IS-54 Digital Cellular Modem Implementation on the TMS320C5x

Application Report

Balaji Srinivasan
Digital Signal Processing Applications — Semiconductor Group

SPRA138
October 1994

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Introduction

Digital cellular and digital mobile radio communication are today's key topics in the communications field. Digital mobile cellular communication systems are being introduced in the U.S., Canada, Europe, Japan, and many other countries. Various standards like the U.S. Digital Cellular (USDC), Global System for Mobile Communications (GSM), and Personal Digital Cellular (PDC) have been proposed in different countries for the development of a mobile cellular communication system. The U.S. Digital Cellular standard is specified by the Telecommunications Industry Association (TIA). The TIA has specified $\pi/4$-DQPSK as the new modulation standard for the emerging U.S. digital cellular communication systems. The focus of this report is on the theory and implementation of the $\pi/4$-DQPSK modem on the TMS320C5x DSP. The TMS320 family of DSPs is well suited for such modem applications. The advanced features of the 'C5x have made the high-data-rate modem implementation possible. This report is organized into the following topics.

- Description of $\pi/4$-QPSK modulation scheme
- Theory of the $\pi/4$-DQPSK modem
- Modem implementation on the TMS320C5x
- Performance results
- Summary

The key features of the TMS320C5x that provide excellent code efficiency and ease of implementation are discussed in the *Modem Implementation on the TMS320C5x* section on page 119.

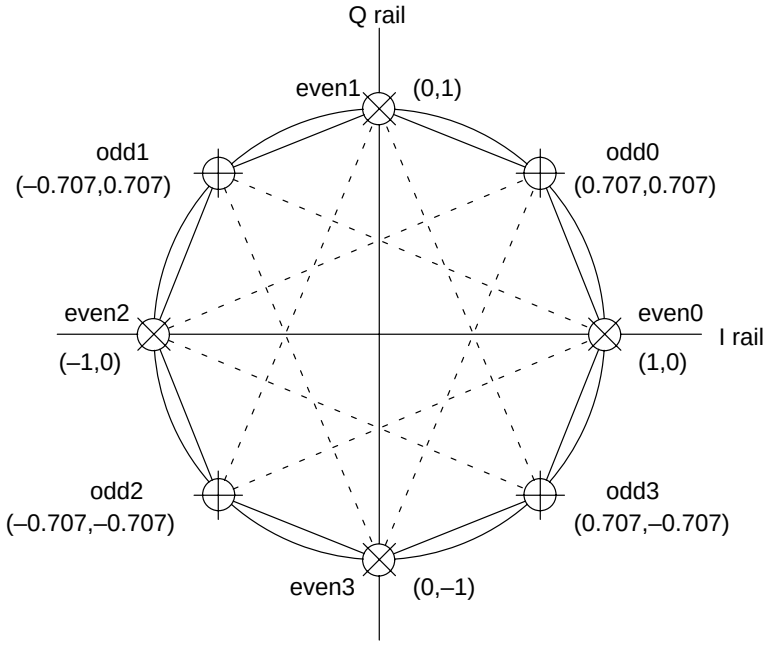
Description of $\pi/4$-QPSK Modulation Scheme

A study of various modulation schemes like QPSK, OQPSK, GMSK, and TFM have been made, and attention has been focused on the use of linear modulation techniques for nonlinearly amplified systems to meet both the power and spectral efficiency requirements of mobile cellular systems. There has been a search for alternative unstaggered linear modulation systems that have low envelope fluctuation. After a thorough analysis, $\pi/4$-QPSK was proposed as the standard modulation technique to be used in the digital cellular environment. $\pi/4$-QPSK is an unstaggered modified version of QPSK with two sets of constellations totaling eight constellation points. This modification to QPSK has carrier phase transitions that are restricted to $\pm\pi/4$ and $\pm 3\pi/4$. Since the phase does not undergo instantaneous $\pm\pi$ transitions as in QPSK, the envelope fluctuation at the output is significantly reduced. Also, as this is not a staggered/offset scheme, coherent as well as noncoherent detection can be applied to $\pi/4$-QPSK. It has been shown that the spectral efficiency obtained is twice that obtained by two-level digital FM, GMSK, or TFM, which are constant envelope modulation techniques.

The $\pi/4$-shifted-QPSK signal constellation can be viewed as the superposition of two QPSK signal constellations offset by 45° relative to each other, resulting in eight signal phases. Symbol phases are alternately selected from one of the two QPSK constellations, and as a result, successive symbols have a relative phase difference that is one of the four angles, $\pm\pi/4$ and $\pm 3\pi/4$. Figure 1 illustrates the $\pi/4$-shifted-QPSK signal constellation and the various possible phase transitions. As Figure 1 shows, two constellation sets, one with four possible phases (0, $\pi/2$, π, and $\pm\pi/2$) and the other with another four possible phases ($\pi/4$, $3\pi/4$, $-3\pi/4$, and $\pm\pi/4$) are used in the actual modulation. There is a relative $\pi/4$ shift between the two constellation sets; hence, the name $\pi/4$-shifted QPSK.

First, the input data is buffered into one of the four possible dibit symbols (namely, 00, 01, 10, or 11). Then, for odd numbered symbols, the output signal phase is chosen from one of four possible phases of the constellation set $\oplus$; for even numbered symbols, the output signal phase is chosen from one of four possible phases of the constellation set $\otimes$. The choice of the particular phase within a constellation set depends on the dibit input. As usual, to reduce dibit errors in the receiver, Gray coding of dibits is done prior to phase selection from a chosen constellation set. This alternate selection of a constellation set can be reversed for odd and even numbered symbols. In conventional QPSK, only one of the constellation sets is chosen. Due to the change of constellation sets in $\pi/4$-shifted QPSK, eight signal constellation points are possible. Although eight constellation points are seen in the constellation diagram and they look like the 8-PSK signal constellation, the choice of signal phases for every symbol is only four; hence, it is still a 4-phase QPSK. In conventional QPSK, the possible phase transitions were 0, $\pm\pi/2$, and π. Here, the possible phase transitions are only $\pm\pi/4$ and $\pm 3\pi/4$, thereby reducing the envelope fluctuations of the modulated output signal. Envelope fluctuations are very important since demodulation becomes difficult when the signal is amplified by nonlinear amplifiers (which are common in cellular systems). An OQPSK (offset QPSK) scheme reduces the fluctuations but restricts the type of demodulation scheme to be coherent. Noncoherent demodulation has certain advantages in the cellular systems, and $\pi/4$-shifted QPSK allows the flexibility to use either coherent or noncoherent demodulation. If differential encoding is also performed prior to signal mapping, the scheme becomes $\pi/4$ DQPSK.

