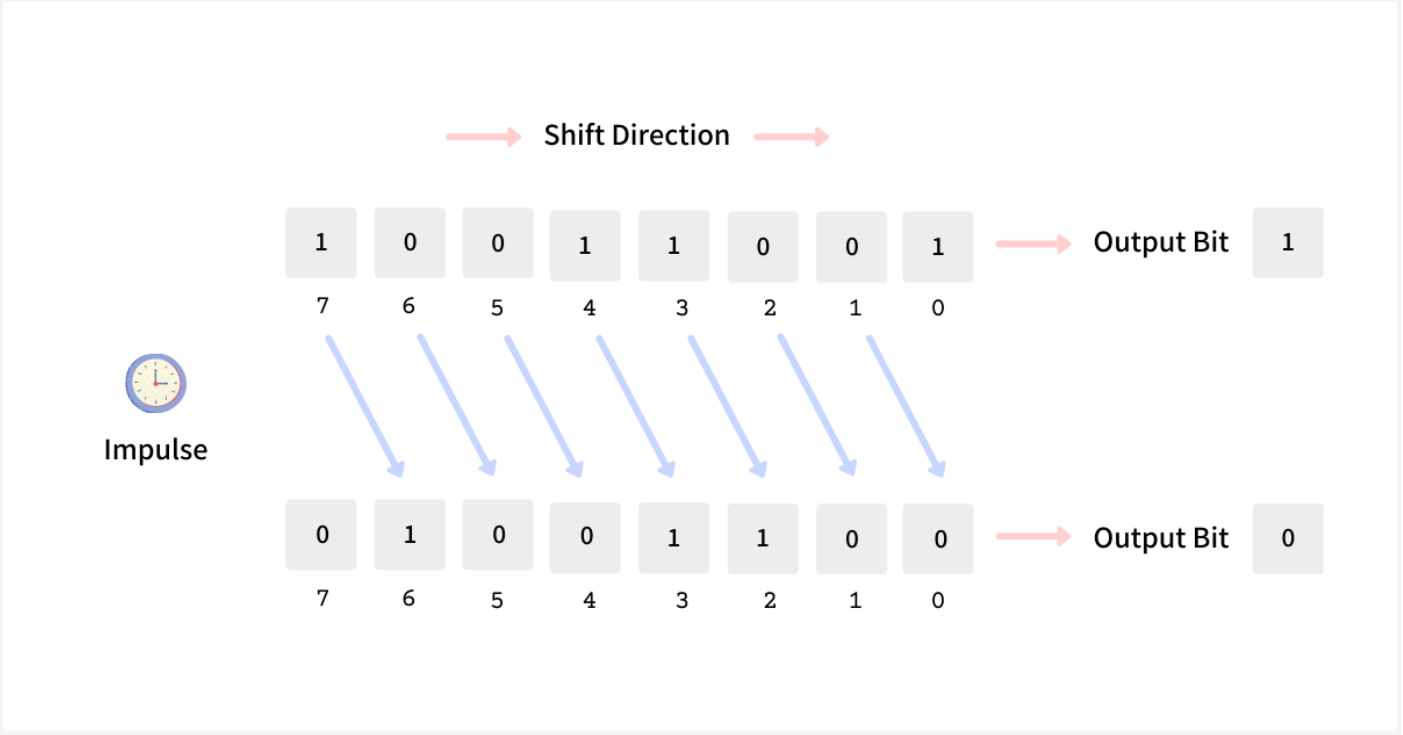
VHDL İle Rastgele Sayı Üretmek

Emir Muhammet Aydemir[1], Kadir Kartal[2]
Bilgisayar Mühendisliği Bölümü, Marmara Üniversitesi, İstanbul, Türkiye
{emiraydemir,kadirkartal}@marun.edu.tr

LFSR Nedir?

