Obiectivele generale ale laboratorului de AC

Obiectivul principal al acestui laborator este dezvoltarea si testarea de procesoare didactice de tip MIPS folosind VHDL, Vivado Webpack si placi de dezvoltare Digilent Development Boards (DDB).
Principalele teme atinse:

- Proiectare cu uneltele Vivado Webpack si DDB.
- Componente hardware VHDL sintetizabile, implementate si testate pe DDB
- Proiectarea VHDL a uneia sau a mai multor variante de procesor MIPS (single cycle, multi cycle, pipeline)

Laborator 01

Introducere in mediul software/hardware de dezvoltare VHDL

1. Obiectiv

Familiarizarea studentilor cu

- Vivado Webpack
- Xilinx Vivado Design Suite User Guide
- Digilent Development Boards **(DDB)**
  - **Digilent Basys Board – Reference Manual**
- Artix 7 FPGA

2. Resurse necesare (Vivado este instalat deja pe stațiile de lucru)

Manual de referinta pentru placa Basys 3 (Artix 7)

- Disponibil pe site la Xilinx

Xilinx Vivado WebPACK – se găsește aici.

- Este parte a Vivado HL Design Suite
- (pt. acasă!) În procesul de instalare (fie cu installer online, fie din kit-ul complet) se selectează Vivado HL WebPACK
- Licența se obține la finalul instalării, este gratuită!
- Pentru Windows pe 64 biți e ok ultima versiune (2016.4)
- Pentru Windows pe 32 biți e ok versiunea 2014.4
- Necesită cel puțin Windows 7.

Help online pentru VHDL

http://vhdl.renerta.com/

3. Activități practice

Notă: dacă este necesar, consultați help-ul online pentru VHDL indicat la Resurse, pe pagina anterioară.

3.1. Implementați un proiect VHDL simplu în Xilinx Vivado prin parcurgerea atentă și completă a tutorialului descris in Anexa 1. În prealabil, faceți o lectură sumară a Anexei 2, pentru reamintirea componentelor digitale de bază.

3.2. Adăugați un numărător binar pe 16 biți, directional, (binary up/down simple 16-bit counter) în proiectul *test_env*, prin descrierea comportamentului numărătorului în arhitectura entității *test_env*. Încercăm controlarea de la un buton a numărătorului.

Mai întâi, declarați un semnal de 16 biți (tip STD_LOGIC_VECTOR) în arhitectură, înainte de *begin*. Dacă este necesar (până când va redobândiți capacitatea de descriere în VHDL), folosiți *Language Templates* (Anexa 1) pentru a extrage descrierea comportamentală a numărătorului.

Folosiți unul dintre butoanele din porturile entității pentru a controla ceasul numărătorului, ca un semnal de activare a ceasului (se adaugă condiția ca semnalul butonului să fie 1, acolo unde se testează apariția frontului crescător în numărător = (!) un if suplimentar, inclus in corpul if-ului care testează frontul crescător de ceas).

Folosiți unul dintre switch-uri pentru a controla direcția de numărare.

Mapați cu atribuire concurentă cei 16 biți ai numărătorului pe LED-uri.

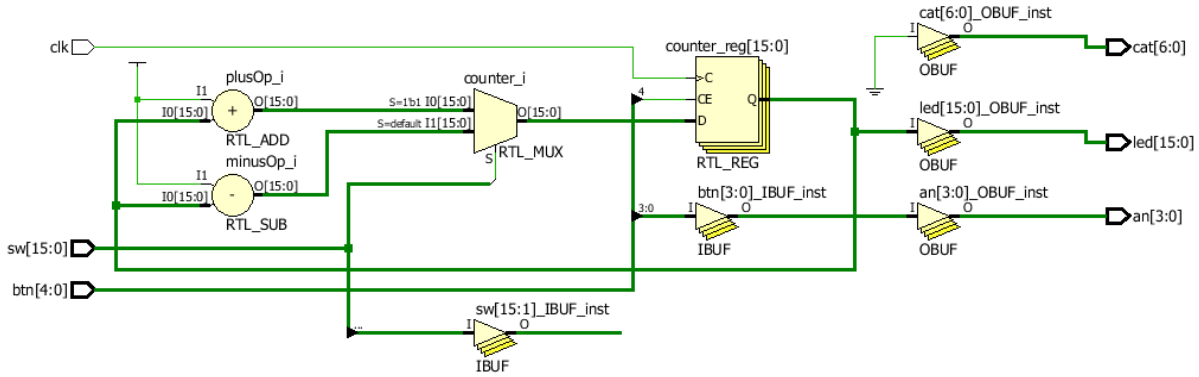


Figura 1: Schema circuitului realizat până acum, se poate vizualiza cu click pe **RTL Analysis – Elaborated Design → Schematic** în panoul **Flow Navigator**

Încărcați proiectul pe placa Basys. Controlați numărătorul de la buton.

…Ce probleme apar?

3.3. Generator de Monoimpuls Sincron (durată o perioadă de ceas) – MonoPulse Generator - MPG

La acest punct lucrați în proiectul de la punctul 3.2.

În proiectele viitoare veți avea nevoie să controlați pas cu pas circuite secvențiale, cu scopul de a trasa și testa fluxul de date și de control din circuitele implementate.

Circuitul necesar, care generează un semnal de ENABLE o singura dată la o apăsare a butonului, este prezentat mai jos.

