Designing Programmable Digital Filters for LSI Implementation

by Lynn A. Schmidt

THE POWER OF DIGITAL SIGNAL PROCESSING now becoming available through the use of large-scale integrated circuits is giving low-frequency spectrum analyzers unprecedented flexibility and computational power at relatively low cost. For example, the HP Model 3582A, described in the preceding article, derives much of its signal-processing capabilities from four single-chip LSI digital filters. This article describes how it was possible to implement these filters on LSI chips.

The arithmetic, storage, and speed requirements of the digital filtering proposed for the Model 3582A presented formidable technical challenges. The requirement for doing three to six million multiplications per second far exceeded the capabilities of available microprocessors. Additional complications were presented by the need for 18- to 21-bit precision in the computations. Thus, a dedicated hardware signal processor of some kind appeared to be the only viable solution.

This processor could have been built with off-the-shelf components, but the cost in terms of component count, board space, power, and reliability would have been high. Implementing the processor by large-scale integration (LSI), however, promised more than a 80% hardware savings, coupled with lower power and higher reliability from the use of fewer components. The development of a custom NMOS digital filter chip was therefore undertaken.

The underlying problem was how to design the filter processor with enough power to do the job within the confines of a reasonably sized chip.

Digital Filtering in Spectrum Analysis

The term "digital filter" refers to a computational process by which a sequence of numbers, usually samples of an analog signal, is transformed into a second sequence of numbers (see box, page 17). In a spectrum analyzer this process corresponds to either low-pass or bandpass filtering.

Digital filtering in a spectrum analyzer also provides a practical means of implementing band-selectable analysis or "zoom" to analyze the spectrum of a narrow frequency band centered on any frequency within the range of the analyzer. The translation of the frequency band to baseband needed for band-selectable analysis is possible by analog techniques but is often impractical because of the high cost of ensuring good gain and phase match between the real and imaginary signal channels. Two or more digital filters, on the other hand, can be matched in gain and phase exactly, regardless of temperature, aging, or production variations.

Span Control

Model 3582A uses a particular type of digital filter called a decimation filter, a type that plays a fundamental role in analyzers based on the fast Fourier transform (FFT).

As explained in the box on page 7, the first step in FFT processing is the collection of N equally-spaced-in-time samples of the analog input during a measurement time interval T. The N samples are used in computing the discrete Fourier transform of the input signal. The result of the transform can be likened to the response of a bank of N narrow bandpass filters spaced at center frequencies of 1/T Hz. The responses of these filters are plotted on the analyzer's CRT to generate the amplitude-vs-frequency display.

An important characteristic of the FFT frequency analysis is that the frequency span width and resolution is directly proportional to the sampling rate ($f_s = N/T$). In accordance with sampled-data theory, the input signal must be band limited to less than $f_s/2$ to avoid aliasing but because of practical filter limitations, the input signal bandwidth is usually limited to something between $f_s/4$ and $f_s/3$.

Spectrum analyzers need to change their frequency span over a broad range to accommodate a variety of input signals. Model 3582A, for example, has fourteen spans ranging from 1 Hz to 25 kHz in a 1-2.5-5 sequence. Providing fourteen sample rates poses no problem, but providing an analog anti-aliasing filter with fourteen programmable bandwidths does. In the Model 3582A, digital filters following the A-to-D converter limit the bandwidth to the frequency span desired. The input signal to the A-to-D converter is band-limited to 25.6 kHz by a fixed, analog, low-pass filter regardless of the frequency span desired, and samples of the input are taken at a fixed rate of 102.4 kHz. Sample words at the output of the digital filters are selectively discarded to effect a resampling at a

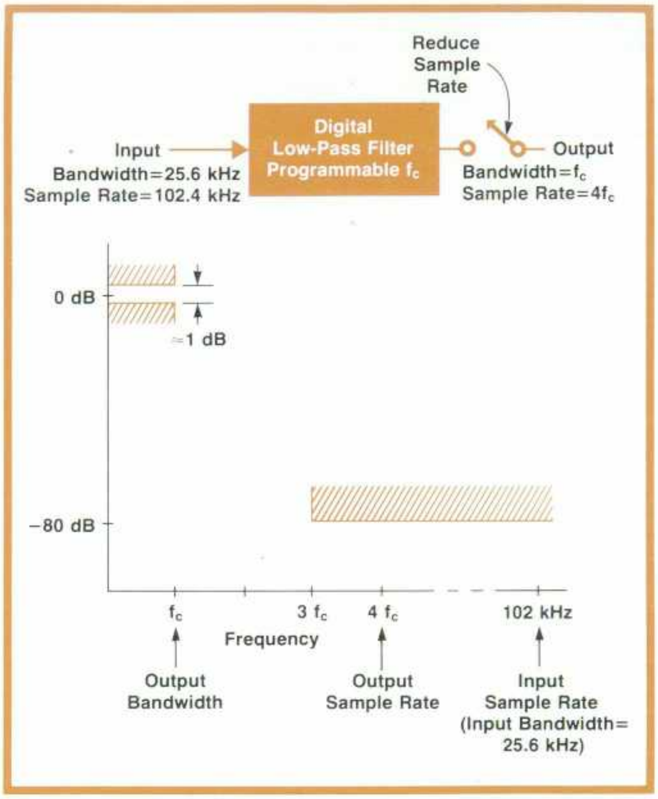


Fig. 1. *Specifications for the band-limiting filters to be used in the Model 3582A Spectrum Analyzer. The filter is to have 14 programmable cut-off frequencies (f_c) ranging from 25 kHz down to 1 Hz.*

lower rate of four times the desired frequency span.

