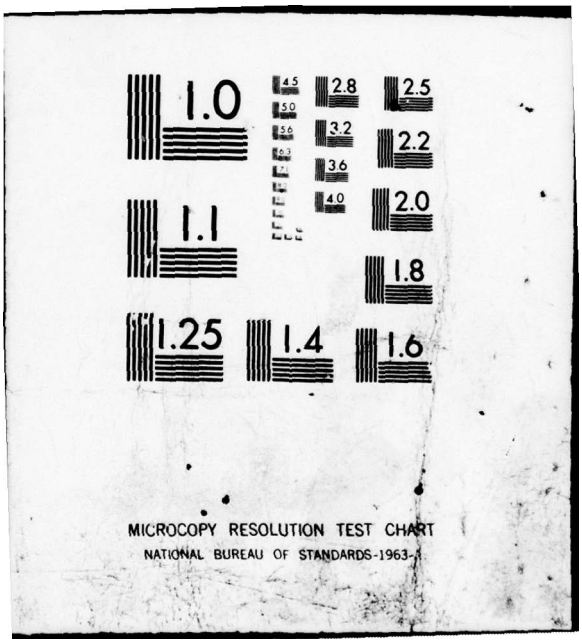
AD-A072 872 ILLINOIS UNIV AT URBANA-CHAMPAIGN DEPT OF COMPUTER SCIENCE F/G 12/1
BURST PROCESSING.(U)
FEB 79 W J POPPELBAUM N00014-75-C-0982
UNCLASSIFIED UIUCDCS-R-79-957 NL

1 OF 1
AD A072872

END
DATE FILMED
9-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

UIUCDCS-R-79-957 UILU-ENG 79 1702

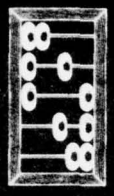
Final Report for the Navy for Contract N000014-75-C-0982

BURST PROCESSING

by

W. J. Poppelbaum
Principal Investigator

February 1979



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

(9) Final Report to the Navy for Contract (15) N000014-75-C-0982

(6) BURST PROCESSING

by

(10) W. J. Poppelbaum
Principal Investigator

(11) Feb 79

(14) UIUCDCS-R-79-957, UILU-ENG-79-1702

(12) 32 p.

0. ABSTRACT

To multiply two numbers with 7 bit (i.e. 1%) accuracy a 2000 transistor microprocessor needs about 10 clock-periods. A stochastic computer gives the same accuracy (within one standard deviation) in 10^4 clock periods--but it only uses a 2 transistor AND for the operation. For the last 4 years the Information Engineering Laboratory in the Department of Computer Science at the University of Illinois has examined a "compromise" called BURST PROCESSING, which uses deterministic methods (like a microprocessor), but averages (like a stochastic machine). The fundamental idea is to perform 1-decimal digit arithmetic and to obtain accuracy by averaging over various inputs. For example 3.4 would be treated as 4444333333 and 6.2 as 7766666666. Using the insight that the average of a sum is the sum of the averages, we can use a simple adder which forms (4 + 7), etc., the average being 9.6 = 3.4 + 6.2!

The actual representation uses a unary PCM frame, e.g. 7 corresponds to a "burst" consisting of the (first) seven slots of a 10-slot frame being filled. This has the advantage that, as long as the number represented does not change, it is immaterial which 10 adjacent slots ("window") we look at: A window contains always the same information. Should the information change (e.g. from 2 to 8 occupied slots), the contents of the window will increase steadily from the initial to the final value. In practice it is useful to introduce so called block-

sum registers (BSR's) which look at 10 adjacent slots and put out a voltage proportional to the number of "ones" in the BSR.

Besides it's simplicity, burst processing has some additional advantages: There are no correlation difficulties as in stochastics, nor do we usually have to staticize the information as in a microprocessor. Also the burst format is only $2\frac{1}{2}$ times less efficient in bandwidth requirements than weighted binary: It can be used directly for audio and video transmissions. The lost bandwidth is compensated for by good noise-immunity under very high noise conditions (i.e. as much as 10% error rate!). Lastly the accuracy of burst processing increases linearly with the number of slots, as contrasted with the square-root law of stochastics.

This report describes both a quasi-analog and a purely digital fashion of performing arithmetic using bursts. On the way the use of non-compacted bursts and the corresponding encoders is brought up. Then the general area of digital filters, realized in burst technology, is discussed and practical examples from the area of convolution, adaptive filtering and RF tuning and demodulation are given. Finally applications to picture processing and "spatial" versions of burst processing are mentioned and suggestions made for future areas of study.

1. INTRODUCTION

The field of computation and communication--initially purely analog--has been almost entirely taken over by weighted binary (WB) digital techniques, simply because the circuit parameter tolerances are so much greater when one makes a binary decision on each pulse; also the binary representation is relatively efficient. The advent of micro-computers and packet switching only reinforced our belief in these binary digital implementations and it may seem preposterous to even suggest alternate, albeit digital, technologies. What we mean by "alternate" is "non weighted binary" (NWB). Historically NWB systems have played an important role: The earliest digital computers were digital differential analyzers (DDA's) in which one simply counted the number of pulses (in a given time or in a given number of time slots). A DDA uses therefore a unary number representations: Each pulse is a marker and one counts markers. Sometimes, of course, the count will have to be displayed--one then uses a binary, octal, decimal or hexadecimal representation. When the pulses are used in a control system (actuators for rudders, antennas etc.) it is usually not necessary to specifically refer to a number system.

More recently considerable success has been obtained by stochastic or probabilistic unary encoding [(-6)(-5)(-4)(-3)(-2)(-1)(0)]. Here a number is represented by the probability of appearance of a pulse in a given time slot. The ultra-simplicity of the computational circuits in this stochastic method is well know, as is the failsoft behavior: A pulse more or less does not matter; Figure 1 shows a common way of multiplying by an AND and adding by two AND's and an OR. The crux of the method is to assess the probabilities by measuring the average frequency (the probability being the limit for an infinite number of time-slot). Unfortunately [(3)] the number of time-slots

(a)

(b)

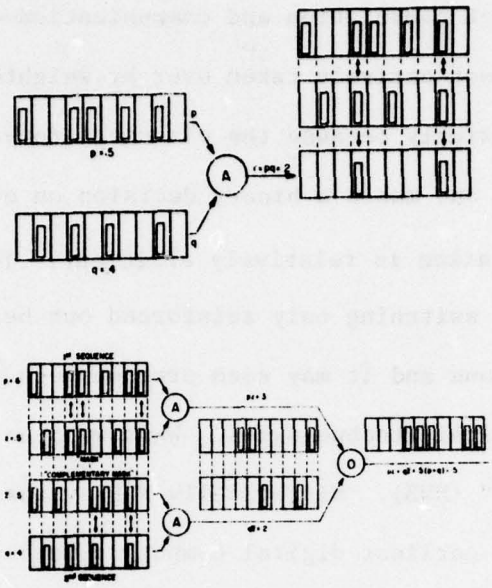


Figure 1

Stochastic Multiplication and Addition

to obtain 10% midrange-precision (with 68% likelihood) is 10^2, for 1% we need 10^4, for 0.1%, 10^6, i.e. the precision increases only as the square root of the number of time slots. Furthermore one must make quite sure that the random pulse sequences (RPS's) which represent 2 variables (to be processed) are uncorrelated. In practice there are many applications (e.g., stochastic feedback controls) in which 1% accuracy is quite sufficient. With a 10MHz clock this means a computation time of 10^{-3}s, an often acceptable value. Note also that negative numbers can be represented by mapping the interval -1,+1 onto the probabilities from 0 to +1. Lately transducers have been built which furnish directly RPS's [7].

The method described in this paper, known as BURST PROCESSING (BP)[3] is the result of attempting to extend averaging techniques to a deterministic unary number representation, making sure that the complexity of the circuitry would be quite a bit lower than in WB representation, yet

perhaps not as low as in stochastic methods. Two versions are discussed. In one use is made of "quantized analog" circuits, better known as "multistate logic" [2][23]. Perhaps the time has come to take such implementations more seriously: The recent success of linear integrated circuits lets one hope that 8- or 10-state devices will become common. But burst processing is by no means predicated on multistate devices; one can very successfully use binary logic (in NWB form!)[9][10][15] for all computational cirucits--this second approach is also treated in detail.

Before going over to a description of the fundamental principles of burst processing, it should be clearly statedthat this method is not a panacea. As a matter of fact it is somewhat wasteful of bandwidth: In the 10-slot version we could represent the integers 0 though 10 by 4 WB digits! But the advantages of BP may well make this sacrifice acceptable. Among the advantages are:

1. Precision increases linearly with the number of time-slots. No correlation difficulties.
2. Simple computational circuitry (typically less than 10% of WB systems).
3. Possibility of on-line processing without staticizing the information: No synchronization is necessary.
4. Great error tolerance because of averaging.
5. Availability of a first appoximation at all times.
6. Novel techniques for comparing, sorting, maximum-operations etc., leading to low cost equipment.
7. Novel techniques for digital filters.
8. RC-element emulation without capacitors, an important point for integrated versions.

