Hardware Modeling [VU] (191.011)
– WS25 –

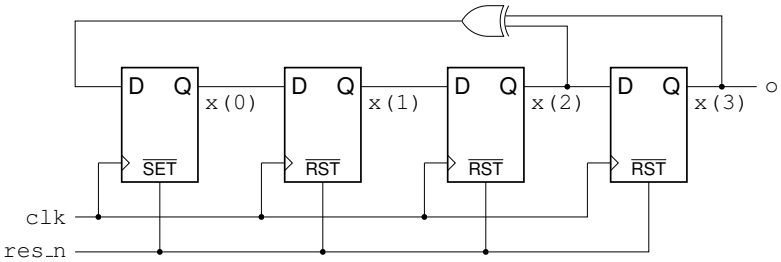
Sequential Circuit Example: LFSR

Florian Huemer & Sebastian Wiedemann & Dylan Baumann

WS 2025/26

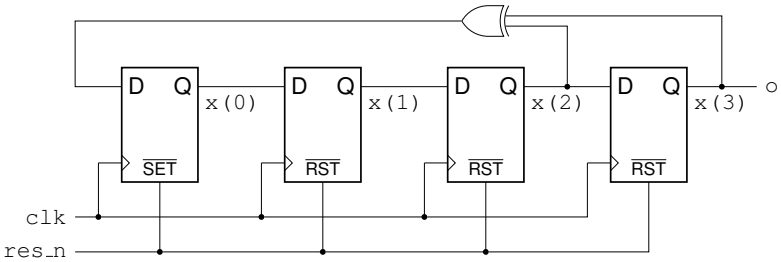
Modified: 2025-12-16, 16:04 (f8a58e9)

Example: 4-bit LFSR



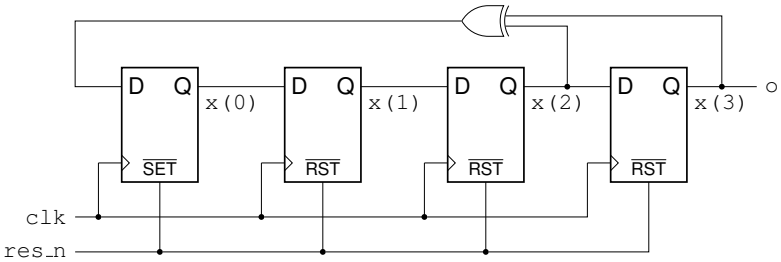
- **L**inear **F**eedback **S**hift **R**egister

Example: 4-bit LFSR



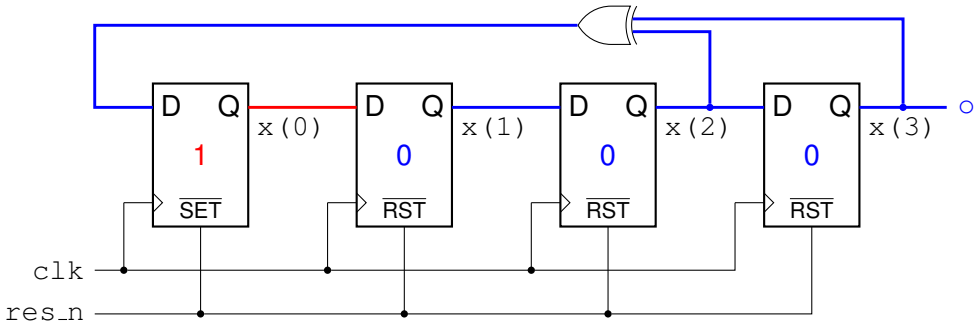
- **L**inear **F**eedback **S**hift **R**egister
- Chain of FFs (*shift-register*) fed by linear combination of its current state

