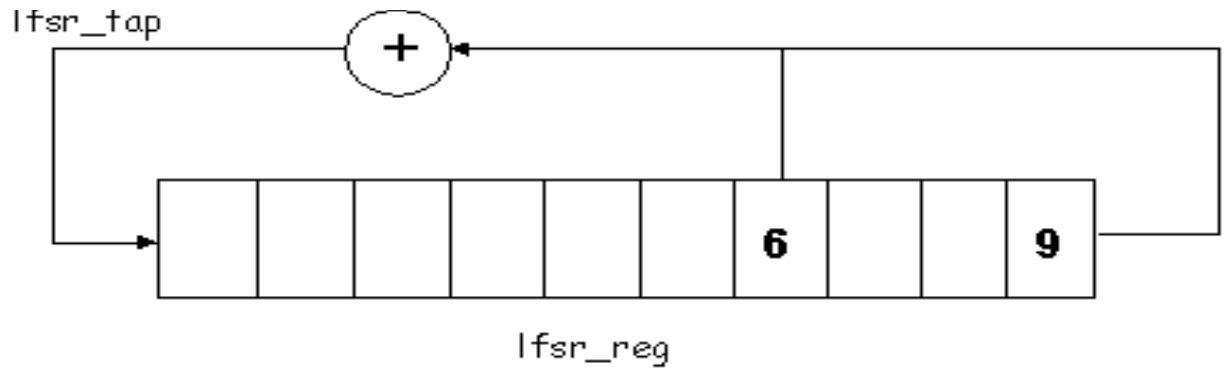
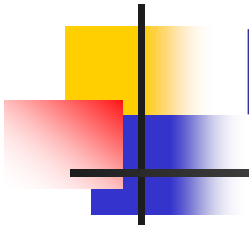
PRBS

Pseudo Random Binary Sequence





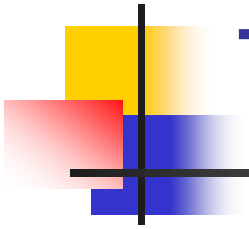
PROPERTIES

- The maximal length LFSR (Linear feedback shift registers) generates data that is almost random.
- Maximal length (2^n) -1.
- 000..00 – A special case.
- The problem: Choose the place for the XOR which provides the maximal length.

The math behind it…

$$G(X) = g_m X^m + g_{m-1} X^{m-1} + g_{m-2} X^{m-2} + \ldots + g_2 X^2 + g_1 X + g_0$$

- Any LFSR can be represented as a polynomial of variable X, referred to as the *generator polynomial*:
- g = tap weights – 1/0 (for connection)
- M = number of LFSR stages (9)
- For us : G(X) = X^9 + X^6 + 1



The math continues….

- The generator polynomial is said to be *primitive* if it cannot be factored, and if it is a factor of (i.e. can evenly divide) X^N+1, where N = 2^m-1 (the length of the m-sequence).
- An LFSR represented by a primitive polynomial will produce a maximal length sequence.

Example of Primitive Polynomials

n	primitive polynomials
1	x
2	$1+x+x^2$
3	$1+x+x^3, 1+x^2+x^3$
4	$1+x+x^4, 1+x^3+x^4$
5	$1+x^2+x^5, 1+x+x^2+x^3+x^5, 1+x^3+x^5, 1+x+x^3+x^4+x^5, 1+x^2+x^3+x^4+x^5,$ $1+x+x^2+x^4+x^5$

S – Edit (6 xor 9)

