Multicarrier Demodulator Architecture for Onboard Processing Satellites

L. P. Eugene,* P. J. Fernandes,* M. M. Jamali,† and S. C. Kwatra‡
University of Toledo, Toledo, Ohio 43606
and
J. Budinger§
NASA Lewis Research Center, Cleveland, Ohio 44135

A parallel pipelined architecture is presented for demultiplexing and demodulating single channel per carrier/frequency division multiple access (SCPC/FDMA) channels in real time. Specific algorithms are selected for each of the operations necessary for multicarrier demodulation. The selection is made based on their suitability for implementation into parallel-pipelined and sharing schemes. The algorithms are analyzed for data dependencies and divided into data dependent and independent sections. The segregated sections are mapped by implementing the independent sections in a parallel scheme and the dependent sections in a pipelined scheme. The demultiplexing is performed by the polyphase fast Fourier transform (FFT) method, which requires a bank of filters followed by an FFT operation. A shared filter bank module and a pipelined FFT module are designed to implement the bank of filters and the FFT operation, respectively. The demodulator uses a single hardware module that is shared amongst all of the channels for the recovery of timing, carrier, and data. This sharing of hardware to perform the demultiplexing and demodulating of all of the channels results in savings of power and hardware. The system is suitable for onboard processing of signals in satellites where power and area requirements are critical. The architecture is also highly modular and therefore lends itself well to very large scale integration (VLSI) implementation. This design is illustrated for the specific case of processing 800 FDMA channels at 64 kbps each.

I. Introduction

PRESENTLY, most satellite communication systems require a large Earth station to supply a network with multiplexed speech and video channels. In the future, satellite communication systems will consist of a large number of small-capacity, multiservice users. For these systems the conventional transmission methods of frequency division multiple access (FDMA) or time division multiple access (TDMA) are no longer efficient. One approach to offer these services at a low cost to the user is to use single channel per carrier/frequency division multiple access (SCPC/FDMA) on the uplink and time division multiplexing (TDM) on the downlink.[1-4] Using FDMA on the uplink and TDM on the downlink has been found to be very effective in improving satellite transponder utilization and reducing the effective isotropic radiated power (EIRP) in both the satellite and the Earth stations. It also results in low cost and less complex Earth terminals, which makes communication via satellites more economical. A drawback with this scheme is that it transfers the burden of computation on-board the satellite, where power and area requirements are critical. Thus, hardware that is efficient in terms of speed, power consumption, and components needs to be developed for performing the computations onboard the satellite.

To perform the FDMA/TDM conversion, a multicarrier demodulator (MCD), baseband switch matrix, TDM multiplexer, and modulator are required onboard the satellite as shown in Fig. 1. The MCD consists of a transmultiplexer followed by a bank of demodulators. The transmultiplexer is required to separate the FDMA signal into individual channels. The bank of parallel demodulators takes the separated channels and recovers the digital data from them. Implementation of the MCD using analog components results in large requirements of power and weight. Reliability is also a serious concern with analog components. To overcome these problems and to take advantage of the developments in the area of very large-scale integration (VLSI) and digital systems, a digital implementation of the MCD has been proposed.[5-9] Digital implementation also allows flexibility in processing any number of channels with different bandwidths and access formats. This can be accomplished by making the transmultiplexer and demodulator programmable and sending programming instructions from the ground for the desired processing.

For digital implementation of the conventional MCD shown in Fig. 1, the channel separation is achieved digitally via a bank of digital filters with desired center frequencies, corresponding to the K carriers in the FDMA signal. After the channel separation, each carrier is downconverted to baseband and decimated. Since the filtering, downconversion and decimation are done at high sampling rates, this scheme is computationally inefficient and has high power requirements.

If the SCPC carriers are uniformly spaced, the sampling rate reduction, the channel separation, and the downconversion are efficiently realized as one signal processing block known as a transmultiplexer (TMUX) or a uniform channel filter bank,[10,11] as shown in Fig. 2. The filter-bank structure is implemented with the aid of a commutator and a bank of polyphase filters, and the discrete Fourier transform (DFT) implemented via the fast Fourier transform (FFT), which

Received Nov. 10, 1989; presented as Paper 90-0812 at the AIAA 13th International Communication Satellite Systems Conference, Los Angeles, CA, March 11-15, 1990; revision received Sept. 11, 1990; accepted for publication Oct. 19, 1990. Copyright © 1990 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental purposes. All other rights are reserved by the copyright owner.

*Graduate Research Assistant, Department of Electrical Engineering.

†Associate Professor, Department of Electrical Engineering.

‡Professor, Department of Electrical Engineering.

§Lead Engineer, Digital Systems Technology Branch, Space Electronics Division.

