

TMS320C6000 Non-Power of 2 Circular Buffers

*Eric Wilbur
Yao-Ting Cheng*

Digital Signal Processing Solutions

ABSTRACT

The TMS320C6000™ circular buffer block size must be specified as a power of 2. However, some filters require coefficient/data table sizes which are not a power of 2. This application report shows an easy work around to implement the circular buffer with any block size.

Contents

1	Design Problem	1
2	Solution	1
	2.1 Procedure	2

List of Figures

Figure 1.	Pointers to the Beginning of the Circular Buffer	2
Figure 2.	Pointer Increment and Reset	3
Figure 3.	Pointer Increment and Reset	3
Figure 4.	Pointer Increment and Reset	4

List of Examples

Example 1.	Code Listing	5
------------	--------------------	---

TMS320C6000 is a trademark of Texas Instruments.

1 Design Problem

Eight C6x registers (A4~A7, B4~B7) can be used for circular addressing. The block size (or buffer size) must be specified as a power of 2 (i.e. 2ⁿ). However, some filters require coefficient/data table sizes which are not a power of 2. So, how then can you implement these algorithms on the C6x? One method to solve the problem is provided below.

2 Solution

Let's start with an example of an FIR filter with 6 16-bit coefficients/samples. The code to implement this solution is given at the end of the document.

$$Y = \sum_{i=1}^6 X_i \cdot C_i \tag{1}$$

This solution does waste some RAM (the 2ⁿ blk size minus the actual table size, in this case: 8-6 = 2 locations), but without any speed penalties. The basic principle is that you create a "hole" of unused samples that circulates through memory. The benefit is that once you set up your tables correctly, the code is very straightforward.

2.1 Procedure

1. Arrange the input samples in incrementing order such as X1,X2,...,X6 and the coefficients in descending order such as C6,C5,...,C1.
2. Set the block size of the circular buffer to the 2^n size greater than the actual table size, select circular register and initialize AMR. In our example, the block size will be 16 bytes or 8 shorts (16-bit). A4 will be used in circular mode (as the sample pointer), therefore AMR= 0x00030001.
3. Initialize the circular buffer pointer A4 to the top of the input samples (i.e. A4=&X1).
4. Initialize another pointer (using linear mode), to the top of the coefficients. In our example B4=&C1.

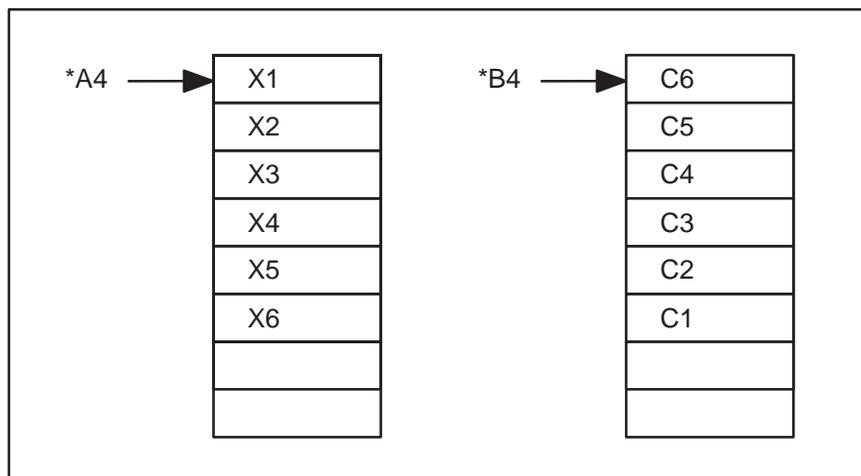


Figure 1. Pointers to the Beginning of the Circular Buffer

5. Set the loop counter to be the number of coefficients minus one. So, B0=5. This might look like a problem because your loop will only execute 5 times instead of 6. However, see step 7 for the solution.
6. Load the value from *A4++ and *+B4[B0] to registers and perform a MAC. As you can see, the count value is being used as a pre-offset to the base of the coefficient table. Therefore, the code will access C1 first, then C2, etc.
7. In the MAC loop, the counter should be decremented, looping a total of 6 times. However, please note that the SUB instruction should occur one cycle AFTER the branch or the code will only loop 5 times. This method allows the code to use B0 (the count value) as a pre-offset for the coefficients. If you are pipelining your algorithm, the same concept applies.