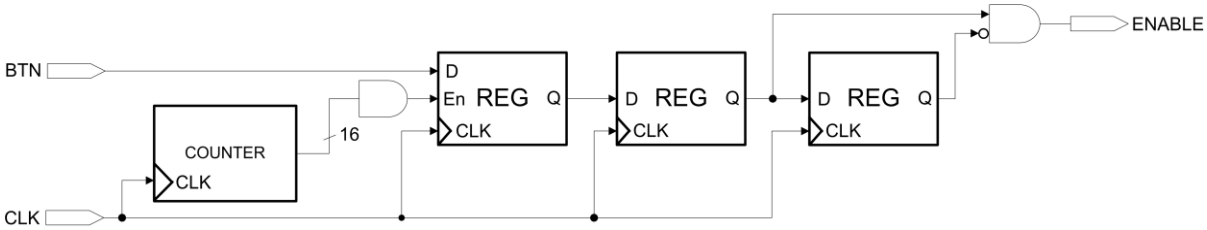


Figura 2: Generator de monoimpuls sincron

Rolul primului registru împreună cu numărătorul este de a asigura robustețe împotriva butoanelor uzate fizic, unde apar multiple activări ale semnalului ENABLE la o apăsare de buton. În funcție de uzură, este posibil să fie nevoie de mai mulți biți (17-20+) din numărător în porta AND, pentru a mări intervalul de eșantionare a butonului.

MPG-ul va fi implementat într-o entitate / fișier sursă nou (meniu **File\Add Source**) și va fi folosit în entitatea *test_env* prin declarerea cu *component* în secțiunea de declarare a semnalelor, respectiv instanțierea cu *port map* după begin în arhitectură.

Intern, componentele din diagrama MPG, regiștrii, numărător, porțile AND, se vor descrie comportamental prin declararea semnalelor necesare, respectiv a proceselor și a atribuirilor concurente în arhitectura MPG.

Pași de urmat:

a. Desenați diagrama de timp a circuitului MPG (hârtie sau la tablă).
b. Scrieți și verificați codul VHDL pentru acest circuit.
c. Includeți MPG în entitatea *test_env*. Vezi indicațiile de mai sus.
d. Folosiți ieșirea ENABLE ca semnal de activare a ceasului numărătorului de 8 biți adăugat la pasul 3.2 în *test_env* (se adaugă condiția ca ENABLE să fie 1, acolo unde se testează apariția frontului crescător de ceas în numărător).

Nu uitați de **RTL Analysis – Elaborated Design → Schematic …**

Încărcați pe placa Basys.

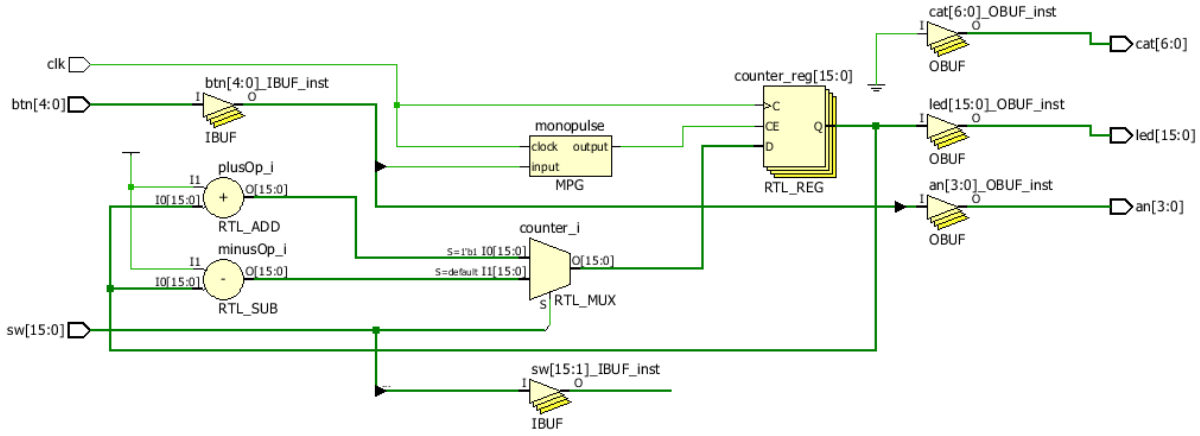
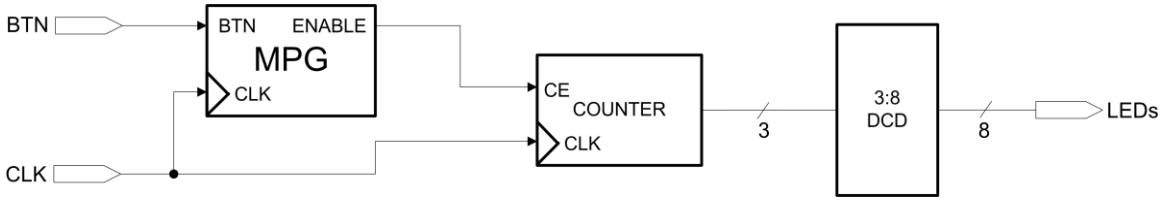


Figura 3: Schema circuitului realizat până acum, se poate vizualiza cu click pe **RTL Analysis – Elaborated Design → Schematic** în panoul **Flow Navigator**

3.4. Creați un nou proiect, de exemplu *test_new*, folosind aceleași porturi ca pentru primul proiect. Practic, trebuie sa reparcurgeți tutorialul din Anexa 1, fără a adăuga nimic în arhitectura *test_*new. Acum, implementați circuitul de mai jos în arhitectura *test_new*.



Trebuie să adăugați fișierul sursă MPG în noul proiect (**File\Add Sources** sau în panoul **Flow Navigator** la **Project Manager / Add Sources**). MPG se importă cu *component* / *port map* în entitatea *test_new*, iar restul componentelor se descriu în arhitectura *test_new* fără (!) entități adiționale. Adăugați un numărător pe 3 biți, și un decodificator 3:8 biți, folosind doar semnale declarate în arhitectura *test_new*, respectiv procese/atribuiri concurente.

Nu uitați de **RTL Analysis – Elaborated Design → Schematic …** Încărcați pe placa Basys.

