01 Jan 1981

A Performance Study of a Microcomputer-IImplemented Microcomputer-Implemented FSK Receiver

Paul D. Stigall
Missouri University of Science and Technology, tigall@mst.edu

Rodger E. Ziemer
Missouri University of Science and Technology

Van T. Pham

Recommended Citation

P. D. Stigall et al., "A Performance Study of a Microcomputer-IImplemented Microcomputer-Implemented FSK Receiver," *IEEE Micro*, vol. 1, no. 1, pp. 43 - 51, Institute of Electrical and Electronics Engineers; Computer Society, Jan 1981.
The definitive version is available at https://doi.org/10.1109/MM.1981.290821

Microprocessors are finding new usefulness in digital signal processing. Here, a frequency shift-keyed receiver is implemented on a two-processor system.

A Performance Study of a Microcomputer-Implemented FSK Receiver

Paul D. Stigall
Rodger E. Ziemer
Van T. Pham
University of Missouri-Rolla

With the advance of large-scale integrated circuit technology, microcomputers have found many applications for which previously they were technically or economically unfeasible. An area of much interest is digital signal processing, where computers are suitable for many tasks, such as filtering or control. One such application is a frequency-shift-keyed receiver implemented on a microcomputer system, the System 80/20, manufactured by Intel Corporation. The system includes two processors – SBC-80/20 and SBC-80/05—a 16K × 8-bit memory board, and a math unit. A fast Fourier transform, or FFT, algorithm implements the two bandpass filters required for the receiver. In this, as in any new application, an understanding of the hardware system's capability is essential to achieve optimum performance. This study investigates the factors affecting system performance in terms of execution statistics of the concurrently programmed receiver and the probability of error of the detection algorithms versus signal-to-noise ratio.

Signal detection system

From the many receiver structures and signaling schemes available, a frequency-shift-keyed, or FSK, receiver[1] was chosen because of the attractiveness of many FFT algorithms in spectral analysis. Figure 1, a block diagram of the FSK receiver, shows the two bandpass filters, implemented by a Gold-Bially FFT algorithm.[2] If necessary, a window can be applied to the input data to reduce the spectral leakage associated with a finite observation interval. The effects of windows have been reviewed and tabulated by Harris.[3]

The receiver takes as input a block of 64 signal-plus-gaussian-noise samples and outputs an indication of whether or not the signal is present. In addition to FFT and detector algorithms, the microcomputer program includes signal and noise generation, timing measurement, and calculation of the probability of error.

The sampled signal input is either

$$s_1(n) = \exp[j2\pi f_0 n/64]$$

or

$$s_2(n) = \exp[j2\pi(f_0 + \Delta f)n/64]$$

where $\Delta f = 32$; $n = 0, 1, \ldots, 63$; and f_0 is a real variable between 0 and 64.

The noise is generated by a technique[4] which can be described as follows:

(1) Pick 12 initial random numbers between $[-(1/2), 1/2]$ from a table of random numbers and store them in an array X.
(2) Generate 12 new numbers $x(i)$, $i = 1,2,\ldots,12$ by the rule $x(i) = [x(i-1) + x(i-12)]$ modulo 1/2, where modulo 1/2 means subtracting 1 from the

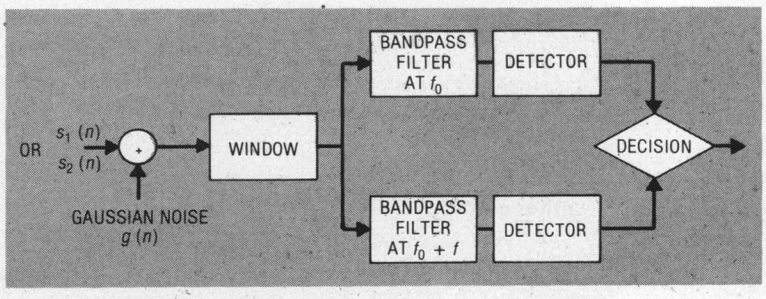


Figure 1. Block diagram of an FSK receiver.

0272-1732/81/0200-0043$00.75 © 1981 IEEE

sum if it is greater than 1/2 or adding 1 to the sum if it is less than $-1/2$.

(3) Add 12 $x(i)$'s together to form one part (real or imaginary) of a complex noise sample.

$$g(n) = \sum_{i=1}^{12} x(i)$$

(4) Repeat (2) and (3) 127 times to provide 64 complex noise samples, one to be added to each signal sample.

Since $x(n)$ has a uniform distribution in $[-(1/2),1/2]$, its variance is 1/12. The noise samples $g(n)$ have, approximately, a gaussian distribution (by the central limit theorem) and unity variance. To make this variance adjustable, each noise sample in the sequence is multiplied by a parameter, denoted B, before being added to the input signal.

In this article, the input signal-to-noise ratio, or *SNR*, to the FFT is defined as

$$SNR_i = \frac{\text{received signal power}}{\text{variance of each noise sample}} = \frac{1}{2B^2}$$

The values of B were chosen to allow the *SNR* to be varied in steps of 2 dB.

Although roundoff noise, Welch[5] points out, is added in the FFT, its effects will not be considered here.

The detectors perform one of the following detection algorithms:

(1) Tone detection: The magnitudes of the two predetermined spectral outputs of the FFT are compared with each other to determine which one was transmitted.

(2) Sum-of-magnitudes: The sums of the magnitudes of the frequency components in the first and second halves of the FFT output are compared with each other to determine which signal was sent.

(3) Energy detection: Same as (2), except that the sums of the squares of the magnitudes are compared.

