C Implementation of the TMS320C62xx Intrinsic Operators

Eric Biscondi *Digital Signal Processing Solutions*

ABSTRACT

The first optimization step that you can perform on C source code for the TMS320C62xx is to use intrinsic operators. Intrinsics are used like functions and produce assembly language statements that would otherwise be inexpressible in C. The problem is that once you have performed the first optimization step, your C source code is no longer ANSI C compatible. The code proposed within this application report, allows you to write C code using intrinsic operators keeping the possibility to validate the code on a workstation (either SUN or PC). The sample code described in this application report can be downloaded from http://www.ti.com/lit/zip/SPRA616.

Contents

List of Figures

1 Design Problem

The first optimization step that you can perform on C source code for the TMS320C62xx is to use intrinsic operators. Intrinsics are used like functions and produce assembly language statements that would otherwise be inexpressible in C. The problem is that once you have performed this first optimization step, your C source code is no more ANSI C compatible.

How can I simulate C code written using intrinsics on a workstation / PC to validate my code?

2 Solution

The first optimization step that you can perform on C source code for the TMS320C62xx is to use intrinsic operators. Intrinsics are used like functions and produce assembly language statements that would otherwise be inexpressible in C. The problem is that once you have performed this first optimization step, your C source code is no more ANSI C compatible.

The code proposed within this note, allows the user to write C code using intrinsic operators keeping the possibility to validate his code on a workstation (either SUN or PC).

There are two different files, *cintrins.c* and *cintrins.h* listed below in the “corresponding code” section. *cintrins.c* includes a C implementation of all the intrinsic operators. *cintrins.h* includes all the declarations.

The first step when compiling code for the ’C62xx is to make sure the different type are correctly defined. In Figure 1, there is an example of file that could be used to define the different types of data depending on the machine used.

```
#if defined(_unix_)
#include ”Long40.h”
typedef Long40              Word40;
typedef int                 Word32;
typedef unsigned int        UWord32;
typedef short               Word16;
typedef unsigned short      UWord16;
#include <assert.h>
#elif defined(_TMS320C6200)
#define assert(a)
typedef long                Word40;
typedef unsigned long       UWord40;
typedef int                 Word32;
typedef unsigned int        UWord32;
typedef short               Word16;
typedef unsigned short      UWord16;
#else
#include ”Long40.h”
typedef Long40              Word40;
typedef long int            Word32;
typedef unsigned long int   UWord32;
typedef short int           Word16;
typedef unsigned short int  UWord16;
#include <assert.h>
#endif
```

Figure 1. Example of type.h file: Type Definition

You must include the header file *cintrinsic.h* in C files which contain intrinsics and compile and link with *cintrinsic.c.*

The ’C6x supports a new type not commonly found on other microprocessors, a 40 bit long. To write validation code for a PC or SUN, a new data type must be defined. This data type is declared with C++ in the file *long40.h*.

The code (*cintrins.c, cintrins.h, Long40.c, Long40.h*) can be accessed from the same location where you download this application report.

All the files have been compiled and tested using the GNU C++ compiler (g++ for SUN and DJGPP for PC).

