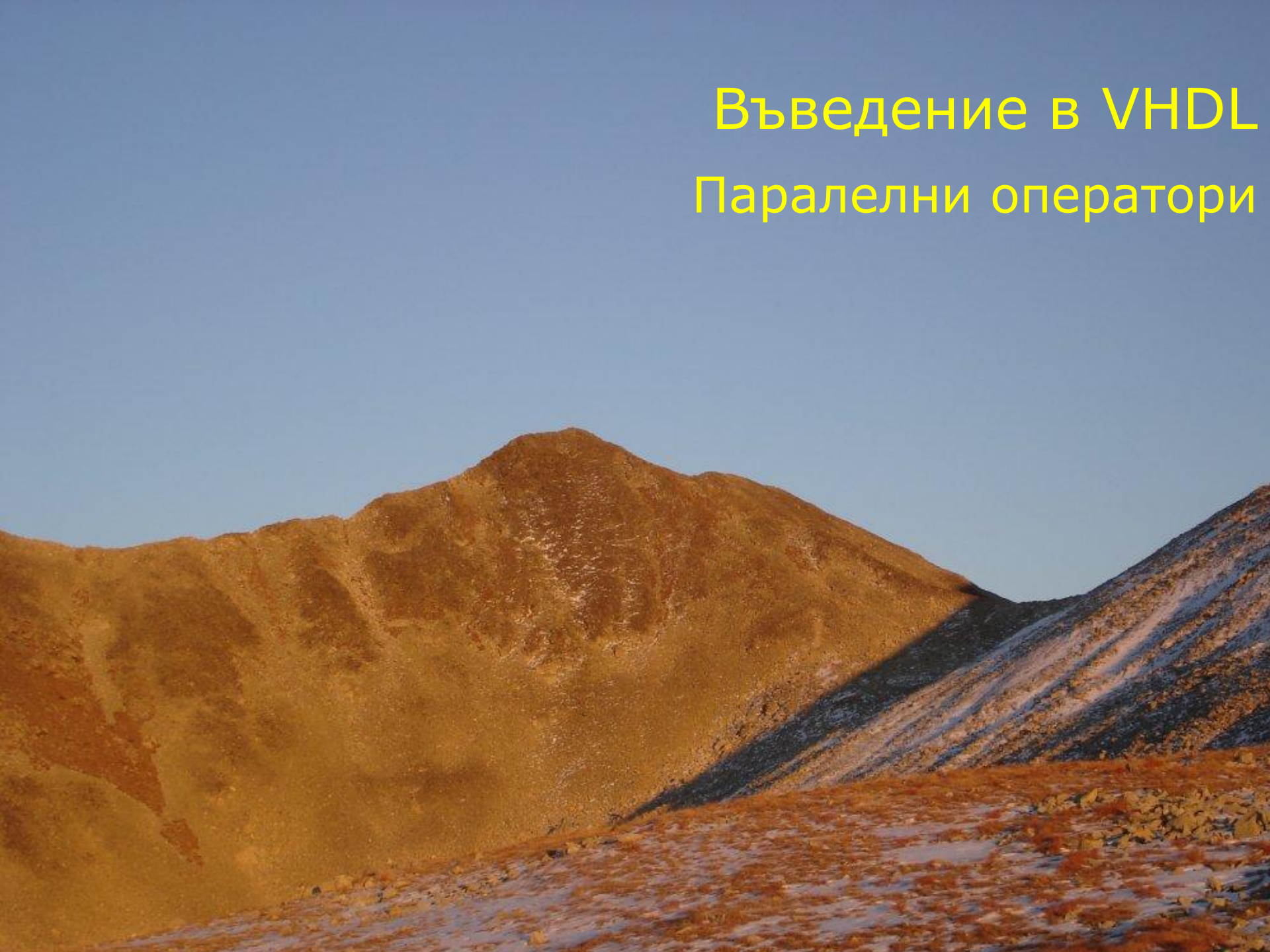


Въведение в VHDL

Паралелни оператори

Паралелни оператори

Паралелен оператор – оператор, който се изпълнява едновременно с другите оператори от този тип.

- Process - Процес
- <= Сигнално присвояване
- Port map – поставяне на компонент
- Извикване на процедура
- Generate

Оператор Process

Process - паралелен оператор, който описва поведение.

Процесът съдържа последователни оператори.

Кога ще се изпълни даден процес зависи от сигналите, спрямо които той е “чувствителен” или от оператори wait.

Process – Синтаксис

[*етикет*:] process [(*списък от сигнали*)]

Декларации...

begin

Последователни оператори...

end process;

списък от сигнали – сигнали спрямо които процесът е “чувствителен”.

Process – Правила

Един процес трябва да има или списък на сигнали, спрямо които е чувствителен или да съдържа оператор wait, но не и двете.

Всеки процес се изпълнява веднъж по време на инициализацията, преди симулацията да е започнала.

Process – Синтез

Не всеки процес подлежи на синтез!

За целите на синтеза се препоръчват следните стандартни форми на оператора process:

Вариант 1 – комбинационна схема

process (входове) – всички входни сигнали са изброени

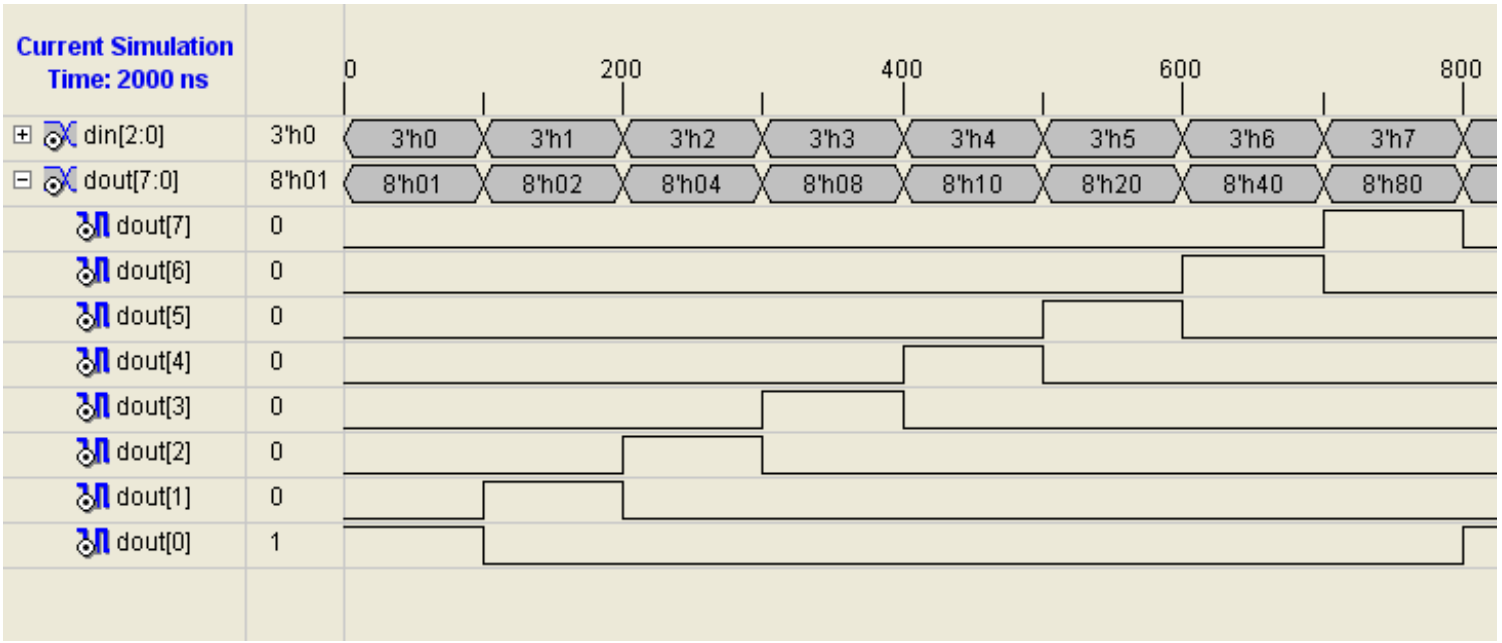
begin

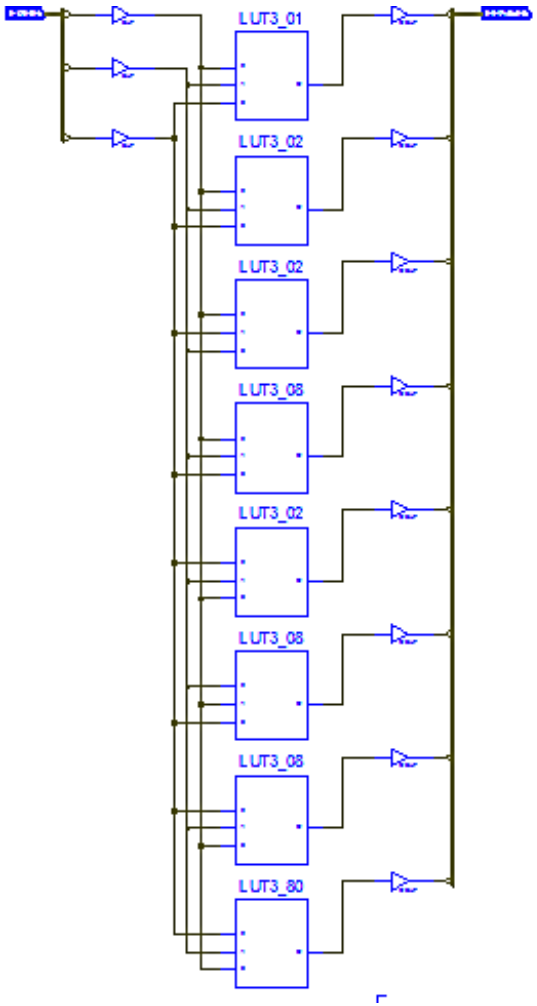
... – за всички входни комбинации се присвояват стойности на изходите

... – отсъства обратна връзка

end process;

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity decoder is
    port (
        DIN: in STD_LOGIC_VECTOR(2 downto 0);
        DOUT: out STD_LOGIC_VECTOR(7 downto 0));
end decoder;

architecture Behavioral of decoder is
begin
process(DIN)
begin
    case DIN is
        when "000" => DOUT <= "00000001";
        when "001" => DOUT <= "00000010";
        when "010" => DOUT <= "00000100";
        when "011" => DOUT <= "00001000";
        when "100" => DOUT <= "00010000";
        when "101" => DOUT <= "00100000";
        when "110" => DOUT <= "01000000";
        when "111" => DOUT <= "10000000";
        when others => NULL;
    end case;
end process;
end Behavioral;
```





Вариант 2 – тригери по ниво + логика

```
process (входове) -- всички входни сигнали са изброени
begin
    if Enable = '1' then
      ...
    end if;
end process;
```

Пример

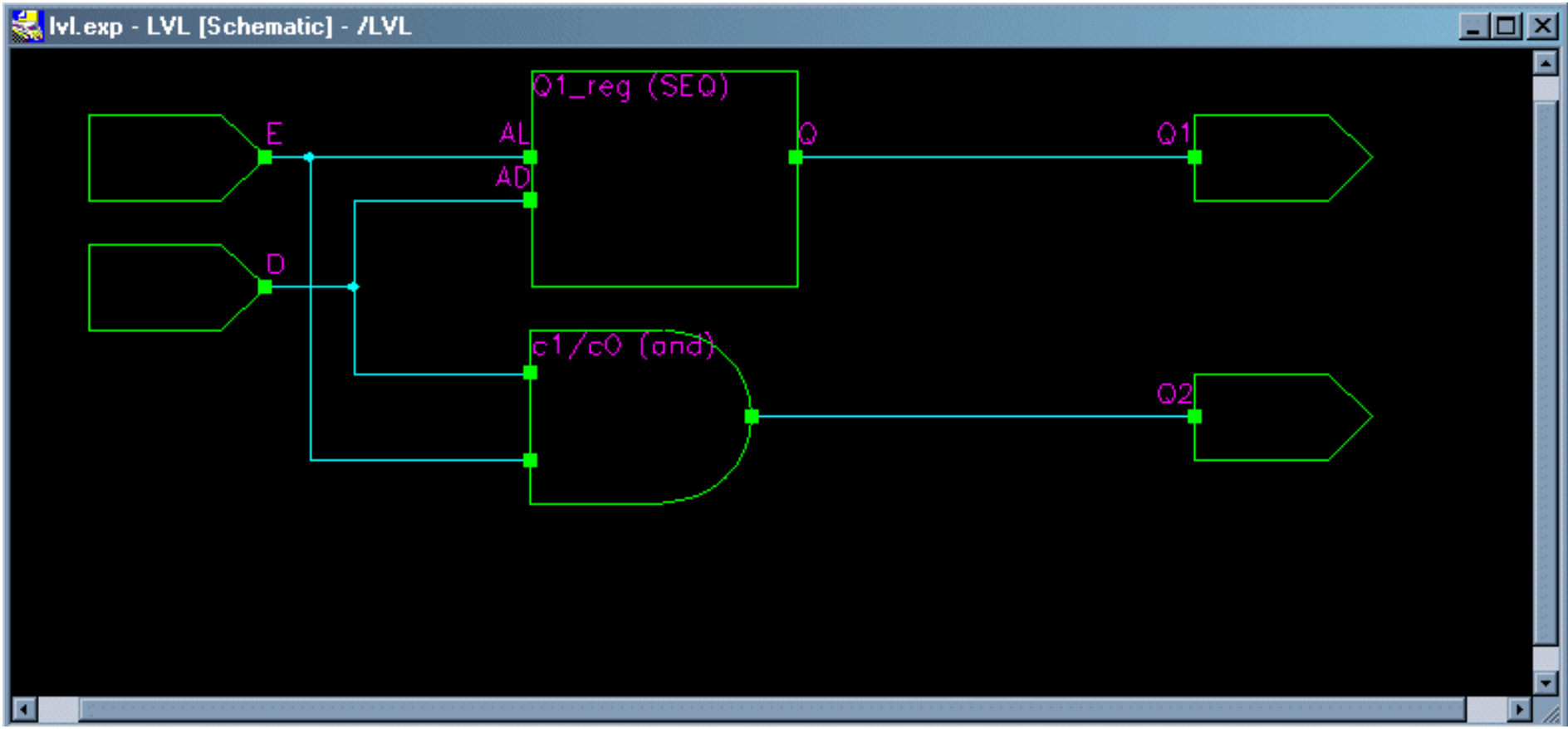
Непълно дефиниране на изхода Q1

```
process (D, E) begin
    if E = '1' then
       Q1 <= D;
    end if;
  end process;
```

Пълно дефиниране на изхода Q2