This process of simultaneous digital low-pass filtering and sample rate reduction is called decimation filtering. Only the band to be analyzed is presented to the FFT processor. Since the resample rate tracks the span width, the FFT algorithm remains the same regardless of the selected span.

Choosing a Filter

The basic requirements for the digital filters in the Model 3582A are shown in Fig. 1. The passband corner frequency, since it sets the frequency span of the analysis, needs to be variable from 25 kHz (no filtering) down to 1 Hz in a 25-10-5-2.5 sequence. To provide proper anti-aliasing, the stopband needs to start at three times the cut-off frequency and should be at least −80 dB relative to the passband to assure that aliased signals are suppressed to an undetectable level. The passband ripple, however, can be on the order of ±0.5 dB since the effects of ripple can be compensated for by weighting the spectral frequency components after the FFT analysis. This compensation is accomplished simply by multiplying the spectrum by the inverse of the ripple function. Phase nonlinearities introduced by the filter can be removed in an identical fashion.

It soon became obvious that it was impractical to meet these requirements with a single filter. The higher reduction ratios (input bandwidth/output bandwidth) that would be needed for the narrow span widths would require coefficient word lengths greater than 30 bits if a recursive infinite-impulse-response (IIR) low-pass filter were used. If a nonrecursive finite-impulse-response (FIR) filter were used, the number of coefficients required would be greater than 100,000.

Coefficient words get very long with IIR filters because the poles of the transfer function in the z plane group closer and closer to the unit circle at $z = +1$ as the bandwidth is reduced. The relative positions of these poles must be maintained precisely to preserve both stability and the desired transfer function. With finite arithmetic, the poles can only assume certain quantized positions so, to obtain the desired transfer function as the bandwidth is made narrower, the precision of the arithmetic needs to be increased.

With FIR low-pass filters, the transfer function is determined by zeros in the z plane. The order of the filter, or equivalently the length of its impulse response, grows in size inversely with bandwidth. The order (number of multiplication coefficients) required for the 1-Hz bandwidth is about 10,000 times that required for the 10-kHz bandwidth.

LSI chip area relates directly to the complexity of the processing algorithms for the LSI filters, so a reduction in processing complexity was a major goal for this project. The solution was to implement the filter as a cascade of several filters with interstage decimation of the sample rate, a common practice with decimation filtering. Each filter stage then needs to accomplish only a modest sample rate reduction with a consequent reduction in the complexity of the filter

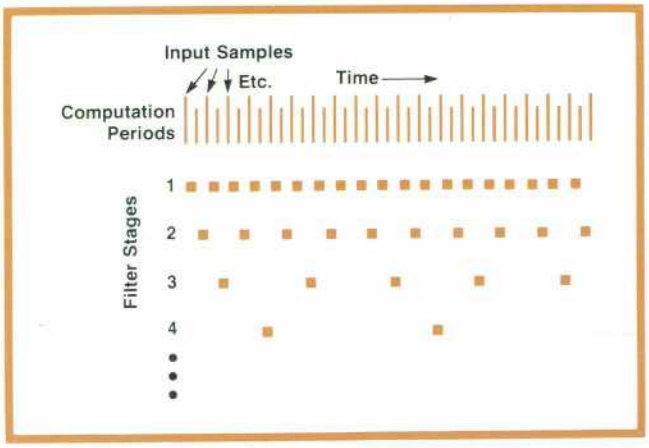


Fig. 2. *Interleaved processing scheme for the digital filter. Two complete filter computations are performed during each input sample period: one for the first filter stage, and one for one of the downstream filters according to this example schedule.*

stage. In addition, the downstream filters in the cascade compute at reduced sample rates, further simplifying computational requirements.

Interleaved Multiplications

In the final filter design, a cascade of eight filters in series is used with the output selected from any of the eight. Each filter stage is allowed to compute either a decimation by two or a decimation by five, giving overall decimation ratios of $2^m \times 5^n$ where $m + n \leq 8$. This easily satisfies the requirement for the 14 frequency spans with decimation ratios ranging from 1 to 25000.

An interesting aspect of this filter structure is found upon examination of the total computational requirements. Consider the case where all eight filters are each performing a decimation by 2. The first filter computes at the input sample rate. The second filter, however, needs to compute at only one-half the rate because of the sample rate reduction. In like manner, the third filter computes at one-fourth the rate, the fourth at one-eighth the rate, and so on, forming a geometric series, $1+\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+$.....,that converges on 2. This means that the total computational requirement of all eight filter stages can be satisfied by a single processor operating at twice the rate required by the first filter stage. Actually, hardware is needed for only one filter stage. The processor has eight channels of memory to store intermediate states of the filters but only one set of arithmetic hardware. Fig. 2 illustrates how the processing steps are scheduled so the eight filter stages can timeshare the hardware.

IIR or FIR?

Much has been written in the literature about the virtues of both IIR and FIR filters. The primary differences between these two classes of filters as far as decimation is concerned are listed below in Table 1. Evidently the advantages of FIR filters have resulted in more decimation filters being implemented with FIR approaches than IIR, but in our case the necessity to implement the filters in LSI required some special considerations.

Table I

FIR	IIR
Only need to compute at output rate	Must compute every input sample
Linear phase	Not linear phase
More storage	Less storage
4-8 multiplications/input	8-12 multiplications/input
N=16 to 38/per stage	5th order structure/per stage
More coefficients	Fewer coefficients
Less noise/bit	More noise/bit

In making the choice between FIR and IIR, let us consider the elements that require chip area in an LSI digital filter. By far the largest consumer of chip area would be the multipliers. For example, a 12×16-bit multiplier requires nearly 10^6 square micrometres of chip area. A second consumer is memory. Dynamic memory takes less space than static, and serial shift registers take much less space than random access memory.