It should be said that BP is only efficient if we can use crude approximations for our averaging process. Happily enough (see Section 4) such crude approximations are perfectly acceptable in the transmission of audio and video signals, in control-system computers (in which the output is averaged), in ranging and windowing equipment (sonar, radar) and, most importantly of all, in sampling devices for purely digital radios. As shown below, very encouraging results have been obtained in hardware realizations in all of these areas.

2. THE FUNDAMENTAL PRINCIPLES OF BURST PROCESSING AND ENCODING

The idea of burst processing (see Figure 2) is to perform very low precision arithmetic (typically on single decimal digits) and to use appropriate averaging procedures to obtain higher accuracy [2][3]. Figure 2 shows for instance how one can add 3.4 and 4.2 by decomposing 3.4 into a sequence of four 4's and six 3's, while 4.2 is decomposed into a sequence of two 5's and eight 4's. A one (decimal-) digit adder then gives--as successive sums--two 9's, two 8's and six 7's. The average of the latter is clearly 7.6, i.e. the sum of 3.4 and 4.2! The fundamental problem is obviously to produce automatically, in some circuit, the required integer sequences; we shall show below how this can be done.

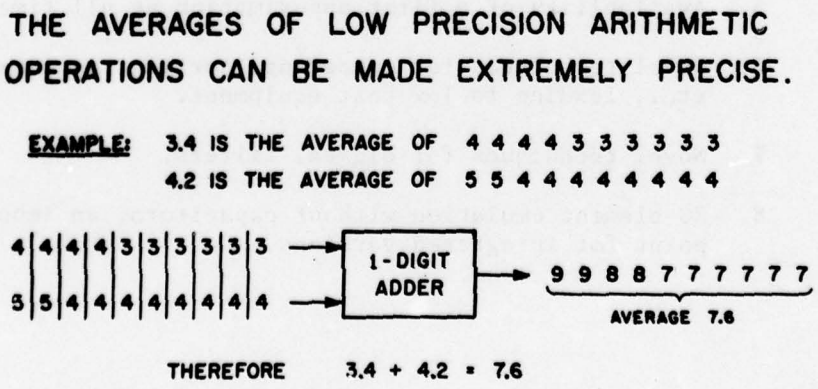


Figure 2

Fundamental Idea of Burst Processing

A possible format for representing the integers from 0 to 10 is a 10-slot frame (called a "block") with the integer n corresponding to n adjacent pulses at the beginning of the frame: This will be called a "compacted burst". Instead of using 10 slots one can, of course, also use m-slots per block; systems have been built with $4 \leq m \leq 16$. If we do not mention the value of m explicitly, it will be assumed that m=10. One of the interesting properties of this representation is that a "window" of length m always contains the same number of pulses, (namely n) [3] as long as we continue to represent n--quite independently of whether the beginning of the window coincides with the beginning of a block or not!

The so called Block Sum Register (BSR) shown in Figure 3a exploits the above "windowing property" in that the (quantized analog or m-state, here 10-state) output is proportional to the number of

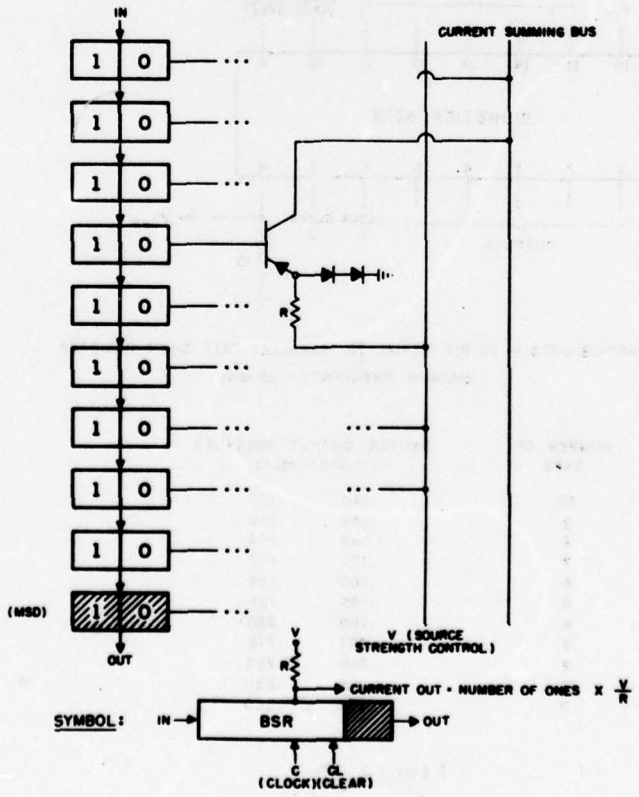
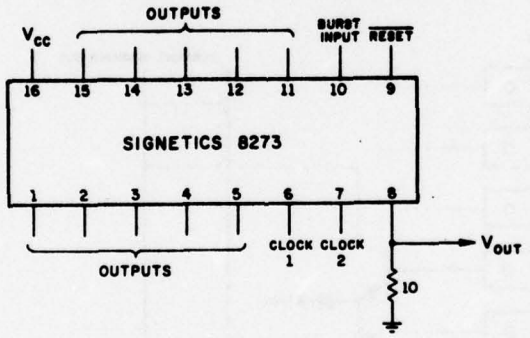


Figure 3a

Block Sum Register

"ones" in the shift register. Note that by adjusting V we can actually multiply this number of "ones" by a predetermined coefficient; this possibility is exploited in building digital filters (see below) using BSR's.

If burst processing were to be adopted on a large scale, it would be trivial to implement a BSR according to Fig. 3a through a semiconductor house, instead of building it from a shift register and current sources. If an accuracy of 2% per step is sufficient, Ma has shown that we can use a sensing resistor (e.g. 10Ω) in the ground return of certain shift registers [23]. The best performance was obtained from a Signetics 8273 as shown in Fig. 3b: Although the variation from chip to chip was from .0085V to .010V per step, for a given chip the linearity was astonishingly good, namely less than 2% variation from step to step!



SIGNETICS 8273 – 10 BIT SERIAL IN PARALLEL OUT SHIFT REGISTER
MINIMUM FREQUENCY – 25 MHz

NUMBER OF ONES	SAMPLE OUTPUT VOLTAGES (±.0002 VOLTS)	
10	.144	.150
9	.153	.159
8	.162	.168
7	.171	.177
6	.180	.186
5	.189	.195
4	.198	.204
3	.207	.213
2	.216	.222
1	.225	.231
0	.234	.240

