SINGLE-CHIP ARCHITECTURE FOR REAL-TIME COMPUTATION OF THE WIGNER DISTRIBUTION OF ACOUSTIC SIGNALS

V. G. Oklobdzija, N. M. Marinovic*, L. Roytman*

IBM T. J. Watson Research Center,
P.O.Box 218
Yorktown Heights, NY 10598
(914) 945-2607

*The City College of New York
Department of Electrical Engineering
New York, NY 10031
(212) 690-8251

ABSTRACT

An architecture for a single-chip VLSI implementation of a Wigner distribution signal processor is presented. ASIC implementation in CMOS technology is considered with complexity of 65,000 gates, achieving a maximum throughput of 12K 16-bit samples per second. The architecture takes advantage of the high level of integration and low power consumption achievable with CMOS technology thus eliminating clock skew and off-chip delays associated with chip-crossing penalties. By integrating the Wigner processor on the single chip, the implementation is made practical and attractive for processing acoustic signals.

INTRODUCTION

The Wigner distribution (WD) has recently received considerable attention as a tool for time-frequency signal analysis. It is a 2-D real function that, loosely speaking, displays the time evolution of the frequency content of the signal. This makes it particularly useful for analysis and characterization of non-stationary and transient signals. In particular, the most recent research results indicate its usefulness in both active and passive underwater acoustic surveillance, identification of target signatures, speech analysis and recognition.

Several authors have addressed the issue of fast hardware implementations of the WD [1-4]. They came up with architectures suitable for real-time computation of the WD of signals with sampling rates up to a few Kilohertz. Early results [1,2] represented direct implementations of the defining equation in hardware. More recently, a microprogrammed implementation based on standard bipolar multiplier-accumulator chips has been proposed [3]. A systolic architecture based on the single modulus quadratic residue number representation has also been advanced lately [4]. However, all of these implementations use many chips or even several boards, resulting in considerable size and power dissipation, yet not providing a corresponding throughput increase.

Rapid advancements in VLSI technology make it possible to integrate more functionality on a single chip and maintain a relatively high processing speed, taking advantage of the proximity of the components and of the elimination of chip-crossing and clock skew penalties.

In this paper we undertook a study of VLSI implementation of Wigner distribution signal processor on a single chip. The estimate of the complexity in terms of the number of gates and propagation delays are based on data for currently available CMOS technology. In the evaluation of the performance and integration level we used data available for a fast turn-around ASIC technology which allows for rapid prototyping and implementation. The study shows that the throughput achievable with this technology is substantial and satisfactory for real-time underwater acoustic and speech signal processing. Using custom design methodology, further increase in the level of integration and throughput is possible. However, given the fast pace of advancements in technology, we believe that the advantage of rapid prototyping and fast turn-around outweighs the potential gains achievable with custom design.

Our estimates show that a single-chip implementation in CMOS-ASIC technology is quite feasible. The performance estimates obtained with a simulation of critical components and paths shows that a significant real time throughput rate is achievable.

COMPUTATION OF WIGNER DISTRIBUTION

In most applications dealing with band-pass signals it is advantageous to use the WD of the corresponding complex analytic signal. In this way, the redundant interfering cross-terms that exist around zero frequency in the WD of the real band-pass signal are eliminated. The imaginary part of the analytic signal is obtained through the Hilbert transformation of its real part which is equal to the original signal. The conventional way of computing the WD of the real signal requires oversampling (interpolation) of the signal by the factor of two. Although the computation of the WD of the analytic signal does not require oversampling, the total number of real samples representing the signal of interest (real vs. analytic) is the same in both cases. Since the WD of the analytic signal contains less redundancy, it is the representation of choice when dealing with band-pass signals, such as underwater acoustic and speech signals. Therefore, the 127-point FIR Hilbert transformer was included on the same chip with the WD processor.

Computation of the WD involves generation of the auto-product of the signal with its displaced and time reversed version, followed by the Fourier transformation. In our implementation, computation is optimized by taking advantage of the symmetry properties

21ACSSC-1/88/0939 $01.00

of the signal auto-product. By suitably combining the two successive auto-products, it is possible to generate two successive time slices of the WD in one step as the real and imaginary part of the Fourier transform of such combined auto-product. Each of these properties effectively doubles the throughput of the computation; resulting in the quadrupling of the performance, relative to the direct implementation.

