МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

Кафедра: КЕОА

Курс: "Схемотехніка"

Лабораторна робота № 1

Тема: Синхронні по фронту елементи пам'яті

Виконав:

Ст. групи ДК-92

Бодак Єгор

Київ 2021

Лабораторна робота №1

Тема: “Синхронні по фронту елементи пам’яті”

Досліджувані схеми:

Завдання на лабораторну роботу

1. В Quartus Prime створіть проект 8-розрядного регістру зсуву. У якості сигналу тактової частоти використайте кнопку KEY[0]. У якості сигналу асинхронного зкидання тригерів регістру використайте кнопку KEY[1]. Вміст регістру виведіть на світлодіоди плати. Значення для молодшого розряду регістру зсуву зчитуйте з перемикача SW[0]. Перевірте роботу створеного регістру на одній з налагоджувальних плат.
2. В Quartus Prime створіть проект 8-розрядного регістру зсуву на мові Verilog. Налаштування вхідних/вихідних портів проекту такі ж самі, як для попереднього завдання. Перевірте роботу створеного регістру на одній з налагоджувальних плат.
3. В Quartus Prime створіть проект 8-розрядного регістру зсуву з лінійним зворотнім зв’язком (LFSR). У якості сигналу тактової частоти використайте кнопку KEY[0]. У якості сигналу асинхронного зкидання тригерів регістру використайте кнопку KEY[1]. Вміст регістру виведіть на світлодіоди плати. Перевірте роботу створеного регістру на одній з налагоджувальних плат.
4. В Quartus Prime створіть проект 8-розрядного регістру зсуву з лінійним зворотнім зв’язком (LFSR) на мові Verilog. Налаштування вхідних/вихідних портів проекту такі ж самі, як для попереднього завдання. Перевірте роботу створеного регістру на одній з налагоджувальних плат.
5. В Quartus Prime створіть проект SPI приймача на мові Verilog (лістинг 1.11). У якості сигналу системної тактової частоти оберіть високочастотний вхідний порт зпоміж імпортованих налаштувань. Зазвичай такий сигнал має частоту

порядку 50 МГц - 100 МГц і ім’я на зразок MAX10_CLK1_50 чи CLOCK_50 (для різних налагоджувальних плат можуть бути різні імена). У якості сигналу асинхронного скидання використайте кнопку KEY[1]. Вміст регістру даних SPI приймача виведіть на світлодіоди налагоджувальної плати. Для сигналу SPI_SCK використайте контакт GPIO[0] роз'єму GPIO. Для сигналу SPI_CS використайте контакт GPIO[1] роз'єму GPIO. Для сигналу SPI_MOSI використайте контакт GPIO[2] роз'єму GPIO. З використанням налагоджувальних плат Nucleo F401RE або Arduino Leonardo перевірте роботу SPI приймача.

6. Перегляньте зміну внутрішніх сигналів SPI приймача в логічному аналізаторі SignalTap.

ХІД РОБОТИ

1.

