Синтезируемые описания цифровых устройств на языке VHDL

Процесс синтеза – сложный процесс преобразования описания цифровых устройств, выполняемый специализированными программным обеспечением в полностью автоматическом или полуавтоматическом режиме.

Не любое описание устройства на языке VHDL может быть *синтезировано*.

Существует подмножество языка VHDL, позволяющее создать *синтезируемое* описание устройства на языке VHDL.

При обработке описания устройства программой синтеза, в случае обнаружения того или функционального узла

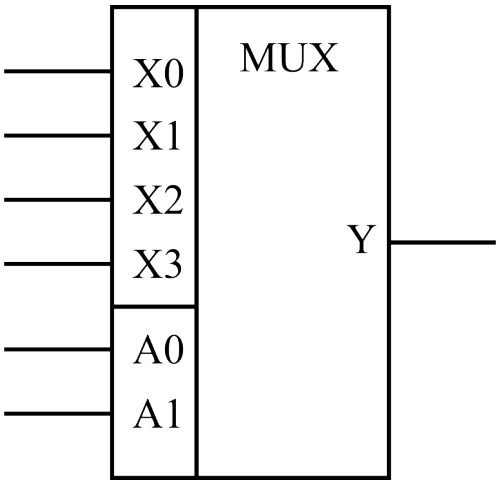
Комбинационная логика (логические операторы)

Логические операторы синтезируются непосредственно в логические вентили.

Операции **and**, **nand**, **nor** требуют незначительных ресурсов и обладают наименьшими задержками, операции **xor** и **xnor** требуют больших ресурсов и обладают большими задержками.

Для микросхем FPGA логические операции в большинстве случаев реализуются на базе блоков LUT (Look-Up Table), представляющих собой программируемую память на один разряд.

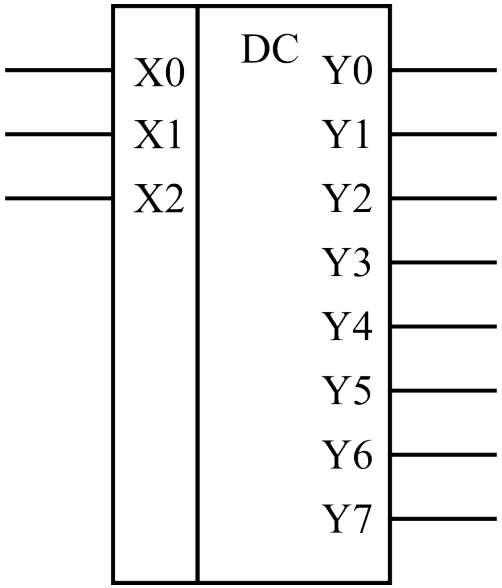
Комбинационная логика (мультиплексор)




```
process (X)
begin
     if    (A = "00")  then Y <= X(0);
     elsif (A = "01")  then Y <= X(1);
     elsif (A = "10")  then Y <= X(2);
     else                   Y <= X(3);
     end if;
end process;
```

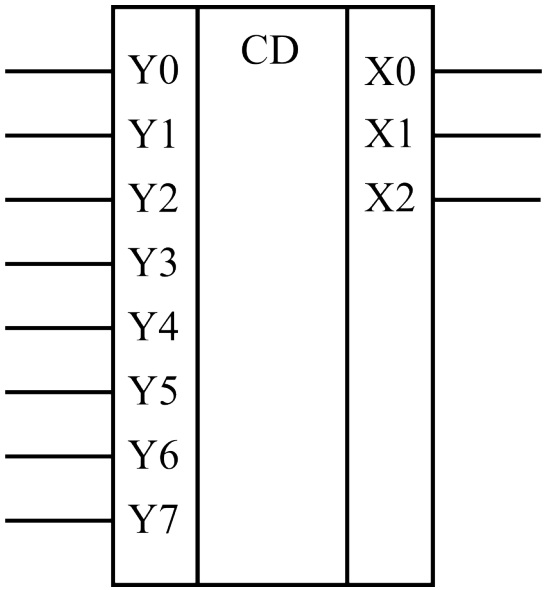
```
process (X)
begin
     case X is
          when "00"   => Y <= X(0);
          when "01"   => Y <= X(1);
          when "10"   => Y <= X(2);
          when others => Y <= X(3);
     end case ;
end process;
```