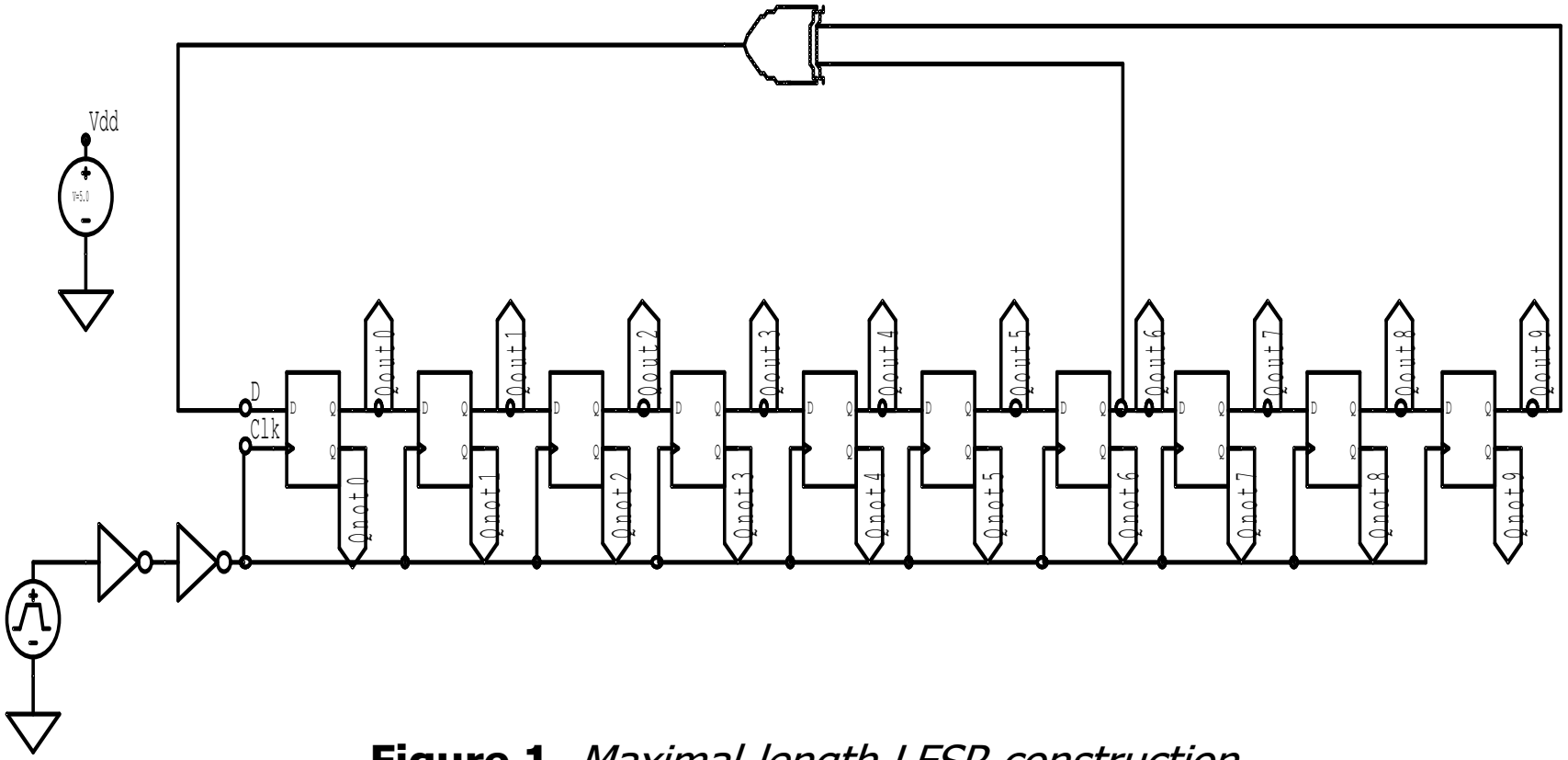


Figure 1. *Maximal-length LFSR construction*

VHDL simulation

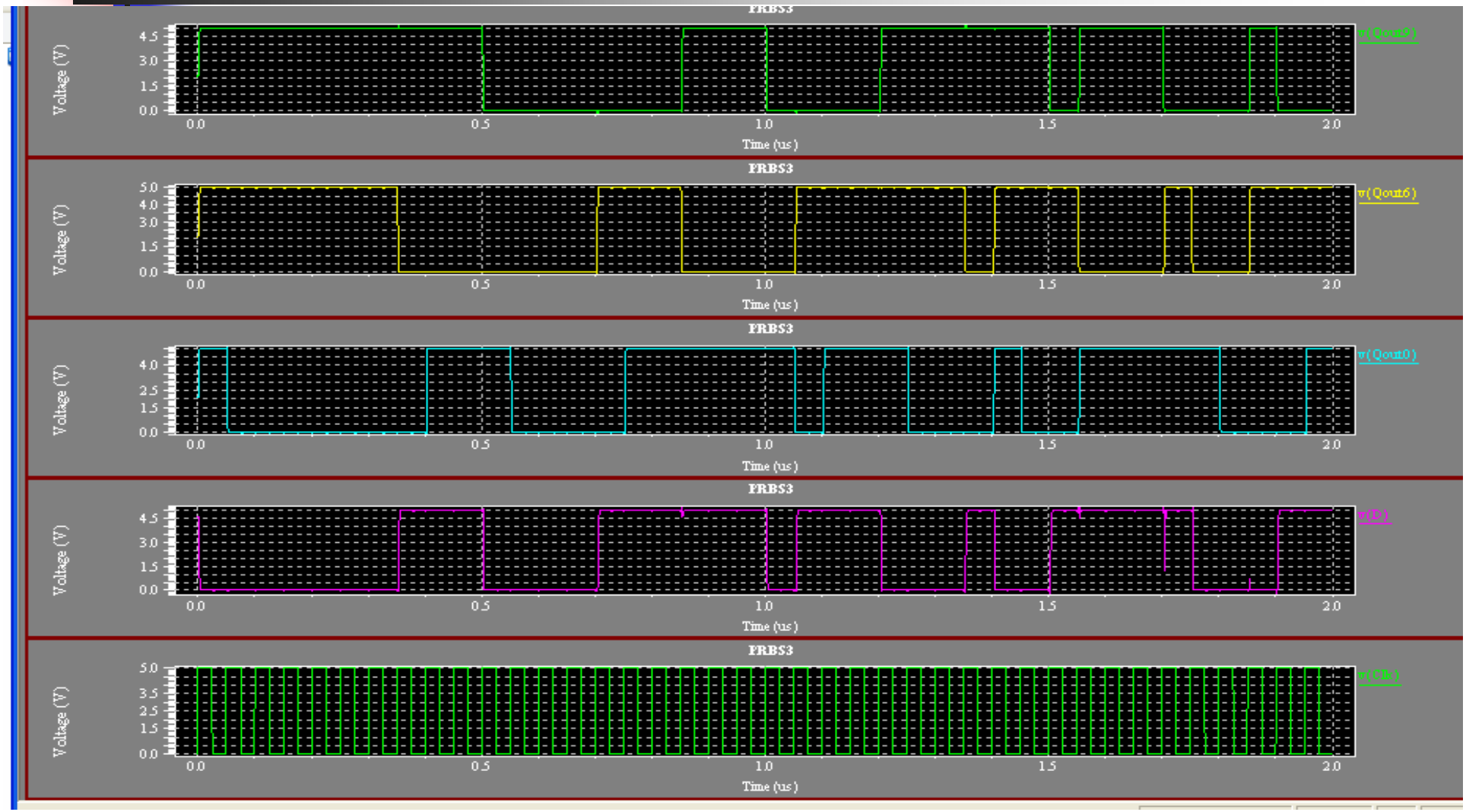
```
entity max_length_lfsr is
port (
  clock    : in bit;
  reset    : in bit;
  data_out : out bit_vector(9 downto 0)
);
end max_length_lfsr;

architecture modular of max_length_lfsr is
  signal lfsr_reg : bit_vector(9 downto 0);
begin
  process (clock)
    variable lfsr_tap : bit;
  begin
    if clock'EVENT and clock='1' then
      if reset = '1' then
        lfsr_reg <= (others => '1');
      else