```
process (D, E) begin
    if E = '1' then
       Q2 <= D;
    else
       Q2 <= '0';
    end if;
  end process;
```

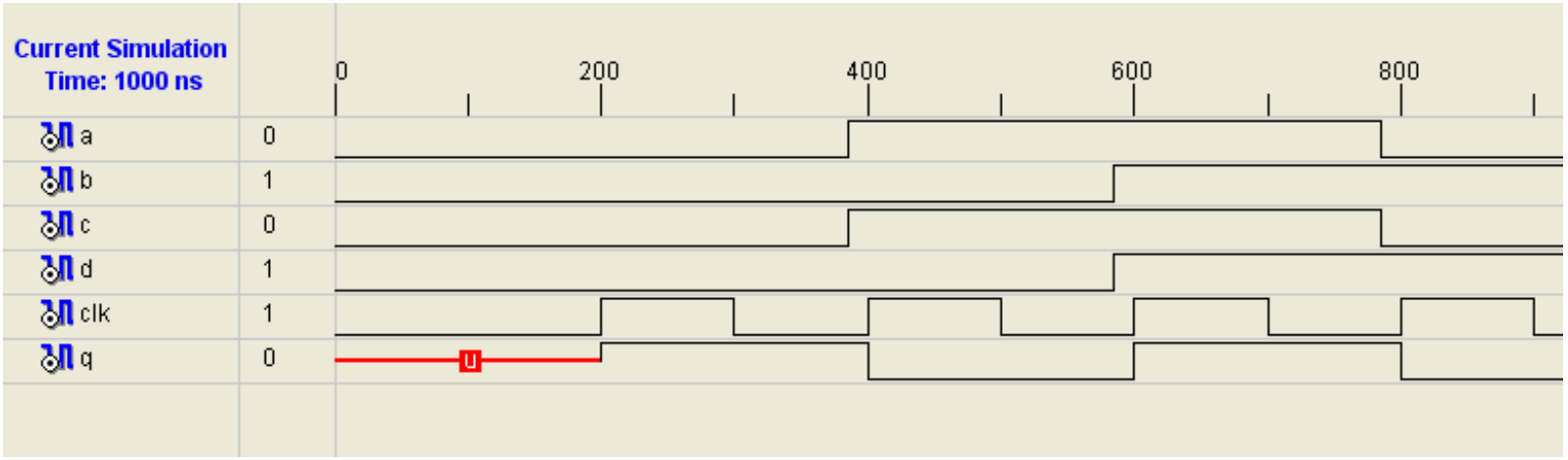
Пример

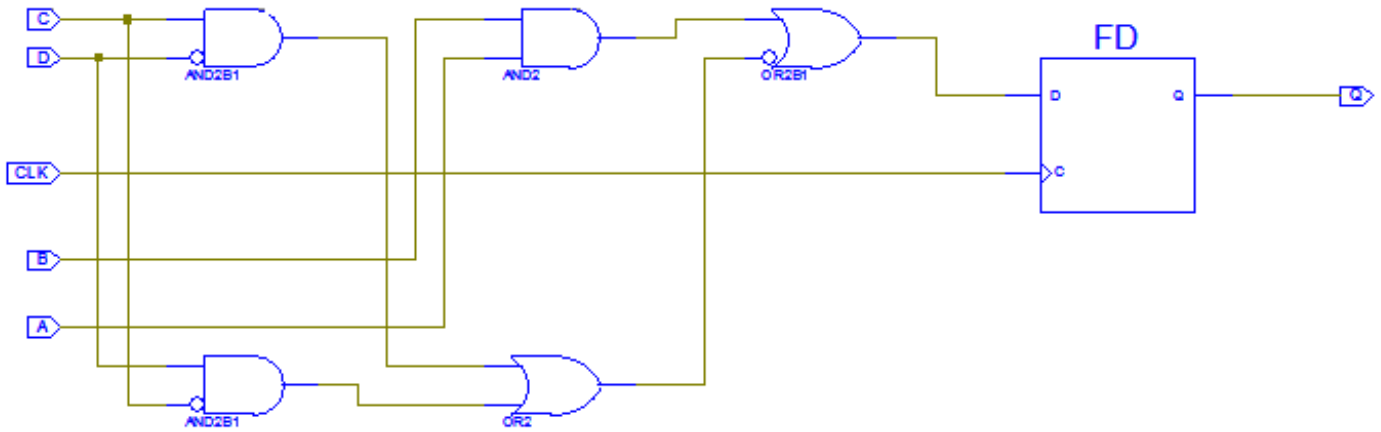


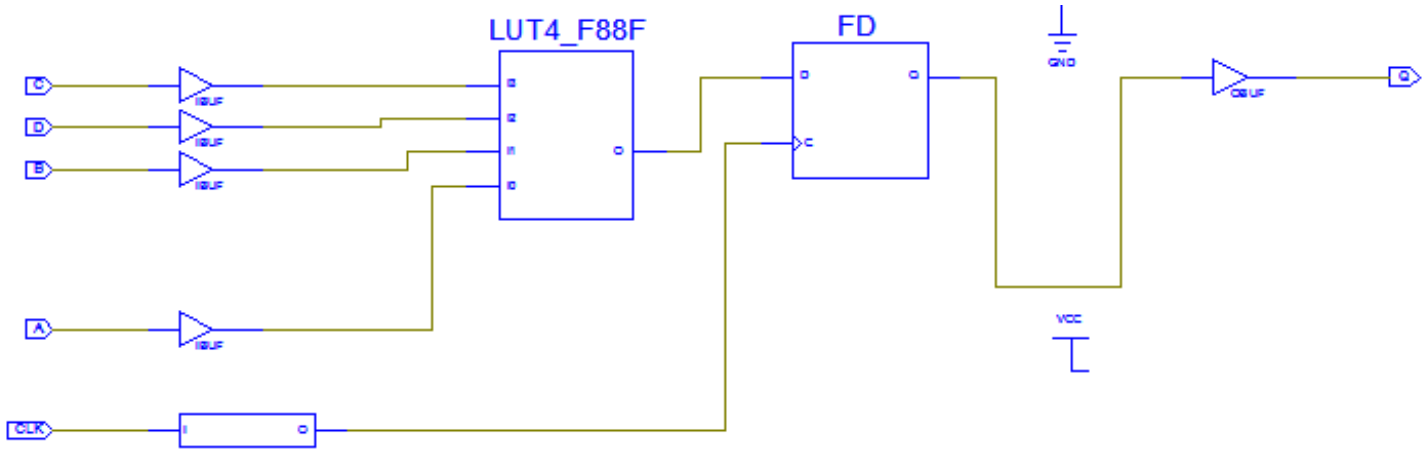
Вариант 3 – тригери + логика

```
process (Clock) -- само такт
begin
    if rising_edge(clock) then --тест за фронт на такта
     ...
    end if;
end process;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ff is
    Port ( A,B,C,D,CLK : in  STD_LOGIC;
           Q : out  STD_LOGIC);
end ff;

