Seth Pregler
ECEN 489-508
02.01.22

Lab 1 Postlab Submission

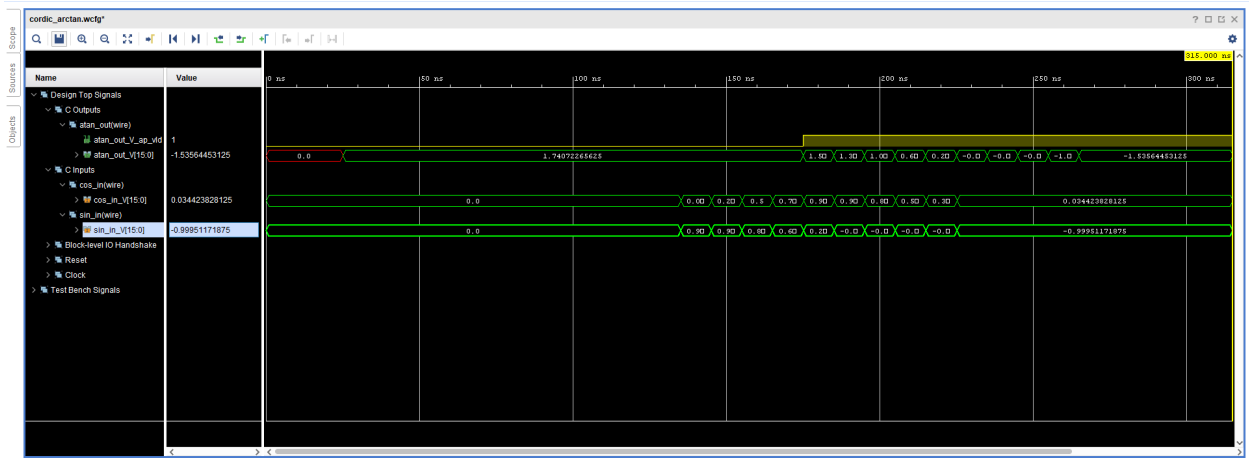

Figure 1 - Behavior Simulation of CORDIC-based arctan

Figure 1 shown above, demonstrates the behavioral simulation of the CORDIC-based arctangent implementation. The sin_in and cos_in input wires represent the 10 angles specified in the testbench file, in radians. The atan_output wire represents the respective output signals calculated from the CORDIC algorithm. The CORDIC-based arctangent implementation was similar to the sin and cos algorithms however, $d_k = -\ sign(y_k)$ instead of $d_k = sign(z_k)$ as it did for sin and cos. This effectively just changed the signs of the HLS pipeline shown below.

```
#pragma HLS pipeline II=1
for(int m=0; m<N; m++){
    // 4)
    if(zi[m] >= 0){
        xi[m+1] = xi[m] - (yi[m] >> m);
        yi[m+1] = yi[m] + (xi[m] >> m);
        zi[m+1] = zi[m] - atan4cordic[m];
    }
    else{
        xi[m+1] = xi[m] + (yi[m] >> m);
        yi[m+1] = yi[m] - (xi[m] >> m);
        zi[m+1] = zi[m] + atan4cordic[m];
    }
}
```

Figure 2.a - Sine/Cosine Pipeline

```
#pragma HLS pipeline II=1
for(int m=0; m<N; m++){
    // 2)
    if(yi[m] >= 0){
        xi[m+1] = xi[m] + (yi[m] >> m);
        yi[m+1] = yi[m] - (xi[m] >> m);
        zi[m+1] = zi[m] + atan4cordic[m];
    }
    else{
        xi[m+1] = xi[m] - (yi[m] >> m);
        yi[m+1] = yi[m] + (xi[m] >> m);
        zi[m+1] = zi[m] - atan4cordic[m];
    }
}
```

Figure 2.b - Arctan Pipeline