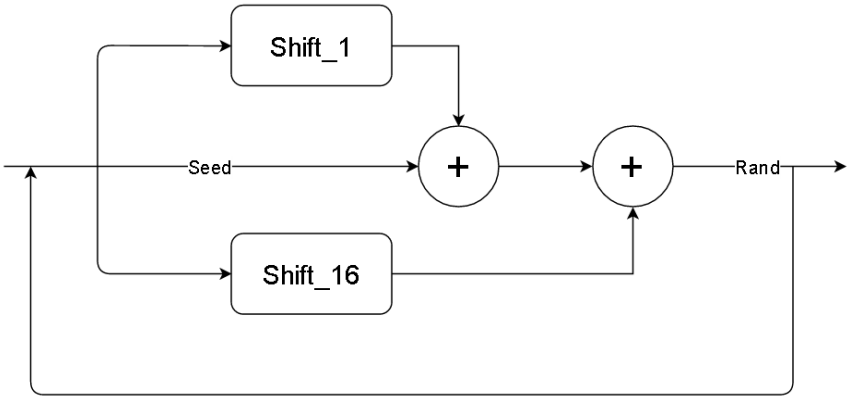
ECSE 323 – Digital System Design
Random Number Generator
g39_RANDU

The RANDU circuit is a random number generator that uses the Linear Congruential Generator algorithm. More specifically, the IBM RANDU version of the algorithm. Linear Congruential Generator method takes the form: $R = mod(a * SEED + b, c)$ where in the IBM RANDU $a = 65539$, $b = 0$, and $c = 2^{31}$.
This works by feeding the circuit with an input SEED once, producing an output R, then replacing SEED with R to recursively generate more outputs.



SEED: 32-bit input
RAND: 32-bit output

Algorithm:

Since $b = 0$, the equation becomes $R = mod(65539 * SEED, 2^{31})$.

I. Multiplication:

To multiply $65539 * SEED$, we can use the fact that $65539_{10} = 10000000000000011_2$.
So $65539 * SEED = Shift\_left(SEED, 16) + Shift\_left(SEED, 1) + Shift\_left(SEED, 0)$

II. Modulo:

mod(X, Y) = X % Y where $Y = 2^n$ is basically setting all the bits of X beyond $(n-1)^{th}$ bit to 0. In other words, mod(X, Y) = X(n-1 DOWNTO 0)

VHDL Implementation:

While the algorithm could be implemented using the LPM library's adders and one of the shifting operations, we chose to implement it using ieee.std_logic_unsigned library.

Addition:

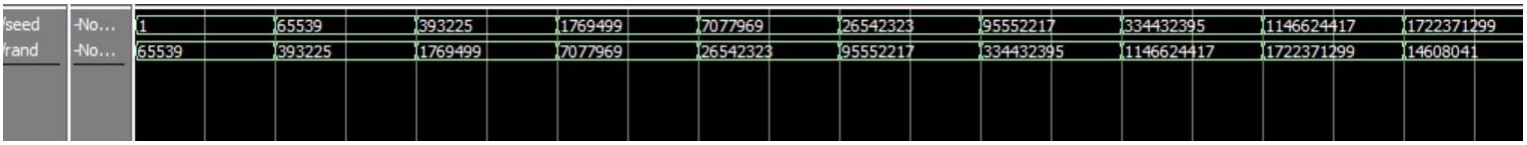
We can simply use "+" sign for addition and the unsigned library will take care of unbalanced bits by padding with zeros.

Shifting:

The "&" sign is used for concatenation. For example, Shift_left(11, 2) = 11 & "00" = 1100
This mimics the left shifting process.

Testing:

To ensure the circuit works properly, we first tested the algorithm on paper for a couple of cases. Then we carried out a comprehensive simulation of the random number generator circuit. This was done with a while seed < "10000000000000000000000000000000" since the R ϵ [0, 2^{31}-1] Below is test cases and simulation plots:



This was done by feeding the circuit with an initial SEED of "00000000000000000000000000000001" as explained in the lab specification file. Then to generate more outputs, we replace the SEED with R.

The recursive relationships is $R_i = mode(a*R_{i-1}, c)$ where R_0 = initial SEED above.

In order to check if the outputs of the simulation are correct, we've wrote a java program that simulates the circuit's function:

```
double a, c, seed, rand;
a = 65539;
c = Math.pow(2, 31);

int iterations = 5;
double[] rands = new double[iterations];

seed = 1;
rand = 0;


int i = 0;
do {
  rand = a*seed % c;
  rands[i] = rand;
  System.out.println(rand);
  if(rand >= c-1) {
    System.out.print("MAX!");
    return;
  }

  seed = rand;
  i++;
} while(i < iterations);

i = iterations-1;
do {
  rand = (rands[i] - (6*rands[i-1]) + (9*rands[i-2])) % c;
  System.out.println(Math.abs(rand));
  i--;
} while(i > 1);
```

The outputs of the programs match the ones seen in the simulation.