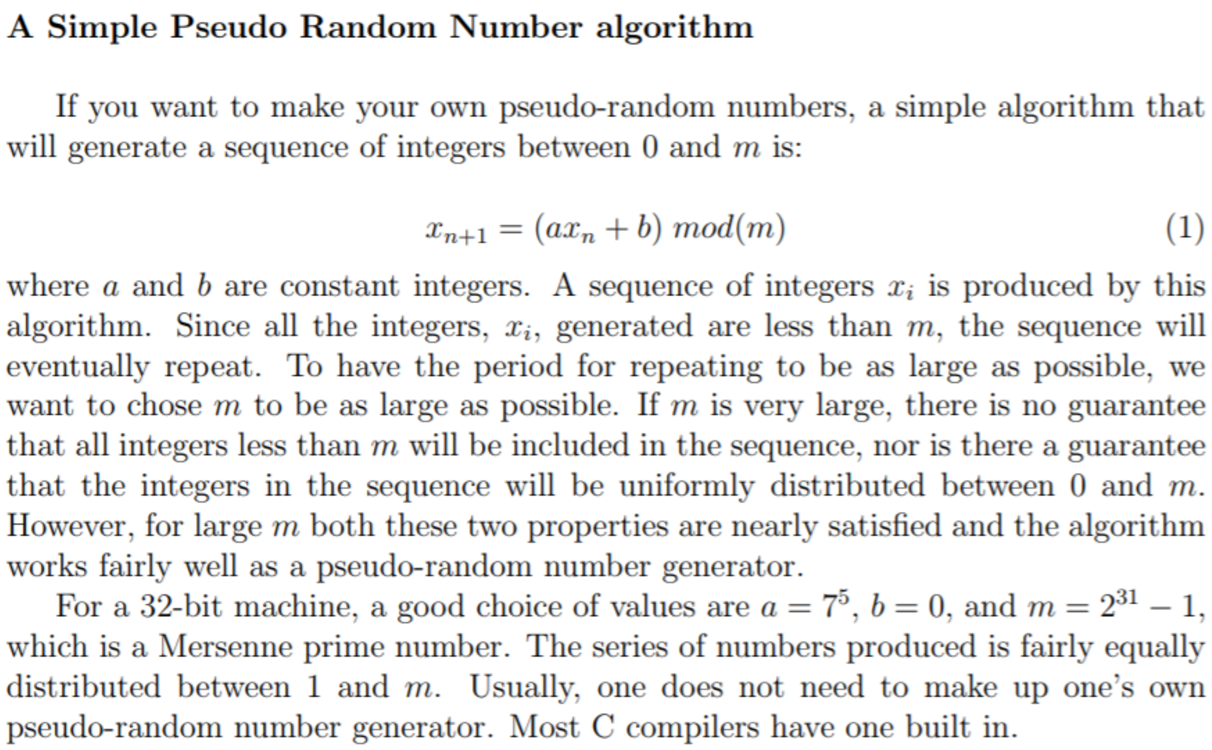
Ideas on creating random numbers

If your FPGA system has a truly asynchronous event, like a human pushing a button, you can create a new random number every time the human presses the button. Simply create a counter continually counting at the system clock rate (100 MHz), and every time the button is pressed, save the lower bits to a register. Note: a 16-bit counter counting at a 100 MHz rate rolls over at around 655 micro-seconds, which is much faster than a human can press a button.

If you need more than one random number

If your system needs several random numbers (like to initialize an array of values for a game), you can use the single random number above as a seed to generate several pseudo-random numbers using an algorithm such as:

From https://www.cpp.edu/~pbsiegel/phy499w16/randnum.pdf



A Simple Pseudo Random Number algorithm

If you want to make your own pseudo-random numbers, a simple algorithm that will generate a sequence of integers between 0 and m is:

$$x_{n+1} = (ax_n + b)\ mod(m) \tag{1}$$

where a and b are constant integers. A sequence of integers x_i is produced by this algorithm. Since all the integers, x_i, generated are less than m, the sequence will eventually repeat. To have the period for repeating to be as large as possible, we want to chose m to be as large as possible. If m is very large, there is no guarantee that all integers less than m will be included in the sequence, nor is there a guarantee that the integers in the sequence will be uniformly distributed between 0 and m. However, for large m both these two properties are nearly satisfied and the algorithm works fairly well as a pseudo-random number generator.

For a 32-bit machine, a good choice of values are $a = 7^5$, $b = 0$, and $m = 2^{31} - 1$, which is a Mersenne prime number. The series of numbers produced is fairly equally distributed between 1 and m. Usually, one does not need to make up one's own pseudo-random number generator. Most C compilers have one built in.

Linear Feedback Shift Register

Here is a Psuedo-Random Number generator (by Nate Bean), using a Linear Feedback Shift Register

```
entity randomNum is
              Port ( clk: in  STD_LOGIC;
                     reset_n : in  STD_LOGIC;
                     rand : out std_logic_vector(7 downto 0)
                     );
end randomNum;

architecture Behavioral of randomNum is

signal curNum, nextNum: std_logic_vector(7 downto 0);
signal feedback: std_logic;

begin

process (clk)
   begin
      if (rising_edge(clk)) then
         if reset_n = '0' then
            curNum <= "00000001";
         else
            curNum <= nextNum;
         end if;
      end if;
end process;


feedback <= curNum(4) XOR curNum(3) XOR curNum(2) XOR curNum(0);
nextNum <= feedback & curNum(7 downto 1);

rand <= curNum;

end Behavioral;
```