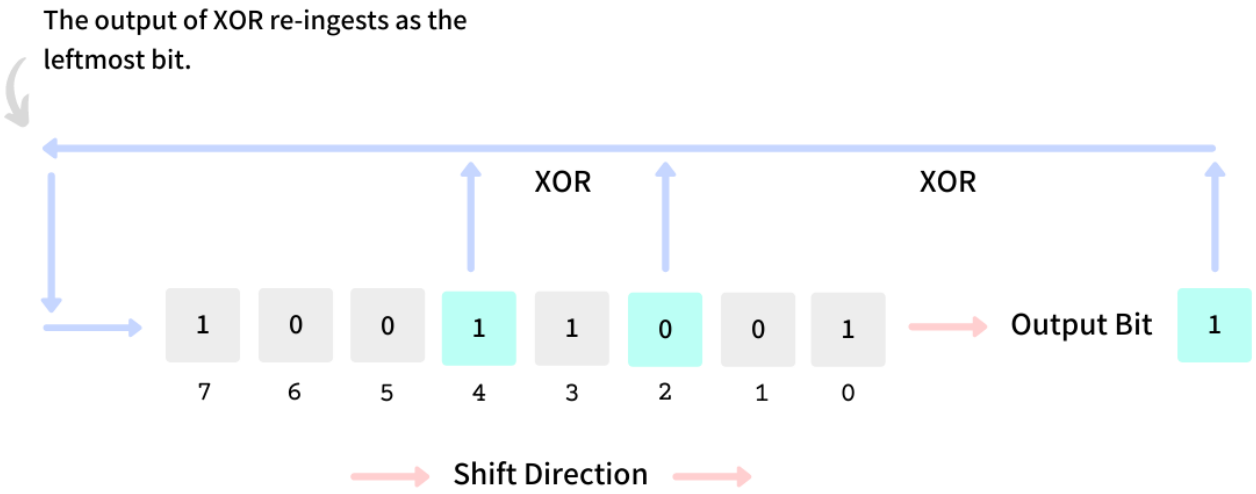
Doğrusal bir geri besleme kaydırma kaydı, tetiklendiğinde kayan bir bitler topluluğudur ve sonraki durum, önceki durumunun doğrusal bir işlevidir. Kaydırma işlemi olarak sağa kaydırma ( ) ve **kaydın** sonraki durumunu oluşturmak için doğrusal fonksiyon olarak **XOR** ( ) kullanıyoruz.



LFSR, tohum adı verilen rastgele bir değerle başlatılır. Kaydın sonraki durumu, önceki durumu ve bahsedilen işlemler kullanılarak deterministik olarak hesaplanabilir.

LFSR iş başında

Bir sonraki satıra bağlanan ve bir zincir oluşturan bir dizi mandala (bit) kayıt denir. Aşağıdaki şema, bir darbe üzerine sağa kayan 8 bitlik bir kaydırma kaydını göstermektedir. Kaydırma sırasında atılan en sağdaki bit, çıkış bitidir.



Kaydırma gerçekleştiğinde, en soldaki mandal boşalır ve ya

- dairesel bir kaydırma yazmacı oluşturan çıkış biti ile doldurulur
- bir programlama dilinin saf sağa kaydırma işlemi gibi sıfırla dolu
- mandallarda bir miktar boole mantığının sonucu ile dolu

Sahte rasgele sayılar oluşturmak için kullandığımız LFSR, üçüncü yaklaşımla gider ve bir boolean XOR'u, taps adı verilen bir dizi seçilmiş mandala uygular ve elde edilen biti en soldaki mandala koyarak bir Doğrusal Geri Besleme oluşturur.

LFSR ile ilgili endişeler

LFSR'ler hem yazılım hem de donanım tarafında çok verimli olmasına rağmen, bunların kullanımıyla ilgili bazı endişeler var.

LFSR'deki bit sayısı sınırlıdır (yapılandırıldığı gibi); kayıt, belirli bir bit seti ürettikten sonra aynı seti tekrarlayacaktır. Döngünün uzunluğu yalnızca tohum değerine ve kılavuz konfigürasyonuna bağlıdır. Bu nedenle, rastgele sayı üretimi için LFSR kullanırken, çok uzun bir döngü sağlamak için iyi bir kademe konumu ve tohum seti seçmek esastır.

LFSR'den üretilen birkaç ardışık rastgele sayı elde ettiğimizi varsayalım. Bu durumda, onları birkaç lineer denkleme koyabilir ve başlangıç konfigürasyonuna ulaşabiliriz, bu da gelecekteki rasgele sayılar kümesini tahmin etmemizi sağlar. LFSR'lerin ne kadar savunmasız olabileceği göz önüne alındığında, kriptografik güce ihtiyaç duyan yerlerde kullanılmazlar.

Proje içerisinde lfsr1.vhd dosyası ve test bench dosyası olan tb_lfsr1.vhd isimli dosya yer almaktadır. lfsr1.vhd dosyası içerisinde aşağıdaki işlemler gerçekleştirilmektedir.