Комбинационная логика (дешифратор)




```
Y <=    "00000001" when X = "000" else
        "00000010" when X = "001" else
        "00000100" when X = "010" else
        "00001000" when X = "011" else
        "00010000" when X = "100" else
        "00100000" when X = "101" else
        "01000000" when X = "110" else
        "10000000";
```

Комбинационная логика (приоритетные шифраторы)




```
X <=    “000” when X(0) = '1' else
        “001” when X(1) = '1' else
        “010” when X(2) = '1' else
        “011” when X(3) = '1' else
        “100” when X(4) = '1' else
        “101” when X(5) = '1' else
        “110” when X(6) = '1' else
        “111” when X(7) = '1' else
        “---”;
```

Комбинационная логика (арифметические операции)

Программы синтеза разных производителей могут поддерживать разный набор арифметических операций.

Программы синтеза фирмы Xilinx (XST) поддерживает реализацию следующих арифметических операций :

- Сумматоры ;
- Вычитатели ;
- Компараторы ;
- Умножители ;
- Делители .

Комбинационная логика (сумматоры и вычитатели)

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
```

Тип используемого пакета определяет тип арифметической операции : знаковый или нет

```
        ...
    SUM <= A + B ;
    SUM <= A + B + CI ;
    SUM <= '0' & A + '0' & B ;

    SUB <= A - B ;
    SUB <= A - B - BI ;
```

Сумматор

Сумматор с учетом переноса

Сумматор, формирующий перенос

Вычитатель

Вычитатель с учетом заема

Комбинационная логика (компараторы, умножители, делители)

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
        ...
    CMP <= '1' when A >= B else '0';
```

Тип используемого пакета определяет тип арифметической операции : знаковый или нет

Компаратор : больше или равно

```
MUL <= A * B ;
```

Умножитель

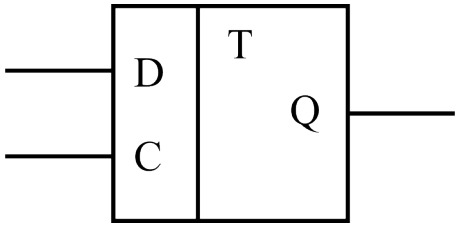
Размерность результата должна быть равна сумме размерностей множителей

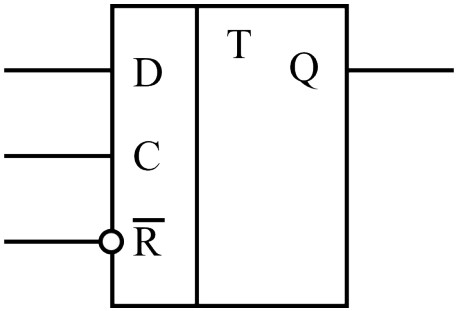
```
DIV <= A / 2 ;
```

Делитель

Делитель должен быть константой

Реализация функциональных узлов последовательного типа (статические триггеры)