        lfsr_tap := lfsr_reg(6) xor lfsr_reg(9);
        lfsr_reg <= lfsr_reg(8 downto 0) & lfsr_tap;
      end if;
    end if;
  end process;
  data_out <= lfsr_reg;
end modular;
```

```
end modular;

architecture modular_2 of max_length_lfsr is
  signal lfsr_reg : bit_vector(9 downto 0);
begin
  process (clock)
    variable lfsr_tap : bit;
  begin
    if clock'EVENT and clock='1' then
      if reset = '1' then
        lfsr_reg <= (others => '1');
      else
        lfsr_tap := lfsr_reg(3) xor lfsr_reg(5);
        lfsr_reg <= lfsr_reg(8 downto 0) & lfsr_tap;
      end if;
    end if;
  end process;
  data_out <= lfsr_reg;
end modular_2;

architecture modular_3 of max_length_lfsr is
  signal lfsr_reg : bit_vector(9 downto 0);
begin
  process (clock)
    variable lfsr_tap : bit;
  begin
    if clock'EVENT and clock='1' then
      if reset = '1' then
        lfsr_reg <= (others => '1');
      else
        lfsr_tap := lfsr_reg(1) xor lfsr_reg(6);
        lfsr_reg <= lfsr_reg(8 downto 0) & lfsr_tap;
      end if;
    end if;
  end process;
  data_out <= lfsr_reg;
end modular_3;
```

