Application Report
SPRA297 - November 2002

Extended Precision Radix-4 Fast Fourier Transform Implemented on the TMS320C62x

Robert Matusiak *Digital Signal Processing Solutions*

ABSTRACT

This application report discusses a method by which the Texas Instruments TMS320C62x™ high-performance, fixed-point digital signal processors (DSPs) overcome the traditional advantage held by floating-point DSPs – precision and speed.

Using the Radix-4 Fast Fourier Transform (FFT), this document illustrates how extended precision arithmetic, multiplication in particular, can be performed on the C62x™. Using the techniques outlined here, the 16-bit multipliers of the C62x can exceed the performance of the 32-bit floating-point arithmetic logic units and multipliers found in floating-point DSPs.

List of Figures

The TMS320C62x generation of high-performance fixed-point DSPs features two independent 16-bit multiplier units. Because the multiplier units have been designed primarily for the processing of 16-bit data, special consideration must be made for implementing algorithms that require multiplication of numbers with precision of greater than 16 bits. This spplication report uses the Radix-4 Fast Fourier Transform (FFT) as an example of how extended precision arithmetic, multiplication in particular, can be performed on the C62x. It is in the findings of this exercise that the C62x can exceed the performance of floating-point DSPs using the techniques outlined.

Typically, the two most notable advantages that a floating-point DSP has over a fixed-point DSs is precision and range. Most floating-point DSPs feature 32-bit floating-point arithmetic logic units (ALUs), multipliers, and a 32-bit register file, whereas most fixed-point DSPs feature 16-bit integer units and a register file. In addition, the floating-point arithmetic units feature hardware that allows numbers to be represented in a wider range than in fixed-point. Floating-point units have the ability to automatically scale numbers. For example, in a 16-bit fixed-point adder unit if we added two numbers together producing a result that was larger than 16-bits, an overflow would occur, and the result would be erroneous. Whereas, a floating-point adder would detect the condition, scale the number, and move the decimal point to the right. In effect, the floating-point adder gains precision in the integer portion of the result, and loses precision from the fraction portion of the result.

Because the C62x DSP features 32-bit ALUs and a 32-bit register file, the precision of floating-point DSPs can be easily achieved. The only stumbling block is that the C62x features a 16-bit multiplier. However, we will show how the C62x can easily perform a 32-bit multiply.

The C62x can perform extended precision multiplication by performing several 16-bit multiplies. In the case of performing a 32-bit multiply, four 16-bit multiplies and some additional arithmetic are required. Let's take a look at how we can multiply two 32-bit numbers, A times B, using 16-bit multiplies. Figure 1 pictorially describes how this would be performed. Note that in the multiplies the u and s to the right and/or left of the multiplication symbol indicate whether the operand is signed or unsigned. Also, it should be noted that a 32-bit multiply generates a 64-bit result. In the example, we keep the most significant, or upper 32-bits. Figure 2 shows a listing of a C function for an extended precision multiply. Figure 3 shows a listing of a C62x C-callable assembly function for an extended precision multiply.

Figure 4 contains the source listing for an extended precision radix-4 FFT implemented as a C62x C-callable assembly language function. This implementation executes a 1024 point radix-4 FFT in 704 usec. Using the 32-bit multiplication technique, we can see that the C62x can perform computations with the precision comparable to 32-bit floating-point processors, at a performance greater than most floating-point processors.

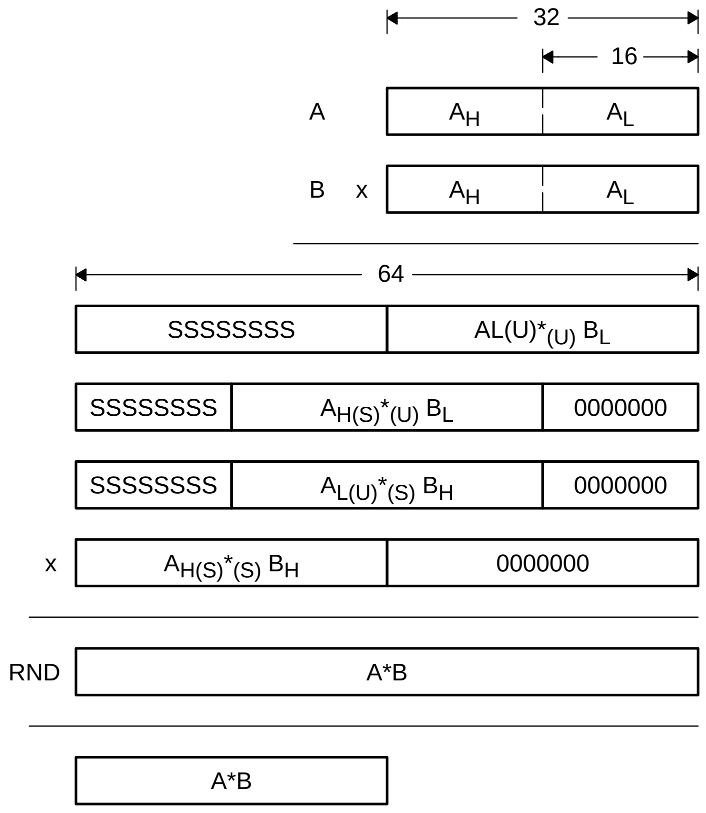

Figure 1. 32-Bit Multiplication Using 16-Bit Multiplies

```
/***********************************************************************
   dpmpy() – function used to multiply two signed 32-bit integers and
   return the most significant 32-bits of the result.
***********************************************************************/
int dpmpy(int A, int B)
{
     int AhBh, AhBl, AlBh, AlBhH, AhBlH;
     unsigned int AlBl, AhBlL, AlBhL, ABLl;
     short Ah, Bh;
     unsigned short Al, Bl;
     long ABL;
     int ABLov, ABH;
     Ah = A>>16;
     Bh = B>>16;
     Al = A & 0x0000FFFF;
     Bl = B & 0x0000FFFF;
     AhBh = Ah * Bh;
     AlBl = Al * Bl;
     AlBh = Al * Bh;
     AhBl = Ah * Bl;

     AhBlH = AhBl >> 16;
     AhBlL = AhBl << 16;
     AlBhH = AlBh >> 16;
     AlBhL = AlBh << 16;
     ABL = AlBl + AlBhL;
     ABL = ABL + AhBlL;
     ABLov = (int)(ABL >> 32);
     ABLl = (unsigned int)(ABL & 0xffffffff);
     ABH = AhBh + AhBlH + AlBhH + ABLov;
     return(ABH<<1);
}
```