1. Proje içerisinde kullanılacak kütüphaneler dahil edilmektedir.

```
library ieee;
  use ieee.std_logic_1164.all;
  use work.lfsr_pkg.all;
```

2. Reset, enable, clock, count değerleri için port tanımlamaları yapılmaktadır.

```
entity lfsr1 is
  port (
    reset  : in  std_logic;
    clk    : in  std_logic;
    en     : in  std_logic;
    count  : out std_logic_vector (LFSR_W-1 downto 0)
  );
end entity;
```

3. Kullanılacak polinom ifadesi ve dizinin tanımlanması yapılmaktadır.

```
architecture Behavioral of lfsr1 is
    signal count_i    : std_logic_vector (LFSR_W-1 downto 0);
    signal feedback     : std_logic:= '0';

begin
```

4. Tanımlanan polinom ifadesine 11 bitlik lfsr algoritması için en uygun değerin ataması yapılmaktadır.

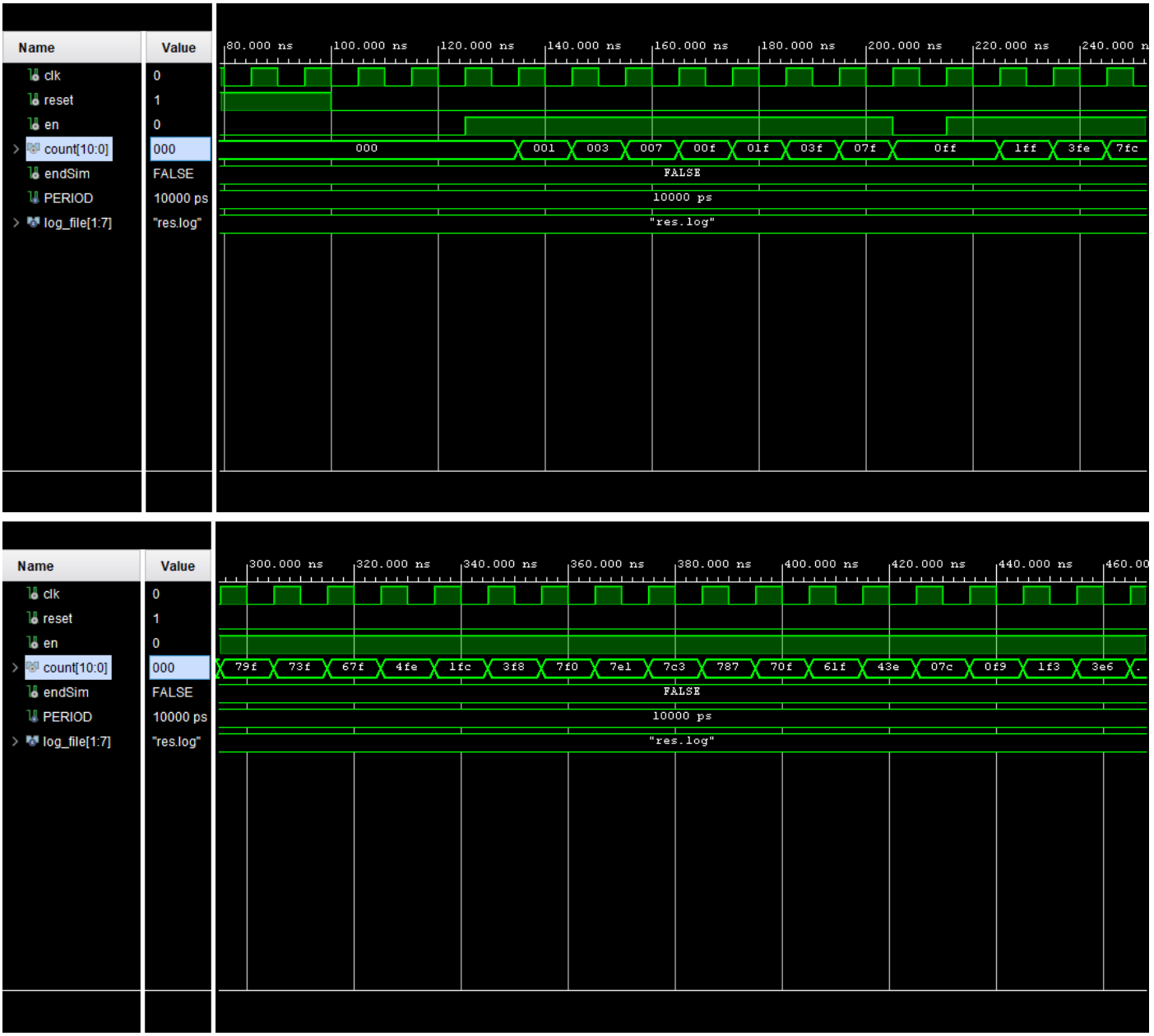
```
feedback <= not(count_i(LFSR_W-1) xor count_i(LFSR_W-3));
```

