A Real Number Domain Theorems

complex = $\vdash_{def}$ complex of (real # real)

Re_def = $\vdash_{def}$ Re (complex (a,b)) = a

Im_def = $\vdash_{def}$ Im (complex (a,b)) = b

CNJ = $\vdash_{def}$ $\forall$ z. CNJ z = complex (Re z,$\neg$ Im z)

principal_root =
$\vdash_{def}$ $\forall$ n k. principal_root n k =
complex (cos n * k * $\Pi/\neg 2$,sin n * k * $\Pi/\neg$ 2)

complex_4_def = $\vdash_{def}$ complex_4 = complex (1 / 4,0)

rec_sum_def =
$\vdash_{def}$ rec_sum (n,0) f =
complex (0,0) $\wedge$ rec_sum (n,SUC m) f =
rec_sum (n,m) f + f (n + m)

real_FFT_def =
$\vdash_{def}$ $\forall$ x k. real_FFT x k =
rec_sum (0,3) (\n. principal_root n k * EL n x)

real_IFFT_def =
$\vdash_{def}$ $\forall$ L k. real_FFT x k =
complex_4 * rec_sum (0,3) (\n. principal_root n k * EL n L)

B Floating-point Domain Theorems

complex = $\vdash_{def}$ complex of (float # float)

float_complex_sum_def =
$\vdash_{def}$ float_complex_sum (n,0) f =
float_complex (float (0,0,0),float (0,0,0)) $\wedge$
float_complex_sum (n,SUC m) f = float_complex_sum (n,m) f + f (n + m)

float_complex_round_def =
$\vdash_{def}$ $\forall$ z. float_complex_round z =
float_complex (float (round float_format To_nearest (Re z))
,float (round float_format To_nearest (Im z)))

float_complex_val_def =
$\vdash_{def}$ $\forall$ z. float_complex_val z =
complex (Val (float_Re z),Val (float_Im z))

```
float_FFT_def =
⊢def  ∀ x k. float_FFT x k =
    float_complex_sum (0,3) (\n. float_principal_root n k * EL n x)

float_IFFT_def =
⊢def  ∀ L k. float_IFFT x k =
    float_complex_4 * float_complex_sum (0,3)
    (\n. float_principal_root_1 n k * EL n L)
```

C Fixed-point Theorems

```
fxp_complex_sum_def =
⊢def  fxp_complex_sum (n,0) X f =
    fxp_complex (fxp (WORD (REPLICATE (streamlength (X)) F),X),
    fxp (WORD (REPLICATE (streamlength (X)) F),X)) ∧
    fxp_complex_sum (n,SUC m) X f =
    fxp_complex_add X (fxp_complex_sum (n,m) X f) (f (n + m))

fxp_complex_round_def =
⊢def  ∀ X z. fxp_complex_round X z =
    fxp_complex (Fxp_round X Re(z), Fxp_round X Im (z))

fxp_complex_value =
⊢def  ∀ z. fxp_complex_value z =
    complex (value fxp_Re (z), value fxp_Im (z))

fxp_FFT_def =
⊢def  ∀ X x k. fxp_FFT X x k =
    fxp_complex_sum (0,3) X (\n. (fxp_complex_mul X
    (fxp_principal_root X n k)  (EL n x) ))

fxp_IFFT_def =
⊢def  ∀  X L k. fxp_IFFT X L k  =
    fxp_complex_mul X (fxp_complex_4 X) (fxp_complex_sum (0,3) X
    (\n.(fxp_complex_mul X (fxp_principal_root_1 X n k)  (EL n L) )))
```

D Real to Floating-point Error Analysis

```
FFT_REAL_TO_FP_ERROR =
  ⊢thm   ∀ x x1 k. ∃e1 e2 e3.
   FFT_REAL_TO_FP_ERROR x x1 k =
    (((((float_complex_val
   ((float_principal_root 0 k)* (EL 0 x1)) * complex ((1 + e3) , 0))
   + float_complex_val ((float_principal_root 1 k) * (EL 1 x1)))
   * complex ((1 + e2) , 0)) + float_complex_val
   ((float_principal_root 2 k) * (EL 2 x1))) * complex ((1 + e1) , 0))
```

```
   - (rec_sum (0,3) (\n.((principal_root n k)* (EL n x)))))
```

