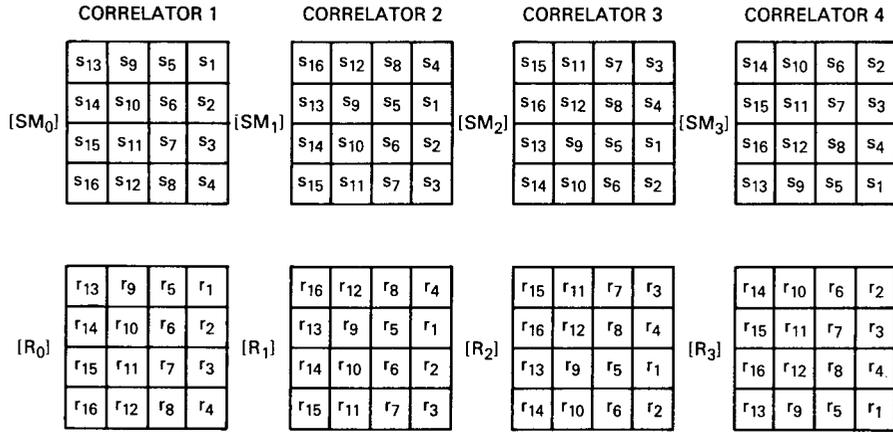


Fig. 10 — Signal and reference register matrices for alternative synchronized serial load correlator for  $N = 16$ ,  $k = 4$

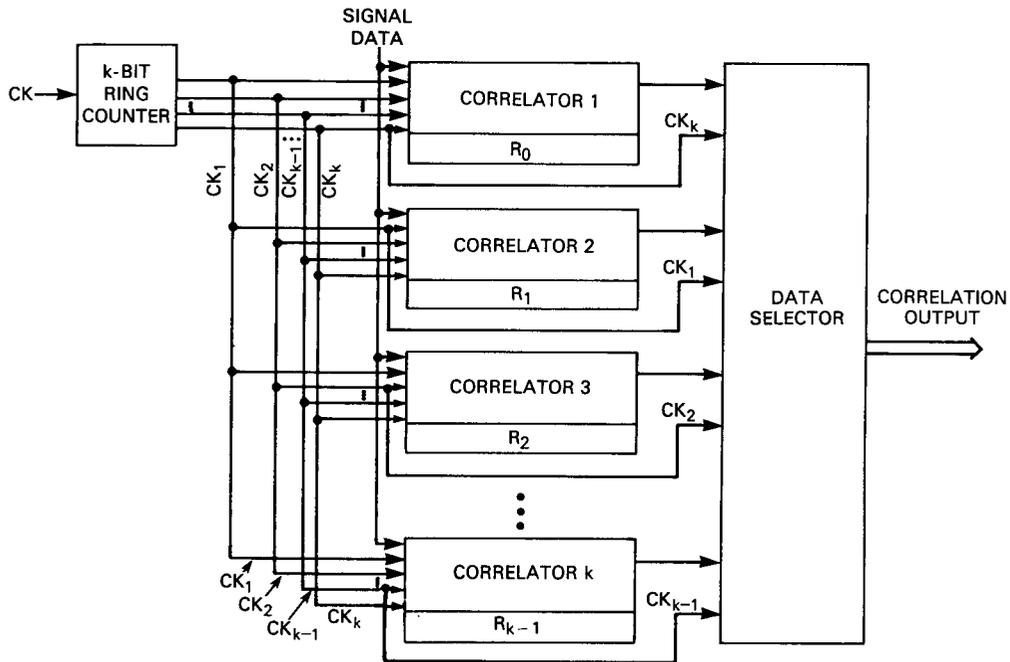


Fig. 11 — Alternative synchronized serial load multiplexed correlator with modified reference

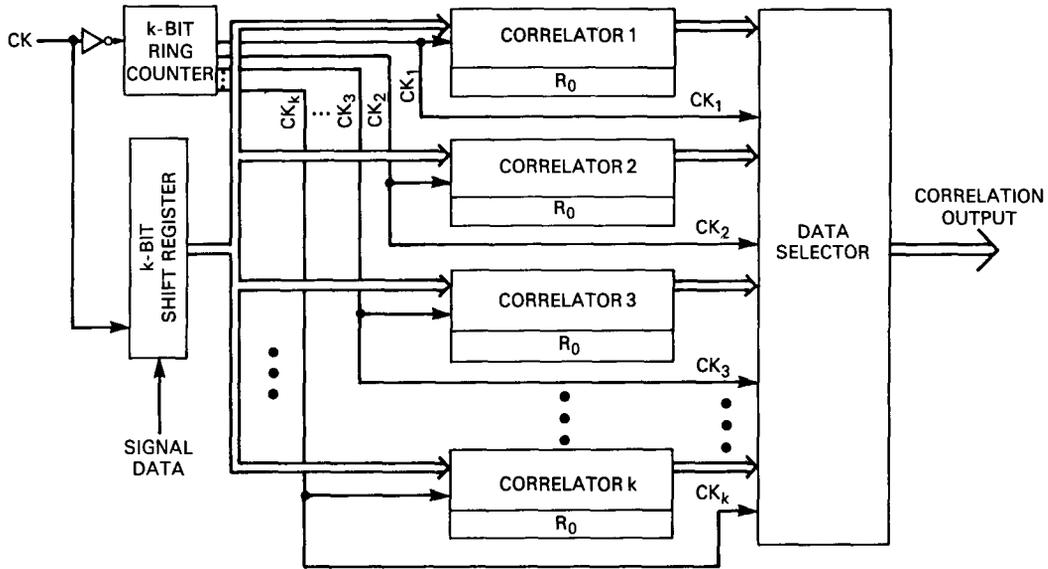


Fig. 12 — Synchronized parallel load multiplexed correlator with correlation output

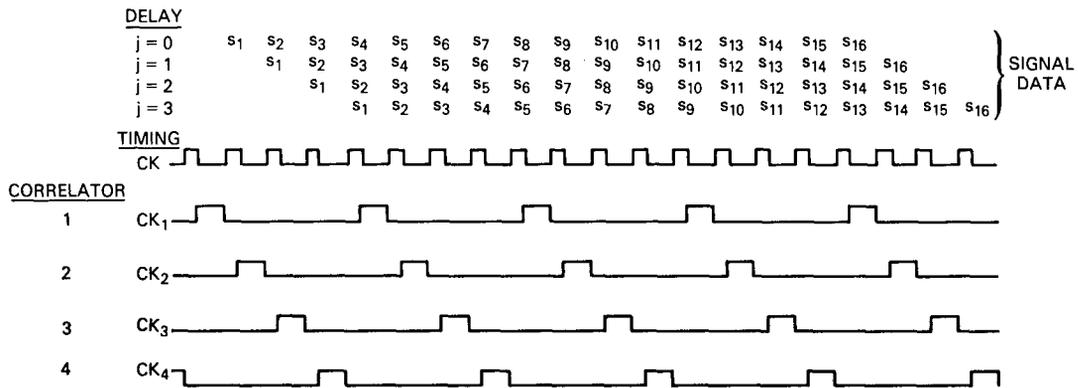


Fig. 13 — Correlator clock timing diagram for synchronized parallel load multiplexed correlator for  $N = 16$ ,  $k = 4$

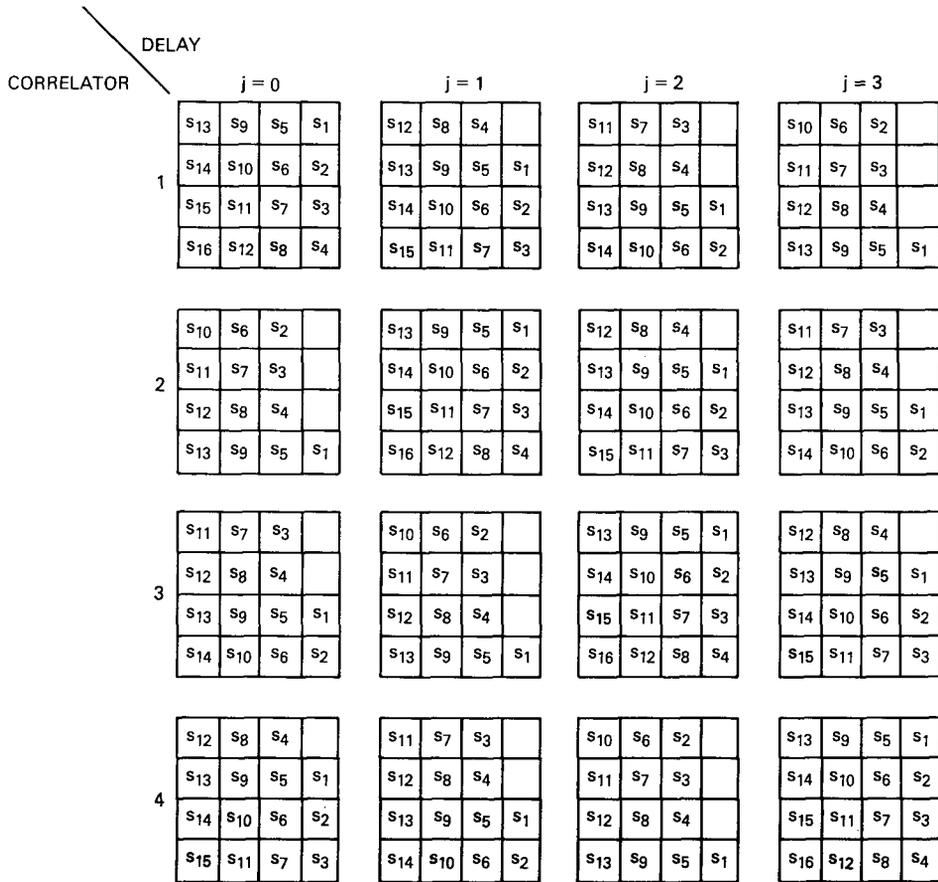


Fig. 14 — Signal register contents of synchronized parallel load correlator after  $N = 16$  signal bit inputs, for the case  $N = 16, k = 4$

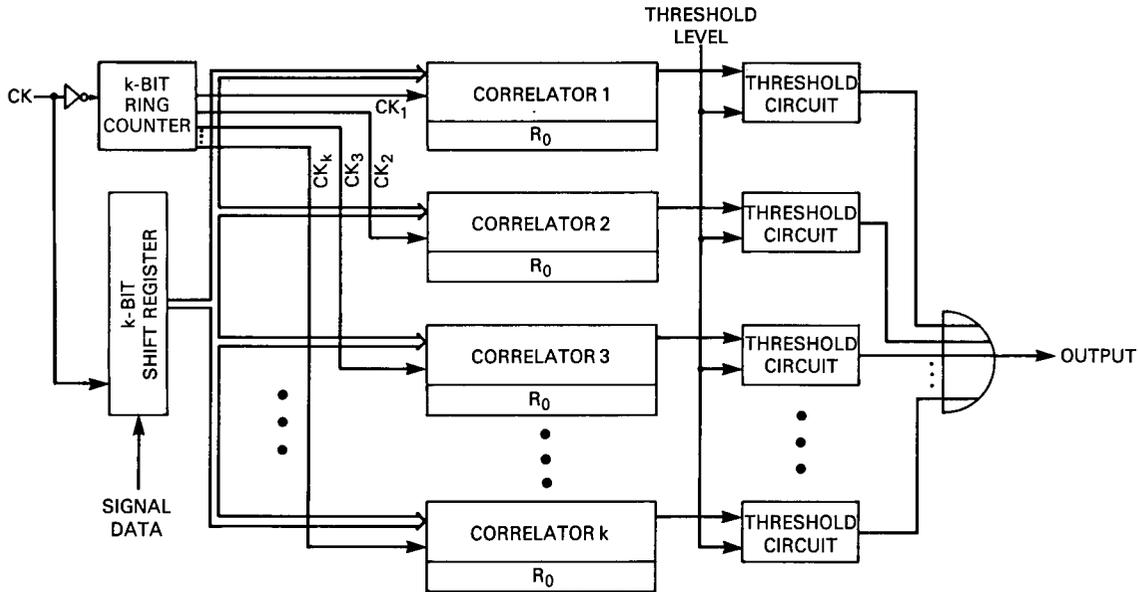


Fig. 15 — Synchronized parallel load multiplexed correlator with correlator peak detection

An alternate approach to the combination of correlator outputs exists for the synchronized parallel load multiplexed correlator. It can be seen from Fig. 14 that the signal register matrix for each correlator will always be arranged in right-to-left column-by-column top-down order for comparison with reference matrix  $[R_0]$ . The resulting bit comparison will be the same as those which would occur in a normal unmultiplexed correlator, but each will be delayed in time. Therefore, the correlation sidelobes from any correlator  $i$ , although displaced in time, will not be any larger than those from a single unmultiplexed correlator. Thus, instead of forming the overall correlator output as shown in Fig. 12, and then applying a threshold to detect a correlation peak, equivalent results can be achieved by applying a threshold circuit to each multiplexed correlator output as shown in Fig. 15. The threshold circuit outputs would be applied to a logical OR gate to indicate a correlation peak detection. The threshold setting applied to the  $k$  threshold circuits would be identical to that used with the overall correlator output with equivalent results as far as probability of detection and frequency of false detections.

