AN2253
Rev. 2, 11/2004

Channel Estimation for a WCDMA Rake Receiver

By Ahsan Aziz

The third-generation universal mobile telecommunications system (UMTS) uses direct-sequence code division multiple access (DS-CDMA) to transmit data. DS-CDMA is well suited to transmit data over the multipath fading channel. The high signal bandwidth for the WCDMA system (5 MHz) allows the received signal to be split into distinct multipaths with high resolution. For the first generation of UMTS handsets, a rake receiver is the receiver of choice because it allocates one rake finger to each multipath, thus maximizing the amount of received signal energy. The rake receiver combines these different paths into a composite signal with substantially better characteristics for demodulation than a single path. To combine the different paths meaningfully, the rake receiver needs such channel parameters as the number of paths, their location (in the delay domain), and their (complex-valued) attenuation. In a WCDMA system, the necessary channel parameters must be estimated and tracked throughout the transmission.

This application note discusses the different methods for channel estimation and then describes an implementation of channel estimation on the StarCore™ SC140/SC1400 cores. It examines the steps in the channel estimation algorithm and presents the channel estimation routine. The maximum load for this implementation is 1.0732 MIPS. The size of the filter is the designer's choice and can be fixed or changed adaptively for best performance based on an estimate of the doppler spread. The simulation results and the DSP implementation match almost exactly. Accuracy is greatly improved because 40-bit adds can be used instead of the 16-bit multiples. The algorithm performs well, as indicated by the plots presented in the results section of this document.

CONTENTS

freescale™ semiconductor

1 Channel Estimation

Typically channel estimation is performed using one of three methods or a combination of these methods. The trade-offs between complexity and performance dictate the choice of algorithm:

- *Data Aided Channel Estimation:* Known pilot symbols are transmitted. At the receiver end, the channel estimation algorithm operates on the received signal along with its stored symbols to generate an estimate of the transmission channel.
- *Decision-Directed Channel Estimation:* A rough estimate of the channel is obtained using a suitable estimation method. Then this estimate is used to make symbol decisions. The channel estimate is further improved using the resulting symbols as "pilot symbols." This type of estimation contains some inherent delay because the symbol decisions occur before the final channel estimate can be made. Also, there may be error propagations because any errors in the symbol decisions affect the final estimate.
- *Blind Channel Estimation:* This estimation process relies not on pilot symbols or symbol decisions but rather on certain characteristics of the modulated signal. For example, the constant modulo algorithm (CMA) uses the amplitude of the signal as the criterion for estimating the channel. In constant energy modulation schemes such as Quadrature Phase Shift Keying (QPSK), the knowledge that all signals are transmitted with equal energy is used as the basis for obtaining the channel estimate. This type of algorithm typically requires a longer convergence time and usually has a higher mean square error (MSE) than the other two schemes.

In a typical WCDMA rake receiver, channel estimates are used to combine the multipaths. **Figure 1** shows how different paths are combined using Maximum Ratio Combining (MRC). In **Figure 1**, $r(t)$ is the received signal, which is split into $r(t-\tau_i)$. Note that, $g(t,\tau_i)$ is the corresponding channel estimate for each path is $r(t-\tau_i)$. The objective is to estimate the channel phase and amplitude (denoted in **Figure 1** as $g(t,\tau i)$) for each of the identified paths. This information is then used for combining each path of the received signal.