IFFT_REAL_TO_FP_ERROR =
$\vdash_{thm}$ ∀ L L1 k.∃e1 e2 e3.

```
   IFFT_REAL_TO_FP_ERROR L L1 k =
   float_complex_val float_complex_4 * ((((((float_complex_val
   ((float_principal_root_1 0 k) * (EL 0 L1)) * complex ((1 + e3) , 0))
   + float_complex_val ((float_principal_root_1 1 k) * (EL 1 L1)))
   * complex ((1 + e2) , 0)) + float_complex_val
   ((float_principal_root_1 2 k) * (EL 2 L1))) * complex ((1 + e1) , 0)))
   * complex (1 + e,0) - (complex_4 * rec_sum (0,3)
   (\n. ((principal_root_1 n k) * ((EL n L)))))
```

Y_product_REAL_TO_FP_ERROR =
$\vdash_{thm}$ ∀ W1 u W1_1 u_1 Y1.∃e1.

```
   Y_product_REAL_TO_FP_ERROR W1 u W1_1 u_1 Y1 =
   MAP (\k. (float_complex_val (float_CNJ (float_FFT u_1 k)))
   * (float_complex_val (EL k W1_1)) * complex (1 + e1,0)
   - (CNJ (real_FFT u k) * EL k W1)) Y1
```

ERROR_REAL_TO_FP_ERROR =
$\vdash_{thm}$ ∀ d d1 y y1 e.∃e1.

```
   ERROR_REAL_TO_FP_ERROR d d1 y y1 e =
   MAP (\k. (float_complex_val (EL k d1)
   - float_complex_val (EL k y1)) * complex (1 + e1,0)
   - (EL k d - EL k y)) e
```

FREQ_PRODUCT_REAL_TO_FP_ERROR =
$\vdash_{thm}$ ∀ u e E u1 e1 E1 P.∃ e2.

```
   FREQ_PRODUCT_REAL_TO_FP_ERROR  u e E u1 e1 E1 P =
   MAP (\k. (float_complex_val (float_CNJ (float_FFT u1 k)))
   * (float_complex_val (EL k (float_error_FFT e1 E1)))
   * complex (1 + e2,0) - (CNJ (real_FFT u k) * ( EL k (error_FFT e E)))) P
```

PRODUCT_GRADIENT_REAL_TO_FP_ERROR =
$\vdash_{thm}$ ∀ W2 W2_1 W3.∃e1.

```
   PRODUCT_GRADIENT_REAL_TO_FP_ERROR W2 W2_1 W3 =
   MAP (\k. (float_complex_val float_complex_2)
   * (float_complex_val ( EL k W2_1)) * complex (1 + e1,0)
   - (complex_2 * ( EL k W2))) W3
```

COEF_UPDATE_REAL_TO_FP_ERROR =
$\vdash_{thm}$ ∀ W1 W3 W1_1 W3_1 W2.∃e1.

```
   COEF_UPDATE_REAL_TO_FP_ERROR W1 W3 W1_1 W3_1 W2 =
   MAP (\k. ((float_complex_val (EL k W1_1))
   + (float_complex_val (EL k W3_1))) * complex (1 + e1,0)
   - (EL k W1 + EL k W3)) W2
```

E Real to Fixed-point Error Analysis

FFT_REAL_TO_FXP_ERROR =

$\vdash_{thm}$ ∀ x x1 X k.∃e1 e2 e3.

```
FFT_REAL_TO_FXP_ERROR x x1 X k =
(((((fxp_complex_value ((fxp_complex_mul X (fxp_principal_root X 0 k)
(EL 0 x1))) + complex (e3 , e3)) + fxp_complex_value (fxp_complex_mul
X (fxp_principal_root X 1 k) (EL 1 x1))) + complex (e2 , e2))
+ fxp_complex_value (fxp_complex_mul X (fxp_principal_root X 2 k)
(EL 2 x1))) + complex (e1, e1)) -  (rec_sum (0,3) (\(n:num).
((principal_root n k) * (EL n x)))))
```