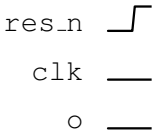
Example: 4-bit LFSR



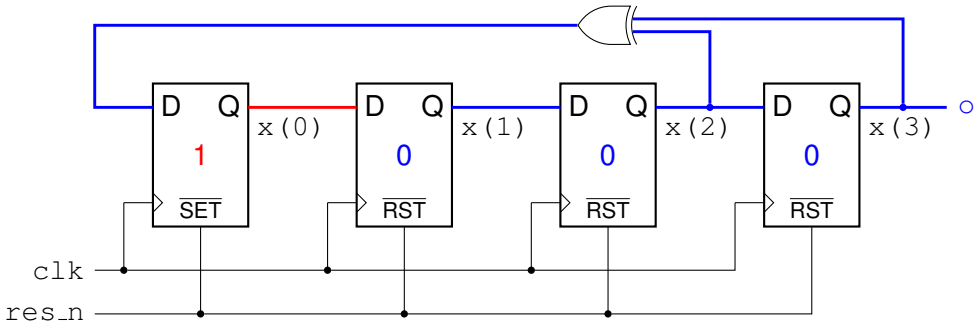
- **L**inear **F**eedback **S**hift **R**egister
- Chain of FFs (*shift-register*) fed by linear combination of its current state
- Pseudo-random sequence of bits