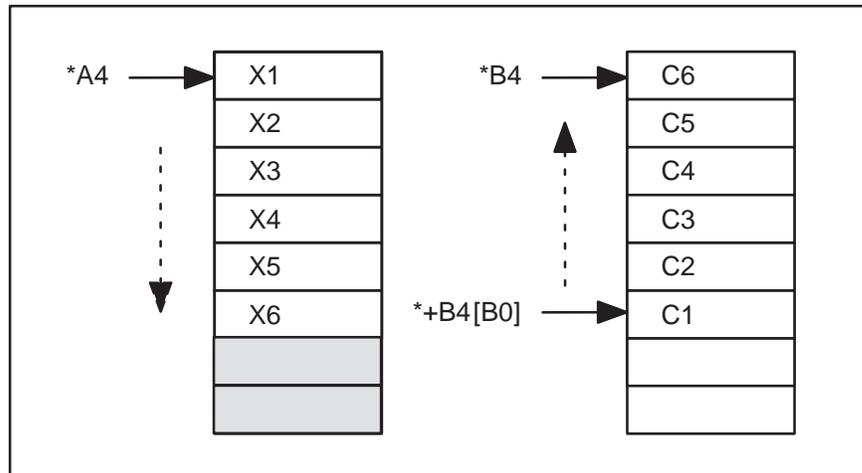


Figure 2. Pointer Increment and Reset

8. Get a new input sample. After the MAC operation, the circular register (e.g. A4) is pointing to the next “empty” location to be filled. When loading a new value into the sample table, if you post increment by $(2^n - \text{actual table size} + 1)$ (in our example this would be $8 - 6 + 1 = 3$), the pointer (A4) will be set to the new “top” of the buffer (the oldest sample).
9. Reset the circular pointer to point to the oldest sample. This can be done separately or as part of loading the new sample (as described in step 8).

