ЦИФРОВА СХЕМОТЕХНИКА

ЛЕКЦИЯ #11

Компании разработващи продукти за проектиране с PLD

- Основни компании, разработващи и пълна гама интегрирани продукти (чипове, среди, технологии за синтез, развойни средства) за проектиране на цифрови устройства с PLD:

❑ Xiinx - http://www.xilinx.com/ Основана 1984, световен лидер в сегмента (прибл. 50% дял), предлага всички видове програмируеми FPGAs, SoCs , 3D ICs. Собственик на над 2500 патента; технологии 28nm/20nm.

- FPGA продукти/серии:

  6 Series FPGA Families (Virtex®-6, Spartan®-6, EasyPath™-6);
  Automotive Device Families (XA Spartan-6/3A/3A DSP/3E);
  Additional Device Families (Virtex-5/4/3A/3AN/3ADSP/Spartan-3E);
  Space-grade Device Families (Space-grade Virtex-5QV, Virtex-4QV);
  Defense-grade Device Families (Defense-grade Artix™-7Q,Kintex™-7Q, Virtex-7Q, Virtex-6Q,Spartan-6Q,Virtex-5Q ,Virtex-4Q.

- CPLD продукти/серии:

  CoolRunner-II;
  Automotive CPLD Device Families (XA 9500XL, XA CoolRunner™-II);
  Additional CPLDs - XC9500XL.

Компании разработващи продукти за проектиране с PLD

❑ Altera - http://www.altera.com/ Създадена през 1984, водеща в производството на продукти и технологии за синтез с PLD продукти, IP cores, развойни средства. Работи съвместно с Taiwan Semiconductor Manufacturing Company (TSMC). Технология – до 28nm.

- FPGA продукти: Cyclone (low-cost)l Stratix (midrange), Arriva (high-end):

	Cyclone FPGA	Cyclone II FPGA	Cyclone III FPGA	Cyclone IV FPGA	Cyclone V FPGA
Year introduced	2002	2004	2007	2009	2011
Process technology	130 nm	90 nm	65 nm	60 nm	28 nm

Device Family	Stratix	Stratix GX	Stratix II	Stratix II GX	Stratix III	Stratix IV	Stratix V
Year of introduction	2002	2003	2004	2005	2006	2008	2010
Process technology	130 nm	130 nm	90 nm	90 nm	65 nm	40 nm	28 nm

	Arria GX FPGA	Arria II GX FPGA	Arria II GZ FPGA	Arria V GX, GT, SX, ST FPGA	Arria V GZ FPGA
Year of introduction	2007	2009	2010	2011	2012
Process technology	90 nm	40 nm	40 nm	28 nm	28 nm

- CPLD продукти: MAX II CPLD, MAX IIG CPLD, MAX IIZ CPLD.

Среди за проектиране и прототипизация с PLDs/HDLs

❑ Среди за проектиране на Xiinx:

- **ISE Design Suite:**
  - Logic Edition / Embedded Edition
  - DSP Edition / System Edition;
  - ISE WebPACK Design Software /безплатен продукт/.

- **Vivado Design Suite:** Революционна IP и системно центрирана среда (разработва се в период от 4 години) за интеграция и ускорение на дизайна на 'All Programmable' FPGAs, SoCs и 3D ICs.

❑ Среди за проектиране на Altera:

- **Quartus II** Subscription Edition Software;
- **Quartus II** Web Edition Software;

- ModelSim /симулатор, версия на Altera/.

- **Nios II Embedded Design Suite** (EDS) (за Embedded design).

- **DSP Builder** (за DSP приложения).

Езици за описание на цифрови структури – HDLs. Език VHDL

- VHDL : възниква в резултат на необходимост от проектиране на VHSIC (Very High-Speed Integrated Circuit) HDL (Hardware Description Language):
  - ❑ разработен от IBM, Intermetrics и TI по идея на Министерството на отбраната на САЩ ,
  - ❑ по-късно е предоставен на IEEE и въведен чрез стандарта IEEE Standard 1076, който е ратифициран през 1987 като VHDL87, многократно модифициран и обогатяван като език в последствие (последно IEEE Standard 1164): 1992,1994,2002;
  - ❑ предназначение: за описание и моделиране на цифрови структури и системи на различни нива на абстракция.

  Други HDL езици: Verilog, ABEL, SystemC, SpecC, System Verilog.

