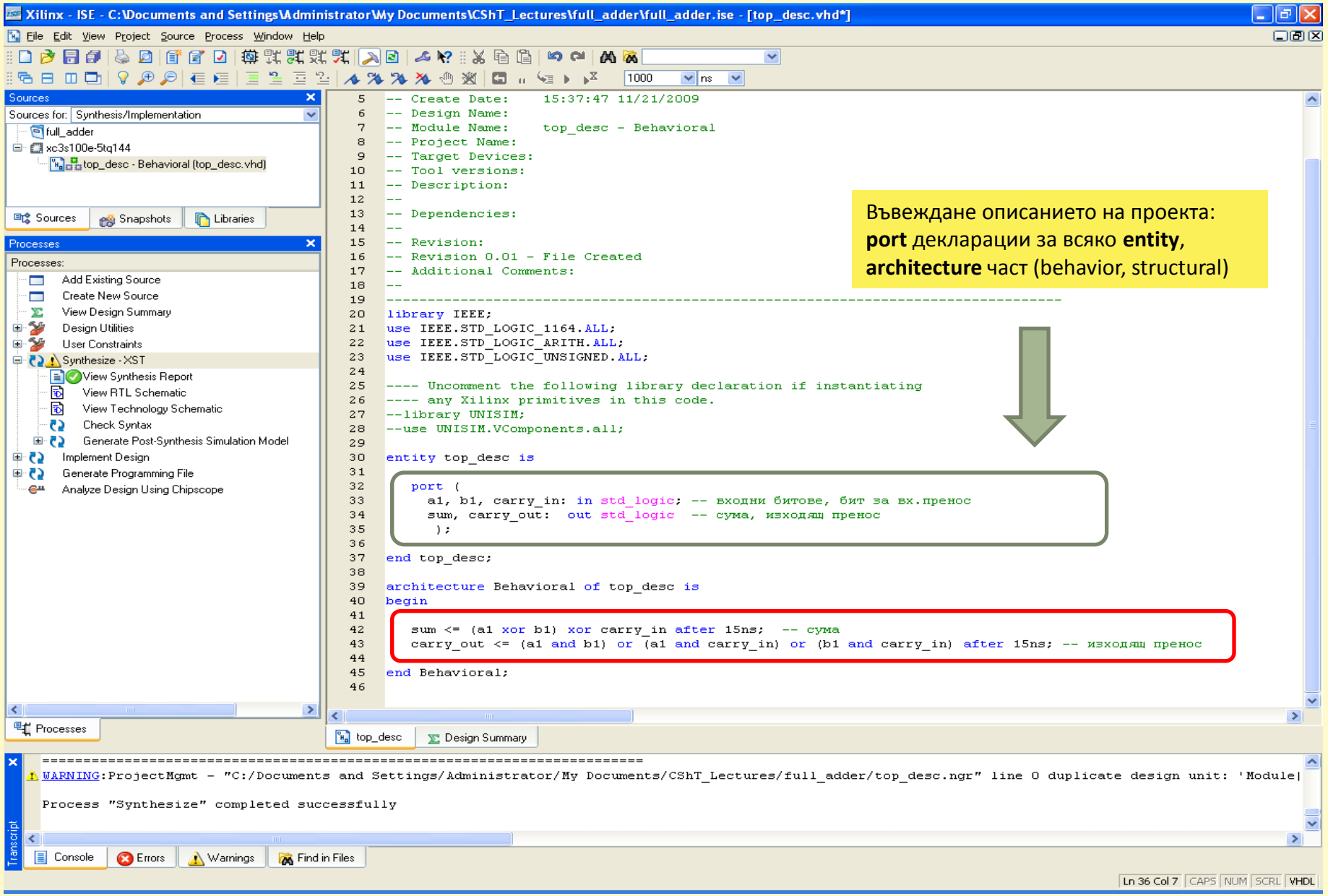
ЦИФРОВА СХЕМОТЕХНИКА

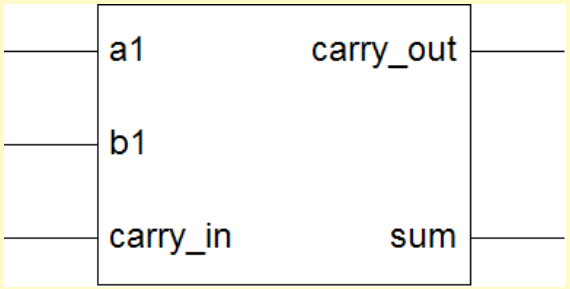
ЛЕКЦИЯ #12

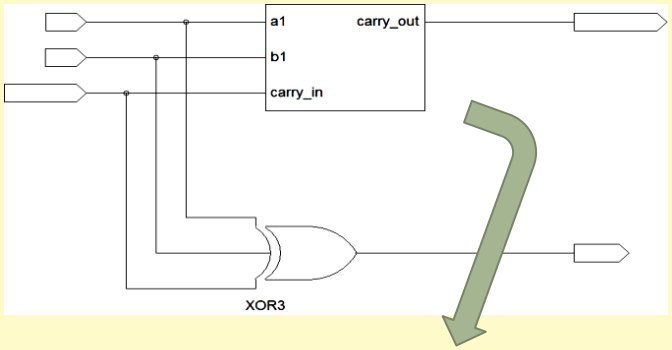
Развойна среда Xilinx ISE Project Navigator (Integrated Synthesis Environment) / WebPack
Разработка и имплементация на проект с протоипна платка BASYS/BASYS2 на Digilent



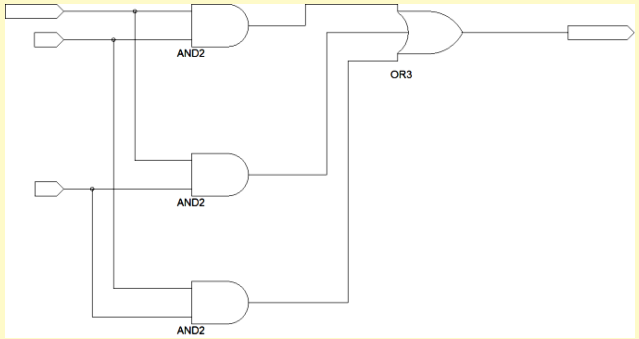
Развойна среда Xilinx ISE WebPack
