S-38.220
Licentiate Course on Signal Processing in Communications,
FALL - 97

Implementation of multiuser receivers

Pekka Karppinen

Nokia Telecommunications
Radio Access Systems/New Radio Systems
P.O. Box 300 FIN-00045 Nokia Group

Email: pekka.karppinen@ntc.nokia.com

Date: 20.11.1997

Abstract

The scope of this paper is to give some idea about available DSP-hardware and design methodology that can be used for the implementation of multiuser receivers. The aim is to point out the requirements for implementation of a real time multiuser receiver and how they might be reached.

1. INTRODUCTION

Implementation of CDMA receivers has recently become more intresting, since the WCDMA (Wideband CDMA) concept is a quite strong choice for the third generation of mobile communication systems. However, the timetable of the third generation products looks such that we should consider what kind of progress is to be expected within the DSP-harware area when we make decisions about the future receiver algorithms. This is why implementation of multiuser receivers is seriously considered.

2. TECHNOLOGY AND METHODS

2.1 ADC and sampling rate

If WCDMA is widely accepted and becomes a standard it is highly probable that the becoming chip rate will be 4.096 MHz. Sampling rate in CDMA receivers is often selected so that there is 4 - 5 time oversampling, which makes total analog-to-digital-conversion rate about 16.384 - 20.480 MHz. Furthermore, ADC has to be capable of sampling quite wide bandwidth - approximately 5 MHz (or 2.5 MHz at the baseband). These facts together limit the number of effective bits of the ADC. Current ADCs fullfilling the requirements are capable of something like 10 - 12 effective bits. This is certainly a fact that has to be considered in receiver design today.

Finite word-length effects on multiuser receiver performance has been studied only on very simple cases and low MAI. It would be intresting to see what effect it would have if everything, along channel estimation, was quantized to these word lengths. Fortunately some good development is going on within ADC field and we can expect better ADC performance in near future. In table 1 there are listed some experimental ADCs and their performance claimed by manufacturers.

Another result from high sampling rate is that we might not want to play around with the code generator clocks. An alternative method for fine adjustments in code-tracking loops is to use fractional delay filters for delaying the data. The effect is that the code 'hits' right moments in the data-stream.

Table 1 : Some experimental ADCs and their performance

according to manufacturers, HP = Hewlett Packard

Manufacturer	number of bits	Msample / s
Rocwell	6	4000
HP	8	2000
Hughes	12	100
Hughes	14	24
HP	18	10

2.2 DSP processors

DSP-processors are arhitectured to accelerate the execution of repetitive, numerically intensive calculations common in DSP. Common processor features are:

- Circuitry to perform multiply-accumulate operation in one clock cycle

- Multiple access memory, which enables the processor to load multiple operands, such as data sample and a filter coefficient, simultaneously and in parallel with an instruction

- Various special memory-addressing modes and program-flow control features designed to speed the execution of repetitive operations

- Special on chip peripherals or I/O interfaces that enable the processor to interface efficiently with other system components, such as ADCs and memory

There are basically two kinds of DSP-processors, fixed - and floating-point ones. A fixed-point processor operates basically with integers, but for convenience numbers are normalized to represent numbers between -1 and 1. A floating point number, however, consists of a sign-bit, exponent bits and fraction bits. Overall wordlength is usually 32 bits for single precision, and 64 bits for double precision floating-point numbers. Floating-point arithmetic is considerably more complex to implement in real hardware than the fixed-point arithmetic. Therefore floating-point processors tend to be bigger in terms of area, power consumption and cost. They are also slower than the fixed-point processors.

Processor manufacturers like to represent their processor capabilities with MIPS - or Mflops that a processor can perform. MIPS means Million Instructions Per Second and Mflops Million floating-point operations per second respectively. These can, however, be quite missleading numbers. What an average processor instruction really does cannot be determined accurately. Different processors need different amount of instructions to accoplish a certain task. Therefore comparing the ratings, given by manufacturers, is not proper way to determine processor performances. Furthermore, manufacturers also give their ratings as high as possible without telling a complete lie. For example Texas Instruments claims that TMS320C62x processors can reach 1600 MIPS. This is true if the program is exactly suitable for the processor. The truth is that this processor is quaranteed to produce 200 MIPS and the rest is up to application. As an controller TMS320C62x might not handle much more than 200 MIPS. TI is going to start marketing a new TMS320C67x floating-point processor capable of 1000 Mflops in the second half of 1998. The processor has very much similarities with the

TMS320C62x. What is said here about TMS320C62x will very probable be the case with this new processor also. What is really coming out in practical situations is not likely to be 1000 Mflops.