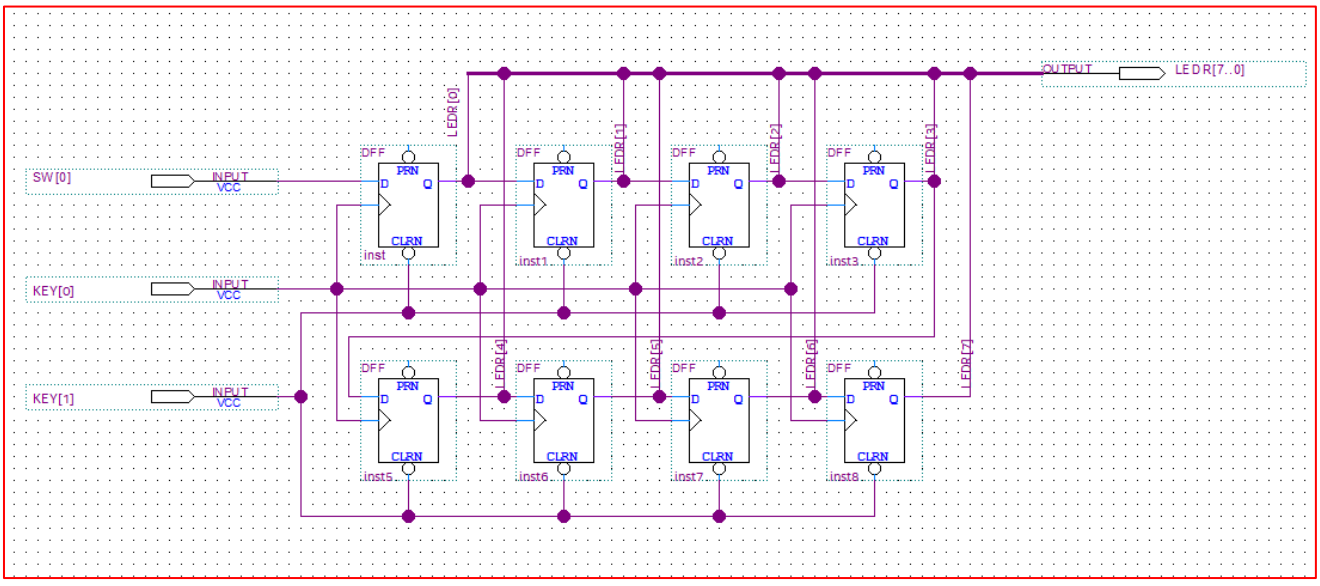

Рис.1. 8-розрядний регістру зсуву

2. Для перевірки роботи описаного пристрою використовував ModelSim, де без наявності плати можна переглянути результат симуляції, а також описати вхідні дані.

C:\intelFPGA_lite\13.0\LABS\lab_4\lab_4.v (/labs_1_) - Default *

```
Ln#
 1  //shift_reg #2
 2  module labs_1_ (KEY, SW, LEDR);
 3
 4
 5  input [1:0] KEY;
 6  input [0:0] SW;
 7
 8  output [7:0] LEDR;
 9
10  wire i_clk = KEY[0];
11  wire i_rst = KEY[1];
12  wire i_data = SW[0];
13
14  reg [7:0] shift_reg;
15
16  assign LEDR = shift_reg;
17
18  always @(posedge i_clk, negedge i_rst) begin if (~i_rst) begin
19     shift_reg <= 4'd0; end else begin
20     shift_reg <= {shift_reg[7:0], i_data};
21     end
22  end
23
24  endmodule
25
```