The latter two algorithms would be used if the frequency of the received signal were uncertain.

The magnitude of a frequency component is approximated[6] by the following formula:

$$|X_n| = \text{Max}[\text{Re}\,X_n, \text{Im}\,X_n] + \tfrac{1}{2}\text{Min}(\text{Re}\,X_n, \text{Im}\,X_n)$$

where Re X_n and Im X_n denote the real and imaginary parts, respectively, of the complex number X_n, and

$$\text{Max}(a,b) = \begin{cases} a \text{ if } a \geq b \\ b \text{ if } a < b \end{cases} \qquad \text{Min}(a,b) = \begin{cases} a \text{ if } a < b \\ b \text{ if } a \geq b \end{cases}$$

Ignoring roundoff noise and assuming independent noise samples at the input of the FFT, the input *SNR* to the detector is given by

$$SNR_d = SNR_i + PG$$

where SNR_i is the *SNR* at the FFT input and PG is the processing gain of the window as tabulated by Harris.[3] All three terms are in dB.

Digital FSK receivers

The signaling format processed by the detection algorithm described in this article is referred to as digital (not to be confused with the digital processing used to implement the receiver) because one of a finite set of signaling waveforms can be detected. Although the number of possible signaling waveforms we have assumed is two, and therefore the term *binary signaling* is applied, one could use a system employing any finite number of waveforms, thus allowing the possibility of conveying one of several possible messages. Since the discrete set of signaling waveforms used here is obtained by varying the frequency of a sinusoidal carrier in discrete steps, the communication system is referred to as *frequency-shift-keyed* (FSK). Common terminology for the two frequencies employed for binary FSK are "marks" and "spaces."

The advantages of the FSK signaling format and the FFT-implemented receiver described in the article are the following:

(1) One of several possible messages can be sent, although only two were employed here.

(2) The receiver is easily implemented as an algorithm on a microcomputer which could be jointly used for other tasks (the signal and noise generation performed in this study would not be required in practice, thus freeing that portion of the microcomputer for other tasks).

(3) The FSK receiver implemented works without the necessity of a phase reference, and is thus termed *noncoherent*. Although simpler than a receiver requiring estimation of carrier phase, some loss in detectability is thereby imposed.

(4) The FSK receiver implemented uses a 64-point FFT and sends only one of two possible frequencies at one time; thus, a large degree of frequency uncertainty can be tolerated when the sum-of-magnitudes and energy-detection algorithms are employed. Alternatively, if the frequencies are known with precision at the receiver, the tone-detection algorithm allows the potential use of 32 possible frequencies to represent a mark or space. Such so-called spread spectrum signaling procedures are advantageous in interference environments.

(5) The FSK signaling used in this study employed wide deviation between mark and space frequencies, thus allowing the flexibility mentioned above. A smaller frequency separation between mark and space tones would allow use of a smaller FFT and faster processing at the expense of poorer noise performance.

Receiver and signal generation program

This program (see Appendixes A and B) is written to execute on a two-CPU microcomputer system.[7] The first part runs on the SBC 80/20 (designated CPU 1); it performs the signal and noise generation, unscrambles the results, and performs detection and input/output, including reading parameters and printing out the results on a Teletype.* The second part, a 64-point radix-4 FFT program, runs on the SBC 80/05 (designated CPU 2). Three parameters requested by the program at run time are the signal frequency f_0, the number of passes, and the type of detection algorithm. Figures 2 and 3 show the flowcharts for parts I and II of this program. Some of its features are described below.

Data representation. Each complex data sample is represented as two 16-bit fixed-point two's-complement binary numbers. The binary point is between the two bytes. The more significant byte contains the sign and the integer; the other byte contains the fraction.

Data structure. Data are generated and stored in a two-dimensional array of columns, starting with the leftmost column.

Input-output. An input routine reads from a Teletype the parameters entered in ASCII code. Placing a zero before each parameter is not necessary. The specified signal frequency should be a real number between 0 and 64, the fraction being preferably a multiple of 0.25 because the input routine will round it off to the nearest quarter. The routine will recognize the last two digits preceding the decimal point as the integer, and the last two digits before a carriage return as the fraction. The maximum number of passes is 9999, since only the four digits entered immediately before a carriage return are considered valid. One of the three detection algorithms—tone, sum-of-magnitudes, or energy—can be entered as shown or abbreviated as "T," "S," or "E."

*Teletype is a registered trademark of the Teletype Corporation.