Temă

1. Finalizați activitățile neterminate la laborator.
2. Recitiți cu atenție **Regulamentul Laboratorului de AC** și reparcurgeți tutorialul din Anexa 1 de creare a unui proiect nou – din laboratorul 2 aceste noțiuni se consideră învățate! Atenție la aspectele din tutorial care nu au fost relevante pentru acest prim proiect: ele vor fi necesare în viitor.
3. (*acest punct se consideră implicit pentru următoarele laboratoare)* Citiți materialul pentru laboratorul următor (va apare pe site până Vineri).

4. Referințe

- Manual de referinta pentru placa Basys 3 (Artix 7), disponibil pe site la Xilinx
- Xilinx Vivado WebPACK – aici.
- Help online pentru VHDL, http://vhdl.renerta.com/

Anexa 1

VIVADO Quick Start Tutorial, adaptat pentru versiunea 15.4

Pornirea mediului VIVADO

Dublu click pe icoana de pe desktop, sau se pornește din **Start→Programs→ Xilinx Design Tools → Vivado 15.4**

Acces la Help

Pentru a accesa Help-ul:

- Din meniul principal lansați **Help/Documentation and Tutorials** din meniul Help. Veți accesa informații diverse despre crearea și mentenanța întregului ciclu de dezvoltare în Vivado.

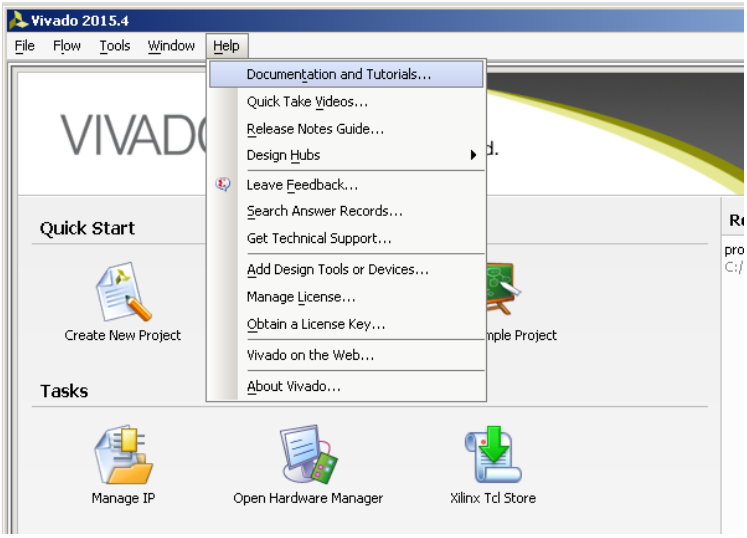


Figura 4: Vivado Help Topics

Crearea unui nou proiect

Creați un nou proiect Vivado, care va fi dedicat pentru dispozitivul FPGA prezent pe placa de dezvoltare Basys 3, FPGA Artix 7.

Pentru a crea noul proiect:

1. Selectați din meniu **File → New Project... .**Se va deschide New Project Wizard. Click **Next**.
2. Introduceți **test_env** în câmpul Name.

3. Introduceți sau accesați o locație (cale de director, amintiți-vă regulile!) pentru proiectul nou. Un subdirector test_env se va crea automat (*Create project subdirectory* trebuie sa fie bifat). Click **Next**.
4. Ați ajuns la alegerea tipului proiectului. Selectați *RTL Project*. Bifați *Do not specify sources…* le veți crea mai târziu. Click **Next**.
5. Se aleg proprietățile plăcii. Completați proprietățile conform listei de mai jos:

   - Product Category: **All**
   - Family: **Artix-7**
   - Package: **cpg236**
   - Speed Grade: **-1**
   - Temperature grade: **C**
   - În tabel alegeți varianta **xc7a35tcpg236-1**.
   - **Next**…