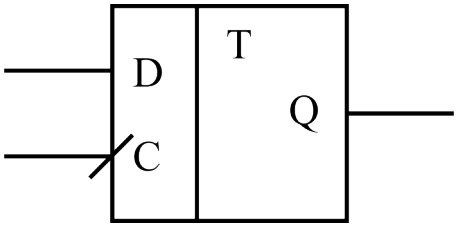


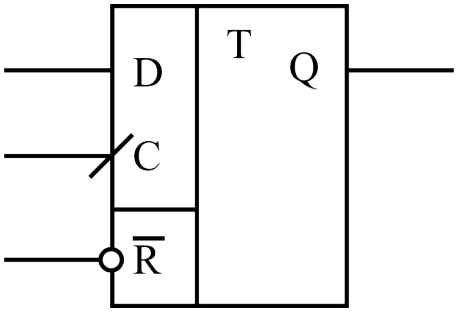



```
process (C,D)
begin
      if C = '1' then
            Q <= D;
      end if;
end process;
```

```
process (C,D,R)
begin
      if R = '0' then
            Q <= '0';
      elsif C = '1' then
            Q <= D;
      end if;
end process;
```

Реализация функциональных узлов последовательного типа (динамические триггеры)

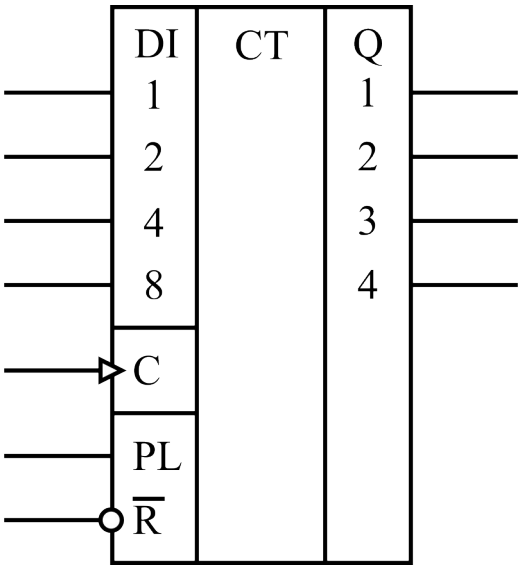





```
process (C)
begin
      if C'event and C = '1' then
            Q <= D;
      end if;
end process;
```

```
process (C,R)
begin
      if R = '0' then
            Q <= '0';
      elsif C'event and C = '1' then
            Q <= D;
      end if;
end process;
```

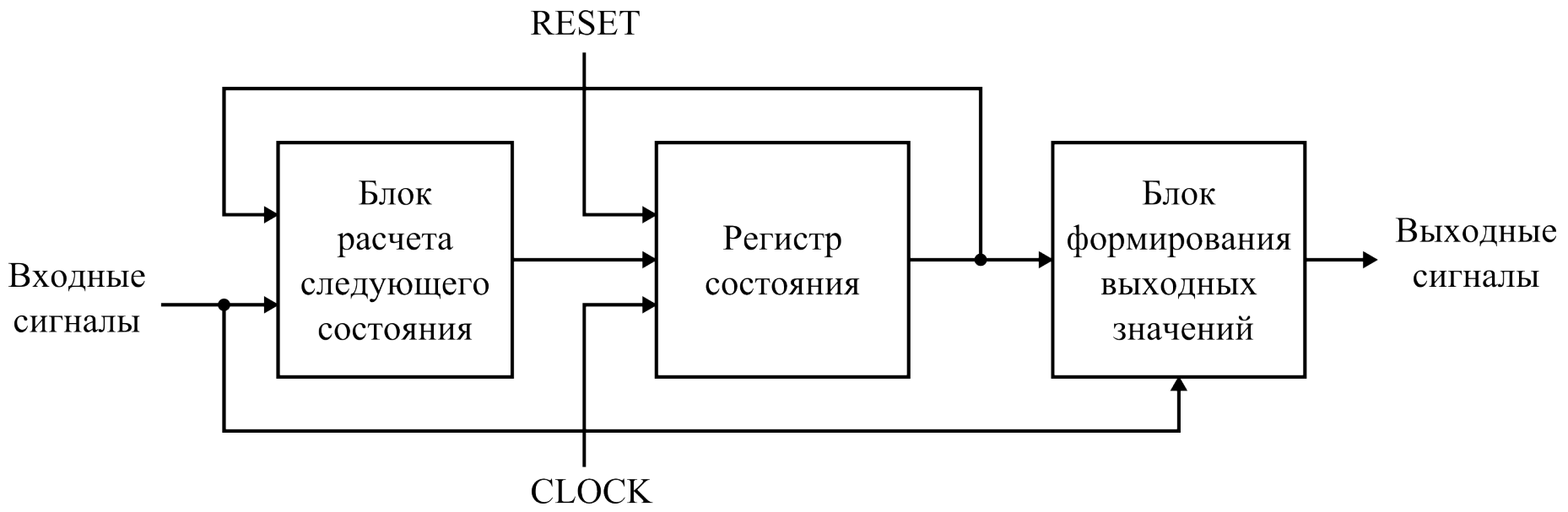
Реализация функциональных узлов последовательного типа (счетчики)