architecture Behavioral of ff is
begin
process (CLK) begin
    if rising_edge(CLK) then
        Q <= (A and B) or not(C xor D);
    end if;
end process;
end Behavioral;
```





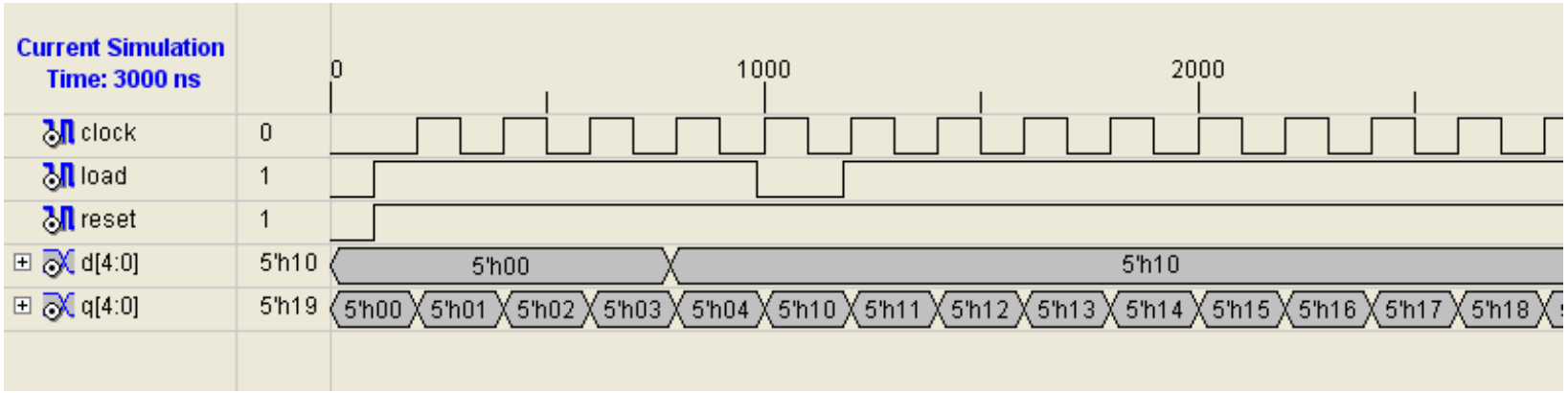


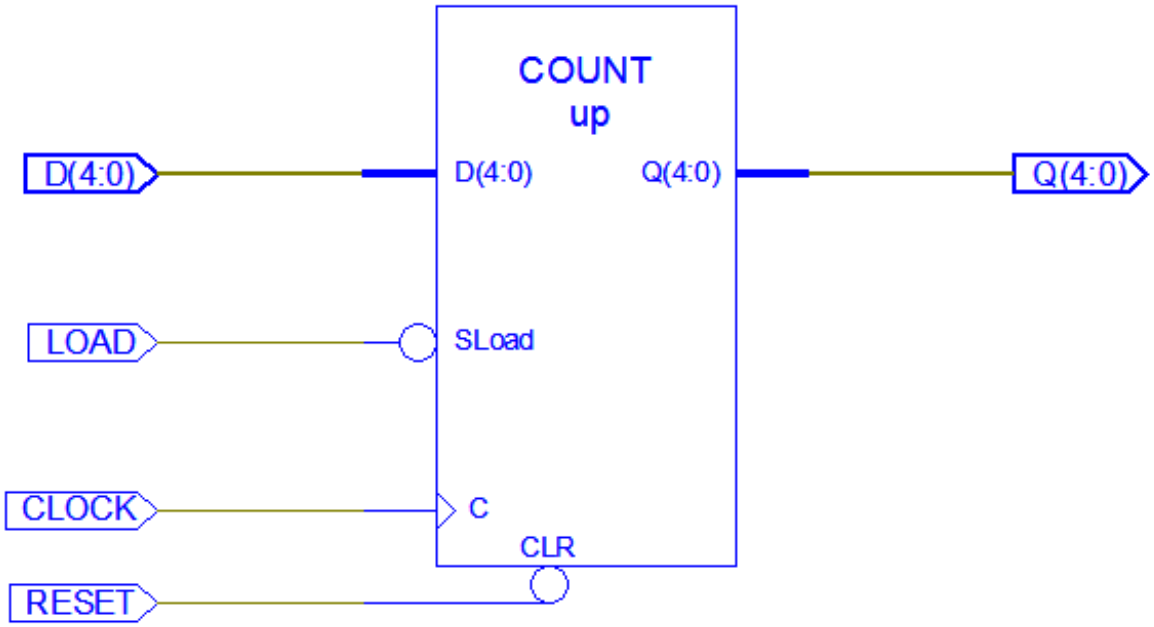
Вариант 4 – тригери + логика

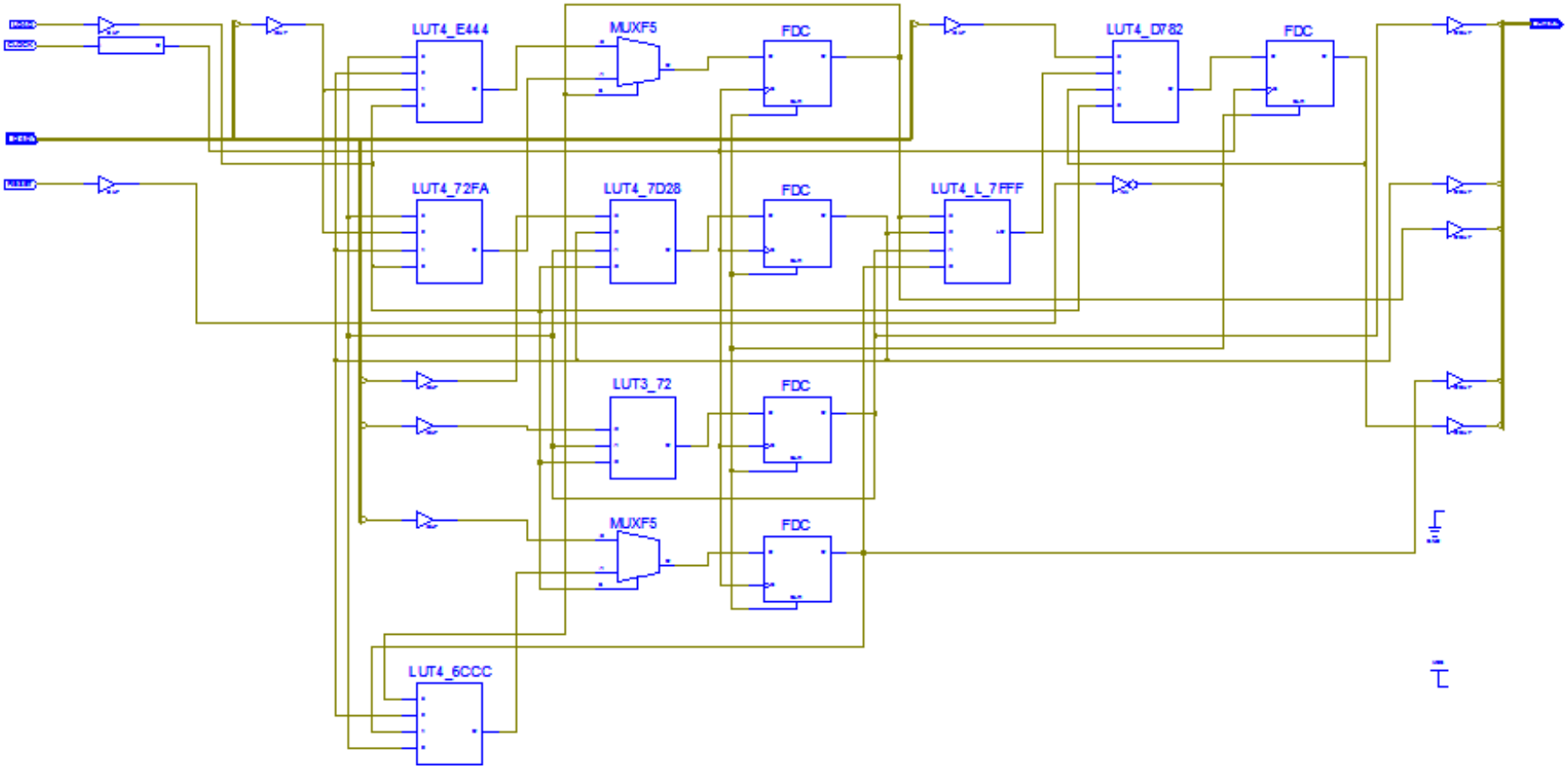
```
process (Clock, Reset) -- само такт и асинхронни сигнали
begin
if Reset = '0' then -- тест за асинхронно нулиране
    ... -- асинхронни действия
elsif rising_edge(Clock) then -- тест за фронт на такта
    ... -- синхронни действия
end if;
end process;
```

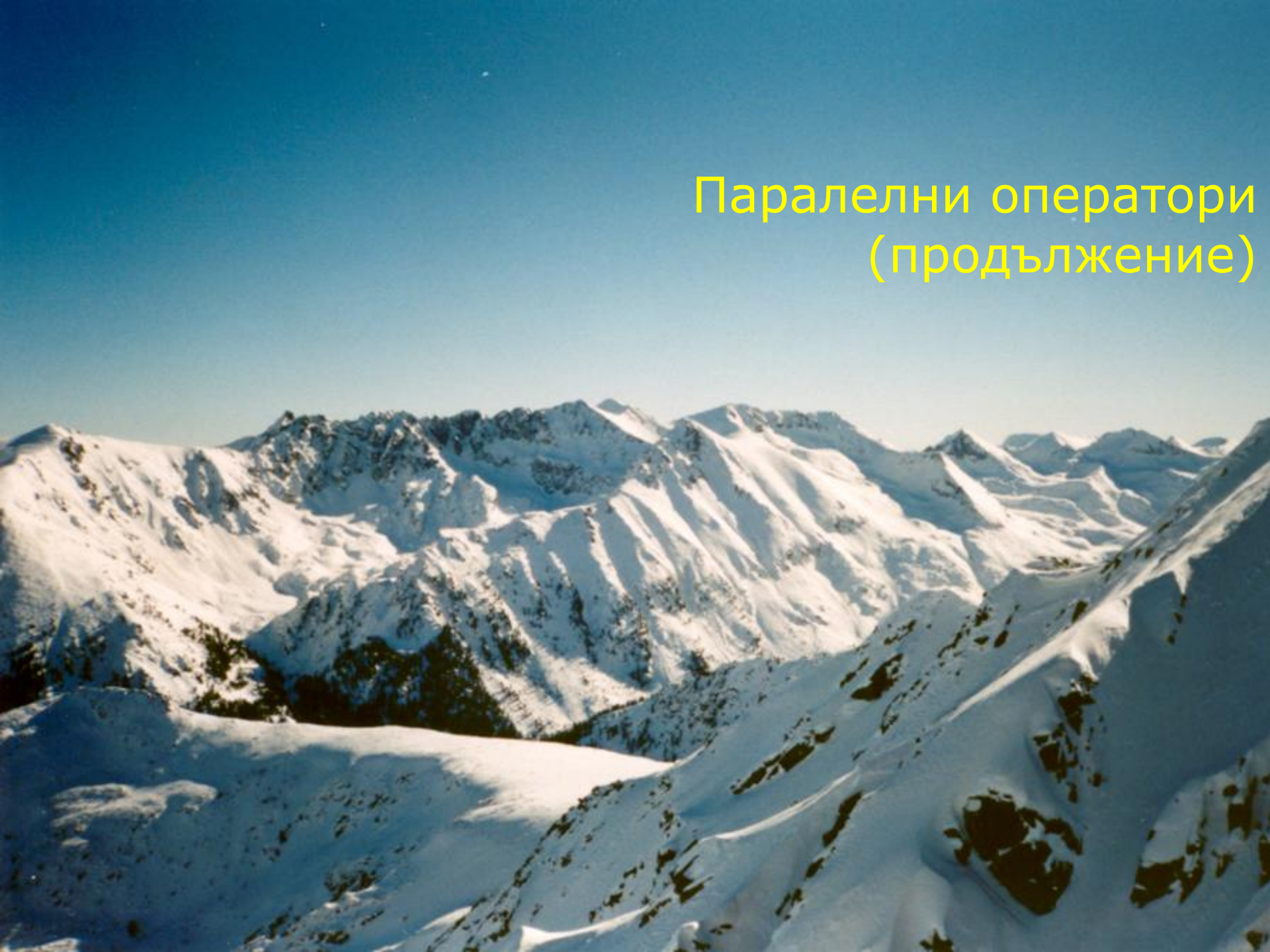
Пример (брояч)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity cnt5 is
Port ( CLOCK, LOAD, RESET : in STD_LOGIC;
         D : in STD_LOGIC_VECTOR (4 downto 0);
         Q : out STD_LOGIC_VECTOR (4 downto 0));
end cnt5;

architecture Behavioral of cnt5 is

signal TMP: std_logic_vector(4 downto 0);
begin
process (CLOCK,RESET) begin
         if RESET = '0' then
         TMP <= (others=>'0');
         elsif rising_edge(CLOCK) then
         if LOAD = '0' then
                  TMP <= D;
         else
              TMP <= TMP + 1;
       end if;
     end if;
  end process;
  Q <= TMP;
end Behavioral;
```