Figure 1. $\pi/4$-Shifted QPSK Signal Constellation



Theory of the π/4-DQPSK Modem

Basic Modem Specifications

The specifications for the U.S. digital cellular modem were set by the TIA. A few of the specifications that are relevant to this application are:

- **Mode of operation**
  - 30-kHz channel structure, each channel operating on TDMA burst mode
  - Gross bit rate of 48.6 kbps
- **Modulation**
  - π/4-shifted differentially encoded quadrature phase shift keying
  - Gray coding used in signal mapping to reduce dibit errors
  - Spectral shaping to limit adjacent channel interference
  - No specific implementation method
- **Baseband filtering**
  - Square-root raised-cosine pulse-shape frequency response
  - Linear phase response
  - Roll-off factor for square-root pulse shaping filter to be 0.35
  - No specific implementation method
- **Demodulation**
  - Any coherent or noncoherent demodulation method
  - No carrier-related specifications (TMS320C5x implementation is a baseband modulation and demodulation)

Modulator

The theory behind signal mapping and baseband filtering for modulation is reproduced from the TIA document [7] here. The block diagram of the π/4-shifted DQPSK modulator is shown in Figure 2. The input 48.6-kbps data stream is converted into symbols as dibits A_k (odd bit) and B_k (even bit). Then the information is differentially encoded (symbols are transmitted as changes in phase between two successive symbols rather than as absolute phases) and mapped into one of the signal phases from either of the two signal constellations described in the *Description of π/4-QPSK Modulation Scheme* section on page 113. The symbols can be first differentially encoded and then mapped into a signal phase as a two-step process, or they can be combined into a single step with a set of equations. The digital data sequences A_k and B_k are encoded as I_k and Q_k according to the following set of equations.

$$I_k = I_{k-1} \cos[\ \Delta\phi(A_k, B_k)\ ] - Q_{k-1} \sin[\ \Delta\phi(A_k, B_k)\ ] \quad \textbf{(1)}$$

$$Q_k = I_{k-1} \sin[\ \Delta\phi(A_k, B_k)\ ] + Q_{k-1} \cos[\ \Delta\phi(A_k, B_k)\ ] \quad \textbf{(2)}$$

I_{k-1} and Q_{k-1} are the previous symbol's I and Q values. $\Delta\phi(A_k, B_k)$ is the phase change in the kth symbol interval and is determined according to Table 1. The phase change values are Gray coded.

Table 1. Phase Calculation

A_k	B_k	$\Delta\phi$	cos($\Delta\phi$)	sin($\Delta\phi$)
0	0	$+\pi/4$	+	+
0	1	$\pm\pi/4$	+	–
1	0	$+3\pi/4$	–	+
1	1	$-3\pi/4$	–	–

Simple trigonometric manipulation easily shows that Equations (1) and (2) are derived from

$$I_k = \cos[\phi_k] = \cos[\phi_{k-1} + \Delta\phi(A_k, B_k)] \quad \textbf{(3)}$$

$$Q_k = \sin[\phi_k] = \sin[\phi_{k-1} + \Delta\,\phi(A_k, B_k)] \quad \textbf{(4)}$$

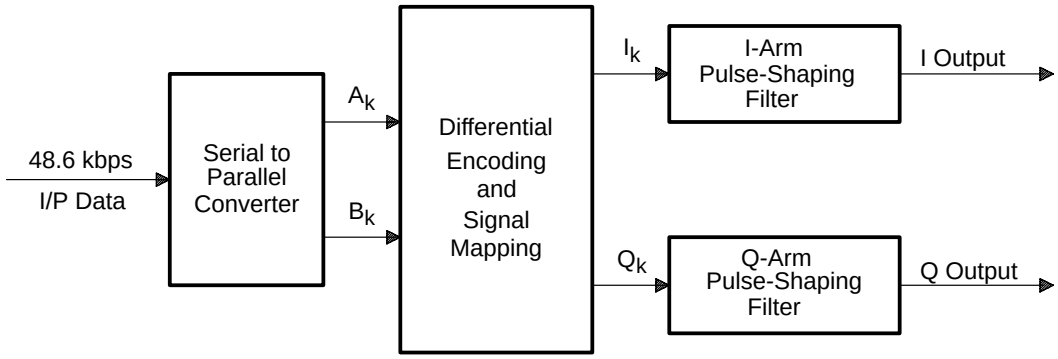
where ϕ_k and ϕ_{k-1} are the absolute phase angles corresponding to the kth and (k–1)th symbol intervals, respectively.

The signals I_k, Q_k at the output of the differential phase encoding block can take one of the five values 0, ± 1, or $\pm \frac{1}{\sqrt{2}}$, as seen from the constellation of Figure 1. Impulses I_k, Q_k are applied to the I and Q baseband pulse-shaping filters. The baseband filters have linear phase and square-root raised-cosine frequency response of the form:

$$|H(f)| = \begin{cases} 1 & : f \le (1-\alpha)/2T \\ \sqrt{0.5\left(1-\sin\left[\pi(2f-1)/2a\right]\right)} & : (1-\alpha)/2T \le f \le (1+\alpha)2T \\ 0 & : f > (1+\alpha)/2T \end{cases} \quad \textbf{(5)}$$

where T is the symbol period. The roll-off factor, α, determines the width of the transition band and is 0.35 as per the specifications.