A third large consumer of chip area is interconnect. We have found that about 40% of the area of most LSI chips is devoted to simply interconnecting the various active devices. Serial arithmetic structures bring an advantage here since only a single interconnect is needed to transfer a 24-bit word. This, however, is at the expense of the total transfer time.

We chose to use serial arithmetic to minimize interconnects, and dynamic shift-register memory. However, the key element by far in our design strategy became the multipliers and associated coefficients.

What is a Digital Filter?

Digital filtering is a computational process or algorithm by which a sampled signal or sequence of numbers, acting as an input, is transformed into a second sequence of numbers called the output. The computational process may correspond to high-pass, low-pass, bandpass, or bandstop filtering, integration, differentiation, or something else. The second sequence can be used for further processing, as in a fast-Fourier-transform analyzer, or it can be converted to an analog signal, producing a filtered version of the original analog signal.

The digital approach offers many advantages over analog approaches:

- Changes resulting from variations in component values—normally associated with filter capacitors and resistors as a result of temperature or aging—are non-existent.
- Periodic calibration is eliminated.
- The performance from unit to unit is stable and repeatable.
- Great flexibility is available since filter response can be altered by changing arithmetic coefficients.
- Arbitrarily high precision can be achieved, limited only by the number of bits involved.
- Very small size, low power, and low cost are possible by large-scale integration.

An Example

Let us consider the following practical situation: the need to reject very strong 60-Hz interference contaminating a 10-Hz signal of interest. The obvious solution is to build a bandstop or notch filter that rejects the 60-Hz component while passing the 10-Hz signal with little or no alteration.

First a familiar analog RC notch filter will be examined. A commonly used RC notch filter network is the "twin-T" shown in Fig. 1 (next page). This familiar network derives its bandstop characteristics from a pair of transmission zeros located in the s plane on the imaginary axis at 60 Hz. It also has two poles on the negative real axis, but they hardly affect the bandstop characteristics. Applying the waveform shown in color in Fig. 1 to the notch filter results in the output shown in black. As can be seen, the twin-T circuit rejects all of the 60-Hz interference and leaves the desired 10-Hz signal.

Changing the center frequency of the notch requires changes in the values of all three capacitors, all three resistors, or the two shunt elements, R′ and C′, together. The position and depth of the notch is very sensitive to parameter changes in the components of the twin-T, so precision components must be used in building the filter if good rejection at exactly 60 Hz is to be maintained.

The Digital Filter

The digital filter solution to this problem is simple yet just as effective as the linear filter and is considerably more flexible. In this case the input signal is sampled by an analog-to-digital converter at a rate of 500 samples per second. That is, the input signal level is measured at intervals of 2 ms and the result forms the input sequence to our digital filter.

Fig. 2 shows the digital counterpart to the twin-T notch circuit. This filter is made using two registers, a digital multiplier, and a 3-input adder. As the difference equation in Fig. 2 shows, the current output sample word [y(n)] is formed by summing the current input sample word [x(n)] with the previous input [x(n−1)] multiplied by the constant −1.46, and with the second previous input [x(n−2)]. When this filter is fed sample words at a rate of 500 samples/s, 60-Hz components in the analog input signal will be eliminated from the regenerated analog output signal.

Just as the Laplace transform can be used to analyze linear filters, the z transform can be used to analyze digital filters. The filter shown in Fig. 2 derives its bandstop characteristics from a pair of transmission zeros located in the z plane on the unit circle at a point corresponding to 60 Hz. No poles are indicated in this transfer function, but these can be implemented when desired.

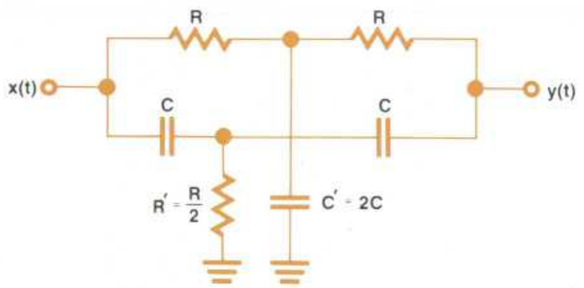
Fig. 2 also shows two cycles of the sampled 10-Hz input signal (in color) which is contaminated as before by 60-Hz interference. Passing this input sequence through the digital bandstop filter results in the output sequence shown in black. It is evident that the filter has rejected all of the 60-Hz interference leaving the desired 10-Hz signal. (It is a simple exercise on a calculator or computer to generate this input signal and compute the output sequence y(n) according to the difference equation shown in Fig. 2).

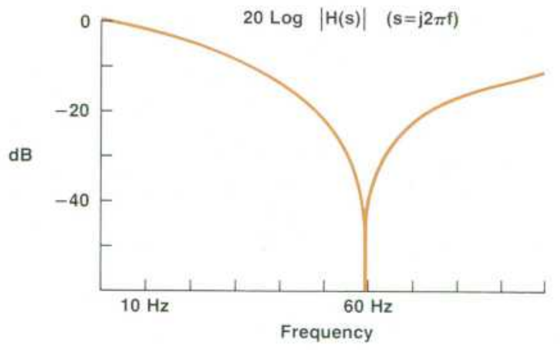
One advantage that the digital filter has in this application is that the position and depth of the bandstop notch will remain constant and will not drift with temperature or age.

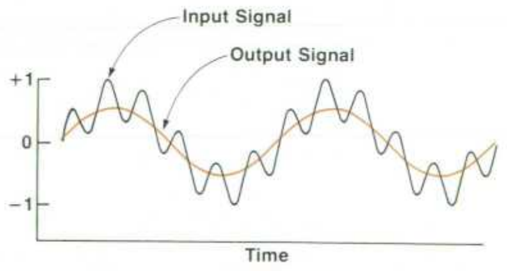
The notch center frequency can be changed in two ways. The first way is simply to change the multiplication coefficient. For example, changing the −1.46 coefficient to −1.62 changes the notch center frequency to 50 Hz.

