Bangladesh University of Engineering and Technology
Electrical & Electronic Engineering

Course No: EEE 458
Course Name: VLSI Circuits and Design II Laboratory

Report on

True Random Number Generator

Date of Submission: 28th February, 2022

Submitted by
Group No – 27

Name & ID:
1606140 – Shakil Ahmed
1606176 – Ishrat Jahan Moon

Table of Contents

Objective

In this project we designed and verified a true random number generator. Random numbers are numbers that occur in a sequence such that two conditions are met:

- The values are uniformly distributed over a defined interval or set, and
- It is impossible to predict future values based on past or present ones.

Random number generation is a process by which, often by means of a random number generator (RNG), a sequence of random numbers or symbols is generated. This means that the particular outcome sequence will contain some patterns detectable in hindsight but unpredictable to foresight. Security protocols and encryption technologies are built on the foundation of random number generators. The **purpose** of this project **is to create a customizable solution using HDL** in situations when physical random number generator sources, such as electrical noise, are unavailable or cumbersome. Any high-speed cryptography technique and encryption/decryption system can benefit from this architecture.

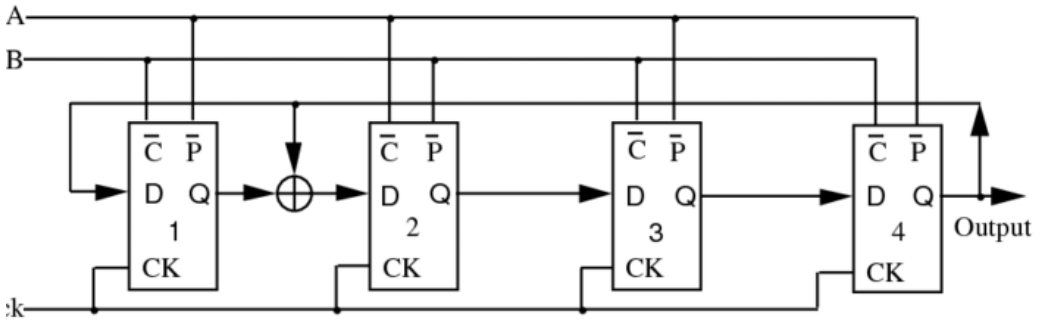
Theory

In computing, a linear-feedback shift register (LFSR) is a shift register whose input bit is a linear function of its previous state.

The most commonly used linear function of single bits is exclusive-or (XOR). Thus, an LFSR is most often a shift register whose input bit is driven by the XOR of some bits of the overall shift register value.

The initial value of the LFSR is called the seed, and because the operation of the register is deterministic, the stream of values produced by the register is completely determined by its current (or previous) state. Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle. However, an LFSR with a well-chosen feedback function can produce a sequence of bits that appears random and has a very long cycle.

Applications of LFSRs include generating pseudo-random numbers, pseudo-noise sequences, fast digital counters, and whitening sequences. Both hardware and software implementations of LFSRs are common.



Clock	Q_1	Q_2	Q_3	Q_4
0	1	0	1	1
1	1	0	0	1
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	1	1	0	0
7	0	1	1	0

Clock	Q_1	Q_2	Q_3	Q_4
8	0	0	1	1
9	1	1	0	1
10	1	0	1	0
11	0	1	0	1
12	1	1	1	0
13	0	1	1	1
14	1	1	1	1
15	1	0	1	1

Figure 1: Linear-feedback shift register (4 bit)

We can see that the sequence is repeating after every 16 cycles. So, its predictable. But if the LFSR is long enough (i.e., 64 bit) it will repeat same sequence after a very long time. Predicting the next output bit is equivalent to knowing the feedback pattern of the LFSR and the states of all LFSR flip-flops. Mathematicians (Berlekamp-Massey) found that:

"Given an N-bit LFSR with unknown feedback pattern, then only 2N bits are needed to predict bit 2N + 1"

So let's say we have and 8-bit LFSR, then we need only 16 bits of the RNG stream before it becomes predictable. LFSR are unsuited for everything that should be unpredictable. So we have two solutions:

1. Use a maximal length $P(x) = x^N + \ldots + 1 \; with \; N >>> (E.g. 65{,}536)$ Very expensive to make! 64K flip-flops.
2. Non -linear Combination Generator.

So, we use arbitrary mixer so that the random input sequence can be ensured and X-OR network to implement this LFSR characteristics. And in between we used a ISU network to bring this Non -linear Combination Generator.

Block Diagram

Logic Block Implementation of our proposed architecture:

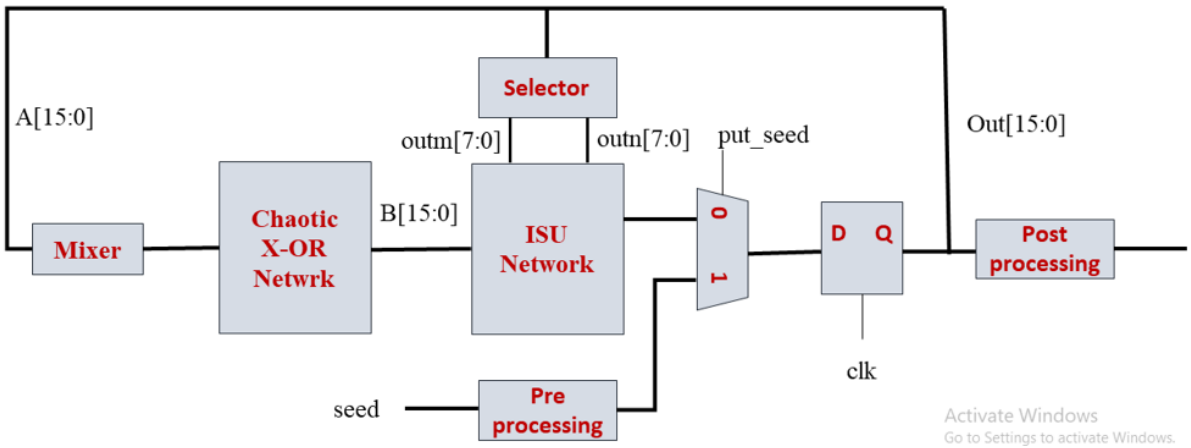


Figure 2: Logic Block Implementation of Random Number Generator

In our proposed architecture, Seed initiates the generation process and put_seed acts as enable pin. When put_seed is 1,16bit seed passes to the mixer through the feedback network. In the mixer these bits combined with multiple zeros generate total 16,16bit numbers. These numbers enter the xor network which performs xor operation on the bits of each number and generates 16 bits. These bits along with the flipflop output produces another 16bit number in ISU network which passes through the multiplexer and flipflop when put_seed is kept 0 and the loop continues. Thus due to the feedback from output to the mixer circuit the final output becomes random.

Algorithm

Our architecture consists of

- Seed network
- Selector circuit
- Mixer
- X-OR network
- Inverter Swap Unit (ISU) Network
- Post Processing unit

Seed Network:

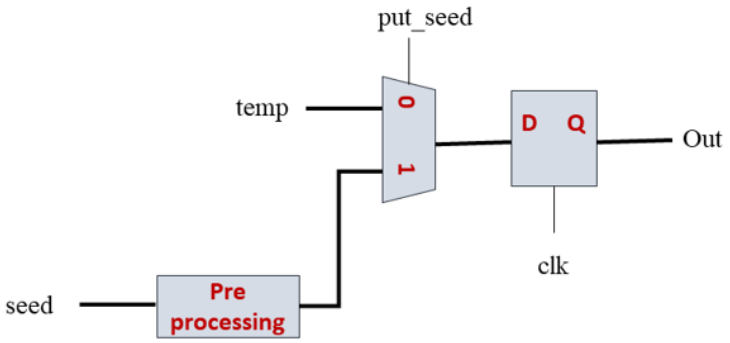


Figure 3: Seed Network Block diagram

In Seed Network there are a pre-processing unit, a two to one multiplexer and a flipflop. This networks takes input and after some processing delivers a random number output. 16bit input is called seed which can be customized by the user. Select pin of the Multiplexer is named put_seed which acts as enable pin of the architecture. When we want to change the seed, put_seed must be kept 1 otherwise it is kept 0 so that a 16bit number(temp) can pass through the multiplexer and enter the flipflip. Temp is generated in the ISU network. The flipflop is positive edge triggered and works as memory segment of our designed architecture. The output from the flipflop is the desired random number.

Mixer:

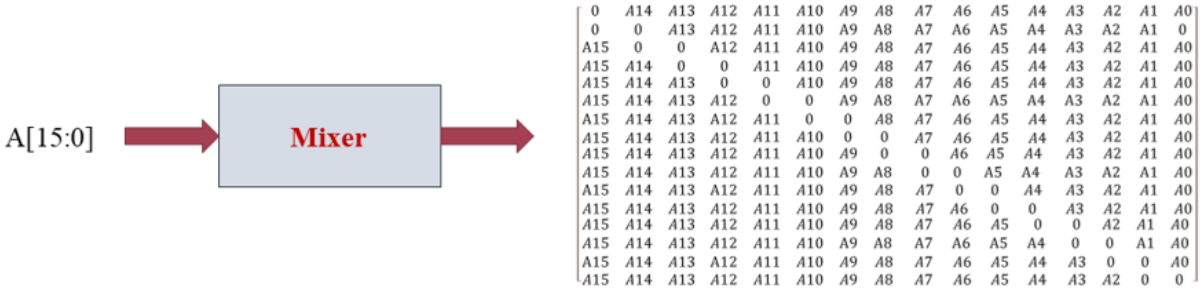


Figure 4: Mixer

Output from the flipflop is fed to the mixer circuit. It takes 16bits as input and mixes some zeroes with them and produces 16 different combination i.e. generates total 16 numbers of 16bit as shown in Figure 3.

