Problem 1: DSP Arithmetic

a), The CORDIC divider works only for z = y/x. In this problem, z = 1/N. The magnitude of N must be greater than 1. The minimum value for N is 2^{-5}. When N is less than 1, the N needs to be multiplied by 2^5 to make sure the N is greater than 1, which is the correct range for the N. The output calculation is $\frac{1}{2^5 * N}$. Therefore, we need to multiply 2^5 again to get the correct result (1/N).

N is an 8-bit input operand with 5 fractional bits.

The maximum of N is $2^1 + 2^0 + 2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5}$ = 3.96875

Shift to the left 5 bits, but we already have 2 bits integer. We need to add 3 bits. s(11, 5)

We need 19 iteration. s(25 ,19) 0.251968504*0.001%=0.00000252; 2^-19=0.000001907;

2^-18=0.000003815. 2^-19 is smaller than 0.00000252. Hence, the fraction part would need 19 bits. The final output bits should be s(25,19)

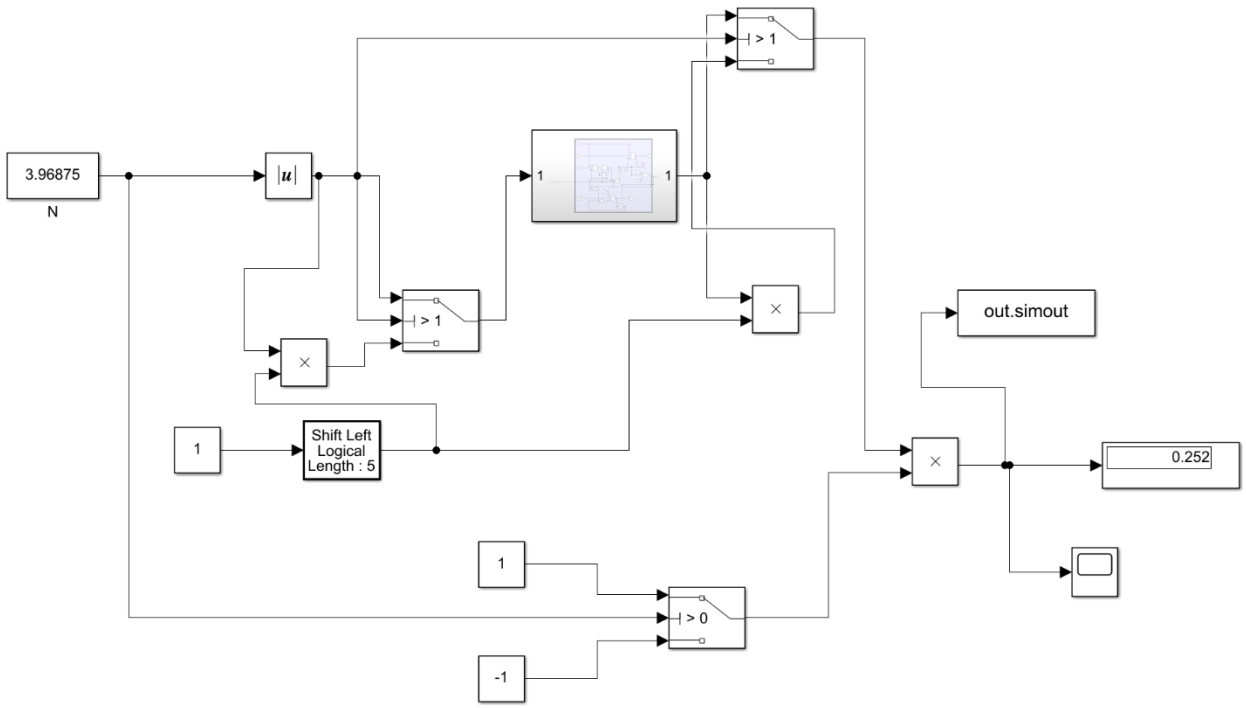
$$x_{i+1} = x_i - 0 \cdot y_i \cdot d_i \cdot 2^{-i} = x_i$$