3 Corresponding Code

```
/*************************************************************************/
/* Texas Instruments France                                              */
/*                                                                       */
/* NAME                Cintrins.c                                        */
/*                                                                       */
/* REVISION            16 Oct.97                                         */
/                                                                        */
/* DESCRIPTION C implementation of the 'C62xx intrinsics                 */
/*                                                                       */
/*************************************************************************/

#if!defined(_TMS320C6200)
#include "cintrins.h"

Word32 SAT_Bit;

/*************************************************************************/
/* _abs C function                                                       */
/*************************************************************************/
Word32 abs_c(Word32 src1)
{
        if(src1 == MIN_32)
                    return((Word40)MAX_32);
        else
                    return((src1<0) ? -src1 : src1);
}

/*************************************************************************/
/* _add2 C function                                                      */
/*************************************************************************/
Word32 add2_c(Word32 src1, Word32 src2)
{
 return(( ((src1&0x0ffff)+(src2&0x0ffff)) &0x0ffff) | ( ( ((src1&0x0ffff0000) >> 16)+
                      ((src2&0x0ffff0000) >> 16) ) << 16 )    );
}

/*************************************************************************/
/* _clr  C function                                                      */
/*************************************************************************/
UWord32 clr_c(UWord32 src1, UWord32 csta, UWord32 cstb)
{
Word32 i;
Word32 maska=0x0;
Word32 maskb=0x0ffffffff;
         if(csta<=cstb)       {
                    for(i=0;i<csta;i++)
                                maska = (maska<<1)+1;
                    for(i=0;i<=cstb;i++)
                                maskb = maskb << 1;
                    maska = maska | maskb;
                    return(maska&src1);
        }
        else{
```

```
                        return(src1);
                }
        }
```

```
/*************************************************************************/
/* _ext  C function                                                      */
/*************************************************************************/
Word32 ext_c(Word32 src1, UWord32 csta, UWord32 cstb)
{
        return( (src1<<csta) >> cstb);
}

/*************************************************************************/
/* _extu C function                                                      */
/*************************************************************************/
UWord32 extu_c(UWord32 src1, UWord32 csta, UWord32 cstb)
{
        return( (src1<<csta) >> cstb);
}

/*************************************************************************/
/* _lmbd C function                                                      */
/*************************************************************************/
UWord32 lmbd_c(UWord32 src1,UWord32 src2)
{
Word32 i=0;
UWord32 mask = 0x80000000;
        if(src1&0x01) {              /* Search for a 1 */
                  while((src2&mask) == 0) {
                            mask = mask >> 1;
                            i++;
                            if(i==32) return(32);
                  }
        }
        else {                                    /* Search for a 0 */
                  while((src2&mask) != 0) {
                            mask = mask >> 1;
                            i++;
                            if(i==32) return(32);
                  }
        }
        return(i);
}

/*************************************************************************/
/* _lnorm C function                                                     */
/*************************************************************************/
#define MASK39 (Word40)0x8000000000
UWord32 lnorm_c(Word40 src2)
{
Word32 i=0;
        if((src2 >> 39) == 0){                    /* Positive Number */
                  while((src2 >> 39) != 1){
                            src2 = src2 << 1;
                            i++;
                            if(i==40) return(39);
                  }
```

```
        }
        else {                                          /* Negative Number */
                while((src2 >> 39) != 0) {
                        src2 = src2 << 1;
                        i++;
                        if(i==40) return(39);
                }
        }
        return(i-1);
}


/*************************************************************************/
/* _lsadd C function                                                     */
/*************************************************************************/
Word40 lsadd_c(int src1, Word40 src2)
{
Word40 result;
        result = src1 + src2;
        if( ( (src1>> 31) ^ ( src2 >> 39)) == 0) {
                if(((result^src2) >> 39) != 0) {
                        if(src2<0)
                                result = ((Word40)MIN_32 << 8);
                        else
                                result = (((Word40)MAX_32 << 8)) + 0xff;
                        SAT_Bit = 1;
                }
        }
        return(result);
