Code Tunaing

First tuning: Jamming (Fusion)

Original Code:

for(int i = 0; i < nn\_2; i++)  
{

x[i] = 0;

}

for(int i = 0; i < nn\_2; i++)

{

h[i] = 0;

}

for(int i = 0; i < nn\_2; i++)

{

y[i] = 0;

}

After jamming:

for(int i = 0; i < nn\_2; i++)

{

x[i] = 0;

h[i] = 0;

y[i] = 0;

}

Code tuning 2: Minimizing work inside arrays

Original code:

double MAX\_VAL = 32767;

// Complex multiplication i think (?)

for(int i = 0 ; i < nn\_2; i+=2)

{

y[i] = (x[i]/MAX\_VAL \* h[i]/MAX\_VAL) - (x[i + 1]/MAX\_VAL \* h[i + 1]/MAX\_VAL);

y[i + 1] = (x[i + 1]/MAX\_VAL \* h[i]/MAX\_VAL) + (x[i]/MAX\_VAL \* h[i + 1]/MAX\_VAL);

}

Tuned code:

for(int i = 0 ; i < nn\_2; i+=2)

{

y[i] = (x[i] \* h[i]) - (x[i + 1] \* h[i + 1]);

y[i + 1] = (x[i + 1] \* h[i]) + (x[i] \* h[i + 1]);

}

//done while reading data

data[i++] = (double)sample/MAX\_VAL;

Code tuning 3: Precompute values at compile time

Original Code

h = (double\*) malloc(sizeof(double) \* nn\_2);

x = (double\*) malloc(sizeof(double) \* nn\_2);

y = (double\*) malloc(sizeof(double) \* nn\_2);

outdata = (double\*)malloc(sizeof(double) \* nn);

Tuned code

#define SIZE\_OF\_DOUBLE sizeof(double)

h = (double\*) malloc(SIZE\_OF\_DOUBLE \* nn\_2);

x = (double\*) malloc(SIZE\_OF\_DOUBLE \* nn\_2);

y = (double\*) malloc(SIZE\_OF\_DOUBLE \* nn\_2);

outdata = (double\*)malloc(SIZE\_OF\_DOUBLE \* nn);