## SIMPLIFIED REALIZATION

Several methods of accomplishing correlation without presynchronization are presented here. Those methods involve serial load and parallel load techniques. The alternative serial load configuration accepts the signal data identically in each of the  $k$  correlators but requires a different reference data storage pattern for each of the  $k$  correlators. Actually the same  $N$  reference bits are used in each correlator, but the subcorrelator  $R$  registers are rotated top-down one position in succeeding correlators. Thus it is obvious that only one set of  $k$   $N/k$ -bit signal and reference registers is required along with  $k$  sets of  $N$ -bit comparator circuitry. The various reference bit patterns can be developed by appropriately rotating the position of the  $R$  registers by the manner in which they are wired to each set of comparator circuits. A block diagram of this approach is shown in Fig. 16. Details of the interconnection of the signal and reference registers and the comparators are shown in Fig. 17 for comparator 2 for the example case  $N = 16$ ,  $k = 4$  where a correlation of signal bits  $s_1, s_2, \dots, s_{16}$  with reference bits  $r_1, r_2, \dots, r_{16}$ , respectively, has been assumed. This example illustrates a one-position rotation in the reference registers and a comparison to the  $k = 4$  signal registers corresponding to the subcorrelators.

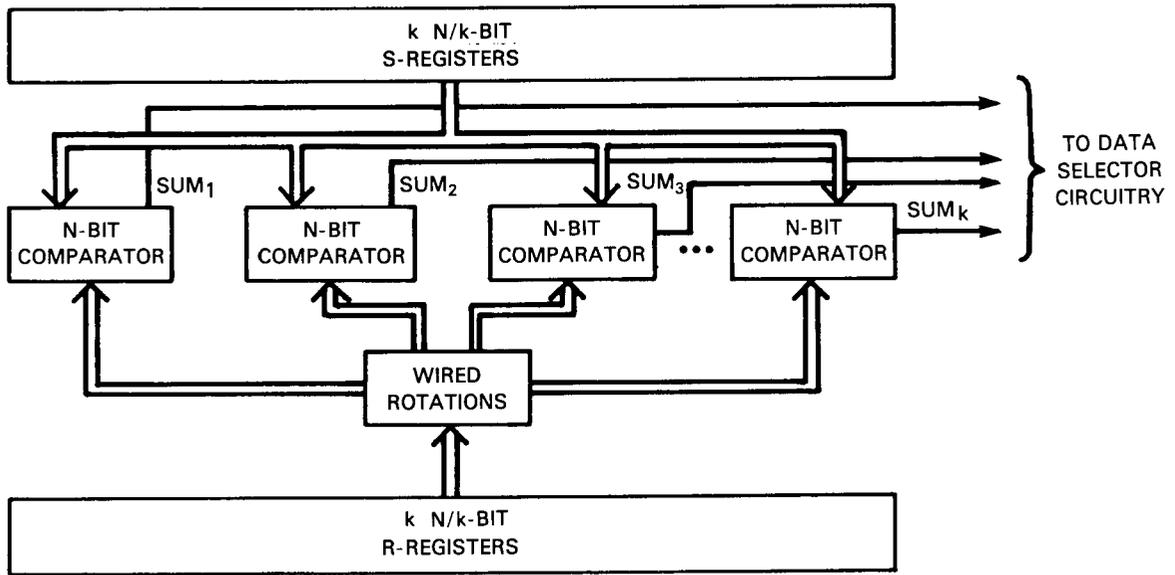


Fig. 16 — Simplified synchronized serial load multiplexed correlator

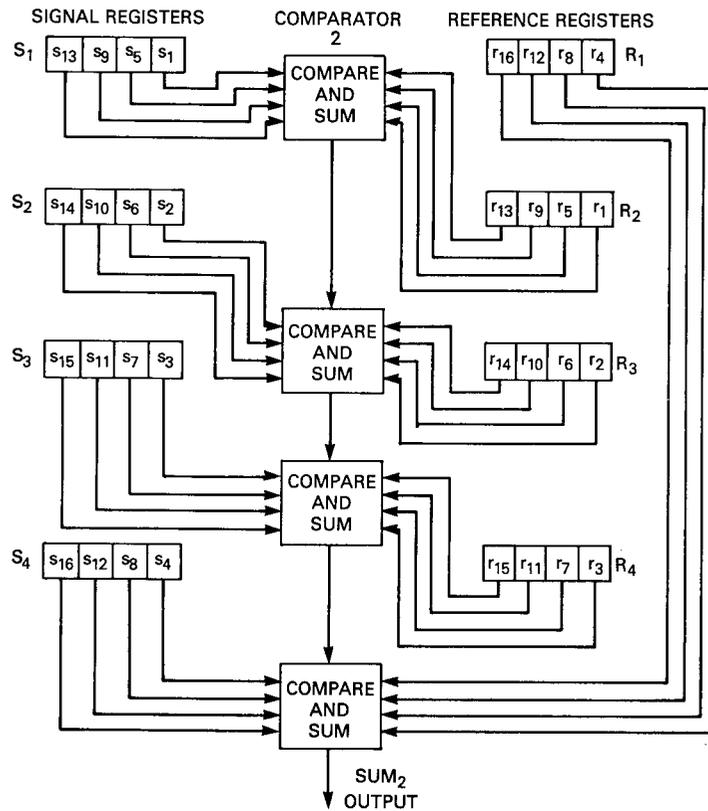


Fig. 17 — Interconnection of signal registers, reference registers, and compare and sum circuits for comparator 2

The other serial load technique uses fixed reference data in each of its  $k$  correlators. In this case, however, the signal data in each of the  $S$  registers is identical but is rotated top-down in position. Thus a common signal register can be utilized with the signal register contents for each correlator obtained by an appropriate wired rotation of signal register positions. This approach is illustrated in Fig. 18. The interconnection of signal registers, reference registers, and compare circuits for comparator 2 for the case  $N = 16$ ,  $k = 4$  will be similar to that of Fig. 16 but with the signal and reference registers interchanged in location. A similar approach can be applied in the case of parallel load multiplexed correlators. Thus, if instead of using available digital correlator devices, an  $N$ -bit synchronized multiplexed correlator were implemented as an integrated circuit device, only one set of  $k$   $N/k$ -bit signal and reference code registers are actually required.

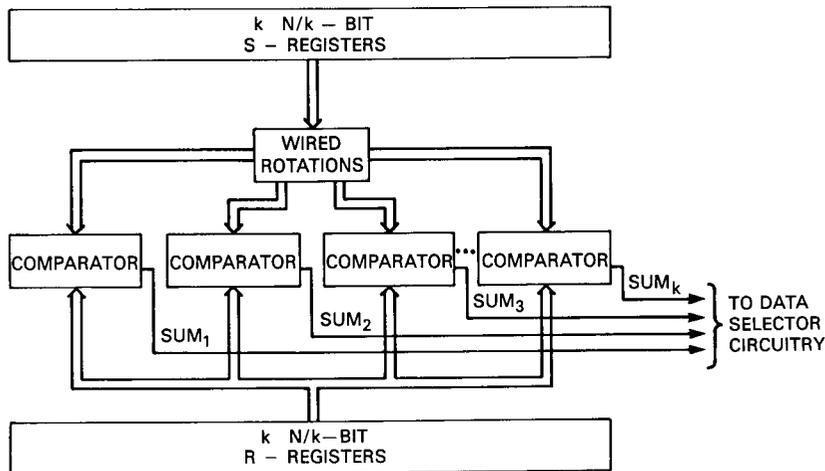


Fig. 18 — Alternative simplified synchronized serial load multiplexed correlator

## SUMMARY AND CONCLUSIONS

The application of multiplexing to increase the effective speed of digital correlators has been considered here. Two basic methods, serial load and parallel load, have been presented, each utilizing  $k$   $N/k$ -bit correlators to provide  $N$ -bit correlation. They are both almost equivalent in complexity and provide overall correlation at a rate of  $k$  times that of the clock rate of individual correlator devices. In those circumstances where exact code synchronization is available, multiplexing requires no significant increase in hardware requirements. However, in these cases, there is a reduction in time resolution by a factor of  $k$  as compared to a digital correlator without multiplexing. In general, code synchronization will not be available. It was shown here that exact synchronization can be achieved by providing  $k$  multiplexed correlators, one for each of the  $k$  total synchronization possibilities. Actually only one set of  $k$   $N/k$ -bit signal and reference code registers along with  $k$  sets of  $N$ -bit comparator circuits are required. The overall correlator output can be formed by properly sampling the outputs of the  $k$  correlators by means of a data selector controlled by the  $k$  multiplexed correlator clock signals. Synchronized multiplexed correlator systems for serial as well as parallel load formats have been developed. The techniques presented here can also have application to charge-coupled-device (CCD) correlators.

NRL has utilized 15 MHz TRW LSI Products TDC 1004J digital correlator devices with analog output for laboratory implementation of a  $64 \times 4 = 256$ -bit multiplexed digital correlator using the parallel load configuration. With synchronization assumed, only a single multiplexed correlator section was implemented as described in Appendix A. This implementation demonstrated the basic logical concepts of multiplexed correlation but operated at limited rates. To fully achieve factor-of-four improvement in maximum correlation rates from the multiplex concept would require high-speed input/output

and clock generation circuitry. A limiting factor in achieving such correlation rates is the set-up and hold times of shift registers such as those used in the TRW devices. This can be overcome by available high-speed flip-flops with suitable timing parameters, used as a buffer at correlator register inputs. The concept of synchronized multiplexed correlation with use of data selectors to form an overall output identical to that from a normal correlator was confirmed by computer simulation as described in Appendix B.

Thus, significant speed improvement in digital correlation can be accomplished by means of multiplexing techniques. Unfortunately, there is an accompanying increase in circuitry and, of course, power required. Where high-speed correlation and the advantages of digital technology are required, multiplexing could provide a possible alternative to other technologies. The tradeoffs involved should become more acceptable with the advent of very high speed integrated circuit (VHSIC) techniques now being developed by the Department of Defense (DOD).