  Поддържани в Xilinx ISE WebPack компилатори за: VHDL, Verilog, ABEL.

Езици за описание на цифрови структури – HDLs. Език VHDL

HLL езици (High Level Languages): езици от високо ниво: C, C++, Java, Pascal, Fortran и др. – неподходящи за ефективно описание (моделиране) на хардуерни (цифрови) структури, поради:

- не поддържат конструкция “процес” – **process** – аналогия на хардуерен модул/блок и синхронизация;
- не дефинират **signals** – аналогия на шина/линия;
- не поддържат (в повечето случаи) задължителните за описание в хардуер логически конструкции: **OR, AND, XOR, XNOR, SHIFT**;
- не дефинират логически нива: **0, 1, Z, X, U**;
- нямат декларация **'event** с която се определя промяната на сигнал
- не специфицират време (закъснение), напр. при присвояване на стойност след определено време. Пример: output_K <= SUM **after** 15ns

! HLL езиците не могат практически да се ползват за ефективно описание на цифрови структури

Език за описание на цифрови структури VHDL Стилове. Основни констукции, ключови думи

- Акцент върху **описанието на базови цифрови структури и йерархично описание на системи в аспект проектиране на цифрово устройство и неговата имплементация** върху FPGA/CPLD матрица, както и на проверката на прототипа.

VHDL предлага различни подходи (стилове) за описание цифрови схеми с оглед оптимизиране на кода и/или дефиниране на йерархични (модулни) конструкции.

- Стилове за описание (моделиране) на архитектурата:

→**Behavioral** (поведенческо) – моделирането се основава на описание реакцията на възли / изходи по отношение промяната на входните сигнали;

→**Structural** (структурно) – описанието се базира на описание модулите, изграждащи структурата;

→**Mixed** (смесено) – представлява съвкупност от двата стила, описани по-горе.

Език за описание на цифрови структури VHDL

Стилове. Основни констукции, ключови думи

Дефиниране на ЦС (Design Unit, блок/единица) във VHDL → **Entity**

! Всяко **Entity** се състои от декларационна част (Entity Declaration) и архитектура (Entity Architecture) – една или повече.

Всяка Architecture съдържа модел на Entity. При повече от една описани архитектури се ползва *последната дефинирана.*

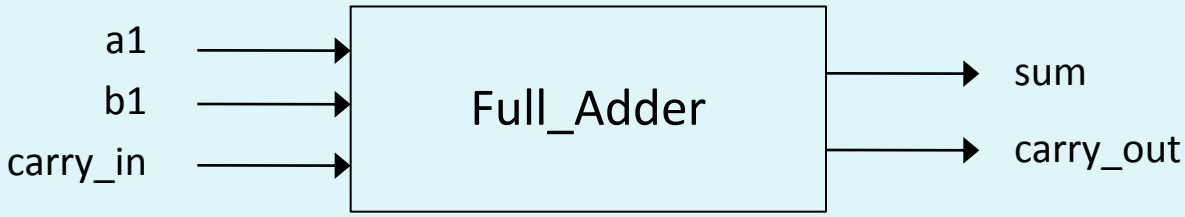
Деклариране на entity :

```
entity <име> is    -- коментар
   port (
          x1, x2,x3:  <режим> <тип>;  -- входни сигнали
          y1, y2:  <режим> <тип>       -- изходни сигнали
         );
end <име> ;
```

Език за описание на цифрови структури VHDL

Стилове. Основни констукции, ключови думи

Деклариране на entity (пример: 1-битов пълен суматор):




```
library ieee;
use ieee.std_logic_1164.all
entity  Full_Adder  is    -- начало на entity дефиницията
    port (
            a1, b1, carry_in:  in std_logic; -- входни битове, бит за вх.пренос
            sum, carry_out: out std_logic -- сума, изходящ пренос
          );
 end Full_Adder;
```

Език за описание на цифрови структури VHDL

Стилове. Основни констукции, ключови думи

library ieee -- *библиотека*

use ieee.std_logic_1164.**all** -- *пакет (package) от ieee библиотеката*
-- включва всички основни типове данни,
-- функции, оператори от езика VHDL

-- начало на Entity дефиницията *! Коментар (не се компилира)*

- *ключовите думи – bold, напр.* **port, entity** и т.н.
- езикът VHDL е “case insensitive” – т.е. не различава малки и големи букви;
- езикът VHDL е “free formatting” – т.е. могат да се вмъкват неограничен брой празни интервали (игнорират се);
- имената на обекти – до 26 символа, започващи с буква (може да включва цифри и ‘ _ ’);
- режим (mode) на сигналите: **in**, **out**, **inout** (двупосочни).

