EECE 731 **Spring 2016**

Lab 1: SubSampling

Due Date: February 12, 2016 30 Points

Objective: Build, simulate and test the primary building block for the specialized peripheral we plan to build.

Tasks:

1) Write verilog files that implement the sub-sampling algorithm we have been developing in class. The number of fractional bits (precBits) should be a parameters in your design. Note eventually Stp will need to be an input to the system, but for now its a parameter. Finally note that a parameter for the total number of bits N will also have to be included. This number N should be the number of bits in the encoder count plus the precBits.

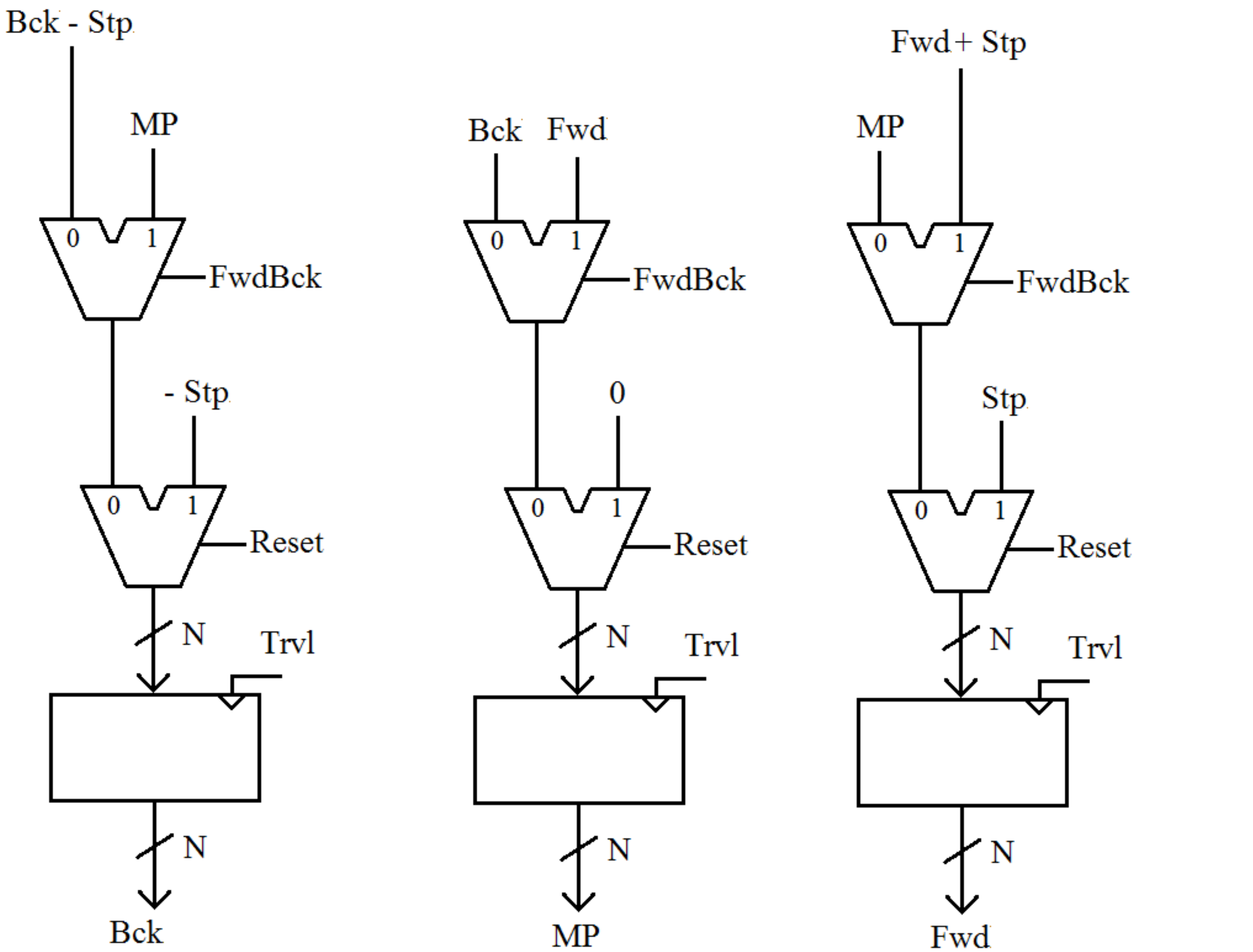
So how large should the encoder count be? Well it we assume that we will scale the counts per revolution of the encoder to get basically 100 counts per inch and we want to handle something irrational for distance, say 10 miles. We have 100 (counts/inch) * 12 (inch/foot) * 5280 (feet/mile) * 10 (miles) equals 63360000 which means we will need (log2(63360000) ~ 26 bits. So if with set N = 38 = (26+12) we should be safe for anything we will do.

