01 Jan 1982

A Performance Study of 16-Bit Microcomputer-Implemented EFT Algorithms

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

Rodger E. Ziemer
Missouri University of Science and Technology

L. Hudec

Recommended Citation

P. D. Stigall et al., "A Performance Study of 16-Bit Microcomputer-Implemented EFT Algorithms," *IEEE Micro*, vol. 2, no. 4, pp. 61 - 66, Institute of Electrical and Electronics Engineers; Computer Society, Jan 1982.
The definitive version is available at https://doi.org/10.1109/MM.1982.291039

An FFT algorithm operating on a 16-bit microcomputer can calculate a 256-point transform as much as 10 times faster than a similar algorithm on an eight-bit microcomputer.

A Performance Study of 16-bit Microcomputer-implemented FFT Algorithms

P. D. Stigall, R. E. Ziemer, and L. Hudec

University of Missouri-Rolla

Microcomputers have been used in many applications to yield a more cost-effective solution and increase flexibility. Such an application is digital signal processing, in which computers are suitable for many tasks including filtering and control. One form of digital filtering is performed with the aid of a fast Fourier transform, or FFT, algorithm. This study investigates the performance—in terms of execution time—of Gold-Bially FFT algorithms[1] implemented on a 16-bit microcomputer, the SBC 86/12A manufactured by Intel Corporation. This particular FFT algorithm is convenient for microcomputer implementation because it is organized on an array basis. The algorithms perform the FFT calculations for radix 2 or 4 and a variable number of input data samples up to 1024. FFT computation times for the 16-bit SBC 86/12A are compared with those of two previously implemented eight-bit systems.[2,3]

FFT algorithms with arbitrary radixes

Quite often, the FFT algorithms employed in signal processing applications involve elementary computations—referred to as butterflies—which produce two output words from two input words. Such an algorithm is termed a radix-2 algorithm and its use implies that the transformed sequence is composed of 2^n members. Suppose an N-point discrete Fourier transform, or DFT, is desired, in which N is a composite number which can be factored into the product of integers

$$N = N_1 N_2 \ldots N_m,$$

where N is not necessarily a power of two. For example, if $N = 64$ and $m = 3$, we might factor N into the product $64 = 4 \times 4 \times 4$ so that the 64-point transform can be viewed

0272-1732/82/1100-0061$00.75

as a three-dimensional $4\times4\times4$ transform. Such an approach is sometimes taken for several reasons. First, each elementary computation (i.e., each four-point transform in the above example) may involve fewer multiplications, additions, or memory cycles than the equivalent number of radix-2 operations. Second, this approach allows one to parallel operations using additional hardware and thereby completely overlap memory cycles and computation cycles, thus making more efficient use of processor hardware. Third, the procedure can be formulated as a series of matrix computations, making it easier to implement the FFT as a concurrently programmed algorithm.

To further explain this approach, suppose that N can be represented as the product of two integers

$$N=N_1N_2.$$

(If N_1 and N_2 are both composite, they can be further broken down into the products of other integers.) The data to be transformed can then be arranged into a two-dimensional array, and DFTs can be done on each row individually. Each element of the resulting array of numbers is then multiplied by a "twiddle factor," which is an appropriate power of $W_N=\exp[-j2\pi/N]$. The resulting matrix is then transformed column by column to obtain the final result.

If N is a prime number, making factorization of N impossible, the original signal can be zero-padded in such a way that the resulting new composite number of points can be factored. If N can be expressed as $N=r^m$, and if the algorithm is carried out by means of a succession of r-point transforms, the resulting FFT is called a radix-r FFT. In a radix-r FFT, an "elementary computation," denoted EC, consists of an r-point DFT followed by multiplication of the r output quantities by the appropriate twiddle factor. The number of ECs required is

$$C_r=\frac{N}{r}\log_r N,$$

which clearly decreases as r increases. Of course, the complexity of an EC increases with increasing r. For $r=2$, the EC (the butterfly mentioned previously) consists of a single complex multiplication and two complex additions, while for $r=4$, the EC requires three complex multiplications and several complex additions.

