Juan Arturo Garza

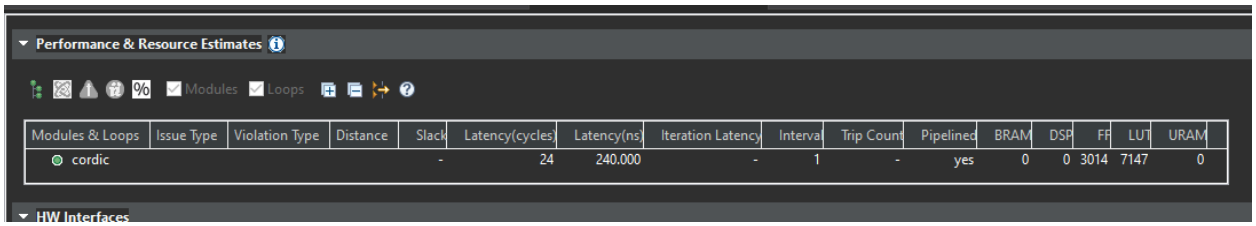
September 29th 2022

Elec 522

High Level Design

The purpose of this project is have a way to find the sine and cosine values of a given theta, and to find the arctan of a given x,y ratio without the use of multipliers, instead shifts can be used and add and subtract operations

Final Resource Usage from HLS



Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
cordic				-	24	240.000	-	1	-	yes	0	0	3014	7147	0

Notice no DSPs were used in this solution

Final Design Choices

I used the HLS Pipeline which helped reduce the clock cycles. This was done as some of the loops within my code could be pipelined. The architecture did require other resources without DSP and may explain why it takes longer to compute in cycles as compared to if it used multiply operation. This design does fit the constraints of NO DSP being used at all however this may come at a cost of cycles being higher however.

For this design there are four varying inputs to the block, x_in, y_in, a theta value, and a state boolean. The state boolean changes whether the block is in either rotation or arctan mode with signal high indicating rotation and signal low dictating arctan mode, the outputs not used in either state are set to zero. The x_in and y_in are for the arctan mode. The theta is for the rotation mode.

For the rotation mode, there were x and y shifts which resulted in a value, which was then shifted again to achieve the final result. Similar process occurred for arctan mode.

Note that the code was based on the number of iterations I would want to do the shifts (32), where each one would make its coverage to a more correct value however it would cost more loops (latency) and power.

Inputs:

X -x coordinate, **Y -** y coordinate, **Theta -**Theta value for arctan **State -** Bool 1 being sin/cos and 0 being arctan mode