Рис.2. Опис 8-розрядного регістру на мові Verilog

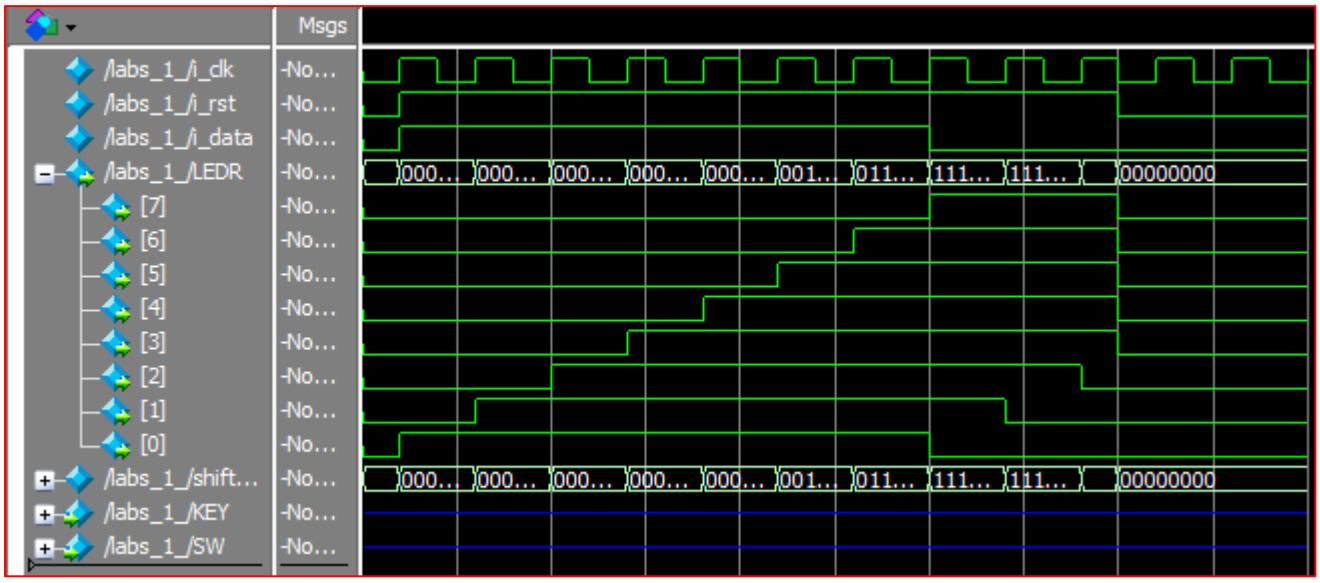

Рис.3. Перевірка роботи в ModelSim

```
13    restart;
14
15    force i_clk  0 0ns, 1 {20ns} -repeat 40ns;
16    force i_rst  0 0ns, 1 20ns, 0 400ns;
17    force i_data 0 0ns, 1 20ns, 0 300ns;
18
19
20
21    run 500ns;
```

Рис.4. Вхідні дані для тестування

3.