X-OR Network:

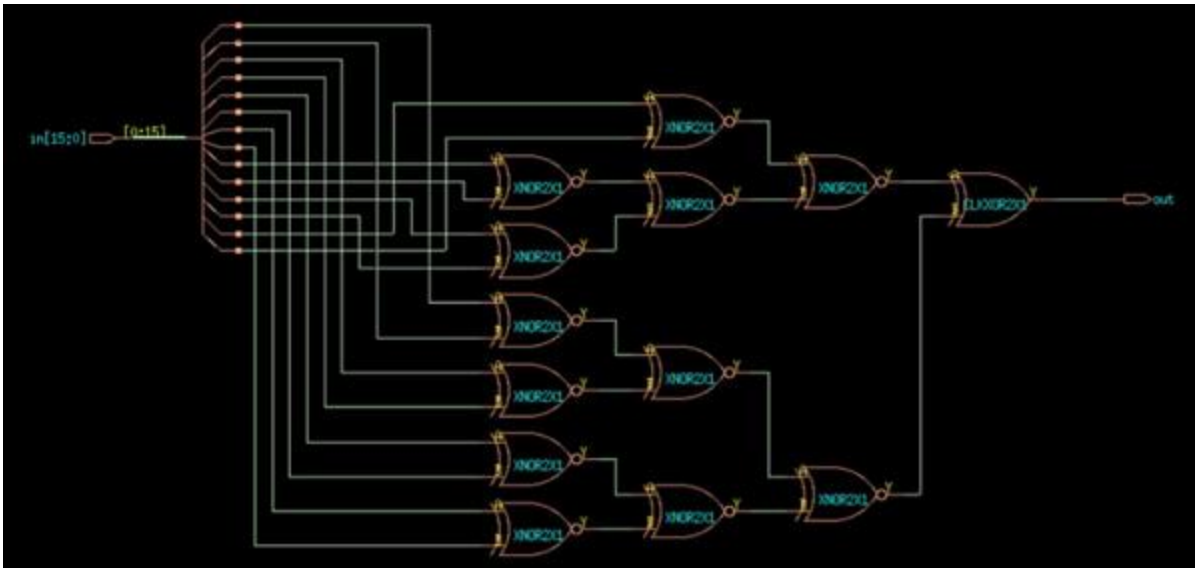


Figure 5: Chaotic X-OR Network

X-OR network consist of 16 X-OR gates. Each gate perform xor operation on 16 bits at a time and produces 0 or 1. Odd number of 1's in the bitstream results in 1 while even number of 1 generates 0. Output from the mixer circuit enters the xor network. Therefore 16, 16bit numbers is converted to a single 16bit number in the xor network.

Inverter Swap Unit (ISU) Network:

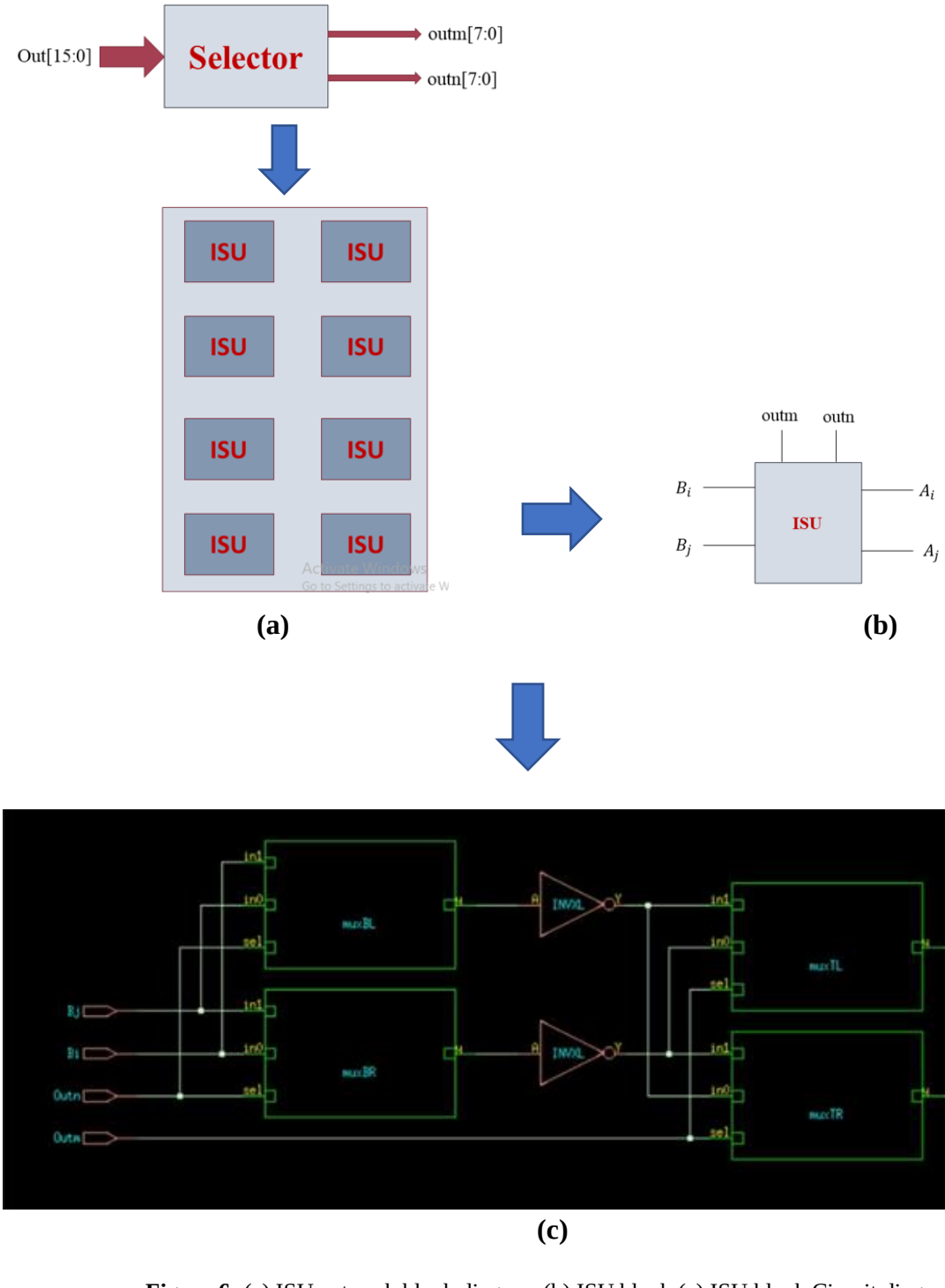


Figure 6: (a) ISU network block diagram (b) ISU block (c) ISU block Circuit diagram

Inverter Swap Unit Network consists of 8 ISU blocks. Each block performs inverting and swap operation on 4 single bit input and provides 2single bit output. Figure 5 shows ISU network block diagram and it's unit circuit. Selector divides a 16bit bitstream into two stream. Each bit stream

now consists of eight bits. These bits work as select pin of each ISU block. The operation of ISU block is demonstrated through Truth Table in Figure 6.

Input				Output	
outm	outn	B_j	B_i	A_j	A_i
0	0	0	0	1	1
0	0	0	1	1	0
0	0	1	0	0	1
0	0	1	1	0	0
0	1	0	0	1	1
0	1	0	1	0	1
0	1	1	0	1	0
0	1	1	1	0	0
1	0	0	0	1	1
1	0	0	1	0	1
1	0	1	0	1	0
1	0	1	1	0	0
1	1	0	0	1	1
1	1	0	1	1	0
1	1	1	0	0	1
1	1	1	1	0	0

Table 1: Truth Table for ISU unit

Code

```
`timescale 1ns / 1ps

module trng (Out, A, B, temp, put_seed, seed, clk);

output [15:0] B, temp;
input put_seed, clk;
   input [15:0] seed;
output reg [15:0] Out, A;
   wire [15:0] seed;


always@(posedge clk)begin
```

```
        if(put_seed) begin
        Out <= seed;
        A <= seed;
        end
        else begin
        Out <= temp;
        A <=  temp;
        end
end



//Chaotic XOR Network
xor16_1 xor16(B[15],{1'b0,A[14:0]});
xor16_1 xor15(B[14],{2'b00,A[13:1],1'b0});
xor16_1 xor14(B[13],{A[15],2'b00,A[12:0]});
xor16_1 xor13(B[12],{A[15:14],2'b00,A[11:0]});
xor16_1 xor12(B[11],{A[15:13],2'b00,A[10:0]});
xor16_1 xor11(B[10],{A[15:12],2'b00,A[9:0]});
xor16_1 xor10(B[9],{A[15:11],2'b00,A[8:0]});
xor16_1 xor9(B[8],{A[15:10],2'b00,A[7:0]});
xor16_1 xor8(B[7],{A[15:09],2'b00,A[6:0]});
xor16_1 xor7(B[6],{A[15:08],2'b00,A[5:0]});
xor16_1 xor6(B[5],{A[15:07],2'b00,A[4:0]});
xor16_1 xor5(B[4],{A[15:06],2'b00,A[3:0]});
xor16_1 xor4(B[3],{A[15:05],2'b00,A[2:0]});
xor16_1 xor3(B[2],{A[15:04],2'b00,A[1:0]});
xor16_1 xor2(B[1],{A[15:03],2'b00,A[0]});
xor16_1 xor1(B[0],{A[15:02],2'b00});

//ISU Network
ISU ISU1_2(temp[0],temp[1],B[0],B[1],Out[0],Out[1]);
ISU ISU3_4(temp[2],temp[3],B[2],B[3],Out[2],Out[3]);
ISU ISU5_6(temp[4],temp[5],B[4],B[5],Out[4],Out[5]);
ISU ISU7_8(temp[6],temp[7],B[6],B[7],Out[6],Out[7]);
ISU ISU9_10(temp[8],temp[9],B[8],B[9],Out[8],Out[9]);
ISU ISU11_12(temp[10],temp[11],B[10],B[11],Out[10],Out[11]);
ISU ISU13_14(temp[12],temp[13],B[12],B[13],Out[12],Out[13]);
ISU ISU15_16(temp[14],temp[15],B[14],B[15],Out[14],Out[15]);

endmodule
```

```
module xor16_1(output out,
                input wire [15:0] in);

xor(out, in[15], in[14], in[13], in[12], in[11], in[10], in[9], in[8], in[7], in[6], in[5], in[4], in[3],
in[2], in[1], in[0]);

endmodule


module mux(y, in1, in0, sel);

output reg y;
input in1, in0, sel;

always@(in1, in0, sel)
        case(sel)
                1'b0: y=in0;
                1'b1: y=in1;
        endcase

endmodule


module ISU(Ai, Aj, Bi, Bj, Outm, Outn);

    output Ai, Aj;
    input Bi, Bj, Outm, Outn;
    wire x,y;

mux muxBL(x,Bi,Bj,Outn);
    mux muxBR(y,Bj,Bi,Outn);
    mux muxTL(Ai,~x,~y,Outm);
    mux muxTR(Aj,~y,~x,Outm);

endmodule
```