}

/*************************************************************************/
/* _lssub C function                                                     */
/*************************************************************************/
Word40 lssub_c(Word16 src1, Word40 src2)
{
Word40 result;
        result = (Word40)src1 - src2;
        if( ( ((Word40)src1>> 31) ^ ( src2 >> 39)) != 0) {
                if( ( (result>>39) ^ ((Word40)src1 >> 31)) != 0) {
                        if(src2<0)
                                result = ((Word40)MIN_32 << 8);
                        else
                                result = (((Word40)MAX_32 << 8)) + 0xff;
                }
        }
        return((Word40)result);
}
```

```
/*************************************************************************/
/* _mpy C function                                                       */
/*************************************************************************/
Word32 mpy_c(Word32 src1,Word32 src2)
{
        return((Word16)src1 * (Word16)src2);
}

/*************************************************************************/
/* _mpysu C function                                                     */
/*************************************************************************/
Word32 mpysu_c(Word32 src1, UWord32 src2)
{
        return((Word16)src1 * (UWord16)src2);
}

/*************************************************************************/
/* _mpyus C function                                                     */
/*************************************************************************/
Word32 mpyus_c(UWord32 src1,Word32 src2)
{
        return((UWord16)src1 * (Word16)src2);
}

/*************************************************************************/
/* _mpyu C function                                                      */
/*************************************************************************/
Word32 mpyu_c(UWord32 src1, UWord32 src2)
{
        return((UWord16)src1 * (UWord16)src2);
}

/*************************************************************************/
/* _mpyh C function                                                      */
/*************************************************************************/
Word32 mpyh_c(Word32 src1,Word32 src2)
{
 return((Word16)( (src1&0x0ffff0000) >> 16) * (Word16)( (src2&0x0ffff0000) >> 16));
}

/*************************************************************************/
/* _mpyhus C function                                                    */
/*************************************************************************/
Word32 mpyhus_c(UWord32 src1,Word32 src2)
{
 return((UWord16)( (src1&0x0ffff0000) >> 16) * (Word16)( (src2&0x0ffff0000) >> 16));
}
```

```
/*************************************************************************/
/* _mpyhsu C function                                                    */
/*************************************************************************/
Word32 mpyhsu_c(Word32 src1,UWord32 src2)
{
return((Word16)( (src1&0x0ffff0000) >> 16) * (UWord16)( (src2&0x0ffff0000) >> 16));
}

/*************************************************************************/
/* _mpyhu C function                                                     */
/*************************************************************************/
Word32 mpyhu_c(UWord32 src1,UWord32 src2)
{
  return((UWord16)( (src1&0x0ffff0000) >> 16) *(UWord16)( (src2&0x0ffff0000) >> 16));
}

/*************************************************************************/
/* _mpyhl C function                                                     */
/*************************************************************************/
Word32 mpyhl_c(Word32 src1,Word32 src2)
{
        return(  (Word16)((src1&0x0ffff0000) >> 16) * (Word16)src2 );
}

/*************************************************************************/
/* _mpyhuls C function                                                   */
/*************************************************************************/
Word32 mpyhuls_c(UWord32 src1,Word32 src2)
{
        return(  (UWord16)((src1&0x0ffff0000) >> 16) * (Word16)src2 );
}

/*************************************************************************/
/* _mpyhslu C function                                                   */
/*************************************************************************/
Word32 mpyhslu_c(Word32 src1,UWord32 src2)
{
        return(  (Word16)((src1&0x0ffff0000) >> 16) * (UWord16)src2 );
}

/*************************************************************************/
/* _mpyhlu C function                                                    */
/*************************************************************************/
Word32 mpyhlu_c(UWord32 src1,UWord32 src2)
{
       return(  (UWord16)((src1&0x0ffff0000) >> 16) * (UWord16)src2 );
}
```

```
/*************************************************************************/
/* _mpylh C function                                                     */
/*************************************************************************/
Word32 mpylh_c(Word32 src1,Word32 src2)
{
        return((Word16)src1 * (Word16)((src2&0x0ffff0000) >> 16)  );
}

/*************************************************************************/
/* _mpyluhs C function                                                   */
/*************************************************************************/
Word32 mpyluhs_c(UWord32 src1,Word32 src2)
{
        return((UWord16)src1 * (Word16)((src2&0x0ffff0000) >> 16)  );
