6.4 OPTIMIZED RADIX-2 DIT FFT

Because the FFT is often just the first step in the processing of a signal, the execution speed is important. The faster the FFT executes, the more time the processor can devote to the remainder of the signal processing task. Improving the execution speed of the FFT in a given algorithm may allow a faster sampling rate for the system, a higher-order filter or a more detailed algorithm.

The radix-2 DIT FFT implemented in a looped form—the *stage* loop, the *group* loop and the *butterfly* loop—is the most compact form of the FFT, in terms of program memory storage requirements. The performance of a looped program which accommodates all stages of the FFT can be improved by mathematical optimization. The fully looped program does not exploit the unique mathematical characteristics of the first and last stage of the FFT. By breaking the first and last FFT stage out of the nested loop structure, a few tricks can be employed to improve the execution speed. The tradeoff for increased speed is a modest increase in program memory storage requirements.

This section focuses on the optimizing of the radix-2 DIT FFT. Similar modifications can be applied to the radix-2 DIF algorithm.

6.4.1 First Stage Modifications

The DIT FFT butterfly equations are as follows.

(7) $x'_0 = x_0 + [\mathbf{C}x_1 - (-\mathbf{S})y_1]$

(8) $y'_0 = y_0 + [\mathbf{C}y_1 + (-\mathbf{S})x_1]$

(9) $x'_1 = x_0 - [\mathbf{C}x_1 - (-\mathbf{S})y_1]$

(10) $y'_1 = y_0 - [\mathbf{C}y_1 + (-\mathbf{S})x_1]$

In the first stage, there are N/2 groups, each containing a single butterfly. Each butterfly uses a twiddle factor W^0, where

$$W^0 = e^{j0} = \cos(0) + j\sin(0) = 1 + j0$$

All of the multiplications in the first stage are by a value of either 0 or 1 and therefore can be removed. The first-stage butterflies do not need multiplications. The butterfly equations reduce to the following.

$$x'_0 = x_0 + x_1$$

$$y'_0 = y_0 + y_1$$

$$x'_1 = x_0 - x_1$$

$$y'_1 = y_0 - y_1$$

Because there is only one butterfly per group in the first stage, the butterfly loop (which would execute only once per group) and the group loop can be combined. The combination of the group and butterfly loops is shown in Listing 6.16. The elimination of the multiplications and the combination of the group and butterfly loops saves eleven clock cycles per group. In a 1024-point FFT, for example, the first stage contains 512 butterfly loops. These simple modifications save 5632 (512x11) clock cycles.

```
          DO group_lp UNTIL CE;
             AR=AX0+AY0, AX1=DM(I2,M0);   {AR=X1+Y1 , AX1=Y0}
             DM(I0,M2)=AR, AR=AX0-AY0;    {store X0', AR=X0-X1}
             DM(I1,M2)=AR, AR=AX1+AY1;    {store X1', AR=Y0+Y1}
             DM(I2,M2)=AR, AR=AX1-AY1;    {store Y0', AR=Y0-Y1}
             DM(I3,M2)=AR;                {store Y1'}
             AX0=DM(I0,M0);               {AX0 = next X0}
             AY0=DM(I1,M0);               {AY0 = next X1}
group_lp:    AY1=DM(I3,M0);               {AY1 = next Y1}
```

Listing 6.16 First Stage Butterfly

Similar modifications can be performed on the block floating-point DIT FFT algorithm. First-stage butterfly multiplications can be reduced to additions, and the group and butterfly loops can be combined. Listing 6.17 contains a block floating-point first-stage butterfly.

```
          DO group_lp UNTIL CE;
             AR=AX0+AY0, AX1=DM(I2,M0);
             SB=EXPADJ AR, DM(I0,M2)=AR;
             AR=AX0-AY0;
             SB=EXPADJ AR;
             DM(I1,M2)=AR, AR=AX1+AY1;
             SB=EXPADJ AR, DM(I2,M2)=AR;
             AR=AX1-AY1, AX0=DM(I0,M0);
             SB=EXPADJ AR, DM(I3,M2)=AR;
             AY0=DM(I1,M0);
group_lp:    AY1=DM(I3,M0);
          CALL bfp_adj;
```