### ACKNOWLEDGMENTS

The author acknowledges the very able efforts of Mr. Daniel P. Garfinkel in performing the laboratory experimentation and software simulations described in the Appendices that he coauthored; Mr. John Layno also contributed to this laboratory and software effort. A great appreciation is expressed for the many helpful comments and suggestions provided by Mr. Carlyle V. Parker.

### REFERENCES

1. R.C. Dixon, *Spread Spectrum Systems* (John Wiley & Sons, New York, 1976).
2. D. Nicholson, "Analysis of Spread-Spectrum Effectiveness in Identification Systems," Final Technical Report, Melpar Div., E-Systems, Inc., June 1980.
3. R.D. Colvin, "Spread Spectrum Devices Cater to New Systems," *Microwaves*, 20(2), 65-80 (Feb. 1981).
4. TRW LSI Products Data Sheet 102B for TDC 1004J Correlator, El Segundo, CA, Jan. 1980.
5. TRW LSI Products Preliminary Data Sheet 116C for TDC 1023J Correlator, El Segundo, CA, Sept. 1980.
6. L.M. Leibowitz, "Comparative Analysis of the use of Dynamic and Static Shift Registers in Digital Signal Processors (U)," NRL Report 7482, Dec. 1972, pp. 6-8.
7. H. Schmid, *Electronic Analog/Digital Conversions*, (Van Nostrand Reinhold Co., New York, 1970).

## Appendix A

### EXPERIMENTAL CIRCUITRY (Coauthored by D. Garfinkel)

A four times multiplexed correlator section of length  $4 \times 64 = 256$  bits consisting of four sub-correlators (TRW TDC 1004J 64-bit correlator devices) was implemented at NRL. The parallel load technique was utilized in this initial experimental model. The objective was to confirm the basic logical operation of the approach presented in the main body of this report by showing that overall correlation at four times the correlator device clock rate could be achieved. In this initial experimental implementation, there was no attempt made to attain maximum speed. Attempts to attain full theoretical speed are planned for subsequent efforts. Figures A1 to A4 are interconnection diagrams of the actual test circuitry used to multiplex four 64-bit digital correlator devices with analog outputs. For purposes of simplification, some redundant circuitry is omitted from the diagrams but is included in the following description.

The circuit of Fig. A1 uses flip-flops to assure synchronous starting of both a free-running clock (A Clock) and a burst of 256 clock pulses (B Clock) at the clock frequency when a bounceless switch is turned on. The 256-pulse clock burst, which is used to clock the four B (reference) and M (mask) registers of the TRW-TDC 1004J correlators, is generated by logically ANDing the clock with a gate signal (G1 in Fig. A1), which is enabled when the bounceless switch is activated, and disabled when the output of a divide-by-256 counter clears the gating flip-flop (U4-1). The free-running clock is also generated by logically ANDing the clock with a gate signal (G2 in Fig. A1) which is also enabled when the bounceless switch is activated, but is disabled only when the bounceless switch is turned off. The divide-by-256 counter is composed of flip-flops U1-1, U2-1, and U3-1 with external logic; the gating flip-flop for the 256-pulse burst is U4-1; and the gating flip-flop for the free-running clock is U6-1. The dual flip-flop circuit (U5-1) produces a single pulse output (SP in Fig. A1) after the bounceless switch is activated. This single pulse clocks the two gating flip-flops (U4-1 and U6-1) to synchronously enable the generation of both A clock and B clock.

The mask function is also generated by the circuitry of Fig. A1 and consists of all logical "1's" with a logical "0" as the last (256th) bit. This ensures that the correlator will be compatible with the 255-bit code length of the eight-stage maximal length pseudorandom (PN) code generator of Fig. A2(a).

There are two main functional circuits shown in Fig. A2(a). The first is an eight-stage maximal length PN code (m-sequence) generator and consists of two 4-bit counters (U1-2, U2-2), seven exclusive OR-gates, and seven mechanical switches. By manually setting these switches one can program a specific 255-bit PN sequence (since we are using an eight-stage shift register, and  $2^8 - 1 = 255$ ). The output PN code [in Fig. A2(a)] goes to two four-stage shift registers, one of which is clocked by the free running clock (A Clock) and the other by the clock burst (B Clock). Only the first of these two is shown in the figure. The code generator is clocked by the free running clock and thus runs continuously.

The second functional circuit indicated in Fig. A2(a) appears in dashed lines and accomplishes the multiplexing function in the test circuit. The multiplexing scheme used is the parallel-load approach. The signal data from the PN code generator is shifted into a four-stage shift register [(U3-2 in Fig. A2(a))]. After each four signal bits have been serially clocked into the shift register, the output of a divide-by-four counter (U4-2 with a three input NOR gate) parallel loads each of the four signal bits

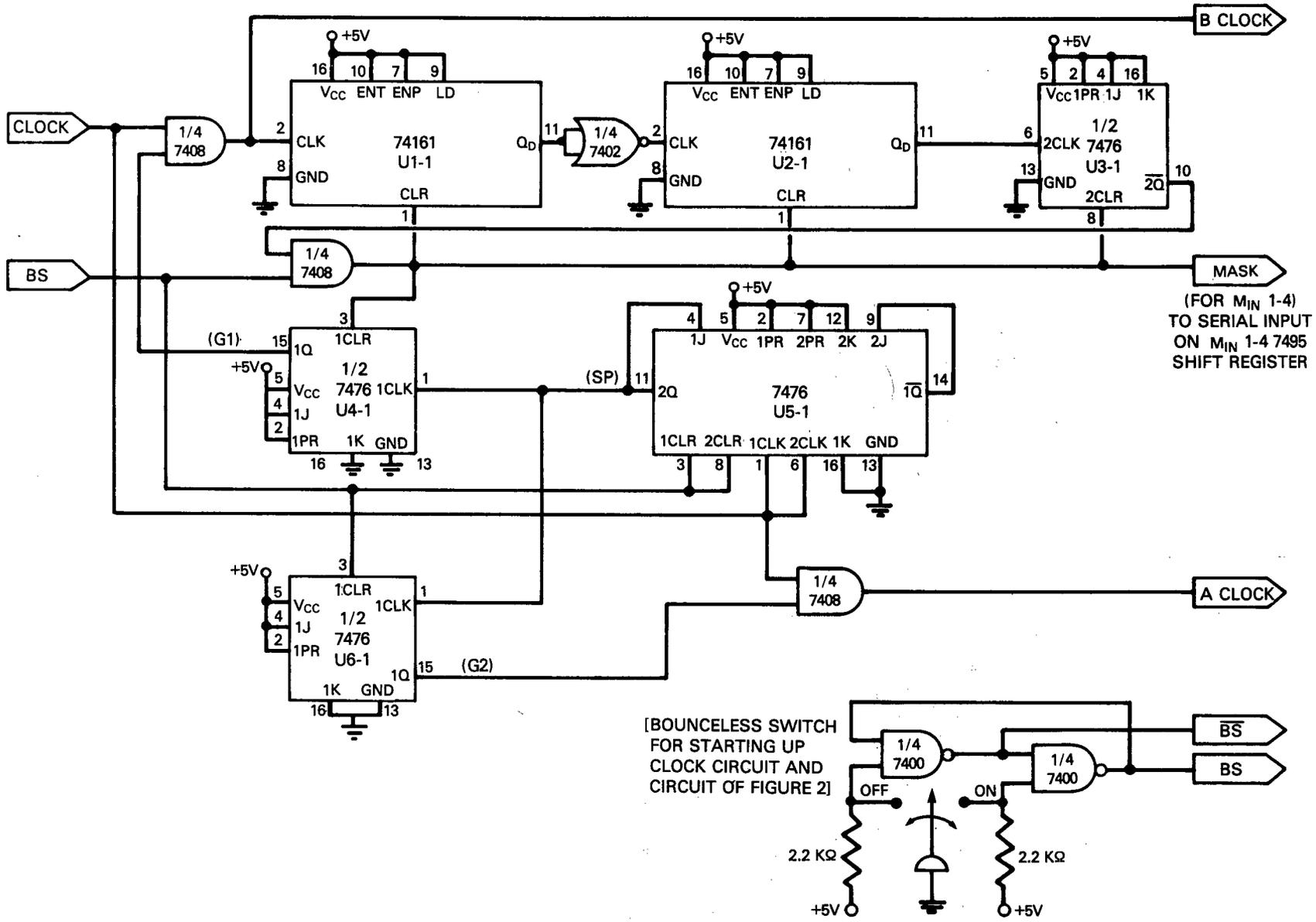
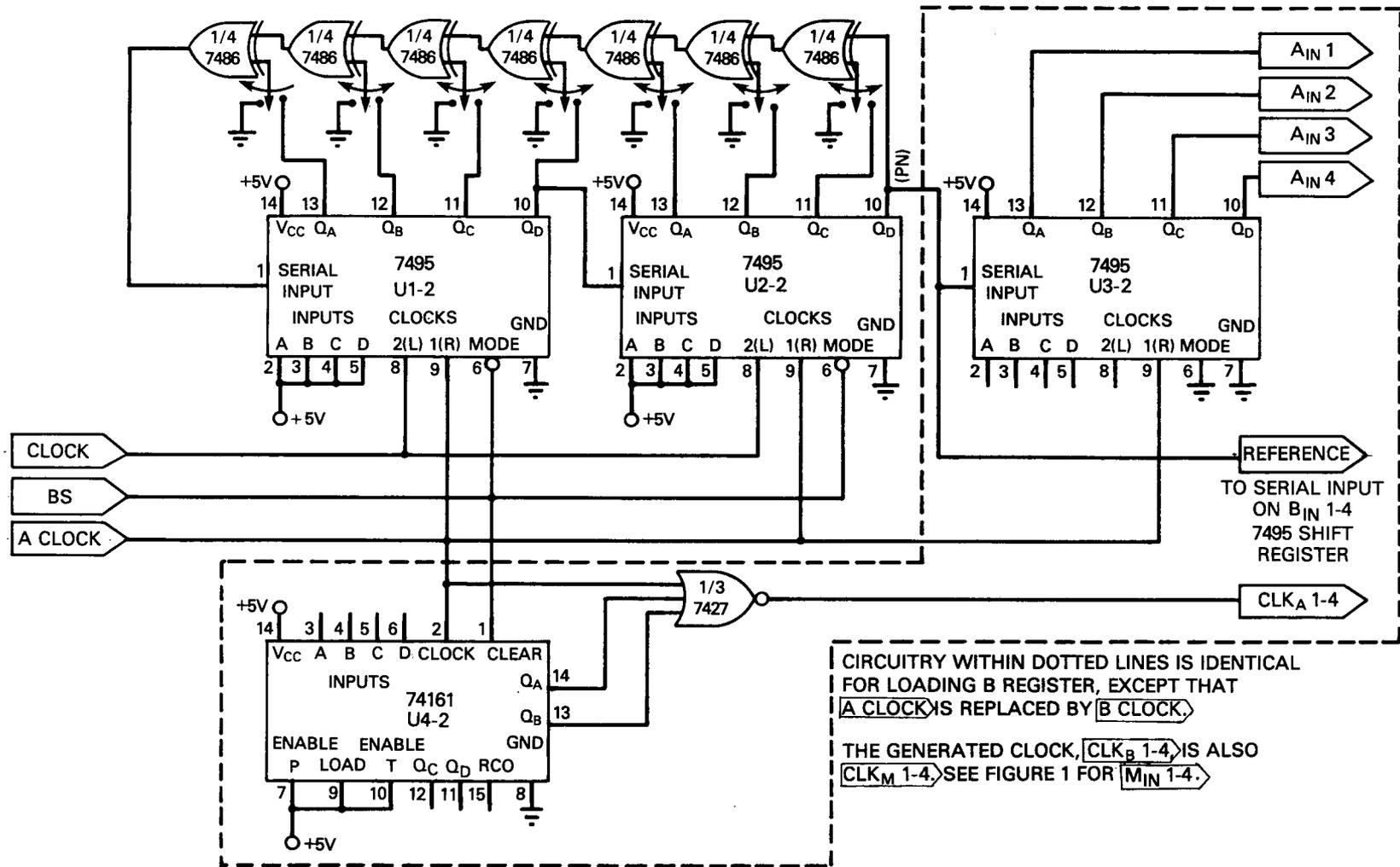


Fig. A1 — Circuit which synchronously starts up clocks for signal (A clock), and reference and mask (B clock) registers of correlators. B clock is a burst of 256 clock pulses, which are coincident with A clock, which is free-running. Mask register data are also generated.