Synthesis

.synth file:

```
// Generated by Cadence Genus(TM) Synthesis Solution 16.13-s036_1
```

```
// Generated on: Feb 21 2022 14:49:37 BDT (Feb 21 2022 08:49:37 UTC)

// Verification Directory fv/trng

module mux(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_1(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_2(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_3(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module ISU(Ai, Aj, Bi, Bj, Outm, Outn);
  input Bi, Bj, Outm, Outn;
  output Ai, Aj;
  wire Bi, Bj, Outm, Outn;
  wire Ai, Aj;
  wire n_0, n_1, x, y;
  mux muxBL(x, Bi, Bj, Outn);
```

```
  mux_1 muxBR(y, Bj, Bi, Outn);
  mux_2 muxTL(Ai, n_0, n_1, Outm);
  mux_3 muxTR(Aj, n_1, n_0, Outm);
  INVXL g4(.A (y), .Y (n_1));
  INVXL g3(.A (x), .Y (n_0));
endmodule

module mux_4(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_5(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_6(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_7(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module ISU_1(Ai, Aj, Bi, Bj, Outm, Outn);
  input Bi, Bj, Outm, Outn;
  output Ai, Aj;
  wire Bi, Bj, Outm, Outn;
```

```
  wire Ai, Aj;
  wire n_0, n_1, x, y;
  mux_4 muxBL(.y (x), .in1 (Bi), .in0 (Bj), .sel (Outn));
  mux_5 muxBR(.y (y), .in1 (Bj), .in0 (Bi), .sel (Outn));
  mux_6 muxTL(.y (Ai), .in1 (n_0), .in0 (n_1), .sel (Outm));
  mux_7 muxTR(.y (Aj), .in1 (n_1), .in0 (n_0), .sel (Outm));
  INVXL g3(.A (x), .Y (n_0));
  INVXL g4(.A (y), .Y (n_1));
endmodule

module mux_8(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_9(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_10(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_11(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module ISU_2(Ai, Aj, Bi, Bj, Outm, Outn);
```

```
  input Bi, Bj, Outm, Outn;
  output Ai, Aj;
  wire Bi, Bj, Outm, Outn;
  wire Ai, Aj;
  wire n_0, n_1, x, y;
  mux_8 muxBL(.y (x), .in1 (Bi), .in0 (Bj), .sel (Outn));
  mux_9 muxBR(.y (y), .in1 (Bj), .in0 (Bi), .sel (Outn));
  mux_10 muxTL(.y (Ai), .in1 (n_0), .in0 (n_1), .sel (Outm));
  mux_11 muxTR(.y (Aj), .in1 (n_1), .in0 (n_0), .sel (Outm));
  INVXL g3(.A (y), .Y (n_1));
  INVXL g4(.A (x), .Y (n_0));
endmodule

module mux_12(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_13(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_14(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_15(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
```

```
endmodule

module ISU_3(Ai, Aj, Bi, Bj, Outm, Outn);
  input Bi, Bj, Outm, Outn;
  output Ai, Aj;
  wire Bi, Bj, Outm, Outn;
  wire Ai, Aj;
  wire n_0, n_1, x, y;
  mux_12 muxBL(.y (x), .in1 (Bi), .in0 (Bj), .sel (Outn));
  mux_13 muxBR(.y (y), .in1 (Bj), .in0 (Bi), .sel (Outn));
  mux_14 muxTL(.y (Ai), .in1 (n_0), .in0 (n_1), .sel (Outm));
  mux_15 muxTR(.y (Aj), .in1 (n_1), .in0 (n_0), .sel (Outm));
  INVXL g3(.A (x), .Y (n_0));
  INVXL g4(.A (y), .Y (n_1));
endmodule

module mux_16(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_17(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_18(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_19(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
```

```
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module ISU_4(Ai, Aj, Bi, Bj, Outm, Outn);
  input Bi, Bj, Outm, Outn;
  output Ai, Aj;
  wire Bi, Bj, Outm, Outn;
  wire Ai, Aj;
  wire n_0, n_1, x, y;
  mux_16 muxBL(.y (x), .in1 (Bi), .in0 (Bj), .sel (Outn));
  mux_17 muxBR(.y (y), .in1 (Bj), .in0 (Bi), .sel (Outn));
  mux_18 muxTL(.y (Ai), .in1 (n_0), .in0 (n_1), .sel (Outm));
  mux_19 muxTR(.y (Aj), .in1 (n_1), .in0 (n_0), .sel (Outm));
  INVXL g3(.A (x), .Y (n_0));
  INVXL g4(.A (y), .Y (n_1));
endmodule

module mux_20(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_21(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_22(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule
```

```
module mux_23(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module ISU_5(Ai, Aj, Bi, Bj, Outm, Outn);
  input Bi, Bj, Outm, Outn;
  output Ai, Aj;
  wire Bi, Bj, Outm, Outn;
  wire Ai, Aj;
  wire n_0, n_1, x, y;
  mux_20 muxBL(.y (x), .in1 (Bi), .in0 (Bj), .sel (Outn));
  mux_21 muxBR(.y (y), .in1 (Bj), .in0 (Bi), .sel (Outn));
  mux_22 muxTL(.y (Ai), .in1 (n_0), .in0 (n_1), .sel (Outm));
  mux_23 muxTR(.y (Aj), .in1 (n_1), .in0 (n_0), .sel (Outm));
  INVXL g3(.A (x), .Y (n_0));
  INVXL g4(.A (y), .Y (n_1));
endmodule

module mux_24(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_25(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_26(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
```

```
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_27(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module ISU_6(Ai, Aj, Bi, Bj, Outm, Outn);
  input Bi, Bj, Outm, Outn;
  output Ai, Aj;
  wire Bi, Bj, Outm, Outn;
  wire Ai, Aj;
  wire n_0, n_1, x, y;
  mux_24 muxBL(.y (x), .in1 (Bi), .in0 (Bj), .sel (Outn));
  mux_25 muxBR(.y (y), .in1 (Bj), .in0 (Bi), .sel (Outn));
  mux_26 muxTL(.y (Ai), .in1 (n_0), .in0 (n_1), .sel (Outm));
  mux_27 muxTR(.y (Aj), .in1 (n_1), .in0 (n_0), .sel (Outm));
  INVXL g4(.A (x), .Y (n_0));
  INVXL g3(.A (y), .Y (n_1));
endmodule

module mux_28(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_29(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_30(y, in1, in0, sel);
  input in1, in0, sel;
```

```
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module mux_31(y, in1, in0, sel);
  input in1, in0, sel;
  output y;
  wire in1, in0, sel;
  wire y;
  MX2X1 g18(.A (in0), .B (in1), .S0 (sel), .Y (y));
endmodule

module ISU_7(Ai, Aj, Bi, Bj, Outm, Outn);
  input Bi, Bj, Outm, Outn;
  output Ai, Aj;
  wire Bi, Bj, Outm, Outn;
  wire Ai, Aj;
  wire n_0, n_1, x, y;
  mux_28 muxBL(.y (x), .in1 (Bi), .in0 (Bj), .sel (Outn));
  mux_29 muxBR(.y (y), .in1 (Bj), .in0 (Bi), .sel (Outn));
  mux_30 muxTL(.y (Ai), .in1 (n_0), .in0 (n_1), .sel (Outm));
  mux_31 muxTR(.y (Aj), .in1 (n_1), .in0 (n_0), .sel (Outm));
  INVXL g3(.A (x), .Y (n_0));
  INVXL g4(.A (y), .Y (n_1));
endmodule

module xor16_1_15(out, in);
  input [15:0] in;
  output out;
  wire [15:0] in;
  wire out;
  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
  wire n_8, n_9, n_10, n_11;
  CLKXOR2X1 g167(.A (n_11), .B (n_10), .Y (out));
  XNOR2X1 g168(.A (n_2), .B (n_8), .Y (n_11));
  XNOR2X1 g169(.A (n_9), .B (n_7), .Y (n_10));
  XNOR2X1 g170(.A (n_1), .B (n_4), .Y (n_9));
  XNOR2X1 g171(.A (n_6), .B (n_3), .Y (n_8));
  XNOR2X1 g172(.A (n_5), .B (n_0), .Y (n_7));
  XNOR2X1 g173(.A (in[7]), .B (in[6]), .Y (n_6));
  XNOR2X1 g174(.A (in[11]), .B (in[10]), .Y (n_5));
```

```
  XNOR2X1 g175(.A (in[13]), .B (in[12]), .Y (n_4));
  XNOR2X1 g176(.A (in[5]), .B (in[4]), .Y (n_3));
  XNOR2X1 g177(.A (in[3]), .B (in[2]), .Y (n_2));
  XNOR2X1 g178(.A (in[15]), .B (in[14]), .Y (n_1));
  XNOR2X1 g179(.A (in[9]), .B (in[8]), .Y (n_0));
endmodule

module xor16_1_14(out, in);
  input [15:0] in;
  output out;
  wire [15:0] in;
  wire out;
  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
  wire n_8, n_9, n_10, n_11;
  CLKXOR2X1 g167(.A (n_11), .B (n_10), .Y (out));
  XNOR2X1 g168(.A (n_2), .B (n_8), .Y (n_11));
  XNOR2X1 g169(.A (n_9), .B (n_7), .Y (n_10));
  XNOR2X1 g170(.A (n_1), .B (n_4), .Y (n_9));
  XNOR2X1 g171(.A (n_6), .B (n_3), .Y (n_8));
  XNOR2X1 g172(.A (n_5), .B (n_0), .Y (n_7));
  XNOR2X1 g173(.A (in[7]), .B (in[6]), .Y (n_6));
  XNOR2X1 g174(.A (in[11]), .B (in[10]), .Y (n_5));
  XNOR2X1 g175(.A (in[13]), .B (in[12]), .Y (n_4));
  XNOR2X1 g176(.A (in[5]), .B (in[4]), .Y (n_3));
  XNOR2X1 g177(.A (in[3]), .B (in[0]), .Y (n_2));
  XNOR2X1 g178(.A (in[15]), .B (in[14]), .Y (n_1));
  XNOR2X1 g179(.A (in[9]), .B (in[8]), .Y (n_0));
endmodule

module xor16_1_13(out, in);
  input [15:0] in;
  output out;
  wire [15:0] in;
  wire out;
  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
  wire n_8, n_9, n_10, n_11;
  CLKXOR2X1 g167(.A (n_11), .B (n_10), .Y (out));
  XNOR2X1 g168(.A (n_2), .B (n_8), .Y (n_11));
  XNOR2X1 g169(.A (n_9), .B (n_7), .Y (n_10));
  XNOR2X1 g170(.A (n_1), .B (n_4), .Y (n_9));
  XNOR2X1 g171(.A (n_6), .B (n_3), .Y (n_8));