The discrete-time WD (DTWD) is defined as [5] :

$$W(n,\theta) = 2\sum_{k=-\infty}^{\infty} z(n+k)\bar{z}(n-k)\exp(-j2\theta k) \qquad (1)$$

where overbar indicates complex conjugation. In the sequel we will take $z(n) = s(n) + j\hat{s}(n)$ where $\hat{s}(n)$ is the Hilbert transform of s(n).

In practice, only a finite amount of data can be processed and therefore windowing has to be employed. The WD of the windowed data then becomes:

$$W(n,\theta) = 2\sum_{k=-M}^{M} z(n+k)\bar{z}(n-k)w(k)\exp(-j\theta k) \qquad (2)$$

We will choose the rectangular window $w(k) = 1$, $-M \le k \le M$, as is usually done in practice. A different choice would result in only slight modification of the design that follows. The WD $W(n,\theta)$ can now be evaluated as a finite number of points $\theta = \frac{\pi}{N}m$, $m \in [0, N-1]$ that cover its basic period $\theta \in [0, \pi)$

$$W(n,m) = 2\sum_{k=-M}^{M} z(n+k)\bar{z}(n-k)\exp(-j\frac{2\pi}{N}mk) \qquad (3)$$

$$= 2\sum_{k=-M}^{M} r_n(k)\exp(-j\frac{2\pi}{N}mk)$$

where $r_n(k) = x_n + jy_n(k) = z(n+k)\bar{z}(n-k)$, $-M \le k \le M$.. Usually, $N = 2M+1$.

The auto-product $r_n(k)$ has a couple of symmetry properties that can be exploited in order to reduce the computation by a factor of 4. First,

$$r_n(k) = \bar{r}_n(-k) \quad , \quad -M \le k \le M \qquad (5)$$

Only the real part of the DFT of such a sequence will be non-zero. On the other hand, the DFT of the combination of two successive sequences

$$R_n(k) = r_n(k) + jr_{n+1}(k) = x_n(k) - y_{n+1}(k) + j[x_{n+1}(k) + y_n(k)] \qquad (6)$$

will have:

$$Re\{DFT[R_n(k)]\} = DFT[r_n(k)] = W(n,m) \qquad (7)$$

$$Im\{DFT[R_n(k)]\} = DFT[r_{n+1}(k)] = W(n+1,m) \qquad (8)$$

Therefore, a single DFT computation on $R_n(k)$ will produce two slices of the DTWD : W(n,m) and W(n+1,m). This reduces the number of DFT computations by 2.

Another symmetry property:

$$R_n(-k) = \bar{r}_n(k) + j\bar{r}_{n+1}(k) = x_n(k) + y_{n+1}(k) + j[x_{n+1}(k) - y_n(k)] \qquad (9)$$

implies that $x_n(k), y_n(k), y_{n+1}(k), x_{n+1}(k)$ should be evaluated only for non-negative k, $0 \le k \le M$. Therefore the number of multiplications required to compute $R_n(k)$ can be halved by exploiting this property :

$$W(n,m) + jW(n+1,m) = 2\sum_{k=-M}^{M} R_n(k)\exp(-j\frac{2\pi}{N}mk)$$

$$= 2R_n(0) + 2\sum_{k=1}^{M} R_n(k)\exp(-j\frac{2\pi}{N}mk) + R_n(-k)\exp(j\frac{2\pi}{N}mk)$$

$$= \{2x_n(0) + 4\sum_{k=1}^{M}[x_n(k)\cos(\frac{2\pi}{N}mk) + y_n(k)\sin(\frac{2\pi}{N}mk)]\}$$

$$+ j\{2x_{n+1}(0) + 4\sum_{k=1}^{M}[x_{n+1}(k)\cos(\frac{2\pi}{N}mk) + y_{n+1}(k)\sin(\frac{2\pi}{N}mk)]\} \qquad (10)$$

As a result we obtain the following algorithm (ignoring the scaling constant 4):

