Белорусский государственный университет информатики и радиоэлектроники

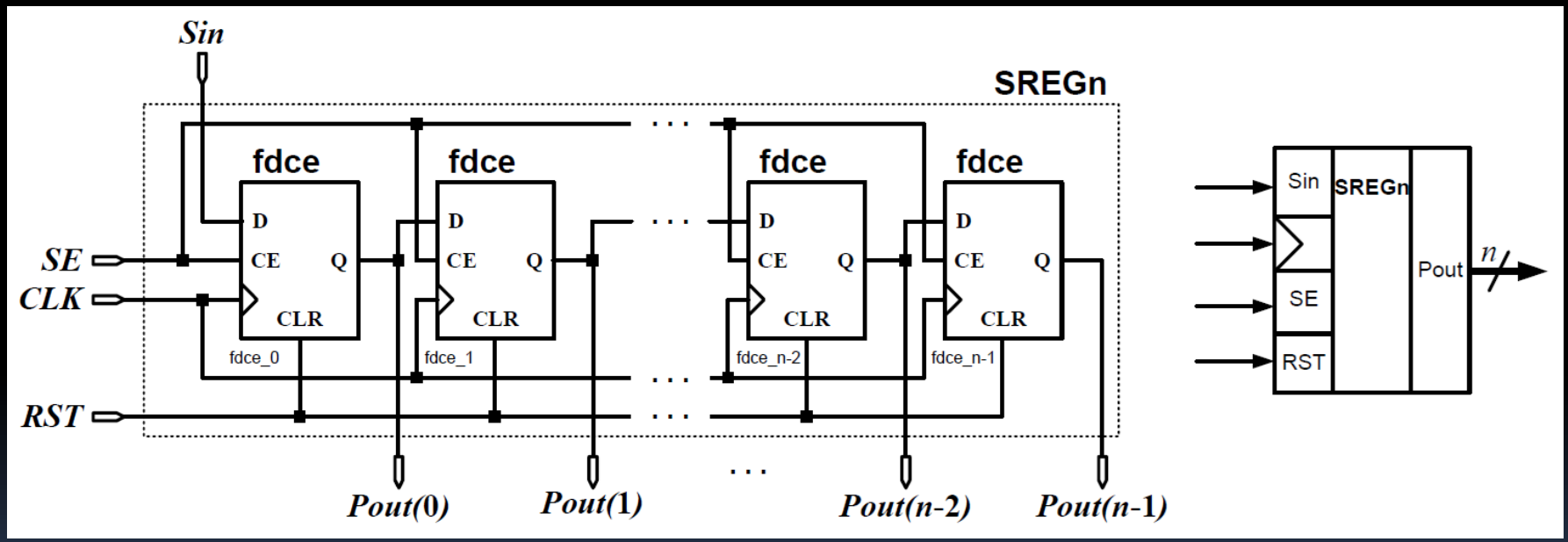
4-ый курс специальности ПОИТ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЦИФРОВОГО ПРОЕКТИРОВАНИЯ

2020

Introducing to VHDL

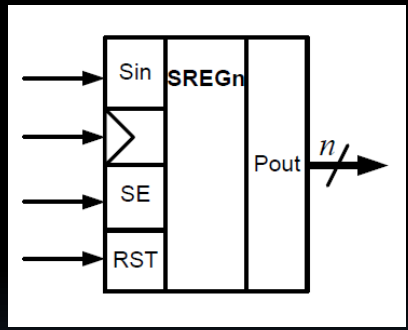
Sequential Logic: shift registers



Introducing to VHDL

Sequential Logic: shift registers :: structural description

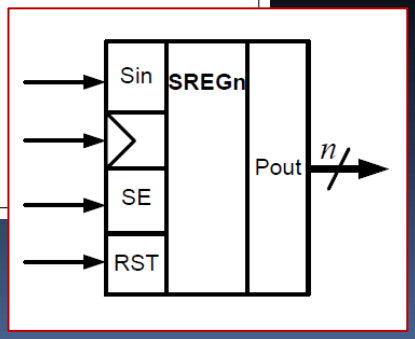
```
1  Library IEEE;
2  Use IEEE.STD_LOGIC_1164.ALL;
3  Library UNISIM;
4  Use UNISIM.vcomponents.ALL;
5
6  Entity SREGn is
7     Generic
8           ( -- число разрядов сдвигового регистра
9             n     : integer := 4 );
10    Port ( -- входной порт сигнала синхронизации
11            CLK   : in std_logic;
12            -- входной порт сигнала инициализации
13            RST   : in std_logic;
14            -- входной порт сигнала разрешения сдвига
15            SE    : in std_logic;
16            -- входной порт последовательного кода
17            Sin   : in std_logic;
18            -- выходной порт параллельного кода
19            Pout : out std_logic_vector( 0 to n-1)
20       );
21 End SREGn;
22
```