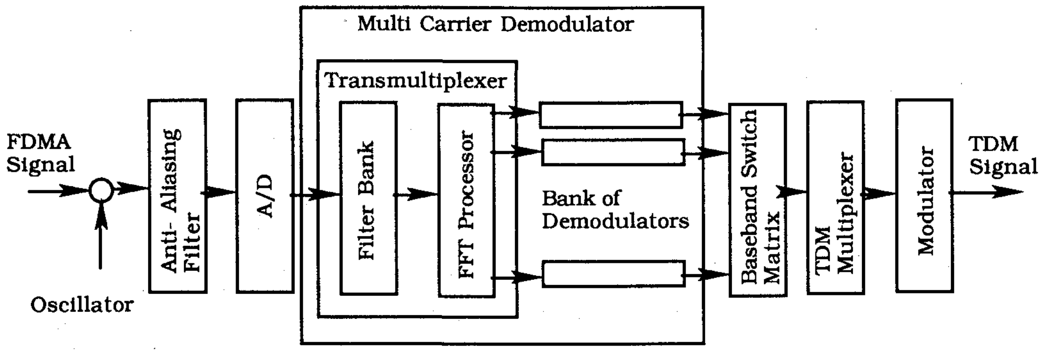

Fig. 1 FDMA/TDM system.

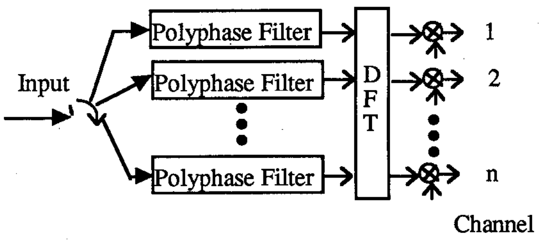

Fig. 2 Polyphase FFT method.

makes this algorithm computationally efficient. However, this scheme requires a large amount of hardware since a separate polyphase filter and a separate demodulator are required for each channel. In this paper, a shared hardware architecture that uses a single polyphase filter and a single demodulator is presented. The FFT is implemented by a pipelined architecture that is compatible with the shared hardware architecture. Demodulation requires modules for carrier recovery, timing recovery, and data detection. Algorithms suitable for shared architecture and for VLSI implementation are selected for various operations of demodulation. These are described in Sec. II. Hardware design of the polyphase filter, FFT processor, and the demodulator is given in Sec. III, and the estimates of the power required to implement the hardware are given in Sec. IV.

II. Algorithm Selection for Demodulation

The demodulator consists of modules for carrier recovery, timing recovery, and data recovery. Several algorithms can be used for the carrier and clock recovery. The nonlinear estimation technique proposed by Viterbi and Viterbi[12] for the carrier recovery and the clock estimation method proposed by Gardner[13] have been chosen for their simplicity in a digital implementation. The carrier recovery is based on the implementation of the Viterbi algorithm in the proposed MCD structure by Del Re and Fantacci.[9] The samples are input to the demodulator at the rate of two for every symbol. The complex sample input to the demodulator is of the form

$$X_n = I_n + j\,Q_n = R_n \cdot e^{j\phi_n} \tag{1}$$

where $R_n = \sqrt{(I_n{}^2 + Q_n{}^2)}$ and $\phi_n = \tan^{-1}(Q_n/I_n)$.

A nonlinear transformation is peformed on the samples to eliminate the modulation information from the phase of the carrier. The new transformed vector is as follows:

$$I_n' + j\,Q_n' = F(R_n)\cdot e^{jm\phi_n} = F(R_n)\cdot e^{jm\phi_c} \tag{2}$$

where ϕ_c is the carrier phase to be estimated, the function F is an arbitrary nonlinear operator, and m is related to the mPSK scheme of modulation [m = 4 for (QPSK)]. The operations performed are a rectangular-to-polar transformation, a phase multiplication by m, an arbitrary nonlinear transformation on R_n, and finally a polar-to-rectangular transformation. In hardware implementation, the values of I_n' and Q_n' are preprogrammed and stored in a look-up table for the given I_n and Q_n values. All of the symbols of various channels use the same look-up table for this transformation. The carrier-phase estimate is given by $\phi_c = 1/m \tan^{-1}(Q_n'/I_n')$. The carrier phase is estimated at the midinterval for a period of samples of each channel. For every $2N+1$ symbols of a given channel, this phase estimate is used to demodulate all of the $2N+1$ symbols.

The timing information in a clock-recovery module is obtained for each of the symbols. The timing-error information for the nth sample is given by Ref. 13.

$$U_n = I_{n-1/2}(I_n - I_{n-1}) + Q_{n-1/2}(Q_n - Q_{n-1}) \tag{3}$$

where the integer-index samples denote the estimated peak values of the samples and the noninteger index samples denote the crossover points that lie midway between the peak samples. The accumulated timing-error is given by

$$W_n = W_{n-1} + U_n \tag{4}$$

This updated output is needed by the interpolator/decimator filter for the correction in the sampling instances. The updated timing error is computed for the samples of all of the channels.

The demodulation process for the samples begins once the phase estimate is available from the carrier recovery. The demodulated samples are given by

$$A_n = I_n \cos\phi_c + Q_n \sin\phi_c \tag{5}$$

$$B_n = Q_n \cos\phi_c - I_n \sin\phi_c \tag{6}$$

where A_n and B_n are the demodulated data values for the inphase and quadrature channels, respectively, and ϕ_c is the estimated carrier phase. The values A_n and B_n are then utilized appropriately to output the digital data.

From the above discussion, it is clear that the major components to be designed for a digital MCD are the polyphase filter module, the FFT module, and the demodulator. In the following section, an efficient implementation of these modules is presented for the specific case of demultiplexing 800 QPSK modulated FDMA channels at 64 kbps and a uniform spacing of 45 kHz each. The total bandwidth of the FDMA signal is 36 MHz. This is a typical application in satellite systems for transmitting voice signals. The concepts developed can be easily extended to design architectures for other similar applications.