Разработка и имплементация на проект с протоипна платка BASYS/BASYS2 на Digilent

Архитектурно описание (входове/изходи)

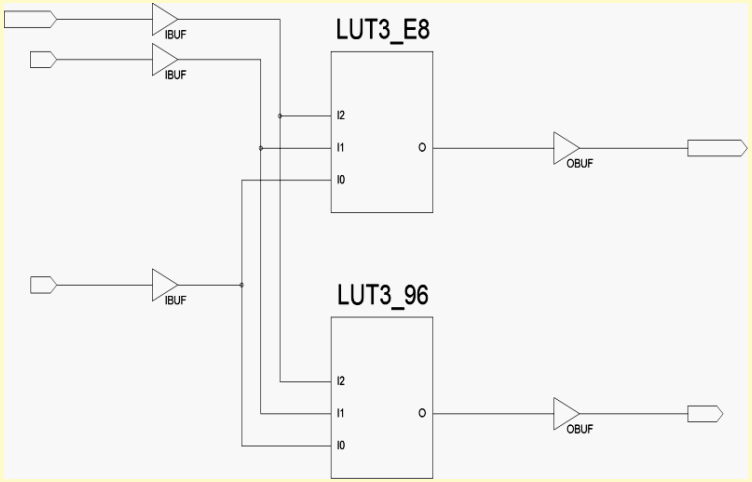




RTL- представяне (Schematic)



Представяне на ниво логически елементи (където е възможно)



Technology Schematic (показва кои точно компоненти от FPGA-чипа са използвани)

Развойна среда Xilinx ISE WebPack
Разработка и имплементация на проект с протоипна платка BASYS/BASYS2 на Digilent