As we are talking about implementation of multiuser CDMA receivers, we should think about how different processors are suitable for multiprocessing environment. The fact is, as can be seen from table 2, that there is no single processor capable of doing any multiuser detection task. What can also be seen is that we can consider using processors only on symbol-rate processing tasks like channel estimation and multipath combining, leaving every chip-rate processing task to ASICs. One processor could only handle despreding of one multipath component. That would mean quite many processors for implementation of 'only' a RAKE -receiver.

Table 2: Some currently available DSPs

Vendor	Processor family	arithmetic	data width	MIPS / Mflops
Analog Devices	ADSP -21xx	Fixed	16	33.3
	ADSP -210xx	Floating	32	40.0
AT&T	DSP16xx	fixed	16	70.0
	DSP32xx	floating	32	20.0
Motorola	DSP5600x	fixed	24	40.0
	DSP561xx	fixed	16	30.0
	DSP563xx	fixed	24	80.0
	DSP96002	floating	32	20.0
NEC	μPD7701x	fixed	16	33.3
Texas Instruments	TMS320C3x	floating	32	25.0
	TMS320C4x	floating	32	30.0
	TMS320C5x	fixed	16	50.0
	TMS320C54x	fixed	16	50.0
	TMS320C62x	fixed	16	1600
Zoran	ZR3800x	fixed	20	33.3

2.3 VHDL, what is possible

VHDL is a hardware descripiton, not a programmming, language. It is used for both hardware modeling and ASIC-design. The subset of VHDL suitable for physical ASIC-design is limited and is called RTL - VHDL. This RTL (register transfer logic) level VHDL can be synthesized to an ASIC with automatized synthesis tools. What kind of logic is going to be synthesized can directly be seen from the rtl-level code.

Such comments as 'Let's map this algorithm to an ASIC via VHDL !' are often heard. This is not going happen as easy as that if the algorithms are not suited for it in the first place. You can not convert C-code matrix calculations and other high level stuff directly to VHDL. These thoughts are probably due to promises given by ASIC-tool vendors. They clame having tools that can create RTL-level code from behavioral level code. They are not really working yet in all cases and the quality of the result is

poor. Even if the tools will work reasonably in the future, the situation is going to be like it has been with Assembly vs. C-code in DSP-processor coding.

2.4 ASICs

Nowadays practically every digital ASIC is designed with some hardware description language, whether it is VHDL or Verilog. HDL (hardware description language) is synthesized with synthesis tools to a certain Vendors (ASIC - manufacturers) library. This library contains standard components such as multipliers adders etc. that have already been drawn into silicon. Synthesis tools take right components from the library and bind them together forming a complete circuit. However, even if we use synthesis and other automatized tools available, designing a complex ASIC is a huge task. It is usually calculated that completing an ASIC-design project will take a year if standard design procedures are followed. If there is going to be floating-point arithmetic or other kind of very special operations that hardware description language do not support directly, then the work load is going to be much higher.

It is quite difficult to estimate how long it would take to design a DSP-ASIC with floating-point accuracy(floating-point arithmetic is often mentioned with MUDs and matrix inversions). At least it would cost. A 16x16 bit multiplier uses roughly 2000 gates depending on multiplier algorithm. A single precision floating-point number contains 32 bits. It is obvious that the cost of a floating-point multiplier is at least twice the cost of a 16x16 bit fixed-point multiplier in terms of gate count.

Accurate multiplications are also slow. ASIC-vendors give some maximum clock frequencies that their certain technologies can use. However, if we go to one clock cycle multiplications (which we probably must do with MUDs), the usable clock frequency is much less than the maximum frequency given by technology specifications. Some estimate is that accurate one clock cycle fixed-point multiplications could be possible at the rate of 100 MHz with the new 0.25 μm tecnologies.

3. MOST SUITABLE ALGORITHMS FOR IMPLEMENTATION

3.1 SIC with PC

Successive interference cancellation alone is not very efficient multiuser approach. However, there has been some ideas of combining SIC and power control in a way that still might put the SIC scheme back in business.

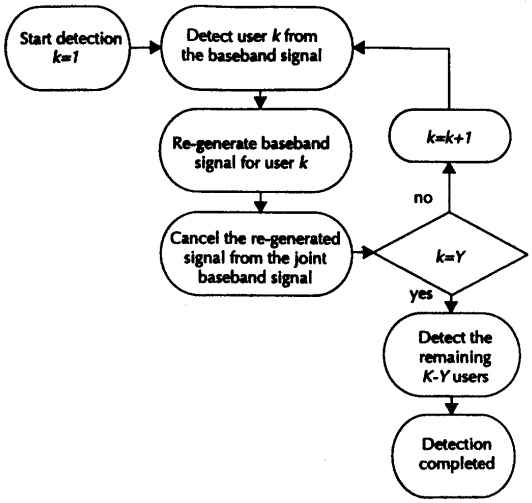

Figure 2: The SIC - scheme [9]

Power control is usually implemented such that it aims for equal received power for every user. An alternative power control strategy suitable for SIC has been proposed. In this strategy powers are adjusted to give each user equal BER with the use of SIC in uplink detection. This means unequal powers deliberately adjusted by PC. Furthermore, the strongest power is allocated to closest user compared to the base station, which results to reduced inter-cell interference. This shouldn't be very difficult since due to code aquisition needed for each user, the distance from each mobile to base station is accurately known.The strategy helps typical fault propagation in SIC - detection and might end up with surprisingly good results.