Figure 3b

Signetics 8273 used as BSR

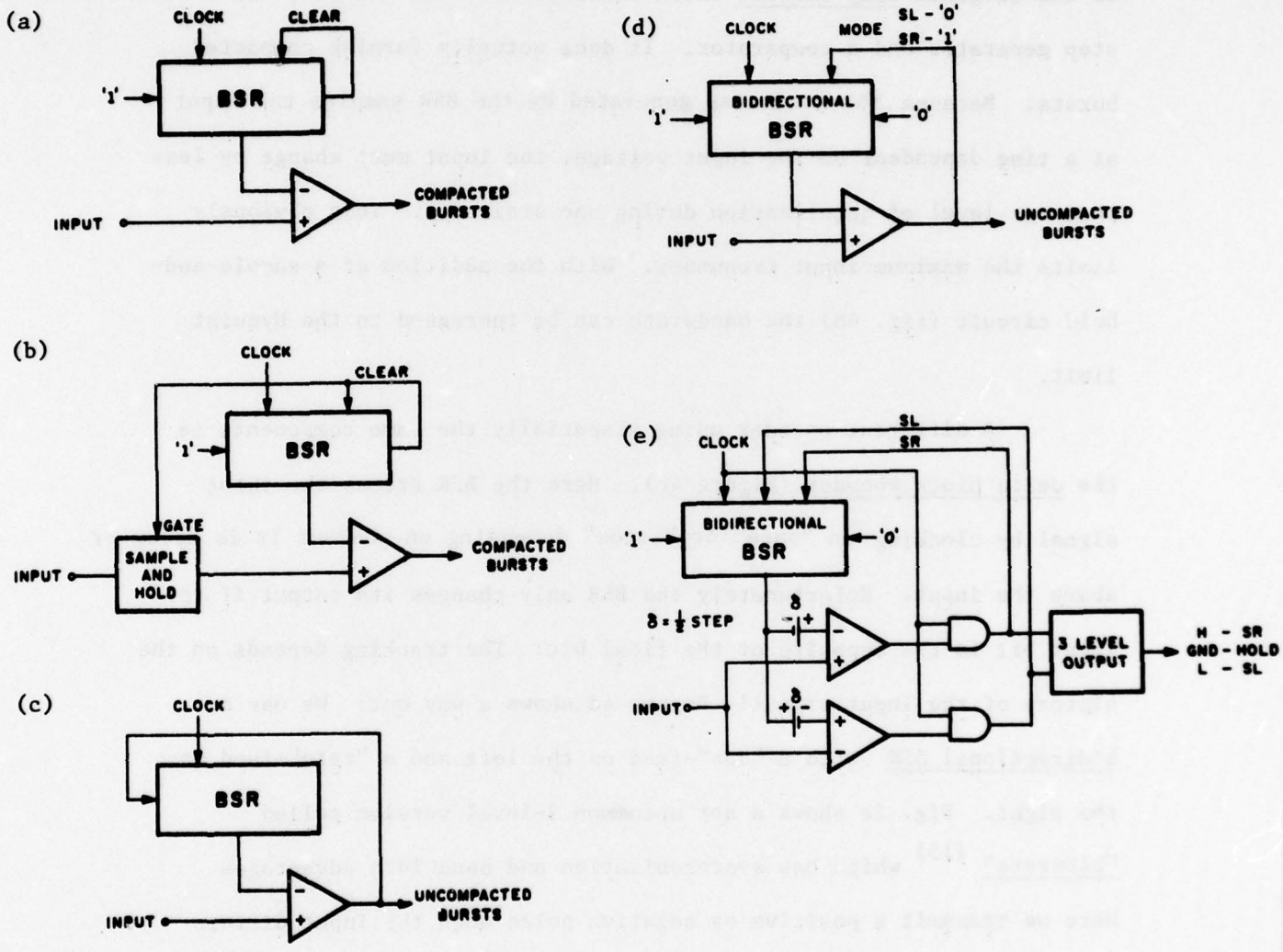


Figure 4

Burst Encoders

It must be mentioned that the compacted burst format, although usually easier to handle, is by no means a necessity. Figure 4 shows a variety of encoders studied by Wolff [15][16]. The first one (Fig. 4a) is the original ramp encoder which consists of a BSR connected as a stair-step generator and a comparator. It does actually furnish compacted bursts. Because the stairstep generated by the BSR samples the input at a time dependent on the input voltage, the input must change by less than one level of quantization during one stairstep. This obviously limits the maximum input frequency. With the addition of a sample-and-hold circuit (Fig. 4b) the bandwidth can be increased to the Nyquist limit.

A different encoder using essentially the same components is the delta block encoder (Figure 4c). Here the BSR tracks the input signal by clocking in "ones" or "zeros" depending on whether it is below or above the input. Unfortunately the BSR only changes its output if the input bit is the opposite of the final bit: The tracking depends on the history of the input signal! Figure 4d shows a way out: We use a bidirectional BSR with a "one"-feed on the left and a "zero"-feed on the right. Fig. 2e shows a not uncommon 3-level version called "biburst" [15] which has synchronization and bandwidth advantages. Here we transmit a positive or negative pulse when the input differs from the BSR value by more than half a step. The receiver therefore does not need a clock: It derives it's clock from the edge of the input pulses and simply freezes its value if none are coming in!

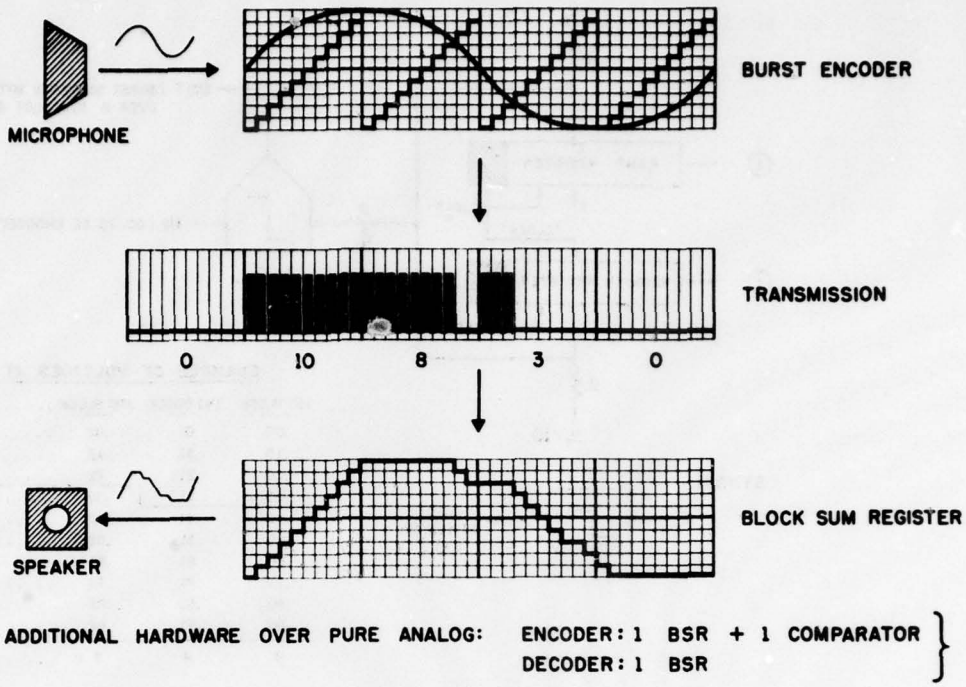
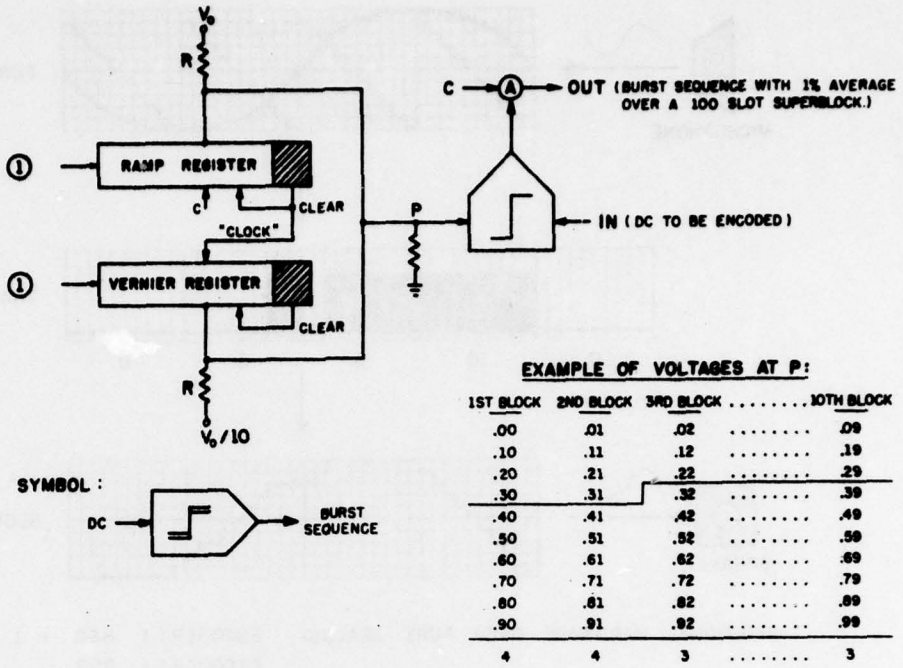


Figure 5

Burst Encoding and Decoding

In Figure 5 we show the action of a ramp-encoder when the input varies and the reconstituted signal is taken from a BSR. It should be noted that a window of length 10, sliding along the transmitted sequences gives, at any moment, the most recent approximation and that it interpolates in so doing. Voice transmission has been shown to be feasible with only 3 slots and a clock-rate of 10KHz.



EXAMPLE OF VOLTAGES AT P:

1ST BLOCK	2ND BLOCK	3RD BLOCK	.....	10TH BLOCK
.00	.01	.02	.....	.09
.10	.11	.12	.....	.19
.20	.21	.22	.....	.29
.30	.31	.32	.....	.39
.40	.41	.42	.....	.49
.50	.51	.52	.....	.59
.60	.61	.62	.....	.69
.70	.71	.72	.....	.79
.80	.81	.82	.....	.89
.90	.91	.92	.....	.99
4	4	3	.....	3

Figure 6

Vernier Encoder

As mentioned before, it is of paramount importance to produce the sequence of bursts which represents a number of higher precision than .1 (i.e. 10 slot system) by an automatic device. A simple circuit is a combination of two BSR's successively filled with "ones", in which one BSR--the "vernier register"--runs at 1/10 the speed of the main BSR, i.e. the "ramp register": Each time the ramp register is cleared (when the "ones" attain the most significant digit), the vernier register adds to the output current of the ramp register one tenth of one step of the former. Normalizing the combined output current (or the voltage at P!) to 1, the table in Figure 6 shows how the slow shift upwards of the stairsteps produces first the longer bursts, then the shorter ones and precisely the requisite number! All that is necessary is to compare the voltage in P with the voltage to be encoded--also normalized to 1--and to switch on the clock pulses by an AND-gate as long as the voltage to be encoded (.32 in

the Figure) is higher than the comparison voltage. This device is called a vernier Encoder [2] [3].