Listing 6.17 Block Floating-Point First Stage Butterfly

6.4.2 Last Stage Modifications

The last stage of the DIT FFT can also be modified to increase performance. This stage consists of a single group of N/2 butterflies. Because there is only one group in this stage, the group loop is not needed. The calculations in previous stages of the number of butterflies per group and the number of groups can also be removed. Furthermore, in the last stage, no setup for the next stage is needed. The code for the last stage with these modifications is shown in Listing 6.18.

```
         I0=^inplacereal;           {I0 -> x0}
         I1=^inplacereal+nover2;    {I1 -> x1}
         I2=^inplaceimag;           {I2 -> y0}
         I3=^inplaceimag+nover2;    {I3 -> y1}
         CNTR=nover2;               {# of butterflies}
         M2=DM(node_space);         {node space modifier}
         M4=1;
         I6=I3;
         I4=^twid_real;     {real twiddle pointer}
         I5=^twid_imag;     {imag. twiddle pointer}
         MY0=PM(I4,M4),MX0=DM(I1,M0);
         MY1=PM(I5,M4);
         DO bfly_lp UNTIL CE;
            MR=MX0*MY0(SS),MX1=DM(I6,M5);
            MR=MR-MX1*MY1(RND),AY0=DM(I0,M0);
            AR=MR1+AY0,AY1=DM(I2,M0);
            DM(I0,M1)=AR,AR=AY0-MR1;
```

(listing continues on next page)

```
            MR=MX0*MY1(SS),DM(I1,M1)=AR;
            MR=MR+MX1*MY0(RND),MX0=DM(I1,M0),MY1=PM(I5,M4);
            AR=AY1-MR1,MY0=PM(I4,M4);
            DM(I3,M1)=AR,AR=MR1+AY1;
 bfly_lp:   DM(I2,M1)=AR;
```

Listing 6.18 Last Stage DIT FFT

6.4.3 Optimized Radix-2 DIT FFT Program Listings

This section contains listings for two optimized FFT routines, one with no scaling (input data scaled) and one that performs block floating-point scaling. Listing 6.19 contains the main module, which performs initialization operations and places the input data in bit-reversed (scrambled) order. Listing 6.20 contains the *fft* module, which performs the FFT with input data scaling. Listing 6.21 contains a second *fft* module that performs the FFT with block floating-point scaling. Either FFT routine can be called from the main module.

The *scramble* and *bfp_adj* routines are the same as for the unoptimized radix-2 DIT FFT program.

```
.MODULE/ABS=4           dit_fft_main;
.CONST                  N=1024, N_div_2=512; {For 1024 points}
.VAR/PM/RAM/CIRC        twid_real [N_div_2];
.VAR/PM/RAM/CIRC        twid_imag [N_div_2], padding [4];
.VAR/DM/RAM/ABS=0       inplacereal [N], inplaceimag [N];
.VAR/DM/RAM/ABS=H#1000  inputreal [N], inputimag [N];
.VAR/DM/RAM             groups, bflys_per_group,
                        node_space, blk_exponent;

.INIT   twid_real: <twid_real.dat>;
.INIT   twid_imag: <twid_imag.dat>;
.INIT   inputreal: <inputreal.dat>;
.INIT   inputimag: <inputimag.dat>;
.INIT   inplaceimag: <inputimag.dat>;
.INIT   groups: N_div_2;
.INIT   bflys_per_group: 2;
.INIT   node_space: 2;
.INIT   blk_exponent: 0;
.INIT   padding: 0,0,0,0;            {Zeros after inplaceimag}

.GLOBAL twid_real, twid_imag;
.GLOBAL inplacereal, inplaceimag;
.GLOBAL inputreal, inputimag;
.GLOBAL groups, bflys_per_group, node_space, blk_exponent;
```

```
.EXTERNAL       scramble, fft_strt;

                CALL scramble;       {subroutine calls}
                CALL fft_strt;
                TRAP;                {halt program}
.ENDMOD;
```