Test Bench HLS

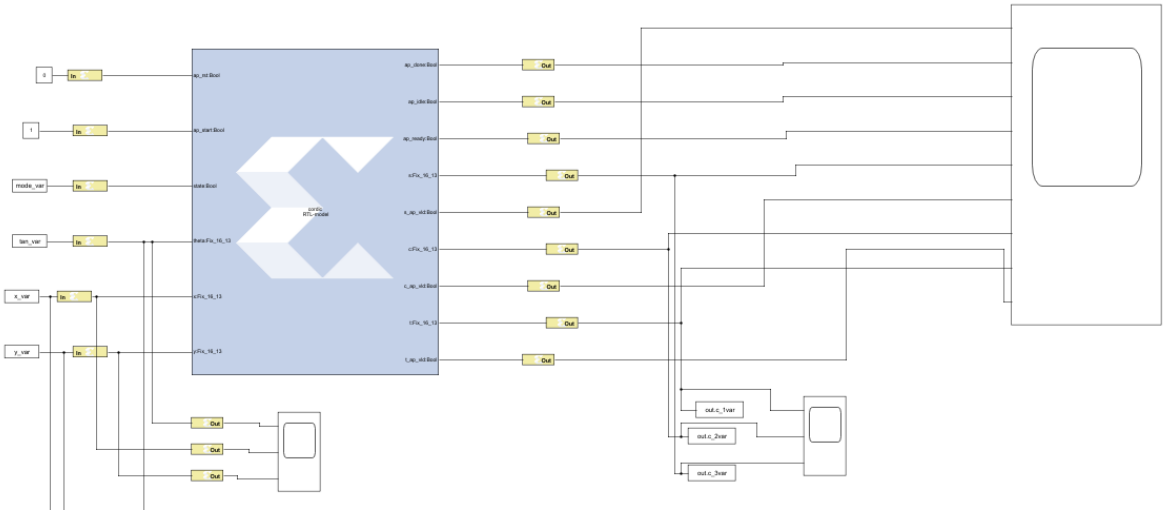
```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
   Compiling ../../../../cordic-top.cpp in debug mode
   Generating csim.exe
degree=-90, radian=-1.570801, cos=-0.002686, sin=-0.996826
degree=-75, radian=-1.309082, cos=0.260010, sin=-0.964355
degree=-60, radian=-1.047241, cos=0.497437, sin=-0.864624
degree=-45, radian=-0.785400, cos=0.706787, sin=-0.706177
degree=-30, radian=-0.523682, cos=0.865356, sin=-0.499634
degree=-15, radian=-0.261841, cos=0.962280, sin=-0.259277
degree=0, radian=0.000000, cos=0.999390, sin=-0.000732
degree=15, radian=0.261719, cos=0.964111, sin=0.258789
degree=30, radian=0.523560, cos=0.864136, sin=0.499146
degree=45, radian=0.785278, cos=0.705688, sin=0.705688
degree=60, radian=1.047119, cos=0.498901, sin=0.864258
degree=75, radian=1.308960, cos=0.258301, sin=0.964111
degree=90, radian=1.570679, cos=-0.000854, sin=0.996582
Total_Error_Sin=inf, Total_error_Cos=-59359.876559,
x_in=0.000000, y_in=-1.000000, t_theta=-1.570312
x_in=0.258789, y_in=-0.965942, t_theta=-1.309082
x_in=0.500000, y_in=-0.866089, t_theta=-1.047119
x_in=0.706909, y_in=-0.707031, t_theta=-0.785156
x_in=0.865967, y_in=-0.500000, t_theta=-0.523438
x_in=0.965820, y_in=-0.258911, t_theta=-0.261475
x_in=1, y_in=0, t_theta=0.000244
x_in=0.965820, y_in=0.258789, t_theta=0.261475
x_in=0.865967, y_in=0.500000, t_theta=0.523438
x_in=0.706909, y_in=0.706909, t_theta=0.785156
x_in=0.500000, y_in=0.865967, t_theta=1.047119
x_in=0.258789, y_in=0.965820, t_theta=1.309082
x_in=0.000000, y_in=1.000000, t_theta=1.570312
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

The values above show both sin/cos mode (rotation) and the tangent mode

```
int main(int argc, char **argv)
{   FILE *fp;
    COS_SIN_TYPE s;         //sine
    COS_SIN_TYPE c;         //cos
    THETA_TYPE_in t;            //cos

    THETA_TYPE radian;      //radian versuin of degree
    bool state;
    COS_SIN_TYPE x;         //x_cord
    COS_SIN_TYPE y;         //y_cord

    //zs=sin, zc=cos using math.h in VivadoHLS
    double zs, zc;          // sine and cos values calculated from math.

    //Error checking
    double Total_Error_Sin=0.0;
    double Total_error_Cos=0.0;
    double error_sin=0.0, error_cos=0.0;

    //fp=fopen("out.dat","w");

    for(int i=0;i<NUM_DEGREE;i=i+15) {
            radian = i*M_PI/180;
            cordic(true, radian, 1.0, 0, s, c, t);
            zs = sin((double)radian);
            zc = cos((double)radian);
            error_sin=(abs_double((double)s-zs)/zs)*100.0;
            error_cos=(abs_double((double)c-zc)/zc)*100.0;
            Total_Error_Sin=Total_Error_Sin+error_sin;
            Total_error_Cos=Total_error_Cos+error_cos;
            fprintf(stdout, "degree=%d, radian=%f, cos=%f, sin=%f\n", i, (double)radian, (double)c, (double)s);
    }
    printf ("Total_Error_Sin=%f, Total_error_Cos=%f, \n", Total_Error_Sin, Total_error_Cos);