One of the advantages of burst processing is that error-pulses can be reasonably well absorbed because the BSR's show averages. In case we use compacted bursts, one can improve the situation by taking a majority vote for every 3 adjacent slots: This fills in 1-slot gaps and suppresses single pulses. It has been shown that under some noise conditions (10% error rate and above) this system gives results superior to Read-Muller codes.[11]

It should also be noted that in case of compacted bursts we can transmit pulses of varying width constituting the envelope of the bursts: burst processing is quantized pulse-width modulation. On the receiving end we then simply AND the envelope with the sytems clock.

3. ARITHMETIC UNITS FOR BURST PROCESSING

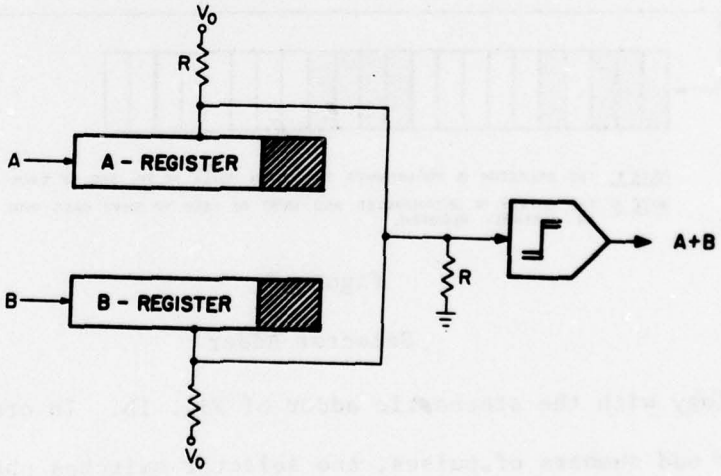
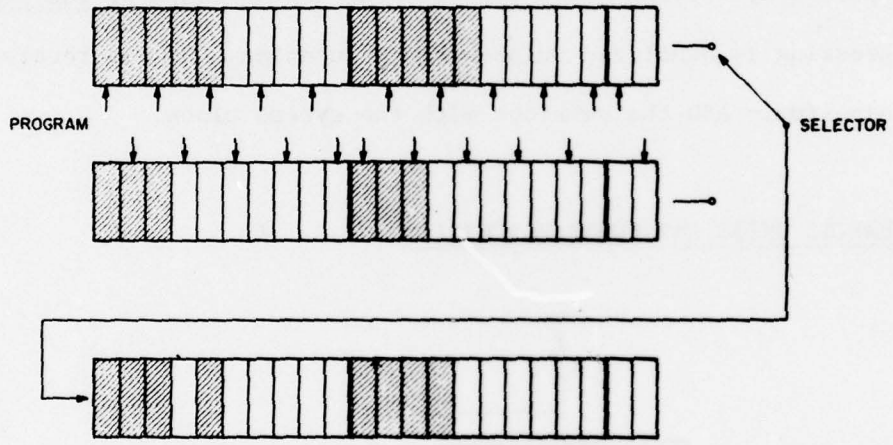


Figure 7

Quasi Analog Burst Adder

Figure 7 shows the most direct way of performing addition (and subtraction when we take the difference of the output of the BSR's) [2][3]: We simply use analog addition of the quantized analog output of BSR's, the result being re-encoded by a ramp encoder or a vernier encoder. Scaling in the quasi analog adder can be obtained by modifying R. The semiconductor houses are perfectly capable of making 8-, 10- or even 16-state elements, but strangely enough most engineers are reticent to use these multistate designs: It is therefore appropriate to consider purely logical implementation using standard binary techniques. Figure 8 shows a so-called selector adder [9] for compacted bursts: Note



NOTE 1: THE SELECTOR IS PROGRAMMED TO SWITCH PHASE AT THE END OF EACH BLOCK.
NOTE 2: THE OUTPUT IS UNCOMPACTED AND MUST BE USED TO SHIFT ONES INTO AN ASSEMBLY REGISTER.

Figure 8

Selector Adder

the analogy with the stochastic adder of Fig. 1b. In order to eliminate any bias for odd numbers of pulses, the selector switches phase at the end of each block. The output is uncompacted--in order to obtain a compact output we shift "ones" into an assembly register whenever a "one" occurs in the bottom sequence.

Logical arithmetic units of many different designs have been built. One of the more sophisticated ones uses the so-called carousel multiplier/divider (CMD) [9][24] shown in Fig. 9. The fundamental idea of

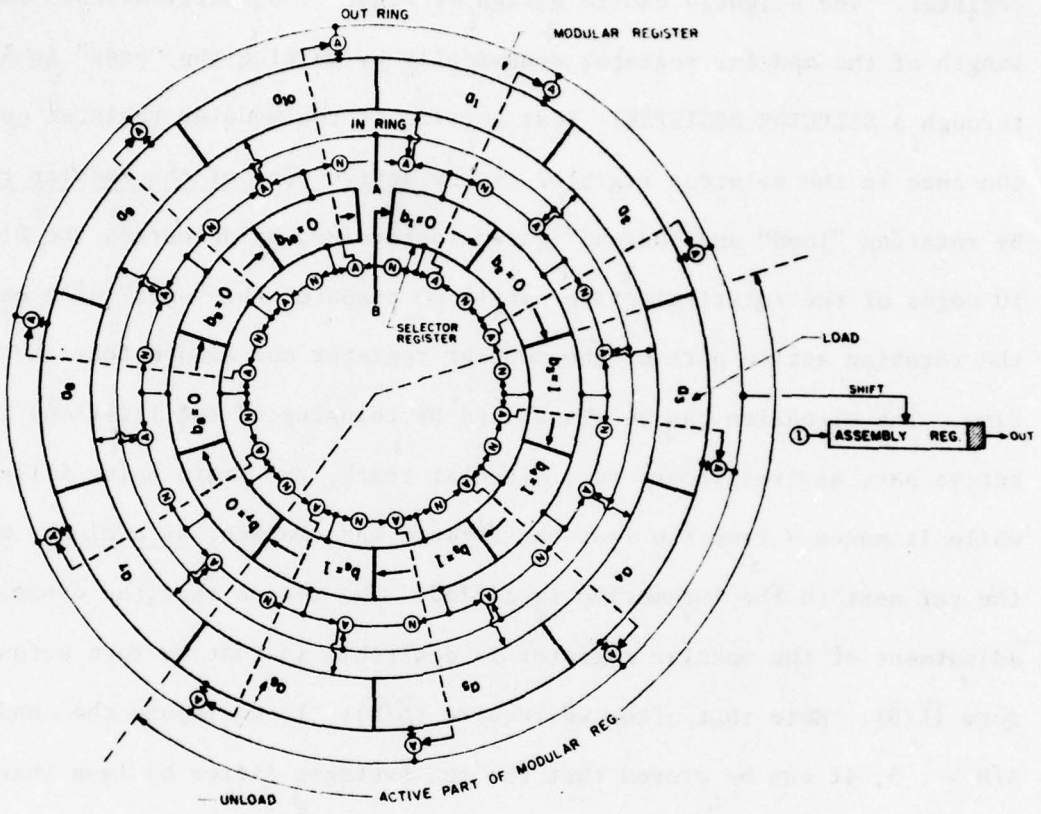


Figure 9

Carousel Multiplier/Divider

the CMD is that both multiplication and division of burst sequences can be thought of as the "filling up" of a register (called MODULAR REGISTER in Figure 9) by a stream of pulses, being careful to note the occurence of overflow. In case of the multiplication of A by B, we make the modular register of length 10 and shift into it as many blocks of B as indicated by A: counting the number of overflows, we then obtain AB/10. Note that if we leave the modular register uncleared at the end, we shall obtain an exact value when we average!