Figure 2. C-Source Listing for an Extended Precision Multiply

```
;*******************************************************************************
; dpmpy.asm – C6x assembly source code for a fixed-point double precision
; multiply C-callable assembly language function. The functions take 2 32-bit
; signed integers and performs a 32-bit by 32-bit multiply which produces a
; 64 bit product. The upper 32-bits of the product are returned as a signed
; integer. The C6x CPU core has 61-bit multipliers, thus 4 16-bit multiplies
; is required to realize a 32-bit multiply.
;*******************************************************************************
; PROTOTYPE
;
;      int dpmpy(int, int);
;
;*******************************************************************************
; USAGE
;
;      int A, B, prod;
;
;      prod = dpmpy(A,B);
;
;*******************************************************************************


       .global _dpmpy

A              .set     a4
B              .set     b4
AlBl           .set     a1
AhBl           .set     b1
AlBh           .set     a2
AhBh           .set     b2
AlBhH          .set     a3
AhBlH          .set     b7
AlBhL          .set     a8
AhBlL          .set     b6
ABH            .set     b5
return         .set     a4
ABLo           .set     a7
ABLe           .set     a6

_dpmpy:

       mpyhslu .m1x     B,A,AlBh           ; Al u*s Bh
||     mpyhslu .m2x     A,B,AhBl           ; Ah s*u Bl

       mpyu    .m1x     A,B,AlBl           ; Al u*u Bl
||     mpyh    .m2x     B,A,AhBh           ; Ah s*s Bh

       shr     .s1      AlBh,16,AlBhH      ; AlBhH = AlBh >>s 16
||     shr     .s2      AhBl,16,AhBlH      ; AhBlH = AhBl >>s 16

       b       .s2      b3                 ; return

       shl     .s1      AlBh,16,AlBhL      ; AlBhL = AlBh << 16
||     shl     .s2      AhBl,16,AhBlL      ; AhBlL = AhBl << 16
||     add     .l2      AhBh,AhBlH,ABH     ; ABH = AhBl + AhBlH


       add     .l2x     ABH,AlBhH,ABH          ; ABH = ABH + AlBhH
||     addu    .l1x     AlBl,AhBlL,ABLo:ABLe   ; (long)ABL = AlBl + AhBlL

       addu    .l1      AlBhL,ABLo:ABLe,ABLo:ABLe
                                               ; (long)ABL = AlBhL + (long)ABL
       add     .l1x     ABLo,ABH,return        ; ABH = ABLhigh + ABH

       shl     .s1      return, 1, return
```

Figure 3. C62x Assembly Listing for a C-Callable Extended Precision Multiply

```
;*****************************************************************************
; FILE
;
; dpfft.asm – C6x assembly source code for a C-callable, double precision
; fixed-point, radix-4, in-place, complex FFT assembly language function.
;
;*****************************************************************************
; PROTOTYPE
;
;      void dfft (int *x, int *y, int *wcos, int *wsin, int n, int m);
;
;      where:  x is a pointer to the real data array of length n
;      y is a pointer to the imaginary data array of length n
;      wcos is a pointer to the real twiddle factors array of length n
;      wsin is a pointer to the imaginary twiddle factors array, length n
;      n is the number of data points (must be a power of 4)
;      m is the number of stages in the radix-4 FFT
;
;*****************************************************************************
; PERFORMANCE
;
;      ~ # of cycles = M * (N/4 * 54 + 37) + N/3 * 7
;
;*****************************************************************************
; MEMORY REQUIREMENTS
;
;      4*N bytes (real data)
;      4*N bytes (imag data)
;      4*N bytes (real coefficents)
;      4*N bytes (imag coefficents)
;      200 bytes (stack)
;      ----------------------------
;      16*N + 200 bytes (Total)
;
;*****************************************************************************
; ASSUMPTIONS
;
;      1) All data is assumed to be in on-chip data memory
;      2) Digit reversal is not performed
;      3) Further optimization could improve performance
;
;*****************************************************************************


xaddr          .set     a4
yaddr          .set     b4
wcaddr         .set     a6
wsaddr         .set     b6
npoints        .set     a8
mstages        .set     b8

STACKSIZE      .set     200
N2             .set     1
E              .set     2
A              .set     3
B              .set     4
C              .set     5
```

Figure 4. C62x C-Callable Assembly Language Functin Source Listing for an Extended Precision Radix-4 FFT

```
I0             .set     6
I1             .set     7
I2             .set     8
I3             .set     9
R1             .set     10
R2             .set     11
S1             .set     12
S2             .set     13
N1             .set     14
N              .set     15
K              .set     16
J              .set     17

stack          .set     b7
KCNT           .set     b0
JCNT           .set     b1
ICNT           .set     b2
n1             .set     b2
n2             .set     b3
e              .set     b5
ea             .set     a5
j              .set     b9
ja             .set     a9
m              .set     a10
n              .set     b8
nt             .set     b1
i0t            .set     b0
i1t            .set     b4
n1t            .set     b4
r4             .set     a4
s4             .set     b4
co1            .set     a8
r3             .set     a1
co1hr3l        .set     a2
co1lr3h        .set     a3
co1lr3l        .set     a4
co1hr3h        .set     a5
co1lr3hH       .set     a0
co1hr3lH       .set     a1
co1lr3hL       .set     a2
co1hr3lL       .set     a3
co1r3H         .set     a5
co1r3Lo        .set     a3
co1r3Le        .set     a2
si1            .set     b8
s3             .set     b1
si1hs3l        .set     b2
si1ls3h        .set     b3
si1ls3l        .set     b4
si1hs3h        .set     b5
si1ls3hH       .set     b0
si1hs3lH       .set     b1
si1ls3hL       .set     b2
si1hs3lL       .set     b3
si1s3H         .set     b5
si1s3Lo        .set     b3
si1s3Le        .set     b2
s3_A           .set     a9
```