Introducing to VHDL

Sequential Logic: shift registers :: structural description

```
23  Architecture Beh of SREGn is
24  -- множество межсоединительных линий
25  signal sreg: std_logic_vector(0 to n-1);
26
27  Begin
28
29    -- триггер с индексом 0
30    FDFF: FDCE port map ( sreg( 0 ), CLK, SE, RST, Sin );
31
32    -- множество всех остальных n-1 триггеров
33    DFFs: for i in 1 to n-1 generate
34      DFFi: FDCE port map ( sreg( i ), CLK, SE, RST, sreg( i - 1 ) );
35    end generate;
36
37    -- передача значений сигналов с выходов всех триггеров
38    -- на выходной порт Pout
39    Pout <= sreg;
40
41  End Beh;
```



Introducing to VHDL

Sequential Logic: shift registers :: Parallel-In Serial-Out

```
1  Entity PISOn is
2     Generic
3           ( -- число разрядов сдвигового регистра
4             n    : integer := 4 );
5      Port ( -- входной порт сигнала синхронизации
6             CLK  : in  std_logic;
7             -- входной порт сигнала инициализации
8             RST  : in  std_logic;
9             -- входной порт сигнала управления
10            -- LS='0' - загрузка параллельного кода в регистр
11            -- LS='1' - последовательный сдвиг
12            LS   : in  std_logic;
13            -- входной порт параллельного кода
14            Pin  : in  std_logic_vector( 0 to n-1 );
15            -- выходной порт последовательного кода
16            Sout : out std_logic
17       );
18 End PISOn;
19
20 Architecture Beh of PISOn is
21 -- n-разрядный сдвиговый регистр
22 signal sreg: std_logic_vector( 0 to n-1 );
23 -- внутренняя шина данных
24 signal sdat: std_logic_vector( 0 to n-1 );
25 Begin
26
```

Introducing to VHDL

Sequential Logic: shift registers :: Parallel-In Serial-Out

```
27    -- последовательностная схема устройства
28    Main: process( CLK, RST, sdat )
29    begin
30      -- условие асинхронной инициализации регистра
31      if RST = '1' then
32        sreg <= ( others => '0' );
33      -- условие наступления переднего фронта
34      -- сигнала синхронизации
35      elsif rising_edge( CLK ) then
36        -- синхронное управление регистром
37        sreg <= sdat;
38      end if;
39    end process;
40
```