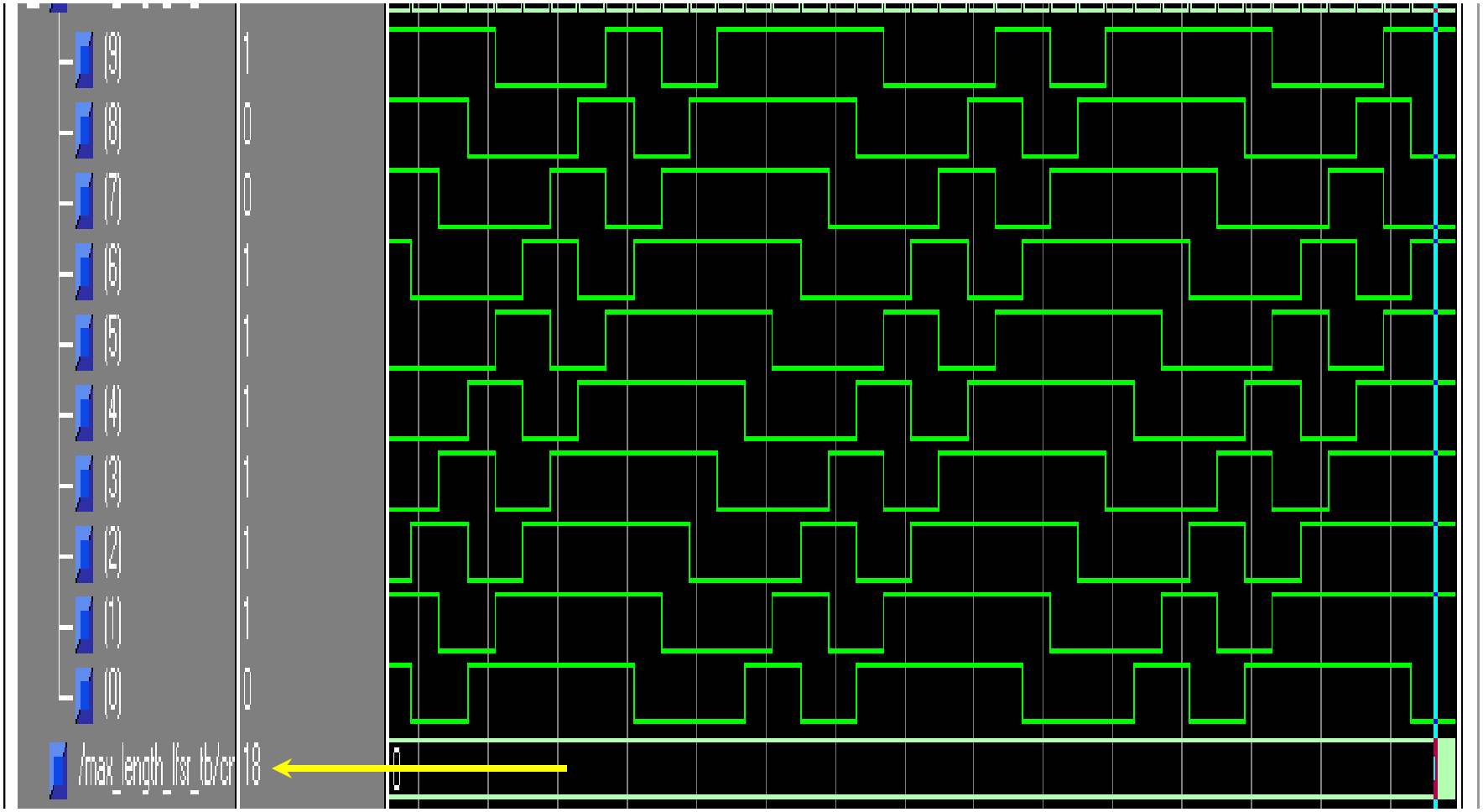
VHDL simulation – test bench

```
ARCHITECTURE arch OF max_length_lfsr_tb IS

    COMPONENT max_length_lfsr
     port (
            clock    : in bit;
            reset    : in bit;
            data_out : out bit_vector(9 downto 0)
          );
    END COMPONENT ;

   type signature_matrix is array(0 to 1023) of bit_vector(9 downto 0);
   SIGNAL matrix : signature_matrix;
   SIGNAL   sclock    : bit;
   SIGNAL   sreset  : bit := '1';
   SIGNAL   sdata_out : bit_vector(9 downto 0);
   CONSTANT period : time      := 20 ps ;
   signal cnt : integer;

FOR dut:max_length_lfsr use configuration WORK.con1;
--FOR dut:max_length_lfsr use configuration WORK.con2;
--FOR dut:max_length_lfsr use configuration WORK.con3;

BEGIN
  dut:max_length_lfsr PORT MAP(
                              clock => sclock,
                              reset => sreset,
                             data_out => sdata_out);
process
begin
       cnt <=0;
      sclock  <='0';
      sreset <= '0' after period;
      --matrix(0) <= sdata_out;
   for i in 0 to 1023 loop
      sclock  <=NOT sclock after period/2;
       wait for 20 ps;
       matrix(i) <= sdata_out;
      sclock  <=NOT sclock after period/2;
      wait for 20 ps;
  end loop;
  -- Check Cycle--
  for j in 0 to 1023 loop
    wait for 1 ps;
    for k in (j+1) to 1023 loop
      if(matrix(j) = matrix(k)) then
        cnt<=k;
        assert(false)
        report "Reached cycle after "
        severity ERROR;
        exit;
      end if;
    end loop;
  end loop;
wait;
end process;
END arch;
```