TABLE OF CONTENTS


```
=========================================================
*                  Synthesis Options Summary            *
=========================================================
---- Source Parameters
Input File Name                    : "top_desc.prj"
Input Format                       : mixed
Ignore Synthesis Constraint File   : NO

---- Target Parameters
Output File Name                   : "top_desc"
Output Format                      : NGC
Target Device                      : xc3s100e-5-tq144

.......

---- Other Options
lso                                : top_desc.lso
Read Cores                         : YES
cross_clock_analysis               : NO
verilog2001                        : YES
safe_implementation                : No
Optimize Instantiated Primitives   : NO
use_clock_enable                   : Yes
use_sync_set                       : Yes
use_sync_reset                     : Yes
```

Отчет след синтеза на ЦУ: съдържание, част от пълния отчет, вкл. “Final Report”

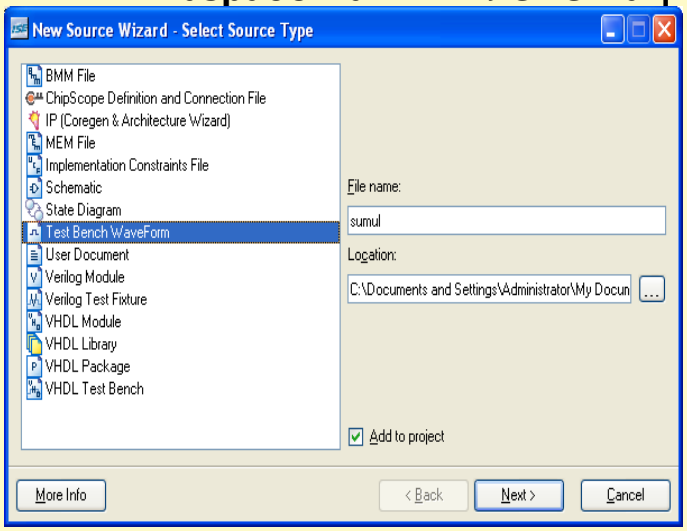
```
=========================================================
*                      Final Report                     *
=========================================================
Final Results
RTL Top Level Output File Name     : top_desc.ngr
Top Level Output File Name         : top_desc
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : NO

Design Statistics
# IOs                              : 5

Cell Usage :
# BELS                             : 2
#  LUT3                            : 2
# IO Buffers                       : 5
#     IBUF                         : 3
#     OBUF                         : 2
=========================================================
Device utilization summary:
---------------------------
Selected Device : 3s100etq144-5

 Number of Slices:              1  out of   960    0%
 Number of 4 input LUTs:        2  out of  1920    0%
 Number of bonded IOBs:         5  out of   108    4%
=========================================================
TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
------------------
No clock signals found in this design
Timing Summary:
Speed Grade: -5
   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 6.465ns
```

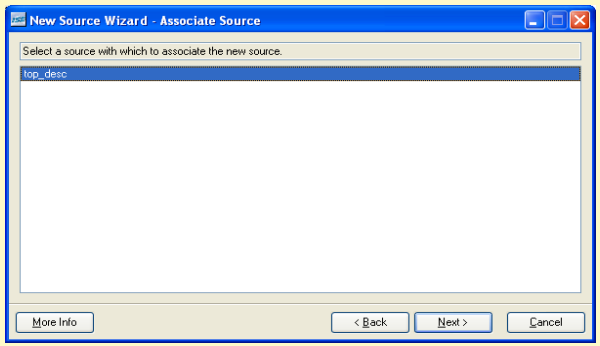
Развойна среда Xilinx ISE WebPack
Разработка и имплементация на проект с протоипна платка BASYS/BASYS2 на Digilent