For division A/B we simply make the modular register of length B and note its overflow as we shift in a fixed number (e.g. ten) blocks of A: Of course, only the "ones" in A produce the filling of the modular register. The slightly exotic design of Figure 9 actually adjusts the length of the modular register dynamically by walking the "ones" in B through a SELECTOR REGISTER: That portion of the modular register opposite the ones in the selector register is the <u>active part of the modular register</u>. By rotating "load" and "unload" gates (determined by detecting the 01 and 10 edges of the rotating active part!) we transfer the "ones" of A onto the rotating active part of the modular register and also detect the overflow. The situation can be visualized by thinking of the flipflops in the active part as freightcars on a circular track, the train being filled up - while it moves - from the last car forward and emptied, <u>as a block</u>, whenever the car next to the locomotive is filled. The reason that the dynamic adjustment of the modular register is desirable is that we then actually form $(\overline{A}/\overline{B})$. Note that often we require $(\overline{A}/\overline{B})$: If we impose the condition $A/B \geq .5$, it can be proved that the two averages differ by less than 5 percent in the 10-slot system.

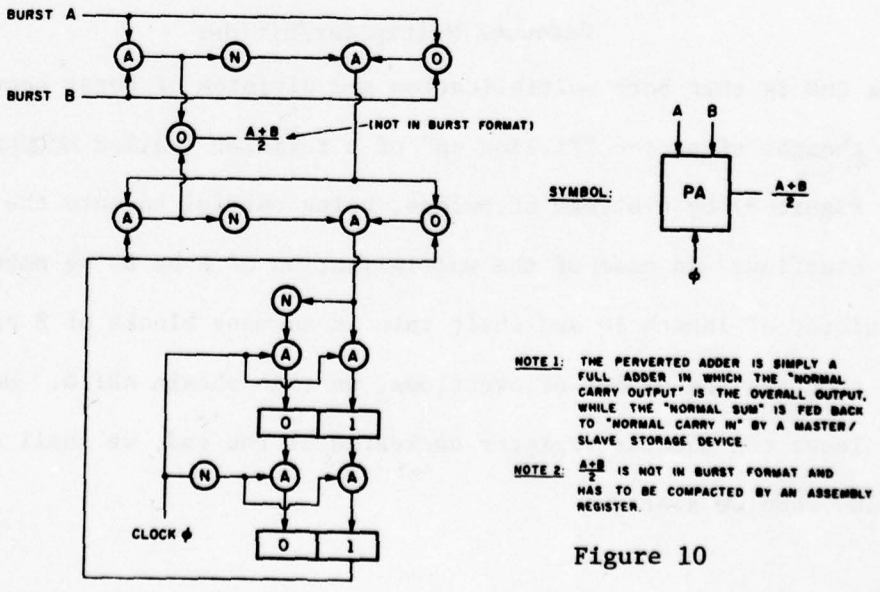


Figure 10

Perverted Adder

It was noted by Taylor and Tietz [11][24] that it is possible to modify a serial full-adder in such a fashion that it furnishes an uncompacted sum of two (uncompacted) bursts with a normalizing factor of 1/2. Figure 10 shows why we call the arrangement a perverted adder (PA): What is normally the carry output of a serial adder gives the (normalized) sum while the sum output in normal operation becomes the input to the carry-delay circuit. The carry delay is in effect a master/slave D-flipflop pair (symbolized by two RS flipflops and appropriate gates), which delays the carry by one clock cycle and then re-inserts it in the next cycle of the adder. Note that contrary to the selector adder, we have several collector delays and that the carry feedback limits the speed. But there are also very attractive features, in particular the fact that we can cascade many PA's if necessary!

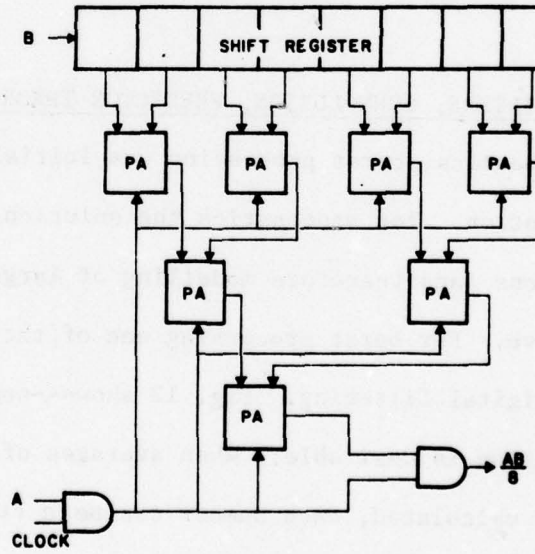


Figure 11

PA Multiplier

The ease with which PA's can be cascaded is used in the PA-Multiplier of Figure 11. Here we use a tree of PA's to (dynamically) add all digits in A whenever B has a "one". It is not hard to see that when we use all "ones" in B, we obtain AB/8 at the output. The figure shows the arrangements for an 8-slot burst format, because this is more efficient than the standard 10-slot format for a tree-structure, but it is, of course, possible to design a 10-slot version!

We have not commented on the complexity of AU's for burst processing: By optimizing the design (LOCOBURST Project) [15][24] we have been able to imitate an 8-bit microprocessor (10MHz clock) with about 10% of the gate count. In these times of ultra-cheap hardware this may not be so impressive, but the critics often forget that a simplification by a factor 10 means also 10 times less power consumption, 10 times less space, and 10 times more reliability!

4. BURST DIGITAL FILTERS, CONVOLUTION, PREDICTOR THEORY and RADIOS.

Like stochastics, burst processing was initially a method in search of an application. For stochastics the solution of systems of differential equations (and therefore modelling of large systems) seems to be most attractive. For burst processing one of the more promising fields is that of digital filtering. Fig. 12 shows--somewhat whimsically--why such an application is desirable: When averages of many (more or less random) numbers are calculated, each number can be a crude approximation, in particular a 8- or 10- slot burst approximation!

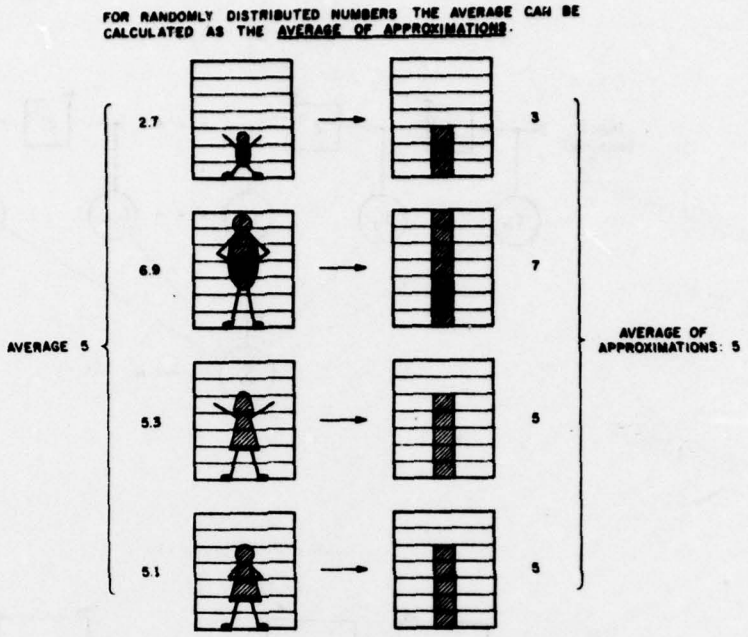
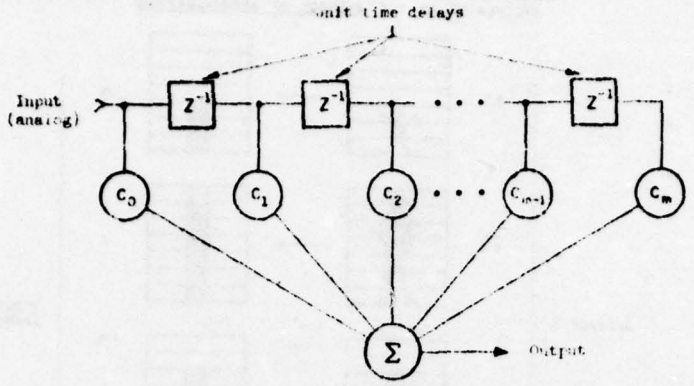


Figure 12

Why Burst Processing Works

We have concentrated mostly on realizing FIR (Finite Impulse Response) filters, i.e. filters in which delayed versions of the signal are multiplied by certain constants and then added [11]: No feedback from output to input is allowed as in IIR (Infinite Impulse Response) filters. The reason for this limitation is that the noise problems in IIR filters are much more severe and the (crude) burst approximations are therefore less attractive. Nonetheless IIR filters have been shown to be viable, for instance in Pitt's PREDICTORBURST project discussed below. The theory of designing FIR digital filters from an analog version is well known and leads to a layout as shown in Fig. 13a. To implement such a filter, we simply replace the circuit delays z^{-1} by n-bit shift registers (typically n=8, 10 or 16) and multiply their burst contents (i.e. the number of "ones") by appropriate constants c_1, c_2, etc. as shown in Fig. 13b.