To see how the radix-r FFT algorithm is developed in this fashion, we begin with the more general case where the composite number N factors as $N=LM$. Let n and k in the DFT sum be represented as

$$k=M\ell+m, n=Lr+s$$
$$(s,\ell=0, 1, \ldots, L-1;$$
$$r, m=0, 1, \ldots, M-1).$$

Using this index notation, the DFT sum can be written as

$$X(Lr+s)=\sum_{m=0}^{M-1}\sum_{\ell=0}^{L-1}x(M\ell+m)\,W_N^{(M\ell+m)(Lr+s)}$$
$$r=0, 1, \ldots, M-1, \quad (1)$$
$$s=0, 1, \ldots, L-1.$$

Now,

$$(M\ell+m)\,(Lr+s)=MLr\ell+M\ell s+Lrm+ms,$$

and

$$W_N^{MLr\ell}=W_N^{N\ell r}=1.$$

Hence, Equation 1 can be rearranged into the form

$$X(Lr+s)+\sum_{m=0}^{M-1}W_N^{Lmr}W_N^{ms}\sum_{\ell=0}^{L-1}x(M\ell+m)\,W_N^{Ms\ell}. \quad (2)$$

Defining

$$q(s,m)=\sum_{\ell=0}^{L-1}x(M\ell+m)\,(W_N^M)^{\ell s}, \quad (3)$$

the DFT sum of Equation 2 can be written as

$$X(Lr+s)=\sum_{M=0}^{M-1}q(m,s)\,W_N^{ms}(W_N^L)^{mr}, \quad (4)$$

where it is important to note two points: First, for any fixed m, $q(s,m)$ can be interpreted as a DFT of each row of the array of $x(n)$s arranged as

$$[x(n)]=\begin{bmatrix} x(0) & x(M) & \cdots & x(ML-M) \\ x(1) & x(M+1) & & x(ML-M+1) \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ x(M-1) & x(2M-1) & & x(ML-1) \end{bmatrix}. \quad (5)$$

Table 1.
Results of transformation of array $[q(s,m)W_{16}^{ms}] = [q(s,m)]$.

r	s	$X(Lr+s)$;		$L=4$	(output unscrambled)
0	0	$X(0)$	=	0	Eq.(a); Col. 0
0	1	$X(1)$	=	0	Col. 1
0	2	$X(2)$	=	0	Col. 2
0	3	$X(3)$	=	0	Col. 3
1	0	$X(4)$	=	8	Eq.(b); Col. 0
1	1	$X(5)$	=	0	Col. 1
1	2	$X(6)$	=	0	Col. 2
1	3	$X(7)$	=	0	Col. 3
2	0	$X(8)$	=	0	Eq.(c); Col. 0
2	1	$X(9)$	=	0	Col. 1
2	2	$X(10)$	=	0	Col. 2
2	3	$X(11)$	=	0	Col. 3
3	0	$X(12)$	=	8	Eq.(d); Col. 0
3	1	$X(13)$	=	0	Col. 1
3	2	$X(14)$	=	0	Col. 2
3	3	$X(15)$	=	0	Col. 3

Since each is an L-point transform, the proper complex exponential multiplier to be used is

$$W_L = \exp[-j2\pi/L] = \exp[-j2\pi M/(ML)] = W_N{}^M, \quad (6)$$

which checks with Equation 3. Second, each element of the array $[q(s,m)]$ is multiplied by the twiddle factor $W_N{}^{sm} = W_N{}^{ms}$ (i.e., W_N is raised to the power of the product of the row and column indices), and an M-point DFT is performed on each *column*. (Note the reversal in the indices for $[q(m,s)]$ in Equation 4 as compared with Equation 3.) The proper W to be used is

$$W_M = \exp(-j2\pi/M) = \exp[(-j2\pi L/(ML)] = W_N{}^L, \quad (7)$$

which checks with Equation 4. In this fashion, the N-point DFT is broken down into a series of L-point DFTs followed by a series of M-point DFTs, where $N = ML$.

To better understand how the DFT is done by taking this approach, consider the example below.

A Gold-Bially DFT computation. Consider $N = 16$, $L = M = 4$, and $x(k) = \cos(\pi k/2)$. Arranging the data into a 4×4 array, we have

$$[x(k)] = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

where $x(0)$, $x(1)$, $x(2)$, and $x(3)$ occupy the first column, and so on. Applying Equation 3 to each row, we obtain

$$q(s,m) = \sum_{\ell=0}^{3} x(4\ell+m)[e^{-j\pi/2}]^{s\ell}.$$

Thus, the four-point EC is

$$q(0,m) = x(m) + x(4+m) + x(8+m) + x(12+m) \quad (a)$$
$$q(1,m) = x(m) - jx(4+m) - x(8+m) + jx(12+m) \quad (b)$$
$$q(2,m) = x(m) - x(4+m) + x(8+m) - x(12+m) \quad (c)$$
$$q(3,m) = x(m) + jx(4+m) - x(8+m) - jx(12+m). \quad (d)$$