Figure 4. C62x C-Callable Assembly Language Functin Source Listing for an Extended Precision Radix-4 FFT (Continued)

```
co1hs3l        .set     a10
co1ls3h        .set     a11
co1ls3l        .set     a12
co1hs3h        .set     a13
co1ls3hH       .set     a8
co1hs3lH       .set     a9
co1ls3hL       .set     a10
co1hs3lL       .set     a11
co1s3H         .set     a13
co1s3Lo        .set     a11
co1s3Le        .set     a10
xi1            .set     a1
xi0            .set     a5
r3_B           .set     b9
si1hr3l        .set     b10
si1lr3h        .set     b11
si1lr3l        .set     b12
si1hr3h        .set     b13
si1lr3hH       .set     b8
si1hr3lH       .set     b9
si1lr3hL       .set     b10
si1hr3lL       .set     b11
si1r3H         .set     b13
si1r3Lo        .set     b11
si1r3Le        .set     b10
yi1            .set     b13
yi0            .set     b5
co2            .set     a0
r2             .set     a1
co2hr2l        .set     a2
co2lr2h        .set     a3
co2lr2l        .set     a4
co2hr2h        .set     a5
co2lr2hH       .set     a0
co2hr2lH       .set     a1
co2lr2hL       .set     a2
co2hr2lL       .set     a3
co2r2H         .set     a5
co2r2Lo        .set     a3
co2r2Le        .set     a2
si2            .set     b0
s2             .set     b1
si2hs2l        .set     b2
si2ls2h        .set     b3
si2ls2l        .set     b4
si2hs2h        .set     b5
si2ls2hH       .set     b0
si2hs2lH       .set     b1
si2ls2hL       .set     b2
si2hs2lL       .set     b3
si2s2H         .set     b5
si2s2Lo        .set     b3
si2s2Le        .set     b2
co2_A          .set     a8
s2_A           .set     a9
```

Figure 4. C62x C-Callable Assembly Language Functin Source Listing for an Extended Precision Radix-4 FFT (Continued)

```
co2hs2l        .set     a10
co2ls2h        .set     a11
co2ls2l        .set     a12
co2hs2h        .set     a13
co2ls2hH       .set     a8
co2hs2lH       .set     a9
co2ls2hL       .set     a10
co2hs2lL       .set     a11
co2s2H         .set     a13
co2s2Lo        .set     a11
co2s2Le        .set     a10
xi2            .set     a1
xi0            .set     a5
si2_B          .set     b8
r2_B           .set     b9
si2hr2l        .set     b10
si2lr2h        .set     b11
si2lr2l        .set     b12
si2hr2h        .set     b13
si2lr2hH       .set     b8
si2hr2lH       .set     b9
si2lr2hL       .set     b10
si2hr2lL       .set     b11
si2r2H         .set     b13
si2r2Lo        .set     b11
si2r2Le        .set     b10
yi2            .set     b13
yi0            .set     b5
co3            .set     a0
r1             .set     a1
co3hr1l        .set     a2
co3lr1h        .set     a3
co3lr1l        .set     a4
co3hr1h        .set     a5
co3lr1hH       .set     a0
co3hr1lH       .set     a1
co3lr1hL       .set     a2
co3hr1lL       .set     a3
co3r1H         .set     a5
co3r1Lo        .set     a3
co3r1Le        .set     a2
si3            .set     b0
s1             .set     b1
si3hs1l        .set     b2
si3ls1h        .set     b3
si3ls1l        .set     b4
si3hs1h        .set     b5
si3ls1hH       .set     b0
si3hs1lH       .set     b1
si3ls1hL       .set     b2
si3hs1lL       .set     b3
si3s1H         .set     b5
si3s1Lo        .set     b3
si3s1Le        .set     b2
co3_A          .set     a8
s1_A           .set     a9
co3hs1l        .set     a10
co3ls1h        .set     a11
```

Figure 4. C62x C-Callable Assembly Language Functin Source Listing for an Extended Precision Radix-4 FFT (Continued)

```
co3ls1l         .set     a12
co3hs1h         .set     a13
co3ls1hH        .set     a8
co3hs1lH        .set     a9
co3ls1hL        .set     a10
co3hs1lL        .set     a11
co3s1H          .set     a13
co3s1Lo         .set     a11
co3s1Le         .set     a10
xi3             .set     a5

si3_B           .set     b8
r1_B            .set     b9
si3hr1l         .set     b10
si3lr1h         .set     b11
si3lr1l         .set     b12
si3hr1h         .set     b13
si3lr1hH        .set     b8
si3hr1lH        .set     b9
si3lr1hL        .set     b10
si3hr1lL        .set     b11
si3r1H          .set     b13
si3r1Lo         .set     b11
si3r1Le         .set     b10
yi3             .set     b13
y               .set     b6
x               .set     b7
i0              .set     b14
i1              .set     b14
i2              .set     b14
i3              .set     b14
wc              .set     a7
ws              .set     a6
a               .set     a14
b               .set     a14
c               .set     a14
bb              .set     a13
cc              .set     a12
xi0t            .set     a0
xi1t            .set     a1
xi2t            .set     a2
xi3t            .set     a3
r1t             .set     a13
r2t             .set     a9
r3t             .set     a10
r4t             .set     a11
yi0t            .set     b0
yi1t            .set     b1
yi2t            .set     b2
yi3t            .set     b3
s1t             .set     b13
s2t             .set     b9
s3t             .set     b10
s4t             .set     b11


                .global  _fft4
```

