

APPENDIX B BENCHMARK PROGRAMS

Table B-1 and Table B-2 provide benchmark numbers for 18 common DSP programs. The two tables are identical except that Table B-1 is for the 20.5-MHz DSP56001 and Table B-2 is for the 27-MHz DSP56001. The following four code examples (Figures B-1 to B-4) are representative of the benchmark programs shown in Tables B-1 and B-2. The code for these and other programs is free and available through the Dr. BuB electronic bulletin board. Figure B-1 is the code for the 20-tap FIR filter shown in Tables B-1 and B-2. Figure B-2 is the code for an FFT using a triple nested DO LOOP. Although this code is easier to understand and very compact, it is not as fast as the code used for the benchmarks shown in Tables B-1 and B-2, which are highly optimized using the symmetry of the FFT and the parallelism of the DSP. Figure B-3 is the code for the 8-pole cascaded canonic biquad IIR filter, which uses four coefficients (see Tables B-1 and B-2). Figure B-4 is the code for a 2N delayed least mean square (LMS) FIR adaptive filter, which is useful for echo cancelation and other adaptive filtering applications.

Table B-1 20.5-MHz Benchmark Results for the DSP56001R20

Benchmark Program	Sample Rate (Hz) or Execution Time	Memory Size (Words)	Number of Clock Cycles
20 - Tap FIR Filter	379.6 kHz	50	54
64 - Tap FIR Filter	144.4 kHz	138	142
67 - Tap FIR Filter	138.5 kHz	144	148
8 - Pole Cascaded Canonic Biquad IIR Filter (4x)	410.0 kHz	40	50
8 - Pole Cascaded Canonic Biquad IIR Filter (5x)	353.5 kHz	45	58
8 - Pole Cascaded Transpose Biquad IIR Filter	292.9 kHz	48	70
Dot Product	585.4 ns	10	12
Matrix Multiply 2x2 times 2x2	2.049 μ s	33	42
Matrix Multiply 3x3 times 3x1	1.659 μ s	29	34
M - to - M FFT 64 Point	129.5 μ s	489	2655
M - to - M FFT 256 Point	645.1 μ s	1641	13255
M - to - M FFT 1024 Point	3.231 ms	6793	66240
P - to - M FFT 64 Point	121.9 μ s	704	2499
P - to - M FFT 256 Point	458.2 μ s	2048	9394
P - to - M FFT 1024 Point	1.958 ms	7424	40144

Table B-2 27-MHz Benchmark Results for the DSP56001R27

Benchmark Program	Sample Rate (Hz) or Execution Time	Memory Size (Words)	Number of Clock Cycles
20 - Tap FIR Filter	500.0 kHz	50	54
64 - Tap FIR Filter	190.1 kHz	138	142
67 - Tap FIR Filter	182.4 kHz	144	148
8 - Pole Cascaded Canonic Biquad IIR Filter (4x)	540.0 kHz	40	50
8 - Pole Cascaded Canonic Biquad IIR Filter (5x)	465.5 kHz	45	58
8 - Pole Cascaded Transpose Biquad IIR Filter	385.7 kHz	48	70
Dot Product	444.4 ns	10	12
Matrix Multiply 2x2 times 2x2	1.556 μ s	33	42
Matrix Multiply 3x3 times 3x1	1.259 μ s	29	34
M-to-M FFT 64 Point	98.33 μ s	489	2655
M-to-M FFT 256 Point	489.8 μ s	1641	13255
M-to-M FFT 1024 Point	2.453 ms	6793	66240
P-to-M FFT 64 Point	92.56 μ s	704	2499
P-to-M FFT 256 Point	347.9 μ s	2048	9394
P-to-M FFT 1024 Point	1.489 ms	7424	40144

page 132,66,0,6
opt rc

```

*****
;Motorola Austin DSP Operation   June 30, 1988
*****
;DSP56000/1
;20 - tap FIR filter
;File name: 1-56.asm
*****
; Maximum sample rate: 379.6 kHz at 20.5 MHz/500.0 kHz at 27.0 MHz
; Memory Size: Prog: 4+6 words; Data: 2x20 words
; Number of clock cycles: 54 (27 instruction cycles)
; Clock Frequency: 20.5 MHz/27.0 MHz
; Instruction cycle time: 97.6 ns/74.1 ns
*****
; This FIR filter reads the input sample
; from the memory location Y:input
; and writes the filtered output sample
; to the memory location Y:output
;
; The samples are stored in the X memory
; The coefficients are stored in the Y memory
*****