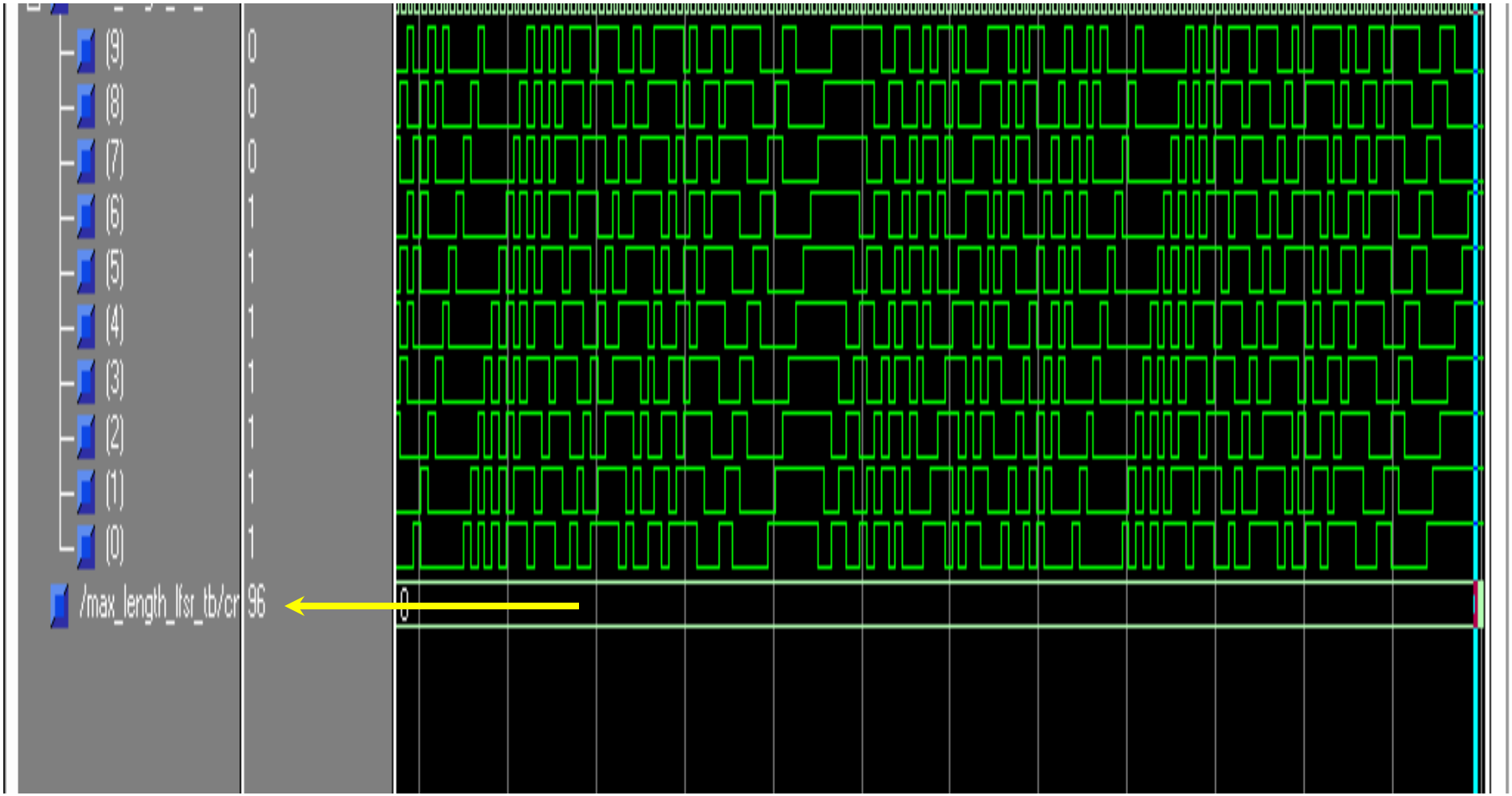
S – Edit Simulation



