

## Better use of random number generator bits?

---

*Source:* <http://sci.tech-archive.net/Archive/sci.math/2006-03/msg00949.html>

---

- *From:* "DAC" <Mister@xxxxxxxxxx>
  - *Date:* Mon, 6 Mar 2006 17:28:12 +1000
- 

If I have a random number generator creating random binary digits (0 or 1), and I want to construct a random integer between 0 and say 40, it appears the accepted no biased way is to take 6 binary digits which will represent numbers 0 to  $2^6-1$  (63).

So to obtain a number between 0 and 40 I must ignore any randomly generated numbers above 40. This means that on average  $24/64 = 37.5\%$  of the time I will have to get another number (6 more bits), and 37.5% of those times another 6, such that on average we will have to obtain 10.5 bits to make a number between 0 and 40 because the random bits are base 2 and not base 40.

So what I am asking is, if the number is greater than 40 what can I do with the bits I have already so that instead of getting another 6 bits I can get less than this, or reuse these bits, with out introducing any bias what-so-ever, if this is possible at all.

One Idea I have, but havent done any work on yet, is if both of the first two numbers are over 40, you can add them together and subtract 80 to get a unbiased random number between 0 and 47 and so I can again check if this number is less than 40, effectively saving need for 6 more bits.

This is important for us because for example to shuffle a deck of cards we are taking 51 random numbers (between 0 and 51 down to between 0 and 1). This on average is a total of about 2880 bits to shuffle a deck (of course there is no maximum on this number and I have encounter one situation that took over 13,000 bits), where you should really only need  $\log_2(52!) = 226$  bits to specify a specific shuffle. So at the moment we are using on average 10 times as many bits as we really need – and quantum generator modules are expensive and soon we will need the numbers at a rate faster than we can produce them.

Also another way we know is to produce a 226 bit random number but the math to do the 226 bit division/modulus etc to produce the actual card sequence is to slow on current computers to be fast enough for our implementation, and obviously we cannot have a look up table of size  $52!$ . We are working on embedded FPGA hardware to do 256 bit math but would be great if any of you math experts here could help us use less bits in generating a random number working with standard integer sizes (32bit).

Better use of random number generator bits?