Project 1 Report

Name: Ajay Rao
Unity Id: abrao2

Input Generation: The function randGen(lower, upper) is used to generate the random input values for the accelerometer, gyro meter and magnetometer within the upper and lower bounds specified. Since the seed for the rand function is constant the sequence of rand values produced will be the same and hence will allow repeatability.

Error Checking:

- For error checking run the program with the macro validate set to 1.
- Before running the validation the output files "op_base_log.txt" and "op_log.txt" must be populated.
- For populating "op_base_log.txt" with the values of q0-q3 run the program with the macro "base" is set to 1 in main function.
- For populating "op_log.txt" with the values of q0-q3 run the program with the macro "base" is set to 0 in main function.

The driver function calls the MadgwickUpdate() for 1000 samples. The average time and the minimum execution time of these 1000 samples is printed out on the terminal.

Performance for different versions of the code and compiler settings:

1. **Starter code with compiler optimization level set to 0:**

   Macro settings:
   validate is set to 0
   base is set to 1
   ver is set to 1

```
CC = gcc
CFLAGS = -c -Wall -march=armv7-a -mfpu=neon -mfloat-abi=hard -mcpu=cortex-a8 -mtune=cortex-a8 -O0 -g

PROF =  -pg -g # Add for gprof

%.o: %.c
	$(CC) $(CFLAGS) $(PROF) -c -o $@ $<

madgwick: main.o MadgwickAHRS.o
	$(CC) $(PROF) main.o MadgwickAHRS.o -lrt -lm -static -o $@

MadgwickAHRS_list.s: MadgwickAHRS.c
	$(CC) -Wa,-adhln -g MadgwickAHRS.c -c > MadgwickAHRS_list.s

clean:
	rm -f *.o madgwick *.s
```

   The execution time for 5 consecutive runs are as follows:

```
debian@beaglebone:~/Project1$ ./madgwick
Average     10.473 us
Minimum      9.958 us
debian@beaglebone:~/Project1$ ./madgwick
Average     10.454 us
Minimum      9.959 us
debian@beaglebone:~/Project1$ ./madgwick
Average     10.551 us
Minimum      9.958 us
debian@beaglebone:~/Project1$ ./madgwick
Average     10.478 us
Minimum     10.041 us
debian@beaglebone:~/Project1$ ./madgwick
Average     10.461 us
Minimum      9.916 us
```

Profiling screenshot after doing perf report:

```
Samples: 171  of event 'cycles:ppp', Event count (approx.): 39259616
Overhead  Command   Shared Object       Symbol
  14.38%  madgwick  madgwick            [.] MadgwickAHRSupdate
  13.64%  madgwick  madgwick            [.] __ieee754_fmod
   9.27%  madgwick  [kernel.kallsyms]   [k] _raw_spin_unlock_irqrestore
   5.01%  madgwick  madgwick            [.] vfprintf
   4.49%  madgwick  madgwick            [.] __mcount_internal
   4.28%  madgwick  madgwick            [.] __fmodl
   3.21%  madgwick  madgwick            [.] main
   3.18%  madgwick  [kernel.kallsyms]   [k] sys_clock_gettime
   3.12%  madgwick  madgwick            [.] __printf_fp_l
   2.51%  madgwick  madgwick            [.] invSqrt
   2.50%  madgwick  madgwick            [.] __ieee754_sqrt
   1.87%  madgwick  [kernel.kallsyms]   [k] vector_swi
   1.86%  madgwick  madgwick            [.] randGen
   1.84%  madgwick  madgwick            [.] hack_digit
   1.52%  madgwick  madgwick            [.] write_gmon
   1.33%  madgwick  madgwick            [.] __mpn_mul_1
   1.30%  madgwick  madgwick            [.] __libc_do_syscall
   1.25%  madgwick  madgwick            [.] __random
   1.07%  madgwick  [kernel.kallsyms]   [k] ext4_journal_check_start
   1.01%  madgwick  [kernel.kallsyms]   [k] v7_flush_icache_all
   0.89%  madgwick  [kernel.kallsyms]   [k] vma_link
   0.83%  madgwick  madgwick            [.] _dl_important_hwcaps
   0.80%  madgwick  [kernel.kallsyms]   [k] __sched_text_end
   0.78%  madgwick  madgwick            [.] strstr
   0.66%  madgwick  [kernel.kallsyms]   [k] __do_div64
   0.64%  madgwick  [kernel.kallsyms]   [k] blk_attempt_plug_merge
   0.64%  madgwick  [kernel.kallsyms]   [k] mpage_process_page_bufs
   0.64%  madgwick  madgwick            [.] free
   0.63%  madgwick  [kernel.kallsyms]   [k] iov_iter_advance
   0.63%  madgwick  madgwick            [.] __printf_fp
   0.63%  madgwick  [kernel.kallsyms]   [k] _raw_spin_lock_irq
   0.63%  madgwick  madgwick            [.] __random_r
   0.63%  madgwick  [kernel.kallsyms]   [k] finish_task_switch
   0.63%  madgwick  [kernel.kallsyms]   [k] arm_copy_to_user
   0.62%  madgwick  madgwick            [.] fprintf
   0.62%  madgwick  madgwick            [.] __mpn_rshift
   0.61%  madgwick  madgwick            [.] strlen
   0.61%  madgwick  [kernel.kallsyms]   [k] iov_iter_copy_from_user_atomic
   0.60%  madgwick  [ip_tables]         [k] ipt_do_table
   0.59%  madgwick  madgwick            [.] _init
   0.59%  madgwick  [kernel.kallsyms]   [k] current_fs_time
Tip: Profiling branch (mis)predictions with: perf record -b / perf report
```

