TMS320C54x DSP Programming Environment

APPLICATION BRIEF: SPRA182

M. Tim Grady
Senior Member, Technical Staff
Texas Instruments
April 1997

TEXAS INSTRUMENTS

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 application 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.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

Contents

Figures

Examples

TMS320C54x DSP Programming Environment

Abstract

This document describes some of the programming techniques that may be used in different applications to take advantage of the full feature set of the TMS320C54x high-performance fixed-point DSP family.

1. Introduction

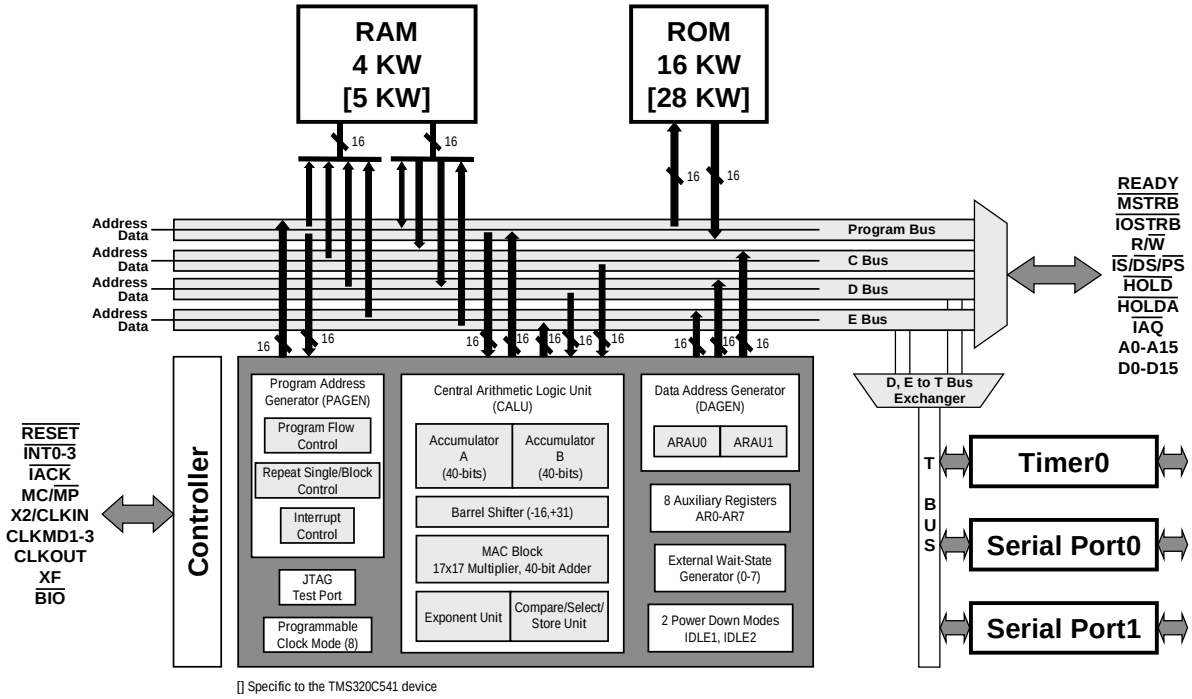
The TMS320C54x DSP family began as the Low-Power Enhanced Architecture DSP (LEAD) project. The project goal was a fixed-point DSP with power-saving characteristics well suited for the cellular telephone market. The result is a multi-bus design with special CPU features such as two accumulators and dual addressing modes that support the design goals.

The TMS320C54x architecture is a departure from the C2x/C5x family, but shares many peripherals and interface features with its cousins. The TMS320C54x User's Guide gives a detailed description of the architecture. This application brief discusses elements of the programming style used with this processor.

2. Basic Architecture

Many DSPs are based around a Harvard architecture with separate data and program memories and associated buses. The TMS320C54x devices share this feature, but additional data buses embellish the architecture and improve throughput. See Figure 1.

Figure 1. TMS320C54x block diagram



Internally, there are several buses:

- The P-Bus, used to fetch instructions from program memory, is also connected to the multiplier to provide an input from program memory.
- A dual data-bus scheme, the C-Bus and D-Bus, permits fetching two operands in the same cycle or a double word in one cycle.
- A separate output data bus, the E-Bus, allows simultaneous reads and writes in one parallel instruction.