III. Hardware Design

The process of multicarrier demodulation starts with the analog/digital (A/D) sampling. The FDMA signal is complex sampled at the rate of 36 MHz. Complex sampling is chosen so that the digital components can operate at a lower rate. In addition, the complex signal representation is compatible with the FFT operation. For a sampling rate of 36 MHz, the system clock period will be less than 27.7 ns (1/36 MHz). Two A/D converters are needed, one for the I, and one for the Q channel. This will give an internal bit rate of 288 Mbits/s (36 MHz $\times$ 8 bits). The samples are input to the polyphase network at the rate of one complex sample every 27.7 ns. These input samples are operated on by the hardware units in the polyphase filter bank, FFT, and the demodulator. This is feasible by incorporating the principles of time multiplexing, pipelining, and using hardware components that can operate at the sampling frequency.

A. Polyphase Filter Architecture

As shown in Fig. 2, a bank of polyphase filters is required. Each filter in the polyphase network is derived from the prototype low pass filter.[10] If the prototype filter has M taps and there are N channels to be demultiplexed, the N polyphase filters would have $K = M/N$ taps each. The polyphase filters are implemented as finite impulse response (FIR) filters because infinite impulse response (IIR) filters are not suitable for this application. The coefficients of the prototype low-pass FIR filter are derived for the specific case of 800 FDMA channels each having a bandwidth of 45 kHz and a bit rate of 64 kb/s. The prototype low-pass filter has 7200 coefficients and therefore each polyphase filter will be implemented as a nine-tap FIR filter.

The individual nine-tap FIR filters can be implemented in a variety of ways depending upon whether speed or economy is of prime importance.[14] For filtering the proposed 800 channels, each of the 800 filters will have 22.22 μs (800/36 MHz) to compute its result. This speed is easily achievable with present technology. However, the implementation of the filter bank directly as shown in Fig. 2 will require hardware components necessary for implementing 800 filters. If high-speed processing components are available (ones that can operate at several times the processing rate necessary for implementing each of the 800 filters), time multiplexing can be used to reduce the number of components. It is proposed that instead of having 800 different nine-tap filters, the hardware of one nine-tap filter be shared in a time-multiplexed manner amongst the 800 polyphase filters. This is accomplished by designing the hardware structure for one nine-tap filter and passing the appropriate coefficients that define a desired filter. The coefficients are stored in a high-speed memory. Using this approach, any desired filter can be configured by accessing the appropriate filter coefficients from memory. Thus, in this architecture the structure for one filter is time multiplexed amongst the 800 filters giving an approximately 1:800 saving in hardware components over directly implementing the polyphase filter bank as shown in Fig. 2.

The architecture of the shared polyphase filter bank for the case of 800 (nine tap) filters is shown in Fig. 3. In this structure, each high speed random access memory (RAM) corresponds to one of the registers in the nine-tap filter. In essence, the RAMs used in the current structure act as multiple registers connected to multipliers. An address generator generates the read and write addresses by counting from 1–800. The read and write addresses are the same through one cycle of reading and writing. When the ouput of the counter is n, channel n's contents are being accessed, i.e., address broadcast to the RAMs is n.

Latches are provided between the RAMs for storing samples. These latches help to pipeline and hold the data through one cycle of read and write. The data held in the latch is written into the RAM following it and is also an input to the multiplier where it will be multiplied by its corresponding filter coefficient. The latches are negative-edge triggered. The multipliers multiply the samples from the RAMs with the corresponding filter coefficients which are stored in a memory. Outputs of the multipliers are input into a tree adder which is pipelined. Thus, once the pipeline is full, a weighted sample is produced every clockcycle.

The shared filter system will operate at a 36 MHz clock rate. The A/D converter produces a complex sample every 27.7 ns. For every clockcycle, a signal is provided to the A/D converter to start sampling. At the same time an address is broadcast to all memories. Once the incoming signal has been completely digitized and the data from the memories have been accessed, the latches are enabled. Thus, the latches hold the data read from the memories. After a setup time, a write signal is broadcast to all of the memories. The write address is the same as the read address. Thus, all the of data available at the latches will be written into the same location in the next RAM. This procedure essentially performs the shifting of data. The data at the output of the latches is also an input to the multipliers. The coefficients of the filter corresponding to the channel being filtered are also sent to the multipliers at the same time as the data is available at the multipliers. Once both the inputs are available at the multipliers, a multiply signal is broadcast.

