

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  use ieee.std_logic_unsigned.all;
5  library work;
6  use work.rand_pack.all;
7
8  -- pseudo random binary sequence generator
9  -- based on shift register that provides shift left
10 -- with serial input that calculated by polynom
11 -- (generic constant G_POLYNOM) of bit position indexes
12 -- of shift register bits that should be XORed togeder
13
14 entity grandom is
15     generic (
16         -- G_SEED      : integer -- optionally initial value of shift register
17         G_POLYNOM     : int_arr_type := (7,5,4,3);
18         G_DATA_WIDTH  : integer     := 8);
19     port
20     (
21         clk  : in  std_logic;
22         rst  : in  std_logic;
23         en   : in  std_logic;
24         dout : out std_logic_vector(G_DATA_WIDTH-1 downto 0)
25     );
26 end entity grandom;
27
28 architecture arc_grandom of grandom is
29     signal buff : std_logic_vector(dout'range);
30 begin
31     random_engine: process (clk, rst) is
32         variable reduced_or : std_logic;
33         variable polynomials: std_logic;
34         variable sin        : std_logic;
35     begin -- process random engine
36         if (rst = C_INIT) then
37             -- may be different from 0 initial value
38             -- defined by optionally generic constants G_SEED
39             -- buff<= conv std logic vector(G_SEED,buff'length);
40             buff<=(others => '0');
41         elsif rising_edge(clk) then
42             -- reduced nor gate prevents continuous 0 sequence
43             reduced_or:='0';
44             for i in buff'range loop
45                 reduced_or:=reduced_or or buff(i);
46             end loop;
47             -- next serial in value
48             polynomials:='0';
49             for j in G_POLYNOM'range loop
50                 polynomials:=polynomials xor buff(G_POLYNOM(j));
51             end loop;
52             -- serial input to shift register (shift left)
53             sin:=polynomials or (not reduced_or);
54             buff<=buff((buff'high-1) downto buff'low) & sin;
55         end if;
56     end process random_engine;
57
58     dout <= buff;
59
60 end architecture arc_grandom;

```