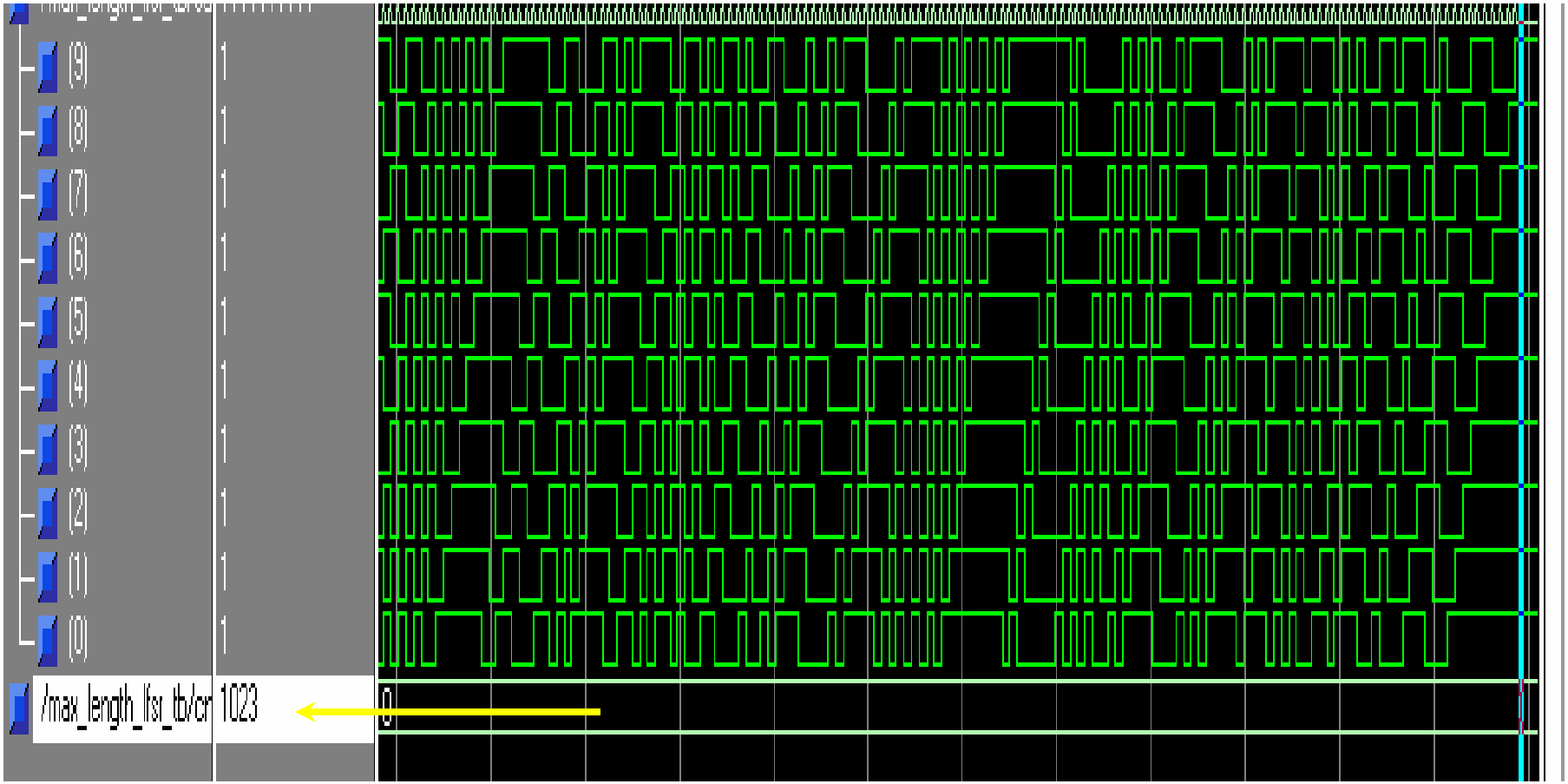
VHDL Module Simulation (3 XOR 5)



