Project 3

ECE 561

Embedded System Design

Speed optimization including polynomial approximation

ADITYA SHIWAJI RASAM
200153631
(arasam@ncsu.edu)

ECE 461/561 Project 3 Report Template

Summary

Table showing execution time at each stage, and brief description of optimizations applied, for example:

Stage	Brief Description of Optimizations	Execution time (msec)
Base Code		347
Optimization 1	Level-3 optimization for time (speed)	334.8
Optimization 2	Replaced 180/Pi & 180/Pi with defined constants	308.4
Optimization 3	Replacing trigonometric functions with approximated functions e.g. cos_32	113
Optimization 4	Replacing all arguments from degrees to radians.	99
Optimization 5	Conversion of all Double constants to float	87.38
Optimization 6	Manually calculating PI-asin (temp)	71.44
Optimization 7	Moving fmodf from cos_32 in trig_approx.c to Compute_Current	67.92

Base Code

Compiler settings

Total execution time: 347msec

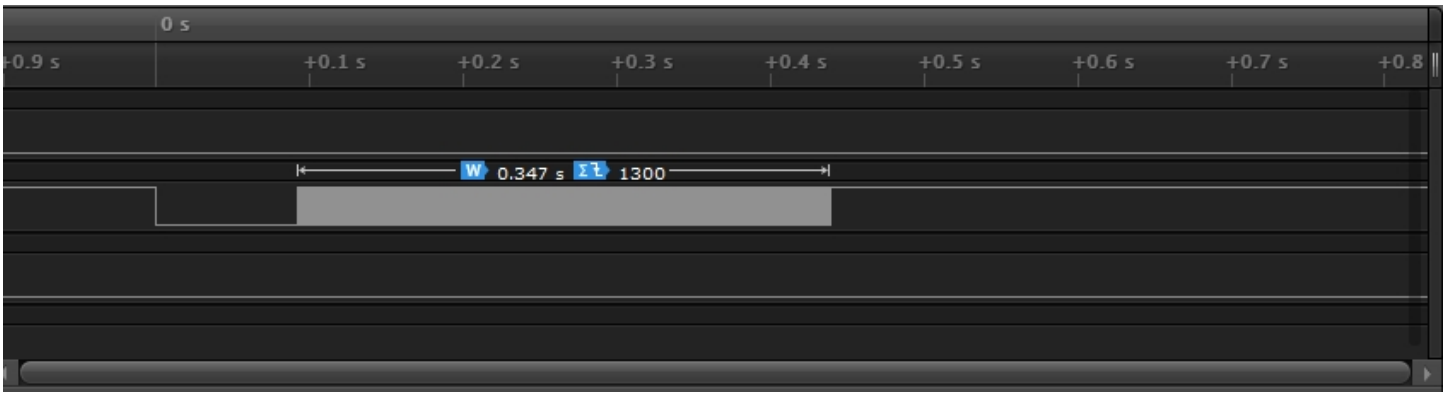
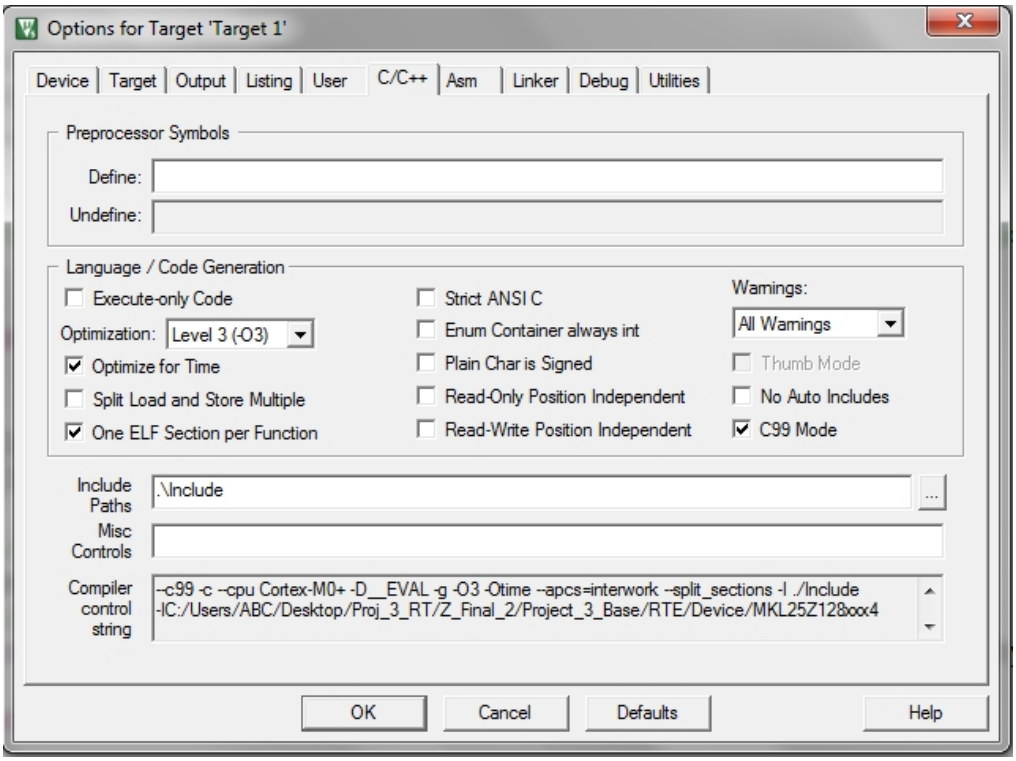