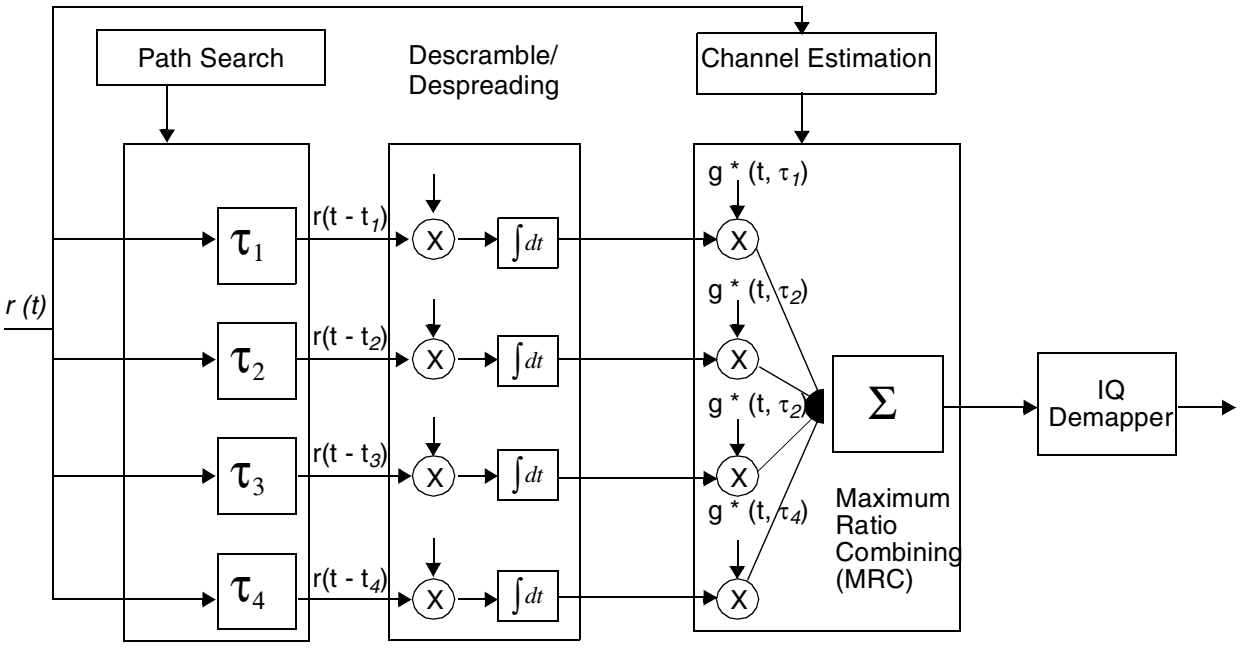


Figure 1. Rake Receiver for WCDMA

As **Figure 1** shows, the following steps occur in a WCDMA receiver (excluding the error correction coding):

1. *Descrambling*. Received signals are multiplied by the scrambling code and delayed versions of the scrambling code. A path searcher determines the delays prior to descrambling. Each delay corresponds to a separate multipath that is to be combined by the rake receiver
2. *Despreading*. The descrambled data of each path is despread by simply multiplying the descrambled data by the spreading code.
3. *Integration and dump*. The despread data is integrated over one symbol period, giving one complex sample output per QPSK symbol. This process is carried out for all paths to be combined by the rake receiver.
4. *Symbol combining*. The same symbols obtained via different paths are then combined using the corresponding channel information and a combining scheme such as MRC.
5. The combined output is sent to a simple decision device to decide on the transmitted bits.

The objective of the channel estimation block is to estimate the channel phase and amplitude (denoted in **Figure 1** as *g(t,ti)*) for each of the identified paths. Once this information is known, it can be used for combining each path of the received signal.

The preceding operations are mathematically expressed in the equations that follow. In**Equation 1**, *u(t)* is the transmitted signal. Here a_k is the complex transmitted symbol, p_k is the complex spreading (OVSF) and scrambling code combined, N is the spreading factor, *f(t)* is the pulse shaping filter (root raised cosine with roll-off factor 0.22), and T is the chip duration.

Equation 1

$$u(t) = \sum_{K=0}^{K-1} a_k \sum_{n=0}^{N-1} p_k(n) f(t - nT - kNT)$$

Equation 2 shows the received signal *y(t)* at the mobile unit. The mobile channel is modeled as a filter with complex taps given by c_j and delays of d_j, where *{j=0..J-1}* for *J-1* different paths and *g(t)* is the Additive White Gaussian Noise (AWGN) with single-sided Power Spectral Density (PSD), N_0

Equation 2

$$y(t) = \sum_{j=0}^{J-1} c_j u(t - d_j) + g(t)$$

At the receiver end the received data is first passed through a matched filter (matched to the transmitter filter). Match filtering maximizes the Signal-to-Noise Ratio (SNR) at the receiver. This process is shown in **Equation 3**.

Equation 3

$$r(t) = \int f^*(\tau - t) y(\tau) d\tau$$

A path searcher estimates the delay for the different paths. Then the received signal is delayed by the amount estimated by the path searcher and multiplied by the scrambling and spreading code (code used for transmission). The descrambled and despread data are then summed over one symbol period as shown in **Equation 4**. For example, if there are four strong paths, four different estimates of the same symbol are generated using **Equation 4** and are combined by the rake receiver with the corresponding channel estimate as shown in **Equation 5**.

Equation 4

$$x_k(t) = \frac{1}{N}\sum_{m=0}^{N-1} p_k^*(n)r(t+mT-kNT)$$

In **Equation 5**, C_j is the channel estimates, d_j is the estimated path delays, J is the estimated number of strong paths, *det( )* is a simple decision device, and â is the estimated bit/symbol obtained at the output of the rake receiver.