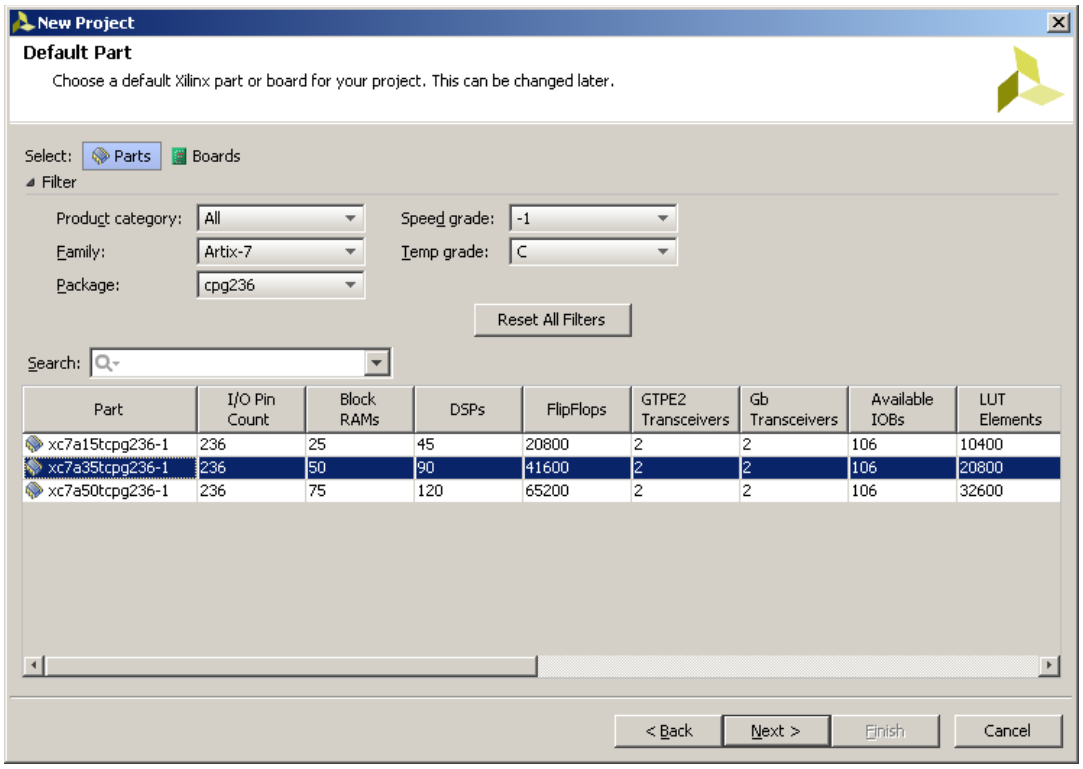


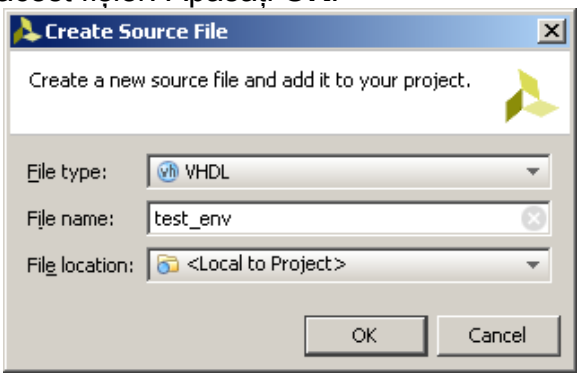
Figura 5 Alegerea proprietăților plăcii

6. Click **Finish**, se va deschide noul proiect, gol deocamdată.

Crearea unui fișier sursă VHDL

1. Click în meniu pe **File\Add Sources**. Sau: în panoul **Flow Navigator** (de obicei este situat în stânga) la **Project Manager / Add Sources**.
2. Selectați *Add or create design sources*, click **Next**.

3. Apăsați *Create file*. Pe dialogul apărut selectați la *File type*: VHDL, introduceți numele fișierului *test_env* (! Nu e obligatoriu sa fie același nume cu proiectul), lăsați *File location* neschimbat. Acest nume (*test_env*) îl va avea și entitatea creată automat în acest fișier. Apăsați **OK**.



4. Apăsați **Finish**. Se va crea noul fișier / entitate și se va deschide automat dialogul de definire a porturilor **Define Module**.
5. Acum declarați porturile pentru entitatea principală ce se va crea, completând informația pentru porturi ca în figura de mai jos. **Atenție: aceste porturi sunt definite în mod particular pentru placa Basys 3, fiind în principiu suficiente pentru majoritatea proiectelor dezvoltate pe parcursul acestui semestru.** Apăsați **OK**.

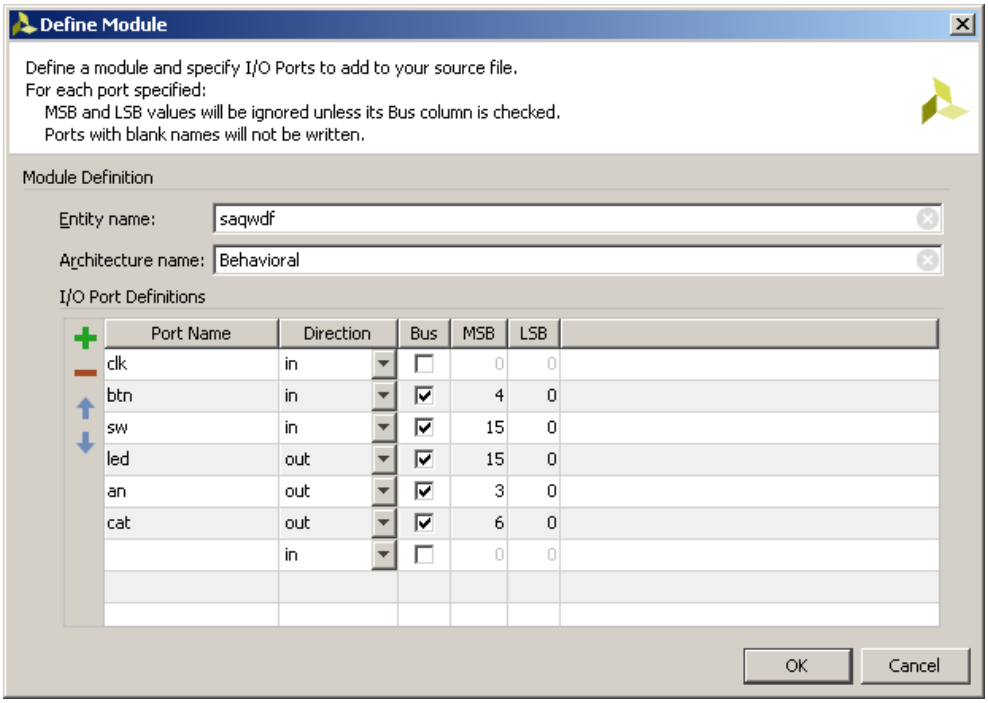


Figura 6: Definirea porturilor prin interfața Vivado

6. S-a finalizat crearea noului fișier sursă.

Observație: se poate sări peste pasul de definire a porturilor în interfață (pas 6), caz în care se pot declara (sau modifica/corecta) porturile in secțiunea de declarare a entității create în noul fișier.

Fișierul sursă care conține declararea entității *test_env* și arhitectura ei este afișat acum în mediul ISE, iar în zona **Hierarchy** (panoul **Sources**) apare ca modulul principal (Top Module) al proiectului curent. Dacă nu va apare fereastra de editare pentru fișierul nou creat, dați dublu click pe test_env în panoul **Sources**.

De reținut: în proiectele cu mai multe surse, daca se schimbă accidental entitatea Top Module, se poate seta alta ca Top Module prin click dreapta pe sursa dorită în **Hierarchy,** după care click pe **Set as Top**.

