# XOR Shift

## Random Number Generator

P. Ram Anudeep P. Pavan

8th March, 2019

# Random Number Generator

- A random number generator (RNG) is a device that generates a sequence of numbers or symbols that cannot be reasonably predicted better than by a random chance.

# Random Number Generator

- A random number generator (RNG) is a device that generates a sequence of numbers or symbols that cannot be reasonably predicted better than by a random chance.
- Random number generators have applications in gambling, statistical sampling, computer simulation, cryptography, completely randomized design, and other areas where producing an unpredictable result is desirable.

# Random Number Generator

- Various applications of randomness have led to the development of several different methods for generating random data, of which some have existed since ancient times, among whose ranks are well-known "classic" examples, including the rolling of dice, coin flipping, the shuffling of playing cards

# Random Number Generator

- Various applications of randomness have led to the development of several different methods for generating random data, of which some have existed since ancient times, among whose ranks are well-known "classic" examples, including the rolling of dice, coin flipping, the shuffling of playing cards
- Because of the mechanical nature of these techniques, generating large numbers of sufficiently random numbers (important in statistics) required a lot of work and/or time.

# XOR Shift

- Xorshift random number generators are a class of pseudorandom number generators.

# XOR Shift

- Xorshift random number generators are a class of pseudorandom number generators.
- They generate the next number in their sequence by repeatedly taking the exclusive or of a number with a bit-shifted version of itself.

# XOR Shift

- Xorshift random number generators are a class of pseudorandom number generators.
- They generate the next number in their sequence by repeatedly taking the exclusive or of a number with a bit-shifted version of itself.
- Xorshift generators are among the fastest non-cryptographically-secure random number generators, requiring very small code and state.

# XOR Shift

- Although they do not pass every statistical test without further refinement, this weakness is well-known and easily amended by combining them with a non-linear function, resulting e.g. in a xorshift+ or xorshift* generator.

# XOR Shift*

- A xorshift* generator takes a xorshift generator and applies an invertible multiplication (modulo the word size) to its output as a non-linear transformation

# XOR Shift+

- Instead of multiplication ,we can use addition as a fast non linear transformation.It adds two consecutive outputs of an underlying xorshift generator based on 32-bit shifts

# Input

- The input to the XOR shift algorithm is a non zero number called as seed.

# Input

- The input to the XOR shift algorithm is a non zero number called as seed.
- The series of random numbers are generated based on the given seed(seeding).

# Input

- The input to the XOR shift algorithm is a non zero number called as seed.
- The series of random numbers are generated based on the given seed(seeding).
- The first random number acts as seed to the next Xor shift implementation.

# Input

- The input to the XOR shift algorithm is a non zero number called as seed.
- The series of random numbers are generated based on the given seed(seeding).
- The first random number acts as seed to the next Xor shift implementation.
- In our project we have done seeding in the code itself.

# Output

- The output generated by icoboard is communicated to Arduino via serial communication.

# Output

- The output generated by icoboard is communicated to Arduino via serial communication.
- Range of our output is 0-255.We generate 64 bit random number.We can define range based on the number of bits of output we are taking.We have taken first 8 bit from 64 bits and given to arduino.

# Output

- The output generated by icoboard is communicated to Arduino via serial communication.
- Range of our output is 0-255.We generate 64 bit random number.We can define range based on the number of bits of output we are taking.We have taken first 8 bit from 64 bits and given to arduino.
- The sequence of random numbers read by arduino is displayed through serial monitor and we displayed the random number with LCD display.

# Algorithm

- input clk_in :
  input for the xor_shift module is coming from clk_in

# Algorithm

- input clk_in :
  input for the xor_shift module is coming from clk_in
- output reg [15:0] data_out :
  data_out is the random number output of the xor_shift module

# Algorithm

- input clk_in :
  input for the xor_shift module is coming from clk_in
- output reg [15:0] data_out :
  data_out is the random number output of the xor_shift module
- state0 = 64'd100000 & state1 = 64'd100 :
  the input seed for xor_shift(seeding)

# Algorithm

```
module
xorshift(
          input clk_in,
          output reg [7:0] data_out,
          output reg output_ready
        );

        reg [63:0] state0, state1;
        reg [63:0] s, t, temp;
        reg clk;
        reg [26:0] clk_delay;
        integer state;

        initial begin
          clk = 0;
          clk_delay = 0;
          state0 = 64'd100000;
          state1 = 64'd100;
          state = 0;
        end
```

Our module has one output 8bit reg for 8 bit random number. state0 and state1 are input seed. clk_delay reg is used for delay purpose. We are initializing our seed value to 100000.

# Algorithm

```
always @ (posedge clk_in) begin
  clk_delay = clk_delay+1;
  if(clk_delay==27'd10)
  begin
    clk_delay = 0;
    clk = ~clk;
  end
end
```

Delay part.

# Algorithm

```
always @ ( posedge clk ) begin
  if(state==0)
  begin
    t <= state0;
    s <= state1;
    state <= 1;
    output_ready <= 0;
  end
  else if(state==1)
  begin
  state0 <= s;
    t <= t^(t<<23);
    state <= 2;
  end
  else if(state==2)
  begin
    t <= t^(t>>17);
    state <= 3;
  end
```

Main xorshift algorithm. We are doing in 4 states as these are non blocking assignments.(All statements executes at a time). In the first state(state==1) t is assigned as t xor(bit shifted version of t) i.e,23 bit shift to left. In the second state(state==2) t is assigned as t xor(bit shifted version of t) i.e,17 bit shift to right.

# Algorithm

```
    else if(state==3)
    begin
      t <= t^(s^(s>>26));
      state <= 4;
    end
    else if(state == 4)
    begin
      state1 <= t;
      temp = t+s;
      data_out <= temp[15:0];
      output_ready <= 1;
      state <= 0;
    end
  end

endmodule
```

In the third state(state==3) t is assigned as t xor(s xor(26 bits shifted to right of s )). In the fourth state(state==4) state1=t(output as input in next clock cycle). We are taking first 8 bits of 64 bit random number $data_out = temp[7 : 0]; heretempis64bitrandomnumber.$

# Implementation

- The random number generator will be implemented as state machine.The random number generator will be initially seeded.The output is shown using arduino and LCD display.The random number is generated using the following algorithms. 1.XOR SHIFT 2.XOR SHIFT* 3.XOR SHIFT+

# References

- https://en.wikipedia.org/wiki/Xorshift

# References


- https://en.wikipedia.org/wiki/Xorshift
- http://icoboard.org/

# References


- https://en.wikipedia.org/wiki/Xorshift
- http://icoboard.org/
- https://www.instructables.com/id/How-to-use-an-LCD-displays-Arduino-Tutorial/