Equation 5

$$\hat{a} = det\left(\sum_{j=0}^{\hat{J}-1} \hat{c_j}^* x_k(\hat{d}_j)\right)$$

2 Downlink Channel Estimation in WCDMA

In WCDMA, channel estimation can be performed using a Common Pilot Channel (CPICH) or the time-multiplexed pilot bits in a "dedicated traffic channel." This section describes both of these processes and considers their advantages/disadvantages. In the downlink traffic channel (DPDCH/DPCCH), pilot symbols (2 to 8 symbols) and control symbols are transmitted in every slot. There are 15 slots per WCDMA frame. Each frame is 10 ms long and has 38400 chips (3.84 Mcps). Channel estimates can be made using these pilot symbols. If these time-multiplexed pilot data bits are used, then the estimate for the data bits in between two consecutive sets of pilot bits (two slots) can be obtained by interpolation. A decision-directed approach can be used to improve performance. **Figure 2** diagrams the process of channel estimation using time-multiplexed symbols.

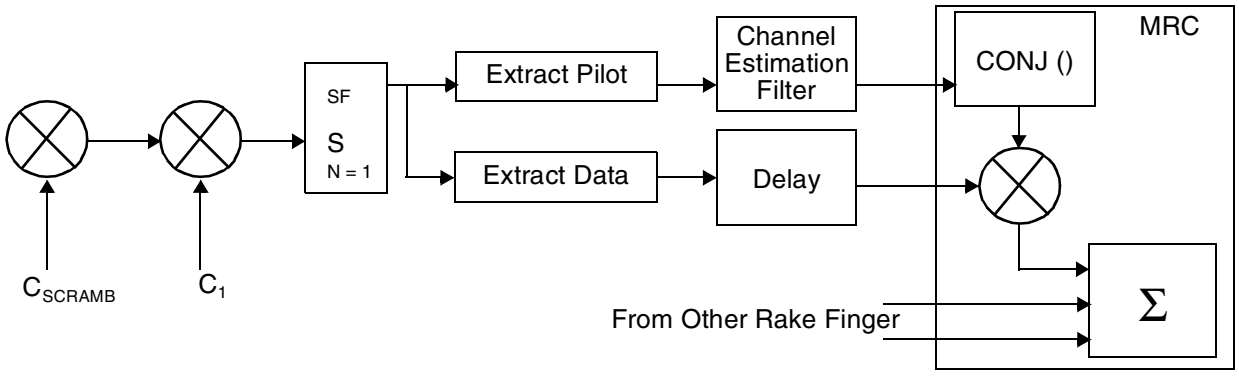


Figure 2. Channel Estimation Using Time-Multiplexed Pilot Symbols

Also, in the downlink of the WCDMA system, a Common Control Channel (CPICH) is transmitted with a higher power than the dedicated traffic channels. All mobiles in the cell receive this channel. The CPICH is transmitted with a constant spreading factor (SF) of 256 and a spreading code of all ones. Thus, there are 10 symbols per slot and 150 symbols per frame of a CPICH. All the symbols of the CPICH are *1-j*. On the receiver end, the symbols work as pilot symbols and can be used for channel estimation.

The advantage of using CPICH for channel estimation is that all the data in the frame can be used as opposed to only a few symbols in the DPCCH/DPDCH. Also, since this data is transmitted with a higher power than in the traffic channel, reception at the handset is improved. One situation in which the time-multiplexed pilot bits become useful is when the mobile is at the cell edge. This is because the dedicated channels are power controlled, whereas the CPICH is not power controlled. **Figure 3** shows a typical downlink WCDMA DPDCH/DPCCH frame.

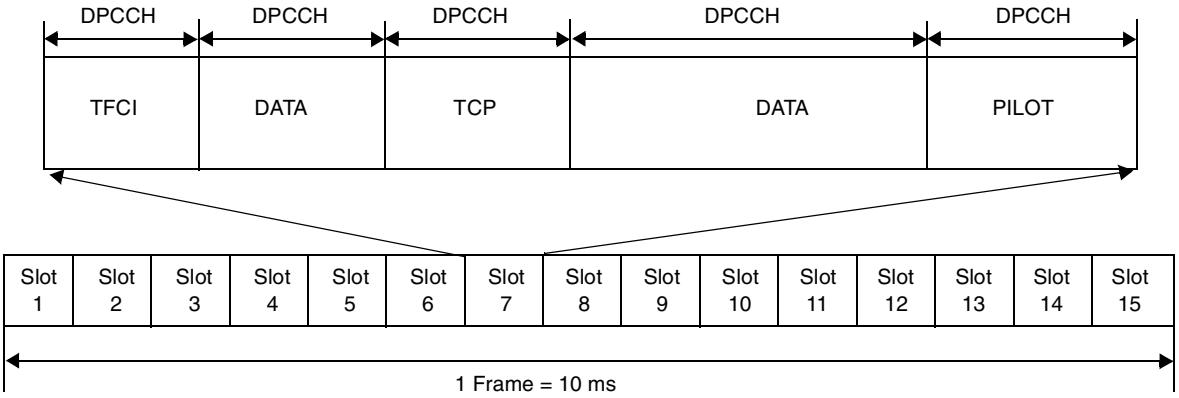


Figure 3. Downlink Dedicated Traffic Channel (DPDCH/DPCCH)

The channel estimation algorithm implemented on the SC140 core uses the CPIC process. **Figure 4** defines the channel estimation process.