Figure 2. Modulator Block Diagram



The baseband-filtered I and Q signals are then multiplied by the carrier and transmitted over the channel. The implementation on the TMS320C5x is a baseband modem, hence; the carrier is not included as part of the modulator block diagram.

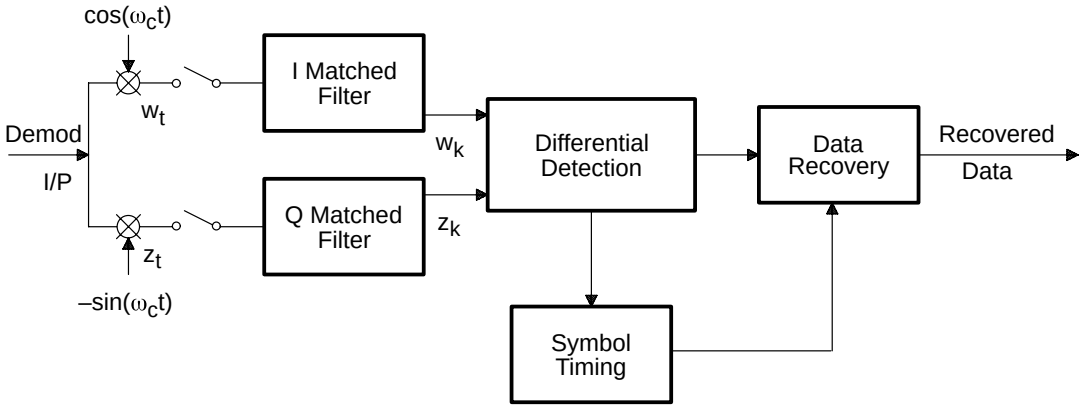
Demodulator

Digital communication systems that operate in power- and bandwidth-limited channels generally employ coherent detection that involves carrier-recovery technique. In a Rayleigh-faded mobile channel with

AWGN, coherent systems have a significant advantage in power efficiency and performance over the noncoherent demodulation involving differential or delay detection techniques. But in a mobile environment, disturbances such as multipath fading, Doppler frequency shifts, and phase noise are present. Coherent detection, which is based on the carrier frequency and phase lock, may suffer disadvantages over the noncoherent detection, though coherent detection has 3-dB power efficiency. Additionally, the noncoherent detection makes the receiver design simpler.

Since the current implementation on the 'C5x is a baseband modem that does not involve the carrier, and since noncoherent demodulation offers significent advantages, baseband differential detection has been chosen as the implementation technique on the 'C5x. Note that the TIA has not recommended any specific demodulation method.

Figure 3. Demodulator Block Diagram



The block diagram of the demodulator is shown in Figure 3. The theory of baseband differential detection [5] is discussed in the following sections.

Since no carrier multiplication is performed in this 'C5x implementation, the signals w_t and z_t are directly available at the demodulator without any cosine/sine multiplication. At this time, w_t and z_t are sampled, and the filtering, differential detection, data recovery, and symbol timing operations are performed. In this implementation, the samples w_k and z_k are made directly available to the demodulator in order to test the modem in the loop-back mode.

Filtering

The samples of w_t and z_t are passed through the matched filters in the receiver. Since the baseband I and Q signals at the transmitter are filtered by square-root raised-cosine pulse-shaping filters, the matched filters at the front end of the receiver are designed to give the same frequency response so that the combined receiver/transmitter response becomes raised cosine.

Differential Detection

Differential detection (delay and multiply) is performed with the filtered samples w_k and z_k according to the following equations.

$$w_k = \cos(\phi_k - \theta) \quad \text{and} \quad z_k = \sin(\phi_k \pm \theta) \tag{6}$$

where ϕ_k is the phase of the carrier at the sampling instant and θ is an arbitrary phase shift that is canceled in the differential operation.

After the detection operation:

$$x_k = w_k w_{k-1} + z_k z_{k-1} = \cos(\phi_k - \theta_{k-1}) \quad \textbf{(7)}$$

$$y_k = z_k w_{k-1} - w_k z_{k-1} = \sin(\phi_k - \theta_{k-1}) \quad \textbf{(8)}$$

where w_{k-1} and z_{k-1} are the one-symbol, time-delayed values of w_k and z_k, respectively.

Data Recovery

Equations (7) and (8) retrieve the phase change between two successive symbol intervals. Using Table 1 and the values of x_k and y_k, it is simple to decode the dibit information transmitted according to the following hard decision rule.

$$a_k = 0 \quad \text{if } x_k > 0 ; \; a_k = 1 \quad \text{if } x_k < 0 \quad \textbf{(9)}$$

$$b_k = 0 \quad \text{if } y_k > 0 ; \; b_k = 1 \quad \text{if } y_k < 0 \quad \textbf{(10)}$$

Symbol Timing

Symbol timing is one of the most important aspects of the demodulator because the hard-decision decoding has to be performed for data recovery in the appropriate sample so that, in the presence of noise, the recovered information is without error. In a TDMA environment where fast synchronization is required, differential detection is more advantageous, as it does not depend on the carrier recovery and phase lock in the beginning. The theory of symbol timing is based on a simple squaring/energy comparison technique [6]. Assuming four samples per symbol, the energy is calculated at every sample as

$$e = x_k^2 + y_k^2 \quad \textbf{(11)}$$

In the beginning of timing acquisition, an assumed mid-baud sample (say sample 3) is used for data recovery. Sample 2 and sample 3 energies are designated as e_p and e_n, respectively. At the assumed sample 3, the value of $e_n - e_p$ is calculated. The symbol timing is varied according to the following algorithm.

```
Let thresh = a threshold value ; counter = a count value
   begin:
      If | en – ep |  > thresh then goto ' correct '
      else goto ' done '
   correct: If  en – ep > 0 then goto ' checkm '
      else goto ' checkl '
   checkm: countl = 0
        If countm - counter = 0 then goto ' advance '
        else
          {  countm = countm + 1
             goto ' done '  }
   checkl: countm = 0
       If countl - counter = 0 then goto ' retard '
       else
         { countl = countl + 1
           goto ' done '}
   advance: " Process to advance the timing by one sample "
          goto ' done '
   retard: " Process to retard the timing by one sample "
   done:
```