Introducing to VHDL

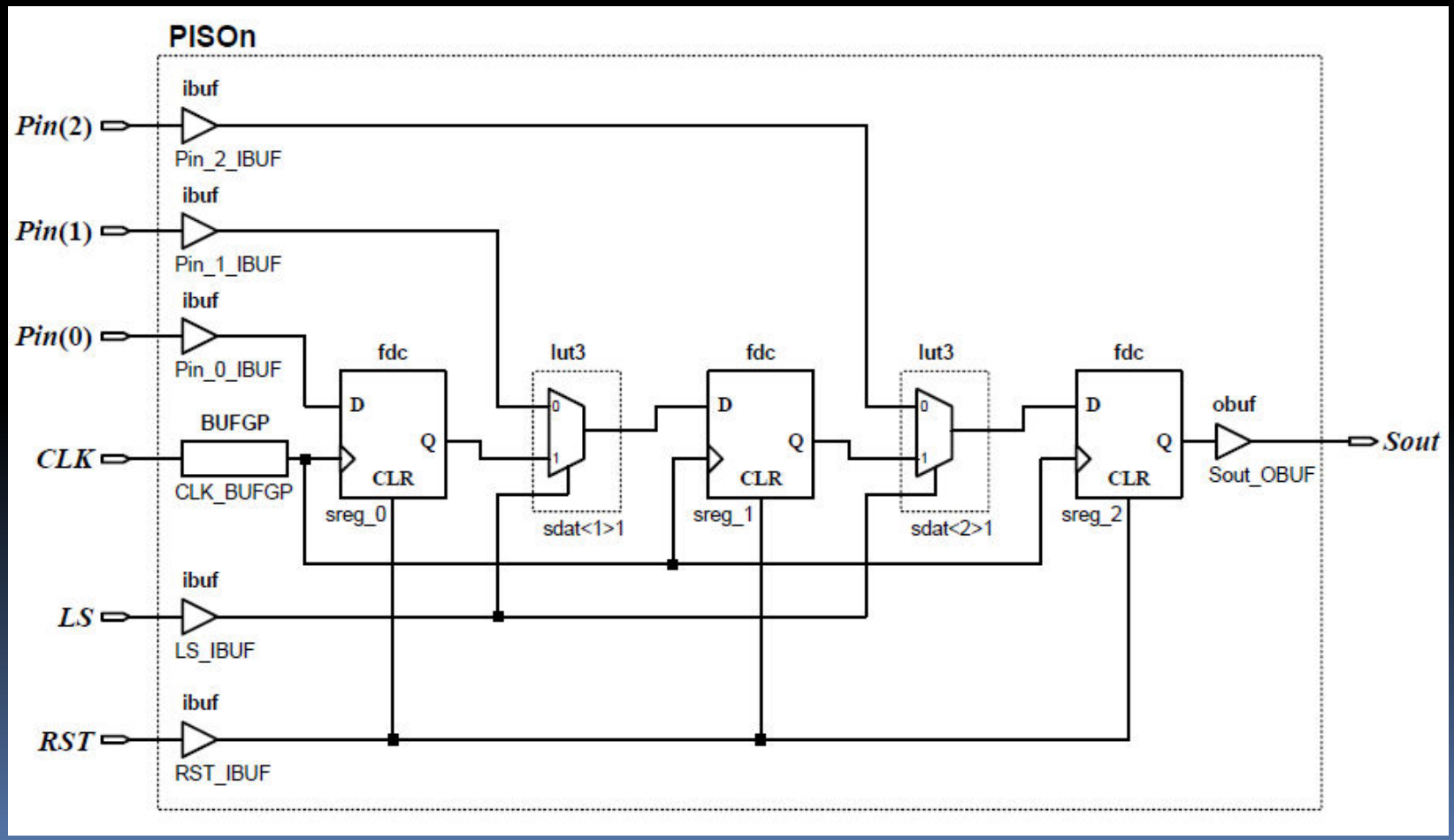
Sequential Logic: shift registers :: Parallel-In Serial-Out

```
41    -- комбинационная схема устройства
42    Data: process( LS, Pin, sreg )
43    begin
44      -- условие загрузки параллельных данных
45      if LS = '0' then
46        sdat <= Pin;
47      -- условие сдвига регистра на один разряд вправо
48      else
49        sdat <= Pin( 0 ) & sreg( 0 to n-2 );
50      end if;
51    end process;
52
53    -- передача на выходной порт последовательного кода
54    -- из старшего разряда сдвигового регистра
55    Sout <= sreg( n-1 );
56
57  End Beh;
```

Introducing to VHDL

Sequential Logic: shift registers :: Parallel-In Serial-Out

Result of Technological Synthesis (n=3)



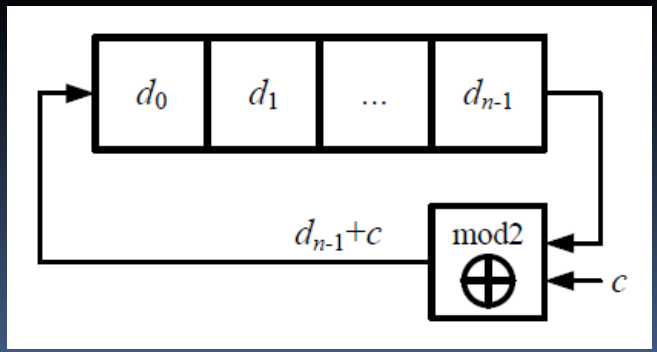
Introducing to VHDL

Sequential Logic: shift registers :: linear feedback

```
-- ...
    -- условие загрузки параллельных данных
    if LS = '0' then
      sdat <= Pin;
    -- условие сдвига регистра на один разряд вправо
    else
      sdat <= sreg( n-1 ) & sreg( 0 to n-2 );
    end if;
-- ...
```