Език за описание на цифрови структури VHDL

Стилове. Основни констукции, ключови думи

✓ Специфични типове данни и оператори:

Тип **std_logic** - дефиниран е в пакета std_logic_1164. Състои се от 9 типа за дефиниране стойността на сигнала:

- ‘0’, ‘1’, ‘Z’ – 1/0/ВИС;
- ‘U’, ‘X’ – недефиниран, неустановен;
- ‘H’, ‘W’ (не ги ползваме – при разширение на езика).

Дефиниране на векторни сигнали – тип **std_logic_vector** – едномерен масив от std_logic елементи:

```
reg_port_A: in  std_logic_vector (15 downto 0) -- за предпочитане;
reg_port_A: in  std_logic_vector (0  to 15);
```

Дефиниране на част от векторен сигнал (някои битове):

```
reg_port_B: in  std_logic_vector (4 downto 0); -- само най-младшите 5 бита
```

Логически оператори: **and**, **or, xor** и т.н. Приоритет. Предефиниране на приоритета – чрез скоби: (a1 **and** a2) **or** (b1 **and** b2)

Език за описание на цифрови структури VHDL

Описание на архитектурата на ЦС

- Поведенческо описание:

```
architecture behavior of Full_Adder is    -- дефиниране на архитектурата
begin
    sum <= (a1 xor b1) xor carry_in after 15ns;    -- определяне бита на сумата
    carry_out <= (a1 and b1) or (a1 and carry_in) or (b1 and carry_in) after 15ns;
                                        -- определяне бита за изходящ пренос
end behavior;
```

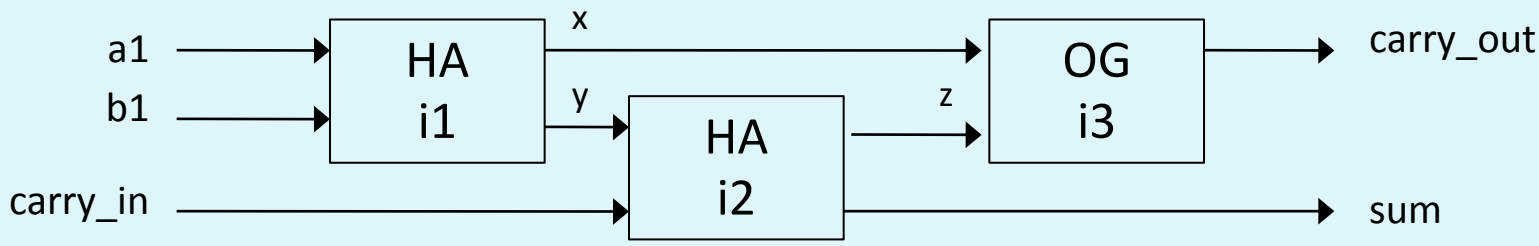
a1	b1	carry_in	sum	carry_out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Език за описание на цифрови структури VHDL

Описание на архитектурата на ЦС

- Структурно описание:

```
architecture struct of Full_Adder is   -- дефиниране на архитектурата
signal x,y,z: bit; -- локални променливи
begin
        i1: entity half_adder(behav)
            port map (a1, b1, x, y);
        i2: entity half_adder(behav)
            port map (y, carry_in, z, sum);
        i3: entity or_gate(behav)
            port map (x, z, carry_out);
end struct;
```



Език за описание на цифрови структури VHDL
Описание на архитектурата на ЦС

- Структурно описание:

```
architecture behav of half_adder is   -- описание архитектурата на полусуматора
begin                                 -- HA (half adder)
      sum <= a xor b after 15ns;
      carry_out <= a and b after 15ns;
end architecture behav;

architecture behav of or_gate is      -- описание архитектурата на ЛЕ ИЛИ
begin                                 -- OG (OR gate)
      out <= a or b after 15ns;
end architecture behav;
```