  XNOR2X1 g172(.A (n_5), .B (n_0), .Y (n_7));
  XNOR2X1 g173(.A (in[7]), .B (in[6]), .Y (n_6));
```

```
  XNOR2X1 g174(.A (in[11]), .B (in[10]), .Y (n_5));
  XNOR2X1 g175(.A (in[13]), .B (in[12]), .Y (n_4));
  XNOR2X1 g176(.A (in[5]), .B (in[4]), .Y (n_3));
  XNOR2X1 g177(.A (in[1]), .B (in[0]), .Y (n_2));
  XNOR2X1 g178(.A (in[15]), .B (in[14]), .Y (n_1));
  XNOR2X1 g179(.A (in[9]), .B (in[8]), .Y (n_0));
endmodule

module xor16_1_12(out, in);
  input [15:0] in;
  output out;
  wire [15:0] in;
  wire out;
  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
  wire n_8, n_9, n_10, n_11;
  CLKXOR2X1 g187(.A (n_11), .B (n_10), .Y (out));
  XNOR2X1 g188(.A (n_9), .B (n_8), .Y (n_11));
  XNOR2X1 g189(.A (n_7), .B (n_6), .Y (n_10));
  XNOR2X1 g190(.A (in[2]), .B (n_5), .Y (n_9));
  XNOR2X1 g191(.A (in[5]), .B (n_3), .Y (n_8));
  XNOR2X1 g192(.A (n_1), .B (n_2), .Y (n_7));
  XNOR2X1 g193(.A (n_4), .B (n_0), .Y (n_6));
  XNOR2X1 g194(.A (in[1]), .B (in[0]), .Y (n_5));
  XNOR2X1 g195(.A (in[11]), .B (in[10]), .Y (n_4));
  XNOR2X1 g196(.A (in[7]), .B (in[6]), .Y (n_3));
  XNOR2X1 g197(.A (in[13]), .B (in[12]), .Y (n_2));
  XNOR2X1 g198(.A (in[15]), .B (in[14]), .Y (n_1));
  XNOR2X1 g199(.A (in[9]), .B (in[8]), .Y (n_0));
endmodule

module xor16_1_11(out, in);
  input [15:0] in;
  output out;
  wire [15:0] in;
  wire out;
  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
  wire n_8, n_9, n_10, n_11;
  CLKXOR2X1 g153(.A (n_11), .B (n_10), .Y (out));
  XNOR2X1 g154(.A (n_2), .B (n_8), .Y (n_11));
  XNOR2X1 g155(.A (n_9), .B (n_7), .Y (n_10));
  XNOR2X1 g156(.A (n_1), .B (n_4), .Y (n_9));
  XNOR2X1 g157(.A (n_6), .B (n_3), .Y (n_8));
  XNOR2X1 g158(.A (n_5), .B (n_0), .Y (n_7));
```

```
  XNOR2X1 g159(.A (in[3]), .B (in[2]), .Y (n_6));
  XNOR2X1 g160(.A (in[11]), .B (in[10]), .Y (n_5));
  XNOR2X1 g161(.A (in[13]), .B (in[12]), .Y (n_4));
  XNOR2X1 g162(.A (in[1]), .B (in[0]), .Y (n_3));
  XNOR2X1 g163(.A (in[7]), .B (in[6]), .Y (n_2));
  XNOR2X1 g164(.A (in[15]), .B (in[14]), .Y (n_1));
  XNOR2X1 g165(.A (in[9]), .B (in[8]), .Y (n_0));
endmodule

module xor16_1_10(out, in);
  input [15:0] in;
  output out;
  wire [15:0] in;
  wire out;
  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
  wire n_8, n_9, n_10, n_11;
  CLKXOR2X1 g153(.A (n_11), .B (n_10), .Y (out));
  XNOR2X1 g154(.A (n_2), .B (n_8), .Y (n_11));
  XNOR2X1 g155(.A (n_9), .B (n_7), .Y (n_10));
  XNOR2X1 g156(.A (n_1), .B (n_4), .Y (n_9));
  XNOR2X1 g157(.A (n_6), .B (n_3), .Y (n_8));
  XNOR2X1 g158(.A (n_5), .B (n_0), .Y (n_7));
  XNOR2X1 g159(.A (in[3]), .B (in[2]), .Y (n_6));
  XNOR2X1 g160(.A (in[11]), .B (in[10]), .Y (n_5));
  XNOR2X1 g161(.A (in[13]), .B (in[12]), .Y (n_4));
  XNOR2X1 g162(.A (in[1]), .B (in[0]), .Y (n_3));
  XNOR2X1 g163(.A (in[7]), .B (in[4]), .Y (n_2));
  XNOR2X1 g164(.A (in[15]), .B (in[14]), .Y (n_1));
  XNOR2X1 g165(.A (in[9]), .B (in[8]), .Y (n_0));
endmodule

module xor16_1_9(out, in);
  input [15:0] in;
  output out;
  wire [15:0] in;
  wire out;
  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
  wire n_8, n_9, n_10, n_11;
  CLKXOR2X1 g153(.A (n_11), .B (n_10), .Y (out));
  XNOR2X1 g154(.A (n_2), .B (n_8), .Y (n_11));
  XNOR2X1 g155(.A (n_9), .B (n_7), .Y (n_10));
  XNOR2X1 g156(.A (n_1), .B (n_4), .Y (n_9));
  XNOR2X1 g157(.A (n_6), .B (n_3), .Y (n_8));
```

```
  XNOR2X1 g158(.A (n_5), .B (n_0), .Y (n_7));
  XNOR2X1 g159(.A (in[3]), .B (in[2]), .Y (n_6));
  XNOR2X1 g160(.A (in[11]), .B (in[10]), .Y (n_5));
  XNOR2X1 g161(.A (in[13]), .B (in[12]), .Y (n_4));
  XNOR2X1 g162(.A (in[1]), .B (in[0]), .Y (n_3));
  XNOR2X1 g163(.A (in[5]), .B (in[4]), .Y (n_2));
  XNOR2X1 g164(.A (in[15]), .B (in[14]), .Y (n_1));
  XNOR2X1 g165(.A (in[9]), .B (in[8]), .Y (n_0));
endmodule

module xor16_1_8(out, in);
  input [15:0] in;
  output out;
  wire [15:0] in;
  wire out;
  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
  wire n_8, n_9, n_10, n_11;
  CLKXOR2X1 g207(.A (n_11), .B (n_10), .Y (out));
  XNOR2X1 g208(.A (n_9), .B (n_6), .Y (n_11));
  XNOR2X1 g209(.A (n_8), .B (n_7), .Y (n_10));
  XNOR2X1 g210(.A (in[6]), .B (n_5), .Y (n_9));
  XNOR2X1 g211(.A (in[9]), .B (n_3), .Y (n_8));
  XNOR2X1 g212(.A (n_0), .B (n_1), .Y (n_7));
  XNOR2X1 g213(.A (n_2), .B (n_4), .Y (n_6));
  XNOR2X1 g214(.A (in[5]), .B (in[4]), .Y (n_5));
  XNOR2X1 g215(.A (in[1]), .B (in[0]), .Y (n_4));
  XNOR2X1 g216(.A (in[11]), .B (in[10]), .Y (n_3));
  XNOR2X1 g217(.A (in[3]), .B (in[2]), .Y (n_2));
  XNOR2X1 g218(.A (in[13]), .B (in[12]), .Y (n_1));
  XNOR2X1 g219(.A (in[15]), .B (in[14]), .Y (n_0));
endmodule

module xor16_1_7(out, in);
  input [15:0] in;
  output out;
  wire [15:0] in;
  wire out;
  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
  wire n_8, n_9, n_10, n_11;
  CLKXOR2X1 g165(.A (n_11), .B (n_10), .Y (out));
  XNOR2X1 g166(.A (n_2), .B (n_8), .Y (n_11));
  XNOR2X1 g167(.A (n_9), .B (n_7), .Y (n_10));
  XNOR2X1 g168(.A (n_1), .B (n_4), .Y (n_9));
```

```
  XNOR2X1 g169(.A (n_6), .B (n_3), .Y (n_8));
  XNOR2X1 g170(.A (n_5), .B (n_0), .Y (n_7));
  XNOR2X1 g171(.A (in[15]), .B (in[14]), .Y (n_6));
  XNOR2X1 g172(.A (in[3]), .B (in[2]), .Y (n_5));
  XNOR2X1 g173(.A (in[5]), .B (in[4]), .Y (n_4));
  XNOR2X1 g174(.A (in[13]), .B (in[12]), .Y (n_3));
  XNOR2X1 g175(.A (in[11]), .B (in[10]), .Y (n_2));
  XNOR2X1 g176(.A (in[7]), .B (in[6]), .Y (n_1));
  XNOR2X1 g177(.A (in[1]), .B (in[0]), .Y (n_0));
endmodule

module xor16_1_6(out, in);
  input [15:0] in;
  output out;
  wire [15:0] in;
  wire out;
  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
  wire n_8, n_9, n_10, n_11;
  CLKXOR2X1 g165(.A (n_11), .B (n_10), .Y (out));
  XNOR2X1 g166(.A (n_2), .B (n_8), .Y (n_11));
  XNOR2X1 g167(.A (n_9), .B (n_7), .Y (n_10));
  XNOR2X1 g168(.A (n_1), .B (n_4), .Y (n_9));
  XNOR2X1 g169(.A (n_6), .B (n_3), .Y (n_8));
  XNOR2X1 g170(.A (n_5), .B (n_0), .Y (n_7));
  XNOR2X1 g171(.A (in[15]), .B (in[14]), .Y (n_6));
  XNOR2X1 g172(.A (in[3]), .B (in[2]), .Y (n_5));
  XNOR2X1 g173(.A (in[5]), .B (in[4]), .Y (n_4));
  XNOR2X1 g174(.A (in[13]), .B (in[12]), .Y (n_3));
  XNOR2X1 g175(.A (in[11]), .B (in[8]), .Y (n_2));
  XNOR2X1 g176(.A (in[7]), .B (in[6]), .Y (n_1));
  XNOR2X1 g177(.A (in[1]), .B (in[0]), .Y (n_0));
endmodule

module xor16_1_5(out, in);
  input [15:0] in;
  output out;
  wire [15:0] in;
  wire out;
  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
  wire n_8, n_9, n_10, n_11;
  CLKXOR2X1 g165(.A (n_11), .B (n_10), .Y (out));
  XNOR2X1 g166(.A (n_2), .B (n_8), .Y (n_11));
  XNOR2X1 g167(.A (n_9), .B (n_7), .Y (n_10));
```

```
  XNOR2X1 g168(.A (n_1), .B (n_4), .Y (n_9));
  XNOR2X1 g169(.A (n_6), .B (n_3), .Y (n_8));
  XNOR2X1 g170(.A (n_5), .B (n_0), .Y (n_7));
  XNOR2X1 g171(.A (in[15]), .B (in[14]), .Y (n_6));
  XNOR2X1 g172(.A (in[3]), .B (in[2]), .Y (n_5));
  XNOR2X1 g173(.A (in[5]), .B (in[4]), .Y (n_4));
  XNOR2X1 g174(.A (in[13]), .B (in[12]), .Y (n_3));
  XNOR2X1 g175(.A (in[9]), .B (in[8]), .Y (n_2));
  XNOR2X1 g176(.A (in[7]), .B (in[6]), .Y (n_1));
  XNOR2X1 g177(.A (in[1]), .B (in[0]), .Y (n_0));
endmodule

module xor16_1_4(out, in);
  input [15:0] in;
  output out;
  wire [15:0] in;
  wire out;
  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
  wire n_8, n_9, n_10, n_11;
  CLKXOR2X1 g203(.A (n_11), .B (n_10), .Y (out));
  XNOR2X1 g204(.A (n_9), .B (n_8), .Y (n_11));
  XNOR2X1 g205(.A (n_7), .B (n_6), .Y (n_10));
  XNOR2X1 g206(.A (in[10]), .B (n_5), .Y (n_9));
  XNOR2X1 g207(.A (in[13]), .B (n_3), .Y (n_8));
  XNOR2X1 g208(.A (n_1), .B (n_2), .Y (n_7));
  XNOR2X1 g209(.A (n_4), .B (n_0), .Y (n_6));
  XNOR2X1 g210(.A (in[9]), .B (in[8]), .Y (n_5));
  XNOR2X1 g211(.A (in[3]), .B (in[2]), .Y (n_4));
  XNOR2X1 g212(.A (in[15]), .B (in[14]), .Y (n_3));
  XNOR2X1 g213(.A (in[5]), .B (in[4]), .Y (n_2));
  XNOR2X1 g214(.A (in[7]), .B (in[6]), .Y (n_1));
  XNOR2X1 g215(.A (in[1]), .B (in[0]), .Y (n_0));
endmodule

module xor16_1_3(out, in);
  input [15:0] in;
  output out;
  wire [15:0] in;
  wire out;
  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
  wire n_8, n_9, n_10, n_11;
  CLKXOR2X1 g159(.A (n_11), .B (n_10), .Y (out));