3. Advantages of Multiple Buses

The dual data buses provide two address generators for fetching operands. These accesses are via the eight auxiliary registers (AR0 to AR7) by indirect addressing. For example, if AR2 points to a table of coefficients and AR3 points to a data array, the multiply can look like:

```
MPY     *AR2,*AR3,A
```

This is a single-cycle instruction. Two buses, two address generators, and straightforward use of the auxiliary registers, result in a clean instruction set. With the TMS320C54x, there is no need to manage the auxiliary register pointer. One simply loads an address into the register and proceeds.

Using the P-bus to read one operand enables coefficients to be stored in program memory. For example, if the coefficients are in ROM in PROGRAM space and the programmer wants to loop through a simple vector multiply, filter-type problem, the coefficient array can be in program space as:

```
      .text
coef  .word   1,2,3,4,...   ; data in program space
```

In the following example, the data are in data memory with an auxiliary register (AR2) pointing to the array. The multiply and adds can then be performed in a single-instruction repeat loop:

```
RPTZ    A,15
MACP    *AR2+,coef,A
```

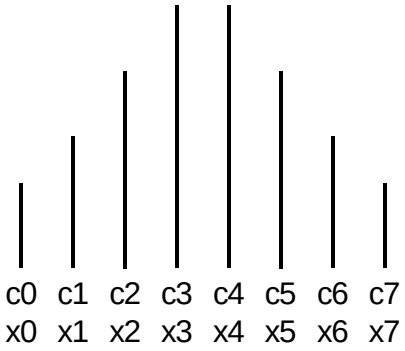
The plus (+) sign following the data pointer is a post-increment of the address to access the next value in the data array.

The plus sign for the coefficient is implied. The pointer to the address in the coefficient array is stored in a register in the program counter (PC) controller. Accessing the coefficient array automatically increments the pointer, so no symbol is used. The RPTZ instruction zeroes accumulator A and repeats the next instruction 15 + 1 = 16 times.

4. Special Instructions

Several instructions specific to the TMS320C54x improve device performance. For example, the FIRS, or symmetric FIR filter instruction uses the P-bus, both data buses and both accumulators. Figure 2 illustrates the filter with a set of symmetric coefficients.

Figure 2. Symmetric filter



In a typical FIR filter, the y values are the sum of the products:

```
y = (c0 * x0) + (c1 * x1) + (c2 * x2) + (c3 * x3) +
(c4 * x4) + ...+ (c7 * x7)
```

Since this is a symmetric filter (c0=c7, c1=c6, etc.) the equation can be simplified to:

```
y = c0*(x0 + x7) + c1*(x1 + x6) + c2*(x2 + x5) +
c3*(x3 + x4)
```

The simplification results in half the number of multiplies.

The FIRS instruction uses the P-bus to point to the coefficients and the two data buses to point to the data pairs:

```
FIRS     *AR2+,*AR3+,coef
```

The FIRS instruction multiplies accumulator A by the value in the coefficient matrix and adds the result to accumulator B. At the same time, it adds the memory operands pointed to by auxiliary registers AR2 and AR3 and places the sum in accumulator A. Accumulator A must be pre-loaded with the first sum.

The TMS320C54x performs an N-tap symmetric FIR filter in N/2 + x cycles, where x is the overhead for setting up the loop. This is a big improvement over most DSPs, which require N + x cycles for the same operation. The TMS320C54x multiple buses, two accumulators, and special instruction hardware make this possible.

Later examples illustrate some of the other special instructions.

5. Addressing Modes

The TMS320C54x has immediate, direct (page + offset), and indirect addressing.

5.1 Immediate Addressing

Immediate is very simple; for example, to place 15h in accumulator A:

```
LD      #15h,A
```

There are many options, such as load with shifts. Other instructions allow immediate addressing with an ALU operation, for example:

```
ANDM    00ffh,*AR1
```

This particular example also uses a non-accumulator destination; that is, ANDing a constant with a value in memory and placing the result in memory. This makes a PLU unnecessary for bit manipulation, and the TMS320C54x does not have one.

5.2 Direct Addressing

Direct addressing requires the use of a data page pointer (DP) and an offset into the 128-word page. While direct addressing has its uses, most algorithms use indirect addressing and the auxiliary registers.