IFFT_REAL_TO_FXP_ERROR =

$\vdash_{thm}$ ∀ L L1 k X.∃ e e1 e2 e3.

```
IFFT_REAL_TO_FXP_ERROR L L1 k X =
((fxp_complex_value (fxp_complex_4 X)) * (((((fxp_complex_value
((fxp_complex_mul X (fxp_principal_root_1 X 0 k)  (EL 0 L1)))
+ complex (e3 , e3)) + fxp_complex_value (fxp_complex_mul X
(fxp_principal_root_1 X 1 k)  (EL 1 L1))) + complex (e2 , e2))
+ fxp_complex_value (fxp_complex_mul X (fxp_principal_root_1 X 2 k)
(EL 2 L1))) + complex (e1, e1))) + complex (e , e)
- (complex_4 * rec_sum (0,3) (\n. ((principal_root_1 n k) * ((EL n L))))))
```

Y_product_REAL_TO_FXP_ERROR =

$\vdash_{thm}$ ∀ W1_2 u W1_1 u_1 X Y1. ∃ e1.

```
Y_product_REAL_TO_FXP_ERROR W1_2 u W1_1 u_1 X Y1 =
MAP (\k. (fxp_complex_value (fxp_CNJ X (fxp_FFT X u_1 k)))
* (fxp_complex_value (EL k W1_1)) + complex (e1 , e1)
- (CNJ (real_FFT u k) * EL k W1_2)) Y1
```

ERROR_REAL_TO_FXP_ERROR =

$\vdash_{thm}$ ∀ d d1 y y1 X e. ∃e1.

```
ERROR_REAL_TO_FXP_ERROR d d1 y y1 X e  =
MAP (\k. (fxp_complex_value (EL k d1) - fxp_complex_value (EL k y1))
+ complex (e1,e1) - (EL k d - EL k y)) e
```

FREQ_PRODUCT_REAL_TO_FXP_ERROR =

$\vdash_{thm}$ ∀ u e E u1 e1 E1 X P. ∃ e2.

```
FREQ_PRODUCT_REAL_TO_FXP_ERROR u e E u1 e1 E1 X P =
MAP (\k.(fxp_complex_value (fxp_CNJ X (fxp_FFT X u1 k)))
* (fxp_complex_value ( EL k (fxp_error_FFT X e1 E1))) + complex (e2, e2)
 - (CNJ (real_FFT u k) * ( EL k (error_FFT e E)))) P
```

PRODUCT_GRADIENT_REAL_TO_FXP_ERROR =

$\vdash_{thm}$ ∀ W2 W2_1 X W3. ∃ e1.

```
PRODUCT_GRADIENT_REAL_TO_FXP_ERROR W2 W2_1 X W3 =
MAP (\k. (fxp_complex_value (fxp_complex_2 X)) * (fxp_complex_value
( EL k W2_1)) + complex (e1, e1) - (complex_2 * ( EL k W2))) W3
```

```
COEF_UPDATE_REAL_TO_FXP_ERROR =
⊢thm   ∀ W11 W3 W1_1 W3_1  W2. ∃ e1.
   COEF_UPDATE_REAL_TO_FXP_ERROR W11 W3 W1_1 W3_1 X W2 =
   MAP (\k. (fxp_complex_value (EL k W1_1)) + (fxp_complex_value
   (EL k W3_1)) + complex (e1, e1) - (EL k W11 + EL k W3)) W2
```