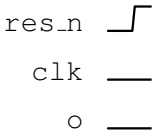
LFSR - Circuit Operation Principle



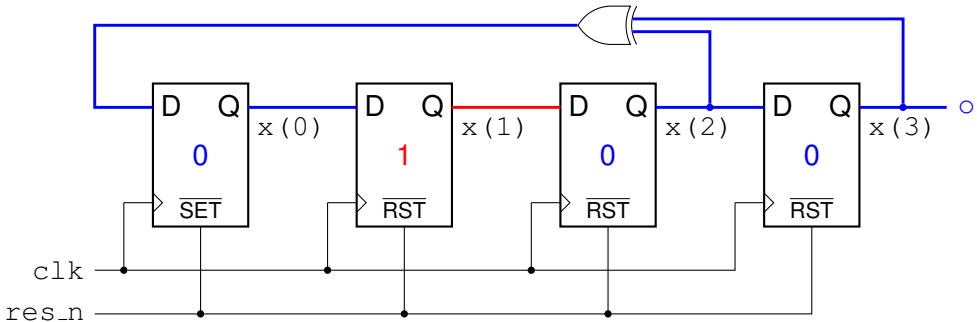


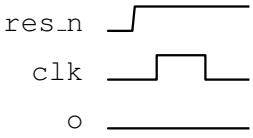
LFSR - Circuit Operation Principle



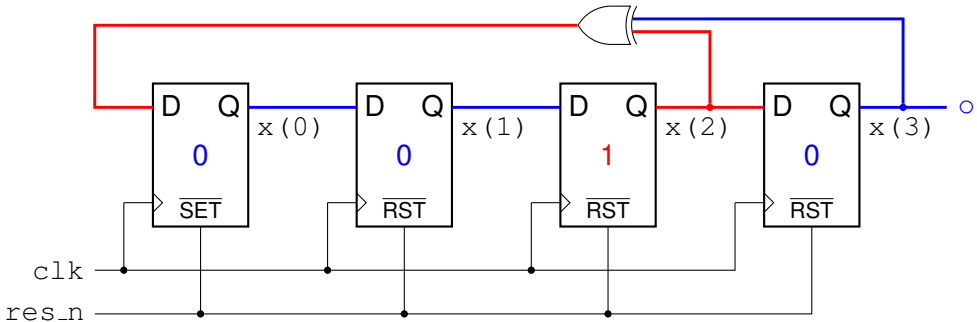


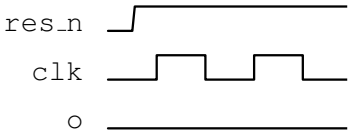
LFSR - Circuit Operation Principle



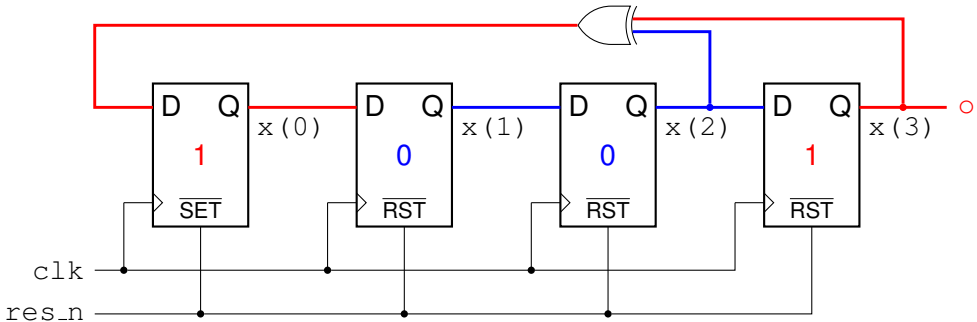


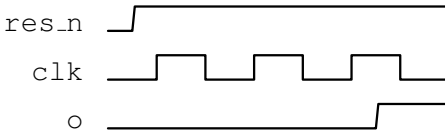
LFSR - Circuit Operation Principle



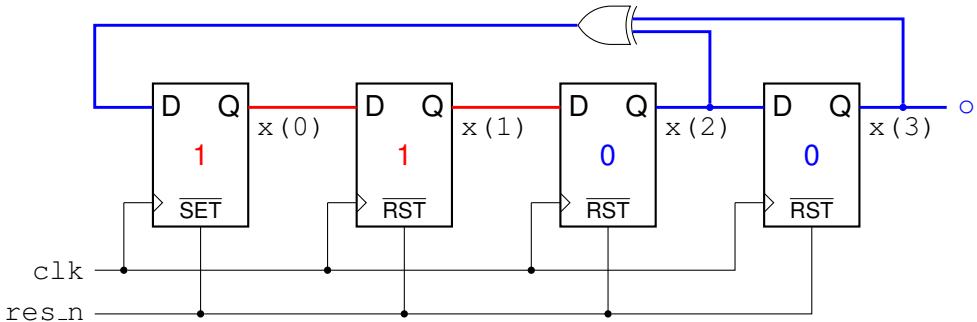


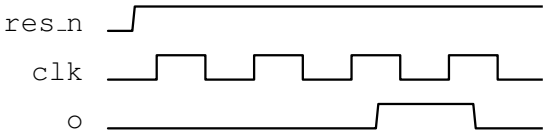
LFSR - Circuit Operation Principle



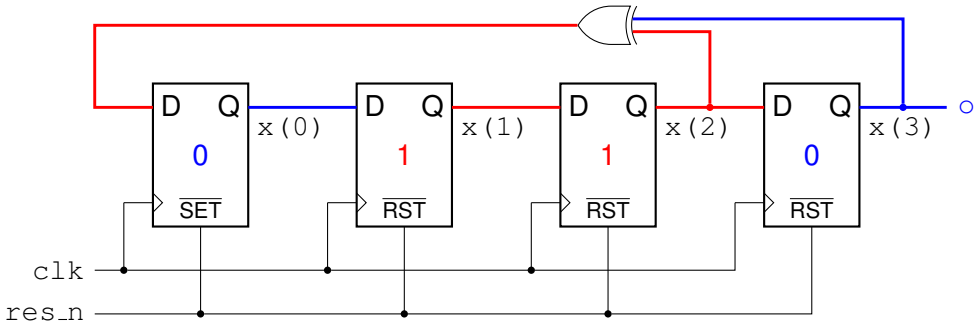


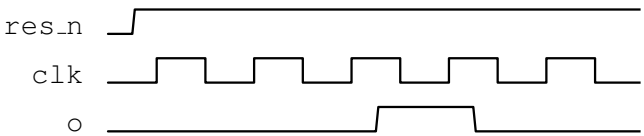
LFSR - Circuit Operation Principle



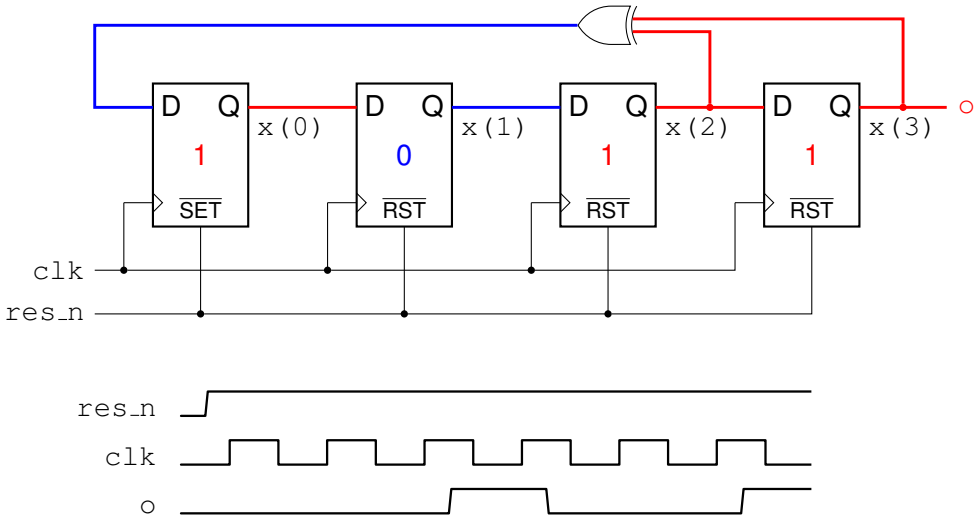


LFSR - Circuit Operation Principle

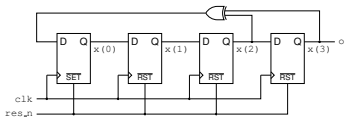




LFSR - Circuit Operation Principle

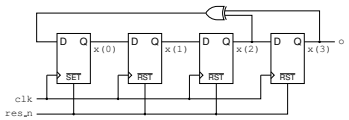


LFSR - VHDL Design




```vhdl
entity lfsr is
  port (
    clk   : in  std_ulogic;
    res_n : in  std_ulogic;
    o     : out std_ulogic
  );
end entity;
```

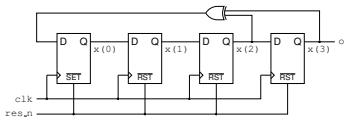
LFSR - VHDL Design




```vhdl
entity lfsr is
  port (
    clk   : in  std_ulogic;
    res_n : in  std_ulogic;
    o     : out std_ulogic
  );
end entity;
```

```vhdl
architecture arch of lfsr is
  signal x : std_ulogic_vector(0 to 3);
begin









end architecture;
```

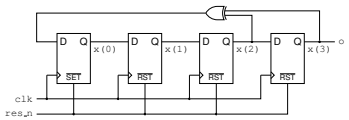
LFSR - VHDL Design




```vhdl
entity lfsr is
  port (
    clk   : in  std_ulogic;
    res_n : in  std_ulogic;
