**In General**

It's a simple logic used in ADCs to represent floating point or decimal point numbers in conventional binary.

Say for e.g. If I want to represent 1 in binary, I will represent it as 1. i.e.; only 1 bit is needed to represent it.

But if I want to represent 0.5 in binary, I can’t do it.

So What we do is that, we will represent 1 as, n-bit binary say for e.g. as $1 \rightarrow 111$ (3-bit binary, i.e.; $n = 3$).

Now in our new system, 000 represents 0 and 111 represents 1. Any binary between 000 and 111 will represent numbers between 0 and 1 i.e.; various decimal point numbers.

Now, Resolution is defined as $1/(2^n - 1) = 1/7$, i.e.; the smallest number above 0 we can represent using our new system. Thus we can now represent decimal point numbers as:

- $000 \rightarrow 0$
- $001 \rightarrow 1/7 = 0.141$
- $010 \rightarrow 1/7 \times 2 = 0.282$
- So on ....and at last $111 \rightarrow 1$

If we increase ‘n’, higher resolution can be obtained and smaller numbers can be represented.

**In CORDIC**

To compute Sine and Cosine for various angles, our digital design should be able to process any angle between -360 to +360 degrees, including decimal point numbers like 102.45 degrees, 99.99 degrees etc. So we use here 32-bit-binary scaling system to represent angles from 0-360, with a very high resolution of $360/2^{32} - 1 = 0.00000008$ degrees! That’s the smallest angle we can process. Now using our 32-bit system, $111\ldots\ldots.1$ (32 ones) → 360, similary 45 degree is represented as 001000\ldots\ldots so on. In the cordic algorithm, we have to create a TAN table of angles from 45 , 26.6 ……TAN ARRAY in the code represents that table in 32-bit system. Since we have to process –ve angles too, we extend 32-bit to 33-bit. MSB now represents the sign bit.
or sign of the angle (1 → -ve number, 0 → +ve number). Rest 32 bits represents magnitude of the angle as usual. For e.g. If you want to input -45 degree as input angle, force it as 10010……………

This is the conventional sign-magnitude form of representing.

But please note that, if the sin or cos of the input angle is a -ve value, it is displayed in 2’s compliment form, NOT in conventional sign-magnitude form I said before.

No radian conversion is happening. Everything is in usual angle notations.

**In Mini CORDIC**

To compute Sine and Cosine for various angles, our digital design should be able to process any angle between -360 to +360 degrees, including decimal point numbers like 102.45 degrees, 99.99 degrees etc. So we use here 15-bit **binary scaling** system to represent angles from 0-360, with a resolution of $360/2^{15}-1 = 0.011$ degrees. That's the smallest angle we can process. Now using our 15-bit system, 111………1 (15 ones) → 360, similary 45 degree is represented as 001000…….. so on. In the cordic algorithm, we have to create a TAN table of angles from 45, 26.6 ……TAN ARRAY in the code represents that table in 15-bit system. Since we have to process -ve angles too, we extend 15-bit to 16-bit. MSB now represents the sign bit or sign of the angle (1 → -ve number, 0 → +ve number). Rest 15 bits represent magnitude of the angle as usual. Note that, for -ve numbers, we use **2's compliment system** to represent the magnitude of the -ve number in 15 bits. 16th bit will be ‘1’.

For examples and more info, please refer to the documentation of the IP-core, which comes with the RAR.

- **Mitu Raj**