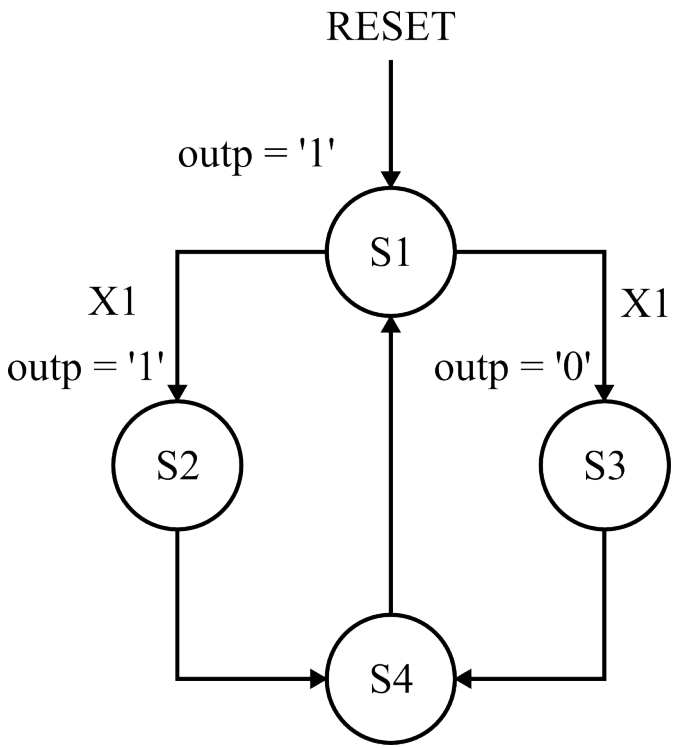

```
process (C,R,PL,DI)
        variable count : std_logic_vector(3 downto 0);
begin
        if R = '0' then
                count := (others => '0');
        elsif PL = '1' then
                count := DI;
        elsif C'event and C = '1' then
                count := count + 1;
        end if;
        Q <= count;
end process;
```

Реализация конечных автоматов (машин состояний) на языке VHDL

Структурная схема конечного автомата :