From the above, it can be seen that a new output sample for each channel will be calculated every clockcycle. Since we are performing nine multiplications corresponding to the nine taps in parallel, the time to compute all of the multiplications will be one multiply time or one clockcycle (27.7 ns). Since the data is complex and the coefficients are real, the complex multiplier could be implemented as two multipliers operating in

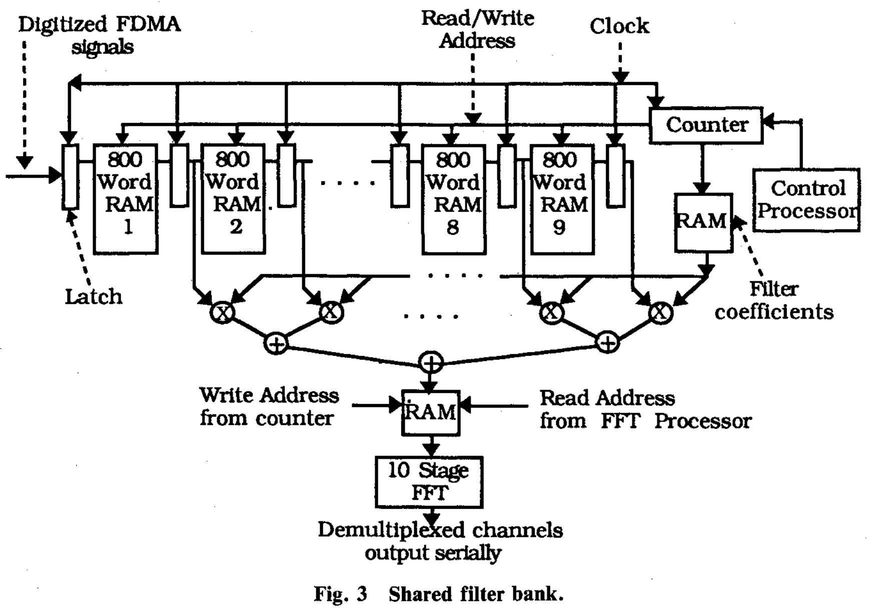

Fig. 3 Shared filter bank.

parallel on the real and complex part of the sample. The rest of the structure consists of pipelined adders. After the pipeline is full, an output is obtained in approximately the same time as it takes to perform one multiplication.

Outputs of the filter bank are passed through the DFT module. It can be seen that 800 samples will be available for DFT computation every 22.22 μs (800 x 27.7 ns). The DFT is implemented via an FFT for efficient computation.

B. Pipelined FFT Implementation

For demultiplexing 800 channels, an 800-point DFT needs to be computed. The FFT algorithm is used for fast computation of the DFT. For efficient FFT implementation, a 1024-point FFT (closest power of two above 800) is computed instead of a 800-point FFT. Thus, the requirement is that a 1024-point FFT has to be computed every 22.22 μs. There are many methods of implementing the FFT processor (perfect shuffle, pipeline, array method) available in the literature.[14-17] After reviewing the various methods of implementing the FFT, the direct pipeline was found to be the most attractive.

An N-point FFT will have $\log_2 N$ stages in it; therefore a 1024-point FFT will have ten stages. Each stage in the pipeline will consist of an arithmetic element (AE), a memory that will store the twiddle factors, a RAM for storing its output, and an address generator. The RAM that stores the output of each stage is divided into two sections. Section 1 ranges from 0–1 k and section 2 varies from 1–2 k. This is to enable simultaneous reading and writing into and from different sections of the same memory. The nth RAM stores the result of the $(n-1)$th stage of the FFT. The implementation of these components in the pipeline structure can be seen in Fig. 4. The data flow can be better visualized if the example of a particular stage in the pipeline is taken. The flow for stage n is as follows.

There are two operating modes for the pipeline. The computations of one stage of the FFT, i.e., $N/2$ butterfly operations, are completed in one mode whose time span is 22 μs. The pipeline operates by alternating between each of the two modes (mode 1, mode 2, mode 1,...). The operations performed for stage n are detailed below. All of the operations for each mode take place in one clockcycle.

In the first mode (mode 1) the address generator (AG) of the Nth stage will supply read address (RA) into section 2 of RAM N and supply write address (WA) to section 1 of RAM $(N + 1)$. That is, in mode 1, the data is supplied to the AE from section 2 of the RAM preceeding it and written into section 1 of the RAM following it. In the second mode (mode 2), the AG of the Nth stage will supply RA into section 1 of RAM N and supply WA to section 2 of RAM $(N + 1)$. The necessary FFT coefficients required by the AE are also accessed from the coefficient memory. The data are then input into the pipelined AE where a butterfly operation takes place. Since the read and write addresses are the same, the write addresses will be delayed by a number of clockcycles equivalent to the number of pipeline stages between the two RAMs, N and $(N + 1)$. There are two reasons for having this sequence of reading and writing in the two different modes. One reason is so that we read only those stages that have been completely computed. Another is so that the AE is kept busy all of the time.

The total number of butterflies required for performing a 1024-point FFT is 5120. Since there are ten stages in a 1024-point FFT, each stage will have to compute 512 butterflies. The time elapsed between the entry of a new set of 800 data inputs from the shared filter module is 22.22 μs. Therefore, the time required for each stage to compute its result is 22.22 μs. This translates into performing 512 butterflies every 22.22 μs. Therefore, the minimum time to perform a butterfly is 42.9 ns (22.22 μs/512). Commercial multipliers are available that can perform a 16 $\times$ 16 multiplication in 20 ns. Considering these computation times, a multiplexed arithmetic element (MAE) has been developed. The architecture of the MAE is shown in Fig. 5. This AE is designed by studying the mathematical operations of the FFT butterfly and realizing that the butterfly operations can be split into two parts. The first part of the butterfly operation is computed in the first 20 ns and the next part in the next 20 ns. Both parts are computed utilizing the same hardware and a little more complicated control sequence. Thus, the butterfly can be computed in the required time of 42.9 ns. The benefit of this approach is that hardware and power consumption can be saved.