  XNOR2X1 g160(.A (n_2), .B (n_8), .Y (n_11));
```

```
  XNOR2X1 g161(.A (n_9), .B (n_7), .Y (n_10));
  XNOR2X1 g162(.A (n_1), .B (n_4), .Y (n_9));
  XNOR2X1 g163(.A (n_6), .B (n_3), .Y (n_8));
  XNOR2X1 g164(.A (n_5), .B (n_0), .Y (n_7));
  XNOR2X1 g165(.A (in[11]), .B (in[10]), .Y (n_6));
  XNOR2X1 g166(.A (in[3]), .B (in[2]), .Y (n_5));
  XNOR2X1 g167(.A (in[5]), .B (in[4]), .Y (n_4));
  XNOR2X1 g168(.A (in[9]), .B (in[8]), .Y (n_3));
  XNOR2X1 g169(.A (in[15]), .B (in[14]), .Y (n_2));
  XNOR2X1 g170(.A (in[7]), .B (in[6]), .Y (n_1));
  XNOR2X1 g171(.A (in[1]), .B (in[0]), .Y (n_0));
endmodule

module xor16_1_2(out, in);
  input [15:0] in;
  output out;
  wire [15:0] in;
  wire out;
  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
  wire n_8, n_9, n_10, n_11;
  CLKXOR2X1 g159(.A (n_11), .B (n_10), .Y (out));
  XNOR2X1 g160(.A (n_2), .B (n_8), .Y (n_11));
  XNOR2X1 g161(.A (n_9), .B (n_7), .Y (n_10));
  XNOR2X1 g162(.A (n_1), .B (n_4), .Y (n_9));
  XNOR2X1 g163(.A (n_6), .B (n_3), .Y (n_8));
  XNOR2X1 g164(.A (n_5), .B (n_0), .Y (n_7));
  XNOR2X1 g165(.A (in[11]), .B (in[10]), .Y (n_6));
  XNOR2X1 g166(.A (in[3]), .B (in[2]), .Y (n_5));
  XNOR2X1 g167(.A (in[5]), .B (in[4]), .Y (n_4));
  XNOR2X1 g168(.A (in[9]), .B (in[8]), .Y (n_3));
  XNOR2X1 g169(.A (in[15]), .B (in[12]), .Y (n_2));
  XNOR2X1 g170(.A (in[7]), .B (in[6]), .Y (n_1));
  XNOR2X1 g171(.A (in[1]), .B (in[0]), .Y (n_0));
endmodule

module xor16_1_1(out, in);
  input [15:0] in;
  output out;
  wire [15:0] in;
  wire out;
  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
  wire n_8, n_9, n_10;
  XNOR2X1 g149(.A (n_10), .B (n_9), .Y (out));
```

```
  XNOR2X1 g150(.A (n_8), .B (n_7), .Y (n_10));
  XNOR2X1 g151(.A (n_3), .B (n_6), .Y (n_9));
  XNOR2X1 g152(.A (in[1]), .B (n_0), .Y (n_8));
  XNOR2X1 g153(.A (n_2), .B (n_4), .Y (n_7));
  XNOR2X1 g154(.A (n_5), .B (n_1), .Y (n_6));
  XNOR2X1 g155(.A (in[11]), .B (in[10]), .Y (n_5));
  XNOR2X1 g156(.A (in[5]), .B (in[4]), .Y (n_4));
  XNOR2X1 g157(.A (in[13]), .B (in[12]), .Y (n_3));
  XNOR2X1 g158(.A (in[7]), .B (in[6]), .Y (n_2));
  XNOR2X1 g159(.A (in[9]), .B (in[8]), .Y (n_1));
  XNOR2X1 g160(.A (in[3]), .B (in[2]), .Y (n_0));
endmodule

module xor16_1(out, in);
  input [15:0] in;
  output out;
  wire [15:0] in;
  wire out;
  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
  wire n_8, n_9, n_10, n_11, n_12;
  XNOR2X1 g237(.A (n_12), .B (n_11), .Y (out));
  XNOR2X1 g238(.A (n_10), .B (n_8), .Y (n_12));
  XNOR2X1 g239(.A (n_9), .B (n_7), .Y (n_11));
  XNOR2X1 g240(.A (in[14]), .B (n_6), .Y (n_10));
  XNOR2X1 g241(.A (n_0), .B (n_3), .Y (n_9));
  XNOR2X1 g242(.A (n_2), .B (n_5), .Y (n_8));
  XNOR2X1 g243(.A (n_1), .B (n_4), .Y (n_7));
  XNOR2X1 g244(.A (in[13]), .B (in[12]), .Y (n_6));
  XNOR2X1 g245(.A (in[9]), .B (in[8]), .Y (n_5));
  XNOR2X1 g246(.A (in[1]), .B (in[0]), .Y (n_4));
  XNOR2X1 g247(.A (in[5]), .B (in[4]), .Y (n_3));
  XNOR2X1 g248(.A (in[11]), .B (in[10]), .Y (n_2));
  XNOR2X1 g249(.A (in[3]), .B (in[2]), .Y (n_1));
  XNOR2X1 g250(.A (in[7]), .B (in[6]), .Y (n_0));
endmodule

module trng(Out, A, B, temp, put_seed, seed, clk);
  input put_seed, clk;
  input [15:0] seed;
  output [15:0] Out, A, B, temp;
  wire put_seed, clk;
  wire [15:0] seed;
  wire [15:0] Out, A, B, temp;
```

```
  wire UNCONNECTED_HIER_Z, UNCONNECTED_HIER_Z0,
UNCONNECTED_HIER_Z1,
      UNCONNECTED_HIER_Z2, UNCONNECTED_HIER_Z3, UNCONNECTED_HIER_Z4,
      UNCONNECTED_HIER_Z5, UNCONNECTED_HIER_Z6;
  wire UNCONNECTED_HIER_Z7, UNCONNECTED_HIER_Z8,
UNCONNECTED_HIER_Z9,
      UNCONNECTED_HIER_Z10, UNCONNECTED_HIER_Z11,
      UNCONNECTED_HIER_Z12, UNCONNECTED_HIER_Z13,
UNCONNECTED_HIER_Z14;
  wire UNCONNECTED_HIER_Z15, UNCONNECTED_HIER_Z16,
      UNCONNECTED_HIER_Z17, UNCONNECTED_HIER_Z18,
      UNCONNECTED_HIER_Z19, UNCONNECTED_HIER_Z20,
      UNCONNECTED_HIER_Z21, UNCONNECTED_HIER_Z22;
  wire UNCONNECTED_HIER_Z23, UNCONNECTED_HIER_Z24,
      UNCONNECTED_HIER_Z25, UNCONNECTED_HIER_Z26,
      UNCONNECTED_HIER_Z27, UNCONNECTED_HIER_Z28,
      UNCONNECTED_HIER_Z29, UNCONNECTED_HIER_Z30;
  assign A[0] = Out[0];
  assign A[1] = Out[1];
  assign A[2] = Out[2];
  assign A[3] = Out[3];
  assign A[4] = Out[4];
  assign A[5] = Out[5];
  assign A[6] = Out[6];
  assign A[7] = Out[7];
  assign A[8] = Out[8];
  assign A[9] = Out[9];
  assign A[10] = Out[10];
  assign A[11] = Out[11];
  assign A[12] = Out[12];
  assign A[13] = Out[13];
  assign A[14] = Out[14];
  assign A[15] = Out[15];
  ISU ISU1_2(temp[0], temp[1], B[0], B[1], Out[0], Out[1]);
  ISU_1 ISU3_4(temp[2], temp[3], B[2], B[3], Out[2], Out[3]);
  ISU_2 ISU5_6(temp[4], temp[5], B[4], B[5], Out[4], Out[5]);
  ISU_3 ISU7_8(temp[6], temp[7], B[6], B[7], Out[6], Out[7]);
  ISU_4 ISU9_10(temp[8], temp[9], B[8], B[9], Out[8], Out[9]);
  ISU_5 ISU11_12(temp[10], temp[11], B[10], B[11], Out[10], Out[11]);
  ISU_6 ISU13_14(temp[12], temp[13], B[12], B[13], Out[12], Out[13]);
  ISU_7 ISU15_16(temp[14], temp[15], B[14], B[15], Out[14], Out[15]);
  xor16_1_15 xor1(B[0], {Out[15:2], UNCONNECTED_HIER_Z0,
      UNCONNECTED_HIER_Z});
```

```
xor16_1_14 xor2(B[1], {Out[15:3], UNCONNECTED_HIER_Z2,
    UNCONNECTED_HIER_Z1, Out[0]});
xor16_1_13 xor3(B[2], {Out[15:4], UNCONNECTED_HIER_Z4,
    UNCONNECTED_HIER_Z3, Out[1:0]});
xor16_1_12 xor4(B[3], {Out[15:5], UNCONNECTED_HIER_Z6,
    UNCONNECTED_HIER_Z5, Out[2:0]});
xor16_1_11 xor5(B[4], {Out[15:6], UNCONNECTED_HIER_Z8,
    UNCONNECTED_HIER_Z7, Out[3:0]});
xor16_1_10 xor6(B[5], {Out[15:7], UNCONNECTED_HIER_Z10,
    UNCONNECTED_HIER_Z9, Out[4:0]});
xor16_1_9 xor7(B[6], {Out[15:8], UNCONNECTED_HIER_Z12,
    UNCONNECTED_HIER_Z11, Out[5:0]});
xor16_1_8 xor8(B[7], {Out[15:9], UNCONNECTED_HIER_Z14,
    UNCONNECTED_HIER_Z13, Out[6:0]});
xor16_1_7 xor9(B[8], {Out[15:10], UNCONNECTED_HIER_Z16,
    UNCONNECTED_HIER_Z15, Out[7:0]});
xor16_1_6 xor10(B[9], {Out[15:11], UNCONNECTED_HIER_Z18,
    UNCONNECTED_HIER_Z17, Out[8:0]});
xor16_1_5 xor11(B[10], {Out[15:12], UNCONNECTED_HIER_Z20,
    UNCONNECTED_HIER_Z19, Out[9:0]});
xor16_1_4 xor12(B[11], {Out[15:13], UNCONNECTED_HIER_Z22,
    UNCONNECTED_HIER_Z21, Out[10:0]});
xor16_1_3 xor13(B[12], {Out[15:14], UNCONNECTED_HIER_Z24,
    UNCONNECTED_HIER_Z23, Out[11:0]});
xor16_1_2 xor14(B[13], {Out[15], UNCONNECTED_HIER_Z26,
    UNCONNECTED_HIER_Z25, Out[12:0]});
xor16_1_1 xor15(B[14], {UNCONNECTED_HIER_Z29, UNCONNECTED_HIER_Z28,
    Out[13:1], UNCONNECTED_HIER_Z27});
xor16_1 xor16(B[15], {UNCONNECTED_HIER_Z30, Out[14:0]});
MDFFHQX1 \Out_reg[6] (.CK (clk), .D0 (temp[6]), .D1 (seed[6]), .S0
    (put_seed), .Q (Out[6]));
MDFFHQX1 \Out_reg[3] (.CK (clk), .D0 (temp[3]), .D1 (seed[3]), .S0
    (put_seed), .Q (Out[3]));
MDFFHQX1 \Out_reg[4] (.CK (clk), .D0 (temp[4]), .D1 (seed[4]), .S0
    (put_seed), .Q (Out[4]));
MDFFHQX1 \Out_reg[5] (.CK (clk), .D0 (temp[5]), .D1 (seed[5]), .S0
    (put_seed), .Q (Out[5]));
MDFFHQX1 \Out_reg[0] (.CK (clk), .D0 (temp[0]), .D1 (seed[0]), .S0
    (put_seed), .Q (Out[0]));
MDFFHQX1 \Out_reg[7] (.CK (clk), .D0 (temp[7]), .D1 (seed[7]), .S0
    (put_seed), .Q (Out[7]));
MDFFHQX1 \Out_reg[8] (.CK (clk), .D0 (temp[8]), .D1 (seed[8]), .S0
    (put_seed), .Q (Out[8]));
```

```
  MDFFHQX1 \Out_reg[9] (.CK (clk), .D0 (temp[9]), .D1 (seed[9]), .S0
      (put_seed), .Q (Out[9]));
  MDFFHQX1 \Out_reg[13] (.CK (clk), .D0 (temp[13]), .D1 (seed[13]), .S0
      (put_seed), .Q (Out[13]));
  MDFFHQX1 \Out_reg[1] (.CK (clk), .D0 (temp[1]), .D1 (seed[1]), .S0
      (put_seed), .Q (Out[1]));
  MDFFHQX1 \Out_reg[11] (.CK (clk), .D0 (temp[11]), .D1 (seed[11]), .S0
      (put_seed), .Q (Out[11]));
  MDFFHQX1 \Out_reg[12] (.CK (clk), .D0 (temp[12]), .D1 (seed[12]), .S0
      (put_seed), .Q (Out[12]));
  MDFFHQX1 \Out_reg[10] (.CK (clk), .D0 (temp[10]), .D1 (seed[10]), .S0
      (put_seed), .Q (Out[10]));
  MDFFHQX1 \Out_reg[14] (.CK (clk), .D0 (temp[14]), .D1 (seed[14]), .S0
      (put_seed), .Q (Out[14]));
  MDFFHQX1 \Out_reg[2] (.CK (clk), .D0 (temp[2]), .D1 (seed[2]), .S0
      (put_seed), .Q (Out[2]));
  MDFFHQX1 \Out_reg[15] (.CK (clk), .D0 (temp[15]), .D1 (seed[15]), .S0
      (put_seed), .Q (Out[15]));
endmodule
```