Table showing the five functions which take most of the program's time, for example:

Function Name	Number of Samples	% of Program Execution Time
dmul	1939	43.78
dsub1	533	12%
ddiv	470	10.6%
Compute_Current	229	5.1%
dsqrt	196	4.4%

Optimization 1

Description of optimizations performed: Level-3 optimization for time (speed)



Total execution time: 334.8 msec

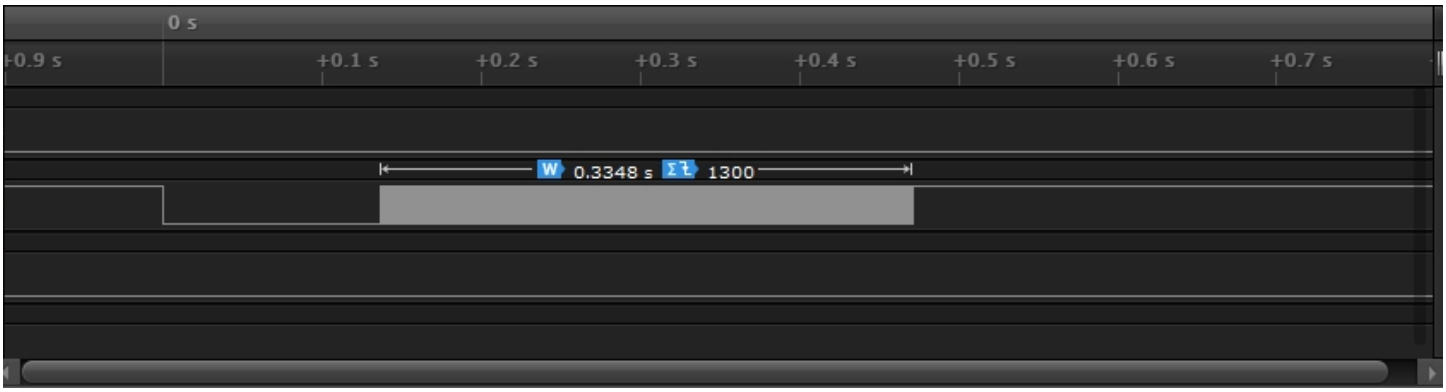


Table showing the five functions which take most of the program's time, for example:

Function Name	Number of Samples	% of Program Execution Time
dmul	2186	52%
dsub1	596	14.2%
ddiv	405	9.6%
dsqrt	300	7.1%
f2d	190	4.5%

Optimization 2

Description of optimizations performed: Replaced 180/Pi & Pi/180 in Compute_Current with defined constants like '***Oneeighty_over_PI'*** & ***'PI_over_180'*** in order to reduce the number of divisions per iteration.

```
#define PI (3.14159265)
#define Two_PI (6.2831853)
#define PI_over_180 (0.017453)
#define Oneeighty_over_PI (57.2958)
```