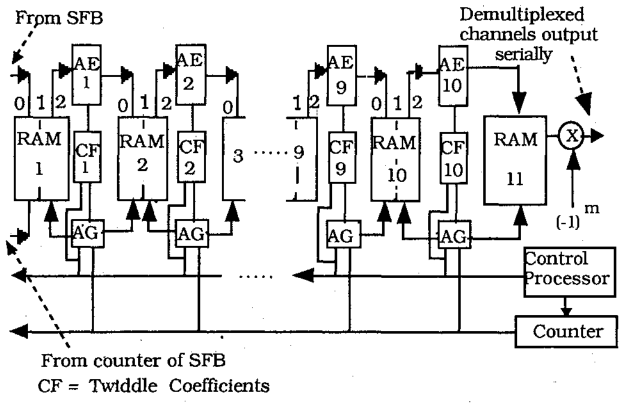

Fig. 4 Ten-stage pipelined FFT.

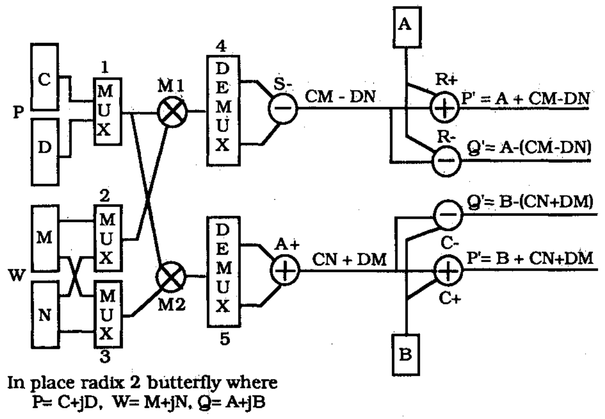

Fig. 5 Multiplexed arithmetic element (MAE).

There are two pairs of input latches (c,d and m,n) in the MAE. One pair of latches (c and d) holds the lower leg of the butterfly inputs and the other pair of latches (m and n) holds the twiddle factor. The outputs of the latches are connected to multiplexers (1,2,3) that route the data to the multipliers (M1,M2). The data after being multiplied is routed by means of two demultiplexers (4,5) to different inputs of an adder (A+) and subtractor (S−) as shown in Fig. 5. The upper leg of the butterfly inputs is held in another set of latches (a and b). The data of the upper leg of the butterfly is added and subtracted to the main data flow by means of two adders (C+,R+) and subtractors (C−,R−). The clockcycle for this architecture is 21 ns. The inputs to the multiplexers are held constant for two clockcycles (42 ns), and a complete result will also be produced every two clocks. This shows that the required 512 butterflies can be performed in 21.5 μs (512 $\times$ 42 ns). Thus, the 800 FDMA channels are demultiplexed and brought down to baseband in real time.

C. Time Shared Demodulator

A single demodulator is shared to demodulate all of the channels in a time multiplexed manner, and consists of three main modules, namely multiplexed carrier recovery module (MCRM), multiplexed timing recovery module (MTRM), and multiplexed data recovery module (MDRM). The hardware is structured to demodulate all of the channels simultaneously. This amounts to multiplexing the samples of all of the channels. The number of channels can be varied as long as the total bit rate is below the maximum bit rate sustained by the

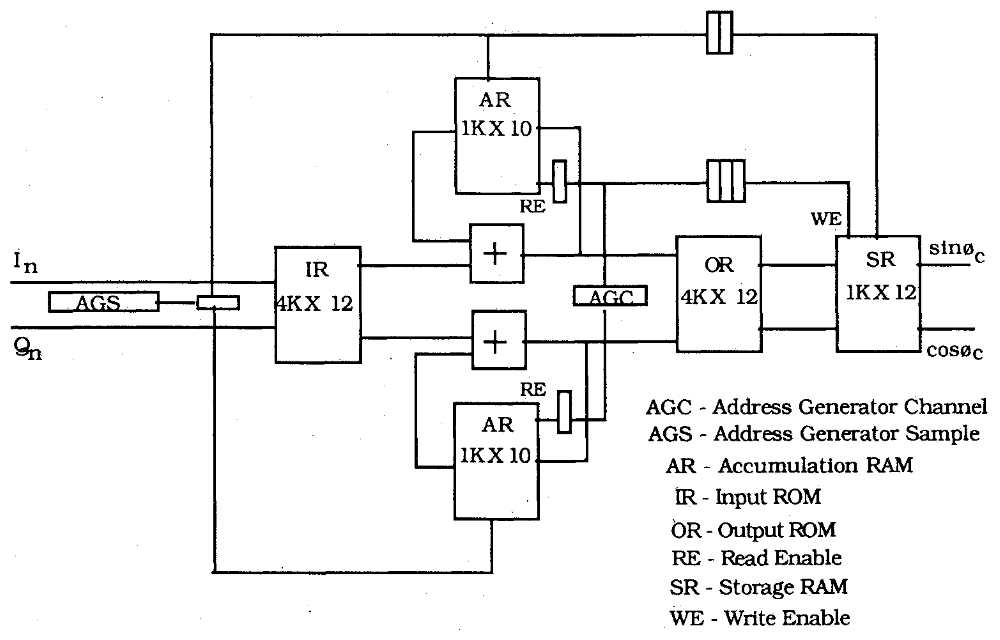

Fig. 6 Multiplexed carrier recovery module (MCRM).

modules. Moreover, the bit rate of a group of channels could also vary. The detailed hardware design is provided in the following sections.