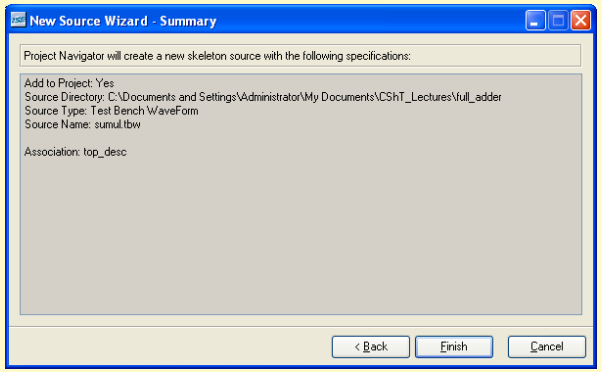
```
==========================================================================
*                      Final Report                      *
==========================================================================
. . . . .
Device utilization summary:
---------------------------
Selected Device : 3s100etq144-5

 Number of Slices:              1  out of    960    0%
 Number of 4 input LUTs:        2  out of   1920    0%
 Number of bonded IOBs:         5  out of    108    4%
==========================================================================
TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
------------------
No clock signals found in this design

Timing Summary:
---------------
Speed Grade: -5

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 6.465ns
```

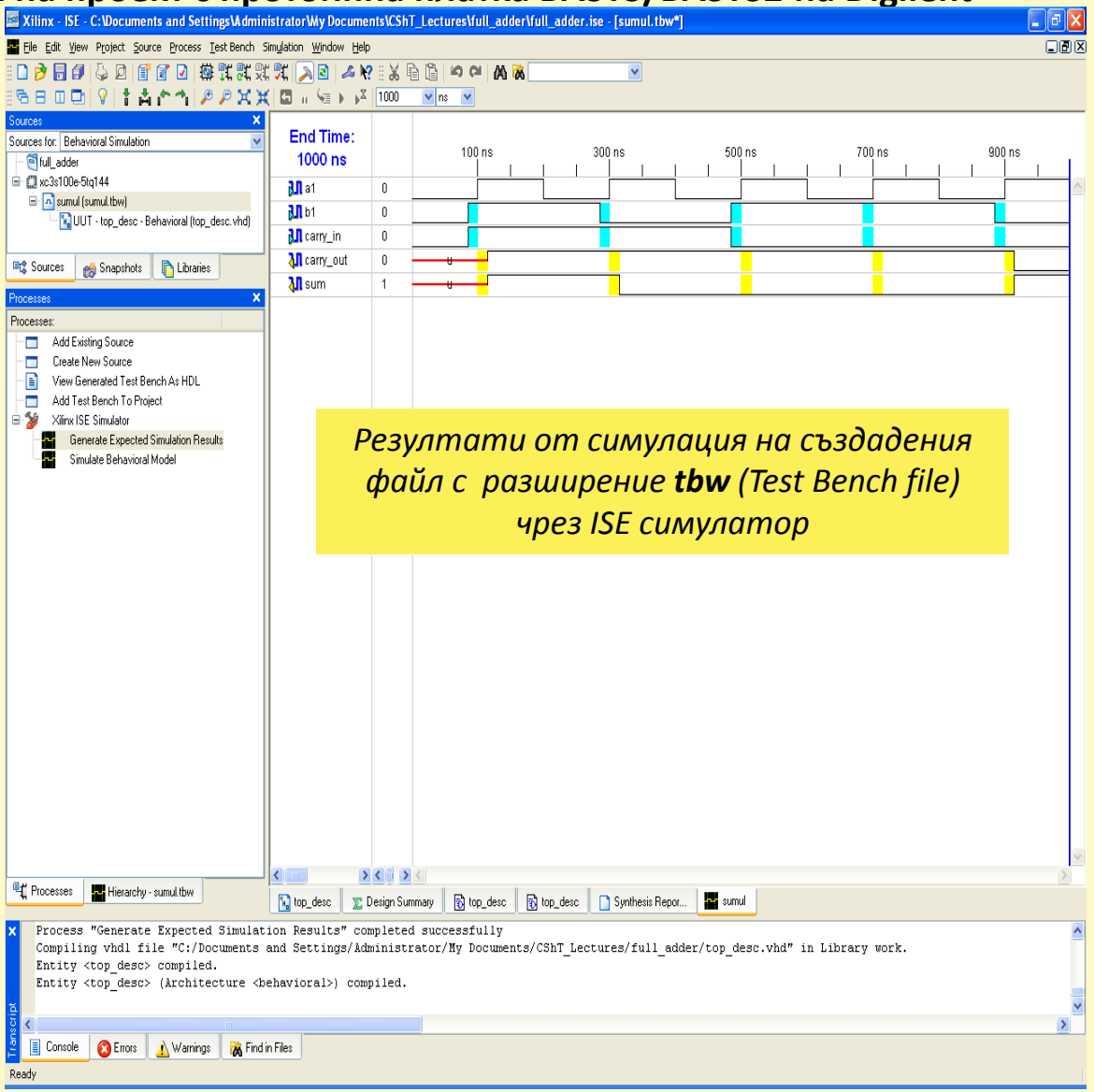
Развойна среда Xilinx ISE WebPack

Разработка и имплементация на проект с протоипна платка BASYS/BASYS2 на Digilent









*Резултати от симулация на създадения файл с разширение **tbw** (Test Bench file) чрез ISE симулатор*

VHDL – особености на езика. Типове, оператори, конструкции

Проектиране на логическо ниво (Gate-level). Проектиране на комбинационни схеми с базови блокове. Разширяване размерността с използване на вече проектирани блокове

Пример: ДШ (decoder) 2→4

- определяне **уравненията (състоянията) на изходите** на КС;
- **HDL описание** /особености на синтакиса, оператори/;
- **симулация**: testbench файл (tbw);
- създаване на **implementation constraints file** (**ucf**);
- **прехвърляне на проекта** върху FPGA чипа;
- **тестване** на прототипа.

А) използване на изградения модул ДШ 2→4 за създаване на ДШ 3→8; симулация, прототипизация.

Б) използване на изградения модул ДШ 2→4 за създаване на ДШ 4→16; симулация, прототипизация.