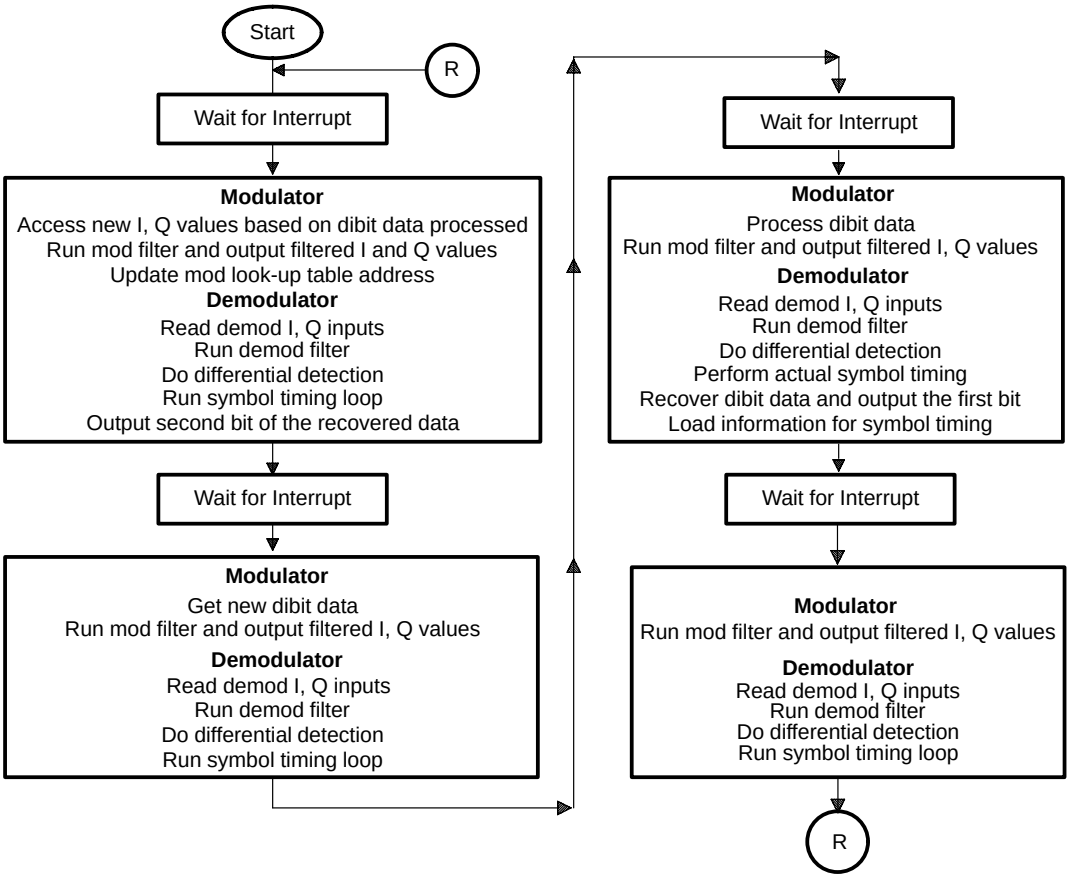
In this algorithm, the values of counter and threshold are initialized in the beginning to estimated values by trial and error. The value of counter can be kept small in the beginning of timing acquisition and later changed to a larger value so that the timing lock is maintained. This method is more stable with phase errors and small frequency shifts, as it does not depend on carrier recovery.

Modem Implementation on the TMS320C5x

Interrupt Organization

The data rate for the modem is 48.6 kbps, per the TIA specifications. The symbol rate for QPSK, then, is 24.3 kbaud/s, as every symbol comprises two bits. The number of samples/baud chosen is four, both for the modulator and demodulator. This means the baseband filters at the modulator need to generate at least four filtered samples/baud; hence, the minimum sampling frequency that is required is 97.2 kHz. The time available to complete the entire modem operation is quite critical, due to this high sampling frequency. For real-time operation, interrupts are generated at this rate. The consecutive interrupt routines are organized in a particular way for ease of implementation and code efficiency. Figure 4 details the operations performed in the consecutive interrupts.

Figure 4. Interrupt Organization

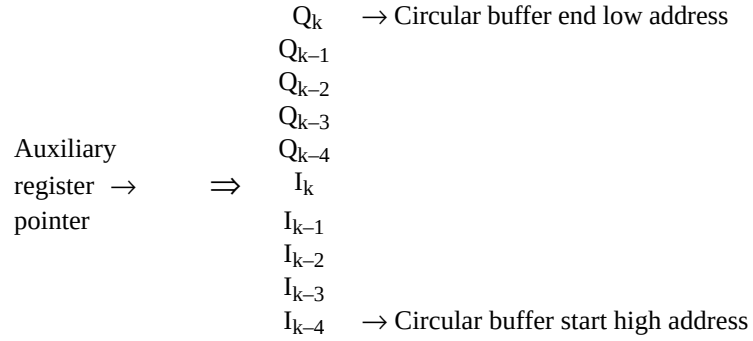


Modulator Implementation

Pulse-Shaping Filter

The pulse-shaping filters are designed using a commercial filter design package [9]. A 20-tap pulse-shaping FIR filter with a roll-off factor of 0.35 is designed, and the coefficients are stored in the program memory. The same set of coefficients are used for both I and Q filtering. The I and Q values (the filter inputs) do not change over a complete symbol period. This means once the modulator look-up table is read in the first interrupt, these values remain unchanged for the next three interrupts. Therefore, the interpolation technique is employed in filtering. An interpolation factor of 4 is achieved. Thus, the number of coefficients used in multiplication is reduced to 20/4 = 5. The number of filter delays used is also 5. This interpolation technique saves three-fourths of the time required to run the normal filter. The delays are updated once in four interrupts; specifically, in the interrupt just before the table look-up is done. The five I delays are immediately followed by five Q delays in the internal dual-access random-access memory (DARAM). The MADS instruction is used in the first three interrupts for multiply and accumulate. The MADD instruction is used in the fourth interrupt because the delays are also updated so that the new I and Q values can be loaded. The BMAR register is loaded with the appropriate address before the modulator filter is called in the main routine. Since it is an interpolation filter, the filter coefficients are rearranged in different blocks of five consecutive locations in program memory, so that the appropriate set of coefficients is used by filters in the four consecutive interrupts.