// x  1      0.9659 0.866  0.707  .5    0.2588
// y  0      0.2588 0.5    0.707  0.866 0.9659
    x= 1.0; y= 0.0;     cordic(false, 0, x, y, s, c, t);
    fprintf(stdout, "x_in=%d, y_in=%d, t_theta=%f\n", (double)x, (double)y, (double)t);
    x= 0.9659; y=  0.2588;  cordic(false, 0, x, y, s, c, t);
    fprintf(stdout, "x_in=%f, y_in=%f, t_theta=%f\n", (double)x, (double)y, (double)t);
    x= 0.866; y=  0.5;  cordic(false, 0, x, y, s, c, t);
    fprintf(stdout, "x_in=%f, y_in=%f, t_theta=%f\n", (double)x, (double)y, (double)t);
    x= 0.707; y=  0.707;    cordic(false, 0, x, y, s, c, t);
    fprintf(stdout, "x_in=%f, y_in=%f, t_theta=%f\n", (double)x, (double)y, (double)t);
    x=0.5;    y= 0.866;     cordic(false, 0, x, y, s, c, t);
    fprintf(stdout, "x_in=%f, y_in=%f, t_theta=%f\n", (double)x, (double)y, (double)t);
    x= 0.2588; y=  0.9659;  cordic(false, 0, x, y, s, c, t);
    fprintf(stdout, "x_in=%f, y_in=%f, t_theta=%f\n", (double)x, (double)y, (double)t);
    x= 0.0;    y=  1.0;     cordic(false, 0, x, y, s, c, t);
    fprintf(stdout, "x_in=%f, y_in=%f, t_theta=%f\n", (double)x, (double)y, (double)t);
```

Model Composer

=



For this design there are four varying inputs to the block, x_in, y_in, a theta value, and a state boolean. The state boolean changes whether the block is in either rotation or arctan mode.

For testing there are two states: tan_mode.m tests the arctan mode of the mode, cos_sin_mode.m is the rotation mode

Resource Usage:

Post Synthesis Resources: Clicking on an instance name highlights corresponding block/subsystem in the model

Name	BRAMs (140)	DSPs (220)	LUTs (53200)	Registers (106400)
▸ Hardware_proj3	0	0	2353	1448

LUT: 2353

Reg: 1448

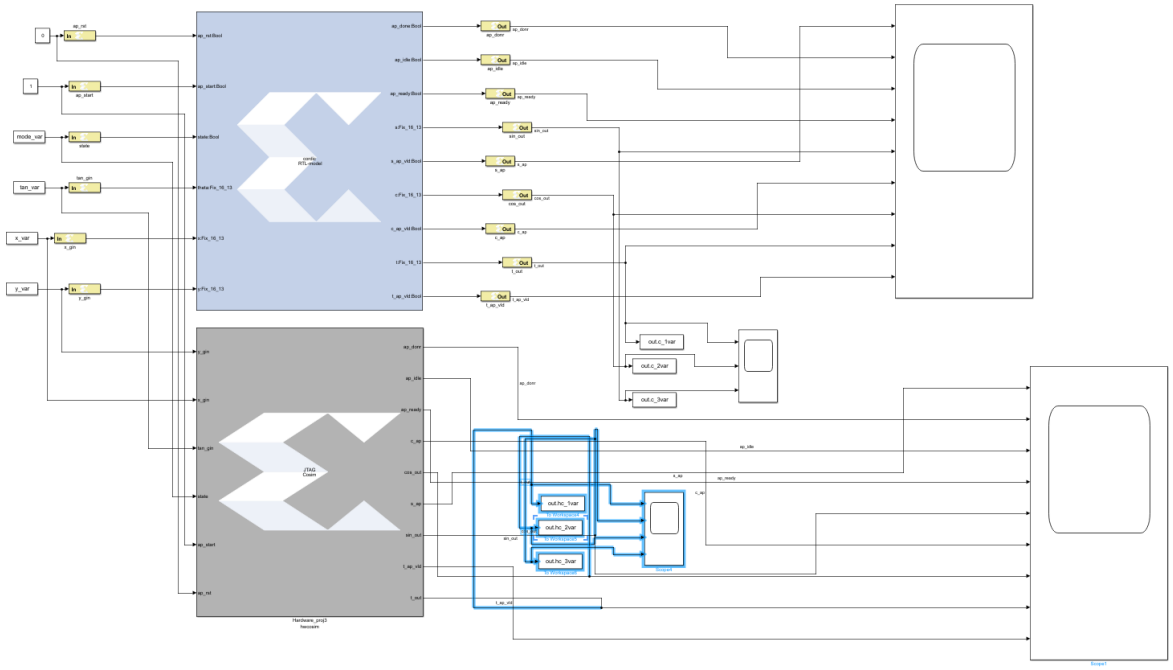
Timing:

Post Synthesis Timing Paths: Clicking on an instance name highlights corresponding block/subsystem in the model

Violation type : setup — Select Columns — Status : PASSED

	Slack (ns)	Delay (ns)	Logic Delay (ns)	Routing Delay (ns)	Levels of Logic	Source	Destination	Source Clock	Destination Clock	Path Constraints
1	3.6460	6.3790	4.6810	1.6980	15	Hardware_proj3/Vitis HLS1	Hardware_proj3/Vitis HLS1	clk	clk	create_clock -name clk -period 10 [get_ports clk]

Hardware On FPGA:



Testing: Put on separate file with Hcosim
Rotation Mode (tested on disp_hcosim_scmode.m)