A fixed-point SIC prototype has been made by WINLAB. They selected pipelined structure, which is quite natural to SIC. Each user has it's own IC-filter, which consist of a RAKE-receiver and a re-generator.

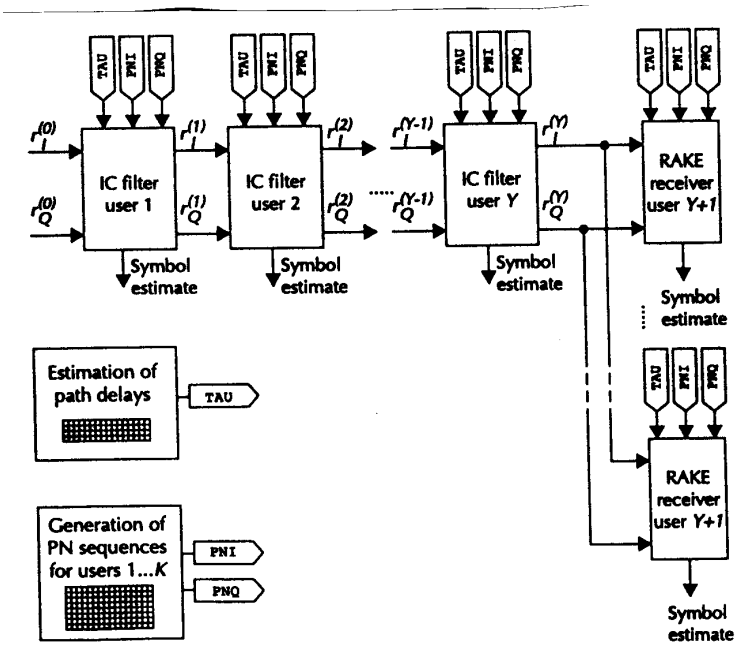

Figure 3: Pipeline [9]

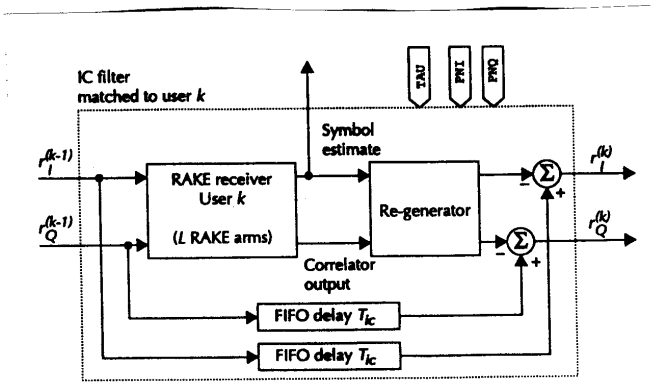

Figure 4: IC-filter [9]

After detecting each user with the RAKE, a interference estimate is calculated by the re-generator. The estimate is then substracted from the received signal. However, due to limitations in maximum tolerable overall delay of the receiver, it is possible that the number of interference cancellations made has to be limited to some practical number. In figure 5 is presented mapping of operations to ASICs and DSPs in one IC-filter. This is quite typical conventional detector structure. Chip-rate as well as parallel(needed in re-generator) processing is done is ASICs, symbol rate processing in DSPs.

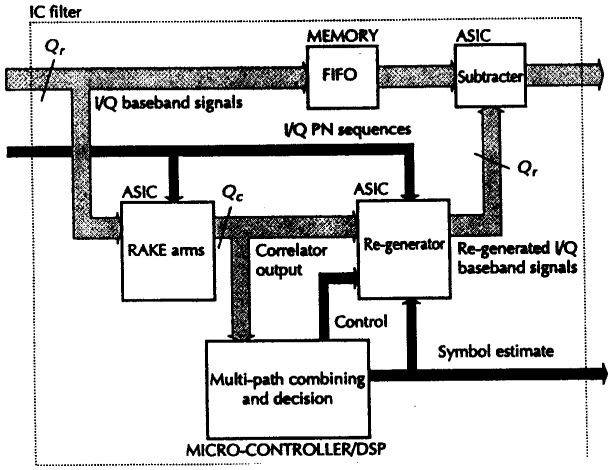

Figure 5 : Hardware mapping [9]

What makes SIC so intresting is the fact that there is not really very much additional hardware needed to implement SIC compared to having only RAKE-receivers for each user. Additional components are the re-generators as well as slightly more complex overall control structures. Some estimates are that the SIC-receiver would be less than 20 % more complex than having only those individual RAKEs.