Паралелни оператори (продължение)

Условно Присвояване – when

[етикет:] цел <=

Израз [after време] when условие else

Израз [after време] when условие else

...

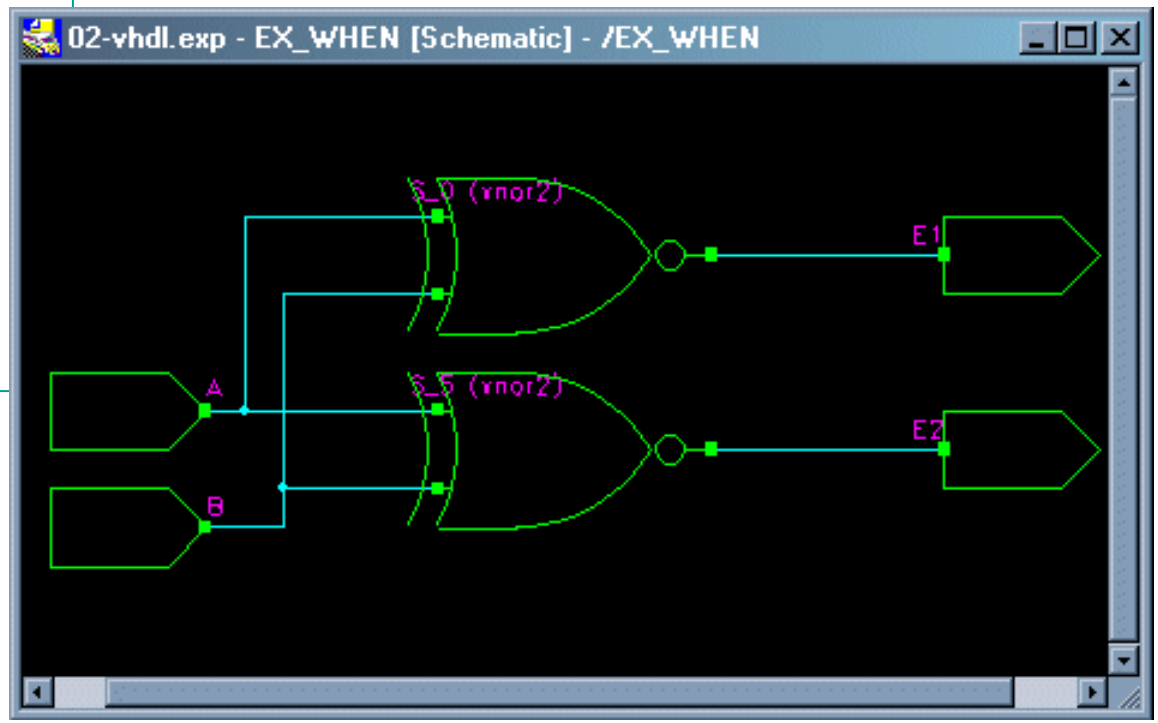
Израз [after време];

Еквивалентен на process с if в него.

Условно Присвояване – Пример

```
process (A, B) begin
  if A=B then
     E2 <= '1';
  else
     E2 <= '0';
  end if;
end process;
```

```
E1 <= '1' when A = B else '0';
```



Условно Присвояване – Синтез

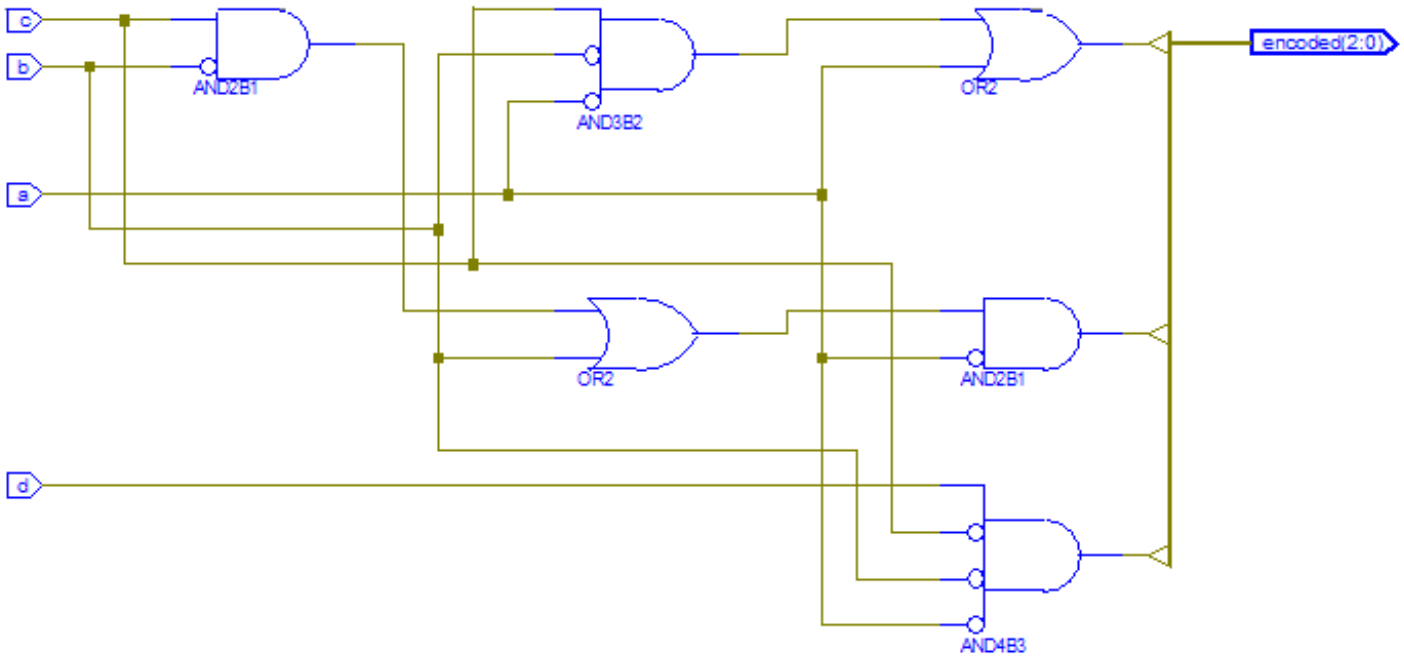
Операторите за условно присвояване се синтезират като комбинационна логика.