Given $s(i)$, $n - M \le i \le n + M + 1$

Repeat:

$$\hat{s}(n) = \sum_{k=0}^{31} h(k)[s(n+63-2k) - s(n-63+2k)]; \qquad (11)$$

$$\hat{s}(n+1) = \sum_{k=0}^{31} h(k)[s(n+64-2k) - s(n-62+2k)]; \qquad (12)$$

For m=0,N-1

$$W(n,m) = \frac{1}{2}[s^2(n) + \hat{s}^2(n)]. \qquad (13)$$

$$W(n+1,m) = \frac{1}{2}[s^2(n+1) + \hat{s}^2(n+1)]. \qquad (14)$$

For k=1,M

$$x_n(k) = s(n+k)s(n-k) + \hat{s}(n+k)\hat{s}(n-k). \qquad (15)$$

$$y_n(k) = \hat{s}(n+k)s(n-k) - s(n+k)\hat{s}(n-k); \qquad (16)$$

$$x_{n+1}(k) = s(n+1+k)s(n+1-k) + \hat{s}(n+1+k)\hat{s}(n+1-k). \qquad (17)$$

$$y_{n+1}(k) = \hat{s}(n+1+k)s(n+1-k) - s(n+1+k)\hat{s}(n+1-k); \qquad (18)$$

For m=0,N-1

$$W(n,m) = W(n,m) + x_n(k)\cos(\frac{2\pi}{N}mk) + y_n(k)\sin(\frac{2\pi}{N}mk). \qquad (19)$$

$$W(n+1,m) =$$

$$W(n+1,m) + x_{n+1}(k)\cos(\frac{2\pi}{N}mk) + y_{n+1}(k)\sin(\frac{2\pi}{N}mk); \qquad (20)$$

n=n+2;

until DONE.

ARCHITECTURE AND FEASIBILITY OF IMPLEMENTATION

The proposed architecture for computing WD trades-off the speed of FFT algorithms for the hardware simplicity and regularity of the direct DFT computation that allows re-use of the same simple hardware at different stages of computation. As a result, the processor fits on a single high-speed VLSI ASIC chip and achieves real-time performance with realistic data block length N=127.

The processor consists of (Fig.1.):

- two register files of 128, 16-bit registers organized as two-port-read, one-port-write (GRA, GRB). GRA contains real parts of data while GRB is used for storing imaginary parts.
- one look-ahead buffer of 64 16-bit registers (LAB).
- two read-only files containing 128, 16-bit constants. The register CRC contains 128 element table of cosine function and CRS contains sine function.
- read-only register file CRH contains table of 32 non-zero FIR Hilbert transformer coefficients, 16-bit wide.
- four 16 by 16 bit integer multipliers: M1,M2,M3,M4.
- two 16-bit adder/subtracters: AS1, AS2.
- The outputs of the AS1 and AS2 are accumulated in adders AD1 and AD2 and read-write register files GR1 and GR2.

At the end of the computational cycle GR1 and GR2 contain two slices of the WD taken at two consecutive times.

This organization of data storage facilitates the time multiplexing of the arithmetic unit since the existing data path is used and there is no need for extra bussing. Two extended accumulators are used to accumulate the results of summation and they are organized as two-port register files of 128 16-bit registers.

Operation

The first register file GRA is filled up with 128 signal samples. Look-ahead buffer (LAB) contains the next 64 samples. This LAB is updated at the signal sampling rate f_s. Both LAB and GRA are used as circular buffers: i.e., the "oldest" sample in LAB is overwritten by the incoming sample, and the "oldest" sample in GRA is overwritten by the "oldest" sample in LAB.

Phase-1: Hilbert transformation

127 point FIR filtering is performed according to the equations (11) and (12), using the data path shown in Fig. 2.

Signal samples s(n+63-2k) are taken from LAB and S(n-63+2k) are taken from the register file port B, passed through the multiplexers MX1 and MX2, multipliers M4 and M2 (being multiplied by 1) and subtracted in AS2.

The result is temporarily stored in SR4. Then the difference (content of SR4) is multiplied by the corresponding filter coefficient h(k), taken from CRH, using multiplier M3 and the result accumulated in SR1.