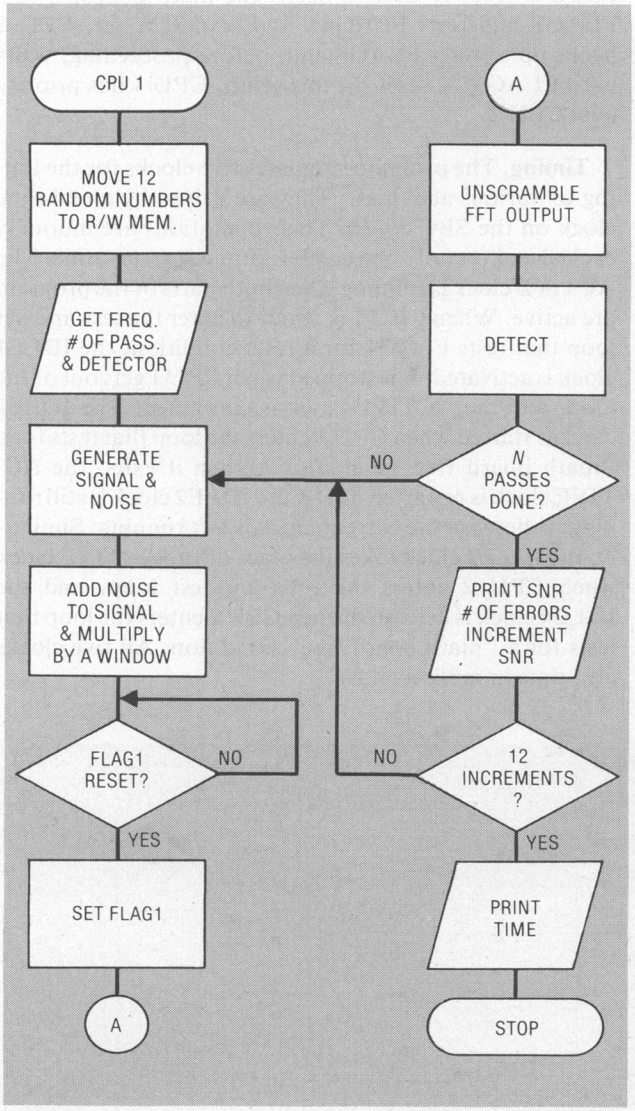


Figure 2. Part I of the signal-detection simulation program (executes on SBC 80/20).

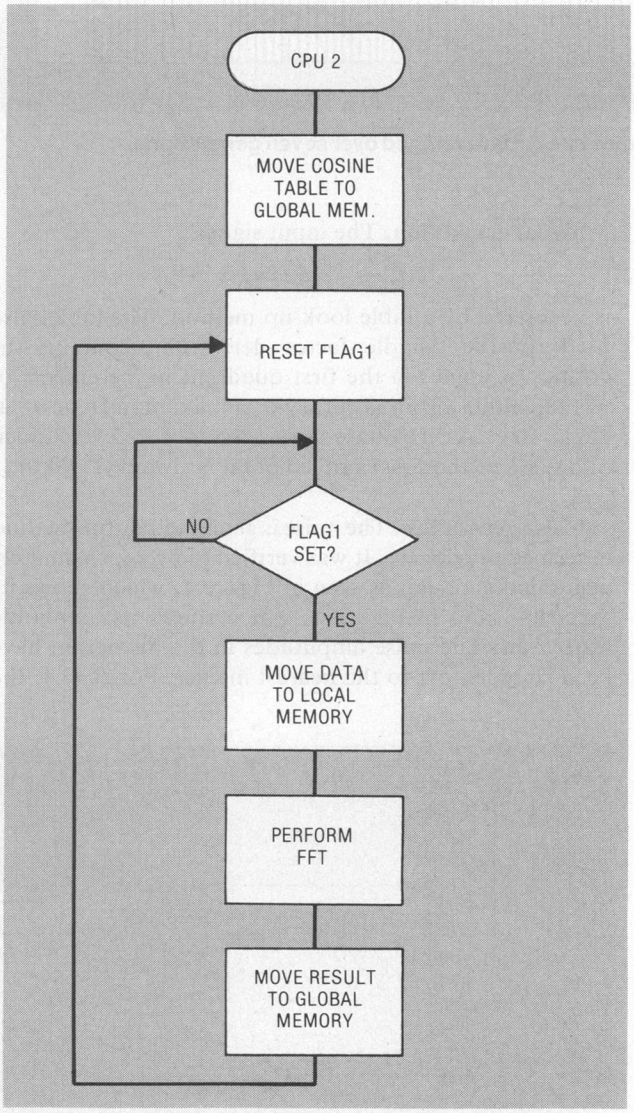


Figure 3. Part II of the signal-detection simulation program (executes on SBC 80/05).

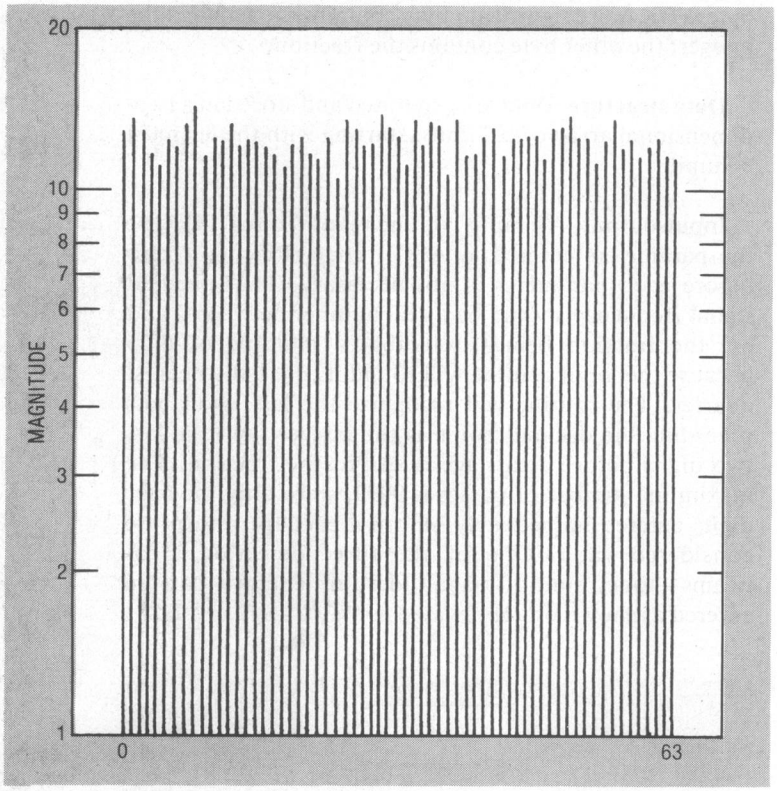


Figure 4. Noise spectrum averaged over seven generations.