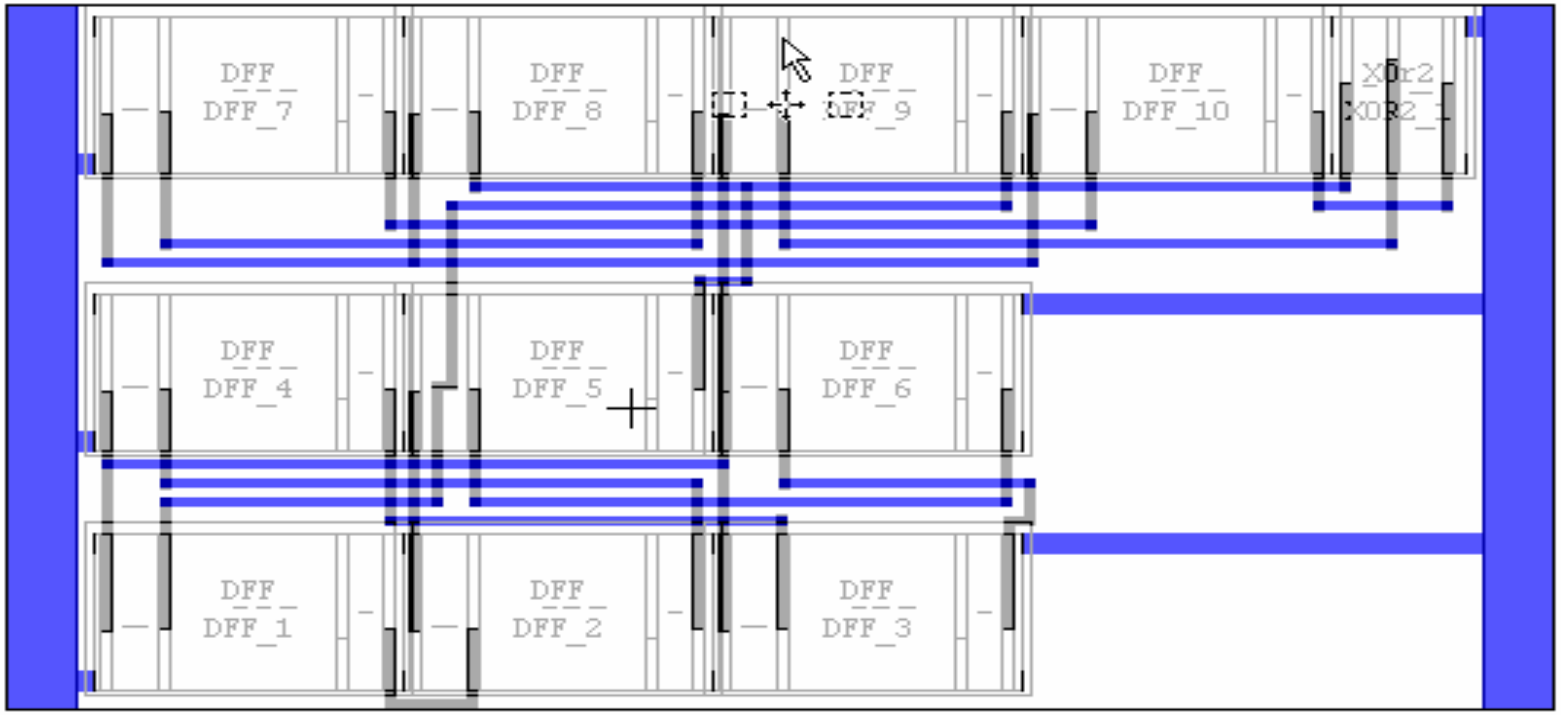
VHDL Module Simulation (1 XOR 6)



VHDL Module Simulation (6 XOR 9) Maximal PRBS



L-Edit


DFF_
DFF_7
DFF_
DFF_8
DFF_
DFF_9
DFF_
DFF_10
XOr2
XOR2_1
DFF_
DFF_4
DFF_
DFF_5
DFF_
DFF_6
DFF_
DFF_1
DFF_
DFF_2
DFF_
DFF_3

L-Edit (2)