Verificare: Accesați **Project Summary**, la secțiunile **Synthesis** și **Implementation** veți vedea la câmpul *Part* numele dispozitivului (plăcii) ales. Pentru Basys 3, trebuie să fie **xc7a35tcpg236-1.** Dacă nu coincide, înseamnă ca ați sărit peste pasul 6 la **Crearea unui nou proiect.** În meniul principal, accesați **Tools/Project Settings**, la secțiunea **General**, câmpul *Project Device*, apăsați "…" și reintroduceți proprietățile corecte ale plăcii.

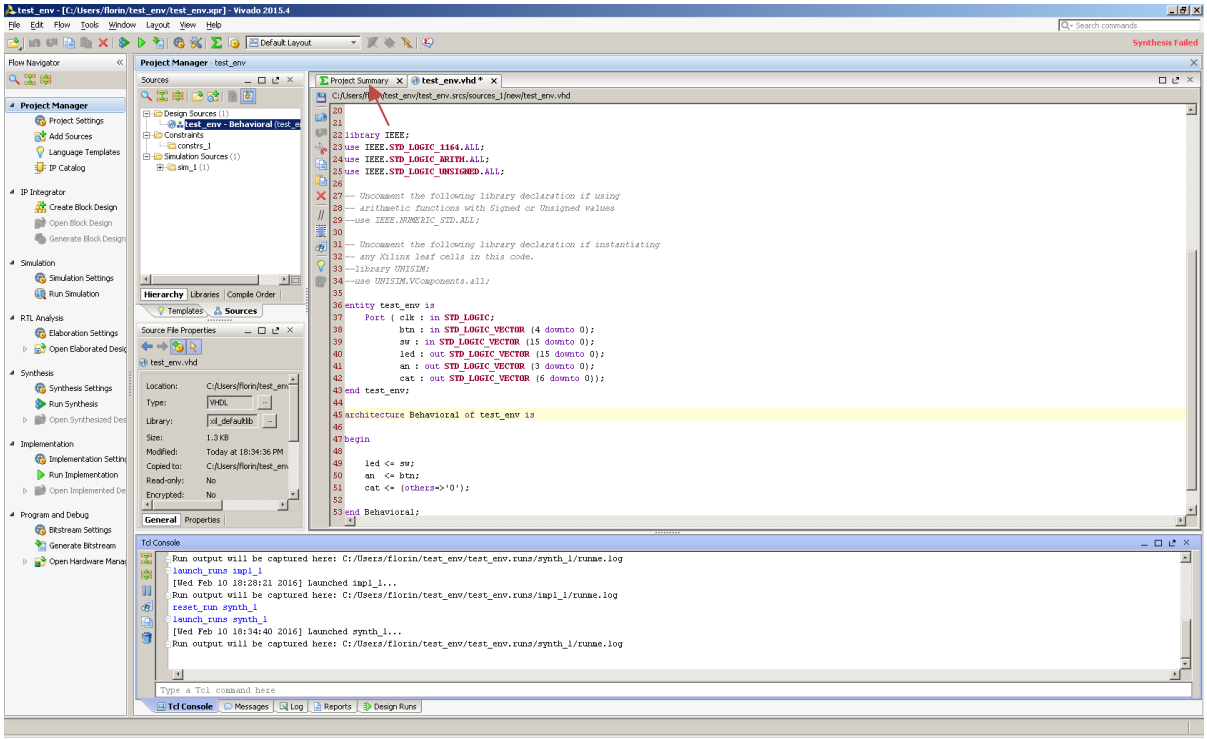


Figura 7: Noul proiect în Vivado. **Project Summary** e indicat de săgeata roșie.

În fereastra de editare, asigurați-vă că următoarele librării sunt incluse în zona de declarare a librăriilor în *test_env*:

```
use IEEE.STD_LOGIC_ARITH.ALL;
```

use IEEE.STD_LOGIC_UNSIGNED.ALL;

Dacă nu sunt incluse, adăugați-le. Valabil pentru orice proiect (sau fișier sursă) viitor care se va crea de la zero.

(*De revenit aici când se va cere…*) Folosirea Language Templates (VHDL) – optional

Următorul pas în procesul de creare a noii surse VHDL este adăugarea descrierii comportamentale pentru entitatea Top Level a proiectului. O modalitatea facilă (pentru început, până vă reamintiți limbajul VHDL) este să folosiți exemple de cod din Vivado Language Templates pe care să le particularizați pentru entitatea voastră.

1. Plasați cursorul de editare sub begin în secțiunea architecture a entității unde doriți să adăugați codul.
2. Deschideți Language Templates prin selectarea din meniu a **Window → Language Templates…**
3. Navigați în ierarhie prin simbolul "+", către exemplele de cod:
   VHDL → Synthesis Constructs → Coding Examples → …
4. Selectați componenta dorită în ierarhie, copiați codul și inserați-l în destinație.
5. Închideți **Language Templates**.
6. Înlocuiți denumirea implicită a semnalelor din codul inserat cu denumirea semnalelor din entitatea pe care o descrieți.

Editarea finală / sintetizarea sursei VHDL (descrierea comportamentului)

1. Adăugați componente (cu *component*) și / sau declarații de semnal în secțiunea de declarații între architecture și begin.
2. Adăugați restul codului (instanțierea componentelor – *port map*, descrierea comportamentală – *process* sau atribuiri concurente, etc.) între begin și end.
3. Pentru acest prim proiect adăugați următoarele atribuiri concurente după begin:

```
led <= sw;
an  <= btn (3 downto 0);
cat <= (others=>'0');
```

4. Salvați fișierul cu **File → Save File or Ctrl + S**.
5. În zona *Hierachy (panoul Sources)* selectați entitatea Top Level, în cazul de față *test_env*.
6. Vizualizați circuitul rezultat sub formă schematică (relevant mai ales pentru următoarele proiecte, unde vor fi circuite mai complexe): în panoul **Flow Navigator** click pe **RTL Analysis – Elaborated Design → Schematic.** Se va deschide o schemă bloc a circuitului principal, dublu click pe diverse componente pentru a vedea organizarea internă. **Ar trebui să recunoașteți cel puțin o parte a componentelor declarate!** Aceasta este o primă metodă de verificare ca ați descris și legat corect componentele dorite.