Genus RTL Synthesised Schematic:

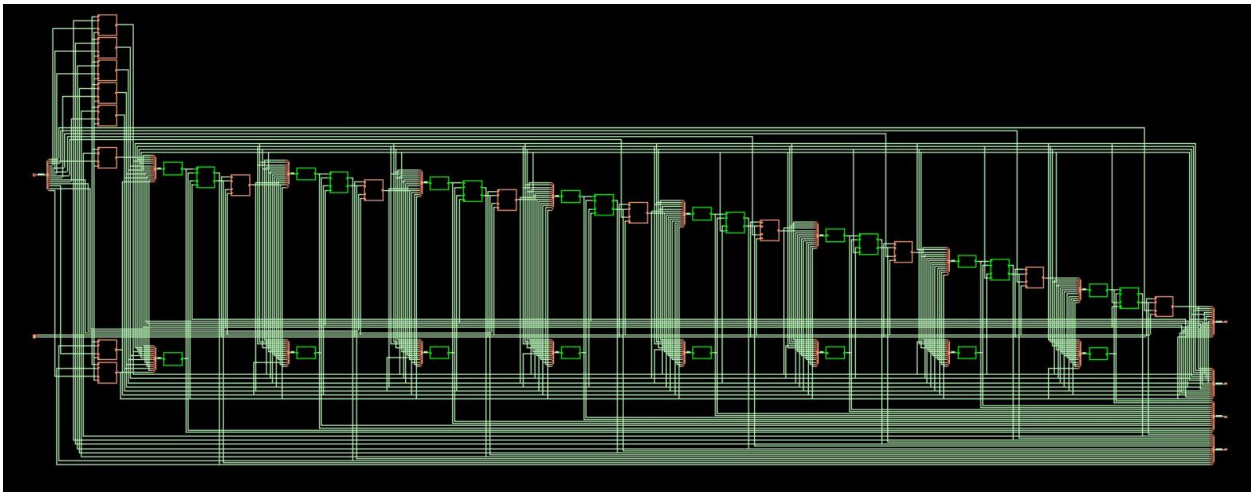


Figure 7: Genus RTL Schematic

Total Gate(272)

Type	Instances	Area	Area %
sequential	16	136.80	18.80
inverter	16	10.94	1.50
logic	240	579.35	79.70
physical_cells	0	0.00	0.00
TOTAL	272	727.09	100.00