Example 1. Direct addressing example:

```
.mmregs
   .global  x,y,Entry
   .bss     x,128,1     ; put "x" on single page

   .sect    "program"
   LD       #0,A        ; zero accumulator A
   LD       #x,DP       ; set DP to page with "x"
   ADD      #1,A,A      ; add 1 to accumulator
   STL      A,@x        ; store accumulator A at "x"
   ADD      #1,A,A
   STL      A,@x+128    ; Wrap back to start of same
                        ;                 page
                        ; (modulo 128)
```

5.3 Indirect Addressing

The indirect addressing modes of the TMS320C54x allow many options. For example, *ARn simply accesses the value ARn points to. On the other hand, *ARn+, *ARn+0, and *ARn+0% respectively, provide post-increment, post-increment with variable step size, and circular buffering. The TMS320C54x offers post-increment for reads and either pre- or post-increment for writes.

Example 2. Indirect addressing example (with indexing):

```
.mmregs
   .global  x,y,Entry
   .bss     x,128,1        ;put "x" on single page

   .sect    "program"
   STM      #2,AR0         ;set index value to 2.
                           ; AR0 == index register
   STM      #x+126,AR1     ;start near end of page
                           ; "x"
   ADD      #1,A,B
   STL      B,*AR1+        ;store at x+126
   ADD      #1,B,A
   STL      A,*AR1+0       ;store at AR1+2*(AR1+2)=A
   ADD      #1,A,B
   STL      B,*(#x+140)    ;store at x+140
                        ;(absolute address plus index)
```

5.4 Accessing Memory-Mapped Registers

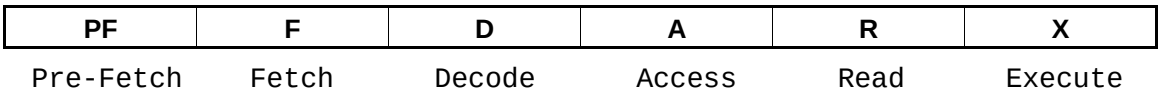
The TMS320C54x has a Memory-Mapped Register file (MMR). All the peripheral control registers are memory mapped. Since the MMRs are all on Data Page Zero, special rules allow register access without first modifying the DP pointer. Typically, there is a one-cycle penalty for using an MMR as an operand; however, the one-cycle penalty is less than the overhead of managing the DP and the associated lines of code:

```
ADD   DRR,0,A        ;two-cycle instruction (DRR ==
                     ; Serial Port data register)
ADD   A,-8,B         ;one-cycle instruction
ORM   0f0fh,SPC      ;three-cycle instruction (SPC ==
                     ;Serial Port Control register)
ORM   0f0fh,*AR4+    ;two-cycle instruction
```

6. Pipeline

The TMS320C54x has a six-level pipeline as shown in Figure 3. The pipeline provides very fast throughput, but requires some attention to detail in programming.

Figure 3. The TMS320C54x pipeline



Most values change at the execution phase. Some changes occur at other times, such as auxiliary register (AR) updates during the access phase. This allows a sequence of instructions such as:

```
ADD     *AR1+,A
MPY     *AR1,#07h,B
```

The second instruction requires AR1 to have the updated value before its access phase. If AR1 were only updated at the execution phase of the first instruction, the second instruction would have to wait two cycles before its access phase could start. Two NOPs or other valid instructions would be required between the two instructions. However, since AR1 is updated during the access phase of the first instruction, the second instruction executes without difficulty or additional code (see below).

Figure 4. Auxiliary register updates

ADD	**D**	**A**	**R**	**X**	
		AR1 modified here			
MPY		**D**	**A**	**R**	**X**
			New AR1 used here		

6.1 Pipeline Conflicts

The pipeline can be an issue in some operations. If both the C- and D-buses are in use on a dual operand instruction, which overlaps a previous instruction that accesses the E-bus at the same time, conflict may occur.

For the most part, however, conflicts are unlikely. The access phase for read occurs in the access phase of the pipeline, while the access phase for write occurs during the read phase of the pipeline. Further, the access for the D-bus occurs during only the first half of the cycle, while the access for the E-bus occurs during the second half of the cycle. So, even if two instructions overlap for the same cycle, conflict is minimized by the two address generators occurring in the same cycle.

The following example shows two instructions that avoid a pipeline conflict:

```
STL     A,*AR3+         ; instruction 1
ADD     *AR4,*AR5,A     ; instruction 2
```

Figure 5. Multiple buses avoid pipeline conflicts

Instruction 1	**A**		**R**		**X**					
						E-bus used				
Instruction 2			**A**			**R**			**X**	
				C-bus used		D-bus used				

In the pipeline, the write to the E-bus occurs during the second half of the execute phase. The C-bus and the D-bus load during the second half of the access phase and the first half of the read phase, respectively.