The modulator filter is implemented using one of the circular buffers in the ’C5x. The circular buffer is initialized in the beginning of the program as a decrement-type buffer. The circular buffer I and Q delays with the auxiliary register pointer (ARP) are as shown below.



The filter code is of the general format

```
rptz      #coefnum
mads/madd      *-
apac
sach
```

The preceding filter code does not involve overhead such as loading scaled filter inputs, loading the filter pointer with the appropriate address, etc. Both I and Q filtering are performed using a single circular buffer with contiguous filter delay locations.

The modulator circular buffer pointer points to location I_k at the start of the first interrupt. As shown in the modulator code, the new value of I_k is accessed from the look-up table and loaded, then decremented in such a way that it points to the Q_k location. The new Q_k value is then loaded and the pointer is modified so that it is reset to the start address. The BMAR register is loaded with the appropriate address so that the filter operates on the appropriate coefficients. The BMAR register allows the dynamic addressing for the filter instructions MADS and MADD. There is no data move involved in the filter for the first three interrupts. In the last interrupt, MADD is used so that a data delay creates the space for the next I and Q values.

Differential Encoding and Signal Mapping

As discussed in the *Modulator* subsection on page 115, Equations (1) and (2) implement differential encoding and signal mapping as a direct one-step process. Those equations can be further reduced and tabulated as shown in Table 2.

Table 2. Reduced Equations†

A_k	B_k	I_k	Q_k
0	0	sincos × $(I_{k-1} - Q_{k-1})$	sincos × $(I_{k-1} + Q_{k-1})$
0	1	–sincos × $(I_{k-1} + Q_{k-1})$	sincos × $(I_{k-1} - Q_{k-1})$
1	0	sincos × $(I_{k-1} + Q_{k-1})$	–sincos × $(I_{k-1} - Q_{k-1})$
1	1	–sincos × $(I_{k-1} - Q_{k-1})$	–sincos × $(I_{k-1} + Q_{k-1})$

† sincos = $\frac{1}{\sqrt{2}} = 0.707$

The following tables for odd and even symbols are generated from the equations in Table 2 and the naming pattern of the constellation in Figure 1. The values inside the parentheses (within the table entry) are the corresponding (I_k, Q_k) values. The column headings of the table represent A_k, B_k and the constellation point per the constellation of Figure 1.

Table 3. Odd-Symbol Look-Up

A_k	B_k	even0 (0)	even1 (1)	even2 (2)	even3 (3)
0	0	(+0.707, +0.707)	(–0.707, +0.707)	(–0,707, –0.707)	(+0,707, –0.707)
0	1	(+0.707, –0.707)	(+0.707, +0.707)	(–0,707, +0.707)	(–0,707, –0.707)
1	0	(–0.707, +0.707)	(–0,707, –0.707)	(+0,707, –0.707)	(+0,707, +0.707)
1	1	(–0.707, –0.707)	(+0,707, –0.707)	(+0,707, +0.707)	(–0,707, +0.707)

Table 4. Even-Symbol Look-Up

A_k	B_k	even0 (0)	even1 (1)	even2 (2)	even3 (3)
0	0	(0,1)	(–1 ,0)	(0, –1)	(1, 0)
0	1	(1,0)	(0 ,1)	(–1, 0)	(0, –1)
1	0	(–1 ,0)	(0, –1)	(1, 0)	(0, 1)
1	1	(0, –1)	(1, 0)	(0, 1)	(–1, 0)

Modulator Look-Up Table

The constellation points are named 0, 1, 2, and 3, whether for an odd symbol or an even symbol. The odd symbols and even symbols are designated as sym0 and sym1, respectively, and pcn stands for previous constellation. For example, pcn0sym0 means that the previous constellation was numbered 0 and the present symbol is odd. The organization of the table is as follows.

Table 5. Modulator Look-Up

Address Naming	Description	Table Entry
Lookup Table Main Base Address		
Set1's Base Address: pcn0sym0	A_k0B_k0	I_k value
		Q_k value
		Next Address†
	A_k0B_k1	- do -
	A_k1B_k0	- do -
	A_k1B_k1	- do -

† This value is the next set's base address for the new symbol, and it is calculated relative to the look-up table's main base address.

There are eight sets of table entries as shown above (pcn0sym1, pcn1sym0, etc.) with each set having entries for A_k0B_k0, A_k0B_k1, A_k1B_k0, and A_k1B_k1, and each A_kB_k having three entries, totaling 96 entries.

Updating the Look-Up Table

An ARP pointer (for example, ar4) is used to point to the look-up table address. The following code excerpt gets new values for I_k and Q_k from the modulator look-up table and updates the table address pointer.

