Baseband Architecture Design for Future Wireless Base-Station Receivers

Sridhar Rajagopal

April 26, 2000

This work is supported by Nokia, Texas Instruments, Texas Advanced Technology Program and NSF
Outline

- Background
- Multiuser Channel Estimation and Detection
- DSP Implementation and Task Partitioning
- Reduced Complexity Algorithms
- VLSI Architecture
- Architecture/Extensions for DSPs and GPPs
Evolution of Wireless Communications

**First Generation**
Voice

**Second/Current Generation**
Voice + Low-rate Data (9.6Kbps)

**Third Generation +**
Voice + High-rate Data (2 Mbps) + Multimedia
W-CDMA
Communication System Uplink

Direct Path

Reflected Paths

Noise + MAI

Base Station

User 1

User 2
Main Processing Blocks

Baseband Layer of Base-Station Receiver

Antenna

Multiple Users

Demodulator

Delay

Pilot

MUX

Data

Multiuser Detection

Channel Estimation

MUX

MUX

Detector

Decision Feedback

d

b

Detected Bits

Baseband Layer of Base-Station Receiver
Real-time Requirements

- Multiple Data Rates by Varying Spreading Factors
- Detection needs to be done in real-time
  - 1953 cycles available in a C6x DSP at 250MHz to detect 1 bit at 128 Kbps

<table>
<thead>
<tr>
<th>Spreading Factor</th>
<th>Number of Bits / Frame</th>
<th>Data Rate Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>10240</td>
<td>1024 Kbps</td>
</tr>
<tr>
<td>32</td>
<td>1280</td>
<td>128 Kbps</td>
</tr>
<tr>
<td>256</td>
<td>160</td>
<td>16 Kbps</td>
</tr>
</tbody>
</table>
Outline

- Background
- **Multiuser Channel Estimation and Detection**
- DSP Implementation and Task Partitioning
- Reduced Complexity Algorithms
- VLSI Architecture
- Architecture/Extensions for DSPs and GPPs
Channel Model

\[ r_i = A_i b_i + \eta_i \]

\[ b_i \in \mathbb{R}^{2^{K}} \quad \text{Bits of K async. users aligned at times I and I-1} \]

\[ r_i \in \mathbb{C}^{N} \quad \text{Received bits of spreading length N for K users} \]

• **Compute Correlation Matrices**

\[ R_{br} = \frac{1}{L} \sum b_i r_i^H \]

\[ R_{bb} = \frac{1}{L} \sum b_i b_i^T \]
Channel Estimation

Solve for the channel estimate, $A_i$

$$R_{bb} A_i = R_{br}$$

$A_i \in C^{2K \times N}$

$A_i = \begin{bmatrix} A_0 & A_l \end{bmatrix}$

Multishot

$$r = \begin{bmatrix} A_0 & A_l & 0 & 0 & 0 \\ 0 & A_0 & A_l & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & A_0 \end{bmatrix} \begin{bmatrix} b_{1,1} \\ \vdots \\ b_{K,1} \\ \vdots \\ b_{1,D} \\ \vdots \\ b_{K,D} \end{bmatrix} + \eta$$

$A \in C^{ND \times KD}$
Differencing Multistage Detection

- **Stage 0- Matched Filter**
  \[ y^0 = \text{Re}[ A^H r ] \]
  \[ d^0 = \text{sign} ( y^0 ) \]

- **Stage 1**
  \[ y^1 = y^0 - \text{Re}[A^H A - S]d^0 \]
  \[ d^1 = \text{sign}(y^1) \]

- **Successive Stages**
  \[ x^l = d^l - d^{l-1} \]
  \[ y^{l+1} = y^l - \text{Re}[A^H A - S]x^l \]
  \[ d^{l+1} = \text{sign}(y^{l+1}) \]

\[ S = \text{diag}(A^H A) \]

- \( y \) - soft decision
- \( d \) - detected bits

(hard decision)
Structure of $A^HA$

Block Bi-Diagonal Matrix

$$A^H A \in \mathbb{R}^{KD \times KD}$$

$$
\begin{bmatrix}
A_0^H A_0 & A_0^H A_l & 0 & 0 \\
A_l^H A_0 & A_0^H A_0 + A_l^H A_l & A_0^H A_l & 0 \\
\vdots & \ddots & \ddots & \vdots \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & A_l^H A_0 & A_0^H A_0 + A_l^H A_l
\end{bmatrix}
$$
Outline

- Background
- Multiuser Channel Estimation and Detection
- DSP Implementation and Task Partitioning
- Reduced Complexity Algorithms
- VLSI Architecture
- Architecture/Extensions for DSPs and GPPs
Current DSP Implementation

Data Rate Comparisons for Matched Filter and Multiuser Detector

Targeted Data Rate = 128Kbps

- Matched Filter (C64)*
- Multiuser Detector (C64)*
- Matched Filter (C67)
- Multiuser Detector (C67)
- Targeted Data Rate