F Floating-point to Fixed-point Error Analysis

```
FFT_FP_TO_FXP_ERROR =
⊢thm  ∀  x xp xf X k. ∃  e1 e2 e3 e4 e5 e6.
   FFT_FP_TO_FXP_ERROR x xp xf X k =
   (((((fxp_complex_value ((fxp_complex_mul X (fxp_principal_root X 0 k)
   (EL 0 xf))) + complex (e3 , e3)) + fxp_complex_value (fxp_complex_mul X
   (fxp_principal_root X 1 k)  (EL 1 xf))) + complex (e2, e2))
   + fxp_complex_value (fxp_complex_mul X (fxp_principal_root X 2 k)
    (EL 2 xf))) + complex (e1, e1)) - (((((float_complex_val
   ((float_principal_root 0 k) * (EL 0 xp)) * complex ((1 + e6) , 0))
   + float_complex_val ((float_principal_root 1 k) * (EL 1 xp)))
   * complex ((1 + e5) , 0)) + float_complex_val((float_principal_root 2 k)
   * (EL 2 xp))) * complex ((1 + e4) , 0)))

IFFT_FP_TO_FXP_ERROR =
⊢thm   ∀  L Lp Lf X k. ∃ e1 e2 e3 e4 e5 e6 e7.
   IFFT_FP_TO_FXP_ERROR L Lp Lf X k =
   ((fxp_complex_value (fxp_complex_4 X)) * (((((fxp_complex_value
   ((fxp_complex_mul X (fxp_principal_root_1 X 0 k)  (EL 0 Lp)))
   + complex (e3 , e3)) + fxp_complex_value (fxp_complex_mul X
   (fxp_principal_root_1 X 1 k)  (EL 1 Lp)))+ complex (e2 , e2))
   + fxp_complex_value (fxp_complex_mul X (fxp_principal_root_1 X 2 k)
   (EL 2 Lp))) + complex (e1, e1))) + complex (e , e)
   - (float_complex_val float_complex_4 * ((((((float_complex_val
   ((float_principal_root_1 0 k) * (EL 0 Lf)) * complex ((1 + e7) , 0))
   + float_complex_val ((float_principal_root_1 1 k) * (EL 1 Lf)))
   * complex ((1 + e6) , 0)) + float_complex_val ((float_principal_root_1 2 k)
   * (EL 2 Lf))) * complex ((1 + e5) , 0))) * complex (1 + e4,0)))

Y_product_FP_TO_FXP_ERROR =
⊢thm   ∀  Wr u W1_1 W1_2 u_2 u_1 X. ∃ e1 e2.
   Y_product_FP_TO_FXP_ERROR  Wr u W1_1 W1_2 u_2 u_1 X =
   MAP (\n. (fxp_complex_value (fxp_CNJ X (fxp_FFT X u_1 n)))
   * (fxp_complex_value (EL n W1_1)) + complex (e1 , e1)
   - (float_complex_val (float_CNJ (float_FFT u_2 n)))
   * (float_complex_val (EL n W1_2)) * complex (1 + e2,0)) [0;1;2;3]

ERROR_FP_TO_FXP_ERROR =
⊢thm   ∀  d d1 d2 y y1 y2 X. ∃ e1 e2.
   ERROR_FP_TO_FXP_ERROR d d1 d2 y y1 y2 X =
   MAP (\n. (fxp_complex_value (EL n d1) - fxp_complex_value (EL n y1))
```

```
   + complex (e1,e1) - (float_complex_val (EL n d2) -
 float_complex_val (EL n y2))  * complex (1 + e2,0)) [0;1;2;3]

FREQ_PRODUCT_FP_TO_FXP_ERROR =
⊢thm  ∀  u u1 u2 e6 e4 e5 E E1 E2 X. ∃  e1 e2.
   FREQ_PRODUCT_FP_TO_FXP_ERROR u u1 u2 e6 e4 e5 E E1 E2 X =
   MAP (\n. (fxp_complex_value (fxp_CNJ X (fxp_FFT X u1 n)))
   * (fxp_complex_value (EL n (fxp_error_FFT X e4 E1)))
   + complex (e1, e1) - (float_complex_val (float_CNJ (float_FFT u2 n)))
   * (float_complex_val (EL n (float_error_FFT e5 E2)))
   * complex (1 + e2,0)) [0;1;2;3]

PRODUCT_GRADIENT_FP_TO_FXP_ERROR =
⊢thm  ∀  W2 W2_1 W2_2 X. ∃  e1 e2.
   PRODUCT_GRADIENT_FP_TO_FXP_ERROR W2 W2_1 W2_2 X =
   MAP (\n. (fxp_complex_value (fxp_complex_2 X)) * (fxp_complex_value
   ( EL n W2_1)) + complex (e1, e1) - (float_complex_val float_complex_2)
   * (float_complex_val (EL n W2_2))* complex (1 + e2,0)) [0;1;2;3]

COEF_UPDATE_FP_TO_FXP_ERROR =
⊢thm  ∀  W11 W3 W1_1 W3_1 W1_2 W3_2 X. ∃ e1 e2.
   COEF_UPDATE_FP_TO_FXP_ERROR W11 W3 W1_1 W3_1 W1_2 W3_2 X =
   MAP (\n. (fxp_complex_value (EL n W1_1)) + (fxp_complex_value (EL n W3_1))
   + complex (e1, e1) - ((float_complex_val (EL n W1_2))
   + (float_complex_val (EL n W3_2))) *  complex (1+ e2,0)) [0;1;2;3]
```