Signal generation. The input signal

$$s(n) = \exp[j2\pi f_0 n/64]$$

is generated by a table look-up method. The table, also used for the twiddle factor derivation, contains the cosines of angles in the first quadrant in increments of $\pi/128$, with a 16-bit accuracy or a maximum error of $\pm 7.6 \times 10^{-6}$. A zero phase shift is assumed for every block of data; i.e., the first sampled point is always (1.0,0.0).

Noise generation. The noise is generated by the method described previously. It was verified to be approximately white and gaussian as seen in Figure 4, which shows its spectrum, and in Figure 5, which shows its amplitude histogram. The noise amplitudes in the histogram have been rounded off to the nearest integer. For $B = 1$, the average of the distribution was found to be $\bar{x} = -0.024$ and the variance was found to be $\sigma^2 = 0.975$.

Program communications. Communication between the two parts of the program is accomplished by means of three flags. When first started or after an FFT is completed, CPU 2 resets FLAG1 and tests it repeatedly. Once CPU 2 finds FLAG1 set, it gets the data, performs the FFT, and returns the result. When CPU 1 completes the signal generation and windowing, it tests FLAG1 for a reset condition. If true, it immediately sets FLAG1 and performs the unscrambling and detection on the FFT output just completed by CPU 2. The relation of FLAG 1 to CPU 1 and CPU 2 is shown in Figures 2 and 3.

The other two flags, FLAGM and FLAGS, are used to prevent possible deadlock caused by math board utilization and to determine priority. To use the math board, CPU 1 must set FLAGM and test FLAGS. If FLAGS is set, CPU 2 is either using or attempting to use the math board, so CPU 1 must wait in a loop, until FLAGS is reset, before proceeding. When CPU 1 has finished using the math board, it resets FLAGM to signal that the math board is free. If CPU 2 needs the math board, it sets FLAGS and tests FLAGM. If FLAGM is set, CPU 2 backs up to reset FLAGS and, before proceeding, waits until FLAGM is reset. In this setup, CPU 1 has priority over CPU 2.

Timing. The program simulates five clocks for the timing of various activities. They are driven by a real-time clock on the SBC 80/20. Their operations are mutually exclusive; i.e., only one clock is running at any time. The ACTIVE clock is running when both parts of the program are active. When CPU 1 is about to enter the test-and-set loop that tests FLAG1 for a reset condition, the IDLE1 clock is activated. It is stopped when CPU 1 gets out of the loop, and the ACTIVE clock is continued. The IDLE2 clock is started when CPU 1 enters the loop that tests for a "math board free" condition. When it exits, the ACTIVE clock is restarted only if the IDLE2 clock is still running; otherwise the current clock is left running. Similarly, the IDLE3 clock takes the place of the ACTIVE clock when CPU 2 enters the reset-and-test loop, and the IDLE4 clock is activated when CPU 2 enters the loop that tests for a "math board free" condition. All five clocks give time in milliseconds.

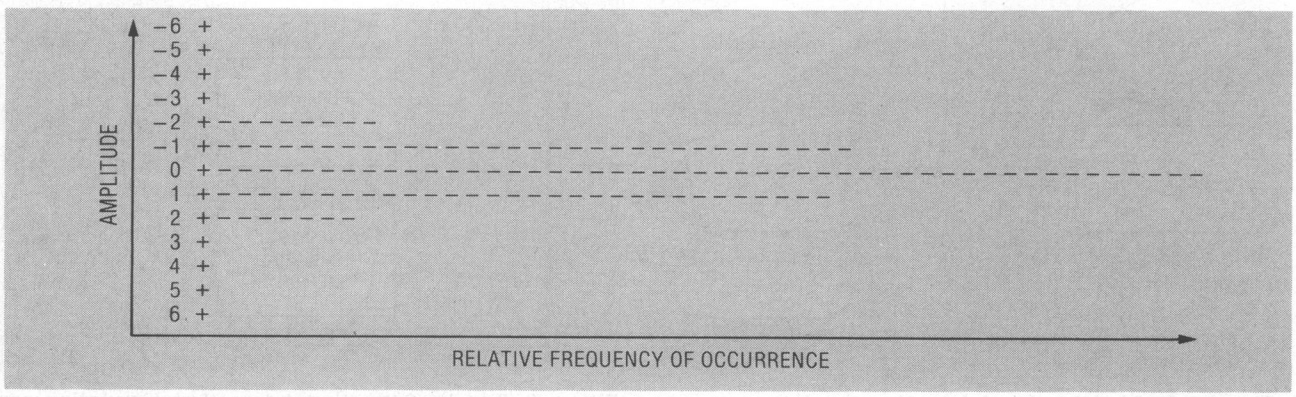


Figure 5. Noise histogram averaged over 12,000 generations.

Experimental investigation

The average execution time for one cycle of the signal-detection program on the System 80/20 is shown in Figure 6. The term "active" in the figure means that the two parts of the program are executing in parallel, while "idle" means that one CPU is waiting for the other or for the math board. There is one unexpected result: the time for one detection cycle of the system employing the tone-detection algorithm should be less than that for the others, since the magnitudes of only two frequency components need to be calculated for the tone-detection algorithm, compared with the magnitudes of 64 components for the other two. But the relative times given in Figure 6 are just the opposite. We suspect that this peculiar result is an effect of concurrent programming; for example, when CPU 1 and CPU 2 try to access the Multibus too often, the time required for a detection cycle increases.