3.2 Decorrelating detectors

Some versions of the decorrelating detector are considered among those that might be implementable in the real world. The decorrelating detector is repeated here to give some idea about the matrix sizes. The output **y** of the matched filter bank for whole data packet can be expressed as

$$\mathbf{y} = RCA\mathbf{b} + \mathbf{w} \qquad (1)$$

where

$$R = \begin{bmatrix} \mathbf{R}^0(0) & \mathbf{R}^{T(1)}(1) & \mathbf{R}^{T(2)}(2) & \dots & \mathbf{0}_{KL} \\ \mathbf{R}^1(1) & \mathbf{R}^0(0) & \mathbf{R}^{T(1)}(1) & \dots & \mathbf{0}_{KL} \\ \mathbf{R}^2(2) & \mathbf{R}^1(1) & \mathbf{R}^0(0) & \dots & \mathbf{0}_{KL} \\ \vdots & \vdots & \vdots & . & \vdots \\ \mathbf{0}_{KL} & \mathbf{0}_{KL} & \mathbf{0}_{KL} & \dots & \mathbf{R}^0(0) \end{bmatrix}$$

$$\in \Re^{N_b KL \times N_b KL} \qquad (2)$$

is the correlation matrix over whole data packet, **b** is transmitted bits and **w** is noise vector.

$$\mathbf{R}^{(n)}(i) = \begin{bmatrix} \mathbf{R}^{(n)}{}_{1,1}(i) & \mathbf{R}^{(n)}{}_{1,1}(i) & \dots & \mathbf{R}^{(n)}{}_{1,K}(i) \\ \mathbf{R}^{(n)}{}_{2,1}(i) & \mathbf{R}^{(n)}{}_{2,2}(i) & \dots & \mathbf{R}^{(n)}{}_{2,K}(i) \\ \vdots & \vdots & . & \vdots \\ \mathbf{R}^{(n)}{}_{K,1}(i) & \mathbf{R}^{(n)}{}_{K,2}(i) & \dots & \mathbf{R}^{(n)}{}_{K,K}(i) \end{bmatrix}$$

$$\in \Re^{KL \times KL} \qquad (3)$$

is the correlation matrix of the n th symbol interval. Matrices

$$(\mathbf{R}^{(n)}{}_{k,k'})_{l,l'} = \int_{-\infty}^{\infty} s_k{}^{(n)}(t - \tau_k - \tau_{k,l}) s_{k'}{}^{(n-i)}(t + iT - \tau_{k'} - \tau_{k',l'}) dt \qquad (4)$$

$$\in \Re^{LxL}$$

$$l, l' \in \{1,2,\dots L\}$$
$$k, k' \in \{1,2,\dots K\}$$

are correlation matrices of users k and k' multipath components l and l' caused by symbol intervals n and (n-i). Number of users is K and number of multipath components is L. Signature waveforms s are expected to be real, which they in practical cases are not.

$$C = diag(\mathbf{C}^{(0)}, \mathbf{C}^{(1)}, \dots, \mathbf{C}^{(N_b - 1)}) \qquad (5)$$

$$\in \mathbf{C}^{N_b KL \times N_b K}$$

is a diagonal matrix of complex channel coefficient at the symbol intervals 0,1,...N_b - 1, where

$$\mathbf{C}^{(n)} = diag(\mathbf{c}_1{}^{(n)}, \mathbf{c}_2{}^{(n)}, \dots, \mathbf{c}_K{}^{(n)}) \qquad (6)$$

$$\in \mathbf{C}^{KL \times K}$$

is a diagonal matrix of complex channel coefficients for each users L multipath components at the symbol interval n. A is a diagonal matrix of transmitted amplitudes for each user at the symbol intervals 0,1,..,N_b -1. The decorrelating detector satisfies

$$R^{(n)} D_d = U \qquad (7)$$

where R is given by (2), D is the detector matrix and U is

$$U = (0_{KL}, 0_{KL}, \dots, 0_{KL}, \mathbf{I}_{KL}, 0_{KL}, \dots, 0_{KL}, 0_{KL}) \qquad (8)$$

Clearly D_d is the inversion of the matrix R. In practice the decorrelating detector has to be truncated to some symbol interval length N. The truncated decorrelating detector satisfies

$$R^{(n)} D_{dN} = U_N \qquad (9)$$

where D_{dN} is the NKL x KL middle block column of the inverse of R. A linear multiuser detector satisfying

$$\left[\bar{R}^{(n)} + \sigma^2 (C^{(n)} A^{(n)} A^{H(n)} C^{H(n)})^{-1} \right] D_{msN} = U_N \qquad (10)$$

is the truncated LMMSE detector. The truncated noise whitening detector on the other hand satisfies

$$L^{(n)} D_{nwN} = U_N \qquad (11)$$

where L is the lower triangular Cholesky factor of R. Let's now assume that we have the correlation matrix R available. We still would have to invert it to get the detector matrix. The computational load of matrix inversion is related to the cubic of the matrix size. In practical situations we would have to have resources for some 50 users and 8 multipath components in the detector. This will explode the computational load sky high. The only way to do the decorrelation is via iterative methods.