Figure 4. C62x C-Callable Assembly Language Functin Source Listing for an Extended Precision Radix-4 FFT (Continued)

```
_fft4:

       ; code to preserve the C runtime enviroment

       mvk     .s2      STACKSIZE,stack    ; move stack size into a reg.
       sub     .l2      B15,  stack, B15   ; allocate space on stack
       stw     .d2      B14,  *B15++[1]    ; push B14 onto stack
       stw     .d2      B13,  *B15++[1]    ; push B13 onto stack
       stw     .d2      B12,  *B15++[1]    ; push B12 onto stack
       stw     .d2      B11,  *B15++[1]    ; push B11 onto stack
       stw     .d2      B10,  *B15++[1]    ; push B10 onto stack
       stw     .d2      B3,   *B15++[1]    ; push B3 onto stack
       stw     .d2      A15,  *B15++[1]    ; push A15 onto stack
       stw     .d2      A14,  *B15++[1]    ; push A14 onto stack
||     mv      .l       wcaddr,wc          ; copy argument to register
       stw     .d2      A13,  *B15++[1]    ; push A13 onto stack
||     mv      .l       wsaddr,ws          ; copy argument to register
       stw     .d2      A12,  *B15++[1]    ; push A12 onto stack
||     mv      .l       xaddr,x            ; copy argument to register
       stw     .d2      A11,  *B15++[1]    ; push A11 onto stack
||     mv      .l       yaddr,y            ; copy argument to register
       stw     .d2      A10,  *B15++[1]    ; push A10 onto stack
       mv      .l       B15,A15            ; copy argument to register

       ; begin FFT processing


       ;n2 = n;
       ;e = 1;
       ;

       stw     .d2      npoints,*+B15[N2]
||     mvk     .s2      1,    e
       stw     .d2      e,    *+B15[E]
||     stw     .d       npoints,*+A15[N]



       ;for(k=0; k<m; k++)
       ;{
       ;       n1 = n2;
       ;       n2 = n2 >> 2;
       ;       a = 0;

       mv      .l2      mstages, KCNT
       stw     .d       KCNT,*+B15[K]

KLOOP:
       ldw     .d2      *+B15[N2],   n1            ; n1 = n2

       zero    .l       a                          ; a = 0
||     zero    .l       j                          ; j = 0

       stw     .d       a, *+A15[A]                ; store a on stack
||     stw     .d       j,*+B15[J]

       stw     .d       j,*+B15[I0]                ; store i0 on stack
||     stw     .d       a, *+A15[B]                ; store b on stack
||     mv      .l       j,i0

       ldw     .d       *+x[i0], xi0t              ; xi0 = x[i0]
```

Figure 4. C62x C-Callable Assembly Language Functin Source Listing for an Extended Precision Radix-4 FFT (Continued)

```
      stw     .d      n1,*+B15[N1]
||    shr     .s2     n1,   2, n2            ; n2 = n2 >> 2;
      stw     .d2     n2,   *+B15[N2]        ; store n2 on stack
||    mv      .l2     n2,   JCNT             ; Initialize JLOOP counter

      stw     .d      a, *+A15[C]            ; store c on stack
||    ldw     .d      *+y[i0], yi0t          ; yi0 = y[i0]
||    add     .l      i0,   n2, i1           ; i1 = i0 + n2

JLOOP:

      ldw     .d      *+x[i1], xi1t       ; xi1 = x[i1]
||    stw     .d      i1,*+A15[I1]        ; store i1 on stack

      ldw     .d      *+y[i1], yi1t       ; yi1 = y[i1]
||    add     .l      i1,   n2, i2        ; i2 = i1 + n2
||    ldw     .d      *+A15[A],a

ILOOP:

      ldw     .d      *+x[i2], xi2t       ; xi2 = x[i2]
||    stw     .d      i2,*+A15[I2]        ; store i2 on stack

      ldw     .d      *+y[i2], yi2t       ; yi2 = y[i2]
||    add     .l      i2,   n2, i3        ; i3 = i2 + n2

      ldw     .d      *+x[i3], xi3t       ; xi3 = x[i3]
||    stw     .d      i3,*+A15[I3]        ; store i3 on stack

      ldw     .d      *+y[i3], yi3t       ; yi3 = y[i3]

      ldw     .d      *+B15[I0], i0       ; store i0 on stack
||    ldw     .d      *+wc[a],co1

      add     .l      xi0t, xi2t,  r1t    ; r1 = x[i0] + x[i2]
||    sub     .s      xi0t, xi2t,  r3t    ; r3 = x[i0] – x[i2]
||    ldw     .d      *+ws[a],si1

      add     .l      yi0t, yi2t,  s1t    ; s1 = y[i0] + y[i2]
||    sub     .s      yi0t, yi2t,  s3t    ; s3 = y[i0] – y[i2]

      add     .l      xi1t, xi3t,  r2t    ; r2 = x[i1] + x[i3]
||    sub     .s      xi1t, xi3t,  r4t    ; r4 = x[i1] – x[i3]


      add     .l      yi1t, yi3t,  s2t    ; s2 = y[i1] + y[i3]
||    sub     .s      yi1t, yi3t,  s4t    ; s4 = y[i1] – y[i3]
||    add     .l      r1t,  r2t,   xi0t   ; xi0 = r1 + r2


      sub     .l      s3t,  r4t,   s3     ; s3 = s3 – r4
||    add     .l      r3t,  s4t,   r3     ; r3 = r3 + s4
||    stw     .d      xi0t,    *+x[i0]    ; x[i0] = r1 + r2



      mpyhslu .m1     r3,co1,co1lr3h      ; co1l u*s r3h
||    mpyhslu .m2     s3,si1,si1ls3h      ; si1l u*s s3h
||    ldw     .d      *+A15[B],b
||    add     .l      s1t,  s2t,   yi0t   ; yi0 = s1 + s2
||    sub     .l      r1t,  r2t,   r2t    ; r2 = r1 – r2
```