2. **Starter code with compiler optimization level set to 1:**

Macro settings:
validate is set to 0
base is set to 1

ver is set to 1

```
CC = gcc
CFLAGS = -c -Wall -march=armv7-a -mfpu=neon -mfloat-abi=hard -mcpu=cortex-a8 -mtune=cortex-a8 -O1 -g

PROF =  -pg -g # Add for gprof

%.o: %.c
	$(CC) $(CFLAGS) $(PROF) -c -o $@ $<

madgwick: main.o MadgwickAHRS.o
	$(CC) $(PROF) main.o MadgwickAHRS.o -lrt -lm -static -o $@

MadgwickAHRS_list.s: MadgwickAHRS.c
	$(CC) -Wa,-adhln -g MadgwickAHRS.c -c > MadgwickAHRS_list.s

clean:
	rm -f *.o madgwick *.s
```

The execution time for 5 consecutive runs are as follows:

```
debian@beaglebone:~/Project1$ ./madgwick
Average       6.401 us
Minimum       6.000 us
debian@beaglebone:~/Project1$ ./madgwick
Average       6.282 us
Minimum       6.000 us
debian@beaglebone:~/Project1$ ./madgwick
Average       6.381 us
Minimum       6.041 us
debian@beaglebone:~/Project1$ ./madgwick
Average       6.348 us
Minimum       6.000 us
debian@beaglebone:~/Project1$ ./madgwick
Average       6.509 us
Minimum       6.000 us
```

Profiling screenshot after doing perf report:

```
Samples: 141  of event 'cycles:ppp', Event count (approx.): 32768446
Overhead  Command   Shared Object       Symbol
  17.39%  madgwick  madgwick            [.] MadgwickAHRSupdate
   9.96%  madgwick  [kernel.kallsyms]   [k] _raw_spin_unlock_irqrestore
   9.81%  madgwick  madgwick            [.] __ieee754_fmod
   9.04%  madgwick  madgwick            [.] __printf_fp_l
   6.19%  madgwick  madgwick            [.] __fmodl
   3.20%  madgwick  [kernel.kallsyms]   [k] sys_clock_gettime
   3.08%  madgwick  madgwick            [.] __mcount_internal
   3.07%  madgwick  madgwick            [.] vfprintf
   2.33%  madgwick  madgwick            [.] hack_digit
   1.56%  madgwick  madgwick            [.] __random
   1.51%  madgwick  [kernel.kallsyms]   [k] vector_swi
   1.50%  madgwick  madgwick            [.] __mpn_mul_1
   1.50%  madgwick  [kernel.kallsyms]   [k] thread_cpu_clock_get
   1.40%  madgwick  [kernel.kallsyms]   [k] __wake_up_bit
   1.26%  madgwick  madgwick            [.] write_gmon
   1.26%  madgwick  [kernel.kallsyms]   [k] handle_mm_fault
   1.19%  madgwick  [kernel.kallsyms]   [k] v7_flush_icache_all
   1.18%  madgwick  [kernel.kallsyms]   [k] get_page_from_freelist
   1.12%  madgwick  madgwick            [.] __memcpy_neon
   1.07%  madgwick  [kernel.kallsyms]   [k] __schedule
   0.94%  madgwick  [kernel.kallsyms]   [k] find_vma
   0.82%  madgwick  [kernel.kallsyms]   [k] _raw_spin_lock_irqsave
   0.79%  madgwick  [kernel.kallsyms]   [k] __list_add
   0.78%  madgwick  [kernel.kallsyms]   [k] __do_softirq
   0.76%  madgwick  [kernel.kallsyms]   [k] posix_cpu_clock_get
   0.76%  madgwick  [kernel.kallsyms]   [k] arm_copy_to_user
   0.76%  madgwick  madgwick            [.] main
   0.76%  madgwick  [kernel.kallsyms]   [k] generic_make_request
   0.76%  madgwick  [kernel.kallsyms]   [k] rw_verify_area
   0.76%  madgwick  [kernel.kallsyms]   [k] ext4_writepages
   0.76%  madgwick  madgwick            [.] __mpn_lshift
   0.76%  madgwick  madgwick            [.] __printf_fp
   0.76%  madgwick  [kernel.kallsyms]   [k] signal_setup_done
   0.75%  madgwick  [kernel.kallsyms]   [k] generic_update_time
   0.75%  madgwick  [kernel.kallsyms]   [k] ext4_da_write_begin
   0.74%  madgwick  madgwick            [.] __clock_gettime
   0.74%  madgwick  madgwick            [.] __random_r
   0.70%  madgwick  [kernel.kallsyms]   [k] switch_task_namespaces
   0.70%  madgwick  [kernel.kallsyms]   [k] tcp_v4_rcv
   0.69%  madgwick  [kernel.kallsyms]   [k] jbd2_journal_stop
   0.69%  madgwick  [kernel.kallsyms]   [k] v7wbi_flush_user_tlb_range
Tip: Treat branches as callchains: perf report --branch-history
```