Modulator Table Manipulation Code

```
Interrupt
1st:    lmmr   bmar,#bmar1       ; bmar reg = base address of 1st set
                                 ; interpolation coeffts in prog mem
        lacc   *+, 13, ar2       ; load ik value
        sach   *-, 0, ar4        ; store in filter's ik input location,
        lacc   *+, 13, ar2       ; load qk value
        sbrk   #coefnum          ; sub coef. no (19) to point the qk
                                 ; filter input location
        calld  Mod_fltr, *, ar2  ; call delayed filter
        sach   *-                ; store acc in filter's qk i/p location
        zap                      ; clear acc. & p reg.
        mar    *,ar4             ; after filtering, arp=ar4
        lar    *,ar4,ar5         ; ar4=next set's base address
2nd:    lacl   *+                ; load 1st bit of dibit data
        sacl   data              ; store in var "data"
        lacl   *+                ; load 2nd bit of dibit data
        sacl   data1             ; store in var "data1"
        lmmr   bmar, #bmar2      ; bmar=interpolation coeff. address
        calld  Mod_fltr, *, ar2  ; call delayed filter
        zap                      ; clear acc. & p reg.
        nop                      ; nop to fill up delayed call
3rd:    lacl   scrdata           ; load 1st scrambled bit
        nop                      ; no operation
        xc     1, gt             ; if that bit is a 1 execute foll ins'n
        adrk   #6                ; add 6 to lookup table pointer
        lmmr   bmar, #bmar3      ; bmar=interpolation coeff. address
        calld  Mod_fltr, *, ar2  ; call delayed filter
        dmov   data1             ; move data1 into data
        zap                      ; clear acc. & p reg.
4th:    lacl   scrdata           ; load 2nd scrambled bit
        nop                      ; no operation
        xc     1, gt             ; if that bit is a 1 execute foll ins'n
        adrk   #3                ; add 3 to lookup table pointer
        lmmr   bmar, #bmar4      ; bmar=interpolation coeff. address
        calld  Mod_fltr, *, ar2  ; call delayed filter
        zap                      ; clear acc. & p reg.
        nop                      ; nop to fill up delayed call
```

In the first interrupt, after the I and Q values are accessed and loaded into the appropriate filter input locations, the filter is executed. The pointer ar4 now points to the location where the next set's base address is available, and this address value is loaded into ar4. In the second interrupt, the new dibit data is read. In the third interrupt, ar4 is incremented by 6 if the first bit of the dibit data is a 1; otherwise it is unchanged. In the last interrupt, ar4 is incremented by 3 if the second bit of the dibit data is a 1; otherwise it is unchanged. This way, ar4 is modified so that it points to the appropriate subset base address in the set chosen in the first interrupt.

The differential encoding and signal mapping only takes three cycles (max) for the ARP modification in any interrupt. The modulator code is found to be highly efficient with this implementation. This is made possible with the powerful features of the 'C5x. The circular buffer feature enables absolute zero-overhead filtering. Dynamic addressing with MADS and MADD makes interpolation filtering easier. Single-cycle decision-making instructions like XC make look-up table pointer modification simpler. The instructions for delayed call, return and branching, and special instructions like ZAP and RPTZ reduce the various branch overheads.

Demodulator Implementation

The demodulator performs I and Q matched filtering, differential detection, data recovery, and symbol timing. Unlike the modulator, the operations performed by the demodulator in four interrupts are the same except for the symbol timing loop.

Input Filtering

The input I and Q matched filters have square-root raised-cosine frequency response. They are 20-tap FIR pulse-shaping filters similar to the modulator. But these filters cannot be implemented as interpolation filters because the sampled I and Q values are always different. Again, four samples/baud are chosen for the demodulator implementation. The I and Q filters are implemented using the second circular buffer, similar to the modulator I and Q circular buffer. The only difference is that MACD is used instead of MADD or MADS because the inputs are updated with every interrupt.

Differential Detection

Every time the demodulator filter is executed, the filtered I_k sample is made available in the accumulator buffer ACCB, and the filtered Q_k sample is made available in the accumulator. This format is used for code-efficient differential detection. The accumulator buffer feature of the ’C5x is very useful as an accumulator backup, and data transfer between the accumulator and its buffer enhances its usage. Differential detection and energy calculation are performed by the following short code excerpt.

```
sach    zk1, 2          ; immly after i/p filtering,
                        ; store acc. in wk1
lacb                    ; load acc. with acc. buffer
sach    wk1, 2          ; store it in wk1
lt      wk1             ; t reg = wk1
mpy     wkp1            ; p reg = wk1.wk1–1
ltp     zk1             ; t reg = zk1, acc = wk1.wk1–1
mpy     zkp1            ; p reg = zk1.zk1–1
mpya    wkp1            ; p reg = zk1.wk1–1
                        ; acc = wk1.wk1–1 + zk1.zk1–1
sach    xk              ; store acc. in xk
ltp     wk1             ; t reg = wk1, acc = zk1.wk1–1
mpy     zkp1            ; p reg = wk1.zk1–1
sqrs    xk              ; p reg = xk2,
                        ; acc = zk1.wk1–1 – wk1.zk1–1
sach    yk              ; store acc. in yk
lacc    #zero           ; clear acc.
sqra    yk              ; p reg = yk2
apac                    ; acc = xk2 + yk2
sach    energy          ; store acc. in energy
lacc    addr            ; load symbol timing address
calad                   ; call delayed with address in acc.
dmov    wk1             ; move wk1 into wk1–1
dmov    zk1             ; move zk1 into zk1–1
```

Notice from Equations (7) and (8) that every new filtered sample w_k and z_k is multiplied by $w_{k–1}$ and $z_{k–1}$, which are one symbol (that is, four samples) delayed. The segregation of interrupts facilitates efficient implementation. There are four sets of w_k, $w_{k–1}$ and z_k, $z_{k–1}$ used for four interrupts. As far as the first interrupt is concerned, w_{k1} and z_{k1} are the current filtered I and Q values and w_{kp1} and z_{kp1} are the one-symbol delayed values. Similarly, w_{kp2} and z_{kp2} are the one-symbol delayed values for the second interrupt, and so on. Hence, after performing the differential detection and energy calculation, two DMOV instructions move w_{k1} and z_{k1} into w_{kp1} and z_{kp1} to be used next time in that particular interrupt. The w_{kp} and z_{kp} values are allocated proper memory locations to perform this. Once differential detection is done, the symbol timing loop is called using CALAD, accommodating the two DMOV cycles. The main differential detection and energy calculation takes just 17 cycles.