Figure 4. C62x C-Callable Assembly Language Functin Source Listing for an Extended Precision Radix-4 FFT (Continued)

```
      mpyhslu .m1      co1,r3,co1hr3l        ; co1h s*u r3l
||    mpyhslu .m2      si1,s3,si1hs3l        ; si1h s*u s3l
||    sub     .l       s1t,  s2t,   s2t      ; s2 = s1 – s2
||    stw     .d       yi0t,    *+y[i0]      ; y[i0] = s1 + s2
||    stw     .d       r2t,*+A15[R2]
||    sub     .l       r3t,  s4t,   r1t      ; r1 = r3 – s4

      mpyu    .m1      co1,r3,co1lr3l        ; co1l u*u r3l
||    mpyu    .m2      si1,s3,si1ls3l        ; si1l u*u s3l
||    stw     .d       s2t,*+B15[S2]
||    add     .l       s3t,  r4t,   s1t      ; s1 = s3 + r4
||    stw     .d       r1t,*+A15[R1]


      mpyh    .m1      r3,co1,co1hr3h        ; co1h s*s r3h
||    mpyh    .m2      s3,si1,si1hs3h        ; si1h s*s s3h
||    mv      .l       s3,s3_A
||    mv      .l       r3,r3_B
||    stw     .d       s1t,*+B15[S1]


      shr     .s1      co1lr3h,16,co1lr3hH   ; co1lr3hH = co1lr3h >>s 16
||    shr     .s2      si1ls3h,16,si1ls3hH   ; si1ls3hH = si1ls3h >>s 16
||    mpyhslu .m1      s3_A,co1,co1ls3h      ; co1l u*s s3h
||    mpyhslu .m2      r3_B,si1,si1lr3h      ; si1l u*s r3h


      shr     .s1      co1hr3l,16,co1hr3lH   ; co1hr3lH = co1hr3l >>s 16
||    shr     .s2      si1hs3l,16,si1hs3lH   ; si1hs3lH = si1hs3l >>s 16
||    mpyhslu .m1      co1,s3_A,co1hs3l      ; co1h s*u s3l
||    mpyhslu .m2      si1,r3_B,si1hr3l      ; si1h s*u r3l
||    ldw     .d       *+ws[b],si2
||    ldw     .d       *+B15[R2],r2

      shl     .s1      co1lr3h,16,co1lr3hL   ; co1lr3hL = co1lr3h << 16
||    shl     .s2      si1ls3h,16,si1ls3hL   ; si1ls3hL = si1ls3h << 16
||    mpyu    .m1      co1,s3_A,co1ls3l      ; co1l u*u s3l
||    mpyu    .m2      si1,r3_B,si1lr3l      ; si1l u*u r3l
||    ldw     .d       *+wc[b],co2
||    ldw     .d       *+B15[S2],s2

      shl     .s1      co1hr3l,16,co1hr3lL   ; co1hr3lL = co1hr3l << 16
||    shl     .s2      si1hs3l,16,si1hs3lL   ; si1hs3lL = si1hs3l << 16
||    mpyh    .m1      s3_A,co1,co1hs3h      ; co1h s*s s3h
||    mpyh    .m2      r3_B,si1,si1hr3h      ; si1h s*s r3h

      add     .l1      co1hr3h,co1hr3lH,co1r3H   ; co1r3H = co1hr3l + co1hr3lH
||    add     .l2      si1hs3h,si1hs3lH,si1s3H   ; si1s3H = si1hs3l + si1hs3lH
||    shr     .s1      co1ls3h,16,co1ls3hH       ; co1ls3hH = co1ls3h >>s 16
||    shr     .s2      si1lr3h,16,si1lr3hH       ; si1lr3hH = si1lr3h >>s 16


      add     .l1      co1r3H,co1lr3hH,co1r3H    ; co1r3H = co1r3H + co1lr3hH
||    add     .l2      si1s3H,si1ls3hH,si1s3H    ; si1s3H = si1s3H + si1ls3hH
||    shr     .s1      co1hs3l,16,co1hs3lH       ; co1hs3lH = co1hs3l >>s 16
||    shr     .s2      si1hr3l,16,si1hr3lH       ; si1hr3lH = si1hr3l >>s 16
```

Figure 4. C62x C-Callable Assembly Language Functin Source Listing for an Extended Precision Radix-4 FFT (Continued)

```
       addu    .l1      co1lr3hL,co1hr3lL,co1r3Lo:co1r3Le      ; (long)co1r3L =
                         co1lr3l + co1hr3lL
||     addu    .l2      si1ls3hL,si1hs3lL,si1s3Lo:si1s3Le      ; (long)si1s3L =
                         si1ls3l + si1hs3lL
||     shl     .s1      co1ls3h,16,co1ls3hL           ; co1ls3hL = co1ls3h << 16
||     shl     .s2      si1lr3h,16,si1lr3hL           ; si1lr3hL = si1lr3h << 16
||     ldw     .d       *+B15[C],c


       addu    .l1      co1lr3l,co1r3Lo:co1r3Le,co1r3Lo:co1r3Le ; (long)co1r3L =
                         co1lr3hL + (long)co1r3L
||     addu    .l2      si1ls3l,si1s3Lo:si1s3Le,si1s3Lo:si1s3Le ;(long)si1s3L =
                         si1ls3hL + (long)si1s3L
||     shl     .s1      co1hs3l,16,co1hs3lL        ; co1hs3lL = co1hs3l << 16
||     shl     .s2      si1hr3l,16,si1hr3lL        ; si1hr3lL = si1hr3l << 16
||     ldw     .d2      *+B15[I1],i1               ; load i1 from stack
||     mpyhslu .m1      r2,co2,co2lr2h             ; co2l u*s r2h
||     mpyhslu .m2      s2,si2,si2ls2h             ; si2l u*s s2h




       add     .l1      co1r3Lo,co1r3H,co1r3H      ; co1r3H = co1r3Lhigh + co1r3H
||     add     .l2      si1s3Lo,si1s3H,si1s3H      ; si1s3H = si1s3Lhigh + si1s3H
||     add     .d1      co1hs3h,co1hs3lH,co1s3H    ; co1s3H = co1hs3l + co1hs3lH
||     add     .d2      si1hr3h,si1hr3lH,si1r3H    ; si1r3H = si1hr3l + si1hr3lH
||     mv      .s       s2,s2_A
||     mv      .s       r2,r2_B
||     mpyhslu .m1      co2,r2,co2hr2l             ; co2h s*u r2l
||     mpyhslu .m2      si2,s2,si2hs2l             ; si2h s*u s2l


       shl     .s1      co1r3H,1,co1r3H
||     shl     .s2      si1s3H,1,si1s3H
||     add     .l1      co1s3H,co1ls3hH,co1s3H     ; co1s3H = co1s3H + co1ls3hH
||     add     .l2      si1r3H,si1lr3hH,si1r3H     ; si1r3H = si1r3H + si1lr3hH
||     mv      .d       co2,co2_A
||     mv      .d       si2,si2_B
||     mpyu    .m1      co2,r2,co2lr2l             ; co2l u*u r2l
||     mpyu    .m2      si2,s2,si2ls2l             ; si2l u*u s2l


       add     .s1x     co1r3H,si1s3H,xi1          ; xi1 = co1*r3 + si1*s3
||     addu    .l1      co1ls3hL,co1hs3lL,co1s3Lo:co1s3Le   ; (long)co1s3L =
                         co1ls3l + co1hs3lL
||     addu    .l2      si1lr3hL,si1hr3lL,si1r3Lo:si1r3Le   ; (long)si1r3L =
                         si1lr3l + si1hr3lL
||     mpyh    .m1      r2,co2,co2hr2h             ; co2h s*s r2h
||     mpyh    .m2      s2,si2,si2hs2h             ; si2h s*s s2h


       addu    .l1      co1ls3l,co1s3Lo:co1s3Le,co1s3Lo:co1s3Le ; (long)co1s3L =
                         co1ls3hL + (long)co1s3L
||     addu    .l2      si1lr3l,si1r3Lo:si1r3Le,si1r3Lo:si1r3Le ; (long)si1r3L =
                         si1lr3hL + (long)si1r3L
||     shr     .s1      co2lr2h,16,co2lr2hH        ; co2lr2hH = co2lr2h >>s 16
||     shr     .s2      si2ls2h,16,si2ls2hH        ; si2ls2hH = si2ls2h >>s 16
||     mpyhslu .m1      s2_A,co2_A,co2ls2h         ; co2l u*s s2h
||     mpyhslu .m2      r2_B,si2_B,si2lr2h         ; si2l u*s r2h
```

