A Hardware Random Number Generator for Transputer Systems

Leslie Smith and Frank Kelly
Department of Computing Science
University of Stirling
Stirling, Scotland, UK

Abstract

A hardware random number generator based round a noise generating diode is presented. Although fast, the spread of numbers produced is not as smooth as hoped. Some suggestions are made for its improvement.

Introduction

Research in Neural Networks at the University of Stirling has frequently required large numbers of random numbers for generating noise for signals and stochastic pseudo-neurons. In simulations of neural nets processing time-varying signals, many thousands of such numbers may be required each second, making fast random number generation important. There is no requirement that precisely the same sequence of random numbers be repeated, so that we need not be restricted to seed-based random number generation. Hardware random number generation was perceived as a relatively straightforward task, something not really borne out by later experience.

The Generator

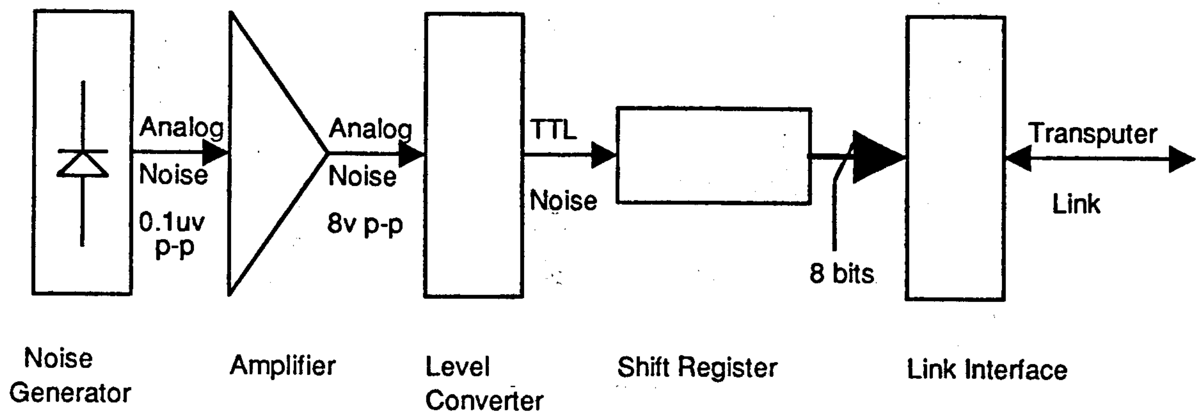

Figure 1: Noise generator block diagram.

biassed noise generating diode (NC202). This is guaranteed to produce white noise from 20hz to 25Mhz. This very small signal (about 0.1 μv) was then amplified by a four stage amplifier based on the Signetics NE5539 wideband operational amplifier. The resulting analogue signal is converted to TTL levels by 1 channel of a DS26LS32 line receiver, followed by a Schmitt trigger. The resultant 'digital' noise signal is then fed to an 8-bit shift register clocked at 10 Mhz. This gives a theoretical limit to the time for number generation of 0.8 μs. This part of the circuit runs continuously.

The shift register is interfaced to a transputer link by a C011 link interface chip, running as a parallel interface. It is set up so that the receipt of a byte on the link causes the generator to send out a stream of random bytes, stopping only when another byte is received on the link. When the C011 is ready to send out a new random byte, it waits for the shift register to receive a new set of 8 bits, and then causes the output of the serial to parallel convertor to be latched. The data is then sent out by the C011. Running the link at 10Mhz, it has been found that the remote transputer receives a new random byte every 2.74 μs. A more detailed circuit description is available from the authors.

Testing the Generator

Three tests have been tried on the generator: frequency checks on single number generation, simple frequency of 1's and 0's , and frequency of pairs of numbers rising, falling, or staying the same. Additionally, the frequency of recurrence of single bytes, of pairs of bytes, and of sequences of three bytes have been checked.

The generator performs worst on frequency of single number generation. Experimentally, it has been found that certain numbers, notably those with strings of 1's, occur more frequently. This sort of systematic error results in very high values of χ^2 when large numbers of random bytes are produced. The actual effect is most pronounced on the number 255, which occurs about 1.8 times as often as it should; the results for other numbers are much less pronounced, with 127 being produced 1.2 times as often, though values for 63 and 191 vary widely.

The results for simple frequency of 1's and 0's are better; the system is slightly biassed towards 1's. On a sample of 100,000 bytes (i.e. 800,000 bits) the χ^2 value produced varies between 3.5 and 11.2, when it should be between 0.00016 and 6.635. Again, this value rises on larger samples.

For the test of pairs of numbers rising or falling, or staying the same, the results are better; using 2,560,000 numbers, values of χ^2 of between 0.35 and 8.49 were found, when [1] suggests 0.0201 to 9.21. We also looked at recurrence of numbers; these were found to occur at about the correct frequency for single bytes (i.e. about 1 per 2^8), for double bytes (about 1 per 2^{16}) and triple bytes (about 1 per 2^{24}).

Comments and Conclusions

Many difficulties were encountered in the construction of the equipment. Largely, these came from an underestimation of what was involved in the mixing of sensitive analogue and standard digital circuitry in one box. Eventually, the analogue circuitry was completely enclosed, and used an entirely independent power supply. There were also problems with oscillation due to lack of experience in non-digital construction. Before trying the noise-generating diode, we tried a number of ordinary and zener diodes. The magnitude of the noise signal generated by these was an order of magnitude smaller, making the problems of amplification much greater.

The results are a little disappointing, although good enough for the application itself. We believe the problems causing the systematic lack of randomness come from two sources: the level conversion, and the bandwidth of the noise signal. The level conversion circuit is originally a balanced line receiver: it is perhaps not as accurate in its detection of a 0v signal as required. This could lead to the slight preponderance of 1's over 0's. The noise generating diode is supposed to give out noise evenly over a very large band: however, we believe that the noise falls away a little towards the high frequency end of the spectrum, leading to numbers which contain strings of the same digit. We do find numbers with strings of 1's, but do not find numbers with strings of 0's. Further, the imbalance between 0's and 1's is very small, so we remain rather uncertain of the precise nature of this problem.

Since the results were good enough for the application, and since the primary aim of the project is in neural net simulation, not random number generation, we have not pursued the matter further. Were we to do so, we would try a higher bandwith noise generating diode, plus some filtering to achieve an optimal spread of noise frequencies (though we note that we are not certain of what that should be).

References

[1] D.E. Knuth *Seminumerical Algorithms*, 2nd Edition, Addison-Wesley, 1981.