3. **Starter code with compiler optimization level set to 3:**

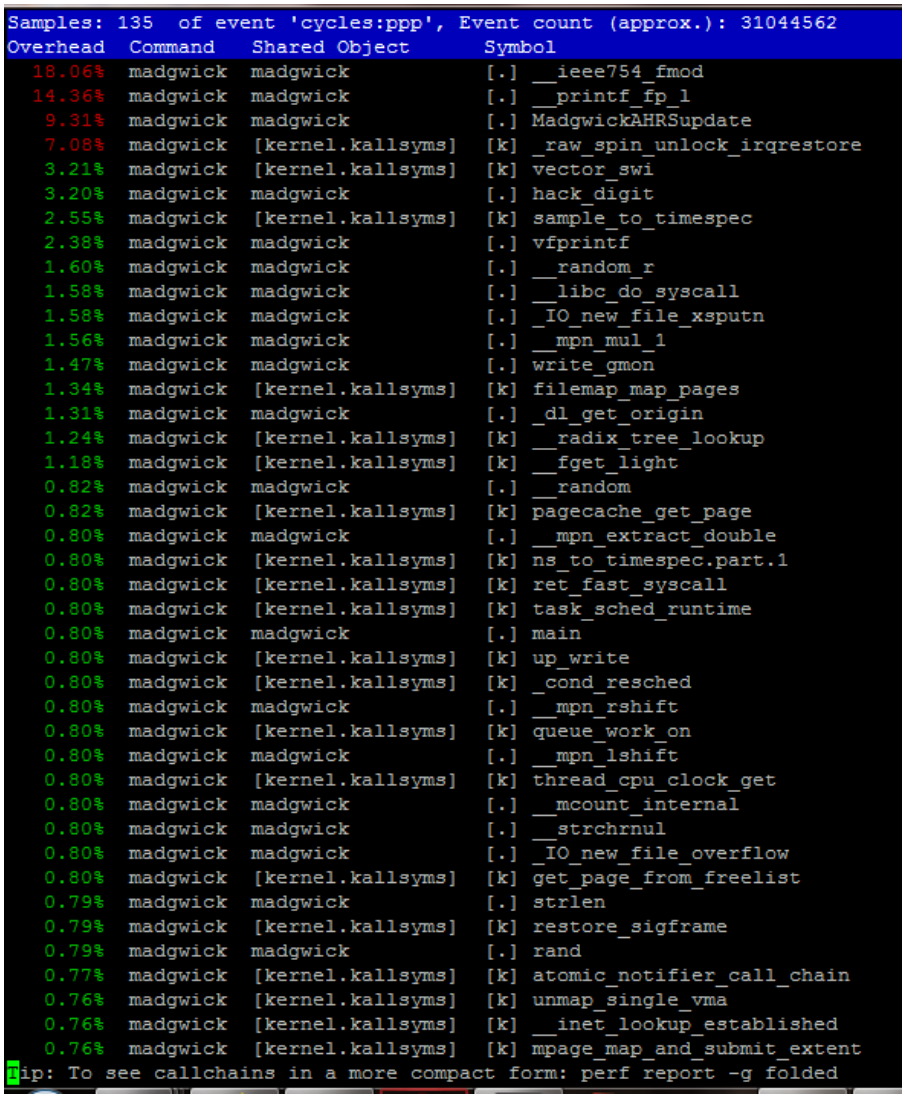
Macro settings:
validate is set to 0
base is set to 1
ver is set to 1

```
CC = gcc
CFLAGS = -c -Wall -march=armv7-a -mfpu=neon -mfloat-abi=hard -mcpu=cortex-a8 -mtune=cortex-a8 -O3 -g

PROF =  -pg -g # Add for gprof

%.o: %.c
	$(CC) $(CFLAGS) $(PROF) -c -o $@ $<

madgwick: main.o MadgwickAHRS.o
	$(CC) $(PROF) main.o MadgwickAHRS.o -lrt -lm -static -o $@

MadgwickAHRS_list.s: MadgwickAHRS.c
	$(CC) -Wa,-adhln -g MadgwickAHRS.c -c > MadgwickAHRS_list.s

clean:
	rm -f *.o madgwick *.s
```