In keeping with array notation, the second argument of $q(s,m)$ indicates the row. Applying this EC to the array $[x(k)]$, we obtain the array

$$[q(s,m)] = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The next step, according to Equation 4, is to multiply by the twiddle factor $W_{16}{}^{ms}$. Since both nonzero factors in the $[q(s,m)]$ array are in the $s = 0$ row, there is no modification by the twiddle factor. The final step is to transform the array $[q(s,m)\,W_{16}{}^{ms}] = [q(s,m)]$ column by column in accordance with Equation 4. (Note the index reversal in Equation 4 on $q(m,s)$.) The EC is identical to that used to obtain $[q(s,m)]$. The results are presented in Table 1.

System structure

The implementations of the Gold-Bially FFT algorithms for radix 2 and radix 4 were done on the 16-bit Intel SBC 86/12A, interfaced to a microcomputer development system, the MDS-225, also manufactured by Intel. The SBC 86/12A is a single-board computer which includes an Intel 8086 CPU, 32K bytes of dual-port RAM, and sockets for 16K bytes of PROM. The SBC is used for calculating the FFT algorithms, while the development system is used for software development, for input data generation, and as an intelligent communication terminal.

The FFT algorithms are written in Intel ASM 86 assembly language to allow two parameters to be chosen—radix and FFT size. The flowchart for this program is shown in Figure 1. (A detailed program listing is available from the author—see address following Professor Stigall's biography, page 66.) A rectangular window is assumed, thus avoiding additional multiplica-

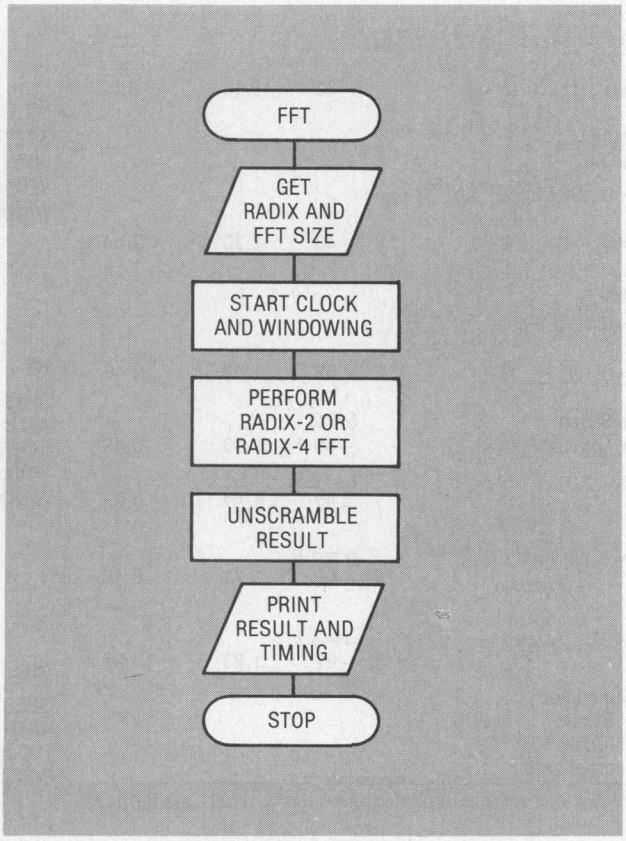


Figure 1. Flowchart for the FFT simulation program.

Windows and their properties

Sampling the spectrum of a finite-duration time-domain signal implies a periodic extension of that signal in the time domain. Unless the signal is periodic with an integer number of periods within the sampling interval, or window, or unless it smoothly approaches zero at each end of the interval, the resulting discontinuities generate additional spectral components. This phenomenon is referred to as spectral leakage. The reason for spectral leakage can be traced directly to the discontinuities implied at either end of the sampled signal. To minimize spectral leakage, data samples can be multiplied by nonrectangular windows which approach zero smoothly at the beginning and end of the sampling interval. Several window functions are available. A few common ones and their properties are listed in the table below. Their effects on a cosine signal with a noninteger number of cycles within the sampling interval are shown in the accompanying figure. In general, these results show that it is not possible to simultaneously have a narrow mainlobe and low sidelobes for a window. However, some windows provide a better compromise between these two parameters than others.