Projected (8x)

C67 at 166MHz
Reasons for Poor Performance

- Sophisticated, Compute-Intensive Algorithms
- Need more MIPs/FLOPs performance
- Unable to fully exploit pipelining or parallelism
- Bit - level computations / Storage
Task Decomposition [Asilomar’99]

Block I
Correlation Matrices (Per Bit)

- $R_{br}[R] O(KN)$
- $R_{br}[I] O(KN)$
- $R_{bb} O(K^2)$

Block II
Inverse

- $R_{bb} A^H = R_{br}[R] O(K^2N)$
- $R_{bb} A^H = R_{br}[I] O(K^2N)$

Block III
Matrix Products

- $A_0^H A_1 O(K^2N)$
- $A_0^H A_0 O(K^2N)$
- $A_1^H A_1 O(K^2N)$

Block IV
Multistage Detection
(Per Window)

- $O(DK^2Me)$

Matched Filter

Channel Estimation
Achieved Data Rates

Data Rates for Different Levels of Pipelining and Parallelism

Data Rate Requirement = 128 Kbps
Task Partitioning Hardware Req.

- $O(K^2)$ processing elements
- 1024 for $K = 32$
- Can meet Real-Time
  - Not feasible in hardware
Outline

- Background
- Channel Estimation and Detection
- DSP Implementation and Task Partitioning
- **Reduced Complexity Algorithms**
- VLSI Architecture
- Architecture/Extensions for DSPs and GPPs
Iterative Scheme for Estimation

\[
R_{bb} = \sum b_i b_i^T \\
R_{br} = \sum b_i r_i^H \\
R_{bb} \ast A_i = R_{br}
\]

\[
R_{bb} = R_{bb} + b_L \ast b_L^T - b_0 \ast b_0^T \\
R_{br} = R_{br} + b_L \ast r_L^H - b_0 \ast r_0^H \\
A = A - \mu (A \ast R_{bb} - R_{br})
\]

- Tracking
- Method of Gradient Descent
- Stable convergence behavior
  - Symmetric, Positive Definite $R_{bb}$
  - $\mu$ - MAI, SNR, Preamble length
- Same Performance
Simulations - AWGN Channel

Detection Window = 12
SINR = 0
Paths = 3
Preamble L = 150
Spreading N = 31
Users K = 15
10000 bits/user

MF – Matched Filter
ML – Maximum Likelihood
ACT – using inversion

Comparison of Bit Error Rates (BER)

$\frac{10^1}{10^2}$

Signal to Noise Ratio (SNR)

$10^{-3} - 10^{-1}$

$O(K^2N)$

$O(K^3 + K^2N)$
Fading Channel with Tracking

Doppler = 10 Hz, 1000 Bits, 15 users, 3 Paths

![Graph showing BER vs SNR for different methods: MF - Static, MF - Tracking, ML - Static, ML - Tracking.]
Pre-computed Preamble

- Preamble bits $b_i$ known at the receiver
- Reduces Complexity, if pre-computed.

$$R_{br} = \frac{1}{L} \sum b_i r_i^H$$

$$R_{bb} = \frac{1}{L} \sum b_i b_i^T$$

$$R_{bb} A_i = R_{br}$$

$$R_{bb} c_i = b_i$$

$$A_i = A_i + c_i r_i$$
Computational Savings in Estimation

- Pre-computed Auto-correlation has large savings
  - Can be used only for quasi-static channels and initial acquisition.

<table>
<thead>
<tr>
<th>Precomputed Preamble</th>
<th>Original</th>
<th>Iterative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>$O(4K^2N)$</td>
<td>$O(2KN)$</td>
</tr>
<tr>
<td>No</td>
<td>$O(6K^3 + 4K^2N)$</td>
<td>$O(4K^2N)$</td>
</tr>
</tbody>
</table>
Detection

\[ x^l = d^l - d^{l-1} \]

\[ y^{l+1} = y^l - \text{Re}[A^H A - S]x^l \]

\[ d^{l+1} = \text{sign}(y^{l+1}) \]

\[
\begin{bmatrix}
A^H A_0 & A_0^H A_l & 0 & 0 \\
A^H A_0 & A_0^H A_0 + A_l^H A & A_0^H A_l & 0 \\
\vdots & \vdots & \vdots & \vdots \\
0 & 0 & A^H A_0 & A_0^H A_0 + A_l^H A_l \\
\end{bmatrix}
\begin{bmatrix}
x_{1,1} \\
x_{k,1} \\
\vdots \\
x_{1,D} \\
x_{k,D} \\
\end{bmatrix}
\]

\[ y_i = y_i - Lx_{i-1} - Cx_i - Rx_{i+1} \]
Pipelined Detection Scheme

Interference from previous bits of other users

Interference from future bits of other users

Desired User

User 1

User j
Block Based Detector

Matched Filter

Stage 1

Stage 2

Stage 3

Bits 2-11

Bits 12-21
Pipelined Detector

Matched Filter

Stage 1

Stage 2

Stage 3
Computational Savings in Detection

- Edge Bits are not computed
- Bit-streaming
- Simpler Hardware Structure
- $6K^2$ per Window Savings