Език за описание на цифрови структури VHDL

Описание на архитектурата на ЦС

- Описание с използване на процес(и):

```
architecture behav of d_latch is    -- описание архитектурата на D-тригер (latch)
begin
      latch_behavior: process is
      begin
          if CLK='1' then  -- синхронизация по ниво
             Q <= D after  8ns;
          end if;
          wait on CLK, D;    -- sensitivity list
       end process latch_behavior;
end architecture behav;
```

wait – не е задължителен. Служи да се прекрати процес. Разновидности:

wait on <списък сигнали> -- до промяна на 1 от сигналите в списъка

wait until <условие> -- до изпълнение на условието

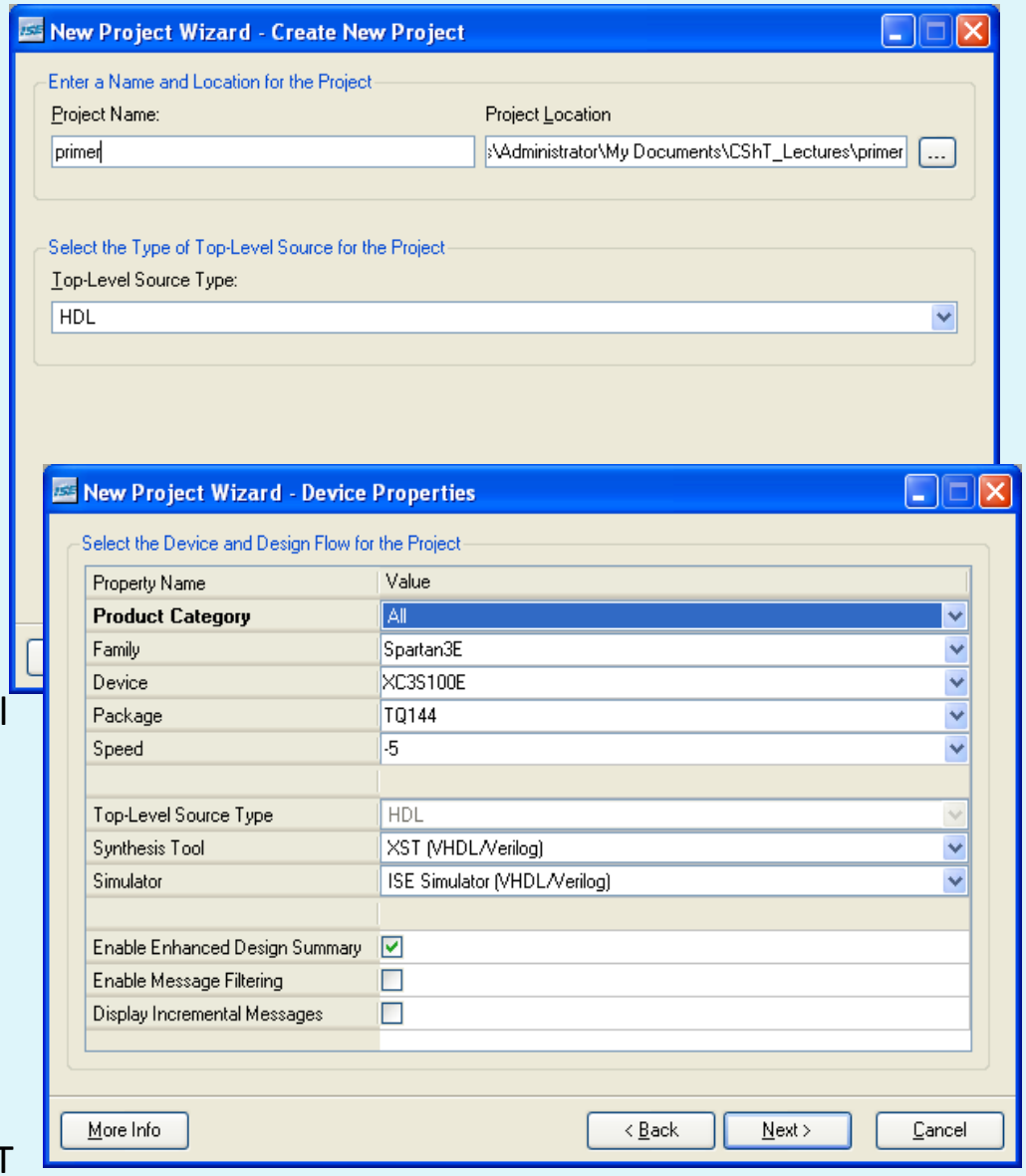
wait for <продължителност / време> -- за определен период време

wait -- безусловно (неопределено).