Total execution time: 308.4 msec

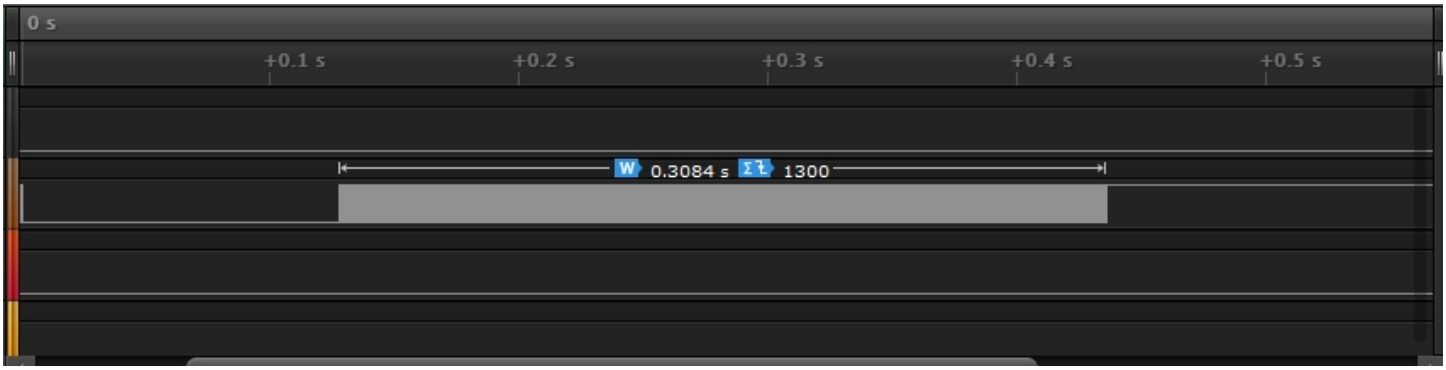


Table showing the five functions which take most of the program's time, for example:

Function Name	Number of Samples	% of Program Execution Time
dmul	2385	59.7%
fdiv	288	7.2%
d2f	284	7.1%
dsub1	121	3.0%
dsqrt	104	2.6%

…

Optimization 3

Description of optimizations performed: Replaced trigonometric functions from Math.h with approximated trigonometric functions in trig_approx.c. Also replacing sqrt with sqrtf () & asin () with asinf (). This reduced a number of double precision operations that was carried out by default in Math.h functions

Total execution time: 113 msec

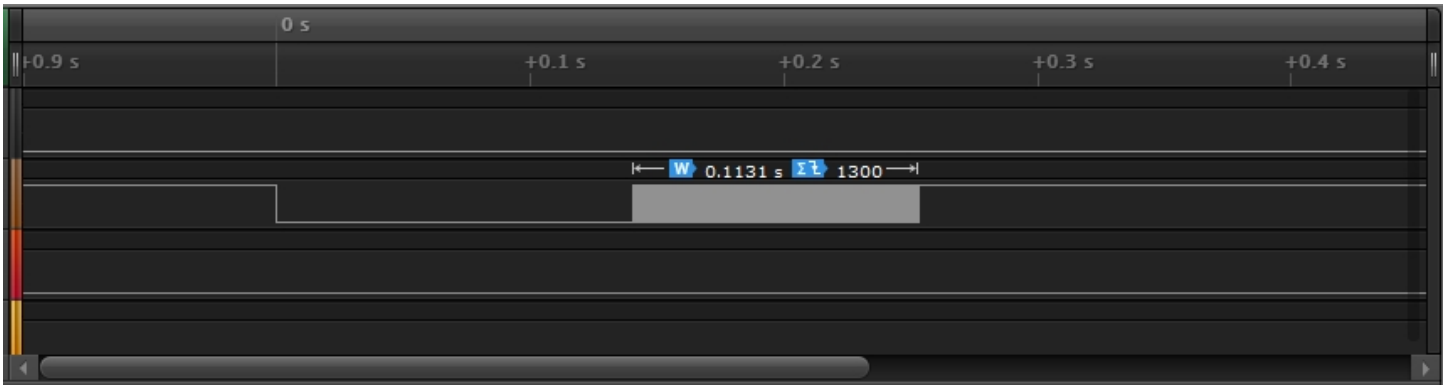


Table showing the five functions which take most of the program's time, for example:

Function Name	Number of Samples	% of Program Execution Time
fmul	495	24.8%
dmul	401	20.1%
fsub	295	14.8%
asinf	199	9.98%
Compute_Current	100	5.0%