The execution time for 5 consecutive runs are as follows:

```
debian@beaglebone:~/Project1$ ./madgwick
Average       6.567 us
Minimum       6.041 us
debian@beaglebone:~/Project1$ ./madgwick
Average       6.436 us
Minimum       6.041 us
debian@beaglebone:~/Project1$ ./madgwick
Average       6.289 us
Minimum       5.958 us
debian@beaglebone:~/Project1$ ./madgwick
Average       6.378 us
Minimum       6.041 us
debian@beaglebone:~/Project1$ ./madgwick
Average       6.233 us
Minimum       6.041 us
```

Profiling screenshot after doing perf report:

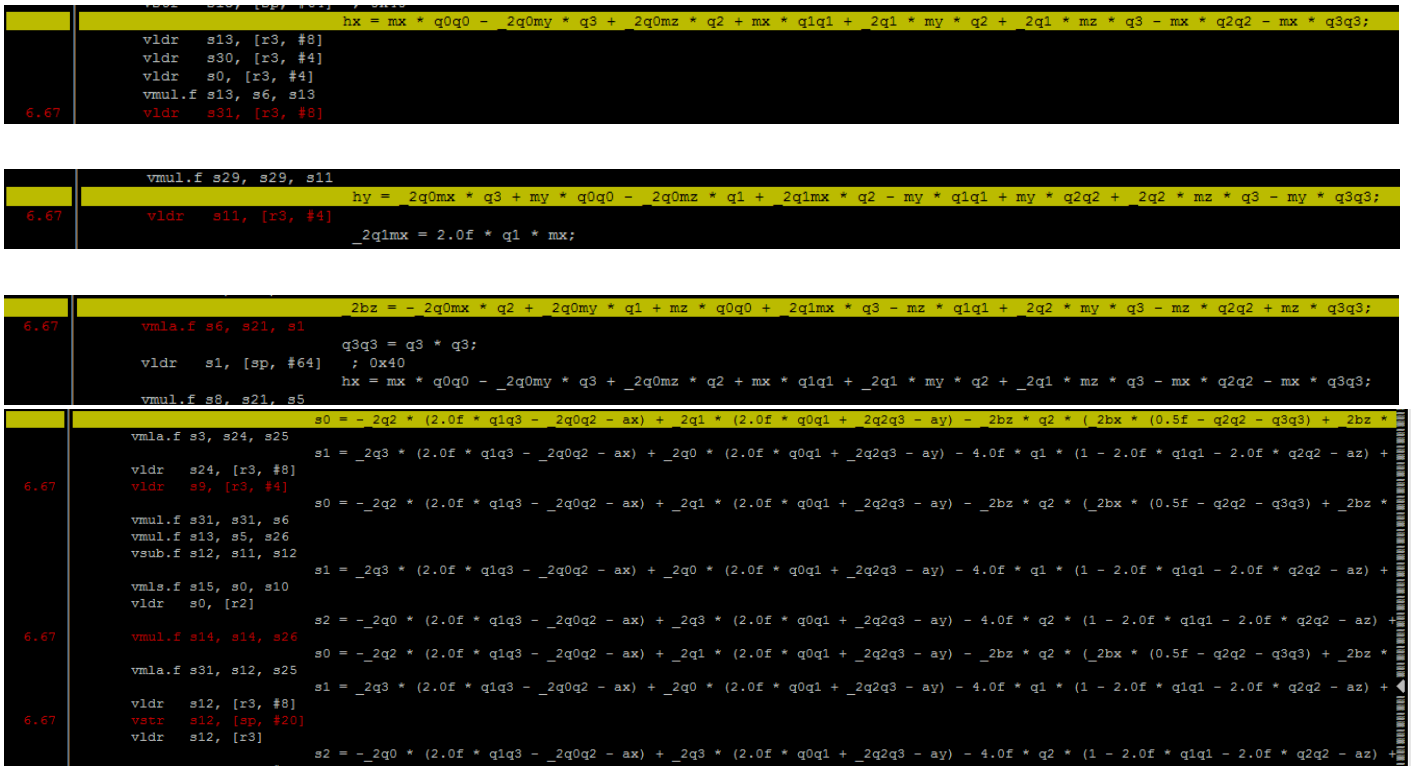


```
Samples: 135  of event 'cycles:ppp', Event count (approx.): 31044562
Overhead  Command   Shared Object       Symbol
  18.06%  madgwick  madgwick            [.] __ieee754_fmod
  14.36%  madgwick  madgwick            [.] __printf_fp_l
   9.31%  madgwick  madgwick            [.] MadgwickAHRSupdate
   7.08%  madgwick  [kernel.kallsyms]   [k] _raw_spin_unlock_irqrestore
   3.21%  madgwick  [kernel.kallsyms]   [k] vector_swi
   3.20%  madgwick  madgwick            [.] hack_digit
   2.55%  madgwick  [kernel.kallsyms]   [k] sample_to_timespec
   2.38%  madgwick  madgwick            [.] vfprintf
   1.60%  madgwick  madgwick            [.] __random_r
   1.58%  madgwick  madgwick            [.] __libc_do_syscall
   1.58%  madgwick  madgwick            [.] _IO_new_file_xsputn
   1.56%  madgwick  madgwick            [.] __mpn_mul_1
   1.47%  madgwick  madgwick            [.] write_gmon
   1.34%  madgwick  [kernel.kallsyms]   [k] filemap_map_pages
   1.31%  madgwick  madgwick            [.] _dl_get_origin