Изразите от дясната страна се мултиплексират към сигнала от лявата страна на оператора.

Получената логика ще бъде приоритетно кодирана, тъй като условията се проверяват последователно.

Пример – приоритетна логика

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity par_encoder is
    Port ( a,b,c,d : in  STD_LOGIC;
           encoded : out STD_LOGIC_VECTOR(2 downto 0));
end par_encoder;

architecture Behavioral of par_encoder is
begin
encoded <= "001" when a = '1' else
     "010" when b = '1' else
     "011" when c = '1' else
     "100" when d = '1' else
     "000";
end Behavioral;
```



Избор – with .. select

[етикет:] with *израз* select

цел <=

израз [after време] when *избори*,

израз [after време] when *избори*,

... ;

Еквивалентен на process с case в него.

Избор - Правила

Типа на управляващият *Израз* трябва да бъде изброим, физически, integer или едномерен масив.

Всяка възможна стойност на *Израз*-а трябва да попада точно в един от *Избор*-ите.

Избор - Синтез

Операторите за избор се синтезират като комбинационна логика.

Изразите от дясната страна се мултиплексират към сигнала от лявата страна на оператора.

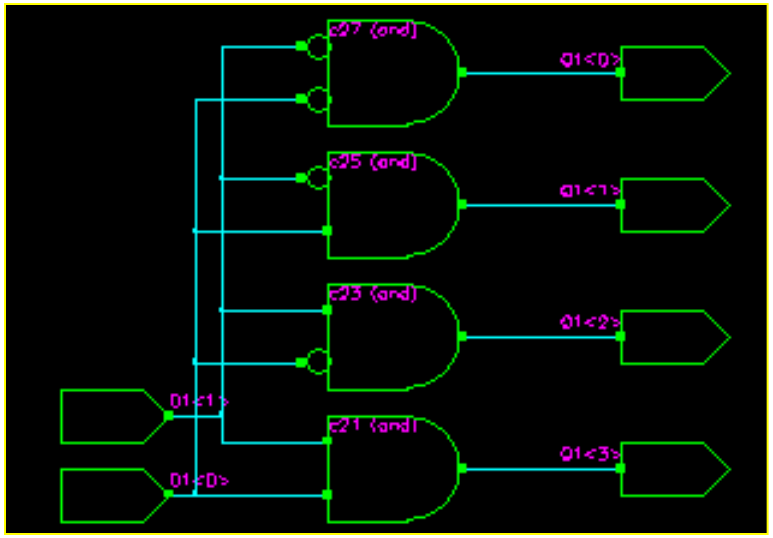
Пример

```
-- CASE

process(D1) begin
    case D1 is
      when "00" => Q1 <= "0001";
      when "01" => Q1 <= "0010";
      when "10" => Q1 <= "0100";
      when "11" => Q1 <= "1000";
      when others => Q1 <= "0000";
    end case;
  end process;
```

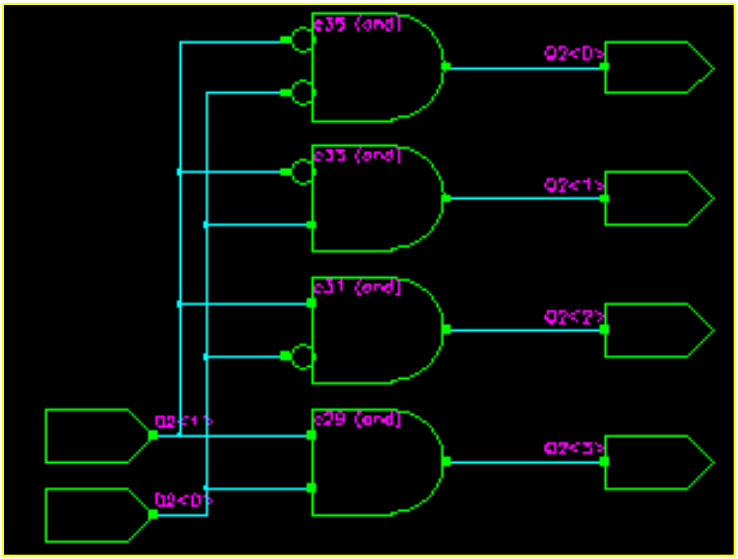
```
-- WITH  SELECT

with D2 select
     Q2 <= "0001" when "00",
       "0010" when "01",
       "0100" when "10",
       "1000" when "11",
       "0000" when others;
```

Пример

CASE



WITH SELECT



Структурни Оператори

Използват се за структурни VHDL модели.

–Component
–Port
–Port map
–For .. generate

Оператор Component

- Компонентът дава възможност да се използва един йерархичен блок в друг такъв.
- Компонента се поставя в архитектура и се асоциира с проектан единица (entity + architecture) от по ниско йерархично ниво.
- Синтаксис:

component *Име на компонента*
 [Generic;]
 [Port;]
end component;

Component - Правила

Името на компонента трябва да съответства на името на проектна единица, която ще бъде използвана на неговото място (освен ако не се укаже явна конфигурация).

Парамерите (generics) и портовете на компонента и проектаната единица трябва да си съответстват по име, режим и тип.

Оператор Port

•Опеартора Port дефинира интерфейсните сигнали на проектната единица.

•Синтаксис:

port
(*име на сигнал*, ... : [*режим*] *тип* [:= *начална стойност*];
...);

режим = in | out | inout | buffer

Оператор Port Map

- Свързва сигнали от архитектурата към портове на компонент използван в тази архитектура.
- Синтаксис

етикет: *Компонент* port map ([*порт* =>] *сигнал*, ...)

- Съответствието между портовете и сигналите се задава посредством позиционна или поименна асоциация.

Пример за асоциация

```
component COUNT
    port (CLK, RESET: in std_logic; UD: in std_logic;
    Q: out std_logic_vector(3 downto 0));
end component;
...
-- позиционна асоциация
G1: COUNT port map (C32, RST, DIR, Count);

-- поименна асоциация (реда е без значение)
G2: COUNT port map ( RESET => RST, CLK => C32,
   Q => Count, UD => DIR);
```

Пример за асоциация

```
component Adder
   port (a, b: in std_logic;
   s, cout: out std_logic);
end component;
...
-- позиционна асоциация
D1: Adder port map (n1(0), n2(0), sum(0), open);

