This came as a reply to someone asking about advice tracing/trapping NaNs in ifort compiled executables on Intel Macs:

Try these options on both compile and link:

-O0 -g -traceback -fpe:0 -check all -fpstkchk

Some comments on these:
-g debug information. Note that -g does NOT IMPLY -O0

-O0 calls out no optimizations explicitly

-traceback will allow a traceback on exceptions

-fpe:0 Floating-point invalid, divide-by-zero, and overflow exceptions are enabled. If any such exceptions occur, execution is aborted. This option sets the -ftz (Linux and Mac OS) or /Qftz (Windows) option; therefore underflow results will be set to zero unless you explicitly specify -no-ftz (Linux and Mac OS) or /Qftz- (Windows).
On IA-32 or Intel(r) EM64T systems, underflow results from SSE instructions, as well as x87 instructions, will be set to zero. By contrast, option -ftz or /Qftz only sets SSE underflow results to zero.

-check all to catch things like array bounds, etc.

-fpstkchk catches conditions where the FP stack is not correct. Typically this is when a real function is called as if it were a subroutine, OR a subroutine is called as if it were a function (return values left of FP stack OR too much data is taken off the FP stack)

Also, you should read over the -fp-model compiler option. This allows varying degrees of IEEE compliance.

As far as the time to solution: it sounds a lot like you are generating denormalized values. Read the docs on -ftz. Denormal numbers can cause a severe performance penalty. -ftz can be used to force denormals to a zero representation. Handling denormals on Intel architecture can result in 1-2 orders of magnitude slowdown. Best to flush those to zero OR use a scaling factor or other mathematical techniques to prevent your fp numbers from getting into the realm of denormalization. I understand sometimes this is hard to prevent, due to the algorithm and nature of the problem under study.