Развойна среда Xilinx ISE Project Navigator (Integrated Synthesis Environment) / WebPack

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

1. Дефиниране на проекта:
   - **име на проекта** (Project Name)
   - разположение (Project Location);
   - **тип описание** във файла от най-високо йерархично ниво (Top-Level Source file): HDL, Schematic, State diagram;
2. Задаване параметрите на използваните **устройства (**Device Properties):
   - **категория** (Product Category): All, General Purpose, Military, Automotive)
   - **фамилия** чипове (Family): QProVirtex, Spartan 3E, Virtex4, CoolRunner2 и др.)
   - **чип** от избраната фамилия (Device): XC3S100E, XC4VLX60 и др.
   - **корпус** (Package) – напр. TQ144;
   - **скорост** (Speed) – -4;-5;
   - **средство за синтеза** (Synthesis Tool) – XST (VHDL/Verilog);
   - **симулатор** (Simulator): ISE Simulator, ModelSimXE.



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

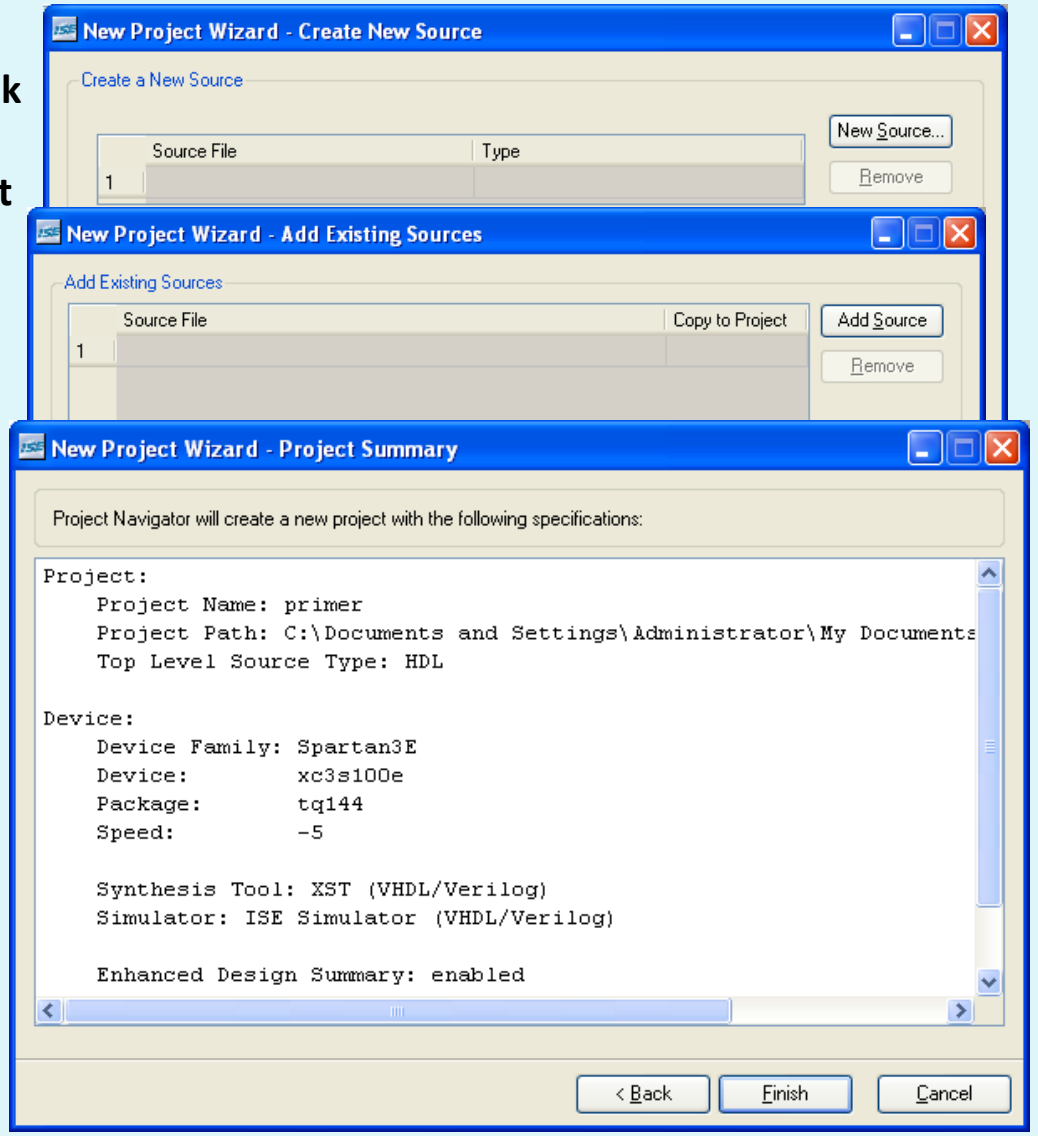
3. Задаване **сорс-файл (**файлове) към проекта (Create New Source):