6.1.1 Resolved Conflict

A conflict may occur if a write is followed by two dual-access reads, the second of which is from the same memory block as the write.

```
STL     A,*AR3+
ADD     AR4+,*AR5+,A
MPY     AR2+,AR3,B
```

Figure 6. Potential pipeline conflict

STL	**A**		**R**		**X**							
				E Addr		E Data						
ADD	**D**		**A**		**R**		**X**					
		C Addr	D Addr	C Data	D Data							
MPY			**D**		**A**		**R**		**X**			
				C Addr	D Addr	C Data	D Data					

The conflict only occurs between the store instruction (STL) and the multiply instruction (MPY) if the accesses via the E-bus and C-bus are to the same block of memory. Otherwise, no conflict occurs.

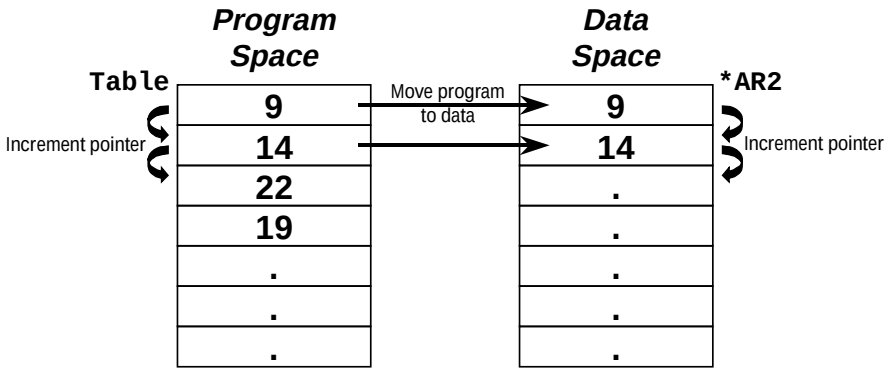
If a conflict does occur, it is automatically resolved by the CPU delaying the write operation of the STL instruction to the execute phase of the ADD instruction. No extra cycles are inserted. If the data from the write operation is required before it is written to memory, it is read directly from an internal bus.

7. Loops and Control Structures

The TMS320C54x has several loop-control mechanisms, including a repeat single and a repeat block. Both may be asserted at the same time, allowing nested loops. A single instruction may be repeated N times, as:

```
RPT   #16–1          ;load repeat counter with 15
                     ; for 16 iterations
MVPD  #table,*AR2+   ; copy value from data list
                     ;pointed to by table
```

Figure 7 – Moving values from program space to data space



This example also illustrates the move from program space to data space command. The pointer to Table is managed and automatically incremented by the program address generation (PAGEN) unit, so must be inside a repeat single. No instructions are fetched during a repeat single, which makes it a great candidate for use when fetching data from program space. In addition, the repeat single may not be interrupted.

A second version of the repeat single is the RPTZ, which first writes a zero to one of the accumulators. For example, to accumulate a sum of products into accumulator A, with N iterations:

```
RPTZ     A,#N-1
MACP     *AR2-%,Table,A
```

The repeat block command allows including a series of instructions in an automatic loop structure. The block repeat counter (BRC) must be loaded before beginning the loop. This may also be a delayed instruction with two delay slots:

```
STM      #N-2, BRC      ;initialize the Block
                  ; repeat counter with Store MMR
LD       ERROR,T        ;initialize a variable
                        ; called ERROR to 0
```

```
   RPTBD    END_LOOP-1     ;establish last line of
                           ; loop
   MPY      *AR4,A         ;delay slot instruction
                     ; that initializes accumulator A
   LMS      *AR3,*AR4+0%   ;2nd delay slot instruction
                           ;loop starts here
   ST       A,*AR3+        ;save filter coefficient
|| MPY      *AR4,A         ;new term calculated
   LMS      *AR3, *AR4+0%  ;LMS instruction for
                           ; adaptive filter
                           ;A = A + *AR3<<16 + 2^15
                           ;B = B + *AR3 x *AR4
END_LOOP:
```

This example also illustrates two other ideas: the parallel instruction and the delayed instruction.