$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$

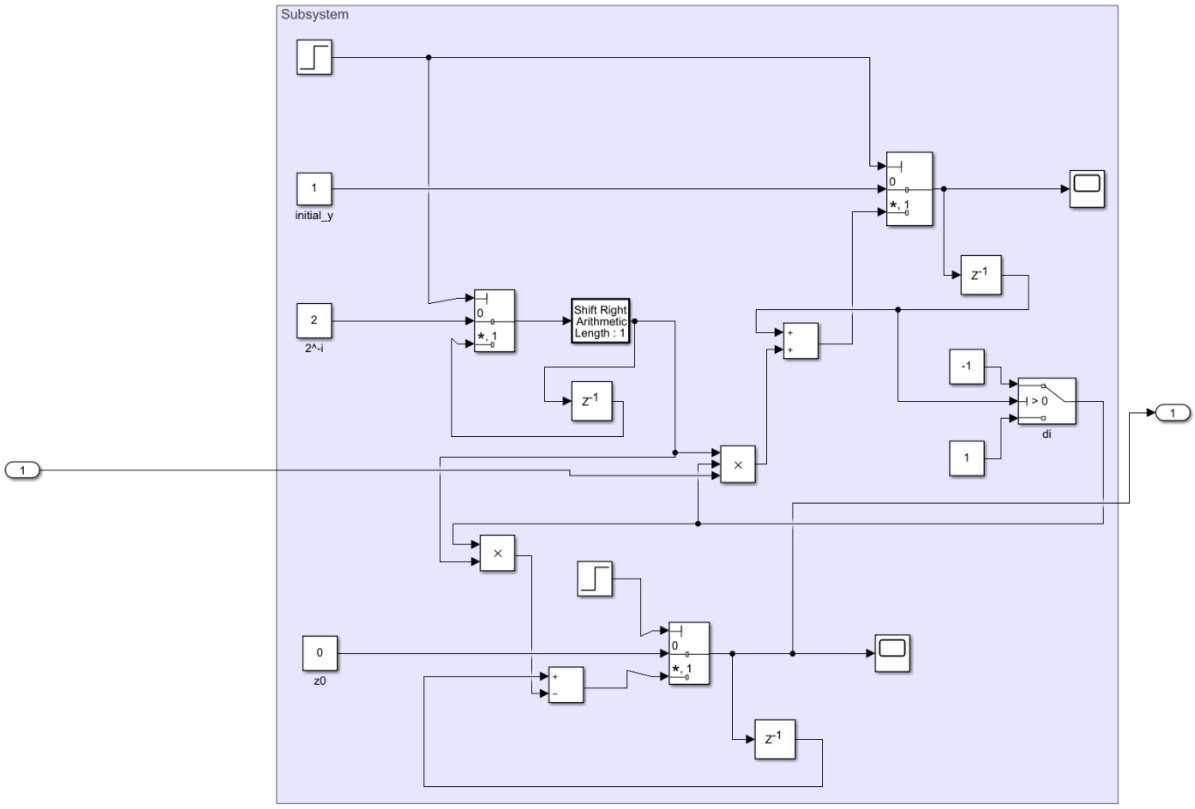
$$z_{i+1} = z_i - d_i \cdot (2^{-i})$$

lecture 7.20

Top level CORDIC Divider Architecture:



CORDIC Divider subsystem:

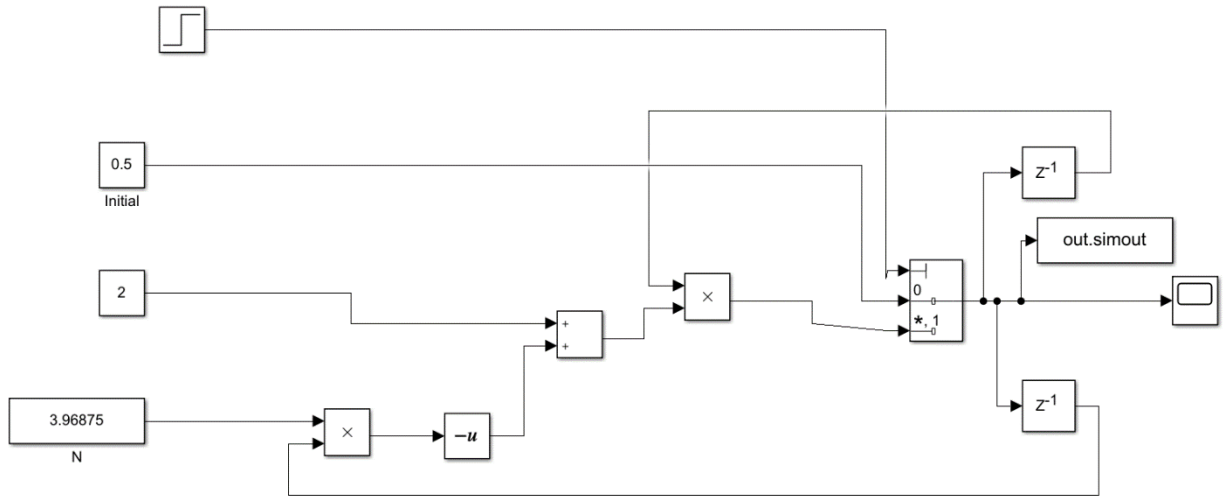


	A	B	C	D	E
1	0		0.251968504		-100.000000%
2	0.5		0.251968504		98.437500%
3	0.25		0.251968504		-0.781250%
4	0.375		0.251968504		48.828125%
5	0.3125		0.251968504		24.023437%
6	0.28125		0.251968504		11.621094%
7	0.265625		0.251968504		5.419922%
8	0.2578125		0.251968504		2.319336%
9	0.25390625		0.251968504		0.769043%
10	0.251953125		0.251968504		-0.006104%
11	0.252929688		0.251968504		0.381470%
12	0.252441406		0.251968504		0.187683%
13	0.252197266		0.251968504		0.090790%
14	0.252075195		0.251968504		0.042343%
15	0.25201416		0.251968504		0.018120%
16	0.251983643		0.251968504		0.006008%
17	0.251968384		0.251968504		-0.000048%
18	0.251976013		0.251968504		0.002980%
19	0.251972198		0.251968504		0.001466%
20	0.251970291		0.251968504		0.000709%
21	0.251969337		0.251968504		0.000331%
22	0.251968861		0.251968504		0.000142%
23	0.251968622		0.251968504		0.000047%

It is 19 iteration.

b), Newton-Rhapson function: x(k+1) = x(k) * (2 – N*x(k)).

Initial condition is 0.5.



1	0.5	0.2519685	98.43750%
2	0.007813	0.2519685	-96.89941%
3	0.015383	0.2519685	-93.89496%
4	0.029826	0.2519685	-88.16264%
5	0.056122	0.2519685	-77.72652%
6	0.099744	0.2519685	-60.41411%
7	0.160003	0.2519685	-36.49865%
8	0.218402	0.2519685	-13.32152%
9	0.247497	0.2519685	-1.77463%
10	0.251889	0.2519685	-0.03149%
11	0.251968	0.2519685	-0.00001%
12	0.251969	0.2519685	0.00000%
13	0.251969	0.2519685	0.00000%

For the Newton-Rhapson, it needs 10 iterations to converge the same accuracy. The Newton-Rhapson has less iterations than CORDIC Divider when they converge to the same accuracy.

c), In order to guarantee convergence in 4 iterations, we firstly need to calculate error (0).

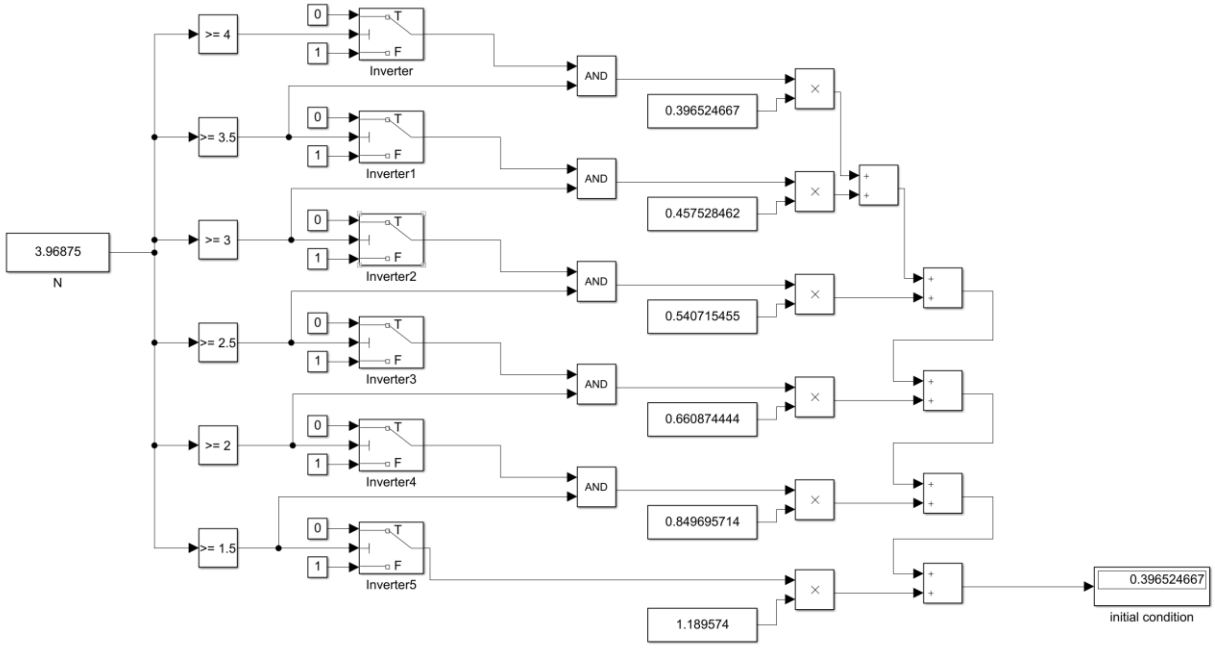
error(4)=0.001%, error(3)=sqrt(0.001%)=0.00316, error(2)=sqrt(error(3))=0.05623, error(1)=sqrt(error(2))=0.2371, error(0)=sqrt(error(0))=0.48697. Based on the error(0), we can calculate the initial condition x(0) with the following equation from lecture 7.26