L. M. LEIBOWITZ

Fig. A2(a) — Maximal length sequences code generator and circuitry-to-load signal (A) registers of correlators

Fig. A2(b) — Timing diagram of code generator and parallel load correlator clock for  $k = 4$



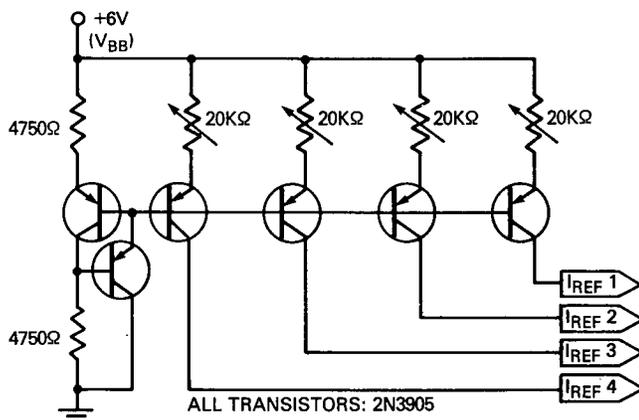


Fig. A3 - Multiple current source circuit for  $I_{REF}$  1-4

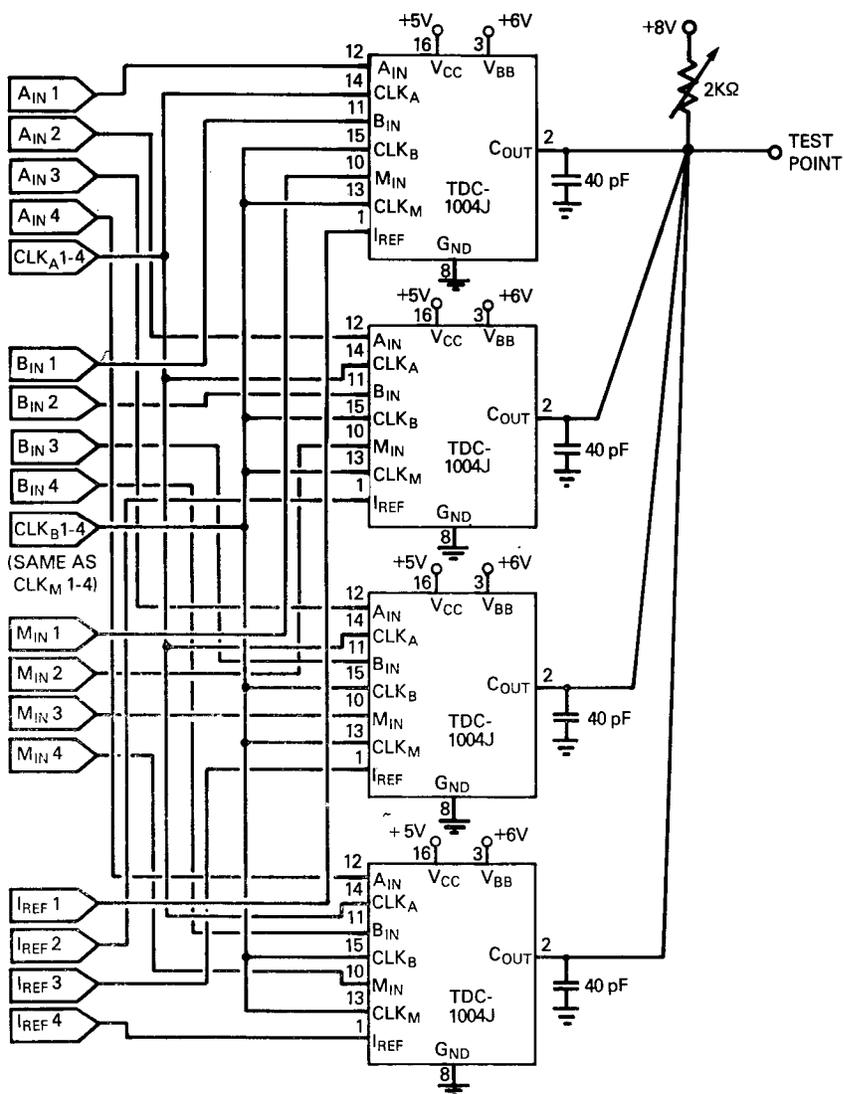


Fig. A4 - Multiplexed correlator built from four individual correlators (test circuit actually built with supporting circuitry of Figs. 1, 2, 3)

into each of the signal registers of the correlators. The divide-by-four clock is common to the signal registers of each correlator device, with timing as shown in Fig. A2(b). Every time the divide-by-four counter output clocks the correlator registers, four new signal data bits are in the four-stage shift register.

The circuit shown in dashed lines in Fig. A2(a) is repeated twice, once for the reference registers and once for the mask registers of the correlators. For the reference register circuit, the four stage shift register and the divide-by-four clock are clocked by B Clock, and the input data to the four stage shift register is generated by the PN code generator output (PN). The outputs of the shift register go to  $B_{in}$  1-4 on the correlators, and the divide-by-four clock is common to CLK B 1-4 on the correlators. For the mask register circuit, the shift register and divide-by-four clock are clocked by B Clock, and the input data is MASK output from Fig. A1.

Figure A3 shows a circuit for generating  $I_{REF}$  1-4, the reference current inputs required by the four correlator devices. The variable resistors are adjusted so that each  $I_{REF}$  is about  $320 \mu A$ .

Figure A4 shows the four correlator devices with all input signals generated in the circuits of Figs. A2(a) and A3. The resulting correlation is indicated by the amount of current going into the  $C_{out}$  pin on each correlator, so, as the correlation (the number of bit agreements) increases, the voltage drop across the  $2 k\Omega$  variable resistor increases and the voltage at the test point decreases, with  $+4.0 V$  indicating maximum correlation and  $+5.5 V$  indicating minimum (zero) correlation.

The operation of this experimental model of one section of a synchronized multiplexed digital correlator is indicated in the oscillographs of Figs. A5 through A7. The basic clock frequency during these tests was 1 MHz while the individual correlator devices were operating at 250 kHz. The top of Fig. A5 shows four periods of the 255-bit PN m-sequence code [8,6,5,3] (see Ref. 1). The lower oscillograph is the autocorrelation signal of that code at the output of the multiplexed correlator section. Due to the odd length of the PN code and the even multiplexed correlator length, the correlation peaks occur only every fourth ( $k$ th) cycle of the PN code. Each of the  $k = 4$  synchronization possibilities occurs on each code cycle, but with the single multiplexed section, only every fourth correlation peak can be observed at the correlator output.

Figure A6 is approximately a four times enlargement of Fig. A5 and shows a little more than one cycle of the [8,6,5,3] PN code and the corresponding autocorrelation output of the single multiplexed correlator section. In Fig. A6 (as well as in Fig. A5) the correlation peak occurs over  $k = 4$  code clock intervals as described for the parallel load case in the main body of this report. In both Figs. A5 and A6 the correlator signal registers were initially loaded with the PN m-sequence code; thus, the nonpeak autocorrelation output corresponds to the level at which agreements minus disagreements ( $A - D$ ) equals  $-1$  (or  $A = (N - 1)/2$ ) for m-sequences as described in Ref. 1. The noise-like appearance of this nonpeak level is the result of clock feed-through due to circuit stray-capacitances in the experimental breadboard setup. With the correlator signal registers initially cleared, during the first cycle of the PN code, the correlation output is other than  $A = (N - 1)/2$  as indicated in Fig. A7. After the first cycle of the PN code has been shifted through the signal registers, the nonpeak autocorrelation will be constant, corresponding to the  $A - D = -1$  or  $A = (N - 1)/2$  level.

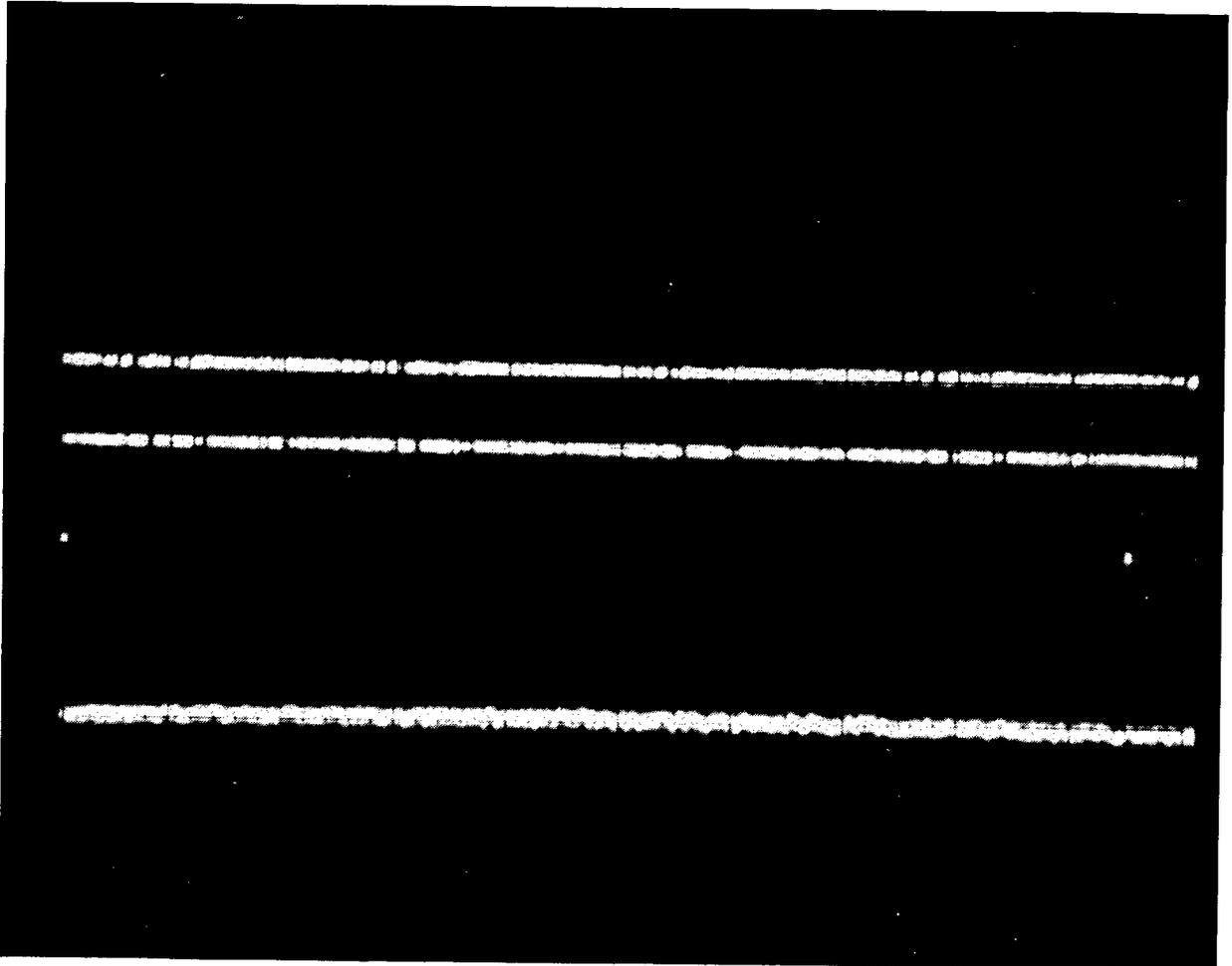


Fig. A5 — Top: 255-bit PN code [8,6,5,3] (5V/div., 100  $\mu$ s/div.) Bottom: Autocorrelation output from single multiplexed correlator section,  $k = 4$ ,  $N/4 = 64$  (0.5V/div., 100  $\mu$ s/div.)