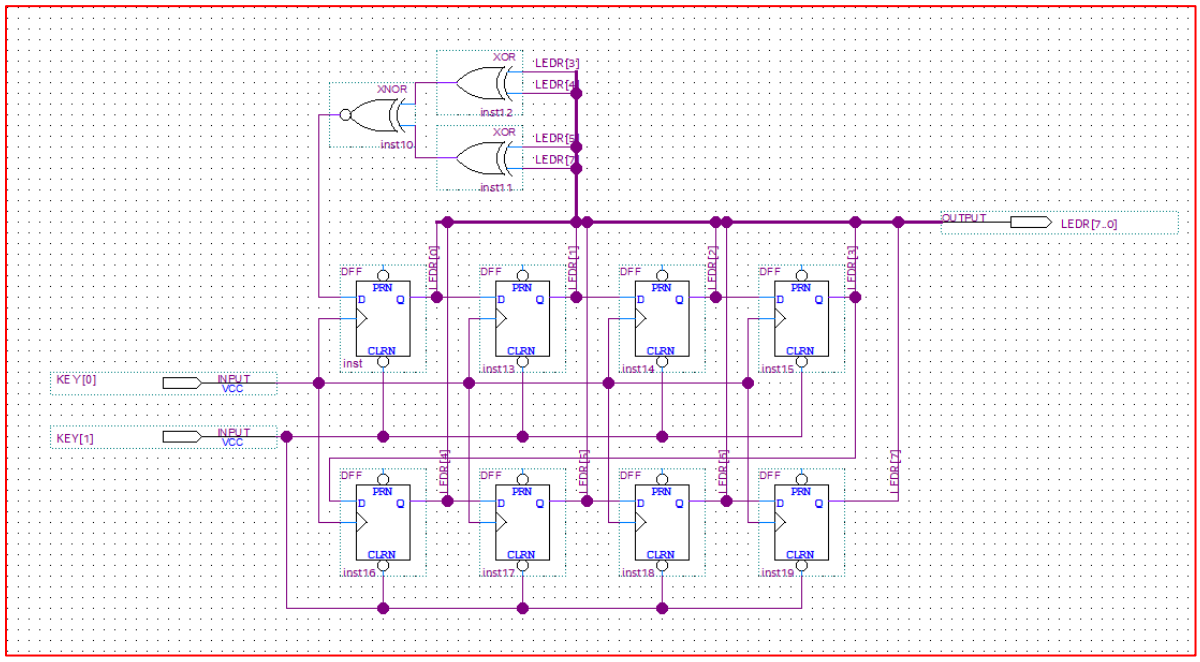

Рис.5. 8-розрядного регістру зсуву з LFSR

4. Перевіряємо роботу 8-розрядного регістру зсуву з лінійним зворотнім зв’язком (LFSR) на мові Verilog в ModelSim, хідні дані не змінюються. Регістр виконує роль генератор псевдовипадкових послідовностей бітів, тобто після скидання видно що послідовність повторюється.

```
28  //shift_reg #4
29  module labs_1_ (KEY, SW, LEDR);
30
31  input [1:0] KEY;
32  input [0:0] SW;
33
34  output [7:0] LEDR;
35
36  wire i_clk = KEY[0];
37  wire i_rst = KEY[1];
38  wire i_data = SW[0];
39
40  reg [7:0] lfsr;
41  assign LEDR = lfsr;
42  wire lfsr_lsb = ~((lfsr[3] ^ lfsr[4]) ^ (lfsr[5] ^ lfsr[7]));
43  always @(posedge i_clk, negedge i_rst) begin
44    if (~i_rst) begin
45      lfsr <= 4'd0;
46    end else begin
47    lfsr <= {lfsr[7:0], lfsr_lsb};
48  end
49  end
50  endmodule
```

Рис.6. Код програми на Verilog

(вхідні дані такі як і в завданні № 2)

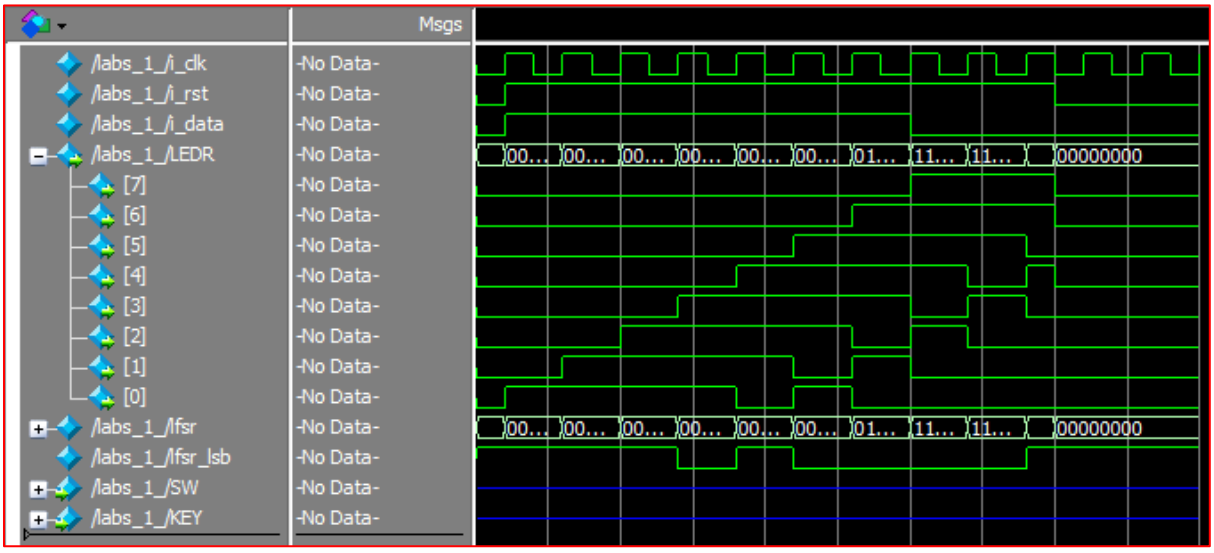

Рис.7. Результат симуляції схеми

Повторюю моделювання з додатковим скиданням щоб впевнитись в логікі псевдовипадкових бітів. У вхідні дані додав параметри і змінив час моделювання.