   1.24%  madgwick  [kernel.kallsyms]   [k] __radix_tree_lookup
   1.18%  madgwick  [kernel.kallsyms]   [k] __fget_light
   0.82%  madgwick  madgwick            [.] __random
   0.82%  madgwick  [kernel.kallsyms]   [k] pagecache_get_page
   0.80%  madgwick  madgwick            [.] __mpn_extract_double
   0.80%  madgwick  [kernel.kallsyms]   [k] ns_to_timespec.part.1
   0.80%  madgwick  [kernel.kallsyms]   [k] ret_fast_syscall
   0.80%  madgwick  [kernel.kallsyms]   [k] task_sched_runtime
   0.80%  madgwick  madgwick            [.] main
   0.80%  madgwick  [kernel.kallsyms]   [k] up_write
   0.80%  madgwick  [kernel.kallsyms]   [k] _cond_resched
   0.80%  madgwick  madgwick            [.] __mpn_rshift
   0.80%  madgwick  [kernel.kallsyms]   [k] queue_work_on
   0.80%  madgwick  madgwick            [.] __mpn_lshift
   0.80%  madgwick  [kernel.kallsyms]   [k] thread_cpu_clock_get
   0.80%  madgwick  madgwick            [.] __mcount_internal
   0.80%  madgwick  madgwick            [.] __strchrnul
   0.80%  madgwick  madgwick            [.] _IO_new_file_overflow
   0.80%  madgwick  [kernel.kallsyms]   [k] get_page_from_freelist
   0.79%  madgwick  madgwick            [.] strlen
   0.79%  madgwick  [kernel.kallsyms]   [k] restore_sigframe
   0.79%  madgwick  madgwick            [.] rand
   0.77%  madgwick  [kernel.kallsyms]   [k] atomic_notifier_call_chain
   0.76%  madgwick  [kernel.kallsyms]   [k] unmap_single_vma
   0.76%  madgwick  [kernel.kallsyms]   [k] __inet_lookup_established
   0.76%  madgwick  [kernel.kallsyms]   [k] mpage_map_and_submit_extent
Tip: To see callchains in a more compact form: perf report -g folded
```