Symbol Timing

Symbol timing is performed using a program address jump with instruction CALA. The organization of the symbol timing loop is as follows. The variable Addr is initialized to Sample1 at the beginning of the program.

```
Sample1  : Output the second bit of the recovered dibit
           information ;
           Addr = Sample2 ;

Sample2  : energy_prev = energy ;
           Addr = Sample3 ;

Sample3  : energy_next = energy ;
           Recover dibit data and output first bit of the
           dibit information
           Run symbol timing algorithm
           If no correction : Addr = Sample4
           If advance correction : Addr = Sample1
           If retard correction : Addr = Sampld

Sample4  : Addr = Sample1

 Sampld  : Addr = Sample4
```

As seen, if the timing is to be advanced, one sample is skipped. If the timing is to be delayed, one extra dummy sample address jump is inserted.

Performance Results

The performance of the modem implemented on the ’C5x under the AWGN environment is summarized here.

Theory

The theory of noise generation and addition is as follows. Note that VAR( ) and Std ( ) represent the variance and the standard deviation functions.

$$SNR_{dB} = 10\log_{10}\left(\frac{Signal\ Power}{Noise\ Power}\right) = 10\log_{10}\left(\frac{Var(signal)}{Var(noise)}\right) \quad \textbf{(12)}$$

$$\therefore\ Var(noise) = \frac{Var(signal)}{10^{\left(SNR_{dB}/10\right)}} \quad \textbf{(13)}$$

For I & Q arms:

$$Var\_i = \frac{Var(I\ signal)}{10^{\left(SNR_{dB}/10\right)}} = \frac{[Std(I\ signal)]^2}{10^{\left(SNR_{dB}/10\right)}} \quad \textbf{(14)}$$

$$Var\_q = \frac{Var(Q\ signal)}{10^{\left(SNR_{dB}/10\right)}} = \frac{[Std(Q\ signal)]^2}{10^{\left(SNR_{dB}/10\right)}} \quad \textbf{(15)}$$

Also

$$Std\_i = \sqrt{Var\_i}\ ;\quad Std\_q = \sqrt{Var\_q} \quad \textbf{(16)}$$

Two independent Gaussian-distributed random-noise sequences, I_noise[k] and Q_noise[k], are generated using the Matlab software. The noise is added to the I and Q signals as shown below.

$$I\_noise[k] = I\_noise[k] \times Std\_i \quad \textbf{(17)}$$

$$Q\_noise[k] = Q\_noise[k] \times Std\_q \quad \textbf{(18)}$$

$$Id[k] = I[k] + I\_noise[k] \quad \textbf{(19)}$$

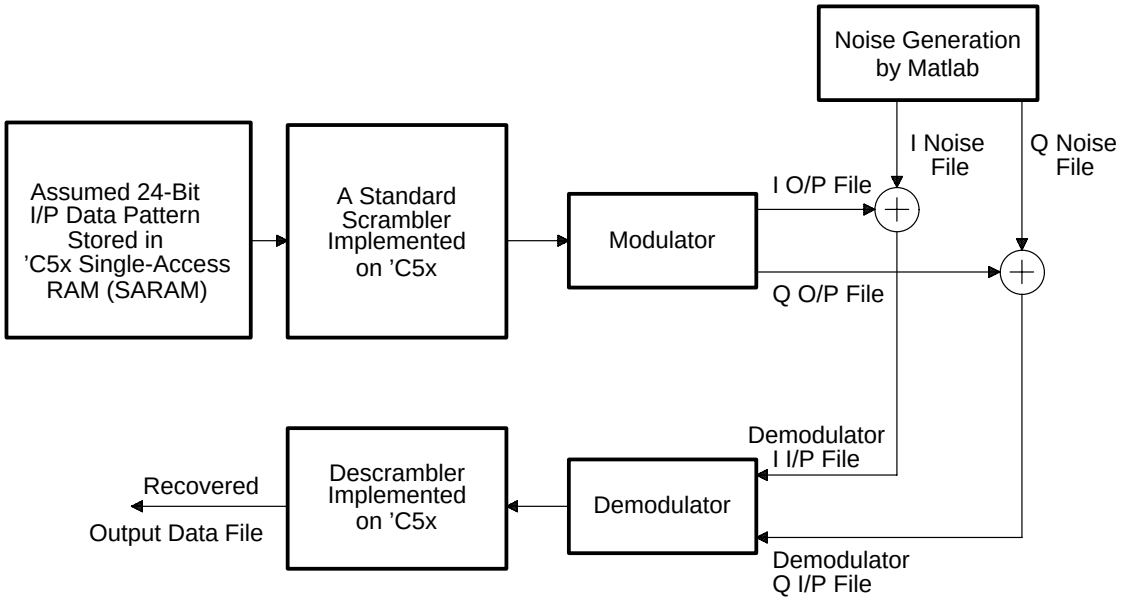
$$Qd[k] = Q[k] + Q\_noise[k] \quad \textbf{(20)}$$

I_d and Q_d are the two new demodulator input points that are generated using Matlab.

Testing

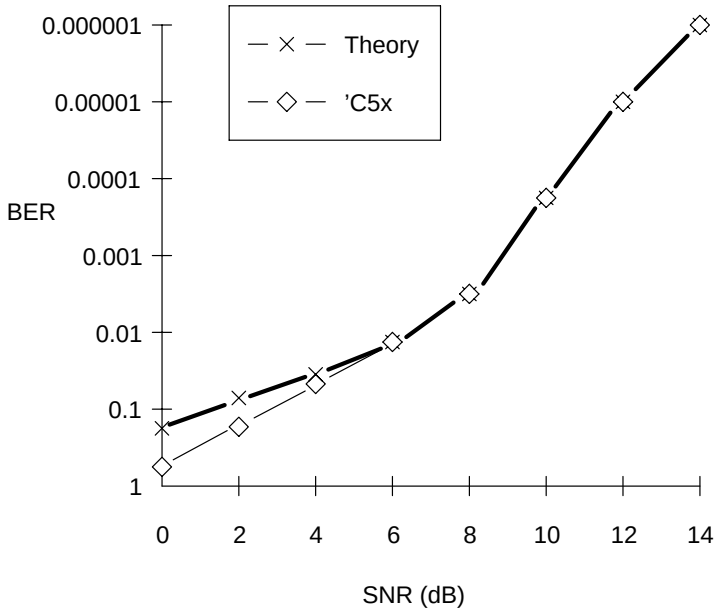
The modem implementation is tested using a file I/O scheme in which the modem runs on a ’C5x EVM card and communicates with the host PC for a nonreal-time file transfer. The testing is performed with the setup shown in Figure 5.