Особености на синтеза:

- използва само логически и аритметични операции за HDL описанието;
- използва базови структури (модули): суматори, компаратори, умножители, мултиплексори (това са всъщност основните FPGA градивни структури – т.е. прави се проектиране на регистрово ниво - RTL design)

VHDL – особености на езика. Типове, оператори, конструкции

Проектиране на логическо ниво (Gate-level). Проектиране на комбинационни схеми с базови блокове. Разширяване размерността с използване на вече проектирани блокове

❑ **Оператори за сравнение:**

`a = b`	равно
`a /= b`	неравно
`a < b`	по-малко
`a > b`	по-голямо
`a >= b`	по-голямо или равно
`a <= b`	по-малко или равно

Сравняват операнди от един и същ тип и връщат резултат от булев тип (boolean)

❑ **Аритметични операции:**

❖ типове данни (операнди) – `integer, natural` `signed, unsigned` (масив от std_logic елементи, в пакета **ieee.numeric_std.all**)

❖ пакет (package) от библиотеката ieee: **ieee.numeric_std.all**

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

`a + b`	събиране
`a – b`	изваждане
`a * b`	умножение (по-особена, в FPGA има отделни умножители, в Spartan3E 18x18 умножение, предварително заложени, ограничен брой).

VHDL – особености на езика. Типове, оператори, конструкции

Проектиране на логическо ниво (Gate-level). Проектиране на комбинационни схеми с базови блокове. Разширяване размерността с използване на вече проектирани блокове

В пакета **std_logic_1164**

`a ** b`	повдигане в степен
`a * b`	умножение
`a / b`	деление
`a + b`	събиране
`a - b`	изваждане

- **Преобразуване на типа**

Типовете `std_logic_vector, signed, unsigned` – РАЗЛИЧНИ, независимо че представляват съвкупност от елементи от тип std_logic !!! Преобразуване на типа на сигналите (**conversion function / type casting**):