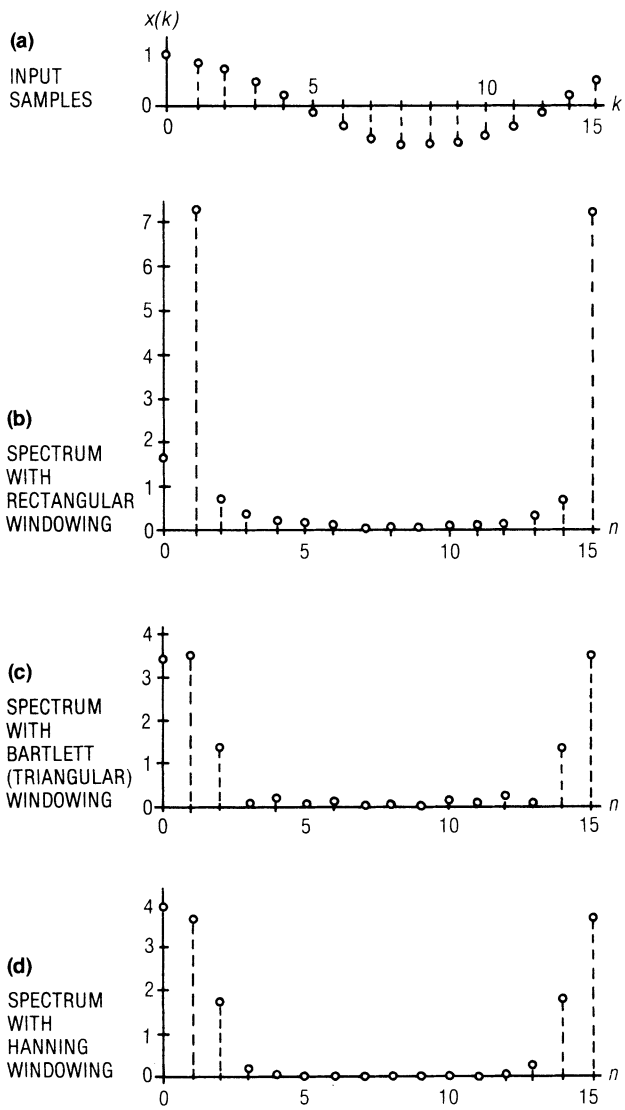
1. F. J. Harris, "On the Use of Windows for Harmonic Analysis with Discrete Fourier Transform," *Proc. IEEE,* Vol. 66, No. 1, Jan. 1978, pp. 51-83.
2. H. H. Nuttull, "Some Windows with Very Good Sidelobe Behavior," *IEEE Trans. Acoustics, Speech and Signal Processing,* Vol. ASSP-29, No. 1, Feb. 1980, pp. 84-91.

Various windows and their characteristics.*

Window	Highest sidelobe level; dB	3-dB width frequency; bins	Coherent gain/N: $[\Sigma w(n)]^2/\Sigma w^2(n)$
1. Rectangular $w(n)=1, n=0,\ldots,N-1$	−13	0.89	1.0
2. Triangular $w(n)=2(n+0.5)/N$ $w(N-n-1)=w(n)$ $n=0,1,2,\ldots,N/2$	−27	1.28	0.5
3. Hanning $w(n)=\frac{1}{2}\left[1-\cos\left(\frac{2(n+0.5)\pi}{N}\right)\right]$ $n=0,1,2,\ldots,N-1$	−32	1.44	0.5
4. Hamming $w(n)=.54-0.46\cos\left(\frac{2\pi(n+0.5)}{N}\right)$ $n=0,1,2,\ldots,N-1$	−43	1.30	0.54
5. Blackman $w(n)=0.42-0.5\cos\left(\frac{2\pi(n+0.5)}{N-1}\right)$ $n=0,1,2,\ldots,N-1$	−58	1.52	0.46
6. Kaiser-Bessel $w(n)=I_0(\pi\alpha\beta)/I_0(\pi\alpha)$ where	$\alpha=2$; −46	1.43	0.49
	$\alpha=2.5$; −57	1.57	0.44
$\beta=\sqrt{1-\left(\frac{2n+1}{N}-1\right)^2}$	$\alpha=3.0$; −69	1.71	0.40
$n=0,1,2,\ldots,N-1$	$\alpha=3.5$; −82	1.83	0.37
$I_0(x)$ = modified Bessel function; order zero.			