LFSR – Linear Feedback Shift Register

Simplest LFSR



Introducing to VHDL

Sequential Logic: shift registers :: LFSR

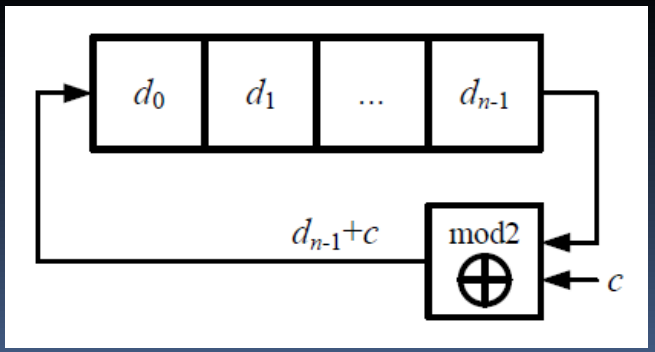
$$D = (d_0, d_1, ..., d_{n-1})$$

$$d_i = d_{i-1}, \ \forall\ i \in \{1, n-1\} \qquad d_0 = d_{n-1} + c.$$

$$\begin{vmatrix} d_0^{(k)} \\ d_1^{(k)} \\ \dots \\ d_{n-1}^{(k)} \end{vmatrix} = \begin{vmatrix} 0 & 0 & \dots & 1 \\ 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{vmatrix} \times \left(\begin{vmatrix} d_0^{(k-1)} \\ d_1^{(k-1)} \\ \dots \\ d_{n-1}^{(k-1)} \end{vmatrix} + \begin{vmatrix} 0 \\ 0 \\ \dots \\ c \end{vmatrix} \right)$$

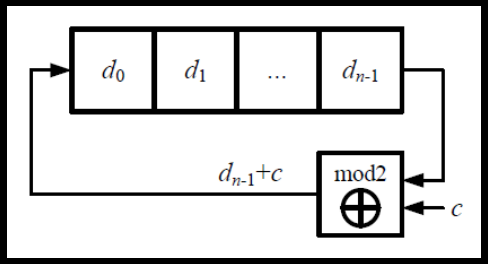
$$D^{(k)} = V(D^{(k-1)} + C)$$

$$V = \begin{vmatrix} 0 & 0 & \dots & 1 \\ 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{vmatrix}, C^T = \begin{vmatrix} 0 \\ 0 \\ \dots \\ c \end{vmatrix}.$$