3.3 The Conjugate Gradient Algorithm

The correlation matrix is symmetric and block-Toepliz, which makes the solving of the detector matrix suitable for iterative algorithms. The conjugate gradient method is suitable for solving equations of the type

$$\mathbf{Ax} = \mathbf{b} \qquad (12)$$

where A is a symmetric positive definite matrix. With this condition solving of (12) is equal to minimizing quadratic function

$$q(\mathbf{x}) = (1/2)\mathbf{x}^T \mathbf{Ax} - \mathbf{x}^T \mathbf{b} \qquad (13)$$

The gradient of (13) is Ax - b, which is zero when q(x) is minimum. An iteration step of this algorithm is

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_k \mathbf{p}_k \qquad (14)$$

where α_k is a scalar step size and p_k is the direction vector. For a given $\mathbf{x}_{k-1}$ and $\mathbf{p}_k$ the scalar α_k is chosen to minimize $q(\mathbf{x}_k)$. That is α_k is the value α for which

$q(\mathbf{x}_{k-1}+\alpha\mathbf{p}_k)$ is minimum. The function $q(\mathbf{x}_{k-1}+\alpha\mathbf{x}_k)$ is quadratic in α and its minimization leads to the condition

$$\alpha_k = \frac{\mathbf{p}_k{}^T \mathbf{r}_{k-1}}{\mathbf{p}_k{}^T \mathbf{A}\mathbf{p}_k} \qquad (15)$$

where $r_{k-1} = b - Ax_{k-1}$ is residual after k - 1 iterations. The residual need not be computed explicitly in each iteration, since it can be computed incrementally by using its value from previous iteration. In the k th iteration the residual can be expressed as

$$\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$$

$$= \mathbf{r}_{k-1} - \alpha_k \mathbf{A}\mathbf{p}_k \qquad (16)$$

Thus the only matrix vector product computed in each iteration is $\mathbf{Ap_k}$, which is already required to compute α_k . If **A** is symmetric positive definite matrix and $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n$ are direction vectors that are conjugate with respect to **A** (that is $\mathbf{p_i}^T\mathbf{Ap_j} = 0$ for all $i > 0$, $j \le n$, $i \neq j$) , then $\mathbf{x}_k$ converges to **x** of equation (12) in at most n iterations, assuming no rounding errors. In practice acceptable convergence is reached within much less than n iterations. The set of conjugate direction vectors is chosen

$$\mathbf{p}_1 = \mathbf{r}_0 = \mathbf{b}$$

$$\mathbf{p}_{k+1} = \mathbf{r}_k + \frac{\|\mathbf{r}_k\|_2^{\,2}}{\|\mathbf{r}_{k-1}\|_2^{\,2}} \mathbf{p}_k \qquad (17)$$

3.4 Preconditioned CG algorithm (PCG)

The convergence speed of the CG algorithm depends on the condition number κ of $R^{(n)}$, which is defined as the eigenvalue ratio

$$\kappa(R^{(n)}) = \frac{\lambda_{\max}(R^{(n)})}{\lambda_{\min}(R^{(n)})} \qquad (18)$$

where λ_{max} and λ_{min} are the largest and smallest eigenvalues of the matrix R. The larger the ratio the slower the convergence rate. Convergence can be speeded by preconditioning

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{x} = \mathbf{P}^{-1}\mathbf{b} \qquad (19)$$

where **P** is the preconditioner. It should be selected so that it is

- easy to construct

- easy to invert
- eigenvalue ratio of combined matrix $\mathbf{P}^{-1}\mathbf{A}$ should be small

However, with time-varying signature waveforms finding proper conditioner is difficult. A proposal for a preconditioner with decorrelating detectors is to use mathed filter bank output for the this purpose.