G Single Theorem for the whole Frequency Domain Equalizer Error Analysis

```
FAST_LMS_FP_TO_FXP_ERROR =
⊢thm  ∀  ! x xp xf Wr u W1_1 W1_2 u_2 u_1 L Lp Lf d d1
   d2 y y1 y2 x_1 xp_1 xf_1 u_0 u1 u2 e_1 e1 e2 E E1 E2 L_1 Lp_1 Lf_1 x_2 xp_2
   xf_2 k W2 W2_1 W2_2 W11 W3 W1_11 W3_1 W1_21 W3_2 d d1 d2 y y1 y2 W11 W3 W1_11
   W3_1 W1_21 W3_2 W2 W2_1 W2_2 k X.
∃  ? e46 e47 e44 e45 e16 e17 e18 e19 e20 e21 e30 e31 e32 e33 e34 e35 e36
   e37  e42 e43 e10 e11 e12 e13 e14 e15 e40 e41 e22 e23 e24 e25 e26 e27 e28 e29
   e38 e39 e7 e8 e9 e4 e5 e6.
  (FAST_LMS_FP_TO_FXP_ERROR x xp xf Wr u W1_1 W1_2 u_2 u_1 L Lp Lf x_1 xp_1 xf_1
   u_0 u1 u2 e_1 e1 e2 E E1 E2 L_1 Lp_1 Lf_1 x_2 xp_2 xf_2 W11 W3 W1_11 W3_1 W1_21
   W3_2 W2 W2_1 W2_2 d d1 d2 y y1 y2 k X) =
  ((MAP (\n. (fxp_complex_value (fxp_complex_mul X (fxp_principal_root X 0 k)
  (EL 0 xf)) + complex (e9,e9) + fxp_complex_value (fxp_complex_mul X
  (fxp_principal_root X 1 k) (EL 1 xf)) + complex (e8,e8) + fxp_complex_value
  (fxp_complex_mul X (fxp_principal_root X 2 k) (EL 2 xf)) + complex (e7,e7) -
  ((float_complex_val (float_principal_root 0 k * EL 0 xp) * complex (1 + e6,0)
  + float_complex_val (float_principal_root 1 k * EL 1 xp)) * complex (1 + e5,0)
  + float_complex_val (float_principal_root 2 k * EL 2 xp)) * complex (1 + e4,0)))
  [(0: num); 1; 2; 3]) +
```

```
(MAP (\n. (fxp_complex_value (fxp_CNJ X (fxp_FFT X u_1 (&n)))) *
(fxp_complex_value (EL n (W1_1: fxp_complex list))) + complex (e38 , e38)
 - (float_complex_val (float_CNJ (float_FFT u_2 (&n)))) * (float_complex_val
(EL n (W1_2: float_complex list))) * complex (1 + e39,0)) [0;1;2;3]) +
 (MAP (\n. (fxp_complex_value (fxp_complex_4 X) *
 (fxp_complex_value (fxp_complex_mul X (fxp_principal_root_1 X 0 k)
 (EL 0 Lp)) + complex (e25,e25) + fxp_complex_value (fxp_complex_mul X
 (fxp_principal_root_1 X 1 k) (EL 1 Lp)) + complex (e24,e24) + fxp_complex_value
 (fxp_complex_mul X (fxp_principal_root_1 X 2 k) (EL 2 Lp)) + complex (e23,e23))
 + complex (e22,e22) - float_complex_val float_complex_4 *
 (((float_complex_val (float_principal_root_1 0 k * EL 0 Lf) * complex (1 + e29,0) +
 float_complex_val (float_principal_root_1 1 k * EL 1 Lf)) * complex (1 + e28,0) +
 float_complex_val (float_principal_root_1 2 k * EL 2 Lf)) *
 complex (1 + e27,0)) * complex (1 + e26,0))) [(0: num); 1; 2; 3]) +
(MAP (\n. fxp_complex_value (EL n d1) - fxp_complex_value (EL n y1) +
complex (e40,e40) - (float_complex_val (EL n d2) -float_complex_val (EL n y2))
* complex (1 + e41,0))[0; 1; 2; 3])  +
(MAP (\n. ((((((fxp_complex_value ((fxp_complex_mul X (fxp_principal_root X 0 k)
(EL 0 (xf_1:fxp_complex list)))) + complex (e12 , e12)) + fxp_complex_value