7. Corectați eventualele erori care sunt raportate în zona **Console / Messages** (partea de jos a Vivado). Începeți procesul de corectură cu prima eroare!
8. Sintetizați proiectul: în zona Flow Navigator (stânga) click pe **Run Synthesize.** Dacă sunt erori nu mergeți mai departe cu pasul următor (**Implementation**).
9. Corectați eventualele erori care sunt raportate în zona **Console / Messages** (partea de jos a Vivado). Începeți procesul de corectură cu prima eroare!

Acum ați finalizat procesul de creare a sursei VHDL, fără erori de sintaxă!

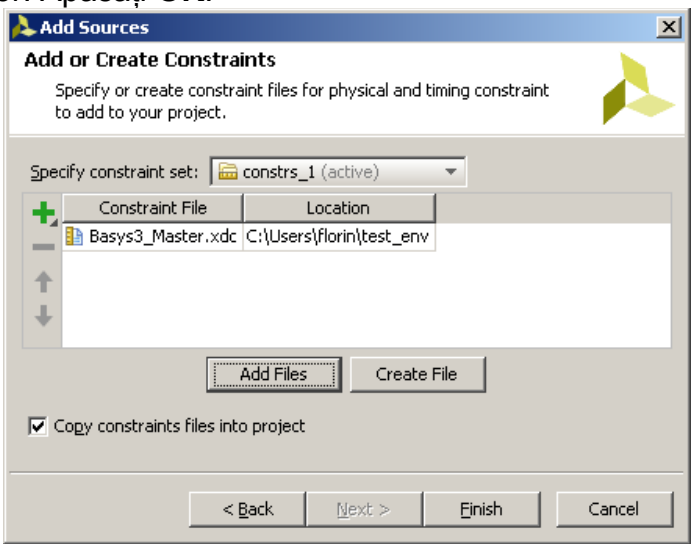
Imlementarea proiectului

7. Implementați proiectul: în meniu **Flow / Run Implementation** sau în panoul **Flow Navigator** la **Implementation / Run Implementation**.
1. Corectați eventualele erori și avertismente (atenție: o parte dintre avertismente nu necesită corectură, dacă sunt irelevante: ex. anumite semnale sunt declarate dar nu sunt legate încă în proiect).

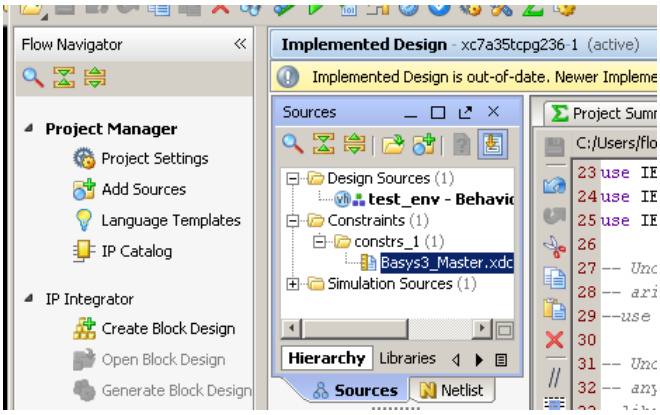
Stabilirea constrângerilor pentru locațiile de pini (atribuirea pinilor)

Specificați care locații de pini de pe placa de dezvoltare vor fi atribuiți porturilor din proiect (entitatea Top Level *test_env*). Există mai multe metode, mai jos aveți abordarea recomandată (rapidă, fără probleme de compatibilitate):

1. În particular, pentru porturile definite în acest tutorial, găsiți constrângerile definite aici. Descărcați fișierul în directorul personal sau al proiectului.
2. Click în meniu pe **File\Add Sources**. Sau: în panoul **Flow Navigator** la **Project Manager / Add Sources**.
3. Selectați *Add or create constraints*, click **Next**.
4. Apăsați *Add file*. Pe dialogul apărut selectați fișierul de constrângeri descărcat la pasul anterior. Apăsați **OK**.



5. Bifați *Copy constraints files into project* și apăsați **Finish**. Fișierul de constrângeri este adăugat la proiect și vizibil în panoul **Sources** la **Constraints**.



6. Click dreapta pe fișierul de constrângeri din zona **Sources** și apăsați *Set as Target Constraint File* pentru al selecta ca fiind asociat cu dispozitivul (placa) de testare.
7. Pentru a edita fișierul de constrângeri dați dublu-click pe el în zona **Sources**. O constrângere uzuală (între un port din entitatea principală și o locație de pin de pe placă) este definită prin două linii de sintaxă. Exemplu pentru butonul din centru:

```
set_property PACKAGE_PIN U18 [get_ports {btn[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {btn[0]}]
```

   unde U18 reprezintă numele pinului de pe placă asociat cu butonul central, iar btn[0] este portul din entitate cu care vrem sa asociem butonul central.

Pentru proiecte viitoare, dacă este necesară adăugarea de noi constrângeri la alți pini, pentru fiecare pin se scriu cele două linii în fișierul de constrângeri. Numele pinului (locației) de pe placă se poate găsi în manualul de referință Basys 3, sau pe placă, lângă locația dorită, în paranteze.