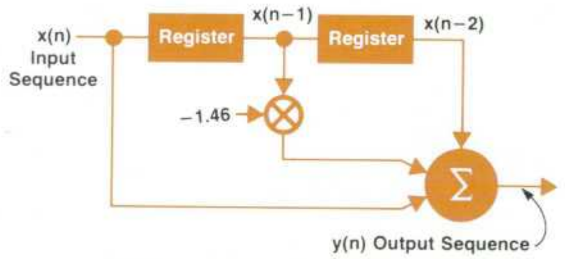
The second way is to change the sample rate. In this particular example, the notch center frequency is 12% of the sample rate. Reducing the sample rate to 417 Hz reduces the notch center to 50 Hz.

In summary, for certain applications digital filters offer advantages in terms of flexibility, stability, and simplicity of control when compared to their analog counterparts.



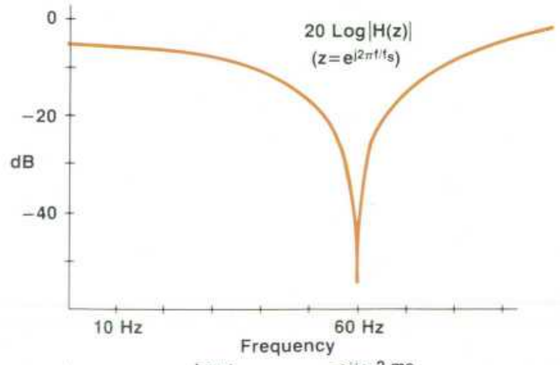






$$y(n)=x(n)-1.46\,x(n-1)+x(n-2)$$

$$H(z)=\frac{Y(z)}{X(z)}=1-1.46z^{-1}+z^{-2}$$



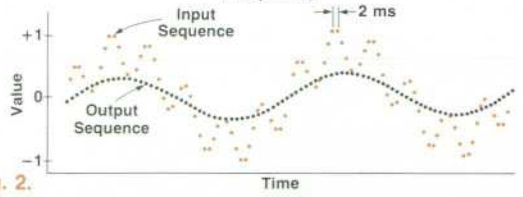


Fig. 2.

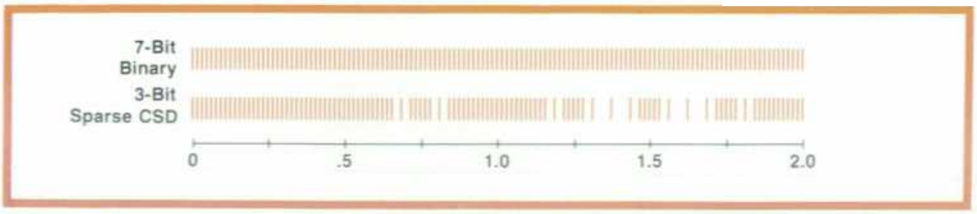


Fig. 3. *Comparison of discrete values between 0 and 2 available with 7-bit binary code and with canonical-signed-digit code limited to three or fewer non-zero bits.*

With reference to Table I, we found that FIR filters require fewer multiplications but these involve more coefficients. IIR filters require less storage but longer coefficients. Linear phase was not of prime importance because the analyzer would have the ability to correct for both gain and phase after the FFT analysis.

To achieve the required multiplication rate with NMOS circuits, several multipliers operating in parallel would have to be included on the chip. Minimizing the number of coefficients that each multiplier had to work with would further simplify the circuitry. Some further space-saving developments with respect to IIR coefficients, to be described next, occurred. Thus an IIR filter design was chosen.

Coefficients

The usual binary coefficients can be represented by the expression:

$$X = \sum_{j=0}^{B-1} x_j 2^j,$$

a weighted sum of B powers of 2 where the weights, x, or bits as we know them, can take on the values 0 or 1.

Another method of representing coefficients, called canonical signed digits (CSD), allows the bits to take on the values 0, 1, and $\bar{1}$ (−1). During the 1950's in the early days of computers, this code was discussed in many publications with reference to fast multiplication.[1]

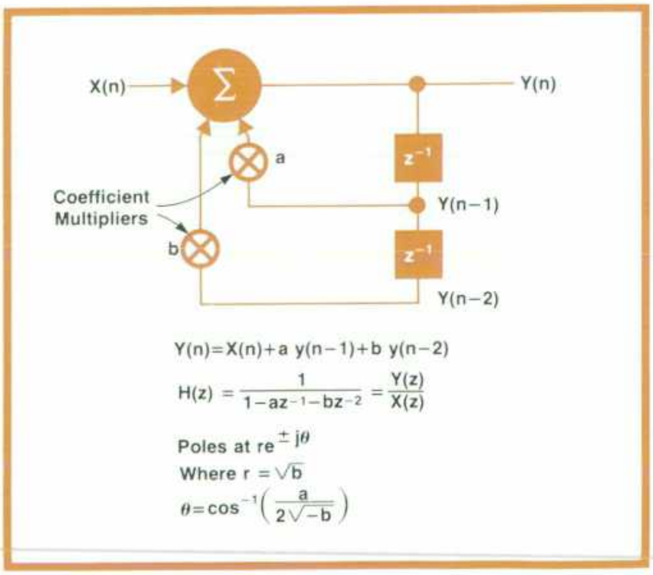
It has been shown that for any integer X, there exists a unique representation in CSD code in which no two consecutive bits are non-zero.[1] Furthermore, *the CSD representation has the least number of non-zero bits*. For example, decimal 31 becomes 011111 in binary but in CSD it is $10000\bar{1}$—only two non-zero bits as compared to five in binary.