(fxp_complex_mul X (fxp_principal_root X 1 k)  (EL 1 (xf_1:fxp_complex list))))
+ complex (e11, e11)) + fxp_complex_value (fxp_complex_mul X (fxp_principal_root X 2 k)
(EL 2 (xf_1:fxp_complex list)))) + complex (e10, e10)) - (((((float_complex_val
((float_principal_root (&0) k) * (EL 0 (xp_1:float_complex list))) * complex
(((1:real) + e15) , 0)) + float_complex_val ((float_principal_root (&1) k) *
(EL 1(xp_1:float_complex list)))) * complex (((1:real) + e14) , 0)) +
float_complex_val((float_principal_root (&2) k) * (EL 2 (xp_1:float_complex list))))
* complex (((1:real) + e13) , 0)))) [(0: num); 1; 2; 3]) +
(MAP (\n. fxp_complex_value (fxp_CNJ X (fxp_FFT X u1 (& n))) *
fxp_complex_value (EL n (fxp_error_FFT X e1 E1)) + complex (e42,e42) -
float_complex_val (float_CNJ (float_FFT u2 (& n))) *
float_complex_val (EL n (float_error_FFT e2 E2)) * complex (1 + e43,0)) [0; 1; 2; 3]) +
(MAP (\n. (fxp_complex_value (fxp_complex_4 X) * (fxp_complex_value
(fxp_complex_mul X (fxp_principal_root_1 X 0 k) (EL 0 Lp_1)) + complex (e33,e33) +
fxp_complex_value (fxp_complex_mul X (fxp_principal_root_1 X 1 k) (EL 1 Lp_1)) +
complex (e32,e32) + fxp_complex_value (fxp_complex_mul X
(fxp_principal_root_1 X 2 k) (EL 2 Lp_1))  + complex (e31,e31)) + complex (e30,e30) -
(float_principal_root_1 0 k * EL 0 Lf_1) * complex (1 + e37,0) + float_complex_val
(float_principal_root_1 1 k * EL 1 Lf_1)) * complex (1 + e36,0) +
float_complex_val (float_principal_root_1 2 k * EL 2 Lf_1)) *
complex (1 + e35,0)) * complex (1 + e34,0))) [(0: num); 1; 2; 3]) +
(MAP (\n. ((((((fxp_complex_value ((fxp_complex_mul X (fxp_principal_root X 0 k)
(EL 0 (xf_1:fxp_complex list)))) + complex (e18 , e18)) + fxp_complex_value
(fxp_complex_mul X (fxp_principal_root X 1 k)  (EL 1 (xf_2:fxp_complex list))))
+ complex (e17, e17)) + fxp_complex_value (fxp_complex_mul X (fxp_principal_root X 2 k)
(EL 2 (xf_2:fxp_complex list)))) + complex (e16, e16)) - (((((float_complex_val
((float_principal_root (&0) k) * (EL 0 (xp_2:float_complex list))) *
complex (((1:real) + e21) , 0)) + float_complex_val ((float_principal_root (&1) k)
* (EL 1(xp_2:float_complex list)))) * complex (((1:real) + e20) , 0)) +
float_complex_val((float_principal_root (&2) k) * (EL 2 (xp_2:float_complex list))))
```

```
* complex (((1:real) + e19) , 0))) ) [(0: num); 1; 2; 3]) +
( MAP (\n. fxp_complex_value (fxp_complex_2 X) * fxp_complex_value (EL n W2_1) +
complex (e44,e44) - float_complex_val float_complex_2 * float_complex_val (EL n W2_2)
* complex (1 + e45,0)) [0; 1; 2; 3]) +( MAP (\n. fxp_complex_value (EL n W1_11) +
fxp_complex_value (EL n W3_1) + complex (e46,e46) - (float_complex_val (EL n W1_21) +
float_complex_val (EL n W3_2)) * complex (1 + e47,0)) [0; 1; 2; 3]))
```