5. Son olarak da clock işlemini başlatıyoruz. Clock işleminde yükselen kenarda ise ve reset değişkeni 1 ise count değişkenine 0 değerini atıyoruz. Reset değişkeninin değeri 0 ise ve enable değişkeninin değeri 1 ise count değişkenine rastgele sayımızı atıyoruz.

```
  sr_pr : process (clk)
    begin
      if (rising_edge(clk)) then
        if (reset = '1') then
          count_i <= (others=>'0');
        elsif (en = '1') then
          count_i <= count_i(LFSR_W-2 downto 0) & feedback;
        end if;
      end if;
    end process sr_pr;
  count <= count_i;

end architecture;
```

Yukarıdaki sıralı kod blokları çalıştırıldığında ise aşağıdaki simülasyon çıktısı ile karşılaşmaktayız.



tb_lfsr1.vhd dosyası içerisinde ise aşağıdaki işlemler gerçekleştirilmektedir.

1. Proje içerisinde kullanılacak kütüphaneler dahil edilmektedir.

```
library ieee;
  use ieee.std_logic_1164.all;
  use ieee.std_logic_textio.all;
  use ieee.numeric_std.ALL;
  use std.textio.all;
  use work.lfsr_pkg.all;

entity tb_lfsr1 is
end entity;
```

2. Clock işleminin periyotları 10 nanosaniye olarak tanımlanmaktadır. Oluşturulan rastgele sayıların loglarının tutulacağı dosya isminin tanımlanması yapılmaktadır.,
3. Clock, reset, enable, count değişkenlerinin ve simülasyonun ne zaman biteceğini belirleyen endSim değişkeninin default değerleri atanmaktadır.
4. Reset, enable, clock, count değerleri için port tanımlamaları yapılmaktadır.

```
architecture test of tb_lfsr1 is

    constant PERIOD  : time   := 10 ns;
    constant log_file: string := "res.log";

    signal clk       : std_logic := '0';
    signal reset     : std_logic := '1';
    signal en        : std_logic := '0';
    signal count     : std_logic_vector (LFSR_W-1 downto 0);
    signal endSim      : boolean   := false;

    component lfsr1 is
      port (
        reset     : in  std_logic;
        clk       : in  std_logic;
        en        : in  std_logic;
        count     : out std_logic_vector (LFSR_W-1 downto 0)
      );
    end component;
```

5. Yükselen ve alçalan kenar için periyodun yarısında bir saat darbesi üretiyoruz. Sonrasında 10 periyotta bir reset değerini 0'a çekiyoruz reset 0 ve enable 1 iken rastgele sayı üretiliyor

```
begin
  clk     <= not clk after PERIOD/2;
  reset   <= '0' after  PERIOD*10;

    -- Main simulation process
    main_pr : process
    begin
        wait until (reset = '0');
        wait until (clk = '1');
        wait until (clk = '1');
        wait until (clk = '1');
        en <= '1';
        for i in 0 to 7 loop
          wait until (clk = '1');
        end loop;
        en <= '0';
        wait until (clk = '1');
        en <= '1';
        while (not endSim) loop
      wait until (clk = '1');
    end loop;
    end process main_pr;
```

```
stop_pr : process
begin
    if (endSim) then
        assert false
            report "End of simulation."
            severity failure;
    end if;
    wait until (clk = '1');
end process stop_pr;

DUT : lfsr1
  port map (
    clk      => clk,
    reset    => reset,
    en       => en,
    count    => count
  );
```

6. Son adım olarak oluşturulan rastgele sayıların yazıp kaydedileceği dosya işlemleri yapılmaktadır.

```
save_data_pr : process
      file        file_id:    text;
      variable    line_num:   line;
      variable    cnt:        integer := 0;
  begin
      -- Open the file
      file_open(file_id, log_file, WRITE_MODE);
      wait until (reset = '0' and en = '1');
      wait until (clk = '1');

      -- Loop and write all values to a file
      for cnt in 0 to 2048*2-1 loop
    write(line_num, to_integer(unsigned(count)) );
        writeline(file_id, line_num);
        wait until (en = '1' and clk = '1');
      end loop;

      file_close(file_id);
      endSim <= true;
      wait until (clk = '1');

  end process save_data_pr;

end architecture;
```