It can be shown that CSD representation generally requires 33% fewer non-zero bits than binary. This bit minimization feature of CSD was very important to the realization of the LSI filter. Consider the operation of multiplication. Although there are many multiplication algorithms, the fundamental technique involves shifting and adding. CSD requires fewer additions than straight binary because an add is needed only for the non-zero bit positions in the multiplier word. Although using a CSD multiplier and a binary multiplicand requires a capability for subtraction, this is just as easy to do as addition. (These techniques result in mathematical operations closely related to the well known Booth's algorithm for multiplication.)

This bit-minimization feature along with further developments to be discussed allowed the implementation of separate multiplier circuits for each of the 12 coefficients in the filter structure.

Modified CSD's

Further reductions in multiplier complexity are possible by limiting the number of non-zero bits in each CSD word to three, a representation we chose to call sparse canonical signed digits. This means, of course, that certain values cannot be represented. However, it turns out that this is not a severe restriction. Fig. 3 indicates all the CSD numbers with three or fewer non-zero bits available between the values of 0 and +2 compared to 7-bit binary (fractional powers of two are present and restricted to no smaller than 2^{-6}). An identical set exists for the range −2 to 0. Most IIR digital filters are designed using second-order sections having coefficient values between +2 and −2, a result of the necessity to keep the poles and zeros on or within the unit circle in the z plane. Thus, very few coefficient values would be missed by using



$$Y(n)=X(n)+a\,y(n-1)+b\,y(n-2)$$

$$H(z)=\frac{1}{1-az^{-1}-bz^{-2}}=\frac{Y(z)}{X(z)}$$

Poles at $re^{\pm j\theta}$

Where $r=\sqrt{b}$

$$\theta=\cos^{-1}\left(\frac{a}{2\sqrt{-b}}\right)$$

Fig. 4. *Second-order canonic digital filter section. The boxes labelled z^{-1} are storage registers that delay output sample words by one sample period.*

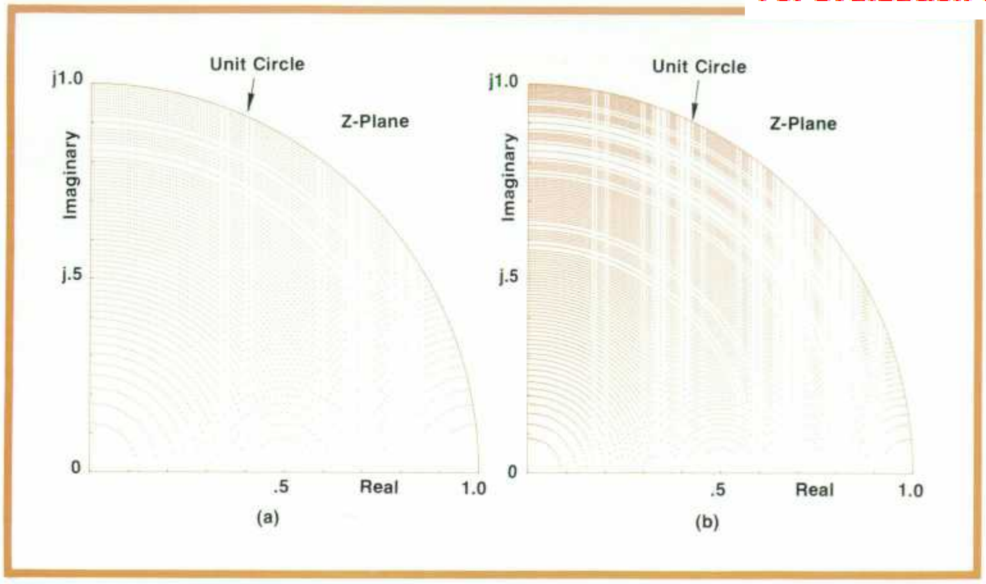


Fig. 5. *(a) Discrete poles realizable with 8-bit canonical-signed-digit code limited to three non-zero bits in the filter of Fig. 5. (b) Poles realizable with CSD code limited to three non-zero bits when nine shift positions are allowed.*

CSD code restricted to three non-zero bits.

Let us now examine the distribution of poles available with sparse CSD in a filter design. Fig. 4 indicates the configuration of a second-order canonic digital filter section (canonic meaning that this structure has the least number of memory elements with respect to other second-order structures; it also involves the least number of multiplications).

The pole and zero locations can be found by substituting all possible sparse CSD values for coefficients a and b in the equations of the z-transform transfer function in Fig. 4. Fig. 5a shows the location of the poles in the first quadrant resulting from this substitution. Since this analysis simply involves finding the root positions of a second-order polynomial, Fig. 5a also shows the location of available zeros. Zeros useful for shaping the stop band of the filter all fall on the unit circle. The missing bands correspond to the missing digits in sparse CSD, as indicated in Fig. 3.

In this analysis the number of non-zero digits was restricted to three and the maximum shift positions to eight, giving the least-significant digit a weight of $\pm 2^{-7}$. The density of pole locations can be increased by allowing more shift positions. Fig. 5b shows the result of allowing an additional shift position but still only three non-zero bits, where the least significant digit has a weight of $\pm 2^{-8}$.

Which Poles and Zeros?