Figure 5. Modem Test Configuration



As shown in Figure 5, the assumed input data pattern is scrambled in the ’C5x to generate randomness in the input data. About 60,000 samples each of I and Q are generated by the modulator and stored in files. The ’C5x EVM talks to the PC through DSP-PC interface software for file transfer. The I and Q noise files are generated by Matlab and added with modulator output files. The demodulation and descrambling is done, and recovered data of 30,000 bits is stored in a file. The number of errors in the demodulator output file are counted in trials with various SNR values. The modem performance for the AWGN channel is shown in Figure 6.

Figure 6. BER Versus SNR for a Static AWGN Channel



Performance

As seen in Figure 6, the performance in an AWGN environment closely follows the theoretical performance. Since this is a cellular modem, its performance also needs to be tested under fading conditions with Doppler shifts due to vehicle speed. The performance of this modem under such conditions is expected to be of moderate standards because the implementation involves restrictions such as fewer samples/baud, etc. The performance could be improved by employing more samples/baud, a sophisticated symbol timing scheme, an automatic AGC, and an equalizer at the front end of the demodulator. As the number of cycles taken by the entire modem function is about 160, other extra features listed above could be accommodated to improve the performance under fading and Doppler-shift conditions.

Speed and Memory Requirements

Table 6 lists the number of 'C5x program and data memory words required by the core modulator/demodulator algorithm. It also provides the maximum number of cycles needed to run the modulator and demodulator. The hardware, I/O interface, and program initialization requirements are not included here, as they do not fall within the time-critical loop of the modem implementation. Note that this table does not include the interrupt handling overheads.

Table 6. Program Memory and Speed Requirements

Module Name	Program Memory	Data Memory	Cycles (Max)
Modulator	76 + 116†	114	32
Demodulator	246 + 20†	68	126

† This is the size of program memory used for loading tables, etc.

The maximum number of words and cycles used by the various modules of the modulator and demodulator, including the different overheads, are shown in the following tables.

Table 7. Modulator Code Size and Execution Time

Module Name	Size in Words	Cycles (Max)
Mod_Main	55 + 96†	13
Mod_Fltr	21 + 20†	19

† This is the size of program memory used for loading tables, etc.

Table 8. Demodulator Code Size and Execution Time

Module Name	Size in Words	Cycles (Max)
Dmd_Fltr	11 + 20†	52
Dmd_Main	124	28
Sym_Time	111	46

† This is the size of program memory used for loading tables, etc.

As the above tables show, both the modulator and demodulator have been well optimized to accommodate future addition of modules, if necessary, for performance improvements. There is also a large portion of unused internal RAM for future memory requirements.

Summary

The IS-54 U.S. digital cellular modem concepts are introduced and the theory of π/4-QPSK with signal constellation is discussed. The modem implementation on the TMS320C5x is explained and the performance of the modem with AWGN is summarized. Also, the requirements of the modem regarding speed and memory are tabulated. The efficiency and capabilities of the TMS320C5x for the high-bit-rate cellular modem application are clearly visible from the modem implementation. This implementation needs to be further studied under Rayleigh fading with co-channel interference and Doppler shift. Improvements for the demodulator are suggested. The modem program is made highly modular and is developed according to the TI Communication Software Library (CSP) developer's guidelines[1]. This π/4-QPSK cellular modem implementation on the TMS320C5x family of DSPs provides guidelines for cellular-systems designers to employ in using the 'C5x DSP for all cellular and related applications.

Code Availability

The associated program files are available from Texas Instruments TMS320 Bulletin Board System (BBS) at (713) 274-2323. Internet users can access the BBS via anonymous ftp at *ti.com.*

1 Refer to "Software Coding Guidelines for 'C5x Developers", in this book.

References

1. Lee, W. C.Y., *Mobile Cellular Telecommunications Systems*, McGraw-Hill, 1989.

2. Proakis, J.G., *Digital Communications*, McGraw-Hill, 1989.

3. Crochiere, R.E., and Rabiner, L.R., *Multirate Digital Signal Processing*, Prentice-Hall, Inc., 1983.

4. Chennakeshu, Sandeep, and Saulnier, Gray J., "Differential Detection of $\pi/4$-Shifted-DQPSK for Digital Cellular Radio", *IEEE Transactions on Vehicular Technology,* Vol. 42., No. 1, February 1993.

5. Feher, Kamilo, "MODEMS for Emerging Digital Cellular-Mobile Radio System", *IEEE Transactions on Vehicular Technology*, Vol. 40, No. 2, May 1991.

6. Troullinos, George, et al., "Theory and Implementation of a Splitband Modem Using the TMS32010", *Digital Signal Processing Applications with the TMS320 Family*, Vol. 2, Prentice-Hall, Inc, 1991.

7. *Cellular System: Dual-Mode Mobile Station – Base Station Compatibility Standard*, IS-54 Project Number 2215, Electronic Industries Association, December 1989.

8. *TMS320C5x User's Guide*, Texas Instruments, 1990.

9. *DFDP3/Plus Digital Filter Design Package Release 1.00*, Atlanta Signal Processors, Inc., 1992.

10. *MATLAB User's Guide*, The Math Works, Inc., 1989.

11. Chishtie, Mansoor A., "Software Coding Guidelines for 'C5x Developers", *Telecommunications Applications With the TMS320C5x DSPs Application Book*, Texas Instruments, 1994, pp. 247–258.

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.