```
>> out_disp_hcosim_scmode

theta_vals_hcosim_scmode =

         0    0.2617    0.5235    0.7853    1.0470    1.3080    1.5700


cos_vals_hcosim_scmode =

    0.9994    0.9641    0.8641    0.7057    0.4990    0.2596    0.0004


sin_vals_hcosim_scmode =

   -0.0007    0.2588    0.4991    0.7057    0.8643    0.9639    0.9980

>>
```

Tan mode with error on bottom tested with out_disp_hcosim_tmode.m

```
>> out_disp_hcosim_tmode

tan_input_hcs =

         0    0.2679    0.5774    1.0000    1.7320    3.7322


arctan_vals =

    0.0002    0.2615    0.5234    0.7852    1.0471    1.3091


expected_vals_thcm =

         0    0.2618    0.5236    0.7854    1.0472    1.3090


errror_hctan =

    0.0244   -0.0279   -0.0199   -0.0242   -0.0066    0.0075
```

Comparing Cordic Built vs my Implementation:
The built in Cordic build was faster by 4 cycles (20) cycles to compute results and my results are shown below:

Out_disp_built_cordic_cos.m tests the rotation mode:

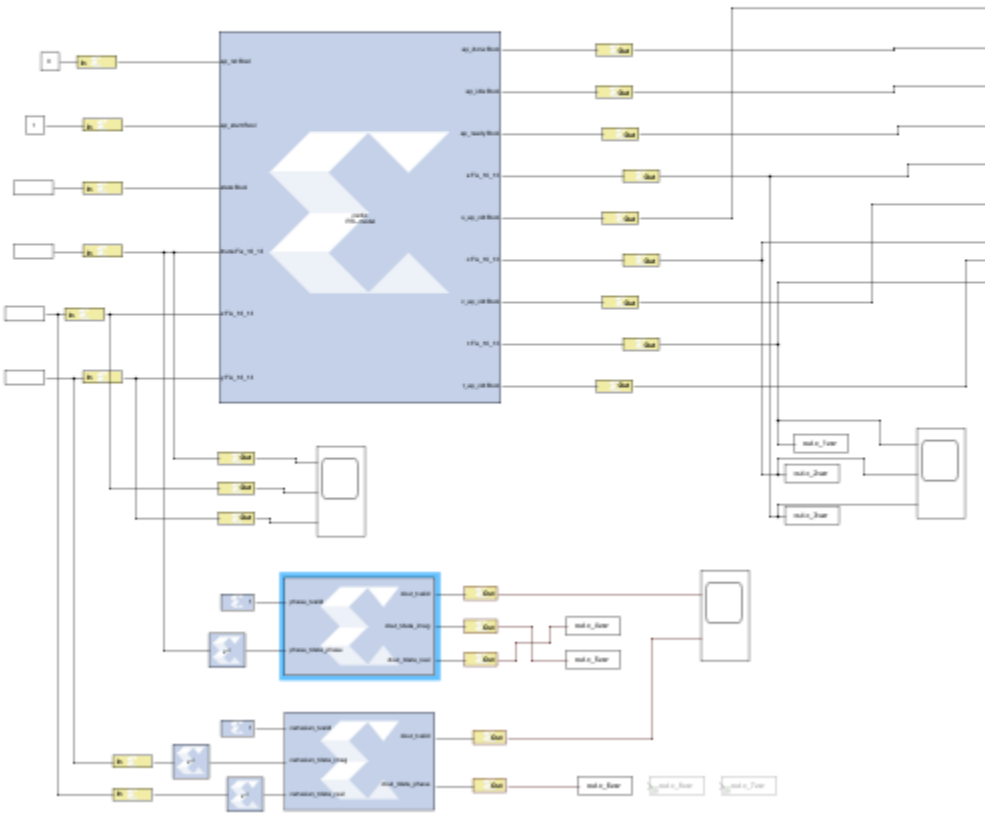
```
>> out_disp_built_cordic_cos