```

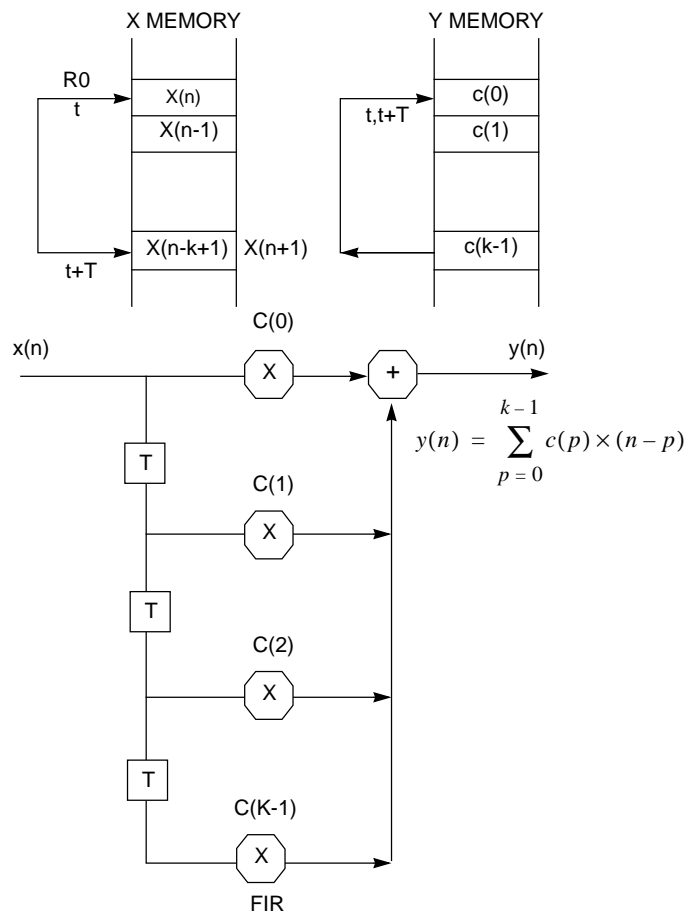


Figure B-1 20-Tap FIR Filter Example (Sheet 1 of 2)


```

;This program originally available on the Motorola DSP bulletin board.
;It is provided under a DISCLAIMER OF WARRANTY available from
;Motorola DSP Operation, 6501 William Cannon Drive, Austin, TX, 78735
;
;Radix-2, In-Place, Decimation-In-Time FFT (smallest code size).
;
;Last Update 30 Sep 86      Version 1.1
;
fftr2a      macro      points,data,coef
fftr2a      ident      1,1
;
;Radix-2 Decimation-In-Time In-Place FFT Routine
;
;   Complex input and output data
;     Real data in X memory
;     Imaginary data in Y memory
;   Normally ordered input data
;   Bit reversed output data
;     Coefficient lookup table
;     -Cosine values in X memory
;     -Sine values in Y memory
;
;Macro Call — ffr2a      points,data,coef
;
;           points          number of points (2-32768, power of 2)
;           data            start of data buffer
;           coef            start of sine/cosine table
;
;Alters Data ALU Registers
;           x1             x0             y1             y0
;           a2             a1             a0             a
;           b2             b1             b0             b
;
;Alters Address Registers
;           r0             n0             m0
;           r1             n1             m1
;
;           r4             n4             m4
;           r5             n5             m5
;           r6             n6             m6
;
;Alters Program Control Registers
;           pc             sr
;
;Uses 6 locations on System Stack
;

```

Figure B-2 Radix 2, In-Place, Decimation-In-Time FFT (Sheet 1 of 2)