Table 2: Gate number & area calculation

Final Layout

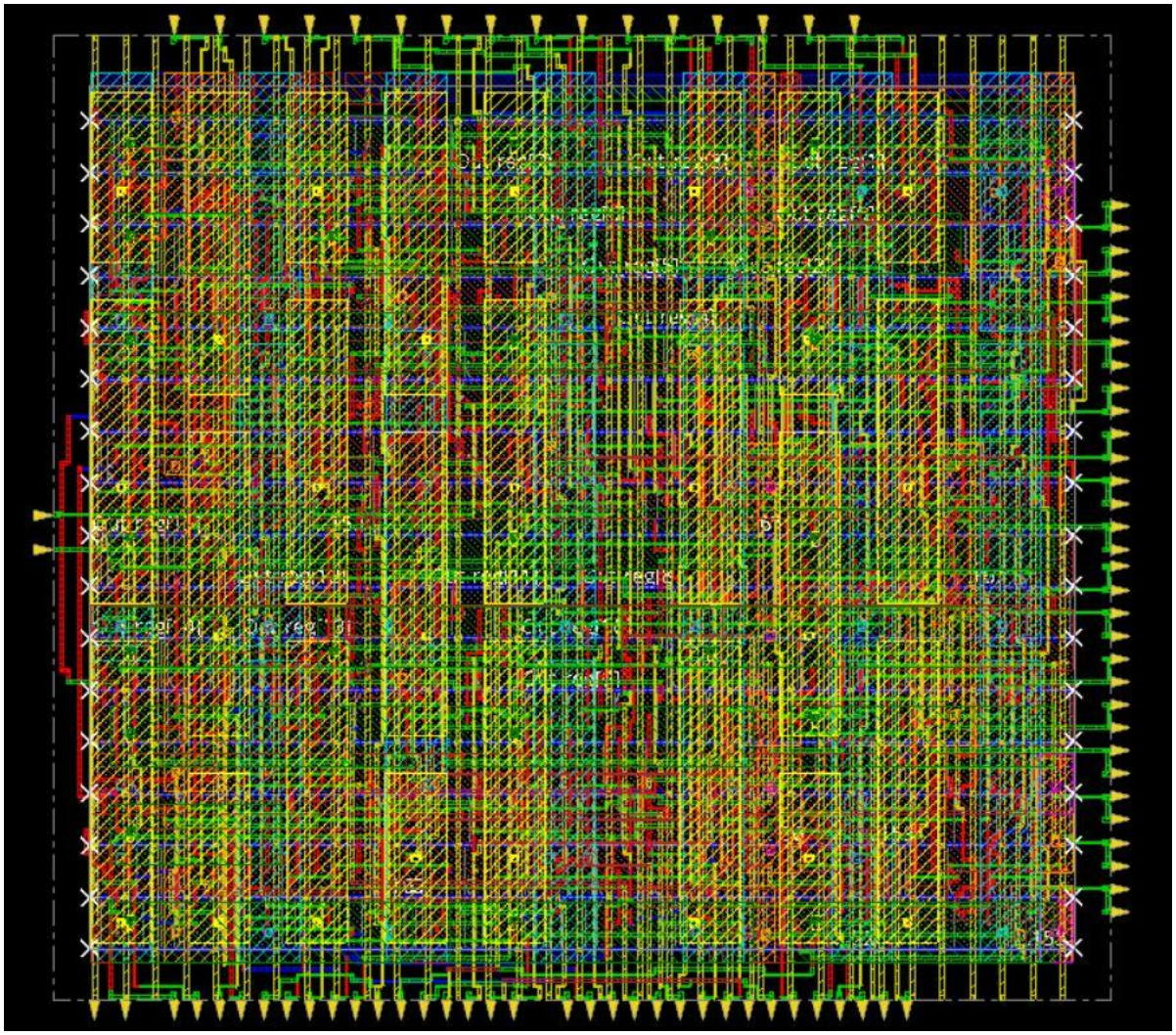


Figure 8: Final Layout after filled

Report Summary

Timing Report:

```
Other End Arrival Time          0.000
- Setup                         0.040
+ Phase Shift                   2.500
= Required Time                 2.460
- Arrival Time                  0.374
= Slack Time                    2.087
     Clock Rise Edge                      0.000
     + Clock Network Latency (Prop)       0.000
     = Beginpoint Arrival Time            0.000
     +-------------------------------------------------------------------------+
     |     Instance     |     Arc     |   Cell    | Delay | Arrival | Required |
     |                  |             |           |       |  Time   |   Time   |
     |------------------+-------------+-----------+-------+---------+----------|
     | Out_reg[6]       | CK ^        |           |       |   0.000 |    2.087 |
     | Out_reg[6]       | CK ^ -> Q v | MDFFHQX1  | 0.096 |   0.096 |    2.183 |
     | xor2/g173        | B v -> Y ^  | XNOR2X1   | 0.044 |   0.141 |    2.228 |
     | xor2/g171        | A ^ -> Y v  | XNOR2X1   | 0.045 |   0.186 |    2.273 |
     | xor2/g168        | B v -> Y ^  | XNOR2X1   | 0.038 |   0.224 |    2.311 |
     | xor2/g167        | A ^ -> Y v  | CLKXOR2X1 | 0.055 |   0.278 |    2.365 |
     | ISU1_2/muxBL/g18 | A v -> Y v  | MX2X1     | 0.039 |   0.318 |    2.405 |
     | ISU1_2/g3        | A v -> Y ^  | INVXL     | 0.013 |   0.330 |    2.417 |
     | ISU1_2/muxTR/g18 | A ^ -> Y ^  | MX2X1     | 0.043 |   0.373 |    2.460 |
     | Out_reg[1]       | D0 ^        | MDFFHQX1  | 0.000 |   0.374 |    2.460 |
     +-------------------------------------------------------------------------+
```

Figure 9: Timing Report

Here we got a positive slack time of 2.087ns. The arrival time of all the gates is less than the required time as expected.

Hold time report:

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Setup views included:
 func@BC_rcbest0.hold

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  2.087  |  2.087  |  2.460  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|   48    |   16    |   32    |
+--------------------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 81.518%
Total number of glitch violations: 0
------------------------------------------------------------
Reported timing to dir ./timingReports
Total CPU time: 2.73 sec
Total Real time: 3.0 sec
Total Memory Usage: 1261.910156 Mbytes
Reset AAE Options
```

Figure 10: Hold Time Report

We see that there is no violating path in our design. So, the design satisfies the timing requirements.

Post-Route timing and SI Optimization:

```
------------------------------------------------------------
     optDesign Final SI Timing Summary
------------------------------------------------------------

Setup views included:
 func@BC_rcbest0.hold
Hold  views included:
 func@BC_rcbest0.hold

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  2.087  |  2.087  |  2.460  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|   48    |   16    |   32    |
+--------------------+---------+---------+---------+

+--------------------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  0.009  |  0.151  |  0.009  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|   48    |   16    |   32    |
+--------------------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 81.518%
Total number of glitch violations: 0
------------------------------------------------------------
**optDesign ... cpu = 0:00:08, real = 0:00:10, mem = 1425.4M, totSessionCpu=0:00:35 **
 ReSet Options after AAE Based Opt flow
*** Finished optDesign ***
Removing temporary dont_use automatically set for cells with technology sites with no row.
0
```

Figure 11: Post Route Timing & SI optimization

There is no timing violation after post-route as well.

Verify Connectivity Report:

```
******** Start: VERIFY CONNECTIVITY ********
Start Time: Tue Mar  1 20:09:30 2022

Design Name: trng
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (35.0000, 31.9200)
Error Limit = 1000; Warning Limit = 50
Check all nets

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Tue Mar  1 20:09:30 2022
Time Elapsed: 0:00:00.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.0  MEM: 1.000M)
```

Figure 12: Verified Connectivity Report

There is no timing violation in connectivity as well.

Verify Geometry Report:

```
innovus 1>  *** Starting Verify Geometry (MEM: 888.3) ***

**WARN: (IMPVFG-257):   verifyGeometry command is replaced by verify_drc command.
  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 1920
  VERIFY GEOMETRY ..... SubArea : 1 of 1
  VERIFY GEOMETRY ..... Cells          :  0 Viols.
  VERIFY GEOMETRY ..... SameNet        :  0 Viols.
  VERIFY GEOMETRY ..... Wiring         :  0 Viols.
  VERIFY GEOMETRY ..... Antenna        :  0 Viols.
  VERIFY GEOMETRY ..... Sub-Area : 1 complete 0 Viols. 0 Wrngs.
VG: elapsed time: 0.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.

**********End: VERIFY GEOMETRY**********
 *** verify geometry (CPU: 0:00:00.2  MEM: 188.3M)
```

Figure 13: Verified Geometry Report

There is no timing violation in geometry as well.

Worst Slack Report (From Genus):

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)
Out_reg[14]/CK				0.00		0.00
Out_reg[14]/Q	MDFFHQX1	18	7.80	53.50	75.60	75.60
xor4/in[14]						
g198/B					0.00	75.60
g198/Y	XNOR2X1	1	0.30	6.10	39.00	114.60
g192/A					0.00	114.60
g192/Y	XNOR2X1	1	0.30	6.30	42.50	157.10
g189/A					0.00	157.10
g189/Y	XNOR2X1	1	0.60	8.00	41.50	198.60
g187/B					0.00	198.60
g187/Y	CLKXOR2X1	3	0.60	8.20	37.10	235.70
xor4/out						
ISU3_4/Bj						
muxBR/in1						
g18/B					0.00	235.70
g18/Y	MX2X1	1	0.20	5.80	34.90	270.60
muxBR/y						
g4/A					0.00	270.60
g4/Y	INVXL	2	0.60	8.90	9.00	279.60
muxTL/in0						
g18/A					0.00	279.60
g18/Y	MX2X1	2	0.50	7.50	33.30	312.90
muxTL/y						
ISU3_4/Ai						
Out_reg[2]/D0	MDFFHQX1				0.00	312.90
Out_reg[2]/CK	setup			0.00	36.30	349.20

Figure 14: Worst Slack Report

Power Report (From Genus):

Instance	Cells	Leakage (nW)	Internal (nW)	Net (nW)	Switching (nW) v
tmg	272	50.24	41007.97	9029.94	50037.91
tmg/xor16	14	2.52	1485.52	334.64	1820.16
tmg/ISU15_16	6	0.69	1337.23	380.96	1718.19
tmg/xor14	13	2.34	1367.05	309.12	1676.17
tmg/ISU13_14	6	0.69	1287.44	366.15	1653.60
tmg/ISU3_4	6	0.68	1266.65	360.48	1627.13
tmg/ISU5_6	6	0.69	1238.66	352.29	1590.94
tmg/ISU11_12	6	0.69	1230.96	349.77	1580.73
tmg/ISU9_10	6	0.69	1224.59	348.19	1572.78
tmg/ISU1_2	6	0.69	1224.55	348.19	1572.74
tmg/xor15	12	2.16	1262.42	286.43	1548.85
tmg/xor13	13	2.34	1247.40	290.21	1537.61
tmg/ISU7_8	6	0.69	1182.31	335.90	1518.21
tmg/xor5	13	2.33	1202.05	276.98	1479.02
tmg/xor4	13	2.34	1187.87	277.92	1465.79
tmg/xor12	13	2.33	1191.26	268.47	1459.73
tmg/xor8	13	2.36	1181.85	267.52	1449.37
tmg/xor6	13	2.33	1173.28	272.25	1445.53
tmg/xor9	13	2.34	1169.51	268.47	1437.98
tmg/xor3	13	2.33	1166.71	270.36	1437.07
tmg/xor7	13	2.33	1166.31	270.36	1436.67
tmg/xor1	13	2.33	1167.00	269.41	1436.41
tmg/xor10	13	2.34	1161.82	267.52	1429.35
tmg/xor2	13	2.33	1159.74	268.47	1428.21
tmg/xor11	13	2.34	1155.03	265.63	1420.66
tmg/ISU15_16/muxTR	1	0.15	326.91	77.20	404.11
tmg/ISU13_14/muxTR	1	0.16	300.92	70.90	371.82
tmg/ISU13_14/muxTL	1	0.16	300.91	70.90	371.81
tmg/ISU3_4/muxTR	1	0.16	300.86	70.90	371.76
tmg/ISU15_16/muxTL	1	0.15	294.48	69.32	363.80
tmg/ISU5_6/muxTR	1	0.15	294.37	69.32	363.70
tmg/ISU3_4/muxTL	1	0.16	287.90	67.75	355.64
tmg/ISU9_10/muxTL	1	0.16	287.86	67.75	355.61
tmg/ISU1_2/muxTR	1	0.16	287.85	67.75	355.60
tmg/ISU11_12/muxTL	1	0.15	287.85	67.75	355.60
tmg/ISU11_12/muxTR	1	0.15	287.84	67.75	355.58
tmg/ISU1_2/muxTL	1	0.16	281.41	66.17	347.58
tmg/ISU5_6/muxTL	1	0.15	281.41	66.17	347.58
tmg/ISU9_10/muxTR	1	0.16	281.39	66.17	347.57
tmg/ISU7_8/muxTR	1	0.15	281.27	66.17	347.44
tmg/ISU15_16/muxBL	1	0.16	313.59	29.62	343.21
tmg/ISU15_16/muxBR	1	0.16	307.04	28.99	336.03
tmg/ISU7_8/muxTL	1	0.15	268.33	63.02	331.35
tmg/ISU13_14/muxBL	1	0.15	300.52	28.36	328.88
tmg/ISU13_14/muxBR	1	0.15	293.98	27.73	321.71
tmg/ISU3_4/muxBL	1	0.15	293.91	27.73	321.64
tmg/ISU3_4/muxBR	1	0.15	293.89	27.73	321.62
tmg/ISU1_2/muxBR	1	0.15	293.86	27.73	321.59
tmg/ISU5_6/muxBR	1	0.16	287.43	27.10	314.53
tmg/ISU5_6/muxBL	1	0.16	287.41	27.10	314.51
tmg/ISU9_10/muxBL	1	0.15	287.40	27.10	314.50
tmg/ISU11_12/muxBR	1	0.16	287.36	27.10	314.46
tmg/ISU9_10/muxBR	1	0.15	280.92	26.47	307.39
tmg/ISU11_12/muxBL	1	0.16	280.89	26.47	307.36
tmg/ISU7_8/muxBR	1	0.16	280.86	26.47	307.33
tmg/ISU1_2/muxBL	1	0.15	274.41	25.84	300.25
tmg/ISU7_8/muxBL	1	0.16	267.90	25.21	293.11

Figure 15: Power Report

DRC (Design Rule Checking) Report:

```
                SIZE: Cumulative Time CPU =         0(s) REAL =         0(s)
    EDGE TOPOLOGICAL: Cumulative Time CPU =         0(s) REAL =         0(s)
    EDGE MEASUREMENT: Cumulative Time CPU =         0(s) REAL =         0(s)
               STAMP: Cumulative Time CPU =         0(s) REAL =         0(s)
       ONE LAYER DRC: Cumulative Time CPU =         0(s) REAL =         0(s)
       TWO LAYER DRC: Cumulative Time CPU =         0(s) REAL =         1(s)
            NET AREA: Cumulative Time CPU =         0(s) REAL =         0(s)
             DENSITY: Cumulative Time CPU =         0(s) REAL =         0(s)
       MISCELLANEOUS: Cumulative Time CPU =         0(s) REAL =         0(s)
             CONNECT: Cumulative Time CPU =         0(s) REAL =         0(s)
              DEVICE: Cumulative Time CPU =         0(s) REAL =         0(s)
                 ERC: Cumulative Time CPU =         0(s) REAL =         0(s)
       PATTERN_MATCH: Cumulative Time CPU =         0(s) REAL =         0(s)
            DFM FILL: Cumulative Time CPU =         0(s) REAL =         0(s)


Total CPU Time                    : 2(s)
Total Real Time                   : 2(s)
Peak Memory Used                  : 22(M)
Total Original Geometry           : 3365(33383)
Total DRC RuleChecks              : 562
Total DRC Results                 : 0 (0)
Summary can be found in file trng.sum
ASCII report database is /home/vlsi27/cds_digital/EEE458_project_group27_v1/trng.drc_errors.ascii
Checking in all SoftShare licenses.


Design Rule Check Finished Normally. Mon Feb 21 04:33:26 2022
```

Figure 16: Virtuoso report of DRC being done

We see that the designed circuit has no DRC error. So, the design is ready to be fabricated.

Testbench/Verification

Testbench Code:

```
`timescale 1ns / 1ps

module trng_test;
wire [15:0] Out, A, B, temp;
reg put_seed, clk;
wire [15:0] seed;

   assign seed = 27492;

trng dut(Out, A, B, temp, put_seed, seed, clk);

initial begin
        clk=0;
        forever #0.05 clk=~clk;
end

initial begin
        put_seed = 1'b1;
        #0.07 put_seed = 1'b0;
end

always@(*)
   $display("%d",Out);

initial begin
$shm_open("shm.db", 1); //opens the waveform database
$shm_probe("AS");         //Saves all the signals to database
#50 $finish;
#100 $shm_close();
end

endmodule                  //end  of stimulus
```

Timing Diagram:

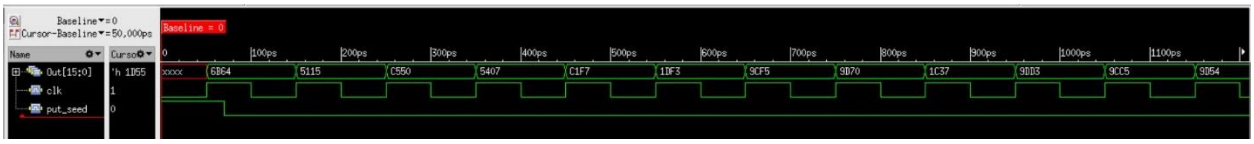


Figure 17: Timing Diagram of TRNG

Verification

But we don't know for sure if the generated sequence is random or not. So generated 10 different sequences for 10 different seed and tested them by **Wald Wolfowitz Runs Test**. We've done this through a python code in google colab. The code is given below:

```
# H0:  the sequence was produced in a random manner
# Ha:  the sequence was not produced in a random manner

# Read test data

import numpy as np
import pandas as pd
import scipy.stats as st

# Assuming number of runs greater than 10
def runs_test(d, v, alpha = 0.05):
    # Get positive and negative values
    mask = d > v
    # get runs mask
    p = mask == True
    n = mask == False
    xor = np.logical_xor(p[:-1], p[1:])
    # A run can be identified by positive
    # to negative (or vice versa) changes
    d = sum(xor) + 1 # Get number of runs

    n_p = sum(p) # Number of positives
    n_n = sum(n)
    # Temporary intermediate values
    tmp = 2 * n_p * n_n
    tmps = n_p + n_n
    # Expected value
    r_hat = np.float64(tmp) / tmps + 1
    # Variance
```

```
    s_r_squared = (tmp*(tmp - tmps)) / (tmps*tmps*(tmps-1))
    # Standard deviation
    s_r =  np.sqrt(s_r_squared)
    # Test score
    z = (d - r_hat) / s_r

    # Get normal table
    z_alpha = st.norm.ppf(1-alpha)
    # Check hypothesis
    return z, z_alpha

filename = "output.txt"

# Load array
d = np.array(np.loadtxt(filename))

(z, p) = runs_test(d, np.median(d))

pd_series = pd.Series(d)
counts = pd_series.value_counts()
e = st.entropy(counts)

print('Wald-Wolfowitz Runs Test')
if p < 0.05 : print('failed')
else : print('passed')
print('z-test statistic: %0.8s' %(z))
print('p-value: %0.8s ' %(p))
print('entropy: %0.8s' %(e))
```

Results:

Test No	Seed	Entropy	z-test statistic	p-value	Result
01	13	2.789964	17.06274	1.644853	passed
02	67	2.789964	0.132755	1.644853	passed
03	583	2.791411	17.03325	1.644853	passed
04	719	2.791411	17.03325	1.644853	passed
05	3881	2.797202	17.00375	1.644853	passed
06	35023	2.784173	17.06274	1.644853	passed
07	17203	2.084238	40.67727	1.644853	passed
08	43	2.079441	4.468141	1.644853	passed
09	27492	2.790154	0.099116	1.644853	passed
10	1347	2.789964	16.85660	1.644853	passed

Table 1: Wald Wolfowitz Runs Test Result

We assumed, **Null hypothesis** H_0: The values are randomly generated (when $\alpha \geq 0.05$)
Alternative hypothesis H_a: The values are NOT randomly generated (when $\alpha < 0.05$)
As all $p\text{-}value > \alpha = 0.05$. So we failed to reject the null hypothesis. Therefore, Numbers generated are truly random

Problems in implementation & Solution

1. Firstly, we got errors using our first architecture, so we have to try 10-12 different architectures so that our model can pass Wald Wolfowitz Runs Test.
2. We got errors after placement due to small area, so we tried different areas and by trial & error we got our minimal area solution that gives no error.
3. We also got errors as there is 82 pins in our project, so we had to place them on all 4 sides of the circuit and spread them on equal space. Therefore, chances of DRC errors get minimized.
4. We placed our ‘clk’ pin at center-left to optimize timing.

Conclusion

We have designed this 'True Random Number Generator' efficiently by placing clock at center left and putting pins on 4 sides almost equally spreaded. So we have minimal possible area of $1117.2\ \mu m^2$ & $t_{setup} < 2.5\ ps$. We followed the methods provided in our lab sheets throughout this project. As a consequence of our project work, we obtained practical expertise with the logic design and synthesis process. The Genus platform was also utilized to extract and assess time and power data. The physical designs were made with Cadence Innovus. It has helped us obtain practical experience in real-world VLSI design at the industry level. As a result, this project gave us a fantastic opportunity to immerse ourselves in the real world of VLSI design.

Reference

[1] Efficient design of chaos based 4 bit true random number generator on FPGA
[2] Hardware implementation of pseudo-random number generators based on chaotic maps
[3] Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Rings
[4] Neil H. E. Weste,David Money Harris - CMOS VLSI Design A Circuits and Systems Perspective, Fourth Edition
[5] A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications

EDA PlayGround Link

Google Colab Link