The problem now was to find appropriate filter transfer functions using the available pole and zero positions of Fig. 5b. An iterative procedure was used. Initially, a prototype analog filter was selected from one of the filter design handbooks. Using the bilinear z-transform, the transfer function H(s) of this filter was converted to H(z). Then, with the aid of an HP 2100 disc-based computer system, the pole positions of the z transform were iteratively adjusted to the neighboring sparse CSD positions of Fig. 5b and the resulting frequency responses evaluated. The optimization strategy was directed towards keeping the stopband rejection to greater than 80 dB while minimizing the passband ripple. Equal positive and

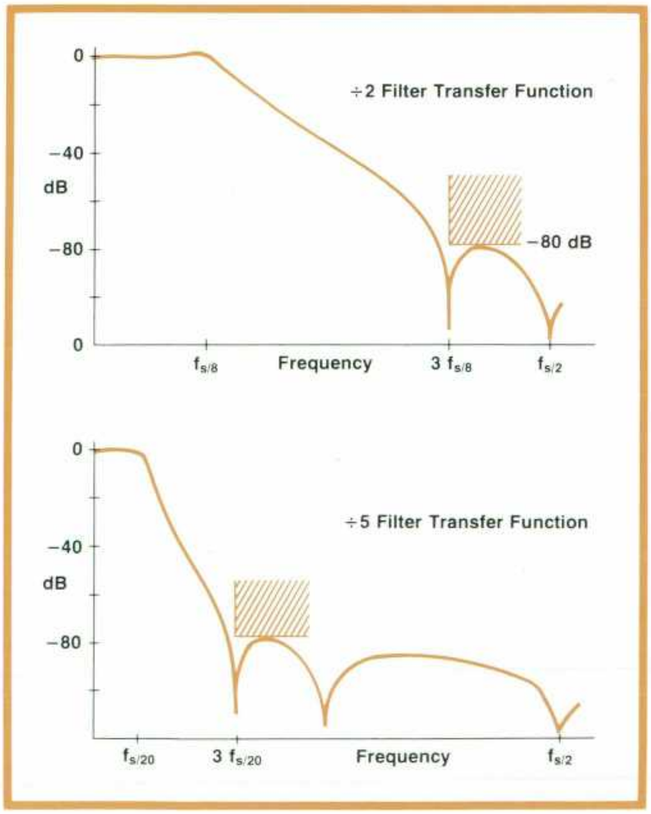


Fig. 6. *Transfer functions of the decimation filter selected with the help of a computer iterative procedure. The procedure derived the functions using pole-zero positions selected from those plotted in Fig. 5b.*

negative excursions of the ripple were desired in both the pass- and stopbands.

This procedure resulted in finding sparse CSD coefficients that contain an average of 1.7 non-zero bits (2.3 for poles, 1.1 for zeros). This compares to the 4 to 7 non-zero bits that normally would be required for conventional binary.

The chosen transfer functions are shown in Fig. 6. The decimation-by-5 filter requires a fifth-order elliptic algorithm having five poles and five zeros and the decimation-by-2 filter uses a fourth-order elliptic algorithm. Although the latter has a less-than-optimum transfer function, it was chosen for reasons to be explained later.

Scaling and Word Lengths

Two other items were considered as an integral part of the design of the filter structure. One concerns overflows at the internal summing nodes. Not only does an overflow cause distortion in the output, but in recursive digital filters an overflow can cause the filter to go into an oscillatory mode that is sustained by repeated overflows. Called overflow oscillation, it is well documented in the literature.

The usual way of dealing with this problem is to scale the signal levels between filter sections to keep the arithmetic values within bounds. The scale factors can be included as part of the feed-forward coefficients that define the zeros. We chose the so-called safe scaling, a conservative approach wherein the theoretical maximum value attainable at each node is determined and the scaling factors are adjusted to keep these values within bounds. Other strategies are available but they require additional overload detection and limiter circuits to prevent oscillations. Safe scaling also insured that no input signal regardless of shape or crest factor could cause overloads in the internal structure of the filter.

Secondly, analysis of the effects of rounding or truncation played a fundamental role in the design. For example, a 27-bit product results from multiplying a 16-bit data word by a 12-bit coefficient. Ordinarily, the product would be trimmed back to 16 bits to conserve storage and chip area but internal noise is generated in filters at every point where this trimming occurs. These noises are summed at the output