   - от New Project Wizard → Create New Source : указва се името и разположението на новия сорс-файл или чрез:

   - Project → New Source

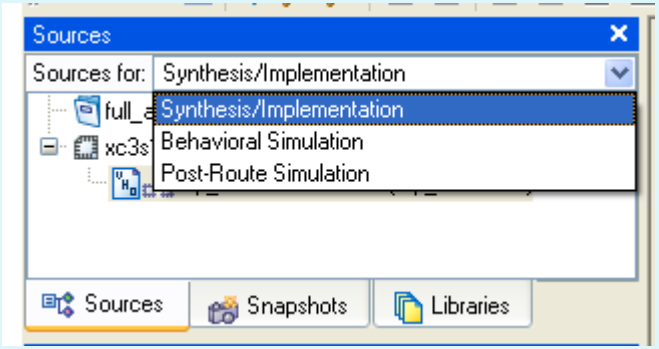
   - както и да се добави като сорс-файл съществуващ такъв.

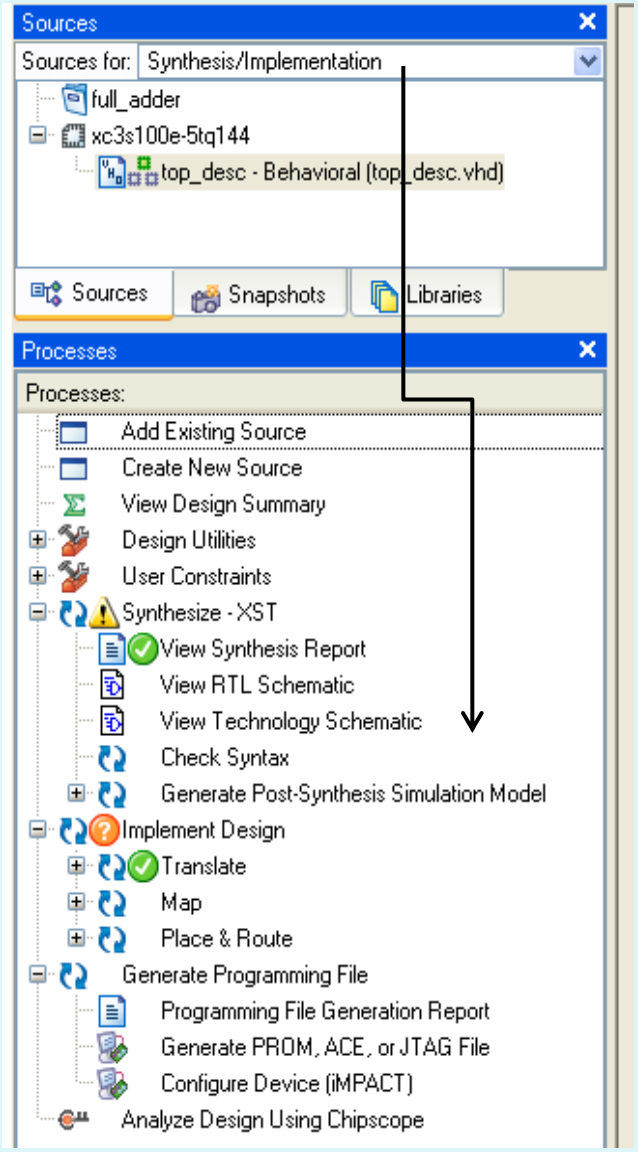
4. Обобщаване **параметрите на проекта** (Project Summary Wizard) – дават се всички зададени параметри за създавания проект с възможност за връщане назад и корекция.