Optimization 4

Description of optimizations performed: Replacing function's arguments from degrees to radians.
All the trigonometric functions accepts data in radians. Hence, all the test cases were manually converted from degrees to radians so that the arguments passed to compute_current are in radians that reduces the number of conversion operations within Compute_Current. Although, the test cases were manually updated however multiplying each angular value of the array by a pre-defined constant of PI_over_180 is also possible within the array.
For test cases entered through hyper terminal which has angle in degrees, provision is made to convert the data into radians before passing it into Compute_Current and is defined before the compute_current.
Hence, all the values entered and displayed through HyperTerminal, are in degrees

```
TEST_CASE Tests[] = {                    //--------Updated table to gi
{{  1.000 , 0.000 },  { 1.414 , 0.7854  },  { 1.000 , 1.5708  }},
{{  1.000 , 0.000 },  { 1.414 , 5.4978  },  { 1.000 , 4.71239 }},
{{  1.000 , 0.000 },  { 0.000 , 0.000 },  { 1.000 , 3.14159 }},
{{  1.000 , 0.000 },  { 2.000 , 0.000 },  { 1.000 , 0.000 }},
{{  1.000 , 1.5708  },  { 1.414 , 0.7854  },  { 1.000 , 0.000 }},
{{  1.000 , -1.5708 },  { 1.414 , -0.7854 },  { 1.000 , 0.000 }},
{{  1.000 , 3.14159 },  { 1.414 , 3.927 },  { 1.000 , 4.71239 }},
{{  1.000 , 0.7854  },  { 1.414 , 0.000 },  { 1.000 , 5.4978  }},
{{  1.000 , 2.3562  },  { 1.414 , 3.14159 },  { 1.000 , 3.927 }},
{{  4.000 , 0.000 },  { 5.000 , -0.6435 },  { 3.000 , 4.71239 }},
{{  4.000 , 2.0944  },  { 5.000 , 2.7379  },  { 3.000 , 3.6652  }},
{{  3.000 , 2.3562  },  { 3.200 , 1.3487  },  { 2.998 , 0.34011 }},
{{  3.000 , 3.14159 },  { 2.300 , 1.9897  },  { 2.946 , 0.794195}}
};
```

Total execution time: 99msec

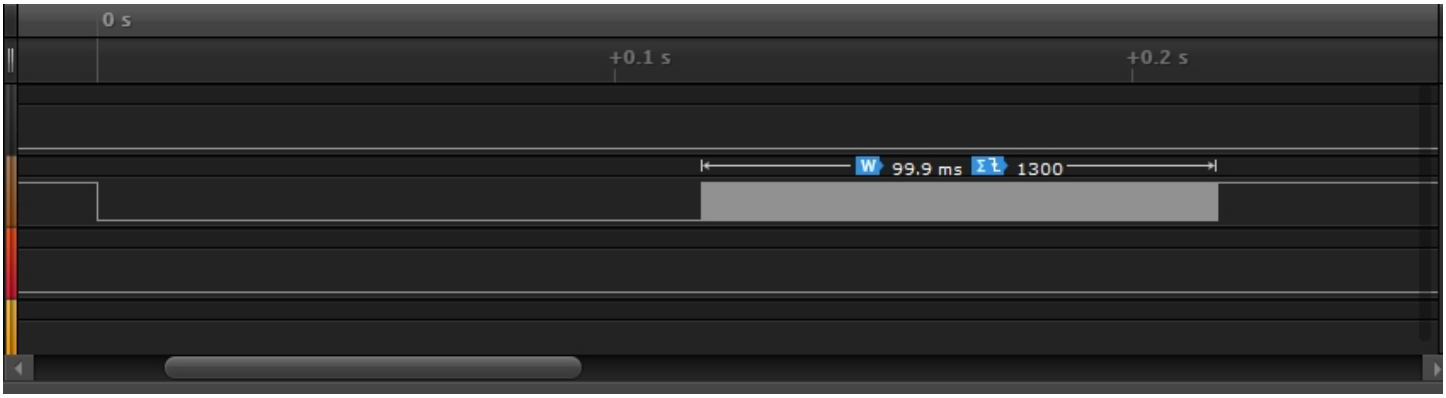


Table showing the five functions which take most of the program's time, for example:

Function Name	Number of Samples	% of Program Execution Time
Compute_Current	383	22.52%
fsub	200	11.75%
dsub1	199	11.7%
dadd1	196	11.52%
fmul	113	6.6%