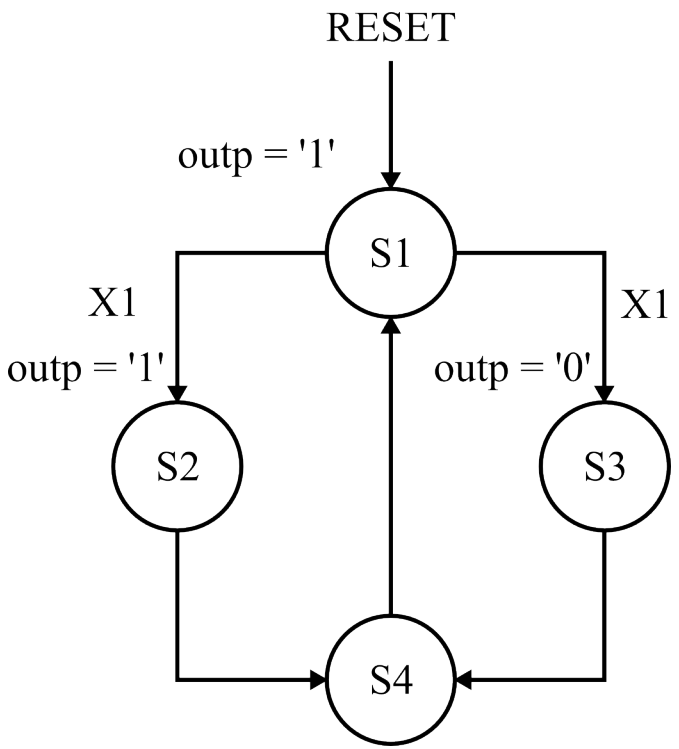


Реализация конечных автоматов (машин состояний) на языке VHDL




```
process (clk,reset)
begin
      if reset = '1' then
            state <= s1;
            outp <= '1'
      elsif clk'event and clk = '1' then
            case state is
                  when s1 =>
                        if x1 = '1' then
                              state <= s2;
                              outp <= '1';
                        else
                              state <= s3;
                              outp <= '0';
                        end if;
                  when s2 => state <= s4; outp <= '0';
                  when s3 => state <= s4; outp <= '0';
                  when s4 => state <= s1; outp <= '1';
            end case;
      end if;
end process;
```

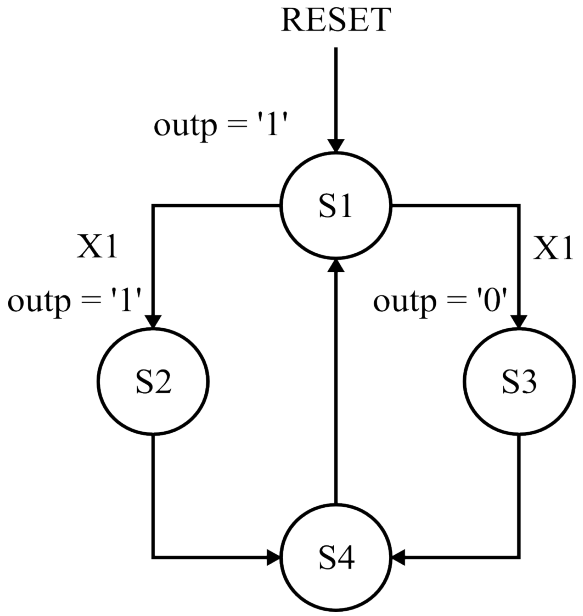
Реализация конечных автоматов (машин состояний) на языке VHDL




```
process1 (clk,reset)
begin
      if reset = '1' then state <= s1;
      elsif clk'event and clk = '1' then
            case state is
                  when s1 =>
                        if x1 = '1' then state <= s2;
                        else state <= s3;
                        end if;
                  when s2 => state <= s4;
                  when s3 => state <= s4;
                  when s4 => state <= s1;
            end case;
      end if;
end process1;

process2(state)
begin
      case state is
            when s1 => outp <= '1';
            when s2 => outp <= '1';
            when s3 => outp <= '0';
            when s4 => outp <= '0';
      end case;
end process2;
```

Реализация конечных автоматов (машин состояний) на языке VHDL




```
process1 : process(clk,reset)
begin
      if reset = '1' then
            state <= s1;
      elsif clk'event and clk = '1' then
            state <= next_state;
      end if;
end process;

process2 : process (state,x1)
begin
      case state is
            when s1 =>
                  if x1 = '1' then next_state <= s2;
                  else next_state <= s3;
                  end if;
            when s2 => next_state <= s4;
            when s3 => next_state <= s4;
            when s1 => next_state <= s1;
      end case;
end process;
```

```
process3 : process(state)
begin
      case state is
            when s1 => outp <= '1';
            when s2 => outp <= '1';
            when s3 => outp <= '0';
            when s4 => outp <= '0';
      end case;
end process2;
```