These two steps take one cycle each. They are repeated for each k ($k \in [0,31]$) to produce one sample of the Hilbert transformation. At the end, the result in SR1 is stored in the corresponding location of the circular buffer GRB(n). These steps are

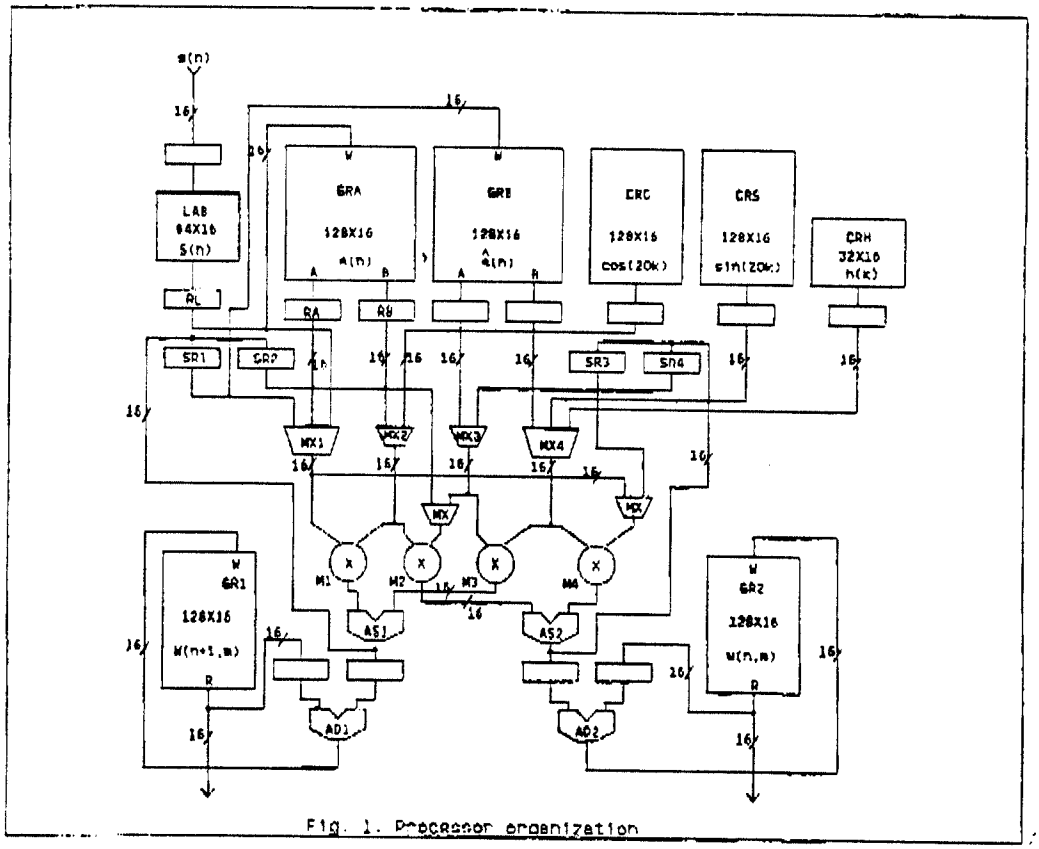

Fig. 1. Processor organization

then repeated to compute $\hat{s}(n+1)$ according to equation (12). Phase-1 takes 64 cycles per input sample, 128 total. At the end of this phase, GRA and GRB contain real and imaginary part of the analytic signal corresponding to the real input.

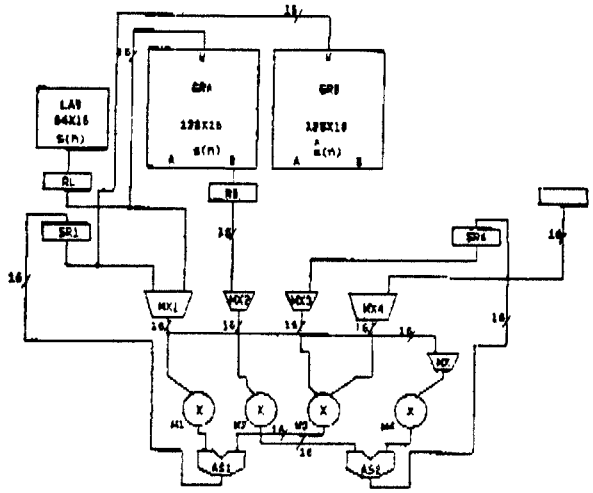

Fig. 2. Data path for Hilbert transformation eqs (11),(12)

Phase-2

Initialization of W(n,m) and W(n+1,m) is performed in this phase according to eqs. (13),(14), using the data path shown in Fig. 3. This process takes 2M cycles.