3.5 PIC

Parallel interference cancellation is among the most intresting multiuser algorithms since it is efficient as well as cost effective in practice. Especially Hard Decision - PIC (HD-PIC) shows to be worth considering in future receivers. The ouput of the n th symbol interval for the HD-PIC detector is

$$\mathbf{y}_{PIC}^{(n)}(m) = \mathbf{y}^{(n)} - \hat{\psi}_{PIC}^{(n)}(m) \qquad (20)$$

$$\hat{\psi}_{PIC}^{(n)}(m) = \sum_{i=-P_{PIC}}^{P_{PIC}} (\mathbf{R}(-i) - \delta_{i,0}\mathbf{I}_{KL})\hat{\mathbf{C}}^{(n+i)}(m-1)\mathbf{A}\hat{\mathbf{b}}^{(n+1)}(m-1) \qquad (21)$$

Here $\hat{\psi}_{\mathrm{PIC}}^{(\mathrm{n})}$ (m) is the multiple access interference estimate for stage m. (21) contains tentative estimates of complex channel coefficients and transmitted bits as well as correlation matrix for delay index i.

4. COMPLEXITY COMPARISONS

Very optimstic complexity calculations has been done in [1], which for example do not take into account the fact that the correlation matrix has to be calculated for each symbol interval when using CG - algorithm or other decorrelating detector versions. However, it has been stated in [1] that it would take $2N_s$ flops to calculate one correlation coefficient. Here N_s is the number of samples in symbol interval. If signature waveforms are time varying, as they are, all $(KL)^2$ correlation coefficients have to be calculated. Therefore the cost of calculating the correlation matrix is $O(2N_s(KL)^2)$ flops for each symbol interval.

An other very heavy calculation is of the type vector multiplied by correlation matrix (2) in decorrelating detector calculations. The correlation matrix is sparse. Using this fact and otherwise doing straight forward calculations (FFT multiplication and using circulant matrices would lead to better result) it would take $O(8N(KL)^2)$ flops to do

the multiplication. If the number of iterations for acceptable convergence (CG) is M, then it would take $O(8MN(KL)^2)$ flops for this type of calculations during symbol interval (CG).

For calculating (21) for MF-output [1] it is required $O(4M(KL)^2)$ flops, where M is the number of stages. Calculating (21) for MF-input will take $O(4MN_sKL)$ flops. Summary of these flops is presented in table 3. It should be reminded that these expressions are representing only simplified number of calculation operations necessary for each algorithm.

Table 3

Algorithm	**flops / symbol interval**
Matched filtering	**$O(4N_sKL)$**
CG - detection	**$O((2N_s+8NM)(KL)^2)$**
HD-PIC MF_out	**$O(4M(KL)^2)$**
HD-PIC MF_in	**$O(4MN_sKL)$**

Let's now calculate what it would roughly take in practical case in mobile communication systems. Then parameters might be

K	=	50	
L	=	8	
spreading ratio=	128		
samples per chip	=	5	
M (CG)	=	124	(used in [1])
M (PIC)	=	2	(used in [1])
N	=	5	
kbaud	=	32	

Numerical results for each symbol interval yields

Table 4

Algorithm	**flops / second**
Matched filtering	**$3{,}27 \bullet 10^{10}$**
CG - detection	**$3.19 \bullet 10^{13}$**
HD-PIC MF_out	**$4.1 \bullet 10^{10}$**
HD-PIC MF_in	**$6.55 \bullet 10^{10}$**

Note that correlator calculations are not included as required flops in tables 3 and 4 (exept for matched filtering).

5. PRACTICAL IMPLEMENTATION ISSUES

There are two very heavy computations remaining with the iterative methods. The other is the correlation matrix computation and the other is of the form a vector

multiplied with a matrix (for example $\mathbf{Ap_k}$ in the CG). If we think about previous example case, then the correlation matrix (2) would be of the size 2000•2000, which might cause considerable trouble in implementation. Therefore we have to think about effective methods for this type of operation.

5.1 Using circulant matrices to reduce complexity

A matrix vector product would normally require O(n^2) operations. However, if the matrix is Toepliz, then the multiplication can be done in O(n log n) operations using circulant matrices and FFT. A matrix is said to be circulant if

$$C_n = \begin{bmatrix} c_0 & c_{-1} & \cdots & c_{2-n} & c_{1-n} \\ c_1 & c_0 & c_{-1} & \cdots & c_{2-n} \\ c_2 & . & . & \cdots & . \\ c_{n-1} & c_{n-2} & \cdots & c_1 & c_0 \end{bmatrix} \qquad (22)$$

where $c_{-i} = c_{n-i}$. A property of circulant matrices is that the columns of the Fourier matrix of order n are the eigenvalues of circulant matrices. So,

$$C_n = F_n^* \Lambda F_n \qquad (23)$$

where Λ is a diagonal matrix of eigenvalues of C_n and F_n is the Fourier matrix. Now, we can embed an n•n Toeplitz matrix in a circulant matrix of size at most 2n. A circulant matrix is uniquely defined by one row - the rest of the rows are circular shifts of that row. We can form the first row by augmenting the first row of the Toeplitz matrix in the reverse order. A numerical example follows,