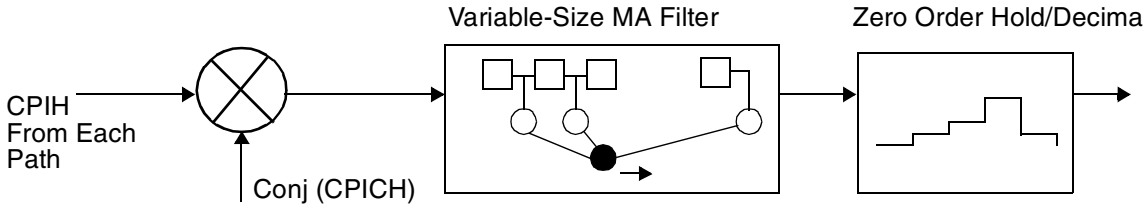


Figure 4. Channel Estimation Using CPICH

For each independent path, the channel estimate is obtained as follows:

1. Remove the modulation form CPICH by simply multiplying the CPICH data by its conjugate. In this case, it is simply *1+j* because all the CPICH symbols are *1+j*. The resulting channel estimate is noisy because of AWGN and multiple-user interference.
2. Pass the noisy channel estimate through a variable-length moving average filter. In the StarCore DSP implementation, the channel length can be N = 8, 16, or 32.
3. Decimate or interpolate the filtered channel estimate obtained in step 2 to match the data rate of the CPICH to the data rate of the DPCCH/DPDCH. The process of interpolation is performed by a simple zero-order hold (that is, a simple repeater). This works well in most cases because the channel is assumed to be stable for the symbol duration.

3 Implementation on the SC140 Core

For the implementation of channel estimation on the SC140 core, it is assumed that all the CPICH data are scaled properly so that it is representable in 16 bits. The channel estimation proceeds as follows:

1. Perform complex multiplication of the received CPICH with the conjugate of the original CPICH symbols to remove the modulation. That is, multiply the received CPICH data by *1-j* on a per symbol basis. If the received CPICH symbols are a_i+jb_i, the complex multiplication is as follows:

Equation 6

$$(a+jb_i)*(1-j)=(a_i+b_i) + j(b_i-a_i)$$

Data is read out of the memory holding the CPICH symbols (in the assembly code, this is pointed by Rx_CPICH). After the additions shown in **Equation 6** are performed, the data is written to the memory identified by the start address **#FIR_buffer**. The memory write starts from the location to which *WCDMA frame start* is pointing. All 150 samples corresponding to the CPICH frame are written from *WCDMA frame start* to *WCDMA frame end*. **Figure 5** shows the structure of **FIR_buffer**.

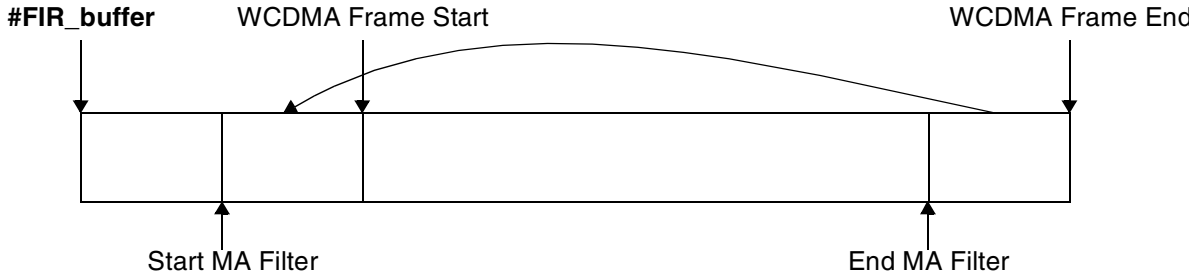


Figure 5. Data Movement in Memory

During initialization of the code, a space equal to 64 CPICH symbols is cleared (**#FIR_buffer** to *WCDMA frame start*). In the implementation presented here, the maximum size of the moving average filter is set to 32, but a memory of size 64 CPICH symbols is cleared for future use (for implementing an MA filter of length 64). The MA filter starts with data located at *Start MA filter* and runs till *End of MA filter*. In the first iteration, the first *N* samples are zeros. The filter runs all the way up to *End MA filter*. This implies that there is a delay of *N* samples in the output.