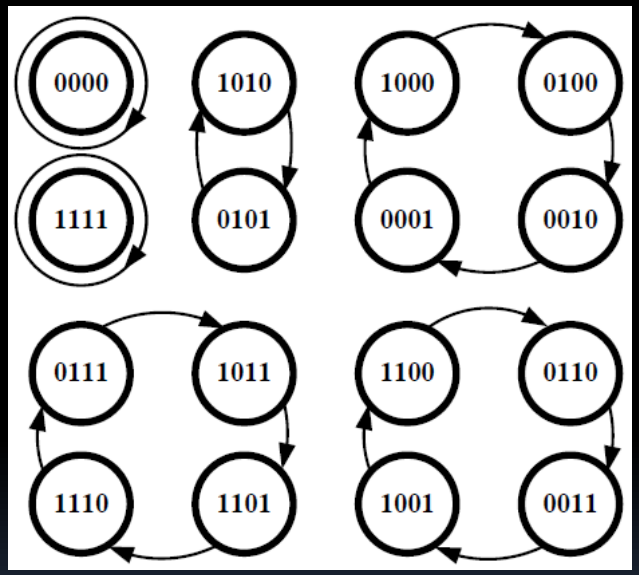


Introducing to VHDL

Sequential Logic: shift registers :: LFSR

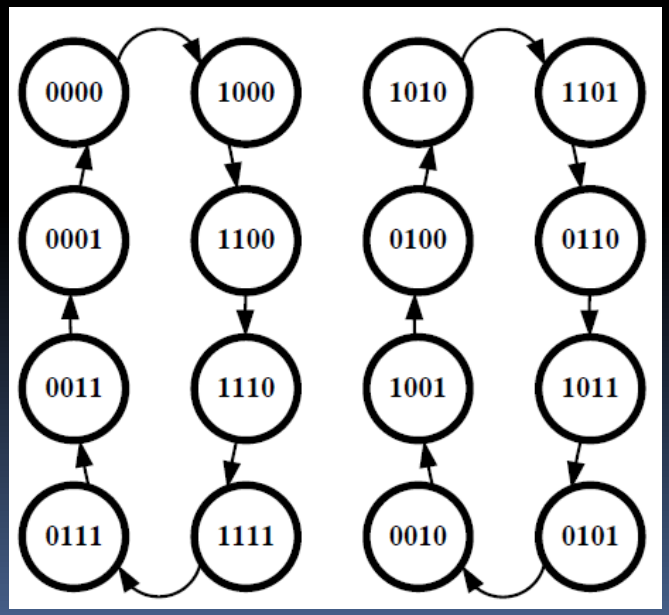


Orbits (n=4, C='0')



Cyclic Sequence Generator

Orbits (n=4, C='1')



Johnson Counter

Introducing to VHDL

Sequential Logic: shift registers :: LFSR :: one-hot sequence

```
1  Entity OneHot_n is
2     Generic
3           ( -- число разрядов генератора
4             n     : integer := 4 );
5      Port ( -- входной порт сигнала синхронизации
6             CLK   : in  std_logic;
7             -- входной потр сигнала инициализации
8             RST   : in  std_logic;
9             -- выходной порт one-hot кода
10            Pout  : out  std_logic_vector( 0 to n-1 )
11       );
12 End OneHot_n;
13
14 Architecture Beh of OneHot_n is
15 -- n-разрядный сдвиговый регистр
16 signal sreg: std_logic_vector( 0 to n-1 );
17 -- внутренняя шина данных
18 signal sdat: std_logic_vector( 0 to n-1 );
19 Begin
20
21   -- последовательностная схема устройства
22   Main: process( CLK, RST, sdat )
```

Introducing to VHDL

Sequential Logic: shift registers :: LFSR :: one-hot sequence

```
23    begin
24      -- условие асинхронной инициализации генератора
25      if RST = '1' then
26        -- начальное состояние генератора (10...0)
27        sreg( 0 ) <= '1';
28        sreg( 1 to n-1 ) <= ( others => '0' );
29      -- условие наступления переднего фронта
30      -- сигнала синхронизации
31      elsif rising_edge( CLK ) then
32        -- синхронное управление генератором
33        sreg <= sdat;
34      end if;
35    end process;
36
37    -- комбинационная схема генератора
38    sdat <= sreg( n-1 ) & sreg( 0 to n-2 );
39
40    -- передача на выходной порт
41    -- символа one-hot последовательности
42    Pout <= sdat;
43
44  End Beh;
```

Introducing to VHDL

Sequential Logic: shift registers :: LFSR :: JC

```
1  Entity JC_n is
2     Generic
3           ( -- число разрядов генератора
4             -- n=2**i
5             i     : integer := 2 );
6      Port ( -- входной порт сигнала синхронизации
7             CLK   : in  std_logic;
8             -- входной порт сигнала инициализации
9             RST   : in  std_logic;
10            -- входной порт сигнала управления
11            -- LS='0' - загрузка начального состояния
12            -- LS='1' - разрешение генерирования
13            LS    : in  std_logic;
14            -- входной порт начального состояния
15            Pin   : in  std_logic_vector( 0 to 2**i-1 );
16            -- выходной порт символов последовательности Джонсона
17            Pout : out  std_logic_vector( 0 to 2**i-1 )
18       );
19 End JC_n;
20
21 Architecture Beh of JC_n is
22 -- n-разрядный сдвиговый регистр
23 signal sreg: std_logic_vector( 0 to 2**i-1 );
24 -- внутренняя шина данных
25 signal sdat: std_logic_vector( 0 to 2**i-1 );
26 Begin
27
```

Introducing to VHDL

Sequential Logic: shift registers :: LFSR :: JC

```
28    -- последовательностная схема устройства
29    Main: process( CLK, RST, sdat )
30    begin
31      -- условие асинхронной инициализации генератора
32      if RST = '1' then
33        -- начальное состояние генератора (00...0)
34        sreg <= ( others => '0' );
35      -- условие наступления переднего фронта
36      -- сигнала синхронизации
37      elsif rising_edge( CLK ) then
38        -- синхронное управление генератором
39        sreg <= sdat;
40      end if;
41    end process;
42
```