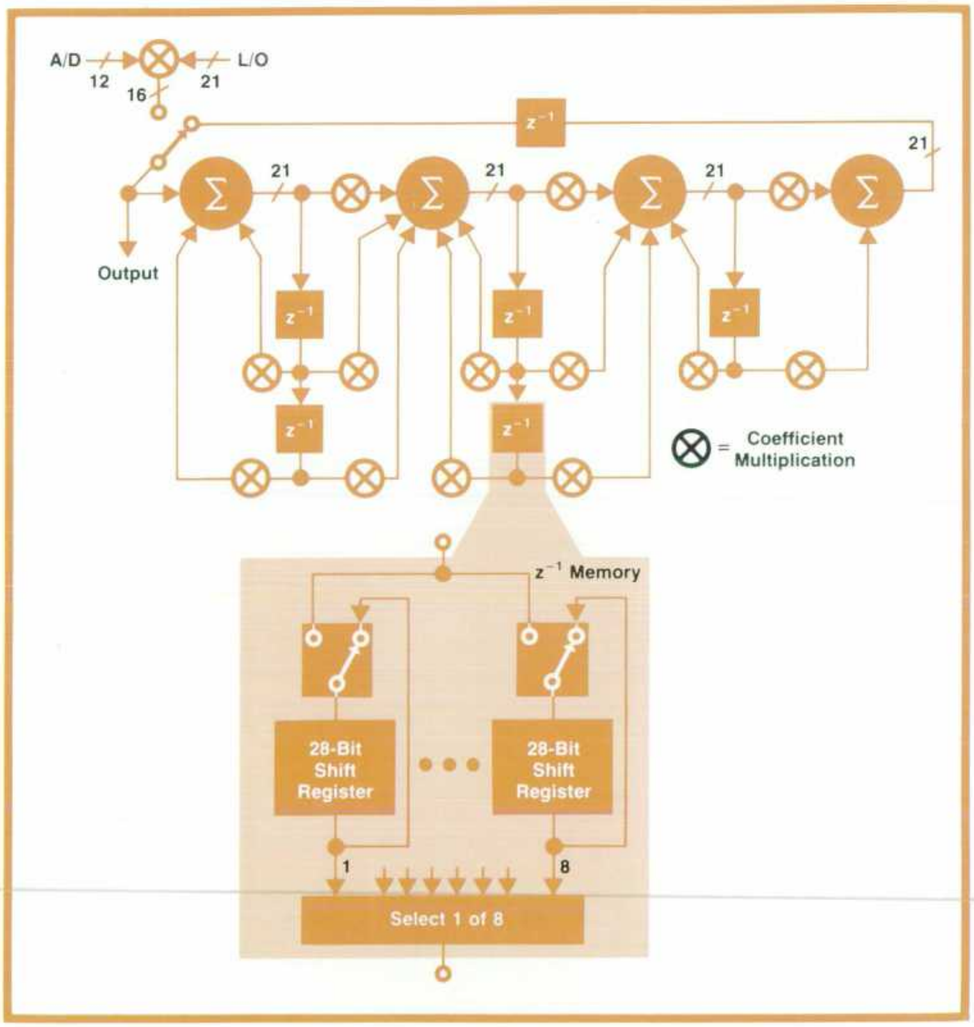


Fig. 7. *Block diagram of the selected filter. This one structure is timeshared by the eight implied filter sections. Eight channels of memory in each z^{-1} block store results from the various filter section computations.*

to form the total self noise of the filter. Analysis and simulation showed that to keep this self-noise at an insignificant level compared to the signal, an interstage word length of 21 bits would be required.

Optimum Structure

The final structure of the filter is related to both the self noise and the scaling. The decimation filter has two second-order zeros, two second-order poles, one first-order zero, and one first-order pole. We chose to implement the filter as three cascaded sections: two second-order and one first-order.

The relative positions of these sections can be permuted within the filter structure, and the poles and zeros can be paired in many combinations. The relative noise-power figures for the various permutations were derived theoretically (and verified once the filter was completed). The quietest structure could then be determined. Actually, the next-to-quietest structure was chosen instead of the quietest because the simpler coefficients used with the non-optimum decimation-by-2 algorithm, mentioned previously, could be formed as a subset of the decimation-by-5 coefficients. This allowed a single processor to be used for both the decimation by 2 and the decimation by 5. Savings in chip area was the reason for this decision.

The resulting arithmetic structure is shown in Fig. 7. The two second-order sections are followed by the first-order section. The multiplier at the filter input multiplies the 12-bit ADC output with the 16-bit "local oscillator" sample to perform the frequency translation for band-selectable analysis (zoom).

The data word length is 21 bits. Of these, 17 bits are necessary for a single filter's operation, an average of 1½ bits are necessary for the combined insertion loss of the eight cascaded filters, and 1½ bits compensate for the accumulated self noise. One bit is added for timing considerations.

As shown in Fig. 7, each memory block has eight shift-register memories to serve as the z^{-1} (sample) delays for the internal nodes of the filters. The feedback memory collects and saves filter outputs to be used as inputs to the subsequent downstream filter operations in the cascade.

Software Emulation

A bit-for-bit model of the filter structure was programmed in software and exercised extensively on the computer to determine the filter's true performance. Roundoff noise was studied and it agreed with predicted performance. Overloading was tested with worst-case inputs.

Probably the most important outcome of this portion of the design was the discovery of small signal limit-cycle problems. These are non-zero outputs even though the input is zero, a consequence of the truncation or rounding of the results of arithmetic operations. Since the normal sample-to-sample changes in the filter input are large compared to the rounding error, the effect is simply the addition of noise, as previously discussed. The limit-cycle non-zero outputs can be either a constant value (deadband), or oscillatory in nature and they are well-documented in the digital-filter literature.

During early investigations of the filter algorithms, analysis indicated that the data word lengths were sufficient to keep the limit-cycle amplitudes at acceptable levels. However, in experiments with the software model, more severe limit cycles appeared. Investigation revealed that several filters in the cascade would limit cycle in a "synchronous" manner such that the limit-cycle output of the last filter in the cascade was well above the noise level.

Rather than increase the data word length to reduce the limit cycle amplitude, we chose to inject a low-