;Latest Revision — September 30, 1986

```

;
    move    #points/2,n0      ;initialize butterflies per group
    move    #1,n2            ;initialize groups per pass
    move    #points/4,n6     ;initialize C pointer offset
    move    #-1,m0           ;initialize A and B address modifiers
    move    m0,m1            ;for linear addressing
    move    m0,m4
    move    m0,m5
    move    #0,m6            ;initialize C address modifier for
                                ;reverse carry (bit-reversed) addressing
;
;Perform all FFT passes with triple nested DO loop
;
    do      #@cvi (@log(points)/@log(2)+0.5),_end_pass
    move    #data,r0         ;initialize A input pointer
    move    r0,r4            ;initialize A output pointer
    lua    (r0)+n0,r1        ;initialize B input pointer
    move    #coef,r6         ;initialize C input pointer
    lua    (r1)-,r5          ;initialize B output pointer
    move    n0,n1            ;initialize pointer offsets
    move    n0,n4
    move    n0,n5

    do      n2,_end_grp
    move    x:(r1),X1        y:(r6),y0      ;lookup -sine and
                                ; -cosine values
    move    x:(r5),a         y:(r0),b       ;preload data
    move    x:(r6)+n6,x0     ;update C pointer

    do      n0,_end_bfy
    mac     x1,y0,b          y:(r1)+,y1      ;Radx 2 DIT
                                ;butterfly kernel
    macr   -x0,y1,b        a,x:(r5)+      y:(r0),a
    subl   b,a             x:(r0),b       b,y:(r4)
    mac    -x1,x0,b        x:(r0)+,a      a,y:(r5)
    macr   -y1,y0,b        x:(r1),x1
    subl   b,a             b,x:(r4)+      y:(r0),b
_end_bfy
    move    a,x:(r5)+n5      y:(r1)+n1,y1    ;update A and B pointers
    move    x:(r0)+n0,x1     y:(r4)+n4,y1
_end_grp
    move    n0,b1            ;divide butterflies per group by two
    lsr    b    n2,a1        ;multiply groups per pass by two
    lsl    a    b1,n0
    move    a1,n2
_end_pass
endm

```

Figure B-2 Radix 2, In-Place, Decimation-In-Time FFT (Sheet 2 of 2)


```

page 132,60,1,1
;newlms2n.asm
; New Implementation of the delayed LMS on the DSP56000 Revision C
;Memory map:
; Initial X
; x(n) x(n-1) x(n-2) x(n-3) x(n-4) H hx h0 h1 h2 h3
; ] ]
; r0 r5 r4
;hx is an unused value to make the calculations faster.
;
opt cc
ntaps equ 4
input equ $FFC0
output equ $FFC1
org x:$0
state ds 5
org y:$0
coef ds 5
;
org p:$40
move #state,r0 ;start of X
move #2,n0
move #ntaps,m0 ;mod 5
move #coef +1,r4 ;coefficients
move #ntaps,m4 ;mod 5
move #coef,r5 ;coefficients
move m4,m5 ;mod 5
_smploop
movep a,x:(r0) y:input,a ;get input sample Prog word 1
move a,x:(r0) ;save input sample 1
;error signal is in y1
;FIR sum in a=a+h(k) old*x(n-k)
;h(k)new in b=h(k)old + error*x(n-k-1)
cir a x:(r0)+,x0 ;x0=x(n) 1 1
move x:(r0)+,x1 y:(r4)+,y0 ;x1=x(n-1),y0=h(0) 1 1
do #taps/2,_lms ; 2 3
mac x0,y0,a y0,b b,y:(r5)+ ;a=h(0)*x(n),b=h(0) 1 1
macr x1,y1,b x:(r0)+,x0 y:(r4)+,y0 ;b=h(0)+e*x(n-1)=h(0)new 1 1
;x0=x(n-2) y0=h(1)
mac x1,y0,a y0,b b,y:(r5)+ ;a=a+h(1)*x(n-1) b=h(1) 1 1
macr x0,y1,b x:(r0)+,x1 y:(r4)+,y0 ;b=h(1)+e*x(n-2) 1 1
;x1=x(n-3) y0=H(2)
;_lms
move b,y:(r5)+ ;save last new c ( ) 1 1
move (r0) -n0 ;pointer update 1 1
;(Get d(n), subtract fir output (reg a), multiply by "u", put
;the result in y1. This section is application dependent.)
movep a,y:output ;output fir if desired
jmp _smploop
end
;
Totals: 11 2N+8

```

Figure B-4 LMS FIR Adaptive Filter