}

/*************************************************************************/
/* _mpylshu C function                                                   */
/*************************************************************************/
Word32 mpylshu_c(Word32 src1,UWord32 src2)
{
        return((Word16)src1 * (UWord16)((src2&0x0ffff0000) >> 16)  );
}

/*************************************************************************/
/* _mpylhu C function                                                    */
/*************************************************************************/
Word32 mpylhu_c(UWord32 src1,UWord32 src2)
{
        return((UWord16)src1 * (UWord16)((src2&0x0ffff0000) >> 16));
}

/*************************************************************************/
/* _norm C function                                                      */
/*************************************************************************/
#define MASK31 (int)0x80000000

UWord32 norm_c(Word32 src2)
{
Word32 i=0;
        if( !(src2 & MASK31)) {               /* Positive Number */
                    while((src2 & MASK31) != MASK31) {
                              src2 = src2 << 1;
                              i++;
                              if(i==32) return(31);
                    }
        }
        else {                                /* Negative Number */
                    while((src2 & MASK31) != 0) {
                              src2 = src2 << 1;
                              i++;
                              if(i==32) return(31);
                    }

        }
        return(i-1);
}
```

```
/*************************************************************************/
/* _sadd C function                                                      */
/*************************************************************************/
Word32 sadd_c(Word32 src1, Word32 src2)
{
Word32 result;
        result = src1 + src2;

        if( ((src1 ^ src2) & MIN_32) == 0) {
                    if((result ^ src1) & MIN_32) {
                                result = (src1<0) ? MIN_32 : MAX_32;
                                SAT_Bit = 1;
                    }
        }
        return(result);
}

/*************************************************************************/
/* _sat C function                                                       */
/*************************************************************************/
Word32 sat_c(Word40 src2)
{
        if(src2 > MAX_32) {
                    return(MAX_32);
                    SAT_Bit = 1;
        }
        else {
                    if(src2 < MIN_32) {
                                SAT_Bit = 1;
                                return(MIN_32);
  }
                    else {
                                return(src2);
                    }
 }
}

/*************************************************************************/
/* _set C function                                                       */
/*************************************************************************/
UWord32 set_c(UWord32 src1,UWord32 csta,UWord32 cstb)
{
Word32 i;
Word32 maska=0x0fffffff;
Word32 maskb=0x01;
        if(csta<=cstb) {
                    for(i=0;i<csta;i++)
                                maska = maska << 1;

                    for(i=0;i<cstb;i++)
                                maskb = (maskb << 1) + 1;

                    maska = maska & maskb;

                    return(maska | src1);
        }
```

```
            else{
                        return(src1);
            }
    }
```

```
/*************************************************************************/
/* _smpy C function                                                      */
/*************************************************************************/
Word32 smpy_c(Word32 src1,Word32 src2)
{
Word32 result;
        result = ((Word16)src1 * (Word16)src2) << 1;
        if (result != MIN_32)
                    return(result);
        else {
                    return(MAX_32);
                    SAT_Bit = 1;
        }
}


/*************************************************************************/
/* _smpyh C function                                                     */
/*************************************************************************/
Word32 smpyh_c(Word32 src1,Word32 src2)
{
Word32 result;

        result = ( (Word16)((src1&0x0ffff0000) >> 16) *
               (Word16)((src2&0x0ffff0000) >> 16)) << 1;

        if ( result != MIN_32)
                    return(result);
        else {
                    SAT_Bit = 1;
                    return(MAX_32);
        }
}

/*************************************************************************/
/* _smpyhl C function                                                    */
/*************************************************************************/
Word32 smpyhl_c(Word32 src1,Word32 src2)
{
Word32 result;
        result = ( (Word16)((src1&0x0ffff0000) >> 16) * (Word16)src2) << 1;
        if (result != MIN_32)
                    return(result);
        else {
                    return(MAX_32);
                    SAT_Bit = 1;
        }
}
```

```
/*************************************************************************/
/* _smpylh C function                                                    */
/*************************************************************************/
Word32 smpylh_c(Word32 src1,Word32 src2)
{
Word32 result;
        result = ((Word16)src1 * (Word16)((src2&0x0ffff0000) >> 16)) << 1;