(a)



(b)

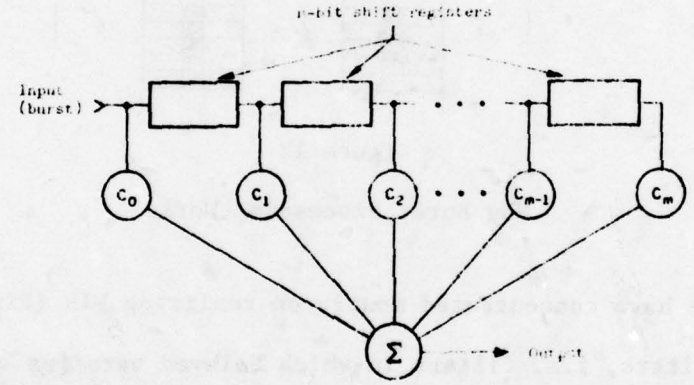


Figure 13

General FIR Filter (a) and Burst Version (b)

As in our arithmetic units there is again a quasi-analog and a purely digital way of realizing Fig. 13b. In the former case c_1, c_2 etc. are simply the return voltages on BSR's and we use analog summation. Fig. 14 shows such an application (developed by Xydes) for an on-line Fourier Transform Unit used in a rather sophisticated frequency analyzer for speech (BURFT) [15] [17]. Fig. 15 shows an adaptive system (PREDICTORBURST) to calculate the predictor coefficients a_i (only stage i=2 is shown explicitly) in view of bandwidth compression of human speech--as is well-known the a_i's vary quite slowly. MDAC is a multiplying digital-to-analog converter

which sends $a_2 \cdot s_{n-2}$ $[+ a_1 s_{n-1} + a_3 s_{n-3} + \ldots]$ to a summing amplifier. The latter forms the error signal. Note the feedback from the error-output to the counters storing the a_i's.

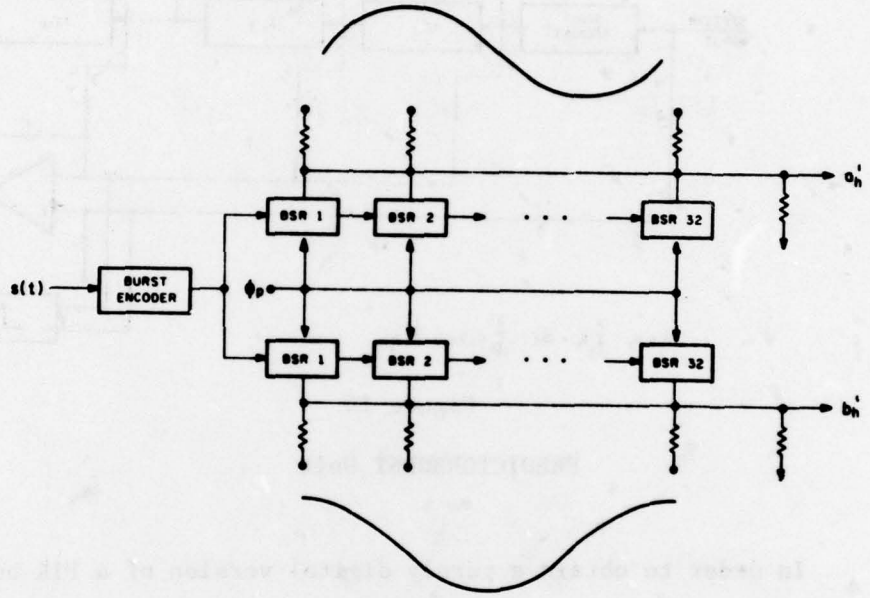


Figure 14

BURFT Fourier Transform Unit

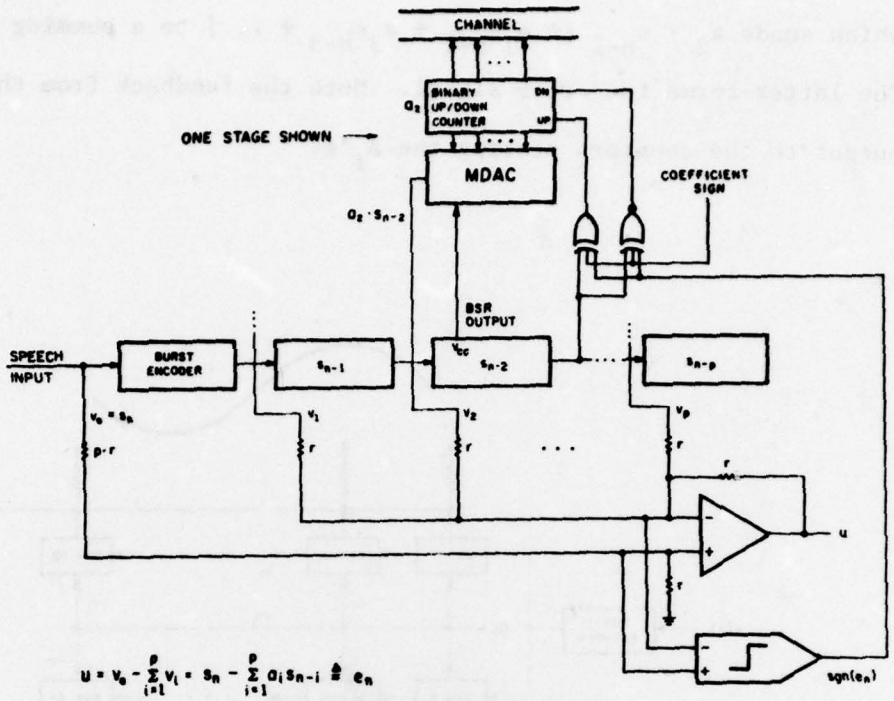


Figure 15

PREDICTORBURST Unit

In order to obtain a purely digital version of a FIR burst filter, we can again revert to a perverted adder (PA) design: Here the coefficients $c_1, c_2 \ldots$ etc. are supposed to be small integers and their influence is realized by appropriate fan-outs inside the PA tree [18]. Fig. 16 shows the practical layout of a Hamming Weighted Filter studied by Wells. Two clock lines (CKS and CKF) are used because the filter is actually multiplexed! In conjunction with some additional burst filters to suppress the periodic peaks due to sampling at 125 KHz, the response is down 39 db at 8 KHz from the center (null-) frequency, and down by better than 70db at 125 KHz. This is comparable to what one obtains in good communication equipment.

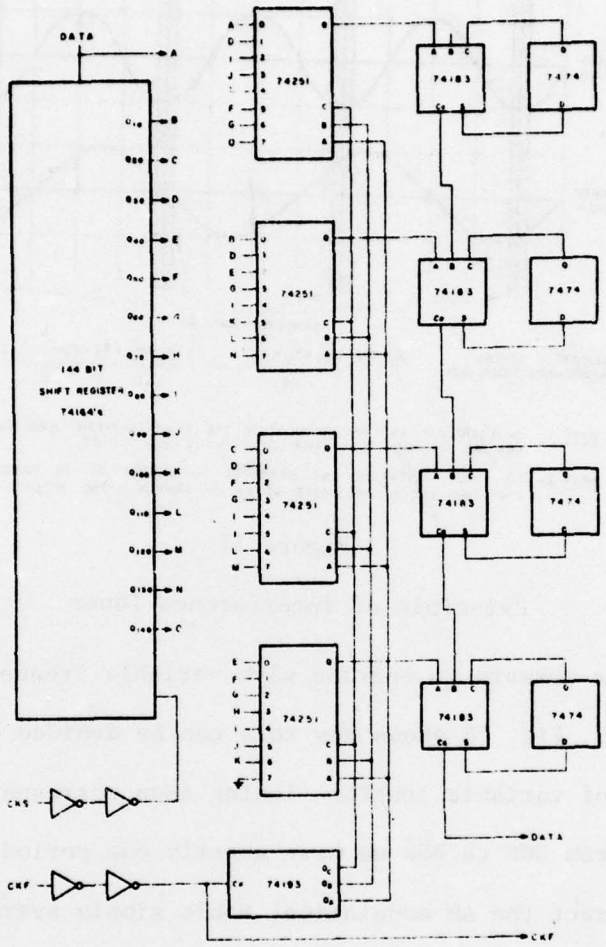


Figure 16

Hamming Weighted Burst Filter

From the outset we became very interested in digital radios and used comb-filters for tuning [5] [9]. The principle is shown (for AM) in Fig. 17. The idea is that if we sample at the frequency of the desired station, we shall obtain constructive interference of the samples for that station, while other stations will be subject to destructive interference. Here it must, of course, be assumed that during the integration over about

100 cycles the amplitude remains essentially constant: This is the case for speech and a carrier above 1 MHz, a condition always met in real life.

