Programowalne Układy Cyfrowe – Ćwiczenie 2

Cele:

- zapoznanie z prostymi generatorami liczb pseudo-losowych,
- realizacja sprzętowa prostych generatorów liczb pseudo-losowych.

Przebieg ćwiczenia:

1. Uruchomić program Active-HDL, stworzyć nowy projekt.
2. W postaci dwóch plików źródłowych wprowadzić następujący program:

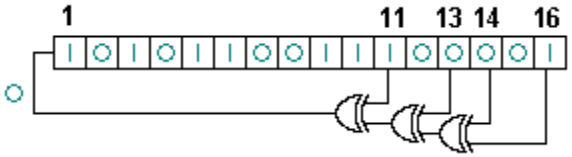
Plik testbench.v

```
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
```

Plik random1.v

```
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
```

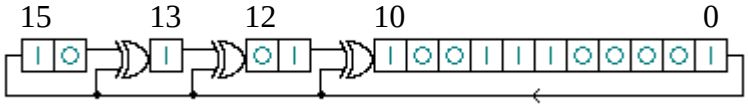
3. Skompilować i uruchomić symulację. Wykreślić przebiegi czasowe dla obu modułów.
4. W którym momencie symulacji, układ losujący przyjmuje nowe tzw. ziarno (seed) ? Jaki jest związek z portem SEED_STB? Z czego wynika właśnie ta chwila?
5. Jaka jest zasada działania generatora liczb pseudolosowych random1?
6. Do programu dodać nowy moduł wykorzystujący 16-bitowy rejestr przesuwny Fibonacci'ego o wielomianie charakterystycznym: $x^{16} + x^{14} + x^{13} + x^{11} + 1.$



Plik random2.v

```
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. W pliku testbench.v dodać instancję modułu „random2" o nazwie np. „ran2".
8. Skompilować i uruchomić symulację. Wykreślić przebiegi czasowe dla wszystkich trzech modułów.
9. Jaka jest zasada działania generatora liczb pseudolosowych random2?
10. Czy przebieg na linii „r" modułu random2 można uznać za pseudolosowy?
11. Do programu dodać trzeci moduł losujący – random3 - wykorzystujący 16-bitowy rejestr przesuwny typu Galois, według rysunku:



12. W pliku testbench.v dodać instancję modułu „random3" o nazwie np. „ran3".
13. Skompilować i uruchomić symulację. Wykreślić przebiegi czasowe.
14. W którym momencie symulacji, układ losujący random2 przyjmuje nowe tzw. ziarno (seed) ? Jaki jest związek z portem SEED_STB? Z czego wynika właśnie ta chwila?
15. Jaka jest zasada działania tego generatora liczb pseudolosowych i czym różni się od random2?
16. Zaprojektować moduł losujący „random123", którzy będzie wewnętrznie wykorzystywał moduły random1, random2 i random3. Wyniki działania tych modułów mają być „mieszane" (sumowane, mnożone, dodawane etc.) w celu uzyskania większej losowości. Interfejs modułu losującego ma pozostać niezmieniony.
17. W pliku testbench.v dodać instancję modułu „random123" i porównać przebiegi czasowe.
18. Jak można by sprawdzić który z przetestowanych modułów jest najlepszy?