2. Before the new frame of the unfiltered channel estimate is written to the **#FIR_buffer**, the last ***N*** samples of the previous frame's unfiltered data are copied to the front of *WCDMA frame start*. This provides the continuity of data for the MA filter (see **Figure 3**). Filtering resumes from *Start MA filter*.

3. The MA filter is defined in **Equation 7**:

Equation 7

$$\hat{c}_j = \sum_{i=0}^{N-1} a(i)\tilde{c}_j(n-i) \qquad \text{Where } a(i) = 1/N$$

$\tilde{c}_j(n-i)$ is the noisy channel estimate from the CPICH, $a(i)$ is the filter coefficient, and $\hat{c}_j$ is the final channel estimate for use by the MRC. All the filter coefficients are equal (1/N). As **Equation 2** shows, the number of memory reads/writes and **MAC** operations can be reduced simply by adding a new sample (scaled by 1/N) to the running sum and removing the oldest sample from the running sum. This is the process used in computing the MA filter output in the DSP. However, the very first sample of the running sum is computed by reading four complex numbers from the buffer at a time (two **move.4f** instructions), performing four **MAC** operations per cycle, and repeating the loop N/4 times (see loop "MA_filter_step_one" in the assembly code).

4. The output of the MA filter is written to memory to which **#Ch_est** points. Before the results are written, a “rate matching” must be performed between the CPICH and the traffic channel (DPCCH/DPDCH). CPICH is always transmitted with a spreading factor of 256, and the traffic channel can be transmitted with a spreading factor in the range from 4 to 512, depending on the data rate needed. This “rate matching” is performed by a simple zero-order hold. A true interpolation filter is not needed because the channel changes much more slowly than the symbol interval. The assembly code presented here assumes that the interpolation factor is provided by some other blocks in the system. This factor is denoted by **SF_MATCH** in the code. **SF_MATCH** can have the following values:

SF	SF_MATCH
4	64
8	32
16	16
32	8
64	4
128	2
256	1
512	–-1

An **SF_MATCH** value of –1 refers to decimation, in which every alternate sample from the output of the MA filter is chosen. The code in **Example 1** shows the SC140 channel estimation routine.