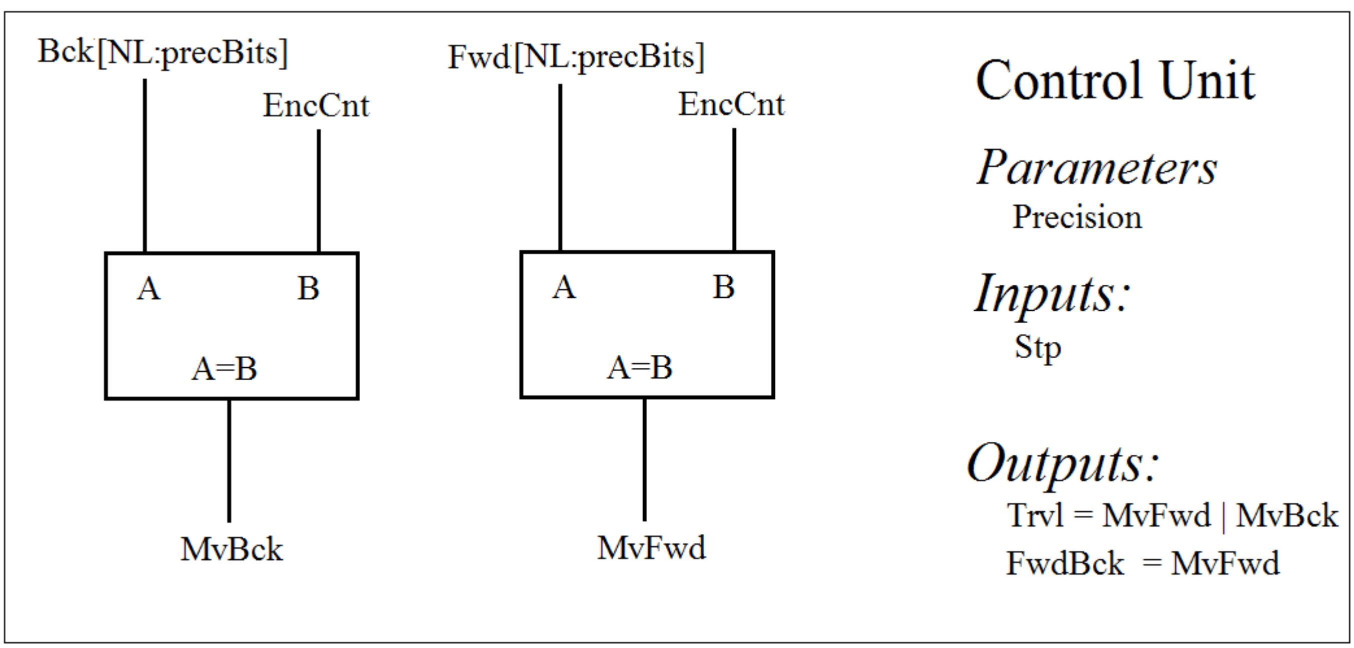
Note that an updated version of algorithm is included as an image of the system and a code verification.

2) Simulate the sub-sampling algorithm assuming a setting of 5.253 counts per inch. This translates into 5.253*4096 = 2l516 = Stp, or and integer part of 5 and a fractional part of 1036. Note that the simulation needs to include the encoder counting system we developed previously, as well as the digital filter for the encoder quadrature channels. At least 20 encoder steps forward and 40 encoder steps backward should be simulated.

3) Finally we will want to actually try it on the encoder. Thus we will reprogram the system to monitor two channels from the encoder, and set the counts per inch to be 29.35. At this setting, the 200 pulse encoder should give us 109 inches for four rotations of the encoder, and 436 inches for 16 turns.