Listing 6.19 Main Module for Optimized Radix-2 DIT FFT

```
{1024-point DIT radix-2 FFT}
{No scaling: input data must be scaled}

.MODULE/ROM   fft;

.CONST        log2N=10, N=1024, nover2 =512, nover4 =256;

.EXTERNAL     twid_real,twid_imag,inplacereal,inplaceimag;
.EXTERNAL     inputreal,inputimag;
.EXTERNAL     groups,bflys_per_group,node_space;
.ENTRY        fft_strt;

fft_strt:     CNTR=log2N-2;         {CNTR=# stages in fft-2}
              M0=0;                 {in place modifier}
              M1=1;                 {advance 1 modifier}
              M3=-1;
              M5=1;
              L1=0;
              L2=0;
              L3=0;
              L4=%twid_real;        {length register for}
              L5=%twid_imag;        {twiddle factor tables}
              L6=0;

{First stage of 1024-point FFT}

              I0=^inplacereal;      {I0 -> x0}
              I1=^inplacereal+1;    {I1 -> x1}
              I2=^inplaceimag;      {I2 -> y0}
              I3=^inplaceimag+1;    {I3 -> y1}
              M2=2;                 {modifier for 1st stage}
                                    {node space}
              CNTR=nover2;          {number of groups}
```

(listing continues on next page)

```
              AX0=DM(I0,M0);        {AX0=x0}
              AY0=DM(I1,M0);        {AY0=x1}
              AY1=DM(I3,M0);        {AY1=y1}

              DO group_lp UNTIL CE;
                 AR=AX0+AY0, AX1=DM(I2,M0); {AR=x1+y1 , AX1=y0}
                 DM(I0,M2)=AR, AR=AX0-AY0;  {store x0', AR=x0-x1}
                 DM(I1,M2)=AR, AR=AX1+AY1;  {store x1', AR=y0+y1}
                 DM(I2,M2)=AR, AR=AX1-AY1;  {store y0', AR=y0-y1}
                 DM(I3,M2)=AR;              {store y1'}
                 AX0=DM(I0,M0);             {AX0=next x0}
                 AY0=DM(I1,M0);             {AY0=next x1}
group_lp:        AY1=DM(I3,M0);             {AY1=next y1}

{——— END STAGE 1 ———}

{——— STAGES 2 THRU 9 ———}

              DO stage_loop UNTIL CE;       {begin looping}
                                            {stage 2 thru stage
9}
                 I0=^inplacereal;           {I0 -> x0}
                 I2=^inplaceimag;           {I2 -> y0}
                 SI=DM(groups);             {SI=# groups in}
                                            {previous stage}
                 SR=ASHIFT SI BY -1(LO);    {SR0=groups/2}
                 CNTR=SR0;                  {CNTR=# groups in}
                                            {current stage}
                 DM(groups)=SR0;            {update “groups”}
                 M4=SR0;                    {Initialize twid}
                                            {pointer modifier}
                 M2=DM(node_space);         {M2=dual node
spacing}
                 I1=I0;                     {points to 1st
x_real}
                 MODIFY(I1,M2);             {points to 1st
y_real}
                 I3=I2;                     {points to 1st
x_imag}
                 MODIFY(I3,M2);             {points to 1st
y_imag}
                 I6=I3;                     {I6->y1}

                 DO group_loop UNTIL CE;
```

```
                    I4=^twid_real;          {I4->real twid
factor}
                    I5=^twid_imag;          {I5->imag twid
factor}
                    CNTR=DM(bflys_per_group);    {# bflies/group}
                    MY0=PM(I4,M4),MX0=DM(I1,M0); {MY0=C, MX0=x1}
                    MY1=PM(I5,M4);               {MY1=(-S)}

                    DO bfly_loop UNTIL CE;
                       MR=MX0*MY0(SS),MX1=DM(I6,M5);
                                   {MR=x(C), MX1=y1}
                       MR=MR-MX1*MY1(SS),AY0=DM(I0,M0);
                                   {MR=x1(C)-y1(-S),AY0=x0}
                       AR=MR1+AY0,AY1=DM(I2,M0);
                                   {AR=x0+[x1(C)-y1(-S)],AY1=y0}
                       DM(I0,M1)=AR,AR=AY0-MR1;
                                   {store x0', calc. x1'}
                       MR=MX0*MY1(SS),DM(I1,M1)=AR;
                                   {MR=x1(-S), store x1'}
                       MR=MR+MX1*MY0(RND),MX0=DM(I1,M0),
                                MY1=PM(I5,M4);
                                {MR=x1(-S)+y1(C),MX0=x1, MY1=(-
S)}
                       AR=AY1-MR1,MY0=PM(I4,M4);
                                             {calc y1', MY0=(C)}
                       DM(I3,M1)=AR,AR=MR1+AY1;
                                             {store y1',calc y0'}
bfly_loop:             DM(I2,M1)=AR;         {store y0'}


                    MODIFY(I0,M2);       {I0 -> x0 in next group}
                    MODIFY(I1,M2);       {I1 -> x1 in next group}
                    MODIFY(I2,M2);       {I2 -> y0 in next group}
group_loop:         MODIFY(I3,M2);       {I3 -> y1 in next group}

                 SI=DM(bflys_per_group);
                 SR=ASHIFT SI BY 1(LO); {SR=bflys_per_group * 2}
                 DM(node_space)=SR0;
stage_loop:      DM(bflys_per_group)=SR0; {update
bflys_per_group}

{—— LAST STAGE ——————}

              I0=^inplacereal;            {I0 -> x0}
```

