# Programmable Digital Systems – Exercise 2

Goals:

- introduction to pseudo-random number generators ,
- hardware implementation of simple pseudo-random number generators.

Exercise:

1. Run Active-HDL, create a new workspace and a new design.
2. Enter the program below into editor:

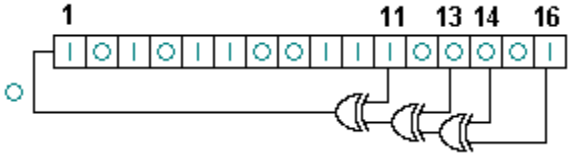
*testbench.v file:*

```
module mytestbenchmodule();
reg clk;
initial clk <= 0;
always #50  clk <= ~clk;

reg seed_stb;
reg rst;
initial
begin
    seed_stb <=0;
    rst <= 0;
    #100;
    rst <= 1;
    #500;
    rst <= 0;
    #505;
    seed_stb <= 0;
    @(posedge clk);
    seed_stb <= 1;
    @(posedge clk);
    seed_stb <= 0;
end

random1 ran1(
    .RST(rst),
    .CLK(clk),
    .SEED_DAT(16'hCAFE),
    .SEED_STB(seed_stb),
    .ENABLE(1),
    .RANDOM_WORD()
);
endmodule
```

*random1.v file:*

```
module random1
(
    input  wire RST,
    input  wire CLK,
    input  wire [15:0] SEED_DAT,
    input  wire        SEED_STB,
    input  wire        ENABLE,
    output wire [15:0] RANDOM_WORD
);

reg [15:0] r;

wire [31:0] new_r = r * 214013 + 2531011;

always @(posedge CLK or posedge RST)
if (RST) begin
    r <= 1;
end else if (SEED_STB) begin
    r <= SEED_DAT;
end else if (ENABLE) begin
    r <=  new_r[31:16];
end

assign RANDOM_WORD = r;

endmodule
```

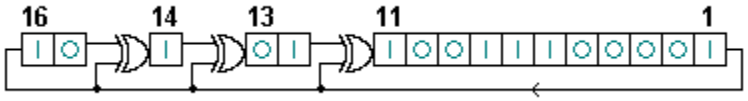
3. Compile and run the simulation. Analyze waveforms of the two modules.
4. At what time, pseudo-random generator take „the initial seed”? What is its relation with SEED_STB port? Why does it occur at this particular moment?
5. What is the principle of work of “random1” module?
6. Add a new module to the design, which uses 16-bitr Fibonacci shirting register with the following characteristic polynomial:: $x^{16} + x^{14} + x^{13} + x^{11} + 1.$




*random2.v file:*

```
module random2
(
    input  wire RST,
    input  wire CLK,
    input  wire [15:0] SEED_DAT,
    input  wire        SEED_STB,
    input  wire        ENABLE,
    output wire [15:0] RANDOM_WORD
);

reg [15:0] r;

wire b = r[0] ^ r[2] ^ r[3] ^ r[5];

always @(posedge CLK or posedge RST)
if (RST) begin
    r <= 1;
end else if (SEED_STB) begin
    r <= SEED_DAT;
end else if (ENABLE) begin
    r <=  {b, r[15:1]};
end

assign RANDOM_WORD = r;

endmodule
```

7. In „testbench.v” file, please add an instancje of „random2” module, named e.g. „ran2”.
8. Compile and run the simulation. Analyze waveforms of all three modules.
9. What is the principle of work of “random2” module?
10. Can waveform of „r” wire can be treated as pseudo-random?
11. Add a new random generator– random3 – which uses 16-bit Galois shifting register, according to the figure below:




12. In „testbench.v” file add an instance of „random3” module, named e.g. „ran3”.
13. Compile and run the simulation. Analyze waveforms.
14. At what time of the simulation pseudo-random generator “random2” take „the initial seed”? What is its relation with SEED_STB port? Why does it occur at this particular moment?
15. What is the principle of work of “random3” module and what are the differences between “random2” module?
16. Design a new „random123” module that internally uses random1, random2 and random3 pseudo-random number generators. Results of those component-modules should be “mixed” together (summed, mupltiplied etc.) in order to obtain better “randomness”. The interface should remain unchanged.
17. In „testbench.v” file add an instancje of „random123” module and compare the waveforms.
18. How can one test, which of those module is the best?