    o     : out std_ulogic
  );
end entity;
```

```vhdl
architecture arch of lfsr is
  signal x : std_ulogic_vector(0 to 3);
begin









end architecture;
```

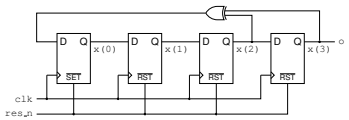
LFSR - VHDL Design




```vhdl
entity lfsr is
  port (
    clk   : in  std_ulogic;
    res_n : in  std_ulogic;
    o     : out std_ulogic
  );
end entity;
```

```vhdl
architecture arch of lfsr is
  signal x : std_ulogic_vector(0 to 3);
begin
  sync : process(clk, res_n)
  begin
    if res_n = '0' then

    elsif rising_edge(clk) then









    end if;
  end process;

end architecture;
```

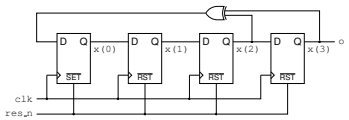
LFSR - VHDL Design




```vhdl
entity lfsr is
  port (
    clk   : in  std_ulogic;
    res_n : in  std_ulogic;
    o     : out std_ulogic
  );
end entity;
```

```vhdl
architecture arch of lfsr is
  signal x : std_ulogic_vector(0 to 3);
begin
  sync : process(clk, res_n)
  begin
    if res_n = '0' then

    elsif rising_edge(clk) then









    end if;
  end process;

end architecture;
```

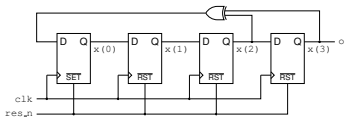
LFSR - VHDL Design




```vhdl
entity lfsr is
  port (
    clk   : in  std_ulogic;
    res_n : in  std_ulogic;
    o     : out std_ulogic
  );
end entity;
```

```vhdl
architecture arch of lfsr is
  signal x : std_ulogic_vector(0 to 3);
begin
  sync : process(clk, res_n)
  begin
    if res_n = '0' then
      x <= (0 => '1', others => '0');
    elsif rising_edge(clk) then




    end if;
  end process;

end architecture;
```