$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 4 & 1 & 2 & 3 \\ 5 & 4 & 1 & 2 \\ 0 & 5 & 4 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 & 0 & 5 & 4 \\ 4 & 1 & 2 & 3 & 0 & 5 \\ 5 & 4 & 1 & 2 & 3 & 0 \\ 0 & 5 & 4 & 1 & 2 & 3 \\ & & & & & \\ 3 & 0 & 5 & 4 & 1 & 2 \\ 2 & 3 & 0 & 5 & 4 & 1 \end{bmatrix}$$

Now to calculate $\mathbf{Ap_k}$ we calculate the embedded matrix vector product

$$C_n \begin{bmatrix} \mathbf{p} & 0 \end{bmatrix}^T = \begin{bmatrix} \mathbf{A} & \mathbf{B}^* \\ \mathbf{B} & \mathbf{V} \end{bmatrix} \begin{bmatrix} \mathbf{p}_k \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{Ap}_k \\ 0 \end{bmatrix} \qquad (24)$$

where C_n is a circulant matrix. Therefore

$$C_n \begin{bmatrix} \mathbf{p} & 0 \end{bmatrix}^T = F^* \Lambda F \begin{bmatrix} \mathbf{p} & 0 \end{bmatrix}^T \qquad (25)$$

can be calculated with two Fourier transforms of order 2n. If we now only divide the calculation correctly we may use the FFT here for this purpose and end up with O(n log n) steps. Thus the matrix vector product can be done in O(n log n) steps. With slight modifications the method can be applied to block-Toeplitz systems also.

5.2 VLSI architecture for computing Arithmetic Fourier Transform

It has been stated [10] though that computing FFT is probably not the most efficient way for Fourier transform calculation in VLSI circuits, since it requires O(n log n) complex multiplications. Studies have been made on Arithmetic Fourier Transform which uses O(N^2) additions and O(N) multiplications for the Fourier transform. The AFT uses Möbius inversion of series and zero-order interpolation to Fourier series computation. However, the AFT is not presented here in detail since it can be read from [10]. The proposed architecture uses systolic array structure which is suitable for highly parallel computations.

The systolic array design differs from the conventional Von Neumann computer in its highly pipelined computation. Once a data item is fetched from memory it can be used effectively at each cell it passes while pumped from cell to cell along the array.
The structure is especially suitable for compute-bound operations. Compute-bound operations are defined to be the ones where the total number of computing operations exceed the number of I / O operations, which we now have the case.
In systolic array data is processed in small processing elements (PE) that regularly take one sample in, process it, and put it back into the data stream. Usually the only control signal is the clock.

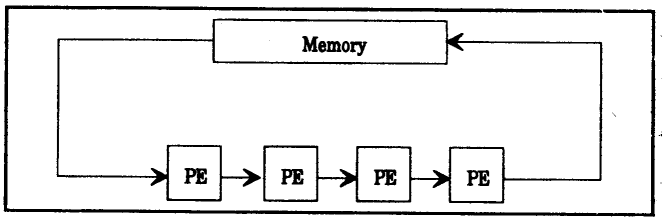

Figure 6 : Basic configuration of systolic arrays [11]

In [10] has been presented two promising structures for calculating (25). The structure is systolic array that produces 2N+1 Fourier coefficients from 2N input samples. One of those is presented here. The overall architecture is shown in figure x. The N PE's in group1 perform computation of the alternating averages. PE_{mul} performs the scaling of alternating computed in group1 and also computes the Fourier coefficients a_0 . The N PEs in group2 compute the Fourier coefficients on-the-fly using the scaled alternating averages computed in PE_{mul} . The total number of PEs is 2N + 1. In table x is the summary of the needed resources.