-- поименна асоциация
D2: Adder port map ( s => sum(0), a => n1(0),
   b => n2(0), cout => open);
```

Пример за параметризиран структурен модел

Примерът описава 21 битов суматор.

Йерархията се състои от три нива:

–add21 – горно ниво – тук се задава разрядността на суматора (21);

–adderN – междинно ниво – тук се намира параметричният модел на N-битов суматор;

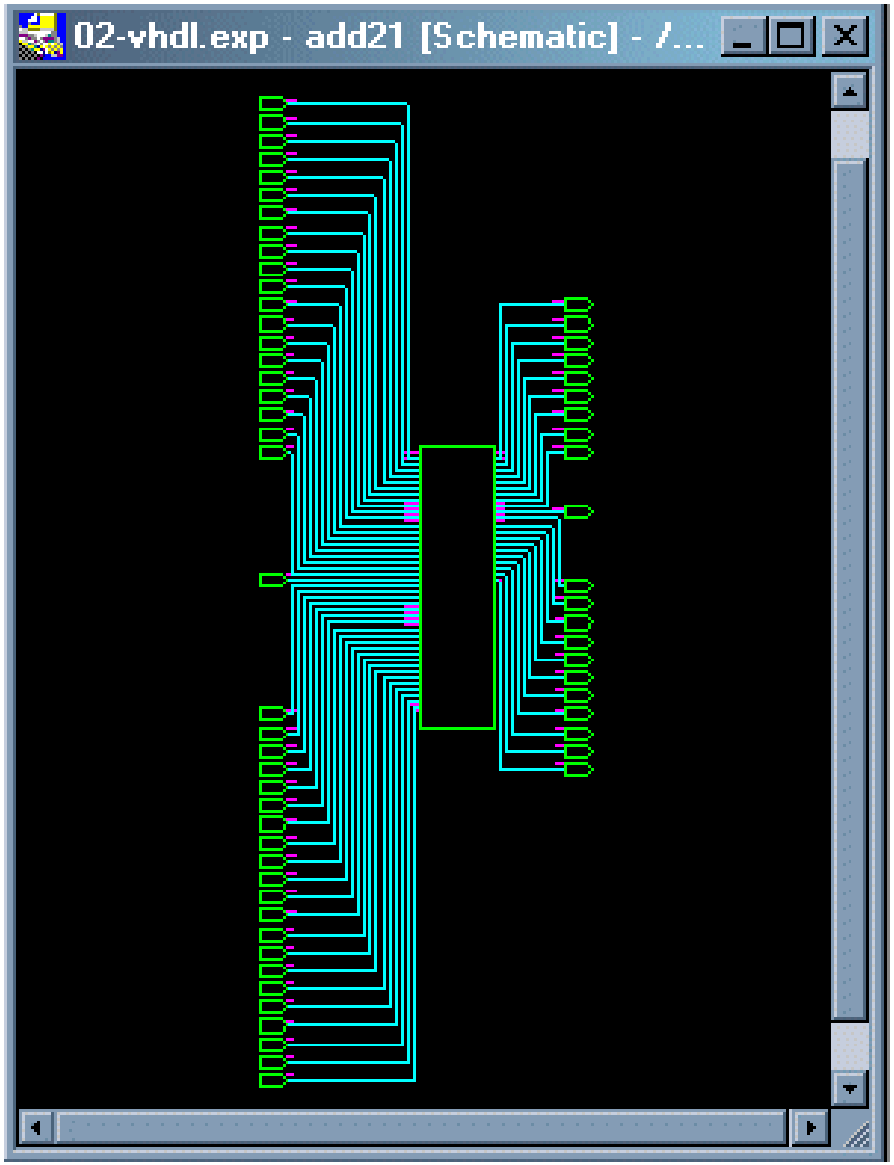
–adder – долно ниво – състои се от еднобитови суматори.

```
entity add21 is
   port (a, b : in std_logic_vector(21 downto 1);
         s    : out std_logic_vector(21 downto 1));
end add21;

architecture structural of add21 is
    component adderN
       generic(N : integer);
       port (a, b : in std_logic_vector(N downto 1);
          cin  : in std_logic;
          s    : out std_logic_vector(N downto 1);
          cout : out std_logic);
    end component;
    signal cin: std_logic;
begin
   U1: adderN generic map (N=>21) port map (a=>a, b=>b,
         cin=>cin, s=>s, cout=>open);
   cin <= '0';
end structural;
```

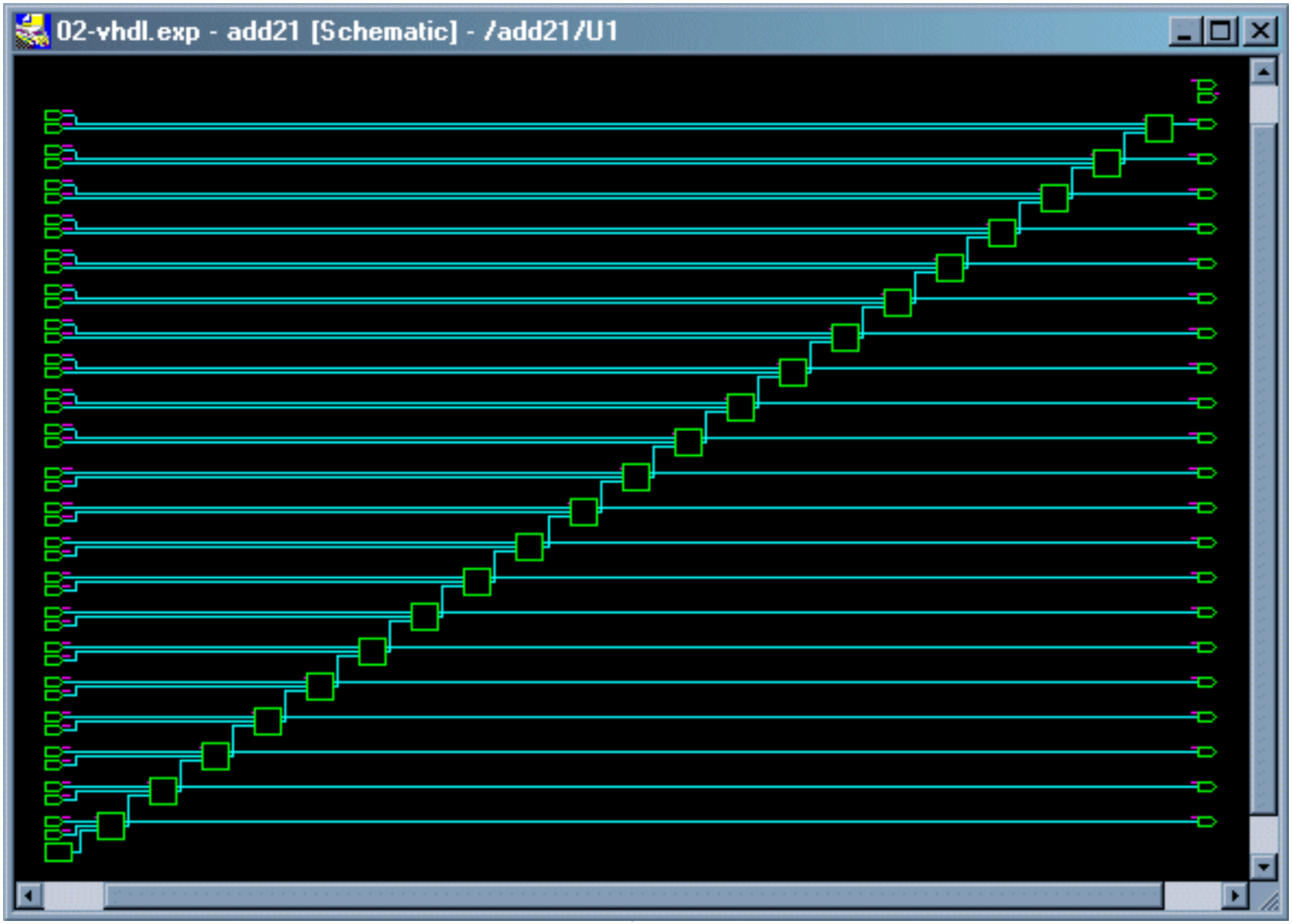
```
entity adderN is
    generic(N : integer);
    port (a, b : in std_logic_vector(N downto 1); cin  : in std_logic;
          s    : out std_logic_vector(N downto 1); cout : out std_logic);
end adderN;

architecture structural of adderN is
    component adder
        port (a, b, cin : in std_logic; s, cout   : out std_logic);
    end component;
    signal c : std_logic_vector(N downto 0);
begin
    c(0) <= cin; cout <= c(N);
    gen: for I in 1 to N generate
        add: adder port map( a=>a(I), b=>b(I), cin=>c(I-1), s=>s(I),
                cout=>c(I));
   end generate;
end structural;
```