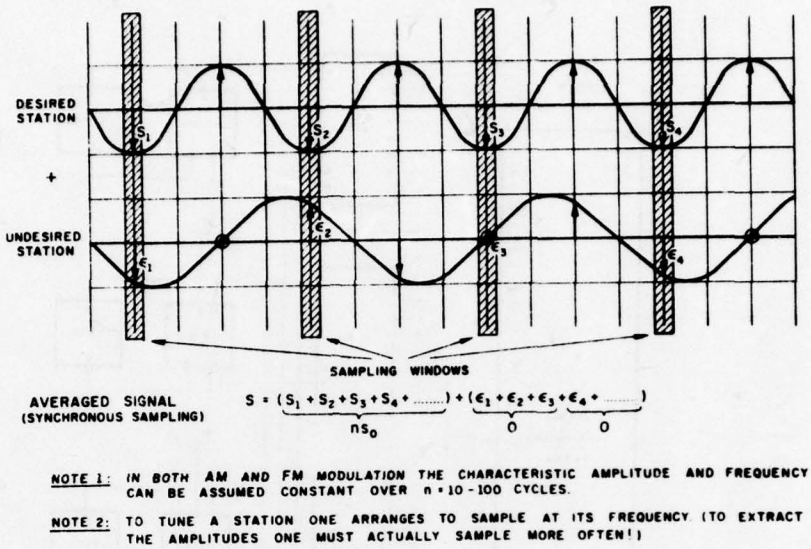


Figure 17

Principle of Interference Tuner

In practice it is awkward to operate with variable frequency clock in order to obtain tuning. Fig. 18 shows how this can be avoided by using (delaying) shift registers of variable length: Tuning then corresponds to adjusting the delays so that from BSR to BSR we have exactly one period of the desired station! To extract the AM modulation, Noble simply averaged the tuned signal, for instance by running it into a counter and watching the overflow.

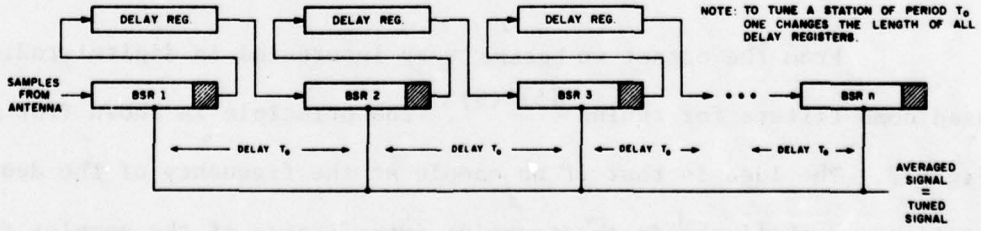
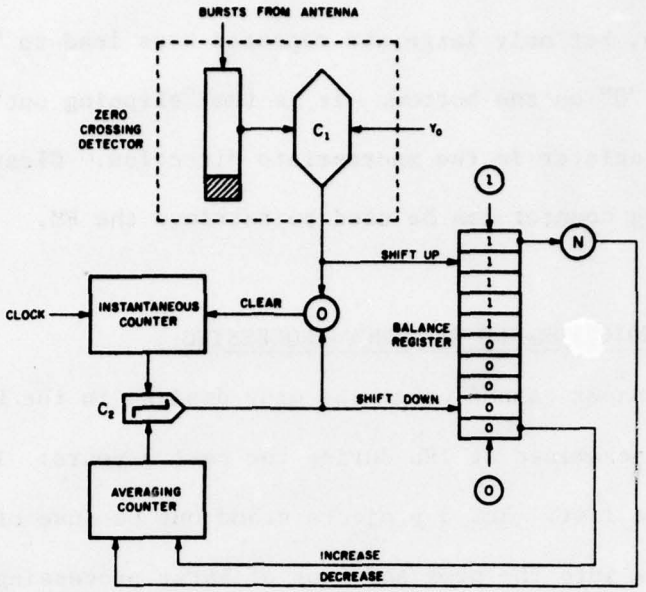


Figure 18

Delay Register Tuner



NOTE 1: IN ESSENCE THE CIRCUIT SIMPLY DETECTS WHETHER THE NEXT ZERO CROSSING OCCURS (OR NOT) BEFORE THE INSTANTANEOUS COUNTER REACHES THE PREVIOUS AVERAGE BETWEEN ZERO CROSSINGS. THE BALANCE REGISTER MODIFIES THE AVERAGE.

NOTE 2: THE WIDTH OF THE WINDOW OF COMPARATOR C_1 CORRESPONDS TO ONE PULSE OF A BURST.

NOTE 3: COMPARATOR C_1 AND (SIMPLIFIED) COMPARATOR C_2 CAN BE REPLACED BY DIGITAL DESIGNS.

Figure 19

PLL for FM Demodulation

To demodulate FM we must determine the average frequency of the (tuned) antenna signal. This can be done by building a burst-implemented phase-locked-loop (PLL) and watching the average time between zero crossings of its output [9][18][20].

The PLL-design in Figure 19 studied by Robinson and Mohan starts, on top, with a zero-crossing detector. Although we indicate a BSR and a comparator in this position, it is clear that we can use a purely digital design. The game is now to count clock-cyles between zero-crossings in an "instantaneous counter". If the count is high, we increase the averaging counter; if it is low, we decrease it. Of course, such a feedback system must have some inertia built into it. This is done by delaying the increase/decrease decision in the "balance register", which has its 1/0 boundary

hop up and down, but only large and repeated hops lead to "1" slipping out on top, or "0" on the bottom. It is the "slipping out" which modifies the averaging register in the appropriate direction. Clearly the contents of the averaging counter can be used to retrieve the FM.

5. PICTURE, INDICATOR-AND SEQUENCY-PROCESSING

This paper cannot cover the many designs in the burst area which have been examined at IEL during the past 3 years: The bibliography gives a complete list. But 3 projects stand out because of the insight they afforded us into the pros and cons of burst processing and also into future generalizations.

In Bracha's WALSHSTORE [21] project we examined the feasibility of storing the "electronic hologram" of a binary input picture in such a way that the destruction of a large percentage of the memory would simply correspond to a lack of definitions of the reconstituted input. This objective was attained by using a two-dimensional Walsh-Transform. Burst processing was used (in conjunction with weighted binary techniques) because the input came from a CCD-type camera which essentially shifted out bursts. But the storage of the transform, (obtained by burst processing) was done in binary: The storage of a number in non-weighted unary fashion would be prohibitively expensive.

In Gostin's INDIRAD [15][25] project we used a spatial version of bursts: In a set of 7 wires we energize just one to indicate the numerical value of waveform-samples taken at times in quadrature, i.e. for sin t = $\pm$ 1 and cos t = $\pm$ 1. The advantage of such an Indicator Representation is that very fast arithmetic becomes trivial: All we have to do is to realize a table-look-up scheme (for sum, product etc.) in hardware. The averaging at the end then smoothes the results. Fig. 20 shows the block-diagram of INDIRAD.

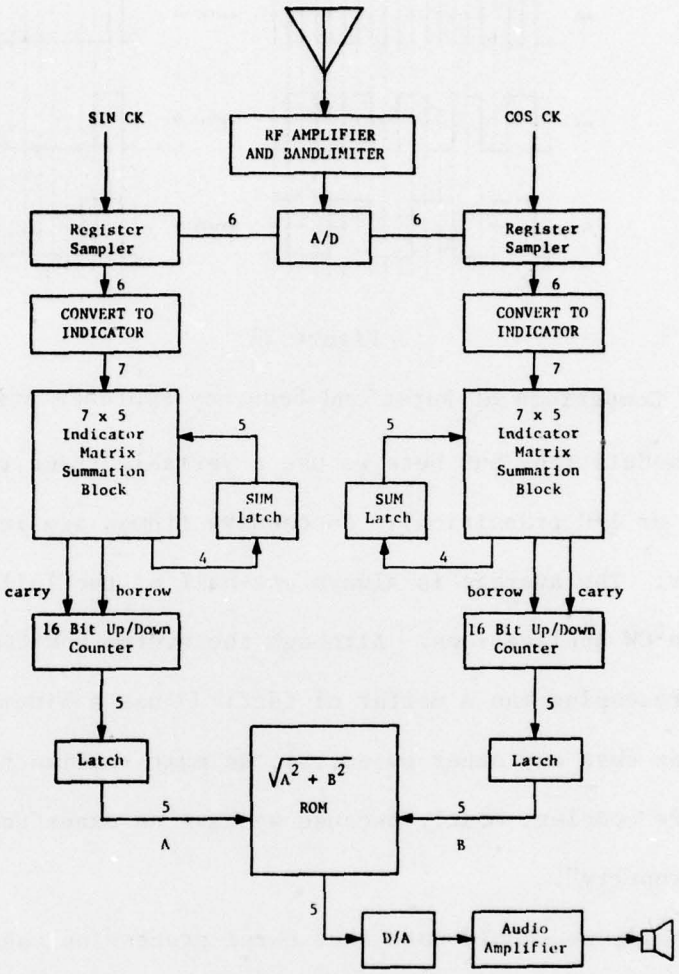


Figure 20

Block Diagram of INDIRAD

We have already mentioned that if a 10-slot burst system is replaced by weighted binary, we gain a factor of 2 1/2 in bandwidth. Sequency Processing [19] tries to gain bandwidth for the average case (→ 5 pulses in 10 slots) by abandoning the fixed frame format--this gives a factor 2 on the average, making it nearly the equivalent in bandwidth of weighted binary. The actual format is shown in Fig. 21 . Again we have a quantized