(listing continues on next page)

```
              I1=^inplacereal+nover2;     {I1 -> x1}
              I2=^inplaceimag;            {I2 -> y0}
              I3=^inplaceimag+nover2;     {I3 -> y1}
              CNTR=nover2;                {# of butterflies}
              M2=DM(node_space);          {node space modifier}
              M4=1;
              I6=I3;
              I4=^twid_real;              {real twiddle pointer}
              I5=^twid_imag;              {imag. twiddle pointer}

{butterfly loop as above}
              MY0=PM(I4,M4),MX0=DM(I1,M0);
              MY1=PM(I5,M4);
              DO bfly_lp UNTIL CE;
                 MR=MX0*MY0(SS),MX1=DM(I6,M5);
                 MR=MR-MX1*MY1(RND),AY0=DM(I0,M0);
                 AR=MR1+AY0,AY1=DM(I2,M0);
                 DM(I0,M1)=AR,AR=AY0-MR1;
                 MR=MX0*MY1(SS),DM(I1,M1)=AR;
                 MR=MR+MX1*MY0(RND),MX0=DM(I1,M0),MY1=PM(I5,M4);
                 AR=AY1-MR1,MY0=PM(I4,M4);
                 DM(I3,M1)=AR,AR=MR1+AY1;
bfly_lp:         DM(I2,M1)=AR;

              RTS;                        {return to main}

.ENDMOD;
```

Listing 6.20 Optimized Radix-2 DIT FFT with Input Scaling

```
{1024 point DIT radix 2 FFT}
{Block Floating Point Scaling}

.MODULE      fft;

{     Calling Parameters
             inplacereal=real input data in scrambled order
             inplaceimag=all zeroes (real input assumed)
             twid_real=twiddle factor cosine values
             twid_imag=twiddle factor sine values
             groups=N/2
             bflys_per_group=1
             node_space=1
```

```
      Return Values
             inplacereal=real FFT results, sequential order
             inplaceimag=imag. FFT results, sequential order

      Altered Registers
             I0,I1,I2,I3,I4,I5,L0,L1,L2,L3,L4,L5
             M0,M1,M2,M3,M4,M5
             AX0,AX1,AY0,AY1,AR,AF
             MX0,MX1,MY0,MY1,MR,SB,SE,SR,SI

      Altered Memory
             inplacereal, inplaceimag, groups, node_space,
             bflys_per_group, blk_exponent
}

.CONST       log2N=10, N=1024, nover2=512, nover4=256;

.EXTERNAL    twid_real, twid_imag;
.EXTERNAL    inplacereal, inplaceimag;
.EXTERNAL    groups, bflys_per_group, node_space;
.EXTERNAL    bfp_adj;
.ENTRY       fft_strt;

fft_strt:    CNTR=log2N - 2;     {Initialize stage counter}
             M0=0;
             M1=1;
             L1=0;
             L2=0;
             L3=0;
             L4=%twid_real;
             L5=%twid_imag;
             L6=0;
             SB=-2;

{—————— STAGE 1 ——————}

             I0=^inplacereal;
             I1=^inplacereal + 1;
             I2=^inplaceimag;
             I3=^inplaceimag + 1;
             M2=2;
```