Table 1 shows the timing of the principal subroutines in this program. The efficiency of concurrent programming is clear. If we add up the times in the table, we see that the total time required if only one CPU is available has been reduced 45 to 49 percent by using two CPUs.

Activities. A counting routine was inserted into the programs at strategic points to count the number of memory accesses, N_{MA}, the number of additions, N_A, and the number of multiplications, N_M, for one FFT or one cycle of the detection program.

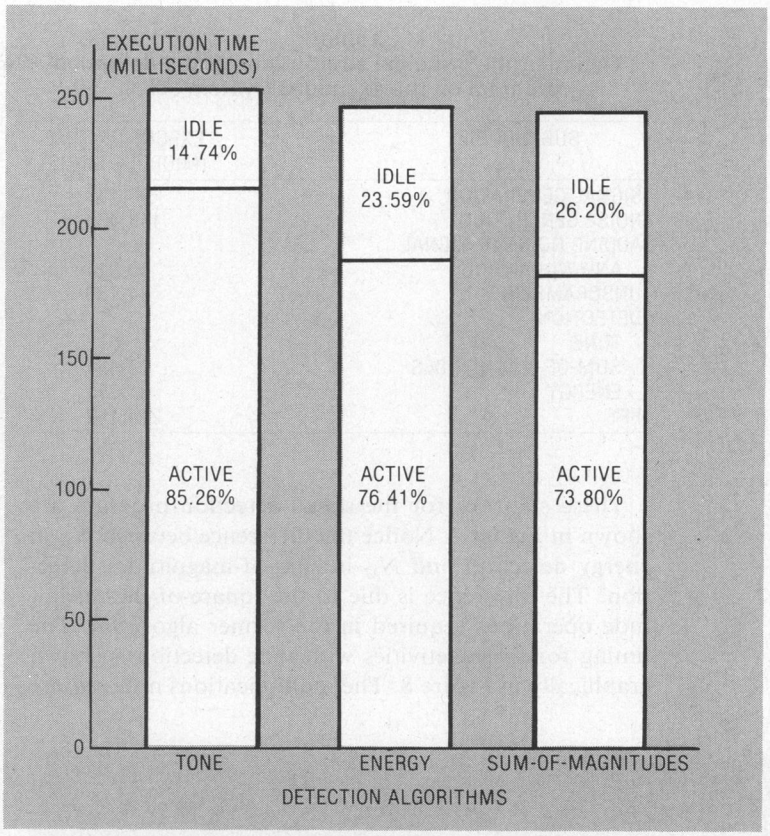


Figure 6. Execution time for one detection cycle of the detection system with three detection algorithms.

Table 1.
Timing of the principal subroutines of the detection program on the expanded System 80/20.

SUBROUTINE	EXECUTION TIME (MICROSECONDS)
SIGNAL GENERATION	28,167
NOISE GENERATION	143,480
ADDING NOISE TO SIGNAL AND WINDOWING	50,914
UNSCRAMBLING	24,130
DETECTION	
TONE	672
SUM-OF-MAGNITUDES	22,484
ENERGY	31,612
FFT	214,152

These statistics for the signal-detection program are shown in Figure 7. Notice the difference between N_M in energy detection and N_M in sum-of-magnitudes detection. The difference is due to the square-of-the-magnitude operations required in the former algorithm. The timing for these activities with tone detection is shown graphically in Figure 8. The multiplications now require 11 percent of the total time, less than the time for either memory accesses or additions. The percentages of time taken by the subroutines that constitute one detection cycle are presented in Figure 9 for the three detection algorithms. The FFT time is not shown, since the FFT executes in parallel with these subroutines. For all three algorithms, noise generation is the most time-consuming task.