$$x_d(k+1) = x_d(k) \cdot (2 - N \cdot x_d(k))$$

$$y_d(k+1) = y_d(k) \cdot (2 - y_d(k))$$

$$e_d(k) = y_d(k) - 1$$

	N range	N middlle	e(0)	y(0)=e(0)+1	x(0)=y(0)/N, initial condition
2	1~1.5	1.25	0.4869675	1.4869675	1.189574
3	1.5~2	1.75	0.4869675	1.4869675	0.849695714
4	2~2.5	2.25	0.4869675	1.4869675	0.660874444
5	2.5~3	2.75	0.4869675	1.4869675	0.540715455
6	3~3.5	3.25	0.4869675	1.4869675	0.457528462
7	3.5~3.96875	3.75	0.4869675	1.4869675	0.396524667
8					



E	F	G	H
1	0.396524667	0.251968504	57.370727%
2	0.169035582	0.251968504	-32.914003%
3	0.224671959	0.251968504	-10.833316%
4	0.249011383	0.251968504	-1.173607%
5	0.251933799	0.251968504	-0.013774%
6	0.251968499	0.251968504	-0.000002%
7	0.251968504	0.251968504	0.000000%

After I change the initial condition to 0.396524667, the iterations decrease to 5 which is less than part b when converge to the same accuracy.

Problem 2: Iteration Bound

$$iteration\ bound = \frac{loop\ delay}{number\ of\ registers\ in\ the\ loop}$$

For the CORDIC divider subsystem, the max loop delay is $2 * Tmult + Tmux + Tadd$. The function has the term x*di*2^-i. di is 1, -1, and 0. Therefore, we would have 2 multipliers, one mux, and one adder. the number of registers is 1 in this loop. The iteration bound = $\frac{2*Tmult+Tmux+Tadd1}{1}$

For the Newton Rhapson, similarly, from the Simulink, we also have one adder, two multipliers, and one mux. the max loop delay is $2 * Tmult + Tmux + Tadd$. The iteration bound is $\frac{2*Tmult+Tmux+Tadd1}{1}$

Problem 3: Quantization Effects and General Knowledge

a) 2's complement: (13, 4)

Sign-magnitude: (13, 4)

1's complement: (13, 4)

b) 2's complement has lower switching activity than sign-magnitude arithmetic: False

2's complement has higher switching than sign-magnitude

c) Each bit of quantization improves SNR by 3dB: False

Each bit of quantization should improve SNR by 6.02dB. $= 6.02 \cdot B + 10.8 - 20log_{10}\left(\frac{X_m}{\sigma_x}\right)$ lecture 6.22

d) Truncation works well in feedback systems: False

Truncation will not work well in feedback systems because it loses more information. It has one offset.

e) Pipelining can reduce energy by Shallower logic reduces required supply voltage

or increase throughput because the pipeline allows the data to continue entering the logic, and it does not need to wait the previous logic to finish to feed the next input data. Therefore, we can finish more instructions at the same time compared to non-pipelining structure under the same Vdd.

f) Inserting pipeline registers into recursive loops does not alter functionality. False

Lecture 8.45, Pipelining loops not possible. # registers in feedback loops must remain fixed. Changing the # delays in a loop alters functionality.

g) In leakage-limited scaling, voltage can decrease faster than transistor size. False

Voltage cannot decrease faster than transistor size. Otherwise, the voltage scaling is broken.

h) Dark silicon (utilization wall) occurs

the computational capabilities of chips are still increasing by 2.8x per process generation; but a utilization wall limits us to only 1.4x of this benefit, which resulting in large swaths of our silicon area remaining underclocked, or dark.