1. Multiplexed Carrier Recovery Module (*MCRM*)

The samples on the in-phase and quadrature-phase channels are input to the MCRM as shown in Fig. 6. The input samples I_n and Q_n need to be transformed to another vector according to Eqs. (1) and (2). The transformed samples are accumulated for an estimation interval and their mean is obtained. The outputs of this module are the sine and cosine values corresponding to the average of the accumulated samples.

The preprogrammed values of I_n' and Q_n' corresponding to the transformation of the input samples I_n and Q_n, according to Eq. (2), are stored in the input read-only memory (ROM)(IR). The values of I_n' and Q_n' are accumulated in the accumulation RAM (AR). An AG for samples supplies the addresses to the AR according to the channel numbers for storing the values of I_n' and Q_n'. The samples of the various channels are stored in their allotted location. The successive samples over an interval length of $2N$ for each of the channels are accumulated. The accumulated result needs to be divided by $2N$ to obtain an average vector (I_n'' and Q_n''). This is achieved by reading all but the last $\log_2(2N)$ least significant bits of I_n' and Q_n'. The new vectors (I_n'' and Q_n'') are addresses to the output ROM (OR) that contains the look-up table for sine and cosine values of the phase estimate. This avoids an extra look up for the $1/m \tan^{-1}$ operation. The sine and cosine values are stored in the storage RAM (SR) for operating on the values of I_n' and Q_n'. The estimated values of the sine and cosine are used by the MDRM.

It is assumed that the word length of the input samples is six bits. Therefore a 4-k $\times$ 12 IR is sufficient for the vector transformation. A 10-bit 1-k AR will be sufficient to accomodate the accumulated $2N$ (16) successive six-bit samples. A 4-k $\times$ 12 IR is used to store the look-up table for the sine and cosine values that are 6 bits each. A four-bit AG for channels and a ten bit AG for samples are sufficient to control the accumulation of up to 16 succesive samples of each channel and to count the 800 channels, respectively.

The AG for samples can be configured from the ground according to the user specifications. This is essential if the user needs to vary the number of channels as well as the bit rate of a group of channels. The clock for AG for channels is slower than the clock for AG for samples by 800 times.

2. Multiplexed Timing Recovery Module (*MTRM*)

The timing recovery algorithm used in this design is as proposed by Gardner,[13] and its implementation for the multichannel case is presented in Fig. 7. The input samples are obtained from the MDRM and are stored in the buffer RAMs (BR). These samples are input into this module every clockcycle. The input samples are stored in the first RAM and are successively moved from one RAM to the next until the three samples needed for the computation of Eq. (3) are received. The data are effectively transferred from one RAM to another using the intermediate latches. On the positive edge of the clock, the input is written into the latch; and on the negative edge, it is written into the appropriate location of the RAM indicated by the AG for samples. It provides the addresses for storing the input samples in the unique channel-space locations allotted in the RAMs. The three sets of samples for all of the channels are stored in the BRs.

The computations will start once the samples in all of the RAMs are available. The timing error for each channel is computed according to Eq. (4) and is updated and stored in a timing RAM (TR). The address for TR is delayed by three clockcycles to compensate for the pipeline delays from the AG for samples and is enabled by the AG for channels with an appropriate delay. This amounts to performing the computations for every other clock of the AG for channel. A correction in timing is available only for successive symbols that have a transition. This error is accumulated with the previous values stored in the memory and is also available for timing correction. All of the values in the RAMs are initialized at the start of these operations.

3. Multiplexed Data Recovery Module (*MDRM*)

The data recovery module utilizes the in-phase and quadrature-phase samples along with the output values of the MCRM to extract the digital data for all of the channels. The samples of the channels are input sequentially into this module. The product of their combinations according to Eqs. (5) and (6) are obtained and stored in a latch. After the necessary computations are performed, the results are stored in unique locations of the RAM. An appropriate delayed address is provided by the AG for samples. The structure of the MDRM is shown in Fig. 8.

The result of the computations stored in the latch can either be negative or positive. The sign of the values stored in the latches is determined by examining the most significant bit