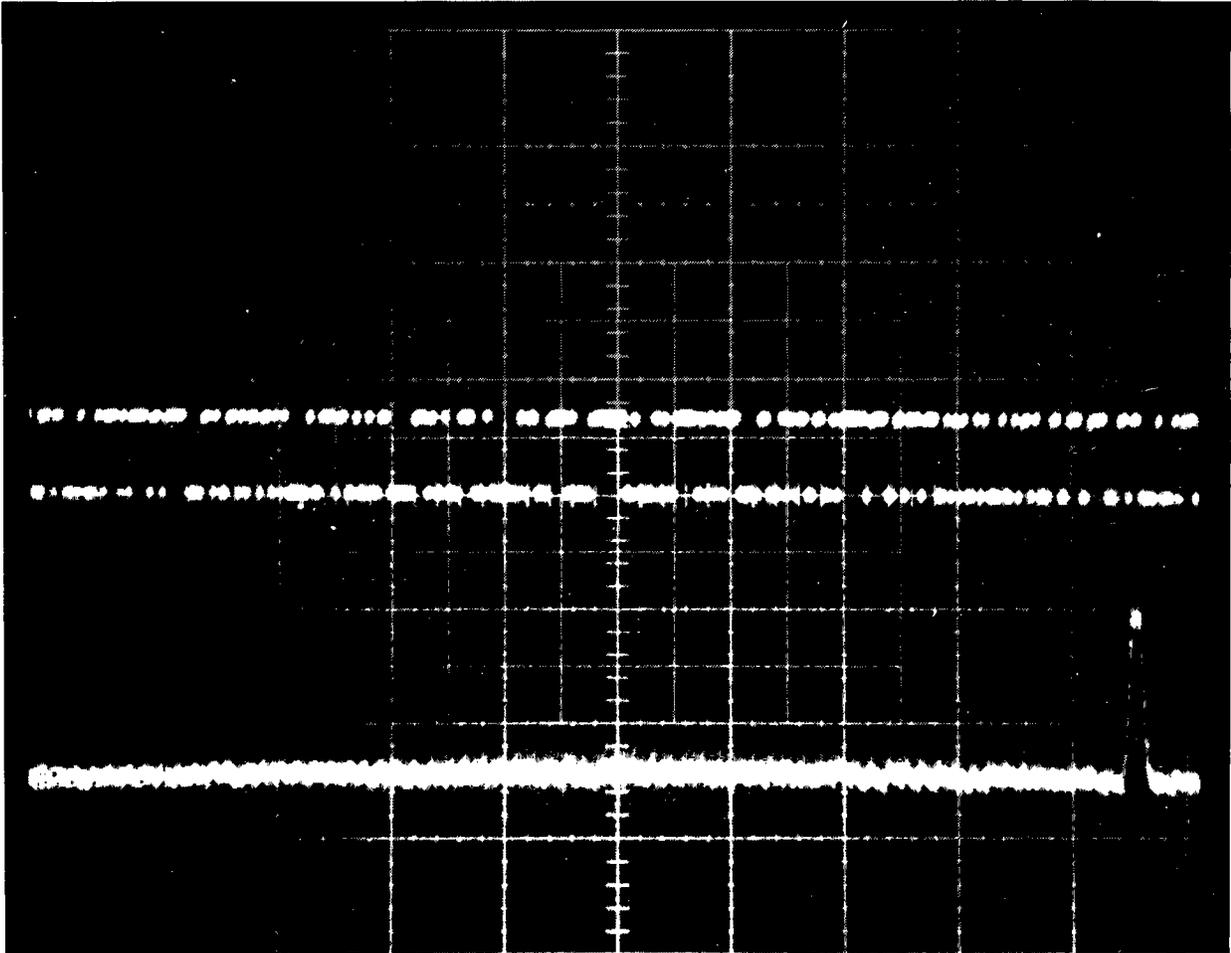


Fig. A6 — Time expansion of PN code and autocorrelation of Fig. A5  
(28  $\mu$ s/div.), same vertical scales as Fig. A5

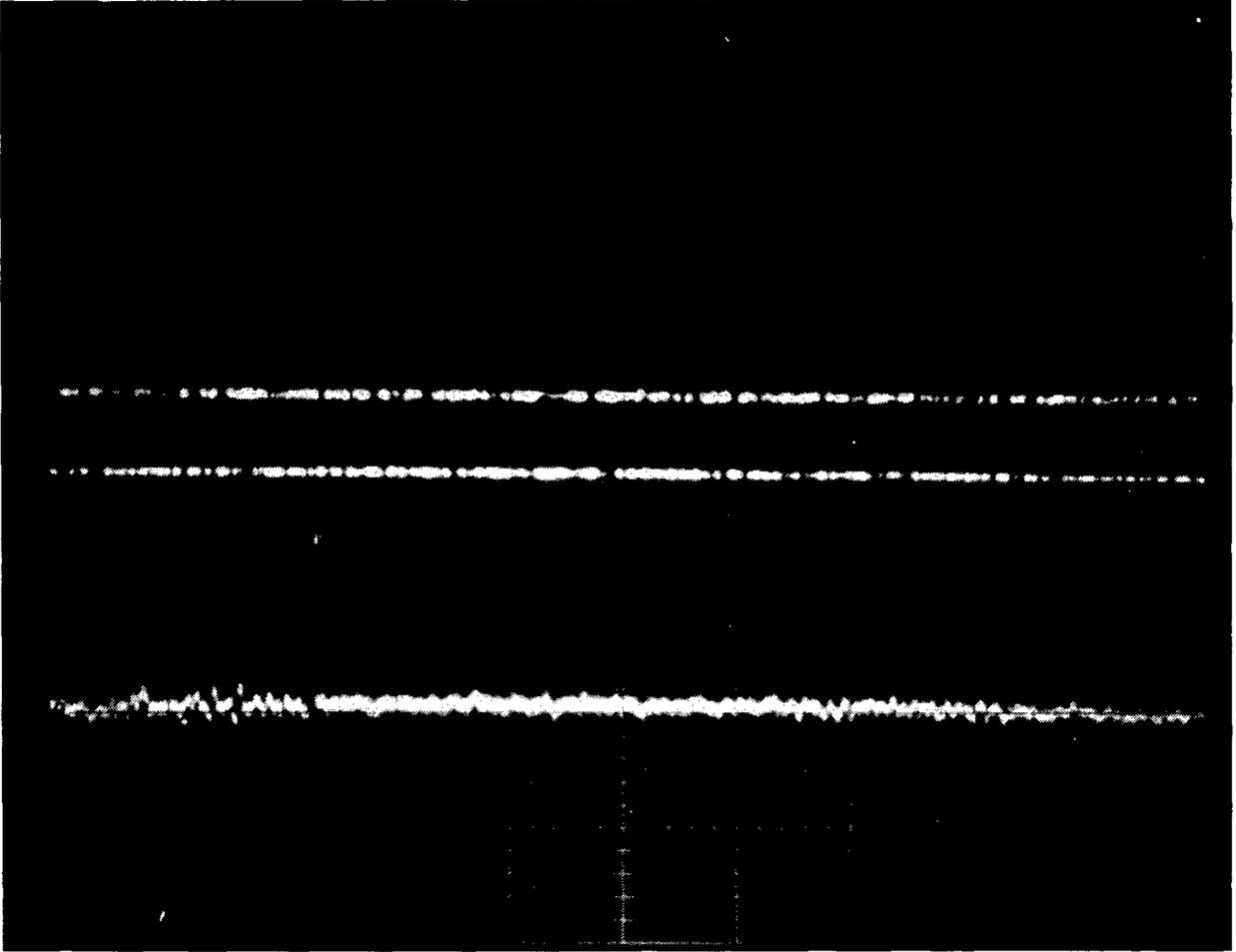


Fig. A7 — Autocorrelation of 255-bit PN code [8,6,5,3] starting with cleared signal registers, (100  $\mu$ s/div.,) same vertical scales as Fig. A5

## Appendix B

### SOFTWARE SIMULATIONS (Coauthored by D. Garfinkel)

The basic operation of serial and parallel load synchronized correlators is described in the main body of this report. Of particular significance is the method of properly forming the overall correlator output from among the outputs of the  $K$  component correlators. These techniques have been confirmed by means of a BASIC language computer simulation performed on an Osborne 1 personal computer, with the results plotted on an Okidata Microliner 82A line printer.

The parallel load approach was simulated first. The flow chart of this simulation program MULTPLX2 appears in Fig. B1 and its listing is shown in Fig. B2. The flow chart for the serial load simulation program MULTPLX3 is shown in Fig. B3 with its listing presented in Fig. B4. The variables used in the flow charts are defined in Table B1. A description of the flow charts follows.

#### Setting Up a Correlator Matrix

After a reference code has been selected and the number of correlators to be multiplexed has been provided, a three-dimensional correlator matrix is set up whose dimensions are  $k$  multiplexed correlators, each consisting of  $k$  subcorrelators, which are of length  $(2^{Lr})/k$  stages. Therefore, each multiplexed correlator can store a reference code  $2^{Lr}$  bits long.

#### Loading Reference Registers

Each of the reference registers of the multiplexed correlators is loaded the same way, so that the reference matrix does not vary between multiplexed correlators, and only a two-dimensional matrix is needed to describe it. The reference code is loaded such that:

The 1st,  $(1 + k)$ th,  $(1 + 2k)$ th, . . .  $(1 + 2^{Lr} - k)$ th bits of the reference code are loaded into the 1st subcorrelator;

The 2nd,  $(2 + k)$ th,  $(2 + 2k)$ th, . . .  $(2 + 2^{Lr} - k)$ th bits of the reference code are loaded into the 2nd subcorrelator; . . . , etc.

and finally

The  $k$ th,  $2k$ th,  $3k$ th, . . .  $(2^{Lr})$ th bits of the reference code are loaded into the  $k$ th subcorrelator.

#### Initializing Signal Code Bit Position

A pointer is placed at the first bit position in the signal code.

Up to this point, both multiplexing schemes (parallel and serial) are identical. The difference between them is in the method of loading the signal registers and sampling the correlation function.