Figure 4. C62x C-Callable Assembly Language Functin Source Listing for an Extended Precision Radix-4 FFT (Continued)

```
       stw     .d2      xi1,*+x[i1]               ; x[i1] = co1*r3 + si1*s3
||     add     .l1      co1s3Lo,co1s3H,co1s3H     ; co1s3H = co1s3Lhigh + co1s3H
||     add     .l2      si1r3Lo,si1r3H,si1r3H     ; si1r3H = si1r3Lhigh + si1r3H
||     shr     .s1      co2hr2l,16,co2hr2lH; co2hr2lH = co2hr2l >>s 16
||     shr     .s2      si2hs2l,16,si2hs2lH; si2hs2lH = si2hs2l >>s 16
||     mpyhslu .m1      co2_A,s2_A,co2hs2l ; co2h s*u s2l
||     mpyhslu .m2      si2_B,r2_B,si2hr2l ; si2h s*u r2l


       shl     .s1      co1s3H,1,co1s3H
||     shl     .s2      si1r3H,1,si1r3H
||     ldw     .d       *+ws[c],si3
||     ldw     .d       *+B15[R1],r1


       sub     .l2x     co1s3H,si1r3H,yi1  ; yi1 = co1*s3 – si1*r3
||     shl     .s1      co2lr2h,16,co2lr2hL; co2lr2hL = co2lr2h << 16
||     shl     .s2      si2ls2h,16,si2ls2hL; si2ls2hL = si2ls2h << 16
||     mpyu    .m1      co2_A,s2_A,co2ls2l ; co2l u*u s2l
||     mpyu    .m2      si2_B,r2_B,si2lr2l ; si2l u*u r2l
||     ldw     .d       *+wc[c],co3
||     ldw     .d       *+B15[S1],s1

       stw     .d2      yi1,*+y[i1]     ; y[i1] = co1*s3 – si1*r3
||     shl     .s1      co2hr2l,16,co2hr2lL; co2hr2lL = co2hr2l << 16
||     shl     .s2      si2hs2l,16,si2hs2lL; si2hs2lL = si2hs2l << 16
||     mpyh    .m1      s2_A,co2_A,co2hs2h ; co2h s*s s2h
||     mpyh    .m2      r2_B,si2_B,si2hr2h ; si2h s*s r2h

       add     .l1      co2hr2h,co2hr2lH,co2r2H   ; co2r2H = co2hr2l + co2hr2lH
||     add     .l2      si2hs2h,si2hs2lH,si2s2H   ; si2s2H = si2hs2l + si2hs2lH
||     shr     .s1      co2ls2h,16,co2ls2hH; co2ls2hH = co2ls2h >>s 16
||     shr     .s2      si2lr2h,16,si2lr2hH; si2lr2hH = si2lr2h >>s 16


       add     .l1      co2r2H,co2lr2hH,co2r2H ; co2r2H = co2r2H + co2lr2hH
||     add     .l2      si2s2H,si2ls2hH,si2s2H ; si2s2H = si2s2H + si2ls2hH
||     shr     .s1      co2hs2l,16,co2hs2lH; co2hs2lH = co2hs2l >>s 16
||     shr     .s2      si2hr2l,16,si2hr2lH; si2hr2lH = si2hr2l >>s 16


       addu    .l1      co2lr2hL,co2hr2lL,co2r2Lo:co2r2Le   ; (long)co2r2L =
                         co2lr2l + co2hr2lL
||     addu    .l2      si2ls2hL,si2hs2lL,si2s2Lo:si2s2Le   ; (long)si2s2L =
                         si2ls2l + si2hs2lL
||     shl     .s1      co2ls2h,16,co2ls2hL; co2ls2hL = co2ls2h << 16
||     shl     .s2      si2lr2h,16,si2lr2hL; si2lr2hL = si2lr2h << 16


       addu    .l1      co2lr2l,co2r2Lo:co2r2Le,co2r2Lo:co2r2Le ;(long)co2r2L =
                         co2lr2hL + (long)co2r2L
||     addu    .l2      si2ls2l,si2s2Lo:si2s2Le,si2s2Lo:si2s2Le ;(long)si2s2L =
                         si2ls2hL + (long)si2s2L
```