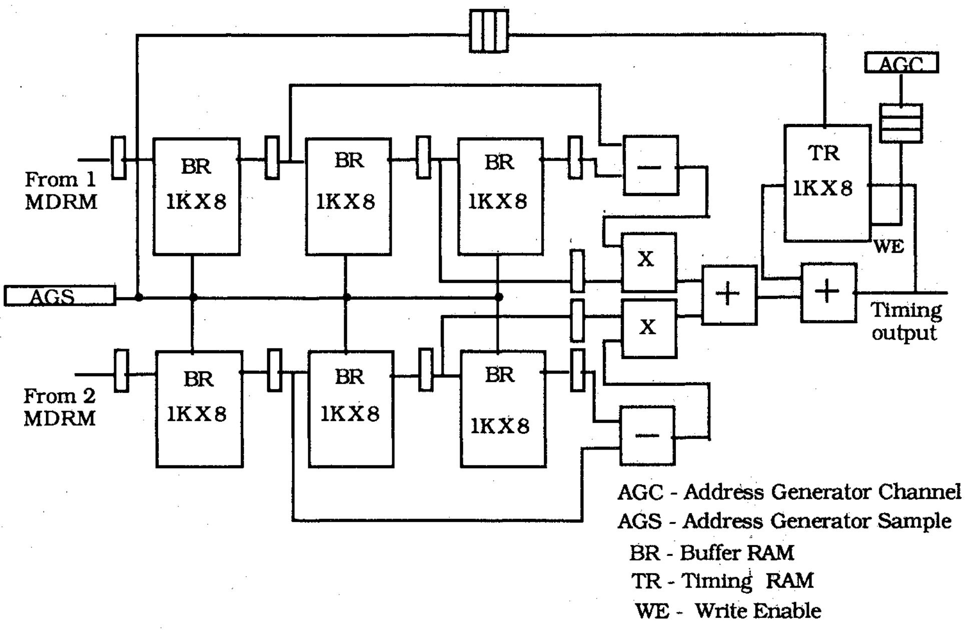

Fig. 7 Multiplexed timing recovery module (MTRM).

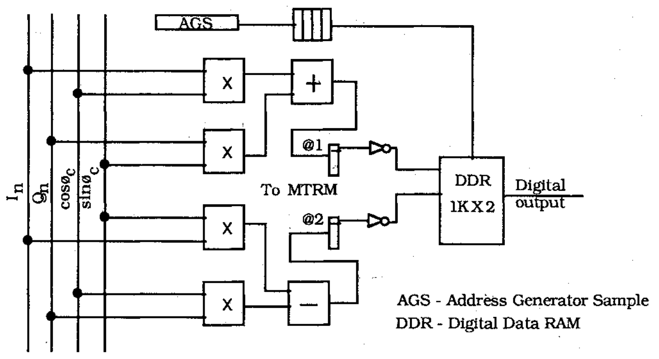

Fig. 8 Multiplexed data recovery module (MDRM).

(MSB) of the latch. Whenever there is a positive value in the latch a "1" is transmitted, and a "0" is transmitted for a negative value. These output values are a two-bit data bus to the digital data RAM (DDR). An AG for samples will address the recovered channel information appropriately to the different locations of the DDR according to the unique channel spacing alloted to them. These stored values are accessed by the switch matrix for further processing on TDM downlink.

The AG for samples provides the addresses for appropriately storing and accessing the channel information. For 800 channels (64 kbps each), the modules have to operate at a 27.7-ns clock period. This throughput can be maintained by the modules once the pipelines are filled and also with all of the components operating at that rate. The 800 samples of the channels are stored in the 800 specific locations of the 1-k RAMs. The width of the RAMs is determined by the word length of the samples. It should also accommodate the growth in word length due to the accumulation of samples. The ROMs are basically look-up tables and therefore operate irrespective of the channel number. The precomputed data stored in the ROMs are output every sample clockcycle. For the MTRM, the 800 different samples of all of the channels are stored and shifted successively to the following RAMs. The computations start once the samples in all of the three RAMs are available. The timing information is stored in 800 locations of the updating RAM. The MDRM also stores the output data in the 800 different locations of the RAM once the computations are performed. As can be observed, the correct storage locations, accessing locations, and the instants of accessing are crucial for the accurate operation of the system.

Table 1 Power consumption of the MCD

	Storage, W	Computational, W
Filter bank	7.0	5.0
FFT module	10.0	6.0
RAM buffer	6.0	—
MCRM	1.0	0.002
MTRM	1.4	0.54
MDRM	0.2	1.02
Total	25.6	12.562

This crucial role is performed by the AG for samples and AG for channels. The AG for samples counts and stores the sample values of each of the 800 channels in the 800 different locations. The AG for channels is used with an appropriate delay to enable or disable the read/write access for several RAMs used in this design. The incoming samples used in the MCRM also need to be stored in a buffer to be operated on later. This will need a RAM buffer design similar to the BRs of the MTRM.

The demodulator is flexible in its need for parameters and is therefore programmable according to the user specifications. The synchronized operation of the three modules provides an output at every clockcycle. This governs the maximum number of channels that can be demodulated. Also, for a larger number of channels, multiple demodulators can be easily operated in parallel. The flexibility in altering the system clock will enable a higher throughput if the storage is made available in the RAMs and the operational units can operate at that rate.

IV. Consumption Estimate

The power-consumption estimate for the multicarrier demodulator presented above is based on a literature survey of currently available memory and logic units.[18-20] The speed requirement of these units is well within the 27.7 ns required for accurate operation of the modules in the proposed MCD. The entire architecture is highly modular and use of application-specific integrated circuits (ASICs) could further reduce the power consumption. The power consumption of each of the modules and the total estimated power is given in Table 1.

V. Conclusions

In this paper, parallel, pipeline, and time-sharing concepts are used in the design of a digital MCD. The MCD consists of the TMUX and the demodulator. The TMUX is implemented by means of a shared filter bank module and a pipelined FFT module. An AE is designed to implement the FFT butterfly operation. The demodulator consists of carrier, timing, and data recovery modules, which have been configured for high-speed demodulation. The shared filter bank and the demodulator process the channels in a time multiplexed manner. This provides considerable hardware and power savings as opposed to having a bank of filters and demodulators. Thus, this system is suitable for onboard processing of the channels in a satellite. It is also modular and lends itself to VLSI implementation. The MCD design is illustrated for the specific case of processing 800 FDMA channels at 64 kb/s each.