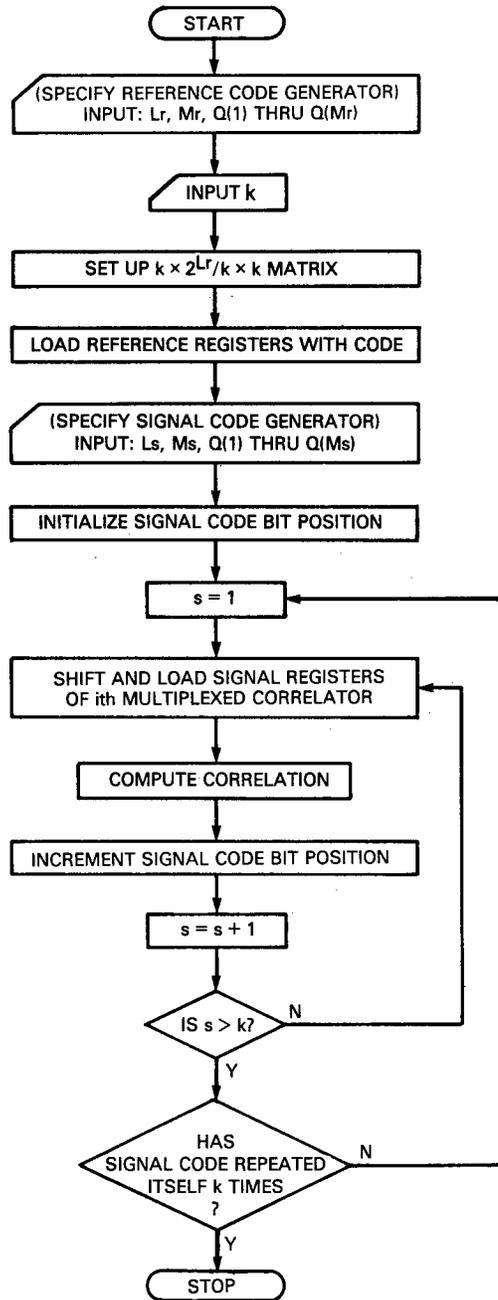


Fig. B1 — Flowchart for parallel load simulation program MULTPLX2

```

10 REM PROGRAM MULTPLX2
20 REM DIGITAL MULTIPLEXED CORRELATOR SIMULATOR
30 REM .....
40 REM
50 PRINT CHR$(26)
60 PRINT "INPUT REFERENCE CODE"
70 U=1
80 REM Subroutine to generate reference code is called.
90 GOSUB 720
100 PRINT:PRINT
110 INPUT "HOW MANY CORRELATORS TO MULTIPLEX";K
120 DIM SIG(K,(256/K),K), REF((256/K),K)
130 PRINT:PRINT
140 REM Subroutine to load reference registers is called.
150 GOSUB 1430
160 ERASE A,Q
170 PRINT "INPUT SIGNAL CODE"
180 U=2
190 REM Subroutine to generate signal code is called.
200 GOSUB 720
210 REM (2^L) bit-long code is converted to ((2^L)-1)
220 REM bit-long maximal-length sequence.
230 FOR I=1 TO (2^L)-1
240 A(I)=A(I+1)
250 NEXT I
260 IF K<>1 THEN 290
270 LPRINT "SERIAL";:LPRINT (2^L);:LPRINT "-BIT CORRELATOR"
280 GOTO 300
290 LPRINT K;:LPRINT "MULTIPLEXED CORRELATORS"
300 LPRINT:PRINT
310 REM Signal registers are initially loaded with 0's.
320 FOR S=1 TO K
330 FOR Y=1 TO K
340 FOR X=1 TO (2^L)/K
350 SIG(S,X,Y)=0
360 NEXT X
370 NEXT Y
380 NEXT S
390 T=1
400 REM Code bit position is initialized.
410 Z=(2^L)-(K-1)
420 IF Z=(2^L) THEN 440
430 GOTO 470
440 Z=1
450 REM Counter "C" determines number of shifts through
460 REM signal registers of correlators.
470 FOR C=1 TO 2*(2^L)/K
480 REM Variable "S" determines which multiplexed
490 REM correlator is being loaded with data (each
500 REM multiplexed correlator consists of K
510 REM individual correlators).

```

Fig. B2 - Listing for program MULTPLX2

```

520 FOR S=1 TO K
530 LET I=Z
540 REM Subroutine to shift correlator registers is called.
550 GOSUB 1590
560 REM Subroutine to load signal data into correlator
570 REM registers is called.
580 GOSUB 1700
590 REM Subroutine to compute correlation and plot
600 REM its function is called.
610 GOSUB 1910
620 Z=Z+1
630 IF Z=(2^L) THEN 650
640 GOTO 660
650 LET Z=1
660 NEXT S
670 NEXT C
680 LPRINT:LPRINT:LPRINT:LPRINT
690 END
700 REM .....
710 REM
720 REM SUBROUTINE CODE GENERATOR
730 REM GENERATES CODE GIVEN SHIFT REGISTER
740 REM LENGTH AND BIT POSITIONS NEEDED
750 REM
760 INPUT "HOW MANY BITS ARE IN REGISTER";L
770 DIM A(2^L), Q(L)
780 INPUT "DECIMAL EQUIV. OF INITIAL REGISTER STATE";S
790 FOR I=L TO 1 STEP -1
800 X(I)=S MOD 2
810 S=FIX(S/2)
820 NEXT I
830 INPUT "HOW MANY BITS NEEDED TO EXCLUSIVE-OR";M
840 PRINT "ENTER BIT POSITIONS ONE AT A TIME"
850 FOR V=1 TO M
860 INPUT P
870 Q(V)=P
880 NEXT V
890 IF U=2 THEN 920
900 LPRINT "REFERENCE CODE GENERATOR FEEDBACK TAPS: [";
910 GOTO 930
920 LPRINT "SIGNAL CODE GENERATOR FEEDBACK TAPS: [";
930 FOR V=1 TO M
940 IF V=1 THEN 960
950 LPRINT ", ";
960 LPRINT Q(V);
970 NEXT V
980 LPRINT "]"
990 LPRINT:LPRINT
1000 FOR I=1 TO (2^L)
1010 LET B=1

```

Fig. B2 (Continued) — Listing for program MULTPLX2

```

1020 FOR W=1 TO M
1030 LET R=Q(W)
1040 LET Y(B)=X(R)
1050 LET B=B+1
1060 NEXT W
1070 IF I>1 THEN 1110
1080 LET A(1)=0
1090 GOTO 1340
1100 REM Begin exclusive-or process.
1110 IF Y(1)=Y(2) THEN 1160
1120 LET Z=1
1130 IF M<>2 THEN 1200
1140 LET N=2
1150 GOTO 1280
1160 LET Z=0
1170 IF M<>2 THEN 1200
1180 LET N=2
1190 GOTO 1280
1200 LET N=3
1210 IF Y(N)=Z THEN 1240
1220 LET Z=1
1230 GOTO 1250
1240 LET Z=0
1250 IF N=M THEN 1280
1260 LET N=N+1
1270 GOTO 1210
1280 REM Exclusive-or process completed.
1290 LET A(I)=X(L)
1300 FOR G=(L-1) TO 1 STEP -1
1310 LET X(G+1)=X(G)
1320 NEXT G
1330 LET X(1)=Z
1340 NEXT I
1350 FOR I=0 TO (2^L)-16 STEP 16
1360 FOR J=1 TO 16
1370 LPRINT A(I+J);
1380 NEXT J
1390 LPRINT:LPRINT
1400 NEXT I
1410 LPRINT:LPRINT:LPRINT
1420 RETURN
1430 REM .....
1440 REM
1450 REM SUBROUTINE REFERENCE LOADER
1460 REM LOADS CODE INTO REFERENCE REGISTER
1470 REM OF CORRELATORS
1480 REM
1490 LET I=1
1500 FOR X=(2^L)/K TO 1 STEP -1
1510 FOR Y=1 TO K

```

Fig. B2 (Continued) — Listing for program MULTPLX2

```

1520 REF(X,Y)=A(I)
1530 I=I+1
1540 NEXT Y
1550 NEXT X
1560 RETURN
1570 REM .....
1580 REM
1590 REM SUBROUTINE SIGNAL SHIFTER
1600 REM SHIFTS SIGNAL REGISTERS OF CORRELATORS
1610 REM
1620 FOR X=(2^L)/K TO 2 STEP -1
1630 FOR Y=1 TO K
1640 SIG(S,X,Y)=SIG(S,X-1,Y)
1650 NEXT Y
1660 NEXT X
1670 RETURN
1680 REM .....
1690 REM
1700 REM SUBROUTINE SIGNAL LOADER
1710 REM LOADS INCOMING SIGNAL DATA INTO SIGNAL
1720 REM REGISTERS OF CORRELATORS
1730 REM
1740 FOR Y=1 TO K
1750 IF C=1 THEN 1770
1760 GOTO 1810
1770 IF I>((2^L)-K) THEN 1790
1780 GOTO 1810
1790 R=0
1800 GOTO 1820
1810 R=A(I)
1820 SIG(S,1,Y)=R
1830 I=I+1
1840 IF I=(2^L) THEN 1860
1850 GOTO 1870
1860 LET I=1
1870 NEXT Y
1880 RETURN
1890 REM .....
1900 REM
1910 REM SUBROUTINE COMPUTE AND PLOT
1920 REM COMPUTES THE AMOUNT OF CORRELATION AND
1930 REM PLOTS VALUES ON THE LINE PRINTER
1940 REM
1950 LET MATCH=0
1960 FOR Y=1 TO K
1970 IF Y<>1 THEN 2000
1980 LET R=((2^L)/K)-1
1990 GOTO 2010
2000 LET R=(2^L)/K
2010 FOR X=1 TO R

```

Fig. B2 (Continued) - Listing for program MULTPLX2

```

2020 IF SIG(S,X,Y)=REF(X,Y) THEN MATCH=MATCH+1
2030 NEXT X
2040 NEXT Y
2050 CORR=(2*MATCH)-((2^L)-1)
2060 ABSC=CINT(CORR*38/((2^L)-1))+20
2070 IF K<>1 THEN 2130
2080 FOR E=1 TO ABSC-1
2090 LPRINT " ";
2100 NEXT E
2110 LPRINT "*";
2120 GOTO 2170
2130 FOR E=1 TO ABSC-5
2140 LPRINT " ";
2150 NEXT E
2160 LPRINT "(";:LPRINT S;:LPRINT ")";:LPRINT "*";
2170 LET ABSC2=66
2180 IF A(T)=1 THEN ABSC2=76
2190 FOR E=ABSC+1 TO ABSC2-1
2200 LPRINT " ";
2210 NEXT E
2220 IF ABSC2=66 THEN 2250
2230 LPRINT "]"
2240 GOTO 2260
2250 LPRINT "["
2260 T=T+1
2270 IF T=(2^L) THEN 2290
2280 GOTO 2300
2290 T=1
2300 RETURN