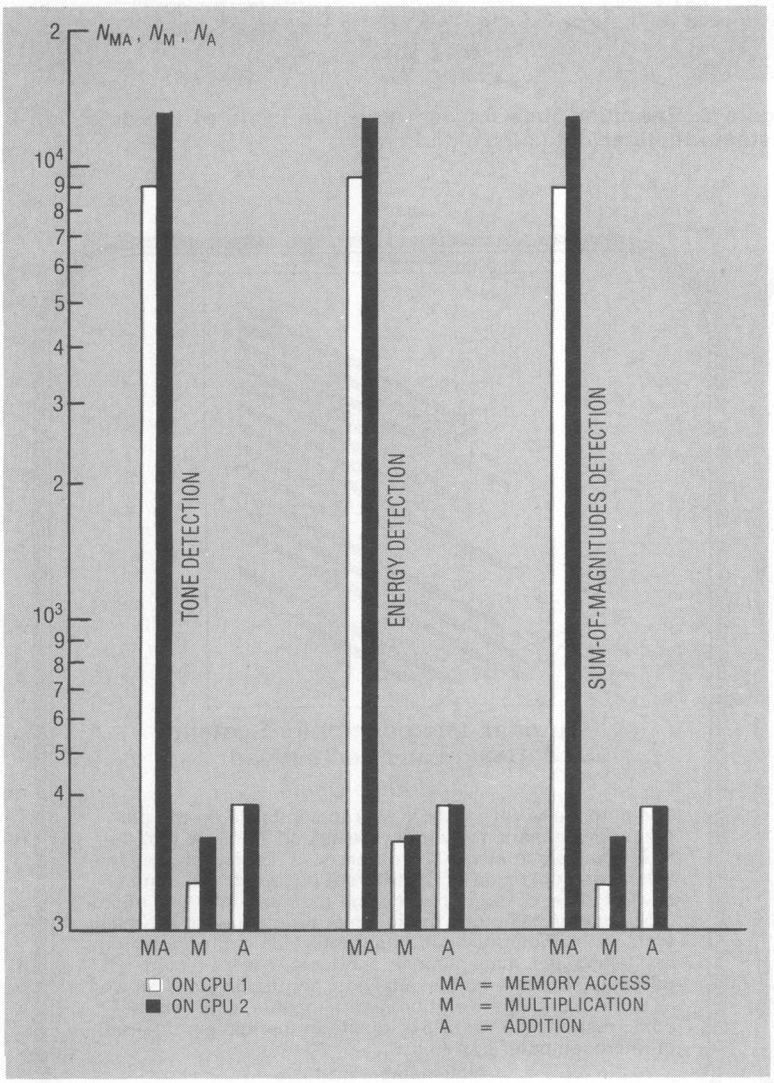


Figure 7. Statistics for the detection simulation program on System 80/20.

System performance. Various combinations of parameters were entered to study their effects on system performance in terms of the probability of error versus signal-to-noise ratio. The following results are grouped according to these combinations.

Effects of detection algorithms. Figure 10 shows the probability of error versus *SNR* at the FFT input for the three detection algorithms. If the noise samples at the FFT input are independent, the *SNR* at the detector input is equal to the *SNR* at the FFT input plus the processing gain, which is 18.06 dB for a 64-sample rectangular window. The tone-detection algorithm has the best performance. The energy-detection algorithm, which takes more time than the sum-of-magnitudes-detection algorithm, performs better than the latter. As expected, both energy detection and sum-of-magnitudes detection are worse than the tone-detection algorithm, which sums the least noise at the FFT output. The first two produce effects typical of the performance of a receiver whose received signal frequency is uncertain. The theoretical error probability versus *SNR* is also shown for a noncoherent FSK receiver. As expected, its performance results are better than those of the tone-detection simulation by one to two dB, depending on the *SNR*.

Effects of windows. Error probability versus *SNR* for five types of windows is shown in Figure 11. The rectangu-

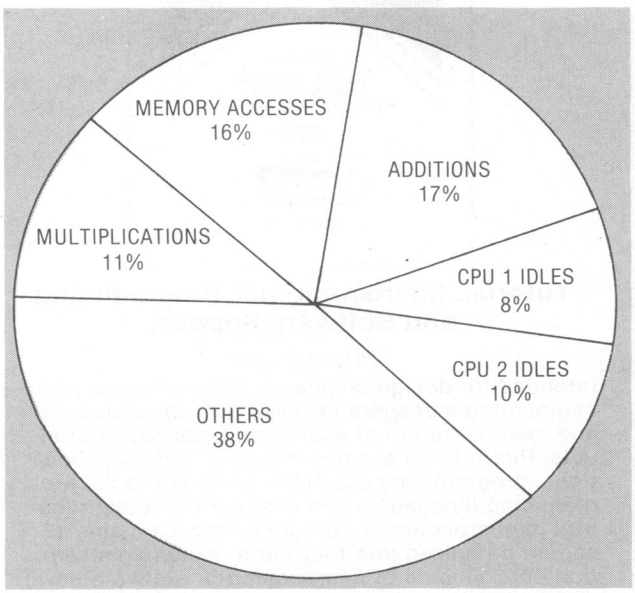


Figure 8. Signal-detection activities in percent of total execution time on System 80/20 with rectangular window and tone detection.

lar window has the best performance, as expected, since the processing gains of the other windows are less than zero dB (varying from -1.24 to -2.38 dB). The SNR_d is degraded by a corresponding amount. Although the other windows may reduce the spectral leakage (which is the result of a non-integer number of signal cycles, $f_0 = 12.75$, occurring in the FFT aperture), in this case the benefit has not offset the loss mentioned above. But this might not be true in a narrowband interference environment.

In this study, the tasks were divided between the two CPUs to give approximately the same execution times; a redistribution of tasks would be necessary for real-time applications, where noise and signal generation (which take more than 60 percent of the time) are not required. One possible approach is to divide the FFT and the detection in halves, with the two CPUs executing in parallel. The FFT and the three detection algorithms are suitable for such division, since the elementary computations in each stage of the FFT are independent of each other; fur-