        if ( result != MIN_32)
                    return(result);
        else {
                    return(MAX_32);
                    SAT_Bit = 1;
        }
}

/*************************************************************************/
/* _sshl C function                                                      */
/*************************************************************************/
Word32 sshl_c(Word32 src2, UWord32 src1)
{
Word32 i;
        src1=src1&0x01f;   /* Consider only the five least significant bits */
        if( norm_c(src2) >= src1) {
                    for(i=0; i<src1 ;i++)
                                src2 = src2 << 1;
                    return(src2);
        }
        else {
                    return((src2>0) ? MAX_32 : MIN_32);
                    SAT_Bit = 1;
        }
}


/*************************************************************************/
/* _ssub C function                                                      */
/*************************************************************************/
Word32 ssub_c(Word32 src1, Word32 src2)
{
Word32 result;
        result = src1 - src2;

        if( ((src1 ^ src2) & MIN_32) != 0)        {
                    if((result ^ src1) & MIN_32)   {
                                result = (src1<0) ? MIN_32 : MAX_32;
                                SAT_Bit = 1;
                    }
        }
        return(result);
}
```

```
/*************************************************************************/
/* _sub2 C function                                                      */
/*************************************************************************/
Word32 sub2_c(Word32 src1, Word32 src2)
{
Word32 temp, temp1;
        return( (((src1 & 0x0ffff) - (src2 & 0x0ffff)) & 0x0ffff) |
        ( ( ((src1 & 0xffff0000) >> 16) - ((src2 & 0xffff0000) >> 16) ) << 16)
);
}

/*************************************************************************/
/* _subc C function                                                      */
/*************************************************************************/
UWord32 subc_c(UWord32 src1, UWord32 src2)
{
        if( src1 >= src2)
                    return( ((src1-src2) << 1) + 1);
        else
                    return(src1<<1);
}
#endif

/*************************************************************************/
* Cintrins.h  header file for the C implementation of the                */
/*************************************************************************/
#include ”types.h”
#define MIN_32 (Word32)0x80000000
#define MAX_32 (Word32)0x7fffffff
#define MIN_40 (Word40)0x8000000000
#define MAX_40 (Word40)0x7fffffffff

#if!defined(_TMS320C6200)

#define  _abs             abs_c
#define  _add2            add2_c
#define  _clr             clr_c
#define  _ext             ext_c
#define  _extu            extu_c
#define  _lmbd            lmbd_c
#define  _norm            norm_c
#define  _lnorm           lnorm_c
#define  _sadd            sadd_c
#define  _lsadd           lsadd_c
#define  _ssub            ssub_c
#define  _lssub           lssub_c
#define  _sat             sat_c
#define  _set             set_c
#define  _sshl            sshl_c
```

```
#define   _sub2          sub2_c
#define   _subc          subc_c
#define  _mpy            mpy_c
#define  _mpyus          mpyus_c
#define  _mpysu          mpysu_c
#define  _mpyu           mpyu_c
#define  _mpyh           mpyh_c
#define  _mpyhus         mpyhus_c
#define  _mpyhsu         mpyhsu_c
#define  _mpyhu          mpyhu_c
#define  _mpylh          mpylh_c
#define  _mpyluhs        mpyluhs_c
#define  _mpylshu        mpylshu_c
#define  _mpylhu         mpylhu_c
#define  _mpyhl          mpyhl_c
#define  _mpyhuls        mpyhuls_c
#define  _mpyhslu        mpyhslu_c
#define  _mpyhlu         mpyhlu_c
#define  _smpy           smpy_c
#define  _smpyh          smpyh_c
#define  _smpylh         smpylh_c
#define  _smpyhl         smpyhl_c

Word32 abs_c(Word32 src1);
Word32 add2_c(Word32 src1, Word32 src2);
UWord32 clr_c(UWord32 src1, UWord32 csta, UWord32 cstb);
Word32 ext_c(Word32 src1, UWord32 csta, UWord32 cstb);
UWord32 extu_c(UWord32 src1, UWord32 csta, UWord32 cstb);
UWord32 lmbd_c(UWord32 src1,UWord32 src2);
UWord32 lnorm_c(Word40 src2);
Word40 lsadd_c(Word32 src1, Word40 src2);

Word32 mpy_c(Word32 src1,Word32 src2);
Word32 mpyus_c(UWord32 src1,Word32 src2);
Word32 mpysu_c(Word32 src1,UWord32 src2);
Word32 mpyu_c(UWord32 src1,UWord32 src2);

Word32 mpyh_c(Word32 src1,Word32 src2);
Word32 mpyhus_c(UWord32 src1,Word32 src2);
Word32 mpyhsu_c(Word32 src1,UWord32 src2);
Word32 mpyhu_c(UWord32 src1,UWord32 src2);

Word32 mpyhl_c(Word32 src1,Word32 src2);
Word32 mpyhuls_c(UWord32 src1,Word32 src2);
Word32 mpyhslu_c(Word32 src1,UWord32 src2);
Word32 mpyhlu_c(UWord32 src1,UWord32 src2);

Word32 mpylh_c(Word32 src1,Word32 src2);
Word32 mpyluhs_c(UWord32 src1,Word32 src2);
Word32 mpylshu_c(Word32 src1,UWord32 src2);
Word32 mpylhu_c(UWord32 src1,UWord32 src2);
```

```
UWord32 norm_c(Word32 src2);
Word32 sadd_c(Word32 src1, Word32 src2);
Word32 sat_c(Word40 src2);
UWord32 set_c(UWord32 src1,UWord32 csta,UWord32 cstb);

Word32 smpy_c(Word32 src1,Word32 src2);
Word32 smpyh_c(Word32 src1,Word32 src2);
Word32 smpyhl_c(Word32 src1,Word32 src2);
Word32 smpylh_c(Word32 src1,Word32 src2);
Word32 sshl_c(Word32 src2, UWord32 src1);
Word32 ssub_c(Word32 src1, Word32 src2);
Word32 sub2_c(Word32 src1, Word32 src2);
UWord32 subc_c(UWord32 src1, UWord32 src2);
#endif
```

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265