<table>
<thead>
<tr>
<th>Original</th>
<th>Pipelined</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(DNK + 3(D + 2)K^2)$</td>
<td>$O(DNK + 3DK^2)$</td>
</tr>
</tbody>
</table>
Outline

- Background
- Real-Time Requirements
- DSP Implementation and Task Partitioning
- Reduced Complexity Algorithms
- VLSI Architecture
- Architecture/Extensions for DSPs and GPPs
VLSI Implementation [ASAP’2000]

- Channel Estimation as a Case Study
- Area - Time Efficient Architecture
- Real - Time Implementation
  - Minimum Area Overhead
- Bit- Level Computations - FPGAs
- Core Operations - DSPs
Area-Time Tradeoffs

- **Area-Constrained Architecture**
  - Pico-cells; lower data rates

- **Time-Constrained Architecture**
  - Maximum achievable data rates

- **Area-Time Efficient Architecture**
  - Real-Time with minimum area overhead

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Full Adder Cells</th>
<th>Memory (Bytes)</th>
<th>Time</th>
<th>Data Rates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>248</td>
<td>16 K</td>
<td>0.262 ms</td>
<td>3.81 Kbps</td>
</tr>
<tr>
<td>Time</td>
<td>2x10^7</td>
<td>4 K</td>
<td>12 ns</td>
<td>83.33 Mbps</td>
</tr>
<tr>
<td>Area-Time</td>
<td>10^4</td>
<td>16 K</td>
<td>4 µs</td>
<td>256 Kbps</td>
</tr>
<tr>
<td>C67 DSP</td>
<td>-</td>
<td>128 K</td>
<td>0.97 ms</td>
<td>1.02 Kbps</td>
</tr>
</tbody>
</table>
Outline

- Background
- Channel Estimation and Detection
- DSP Implementation and Task Partitioning
- Reduced Complexity Algorithms
- VLSI Architecture
- Architecture/Extensions for DSPs and GPPs
Motivation for Architecture

- Wireless, the next wave after Multimedia
- Highly Compute-Intensive Algorithms
- Real-Time Requirements
Characteristics of Wireless Algorithms

- Massive Parallelism
- Bit-level Computations
- Matrix Based Operations
  - Memory Intensive
- Complex-valued Data
- Approximate Computations
Why Reconfigurable

- Adapt algorithms to environment
- Seamless and Continuous Data Processing during Handoffs
Different Protocols

- MPEG-4, H.723 - Voice, Multimedia
- Convolutional, Turbo - Channel Coding
Reconfigurable Support

- **Configuration Caches**
  - Recently Displaced Configurations (5 cycles)
  - Can hold 4 full size Configurations

- **Independent Execution**
Permutation Based Interleaved Memory

- High Memory Bandwidth Needed
- Stride-Insensitive Memory System for Matrices
- Randomizes access
- Multiple Banks
- Sustained Peak Throughput (95%)
Instruction Set Extensions

- To accelerate Bit level computations in Wireless

- Integer - Bit Multiplications
  - Multiuser Detection, Decoding, Cross Correlation

- Bit - Bit Multiplications
  - Auto-Correlation, Channel Estimation

- Useful in other Signal Processing applications
  - Speech, Video,
SIMD Parallelism

64-bit Register A

64-bit Register B

64-bit Register C
For $i = 1..8$, $j = 1..8$

\[ D[i][j] = D[i][j] + b[i] \times C[j] \]  
(Cross-Correlation)
Computational Savings

- Avoid bit multiplications and control structures
- 4 8-bit Multiply
  - Latency 3
- 8 8-bit Add
  - Latency 1
- Cross-Correlation Example
  - 64 multiply, 64 add

<table>
<thead>
<tr>
<th>Original SIMD Instruction Set</th>
<th>$12 \times 3 + 8 \times 1 = 54$ cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>With Extensions</td>
<td>$8 \times 1 = 8$ cycles</td>
</tr>
</tbody>
</table>
Truncated Multipliers

- Many applications need approximate computations
- Adaptive Algorithms: $Y = Y + \mu(Y \times C)$
- Truncate lower bits
- Half the area/half the delay
- Can do 2 truncated multiplies in parallel with regular

**ALU Multipliers**

| Multiplier 1 | Multiplier 2 | Truncated Multiplier |
Future Work

- Long Codes - Implementation
- Online Arithmetic
- Multiprocessing on DSPs and FPGAs
Conclusions

- Architecture and Algorithms to meet real-time
- Task Decomposition
  - Real Time with Multiple Processing Elements
- Iterative Algorithms
  - Reduce Complexity, Simpler Implementation
- VLSI Implementation
  - Real-Time with minimum Area Overhead
- Architecture/Extensions to DSPs and GPPs