Example 1. SC140 Core Channel Estimation Routine

```
;*************************************************************************
;;*
;*  File:             channel_estimation.asm
;*  Function:         Channel Estimation for WCDMA Rake Receiver
;*  Author:           Ahsan Aziz
;*  Version/Date:     oct 10,2001
;*
;*  Target Processor: StarCore SC140
;*
;*  Description:
;*  Module Details:
;*
;* Registers Used:
;*  d0,d1,d2,d3,d4,d5,d6,d7,d8,d9,d10,d11,d12,d13,d14,d15
;*  r0,r1,r2,r3,r4,r5,r8,r9
;*  n0
;*  entry : jsr
;*************************************************************************
;*
;*  Revision History:   Date       Change Details           Initials
;*                      ----       --------------           --------
;*************************************************************************

        section .data local
        include 'channel_est_param.asm'
        endsec
```

```
;*********************WCDMA Channel Estimation*********************************

        section .text local
        global _main_channel_estimation
_main_channel_estimation type func
 move.l #N,d0   ; N=Number of tap of the MA filter
 move.l #8,d1
 cmpeq d0,d1
 move.l #16,d1
 ift move.l #3,d15                                    ; if N=8 the tap weights are
                                                      ; 1/8 = right shift results by 3
cmpeq d0,d1
 move.l #32,d1
 ift move.l #4,d15                                    ; if N=16 the tap weights are
                                                      ; 1/16 = right shift results by 4
cmpeq d0,d1
 nop
 ift move.l #5,d15                                    ; if N=32 the tap weights are
                                                      ; 1/32 = right shift results by 5
                                                      ; d15 holds the scale factor
;---------------Load the Scaled channel impulse response to memory-----------
               move.l #input_port,r5
               move.l #output_port,r15
               doensh0 #CPICH_FRAME_LEN*2
               move.l #Rx_CPICH,r0
               loopstart0
      fill_up:
                move.f (r5),d0                       ;load the Received CPICH in to memory
                move.f d0,(r0)+                      ; write the Rx_CPICH to memory
                loopend0
nop
;-------------------------------------------------------------------------
;             Channel Estimation starts from here
;-------------------------------------------------------------------------
 move.l #FIR_buffer,r2
 move.l #Rx_CPICH,r0
 dosetup0 fill_buffer
 move.l #N*4,n0
 doen0 #CPICH_FRAME_LEN/2          adda #256,r2,r2    ; r2->FRAME_START (64 previous symbols
                                                        ; are filled with zeros)
 tfra r2,r4
 adda #8,r0,r1                      suba n0,r4
 tfra r2,r7                          tfra r4,r9        ; r4->MA_START =FRAME_START- (N*4)
                                                       ; r2 & r7->FRAME_START
 adda #8,r2,r3                      tfra r4,r6         ; r6->MA_START

 loopstart0
fill_buffer:
  move.4f (r0)+,d0:d1:d2:d3         move.4f (r1)+,d4:d5:d6:d7
  [ add d0,d1,d0                    sub d0,d1,d1
    add d2,d3,d2                    sub d2,d3,d3
   ]
  [ add d4,d5,d4                    sub d4,d5,d5
    add d6,d7,d6                    sub d6,d7,d7
   ]
  moves.4f d0:d1:d2:d3,(r2)+        moves.4f d4:d5:d6:d7,(r3)+
 loopend0
```

```
;---------Now Satart the MA filter ------------------------
  dosetup0 MA_step_one doen0 #N/4                                ; loop to compute the first
                                                                 ; runing sum
  adda #8,r4,r5                    clr d8 clr d9                ; r5 -> MA_STRAT+2 symbols
  move.4f (r4),d0:d1:d2:d3          move.4f (r5),d4:d5:d6:d7

 loopstart0
MA_step_one:
  [ add d0,d4,d0                    add d1,d5,d1
    add d2,d6,d2                    add d3,d7,d3
    adda #16,r4,r4
   ]
  [ add d0,d2,d0                    add d1,d3,d1
    adda #16,r5,r5
   ]
  [ add d0,d8,d8                    add d1,d9,d9
    move.4f (r4),d0:d1:d2:d3        move.4f (r5),d4:d5:d6:d7
   ]
 loopend0
move.l #Ch_est,r8
  [ tfr d8,d6 tfr d9,d7             move.2f (r6)+,d0:d1
    move.2f (r7)+,d2:d3                                          ; set up r8->Ch_est
   ]
    asrr d15,d6                     asrr d15,d7
  [ moves.2f d6:d7,(r8)+            sub d0,d8,d8
    sub d1,d9,d9                    move.f #.5,d13                ; dummy load in d13 for
decimation
   ]
  [ add d2,d8,d8                    add d3,d9,d9
    move.f #.5,d12                                               ; dummy load in d12 for
decimation
   ]
;-------------------------------------------------------------
 move.l #SF_MATCH,d14
 tstge d14                                                  ; if SF_MATCH= -1 then decimate
 nop
 iff clr d14
 bt MA_filter_interp
 dosetup0 MA_filter_decimate        doen0 #CPICH_FRAME_LEN
  loopstart0
MA_filter_decimate
  [ tfr d8,d10                      tfr d9,d11
    tsteq d12                       move.2f (r6)+,d0:d1
    move.f #.5,d12
   ]
    asrr d15,d10                    asrr d15,d11
  [ sub d0,d8,d8                    sub d1,d9,d9
    move.2f (r7)+,d2:d3             moves.2f d10:d11,(r8)+
   ]
  [ iff clr d12                     suba #4,r8
    ifa add d2,d8,d8                add d3,d9,d9
   ]
  loopend0
 jmp _desimation_done
;-------------------------------------------------------------
MA_filter_interp
  dosetup0 MA_filter_interpolate    doen0 #CPICH_FRAME_LEN
  loopstart0
MA_filter_interpolate

  [ tfr d8,d10                      tfr d9,d11
    doensh1 d14
   ]
```

```
    move.2f (r6)+,d0:d1             move.2f (r7)+,d2:d3
    asrr d15,d10                    asrr d15,d11
      loopstart1
         moves.2f d10:d11,(r8)+
      loopend1
    sub d0,d8,d8                    sub d1,d9,d9
    add d2,d8,d8                    add d3,d9,d9

  loopend0

_desimation_done:

 suba n0,r7                         doensh1 #N/2
 nop
;-------------Copy the Last N Symbols to the fornt of the FIR_Buffer-----
   loopstart1
     move.4f (r7)+,d0:d1:d2:d3
     moves.4f d0:d1:d2:d3,(r9)+
   loopend1

 rts
```

4 Results

This section presents the DSP results and analysis. **Figure 6** shows the I and Q part of the estimated channel taps generated by the DSP and the actual channel taps (ideal) at some instant in time. In this example the Spreading factor is 32 (**SF_MATCH** = 8) and the channel conditions are as follows:

SNR=5dB; Path_delay_ns=[0 260 521 781]ns; power_dB=[0 -3 -6 -9]dB; v=120 Km/h.