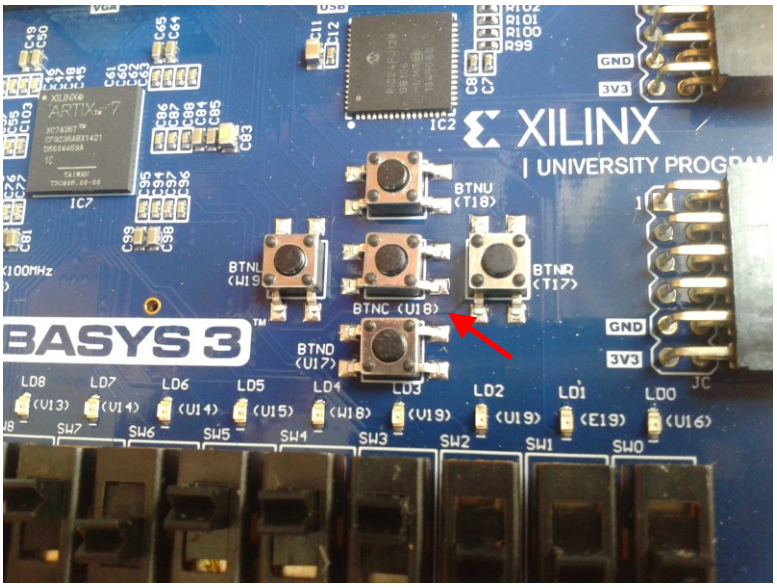


Figure 8 Numele locațiilor de pini imprimate pe placă. Cu săgeată roșie: pentru butonul central locația de pin este U18.

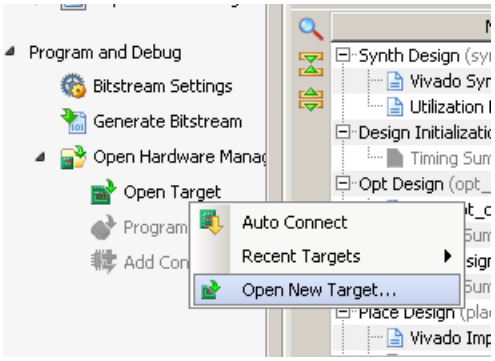
Generarea fișierului de programare (*.bit)

În panoul **Flow Navigator / Program and debug** sau din meniul **Flow** se apasă **Generate Bitstream**. Se va crea fișierul *.bit care se va încărca pe placă.

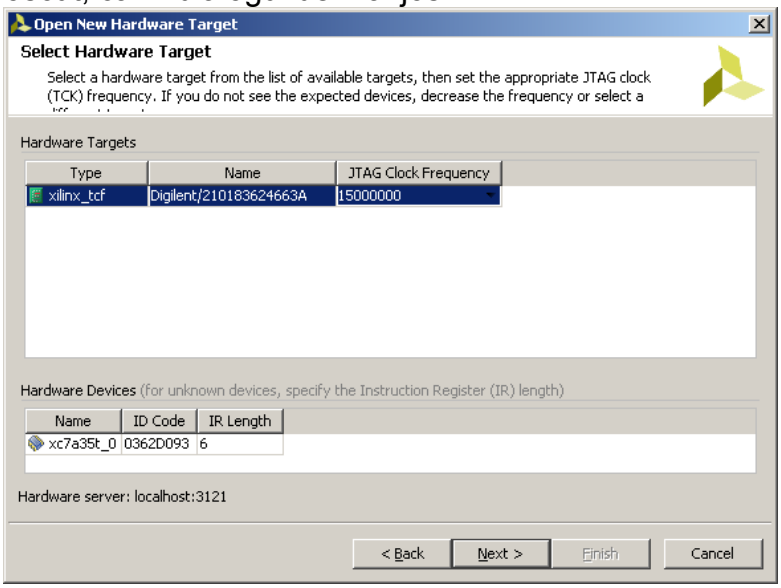
În acest moment, dacă nu există erori, fișierul test_env.bit a apărut în structura de directoare ale proiectului. Altfel, vedeți erorile în zona de consolă și corectați-le. Erorile din acest pas sunt de obicei din cauză că ați sărit peste pasul 6 a **Crearea unui nou proiect.** În meniul principal, accesați **Tools/Project Settings**, la secțiunea **General**, câmpul *Project Device*, apăsați "…" și reintroduceți proprietățile corecte ale plăcii.

Încărcarea proiectului (*.bit) pe placa de dezvoltare Basys 3

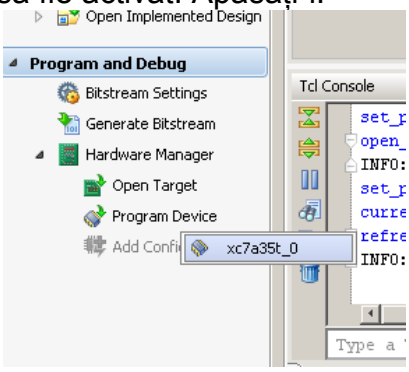
1. Conectați placa la portul USB (nu forțați portul de microUSB, asigurați-vă ca îl introduceți în poziția corectă). Porniți placa de la switch-ul de lângă portul de microUSB.
2. In panoul **Flow Navigator**, secțiunea **Program and Debug**, dați click pe **Open Target**, iar în meniul deschis alegeți **Open New Target**.



3. **Next**. Lăsați *Local server* la câmpul Connect to. **Next**. Ar trebui ca dispozitivul sa fie recunoscut, ca în dialogul de mai jos.



4. **Next. Finish.**
5. Dacă placa a fost recunoscută, ar trebui ca **Program Device (Flow Manager / Program and Debug)** să fie activat. Apăsați-l.



6. Va apare un meniu pop-up cu lista de dispozitive conectate, unul singur in cazul de față (Basys 3 – xc7a35t_0). Apăsați-l.
7. In dialogul deschis trebuie sa selectați calea spre fișierul *.bit cu programarea. Acest fișier se găsește de obicei în directorul proiectului, în calea *nume_proiect*.runs/impl_1/. Pentru proiectul test_env: test_env.runs/impl_1/test_env.bit. Click **Program**.

8. În acest moment proiectul este încărcat pe placă. Experimentați (switch-uri, butoane, leduri)!
9. Pentru re-programare, dacă nu e activ **Program Device**, se poate apăsa **Open Target / Auto Connect**.