theta_vals =

         0    0.2617    0.5235    0.7853    1.0470    1.3080    1.5700


cos_hls_out =

    0.9994    0.9641    0.8641    0.7057    0.4990    0.2596    0.0004    0.9994


sin_hls_out =

   -0.0007    0.2588    0.4991    0.7057    0.8643    0.9639    0.9980   -0.0007


x_cordic =

    1.0000    0.9659    0.8660    0.7072    0.5001    0.2598    0.0008    1.0000


y_cordic =

   -0.0001    0.2587    0.5000    0.7070    0.8659    0.9656    1.0000   -0.0001


diff_x =

    0.0006    0.0018    0.0018    0.0015    0.0010    0.0002    0.0004    0.0006


diff_y =

    0.0007    0.0001    0.0009    0.0013    0.0016    0.0018    0.0020    0.0007
```

This shows how similar my implementation and the prebuilt vitis cordic build are in the value, diff_x and diff_y are the error shown between the two vitis cordic and my implementation

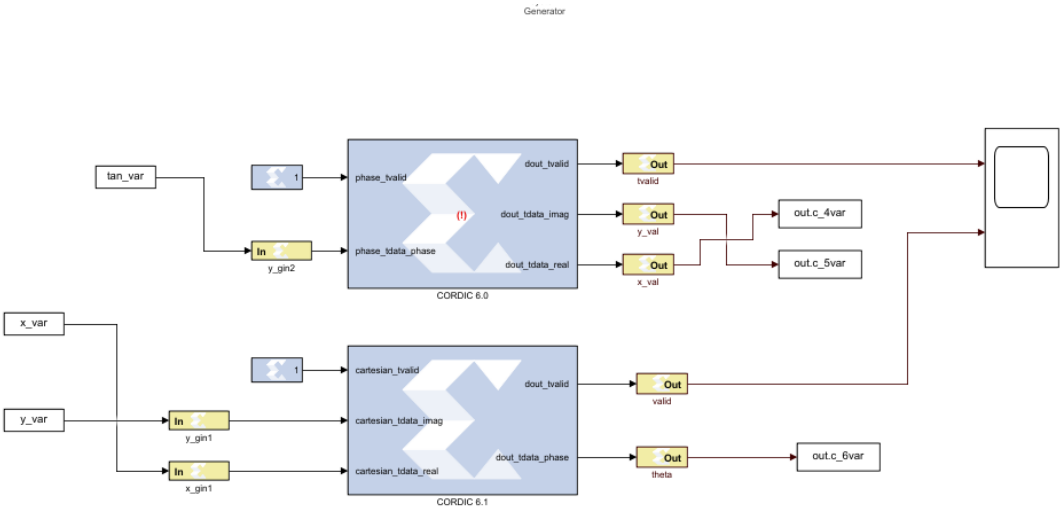
Tan mode with vitis vs mine: (tested with out_disp_built_cordic_tan.m)

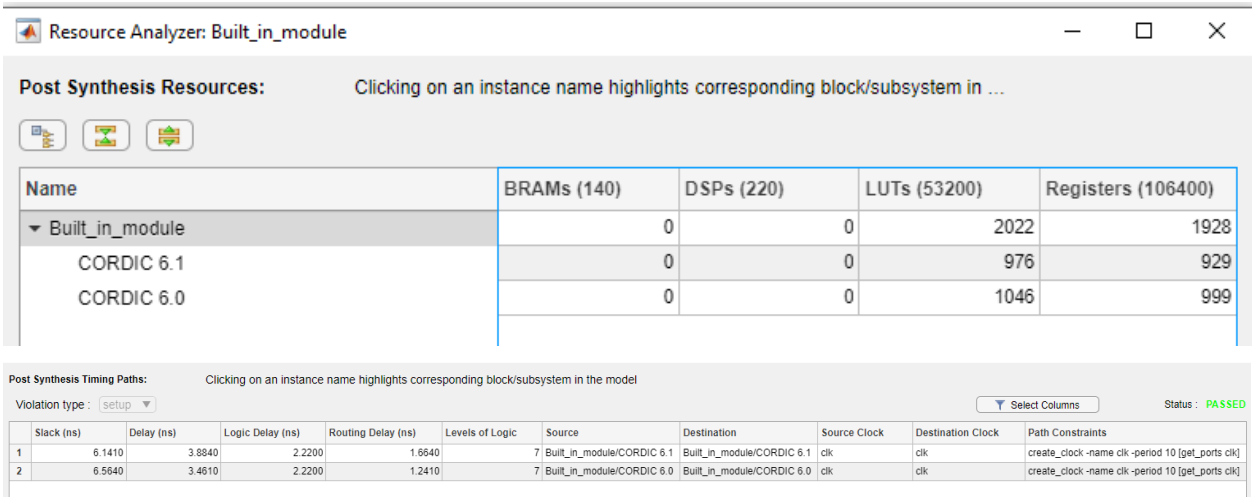
```
>> out_disp_built_cordic_tan