Acknowledgments

This research is partly supported by NASA Grants NAG3-799 and NAG3-865 and is part of the M.S. thesis requirements of P. J. Fernandes and L. P. Eugene.

References

[1]Evans, B. G., El Amin, M. H., Casewell, I. E., and Craig, A. D., "An OBP Satellite Payload for European Mobile Communication," *Proceedings of the International Conference on Digital Satellite Communication—7*, Munich, Germany, 1986, pp. 545–550.

[2]Evans, B. G., Coakley, F. P., El Amin, M. H., Lu, S. C., and Wong, C. H., "Baseband Switches and Transmultiplexers for Use in OBP Mobile/Business Satellite System," *Proceedings of the International Conference on Digital Satellite Communication—7*, Munich, Germany, 1986, pp. 587–592.

[3]Nuspl, P. P., and Peters, R., "OBP for Communication Satellite Systems," *Proceedings of the International Conference on Digital Satellite Communication—7*, Munich, Germany, 1986, pp. 137–148.

[4]Perrotta, G., and Losquadro, G., "FDMA/TDM Satellite Communication Systems for Domestic/Business Services," *Proceedings of the International Conference on Digital Satellite Communication—7*, Munich, Germany, 1986, pp. 155–162.

[5]Evans, B. G., and Coakley, F. G., "Multicarrier Demodulators for OBP Satellites," *International Journal of Satellite Communication*, Vol. 6, Jan.–March 1988, pp. 243–251.

[6]Ananasso, F., and Del Re, E., "Clock and Carrier Synchronization in FDMA/TDM User-Oriented Satellite Systems," *IEEE International Conference on Communications,"* Seattle, WA, 1987, pp. 1473–1477.

[7]Ananasso, F., and Saggese, E., "User-Oriented Satellite Systems for the 1990's," *11th AIAA Communication Satellite Systems Conference*, AIAA Paper 86-0601, San Diego, CA, 1986, pp. 1–11.

[8]Ohsawa, T., and Namiki, J., "Digital Group Demodulation System for Multiple PSK Carriers," *11th AIAA Communication Satellite Systems Conference*, AIAA Paper 86-0652, San Diego, CA, 1986, pp. 313–320.

[9]Del Re, E., and Fantacci, R., "Alternatives for Onboard Digital Multicarrier Demodulation," *International Journal of Satellite Communications,* Vol. 6, Jan.–March 1988, pp. 267–281.

[10]Crochiere, R., and Rabiner, L., *Multirate Digital Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1983, Chap. 7.

[11]Bellanger, M., and Daguet, P., "FDM/TDM Transmultiplexers: Digital Polyphase and FFT," *Transactions on Communication,* Vol. 9, 1974, pp. 1199–1205.

[12]Viterbi, A. J., and Viterbi, A. M., "Nonlinear Estimator of PSK Modulated Carrier Phase with Application to Burst Digital Transmission," *IEEE Transactions on Information Theory*, Vol. IT-29, July 1983, pp. 543–551.

[13]Gardner, F. M., "A BPSK/QPSK Timing Error Detector for Sampled Receivers," *IEEE Transactions on Communication,* Vol. COM-34, May 1986, pp. 423–429.

[14]Rabiner, L. R., and Gold, B., *Theory and Application of Digital Signal Processing,* Prentice Hall, Englewood Cliffs, NJ, 1975, Chap. 3.

[15] Yamashita, K., Kanasuuii, A., Hijiya, S., Goto, G., Matsumura, N., and Shirato, T., "Wafer Scale 170,000 Gate FFT Processor," *IEEE Journal of Solid State Circuits,* Vol. 23, April 1988, pp. 335–340.

[16]Owen, R. B., "A 15 ns Complex Multiplier Accumulator for FFT's," *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing,* Dallas, TX, 1987, pp. 527–530.

[17]Stone, H. S., "Parallel Processing with the Perfect Shuffle," *IEEE, Transactions on Computers*, Vol. C-20, Feb. 1971, pp. 153–161.

[18]Yano, K., Yamanaka, T., Nishida, T., Saito, M., Shimohigashi, K., and Shimizu, A., "A 3.8-ns CMOS 16×16-b Multiplier Using Complementary Pass-Transistor and Logic," *IEEE Journal of Solid-State Circuits,* Vol. SC-25, April 1990, pp. 388–394.

[19]Gabillard, B., Ducourant, T., Rocher, C., Prost, M., and Maluenda, J., "A 200-mW GaAs 1-k SRAM with 2-ns Cycle Time," *IEEE Journal of Solid-State Circuits*, Vol. SC-22, Oct. 1987, pp. 693–698.

[20]Chu, S. T., Dikken, J., Hartgring, C. D., List, F. J., Raemakers, J. G., Bell, S. A., Walsh, B., and Salters, R. H. W., "A 25-ns Low Power Full-CMOS 1-Mbit (128K × 8) SRAM," *IEEE Journal of Solid-State Circuits*, Vol. SC-23, Oct. 1988, pp. 1078–1084.

Paul F. Mizera
Associate Editor