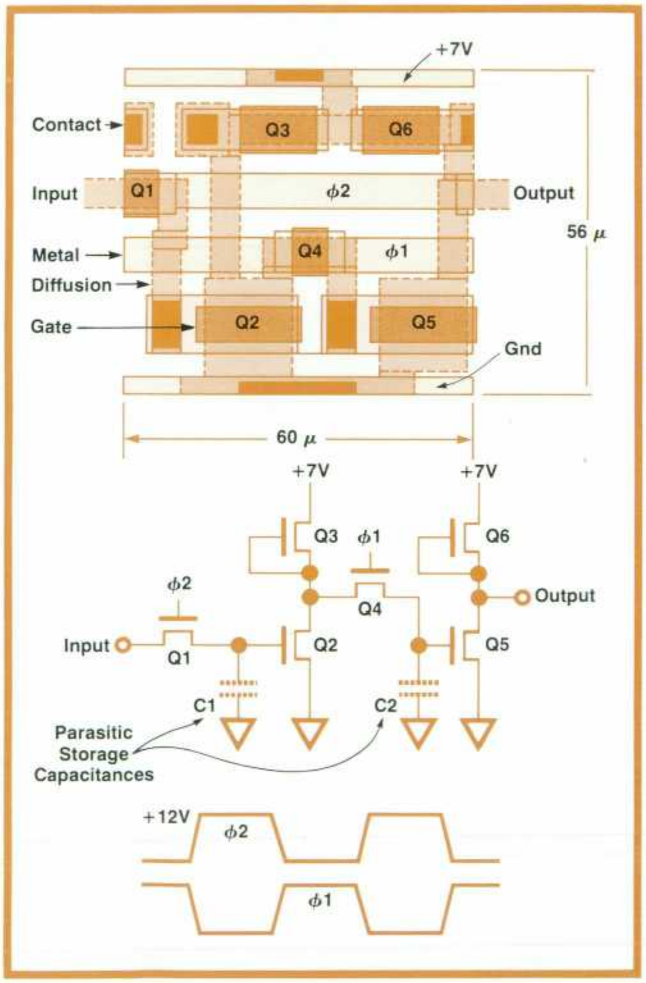


Fig. 8. *Shift register memory cell used in the LSI implementation of the digital filter. A bit is stored in parasitic capacitance C1 when clock φ2 occurs, and it is transferred to capacitance C2 on clock φ1.*

level dither signal into the rounding input of the summation nodes of the two second-order sections. The dither variance is sufficient to break up the limit cycles but low enough to be invisible compared to the signal. As a further precaution, the dominant spectral energy of the dither signal was placed in the filter's transition band.

Hardware Simulation

The digital filters were then simulated in hardware. This required about 450 TTL IC's per filter, this many being required because the simulated filters were designed to be as close as possible to the proposed LSI design. Many of the details and problems of the distributed control and timing system were debugged and improved with the aid of these filters. Along with the debugging performed with the software model, this proved the integrity of the design before the LSI design was complete.

The simulated filters also enabled lab prototypes of the spectrum analyzer to be built and evaluated long before the filter LSI filter chips became available.

LSI Implementation

The final design was implemented in LSI with the Hewlett-Packard NMOS-II process.[2] Typical of the kind of circuits used is the one-bit memory cell shown in Fig. 8. This design was chosen because it uses less chip area and loads the clock less than other designs. Twenty-eight of these are cascaded to form each of the eight circulating shift-register memories in each delay element shown in Fig. 7. These registers operate at a 6-MHz clock rate.

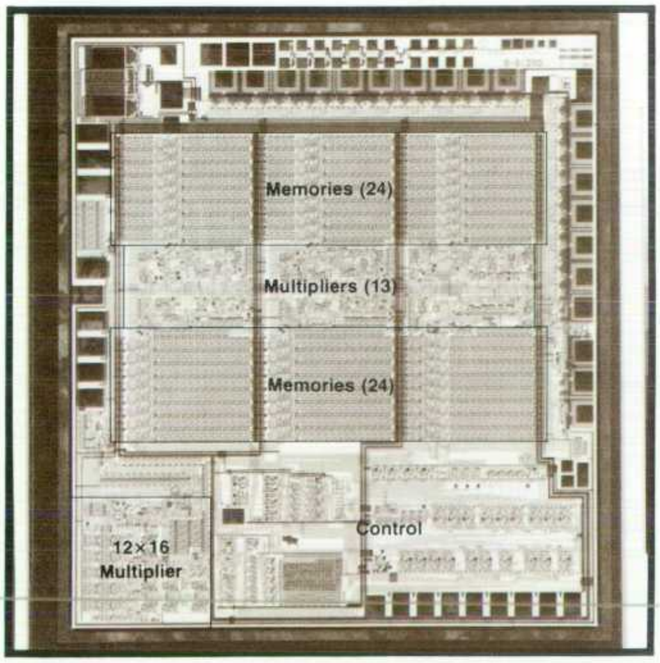


Fig. 9. *Digital filter chip contains about 16,500 transistors and measures 4.73 ×5.27 mm.*

Other circuits on the chip include full adders, counters, R-S flip-flops, D-latches, ROMs, and assorted logic gates. A key element in the design phase was the inclusion of about seventy-five 20μm-square test pads at critical points. During the debug phase, digital sequences at these points resulting from a test input were compared to sequences generated by the software simulator. Debugging could therefore be performed in a manner similar to signal tracing. This saved a substantial amount of time during the chip turn-on phase.

The final chip is shown in Fig. 9. Of special interest here are the areas devoted to multipliers. All of the arithmetic required to perform the decimation-by-2 and decimation-by-5 algorithms is located in the central portion of the chip. They perform the equivalent of 13 multiplications every 5 μs, using the sparse canonical signed digits. By contrast, the input "mixer" (multiplier) at the lower left of the chip is of standard design, it performs one 12×16 multiplication every 5 μs.

Acknowledgments

No IC is designed without the help of many people and the full support of an organization. My thanks to all. Special thanks go to Lou Scheffer who performed most of the chip device design and layout, to Yvette Norman and other members of the LID computer-aided artwork department for their expert help in managing the multitude of rectangles that went into the chip design, and to Dick Toftness and John Stanback of the IC engineering department for their process and design guidance.

References

1. G.W. Reitweisner, "Binary Arithmetic," Advances in Computers, Vol. 1, Academic Press 1960.
2. J.E. Deweese and T.R. Ligon, "An NMOS Process for High-Performance LSI Circuits," Hewlett-Packard Journal, November 1977.

Lynn A. Schmidt
A native of Colorado, Lynn Schmidt did his undergraduate work at Rensselaer Polytechnic Institute and the Rochester (N.Y.) Institute of Technology, earning a BSEE degree at the latter in 1969. He then worked in sonar signal processing, obtaining an MSEE degree at Syracuse University along the way (1972). Lynn joined HP in 1974 and did some circuit design in voltmeters and signal sources before joining the 3582A project team. His outside interests include skiing and making solid oak furniture for his home. He and his wife and three children live in Loveland, Colorado.