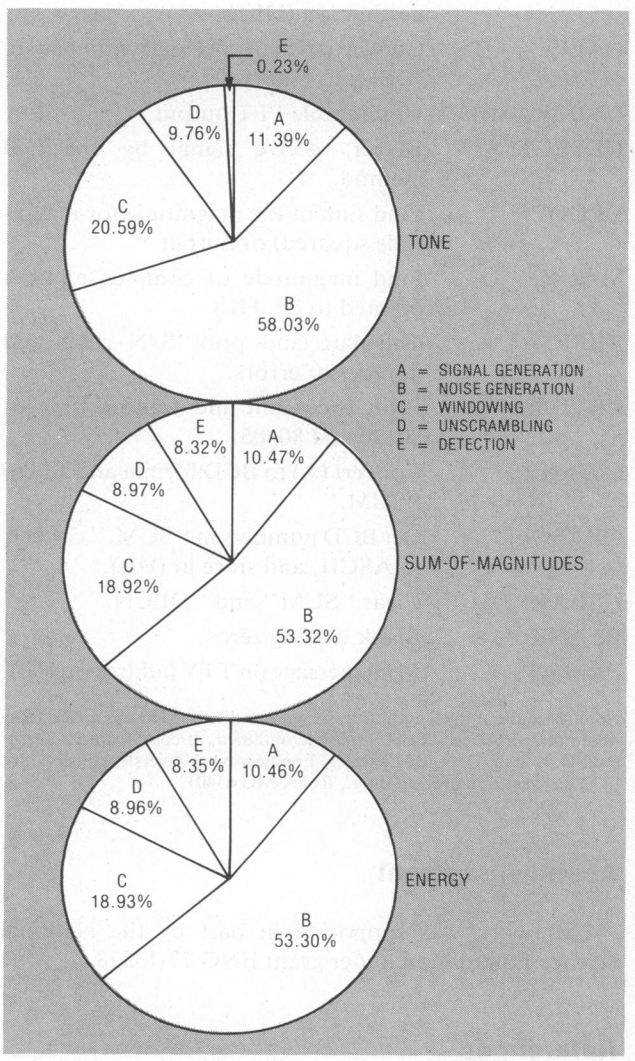


Figure 9. Comparison of the execution times of the principal subroutines in one detection cycle for three detection algorithms.

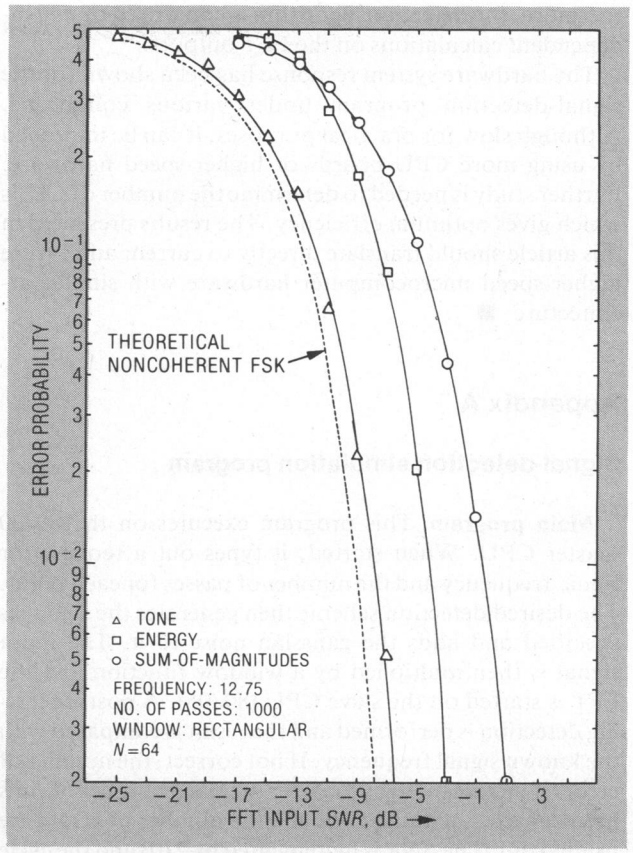


Figure 10. Error probability versus *SNR* at the FFT input for various detection algorithms.

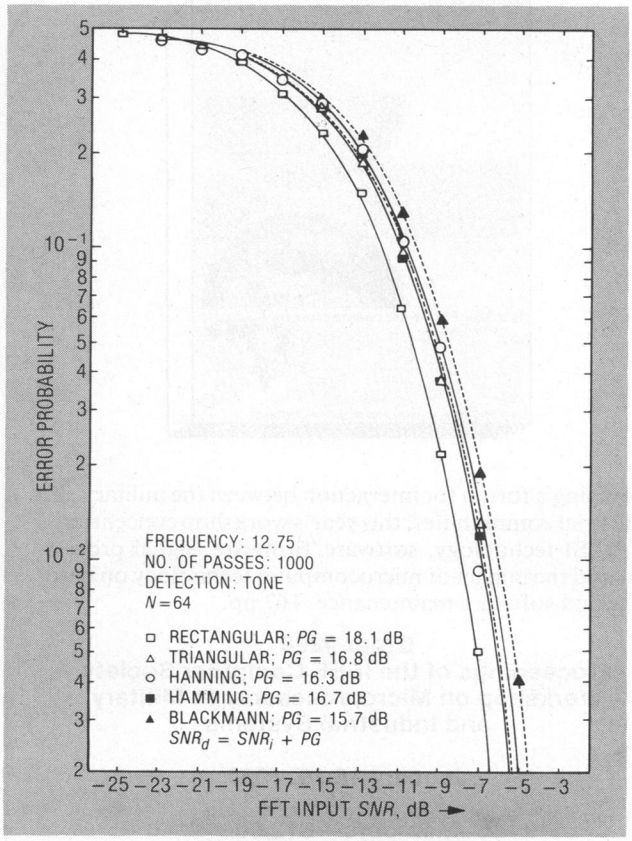


Figure 11. Effect of windows on error probability.

thermore, the detection algorithm is comprised of two independent calculations on the FFT outputs.

The hardware system response has been shown for the signal-detection program under various conditions. Although slow for practical purposes, it can be improved by using more CPU boards or higher-speed hardware. Further study is needed to determine the number of CPUs which gives optimum efficiency. The results presented in this article should translate directly to current and future higher-speed microcomputer hardware with similar architecture. ■

Appendix A

Signal-detection simulation program

Main program. This program executes on the 80/20 master CPU. When started, it types out a request for signal frequency and the number of passes for each point. The desired detection scheme then generates the signal as specified and adds the gaussian noise to it. The input signal is then multiplied by a window function and the FFT is started on the slave CPU (80/05). A postprocessing detection is performed and the result is compared with the known signal frequency. If not correct, the number of errors is incremented by 1. After the fixed number of runs has been executed, the *SNR* and the number of errors are printed out. The *SNR* is incremented by 2 dB and the cycle is repeated 12 times. After that, it prints out the timing and stops. Enter a real number for a signal frequency (format F4.2), up to 4 digits for the number of passes, "sum-of-magnitudes" or "energy" for detection scheme, and a carriage return to proceed.