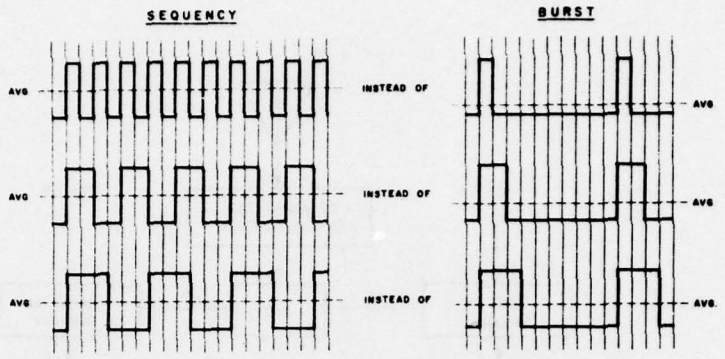


Figure 21

Comparison of Burst and Sequency Representation

pulse-width modulation, but here we use a variable frame reference (namely the last 0→1 or 1→0 transition). Successive frames are inverted with respect to each other: The average is always one-half of the full swing--this has advantages in CW applications. Although the system updatesmore frequently than burst processing (as a matter of fact: It has a binary version which updates faster than any other sytem...), we must realize that arithmetic units are more complex, mostly because we have no exact equivalent of the "windowing property".

Finally we should note that burst processing uses only a small number of standard elements (BSR's, comparators, counters, ANDs, ORs, and NOTs) and that designing with it is rather easy. In case relatively untrained personnel has to be used for design and maintenance, this "modular kit" feature could be very attractive!

6. BIBLIOGRAPHY

-6 Petrovic, Siljak: "Multiplication by means of Coincidence", ACTES Proc. of 3rd Intern, Analog Comp. Meeting, Summer 1962.

-5 Von Neumann: "Probabilistic logics and the synthesis of reliable organisms from unreliable components", Collected Works Vol. 6 Macmillan 1963.

-9 Afuso: Quarterly Technical Progress Reports of the DCS at the UofI, starting in January 1964.

-3 Ribeiro: "Comments on pulsed data hybrid computers", IEEE Transactions on Computers, Vol. EC-13 Oct. 1964.

-2 Gaines: "Stochastic Computing" AFIPS Proc. SJCC Vol. 30, Summer 1967.

-1 Poppelbaum, Afuso, Esch: "Stochastic Computing Elements and Systems" Proc. Fall JCC 1967.

0. Poppelbaum: "Computer Hardware Theory", Macmillan, New York 1972.

1. Poppelbaum: "A Practical Program in Stochastic Processing", Proposal to ONR for 73/74 June 1973.

2. Poppelbaum: Appendix I to above: "Novel PCM Transmission and Processing system called BURST PROCESSING", Proposal to ONR for 74/75 March 1974.

3. Poppelbaum: "Statistical Processors", DCS Report, May 1974. Also appeared in "Advances in Computers", Vol. 14, 1976.

4. Tse: "BURP-A Bundle Repeater and Restorer", MS-Thesis, DCS Report 689, Dec. 1974.

5. Mohan, Bracha, Liu: "Performance Evaluation of the Digital AM Receiver", DCS Report 757, April 1975.

6. Huberts: "NORMAN", MS-Thesis, DCS Report 715, April 1975.

7. Cutler: "Molecular Stochastics: A Study of Direct Production of Stochastic Sequences from Transducers", Ph.D. Thesis, DCS Report 723, April 1975.

8. Cutler, Ficke: "A Stochastic Control System", DCS Report 752, June 1975.

9. Poppelbaum: "Application of Stochastic and Burst Processing to Communication and Computing Systems", Proposal to ONR for 75/76, July 1975.

10. Bracha: "BURSTCALC", MS-Thesis, DCS Report 769, Oct. 1975.

11. Taylor: "An Analysis of Burst Encoding Methods and Transmission Properties", MS-Thesis, DCS Report 770, Dec., 1975.

12. Holberger: "An Addressable Ring Conveyor", MS-Thesis, DCS Report, 778 Jan., 1976.

13. Mohan: "The Application of Burst Processing to Digital FM Receivers", MS-Thesis, DCS Report 780, January 1976.

14. Pleva: "A Microprocessor - Controlled Interface for Burst Processing", MS-Thesis, DCS Report 812, July 1976.

15. Poppelbaum: "Application of Stochastic and Burst Processing to Communication and Computing Systems," Proposal to ONR for 76/77. July 1976.

16. Wolff: "Transmission of Analog Signals Using Burst Techniques", MS-Thesis, DCS Report 838, Jan. 1977.

17. Xydes: "Application of Burst Processing to the Spectral Decomposition of Speech" MS-Thesis, DCS Report 870, July 1977.

18. Wells: "Digital Filtering Using Burst Processing Techniques", MS-Thesis DCS Report 871, July 1977.

19. Poppelbaum: "Application of Stochastic and Burst Processing to Communication and Computing Systems," Proposal to ONR for 77/78, July 1977.

20. Robinson: "BURSTLOCK: A Digital Phase-Locked Loop Using Burst Techniques", MS-Thesis, DCS Report 872, May 1977.

21. Bracha: "WALSHSTORE: The Application of Burst Processing to Fail-Soft Storage Systems Using Walsh Transforms", Ph.D.-Thesis, DCS Report 878, Oct. 1977.

22. Pitt, Poppelbaum, Xydes: "OPTOBUNDLE - A Unique Fiber Optic Multiplier", DCS Report 882, June 1977.

23. Ma: "Block Sum Register and Burst Arithmetic", MS-Thesis, DCS Report 890, Aug. 1977.

24. Tietz: "BURSTLOGIC: Design and Analysis of Logic Circuitry to Perform Arithmetic on Data in Burst Format", Ph.D.-Thesis, DCS Report 895, May 1977.

25. Gostin: "INDIRAD: A Digital Receiver for Amplitude Modulated Signals Using Indicator Processing", MS-Thesis, DCS Report 923, June 1978.

Additional information may be obtained from the Quarterly Progress Report of the IEL and the Annual Reports of the Department of Computer Science of the University of Illinois.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-79-957	2.	3. Recipient's Accession No.
4. Title and Subtitle Final Report for the Navy for Contract N000014-75-C-0982 BURST PROCESSING			5. Report Date February 1979 6.
7. Author(s) W. J. Poppelbaum, Principal Investigator			8. Performing Organization Rept. No. UIUCDCS-R-79-957
9. Performing Organization Name and Address Department of Computer Science University of Illinois Urbana, IL 61801			10. Project/Task/Work Unit No. 11. Contract/Grant No.
12. Sponsoring Organization Name and Address Same as above.			13. Type of Report & Period Covered Final Report 14.

15. Supplementary Notes

16. Abstracts

To multiply two numbers with 7 bit (i.e. 1%) accuracy a 2000 transistor microprocessor needs about 10 clock-periods. A stochastic computer gives the same accuracy (within one standard deviation) in 10^4 clock periods--but it only uses a 2 transistor AND for the operation. For the last 4 years the Information Engineering Laboratory in the Department of Computer Science at the Unives-ity of Illinois has examined a "compromise" called BURST PROCESSING, which uses deterministic methods (like a microprocessor), but averages (like a stochastic machine). The fundamental idea is to perform 1-decimal digit arithmetic and to obtain accuracy by averaging over various inputs. For example 3.4 would be treated as 4444333333 and 6.2 as 7766666666. Using the insight that the average of a sum is the sum of the averages, we can use a simple adder which forms (4 + 7), etc., the average being 9.6 = 3.4 + 6.2!

17. Key Words and Document Analysis. 17a. Descriptors

Unary process
Burst Representative
Digital radios
Transverseful filter
Predictor theory

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 30
	20. Security Class (This Page UNCLASSIFIED	22. Price

FORM NTIS-35 (10-70) USCOMM-DC 40329-P71