Floating point random number distributions

Implement the Box-Muller transform to convert two random numbers to an approximation of a Gaussian random number distribution. The input are two random numbers (Provided by a test bench), A random numbers is produced as output.

$$a \leftarrow \sqrt{-2\ln(U_1)}$$
$$b \leftarrow 2\pi U_2$$
$$res \leftarrow a\sin(b)$$

U1,U2 are uniformly distributed random numbers ranging from 0 to 1. Zero and One will not appear as U1 or U2. (Removed from the data to make special cases easier)

note: In the literature, a cos(b) is also a random number. This is discarded in this project due to the fixed relation between a sin(b) and a cos(b).

The goal is to get 8 digits of precision in the results. Double precision floating point is required.

Implement this algorithm in verilog using double precision floating point

Since U1 ranges from 0 to 1, this calculation is performed with an interpolated table lookup. The table takes as input a fraction from 0 to 1. (Note that 0 and 1 are never present). The fraction is used to access the table. The table uses a 9 bit fraction. This requires denormalizing the original number. The bits not used form a delta from the 9 bit fraction. This resulting delta should be normalized to be a floating point number, and then a is calculated by:

$$a \leftarrow A\Delta^3 + B\Delta^2 + C\Delta + D$$

'b' is calculated using table lookup. This is a 10 bit table lookup producing A,B, and C as described above. U2 will be denormalized, and a delta created by renormalizing the remaining bits of the fraction. The table handles the factor of 2π The final result is then calculated by:

$$\sin(2\pi U_2) \leftarrow A\Delta^2 + B\Delta + C$$

The final result is formed by a*sin(b)

The design should run at 220 MHz, and produce an answer every pushed clock cycle.

The design has the following interface signals

Name	Bits	Dir	Comment
clk	1	In	The positive edge clock signal
rst	1	In	System reset
pushin	1	In	Push in for U1 and U2
U1	64	In	64 bit floating point number representing U1

Name	Bits	Dir	Comment
U2	64	In	64 bit floating point number representing U2
pushout	1	Out	Pushes a result out
Z	64	Out	64 bit floating point number result