4. **Optimized code using common sub-expression elimination (CSE) with compiler optimization level set to 3:**

Macro settings:
validate is set to 0
base is set to 0
ver is set to 2

Hot spots observed during the previous executions:



The following changes were made to the source code based on the hot spots. Common sub expressions were calculated so that the compiler can re-use these variables:

```
if (ver == 2){
    hx = mx*(q0q0 + q1q1 - q2q2 - q3q3) - _2q0my * q3 + _2q0mz * q2  + _2q1 * (my * q2 + mz * q3);
    hy = my*(q0q0 - q1q1 + q2q2 - q3q3) + _2q0mx * q3 - _2q0mz * q1 + _2q1mx * q2 + _2q2 * mz * q3;
    _2bz = mz*(q0q0 - q1q1 - q2q2 + q3q3) -_2q0mx * q2 + _2q0my * q1 + _2q1mx * q3 + _2q2 * my * q3;
}
```

```
if (ver == 2){
    float temp1 = 2.0f * q1q3 - _2q0q2 - ax;
    float temp2 = 2.0f * q0q1 + _2q2q3 - ay;
    float temp3 = _2bx * (0.5f - q2q2 - q3q3) + _2bz * (q1q3 - q0q2) - mx ;
    float temp4 = _2bx * (q1q2 - q0q3) + _2bz * (q0q1 + q2q3) - my ;
    float temp5 = _2bx * (q0q2 + q1q3) + _2bz * (0.5f - q1q1 - q2q2) - mz ;
    float temp6 = 4.0f * (1 - 2.0f * q1q1 - 2.0f * q2q2 - az);
    s0 = -_2q2 * temp1 + _2q1 * temp2 - _2bz * q2 * temp3 + (-_2bx * q3 + _2bz * q1) * temp4 + _2bx * q2 * temp5;
    s1 = _2q3 * temp1 + _2q0 * temp2 - q1 * temp6 + _2bz * q3 * temp3 + (_2bx * q2 + _2bz * q0) * temp4 + (_2bx * q3 - _4bz * q1) * temp5;
    s2 = -_2q0 * temp1 + _2q3 * temp2 - q2 * temp6 + (-_4bx * q2 - _2bz * q0) * temp3 + (_2bx * q1 + _2bz * q3) * temp4 + (_2bx * q0 - _4bz * q2) * temp5;
    s3 = _2q1 * temp1 + _2q2 * temp2 + (-_4bx * q3 + _2bz * q1) * temp3 + (-_2bx * q0 + _2bz * q2) * temp4 + _2bx * q1 * temp5;
}
```

```
if(ver == 2){

    _2q0 = 2.0f * q0;
    _2q1 = 2.0f * q1;
    _2q2 = 2.0f * q2;
    _2q3 = 2.0f * q3;
    _2q0mx = _2q0 * mx;
    _2q0my = _2q0 * my;
    _2q0mz = _2q0 * mz;
    _2q1mx = _2q1 * mx;
    _2q0q2 = _2q0 * q2;
    _2q2q3 = _2q2 * q3;
    q0q0 = q0 * q0;
    q0q1 = q0 * q1;
    q0q2 = q0 * q2;
    q0q3 = q0 * q3;
    q1q1 = q1 * q1;
    q1q2 = q1 * q2;
    q1q3 = q1 * q3;
    q2q2 = q2 * q2;
    q2q3 = q2 * q3;
    q3q3 = q3 * q3;

}
```

```
if(ver == 2){
    float sampFreq = (1.0f / sampleFreq);
    q0 += qDot1 * sampFreq;
    q1 += qDot2 * sampFreq;
    q2 += qDot3 * sampFreq;
    q3 += qDot4 * sampFreq;
}
```

The execution time for 5 consecutive runs are as follows:

```
debian@beaglebone:~/Project1$ ./madgwick
Average       6.006 us
Minimum       5.666 us
debian@beaglebone:~/Project1$ ./madgwick
Average       5.994 us
Minimum       5.708 us
debian@beaglebone:~/Project1$ ./madgwick
Average       5.966 us
Minimum       5.667 us
debian@beaglebone:~/Project1$ ./madgwick
Average       5.934 us
Minimum       5.666 us
debian@beaglebone:~/Project1$ ./madgwick
Average       5.916 us
Minimum       5.667 us
debian@beaglebone:~/Project1$
```

Diff between the original o/p and optimized code o/p:

As there are no differences in o/p the diff command doesn't show any different values between the two files. Hence the output is within the error margin of 0.01%. Screenshot of diff is as shown below:

```
debian@beaglebone:~/Project1$ diff op_base_log.txt op_log.txt
debian@beaglebone:~/Project1$
```

5. **Optimized code using common sub-expression elimination (CSE) with compiler optimization level set to 3 and removing the volatile keyword for the outputs q0-q3:**

Macro settings:
validate is set to 0
base is set to 0
ver is set to 2

The output variables are declared as extern volatile float in the MadgwickAHRSUpdate.h files. As we know if we declare variables as volatile the compiler turns off its optimization. These variables will be loaded from memory each time they are required instead of keeping them in registers.
These variables are declared volatile because their values are updated by MadgwickAHRSUpdate.c file but are being used in the main.c file. As the output variables are not being changed in the main function and only being used we can remove the volatile keyword and just keep it as extern float.

The following changes were made to the code:

In MadgwickAHRSUpdate.c file remove the declaration of q0-q3:

```
#if 0
volatile float q0 = 1.0f, q1 = 0.0f, q2 = 0.0f, q3 = 0.0f;      // quaternion of sensor frame relative to auxiliary frame
#endif
```

In MadgwickAHRSUpdate.h file make the following updates, change the declaration of q0-q3 from extern volatile float to extern float:

```
#if 0
extern volatile float q0, q1, q2, q3;   // quaternion of sensor frame relative to auxiliary frame
#endif

#if 1
extern float q0, q1, q2, q3;
#endif
//---------------------------------------------------------------------------------------------------
```

In main.c add the declaration of q0-q3:

```
#if 1
float q0 = 1.0f, q1 = 0.0f, q2 = 0.0f, q3 = 0.0f;
#endif
```

The execution time for 5 consecutive runs are as follows:

```
debian@beaglebone:~/Project1$ ./madgwick
Average       5.397 us
Minimum       5.125 us
debian@beaglebone:~/Project1$ ./madgwick
Average       5.518 us
Minimum       5.083 us
debian@beaglebone:~/Project1$ ./madgwick
Average       5.386 us
Minimum       5.083 us
debian@beaglebone:~/Project1$ ./madgwick
Average       5.334 us
Minimum       5.084 us
debian@beaglebone:~/Project1$ ./madgwick
Average       5.518 us
Minimum       5.083 us
debian@beaglebone:~/Project1$
```