(listing continues on next page)

```
             CNTR=nover2;
             AX0=DM(I0,M0);
             AY0=DM(I1,M0);
             AY1=DM(I3,M0);

             DO group_lp UNTIL CE;
                AR=AX0+AY0, AX1=DM(I2,M0);
                SB=EXPADJ AR, DM(I0,M2)=AR;
                AR=AX0-AY0;
                SB=EXPADJ AR;
                DM(I1,M2)=AR, AR=AX1+AY1;
                SB=EXPADJ AR, DM(I2,M2)=AR;
                AR=AX1-AY1, AX0=DM(I0,M0);
                SB=EXPADJ AR, DM(I3,M2)=AR;
                AY0=DM(I1,M0);
group_lp:       AY1=DM(I3,M0);

             CALL bfp_adj;

{——————————————————————————————}

             DO stage_loop UNTIL CE;{Compute all stages in FFT}
                I0=^inplacereal; {I0 ->x0 in 1st grp of stage}
                I2=^inplaceimag; {I2 ->y0 in 1st grp of stage}
                SI=DM(groups);
                SR=ASHIFT SI BY -1(LO); {groups / 2}
                DM(groups)=SR0;         {groups=groups / 2}
                CNTR=SR0;        {CNTR=group counter}
                M4=SR0;          {M4=twiddle factor modifier}
                M2=DM(node_space);  {M2=node space modifier}
                I1=I0;
                MODIFY(I1,M2);   {I1 ->y0 of 1st grp in stage}
                I3=I2;
                MODIFY(I3,M2);   {I3 ->y1 of 1st grp in stage}

                DO group_loop UNTIL CE;
                   I4=^twid_real;    {I4 -> C of W0}
                   I5=^twid_imag;    {I5 -> (-S) of W0}
                   CNTR=DM(bflys_per_group);
                                     {CNTR=butterfly counter}
                   MY0=PM(I4,M4),MX0=DM(I1,M0); {MY0=C,MX0=x1 }
                   MY1=PM(I5,M4),MX1=DM(I3,M0); {MY1=-S,MX1=y1}
                   DO bfly_loop UNTIL CE;
                      MR=MX0*MY1(SS),AX0=DM(I0,M0);
```

```
                              {MR=x1(-S),AX0=x0}
                      MR=MR+MX1*MY0(RND),AX1=DM(I2,M0);
                              {MR=(y1(C)+x1(-S)),AX1=y0}
                      AY1=MR1,MR=MX0*MY0(SS);
                            {AY1=y1(C)+x1(-S),MR=x1(C)}
                      MR=MR-MX1*MY1(RND);  {MR=x1(C)-y1(-S)}
                      AY0=MR1,AR=AX1-AY1;
                        {AY0=x1(C)-y1(-S),AR=y0-[y1(C)+x1(-S)]}
                      SB=EXPADJ AR,DM(I3,M1)=AR;
                      {Check for bit growth, y1=y0-[y1(C)+x1(-
S)]}
                      AR=AX0-AY0,MX1=DM(I3,M0),MY1=PM(I5,M4);
                 {AR=x0-[x1(C)-y1(-S)], MX1=next y1,MY1=next (-
S)}
                      SB=EXPADJ AR,DM(I1,M1)=AR;
                      {Check for bit growth, x1=x0-[x1(C)-y1(-
S)]}
                      AR=AX0+AY0,MX0=DM(I1,M0),MY0=PM(I4,M4);
                   {AR=x0+[x1(C)-y1(-S)], MX0=next x1,MY0=next C}
                      SB=EXPADJ AR,DM(I0,M1)=AR;
                      {Check for bit growth, x0=x0+[x1(C)-y1(-
S)]}
                      AR=AX1+AY1;  {AR=y0+[y1(C)+x1(-S)]}
bfly_loop:            SB=EXPADJ AR,DM(I2,M1)=AR;
                      {Check for bit growth, y0=y0+[y1(C)+x1(-
S)]}
                   MODIFY(I0,M2);  {I0 ->1st x0 in next group}
                   MODIFY(I1,M2);  {I1 ->1st x1 in next group}
                   MODIFY(I2,M2);  {I2 ->1st y0 in next group}
group_loop:        MODIFY(I3,M2);  {I3 ->1st y1 in next group}

                CALL bfp_adj;      {Compensate for bit growth}
                SI=DM(bflys_per_group);
                SR=ASHIFT SI BY 1(LO);
                DM(node_space)=SR0; {node_space=node_space / 2}
stage_loop:     DM(bflys_per_group)=SR0;  {bflys_per_group= }
                                          {bflys_per_group / 2}

{———— LAST STAGE —————}

             I0=^inplacereal;
             I1=^inplacereal+nover2;
             I2=^inplaceimag;
```