Encoder SubSample Motion Detection System.





```c
#include "stdio.h"
// Set of sub sampling parameters and states for testing.
int PrecBits = 12;
long      StepDelta;
long      CurrCount;  // Current Encoder Count.
long       Forward;   // Forward Threshold.
long      MidPoint;   // Backward Threshold
long      Backward;   // Backward Threshold

// Set up sampling.
void InitSampling(int i, int precbits)
{
       PrecBits = precbits;
       StepDelta = i;

       // Initialize thresholds
       Forward = StepDelta;
       MidPoint = 0;
       Backward = -StepDelta;

} // end of InitSampling

int SampleMove()
{
       if (CurrCount == (Forward>>PrecBits))  // If past threshold.
       {
              Backward = MidPoint;    // save this into backward threshold
              MidPoint = Forward;     // Mid point form forward.
              Forward += StepDelta;          // Compute next threshold.

              return 1;  // Indicates moving forward.
       }

       else if (CurrCount == (Backward >> PrecBits))  // If below or at backward threshold
       {
              Forward = MidPoint;  // Save backward into forward.
              MidPoint = Backward;    // MidPoint from backward.
              Backward -= StepDelta;        // Compute backward threshold.

              return -1;  // Indicates motion backward.
       }
       return 0;  // Indicates no motion.
} // end of SampleMove

void main()
{
       // double and floating point based systems
       double x, res;
       int i, inches; // integer inch counter.
       FILE *File_Out;

       fopen_s(&File_Out, "SubSampling.csv", "w");  // File to record data

       res = 122856.0 / 1200; // Compute floating point Counts per inch
       i = (int)(res*4096.0 + 0.5);

       InitSampling( i, 12); // Initialize integer sub-sampling system

       x = 0.0;      // Start all sampling at 0 location.
       inches = 0;

       for (i = 0; i < 50000; i++) // Move forward 50000 encoder counts.
       {
              CurrCount++;

              if (SampleMove())  // Check if we have integer sub-sampling believes we have moved an inch.
              {
                     inches++;
                     x += res;

                     printf("Moved to %d, %f\n", inches, x);  // Display results and
                     fprintf(File_Out, "%lf, %lf, %d, %d\n",  // Record in file.
                                   x, (((double)MidPoint)/4096.0),inches, MidPoint);
              } // end of inch travel check.
       } // end of forward for loop.
```

```
        // Show end of forward motion.
        printf("Floating point calculation = %18.16lg\n", x / res);
        getchar();

        for (i = 0; i < 100000; i++) // Move backward 100000 encoder counts.
        {
                CurrCount--;         // Same as previous only backward
                if (SampleMove())
                {
                        inches--;
                        x -= res;

                        printf("Moved to %d, %f\n", inches, x);  // Display results and
                        fprintf(File_Out, "%lf, %lf, %d, %d\n",  // Record in file.
                                x, ((double)MidPoint / 4096.0), inches, MidPoint);
                } // end of inch travel check.
        } // end of backward for loop.

        printf("Floating point calculation = %18.16lg\n", x / res);
        getchar();

        for (i = 0; i < 50000; i++) // Move forward 50000 encoder counts.
        {
                CurrCount++;

                if (SampleMove())  // Check if we have integer sub-sampling believes we have moved an inch.
                {
                        inches++;
                        x += res;

                        printf("Moved to %d, %f\n", inches, x);  // Display results and
                        fprintf(File_Out, "%lf, %lf, %d, %d\n",  // Record in file.
                                x, ((double)MidPoint / 4096.0), inches, MidPoint);
                } // end of inch travel check.

        } // end of forward for loop.

        printf("Floating point calculation = %18.16lg\n", x / res);
        getchar();


}  // end of main.
```

Double	MidPnt/4096	Inch	MidPnt	Abs Rel Error
102.38	102.379883	1	419348	2.857E-07
204.76	204.759766	2	838696	2.857E-07
307.14	307.139648	3	1258044	2.86514E-07
409.52	409.519531	4	1677392	2.86311E-07
511.9	511.899414	5	2096740	2.86189E-07
614.28	614.279297	6	2516088	2.86107E-07
49551.92	49551.86328	484	202964432	2.8616E-07
49654.3	49654.24316	485	203383780	2.86159E-07
49756.68	49756.62305	486	203803128	2.86158E-07
49859.06	49859.00293	487	204222476	2.86157E-07
49961.44	49961.38281	488	204641824	2.86156E-07
49859.06	49859.00293	487	204222476	2.86157E-07
49756.68	49756.62305	486	203803128	2.86158E-07
49654.3	49654.24316	485	203383780	2.86159E-07
49551.92	49551.86328	484	202964432	2.8616E-07
409.52	409.519531	4	1677392	2.86311E-07
307.14	307.139648	3	1258044	2.86514E-07
204.76	204.759766	2	838696	2.857E-07
102.38	102.379883	1	419348	2.857E-07
0	0	0	0	0
-102.38	-102.379883	-1	-419348	2.857E-07
-204.76	-204.759766	-2	-838696	2.857E-07
-307.14	-307.139648	-3	-1258044	2.86514E-07
-409.52	-409.519531	-4	-1677392	2.86311E-07
-49654.3	-49654.24316	-485	-203383780	2.86159E-07
-49756.68	-49756.62305	-486	-203803128	2.86158E-07
-49859.06	-49859.00293	-487	-204222476	2.86157E-07
-49961.44	-49961.38281	-488	-204641824	2.86156E-07
-49859.06	-49859.00293	-487	-204222476	2.86157E-07
-49756.68	-49756.62305	-486	-203803128	2.86158E-07
-49654.3	-49654.24316	-485	-203383780	2.86159E-07
-716.66	-716.65918	-7	-2935436	2.86049E-07
-614.28	-614.279297	-6	-2516088	2.86107E-07
-511.9	-511.899414	-5	-2096740	2.86189E-07
-409.52	-409.519531	-4	-1677392	2.86311E-07
-307.14	-307.139648	-3	-1258044	2.86514E-07
-204.76	-204.759766	-2	-838696	2.857E-07
-102.38	-102.379883	-1	-419348	2.857E-07
0	0	0	0	0