Introducing to VHDL

Sequential Logic: shift registers :: LFSR :: JC

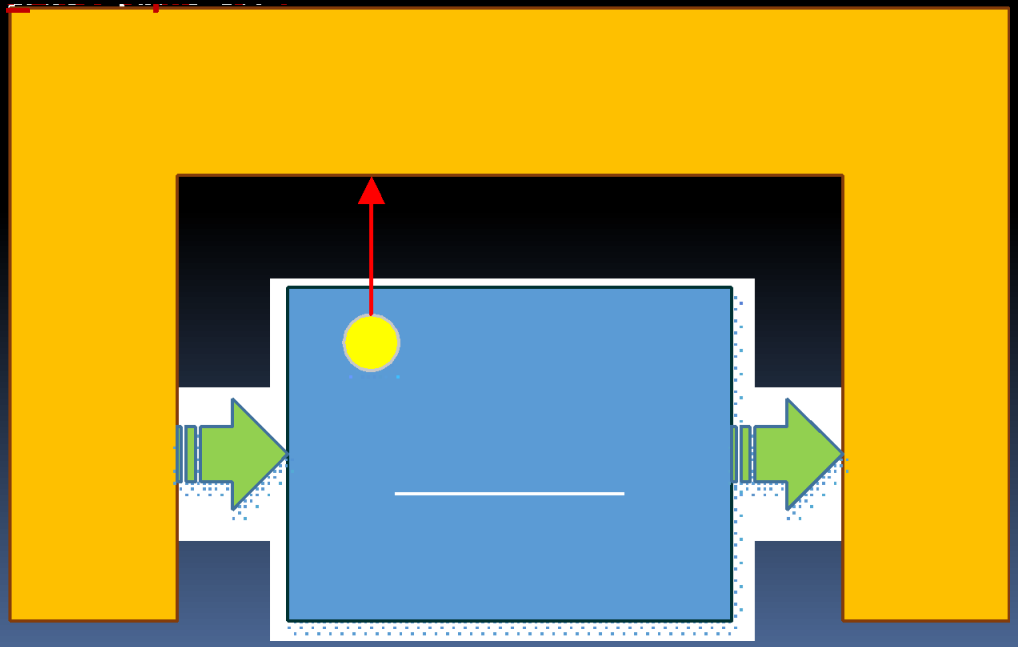
```
43    -- комбинационная схема генератора
44    Data: process( LS, Pin, sreg )
45    begin
46    -- условие загрузки начального состояния
47      if LS = '0' then
48        sdat <= Pin;
49      -- условие генерирования символов последовательности Джонсона
50      else
51        sdat <= not(sreg( 2**i-1 )) & sreg( 0 to 2**i-2 );
52      end if;
53    end process;
54
55    -- передача на выходной порт
56    -- символа последовательности Джонсона
57    Pout <= sdat;
58
59  End Beh;
```

Introducing to VHDL

Sequential Logic: shift registers :: LFSR :: JC

How to test?

1. Manually (Waveform Analysis)
2. TestBench Component



Introducing to VHDL

Sequential Logic: shift registers :: LFSR :: JC_TB

```
1  -- VHDL Test Bench Created by ISE for module: JC_n
2  --
3  -- Dependencies:
4  --
5  -- Revision:
6  -- Revision 0.01 - File Created
7  -- Additional Comments:
8  --
9  -- Notes:
10 -- This testbench has been automatically generated using types std_logic and
11 -- std_logic_vector for the ports of the unit under test.  Xilinx recommends
12 -- that these types always be used for the top-level I/O of a design in order
13 -- to guarantee that the testbench will bind correctly to the post-implementation
14 -- simulation model.
15 --------------------------------------------------------------------------------
16 LIBRARY ieee;
17 USE ieee.std_logic_1164.ALL;
18
19 ENTITY JC_TB IS
20 END JC_TB;
21
22 ARCHITECTURE behavior OF JC_TB IS
23
24     -- Component Declaration for the Unit Under Test (UUT)
25
26     COMPONENT JC_n
27     PORT(
28          CLK  : IN   std_logic;
29          RST  : IN   std_logic;
30          LS   : IN   std_logic;
31          Pin  : IN   std_logic_vector(0 to 3);
32          Pout : OUT  std_logic_vector(0 to 3)
33        );
34     END COMPONENT;
35
```