The ST || MPY is a parallel instruction made possible by the separate and parallel units within the TMS320C54x.

The RPTBD instruction is a delayed version of a block repeat, which has two delay slots. The delay slots are available for tasks such as initializing parts of the loop operation.

The LMS instruction is one of the special instructions used in adaptive filtering algorithms. It makes use of both accumulators and the parallel bus structure.

8. Interrupts – Special Features

The TMS320C54x device uses interrupts much like other DSPs and processors, but there are additional features. The TMS320C54x has four external interrupts and the usual assortment of internal interrupts including serial port, timer and traps. There are two interesting features of TMS320C54x interrupts: the way the vector table is built and a special fast interrupt feature.

The vector table occupies four words. Thus, when an interrupt is taken and the PC switches to the vector table, it is possible to have a traditional branch instruction such as:

```
BR              ;1st entry
isr-1           ;2nd entry
not used        ;3rd entry
not used        ;4th entry
```

This results in two unused instructions. If, however, the first entry is a delayed return statement, the two delay slots could be used:

```
RETED           ;1st entry
LDM     DRR,A   ;2nd entry, first delay slot
                ;3rd entry, 2nd delay slot.
                ;Note: MMRs take 2 cycles
not used        ;4th entry not used because
                ; transfer has occurred
```

The RETE instruction requires five cycles, while the RETED takes three cycles and automatically re-enables the global interrupts. Using the four vector table slots is faster than the typical overhead of branching to an ISR and returning. An even faster method is to use the RETF instruction:

```
RETFD           ; 1st entry
ADD   1, *AR4   ; 2nd delay slot
LD    #5, ASM   ; 3rd delay slot
not used        ; 4th entry
```

The RETFD instruction takes only one cycle and speeds short interrupt routine handling. The interrupt latency for a fast interrupt is very low. Its speed is made possible by a special register in the PAGEN unit in which the return address is stored.

9. More on Special Instructions

Examples of the FIRS instruction and the LMS instruction are shown earlier in this article. Several other special instructions are supported by hardware, including a series of double add and double subtract instructions which, combined with the Compare, Select and Store instruction (CMPS), provide excellent throughput on algorithms such as the Viterbi decoder.

A concept in Viterbi decoding is to compare two metrics to determine and store the larger distance. In addition, a record of which metric was stored is required. The TMS320C54x CMPS unit performs this function.

With either accumulator as an input, the CMPS compares the two halves and writes one to memory. It also updates the transition (TRN) register by shifting the contents to the left one place and ORing in a new value (either 1 or 0, depending on which half is updated).

The accumulator used as input to the CMPS has two 16-bit values calculated by two parallel adds or subtracts. A value is placed in the T register, and the AR register points to a double word. Half the double word is either ADDed or SUBtracted from the T register and the two results are stored in the 32-bit accumulator.

For example:

```
STM     #2-1, BRC   ;initialize repeat counter
RPTBD   end-1       ;begin repeat block
LD      *AR7,T      ;initialize T in delay slot
NOP                 ;second delay slot
DADST   *AR5, A     ;A(h) = OLDValue[i] + T
                    ;A(l) = OLDValue[j] – T
              ;DADST means double add and subtract
DSADT   *AR5+,B     ;B(h) = OLDValue[i] – T
                    ;B(l) = OLDValue[j] + T
              ;DSADT means double subtract and add
CMPS    A,*AR4+%    ;Write one-half of A to
                    ; memory and update TRN
CMPS    B,*AR3+%    ;Write one-half of B to
                    ; memory and update TRN
```

While not a complete algorithm, this segment illustrates the use of some of the special instructions. All the instructions shown are single-cycle instructions.

10. Summary

The TMS320C54x device is a fast, low-power fixed-point machine with performance enhancements. It features dual data buses and dual address generators to fetch dual operands or double words in one cycle. Two accumulators and hardware support double adds and/or subtracts. The P-bus is used to fetch instruction words, but may also be an input to the multiplier. This simplifies several instructions that use data buses for operands and the program bus for a third operand.

The TMS320C54x includes a compare, select and store unit, which improves the performance of GSM and Viterbi algorithms. Hardware support for special instructions such as the LMS instruction used in adaptive filter algorithms, further enhance the device.

The TMS320C54x may also be used as a DSP in non-telecom applications. The loop control structures, addressing modes, and fast interrupt features enable fast, compact code.