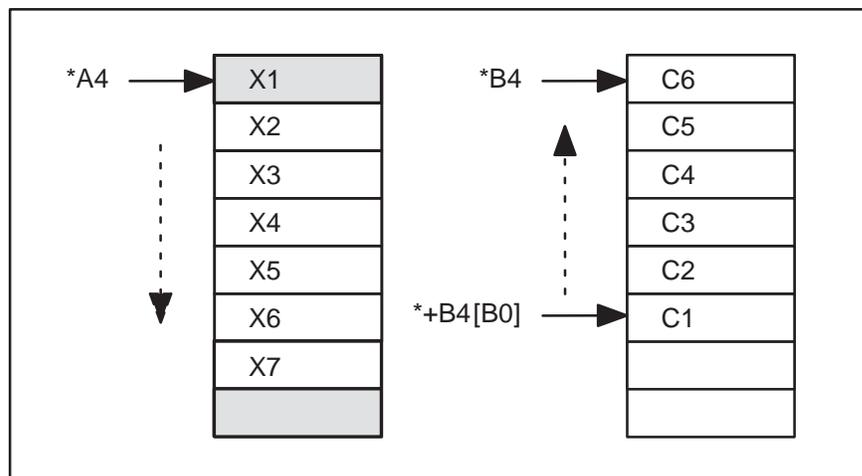


Figure 3. Pointer Increment and Reset

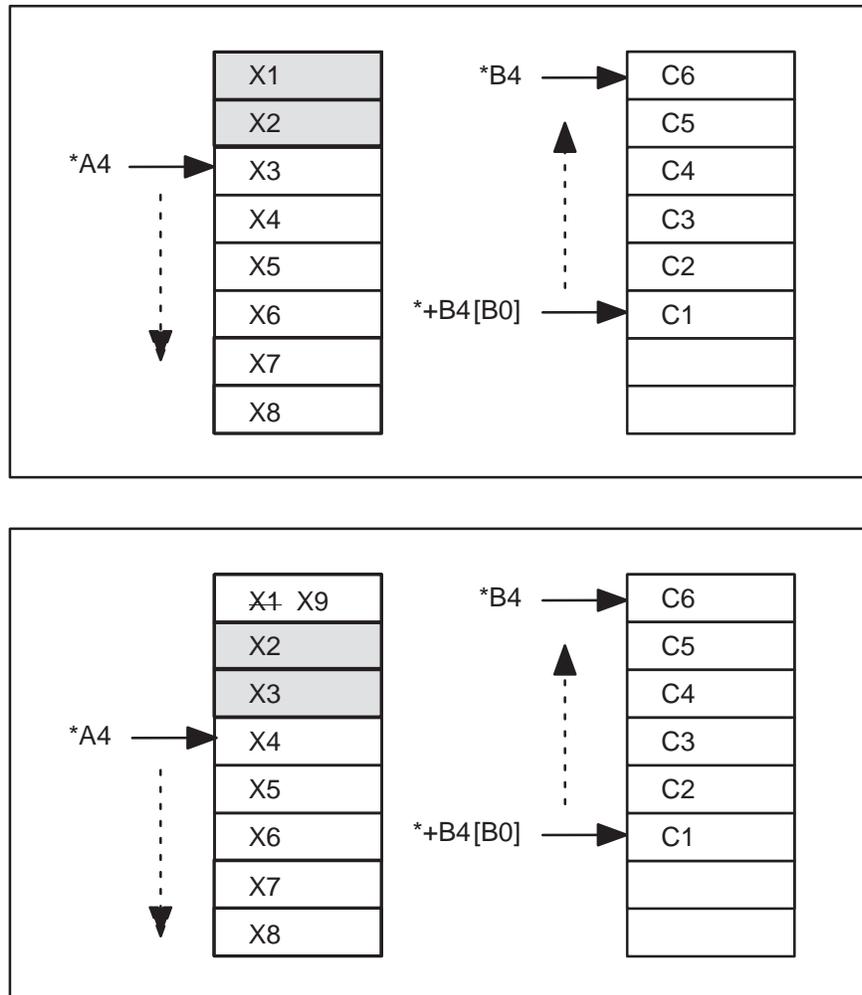


Figure 4. Pointer Increment and Reset

10. Go to step 5. As you can see in the diagram above, a “hole” of unused samples is created inside the buffer. Let’s say X9 is the newest sample. The filter will run from X4 through X8 and then X9. In this case, X2 and X3 locations were NOT used. As the filter continues, this “hole” of size 2 moves in a circular fashion. So, when X10 comes in, it overwrites X2, the filter runs from X5–X10 and X3/X4 locations are not used, etc. There is no wasted space in the coefficient table.
11. Note: This example does not include any software pipelining. However, the same concept can be applied to a s/w-pipelined loop. The goal was to provide a methodology for dealing with an algorithm using a block size which is not a power of 2.

Example 1. Code Listing

```

        .title  "fir.asm"           ; non 2^n Circular Buffer
        .def    init
half    .equ    2                   ; half is defined as 2 bytes
        .data
coeff   .short  6,5,4,3,2,1       ; FIR filter coefficients
initsmp .short  1,2,3,4,5,6       ; input samples
new_smp .short  7,8,9,0xa         ; new input samples
        .bss   sample,8*half,8*half
        .bss   output,1*half,1*half
        .text
init_AMR:
        MVK    .S2    0x0001,B0 ; setup cir. buf AMR=0001_0003h
        MVKLNH .S2    0x0003,B0 ; setup cir. buf AMR=0001_0003h
        MVC    .S2    B0,AMR ; blk size is 16 bytes, pointer is A4
init_buffer: ; assume the variable has been initialized
init_pointers:
        MVK    .S2    new_smp,B7 ; init pointers
        MVKH   .S2    new_smp,B7
        MVK    .S2    coeff,B4
        MVKH   .S2    coeff,B4
        MVK    .S1    output,A9
        MVKH   .S1    output,A9
        MVK    .S1    sample,A4
        MVKH   .S1    sample,A4
fir:    ZERO    .L1    A7
        MVK    .S2    5,B0
loop:   LDH     .D1    *A4++,A5      ; load samples
        LDH     .D2    *+B4[B0],B5  ; load coefficients
        NOP
        MPY    .M1x   A5,B5,A6
        NOP
        ADD    .L1    A7,A6,A7
        [B0] B   .S1    loop
        [B0] SUB .L2    B0,1,B0
        NOP
        4
get_new_smp:
        LDH     .D2    *B7++,A8      ; get new sample
        STH     .D1    A7,*A9        ; store the result to output
        NOP
        3
        STH     .D1    A8,*A4--[5] ; store new sample to buffer
        ; and init circular pointer
        B       .S1    fir
        NOP
        5
        B       .S1    $
        NOP
        5
    
```

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products 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.

Customers are responsible for their applications using TI components.

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

TI assumes no liability for applications assistance or customer product design. TI does not 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. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.