Figure 4. C62x C-Callable Assembly Language Functin Source Listing for an Extended Precision Radix-4 FFT (Continued)

```
||     shl     .s1      co2hs2l,16,co2hs2lL; co2hs2lL = co2hs2l << 16
||     shl     .s2      si2hr2l,16,si2hr2lL; si2hr2lL = si2hr2l << 16
||     ldw     .d2      *+B15[I2],i2    ; load i2 from stack
||     mpyhslu .m1      r1,co3,co3lr1h     ; co3l u*s r1h
||     mpyhslu .m2      s1,si3,si3ls1h     ; si3l u*s s1h


       add     .l1      co2r2Lo,co2r2H,co2r2H  ; co2r2H = co2r2Lhigh + co2r2H
||     add     .l2      si2s2Lo,si2s2H,si2s2H  ; si2s2H = si2s2Lhigh + si2s2H
||     add     .d1      co2hs2h,co2hs2lH,co2s2H   ; co2s2H = co2hs2l + co2hs2lH
||     add     .d2      si2hr2h,si2hr2lH,si2r2H   ; si2r2H = si2hr2l + si2hr2lH
||     mpyhslu .m1      co3,r1,co3hr1l     ; co3h s*u r1l
||     mpyhslu .m2      si3,s1,si3hs1l     ; si3h s*u s1l
||     mv      .s       s1,s1_A
||     mv      .s       r1,r1_B


       shl     .s1      co2r2H,1,co2r2H
||     shl     .s2      si2s2H,1,si2s2H
||     add     .l1      co2s2H,co2ls2hH,co2s2H  ; co2s2H = co2s2H + co2ls2hH
||     add     .l2      si2r2H,si2lr2hH,si2r2H   ; si2r2H = si2r2H + si2lr2hH
||     mpyu    .m1      co3,r1,co3lr1l     ; co3l u*u r1l
||     mpyu    .m2      si3,s1,si3ls1l     ; si3l u*u s1l
||     mv      .d       co3,co3_A
||     mv      .d       si3,si3_B

       add     .s1x     co2r2H,si2s2H,xi2  ; xi2 = co2*r2 + si2*s2
||     addu    .l1      co2ls2hL,co2hs2lL,co2s2Lo:co2s2Le   ; (long)co2s2L =
               co2ls2l + co2hs2lL
||     addu    .l2      si2lr2hL,si2hr2lL,si2r2Lo:si2r2Le   ; (long)si2r2L =
                         si2lr2l + si2hr2lL
||     mpyh    .m1      r1,co3,co3hr1h     ; co3h s*s r1h
||     mpyh    .m2      s1,si3,si3hs1h     ; si3h s*s s1h


       addu    .l1      co2ls2l,co2s2Lo:co2s2Le,co2s2Lo:co2s2Le ; (long)co2s2L =
                         co2ls2hL + (long)co2s2L
||     addu    .l2      si2lr2l,si2r2Lo:si2r2Le,si2r2Lo:si2r2Le ; (long)si2r2L =
                         si2lr2hL + (long)si2r2L
||     shr     .s1      co3lr1h,16,co3lr1hH; co3lr1hH = co3lr1h >>s 16
||     shr     .s2      si3ls1h,16,si3ls1hH; si3ls1hH = si3ls1h >>s 16
||     mpyhslu .m1      s1_A,co3_A,co3ls1h ; co3l u*s s1h
||     mpyhslu .m2      r1_B,si3_B,si3lr1h ; si3l u*s r1h


       stw     .d2      xi2,*+x[i2]     ; x[i2] = co2*r2 + si2*s2
||     add     .l1      co2s2Lo,co2s2H,co2s2H  ; co2s2H = co2s2Lhigh + co2s2H
||     add     .l2      si2r2Lo,si2r2H,si2r2H  ; si2r2H = si2r2Lhigh + si2r2H
||     shr     .s1      co3hr1l,16,co3hr1lH; co3hr1lH = co3hr1l >>s 16
||     shr     .s2      si3hs1l,16,si3hs1lH; si3hs1lH = si3hs1l >>s 16
||     mpyhslu .m1      co3_A,s1_A,co3hs1l ; co3h s*u s1l
||     mpyhslu .m2      si3_B,r1_B,si3hr1l ; si3h s*u r1l


       shl     .s1      co2s2H,1,co2s2H
||     shl     .s2      si2r2H,1,si2r2H
||     ldw     .d       *+B15[N1],n1t
```

Figure 4. C62x C-Callable Assembly Language Functin Source Listing for an Extended Precision Radix-4 FFT (Continued)

```
       sub     .l2x     co2s2H,si2r2H,yi2  ; yi2 = co2*s2 – si2*r2
||     shl     .s1      co3lr1h,16,co3lr1hL; co3lr1hL = co3lr1h << 16
||     shl     .s2      si3ls1h,16,si3ls1hL; si3ls1hL = si3ls1h << 16
||     mpyu    .m1      co3_A,s1_A,co3ls1l ; co3l u*u s1l
||     mpyu    .m2      si3_B,r1_B,si3lr1l ; si3l u*u r1l
||     ldw     .d       *+B15[I0],i0t

       shl     .s1      co3hr1l,16,co3hr1lL; co3hr1lL = co3hr1l << 16
||     shl     .s2      si3hs1l,16,si3hs1lL; si3hs1lL = si3hs1l << 16
||     mpyh    .m1      s1_A,co3_A,co3hs1h ; co3h s*s s1h
||     mpyh    .m2      r1_B,si3_B,si3hr1h ; si3h s*s r1h
||     stw     .d2      yi2,*+y[i2]     ; y[i2] = co2*s2 – si2*r2

       add     .l1      co3hr1h,co3hr1lH,co3r1H   ; co3r1H = co3hr1l + co3hr1lH
||     add     .l2      si3hs1h,si3hs1lH,si3s1H   ; si3s1H = si3hs1l + si3hs1lH
||     shr     .s1      co3ls1h,16,co3ls1hH; co3ls1hH = co3ls1h >>s 16
||     shr     .s2      si3lr1h,16,si3lr1hH; si3lr1hH = si3lr1h >>s 16
||     ldw     .d       *+B15[N],nt

       add     .l1      co3r1H,co3lr1hH,co3r1H ; co3r1H = co3r1H + co3lr1hH
||     add     .l2      si3s1H,si3ls1hH,si3s1H ; si3s1H = si3s1H + si3ls1hH
||     shr     .s1      co3hs1l,16,co3hs1lH; co3hs1lH = co3hs1l >>s 16
||     shr     .s2      si3hr1l,16,si3hr1lH; si3hr1lH = si3hr1l >>s 16
||     ldw     .d2      *+B15[N2],   n2


       addu    .l1      co3lr1hL,co3hr1lL,co3r1Lo:co3r1Le   ; (long)co3r1L =
                         co3lr1l + co3hr1lL
||     addu    .l2      si3ls1hL,si3hs1lL,si3s1Lo:si3s1Le   ; (long)si3s1L =
                         si3ls1l + si3hs1lL
||     shl     .s1      co3ls1h,16,co3ls1hL; co3ls1hL = co3ls1h << 16
||     shl     .s2      si3lr1h,16,si3lr1hL; si3lr1hL = si3lr1h << 16


       addu    .l1      co3lr1l,co3r1Lo:co3r1Le,co3r1Lo:co3r1Le ; (long)co3r1L =
                         co3lr1hL + (long)co3r1L
||     addu    .l2      si3ls1l,si3s1Lo:si3s1Le,si3s1Lo:si3s1Le ; (long)si3s1L =
                         si3ls1hL + (long)si3s1L
||     shl     .s1      co3hs1l,16,co3hs1lL; co3hs1lL = co3hs1l << 16
||     shl     .s2      si3hr1l,16,si3hr1lL; si3hr1lL = si3hr1l << 16
||     ldw     .d2      *+B15[I3],i3    ; load i3 from stack


       add     .l1      co3r1Lo,co3r1H,co3r1H  ; co3r1H = co3r1Lhigh + co3r1H
||     add     .l2      si3s1Lo,si3s1H,si3s1H  ; si3s1H = si3s1Lhigh + si3s1H
||     add     .s       co3hs1h,co3hs1lH,co3s1H   ; co3s1H = co3hs1l + co3hs1lH
||     add     .d2      si3hr1h,si3hr1lH,si3r1H   ; si3r1H = si3hr1l + si3hr1lH
||     add     .s       i0t,  n1t,   i0t
||     ldw     .d       *+A15[J],ja


       shl     .s1      co3r1H,1,co3r1H
||     shl     .s2      si3s1H,1,si3s1H
||     add     .l1      co3s1H,co3ls1hH,co3s1H  ; co3s1H = co3s1H + co3ls1hH
||     add     .d2      si3r1H,si3lr1hH,si3r1H   ; si3r1H = si3r1H + si3lr1hH
```