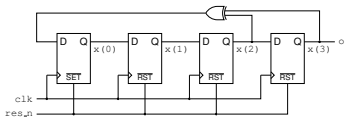
LFSR - VHDL Design




```vhdl
entity lfsr is
  port (
    clk   : in  std_ulogic;
    res_n : in  std_ulogic;
    o     : out std_ulogic
  );
end entity;
```

```vhdl
architecture arch of lfsr is
  signal x : std_ulogic_vector(0 to 3);
begin
  sync : process(clk, res_n)
  begin
    if res_n = '0' then
      x <= (0 => '1', others => '0');
    elsif rising_edge(clk) then
      x(0) <= x(2) xor x(3);
      x(1) <= x(0);
      x(2) <= x(1);
      x(3) <= x(2);
    end if;
  end process;

end architecture;
```

LFSR - VHDL Design




```vhdl
entity lfsr is
  port (
    clk   : in  std_ulogic;
    res_n : in  std_ulogic;
    o     : out std_ulogic
  );
end entity;
```

```vhdl
architecture arch of lfsr is
  signal x : std_ulogic_vector(0 to 3);
begin
  sync : process(clk, res_n)
  begin
    if res_n = '0' then
      x <= (0 => '1', others => '0');
    elsif rising_edge(clk) then
      x(0) <= x(2) xor x(3);
      x(1) <= x(0);
      x(2) <= x(1);
      x(3) <= x(2);
    end if;
  end process;
  o <= x(3);
end architecture;
```

LFSR - Testbench


```
1  architecture bench of lfsr_tb is
2    signal clk : std_ulogic;
3    signal res_n : std_ulogic;
4    signal o :  std_ulogic;
5
6
7
8  begin
9
10   uut : entity work.lfsr
11   port map (
12     clk => clk,
13     res_n => res_n,
14     o => o
15   );
```

LFSR - Testbench


```
1  architecture bench of lfsr_tb is
2    signal clk : std_ulogic;
3    signal res_n : std_ulogic;
4    signal o :  std_ulogic;
5
6
7
8  begin
9
10   uut : entity work.lfsr
11   port map (
12     clk => clk,
13     res_n => res_n,
14     o => o
15   );
```

LFSR - Testbench


```
1  architecture bench of lfsr_tb is
2    signal clk : std_ulogic;
3    signal res_n : std_ulogic;
4    signal o :  std_ulogic;
5
6    constant CLK_PERIOD : time := 10 ns;
7    signal stop_clk : boolean := false;
8  begin
9
10   uut : entity work.lfsr
11   port map (
12     clk => clk,
13     res_n => res_n,
14     o => o
15   );
```

LFSR - Testbench


```
1  architecture bench of lfsr_tb is
2    signal clk : std_ulogic;
3    signal res_n : std_ulogic;
4    signal o :  std_ulogic;
5
6    constant CLK_PERIOD : time := 10 ns;
7    signal stop_clk : boolean := false;
8  begin
9
10   uut : entity work.lfsr
11   port map (
12     clk => clk,
13     res_n => res_n,
14     o => o
15   );
```

LFSR - Testbench


```
1  architecture bench of lfsr_tb is
2    signal clk : std_ulogic;
3    signal res_n : std_ulogic;
4    signal o :  std_ulogic;
5
6    constant CLK_PERIOD : time := 10 ns;
7    signal stop_clk : boolean := false;
8  begin
9
10   uut : entity work.lfsr
11   port map (
12     clk => clk,
13     res_n => res_n,
14     o => o
15   );
```

LFSR - Testbench


```vhdl
architecture bench of lfsr_tb is
  signal clk : std_ulogic;
  signal res_n : std_ulogic;
  signal o :  std_ulogic;

  constant CLK_PERIOD : time := 10 ns;
  signal stop_clk : boolean := false;
begin

  uut : entity work.lfsr
  port map (
    clk => clk,
    res_n => res_n,
    o => o
  );
```

```vhdl
clkgen : process
begin
  while not stop_clk loop
    clk <= '0';
    wait for 0.5*CLK_PERIOD;
    clk <= '1';
    wait for 0.5*CLK_PERIOD;
  end loop;
  wait;
end process;
```