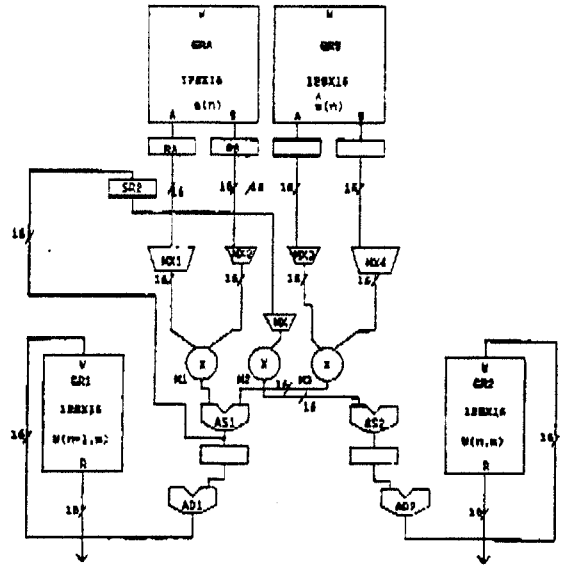

Fig. 3. Data path for initialization of W(n,m) and W(n+1,m), eqs (13),(14)

Phase-3

In this phase two successive complex auto-products (15),(16) and (17),(18) are computed. The computation process is depicted in Fig.4.

The results x_n and y_n are saved temporarily in SR2 and SR3 while x_{n+1} and y_{n+1} are stored in SR1 and SR4. This is done in two cycles.

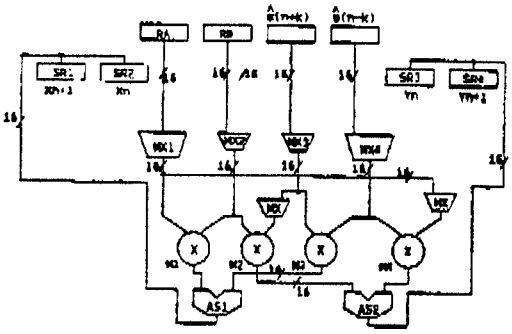

Fig. 4. Data path for computation of $x_n(k)$, $y_n(k)$ and $x_{n+1}(k)$, $y_{n+1}(k)$, eqs (15)-(18)

Phase-4

In Phase-4 $x_n(k)$, $y_n(k)$, $x_{n+1}(k)$, $y_{n+1}(k)$ are kept in SR1 - SR4 and used to compute partial sums (19) and (20). The computation is depicted in Fig.5. and takes N cycles.

The Phase-3 and Phase-4 are repeated M times for each value of k. The time spent in Phases 3 and 4 is (2+N)M clock cycles. The total computation per pair of input data samples takes $(2+N)M+128+2M = M(2M+5)+128 \cong 2M^2$ cycles. The number of cycles per one sample is then $M^2 \cong N^2/4$. This implies that the clock rate must be $f_n \leq \frac{N^2}{4} f_s$.

For each new pair of samples the process is restarted at Phase-1.

Our objective is to achieve real time performance with 12KHz sampling rate in a single chip implementation. With this archi-

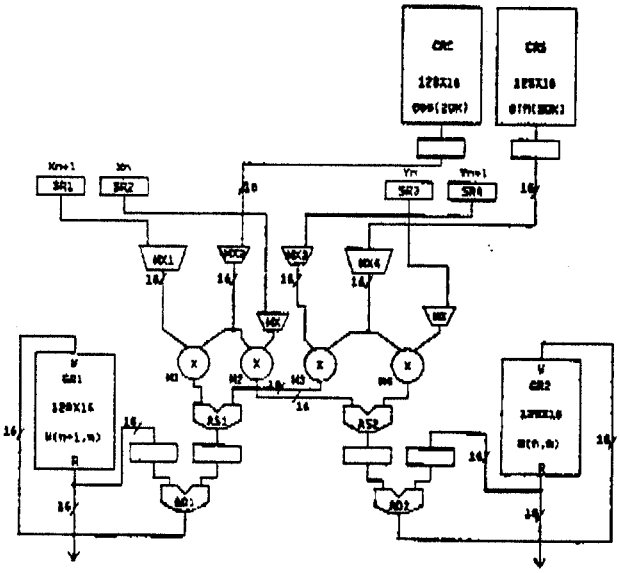

Fig. 5. Data path for accumulation of W(n,m) and W(n+1,m), eqs (19),(20)

tecture the maximum sampling rate for real time operation is limited to $f_s = 4f_{cl}/N^2$, where f_{cl} is the processor clock rate and N is the data block length. Maximal block length of 127 data points each represented with 16-bits is specified given the current technology constraints. By pipelining K such processors the throughput can be increased by K.