Тип на сигнала `a`	Преобразуване в тип	Функция (conversion function)
`unsigned,signed`	`std_logic_vector`	**`std_logic_vector(a)`**
`signed,std_logic_vector`	`unsigned`	**`unsigned(a)`**
`unsigned,std_logic_vector`	`signed`	**`signed(a)`**
`unsigned,signed`	`integer`	**`to_integer(a)`**
`natural`	`unsigned`	**`to_unsigned(a,size)`**
`integer`	`signed`	**`to_signed(a, size)`**

`std_logic_vector` НЕ МОЖЕ да се преобразува директно в тип `integer`

VHDL – особености на езика. Типове, оператори, конструкции

Проектиране на логическо ниво (Gate-level). Базови блокове комбинационни структури

Примери за преобразуване на типа:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
....
signal   in_s1, in_s2, in_s3: std_logic_vector(3 downto 0);
signal   in_u1, in_u2, in_u3: unsigned(3 downto 0);
 ....
in_s3 <= 8          -- неправилно
in_s2 <= in_u3   -- неправилно
in_u1 <=  unsigned(in_s1)    -- правилно
in_u2 <=unsigned(5,4)         -- правилно

in_u3 <= in_u1 + in_u2        -- правилно, от един и същ тип
```

Други (не ieee) аритметични пакети:
std_logic_arith (подобно на std_numeric),
std_logic_unsigned, std_logic_signed : въвеждат работа върху данни тип std_logic_vector (предефинирани, без конверсия).

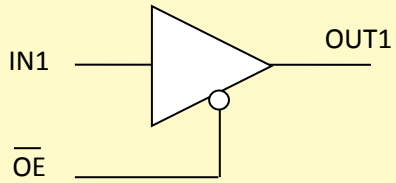
VHDL – особености на езика. Типове, оператори, конструкции

Проектиране на логическо ниво (Gate-level). Базови блокове комбинационни структури

- **Concatenation** оператор & :

```
signal   a1: std_logic;
signal   a4: std_logic_vector(3 downto 0);
signal   a8, b8, c8: std_logic_vector(7 downto 0)
…
b8 <= a4 & a4;
c8 <=  a4 & ‘00’ & a1 & a1;
```

- **Дефиниране на ВИС (High Impedance, Z-state):**



OE	OUT1
1	Z (H.I.)
0	IN1

- реализация: чрез от буфер с ВИС

VHDL описание:

```
OUT1 <=  IN1  when  OE=‘0’  else  ‘Z’;
```
-- повторител с ВИС разрешен с OE=L

- физически буфери – <u>само в I/O блоковете</u> и асоциирани с изводите на чипа

VHDL – особености на езика. Типове, оператори, конструкции

Проектиране на логическо ниво (Gate-level). Базови блокове комбинационни структури

❑ Оператор за условно присвояване стойност на сигнал:

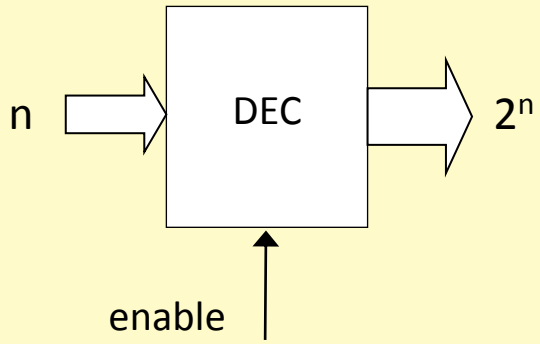
<име на сигнал> **<=** *<аритм.израз_1>* **when** *<булев израз_1>* **else**
<аритм.израз_2> **when** *<булев израз_2>* **else**
. . . .
<аритм.израз_n> ;

Пример: calc_ex **<=** a1 + b1 + e1 **when** m>n **else**
c1 **when** m<n **else**
d1; -- за неописаните условия т.е. при m=n

! Трасиране на път : реализация чрез MUX 2→1 (един или няколко).

Пример: ДШ 2→4. Структура, I/O променливи, VHDL код (**архитектура чрез условен оператор**).