Diff between the original o/p and optimized code o/p:

As there are no differences in o/p the diff command doesn't show any different values between the two files. Hence the output is within the error margin of 0.01%. Screenshot of diff is as shown below:

```
debian@beaglebone:~/Project1$ diff op_base_log.txt op_log.txt
debian@beaglebone:~/Project1$
```

6. **Moving the comparison operation of ax,ay,az and mx,my,mz to main function :**

Macro settings:
validate is set to 0
base is set to 0
ver is set to 2

The hot spot after executing the optimized code in section 5 was the "it" instruction that utilized the NEON comparison unit for the floating point comparison of ax,ay,az and mx,my,mz with 0 as shown below:

```
                if(!((ax == 0.0f) && (ay == 0.0f) && (az == 0.0f))) {
       10aa6:   vcmp.f s5, #0.0
       10aaa:   vmrs   APSR_nzcv, fpscr
       10aae:   vcmp.f s3, #0.0
 20.00 10ab2:   it     ne
       10ab4:   orrne. r3, r3, #1
       10ab8:   vmrs   APSR_nzcv, fpscr
```

To eliminate this, the comparison was moved to main function and the result is passed as argument to the MadgwickAHRSUpdate function. The comparison is done by type casting the values of mx,mymz,ax,ay and az to int. The following are the source code updates:

In main.c:

```
if (((int)az == 0) && ((int)ay == 0) && ((int)az == 0)){
      flag_a = 1;
}
else{
       flag_a = 0;
}
if (((int)mx == 0) && ((int)my == 0) && ((int)mz == 0)){
      flag_m = 1;
}
else{
      flag_m = 0;
}
```

In MadgwichAHRSUpdate.h:

```
void MadgwickAHRSupdate(float gx, float gy, float gz, float ax, float ay, float az, float mx, float my, float mz,int flag_a,int flag_m);
void MadgwickAHRSupdateIMU(float gx, float gy, float gz, float ax, float ay, float az,int flag_a);
```

In MadgwichAHRSUpdate.c:

```
if(flag_m) {
        MadgwickAHRSupdateIMU(gx, gy, gz, ax, ay, az,flag_a);
        return;
}

// Rate of change of quaternion from gyroscope
qDot1 = 0.5f * (-q1 * gx - q2 * gy - q3 * gz);
qDot2 = 0.5f * (q0 * gx + q2 * gz - q3 * gy);
qDot3 = 0.5f * (q0 * gy - q1 * gz + q3 * gx);
qDot4 = 0.5f * (q0 * gz + q1 * gy - q2 * gx);

// Compute feedback only if accelerometer measurement valid (a
if(!flag_a) {
```

The execution time for 5 consecutive runs are as follows:

```
debian@beaglebone:~/Project1$ ./madgwick
Average       5.103 us
Minimum       4.917 us
debian@beaglebone:~/Project1$ ./madgwick
Average       5.372 us
Minimum       4.958 us
debian@beaglebone:~/Project1$ ./madgwick
Average       5.254 us
Minimum       4.958 us
debian@beaglebone:~/Project1$ ./madgwick
Average       5.205 us
Minimum       4.916 us
debian@beaglebone:~/Project1$ ./madgwick
Average       5.186 us
Minimum       4.958 us
debian@beaglebone:~/Project1$
```

7. **Using different algorithm for calculating the inverse square root :**

Macro settings:

validate is set to 0
base is set to 0
ver is set to 2

The source code for the inverse square root function was changed to implement log2 and Newton's Algorithm as shown below:

```
float invSqrt(float x) {

        long i = *(long*)&x;
        i -= 1<<23;
        i >>= 1;
        i += 1<<29;
        //float halfx = 0.5f * x;
        //float y = x;
        //long i = /*(long)x;*/   *(long*)&y;
        //i = 0x5f3759df - (i>>1);
        //y = *(float*)&i;
        //y = y * (1.5f - (halfx * y * y));
        return  1/(*(float*)&i); //y
}
```

The execution time for 5 consecutive runs are as follows:

```
debian@beaglebone:~/Project1$ ./madgwick
Average       5.039 us
Minimum       4.791 us
debian@beaglebone:~/Project1$ ./madgwick
Average       4.928 us
Minimum       4.750 us
debian@beaglebone:~/Project1$ ./madgwick
Average       4.926 us
Minimum       4.750 us
debian@beaglebone:~/Project1$ ./madgwick
Average       5.185 us
Minimum       4.750 us
debian@beaglebone:~/Project1$ ./madgwick
Average       5.002 us
Minimum       4.750 us
debian@beaglebone:~/Project1$
```

The error for this approach was greater than 0.01% for approx 30 out of the 4000 outputs. The error output is as shown below:

Macro settings:
validate is set to 1

```
debian@beaglebone:~/ErrorCheck$ ./errorCheck
Error greater than 0.01 0.005213 0.006185
Error greater than 0.01 0.001772 0.002739
Error greater than 0.01 for -0.002251 -0.001282
Error greater than 0.01 for -0.007689 -0.006731
Error greater than 0.01 for 0.007630 0.006802
Error greater than 0.01 for 0.001157 0.000850
Error greater than 0.01 for 0.004490 0.003647
Error greater than 0.01 -0.004938 -0.005804
Error greater than 0.01 -0.007705 -0.008595
Error greater than 0.01 -0.011456 -0.013101
Error greater than 0.01 -0.010276 -0.011918
Error greater than 0.01 -0.008929 -0.010576
Error greater than 0.01 -0.003762 -0.005394
Error greater than 0.01 0.000114 -0.001518
Error greater than 0.01 -0.000946 -0.002573
Error greater than 0.01 -0.000934 -0.002558
Error greater than 0.01 for 0.001583 -0.000052
Error greater than 0.01 for 0.010151 0.008536
Error greater than 0.01 for 0.012444 0.010837
Error greater than 0.01 for -0.004873 -0.004334
Error greater than 0.01 0.003611 0.004144
Error greater than 0.01 for -0.015345 -0.013510
Error greater than 0.01 for -0.011301 -0.009454
Error greater than 0.01 for -0.004018 -0.002132
Error greater than 0.01 for -0.004006 -0.002109
Error greater than 0.01 0.001165 0.003083
Error greater than 0.01 0.006761 0.008695
```

Summary of the optimization performed and the execution time:

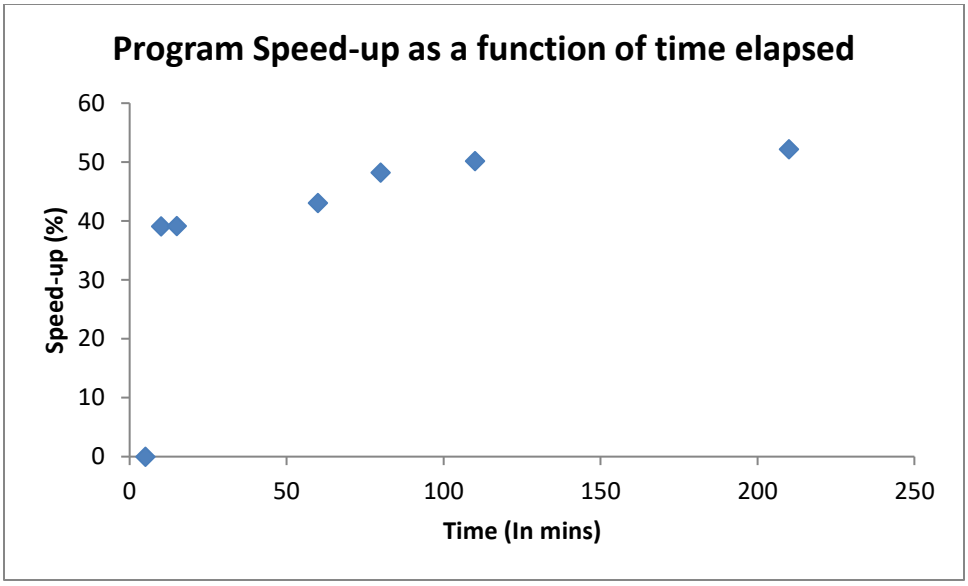
Sl No.	Version	Compiler Optimization	Exec Time 1 (us)	Exec Time 2 (us)	Exec Time 3 (us)	Exec Time 4 (us)	Exec Time 5 (us)	Average time (us)	Speed Up (%)
1	Starter Code (Version 1)	O0	10.473	10.454	10.551	10.478	10.461	10.4834	N/A
2	Starter Code (Version 1)	O1	6.401	6.282	6.381	6.348	6.509	6.3842	39.10182
3	Starter Code (Version 1)	O3	6.567	6.436	6.289	6.378	6.233	6.3806	39.13616
4	CSE (Version2)	O3	6.006	5.994	5.996	5.934	5.916	5.9692	43.06046
5	CSE (Version2) with removing volatile for q0-q3	O3	5.397	5.518	5.386	5.334	5.518	5.4306	48.1981
6	Moving the comparision to 0	O3	5.103	5.372	5.254	5.205	5.186	5.224	50.16884
7	Different algo for inv sqrt	O3	5.039	4.928	4.926	5.185	5.002	5.016	52.15293

Table 1

Scatter Plot:

Version (corresponds to sl no. in the above table 1)	Time (mins)	Speed Up (%)
1	5	0
2	10	39.10182
3	15	39.13616
4	60	43.06046
5	80	48.1981
6	110	50.16884
7	210	52.15293

Table 2


Program Speed-up as a function of time elapsed
Speed-up (%)
60
50
40
30
20
10
0
0
50
100
150
200
250
Time (In mins)