Figure 4. C62x C-Callable Assembly Language Functin Source Listing for an Extended Precision Radix-4 FFT (Continued)

```
||     cmplt   .l       i0t,  nt, ICNT
||     stw     .d       i0t,*+A15[I0]         ; store i0 on stack

       add     .s1x     co3r1H,si3s1H,xi3  ; xi3 = co3*r1 + si3*s1
||     addu    .l1      co3ls1hL,co3hs1lL,co3s1Lo:co3s1Le
                              ; (long)co3s1L = co3ls1l + co3hs1lL
||     addu    .l2      si3lr1hL,si3hr1lL,si3r1Lo:si3r1Le
                              ; (long)si3r1L = si3lr1l + si3hr1lL
||[ICNT]       b        .s    ILOOP
||     ldw     .d       *+y[i0t], yi0t     ; yi0 = y[i0]
||     ldw     .d       *+A15[E],ea

       addu    .l1      co3ls1l,co3s1Lo:co3s1Le,co3s1Lo:co3s1Le ; (long)co3s1L =
                        co3ls1hL + (long)co3s1L
||     addu    .l2      si3lr1l,si3r1Lo:si3r1Le,si3r1Lo:si3r1Le ; (long)si3r1L =
                        si3lr1hL + (long)si3r1L
||     ldw     .d       *+x[i0t],    xi0t   ; xi0 = x[i0]
||     add     .s       i0t,  n2, i1t ; i1 = i0 + n2


       stw     .d2      xi3,*+x[i3]        ; x[i3] = co3*r1 + si3*s1
||     add     .l1      co3s1Lo,co3s1H,co3s1H  ; co3s1H = co3s1Lhigh + co3s1H
||     add     .l2      si3r1Lo,si3r1H,si3r1H  ; si3r1H = si3r1Lhigh + si3r1H
||     stw     .d       i1t,*+A15[I1]         ; store i1 on stack



       shl     .s1      co3s1H,1,co3s1H
||     shl     .s2      si3r1H,1,si3r1H
||     ldw     .d       *+x[i1t],    xi1t   ; xi1 = x[i1]
||     add     .l       ja,   1,  ja          ; j=j+1

       sub     .l2x     co3s1H,si3r1H,yi3  ; yi3 = co3*s1 – si3*r1
||[ICNT]       ldw      .d    *+y[i1t], yi1t   ; yi1 = y[i1]
||[!ICNT]      stw      .d    ja,*+A15[J]

       stw     .d2      yi3,*+y[i3]     ; y[i3] = co3*s1 – si3*r1
||     add     .s       i1t,  n2, i2    ; i2 = i1 + n2
||     cmplt   .l       ja,   n2, JCNT
||     ldw     .d       *+A15[A],a

ILOOPEND:

[JCNT] b       .s2      JLOOP
||     mpy     .m       ja,   ea, a        ; a = (j + 1)*e

       mv      .l       ja,i0

       stw     .d       a, *+A15[A]     ; store a on stack
||     add     .l       a,    a,  bb       ; b = a + a
||     stw     .d       i0,*+B15[I0]    ; store i0 on stack

       stw     .d       bb, *+A15[B]    ; store b on stack
||     add     .l       bb,   a,  cc       ; c = b + a

       ldw     .d       *+x[i0], xi0t   ; xi0 = x[i0]
```

Figure 4. C62x C-Callable Assembly Language Functin Source Listing for an Extended Precision Radix-4 FFT (Continued)

```
      stw     .d       cc, *+A15[C]    ; store c on stack
||    ldw     .d       *+y[i0], yi0t   ; yi0 = y[i0]
||    add     .l       i0,   n2, i1       ; i1 = i0 + n2



JLOOPEND:
      ldw     .d       *+B15[E],e
      ldw     .d       *+B15[K],KCNT

      nop              3
      shl     .s       e,    2, e         ; e = e << 2;
      stw     .d       e,*+B15[E]
||[KCNT]      sub      .l    KCNT, 1, KCNT
      stw     .d       KCNT,*+B15[K]
||[KCNT]      b        .s    KLOOP
      nop              5
KLOOPEND:



      ; code to restore the C runtime enviroment, and return to calling
      function

      ldw     .d2      *––B15[1],   A10 ; pop A10 from the stack
      ldw     .d2      *––B15[1],   A11 ; pop A11 from the stack
      ldw     .d2      *––B15[1],   A12 ; pop A12 from the stack
      ldw     .d2      *––B15[1],   A13 ; pop A13 from the stack
      ldw     .d2      *––B15[1],   A14 ; pop A14 from the stack
      ldw     .d2      *––B15[1],   A15 ; pop A15 from the stack
      ldw     .d2      *––B15[1],   B3  ; pop B3 from the stack
      ldw     .d2      *––B15[1],   B10 ; pop B10 from the stack
      ldw     .d2      *––B15[1],   B11 ; pop B11 from the stack
      ldw     .d2      *––B15[1],   B12 ; pop B12 from the stack
      ldw     .d2      *––B15[1],   B13 ; pop B13 from the stack
      b       .s2      B3
||    ldw     .d2      *––B15[1],   B14 ; pop B14 from the stack
      mvk     .s2      STACKSIZE,stack    ; move stack size into a reg.
      add     .l2      B15,  stack, B15 ; de-allocate space on stack
      nop              3
```

Figure 4 C62x C-Callable Assembly Language Functin Source Listing for an Extended Precision Radix-4 FFT (Continued)

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265