```
library ieee
use ieee.std_logic_1164.all
entity  dec_2_to_4  is
        port (
                a:   in std_logic_vector(1 downto 0);
                y:   out std_logic_vector(3 downto 0);
                en: in std_logic_vector
                );
end  dec_2_to_4 ;
```



VHDL – особености на езика. Типове, оператори, конструкции

Проектиране на логическо ниво (Gate-level). Базови блокове комбинационни структури

Пример: ДШ 2→4 (продължение)

```
architecture cond_dist of dec_2_to_4 is
begin
    y <=  "0000" when (en='0') else   -- липсва разрешение /ДШ разрешен при en=1/
          "0001" when (a='00') else
          "0010" when (a='01') else
          "0100" when (a='10') else
          "1000";
end cond_dist;
```

❑ **Пренасочване със "селекторен" оператор**:

```
with <селектор> select
    <име на сигнал> <= <аритм.израз_1> when <селектор_1> ,
                       <аритм.израз_2> when < селектор_2> ,
                       . . . .
                       <аритм.израз_n> when others;
```

Пример:

```
signal sel_key: std_logic_vector(1 downto 0);
…
with sel_key select
        res <= a1+b1 when "00",
               a1-b1 when "01",
               c1    when others;  -- т.е. При "10" и "11"
```

VHDL – особености на езика. Типове, оператори, конструкции

Проектиране на логическо ниво (Gate-level). Базови блокове комбинационни структури

Пример: ДШ 2→4. VHDL код (**архитектура чрез селекторен оператор**)

```
architecture  selec_dist   of  dec_2_to_4  is
signal  sel_k: std_logic_vector(2 downto 0);
    begin
        sel_k <= en & a;    -- обединяване на  en  и входният вектор  a
        with sel_k  select
                y <= "0000" when "000"|"001"|"010"|"011",   -- при липса на разрешение
                     "0001" when "100",
                     "0010" when "101",
                     "0100" when "110",
                     "1000" when  others;   -- т.е. само при "111"
end  selec_dist;
```

VHDL – особености на езика. Типове, оператори, конструкции

Проектиране на логическо ниво (Gate-level). Базови блокове комбинационни структури

❑ **Описание чрез процес (process)**:

! Съвкупност от операции, изпълнявани съвместно в тялото на отделна конструкция: "процес". Съответства най-добре на модела на функциониране на ЦС (най-често тактувани структури, но описва и асинхронни блокове).

Синтаксис:

```
process (sensitivity list)   -- списък от променливи към които процесът е
                             -- "чувствителен" /включва една или повече пром./
begin
. . .
последователностни оператори
. . .
end process;
```

Последователностни оператори: *сигнал* **<=** *стойност на израз*

Пример:

```
process (e1, e2)
begin
  e3 <= e1 or e2;
  e4 <= e1 and e2;
  e4 <= e1 or e2;  -- допустимо става e4=e1 or e2 игнорира предходното присвояване
end process;
```

VHDL – особености на езика. Типове, оператори, конструкции

Проектиране на логическо ниво (Gate-level). Базови блокове комбинационни структури

- Условни оператори: **if** , **case**
  - Оператор **if …elsif…else… end if (if… else…endif)**

Пълен синтаксис:
(при множествена проверка)

```
if   <булев израз_1>  then
     последователностни оператори ;
elsif   <булев израз_2>  then
     последователностни оператори ;
elsif   <булев израз_3>  then
     последователностни оператори ;
. . .
else
     последователностни оператори ;
end if;
```

Опростен синтаксис:
(при само 1 проверка)

```
if   <булев израз>  then
   последователностни оператори ;
else
   последователностни оператори ;
end if;
```

VHDL – особености на езика. Типове, оператори, конструкции

Проектиране на логическо ниво (Gate-level). Базови блокове комбинационни структури

- Пример с условен оператор **if**

```
process (a1, b1, c1,d1,e1,m,n)
begin
  if  m>n  then
        calc_ex   <=    a1 + b1 + e1
  elsif  m<n  then
        calc_ex <= c1;
  else
        calc_ex <= d1;
  end  if;
end process;
```