*See Harris.[1] For a correction to, and extension of, Harris' results, see Nuttull.[2]

The effects of various windows on the spectrum of a sampled cosine waveform.



tions on the input to the FFT. (For an overview of nonrectangular windows, see box at left.) Some of the program features are described below.

Data representation. Each complex data sample is represented as two 16-bit two's-complement binary numbers. Since a popular and practical size for A/D and D/A converters is 12 bits, the integer range of the data samples is limited to -2048 to $+2047$. The most significant bit is the sign bit, while the eleven least significant bits are used for data.

Data structure. The complex data samples are stored as double words in memory in the following order:

- Radix 4—data are stored as a two-dimensional array with $N/4$ rows and 4 columns.
- Radix 2—data are stored as a two-dimensional array with $N/2$ rows and 2 columns.

Twiddle factors. The exponential values of the complex twiddle factor required for the Gold-Bially FFT algorithms are derived from a table containing the cosines of angles in the first quadrant with an increment of $\pi/512$. Each value in the table has a 16-bit accuracy, or a maximum error of $\pm 7.6 \times 10^{-6}$.

Overflow check. An overflow occurs when the sum of two numbers is equal to or greater than 8, since there are only three bits to represent the integer part. An overflow check is performed before each addition by temporarily adding the integer parts. If the result is greater than 6, all data are divided by the radix, and the number of scalings is incremented by 1. This scaling factor is printed out when the FFT is completed.

Input/output. The two parameters, radix and FFT size, are entered by the user. The output of the program includes the FFT input data, the results, the execution time, and the scaling factor.

System performance

Many applications in digital signal processing require FFT algorithms to operate in a fixed time period. This is especially true in real-time processing, where new data is sampled at regular intervals. Hence, the execution time of FFT algorithms on a modern 16-bit microcomputer is an important performance measurement.

16-bit microcomputer performance. Table 2 gives the execution times and the number of scaling operations for two different data inputs and a given number of samples. The execution times are in seconds. The scaling operations result from checking and preventing overflow, as discussed above. The two different inputs are (1) a set of data samples of a sine function with one period and an amplitude of 2047 units, and (2) a set of data samples of random numbers from -2048 to $+2047$.[3] When random data are used for an input, the results are averaged for ten different random sets of data.

Comparison with eight-bit microcomputers. Table 3 compares the execution times of three different microcomputer implementations of the previously discussed Gold-Bially FFT algorithms. The Futuredata (Microkit 8/16) is a microcomputer development system with 16K bytes of RAM, a keyboard, a CRT display, and a teleprinter. Its shortest execution time for an instruction is two microseconds. It has no hardware multiplication. The System 80/20, manufactured by Intel Corporation, includes a single-board computer (an SBC 80/20), 2K

Table 2.
Timing and scaling operations for different-sized FFTs on a 16-bit microcomputer (SBC 86/12A).

NUMBER OF SAMPLES	EXECUTION TIME (SECONDS)		NUMBER OF SCALING OPERATIONS	
	RADIX 4	RADIX 2	RADIX 4	RADIX 2
1024	1.98* (1.90)**	3.43 (3.33)	5 (3.60)	9 (6.60)
512	—	1.53 (1.49)	—	8 (6.00)
256	0.390 (0.370)	0.679 (0.659)	4 (3.00)	7 (5.40)
128	—	0.296 (0.291)	—	6 (5.00)
64	0.071 (0.068)	0.126 (0.120)	3 (2.10)	5 (4.00)
32	—	0.0520 (0.0509)	—	4 (3.60)
16	0.0109 (0.0106)	0.0204 (0.0201)	2 (1.90)	3 (3.00)

*Sine data used for an input.
**Random data used for an input (average of 10 input sets).

Table 3.
Timings for different-sized FFTs implemented on different microcomputer systems.

NUMBER OF SAMPLES	EXECUTION TIME (SECONDS)					
	FUTUREDATA		SYSTEM 80/20		SBC 86/12A	
	RADIX 4	RADIX 2	RADIX 4	RADIX 2	RADIX 4	RADIX 2
1024	19.41*	22.95	—	—	1.90	3.33
256	3.788	4.389	1.293	1.447	0.370	0.659
64	0.687	0.772	0.195	0.229	0.068	0.120
16	0.118	0.117	0.046	0.043	0.011	0.020

*Random data used for an input (average of 10 input sets).