tan_input =

         0    0.2679    0.5774    1.0000    1.7320    3.7322


arctan_output_hls =

    0.0002    0.2615    0.5234    0.7852    1.0471    1.3091    1.5703    0.0002


arctan_cordic =

   -0.0001    0.2617    0.5236    0.7853    1.0471    1.3090    1.5707   -0.0001


arctan_diff =

   1.0e-03 *

    0.3662    0.2441    0.1221    0.1221         0    0.1221    0.3662    0.3662
```

Area Usage/resource difference:



Circuit used for the resource analysis:





This shows that the built in module uses less resources and from my timing analysis also uses less clock cycles. I would say that my design needs some work before it can match that of the pre built module, nonetheless it is sufficient for the calculations

Arm Vitis Section:

```
--------------------Rotation MODE--------------------

State: 1,  x: 1,  y: 0,  Theta_in:0, cos: 0.99939,  sin: -0.000732422,  theta_out: 0

State: 1,  x: 1,  y: 0,  Theta_in:15, cos: 0.964111,  sin: 0.258789,  theta_out: 0

State: 1,  x: 1,  y: 0,  Theta_in:30, cos: 0.864136,  sin: 0.499146,  theta_out: 0

State: 1,  x: 1,  y: 0,  Theta_in:45, cos: 0.705566,  sin: 0.705811,  theta_out: 0

State: 1,  x: 1,  y: 0,  Theta_in:60, cos: 0.498901,  sin: 0.864258,  theta_out: 0

State: 1,  x: 1,  y: 0,  Theta_in:75, cos: 0.258301,  sin: 0.964111,  theta_out: 0

State: 1,  x: 1,  y: 0,  Theta_in:90, cos: -0.000854492,  sin: 0.996582,  theta_out: 0
--------------------ARCTAN MODE--------------------
State: 0,  x: 0,  y: -1,  Theta_in:-90, cos: 0,  sin: 0,  theta_out: -1.57031

State: 0,  x: 0.5,  y: -0.865967,  Theta_in:-60, cos: 0,  sin: 0,  theta_out: -1.04712

State: 0,  x: 0.706909,  y: -0.706909,  Theta_in:-45, cos: 0,  sin: 0,  theta_out: -0.7854

State: 0,  x: 0.96582,  y: -0.258789,  Theta_in:-15, cos: 0,  sin: 0,  theta_out: -0.261475

State: 0,  x: 1,  y: 0,  Theta_in:0, cos: 0,  sin: 0,  theta_out: 0.000244141

State: 0,  x: 0.96582,  y: 0.258789,  Theta_in:15, cos: 0,  sin: 0,  theta_out: 0.261475

State: 0,  x: 0.865967,  y: 0.5,  Theta_in:30, cos: 0,  sin: 0,  theta_out: 0.523438

State: 0,  x: 0.706909,  y: 0.706909,  Theta_in:45, cos: 0,  sin: 0,  theta_out: 0.785156

State: 0,  x: 0.5,  y: 0.865967,  Theta_in:60, cos: 0,  sin: 0,  theta_out: 1.04712

State: 0,  x: 0.258789,  y: 0.96582,  Theta_in:75, cos: 0,  sin: 0,  theta_out: 1.30908

State: 0,  x: 0,  y: 1,  Theta_in:90, cos: 0,  sin: 0,  theta_out: 1.57031
```

This section tests both the Rotation and sin/cos section of the code, as seen above, the numbers do match those from HLS and model composer meaning the behavior is doing as expected.