```

Fig. B2 (Continued) — Listing for program MULTPLX2

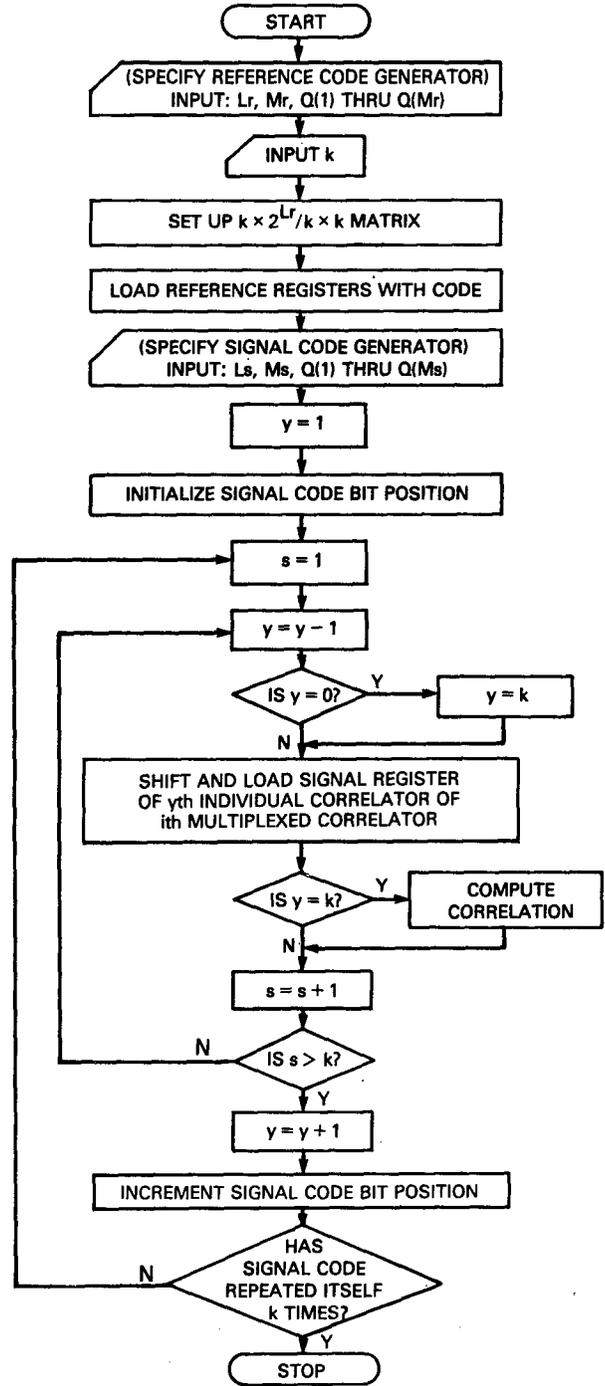


Fig. B3 — Flowchart for serial load simulation program MULTPLX3

```

10 REM PROGRAM MULTPLX3
20 REM DIGITAL MULTIPLEXED CORRELATOR SIMULATOR
30 REM (SERIAL-LOADING SCHEME)
40 REM .....
50 REM
60 PRINT CHR$(26)
70 PRINT "INPUT REFERENCE CODE"
80 U=1
90 REM Subroutine to generate reference code is called.
100 GOSUB 630
110 PRINT:PRINT
120 INPUT "HOW MANY CORRELATORS TO MULTIPLEX";K
130 DIM SIG(K,(256/K),K), REF((256/K),K)
140 PRINT:PRINT
150 REM Subroutine to load reference registers is called.
160 GOSUB 1360
170 ERASE A,Q
180 PRINT "INPUT SIGNAL CODE"
190 U=2
200 REM Subroutine to generate signal code is called.
210 GOSUB 630
220 REM  $(2^L)$  bit-long code is converted to  $((2^L)-1)$ 
230 REM bit-long maximal-length sequence.
240 FOR I=1 TO  $(2^L)-1$ 
250 A(I)=A(I+1)
260 NEXT I
270 IF K<>1 THEN 300
280 LPRINT "SERIAL";:LPRINT  $(2^L)$ ;:LPRINT "-BIT CORRELATOR"
290 GOTO 310
300 LPRINT K;:LPRINT "MULTIPLEXED CORRELATORS"
310 LPRINT:LPRINT
320 REM Signal registers are initially loaded with 0's.
330 FOR S=1 TO K
340 FOR Y=1 TO K
350 FOR X=1 TO  $(2^L)/K$ 
360 SIG(S,X,Y)=0
370 NEXT X
380 NEXT Y
390 NEXT S
400 T=1
410 Y=K+1
420 REM Code bit position is initialized.
430 Z=1
440 REM Counter "C" determines number of shifts through
450 REM signal registers of correlators.
460 FOR C=1 TO  $(2^L)*K$ 
470 REM Subroutine to shift correlator signal registers and load
480 REM incoming serial data is called.
490 GOSUB 1570
500 Z=Z+1
510 IF Z= $(2^L)$  THEN 530

```

Fig. B4 — Listing for program MULTPLX3

```

520 GOTO 540
530 LET Z=1
540 NEXT C
550 LPRINT:LPRINT:LPRINT:LPRINT
560 END
570 REM .....
580 REM
590 REM  SUBROUTINE CODE GENERATOR
600 REM  GENERATES CODE GIVEN SHIFT REGISTER
610 REM  LENGTH AND BIT POSITIONS NEEDED
620 REM
630 INPUT "HOW MANY BITS ARE IN REGISTER";L
640 DIM A(2^L), Q(L)
650 INPUT "DECIMAL EQUIV. OF INITIAL REGISTER STATE";S
660 FOR I=L TO 1 STEP -1
670 X(I)=S MOD 2
680 S=FIX(S/2)
690 NEXT I
700 INPUT "HOW MANY BITS NEEDED TO EXCLUSIVE-OR";M
710 PRINT "ENTER BIT POSITIONS ONE AT A TIME"
720 FOR V=1 TO M
730 INPUT P
740 Q(V)=P
750 NEXT V
760 IF U=2 THEN 790
770 LPRINT "REFERENCE CODE GENERATOR FEEDBACK TAPS: [";
780 GOTO 800
790 LPRINT "SIGNAL CODE GENERATOR FEEDBACK TAPS: [";
800 FOR V=1 TO M
810 IF V=1 THEN 830
820 LPRINT ",";
830 LPRINT Q(V);
840 NEXT V
850 LPRINT "]"
860 LPRINT:LPRINT
870 FOR I=1 TO (2^L)
880 LET B=1
890 FOR W=1 TO M
900 LET R=Q(W)
910 LET Y(B)=X(R)
920 LET B=B+1
930 NEXT W
940 IF I>1 THEN 980
950 LET A(1)=0
960 GOTO 1210
970 REM  Begin exclusive-or process.
980 IF Y(1)=Y(2) THEN 1030
990 LET Z=1
1000 IF M<>2 THEN 1070
1010 LET N=2

```

Fig. B4 (Continued) — Listing for program MULTPLX3

```

1020 GOTO 1150
1030 LET Z=0
1040 IF M<>2 THEN 1070
1050 LET N=2
1060 GOTO 1150
1070 LET N=3
1080 IF Y(N)=Z THEN 1110
1090 LET Z=1
1100 GOTO 1120
1110 LET Z=0
1120 IF N=M THEN 1150
1130 LET N=N+1
1140 GOTO 1080
1150 REM Exclusive-or process completed.
1160 LET A(I)=X(L)
1170 FOR G=(L-1) TO 1 STEP -1
1180 LET X(G+1)=X(G)
1190 NEXT G
1200 LET X(1)=Z
1210 NEXT I
1220 FOR I=0 TO (2^L)-16 STEP 16
1230 FOR J=1 TO 16
1240 LPRINT A(I+J);
1250 NEXT J
1260 LPRINT:LPRINT
1270 NEXT I
1280 LPRINT:LPRINT:LPRINT
1290 RETURN
1300 REM .....
1310 REM
1320 REM SUBROUTINE REFERENCE LOADER
1330 REM LOADS CODE INTO REFERENCE REGISTER
1340 REM OF CORRELATORS
1350 REM
1360 LET I=1
1370 FOR X=(2^L)/K TO 1 STEP -1
1380 FOR Y=1 TO K
1390 REF(X,Y)=A(I)
1400 I=I+1
1410 NEXT Y
1420 NEXT X
1430 RETURN
1440 REM .....
1450 REM
1460 REM SUBROUTINE SIGNAL SHIFTER/LOADER
1470 REM SHIFTS SIGNAL REGISTERS OF CORRELATORS
1480 REM AND LOADS THE INCOMING SERIAL DATA INTO THEM
1490 REM
1500 REM Variable "S" determines which multiplexed correlator
1510 REM is being clocked with data (each multiplexed

```

Fig. B4 (Continued) — Listing for program MULTPLX3

```

1520 REM      correlator consists of K individual correlators,
1530 REM      each one  $(2^L)/K$  bits long).  Variable "Y"
1540 REM      determines which of the individual correlators
1550 REM      of each multiplexed correlator is being clocked.
1560 REM
1570 FOR S=1 TO K
1580 Y=Y-1
1590 IF Y=0 THEN 1610
1600 GOTO 1620
1610 Y=K
1620 REM      Data in individual correlator is shifted one bit.
1630 FOR X= $(2^L)/K$  TO 2 STEP -1
1640 SIG(S, X, Y)=SIG(S, X-1, Y)
1650 NEXT X
1660 SIG(S, 1, Y)=A(Z)
1670 REM      Subroutine to compute and plot the amount of
1680 REM      correlation is called if the Kth individual
1690 REM      correlator of multiplexed correlator "S" was
1700 REM      just loaded with data.
1710 IF Y=K THEN 1730
1720 GOTO 1740
1730 GOSUB 1830
1740 NEXT S
1750 Y=Y+1
1760 RETURN
1770 REM .....
1780 REM
1790 REM      SUBROUTINE COMPUTE AND PLOT
1800 REM      COMPUTES THE AMOUNT OF CORRELATION AND
1810 REM      PLOTS VALUES ON THE LINE PRINTER
1820 REM
1830 LET MATCH=0
1840 FOR B=1 TO K
1850 IF B<>1 THEN 1880
1860 LET R= $((2^L)/K)-1$ 
1870 GOTO 1890
1880 LET R= $(2^L)/K$ 
1890 FOR X=1 TO R
1900 IF SIG(S, X, B)=REF(X, B) THEN MATCH=MATCH+1
1910 NEXT X
1920 NEXT B
1930 CORR=(2*MATCH)- $((2^L)-1)$ 
1940 ABSC=CINT(CORR*38/ $((2^L)-1)$ )+20
1950 IF K<>1 THEN 2010
1960 FOR E=1 TO ABSC-1
1970 LPRINT " ";
1980 NEXT E
1990 LPRINT "*";
2000 GOTO 2050
2010 FOR E=1 TO ABSC-5

```

Fig. B4 (Continued) — Listing for program MULTPLX3

```

2020 LPRINT " ";
2030 NEXT E
2040 LPRINT "(";:LPRINT S;:LPRINT ")";:LPRINT "*";
2050 LET ABSC2=66
2060 IF A(T)=1 THEN ABSC2=76
2070 FOR E=ABSC+1 TO ABSC2-1
2080 LPRINT " ";
2090 NEXT E
2100 IF ABSC2=66 THEN 2130
2110 LPRINT "]"
2120 GOTO 2140
2130 LPRINT "["
2140 T=T+1
2150 IF T=(2^L) THEN 2170
2160 GOTO 2180
2170 T=1
2180 RETURN

```

Fig. B4 (Continued) — Listing for program MULTPLX3

Table B1 — Variables Used in Flow Charts

<i>Lr, Ls</i>	Number of stages in shift register to generate reference/signal PN code
<i>Mr, Ms</i>	Number of feedback taps on reference/signal PN code generator shift register to be modulo-2 added together
<i>Q(1) through Q(Mr), Q(1) through Q(Ms)</i>	Locations of feedback taps on reference/signal PN code generator shift registers
<i>k</i>	Number of correlators to multiplex
<i>s</i>	Variable. Determines which multiplexed correlator (from 1 to <i>k</i> ) is being accessed. [Each multiplexed correlator consists of <i>k</i> subcorrelators]
<i>Y</i>	Variable. Determines which subcorrelator (from 1 to <i>k</i> ) in a multiplexed correlator is being loaded.

## Parallel Load

In the parallel load scheme, the signal data are loaded in groups of  $k$  bits. Each of these signal bits is loaded into the first stage of each of the  $k$  signal registers of a multiplexed correlator. The correlation in that multiplexed correlator is calculated and will be the latest contribution to the overall correlation output. Then, the signal data pointer is incremented one bit, and the latest  $k$  signal data bits (starting with the bit the pointer specifies) are loaded into the signal registers of the next multiplexed correlator. This process continues with the latest group of  $k$  data bits being loaded into succeeding multiplexed correlators (with the first multiplexed correlator succeeding the  $k$ th).