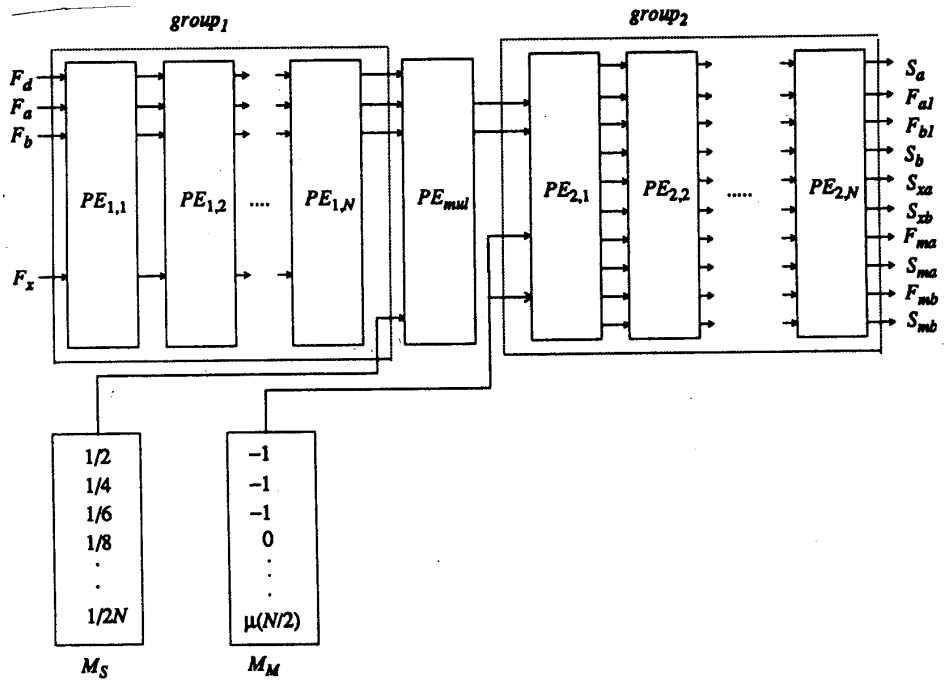

Figure 7 : The overall architecture [10]

Table 5: Needed resources for AFT VLSI implementation [10]

Resource type	Count
adders	2N
multipliers	N
registers	O(N)
delay	5N + N/2
input period	2N
I / O bandwidth	fixed
external memory size	≤ 1.5 N

It now seems that we might survive the matrix vector operation in the CG algorithm with reasonable costs after all.

5.3 Things to be considered

In real applications the only problem is not to get the algorithm working. It is also supposed to work synchronized to every other operation in the system. For example in real systems we have some form of random access channels for call establishment purposes. Catching these may be quite challenging with multiuser detectors.

Protyping MUDs is going to be tricky. The hardware capacity needed to implement any MUDs is so huge that the only possibility is to go straight away to full scale ASICs. This is expencive since the process might need many iterations.

6. CONCLUSIONS

The implemenation of multiuser detectors looks quite tough cookie. With present tehcnology it seems that the only multiuser detectors that are within reach are the versions of PIC and SIC. Implementing a multiuser detector is so huge and costly design task that even big companies will not start the process unless there are quaranteed large markets for them. These markets might be the third generation of mobile communication systems. However, it looks quite obvious that even if WCDMA is selected and accepted widely, the multiuser detectors will not be there in the first system versions. It might be so that some kind of PIC version will be updated to systems after the markets are opened and profitable.

7. REFERENCES

[1] Markku Juntti: "Multiuser demodulation for ds-cdma systems in fading channels", Oulu university press, 1997

[2] Suman Das, Joseph R. Cavallaro, Benhaam Aazhang : "Computationally Efficient Multiuser Detectors", Proceedings of PIMRC'97, Helsinki, Finland, September 1997, pp 62-67.

[3] Kazuyski Yasukawa and Laurence B. Milstein: " Finite Word Length effects on Performance of MMSE Receiver for DS-CDMA Systems", Proceedings of PIMRC'97, Helsinki, Finland, September 1997, pp 724-728.

[4] P.M. Grant : "DSP-chips for CDMA Mobile Communications", Proceedings of ISSTA'96, Mainz, Germany , September 1996, pp. 18-26.

[5] Ravi Subramanian, Marc Berberis, Frank Gerbic, Biswa Ghosh: "Design and

Implementation of All-Digital Receivers For Mobile Communications", Proceedings of IEEE VTS 46 th Vehicular Technology Conference, Atlanta, USA, April 28 - May 1 1996, pp 1043-1047.

[6] Jefferey A. Wepman: "Analog-to-digital Converters and Their Applications in Radio Receivers", IEEE Communications Magazine,May 1995 Vol. 33 No. 5, pp 39-45.

[7] Vipin Kumar, Ananth Grama, Anshul Gupta, George Karypis, "Introduction to Parallel Computing", The Benjamin/Cummings Publihing Company Inc., 1994

[8] Philip Lapsley, Garrick Blalock; "How to estimate DSP processor performance", IEEE Spectrum, July 1996, pp. 74 - 78.

[9] Klaus I. Pedersen, Troels E. Kolding, Ivan Seskar, Jack M. Holtzman: "Practical Implementation of Successive Interference Cancellation in DS/CDMA Systems", Proceedings of 5 th International Conference on Uneversal Personal Communications, October 1996, pp. 1-4.

[10] Heonchul Park, Viktor K. Prasanna: " Modular VLSI Architectures for Computing the Arithmetic Fourier Transform", IEEE Transactions on Signal Processing, June 1993, Vol 41, pp. 2236-2246.

[11] Ramin Baghaie:" Systolic Implementation of the Discrete Fourier & the Discrete Hartley Transforms, Report 14, Laboratory of Signal Processing and Computer technology, 1994.