```
calc_ex   <=   a1 + b1 + e1   when   m>n   else
               c1             when   m<n   else
               d1;    -- при m=n
```

❖ДШ 2→4 описан с оператор **if**

```
architecture  if_arch   of   dec_2_to_4  is
begin
  process (en, a)
       if  (en='0')  then  y <= "0000";     -- забрана при en=0
          elsif  (a='00') then y <= "0001";
          elsif  (a='01') then y <= "0010";
          elsif  (a='10') then y <= "0100";
          else y <= "1000";
       end if;
  end process;
end if_arch;
```

VHDL – особености на езика. Типове, оператори, конструкции

Проектиране на логическо ниво (Gate-level). Базови блокове комбинационни структури

- Оператор **case**

Синтаксис:

case <селектор> **is**
  when <селектор_1> =>
    последователностни оператори **;**
  when <селектор_2> =>
    последователностни оператори **;**
  . . .
  when others =>
    последователностни оператори **;**
end case;

Пример:

```
process (a1, b1, sel_key)
begin
  case sel_key is
      when "00" =>
                  res <= a1+b1;
      when "01" =>
                  res <=a1-b1;
      when others =>
                  res <=c1;
  end case;
end process;
```

```
with sel_key select
            res <= a1+b1 when "00",
                   a1-b1 when "01",
                   c1    when others;
```

VHDL – особености на езика. Типове, оператори, конструкции

Проектиране на логическо ниво (Gate-level). Базови блокове комбинационни структури

- Константи във VHDL
  - използване: при еднотипен дизайн с променящи се параметри;
  - предимства: читаем код, лесна вариативност;
  - деклариране – в **architecture** тялото.

Синтаксис:

constant *име на константа*: *тип* **:=** *стойност* **;**

Примери:

```
constant  M_AN: integer:= 128;
constant  D_BIT: integer:= 8;
constant  D_RANGE: integer:= 2**D_BIT-1;      -- преизчислява D_RANGE=255
                                              -- при D_BIT вече дефинирано
```

VHDL – особености на езика. Типове, оператори, конструкции

Проектиране на логическо ниво (Gate-level). Базови блокове комбинационни структури

- Константи във VHDL

Приложение (пример):

```
library ieee;
use  ieee.std.logic_1164.all;
use ieee.numeric_std.all;

entity adder_carry_out  is
        port (
           in1, in2:  in std_logic_vector (3 downto 0);      -- входни операнди с
                                                             -- размерност 4
           cout:  out std_logic;                             -- изходящ пренос
           sum:  out std_logic_vector (3 downto 0);          -- сума размерност 4 без
                                                             -- пренос
                );
end adder_carry_out;
```

VHDL – особености на езика. Типове, оператори, конструкции

Проектиране на логическо ниво (Gate-level). Базови блокове комбинационни структури

- Константи във VHDL

Приложение (пример, продължение):

```
architecture const_use  of  adder_carry_out  is

    constant N: integer:= 4;
    signal in1_ext, in2_ext, sum_ext: unsigned (N downto 0);    -- размерност 5 бита

begin
    in1_ext  <=unsigned ('0' & in1);          -- разширяване с 1 бит до 5 бита
    in2_ext  <= unsigned ('0' & in2);         -- разширяване с 1 бит до 5 бита
    sum_ext <= in1_ext + in2_ext;             -- междинна сума
    sum <=  std_logic_vector (sum_ext (N-1 downto 0));   -- крайна сума без
                                                         -- последния бит
    cout <= sum_ext(N);                       -- изходящ пренос е MSB бита
end const_use;
```

VHDL – особености на езика. Типове, оператори, конструкции

Проектиране на логическо ниво (Gate-level). Базови блокове комбинационни структури

- Конструкция **Generic**

- приложение: алтернатива на **constant**
- задаване: в **entity** декларативното тяло, преди **port**.

Синтаксис:
```
entity име_на_entity is
generic (
  име_на_generic : тип := стойност ;
  . . .
  име_на_generic : тип := стойност ;
        );
port (
   . . .
      );
end име_на_entity ;
```