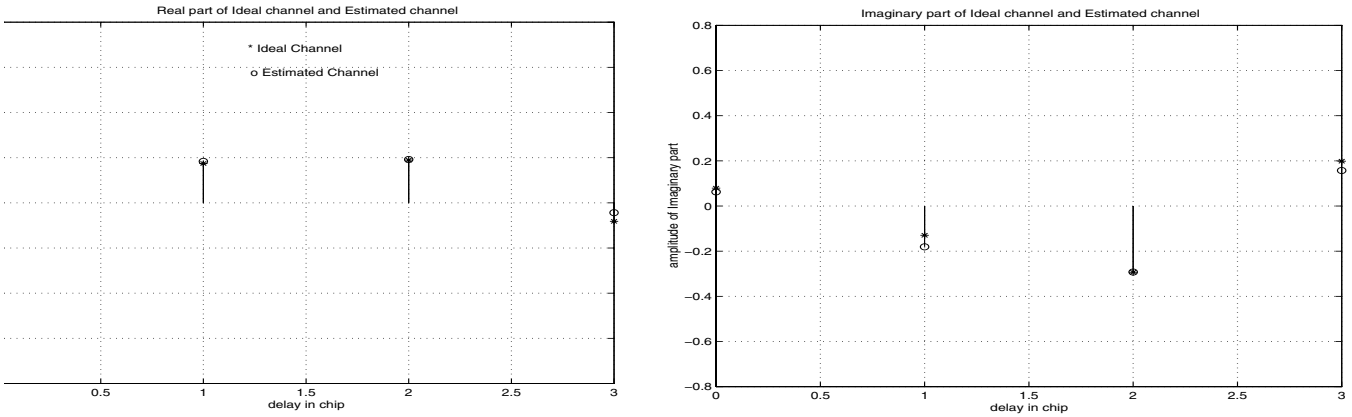
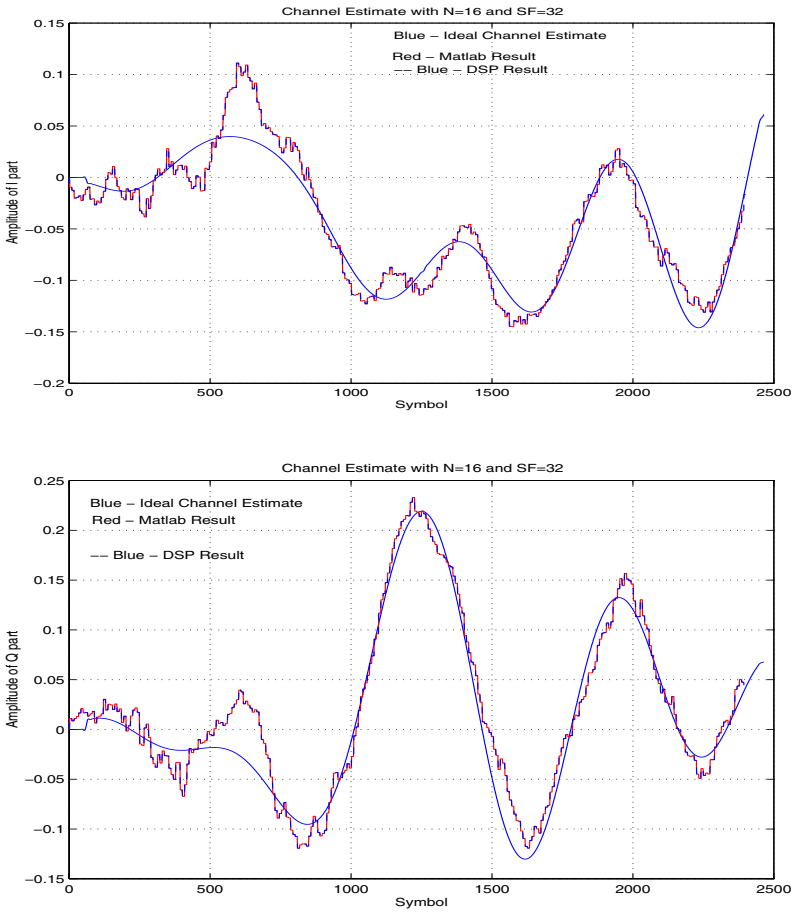