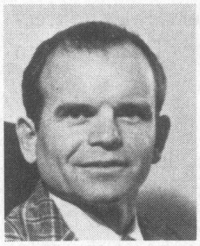
bytes of RAM, 2K bytes of EPROM (on which the FFT program resides), and a teleprinter. It also has a math board capable of fixed- and floating-point operations. Both the Microkit 8/16 and the System 80/20 use an eight-bit microprocessor, the Intel 8080. The third microcomputer in the comparison is the 16-bit SBC 86/12A described previously.

Sixteen-bit microcomputers have yielded economical solutions to previously uneconomical applications. As demonstrated in this article, an FFT algorithm operating on a 16-bit microcomputer can calculate a 256-point transform in less than 400 milliseconds. This is approximately 3.5 (using an external-multiply board) and 10 (using a software-multiply routine) times faster than a similar algorithm implemented on eight-bit microcomputers having the same accuracy. The 16-bit implementation's speed would be very desirable for applications involving human interaction.

The execution time of the Gold-Bially FFT algorithm depends on the radix and on the type of input data. In some applications other window functions, rather than the rectangular window function implemented here, may be desirable. ■

References

1. B. Gold and T. Bailly, "Parallelism in Fast Fourier Transform Hardware," *IEEE Trans. Audio Electroacoustics,* Vol. AU-21, Feb. 1973, pp. 5-16.
2. P. D. Stigall, R. E. Ziemer, and V. T. Pham, "Performance Studies of Microcomputer-Implemented Fast Fourier Transforms," *Proc. 1979 Int'l Micro and Mini Computer Conf.*, Nov. 1979, pp. 187-190.
3. P. D. Stigall, R. E. Ziemer, and V. T. Pham, "A Performance Study of a Microcomputer-Implemented FSK Receiver," *IEEE Micro*, Vol. 1, No. 1, Feb. 1981, pp. 43-51.

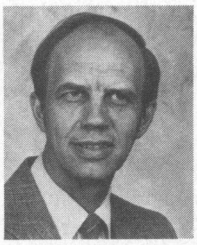


Paul D. Stigall is a professor of electrical engineering and computer science at the University of Missouri-Rolla, which he joined in 1970. His research interests include computer architecture and systems, digital signal processing, microprocessor and minicomputer application, digital circuits, and fault-tolerant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

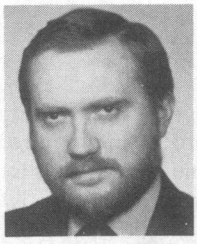
Stigall received the BS degree in electrical engineering from the University of Missouri-Rolla in 1962 and the MS and PhD in electrical engineering from the University of Wyoming in 1965 and 1968. A registered professional engineer in Missouri, he is a member of ACM, Eta Kappa Nu, Tau Beta Pi, and Sigma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

Stigall's address is the Department of Electrical Engineering, University of Missouri, 123 Electrical Engineering Building, Rolla, MO 65401.



Rodger E. Ziemer has been on the faculty of the University of Missouri-Rolla since 1968, where he is currently a professor of electrical engineering. In addition to teaching courses and performing research in communications and signal processing, he has been a consultant with several industries and government agencies on problems involving signal processing in communications and radar systems. During the academic year 1980-81 he was on leave while doing research and development on high-data-rate communications systems at the Communications Research Facility of the Motorola Government Electronics Division, Scottsdale, Arizona.

A member of Tau Beta Pi, Eta Kappa Nu, Sigma Xi, and the American Society for Engineering Education, Ziemer is a registered professional engineer. He received BS, MSEE, and PhD degrees from the University of Minnesota in 1960, 1962, and 1965, respectively.



Ladislav Hudec is an assistant professor of electrical engineering at the Slovak Institute of Technology, Bratislava, Czechoslovakia. His research interests include digital design, microprocessor applications, computer architecture, and fault-tolerant computing. In 1981, as a fellow of the United Nations Industrial Development Organization, he worked in the Department of Electrical Engineering of the University of Missouri-Rolla.

Hudec received the MS in electronics from the Czech Institute of Technology, Prague, in 1974. He is a member of the Slovak Society for Science and Technology.

The views expressed in this article are those of the authors. They are not necessarily those of the United Nations Industrial Development Organization, or of the countries participating in the UNIDO Fellow Program.