```
entity adder is
  port (a,b,cin    : in std_logic;
        s,cout     : out std_logic);
end adder;

architecture rtl of adder is
begin
    s <= (a xor b) xor cin;
    cout <= (a and b) or (cin and a) or (cin and b);
end rtl;
```

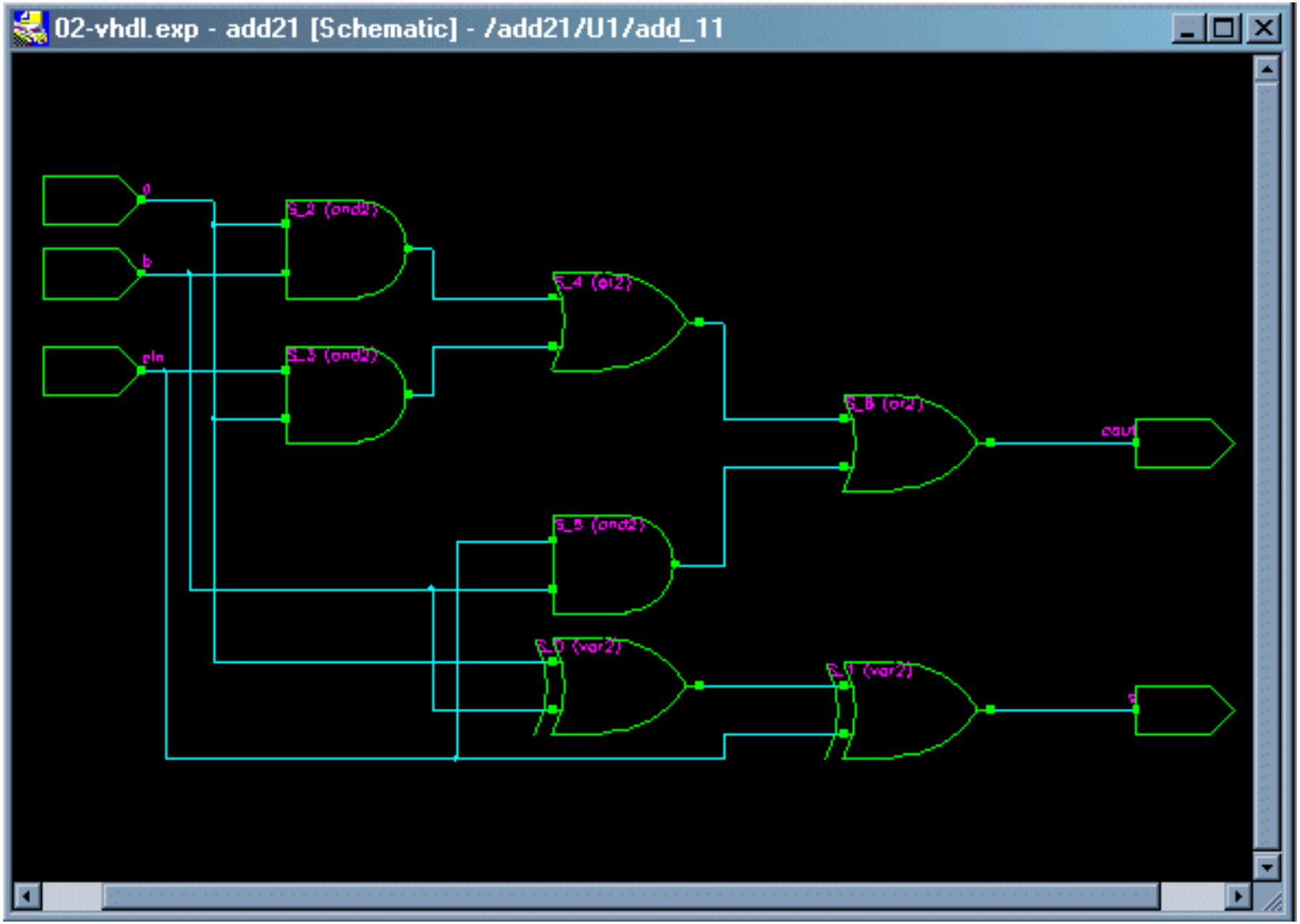
Синтез – add21



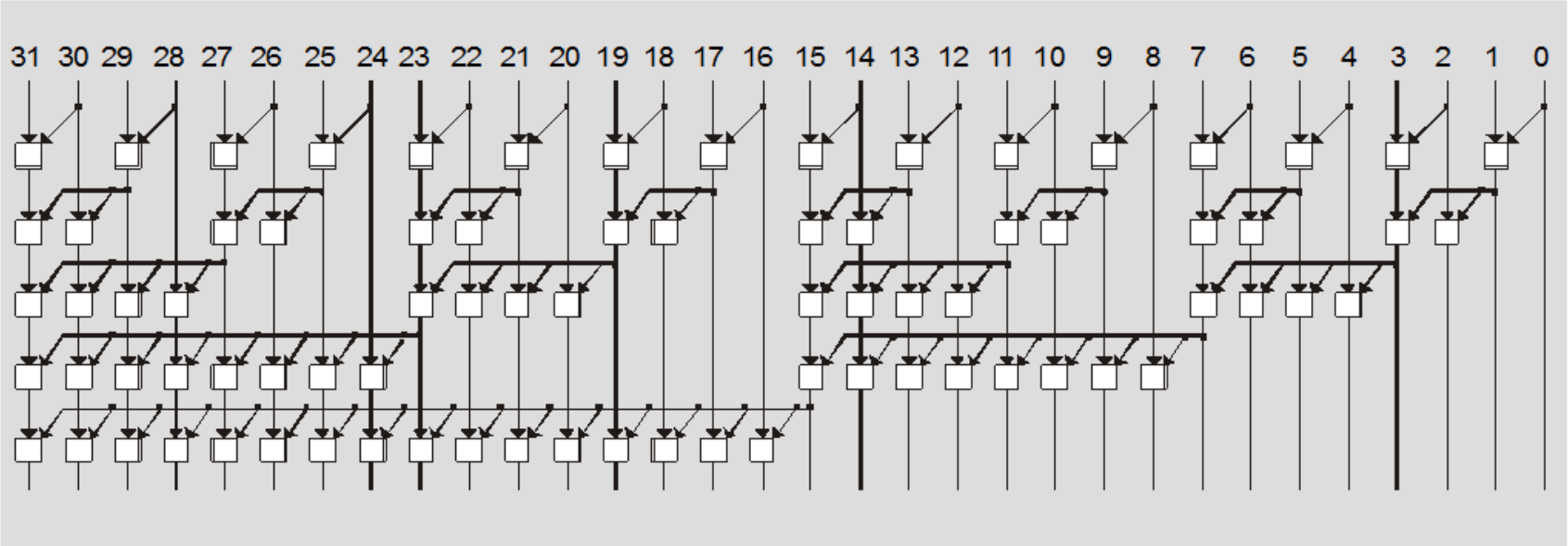
Синтез – adderN (N=21)



Синтез – adder



Пример – суматор на Sklansky



```
...
begin
   GEN1:
   for i in 0 to N-1 generate
      GEN2:
      if i = 0 generate
         U1:
         SQR_CIN port map (a => a(i), b => b(i), p => cp(i)(0),
               g => cg(i)(0), cin => cin);
      end generate;
      GEN3:
      if i > 0 generate
         U2:
         SQR port map (a => a(i), b => b(i),
             p => cp(i)(0), g => cg(i)(0));
      end generate;
   end generate;
   GEN4:
   for i in 0 to N-1 generate
      GEN5:
      for j in 1 to JMAX-1 generate
         GEN6:
         if bit_is_1(i,j-1) generate
            GEN7:
            if j > log2(i) generate
    U3: G port map (pa => cp(i)(j-1), ga => cg(i)(j-1),
                 gb => cg(i-i mod 2**(j-1)-1)(j-1),
                 gout => cg(i)(j));
            end generate;
    GEN8:
            if j <= log2(i) generate
    U4: GP port map (pa => cp(i)(j-1), ga => cg(i)(j-1),
                  pb => cp(i-i mod 2**(j-1)-1)(j-1),
                  gb => cg(i-i mod 2**(j-1)-1)(j-1),
                pout => cp(i)(j), gout => cg(i)(j));
    end generate;
         end generate;
         ...
```

Виж целият модел: sklansky.vhd