Figure 6. I and Q Part of Ideal and Estimated Channel Taps

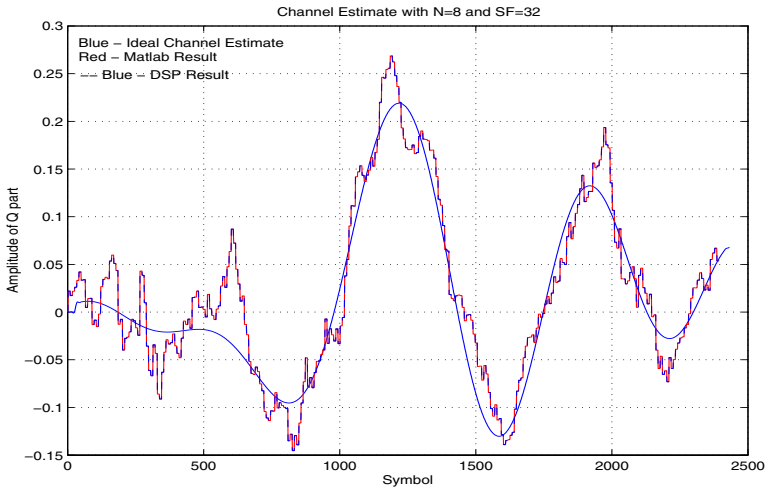
Figure 7 shows the tracking of the fading envelope for two frames of data for the weakest path of the following channel (no MAI is assumed). The results shown are the fading envelop and the estimate of the fade for two frames of data for the weakest path and following channel condition (no MAI is assumed).



Dashed Blue = StarCore SC140 Core
Solid Red = Matlab
Solid Blue = Perfect Channel Estimate for N=16, Spreading Factor = 32, and Mobile Speed = 120km/h

Figure 7. I and Q Part of the Channel Estimate

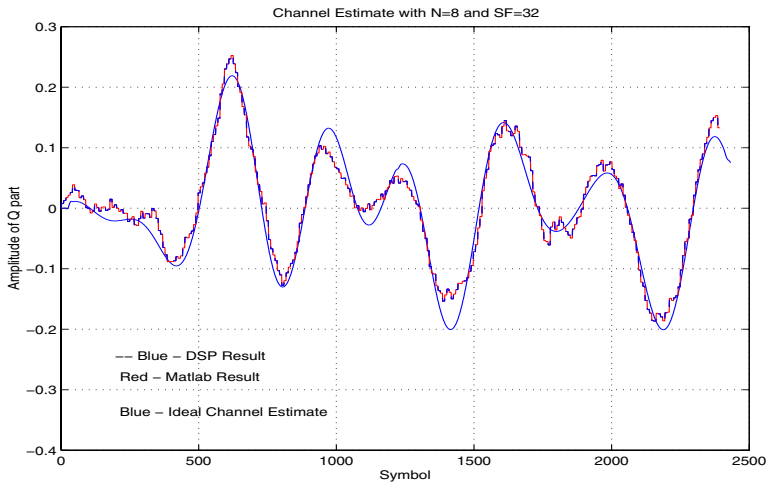
When the size of the MA filter is reduced from N=16 taps to N=8 taps, the channel estimate gets noisier as expected, but having fewer taps helps in taking the fast fading envelops (**Figure 8**).



Dashed Blue = StarCore SC140 Core
Solid Red = Matlab
Solid Blue = Perfect Channel Estimate for N=8, Spreading Factor = 32, and Mobile Speed = 120km/h

Figure 8. Reduced Size of MA Filer, Q of the Channel Estimate

Figure 9 shows the performance of the channel estimation algorithm in tracking a fast fading channel (for the same channel condition the mobile is traveling at twice the speed). The shorter the length of the MA filter, the better it tracks the fast fading channel. Parameters are as follows: SNR=5dB; Path_delay_ns=[0 260 521 781]ns; power_dB=[0 -3 -6 -9]dB; v=240; and km/h.



Dashed Blue = StarCore SC140 Core
Solid Red = Matlab
Solid Blue = Perfect Channel Estimate for N=8, Spreading Factor = 32, and Mobile Speed = 120km/h

Figure 9. Tracking a Fast-Fading Channel, Q of the Channel Estimate

The length of the MA filter is left as a variable for the designers choice. To estimate the fast fading channel, reducing the filter length helps in efficiently tracking the channel. The number of DSP cycles needed for channel estimation can be expressed as follows:

*DSP cycles for Interpolation = (SF_MATCH*150) + (7N/4) + 780*
DSP cycles for Decimation= (7N/4) + 780

5 References

[1] Holma, H. and A. Toskala, *WCDMA for UMTS Radio Access For Third-Generation Mobile Communications,* John Wiley & Sons, Inc., 2001.

[2] Mayer, Moeneclaey, Fechtel, *Digital Communication Receivers Synchronization, Channel Estimation and Signal Processing,* John Wiley & Sons, Inc., 1998.

NOTES:

NOTES:

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations not listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GMBH
Technical Information Center
Schatzbogen 7
81829 München, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
+800 2666 8080

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

AN2253
Rev. 2
11/2004

freescale™
semiconductor