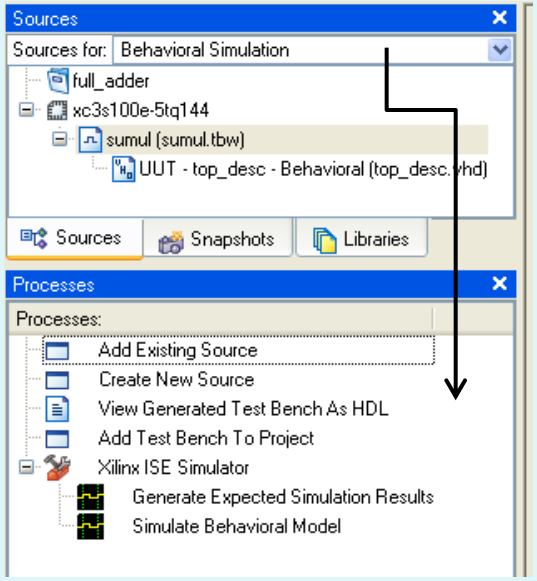


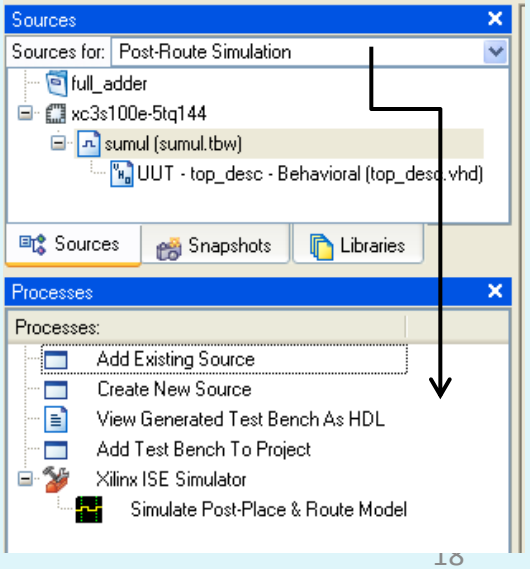
- Завършване процеса на създаване на проект – бутон **Finish**

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









<u>Структура на проекта</u>:

! **Sources** for (файлове за), <u>асоциирани към даден *тип описание*</u>:

- **Synthesis/Implementation;**
- **Behavioral Simulation;**
- **Post-Route Simulation.**

! За всеки тип описание <u>проектирането се представя като съвкупност от процеси</u>
→ **Processes**

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

Методологична схема за проектиране на ЦУ в среда ISE/WebPack:

Синтез (Synthesize - XST) →

- *описание структурата на проекта: устройството се описва с HDL език (VHDL/Verilog, .VHD), директно със схемата му (Schematic, .SCH) или по друг начин;*
- *компилация на проекта → отстраняване на синтактични грешки;*
- разглеждане на отчет (Report) по отношение синтеза на проекта, вкл. Device Utilization Summary, Timing Summary;
- визуализация на RTL Schematic като резултат от синтеза (възможност за разглеждане фрагменти на ниво ЛЕ);
- визуализация на Technology Schematic (кои конкретно ресурси от чипа ползва проекта);

Имплементация (Implement Design)→

- транслация (Translate);
- разположение/карта (Map);
- разположение и създаване на връзки (Place & Route);

Генериране на програмен (конфигурационен) файл (Generate Programming File) →

- отчет (Programming File generation Report);
- генериране PROM файл (JTAG, ACE);
- конфигуриране на устройството (Configure Device / iMPACT).

N.B. Фирмата **Digilent** използва отделно приложение за прехвърляне на проекта към FPGA чипа - **ExPort**.

*** Ограничения (User Constraints) →**

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

Методологична схема за проектиране на ЦУ в среда ISE/WebPack:

Ограничения (User Constraints) →

***** *могат да се налагат във всеки етап на изграждане на проекта :*

- *в самото начало на проектирането;*
- *непосредствено преди имплементацията.*

Отнася се до дефинирането на конкретни спецификации на ползваните входно/изходни връзки и/или времеви ограничения. Включва:

- налагане времеви ограничения (**Create Timing Constraints**);
- **Assign Package Pins** – задава се атрибутиране на сигнали към определени I/O изводи (банки);
- **Create Area constraints** – стартира отделно приложение за дефиниране на тези ограничения (присвоявания): **PACE**- Pinout Area Constraints Editor;