Appendix B

Subroutine summary

CLKSV:	Increment the clock.
CKMUL:	Check data in (HL). Return if zero; otherwise perform 16 × 16 multiplication and adjust sign.
MUL:	16 × 16-bit multiply.
NEGATE:	Negate the content of (HL).
ABSHL:	Convert (HL) to absolute value and put sign in (C).
TWIDDL:	Find twiddle factor corresponding to the index in (A).
DIV4:	Divide (HL) by 4 or 2.
RETI:	Move content of (DE) and (BC) to destination (HL).
CALI:	Convert 2-digit decimal number in (DE) into binary.
UNSCRAMB:	Unscramble FFT output.
DETECT:	Detect receive signal by specified method.
DLOOP:	Find sum of the magnitude (or magnitude squared) of output.
MAGN:	Find magnitude of complex number pointed to by (HL).
PRPE:	Calculate and print S/N ratio and number of errors.
PRTIME:	Print execution and waiting time of 80/20 and 80/05.
CVBCD:	Convert (A) to BCD format and add to "SUM."
CVASCI:	Get BCD numbers in "SUM," convert to ASCII, and store in (HL).
CLEAR:	Clear "SUM" and "DECN."
BLANK:	Blank leading zeros.
PRMSG:	Print message on TTY (address in HL).

Note: Because of the length of the programs discussed in this article, they were not reproduced. Copies may be obtained, at cost, from Professor Paul D. Stigall, Dept. of Electrical Engineering, University of Missouri, 123 Electrical Engineering Bldg., Rolla, MO 65401.

Acknowledgment

This work was supported in part by the National Science Foundation under grant ENG 77-13878.

References

1. R. E. Ziemer and W. H. Tranter, *Principles of Communications,* Houghton Mifflin Co., Boston, 1975.

2. B. Gold and T. Bially, "Parallelism in Fast Fourier Transform Hardware," *IEEE Trans. Audio Electroacoustics,* Vol. AU-21, Feb. 1973, pp. 5-16.

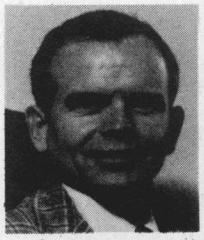
3. F. J. Harris, "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," *Proc. IEEE,* Vol. 66, No. 1, Jan. 1978, pp. 51-83.

4. L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing,* Prentice-Hall Inc., Englewood Cliffs, NJ, 1975, pp. 567-568.

5. P. D. Welch, "A Fixed Point Fast Fourier Transform Analysis," *IEEE Trans. Audio Electroacoustics,* Vol. AU-17, June 1969, pp. 151-157.

6. A. E. Filip, "A Baker's Dozen Magnitude Approximations and Their Detection Statistics," *IEEE Trans. Aerospace & Electronics Systems,* Vol. AES-12, No. 1, Jan. 1976, pp. 86-89.

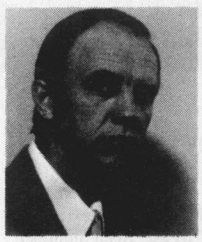
7. V. T. Pham, "Simulation of a Signal Detection System on a Microcomputer," M.S. Thesis, University of Missouri-Rolla, Dec. 1978.



Paul D. Stigall is a professor of electrical engineering and computer science at the University of Missouri-Rolla, which he joined in 1970. His research interests include computer architecture and systems, digital signal processing, microprocessor and minicomputer application, digital circuits, and fault-tolerant computing. He is the author of several technical publications and the principal investigator on several research projects.

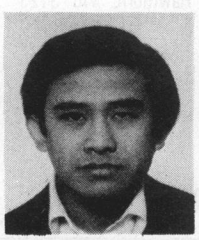
Before joining the university, Stigall worked variously for McDonnell Douglas, the Navy Electronics Laboratory, and the Collins Radio Company. He also was an instructor at the University of Wyoming.

Stigall received the BS degree in electrical engineering from the University of Missouri-Rolla in 1962 and the MS and PhD in electrical engineering from the University of Wyoming in 1965 and 1968. A registered professional engineer in Missouri, he is a member of ACM, Eta Kappa Nu, Tau Beta Pi, and Sigma Xi, and a senior member of the IEEE.



Rodger E. Ziemer joined the University of Missouri-Rolla in 1968; he is a professor of electrical engineering. His recent research and publications have been concerned with communications through multipath, fading, and nongaussian-interference channels. As a consultant, Ziemer has worked with companies and government agencies on the use of digital signal processing in communications and radar systems. He has written, with W. H. Tranter, a textbook entitled *Principles of Communications: Systems, Modulation, and Noise,* and he is an associate editor of *IEEE Transactions on Communications.*

Ziemer received the BS in 1960, the MSEE in 1962, and the PhD in 1965, all from the University of Minnesota. He is a member of Tau Beta Pi, Eta Kappa Nu, Sigma Xi, and the American Society for Engineering Education, and a senior member of the IEEE.



Van Thanh Pham received the BS degree in electronic engineering from the National Institute of Technology, Saigon, Vietnam, in 1973 and the MS in electrical engineering from the University of Missouri-Rolla in 1978. Since 1978 he has been employed with Sperry Univac in Roseville, Minnesota.