## Serial Load

The philosophy behind the serial loading scheme is to load the signal data as if there were a one-count clock delay in the divide-by- $k$  ring counters from one multiplexed correlator to the next. In this technique, the first signal bit is loaded into the first subcorrelator of the first multiplexed correlator, the  $k$ th subcorrelator of the second multiplexed correlator, the  $(k - 1)$ th subcorrelator of third multiplexed correlator, and finally the 2nd subcorrelator of the  $k$ th multiplexed correlator. Succeeding signal bits are loaded into the succeeding subcorrelators of each multiplexed correlator (with the first subcorrelator always succeeding the  $k$ th). Whenever a signal bit is loaded into the  $k$ th subcorrelator of a multiplexed correlator, the output of that multiplexed correlator will be the latest contribution to the overall correlation output.

## Computing the Correlation

Each of the  $k$  signal registers is compared bit-for-bit with its corresponding reference register. The correlation is calculated as the total number of bit agreements minus the total number of bit disagreements. The correlation is computed in only one multiplexed correlator at a time and is printed out as the latest contribution to the overall correlation.

## Simulation Results

Using the programs described above, computer simulations were performed for autocorrelation of the 255-bit PN  $m$ -sequence [8,4,3,2] for parallel and serial load synchronized multiplexed correlators. Outputs were also obtained from a simulation of a correlator without multiplexing that would be operating at  $k$  times the speed of the multiplexed correlator devices. These multiplexed correlation simulation programs implement the data selection technique for forming the correct overall output from samples of outputs of component correlators as described in the main body of this report. The results of these simulations are presented in the plots of Fig. B5 as formed from individual printer generated computer plots. The  $m$ -sequence PN code [8,4,3,2] appears in Fig. B5 (a). As would be the case with an actual realization, the simulated correlators actually provide for 256 bits. The 255-bit autocorrelation is handled by masking the correlation of the oldest signal bit in the correlator. For example, the TRW 64-bit digital correlator devices have a masking function for such a purpose. The correlation output is thus determined by comparison of the most recent 255 signal bits with the 255-bit PN reference code.

The output corresponding to that of a parallel load synchronized multiplexed correlator appears in Fig. B5 (b). Successive correlation outputs are selected from succeeding correlators as discussed in the report. The plot indicates in parentheses the specific multiplexed correlator from which each output sample was obtained. For the parallel load case, multiplexed correlator  $i$  produces the current contribution to the overall output when it receives the most recent  $k = 4$  bits. The parallel load correlation of Fig. B5 (b) began with the first bit of the 255-bit [8,4,3,2]  $m$ -sequence corresponding to the first basic clock pulse following a correlator 1 shift pulse (i.e., in synchronization with correlator 1). It can be seen from the timing diagram of Fig. 13 (in the main body of this report) that correlator 2 provides the

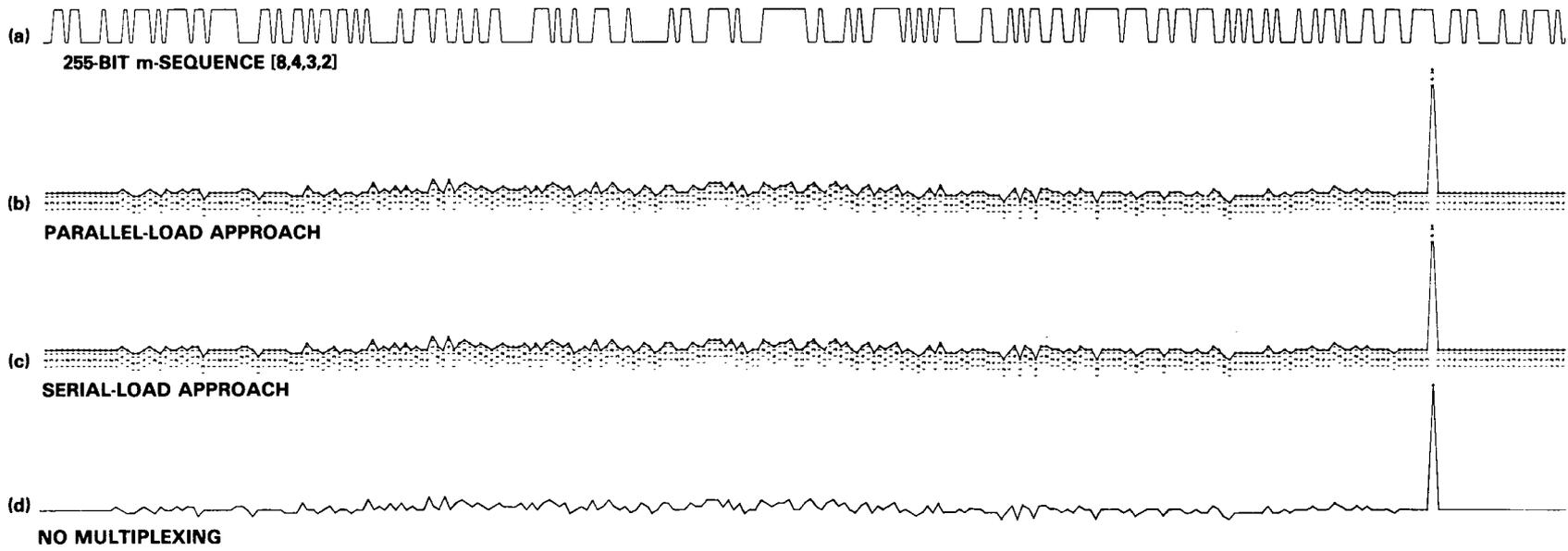


Fig. B5 — Multiplexed correlator simulation results ( $i = 1$ )

corresponding overall multiplexed correlator output sample. When the 255th and last bit of the  $m$ -sequence occurs, correlator 4 provides the peak autocorrelation output sample of the overall multiplexed correlation.

The output, corresponding to that of a serial load synchronized multiplexed correlator, appears in Fig. B5 (c). Again, the selection of outputs from successive correlators to form the overall output is as indicated in parentheses. For the serial load case, each correlator output sample is obtained from the correlator that had the most recent signal bit clocked into its  $k$ th subcorrelator. The serial load correlation of Fig. B5 (c) began with the first bit of the  $m$ -sequence corresponding to a clock pulse for subcorrelator 1 of correlator 1 (i.e., in synchronization with correlator 1). From the timing diagram of Fig. 8 (in the main body of this report) it can be seen that in this case, also, correlator 2 provides the corresponding overall multiplexed correlator output sample. As in the parallel load case, the overall correlator output sample corresponding to the 255th bit of the  $m$ -sequence is provided by correlator 4.

For comparison purposes, the autocorrelation output for the [8,4,3,2] code from a simulated 256-bit digital correlator without multiplexing was also computed and plotted as shown in Fig. B5 (d). It can be seen that the simulation results obtained for the parallel and serial load multiplexed correlators agree exactly with that of the unmultiplexed correlator.

From the results of Fig. B5, we see for either the serial load or parallel load case, that when the first bit of the 255-bit  $m$ -sequence was in synchronization with correlator 1, the autocorrelation peak sample occurred in correlator 4. In general, when the first bit of an  $N$ -bit sequence synchronizes with the  $i$ th of the  $k$  multiplexed correlators, the autocorrelation peak output occurs in correlator  $m$  where  $m = ((N + i - 1) \bmod k) + 1$ , ( $x \bmod y$  indicates the remainder when  $x$  is divided by  $y$ ). With the correlators numbered 0 through  $k - 1$ ,  $m = (N + i) \bmod k$ . However, with the correlator numbering (1 through  $k$ ) used here,  $m$  must be modified by subtracting 1 under the modulo  $k$  operation and adding it back in after the modulo  $k$  operation.

Using the simulation program for the serial load case with the 255-bit [8,4,3,2]  $m$ -sequence, we ran simulations for synchronization of the first bit of the code with each of the multiplexed correlators (i.e.,  $i = 1, 2, 3, 4$ ). The results appear in Fig. B6 (a), (b), (c), and (d), respectively. It can be seen that in these cases the autocorrelation peak samples come from correlators 4,1,2,3 respectively. Note that synchronization of the signal sequence with correlator  $i$  can also be looked upon as the sequence being out of synchronization (with correlator 1) by  $j = i - 1$ . Thus the plots of Fig. B6 (a), (b), (c), and (d) also correspond to the signal synchronization conditions  $j = 0, 1, 2, 3$ , respectively. It can therefore be seen that the ordering of correlators that sequentially contribute samples to the overall correlation output is circular. Thus the actual numbering of these  $k$  correlators is only relative.

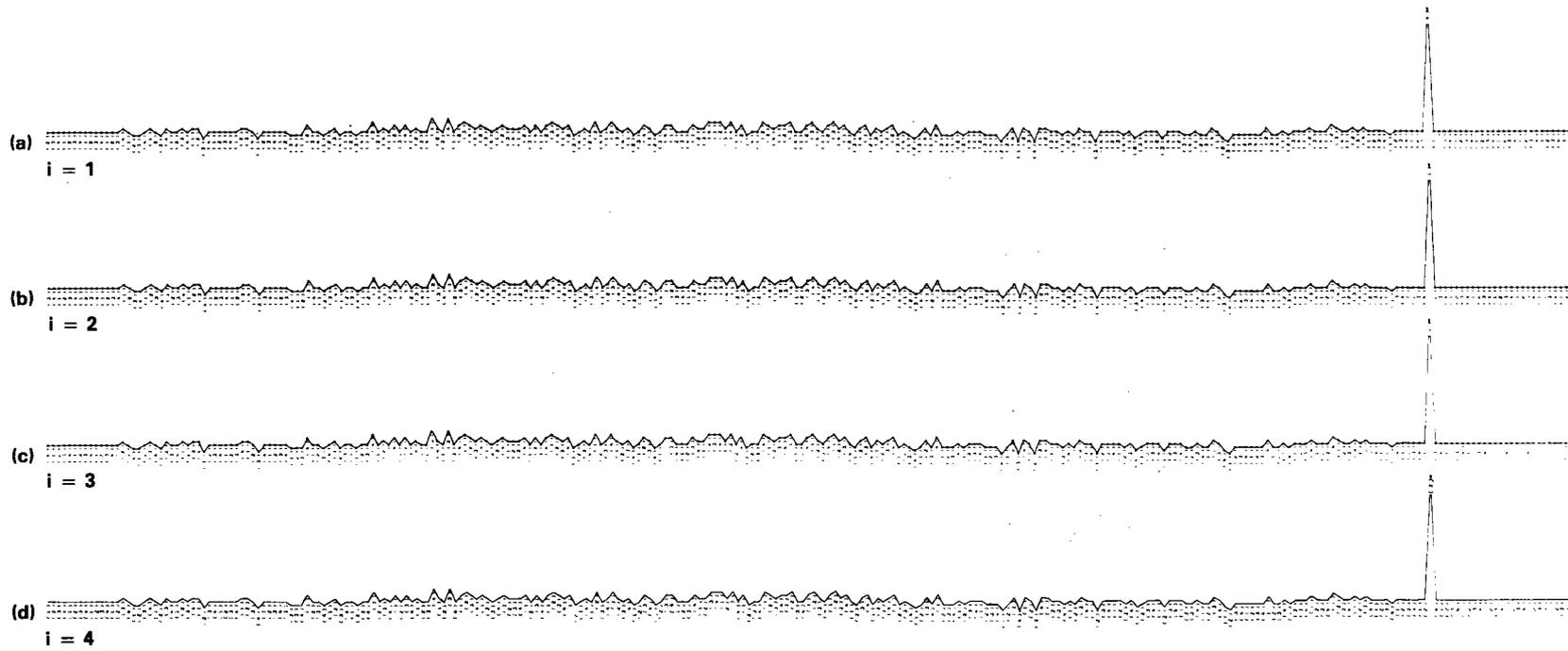


Fig. B6 — Serial load multiplexed correlator simulation results for various synchronization conditions ( $i = 1, 2, 3, 4$ )