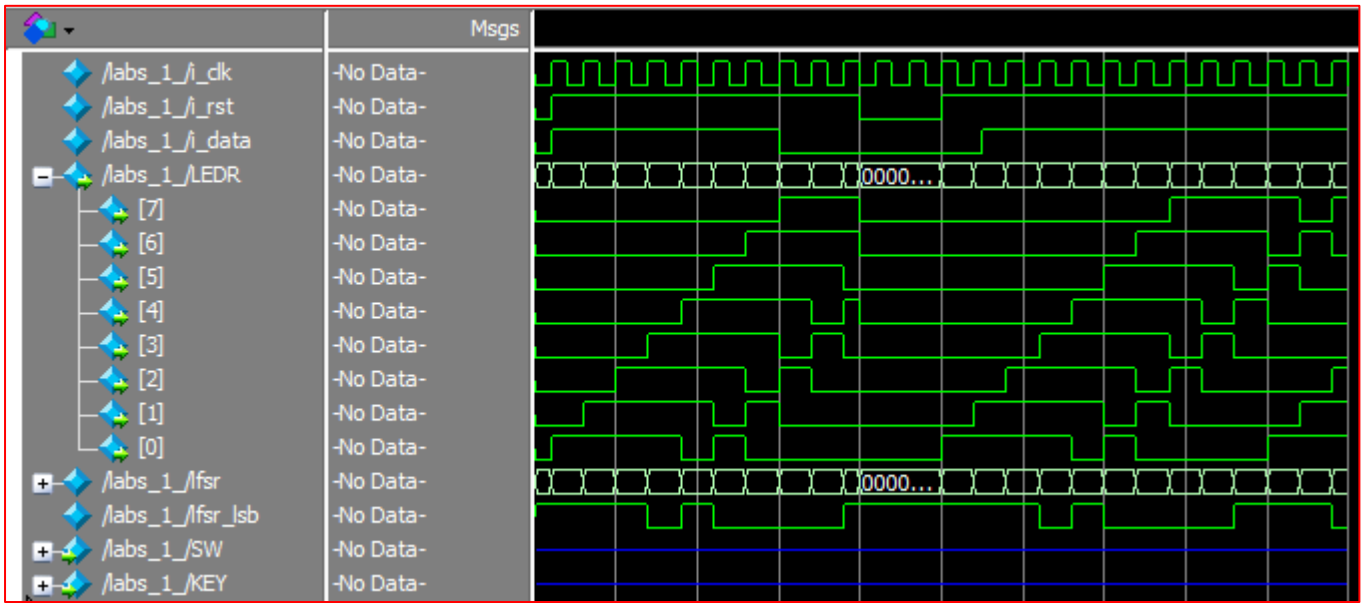

Рис.8. Результат симуляції схеми з додатковим скиданням

```
13    restart;
14
15    force i_clk  0 0ns, 1 {20ns} -repeat 40ns;
16    force i_rst  0 0ns, 1 20ns, 0 400ns, 1 500ns;
17    force i_data 0 0ns, 1 20ns, 0 300ns, 1 550ns;
18
19    run 1000ns;
```

Рис.9. Вхідні дані для поточного моделювання

Висновки

Підчас виконання роботи ознайомився з середовищем розробки пристроїв архітектури FPGA. Ознайомився з принципом роботи Д-тригерів, створення регістрів зсуву, генератор псевдо випадкових значень, яким чином синхронізуються входи FPGA, детектори фронту, як влаштований SPI і де його потрібно застосовувати. Переглянув результат роботи програмного коду Verilog в ModelSim в часовій залежності, це

достатньо зручний метод тестування так як вхідні дані можна задавати не на фізичні входи плати для тестування, а на провідники описані в коді.