Probleme legate de programare și eventuale soluții

Problemă

Placa Basys 3 nu este recunoscută

a) Încercați alt port USB (față sau spate), dacă începe automat un proces de instalare a driver-ului și vă cere drepturi de administrator, cereți ajutorul profesorului.
b) Încercați cu alt cablu.
c) Încercați altă placă (raportați asta profesorului).
d) Reporniți mediul Vivado / Calculatorul.
e) Schimbați stația de lucru.

Problemă

Mediul Vivado se oprește după încărcarea pe placă.

a) Reporniți mediul Vivado.
b) Reporniți stația de lucru.

Anexa 2 – Componente de bază

1.1. Logical gates

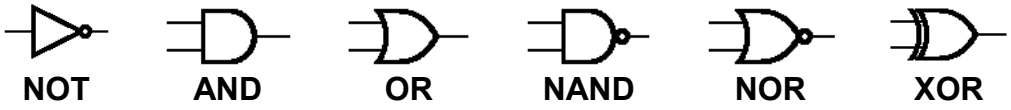


Figure 9: Logical gates

A	NOT
0	1
1	0

A	B	AND	OR	NAND	NOR	XOR
0	0	0	0	1	1	0
0	1	0	1	1	0	1
1	0	0	1	1	0	1
1	1	1	1	0	0	0

Tabel 1: Truth tables

1.2. Latches

A latch is an electronic circuit which has two stable states and thereby can store one bit of information. XST can recognize latches with asynchronous set/reset control signals. Latches can be described in VHDL by using: processes or concurrent statement assignment. XST does not support wait statements (VHDL) for latch descriptions.

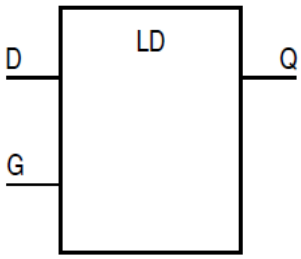


Figure 10: Latch with Positive Gate

IO Pins	Description
D	Data Input
G	Positive Gate
Q	Data Output

Table 2: Latch with Positive Gate Pin Description

1.3. Flip-Flops

A flip-flop is an electronic circuit that has two stable states and is capable of serving as one bit of memory. A flip-flop is usually controlled by one or two control signals and/or a gate or clock signal. XST recognizes flip-flops with the following control signals: asynchronous Set/Reset, synchronous Set/Reset or clock enable.

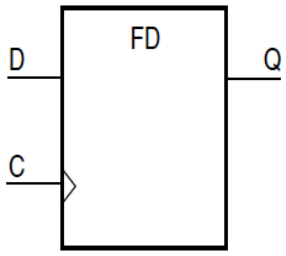


Figure 11: Flip-flop with Positive Edge Clock

IO Pins	Description
D	Data Input
C	Positive Edge Clock
Q	Data Output

Table 3: Flip-Flop with Positive-Edge Clock Pin Descriptions

When using VHDL for a positive-edge clock, instead of using:

```
if (C'event and C='1') then
```

you can also use:

```
if (rising_edge(C)) then
```

1.4. Multiplexers

A multiplexer or mux is a device that performs multiplexing; it selects one of many analog or digital input signals and outputs that into a single line. A multiplexer of 2^n inputs has n select bits, which are used to select which input line to send to the output. XST supports different description styles for multiplexers (MUXs), such as If-Then-Else or Case.

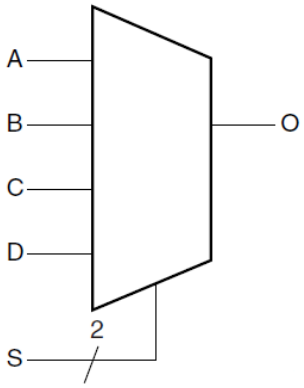


Figure 12: 4-to-1 1-Bit MUX

IO Pins	Description
A, B, C, D	Data Inputs
S	Mux Selector
O	Data Output

Table 4: 4-to-1 1-Bit MUX Pin Descriptions

1.5. Decoders

In digital electronics a decoder is a multiple-input, multiple-output logic circuit that converts coded inputs into coded outputs, where the input and output codes are different. E.g.: n-to-2^n decoders, BCD decoders.

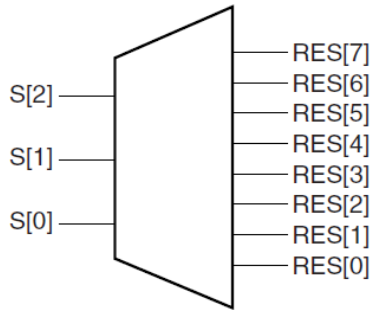


Figure 13: 3-of-8 Decoder

IO Pins	Description
S	Selector
RES	Data Output

Table 5: 3-of-8 Decoder Pin Descriptions

1.6. Counters

In digital logic and computing, a counter is a device which the number of times a particular event or process has occurred, often in relationship to a clock signal. XST recognizes counters with the following control signals: asynchronous Set/Reset, synchronous Set/Reset, asynchronous/synchronous Load (signal or constant or both), clock enable, modes (up, down, up/down) or a mixture of all.

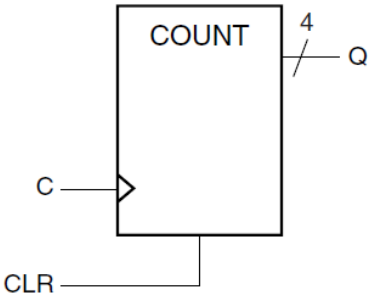


Figure 14: 4-Bit Up Counter with Asynchronous Reset

IO Pins	Description
C	Positive Edge Clock
CLR	Asynchronous Reset (Active High)
Q	Data Output

Table 6: 4-Bit Up Counter with Asynchronous Reset Pin Descriptions