(listing continues on next page)

```
                I3=^inplaceimag+nover2;

                CNTR=nover2;
                M2=DM(node_space);
                M4=1;
                I4=^twid_real;
                I5=^twid_imag;

                MY0=PM(I4,M4),MX0=DM(I1,M0);     {MY0=C,MX0=x1}
                MY1=PM(I5,M4),MX1=DM(I3,M0);     {MY1=-S,MX1=y1}
                DO bfly_lp UNTIL CE;
                   MR=MX0*MY1(SS),AX0=DM(I0,M0);{MR=x1(-S),AX0=x0}
                   MR=MR+MX1*MY0(RND),AX1=DM(I2,M0);
                                    {MR=(y1(C)+x1(-S)),AX1=y0}
                   AY1=MR1,MR=MX0*MY0(SS);
                                    {AY1=y1(C)+x1(-S),MR=x1(C)}
                   MR=MR-MX1*MY1(RND); {MR=x1(C)-y1(-S)}
                   AY0=MR1,AR=AX1-AY1;
                          {AY0=x1(C)-y1(-S), AR=y0-[y1(C)+x1(-S)]}
                   SB=EXPADJ AR,DM(I3,M1)=AR;
                       {Check for bit growth, y1=y0-[y1(C)+x1(-S)]}
                   AR=AX0-AY0,MX1=DM(I3,M0),MY1=PM(I5,M4);
                    {AR=x0-[x1(C)-y1(-S)], MX1=next y1,MY1=next (-
S)}
                   SB=EXPADJ AR,DM(I1,M1)=AR;
                       {Check for bit growth, x1=x0-[x1(C)-y1(-S)]}
                   AR=AX0+AY0,MX0=DM(I1,M0),MY0=PM(I4,M4);
                       {AR=x0+[x1(C)-y1(-S)], MX0=next x1,MY0=next
C}
                   SB=EXPADJ AR,DM(I0,M1)=AR;
                       {Check for bit growth, x0=x0+[x1(C)-y1(-S)]}
                   AR=AX1+AY1;  {AR=y0+[y1(C)+x1(-S)]}
bfly_lp:           SB=EXPADJ AR,DM(I2,M1)=AR;
                       {Check for bit growth}

                CALL bfp_adj;

                RTS;
.ENDMOD;
```

Listing 6.21 Optimized Radix-2 Block Floating-Point DIT FFT