LFSR - Testbench


```
1  architecture bench of lfsr_tb is
2    signal clk : std_ulogic;
3    signal res_n : std_ulogic;
4    signal o :  std_ulogic;
5
6    constant CLK_PERIOD : time := 10 ns;
7    signal stop_clk : boolean := false;
8  begin
9
10   uut : entity work.lfsr
11   port map (
12     clk => clk,
13     res_n => res_n,
14     o => o
15   );
```

```
1   clkgen : process
2   begin
3     while not stop_clk loop
4       clk <= '0';
5       wait for 0.5*CLK_PERIOD;
6       clk <= '1';
7       wait for 0.5*CLK_PERIOD;
8     end loop;
9     wait;
10  end process;
```

LFSR - Testbench


```
1  architecture bench of lfsr_tb is
2    signal clk : std_ulogic;
3    signal res_n : std_ulogic;
4    signal o :  std_ulogic;
5
6    constant CLK_PERIOD : time := 10 ns;
7    signal stop_clk : boolean := false;
8  begin
9
10   uut : entity work.lfsr
11   port map (
12     clk => clk,
13     res_n => res_n,
14     o => o
15   );
```

```
1   clkgen : process
2   begin
3     while not stop_clk loop
4       clk <= '0';
5       wait for 0.5*CLK_PERIOD;
6       clk <= '1';
7       wait for 0.5*CLK_PERIOD;
8     end loop;
9     wait;
10  end process;
```

LFSR - Testbench


```vhdl
architecture bench of lfsr_tb is
  signal clk : std_ulogic;
  signal res_n : std_ulogic;
  signal o :  std_ulogic;

  constant CLK_PERIOD : time := 10 ns;
  signal stop_clk : boolean := false;
begin

  uut : entity work.lfsr
  port map (
    clk => clk,
    res_n => res_n,
    o => o
  );
```

```vhdl
clkgen : process
begin
  while not stop_clk loop
    clk <= '0';
    wait for 0.5*CLK_PERIOD;
    clk <= '1';
    wait for 0.5*CLK_PERIOD;
  end loop;
  wait;
end process;
```

LFSR - Testbench


```vhdl
architecture bench of lfsr_tb is
  signal clk : std_ulogic;
  signal res_n : std_ulogic;
  signal o :  std_ulogic;

  constant CLK_PERIOD : time := 10 ns;
  signal stop_clk : boolean := false;
begin

  uut : entity work.lfsr
  port map (
    clk => clk,
    res_n => res_n,
    o => o
  );
```

```vhdl
clkgen : process
begin
  while not stop_clk loop
    clk <= '0';
    wait for 0.5*CLK_PERIOD;
    clk <= '1';
    wait for 0.5*CLK_PERIOD;
  end loop;
  wait;
end process;

stimulus : process
begin
  res_n <= '0';
  wait until rising_edge(clk);
  wait until rising_edge(clk);
  res_n <= '1';
  wait for 6*CLK_PERIOD;
  stop_clk <= true;
  wait;
end process;
```

LFSR - Testbench


```
1  architecture bench of lfsr_tb is
2    signal clk : std_ulogic;
3    signal res_n : std_ulogic;
4    signal o :  std_ulogic;
5
6    constant CLK_PERIOD : time := 10 ns;
7    signal stop_clk : boolean := false;
8  begin
9
10   uut : entity work.lfsr
11   port map (
12     clk => clk,
13     res_n => res_n,
14     o => o
15   );
```

```
1   clkgen : process
2   begin
3     while not stop_clk loop
4       clk <= '0';
5       wait for 0.5*CLK_PERIOD;
6       clk <= '1';
7       wait for 0.5*CLK_PERIOD;
8     end loop;
9     wait;
10  end process;
11
12  stimulus : process
13  begin
14    res_n <= '0';
15    wait until rising_edge(clk);
16    wait until rising_edge(clk);
17    res_n <= '1';
18    wait for 6*CLK_PERIOD;
19    stop_clk <= true;
20    wait;
21  end process;
```

Lecture Complete!