Introducing to VHDL

Sequential Logic: shift registers :: LFSR :: JC_TB

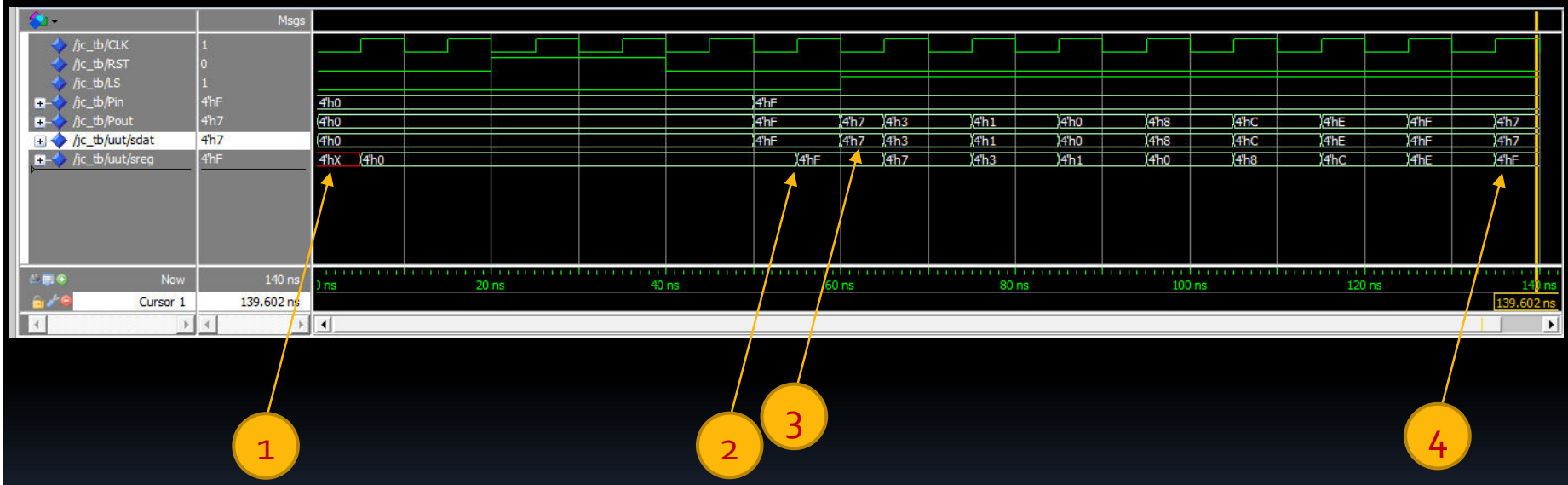
```
36
37    --Inputs (initial values)
38    signal CLK : std_logic := '0';
39    signal RST : std_logic := '0';
40    signal LS  : std_logic := '0';
41    signal Pin : std_logic_vector(0 to 3) := (others => '0');
42
43    --Outputs
44    signal Pout : std_logic_vector(0 to 3);
45
46    -- Clock period definitions
47    constant CLK_period : time := 10 ns;
48
49  BEGIN
50
51    -- Instantiate the Unit Under Test (UUT)
52    uut: JC_n PORT MAP (
53           CLK  => CLK,
54           RST  => RST,
55           LS   => LS,
56           Pin  => Pin,
57           Pout => Pout
58         );
59
60    -- Clock process definitions
61    CLK_process :process
62    begin
63       CLK <= '0';
64       wait for CLK_period/2;
65       CLK <= '1';
66       wait for CLK_period/2;
67    end process;
```

```
68
69
70    -- Stimulus process
71    stim_proc: process
72    begin
73       wait for CLK_period;
74
75       -- activate reset
76       RST <= '0'; wait for CLK_period;
77       RST <= '1'; wait for 2*CLK_period;
78       RST <= '0'; wait for CLK_period;
79
80       -- load initial seed
81       Pin <= "1111"; wait for CLK_period;
82
83       -- start to count
84       LS  <= '1'; wait for 8*CLK_period;
85
86       wait;
87    end process;
88
89  END;
90
```

Introducing to VHDL

Sequential Logic: shift registers :: LFSR :: JC_TB

ModelSim screenshot:



1 – force unknown state of SREG
2 – synchronous load
3 – half period state (not good!)
4 – looks like JC orbit (F-7-3-1-0-8-C-E-F)