The implementation takes advantage of different rates at which signals need to be processed at different stages of computation. Therefore by time multiplexing an arithmetic unit designed to perform fast complex multiplication it is possible to save on hardware without significantly degrading the processor throughput. The arithmetic unit is utilized with such a flexibility that the Hilbert transformation, complex signal samples multiplication and multiplication with the complex exponential constants in the kernel of the Fourier transformation are all performed with the same unit. The arithmetic unit is designed for maximum speed since it determines the minimum clock cycle and therefore a maximum achievable throughput. It consists of four 16 bit multipliers and two 32 bit adder-subtracters. To alleviate the round-off errors 32-bit addition is performed after the multiplication of the complex terms. However this addition is hidden in an extended column compression structure which is, in essence, performing a carry-save function in adding the operand from the associated multiplier. The final addition which involves carry propagation is the addition of two product terms.

The multiplier is implemented using 2's complement Booth recoding scheme in order to reduce the number of product terms. The recoded multiplicand is used for both neighboring multipliers since it is wired to the same input and therefore the gate count is reduced. There are three recoded multiplicands used in the multipliers 1-2 , 2-3 and 3-4 respectively. The entire operation of multiplying four operands and adding the corresponding parts together yielding in an real and imaginary part of the complex number product is performed in one 20nS single clock cycle.

Complexity of the Implementation

In order to fit this implementation on a single chip, the largest currently available ASIC chip is required [6].

In this case we need approximately 65,000 gates which is currently available [5]. Estimated use of these gates is: 48,000 in arrays, 12,000 in the data path and 5,000 in the control logic. The size of the chip can be further reduced by using custom designed register files placed inside the ASIC chip.

The chip requires a relatively low pin-count 64-pin package which reduces the cost of manufacturing. With the clock cycle of 20nS, 12KHz throughput is achievable with the currently available CMOS technology.

It is possible to place this implementation on a smaller ASIC chip, like LCA10129 [7],[8], in which case the data block size needs to be reduced to by a factor C and loss in frequency domain resolution is to be expected. This, on the other hand, allows increase in the sampling frequency $f_s \le \frac{4f_{cl}}{N^2}$ by C^2

CONCLUSION

We have shown that it is possible to integrate the WD processor on a single chip without sacrificing resolution or performance. The algorithm for computing WD is tailored for time-sharing of relatively simple and fast hardware modules. This makes implementation of WD practical and attractive for acoustic signal processing.

REFERENCES

[1] D. Chester, F. Taylor, M. Doyle, *On the Wigner Distribution*, Proc. ICASSP-83, pp.491-494, March 1983.

[2] P. Fludrin, W. Martin, M. Zakharia, *On a Hardware Implementation of the Wigner-Ville Transform* , Proc. Intl. Conf. on Digital Signal Proc., Sept. 1984.

[3] B. Boashash, P. Black, H. Whitehouse, *An Efficient Implementation for Real-Time Applications of Wigner Distribution* , Proc. SPIE, vol. 698, pp 22-33, Aug. 1986.

[4] J. Wilbur, F. Taylor, *High-Speed Wigner Processing Based on a Single Modulus Quadratic Residue Number System* , Proc. ICASSP-87, pp. 1031-1034, Apr. 1987.

[5] T.A.C.M. Claasen, W.F.G. Mecklenbrauker, *The Wigner Distribution - A Tool for Time-Frequency Signal Analysis*, Part II, Philips J. Res. 35, pp. 276-300, 1980.

[6] *LSI Packs 100K Gates On Chip*, Electronics Engineering Times, October 26, 1987.

[7] *LCA 100000 Compacted Array Series* , LSI Logic Corp., June, 1986.

[8] *A High Performance 129K Gate CMOS Array* , available from LSI Logic Corp.