Optimization 5

Description of optimizations performed: Conversion of all Double constants to float in Drift_Calculation, Main & Trignometric approximation.

```
12  // Math constants we'll use
13  #define DP_PI (3.14159265358979323846433f)  // pi
14  float const twopi=2.0f*DP_PI;      // pi times 2
15  float const two_over_pi= 2.0f/DP_PI;     // 2/pi
16  float const halfpi=DP_PI/2.0f;       // pi divided by 2
17
18  float const threehalfpi=3.0f*DP_PI/2.0f;       // pi times 3/2, used in tan routines
19  float const four_over_pi=4.0f/DP_PI;     // 4/pi, used in tan routines
20  float const qtrpi=DP_PI/4.0f;      // pi/4.0, used in tan routines
21  float const sixthpi=DP_PI/6.0f;      // pi/6.0, used in atan routines
22  float const tansixthpi=0.577350269189626f;     // tan(pi/6), used in atan routines
23  float const twelfthpi=DP_PI/12.0f;       // pi/12.0, used in atan routines
24  float const tantwelfthpi=0.267949192431123f;  // tan(pi/12), used in atan routines
```

Total execution time: 87.38 msec

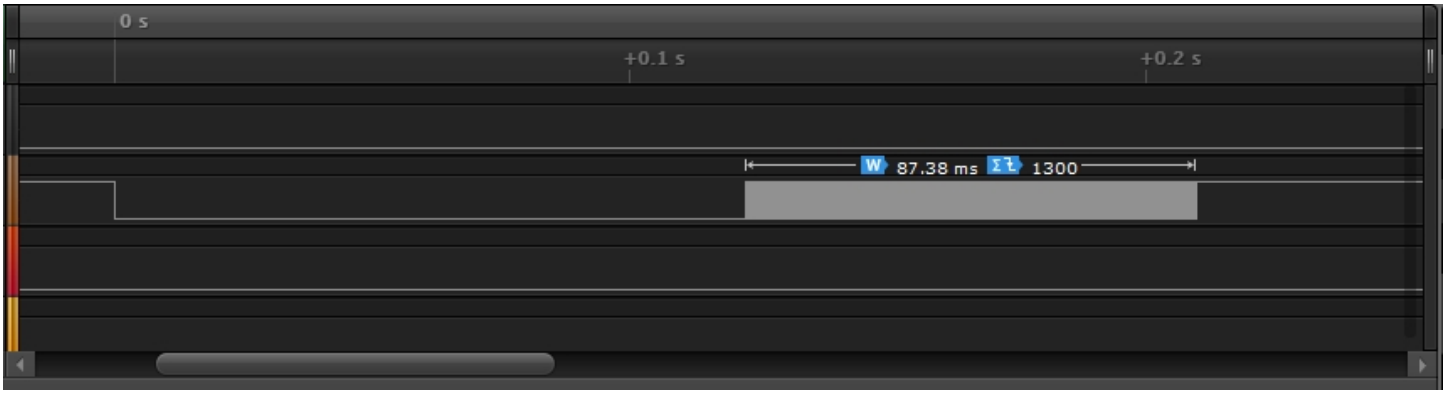


Table showing the five functions which take most of the program's time, for example:

Function Name	Number of Samples	% of Program Execution Time
fsqrt	447	28.95%
fsub	248	16%
fmul	244	15.8%
fadd	153	9.9%
frem	146	9.45%

Optimization 6

Description of optimizations performed: Manually calculating PI-asin (temp) for limited cases (temp>1 & temp<-1) and adding them directly to angle heading

```
if (temp > 1)
{
temp = 1;
local_angle_current= 1.5708f+ angle_heading;
}
else if (temp < -1)
{
  temp = -1;
local_angle_current= 4.712389f+ angle_heading;
}
else
{
local_angle_current = (PI-(asinf(temp))) + angle_heading;
}
```

Total execution time: 71.44 msec

Table showing the five functions which take most of the program's time, for example:

Function Name	Number of Samples	% of Program Execution Time
fmul	257	20.0%
fsub	229	17.89%
fsqrt	212	16.5%
Cos_32	132	10.3%
Compute_Current	74	5.78%

Optimization 7

Description of optimizations performed: Removing fmod from cos_32 in trig_approx.c and inserted it in Compute_Current
This function was called twice through sin_32 & cos_32. Hence instead of calculating the value twice now it is just called once in Compute_Current.

```
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  Modified Aditya  %%%%%%%%%%%%%%%%%%%%%%%%
angle_drift_rad=fmodf(angle_drift_rad, Two_PI);
```

Total execution time: 67.92 msec

Table showing the five functions which take most of the program's time, for example:

Function Name	Number of Samples	% of Program Execution Time
fsqrt	293	23.14%
fmul	289	22.82%
fsub	158	12.48%
fadd	116	9.16%
Compute_Current	100	7.9%